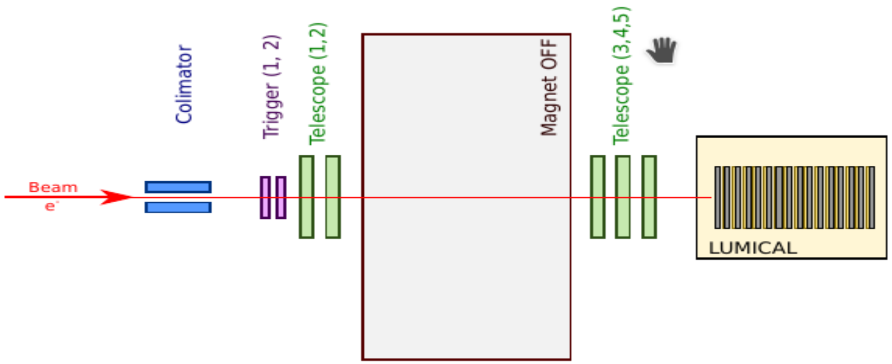
# Overview on simulation implementation for the testbeam 2020

Alina – Tania Neagu
Mihai Potlog

# Outline

- Testbeam setup:
  
  setup description, Lumical configuration,
  
  code check: geometry construction and overlaping
- Data collection, .root file structure
- Results

# Testbeam setup



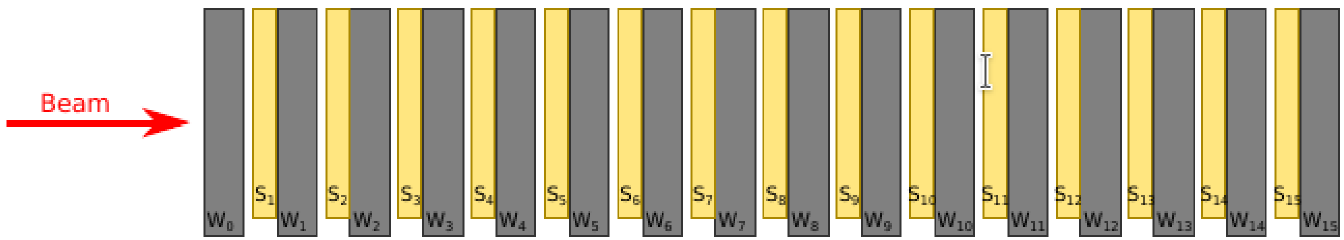

**➢ Regular configuration:**

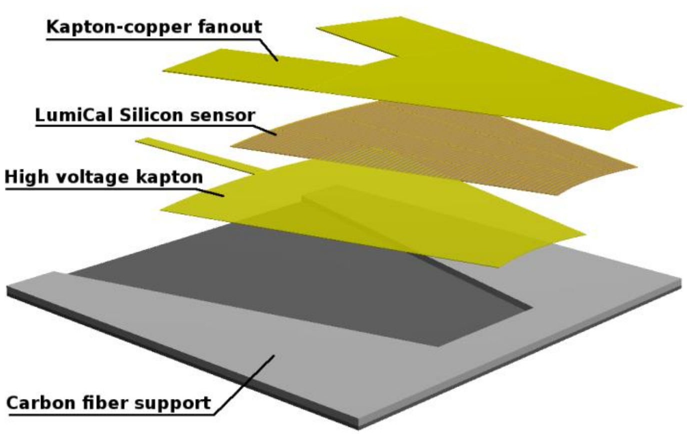
- Beam spot after the colimator ~5mm x 5mm
- Two scintilator triggers operating in coincidence mode
- 5 telescope planes – 2 before and 3 after the magnet
- Magnet switched OFF
- LumiCal placed on movable table

**➢ LumiCal configuration:**

✓ 15 sensor layers (S1 - S15) glued to tungsten absorbers (W1 – W15) ->config. A

✓ Additional tungsten layer in front of the stack (W0) -> config. AA




**➢ Sensor components:**

✓ kapton fan-out

✓ LumiCal sensor

✓ high-voltage kapton

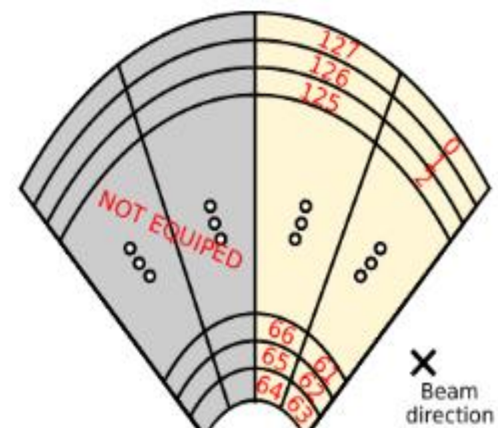
✓ carbon fiber support

**➢ Sensor position:**

✓ only sectors R1 and R2 are equiped

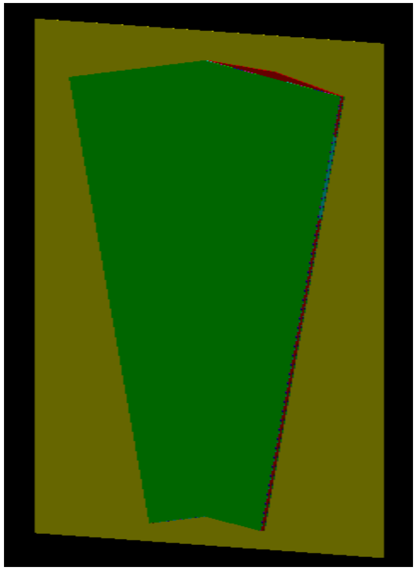
✓

✓

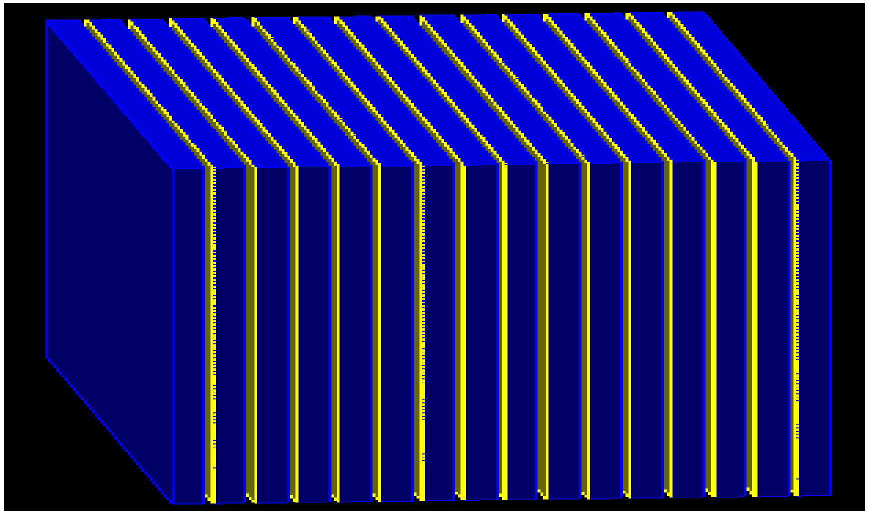


# Testbeam setup: visualisation

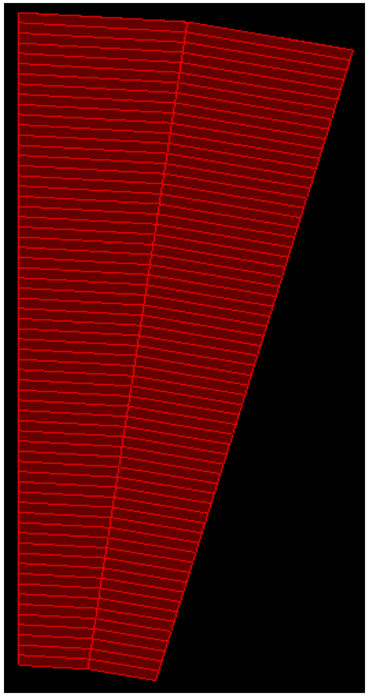
Geometry construction for testbeam configurations



Lumical sensor components



Lumical detector configuration A (15 sensors)



sectors R1 and R2
64 pads each

FCAL Collaboration High precision design

# Geometry construction

INSTITUTE OF SPACE SCIENCE www.spacescience.ro

CURRENT SITUATION

*Geometry construction for testbeam configurations*

- Derive own concrete class from G4VUserDetectorConstruction abstract base class
- Implement the method Construct()

1. construct all necessary materials

```
// Absorber
G4Material* W_abs93 = new G4Material(name="Abs_93", density = 17.7*g/cm3, ncomponents=3);
W_abs93->AddMaterial(cu, 1.75*perCent);
W_abs93->AddMaterial(ni, 5.25*perCent);
W_abs93->AddMaterial(W,  93.*perCent);

// Epoxy
G4Material* Epoxy = new G4Material("Epoxy" , density= 1.3*g/cm3, ncomponents=3);
Epoxy->AddElement(elH, natoms=44);
Epoxy->AddElement(elC, natoms=15);
Epoxy->AddElement(elO, natoms=7);

// Carbon fiber
G4Material* CarbonFiber = new G4Material(name="CarbonFiber", density = 1.6*g/cm3, ncomponents=2);
CarbonFiber->AddMaterial(C,     50.*perCent);
CarbonFiber->AddMaterial(Epoxy, 50.*perCent);
```

2. define shapes, logical volumes and position of the experimental hall

```
//================================== experimental hall ==============================
G4double worldX = 2500.0*mm;
G4double worldY = 2500.0*mm;
G4double worldZ = 20500.0*mm;
G4Box* world = new G4Box("world",worldX,worldY,worldZ);
   worldLog  = new G4LogicalVolume(world,Air,"worldLog",0,0,0);
   worldPhys = new G4PVPlacement(0,G4ThreeVector(),worldLog,"worldPhys",0,false,0);
```

3. define shapes, logical volumes and position of the coverage of LumiCal – Al foil

```
//================================== Al foil - Lumical coverage ========================================
G4double foilAlX = 50.*mm;
G4double foilAlY = 70.*mm;
G4double wfoilAl = 0.05*mm;

G4Box* Alfoil = new G4Box("Alfoil",foilAlX,foilAlY,wfoilAl);
    AlfoilLog = new G4LogicalVolume(Alfoil,Al,"AlfoilLog");

// distance from Absorber placed in front of sensor to Al foil covering the Lumical
G4double dAbsAlfoil = (wSlot + dSlotfAl)*mm;              // = 62.5 mm
   AlfoilPhys = new G4PVPlacement(0, G4ThreeVector(translateX,0,-(dAbsAlfoil + wfoilAl/2)*mm), AlfoilLog,
                                  "alfoilPhys", worldLog, false, 0, fCheckOverlaps);
```

