Cover Sheet

Optical Photon Processes in GEANT4

Peter Gumplinger, TRIUMF/GEANT4

G4 Space Users' Forum at ESTEC, January 2003

Abstract

GEANT4 can realistically model the optics of scintillation and Cerenkov detectors. The simulation may commence with the propagation of a charged particle and end with the detection of the ensuing optical photons on photo sensitive areas, all within the same event loop. The lecture will introduce this functionality and explain how it can be employed by the user

Optical Photon Processes in GEANT4

- Concept of "optical Photon" in G4

  λ >> atomic spacing

- G4OpticalPhoton: wave like nature of EM radiation

- G4OpticalPhoton <=|=> G4Gamma

  (no smooth transition)

```
// optical photon
G4OpticalPhoton::OpticalPhoton();
```

- Define a (spin) vector for the photon, added as data member to the G4DynamicParticle description class:

```
aphoton->SetPolarization(ux,uy,uz); // unit vector!!!
```

Optical photon production in Geant4

in /processes/electromagnetic/xrays

- Cerenkov Process
- Scintillation Process
- Transition Radiation

Geant4 catalogue of Processes at optical Wavelengths

in /processes/optical

- Refraction and Reflection at medium boundaries
- Bulk Absorption
- Rayleigh scattering

ExampleN06 at /examples/novice/N06

- The optical properties of the medium which are key to the implementation of these types of processes are stored as entries in a properties table linked to the material.
- Properties are expressed as a function of the photon's momentum.

```
// Water

G4double a = 1.01*g/mole;
G4Element* elH = new G4Element(name="Hydrogen", symbol="H", z=1.,a);
a = 16.00*g/mole;
G4Element* elO = new G4Element(name="Oxygen",symbol="O",z=8.,a)

density = 1.0*g/cm3;
G4Material* Water = new G4Material(name="Water",density,nel=2);
Water -> AddElement(elH,2);
Water -> AddElement(elO,1);

const G4int NUMENTRIES = 32;

G4double ppckov[NUMENTRIES] = { 2.034*eV, ……, 4.136*eV};
G4double rindex[NUMENTRIES] = 1.3435, ……., 1.3608};
G4double absorption[NUMENTRIES] = 344.8*cm, ……., 1450.0*cm};

G4MaterialPropertiesTable *MPT = new G4MaterialPropertiesTable();

MPT -> AddProperty("RINDEX",ppckov,rindex,NUMENTRIES};
MPT -> AddProperty("ABSLENGTH",ppckov,absorption,NUMENTRIES};

water -> SetMaterialPropertiesTable(MPT);
```

Cerenkov Process

- Cerenkov light occurs when a charged particle moves through a medium faster than the medium's group velocity of light.
- Photons are emitted on the surface of a cone, and as the particle slows down:

  (a) the cone angle decreases

  (b) the emitted photon frequency increases

  (c) and their number decreases
- Cerenkov photons have inherent polarization perpendicular to the cone's surface.

G4Cerenkov Implementation Details

- Cerenkov photon origins are distributed rectilinear over the step even in the presence of a magnetic field
- Cerenkov photons are generated only in media where the user has provided an index of refraction
- An average number of photon is calculated for the wavelength interval in which the index of refraction is given

User Options

- Suspend primary particle and track Cerenkov photons first
- Set the average number of Cerenkov photons per step (The actual number generated in any given step will be slightly different because of the statistical nature of the process)

in ExptPhysicsList:

```
#include "G4Cerenkov.hh"

G4Cerenkov* theCerenkovProcess = new G4Cerenkov("Cerenkov");

theCerenkovProcess -> SetTrackSecondariesFirst(true);

G4int MaxNumPhotons = 300;
theCerenkovProcess->SetMaxNumPhotonsPerStep(MaxNumPhotons);
```

Scintillation Process

- Number of photons generated proportional to the energy lost during the step
- Emission spectrum sampled from empirical spectra
- Isotropic emission
- Uniform along the track segment
- With random linear polarization
- Emission time spectra with one or two exponential (fast/slow) decay time constant(s).

