# QWeak GEANT4 Code Manual

Klaus Grimm

and

Michael Gericke

August 10, 2007

# Contents

# Chapter 1

# Introduction

The QWeak simulation package *QweakSimG4* is a collection of C++ *classes* that can be compiled and run for the purpose of physics process simulation in the experiment. The code aims to eventually include all objects and properties found in the current design of the QWeak experiment, as well as most physics processes that are expected to emerge during the experiment. The implementation of these objects and properties is at various stages of development, the detail of which is indicated in the extent of the list of contents. The code is based on the GEANT4[1] simulation code [1] and the ROOT[2] analysis package and requires these to compile.

The purpose of this manual is to describe the classes and their routines and how they are used in the simulation. The manual is supposed to enable other people to extend, repair or otherwise change the QWeak simulation program to suit their purposes. Most importantly, it is meant to document the algorithms used in implementing the physical processes in the simulation, in as far as they go beyond what is implemented in GEANT4 itself and adequately described in its manual. The detail with which the code is described will be evolving over time. Please make sure that you consult the most recent version before contacting us with questions. If you cannot find the answer to your problem in here, you can contact us at `grimm@jlab.org or mgericke@jlab.org`.

A few points to consider before using this manual are that, (1) we do not (can not) spent much time explaining any details regarding the usage of the ROOT or GEANT4 packages and its class libraries. So you should be or become acquainted with (most) aspects of the packages before trying to make any serious changes. You can find details about ROOT and GEANT4 from their websites and the ROOT manual itself [2]. Also, (2) this library is intended to work for *Linux* only. We have made no attempt to make this run on Windows or Mac.

[1] `http://wwwasd.web.cern.ch/wwwasd/geant4/geant4.html`

[2] Copyright ©1995-2004 Rene Brun & Fons Rademakers. All rights reserved. `http://root.cern.ch`

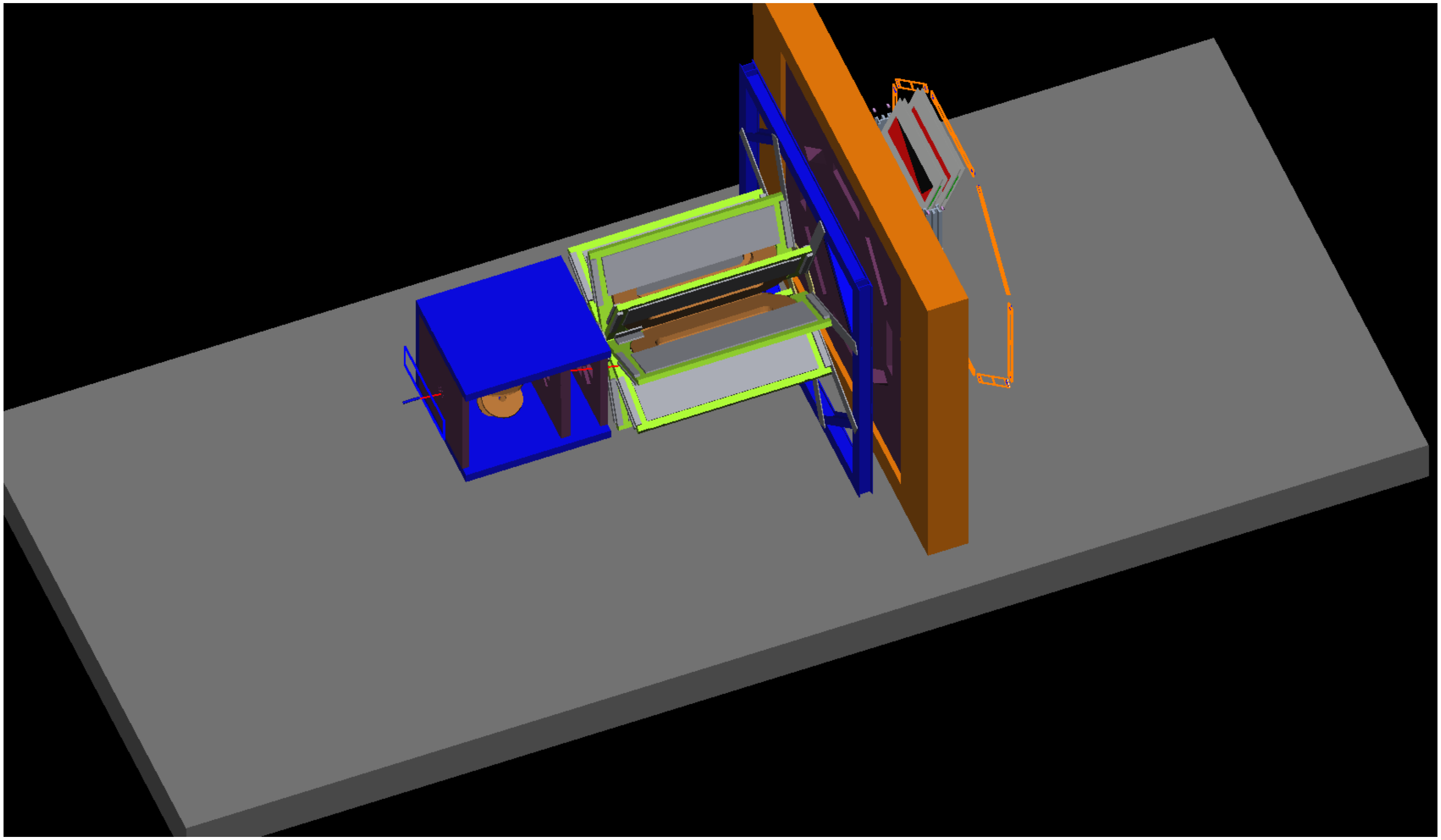

Figure 1.1: QWeak Geant4 Simulation Setup

This manual is organized as follows:

The first chapter consists of very general information, such as a list of class names and their current status or level of usage in the simulation, the compile requirements such as the structure of the *Makefile* and where the source files have to be located with respect to each other, environment variable definitions, etc.. In other words, this portion of the document should be enough to compile the simulation as it stands, provided that both ROOT and GEANT4 have been installed properly. The second chapter provides a detailed description of the event generator used in the simulations and attempts to give a detailed explanation of the physics processes included. The following chapters (one for each class) provide a detailed reference explaining what the classes do, how they do it and how to use the routines to access the class members, change variables, etc.. This portion is necessary for someone who actually wants to change the code. The details about the implementation of the physics processes used in the simulation are given together with its code in the appropriate chapters.

# Chapter 2

# Qweak Simulation General Structure and Compile Information

## 2.1 Class Structure

Tables 2.1 through 2.3 show the list of C++ source and header files currently included in the simulation. Every class or object in the program **always** has a header and source file in which the class declaration and implementation are separated from each other. This enhances readability and organization.

Not all of theses classes are currently used in the *QweakSimG4* program. The dictionary files (xxxDict.h or xxxDict.cc) are automatically generated by ROOT (see [2]) and should not be edited. The classes *QweakSimMagnetFieldMapMessenger*, *QweakSimMiniMagnetMessenger*, *QweakSimSteppingVerbose*, and *QweakSimTrajectory* are available but currently not used. Most of the *QweakSimG4* classes are derived, i.e. inherit some of their properties, from a small number of GEANT4 classes and some of the base classes are ROOT extensions.

| **Header** | **Source** |
|---|---|
| QweakSimAnalysis.hh | QweakSimAnalysis.cc |
| QweakSimCerenkovDetector.hh | QweakSimCerenkovDetector.cc |
| QweakSimCerenkov_DetectorHit.hh | QweakSimCerenkov_DetectorHit.cc |
| QweakSimCerenkovDetectorMessenger.hh | QweakSimCerenkovDetectorMessenger.cc |
| QweakSimCerenkovDetector_PMTHit.hh | QweakSimCerenkovDetector_PMTHit.cc |
| QweakSimCerenkovDetector_PMTSD.hh | QweakSimCerenkovDetector_PMTSD.cc |
| QweakSimCerenkov_DetectorSD.hh | QweakSimCerenkov_DetectorSD.cc |
| QweakSimCollimator.hh | QweakSimCollimator.cc |
| QweakSimCollimatorSupport.hh | QweakSimCollimatorSupport.cc |
| QweakSimDetectorConstruction.hh | QweakSimDetectorConstruction.cc |
| QweakSimDetectorMessenger.hh | QweakSimDetectorMessenger.cc |
| QweakSimEventAction.hh | QweakSimEventAction.cc |
| QweakSimG3Ntuple.hh | QweakSimG3Ntuple.cc |
| QweakSimG3NtupleReader.hh | QweakSimG3NtupleReader.cc |
| QweakSimGEM.hh | QweakSimGEM.hh |
| QweakSimGEMMessenger.hh | QweakSimGEMMessenger.hh |
| QweakSimGEM_WirePlaneHit.hh | QweakSimGEM_WirePlaneHit.hh |
| QweakSimGEM_WirePlaneSD.hh | QweakSimGEM_WirePlaneSD.hh |
| QweakSimGlobalMagnetField.hh | QweakSimGlobalMagnetField.cc |
| QweakSimHDC.hh | QweakSimHDC.cc |
| QweakSimHDCMessenger.hh | QweakSimHDCMessenger.cc |
| QweakSimHDC_WirePlaneHit.hh | QweakSimHDC_WirePlaneHit.cc |
| QweakSimHDC_WirePlaneSD.hh | QweakSimHDC_WirePlaneSD.cc |
| QweakSimInputRootFile_EventReader.hh | QweakSimInputRootFile_EventReader.cc |
| QweakSimMagnet_CoilParameterisation.hh | QweakSimMagnet_CoilParameterisation.cc |
| QweakSimMagnetFieldMap.hh | QweakSimMagnetFieldMap.cc |
| QweakSimMagnetFieldMapMessenger.hh | QweakSimMagnetFieldMapMessenger.cc |

Table 2.1: List of classes

| Header | Source |
| --- | --- |
| QweakSimMagnet_MiniTorusCoilParameterisation.hh | QweakSimMagnet_MiniTorusCoilParameterisation.cc |
| QweakSimMainMagnet.hh | QweakSimMainMagnet.cc |
| QweakSimMaterial.hh | QweakSimMaterial.cc |
| QweakSimMiniMagnet.hh | QweakSimMiniMagnet.cc |
| QweakSimMiniMagnetMessenger.hh | QweakSimMiniMagnetMessenger.cc |
| QweakSimPhysicsList.hh | QweakSimPhysicsList.cc |
| QweakSimPhysicsListMessenger.hh | QweakSimPhysicsListMessenger.cc |
| QweakSimPrimaryGeneratorAction.hh | QweakSimPrimaryGeneratorAction.cc |
| QweakSimPrimaryGeneratorActionMessenger.hh | QweakSimPrimaryGeneratorActionMessenger.cc |
| QweakSimRunAction.hh | QweakSimRunAction.cc |
| QweakSimShieldingWall.hh | QweakSimShieldingWall.cc |
| QweakSimShieldingWallMessenger.hh | QweakSimShieldingWallMessenger.cc |
| QweakSimStackingAction.hh | QweakSimStackingAction.cc |
| QweakSimSteppingAction.hh | QweakSimSteppingAction.cc |
| QweakSimSteppingVerbose.hh | QweakSimSteppingVerbose.cc |
| QweakSimTarget.hh | QweakSimTarget.cc |
| QweakSimTargetMessenger.hh | QweakSimTargetMessenger.cc |
| QweakSimTrackHistory.hh | QweakSimTrackHistory.cc |
| QweakSimTrackInformation.hh | QweakSimTrackInformation.cc |
| QweakSimTrackingAction.hh | QweakSimTrackingAction.cc |
| QweakSimTrackingActionMessenger.hh | QweakSimTrackingActionMessenger.cc |
| QweakSimTrajectory.hh | QweakSimTrajectory.cc |
| QweakSimTriggerScintillator.hh | QweakSimTriggerScintillator.cc |
| QweakSimTriggerScintillator_DetectorHit.hh | QweakSimTriggerScintillator_DetectorHit.cc |
| QweakSimTriggerScintillator_DetectorSD.hh | QweakSimTriggerScintillator_DetectorSD.cc |
| QweakSimTriggerScintillatorMessenger.hh | QweakSimTriggerScintillatorMessenger.cc |
| QweakSimUserCerenkov_DetectorEvent.hh | QweakSimUserCerenkov_DetectorEvent.cc |

Table 2.2: List of classes

| **Header** | **Source** |
|---|---|
| QweakSimUserCerenkov_MainEvent.hh | QweakSimUserCerenkov_MainEvent.cc |
| QweakSimUserCerenkov_OctantEvent.hh | QweakSimUserCerenkov_OctantEvent.cc |
| QweakSimUserCerenkov_PMTEvent.hh | QweakSimUserCerenkov_PMTEvent.cc |
| QweakSimUserGEM_MainEvent.hh | QweakSimUserGEM_MainEvent.cc |
| QweakSimUserGEM_SingleGEMEvent.hh | QweakSimUserGEM_SingleGEMEvent.cc |
| QweakSimUserGEM_WirePlaneEvent.hh | QweakSimUserGEM_WirePlaneEvent.cc |
| QweakSimUserHDC_MainEvent.hh | QweakSimUserHDC_MainEvent.cc |
| QweakSimUserHDC_SingleHDCEvent.hh | QweakSimUserHDC_SingleHDCEvent.cc |
| QweakSimUserHDC_WirePlaneEvent.hh | QweakSimUserHDC_WirePlaneEvent.cc |
| QweakSimUserInformation.hh | QweakSimUserInformation.cc |
| QweakSimUserMainEvent.hh | QweakSimUserMainEvent.cc |
| QweakSimUserPrimaryEvent.hh | QweakSimUserPrimaryEvent.cc |
| QweakSimUserTriggerScintillator_DetectorEvent.hh | QweakSimUserTriggerScintillator_DetectorEvent.cc |
| QweakSimUserTriggerScintillator_MainEvent.hh | QweakSimUserTriggerScintillator_MainEvent.cc |
| QweakSimUserVDC_Config.hh | QweakSimUserVDC_Config.cc |
| QweakSimUserVDC_DriftCellEvent.hh | QweakSimUserVDC_DriftCellEvent.cc |
| QweakSimUserVDC_MainEvent.hh | QweakSimUserVDC_MainEvent.cc |
| QweakSimUserVDC_SingleVDCEvent.hh | QweakSimUserVDC_SingleVDCEvent.cc |
| QweakSimUserVDC_WirePlaneEvent.hh | QweakSimUserVDC_WirePlaneEvent.cc |
| QweakSimVDC.hh | QweakSimVDC.cc |
| QweakSimVDC_DriftCellBackSD.hh | QweakSimVDC_DriftCellBackSD.cc |
| QweakSimVDC_DriftCellFrontSD.hh | QweakSimVDC_DriftCellFrontSD.cc |
| QweakSimVDC_DriftCellHit.hh | QweakSimVDC_DriftCellHit.cc |
| QweakSimVDC_DriftCellParameterisation.hh | QweakSimVDC_DriftCellParameterisation.cc |
| QweakSimVDCMessenger.hh | QweakSimVDCMessenger.cc |
| QweakSimVDCRotator.hh | QweakSimVDCRotator.cc |
| QweakSimVDC_WirePlaneHit.hh | QweakSimVDC_WirePlaneHit.cc |
| QweakSimVDC_WirePlaneSD.hh | QweakSimVDC_WirePlaneSD.cc |

Table 2.3: List of classes

Figure 2.1 and 2.2 show the inheritance structure for the simulation classes. Please refer to the GEANT4 manual on the properties of the base classes. Figure 2.1 only shows the base classes, some of which are turned into ROOT objects. All of the ROOT-object *QweakSimG4* base classes (shown in yellow) are event counters which are stored in a ROOT *tree* object after the simulation has completed and must therfore be declared as ROOT objects.

Figure **??** shows the interdependence of the classes. That is, one ore more instances of a class object is used in some other class. This is different from class interdependence due to inheritance. Each class in the library has a reasonably well defined purpose and the class names are supposed to be indicative of this purpose. The details for each class are described in chapter 4, including their overall purpose and an explanation of their member function.

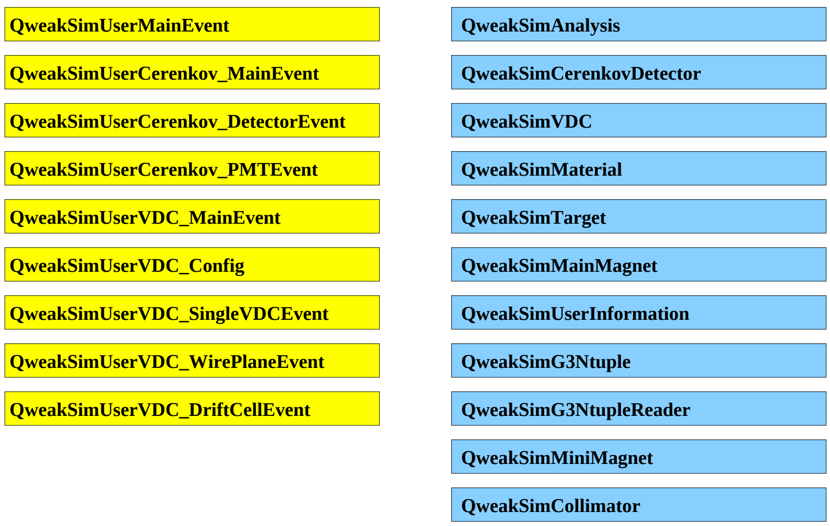


Figure 2.1: Base classes within the *QweakSimG4* hierarchy. The "yellow" boxes indicate base classes that are turned into ROOT objects. The "blue" boxes indicate regular non-ROOT base classes.

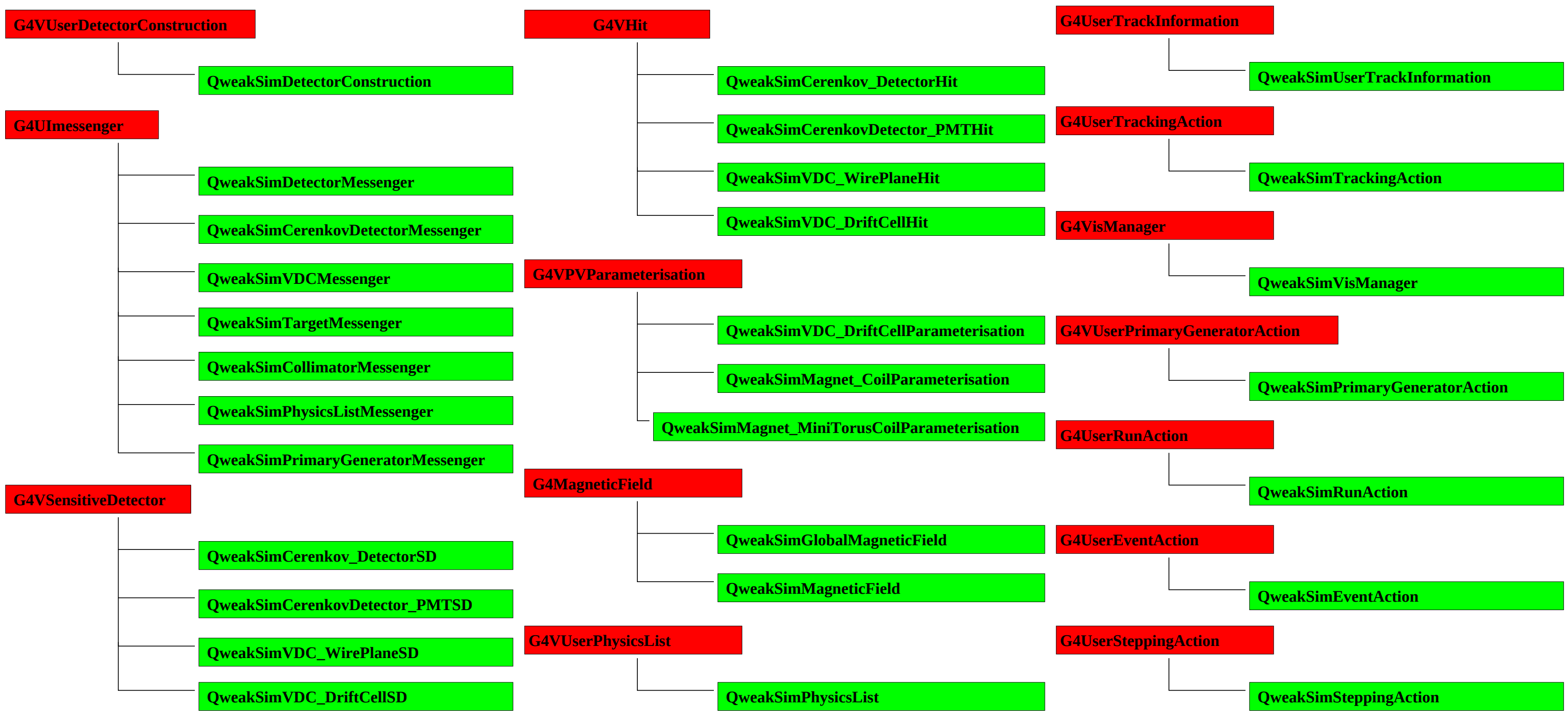


Figure 2.2: Class hierarchy within the *QweakSimG4* program. The “red” boxes indicate GEANT4 classes.

## 2.2 Preliminaries: Installing Geant4 with Coin3D and ROOT

This simulation package requires a LINUX operating system (MAC should work, Windows only via Cygwin) with the ROOT libraries and programs installed. In addition, you need the OpenGL (or mesa) libraries and source code (including the development libraries). Your search path should contain the directory with the correct ROOT installation.

For example, for the *bash* shell you should have the following statements in your *.bashrc* file:

```
export ROOTSYS=/usr/local/root
export PATH=$ROOTSYS/bin:$PATH
```

You should also have defined the correct library path, as in:

```
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
```

But if ROOT is currently running on your system, these should be already defined somwhere.

Also, the successful installation of Geant4 itself will have required that the following switches are in your search path (this is for LINUX only), with the directories changed accordingly:

```
#######################################
#
# ROOT
#
#+

export ROOTSYS=/usr/local/root
export PATH=$ROOTSYS/bin:$PATH
export PATH=/usr/local/root/hbook:$PATH
export LD_LIBRARY_PATH=$ROOTSYS/lib:$LD_LIBRARY_PATH
export OPENGL=/usr
export MYSQL=/usr

#######################################
#
# OpenInventor and Coin3D
#
#+

export COIN=/usr/local/Coin3D
```

```
export PATH=$COIN/bin:$PATH
export OIVHOME=$COIN
export SOXT=/usr/local/SoXt
export OIVFLAGS="-I$COIN/include -DINVENTOR2_1 -I$SOXT/include"
export OIVLIBS="-L$COIN/lib -lCoin -L$SOXT/lib -lSoXt"
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$COIN/lib:$SOXT/lib
export COIN_FULL_INDIRECT_RENDERING=1


######################################
#
# GEANT4
#
#+

export G4SYSTEM="Linux-g++"
export G4INSTALL=/usr/local/geant4.8.2
export G4INCLUDE=/usr/local/geant4.8.2/include
export G4LIB=/usr/local/geant4.8.2/lib
export G4LEVELGAMMADATA=/usr/local/geant4.8.2/data/PhotonEvaporation2.0
export G4RADIOACTIVEDATA=/usr/local/geant4.8.2/data/RadiativeDecay3.1
export G4LEDATA=/usr/local/geant4.8.2/data/G4EMLOW4.2
export NeutronHPCrossSections=/usr/local/geant4.8.2/data/G4NDL3.10
#export G4ELASTICDATA=/usr/local/geant4.8.2/data/G4ELASTIC1.1
export CLHEP_BASE_DIR=/usr/local
export CLHEP_INCLUDE_DIR=/usr/local/include
export CLHEP_LIB_DIR=/usr/local/lib
export CLHEP_LIB=CLHEP
export G4LISTS_BASE=$G4LIB

export G4VIS_BUILD_OIX_DRIVER=1
export G4VIS_USE_OIX=1

export G4VIS_BUILD_DAWN_DRIVER=1
export G4VIS_USE_DAWN=1

export G4VIS_BUILD_OPENGLX_DRIVER=1
export G4VIS_USE_OPENGLX=1
export OGLHOME=/usr
export OGLFLAGS="-I$OGLHOME/include/GL"
export OGLFLAGS="-I$OGLHOME/lib"
export G4VIS_BUILD_OGLIX_DRIVER=1
export G4VIS_USE_OGLIX=1
export G4UI_USE_XM=1
export G4UI_BUILD_XM_SESSION=1
```

```
export G4VIS_BUILD_VRML_DRIVER=1
export G4VIS_USE_VRML=1
export G4LIB_BUILD_SHARED=1
export G4LIB_USE_GRANULAR=1

export G4UI_USE_TCSH=1
export G4ANALYSIS_USE=1
export G4LIB_BUILD_G3TOG4=1
export G4LIB_USE_G3TOG4=1
export G4LIB_USE_ZLIB=1

export LD_LIBRARY_PATH=$CLHEP_BASE_DIR/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$G4LIB:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH=$G4LIB/$G4SYSTEM:$LD_LIBRARY_PATH$
```

For visualization, the installation of Coin3D is desired. While there are other forms of visualization that can be used, this appears to be the most versatile and is relatively easy to use. The process to install Geant4 with Coin3D is outlined below.

1. http://www.coin3d.org and follow the Download link of Coin3D
2. Copy and unpack into some source directory, like
   */usr/local/src/Coin3D/Coin-2.4.5*
   or whatever current version it is.
3. Run: *./configure –prefix=/usr/local/Coin3D/Coin-2.4.5*
   and note that the installation directory should be different from the source directory.
4. Run: *make install*
   Which compiles Coin3D and generates includes, binaries, and libraries in */usr/local/Coin3D/Coin-2.4.5*
5. Download the SoXt package from the Coin3D site.
6. Copy and unpack into the source directory
   */usr/local/src/Coin3D/SoXt-1.2.2*
   or whatever current version it is.
7. Run: *./configure –prefix=/usr/local/Coin3D/SoXt-1.2.2*
   and note that the installation directory should be different from the source directory.
8. Run: *make install*
   Which compiles SoXt and generates includes, binaries, and libraries in */usr/local/Coin3D/SoXt-1.2.2*

9. Update your environment (see above).

10. Compile (or recompile) Geant4.
    Be sure that your xterm window (in which you will compile Geant4) knows about the updated environment settings (e.g. do a "source /.bashrc)

    Important: Before compiling Geant4, modify the following Geant4 file:
    *G4INSTALL/config/sys/Linux-g++.gmk*
    remove "-pedantic"
    It should look now like this:

```
# CXXFLAGS := -W -Wall -ansi -pedantic -Wno-non-virtual-dtor-Wno-long-long
CXXFLAGS := -W -Wall -ansi           -Wno-non-virtual-dtor-Wno-long-long
```

## 2.3 Compiling QweakSim

The source code is availabe as an archive (a gzipped *tar* file) from the Qweak simulation logbook[1] or via CVS (put info here). Upon downloading the archive and placing it in a directory of your choice, unload the archive by typing at the LINUX terminal prompt:

```
tar -xvzf qweaksim.tar.gz
```

This should create the directories *./include*, *./src*, and *Documentation*. Create a directiry called *tmp*:

```
mkdir tmp
```

Set your current working directory:

```
export G4WORKDIR=‘pwd‘
export G4TMP=‘pwd‘/tmp
```

Compile the simulation:

```
make clean
make
```

[1] http://dilbert.physics.wm.edu/elog/Software/

## 2.4 Running the Simulation

After the very first compilationof the simulation: make a symbolic link to the binary

```
ln -s bin/Linux-g++/QweakSimG4 QweakSimG4
```

Also make symbolic links to the field maps (wherever they are located):

```
ln -s ../MagnetFieldMaps/MainMagnet_FieldMap.dat MainMagnet_FieldMap.dat
ln -s ../MagnetFieldMaps/MiniTorus_FieldMap.dat  MiniTorus_FieldMap.dat
```

Also make a symbolic link to the primary event file:

```
ln -s  ../PrimaryEvents/ROOT/v5.08/ep_m84_qweak_final.root ep_elastic.root
```

Finally, the simulation is started with the following commands

For visualization with new events, using Coin3D

```
   ./QweakSimG4 > mylogfile.txt
```

For running with several k events without visualization

```
./QweakSimG4 myRun.mac > /dev/zero
```

### 2.4.1 Using the visualization

Executing the command:

```
./QweakSimG4 > mylogfile.txt
```

will open a G4UIXm window for steering the simulation.

Within this window click on:

```
Vis->myVis.mac (OIX)
```

This will execute the Geant4 input file *myVis.mac* which will open a Coin3D window for geometry/track visualization. The Coin3D is black/blank initially. Click in the Coin3D window on,

```
Etc->visible mother + visible daughters
```

which will then display the geometry.

Click

```
  Help->Controls
```

for how to use the mouse pointer or the wheels for rotating, zooming, or moving the geometry.

Back in the G4UIXm you can change parameters of the simulation: click on

```
 DetectorHut->ShowTopWall
```

then click in Coin3D on

```
Etc->visible mother + visible daughters
```

Click on

```
TrackingAction->Track primaries only
```

this will track/transport only the primary electrons, as defined by the event generator (rootfile). By default, secondaries are produced but not tracked.

After you have selected your options, click finally on

```
  Run->beamOn X
```

to start the simulation.

If, in Coin3D, the screen gets erased, again select

```
  Etc->visible mother + visible daughters
```

for displaying geometry and tracks. In addition you might erase tracks and/or geometry in Coin3D using

```
  Etc->Erase event
```

or

```
Etc->Erase detector
```

Beware that selecting more than 100 tracks will significantly slow down the visualization, depending on the speed of your machine.

# Chapter 3

# Qweak Primary Event Generation

# Chapter 4

# The QweakSimG4.cc *main(...)* Function

The *QweakSimG4.cc* file implements the standard *main()* function for a GEANT4 simulation. As described in the GANT4 manual, an instance of the run manager object (*runManager*) is created (Line 72 in Fig. 4.1) and initialized using the QWeak objects (Target, Detectors, etc ...), defined in the class *QweakSimDetectorConstruction*, and list of physics processes, defined in the class *QweakSimPhysicsList*. The objects *myQweakSimAnalysis* and *myQweak-SimUserInformation* are instantiated on lines 75 and 76 and are used when the run manager object user actions are defined on lines 90 through 95. If no **.mac* input file is specified when executing the simulation then the code between lines 103 and 114 will be executed, depending on the setting of the environment variables regarding the user interface to be used (see the GEANT4 installation manual). If a **.mac* file is specified as the first argument when executing the simulation then the application is running in batch mode. In batch mode all commands are specified in the **.mac* file.

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

int main(int argc,char** argv) {

  //my Verbose output class
  G4VSteppingVerbose::SetInstance( new QweakSimSteppingVerbose() );

  // Run manager
  G4RunManager * runManager = new G4RunManager();

  // UserInitialization classes (mandatory)
  QweakSimUserInformation*       myQweakSimUserInformation  = new QweakSimUserInformation();
  QweakSimDetectorConstruction*  myQweakSimExperiment       = new QweakSimDetectorConstruction(myQweakSimUserInformation);
  QweakSimAnalysis*              myQweakSimAnalysis         = new QweakSimAnalysis();


  runManager->SetUserInitialization(myQweakSimExperiment);
  runManager->SetUserInitialization(new QweakSimPhysicsList() );








  // UserAction classes
  runManager->SetUserAction( new QweakSimPrimaryGeneratorAction(myQweakSimUserInformation) );
  runManager->SetUserAction( new QweakSimRunAction(myQweakSimAnalysis) );
  runManager->SetUserAction( new QweakSimEventAction(myQweakSimAnalysis, myQweakSimUserInformation) );
  runManager->SetUserAction( new QweakSimSteppingAction(myQweakSimUserInformation) );
  runManager->SetUserAction( new QweakSimStackingAction() );
  runManager->SetUserAction( new QweakSimTrackingAction(myQweakSimUserInformation) );

  //Initialize G4 kernel
  runManager->Initialize();

  G4UIsession* session = 0;

 if (argc==1)   // Define UI session for interactive mode.
 {
    // G4UIterminal is a (dumb) terminal.
   #if defined(G4UI_USE_XM)
    session = new G4UIXm(argc,argv);
   #elif defined(G4UI_USE_WIN32)
    session = new G4UIWin32();
   #elif defined(G4UI_USE_TCSH)
    session = new G4UIterminal(new G4UItcsh);
   #else
     session = new G4UIterminal();
   #endif
 }


#ifdef G4VIS_USE
  // Visualization, if you choose to have it!
  //
  // Simple graded message scheme - give first letter or a digit:
  //  0) quiet,         // Nothing is printed.
  //  1) startup,       // Startup and endup messages are printed...
  //  2) errors,        // ...and errors...
  //  3) warnings,      // ...and warnings...
  //  4) confirmations, // ...and confirming messages...
  //  5) parameters,    // ...and parameters of scenes and views...
  //  6) all            // ...and everything available.

  G4VisManager* visManager = new G4VisExecutive;
```

Figure 4.1:

```
  //visManager -> SetVerboseLevel (1);
  visManager ->Initialize();
#endif


  //get the pointer to the User Interface manager
  G4UImanager * UI = G4UImanager::GetUIpointer();

 if (session)   // Define UI session for interactive mode.
    {
      // G4UIterminal is a (dumb) terminal.
      //UI->ApplyCommand("/control/execute myVis.mac");

#if defined(G4UI_USE_XM) || defined(G4UI_USE_WIN32)
      // Customize the G4UIXm,Win32 menubar with a macro file :
      UI->ApplyCommand("/control/execute gui.mac");
#endif

      session->SessionStart();
      delete session;
    }
  else          // Batch mode
    {
#ifdef G4VIS_USE
      visManager->SetVerboseLevel("quiet");
#endif
      G4String command = "/control/execute ";
      G4String fileName = argv[1];
      UI->ApplyCommand(command+fileName);
    }



#ifdef G4VIS_USE
  delete visManager;
#endif

  delete runManager;

  return 0;
}

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

//=======================================================
//   ----------------------
//  | CVS File Information |
//  ----------------------
//
//      $Revisions$
//      $Log: QweakSimG4.cc,v $
//      Revision 1.2  2005/12/27 19:09:32  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 4.2:

# Chapter 5

# Qweak Overall Construction and Layout

This class implements the QWeak geometry, including the physical layout of the components, their material properties and any fields inside them. The objects and properties currently implemented are listed in table 5.1, together with the respective class names. The experimental layout is determined through this class because the mother volume (experimental hall) is defined here and all other components are placed inside it and with respect to its origin. The properties for the individual components are defined in their respective classes.

