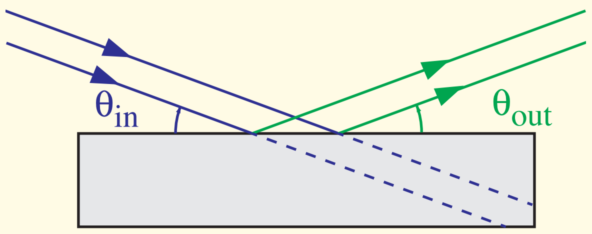
X-Ray reflection process

Ideas as presented in the 59th Geant4 Technical Forum on April 6

implementation strategy further discussed with

Vladimir Ivantchenko, Mihaly Novak and Gerardo Ganis



basic implementation concepts and naming for new process

discussed and agreed in the EM physics meeting in June

electromagnetic/xrays/src/G4XrayReflection.cc, hh G4EmProcessSubType fGammaReflection = 26

extend TestEm16 (Synchr.Rad.) to test X-Ray reflection

GetMeanFreePath calling Reflectivity routine providing value between 0 and 1 to reflect or just continue

PostStepDoIt reflect on surface

Acknowledgement, advice on modeling

Roberto Kersevan, Marton Ady / CERN and G4 collaborators Daren Sawkey, Giovanni Santin

Strong motivation : FCC-ee studies, with in particular FCC-MDI Manuela Boscolo et al.

Reference

B.L. **Henke**, E.M. Gullikson, and J.C. Davis. *X-ray interactions: photoabsorption, scattering, transmission, and reflection at E=50-30000 eV, Z=1-92*, Atomic Data and Nuclear Data Tables Vol. 54 (no.2), 181-342 (July 1993) used here ; appears to be the acccepted standard, see X-ray data booklet *X-Ray Interactions With Matter*, by the LBNL based CXRO Centre for X-Ray-Optics

plenty of other related literature, for example :

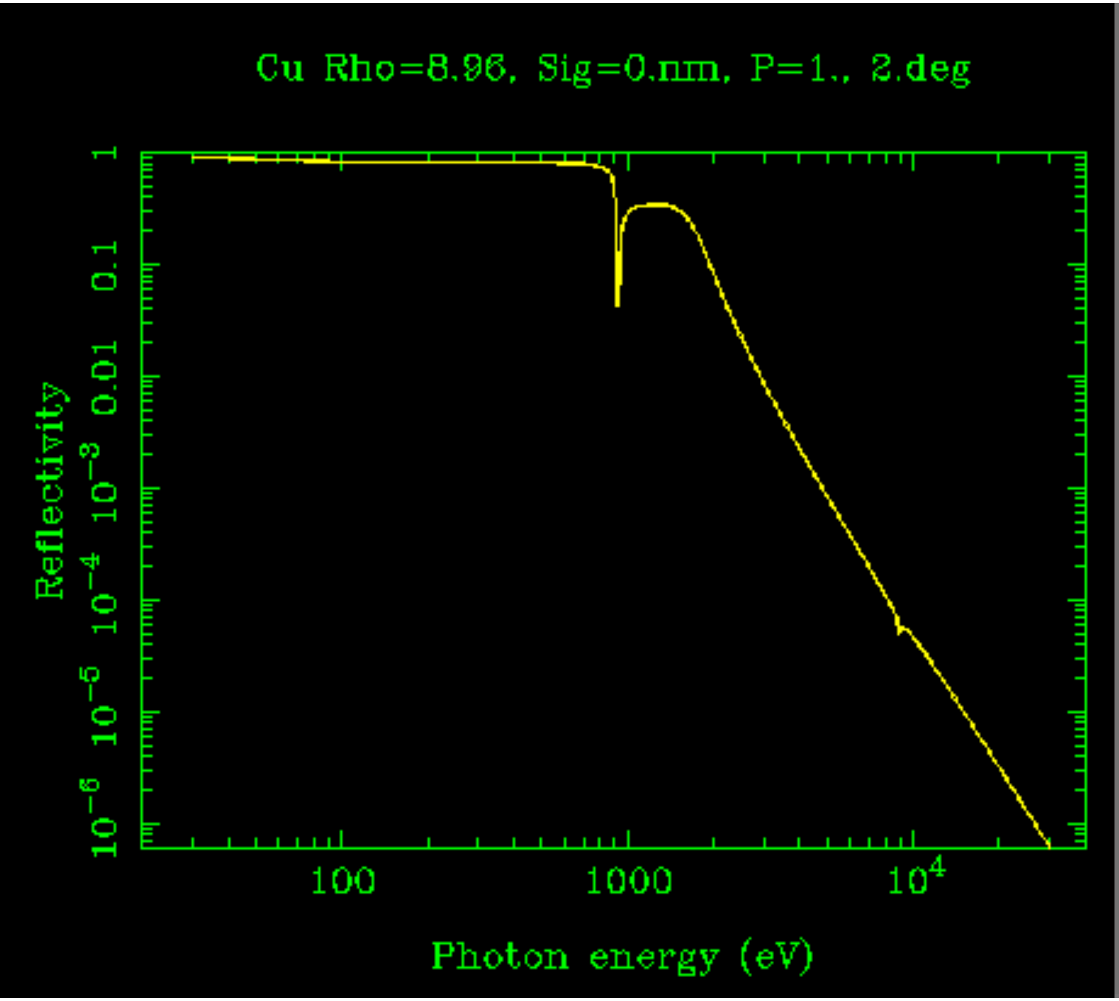
H. Kissing, *Untersuchungen zur Totalreflexion von Röntgenstrahlen*, Annalen der Physik (1931)
L.G.Parratt, *Surface studies of solids by total reflection of X-rays*, Phys. Rev., 95:359–369, (1954)
D.H. Bilderback and S. Hubbard, *X-ray mirror reflectivities from 3.8 to 50 keV*, NIM, NIM (1982)
Batterman and Bilderback in *Handbook on Synchrotron Radiation Vol.3* North Holland (1991) p. 120-124
E.-J. Buis and Giuseppe Vacanti, *X-ray tracing using Geant4*, NIMA 599 (2009)
G. Dugan and D. Sagan. *Simulating SR in Acc including diffuse and specular reflections*, PRACC (2017)
Penkov, Kopylets, Khadem, Qin. *X-Ray Calc*, SoftwareX, 12:100528 (2020)
E. La Francesca et al. *Reflectivity and photoelectron yield from copper in accelerators* PRACC (2020)

