A Geant4 ray tracer for Borexino

L.N.G.S 30/10/98

Why a new tracking code ?

- ❋ Separate physics processes from geometry and tracking.
- ❋ Modularized code: easy to manage and modify
- ❋ Take advantage of an existing and supported code (CERN)
- ❋ A proposed solution for a unified tracking code in the collaboration

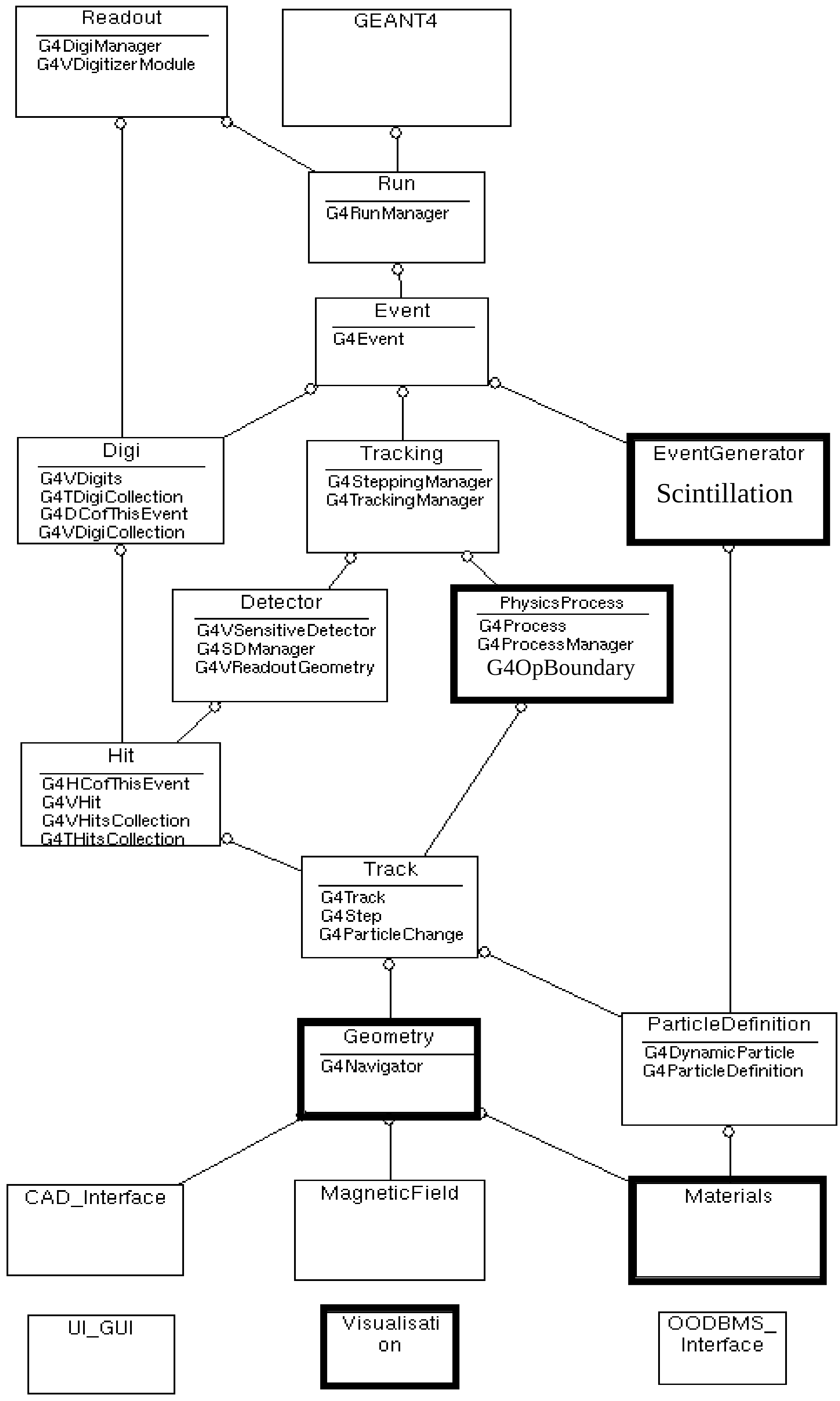
The Geant4 project

* The stated goal of the GEANT4 project is to produce the GEANT4 detector simulation toolkit exploiting Object-Oriented technology to improve the transparency of the physics implementation and hence the possibility of validating the physics results.
* A first prototype was delivered at the end of 1995, the first alpha version was released in spring 1997 and the first beta version was announced at mid-1998.
* The first production version is scheduled by the end of 1998. With the release of the first production version the project will have met its goals and will have been completed.
* The GEANT4 software has been developed via the collaboration world-wide of about 100 scientists, coming from over 40 institutes and experiments in Europe,

What can you do with Geant4 toolkit

- GEANT4 is a toolkit for both full and fast MonteCarlo simulation of detectors in High Energy Physics. It is designed to take into account also the requirements of space and cosmic ray applications, nuclear and radiation computations, heavy ion and medical applications.
- It allows the description of the geometry and material of the detector elements. The kinematics of the primary physics events is used to generate particles which are then tracked through the detector, simulating their physics interactions in matter and the effect of fields and boundaries on their trajectories.
- Finally, the electronic response of the sensitive detector elements can be reproduced.

Class overview



L.N.G.S 30/10/98

What is in your main program ?

❋ How your detector should be built.

❋ What kinds of physics processes you are interested in.

❋ How the primary particle(s) in an event should be produced.

❋ Additional demands during the procedures of simulation.

User actions :

- G4UserRunAction
- G4UserEventAction
- G4UserTrackingAction
- G4UserSteppingAction

Tracking of optical photons

- Optical boundary processes (reflection refraction)
- Intersection of rays in a geometry hierarchy: tracking
- User defined processes :
  - ✔ Absorption
  - ✔ Absorption and reemission
  - ✔ Light scattering

How to define a custom physical process

```
class G4OpAbsorption : public G4VDiscreteProcess  {
 public:
      ////////////////////////////////
      // Constructors and Destructor
      ////////////////////////////////

      G4OpAbsorption(const G4String& processName =
     "Absorption");
     ~G4OpAbsorption();

     ////////////
     // Methods
      ////////////

      G4bool IsApplicable (const G4ParticleDefinition&
     aParticleType);
     G4double GetMeanFreePath (const G4Track& aTrack,
                                        G4double previousStepSize,
                                        G4ForceCondition* condition);
     G4VParticleChange* PostStepDoIt(const G4Track& aTrack,
                                             const G4Step&  aStep); };
```

Comparison with current tracking code

- Arrival time distribution:
  - ✔ no difference
- Detected photons wavelength distribution:
  - ✔ no difference
- CPU execution time:
  - ✔ slower
- Much more flexible
- Some nice pictures