GEOMETRY IN GEANT4

WORKSHOP ON MONTE-CAROL SIMULATIONS-APPLICATIONS IN SCIENCE AND TECHNOLOGY

MAY 15-17, 2017

S.AOGAKI

BEFORE PRESENTATION

- I made one example including my talks
- Please download from
- https://github.com/aogaki/Workshop2017
- e-mail: sohichiroh.aogaki@nipne.ro

BEFORE PRESENTATION

- I will explain about DetectorConstruction class

```
int main(int argc, char **argv)
{
   …
   // Construct the default run manager
#ifdef G4MULTITHREADED
   G4MTRunManager *runManager = new G4MTRunManager();
   runManager->SetNumberOfThreads(G4Threading::G4GetNumberOfCores());
#else
   G4RunManager *runManager = new G4RunManager();
#endif

   // Detector construction
   runManager->SetUserInitialization(new MyDetectorConstruction());

   // Physics list
   G4VModularPhysicsList *physicsList = new MyPhysicsList();
   physicsList->SetVerboseLevel(0);
   runManager->SetUserInitialization(physicsList);

   // Primary generator action and User action initialization
   runManager->SetUserInitialization(new MyActionInitialization());

   // Initialize G4 kernel
   //
   runManager->Initialize();

   …

   return 0;
}
```

CONTENTS

INTRODUCTION

INTRODUCTION

- Geant4 has many classes about the geometry
- In this presentation, very basics only
  - Box and cylinder shape only
- How to make UI commands
  - User defining commands allow you to change something when the program running
  - This is not only for Geometry.

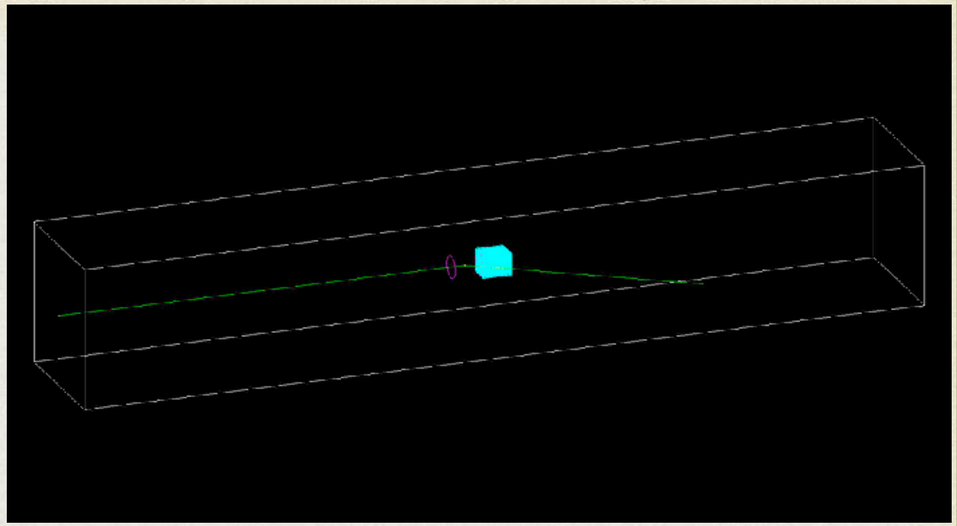
INTRODUCTION

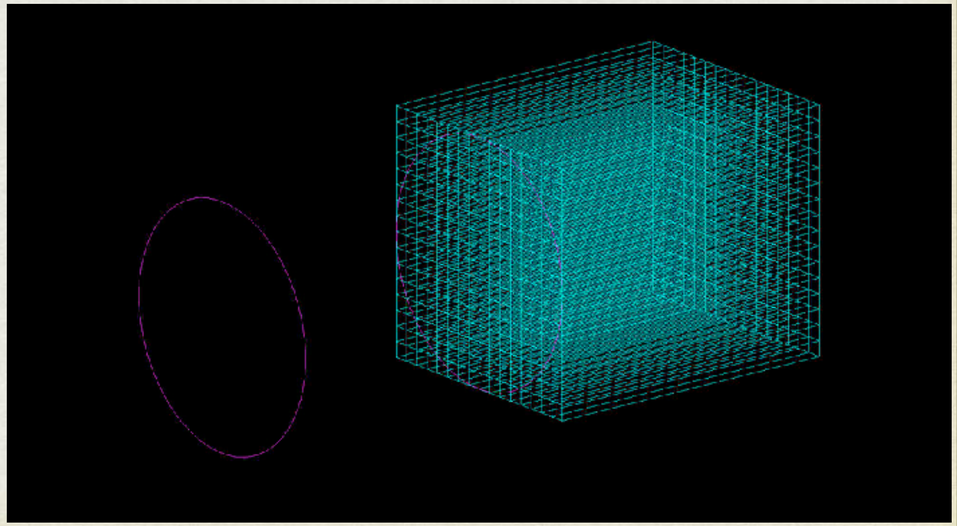
- Making user DetectorConstruction class
  - Inheritance: G4VUserDetectorConstruction
- Making Construct member function

