

UI Interface

Koichi Murakami

KEK/CRC

V. Ivanchenko adaptation to CERN Geant4 tutorial

15-19 February 2010



How to use UI session (CLI, GUI, batch)

UI SESSION

UI SESSION

Interactive Front Ends

G4Uterminal

- CLI (command-line interface)
- runs on all Geant4-supported platforms
- G4Uitcsh available
 - ✓ alternative shell of G4Uterminal
 - ✓ tcsh-like read-line
 - ✓ command completion, history (across sessions), etc.

G4UIQt, G4UIXm, G4UIXaw, G4UIXWin32

- GUI (graphical user interface)
- G4Uterminal implemented over Qt, Motif, Athena and WIN32 widgets

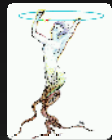
G4UIGAG

- interface with GAG/MOMO, Java-based GUI interface
- runs on all Geant4-supported platforms

Geant4 (User) Interface and Applications

Analysis Tools

- AIDA Interface
- JAS3
- ROOT
- Python binding
- PAIDA
- ROOT-Python



GUI Tools

- MOMO; Java-based tools
- for editing Geometry/Physics List
- interactive session (*GAG*)
- OpenScientist; interactive environment



Pythonized Applications

- Dynamic configuration of user app.-s
- GUIs / web app.-s

User Applications (C++)

Batch

macro script

Terminal Front End

simple command-line
tcsh-like shell

GUI

Qt
Motif
Java (*GAG*)

MPI Session

Interfaces

Python as software component bus



Python Front End

```
>>> import Geant4
```

Python Interface

C++ classes are directly bridged.