Roughness:

L. Névot, L. and P. Croce, *Caracterisation des surfaces par reflexion rasante de rayons X*. Rev. Phys. Appl. (1980)
S. K. Sinha, E. B. Sirota, S. Garoff, H. B. Stanley. *X-ray and neutron scattering from rough surfaces*. PRB (1988)
I. Feranchuk, AA Minkevich, Alex Ulyanenkov. *..Debye-Waller factor at large scattering vectors*, EuPhysJAppPhys (2003)
M. Wen, I. V. Kozhevnikov Z. Wang. *Reflection of X-rays from a rough surface ….* Opt. Express *(2015)*
Y. Fujii. *Analysis of surface roughness correlation function by X-ray reflectivity* 48(11):1136–1138 (2016)
Yuka Esashi et al., I*nfluence of surface and interface roughness .. comparison ..*, OSA Continuum (2021)

Calculator and data linked to the X-ray data booklet Xray interactions with Matter

https://henke.lbl.gov/optical_constants/mirror2.html



Mirror Reflectivity

- Choose from a list of common material: Enter Formula
- Chemical Formula: Si
- Density: -1 gm/cm^3 (enter negative value to use tabulated densities.)
- RMS Roughness: 0 nm.
- Polarization: 1 (-1 < pol < 1) where s=1, p=-1 and unpolarized=0.
- Scan Photon Energy (eV) from 30 to 30000 in 100 steps (< 500).
  (NOTE: Energies must be in the range 30 eV < E < 30,000 eV, Wavelength between 0.041 nm < Wavelength < 41 nm, and Angles between 0 & 90 degrees.)
- At fixed Angle (deg) = 1

To request a Linear Plot press this button: Submit Request

To reset to default values, press this button: Reset.

Explanation of Tables

Reflectivity
: Calculated using the Fresnel equations for a semi-infinite medium.

Material
: The chemical formula is required here. Note that this is case sensitive (e.g. CO for Carbon Monoxide vs Co for Cobalt).

Density
: If a negative value is entered, the chemical formula is checked against a list of some common materials. If no match is found then the density of the first element in the formula is used.

Grazing Angle
: In keeping with the standard notation for the x-ray region the incidence angle is measured relative to the surface (NOT the surface normal).

Roughness
: The effect of roughness is included in the approximation given by the Nevot-Croce factor.

Polarization
: Pol = 1 corresponds to s-polarization (electric field perpendicular to the plane of incidence). Pol=-1 corresponds to p-polarization (electric field in the plane of incidence). Pol=0 for unpolarized radiation.

Output
: A GIF plot may be generated for quick viewing of the results. If you need anything fancier, the results are provided as a text file for use with your favorite plotting package.

data : henke_physical_reference

folder with README

and data for 92 elements

ac.nff	Actinium
ag.nff	Silver
…	
cs.nff	Caesium
cu.nff	Copper
…	
zn.nff	Zinc
zr.ff	Zirconium

Example copper cu.nff 505 lines

E(eV)	f1	f2
10.0000	-9999.	1.30088
10.1617	-9999.	1.33374
...		
28.8337	-9999.	4.62195
29.3000	1.55250	4.70800
29.7739	1.65095	4.79565
30.2555	1.75782	4.88494
...		
29052.6	29.2669	0.502455
29522.5	29.2617	0.487378
30000.0	29.2564	0.472735

formatted text files, total 2.1 MByte or small compared to G4DATA, already 2 GByte

Reflection model

[Henke], p 193 C. *Non-Bragg Fresnel Reflection at Small Angles for the Semi-Infinite Solid*
1. *Reflection from an Ideally Smooth Surface*

$$\delta = \frac{N r_e \lambda^2}{2\pi} f_1$$

$$\beta = \frac{N r_e \lambda^2}{2\pi} f_2$$

λ wavelength r_e classical electron radius

f_1, f_2 Henke element data

N number of atoms per unit volume

$n = 1 - \delta - i\beta$

complex refraction index

not directly used here

$$\rho^2 = \frac{1}{2}\left[\sin^2\theta - 2\delta + \sqrt{(\sin^2\theta - 2\delta)^2 + 4\beta^2}\right]$$

θ angle to surface

$$R_\sigma = \frac{\rho^2(\sin\theta - \rho)^2 + \beta^2}{\rho^2(\sin\theta + \rho)^2 + \beta^2}$$

