Scoring and analysis

T.Aso, NIT Toyama

Outline

- Command-based scorers
  - Define a mesh
  - Attach scorer
  - Draw and Save results

- Sensitive detectors, hits and hits collections
  - Extracting information
  - Storing in temporal record
  - Process the records to quantities
- Analysis Manager
  - Saving quantities
- Analysis
  - ROOT analysis tool

Command-based scorers

Command-based scoring

- Command-based scoring offers a build-in scoring mesh and various scorers for commonly-used physics quantities.
- To use this functionality, you need to activate the G4ScoringManager after the instantiation of G4RunManager in the main() program
  - This creates the UI commands for scoring that starts from “/score”

Main program such as main.cc

```
#include  “G4ScoringManager.hh”

int main(){
    // … snippet
    // Construct the default run manager
  auto* runManager =
    G4RunManagerFactory::CreateRunManager(G4RunManagerType::Default);
   // …
   // Scoring Manager
  G4ScoringManager* scManager = G4ScoringManager::GetScoringManager();
   // …
}
```

Defining Scoring Mesh

- **Define scoring mesh**
  - /score/create/boxMesh {meshName}
  - /score/mesh/boxSize {dx} {dy} {dz} {unit}
  - /score/mesh/nBin {nx} {ny} {nz}
  - /score/mesh/translate/xyz {x} {y} {z} {unit}
- **Assign scoring quantities**
  - /score/quantity/{scorer} {scoreName} {unit}
- **Close mesh**
  - /score/close

Example commands

```
# Define scoring mesh
/score/create/boxMesh  boxMesh
/score/mesh/boxSize  100.  100.  100. cm
/score/mesh/nBin  30  30  30
/score/mesh/translate/xyz  0.  0.  50. cm
#
# Define scoring quantity
/score/quantity/energyDeposit  eDep  keV
#
# Close mesh
/score/close
```

Available scorers in the command-line scorer

- Following scorers are available
  - See Application Developer Guide in detail

Name of primitive scorer	Description	Default unit	x-axis of 1-D histogram	y-axis of 1-D histogram
cellCharge	deposited charge in the volume	e+	n/a	n/a
cellFlux	sum of track length divided by the volume	cm^{-2}	Ek in MeV	weighted cell flux
doseDeposit	deposited dose in the volume	Gy	dose per step in Gy	track weight
energyDeposit	deposited energy in the volume	MeV	eDep per step in MeV	track weight
flatSurfaceCurrent	surface current on -z surface to be used only for Box	cm^{-2}	Ek in MeV	weighted current
flatSurfaceFlux	surface flux (1/cos(theta)) on -z surface to be used only for Box	cm^{-2}	Ek in MeV	weighted flux
nOfCollision	number of steps made by physics interaction	n/a	n/a	n/a
nOfSecondary	number od secondary tracks generated in the volume	n/a	Ek in MeV	track weight

Name of primitive scorer	Description	Default unit	x-axis of 1-D histogram	y-axis of 1-D histogram
nOfStep	number of steps in the volume	n/a	step length in mm	entry (unweighted)
nOfTerminatedTrack	numver of tracks terminated in the volume (due to decay, interaction, stop, etc.)	n/a	n/a	n/a
nOfTrack	number of tracks in the volume (including both passing and terminated tracks)	n/a	Ek in MeV	track weight
passageCellCurrent	number of tracks that pass through the volume	n/a	Ek in MeV	track weight
passageCellFlux	sum of track length divided by the volume counted only for tracks that pass through the volume	cm^{-2}	Ek in MeV	weighted cell flux
passageTrackLength	sum of track length in the volume for tracks that pass through the volume	mm	track length in mm	entry (unweighted)

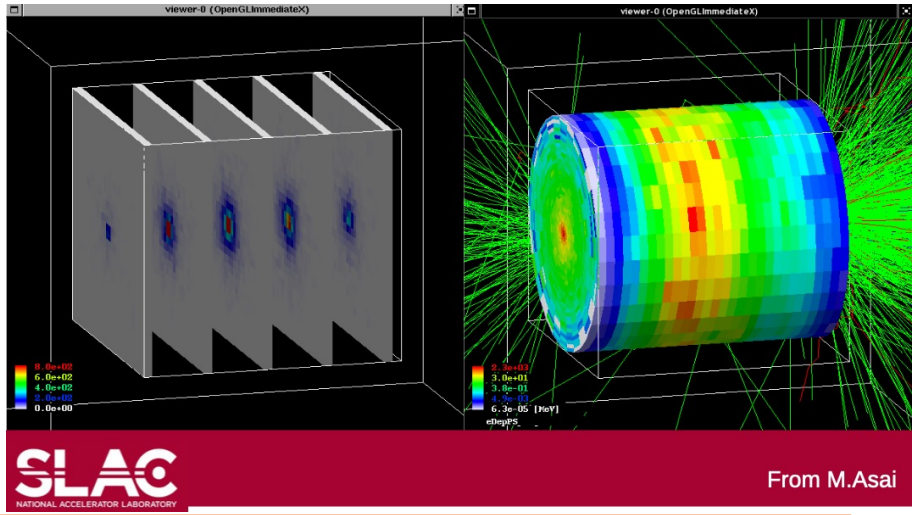
Name of primitive scorer	Description	Default unit	x-axis of 1-D histogram	y-axis of 1-D histogram
population	number of tracks in the volume that are unique in an event	n/a	n/a	n/a
trackLength	total track length in the volume (including both passing and terminated tracks)	mm	n/a	n/a
volumeFlux	number of tracks getting into the volume	n/a	Ek in MeV	track weight

Drawing a scored result

- **Drawing a projected profile**
  - /score/drawProjection {meshName} {scoreName}
- **Drawing a slice**
  - /score/drawColumn {meshName} {scoreName} {plane} {column}
    - {plane} is one of 0:x-y, 1:y-z, 2:z-x
    - /score/colorMap/setMinMax {minValue} {maxValue}
      can set the minimum and maximum values of the colormap.