# Geometry construction

CURRENT SITUATION

4. define shapes, logical volumes and position of the telescope

```
//=========================== Telescope - 2 before and 3 after magnet ===============================
// telescope plan dimensions
G4double telescopeX = 10.6*mm;
G4double telescopeY = 5.3*mm;
G4double telescopeZ = 0.05*mm;

G4Box* telescope = new G4Box("telescope",telescopeX,telescopeY,telescopeZ);

// distances from telescope planes to the Lumical coverage
G4double dLumTel5  = 364*mm;     // distance from Lumical to Telescope plan 5
G4double dTel5Tel4 = 90*mm;      // distance from Telescope plan 5 to Telescope plan 4
G4double dTel4Tel3 = 89*mm;      // distance from Telescope plan 4 to Telescope plan 3
G4double dTel3Tel2 = 2167*mm;    // distance from Telescope plan 3 to Telescope plan 2
G4double dTel2Tel1 = 173*mm;     // distance from Telescope plan 2 to Telescope plan 1

// distance from Absorber placed in front of sensor to Telescope plan 5
G4double dAbsTel5 = (dAbsAlfoil + wfoilAl + dLumTel5)*mm;

// positions of Telescope planes
G4double ZTelPlan5 = - (dAbsTel5 + telescopeZ/2)*mm;
G4double ZTelPlan4 = - (dAbsTel5 + dTel5Tel4 + telescopeZ)*mm;
G4double ZTelPlan3 = - (dAbsTel5 + dTel5Tel4 + dTel4Tel3 + (2*telescopeZ) + telescopeZ/2)*mm;
G4double ZTelPlan2 = - (dAbsTel5 + dTel5Tel4 + dTel4Tel3 + dTel3Tel2 + (3*telescopeZ) + telescopeZ/2 )*mm;
G4double ZTelPlan1 = - (dAbsTel5 + dTel5Tel4 + dTel4Tel3 + dTel3Tel2 + dTel2Tel1 + (4*telescopeZ) + telescopeZ/2)*mm;

G4double ZTelPlans[5] = {ZTelPlan1, ZTelPlan2, ZTelPlan3, ZTelPlan4, ZTelPlan5} ;

for( int b = 0; b < 5; b++ ){
   teleplanLog[b] = new G4LogicalVolume(telescope, si, "teleplanLog[b]");
     teleplanPhys = new G4PVPlacement(0, G4ThreeVector(0, 0, ZTelPlans[b]*mm), teleplanLog[b], "teleplanPhys",
                                      worldLog, false, b, fCheckOverlaps);
}
```

5. define shapes, logical volumes and placed the scintillators

```
//================================= Trigger - 2 scintillators =====================================
// scintillator dimensions
G4double scintillatorX = 15.*mm;
G4double scintillatorY = 15.*mm;
G4double scintillatorZ = 2.064*mm;            // 14.mm; - 2 dimensions used in previous experiments

G4Box* scintplan = new G4Box("scintplan",scintillatorX,scintillatorY,scintillatorZ);

// distances from scintillators to telescope planes
G4double dScint2Tel1 = 30*mm;
G4double dScint2Scint1 = 40*mm;

G4double Zscint2 = - (-ZTelPlan1 + telescopeZ/2 + dScint2Tel1 + scintillatorZ/2)*mm;
G4double Zscint1 = - (-ZTelPlan1 + telescopeZ/2 + dScint2Tel1 + dScint2Scint1 + scintillatorZ + scintillatorZ/2)*mm;

   scintplanLog = new G4LogicalVolume(scintplan, Polyvinyltoluene," scintplanLog");
  scintplanPhys1 = new G4PVPlacement(0, G4ThreeVector(0.*mm, 0, Zscint1*mm), scintplanLog, "scintplanPhys1", worldLog, false, 1, fCheckOverlaps);
  scintplanPhys2 = new G4PVPlacement(0, G4ThreeVector(0.*mm, 0, Zscint2*mm), scintplanLog, "scintplanPhys2", worldLog, false, 2, fCheckOverlaps);
```

6. define shapes, logical volumes and placed the tungsten absorbers

```
//***************************************** Tungsten absorbers *******************************************
G4double absorberX = 65 *mm;
G4double absorberY = 65 *mm;

G4double ZAbsorber[16] = { 3.470*mm, 3.542*mm, 3.505*mm, 3.490*mm, 3.584*mm, 3.521*mm, 3.645*mm, 3.470*mm,
    3.550*mm, 3.558*mm, 3.543*mm, 3.543*mm, 3.550*mm, 3.550*mm, 3.528*mm, 3.52*mm };
    // each absorber plate has a different thickness

for( int k = 0; k < 16; k++ ){
   absorberSolid[k] = new G4Box("Absorber",absorberX,absorberY,ZAbsorber[k]/2);
   absorberLog[k] = new G4LogicalVolume(absorberSolid[k], W_abs95,"absorberLog[k]");

   G4double wW0 = (k*wSlot+wSiSensorAll + ZAbsorber[k]/2 )*mm;

   absorberPhys[k] = new G4PVPlacement(0, G4ThreeVector(translateX, tranTY, wW0), absorberLog[k],
                                       "absorberPhys", worldLog, false, k, fCheckOverlaps);
}
```

# Geometry construction

CURRENT SITUATION

7. define shapes and logical volumes of silicon sensors with all their components starting with kapton fan-out

```
//***************************************** Silicon sensors ******************************
//-------------------------------------- kapton fan-out --------------------------------
G4double KRint = 80.*mm;
G4double KRext = 195.2*mm;
G4double KstartAngle = (90.*deg - 15.*deg);
G4double KtotAngle = 30.*deg;

// polyimide 1
G4Tubs* stratK1 = new G4Tubs("stratK1", KRint, KRext, wK1/2, KstartAngle, KtotAngle);
    stratK1_log = new G4LogicalVolume(stratK1, Kapton, "K1log");
// adhesive 1
G4Tubs* stratAd1 = new G4Tubs("stratAd1", KRint, KRext, wAd1/2, KstartAngle, KtotAngle);
    stratAd1_log = new G4LogicalVolume(stratAd1, Epoxy, "Ad1log");
// copper 1
G4Tubs* stratCu1 = new G4Tubs("stratCu1", KRint, KRext, wCu1/2, KstartAngle, KtotAngle);
    stratCu1_log = new G4LogicalVolume(stratCu1, cu, "Cu1log");
// polyimide 2
G4Tubs* stratK2 = new G4Tubs("stratK2", KRint, KRext, wK2/2, KstartAngle, KtotAngle);
    stratK2_log = new G4LogicalVolume(stratK2, Kapton, "K2log");
// adhesive 2
G4Tubs* stratAd2 = new G4Tubs("stratAd2", KRint, KRext, wAd2/2, KstartAngle, KtotAngle);
    stratAd2_log = new G4LogicalVolume(stratAd2, Epoxy, "Ad2log");
// polyimide 3
G4Tubs* stratK3 = new G4Tubs("stratK3", KRint, KRext, wK1/2, KstartAngle, KtotAngle);
    stratK3_log = new G4LogicalVolume(stratK3, Kapton, "K3log");
```

8. define shapes, logical volumes and placed the silicon sensor

➢ silicon sensors are constructed using G4PVReplica method to identify easily the pads

```
//------------------------------------- Silicon sensor --------------------------------
// aluminuim foil in front of Si sensor
G4Tubs* stratAl1 = new G4Tubs("stratAl1", KRint, KRext, wAl/2, KstartAngle, KtotAngle);
    stratAl1_log = new G4LogicalVolume(stratAl1, Al, "Al1log");

//silicon layer - to identify pads we use replica of SensorPad and SenzorZone inside Sensor
G4Tubs *solidSensor = new G4Tubs("solidSensor", KRint, KRext, wSi/2, KstartAngle+15.*deg, KtotAngle/2);
    logicCalSensor = new G4LogicalVolume(solidSensor, si, "logicCalSensor");
G4Tubs *solidSensorZone = new G4Tubs("solidSensorZone", KRint, KRext, wSi/2, -3.75*deg, 7.5*deg);
    logicCalZone = new G4LogicalVolume(solidSensorZone, si, "logicCalZone");
G4Tubs *solidSensorPad = new G4Tubs("solidSensorPad", KRint, KRext, wSi/2, -3.75*deg, 7.5*deg);
   fLogicCalPad = new G4LogicalVolume(solidSensorPad, si, "logicCalPad");

new G4PVReplica("SiCalZone", logicCalZone, logicCalSensor, kPhi, 2, 7.5*deg, KstartAngle+15.*deg);
new G4PVReplica("SiCalPad", fLogicCalPad, logicCalZone, kRho, 64, 1.8*mm, KRint);

// aluminuim foil behind Si sensor
G4Tubs* stratAl2 = new G4Tubs("stratAl1", KRint, KRext, wAl/2, KstartAngle, KtotAngle);
    stratAl2_log = new G4LogicalVolume(stratAl2, Al, "Al2log");
```

# Geometry construction

9. define assembly of an entire silicon sensor

```
// start for assembly volume: Kapton fan-out, Al, glue, K_HV, epoxy_glue, Carbon_fiber .... (all but absorber plate)
G4AssemblyVolume* assamblyKfo = new G4AssemblyVolume();

G4ThreeVector Ta;
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZK1 );
assamblyKfo->AddPlacedVolume( stratK1_log, Ta, 0 );                  // polyimide 1
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZAd1 );
assamblyKfo->AddPlacedVolume( stratAd1_log, Ta, 0 );                 // adhesive 1
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZCu1 );
assamblyKfo->AddPlacedVolume( stratCu1_log, Ta, 0 );                 // copper 1
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZK2 );
assamblyKfo->AddPlacedVolume( stratK2_log, Ta, 0 );                  // polyimide 2
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZAd2 );
assamblyKfo->AddPlacedVolume( stratAd2_log, Ta, 0 );                 // adhesive 2
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZK3 );
assamblyKfo->AddPlacedVolume( stratK3_log, Ta, 0 );                  // polyimide 3
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZAraldite1 );
assamblyKfo->AddPlacedVolume( stratAraldite1_log, Ta, 0 );           // epoxy glue 1 - Araldite
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZAl1 );
assamblyKfo->AddPlacedVolume( stratAl1_log, Ta, 0 );                 // aluminuim foil in front of Si sensor
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZSi );
assamblyKfo->AddPlacedVolume( logicCalSensor, Ta, 0 );               // Si layer
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZAl2 );
assamblyKfo->AddPlacedVolume( stratAl2_log, Ta, 0 );                 // aluminuim foil behind Si sensor
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( Zcg );
assamblyKfo->AddPlacedVolume( stratconductive_glue_log, Ta, 0 );     // conductive glue
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZCu_HV );
assamblyKfo->AddPlacedVolume( stratCu_HV_log, Ta, 0 );              // copper
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZK_HV );
assamblyKfo->AddPlacedVolume( stratK_HV_log, Ta, 0 );                // polyimide
Ta.setX( translateX*mm ); Ta.setY( translateY*mm ); Ta.setZ( ZAraldite2 );
assamblyKfo->AddPlacedVolume( stratAraldite2_log, Ta, 0 );           // epoxy glue 2 - Araldite
Ta.setX( translateX*mm ); Ta.setY( tranTY*mm ); Ta.setZ( ZCf );
assamblyKfo->AddPlacedVolume( CfiberLog, Ta, 0 );                   // carbon fiber support


for( int i = 0; i < 15; i++ ) {
  G4ThreeVector Tm( 0,0,(wSlot*i)*mm);
  assamblyKfo->MakeImprint( worldLog, Tm, 0, 0, pSurfChk );
}
```

