Experimental Techniques in Particle Physics

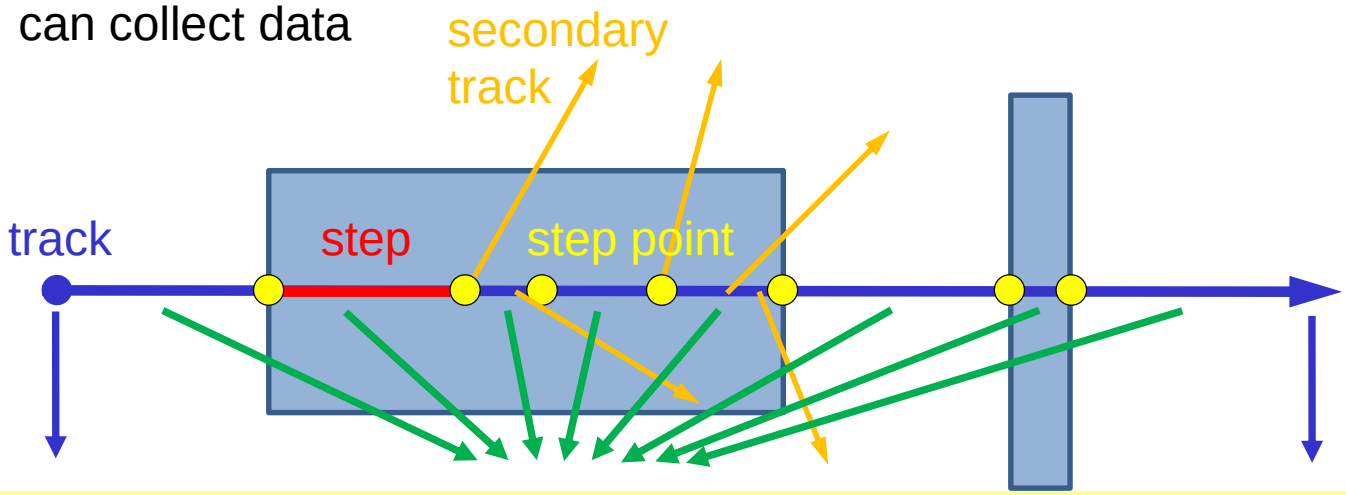
Geant4: Sensitive Detectors

Andreas Nowack
nowack@physik.rwth-aachen.de
RWTH Aachen University

WS 2020/21

Event Processing: Collecting Useful Information

- user classes
  - called during event processing
  - can collect data



PreTracking Action | Stepping Action | PostTracking Action

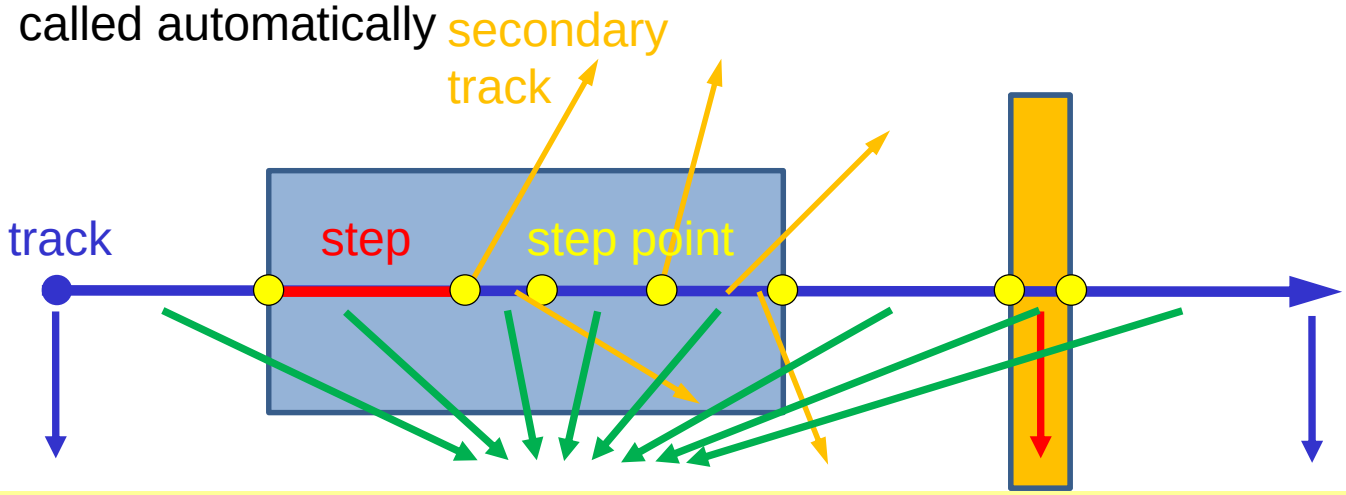
User Application

G4Track | G4Step | G4Track

Geant4

Event Processing: Sensitive Detector

- special user class: sensitive detector
  - attached to logical volume
  - called automatically



