提　纲

- 一、Geant4运行模式
  - “purely hard-coded” batch mode
  - Batch mode，macro commands
  - Interactive mode，command lines
- 二、材料定义
  - 简单物质
  - 分子定义
  - 混合物
  - 同位素
  - Geant4自定义
- 三、Geant4中使用物理量的单位
  - 基本单位
  - 输入
  - 输出
- 四、几何结构定义
  - 简单结构定义
  - 复杂结构定义
  - 颜色显示
- 五PrimaryGeneratorAction

核技术论坛
www.hejishult.cn

核技术论坛
www.hejishult.cn

一、Geant4运行模式

- Geant4运行模式：
  - “purely hard-coded” batch mode
  - Batch mode，macro commands
  - Interactive mode，command lines

每种模式的选择必须在Main文件中编制相应的代码以实现其对应模式。通常采用混合模式。

…….

头文件 **Purely batch-coded mode**

…….

```
int main()
{
. . ….. G4runManager初始化…
. . ….. class初始化…
………G4核初始化…
  runManager->Initialize();
//start run
G4int numberOfEvent = 3;
 runManager->BeamOn(numberOfEvent);
delete runManager;
  return 0;
}
```

…….

头文件 **Batch-coded with macro file mode**

…….

```
int main(int argc,char** argv)
{
. . ….. G4runManager初始化…
. . ….. class初始化…
………G4核初始化…
  runManager->Initialize();
// Get the pointer to the UI manager
G4UImanager* UI = G4UImanager::GetUIpointer();
G4String command = "/control/execute ";
G4String fileName = argv[1];
UI->applyCommand(command+fileName);
delete runManager;
  return 0;
}
```

核技术论坛 www.hejishult.cn

……

头文件 **Interactive by command line mode**

……

- #include "G4UIterminal.hh"
- #include "G4UIsession.hh"
- int main(int argc,char** argv)
- {
- . . ….. G4runManager初始化…
- . . ….. class初始化…
- ………G4核初始化…
- runManager->Initialize();

// Terminal initialization;

G4UIsession* session = new G4UIterminal;

session->SessionStart();

- delete runManager;
- return 0;
- }

Macro file:

命令行序列:

- #2008.6.9
- #mac file for visulization
- /vis/scene/create
- /vis/open OGLIX
- /vis/scene/add/trajectories
- /vis/scene/add/hits
- /vis/scene/endOfEventAction accumulate
- /vis/viewer/set/viewpointThetaPhi 72 25
- /vis/viewer/zoom 16
- /run/beamOn 10000
- /vis/viewer/set/style w
- /vis/viewer/set/style s
- 以文本格式保存

核技术论坛
www.hejishult.cn

Mixed mode

核技术论坛
www.hejishult.cn

```
#include "G4UImanager.hh "
#include "G4RunManager.hh"
……………
#include "G4VisManager.hh"
#include "G4VisExecutive.hh"
#include "G4UItcsh.hh"
#include "G4UIterminal.hh"
#include "G4UIsession.hh"
int main(int argc,char** argv)
{
G4RunManager* MyRun = new G4RunManager;
//this part is for the visualization
G4VisManager* VisManager = new G4VisExecutive;
VisManager->Initialize();
……………………
……………………
MyRun->Initialize();
//G4UIsession* session = new G4UIterminal(new G4UItcsh);
```

Mixed mode

```
G4UImanager* UI = G4UImanager::GetUIpointer();
if(argc==1)
{       UI->ApplyCommand("/run/verbose 2");
G4String command="/control/execute /g4work/pphotonelectron/vis.mac";
        UI->ApplyCommand(command);
        //session->SessionStart();
}
else
{       UI->ApplyCommand("/run/verbose 2");
        G4String command="/control/execute /g4work/pphotonelectron/";
        G4String fileName = argv[1];
        UI->ApplyCommand(command+fileName);
        //session->SessionStart();
        }
//delete session;
delete VisManager;
delete MyRun;
return 0;
}
```

核技术论坛
www.hejishult.cn

核技术论坛
www.hejishult.cn

二、材料定义

- 一般的物质（化合物，混合物）是由元素构成，元素又由同位素组成。按照这种物质的划分层次，可以通过类似的概念定义宏观世界中的物质。
- Geant4中定义的物质类（class）有两种：
  - G4Element
  - G4Material

