How to build/run user applications

山下智弘
JST CREST/神戸大学

Borrowing especially from presentations of M. Asai(SLAC)

Geant4 Tutorial @ Japan 2007

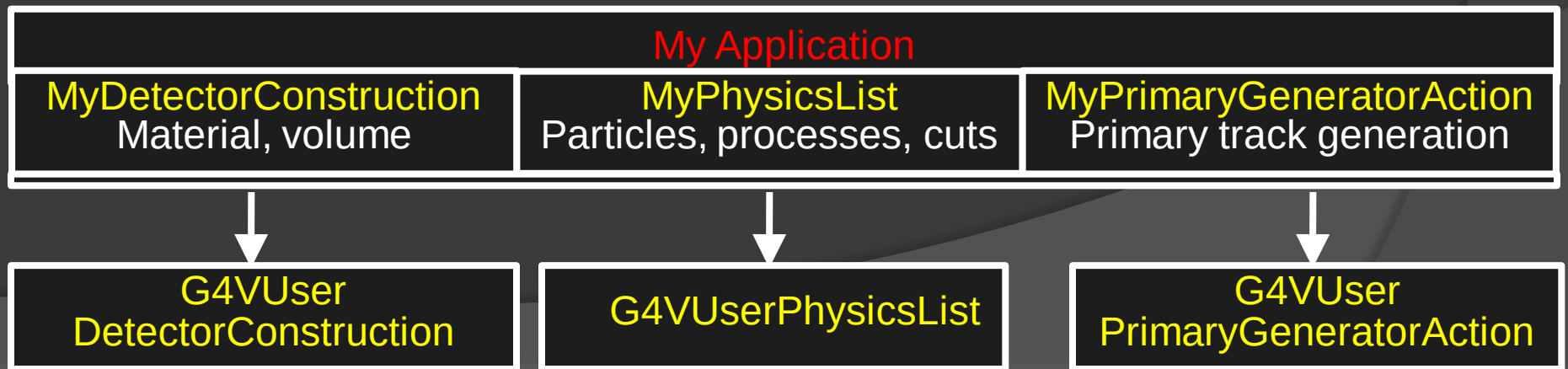
17-19 Oct, 2007 @ RCNS, based on Geant4 9.0.p01

Outline

I. Mandatory user classes
II. User actions
III. How to write main()
IV. How to configure/build user applications
V. How to run user applications
VI. UI commands and macro
VII. How to visualize

I. Mandatory user classes

- Geant4 is a toolkit. You have to build an application.
- To make an application, you have to
  1. Define your geometrical setup
     G4VUserDetectorConstruction - Material, volume
  2. Define physics to get involved
     G4VUserPhysicsList - Particles, physics processes/models
     Production thresholds
  3. Define how an event starts
     G4VUserPrimaryGeneratorAction - Primary track generation



I. Mandatory user classes
1. DetectorConstruction

- Derive your own concrete class from G4VUserDettectorConstruction abstract base class.
  - e.g. class MyDetectorConstruction : public G4VUserDetectorConstruction
- In the virtual method Construct() of your UserDetectorConstruction
  - Define all necessary materials
  - Define volumes of your detector geometry
- Optionally you can define
  - your sensitive detector classes and set them to the corresponding logical volumes
  - Regions for any part of your detector
  - Visualization attributes (color, visibility, etc.) of your detector elements

I. Mandatory user classes
2. PhysicsList

- Geant4 does not have any default particles or processes.
  - Even for the particle transportation, you have to define it explicitly.
- Derive your own concrete class from G4VUserPhysicsList abstract base class.
  - Define all necessary particles
  - Define all necessary processes and assign them to proper particles
  - Define cut-off ranges applied to the world (and each region)
- Geant4 provides lots of utility classes/methods and examples.
  - "Educated guess" physics lists for defining hadronic processes for various use-cases.

I. Mandatory user classes
3. PrimaryGeneratorAction

- Derive your concrete class from G4VUserPrimaryGeneratorAction abstract base class.
- Pass a G4Event object to one or more primary generator concrete class objects which generate primary vertices and primary particles.
- Geant4 provides several generators in addition to the G4VPrimaryGenerator base class.
  - G4ParticleGun
  - G4HEPEvtInterface, G4HepMCInterface
    - Interface to /hepevt/ common block or HepMC class
  - G4GeneralParticleSource
    - Define radioactivity

II. User actions(1)

- 5 user action classes
- methods of which are invoked during “Beam On”
  - G4UserRunAction
  - G4UserEventtAction
  - G4UserStackingAction
  - G4UserTrackingAction
  - G4UserSteppingAction

