Geant4进展README

Geant4进展README

- 输出ROOT文件（FourthVersion）
  - RunAction.cc
  - SteppingAction.cc
  - 结果图
    - Energy
    - ProcessName
    - Position
    - Momentum
- 数据从Step直接传到Run(ThirdVersion)
  - RunAction.hh
  - RunAction.cc
  - SteppingAction.hh
  - SteppingAction.cc
  - 结果文档
- 数据先从Step传到Event，再从Event传到Run(SecondVersion)
  - RunAction.hh
  - RunAction.cc
  - EventAction.hh
  - EventAction.cc
  - SteppingAction.hh
  - SteppingAction.cc
  - 结果文档

输出ROOT文件（FourthVersion）

RunAction.cc

```
#include "RunAction.hh"
#include "G4RunManager.hh"
#include "G4Run.hh"
#include "G4AnalysisManager.hh"
#include "G4UnitsTable.hh"
#include "G4SystemOfUnits.hh"

namespace BTS
{

RunAction::RunAction()
{
  auto analysisManager = G4AnalysisManager::Instance();
  analysisManager->SetVerboseLevel(1);
  analysisManager->SetNtupleMerging(true);

  // Creating ntuple
  //
  analysisManager->CreateNtuple("Ntuple1", "BTSRoot");
```

```
  // Energy and ProcessName
  analysisManager->CreateNtupleDColumn("Energy"); //Column0
  analysisManager->CreateNtupleSColumn(0, "ProcessName"); //Column1

  // Position
  analysisManager->CreateNtupleDColumn("Position_X"); //Column2
  analysisManager->CreateNtupleDColumn("Position_Y"); //Column3
  analysisManager->CreateNtupleDColumn("Position_Z"); //Column4

  // Momentum
  analysisManager->CreateNtupleDColumn("Momentum_X"); //Column5
  analysisManager->CreateNtupleDColumn("Momentum_Y"); //Column6
  analysisManager->CreateNtupleDColumn("Momentum_Z"); //Column7

  analysisManager->FinishNtuple();
}

void RunAction::BeginOfRunAction(const G4Run*)
{
  // Get analysis manager
  auto analysisManager = G4AnalysisManager::Instance();
    G4String fileName = "BTS.root";
  analysisManager->OpenFile(fileName);
  G4cout << "Using " << analysisManager->GetType() << G4endl;
}

void RunAction::EndOfRunAction(const G4Run*)
{
  auto analysisManager = G4AnalysisManager::Instance();
  analysisManager->Write();
  analysisManager->CloseFile();
}

}
```

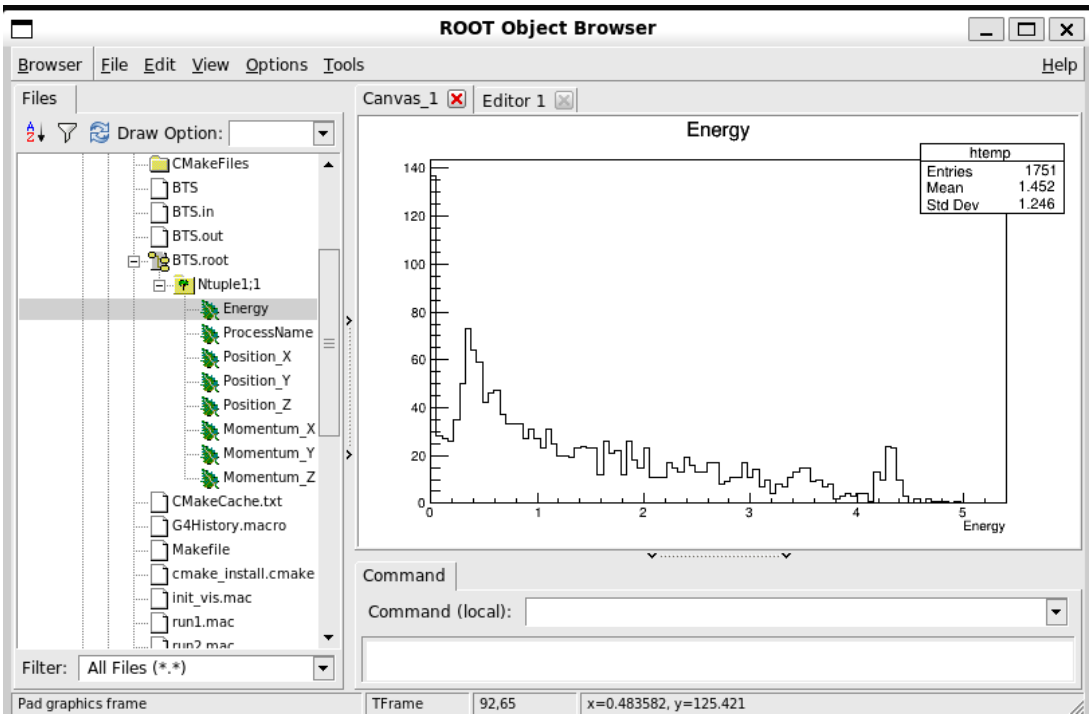
SteppingAction.cc

```
#include "SteppingAction.hh"
#include "EventAction.hh"
#include "G4RunManager.hh"
#include "G4Step.hh"
#include "G4Track.hh"
#include "G4StepPoint.hh"
#include "G4AnalysisManager.hh"
#include "G4Event.hh"
#include "G4UnitsTable.hh"
#include "Randomize.hh"
#include <iomanip>

namespace BTS
{

SteppingAction::SteppingAction(EventAction* eventAction):
fEventAction(eventAction)
{}
```