G4Scintillation Implementation Details

- Scintillation material has a characteristic light yield
- The statistical yield fluctuation is either broadened due to impurities for doped crystals or narrower as a result of the Fano Factor
- Suspend primary particle and track scintillation photons first

Note: Material properties were up to now attached to the process (and not the material). This meant that GEANT4 could only accommodate one scintillation material in any given application. The new public release (5.0) no longer suffers from this limitation.

in ExptPhysicsList: *(new G4 release 5.0)*

```
#include "G4Scintillation.hh"

G4Scintillation* theScintProcess = new G4Scintillation("Scintillation");

theScintProcess -> SetTrackSecondariesFirst(true);9
theScintProcess -> SetScintillationYieldFactor(1.0);
```

(The 'YieldFactor' allows for different scintillation yields depending on the particle type – in such case, separate scintillation processes must be attached to the various particles.)

```
#include “G4Material.hh (next release)

// Liquid Xenon

G4Element* elementXe = new G4Element(“Xenon”,”Xe”,54.,131.29*g/mole);

G4Material* LXe = new G4Material (“LXe”,3.02*g/cm3,1,kStateLiquid,
                                                    173.15*kelvin,1.5*atmosphere);
LXe -> AddElement(elementXe, 1);

const G4int NUMENTRIES = 9;

G4double LXe_PP[NUMENTRIES] = {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};
G4double LXe_SCINT[NUMENTRIES] = {0.000134, 0.004432, 0.053991,
                    0.241971, 0.398942, 0.000134, 0.004432, 0.053991,0.241971};

G4double LXe_RIND[NUMENTRIES] = { 1.57, 1.57, 1.57, 1.57, 1.57, 1.57,
                                                    1.57, 1.57, 1.57};

G4double LXe_ABSL[NUMENTRIES] = { 35.*cm, 35.*cm, 35.*cm, 35.*cm,
                                    35.*cm, 35.*cm, 35.*cm, 35.*cm, 35*cm };

G4MaterialPropertiesTable* LXe_MPT = new G4MaterialPropertiesTable();

LXe_MPT -> AddProperty(“FASTCOMPONENT”,LXe_PP,
                                                    LXe_SCINT,NUMENTRIES);
LXe_MPT -> AddProperty(“RINDEX”, LXe_PP,LXe_RIND,NUMENTRIES);
LXe_MPT -> AddProperty(“ABSLENGTH”,LXe_PP,
                                                    LXe_ABSL,NUMENTRIES);
LXe_MPT -> AddConstProperty (“SCINTILLATIONYIELD”,
                                                    100./MeV);
LXe_MPT -> AddConstProperty(“RESOLUTIONSCALE”,1.0)
LXe_MPT -> AddConstProperty(“FASTTIMECONSTANT”,45.*ns);
LXe_MPT -> AddConstProperty(“YIELDRATIO”,1.0);
LXe -> SetMaterialPropertiesTable(LXe_MPT);
```

G4Absorption

- Bulk Absorption - ‘kills’ the particle

MPT->AddProperty(“ABSLENGTH”,ppckov,abslength,NUMENTRIES};

Rayleigh Scattering

- The cross section is proportional to cos2(α), where α is the angle between the initial and final photon polarization.
- The scattered photon direction is perpendicular to the new photon’s polarization in such a way that the final direction, initial and final polarization are all in one plane.
- Rayleigh scattering attenuation coefficient is calculated for water following the Einstein-Smoluchowski formula, but in all other cases it must be provided by the user:

MPT -> AddProperty(“RAYLEIGH”,ppckov,scattering,NUMENTRIES};

Boundary Process

- Dielectric - Dielectric

  Depending on the photon's wave length, angle of incidence, (linear) polarization, and refractive index on both sides of the boundary:

  (a) total internal reflected

  (b) Fresnel refracted

  (c) Fresnel reflected

- Dielectric - Metal

  (a) absorbed (detected)

  (b) reflected

G4BoundaryProcess Implementation Details

- A ‘discrete process’, called at the end of every step
- never limits the step (done by the transportation)
- sets the ‘forced’ condition.
- Logic such that:
- preStepPoint: is still in the old volume
  postStepPoint: is already in the new volume

  so information is available from both

Surface Concept

Split into two classes

- Conceptual class: G4LogicalSurface (in the geometry category) holds:

  (i) pointers to the relevant physical or logical volumes

  (ii) pointer to a G4OpticalSurface

  These classes are stored in a table and can be retrieved by specifying:

  (i) an **ordered** pair of physical volumes touching at the surface [G4LogicalBorderSurface] - in principle allows for different properties depending on which direction the photon arrives.

  (ii) a logical volume entirely surrounded by this surface [G4LogicalSkinSurface] - *useful when the volume is coded by a reflector and placed into many volumes* (*limitation: only one and the same optical property for all the enclosed volume's sides*).

