

# HITS AND DIGITS

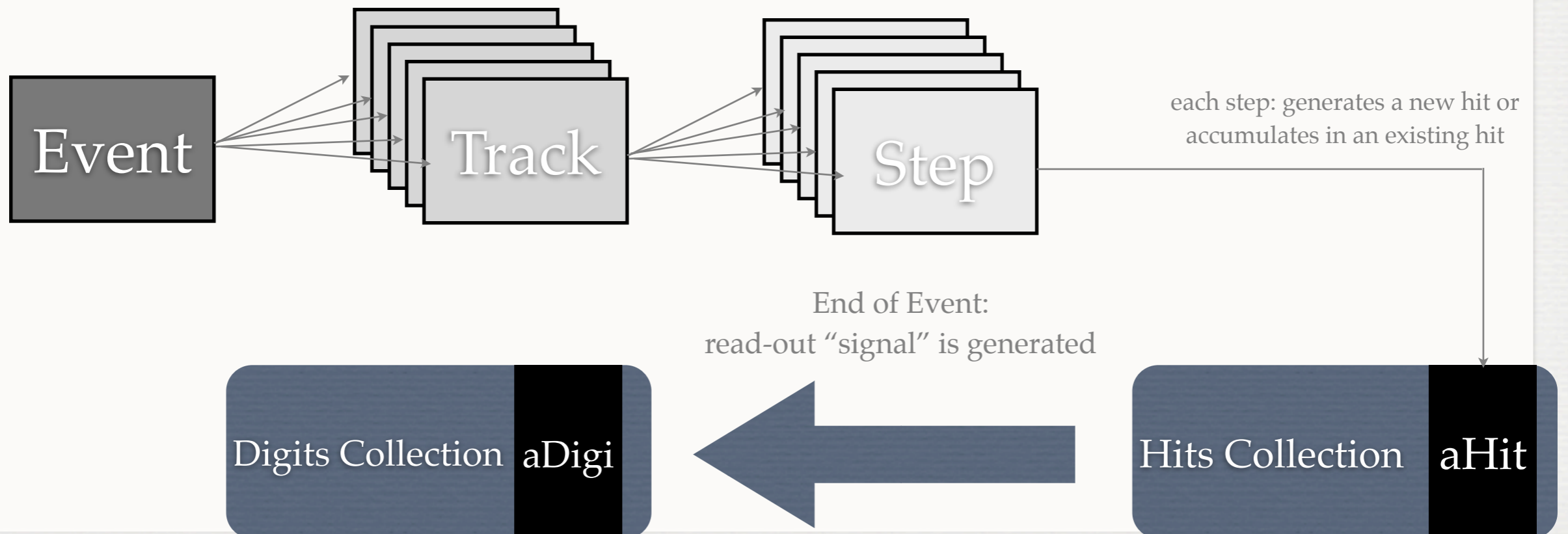
---

# 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 tracker the SD is the active part of the Si wafer, the electronics circuits do not participate in collecting charge, they are not SD thus do not generate 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



# 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 SiHit class: <http://www-zeuthen.desy.de/ILC/geant4/g4course2010/task2/classSiHit.html>
  - Accumulates energy of all steps in each strip
  - 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

SiHit
planeNumber : int
stripNumber : int
eDep : double
position : G4ThreeVector
isPrimary : bool


# Sensitive Detector

- Each **logical volume** can have an associated SD: a user-defined class derived from G4VSensitiveDetector
- See SensitiveDetector class: <http://www-zeuthen.desy.de/ILC/geant4/g4course2010/task2/classSensitiveDetector.html>
- **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 Si planes
- SD is created and associated to Si planes of the detector in DetectorConstruction class in Construct method. See: <http://www-zeuthen.desy.de/ILC/geant4/g4course2010/task2/classDetectorConstruction.html>

# The SD Interface - 1

## Constructor:

```
00028 SensitiveDetector::SensitiveDetector(G4String SDname)
00029   : G4VSensitiveDetector(SDname)
00030 {
00031
00032   // 'collectionName' is a protected data member of base class
G4VSensitiveDetector.
00033   // Here we declare the name of the collection we will be using.
00034   collectionName.insert("SiHitCollection");
00035
00036   // Note that we may add as many collection names we would wish: ie
00037   // a sensitive detector can have many collections.
00038 }
```

 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 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 - 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 SensitiveDetector::Initialize(G4HCofThisEvent* HCE)