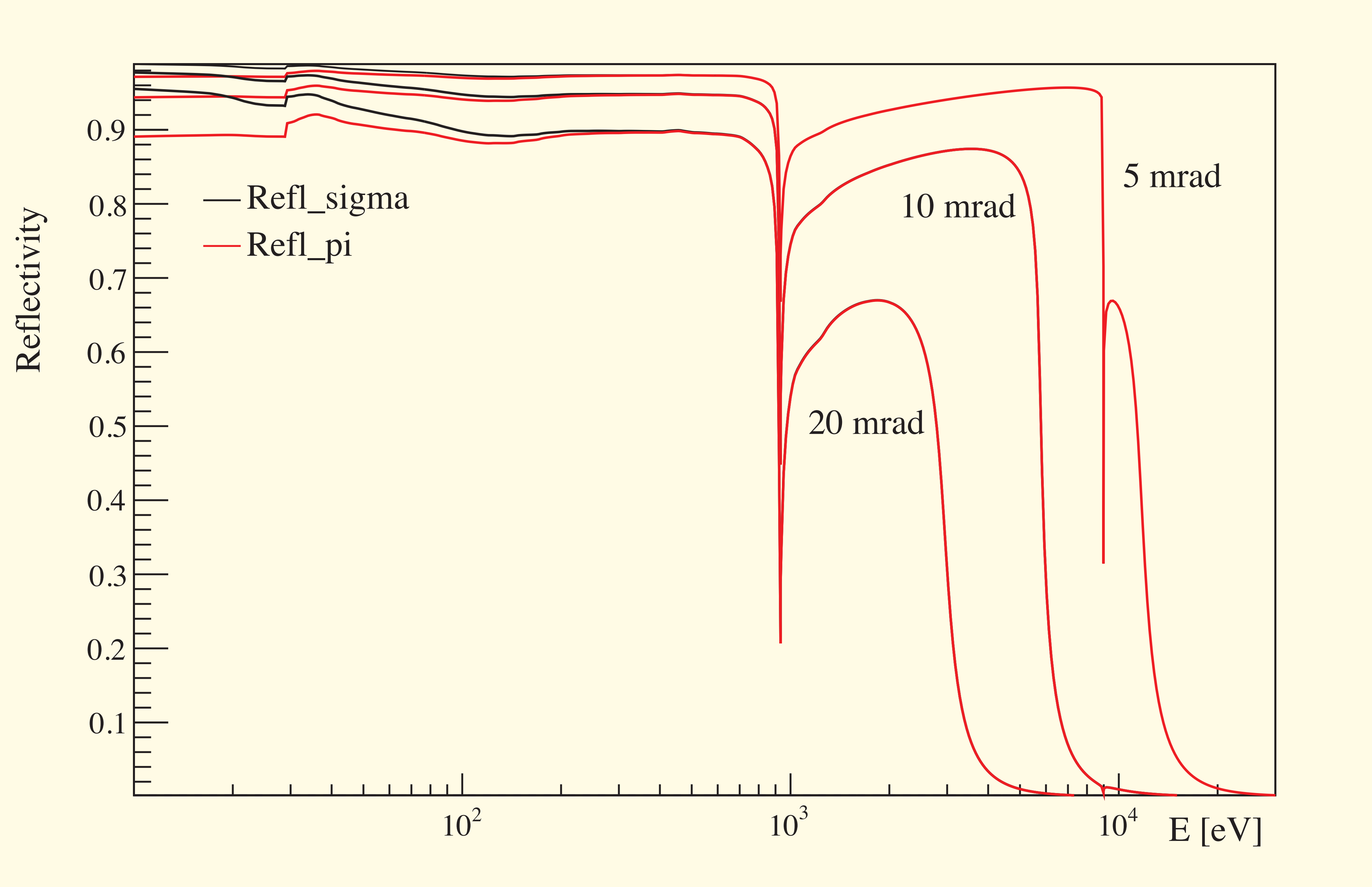
$$R_\pi = R_\sigma \frac{\rho^2(\rho - \cos\theta\cot\theta)^2 + \beta^2}{\rho^2(\rho + \cos\theta\cot\theta)^2 + \beta^2}$$

X-ray reflectivity for σ and π polarization

unpolarized $R = (R_\sigma + R_\pi) / 2$

$\exp(-2k_{i,z}\, k_{j,z}\, \sigma^2)$ roughness σ if > 0 apply additional simple Névot-Croce factor attenuation

Using data + model, example Cu



implementation, G4XrayReflection.cc

goals : simple, efficient and flexible use existing data structures when possible
after DetectorConstruction::ConstructVolumes() with gdml geometry+media loading
when all media defined, call

```
void G4XrayReflection::SaveHenkeDataAsMaterialProperty()
{ // loop through the material table and load set up MaterialPropertiesTable
  // with Henke data used to calculate the reflection
  auto materialTable = G4Material::GetMaterialTable();
  for (auto matItr = materialTable->begin(); matItr != materialTable->end(); ++matItr) {
    auto N = (*matItr)->GetTotNbOfAtomsPerVolume(); // N number of atoms per unit volume
    if ((*matItr)->GetNumberOfElements() == 1 && (*matItr)->GetDensity() > 1)
    ..      // for the moment do only for medium of single element
    ReadHenkeXrayData(theElement->GetName(), Ephot, f1, f2);
    ..
    calculate   δ, β  as function of  λ    ~ 1/Ephot
    ..      save δ, β  in the existing MaterialPropertiesTable structure
    auto property = new G4MaterialPropertiesTable();
    property->AddProperty("REALRINDEX", Ephot, RealIndex);
    property->AddProperty("IMAGINARYRINDEX", Ephot, ImagIndex);
    (*matItr)->SetMaterialPropertiesTable(property);
```

in `G4XrayReflection::GetMeanFreePath`

at entry to dense medium call

Reflectivity(GamEner, SinIncidentAngle, theMat)