Aside from the constructor and destructor, there are 5 public access member functions declared in the header. In the order they are shown in fig. **??** (lines 48 trough 52), they provide a mechanism to construct the geometry (called by the run manager), they provide access to the private data members which specify the size of the world volume (the top most volume), and the last one provides a way to change the geometry between runs, using the **.mac* file.

| **Object / Property** | **Class** |
|---|---|
| Material Type | QweakSimMaterial |
| Magnetic Field | QweakSimMainMagnet |
| Target | QweakSimTarget |
| Vertical Drift Chambers | QweakSimVDC |
| Cerenkov Detectors | QweakSimCerenkovDetector |
| Main Shielding Wall | QweakSimCerenkovDetector |

Table 5.1: List of experiment components and their classes which are currently implemented in the simulation.

```

#ifndef QweakSimDetectorConstruction_h
#define QweakSimDetectorConstruction_h 1

// system includes
#include "cpp_include.h"
#include "Root_include.h"
#include "Geant4_include.hh"

// user includes
#include "QweakSimDetectorMessenger.hh"

#include "QweakSimMaterial.hh"
#include "QweakSimTarget.hh"
#include "QweakSimTargetMessenger.hh"
#include "QweakSimCollimator.hh"
#include "QweakSimCollimatorSupport.hh"
#include "QweakSimShieldingWall.hh"
#include "QweakSimMainMagnet.hh"
#include "QweakSimMiniMagnet.hh"
#include "QweakSimVDC.hh"
#include "QweakSimVDCRotator.hh"
#include "QweakSimHDC.hh"
#include "QweakSimGEM.hh"
#include "QweakSimTriggerScintillator.hh"
#include "QweakSimCerenkovDetector.hh"
#include "QweakSimGlobalMagnetField.hh"
#include "QweakSimUserInformation.hh"

// user classes
class QweakSimDetectorMessenger;

class QweakSimMaterial;
class QweakSimTarget;
class QweakSimTargetMessenger;
class QweakSimCollimator;
class QweakSimCollimatorSupport;
class QweakSimShieldingWall;
class QweakSimMainMagnet;
class QweakSimMiniMagnet;
class QweakSimVDC;
class QweakSimVDCRotator;
class QweakSimHDC;
class QweakSimGEM;
class QweakSimTriggerScintillator;
class QweakSimCerenkovDetector;
class QweakSimGlobalMagnetField;

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

class QweakSimDetectorConstruction : public G4VUserDetectorConstruction
{
  public:

     QweakSimDetectorConstruction(QweakSimUserInformation*);
    ~QweakSimDetectorConstruction();

public:

  G4VPhysicalVolume* Construct();

  void UpdateGeometry();
  void SetGlobalMagneticField();
  void ShowHallFloor();
  void HideHallFloor();

  G4double   GetWorldFullLengthInX()    {return fWorldLengthInX;}
```

Figure 5.1: QweakSimDetectorConstruction Header File

```
  G4double   GetWorldFullLengthInY()    {return fWorldLengthInY;}
  G4double   GetWorldFullLengthInZ()    {return fWorldLengthInZ;}

private:

   QweakSimUserInformation* myUserInfo;
   G4VPhysicalVolume*      ConstructQweak();

  void DumpGeometricalTree(G4VPhysicalVolume* aVolume,G4int depth=0);


  QweakSimMaterial*         pMaterial;
  QweakSimTarget*           pTarget;


  QweakSimCollimator*       pCollimator1;
  QweakSimCollimator*       pCollimator2;
  QweakSimCollimator*       pCollimator3;

  QweakSimCollimatorSupport* pCollimatorSupport;

  QweakSimShieldingWall*     pShieldingWall;

  QweakSimMainMagnet*        pMainMagnet;
  QweakSimMiniMagnet*        pMiniMagnet;

  QweakSimVDC*               pVDC;
  QweakSimHDC*               pHDC;
  QweakSimGEM*               pGEM;

  QweakSimVDCRotator*          pVDCRotator;

  QweakSimTriggerScintillator*  pTriggerScintillator;
  QweakSimCerenkovDetector*     pCerenkovDetector;
  //G4VReadOutGeometry*        pROHitPlane;

  QweakSimDetectorMessenger*      detectorMessenger;  // pointer to the Messenger

     G4Box*             experimentalHall_Solid;     // pointer to the solid envelope
     G4LogicalVolume*   experimentalHall_Logical;   // pointer to the logical envelope
     G4VPhysicalVolume* experimentalHall_Physical;  // pointer to the physical envelope
     G4Material*        experimentalHall_Material;

     G4Box*             HallFloor_Solid;     // pointer to the solid envelope
     G4LogicalVolume*   HallFloor_Logical;   // pointer to the logical envelope
     G4VPhysicalVolume* HallFloor_Physical;  // pointer to the physical envelope
     G4Material*        HallFloor_Material;
     G4VisAttributes*   HallFloor_VisAtt;

     G4double fWorldLength;              // Full length of the world volume
     G4double fWorldLengthInX;           // Full length of the world volume
     G4double fWorldLengthInY;           // Full length of the world volume
     G4double fWorldLengthInZ;           // Full length of the world volume

     G4double fFloorLengthInX;
     G4double fFloorLengthInY;
     G4double fFloorLengthInZ;
     G4double fFloorPositionInY;

  //---------------------
  // global magnet section
  //---------------------
  //
  QweakSimGlobalMagnetField*      pGlobalMagnetField;

     G4FieldManager*         fGlobalFieldManager;
     G4ChordFinder*          fGlobalChordFinder;
```

Figure 5.2: QweakSimDetectorConstruction Header File

```
    G4Mag_UsualEqRhs*       fGlobalEquation;
    G4MagIntegratorStepper* fGlobalStepper;

    G4double                fMinStep;

};

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

#endif

```

Figure 5.3: QweakSimDetectorConstruction Header File

```

#include "QweakSimDetectorConstruction.hh"

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

//===================================================================
//       Qweak *Geant4* Geometry Conventions:
//      The origin is at the center of the main toroidal magnet with
//       the z-axis pointing along the beam direction, the y-axis
//       pointing toward the ceiling, and the x-axis pointing toward
//       beam-left so as to form a right-handed coordinate system.
//       Octants are numbered from 1 to 8, clockwise with #1 at the
//       12 o-clock position.
//===================================================================

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

QweakSimDetectorConstruction::QweakSimDetectorConstruction(QweakSimUserInformation *userInfo)
{
  // initialize pointers and variables
  experimentalHall_Solid    = NULL;
  experimentalHall_Logical  = NULL;
  experimentalHall_Physical = NULL;

  HallFloor_Solid    = NULL;
  HallFloor_Logical  = NULL;
  HallFloor_Physical = NULL;

  detectorMessenger = NULL;
  pMaterial         = NULL;

  pGlobalMagnetField = NULL;

  pTriggerScintillator  = NULL;
  pCerenkovDetector     = NULL;

  pCollimator1       = NULL;
  pCollimator2       = NULL;
  pCollimator3       = NULL;
  pCollimatorSupport = NULL;


  pShieldingWall     = NULL;

  pGEM               = NULL;
  pHDC               = NULL;
  pVDC               = NULL;
  pVDCRotator        = NULL;

  pTarget            = NULL;
  pMainMagnet        = NULL;
  pMiniMagnet        = NULL;

  fWorldLengthInX = 0.0*cm;
  fWorldLengthInY = 0.0*cm;
  fWorldLengthInZ = 0.0*cm;

  fFloorLengthInX   = 0.0*cm;
  fFloorLengthInY   = 0.0*cm;
  fFloorLengthInZ   = 0.0*cm;
  fFloorPositionInY = 0.0*cm; // Top positon, not the center pos

  fGlobalEquation    = NULL;
  fGlobalStepper     = NULL;
  fGlobalChordFinder = NULL;

  HallFloor_VisAtt = NULL;
```

Figure 5.4: QweakSimDetectorConstruction Source File

```
  HallFloor_VisAtt = new G4VisAttributes();

  myUserInfo = userInfo;

  detectorMessenger = new QweakSimDetectorMessenger(this);

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

QweakSimDetectorConstruction::~QweakSimDetectorConstruction()
{
  // I'm deleting the objects in the reverse order they were created (~FILO)

  if (pGlobalMagnetField) delete pGlobalMagnetField;

  if (pVDCRotator) delete pVDCRotator;
  if (pGEM)       delete pGEM;
  if (pHDC)       delete pHDC;
  //if (pVDC) delete pVDC; // something broken here


  if (pCerenkovDetector)    delete pCerenkovDetector;
  if (pTriggerScintillator) delete pTriggerScintillator;


  if (pCollimator1)       delete pCollimator1;
  if (pCollimator2)       delete pCollimator2;
  if (pCollimator3)       delete pCollimator3;
  if (pCollimatorSupport)   delete pCollimatorSupport;


  if (pShieldingWall)      delete pShieldingWall;

  if (pTarget)            delete pTarget;
  if (pMainMagnet)        delete pMainMagnet;
  if (pMiniMagnet)        delete pMiniMagnet;

  if (detectorMessenger)   delete detectorMessenger;
  if (pMaterial)          delete pMaterial;

}

G4VPhysicalVolume* QweakSimDetectorConstruction::Construct()
{
  return ConstructQweak();
}

G4VPhysicalVolume* QweakSimDetectorConstruction::ConstructQweak()
{

  pTarget            = new QweakSimTarget();

  pCollimator1       = new QweakSimCollimator();
  pCollimator2       = new QweakSimCollimator();
  pCollimator3       = new QweakSimCollimator();

  pShieldingWall     = new QweakSimShieldingWall();


  pMiniMagnet        = new QweakSimMiniMagnet(); // MiniTorus Geometry (decoupled from field)
  pMainMagnet        = new QweakSimMainMagnet(); // QTOR Geometry (decoupled from field)

  pGEM               = new QweakSimGEM();
  pHDC               = new QweakSimHDC();
  pVDC               = new QweakSimVDC();

```

Figure 5.5: QweakSimDetectorConstruction Source File

```
//pCerenkovDetector    = new QweakSimCerenkovDetector();
pCerenkovDetector    = new QweakSimCerenkovDetector(myUserInfo);

pTriggerScintillator = new QweakSimTriggerScintillator();


//--------- Definitions of Solids, Logical Volumes, Physical Volumes ---------

fWorldLengthInX =  15.0*m;
fWorldLengthInY =  15.0*m;
fWorldLengthInZ =  30.0*m;


// experimentalHall_Material   = pMaterial->GetMaterial("HeGas");
// Note: experimentalHall_Material was HeGas all the time up to 12-28-2005 !!!
experimentalHall_Material   = pMaterial->GetMaterial("Air");

experimentalHall_Solid = new G4Box("ExpHall_Sol",
                      0.5* fWorldLengthInX ,
                      0.5* fWorldLengthInY ,
                      0.5* fWorldLengthInZ );

experimentalHall_Logical = new G4LogicalVolume( experimentalHall_Solid,
                                experimentalHall_Material,
                                "ExpHall_Logical",
                                0, 0, 0);

//  Must place the World Physical volume unrotated at (0,0,0).
//
experimentalHall_Physical = new G4PVPlacement(0,              // no rotation
                              G4ThreeVector(), // at (0,0,0)
                              experimentalHall_Logical,      // its logical volume
                              "ExpHall_Physical",        // its name
                              0,             // its mother  volume
                              false,          // no boolean operations
                              0);            // no field specific to volume

//==========================
// Defining the Hall Floor
//==========================

fFloorLengthInX =  12.0*m;
fFloorLengthInY =   1.0*m;
fFloorLengthInZ =  28.0*m;

fFloorPositionInY =  -396.25*cm; // Top positon, not the center pos


HallFloor_Material   = pMaterial->GetMaterial("ShieldingConcrete");

HallFloor_Solid = new G4Box("HallFloor_Sol",
                  0.5* fFloorLengthInX ,
                  0.5* fFloorLengthInY ,
                  0.5* fFloorLengthInZ );

HallFloor_Logical = new G4LogicalVolume( HallFloor_Solid,
                          HallFloor_Material,
                          "HallFloor_Logical",
                          0, 0, 0);

//  Must place the World Physical volume unrotated at (0,0,0).
//
HallFloor_Physical = new G4PVPlacement(0,              // no rotation
                        G4ThreeVector(0.,fFloorPositionInY -0.5* fFloorLengthInY,0.),
                        "HallFloor_Physical",         // its name
                        HallFloor_Logical,            // its logical volume
                        experimentalHall_Physical ,   // its physical mother volume
```

Figure 5.6: QweakSimDetectorConstruction Source File

```
                          false,                 // no boolean operations
                          0);                    // no field specific to volume


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

  G4Colour  grey      ( 127/255., 127/255., 127/255.);

  HallFloor_VisAtt->SetColor(grey);
  HallFloor_VisAtt->SetVisibility(true);
  //HallFloor_VisAtt->SetVisibility(false);
  //HallFloor_VisAtt->SetForceWireframe(true);
  //HallFloor_VisAtt->SetForceSolid(true);
  HallFloor_Logical->SetVisAttributes(HallFloor_VisAtt);

  //============================================
  // create/place target body into MotherVolume
  //============================================
  //
  pTarget -> ConstructComponent(experimentalHall_Physical);
  pTarget -> SetTargetCenterPositionInZ(-650*cm);

  //===============================================
  // create/place MainMagnet body into MotherVolume
  //===============================================
  //
  if(pMiniMagnet){
    pMiniMagnet -> ConstructComponent(experimentalHall_Physical);
    pMiniMagnet -> SetCenterPositionInZ(-465.31*cm);
  }

  //===============================================
  // create/place MainMagnet body into MotherVolume
  //===============================================
  //
  if(pMainMagnet){
    pMainMagnet -> ConstructComponent(experimentalHall_Physical);
    pMainMagnet -> SetCenterPositionInZ(0.0*cm);
    pMainMagnet -> Construct_UpstreamSpider(experimentalHall_Physical);
    pMainMagnet -> Construct_ClampPlates(experimentalHall_Physical);
    pMainMagnet -> Construct_UpStreamMiniClampPlates(experimentalHall_Physical);
    pMainMagnet -> Construct_CoilFrames(experimentalHall_Physical);
    pMainMagnet -> Construct_RadialMountingBlocks(experimentalHall_Physical);
    pMainMagnet -> Construct_SupportFrame(experimentalHall_Physical);
    pMainMagnet -> Construct_DownstreamSpider(experimentalHall_Physical);
  }



  //Collimator 1 configuration
  pCollimator1->SetCollimatorNumber(1);
  pCollimator1->SetCollimatorHousing_FullLengthInX(240.0*cm);
  pCollimator1->SetCollimatorHousing_FullLengthInY(240.0*cm);
  pCollimator1->SetCollimatorHousing_FullLengthInZ(15.24*cm);

  pCollimator1->SetOctantCutOutFrontFullLength_Y(3.28*cm);
  pCollimator1->SetOctantCutOutFrontFullLength_X1(7.66*cm);
  pCollimator1->SetOctantCutOutFrontFullLength_X2(7.66*cm);
  pCollimator1->SetOctantCutOutBackFullLength_Y(6.24*cm);
  pCollimator1->SetOctantCutOutBackFullLength_X1(7.66*cm);
  pCollimator1->SetOctantCutOutBackFullLength_X2(7.66*cm);

  pCollimator1->SetBeamlineCutoutDiameter(8.0*cm);

  pCollimator1->SetOctantCutOutFrontInnerDiameter(100.0*mm);
  pCollimator1->SetOctantCutOutFrontOuterDiameter(261.4*mm);
  pCollimator1->SetOctantCutOutBackInnerDiameter(124.0*mm);
```

Figure 5.7: QweakSimDetectorConstruction Source File

```
pCollimator1->SetOctantCutOutBackOuterDiameter(261.4*mm);
pCollimator1->SetOctantCutOutStartingPhiAngle((-16.344+90.0)*degree);
pCollimator1->SetOctantCutOutDeltaPhiAngle(2.0*16.344*degree);
pCollimator1->SetOctantCutOutRadialOffset(0.0*cm);

pCollimator1->ConstructCollimator(experimentalHall_Physical);

pCollimator1->SetCollimatorHousing_CenterPositionInZ(-575.79*cm);
pCollimator1->SetCollimatorHousingMaterial("CDA943");


//Collimator 2
pCollimator2->SetCollimatorNumber(2);
pCollimator2->SetCollimatorHousing_FullLengthInX(240.0*cm);
pCollimator2->SetCollimatorHousing_FullLengthInY(240.0*cm);
pCollimator2->SetCollimatorHousing_FullLengthInZ(21.66*cm);

pCollimator2->SetOctantCutOutFrontFullLength_Y(16.96*cm);
pCollimator2->SetOctantCutOutFrontFullLength_X1(20.16*cm); //lower edge
pCollimator2->SetOctantCutOutFrontFullLength_X2(20.08*cm); //was 20.08//upper edge
pCollimator2->SetOctantCutOutBackFullLength_Y(21.96*cm);
pCollimator2->SetOctantCutOutBackFullLength_X1(20.16*cm);
pCollimator2->SetOctantCutOutBackFullLength_X2(20.06*cm); //was 20.06

pCollimator2->SetBeamlineCutoutDiameter(8.0*cm);

pCollimator2->SetOctantCutOutFrontInnerDiameter(31.47*cm);
pCollimator2->SetOctantCutOutFrontOuterDiameter(48.75*cm);
pCollimator2->SetOctantCutOutBackInnerDiameter(34.87*cm);
pCollimator2->SetOctantCutOutBackOuterDiameter(48.75*cm);
pCollimator2->SetOctantCutOutStartingPhiAngle((-22.467+90.0)*degree);
pCollimator2->SetOctantCutOutDeltaPhiAngle(2.0*22.467*degree);
pCollimator2->SetOctantCutOutRadialOffset(15.665*cm);

pCollimator2->ConstructCollimator(experimentalHall_Physical);

pCollimator2->SetCollimatorHousing_CenterPositionInZ(-349.889*cm);
pCollimator2->SetCollimatorHousingMaterial("CDA943");

//Collimator 3
pCollimator3->SetCollimatorNumber(3);
pCollimator3->SetCollimatorHousing_FullLengthInX(240.0*cm);
pCollimator3->SetCollimatorHousing_FullLengthInY(240.0*cm);
pCollimator3->SetCollimatorHousing_FullLengthInZ(15.24*cm);

pCollimator3->SetOctantCutOutFrontFullLength_Y(30.37*cm);
pCollimator3->SetOctantCutOutFrontFullLength_X1(34.44*cm); //lower edge
pCollimator3->SetOctantCutOutFrontFullLength_X2(34.44*cm); //upper edge
pCollimator3->SetOctantCutOutBackFullLength_Y(33.37*cm);
pCollimator3->SetOctantCutOutBackFullLength_X1(34.44*cm);
pCollimator3->SetOctantCutOutBackFullLength_X2(34.44*cm);

pCollimator3->SetBeamlineCutoutDiameter(8.0*cm);

pCollimator3->SetOctantCutOutFrontInnerDiameter(2.0*38.0*cm);
pCollimator3->SetOctantCutOutFrontOuterDiameter(2.0*48.63*cm);
pCollimator3->SetOctantCutOutBackInnerDiameter(2.0*39.5*cm);
pCollimator3->SetOctantCutOutBackOuterDiameter(2.0*48.63*cm);
pCollimator3->SetOctantCutOutStartingPhiAngle((-19.499+90.0)*degree);
pCollimator3->SetOctantCutOutDeltaPhiAngle(2.0*19.499*degree);
pCollimator3->SetOctantCutOutRadialOffset(0.0*cm);

pCollimator3->ConstructCollimator(experimentalHall_Physical);

pCollimator3->SetCollimatorHousing_CenterPositionInZ(-264.239*cm);
pCollimator3->SetCollimatorHousingMaterial("CDA943");
```

Figure 5.8: QweakSimDetectorConstruction Source File

```
//================================================
// create/place Collimator Support body into MotherVolume
//================================================
//
  pCollimatorSupport   = new QweakSimCollimatorSupport( pCollimator1 ,pCollimator3 );
  pCollimatorSupport -> ConstructSupport(experimentalHall_Physical);


// #==================================================
// # create/place ShieldingWall body into MotherVolume
// #==================================================

  pShieldingWall->SetCollimatorWall_FullLengthInX(670.56*cm);
  pShieldingWall->SetCollimatorWall_FullLengthInY(670.56*cm);
  pShieldingWall->SetCollimatorWall_FullLengthInZ( 50.0*cm);

  pShieldingWall->SetOctantCutOut_Trap_RadialDistance  (250.75*cm);
  pShieldingWall->SetOctantCutOut_Trap_FullLengthFront (150.00*cm);
  pShieldingWall->SetOctantCutOut_Trap_FullLengthBack  (164.00*cm);
  pShieldingWall->SetOctantCutOut_Trap_FullHeightFront ( 34.50*cm);
  pShieldingWall->SetOctantCutOut_Trap_FullHeightBack  ( 30.50*cm);
  pShieldingWall->SetOctantCutOut_Trap_PolarAngle      ( 20.57*degree);

  pShieldingWall->ConstructShieldingWallHousing_UsingTrapezoids(experimentalHall_Physical);
  pShieldingWall->SetCollimatorWall_CenterPositionInZ(355*cm);

  pShieldingWall->SetCollimatorWallMaterial("ShieldingConcrete");
  //pShieldingWall->SetCollimatorWallMaterial("Lead");

  pShieldingWall->ConstructFrontWall(experimentalHall_Physical);
  pShieldingWall->ConstructBackWall(experimentalHall_Physical);
  pShieldingWall->ConstructBeamLeftSideWall(experimentalHall_Physical);
  pShieldingWall->ConstructBeamRightSideWall(experimentalHall_Physical);
  pShieldingWall->ConstructTopWall(experimentalHall_Physical);

  //================================================
  // create/place Drift Chambers into MotherVolume
  //================================================
  //
  pGEM->ConstructComponent(experimentalHall_Physical);
  pHDC->ConstructComponent(experimentalHall_Physical);
  pVDC->ConstructComponent(experimentalHall_Physical);

  //================================================
  // create/place VDC Rotator into MotherVolume
  //================================================
  //
  pVDCRotator  = new QweakSimVDCRotator(pVDC);
  pVDCRotator->SetMotherVolume(experimentalHall_Physical);
  pVDCRotator->ConstructRings();
  pVDCRotator->ConstructRails();
  pVDCRotator->ConstructMount();
  pVDCRotator->ConstructSliderSupport();
  pVDCRotator->SetRotationAngleInPhi( 0.0*degree);

  //==========================================
  // create/place Cerenkov into MotherVolume
  //==========================================
  //
  pCerenkovDetector->ConstructComponent(experimentalHall_Physical);

  //=====================================================
  // create/place Trigger Scintillator into MotherVolume
  //=====================================================
  //
  pTriggerScintillator->ConstructComponent(experimentalHall_Physical);
```

Figure 5.9: QweakSimDetectorConstruction Source File

```
//--------- Visualization attributes -------------------------------

  // Invisible Volume
  experimentalHall_Logical->SetVisAttributes (G4VisAttributes::Invisible);

  G4cout << G4endl << "The geometrical tree defined are : " << G4endl << G4endl;
  DumpGeometricalTree(experimentalHall_Physical);

  G4cout << G4endl << "###### Leaving QweakSimDetectorConstruction::Construct() " << G4endl << G4endl;


  SetGlobalMagneticField();

  // Construct() *MUST* return the pointer of the physical World !!!
  return experimentalHall_Physical;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimDetectorConstruction::DumpGeometricalTree(G4VPhysicalVolume* aVolume,G4int depth)
{
  for(int isp=0;isp<depth;isp++)
  { G4cout << "  "; }
  G4cout << aVolume->GetName() << "[" << aVolume->GetCopyNo() << "] "
         << aVolume->GetLogicalVolume()->GetName() << " "
         << aVolume->GetLogicalVolume()->GetNoDaughters() << " "
         << aVolume->GetLogicalVolume()->GetMaterial()->GetName();
  if(aVolume->GetLogicalVolume()->GetSensitiveDetector())
  {
    G4cout << " " << aVolume->GetLogicalVolume()->GetSensitiveDetector()
                        ->GetFullPathName();
  }
  G4cout << G4endl;
  for(int i=0;i<aVolume->GetLogicalVolume()->GetNoDaughters();i++)
  { DumpGeometricalTree(aVolume->GetLogicalVolume()->GetDaughter(i),depth+1); }
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimDetectorConstruction::UpdateGeometry()
{
  G4cout << G4endl << "###### Calling QweakDetectorConstruction::UpdateGeometry() " << G4endl << G4endl;

  // taken from LXe example
  G4GeometryManager::GetInstance()->OpenGeometry();

  // clean up previous geometry
  G4PhysicalVolumeStore  ::GetInstance()->Clean();
  G4LogicalVolumeStore   ::GetInstance()->Clean();
  G4SolidStore           ::GetInstance()->Clean();
  G4LogicalBorderSurface ::CleanSurfaceTable();


  // define new geometry
  G4RunManager::GetRunManager()->DefineWorldVolume(ConstructQweak());
  G4RunManager::GetRunManager()->GeometryHasBeenModified();


  G4cout << G4endl << "###### Leaving QweakDetectorConstruction::UpdateGeometry() " << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimDetectorConstruction::SetGlobalMagneticField()
{

  //--------- Magnetic Field -------------------------------

  //============================================
```

Figure 5.10: QweakSimDetectorConstruction Source File

```
//  Define the global magnet field Manager
//=============================================
pGlobalMagnetField = new QweakSimGlobalMagnetField();

// Get transportation, field, and propagator  managers
fGlobalFieldManager = G4TransportationManager::GetTransportationManager()->GetFieldManager();

// perform navigation/propagation in a general non-uni-form magnetic field
//G4PropagatorInField* pGlobalFieldPropagator = G4TransportationManager::GetTransportationManager()->GetPropagatorInField();

// G4double minEps =  1.0e-5;
// G4double maxEps =  1.0e-4;

//pGlobalFieldPropagator->SetMinimumEpsilonStep(minEps);
//pGlobalFieldPropagator->SetMaximumEpsilonStep(maxEps);


fGlobalFieldManager->SetDetectorField(pGlobalMagnetField);

fGlobalEquation = new G4Mag_UsualEqRhs(pGlobalMagnetField);

// taken from one of the Geant4 presentation:
// - If the field is calculated from a field map, a lower order stepper
//   is recommended: the less smooth the field is, the lower the order of the
//   stepper that should be used. For very rough fields one should use 1st order
//   stepper, for a somewhat smooth field one must choose between 1st and 2nd
//   order stepper.

//fGlobalStepper  = new G4ClassicalRK4(fGlobalEquation);  // classical 4th order stepper
//fGlobalStepper  = new G4ExplicitEuler(fGlobalEquation); //          1st order stepper
//fGlobalStepper  = new G4ImplicitEuler(fGlobalEquation); //          2nd order stepper
fGlobalStepper  = new G4SimpleRunge(fGlobalEquation);   //          2nd order stepper


// Provides a driver that talks to the Integrator Stepper, and insures that
//   the error is within acceptable bounds.
G4MagInt_Driver* fGlobalIntgrDriver = new G4MagInt_Driver(1.0e-3*mm,
                                          fGlobalStepper,
                                          fGlobalStepper->GetNumberOfVariables());

fGlobalChordFinder = new G4ChordFinder(fGlobalIntgrDriver);



//       G4bool fieldChangesEnergy = false;
//       G4FieldManager* pFieldMgr = new G4FieldManager(myField,pChordFinder,FieldChangeEnergy);
//       LocalLogicalVolume = new G4LogicalVolume(shape, material,"name",pFieldMgr,0,0);

//   // minimal step of 1 mm is default
//   fMinStep = 0.01*mm ;
//
//   fGlobalChordFinder = new G4ChordFinder (pGlobalMagnetField,
//                                    fMinStep,
//                                    fGlobalStepper);

fGlobalFieldManager->SetChordFinder(fGlobalChordFinder);

//==================================================================================
// you can use this in DetectorConstruction class to make more smooth visulisation:

G4TransportationManager* tmanager = G4TransportationManager::GetTransportationManager();
tmanager->GetPropagatorInField()->SetLargestAcceptableStep(1*mm);
//==================================================================================
}
```

Figure 5.11: QweakSimDetectorConstruction Source File

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimDetectorConstruction::ShowHallFloor()
{
  G4cout << "###### Calling QweakSimDetectorConstruction::ShowHallFloor() " << G4endl << G4endl;

  HallFloor_VisAtt->SetVisibility(true);

  G4cout << "###### Leaving QweakSimDetectorConstruction::ShowHallFloor() " << G4endl << G4endl;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimDetectorConstruction::HideHallFloor()
{
  G4cout << "###### Calling QweakSimDetectorConstruction::HideHallFloor() " << G4endl << G4endl;

  HallFloor_VisAtt->SetVisibility(false);

