GEANT4 PYTHON INTERFACE

Koichi Murakami (KEK/CRC)

Geant4 Collaboration Meeting 2016

PYTHON

SHELL ENVIRONMENT

- □ CLI : *UI terminal*
- □ script language : *UI macro*

PROGRAMMING LANGUAGE

- □ oop, much easier than C++ : barrier to start
- □ multi-language binding (C-API)
- □ dynamic binding
    - □ modularization of software components
    - □ many third-party modules
    - □ software component bus (glue)

BRIDGING C++ TO PYTHON FOR GEANT4

generally easy, also different ways :

- □ boost::python, Py++, swing, pyrex/cython, ctypes, …

still some tricky parts exist in Geant4 :

- □ global object for singleton
- □ object life-time : depends on
- □ consider life-time of returned pointer : who has it?
- □ copy by value, reference of existing obj (potential danger)

boost::python :

“all you need to write is c++ code and there's no additional script, interface definition file, etc.”

PYTHON BRIDGE

IMPROVING FUNCTIONALITIES OF GEANT4 UI

- □ more powerful scripting environment
- □ flow control, variables, arithmetic operation

FLEXIBILITY IN THE CONFIGURATION OF USER APPLICATIONS

- □ Modularization of user classes with dynamic loading scheme
  - □ DetectorConstruction, PhysicsList, PrimaryGeneratorAction, UserAction-s
  - □ helping avoid code duplication.
- □ quick prototyping and testing

SOFTWARE COMPONENT BUS

- □ interconnectivity with many Python external module : analysis tools, e.g. ROOT, matplotlib
- □ middleware for application developers : GUI applications/web applications

RUNTIME PERFORMANCE

- □ Depends on usages : interpreter << thin wrapper

Geant4Py

"Geant4Py" was included in "environments/g4py/"

"Natural pythonization" of Geant4

- □ not specific to particular applications
- □ There are no invention of new conceptual ideas and terminologies!
  - □ same class names and their methods
  - □ *>>>gRunManager.BeamOn(10)*
- □ keeping compatibility with the current UI scheme
- □ exposing secure methods only
  - □ avoiding to expose "internal" methods

NOT all methods are exposed.

- □ only safe methods : getting object attributes, limited setter methods

Both Python2 and Python3 supported

Perspective for MT: thin wrapper might be possible.

EXPOSED CLASSES

Over 100 classes in different categories are exposed to Python.

CLASSES FOR GEANT4 MANAGERS

- □ G4RunManager, G4EventManager, …
- □ automatically instantiated as global variables
  - □ *gRunManager, gEventManager, …*

CLASSES OF BASE CLASSES OF USER ACTIONS

- □ G4UserDetetorConstruction, G4UserPhysicsList, G4UserXXXAction
  - □ PrimaryGenerator, Run, Event, Stepping,…
- □ can be inherited in the Python side

CLASSES HAVING INFORMATION TO BE ANALYZED

- □ G4Step, G4Track, G4StepPoint, G4ParticleDefinition, …
- □ Only safe methods are exposed.
  - □ Getting internal information are exposed. Some setter methods might be dangerous.

CLASSES FOR DESCRIBING USER INPUTS

- □ G4ParticleGun, G4Box, G4PVPlacement, …
- □ G4String, G4ThreeVector, G4RotationMatrix,… as utility classes

HOW TO EXPOSE

```
#include <boost/python.hpp>
#include "G4Step.hh"
using namespace boost::python;

void export_G4Step()
{
  class_<G4Step, G4Step*>("G4Step", "step class")
    .def("GetTrack",                 &G4Step::GetTrack,
         return_value_policy<reference_existing_object>())
    .def("GetPreStepPoint",          &G4Step::GetPreStepPoint,
         return_internal_reference<>())
    .def("GetPostStepPoint",         &G4Step::GetPostStepPoint,
         return_internal_reference<>())
    .def("GetTotalEnergyDeposit",    &G4Step::GetTotalEnergyDeposit)
    .def("GetStepLength",            &G4Step::GetStepLength)
    …
    .def("GetDeltaEnergy",           &G4Step::GetDeltaEnergy)
    ;
}
```

GEANT4PY MODULE STRUCTURE

PYTHON PACKAGE NAME :

- □ **Geant4** (should be geant4?)
- □ It consists of a collection of submodules same as Geant4 directory structure.
  - □ run/event/particle/geometry/track/…

```
#__init__.py
from G4global import *
from G4run import *
from G4event import *
…
```

FROM USERS SIDE,

- □ >>> from Geant4 import *
- □ ENV vars:
  - □ (DY)LD_LIBRARY_PARH is not necessary in most cases
  - □ PYTHON_PATH should be specified.

