Geant4 v8.3

Stanford
Linear
Accelerator
Center

User Interface II

Makoto Asai (SLAC)

Geant4 Tutorial Course

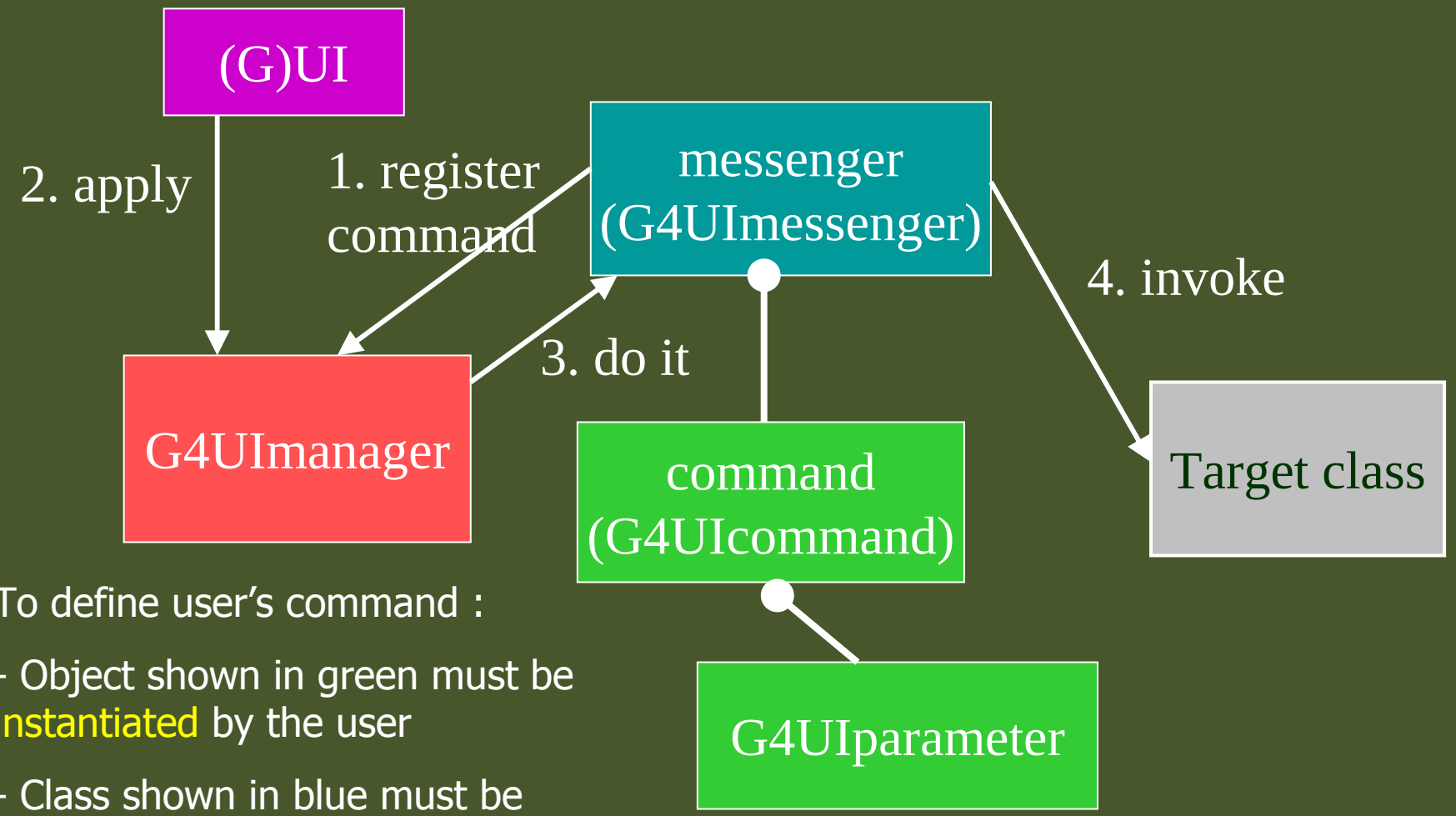
Geant 4

Contents

- Mechanism of UI command
- Defining basic UI command
- Defining complicated UI command

Mechanism of UI command

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

Messenger class

- 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 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)
    - Convert "newValue" parameter string to appropriate value(s) and invoke an appropriate method of the target class
    - This method is invoked when a command is issued.
  - G4String GetCurrentValue(G4UIcommand* command)
    - Access to an appropriate get-method of the target class and convert the current value(s) to a string
    - This method is invoked when the current value(s) of parameter(s) of a command is asked by (G)UI.