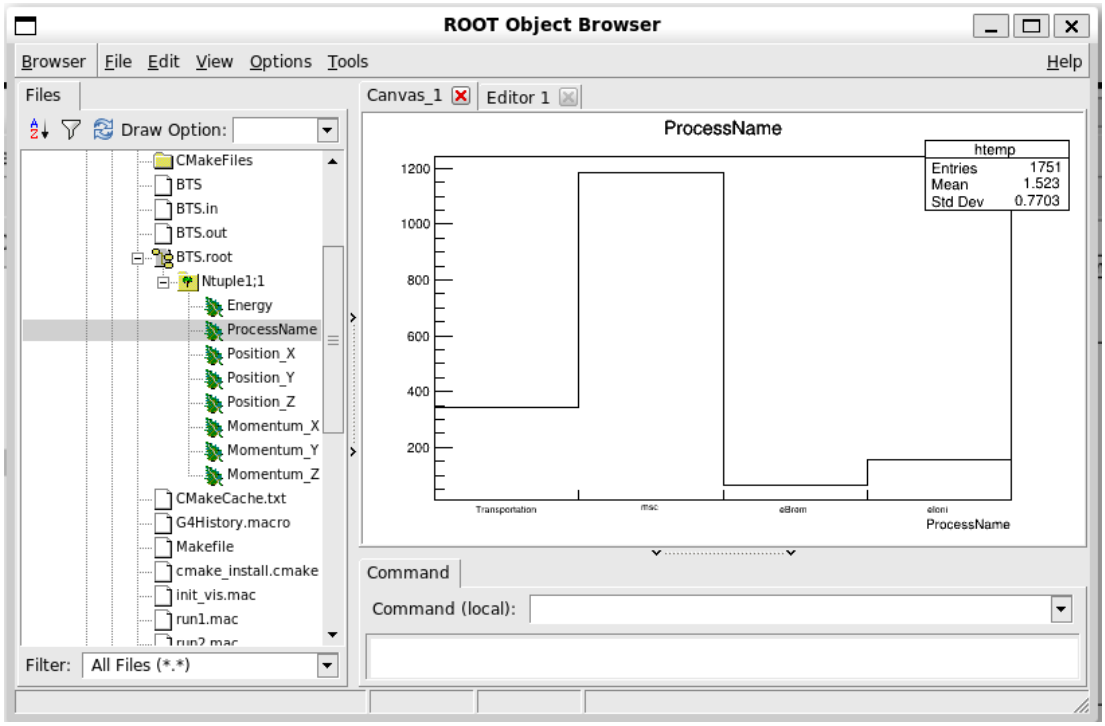
```
void SteppingAction::UserSteppingAction(const G4Step* step)
{
  G4Track* track = step->GetTrack();
  G4VPhysicalVolume* volume = track->GetVolume();

  auto analysisManager = G4AnalysisManager::Instance();

  G4int trackId = track->GetTrackID();
  G4int parentId = track->GetParentID();
  G4double energy = track->GetKineticEnergy();

  G4ThreeVector position = track->GetPosition();
  G4double pos_x = position.x();
  G4double pos_y = position.y();
  G4double pos_z = position.z();

  G4ThreeVector momentum = track->GetMomentum();
  G4double mmt_x = momentum.x();
  G4double mmt_y = momentum.y();
  G4double mmt_z = momentum.z();

  G4String particleId = track->GetParticleDefinition()->GetParticleName();
  G4String processName = step->GetPostStepPoint()->GetProcessDefinedStep()-
>GetProcessName();

if (track->GetDefinition()->GetParticleName()!="gamma" && volume-
>GetName()!="World")
  {
  analysisManager->FillNtupleDColumn(0, 0, energy);
  analysisManager->FillNtupleSColumn(0, 1, processName);

  analysisManager->FillNtupleDColumn(0, 2, pos_x);
  analysisManager->FillNtupleDColumn(0, 3, pos_y);
  analysisManager->FillNtupleDColumn(0, 4, pos_z);

  analysisManager->FillNtupleDColumn(0, 5, mmt_x);
  analysisManager->FillNtupleDColumn(0, 6, mmt_y);
  analysisManager->FillNtupleDColumn(0, 7, mmt_z);

  analysisManager->AddNtupleRow();
  }
}

}
```

