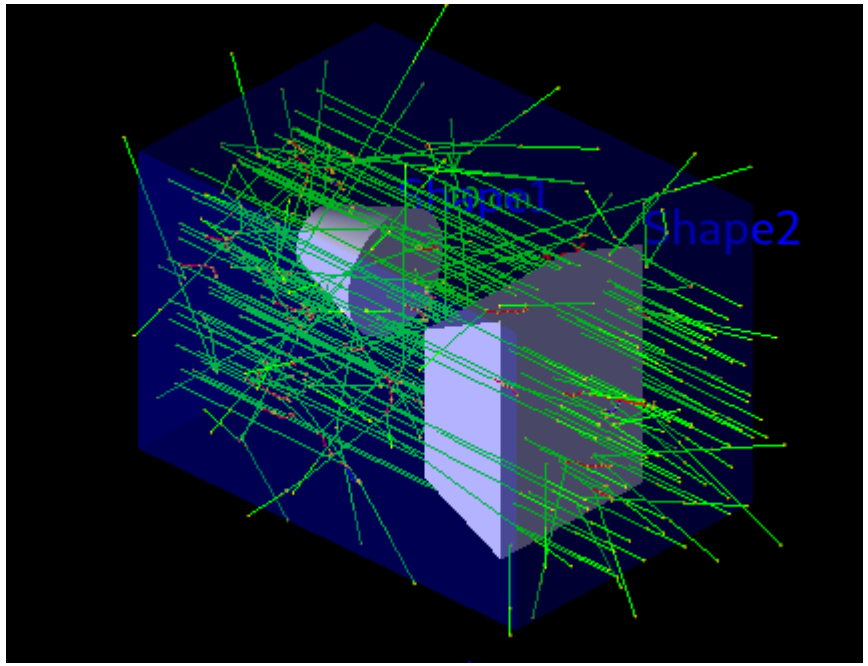
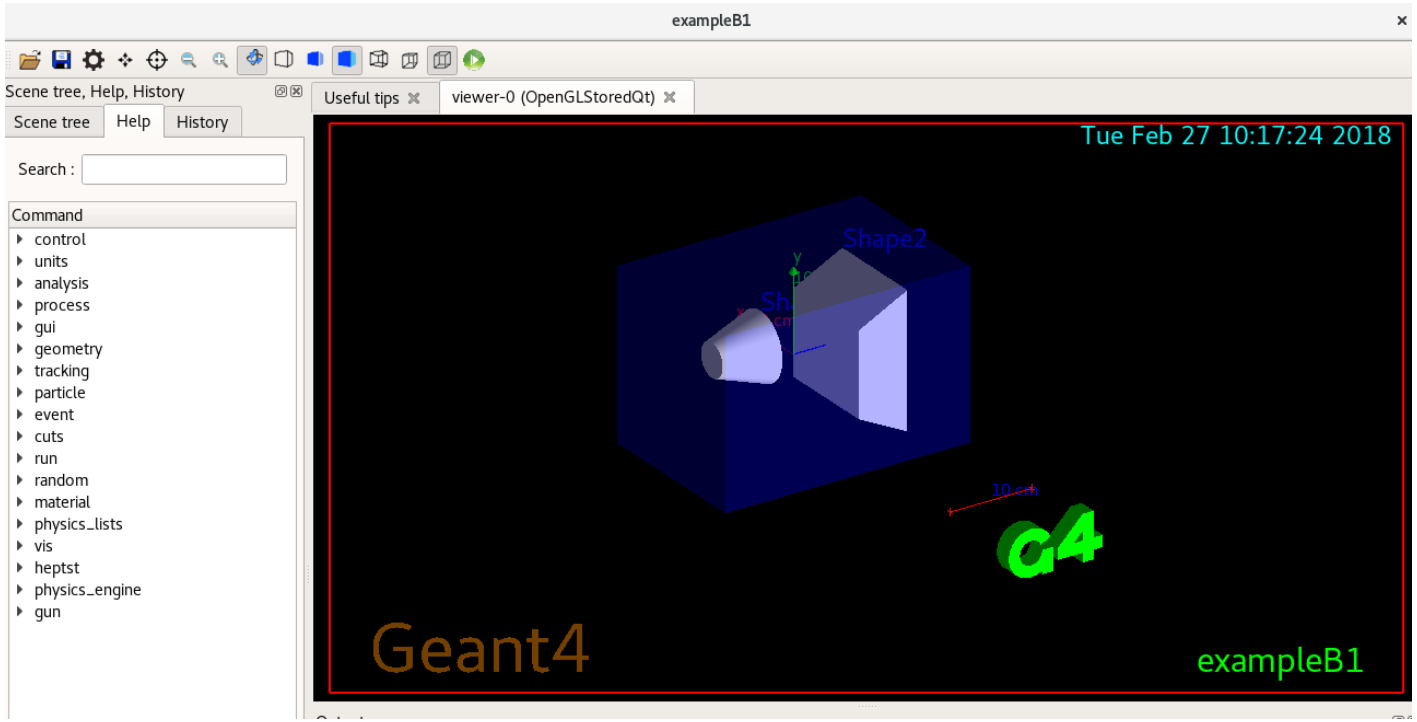
# Basic -> ExampleB1

例子编号：1

整理人：王思广

- Simple geometry with a few solids
- Geometry with simple placements (G4PVPlacement)
- Scoring total dose in a selected volume user action classes
- Geant4 physics list (QBBC)



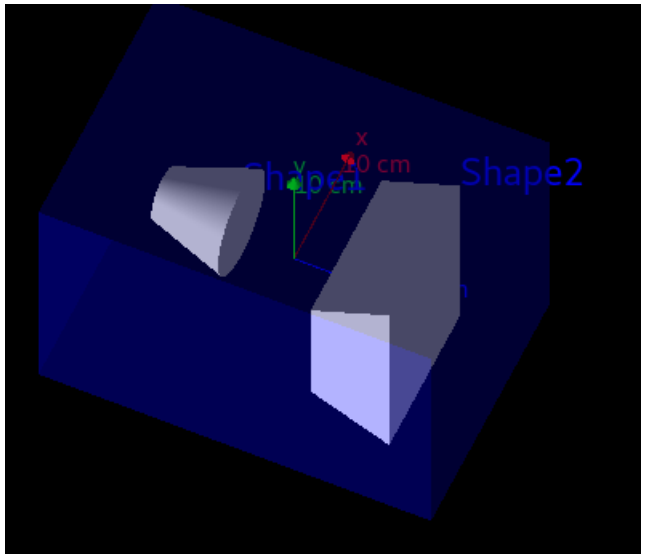

siguang@pku.edu.cn

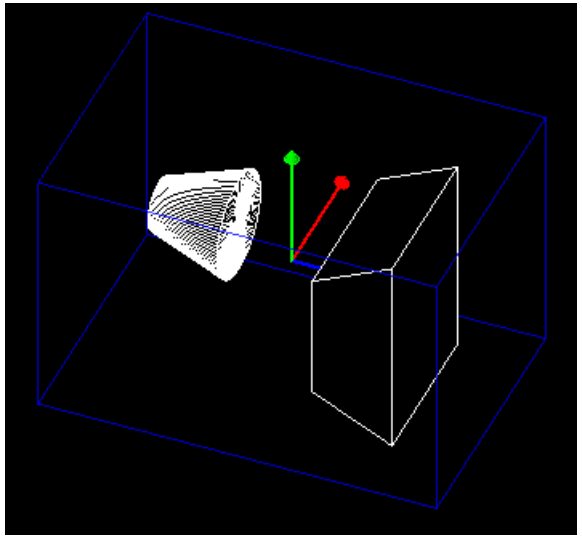
# 建立文件夹、编译运行

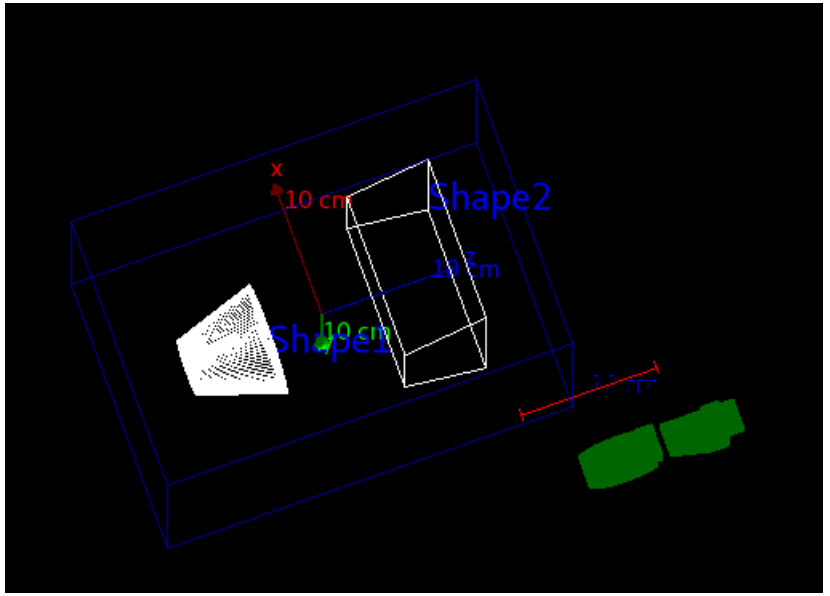
不同版本及安装路径略有不同

- cd g4root@debian:~/work/g4/10.4/examples/basic/B1$
- cd g4root@debian:~/work/g4/10.4/examples/basic/B1$mkdir bin
- g4root@debian:~/work/g4/10.4/examples/basic/B1$ cd bin/
- g4root@debian:~/work/g4/10.4/examples/basic/B1/bin$ cmake ../
- g4root@debian:~/work/g4/10.4/examples/basic/B1/bin$ make
- g4root@debian:~/work/g4/10.4/examples/basic/B1/bin$ ./exampleB1

# 输出图形





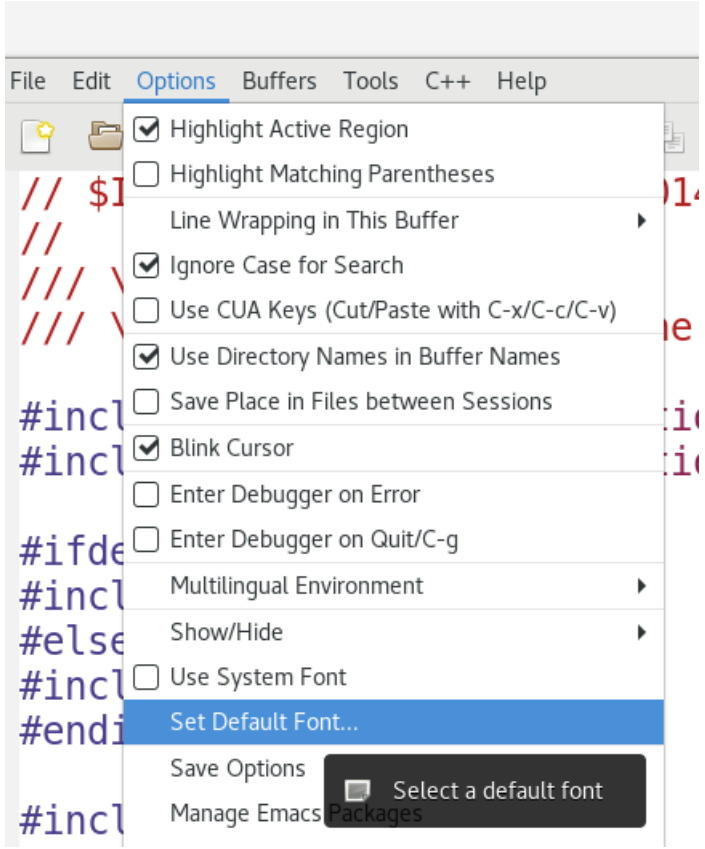
# 文件结构

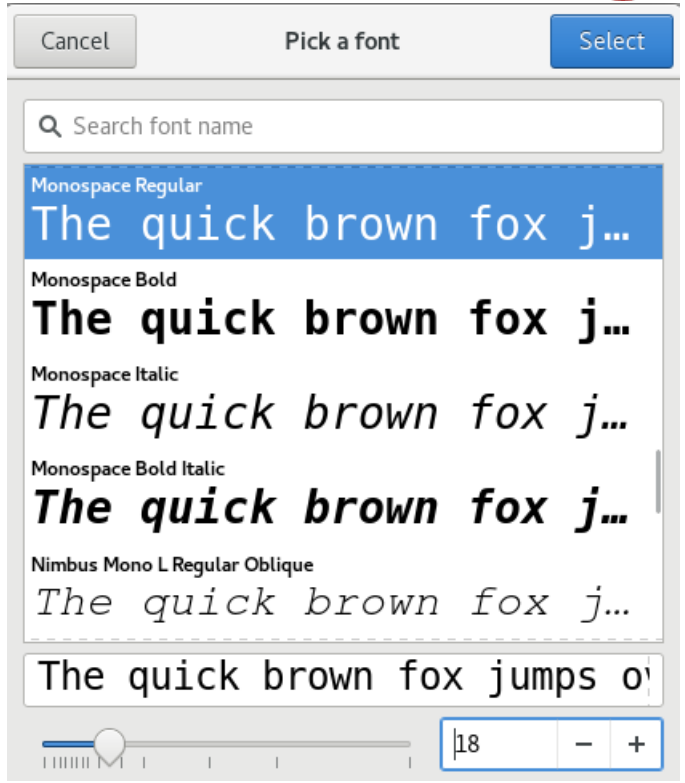
- exampleB1.cc
  - include$ ls
    - B1ActionInitialization.hh
    - B1DetectorConstruction.hh
    - B1EventAction.hh
    - B1PrimaryGeneratorAction.hh
    - B1RunAction.hh
    - B1SteppingAction.hh
  - src$ ls
    - B1ActionInitialization.cc
    - B1DetectorConstruction.cc
    - B1EventAction.cc
    - B1PrimaryGeneratorAction.cc
    - B1RunAction.cc
    - B1SteppingAction.cc