THE STRUCTURE OF GEOMETRY

THE STRUCTURE OF GEOMETRY

- World
  - Some volumes
    - Some volumes
- Making hierarchy of geometries





THE STRUCTURE OF GEOMETRY

- You know Solid, Logical volume and Physical volume
- Making the Shape of geometry (Solid)
- Making the object including the Solid and material information (Logical volume)
- Placing the logical volume in the mother logical volume (Physical volume)

HOW TO DEFINE MATERIALS

HOW TO DEFINE MATERIALS

- Three materials are defined
- Vacuum, NaI, LGSO
- The materials are G4Material class

```
#include <G4NistManager.hh>

#include "MyDetectorConstruction.hpp"
#include "MySD.hpp"

MyDetectorConstruction::MyDetectorConstruction()
   : fVacuumMat(nullptr),
     fScinti1Mat(nullptr),
     fScinti2Mat(nullptr)
{
   …
   fCheckOverlap = true;
   DefineMaterials();
   …
}

void MyDetectorConstruction::DefineMaterials()
{
   G4NistManager *manager = G4NistManager::Instance();

   // NIST database materials
   fVacuumMat = manager->FindOrBuildMaterial("G4_Galactic");
   fScinti1Mat = manager->FindOrBuildMaterial("G4_SODIUM_IODIDE");

   // LGSO
   G4Element *Si = manager->FindOrBuildElement("Si");
   G4Element *O = manager->FindOrBuildElement("O");
   G4Element *Lu = manager->FindOrBuildElement("Lu");
   G4Element *Gd = manager->FindOrBuildElement("Gd");

   fScinti2Mat = new G4Material("LGSO", 6.5*g/cm3, 4, kStateSolid);
   fScinti2Mat->AddElement(Lu, 72.86*perCent);
   fScinti2Mat->AddElement(Gd, 3.45*perCent);
   fScinti2Mat->AddElement(Si, 6.16*perCent);
   fScinti2Mat->AddElement(O, 17.53*perCent);
}
```

HOW TO DEFINE MATERIALS

- Some materials are predefined in Geant4
- You can use from NIST manager (G4NistManager)
- Please check Geant4 material database
- http://geant4.web.cern.ch/geant4/workAreaUserDocKA/Backup/Docbook_UsersGuides_beta/ForApplicationDeveloper/html/apas08.html

```
#include <G4NistManager.hh>

#include "MyDetectorConstruction.hpp"
#include "MySD.hpp"

MyDetectorConstruction::MyDetectorConstruction()
   : fVacuumMat(nullptr),
     fScinti1Mat(nullptr),
     fScinti2Mat(nullptr)
{
   …
   fCheckOverlap = true;
   DefineMaterials();
   …
}

void MyDetectorConstruction::DefineMaterials()
{
   G4NistManager *manager = G4NistManager::Instance();

   // NIST database materials
   fVacuumMat = manager->FindOrBuildMaterial("G4_Galactic");
   fScinti1Mat = manager->FindOrBuildMaterial("G4_SODIUM_IODIDE");

   // LGSO
   G4Element *Si = manager->FindOrBuildElement("Si");
   G4Element *O = manager->FindOrBuildElement("O");
   G4Element *Lu = manager->FindOrBuildElement("Lu");
   G4Element *Gd = manager->FindOrBuildElement("Gd");

   fScinti2Mat = new G4Material("LGSO", 6.5*g/cm3, 4, kStateSolid);
   fScinti2Mat->AddElement(Lu, 72.86*perCent);
   fScinti2Mat->AddElement(Gd, 3.45*perCent);
   fScinti2Mat->AddElement(Si, 6.16*perCent);
   fScinti2Mat->AddElement(O, 17.53*perCent);
}
```