  (b) Physical class: G4OpticalSurface (in the material category) keeps information about the physical properties of the surface itself.

G4OpticalSurface

- Set the simulation model used by the boundary process:

  enum G4OpticalSurfaceModel {glisur, unfied};

  GLISUR-Model: original G3 model

  UNIFIED-Model: adopted from *DETECT* (TRIUMF)

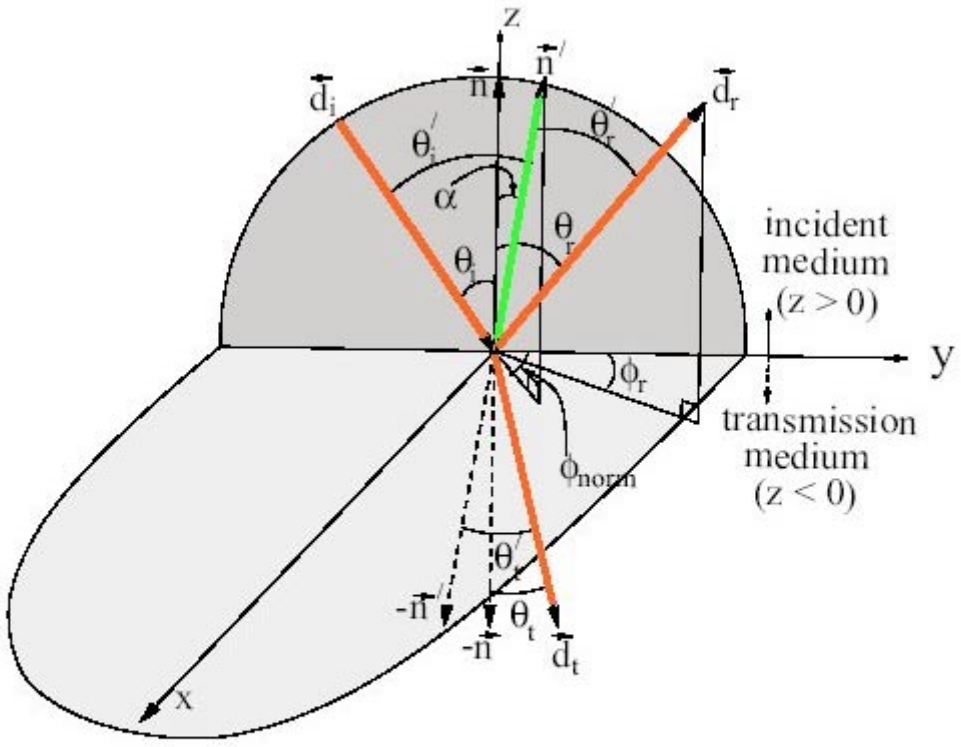
- Set the type of interface:

  enum G4OpticalSurfaceType { dielectric_metal, dielectric_dielectric};

- Set the surface finish:

  enum G4OpticalSurfaceFinish {
  polished, // smooth perfectly polished surface
  polishedfrontpainted, // polished top-layer paint
  polishedbackpainted, // polished (back) paint/foil
  ground, // rough surface
  groundfrontpainted, // rough top-layer paint
  groundbackpainted }; // rough (back) paint/foil

The assumption is that a rough surface is a collection of 'micro-facets'



Coordinate system used for ***ground*** surfaces along with the definition of geometrical parameters.

Surface effects

POLISHED: In the case where the surface between two bodies is perfectly polished, the normal used by the G4BoundaryProcess is the normal to the surface defined by:

(a) the daughter solid entered; or else
(b) the solid being left behind

GROUND: The incidence of a photon upon a rough surface requires choosing the angle, α, between a 'micro-facet' normal and that of the average surface.

