Geant4Py

Geant4 Python Interface

Koichi Murakami

KEK / CRC

boost C++ LIBRARIES

Geant4

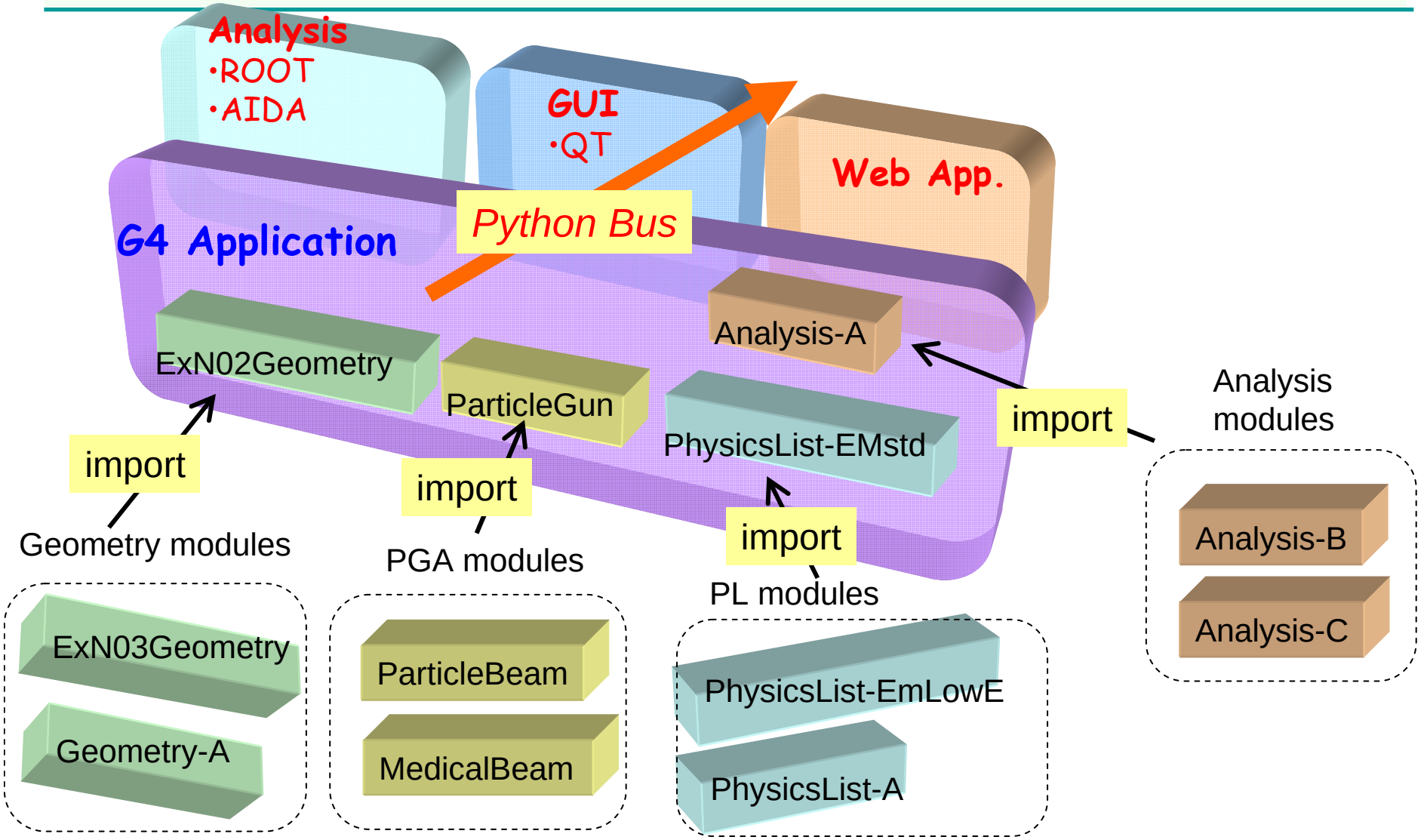
Python

- 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 can be tunable between speed and interactivity.

