

# Интерфейс пользовательских команд



---

ЛЕКЦИЯ 10



# Содержание

---

- ☐ Режимы работы
- ☐ Интерактивный режим
- ☐ Структура команд
- ☐ Реализация новых команд
- ☐ **G4Uicommand** и его потомки



# Команды в Geant4: введение

---

Geant4 позволяет использовать команды, позволяющие управлять отдельными этапами моделирования напрямую во время работы проекта.

В Geant4 управление пользовательским интерфейсом, а так же обработку всех команд осуществляет класс **G4UImanager**. Пользователь **НЕ ДОЛЖЕН** осуществлять вызов конструктора данного класса или наследовать его.

Вместо это в процессе загрузки необходимо вызвать статический метод данного класса, возвращающий указатель на уже существующий объект данного класса.

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



# Режимы работы

В Geant4, по умолчанию, предполагается 2 режима работы: интерактивный или пакетный.

Под **пакетным** режимом работы следует понимать использование заранее созданного макрос файла представляющего собой список команд в планируемом порядке их запуска. Принято что данный файл должен иметь расширение **\*.mac**:

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

if (argc != 1) {
    // batch mode
    std::string command = "/control/execute ";
    std::string fileName = argv[1];
    UImanager->ApplyCommand(command + fileName);
}
```



# Режимы работы: продолжение

Под **интерактивным** режимом подразумевается вызов визуальной оболочки Geant4(если настроена), а так же интерфейса командной строки для последующего ввода команд:

```
    else {  
        // interactive mode : define UI session  
#ifdef G4UI_USE  
    G4UIExecutive *ui = new G4UIExecutive(argc, argv);  
#ifdef G4VIS_USE  
    UImanager->ApplyCommand("/control/execute init_vis.mac");  
#else  
    UImanager->ApplyCommand("/control/execute init.mac");  
#endif  
    ui->SessionStart();  
    delete ui;  
#endif  
}
```



# Режимы работы: интерактивный режим

1. Список доступных команд (вкладка help)

2. Описание команды, и используемых параметров

3. Командная строка для ввода команд

The screenshot shows the Geant4 interactive mode interface. The main window displays a 3D visualization of a cube. The left sidebar contains a 'Scene tree' and a 'Help' tab. The 'Help' tab is active, showing a list of commands under the 'run' category. The 'beamOn' command is selected. Below the command list, there is a section for the 'beamOn' command, including a 'Guidance' section and a table of parameters.

Parameter	Guidance	Type
1	numberOfEvent	i
2	macroFile	s
3	nSelect	i

The output window at the bottom shows the following text:

```
Threads: All
Visualization verbosity changed to quiet (0)
/tracking/storeTrajectory 2
```

The session bar at the bottom indicates the current session is `/run/beamOn`.



# Структура команд

Команды пользовательского интерфейса строятся следующим образом:



*Данная команда устанавливает уровень подробности выводимой информации на 1*

Следует отметить что командная директория может содержать внутренние каталоги, следовательно командой является всегда самый последний аргумент, к примеру



*Данная команда очищает экран*



# Структура команд: параметры

- ❑ Параметры команд могут быть следующих типов: **G4String**, **G4int**, **G4bool** и **G4double**
- ❑ Пробел играет роль разделителя
- ❑ Двойные-кавычки (""") используются для string-параметров, содержащий пробел.

Кроме того параметр команды может быть не указан. В этом случае используется значение по умолчанию для пропущенного параметра.

- ❑ Значение «по умолчанию» это predetermined значение или текущее значение, указанное в определении
- ❑ Если нужно использовать значение « по умолчанию» для первого параметра, а для второго-указать то для первого параметра следует установить знак «!» в качестве значения:

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





# Реализация новых команд

В Geant4 предусмотрена возможность по созданию новых команд, в добавление к уже существующим. Для этого следует создать объект класса, унаследованного от **G4UImessenger**, а для реализации действий, предусмотренных данными командами, следует переопределить метод **SetNewValue(G4UCommand\*, G4String)**.

К примеру реализуем команды для изменения геометрии. Шаблон класса команд может выглядеть следующим образом:

```
8  #include <G4UImessenger.hh>
9  #include "Geometry.hh"
10
11  class Geometry;
12
13  class GeometryMessenger: public G4UImessenger {
14  public:
15      GeometryMessenger(Geometry *det);
16
17      ~GeometryMessenger();
18
19      void SetNewValue(G4UCommand *command, G4String newValue);
20  };
```



# Реализация новых команд: Геометрия

Обратите внимание, что конструктор выше представленного класса содержит указатель на объект класса **Geometry**, т.е. класса, в котором мы планируем изменить какой-либо параметр. Тогда:

- ❑ Для класса **Geometry** следует ввести дополнительное «поле-указатель» на объект нашего класса **GeometryMessenger**
- ❑ Инициализировать данный объект в конструкторе класса **Geometry** следующим образом:

```
geometryMessenger = new GeometryMessenger(this);
```

Где, при создании нашей геометрии, объекту класса обработки команд передастся указатель на создаваемую геометрию с помощью ключевого слова **this**.



# Реализация новых команд: Геометрия

Рассмотрим остальные особенности, необходимые для использования команд в классе **Geometry**:

- Пусть метод **Construct()** класса **Geometry** выглядит следующим образом (*построение простейшего кубика*):

```
36 G4VPhysicalVolume* Geometry::Construct() {  
37  
38     G4Box *box = new G4Box("box", det_size / 2., det_size / 2., det_size / 2.);  
39     G4LogicalVolume *box_log = new G4LogicalVolume(box, box_mat, "box_log");  
40     new G4PVPlacement(0, G4ThreeVector(0,0,0), box_log, "pvp_box", world_log, false, 0);  
41  
42     return world_VP;  
43 }
```

- Где **det\_size** и **box\_mat** параметры, которые мы будем изменять. Реализуем метод для установки нового значения, к примеру для **det\_size**:

```
44 void Geometry::SetDetSize(G4double newValue){  
45     det_size = newValue;  
46 }
```



# Реализация новых команд: Геометрия

При вызове нашей команды необходимо перестроить геометрию заново, но, т.к. она уже была построена при инициализации проекта, её необходимо очистить. Для этого перепишем метода **Construct()** следующим образом:

```
36 G4VPhysicalVolume* Geometry::Construct() {  
37     G4GeometryManager::GetInstance()->OpenGeometry();  
38     G4PhysicalVolumeStore::GetInstance()->Clean();  
39     G4LogicalVolumeStore::GetInstance()->Clean();  
40     G4SolidStore::GetInstance()->Clean();  
41  
42     G4double size = 50 * m;  
43     world = new G4Box("world", size / 2., size / 2., size / 2.);  
44     world_log = new G4LogicalVolume(world, world_mat, "world_log");  
45     world_VP = new G4PVPlacement(0, G4ThreeVector(), world_log, "world_PV", 0, false, 0);  
46  
47     G4Box *box = new G4Box("box", det_size / 2., det_size / 2., det_size / 2.);  
48     G4LogicalVolume *box_log = new G4LogicalVolume(box, box_mat, "box_log");  
49     new G4PVPlacement(0, G4ThreeVector(0,0,0), box_log, "pvp_box", world_log, false, 0);  
50  
51     return world_VP;  
52 }
```

Где в первых четырех строчках осуществляется вызов существующей геометрии (*если есть*) после чего поочередно очищаются контейнеры с физическими, логическими и геометрическими объемами. **Обратите внимание**, что мир следует инициировать так же в методе **Construct()** иначе он будет уничтожен и не восстановлен.



# Реализация новых команд: Геометрия

---

Кроме того, при изменении параметра необходимо сообщить **RunManager** о том что геометрию следует перестроить. Это можно осуществить добавлением следующих строк в метод по изменению параметра:

```
51 void Geometry::SetDetSize(G4double newValue){  
52     det_size = newValue;  
53     G4RunManager::GetRunManager()->DefineWorldVolume(Construct(  
54     G4RunManager::GetRunManager()->ReinitializeGeometry());  
55 }
```



# Реализация новых команд: G4UICmdWithADouble

Вернемся к классу **GeometryMessenger**

- ❑ В конструкторе создадим директорию (**G4UIdirectory**) с пользовательскими командами.
- ❑ Создадим простейшую команду принимающую новое значение типа **G4double**

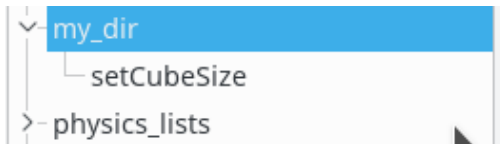
```
7 GeometryMessenger::GeometryMessenger(Geometry* Det):det(Det){  
8     my_dir = new G4UIdirectory("/my_dir/");  
9     my_dir ->SetGuidance("UI commands specific to this example");  
10  
11     setSize = new G4UICmdWithADouble("/my_dir/setCubeSize",this);  
12     setSize->SetGuidance("Select Size of the box.");  
13     setSize->SetParameterName("size", false);  
14 }
```

- ❑ Метод **SetGuidance(G4String)** позволяет добавлять к команде описание, отображаемое в режиме визуализации (как для директории так и для команд)
- ❑ Метод **SetParameterName(G4String, G4bool)** позволяет устанавливать отображаемое имя параметра, **false** в данном случае означает что параметр не может быть опущен и должен быть обязательно указан.

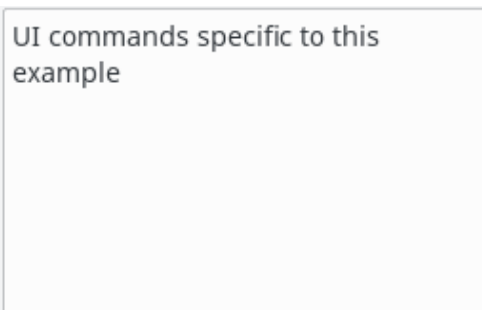
# Реализация новых команд: Визуальное представление



В интерактивном режиме в списке команд директория будет выглядеть следующим образом:



В окне описания команд будет выведена ранее заданная информация (с помощью метода **SetGuidance()**):



# Реализация новых команд: Визуальное представление



При выборе созданной команды из списка отобразится следующее сообщение в окне описания команд:

<b>Command</b> /my_dir/setCubeSize <b>Guidance</b> : Select Size of the box.							
	Parameter	Guidance	Type	Ommitable	Default	Range	Candidate
1	size		d	False			

Тем не менее при вызове данной команды ничего не произойдет и параметр не изменится.



# Реализация новых команд: установка значения



Чтобы изменения вступили в силу необходимо реализовать метод `SetNewValue(G4UIcommand*, G4String)`.

```
21 void GeometryMessenger::SetNewValue(G4UIcommand *command, G4String newValue) {  
22     if (command == setSize) {  
23         det->SetDetSize(setSize->GetNewDoubleValue(newValue));  
24     }  
25 }
```

В котором осуществляется проверка того, какая именно команда была вызвана, и при необходимости значение передаваемое в этой команде можно привести к нужному формату.

К примеру, в данном случае преобразование к типу `G4double` осуществляет встроенный метод `G4cmdWithADouble::GetNewDoubleValue(G4String)`

Полученное значение передается объекту класса `Geometry` через ранее заданный метод `Geometry::SetDetSize(G4double)`, в результате чего запускается перезагрузка геометрии с новым значением



# Реализация новых команд: G4UICmdWithAString

Для изменения материала куба, можно реализовать команду которая принимает **G4String** значение:

```
setMaterial = new G4UICmdWithAString("/my_dir/setMaterial", this);  
setMaterial->SetGuidance("Select Material of the box.");  
setMaterial->SetParameterName("name", false);
```

Для того чтобы передать новое значение в **SetNewValue(G4Uicommand\*, G4String)**:

```
if (command == setMaterial){  
    det->SetDetMaterial(newValue);  
}
```

Где **Geometry::SetDetMaterial(G4String)** выглядит следующим образом:

```
57 void Geometry::SetDetMaterial(G4String newValue){  
58     box_mat = nist->FindOrBuildMaterial(newValue);  
59     G4RunManager::GetRunManager()->DefineWorldVolume(Construct());  
60     G4RunManager::GetRunManager()->ReinitializeGeometry();  
61 }
```



# Реализация новых команд: Визуальное представление для **G4UICmdWithAString**

При выборе созданной команды из списка отобразится следующее сообщение в окне описания команд:

**Command** /my\_dir/setMaterial  
**Guidance** : Select Material of the box.

	Parameter	Guidance	Type	Ommitable	Default	Range	Candidate
1	name		s	False			

Если существует необходимость заполнить остальные ячейки таблицы то команду так же можно реализовать через **G4Uicommand** (см. след. слайд)



# Реализация новых команд: G4Uicommand

```
setMaterial = new G4Uicommand("/my_dir/setMaterial",this);  
setMaterial->SetGuidance("Select Material of the box.");
```

```
G4UIparameter* mat = new G4UIparameter("name",'s',true);  
mat->SetGuidance("Name from NIST");  
mat->SetDefaultValue("G4_Li");  
mat->SetParameterCandidates("G4_Na G4_Mn");
```

```
setMaterial->SetParameter(mat);
```

где **G4Uicommand()** представляет собой конструктор команды без параметра, а **G4UIparameter\*** является указателем на объект класса параметров команд где в конструкторе **G4UIparameter(const char \* *theName*, char *theType*, G4bool *theOmittable*)**:

- ***theName*** – имя параметра
- ***theType*** – тип параметра задаваемый символом (в данном случае это «строка»)
- ***theOmittable*** – **true/false** в зависимости от того можно/нельзя вызывать команду без параметра



# Реализация новых команд: методы G4UIparameter

- ❑ **SetGuidance(G4String)** – добавить описание к параметру в таблицу
- ❑ **SetDefaultValue(G4String)** – установить значение по умолчанию (т.е. если команда вызовется без параметра то в качестве значения будет передано это значение (*в данном случае "G4\_Li"*))
- ❑ **SetParameterCandidates(const char\*)** – позволяет перечислить допустимые значения параметра (*через пробел*). В случае попытки передать недопустимое значение команда будет отклонена, а также отобразится следующее сообщение:

```
parameter value (G4_Fe) is not listed in the candidate  
List.  
command refused (500):"/my_dir/setMaterial G4_Fe"
```

# Реализация новых команд: Визуальное представление для **G4Uicommand**



При выборе созданной команды из списка отобразится следующее сообщение в окне описания команд:

**Command** /my\_dir/setMaterial  
**Guidance** : Select Material of the box.

	Parameter	Guidance	Type	Ommitable	Default	Range	Candidate
1	name	Name from NIST	s	True	G4_Li		G4_Na G4_Mn



# G4Uicommand и его потомки

---

- ❑ **G4Uicommand** – является базовым классом для всех классов команд
- ❑ **G4UicmdWithoutParameter** – команда без параметра
- ❑ **G4UicmdWithABool** – команда принимающая один параметр **G4bool**
- ❑ **G4UicmdWithAnInteger** – команда принимающая один параметр типа **G4int**
- ❑ **G4UicmdWithADouble** – команда принимающая один параметр типа **G4double**
- ❑ **G4UicmdWithAString** – команда принимающая один параметр типа **G4String**
- ❑ **G4UicmdWith3Vector** – команда принимающая один параметр типа **G4ThreeVector**

Таким образом все команды по умолчанию принимают один параметр. Если же нужно добавить более одного параметра, то следует использовать базовый класс **G4Uicommand**, добавляя параметры через **G4Uiparameter** (аналогично примеру с именем материала).