Zastosowanie pakietu Geant4 w fizyce jądrowej Wykład 10

Aleksandra Fijałkowska

14 maja 2020

Wyniki symulacji, plik z wynikami

Dziś przed nami mały, wdzięczny i bardzo ważny temat. Przećwiczymy tworzenie binarnych plików wyjściowych oraz pokażę przykładowe skrypty umożliwiające wczytanie takich plików, zrobienie histogramów, dodawanie histogramów, stawienie jakichś bramek itd. Czyli wszystkiego, co będzie przydatne do zrobienia prostej analizy wyników symulacji.
Do tej pory tworzyliśmy pliki tekstowe, albo wypisywaliśmy jakieś dane na ekran. Jest to dobre rozwiązanie, jeśli chcemy coś „na szybko" podejrzeć. Jadnakże zarówno ze względu na rozmiar pliku, jak i dostępne narzędzia, nie jest to najlepsze rozwiązanie w przypadku prawdziwych, dużych symulacji.
W tych zajęciach będziemy wykorzystywać pliki ROOT. W pierwszym materiale, oprócz instrukcji instalacji GEANTA, podałam wskazówki dotyczące instalacji root-a. Jest to bardzo łatwe. Zazwyczaj wystarczy ściągnąć plik zip, rozpakować i ustawić dobre ścieżki w .bashrc.

Nie trzeba nawet nic kompilować i robić innych trudnych rzeczy. Mam nadzieję, że wszystkim udało się to zrobić.

Wykorzystanie klas ROOT'a

Aby skorzystać z możliwości, jakie daje ROOT, należy przede wszystkim dodać go do listy wykorzystywanych bibliotek w CMakeList.txt, czyli DOPISAĆ następujące linie do tego pliku:

```
# root
find_package(ROOT)
include_directories(${ROOT_INCLUDE_DIRS})
```

oraz przy linkowaniu:

```
target_link_libraries(VANDLESim ${ROOT_LIBRARIES})
```

Następnie można stworzyć klasę (np **Output**), która stworzy plik z drzewem ROOT-a a następnie zapisze do niego wyniki naszej symulacji.

Wykorzystanie klas ROOT'a

Jak taka klasa mogłaby wyglądać?

```
#ifndef Output_H
#define Output_H
#include <string>
#include "TFile.h"
#include "TTree.h"
#include "PMTHit.h"

class Output {
public:
    Output(std::string fileName);
    ~Output();
    void AddHit(PMTHit &hit);
    //alternatywnie
    void AddHit(G4int modIndex, G4double enDep, G4double hitTime);

private:
    TFile* outputFile;
    TTree* outputTree;
    int modIndex;
    double enDep;
    double hitTime;
};
#endif
```

Klasa może być **singletonem**, wtedy trzeba konstruktor uczynić prywatnym i dodać publiczną statyczną metodę zwracającą instancję obiektu.

Wykorzystanie klas ROOT'a cd.

```
#include "Output.hh"
#include <iostream>
Output::Output(std::string filename)
{
    //stworzenie i otwarcie pliku
    outputFile = new TFile(filename.c_str(), "recreate");
    //stworzenie drzewa (może być wiecej niż jedno)
    outputTree = new TTree("outputTree", "outputTree");
    //stworzenie gałęzi i przepisanie im adresów
    outputTree->Branch("modIndex", &modIndex);
    outputTree->Branch("enDep", &enDep);
    outputTree->Branch("hitTime", &hitTime);
}
//w destruktorze zapisujemy i zamykamy plik
//jeśli obiekt jest singletonem warto stworzyć medotę do
//zamykania, aby destruktor pozostał prywatny
Output:: ~Output()
{
    outputFile->Write();
    outputFile->Close();
    delete outputFile;
}
//dodawanie hitów
void Output::AddHit(G4int modIndexV, G4double enDepV, G4double hitTimeV)
{
    modIndex = modIndexV;
    enDep = enDepV;
    hitTime = hitTimeV;
    //wpisz do drzewa!
    outputTree->Fill();
}
```

Wykorzystanie klas ROOT'a cd.

Do podglądania wyników można użyć **TBrowser()**, albo napisać prosty skrypt

```
void PlotTime()
{
    //otwarcie pliku
    TFile* f = new TFile("nazwaPliku");
    TTree* t=(TTree*)f->Get("nazwaDrzewa");
    //stworzenie histogramu (liczby przykładowe!)
    double xMin = 0;
    double xMax = 7000;
    int nBins = 1*(xMax-xMin);
    TH1F* time = new TH1F("time","time",nBins,xMin,xMax);
    //możemy stawiać bramki na innych gałęziach
    TCut cut = "modIndex == 39";

    t->Draw("hitTime>>time",cut);
    gPad->SetLogy();
    time->Draw();
}
```

Zadanie

- Znajdź położenie x i y niezerowych depozytów energii (w całej geometrii, nie tylko wybranej bryle)
- Stwórz drzewo root-owe, które będzie miało następujące liście: położenie x, położenie y, depozyt energii
- Wpisz do drzewa kroku (stepy) o niezerowym depozycie energii. Wpisywanie będzie się odbywało więc w SteppingAction, nie EventAction.
- Narysuj histogram 2D położenia x i y kroku
- Narysuj wykres 3D położenia x i y i depozytu energii w kroku