HOW TO DEFINE MATERIALS

- In case of no predefined material
- You can make an own material with G4Element class
- I use percentage for ratio of elements
- You can write as H_2O

```
fScinti2Mat->AddElement(H, 2);
fScinti2Mat->AddElement(O, 1);
```

```
#include <G4NistManager.hh>

#include "MyDetectorConstruction.hpp"
#include "MySD.hpp"

MyDetectorConstruction::MyDetectorConstruction()
    : fVacuumMat(nullptr),
      fScinti1Mat(nullptr),
      fScinti2Mat(nullptr)
{
    …
    fCheckOverlap = true;
    DefineMaterials();
    …
}

void MyDetectorConstruction::DefineMaterials()
{
    G4NistManager *manager = G4NistManager::Instance();

    // NIST database materials
    fVacuumMat = manager->FindOrBuildMaterial("G4_Galactic");
    fScinti1Mat = manager->FindOrBuildMaterial("G4_SODIUM_IODIDE");

    // LGSO
    G4Element *Si = manager->FindOrBuildElement("Si");
    G4Element *O = manager->FindOrBuildElement("O");
    G4Element *Lu = manager->FindOrBuildElement("Lu");
    G4Element *Gd = manager->FindOrBuildElement("Gd");

    fScinti2Mat = new G4Material("LGSO", 6.5*g/cm3, 4, kStateSolid);
    fScinti2Mat->AddElement(Lu, 72.86*perCent);
    fScinti2Mat->AddElement(Gd, 3.45*perCent);
    fScinti2Mat->AddElement(Si, 6.16*perCent);
    fScinti2Mat->AddElement(O, 17.53*perCent);
}
```

HOW TO DEFINE GEOMETRIES

HOW TO DEFINE GEOMETRIES

- Making Solid
- Making Logical volume
- Placing (Making Physical volume)
- Its are done in Constructor member function

HOW TO DEFINE GEOMETRIES

- World volume
- This is the return value of Constructor

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   // world volume
   G4double worldX = 0.1*m;
   G4double worldY = 0.1*m;
   G4double worldZ = 0.6*m;

   G4Box *worldS = new G4Box("World", worldX / 2., worldY / 2., worldZ / 2.);
   G4LogicalVolume *worldLV = new G4LogicalVolume(worldS, fVacuumMat, "World");
   G4VPhysicalVolume *worldPV
      = new G4PVPlacement(nullptr, G4ThreeVector(), worldLV, "World", 0,
                          false, 0, fCheckOverlap);

   return worldPV;
}
```

HOW TO DEFINE GEOMETRIES

- Making a Solid as G4Box
- G4Box needs name and size in 3D

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   // world volume
   G4double worldX = 0.1*m;
   G4double worldY = 0.1*m;
   G4double worldZ = 0.6*m;

   G4Box *worldS = new G4Box("World", worldX / 2., worldY / 2., worldZ / 2.);
   G4LogicalVolume *worldLV = new G4LogicalVolume(worldS, fVacuumMat, "World");
   G4VPhysicalVolume *worldPV
      = new G4PVPlacement(nullptr, G4ThreeVector(), worldLV, "World", 0,
                          false, 0, fCheckOverlap);

   return worldPV;
}
```

G4Box (const G4String &pName, G4double pX, G4double pY, G4double pZ)

HOW TO DEFINE GEOMETRIES

- Making Logical volume
- G4LogicalVolume needs G4Solid, G4Material and name

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   // world volume
   G4double worldX = 0.1*m;
   G4double worldY = 0.1*m;
   G4double worldZ = 0.6*m;

   G4Box *worldS = new G4Box("World", worldX / 2., worldY / 2., worldZ / 2.);
   G4LogicalVolume *worldLV = new G4LogicalVolume(worldS, fVacuumMat, "World");
   G4VPhysicalVolume *worldPV
      = new G4PVPlacement(nullptr, G4ThreeVector(), worldLV, "World", 0,
                          false, 0, fCheckOverlap);

   return worldPV;
}
```

G4LogicalVolume (G4VSolid *pSolid, G4Material *pMaterial, const G4String &name, G4FieldManager *pFieldMgr=0, G4VSensitiveDetector *pSDetector=0, G4UserLimits *pULimits=0, G4bool optimise=true)

HOW TO DEFINE GEOMETRIES

- Making Physical volume
- G4LogicalVolume needs G4Solid, G4Material and name

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   // world volume
   G4double worldX = 0.1*m;
   G4double worldY = 0.1*m;
   G4double worldZ = 0.6*m;

   G4Box *worldS = new G4Box("World", worldX / 2., worldY / 2., worldZ / 2.);
   G4LogicalVolume *worldLV = new G4LogicalVolume(worldS, fVacuumMat, "World");
   G4VPhysicalVolume *worldPV
      = new G4PVPlacement(nullptr, G4ThreeVector(), worldLV, "World", 0,
                          false, 0, fCheckOverlap);

   return worldPV;
}
```