00087 {
00088     // -- G4cout << "Initialize method of SD ` " << GetName() << "' called." << G4endl;
00089
00090     // -----
00091     // -- Creation of the collection
00092     // -----
00093     // -- collectionName[0] as declared in constructor
00094     hitCollection = new SiHitCollection(GetName(), collectionName[0]);
00095
00096     // -----
00097     // -- and attachment of this collection to the "Hits Collection of this Event":
00098     // -----
00099     // -- To insert the collection, we need to get an index for it. This index
00100     // -- is unique to the collection. It is provided by the GetCollectionID(...)
00101     // -- method (which calls what is needed in the kernel to get this index).
00102     static G4int HCID = -1;
00103     if (HCID<0) HCID = GetCollectionID(0); // <<-- this is to get an ID for collectionName[0]
00104     HCE->AddHitsCollection(HCID, hitCollection);
```



# 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 SensitiveDetector::Initialize(G4HCofThisEvent* HCE)

00087 {
00088     // -- G4cout << "Initialize method of SD ` " << GetName() << "' called." << G4endl;
00089
00090     // -----
00091     // -- Creation of the collection
00092     // -----
00093     // -- collectionName[0] as declared in constructor
00094     hitCollection = new SiHitCollection(GetName(), collectionName[0]);
00095
00096     // -----
00097     // -- and attachment of this collection to the "Hits Collection of this Event":
00098     // -----
00099     // -- To insert the collection, we need to get an index for it. This index
00100     // -- is unique to the collection. It is provided by the GetCollectionID(...)
00101     // -- method (which calls what is needed in the kernel to get this index).
00102     static G4int HCID = -1;
00103     if (HCID<0) HCID = GetCollectionID(0); // <<-- this is to get an ID for collectionName[0]
00104     HCE->AddHitsCollection(HCID, hitCollection);
```

# 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 SensitiveDetector::Initialize(G4HCofThisEvent* HCE)

00087 {
00088     // -- G4cout << "Initialize method of SD ` " << GetName() << " ' called." << G4endl;
00089
00090     // -----
00091     // -- Creation of the collection
00092     // -----
00093     // -- collectionName[0] as declared in constructor
00094     hitCollection = new SiHitCollection(GetName(), collectionName[0]);
00095
00096     // -----
00097     // -- and attachment of this collection to the "Hits Collection of this Event":
00098     // -----
00099     // -- To insert the collection, we need to get an index for it. This index
00100     // -- is unique to the collection. It is provided by the GetCollectionID(...)
00101     // -- method (which calls what is needed in the kernel to get this index).
00102     static G4int HCID = -1;
00103     if (HCID<0) HCID = GetCollectionID(0); // <<-- this is to get an ID for collectionName[0]
00104     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

```
00043 G4bool SensitiveDetector::ProcessHits(G4Step *step, G4TouchableHistory *)
00044 {
00045     // step is guaranteed to be in Strip volume : no need to check for volume
00046
00047     G4TouchableHandle touchable = step->GetPreStepPoint()->GetTouchableHandle();
00048     // energy deposit in this step
00049     G4double edep = step->GetTotalEnergyDeposit();
00050
00051     // get step points in world coordinate system
00052     G4ThreeVector point1 = step->GetPreStepPoint()->GetPosition();
00053     G4ThreeVector point2 = step->GetPostStepPoint()->GetPosition();
00054
00055     // randomize point of energy deposition
00056     G4ThreeVector pointE = point1 + G4UniformRand()*(point2 - point1);
00057
00058
00059     G4int stripCopyNo = touchable->GetReplicaNumber();
00060     G4int planeCopyNo = touchable->GetReplicaNumber(1);
00061
00062
00063     SiHit* hit = new SiHit(stripCopyNo,planeCopyNo,isPrimary);
00064     hitCollection->insert(hit); // size of collection is returned by insert(..)
00065
00066     // set energy deposition
00067     hit->AddEdep(edep);
00068     // store position of energy deposition
00069     hit->SetPosition(pointE);
```

# The SD Interface - 3

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

```
00043 G4bool SensitiveDetector::ProcessHits(G4Step *step, G4TouchableHistory *)
00044 {
00045     // step is guaranteed to be in Strip volume : no need to check for volume
00046
00047     G4TouchableHandle touchable = step->GetPreStepPoint()->GetTouchableHandle();
00048     // energy deposit in this step
00049     G4double edep = step->GetTotalEnergyDeposit();
00050
00051     // get step points in world coordinate system
00052     G4ThreeVector point1 = step->GetPreStepPoint()->GetPosition();
00053     G4ThreeVector point2 = step->GetPostStepPoint()->GetPosition();
00054
00055     // randomize point of energy deposition
00056     G4ThreeVector pointE = point1 + G4UniformRand()*(point2 - point1);
00057
00058
00059     G4int stripCopyNo = touchable->GetReplicaNumber();
00060     G4int planeCopyNo = touchable->GetReplicaNumber(1);
00061
00062
00063     SiHit* hit = new SiHit(stripCopyNo,planeCopyNo,isPrimary);
00064     hitCollection->insert(hit); // size of collection is returned by insert(..)
00065
00066     // set energy deposition
00067     hit->AddEdep(edep);
00068     // store position of energy deposition
00069     hit->SetPosition(pointE);
```

# The SD Interface - 3

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

```
00043 G4bool SensitiveDetector::ProcessHits(G4Step *step, G4TouchableHistory *)
00044 {
00045     // step is guaranteed to be in Strip volume : no need to check for volume
00046
00047     G4TouchableHandle touchable = step->GetPreStepPoint()->GetTouchableHandle();
00048     // energy deposit in this step
00049     G4double edep = step->GetTotalEnergyDeposit();
00050
00051     // get step points in world coordinate system
00052     G4ThreeVector point1 = step->GetPreStepPoint()->GetPosition();
00053     G4ThreeVector point2 = step->GetPostStepPoint()->GetPosition();
00054
00055     // randomize point of energy deposition
00056     G4ThreeVector pointE = point1 + G4UniformRand()*(point2 - point1);
00057
00058
00059     G4int stripCopyNo = touchable->GetReplicaNumber();
00060     G4int planeCopyNo = touchable->GetReplicaNumber(1);
00061
00062
00063     SiHit* hit = new SiHit(stripCopyNo,planeCopyNo,isPrimary);
00064     hitCollection->insert(hit); // size of collection is returned by insert(..)
00065
00066     // set energy deposition
00067     hit->AddEdep(edep);
00068     // store position of energy deposition
00069     hit->SetPosition(pointE);
```