II. User actions(2)

- G4UserRunAction
  - G4Run* GenerateRun()
    - Instantiate user-customized run object
  - void BeginOfRunAction(const G4Run*)
    - Define histograms, TTree
  - void EndOfRunAction(const G4Run*)
    - Store histograms, TTree
- G4UserEventAction
  - void BeginOfEventAction(const G4Event*)
    - Event selection
  - void EndOfEventAction(const G4Event*)
    - Analyze the event

II. User actions(3)

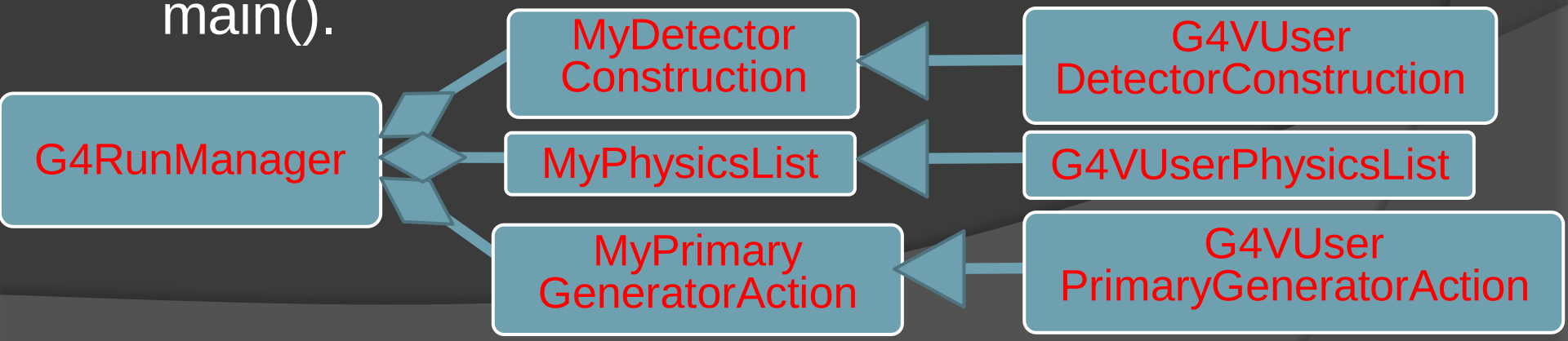
- G4UserStackingAction
  - void PrepareNewEvent()
    - Reset priority control
  - G4ClassificationOfNewTrack ClassifyNewTrack(const G4Track*)
    - Invoked every time a new track is pushed
    - Classify a new track - priority control
    - Urgent, Waiting, PostponeToNextEvent, Kill
  - void NewStage()
    - Invoked when the Urgent stack becomes empty
    - Change the classification criteria
    - Event filtering (Event abortion)

II. User actions(4)

- G4UserTrackingAction
  - void PreUserTrackingAction(const G4Track*)
    - Decide trajectory should be stored or not
    - Create user-defined trajectory
  - void PostUserTrackingAction(const G4Track*)
- G4UserSteppingAction
  - void UserSteppingAction(const G4Step*)
    - Kill / suspend / postpone the track
    - Draw the step (for a track not to be stored as a trajectory)

III. How to write main

- Geant4 does not provide the main()
- In your main(), you have to
  - Construct G4RunManager (or your derived class)
  - Set user mandatory classes to RunManager
    - Use G4RunManager::SetUserInitialization() to set
      - G4VUserDetectorConstruction
      - G4VUserPhysicsList
    - Use G4RunManager:::SetUserAction() to set
      - G4VUserPrimaryGeneratorAction
      - And optional user action classes
- You can define VisManager, (G)UI session in your main().



III. How to write main(2) example

```
int main(int argc, char** argv)
{
    G4RunManager* runManager = new G4RunManager;

    G4VUserDetectorConstruction* detector =
            new MyDetectorConstruction();
    runManager->SetUserInitialization(detector);
    //
    G4VUserPhysicsList* physics =
            new HadrontherapyPhysicsList;
    runManager->SetUserInitialization(physics);

    G4VUserPrimaryGeneratorAction* gen_action =
            new MyPrimaryGeneratorAction();
    runManager->SetUserAction(gen_action);
...
```