- G4Element class 描述微观层面的原子的性质如:
    - 原子序数
    - 核子数
    - 壳层能量
    - 诸如原子截面等
- G4Isotope class 描述微观层面的原子的性质如:
    - 原子序数
    - 核子数
    - 莫尔质量等
- G4Material class 描述微观层面的原子的性质如:
    - 原子序数
    - 核子数
    - 壳层能量
    - 诸如原子截面等

核技术论坛
www.hejishult.cn

简单物质定义：

```
G4double density = 1.390*g/cm3;
G4double a = 39.95*g/mole;
G4Material* lAr = new G4Material(
                    name="liquidArgon", z=18., a, density);
```

分子定义：

```
a = 1.01*g/mole;
G4Element* elH = new G4Element(
                    name="Hydrogen",symbol="H" , z= 1., a);
a = 16.00*g/mole;
G4Element* elO = new G4Element(
                    name="Oxygen" ,symbol="O" , z= 8., a);
density = 1.000*g/cm3;
G4Material* H2O = new G4Material(
                    name="Water",density,ncomponents=2);
H2O->AddElement(elH, natoms=2);
H2O->AddElement(elO, natoms=1);
```

核技术论坛
www.hejishult.cn

混合物定义（质量份数）：

- a = 14.01*g/mole;
- G4Element* elN = new G4Element(
  name="Nitrogen",symbol="N" , z= 7., a);
- a = 16.00*g/mole;
- G4Element* elO = new G4Element(
  name="Oxygen" ,symbol="O" , z= 8., a);
- density = 1.290*mg/cm3;
- G4Material* Air = new G4Material(
  name="Air ",density,ncomponents=2);
- Air->AddElement(elN, fractionmass=70*perCent);
- Air->AddElement(elO, fractionmass=30*perCent);

核技术论坛
www.hejishult.cn

Geant4自定义：

- G4NistManager* man = G4NistManager::Instance();
- G4Material* H2O = man->FindOrBuildMaterial("G4_WATER");
- G4Material* Air = man->FindOrBuildMaterial("G4_AIR");

同位素定义：

- G4Isotope* U5 = new G4Isotope(
  name="U235", iz=92, n=235, a=235.01*g/mole);
- G4Isotope* U8 = new G4Isotope(
  name="U238", iz=92, n=238, a=238.03*g/mole);
- G4Element* elU = new G4Element(
  name="enriched Uranium", symbol="U", ncomponents);
- elU->AddIsotope(U5, abundance= 90.*perCent);
- elU->AddIsotope(U8, abundance= 10.*perCent);

核技术论坛
www.hejishult.cn

三、Geant4中使用物理量的单位

核技术论坛
www.hejishult.cn

基本单位：

Geant4中用户可以为选定的物理量选择各种单位，但是Geant4内核内定义了如下基本单位：

- millimeter (mm)
- nanosecond (ns)
- Mega electron Volt (MeV)
- positron charge (eplus)
- degree Kelvin (kelvin)
- the amount of substance (mole)
- luminous intensity (candela)
- radian (radian)
- steradian (steradian)

其它所有单位都以上述单位为基础得到，如：

Millimeter＝ mm ＝1；

Meter ＝ m ＝1000*mm；

核技术论坛
www.hejishult.cn

用户输入数据单位：

- Geant4中用户输入的数据必须带有单位（系统默认强烈建议不使用），如：
- G4double Size = 15*km, KineticEnergy = 90.3*GeV, density = 11*mg/cm3;
- 同样，如果数据为数组格式或者从文件读入，也必须带有单位，如：
- for (int j=0, j<jmax, j++) CrossSection[j] *= millibarn;

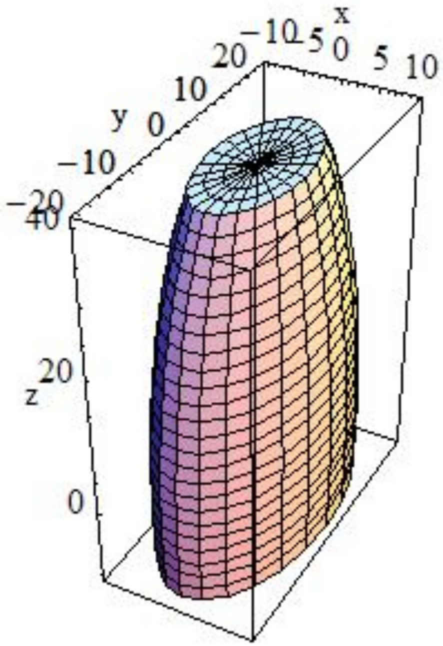
数据输出单位：

- 带单位数据输出格式如：
- G4cout << KineticEnergy/keV << " keV";
- G4cout << density/(g/cm3) << " g/cm3";
- 或者给出Geant4选择的最优化单位：
- G4cout << G4BestUnit(StepSize, "Length");