# The SD Interface - 3

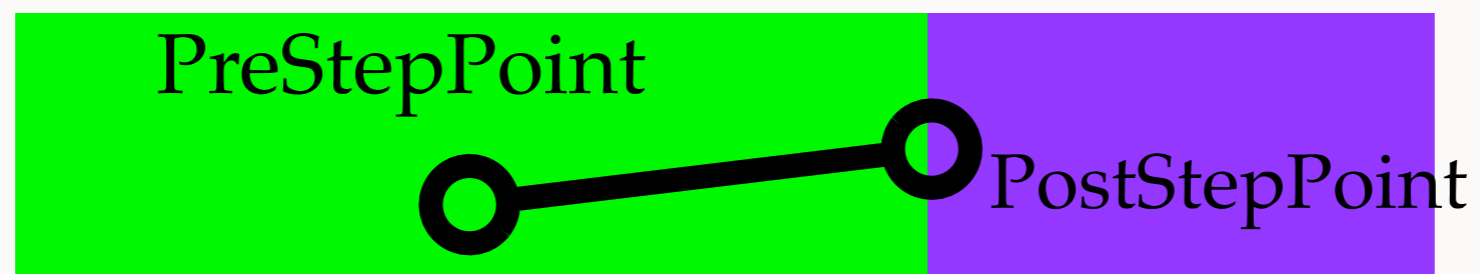
- For each **G4Step** occurring in the (logical) volume to which this SD is attached the **ProcessHits** method is invoked

```
00043 G4bool SensitiveDetector::ProcessHits(G4Step *step, G4TouchableHistory *)
00044 {
00045     // step is guaranteed to be in Strip volume : no need to check for volume
00046
00047     G4TouchableHandle touchable = step->GetPreStepPoint()->GetTouchableHandle();
00048     // energy deposit in this step
00049     G4double edep = step->GetTotalEnergyDeposit();
00050
00051     // get step points in world coordinate system
00052     G4ThreeVector point1 = step->GetPreStepPoint()->GetPosition();
00053     G4ThreeVector point2 = step->GetPostStepPoint()->GetPosition();
00054
00055     // randomize point of energy deposition
00056     G4ThreeVector pointE = point1 + G4UniformRand()*(point2 - point1);
00057
00058
00059     G4int stripCopyNo = touchable->GetReplicaNumber();
00060     G4int planeCopyNo = touchable->GetReplicaNumber(1);
00061
00062
00063     SiHit* hit = new SiHit(stripCopyNo,planeCopyNo,isPrimary);
00064     hitCollection->insert(hit); // size of collection is returned by insert(..)
00065
00066     // set energy deposition
00067     hit->AddEdep(edep);
00068     // store position of energy deposition
00069     hit->SetPosition(pointE);
```

# Step

Reminder

- 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.
- Example: the detector you have built in Task1.1 is made of 3 identical planes of Si, each one made of 48 identical strips
- Strips have been created as “replica”
  - In memory there is only one volume object “strip”. Its position is parametrized by its replica number
  - We also need the number of the “mother volume” containing the strip: the plane number
- Touchables can retrieve these number

Remember: PostStep belongs to NEXT volume, use PreStepPoint!

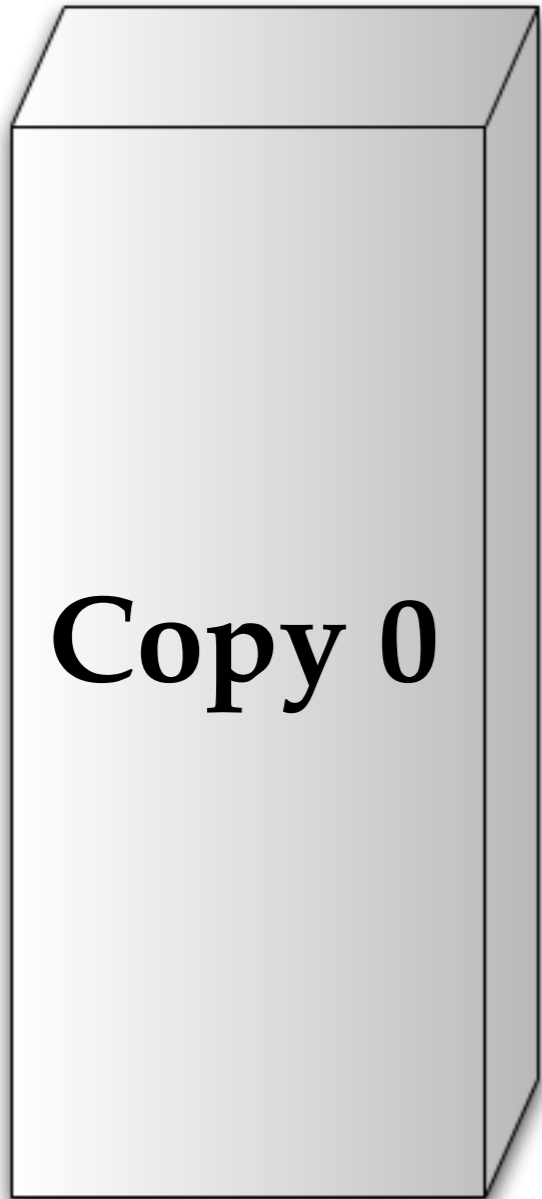
```
00047  G4TouchableHandle touchable = step->GetPreStepPoint()->GetTouchableHandle();
00069  G4int stripCopyNo = touchable->GetReplicaNumber();
00070  G4int planeCopyNo = touchable->GetReplicaNumber(1);
```