Motivation of Geant4-Python Bridge

- Missing functionalities of current Geant4 UI
  - ✓ more powerful scripting environment
    - » flow control, variables, arithmetic operation
  - ✓ direct object handling of G4XXX classes
    - » G4UImessenger can do something, but limited …
- Python is most promising 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
    - » Interconnectivity with many Python external modules,
      - – analysis tools (ROOT/AIDA), web interface,…

Modular Approach and Component Bus



Use-cases 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

✓ Various level of pythonized application can be realized.
✓ Performance can be tunable between speed and interactivity.

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
    - » NOT exposing "*kernel-internal*" methods
      - – "set"-methods which are internally used
  - ✓ minimal dependencies of external packages
    - » only depending on *Boost-Python C++ Library*
      - – a common, well-established and freely available library.

- Geant4Py is neither application nor application framework.
  - ✓ just providing a mechanism for exposing G4 classes
  - ✓ Users have to take care of their applications by themselves.

What is/isnot Exposed

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
  - ✓ CSG solids
  - ✓ Replica
  - ✓ EZgeometry
    - » CSG solids
    - » replication
    - » voxelization
  - ✓ Field
  - ✓ Touchable
- Material
  - ✓ material table
  - ✓ NIST materials
- Track
  - ✓ Track
  - ✓ Step
  - ✓ StepPoint
  - ✓ ...
- Particle
  - ✓ particle table
  - ✓ particle definition
  - ✓ dynamic particle
  - ✓ primary particle/vertex
- Process
  - ✓ process table
- Global
  - ✓ ThreeVector/Rotation
  - ✓ Random Generator
  - ✓ G4String
  - ✓ State Manager
- Run/Event
  - ✓ Run/Event
  - ✓ Run/Event Manager
- User Physics List
- Primary Generator Action
  - ✓ particle gun
- User Actions
  - ✓ SteppingAction
  - ✓ RunAction
  - ✓ EventAction
  - ✓ ...
- Sensitive Detector
- UI
  - ✓ UI commands
  - ✓ UI session
- Visualization

Expose with Boost-Python

```
#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("GetDeltaPosition",        &G4Step::GetDeltaPosition)
    .def("GetDeltaTime",            &G4Step::GetDeltaTime)
    .def("GetDeltaMomentum",        &G4Step::GetDeltaMomentum)
    .def("GetDeltaEnergy",          &G4Step::GetDeltaEnergy)
    ;
}
```

Name Policy and Global Objects

- Names of classes as well as methods are same as used in Geant4.
  - ✓ This makes it easy to translate from C++ to Python, and vice versa.

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

- Some global variables/functions starting with "**g**" are predefined.
  - ✓ Singleton objects / methods of pure singleton classes
    - » `gRunManager`
    - » `gVisManager`
    - » `gParticleTable`
    - » `gApplyUIcommand()`
    - » `gStartUISession()`
    - » `...`
  - ✓ *gRunManager* and *gVisManager* are taken care not so as to be doubly instantiated, so that users do not have to take any more care about the timing of object instantiation in python side.
  - ✓ All of visualization drivers (OpenGL, VRML, DAWN, ...) are automatically registered. So users are now free from implementation of VisManager.

Geant4Py Module Structure

- Python package name : "**Geant4**"
  - ✓ It consists of a collection of modules same as Geant4 directory structure.
    - » event / graphics_reps / particles / run / geometry / interface / processes / track / visualization / digits_hits / global / materials / tracking
  - ✓ including CLHEP components typedef-ed as G4XXX
    - » G4ThreeVector, G4RotationMatrix, ...
    - » Units definition (mm, cm, kg, ...)

- From users side,

```
>>> import Geant4  (from Geant4 import *)
*************************************************************
 Geant4 version Name: geant4-07-01-patch-01    (25-October-2005)
                      Copyright : Geant4 Collaboration
                      Reference : NIM A 506 (2003), 250-303
                            WWW : http://cern.ch/geant4
*************************************************************
Visualization Manager initialising...
Registering graphics systems...

>>>
```

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.
  - ✓ come back to Python front end when UIsession is terminated.