核技术论坛
www.hejishult.cn

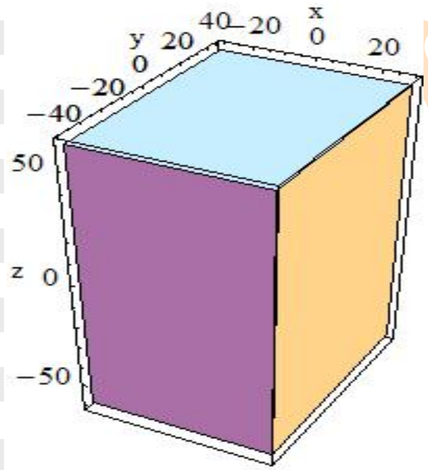
四、几何结构定义

- Geant4中以“体”（Volume）的概念定义几何形状。模型中最大的“体”称为“世界体”（World Volume），其它的体都位于世界体内部。“世界体”内的“体”之间是包含和被包含关系，被包含的称包含它的体为母体（mother Volume）。

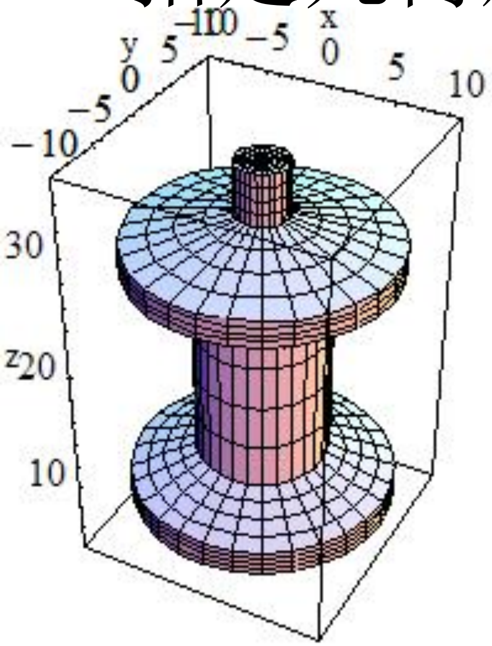


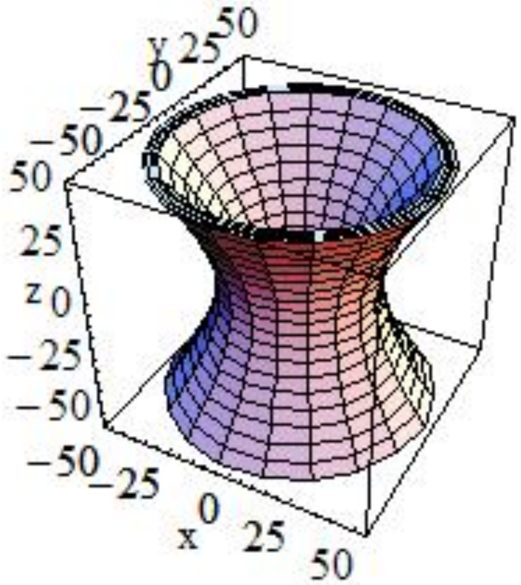
- Geant4中的体的定义包含三个层次：
  - 1.定义几何形状；
  - 2.定义物理属性；
  - 3.定义所在母体的位置；