Go up by one in hierarchy: the plane

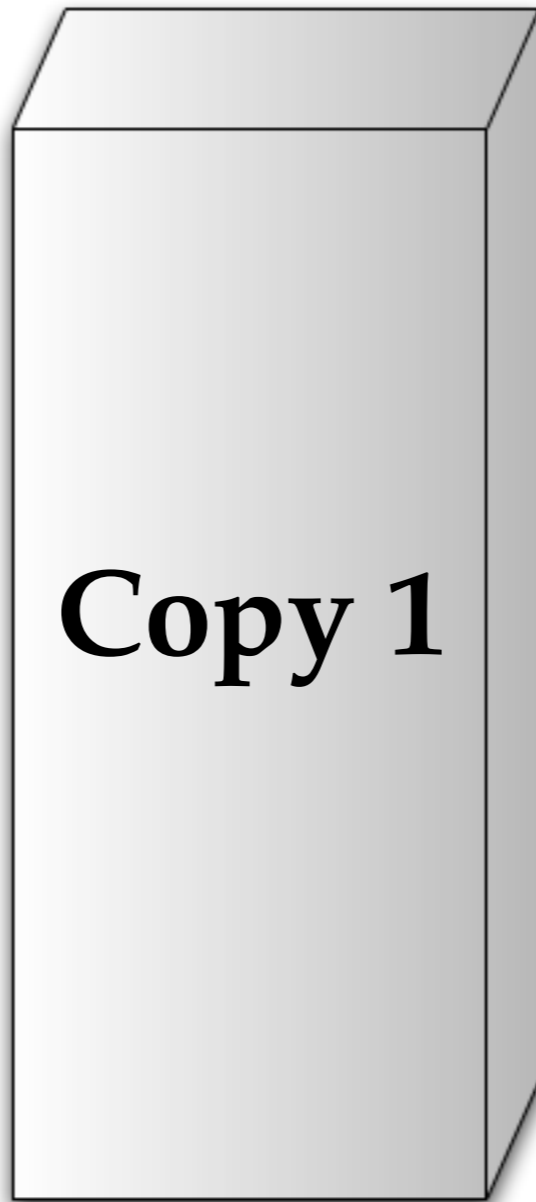


# Touchable: Locate A

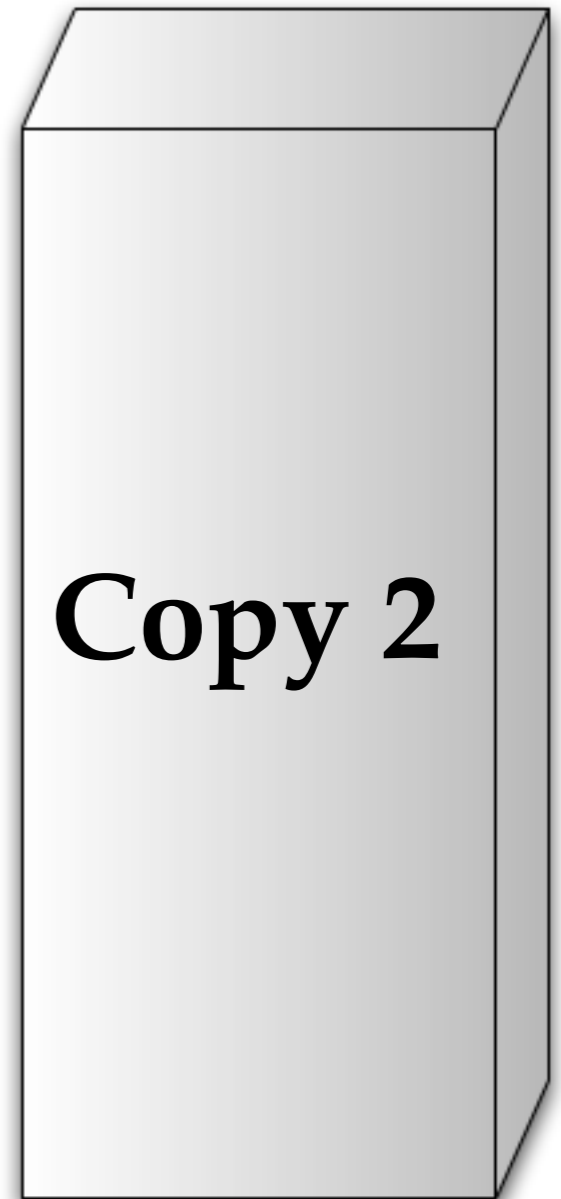
- It would be too complex to position (G4ThreeVector). E
- Example: the detector you l



of 48 ide  
reated as  
re is only  
y its repl  
ie numbe



tep below  
ch volum  
nade of  
ip". Its p  
ie" conta



trieve the  
: PostStep

e, use PreStepPoint!