  G4cout << "###### Leaving QweakSimDetectorConstruction::HideHallFloor() " << G4endl << G4endl;
}
```

Figure 5.12: QweakSimDetectorConstruction Source File

# Chapter 6

# Qweak Main Cerenkov Detector

This chapter explains the implementation of the QWeak main detector in the simulation, including the physical layout and material properties as well as event data readout. The chapter also provides a discussion (code trace) of the way GEANT4 handles the optical transport physics. This is discussed somewhat in the GEANT4 physics reference manual [3][1], but not in very much detail. Here, we verify the physics by looking at code itself and relating it to the material and geometry specifications made in the QWeak simulation. For the QWeak main detectors, the two most relevant GEANT4 classes which specify the physics are *G4Cerenkov* and *G4OpBoundaryProcess*.

The overal detector geometry and properties are implemented in the class *QweakSimCerenkovDetector*. However, some of the detector material properties, the data readout and the sensitivity are implemented in combination with other classes listed in table 6.1, some of which serve multiple objects at the same time. The relation to and use of each of these classes for the main detectors is discussed in separate sections.

## 6.1 Detector Implementation

The detector atributes or properties that are currently implemented are listed in table 6.2. Each property has one or several sections devoted to it, describing the details of the code. Please refer to the code listings provided at the end of each chapter for each class. Components placed inside another volume (their respective parent volume) are positioned with respect to the center of the parent volume.

Aside from the constructor and destructor, there are 16 public access member functions declared in the header. Shown in fig. 6.10 (lines 26 trough 46), they provide mechanisms to

[1] `http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/PhysicsReferenceManual/html/index.html`

| Object / Property | Class | Use |
|---|---|---|
| Quartz Sensitive Volume | Cerenkov_DetectorSD | event definition and sensitivity cuts |
| PMT Sensitive Volume | CerenkovDetector_PMTSD | event definition and sensitivity cuts |
| Material Type | Material | obtain material properties |
| Event Info Container | UserInformation | pass event information between objects |
| Partcile Step Info | SteppingAction | extract information for each particle step |
| Partcile Track Info | trackingAction | extract information for each particle track (used for secondaries) |
| Event Readout | UserCerenkov_MainEvent<br>UserCerenkov_OctantEvent<br>UserCerenkov_DetectorEvent<br>UserCerenkov_PMTEvent | define the tree structure for the Cerenkov and PMT data readout |

Table 6.1: List of detector and connected classes which are currently implemented.

control the detector position, construct or remove the detector geometry (called by *QweakSimDetectorConstruction*, p. 34), update the geometry between runs, as specified by the position variables passed to the setters (lines 34 through 38), and provide access to the sensitive logical and physics volume which are private data members used by the Geant4 run manager. The function *CerenkovGeometryPVUpdate* provides a way to change the geometry between runs, using the input (**.mac*) files (see p. 21). The meat of the detector implementation is handled in the function *ConstructComponent*. Some of the functions, such as *DefineCerenkovGeometry* are currently not used.

### 6.1.1 Geometry

The detector geometry is specified via the use of containers within containers. The configuration is shown in figure 6.1.

## Master Container

The top level volume for each of the 8 detectors is named *CerenkovMasterContainer*. The container and its attributes are declared in the header file (lines 59 through 62, fig. 6.10) and the 8 volumes are defined and positioned in their octant position in the function *PlacePVCerenkovMasterContainer* in the source file (see fig. 6.36). The mother volume is the experimental hall and a pointer of it is passed to the class via the function *ConstructComponent* (fig. 6.16, line 171), from the class *QweakSimDetectorConstruction* (p. 34, line 396 in chapter 15). All other volumes, from which the detectors are constructed are contained within *PlacePVCerenkovMasterContainer*. Lines 180 through 214 of the source file (pp. 70- 71) define the logical volume of the master container. Note that the physical volume definition and the positioning there have been replaced with the code in the functions *CerenkovGeometryPVUpdate* and *PlacePVCerenkovMasterContainer* (pp. 85- 86), which is called at the end of the function *ConstructComponent* (line 1025, p. 80). The main container (*CerenkovMasterContainer*) is not a sensitive volume (no hits are counted in it) and has the same material properties as the surrounding experimental hall volume (air) (line 100, p. 65) and is therefore indistinguishable from the mother volume. It's only purpose is to combine all other volumes making up the detector and to facilitate their positioning as a unit.

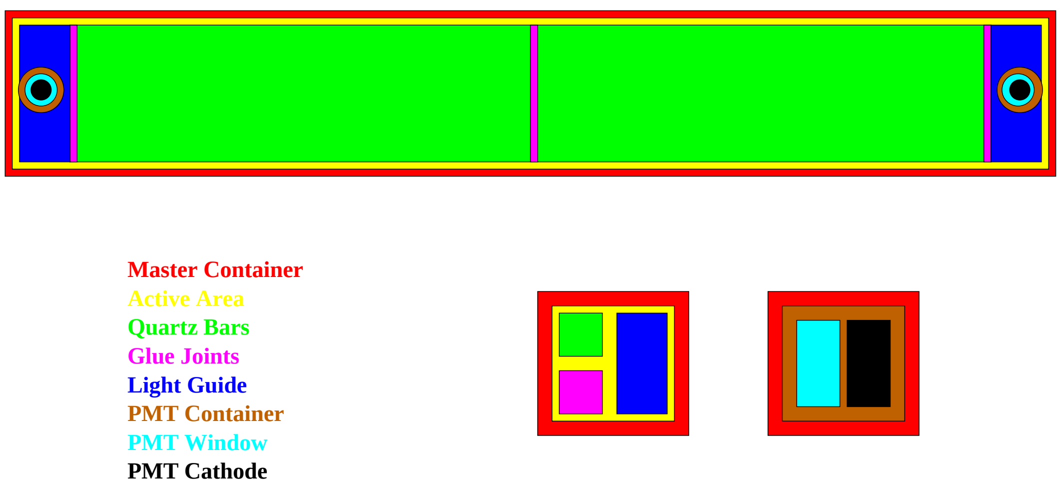


Figure 6.1: Volumes that make up the detector implentation and their containment within and with respect to each other. The detector active volume combines the quartz bars, the glue joints, and the light guides. The active area and the PMT container are contained within the master container, the PMT window and cathode, as well as an additional glue joint (not shown) are contained within the PMT container. Also not shown are several volumes used to simulate mirrored surfaces.

## Active Volume

The active volume (*ActiveArea_Physical*) is placed inside the logical master container (lines 217 - 239, pp. 67) and combines the volumes of the 2 quartz bars, three glue joints, one

| Property | Detail | Class |
|---|---|---|
| Geoemtry | 13 main components for each octant including: 2 quartz bars, 2 light guides, 5 glue joints, 2 cathodes, and 2 PMT entrance windows | CerenkovDetector |
| Optical transport | -refelectivity as a function of photon energy<br>-absorption as a function of energy<br>-index of refraction as a function of energy<br>-cathode quantum efficiency<br>-dielectric to dielectric boundary properties | CerenkovDetector<br>Material<br>UserInformation<br>EventAction |
| Sensitive Volumes | The quartz bars and PMT cathodes are both implemented as sensitive volumes. | Cerenkov_CerenkovSD<br>CerenkovDetector_PMTSD |
| Cuts and Event Readout | -$\gamma$ & x-ray energy cuts<br>-particle definition and energy storage at time of hit<br>-secondary particle origin and creator process<br>-quartz energy deposition<br>-optical photon count | CerenkovDetector<br>Cerenkov_CerenkovSD<br>CerenkovDetector_PMTSD<br>SteppingAction |

Table 6.2: List of detector properties that are currently implemented and the corresponding classes where the code can be found.

between the bar and two at the ends, interfacing to the light guides and the light guides themselves. The active volume's size is exactly that of these 5 volumes combined and is specified as the sensitive volume (lines 1181-1187, p. 82), for which hits are collected. The active volume material is air and as such, there is no distinction between this volume and the master container or the mother volume (the experimental hall); it simply serves as a hit counter. Valid events are established when a particle generates optical photons in the quartz bars or light guides. The corresonding energy deposition and optical photon count for each event are collected in the class *QweakSimSteppingAction* (chapter 17). The active volume is also used to define the logical border surface for the detector wrapping with various materials (millipore in this case) (lines 1143 - 1172, pp. 81 - 82). The active volume is specified as wrapped with millipore paper (lines 1150 - 1169, p. 80). This simulates the wrapping of the quartz bars with some air between the bars and wrapping material.

### Quartz Bars

Lines 247 through 510 of the source code (pp. 67 - 71) implement the quartz radiator, consisting of two separate meter long quartz bars and three glue joints. The material properties for the quartz are specified in the class *QweakSimMaterial* (chapter 18, lines 320 - 326, p. 187). The quartz bars are defined with chamfers that can be adjusted for scew, which is the degree to which they are not perfectly parallel with the edge of the quartz bar. This amounts to a rotation around Z and Y, while the quartz bars are oriented with their long axis along X. THis is done for all four long edges of each meter long bar. Currently, for lack of complete knowledge about the chemical composition, the glue joints are implemented with the same material properties as the quartz bars. However, the actual glue joints used in the detector construction are made from a silicon elastomere, so this should be good first order approximation.

### Light Guides

Lines 512 through 704 (pp. 72 - 74) implement the light guide geometry. The guides are also made of quartz and are chamfered as well. However, chamfer scew is not currently implemented for the light guides. Although the light guide design is now fixed to be rectangular, the guides are implemented here as trapeziods (with the dimensions set to form a rectangle), because they were adjusted to have various trpeziodal shapes during the design process. The sculpting of the guide edges is also implemented so that the left-right edges of the detector assembly can be cut at angles. None of the guide sculpting is currently used (see lines 515 and 536), leaving plain rectangular guides with 90 degree chamfered edges everywhere. The mirroring of the guide faces and edges is implemented between line 706 (p. 75) and line 800 (p. 76). The face mirrors are currently turned off. The mirror material and optical properties are defined in the class *QweakSimMaterial* (lines 363 - 370, p 188, lines 800 - 818, p. 195).

### Pre (Shower) Radiator

The implementation of the pre-radiator (lines 806 - 828, p. 76) consists of a lead or tungsten bar placed in fornt of the quartz (this is currently turned off).

### PMT Geometry

Lines 833 (p. 77) through 993 (p. 79) implement the PMT, consisting of (in order, as seen by a photon coming from the detector and traversing the PMT) an entrance window, a glue layer, and a cathode (see figs. 6.2 and 6.3), all of which are enclosed in the PMT container volume. The thickness of the PMT entrance window and cathode are rather arbitrarily set to be 1 mm. However, both volumes are typically much thinner in reality and any simulated absorption in the window would therefore produce a lower value in the simulated number of photons hitting the cathode, than what would be expected on these grounds in the actual PMT. Lines 836 through 839 define variables needed to change the PMT position when the light guides are changed to arbitrary trapezoids. Since the light guides are now fixed to be rectangular, these variables are no longer used.

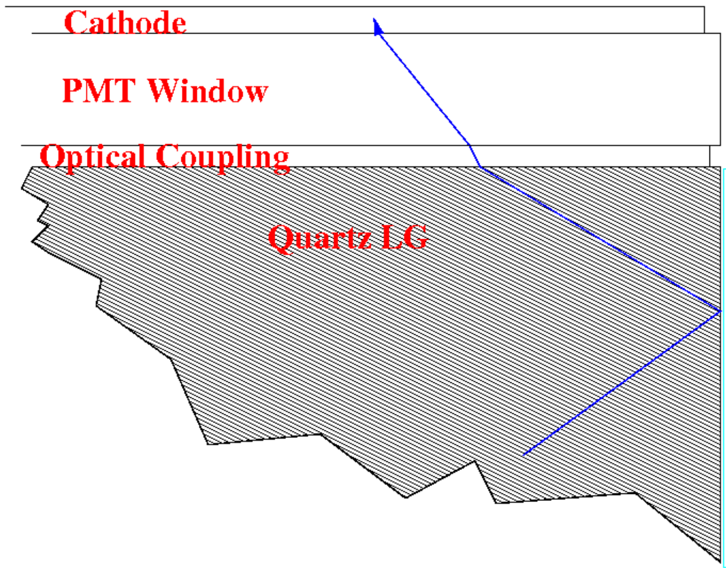


Figure 6.2: Illustration of the PMT geometry and a photon ray through it. The right side of the light guide is shown as mirrored.

### 6.1.2 Optical and Related Properties

The optical properties of the main detector are specified for the quartz bars, the light guides, the PMT cathode and entrance windows, all glue joints, and a wrapping layer around the active volume. The implemented optical properties include wavelength dependent index of refraction, absorption coefficient (length), reflectivity as well as surface properties (roughness, interface type, etc..). Most of these are specified in the *QweakSimCerenkovDetector* and *QweakSimMaterial*. The only exception is the quantum efficiency of the S20 cathode used in the experiment (fig. 6.4), which is specified in *QweakSimUserInformation* (lines 54, p. 200 through 154, p. 202).

#### Quartz

The emission wavelength range for the quartz bars is 250 to 800 nm, but sharply peaked around 280 nm. Based on that, the optical properties are specified for wavelengths for 210 to 800 nm wavelengths. The optical properties for the quartz (including the light guides)(except for the reflectivity) are specified in the class *QweakSimMaterial*, (chapter 18, lines 708 -

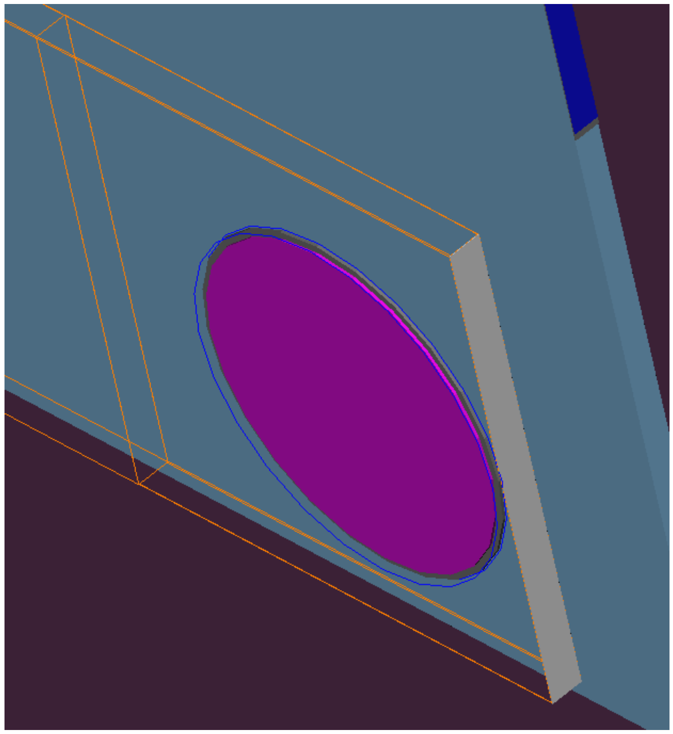

Figure 6.3: PMT geometry in the simulation, with an entrance window (grey), a cathode (magenta), and the container (blue wire frame). The glue joint is not visible.

758, pp. 193- 194). The reflectivity is implemented in the detector source file itself (lines 1030-1057, p. 80 and line 1135, p. 81), together with the surface definitions, since it is a surface property rather than a bulk material property. The surface polish specified to the bar manufacturer is 25 Angstroms [4] with an average reflectivity of 0.997. The specified reflectivity follows an emperical equation (eqn. 6.1) [5].

$$R = 1 - 0.027e^{-0.0046\lambda} \ . \tag{6.1}$$

Lines 1059 through 1140 define the border surfaces between the quartz bar and the active volume, including the surface roughness (polish), the interface type (dielectric to dielectric) and the model (glisur, see section 6.2).

### PMT

Since the real PMT windows are specifically UV transmitting, the simulated window material has been set to quartz (i.e. a perfect index of refraction match). However, the index of refraction for the glue joints used here (and between the quartz bars and light guides described above) is different from the one used for the quartz (see class *QweakSimMaterial*, lines 767 - 777, p. 194). Currently, there is only one value available for the index of refraction of the glue [**?**] which has been used over the wavelength range of interest. There is currently also no information on hand for the absorption length of the elastomere glue, which has therefore been set to be identical to that of the quartz [6].

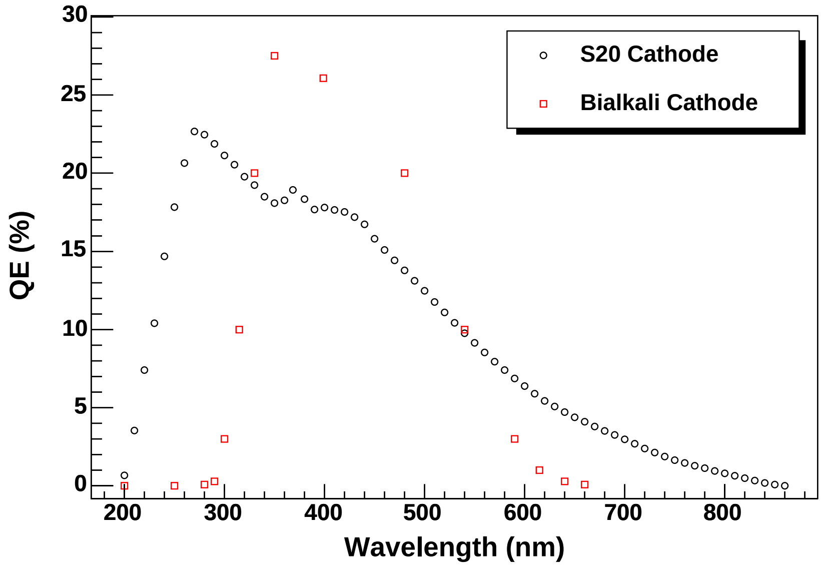


Figure 6.4: Quantum efficiency for the S20 cathode PMT to be used in the experiment [7].

### Detector Wrapping

The millipore paper optical properties are specified in lines 1156 through 1160. The reflectivity is a measured quantity [8] (fig. 6.5) for part of our wavelength range. The values for the remaining range are guessed, based on the assumption that the trend of the curve continues uniformely beyond the plotted range. The significance and meaning of the boundary interface models, surface finish, and other settings are discussed in section 6.2. For some models, the specification of the relative strengths of the type of light scattering is required. Lines 1156 through 1160 set the reflectivity of the wrapping material itself and the strengths of 3 of the 4 scattering types, where the fourth is Lambertian reflection, which is set by GEANT4 such that all four types add up to 1. The actual relative strengths are often not known for materials, so that, in this case, the values are guessed according to what the material "looks" like. For millipore paper, the values for backscatter and specular reflection are set to be 30% compared to 70% for lambertian or diffuse reflection, since the material is known to have a rather dull appearance. It is assumed that all light is reflected in some way or another and the absorption length for the wrapping material is currently not set (meaning that GEANT4 sets it to zero).

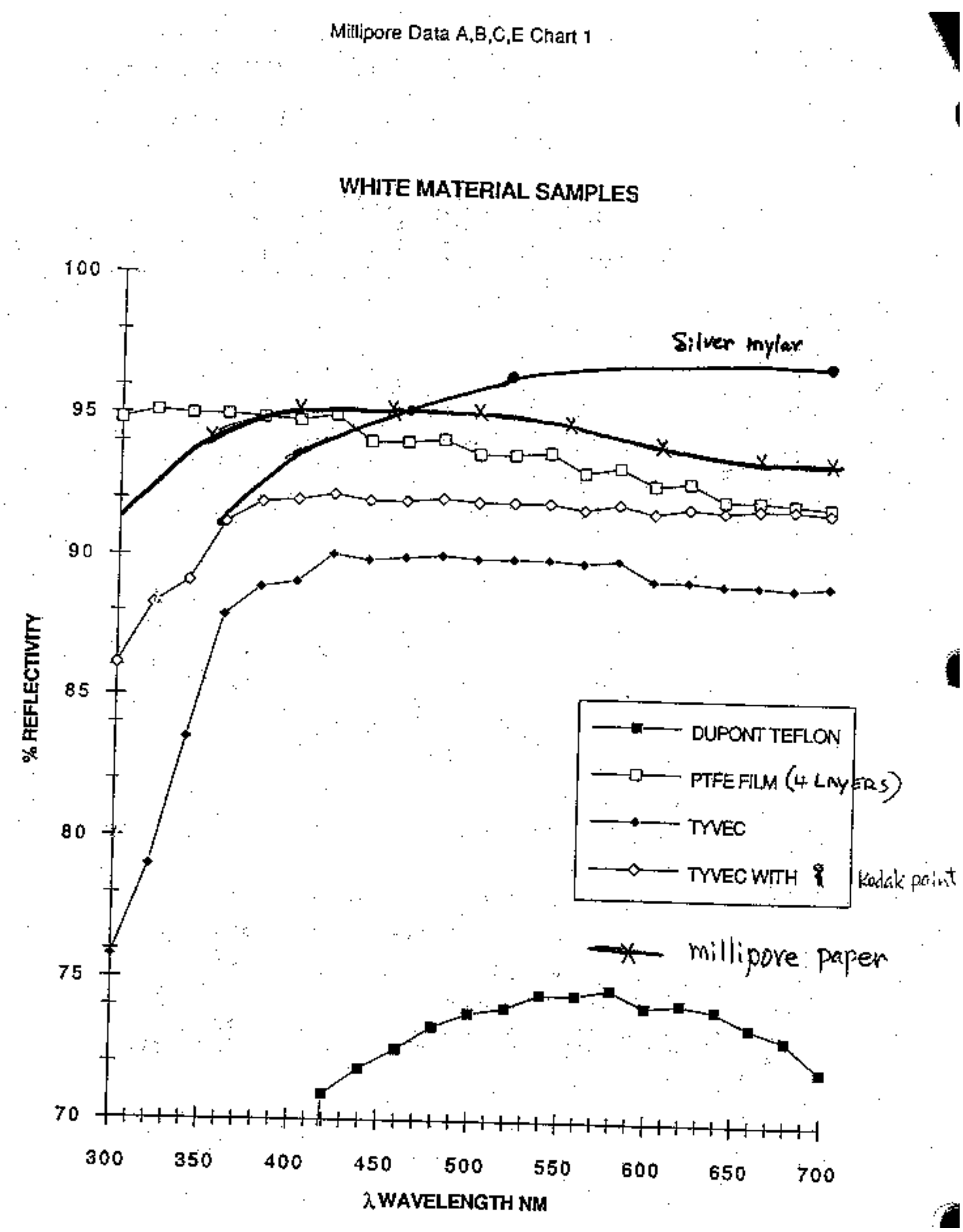


Figure 6.5: Reflectance of various detector wrapping materials [8].

## 6.2 Description of the Geant4 Optical Transport and Transmission Code (*G4OpBoundaryProcess*)

The GEANT4 class *G4OpBoundaryProcess* handles reflection and refraction at optical boundaries between two volumes, taking into account the photon polarization. A detailed trace was done of this class to benchmark the simulation and to be able to correctly implement the optical properties described above, as well as to correctly interpret the simulation output. The generation of Cerenkov light and bulk absorption are handled in classes *G4Cerenkov* and *G4OpAbsorption* respectivley.

A GEANT4 event is split into tracks of the primary and possible secondary particles. Each track is separated into steps, with each individual step size being determined by a particular physics interaction with the shortest associated range in a given material. Please consult the GEANT4 manuals for more information [9]. A step in GEANT4 is defined by its two end points *preStepPoint* and *postStepPoint*. All registered continuous and discrete processes are called for each step. The class *G4OpBoundaryProcess* implements a discrete process and these are handled by the stepping manager class after the step is completed, by calling tha function *PostStepDoIt* in the *G4OpBoundaryProcess* class.

### 6.2.1 Function *PostStepDoIt*

The function begins by checking whether the photon is traversing a boundary (*postStepPoint* lies in the next volume) and if the step size would be large enough to actually push the photon into the next volume and returns without doing anything if either condition is not met (fig. 6.6).

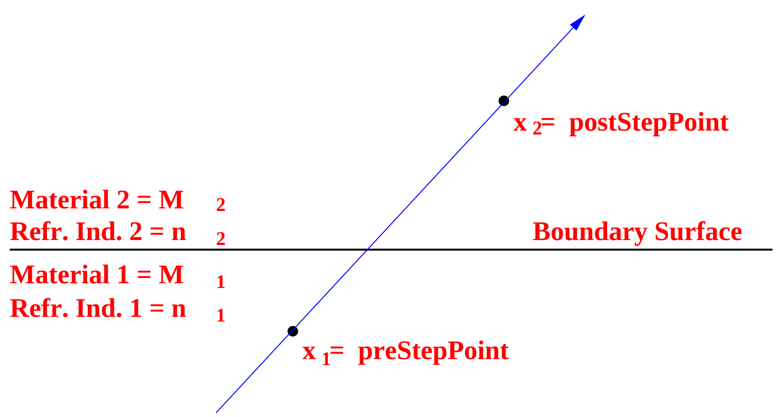


Figure 6.6: Illustration of optical boundary variables.

The variables for the material definitions ($M_1,\ M_2$), indicies of refraction ($n_1, n_2$), old (before the boundary process is invoked) photon momentum ($|\vec{k}_1| \equiv k_1$), direction ($\hat{k}_1$) and polarization ($\hat{h}_1$) are obtained. The index of refraction in material 1 ($n_1$) is obtained according to the momentum of the incident photon. For Cerenkov light, the photon momentum and polarization is determined in the class *G4Cerenkov*.

If the electron momentum direction is taken to lie along the the positve Z axis then the photon momentum lies along a cone with opening angle ($\theta$) such that

$$\sin^2\theta \leq \sin^2\theta_{max},$$

with

$$\cos\theta = \frac{1}{\beta n(f)}$$

and $\sin\theta_{max}$ is set by $\cos\theta_{max} = 1/\beta/n(f_{max})$, where $f_{max}$ is the maximum user specified index of refraction, corresponding to the maximum user specified photon momentum (see class *QweakSimMaterial*, chapter 18, lines 708 - 758, pp. 193- 194). In a general right handed coordinate system, the photon momentum is then simply set to

$$\begin{aligned}
\hat{k_x} &= \sin\theta\cos\phi \\
\hat{k_y} &= \sin\theta\sin\phi \\
\hat{k_x} &= \cos\theta\ .
\end{aligned}$$

The photon polarization is set to be linear and orthogonal to the momentum direction, in accordance with the property of transverse polarization in electromagnetic waves (fig. 6.7).

$$\begin{aligned}
\hat{h_x} &= \cos\theta\cos\phi \\
\hat{h_y} &= \cos\theta\sin\phi \\
\hat{h_x} &= -\sin\theta
\end{aligned}$$

The azimuthal angle is chosen randomly from a uniform distribution, between 0 and $2\pi$.

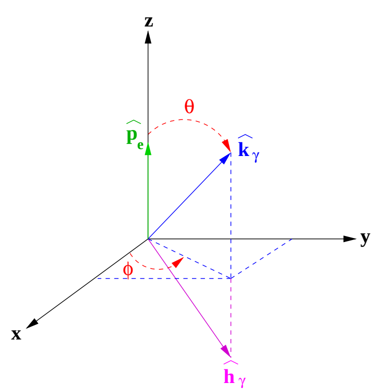


Figure 6.7: Cerenkov photon momentum and polarization with respect to the direction of an incident electron, in Geant4.

Next, a pointer to the logical border surface ($s_1$) between the two volumes containing $x_1$ and $x_2$ is obtained. This is one of the surfaces specified in class *QweakSimCerenkovDetector* (lines 1071 through 1095, pp. 80, 81 and line 1147, p. 82). From this in turn, the optical surface properties are obtained, providing access to the interface type (dielectric-dielectric, dielectric-metal, etc..), the physics treatment model (glisur or unified [10], see below), and the surface finish (polished, ground, *backpainted*, *frontpainted*, see below). The optical surface object also provides access to the material properties as specified on lines 1135 through 1142 (p. 81) and lines 1165 through 1175 (p. 82).

If the model is glisur, only the reflectivity and efficiency (for absorption at the surface, which is not to be confused with the bulk absorption length of the material, set in the class *QweakSimMaterial*, see above) are accessed according to the photon momentum. If they are not specified, then the reflectivity is set to unity and the absorption efficiency is set to zero. The surface absorption efficiency of the quartz is not currently set. If the model is unified, then, in addition to the efficiency and absorption, the particular scattering properties (see section on wrapping above) are accessed as well. If any of the specular reflection types or backscatter are not specified, they are set to zero, which automatically makes all scattering maximally diffuse (lambertian).

In the case of use of the glisur model and if the finish is polished or ground and the interface is between two dielectric materials, the simulation attempts to get the index of refraction of material two ($n_2$) from the surface material properties table. This is done unless material one is identical to material two, in which case no further action is taken (the photon is returned to the stepping manager). If the finish is ground or polished *backpainted* (smooth and rough wrapping respectively, using the unified model – don't blame me, didn't make up those names), the simulation attempts to get the index of refraction of the layer between the surface and the wrapping.

```
NOTE: If the surface material properties table is not given for a
      specified optical surface, and the surface finish is
      polishedbackpainted or groundbackpainted, then GEANT4
      kills the track. I.e. the photon is lost at the boundary.

      The same is true for polished or ground glisur model surface
      specifications if no index of refraction was specified for
      material two.
```

Lines 1100 through 1133 (p. 81) use the glisur model with a polished surface finish requiring the specification of a polish level for the quartz. This surface interface is illustrated in fig. 6.6. Lines 1152 through 1165 (p. 82) implement the detector wrapping and use the unified model with a *groundbackpainted* finish, including the specification of all scattering parameters and the wrapping material roughness parameter ($\sigma_\alpha$). This setup is illustrated in fig. 6.8. If the finish is specified to be *frontpainted* the configuration corresponds to that in fig. 6.8, but without the layer between the optical surface and the wrapping; this then corresponds to more of a coating rather than a wrapping.

Still in the function *PostStepDoIt*, the simulation then obtains the global position ($\vec{x}_g$) of the photon, as given by the *postStepPoint* (see fig. 6.6). The global position vector is then transformed to the local volume coordinate system $\vec{x}_l = \mathbf{T}\vec{x}_g$. The reverse is done for the vector normal to the volume surface which the photon is hitting ($\hat{n}_l$), which is given in the local volume coordinate system. The local normal points away from the volume and the negative ($\hat{n}_l^{\prime} \equiv -\hat{n}_l$) of this vector is transformed to the global coordinate system $\hat{n}_g = \mathbf{T}^{-1}\hat{n}_l^{\prime}$. The transformation matrix ($\mathbf{T}$) transforms around the local normal ($\hat{n}_l^{\prime}$). These transformations are necessary since all Geant4 particle momenta are specified in global coordinates.

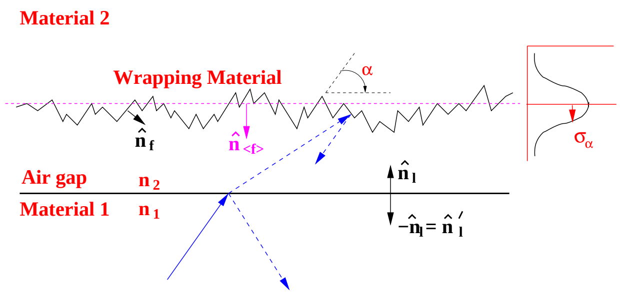


Figure 6.8: Illustration of the material index of refraction for a wrapped detector configuration, inlcuding specification of the wrapping material roughness.

Next, the simulation checks ot make sure that $\hat{k}_1 \cdot \hat{n}_g \leq 0$, and otherwise forces this to be the case by setting $\hat{n}_g \Rightarrow -\hat{n}_g$. Then, if the interface type is dielectric-metal, the function *DielectricMetal* is called. If the interface type is dielectric-dielectric and the surface finish is *frontpainted* (coating) then it is randomly determined of the photon is absorbed or reflected. This is done by uniformly sampling a random variable between zero and one and choosing reflection if the random variable is less than the specified reflectivity and choosing absorption otherwise (See the function *G4BooleanRand* in the *G4OpBoundaryProcess* header file). If the photon is reflected and the finish is *groundfrontpainted* (a rough coating) then the reflection is lambertian. If the interface is dielectric-dielectric and the finish is anything other than *frontpainted* then the function *DielectricDielectric* is called. This is the case we concentrate on here. This function handles the details of reflection, transmission/refraction and absorption at the boundary between two dielectric materials, and includes the implementation of wrapping or coating.

### 6.2.2 Function *DielectricDielectric*

The function *DielectricDielectric* consists of two nested loops that are cycled through as long as the photon has either not made a transmission from the origin volume to another volume nor a valid reflection back into the same volume (inner loop), or as long as the photon continues to be reflected from the wrapping (or the surface of the volume the photon is transmitted to) (outer loop).

The function begins by comupting the so called facet normal $\hat{n}_f$ (see fig. 6.8) if the surface finish is *ground* or *groundbackpainted*. Otherwise the facet normal is set to be equal to the global normal $\hat{n}_g \equiv \hat{n}_{\langle f \rangle}$, which is assumed to be parallel with the surface normal of volume 1, containing the incident photon. The facet normal can either correspond to the normal of a small area on the wrapping material or to the normal of a facet on the rough surface of volume 1.

The facet normal is obtained as follows:

If the surface model is not unified (i.e. glisur) then the function obtains the polish of the volume 1 surface (note the volumes 1 and 2 pointers passed to the *G4LogicalBorderSurface* constructor starting on line 1071 p. 80). If the polish is 1 (100%) then the facet normal is set to be equal to the global normal ($\hat{n}_f = \hat{n}_g$). If the polish smaller than 1, then the following procedure is used to randomly find the facet normal:

1. establish a vector $\vec{s}$, such that
   (a) $-1 \leq s_x \leq 1$ randomly chosen with uniform distribution
   (b) $-1 \leq s_y \leq 1$ randomly chosen with uniform distribution
   (c) $-1 \leq s_z \leq 1$ randomly chosen with uniform distribution

   with the condition that $|\vec{s}| \leq 1$
2. set $\vec{s} = (1 - \text{Polish})\vec{s}$
3. set $\vec{n}_f = \hat{n}_g + \vec{s}$
4. repeat this process until $\vec{k}_1 \cdot \vec{n}_f < 0$
5. renormalize $\vec{n}_f \Rightarrow \hat{n}_f$

If the model is unified, the following procedure is used:

1. The process obtains $\sigma_\alpha$ from the optical surface.
   This parameter specifies the surface roughness of the reflector (wrapping). See fig 6.8. $\alpha$ is the angle a facet surface makes with respect to the average surface. $\alpha$ is randomly choosen repeatedly, from a gaussian distribution with standard deviation $\sigma_\alpha$, until the following conditions are met:
   (a) $f_m \times r \leq \sin \alpha$ or $\alpha < \pi/2$,
   where $r$ is a random variable between 0 and 1, with uniform distribution and,
   (b) $f_m = 4\sigma_\alpha$ as long as $4\sigma_\alpha < 1.0$ and $f_m = 1.0$ otherwise.

2. A unit vector ($\hat{u}$) is established from
   $u_x = \sin\alpha\cos\phi$
   $u_y = \sin\alpha\sin\phi$
   $u_z = \cos\alpha$
   where phi is randomly chosen between 0 and $2\pi$, with uniform distribution.

3. $\hat{n}_f \equiv \hat{u}$

4. this process is repeated until $\vec{k}_1 \cdot \vec{n}_f < 0$

The simulation then determines whether the photon undergoes total internal reflection, refraction or is absorbed at the boundary. The program calculates

$$\begin{aligned} \hat{k}_1 \cdot \hat{n}_f &= \cos\theta_{k_1} \quad \text{(Note that this is negative)} \\ \hat{h}_1 \cdot \hat{n}_f &= \cos\theta_{h_1} \end{aligned}$$

If the incident angle is oblique ($-\cos\theta_{k_1} < 1.0$), the simulation code calculates the outgoing photon angle (fig. 6.9)

$$\sin\theta_{k_2} = \frac{n_1}{n_2}\sin\theta_{k_1}$$

in the standard way, according to Snell's Law. Note that, if the surface finish was specified to be *backpainted*, then $n_2$ is the index of refraction for the material layer between volume 1 and the wrapping (see fig. 6.8). If the photon is normally incident ($-\cos\theta_{k_1} \geq 1.0$) the simulation sets $\sin\theta_{k_1} = \sin\theta_{k_2} = 0$.

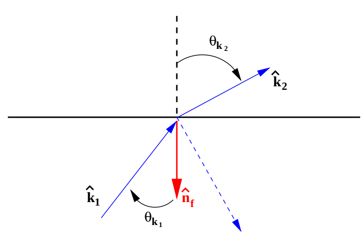


Figure 6.9: Illustration of standard reflection and refraction at the boundary and the variables used here.

The program continues to go through the simulation for total internal reflection if $\sin\theta_{k_2} \geq 1$ and otherwise goes on to simulate transmission/refraction and reflection, including polarization.

### Total Internal Reflection

For total internal reflection, if the model is unified and the finish is not polished, the program determines a specific reflection type from the photon wavelength dependend parameters, specified by the user, specular spike ($\rho_{ss}$), specular lobe ($\rho_{sl}$), backscatter ($\rho_{bs}$). Together with the parameter for lambertian reflection, which is set automatically, these parameters have to add to unity. A uniformely distributed random variable ($r$) is then sampled between 0 and 1, and the reflection type is determined as

1. specular spike reflection if: $0 \leq r < \rho_{ss}$ (In this case $\hat{n}_f = \hat{n}_g$)
2. specular lobe reflection if : $\rho_{ss} \leq r \leq \rho_{ss} + \rho_{sl}$
3. backscatter if : $\rho_{ss} + \rho_{sl} < r < \rho_{ss} + \rho_{sl} + \rho_{bs}$ and
4. Lambertian reflection if : $\rho_{ss} + \rho_{sl} + \rho_{bs} \leq r \leq 1$ .

If the reflection is Lambertian (diffuse), a new momentum is randomly chosen around the global normal ($\hat{n}_g$), such that it lies in the same hemisphere as the global normal. The facet normal is then calculated from $\hat{n}_f = (\vec{k}_2 - \vec{k}_1)/(|\vec{k}_2 - \vec{k}_1|)$ and the new polarization is calculated from $\hat{h}_2 = -\hat{h}_1 + 2(\hat{h}_1 \cdot \hat{n}_f)\hat{n}_f$. This just sets $\vec{E}_{new} = -\vec{E}_{old}$ ($\vec{E} \equiv \vec{h}$), making sure that this happens around the facet normal and in the plane of incidence. For total internal reflection, this satisfies the standard boundary conditions (see, for example, [11, 12])

$$\begin{aligned} \epsilon_1 \left( E_I^{\perp} + E_R^{\perp} \right) &= \epsilon_2 E_T^{\perp} \\ E_I^{\parallel} + E_R^{\parallel} &= E_T^{\parallel} \ . \end{aligned}$$

If the photon is backscattered, then $\hat{k}_2 = -\hat{k}_1$ and $\hat{h}_2 = -\hat{h}_1$. If the photon is neither backscattered nor Lambertian reflected (i.e. the reflection is specular lobe or specular spike), then

$$\begin{aligned} \hat{k}_2 &= \hat{k}_1 - 2(\hat{k}_1 \cdot \hat{n}_f)\hat{n}_f \\ \hat{h}_2 &= -\hat{h}_1 + 2(\hat{h}_1 \cdot \hat{n}_f)\hat{n}_f \ . \end{aligned}$$

For total internal reflection, the function *DielectricDielectric* then exits with $\hat{k}_2 \cdot \hat{n}_g \geq 0$ and $\hat{k}_1 = \hat{k}_2$, $\hat{h}_1 = \hat{h}_2$, returning the new momentum and polarization after the reflection to the stepping manager which continues to step the photon through the volume, until it encounters another boundary or is killed by some other process.

**Calculation of Transmission Amplitude**

Here, the simulation determines whether a photon is reflected or transmitted, based on the size of the calculated transmission coefficient.

If $\sin\theta_{k_2} < 1$, then the simulation calculates

$$\begin{aligned}\cos\theta_{k_2} &= \sqrt{1-\sin^2\theta_{k_2}} \quad \text{for } \cos\theta_{k_1} > 0\\ \cos\theta_{k_2} &= -\sqrt{1-\sin^2\theta_{k_2}} \quad \text{for } \cos\theta_{k_1} \leq 0\end{aligned}$$

For oblique angles of incidence ($\sin\theta_{k_1} > 0$), the parallel and perpendicular components of polarization are calculated as follows.

1. Determine a unit vector perpendicular to the plane of incidence:
   $\hat{A}_\perp = \frac{\vec{k}_1 \times \hat{n}_f}{|\vec{k}_1 \times \hat{n}_f|}$
   If, in figure 6.9, $\hat{k}_1$, $\hat{k}_1$, and $\hat{n}_f$ are all in the plane of the page, then $\hat{A}_\perp$ is pointing out of the page.

2. The perpendicular component of $\vec{E}$ is then calculated to be
   $\vec{E}_I^\perp = \left(\hat{A}_\perp \cdot \vec{h}_1\right)\hat{A}_\perp$

3. and the parallel component is calculated to be
   $\vec{E}_I^\parallel = \vec{E}_I - \vec{E}_I^\perp = \vec{h}_1 - \vec{E}_I^\perp$

If $\sin\theta_{k_1} \leq 0$, then the program sets $\vec{A}_\perp = \vec{h}_1$ and, in accordance with the conventions used in [11], $E_I^\perp \equiv |\vec{E}_I^\perp| = 0.0$ and $E_I^\parallel \equiv |\vec{E}_I^\parallel| = 1.0$. The program then proceeds to calculate [12]

$$\begin{aligned}E_T^\perp &= 2E_I^\perp n_1\cos\theta_{k_1}\frac{1}{n_1\cos\theta_{k_1}+n_2\cos\theta_{k_2}}\\ E_T^\parallel &= 2E_I^\parallel n_1\cos\theta_{k_1}\frac{1}{n_2\cos\theta_{k_1}+n_1\cos\theta_{k_2}}\\ E_T^2 &= (E_T^\perp)^2+(E_T^\parallel)^2\ .\end{aligned}$$

If $\cos\theta_{k_1}! = 0$, then the transmission coefficient is calulated as

$$T = \frac{n_2\cos\theta_{k_2}E_T^2}{n_1\cos\theta_{k_1}}$$

, otherwise $T = 0$.

The transmission coefficient is a number between 0 and 1. To determine whether the photon is transmitted or reflected, the program samples a random variable with uniform distribution between 0 and 1. If the variable is less than the transmission coefficient, then the program proceeds to simulate transmission. Otherwise the photon is reflected back into the volume, with new polarization.

In all of this, it is important to keep in mind that each ray corresponds to a single photon which can only be either reflected, transmitted, or absorbed.

**Reflection**

The reflection here is handled exactly as it was done for total internal reflection, except that if we have neither Lambertian reflection nor backscatter (which in this case means that we have Fresnel reflection), we calculate the new polarization as follows.

For oblique angles, the program obtains:

1.
$$\begin{aligned} E_R^{\parallel} &= E_T^{\parallel}\frac{n_2}{n_1} - E_I^{\parallel} \\ &= E_I^{\parallel}\left(\frac{n_2\cos\theta_{k_1} - n_1\cos\theta_{k_2}}{n_2\cos\theta_{k_1} + n_1\cos\theta_{k_2}}\right) \end{aligned}$$

2.
$$\begin{aligned} E_R^{\perp} &= E_T^{\perp} - E_I^{\perp} \\ &= E_I^{\perp}\left(\frac{n_2\cos\theta_{k_1} - n_2\cos\theta_{k_2}}{n_1\cos\theta_{k_1} + n_2\cos\theta_{k_2}}\right) \end{aligned}$$

3.
$$E_R^2 = (E_R^{\perp})^2 + (E_R^{\parallel})^2$$

4. A unit vector parallel to the plane of incidence is determined

$$\hat{A}_{\parallel} = \frac{\hat{k}_2 \times \hat{A}_{\perp}}{|\hat{k}_2 \times \hat{A}_{\perp}|}$$

which, in figure 6.9, lies in the plane of the page, pointing into the neighboring volume.

5. The new polarization is then calculated as

$$\vec{h}_2 = \frac{E_R^{\parallel}}{E_R}\hat{A}_{\parallel} + \frac{E_R^{\perp}}{E_R}\hat{A}_{\perp} \tag{6.2}$$

If the photon incident angle is not oblique then, if $n_2 > n_1$, the program sets $\vec{h}_2 = -\vec{h}_1$ and $\vec{h}_2 = \vec{h}_1$, if $n_2 \leq n_1$.

**Transmission**

For oblique angles of incidence, the program calculates the new momentum from

$$\vec{k}_2 = \vec{k}_1 + \left(\cos\theta_{k_1} - \frac{n_2}{n_1}\cos\theta_{k_2}\right)\hat{n}_f$$

To verify this, in [12], we are given that

$$\begin{aligned} n_1(\hat{k}_1 \times \hat{n}_f) &= n_2(\hat{k}_2 \times \hat{n}_f) \\ &\Rightarrow \\ n_2\hat{k}_2 - n_1\hat{k}_1 &= -\left(n_2\cos\theta_{k_2} - n_1\cos\theta_{k_1}\right)\hat{n}_f \\ &\Rightarrow \\ \frac{n_2}{n_1}\hat{k}_2 &= \hat{k}_1 + \left(\cos\theta_{k_1} - \frac{n_2}{n_1}\cos\theta_{k_2}\right)\hat{n}_f \ . \end{aligned}$$

Where the extra negative sign on the right hand side of the second line is a result of the normal vector having the opposite sign, as compared to the one used in [12]. So, the above treatment is correct, since the transmitted momentum vector is renormalized, removing the ratio of refraction indicies (note that there is no change in photon wavelength/energy across the boundary). Then, as was done above, a unit vector parallel to the plane of incidence is computed with the new momentum and the new polarization is calculated as shown in eqn. 6.2.

If the angle of incidence is not oblique, then the program sets $\vec{k}_2 = \vec{k}_1$ and $\vec{h}_2 = \vec{h}_1$.

The simulation for the optical boundary interaction is completed, if the following conditions are satisfied:

1. If the photon was Fresnel Refracted and

$$\vec{k}_2 \cdot \hat{n}_g \leq 0$$

   or

2. if the photon was reflected and

$$\vec{k}_2 \cdot \hat{n}_g \geq 0$$

and

3. the boolean variable *Inside* is false and/or the boolean variable *Swap* is true.

Otherwise, the program begins the boundary process again. The variable *Inside* is toggled each time the photon is transmitted from one volume to another. The material properties are defined for an "inside" volume (the photon is inside volume 2, after transmission) and "outside" volumes (the photon is inside volume 1, before transmission). If prior to transmission, the outside volume had properties $(n_1, M_1)$ and the inside volume had properties $(n_2, M_2)$ , then after transmission the outside volume has properties $(n_2, M_2)$ and the inside volume as properties $(n_1, M_1)$. This is achieved by simply reassigning the values for the two volumes defined internal to the *G4OpBoundaryProcess* class and has no effect on the global propeties of any defined volume within the Geant4 simulation. The boolean variable *Swap* is toggeld when this reassignment takes place.

If the photon was transmitted (*Inside* is true) and the volumes where not yet swapped (*Swap* is false), the program enters another loop if the finsih is *backpainted* and the photon is not absorbed. Here, if the photon is Fresnel refracted, the simulation reverses the sign of the global normal and does another reflection (this would be from the wrapping). Otherwise, the volumes are swapped as described above. In any case, if the two boolean variables ahve the correct setting above and the finish is *backpainted*, then the boundary process starts all over again, this time going back toward volume of origin.

## 6.3 Event Data and Readout

As described in chapter 14, the event readout is implemented using several event classes that define the event ROOT tree that is stored in a ROOT file at the end of a run. The point of departure in this chapter is the class *QweakSimUserCerenkov_MainEvent*. At the moment, the only thing this class does, is to create the data containers for each of the eight Cerenkov detectors in each octant (*QweakSimUserCerenkov_OctantEvent*). This class then, in turn, creates the two additional data containers *QweakSimUserCerenkov_DetectorEvent* and *QweakSimUserCerenkov_PMTEvent*, which hold the actual data members to be filled.

### 6.3.1 Class *QweakSimUserCerenkov_DetectorEvent*

Pages 94 through 98 show the definitions of the data members for the detector hits. For the most part, the data member names should be self explanatory. However, a few comments are in order: The origin vertex refers to the origin of the track that was registered as a detector hit, which could be either a primary electron or any seondary particle. The secondary particle information *SecPartLocal...* is collected for each step in the class *QweakSimEventAction* (chapter 17, lines 83-104, p. 176) and stored in the class *QweakSimUserInformation* in an array which has a size equal to the number of secondary particles that were detected. This includes the collection of all secondary particles that were created during a step, for all steps along the primary particles track, which hit the main Cerenkov detector. The same thing is done for the collection of the energy deposit (line 85, p. 176), and the number and energy of optical photons that were created inside the quartz, for each step (lines 64-73, pp. **??** and 176).