The UNIFIED model assumes that the probability of micro-facet normals populates the annulus of solid angle sin(α)dα will be proportional to a gaussian of SigmaAlpha:

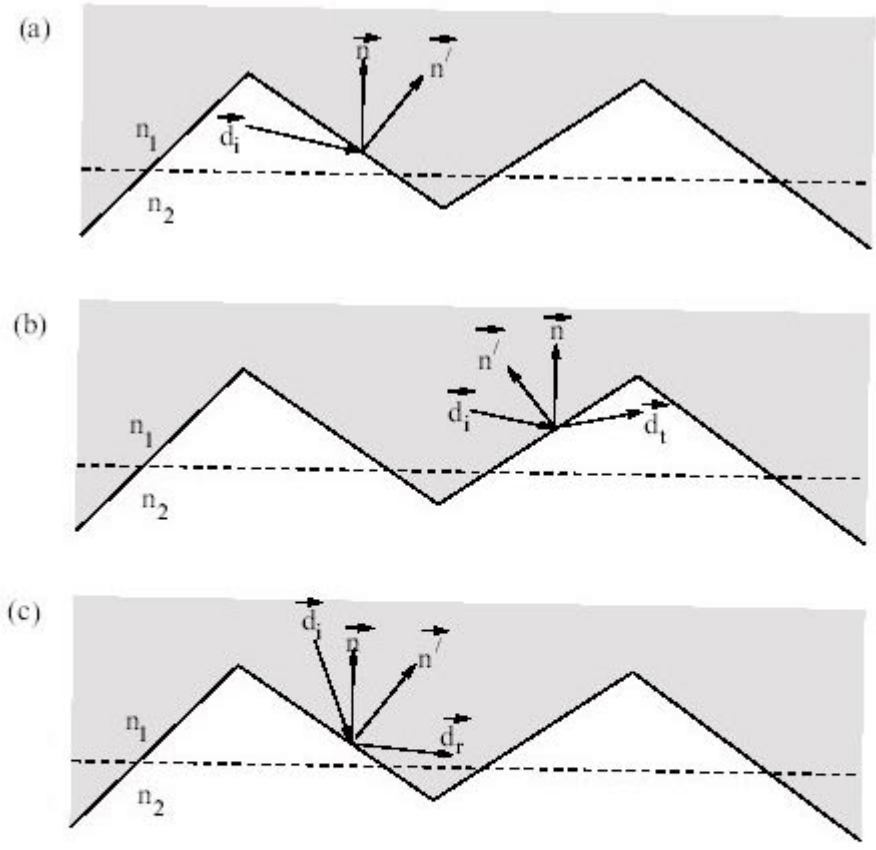
theOpSurface -> SetSigmaAlpha(0.1);

where sigma_alpha is in [rad]

In the GLISUR model this is indicated by the value of polish; when it is <1, then a random point is generated in a sphere of radius (1-polish), and the corresponding vector is added to the normal. The value 0 means maximum roughness with effective plane of reflection distributed as cos(α).

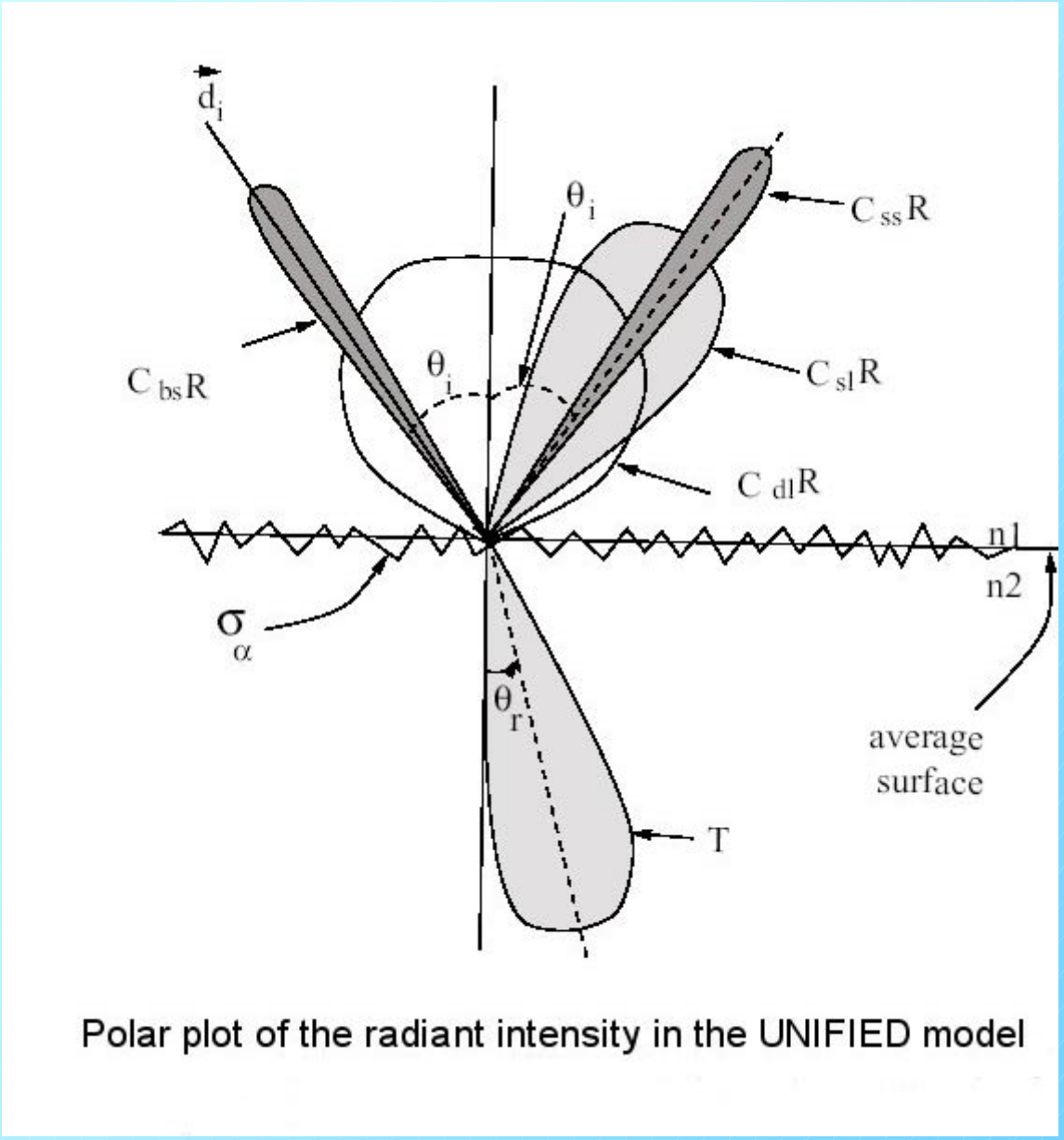
theOpSurface -> SetPolish(0.0);

