Geant4 Python Interface

Koichi Murakami
KEK / CRC

boost C++ LIBRARIES

Geant 4

Introduction

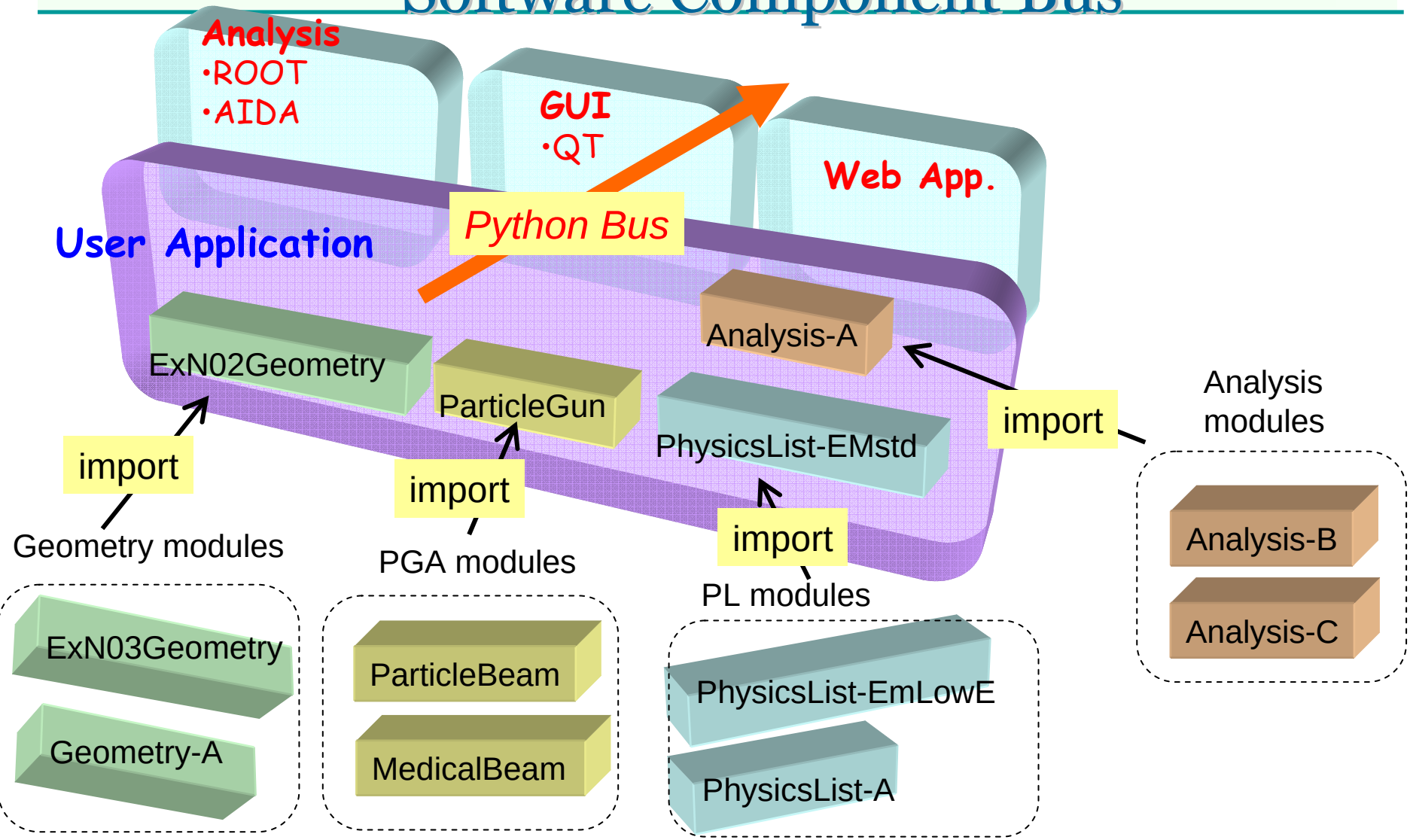
Python

- http://www.python.org/
- Shell Environment
  - ✓ front end shell
  - ✓ script language
- Programming Language
  - ✓ easy to program
    - » Scripting language is much easier than C++.
  - ✓ supporting Object-Oriented programming
  - ✓ providing multi-language binding (C-API)
  - ✓ dynamic binding
    - » modularization
    - » software component bus
- Runtime Performance
  - ✓ slower than compiled codes, but not so slow.
  - ✓ Performance could be tunable between speed and interactivity.

