

Stanford  
Linear  
Accelerator  
Center



# Primitive scorer and related classes

A. Kimura (Ashikaga I.T.)

T. Aso (Toyama NCMT)

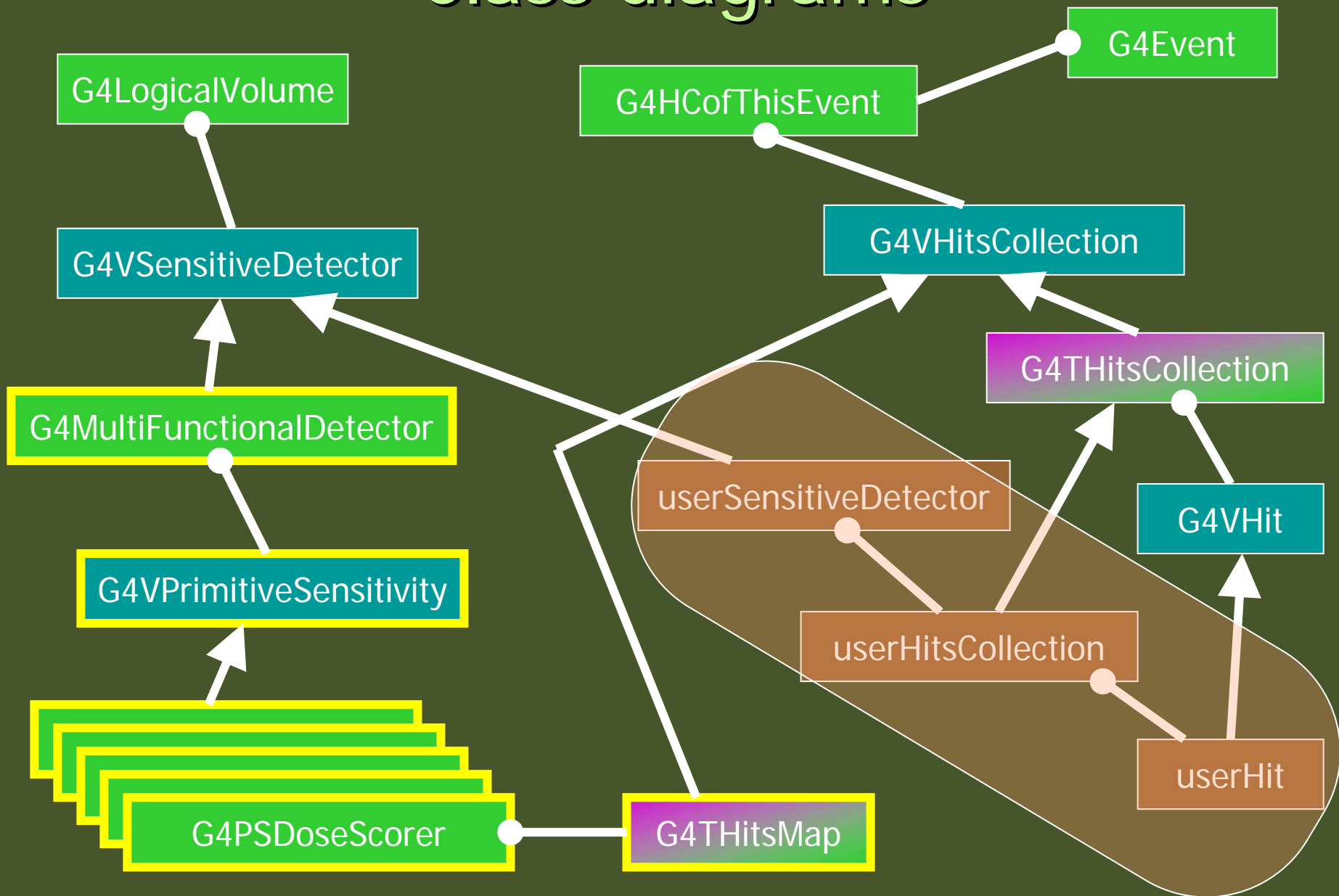
M. Asai (SLAC)

# Geant4

# Concrete sensitivity classes

- ▶ Up to the previous version (7.1), Geant4 provided only an abstract base class (**G4VSensitiveDetector**) for the user to define his/her detector sensitivity.
  - ▶ Various example detector classes are provided.
    - ▶ This is enough for HEP experiments, since their interest is mostly **storing** hits in their detectors.
  - ▶ But not convenient for many space and medical applications.
    - ▶ Their interest is mainly **scoring** dose/flux.
- ▶ We introduce **G4MultiFunctionalDetector** (concrete class derived from G4VSensitiveDetector) that allows the user to **register** concrete class objects of **G4VPrimitiveSensitivity** to build a scoring detector of his/her needs.
  - ▶ **G4PSEnergyDeposit**, **G4PSFlatSurfaceFlux**, **G4PSDoseDeposit**, **G4PSTrackLength**, etc. are provided.
    - ▶ See later slide for full listing.
  - ▶ We will continue working for additional primitive sensitivity concrete classes.

