

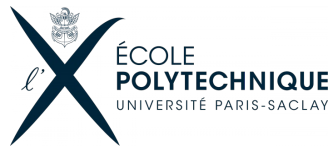


**GEANT4**  
A SIMULATION TOOLKIT

Version 11.1

# Definition of UI commands

Makoto Asai (Jefferson Lab)  
Geant4 Tutorial Course



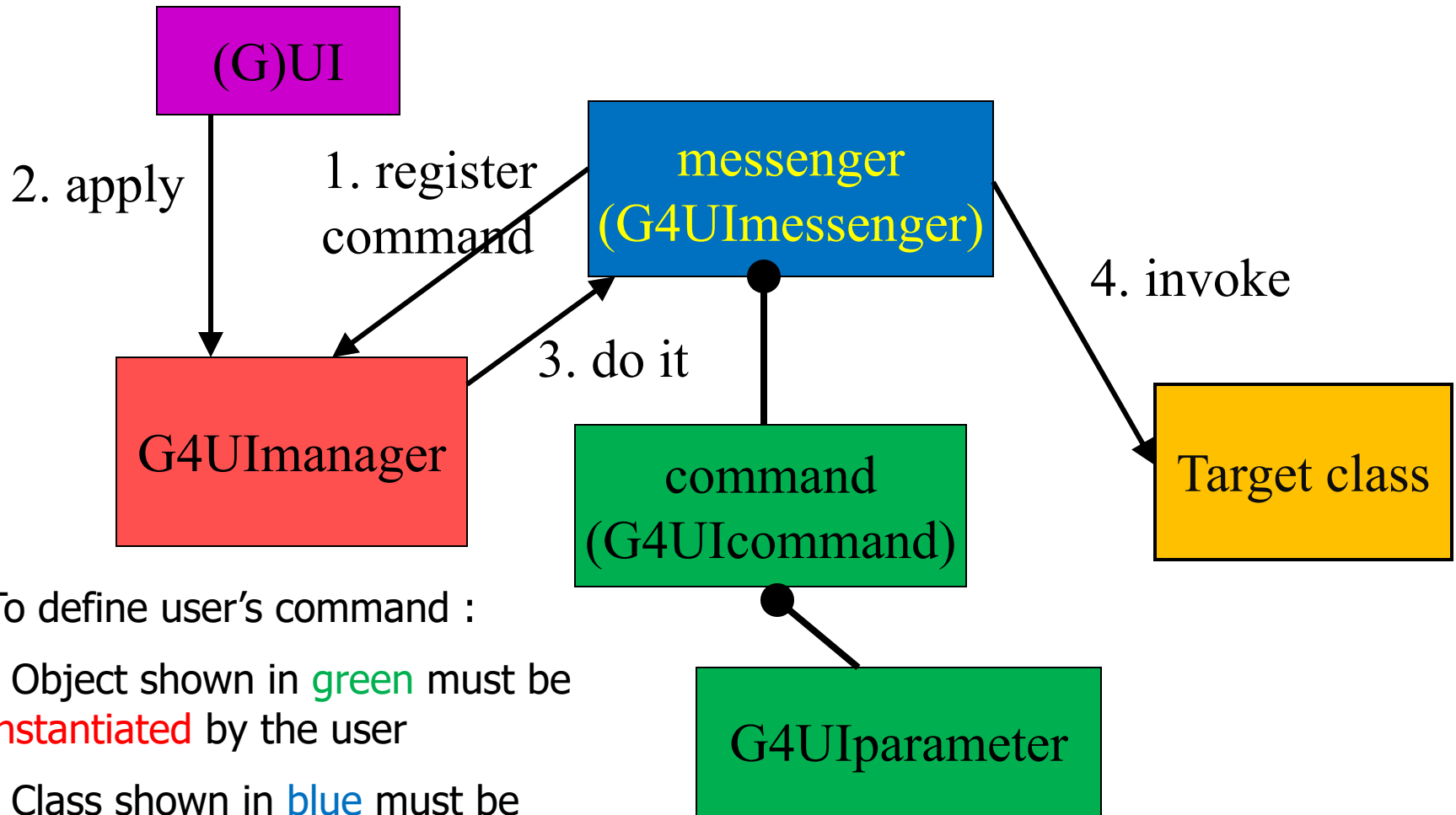
# Contents

- Mechanism of UI command
- Defining basic UI command
- Defining complicated UI command
- G4GenericMessenger
- UI command in multithreaded mode
- Some useful UI control commands

# Contents

- Mechanism of UI command
- Defining basic UI command
- Defining complicated UI command
- G4GenericMessenger
- UI command in multithreaded mode
- Some useful UI control commands

