

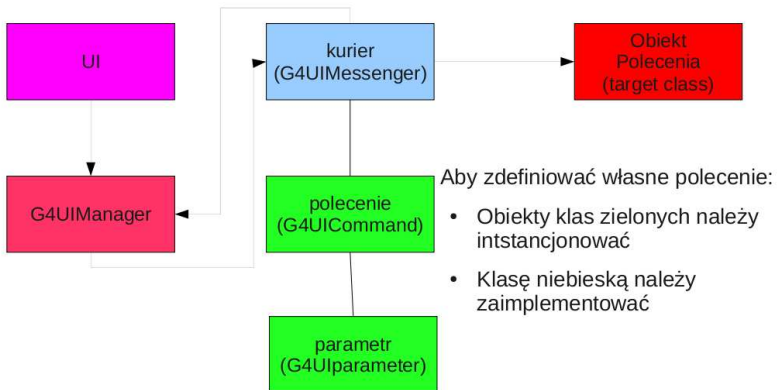
Środowisko programistyczne GEANT4

Leszek Adamczyk

Wydział Fizyki i Informatyki Stosowanej
Akademia Górniczo-Hutnicza

Wykłady w semestrze zimowym 2013/2014

Polecenia interfejsu użytkownika



G4UImessenger

- Klasa bazowa **G4UImessenger** służy do wyprowadzania własnych klas definiujących polecenia UI.
- Obiekt takiej klasy powinien być utworzony (zniszczony) przez konstruktora (destruktor) obiektu na który działa dane polecenie (**target class**);
- Metody klasy **G4UImessenger**:
 - **Constructor** - definicja (+instancja) poleceń UI;
 - **Destructor** - unicestwienie poleceń UI;
 - **SetNewValue(G4UICommand *, G4String NewValue)** konwersja "NewValue" na odpowiedni typ wartości oraz wywołanie odpowiedniej funkcji klasy **target**
 - **GetCurrentValue(G4UICommand *)** wywołuje odpowiednią funkcję klasy **target** i konwertuje wartość parametru na typ G4String

G4UImessenger- Przykład tworzenia polecenia

- Implementacja polecenia w konstruktorze klasy wyprowadzonej z **G4UImessenger**.
- Przykład: Załóżmy, że chcemy z poziomu UI zmieniać grubość tarczy berylowej.
- Wyprowadzamy klasę **MyDetectorMessenger** z klasy bazowej **G4UImessenger**

```
MyDetectorMessenger::MyDetectorMessenger
(MyDetectorConstruction* detector) :fTarget(detector)

{
    detDir = new G4UIDirectory("/My/");
    detDir->SetGuidance("My detector setup commands.");
    fBerCmd = new
        G4UICmdWithADoubleAndUnit("/My/beryllium", this);
    fBerCmd->SetGuidance("Beryllium target.");
    fBerCmd->SetParameterName("depth", true);
    fBerCmd->SetRange("depth>0. && depth< 40.");
    fBerCmd->SetDefaultValue(15);
    fBerCmd->SetDefaultUnit("cm"); }
}
```

G4UICommand

- Klasa **G4UICommand** reprezentuje polecenia UI a **G4UIParameter** reprezentuje parametry poleceń.
- GEANT ma szereg gotowych klas wyprowadzonych z klasy **G4UICommand** w zależności od typu parametrów z nią stowarzyszonych:
 - **G4UICmdWithoutParameter**
 - **G4UICmdWithAString**
 - **G4UICmdWithABool**
 - **G4UICmdWithAnInteger**
 - **G4UICmdWithADouble**, **G4UICmdWithADoubleAndUnit**
 - **G4UICmdWith3Vector**, **G4UICmdWith3VectorAndUnit**
 - **G4UIDirectory**
- Polecenia z innymi typami parametrów należy samemu wyprowadzić z klasy bazowej **G4UICommand**

G4UICommand - Nazwa parametru

- Wszystkie klasy typu **G4UICmdWithAxxxx** posiadające parametr mają zaimplementowaną metodę:

```
SetParameterName(  
    const char* name,  
    G4bool par2,  
    G4bool par3=false)
```

- Nazwa parametru używana jest w helpie oraz w definicji zakresu parametrów;
- Jeśli `par2=true` to polecenie można wydać nie określając wartości parametru, użyta zostanie wartość domyślna;
- Jeśli `par3=true` to aktualna wartość parametru jest przyjęta jako domyślna, w przeciwnym przypadku wartość domyślną należy określić korzystając z **SetDefaultValue**

G4UICommand - zakres, jednostka parametru

- `void SetRange (const char* range)`
dostępna dla poleceń z parametrami numerycznymi.
Warunki zapisujemy za pomocą składni c++
`cmd->SetRange("x>0. && x>y && x+y<z")`
- `void SetDefaultUnit (const char* unit)`
funkcja dostępna dla poleceń z określoną jednostką
Po zdefiniowaniu jednostki polecenie nie zostanie zaakceptowane z jednostką o innym wymiarze.