G4XrayReflection::Reflectivity

```
G4double G4XrayReflection::Reflectivity(const G4double GamEner, const G4double SinIncidentAngle,
                                        const G4Material* theMat) const
{ G4double theReflectivity = 0;
  const G4MaterialPropertiesTable* theMatProp = theMat->GetMaterialPropertiesTable();
  if (SinIncidentAngle < 0.9 && theMatProp)
  {  // avoid perpendicular refl. at straight entry  and require
    // data available
    G4MaterialPropertyVector* RealIndex = theMatProp->GetProperty("REALRINDEX");
    G4MaterialPropertyVector* ImagIndex = theMatProp->GetProperty("IMAGINARYRINDEX");
    const G4double delta = RealIndex->Value(GamEner);
    const G4double beta = ImagIndex->Value(GamEner);
    const G4double sin2 = std::pow(SinIncidentAngle, 2);
    const G4double rho2 =
    0.5 * (sin2 - 2 * delta + std::sqrt(std::pow(sin2 - 2 * delta, 2) + 4 * beta * beta));
    const G4double rho = std::sqrt(rho2);
    const G4double Refl_sigma =
    (rho2 * std::pow(SinIncidentAngle - rho, 2) + std::pow(beta, 2))
    / (rho2 * std::pow(SinIncidentAngle + rho, 2) + std::pow(beta, 2));
    const G4double coscot = std::sqrt(1 - sin2) / SinIncidentAngle;
    const G4double pi_over_sigma = (rho2 * std::pow(rho - coscot, 2) + std::pow(beta, 2))
    / (rho2 * std::pow(rho + coscot, 2) + std::pow(beta, 2));
    const G4double Refl_pi = Refl_sigma * pi_over_sigma;
    theReflectivity = 0.5 * (Refl_sigma + Refl_pi);  // unpolarized
    G4double RoughAtten = 1;
    if (fSurfaceRoughness > 0) {
      G4double kiz = SinIncidentAngle * GamEner / CLHEP::hbarc;
      G4double kjz = SinIncidentAngle * (1 - delta) * GamEner / CLHEP::hbarc;
      RoughAtten = G4Exp(-2 * kiz * kjz * fSurfaceRoughness * fSurfaceRoughness);  // Nevot-Croce
      theReflectivity *= RoughAtten;
    }
 }
  return theReflectivity;
}
```

δ, β from PropTable

ρ^2

Reflectivity calculation

simple surface roughness attenuation

TestEm16 extended to provide first example & test

geant4-dev/-/merge_requests/3932 submitted 20/09/2023

emutils-V11-01-15, xrays-V11-01-01, testem16-V11-01-01: new process XrayReflection

for the moment data TestEm16, copied to local directory on cmake level

with TestReflection.mac :