Motivation of Geant4-Python Bridge

- Improving functionalities of current Geant4 UI
  - ✓ more powerful scripting environment
    - » flow control, variables, arithmetic operation
  - ✓ direct object handling for G4XXX classes
    - » only limited “manager-like” classes can be exposed via G4UImessenger.
- Python is the most promising technological choice in terms of
  - ✓ A powerful scripting language
    - » Python can work as a interactive front end.
    - » Modularization of user classes with dynamic loading scheme
      - – *DetectorConstruction, PhysicsList, PrimaryGeneratorAction, UserAction-s*
      - – It helps avoid code duplication.
  - ✓ Software Component Bus
    - » C++ objects can be exposed to python.
    - » Interconnectivity with many Python external modules,
      - – analysis tools (ROOT/AIDA), web interface,...

Modular Approach and Software Component Bus



Technical Aspects

Conceptual Design of Geant4Py

- "*Natural Pythonization*" of Geant4
  - ✓ not specific to particular applications
  - ✓ There are no invention of new conceptual ideas and terminologies!
  - ✓ keeping compatibility with the current UI scheme
  - ✓ exposing secure methods only
    - » avoiding to expose "*kernel-internal-control*" methods
  - ✓ minimal dependencies of external packages
    - » only depending on *Boost-Python C++ Library*, which is a common, well-established and freely available library.
- Python package name : "**Geant4**"
  - ✓ From users side,

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

Name Policy in Python Side

- Names of classes as well as methods are same as used in Geant4.

```
>>> gRunManager= Geant4.G4RunManager()
>>> gRunManager.BeamOn(10)
```

  - ✓ This makes it easy to translate from C++ to Python, and vice versa.

- As an exception, pure singleton class, which has no public constructor, like G4UImanager, can not be exposed in Boost-Python. So, necessary members of such classes are exposed directly in the Geant4 namespace.

```
>>> Geant4.gApplyUIcommand("/run/beamOn")
>>> Geant4.gGetCurrentValues("/run/verbose")
>>> Geant4.gStartUISession()
```

  - » Each name is still similar to a corresponding method.

What is/isnot Exposed to Python

What is exposed:

- Classes for main Geant4 flow control
  - ✓ G4RunManager, G4UImanager, G4UIterminal
- Some 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, ...

What is not exposed:

- NOT all methods are exposed.
  - ✓ only safe methods (getting internal information) are exposed.
- Out of Scope
  - ✓ implementation of physics processes
  - ✓ implementation of internal control flows
  - ✓ It just ends in deterioration of performance.

Functionality

- Geometry Construction
  - ✓ CSG solids
  - ✓ Replication
  - ✓ EZgeometry
    - » CSG solids
    - » replication
    - » voxelization
  - ✓ Field
- Primary Generator Action
  - ✓ particle gun
  - ✓ beams
- Sensitive Detector
- Step/Track
- Physics List
- User Actions
  - ✓ SteppingAction
  - ✓ RunAction
  - ✓ EventAction
- UI
  - ✓ UI commands
  - ✓ UI session
- Particle
- Process
- RunManager
- Static Tables

Global Variables/Functions

- Some global variables/functions starting with "g“ are predefined.
  - ✓ Singleton objects / methods of pure singleton classes / static member functions;

    - gRunManager
    - gEventManager
    - gStackManager
    - gTrackingManager
    - gStateManager
    - gTransportationManager
    - gParticleTable
    - gProcessTable
    - gNistManager
    - gVisManager
    - gMaterialTable
    - gElementTable

    - gApplyUICommand()
    - gGetCurrentValues()
    - gStartUISession()
    - gControlExecute()

Bridge to G4UImanager

- 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.

Site-module package

- We will also provide site-module package as pre-defined components.
  - ✓ Material
    - » sets of pre-defined materials
      - – NIST materials via G4NistManager
  - ✓ Geometry
    - » "exN03" geometry as pre-defined geometry
    - » *"EZgeometry"*
      - – provides functionalities for easy geometry setup (applicable to target experiments)
  - ✓ Physics List
    - » pre-defined physics lists
    - » (easy access to cross sections, stopping powers, ... via *G4EmCalculator)*
  - ✓ Primary Generator Action
    - » particle gun / particle beam
  - ✓ Sensitive Detector
    - » calorimeter type / tracker type
- They can be used just by importing modules.
- They can be combined and connected to higher application layers (Analysis / GUI components).

Various Levels of Pythonization

- Various level of pythonized application can be realized.
  - ✓ It is completely up to users!
- There are two metrics;
  - ✓ Execution Speed
    - » just wrapping current existing applications
  - ✓ Interactivity
    - » interactive analysis
    - » rapid prototyping
    - » educational use
- Optimized point depends on what you want to do in Python.
  - ✓ no performance loss in case of object controller
  - ✓ pay performance penalty to interpretation in stepping actions.

Use-case of Pythonization

modularized "compiled" components.
- detector construction
- physics list
- user actions
, and handling components by scripting

Execution Speed

Productions changing conditions
- physics validation/verification

Coworking with other software components
- interactive analysis

compiled modules and scripting user actions
- filling histograms

Fully scripting
- rapid prototyping
- educational uses

using "predefined" module and scripting

Interactivity/ Pythonization

Performance can be tunable between speed and interactivity.

