Sensitive Detector & Hits

A. Schälicke
(based on slides by A. Dotti)

Collecting Information

- ***User Action*** *allow to interact with the simulation of the physics and collect information for analysis*
- ***Hits*** *simplify the job in collecting information for active parts of the detector*
- ***Hits*** *are created only for the pieces of the detector that are defined sensitive:* ***SensitiveDetector****.*

  Example: in a Sandwich-Calorimeter the SD is the active layer

Sensitive Detector

Sensitive Detector

- *Each logical volume can have an associated SD: a user-defined class derived from G4VSensitiveDetector*

```
sensitive= new ScintillatorSD("/myDet/Adsorber");
G4SDManager::GetSDMpointer()->AddNewDetector(sensitive);
//needed to garantee calls to Initialize and EndOfEvent methods!

logicScintillator->SetSensitiveDetector(sensitive);
```

Sensitive Detector

- *Each logical volume can have an associated SD: a user-defined class derived from G4VSensitiveDetector*
- *SDs must have a unique name, however the same SD can be shared between different logical volumes. In our exercise, the same SD is shared between all active layers of the calorimeter.*
- *SD is created and associated to detector planes in DetectorConstruction class in Construct method.*

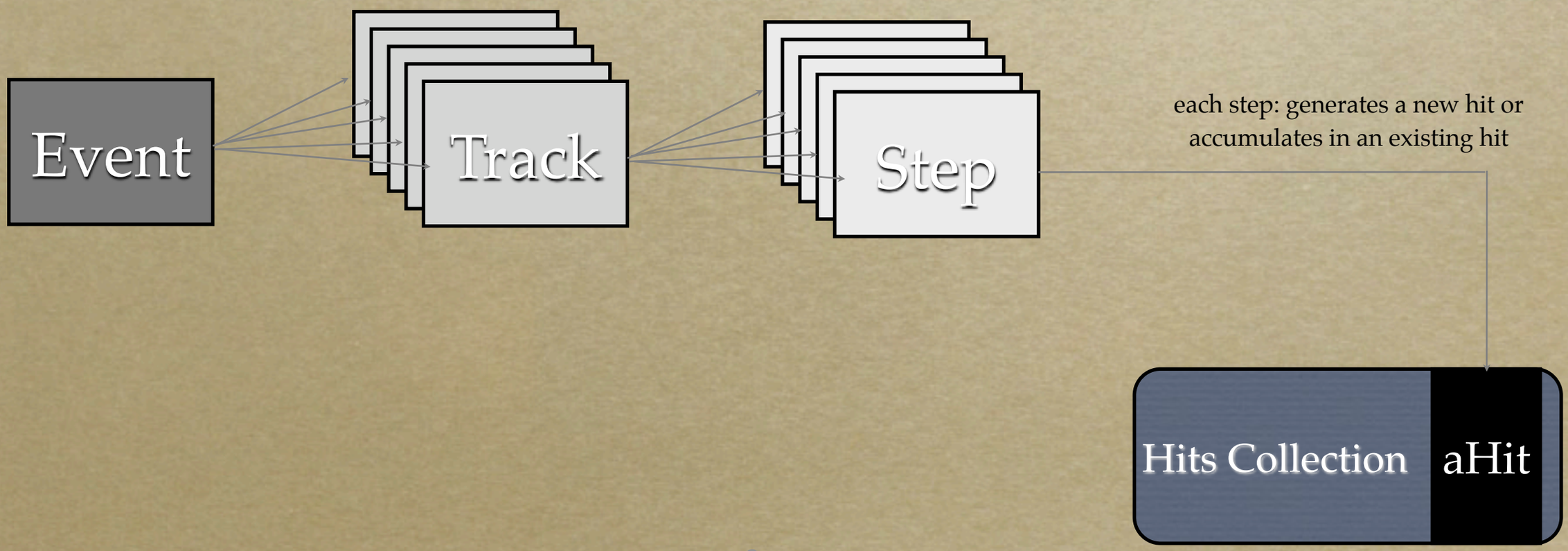
Sensitive Detector class

```
class ScintillatorSD : public G4VSensitiveDetector
{
public:
  /// Constructor
  ScintillatorSD(G4String SDname);
  /// Destructor
  ~ScintillatorSD();

public:
  /// @name methods from base class G4VSensitiveDetector
  //@{
  /// Mandatory base class method : it must to be overloaded:
   G4bool ProcessHits(G4Step *step, G4TouchableHistory *ROhist);

  /// (optional) method of base class G4VSensitiveDetector
  void Initialize(G4HCofThisEvent* HCE);
  /// (optional) method of base class G4VSensitiveDetector
  void EndOfEvent(G4HCofThisEvent* HCE);
  //@}


private:
  SiHitCollection*      hitCollection;
};
```