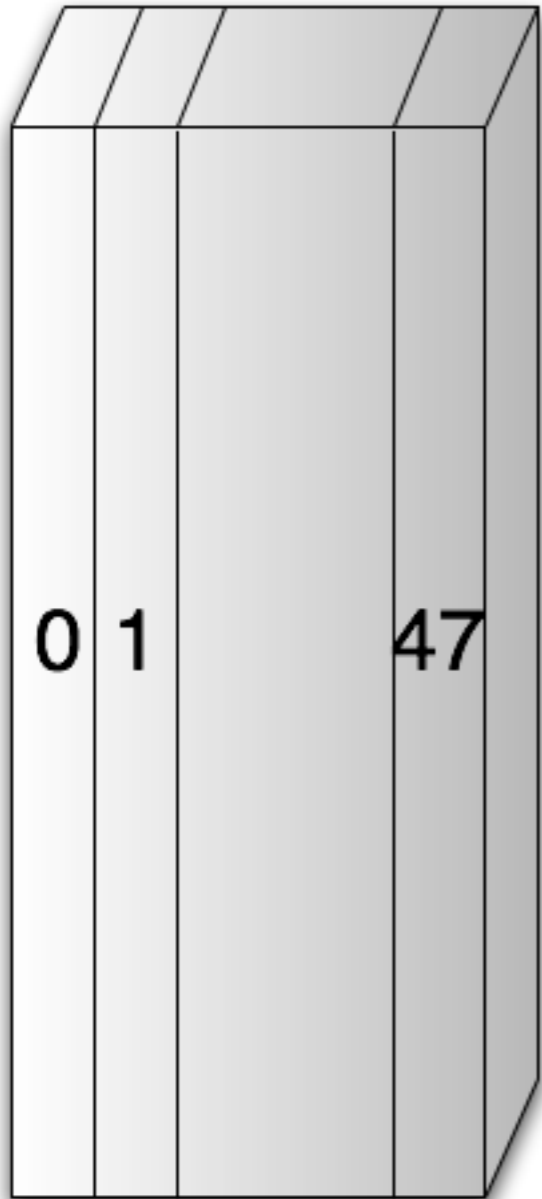
0004  
0006  
0007

```
hable = step->GetPreStepPoint()->GetTouchableHandle();  
uchable->GetReplicaNumber();  
uchable->GetReplicaNumber(1);
```

Go up by one in hierarchy: the plane

# Touchable: Locate A

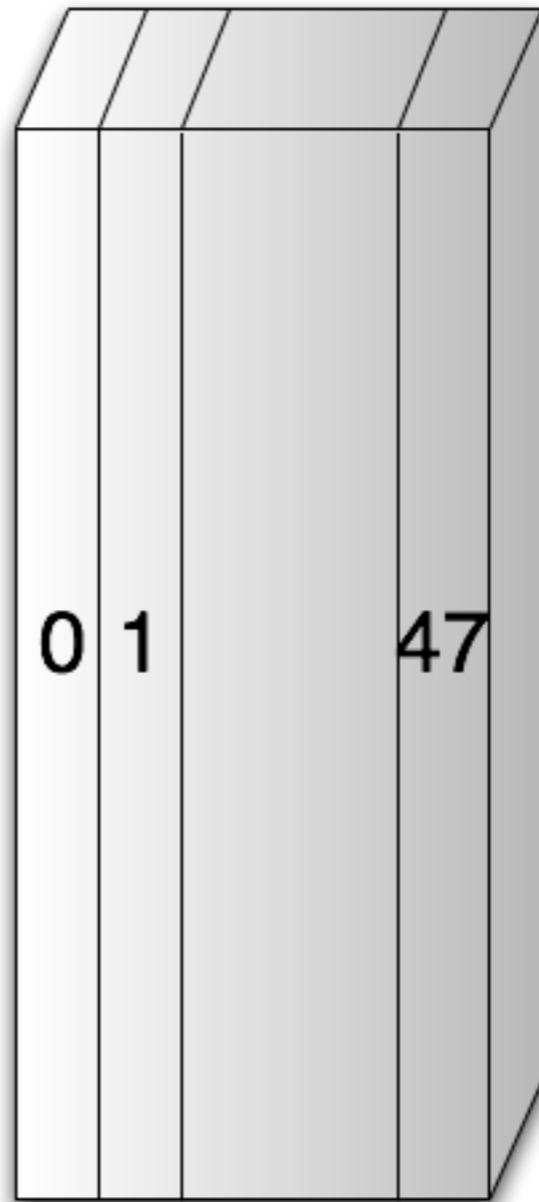
- It would be too complex to position (G4ThreeVector). E
- Example: the detector you l