# 保障格式统一的特殊说明（同学们可以去掉本页）

```
#include "B1DetectorConstruction.hh"
#include "B1ActionInitialization.hh"
```

截取本程序特写的include文件名，以确保一级级找下去相互之间的关系




缺省的字体
18号大小

# exampleB1.cc

```
#include "B1DetectorConstruction.hh"
#include "B1ActionInitialization.hh"

int main(int argc,char** argv)
{
  // Detect interactive mode (if no arguments) and define UI session
  //
  G4UIExecutive* ui = 0;
  if ( argc == 1 ) {
    ui = new G4UIExecutive(argc, argv);
  }

  // Choose the Random engine
  G4Random::setTheEngine(new CLHEP::RanecuEngine);

  // Construct the default run manager
  //
#ifdef G4MULTITHREADED
  G4MTRunManager* runManager = new G4MTRunManager;
#else
  G4RunManager* runManager = new G4RunManager;
#endif
```

直接运行$./exampleB1
argc=1

这个变量是否定义与安装G4的时候的cmake选项有关：
-DGEANT4_BUILD_MULTITHREADED=ON

多线程运行

单线程，高能的同学在服务器上尽量用这个

G4.11.1.1:

```
// Construct the default run manager
//
auto* runManager =
  G4RunManagerFactory::CreateRunManager(G4RunManagerType::Default);
```

# exampleB1.cc （续）

```
// Set mandatory initialization classes
//
// Detector construction
runManager->SetUserInitialization(new B1DetectorConstruction());

// Physics list
G4VModularPhysicsList* physicsList = new QBBC;
physicsList->SetVerboseLevel(1);
runManager->SetUserInitialization(physicsList);

// User action initialization
runManager->SetUserInitialization(new B1ActionInitialization());

// Initialize visualization
//
G4VisManager* visManager = new G4VisExecutive;
// G4VisExecutive can take a verbosity argument - see /vis/verbose guidance.
// G4VisManager* visManager = new G4VisExecutive("Quiet");
visManager->Initialize();
```

electromagnetic and hadronic processes

# exampleB1.cc （续）

```
// Get the pointer to the User Interface manager
G4UImanager* UImanager = G4UImanager::GetUIpointer();

// Process macro or start UI session
//
if ( ! ui ) {
  // batch mode
  G4String command = "/control/execute ";
  G4String fileName = argv[1];
  UImanager->ApplyCommand(command+fileName);
}
else {
  // interactive mode
  UImanager->ApplyCommand("/control/execute init_vis.mac");
  ui->SessionStart();
  delete ui;
}

// Job termination
// Free the store: user actions, physics_list and detector_description are
// owned and deleted by the run manager, so they should not be deleted
// in the main() program !

delete visManager;
delete runManager;
```

运行可视化界面，直到用exit命令退出

# include-> B1DetectorConstruction.hh

```
class B1DetectorConstruction : public G4VUserDetectorConstruction
{
  public:
    B1DetectorConstruction();
    virtual ~B1DetectorConstruction();

    virtual G4VPhysicalVolume* Construct();

    G4LogicalVolume* GetScoringVolume() const { return fScoringVolume; }

  protected:
    G4LogicalVolume*  fScoringVolume;
};
```

# src-> B1DetectorConstruction.cc

```
G4VPhysicalVolume* B1DetectorConstruction::Construct()
{
  // Get nist material manager
  G4NistManager* nist = G4NistManager::Instance();

  // Envelope parameters
  //
  G4double env_sizeXY = 20*cm, env_sizeZ = 30*cm;
  G4Material* env_mat = nist->FindOrBuildMaterial("G4_WATER");

  // Option to switch on/off checking of volumes overlaps
  //
  G4bool checkOverlaps = true;

  //
  // World
  //
  G4double world_sizeXY = 1.2*env_sizeXY;
  G4double world_sizeZ  = 1.2*env_sizeZ;
  G4Material* world_mat = nist->FindOrBuildMaterial("G4_AIR");
```

水
Define a Material from the GEANT4 Material Database

空气

# NIST名字

- https://geant4-userdoc.web.cern.ch/UsersGuides/ForApplicationDeveloper/fo/BookForApplicationDevelopers.pdf （版本不断更新）

### 11.6.2 NIST Compounds

```
 Ncomp              Name       density(g/cm^3)   I(eV) ChFormula
 6             G4_A-150_TISSUE       1.127       65.1
         1       0.101327
         6         0.7755
         7       0.035057
         8      0.0523159
         9       0.017422
        20       0.018378
 3                 G4_ACETONE      0.7899       64.2
         6               3
         1               6
         8               1
 2               G4_ACETYLENE  0.0010967       58.2
         6               2
         1               2
 3                 G4_ADENINE        1.6       71.4
```

(continues on next page)



```
2                  G4_WATER          1         78    H_2O
         1               2
         8               1
```



### 11.6.1 Simple Materials (Elements)

| Z | Name | density(g/cm^3) | I(eV) |
|---|---|---|---|
| 1 | G4_H | 8.3748e-05 | 19.2 |
| 2 | G4_He | 0.000166322 | 41.8 |
| 3 | G4_Li | 0.534 | 40 |
| 4 | G4_Be | 1.848 | 63.7 |
| 5 | G4_B | 2.37 | 76 |
| 6 | G4_C | 2 | 81 |
| 7 | G4_N | 0.0011652 | 82 |
| 8 | G4_O | 0.00133151 | 95 |
| 9 | G4_F | 0.00158029 | 115 |
| 10 | G4_Ne | 0.000838505 | 137 |
| 11 | G4_Na | 0.971 | 149 |
| 12 | G4_Mg | 1.74 | 156 |
| 13 | G4_Al | 2.699 | 166 |
| 14 | G4_Si | 2.33 | 173 |
| 15 | G4_P | 2.2 | 173 |
| 16 | G4_S | 2 | 180 |
| 17 | G4_Cl | 0.00299473 | 174 |
| 18 | G4_Ar | 0.00166201 | 188 |
| 19 | G4_K | 0.862 | 190 |

Continued on next page



siguang@pku.edu.cn

# 可改变材料的密度

Listing 2.10: Defining air and water from the internal GEANT4 database.

```
G4NistManager* man = G4NistManager::Instance();

G4Material* H2O  = man->FindOrBuildMaterial("G4_WATER");
G4Material* Air  = man->FindOrBuildMaterial("G4_AIR");
```

## 2.3.6 Define a Material from the Base Material

It is possible to build new material on base of an existing "base" material. This feature is useful for electromagnetic physics allowing to peak up for the derived material all correction data and precomputed tables of stopping powers and cross sections of the base material. In the example below, two methods how to create water with unusual density are shown.

Listing 2.11: Defining water with user defined density on base of G4_WATER.

```
G4double density;

density = 1.05*mg/cm3;
G4Material* water1 = new G4Material("Water_1.05",density,"G4_WATER");

density = 1.03*mg/cm3;
G4NistManager* man = G4NistManager::Instance();
G4Material* water2 = man->BuildMaterialWithNewDensity("Water_1.03","G4_WATER",density);
```

# src-> B1DetectorConstruction.cc （续）

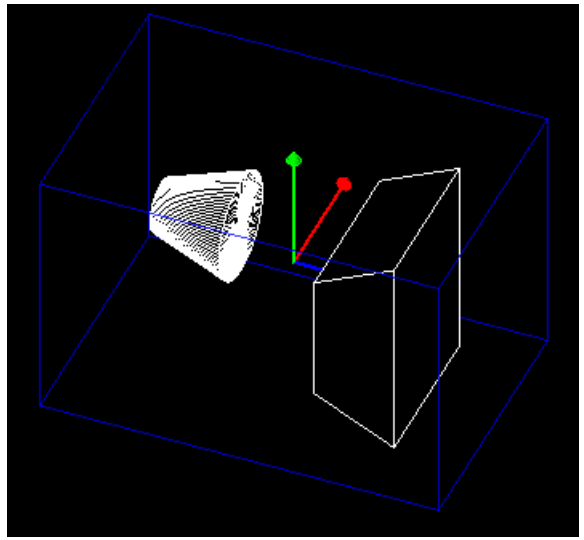
```
G4Box* solidWorld =
  new G4Box("World",                          //its name
     0.5*world_sizeXY, 0.5*world_sizeXY, 0.5*world_sizeZ);     //its size

G4LogicalVolume* logicWorld =
  new G4LogicalVolume(solidWorld,             //its solid
                      world_mat,              //its material
                      "World");               //its name

G4VPhysicalVolume* physWorld =
  new G4PVPlacement(0,                        //no rotation
                    G4ThreeVector(),          //at (0,0,0)
                    logicWorld,               //its logical volume
                    "World",                  //its name
                    0,                        //its mother  volume
                    false,                    //no boolean operation
                    0,                        //copy number
                    checkOverlaps);           //overlaps checking
```

右手坐标系：

X 红色箭头

Y 绿色箭头

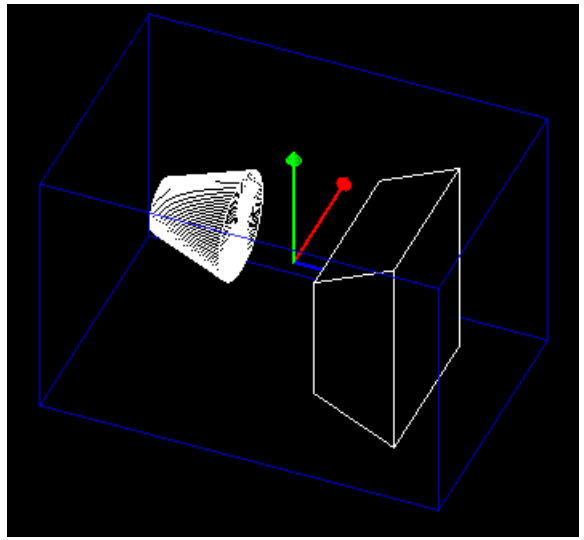
Z 蓝色箭头



Word看不见
材质G4_AIR

# src-> B1DetectorConstruction.cc （续）

```
//
// Envelope
//
G4Box* solidEnv =
  new G4Box("Envelope",                    //its name
      0.5*env_sizeXY, 0.5*env_sizeXY, 0.5*env_sizeZ); //its size

G4LogicalVolume* logicEnv =
  new G4LogicalVolume(solidEnv,            //its solid
                      env_mat,             //its material
                      "Envelope");         //its name

new G4PVPlacement(0,                       //no rotation
                  G4ThreeVector(),         //at (0,0,0)
                  logicEnv,                //its logical volume
                  "Envelope",              //its name
                  logicWorld,              //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  checkOverlaps);          //overlaps checking
```