```
#ifndef QweakSimCerenkovDetector_h
#define QweakSimCerenkovDetector_h

// system includes
#include "cpp_include.h"
#include "Root_include.h"
#include "Geant4_include.hh"

// user includes
#include "QweakSimCerenkovDetectorMessenger.hh"
#include "QweakSimCerenkov_DetectorSD.hh"
#include "QweakSimCerenkovDetector_PMTSD.hh"
// #include "QweakSimPMTEntranceWindowSD.hh"
#include "QweakSimMaterial.hh"
#include "QweakSimUserInformation.hh"
// user classes
class QweakSimCerenkovDetectorMessenger;
//class QweakSimMaterial;

class QweakSimCerenkovDetector
{
public:
  QweakSimCerenkovDetector(QweakSimUserInformation*);
  ~QweakSimCerenkovDetector();

  void SetMotherVolume(G4VPhysicalVolume* mv) {theMotherPV = mv;}
  void PlacePVCerenkovMasterContainer();

  void ConstructComponent(G4VPhysicalVolume* MotherVolume);
  void DefineCerenkovGeometry();
  void DestroyComponent();
  void SetCerenkovDetectorMaterial(G4String materialName);

  void SetCerenkovDetectorCenterPositionInX(G4double xPos);
  void SetCerenkovDetectorCenterPositionInY(G4double yPos);
  void SetCerenkovDetectorCenterPositionInZ(G4double zPos) ;
  void SetCerenkovDetectorTiltAngle(G4double tiltangle);
  void SetCerenkovDetectorThickness(G4double thickness);

  void CerenkovGeometryPVUpdate();

  G4LogicalVolume*   GetCerenkovDetector_LogicalVolume()    {return ActiveArea_Logical;}
  G4VPhysicalVolume* GetCerenkovDetector_PhysicalVolume()   {return ActiveArea_Physical;}

  G4LogicalVolume*   GetPMT_LogicalVolume()    {return Cathode_Logical;}
  G4VPhysicalVolume* GetPMT_PhysicalVolume()   {return Cathode_Physical;}

private:

  QweakSimUserInformation *myUserInfo;

  QweakSimCerenkovDetectorMessenger* CerenkovDetectorMessenger;  // pointer to the Messenger

  QweakSimMaterial*  pMaterial;

  G4VPhysicalVolume* theMotherPV;

 // needed for manual coil placement
   std::vector< G4VPhysicalVolume* > CerenkovMasterContainer_Physical;
   std::vector< G4double >           AnglePhi_CerenkovMasterContainer;
   std::vector< G4ThreeVector >      Translation_CerenkovMasterContainer;
   std::vector< G4RotationMatrix* >  Rotation_CerenkovMasterContainer;

  G4LogicalVolume*   CerenkovContainer_Logical;
  G4VPhysicalVolume* CerenkovContainer_Physical;
  G4Material*        CerenkovContainer_Material;

```

Figure 6.10: Header File

```
  G4LogicalVolume*   ActiveArea_Logical;
  G4VPhysicalVolume* ActiveArea_Physical;
  G4Material*        ActiveArea_Material;

  G4LogicalVolume*   QuartzBar_LogicalLeft;
  G4LogicalVolume*   QuartzBar_LogicalRight;
  G4VPhysicalVolume* QuartzBar_PhysicalLeft;
  G4VPhysicalVolume* QuartzBar_PhysicalRight;
  G4Material*        QuartzBar_Material;

  G4LogicalVolume*   LightGuide_LogicalLeft;
  G4LogicalVolume*   LightGuide_LogicalRight;
  G4VPhysicalVolume* LightGuide_PhysicalLeft;
  G4VPhysicalVolume* LightGuide_PhysicalRight;
  G4Material*        LightGuide_Material;

  G4LogicalVolume*   QuartzGlue_Logical;
  G4VPhysicalVolume* QuartzGlue_PhysicalLeft;
  G4VPhysicalVolume* QuartzGlue_PhysicalCenter;
  G4VPhysicalVolume* QuartzGlue_PhysicalRight;
  G4Material*        QuartzGlue_Material;

  G4LogicalVolume*   Radiator_Logical;
  G4VPhysicalVolume* Radiator_Physical;
  G4Material*        Radiator_Material;

  G4LogicalVolume*   PMTContainer_Logical;
  G4VPhysicalVolume* PMTContainer_PhysicalLeft;
  G4VPhysicalVolume* PMTContainer_PhysicalRight;
  G4Material*        PMTContainer_Material;

  G4LogicalVolume*   PMTQuartzOpticalFilm_Logical;
  G4VPhysicalVolume* PMTQuartzOpticalFilm_Physical;
  G4Material*        PMTQuartzOpticalFilm_Material;

  G4LogicalVolume*   PMTEntranceWindow_Logical;
  G4VPhysicalVolume* PMTEntranceWindow_Physical;
  G4Material*        PMTEntranceWindow_Material;

  G4LogicalVolume*   Cathode_Logical;
  G4VPhysicalVolume* Cathode_Physical;
  G4Material*        Cathode_Material;

//   G4Box           *QuartzBar_Solid;
//   G4Box           *PMTEntranceWindow_Solid;
//   G4Box           *PMT_Solid;
//   G4Box           *PMTContainer_Solid;

  G4double Container_FullLength_X;
  G4double Container_FullLength_Y;
  G4double Container_FullLength_Z;

  G4double ActiveArea_FullLength_X;
  G4double ActiveArea_FullLength_Y;
  G4double ActiveArea_FullLength_Z;

  G4double GlueFilm_FullLength_X;
  G4double GlueFilm_FullLength_Y;
  G4double GlueFilm_FullLength_Z;

  G4double Chamfer_FullLength;
  G4double Chamfer_FullHeight;
  G4double Chamfer_FullThickness;

  G4double QuartzBar_FullLength;
  G4double QuartzBar_FullHeight;
  G4double QuartzBar_FullThickness;
```

Figure 6.11: Header File

```
  G4double LightGuide_FullLength;
  G4double LightGuide_FullWidth1;
  G4double LightGuide_FullWidth2;
  G4double LightGuide_FullThickness;

  G4double PMTContainer_Diameter;
  G4double PMTContainer_FullLength_Z;

  G4double PMTQuartzOpticalFilm_Diameter;
  G4double PMTQuartzOpticalFilm_Thickness;

  G4double PMTEntranceWindow_Diameter;
  G4double PMTEntranceWindow_Thickness;

  G4double Cathode_Diameter;
  G4double Cathode_Thickness;

  G4double CerenkovDetectorCenterZPosition;

  G4double LGAngCutXDim;
  G4double LGAngCutYDim;
  G4double LGAngCutZDim;


 // needed for boolean union
  std::vector< G4SubtractionSolid* >  RightQuartz_Solid;
  std::vector< G4SubtractionSolid* >  LeftQuartz_Solid;
  std::vector< G4SubtractionSolid* >  LeftGuide_Solid;
  std::vector< G4SubtractionSolid* >  RightGuide_Solid;

//   std::vector< G4Box* >  mirror_solid;
  G4Material* mirror_material;
  std::vector< G4LogicalVolume* > mirror_logical;
  std::vector< G4VPhysicalVolume* > mirror_physical;

  G4double Tilting_Angle;     // total tilting angle towards mean track
  G4double Kink_Angle;        // Vshape angle
  G4double Thickness;


  // placing the container
  G4ThreeVector    Position_CerenkovContainer;
  G4RotationMatrix* Rotation_CerenkovContainer;

  // placing the chamfers
  G4ThreeVector    Position_Chamfer1;
  G4RotationMatrix Rotation_Chamfer1;
  G4ThreeVector    Position_Chamfer2;
  G4RotationMatrix Rotation_Chamfer2;
  G4ThreeVector    Position_Chamfer3;
  G4RotationMatrix Rotation_Chamfer3;
  G4ThreeVector    Position_Chamfer4;
  G4RotationMatrix Rotation_Chamfer4;

  G4ThreeVector    Position_AngCut1;
  G4RotationMatrix Rotation_AngCut1;

  G4ThreeVector    Position_AngCut2;
  G4RotationMatrix Rotation_AngCut2;

  G4ThreeVector    Position_LGRight;
  G4RotationMatrix Rotation_LGRight;
  G4ThreeVector    Position_LGLeft;
  G4RotationMatrix Rotation_LGLeft;

  G4ThreeVector    Position_LGFaceMirrorLeft;
```

Figure 6.12: Header File

```
  G4RotationMatrix Rotation_LGFaceMirrorLeft;

  G4ThreeVector    Position_LGEdgeMirrorLeft;
  G4RotationMatrix Rotation_LGEdgeMirrorLeft;

  G4ThreeVector    Position_LGFaceMirrorRight;
  G4RotationMatrix Rotation_LGFaceMirrorRight;

  G4ThreeVector    Position_LGEdgeMirrorRight;
  G4RotationMatrix Rotation_LGEdgeMirrorRight;

  // placing the left SingleBar
  G4ThreeVector    Translation_SingleBarLeft;
  G4RotationMatrix Rotation_SingleBarLeft;

  // placing the right SingleBar
  G4ThreeVector    Translation_SingleBarRight;
  G4RotationMatrix Rotation_SingleBarRight;

  // placing the left PMTContainer
  G4ThreeVector     Translation_PMTContainerLeft;
  G4RotationMatrix Rotation_PMTContainerLeft;

  // placing the right PMTContainer
  G4ThreeVector     Translation_PMTContainerRight;
  G4RotationMatrix Rotation_PMTContainerRight;

  // placing the  PMT
  G4ThreeVector    Translation_Cathode;
  G4RotationMatrix Rotation_Cathode;

  // placing the  PMTEntranceWindow
  G4ThreeVector    Translation_PMTQuartzOpticalFilm;
  G4RotationMatrix Rotation_PMTQuartzOpticalFilm;

  G4ThreeVector    Translation_PMTEntranceWindow;
  G4RotationMatrix Rotation_PMTEntranceWindow;


 // pointer to the sensitive detector
  G4VSensitiveDetector* CerenkovDetectorSD;
  G4VSensitiveDetector* CerenkovDetector_PMTSD;

  G4double Position_CerenkovContainer_X;
  G4double Position_CerenkovContainer_Y;
  G4double Position_CerenkovContainer_Z;


};
#endif

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

Figure 6.13: Header File

```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;

  CerenkovMasterContainer_Logical  = NULL;
  CerenkovMasterContainer_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;

  QuartzBar_LogicalLeft      = NULL;
  QuartzBar_LogicalRight     = NULL;
  QuartzBar_PhysicalLeft     = NULL;
  QuartzBar_PhysicalRight    = NULL;
  QuartzBar_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;

  PMTQuartzOpticalFilm_Logical  = NULL;
  PMTQuartzOpticalFilm_Physical = NULL;
  PMTQuartzOpticalFilm_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
```

Figure 6.14: Source File

```
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");
CerenkovMasterContainer_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*mm;
ActiveArea_FullLength_Y   =   QuartzBar_FullHeight + 1.0*mm;
ActiveArea_FullLength_Z   =   QuartzBar_FullThickness + 40.0*mm;

Container_FullLength_X    = 2.0*(LightGuide_FullLength + QuartzBar_FullLength +GlueFilm_FullLength_X) + GlueFilm_FullLength_X + 2.0*mm; //ActiveArea_FullLength_X + 20.0*cm;
Container_FullLength_Y    = QuartzBar_FullHeight +  4.0*cm;
Container_FullLength_Z    = QuartzBar_FullHeight + 10.0*cm;

```

Figure 6.15: Source File

```
  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;

  Position_CerenkovMasterContainer_X =   0.0*cm;
  Position_CerenkovMasterContainer_Y = 319.0*cm; // given by SolidWorks (or later by Juliette)
  Position_CerenkovMasterContainer_Z = 570.0*cm; // given by SolidWorks (or later by Juliette)

}

//....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**********************************************

  G4Box* CerenkovMasterContainer_Solid  = new G4Box("CerenkovMasterContainer_Solid",
                                      0.5 * Container_FullLength_X + 1.0*cm,    // half X length required by Geant4
                                      0.5 * Container_FullLength_Y + 1.0*cm,    // half Y length required by Geant4
                                      0.5 * Container_FullLength_Z + 1.0*cm);   // half Z length required by Geant4

  CerenkovMasterContainer_Logical  = new G4LogicalVolume(CerenkovMasterContainer_Solid,
                                          CerenkovMasterContainer_Material,
                                          "CerenkovMasterContainer_Logical",
                                          0,0,0);
```

Figure 6.16: Source File

```


 Position_CerenkovContainer  = G4ThreeVector(0,0,0);

 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(0,Position_CerenkovContainer,
                                  CerenkovContainer_Logical,
                                  "CerenkovContainer_Physical",
                                  CerenkovMasterContainer_Logical,
                                  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
```

Figure 6.17: Source File

```
                    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,
```

Figure 6.18: Source File

```
                              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);

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

Figure 6.19: Source File

```

//****************************************************************************************************
//******************************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);
```

Figure 6.20: Source File

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

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

  QuartzGlue_PhysicalLeft  = new G4PVPlacement(0,Position_LeftGlueFilm,
                               QuartzGlue_Logical,
                               "QuartzGlue_PhysicalLeft",
                               ActiveArea_Logical,
                               false,1);
```

Figure 6.21: Source File

```

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

//*****************************************************************************************************
//*****************************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);

```

Figure 6.22: Source File

```

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,
```

Figure 6.23: Source File

```
                                   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",
                                  CerenkovContainer_Logical,
//                                   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",
                                  CerenkovContainer_Logical,
```

Figure 6.24: Source File

```
//                                          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);

```

Figure 6.25: Source File

```
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",
                           CerenkovContainer_Logical,
//                                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",
                           CerenkovContainer_Logical,
//                                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);

```

Figure 6.26: Source File

```
//   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 = 1.0*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
  //-----------------------------------------------------------------------------

```

Figure 6.27: Source File

```
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) + PMTQuartzOpticalFilm_Thickness;
// 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",
                                  CerenkovContainer_Logical,
//                                     ActiveArea_Logical,
                                  false,
                                  0); // copy number for left PMTContainer
// right side
PMTContainer_PhysicalRight = new G4PVPlacement(Transform3D_PMTContainerRight,
                                 PMTContainer_Logical,
                                 "PMTContainer_Physical",
                                  CerenkovContainer_Logical,
//                                    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,
```

Figure 6.28: Source File

```
                              "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





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



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


G4ThreeVector Position_CerenkovMasterContainer  = G4ThreeVector(Position_CerenkovMasterContainer_X,
                                      Position_CerenkovMasterContainer_Y,
                                      Position_CerenkovMasterContainer_Z);

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

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

Figure 6.29: Source File

```

//==================================================================
// 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_Bord
```

Figure 6.30: Source File

```
erSurface",
                                           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");

```

Figure 6.31: Source File

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

  ActiveArea_OpticalSurface->SetType(dielectric_dielectric);
  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]      = {0.94,0.94,0.945,0.95,0.95,0.91,0.85,0.80,0.80};      //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.);
```

Figure 6.32: Source File

```
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);
CerenkovMasterContainer_Logical->SetVisAttributes(CerenkovContainerVisAtt);
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);
```

Figure 6.33: Source File

```
   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_CerenkovMasterContainer_X =xPos;

    CerenkovGeometryPVUpdate();

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

Figure 6.34: Source File

```
}

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

    Position_CerenkovMasterContainer_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_CerenkovMasterContainer_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
```

Figure 6.35: Source File

```
    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_CerenkovMasterContainer_X);
   centerVector->setY(Position_CerenkovMasterContainer_Y);
   centerVector->setZ(Position_CerenkovMasterContainer_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",
                                                            CerenkovMasterContainer_Logical,
                                                            theMotherPV,
                                                            false,
                                                            n);

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

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

```

Figure 6.36: Source File

```
//=============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
/**

   \file QweakSimUserCerenkov_MainEvent.hh
   $Revision: 1.2 $
   $Date: 2005/12/27 19:28:38 $
   \author Klaus Hans Grimm

*/
//=============================================================================
//
//=============================================================================
//
//   ---------------------------
//  | Doxygen Class Information |
//  ---------------------------
/**
   \class QweakSimUserCerenkov_MainEvent

   \brief ROOT Subtree structure for Cerenkov MainEvent

   Placeholder for a long explaination

 */
//=============================================================================
//
//=============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:28:38 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//=============================================================================

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
#ifndef QweakSimUserCerenkov_MainEvent_h
#define QweakSimUserCerenkov_MainEvent_h
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

// system includes
#include "cpp_include.h"
#include "Root_include.h"

#ifndef __CINT__
#include "Geant4_include.hh"
#endif

// user include
#include "QweakSimUserCerenkov_OctantEvent.hh"

// user classes
class QweakSimUserCerenkov_OctantEvent;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
//class QweakSimUserMainEvent : public TObject
```

Figure 6.37: Header File

```
class QweakSimUserCerenkov_MainEvent
{

private:

public:

     vector <QweakSimUserCerenkov_OctantEvent> Octant;

public:

  // Constructor
  QweakSimUserCerenkov_MainEvent();
  // Destructor
 virtual ~QweakSimUserCerenkov_MainEvent();

  // define a new Class known to ROOT
  ClassDef(QweakSimUserCerenkov_MainEvent,1)

};  // end class QweakSimCerenkov_MainEvent

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

#endif

//================================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_MainEvent.hh,v $
//      Revision 1.2  2005/12/27 19:28:38  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 6.38: Header File

```
//============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
//
/**

   \file QweakSimUserCerenkov_MainEvent.cc

   $Revision: 1.2 $
   $Date: 2005/12/27 19:16:49 $

   \author Klaus Hans Grimm

*/
//============================================================================

//============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:16:49 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//===========================================================================


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

#include "QweakSimUserCerenkov_MainEvent.hh"

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

ClassImp(QweakSimUserCerenkov_MainEvent)

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserCerenkov_MainEvent::QweakSimUserCerenkov_MainEvent()
{
 Octant.clear();
 Octant.resize(8);
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserCerenkov_MainEvent::~QweakSimUserCerenkov_MainEvent()
{;}

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

//=======================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_MainEvent.cc,v $
```

Figure 6.39: Source File

```
//      Revision 1.2  2005/12/27 19:16:49  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 6.40: Source File

```
//=============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
/**

   \file QweakSimUserCerenkov_OctantEvent.hh
   $Revision: 1.2 $
   $Date: 2005/12/27 19:28:38 $
   \author Klaus Hans Grimm

*/
//=============================================================================
//
//=============================================================================
//
//   ---------------------------
//  | Doxygen Class Information |
//  ---------------------------
/**
   \class QweakSimUserCerenkov_OctantEvent

   \brief ROOT Subtree structure for Cerenkov OctantEvent

   Placeholder for a long explaination

 */
//=============================================================================
//
//=============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:28:38 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//=============================================================================

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
#ifndef QweakSimUserCerenkov_OctantEvent_h
#define QweakSimUserCerenkov_OctantEvent_h
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

// system includes
#include "cpp_include.h"
#include "Root_include.h"

#ifndef __CINT__
#include "Geant4_include.hh"
#endif

// user include
#include "QweakSimUserCerenkov_DetectorEvent.hh"
#include "QweakSimUserCerenkov_PMTEvent.hh"

// user classes
class QweakSimUserCerenkov_DetectorEvent;
class QweakSimUserCerenkov_PMTEvent;

```

Figure 6.41: Header File

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
//class QweakSimUserOctantEvent : public TObject
class QweakSimUserCerenkov_OctantEvent
{

private:

public:

  QweakSimUserCerenkov_DetectorEvent  Detector;
  QweakSimUserCerenkov_PMTEvent       PMT;

public:

  // Constructor
  QweakSimUserCerenkov_OctantEvent();
  // Destructor
 virtual ~QweakSimUserCerenkov_OctantEvent();


  //void SetTree(TTree *data){  Detector.SetTree(data); };

  // define a new Class known to ROOT
  ClassDef(QweakSimUserCerenkov_OctantEvent,1)

};  // end class QweakSimCerenkov_OctantEvent

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

#endif

//=====================================================================================
==
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_OctantEvent.hh,v $
//      Revision 1.2  2005/12/27 19:28:38  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 6.42: Header File

```
//=============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//   ---------------------------
//
/**

   \file QweakSimUserCerenkov_OctantEvent.cc

   $Revision: 1.2 $
   $Date: 2005/12/27 19:16:49 $

   \author Klaus Hans Grimm

*/
//=============================================================================

//=============================================================================
//   -----------------------
//  | CVS File Information |
//   -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:16:49 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//============================================================================


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

#include "QweakSimUserCerenkov_OctantEvent.hh"

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

ClassImp(QweakSimUserCerenkov_OctantEvent)

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserCerenkov_OctantEvent::QweakSimUserCerenkov_OctantEvent()
{

}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserCerenkov_OctantEvent::~QweakSimUserCerenkov_OctantEvent()
{;}

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

//=======================================================
//   -----------------------
//  | CVS File Information |
//   -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_OctantEvent.cc,v $

```

Figure 6.43: Source File

```

//============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
/**

   \file QweakSimUserCerenkov_DetectorEvent.hh
   $Revision: 1.3 $
   $Date: 2006/01/06 20:31:24 $
   \author Klaus Hans Grimm

*/
//============================================================================
//
//============================================================================
//
//   ---------------------------
//  | Doxygen Class Information |
//  ---------------------------
/**
   \class QweakSimUserCerenkov_DetectorEvent

   \brief ROOT Subtree structure for Cerenkov DetectorEvent

   Placeholder for a long explaination

 */
//============================================================================
//
//============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2006/01/06 20:31:24 $
//  CVS/RCS Revision: $Revision: 1.3 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//============================================================================

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
#ifndef QweakSimUserCerenkov_DetectorEvent_h
#define QweakSimUserCerenkov_DetectorEvent_h
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

// system includes
#include "cpp_include.h"
#include "Root_include.h"
// #include "QweakSimUserCerenkov_SecondaryParticleEvent.hh"

#ifndef __CINT__
#include "Geant4_include.hh"
#endif

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
class QweakSimUserCerenkov_DetectorEvent
{

private:

```

Figure 6.44: Header File

```
//   TTree   *DataTree;
//   TBranch *secondaryElectronBranch;

   Int_t   DetectorID;

   Float_t TrackID;
   Float_t GlobalTimeOfHit;

   Int_t HasBeenHit;
   Int_t EdgeEventFlag;
   Int_t NbOfHits;
   Int_t SecondaryParticleCount;
   Int_t SecondaryElectronCount;
   Int_t SecondaryPhotonCount;
   Int_t SecondaryPositronCount;
   Int_t OpticalPhotonCount;

//   QweakSimUserCerenkov_SecondaryParticleEvent *secondaryElectronEvent;

   Float_t *SecPartLocalOriginX; //[SecondaryParticleCount]
   Float_t *SecPartLocalOriginY; //[SecondaryParticleCount]
   Float_t *SecPartLocalOriginZ; //[SecondaryParticleCount]

   Float_t *SecPartLocalMomentumX; //[SecondaryParticleCount]
   Float_t *SecPartLocalMomentumY; //[SecondaryParticleCount]
   Float_t *SecPartLocalMomentumZ; //[SecondaryParticleCount]

   Float_t *SecPartLocalEnergy; //[SecondaryParticleCount]
   Float_t *SecPartLocalCharge; //[SecondaryParticleCount]

   vector <Double_t> CerenkovPhotonEnergy;

   Float_t HitLocalPositionX;
   Float_t HitLocalPositionY;
   Float_t HitLocalPositionZ;
   Float_t HitLocalExitPositionX;
   Float_t HitLocalExitPositionY;
   Float_t HitLocalExitPositionZ;
   Float_t HitGlobalPositionX;
   Float_t HitGlobalPositionY;
   Float_t HitGlobalPositionZ;

   Float_t OriginVertexPositionX;
   Float_t OriginVertexPositionY;
   Float_t OriginVertexPositionZ;

   Float_t OriginVertexMomentumDirectionX;
   Float_t OriginVertexMomentumDirectionY;
   Float_t OriginVertexMomentumDirectionZ;

   Float_t OriginVertexThetaAngle;
   Float_t OriginVertexPhiAngle;

   Float_t OriginVertexKineticEnergy;
   Float_t OriginVertexTotalEnergy;

   Float_t LocalVertexTotalEnergy;

   Float_t PrimaryQ2;
   Float_t CrossSectionWeight;

   Float_t GlobalPhiAngle;
   Float_t GlobalThetaAngle;

   TString ParticleName;
   Int_t   ParticleType;

```

Figure 6.45: Header File

```
  Float_t TotalEnergy;
  Float_t KineticEnergy;

public:

  // Constructor
  QweakSimUserCerenkov_DetectorEvent();
  // Destructor
  virtual ~QweakSimUserCerenkov_DetectorEvent();

   void Initialize();

  //-----------------
  void     StoreDetectorID(Int_t did)  { DetectorID = did; }
  Int_t    GetDetectorID() const {return DetectorID;}
  //-----------------

  void     StoreTrackID(Float_t tid)  { TrackID = tid; }
  Float_t    GetTrackID() const {return TrackID;}

  void     StoreParticleName(TString pn)    { ParticleName = pn; }
  TString    GetParticleName() const {return  ParticleName;}

  void     StoreParticleType(Int_t pt)      { ParticleType = pt; }
  Int_t      GetParticleType() const {return  ParticleType;}


  void     StoreGlobalTimeOfHit(Float_t gtime) { GlobalTimeOfHit = gtime; }
  Float_t    GetGloablTimeOfHit()  const {return GlobalTimeOfHit;}

  void     StoreTotalEnergy(Float_t te)   { TotalEnergy = te; }
  Float_t    GetTotalEnergy() const {return TotalEnergy;}

  void     StoreKineticEnergy(Float_t ke)   { KineticEnergy = ke; }
  Float_t    GetKineticEnergy() const {return KineticEnergy;}


  //-----------------
  void     StoreDetectorHasBeenHit(Int_t n)      { HasBeenHit = n; }
  Int_t      GetDetectorHasBeenHit() const {return HasBeenHit;}
  //-----------------
  void     StoreDetectorNbOfHits(Int_t nd)      { NbOfHits = nd; }
  Int_t      GetDetectorNbOfHits() const {return  NbOfHits;}
  //-----------------
  void     StoreDetectorLocalPositionX(Float_t lx)   { HitLocalPositionX = lx; }
  Float_t    GetDetectorLocalPositionX() const {return HitLocalPositionX;}

  void     StoreDetectorLocalPositionY(Float_t ly)   { HitLocalPositionY = ly; }
  Float_t    GetDetectorLocalPositionY() const {return HitLocalPositionY;}

  void     StoreDetectorLocalPositionZ(Float_t lz)   { HitLocalPositionZ = lz; }
  Float_t    GetDetectorLocalPositionZ() const {return HitLocalPositionZ;}
  //-----------------
  void     StoreDetectorLocalExitPositionX(Float_t lx)   { HitLocalExitPositionX = lx; }
  Float_t    GetDetectorLocalExitPositionX() const {return HitLocalExitPositionX;}

  void     StoreDetectorLocalExitPositionY(Float_t ly)   { HitLocalExitPositionY = ly; }
  Float_t    GetDetectorLocalExitPositionY() const {return HitLocalExitPositionY;}

  void     StoreDetectorLocalExitPositionZ(Float_t lz)   { HitLocalExitPositionZ = lz; }
  Float_t    GetDetectorLocalExitPositionZ() const {return HitLocalExitPositionZ;}
  //---
  void     StoreDetectorGlobalPositionX(Float_t lx)   { HitGlobalPositionX = lx; }
  Float_t    GetDetectorGlobalPositionX() const {return HitGlobalPositionX;}

  void     StoreDetectorGlobalPositionY(Float_t ly)   { HitGlobalPositionY = ly; }
  Float_t    GetDetectorGlobalPositionY() const {return HitGlobalPositionY;}
```

Figure 6.46: Header File

```
void     StoreDetectorGlobalPositionZ(Float_t lz)   { HitGlobalPositionZ = lz; }
Float_t    GetDetectorGlobalPositionZ() const {return HitGlobalPositionZ;}
//-----------------
void     StoreOriginVertexPositionX(Float_t vx)   { OriginVertexPositionX = vx; }
Float_t    GetOriginVertexPositionX() const {return OriginVertexPositionX;}

void     StoreOriginVertexPositionY(Float_t vy)   { OriginVertexPositionY = vy; }
Float_t    GetOriginVertexPositionY() const {return OriginVertexPositionY;}

void     StoreOriginVertexPositionZ(Float_t vz)   { OriginVertexPositionZ = vz; }
Float_t    GetOriginVertexPositionZ() const {return OriginVertexPositionZ;}
//-----------------
void     StoreOriginVertexMomentumDirectionX(Float_t vx)   { OriginVertexMomentumDirectionX = vx; }
Float_t    GetOriginVertexMomentumDirectionX() const {return OriginVertexMomentumDirectionX;}

void     StoreOriginVertexMomentumDirectionY(Float_t vy)   { OriginVertexMomentumDirectionY = vy; }
Float_t    GetOriginVertexMomentumDirectionY() const {return OriginVertexMomentumDirectionY;}

void     StoreOriginVertexMomentumDirectionZ(Float_t vz)   { OriginVertexMomentumDirectionZ = vz; }
Float_t    GetOriginVertexMomentumDirectionZ() const {return OriginVertexMomentumDirectionZ;}
//-----------------
void     StoreOriginVertexThetaAngle(Float_t theta) { OriginVertexThetaAngle = theta; }
Float_t    GetOriginVertexThetaAngle() const  {return OriginVertexThetaAngle;}

void     StoreOriginVertexPhiAngle(Float_t phi)  { OriginVertexPhiAngle = phi; }
Float_t    GetOriginVertexPhiAngle() const {return OriginVertexPhiAngle;}
//-----------------
void     StoreOriginVertexKineticEnergy(Float_t ekin) { OriginVertexKineticEnergy = ekin; }
Float_t    GetOriginVertexKineticEnergy() const {return OriginVertexKineticEnergy;}

void     StoreOriginVertexTotalEnergy(Float_t etot) { OriginVertexTotalEnergy = etot; }
Float_t    GetOriginVertexTotalEnergy() const {return OriginVertexTotalEnergy;}

void     StoreDetectorLocalVertexTotalEnergy(Float_t etot) { LocalVertexTotalEnergy = etot; };
Float_t    GetDetectorLocalVertexTotalEnergy() {return LocalVertexTotalEnergy;};
//----------------

void     StorePrimaryQ2(Float_t pq2)  { PrimaryQ2 = pq2; }
Float_t    GetPrimaryQ2() const {return PrimaryQ2; }
//-----------------
void     StoreCrossSectionWeight(Float_t csw)   {CrossSectionWeight = csw;}
Float_t    GetCrossSectionWeight() const {return CrossSectionWeight; }
//-----------------

void     StoreGlobalThetaAngle(Float_t theta) { GlobalThetaAngle = theta; }
Float_t    GetGlobalThetaAngle() const  {return GlobalThetaAngle;}

void     StoreGlobalPhiAngle(Float_t phi)  { GlobalPhiAngle = phi; }
Float_t    GetGlobalPhiAngle() const {return GlobalPhiAngle;}


void AddSecondaryParticleEvent(Float_t XO, Float_t YO, Float_t ZO,
                         Float_t XM, Float_t YM, Float_t ZM,
                         Float_t Eng, Float_t Charge);

void     StoreEdgeEventFlag(Int_t flag) {EdgeEventFlag = flag;};
Int_t    GetEdgeEventFlag() {return EdgeEventFlag;};

void     StoreOpticalPhotonCount(Int_t cnt){OpticalPhotonCount = cnt;};

void StoreCerenkovPhotonEnergy(Double_t eng) {CerenkovPhotonEnergy.push_back(eng);};

// define a new Class known to ROOT
ClassDef(QweakSimUserCerenkov_DetectorEvent,1)

};  // end class QweakSimUserCerenkov_DetectorEvent
```

Figure 6.47: Header File

```

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

#endif

//==============================================================================
==
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_DetectorEvent.hh,v $
//      Revision 1.3  2006/01/06 20:31:24  grimm
//      Added KineticEnergy, TotalEnergy, ParticleType, and ParticleName into the root tree
//
//      Revision 1.2  2005/12/27 19:28:31  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 6.48: Header File

```
//==========================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
//
/**

   \file QweakSimUserCerenkov_DetectorEvent.cc

   $Revision: 1.3 $
   $Date: 2006/01/06 21:43:04 $

   \author Klaus Hans Grimm

*/
//==========================================================================

//==========================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2006/01/06 21:43:04 $
//  CVS/RCS Revision: $Revision: 1.3 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//=========================================================================

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

#include "QweakSimUserCerenkov_DetectorEvent.hh"

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

ClassImp(QweakSimUserCerenkov_DetectorEvent)

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
 QweakSimUserCerenkov_DetectorEvent::QweakSimUserCerenkov_DetectorEvent()
{
 SecondaryParticleCount        = 0;
 Initialize();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserCerenkov_DetectorEvent::~QweakSimUserCerenkov_DetectorEvent()
{;}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimUserCerenkov_DetectorEvent::Initialize()
{
   DetectorID                   = 0;
   TrackID                      = 0;
   HasBeenHit                   = 0;
   NbOfHits                     = 0;
   GlobalTimeOfHit              = 0.0;
   EdgeEventFlag                = 0;

```

Figure 6.49: Source File

```
  if(SecondaryParticleCount){
    delete[] SecPartLocalOriginX;
    delete[] SecPartLocalOriginY;
    delete[] SecPartLocalOriginZ;

    delete[] SecPartLocalMomentumX;
    delete[] SecPartLocalMomentumY;
    delete[] SecPartLocalMomentumZ;

    delete[] SecPartLocalEnergy;
    delete[] SecPartLocalCharge;
  }


  SecPartLocalOriginX = NULL;
  SecPartLocalOriginY = NULL;
  SecPartLocalOriginZ = NULL;

  SecPartLocalMomentumX = NULL;
  SecPartLocalMomentumY = NULL;
  SecPartLocalMomentumZ = NULL;

  SecPartLocalEnergy = NULL;
  SecPartLocalCharge = NULL;

  OpticalPhotonCount = 0;
  CerenkovPhotonEnergy.clear();
  CerenkovPhotonEnergy.resize(0);

  SecondaryParticleCount = 0;
  SecondaryElectronCount = 0;
  SecondaryPhotonCount   = 0;
  SecondaryPositronCount = 0;

  HitLocalPositionX            = 0.0;
  HitLocalPositionY            = 0.0;
  HitLocalPositionZ            = 0.0;

  HitGlobalPositionX           = 0.0;
  HitGlobalPositionY           = 0.0;
  HitGlobalPositionZ           = 0.0;

  OriginVertexPositionX        = 0.0;
  OriginVertexPositionY        = 0.0;
  OriginVertexPositionZ        = 0.0;

  OriginVertexMomentumDirectionX = 0.0;
  OriginVertexMomentumDirectionY = 0.0;
  OriginVertexMomentumDirectionZ = 0.0;

  OriginVertexThetaAngle       = 0.0;
  OriginVertexPhiAngle         = 0.0;
  OriginVertexKineticEnergy    = 0.0;
  OriginVertexTotalEnergy      = 0.0;

  PrimaryQ2                    = 0.0;
  CrossSectionWeight           = 0.0;

  EdgeEventFlag                = 0;

  ParticleName                 = "None";
  ParticleType                 = -1;
  TotalEnergy                  = 0.;
  KineticEnergy                = 0.;

}

```

Figure 6.50: Source File

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

void QweakSimUserCerenkov_DetectorEvent::AddSecondaryParticleEvent(Float_t XO, Float_t YO, Float_t ZO,
                                                                  Float_t XM, Float_t YM, Float_t ZM,
                                                                  Float_t Eng, Float_t charge)
{
  G4int cnt = SecondaryParticleCount;
  Float_t *tmp1X = NULL;
  Float_t *tmp1Y = NULL;
  Float_t *tmp1Z = NULL;

  Float_t *tmp2X = NULL;
  Float_t *tmp2Y = NULL;
  Float_t *tmp2Z = NULL;

  Float_t *tmp3 = NULL;
  Float_t *tmp4 = NULL;

  if(cnt){
    tmp1X = new Float_t[cnt];
    tmp1Y = new Float_t[cnt];
    tmp1Z = new Float_t[cnt];

    tmp2X = new Float_t[cnt];
    tmp2Y = new Float_t[cnt];
    tmp2Z = new Float_t[cnt];

    tmp3 = new Float_t[cnt];
    tmp4 = new Float_t[cnt];
  }

  for(Int_t i = 0; i < cnt; i++){
    tmp1X[i] = SecPartLocalOriginX[i];
    tmp1Y[i] = SecPartLocalOriginY[i];
    tmp1Z[i] = SecPartLocalOriginZ[i];

    tmp2X[i] = SecPartLocalMomentumX[i];
    tmp2Y[i] = SecPartLocalMomentumY[i];
    tmp2Z[i] = SecPartLocalMomentumZ[i];

    tmp3[i]  = SecPartLocalEnergy[i];
    tmp4[i]  = SecPartLocalCharge[i];
  }

  if(cnt && SecPartLocalOriginX) delete[] SecPartLocalOriginX;
  if(cnt && SecPartLocalOriginY) delete[] SecPartLocalOriginY;
  if(cnt && SecPartLocalOriginZ) delete[] SecPartLocalOriginZ;

  if(cnt && SecPartLocalMomentumX) delete[] SecPartLocalMomentumX;
  if(cnt && SecPartLocalMomentumX) delete[] SecPartLocalMomentumY;
  if(cnt && SecPartLocalMomentumX) delete[] SecPartLocalMomentumZ;

  if(cnt && SecPartLocalEnergy) delete[] SecPartLocalEnergy;
  if(cnt && SecPartLocalCharge) delete[] SecPartLocalCharge;

  SecPartLocalOriginX   = new Float_t[cnt+1];
  SecPartLocalOriginY   = new Float_t[cnt+1];
  SecPartLocalOriginZ   = new Float_t[cnt+1];

  SecPartLocalMomentumX = new Float_t[cnt+1];
  SecPartLocalMomentumY = new Float_t[cnt+1];
  SecPartLocalMomentumZ = new Float_t[cnt+1];

  SecPartLocalEnergy    = new Float_t[cnt+1];
  SecPartLocalCharge    = new Float_t[cnt+1];

  for(Int_t i = 0; i < cnt; i++) {
```

Figure 6.51: Source File

```
      SecPartLocalOriginX[i]   = tmp1X[i];
      SecPartLocalOriginY[i]   = tmp1Y[i];
      SecPartLocalOriginZ[i]   = tmp1Z[i];

      SecPartLocalMomentumX[i] = tmp2X[i];
      SecPartLocalMomentumY[i] = tmp2Y[i];
      SecPartLocalMomentumZ[i] = tmp2Z[i];

      SecPartLocalEnergy[i]    = tmp3[i] ;
      SecPartLocalCharge[i]    = tmp4[i] ;
    }

    SecPartLocalOriginX[cnt]   =  XO;
    SecPartLocalOriginY[cnt]   =  YO;
    SecPartLocalOriginZ[cnt]   =  ZO;

    SecPartLocalMomentumX[cnt] =  XM;
    SecPartLocalMomentumY[cnt] =  YM;
    SecPartLocalMomentumZ[cnt] =  ZM;

    SecPartLocalEnergy[cnt]    =  Eng;
    SecPartLocalCharge[cnt]    =  charge;

    if(cnt){
      delete[] tmp1X;
      delete[] tmp1Y;
      delete[] tmp1Z;

      delete[] tmp2X;
      delete[] tmp2Y;
      delete[] tmp2Z;

      delete[] tmp3;
      delete[] tmp4;
    }

    if(charge == -1)  SecondaryElectronCount++;
    if(charge == 0)   SecondaryPhotonCount++;
    if(charge == 1)   SecondaryPositronCount++;
    SecondaryParticleCount++;
  }
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

//=======================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_DetectorEvent.cc,v $
//      Revision 1.3  2006/01/06 21:43:04  grimm
//       Added initialization of:
//
//        ParticleName                  = "None";
//        ParticleType                  = -1;
//        TotalEnergy                   = 0.;
//        KineticEnergy                 = 0.;
//
//      Revision 1.2  2005/12/27 19:16:42  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 6.52: Source File

```

//============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
/**

   \file QweakSimUserCerenkov_PMTEvent.hh
   $Revision: 1.2 $
   $Date: 2005/12/27 19:29:20 $
   \author Klaus Hans Grimm

*/
//============================================================================
//
//============================================================================
//
//   ---------------------------
//  | Doxygen Class Information |
//  ---------------------------
/**
   \class QweakSimUserCerenkov_PMTEvent

   \brief ROOT Subtree structure for Cerenkov PMTEvent

   Placeholder for a long explaination

 */
//============================================================================
//
//============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:29:20 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//============================================================================
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
#ifndef QweakSimUserCerenkov_PMTEvent_h
#define QweakSimUserCerenkov_PMTEvent_h
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

#include "cpp_include.h"
#include "Root_include.h"

#ifndef __CINT__
#include "Geant4_include.hh"
#endif

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
class QweakSimUserCerenkov_PMTEvent
{

private:

  Int_t   DetectorID;

  Float_t TrackID;
```

