or `G4Radioactivation( "Radioactivation", 1.0e+60*CLHEP::year )`
(this is for custom physics lists, before run initialization).

In the examples, we have followed the first method.

5.2.6 Gamma-nuclear and Lepto-nuclear Processes

Gamma-nuclear and lepto-nuclear reactions are handled in GEANT4 as hybrid processes which typically require both electromagnetic and hadronic models for their implementation. While neutrino-induced reactions are not currently provided, the GEANT4 hadronic framework is general enough to include their future implementation as a hybrid of weak and hadronic models.

The general scheme followed is to factor the full interaction into an electromagnetic (or weak) vertex, in which a virtual particle is generated, and a hadronic vertex in which the virtual particle interacts with a target nucleus. In most cases the hadronic vertex is implemented by an existing GEANT4 model which handles the intra-nuclear propagation.

The cross sections for these processes are parameterizations, either directly of data or of theoretical distributions determined from the integration of lepton-nucleon cross sections double differential in energy loss and momentum transfer.

Electro-nuclear reactions in GEANT4 are handled by the classes `G4ElectronNuclearProcess` and `G4PositronNuclearProcess`, which are both implmented by `G4ElectroVDNuclearModel`. This model consists of three sub-models: code which generates the virtual photon from the lepton-nucleus vertex, the Bertini-style cascade to handle the low and medium energy photons, and the FTFP model to handle the high energy photons.

Muon-nuclear reactions are handled similarly. The process `G4MuonNuclearProcess` can be assigned the `G4MuonVDNuclearModel` which in turn is implemented by three sub-models: virtual gamma generation code, Bertini-style cascade and the FTFP model.

Resonance effects

Note that the model that generates secondary particles does not take resonance effects into account. This may be particularly important in the region of the giant dipole resonance, at ~20MeV for high-Z materials (an important region for shielding in some medical applications.) Here, about 2/3 of photonuclear interactions may not produce neutrons.

In order to obtain correct photonuclear production in this regime, the user should use a LEND, data-driven model. Alternatively, if the environment variable `G4CASCADE_CHECK_PHOTONUCLEAR` is set, the model will ensure that, for incident energies less than 50 MeV, the mass of the nucleus changes during the interaction.

5.2.7 Optical Photon Processes

A photon is considered to be *optical* when its wavelength is much greater than the typical atomic spacing. In GEANT4 optical photons are treated as a class of particle distinct from their *gamma* cousins. Optical photons have different processes than gamma particles. An important use case for optical photons is that they interact with boundaries between volumes, undergoing reflection, refraction, etc.

Note: There is no transition between the optical photon and gamma particle classes: a *gamma* will never become an *optical photon*.

Optical photons are generated in GEANT4 by

- Cerenkov effect (class `G4Cerenkov`)
- Scintillation (class `G4Scintillation`)

The source code for these classes is in the `source/processes/electromagnetic/xrays` directory.

Optical photons interact through the processes

- Absorption (class `G4OpAbsorption`)
- Rayleigh scattering (class `G4OpRayleigh`)
- Mie scattering (class `G4OpMieHG`)
- Wave-length shifting (classes `G4OpWLS` and `G4OpWLS2`)
- Boundary scattering (class `G4OpBoundary`)

The source code for these classes is in the `source/processes/optical` directory.

Optical photons are generated in GEANT4 without regard to energy conservation and their energy must therefore not be tallied as part of the energy balance of an event.

The are several steps to simulate optical photons in GEANT4.

- The optical photon and optical processes must be defined and configured
- Optical properties need to be assigned to relevant materials and surfaces
- If an optical photon is a primary particle, its polarization should be set

Defining optical processes: `G4OpticalPhysics` constructor

The most straightforward way of using optical physics is to use the `G4OpticalPhysics` constructor in main(), as in the extended optical examples. This automatically includes all the optical physics processes and provides a default configuration. The configuration is stored in the class `G4OpticalParameters`, and can be accessed via the class `G4OpticalParametersMessenger`. Both macro commands and C++ methods are available.

An example of using `G4OpticalPhysics` is shown in Listing 5.3.

Listing 5.3: An example of using the `G4OpticalPhysics` constructor in main().

```
#include "G4OpticalPhysics.hh"

...

G4VModularPhysicsList* physicsList = new FTFP_BERT;  // for example
G4OpticalPhysics* opticalPhysics = new G4OpticalPhysics();

physicsList->RegisterPhysics(opticalPhysics);
runManager->SetUserInitialization(physicsList);

// to set parameters in code, if wanted
auto opticalParams = G4OpticalParameters::Instance();
opticalParams->SetWLSTimeProfile("delta");
```

See the sections for each process for details on process-specific commands in the class `G4OpticalParametersMessenger`. There are two general commands:

1. `/process/optical/verbose int` sets the verbosity of all processes to the given value. 0 is silent, 1 is printing during initialization only, 2 is printing during tracking
2. `/process/optical/processActivation name bool` is used to deactivate individual processes. `name` may be one of Cerenkov, Scintillation, OpAbsorption, OpRayleigh, OpMieHG, OpBoundary, OpWLS, OWLS2. By default, all the processes are activated.

The optical parameters can be printed by invoking `G4OpticalParameters::Instance()->Dump()`.

Note: In version 10.7, redundant commands were marked as deprecated, but were still available. In version 11, these

commands have been removed.

Setting the polarization

For the simulation of optical photons to work correctly in GEANT4, they must have a linear polarization. This is unlike most other particles in GEANT4 but is automatically and correctly done for optical photons that are generated as secondaries by existing processes in GEANT4. If the user wishes to start optical photons as primary particles, they must set the linear polarization using particle gun methods, the General Particle Source, or their PrimaryGeneratorAction. For an unpolarized source, the linear polarization should be sampled randomly for each new primary photon. See the extended optical examples for methods to set the polarization.

Defining material properties

The optical properties of the medium which are key to the implementation of these types of processes are stored as entries in a `G4MaterialPropertiesTable` which is a private data member of the `G4Material` class.

Each entry in the `G4MaterialPropertiesTable` consists of a *key* and *value* pair. The key is used to retrieve the corresponding value. Keys are defined in two `enums` (and thus are `G4ints`). Keys are also available as `G4Strings`. Properties are added using the `G4String` key, but may be accessed by either the `G4String` key or the `G4int` key. Material property names are listed in tables for each process, below.

These properties may be independent of photon energy (denoted "Constant" or "Const") or they may be expressed as a function of the photon's energy. In the case of Constant parameters, the value is a `G4double`. Methods to access the Constant parameters have "Const" in their names.

In the case of energy-dependent properties, the value is a `G4MaterialPropertyVector` (which is a typedef to `G4PhysicsFreeVector`). Energy dependent properties may be added by specifying a `G4MaterialPropertyVector`, or by passing two std::vectors of type `G4double`. The first vector is the energy and the second vector is the property value at that energy. The numbers of elements in the two vectors must be the same (if std::vectors are used, GEANT4 will check that the numbers of elements in the two vectors are the same).

For backwards compatibility, energy dependent properties may also be created by specifying two C arrays of doubles. In this case, it is up to the user to ensure that both arrays have the same number of values, and the number of values must be passed as an argument.

Typically, a user only needs to add properties. Authors of processes that need these properties will need to access the values. Methods to add properties are:

- `void AddConstProperty(const G4String& key, G4double PropertyValue)`
- `void AddConstProperty(const char* key, G4double PropertyValue)`
- `G4MaterialPropertyVector* AddProperty(const G4String& key, const std::vector<G4double>& photonEnergies, const std::vector<G4double>& propertyValues, G4bool createNewKey = false, G4bool spline = false)`
- `G4MaterialPropertyVector* AddProperty(const char* key, G4double* PhotonEnergies, G4double* PropertyValues, G4int NumEntries, G4bool createNewKey = false, G4bool spline = false)`
- `void AddProperty(const G4String& key, G4MaterialPropertyVector* opv, G4bool createNewKey = false)`
- `void AddProperty(const char* key, G4MaterialPropertyVector* opv, G4bool createNewKey = false)`
- `void AddProperty(const G4String& key, const G4String& mat)`

The createNewKey argument, if `false`, will check that the key string is one of the default keys. See *User-defined properties* for more information. The spline argument enables spline interpolation of the data.

An example of adding material properties to a material is shown in Listing 5.4. In this example the interpolation of the `G4MaterialPropertyVector` is to be done by a spline fit. The default is a linear interpolation.

Listing 5.4: Example of optical properties added to a `G4MaterialPropertiesTable` and linked to a `G4Material`

```
G4Material* scintillator = new G4Material(/*...*/);

std::vector<G4double> energy     = {2.034*eV, 3.*eV, 4.136*eV};
std::vector<G4double> rindex     = {1.3435, 1.351, 1.3608};
std::vector<G4double> absorption = {344.8*cm, 850.*cm, 1450.0*cm};

G4MaterialPropertiesTable* MPT = new G4MaterialPropertiesTable();

// property independent of energy
MPT->AddConstProperty("SCINTILLATIONYIELD", 100./MeV);

// properties that depend on energy
MPT->AddProperty("RINDEX", energy, rindex);
MPT->AddProperty("ABSLENGTH", energy, absorption);

scintillator->SetMaterialPropertiesTable(MPT);
```

Note: Starting in version 11.0, GEANT4 will check that the property name is in the list of pre-defined properties. This avoids errors due to spelling mistakes. If you want to define a new property name, see *User-defined properties*.

Note: Digitized data of refractive indices for many materials can be accessed for instance at https://refractiveindex.info.