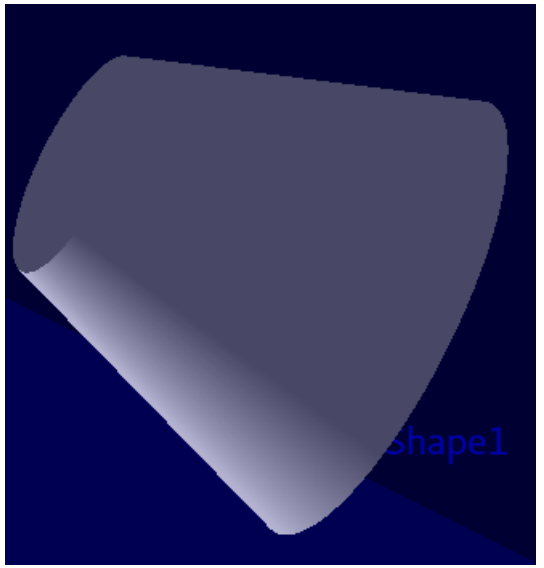


Envelope为蓝框
材质G4_WATER

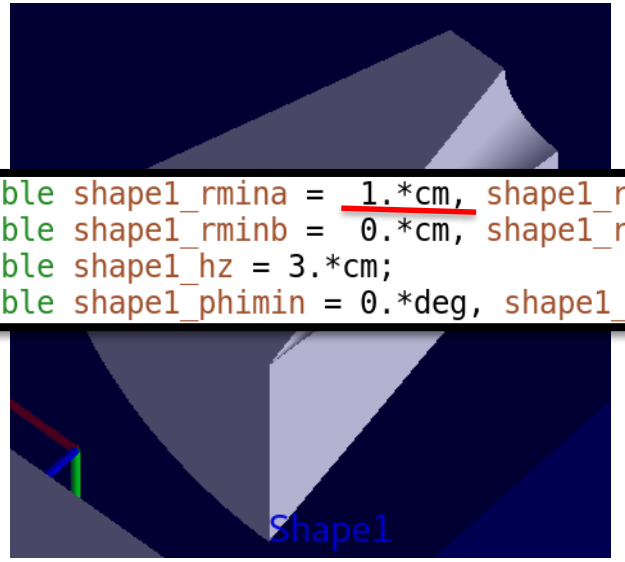
# src-> B1DetectorConstruction.cc （续）

```
//
// Shape 1
//
G4Material* shape1_mat = nist->FindOrBuildMaterial("G4_A-150_TISSUE");
G4ThreeVector pos1 = G4ThreeVector(0, 2*cm, -7*cm);

// Conical section shape
G4double shape1_rmina =  0.*cm, shape1_rmaxa = 2.*cm;
G4double shape1_rminb =  0.*cm, shape1_rmaxb = 4.*cm;
G4double shape1_hz = 3.*cm;
G4double shape1_phimin = 0.*deg, shape1_phimax = 360.*deg;
G4Cons* solidShape1 =
  new G4Cons("Shape1",
  shape1_rmina, shape1_rmaxa, shape1_rminb, shape1_rmaxb, shape1_hz,
  shape1_phimin, shape1_phimax);

G4LogicalVolume* logicShape1 =
  new G4LogicalVolume(solidShape1,         //its solid
                      shape1_mat,          //its material
                      "Shape1");           //its name

new G4PVPlacement(0,                       //no rotation
                  pos1,                    //at position
                  logicShape1,             //its logical volume
                  "Shape1",                //its name
                  logicEnv,                //its mother  volume
                  false,                   //no boolean operation
                  0,                       //copy number
                  checkOverlaps);          //overlaps checking
```

一半的高度



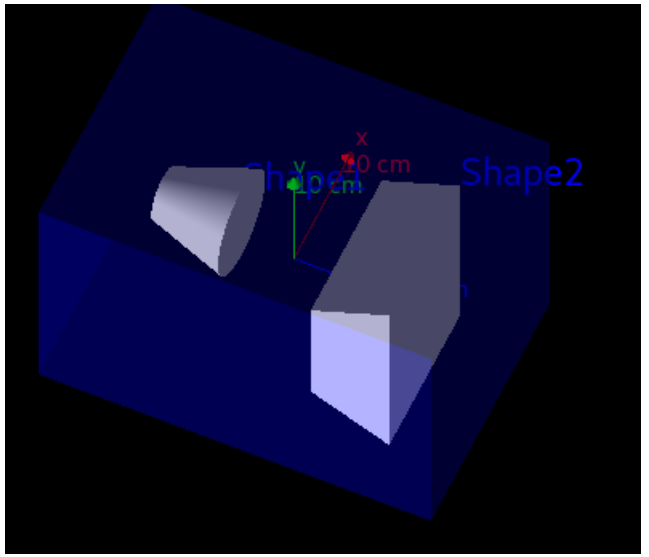

```
G4double shape1_rmina =  1.*cm, shape1_rmaxa = 2.*cm;
G4double shape1_rminb =  0.*cm, shape1_rmaxb = 4.*cm;
G4double shape1_hz = 3.*cm;
G4double shape1_phimin = 0.*deg, shape1_phimax = 90.*deg;
```




siguang@pku.edu.cn

# src-> B1DetectorConstruction.cc （续）

```
// Shape 2
//
G4Material* shape2_mat = nist->FindOrBuildMaterial("G4_BONE_COMPACT_ICRU");
G4ThreeVector pos2 = G4ThreeVector(0, -1*cm, 7*cm);

// Trapezoid shape
G4double shape2_dxa = 12*cm, shape2_dxb = 12*cm;
G4double shape2_dya = 10*cm, shape2_dyb = 16*cm;
G4double shape2_dz  = 6*cm;
G4Trd* solidShape2 =
  new G4Trd("Shape2",                          //its name
            0.5*shape2_dxa, 0.5*shape2_dxb,
            0.5*shape2_dya, 0.5*shape2_dyb, 0.5*shape2_dz); //its size

G4LogicalVolume* logicShape2 =
  new G4LogicalVolume(solidShape2,             //its solid
                      shape2_mat,              //its material
                      "Shape2");               //its name

new G4PVPlacement(0,                           //no rotation
                  pos2,                        //at position
                  logicShape2,                 //its logical volume
                  "Shape2",                    //its name
                  logicEnv,                    //its mother  volume
                  false,                       //no boolean operation
                  0,                           //copy number
                  checkOverlaps);              //overlaps checking
```




右手坐标系：

X 红色箭头

Y 绿色箭头

Z 蓝色箭头

# src-> B1DetectorConstruction.cc （续）

```
  // Set Shape2 as scoring volume
  //
  fScoringVolume = logicShape2;

  //
  //always return the physical World
  //
  return physWorld;
}
```

记录区域

```
                                          //its name
      ld_sizeXY, 0.5*world_sizeXY, 0.5*world_sizeZ);

  calVolume* logicWorld =
ew G4LogicalVolume(solidWorld,            //its solid
                   world_mat,             //its material
                   "World");              //its name

G4VPhysicalVolume* physWorld =
  new G4PVPlacement(0,                    //no rotation
                    G4ThreeVector(),      //at (0,0,0)
                    logicWorld,           //its logical volume
                    "World",              //its name
                    0,                    //its mother  volume
                    false,                //no boolean operati
                    0,                    //copy number
                    checkOverlaps);       //overlaps
```

前面定义的具体信息

```
class B1DetectorConstruction : public G4VUserDetectorConstruction
{
  public:
    B1DetectorConstruction();
    virtual ~B1DetectorConstruction();

    virtual G4VPhysicalVolume* Construct();

    G4LogicalVolume* GetScoringVolume() const { return fScoringVolume; }

  protected:
    G4LogicalVolume*  fScoringVolume;
};
```

# include -> B1ActionInitialization.hh

```
class B1ActionInitialization : public G4VUserActionInitialization
{
  public:
    B1ActionInitialization();
    virtual ~B1ActionInitialization();

    virtual void BuildForMaster() const;
    virtual void Build() const;
};
```

# src -> B1ActionInitialization.cc

```
#include "B1ActionInitialization.hh"
#include "B1PrimaryGeneratorAction.hh"
#include "B1RunAction.hh"
#include "B1EventAction.hh"
#include "B1SteppingAction.hh"

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo..

B1ActionInitialization::B1ActionInitialization()
 : G4VUserActionInitialization()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo..

B1ActionInitialization::~B1ActionInitialization()
{}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo..

void B1ActionInitialization::BuildForMaster() const
{
  B1RunAction* runAction = new B1RunAction;
  SetUserAction(runAction);
}
```

最后汇总被调用
如果单线程不需要
这个。

Run->Event->Step

siguang@pku.edu.cn

# src -> B1ActionInitialization.cc

```
void B1ActionInitialization::Build() const
{
  SetUserAction(new B1PrimaryGeneratorAction);

  B1RunAction* runAction = new B1RunAction;
  SetUserAction(runAction);

  B1EventAction* eventAction = new B1EventAction(runAction);
  SetUserAction(eventAction);

  SetUserAction(new B1SteppingAction(eventAction));
}
```

各个运行线程调用

多线程与单线程的区别？利弊？

# include-> B1RunAction.hh

```
class B1RunAction : public G4UserRunAction
{
  public:
    B1RunAction();
    virtual ~B1RunAction();

    // virtual G4Run* GenerateRun();
    virtual void BeginOfRunAction(const G4Run*);
    virtual void   EndOfRunAction(const G4Run*);

    void AddEdep (G4double edep);

  private:
    G4Accumulable<G4double> fEdep;
    G4Accumulable<G4double> fEdep2;
};
```

自己添加的

vector类似的

# src -> B1ActionInitialization.cc -> B1RunAction.cc

```
#include "B1RunAction.hh"
#include "B1PrimaryGeneratorAction.hh"
#include "B1DetectorConstruction.hh"

B1RunAction::B1RunAction()
: G4UserRunAction(),
  fEdep(0.),
  fEdep2(0.)
{
  // add new units for dose
  //
  const G4double milligray = 1.e-3*gray;
  const G4double microgray = 1.e-6*gray;
  const G4double nanogray  = 1.e-9*gray;
  const G4double picogray  = 1.e-12*gray;

  new G4UnitDefinition("milligray", "milliGy" , "Dose", milligray);
  new G4UnitDefinition("microgray", "microGy" , "Dose", microgray);
  new G4UnitDefinition("nanogray" , "nanoGy"  , "Dose", nanogray);
  new G4UnitDefinition("picogray" , "picoGy"  , "Dose", picogray);

  // Register accumulable to the accumulable manager
  G4AccumulableManager* accumulableManager = G4AccumulableManager::Instance();
  accumulableManager->RegisterAccumulable(fEdep);
  accumulableManager->RegisterAccumulable(fEdep2);
}
```

# src -> B1ActionInitialization.cc -> B1RunAction.cc

```
void B1RunAction::BeginOfRunAction(const G4Run*)
{
  // inform the runManager to save random number seed
  G4RunManager::GetRunManager()->SetRandomNumberStore(false);

  // reset accumulables to their initial values
  G4AccumulableManager* accumulableManager = G4AccumulableManager::Instance();
  accumulableManager->Reset();

}
```

# src -> B1ActionInitialization.cc -> B1RunAction.cc

```
void B1RunAction::EndOfRunAction(const G4Run* run)
{
  G4int nofEvents = run->GetNumberOfEvent();
  if (nofEvents == 0) return;

  // Merge accumulables
  G4AccumulableManager* accumulableManager = G4AccumulableManager::Instance();
  accumulableManager->Merge();

  // Compute dose = total energy deposit in a run and its variance
  //
  G4double edep  = fEdep.GetValue();
  G4double edep2 = fEdep2.GetValue();

  G4double rms = edep2 - edep*edep/nofEvents;
  if (rms > 0.) rms = std::sqrt(rms); else rms = 0.;

  const B1DetectorConstruction* detectorConstruction
   = static_cast<const B1DetectorConstruction*>
     (G4RunManager::GetRunManager()->GetUserDetectorConstruction());
  G4double mass = detectorConstruction->GetScoringVolume()->GetMass();
  G4double dose = edep/mass;
  G4double rmsDose = rms/mass;
```

# src -> B1ActionInitialization.cc -> B1RunAction.cc

```
// Run conditions
//  note: There is no primary generator action object for "master"
//        run manager for multi-threaded mode.
const B1PrimaryGeneratorAction* generatorAction
 = static_cast<const B1PrimaryGeneratorAction*>
   (G4RunManager::GetRunManager()->GetUserPrimaryGeneratorAction());
G4String runCondition;
if (generatorAction)
{
  const G4ParticleGun* particleGun = generatorAction->GetParticleGun();
  runCondition += particleGun->GetParticleDefinition()->GetParticleName();
  runCondition += " of ";
  G4double particleEnergy = particleGun->GetParticleEnergy();
  runCondition += G4BestUnit(particleEnergy,"Energy");
}
```

获取初始粒子名字及能量的方法

# src -> B1ActionInitialization.cc -> B1RunAction.cc

```
  // Print
  //
  if (IsMaster()) {
    G4cout
     << G4endl
     << "--------------------End of Global Run-----------------------";
  }
  else {
    G4cout
     << G4endl
     << "--------------------End of Local Run-----------------------";
  }

  G4cout
     << G4endl
     << " The run consists of " << nofEvents << " "<< runCondition
     << G4endl
     << " Cumulated dose per run, in scoring volume : "
     << G4BestUnit(dose,"Dose") << " rms = " << G4BestUnit(rmsDose,"Dose")
     << G4endl
     << "------------------------------------------------------------"
     << G4endl
     << G4endl;
}
```