Construct G4RunManager

Construct mandatory initialization classes

Set mandatory initialization classes to G4RunManager

Construct mandatory action class and set to G4RunManager

III. How to write main(2) example *cotnd.*

```
…
   G4UserRunAction* run_action = new MyRunAction;
   runManager->SetUserAction(run_action);

   runManager->Initialize();

   if (argc != 1)  {    // batch mode
      G4UImanager * UI = G4UImanager::GetUIpointer();
      G4String command = "/control/execute ";
      G4String fileName = argv[1];
      UI->ApplyCommand(command+fileName);
    } else  {            // interactive mode : define UI terminal
       G4UIsession * session = new G4UIterminal(new G4UItcsh);
       session->SessionStart();
       delete session;
     }

   delete runManager;
   return 0;
}
```

Construct non-mandatory action class and set to G4RunManager

Initialize G4 kernel

Start a run

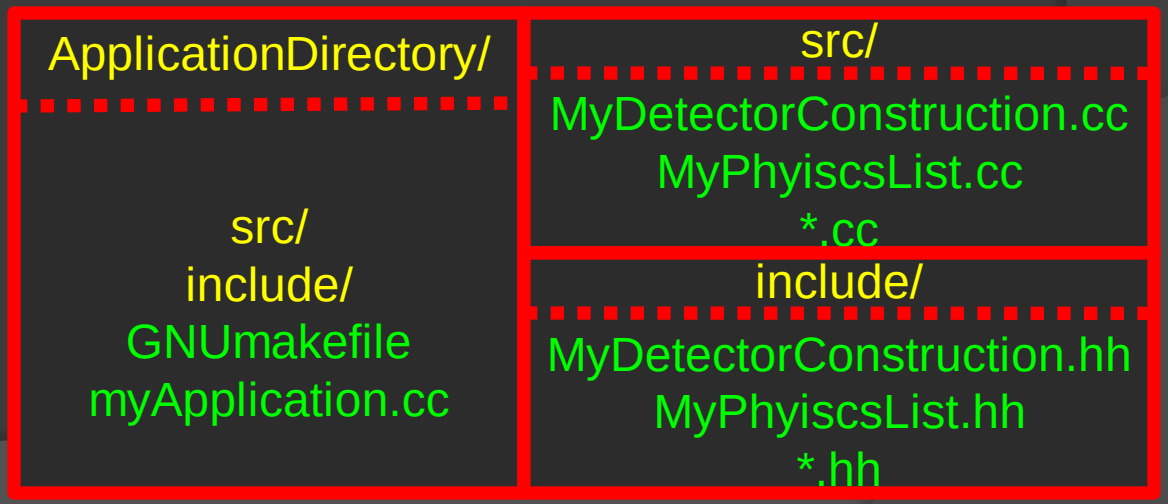
Delete G4 kernel

IV. How to configure/build user applications
1. Environment variables

- Mandatory variables
  - G4SYSTEM – OS (e.g. Linux-g++)
  - G4INSTALL – base directory of Geant4
  - CLHEP_BASE_DIR – base directory of CLHEP
- Variables for physics processes in case corresponding processes are used
  - G4LEVELGAMMADATA - photon evaporation
  - G4LEDATA - cross-sections for Low-E EM module
  - G4RADIOACTIVEDATA - radioactive decay
  - G4NEUTRONHPDATA - neutron cross-section
- Additional variables for GUI/Vis/Analysis

IV. How to configure/build user applications
2. Source files

- Prepare directory for your application. In that directory
  - myApplication.cc
    - File in which main() is defined
  - src/
    - Source files of mandatory user classes and optional user run action classes
  - includes/
    - Header files
  - GNUmakefile



IV. How to configure/build user applications
3. GNUmakefile

```
name := myApplication
G4TARGET := $(name)
G4EXLIB := true
G4WORKDIR := .

.PHONY: all
all: lib bin

include $(G4INSTALL)/config/binmake.gmk
```

- Command gmake will create
  - executable in bin/$G4SYSTEM/
  - temporary object files in tmp/$G4SYSTEM/myApplication/

V. How to run user applications

1. Ways to run Geant4 application

- There are 3 Ways to run Geant4 application
  - ~~Hard-coded batch mode~~
  - interactive mode, driven by command lines
    - Use G4UIsession
  - batch mode, but reading a macro of commands
    - Use G4UIsession
  - interactive mode via a Graphical User Interface

V. How to run user applications
2. Using G4UIsession

- In your main main(), according to your computer environments, construct a G4UIsession concrete class provided by Geant4 and invoke its sessionStart() method.
- Geant4 provides
  - G4UIterminal and G4UItcsh
    - character terminal
  - G4UIXm, G4UIXaw and G4UIWin32
    - variations of the upper terminal by using a Motif, Athena or Windows widget
  - G4UIGAG and G4UIGainServer
    - a fully Graphical User Interface and its extension GainServer of the client/server type