Pre-defined properties

Starting in version 11.0, some optical material properties are defined in GEANT4. Currently, these are the refractive indices for "Air", "Water", "PMMA", and "Fused Silica". The precise values are found in `source/materials/include/G4OpticalMaterialProperties.hh`.

To use them, add them to the material properties table as follows:

```
G4MaterialPropertiesTable* MT = new G4MaterialPropertiesTable();
MT->AddProperty("RINDEX", "Fused Silica");
```

User-defined properties

One may create their own properties, for example, for use in custom processes. To facilitate this, the various `AddProperty()`/`AddConstProperty()` methods have a default argument `createNewKey`. Set this to `true` to allow a new key name.

```
myMPT->AddConstProperty("USERDEFINEDCONST", 3.14, true);
```

This value may accessed by the string name:

```
G4double val = myMPT->GetConstProperty("USERDEFINEDCONST");
```

but there is a potential speed-up. Material properties are stored internally in a vector. To access a material property, first find the index of the property (e.g. at initialization), then use the index during the event loop.

```
G4int index = myMPT->GetConstPropertyIndex("USERDEFINEDCONST");
...
G4double val = myMPT->GetConstProperty(index);
```

It is possible to test if a property has been defined. In the case of constant properties, the methods `ConstPropertyExists(const G4String& key)`, `ConstPropertyExists(const char* key)`, `ConstPropertyExists(const G4int index)` return `true` if the property is defined, false otherwise. For energy-dependent properties, the `GetProperty(...)` methods return `nullptr` if the property has not been defined.

Cerenkov Effect

The radiation of Cerenkov light occurs when a charged particle moves through a dispersive medium faster than the group velocity of light in that medium. Photons are emitted on the surface of a cone, whose opening angle with respect to the particle's instantaneous direction decreases as the particle slows down. At the same time, the frequency of the photons emitted increases, and the number produced decreases. When the particle velocity drops below the local speed of light, the radiation ceases and the emission cone angle collapses to zero. The photons produced by this process have an inherent polarization perpendicular to the cone's surface at production.

To generate Cerenkov optical photons in a material, refractive index must be specified using the material property name `RINDEX`.

The flux, spectrum, polarization and emission of Cerenkov radiation in the `AlongStepDoIt` method of the class `G4Cerenkov` follow well-known formulae, with two inherent computational limitations. The first arises from stepwise simulation, and the second comes from the requirement that numerical integration calculate the average number of Cerenkov photons per step. The process makes use of a `G4PhysicsTable` which contains incremental integrals to expedite this calculation.

The time and position of Cerenkov photon emission are calculated from quantities known at the beginning of a charged particle's step. The step is assumed to be rectilinear even in the presence of a magnetic field. The user may limit the step size by specifying a maximum (average) number of Cerenkov photons created during the step, using the `setMaxPhotons` command. The actual number generated will necessarily be different due to the Poissonian nature of the production. In the present implementation, the production density of photons is distributed evenly along the particle's track segment, even if the particle has slowed significantly during the step. The step can also be limited with the `setMaxBetaChange` command, where the argument is the allowed change in percent.

The large number of optical photons that can be produced (about 300/cm in water) can fill the available memory. GEANT4 by default will track the Cerenkov photons produced in a step before continuing to track the primary particle. This may be changed using the `setTrackSecondariesFirst` command.

Configuration

Material property names used in the process are given in the following table.

Table 5.2: Material properties for the Cerenkov process.

Name	Type	Description	Unit Category
RINDEX	Energy-dependent	Refractive index	Unitless

These parameters are available to configure the process.

- Set the step size to limit the number of photons produced (on average) to a given value (an integer N)
  - macro command: /process/optical/cerenkov/setMaxPhotons N
  - C++ statement: G4OpticalParameters::Instance()->SetMaxNumPhotonsPerStep(G4int);
  - default value: 100

- Set the maximum change in $\beta = v/c$ in a step, expressed in percent.
  - macro command: /process/optical/cerenkov/setMaxBetaChange X.X
  - C++ statement: G4OpticalParameters::Instance()->SetMaxBetaChangePerStep(G4double);
  - default value: 10.0
- Specify whether to add Cerenkov photons to the stack, and track them.
  - macro command: /process/optical/cerenkov/setStackPhotons true
  - C++ statement: G4OpticalParameters::Instance()->SetCerenkovStackPhotons(G4bool);
  - default value: true
- Specify whether to track secondaries produced in the step before continuing with primary.
  - macro command: /process/optical/cerenkov/setTrackSecondariesFirst true
  - C++ statement: G4OpticalParameters::Instance()->SetCerenkovTrackSecondariesFirst(G4bool);
  - default value: true
- Set the verbosity of the process. 0 = silent; 1 = initialisation; 2 = during tracking
  - macro command: /process/optical/cerenkov/verbose
  - C++ statement: G4OpticalParameters::Instance()->SetCerenkovVerbosity(G4int);
  - default value: 1

Scintillation

A scintillating material is characterised by the number of optical photons produced (the yield), their spectrum, and the distribution of emission times. The yield may be dependent on the type of primary particle. The distribution of emission times can be characterised by an exponential rise time, and a decay with one or more time constants. Starting with GEANT4 version 10.7, it is possible to specify up to three decay time constants, for particle-independent and particle-dependent yields. This new method uses different material property names than used in previous versions. The method used in previous versions is no longer available in GEANT4 version 11.0. All of the capabilities of the previous method are available with the new method.

The scintillation yield may depend on the type of primary particle. In this case, the material parameter names are modified from the case where the yield is independent of primary particle.

Each photon's frequency is sampled from the empirical spectrum. The photons originate evenly along the track segment and are emitted uniformly into the 4π solid angle with a random linear polarization perpendicular to their momentum direction, and an emission time characteristic for the scintillation component.

Important: Starting in GEANT4 version 11.0, only the new method (called "enhanced" in version 10.7) is available to specify decay time constants. It is no longer necessary to set the optical parameter `setEnhancedTimeConstants` to true.

Scintillation independent of particle type

If the scintillation yield is independent of particle type, the yield is specified using the material constant property `SCINTILLATIONYIELD`. The mean number of optical photons in a step is calculated as the `SCINTILLATIONYIELD` property, times the yield factor, times either the `VisibleEnergyDepositAtAStep` (if Birks' saturation is used) or the total energy deposit (otherwise). (Note the yield factor is redundant here and should not be used. See the description of the old scintillation methods for its use.) The property `SCINTILLATIONYIELD` is expressed in number per energy.

If the calculated mean number of optical photons for the step is less than or equal to 10, the actual number is determined from a Poisson distribution with that mean, then converted to an integer. If the mean number of photons is greater than 10, the number of photons is chosen from a Gaussian distribution with that mean, and a sigma equal to the square root of the the mean times a factor called the resolution scale. This factor is set using the material constant property `RESOLUTIONSCALE`. A resolution scale of zero produces no fluctuation.

There may be 1 to 3 decay component with independent spectra and rise and decay time constants.

If there is one component, specify the decay time constant with the material constant property SCINTILLATIONTIMECONSTANT1. If a non-zero rise time is wanted, set the optical parameter setFiniteRiseTime to true, and set the material constant property SCINTILLATIONRISETIME1 to the desired value. The creation time of the photon is chosen from a distribution with these characteristics.

The energy spectrum of the emitted photons is specified using the energy-dependent material property SCINTILLATIONCOMPONENT1.

If there are 2 or 3 decay component, the time constant, rise time, and spectrum must be specified for the additional component. The property names are the same as for the first component, with the "1" at the end of the property name replaced with "2" or "3". Additionally, the fraction of photons in each component must be specified. Set the material constant properties SCINTILLATIONYIELD1, SCINTILLATIONYIELD2, and SCINTILLATIONYIELD3 (if there are three components) to the relative amount of photons produced in each component. The values will be automatically normalized. The spectra are specified using the material properties SCINTILLATIONCOMPONENT2 and SCINTILLATIONCOMPONENT3. The values of the time constants do not need to be ordered (i.e., SCINTILLATIONTIMECONSTANT2 can be greater or less than SCINTILLATIONTIMECONSTANT1).

Scintillation dependent on particle type

The scintillation yield, and the strength of each component, may depend on particle type. If so, set the optical parameter setByParticleType true, and specify a yield vector for each particle type using material property names such as ALPHASCINTILLATIONYIELD. The mean number of photons produced in the step is calculated as the difference in the value of the yield vector at the pre-step kinetic energy of the primary particle, and the value of the yield vector at the pre-step kinetic energy minus the energy deposit in the step:

```
ScintillationYield = yieldVector->Value(PreStepKineticEnergy)
 - yieldVector->Value(PreStepKineticEnergy - StepEnergyDeposit);
```

The relative amounts of photons produced in each component are specified using the material constant properties with key names such as ALPHASCINTILLATIONYIELD1, etc. If there is only one component for a primary particle, it is not necessary to specify the per-component yield e.g. ALPHASCINTILLATIONYIELD1.

Note: Starting in GEANT4 version 11.2, it is possible to specify different decay time constants for different particles. Existing code will continue to work without change.

The decay time constants may either be the same for all particles, or specified for particular particles. In order for the time constants to be different for different particles, material constant properties such as PROTONSCINTILLATIONTIMECONSTANT1 need to be specified. For any given particle, if this property is not specified, the value specified by SCINTILLATIONTIMECONSTANT1 is used (and similarly for channels and 3). In this way, existing code will run unchanged.