Figure 6.53: Header File

```

  Int_t PMTHasBeenHit;

  Int_t PMTLeftNbOfHits;
  Int_t PMTRightNbOfHits;
  Int_t PMTTotalNbOfHits;

  Float_t PMTLeftNbOfPEs;
  Float_t PMTRightNbOfPEs;
  Float_t PMTTotalNbOfPEs;

public:

  // Constructor
  QweakSimUserCerenkov_PMTEvent();
  // Destructor
  virtual ~QweakSimUserCerenkov_PMTEvent();

  void Initialize();

  //-----------------
  void     StoreDetectorID(Int_t did)  { DetectorID = did; }
  Int_t    GetDetectorID() const {return DetectorID;}
  //-----------------

  //-----------------
  void     StoreTrackID(Float_t tid)    { TrackID = tid; }
  Float_t    GetTrackID() const {return TrackID;}
   //-----------------
  void     StorePMTHasBeenHit(Int_t np)      { PMTHasBeenHit = np; }
  Int_t      GetPMTHasBeenHit() const {return  PMTHasBeenHit;}
  //-----------------
  void     StorePMTLeftNbOfHits(Int_t npl)      { PMTLeftNbOfHits = npl; }
  Int_t      GetPMTLeftNbOfHits() const {return   PMTLeftNbOfHits;}

  void     StorePMTRightNbOfHits(Int_t npr)      { PMTRightNbOfHits = npr; }
  Int_t      GetPMTRightNbOfHits() const {return   PMTRightNbOfHits;}

  void     StorePMTTotalNbOfHits(Int_t npt)      { PMTTotalNbOfHits = npt; }
  Int_t      GetPMTTotalNbOfHits() const {return   PMTTotalNbOfHits;}
  //-----------------
  void     StorePMTLeftNbOfPEs(Float_t npl)      { PMTLeftNbOfPEs = npl; }
  Float_t      GetPMTLeftNbOfPEs() const {return   PMTLeftNbOfPEs;}

  void     StorePMTRightNbOfPEs(Float_t npr)      { PMTRightNbOfPEs = npr; }
  Float_t    GetPMTRightNbOfPEs() const {return   PMTRightNbOfPEs;}

  void     StorePMTTotalNbOfPEs(Float_t npt)      { PMTTotalNbOfPEs = npt; }
  Float_t    GetPMTTotalNbOfPEs() const {return   PMTTotalNbOfPEs;}

  // define a new Class known to ROOT
  ClassDef(QweakSimUserCerenkov_PMTEvent,1)

};  // end class QweakSimUserCerenkov_DetectorEvent

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

#endif

//==========================================================================================
==
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_PMTEvent.hh,v $
```

Figure 6.54: Header File

```
//      Revision 1.2  2005/12/27 19:29:20  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//
```

Figure 6.55: Header File

```
//=========================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
//
/**

   \file QweakSimUserCerenkov_PMTEvent.cc

   $Revision: 1.2 $
   $Date: 2005/12/27 19:16:56 $

   \author Klaus Hans Grimm

*/
//=========================================================================

//=========================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:16:56 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//========================================================================


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

#include "QweakSimUserCerenkov_PMTEvent.hh"

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

ClassImp(QweakSimUserCerenkov_PMTEvent)

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
 QweakSimUserCerenkov_PMTEvent::QweakSimUserCerenkov_PMTEvent()
{
  Initialize();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserCerenkov_PMTEvent::~QweakSimUserCerenkov_PMTEvent()
{;}

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

void QweakSimUserCerenkov_PMTEvent::Initialize()
{
  DetectorID                  = 0;
  TrackID                     = 0;

  PMTHasBeenHit               = 0;
  PMTLeftNbOfHits             = 0;
  PMTRightNbOfHits            = 0;
```

Figure 6.56: Source File

```
  PMTTotalNbOfHits            = 0;
  PMTLeftNbOfPEs              = 0;
  PMTRightNbOfPEs             = 0;
  PMTTotalNbOfPEs             = 0;

}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

//=======================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserCerenkov_PMTEvent.cc,v $
//      Revision 1.2  2005/12/27 19:16:56  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 6.57: Source File

# Chapter 7

# Qweak VDC Detectors

# Chapter 8

# Qweak HDC Detectors

# Chapter 9

# Qweak GEM Detectors

# Chapter 10

# Qweak Trigger Scintillator

# Chapter 11

# Qweak Target

# Chapter 12

# Qweak Magnets and Fields

# Chapter 13

# Qweak Physics Lists

# Chapter 14

# Qweak Main Data Tree Structure and Readout

The event readout is implemented using several event data classes that define the event ROOT tree that is stored in a ROOT file at the end of a run. Each sensitive detector has at least one, but possibly several, event classes that have various data members which are newly filled for every event. In addition the generated primary event data is stored in its own event class. An event is completed after the primary track/particle and all secondary tracks have been killed by some process or have left the user defined world volume. An event is valid, if it has triggered at least one hit in some detector. This condition is tested in the class *QweakSimEventAction* (see lines 340-346, p. 145). If an event is valid (has detector hits), *QweakSimEventAction* collects the information for the hits from various classes, including the hit collectors for the sensitive detectors and *QweakSimUserInformation*, in which the data has been assmbled as the event is stepped through the simulation. The program then fills the event class data members and writes the data to the ROOT tree (line 1502, p. 165), which is defined and implemented in the class *QweakSimAnalysis*. The data members of the event classes are cleared after the tree is filled with the current event, or old, stored, data values are simply overwritten for each new event.

Figure 14.1, shows the tree structure up to the 6 event classes that are specific to a particular volume/detector and the primary event data. Each of these 6 classes is inherited by one or more other classes which further define the event structure, except for the primary event structure, which has only one class. The details of these 6 classes are described in other chapters, together with the specific detector/object for which they store the data.

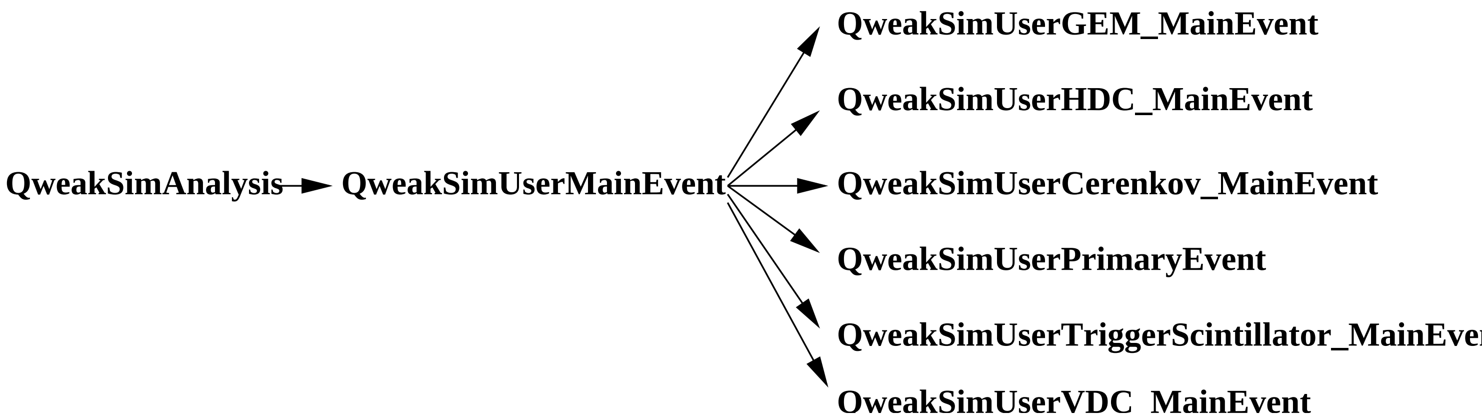
QweakSimAnalysis
QweakSimUserMainEvent
QweakSimUserGEM_MainEvent
QweakSimUserHDC_MainEvent
QweakSimUserCerenkov_MainEvent
QweakSimUserPrimaryEvent
QweakSimUserTriggerScintillator_MainEve
OweakSimUserVDC MainEvent

Figure 14.1:

```
//============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
/**

   \file QweakSimAnalysis.hh
   $Revision: 1.2 $
   $Date: 2005/12/27 19:22:16 $
   \author Klaus Hans Grimm

*/
//============================================================================
//
//============================================================================
//
//   ---------------------------
//  | Doxygen Class Information |
//  ---------------------------
/**
   \class QweakSimAnalysis

   \brief Handling of the output ROOT file

   Placeholder for a long explaination

 */
//============================================================================
//
//============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:22:16 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//============================================================================

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
#ifndef QweakSimAnalysis_h
#define QweakSimAnalysis_h
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

// system includes
#include "cpp_include.h"
#include "Root_include.h"
#include "Geant4_include.hh"

// user includes
#include "QweakSimUserMainEvent.hh"

// user classes
class QweakSimUserMainEvent;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
class QweakSimAnalysis {
public:
  QweakSimAnalysis();
  virtual ~QweakSimAnalysis();
```

Figure 14.2: Header File

```

public:

  void BeginOfRun();
  void EndOfRun();
  void EndOfEvent(G4int flag);

  void Init();
  void Finish();

  void Fill_RootNtuple() {QweakSimG4_RootNtuple->Fill();}
  void AutoSaveRootNtuple();

  QweakSimUserMainEvent* QweakSimG4_RootEvent;


private:

  void ConstructRootNtuple();

  TTree*   QweakSimG4_RootNtuple;
  TBranch* QweakSimG4_RootBranch;
  TFile*   QweakSimG4_RootFile;

};

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

#endif

//===========================================================================
==
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimAnalysis.hh,v $
//      Revision 1.2  2005/12/27 19:22:16  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//
```

Figure 14.3: Header File

```
//=========================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
//
/**

   \file QweakSimAnalysis.cc

   $Revision: 1.4 $
   $Date: 2006/01/06 19:12:25 $

   \author Klaus Hans Grimm

*/
//=========================================================================

//=========================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2006/01/06 19:12:25 $
//  CVS/RCS Revision: $Revision: 1.4 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//==========================================================================


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

#include "QweakSimAnalysis.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimAnalysis::QweakSimAnalysis()
{
   // Initialize
   QweakSimG4_RootEvent    = NULL;
   QweakSimG4_RootNtuple   = NULL;
   QweakSimG4_RootBranch   = NULL;
   QweakSimG4_RootFile     = NULL;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimAnalysis::~QweakSimAnalysis()
{
  Finish();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimAnalysis::Init()
{;}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimAnalysis::Finish()
{
  if (QweakSimG4_RootEvent)    delete QweakSimG4_RootEvent;
```

Figure 14.4: Source File

```
  if (QweakSimG4_RootNtuple)   delete QweakSimG4_RootNtuple;
  if (QweakSimG4_RootBranch)   delete QweakSimG4_RootBranch;
  if (QweakSimG4_RootFile)     delete QweakSimG4_RootFile;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimAnalysis::BeginOfRun()
{
  QweakSimG4_RootFile = new TFile( "QweakSim.root","RECREATE","W&M Qweak ROOT file");

  ConstructRootNtuple();

}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimAnalysis::EndOfRun()
{

  QweakSimG4_RootFile->Write(); // Writing the data to the ROOT file
  QweakSimG4_RootFile->Close();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimAnalysis::EndOfEvent(G4int flag)
{
// This member is called at the end of every event
  if(!flag) return;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
void QweakSimAnalysis::ConstructRootNtuple()
{

  // create ROOT tree

  QweakSimG4_RootNtuple = new TTree("QweakSimG4_Tree","QweakSimG4_Tree");

  // save the file after so many bytes. Avoids complete data loss after crash
  //QweakSimG4_RootNtuple ->SetAutoSave(1000000);  //AutoSave after every 1 Mbyte written to disk

  // Instance of data structure to be written into ROOT file
  QweakSimG4_RootEvent  = new QweakSimUserMainEvent();

  // Create a branch with the data structure defined by QweakSimG4_Event

  int bufsize = 64000;
  int split   = 10;

  QweakSimG4_RootBranch = QweakSimG4_RootNtuple->Branch("QweakSimUserMainEvent", "QweakSimUserMainEvent", &QweakSimG4_RootEvent, 64000, 10);


}

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

void QweakSimAnalysis::AutoSaveRootNtuple()
{
   // save the current ntuple:
   // In case your program crashes before closing the file holding this tree,
   // the file will be automatically recovered when you will connect the file
   // in UPDATE mode.
   // The Tree will be recovered at the status corresponding to the last AutoSave.
   //
   // if option contains "SaveSelf", gDirectory->SaveSelf() is called.
   // This allows another process to analyze the Tree while the Tree is being filled.
   //
```

Figure 14.5: Source File

```
    // see http://root.cern.ch/root/html/TTree.html#TTree:AutoSave

    //QweakSimG4_RootNtuple -> AutoSave("SaveSelf");
    QweakSimG4_RootNtuple -> AutoSave();

}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

//========================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimAnalysis.cc,v $
//      Revision 1.4  2006/01/06 19:12:25  grimm
//      Bogus commit due to time stamp mismatch
//
//      Revision 1.3  2006/01/06 18:00:00  grimm
//      Bogus commit. CVS time on dogbert was in the future
//
//      Revision 1.2  2005/12/27 19:01:03  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 14.6: Source File

```
//==============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
/**

   \file QweakSimUserMainEvent.hh
   $Revision: 1.2 $
   $Date: 2005/12/27 19:30:33 $
   \author Klaus Hans Grimm

*/
//==============================================================================
//
//==============================================================================
//
//   ---------------------------
//  | Doxygen Class Information |
//  ---------------------------
/**
   \class QweakSimUserMainEvent

   \brief Defines Top ROOT Tree structure of the ROOT file for each event.

         Primary.

    Region1.

    Region2.

    Region3.

    Cerenkov.

   Placeholder for a long explaination

 */
//==============================================================================
//
//==============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:30:33 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//=============================================================================

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
#ifndef QweakSimUserMainEvent_h
#define QweakSimUserMainEvent_h
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....

// system include
#include "cpp_include.h"
#include "Root_include.h"

#ifndef __CINT__
#include "Geant4_include.hh"
```

Figure 14.7: Header File

```
#endif

// user includes
#include "QweakSimUserPrimaryEvent.hh"
#include "QweakSimUserGEM_MainEvent.hh"
#include "QweakSimUserHDC_MainEvent.hh"
#include "QweakSimUserVDC_MainEvent.hh"
#include "QweakSimUserTriggerScintillator_MainEvent.hh"
#include "QweakSimUserCerenkov_MainEvent.hh"

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

// user classes
class QweakSimUserPrimaryEvent;
class QweakSimUserGEM_MainEvent;
class QweakSimUserHDC_MainEvent;
class QweakSimUserVDC_MainEvent;
class QweakSimUserTriggerScintillator_MainEvent;
class QweakSimUserCerenkov_MainEvent;

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
//class QweakSimUserMainEvent : public TObject
class QweakSimUserMainEvent
{

private:

public:

  // top directoty of Root output tree:

  QweakSimUserPrimaryEvent           Primary;          // tree containing primart particle info

  QweakSimUserGEM_MainEvent          Region1;          // tree containing HDC info
  QweakSimUserHDC_MainEvent          Region2;          // tree containing HDC info
  QweakSimUserVDC_MainEvent          Region3;          // tree containing VDC info

  QweakSimUserTriggerScintillator_MainEvent    TriggerScintillator;  // tree containing TriggerScintilliator info

  QweakSimUserCerenkov_MainEvent         Cerenkov;        // tree containing Cdetector info

public:

  // Constructor
  QweakSimUserMainEvent();
  // Destructor
 virtual ~QweakSimUserMainEvent();

  //void SetTree(TTree *data){Cerenkov.SetTree(data);};


  // define a new Class known to ROOT
  ClassDef(QweakSimUserMainEvent,1)

};  // end class QweakSimMainEvent

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

#endif

//===================================================================================
==
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
```

Figure 14.8: Header File

```
//      $Log: QweakSimUserMainEvent.hh,v $
//      Revision 1.2  2005/12/27 19:30:33  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 14.9: Header File

```
//=============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//   ---------------------------
//
/**

   \file QweakSimUserMainEvent.cc

   $Revision: 1.2 $
   $Date: 2005/12/27 19:17:58 $

   \author Klaus Hans Grimm

*/
//=============================================================================

//=============================================================================
//   -----------------------
//  | CVS File Information |
//   -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2005/12/27 19:17:58 $
//  CVS/RCS Revision: $Revision: 1.2 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//============================================================================


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

#include "QweakSimUserMainEvent.hh"

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

ClassImp(QweakSimUserMainEvent)

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserMainEvent::QweakSimUserMainEvent()
{;}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
QweakSimUserMainEvent::~QweakSimUserMainEvent()
{;}

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

//=======================================================
//   -----------------------
//  | CVS File Information |
//   -----------------------
//
//      $Revisions$
//      $Log: QweakSimUserMainEvent.cc,v $
//      Revision 1.2  2005/12/27 19:17:58  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
```

Figure 14.10: Source File

```
//
//

```

Figure 14.11: Source File

```
//=============================================================================
/**

   \file QweakSimEventAction.hh
   $Revision: 1.4 $
   $Date: 2006/01/06 21:29:35 $
   \author Klaus Hans Grimm

*/
//=============================================================================
//
//=============================================================================
/**
   \class QweakSimEventAction

   \brief Mainly filling/storing the hit event structure at the end of an event

   Placeholder for a long explaination

 */
//=============================================================================
//
//=============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2006/01/06 21:29:35 $
//  CVS/RCS Revision: $Revision: 1.4 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS/RCS Log at end of file !!
// ===================================
//
//=============================================================================

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
#ifndef QweakSimEventAction_h
#define QweakSimEventAction_h 1
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

// system includes
#include "cpp_include.h"
#include "Root_include.h"
#include "Geant4_include.hh"

// user includes
#include "QweakSimAnalysis.hh"
#include "QweakSimUserInformation.hh"

#include "QweakSimGEM_WirePlaneHit.hh"
#include "QweakSimHDC_WirePlaneHit.hh"

#include "QweakSimVDC_WirePlaneHit.hh"
#include "QweakSimVDC_DriftCellHit.hh"

#include "QweakSimTriggerScintillator_DetectorHit.hh"
#include "QweakSimTriggerScintillator_PMTHit.hh"

#include "QweakSimCerenkov_DetectorHit.hh"
#include "QweakSimCerenkovDetector_PMTHit.hh"

#include "QweakSimTrajectory.hh"


```

Figure 14.12: Header File

```
// user classes
class QweakSimAnalysis;
class QweakSimUserInformation;

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

class QweakSimEventAction : public G4UserEventAction
{
public:
   //!Constructor
   QweakSimEventAction(QweakSimAnalysis* AN, QweakSimUserInformation* myUI);

   //!Destructor
   ~QweakSimEventAction();

public:

  void BeginOfEventAction(const G4Event* evt);
  void EndOfEventAction(const G4Event* evt);

private:

  void Initialize();
  G4double GetDistance(G4ThreeVector,G4ThreeVector);

  G4int GEM_WirePlane_CollID;
  G4int HDC_WirePlane_CollID;
  G4int VDC_WirePlane_CollID;
  G4int VDC_DriftCellFront_CollID;
  G4int VDC_DriftCellBack_CollID;
  G4int TriggerScintillatorDetector_CollID;
  G4int TriggerScintillatorPMT_CollID;
  G4int CerenkovDetector_CollID;
  G4int CerenkovDetectorPMT_CollID;

  QweakSimAnalysis*          analysis;
  QweakSimUserInformation*   myUserInfo;

  G4int n_GEMhitWirePlane;
  G4int n_HDChitWirePlane;
  G4int n_VDChitWirePlane;
  G4int n_VDChitDCFront;
  G4int n_VDChitDCBack;
  G4int n_hitTriggerScintillator;
  G4int n_hitTriggerScintillatorPMT;
  G4int n_hitCerenkov;
  G4int n_hitCerenkovPMT;

  G4int  OriginVertexPDGcode;
  Int_t rOriginVertexPDGcode;


  G4double OriginVertexParticleMass;
  Float_t rOriginVertexParticleMass;

  G4double  OriginVertexThetaAngle;
  Float_t  rOriginVertexThetaAngle;

  G4double  OriginVertexPhiAngle;
  Float_t  rOriginVertexPhiAngle;

  // get local position of hit
  G4ThreeVector localPosition;
  Float_t     rLocalPositionX;
  Float_t     rLocalPositionY;
  Float_t     rLocalPositionZ;

```

Figure 14.13: Header File

```
G4ThreeVector localExitPosition;
Float_t      rLocalExitPositionX;
Float_t      rLocalExitPositionY;
Float_t      rLocalExitPositionZ;

G4ThreeVector SecondaryParticleOrigin;
Float_t       rSecondaryPartOriginX;
Float_t       rSecondaryPartOriginY;
Float_t       rSecondaryPartOriginZ;

G4ThreeVector SecondaryParticleMomentum;
Float_t       rSecondaryPartMomentumX;
Float_t       rSecondaryPartMomentumY;
Float_t       rSecondaryPartMomentumZ;

Float_t       rSecondaryPartEnergy;
Float_t       rSecondaryPartCharge;

// get world position of hit
G4ThreeVector globalPosition;
Float_t      rGlobalPositionX;
Float_t      rGlobalPositionY;
Float_t      rGlobalPositionZ;

// get local momentum of hit
G4ThreeVector localMomentum;
Float_t      rLocalMomentumX;
Float_t      rLocalMomentumY;
Float_t      rLocalMomentumZ;

// get world momentum of hit
G4ThreeVector globalMomentum;
Float_t      rGlobalMomentumX;
Float_t      rGlobalMomentumY;
Float_t      rGlobalMomentumZ;


G4ThreeVector originVertexPosition;
G4double OriginVertexPositionX;
G4double OriginVertexPositionY;
G4double OriginVertexPositionZ;
Float_t rOriginVertexPositionX;
Float_t rOriginVertexPositionY;
Float_t rOriginVertexPositionZ;


G4ThreeVector originVertexMomentumDirection;
G4double OriginVertexMomentumDirectionX;
G4double OriginVertexMomentumDirectionY;
G4double OriginVertexMomentumDirectionZ;
Float_t rOriginVertexMomentumDirectionX;
Float_t rOriginVertexMomentumDirectionY;
Float_t rOriginVertexMomentumDirectionZ;


G4double  originVertexKineticEnergy;
Float_t  rOriginVertexKineticEnergy;

G4double  originVertexTotalEnergy;
Float_t  rOriginVertexTotalEnergy;

//---------------------------------
G4double   GlobalThetaAngle;
Float_t   rGlobalThetaAngle;

G4double   GlobalPhiAngle;
Float_t   rGlobalPhiAngle;
```

Figure 14.14: Header File

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

  G4double primaryQ2;
  Float_t rPrimaryQ2;

  G4double  crossSectionWeight;
  Float_t  rCrossSectionWeight;

  G4int  primaryEventNumber;
  Int_t rPrimaryEventNumber;

  G4double globalTime;
  Float_t rGlobalTime;

  G4double rDCWidthOnFrame;
  G4double rDCFullThickness;
  G4double rDCUPlaneWireAngle;
  G4double rDCVPlaneWireAngle;

  vector <G4int>    pmtHitsLeft;
  vector <G4int>    pmtHitsRight;
  vector <G4double> pmtNPELeft;
  vector <G4double> pmtNPERight;

  Int_t edgeEvent;

  G4String   particleName;
  TString   rParticleName;

  G4int      particleType;
  Int_t     rParticleType;

  G4double   totalEnergy;
  Float_t   rtotalEnergy;

  G4double   kineticEnergy;
  Float_t   rkineticEnergy;

  Int_t G4IndexToOctantNumber[8];

  G4int  detectorID;
  G4int  octantID;
  Int_t  rOctantID;

};

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

#endif

//=======================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimEventAction.hh,v $
//      Revision 1.4  2006/01/06 21:29:35  grimm
//      Adding variables for storing these for VDC and Cerenkov:
//
//        G4String   particleName;
//        TString   rParticleName;
//
//        G4int      particleType;
//        Int_t     rParticleType;
//
//        G4double   totalEnergy;
```

Figure 14.15: Header File

```
//        Float_t   rtotalEnergy;
//
//        G4double   kineticEnergy;
//        Float_t   rkineticEnergy;
//
//      Revision 1.3  2005/12/28 23:05:53  grimm
//      Testing: Extract trajectories collected with QweakSimTrajectory (following LXe example)
//
//      Revision 1.2  2005/12/27 19:23:34  grimm
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//


```

Figure 14.16: Header File

```
//============================================================================
//
//   ---------------------------
//  | Doxygen File Information |
//  ---------------------------
//
/**

   \file QweakSimEventAction.cc

   $Revision: 1.5 $
   $Date: 2006/05/05 21:37:16 $

   \author Klaus Hans Grimm

*/
//============================================================================

//============================================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//  Last Update:      $Author: grimm $
//  Update Date:      $Date: 2006/05/05 21:37:16 $
//  CVS/RCS Revision: $Revision: 1.5 $
//  Status:           $State: Exp $
//
// ===================================
//  CVS Revision Log at end of file !!
// ===================================
//
//===========================================================================

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

#include "QweakSimEventAction.hh"

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
QweakSimEventAction::QweakSimEventAction(QweakSimAnalysis* AN, QweakSimUserInformation* myUI)
{

//---------------------------------------------------------------------------------------------
//! Constructor of QweakSimEventAction
 /*!

 \param  QweakSimAnalysis*          - class containing the Geamt4 hit data structure
 \param  QweakSimUserInformation*   - class containing user information like Q2 for this event or QE of some PMTs
                                      which is needed for processing/saving hit information


 */
//---------------------------------------------------------------------------------------------


   GEM_WirePlane_CollID                  = -1;
   HDC_WirePlane_CollID                  = -1;
   VDC_WirePlane_CollID                  = -1;
   VDC_DriftCellFront_CollID             = -1;
   VDC_DriftCellBack_CollID              = -1;
   TriggerScintillatorDetector_CollID    = -1;
   //TriggerScintillatorPMT_CollID         = -1;
   CerenkovDetector_CollID               = -1;
```

Figure 14.17: Source File

```
    CerenkovDetectorPMT_CollID          = -1;

    analysis    = AN;
    myUserInfo  = myUI;
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
QweakSimEventAction::~QweakSimEventAction()
{;}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimEventAction::BeginOfEventAction(const G4Event* evt)
{


  G4SDManager * SDman = G4SDManager::GetSDMpointer();

  // check for existing GEM_WirePlane Collection ID (if it's -1 it will be assigned)
  if (GEM_WirePlane_CollID==-1) {
     GEM_WirePlane_CollID = SDman->GetCollectionID("GEMWirePlaneSD/GEMWirePlaneCollection");
  }

  // check for existing HDC_WirePlane Collection ID (if it's -1 it will be assigned)
  if (HDC_WirePlane_CollID==-1) {
     HDC_WirePlane_CollID = SDman->GetCollectionID("HDCWirePlaneSD/HDCWirePlaneCollection");
  }

  // check for existing VDC_WirePlane Collection ID (if it's -1 it will be assigned)
  if (VDC_WirePlane_CollID==-1) {
     VDC_WirePlane_CollID = SDman->GetCollectionID("VDCWirePlaneSD/VDCWirePlaneCollection");
  }

  // check for existing VDC_DriftCellFront Collection ID (if it's -1 it will be assigned)
  if (VDC_DriftCellFront_CollID==-1) {
     VDC_DriftCellFront_CollID = SDman->GetCollectionID("VDCDriftCellFrontSD/DriftCellFrontCollection");
  }

 // check for existing VDC_DriftCellBack Collection ID (if it's -1 it will be assigned)
  if (VDC_DriftCellBack_CollID==-1) {
     VDC_DriftCellBack_CollID = SDman->GetCollectionID("VDCDriftCellBackSD/DriftCellBackCollection");
  }


  // check for existing TriggerScintillator Collection ID (if it's -1 it will be assigned)
  if (TriggerScintillatorDetector_CollID==-1) {
     TriggerScintillatorDetector_CollID = SDman->GetCollectionID("TriggerScintillatorSD/TriggerScintillatorCollection");
  }

//   // check for existing CerenkovDetectorPMT Collection ID (if it's -1 it will be assigned)
//   if (TriggerScintillatorPMT_CollID==-1) {
//      TriggerScintillatorPMT_CollID = SDman->GetCollectionID("TriggerScintillatorPMTSD/TriggerScintillatorPMTHitCollection");
//   }


  // check for existing CerenkovDetector Collection ID (if it's -1 it will be assigned)
  if (CerenkovDetector_CollID==-1) {
     CerenkovDetector_CollID = SDman->GetCollectionID("CerenkovDetectorSD/CerenkovDetectorCollection");
  }

  // check for existing CerenkovDetectorPMT Collection ID (if it's -1 it will be assigned)
  if (CerenkovDetectorPMT_CollID==-1) {
     CerenkovDetectorPMT_CollID = SDman->GetCollectionID("CerenkovPMTSD/PMTHitCollection");
  }
}

```

Figure 14.18: Source File

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

void QweakSimEventAction::EndOfEventAction(const G4Event* evt)
{

//---------------------------------------------------------------------------
// I'm playing with the QweakSimTrajectory
// Startup: LXe example
// Goal: sace track or track points into ROOT file
//
  G4TrajectoryContainer* trajectoryContainer = evt->GetTrajectoryContainer();

  G4int n_trajectories = 0;

  if (trajectoryContainer) n_trajectories = trajectoryContainer->entries();
  G4cout << "QweakSimEventAction::EndOfEventAction, we have so many trajectories stored :"<< n_trajectories << G4endl;


  // extract the trajectories and draw them
  if (G4VVisManager::GetConcreteInstance()){

     G4cout << "Inside G4VVisManager::GetConcreteInstance()"<< G4endl;

     for (G4int i=0; i<n_trajectories; i++){

   QweakSimTrajectory* trj = (QweakSimTrajectory*) ((*(evt->GetTrajectoryContainer()))[i]);

   //trj->SetForceNoDrawTrajectory(false);


//          if( trj->GetParticleDefinition() == G4OpticalPhoton ::OpticalPhotonDefinition() ){
//             //trj->SetForceDrawTrajectory(true);
//             trj->SetForceNoDrawTrajectory(true);
//          }
//          else {
//             trj->SetForceNoDrawTrajectory(false);
//        }
//

 //  trj->ShowTrajectory();
          trj->DrawTrajectory(50);

     }
  }

//---------------------------------------------------------------------------


  // preset variables for hit collection
  Initialize();

  // Get current Event Number
  G4int event_id = evt->GetEventID();

  G4HCofThisEvent * HCE = evt->GetHCofThisEvent();

  // initialize HitsCollection pointers
  QweakSimGEM_WirePlane_HitsCollection*                 GEM_WirePlane_HC                 = 0;
  QweakSimHDC_WirePlane_HitsCollection*                 HDC_WirePlane_HC                 = 0;
  QweakSimVDC_WirePlane_HitsCollection*                 VDC_WirePlane_HC                 = 0;
  QweakSimVDC_DriftCellHitsCollection*                  VDC_DriftCellFront_HC            = 0;
  QweakSimVDC_DriftCellHitsCollection*                  VDC_DriftCellBack_HC             = 0;
  QweakSimTriggerScintillator_DetectorHitsCollection*   TriggerScintillatorDetector_HC   = 0;
  //QweakSimTriggerScintillator_PMTHitsCollection*      TriggerScintillatorPMT_HC        = 0;
  QweakSimCerenkovDetectorHitsCollection*               CerenkovDetector_HC              = 0;
  QweakSimCerenkovDetector_PMTHitsCollection*           CerenkovDetectorPMT_HC           = 0;
```

Figure 14.19: Source File

```
if(HCE){

  // get  GEM_WirePlane Hit Collector pointer
  GEM_WirePlane_HC       = (QweakSimGEM_WirePlane_HitsCollection*)(HCE->GetHC(GEM_WirePlane_CollID));

    // get  HDC_WirePlane Hit Collector pointer
  HDC_WirePlane_HC       = (QweakSimHDC_WirePlane_HitsCollection*)(HCE->GetHC(HDC_WirePlane_CollID));

  // get  VDC_WirePlane Hit Collector pointer
  VDC_WirePlane_HC       = (QweakSimVDC_WirePlane_HitsCollection*)(HCE->GetHC(VDC_WirePlane_CollID));

  // get  VDC_DriftCellFront Hit Collector pointer
  VDC_DriftCellFront_HC  = (QweakSimVDC_DriftCellHitsCollection*)(HCE->GetHC(VDC_DriftCellFront_CollID));

  // get  VDC_DriftCellFront Hit Collector pointer
  VDC_DriftCellBack_HC   = (QweakSimVDC_DriftCellHitsCollection*)(HCE->GetHC(VDC_DriftCellBack_CollID));

  // get  TriggerScintillator Hit Collector pointer
  TriggerScintillatorDetector_HC  = (QweakSimTriggerScintillator_DetectorHitsCollection*)(HCE->GetHC(TriggerScintillatorDetector_CollID));

  // get  TriggerScintillatorPMT Hit Collector pointer
  //TriggerScintillatorPMT_HC = (QweakSimTriggerScintillator_PMTHitsCollection*)(HCE->GetHC(TriggerScintillatorPMT_CollID));

  // get  CerenkovDetector Hit Collector pointer
  CerenkovDetector_HC    = (QweakSimCerenkovDetectorHitsCollection*)(HCE->GetHC(CerenkovDetector_CollID));

  // get  CerenkovDetectorPMT Hit Collector pointer
  CerenkovDetectorPMT_HC = (QweakSimCerenkovDetector_PMTHitsCollection*)(HCE->GetHC(CerenkovDetectorPMT_CollID));
}


// Get number of entries for this event
n_GEMhitWirePlane             = GEM_WirePlane_HC             -> entries();
n_HDChitWirePlane             = HDC_WirePlane_HC             -> entries();
n_VDChitWirePlane             = VDC_WirePlane_HC             -> entries();
n_VDChitDCFront               = VDC_DriftCellFront_HC        -> entries();
n_VDChitDCBack                = VDC_DriftCellBack_HC         -> entries();
n_hitTriggerScintillator      = TriggerScintillatorDetector_HC -> entries();
//n_hitTriggerScintillatorPMT    = TriggerScintillatorPMT_HC      -> entries();
n_hitCerenkov                 = CerenkovDetector_HC          -> entries();
n_hitCerenkovPMT              = CerenkovDetectorPMT_HC       -> entries();

cout << "Number of hit in the GEMs         = " << n_GEMhitWirePlane        << endl;
cout << "Number of hit in the HDCs         = " << n_HDChitWirePlane        << endl;
cout << "Number of hit in the VDCs         = " << n_VDChitWirePlane        << endl;
cout << "Number of hit in the VDC DC Front  = " << n_VDChitDCFront          << endl;
cout << "Number of hit in the VDC DC Back   = " << n_VDChitDCBack           << endl;
cout << "Number of hit in the TS           = " << n_hitTriggerScintillator << endl;
cout << "Number of hit in the Cerenkov     = " << n_hitCerenkov            << endl;



// Initialize/Clear Event variables, initialize Cerenkov Detector with NoHit Flag
  for (int noctant=0;noctant<8;noctant++) {
analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].Detector.Initialize();
analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.Initialize();

analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].Detector.StoreDetectorHasBeenHit(0);
  }

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

Figure 14.20: Source File

```
// Initialize/Clear Event variables in Region 1
analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.Initialize();
analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.Initialize();
//
// initialize Region 1 readout plane with NoHit Flag
analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneHasBeenHit(0);
analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneHasBeenHit(0);
//------------------------------------------------------------------------------------------------------------------------------------------


//------------------------------------------------------------------------------------------------------------------------------------------
// Initialize/Clear Event variables in Region 2
analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane2.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane3.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane4.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane5.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane6.Initialize();
//
analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane2.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane3.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane4.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane5.Initialize();
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane6.Initialize();
//
// initialize Region 2 wire planes (6: xuv x'u'v') with NoHit Flag
//
analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane2.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane3.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane4.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane5.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane6.StorePlaneHasBeenHit(0);
//
analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane2.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane3.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane4.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane5.StorePlaneHasBeenHit(0);
//   analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane6.StorePlaneHasBeenHit(0);
//------------------------------------------------------------------------------------------------------------------------------------------


//------------------------------------------------------------------------------------------------------------------------------------------
// initialize Region 3 wire planes (2: u,v ) with NoHit Flag
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneHasBeenHit(0);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneHasBeenHit(0);
//
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneHasBeenHit(0);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneHasBeenHit(0);
//
// initialize DriftCells with NoHit Flag
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.DriftCell.StoreUDriftCellHasBeenHit(0);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.DriftCell.StoreVDriftCellHasBeenHit(0);
//
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.DriftCell.StoreUDriftCellHasBeenHit(0);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.DriftCell.StoreVDriftCellHasBeenHit(0);
//------------------------------------------------------------------------------------------------------------------------------------------

//------------------------------------------------------------------------------------------------------------------------------------------

// Initialize/Clear Event variables, initialize TriggerScintillator with NoHit Flag
  for (int ndet=0;ndet<2;ndet++) {
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[ndet].Initialize();
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[ndet].StoreDetectorHasBeenHit(0);
  }
```

Figure 14.21: Source File

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

//######################################################################################################################################################
//######################################################################################################################################################
//
//
//                        ==============================================================
//                         The Main "Software DAQ Trigger": setting the coincidence level
//
//                         or: what is required for filling the Root ntuple for this event
//                        ===============================================================
//
//######################################################################################################################################################
//######################################################################################################################################################
//
// if ( (n_hitWirePlane == 4)&&(n_hitDCFront >0)&&(n_hitDCBack >0)&&(n_hitCerenkov >0) )      // ask for 4 fold coincidence
// if ( (n_VDChitWirePlane >= 2)&&(n_VDChitDCFront >0)&&(n_VDChitDCBack >0) )                        //  ask for 3 fold coincidence
if (n_hitCerenkov > 0)
//     if (n_GEMhitWirePlane > 0)  // Triggering on GEM only