V. How to run user applications

3. How to write main() for using UIsession

```
...
int main(int argc, char** argv) {
...
if (argc != 1)  {  // batch mode
     G4UImanager * UI = G4UImanager::GetUIpointer();
     G4String command = "/control/execute ";
     G4String fileName = argv[1];
     UI->ApplyCommand(command+fileName);
  }
 else  {              // interactive mode : define UI terminal
      G4UIsession * session =
            new G4UIterminal(new G4UItcsh);
      session->SessionStart();
      delete session;
  }
...
```

VI. UI commands and macro

- A command consists of
  - Command directory
  - Command
  - Parameter(s)

/run/beamOn 1

- A parameter can be a type of string, boolean, integer or double.
  - Space is a delimiter.
  - Use double-quotes (“”) for string with space(s).
- A parameter may be “omittable”. If it is the case, a default value will be taken if you omit the parameter.
  - Default value is either predefined default value or current value according to its definition
  - If you want to use the default value for your first parameter while you want to set your second parameter, use “!” as a place holder.

    dir/command ! Second

VI. UI commands and macro
1. Command submission

- Geant4 UI command can be issued by
  - ~~Hard-coded implementation~~
  - (G)UI interactive command submission
  - Macro file
- The availability of individual command, the ranges of parameters vary
  - implementation of your application
  - May vary dynamically during the execution
- some commands are available only for limited Geant4 application state(s).

  E.g. **/run/beamOn** is available only for Idle states.

VI. UI commands and macro
2. Marco file

- Macro file is an ASCII file contains UI commands.
- All commands must be given with their full-path directories.
- Use “#” for comment line.
  - First “#” to the end of the line will be ignored.
  - Comment lines will be echoed if **/control/verbose is set to 2.**
- Macro file can be executed
  - interactively or in (other) macro file
    /control/execute *file_name*
  - ~~hard-coded~~

    ~~**G4UImanager* UI = G4UImanager::GetUIpointer();**~~

    ~~**UI->ApplyCommand("/control/execute *file_name*");**~~

VI. UI commands and macro
3. Command refusal

- Command will be refused if
  - Wrong application state
  - Wrong type of parameter
  - Insufficient number of parameters
  - Parameter out of its range
  - For integer or double type parameter
  - Parameter out of its candidate list
  - For string type parameter
  - Command not found

VI. UI commands and macro
4. G4UIterminal

- G4UIterminal is a concrete implementation derived from G4UIsession abstract class. It provides character-base interactive terminal functionality to issue Geant4 UI commands
  - C-shell or TC-shell (Linux only)
- It supports some Unix-like commands for directory manipulation
  - **cd, pwd -** change and display current command directory
    - By setting the current command directory, you may omit (part of) directory string
  - **ls** - list available UI commands and sub--directories
- It also supports some other commands
  - **history** - show previous commands
  - ! *historyID -re-issue previous command*
  - **arrow keys** (TCarrow TC--shell only)
  - **?** *UIcommand* - show current value
  - **help** [UIcommand] – help
  - **exit** – job termination
- Above commands are interpreted in G4UIterminal and are not passed to Geant4 kernel. Cannot use them in a macro file.

VI. UI commands and macro
4. G4UIterminal example

```
Idle> ls
Command directory path : /
 Sub-directories :
   /control/   UI control commands.
   /units/   Available units.
   /persistency/   Control commands for Persistency package
   /geometry/   Geometry control commands.
   /tracking/   TrackingManager and SteppingManager control commands.
   /event/   EventManager control commands.
   /run/   Run control commands.
   /random/   Random number status control commands.
   /particle/   Particle control commands.
   /process/   Process Table control commands.
   /physics/   ...Title not available...
   /gun/   Particle Gun control commands.
   /vis/   Visualization commands.
   /material/   Commands for materials
   /hits/   Sensitive detectors and Hits
 Commands :
Idle>
```

VI. UI commands and macro
4. G4UIterminal example contd.

```
Idle> cd control/
Idle>ls
Command directory path : /control/


Guidance :
UI control commands.

 Sub-directories :
   /control/matScan/    Material scanner commands.
 Commands :
   execute * Execute a macro file.
   loop * Execute a macro file more than once.
   foreach * Execute a macro file more than once.
   suppressAbortion * Suppress the program abortion caused by G4Exception.
   verbose * Applied command will also be shown on screen.
…
   createHTML * Generate HTML files for all of sub-directories and commands.
   maximumStoredHistory * Set maximum number of stored UI commands.
Idle>
```