The rise time constants and the emission spectra are not dependent on particle type. These are specified in the same way as for the scintillation yield independent of particle type. RESOLUTIONSCALE also is equivalent.

Listing 5.5: Specification of scintillation properties in DetectorConstruction using enhanced time constants (from extended example LXe).

```
//Liquid Xenon
fLXe = new G4Material("LXe",z=54.,a=131.29*g/mole,density=3.020*g/cm3);

std::vector<G4double> lxe_Energy = {7.0*eV, 7.07*eV, 7.14*eV};
```

(continues on next page)

(continued from previous page)

```
std::vector<G4double> lxe_SCINT = {0.1, 1.0, 0.1};
std::vector<G4double> lxe_RIND  = {1.59, 1.57, 1.54};
std::vector<G4double> lxe_ABSL  = {35.*cm, 35.*cm, 35.*cm};
fLXe_mt = new G4MaterialPropertiesTable();
fLXe_mt->AddProperty("SCINTILLATIONCOMPONENT1", lxe_Energy, lxe_SCINT);
fLXe_mt->AddProperty("SCINTILLATIONCOMPONENT2", lxe_Energy, lxe_SCINT);
fLXe_mt->AddProperty("RINDEX",        lxe_Energy, lxe_RIND);
fLXe_mt->AddProperty("ABSLENGTH",     lxe_Energy, lxe_ABSL);
fLXe_mt->AddConstProperty("SCINTILLATIONYIELD", 12000./MeV);
fLXe_mt->AddConstProperty("RESOLUTIONSCALE", 1.0);
fLXe_mt->AddConstProperty("SCINTILLATIONTIMECONSTANT1", 20.*ns);
fLXe_mt->AddConstProperty("SCINTILLATIONTIMECONSTANT2", 45.*ns);
fLXe_mt->AddConstProperty("SCINTILLATIONYIELD1", 1.0);
fLXe_mt->AddConstProperty("SCINTILLATIONYIELD2", 0.0);
fLXe->SetMaterialPropertiesTable(fLXe_mt);

// Set the Birks Constant for the LXe scintillator
fLXe->GetIonisation()->SetBirksConstant(0.126*mm/MeV);
```

Configuration

These parameters are available to configure the scintillation process.

- Enable particle-dependent yields
  - macro command: /process/optical/scintillation/setByParticleType
  - C++ statement: G4OpticalParameters::Instance()->SetScintByParticleType(G4bool val)
  - default value: false
- Record track information
  - macro command: /process/optical/scintillation/setTrackInfo
  - C++ statement: G4OpticalParameters::Instance()->SetScintTrackInfo(G4bool val)
  - default value: false
- Whether to track secondaries before resuming tracking of primary particle
  - macro command: /process/optical/scintillation/setTrackSecondariesFirst
  - C++ statement: G4OpticalParameters::Instance()->SetScintTrackSecondariesFirst(G4bool val)
  - default value: true
- Whether to use a finite rise time for the secondary emission time
  - macro command: /process/optical/scintillation/setFiniteRiseTime
  - C++ statement: G4OpticalParameters::Instance()->SetScintFiniteRiseTime(G4bool val)
  - default value: false
- Whether to add produced optical photons to the stack (the alternative is to kill them)
  - macro command: /process/optical/scintillation/setStackPhotons
  - C++ statement: G4OpticalParameters::Instance()->SetScintStackPhotons(G4bool val)
  - default value: true
- Set the verbosity level. 0 = silent, 1 = initialisation, 2 = during tracking
  - macro command: /process/optical/scintillation/verbose
  - C++ statement: G4OpticalParameters::Instance()->SetScintVerboseLevel(G4int val)
  - default value: 1

Table 5.3: Material properties for the optical scintillation process.

Name	Type	Description	Unit category
ALPHASCINTILLATIONYIELD	Energy-dependent	Yield vector for alphas	1/Energy
ALPHASCINTILLATIONYIELD1	Constant	Relative yield of component 1 for alphas	Unitless

continues on next page

Table 5.3 – continued from previous page

Name	Type	Description	Unit category
ALPHASCINTILLATIONYIELD2	Constant	Relative yield of component 2 for alphas	Unitless
ALPHASCINTILLATIONYIELD3	Constant	Relative yield of component 3 for alphas	Unitless
DEUTERONSCINTILLATIONYIELD	Energy-dependent	Yield vector for deuterons	1/Energy
DEUTERONSCINTILLATIONYIELD1	Constant	Relative yield of component 1 for deuterons	Unitless
DEUTERONSCINTILLATIONYIELD2	Constant	Relative yield of component 2 for deuterons	Unitless
DEUTERONSCINTILLATIONYIELD3	Constant	Relative yield of component 3 for deuterons	Unitless
ELECTRONSCINTILLATIONYIELD	Energy-dependent	Yield vector for electrons	1/Energy
ELECTRONSCINTILLATIONYIELD1	Constant	Relative yield of component 1 for electrons	Unitless
ELECTRONSCINTILLATIONYIELD2	Constant	Relative yield of component 2 for electrons	Unitless
ELECTRONSCINTILLATIONYIELD3	Constant	Relative yield of component 3 for electrons	Unitless
IONSCINTILLATIONYIELD	Energy-dependent	Yield vector for ions	1/Energy
IONSCINTILLATIONYIELD1	Constant	Relative yield of component 1 for ions	Unitless
IONSCINTILLATIONYIELD2	Constant	Relative yield of component 2 for ions	Unitless
IONSCINTILLATIONYIELD3	Constant	Relative yield of component 3 for ions	Unitless
PROTONSCINTILLATIONYIELD	Energy-dependent	Yield vector for protons	1/Energy
PROTONSCINTILLATIONYIELD1	Constant	Relative yield of component 1 for protons	Unitless
PROTONSCINTILLATIONYIELD2	Constant	Relative yield of component 2 for protons	Unitless
PROTONSCINTILLATIONYIELD3	Constant	Relative yield of component 3 for protons	Unitless
RESOLUTIONSCALE	Constant	Factor to vary width of yield distribution	Unitless
SCINTILLATIONCOMPONENT1	Energy-dependent	Energy spectrum for decay component 1	Unitless
SCINTILLATIONCOMPONENT2	Energy-dependent	Energy spectrum for decay component 2	Unitless
SCINTILLATIONCOMPONENT3	Energy-dependent	Energy spectrum for decay component 3	Unitless
SCINTILLATIONRISETIME1	Constant	Rise time for component 1	Time
SCINTILLATIONRISETIME2	Constant	Rise time for component 2	Time
SCINTILLATIONRISETIME3	Constant	Rise time for component 3	Time
SCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1	Time

continues on next page

Table 5.3 – continued from previous page

Name	Type	Description	Unit category
SCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2	Time
SCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3	Time
SCINTILLATIONYIELD	Constant	Mean yield (number of particle produce per energy)	1/Energy
SCINTILLATIONYIELD1	Constant	Relative yield of component 1	Unitless
SCINTILLATIONYIELD2	Constant	Relative yield of component 2	Unitless
SCINTILLATIONYIELD3	Constant	Relative yield of component 3	Unitless
TRITONSCINTILLATIONYIELD	Energy-dependent	Yield vector for tritons	1/Energy
TRITONSCINTILLATIONYIELD1	Constant	Relative yield of component 1 for tritons	Unitless
TRITONSCINTILLATIONYIELD2	Constant	Relative yield of component 2 for tritons	Unitless
TRITONSCINTILLATIONYIELD3	Constant	Relative yield of component 3 for tritons	Unitless
PROTONSCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1 for protons	Time
PROTONSCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2 for protons	Time
PROTONSCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3 for protons	Time
DEUTERONSCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1 for deuterons	Time
DEUTERONSCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2 for deuterons	Time
DEUTERONSCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3 for deuterons	Time
TRITONSCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1 for tritons	Time
TRITONSCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2 for tritons	Time
TRITONSCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3 for tritons	Time
ALPHASCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1 for alphas	Time
ALPHASCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2 for alphas	Time
ALPHASCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3 for alphas	Time
IONSCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1 for ions	Time
IONSCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2 for ions	Time
IONSCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3 for ions	Time

continues on next page

Table 5.3 – continued from previous page

Name	Type	Description	Unit category
ELECTRONSCINTILLATIONTIMECONSTANT1	Constant	Time constant for component 1 for electrons	Time
ELECTRONSCINTILLATIONTIMECONSTANT2	Constant	Time constant for component 2 for electrons	Time
ELECTRONSCINTILLATIONTIMECONSTANT3	Constant	Time constant for component 3 for electrons	Time

Absorption

The implementation of optical photon bulk absorption, `G4OpAbsorption`, is trivial in that the process merely kills the particle. The procedure requires the user to fill the relevant `G4MaterialPropertiesTable` with empirical data for the absorption length, using `ABSLENGTH` as the property key. The absorption length is the average distance traveled by a photon before being absorbed by the medium; i.e., it is the mean free path returned by the `GetMeanFreePath` method.

Table 5.4: Material properties for the optical absorption process.

Name	Type	Description	Unit category
ABSLENGTH	Energy-dependent	Absorption length	Length

This `G4OpticalParameters` command can be used to configure the process.

- Set the verbosity of the absorption process. 0 = silent; 1 = initialisation; 2 = during tracking.
  - macro command: /process/optical/absorption/verbose 1
  - C++ statement: G4OpticalParameters::Instance()->SetAbsorptionVerboseLevel(G4int verboseLevel);
  - default value: 1

Rayleigh Scattering

The differential cross section in Rayleigh scattering, $d\sigma/d\Omega$, is proportional to $1 + \cos^2\theta$, where θ is the angle of the new polarization vector with respect to the old polarization vector. The `G4OpRayleigh` scattering process samples this angle accordingly and then calculates the scattered photon's new direction by requiring that it be perpendicular to the photon's new polarization in such a way that the final direction, initial and final polarizations are all in one plane. This process thus depends on the particle's polarization (spin). A photon which does not have a polarization will not be Rayleigh scattered.