结果图

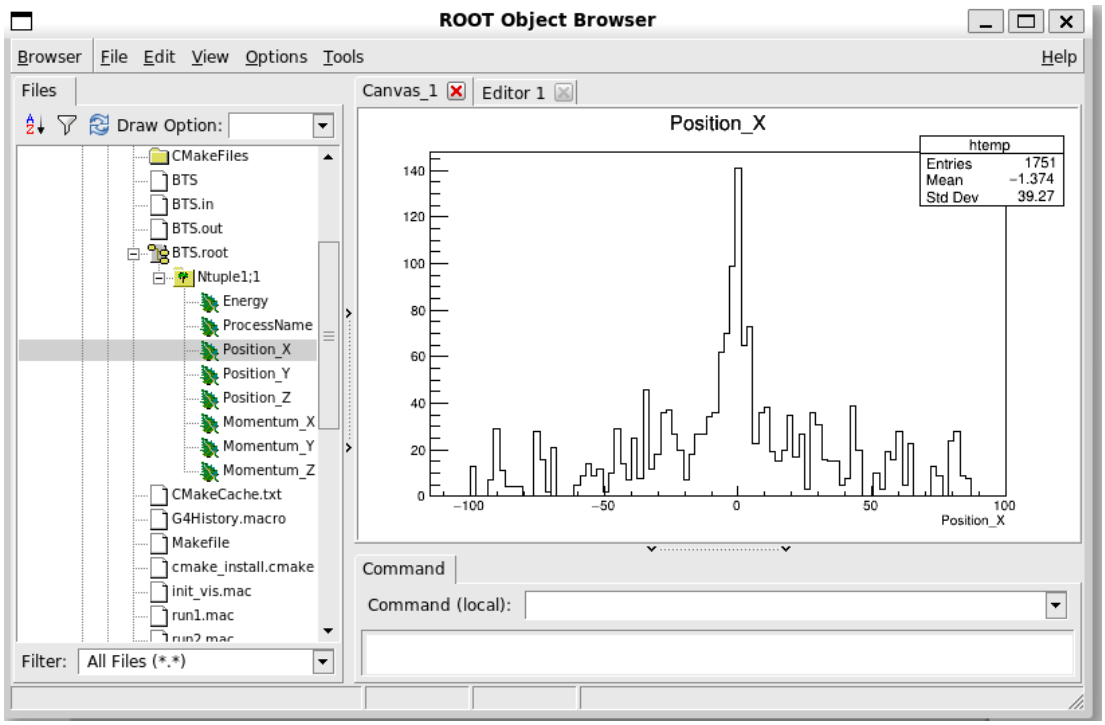
Energy

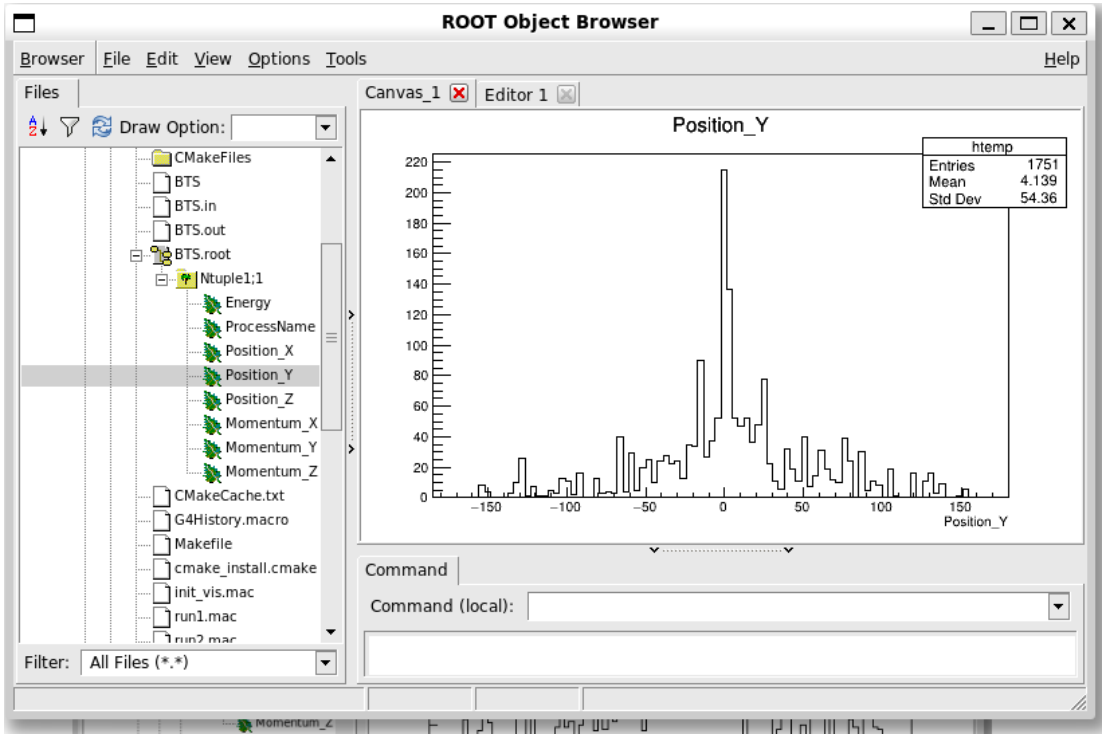


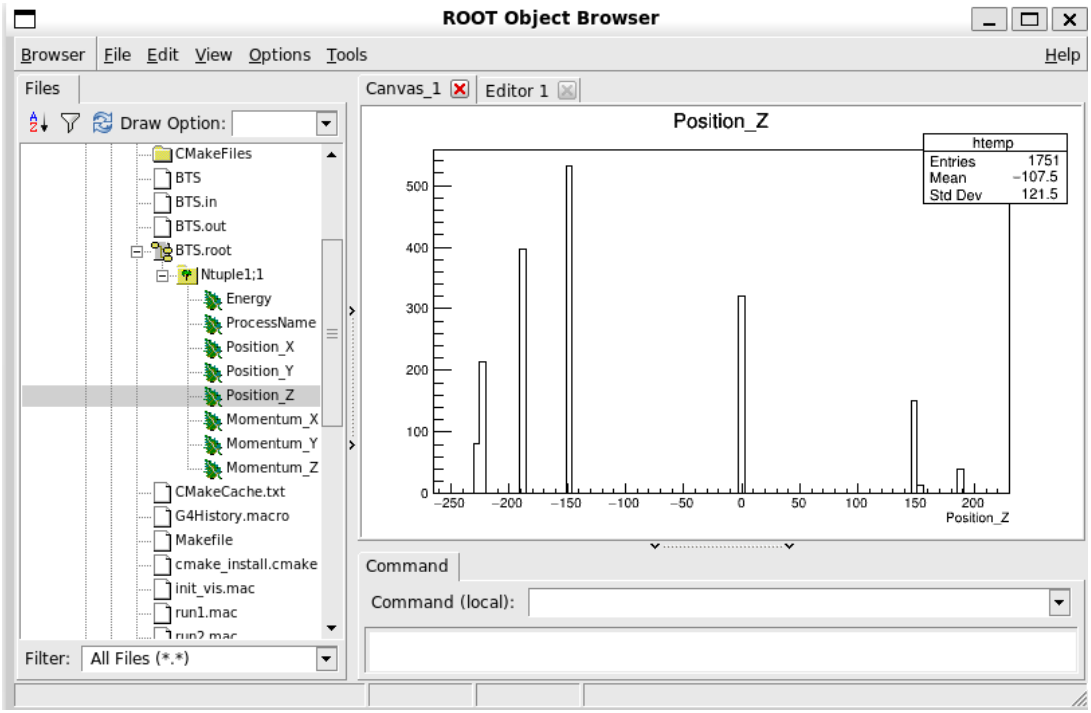
ProcessName



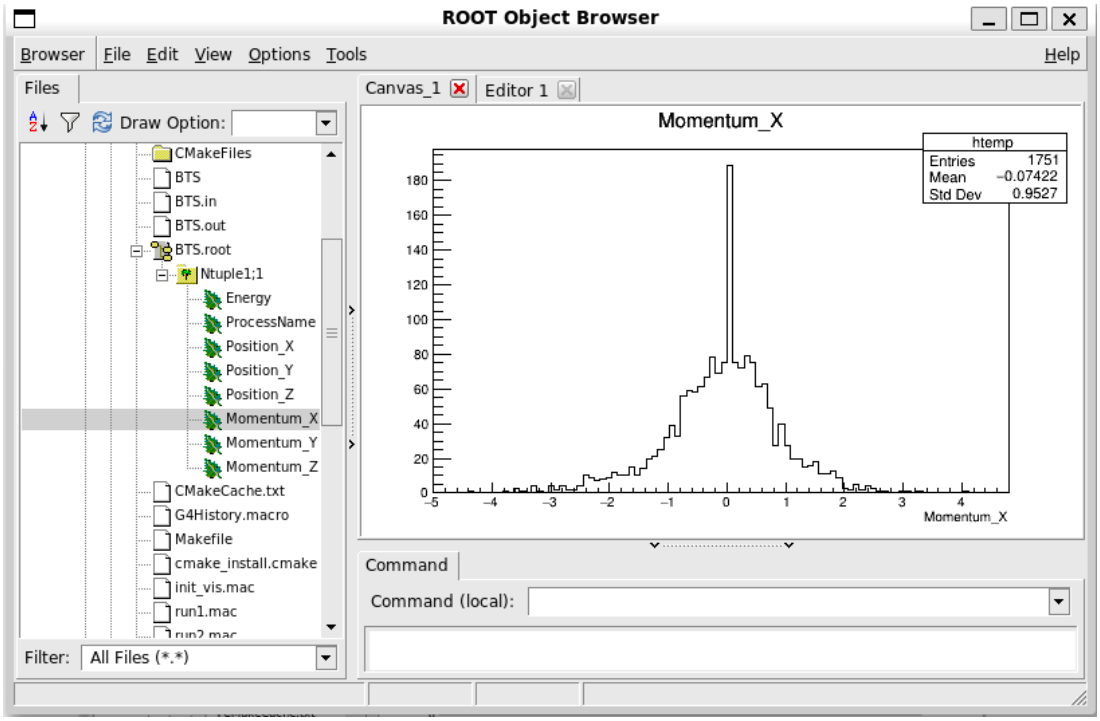
Position

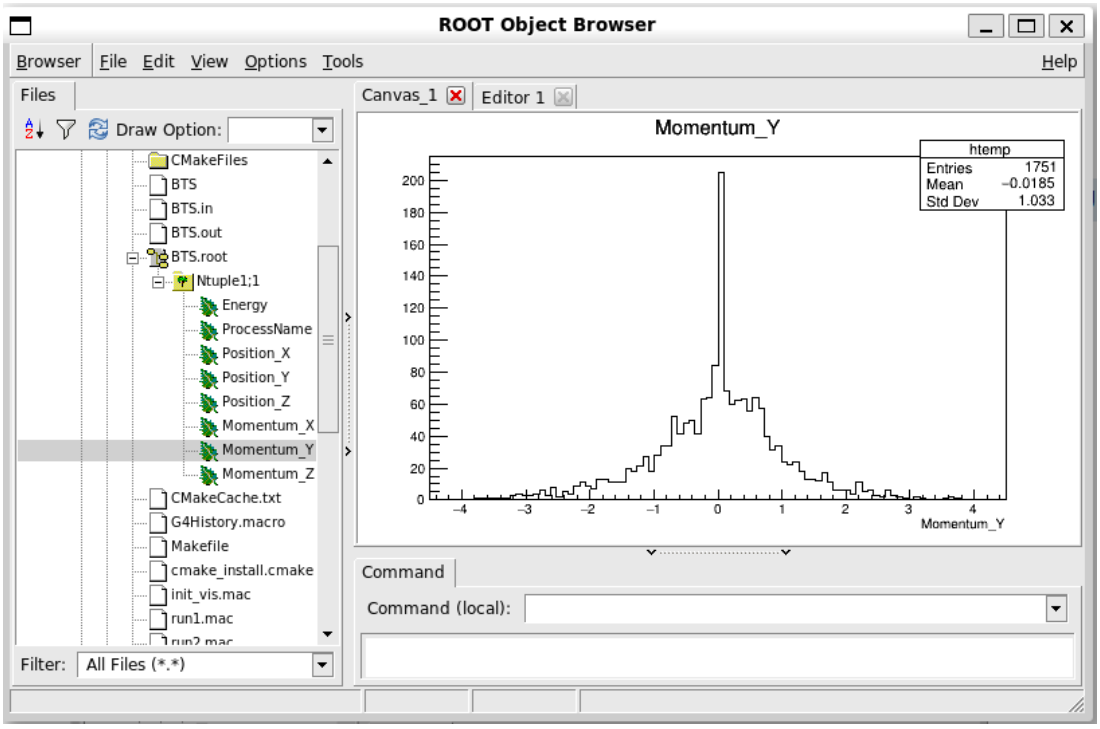

ROOT Object Browser
Browser File Edit View Options Tools Help