/testem/det/GeomFile **Box_1m_Cu.gdml** # load geometry to test reflection

/run/initialize

/testem/phys/SetXrayReflectionRoughness 5 nm # set optional surface roughness

/gun/particle gamma

/analysis/h1/set 1 100 0 10 keV # set up histogram

/gun/direction 0 -0.0174524064372835 0.999847695156391 # **1 degree angle**

/gun/position 0 0.00872620321864176 -0.499923847578196 m # to hit at origin

loop over photon energy

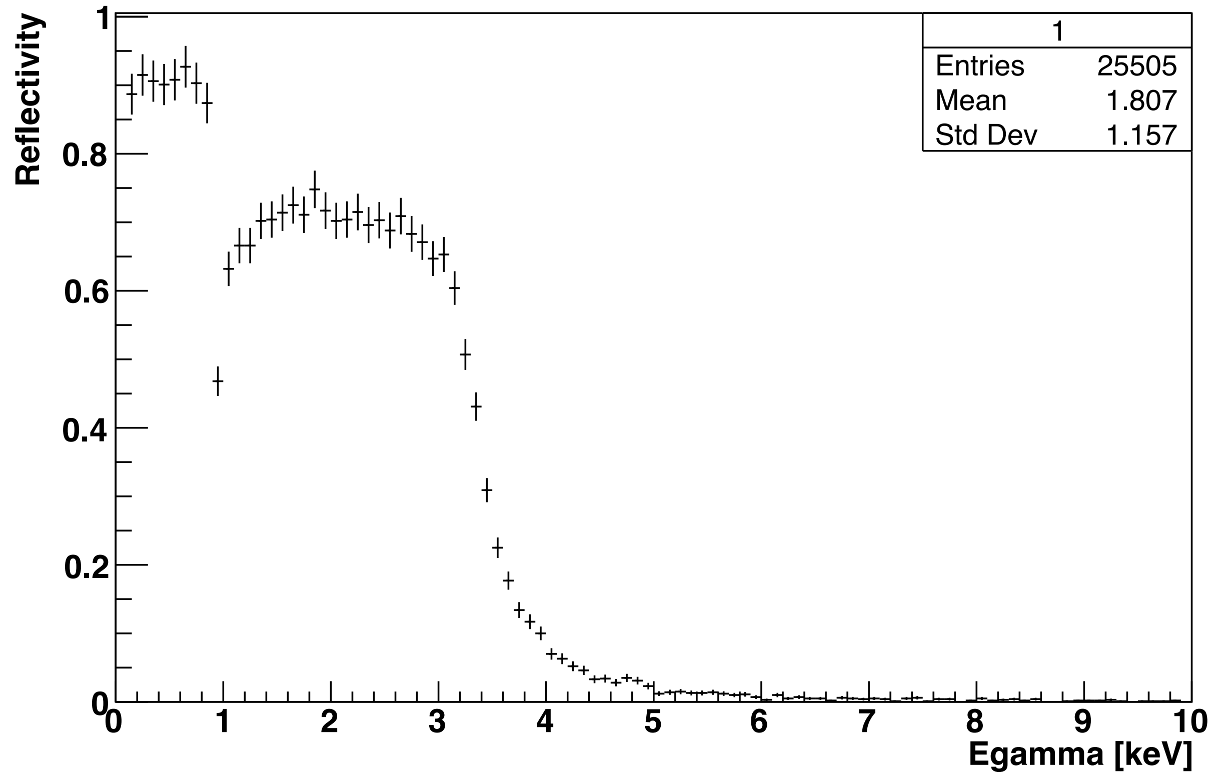
/gun/energy 0.05 keV

/run/beamOn 1000

/gun/energy 0.15 keV

/run/beamOn 1000

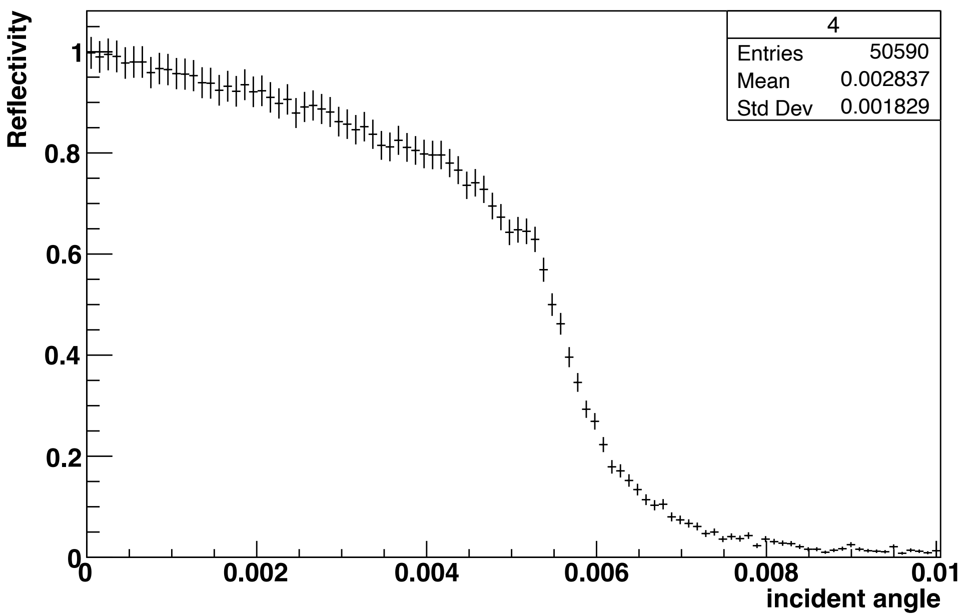
..

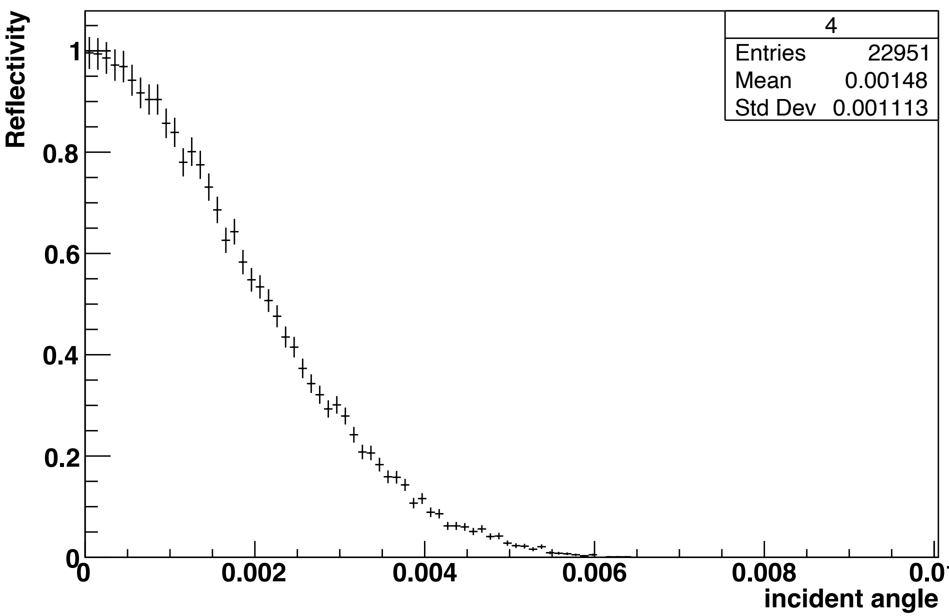


TestEm16, TestReflection scan in angle


```
/testem/det/GeomFile  Box_1m_Cu.gdml
/gun/energy 10 keV
/analysis/h1/set 4 100 0 0.01005     # histogram for angle scan
# angle=5e-05
/gun/position  0  2.49999999895833e-05 -0.499999999375 m
/gun/direction 0 -4.99999999791667e-05 0.99999999875
/run/beamOn 1000
# angle=0.00015
/gun/position  0  7.499999971875e-05 -0.499999994375 m
/gun/direction 0 -0.0001499999994375 0.99999998875
/run/beamOn 1000
```