The process requires Rayleigh scattering attenuation length data. The Rayleigh scattering attenuation length is the average distance traveled by a photon before it is Rayleigh scattered in the medium and it is the distance returned by the `GetMeanFreePath` method. The attenuation length may be provided by the user filling a `G4MaterialPropertiesTable` with key `RAYLEIGH`.

The `G4OpRayleigh` class provides a method which can be used to calculate the attenuation coefficient of a medium following the Einstein-Smoluchowski formula. The derivation of this formula requires the use of statistical mechanics, includes temperature, and depends on the isothermal compressibility of the medium. This function is convenient when the Rayleigh attenuation length is not known from measurement but may be calculated from first principles using the above material constants. If the material property `RAYLEIGH` is not set, the attenuation coefficient will be calculated if either the material name is "Water", or the material's isothermal compressibility is provided using the material property `ISOTHERMAL_COMPRESSIBILITY`. For the material "Water", a temperature of 10ºC and isothermal compressibility of 7.658 x 10^{-23} m^3/MeV is used. For other materials, the temperature is determined by calling them material's `GetTemperature()`. The calculated attenuation length may be scaled by provided a material property `RS_SCALE_FACTOR`.

Table 5.5: Material properties for the optical Rayleigh scattering process.

Name	Type	Description	Unit category
ISOTHERMAL_COMPRESSIBILITY	Constant	Isothermal compressibility. Can be used to calculate mean free path	Volume/Energy
RAYLEIGH	Energy-dependent	Attenuation length	Length
RS_SCALE_FACTOR	Constant	If set, multiply the calculated mean free path by this factor	Unitless

This `G4OpticalParameters` command can be used to configure the process.

- Set the verbosity of the Rayleigh scattering process. 0 = silent; 1 = initialisation; 2 = during tracking.
  - macro command: /process/optical/rayleigh/verbose 1
  - C++ statement: G4OpticalParameters::Instance()->SetRayleighVerboseLevel(G4int verboseLevel);
  - default value: 1

Wavelength Shifting

Wavelength shifting (WLS) fibers are used in many high-energy particle physics experiments. They absorb light at one wavelength and re-emit light at a different wavelength and are used for several reasons. For one, they tend to decrease the self-absorption of the detector so that as much light reaches the PMTs as possible. WLS fibers are also used to match the emission spectrum of the detector with the input spectrum of the PMT.

A WLS material is characterized by its photon absorption and photon emission spectrum and by a possible time delay between the absorption and re-emission of the photon. Wavelength Shifting may be simulated by specifying these empirical parameters for each WLS material in the simulation. It is sufficient to specify a relative spectral distribution as a function of photon energy for the WLS material. `WLSABSLENGTH` is the absorption length of the material as a function of the photon's energy. `WLSCOMPONENT` is the relative emission spectrum of the material as a function of the photon's energy, and `WLSTIMECONSTANT` accounts for any time delay which may occur between absorption and re-emission of the photon. An example is shown in the Listing 5.6.

Listing 5.6: Specification of WLS properties in `DetectorConstruction`.

```
std::vector<G4double> PhotonEnergy = { 6.6*eV, 6.7*eV, 6.8*eV, 6.9*eV,
                                       7.0*eV, 7.1*eV, 7.2*eV, 7.3*eV, 7.4*eV};

std::vector<G4double> RIndexFiber =
          {1.60, 1.60, 1.60, 1.60, 1.60, 1.60, 1.60, 1.60, 1.60};
std::vector<G4double> AbsFiber =
          {0.1*mm, 0.2*mm, 0.3*mm, 0.4*cm, 1.0*cm, 10.0*cm, 1.0*m, 10.0*m, 10.0*m};
std::vector<G4double> EmissionFiber =
          {0.0, 0.0, 0.0, 0.1, 0.5, 1.0, 5.0, 10.0, 10.0};

G4Material* WLSFiber = new G4Material(/*...*/);
G4MaterialPropertiesTable* MPTFiber = new G4MaterialPropertiesTable();

MPTFiber->AddProperty("RINDEX", PhotonEnergy, RIndexFiber);
MPTFiber->AddProperty("WLSABSLENGTH", PhotonEnergy, AbsFiber);
MPTFiber->AddProperty("WLSCOMPONENT", PhotonEnergy, EmissionFiber);
MPTFiber->AddConstProperty("WLSTIMECONSTANT", 0.5*ns);

WLSFiber->SetMaterialPropertiesTable(MPTFiber);
```

The way the `WLSTIMECONSTANT` is used depends on the time profile method chosen by the user. If the "exponential" option is set, the time delay between absorption and re-emission of the photon is sampled from an exponential

distribution, with the decay term equal to `WLSTIMECONSTANT`. If, on the other hand, the "delta" option is chosen, the time delay is a delta function and equal to `WLSTIMECONSTANT`. The default is "delta".

The number of secondaries emitted may be configured. By default, each WLS interaction generates one secondary. If the material property `WLSMEANNUMBERPHOTONS` is set, the number of secondaries is chosen from a Poisson distribution with mean equal to `WLSMEANNUMBERPHOTONS`.

Important: New in GEANT4 version 10.7: definition of two WLS processes.

It is possible to have a material with two WLS processes. The `G4OpticalPhysics` constructor builds two processes by default, named `OpWLS` and `OpWLS2`. Usage of the second process is analogous to the first. The material property names for the `OpWLS2` process are the same as for `OpWLS`, with a "2" appended. The time profile may be chosen to be "exponential" or "delta" using macro or C++ commands. In order to activate the `OpWLS2` process in a material, the user needs to define the material properties.

Material properties are shown in Table 5.6.

Table 5.6: Material properties for WLS processes.

Name	Type	Description	Unit category
WLSABSLENGTH	Energy-dependent	Absorption length of first WLS process	Length
WLSABSLENGTH2	Energy-dependent	Absorption length of second WLS process	Length
WLSCOMPONENT	Energy-dependent	Emission spectrum of first WLS process	Unitless
WLSCOMPONENT2	Energy-dependent	Emission spectrum of second WLS process	Unitless
WLSMEANNUMBER-PHOTONS	Constant	Mean number of photons emitted per interaction, for first WLS process	Unitless
WLSMEANNUMBER-PHOTONS2	Constant	Mean number of photons emitted per interaction, for second WLS process	Unitless
WLSTIMECONSTANT	Constant	Time constant for emission, for first WLS process	Time
WLSTIMECONSTANT2	Constant	Time constant for emission, for second WLS process	Time

These parameters are available to configure the process:

- Set the time profile of the first WLS process to be either "delta" or "exponential":
  - macro command: /process/optical/wls/setTimeProfile value
  - C++ statement: G4OpticalParameters::Instance()->SetWLSTimeProfile(const G4String& val);
  - default value: delta
- Set the verbosity of the first WLS process. 0 = silent; 1 = initialisation; 2 = during tracking.
  - macro command: /process/optical/wls/verbose 1
  - C++ statement: G4OpticalParameters::Instance()->SetWLSVerboseLevel(G4int verboseLevel);
  - default value: 1
- Set the time profile of the second WLS process to be either "delta" or "exponential":
  - macro command: /process/optical/wls/setTimeProfile value
  - C++ statement: G4OpticalParameters::Instance()->SetWLSTimeProfile(const G4String& val);
  - default value: delta
- Set the verbosity of the second WLS process. 0 = silent; 1 = initialisation; 2 = during tracking.
  - macro command: /process/optical/wls/verbose 1
  - C++ statement: G4OpticalParameters::Instance()->SetWLSVerboseLevel(G4int verboseLevel);
  - default value: 1

Mie Scattering

Mie Scattering (or Mie solution) is an analytical solution of Maxwell's equations for scattering of optical photons by spherical particles. It is significant only when the radius of the scattering object is of order of the wave length. The analytical expressions for Mie Scattering are complicated since they are a series sum of Bessel functions. One common approximation made is called *Henyey-Greenstein (HG)*. The implementation in GEANT4 follows the HG approximation (for details see the Physics Reference Manual) and the treatment of polarization and momentum are similar to that of Rayleigh scattering. We require the final polarization direction to be perpendicular to the momentum direction. We also require the final momentum, initial polarization, and final polarization to be in the same plane.

The process requires a `G4MaterialPropertiesTable` to be filled by the user with Mie scattering length data (entered with the name `MIEHG`) analogous to Rayleigh scattering. The Mie scattering attenuation length is the average distance traveled by a photon before it is Mie scattered in the medium and it is the distance returned by the GetMeanFreePath method. In practice, the user not only needs to provide the attenuation length of Mie scattering, but also needs to provide the constant parameters of the approximation g_f, g_b, and r_f, with `AddConstProperty` and with the names `MIEHG_FORWARD`, `MIEHG_BACKWARD`, and `MIEHG_FORWARD_RATIO`, respectively; see extended example optical/OpNovice.

Table 5.7: Material properties for the optical Mie scattering process.

Name	Type	Description	Unit category
MIEHG	Energy-dependent	Attenuation length	Length
MIEHG_BACKWARD	Constant	Parameter used in sampling of scattering direction	Unitless
MIEHG_FORWARD	Constant	Parameter used in sampling of scattering direction	Unitless
MIEHG_FORWARD_RATIO	Constant	Parameter used in sampling of scattering direction	Unitless

This `G4OpticalParameters` command can be used to configure the process.