HOW TO DEFINE GEOMETRIES

```
G4VPhysicalVolume *worldPV
   = new G4PVPlacement(nullptr, G4ThreeVector(), worldLV, "World", nullptr,
                       false, 0, fCheckOverlap);
```

G4PVPlacement (G4RotationMatrix *pRot, const G4ThreeVector &tlate, G4LogicalVolume *pCurrentLogical, const G4String &pName, G4LogicalVolume *pMotherLogical, G4bool pMany, G4int pCopyNo, G4bool pSurfChk=false)

- Rotation matrix should be pointer
  - Thus, you must not delete or change the rotation matrix
  - The pointer of mother volume is nullptr

HOW TO DEFINE GEOMETRIES

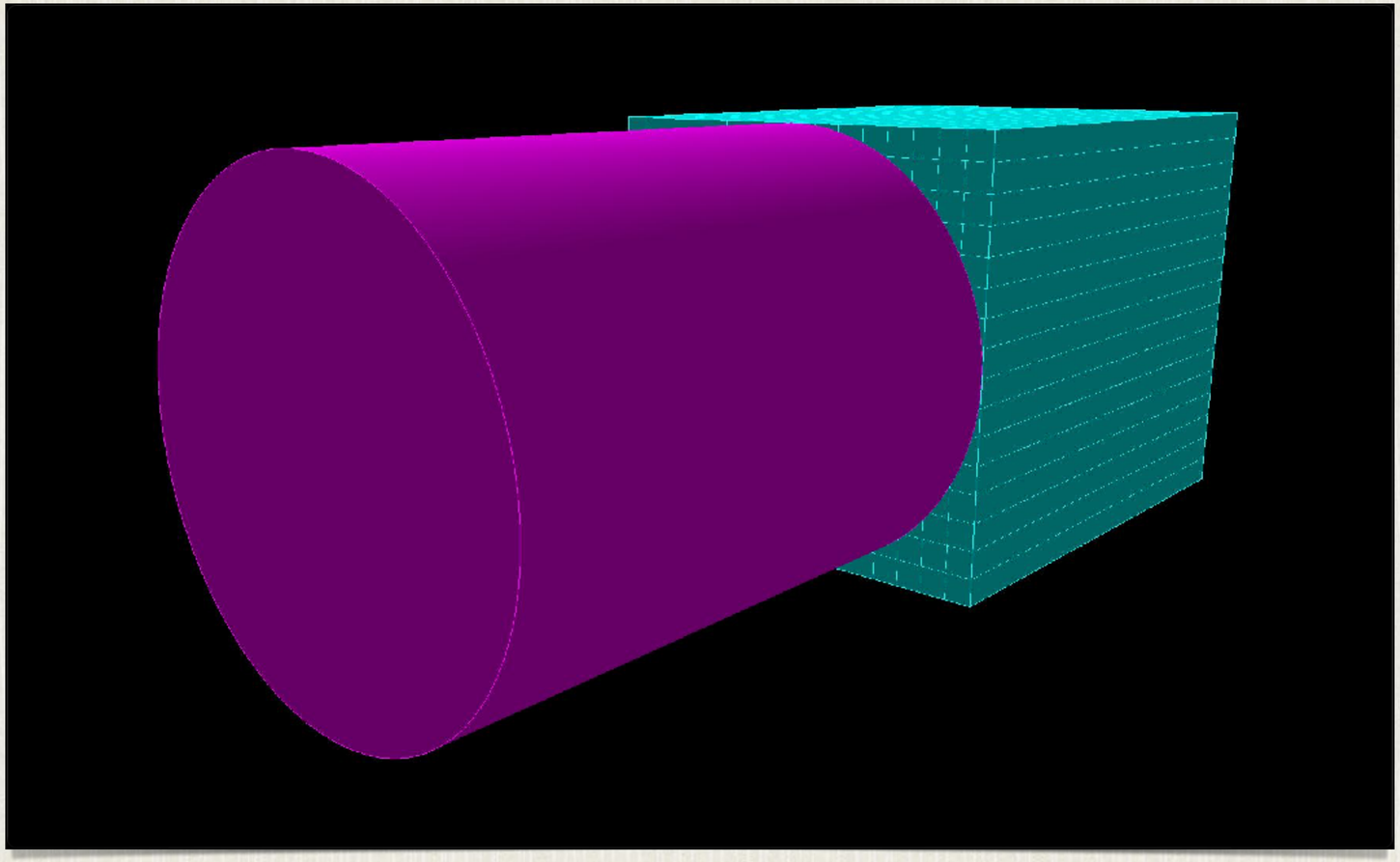
```
G4VPhysicalVolume *worldPV
   = new G4PVPlacement(nullptr, G4ThreeVector(), worldLV, "World", nullptr,
                       false, 0, fCheckOverlap);
```