Example commands

```
# Draw projection w/ default colormap
/score/drawProjection  boxMesh  eDep
#
# Draw slice in plane x-y and column = 1
/score/drawColumn  boxMesh   eDep    0    1
```



From M.Asai

Saving the scored quantities

- Following commands output the scored quantities to file
  - /score/dumpQuantityToFile {meshName} {scoreName} {fileName}
  - /score/dumpAllQuantitiesToFile {meshName} {fileName}

Example commands

```
# Save eDep in boxMesh_1 to eDep.txt
/score/dumpQuantityToFile boxMesh_1 eDep  eDep.txt
```

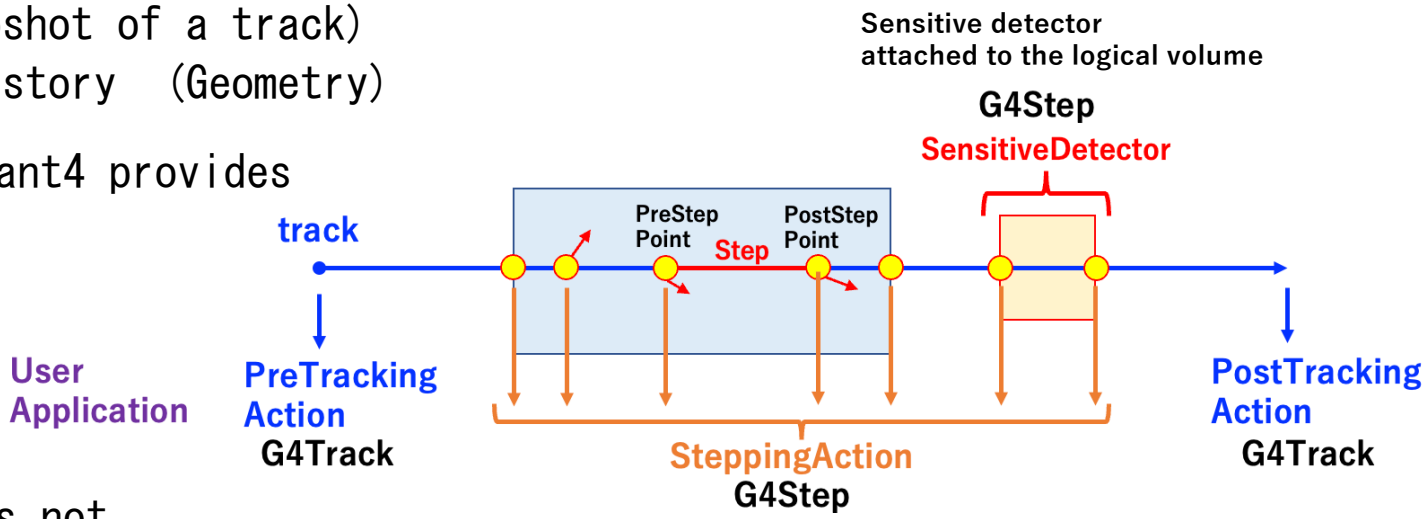
Example : eDep.txt

```
# mesh name: boxMesh_1
# primitive scorer name: eDep
# iX, iY, iZ, total(value) [MeV], total(val^2), entry
0,0,0,0,0,0
0,0,1,0,0,0
0,0,2,0,0,0
0,0,3,0,0,0
0,0,4,0,0,0
0,0,5,0,0,0

 .........

0,9,25,0.1752558887975195,0.03071462655820853,1
0,9,26,0.006387395158052346,4.079881690511056e-05,1
0,9,27,0.001942708619491945,3.7741167802483e-06,1
0,9,28,0.0311381657881858,0.0009695853686525448,1
0,9,29,0,0,0
0,10,0,0,0,0
0,10,1,0,0,0
0,10,2,0,0,0
0,10,3,0,0,0
```

Sensitive detectors, hits and hits collections

Extracting information from Geant4

- Geant4 performs a simulation using the geometry, material, particle, track information
- The user needs to access the information in Geant4 for scoring
  - Such information is stored as the Geant4 object of:
    - G4Step
      - G4StepPoint (PreStepPoint and PostStepPoint)
        - G4Track (Snapshot of a track)
        - G4TouchbaleHistory (Geometry)
- To access these objects, Geant4 provides user-action classes :
  - TrackingAction
  - SteppingAction
  - SensitiveDetector
- Scoring in SteppingAction is not recommended to avoid degradation of computational efficiency

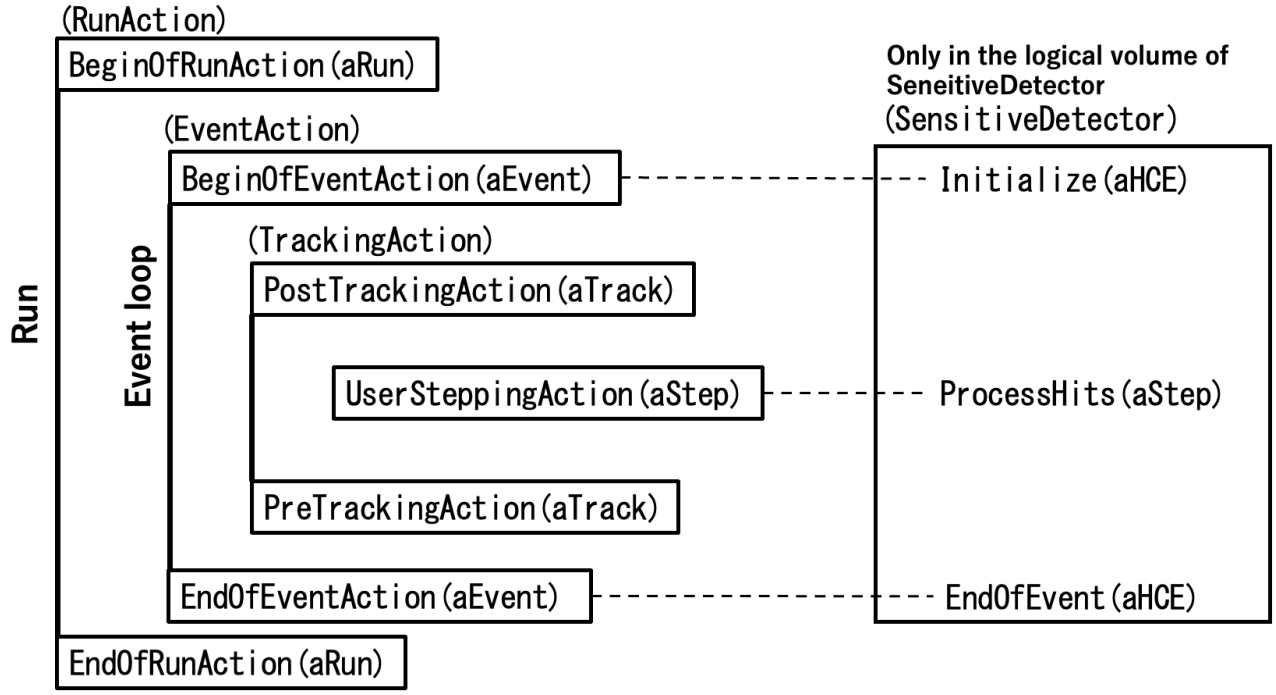


Extracting information from Geant4

- There are two more user-action classes
  - UserRunAction (G4Run)
  - UserEventAction (G4Event)