G4UICommand - konwersje String-wartość numeryczna

Klasy wyprowadzone z klasy **G4UICommand** z numerycznymi lub logicznymi parametrami mają wbudowane funkcje do konwersji:

- String – wartość numeryczna:
 - `G4bool` `GetNewBoolValue (const char*)`
 - `G4int` `GetNewIntValue (const char*)`
 - `G4double` `GetNewDoubleValue(const char*)`
 - `G4ThreeVector` `GetNew3VectorValue(const char*)`
 - Należy ich używać w funkcji `SetNewValue()`
 - Jednostka uwzględniana jest automatycznie
- Wartość numeryczna - String
 - `G4String` `ConvertToString(.....)`
 - - `G4String` `ConvertToString(....., const char* unit)`
 - Należy ich używać w funkcji `GetCurrentValue()`

G4UICommand - Przykład

```
void MyDetectorMessenger::SetNewValue
    (G4UIcommand* command, G4String newValue)
{
    if (command == fBerCmd)
        fTarget->SetBerylliumDepth
            (fBerCmd->GetNewDoubleValue(newValue));
}
```

```
G4String MyDetectorMessenger::GetCurrentValue
    (G4UIcommand* command)
{
    if (command == fBerCmd)
        return fBerCmd->ConvertToString
            (fTarget->GetBerylliumDepth(), "cm");
}
```

Błędy implementacji geometrii

- Obszar fizyczny nie jest całkowicie zawarty w obszarze w którym został umieszczony;
- Dwa obszary fizyczne umieszczone we wspólnym obszarze mają części wspólne;
- GEANT nie wykrywa tego typu błędów a jego działanie w ich obecności jest nieprzewidywalne;
- Istnieją narzędzia pomocne w wykrywaniu niepoprawnej implementacji geometrii:
 - opcjonalny test wykonywane podczas konstrukcji obszaru fizycznego;
 - testy wykonywane za pomocą poleceń UI
 - narzędzie graficzne (DAVID)

Test podczas tworzenia obszaru fizycznego

- Konstruktory obiektów klas **G4VPVPlacement** oraz **G4VPVParameterised** posiadają opcjonalny argument sterujący wykonaniem testu:

```
G4VPVPlacement( G4RotationMatrix*  Rot,  
                const G4ThreeVector&  trans,  
                G4LogicalVolume*      CurrentLogical,  
                const G4String&        Name,  
                G4LogicalVolume*      MotherLogical,  
                G4bool                 Many,  
                G4int                  CopyNo,  
                G4bool                 SurfChk=false )
```

- Jeśli ostatni argument ma wartość logiczną **true** to podczas tworzenia obszaru fizycznego wykonywany jest test polegający na losowej generacji punktów na powierzchni obszaru i sprawdzeniu czy leżą one wewnątrz obszaru matki i nie należą do innych obszarów fizycznych;
- Test zajmuje bardzo dużo czasu.

Testy z poziomu UI

- Polecenia UI służące do weryfikacji geometrii znajdują się w katalogu [/geometry/test](#)
- Test polega na zdefiniowaniu krzywej (prosta lub okrąg) wzdłuż której GEANT transportuje wirtualną cząstkę i informuje o błędzie gdy:
 - cząstka wchodzi do jednego obszaru przed opuszczeniem poprzedniego a oba obszary są umieszczone we wspólnym obszarze;
 - cząstka poruszająca się w jakimś obszarze opuszcza jej obszar "matkę"
- Możliwe jest zdefiniowanie całej sieci krzywych wzdłuż których przeprowadzany jest test.

Zewnętrzne narzędzie graficzne - DAVID

- **DAVID** jest narzędziem graficznym wykorzystującym opis geometrii detektora zawarty w pliku zapisanym przez interfejs do wizualizacji w formacie **DAWNFILE**;
- Należy mieć zainstalowany program DAWN. Oba programy: DAVID i DAWN można ściągnąć z <http://geant4.kek.jp/~tanaka>
- DAVID wizualizuje na czerwono przecinające się obszary detektora (zapisuje również raport tekstowy)