- placement of all 15 silicon sensors using MakeImprint

10. define a region for sensors in order to apply various conditions on it

```
//--------------- define region----------
G4Region* aPad = new G4Region("aPad");
aPad->AddRootLogicalVolume(logicCalSensor);
```

11. visualization of all constructed shapes

```
//======================================= visualization ========================
// Invisible world volume
worldLog->SetVisAttributes(G4VisAttributes::Invisible);
// trigger - 2 scintillators
scintplanLog->SetVisAttributes(G4Colour::Red());
// telescope - 2 before and 3 after magnet
for( int c = 0; c < 5; c++ ) teleplanLog[c]->SetVisAttributes(G4Colour::Green());
// magnet
//magnetLog->SetVisAttributes(G4Colour::Red());
//----- Al foil
AlfoilLog->SetVisAttributes(G4Colour::Gray());
//----- tungsten absorber
for( int l = 0; l < 16; l++ ) absorberLog[l]->SetVisAttributes(G4Colour::Blue());
//----- Sensor layers
stratK1_log ->SetVisAttributes(G4Colour::Green());
stratAd1_log->SetVisAttributes(G4Colour::Magenta());
stratCu1_log->SetVisAttributes(G4Colour::Blue());
stratK2_log ->SetVisAttributes(G4Colour::Green());
stratAd2_log->SetVisAttributes(G4Colour::Magenta());
stratK3_log ->SetVisAttributes(G4Colour::Green());
stratAraldite1_log->SetVisAttributes(G4Colour::Grey());
stratAl1_log->SetVisAttributes(G4Colour::Cyan());
logicCalSensor->SetVisAttributes(G4VisAttributes::Invisible);
logicCalZone->SetVisAttributes(G4VisAttributes::Invisible);
fLogicCalPad->SetVisAttributes(G4Colour::Red());
stratAl2_log->SetVisAttributes(G4Colour::Cyan());
stratconductive_glue_log->SetVisAttributes(G4Colour::Brown());
stratCu_HV_log->SetVisAttributes(G4Colour::Blue());
stratK_HV_log->SetVisAttributes(G4Colour::Green());
stratAraldite2_log->SetVisAttributes(G4Colour::Grey());
CfiberLog->SetVisAttributes(G4Colour::Yellow());
```

- Instantiate **sensitive detectors** and set them to corresponding volumes

```
void DetectorConstruction::ConstructSDandField()
{
    G4SDManager* sdManager = G4SDManager::GetSDMpointer();

    LumicalSD* SenDetLumical = new LumicalSD("sdLumical");
    sdManager -> AddNewDetector(SenDetLumical);
    fLogicCalPad->SetSensitiveDetector(SenDetLumical);
}
```

# Overlap checking

Each one of the geometry element constructed had the option for the checking of the volume overlaps activated

- **tungsten absorbers**

```
absorberPhys[k] = new G4PVPlacement(0, G4ThreeVector(translateX, tranTY, wW0),
                                    absorberLog[k], "absorberPhys", worldLog, false, k, fCheckOverlaps);
```

- **silicon sensors assembly components**

```
for( int i = 0; i < 15; i++ ) {
  G4ThreeVector Tm( 0,0,(wSlot*i)*mm);
  assamblyKfo->MakeImprint( worldLog, Tm, 0, 0, pSurfChk );
}
```

- **Outcome of the overlap checking**

```
Using Root analysis manager
Checking overlaps for volume alfoilPhys (G4Box) ... OK!
Checking overlaps for volume teleplanPhys (G4Box) ... OK!
Checking overlaps for volume teleplanPhys (G4Box) ... OK!
Checking overlaps for volume teleplanPhys (G4Box) ... OK!
Checking overlaps for volume teleplanPhys (G4Box) ... OK!
Checking overlaps for volume teleplanPhys (G4Box) ... OK!
Checking overlaps for volume scintplanPhys1 (G4Box) ... OK!
Checking overlaps for volume scintplanPhys2 (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume absorberPhys (G4Box) ... OK!
Checking overlaps for volume av_1_impr_1_K1log_pv_0 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Ad1log_pv_1 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Cu1log_pv_2 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_K2log_pv_3 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Ad2log_pv_4 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_K3log_pv_5 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Araldite1log_pv_6 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Al1log_pv_7 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_logicCalSensor_pv_8 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Al2log_pv_9 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_conductivegluelog_pv_10 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_CuHVlog_pv_11 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_KHVlog_pv_12 (G4Tubs) ... OK!
Checking overlaps for volume av_1_impr_1_Araldite2log_pv_13 (G4Tubs) OK!
Checking overlaps for volume av_1_impr_1_CfiberLog_pv_14 (G4Box) ... OK!
```

# Geometry construction and overlap checking

**Geometry construction:**

- rather simplist with some minor point of high interest;
- geometry constructed having placed the first sensors in the (0,0,0) coordinates; all other geometries are placed with respect to this one;

**telescope:**

- a G4Box shape;
- positioned using Z axis coordinates taken from an array;

**scintillators:**

- a G4Box shape;
- each placed individually using regular G4PVPlacement class;

**tungsten absorbers:**

- a G4Box shape;
- each absorber has a different width;
- construction of solid made inside a loop with the width for absorbers taken from an array;

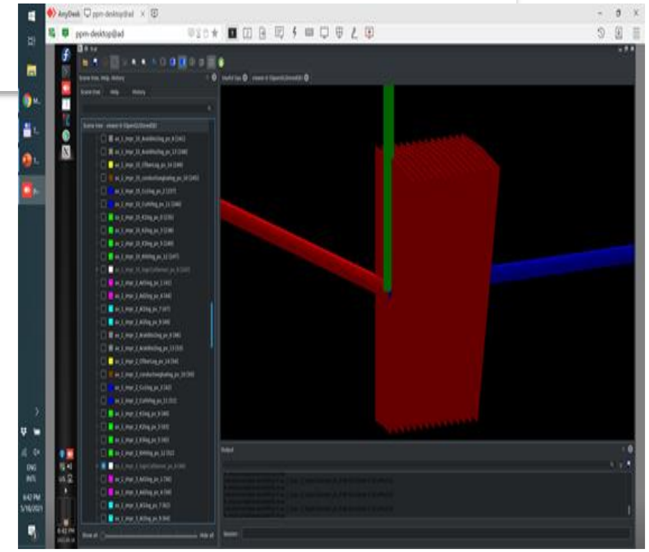
**sensors:**

- each layer constructed individually from kapton fan-out to carbon fiber support;
- silicon sensor build using G4PVReplica which replicates the pads horizontally and the sectors vertically;
- everything wrapped in an assembly;
- the assembly is positioned several times using a loop and a geometry overlap checking.

# Data collection

*Data are collected using Sensitive Detectors which has the goal of creating hits objects through the following virtual methods*

- Initialize()
- ProcessHits()
- EndOfEvent()



**➢ *Intialize()* - create a hit collection at start of an event**

```
fHitsCollection = new LumicalHitsCollection(SensitiveDetectorName, collectionName[0]);
if (fLumiCalHCID<0)
{ fLumiCalHCID = G4SDManager::GetSDMpointer()->GetCollectionID(fHitsCollection); }
hce->AddHitsCollection(fLumiCalHCID,fHitsCollection);

// Hit collection is created at the begining of each event
for(G4int plan=33; plan<244; plan+=15){
    for(G4int zone=0; zone<4; zone++){
        for(G4int pad=0; pad<64; pad++){
            G4int plan1 = (plan-33)/15;
            fHitsCollection->insert(new LumicalHit(plan1, zone, pad));
        }
    }
}
```

**➢ at each step in a sensitive detector, the *ProcessHit()* method is invoked which create, fill and stores the Hit objects**

```
G4double edep = (step->GetTotalEnergyDeposit()/MeV);
if (edep <= 0.) return true;


G4StepPoint* preStepPoint = step->GetPreStepPoint();
G4TouchableHistory* touchable = (G4TouchableHistory*)(preStepPoint->GetTouchable());
// G4ThreeVector position = step->GetTrack()->GetPosition();    // position in mm

//G4int plan = touchable->GetCopyNumber(3);
G4int plan = touchable->GetReplicaNumber(2);
G4int zone = touchable->GetReplicaNumber(1);
G4int pad  = touchable->GetReplicaNumber(0);

G4int padID = pad + 64 * zone + 256 * ((plan-33)/15);
LumicalHit *hit = (*fHitsCollection)[padID];

hit->fEdep += edep;
```

# Processing hit information

*Data are retrieved using EventAction:EndOfEventAction() class*

- **get the index number of the hit collection - which is unique and don't change during a run**

```
 if (fLumiCalHCID==-1) {
    G4SDManager* sdManager = G4SDManager::GetSDMpointer();
    fLumiCalHCID = sdManager->GetCollectionID("sdLumical/LumicalCollect");
}
```

- **retrieve the pointer to the hits collection with the GetHC() method using the collection index**

```
G4HCofThisEvent* hce = event->GetHCofThisEvent();
if (!hce) return;

// Get hits collections
LumicalHitsCollection* lumicalHC = static_cast<LumicalHitsCollection*>(hce->GetHC(fLumiCalHCID));
if ( !lumicalHC ) return;
```

