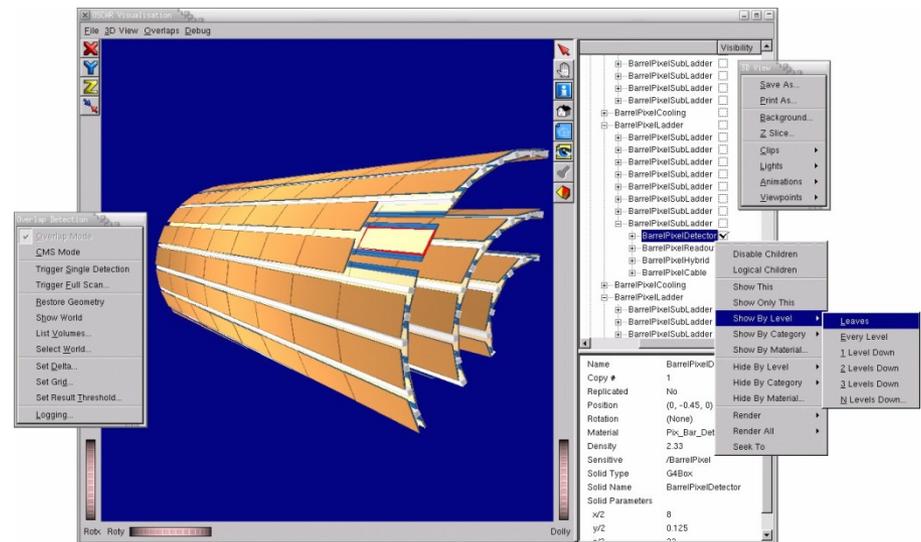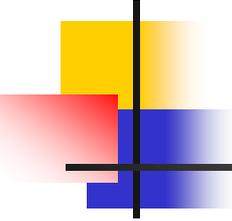# Geant4 (G)UI
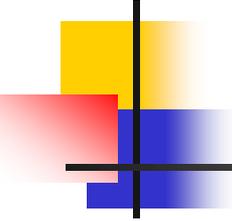
Luciano Pandola

INFN-LNGS and LNS

Steering the simulation - 1

- A Geant4 simulation can be **steered in three ways**:
  - everything hard-coded in the C++ source (also the number of events to be shot). You need to re-compile for any change (not very smart, actually!)
  - batch session (via an ASCII macro)
 - commands captured from an interactive session

Steering the simulation - 2

- Setting up batch mode (namely, read commands from a macro file) in the main()

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

  G4String command = "/control/execute";

  G4String fileName = argv[1];

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

  takes the first argument after the executable as the macro name and runs it

- Your executable can be run as

  `myExecutable mymacro.mac`

- To execute a macro interactively:

  `/control/execute mymacro.mac`
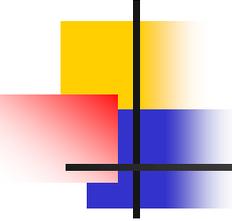
Steering the simulation - 3

- Setting up **interactive mode** is also easy – but there are many choices of interface
  - All of them must be derived from the abstract class `G4UIsession`
  - Geant4 provides several implementations
- In the `main(),` according to the computer environments, construct a `G4UIsession` concrete class provided by Geant4 and invoke its `SessionStart()` method
  - The `G4UIExecutive` takes care of selecting the most appropriate UI given the system environment
    - GUI's are given higher priority, terminal-like lower priority
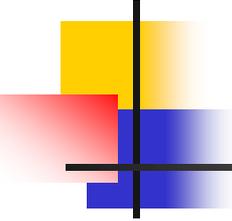
Select G(UI)

- Geant4 provides several interfaces for various (G)UI:
  - **G4UIterminal**: C-shell like character terminal
  - **G4UItcsh**: tcsh-like character terminal with command completion, history, etc
  - **G4UIGAG**: Java based graphic UI (GUI)
  - **G4UIXm**: Motif-based GUI, command completion
  - **G4UIQt**: GUI based on Qt libraries
- Define and invoke them like G4UIterminal
  ```
  session = new G4UIGAG();
  session->StartSession();
  ```
- Or (better) use the **G4UIExecutive**

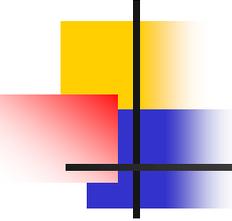# An example of interactive session – let G4UIExecutive choose

- **For instance:** in the `main()`

```
G4UIExecutive* session =
    new G4UIExecutive(argc, argv);
```
Create an instance of the G4UIExecutive

```
if (argc==1)
{
```
If there are no arguments after the executable, starts an interactive session

```
    session->SessionStart();
    delete session;
}
```
Start the session → gives the prompt

Don't forget to delete it

On my system:

```
Available UI session types: [GAG, tcsh, csh]
```

An example of interactive session – you make the choice (v. 1)