Defining a Sensitive Detector

- sensitive detector objects
  - created and assigned to logical volumes
    in user detector construction class in ConstructSDandField()
  - each detector must have a unique name
  - instances of the same class must be defined with different detector names

```
G4VSensitiveDetector* mySD = new MySD("MySD", "MyHitsCollection");
```

  - assigning to a logical volume via volume name or pointer

```
SetSensitiveDetector("MyLVName", mySD);
  or
SetSensitiveDetector(myLV, mySD);
```

  - multiple sensitive detectors can be assigned to the same logical volume using G4VUserDetectorConstruction::SetSensitiveDetector()
  - each sensitive detector has to be registered in G4SDManager

```
G4SDManager::GetSDMpointer()->AddNewDetector(mySD);
```

Sensitive Detector Class

- sensitive detector class MySD is derived from G4VSensitiveDetector
  - three user methods called during event processing:
    - at beginning of each event: Initialize(…)
    - in a step (if in the associated volume): ProcessHits(…)
      - 1st argument: G4Step object
        - see stepping action
      - 2nd argument: G4TouchableHistory object
        - Readout geometry (or NULL)
      - process hits and fill them into hit collection
    - at end of each event: EndOfEvent(…)

Sensitive Detector Class

- needs dedicated hit and hit collection classes
  derived from G4VHit and G4THitsCollection, respectively
  - various possibilities: position, time, momentum, energy,
    energy deposition, geometrical information, …

- processing of collected hits of all detectors in EndOfEventAction()
  - write (relevant) hits into file, analyse it later
  - analyse hits on the fly

Multi-functional Detector

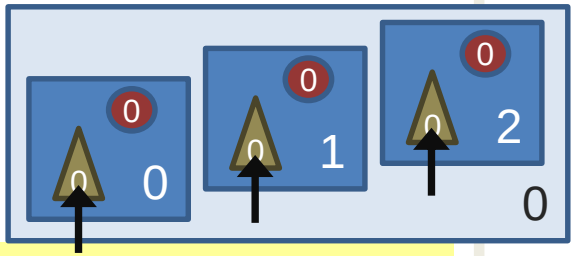
- very generic detector class available: G4MultiFunctionalDetector
- use scorer classes derived from G4VPrimitiveScorer in order to collect specific data
  - track length — G4PSTrackLength
  - passage track length — G4PSPassageTrackLength
  - energy deposition — G4PSEnergyDeposit
  - dose deposition — G4PSDoseDeposit
  - flat surface current — G4PSFlatSurfaceCurrent
  - sphere surface current — G4PSSphereSurfaceCurrent
  - passage current — G4PSPassageCurrent
  - flat surface flux — G4PSFlatSurfaceFlux
  - cell flux — G4PSCellFlux
  - passage cell flux — G4PSPassageCellFlux
  - minimum kinetic energy of secondary particles — G4PSMinKinEAtGeneration
  - number of secondary particles — G4PSNofSecondary
  - number of steps — G4PSNofStep
  - total charge of particles stopped — G4PSCellCharge

Multi-functional Detector

- scorer classes above produce one G4THitsMap<G4double> object per event
  - mapping copy number of the volume to measured quantity
  - parameter depth determines which copy number is used (useful for replicated structures)
    - depth = 0: copy number of the physical volume itself
    - depth = 1: copy number of the mother volume
    - depth = 2: copy number of the mother volume of the mother volume
    - …
- collected data can be filtered using classes derived from G4VSDfilter
  - all charged particles — G4SDChargedFilter
  - all neutral particles — G4SDNeutralFilter
  - particles of given species — G4SDParticleFilter
  - particles in a given kinetic energy range — G4SDKineticEnergyFilter
  - given species in a given energy range — G4SDParticleWithEnergyFilter

Multi-functional Detector: Example Detector Construction

- energy deposition of all particles
- number of secondary gammas produced in the volume
- track lengths of all electrons and positrons