- **loop through the entries of hits collection to access individual hits**

```
G4int hitsNb = 0;
G4double Edepo[15] = {0.};


for (unsigned long i = 0; i < lumicalHC->GetSize(); ++i){
    auto hit = static_cast<LumicalHit*>(lumicalHC->GetHit(i));
    if(hit->fEdep <= 0.) continue;
    hitsNb++;
    fCalPlan.push_back(hit->fPlan);
    fCalZone.push_back(hit->fZone);
    fCalPad.push_back(hit->fPad);
    fCalEne.push_back(hit->fEdep);

    fCalX.push_back(hit->fX);
    fCalY.push_back(hit->fY);
    fCalZ.push_back(hit->fZ);
    fCalPx.push_back(hit->fPx);
    fCalPy.push_back(hit->fPy);
    fCalPz.push_back(hit->fPz);

    for (G4int k=0;k<15;k++) {
       if (hit->fPlan == k) { Edepo[k] += hit->fEdep; }
    }
}
```

- **store the output in analysis objects**

```
analysisManager->FillNtupleIColumn(0, hitsNb);

for (G4int j=0;j<15;j++) {
    analysisManager->FillNtupleDColumn(j+11, Edepo[j]);
}

analysisManager->AddNtupleRow();
```

# Output datafile

*Data are collected using Sensitive Detectors which has the goal of creating hits objects through the following virtual metods:*

- Initialize()
- ProcessHits()
- EndOfEvent()

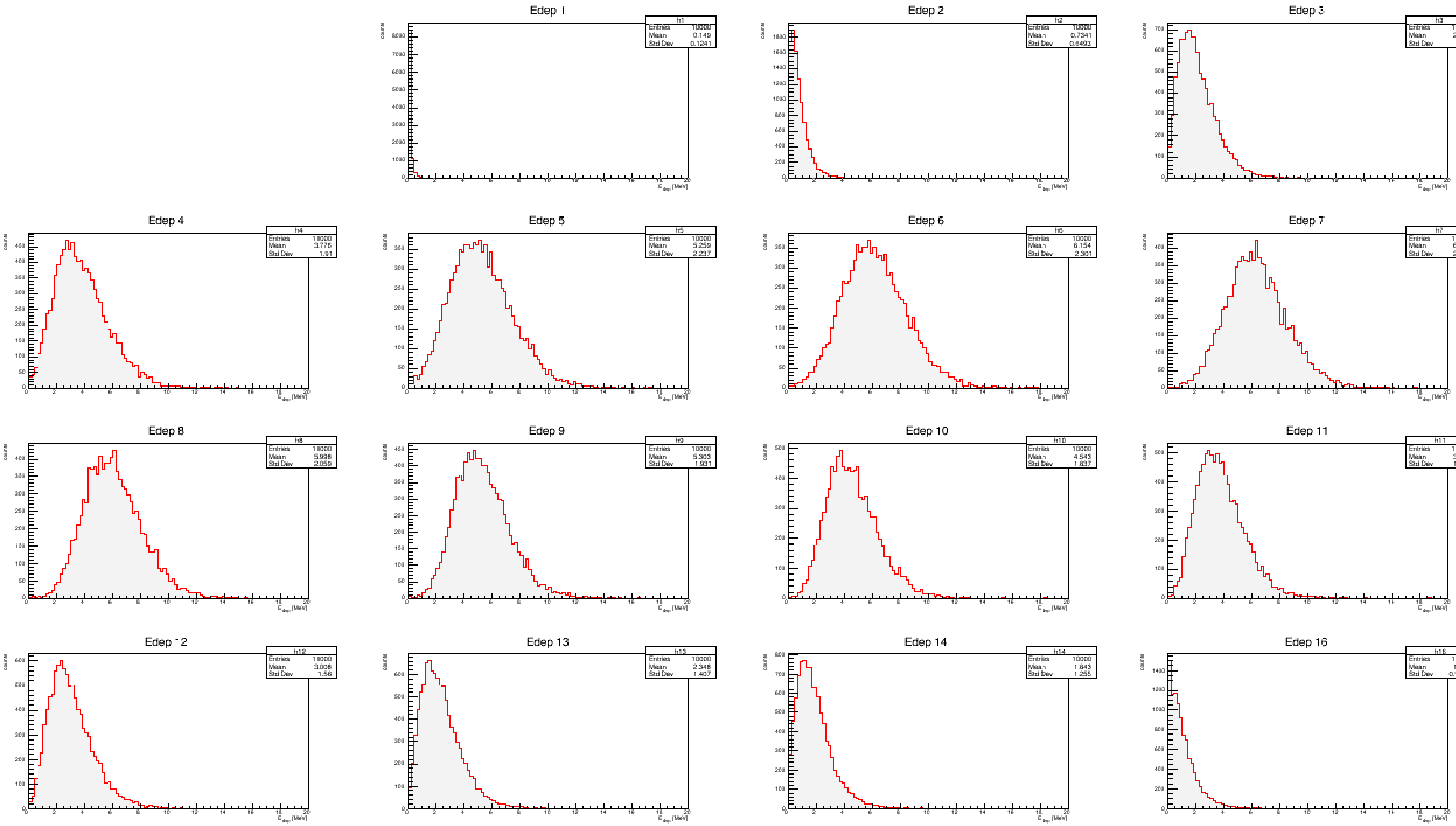
➢RunAction class
Create analysis manager, Createie ntuples

➢ Energy deposited in each plane

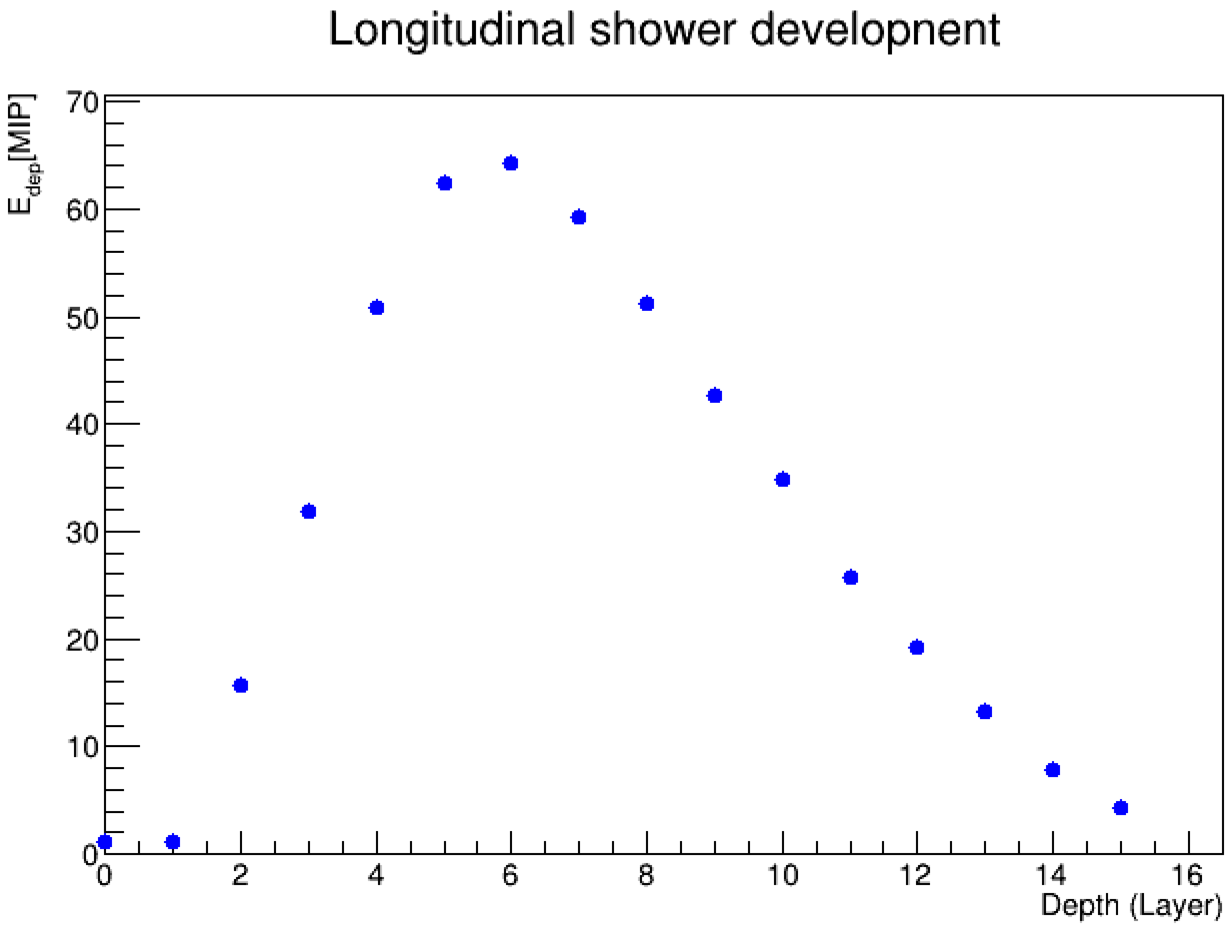
```
// Default settings
  analysisManager->SetVerboseLevel(1);
  analysisManager->SetFileName("fcal");

 // Creating ntuples
  if ( fEventAction ) {
    analysisManager->CreateNtuple("LumicalTree", "Calorimeter data");

    analysisManager->CreateNtupleIColumn("nHits");                    // Id = 0
    analysisManager->CreateNtupleIColumn("Plan", fEventAction->fCalPlan);     // Id = 1
    analysisManager->CreateNtupleIColumn("Sector", fEventAction->fCalZone);    // Id = 2
    analysisManager->CreateNtupleIColumn("Pad",  fEventAction->fCalPad);       // Id = 3
    analysisManager->CreateNtupleDColumn("Energy", fEventAction->fCalEne);     // Id = 4

    analysisManager->CreateNtupleDColumn("pos_x", fEventAction->fCalX);        // Id = 5
    analysisManager->CreateNtupleDColumn("pos_y", fEventAction->fCalY);        // Id = 6
    analysisManager->CreateNtupleDColumn("pos_z", fEventAction->fCalZ);        // Id = 7
    analysisManager->CreateNtupleDColumn("px", fEventAction->fCalPx);          // Id = 8
    analysisManager->CreateNtupleDColumn("py", fEventAction->fCalPy);          // Id = 9
    analysisManager->CreateNtupleDColumn("pz", fEventAction->fCalPz);          // Id = 10 2

stringstream nameformat;
    G4String edname;
    for(G4int k = 1; k < 16; k++){
      G4String columname = "Edep";
      nameformat << k;
      columname += nameformat.str();
      nameformat.str("");
      edname = columname;

      analysisManager->CreateNtupleDColumn(edname);
    }
   analysisManager->FinishNtuple();
```