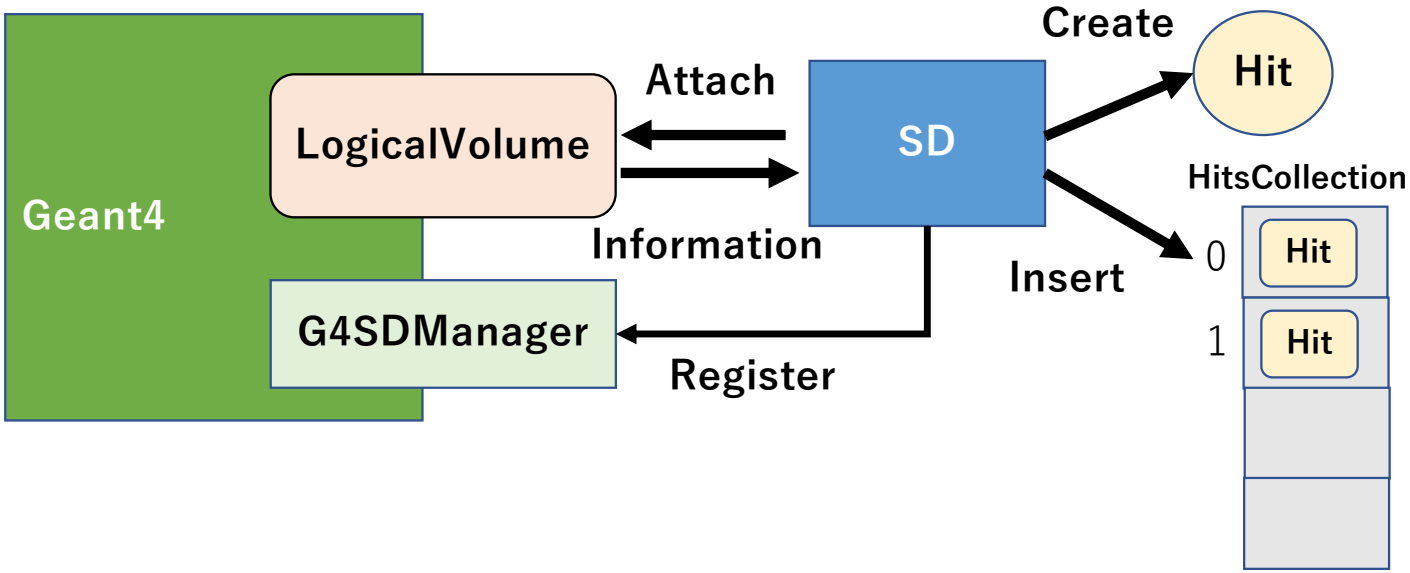
- We assume to use SensitiveDetector in the following lectures.

Flow of user-action classes



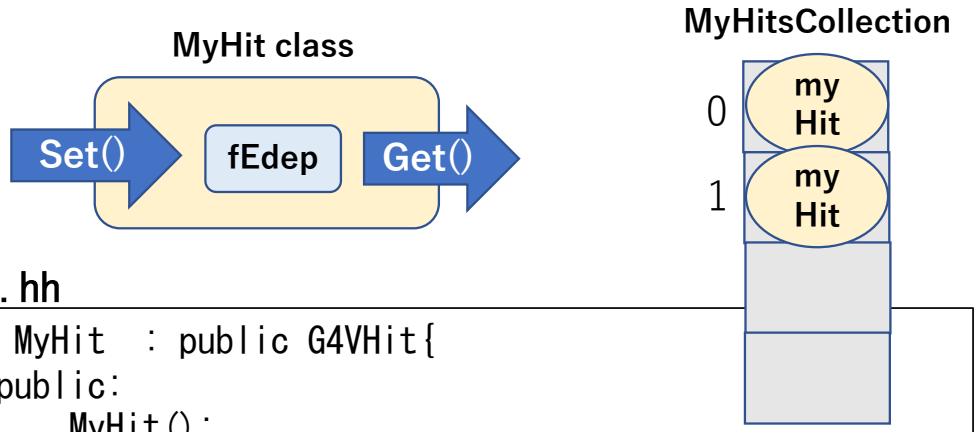
SD, Hit and HitsCollection

- SD performs to
  - Get information from Geant4 and create a hit
    - The hit is a unit of record
  - Insert the hit to HitsCollection (HC)
    - The HC is an array of Hits as a temporal storage



Hit and HitsCollection

- Hit class
  - The user is required to implement Hit class on the base class of G4VHit that should has:
    - Data members of quantities
    - Access methods of SetXXX() and GetXXX().
    - new and delete operator using G4Allocator class which handles efficient memory allocation and de-allocation of Hit objects
- HitsCollection
  - Defined with the Hit class



MyHit.hh

```
class MyHit  : public G4VHit{
     public:
         MyHit();
         ~MyHit() override = default;
         inline void * operator new(size_t);
         inline void operator delete(void * aHit);
         //
        void  SetEdep(G4double edep ) { fEdep = edep; };
        G4double GetEdep() const {  return fEdep; };
     private:
        G4double fEdep;
};
Using MyHitsCollection = G4THitsCollection<MyHit>;
…
```

SD sample implementation

MySD.cc

```
void MySD::Initialize (G4HCofThisEvent* HCE) {
    // Create HC
    fHitsCollection = new MyHitsCollection(SDname, collectionName[0]);
    // Make a link between HC and HCE (HC of this Event)
    HCID = G4SDManager::GetSDManager()->GetCollectionID(collectionName[0]);
    HCE->AddHitsCollection(HCID, fHitsCollection); // Register to HCE
}
void MySD::ProcessHits(G4Step* step, G4TouchableHistory* /*history*/) {
    // Create a hit
    MyHit* newHit = new MyHit();
    // Set some properties to the hit
    newHit->SetEdep(step->GetEnergyDeposit());
    // Add the hit in the SD hits collection
    fHitsCollection->insert(newHit);
}
void MySD::EndOfEvent(G4HCofThisEvent* HCE){
}
```

SDname and collectionName[0] must be unique names.

HCID is the identification number of the HC.

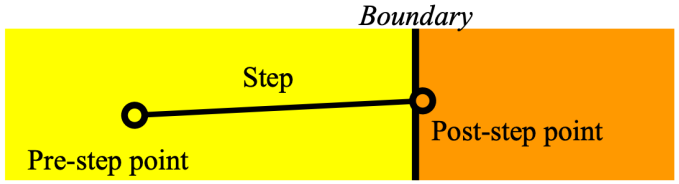
HCE is a data member of G4Event object so that it can access form the EventAction()

The HC object is deleted at the end of event automatically.