The 'facet normal' is accepted if the refracted wave is still inside the original volume.



Special cases handled by the UNIFIED model:
(a) when the incident photon does not aim toward the local micro-facet; or when the transmitted (b) or reflected (c) photon heads in the wrong direction with respect to the average surface normal.

In cases (b) and (c), multiple interactions with the boundary are possible within the Process itself and without the need for relocation by the G4Navigator.



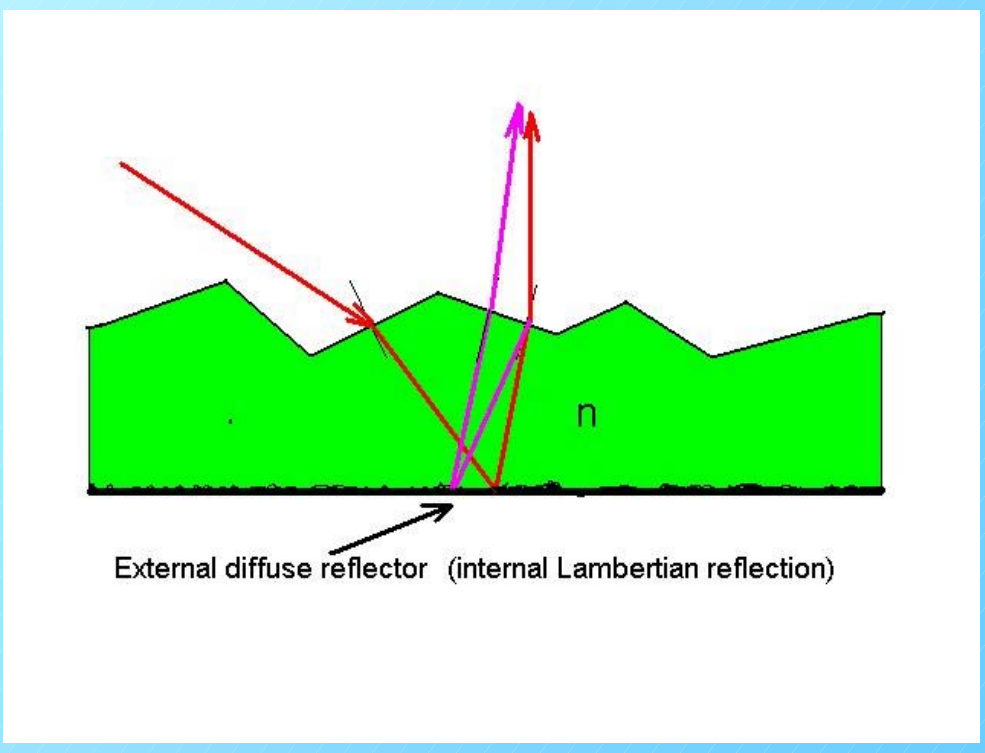
Polar plot of the radiant intensity in the UNIFIED model