GLOBAL VARS/FUNCS, AUTO INSTANCE

Some global variables/functions starting with "***g***" are pre-instantiated at the importing time.

singleton objects :

- □ *gRunManager*
- □ *gEventManager*
- □ *gVisManager, …*

short cuts methods :

- □ *gControlExecute()*
- □ *gApplyUIcommand()*
- □ *gStartUISession()*

All of available visualization drivers (OpenGL, VRML, DAWN, ...) are automatically registered.

CO-WORK WITH G4UIMANAGER

Geant4Py provides a bridge to G4UImanager.

- □ keeping compatibility with current usability

UI COMMANDS

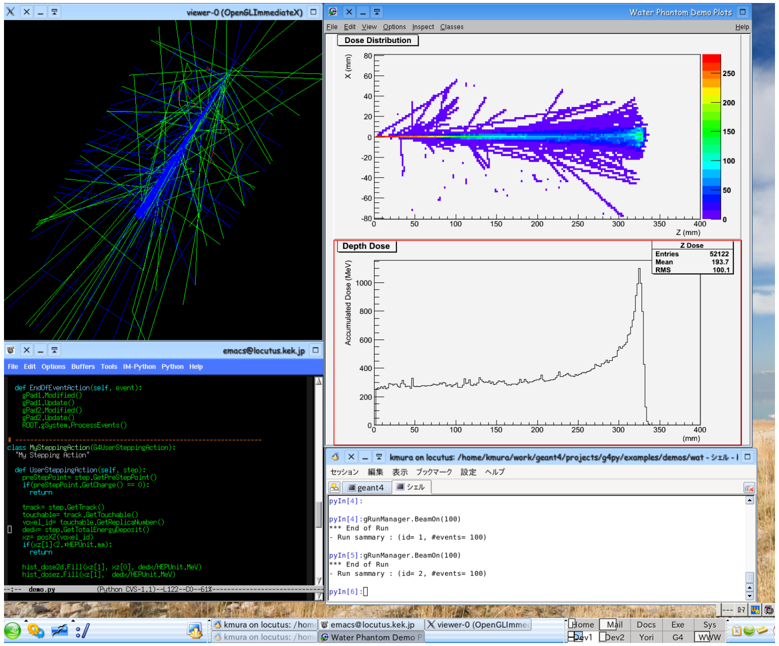
- □ some shortcuts are provided.
- □ gApplyUICommand(“/xxx/xxx”) allows to execute any G4UI commands.
- □ Current values can be obtained by gGetCurrentValues(“/xxx/xxx”).
- □ G4 macro files can be executed: gControlExecute(“macro_file_name”)

UI SESSION

G4 frontend shell can be used from Python.

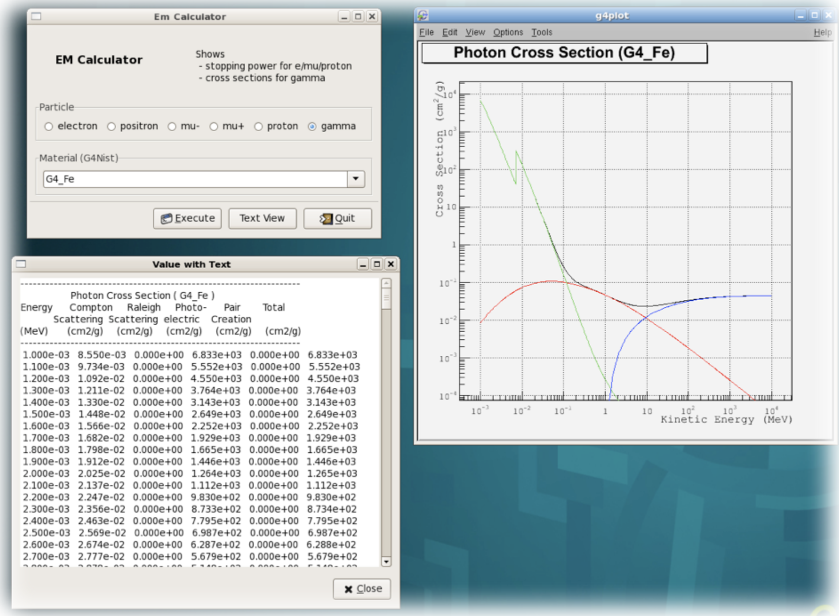
- □ gStartUISession() starts G4UIsession.
- □ g4py(Idle): // invoke a G4UI session
- □ when exit the session, go back to the Python front end