// if (n_hitTriggerScintillator > 0) // Qweak triggers DAQ on a hit in the trigger scintillator
  {


  //=========================================
  // Store Primary Information into /Primary
  //=========================================


  G4PrimaryParticle* primary = evt->GetPrimaryVertex(0)->GetPrimary(0);
  //-----------------------------------------------------------------------------------------

  OriginVertexMomentumDirectionX = primary->GetMomentum().x();
  OriginVertexMomentumDirectionY = primary->GetMomentum().y();
  OriginVertexMomentumDirectionZ = primary->GetMomentum().z();

  rOriginVertexMomentumDirectionX = (Float_t) OriginVertexMomentumDirectionX/MeV;
  rOriginVertexMomentumDirectionY = (Float_t) OriginVertexMomentumDirectionY/MeV;
  rOriginVertexMomentumDirectionZ = (Float_t) OriginVertexMomentumDirectionZ/MeV;

  analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexMomentumDirectionX( rOriginVertexMomentumDirectionX );
  analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexMomentumDirectionY( rOriginVertexMomentumDirectionY );
  analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexMomentumDirectionZ( rOriginVertexMomentumDirectionZ );


  //-----------------------------------------------------------------------------------------
  OriginVertexPositionX = evt->GetPrimaryVertex(0)->GetPosition().x();
  OriginVertexPositionY = evt->GetPrimaryVertex(0)->GetPosition().y();
  OriginVertexPositionZ = evt->GetPrimaryVertex(0)->GetPosition().z();

  rOriginVertexPositionX = (Float_t) OriginVertexPositionX/mm;
  rOriginVertexPositionY = (Float_t) OriginVertexPositionY/mm;
  rOriginVertexPositionZ = (Float_t) OriginVertexPositionZ/mm;

  analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexPositionX( rOriginVertexPositionX );
```

Figure 14.22: Source File

```
analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexPositionY( rOriginVertexPositionY );
analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexPositionZ( rOriginVertexPositionZ );

//-------------------------------------------------------------------------------------------
// my phi determination that really works ....
// (Beware: atan2 returns the arctangent of Y/X in the range -PI to PI)
// see also http://root.cern.ch/root/html/TVector2.h
OriginVertexPhiAngle = (TMath::ATan2(-1.0*rOriginVertexMomentumDirectionY, -1.0*rOriginVertexMomentumDirectionX) )*TMath::RadToDeg()*degree + 90.0*degree;
rOriginVertexPhiAngle = OriginVertexPhiAngle/degree;

analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexPhiAngle( rOriginVertexPhiAngle );

//-------------------------------------------------------------------------------------------
OriginVertexThetaAngle = (TMath::ATan2( rOriginVertexMomentumDirectionY, rOriginVertexMomentumDirectionZ) )*TMath::RadToDeg()*degree;
rOriginVertexThetaAngle = OriginVertexThetaAngle/degree;

analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexThetaAngle( rOriginVertexThetaAngle );


//-------------------------------------------------------------------------------------------
rOriginVertexKineticEnergy = TMath::Sqrt(    rOriginVertexMomentumDirectionX * rOriginVertexMomentumDirectionX
                                   + rOriginVertexMomentumDirectionY * rOriginVertexMomentumDirectionY
                                   + rOriginVertexMomentumDirectionZ * rOriginVertexMomentumDirectionZ );

analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexKineticEnergy( rOriginVertexKineticEnergy ); //MeV
//-------------------------------------------------------------------------------------------

OriginVertexParticleMass = evt->GetPrimaryVertex(0)->GetPrimary(0)->GetMass();
rOriginVertexParticleMass = OriginVertexParticleMass/MeV;

rOriginVertexTotalEnergy = TMath::Sqrt(rOriginVertexKineticEnergy*rOriginVertexKineticEnergy + rOriginVertexParticleMass*rOriginVertexParticleMass );

analysis->QweakSimG4_RootEvent->Primary.StoreOriginVertexTotalEnergy( rOriginVertexTotalEnergy ); //MeV

//-------------------------------------------------------------------------------------------
OriginVertexPDGcode = evt->GetPrimaryVertex(0)->GetPrimary(0)->GetPDGcode();
rOriginVertexPDGcode = (Int_t) OriginVertexPDGcode;

analysis->QweakSimG4_RootEvent->Primary.StorePDGcode( rOriginVertexPDGcode );
//-------------------------------------------------------------------------------------------

G4double primaryQ2          = myUserInfo->GetPrimaryQ2();
G4double crossSectionWeight = myUserInfo->GetCrossSectionWeight();
G4int    primaryEventNumber = myUserInfo->GetPrimaryEventNumber();

analysis->QweakSimG4_RootEvent->Primary.StorePrimaryQ2          ( (Float_t) primaryQ2);
analysis->QweakSimG4_RootEvent->Primary.StoreCrossSectionWeight ( (Float_t) crossSectionWeight);
analysis->QweakSimG4_RootEvent->Primary.StorePrimaryEventNumber ( (Int_t)   primaryEventNumber);

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

//============================================
// Store Number Of Hits of each Detector
//============================================


// Store Number of Hits for: UPlane DriftCell of Front Chamber
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.DriftCell.StoreUDriftCellNbOfHits(n_VDChitDCFront);

// Store Number of Hits for: VPlane DriftCell of Front Chamber
```

Figure 14.23: Source File

```
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.DriftCell.StoreVDriftCellNbOfHits(n_VDChitDCBack);

// Store Number of Hits for: Cerenkov Detector
//analysis->QweakSimG4_RootEvent->Cerenkov.Detector.StoreDetectorNbOfHits(n_hitCerenkov);


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


//=========================================
// Store VDC Hit Information into /Region3
//=========================================


// loop over wire plane hits
for(int i1=0;i1<n_VDChitWirePlane;i1++){

  // get hit pointer for each hit
  QweakSimVDC_WirePlaneHit* aHit = (*VDC_WirePlane_HC)[i1];

  //aHit->Print();

  // get local position of hit
   localPosition  = aHit->GetLocalPosition();
  rLocalPositionX = (Float_t) localPosition.x()/cm;
  rLocalPositionY = (Float_t) localPosition.y()/cm;
  rLocalPositionZ = (Float_t) localPosition.z()/cm;

  // get world position of hit
   globalPosition  = aHit->GetWorldPosition();
  rGlobalPositionX = (Float_t) globalPosition.x()/cm;
  rGlobalPositionY = (Float_t) globalPosition.y()/cm;
  rGlobalPositionZ = (Float_t) globalPosition.z()/cm;

  // get local Momentum of hit
   localMomentum  = aHit->GetLocalMomentum();
  rLocalMomentumX = (Float_t) localMomentum.x()/MeV;
  rLocalMomentumY = (Float_t) localMomentum.y()/MeV;
  rLocalMomentumZ = (Float_t) localMomentum.z()/MeV;

  // get world Momentum of hit
   globalMomentum  = aHit->GetWorldMomentum();
  rGlobalMomentumX = (Float_t) globalMomentum.x()/MeV;
  rGlobalMomentumY = (Float_t) globalMomentum.y()/MeV;
  rGlobalMomentumZ = (Float_t) globalMomentum.z()/MeV;


  // get total Energy of hit
   totalEnergy     = aHit->GetTotalEnergy();
  rtotalEnergy     = (Float_t) totalEnergy/MeV;

  // get kinetic Energy of hit
   kineticEnergy     = aHit->GetKineticEnergy();
  rkineticEnergy     = (Float_t) kineticEnergy/MeV;


  originVertexPosition  = aHit->GetOriginVertexPosition();
  rOriginVertexPositionX      = (Float_t) originVertexPosition.x()/cm;
  rOriginVertexPositionY      = (Float_t) originVertexPosition.y()/cm;
  rOriginVertexPositionZ      = (Float_t) originVertexPosition.z()/cm;


  originVertexMomentumDirection = aHit->GetOriginVertexMomentumDirection();

```

Figure 14.24: Source File

```
      originVertexKineticEnergy = aHit->GetOriginVertexKineticEnergy();
      rOriginVertexKineticEnergy = (Float_t ) originVertexKineticEnergy/MeV;

      primaryQ2  = aHit->GetPrimaryQ2();
      rPrimaryQ2  = (Float_t) primaryQ2;

      crossSectionWeight = aHit->GetCrossSectionWeight();
      rCrossSectionWeight = (Float_t) crossSectionWeight;

      primaryEventNumber = aHit->GetPrimaryEventNumber();
      rPrimaryEventNumber = (Int_t) primaryEventNumber;


      globalTime = aHit->GetGlobalTime();
      rGlobalTime = (Float_t) globalTime/ns;

       GlobalThetaAngle = globalMomentum.theta();
      rGlobalThetaAngle = (Float_t) GlobalThetaAngle/degree;

       GlobalPhiAngle   = globalMomentum.phi() -90.0*degree;
      rGlobalPhiAngle   = (Float_t) GlobalPhiAngle/degree;

//      G4cout <<"%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" << G4endl;
//      G4cout <<"  Global Theta Angle =  " << GlobalThetaAngle / degree << G4endl;
//      G4cout <<"  Global Phi   Angle =  " << GlobalPhiAngle   / degree << G4endl;
//      G4cout <<"%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%" << G4endl;


       particleName = aHit->GetParticleName();
      rParticleName = TString(particleName);

       particleType = aHit->GetParticleType();
      rParticleType = (Int_t) particleType;

//      G4cout <<" VDC wire Plane was hit by  =  " <<  particleName << G4endl;
//      G4cout <<" VDC wire Plane was hit by  =  " << rParticleName << G4endl;

      //----------------------------------
      // Hit in Front VDC, First WirePlane
      //----------------------------------
      if((aHit->GetVDCID()==0) && (aHit->GetWirePlaneID()==0)) {

   // mark wire plane as been hit
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneHasBeenHit(5);

        analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreParticleName(rParticleName);
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreParticleType(rParticleType);

   // store total+kinetic energy of hit
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreTotalEnergy(rtotalEnergy);
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreKineticEnergy(rkineticEnergy);

   // store origin vertex info
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreOriginVertexPositionX(rOriginVertexPositionX);
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreOriginVertexPositionY(rOriginVertexPositionY);
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreOriginVertexPositionZ(rOriginVertexPositionZ);

   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);

   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StorePrimaryQ2(rPrimaryQ2);
   analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreCrossSectionWeight(rCrossSectionWeight);

```

Figure 14.25: Source File

```
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StorePrimaryEventNumber(rPrimaryEventNumber);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreGlobalTimeOfHit(rGlobalTime);
// store wire plane hit position
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneLocalPositionX(rLocalPositionX);

analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneLocalPositionY(rLocalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneLocalPositionZ(rLocalPositionZ);

analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalPositionX(rGlobalPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalPositionY(rGlobalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalPositionZ(rGlobalPositionZ);
// store wire plane hit momentum
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneLocalMomentumX(rLocalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneLocalMomentumY(rLocalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneLocalMomentumZ(rLocalMomentumZ);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalMomentumX(rGlobalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalMomentumY(rGlobalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalMomentumZ(rGlobalMomentumZ);
// store global track angles Phi and Theta
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalPhiAngle(rGlobalPhiAngle);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreUPlaneGlobalThetaAngle(rGlobalThetaAngle);
  }
  //----------------------------------
  // Hit in Front VDC, Second WirePlane
  //----------------------------------
  if((aHit->GetVDCID()==0) && (aHit->GetWirePlaneID()==1)) { // Front VDC, Back Wireplane
// mark wire plane as been hit
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneHasBeenHit(5);
    analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneLocalPositionX(rLocalPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneLocalPositionY(rLocalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneLocalPositionZ(rLocalPositionZ);

analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalPositionX(rGlobalPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalPositionY(rGlobalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalPositionZ(rGlobalPositionZ);
// store wire plane hit momentum
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneLocalMomentumX(rLocalMomen
```

Figure 14.26: Source File

```
tumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneLocalMomentumY(rLocalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneLocalMomentumZ(rLocalMomentumZ);

analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalMomentumX(rGlobalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalMomentumY(rGlobalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalMomentumZ(rGlobalMomentumZ);

// store global track angles Phi and Theta
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalPhiAngle(rGlobalPhiAngle);
analysis->QweakSimG4_RootEvent->Region3.ChamberFront.WirePlane.StoreVPlaneGlobalThetaAngle(rGlobalThetaAngle);

    }


    //----------------------------------
    // Hit in Back VDC, First WirePlane
    //----------------------------------
    if((aHit->GetVDCID()==1) && (aHit->GetWirePlaneID()==0)) {

// mark wire plane as been hit
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneHasBeenHit(5);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreParticleName(rParticleName);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreParticleType(rParticleType);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreOriginVertexPositionX(rOriginVertexPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreOriginVertexPositionY(rOriginVertexPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreOriginVertexPositionZ(rOriginVertexPositionZ);

// store total+kinetic energy of hit
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreTotalEnergy(rtotalEnergy);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreKineticEnergy(rkineticEnergy);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StorePrimaryQ2(rPrimaryQ2);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreCrossSectionWeight(rCrossSectionWeight);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StorePrimaryEventNumber(rPrimaryEventNumber);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreGlobalTimeOfHit(rGlobalTime);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneLocalPositionX(rLocalPositionX);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneLocalPositionY(rLocalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneLocalPositionZ(rLocalPositionZ);


analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalPositionX(rGlobalPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalPositionY(rGlobalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalPositionZ(rGlobalPositionZ);

```

Figure 14.27: Source File

```
// store wire plane hit momentum
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneLocalMomentumX(rLocalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneLocalMomentumY(rLocalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneLocalMomentumZ(rLocalMomentumZ);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalMomentumX(rGlobalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalMomentumY(rGlobalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalMomentumZ(rGlobalMomentumZ);

// store global track angles Phi and Theta
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalPhiAngle(rGlobalPhiAngle);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreUPlaneGlobalThetaAngle(rGlobalThetaAngle);


    }


    //----------------------------------
    // Hit in Back VDC, Second WirePlane
    //----------------------------------
    if((aHit->GetVDCID()==1) && (aHit->GetWirePlaneID()==1)) {

  // mark wire plane as been hit
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneHasBeenHit(5);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneLocalPositionX(rLocalPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneLocalPositionY(rLocalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneLocalPositionZ(rLocalPositionZ);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalPositionX(rGlobalPositionX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalPositionY(rGlobalPositionY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalPositionZ(rGlobalPositionZ);

// store wire plane hit momentum
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneLocalMomentumX(rLocalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneLocalMomentumY(rLocalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneLocalMomentumZ(rLocalMomentumZ);

analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalMomentumX(rGlobalMomentumX);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalMomentumY(rGlobalMomentumY);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalMomentumZ(rGlobalMomentumZ);

// store global track angles Phi and Theta
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalPhiAngle(rGlobalPhiAngle);
analysis->QweakSimG4_RootEvent->Region3.ChamberBack.WirePlane.StoreVPlaneGlobalThetaAngle(rGlobalThetaAngle);
```

Figure 14.28: Source File

```
      }

    }

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

    //----------------------------------
    // Hit in Front VDC, Front DriftCells
    //----------------------------------
    if(n_VDChitDCFront){

  // loop over DriftCell hits
  for(G4int i1=0;i1<n_VDChitDCFront;i1++)
  {

     QweakSimVDC_DriftCellHit* aHit = (*VDC_DriftCellFront_HC)[i1];

     //aHit->Print();

  } // end  for(int i1=0;i1<n_hitDCFront;i1++)


  // Extract the DriftCell Config from the 1st DC hit
  QweakSimVDC_DriftCellHit* aHit = (*VDC_DriftCellFront_HC)[0];


  rDCWidthOnFrame    = (Float_t) aHit->GetDCWidthOnFrame()/mm;
  rDCFullThickness   = (Float_t) aHit->GetDCFullThickness()/mm;
  rDCUPlaneWireAngle = (Float_t) aHit->GetDCUPlaneWireAngle()/degree;
  rDCVPlaneWireAngle = (Float_t) aHit->GetDCVPlaneWireAngle()/degree;

  // Store DriftCell Setup Parameter
  analysis->QweakSimG4_RootEvent->Region3.Config.StoreDCWidthOnFrame(rDCWidthOnFrame);
  analysis->QweakSimG4_RootEvent->Region3.Config.StoreDCFullThickness(rDCFullThickness);
  analysis->QweakSimG4_RootEvent->Region3.Config.StoreDCUPlaneWireAngle(rDCUPlaneWireAngle);
  analysis->QweakSimG4_RootEvent->Region3.Config.StoreDCVPlaneWireAngle(rDCVPlaneWireAngle);

    } // end of   if(n_VDChitDCFront)


    //----------------------------------
    // Hit in Front VDC, Back DriftCells
    //----------------------------------
    if(n_VDChitDCBack){
      // loop over hits
  for(G4int i1=0;i1<n_VDChitDCBack;i1++)
  {

   QweakSimVDC_DriftCellHit* aHit = (*VDC_DriftCellBack_HC)[i1];

   //aHit->Print();

  } // end for(int i1=0;i1<n_hitBack;i1++


    } // end of if(n_VDChitDCBack)



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

    //==========================================================
    // Store Cerenkov Detector hits into /Cerenkov
    //==========================================================

```

Figure 14.29: Source File

```
    if (n_hitCerenkov >0){
      // loop over hits
      for(int i1=0;i1<n_hitCerenkov;i1++){

  QweakSimCerenkov_DetectorHit* aHit = (*CerenkovDetector_HC)[i1];

  rOctantID = G4IndexToOctantNumber[ (Int_t) aHit->GetDetectorID()];

//        //aHit->Print();

  // get local position of hit
  localPosition  = aHit->GetLocalPosition();
  rLocalPositionX = (Float_t) localPosition.x()/cm;
  rLocalPositionY = (Float_t) localPosition.y()/cm;
  rLocalPositionZ = (Float_t) localPosition.z()/cm;

  // get world position of hit
  globalPosition  = aHit->GetWorldPosition();
  rGlobalPositionX = (Float_t) globalPosition.x()/cm;
  rGlobalPositionY = (Float_t) globalPosition.y()/cm;
  rGlobalPositionZ = (Float_t) globalPosition.z()/cm;

  // get local Momentum of hit
  localMomentum  = aHit->GetLocalMomentum();
  rLocalMomentumX = (Float_t) localMomentum.x()/MeV;
  rLocalMomentumY = (Float_t) localMomentum.y()/MeV;
  rLocalMomentumZ = (Float_t) localMomentum.z()/MeV;

  // get world Momentum of hit
  globalMomentum  = aHit->GetWorldMomentum();
  rGlobalMomentumX = (Float_t) globalMomentum.x()/MeV;
  rGlobalMomentumY = (Float_t) globalMomentum.y()/MeV;
  rGlobalMomentumZ = (Float_t) globalMomentum.z()/MeV;

  localExitPosition = myUserInfo->GetLocalCerenkovExitPosition();
  rLocalExitPositionX = (Float_t) localExitPosition.x()/cm;
  rLocalExitPositionY = (Float_t) localExitPosition.y()/cm;
  rLocalExitPositionZ = (Float_t) localExitPosition.z()/cm;

  originVertexPosition  = aHit->GetOriginVertexPosition();
  rOriginVertexPositionX      = (Float_t) originVertexPosition.x()/cm;
  rOriginVertexPositionY      = (Float_t) originVertexPosition.y()/cm;
  rOriginVertexPositionZ      = (Float_t) originVertexPosition.z()/cm;

  originVertexMomentumDirection = aHit->GetOriginVertexMomentumDirection();

  originVertexKineticEnergy =   aHit->GetOriginVertexKineticEnergy();
  rOriginVertexKineticEnergy = (Float_t ) originVertexKineticEnergy/MeV;

  originVertexTotalEnergy =   aHit->GetOriginVertexTotalEnergy();
  rOriginVertexTotalEnergy = (Float_t ) originVertexTotalEnergy/MeV;

  primaryQ2  = aHit->GetPrimaryQ2();
  rPrimaryQ2  = (Float_t) primaryQ2;

  crossSectionWeight =  aHit->GetCrossSectionWeight();
  rCrossSectionWeight = (Float_t) crossSectionWeight;


  globalTime = aHit->GetGlobalTime();
  rGlobalTime = (Float_t) globalTime/ns;


   GlobalThetaAngle = globalMomentum.theta();
        rGlobalThetaAngle = (Float_t) GlobalThetaAngle/degree;

```

Figure 14.30: Source File

```
 GlobalPhiAngle   = globalMomentum.phi() -90.0*degree;
rGlobalPhiAngle   = (Float_t) GlobalPhiAngle/degree;


 particleName = aHit->GetParticleName();
rParticleName = TString(particleName);

 particleType = aHit->GetParticleType();
rParticleType = (Int_t) particleType;

// get total Energy of hit
 totalEnergy     = aHit->GetTotalEnergy();
rtotalEnergy     = (Float_t) totalEnergy/MeV;

// get kinetic Energy of hit
 kineticEnergy     = aHit->GetKineticEnergy();
rkineticEnergy     = (Float_t) kineticEnergy/MeV;


//          edgeEvent = myUserInfo->GetEdgeEventDetected();

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

    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorID(rOctantID);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorHasBeenHit(5);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreParticleName(rParticleName);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreParticleType(rParticleType);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreGlobalTimeOfHit(rGlobalTime);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreOriginVertexPositionX(rOriginVertexPositionX);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreOriginVertexPositionY(rOriginVertexPositionY);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreOriginVertexPositionZ(rOriginVertexPositionZ);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreOriginVertexTotalEnergy(rOriginVertexTotalEnergy);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StorePrimaryQ2(rPrimaryQ2);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreCrossSectionWeight(rCrossSectionWeight);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalPositionX(rLocalPositionX);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalPositionY(rLocalPositionY);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalPositionZ(rLocalPositionZ);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalExitPositionX(rLocalExitPositionX);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalExitPositionY(rLocalExitPositionY);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalExitPositionZ(rLocalExitPositionZ);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorGlobalPositionX(rGlobalPositionX);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorGlobalPositionY(rGlobalPositionY);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorGlobalPositionZ(rGlobalPositionZ);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreDetectorLocalVertexTotalEnergy((Float_t ) aHit->GetTotalEnergy()/MeV);

    // store global track angles Phi and Theta
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreGlobalPhiAngle(rGlobalPhiAngle);
    analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreGlobalThetaAngle(rGlobalThetaA
```

Figure 14.31: Source File

```
ngle);

        // store total+kinetic energy of a hit
        analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreTotalEnergy(rtotalEnergy);
        analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreKineticEnergy(rkineticEnergy);

        //-----------------------------------------------------------------------------

        for(int cp = 0; cp < myUserInfo->GetCerenkovOpticalPhotonCount(); cp ++){
          analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreCerenkovPhotonEnergy((Double_
t)myUserInfo->GetCerenkovPhotonEnergyAtIndex(cp));
        }

        for(int sec = 0; sec < myUserInfo->GetCerenkovSecondaryParticleCount(); sec++){

           SecondaryParticleOrigin = myUserInfo->GetCerenkovSecondaryParticleOrigin(sec);
           rSecondaryPartOriginX = (Float_t) SecondaryParticleOrigin.x()/cm;
           rSecondaryPartOriginY = (Float_t) SecondaryParticleOrigin.y()/cm;
           rSecondaryPartOriginZ = (Float_t) SecondaryParticleOrigin.z()/cm;

           SecondaryParticleMomentum = myUserInfo->GetCerenkovSecondaryParticleMomentum(sec);
           rSecondaryPartMomentumX = (Float_t) SecondaryParticleMomentum.x()/MeV;
           rSecondaryPartMomentumY = (Float_t) SecondaryParticleMomentum.y()/MeV;
           rSecondaryPartMomentumZ = (Float_t) SecondaryParticleMomentum.z()/MeV;

           rSecondaryPartEnergy = (Float_t) myUserInfo->GetCerenkovSecondaryParticleEnergy(sec)/MeV;
           rSecondaryPartCharge = (Float_t) myUserInfo->GetCerenkovSecondaryParticleCharge(sec);

           analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.AddSecondaryParticleEvent(rSeconda
ryPartOriginX,
                                                        rSecondaryPartOriginY,
                                                        rSecondaryPartOriginZ,
                                                        rSecondaryPartMomentumX,
                                                        rSecondaryPartMomentumY,
                                                        rSecondaryPartMomentumZ,
                                                        rSecondaryPartEnergy,
                                                        rSecondaryPartCharge);

        } // end for
        //-----------------------------------------------------------------------------

        //------------------------------------------------------------------------------------------
        // Check if the track passed entirely thru the cerenkov detector without getting stuck
        // or hitting an edge
        if(GetDistance(localPosition,localExitPosition)/cm < 1.15)
           edgeEvent = 1;
        else
           edgeEvent = 0;

        analysis->QweakSimG4_RootEvent->Cerenkov.Octant[rOctantID].Detector.StoreEdgeEventFlag(edgeEvent);
        // G4cout << "Edge Event Flag = " << edgeEvent << G4endl;
        //------------------------------------------------------------------------------------------


        } // end  for(int i1=0;i1<n_hitCerenkov;i1++)
      } // end    if (n_hitCerenkov >0)


      //==========================================================
      // Store Number of Photoelectrons of Cerenkov Detector hits
      //==========================================================

      if (n_hitCerenkov >0)
        {
    // loop over hits
    for(int i1=0;i1<n_hitCerenkovPMT;i1++)
      {
```

Figure 14.32: Source File

```

    QweakSimCerenkovDetector_PMTHit* aHit = (*CerenkovDetectorPMT_HC)[i1];

    rOctantID = G4IndexToOctantNumber[(Int_t) aHit->GetDetectorID()];
    //rOctantID = G4IndexToOctantNumber[6];


    //------------------------------------------------------------------------
    if( (aHit->GetPMTID() == 0) ) // left PMT
      {
        pmtHitsLeft[rOctantID] =  pmtHitsLeft[rOctantID] +1;
        if(aHit->IsHitValid()){
          pmtNPELeft[rOctantID] += myUserInfo->GetNumberOfPhotoelectronsS20(aHit->GetPhotonEnergy()*1.0e6);
        }
      }

    if(  (aHit->GetPMTID() == 1)  ) // right PMT
      {
        pmtHitsRight[rOctantID] =  pmtHitsRight[rOctantID] +1;
        if(aHit->IsHitValid()){
          pmtNPERight[rOctantID] += myUserInfo->GetNumberOfPhotoelectronsS20(aHit->GetPhotonEnergy()*1.0e6);
        }
      }
    //------------------------------------------------------------------------


  } // end for(int i1=0;i1<n_hitCerenkovPMT;i1++)

   } //end if (n_hitCerenkov >0)


  //---------------------------------------------
  // store number of hits for left and right PMT
  //---------------------------------------------
  for(int noctant=0; noctant<8; noctant++)
 {
 analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.StorePMTLeftNbOfHits(pmtHitsLeft[noctant]);
 analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.StorePMTRightNbOfHits(pmtHitsRight[noctant]);
 analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.StorePMTTotalNbOfHits(pmtHitsLeft[noctant] + pmtHitsRight[noctant]);
 analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.StorePMTLeftNbOfPEs(pmtNPELeft[noctant]);
 analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.StorePMTRightNbOfPEs(pmtNPERight[noctant]);
 analysis->QweakSimG4_RootEvent->Cerenkov.Octant[noctant].PMT.StorePMTTotalNbOfPEs(pmtNPELeft[noctant] + pmtNPERight[noctant]);
  }

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

  //==============================
  // Store HDC hits into /Region2
  //==============================

 if (n_HDChitWirePlane > 0)
 {

    // loop over wire plane hits
    for(int i1=0;i1<n_HDChitWirePlane;i1++){

  // get hit pointer for each hit
  QweakSimHDC_WirePlaneHit* aHit = (*HDC_WirePlane_HC)[i1];

  G4cout << G4endl << "###### Printing HDC hit info within QweakSimEventAction::EndOfEventAction() " << G4endl << G4endl;
  aHit->Print();
```

Figure 14.33: Source File

```
// get local position of hit
 localPosition  = aHit->GetLocalPosition();
rLocalPositionX = (Float_t) localPosition.x()/cm;
rLocalPositionY = (Float_t) localPosition.y()/cm;
rLocalPositionZ = (Float_t) localPosition.z()/cm;

// get world position of hit
 globalPosition  = aHit->GetWorldPosition();
rGlobalPositionX = (Float_t) globalPosition.x()/cm;
rGlobalPositionY = (Float_t) globalPosition.y()/cm;
rGlobalPositionZ = (Float_t) globalPosition.z()/cm;

// get local Momentum of hit
 localMomentum  = aHit->GetLocalMomentum();
rLocalMomentumX = (Float_t) localMomentum.x()/MeV;
rLocalMomentumY = (Float_t) localMomentum.y()/MeV;
rLocalMomentumZ = (Float_t) localMomentum.z()/MeV;

// get world Momentum of hit
 globalMomentum  = aHit->GetWorldMomentum();
rGlobalMomentumX = (Float_t) globalMomentum.x()/MeV;
rGlobalMomentumY = (Float_t) globalMomentum.y()/MeV;
rGlobalMomentumZ = (Float_t) globalMomentum.z()/MeV;

 originVertexPosition  = aHit->GetOriginVertexPosition();
rOriginVertexPositionX      = (Float_t) originVertexPosition.x()/cm;
rOriginVertexPositionY      = (Float_t) originVertexPosition.y()/cm;
rOriginVertexPositionZ      = (Float_t) originVertexPosition.z()/cm;


originVertexMomentumDirection = aHit->GetOriginVertexMomentumDirection();

 originVertexKineticEnergy = aHit->GetOriginVertexKineticEnergy();
rOriginVertexKineticEnergy = (Float_t ) originVertexKineticEnergy/MeV;

 primaryQ2  = aHit->GetPrimaryQ2();
rPrimaryQ2  = (Float_t) primaryQ2;

 crossSectionWeight = aHit->GetCrossSectionWeight();
rCrossSectionWeight = (Float_t) crossSectionWeight;

 primaryEventNumber = aHit->GetPrimaryEventNumber();
rPrimaryEventNumber = (Int_t) primaryEventNumber;


 globalTime = aHit->GetGlobalTime();
rGlobalTime = (Float_t) globalTime/ns;


 GlobalThetaAngle = globalMomentum.theta();
rGlobalThetaAngle = (Float_t) GlobalThetaAngle/degree;

 GlobalPhiAngle   = globalMomentum.phi() -90.0*degree;
rGlobalPhiAngle   = (Float_t) GlobalPhiAngle/degree;

// get total Energy of hit
totalEnergy     = aHit->GetTotalEnergy();
rtotalEnergy     = (Float_t) totalEnergy/MeV;

// get kinetic Energy of hit
kineticEnergy     = aHit->GetKineticEnergy();
rkineticEnergy     = (Float_t) kineticEnergy/MeV;

//----------------------------------

if((aHit->GetHDCID()==0) && (aHit->GetWirePlaneID()==0)) {
```

Figure 14.34: Source File

```
        // mark wire plane as been hit
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneHasBeenHit(5);
        // store origin vertex info
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreOriginVertexPositionX(rOriginVertexPositionX);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreOriginVertexPositionY(rOriginVertexPositionY);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreOriginVertexPositionZ(rOriginVertexPositionZ);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePrimaryQ2(rPrimaryQ2);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreCrossSectionWeight(rCrossSectionWeight);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePrimaryEventNumber(rPrimaryEventNumber);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreGlobalTimeOfHit(rGlobalTime);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneLocalPositionX(rLocalPositionX);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneLocalPositionY(rLocalPositionY);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneLocalPositionZ(rLocalPositionZ);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneGlobalPositionX(rGlobalPositionX);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneGlobalPositionY(rGlobalPositionY);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneGlobalPositionZ(rGlobalPositionZ);
        // store wire plane hit momentum
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneLocalMomentumX(rLocalMomentumX);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneLocalMomentumY(rLocalMomentumY);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneLocalMomentumZ(rLocalMomentumZ);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneGlobalMomentumX(rGlobalMomentumX);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneGlobalMomentumY(rGlobalMomentumY);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StorePlaneGlobalMomentumZ(rGlobalMomentumZ);
        // store global track angles Phi and Theta
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreGlobalPhiAngle(rGlobalPhiAngle);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreGlobalThetaAngle(rGlobalThetaAngle);
        // store total+kinetic energy of hit
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreTotalEnergy(rtotalEnergy);
        analysis->QweakSimG4_RootEvent->Region2.ChamberFront.WirePlane1.StoreKineticEnergy(rkineticEnergy);
    } //end of if((aHit->GetHDCID()==0) && (aHit->GetWirePlaneID()==0))
```

Figure 14.35: Source File

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

if((aHit->GetHDCID()==1) && (aHit->GetWirePlaneID()==0)) {

    // mark wire plane as been hit
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneHasBeenHit(5);

    // store origin vertex info
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreOriginVertexPositionX(rOriginVertexPositionX);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreOriginVertexPositionY(rOriginVertexPositionY);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreOriginVertexPositionZ(rOriginVertexPositionZ);

    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);

    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePrimaryQ2(rPrimaryQ2);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreCrossSectionWeight(rCrossSectionWeight);

    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePrimaryEventNumber(rPrimaryEventNumber);

    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreGlobalTimeOfHit(rGlobalTime);



    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneLocalPositionX(rLocalPositionX);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneLocalPositionY(rLocalPositionY);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneLocalPositionZ(rLocalPositionZ);

    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneGlobalPositionX(rGlobalPositionX);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneGlobalPositionY(rGlobalPositionY);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneGlobalPositionZ(rGlobalPositionZ);

    // store wire plane hit momentum
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneLocalMomentumX(rLocalMomentumX);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneLocalMomentumY(rLocalMomentumY);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneLocalMomentumZ(rLocalMomentumZ);

    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneGlobalMomentumX(rGlobalMomentumX);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneGlobalMomentumY(rGlobalMomentumY);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StorePlaneGlobalMomentumZ(rGlobalMomentumZ);

    // store global track angles Phi and Theta
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreGlobalPhiAngle(rGlobalPhiAngle);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreGlobalThetaAngle(rGlobalThetaAngle);

    // store total+kinetic energy of hit
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreTotalEnergy(rtotalEnergy);
    analysis->QweakSimG4_RootEvent->Region2.ChamberBack.WirePlane1.StoreKineticEnergy(rkineticEnergy);

```

Figure 14.36: Source File

```
    } // end of if((aHit->GetHDCID()==1) && (aHit->GetWirePlaneID()==0)) {

    //----------------------------------


      } // end of  for(int i1=0;i1<n_HDChitWirePlane;i1++){

  } // end of if ( (n_HDChitWirePlane == 6)


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

    //================================
    // Store GEM hits into /Region1
    //================================

  if (n_GEMhitWirePlane > 0)
  {

      // loop over wire plane hits
      // up to now there should be only one GEM per octant
      for(int i1=0;i1<n_GEMhitWirePlane;i1++){

    // get hit pointer for each hit
    QweakSimGEM_WirePlaneHit* aHit = (*GEM_WirePlane_HC)[i1];

    G4cout << G4endl << "###### Printing GEM hit info within QweakSimEventAction::EndOfEventAction() " << G4endl << G4endl;
    aHit->Print();

    // get local position of hit
     localPosition  = aHit->GetLocalPosition();
    rLocalPositionX = (Float_t) localPosition.x()/cm;
    rLocalPositionY = (Float_t) localPosition.y()/cm;
    rLocalPositionZ = (Float_t) localPosition.z()/cm;

    // get world position of hit
     globalPosition  = aHit->GetWorldPosition();
    rGlobalPositionX = (Float_t) globalPosition.x()/cm;
    rGlobalPositionY = (Float_t) globalPosition.y()/cm;
    rGlobalPositionZ = (Float_t) globalPosition.z()/cm;

    // get local Momentum of hit
     localMomentum  = aHit->GetLocalMomentum();
    rLocalMomentumX = (Float_t) localMomentum.x()/MeV;
    rLocalMomentumY = (Float_t) localMomentum.y()/MeV;
    rLocalMomentumZ = (Float_t) localMomentum.z()/MeV;

    // get world Momentum of hit
     globalMomentum  = aHit->GetWorldMomentum();
    rGlobalMomentumX = (Float_t) globalMomentum.x()/MeV;
    rGlobalMomentumY = (Float_t) globalMomentum.y()/MeV;
    rGlobalMomentumZ = (Float_t) globalMomentum.z()/MeV;

     originVertexPosition  = aHit->GetOriginVertexPosition();
    rOriginVertexPositionX      = (Float_t) originVertexPosition.x()/cm;
    rOriginVertexPositionY      = (Float_t) originVertexPosition.y()/cm;
    rOriginVertexPositionZ      = (Float_t) originVertexPosition.z()/cm;


    originVertexMomentumDirection = aHit->GetOriginVertexMomentumDirection();

     originVertexKineticEnergy = aHit->GetOriginVertexKineticEnergy();
    rOriginVertexKineticEnergy = (Float_t ) originVertexKineticEnergy/MeV;

     primaryQ2  = aHit->GetPrimaryQ2();
```