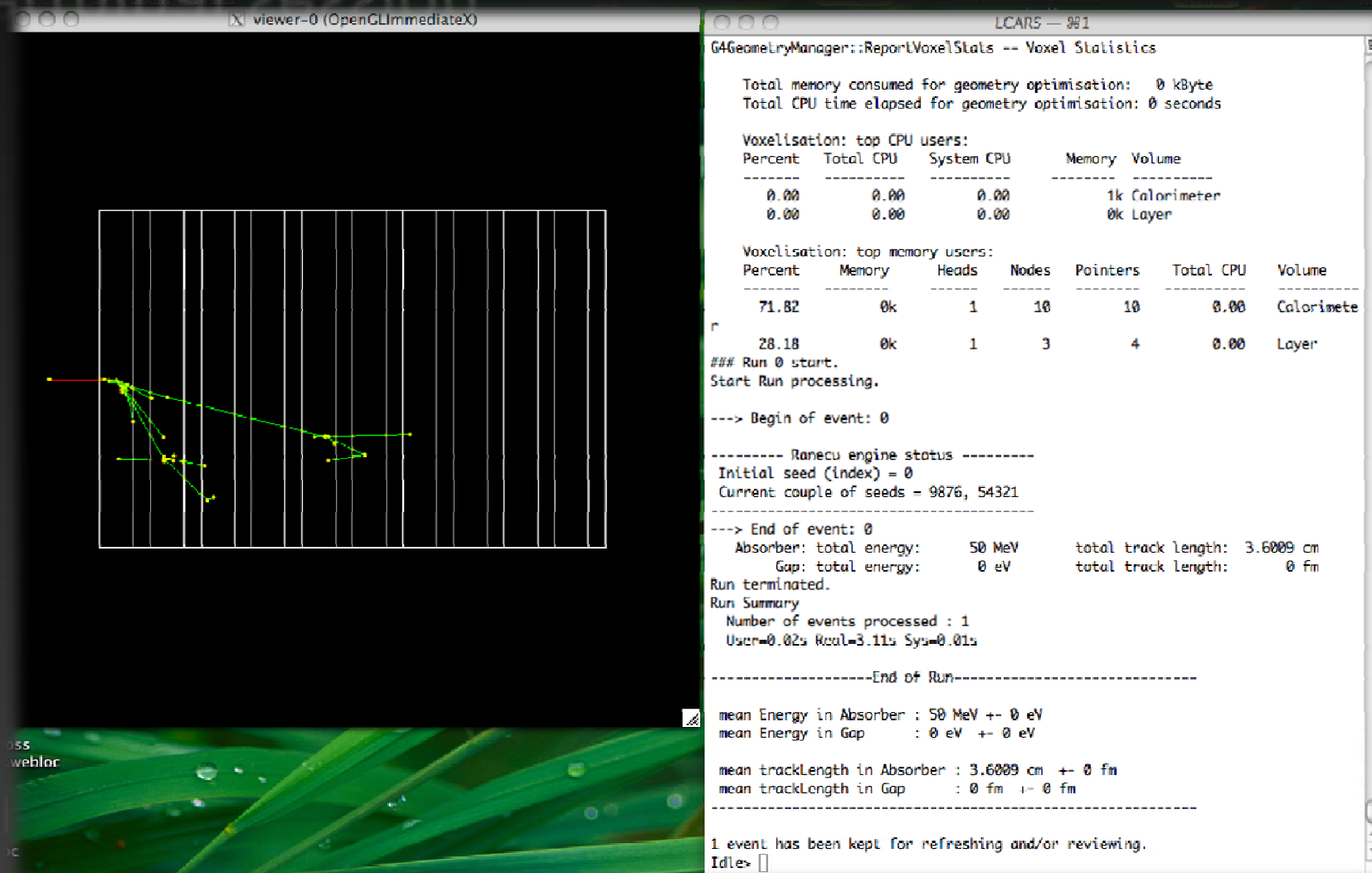
UI command

Intercoms

Geant4

Geant 4

Terminal session



The image shows a terminal window with two panes. The left pane displays a 3D visualization of a particle track, showing a series of yellow and green points connected by lines, set against a background of vertical white lines. The right pane shows the terminal output of a Geant4 simulation, including voxel statistics, event processing details, and a run summary.

```

G4GeometryManager::ReportVoxelStats -- Voxel Statistics

Total memory consumed for geometry optimisation: 0 kByte
Total CPU time elapsed for geometry optimisation: 0 seconds

Voxelisation: top CPU users:
Percent   Total CPU   System CPU   Memory   Volume
-----
0.00      0.00        0.00        0k       1k Calorimeter
0.00      0.00        0.00        0k       0k Layer

Voxelisation: top memory users:
Percent   Memory     Heads   Nodes   Pointers   Total CPU   Volume
-----
71.82     0k         1       10      10         0.00       Calorimete
28.18     0k         1       3       4          0.00       Layer

### Run 0 start.
Start Run processing.

---> Begin of event: 0

----- Ranecu engine status -----
Initial seed (index) = 0
Current couple of seeds = 9876, 54321

---> End of event: 0
Absorber: total energy: 50 MeV      total track length: 3.6009 cm
Gap: total energy: 0 eV           total track length: 0 fm
Run terminated.
Run Summary
Number of events processed : 1
User=0.02s Real=3.11s Sys=0.01s

-----End of Run-----

mean Energy in Absorber : 50 MeV +- 0 eV
mean Energy in Gap      : 0 eV +- 0 eV

mean trackLength in Absorber : 3.6009 cm +- 0 fm
mean trackLength in Gap      : 0 fm +- 0 fm

-----

1 event has been kept for refreshing and/or reviewing.
Idle>

```

Qt interface (used for current tutorial)

The screenshot shows a Linux desktop with a green grass background. Two windows are open:

- G4UI Session:** A terminal window showing the output of a Geant4 simulation. It includes event statistics for an absorber and a gap, a run summary, and a command prompt with the command `/run/beamOn 10`.
- viewer-D (OpenGLImmediateQt):** A 3D visualization window showing a wireframe rectangular volume. Inside, numerous green lines represent particle tracks originating from a central point and spreading outwards.