# src -> B1ActionInitialization.cc -> B1RunAction.cc

```
void B1RunAction::AddEdep(G4double edep)
{
  fEdep  += edep;
  fEdep2 += edep*edep;
}
```

AddEdep调用位置：

g4root@debian:~/work/g4/10.4/examples/basic/B1$ grep -i -r AddEdep
Binary file bin/exampleB1 matches
Binary file bin/CMakeFiles/exampleB1.dir/src/B1EventAction.cc.o matches
Binary file bin/CMakeFiles/exampleB1.dir/src/B1RunAction.cc.o matches
include/B1EventAction.hh:    void AddEdep(G4double edep) { fEdep += edep; }
include/B1RunAction.hh:    void AddEdep (G4double edep);
**src/B1RunAction.cc:void B1RunAction::AddEdep(G4double edep)**
**src/B1EventAction.cc:  fRunAction->AddEdep(fEdep);**
**src/B1SteppingAction.cc:  fEventAction->AddEdep(edepStep);**
g4root@debian:~/work/g4/10.4/examples/basic/B1$ cd include/

# include->B1EventAction.hh

头文件在B1EventAction.cc中包含的：
#include "B1RunAction.hh"

```
#ifndef B1EventAction_h
#define B1EventAction_h 1

#include "G4UserEventAction.hh"
#include "globals.hh"
class B1RunAction;

/// Event action class
///

class B1EventAction : public G4UserEventAction
{
  public:
    B1EventAction(B1RunAction* runAction);
    virtual ~B1EventAction();

    virtual void BeginOfEventAction(const G4Event* event);
    virtual void EndOfEventAction(const G4Event* event);

    void AddEdep(G4double edep) { fEdep += edep; }

  private:
    B1RunAction* fRunAction;
    G4double     fEdep;
};
```

# src -> B1ActionInitialization.cc -> B1RunAction.cc ->B1EventAction.cc

```
#include "B1EventAction.hh"
#include "B1RunAction.hh"

B1EventAction::B1EventAction(B1RunAction* runAction)
: G4UserEventAction(),
  fRunAction(runAction),
  fEdep(0.)
{}
```

赋初始值

```
void B1EventAction::BeginOfEventAction(const G4Event*)
{
  fEdep = 0.;
}

//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo

void B1EventAction::EndOfEventAction(const G4Event*)
{
  // accumulate statistics in run action
  fRunAction->AddEdep(fEdep);
}
```

回调：下级将信息报告上级

# include ->B1SteppingAction.hh

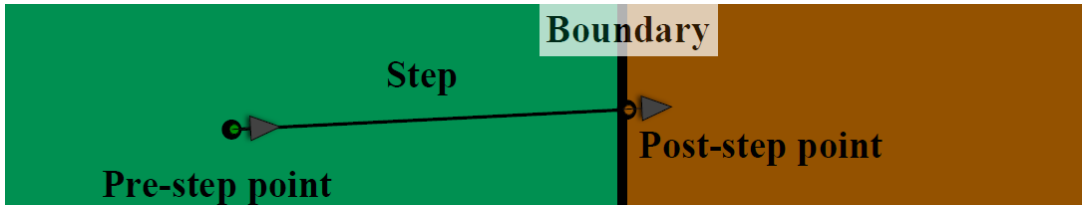
```
#ifndef B1SteppingAction_h
#define B1SteppingAction_h 1

#include "G4UserSteppingAction.hh"
#include "globals.hh"

class B1EventAction;

class G4LogicalVolume;

/// Stepping action class
///

class B1SteppingAction : public G4UserSteppingAction
{
  public:
    B1SteppingAction(B1EventAction* eventAction);
    virtual ~B1SteppingAction();

    // method from the base class
    virtual void UserSteppingAction(const G4Step*);

  private:
    B1EventAction*  fEventAction;
    G4LogicalVolume* fScoringVolume;
};
```

头文件在.cc 中包含

# src->B1SteppingAction.cc

```
#include "B1SteppingAction.hh"
#include "B1EventAction.hh"
#include "B1DetectorConstruction.hh"

void B1SteppingAction::UserSteppingAction(const G4Step* step)
{
  if (!fScoringVolume) {
    const B1DetectorConstruction* detectorConstruction
      = static_cast<const B1DetectorConstruction*>
        (G4RunManager::GetRunManager()->GetUserDetectorConstruction());
    fScoringVolume = detectorConstruction->GetScoringVolume();
  }

  // get volume of the current step
  G4LogicalVolume* volume
    = step->GetPreStepPoint()->GetTouchableHandle()
      ->GetVolume()->GetLogicalVolume();

  // check if we are in scoring volume
  if (volume != fScoringVolume) return;

  // collect energy deposited in this step
  G4double edepStep = step->GetTotalEnergyDeposit();
  fEventAction->AddEdep(edepStep);
}
```

每走一个步长结束被调动一次




边界限制的粒子：
1、物理上在边界上，即：G4Step::GetPostStepPoint()->GetStepStatus() 是fGeomBoundary
2、逻辑上属于下一个区间

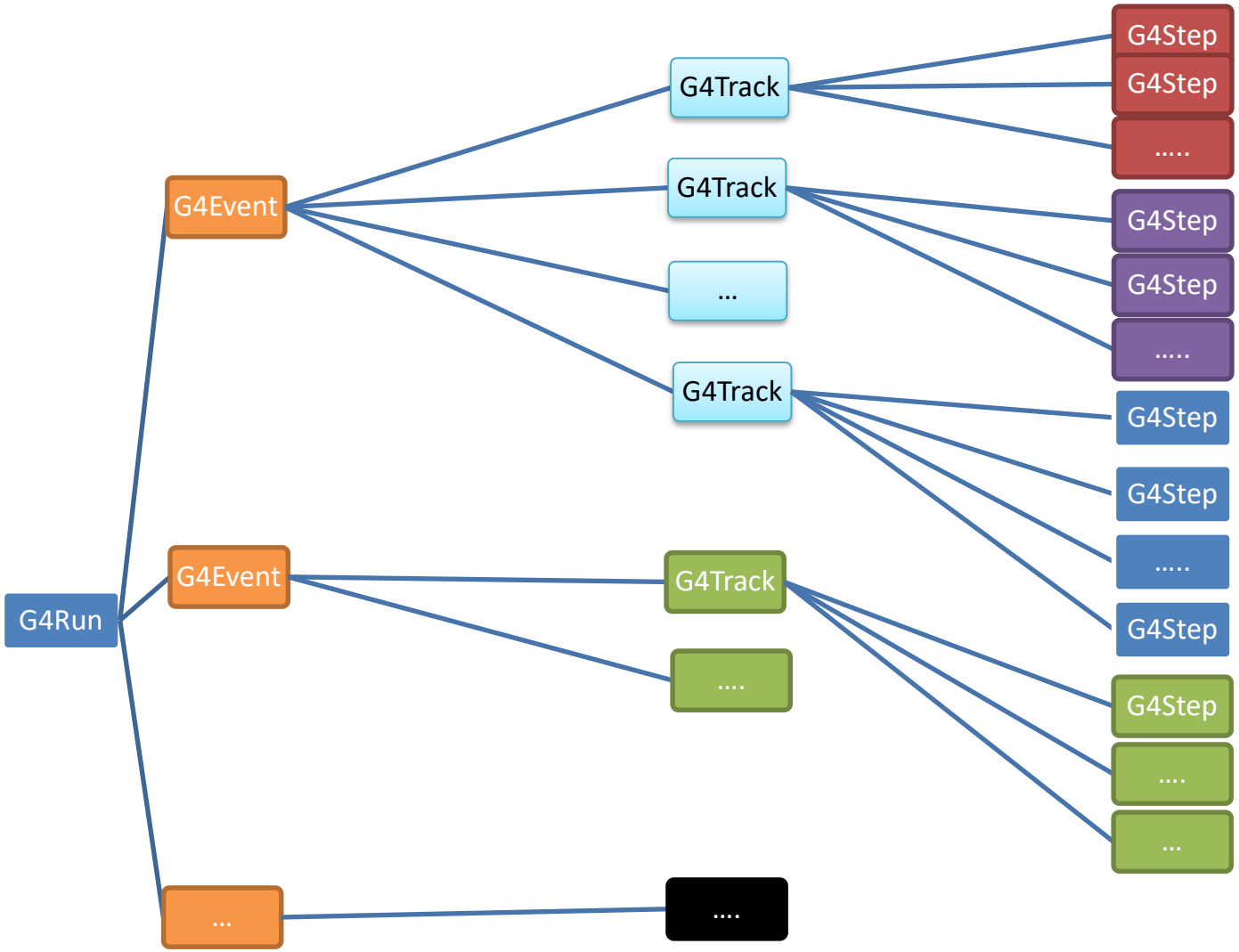
# fScoringVolume出现的位置汇总

```
g4root@debian:~/work/g4/10.4/examples/basic/B1$ grep -r -i fScoringVolume
include/B1DetectorConstruction.hh:    G4LogicalVolume* GetScoringVolume() const { return fScoringVolume; }
include/B1DetectorConstruction.hh:    G4LogicalVolume*  fScoringVolume;
include/B1SteppingAction.hh:    G4LogicalVolume* fScoringVolume;

src/B1DetectorConstruction.cc:  fScoringVolume(0)
src/B1DetectorConstruction.cc:  fScoringVolume = logicShape2;

src/B1SteppingAction.cc:  fScoringVolume(0)
src/B1SteppingAction.cc:  if (!fScoringVolume) {
src/B1SteppingAction.cc:    fScoringVolume = detectorConstruction->GetScoringVolume();
src/B1SteppingAction.cc:  if (volume != fScoringVolume) return;
```

# 从G4Step可获取的重要信息(可作为常用字典）

- G4StepPoint *preStp = step->GetPreStepPoint();
- G4VPhysicalVolume  *physVol = step->GetPhysicalVolume();
- G4double stpEdep = step->GetTotalEnergyDeposit();
- G4double stpLength = step->GetStepLength();
- G4Track *theTrack = step->GetTrack();
- const G4ParticleDefinition *partDef = theTrack->GetParticleDefinition();
- const G4DynamicParticle *partDyn = theTrack->GetDynamicParticle();
- G4double partCharge = partDef->GetPDGCharge();
- G4double postStpEkin = step->GetPostStepPoint()->GetKineticEnergy();
- 或者 G4double postStpEkin = partDyn->GetKineticEnergy();
- G4double preStpEkin = step->GetKineticEnergy();

# Run、Event、Track、Step关系




siguang@pku.edu.cn

# G4Event （详细了解）

- **G4Event是由一些G4Track的模拟构成**
- 事件之开始阶段(**at the beginning** of an event):
  - **初始G4Track产生** (位置、四动量) 并压入径迹堆栈（**track-stack**）
  - **从径迹堆栈中弹出一个G4Track并输运、追踪、模拟**（ transported、tracked、simulated ）:
    - 径迹一步步传播的同时位置、动量等在一步步作为**G4Step进行更新**
  - 一步步由物理相互作用（physics interaction）或几何边界（geometry boundary）区分
  - **边界上的点将在下一步进入下一个区域，所记录的物理信息为上一步物理作用结果。**
  - **物理作用产生的次级G4Track-s**也压入径迹堆栈
  - **G4Track被跟踪到：**
    - 离开最外层的世界 (World)
    - 彻底挂了(例如光电吸收，彻底消失了)
    - 动能足够小没有可进一步发生的物理作用了，认为全部能量沉积下来
    - 用户强制终止该径迹的跟踪并杀掉（Kill）
  - 模拟完一个**G4Track，新的G4Track** (次级或原初径迹) 从堆栈中弹出
  - 堆栈中没有任何**G4Track的时候一个**event结束
- **事件结束后（at the end** of an event）,**G4Event将调用结束程序存储期间的需要保存的信息，例如原初粒子的信息，径迹信息，沉积能量信息等输入及模拟过程中的输出信息。**

# G4Event - G4UserEventAction

- optional user action class, with the possibility to get the control before/after an event processing
- `virtual BeginOfEventAction(const G4Event* anEvent);`
  - **called before** a new **event processing** starts by the **`G4EventManager`**
- `virtual EndOfEventAction(const G4Event* anEvent);`
  - **called after** an **event processing competed** by the **`G4EventManager`**
- in both cases, the **`G4EventManager`** provides access to the corresponding **`G4Event`** object (see this class in Geant4 `/source/event/include/G4UserEventAction.hh`)
- users can implement their own **`YourEventAction`** class by:
  - extending the **`G4UserEventAction`** class
  - providing their own implementation of the two methods mentioned above
  - creating and registering the corresponding object in the **`ActionInitialisation::Build()`** interface method
- at the beginning of the event (`BeginOfEventAction`) we usually clear some data structures that we want to use to accumulate information during the processing of the current event (populated after each step in the **`UserSteppingAction`**) while at the end of the event (`EndOfEventAction`) we usually write the accumulated information to an upper (Run global) level