What sort of information is available from Geant4??

- G4Step

	G4Step ()
	~G4Step ()
	G4Step (const G4Step &)
G4Step &	operator= (const G4Step &)
G4Track *	GetTrack () const
void	SetTrack (G4Track *value)
G4StepPoint *	GetPreStepPoint () const
void	SetPreStepPoint (G4StepPoint *value)
G4StepPoint *	GetPostStepPoint () const
void	SetPostStepPoint (G4StepPoint *value)
G4double	GetStepLength () const
void	SetStepLength (G4double value)
G4double	GetTotalEnergyDeposit () const
void	SetTotalEnergyDeposit (G4double value)
G4double	GetNonIonizingEnergyDeposit () const
void	SetNonIonizingEnergyDeposit (G4double value)
G4SteppingControl	GetControlFlag () const
void	SetControlFlag (G4SteppingControl StepControlFlag)
void	AddTotalEnergyDeposit (G4double value)
void	ResetTotalEnergyDeposit ()
void	AddNonIonizingEnergyDeposit (G4double value)
void	ResetNonIonizingEnergyDeposit ()
G4bool	IsFirstStepInVolume () const
G4bool	IsLastStepInVolume () const
void	SetFirstStepFlag ()
void	ClearFirstStepFlag ()
void	SetLastStepFlag ()
void	ClearLastStepFlag ()
G4ThreeVector	GetDeltaPosition () const
G4double	GetDeltaTime () const
G4ThreeVector	GetDeltaMomentum () const
G4double	GetDeltaEnergy () const
void	InitializeStep (G4Track *aValue)
void	UpdateTrack ()
void	CopyPostToPreStepPoint ()
G4Polyline *	CreatePolyline () const
const std::vector< const G4Track * > *	GetSecondaryInCurrentStep () const
const G4TrackVector *	GetSecondary () const
G4TrackVector *	GetfSecondary ()
G4TrackVector *	NewSecondaryVector ()
void	DeleteSecondaryVector ()
void	SetSecondary (G4TrackVector *value)
void	SetPointerToVectorOfAuxiliaryPoints (std::vector< G4ThreeVector > *theNewVectorPointer)
std::vector< G4ThreeVector > *	GetPointerToVectorOfAuxiliaryPoints () const



- G4StepPoint

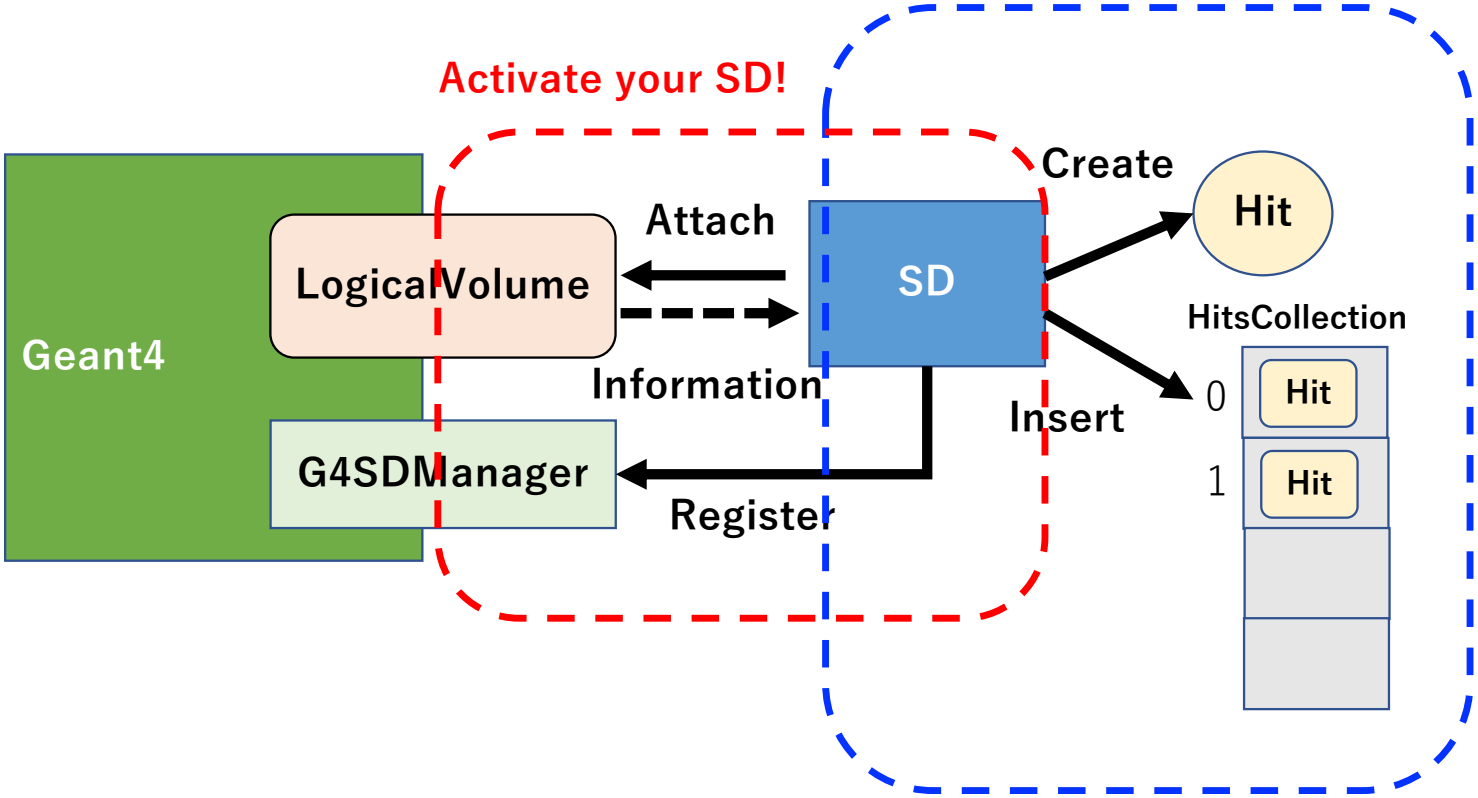
	G4StepPoint ()
	~G4StepPoint ()
	G4StepPoint (const G4StepPoint &)
G4StepPoint &	operator= (const G4StepPoint &)
const G4ThreeVector &	GetPosition () const
void	SetPosition (const G4ThreeVector &aValue)
void	AddPosition (const G4ThreeVector &aValue)
G4double	GetLocalTime () const
void	SetLocalTime (const G4double aValue)
void	AddLocalTime (const G4double aValue)
G4double	GetGlobalTime () const
void	SetGlobalTime (const G4double aValue)
void	AddGlobalTime (const G4double aValue)
G4double	GetProperTime () const
void	SetProperTime (const G4double aValue)
void	AddProperTime (const G4double aValue)
const G4ThreeVector &	GetMomentumDirection () const
void	SetMomentumDirection (const G4ThreeVector &aValue)
void	AddMomentumDirection (const G4ThreeVector &aValue)
G4ThreeVector	GetMomentum () const
G4double	GetTotalEnergy () const
G4double	GetKineticEnergy () const
void	SetKineticEnergy (const G4double aValue)
void	AddKineticEnergy (const G4double aValue)
G4double	GetVelocity () const
void	SetVelocity (G4double v)
G4double	GetBeta () const
G4double	GetGamma () const
G4VPhysicalVolume *	GetPhysicalVolume () const
const G4VTouchable *	GetTouchable () const
const G4TouchableHandle &	GetTouchableHandle () const
void	SetTouchableHandle (const G4TouchableHandle &apValue)
G4Material *	GetMaterial () const
void	SetMaterial (G4Material *)
const G4MaterialCutsCouple *	GetMaterialCutsCouple () const
void	SetMaterialCutsCouple (const G4MaterialCutsCouple *)
G4VSensitiveDetector *	GetSensitiveDetector () const
void	SetSensitiveDetector (G4VSensitiveDetector *)
G4double	GetSafety () const
void	SetSafety (const G4double aValue)
const G4ThreeVector &	GetPolarization () const
void	SetPolarization (const G4ThreeVector &aValue)
void	AddPolarization (const G4ThreeVector &aValue)
G4StepStatus	GetStepStatus () const
void	SetStepStatus (const G4StepStatus aValue)
const G4VProcess *	GetProcessDefinedStep () const
void	SetProcessDefinedStep (const G4VProcess *aValue)
G4double	GetMass () const
void	SetMass (G4double value)
G4double	GetCharge () const
void	SetCharge (G4double value)
G4double	GetMagneticMoment () const
void	SetMagneticMoment (G4double value)
void	SetWeight (G4double aValue)
G4double	GetWeight () const