Csl: Reflection prob. about the normal of a micro facet
Css: Reflection prob. about the average surface normal
Cdl: Prob. of internal Lambertian reflection
Cbs: Prob. of reflection within a deep grove with the ultimate result of exact back scattering.

The G4OpticalSurface also has a pointer to a G4MaterialPropertiesTable

In case the surface is painted, wrapped, or has a cladding, the table may include the thin layer's index of refraction.

This allows the simulation of boundary effects both at the intersection between the medium and the surface layer, as well as the far side of the thin layer, all within the process itself and without invoking the G4Navigator; the thin layer does not have to be defined as a G4 tracking volume.



Example

```
G4LogicalVolume * volume_log;
G4VPhysicalVolume * volume1;
G4VPhysicalVolume * volume2;

// Surfaces

G4OpticalSurface* OpWaterSurface = new G4OpticalSurface("WaterSurface");

OpWaterSurface -> SetModel(glisur);
OpWaterSurface -> SetType(dielectric_metal);
OpWaterSurface -> SetFinish(polished);

G4LogicalBorderSurface* WaterSurface = new
   G4LogicalBorderSurface("WaterSurface",volume1,volume2,OpWaterSurface);

G4OpticalSurface * OpAirSurface = new G4OpticalSurface("AirSurface");

OpAirSurface -> SetModel(unified);
OpAirSurface -> SetType(dielectric_dielectric);
OpAirSurface -> SetFinish(ground);

G4LogicalSkinSurface * AirSurface = new G4LogicalSkinSurface("AirSurface",
volume_log,OpAirSurface);
```

```
G4OpticalSurface * OpWaterSurface = new G4OpticalSurface(“WaterSurface”);

OpWaterSurface -> SetModel(unified);
OpWaterSurface -> SetType(dielectric_dielectric);
OpWaterSurface -> SetFinish(groundbackpainted);

Const G4int NUM = 2;

G4double pp[NUM] = {2.038*eV, 4.144*eV};

G4double specularlobe[NUM] = {0.3, 0.3};
G4double specularspike[NUM] = {0.2, 0.2};
G4double backscatter[NUM] = {0.1, 0.1};

G4double rindex[NUM] = {1.35, 1.40};

G4double reflectivity[NUM] = {0.3, 0.5};
G4double efficiency[NUM] = {0.8, 1.0};

G4MaterialPropertiesTable *SMPT = new G4MaterialPropertiesTable();

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

OpWaterSurface -> SetMaterialPropertiesTable(SMPT);
```

The logic in G4OpBoundaryProcess:PostStepDoIt is as follows:

(1) Make sure:
(a) the photon is at a boundary (StepStatus = fGeomBoundary)
(b) the last step taken is not a very short step (StepLength >=kCarTolerance/2) as it can happen upon reflection
ELSE do nothing and RETURN

(2) If the two media on either side are identical do nothing and RETURN

(3) If the original medium had no G4MaterialPropertiesTable defined kill the photon and RETURN
ELSE get the refractive index

(4) Get the refractive index for the medium on the other side of the boundary, if there is one.

(5) See, if a G4LogicalSurface is defined between the two volumes, if so get the G4OpticalSurface which contains physical surface parameters.

(6) Default to *glisur* model and *polished* surface

(7) If the new medium had a refractive index, set the surface type to 'dielectric-dielectric'
ELSEIF get the refractive index from the G4OpticalSurface
ELSE kill the photon.

(8) Use (as far as it has the information) G4OpticalSurface to model the surface ELSE use Default.

As a consequence:

(1) For polished interfaces between two media, no ‘surface’ needs to be specified. All that is required is that the two media have an index of refraction stored in their respective G4MaterialPropertiesTable.

(2) The boundary process implementation is rigid about what it expects the G4Navigator does upon reflection on a boundary.

(3) G4BoundaryProcess with ‘surfaces’ is only possible for volumes that have been positioned by using placement rather than replica or parameterised volumes .

ExampleN06