Practical Usage of Geant4Py

Koichi Murakami
KEK / CRC

Summary Report

- Geant4Py presented by Koichi
- Users comment by Michel
- Hot discussions
  - Ana, John, John, Gabriele, Michel, Joseph, Fang, Witold, Takashi, Vladimir, Koichi, Hajime,

Contents

- Installation notes
- Exposed classes/methods in usecases
- Wrapping out your applications
- Connection to analysis tools
- Examples

Geant4 global shared library

- Shared libraries are required because of dynamic binding.
  - ✓ Any external libraries are also required to be built in shared libraries.
- Global libraries are required because Geant4Py does not know which granular libraries are used in your application.
- How to build library
  - ✓ You can co-work with "normal" granular static libraries.

```
# setenv G4BUILD_SHARED =1
# setenv G4TMP = G4INSTALL/tmp-slib
# setenv G4LIB = G4INSTALL/slib
# make global
```

  - ✓ *Once the library is build, these environment variables are NOT required any more in the Geant4Py side.*
- Don't forget to collect header files

```
# make includes
```

How to Build Geant4Py

- There is a configuration script for building the package.
  - ✓ configure --help shows more detailed options.

```
# ./configure linux
  --with-g4-incdir=/opt/heplib/Geant4/geant4.8.1/include
  --with-g4-libdir=/opt/heplib/Geant4/geant4.8.1/slib/Linux-g++
  --with-clhep-incdir=/opt/heplib/CLHEP/2.0.2.3/include
  --with-clhep-libdir=/opt/heplib/CLHEP/2.0.2.3/lib
  --with-clhep-lib=CLHEP-2.0.2.3
```

- Practical comments for CLHEP deployment
  - ✓ In case of both libXXX.a and libXXX.so existing, linker will link with the shared library.
    - » libCLHEP.a : link to libCLHEP-2.0.2.3.a
    - » libCLHEP.so -> remove it
    - » libCLHEP-2.0.2.3.so // use it in case of using shared library

- After executing configure script, you can go ahead to building procedures.

```
# make
# make install
```

- Set "PYTHONPATH" to the library path
  - ✓ `setenv PYTHONPATH ${PYTHONPATH}:"G4PY_LIBPATH":"ROOT_LIBPATH"`
- Let's try IPython
  - ✓ IPython enforces the Python front end!
    - » support *readline*, command completion
      - – Python words/classes/functions/variables
  - ✓ Let's use "`run xxx.py`" instead of "`execfile("xxx.py")`".
  - ✓ http://ipython.scipy.org/
  - ✓
- Importing module

```
>>> import Geant4 / from Geant4 import *
```

What is Exposed to Python

- Currently, over 100 classes over different categories are exposed to Python.
  - ✓ Classes for Geant4 managers
    - » G4RunManager, G4EventManager, …
  - ✓ UI classes
    - » G4UImanager, G4UIterminal, G4UIcommand, …
  - ✓ Utility classes
    - » G4String, G4ThreeVector, G4RotationMatrix, …
  - ✓ Classes of base classes of user actions
    - » G4UserDetetorConstruction, G4UserPhysicsList,
    - » G4UserXXXAction
      - – PrimaryGenerator, Run, Event, Stepping,…
    - » can be inherited in Python side
  - ✓ Classes having information to be analyzed
    - » G4Step, G4Track, G4StepPoint, G4ParticleDefinition, …
  - ✓ Classes for construction user inputs
    - » G4ParticleGun, G4Box, G4PVPlacement, …

- NOT all methods are exposed.
  - ✓ Only safe methods are exposed.
    - » Getting internal information are exposed.
    - » Some setter methods can easily break simulation results.

Global Variables/Functions

- Some global variables/functions starting with "g" are predefined;
  - ✓ Singleton objects / methods of singleton classes / static-like methods
  - ✓ Doubly instantiation is taken care. (Don't worry.)
  - ✓ All of available visualization drivers (OpenGL, VRML, DAWN, ...) are automatically registered.
  - ✓ defined in "`__init__.py`"

- gRunManager
- gEventManager
- gStackManager
- gTrackingManager
- gStateManager
- gTransportation Manager
- gParticleTable
- gProcessTable

- gNistManager
- gLossTableManager
- gProductionCutsTable
- gEmCalculator
- gVisManager
- gMaterialTable
- gElementTable

- gApplyUICommand()
- gGetCurrentValues()
- gStartUISession()
- gControlExecute()
- gCalculatePhoton CrossSection()
- gCalculateDEDX()

UI commands / UI session

- Geant4Py provides a bridge to G4UImanager.
  - ✓ Keeping compatibility with current usability

- UI Commands
  - ✓ `gApplyUICommand("/xxx/xxx")` allows to execute any G4UI commands.
  - ✓ Current values can be obtained by `gGetCurrentValues("/xxx/xxx")`.
- Existing G4 macro files can be reused.
  - ✓ `gControlExecute("macro_file_name")`
- Front end shell can be activated from Python
  - ✓ `gStartUISession()` starts G4UIsession.
    - » `g4py(Idle): //` invoke a G4UI session
    - » when exit the session, go back to the Python front end

Your Modules / Your Implementations

- Your own classes can be exposed, and create your own modules in the Boost-Python manner.

```
BOOST_PYTHON_MODULE(mymodule){
    class_<MyApplication>("MyApplication", "my application")
        .def("Configure", &MyApplication::Configure)
       ;
}
```

- Once an abstract class is exposed to Python, you can implement/override its derived class in the Python side.