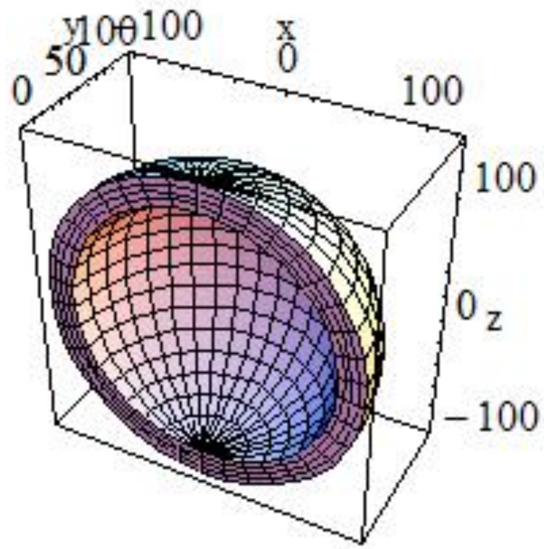
- 定义几何形状（solid）：
  - G4double expHall_x = 30.0*m;
  - G4double expHall_y = 40.0*m;
  - G4double expHall_z = 60.0*m;
  - G4Box* experimentalHall_box
  - = new G4Box("expHall_box",expHall_x,expHall_y,expHall_z);



- 指定几何形状的名称、尺寸等信息。







- 定义物理属性：

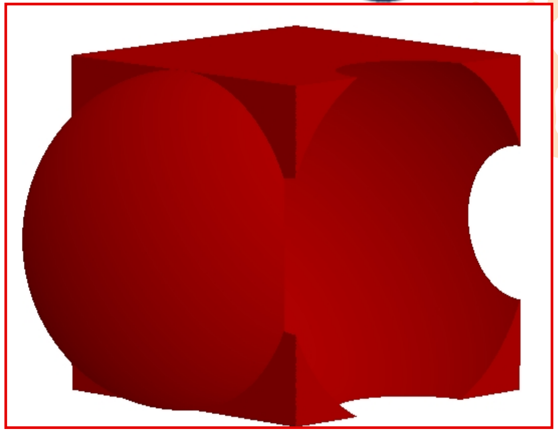
G4LogicalVolume* experimentalHall_log
= new G4LogicalVolume(
experimentalHall_box,Ar,"expHall_log");

给定物理属性指向的几何体，该几何体对应的物质，该几何体的名称等信息。

G4LogicalVolume* tracker_log = new G4LogicalVolume(tracker_tube,Al,"tracker_log");

<指针传递>

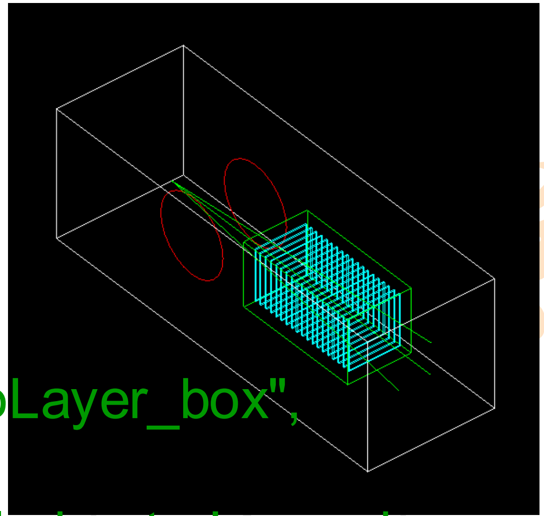
定义物理位置



```
G4double trackerPos_x = -1.0*meter;
G4double trackerPos_y =  0.0*meter;
G4double trackerPos_z =  0.0*meter;
G4VPhysicalVolume* tracker_phys
  = new G4PVPlacement(0,                        // no rotation
           G4ThreeVector(trackerPos_x,trackerPos_y,trackerPos_z),
                                   // translation position
           tracker_log,            // its logical volume
           "tracker",              // its name
           experimentalHall_log,   // its mother (logical) volume
           false,                  // no boolean operations
           0);                     // its copy number
```