- G4TouchableHistory

	G4TouchableHistory (const G4NavigationHistory &history)
	G4TouchableHistory ()
	~G4TouchableHistory ()
G4VPhysicalVolume *	GetVolume (G4int depth=0) const
G4VSolid *	GetSolid (G4int depth=0) const
const G4ThreeVector &	GetTranslation (G4int depth=0) const
const G4RotationMatrix *	GetRotation (G4int depth=0) const
G4int	GetReplicaNumber (G4int depth=0) const
G4int	GetHistoryDepth () const
G4int	MoveUpHistory (G4int num_levels=1)
void	UpdateYourself (G4VPhysicalVolume *pPhysVol, const G4NavigationHistory *history=0)
const G4NavigationHistory *	GetHistory () const
void *	operator new (size_t)
void	operator delete (void *aTH)

```
G4Step* aStep;
G4StepPoint* preStepPoint =
                    aStep->GetPreStepPoint();
G4StepPoint* postStepPoint =
                    aStep->GetPostStepPoint();
G4TouchableHistory* theTouchable =
  (G4TouchbaleHistory*)(preStepPoint->GetTouchable());
G4int copyNo =
          theTouchable->GetVolume()->GetCopyNo();
G4int motherCopyNo =
          theTouchable->GetVolume(1)->GetCopyNo();
```

Now you are ready to activate your SD !



Activate your SensitiveDetector (SD)

- Let’s call your SD as MySD for example
- The MySD must be instantiated in DetectorConstruction::ConstructSDandField() method.

DetectorConstuction.cc

```
#include “MySD.h”
…
void DetectorConstuction::ConstructSDandField(){
    //
    // Instantiate SD with a unique name
    auto mySD = new MySD(” MySD” );
    //
    // Register the SD to G4SDManager in Geant4.
    G4SDManager::GetSDMpointer()->AddNewDetector(mySD);
    //
    // Attach the SD object to the logical volume by the logical volume name (e.g. lvname)
    G4String lvName = ” voxel” ;
    SetSensitiveDetector( lvName, mySD);
}
```

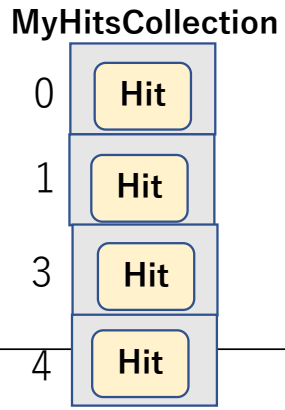
How do we access to the HC in EventAction

- We can access to the HC in the EventAction through the G4Event object and the G4HCofThisEvent object.

EventAction.cc

```
void  EventAction::EndOfEventAction (G4Event* aEvent) {
      //
      // Get HCID from G4SDManager by its SD name and collection name.
      // Then get HC through G4HCofThisEvent object from G4Event object.
      fHCID = G4SDManager::GetSDMpointer->GetCollectionID(" MySD/HitsCollection" );
      auto HC  = static_cast<MyHitsCollection*>(aEvent->GetHCofThisEvent()->GetHC(fHCID));

      // Loop over the hits
     for (size_t  I = 0;  I < HC->entries() ; i++){
          MyHit*  Hit = (*HC)[i];
           ….
      }
}
```

"SDname/HCname"



You also calculate physical quantities from the value in hits and make an output file here.

Analysis

- How to analyze using Hits
- How to save the results

How to manage the results using G4AnalysisManager

- G4AnalysisManager provides:
  - Create or get the instance
  - Define and fill histograms
  - Define and fill ntuples
    - ntuples is a sequence of n elements (quantities), that may be of different types.
  - Outout the histograms and ntuples to file in the ROOT, XML, CSV file formats
  - (Supporting merging-options in multi-threading environment)