# Mechanism of UI command



To define user's command :

- Object shown in **green** must be **instantiated** by the user
- Class shown in **blue** must be **implemented and instantiated** by the user

- Each messenger class must be derived from **G4UImessenger** base class. A messenger class can handle more than one UI commands.
- A messenger class **should be instantiated by** the constructor of the **target class** to which commands should be delivered, and it **should be deleted** by the destructor of the target class.
- Methods of messenger class
  - **Constructor**
    - Define (instantiate) commands / command directories
  - **Destructor**
    - Delete commands / command directories
  - void **SetNewValue**(G4UIcommand\* command, G4String newValue)
    - This method is invoked when a command is issued.
    - Convert "newValue" parameter string to appropriate value(s) and invoke an appropriate method of the target class
  - G4String **GetCurrentValue**(G4UIcommand\* command)
    - This method is invoked when the current value(s) of parameter(s) of a command is asked by (G)UI.
    - Access to an appropriate get-method of the target class and convert the current value(s) to a string

# Contents

- Mechanism of UI command
- **Defining basic UI command**
- Defining complicated UI command
- G4GenericMessenger
- UI command in multithreaded mode
- Some useful UI control commands

# Definition (instantiation) of a command

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

```
A01DetectorConstMessenger::A01DetectorConstMessenger
(A01DetectorConstruction* tgt)
:target(tgt)
{
    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");
}
```

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

- **G4UIcommand** is a class which represents 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) so that 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 necessary G4UIparameter objects.



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

```
void SetParameterName (  
    const char*parName,  
    G4bool omittable,  
    G4bool currentAsDefault=false) ;  
  
void SetParameterName (  
    const char*nam1, const char*nam2, const char*nam3,  
    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 parameter, and the default value will be used.
- If "currentAsDefault" is true, current value of the parameter is used as a default value, otherwise default value must be defined with SetDefaultValue() method.

**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.

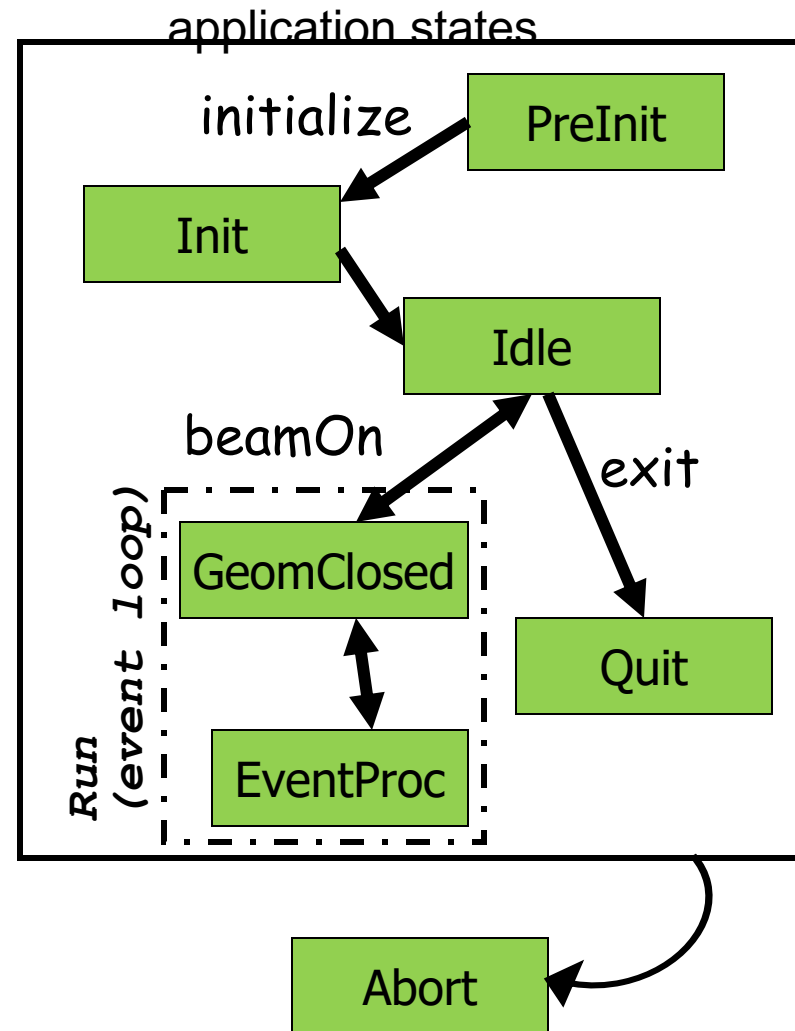
**void SetUnitCategory**(const char\* unitCategory)

**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.

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

- Define command's applicability for Geant4
- Geant4 has six application states.
  - G4State\_PreInit
    - Initial condition
  - G4State\_Init
    - During initialization
  - 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, G4State\_Abort
    - UI command unavailable



- Derivatives of G4UIcommand with numeric or boolean parameter(s) 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 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;
}
```