Hits

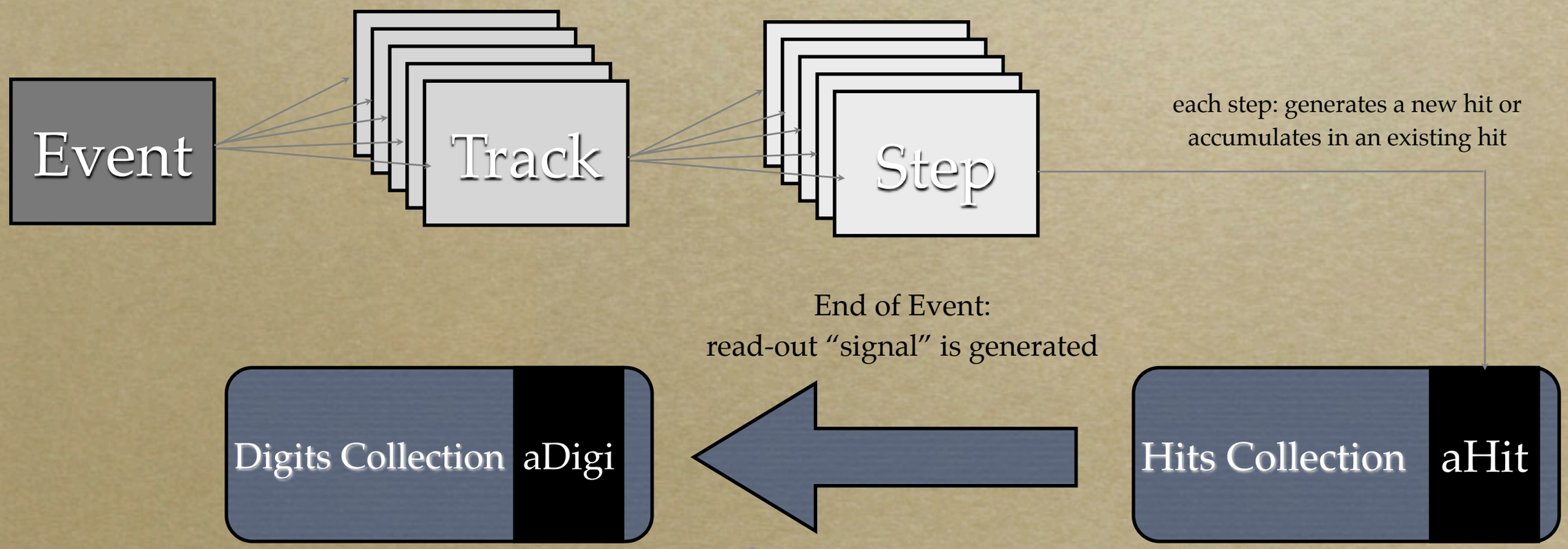
Hits Vs Digits

- *Hits are a "snapshot" of the physical interaction of a track (step) or an accumulation of interactions of tracks in the sensitive region of the detector, thus hits represent the "true" energy deposited in the detector*
- *Digits are instead intended to be used to simulate the process of reading-out of the signal: for example "true" energy is transformed into collected charge, electronic noise can be applied together with all instrumental effects*