At the bottom right of the desktop, there is a terminal window with the following text:

```
Xlib: extension "Generic Event Extension" missing on display "localhost:10.0".  
Xlib: extension "Generic Event Extension" missing on display "localhost:10.0".  
Xlib: extension "Generic Event Extension" missing on display "localhost:10.0".  
Xlib: extension "Generic Event Extension" missing on display "localhost:10.0".  
  
----- Ranecu engine status -----  
Initial seed (index) = 0  
Current couple of seeds = 9876, 54321  
-----
```

MOMO



MOMO (environments/MOMO/)

Env. Panel

GGE

GPE

GAG/Gain

Panel for setting environment variables

Geometry Editor

Physics List Editor

Momomake.gmk

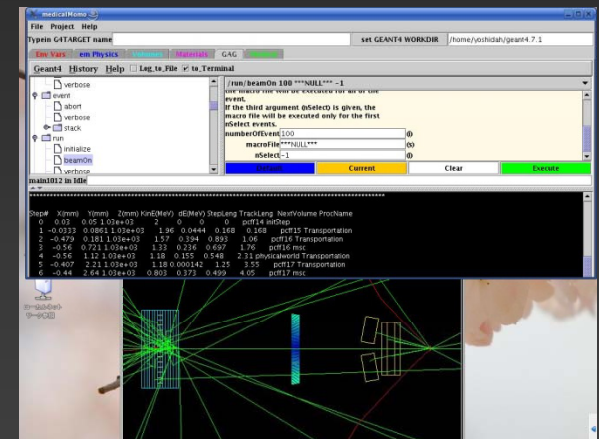
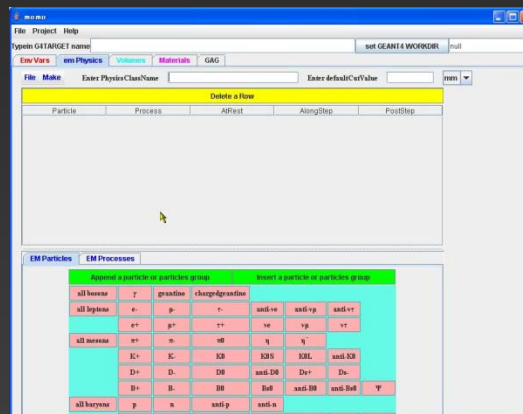
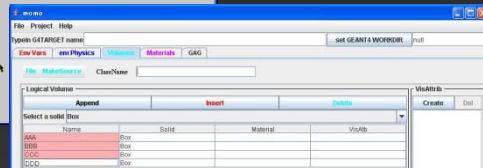
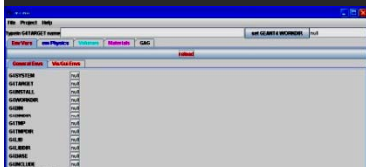
Geometry.cc
Geometry.o

PhysicsList.cc
PhysicsList.o

MySimulation.cc, *.cc
MySimulation.o, *.o

compile them!

User Applications w/ GAG interface



Notes on building/using sessions

To build/use G4UIQt, G4UIXm, G4UIXaw, or G4UIXWin32, extra environment variables are necessary;

G4UI_BUILD_QT_SESSION / G4UI_USE_QT

G4UI_BUILD_XM_SESSION / G4UI_USE_XM

G4UI_BUILD_XAW_SESSION / G4UI_USE_XAW

G4UI_BUILD_WIN32_SESSION / G4UI_USE_WIN32

None required to use G4Uiterminal, G4Uitcsh, G4UIGAG

- these sessions do not need external libraries, so they are automatically built and linked.

Practical Usage (conventional way)

In your main(),

```
#include "G4UIxxx.hh"  
// xxx = terminal, Qt, Xm, Xaw, Win32, GAG  
#include "G4UITcsh.hh" // if use a 'tcsh' module  
  
G4UISession* session = new G4UIxxx;  
session->SessionStart(); // main loop starts..  
delete session;
```

For a tcsh-like session, a session is instantiated by

```
G4UISession* session =  
    new G4UITerminal(new G4UITcsh);
```

Practical Usage (G4UIExecutive)

G4UIExecutive is available in 9.3 release.

- convenient class for selecting a UI session according to environment variables, G4UI_USE_XXX.
 - ✓ TCSH, XM, WIN32, QT, Uiterminal (default)
 - ✓ select a session type by the order above
- *Pros*: just one line cares the selection of a session type
- *Cons*: Environment variables might cause unexpected behaviors.

In your main(),

```
#include "G4UIExecutive.hh"
```

```
G4UIExecutive* ui = new G4UIExecutive(argc, argv);  
ui->SessionStart();  
delete ui;
```

More practical implementation, see main() in novice examples.

G4Uterminal (command line interface)

Geant4 can be driven by a series of commands, which are tidied up in categorized directory structure.

