G4 come ray tracer nell'esperimento Borexino

Magni Stefano
I.N.F.N Milano

II Workshop I.N.F.N sul Software
e Calcolo Moderno
Perugia 15-17 Giugno 1998

Borexino e' un rivelatore in tempo reale ad alta radiopurezza per spettroscopia di neutrini di bassa energia (860 keV) costituito da:

- 300 ton di scintillatore organico liquido
- visto da 2200 PMT
- schermato da un liquido ad elevata radiopurezza

Tracciamento di fotoni ottici in geometrie complesse

- Attuale codice: FORTRAN stand alone
- Difficolta' di gestione della geometria

Necessita' di :

- Descrizione di geometrie complesse
- Ray tracing su superfici complesse
- Creazione o modifica di processi fisici in funzione delle esigenze dell'esperimento.

Geant4

- La definizione della geometria del rivelatore e' semplificata utilizzando "oggetti".
- Grazie alla tecnologia OO semplice possibilita' di definizione di un processo fisico "custom": assorbimento e riemissione di fotoni
- Tools di visualizzazione grafica dell'evento. Utili nella fase di descrizione geometrica.
- Tools (classi C++) di utilita' generale (random engines, vettori, matrici...)

Esempio: costruzione della geometria

Utilizzo di polimorfismi

```
G4Sphere* InnerVessel = new G4Sphere("InnerVessel", 0.0*cm, 105.05*cm,
        0.0*deg, 360.0*deg,0.0*deg, 180.0*deg);
G4LogicalVolume * InnerVessel_log
= new G4LogicalVolume(InnerVessel, Nylon, "InnerVessel_L", 0,0,0);
G4VPhysicalVolume * InnerVessel_phys
= new G4PVPlacement(0,G4ThreeVector(),"InnerVessel_P",
                    InnerVessel_log, Tank_phys, false, 0);
```

G4Sphere e' un G4VSolid, posso quindi ridefinire il mio oggetto InnerVessel, come un qualunque altro G4VSolid, ad esempio un G4Box :

```
G4Box * InnerVessel
    = new G4Box("InnerVessel", 50.0*cm, 50.0*cm, 50.0*cm );
```

Sfruttando il polimorfismo degli oggetti sono quindi riuscito a modificare una parte del mio rivelatore in modo completamente trasparente rispetto al funzionamento del resto del codice.

Esempio: definizione di un processo fisico

Include File

```
class G4OpAbsorption : public G4VDiscreteProcess
{
private:
        ///////////////
        // Operators
        ///////////////
        // G4OpAbsorption& operator=(const G4OpAbsorption &right);
public:
        ////////////////////////////////
        // Constructors and Destructor
        ////////////////////////////////
        G4OpAbsorption(const G4String& processName = "Absorption");
        // G4OpAbsorption(const G4OpAbsorption &right);
 ~G4OpAbsorption();
 ////////////
 // Methods
        ////////////
        G4bool IsApplicable(const G4ParticleDefinition& aParticleType);
 G4double GetMeanFreePath(const G4Track& aTrack,
    G4double previousStepSize,
    G4ForceCondition* condition);
 G4VParticleChange* PostStepDoIt(const G4Track& aTrack,
          const G4Step&  aStep);
       void DumpPhysicsTable() const;
};
```

Una volta definiti i metodi GetMeanFreePath e PostStepDoIt il nuovo "oggetto" processo puo' essere inserito nel package senza che si necessaria alcuna ulteriore conoscenza di G4

Implementazione

```
// PostStepDoIt
// -------------
//
G4VParticleChange*
G4OpAbsorption::PostStepDoIt(const G4Track& aTrack, const G4Step& aStep)
{
  aParticleChange.Initialize(aTrack);
 aParticleChange.SetStatusChange(fStopAndKill);
  if (verboseLevel>0) {
  cout << "\n** Photon Absorbed! **" << endl;
 }
 return G4VDiscreteProcess::PostStepDoIt(aTrack, aStep);
}
// GetMeanFreePath
// ---------------
//
G4double G4OpAbsorption::GetMeanFreePath(const G4Track& aTrack,
             G4double previousStepSize,
            G4ForceCondition* condition)
{
 const G4DynamicParticle* aParticle = aTrack.GetDynamicParticle();
  const G4Material* aMaterial = aTrack.GetMaterial();
 G4double thePhotonMomentum = aParticle->GetTotalMomentum();
 G4MaterialPropertiesTable* aMaterialPropertyTable;
 G4MaterialPropertyVector* AttenuationLengthVector;
  G4double AttenuationLength = DBL_MAX;
 aMaterialPropertyTable = aMaterial->GetMaterialPropertiesTable();
 if ( aMaterialPropertyTable ) {
  AttenuationLengthVector = aMaterialPropertyTable->GetProperty("ABSLENGTH");
  AttenuationLength = AttenuationLengthVector->GetProperty (thePhotonMomentum
  }
 else {
  cout << "No Absorbtion length specified" << endl;
  return AttenuationLength;
  }
 G4bool isOutRange;
 return AttenuationLength;
}
```

Utilizzare il C++ .....

- Divario sempre piu' vasto tra sviluppatore ed utilizzatore
- Alto livello di professionalita' richiesto allo sviluppatore (C++ piu' complesso del C o Fortran)
- Necessita' di fornire all'utilizzatore:

- Prodotto ben funzionante: l'utilizzatore deve lavorare con metodi e interfaccie e deve disinteressarsi delle implementazioni.
- Documentazione efficiente e completa. Descrizione classi ed interfaccie, esempi di utilizzo

...utilizzare il C++

- Individuazione dei Memory leak
- Gestione dei puntatori

Transizione all'OO

- Istituzione di corsi introduttivi all'OO ed ai linguaggi specifici
- Necessita' di tools che il fisico possa utilizzare nel proprio lavoro (cernlib). Ad oggi ROOT.