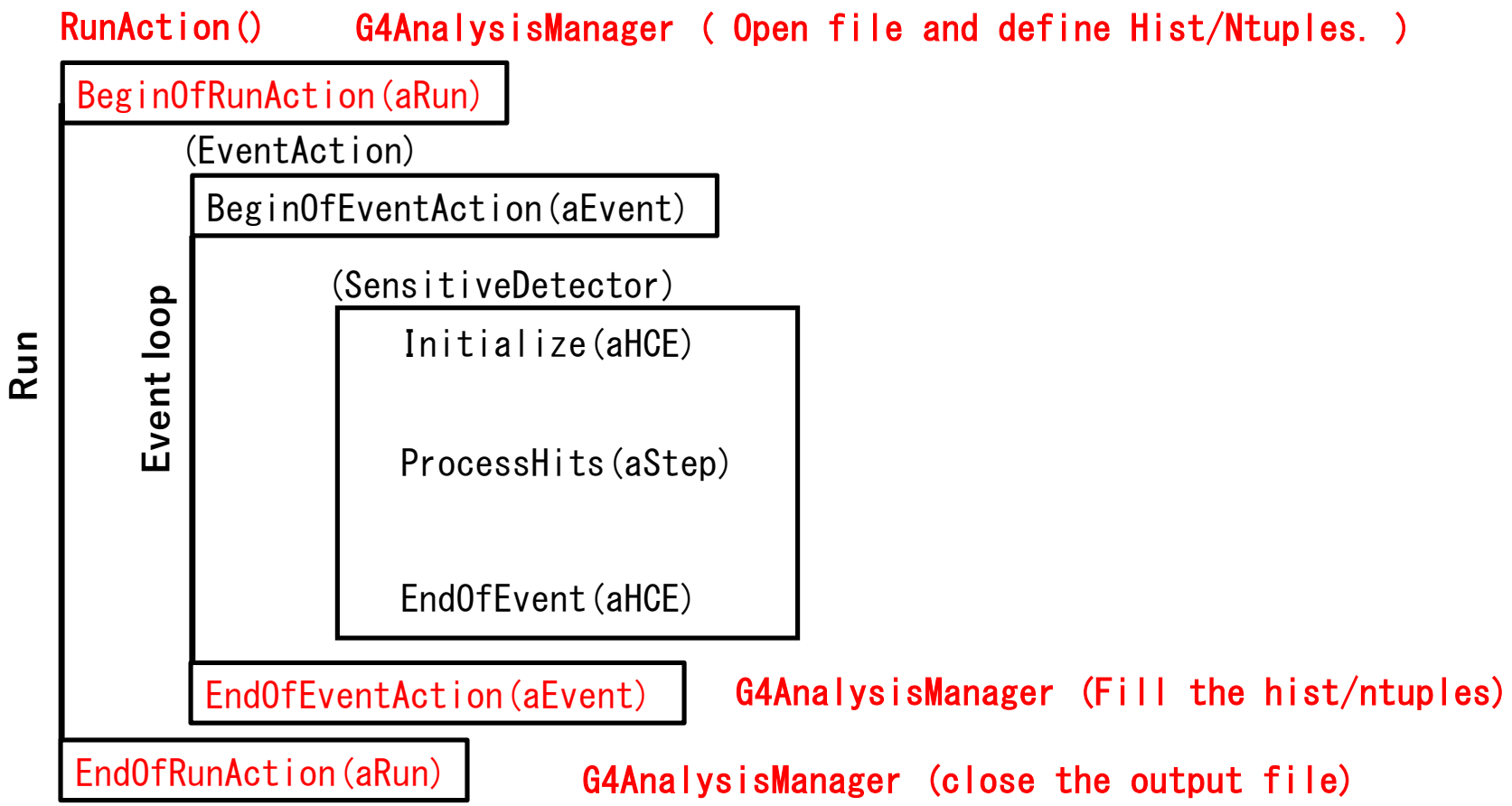
```
#include "G4AnalysisManager.hh"

// Create analysis manager
auto analysisManager = G4AnalysisManager::Instance();
analysisManager->SetVerboseLevel(1);

// Book histograms, ntuple
analysisManager->CreateH1("Eabs","Edep in absorber", 100, 0., 800*MeV);
analysisManager->CreateH1("Egap","Edep in gap", 100, 0., 100*MeV);
//
analysisManager->CreateNtuple("B4", "Edep and TrackL");
analysisManager->CreateNtupleDColumn("Eabs");
analysisManager->CreateNtupleDColumn("Egap");
analysisManager->FinishNtuple();

// Open an output file
G4AnalysisManager::Instance()->OpenFile("B4.root");

// Fill histograms, ntuple
analysisManager->FillH1(0, value);
analysisManager->FillH1(1, value);
analysisManager->FillNtupleDColumn(0, eabs);
analysisManager->FillNtupleDColumn(1, egap);
AnalysisManager->AddNtupleRow();

// Save histograms & ntuple
analysisManager->Write();
analysisManager->CloseFile();
```

Example of ntuple

SEQ	Eabs	Egap
0	10.2	0.1
1	8.9	0.5

Code extracted from Geant4 basic example B4

G4AnalysisManager and Geant4 user-actions



1)RunAction::RunAction()
Create Analysis Manager and book histograms and ntuples

RunAction.cc (Create AnalysisManager)

```
#include  "G4AnalysisManager.hh"
RunAction::RunAction(){  // Constructor
 //
 auto analysisManager = G4AnalysisManager::Instance();//Create the instance
 analysisManager->SetNtupleMerging(true);  //Optional.(Default "false")
 // 1-D,2-D,3-D histograms are automatically merged at the end of Run in MT mode.
 // By default, Ntuples are not merged. Those are store in separated files.
 //
 analysisManager->SetVerboseLevel(1);
 analysisManager->SetFirstH1Id(100); //HistID of 1-D starts from 100. Optional.
 analysisManager->SetFirstH2Id(200); //HistID of 2-D starts from 200. Optional.
 analysisManager->SetFirstH3Id(300); //HistID of 3-D starts from 300. Optional.
 analysisManager->SetFirstNtupleId(1000); // HistID of ntuple starts from 1000.
                                          // Optional.

  // Defining histograms here … See next slide
}
```

1)RunAction::RunAction()
Create Analysis Manager and book histograms and ntuples

RunAction.cc (Create histograms)

```
#include   "G4AnalysisManager.hh"

RunAction::RunAction(){ //Constructor
  // ---- snippet

  // Create 1-D histogram   HIDs are automatically incremented by a definition for each category
  G4int h1Id = 0;
  h1Id = analysisManager->CreateH1("H1_1","Depth-dose in iz", 150, 0., 150);
  // (h1Id = 100)                    name,    Title            , nbins, xmin, ymin
  G4int h2Id = 0;
  h2Id = analysisManager->CreateH2("H2_1","x-y dose at iz=0",
                                   150, -150., 150, 150, -150., 150.);
  //  (h2Id = 200)   CreateH2(name, title, nbinsx, xmin, xmax, nbinsy, ymin, ymax);
  // The HIDs are requested in filling process. Therefore, you need to remember the HIDs.

  //   Ntuple See next slide.
}
```

1)RunAction::RunAction()
Create Analysis Manager and book histograms and ntuples

- The ntuple is a sequence of n elements (quantities), that may be of different types.