Figure 14.37: Source File

```
  rPrimaryQ2  = (Float_t) primaryQ2;

   crossSectionWeight = aHit->GetCrossSectionWeight();
  rCrossSectionWeight = (Float_t) crossSectionWeight;

   primaryEventNumber = aHit->GetPrimaryEventNumber();
  rPrimaryEventNumber = (Int_t) primaryEventNumber;


   globalTime = aHit->GetGlobalTime();
  rGlobalTime = (Float_t) globalTime/ns;

   GlobalThetaAngle = globalMomentum.theta();
  rGlobalThetaAngle = (Float_t) GlobalThetaAngle/degree;

   GlobalPhiAngle   = globalMomentum.phi() -90.0*degree;
  rGlobalPhiAngle   = (Float_t) GlobalPhiAngle/degree;

  // get total Energy of hit
  totalEnergy     = aHit->GetTotalEnergy();
 rtotalEnergy     = (Float_t) totalEnergy/MeV;

  // get kinetic Energy of hit
  kineticEnergy     = aHit->GetKineticEnergy();
 rkineticEnergy     = (Float_t) kineticEnergy/MeV;

 //-----------------------------------

  if((aHit->GetGEMID()==0) && (aHit->GetWirePlaneID()==0)) {

      // mark wire plane as been hit
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneHasBeenHit(5);


      // store origin vertex info
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreOriginVertexPositionX(rOriginVertexPositionX);
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreOriginVertexPositionY(rOriginVertexPositionY);
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreOriginVertexPositionZ(rOriginVertexPositionZ);

      //------------------------------------------------------------------------------------------------------------------------------------

      for (int noctant=0;noctant<8;noctant++) {

      }
      //------------------------------------------------------------------------------------------------------------------------------------

      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);

      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePrimaryQ2(rPrimaryQ2);
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreCrossSectionWeight(rCrossSectionWeight);

      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePrimaryEventNumber(rPrimaryEventNumber);

      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreGlobalTimeOfHit(rGlobalTime);


      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneLocalPositionX(rLocalPositionX);
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneLocalPositionY(rLocalPositionY);
      analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneLocalPositionZ(rLocalPositionZ
```

Figure 14.38: Source File

```
);

        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneGlobalPositionX(rGlobalPositionX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneGlobalPositionY(rGlobalPositionY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneGlobalPositionZ(rGlobalPositionZ);

        // store wire plane hit momentum
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneLocalMomentumX(rLocalMomentumX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneLocalMomentumY(rLocalMomentumY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneLocalMomentumZ(rLocalMomentumZ);

        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneGlobalMomentumX(rGlobalMomentumX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneGlobalMomentumY(rGlobalMomentumY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StorePlaneGlobalMomentumZ(rGlobalMomentumZ);

        // store global track angles Phi and Theta
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreGlobalPhiAngle(rGlobalPhiAngle);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreGlobalThetaAngle(rGlobalThetaAngle);

        // store total+kinetic energy of hit
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreTotalEnergy(rtotalEnergy);
        analysis->QweakSimG4_RootEvent->Region1.ChamberFront.WirePlane.StoreKineticEnergy(rkineticEnergy);

    } //end of if((aHit->GetGEMID()==0) && (aHit->GetWirePlaneID()==0))


    //-----------------------------------

    if((aHit->GetGEMID()==1) && (aHit->GetWirePlaneID()==0)) {

        // mark wire plane as been hit
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneHasBeenHit(5);

        // store origin vertex info
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreOriginVertexPositionX(rOriginVertexPositionX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreOriginVertexPositionY(rOriginVertexPositionY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreOriginVertexPositionZ(rOriginVertexPositionZ);

        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);

        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePrimaryQ2(rPrimaryQ2);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreCrossSectionWeight(rCrossSectionWeight);

        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePrimaryEventNumber(rPrimaryEventNumber);

        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreGlobalTimeOfHit(rGlobalTime);



        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneLocalPositionX(rLocalPositionX);
```

Figure 14.39: Source File

```
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneLocalPositionY(rLocalPositionY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneLocalPositionZ(rLocalPositionZ);

        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneGlobalPositionX(rGlobalPositionX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneGlobalPositionY(rGlobalPositionY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneGlobalPositionZ(rGlobalPositionZ);

        // store wire plane hit momentum
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneLocalMomentumX(rLocalMomentumX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneLocalMomentumY(rLocalMomentumY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneLocalMomentumZ(rLocalMomentumZ);

        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneGlobalMomentumX(rGlobalMomentumX);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneGlobalMomentumY(rGlobalMomentumY);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StorePlaneGlobalMomentumZ(rGlobalMomentumZ);

        // store global track angles Phi and Theta
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreGlobalPhiAngle(rGlobalPhiAngle);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreGlobalThetaAngle(rGlobalThetaAngle);

        // store total+kinetic energy of hit
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreTotalEnergy(rtotalEnergy);
        analysis->QweakSimG4_RootEvent->Region1.ChamberBack.WirePlane.StoreKineticEnergy(rkineticEnergy);

     } // end of if((aHit->GetGEMID()==1) && (aHit->GetWirePlaneID()==0)) {

     //-----------------------------------


      } // end of  for(int i1=0;i1<n_GEMhitWirePlane;i1++){

   } // end of if ( (n_GEMhitWirePlane == 1)


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

   //==========================================================
   // Store Trigger Scintillator hits into /TriggerScintillator
   //==========================================================


   if (n_hitTriggerScintillator >0){

      // loop over hits
      for(int i1=0;i1<n_hitTriggerScintillator;i1++){

    QweakSimTriggerScintillator_DetectorHit* aHit = (*TriggerScintillatorDetector_HC)[i1];

    //aHit->Print();

    // get local position of hit
     localPosition  = aHit->GetLocalPosition();
    rLocalPositionX = (Float_t) localPosition.x()/cm;
    rLocalPositionY = (Float_t) localPosition.y()/cm;
```

Figure 14.40: Source File

```
   rLocalPositionZ = (Float_t) localPosition.z()/cm;

   // get world position of hit
    globalPosition  = aHit->GetWorldPosition();
   rGlobalPositionX = (Float_t) globalPosition.x()/cm;
   rGlobalPositionY = (Float_t) globalPosition.y()/cm;
   rGlobalPositionZ = (Float_t) globalPosition.z()/cm;

   // get local Momentum of hit
   localMomentum  = aHit->GetLocalMomentum();
   rLocalMomentumX = (Float_t) localMomentum.x()/MeV;
   rLocalMomentumY = (Float_t) localMomentum.y()/MeV;
   rLocalMomentumZ = (Float_t) localMomentum.z()/MeV;

   // get world Momentum of hit
   globalMomentum  = aHit->GetWorldMomentum();
   rGlobalMomentumX = (Float_t) globalMomentum.x()/MeV;
   rGlobalMomentumY = (Float_t) globalMomentum.y()/MeV;
   rGlobalMomentumZ = (Float_t) globalMomentum.z()/MeV;



    originVertexPosition  = aHit->GetOriginVertexPosition();
   rOriginVertexPositionX      = (Float_t) originVertexPosition.x()/cm;
   rOriginVertexPositionY      = (Float_t) originVertexPosition.y()/cm;
   rOriginVertexPositionZ      = (Float_t) originVertexPosition.z()/cm;

    originVertexMomentumDirection = aHit->GetOriginVertexMomentumDirection();

    originVertexKineticEnergy =   aHit->GetOriginVertexKineticEnergy();
   rOriginVertexKineticEnergy = (Float_t ) originVertexKineticEnergy/MeV;

    originVertexTotalEnergy =   aHit->GetOriginVertexTotalEnergy();
   rOriginVertexTotalEnergy = (Float_t ) originVertexTotalEnergy/MeV;

    primaryQ2  = aHit->GetPrimaryQ2();
   rPrimaryQ2  = (Float_t) primaryQ2;

    crossSectionWeight =  aHit->GetCrossSectionWeight();
   rCrossSectionWeight = (Float_t) crossSectionWeight;


    globalTime = aHit->GetGlobalTime();
   rGlobalTime = (Float_t) globalTime/ns;

    GlobalThetaAngle = globalMomentum.theta();
   rGlobalThetaAngle = (Float_t) GlobalThetaAngle/degree;

    GlobalPhiAngle   = globalMomentum.phi() -90.0*degree;
   rGlobalPhiAngle   = (Float_t) GlobalPhiAngle/degree;


//           edgeEvent = myUserInfo->GetEdgeEventDetected();

// mark TriggerScintillator detector as been hit
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorHasBeenHit(5);

// store global time of hit
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreGlobalTimeOfHit(rGlobalTime);

// store origin vertex info
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreOriginVertexPositionX(rOriginVertexPositionX);
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreOriginVertexPositionY(rOriginVertexPositionY);
analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreOriginVertexPositionZ(rOriginVertexPositionZ);
```

Figure 14.41: Source File

```

    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreOriginVertexKineticEnergy(rOriginVertexKineticEnergy);
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreOriginVertexTotalEnergy(rOriginVertexTotalEnergy);

    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StorePrimaryQ2(rPrimaryQ2);
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreCrossSectionWeight(rCrossSectionWeight);

    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalPositionX(rLocalPositionX);
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalPositionY(rLocalPositionY);
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalPositionZ(rLocalPositionZ);

//        analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalExitPositionX(rLocalExitPositionX);
//        analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalExitPositionY(rLocalExitPositionY);
//        analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalExitPositionZ(rLocalExitPositionZ);

    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorGlobalPositionX(rGlobalPositionX);

    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorGlobalPositionY(rGlobalPositionY);
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorGlobalPositionZ(rGlobalPositionZ);

    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreDetectorLocalVertexTotalEnergy((Float_t ) aHit->GetTotalEnergy()/MeV);

    // store global track angles Phi and Theta
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreGlobalPhiAngle(rGlobalPhiAngle);
    analysis->QweakSimG4_RootEvent->TriggerScintillator.Detector[0].StoreGlobalThetaAngle(rGlobalThetaAngle);
        //------------------------------------------------------------------------------------------

       } // end  for(int i1=0;i1<n_hitTriggerScintillator;i1++)
      } // end    if (n_hitTriggerScintillator >0)



    G4cout << "$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$" << G4endl;

      // Finally fill our event ntuple
      analysis->Fill_RootNtuple();

    } //end of if( (n_hitWirePlane == 2)&&(n_hitFront >0)&&(n_hitBack >0)&&(n_hitCerenkov >0) )

    myUserInfo->ResetCerenkovSecondaryParticleInfo();



//======================================================================
// Save the Ntuple periodically so we have some data in case of a crash

    G4int eventNumber = evt->GetEventID();

   if (eventNumber%25000 == 1) analysis->AutoSaveRootNtuple();
//======================================================================


} // end of  QweakSimEventAction::EndOfEventAction()

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimEventAction::Initialize()
{

   n_HDChitWirePlane   = 0;
   n_VDChitWirePlane   = 0;
```

Figure 14.42: Source File

```
n_VDChitDCFront     = 0;
n_VDChitDCBack      = 0;
n_hitCerenkov       = 0;
n_hitCerenkovPMT    = 0;

// get local position of hit
 localPosition   = G4ThreeVector(0.,0.,0.);
rLocalPositionX  = 0.;
rLocalPositionY  = 0.;
rLocalPositionZ  = 0.;

// get world position of hit
 globalPosition   = G4ThreeVector(0.,0.,0.);
rGlobalPositionX  = 0.;
rGlobalPositionY  = 0.;
rGlobalPositionZ  = 0.;


 originVertexPosition   = G4ThreeVector(0.,0.,0.);
rOriginVertexPositionX  = 0.;
rOriginVertexPositionY  = 0.;
rOriginVertexPositionZ  = 0.;


originVertexMomentumDirection = G4ThreeVector(0.,0.,0.);

 originVertexKineticEnergy = 0.;
rOriginVertexKineticEnergy = 0.;

 originVertexTotalEnergy = 0.;
rOriginVertexTotalEnergy = 0.;

 GlobalThetaAngle = 0.0;
rGlobalThetaAngle = 0.0;

 GlobalPhiAngle   = 0.0;
rGlobalPhiAngle   = 0.0;


 primaryQ2 = 0.;
rPrimaryQ2 = 0.;

 crossSectionWeight = 0.0;
rCrossSectionWeight = 0.0;

 primaryEventNumber = 0;
rPrimaryEventNumber = 0;


 globalTime = 0.;
rGlobalTime = 0.;

rDCWidthOnFrame     = 0.;
rDCFullThickness    = 0.;
rDCUPlaneWireAngle  = 0.;
rDCVPlaneWireAngle  = 0.;

//----------------------
pmtHitsLeft.clear();
pmtHitsLeft.resize(8);

pmtHitsRight.clear();
pmtHitsRight.resize(8);

pmtNPELeft.clear();
pmtNPELeft.resize(8);

```

Figure 14.43: Source File

```
  pmtNPERight.clear();
  pmtNPERight.resize(8);

  for (int n=0;n<8;n++) {
     pmtHitsLeft[n]  = 0;
     pmtHitsRight[n] = 0;
     pmtNPELeft[n]   = 0.0;
     pmtNPERight[n]  = 0.0; }
  //----------------------


   particleType = -1;
  rParticleType = -1;

   kineticEnergy  = 0.;
  rkineticEnergy  = 0.;

   totalEnergy  = 0.;
  rtotalEnergy  = 0.;


  // aHit->GetDetectorID() returnes the Geant4 index of the cerenkov MV copy numbers that needs to
  // be converted in intuitive octant numbers. Octant #1 is at 12o'clock
  // So here I define some sort of lookup table:

  G4IndexToOctantNumber[6] = 0; // 12o'clock octant
  G4IndexToOctantNumber[7] = 1;
  G4IndexToOctantNumber[0] = 2;
  G4IndexToOctantNumber[1] = 3;
  G4IndexToOctantNumber[2] = 4;
  G4IndexToOctantNumber[3] = 5;
  G4IndexToOctantNumber[4] = 6;
  G4IndexToOctantNumber[5] = 7;

   detectorID = 0;
   octantID   = 0;
  rOctantID   = 0;

}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4double QweakSimEventAction::GetDistance(G4ThreeVector p1,G4ThreeVector p2)
{
  return sqrt((p1.x()-p2.x())*(p1.x()-p2.x())+
       (p1.y()-p2.y())*(p1.y()-p2.y())+
       (p1.z()-p2.z())*(p1.z()-p2.z()));
}

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

//=======================================================
//   -----------------------
//  | CVS File Information |
//  -----------------------
//
//      $Revisions$
//      $Log: QweakSimEventAction.cc,v $
//      Revision 1.5  2006/05/05 21:37:16  grimm
//      Records now the kinetic and total energy of all drift chambers
//
//      Revision 1.4  2006/01/06 21:39:30  grimm
//      kineticEnergy and totalEnergy will be filled
//
//      Revision 1.3  2005/12/28 23:05:44  grimm
//      Testing: Extract trajectories collected with QweakSimTrajectory (following LXe example)
//
//      Revision 1.2  2005/12/27 19:08:00  grimm
```

Figure 14.44: Source File

```
//      - Redesign of Doxygen header containing CVS info like revision and date
//      - Added CVS revision log at the end of file
//
//

```

Figure 14.45: Source File

# Chapter 15

# Qweak Collimator and Shielding Definitions

# Chapter 16

# Tracking Action and Track History

# Chapter 17

# Stepping Action and Step by Step Data Collection

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
#ifndef QweakSimSteppingAction_h
#define QweakSimSteppingAction_h 1
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

// system includes
#include "cpp_include.h"
#include "Root_include.h"
#include "Geant4_include.hh"
#include "QweakSimUserInformation.hh"
#include "QweakSimSteppingVerbose.hh"
#include "QweakSimTrackInformation.hh"

// system classes
//class G4Step;


//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
class QweakSimSteppingAction : public G4UserSteppingAction
{

public:
  QweakSimSteppingAction(QweakSimUserInformation* myUInfo);
  ~QweakSimSteppingAction(){};

  void UserSteppingAction(const G4Step*);

  G4int GetNumOfAtRestSecondaries(){return fpSteppingManager->GetfN2ndariesAtRestDoIt();};
  G4int GetNumOfAlongStepSecondaries(){return fpSteppingManager->GetfN2ndariesAlongStepDoIt();};
  G4int GetNumOfPostStepSecondaries(){return fpSteppingManager->GetfN2ndariesPostStepDoIt();};
  G4int GetTotalNumOfSecondaries(){return GetNumOfAtRestSecondaries() + GetNumOfAlongStepSecondaries() + GetNumOfPostStepSecondaries();};

  G4int GetTrackVectorStartIndex();
  G4int GetTrackVectorSize();

  G4ParticleDefinition *GetSecondaryParticleDefinition(G4int idx);
  G4String              GetSecondaryParticleName(G4int idx);
  G4double              GetSecondaryParticleTotalEnergy(G4int idx);
  G4double              GetSecondaryParticleKineticEnergy(G4int idx);
  G4double              GetSecondaryParticleXOrigin(G4int idx);
  G4double              GetSecondaryParticleYOrigin(G4int idx);
  G4double              GetSecondaryParticleZOrigin(G4int idx);
  G4ThreeVector         GetSecondaryParticleOrigin(G4int idx);
  G4ThreeVector         GetSecondaryParticleMomentum(G4int idx);
  G4String              GetSecondaryCreatorProcessName(G4int idx);

private:

  G4TrackVector *fSecondary;
  QweakSimUserInformation* myUserInfo;

};

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

#endif
```

Figure 17.1: Header File

```

#include "QweakSimSteppingAction.hh"

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

QweakSimSteppingAction::QweakSimSteppingAction(QweakSimUserInformation* myUInfo)
{

G4cout << "###### Calling QweakSimSteppingAction::QweakSimSteppingAction() " << G4endl;

  fSecondary = NULL;
  myUserInfo = myUInfo;

G4cout << "###### Leaving QweakSimSteppingAction::QweakSimSteppingAction() " << G4endl;

}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
void QweakSimSteppingAction::UserSteppingAction(const G4Step* theStep)
{

  //G4cout << "###### Calling QweakSimSteppingAction::UserSteppingAction() " << G4endl;

  fSecondary = fpSteppingManager->GetfSecondary();
  //G4cout << " got fSecondary" << G4endl;

  G4Track*              theTrack     = theStep->GetTrack();
  G4StepPoint*          thePrePoint  = theStep->GetPreStepPoint();
  G4VPhysicalVolume*    thePrePV     = thePrePoint->GetPhysicalVolume();
  G4StepPoint*          thePostPoint = theStep->GetPostStepPoint();
  G4VPhysicalVolume*    thePostPV    = thePostPoint->GetPhysicalVolume();
  G4TouchableHistory*   theTouchable = (G4TouchableHistory*)(thePrePoint->GetTouchable());
  G4int                 ReplicaNo    = 0;
  G4ParticleDefinition* particleType = theTrack->GetDefinition();
  G4String              particleName = theTrack->GetDefinition()->GetParticleName();
  G4ProcessManager*     pm           = particleType->GetProcessManager();
  G4int                 nprocesses   = pm->GetProcessListLength();
  G4ProcessVector*      pv           = pm->GetProcessList();
  G4VSteppingVerbose   *theVerbStep  = G4VSteppingVerbose::GetInstance();
  G4double              charge       = particleType->GetPDGCharge();


  G4int nSecAtRest       = GetNumOfAtRestSecondaries();
  G4int nSecAlong        = GetNumOfAlongStepSecondaries();
  G4int nSecPost         = GetNumOfPostStepSecondaries();
  G4int nSecTotal        = GetTotalNumOfSecondaries();

  QweakSimTrackInformation* info = (QweakSimTrackInformation*)(theTrack->GetUserInformation());

  for(G4int i = GetTrackVectorSize()-nSecTotal; i < GetTrackVectorSize(); i++){

    if((*fSecondary)[i]->GetUserInformation()==0){
      QweakSimTrackInformation* infoNew = new QweakSimTrackInformation(info);

      infoNew->StoreParticleDefinition(GetSecondaryParticleDefinition(i));
      infoNew->StoreParentEnergy(theTrack->GetTotalEnergy());
      infoNew->StorePrimaryKineticEnergy(GetSecondaryParticleKineticEnergy(i));
      infoNew->StoreCerenkovHitEnergy(-1,-1.0*MeV);
      infoNew->StoreCreatorProcess(GetSecondaryCreatorProcessName(i));
      infoNew->StoreOriginVertex(GetSecondaryParticleOrigin(i));
      (*fSecondary)[i]->SetUserInformation(infoNew);
    }

    if(particleType==G4Electron::ElectronDefinition() && theTrack->GetParentID() == 0 &&
//       !strcmp(thePrePV->GetName(),"CerenkovDetector_Physical")){
       (!strcmp(thePrePV->GetName(),"QuartzBar_PhysicalRight") || !strcmp(thePrePV->GetName(),"QuartzBar_PhysicalLeft") )){
```

Figure 17.2: Source File

```
      if(GetSecondaryParticleDefinition(i) == G4OpticalPhoton::OpticalPhotonDefinition() &&
   GetSecondaryParticleTotalEnergy(i)/eV <= 4.9594){
  myUserInfo->IncrementCerenkovOpticalPhotonCount();
  myUserInfo->StoreCerenkovPhotonEnergy(GetSecondaryParticleTotalEnergy(i));
      }
    }
  }




  G4cout << "Particle Name = " << particleType->GetParticleName() << G4endl;

//   if(!strcmp(thePrePV->GetName(),"CerenkovDetector_Physical")){
//!strcmp(thePrePV->GetName(),"LightGuide_PhysicalRight") || !strcmp(thePrePV->GetName(),"LightGuide_PhysicalLeft")

  if(!strcmp(thePrePV->GetName(),"QuartzBar_PhysicalRight") || !strcmp(thePrePV->GetName(),"QuartzBar_PhysicalLeft")){

    myUserInfo->AddCerenkovEnergyDeposit(theStep->GetTotalEnergyDeposit());


    if(theTrack->GetParentID() > 0 && (particleType==G4Electron::ElectronDefinition() ||
                               particleType==G4Positron::PositronDefinition() ||
                               particleType==G4Gamma::GammaDefinition())){

//       if(!strcmp(myUserInfo->GetStoredStepVolumeName(),"CerenkovContainer_Physical") &&
  //      !strcmp(thePrePV->GetName(),"CerenkovDetector_Physical")
//        ){

      if(!strcmp(myUserInfo->GetStoredStepVolumeName(),"ActiveArea_Physical")){

  myUserInfo->StoreCerenkovSecondaryParticleInfo(theTrack->GetVertexPosition(),
                                         theTrack->GetMomentum(),
                                         theTrack->GetTotalEnergy(),
                                         charge);
      }
    }
  }



  if(particleType==G4Electron::ElectronDefinition()){

    G4ThreeVector worldPos = thePrePoint->GetPosition();
    G4ThreeVector localPos = theTouchable->GetHistory()->GetTopTransform().TransformPoint(worldPos);

    if((!strcmp(myUserInfo->GetStoredStepVolumeName(),"QuartzBar_PhysicalRight") ||
  !strcmp(myUserInfo->GetStoredStepVolumeName(),"QuartzBar_PhysicalLeft")) &&
       (!strcmp(thePrePV->GetName(),"ActiveArea_Physical") ||
  !strcmp(thePrePV->GetName(),"CerenkovMasterContainer_Physical"))){

      //     if(!strcmp(myUserInfo->GetStoredStepVolumeName(),"CerenkovDetector_Physical") &&
      //        !strcmp(thePrePV->GetName(),"CerenkovContainer_Physical")){
      myUserInfo->StoreLocalCerenkovExitPosition(localPos);

    }
  }

  QweakSimTrackInformation *TrackInfo = (QweakSimTrackInformation*)theTrack->GetUserInformation();
  G4cout << "Particle History For This Particle" << G4endl;
  G4cout << "Event ID = " << myUserInfo->GetPrimaryEventNumber() << G4endl;
  //  G4cout << "Hit ID = " << myUserInfo->GetCurrentPMTHit()->GetHitID() << G4endl;
  for(int i = 0; i < TrackInfo->GetParticleHistoryLength(); i++){
    G4cout << "Particle "<< i <<" = " << TrackInfo->GetParticleDefinitionAtIndex(i)->GetParticleName() << std::setw(9)
```

Figure 17.3: Source File

```
          << " at position " << G4BestUnit(TrackInfo->GetOriginVertex(i),"Length") << std::setw(9);
      if(i == TrackInfo->GetParticleHistoryLength()-1)
        G4cout << " Parent Eng = " << theTrack->GetTotalEnergy()/MeV;
      else
        G4cout << " Parent Eng = " << TrackInfo->GetParentEnergyAtIndex(i+1)/MeV;
      G4cout << std::setw(18);
      G4cout << " Creator Process = " << TrackInfo->GetCreatorProcessAtIndex(i);
      G4cout << std::setw(18);
      G4cout << " Kinetic Energy = " <<TrackInfo->GetPrimaryKineticEnergy();
      G4cout << " Cerenkov Hit Energy = " << TrackInfo->GetCerenkovHitEnergyAtIndex(i) << G4endl;
    }
    if(TrackInfo->GetParticleHistoryLength() > 1 &&
       TrackInfo->GetParticleDefinitionAtIndex(TrackInfo->GetParticleHistoryLength()-1) == G4Gamma::GammaDefinition() &&
       TrackInfo->GetParticleDefinitionAtIndex(TrackInfo->GetParticleHistoryLength()-2) != G4Electron::ElectronDefinition() &&
       TrackInfo->GetParticleDefinitionAtIndex(TrackInfo->GetParticleHistoryLength()-2) != G4Positron::PositronDefinition())
      {
        G4cout << "Gamma Created by " << TrackInfo->GetParticleDefinitionAtIndex(TrackInfo->GetParticleHistoryLength()-2)->GetParticleName() << G4endl;
      }

    if(particleType==G4OpticalPhoton::OpticalPhotonDefinition()){

      if(!strcmp(myUserInfo->GetStoredStepVolumeName(),"PMTEntranceWindow_Physical")){
        if(!strcmp(thePrePV->GetName(),"Cathode_Physical")){



//      G4int index = 0;
//      if(TrackInfo->GetParticleHistoryLength() < 3)
// //     myUserInfo->SetPhotonFromPrimary(TrackInfo->GetParticleDefinitionAtIndex(index));
//      else{
//        index = TrackInfo->GetParticleHistoryLength()-3;
// //     myUserInfo->SetPhotonFromParticle(TrackInfo->GetParticleDefinitionAtIndex(index));
//      }

   myUserInfo->GetCurrentPMTHit()->SetHitValid(True);
   //     theTrack->SetTrackStatus(fStopAndKill);
        }
      }
    }

    myUserInfo->StoreStepVolumeName(thePrePV->GetName());

//==================================================================
//   Stolen from GATE code:
//
//   In a few random cases, a particle gets 'stuck' in an
//   an infinite loop in the geometry. It then oscillates until GATE
//   crashes on some out-of-memory error.
//   To prevent this from happening, I've added below a quick fix where
//   particles get killed when their step number gets absurdely high

    if ( theStep->GetTrack()->GetCurrentStepNumber() > 10000 )
       theStep->GetTrack()->SetTrackStatus(fStopAndKill);
//
//==================================================================

   // G4cout << "###### Leaving QweakSimSteppingAction::UserSteppingAction() " << G4endl;

    return;
}      // end of QweakSimSteppingAction::UserSteppingAction

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

```

Figure 17.4: Source File

```
G4int QweakSimSteppingAction::GetTrackVectorStartIndex()
{
  if(!fSecondary) return -1;

  return (*fSecondary).size() - GetTotalNumOfSecondaries();
}

//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4int QweakSimSteppingAction::GetTrackVectorSize()
{
  if(!fSecondary) return 0;
  return (*fSecondary).size();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4ParticleDefinition *QweakSimSteppingAction::GetSecondaryParticleDefinition(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return NULL;

  return (*fSecondary)[idx]->GetDefinition();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
 G4String QweakSimSteppingAction::GetSecondaryParticleName(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return "undefined";

  return (*fSecondary)[idx]->GetDefinition()->GetParticleName();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4double QweakSimSteppingAction::GetSecondaryParticleTotalEnergy(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return -1;

  return (*fSecondary)[idx]->GetTotalEnergy();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4double QweakSimSteppingAction::GetSecondaryParticleKineticEnergy(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return -1;

  return (*fSecondary)[idx]->GetKineticEnergy();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4double QweakSimSteppingAction::GetSecondaryParticleXOrigin(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return 1e6;

  return (*fSecondary)[idx]->GetPosition().x();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4double QweakSimSteppingAction::GetSecondaryParticleYOrigin(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return 1e6;

  return (*fSecondary)[idx]->GetPosition().y();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4double QweakSimSteppingAction::GetSecondaryParticleZOrigin(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return 1e6;

  return (*fSecondary)[idx]->GetPosition().z();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4ThreeVector QweakSimSteppingAction::GetSecondaryParticleOrigin(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return 1e6;

```

Figure 17.5: Source File

```
  return (*fSecondary)[idx]->GetPosition();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4ThreeVector QweakSimSteppingAction::GetSecondaryParticleMomentum(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return 1e6;

  return (*fSecondary)[idx]->GetMomentumDirection();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
G4String QweakSimSteppingAction::GetSecondaryCreatorProcessName(G4int idx)
{
  if(!fSecondary || idx >= GetTrackVectorSize() || idx < GetTrackVectorStartIndex()) return "undefined";
  return (*fSecondary)[idx]->GetCreatorProcess()->GetProcessName();
}
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
```

Figure 17.6: Source File

# Chapter 18

# Material Definitions

```
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
#ifndef QweakSimMaterial_H
#define QweakSimMaterial_H 1
//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......

// system includes
#include "cpp_include.h"
//#include "Root_include.h"
#include "Geant4_include.hh"

// user includes

// system classes
//class G4Material;


//....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo......
class QweakSimMaterial
{
public:
  QweakSimMaterial();
  ~QweakSimMaterial();

public:
  void  DefineMaterials();
  G4Material* GetMaterial(G4String); //returns the material

private:


};

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

#endif
```

Figure 18.1: Header File

```
#include "QweakSimMaterial.hh"

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

QweakSimMaterial::QweakSimMaterial()
{
G4cout << G4endl << "###### Calling QweakSimMaterial::QweakSimMaterial() " << G4endl << G4endl;
}

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

QweakSimMaterial::~QweakSimMaterial()
{
G4cout << G4endl << "###### Calling/Leaving QweakSimMaterial::~QweakSimMaterial() " << G4endl << G4endl;
}

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

void QweakSimMaterial::DefineMaterials()
{
G4cout << G4endl << "###### Calling QweakSimMaterial::DefineMaterials() " << G4endl << G4endl;


  // Define required materials

  G4double A;       // atomic mass
  G4double Z;       // atomic number
  G4double density;  // density

  G4double temperature;
  G4double pressure;
  G4double fractionmass;

  G4String name;
  G4String symbol;

  G4int natoms;
  G4int ncomponents;
  G4int nelements;


  //
  // Define general elements
  //

  // G4Element describes the properties of the atoms:
  // atomic number, number of nucleons, atomic mass, shell energy ...


  // see http://pcitapiww.cern.ch/asd/geant4/G4UsersDocuments/UsersGuides/ForApplicationDeveloper/html/Detector/material.html
  //
  // The whole list is availbale at: http://physics.nist.gov/PhysRefData/Star/Text/method.html
  G4NistManager* man = G4NistManager::Instance();
  man->SetVerbose(1);


  //
  // define pure NIST materials
  //

  G4Material* Al = man->FindOrBuildMaterial("G4_Al");
  G4Material* Cu = man->FindOrBuildMaterial("G4_Cu");

  //---------------------------------------------------------
  name   = "Element_Hydrogen";
```

Figure 18.2: Source File

```
symbol = "H";
Z      = 1.;
A      = 1.00794*g/mole;
G4Element* elH = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Helium";
symbol = "He";
Z      = 2.;
A      = 4.0026*g/mole;
G4Element* elHe = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Boro";
symbol = "B";
Z      = 5.;
A      = 10.811*g/mole;
G4Element* elB = new G4Element(name, symbol, Z, A);
 //---------------------------------------------------------
name   = "Element_Carbon";
symbol = "C";
Z      = 6.;
A      = 12.011*g/mole;
G4Element* elC = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Nitrogen";
symbol = "N";
Z      = 7.;
A      = 14.01*g/mole;
G4Element* elN = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Oxygen";
symbol = "O";
Z      = 8.;
A      = 16.00*g/mole;
G4Element* elO = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Fluorine";
symbol = "F";
Z      = 9.;
A      = 19.00*g/mole;
G4Element* elF = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Sodium";
symbol = "Na";
Z      = 11.;
A      = 22.99*g/mole;
G4Element* elNa = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Magnesium";
symbol = "Mg";
Z      = 12.;
A      = 24.305*g/mole;
G4Element* elMg = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Aluminum";
symbol = "Al";
Z      = 13.;
A      = 26.981539*g/mole;
G4Element* elAl = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Silicon";
symbol = "Si";
Z      = 14.;
A      = 28.09*g/mole;
G4Element* elSi  = new G4Element(name, symbol, Z, A);
//---------------------------------------------------------
name   = "Element_Chlorine";
symbol = "Cl";
```

Figure 18.3: Source File

```
Z      = 17.;
A      = 35.453*g/mole;
G4Element* elCl = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Argon";
symbol = "Ar";
Z      = 18.;
A      = 39.95*g/mole;
G4Element* elAr  = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Potassium";
symbol = "K";
Z      = 19.;
A      = 39.0983*g/mole;
G4Element* elK = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Calcium";
symbol = "Ca";
Z      = 20.;
A      = 40.08*g/mole;
G4Element* elCa = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Chromium";
symbol = "Cr";
Z      = 24.;
A      = 52.00*g/mole;
G4Element* elCr = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Magnesium";
symbol = "Mn";
Z      = 25.;
A      = 54.94*g/mole;
G4Element* elMn = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Iron";
symbol = "Fe";
Z      = 26.;
A      = 55.85*g/mole;
G4Element* elFe  = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Nickel";
symbol = "Ni";
Z      = 28.;
A      = 58.70*g/mole;
G4Element* elNi  = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------
name   = "Element_Xenon";
symbol = "Xe";
Z      = 52.;
A      = 131.29*g/mole;
G4Element* elXe  = new G4Element(name, symbol, Z, A);
//----------------------------------------------------------

//
// define simple materials
//
//
// The G4Material class describes the macroscopic properties of the matter:
// density, state, temperature, pressure, radiation length, mean free path,
// dE/dx ...
//
// My name convention: all materials start with matXyz
//
// Sorted by Z

// Liquid H2
name     = "H2Liquid";
```

Figure 18.4: Source File

```
density  = 0.0708*g/cm3;
nelements = 1;
G4Material* matLiquidHydrogen = new G4Material(name, density, nelements);
        matLiquidHydrogen -> AddElement(elH,1);
    matLiquidHydrogen -> GetIonisation() -> SetMeanExcitationEnergy(21.8*eV);

// Helium gas
name     = "HeGas";
density  = 0.1787*mg/cm3;
nelements = 1;
G4Material* matGasHelium = new G4Material(name,density,nelements, kStateGas, 273.15*kelvin,1.*atmosphere);
        matGasHelium ->AddElement(elHe,1);

//Liquid  Helium 4
name     = "HeLiquid";
density  = 0.1249*g/cm3;
nelements = 1;
G4Material* matLiquidHelium = new G4Material(name, density, nelements);
        matLiquidHelium -> AddElement(elHe,1);


 // Al material
name     = "Aluminum";
density  = 2.700*g/cm3;
nelements = 1;
G4Material* matAl = new G4Material(name,density,nelements);
        matAl -> AddElement(elAl,1);
//          matAl -> GetIonisation() -> SetMeanExcitationEnergy(166*eV);

// gaseous Argon
name      = "ArgonGas";
density   = 1.7836*mg/cm3 ;       // STP
nelements = 1;
G4Material* matArgonGas = new G4Material(name,density,nelements, kStateGas, 273.15*kelvin,1.*atmosphere);
        matArgonGas -> AddElement(elAr, 1);
//          matArgonGas -> GetIonisation() -> SetMeanExcitationEnergy(188*eV);


// Iron material
name    = "Iron";
A       = 55.85*g/mole;
Z       = 26.;
density = 7.87*g/cm3;
G4Material* matIron = new G4Material(name,Z,A,density);

// Copper material
name    = "Copper";
A       = 63.54*g/mole;
Z       = 29.;
density = 8.96*g/cm3;
G4Material* matCopper = new G4Material(name,Z,A,density);

// Tin material
name    = "Tin";
A       = 118.69*g/mole;
Z       = 50.;
density =  7.28*g/cm3;
G4Material* matSn = new G4Material(name,Z,A,density);

// Lead material
name    = "Lead";
A       = 207.19*g/mole;
Z       = 82.;
density = 11.35*g/cm3;
G4Material* matPb = new G4Material(name,Z,A,density);


```

Figure 18.5: Source File

```
// photocathode material, approximated as elemental cesium
 name      = "Photocathode";
 density   = 5.0*g/cm3; // true??
 nelements = 1;
 G4Material * matPhotocathode =  new G4Material(name,density,nelements);
            matPhotocathode -> AddElement(elK, 1);

 //==============================================================
 //
 // define a material from elements.   case 1: chemical molecule
 //
 //==============================================================

  // Xe gas
 name        = "XenonGas";
 density     = 5.458*mg/cm3;
 ncomponents = 1;
 G4Material* matXe =  new G4Material(name,density, ncomponents, kStateGas,273.15*kelvin,1.*atmosphere);
           matXe -> AddElement(elXe,1);

 // CO2 , STP
 name        = "CO2";
 density     = 1.818*mg/cm3;
 ncomponents = 2;
 G4Material* matCO2 = new G4Material(name,density, ncomponents, kStateGas,273.15*kelvin,1.*atmosphere);
           matCO2-> AddElement(elC, natoms=1);
           matCO2-> AddElement(elO, natoms=2);
      matCO2-> GetIonisation() -> SetMeanExcitationEnergy(85*eV);

 // Water
 name        = "Water" ;
 density     = 1.000*g/cm3;
 ncomponents = 2;
 G4Material* matH2O = new G4Material(name,density ,ncomponents);
           matH2O -> AddElement(elH,natoms=2);
           matH2O -> AddElement(elO,natoms=1);
           matH2O -> GetIonisation() -> SetMeanExcitationEnergy(75.0*eV);

 // Scintillator
 name        = "Scintillator";
 density     = 1.032*g/cm3;
 ncomponents = 2;
 G4Material* matScint = new G4Material(name, density, ncomponents);
           matScint->AddElement(elC, natoms=9);
           matScint->AddElement(elH, natoms=10);


 // Quartz SiO2 (e.g. Spectrosil 2000), optical properties will be added
 name        = "Quartz";
 density     = 2.200*g/cm3;
 ncomponents = 2;
 G4Material* matQuartz = new G4Material(name,density, ncomponents);
           matQuartz->AddElement(elSi, natoms=1);
           matQuartz->AddElement(elO , natoms=2);

 // Quartz SiO2 (e.g. Spectrosil 2000) without optical properties
 name        = "SiO2";
 density     = 2.200*g/cm3;
 ncomponents = 2;
 G4Material* matSiO2 = new G4Material(name,density, ncomponents);
           matSiO2->AddElement(elSi, natoms=1);
           matSiO2->AddElement(elO , natoms=2);
```

Figure 18.6: Source File