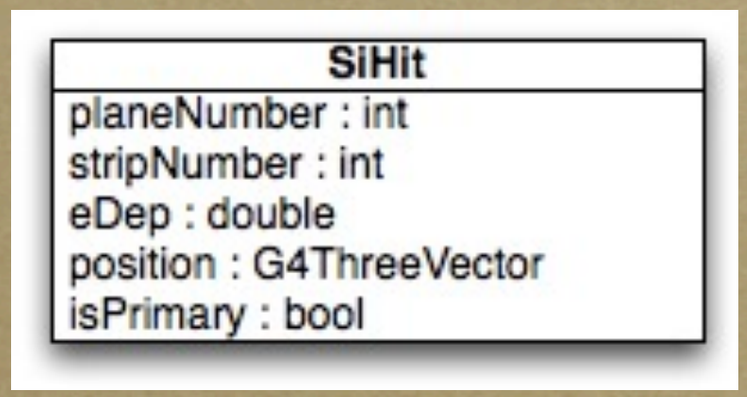
Hits Vs Digits

- *Hits are a "snapshot" of the physical interaction of a track (step) or an accumulation of interactions of tracks in the sensitive region of the detector, thus hits represent the "true" energy deposited in the detector*
- *Digits are instead intended to be used to simulate the process of reading-out of the signal: for example "true" energy is transformed into collected charge, electronic noise can be applied together with all instrumental effects*



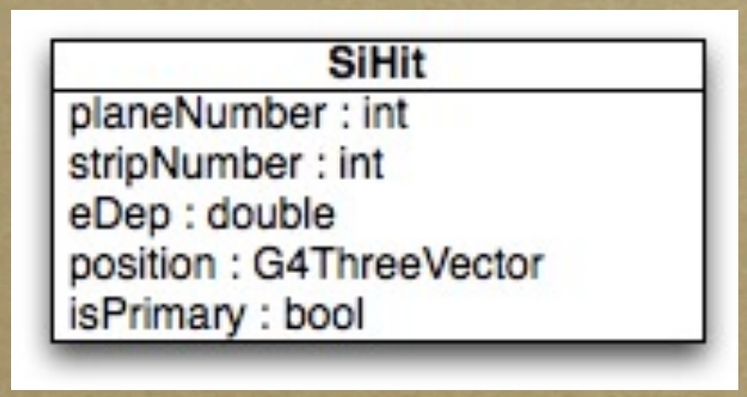
Implementing your Hit class

- *Hit is a user-defined class derived from G4VHit*
- *You can store any type of information by implementing your concrete Hit class. For example: position of the step, energy deposition of the step*
- *See SimpleHit class:*
  - *Accumulates energy of all steps in each layer*
  - *Contains also information about absolute position of the energy deposit*



Implementing your Hit class

- *Hit is a user-defined class derived from G4VHit*
- *You can store any type of information by implementing your concrete Hit class. For example: position of the step, energy deposition of the step*
- *See SimpleHit class:*
  - *Accumulates energy of all steps in each layer*
  - *Contains also information about absolute position of the energy deposit*



- Hits must be stored in a collection of hits instantiated from G4THitsCollection template class
- G4 provides optimized allocators for memory management

The Hits collection

- *Hits are accumulated in the hits collection*
- *Each collection has a unique name (a string):*
  multiple collections can be retrieved by name
- *However searching a string can be time consuming: a unique ID (integer) is also (automatically) associated to each collection*
  - *Ask G4 which ID corresponds to your name and use ID to get the collection*