# G4Run

- G4Run is a collection of G4Event-s (a G4Event is a collection of G4Track-s)
- during a run, events are taken and processed one by one in an event-loop
- before the start of a run i.e. at run initialization (G4RunManager::Initialize()): the geometry is constructed and physics is initialized
- at the start of a run (G4RunManager::BeamOn()): the geometry is optimized for tracking the event processing starts i.e. entering into the event-loop
- as long as the event processing is running, i.e. during the run, the user cannot modify neither the geometry (i.e. the detector setup) nor the physics settings
- they can be changed though between run-s but the G4RunManager needs to be informed (re-optimize or re-construct geometry, re-build physics tables):
- if the geometry has been changed, depending on the modifications:
  - GeometryHasBeenModified() re-voxelization（像素化; 体元化） but no re-Construct
  - ReinitializeGeometry() complete re-Construct

    or with the UI commands /run/geometryModified or /run/reinitializeGeometry
- same for the physics: PhysicsHasBeenModifie() or /run/physicsModifie

# GeometryHasBeenModified的例子

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo.
void DetectorConstruction::SetSurfacePolish(G4double v) {
  fSurface->SetPolish(v);
  G4RunManager::GetRunManager()->GeometryHasBeenModified();

  G4cout << "Surface polish set to: " << fSurface->GetPolish()
         << G4endl;
}
```

表面参数的变化

```
void DetectorConstruction::SetEcalWidth  (G4double val)
{
  if(val > 0.0) {
    fEcalWidth = val;
    G4RunManager::GetRunManager()->GeometryHasBeenModified();
  }
}
```

尺寸变化

# GeometryHasBeenModified的例子

```
void GB03DetectorConstruction::SetAbsorberMaterial(G4String materialChoice)
{
  // search the material by its name
  G4Material* pttoMaterial = G4Material::GetMaterial(materialChoice);
  if(pttoMaterial)
  {
    fAbsorberMaterial = pttoMaterial;
    if(fConstructed)
    {
      fCalorLogical->SetMaterial(fAbsorberMaterial);
      fLayerLogical->SetMaterial(fAbsorberMaterial);
    }
    G4RunManager::GetRunManager()->GeometryHasBeenModified();
    if (GetVerboseLevel()>1) {
      PrintCalorParameters();
    }
  }
  else
  {
    G4cerr
      << materialChoice << " is not defined. - Command is ignored." << G4endl;
  }
}
```

材料变化

# ReinitializeGeometry例子

- extended/hadronic/Hadr04/src/DetectorConstruction.cc

```
//....oooOOOOOooo........oooOOOOOooo........oooOOOOOooo........oooOOOOOooo.

void DetectorConstruction::SetSize(G4double value)
{
  fBoxSize = value;
  G4RunManager::GetRunManager()->ReinitializeGeometry();
}
```

# PhysicsHasBeenModified

- extended/hadronic/Hadr04/src/DetectorConstruction.cc

```
void DetectorConstruction::SetMaterial(G4String materialChoice)
{
  // search the material by its name
  G4Material* pttoMaterial =
     G4NistManager::Instance()->FindOrBuildMaterial(materialChoice);

  if (pttoMaterial) {
    if(fMaterial != pttoMaterial) {
      fMaterial = pttoMaterial;
      if(fLBox) { fLBox->SetMaterial(pttoMaterial); }
      G4RunManager::GetRunManager()->PhysicsHasBeenModified();
    }
  } else {
    G4cout << "\n--> warning from DetectorConstruction::SetMaterial : "
           << materialChoice << " not found" << G4endl;
  }
}
```

加入了新材料

# G4Run - G4UserRunAction

- optional user action class, with the possibility to get the control before/after a run and to provide custom run-object
- `virtual BeginOfRunAction(const G4Run* aRun);`
  - **called before** the **run starts** i.e. **before the first event processing** starts by the **`G4RunManager`**
- `virtual EndOfRunAction(const G4Run* aRun);`
  - **called after** a **run completed** i.e. **after the last event processing competed** by the **`G4RunManager`**
- in both cases, the **`G4RunManager`** provides access to the corresponding **`G4Run`** object (see this class in Geant4 `/source/run/include/G4UserRunAction.hh`)
- users can implement their own **`YourRunAction`** class by:
  - extending the **`G4UserRunAction`** class
  - providing their own implementation of the two methods mentioned above
  - creating and registering the corresponding object in the **`ActionInitialisation::Build()`** interface method
- note, at the beginning of the run (`BeginOfRunAction`) we usually allocate/initialise some data structures/histograms that we want to use during the whole run to collect the final simulation results that we usually print out at the end of the run (`EndOfRunAction`)

# include->B1PrimaryGeneratorAction.hh

```
class B1PrimaryGeneratorAction : public G4VUserPrimaryGeneratorAction
{
  public:
    B1PrimaryGeneratorAction();
    virtual ~B1PrimaryGeneratorAction();

    // method from the base class
    virtual void GeneratePrimaries(G4Event*);

    // method to access particle gun
    const G4ParticleGun* GetParticleGun() const { return fParticleGun; }

  private:
    G4ParticleGun*  fParticleGun; // pointer a to G4 gun class
    G4Box* fEnvelopeBox;
};
```

# src -> B1ActionInitialization.cc->B1PrimaryGeneratorAction.cc

```
void B1ActionInitialization::Build() const
{
  SetUserAction(new B1PrimaryGeneratorAction);

  B1RunAction* runAction = new B1RunAction;
  SetUserAction(runAction);

  B1EventAction* eventAction = new B1EventAction(runAction);
  SetUserAction(eventAction);

  SetUserAction(new B1SteppingAction(eventAction));
}
```

```
B1PrimaryGeneratorAction::B1PrimaryGeneratorAction()
: G4VUserPrimaryGeneratorAction(),
  fParticleGun(0),
  fEnvelopeBox(0)
{
  G4int n_particle = 1;
  fParticleGun  = new G4ParticleGun(n_particle);

  // default particle kinematic
  G4ParticleTable* particleTable = G4ParticleTable::GetParticleTable();
  G4String particleName;
  G4ParticleDefinition* particle
    = particleTable->FindParticle(particleName="gamma");
  fParticleGun->SetParticleDefinition(particle);
  fParticleGun->SetParticleMomentumDirection(G4ThreeVector(0.,0.,1.));
  fParticleGun->SetParticleEnergy(6.*MeV);
}
```

# src -> B1ActionInitialization.cc->B1PrimaryGeneratorAction.cc

```
void B1PrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
  //this function is called at the begining of ecah event
  //

  // In order to avoid dependence of PrimaryGeneratorAction
  // on DetectorConstruction class we get Envelope volume
  // from G4LogicalVolumeStore.

  G4double envSizeXY = 0;
  G4double envSizeZ = 0;

  if (!fEnvelopeBox)
  {
    G4LogicalVolume* envLV
      = G4LogicalVolumeStore::GetInstance()->GetVolume("Envelope");
    if ( envLV ) fEnvelopeBox = dynamic_cast<G4Box*>(envLV->GetSolid());
  }

  if ( fEnvelopeBox ) {
    envSizeXY = fEnvelopeBox->GetXHalfLength()*2.;
    envSizeZ = fEnvelopeBox->GetZHalfLength()*2.;
  }
  else  {
```

下一个if可以放这里保护，确保只调用一次（尺寸变量改为类成员变量。）

# src -> B1ActionInitialization.cc->B1PrimaryGeneratorAction.cc

```
  else  {
    G4ExceptionDescription msg;
    msg << "Envelope volume of box shape not found.\n";
    msg << "Perhaps you have changed geometry.\n";
    msg << "The gun will be place at the center.";
    G4Exception("B1PrimaryGeneratorAction::GeneratePrimaries()",
     "MyCode0002",JustWarning,msg);
  }

  G4double size = 0.8;
  G4double x0 = size * envSizeXY * (G4UniformRand()-0.5);
  G4double y0 = size * envSizeXY * (G4UniformRand()-0.5);
  G4double z0 = -0.5 * envSizeZ;

  fParticleGun->SetParticlePosition(G4ThreeVector(x0,y0,z0));

