Zastosowanie pakietu Geant4 w fizyce jądrowej Polecenia sterujące symulacją

Aleksandra Fijałkowska

22 marca 2020

Sterowanie symulacją

Polecenia umieszczone w skrypcie vis.mac są tymi samymi poleceniami, które możemy podać w konsoli geant'a.
Dodatkowo, te same polecenia możemy wpisywać bezpośrednio w kodzie programu, jako argumenty metody `ApplyCommand` klasy `G4UImanager`:

```
UImanager→ApplyCommand("/control/execute commend");
```

Oczywiście wpisywanie komend sterujących symulacją w samym kodzie byłoby szalenie niewygodne. Przy każdej zmianie ustawień musielibyśmy kompilować kod. Sama idea wprowadzenia komend sterujących symulacją ma na celu możliwie duże uproszczenie kodu kompilowanego, przez co zapewnienei elastyczności końcowego programu.

Sterowanie symulacją

Sterowanie wykonaniem symulacji w środowisku GEANT4 można więc zrealizować na trzy sposoby:

1. Za pomocą kodu w programie main. Każda zmiana w sposobie realizacji symulacji wymaga zmiany kodu i jego ponownej kompilacji.
   Opcja niewygodna, wymagająca kompilacji kodu po każdej zmianie.
2. Za pomocą komend wpisywanych w oknie UI (jak na przykład wysłanie cząstek w załączonym filmie).
   Przydatna opcja na etapie projaktowania symulacji, robienia małych testów itd. Jednakże doprowadzenie programu do realizowania pożądanej symulacji z ładnym wyświetlaniem wymaga wpisania kilkunastu komend, które przy tym rozwiązaniu należy mieć w pamięci. Warto posiłkować się więc dodatkowym skryptem.
3. Za pomocą skryptów. Uzgodniliśmy, że będziemy używać dwóch typów skryptów:
   - 3.1 skrypt "domyślny" - "vis.mac", który jest wczytywany za każdym razem, gdy użytkownik nie poda argumentu wejściowego. Skrypt będzie zawierał podstawowe komendy dotyczące wyświetlania.
   - 3.2 skrypt dostosowany do konkretnej symulacji, którego nazwa będzie podawana podczas uruchomienia programu. To rozwiązanie jest stosowane najczęściej, gdy nie chcemy wyświetlać przebiegu symulacji, zależy nam na jak największej wydajności i szybkości działania programu.

Realizacja trzeciej strategii

1. Stworzenie instancji User Interface Manager:

```
G4UImanager* UImanager = G4UImanager::GetUIpointer();
```

2. Stworzenie i zainicjowanie instancji klasy obsługującej wizualizację:

```
G4VisManager* visManager = new G4VisExecutive;
visManager->Initialize();
```

3. Stworzenie instancji klasy obsługującej interfejs zgodny ze zdefiniowaną zmienną środowiskową (np G4UI_USE_QT):

```
G4UIExecutive* ui = new G4UIExecutive(argc, argv);
```

4. Sprawdzenie liczby podanych argumentów wejściowych, w przypadku braku argumentu uruchomienie skryptu "vis.mac", zaś w przeciwnym sczytanie nazwy skryptu i uruchomienie go:

```
if(argc == 1)
    UImanager->ApplyCommand("/control/execute ../vis.mac");
else
{
    G4String filename = argv[1];
    UImanager->ApplyCommand("/control/execute " + filename);
}
```

5. Rozpoczęcie sesji ui:

```
ui->SessionStart();
```

Przykładowy plik vis.mac

```
/vis/open OGL 600x600-0+0
/vis/drawVolume
/vis/scene/add/trajectories smooth
/vis/scene/endOfEventAction accumulate
/vis/modeling/trajectories/create/drawByCharge
/vis/modeling/trajectories/drawByCharge-0/default/setDrawStepPts true
/vis/modeling/trajectories/drawByCharge-0/default/setStepPtsSize 2
/vis/scene/add/axes 0 0 0 2.0 m
```

użyj openGL do wyświetlania
narysuj bryły zdefiniowane w kodzie
rysuj tory cząstek
wyświetlaj wyniki wielu zdarzeń (Event) na jednym obrazku
nadaj torom cząstek różne kolory w zależności od ich ładunku
rysuj punkty interakcji (końce kroków) w postaci punktów
określa rozmiar punktu oznaczającego koniec kroku
dodaj osie w punkcie (0,0,0) i długości 2 m

Podstawowe polecenia sterujące symulacją

```
/control/   UI control commands.
/units/   Available units.
/analysis/   ...Title not available...
/process/   Process Table control commands.
/particle/   Particle control commands.
/geometry/   Geometry control commands.
/tracking/   TrackingManager and SteppingManager control commands.
/event/   EventManager control commands.
/cuts/   Commands for G4VUserPhysicsList.
/run/   Run control commands.
/random/   Random number status control commands.
/gun/   Particle Gun control commands.
/material/   Commands for materials
/vis/   Visualization commands.
/gui/   UI interactors commands.
```

Najpopularniejsze przykłady

```
/run/beamOn <nrOfEvents>
#wysyła nrOfEvents cząstek pierwotnych

/gun/particle <particleName>
#Określa rodzaj cząstki pierwotnej (gamma, neutron, e-, e+ itd.)

/gun/energy <Energy> <Unit>
#Ustala energię cząstki pierwotnej

/gun/position <X> <Y> <Z> <Unit>
#Określa pozycję, z której są wysyłane cząstki

/gun/direction <ex> <ey> <ez>
#Określa kierunek cząstek pierwotnych

/vis/viewer/zoom <multiplier>
#Dziś już zapomniana komenda do zwiększania obrazu wizualizacji.
#Teraz wystarczy użyć kółka myszki.
```

Przykład użycia:

```
/run/beamOn 100
/gun/particle neutron
/gun/energy 0.5 MeV
/gun/direction 1 0.5 2
#wektor direction nie musi być znormalizowany
```