The SD interface - 1

- *Constructor:*

```
ScintillatorSD::ScintillatorSD(G4String SDname)
  : G4VSensitiveDetector(SDname), hitCollection(0)
{
  G4cout<<"Creating SD with name:"<<SDname<<G4endl;

  // 'collectionName' is a protected data member of
  // base class   G4VScintillatorSD.
  // Here we declare the name of the collection we will be using.
  collectionName.insert("SiHitCollection");

  // Note that we may add as many collection names we would wish: ie
  // a sensitive detector can have many collections.
}
```

- *In the constructor, define the name of the hits collection handled by this SD*
- *In case your sensitive detector generates more than one kind of hits, define all collection names*

The SD Interface - 2

- *Initialize() method is invoked at beginning of each event*
- *You can get here the unique ID associated to your collection*
- *Instantiate the hits collection and attach it to G4HCofThisEvent passed as argument*

```
void ScintillatorSD::Initialize(G4HCofThisEvent* HCE)
{
  // ------------------------------
  // -- Creation of the collection
  // ------------------------------
  // -- collectionName[0] is "SiHitCollection", as declared in constructor
  std::cout<<"create new hitcollection "<<GetName()<<" "<<collectionName[0]<<std::endl;
  hitCollection = new SiHitCollection(GetName(), collectionName[0]);

  // ---------------------------------------------------------------------
  // -- and attachment of this collection to the "Hits Collection of this Event":
  // ---------------------------------------------------------------------
   static G4int HCID = -1;
  if (HCID<0) HCID = GetCollectionID(0); // <<-- this is to get an ID for collectionName[0]
  HCE->AddHitsCollection(HCID, hitCollection);
}
```

The SD Interface - 3

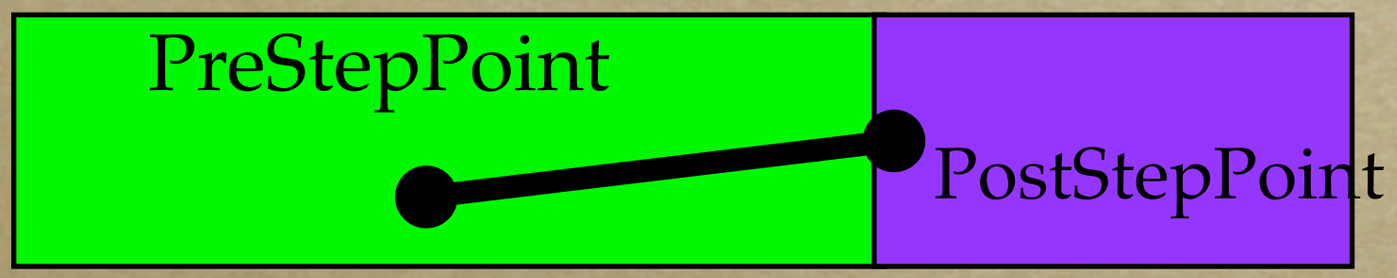
- *For each G4Step occurring in the (logical) volume to which this SD is attached the ProcessHits method is invoked*

```
G4bool ScintillatorSD::ProcessHits(G4Step *step, G4TouchableHistory *)
{
  // step is guaranteed to be in Scintillator volume :
  //    no need to check for volume !

  G4TouchableHandle touchable = step->GetPreStepPoint()->GetTouchableHandle();
  // energy deposit in this step
  G4double edep = step->GetTotalEnergyDeposit();

  if (edep <= 0.) return false;

  return true;
}
```