G4Uterminal supports some Unix-like commands for directory.

- **cd, pwd** : change and display current command directory
- **ls / lc** : list available UI commands and sub-directories in the current directory

also supports some built-in commands.

- **history** : show previous commands
- **!historyID** : re-issue previous command
- **arrow keys** : scroll command history (TC-shell only)
- **?command** : show current parameter values of the command
- **help [UIcommand]** : show command help
- **exit** : terminate the session

Notes: These commands are not passed to the Geant4 kernel, so that you cannot use them in a macro file!

Batch Mode (not used in current tutorial)

A Geant4 simulation can be executed in a batch mode.

- A macro file consists of a series of UI commands
- A macro file can be specified as an argument.

```
$ task2a myrun.mac >& myrun.log (csh)
```

```
# task2a myrun.mac > myrun.log 2>&1 (bash)
```

To enable batch mode,

- In your main(),

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

```
G4String command = “/control/execute ”;
```

```
G4String fileName = argv[1];
```

```
UI-> applyCommand(command+fileName);
```

What is UI commands

UI COMMAND

UI COMMAND

Geant4 UI command

A G4UIcommand consists of

- Command directory
- Command
- Parameter(s)

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

A parameter can be a type of

- *string, bool, integer or double*
- Space is a delimiter
- Use double-quotes (“”) for string with space(s).

A parameter may be **omittable**.

- 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.
- A default value can be specified by *“!”*;
`/dir/command ! 123`

Command Execution

Geant4 UI commands can be executed

- in a UI session / terminal
- in a macro file
- by hard-coded implementation
 - ✓ a way to bypass a class pointer (C++ access)
 - ✓ Using inside an event loop is NOT recommended!!

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

Some commands are available only for limited Geant4 application states

- Geant4 is a state machine.
- E.g. /run/beamOn is *available only for Idle states*.

Built-in Commands

There are built-in commands roughly organized according to Geant4 categories.

```
Idle> ls
```

```
Command directory path : /
```

```
Sub-directories :
```

```
  /control/   UI control commands.  
  /units/     Available units.  
  /geometry/  Geometry control commands.  
  /tracking/  TrackingManager and SteppingManager control commands.  
  /event/     EventManager control commands.  
  /run/       Run control commands.  
  /random/    Random number status control commands.  
  /particle/  Particle control commands.  
  /process/   Process Table control commands.  
  /material/  Commands for Materials  
  /vis/       Visualization commands.  
  /gun/       Particle Gun control commands.
```

```
Commands :
```


UI Macro

Code reviewed in 9.1 release.

A macro file contains **a series of UI commands**

- one command in each line
- All commands must be given in *their full-paths*.
- White spaces at the head of a line are allowed.
- continued line by `\` or `'_'`

`"#"` is used for **a comment line**.

- First `"#"` to the end of the line will be ignored.
- Comment lines will be echoed if `/control/verbose` is set to 2.

A macro file can be executed

- *interactively or in another macro file*
`/control/execute file_name`
- From c++ code (**NOT recommended – use in exceptional case**)

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

Alias: *Scripting with UI commands*

Alias can be defined by

- `/control/alias [name] [value]`
- Aliased value is always treated as a string.
- Use with curly brackets, `{and}`.

Example:

```
/control/alias tv /tracking/verbose  
{tv} 1
```

Aliases can be used recursively.

```
/control/alias file1 /diskA/dirX/fileXX.dat  
/control/alias file2 /diskB/dirY/fileYY.dat  
/control/alias run 1  
/myCmd/getFile {file{run}}
```

Loop: Scripting with UI commands

`/control/loop` and `/control/foreach` commands repeatedly execute a macro file.

- Aliased variable name can be used inside the macro file.

```
/control/loop [macroFile] [counterName] [initialValue]  
              [finalValue] [stepSize]
```

- `counterName` is aliased to the number as a loop counter

```
/control/foreach [macroFile] [counterName] [valueList]
```

- `counterName` is aliased to a value in `valueList`.
- `valueList` must be enclosed by double quotes (" ").