Files
Draw Option:
CMakeFiles
BTS
BTS.in
BTS.out
BTS.root
Ntuple1;1
Energy
ProcessName
Position_X
Position_Y
Position_Z
Momentum_X
Momentum_Y
Momentum_Z
CMakeCache.txt
G4History.macro
Makefile
cmake_install.cmake
init_vis.mac
run1.mac
Filter: All Files (*.*)
Canvas_1
Editor 1
Position_X
htemp
Entries 1751
Mean −1.374
Std Dev 39.27
Position_X
Command
Command (local):

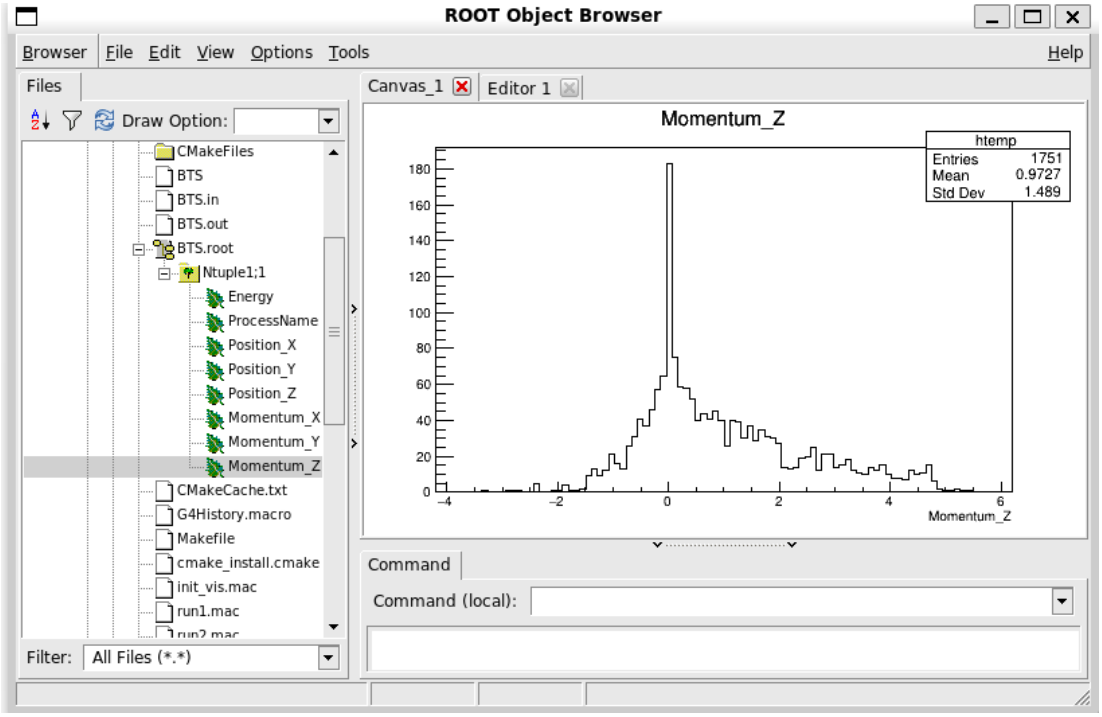

ROOT Object Browser
Browser File Edit View Options Tools Help
Files
Draw Option:
CMakeFiles
BTS
BTS.in
BTS.out
BTS.root
Ntuple1;1
Energy
ProcessName
Position_X
Position_Y
Position_Z
Momentum_X
Momentum_Y
Momentum_Z
CMakeCache.txt
G4History.macro
Makefile
cmake_install.cmake
init_vis.mac
run1.mac
Filter: All Files (*.*)
Canvas_1
Editor 1
Position_Y
htemp
Entries 1751
Mean 4.139
Std Dev 54.36
Position_Y
Command
Command (local):