# Contents

- Mechanism of UI command
- Defining basic UI command
- **Defining complicated UI command**
- G4GenericMessenger
- UI command in multithreaded mode
- Some useful UI control commands

- Complicated UI command means a UI command with parameters which is not included in the deliverable classes.
  - G4UICmdWithoutParameter, G4UICmdWithAString, G4UICmdWithABool, G4UICmdWithAnInteger, G4UICmdWithADouble, G4UICmdWithADoubleAndUnit, G4UICmdWith3Vector, G4UICmdWith3VectorAndUnit
- A UI command with other type of parameters must be defined by G4UIcommand base class with necessary numbers of G4UIparameter objects.
  - G4UIparameter**(const char \* parName, char theType, G4bool theOmittable);
    - “parName” is the name of the parameter which will be used by the range checking and help
    - “theType” is the type of the parameter.
      - ‘b’ (boolean), ‘i’ (integer), ‘l’ (long int), ‘d’ (double), or ‘s’ (string)
    - Each parameter can take one line of guidance, a default value in case “theOmittable” is true, a range (for numeric type parameter), and a candidate list (for string type parameter).

- A G4UIcommand object can take arbitrary number of G4UIparameter objects.
  - Names of parameter must be different to each other (within the command).
  - It takes arbitrary number of guidance lines.
  - Availability for Geant4 states can be set.
  - In addition to ranges defined to individual parameters, it may take another range definition where values of more than one parameters can be compared to others.
- When you define a parameter that is used as a unit,
  - G4UIparameter::SetDefaultUnit() should be used to specify the default unit as well as the dimension.
  - Once the default unit is defined, no other unit of different dimension will be accepted.



# /gun/ion command

```
ionCmd = new G4UIcommand("/gun/ion",this);
ionCmd->SetGuidance("Set properties of ion to be generated.");
ionCmd->SetGuidance("[usage] /gun/ion Z A Q E");
ionCmd->SetGuidance("  Z:(int) AtomicNumber");
ionCmd->SetGuidance("  A:(int) AtomicMass");
ionCmd->SetGuidance("  Q:(int) Charge of Ion in unit of e (-1
                    for Z)");
ionCmd->SetGuidance("  E:(double) Excitation energy (in keV)");

G4UIparameter* param;
param = new G4UIparameter("Z", 'i', false);
ionCmd->SetParameter(param);
param = new G4UIparameter("A", 'i', false);
ionCmd->SetParameter(param);
param = new G4UIparameter("Q", 'i', true);
param->SetDefaultValue("-1");
ionCmd->SetParameter(param);
param = new G4UIparameter("E", 'd', true);
param->SetDefaultValue("0.0");
ionCmd->SetParameter(param);
```

Parameters are  
registered by  
their orders.

# Converting string to values

- For complicated command, convenient conversion method is not available. Use G4Tokenizer to tokenize the string and convert each token to numerical values.

```
SetNewValue(G4UIcommand * command, G4String newValues)
```

```
{  
    G4Tokenizer next( newValues );  
    fAtomicNumber = StoI(next());  
    fAtomicMass = StoI(next());  
    G4String sQ = next();  
    if (sQ.isNull() || sQ=="-1") {  
        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.

# Contents

- Mechanism of UI command
- Defining basic UI command
- Defining complicated UI command
- **G4GenericMessenger**
- UI command in multithreaded mode
- Some useful UI control commands

- G4GenericMessenger is a short-cut for defining simple UI commands that are
  - associated with one target class, and grouped in one command directory
  - dealing only with native variable types (string, Boolean, integer, long int, double) or G4ThreeVector.
- G4GenericMessenger is a concrete class you can just instantiate without implementing a dedicated messenger class.