An example:

```
/control/loop myRun.mac Ekin 10. 20. 2.
```

`myRun.mac`

```
/control/foreach mySingleRun.mac pname "p pi- mu-"
```

`mySingleRun.mac`

```
/gun/particle {pname}  
/gun/energy {Ekin} GeV  
/run/beamOn 100
```

Extra Functionality of UI Command

G4UIcommands abort execution or issue warning in following cases:

- Wrong application state
- Wrong type of parameter
- Insufficient number of parameters
- Parameter out of its range
 - ✓ *for integer or double type parameters*
- Parameter out of its candidate list
 - ✓ *for string type parameters*
- Command not found

How to define your own UI commands

USER-DEFINED COMMAND

User-defined Commands

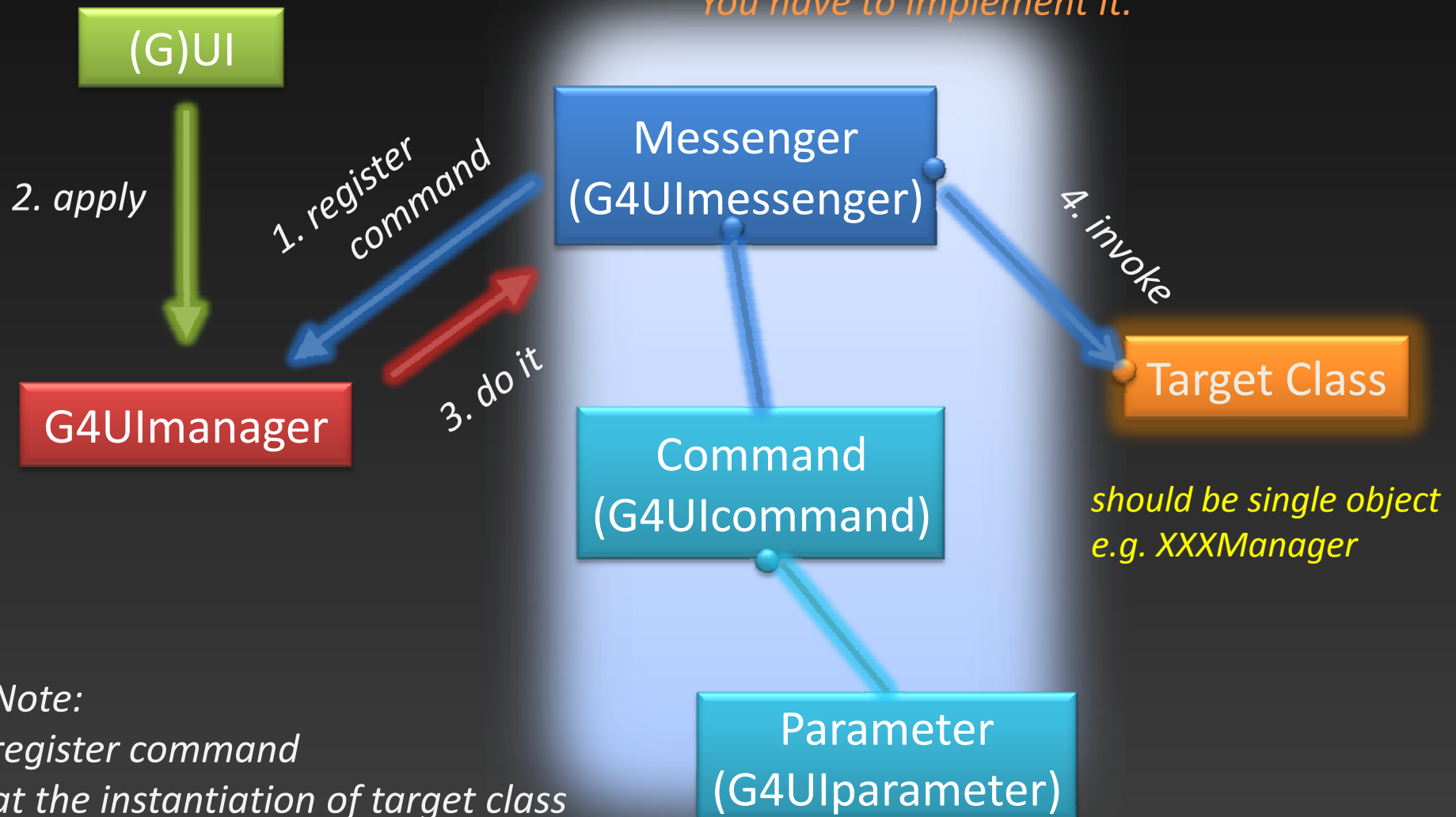
If built-in commands are not enough, **you define your own command.**