ROOT Object Browser
Browser File Edit View Options Tools Help
Files
Draw Option:
CMakeFiles
BTS
BTS.in
BTS.out
BTS.root
Ntuple1;1
Energy
ProcessName
Position_X
Position_Y
Position_Z
Momentum_X
Momentum_Y
Momentum_Z
CMakeCache.txt
G4History.macro
Makefile
cmake_install.cmake
init_vis.mac
run1.mac
Filter: All Files (*.*)
Canvas_1
Editor 1
Position_Z
htemp
Entries 1751
Mean −107.5
Std Dev 121.5
Position_Z
Command
Command (local):

Momentum


ROOT Object Browser
Browser File Edit View Options Tools Help
Files
Draw Option:
CMakeFiles
BTS
BTS.in
BTS.out
BTS.root
Ntuple1;1
Energy
ProcessName
Position_X
Position_Y
Position_Z
Momentum_X
Momentum_Y
Momentum_Z
CMakeCache.txt
G4History.macro
Makefile
cmake_install.cmake
init_vis.mac
run1.mac
Filter: All Files (*.*)
Canvas_1
Editor 1
Momentum_X
htemp
Entries 1751
Mean −0.07422
Std Dev 0.9527
Momentum_X
Command
Command (local):





数据从Step直接传到Run(ThirdVersion)

RunAction.hh

```
#ifndef BTSRunAction_h
#define BTSRunAction_h 1

#include "G4UserRunAction.hh"
#include "G4Accumulable.hh"
#include "globals.hh"
#include "G4ThreeVector.hh"
#include <fstream>

class G4Run;
```

```
namespace BTS
{

class RunAction : public G4UserRunAction
{
  public:
    RunAction();
    ~RunAction() override = default;

    void BeginOfRunAction(const G4Run*) override;
    void EndOfRunAction(const G4Run*) override;
    void SavePosition(G4ThreeVector PosR);
    G4ThreeVector Position_Run;
  private:
    std::ofstream m_File; // 输出文档
};

}

#endif
```

RunAction.cc

```
#include "G4AccumulableManager.hh"
#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4UnitsTable.hh"
#include "G4SystemOfUnits.hh"
#include <fstream>

namespace BTS
{

RunAction::RunAction()
{
}

void RunAction::BeginOfRunAction(const G4Run*)
{
  // Open txt
  m_File.open("ResultOfSimulation", std::ios::out);
  if (!m_File.is_open())
  {
    G4cerr << "Error opening file "<< G4endl;
    exit(1);
  }
}

void RunAction::EndOfRunAction(const G4Run* run)
{
  // Close txt
  m_File.close();
}
```

```
void RunAction::SavePosition(G4ThreeVector PosR)
{
  Position_Run = PosR;
  m_File << "Hit position: " << Position_Run << "\n";
  //G4cout << PosR << G4endl;
}

}
```

SteppingAction.hh

```
#ifndef BTSSteppingAction_h
#define BTSSteppingAction_h 1

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

class G4LogicalVolume;

namespace BTS
{

class RunAction;

class SteppingAction : public G4UserSteppingAction
{
  public:
    SteppingAction(RunAction* runAction);
    ~SteppingAction() override = default;

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

  private:
    RunAction* fRunAction = nullptr;
};

}

#endif
```

SteppingAction.cc

```
#include "RunAction.hh"
#include "SteppingAction.hh"
#include "EventAction.hh"
#include "DetectorConstruction.hh"
#include "G4Track.hh"
#include "G4Step.hh"
#include "G4Event.hh"
#include "G4RunManager.hh"
#include "G4LogicalVolume.hh"

namespace BTS
```

```
{

SteppingAction::SteppingAction(RunAction* runAction)
: fRunAction(runAction)
{}

void SteppingAction::UserSteppingAction(const G4Step* step)
{
  auto track = step->GetTrack();
  G4LogicalVolume* volume = step->GetPreStepPoint()->GetTouchableHandle()-
>GetVolume()->GetLogicalVolume();
  G4ThreeVector position = track->GetPosition();

  //G4cout << position << G4endl;
  if (track->GetDefinition()->GetParticleName()!="gamma" && volume-
>GetName()!="World")
  {
    fRunAction->SavePosition(position);
  }
}

}
```

结果文档

- 发射100个例子，有841条数据，是正常的

数据先从Step传到Event，再从Event传到Run(SecondVersion)