Defining basic UI command

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 utilized as a short description of the command.

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
  - G4UIcmdWithAInteger
  - G4UIcmdWithADouble, G4UIcmdWithADoubleAndUnit
  - G4UIcmdWith3Vector, G4UIcmdWith3VectorAndUnit
  - G4UIdirectory
- A UI command with other type of parameters must be defined by G4UIcommand base class with G4UIparameter.

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);
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 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)
```

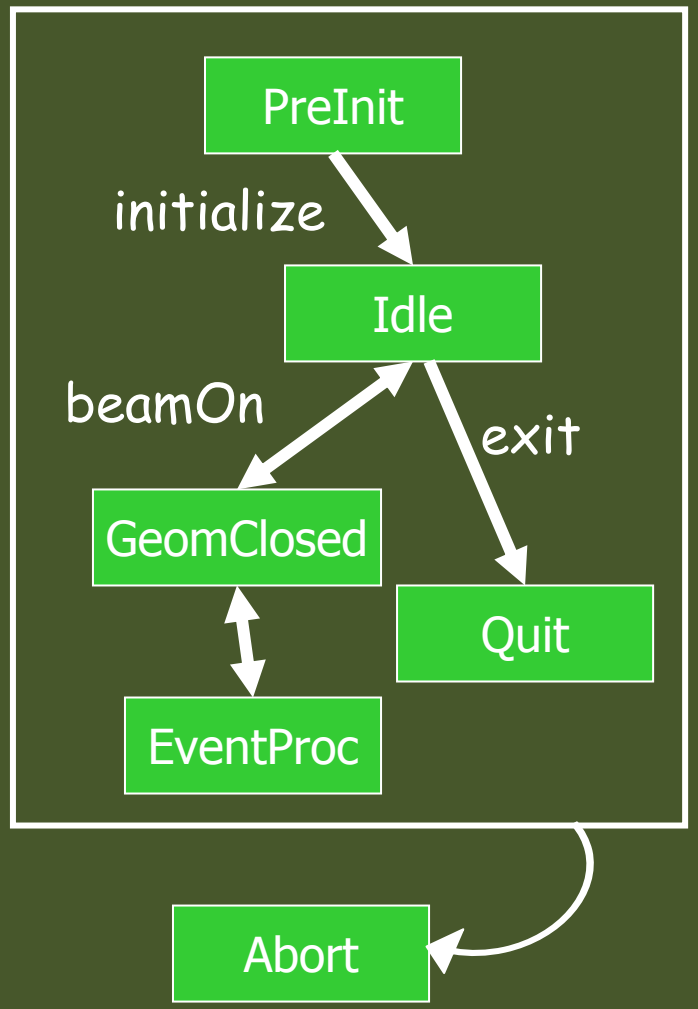
```
void SetCandidates(const char* candidateList)
```

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

Available state

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

- Define command's applicability for Geant4 application states.
- Geant4 has six application states.
  - G4State_PreInit
    - Material, Geometry, Particle and/or Physics Process need to be initialized
  - 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



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 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;
}
```

Defining complicated UI command

Complicated UI command

- 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 G4UIparameter.

```
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), 'd' (double), and '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).

Complicated UI command

- 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 each other.

/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");
ionCmd->SetGuidance("        Z:(int) AtomicNumber");
ionCmd->SetGuidance("        A:(int) AtomicMass");
ionCmd->SetGuidance("        Q:(int) Charge of Ion (in unit of e)");
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("0");
ionCmd->SetParameter(param);
param = new G4UIparameter("E",'d',true);
param->SetDefaultValue("0.0");
ionCmd->SetParameter(param);
```

Parameters are registered along their orders.

Converting string to values

- For complicated command, convenient conversion method is not available. Please 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()) {
          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.