```
class G4GenericMessenger : public G4UImessenger
```

```
{  
public: Target class object Command directory  
G4GenericMessenger(void* obj, const G4String& dir ← "", const G4String& doc = "");  
Command& DeclareProperty  
  (const G4String& name, const G4AnyType& variable, const G4String& doc = "");  
Command& DeclarePropertyWithUnit  
  (const G4String& name, const G4String& defaultUnit, Target class data member  
   const G4AnyType& variable, const G4String& doc = "");  
Command& DeclareMethod  
  (const G4String& name, const G4AnyMethod& fun, const G4String& doc = "");  
Command& DeclareMethodWithUnit Target class public set method  
  (const G4String& name, const G4String& defaultUnit, Command name  
   const G4AnyMethod& fun, const G4String& doc = "");
```

# example

```
B5PrimaryGeneratorAction::B5PrimaryGeneratorAction()
{
    fMessenger = new G4GenericMessenger(this,
        "/B5/generator/", "Primary generator control");

    // momentum command
    G4GenericMessenger::Command& momentumCmd
        = fMessenger->DeclarePropertyWithUnit("momentum", "GeV", fMomentum,
            "Mean momentum of primaries.");
    momentumCmd.SetParameterName("p", true);
    momentumCmd.SetRange("p>=0.");
    momentumCmd.SetDefaultValue("1.");
}

B5PrimaryGeneratorAction::~~B5PrimaryGeneratorAction()
{
    delete fMessenger;
}
```

*Default unit*

*Data member of  
the target class*

# Contents

- Mechanism of UI command
- Defining basic UI command
- Defining complicated UI command
- G4GenericMessenger
- **UI command in multithreaded mode**
- Some useful UI control commands

- In multithreaded mode, each thread (both master and worker) has its own G4UImanager object.
  - Commands instantiated in a thread are registered to the G4UImanager of that thread.
- Every UI command is executed in the master thread if the command is issued in the master thread (including interactive session or macro file), and then broadcasted to each worker thread prior to the beginning of thread-local event loop (i.e. at the time of */run/beamOn*).
  - If the user wants to broadcast command(s) immediately, use */run/workersProcessCmds* command.
  - If a command should not be distributed to worker threads, *SetToBeBroadcasted(false)* should be set for that command.
  - If a command is hard-coded in an object in a worker thread, it is executed only in that worker thread. It won't be transmitted to other worker threads.

# Contents

- Mechanism of UI command
- Defining basic UI command
- Defining complicated UI command
- G4GenericMessenger
- UI command in multithreaded mode
- **Some useful UI control commands**



- Alias can be defined by
  - `/control/alias [name] [value]`
    - It is also set with `/control/loop` and `/control/foreach` commands
    - Aliased value is always treated as a string even if it contains numbers only.
- Alias is to be used with other UI command.
  - Use curly brackets, `{` and `}`.
  - For example, frequently used lengthy command can be shortened by aliasing.

```
/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/getEnv runNo
```

```
/myCmd/getData {file{runNo}}
```

Get a shell variable value and define it as an alias

- **/control/loop** and **/control/foreach** commands execute a macro file more than once. 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 (" ")
- on UI terminal or other macro file  
    **/control/loop** myRun.mac **Ekin** 10. 20. 2.
- in myRun.mac   
    **/control/foreach** mySingleRun.mac **pname** "p pi- mu-"
- in mySingleRun.mac   
    **/gun/particle** {**pname**}  
    **/gun/energy** {**Ekin**} GeV  
    **/run/beamOn** 100

- `/control/shell <shell_command>`
  - Execute a Unix shell command (e.g. `/control/shell ls`)
- `/control/getEnv <shell_variable_name>`
  - Get a shell variable value and define it as an alias.
- `/control/getVal <alias_name> <UI_command> <index>`
  - Get the current value of the UI command and define it as an alias.
  - `<index>` is the index of the parameter to take if the command has more than one parameters.
- `/control/add <alias_name> <val_1> <val_2>`
- `/control/subtract <alias_name> <val_1> <val_2>`
- `/control/multiply <alias_name> <val_1> <val_2>`
- `/control/divide <alias_name> <val_1> <val_2>`
- `/control/remainder <alias_name> <val_1> <val_2>`
  - `<val_1>` and/or `<val_2>` can be aliases.
  - If `<alias_name>` already exists, value is overwritten.

# Some other useful UI commands

- `/control/doif <val_1> <comp> <val_2> <UI_command>`
- `/control/if <val_1> <comp> <val_2> <macro_file>`
  - Equivalent to `/control/doif <val_1> <comp> <val_2> /control/execute <macro_file>`
- `/control/strdoif <str_1> <comp> <str_2> <shell_command>`
- `/control/strif <str_1> <comp> <str_2> <macro_file>`
  - E.g. `/control/getVal particleName /gun/particle`  
`/control/strif {particleName} == proton myMacro.mac`
- `/control/doifBatch <UI_command>`
- `/control/ifBatch <macro_file>`
- `/control/doifInteractive <UI_command>`
- `/control/ifInteractive <macro_file>`
  - E.g. `/control/ifInteractive vis.mac`
- `/control/useDoublePrecision`
  - Use double precision for printing out the current parameter value.