复杂几何体结构

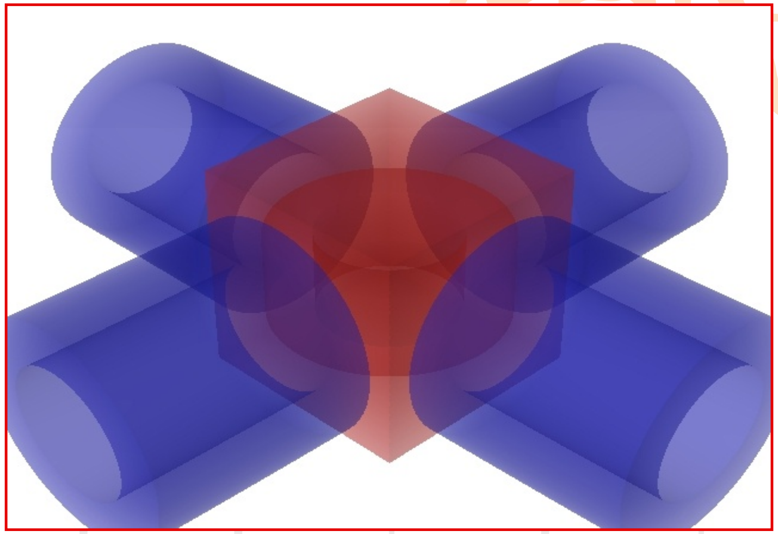
- C＋＋循环设计



```
G4double calo_x = 1.*cm;
G4double calo_y = 40.*cm;
G4double calo_z = 40.*cm;
G4Box* calorimeterLayer_box = new G4Box("caloLayer_box",
                              calo_x,calo_y,calo_z);
calorimeterLayer_log = new G4LogicalVolume(calorimeterLayer_box,
                              Al,"caloLayer_log",0,0,0);
for(G4int i=0;i<19;i++) // loop for 19 layers
{
  G4double caloPos_x = (i-9)*10.*cm;
  G4double caloPos_y = 0.0*m;
  G4double caloPos_z = 0.0*m;
  calorimeterLayer_phys = new G4PVPlacement(0,
        G4ThreeVector(caloPos_x,caloPos_y,caloPos_z),
        calorimeterLayer_log,"caloLayer",calorimeterBlock_log,false,i);
}
```

- Replica 和 Parameterised Volumes 方法
- Division

- 几何结构显示的颜色和半透明处理



```
#include "G4VisAttributes.hh"
#include "G4Colour.hh "
………
………
/**Colour of the volume**/
experimentalHall_log->SetVisAttributes(G4Colour(0,0,0));
tracker_log->SetVisAttributes(G4Colour(1,0,0));
calorimeterBlock_log->SetVisAttributes(G4Colour(0,2,0,0));
calorimeterLayer_log->SetVisAttributes(G4Colour(0,1,1,0));
```

RGB配色，G4Colour(0,1,1,0)

核技术论坛
www.hejishult.cn

五、PrimaryGeneratorAction

- G4VUserPrimaryGeneratorAction是Geant4要求的强制类之一，通过该类指定产生的主粒子事件。在该类中，用户可以指定粒子产生的方式。

- 在G4VUserPrimaryGeneratorAction中，用户必须对主粒子产生器实例化。实际上，在G4VUserPrimaryGeneratorAction中，用户定义的是主粒子产生的排列顺序及初始化设定。

核技术论坛
www.hejishult.cn

产生主粒子事件的方法

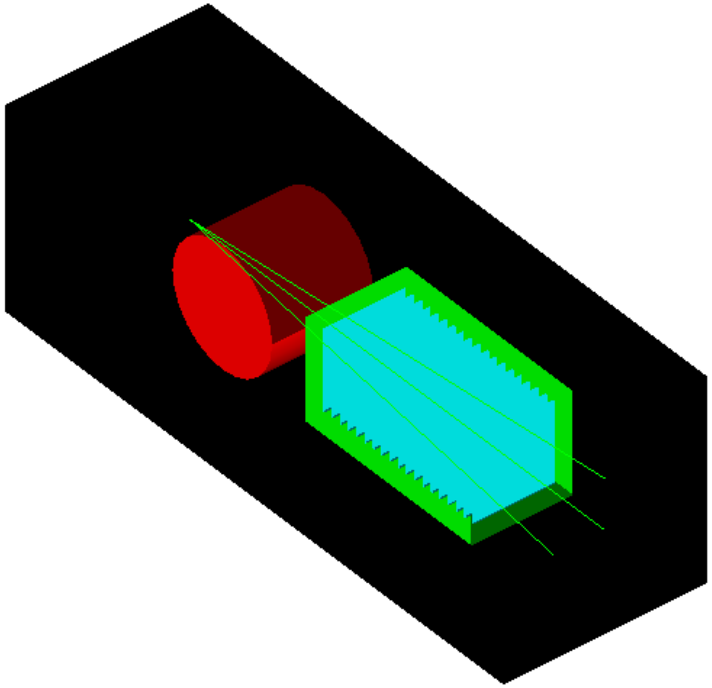
- ParticleGun
  - 射线束，可以选择粒子类型、能量、射线束极化度等信息
- G4HEPEvtInterface
  - 许多是以FORTRAN编写的高能物理过程所需的粒子事件
  - ASCII 文件输入
- GeneralParticleSource（GPS）
  - 提供了很多方便的函数描述粒子源
  - 模拟空间环境、地下放射性源等；（类似于现实中的源）
- 通过继承 *G4VUserPrimaryGeneratorAction*类自己写