```
// SiElast_Glue  The glue used to glue together the quartz pieces
name        = "SiElast_Glue";
density     = 2.200*g/cm3;
ncomponents = 2;
G4Material* matSiElast = new G4Material(name,density, ncomponents);
        matSiElast->AddElement(elSi, natoms=1);
        matSiElast->AddElement(elO , natoms=2);

// Lime Glass
name        = "LimeGlass";
density     = 2.200*g/cm3;
ncomponents = 2;
G4Material* matLimeGlass = new G4Material(name,density, ncomponents);
        matLimeGlass->AddElement(elSi, natoms=1);
        matLimeGlass->AddElement(elO , natoms=2);


//Mylar
name        = "Mylar";
density     = 1.397 *g/cm3;
ncomponents = 3;
G4Material* matMylar = new G4Material(name,density , ncomponents);
        matMylar -> AddElement(elH, natoms= 8);
        matMylar -> AddElement(elC, natoms=10);
        matMylar -> AddElement(elO, natoms= 4);

//Mirror
name        = "Mirror";
density     = 1.397 *g/cm3;
ncomponents = 3;
G4Material* matMirror = new G4Material(name,density , ncomponents);
        matMirror -> AddElement(elH, natoms= 8);
        matMirror -> AddElement(elC, natoms=10);
        matMirror -> AddElement(elO, natoms= 4);


// Tyvek (High density Polyethylene)
//  (...-CH2-CH2-...)*n
name        = "Tyvek";
density     = 0.96 *g/cm3;
ncomponents = 2;
G4Material* matTyvek = new G4Material(name,density , ncomponents);
        matTyvek -> AddElement(elH, natoms= 2);
        matTyvek -> AddElement(elC, natoms= 1);

// Kevlar
// (-NH-C6H4-NH-CO-C6H4-CO-)*n
name        =    "Kevlar";
density     = 1.44 *g/cm3;
ncomponents = 4;
G4Material* matKevlar = new G4Material(name,density , ncomponents);
        matKevlar -> AddElement(elH, natoms=10 );
        matKevlar -> AddElement(elC, natoms=14);
        matKevlar -> AddElement(elO, natoms= 2);
        matKevlar -> AddElement(elN, natoms= 2);

//
//    ---  H        O  -----
//       -N-(CH2)5-C-
//
name        = "Nylon";
density     = 0.805*g/cm3;
ncomponents = 4;
G4Material* matNylon = new G4Material(name,density , ncomponents);
        matNylon -> AddElement(elH, natoms=11 );
```

Figure 18.7: Source File

```
            matNylon -> AddElement(elC, natoms= 6);
            matNylon -> AddElement(elO, natoms= 1);
            matNylon -> AddElement(elN, natoms= 1);



//    H H
//  ---C-C---
//    H COOCH3
name       = "Acrylic";
density    = 1.14*g/cm3;
ncomponents = 3;
G4Material* matAcrylic = new G4Material(name,density , ncomponents);
            matAcrylic -> AddElement(elH, natoms= 6);
            matAcrylic -> AddElement(elC, natoms= 4);
            matAcrylic -> AddElement(elO, natoms= 2);

//
//  Nema grade G10 or FR4
//
name       = "NemaG10";
density    = 1.70*g/cm3;
ncomponents = 4;
G4Material* matG10 = new G4Material(name,density , ncomponents);
     matG10 -> AddElement(elSi, natoms=1);
     matG10 -> AddElement(elO , natoms=2);
     matG10 -> AddElement(elC , natoms=3);
     matG10 -> AddElement(elH , natoms=3);


//=====================================================================
//
// define a material from elements.   case 2: mixture by fractional mass
//
//=====================================================================

// Air material: Air 18 degr.C and 58% humidity
name       = "Air";
density    = 1.214*mg/cm3;
ncomponents = 4;
G4Material* matAir = new G4Material(name,density,ncomponents);
            matAir -> AddElement(elN,  fractionmass=0.7494);
            matAir -> AddElement(elO,  fractionmass=0.2369);
     matAir -> AddElement(elAr, fractionmass=0.0129);
     matAir -> AddElement(elH,  fractionmass=0.0008);
//          matAir -> GetIonisation() -> SetMeanExcitationEnergy(85.7*eV);

// Kapton
name       = "Kapton";
density    = 1.42*g/cm3;
ncomponents = 4;
G4Material* matKapton = new G4Material(name,density, ncomponents);
            matKapton -> AddElement(elH, fractionmass = 0.0273);
            matKapton -> AddElement(elC, fractionmass = 0.7213);
            matKapton -> AddElement(elN, fractionmass = 0.0765);
            matKapton -> AddElement(elO, fractionmass = 0.1749);
     matKapton -> GetIonisation() -> SetMeanExcitationEnergy(79.6*eV);


// Polyethylene
name       = "Polyethylene";
density    = 0.94 * g/cm3;
ncomponents = 2;
G4Material* matPolyethylene = new G4Material(name,density, ncomponents);
            matPolyethylene -> AddElement(elH, fractionmass=0.14);
            matPolyethylene -> AddElement(elC, fractionmass=0.86);

```

Figure 18.8: Source File

```
// Polyacrylate
name       = "Polyacrylate";
density    = 1.19 * g/cm3;
ncomponents = 3;
G4Material* matPolyacrylate = new G4Material(name,density,ncomponents);
        matPolyacrylate -> AddElement(elH, fractionmass=0.08);
        matPolyacrylate -> AddElement(elC, fractionmass=0.60);
        matPolyacrylate -> AddElement(elO, fractionmass=0.32);

// VDC ArCO2 80/20
name       = "ArCO2";
density    = 0.0018*g/cm3; // to be checked
ncomponents = 2;
G4Material* matArCO2 = new G4Material(name,density,ncomponents);
        matArCO2->AddMaterial(matArgonGas, fractionmass = 0.8);
        matArCO2->AddMaterial(matCO2,      fractionmass = 0.2);


// ShieldingConcrete: must  check recipe for concrete

name       = "ShieldingConcrete";
density    = 2.5*g/cm3;
ncomponents = 6;
G4Material* matConcrete = new G4Material(name,density,ncomponents);
        matConcrete -> AddElement(elO,  fractionmass = 0.520);
        matConcrete -> AddElement(elSi, fractionmass = 0.325);
        matConcrete -> AddElement(elCa, fractionmass = 0.060);
        matConcrete -> AddElement(elNa, fractionmass = 0.015);
        matConcrete -> AddElement(elFe, fractionmass = 0.040);
        matConcrete -> AddElement(elAl, fractionmass = 0.040);
    matConcrete -> GetIonisation() -> SetMeanExcitationEnergy(135.2*eV);


// material for the collimators: High Leaded Tin Bronze
// Copper Alloy No. C94300
// see http://www/anchorbronze.com/c94300.html
name       = "CDA943";
density    = 9.29 * g/cm3;
ncomponents = 3;
G4Material* matCollimator = new G4Material(name,density,ncomponents);
        matCollimator-> AddMaterial(matCopper, fractionmass = 0.695);
        matCollimator-> AddMaterial(matPb    , fractionmass = 0.25);
        matCollimator-> AddMaterial(matSn    , fractionmass = 0.055);


// Stainless steel (Medical Physics, Vol 25, No 10, Oct 1998)
name       = "StainlessSteel";
density    = 8.02 * g/cm3;
ncomponents = 5;
G4Material* matStainlessSteel = new G4Material(name,density,ncomponents);
        matStainlessSteel-> AddElement(elMn   , fractionmass = 0.01);
        matStainlessSteel-> AddElement(elSi   , fractionmass = 0.02);
        matStainlessSteel-> AddElement(elCr   , fractionmass = 0.19);
    matStainlessSteel-> AddElement(elNi   , fractionmass = 0.10);
    matStainlessSteel-> AddElement(elFe   , fractionmass = 0.68);


 // TRT_CH2
 name       = "CH2";
 density    = 0.935*g/cm3;
 ncomponents = 2;
 G4Material* matCH2 = new G4Material(name, density, ncomponents);
         matCH2->AddElement(elC, natoms=1);
         matCH2->AddElement(elH, natoms=2);


//vacuum
```

Figure 18.9: Source File

```
  name       = "Vacuum";
  A          = 1.01*g/mole;
  Z          = 1.;
  density    = 1.e-25 *g/cm3;
  pressure   = 3.e-18*pascal;
  temperature = 2.73*kelvin;
  G4Material* matVacuum = new G4Material("Vacuum", Z, A, density,kStateGas,temperature,pressure);


//========================================
// Hydrocarbones, metane and others
//========================================

  // CH4: Metane, STP
  name       = "Methane";
  density    = 0.7174*mg/cm3 ;
  ncomponents = 2;
  G4Material* matMetane = new G4Material(name,density,ncomponents);
            matMetane->AddElement(elC, natoms= 1) ;
            matMetane->AddElement(elH, natoms= 4) ;

  // C3H8: Propane, STP
  name       = "Propane";
  density    = 2.005*mg/cm3 ;
  ncomponents = 2;
  G4Material* matPropane = new G4Material(name,density,ncomponents);
            matPropane->AddElement(elC, natoms= 3) ;
            matPropane->AddElement(elH, natoms= 8) ;

   // C4H10 : iso-Butane (methylpropane), STP
   name       = "IsoButane";
   density    = 2.67*mg/cm3 ;
   ncomponents = 2;
   G4Material* matIsobutane = new G4Material(name,density,ncomponents);
             matIsobutane->AddElement(elC,natoms=  4) ;
             matIsobutane->AddElement(elH,natoms= 10) ;

  // C2H6 : Ethane, STP
   name       = "Ethane";
   density    = 1.356*mg/cm3 ;
   ncomponents = 2;
   G4Material* matEthane = new G4Material(name,density,ncomponents);
             matEthane -> AddElement(elC, natoms= 2) ;
             matEthane -> AddElement(elH, natoms= 6) ;
       matEthane -> GetIonisation() -> SetMeanExcitationEnergy(45.4*eV);


// Argon-Ethane 40-60 by mass, STP
   name       = "Ar-C2H6_40-60";
   density    = 1.46920*mg/cm3 ;
   ncomponents = 2;
   G4Material* matVDCGas = new G4Material(name,density,ncomponents);
             matVDCGas -> AddMaterial(matArgonGas , fractionmass = 0.40) ;
             matVDCGas -> AddMaterial(matEthane   , fractionmass = 0.60) ;


  // print out Material Table
  //G4cout << *(G4Material::GetMaterialTable()) << G4endl;

  //=====================================================================
=====================
  //                     Optical Propeties
  //=====================================================================
=====================

  const G4int nEntries = 9;

```

Figure 18.10: Source File

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

//===========================
//  Optical Properties of Air
//===========================

// exact values can be taken from KamLAND code

  G4double RefractiveIndex_Air[nEntries] =
          {     1.00, // 800 nm
           1.00, // 700 nm
           1.00, // 600 nm
           1.00, // 500 nm
           1.00, // 400 nm
           1.00, // 300 nm
           1.00, // 250 nm
                1.00, // 225 nm
           1.00  // 210 nm
     };

  // normally air is very transparent to light in the visual spectrum,
  // but there I'm suppressing the optical tracking in air:
  // Don't show the cerenkov light leakage (detector->air)
//   G4double  AbsorptionCoeff_Air[nEntries] =
//           {     1e-3*m, // 800 nm
//                 1e-3*m, // 700 nm
//                 1e-3*m, // 600 nm
//                 1e-3*m, // 500 nm
//                 1e-3*m, // 400 nm
//                 1e-3*m, // 300 nm
//                 1e-3*m, // 250 nm
//                  1e-3*m, // 225 nm
//                 1e-3*m  // 210 nm
//          };
  G4double  AbsorptionCoeff_Air[nEntries] ={
    1e1*m, // 800 nm
    1e1*m, // 700 nm
    1e1*m, // 600 nm
    1e1*m, // 500 nm
    1e1*m, // 400 nm
    1e1*m, // 300 nm
    1e1*m, // 250 nm
    1e1*m, // 225 nm
    1e1*m  // 210 nm
  };

  G4MaterialPropertiesTable* myMPT_Air = new G4MaterialPropertiesTable();
  myMPT_Air->AddProperty("RINDEX",    PhotonEnergy, RefractiveIndex_Air, nEntries);
  myMPT_Air->AddProperty("ABSLENGTH", PhotonEnergy, AbsorptionCoeff_Air, nEntries);

   matAir->SetMaterialPropertiesTable(myMPT_Air);


//========================================
//  Optical Properties of Soda Lime Glass
//========================================

```

Figure 18.11: Source File

```
 //**************************************************
 // could not find anything in the literature + web
 // about the optical properties of lime glass ...
 // so here we have only "educated guesses"
 //***************************************************

// values taken from KamLAND code

  G4double RefractiveIndex_LimeGlass[nEntries] =
       { 1.52, // 800 nm
   1.52, // 700 nm
   1.52, // 600 nm  pretty close
   1.52, // 500 nm
   1.52, // 400 nm  pretty close
   1.52, // 300 nm
   1.52, // 250 nm
   1.52, // 225 nm  fiction
   1.52  // 210 nm
     };


  G4double  AbsorptionCoeff_LimeGlass[nEntries] =
       { 1.0e3*m, // 800 nm
   1.0e3*m, // 700 nm
   1.0e3*m, // 600 nm
   1.0e3*m, // 500 nm
   1.0e3*m, // 400 nm
   1.0e3*m, // 300 nm
   1.0e3*m, // 250 nm
   1.0e3*m, // 225 nm
   1.0e3*m  // 210 nm
 };

  G4MaterialPropertiesTable* myMPT_LimeGlass = new G4MaterialPropertiesTable();
  myMPT_LimeGlass->AddProperty("RINDEX",    PhotonEnergy, RefractiveIndex_LimeGlass , nEntries);
  myMPT_LimeGlass->AddProperty("ABSLENGTH", PhotonEnergy, AbsorptionCoeff_LimeGlass , nEntries);

  matLimeGlass->SetMaterialPropertiesTable(myMPT_LimeGlass);


//====================================
//  Optical Properties of Fused Silica
//====================================

  // Fused Silica (Spectrosil 2000) for the Cerenkov Detector
  // See Elog entry #43, Software
  G4double RefractiveIndex_FusedSilica[nEntries] =
    { 1.45338,  // 800 nm
      1.45536,  // 700 nm
      1.45810,  // 600 nm
      1.46239,  // 500 nm
      1.47018,  // 400 nm
      1.48786,  // 300 nm
      1.50751,  // 250 nm
      1.52422,  // 225 nm
      1.53842   // 210 nm
    };

  // Given by the BaBar Collaboration for the DIRC bar
  // BaBar Note #220
//   G4double AbsorptionCoeff_FusedSilica[nEntries] =
//          {1/0.0038*m,   // 800 nm pi*thumb extrapolated
//          1/0.0040*m,   // 700 nm pi*thumb extrapolated
//          1/0.0044*m,   // 600 nm
//          1/0.0050*m,   // 500 nm
//          1/0.0076*m,   // 400 nm
//          1/0.0620*m,   // 300 nm
```

Figure 18.12: Source File

```
//          1/1.3500*m,   // 250 nm
//          1/8.0000*m,   // 225 nm
//            1/100.00*m    // 210 nm
//          };

  G4double AbsorptionCoeff_FusedSilica[nEntries] =
          { 263.16*m,   // 800 nm pi*thumb extrapolated
     250.00*m,   // 700 nm pi*thumb extrapolated
     227.27*m,   // 600 nm
     200.00*m,   // 500 nm
     131.58*m,   // 400 nm
      16.13*m,   // 300 nm
       0.74*m,   // 250 nm
       0.125*m,   // 225 nm
            0.010*m    // 210 nm
     };

  G4MaterialPropertiesTable* myMPT_FusedSilica = new G4MaterialPropertiesTable();
  myMPT_FusedSilica->AddProperty("RINDEX",    PhotonEnergy, RefractiveIndex_FusedSilica , nEntries);
  myMPT_FusedSilica->AddProperty("ABSLENGTH", PhotonEnergy, AbsorptionCoeff_FusedSilica , nEntries);

  matQuartz->SetMaterialPropertiesTable(myMPT_FusedSilica);

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

//====================================
//  Optical Properties of Silicon Elastomer Glue
//====================================

  // Fused Silica (Spectrosil 2000) for the Cerenkov Detector
  // See Elog entry #43, Software
  G4double RefractiveIndex_SilElast[nEntries] =
    { 1.405,  // 800 nm
      1.405,  // 700 nm
      1.405,  // 600 nm
      1.405,  // 500 nm
      1.405,  // 400 nm
      1.405,  // 300 nm
      1.405,  // 250 nm
      1.405,  // 225 nm
      1.405   // 210 nm
    };


  G4double AbsorptionCoeff_SilElast[nEntries] =
          { 263.16*m,   // 800 nm pi*thumb extrapolated
     250.00*m,   // 700 nm pi*thumb extrapolated
     227.27*m,   // 600 nm
     200.00*m,   // 500 nm
     131.58*m,   // 400 nm
      16.13*m,   // 300 nm
       0.74*m,   // 250 nm
       0.125*m,   // 225 nm
            0.010*m    // 210 nm
     };

  G4MaterialPropertiesTable* myMPT_SilElast = new G4MaterialPropertiesTable();
  myMPT_SilElast->AddProperty("RINDEX",    PhotonEnergy, RefractiveIndex_SilElast , nEntries);
  myMPT_SilElast->AddProperty("ABSLENGTH", PhotonEnergy, AbsorptionCoeff_SilElast , nEntries);

  matSiElast->SetMaterialPropertiesTable(myMPT_SilElast);

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

```

Figure 18.13: Source File

```
  G4double reflind_Mirror[nEntries] ={
    1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0
  };
  G4double refrind_Mirror[nEntries] ={
    0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0
  };

  G4double abslength_Mirror[nEntries] ={
    1.0e3*m,1.0e3*m,1.0e3*m,1.0e3*m,1.0e3*m,1.0e3*m,1.0e3*m,1.0e3*m,1.0e3*m
  };

  G4MaterialPropertiesTable* myMPT_Mirror = new G4MaterialPropertiesTable();
  myMPT_Mirror->AddProperty("REFLECTIVITY", PhotonEnergy, reflind_Mirror, nEntries);
  myMPT_Mirror->AddProperty("RINDEX",    PhotonEnergy, refrind_Mirror, nEntries);
  myMPT_Mirror->AddProperty("ABSLENGTH", PhotonEnergy, abslength_Mirror, nEntries);

  matMirror->SetMaterialPropertiesTable(myMPT_Mirror);

//   G4double reflind_Photocathode[nEntries] ={
//     1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0,1.0
//   };
//   G4double refrind_Photocathode[nEntries] ={
//     1.52,1.52,1.52,1.52,1.52,1.52,1.52,1.52,1.52
//   };

//   G4double abslength_Photocathode[nEntries] ={
//     1.0e-3*m,1.0e-3*m,1.0e-3*m,1.0e-3*m,1.0e-3*m,1.0e-3*m,1.0e-3*m,1.0e-3*m,1.0e-3*m
//   };

//   G4MaterialPropertiesTable* myMPT_Photocathode = new G4MaterialPropertiesTable();
// //   myMPT_Photocathode->AddProperty("REFLECTIVITY", PhotonEnergy, reflind_Photocathode, nEntries);
//   myMPT_Photocathode->AddProperty("RINDEX",    PhotonEnergy, refrind_Photocathode, nEntries);
//   myMPT_Photocathode->AddProperty("ABSLENGTH", PhotonEnergy, abslength_Photocathode, nEntries);

//   matPhotocathode->SetMaterialPropertiesTable(myMPT_Photocathode);

 G4cout << G4endl << "###### Leaving QweakSimMaterial::DefineMaterials() " << G4endl << G4endl;

 }

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

G4Material* QweakSimMaterial::GetMaterial(G4String material)
{
G4cout << G4endl << "###### Calling QweakSimMaterial::GetMaterial() " << G4endl << G4endl;

  G4Material* pttoMaterial = G4Material::GetMaterial(material);

  return pttoMaterial;

G4cout << G4endl << "###### Leaving QweakSimMaterial::GetMaterial() " << G4endl << G4endl;
}

```

Figure 18.14: Source File

# Chapter 19

# User Classes

```
#ifndef QweakSimUserInformation_h
#define QweakSimUserInformation_h

// system includes
#include "cpp_include.h"
#include "Root_include.h"
#include "Geant4_include.hh"
#include "QweakSimCerenkovDetector_PMTHit.hh"

class QweakSimUserInformation
{
public:

   QweakSimUserInformation();
  ~QweakSimUserInformation();

private:

  G4double primaryQ2;
  G4double crossSectionWeight;
  G4double CerEngDep;

  G4int    primaryEventNumber;
  G4int    edgeEventDetected;
  G4int    leftPMTHitValid;
  G4int    rightPMTHitValid;

  G4int    cerenkovSecondaryParticleCount;
  G4int    cerenkovSecondaryElectronCount;
  G4int    cerenkovSecondaryPhotonCount;
  G4int    cerenkovSecondaryPositronCount;
  G4int    cerenkovOpticalPhotonCount;

  G4MaterialPropertyVector* PMTQE_XP4572;
  G4MaterialPropertyVector* PMTQED753WKBS20;

  G4String StepVolumeName;
  G4ThreeVector cerenkovEventExitPos;
  G4ThreeVector *cerenkovSecondaryPartOrig;
  G4ThreeVector *cerenkovSecondaryPartMom;
  G4double *cerenkovSecondaryPartEng;
  G4double *cerenkovSecondaryPartCharge;

  vector <G4double> CerenkovPhotonEnergy;

  QweakSimCerenkovDetector_PMTHit *PMTHit;
  G4int PMTSide;

 public:

  void Print() const;
  void Initialize();

  void     IncrementCerenkovOpticalPhotonCount() {cerenkovOpticalPhotonCount++;}
  G4int    GetCerenkovOpticalPhotonCount() {return cerenkovOpticalPhotonCount;}

  void     SetPrimaryQ2(G4double q2)   {primaryQ2 = q2;}
  G4double GetPrimaryQ2() const {return primaryQ2;}

  void     SetCrossSectionWeight(G4double csw)   {crossSectionWeight = csw;}
  G4double GetCrossSectionWeight() const {return crossSectionWeight;}

  void     SetPrimaryEventNumber(G4int en)      {primaryEventNumber = en;}
  G4int    GetPrimaryEventNumber() const {return primaryEventNumber;}

  G4double GetNumberOfPhotoelectrons(G4double eng);
  G4double GetNumberOfPhotoelectronsS20(G4double eng);
```

Figure 19.1: Header File

```

    void     SetLeftPMTHitValid(G4int state) {leftPMTHitValid = state;};
    G4int    GetLeftPMTHitValid() {return leftPMTHitValid; leftPMTHitValid = 0;};
    void     SetRightPMTHitValid(G4int state){rightPMTHitValid = state;};
    G4int    GetRightPMTHitValid() {return rightPMTHitValid; rightPMTHitValid = 0;};

    void     StoreStepVolumeName(G4String name) {StepVolumeName = name;};
    G4String GetStoredStepVolumeName() {return StepVolumeName;};

    void     SetEdgeEventDetected(G4int det){edgeEventDetected = det;};
    G4int    GetEdgeEventDetected(){return edgeEventDetected;};

    void     StoreLocalCerenkovExitPosition(G4ThreeVector ep) {cerenkovEventExitPos = ep;};
    G4ThreeVector GetLocalCerenkovExitPosition() {return cerenkovEventExitPos;};

    void     StoreCerenkovSecondaryParticleInfo(G4ThreeVector ep, G4ThreeVector ee, G4double eng, G4double charge);
    void     ResetCerenkovSecondaryParticleInfo();
    G4ThreeVector GetCerenkovSecondaryParticleOrigin(G4int indx);
    G4ThreeVector GetCerenkovSecondaryParticleMomentum(G4int indx);
    G4double GetCerenkovSecondaryParticleEnergy(G4int indx);
    G4double GetCerenkovSecondaryParticleCharge(G4int indx);
    G4int    GetCerenkovSecondaryParticleCount() {return  cerenkovSecondaryParticleCount;};
    G4int    GetCerenkovSecondaryElectronCount() {return  cerenkovSecondaryElectronCount;};
    G4int    GetCerenkovSecondaryPhotonCount() {return  cerenkovSecondaryPhotonCount;};
    G4int    GetCerenkovSecondaryPositronCount() {return  cerenkovSecondaryPositronCount;};

    void SetCurrentPMTHit(QweakSimCerenkovDetector_PMTHit* hit, G4int side){PMTHit = hit; PMTSide = side;};
    QweakSimCerenkovDetector_PMTHit *GetCurrentPMTHit(){return PMTHit;};
    G4int GetCurrentPMTSide() {return PMTSide;};

    void     AddCerenkovEnergyDeposit(G4double eng){CerEngDep += eng;};
    G4double GetCerenkovEnergyDeposit(G4bool zero = true){G4double tmp = CerEngDep; if(zero) CerEngDep = 0.0; return tmp;};


    void StoreCerenkovPhotonEnergy(G4double eng) {CerenkovPhotonEnergy.push_back(eng);};
    G4double GetCerenkovPhotonEnergyAtIndex(G4int ind) {return CerenkovPhotonEnergy[ind];};

};

#endif
```

Figure 19.2: Header File

```
#include "QweakSimUserInformation.hh"

QweakSimUserInformation::QweakSimUserInformation()
{
 cerenkovSecondaryParticleCount = 0;
 Initialize();
}

QweakSimUserInformation::~QweakSimUserInformation()
{
}

void QweakSimUserInformation::Print() const
{
}

void QweakSimUserInformation::Initialize()
{

  primaryQ2                  = 0.0;
  crossSectionWeight         = 0.0;
  primaryEventNumber         = 0;

  CerEngDep                  = 0.0;

  leftPMTHitValid            = 0;
  rightPMTHitValid           = 0;
  StoreStepVolumeName("none");
  SetEdgeEventDetected(0);

  if(cerenkovSecondaryParticleCount){
    delete[] cerenkovSecondaryPartOrig;
    delete[] cerenkovSecondaryPartMom;
    delete[] cerenkovSecondaryPartEng;
    delete[] cerenkovSecondaryPartCharge;
  }
  cerenkovSecondaryParticleCount = 0;
  cerenkovSecondaryElectronCount = 0;
  cerenkovSecondaryPhotonCount = 0;

  cerenkovOpticalPhotonCount = 0;
  CerenkovPhotonEnergy.clear();
  CerenkovPhotonEnergy.resize(0);

  cerenkovSecondaryPositronCount = 0;
  cerenkovSecondaryPartOrig     = NULL;
  cerenkovSecondaryPartMom      = NULL;
  cerenkovSecondaryPartEng      = NULL;
  cerenkovSecondaryPartCharge   = NULL;

  G4ThreeVector tmp(1000,1000,1000);
  cerenkovEventExitPos = tmp;

  G4double D753WKBS20_QE[65][2] = {
    {200.0*nanometer, 0.68},
    {210.0*nanometer, 3.55},
    {220.0*nanometer, 7.40},
    {230.0*nanometer, 10.4},
    {240.0*nanometer, 14.6},
    {250.0*nanometer, 17.8},
    {260.0*nanometer, 20.6},
    {270.0*nanometer, 22.6},
    {280.0*nanometer, 22.4},
    {290.0*nanometer, 21.8},
    {300.0*nanometer, 21.1},
    {310.0*nanometer, 20.5},
    {320.0*nanometer, 19.7},
```

Figure 19.3: Source File

```
  {330.0*nanometer, 19.2},
  {340.0*nanometer, 18.4},
  {350.0*nanometer, 18.0},
  {360.0*nanometer, 18.2},
  {370.0*nanometer, 18.8},
  {380.0*nanometer, 18.3},
  {390.0*nanometer, 17.6},
  {400.0*nanometer, 17.7},
  {410.0*nanometer, 17.6},
  {420.0*nanometer, 17.5},
  {430.0*nanometer, 17.1},
  {440.0*nanometer, 16.7},
  {450.0*nanometer, 15.8},
  {460.0*nanometer, 15.0},
  {470.0*nanometer, 14.4},
  {480.0*nanometer, 13.7},
  {490.0*nanometer, 13.1},
  {500.0*nanometer, 12.4},
  {510.0*nanometer, 11.7},
  {520.0*nanometer, 11.0},
  {530.0*nanometer, 10.4},
  {540.0*nanometer, 9.77},
  {550.0*nanometer, 9.15},
  {560.0*nanometer, 8.53},
  {570.0*nanometer, 7.95},
  {580.0*nanometer, 7.39},
  {590.0*nanometer, 6.87},
  {600.0*nanometer, 6.38},
  {610.0*nanometer, 5.90},
  {620.0*nanometer, 5.45},
  {630.0*nanometer, 5.07},
  {640.0*nanometer, 4.71},
  {650.0*nanometer, 4.39},
  {660.0*nanometer, 4.10},
  {670.0*nanometer, 3.79},
  {680.0*nanometer, 3.51},
  {690.0*nanometer, 3.25},
  {700.0*nanometer, 2.98},
  {710.0*nanometer, 2.68},
  {720.0*nanometer, 2.40},
  {730.0*nanometer, 2.13},
  {740.0*nanometer, 1.88},
  {750.0*nanometer, 1.65},
  {760.0*nanometer, 1.47},
  {770.0*nanometer, 1.30},
  {780.0*nanometer, 1.13},
  {790.0*nanometer, 0.96},
  {800.0*nanometer, 0.80},
  {810.0*nanometer, 0.65},
  {820.0*nanometer, 0.48},
  {830.0*nanometer, 0.33},
  {840.0*nanometer, 0.18}
};

G4double XP4572_QE[15][2] = {
  {200.0*nanometer ,  0.0} ,
  {250.0*nanometer ,  0.0} ,
  {280.0*nanometer ,  0.1} ,
  {290.0*nanometer ,  0.3} ,
  {300.0*nanometer ,  3.0} ,
  {315.0*nanometer , 10.0} ,
  {330.0*nanometer , 20.0} ,
  {350.0*nanometer , 27.5} ,
  {400.0*nanometer , 26.0} ,
  {480.0*nanometer , 20.0} ,
  {540.0*nanometer , 10.0} ,
  {590.0*nanometer ,  3.0} ,
```

Figure 19.4: Source File

```
    {615.0*nanometer ,  1.0} ,
    {640.0*nanometer ,  0.3} ,
    {660.0*nanometer ,  0.1}
  };

  PMTQE_XP4572      = new G4MaterialPropertyVector();
  PMTQED753WKBS20   = new G4MaterialPropertyVector();
  G4double E_value;

  for (G4int kk=0; kk<65 ; kk++)
    {
      if(kk < 15 ){
 E_value= 2*pi*hbarc/( XP4572_QE[kk][0] *nanometer);
 G4cout << "E_value " << kk << " = " << E_value << " QE = " << XP4572_QE[kk][1] << G4endl;
 PMTQE_XP4572->AddElement(E_value, XP4572_QE[kk][1]/100.);
      }

      E_value= 2*pi*hbarc/( D753WKBS20_QE[kk][0] *nanometer);
      PMTQED753WKBS20->AddElement(E_value, D753WKBS20_QE[kk][1]/100.);
    }
}

G4double QweakSimUserInformation::GetNumberOfPhotoelectrons(G4double eng)
{
  return PMTQE_XP4572->GetProperty(eng);
}

G4double QweakSimUserInformation::GetNumberOfPhotoelectronsS20(G4double eng)
{
  return PMTQED753WKBS20->GetProperty(eng);
}

void QweakSimUserInformation::StoreCerenkovSecondaryParticleInfo(G4ThreeVector ev,
                                                   G4ThreeVector em,
                                                   G4double eng,
                                                   G4double charge)
{
  G4int cnt = cerenkovSecondaryParticleCount;
  G4ThreeVector *tmp1 = NULL;
  G4ThreeVector *tmp2 = NULL;
  G4double      *tmp3 = NULL;
  G4double      *tmp4 = NULL;

  if(cnt){
    tmp1 = new G4ThreeVector[cnt];
    tmp2 = new G4ThreeVector[cnt];
    tmp3 = new G4double[cnt];
    tmp4 = new G4double[cnt];
  }

  for(G4int i = 0; i < cnt; i++){
    tmp1[i] = cerenkovSecondaryPartOrig[i];
    tmp2[i] = cerenkovSecondaryPartMom[i];
    tmp3[i] = cerenkovSecondaryPartEng[i];
    tmp4[i] = cerenkovSecondaryPartCharge[i];
  }

  if(cnt && cerenkovSecondaryPartOrig) delete[] cerenkovSecondaryPartOrig;
  if(cnt && cerenkovSecondaryPartMom) delete[] cerenkovSecondaryPartMom;
  if(cnt && cerenkovSecondaryPartEng) delete[] cerenkovSecondaryPartEng;
  if(cnt && cerenkovSecondaryPartCharge) delete[] cerenkovSecondaryPartCharge;

  cerenkovSecondaryPartOrig = new G4ThreeVector[cnt+1];
  cerenkovSecondaryPartMom = new G4ThreeVector[cnt+1];
  cerenkovSecondaryPartEng = new G4double[cnt+1];
  cerenkovSecondaryPartCharge = new G4double[cnt+1];

```

Figure 19.5: Source File

```
    for(G4int i = 0; i < cnt; i++) {
      cerenkovSecondaryPartOrig[i] = tmp1[i];
      cerenkovSecondaryPartMom[i] = tmp2[i];
      cerenkovSecondaryPartEng[i] = tmp3[i];
      cerenkovSecondaryPartCharge[i] = tmp4[i];
    }

    cerenkovSecondaryPartOrig[cnt] = ev;
    cerenkovSecondaryPartMom[cnt] = em;
    cerenkovSecondaryPartEng[cnt] = eng;
    cerenkovSecondaryPartCharge[cnt] = charge;

    if(cnt){
      delete[] tmp1;
      delete[] tmp2;
      delete[] tmp3;
      delete[] tmp4;
    }

    if(charge == -1)   cerenkovSecondaryElectronCount++;
    if(charge == 0)   cerenkovSecondaryPhotonCount++;
    if(charge == 1)   cerenkovSecondaryPositronCount++;
    cerenkovSecondaryParticleCount++;
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
  G4ThreeVector QweakSimUserInformation::GetCerenkovSecondaryParticleOrigin(G4int idx)
  {
    G4ThreeVector tmp(1000,1000,1000);
    if(!cerenkovSecondaryParticleCount) return tmp;
    if(idx < 0 || idx >= cerenkovSecondaryParticleCount) return tmp;
    return cerenkovSecondaryPartOrig[idx];
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
  G4ThreeVector QweakSimUserInformation::GetCerenkovSecondaryParticleMomentum(G4int idx)
  {
    G4ThreeVector tmp(1000,1000,1000);
    if(!cerenkovSecondaryParticleCount) return tmp;
    if(idx < 0 || idx >= cerenkovSecondaryParticleCount) return tmp;
    return cerenkovSecondaryPartMom[idx];
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
  G4double QweakSimUserInformation::GetCerenkovSecondaryParticleEnergy(G4int idx)
  {
    if(!cerenkovSecondaryParticleCount) return 0;
    if(idx < 0 || idx >= cerenkovSecondaryParticleCount) return 0;
    return cerenkovSecondaryPartEng[idx];
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
  G4double QweakSimUserInformation::GetCerenkovSecondaryParticleCharge(G4int idx)
  {
    if(!cerenkovSecondaryParticleCount) return 0;
    if(idx < 0 || idx >= cerenkovSecondaryParticleCount) return 0;
    return cerenkovSecondaryPartCharge[idx];
  }

  //....oooOO0OOooo........oooOO0OOooo........oooOO0OOooo........oooOO0OOooo....
  void QweakSimUserInformation::ResetCerenkovSecondaryParticleInfo()
  {
    if(cerenkovSecondaryParticleCount){
      delete[] cerenkovSecondaryPartOrig;
      delete[] cerenkovSecondaryPartMom;
      delete[] cerenkovSecondaryPartEng;
      delete[] cerenkovSecondaryPartCharge;
```

Figure 19.6: Source File

```
  }
  cerenkovOpticalPhotonCount = 0;
  CerenkovPhotonEnergy.clear();
  CerenkovPhotonEnergy.resize(0);

  cerenkovSecondaryParticleCount = 0;
  cerenkovSecondaryElectronCount = 0;
  cerenkovSecondaryPhotonCount = 0;
  cerenkovSecondaryPositronCount = 0;
  cerenkovSecondaryPartOrig      = NULL;
  cerenkovSecondaryPartMom       = NULL;
  cerenkovSecondaryPartEng       = NULL;
  cerenkovSecondaryPartCharge    = NULL;
}
```

Figure 19.7: Source File

# Bibliography


[1] S. Agostinelli *et al.* GEANT4: A Simulation Toolkit. *NIM A*, 506:250–303, 2003.

[2] R. Brun, F. Rademakers, S. Panacek, I. Antcheva and D. Buskulic. ROOT, an Object-Oriented Data Analysis Framework Users Guide. Technical report, CERN, FNAL, and Universit de Savoie (LAPP). `http://root.cern.ch`.

[3] . GEANT4: Physics Reference Manual. Technical report, CERN.

[4] D. Mack. Fused Silica Radiator Specifications for the Qweak Main Cerenkov Detectors . Technical report, JLab. `http://qweak.jlab.org/doc-private/ShowDocument?docid=228`.

[5] N. Simicevic. Design of a Cerenkov Counter for the Qweak Experiment. Technical report, Louisiana Tech. `http://qweak.jlab.org/DocDB/0000/000028/001/28.pdf`.

[6] BaBar Collaboration. The BaBar DIRC bar . Technical report. .

[7] Electron Tubes. Obtained from PMT manufacturer.

[8] Larry Lee. Private communication (unpublished TRIUMF and Russian Measurements).

[9] GEANT4 Collaboration. Users Guide For Application Developers (`http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/ch05.html`). Technical report, CERN, 2007.

[10] A. Levin and C. Moisan. A more physical approach to model the surface treatment of scintillation counters and its implementation into detect. *IEEE Nuclear Science Symposium*, 1996.

[11] Jackson. *Electromagnetism.* ?, ?

[12] E.Hecht. *Optics.* Addison Wesley, 1998.