of 48 ide  
reated as  
re is only  
y its repl  
ie numbe

trieve the

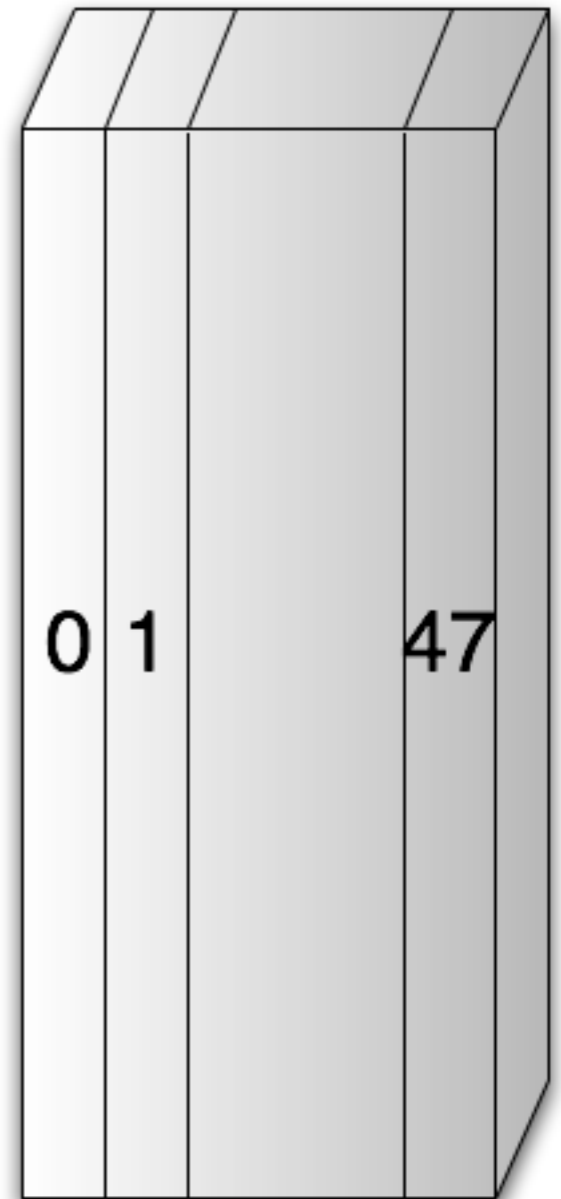
: PostStep



tep below  
ch volum  
nade of

ip". Its p  
ie" conta

e, use PreStepPoint!



0004  
0006  
0007

```

touchable = step->GetPreStepPoint()->GetTouchableHandle();
touchable->GetReplicaNumber();
touchable->GetReplicaNumber(1);
    
```

Go up by one in hierarchy: the plane

# Retrieving Hits

- Hits collections can be retrieved by name.
  - First retrieve the collection ID
- Hits are associated to the current G4Event object, it is possible to retrieve the hits collection:

```
void EventAction::EndOfEventAction(const G4Event* anEvent)
00049 {
00044   G4SDManager * SDman = G4SDManager::GetSDMpointer();
00045   G4int hitsCollID = SDman->GetCollectionID(hitsCollName);
00061   //Retrieve digits collection
00065   G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();
00066   SiHitCollection* hits = 0;
00069   hits = static_cast<SiHitCollection*>( hitsCollections->GetHC
(hitsCollID) );
```

# Retrieving Hits

- Hits collections can be retrieved by name.
  - First retrieve the collection ID
- Hits are associated to the current G4Event object, it is possible to retrieve the hits collection:

```
void EventAction::EndOfEventAction(const G4Event* anEvent)
00049 {
00044   G4SDManager * SDman = G4SDManager::GetSDMpointer();
00045   G4int hitsCollID = SDman->GetCollectionID(hitsCollName);
00061   //Retrieve digits collection
00065   G4HCofThisEvent* hitsCollections = anEvent->GetHCofThisEvent();
00066   SiHitCollection* hits = 0;
00069   hits = static_cast<SiHitCollection*>( hitsCollections->GetHC
(hitsCollID) );
```

# Digitization

- At the end of one event the G4HCofThisEvent object contains the hits we have created
- **The Digitizer module** (electronic read-out simulator) can be used **to transform Hits to Digits**
- SiDigi class, inherits from G4VDigi. Digits are stored in container, an instance of G4TDigiCollection class: very similar to hits mechanism
- **See:** <http://www-zeuthen.desy.de/ILC/geant4/g4course2010/task2/classSiDigi.html>

# Digitizer

- Digitizer is identified by name and has to be registered to the DigiManager singleton
- The SiDigitizer class inherits from G4VDigitizerModule base class and implements the Digitize() method
  - Warning: this method has to be called explicitly at the end of the event

```
EventAction::EventAction()  
00026 {  
00027     //We build the digitization module  
00028     SiDigitizer* digitizer = new SiDigitizer("SiDigitizer");  
00029     G4DigiManager * digiManager = G4DigiManager::GetDMpointer();  
00030     digiManager->AddNewModule( digitizer );  
00031 }  
  
00048 void EventAction::EndOfEventAction(const G4Event* anEvent)  
00049 {  
00050     //Digitize!!  
00051     G4DigiManager * digiManager = G4DigiManager::GetDMpointer();  
00052     SiDigitizer* digiModule = static_cast<SiDigitizer*>( digiManager->  
>FindDigitizerModule("SiDigitizer") );  
00055     digiModule->Digitize();
```