RunAction.cc (Create Ntuple)

```
#include   "G4AnalysisManager.hh"

RunAction::RunAction(){  //Constructor
  // ---- snippet

  //   Ntuple.
  G4int ntupleId = analysisManager->CreateNtuple(" MyNtuple", " VoxelID and Edep");
  //                                                  name          Title
  analysisManager->CreateNtupleIColumn(ntupleId,"ix"); //ColumnId=0 (ntupleId," name" )
  analysisManager->CreateNtupleIColumn(ntupleId,"iy"); //ColumnId=1
  analysisManager->CreateNtupleIColumn(ntupleId,"iz"); //ColumnId=2
  analysisManager->CreateNtupleFColumn(ntupleId,"de"); //ColumnId=3
  analysisManager->FinishNtuple(ntupleId);  // End of ntuple definition.

}
```

Column-id is automatically incremented and assigned.
You also need to remember the columnId to fill the ntuple.

CreateNtupleDColumn() for double value, and
CreateNtupleSColumn() for string are also available.

(2) RunAction::BeginOfRunAction() and EndOfRunAction
Open, write and close an output file

RunAction.cc（Open file and Write/close the file）

```
#include "G4AnalysisManager.hh"

void RunAction::BeginOfRunAction(const G4Run* run) {
   // Get analysis manager
   auto analysisManager = G4AnalysisManager::Instance(); //Get AnalysisManager
   // Open an output file
   analysisManager->OpenFile("MyFile.root");          // The extension of file
determine the file format.
}
….
void RunAction::EndOfRunAction(const G4Run* run) {
   // Get analysis manager
   auto analysisManager = G4AnalysisManager::Instance(); //Get AnalysisManager
   // Write and close the output file
   analysisManager->Write();
   analysisManager->CloseFile();
}
```

```
w/o SetNtupleMerging(true),
each thread saves ntuples to separate files.
e.g. MyFile_t0.root, MyFile_t1.root …
```

(3) EventAction::EndofEventAction()
Fill histograms and nutples

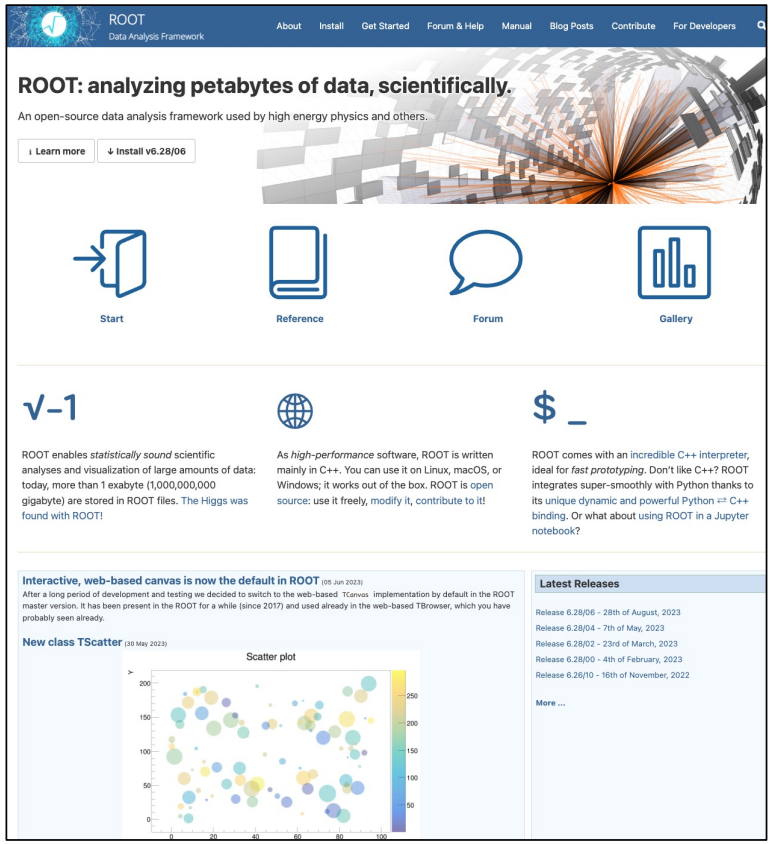
EventAction.cc

```
#include "G4AnalysisManager.hh"

void EventAction::EndOfEventAction(const G4Run* run) {
   // Get analysis manager
   auto analysisManager = G4AnalysisManager::Instance();
  // …snippet (See 18 page, Loop over the Hits. )
  analysisManager->FillH1(100, fEdep );   // HID=100 is for energy deposit.
  //                                        HID,  value
  analysisManager->FillH2(200, ix, iy, fDose);   // HID=100 is for energy deposit.
  //                                        HID,  xvalue, yvalue,  weight
  //                     ntupleId = 1000, Columnid, value
 analysisManager->FillNtupleIColumn(1000, 0,ix);
 analysisManager->FillNtupleIColumn(1000, 1,iy);   // ntupleId = 1000, Columnid =1,  int-value
 analysisManager->FillNtupleIColumn(1000, 2,iz);   // ntupleId = 1000, Columnid =2,  int-value
 analysisManager->FillNtupleFColumn(1000, 3,fEdep);// ntupleId = 1000, Columnid =3,  float-value
 analysisManager->AddNtupleRow(1000); // Add to ntupleId = 1000.
}
```

Analysis using ROOT (Minimum lecture)

- Analysis of output files with external tool of ROOT.
  - ROOT, https://root.cern/

- We use the ROOT in this tutorial, to check the histograms and ntuples in the output file.



The minimum commands that you should know for the hands-on

- Start a ROOT session and open a ROOT output file.

  $ root MyFile.root

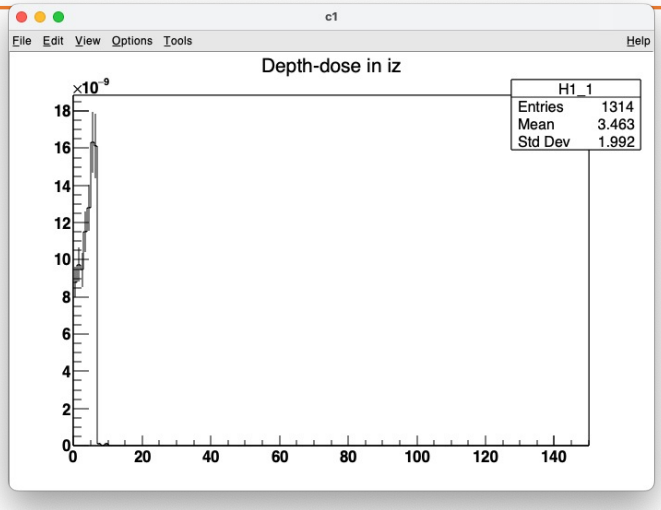
- Check the list of histograms and ntuples