- Set the verbosity of the Mie scattering process. 0 = silent; 1 = initialisation; 2 = during tracking.
  - macro command: /process/optical/mie/verbose 1
  - C++ statement: G4OpticalParameters::Instance()->SetMieVerboseLevel(G4int verboseLevel);
  - default value: 1

Boundary Process

Optical photons interact with boundaries between volumes, for example, to reflect or refract. There are various methods of specifying a surface and assigning properties to it.

Reference: E. Hecht and A. Zajac, Optics [Hecht1974]

When a photon arrives at a medium boundary its behavior depends on the nature of the two materials that join at that boundary. Medium boundaries may be formed between two dielectric materials or a dielectric and a metal.

In the case of an interface between a dielectric and a metal, the photon can be absorbed by the metal or reflected back into the dielectric. If the photon is absorbed it can be detected according to the photoelectron efficiency of the metal.

As expressed in Maxwell's equations, Fresnel reflection and refraction are intertwined through their relative probabilities of occurrence. Therefore neither of these processes, nor total internal reflection, are viewed as individual processes deserving separate class implementation. Nonetheless, an attempt was made to adhere to the abstraction of having independent processes by splitting the code into different methods where practicable.

The program defaults to the GLISUR model and *polished* surface finish when no specific model and surface finish is specified by the user. In the case of a dielectric-metal interface, or when the GLISUR model is specified, the only surface finish options available are *polished* or *ground*. For dielectric-metal surfaces, the `G4OpBoundaryProcess` also defaults to unit reflectivity and zero detection efficiency. In cases where the user specifies the UNIFIED model (Fig. 5.1), but does not otherwise specify the model reflection probability constants, the default becomes Lambertian reflection.

When an optical photon arrives at a boundary it is absorbed if the medium of the volume being left behind has no index of refraction defined. A photon is also absorbed in case of a dielectric-dielectric polished or ground surface when the medium about to be entered has no index of refraction. It is absorbed for backpainted surfaces when the surface has no index of refraction.

The boundary process may produce warnings that can likely be ignored. These warnings result from the way the boundary process handles changes to which volume the particle is in. If the step length is 0, the boundary status is set to `StepTooSmall` and the appropriate group velocity is set. In practice, rather than comparing the step length to 0, the step is compared to the geometrical tolerance. For steps with length slightly over the geometric tolerance, it is not clear in the code logic which volume the photon is entering, so the group velocity is not set and a warning issued.

```
-------- WWWW ------- G4Exception-START -------- WWWW -------
*** G4Exception : OpBoun06
      issued by : G4OpBoundaryProcess
G4OpBoundaryProcess: Opticalphoton step length: 2.46916e-09 mm.
This is larger than the threshold 1e-09 mm to set status StepTooSmall.
Boundary scattering may be incorrect.

*** This is just a warning message. ***
-------- WWWW -------- G4Exception-END --------- WWWW -------
```

In most cases this warning may be ignored. It may be suppressed by setting optical verbosity, or boundary process verbosity, to 0.

Specifying the surface

The physical surface object specifies which model the boundary process should use to simulate interactions with that surface. In addition, the physical surface can have a material property table all its own. The usage of this table allows all specular constants to be wavelength dependent. In case the surface is painted or wrapped (but not a cladding), the table may include the thin layer's index of refraction. This allows the simulation of boundary effects at the intersection between the medium and the surface layer, as well as the Lambertian reflection at the far side of the thin layer. This occurs within the process itself and does not invoke the `G4Navigator`. Combinations of surface finish properties, such as *polished* or *ground* and *front painted* or *back painted*, enumerate the different situations which can be simulated.

There are three methods of specifying an optical surface.

1. For the simple case of a perfectly smooth interface between two dielectric materials, all the user needs to provide are the refractive indices (`RINDEX`) of the two materials stored in their respective `G4MaterialPropertiesTables`.
2. A *skin surface* is the surface entirely surrounding a logical volume. This is useful where a volume is coated with a reflector and placed into many different mother volumes. A limitation is that the skin surface can only have one and the same optical property for all of the enclosed volume's sides. An optical surface is created and defined using `G4OpticalSurface`. This surface is assigned to a logical volume using the class `G4LogicalSkinSurface`.
3. A *border surface* is defined by specifying the ordered pair of physical volumes touching at the surface. Because the pair of physical volumes is ordered, the user may specify different optical properties for photons arriving from the reverse side of the same interface. For the optical boundary process to use a border surface, the two volumes must have been positioned with `G4PVPlacement`. The ordered combination can exist at many

places in the simulation. An optical surface is created and defined using `G4OpticalSurface`. This surface is assigned to the ordered pair of physical volumes using the class `G4LogicalBorderSurface`.

If the geometry boundary has a *border surface* this surface takes precedence, otherwise the program checks for *skin surfaces*. The *skin surface* of the daughter volume is taken if a daughter volume is entered, or else the program checks for a *skin surface* of the current volume. When the optical photon leaves a volume without entering a daughter volume the *skin surface* of the current volume takes precedence over that of the volume about to be entered.

Listing 5.7: Defining border and skin surfaces (from extended example OpNovice).

```
#include "G4LogicalBorderSurface.hh"
#include "G4LogicalSkinSurface.hh"
#include "G4OpticalSurface.hh"

// The experimental Hall
//
G4Box* expHall_box = new G4Box("World", fExpHall_x, fExpHall_y, fExpHall_z);

G4LogicalVolume* expHall_log =
  new G4LogicalVolume(expHall_box, air, "World", 0, 0, 0);

G4VPhysicalVolume* expHall_phys =
  new G4PVPlacement(0, G4ThreeVector(), expHall_log, "World", 0, false, 0);

// The Water Tank
//
G4Box* waterTank_box = new G4Box("Tank", fTank_x, fTank_y, fTank_z);

G4LogicalVolume* waterTank_log =
  new G4LogicalVolume(waterTank_box, water, "Tank", 0, 0, 0);

G4VPhysicalVolume* waterTank_phys = new G4PVPlacement(
  0, G4ThreeVector(), waterTank_log, "Tank", expHall_log, false, 0);

// The Air Bubble
//
G4Box* bubbleAir_box = new G4Box("Bubble", fBubble_x, fBubble_y, fBubble_z);

G4LogicalVolume* bubbleAir_log =
  new G4LogicalVolume(bubbleAir_box, air, "Bubble", 0, 0, 0);

new G4PVPlacement(0, G4ThreeVector(0., 2.5 * m, 0.), bubbleAir_log, "Bubble",
                  waterTank_log, false, 0);

G4OpticalSurface* opWaterSurface = new G4OpticalSurface("WaterSurface");
opWaterSurface->SetType(dielectric_LUTDAVIS);
opWaterSurface->SetFinish(Rough_LUT);
opWaterSurface->SetModel(DAVIS);

G4LogicalBorderSurface* waterSurface = new G4LogicalBorderSurface(
  "WaterSurface", waterTank_phys, expHall_phys, opWaterSurface);

G4OpticalSurface* opAirSurface = new G4OpticalSurface("AirSurface");
opAirSurface->SetType(dielectric_dielectric);
opAirSurface->SetFinish(polished);
opAirSurface->SetModel(glisur);

G4LogicalSkinSurface* airSurface =
  new G4LogicalSkinSurface("AirSurface", bubbleAir_log, opAirSurface);
```

Surface models: Defining the boundary properties

There are several models to describe the interactions of optical photons at a surface. These are defined in an emum:

Listing 5.8: Optical surface models.

```
enum G4OpticalSurfaceModel
{
  glisur,   // original GEANT3 model
  unified,  // UNIFIED model
  LUT,      // Look-Up-Table model (LBNL model)
  DAVIS,    // DAVIS model
  dichroic  // dichroic filter
};
```

The model is specified by calling the `SetModel(G4int model)` method of the surface.

No surface defined

In the special case where no surface has been defined, but the two volumes defining the surface have the refractive index defined using the material property `RINDEX`, the surface is taken to be perfectly smooth, and both materials are taken to be dielectric. The photon can undergo total internal reflection, refraction or reflection, depending on the photon's wavelength, angle of incidence, and the refractive indices on both sides of the boundary. Furthermore, reflection and transmission probabilities are sensitive to the state of linear polarization.

The UNIFIED model

One implementation of the `G4OpBoundaryProcess` class employs the UNIFIED model [Levin1996] of the DETECT program [Knoll1988]. It applies to dielectric-dielectric interfaces and tries to provide a realistic simulation, which deals with all aspects of surface finish and reflector coating. The surface may be assumed as smooth and covered with a metallized coating representing a specular reflector with given reflection coefficient, or painted with a diffuse reflecting material where Lambertian reflection occurs. The surfaces may or may not be in optical contact with another component and most importantly, one may consider a surface to be made up of micro-facets with normal vectors that follow given distributions around the nominal normal for the volume at the impact point. For very rough surfaces, it is possible for the photon to inversely aim at the same surface again after reflection of refraction and so multiple interactions with the boundary are possible within the process itself and without the need for relocation by `G4Navigator`.