Geant4 provides several command classes, derived from *G4UIcommand*:

- G4UIcmdWith3Vector
- G4UIcmdWithADoubleAndUnit
- G4UIcmdWith3VectorAndUnit
- G4UIcmdWithAString
- G4UIcmdWithABool
- G4UIcmdWithAnInteger
- G4UIcmdWithADouble
- G4UIcmdWithoutParameter

Mechanism of UI command

You have to implement it.



Messenger class

You have to implement a **messenger** associating with a target class.

- Each messenger class is **derived from *G4UImessenger* base class**.
- A messenger class should be instantiated **in the constructor of the target class**.
- Target class should **be a single-object (singleton) class**
 - ✓ e.g. *XXXManager*
 - ✓ If not, there is a mess!

Implementation of your messenger classes

- Define command location: ***directory / command***
- void **SetNewValue**(G4UIcommand* command, G4String newValue)
 - ✓ Convert "*newValue*" parameter string to appropriate value(s) and invoke a method of the target class
- G4String **GetCurrentValue**(G4UIcommand* command)
 - ✓ Access to a get-method of the target class and convert the current values to a string

An example of command definition

```
A01DetectorConstMessenger::A01DetectorConstMessenger
(A01DetectorConstruction* a01)
: target_class(a01)
{
  mydetDir = new G4UIDirectory("/mydet/");
  mydetDir-> SetGuidance("A01 detector setup commands.");

  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");
}
```

Parameters

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

If "*omittable*" is true, the command can be issued without specifying a parameter value.

If "*currentAsDefault*" is true, the current value of the parameter is used as a default value.

- The 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)");`
- 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.
- You can also define a dimension (*unit category*).
 - ✓ `void SetUnitCategory(const char* unitCategory)`

`void SetCandidates(const char* candidateList)`

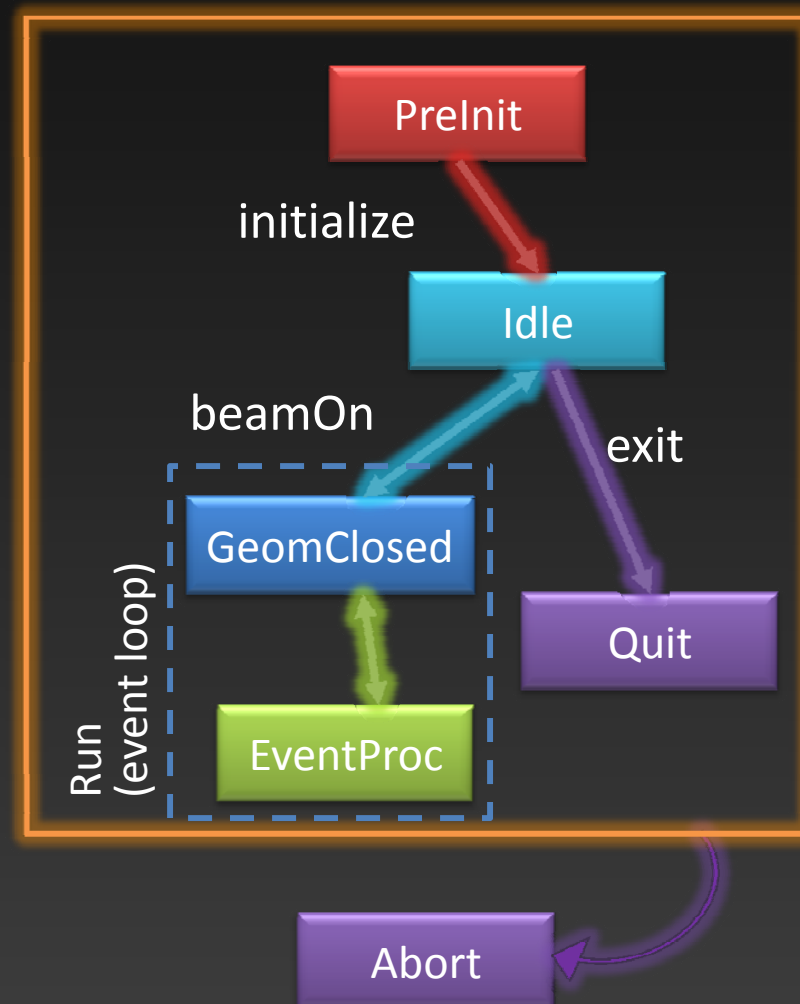
- Available for a command with string type parameter
- Candidates must be *delimited by a space*.