G4PVPlacement (G4RotationMatrix *pRot, const G4ThreeVector &tlate, G4LogicalVolume *pCurrentLogical, const G4String &pName, G4LogicalVolume *pMotherLogical, G4bool pMany, G4int pCopyNo, G4bool pSurfChk=false)

- Copy number
- In case of placing same Logical volume at some different place, we need identifier of Physical volume
- Name is one solution
- Copy number is simple solution, you do not need to change the name

Same Logical volume and Same name

Different copy number is required



HOW TO DEFINE GEOMETRY

NaI cylinder detector

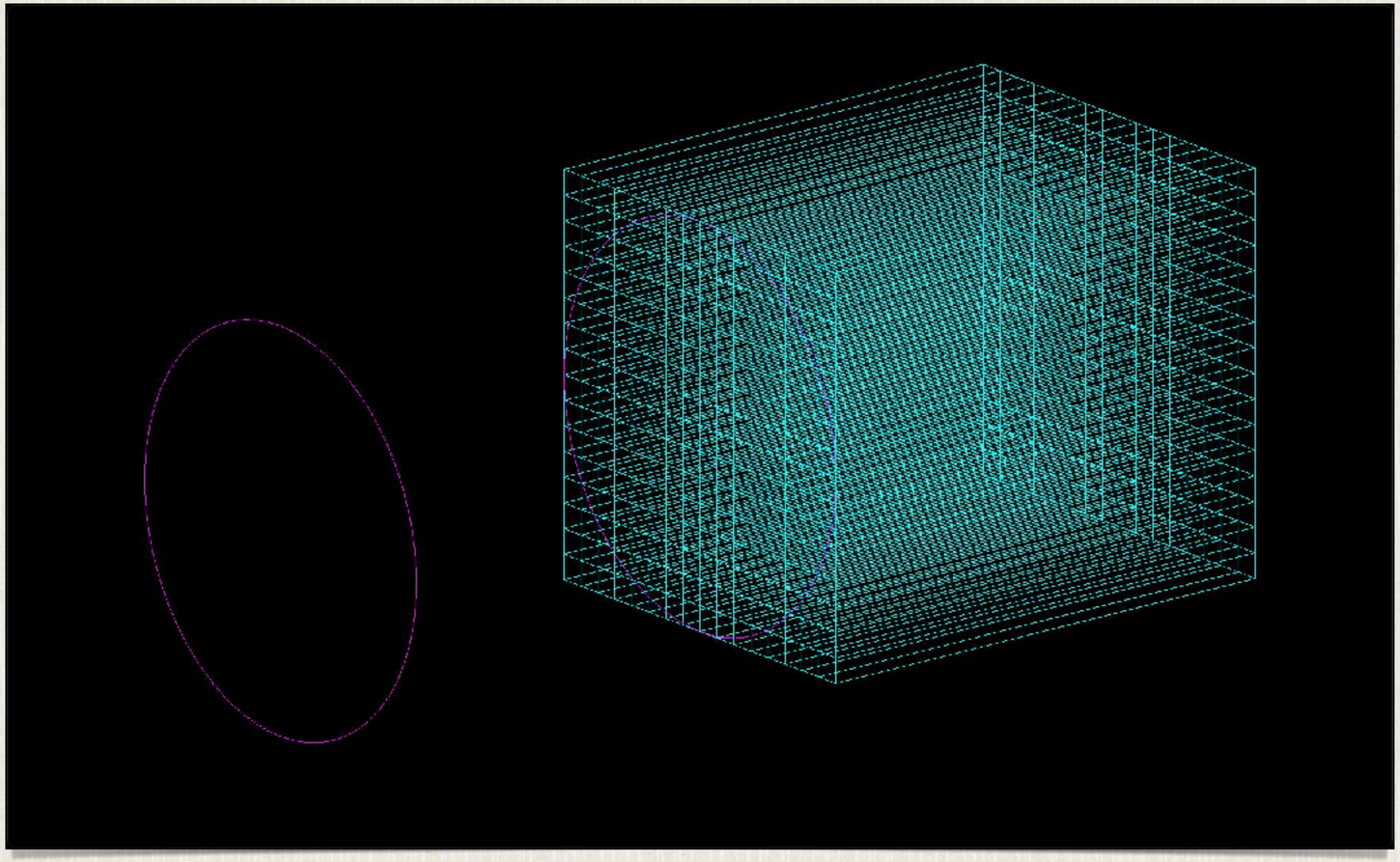
HOW TO DEFINE GEOMETRY

- Making cylinder shape NaI detector
- Using G4Tubs
- fScinti1T is member variable