# Digitizer

- Digitizer is identified by name and has to be registered to the DigiManager singleton
- The SiDigitizer class inherits from G4VDigitizerModule base class and implements the Digitize() method
  - Warning: this method has to be called explicitly at the end of the event

```
EventAction::EventAction()
```

```
00026 {  
00027     //We build the digitization module  
00028     SiDigitizer* digitizer = new SiDigitizer("SiDigitizer");  
00029     G4DigiManager * digiManager = G4DigiManager::GetDMpointer();  
00030     digiManager->AddNewModule( digitizer );  
00031 }
```

```
00048 void EventAction::EndOfEventAction(const G4Event* anEvent)
```

```
00049 {  
00050     //Digitize!!  
00051     G4DigiManager * digiManager = G4DigiManager::GetDMpointer();  
00052     SiDigitizer* digiModule = static_cast<SiDigitizer*>( digiManager->  
FindDigitizerModule("SiDigitizer") );  
00055     digiModule->Digitize();
```

# Digitizer

- Digitizer creates the digit collection, similarly to hits, these are identified by a collection name

```
00021 SiDigitizer::SiDigitizer(G4String aName) :  
00022   G4VDigitizerModule(aName)  
00043 {  
00044     collectionName.push_back( digiCollectionName );  
00047 }
```

- Digits are created and added to the collection:

```
00049 void SiDigitizer::Digitize()  
00050 {  
00052   SiDigiCollection * digiCollection = new SiDigiCollection("SiDigitizer",digiCollectionName);  
00053   //Create a empty collection with one digits for each strip  
00055   const G4int numPlanes = 3; //Number of Si detectors  
00056   const G4int numStrips = 48; //Number of strip per plane  
00067   for ( G4int plane = 0 ; plane < numPlanes ; ++plane ) {  
00068     for ( G4int strip = 0 ; strip < numStrips ; ++strip )  
00069       {  
00070         SiDigi* newDigi = new SiDigi(plane,strip);  
00072         digiCollection->insert(newDigi);  
00073       }  
00074   }  
00129   StoreDigiCollection(digiCollection);
```



# Digitizer

- Digitizer creates the digit collection, similarly to hits, these are identified by a collection name

```
00021 SiDigitizer::SiDigitizer(G4String aName) :  
00022     G4VDigitizerModule(aName)  
00043 {  
00044     collectionName.push_back( digiCollectionName );  
00047 }
```

- Digits are created and added to the collection:

```
00049 void SiDigitizer::Digitize()  
00050 {  
00052     SiDigiCollection * digiCollection = new SiDigiCollection("SiDigitizer",digiCollectionName):  
00053     //Create a empty collection with one digits for each strip  
00055     const G4int numPlanes = 3; //Number of Si detectors  
00056     const G4int numStrips = 48; //Number of strip per plane  
00067     for ( G4int plane = 0 ; plane < numPlanes ; ++plane ) {  
00068         for ( G4int strip = 0 ; strip < numStrips ; ++strip )  
00069             {  
00070                 SiDigi* newDigi = new SiDigi(plane,strip);  
00072                 digiCollection->insert(newDigi);  
00073             }  
00074     }  
00129     StoreDigiCollection(digiCollection);
```

# Digitizer

- Digitizer creates the digit collection, similarly to hits, these are identified by a collection name

```
00021 SiDigitizer::SiDigitizer(G4String aName) :  
00022   G4VDigitizerModule(aName)  
00043 {  
00044     collectionName.push_back( digiCollectionName );  
00047 }
```

- Digits are created and added to the collection:

```
00049 void SiDigitizer::Digitize()  
00050 {  
00052   SiDigiCollection * digiCollection = new SiDigiCollection("SiDigitizer",digiCollectionName);  
00053   //Create a empty collection with one digits for each strip  
00055   const G4int numPlanes = 3; //Number of Si detectors  
00056   const G4int numStrips = 48; //Number of strip per plane  
00067   for ( G4int plane = 0 ; plane < numPlanes ; ++plane ) {  
00068     for ( G4int strip = 0 ; strip < numStrips ; ++strip )  
00069     {  
00070       SiDigi* newDigi = new SiDigi(plane,strip);  
00072       digiCollection->insert(newDigi);  
00073     }  
00074   }  
00129   StoreDigiCollection(digiCollection);
```

# Digitizer

- Digitizer creates the digit collection, similarly to hits, these are identified by a collection name

```
00021 SiDigitizer::SiDigitizer(G4String aName) :  
00022     G4VDigitizerModule(aName)  
00043 {  
00044     collectionName.push_back( digiCollectionName );  
00047 }
```

- Digits are created and added to the collection:

```
00049 void SiDigitizer::Digitize()  
00050 {  
00052     SiDigiCollection * digiCollection = new SiDigiCollection("SiDigitizer",digiCollectionName);  
00053     //Create a empty collection with one digits for each strip  
00055     const G4int numPlanes = 3; //Number of Si detectors  
00056     const G4int numStrips = 48; //Number of strip per plane  
00067     for ( G4int plane = 0 ; plane < numPlanes ; ++plane ) {  
00068         for ( G4int strip = 0 ; strip < numStrips ; ++strip )  
00069             {  
00070                 SiDigi* newDigi = new SiDigi(plane,strip);  
00072                 digiCollection->insert(newDigi);  
00073             }  
00074     }  
00129     StoreDigiCollection(digiCollection);
```