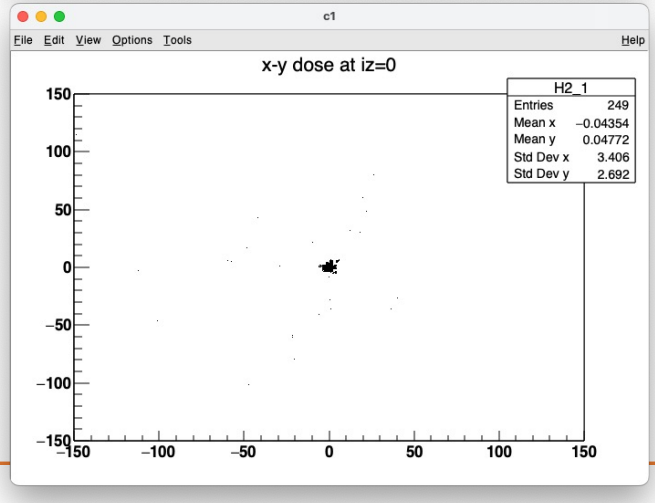
  root[] .ls

```
root[] .ls
TFile** MyFile.root
 TFile* MyFile.root
  KEY: TTree MyNtuple;1 VoxelID and Edep
  KEY: TH1D H1_1;1 Depth-dose in iz
  KEY: TH2D H2_1;1 x-y dose at iz=0
```

Types Name

- Draw 1D or 2D histogram
  - root[] H1_1->Draw() or H1_1->Draw(“h”)
  - root[] H2_1->Draw()





The minimum commands that you should know for the hands-on

- Check quantities in the ntuple
  - root[] MyNtuple->Print()
- Check the values
  - root[] MyNtuple->Scan()
    - Scroll [space]
    - quit [q]

MyNtuple->Scan()

```
*************************************************************
*    Row   *        ix *        iy *        iz *        de *
*************************************************************
*        0 *         4 *         5 *         0 * 17968.398 *
*        1 *         4 *         5 *         1 * 18515.611 *
*        2 *         4 *         5 *         2 * 22017.998 *
*        3 *         4 *         5 *         3 * 16309.317 *
*        4 *         4 *         5 *         3 * 8326.9472 *
*        5 *         4 *         5 *         3 * 1101.8018 *
*        6 *         4 *         5 *         3 * 1578.2528 *
*        7 *         4 *         5 *         3 * 22901.800 *
*        8 *         4 *         5 *         3 * 6590.2241 *
*        9 *         4 *         5 *         3 * 8972.4863 *
```

MyNtuple->Print()

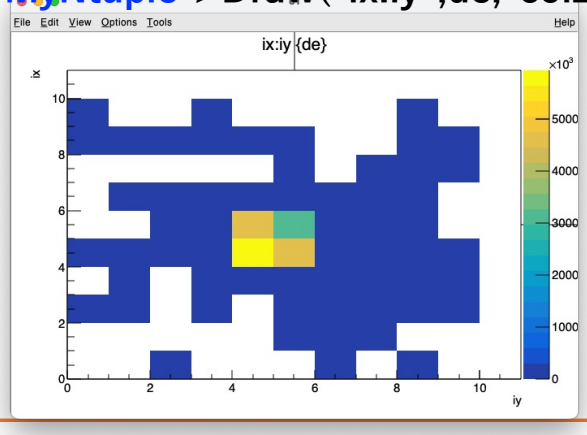
```
******************************************************************************
*Tree    :MyNtuple    : VoxelID and Edep                                     *
*Entries :     1314 : Total =           88955 bytes  File  Size =      36581 *
*        :          : Tree compression factor =   2.38                       *
******************************************************************************
*Br    1 :ix        : Int_t Galet                                            *
*Entries :     1314 : Total  Size=      11072 bytes  File Size  =       2556 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   4.14     *
*............................................................................*
*Br    2 :iy        : Int_t Galet                                            *
*Entries :     1314 : Total  Size=      11072 bytes  File Size  =       2516 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   4.21     *
*............................................................................*
*Br    3 :iz        : Int_t Galet                                            *
*Entries :     1314 : Total  Size=      11072 bytes  File Size  =       2800 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   3.78     *
*............................................................................*
*Br    4 :de        : Float_t Galet                                          *
*Entries :     1314 : Total  Size=      11072 bytes  File Size  =       7008 *
*Baskets :        1 : Basket Size=      32000 bytes  Compression=   1.51     *
*............................................................................*
```

The minimum commands that you should know for the hands-on

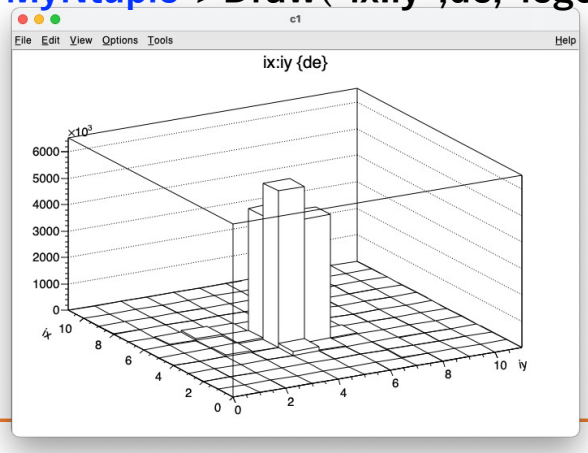
- Draw iz vs. sum(de)
  - root[] MyNtuple->Draw(“iz” , de, “h”)
    - Horizontal, vertical weight, option
- Draw sum(de) in x-y plane
  - root[] MyNtuple->Draw(“ix:iy” ,de,” colz”)
    - X : Y , weight, option
  - root[] MyNtuple->Draw(“ix:iy” ,de,” lego”)

- Quit the ROOT
  - root[] .q
  - $

MyNtuple->Draw(“ix:iy”,de,”colz”)



MyNtuple->Draw(“ix:iy”,de,”lego”)



Summary

- Usage of command-based scorer
  - Easiest way of scoring if the scorer is available.
- Sensitive detector, Hit and HitsCollection
  - SD corrects information from G4step, G4StepPoint and G4TouchableHistory
  - Hit is a unit of information to store
  - HitsCollection is an array of hits
  - The user can access to Hits in EventAction::EndOfEventAction() through Event -> HCofThisEvent -> HitsCollection -> Hit.
- G4AnalysisManager
  - Manage histograms and ntuples
    - Define histograms/ntuples (RunAction::RunAction())
    - Open file (RunAction::BeginOfRunAction())
    - Fill histograms/ntuples (EventAction::EndOfEventAction())
    - Output to file (RunAction::EndOfRunAction())