# Results

## energy deposited spectrum from different X0 - config. AA



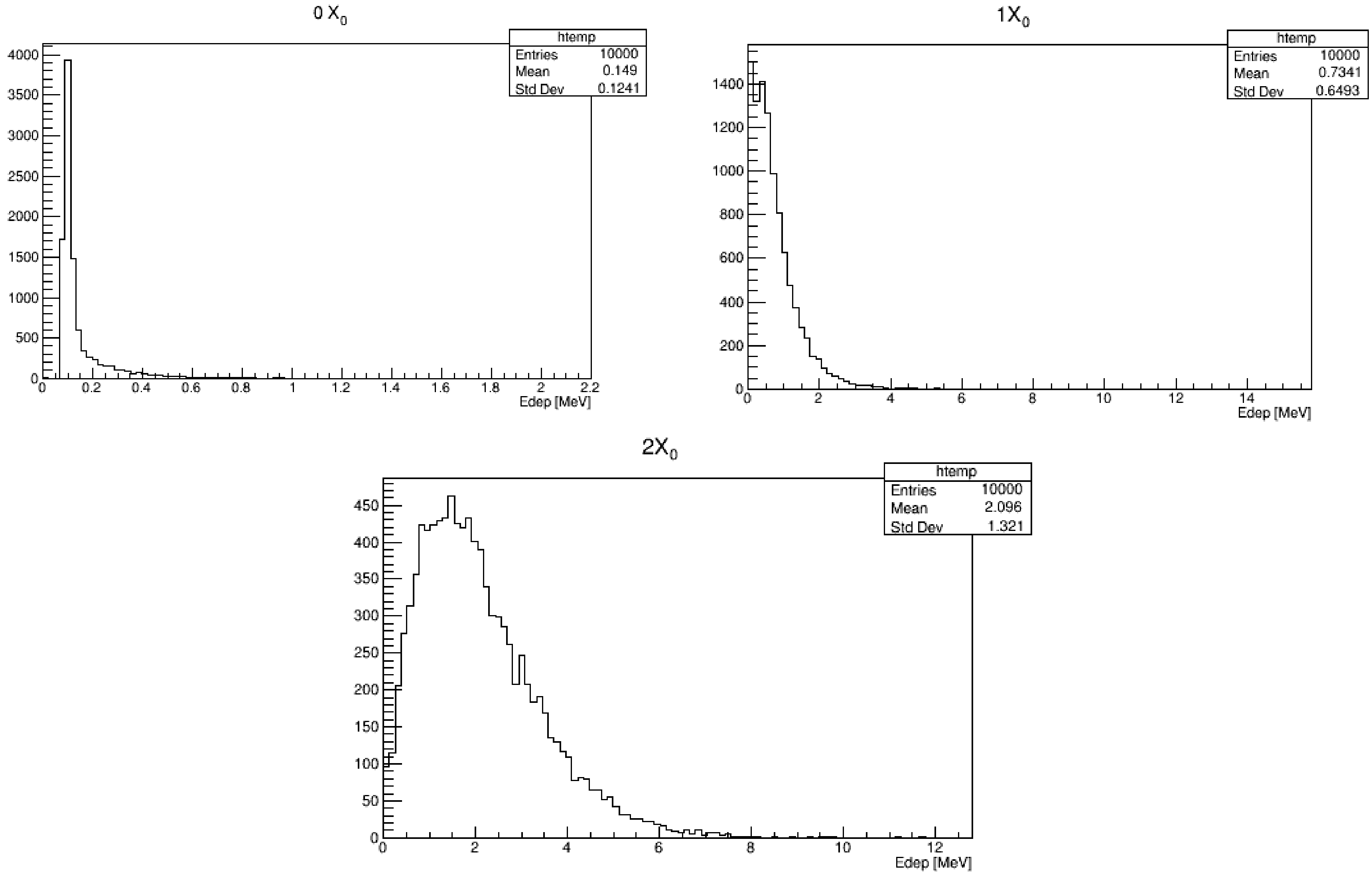
# Results



Longitudinal shower developnent
$E_{dep}$[MIP]
70
60
50
40
30
20
10
0
0
2
4
6
8
10
12
14
16
Depth (Layer)

# THANK YOU!

# Results


0 X0
htemp
Entries 10000
Mean 0.149
Std Dev 0.1241
Edep [MeV]
1X0
htemp
Entries 10000
Mean 0.7341
Std Dev 0.6493
Edep [MeV]
2X0
htemp
Entries 10000
Mean 2.096
Std Dev 1.321
Edep [MeV]

# Testbeam setup

| layer | sensor | plate | Thickness [mm] |
|---|---|---|---|
| 1 | - | Plansse 2 | 3.520 |
| 2 | 52 | Plansse 3 | 3.470 |
| 3 | 51 | MGS3 | 3.542 |
| 4 | 29 | Plansse 1 | 3.505 |
| 5 | 59 | Plansse 5 | 3.490 |
| 6 | 10 | MGS1 | 3.584 |
| 7 | 57 | MGS2 | 3.521 |
| 8 | Free | MGS5 | 3.645 |
| 9 | 53 | MGS6 | 3.470 |
| 10 | 60 | A2 | 3.55 |
| 11 | 64 | A8 | 3.588 |
| 12 | 42S | B24 | 3.543 |
| 13 | Old T2 | B23 | 3.543 |
| 14 | Old C3 | B12 | 3.55 |
| 15 | 61 | B17 | 3.55 |
| 16 | Old C4 | A5 | 3.538 |
| 17 | 58 | Plansse 4 | 3.474 |

LumiCal Module components

| Module Component | part components | Thickness (µm) | | | |
|---|---|---|---|---|---|
| | | | Min | Max | Average |
| Kapton fan-out[1] | | | 130 | 160 | 150 |
| | polyimide | 12.7 | | | |
| | adhesive | 25.4 | | | |
| | copper (partly etched) | 35.56 | | | |
| | polyimide | 25.4 | | | |
| | copper (not in sensor area) | ~~17.78~~ | | | |
| | adhesive | 12.7 | | | |
| | polyimide | 12.7 | | | |
| epoxy glue[2] | | | 10 | 15 | 10 |
| Si sensor | | | 360 | 360 | 360 |
| | Al | 20 | | | |
| | Si | 320±15 | | | |
| | Al | 20 | | | |
| conductive glue[3] | | | 20 | 50 | 40 |
| HV kapton[1] | | | 90 | 90 | 90 |
| | ENIG | 0.1 | | | |
| | copper | 24.9 | | | |
| | polyimide | 75 | | | |
| epoxy glue[2] | | | 20 | 20 | 20 |
| Carbon-fiber support[4] | | | 110 | 120 | 115 |

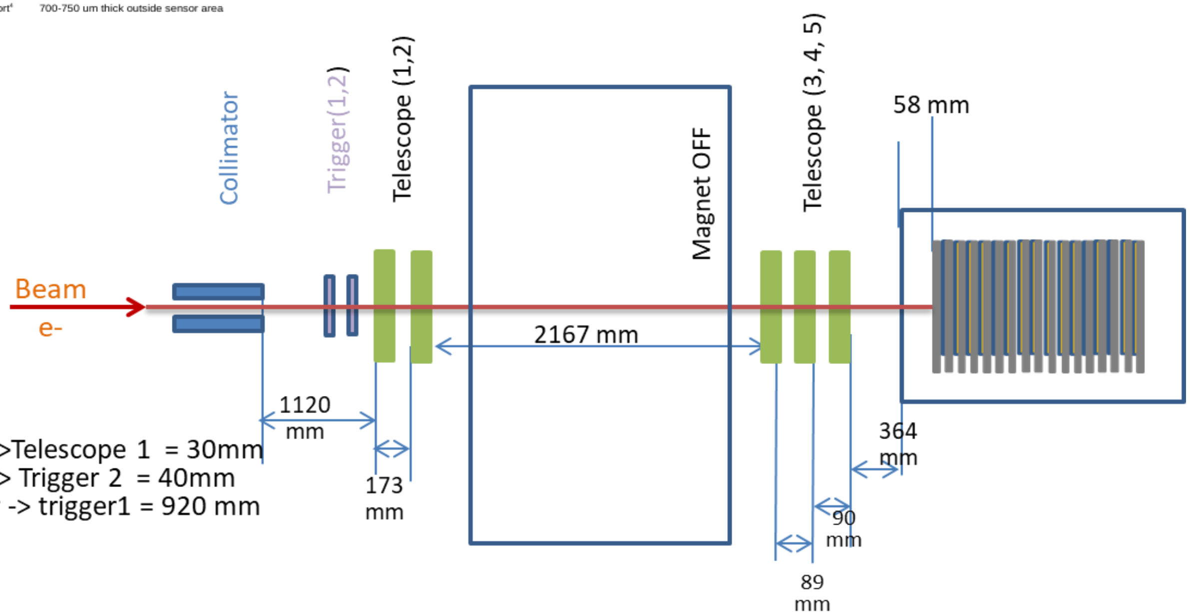
Kapton[1] There was another version, where kapton thickness was 20 um less. It was used for modules 20 and 21.

epoxy glue[2] Araldite
Part A Bisphenol A epoxy resin
Part B N(3-dimethylaminopropyl)-1,3-propylenediamine
Mixed 1:1

conductive glue[3] TRA-DUCT 2902 epoxy with silver filling
http://bondingsource.com/techdata/TRA-DUCT%202902-EN.pdf