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   …
   // NaI
   G4double scinti1D = 16.*mm;

   G4Tubs *scinti1S = new G4Tubs("NaI", 0., scinti1D / 2., fScinti1T / 2., 0., 360.*deg);
   G4LogicalVolume *scinti1LV = new G4LogicalVolume(scinti1S, fScinti1Mat, "NaI");
   G4ThreeVector scinti1Pos = G4ThreeVector(0., 0., -fScinti1T / 2.);
   fScinti1PV = new G4PVPlacement(nullptr, scinti1Pos, scinti1LV, "NaI", worldLV,
                                  false, 0, fCheckOverlap);
   …
}
```

G4Tubs (const G4String &pName, G4double pRMin, G4double pRMax, G4double pDz, G4double pSPhi, G4double pDPhi)



HOW TO DEFINE GEOMETRY

LGSO matrix detector

HOW TO DEFINE GEOMETRY

- First
- Make frame
- Using G4Box

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   …
   // LGSO
   G4double scinti2W = 16.*mm;
   G4double scinti2H = 16.*mm;

   G4Box *scinti2S = new G4Box("LGSO", scinti2W / 2., scinti2H / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2LV = new G4LogicalVolume(scinti2S, fScinti2Mat, "LGSO");

   G4ThreeVector scinti2Pos = G4ThreeVector(0., 0., fScinti2T / 2.);
   fScinti2PV = new G4PVPlacement(nullptr, scinti2Pos, scinti2LV, "LGSO", worldLV,
                                  false, 0, fCheckOverlap);
   …
}
```

HOW TO DEFINE GEOMETRY

- Next
- Make Matrix
- Using G4VReplica

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   …
   // Make LGSO matrix
   G4double meshSize = 1.*mm;

   G4Box *scinti2ColumnS
      = new G4Box("LGSOColumn", meshSize / 2., scinti2H / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2ColumnLV
      = new G4LogicalVolume(scinti2ColumnS, fScinti2Mat, "LGSOColumn");
  fScinti2ColumnPV = new G4PVReplica("LGSOColumn", scinti2ColumnLV,
                                     scinti2LV, kXAxis, 16, meshSize);

   G4Box *scinti2RowS
      = new G4Box("LGSORow", meshSize / 2., meshSize / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2RowLV
      = new G4LogicalVolume(scinti2RowS, fScinti2Mat, "LGSORow");
  fScinti2RowPV = new G4PVReplica("LGSORow", scinti2RowLV,
                                  scinti2ColumnLV, kYAxis, 16, meshSize);
  …
}
```

G4PVReplica (const G4String &pName, G4LogicalVolume *pLogical, G4LogicalVolume *pMother, const EAxis pAxis, const G4int nReplicas, const G4double width, const G4double offset=0)

HOW TO DEFINE GEOMETRY

- Make Column
- x-direction

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   …
   // Make LGSO matrix
   G4double meshSize = 1.*mm;

   G4Box *scinti2ColumnS
      = new G4Box("LGSOColumn", meshSize / 2., scinti2H / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2ColumnLV
      = new G4LogicalVolume(scinti2ColumnS, fScinti2Mat, "LGSOColumn");
  fScinti2ColumnPV = new G4PVReplica("LGSOColumn", scinti2ColumnLV,
                                     scinti2LV, kXAxis, 16, meshSize);

   G4Box *scinti2RowS
      = new G4Box("LGSORow", meshSize / 2., meshSize / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2RowLV
      = new G4LogicalVolume(scinti2RowS, fScinti2Mat, "LGSORow");
  fScinti2RowPV = new G4PVReplica("LGSORow", scinti2RowLV,
                                  scinti2ColumnLV, kYAxis, 16, meshSize);
  …
}
```

G4PVReplica (const G4String &pName, G4LogicalVolume *pLogical, G4LogicalVolume *pMother, const EAxis pAxis, const G4int nReplicas, const G4double width, const G4double offset=0)