# Class diagrams



# Scorer classes in v8.0

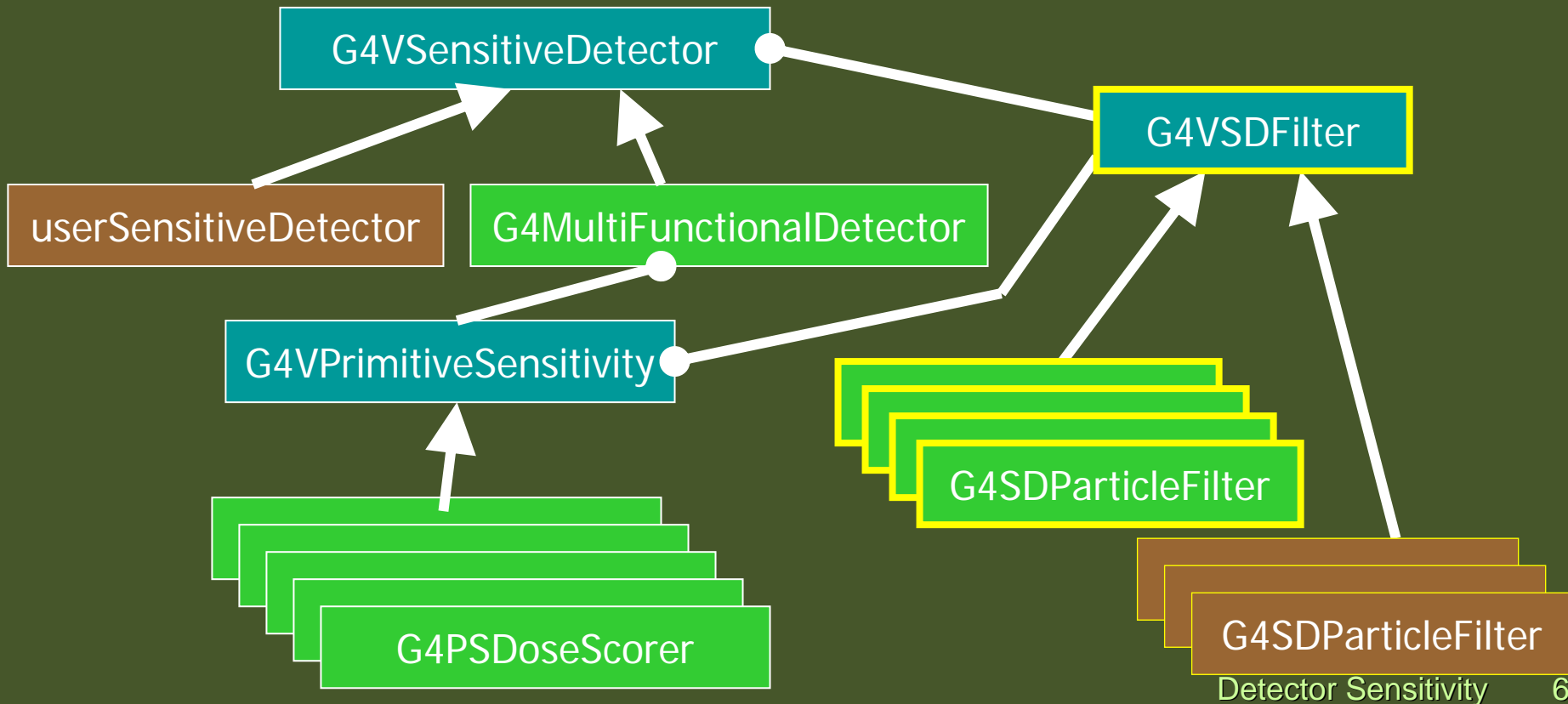
- ▶ Deposited energy scorers
  - ▶ G4PSEnergyDeposit, G4PSDoseDeposit
    - ▶ The dose deposit is defined by the sum of energy deposits at each step in the cell divided by the mass of the cell.
- ▶ Current and flux scorers
  - ▶ G4PSFlatSurfaceCurrent, G4PSSphereSurfaceCurrent, G4PSPassageCurrent
  - ▶ G4PSFlatSurfaceFlux, G4PSCellFlux, G4PSPassageCellFlux
    - ▶ Refer to the next page regarding these terminologies.
- ▶ Track length scorers
  - ▶ G4PSTrackLength, G4PSPassageTrackLength
    - ▶ Track length is sum of the step lengths in the cell.
    - ▶ “PassageTrackLength” means the track length only which passes through the cell.
- ▶ Other scorers
  - ▶ G4PSMinKinEAtGeneration, G4PSNofSecondary, G4PSNofStep

# G4THitsMap

- ▶ Each G4VPrimitiveSensitivity class generates **one** hits collection per event. By registering more than one classes of G4VPrimitiveSensitivity, G4MultiFunctionalDetector generates more than one collections.
- ▶ **G4THitsMap** template class (an alternative to **G4THitsCollection**) is introduced. It is also a derived class of **G4VHitsCollection**.
  - ▶ It is more convenient for scoring purposes. It does NOT mandate G4VHit concrete class to be stored, but for example a simple double value can be mapped with a copy number.
  - ▶ All new primitive sensitivity classes use G4THitsMap.