Practical Aspects

Software Requirements

- All libraries should be compiled in shared libraries.
- Python
- BOOST-Python
  - ✓ *1.32, latest*
- Geant4
  - ✓ *7.0 or later*
  - ✓ should be built in "global" and "shared" libraries.
  - ✓ All header files should be collected into $(G4INSTALL)/include by "make includes"
- CLHEP
  - ✓ *1.9.1.1 or later*
  - ✓ building shared objects is supported since version 1.9.
- Platforms
  - ✓ Linux system is ok.
  - ✓ Mac OSX has some troubles with
    - » Boost-python
    - » Creating shared library of CLHEP1.9.xx
  - ✓ Win32-Cygwin is bad idea. Win32-VC could be better.

How to Build Geant4Py

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

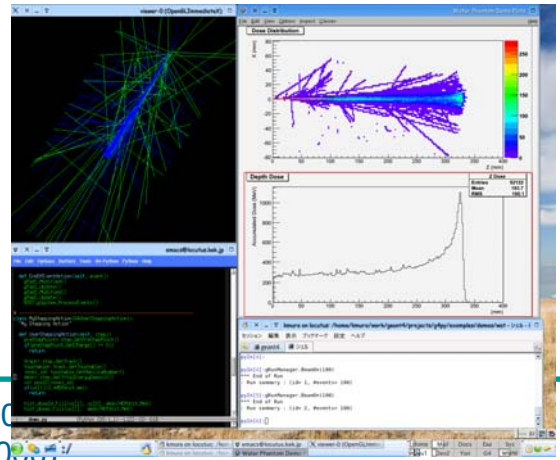
```
# ./configure linux
  --with-g4-incdir=/opt/heplib/Geant4/geant4.8.0/include
  --with-g4-libdir=/opt/heplib/Geant4/geant4.8.0/slib/Linux-g++
  --with-clhep-incdir=/opt/heplib/CLHEP/1.9.2.2/include
  --with-clhep-libdir=/opt/heplib/CLHEP/1.9.2.2/lib
  --with-clhep-lib=CLHEP-1.9.2.2
```

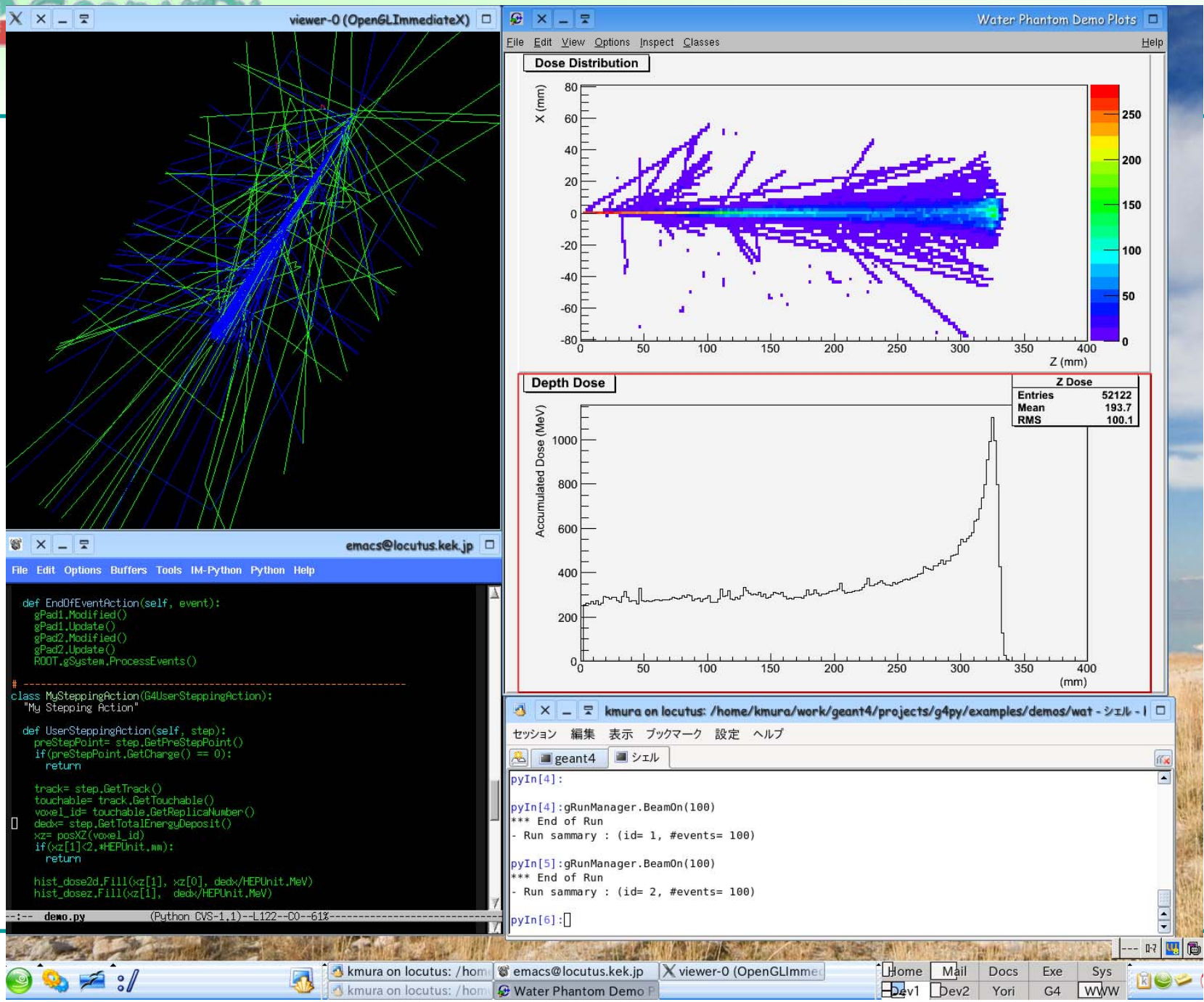
- After executing configure script, you can go ahead to building procedures. Just type

```
# make
# make install
```