HOW TO DEFINE GEOMETRY

- Make Row
- y-direction

```
G4VPhysicalVolume *MyDetectorConstruction::Construct()
{
   …
   // Make LGSO matrix
   G4double meshSize = 1.*mm;

   G4Box *scinti2ColumnS
      = new G4Box("LGSOColumn", meshSize / 2., scinti2H / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2ColumnLV
      = new G4LogicalVolume(scinti2ColumnS, fScinti2Mat, "LGSOColumn");
  fScinti2ColumnPV = new G4PVReplica("LGSOColumn", scinti2ColumnLV,
                                     scinti2LV, kXAxis, 16, meshSize);

   G4Box *scinti2RowS
      = new G4Box("LGSORow", meshSize / 2., meshSize / 2., fScinti2T / 2.);
   G4LogicalVolume *scinti2RowLV
      = new G4LogicalVolume(scinti2RowS, fScinti2Mat, "LGSORow");
  fScinti2RowPV = new G4PVReplica("LGSORow", scinti2RowLV,
                                  scinti2ColumnLV, kYAxis, 16, meshSize);
  …
}
```

G4PVReplica (const G4String &pName, G4LogicalVolume *pLogical, G4LogicalVolume *pMother, const EAxis pAxis, const G4int nReplicas, const G4double width, const G4double offset=0)

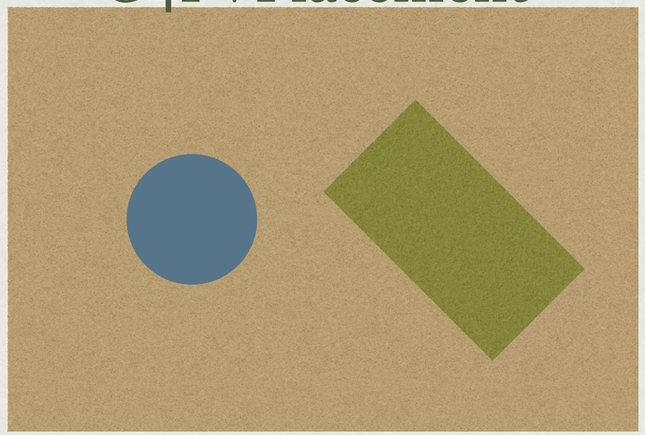
HOW TO DEFINE GEOMETRIES

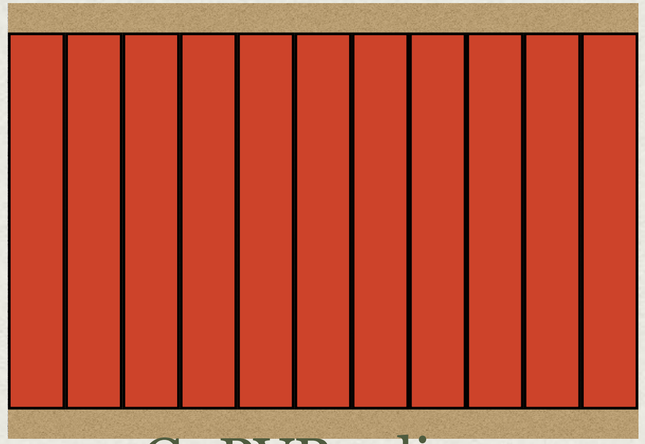
- Why using G4PVReplica?
- In this case, G4PVPlacement does not make problem
- But, in case of thousands number of mesh, Using G4PVPlacement makes problem
- G4PVPlacement makes same number of instance as meshes. Sometime it uses all memory space and make problem
- G4PVReplica makes only one instance. It is used at each position of mesh

HOW TO DEFINE GEOMETRIES

- The difference between G4PVPlacement and G4PVReplica
- G4PVPlacement: You can please any shape of volume at any position
- G4PVReplica: You can place only one volume without any gap between volumes

G4PVPlacement

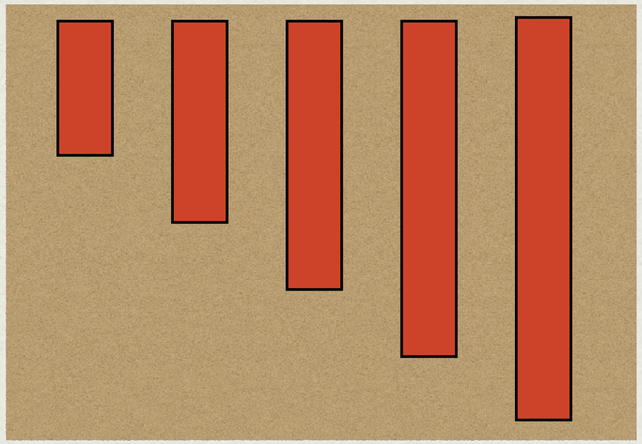




G4PVReplica

HOW TO DEFINE GEOMETRIES

- Between G4PVPlacement and G4PVReplica, we can use G4PVParametrized
- This also reduce memory used