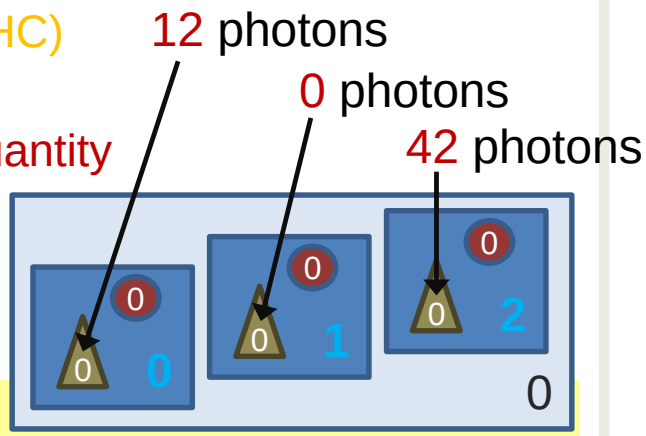



```
G4SDParticleFilter* gammaFilter =  new G4SDParticleFilter("gammaFilter", "gamma");
                                                  name of filter   name of particle
G4SDParticleFilter* epFilter = new G4SDParticleFilter("eeFilter");
epFilter->add("e-");
epFilter->add("e+");

G4MultiFunctionalDetector* det = new G4MultiFunctionalDetector("myDet");

det->RegisterPrimitive(new G4PSEnergyDeposit("eDep", 1));   depth: use copy number of mother volume

G4VPrimitiveScorer* primitive = new G4PSNofSecondary("nGamma", 1);
primitive->SetFilter(gammaFilter);
det->RegisterPrimitive(primitive);

primitive = new G4PSTrackLength("trackLength", 1);
primitive->SetFilter(epFilter);
det->RegisterPrimitive(primitive);

G4SDManager::GetSDMpointer()->AddNewDetector(det);
SetSensitiveDetector("detDougtherLogVolume", det);
```

in: DetectorPhysDetectorConstruction::ConstructSDandField()

Multi-functional Detector: Example Measurement

- end of event
  - all measured data collected in Hit Collections (HC)
    - identified by Hit Collection ID (HCID)
    - assignment: copy number → measured quantity
      - example (copy number of mother):
        - 0: 12
        - 2: 42

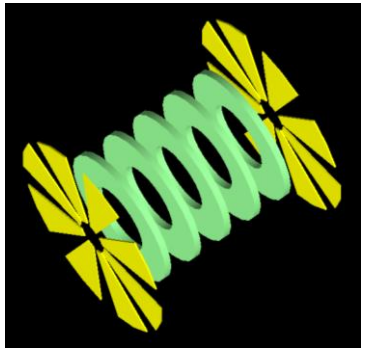



```
void DetectorPhysEventAction::EndOfEventAction(const G4Event* evt) {
  if ( fmyDetnGammaHCID == -1 ) {   // Get hit collection ID
    fmyDetnGammaHCID = G4SDManager::GetSDMpointer()->GetCollectionID("myDetnGamma");
  }
  G4THitsMap<G4double>* hitsMap =
    static_cast<G4THitsMap<G4double>*>(evt->GetHCofThisEvent()->GetHC(fmyDetnGammaHCID));

  G4cout << "---> End of event: " << evt->GetEventID() << G4endl;

  std::map<G4int, G4double*>::iterator it;
  for (it = hitsMap->GetMap()->begin(); it != hitsMap->GetMap()->end(); it++) {
    G4cout << “detector part “ it->first << ": " << *(it->second) << “secondary photon(s)” << G4endl;
  }
}
```

name of collection (detector + scorer)

Exercise: Particle Sources

1. Download DetectorPhys_T10.tar.gz and decompress it.
   - This code defines already a multi-functional detector that measures the dose in all “petals”-like parts (yellow)
2. Compile the code and run it.
   - shoot a particle and look at the text output
3. Add a new multi-functional detector
   - to measure the track length and the deposited energy
   - of particles with more than 200 keV kinetic energy
   - in the five disks (green detector parts) of “Linear Array” separately
4. Show the measured results in the text output event by event
5. Compile the code and test it.
6. Optional: Determine the coordinates of the hits in the five disks.



All measurements are in Geant4’s standard units.
Divide the number by the unit you want in order to get the correct number (e.g. GeV: `a_energy/GeV`, MeV: `a_energy/MeV`, …).
See also command: `/units/list`