  fParticleGun->GeneratePrimaryVertex(anEvent);
}
```

# 结构梳理：主程序中的 G4RunManager

- **Geant4** do not provide a main program:
  - there are components of a simulation, like the **geometry**, **physics** settings and the **primary** particle **generation** that are changing from problem to problem，therefore, the **user needs to provide these** settings **in the** **main** **method** of their **Geant4** application

1. Create a **G4RunManager** object **(mandatory)**:

- the **only** mandatory **manager object** that user needs to create: all others (**G4EventManager**, **G4SteppingManger**, etc.) are created and deleted automatically
  - the **G4RunManager** is responsible to **control the** flow of a **run**, the top level simulation unit
  - this includes **initialization of the run** i.e. building, **setting up the** simulation **environment**
  - all problem specific information need to be given to the **G4RunManager** by the user through the interfaces provided by the **Geant4** toolkit:
    - **G4VUserDetectorConstruction**(mandatory): how the geometry should be constructed, built
    - **G4VUserPhyscsList**(mandatory): all the particles and their physics interactions to be simulated
    - **G4VUserActionInitialization** (mandatory):
    - **G4VUserPrimaryGeneratorAction** (mandatory): how the primary particle(s) in an event should be produced
    - additional, **optional user actions** (**G4UserRunAction**, **G4UserEventAction**, **G4UserSteppingAction**, etc..)
- **- MT note:** **G4MTRunManager** object needs to be created in case of **Geant4 MT**

# 结构梳理：主程序中的 G4VUserDetectorConstruction

## 2. Create `YourDetectorConstruction` object and register it in your `G4RunManager` object **(mandatory)**:

- the `G4VUserDetectorConstruction` interface is provided by the **`Geant4`** toolkit to **describe** the **geometrical setup**, including all volumes with their shape, position and **material definition**
  - its `G4VUserDetectorConstruction::Construct()` interface method (pure virtual) is invoked by the **`G4RunManager`** at initialization
  - **derive your own detector description**, e.g. `YourDetectorConstruction` class from this base class and implement the `Construct()` interface method:
    - create all materials will need to use in your geometry
    - describe your detector geometry by creating and positioning all volumes
    - return the pointer to the root of your geometry hierarchy i.e. the pointer to your "World" **`G4VPhysicalVolume`**
- create `YourDetectorConstruction` object and register it in your `G4RunManager` object by using the `G4RunManager:SetUserinitialization` method (see this in the source!)
- **MT note:** the `Construct()` interface method is **invoked only by the Master Thread** in caseof **`Geant4 MT`** (i.e. only one detector object), while the other `ConstructSDandField()` interface method is **invoked by each Worker Threads** (i.e. thread local objects created)

# 结构梳理：主程序中的 G4VUserPhysicsList

3. Create **YourPhysicsList** object and register it in your **G4RunManager** object **(mandatory)**:

- the **G4VUserPhysicsList** interface is provided by the **Geant4** toolkit to **describe** the **physics setup**, including **definition of** all **particles** and their physics interactions, **processes**
- its **G4VUserPhysicsList::ConstructParticle()** and **::ConstructProcess()** interface methods (pure virtual) are invoked by the **G4RunManager** (actually by the **G4RunManagerKernel** and process construction is invoked indirectly) at initialization
- **derive your own physics list**, e.g. **YourPhysicsList** class from this base class and implement the **ConstructParticle()** and **ConstructProcess()** interface methods:
  - create all particles in the **ConstructParticle()** method
  - create all processes and sign them to particles in the **ConstructProcess()** method
  - create **YourPhysicsList** object and register it in your **G4RunManager** object by using the `G4RunManager:SetUserinitialization` method

- **Geant4** provides possibilities with different level of granularity to build up or obtain even complete pre-defined physics list

# 结构梳理：主程序中的 G4VUserPrimaryGeneratorAction

4. Create `YourPrimaryGeneratorAction` object **(mandatory**, see next slide how to register**)**:

- the `G4VUserPrimaryGeneratorAction` interface is provided by the **`Geant4`** toolkit to describe how the primary particle(s) in an event should be produced
  - its `G4VUserPrimaryGeneratorAction::GeneratePrimaries()` interface method (pure virtual) is invoked by **`G4RunManager`** during event-loop(`G4RunManager::GenerateEvent()` method)
- **derive your own primary generator action,** e.g. `YourPrimaryGeneratorAction` class from this base class and implement the `GeneratePrimaries()` interface method:
  - describe how the primary particle(s) in an event should be produced
  - use a **`G4ParticleGun`** object, provided by the **`Geant4`** toolkit, to generate primary particles with defined kinematics

- note:
  - the **Detector-Construction** and the **Physics-List** need to be **created directly in the main** program and **registered directly in the `G4RunManager`** object
  - all **User-Actions** needs to be **created** and **registered** in the **User-Action-Initialization** (**including the** only **mandatory Primary-Generator-Action** as well as **all** other, **optional User-Actions**)

# 结构梳理：主程序中的 G4VUserActionInitialization

5. Create `YourActionInitialization` object and register it in your `G4RunManager` object **(mandatory)**:

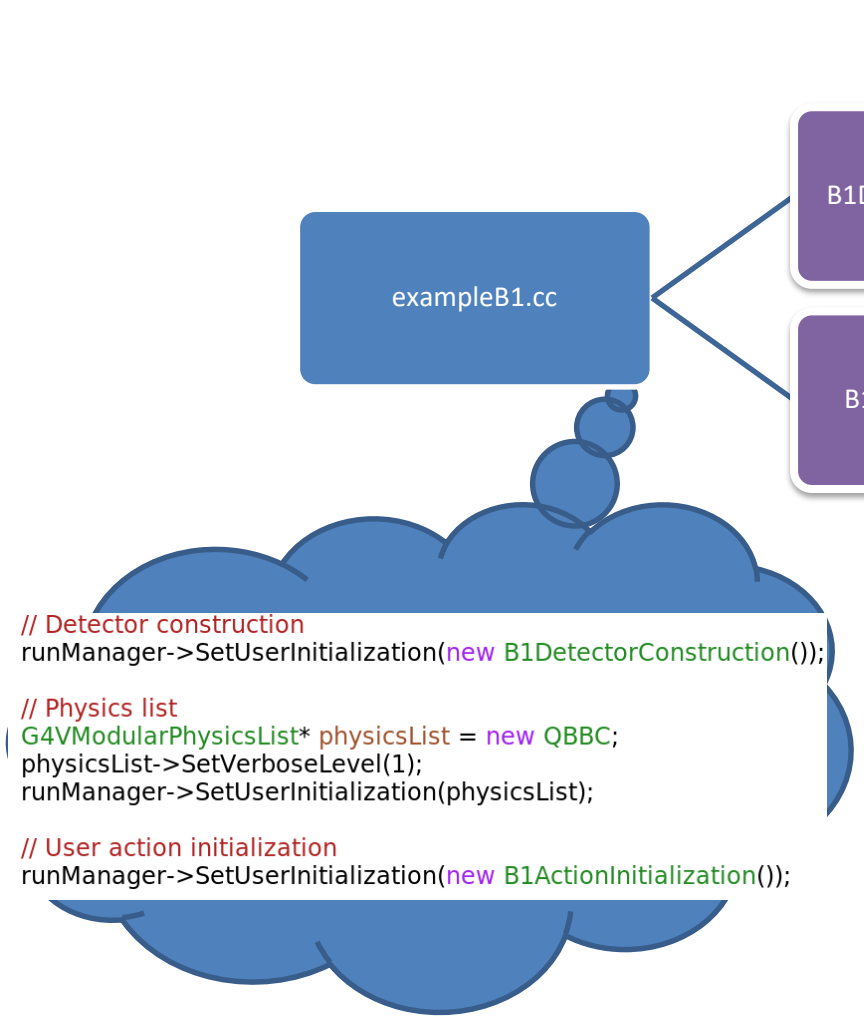
- the `G4VUserActionInitialization` interface is provided by the **`Geant4`** toolkit to **create** and **register:**
  - **the only one mandatory `G4VUserPrimaryGeneratorAction` user action**
  - all other **optional user actions** (**`G4UserRunAction`**, **`G4UserEventAction`**, etc..)
- its `G4VUserActionInitialization::Build()` interface method (pure virtual) is invoked by the **`G4RunManager`** at initialization
- **derive your own action initialization**, e.g. `YourActionInitialization` class from this base class and implement the `Build()` interface methods:
  - **create** an object from `YourPrimaryGeneratorAction` and **register** by calling the corresponding `G4VUserActionInitialization::SetUserAction()` base class method
  - **create all** additional, **optional user action** objects and **register them** similarly

- **MT note:**

- the above `Build()` method is invoked **by all Worker Thread** while the additional `BuildForMaster()` method is invoked **only by the Master Thread** in **`Geant4 MT`**
- the **only user action,** that is supposed **to be created in** this `BuildForMaster()` method, is the **`G4UserRunAction` for the Master Thread**
- this **`G4UserRunAction`** is the same or different compared to that used in case of the **Worker Threads**
- but contains a **`G4Run`** generation that will generate the **run-global, thread-global** i.e. **Master `G4Run` object** to **which all** other **Worker `G4Run` objects will be Merged at the end** by calling the **`G4Run::Merge`** method
- the `BuildForMaster()` method is invoked only in case of **`Geant4 MT`**

```
void ActionInitialization::BuildForMaster() const
{
  auto runAction = new RunAction;
  SetUserAction(runAction);
}
```

# Basic-B1的结构图



# 宏文件

- /examples/basic/B1/bin$ ll *.mac
- -rw-r--r-- 1 g4root g4root  472 Feb 27 14:11 run2.mac
- -rw-r--r-- 1 g4root g4root  338 Feb 27 14:04 init_vis.mac
- -rw-r--r-- 1 g4root g4root  553 Feb 27 14:04 run1.mac
- -rw-r--r-- 1 g4root g4root 3931 Feb 27 14:04 vis.mac

# init_vis.mac

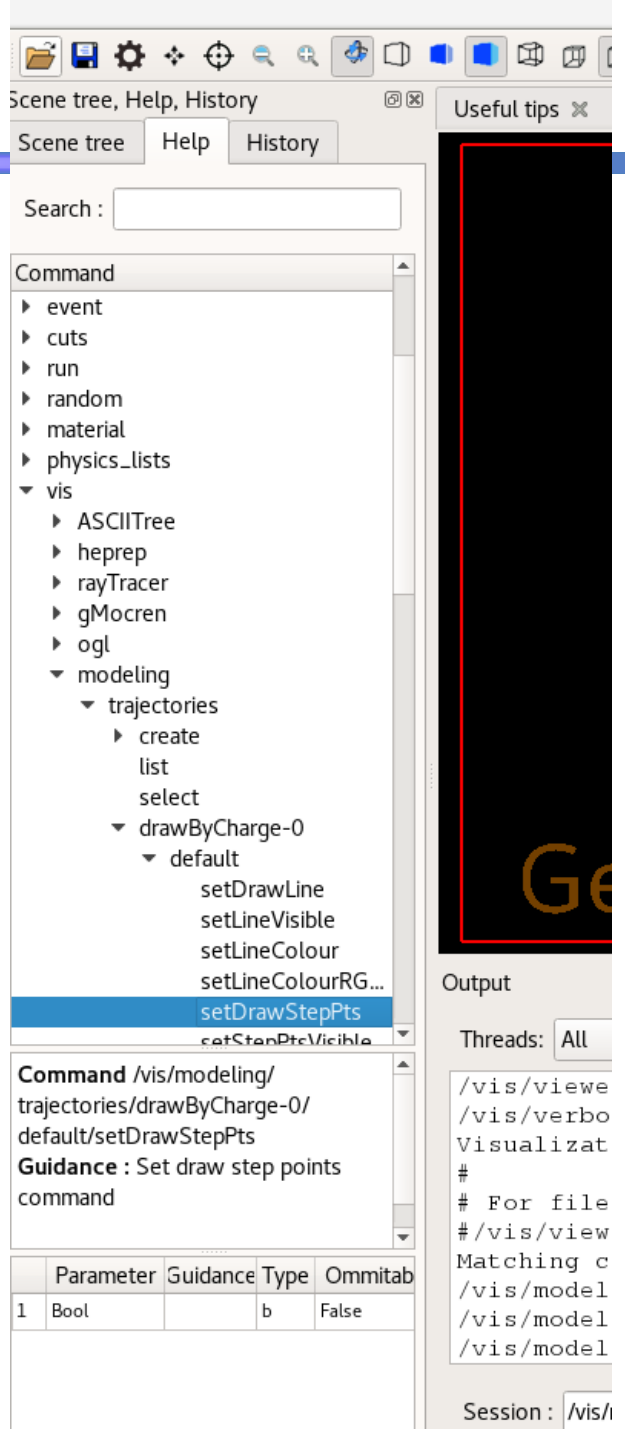
```
# Macro file for the initialization of example B1
# in interactive session
#
# Set some default verbose
/control/verbose 2
/control/saveHistory
/run/verbose 2
#
# Change the default number of threads (in multi-threaded mode)
#/run/numberOfThreads 4
#
# Initialize kernel
/run/initialize
#
# Visualization setting
/control/execute vis.mac
```

# vis.mac

```
# Macro file for the visualization setting in the initialization phase
# of the B1 example when running in interactive mode
#

# Use these open statements to open selected visualization
#
# Use this open statement to create an OpenGL view:
/vis/open OGL 600x600-0+0
#
# Use this open statement to create an OpenInventor view:
#/vis/open OI
#
# Use this open statement to create a .prim file suitable for
# viewing in DAWN:
#/vis/open DAWNFILE
#
# Use this open statement to create a .heprep file suitable for
# viewing in HepRApp:
#/vis/open HepRepFile
#
# Use this open statement to create a .wrl file suitable for
# viewing in a VRML viewer:
#/vis/open VRML2FILE
#
```

# vis.mac （续）

```
# Disable auto refresh and quieten vis messages whilst scene and
# trajectories are established:
/vis/viewer/set/autoRefresh false
/vis/verbose errors
#
# Draw geometry:
/vis/drawVolume
#
# Specify view angle:
/vis/viewer/set/viewpointVector -1 0 0
/vis/viewer/set/lightsVector -1 0 0
#
# Specify style (surface, wireframe, auxiliary edges,...)
/vis/viewer/set/style wireframe
/vis/viewer/set/auxiliaryEdge true
/vis/viewer/set/lineSegmentsPerCircle 100
#
# Draw smooth trajectories at end of event, showing trajectory points
# as markers 2 pixels wide:
/vis/scene/add/trajectories smooth
/vis/modeling/trajectories/create/drawByCharge
/vis/modeling/trajectories/drawByCharge-0/default/setDrawStepPts true
/vis/modeling/trajectories/drawByCharge-0/default/setStepPtsSize 2
# (if too many tracks cause core dump => /tracking/storeTrajectory 0)
```



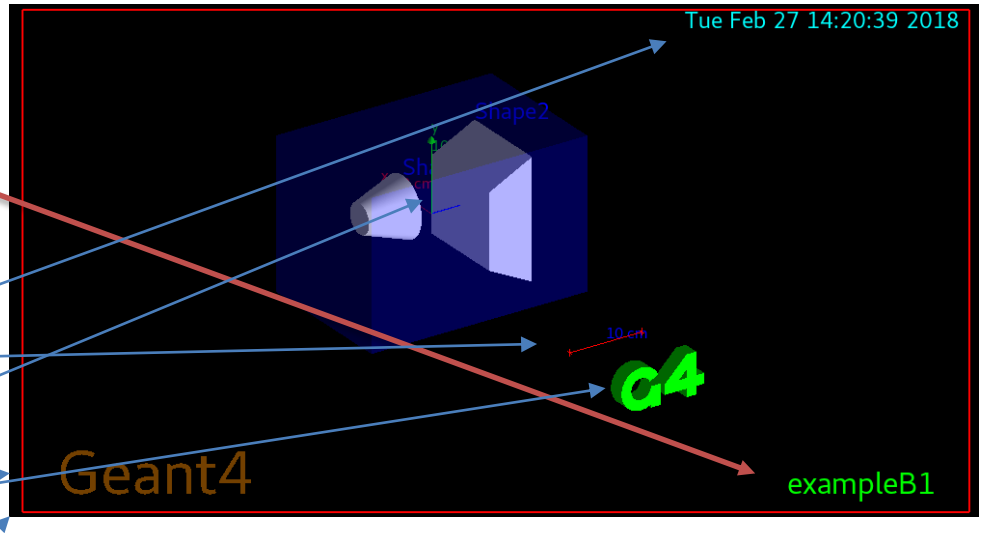

siguang@pku.edu.cn

# vis.mac （续）