The UNIFIED model (Fig. 5.1) provides for a range of different reflection mechanisms. The specular lobe constant (material property name `SPECULARLOBECONSTANT`) represents the reflection probability about the normal of a micro facet. The specular spike constant (material property name `SPECULARSPIKECONSTANT`), in turn, illustrates the probability of reflection about the average surface normal. The diffuse lobe constant is for the probability of internal Lambertian reflection, and finally the back-scatter spike constant (material property name `BACKSCATTERCONSTANT`) is for the case of several reflections within a deep groove with the ultimate result of exact back-scattering. The four probabilities add up to one, with the diffuse lobe constant being calculated by the code from other other three values that the user entered. The reader may consult the reference for a thorough description of the model.

It is possible to specify that a given fraction of photons are absorbed at the surface, or transmitted without change in direction or polarization. This is applicable for dielectric_dielectric interfaces that are not backpainted. The material properties `REFLECTIVITY` and `TRANSMITTANCE` are used. By default, `REFLECTIVITY` equals 1 and `TRANSMITTANCE` equals 0. At a surface interaction, a random number is chosen. If the random number is greater than the sum of the values of `REFLECTIVITY` and `TRANSMITTANCE` at the photon energy, the photon is absorbed. Otherwise, if the random number is greater than the `REFLECTIVITY` value, the photon is transmitted. Otherwise, the usual calculation of scattering takes place.

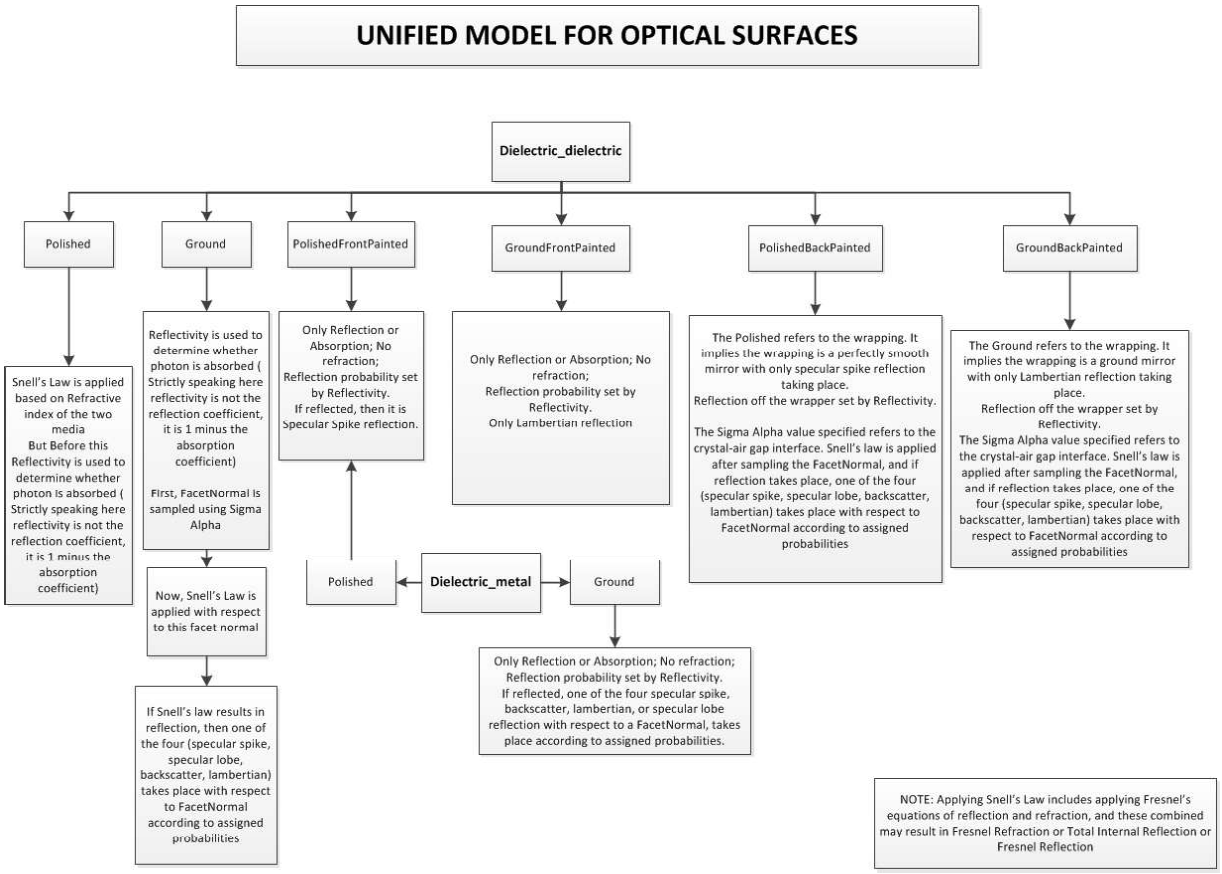

Fig. 5.1: Diagram of the UNIFIED Model for Optical Surfaces (courtesy A. Shankar)

If a photon is absorbed at the boundary, it may by detected–that is, its status set to "Detect" and its energy deposited locally. The material property `EFFICIENCY` is used to specify the fraction of photons detected.

The `sigma_alpha` parameter allows sepcification of the surface roughness. The facet normal is chosen from a Gaussian distribution with this sigma, with a maximum of the lower of 1 and 4 times sigma_alpha.

Listing 5.9: Dielectric-dielectric surface properties defined via the `G4OpticalSurface`.

```
G4VPhysicalVolume* volume1;
G4VPhysicalVolume* volume2;

G4OpticalSurface* OpSurface = new G4OpticalSurface("name");

G4LogicalBorderSurface* Surface = new
  G4LogicalBorderSurface("name",volume1,volume2,OpSurface);

OpSurface->SetType(dielectric_dielectric);
OpSurface->SetModel(unified);
OpSurface->SetFinish(groundbackpainted);
OpSurface->SetSigmaAlpha(0.1);

std::vector<G4double> pp = {2.038*eV, 4.144*eV};
std::vector<G4double> specularlobe = {0.3, 0.3};
std::vector<G4double> specularspike = {0.2, 0.2};
std::vector<G4double> backscatter = {0.1, 0.1};
std::vector<G4double> rindex = {1.35, 1.40};
std::vector<G4double> reflectivity = {0.3, 0.5};
std::vector<G4double> efficiency = {0.8, 0.1};

G4MaterialPropertiesTable* SMPT = new G4MaterialPropertiesTable();

SMPT->AddProperty("RINDEX", pp, rindex);
SMPT->AddProperty("SPECULARLOBECONSTANT", pp, specularlobe);
SMPT->AddProperty("SPECULARSPIKECONSTANT", pp, specularspike);
SMPT->AddProperty("BACKSCATTERCONSTANT", pp, backscatter);
SMPT->AddProperty("REFLECTIVITY", pp, reflectivity);
SMPT->AddProperty("EFFICIENCY", pp, efficiency);

OpSurface->SetMaterialPropertiesTable(SMPT);
```

The Glisur model

The original GEANT3.21 implementation of this process is also available via the GLISUR methods flag, as shown in Listing 5.10. Note that there is considerable overlap of the Glisur and UNIFIED models in the code. The surface roughness is specified with the `polish` parameter.

Listing 5.10: Dielectric metal surface properties defined via the `G4OpticalSurface`.

```
G4LogicalVolume* volume_log;

G4OpticalSurface* OpSurface = new G4OpticalSurface("name");

G4LogicalSkinSurface* Surface = new
  G4LogicalSkinSurface("name",volume_log,OpSurface);

OpSurface->SetType(dielectric_metal);
OpSurface->SetFinish(ground);
OpSurface->SetModel(glisur);
OpSurface->SetPolish(0.8);
```

(continues on next page)

(continued from previous page)

```
G4MaterialPropertiesTable* OpSurfaceProperty = new G4MaterialPropertiesTable();

OpSurfaceProperty->AddProperty("REFLECTIVITY", pp, reflectivity);
OpSurfaceProperty->AddProperty("EFFICIENCY", pp, efficiency);

OpSurface->SetMaterialPropertiesTable(OpSurfaceProperty);
```

LBNL look-up tables (LUT)

Janecek and Moses [Janecek2010] built an instrument for measuring the angular reflectivity distribution inside of BGO crystals with common surface treatments and reflectors applied. These results have been incorporated into the GEANT4 code. A third class of reflection type besides dielectric_metal and dielectric_dielectric is added: dielectric_LUT. The distributions have been converted to 21 look-up-tables (LUT); so far for 1 scintillator material (BGO) x 3 surface treatments x 7 reflector materials. The modified code allows the user to specify the surface treatment (rough-cut, chemically etched, or mechanically polished), the attached reflector (Lumirror, Teflon, ESR film, Tyvek, or TiO2 paint), and the bonding type (air-coupled or glued). The glue used is MeltMount, and the ESR film used is VM2000. Each LUT consists of measured angular distributions with 4º by 5º resolution in theta and phi, respectively, for incidence angles from 0º to 90º, in 1º-steps. The code might in the future be updated by adding more LUTs, for instance, for other scintillating materials (such as LSO or NaI). To use these LUT the user has to download them from Geant4 Software Download and set an environment variable, `G4REALSURFACEDATA`, to the directory of `geant4/data/RealSurface2.2`.

