

Parallel Session Interfaces

- This session is intended to be a working session with a couple of introductory presentations
- “Geant4 Web Application” by Hajime Yoshida
- “Python Interface, Geant4 Education and Geant4 on Web” by Koichi Murakami
- Working

Geant4 Web Application

Its Design and Use

Hajime Yoshida

Naruto U.E.

12th Geant4 Collaboration Workshop

Hebden House

17 September 2007

Existing GUIs

- MOMO
 - Java-based stand-alone packages
 - Independent processes (Geant4 and JVM)
- Geant4Py
 - Python-based wrapper
 - Built-in GUI packages
 - Tkinter (Python + Tk)
 - New example by Jean and Michel; TestEm0
 - wxPython ‘Python + wxWidgets)

Geant4 Web Application

- Some use cases of Geant4 must be transparent from all difficult features of its implementation, deployment, maintenance etc.
 - Learning physics of particle interactions with Geant4
 - Providing basic features of Geant4 thru its execution etc.
- There are cases where CPU time are little and are well suited for atomic transaction model of HTTP or REST

Design issues

- Choice of Web frameworks: server + script + database
 - LAMP (linux, apache, mysql, php/perl/python)
 - Rails on Ruby is a la mode
 - Python-based frameworks
 - Zope, Django, Turbogears, Pylons etc.
- Good implementation of MVC model
 - Model = database or Geant4
 - Controller = web server
 - View = template (+ AJAX)
- Our preliminary choice was Turbogears : will change

A working example

- Only for demonstration
 - <http://erluna.naruto-u.ac.jp:80/>
 - A part of the preliminary prototype for “Geant4 for Education” project.
 - Course materials are stored in DB.
 - “Virtual lab.”
 - TestEm0
 - Queuing, using SimPy, a discrete system simulation

Model

- Model which create and store data
 - Geant4 data
 - Numerical output : G4cout or file
 - Visualization output files : heprep, vrml, dawn
 - Analysis output files : root
 - Database
 - User registration
 - Documents
 - Etc.

```
#!/usr/bin/python
# python script for TestEm0 python version
from Geant4 import *
import TestEm0
import sys
import os
myDC= TestEm0.DetectorConstruction()
gRunManager.SetUserInitialization(myDC)
myPL= TestEm0.PhysicsList()
gRunManager.SetUserInitialization(myPL)
# set user actions...
myPGA= TestEm0.PrimaryGeneratorAction(myDC)
gRunManager.SetUserAction(myPGA)
myRA= TestEm0.RunAction(myDC,myPGA)
# set user action classes
gRunManager.SetUserAction(myRA)
gRunManager.Initialize()
pg = G4ParticleGun()
materialList = TestEm0.getMaterialTable();
particleList = TestEm0.getParticleTable()
```

```
def cmd_beamOn(particle='proton', material='Water', energy='1 MeV', cuts='1 um')
:
# now GUI actions and I switch sys.stdout to a file
SetG4PyCoutDestination()
old_stdout = sys.stdout
sys.stdout = open('TestEm0.out', 'w')
gApplyUICommand("/gun/particle " + particle)
gApplyUICommand("/testem/det/setMat " + material)
gApplyUICommand("/gun/energy " + energy)
gApplyUICommand("/testem/phys/setCuts " + cuts)
gRunManager.BeamOn(1)
# I recover my original sys.stdout
ResetG4PyCoutDestination()
sys.stdout.close()
sys.stdout=old_stdout
if __name__ == '__main__':
    cmd_beamOn()
```

Control and View

- Controllers: load balancing etc.
 - Atomic transaction
 - HTTP session < 30 min.
 - Long Geant4 session
- View :
 - Template
 - Ajax

Discussions

- What we can present with this new interface
 - Target users
 - Educational courses
 - Materials to be presented
 - Already existing examples and tests
 - And new ones?
- Discussion and Work