Carbon-fiber support[4] 700-750 um thick outside sensor area

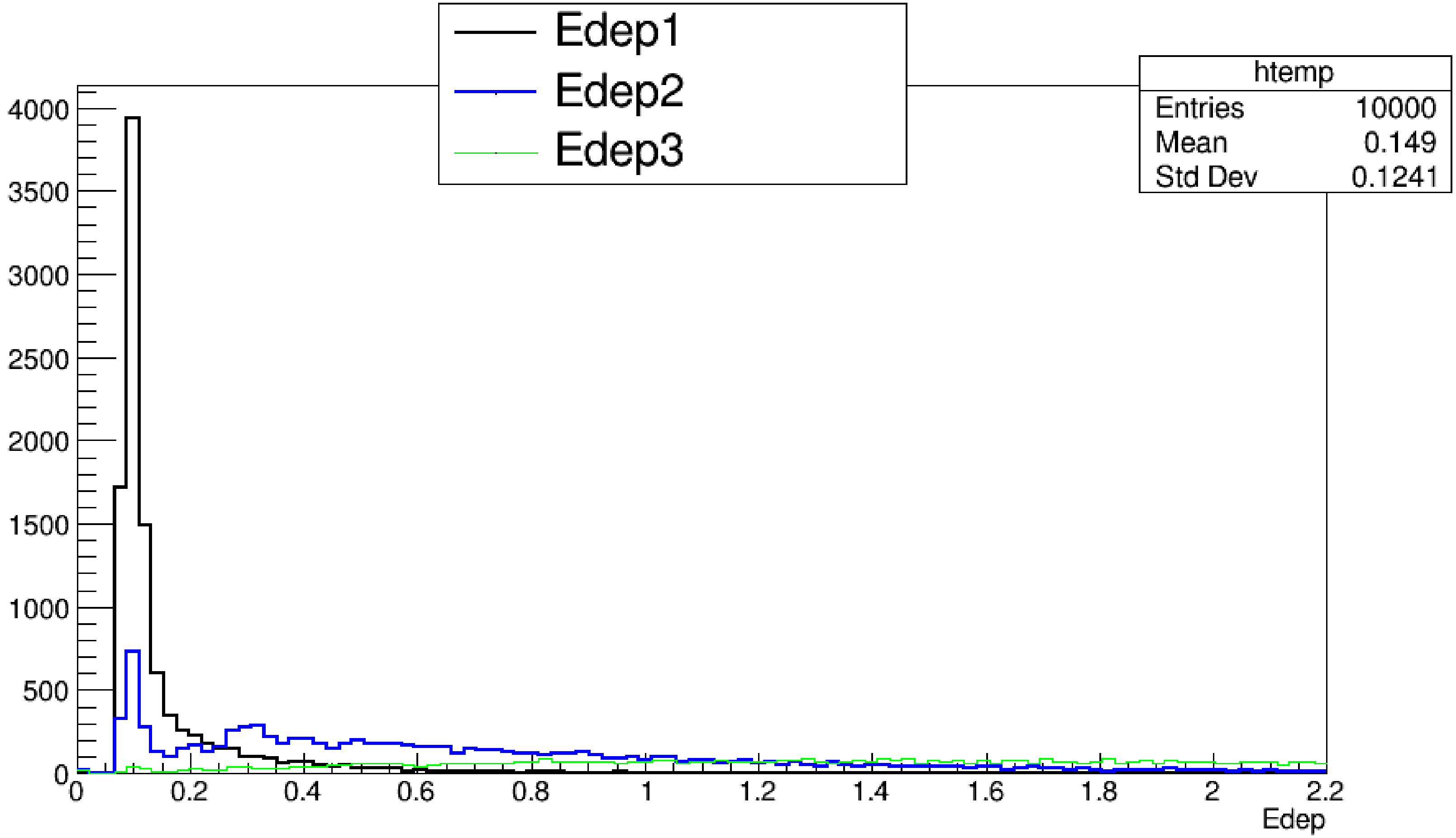
| Brand | Density | Chemical composition | | |
|---|---|---|---|---|
| | | W | Ni | Cu |
| Plansee | 18 | 95% | 3.5% (estimated) | 1.5% (estimated) |
| MGS | 17.7 | 93% | 5.25% | 1.75% |
| JINR | 18 | 95% | 3.5% (estimated) | 1.5% (estimated) |




estimated
Trigger 2 ->Telescope 1 = 30mm
Trigger 1 -> Trigger 2 = 40mm
Collimator -> trigger1 = 920 mm

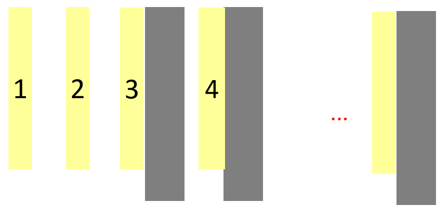
# Part II

**Suggestion 1**: to apply the same bin for Edep 1 and Edep 2

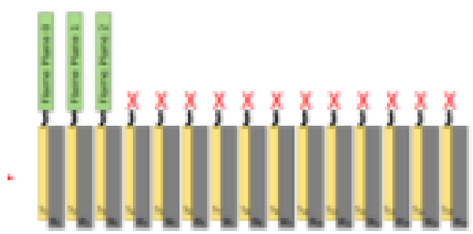


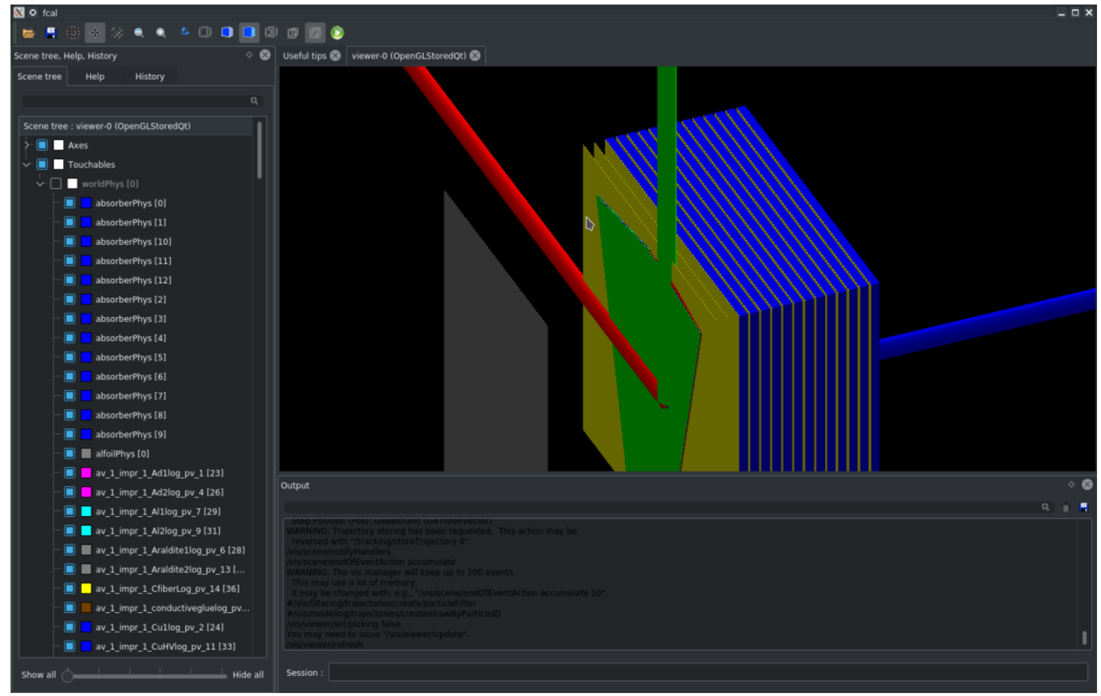
**Suggestion 2**: to simulate a configuration without the first 2 W absorber in order to studied the behavior of sensors 2 and 3