RunAction.hh

```
#ifndef BTSRunAction_h
#define BTSRunAction_h 1

#include "G4UserRunAction.hh"
#include "G4Accumulable.hh"
#include "globals.hh"
#include "G4ThreeVector.hh"
#include <fstream>

class G4Run;

namespace BTS
{

class RunAction : public G4UserRunAction
{
  public:
    RunAction();
    ~RunAction() override = default;

    void BeginOfRunAction(const G4Run*) override;
    void EndOfRunAction(const G4Run*) override;
```

```
    void SavePosition(G4ThreeVector PosR);

    G4ThreeVector Position_Run;
  private:
    std::ofstream m_File;
};

}

#endif
```

RunAction.cc

```
#include "G4LogicalVolumeStore.hh"
#include "G4LogicalVolume.hh"
#include "G4UnitsTable.hh"
#include "G4SystemOfUnits.hh"
#include <fstream>

namespace BTS
{

RunAction::RunAction()
{
}

void RunAction::BeginOfRunAction(const G4Run*)
{
  // Open txt
  m_File.open("ResultOfSimulation", std::ios::out);
  if (!m_File.is_open())
  {
    G4cerr << "Error opening file "<< G4endl;
    exit(1);
  }
}


void RunAction::EndOfRunAction(const G4Run* run)
{
  // Close txt
  m_File.close();
}

void RunAction::SavePosition(G4ThreeVector PosR)
{
  Position_Run = PosR;
  m_File << "Hit position: " << Position_Run << "\n";
  //G4cout << PosR <<G4endl;
}

}
```

EventAction.hh

```
#ifndef BTSEventAction_h
#define BTSEventAction_h 1

#include "G4UserEventAction.hh"
#include "globals.hh"
#include "G4ThreeVector.hh"

namespace BTS
{

class RunAction;

class EventAction : public G4UserEventAction
{
  public:
    EventAction(RunAction* runAction);
    ~EventAction() override = default;

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

    void GetPosition(G4ThreeVector Pos);
    G4ThreeVector Position_Event;
    //void AddEdep(G4double edep) { fEdep += edep; }

  private:
    RunAction* fRunAction = nullptr;
    //G4double   fEdep = 0.;
};

}

#endif
```

EventAction.cc

```
#include "EventAction.hh"
#include "RunAction.hh"
#include "G4ThreeVector.hh"
#include "G4Event.hh"
#include "G4RunManager.hh"

namespace BTS
{

EventAction::EventAction(RunAction* runAction)
: fRunAction(runAction)
{}

void EventAction::BeginOfEventAction(const G4Event*)
{
}
```

```
void EventAction::EndOfEventAction(const G4Event*)
{
    fRunAction->SavePosition(Position_Event);
}

void EventAction::GetPosition(G4ThreeVector Pos)
{
    Position_Event = Pos;
    //G4cout << Position_Event << G4endl; 这里的输出没有损失，若发射10个例子，传到Run后就只剩10个数据
    //G4cout<< "Geant4  Test_Event_MemberFunction  Geant4" <<G4endl;
}

}
```

SteppingAction.hh

```
#ifndef BTSSteppingAction_h
#define BTSSteppingAction_h 1

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

class G4LogicalVolume;

namespace BTS
{

class EventAction;

class SteppingAction : public G4UserSteppingAction
{
  public:
    SteppingAction(EventAction* eventAction);
    ~SteppingAction() override = default;

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

  private:
    EventAction* fEventAction = nullptr;
    G4LogicalVolume* fScoringVolume = nullptr;
};

}

#endif
```

SteppingAction.cc

```
#include "SteppingAction.hh"
#include "EventAction.hh"
#include "DetectorConstruction.hh"
#include "G4Track.hh"
#include "G4Step.hh"
#include "G4Event.hh"
#include "G4RunManager.hh"
#include "G4LogicalVolume.hh"

namespace BTS
{

SteppingAction::SteppingAction(EventAction* eventAction)
: fEventAction(eventAction)
{}


void SteppingAction::UserSteppingAction(const G4Step* step)
{
  auto track = step->GetTrack();
  G4LogicalVolume* volume = step->GetPreStepPoint()->GetTouchableHandle()-
>GetVolume()->GetLogicalVolume();
  G4ThreeVector position = track->GetPosition();

  //G4cout << position << G4endl;
  fEventAction->GetPosition(position);
}

}
```

结果文档

- 发射100个例子，只有100条数据，而传到Event那步都还是有841条，从Event传到Run就丢了