/testem/phys/SetXrayReflectionRoughness 5 nm





Concluding remarks

A first implementation of the process of specular X-Ray reflection has just been submitted for merging into the G4 repository, for first tests and use, for the moment via TestEm16

The reflection model is based on the Henke et al. model with element data (10 eV to 30 keV) for specular reflection from low density (vacuum, air) onto solid surfaces

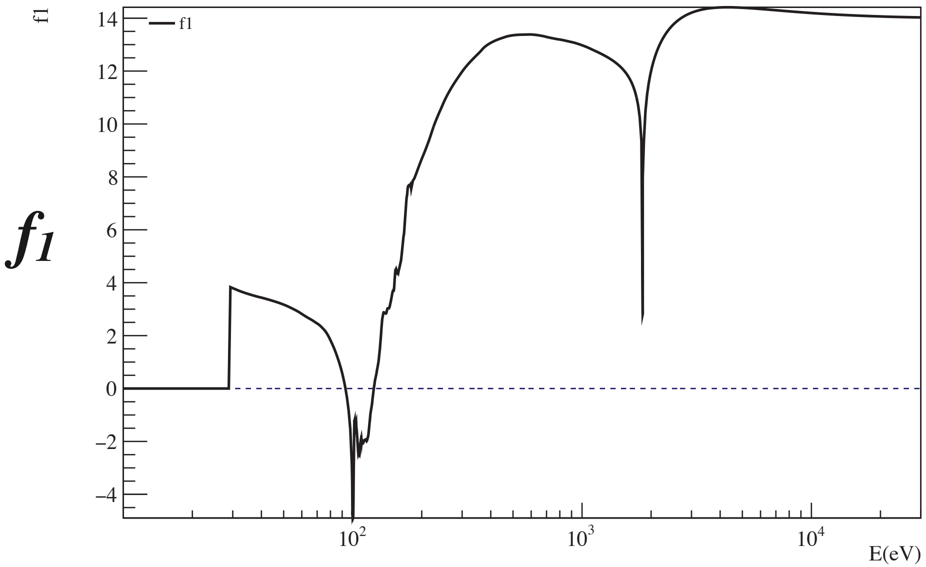
+ optional very simple exponential surface roughness attenuation

Written in a modular way, such that extension for specific cases like multi-layer surfaces should be rather straight-forward (provide a specific reflectivity function)

