Quick Intro to

UI & Messenger in Geant 4 - Basics

Karim Laihem

Geant4 Training event – Calorimetry in HEP

DESY Zeuthen 10 -13 May 2011

Refrence talk: Makoto Asai (SLAC)

Contents

- Command syntax
- Macro file
- G4UIterminal
- Defining basic UI command

Geant4 UI command

- A command consists of
  - Command
  - Command
  - Parameter(s)

```
/run/verbose 1
/vis/viewer/flush
```

- A parameter can be a type of string, boolean, integer or double.
  - Space is a delimiter.
  - Use double-quotes (“”) for string with space(s).
- A parameter may be “omittable”. If it is the case, a default value will be taken if you omit the parameter.
  - Default value is either predefined default value or current value according to its definition.
  - If you want to use the default value for your first parameter while you want to set your second parameter, use “!” as a place holder.

```
/dir/command ! second
```

Command submission

- Geant4 UI command can be issued by
  - (G)UI interactive command submission
  - Macro file
  - Hard-coded implementation
    - Should not be used inside an event loop

```
G4UImanager* UI = G4UImanager::GetUIpointer();
UI->ApplyCommand("/run/verbose 1");
```

- The availability of individual command, the ranges of parameters, the available candidates on individual command parameter may vary according to the implementation of your application and may even vary dynamically during the execution of your job.
- some commands are available only for limited Geant4 application state(s).
  - E.g. **`/run/beamOn`** is available only for *Idle*

Command refusal

- Command will be refused in case of
  - Wrong application state
  - Wrong type of parameter
  - Insufficient number of parameters
  - Parameter out of its range
    - For integer or double type parameter
  - Parameter out of its candidate list
    - For string type parameter
  - Command not found

Macro file

- Macro file is an ASCII file contains UI commands.
- All commands must be given with their full-path directories.
- Use "#" for comment line.
  - First "#" to the end of the line will be ignored.
  - Comment lines will be echoed if **`/control/verbose`** is set to 2.
- Macro file can be executed
  - interactively or in (other) macro file

    `/control/execute` ***`file_name`***
  - hard-coded

```
G4UImanager* UI = G4UImanager::GetUIpointer();
UI->ApplyCommand("/control/execute file_name");
```

Batch mode / interactive mode

- In your *main()*

```
int main(int argc, char** argv)
{
      …
  if (argc != 1)
   { // batch mode
     G4String command = "/control/execute ";
     G4String fileName = argv[1];
     UImanager->ApplyCommand(command+fileName);
   }
  else
   {  // interactive mode : define UI session
     G4UIExecutive* ui = new G4UIExecutive(argc, argv);
     ui->SessionStart();
     delete ui;
   }
```

Terminal commands

- Interactive terminal supports some Unix-like commands for directory.
  - **`cd`**, - change and display current command directory
    - By setting the current command directory, you may omit (part of) directory string.
  - **`ls`** - list available UI commands and sub-directories
- It also supports some other commands.
  - **`history`** - show previous commands
  - **`!`***`historyID`* - re-issue previous command
  - arrow keys and tab (TC-shell only)
  - **`?`***`UIcommand`* - show current parameter values of the command
  - **`help`** [*`UIcommand`*] - help
  - **`exit`** - job termination
- Above commands are interpreted in the interactive terminal and are not passed to Geant4 kernel. You cannot use them in a macro file.

Definition (instantiation) of a command

- To be implemented in the constructor of a messenger class.

```
A01DetectorConstMessenger::A01DetectorConstMessenger
(A01DetectorConstruction* tgt)
:target(tgt)
{
  mydetDir = new G4UIdirectory("/MyCalorimeter/det/");
  mydetDir->SetGuidance("Calorimeter geometry.");

  armCmd = new G4UIcmdWithADoubleAndUnit("/mydet/armAngle",this);
  armCmd->SetGuidance("Rotation angle of the second arm.");
  armCmd->SetParameterName("angle",true);
  armCmd->SetRange("angle>=0. && angle<180.");
  armCmd->SetDefaultValue(30.);
  armCmd->SetDefaultUnit("deg");
}
```

- Guidance can (should) be more than one lines. The first line is utilized as a short description of the command.

Parameter name(s)

- These methods are available for derivative command classes which take parameter(s).

```
void SetParameterName(
     const char*parName,
     G4bool omittable,
     G4bool currentAsDefault=false);
```

- Parameter names are used in help, and also in the definition of parameter range.
- If "omittable" is true, the command can be issued without this particular parameter, and the default value will be used.
- If "currentAsDefalult" is true, current value of the parameter is used as a default value, otherwise default value must be defined with SetDefaultValue() method.