The enumeration G4OpticalSurfaceFinish includes:

```
polishedlumirrorair,         // mechanically polished surface, with lumirror
polishedlumirrorglue,        // mechanically polished surface, with lumirror & meltmount
polishedteflonair,           // mechanically polished surface, with teflon
polishedtioair,              // mechanically polished surface, with tio paint
polishedtyvekair,            // mechanically polished surface, with tyvek
polishedvm2000air,           // mechanically polished surface, with esr film
polishedvm2000glue,          // mechanically polished surface, with esr film & meltmount
etchedlumirrorair,           // chemically etched surface, with lumirror
etchedlumirrorglue,          // chemically etched surface, with lumirror & meltmount
etchedteflonair,             // chemically etched surface, with teflon
etchedtioair,                // chemically etched surface, with tio paint
etchedtyvekair,              // chemically etched surface, with tyvek
etchedvm2000air,             // chemically etched surface, with esr film
etchedvm2000glue,            // chemically etched surface, with esr film & meltmount
groundlumirrorair,           // rough-cut surface, with lumirror
groundlumirrorglue,          // rough-cut surface, with lumirror & meltmount
groundteflonair,             // rough-cut surface, with teflon
groundtioair,                // rough-cut surface, with tio paint
groundtyvekair,              // rough-cut surface, with tyvek
groundvm2000air,             // rough-cut surface, with esr film
groundvm2000glue             // rough-cut surface, with esr film & meltmount
```

To use an LBNL look-up-table, all the user needs to specify for an `G4OpticalSurface` is: `SetType(dielectric_LUT), SetModel(LUT)` and for example, `SetFinish(polishedtyvekair)`.

Note that the LBNL look-up tables were the first optical surface LUT implemented in GEANT4. Where the term "LUT" is used, it often refers to the LBNL LUT.

Davis look-up tables (LUTDAVIS)

Another model for optical surface interactions is called the LUT Davis model [RoncaliCherry2013], [Stockhoff2017], [Roncali2017]. The model is based on measured surface data and allows the user to choose from a list of available surface finishes. Provided are a rough and a polished L(Y)SO surface that can be used without reflector, or in combination with a specular reflector (e.g. ESR) or a Lambertian reflector (e.g. Teflon). The specular reflector can be coupled to the crystal with air or optical grease. Teflon tape is wrapped around the crystal with 4 layers.

Table 5.8: Surface names of available LUTs.

	Bare	Teflon	ESR + Air	ESR + Optical Grease
Rough	Rough_LUT	RoughTeflon_LUT	RoughESR_LUT	RoughESRGrease_LUT
Polished	Polished_LUT	PolishedTeflon_LUT	PolishedESR_LUT	PolishedESRGrease_LUT

In the LUT database, typical roughness parameters obtained from the measurements are provided to characterize the type of surface modelled:

$$\begin{aligned} \text{ROUGH} &: R_a = 0.48\mu\text{m}, \sigma = 0.57\mu\text{m}, R_{pv} = 3.12\mu\text{m} \\ \text{POLISHED} &: R_a = 20.8\ \text{nm}, \sigma = 26.2\ \text{nm}, R_{pv} = 34.7\ \text{nm} \end{aligned} \tag{5.1}$$

with R_a = average roughness; σ = rms roughness, R_{pv} = peak-to-valley ratio.

The detector surface, called `Detector_LUT`, defines a polished surface coupled to a photodetector with optical grease or a glass interface (similar index of refraction 1.5). To use `Detector_LUT`, the surface property `EFFICIENCY` must be greater than 0. Any surface can be used as a detector surface when the `EFFICIENCY` is set to 1.

To enable the LUT Davis Model, the user needs to specify for a `G4OpticalSurface`: `SetType(dielectric_LUTDAVIS)`, `SetModel(DAVIS)` and also, for example, `SetFinish(Rough_LUT)`. The user also has to download data files from Geant4 Software Download and set an environment variable, `G4REALSURFACEDATA`, to the directory of `geant4/data/RealSurface2.2`.

Background

The crystal topography is obtained with atomic force microscopy (AFM). From the AFM data, the probability of reflection (1) and the reflection directions (2) are computationally determined, for incidence angles ranging from 0° to 90°. Each LUT is computed for a given surface and reflector configuration. The reflection probability in the LUT combines two cases: directly reflected photons from the crystal surface and photons that are transmitted to the reflector surface and later re-enter the crystal. The key operations of the reflection process are the following: The angle between the incident photon (Old Momentum) and the surface normal are calculated. The probability of reflection is extracted from the first LUT. A Bernoulli test determines whether the photon is reflected or transmitted. In case of reflection two angles are drawn from the reflection direction LUT.

Thin film optical coatings

L.Cappellugola, M. Dupont, S. Curtoni and C. Morel, Aix Marseille Univ.

Photons can be transmitted through a thin film that has a thickness of the order of light wavelength. Interference phenomena and frustrated transmission beyond the limit angle have then to be considered.

New in version 11.1, Geant4can account for thin film interfaces. This is achieved by defining a new method that provides a transmission probability for optical tracking of optical photons. This new method is called `CoatedDielectricDielectric()` and considers only two physical and logical media, and the refractive index and thickness of the thin film separating these two media.

Reflectance and transmittance of an optical coating

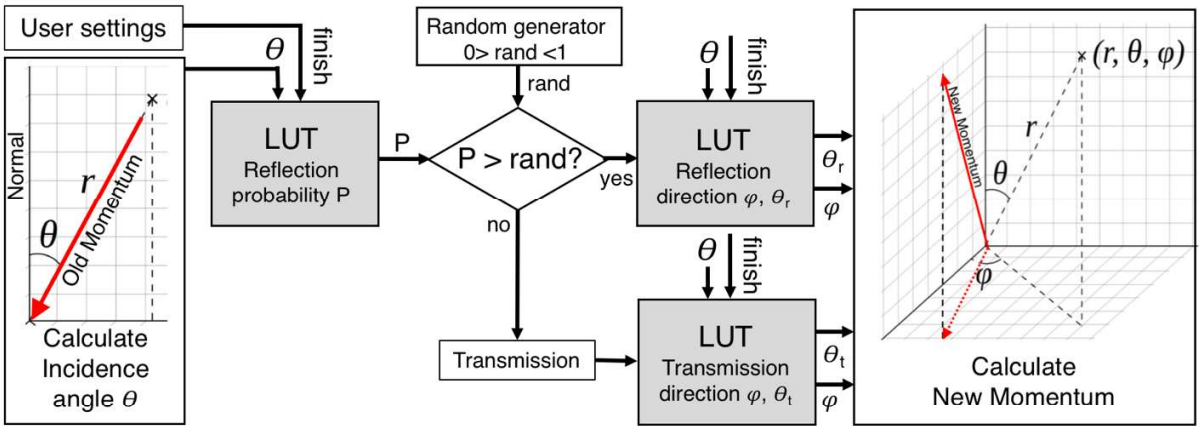

Fig. 5.2: Old Momentum to New Momentum. The old momentum is the unit vector that describes the incident photon. The reflected/transmitted photon is the New Momentum described by two angles ϕ and θ.

Let r_{ij} be the Fresnel coefficient describing an interface between media i and j, with refractive indices n_i and n_j. If the incidence angle θ_i is greater than the limit angle $\theta_{\lim}$, the term $n_j cos(\theta_j)$ in the Fresnel coefficients with $n_i sin(\theta_i) = n_j sin(\theta_j)$ will be replaced by $i\gamma = \sqrt{n_i^2 \sin {\theta_i}^2 - n_j^2}$. Hence we get for $\theta > \theta_l$:

$$r_{TE} = \frac{n_i cos(\theta_i) - i\gamma}{n_i cos(\theta_i) + i\gamma}$$

$$r_{TM} = \frac{n_i i\gamma - n_j^2 cos(\theta_i)}{n_i i\gamma + n_j^2 cos(\theta_i)}$$

In case of a transmission from medium 1 to medium 3 through a thin passivation film (medium 2), the reflection coefficient for incidence angles lesser and greater than the limit angle $\theta_{\lim} = \arcsin(n_2/n_1)$ is given by:

$$r_{\theta<\theta_{\lim}} = \frac{r_{12} + r_{23}e^{2i\beta'}}{1 + r_{12}r_{23}e^{2i\beta'}} \tag{5.2}$$

$$r_{\theta>\theta_{\lim}} = \frac{r_{12} + r_{23}e^{2\beta''}}{1 + r_{12}r_{23}e^{2\beta''}} \tag{5.3}$$

where $\beta' = k_2 d \cos\theta_2$ and $\beta'' = -kd\gamma$ with d the layer thickness and $k_2 = 2\pi/\lambda_2 = 2\pi n_2/\lambda$ the photon wavenumber in medium 2 with λ and k the photon wavelength and wavenumber in vacuum.

Integration of frustrated transmission in Geant4

The main method of `G4OpBoundaryProcess` class is the `PostStepDoIt()` method. This method starts by getting refractive indices of both media on each side of the interface, instantiating the `Type`, `Model` and `Finish` properties of this optical surface and calling the corresponding function in order to analyse the interaction on the interface correctly.

A new surface type named `coated` is used for an interface between two dielectric volumes separated by a thin film and a new method named `CoatedDielectricDielectric()`, which is called by `PostStepDoIt()`. Only the two volumes on each side of the thin film have to be physically and logically defined. The surface can be parametrized with two arrays named `CoatedRindex` and `CoatedThickness`, which correspond to the thin film layer refractive index and thickness, respectively, as a function of wavelength. Frustrated transmission for incidence angles superior to the limit angle can be enabled or disabled via the boolean parameter `CoatedFrustratedTransmission` in order to assess the impact of frustrated transmission.

An example is given with the extended example OpNovice2 and macro coated.mac.