Trace of Geant4 Version

- "G4VERSION_NUMBER" will be introduced in `global/management/include/version.hh` (8.0) for identifying the each G4 version number.
  - ✓ "setenv G4VERSION_NUMBER" is required for versions (7.0/7.0.p1/7.1/7.1.p1).

```
// Numbering rule for "G4VERSION_NUMBER":
// - The number is consecutive (i.e. 711) as an integer.
// - The meaning of each digit is as follows;
//
//   711
//   |--> major version number
//    |--> minor version number
//     |--> patch number

#ifndef G4VERSION_NUMBER
#define G4VERSION_NUMBER  711
#endif

#ifndef G4VERSION_TAG
#define G4VERSION_TAG "$Name:  $"
#endif

static const G4String G4Version = "$Name:  $";
static const G4String G4Date    = "(25-Oct-2005)";
```

Site-module Package

- We will also provide site-module package as pre-defined components.
  - ✓ Material
    - » pre-defined materials
      - – NIST materials
  - ✓ Geometry
    - » "exN03" geometry as pre-defined geometry
    - » *"EZgeometry"*
      - – provides functionalities for easy geometry setup (applicable to target experiments)
  - ✓ Physics List
    - » pre-defined physics lists
  - ✓ Primary Generator Action
    - » particle gun / particle beam
  - ✓ Sensitive Detector
    - » calorimeter type / tracker type
- They can be used just by importing modules.

Software Requirements

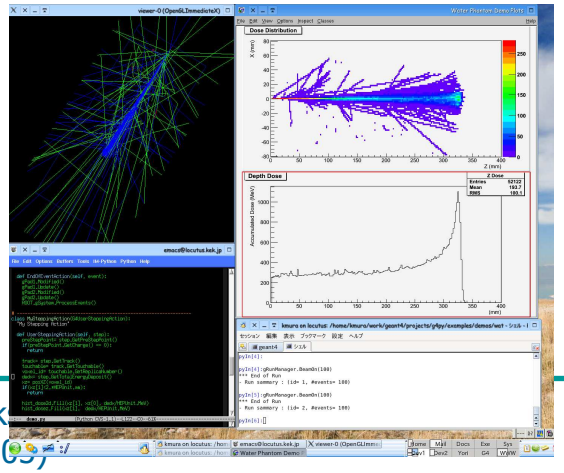
- All libraries should be compiled in shared libraries.
- Python
- BOOST-Python
  - ✓ *1.32, latest*
- Geant4
  - ✓ *7.0 or later*
  - ✓ should be built in "global" and "shared" libraries.
  - ✓ All header files should be collected into $(G4INSTALL)/include by "make includes"
- CLHEP
  - ✓ *1.9.1.1 or later*
  - ✓ building shared objects is supported since version 1.9.
- Platforms
  - ✓ Linux system is ok.
    - » SUSE Linux 9.3 is a development environment.
      - – It is the easiest way to go, because Boost C++ library is preinstalled.
    - » Scientific Linux (SL3/SL4) is checked for well working.
  - ✓ 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.

