IEEE Nuclear Science Symposium and Medical Imaging Conference

Short Course

Simulation Techniques Using Geant4

Maria Grazia Pia (*INFN Genova, Italy)*
MariaGrazia.Pia@ge.infn.it

Dresden, 18 October 2008

http://www.ge.infn.it/geant4/events/nss2008/geant4course.html

This course exploits training material developed by several Geant4 Collaboration members: thanks to all of them!

User Interface

Basic concepts

Steering the simulation

A Geant4 simulation can be **steered in three ways**:

- everything hard-coded in the C++ source *(also the number of events to be shot)*
  - You need to re-compile for any change (not very smart!)
- batch session
  - via a ASCII macro
- commands captured from an interactive session

Steering the simulation

- Setting up batch mode (namely, read commands from a macro file) in the `main()`

```
G4UImanager* UI = G4UImanager::GetUIpointer();
G4String command = "/control/execute";
G4String fileName = argv[1];
UI->applyCommand(command+fileName);
```

takes the first argument after the executable as the macro name and runs it

- Your executable can be run as
  - `myExecutable mymacro.mac`
- To execute a macro interactively:

```
/control/execute mymacro.mac
```

Steering the simulation

- Setting up **interactive mode** is easy – but there are many choices of interface
  - All of them must be derived from the abstract class **`G4UIsession`**
  - Geant4 provides several implementations
- In the **`main()`**, according to the computer environments, construct a **`G4UIsession`** concrete class provided by Geant4 and invoke its **`SessionStart()`** method

Select (G)UI

- Geant4 provides several interfaces for various (G)UI:
  - **`G4UIterminal`**: C-shell like character terminal
  - **`G4UItcsh`**: tcsh-like character terminal with command completion, history, etc
  - **`G4UIGAG`**: Java based graphic UI (GUI)
  - **`G4UIXm`**: Motif-based GUI, command completion

- Define and invoke them like `G4UIterminal`

```
session = new G4UIGAG();
session->StartSession();
```

- Note for **`G4UItcsh`**, it must be defined as

```
session = new G4UIterminal (new G4UItcsh);
```

Environmental variables

- Users can select and plug in (G)UI by setting *environmental variables* before compilation, similar to what seen for visualization drivers
  - `setenv G4UI_USE_GUINAME`
- Example:
  - `setenv G4UI_USE_TERMINAL 1` (default)
  - `setenv G4UI_USE_GAG 1`
  - `setenv G4UI_USE_XM 1`

User interface choices

- `G4UIterminal` – C-shell-like character terminal
  - runs on all Geant4-supported platforms
- `G4UItcsh` – tcsh-like character terminal with command completion, history, etc.
  - runs only on Solaris and Linux
- `G4UIXm, G4UIXaw, G4UIXWin32` – G4UIterminal implemented over Motif, Athena and WIN32 libraries
  - runs on Unix/linux and Windows, respectively
- `G4UIGAG` – Java-based GUI
  - runs on all Geant4 platforms

Useful GUI Tools Released by Geant4 Developers

- GGE: Geometry editor based on Java GUI
  - http://erpc1.naruto-u.ac.jp/~geant4
- GPE: Physics editor based on Java GUI
  - http://erpc1.naruto-u.ac.jp/~geant4

Build-it user commands

- Geant4 provides a number of general-purpose user interface commands which can be used:
  - interactively via a (G)UI
    ```
    Idle> /run/setCut [value] [unit]
    ```
  - in a macro file
  - within C++ code using the `ApplyCommand()` method of `G4UImanager`
    ```
    G4UImanager::GetUIpointer()
    ->ApplyCommand("/run/setCut 1 cm");
    ```
- A complete list of built-in commands is available in the Geant4 Application Developers Guide, Chapter 7.1

User-defined commands

- If built-in commands are not enough, you can make your own (e.g. change at run-time parameters of primary generator, etc.)

- Geant4 provides several command classes, all derived from **`G4UIcommand`**, according to the type of argument they take
  - **`G4UIcmdWithoutParameter`**
  - **`G4UIcmdWithABool`**
  - **`G4UIcmdWithADouble`**
  - **`G4UIcmdWithADoubleAndUnit`**
  - **`...`**

User-defined commands

- Commands have to be defined in messenger classes, that **inherit from `G4UImessenger`**
- Define the command in the constructor:

```
G4UIcmdWithADoubleAndUnit* fThetaCmd =
  new G4UIcmdWithADoubleAndUnit
   ("/prim/angle",this);
```

Command taking as argument a double and a unit, called `/prim/angle`

```
fThetaCmd->SetGuidance(“Opening angle of the source");
fThetaCmd->SetDefaultUnit("deg");
fThetaCmd->SetUnitCandidates(“deg rad”);
```

Sets guidance, default unit, etc.

- Delete the command in the destructor

User-defined commands

- Define the action of the command in the SetNewValue() method of the messenger:

```
void MyMessenger::SetNewValue
  (G4UIcommand* cmd,G4String string)
{
 if (cmd == fThetaCmd)
 {
   G4double value = fThetaCmd
    ->GetNewDoubleValue(string);

   ...->DoSomething(value);
 }
}
```

Retrieve a G4double value from the **(string)** argument given to the command

Use the value in the way it is needed (e.g. pass it to other classes: opening angle for primary generator)