Conf aaaa

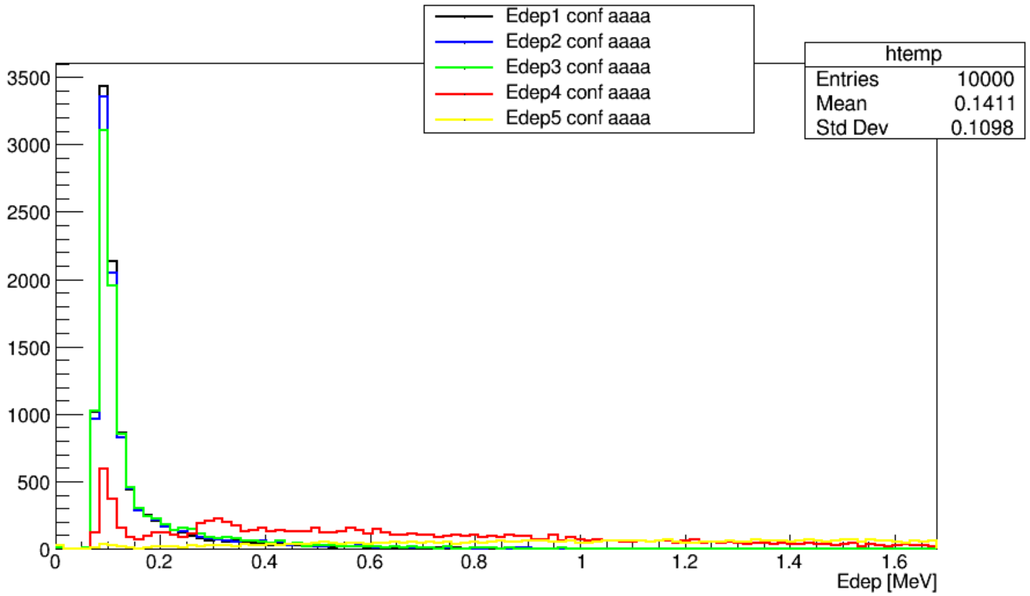



Conf AA

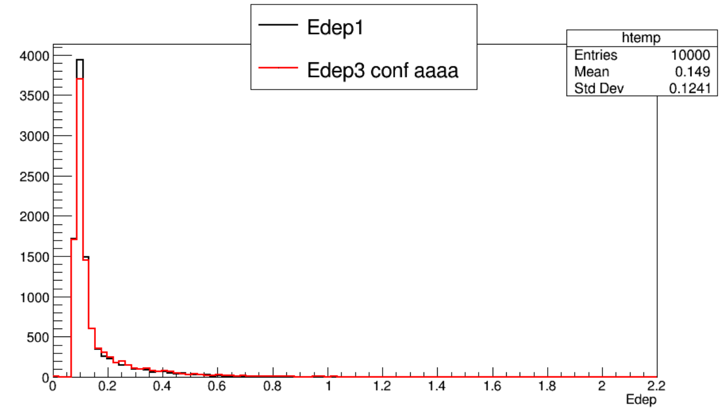


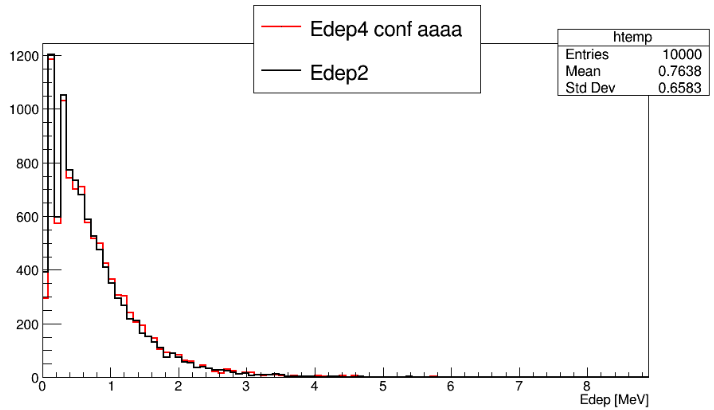


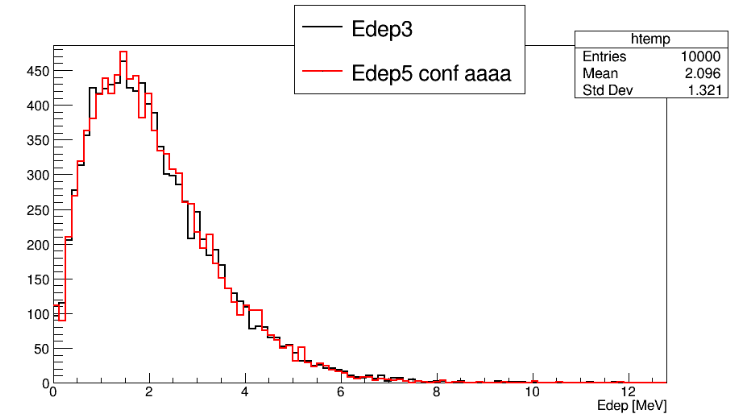
# Energy deposited in sensors



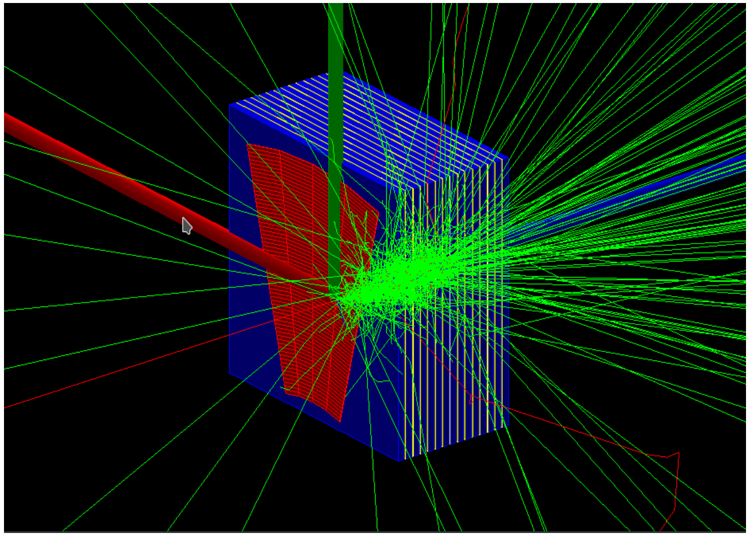

## un-normalised

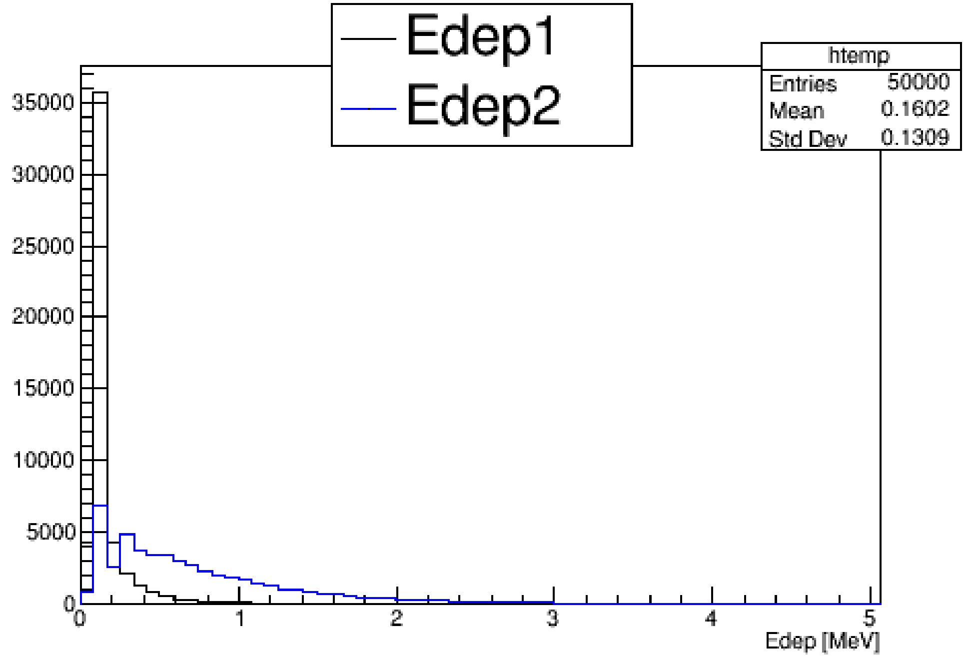


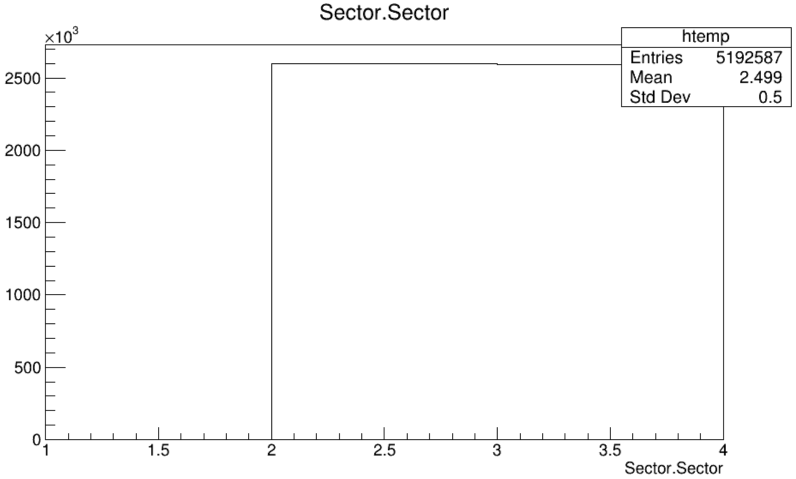



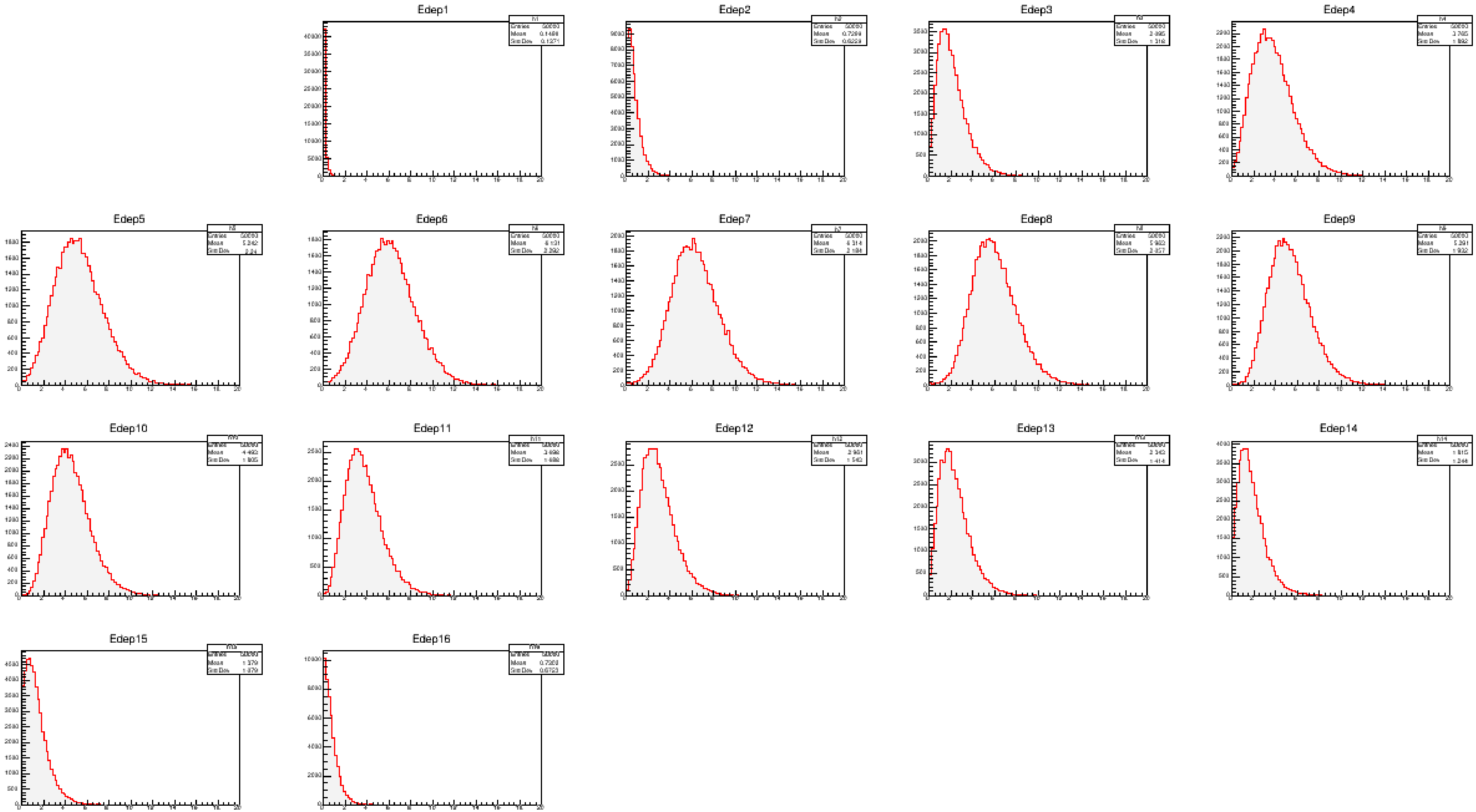
**Suggestion 3**: to move beam in pad 27 (like in experimental data)
to implement all 4 sectors
to increase statistics at 50000 events