APPLICATION EXAMPLES


viewer-0 (OpenGLImmediateX)
Water Phantom Demo Plots
File Edit View Options Inspect Classes Help
Dose Distribution
X (mm)
Z (mm)
Depth Dose
Z Dose
Entries 52122
Mean 193.7
RMS 100.1
Accumulated Dose (MeV)
(mm)
emacs@locutus.kek.jp
File Edit Options Buffers Tools IM-Python Python Help
kmura on locutus: /home/kmura/work/geant4/projects/g4py/examples/demos/wat - シェル - I
セッション 編集 表示 ブックマーク 設定 ヘルプ
geant4
シェル
Home Mail Docs Exe Sys
Dev1 Dev2 Yori G4 WWW
Water Phantom Demo P

```
  def EndOfEventAction(self, event):
    gPad1.Modified()
    gPad1.Update()
    gPad2.Modified()
    gPad2.Update()
    ROOT.gSystem.ProcessEvents()

# ---------------------------------------------------------------
class MySteppingAction(G4UserSteppingAction):
  "My Stepping Action"

  def UserSteppingAction(self, step):
    preStepPoint= step.GetPreStepPoint()
    if(preStepPoint.GetCharge() == 0):
      return

    track= step.GetTrack()
    touchable= track.GetTouchable()
    voxel_id= touchable.GetReplicaNumber()
    dedx= step.GetTotalEnergyDeposit()
    xz= posXZ(voxel_id)
    if(xz[1]<2.*HEPUnit.mm):
      return

    hist_dose2d.Fill(xz[1], xz[0], dedx/HEPUnit.MeV)
    hist_dosez.Fill(xz[1],  dedx/HEPUnit.MeV)
--:--  demo.py          (Python CVS-1.1)--L122--C0--61%--------------
```

```
pyIn[4]:

pyIn[4]:gRunManager.BeamOn(100)
*** End of Run
- Run sammary : (id= 1, #events= 100)

pyIn[5]:gRunManager.BeamOn(100)
*** End of Run
- Run sammary : (id= 2, #events= 100)

pyIn[6]:
```



EM Calculator

Shows
- stopping power for e/mu/proton
- cross sections for gamma

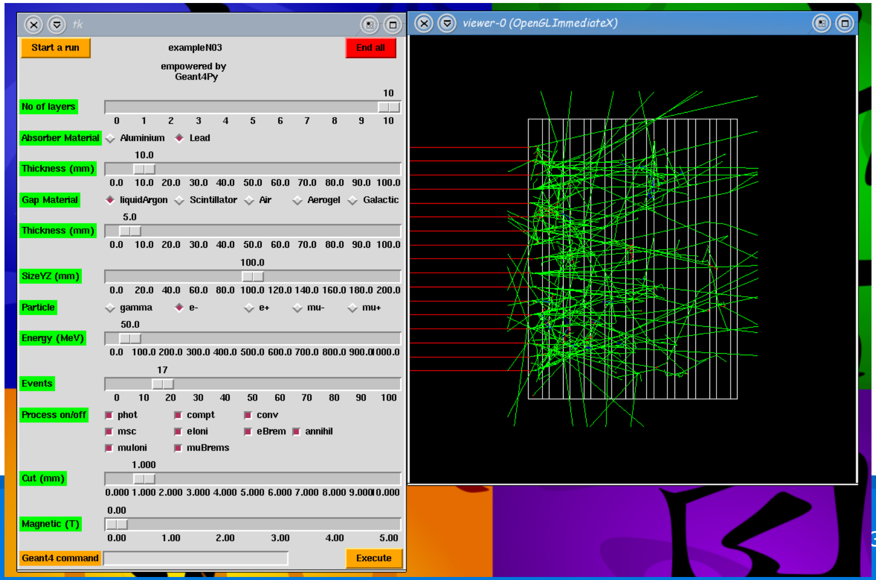
Particle
○ electron ○ positron ○ mu- ○ mu+ ○ proton ● gamma