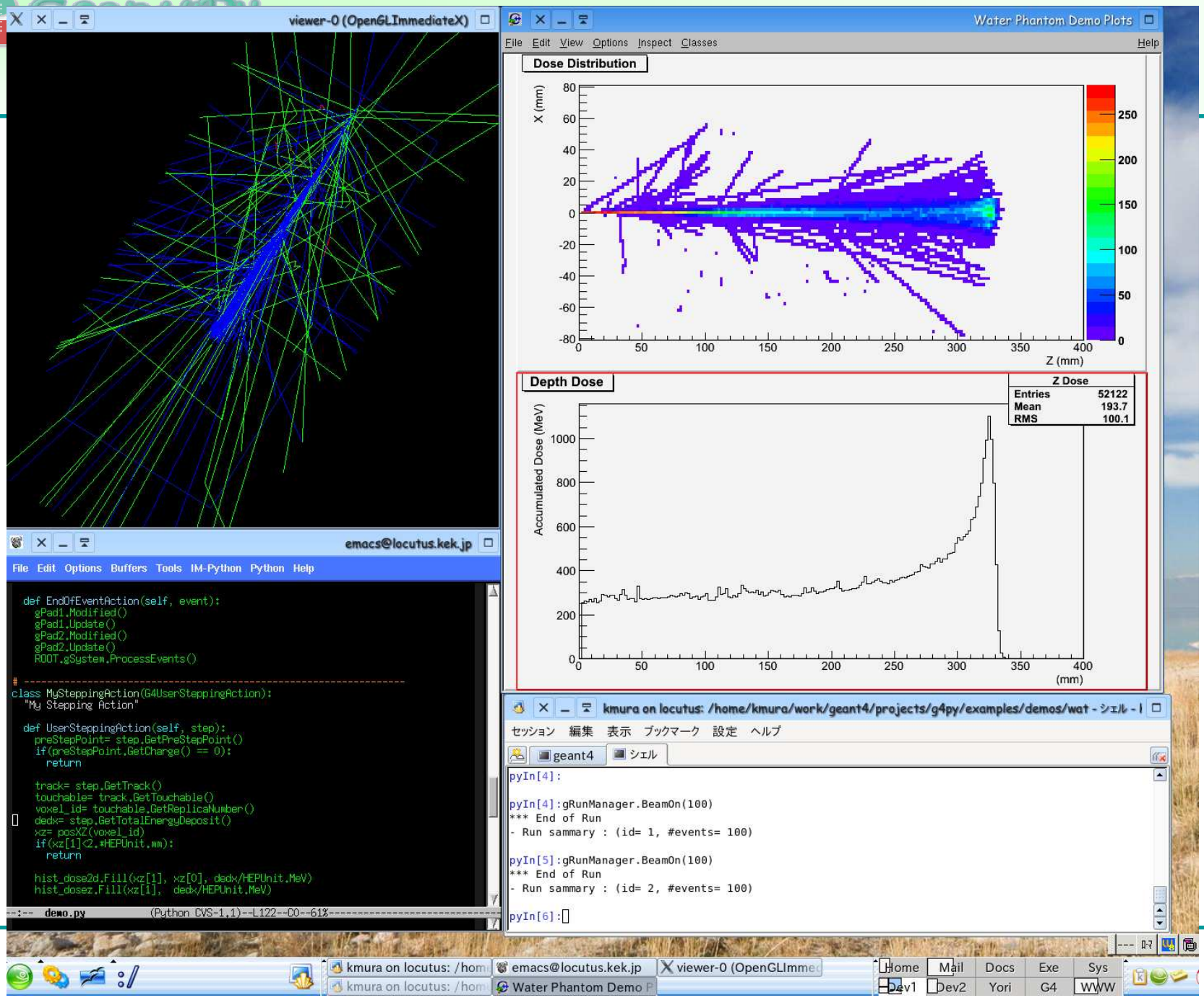
Some Comments

- Some C++ has no Python equivalent.
  - ✓ pointer arithmetic
    - » pointer vs. reference
    - » pointer/vector – list/tuple conversion
    - » These treatments cannot be automated.
  - ✓ memory management
- Memory management is different between C++ and Python.
  - ✓ In C++, object life span should be controlled manually.
    - » new / delete
  - ✓ In Python, objects will be deleted automatically when they have zero reference counts.
  - ✓ Take care of object life time!
    - » To keep objects beyond functions, they should be global.
- Problem with termination time
  - ✓ At termination time, Python session ends with segmentation fault because object deletions are in chaos.
  - ✓ Please be patient for a while . Don't worry, nothing is loosed.

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

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()
    ...

class MyPrimaryGeneratorAction(G4VUserPrimaryGeneratorAction):
  def __init__(self):
    G4VUserPrimaryGeneratorAction.__init__(self)
    self.particleGun= G4ParticleGun(1)

  def GeneratePrimaries(self, event):
    self.particleGun.GeneratePrimaryVertex(event)
```

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

# 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/
- 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 next December release.
  - ✓ “environments/” directory is a suitable position