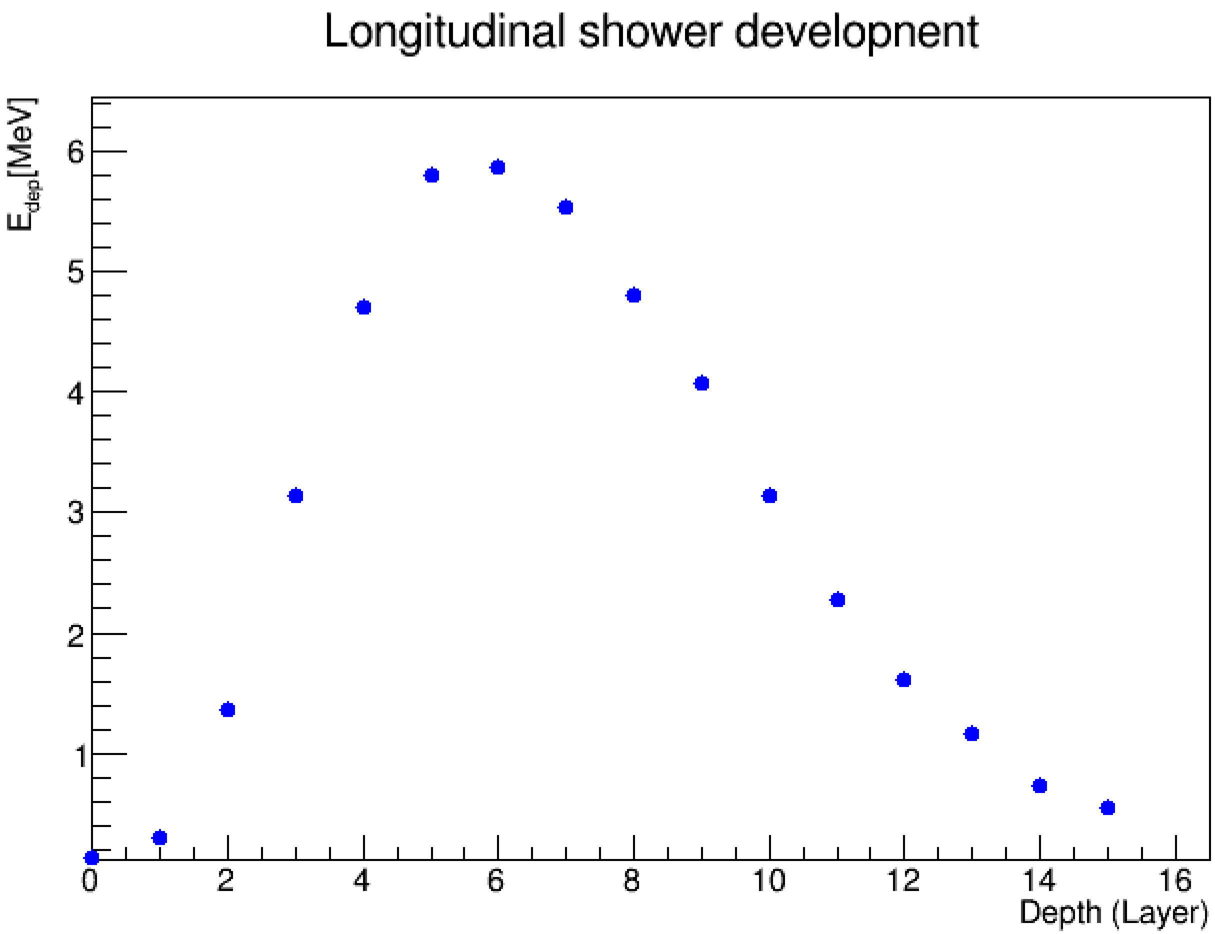




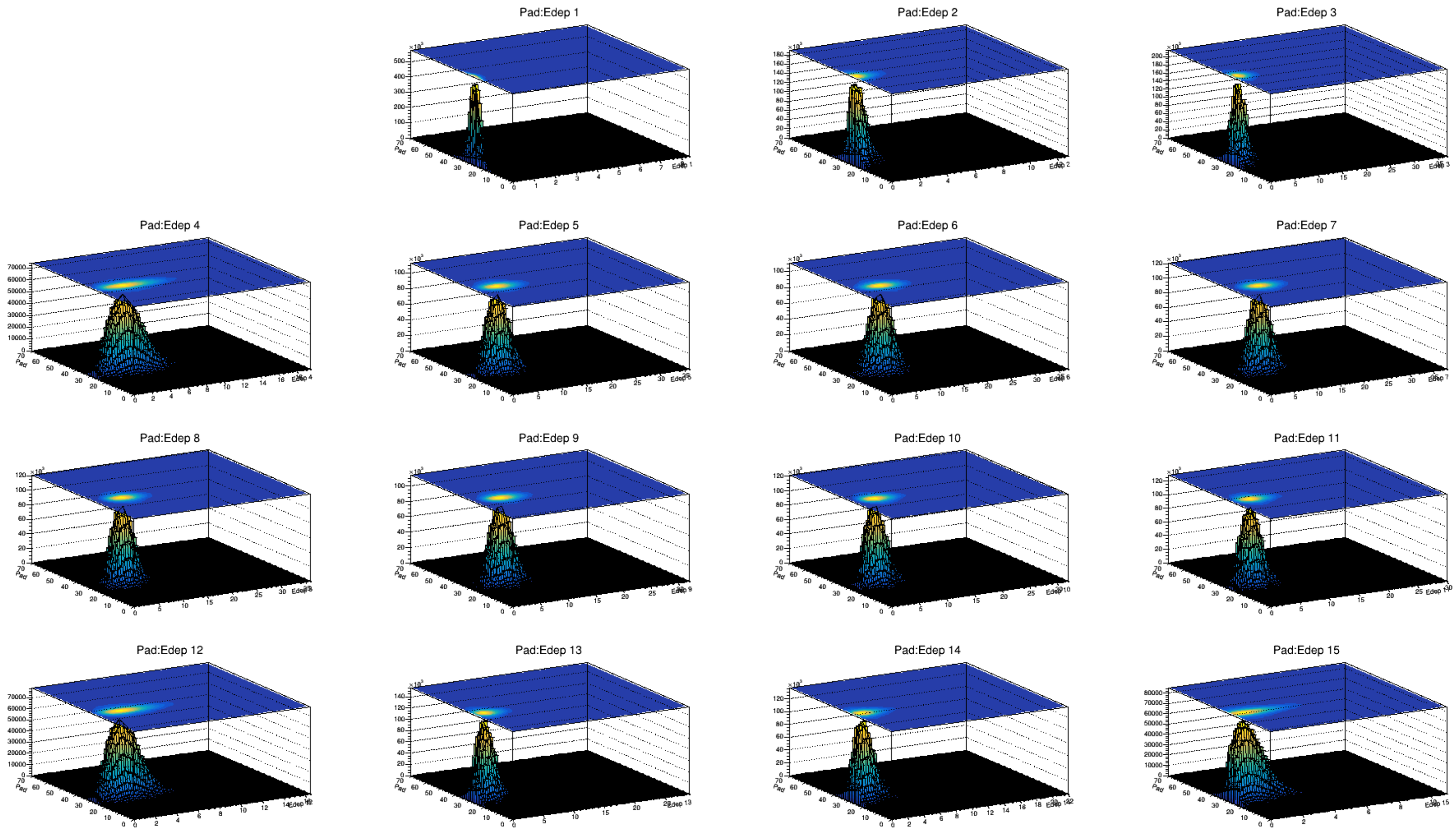
# energy deposited spectrum from different $X_0$ - config. AA





Longitudinal shower developnent
E_dep[MeV]
Depth (Layer)

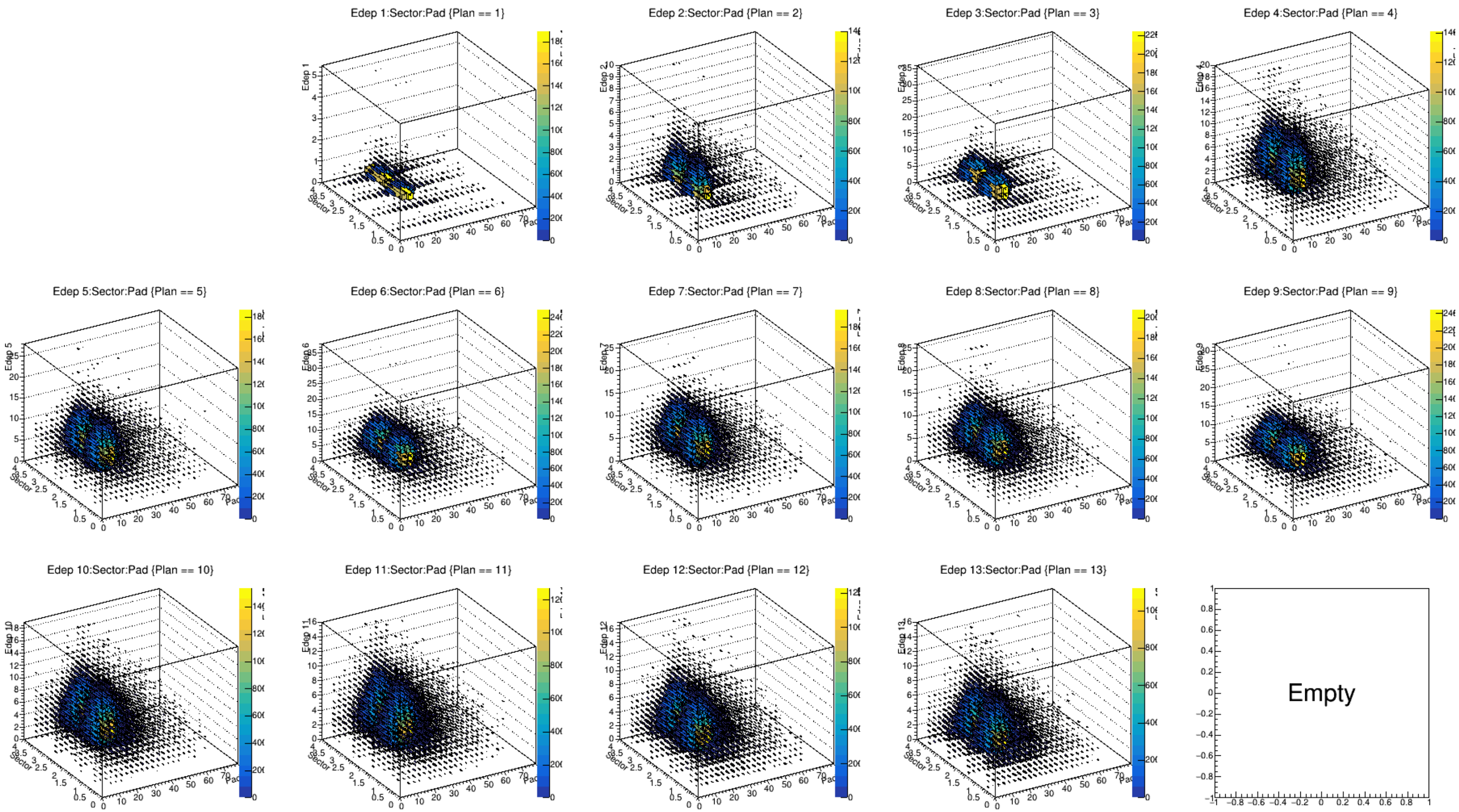
# **Suggestion 4**: to plot energy deposited in each pad for each plane



## Conclusions

- Still needed further investigation on energy deposition
- Ghost hits at boundary mistaken as real hits;

# 3D representation of energy deposited in each plan for each sector/pad *




* For a configuration where energy deposited was collected in all 4 sectors