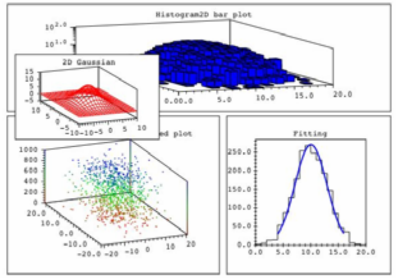
```
class MyRunAction(G4UserRunAction):
    """My Run Action"""
    def BeginOfRunAction(self, run):
        print "*** #event to be processed (BRA)=", \
        run.GetNumberOfEventToBeProcessed()
    def EndOfRunAction(self, run):
     print "*** run end run(ERA)=", run.GetRunID()
```

How to expose your applications

- ExN03 setup as an example
  - ✓ Each user component can be build as a Python module.
- Detector Construction
  - ✓ site-modules/geometries/ExN03geom/
- Physics List
  - ✓ site-modules/physics_lists/ExN03pl/
- Primary Generator Action as particle gun
  - ✓ site-modules/primaries/ParticleGun/
  - ✓ reusable in most cases

Connection to Analysis Tools

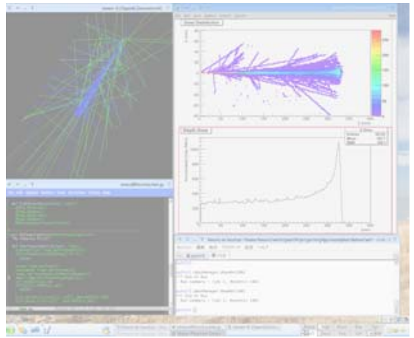
- **Analysis tools**
  - ✓ ROOT-Python interface
    - » plot example : examples/emplot/
    - » (online) histogram example:
      - – examples/demos/water_phantom/
    - » tree example:
      - – site-modules/utils/MCScore/
  - ✓ PAIDA
    - » AIDA Python implementation



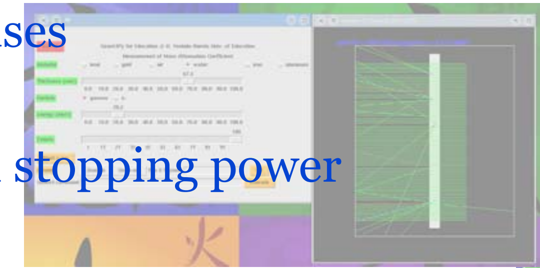
- **Plotting tools**
  - ✓ matplotlib
    - » histogramming interface (mathist) is in development.
      - – site-modules/utils/mathist

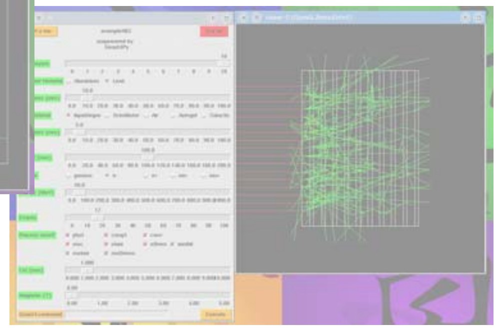
- "tests/" directory contains some basic exampls
  - ✓ test00-13: basic tests for Boost-Python
  - ✓ gtest01:
    - » an example of wrapping out users application
      - – Python module of users C++ library
    - » Python inheritances of users actions
    - » Python implementation of magnetic field
  - ✓ gtest02: test for using site-module packages
    - » fully scripting
    - » combination of predefined modules
  - ✓ gtest03: test for EZsim package
    - » geometry construction using EZgeom module
  - ✓ gtest04 : test for getting command tree and command information

more Example – in examples/

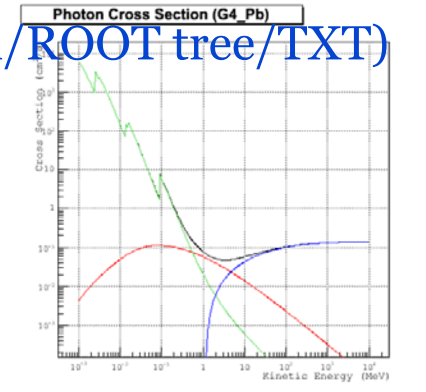


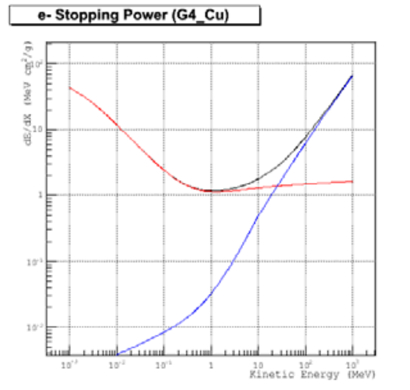
- demos/water_phantom/
  - ✓ This demo program shows that a Geant4 application coworks with ROOT on the Python front end.
- education/lesson1/2
  - ✓ GUI example for educational uses