A Medical Application Example

- Several examples of using Python interface are/will be presented.
- An example of “water phantom dosimetry”
  - ✓ *This demo program shows that a Geant4 application well coworks with ROOT on the Python front end.*
- You can look features of;
  - ✓ dose calculation in a water phantom
  - ✓ Python implementation of sensitive detector
  - ✓ Python overloading of user actions
  - ✓ on-line histogramming with ROOT
  - ✓ visualization




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
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%
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
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]:
kmura on locutus: /hom
emacs@locutus.kek.jp
viewer-0 (OpenGLImme
kmura on locutus: /hom
Water Phantom Demo P
Home Mail Docs Exe Sys
Dev1 Dev2 Yori G4 WWW

Example of A Python Script

```
from Geant4 import *
import demo_wp    # module of a user G4 application
import ROOT

#  ------------------------------------------------
class ScoreSD(G4VSensitiveDetector):
  "SD for score voxels“
  def __init__(self):
    G4VSensitiveDetector.__init__(self, "ScoreVoxel")
  def ProcessHits(self, step, rohist):
    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)
    hist_dose2d.Fill(xz[1], xz[0], dedx/MeV)
    hist_dosez.Fill(xz[1],  dedx/MeV)
```

```
# user detector construction (C++)
myDC= demo_wp.MyDetectorConstruction()
gRunManager.SetUserInitialization(myDC)

# user physics list (C++)
myPL= demo_wp.MyPhysicsList()
gRunManager.SetUserInitialization(myPL)

# user P.G.A (Python)
myPGA= MyPrimaryGeneratorAction()
gRunManager.SetUserAction(myPGA)

# setting particle gun
pg= myPGA.particleGun
pg.SetParticleByName("proton")
pg.SetParticleEnergy(230.*MeV)
pg.SetParticleMomentumDirection(G4ThreeVector(0., 0., 1.))
pg.SetParticlePosition(G4ThreeVector(0.,0.,-20.)*cm)

gRunManager.Initialize()

# set SD (A SD should be set after geometry construction)
scoreSD= ScoreSD()
myDC.SetSDtoScoreVoxel(scoreSD)

gApplyUICommand("/control/execute vis.mac")
gRunManager.BeamOn(100)
```

Resources

- Project Home Page
  - ✓ http://www-geant4.kek.jp/projects/Geant4Py/
- Anonymous CVS Access

```
# cvs -d :pserver:anonymous@geant4.kek.jp:/home/projects/Geant4Py/CVS login
# cvs -z3 -d :pserver:anonymous@geant4.kek.jp:/home/projects/Geant4Py/CVS co g4
# cvs -d :pserver:anonymous@geant4.kek.jp:/home/projects/Geant4Py/CVS logout
```

- CVS view
  - ✓ http://www-geant4.kek.jp/projects/Geant4Py/cvs/
- Forum
  - ✓ Forums for developer and users
  - ✓ http://www-geant4.kek.jp/forum/

Summary

- Python Interface of Geant4 (Geant4Py) has been well designed and implementation is now rapidly on-going.

- Python as a powerful scripting language
  - ✓ much better interactivity
    - » configuration
    - » rapid prototyping
- Python as “Software Component Bus”
  - ✓ interconnectivity with various kind of software components.
    - » histogramming with ROOT
    - » system integration

- We have a plan to commit the beta release into the future release.
  - ✓ “environments/” directory is a suitable position