```
# Draw hits at end of event:
#/vis/scene/add/hits
#
# To draw only gammas:
#/vis/filtering/trajectories/create/particleFilter
#/vis/filtering/trajectories/particleFilter-0/add gamma
#
# To invert the above, drawing all particles except gammas,
# keep the above two lines but also add:
#/vis/filtering/trajectories/particleFilter-0/invert true
#
# Many other options are available with /vis/modeling and /vis/filtering.
# For example, to select colour by particle ID:
#/vis/modeling/trajectories/create/drawByParticleID
#/vis/modeling/trajectories/drawByParticleID-0/default/setDrawStepPts true
# To select or override default colours (note: e+ is blue by default):
#/vis/modeling/trajectories/list
#/vis/modeling/trajectories/drawByParticleID-0/set e+ yellow
#
# To superimpose all of the events from a given run:
/vis/scene/endOfEventAction accumulate
```

# vis.mac （续）

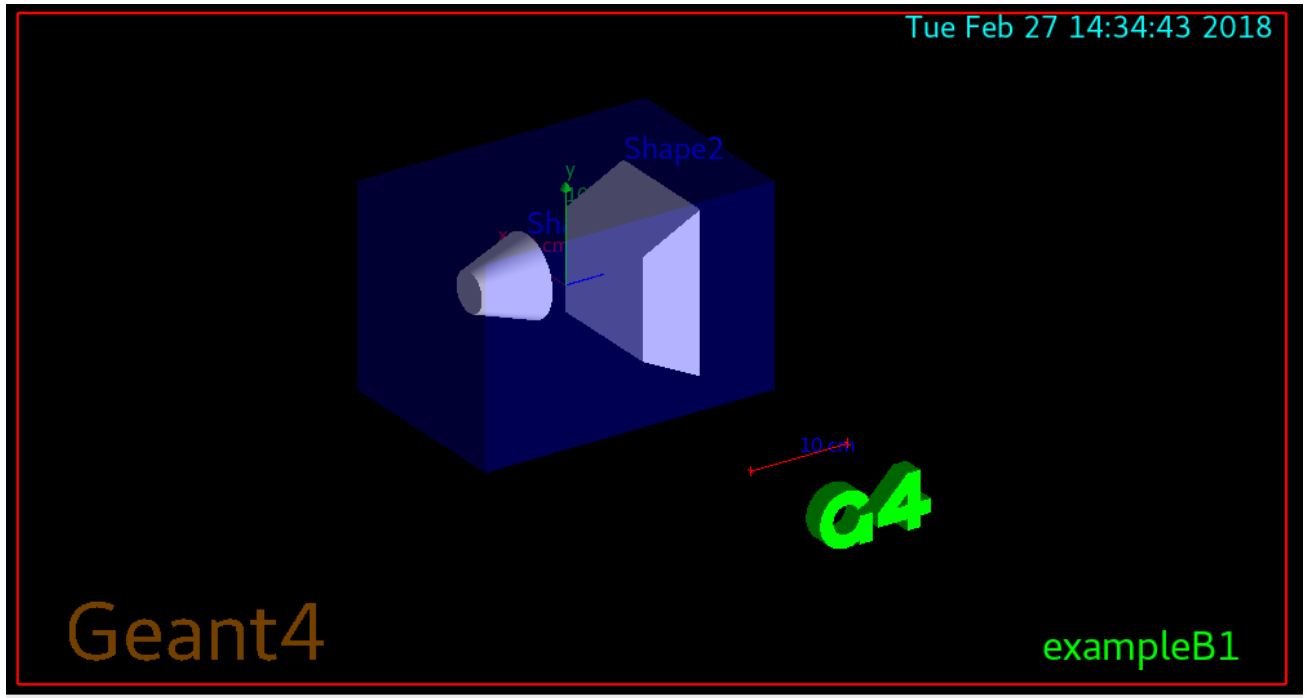
```
# Decorations
# Name
/vis/set/textColour green
/vis/set/textLayout right
/vis/scene/add/text2D 0.9 -.9 24 ! ! exampleB1
# or, if your system does not support right-adjustment
#/vis/scene/add/text2D 0 -.9 24 ! ! exampleB1
/vis/set/textLayout    # Revert to normal (left adjusted) layout
/vis/set/textColour    # Revert to default text colour (blue)
#
# Axes, scale, etc.
/vis/scene/add/scale   # Simple scale line
/vis/scene/add/axes    # Simple axes: x=red, y=green, z=blue.
/vis/scene/add/eventID # Drawn at end of event
/vis/scene/add/date    # Date stamp
/vis/scene/add/logo2D  # Simple logo
/vis/scene/add/logo    # 3D logo
# Frame
/vis/set/colour red
/vis/set/lineWidth 2
/vis/scene/add/frame   # Simple frame around the view
/vis/set/colour        # Revert to default colour (white)
/vis/set/lineWidth     # Revert to default line width (1.)
```



# vis.mac （续）

```
# Attach text to one edge of Shape1, with a small, fixed offset
/vis/scene/add/text 0 6 -4 cm 18 4 4 Shape1
# Attach text to one corner of Shape2, with a small, fixed offset
/vis/scene/add/text 6 7 10 cm 18 4 4 Shape2
#
# To get nice view
# Make the "World" box invisible
/vis/geometry/set/visibility World 0 false
# "Envelope" is transparent blue to represent water
/vis/geometry/set/colour Envelope 0 0 0 1 .3
/vis/viewer/set/style surface
/vis/viewer/set/hiddenMarker true
/vis/viewer/set/viewpointThetaPhi 120 150
#
# Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/verbose warnings
#
# For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush
```

# ./exampleB1

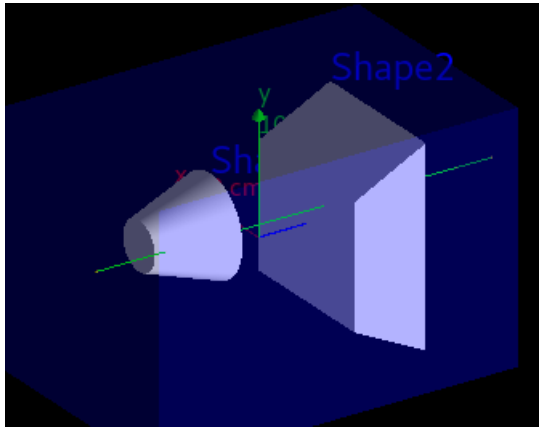



Output

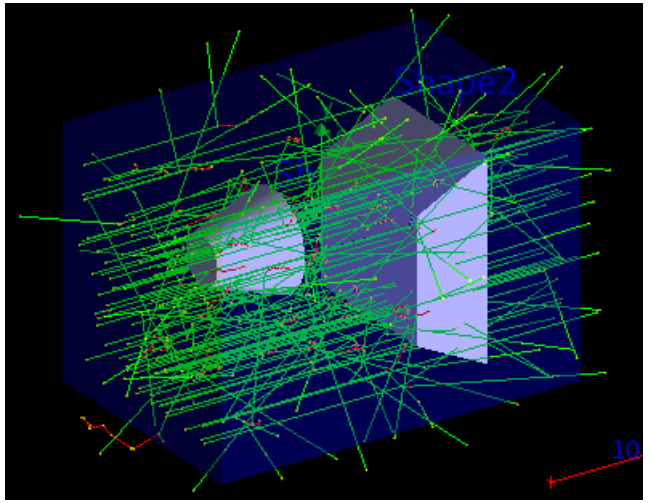
Threads: All

```
/vis/viewer/set/viewpointThetaPhi 120 150
#
# Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/viewer/refresh
/vis/verbose warnings
Visualization verbosity changed to warnings (3)
#
# For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush
```

Session : /run/beamOn 1




/run/beamOn 1



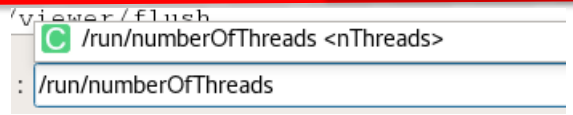

/run/beamOn 1000

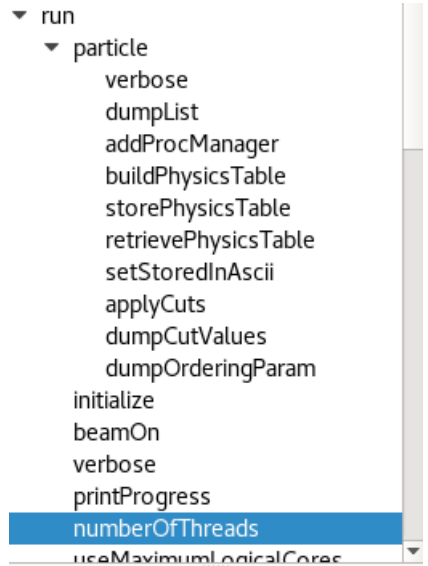
siguang@pku.edu.cn

# run1.mac

```
# Macro file for example B1
#
# Can be run in batch, without graphic
# or interactively: Idle> /control/execute run1.mac
#
# Change the default number of workers (in multi-threading mode)
#/run/numberOfWorkers 4
#
# Initialize kernel
/run/initialize
#
/control/verbose 2
/run/verbose 2
/event/verbose 0
/tracking/verbose 1
#
# gamma 6 MeV to the direction (0.,0.,1.)
#
/gun/particle gamma
/gun/energy 6 MeV
#
/run/beamOn 5
```

G4.10.4不认识该命令
用numberOfThreads代替
G4.11.1.1这个命令也不能执行了




```
/run/numberOfThreads 4
illegal application state -- command refused:"/run/numberOfThreads 4"
```

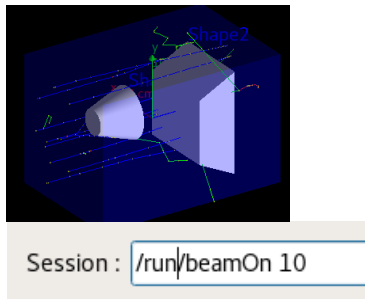
siguang@pku.edu.cn

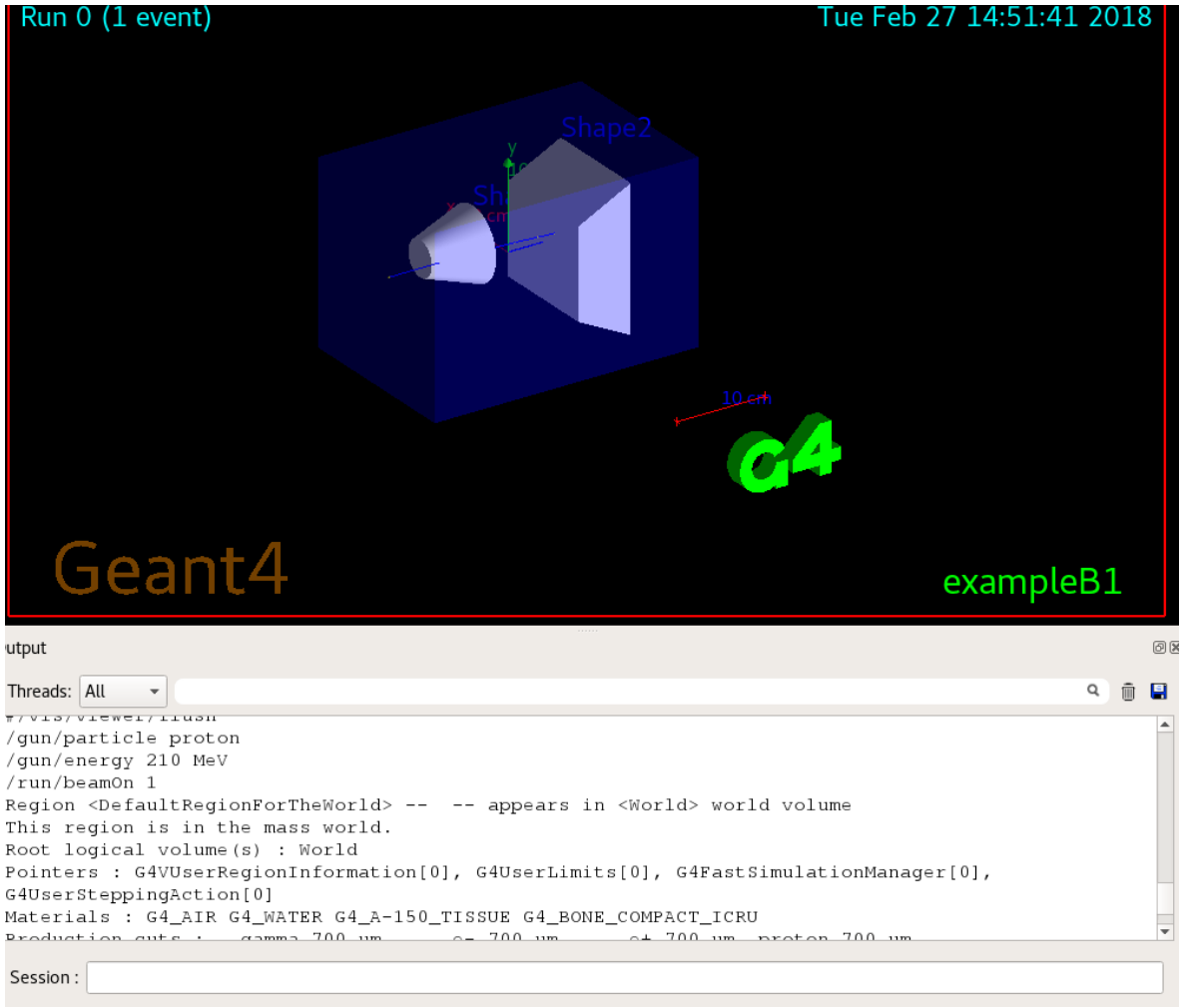
# run1.mac