Material (G4Nist)
G4_Fe

```
-----------------------------------------------------------------
              Photon Cross Section ( G4_Fe )
Energy     Compton    Raleigh    Photo-      Pair       Total
         Scattering  Scattering  electric   Creation
(MeV)      (cm2/g)    (cm2/g)    (cm2/g)    (cm2/g)    (cm2/g)
-----------------------------------------------------------------
1.000e-03  8.550e-03  0.000e+00  6.833e+03  0.000e+00  6.833e+03
1.100e-03  9.734e-03  0.000e+00  5.552e+03  0.000e+00  5.552e+03
1.200e-03  1.092e-02  0.000e+00  4.550e+03  0.000e+00  4.550e+03
1.300e-03  1.211e-02  0.000e+00  3.764e+03  0.000e+00  3.764e+03
1.400e-03  1.330e-02  0.000e+00  3.143e+03  0.000e+00  3.143e+03
1.500e-03  1.448e-02  0.000e+00  2.649e+03  0.000e+00  2.649e+03
1.600e-03  1.566e-02  0.000e+00  2.252e+03  0.000e+00  2.252e+03
1.700e-03  1.682e-02  0.000e+00  1.929e+03  0.000e+00  1.929e+03
1.800e-03  1.798e-02  0.000e+00  1.665e+03  0.000e+00  1.665e+03
1.900e-03  1.912e-02  0.000e+00  1.446e+03  0.000e+00  1.446e+03
2.000e-03  2.025e-02  0.000e+00  1.264e+03  0.000e+00  1.265e+03
2.100e-03  2.137e-02  0.000e+00  1.112e+03  0.000e+00  1.112e+03
2.200e-03  2.247e-02  0.000e+00  9.830e+02  0.000e+00  9.830e+02
2.300e-03  2.356e-02  0.000e+00  8.733e+02  0.000e+00  8.734e+02
2.400e-03  2.463e-02  0.000e+00  7.795e+02  0.000e+00  7.795e+02
2.500e-03  2.569e-02  0.000e+00  6.987e+02  0.000e+00  6.987e+02
2.600e-03  2.674e-02  0.000e+00  6.287e+02  0.000e+00  6.288e+02
2.700e-03  2.777e-02  0.000e+00  5.679e+02  0.000e+00  5.679e+02
```

EM Calculator :

- pyGTK is used.
- Show stopping power for e/mu/proton and cross section for gamma.
- Select a particle and select/set a NIST material.
- Show stopping power / cross sections on ROOT graph and text data



exampleN03

empowered by
Geant4Py

No of layers: 10

Absorber Material: ◇ Aluminium ◆ Lead

Thickness (mm): 10.0

Gap Material: ◆ liquidArgon ◇ Scintillator ◇ Air ◇ Aerogel ◇ Galactic

Thickness (mm): 5.0

SizeYZ (mm): 100.0

Particle: ◇ gamma ◆ e- ◇ e+ ◇ mu- ◇ mu+

Energy (MeV): 50.0

Events: 17

Process on/off: ☑ phot ☑ compt ☑ conv ☑ msc ☑ eIoni ☑ eBrem ☑ annihil ☑ muIoni ☑ muBrems

Cut (mm): 1.000

Magnetic (T): 0.00

GUI control panel for educational uses

DEVELOPMENT ASPECTS

CMake migration was done.

- □ installation is much easier than before

using the following find_packages :

- □ find_package(Geant4 REQUIRED)
- □ find_package(PythonInterp REQUIRED)
- □ find_package(PythonLibs REQUIRED)
- □ find_package(Boost)
- □ find_package(XercesC)
- □ find_package(ROOT)

Env. vars might be required for searching in non-default paths

- □ *GEANT4_INSTALL* as a path hint of find_package(Geant4)

RPATH is embedded in a module.

ToDo items : CTest item for system testing & unit tests scheme

Jupyter

Former IPython notebook

IPython is much more powerful frontend than python CLI.

IPython notebook works on web browser

- □ save session logs
- □ inline interactivity : plots, images, …
- □ familiar with github

Other external language (R, Julia, SQL,..) can be run on Jupyter.

Jupyter and Geant4

- □ Geant4Py can run on Jupyter as native Python script
- □ Alternative : Jupyter external kernel of Geant4 as other language support?

Idea : still open question

- □ A list of UI commands are defined as Jupyter kernel?
- □ Some shortcuts for global / static obj / funcs

PERSPECTIVES

load_ext g4 : how does it work?

- □ provides another UI terminal instance
- □ connecting to user applications
- □ retrieves a command set like GAG approach
- □ sever-client model : zeromq for distributed messaging
- □ not only local client but also cloud service
- □ cout/cin/cerr are redirected to Jupyter

UI command like shell

- □ command completion
- □ argument list
- □ command guides

Vis. component, still open

- □ showing images, interactivity,…

SUMMARY

Geant4Py is a python interface with Geant4.

- □ boost::python can exposer C++ classes
- □ can control Geant4 applications on Python
- □ can build application with Python
- □ from thin wrapper to scripting

Dev. aspects:

- □ CMake migration was done. unit testing fw is under consideration.
- □ Play with Jupyter : geant4py can work on Jupyter.
- □ Potentially, alternative UI session – shell (server-client)
- □ Implementing Jupyter external kernel as another language support
  - □ UI commands, some util. stuffs