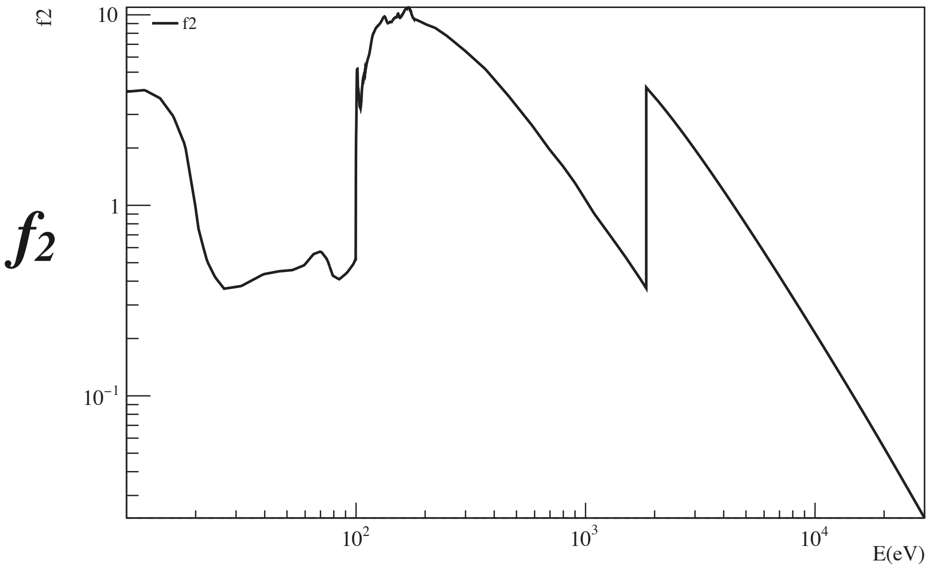
Backup

Henke f_1, f_2 , δ, β as function of Eγ example Si

f_1

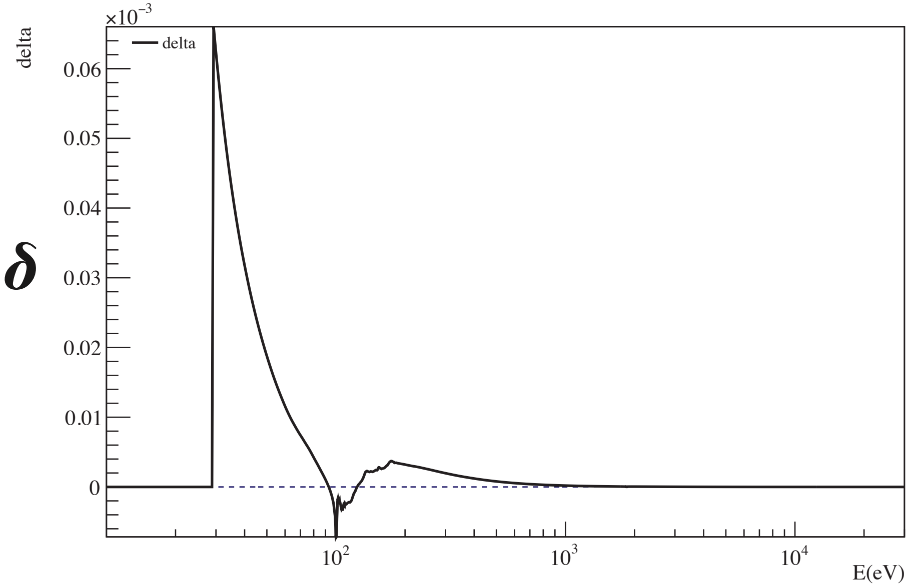


f_2

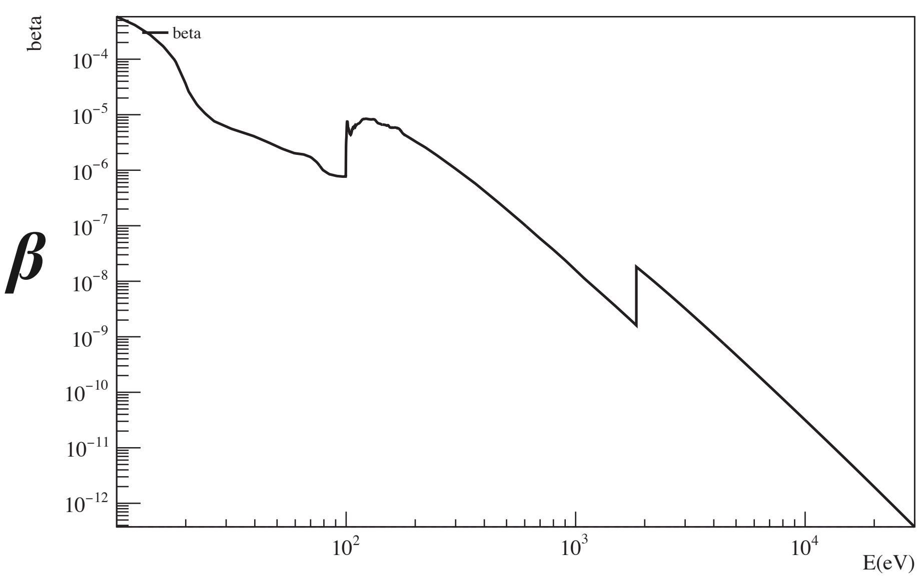


Henke physical reference Si f1 f2, f1 can go negative, f2 always positive

δ

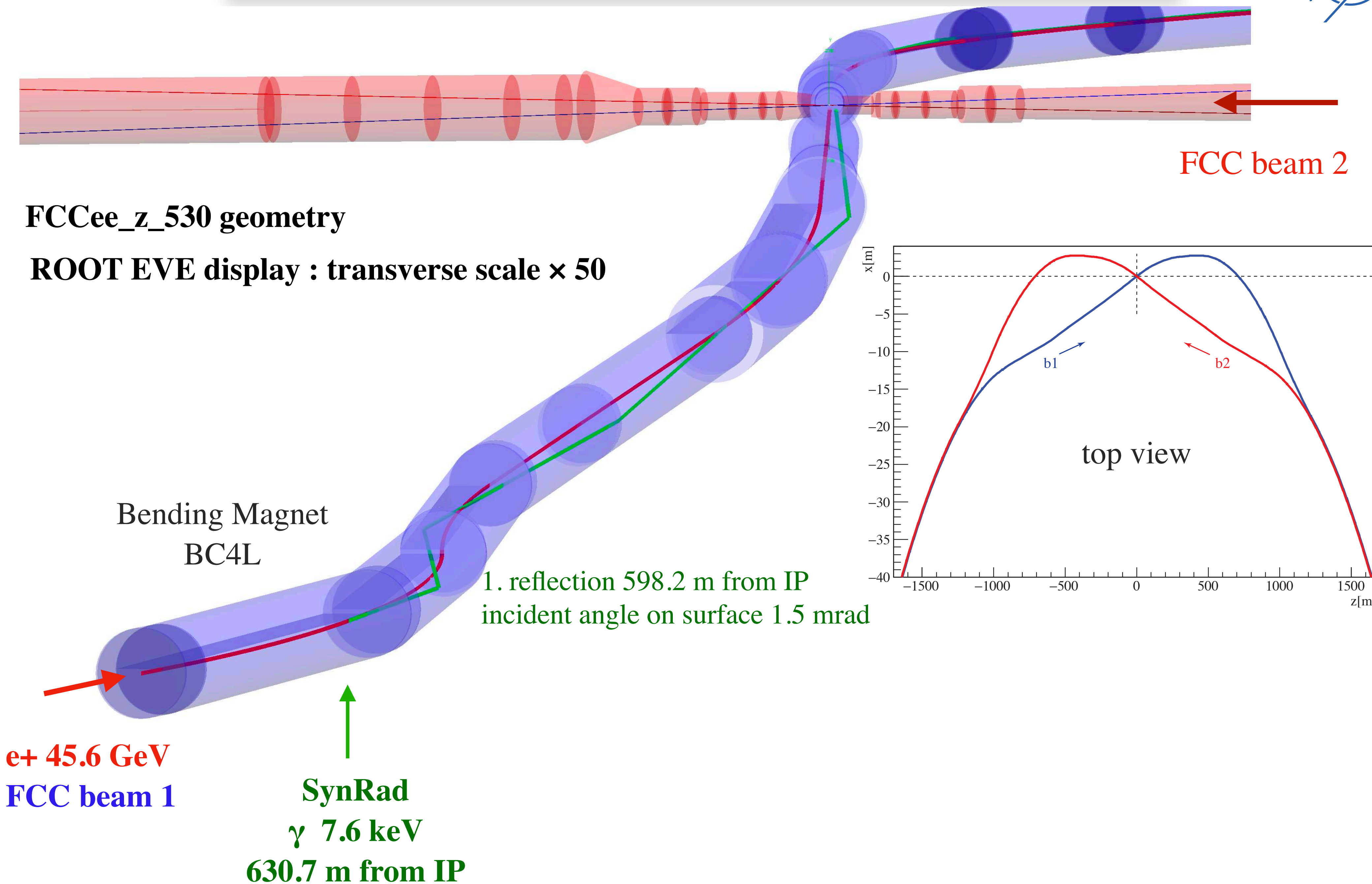


β



Henke physical reference Si refractive index real and imag parts

test of more complex geometry with multiple reflections



GetMeanFreePath

```
G4double G4XrayReflection::GetMeanFreePath(const G4Track& aTrack, G4double previousStepSize,
G4ForceCondition* condition) {
  *condition = NotForced;
  G4double GamEner = aTrack.GetDynamicParticle()->GetTotalEnergy();
  if (GamEner < 10. * eV || GamEner > 30. * keV)
    return DBL_MAX;  // do nothing below 10 eV and above 30 keV
 G4double MeanFreePath = DBL_MAX;  // by default no reflection
  G4VPhysicalVolume* Volume = aTrack.GetVolume();
  if (fLastVolume && Volume != fLastVolume && aTrack.GetTrackLength() > 0) {  // at a boundary
    const G4Material* theLastMat = fLastVolume->GetLogicalVolume()->GetMaterial();
    const G4Material* theMat = Volume->GetLogicalVolume()->GetMaterial();
    G4double last_density = theLastMat->GetDensity();
    G4double density = theMat->GetDensity();
    if (density > last_density) {  // density has increased
      G4Navigator* theNavigator =
        G4TransportationManager::GetTransportationManager()->GetNavigatorForTracking();
      G4bool valid = false;
      G4ThreeVector theSurfaceNormal =
        theNavigator->GetGlobalExitNormal(aTrack.GetPosition(), &valid);
      if (valid) fSurfaceNormal = theSurfaceNormal;
      G4double SinIncidentAngle =
        aTrack.GetDynamicParticle()->GetMomentumDirection() * fSurfaceNormal;
      if (G4UniformRand() < Reflectivity(GamEner, SinIncidentAngle, theMat)) MeanFreePath = 0;
    }
  }
  fLastVolume = Volume;