Reminder

Step

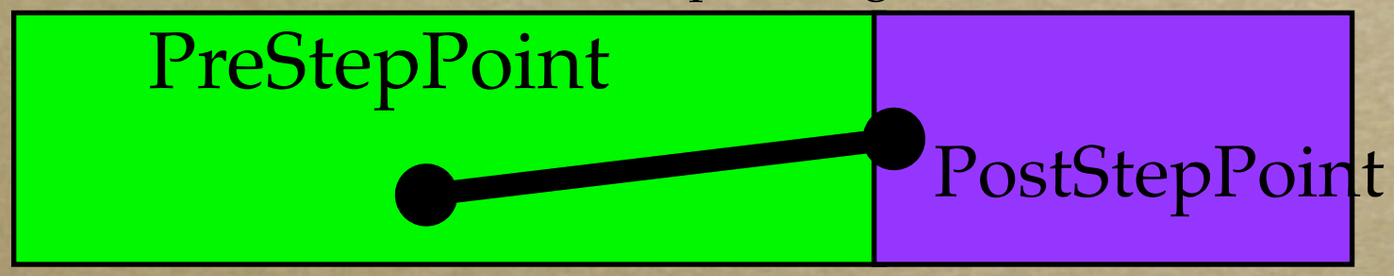
- *Step has two points and "delta" information of a particle (energy loss along the step, time-of-flight, etc)*
- *Each point knows the volume (and material) associated to it*
- *A step is always limited by geometry boundaries (i.e. never spans across boundaries)*
  - *If the step is limited by a boundary, the post-step point stands on the boundary and it logically belongs to the next volume*
  - *Get the volume information from the PreStepPoint*



Touchable: locate a Hit

- *It would be too complex to locate which strip the step belongs to from its position (G4ThreeVector). Each G4Step knows which volume it is in.*
- *Layers have been created as "replica"*
  - *In memory there is only one volume object "strip". Its position is parametrized by its replica number*
- ***Touchables** can retrieve these number*

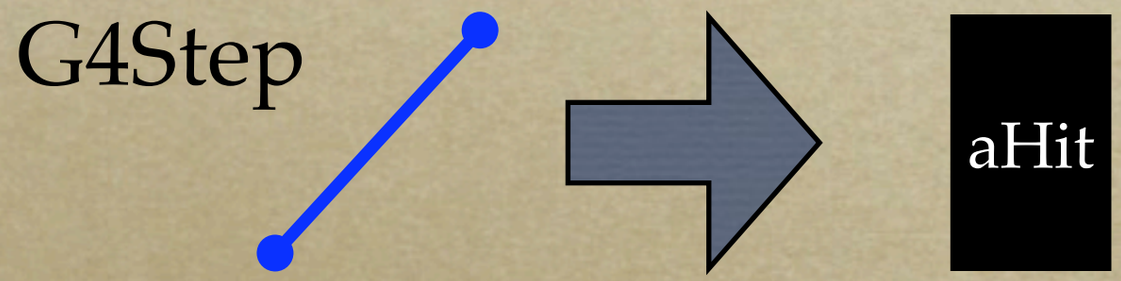
Remember: PostStep belongs to NEXT volume, use PreStepPoint!