# Retrieving Digits

- Digits collection can be retrieved, by name, via the **DigiManager singleton**

```
00051  G4DigiManager * digiManager = G4DigiManager::GetDMpointer();
00062  G4int digiCollID = digiManager->GetDigiCollectionID( digitsCollName );
00063  const SiDigiCollection* digits = static_cast<const SiDigiCollection*>
( digiManager->GetDigiCollection(digiCollID) );
```

- Remember retrieval is always a two-step process:

name (string)  $\Rightarrow$  ID (integer)  $\Rightarrow$  collection (pointer)

- Since IDs do not change during a run you can (should) optimize your code: do the first search only once
- Important: hits and digits collection are associated to current G4Event: you can retrieve them via `G4Event::GetHC(hitCollID)` and `G4Event::GetDC(digiCollID)`. See code

# Summary



# Summary



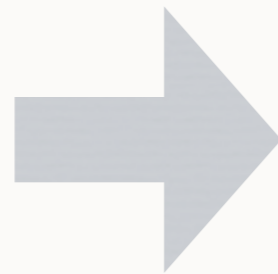
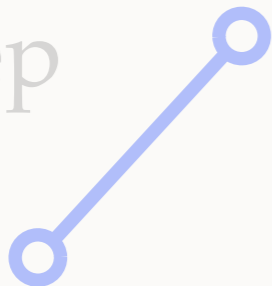
# Summary



Repeat for each step in the event

# Summary

G4Step

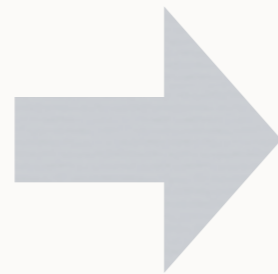
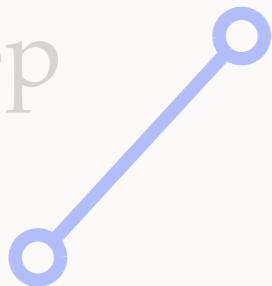


End of the event

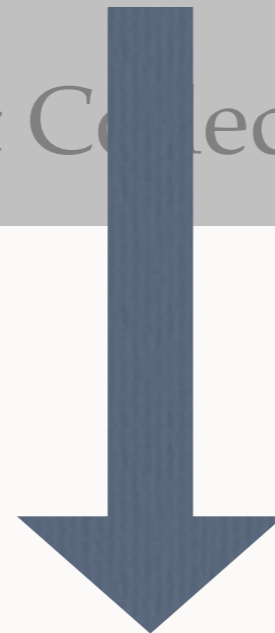


# Summary

G4Step

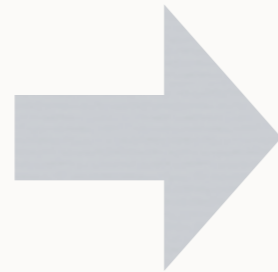
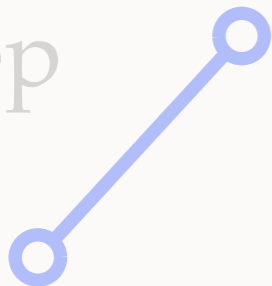


End of the event: Digitize



# Summary

G4Step

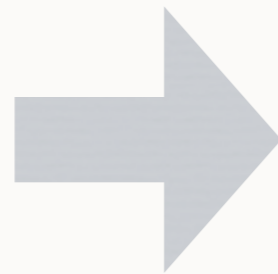
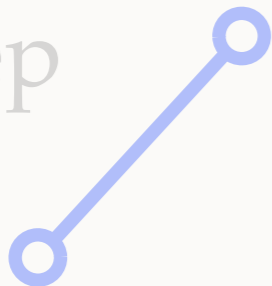


End of the event: Digitize

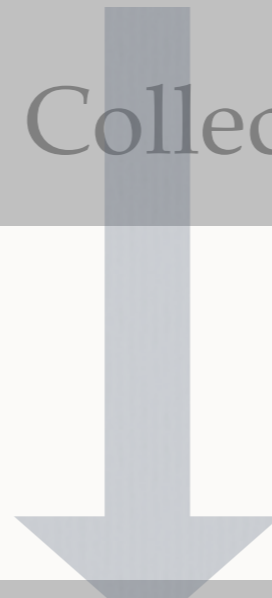


# Summary

G4Step

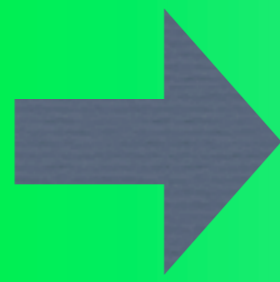
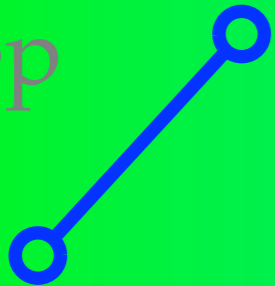


End of the event



# Summary

G4Step



SensitiveDetector



DigitizerModule



# Exercises For Task 1.2

- <http://www.ifh.de/geant4/g4course2010>
- Exercise 1.2.2 : Add a Sensitive Detector
- Exercise 1.2.3 : Modify simulation of noise
- Optional Exercise 1.2.4: Extend hits / digits classes