G4PVParametrized

HOW TO ADD UI COMMANDS

HOW TO ADD UI COMMANDS

- Using G4GenericMessenger class
- For example, adding command for changing detector length

HOW TO ADD UI COMMANDS

- G4GenericMesseger class has DeclearMethod* member functions

```
MyDetectorConstruction::MyDetectorConstruction()
{
   …
   DefineCommands();
   …
}

void MyDetectorConstruction::DefineCommands()
{
   fMessenger = new G4GenericMessenger(this, "/MyCommands/Geo/",
                                       "Scinti control");

   // Scinti 1
   G4GenericMessenger::Command &scinti1TCmd
      = fMessenger->DeclareMethodWithUnit("Scinti1T", "mm",
                                          &MyDetectorConstruction::SetScinti1T,
                                          "Set the thickness of NaI.");
   scinti1TCmd.SetParameterName("t", true);
   scinti1TCmd.SetRange("t>=0. && t<100.");
   scinti1TCmd.SetDefaultValue("20.0");

   // Scinti 2
   G4GenericMessenger::Command &scinti2TCmd
      = fMessenger->DeclareMethodWithUnit("Scinti2T", "mm",
                                          &MyDetectorConstruction::SetScinti2T,
                                          "Set the thickness of NaI.");
   scinti2TCmd.SetParameterName("t", true);
   scinti2TCmd.SetRange("t>=0. && t<100.");
   scinti2TCmd.SetDefaultValue("20.0");

}
```

HOW TO ADD UI COMMANDS

- In the command function (Scinti1T)
- Taking Solid from PV
- And changing the size

```
void MyDetectorConstruction::DefineCommands()
{
    …
    // Scinti 1
    G4GenericMessenger::Command &scinti1TCmd
       = fMessenger->DeclareMethodWithUnit("Scinti1T", "mm",
                                           &MyDetectorConstruction::SetScinti1T,
                                           "Set the thickness of NaI.");
    scinti1TCmd.SetParameterName("t", true);
    scinti1TCmd.SetRange("t>=0. && t<100.");
    scinti1TCmd.SetDefaultValue("20.0");
    …
}

void MyDetectorConstruction::SetScinti1T(G4double t)
{
    fScinti1T = t;
    G4Box *scinti = (G4Box*)(fScinti1PV->GetLogicalVolume()->GetSolid());
    scinti->SetZHalfLength(fScinti1T / 2.);
    fScinti1PV->SetTranslation(G4ThreeVector(0., 0., -fScinti1T / 2.));

    G4RunManager::GetRunManager()->GeometryHasBeenModified();
}
```

HOW TO ADD UI COMMANDS

- In the command function (Scinti1T)
- Changing the position
- Informing to run manager

```
void MyDetectorConstruction::DefineCommands()
{
   …
   // Scinti 1
   G4GenericMessenger::Command &scinti1TCmd
      = fMessenger->DeclareMethodWithUnit("Scinti1T", "mm",
                                          &MyDetectorConstruction::SetScinti1T,
                                          "Set the thickness of NaI.");
   scinti1TCmd.SetParameterName("t", true);
   scinti1TCmd.SetRange("t>=0. && t<100.");
   scinti1TCmd.SetDefaultValue("20.0");
   …
}

void MyDetectorConstruction::SetScinti1T(G4double t)
{
   fScinti1T = t;
   G4Box *scinti = (G4Box*)(fScinti1PV->GetLogicalVolume()->GetSolid());
   scinti->SetZHalfLength(fScinti1T / 2.);
   fScinti1PV->SetTranslation(G4ThreeVector(0., 0., -fScinti1T / 2.));

   G4RunManager::GetRunManager()->GeometryHasBeenModified();
}
```

HOW TO ADD UI COMMANDS

- For command you can input not only values, but also strings
- For example, changing material of volume, input the material name and change it
- Please check the class reference

CONCLUSION

- The materials can be taken from the data base
- In case of no information in the data base, you can make
- The volumes should be hierarchical structure
  - The top volume is returned by Construct member function
- In case of the number of volumes is low, G4PVPlacement is easy to understand and manage volume
- In case of high, using G4PVReplica or G4PVParametrized are solutions for reduce the memory used

CONCLUSION

- UI commands help you
  - Taking data with various thickness scintillator
  - Various materials
  - Various positions
  - etc.

CONCLUSION

- In the DetectorConstruction class
- We can define the detector
- It records some information about the interaction
- In the next session, I will talk about this