Results of the implementation

Transmittance through a thin optical coating

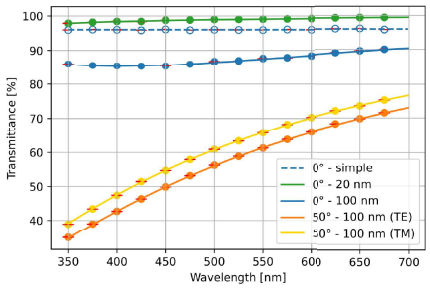

Fig. 5.3: Transmittance as a function of wavelength through an interface with a thin film. In blue and green: transmittances for a normal incidence, in orange and yellow: transmittances for a 50° incidence. Curves represent theoretical transmittances for a simple interface (dotted line) and for a thin film interface (continuous lines). Markers represent the transmittance estimated by the Geant4 simulation for a simple interface (empty dots) and for a thin film interface (full dots).

We fix $n_1 = n_3 = 1.5$ and $n_2 = 1.0$ (the limit angle between medium 1 and 2 is $\theta_l = 41.8^\circ$. Two examples are presented in Fig. 5.3 for different coating thicknesses d and incidence angle θ. The first example stands for a simple interface, comprising only one dioptre from medium 1 to medium 2. The theoretical transmittance, which is given by:

$$T = 1 - R = 1 - rr^* \tag{5.4}$$

with $r = r_{TE} = r_{TM}$ given by the equations above for $\theta_i = \theta_j = 0$ corresponding to a normal incidence, is presented with dashed blue line and the fraction of photons transmitted through this dioptre estimated by the Geant4 simulation of 2,800,000 visible photons are represented by blue empty dots. The second example highlights the consequences of light transmission through a thin layer by adding a medium 3 after the medium 2 in such a way that the thickness d of the medium 2 is smaller than the wavelength of the light. Theoretical transmittances for different thicknesses d and incidence angles θ ($\theta = 0$ for a normal incidence and $\theta = 50^\circ$ for an incidence angle superior to the limit angle θ_{lim}) are represented with continuous lines. The fraction of photons transmitted through these different interfaces estimated by the Geant4 simulation of 2,800,000 visible photons are represented by full dots.

In case of an oblique incidence, the transmittance for both the Transverse Electric (TE) and the Transverse Magnetic (TM) polarizations are shown and compared to the theoretical transmittances using (5.4) with $r = r_{TE}$ or $r = r_{TM}$, respectively. For an incidence angle superior to the limit angle (orange curves), the transmittance is non-zero because of the frustrated transmission of light.

The perfect agreement between the simulated and theoretical transmittances assess the correct implementation of the model in the Geant4 software.

Transmittance through a window optically coated on both its faces

In this section we model two successive interfaces at the input and output faces of a window of refractive index $n_3 = 2.0$ that are optically coated with 100 nm of an oxide of refractive index $n_2 = 1.5$. This coated window is placed in air ($n_1 = 1.0$).

For a normal incidence, the theoretical transmittance involving multiples reflections (MR) of light inside the window is estimated by T_{MR}:

$$T_{MR} = \frac{T^2}{1 - R^2}$$

where $R = rr^*$ is the reflectance with r given by the Eqs. (5.2) or (5.3) and $T = 1 - R$.

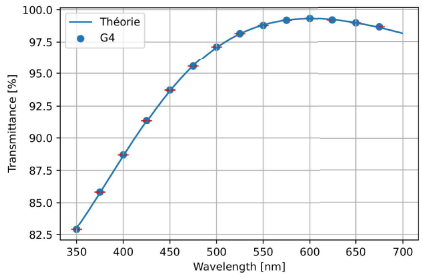

Fig. 5.4: Transmittance for normal incidence through a coated window ($n_1 = 1.0$, $n_2 = 1.5$ and $n_3 = 2.0$). The continuous line represents the theoretical transmittance and the markers represents the transmittance estimated by the Geant4 simulation of 2,800,000 visible photons.

Fig. 5.4 shows the transmittance for a normal incidence through the coated window estimated by the simulation of 2,800,000 visible photons. Here again, the perfect agreement between the simulated and theoretical transmittances assess the correct implementation of the model in the Geant4software.

Complex index of refraction

The reflectivity off a metal surface can also be calculated by way of a complex index of refraction. Instead of storing the `REFLECTIVITY` directly, the user stores the real part (`REALRINDEX`) and the imaginary part (`IMAGINARYRINDEX`) as a function of photon energy separately in the G4MaterialPropertyTable. GEANT4 then calculates the reflectivity depending on the incident angle, photon energy, degree of TE and TM polarization, and this complex refractive index.

If it is desired that photons are not absorbed at the surface, but instead are transmitted into the material, the user can set the transmittance (TRANSMITTANCE) to 1. If this setting is used together with a ground finish and a non-zero value of sigma_alpha, some photons are still absorbed at the surface. If the absorption at the surface should be completely prevented, sigma_alpha must be zero or a polished finish needs to be used. Additionally, the absorption length (ABSLENGTH) should be defined for the material, otherwise the material would be fully transparent. Note that Geant4 does not calculate the absorption length, which is a material property, from the imaginary part of the refractive index, which is a surface property. The user needs to provide both properties individually. The macro complexRindex.mac (below) in the example OpNovice2 demonstrates the usage of the complex refractive index.

Listing 5.11: A macro for use with OpNovice2 demonstrating the usage of the complex index of refraction.

```
/control/verbose 2
/tracking/verbose 0
/run/verbose 0
/process/optical/verbose 1
/control/cout/ignoreThreadsExcept 0

/opnovice2/worldMaterial G4_Al
/opnovice2/worldProperty GROUPVEL  0.000002 299.792     0.000008 299.792
/opnovice2/worldProperty RINDEX    0.000002 1.38        0.000004 0.28        0.000006 0.13
↪ 0.000008 0.077
/opnovice2/worldProperty ABSLENGTH 0.000002 6.75e-9     0.000004 6.65e-9     0.000006 6.96e-9
↪ 0.000008 7.45e-9

/opnovice2/boxMaterial G4_AIR
/opnovice2/boxProperty RINDEX    0.000002 1.01    0.000008 1.01
/opnovice2/boxProperty ABSLENGTH 0.000002 1000000 0.000005 2000000 0.000008 3000000
```

(continues on next page)

(continued from previous page)

```
/opnovice2/surfaceModel unified
/opnovice2/surfaceType dielectric_metal
/opnovice2/surfaceFinish polished
/opnovice2/surfaceSigmaAlpha 0 # only relevant for ground finish

/opnovice2/surfaceProperty REALRINDEX           0.000002 1.38     0.000004 0.28     0.000006 0.
↪13     0.000008 0.077
/opnovice2/surfaceProperty IMAGINARYRINDEX      0.000002 7.31     0.000004 3.71     0.000006 2.
↪40     0.000008 1.71
/opnovice2/surfaceProperty TRANSMITTANCE        0.000002 1        0.000006 1
/opnovice2/surfaceProperty SPECULARLOBECONSTANT 0.000002 1        0.000006 1 # only relevant for␣
↪ground finish

/run/initialize

/gun/particle opticalphoton
/gun/energy 4 eV
/gun/position 90 0 0 cm
/gun/direction 1 0 0
/opnovice2/gun/optPhotonPolar 0. deg
/opnovice2/gun/randomDirection true

/analysis/setFileName complexRindex_0
/analysis/h1/set 20 90 0 90  # deg
/analysis/h1/set 21 90 0 90  # deg
/run/printProgress 100000
/run/beamOn 500000

/analysis/setFileName complexRindex_90
/opnovice2/gun/optPhotonPolar 90. deg
/run/beamOn 500000
```

Dichroic filter

A dielectric-dichroic boundary may be specified by supplying a dichroic vector to the optical surface.

Configuration

Table 5.9: Material and surface properties for the optical boundary scattering process.

Name	Type	Material or surface	Description
BACKSCATTERCONSTANT	Energy-dependent	Surface	Unified model
EFFICIENCY	Energy-dependent	Surface	Chance of an absorbed photon to be detected
COATEDFRUSTRATEDTRANSMISSION	Constant	Surface	Whether to use frustrated transmission for thin coatings. Values are either 1 (on) or 0 (off)
GROUPVEL	Energy-dependent	Material	If specified, set the photon velocity to this value on entering the material, if undergoing Fresnel refraction or transmission
IMAGINARYRINDEX	Energy-dependent	Either	Imaginary part of complex refractive index
REFLECTIVITY	Energy-dependent	Surface	1 minus the absorption coefficient
REALRINDEX	Energy-dependent	Either	Real part of complex refractive index
RINDEX	Energy-dependent	Either	Refractive index
COATEDRINDEX	Energy-dependent	Surface	Refractive index of thin coating
SPECULARLOBECONSTANT	Energy-dependent	Surface	Unified model
SPECULARSPIKECONSTANT	Energy-dependent	Surface	Unified model
SURFACEROUGHNESS	Constant	Surface	Parameter to express surface roughness for dielectric_dielectric surfaces, leading to Lambertian scattering
COATEDTHICKNESS	Energy-dependent	Surface	Thickness of thin coating
TRANSMITTANCE	Energy-dependent	Surface	For dielectric_dielectric surfaces

These `G4OpticalParameters` commands can be used to configure the process.

- Set the verbosity of the boundary scattering process. 0 = silent; 1 = initialisation; 2 = during tracking.
  - macro command: /process/optical/boundary/verbose 1
  - C++ statement: G4OpticalParameters::Instance()->SetBoundaryVerboseLevel(G4int verboseLevel);
  - default value: 1
- Call the sensitive detector automatically.
  - macro command: /process/optical/boundary/setInvokeSD
  - C++ statement: G4OpticalParameters::Instance()->SetBoundaryInvokeSD(G4bool val)
  - default value: false