```
# proton 210 MeV to the direction (0.,0.,1.)
#
/gun/particle proton
/gun/energy 210 MeV
/tracking/verbose 2
#
/run/beamOn 1
```

$./exampleB1 run1.mac

```
Number of memory pools allocated: 4; of which, static: 0
Dynamic pools deleted: 4 / Total memory freed: 0.0058 MB
============================================================
G4Allocator objects are deleted.
UImanager deleted.
StateManager deleted.
RunManagerKernel is deleted. Good bye :)
g4root@debian:~/work/g4/10.4/examples/basic/B1/bin$
```




输入 exit 退出

siguang@pku.edu.cn

# run2.mac

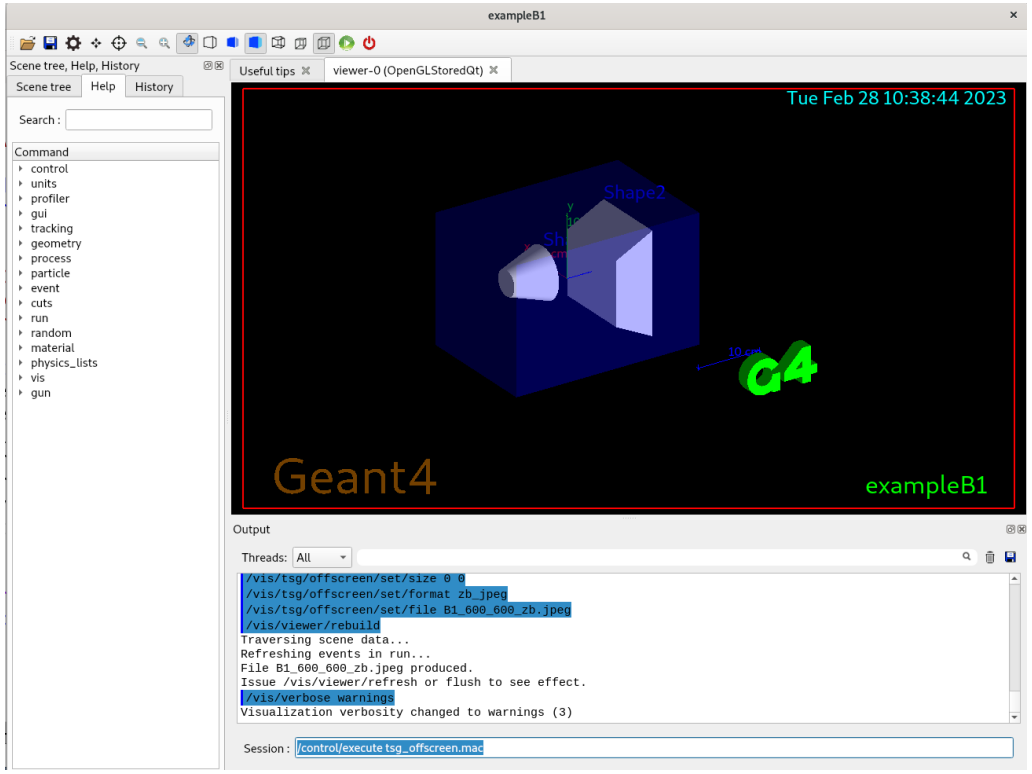
```
# Macro file for example B1
#
# To be run preferably in batch, without graphics:
# % exampleB1 run2.mac
#
#/run/numberOfWorkers 4  #Never use after G4.10.4!!!
/run/numberOfThreads 1
/run/initialize
#
/control/verbose 2
/run/verbose 2
#
# gamma 6 MeV to the direction (0.,0.,1.)
# 10000 events
#
/gun/particle gamma
/gun/energy 6 MeV
#
/run/printProgress 100
/run/beamOn 1000
```

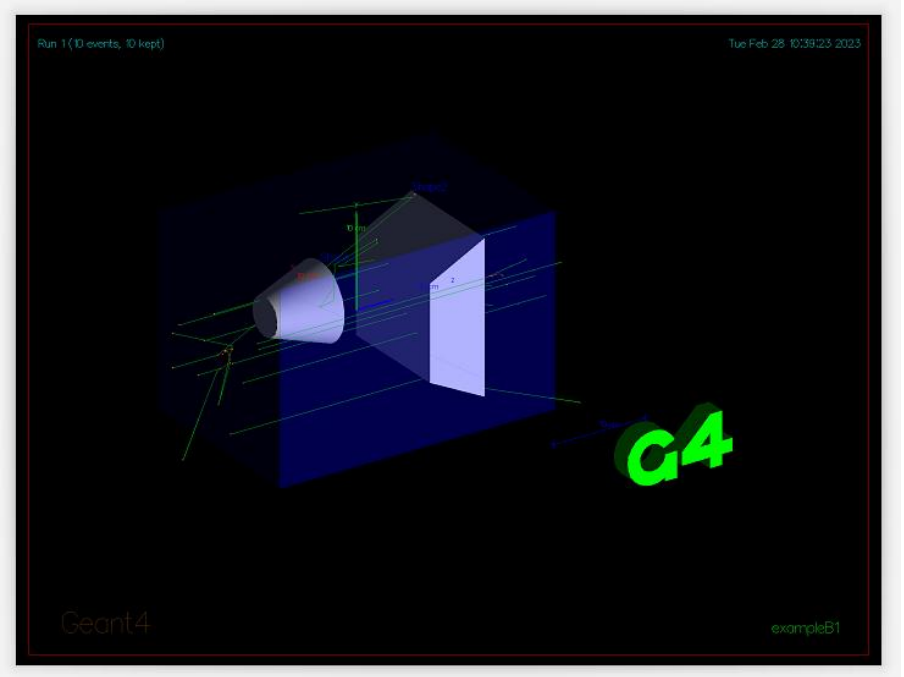
```
#
# proton 210 MeV to the direction (0.,0.,1.)
# 1000 events
#
/gun/particle proton
/gun/energy 210 MeV
#
/run/beamOn 1000
```

siguang@pku.edu.cn

# tsg_offscreen.mac （G4.11.1.1）

- 两步：
  - ./exampleB1
  - 交互对话框输入：/control/execute tsg_offscreen.mac



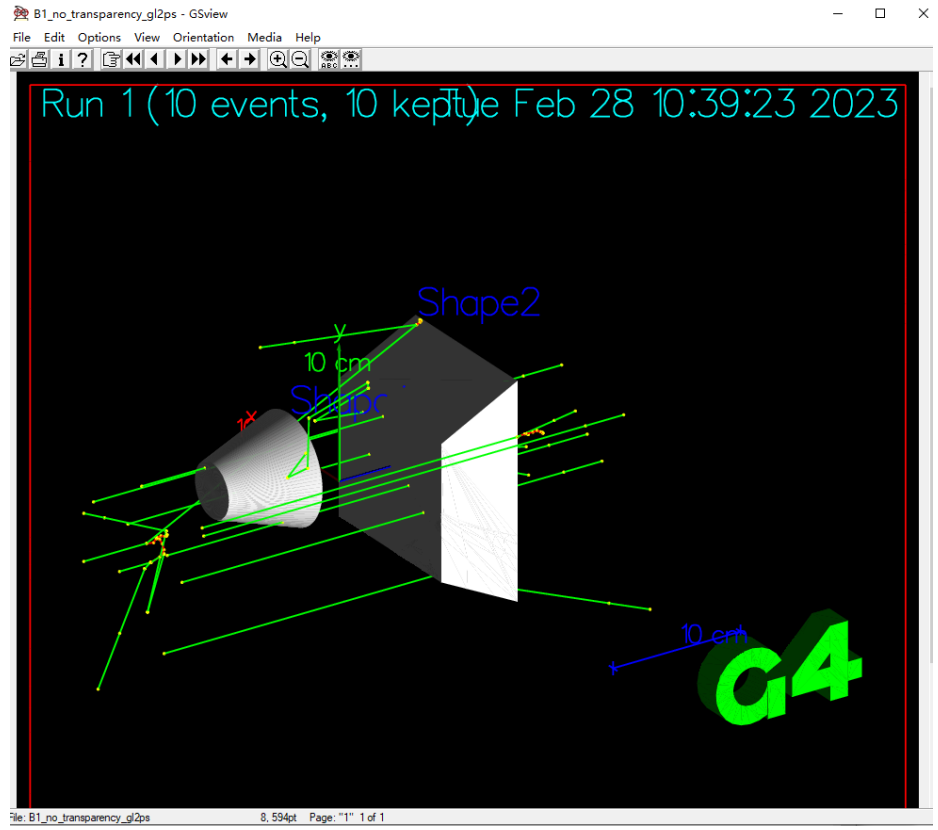

执行后生成很多图片文件

# tsg_offscreen.mac

```
#  Below is a sequence to produce files at the various formats
# (execute vis.mac first to have some scene to visualize).
#  Note that a TSG offscreen viewer is not an "auto refresh" one, then
# to produce a picture, you have to do:
#   /vis/viewer/rebuild
# (a /vis/viewer/refresh or flush may not be sufficient, for example with plotting).

/vis/verbose confirmations

/vis/open TSG_OFFSCREEN
/vis/viewer/rebuild  # to produce the default g4tsg_offscreen_zb_png_1.png file.
/run/beamOn 10
/vis/viewer/rebuild  # to produce g4tsg_offscreen_zb_png_2.png file.

/vis/tsg/offscreen/set/file auto B1_ true  #true is to reset the index.
/vis/viewer/rebuild  # to produce B1_1.png file.
/run/beamOn 10
/vis/viewer/rebuild  # to produce B1_2.png file.
```

```
# gl2ps does not handle transparency, it could
# be usefull to not draw the transparent objects:
/vis/tsg/offscreen/set/transparency false
/vis/tsg/offscreen/set/file B1_no_transparency_gl2ps.ps
/vis/viewer/rebuild
/vis/tsg/offscreen/set/transparency true
```

无透明物的效果



# CMakeLists.txt

```
#----------------------------------------------------------------------------
# Setup the project
cmake_minimum_required(VERSION 2.6 FATAL_ERROR)
project(B1)

#----------------------------------------------------------------------------
# Find Geant4 package, activating all available UI and Vis drivers by default
# You can set WITH_GEANT4_UIVIS to OFF via the command line or ccmake/cmake-gui
# to build a batch mode only executable
#
option(WITH_GEANT4_UIVIS "Build example with Geant4 UI and Vis drivers" ON)
if(WITH_GEANT4_UIVIS)
  find_package(Geant4 REQUIRED ui_all vis_all)
else()
  find_package(Geant4 REQUIRED)
endif()

#----------------------------------------------------------------------------
# Setup Geant4 include directories and compile definitions
# Setup include directory for this project
#
include(${Geant4_USE_FILE})
include_directories(${PROJECT_SOURCE_DIR}/include)
```

# CMakeLists.txt (续)

```
#----------------------------------------------------------------
# Locate sources and headers for this project
# NB: headers are included so they will show up in IDEs
#
file(GLOB sources ${PROJECT_SOURCE_DIR}/src/*.cc)
file(GLOB headers ${PROJECT_SOURCE_DIR}/include/*.hh)

#----------------------------------------------------------------
# Add the executable, and link it to the Geant4 libraries
#
add_executable(exampleB1 exampleB1.cc ${sources} ${headers})
target_link_libraries(exampleB1 ${Geant4_LIBRARIES})
```

# CMakeLists.txt (续)

```
#----------------------------------------------------------------------------
# Copy all scripts to the build directory, i.e. the directory in which we
# build B1. This is so that we can run the executable directly because it
# relies on these scripts being in the current working directory.
#
set(EXAMPLEB1_SCRIPTS
  exampleB1.in
  exampleB1.out
  init_vis.mac
  run1.mac
  run2.mac
  vis.mac
  )

foreach(_script ${EXAMPLEB1_SCRIPTS})
  configure_file(
    ${PROJECT_SOURCE_DIR}/${_script}
    ${PROJECT_BINARY_DIR}/${_script}
    COPYONLY
    )
endforeach()
```