Range, unit and candidates

```
void SetRange(const char* rangeString)
```

- Available for a command with numeric-type parameters.
- Range of parameter(s) must be given in C++ syntax.

```
aCmd->SetRange("x>0. && y>z && z>(x+y)");
```

- Not only comparison with hard-coded number but also comparison between variables and simple calculation are available.
- Names of variables must be defined by SetParameterName() method.

```
void SetDefaultUnit(const char* defUnit)
```

- Available for a command which takes unit.
- Once the default unit is defined, no other unit of different dimension will be accepted.
- Alternatively, you can define a dimension (unit category) without setting a default unit.

```
void SetUnitCategory(const char* unitCategory)
```

```
void SetCandidates(const char* candidateList)
```

- Available for a command with string type parameter
- Candidates must be delimited by a space.
- Candidates can be dynamically updated.

G4UIcommand and its derivatives

- G4UIcommand is a class which represent a UI command. G4UIparameter represents a parameter.
- G4UIcommand can be directly used for a UI command. Geant4 provides its derivatives according to the types of associating parameters. These derivative command classes already have necessary parameter class object(s), thus you don't have to instantiate G4UIparameter object(s).
  - G4UIcmdWithoutParameter
  - G4UIcmdWithAString
  - G4UIcmdWithABool
  - G4UIcmdWithAnInteger
  - G4UIcmdWithADouble, G4UIcmdWithADoubleAndUnit
  - G4UIcmdWith3Vector, G4UIcmdWith3VectorAndUnit
  - G4UIdirectory
- A UI command with other type of parameters must be defined by G4UIcommand base class with G4UIparameter.

Converting between string and values

- Derivatives of G4UIcommand with numeric and boolean parameters have corresponding conversion methods.
- From a string to value

```
G4bool GetNewBoolValue(const char*)
G4int GetNewIntValue(const char*)
G4double GetNewDoubleValue(const char*)
G4ThreeVector GetNew3VectorValue(const char*)
```

  - To be used in SetNewValue() method in messenger.
  - Unit is taken into account automatically.
- From value to string

```
G4String ConvertToString(...)
G4String ConvertToString(...,const char* unit)
```

  - To be used in GetCurrentValue() method in messenger.

SetNewValue and GetCurrentValue

```
void DetectorMessenger
::SetNewValue(G4UIcommand* command,G4String newValue)
{
  if( command==armCmd )
  { target->SetArmAngle(armCmd->GetNewDoubleValue(newValue)); }
}

G4String DetectorMessenger
::GetCurrentValue(G4UIcommand* command)
{
  G4String cv;
  if( command==armCmd )
  { cv = armCmd->ConvertToString(target->GetArmAngle(),"deg"); }
  return cv;
}
```

```
// UI: Task 1
// In DetectorConstruction.cc
//---------------------------------------------------
// Create a new function to set the thickness of the scintillator layer
//---------------------------------------------------
// Step 1: Create a new "void" function called "setScintLayerThickness" with a G4double variable "ScintThikness". (See the above function "setPbLayerThickness" for guidance)
// Step 2: Assign this variable "ScintThikness" to "ScintillatorZ". The variable "ScintillatorZ" is the thickness of the scintillator layer already defined in "Geometry Task 6". Remark: "ScintillatorZ" sould be globaly declared in DetectorConstruct.hh -> "private" field.
// Remark: To be on the safe side force the new variable to be positive and non-zero, and display a message "********** The thikness of the Scintillator layer should be >0" in a case that the new value is equal to zero or negative.(See the above function "setPbLayerThickness" for guidance)
// Step 3: Declare this new function "setScintLayerThickness" in the header file "DetectorConstruction.hh" as public. (See previous declarations there)
// Step 4: Compile your code (gmake)
// Step 5: Now go to the "UI Task 2" which is in the file "DetectorMessenger.cc" to create a new command in the UI to have the option to change the scintillator thickness interactively via UI commands.
//------------------------------

void DetectorConstruction::setScintLayerThickness(G4double ScintThikness)
{
  if(ScintThikness>0.)
    {
      ScintillatorWidthZ = ScintThikness;
    }
  else
    {
      G4cout<<"********** The thickness of the Scintillator layer should be >0"<<G4endl;
    }
}
```