PrimaryGeneratorAction示例（N01）

- 例子N01中PrimaryGeneratorAction使用ParticleGun作为的主粒子产生器

每次产生一个粒子（Geantino），图中绿色线条；

GPS的使用和ParticleGun类似，替代即可；



PrimaryGeneratorAction.hh N01

```
#ifndef ExN01PrimaryGeneratorAction_h
#define ExN01PrimaryGeneratorAction_h 1

#include "G4VUserPrimaryGeneratorAction.hh"

class G4ParticleGun;
class G4Event;

class ExN01PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction

{

public:

        ExN01PrimaryGeneratorAction();
        ~ExN01PrimaryGeneratorAction();

public:

        void generatePrimaries(G4Event* anEvent);

private:

        G4ParticleGun* particleGun;

};

#endif
```

ParticleGun实例化

PrimaryGeneratorAction.cc N01

```
#include "ExN01PrimaryGeneratorAction.hh"
#include "G4Event.hh"
#include "G4ParticleGun.hh"
#include "G4ThreeVector.hh"
#include "G4Geantino.hh"
#include "globals.hh"
ExN01PrimaryGeneratorAction::ExN01PrimaryGeneratorAction()
{
G4int n_particle = 1;
particleGun = new G4ParticleGun(n_particle);
particleGun->SetParticleDefinition(
                    G4Geantino::GeantinoDefinition());
particleGun->SetParticleEnergy(1.0*GeV);
particleGun->SetParticlePosition(G4ThreeVector(-2.0*m,0.0*m,0.0*m));
}
ExN01PrimaryGeneratorAction::~ExN01PrimaryGeneratorAction()
{
    delete particleGun;
}
```

核技术论坛
www.hejishult.cn

PrimaryGeneratorAction.cc N01

```
void ExN01PrimaryGeneratorAction::generatePrimaries(G4Event* anEvent)
{
G4int i = anEvent->get_eventID() % 3;
switch(i)
{
case 0:
particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.0,0.0));
break;
case 1:
particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.1,0.0));
break;
case 2:
particleGun->SetParticleMomentumDirection(G4ThreeVector(1.0,0.0,0.1));
break;
}
particleGun->generatePrimaryVertex(anEvent);
}
```

G4ParticleGun 方法

- void SetParticleDefinition(G4ParticleDefinition*)
- void SetParticleMomentum(G4ParticleMomentum)
- void SetParticleMomentumDirection(G4ThreeVector)
- void SetParticleEnergy(G4double)
- void SetParticleTime(G4double)
- void SetParticlePosition(G4ThreeVector)
- void SetParticlePolarization(G4ThreeVector)
- void SetNumberOfParticles(G4int)

G4GeneralParticleSource

2D Surface sources	3D Surface sources	Volume sources	Angular distribution	Energy spectrum
• circle • ellipse • square • rectangle • Gaussian beam profile	• sphere • ellipsoid • cylinder • paralellapiped (incl. cube & cuboid)	• sphere • ellipsoid • cylinder • paralellapiped (incl. cube & cuboid)	• isotropic • cosine-law • user-defined (through histograms)	• mono-energetic • Gaussian • Linear • Exponential • power-law • bremsstrahlung • black-body • CR diffuse • user-defined (through histograms or point-wise data)

- 注意ParticleGun和GPS的区别（很明显吧^_^）
- 在模拟放射源时，使用GPS可以带来更多的方便，另外它还提供多种分布抽样
- 当然使用ParticleGun对方向抽样也行，此时需要通过随机数根据需要编写抽样函数即可
- ParticleGun多用于一些打靶实验，此外，很多时候它都需要配合抽样函数方便使用