Available states

```
void AvailableForStates
(G4ApplicationState s1,...)
```

Geant4 has *6 application states*.

- **G4State_PreInit**
 - ✓ Material, Geometry, Particle and/or Physics Process need to be initialized/defined
- **G4State_Idle**
 - ✓ Ready to start a run
- **G4State_GeomClosed**
 - ✓ Geometry is optimized and ready to process an event
- **G4State_EventProc**
 - ✓ An event is processing
- **G4State_Quit**
 - ✓ (Normal) termination
- **G4State_Abort**
 - ✓ A fatal exception occurred and program is aborting



Converting between string and values

Derivatives of *G4UCommand* with numeric and boolean parameters have corresponding conversion methods.

From a string to value

- used in *SetNewValue()* method in a messenger
- Unit is taken into account automatically

```
G4bool GetNewBoolValue(const char*)
```

```
G4int GetNewIntValue(const char*)
```

```
G4double GetNewDoubleValue(const char*)
```

```
G4ThreeVector GetNew3VectorValue(const char*)
```

From a value to string

- used in *GetCurrentValue()* method in a messenger

```
G4String ConvertToString(...)
```

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

SetNewValue() and GetCurrentValue()

```
void A01DetectorConstMessenger
    ::SetNewValue(G4UIcommand* command, G4String newValue)
{
    if( command==armCmd ) {
        target-> SetArmAngle(armCmd-> GetNewDoubleValue(newValue));
    }
}
```

```
G4String A01DetectorConstMessenger
    ::GetCurrentValue(G4UIcommand* command)
{
    G4String cv;
    if( command==armCmd ){
        cv = armCmd-> ConvertToString(target->GetArmAngle(),"deg");
    }
    return cv;
}
```

Complicated UI command

UI command with any number of parameters with different types.

- A UI command with other types of parameters can be directly defined by *G4UIcommand* and *G4UIparameter*.

```
G4UIparameter(const char * parName,  
              char theType,  
              G4bool theOmittable);
```

- "*theType*" is the type of the parameter.
 - ✓ 'b' (boolean), 'i' (integer), 'd' (double), and 's' (string)
- Each *parameter* can take *guidance*, *default value* (in case "*theOmittable*" is true), *parameter range*, and *candidate list*.

Parameters can be added to a command by

```
G4UIcommand::SetParameter(G4UIparameter* const)
```

Converting string to values

For complicated command, convenient conversion method is not available.

G4Tokenizer tokenizes a string and converts each token to a numerical value.

```
SetNewValue(G4UIcommand* command, G4String newValues) {  
    G4Tokenizer next(newValues);  
    fAtomicNumber = StoI(next());  
    fAtomicMass = StoI(next());  
    G4String sQ = next();  
    if (sQ.isNull()) {  
        fIonCharge = fAtomicNumber;  
    } else {  
        fIonCharge = StoI(sQ);  
        sQ = next();  
        if (sQ.isNull()) {  
            fIonExciteEnergy = 0.0;  
        } else {  
            fIonExciteEnergy = StoD(sQ) * keV;  
        }  
    }  
}
```

G4UIcommand class has some basic conversion methods.

StoI() : convert string to int

StoD() : convert string to double

ItoS() : convert int to string

DtoS() : convert double to string

Be careful of “omittable” parameters.

Personal comment

Many applications are built on top of the Geant4 toolkit by creating original user interface

Majority of these applications use and extend existing UI

- HEP: ATLAS, CMS, LHCb ...
- Space science: GRAS, ...
- Medicine: GATE, GAMOS, ...

THANKS