- You can drive the choice of the **`G4UIExecutive`** at run-time

`Available UI session types: [GAG, tcsh, csh]`

- If you do not want GAG (highest priority) and you want TCSH:

   `unset G4UI_GAG_USE`

   `export G4UI_TCSH_USE=1`

An example of interactive session – you make the choice (v. 2)

- **For instance:** in the `main()`

```
G4UIsession* session=0;
```
Create a (null) pointer to the base session class

```
if (argc==1)
 {
```
If there are no arguments after the executable, starts an interactive session

```
    session = new G4UIterminal;
    session->SessionStart();
    delete session;
 }
```
Define the session as a dumb terminal, and **starts** it
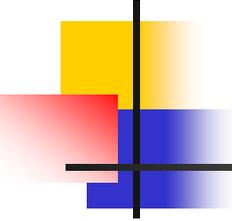
Don't forget to delete it
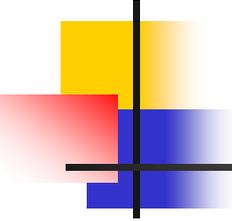
User Interface Choices

- `G4UIterminal` – C-shell-like character terminal
  - runs on all Geant4-supported platforms
- `G4UItcsh` – tcsh-like character terminal with command completion, history, etc.
  - runs only on Solaris and Linux
- `G4UIXm, G4UIXaw, G4UIXWin32` – G4UIterminal implemented over Motif, Athena and WIN32 libraries
  - runs on Unix/Linux and Windows, respectively
- `G4UIGAG` – Java-based GUI
  - runs on all Geant4 platforms
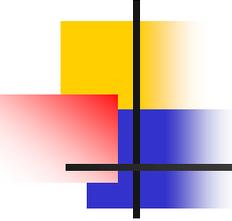
Useful GUI Tools Released by Geant4 Developers

- GGE: Geometry editor based on Java GUI
  - http://erpc1.naruto-u.ac.jp/~geant4
- GPE: Physics editor based on Java GUI
  - http://erpc1.naruto-u.ac.jp/~geant4
- OpenScientist: interactive environment for analysis
 - http://www.lal.in2p3.fr/OpenScientist

Built-in user commands

- Geant4 provides a number of general-purpose user interface commands which can be used:
  - interactively via a (G)UI
   ```
    Idle> /run/setCut [value] [unit]
    ```
  - in a macro file
- Within C++ code using the `ApplyCommand()` method of `G4UImanager`
  ```
  G4UImanager::GetUIpointer()
  ->ApplyCommand("/run/setCut 1 cm");
  ```
- A complete list of built-in commands is available in the Geant4 Application Developers Guide, Chapter 7.1
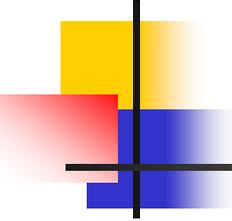
# User-defined commands (1)

- If built-in commands are not enough, you can make your own (e.g. change at run-time parameters of primary generator, etc.)

- Geant4 provides several command classes, all derived from **G4UIcommand**, according to the type of argument they take
  - **G4UIcmdWithoutParameter**
  - **G4UIcmdWithABool**
  - **G4UIcmdWithADouble**
  - **G4UIcmdWithADoubleAndUnit**
  - **...**
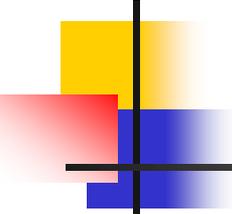
# User-defined commands (2)

- Commands have to be defined in messenger classes, that **inherit from G4UImessenger**

- Define the command in the constructor:

```
G4UIcmdWithADoubleAndUnit* fThetaCmd =
    new G4UIcmdWithADoubleAndUnit
        ("/prim/angle",this);
```

Command taking as argument a double and a unit, called /prim/angle

```
fThetaCmd->SetGuidance("Opening angle of source");
fThetaCmd->SetDefaultUnit("deg");
fThetaCmd->SetUnitCandidates("deg rad");
```

Sets guidance, default unit, etc.

- Delete the command in the destructor

User-defined commands (3)

- Define the action of the command in the **SetNewValue()** method of the messenger:

```
void MyMessenger::SetNewValue
   (G4UIcommand* cmd,G4String string)
{
  if (cmd == fThetaCmd)
  {
     G4double value = fThetaCmd
     ->GetNewDoubleValue(string);

     ...->DoSomething(value);
  }
}
```

Retrieve a G4double value from the **(string)** argument given to the command

Use the value in the way it is needed (e.g. pass it to other classes: opening angle for primary generator)
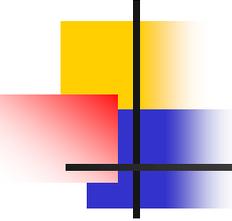
Summary

- **Interactive sessions** where user can give commands by keyboard can be used (from dumb terminals to graphic interfaces)

- A number of general-purpose commands are provided by Geant4, but **users can define more**, according to their needs → flexibility!