# G4VSDFilter

- ▶ New class **G4VSDFilter** is introduced. It can be attached to G4VSensitiveDetector and/or G4VPrimitiveSensitivity to define which kinds of tracks are to be scored.
  - ▶ E.g., surface flux of protons can be scored by **G4PSFlatSurfaceFlux** with a filter.



# For example...

```
MyDetectorConstruction::Construct()
```

```
{ ... G4LogicalVolume* myCellLog = new G4LogicalVolume(...);  
      G4VPhysicalVolume* myCellPhys = new G4PVParametrised(...);  
      G4MultiFunctionalDetector* myScorer = new G4MultiFunctionalDetector("myCellScorer");  
      G4SDManager::GetSDMpointer()->AddNewDetector(myScorer);  
      myCellLog->SetSensitiveDetector(myScorer);  
      G4VPrimitiveSensitivity* totalSurfFlux = new G4PSFlatSurfaceFlux("TotalSurfFlux");  
      myScorer->Register(totalSurfFlux);  
      G4VPrimitiveSensitivity* protonSurfFlux = new G4PSFlatSurfaceFlux("ProtonSurfFlux");  
      G4VSDFilter* protonFilter = new G4SDParticleFilter("protonFilter");  
      protonFilter->Add("proton");  
      protonSurfFlux->SetFilter(protonFilter);  
      myScorer->Register(protonSurfFlux);  
      G4VPrimitiveSensitivity* totalDose = new G4PSDoseDeposit("TotalDose");  
      myScorer->Register(totalDose);  
}
```

**No need of implementing  
sensitive detector !**

# Customized run class

```
#include "G4Run.hh"
#include "G4Event.hh"
#include "G4THitsMap.hh"
Class MyRun : public G4Run
{
public:
    MyRun();
    virtual ~MyRun();
    virtual void RecordEvent(const G4Event*);
private:
    G4int nEvent;
    G4int totalSurfFluxID, protonSurfFluxID, totalDoseID;
    G4THitsMap<G4double> totalSurfFlux;
    G4THitsMap<G4double> protonSurfFlux;
    G4THitsMap<G4double> totalDose;
    G4THitsMap<G4double>* eventTotalSurfFlux;
    G4THitsMap<G4double>* eventProtonSurfFlux;
    G4THitsMap<G4double>* eventTotalDose;
public:
    ... access methods ...
};
```

## Note :

**This sample code uses newly introducing concrete sensitivity classes.**

**Implement how you accumulate event data**



# Customized run class

```
MyRun::MyRun() : nEvent(0)
```

**name of G4MultiFunctionalDetector object**



```
{  
  G4SDManager* SDM = G4SDManager::GetSDMpointer();  
  totalSurfFluxID = SDM->GetCollectionID("myCellScorer/TotalSurfFlux");  
  protonSurfFluxID = SDM->GetCollectionID("myCellScorer/ProtonSurfFlux");  
  totalDoseID = SDM->GetCollectionID("myCellScorer/TotalDose");  
}
```

**name of G4VPrimitiveSensitivity object**



```
void MyRun::RecordEvent(const G4Event* evt)
```

```
{  
  nEvent++;  
  G4HCofThisEvent* HCE = evt->GetHCofThisEvent();  
  eventTotalSurfFlux = (G4THitsMap<G4double>*)(HCE->GetHC(totalSurfFluxID));  
  eventProtonSurfFlux = (G4THitsMap<G4double>*)(HCE->GetHC(protonSurfFluxID));  
  eventTotalDose = (G4THitsMap<G4double>*)(HCE->GetHC(totalDose));  
  totalSurfFlux += *eventTotalSurfFlux;  
  protonSurfFlux += *eventProtonSurfFlux;  
  totalDose += *eventTotalDose;  
}
```

**No need of loops.  
+= operator is provided !**

# RunAction with customized run

```
G4Run* MyRunAction::GenerateRun()
{ return (new MyRun()); }

void MyRunAction::EndOfRunAction(const G4Run* aRun)
{
    MyRun* theRun = (MyRun*)aRun;
    // ... analyze / record / print-out your run summary
    // MyRun object has everything you need ...
}
```

- ▶ As you have seen, to accumulate event data, you do **NOT** need
  - ▶ Event / tracking / stepping action classes
- ▶ All you need are your **Run and RunAction** classes.
- ▶ With newly introducing concrete sensitivity classes, you do **NOT** even need
  - ▶ Sensitive detector implementation

# Examples

- ▶ Examples/novice/N07 had been fully revised with new scorer classes.
- ▶ New example extended/runAndEvent/RE02 has been added for full demonstration.
- ▶ New tutorial hands-on example is under development. It should be publicized along with SLAC tutorial materials in March.