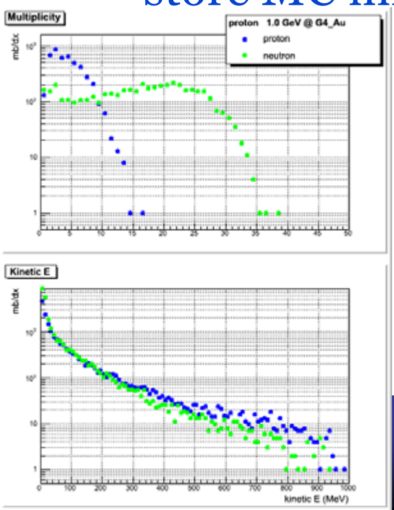




- emplot/
  - ✓ photon cross sections/electron stopping power
  - ✓ using EZgeom
- hadrontest/
  - ✓ proton/neutron/pion production
  - ✓ using EZgeom
  - ✓ store MC information (histogram/ROOT tree/TXT)







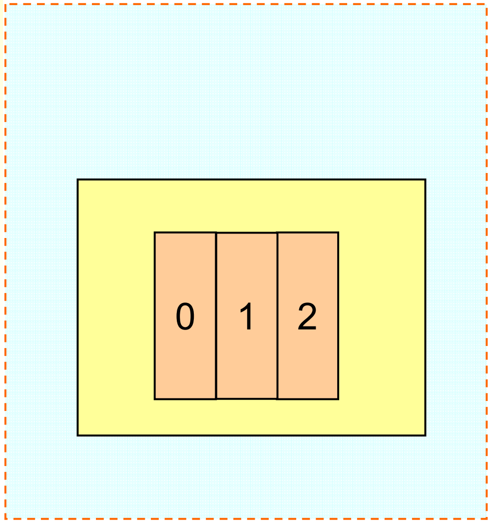
Yet Another Way to Create Geometry

- "EZgeom" module provides an easy way to create simple users geometries;
  - ✓ structure of geometry construction is hidden;
    - » Solid/Logical Volume/World Volume
    - » "EZvolume" is the only gateway to a physical volume from users side.
  - ✓ automatic creation of the world volume
    - » *volume size should be cared.*
  - ✓ creating CSG-solid volumes (Box, Tube, Sphere, ...)
  - ✓ changing volume materials
  - ✓ creating nested volumes
    - » placing a volume in the world by default
  - ✓ creating replicas / voxelizing BOX volumes
  - ✓ setting detector sensitivities
  - ✓ setting visualization attributes

Example of using EZgeom package

```
import NISTmaterials
from  EZsim import EZgeom
from EZsim.EZgeom import G4EzVolume

NISTmaterials.Construct()
# set DetectorConstruction to the RunManager
EZgeom.Construct()

# reset world material
air= gNistManager.FindOrBuildMaterial("G4_AIR")
EZgeom.SetWorldMaterial(air)

# dummy box
detector_box=G4EzVolume("DetectorBox")
detector_box.CreateBoxVolume(air, 20.*cm, 20.*cm, 40.*cm)
detector_box_pv=
detector_box.PlaceIt(G4ThreeVector(0.,0.,20.*cm))

# calorimeter placed inside the box
cal= G4EzVolume("Calorimeter")
nai= gNistManager.FindOrBuildMaterial("G4_SODIUM_IODIDE")
cal.CreateBoxVolume(nai, 5.*cm, 5.*cm, 30.*cm)
dd= 5.*cm
for ical in range(-1, 2):
  calPos= G4ThreeVector(dd*ical, 0., 0.)
  cal.PlaceIt(calPos, ical+1, detector_box)
```

less than 20 lines!!



What a user expects from a graphical interface ?

- a toolkit, easy to use, which allows him to built his own interactive Geant4 application
  - easy = by non-expert

remark 1 : an interactive application include necessarely visualization and analysis tool

remark 2 : the graphical interactive mode must be compatible with more 'classical' approach : commands line or batch

- Compatibility of libraries

remark 3 : the toolkit itself must be easy to install

What a user expects from a graphical interface ?

- a toolkit, easy to use, which allows him to built his own interactive Geant4 application
  - easy = by non-expert

remark 1 : an interactive application include necessarely visualization and analysis tool

remark 2 : the graphical interactive mode must be compatible with more 'classical' approach : commands line or batch

- Compatibility of libraries

remark 3 : the toolkit itself must be easy to install