  return MeanFreePath;
}
```

PostStepDoIt

```
G4VParticleChange* G4XrayReflection::PostStepDoIt(const G4Track& aTrack, const G4Step& aStep)
{
  aParticleChange.Initialize(aTrack);  // copy the current position to the changed particle
  G4ThreeVector PhotDir = aTrack.GetDynamicParticle()->GetMomentumDirection();
  G4ThreeVector para_part = (PhotDir * fSurfaceNormal) * fSurfaceNormal;
  G4ThreeVector photon_reflected = PhotDir - 2 * para_part;  // invert the parallel component
 aParticleChange.ProposeTrackStatus(
    fStopAndKill);  // needed when working with primary gamma to get rid of
  // primary
  G4DynamicParticle* ReflectedPhoton = new G4DynamicParticle(
    G4Gamma::Gamma(), photon_reflected, aTrack.GetDynamicParticle()->GetTotalEnergy());
  aParticleChange.AddSecondary(ReflectedPhoton);
  return &aParticleChange;
}
```

ReadHenkeXrayData

```
G4int G4XrayReflection::ReadHenkeXrayData(std::string ElName, std::vector<G4double>& Ephot,
                                          std::vector<G4double>& f1, std::vector<G4double>& f2)
{
  std::transform(ElName.begin(), ElName.end(), ElName.begin(),
                 ::tolower);  // henke_physical_reference uses lower case filanames
  G4String XrayReflectionDataDir = "henke_physical_reference/";
  const std::string InpFname = XrayReflectionDataDir + ElName + ".nff";
  std::ifstream infile(InpFname);
  if (!infile.is_open()) {
    G4cout << "ReadHenkeXrayReflData " << InpFname << " not found" << G4endl;
    return 1;  // failure
  }
  std::vector<std::string> VarName(3);
  infile >> VarName[0] >> VarName[1] >> VarName[2];
  G4double E_eV_i, f1_i, f2_i;
  Ephot.resize(0);
  f1.resize(0);
  f2.resize(0);
  for (;;) {
    infile >> E_eV_i >> f1_i >> f2_i;
    if (infile.eof()) break;
    Ephot.push_back(E_eV_i * eV);
    f1.push_back(f1_i);
    f2.push_back(f2_i);
  }
  return 0;  // success
}
```