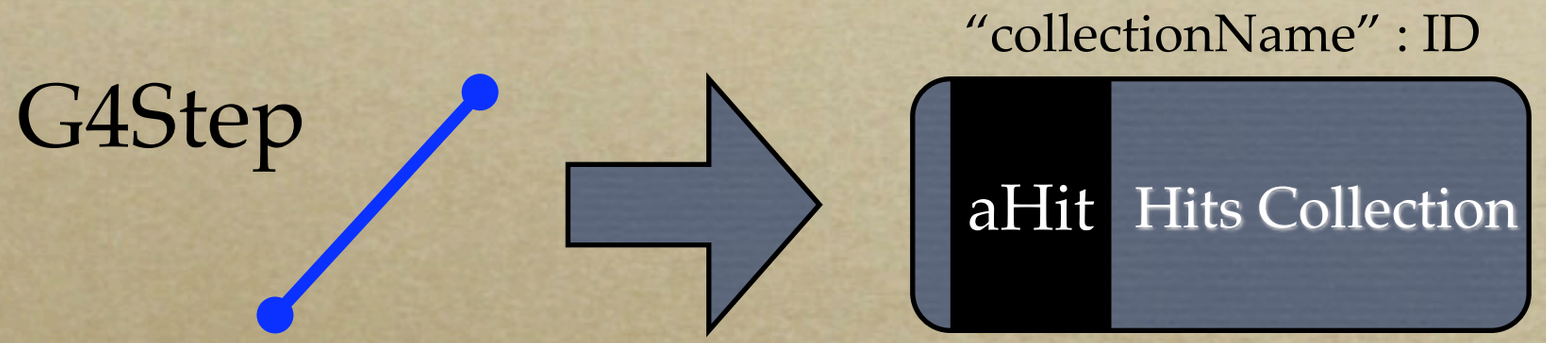
Summary

G4Step

Summary



Summary



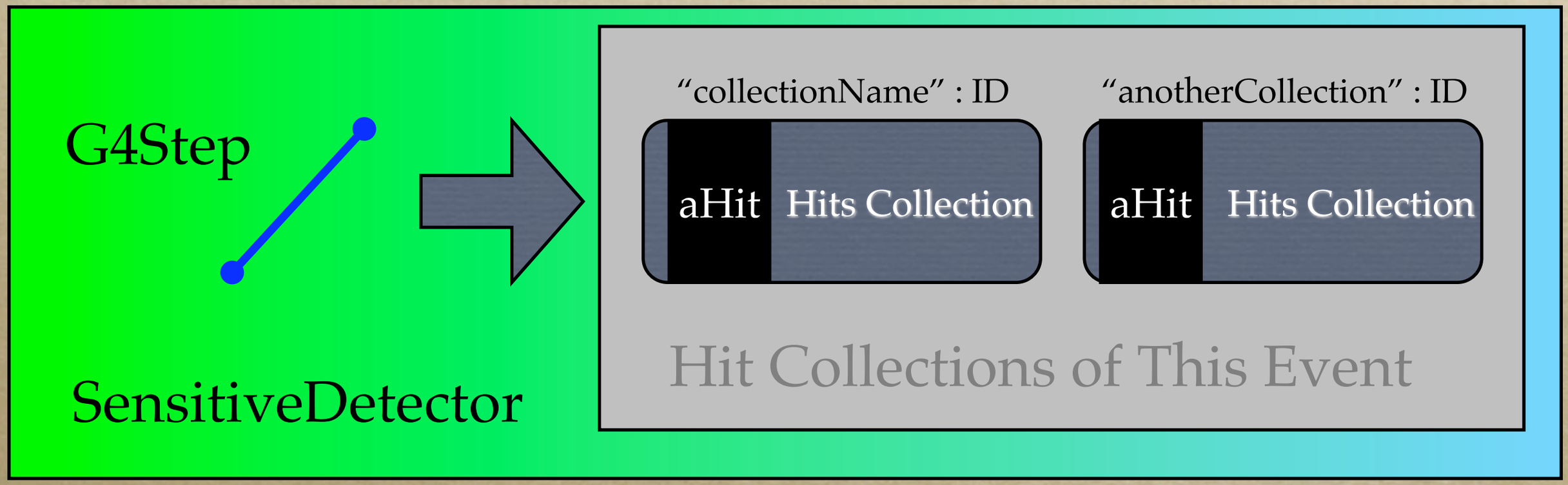
Summary


G4Step
"collectionName" : ID
aHit
Hits Collection
"anotherCollection" : ID
aHit
Hits Collection

Summary


G4Step
"collectionName" : ID
aHit
Hits Collection
"anotherCollection" : ID
aHit
Hits Collection
Hit Collections of This Event

Summary


G4Step
SensitiveDetector
"collectionName" : ID
aHit
Hits Collection
"anotherCollection" : ID
aHit
Hits Collection
Hit Collections of This Event

Exercises for Day 3

- http://www.ifh.de/geant4/g4course2011
- *Add a Sensitive Detector*
- *Create Hit collection*
- *Fill Histogramms*