VI. UI commands and macro
4. G4UIterminal example contd.

```
Idle> help /gun/energy

Command /gun/energy
Guidance :
Set kinetic energy.

Parameter : Energy
 Parameter type  : d
 Omittable : True
 Default value   : taken from the current value

Parameter : Unit
 Parameter type  : s
 Omittable       : True
 Default value   : GeV
 Candidates      : eV keV MeV GeV TeV PeV J electronvolt kiloelectronvolt
      megaelectronvolt gigaelectronvolt teraelectronvolt petaelectronvolt joule

Idle>
```

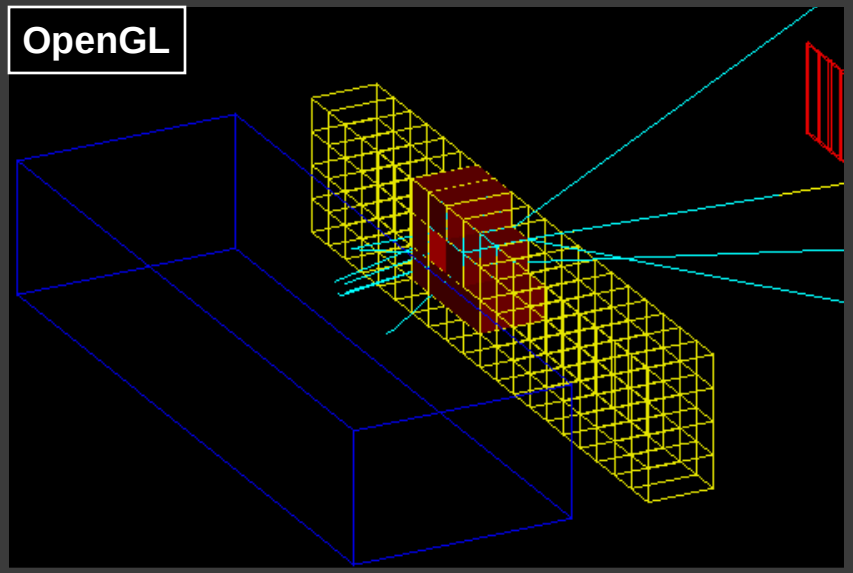
VII. How to visualize
1. What can be visualized

- Simulation data can be visualized:
  - Detector components
  - Particle trajectories and tracking steps
  - Hits of particles in detector components
- Other user defined objects can be visualized:
  - Polylines
    - such as coordinate axes
  - 3D Markers
    - such as eye guides
  - Text
    - descriptive character strings
    - comments or titles …

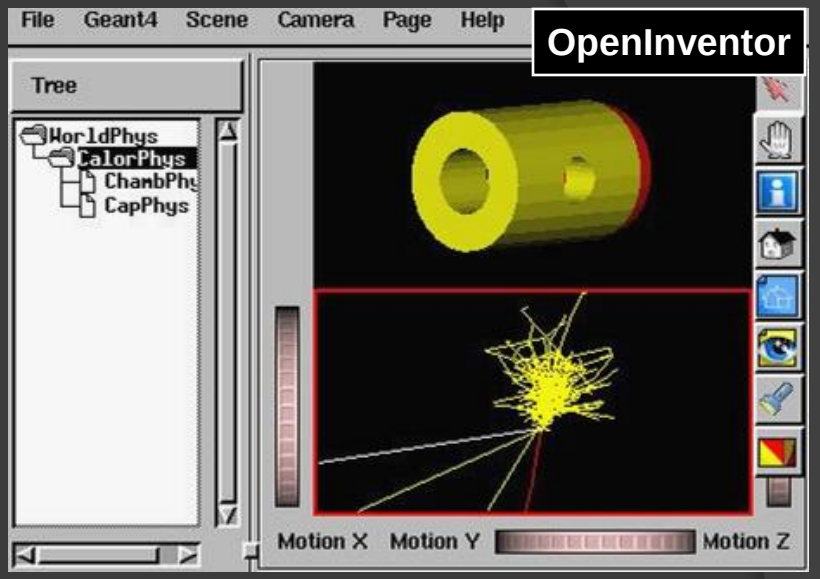
VII. How to visualize
2. Seven Visualization Drivers

- OpenGL
- OpenInventor
- HepRep/WIRED (and FRED)
- DAWN
- VRML
- RayTracer
- ASCIITree

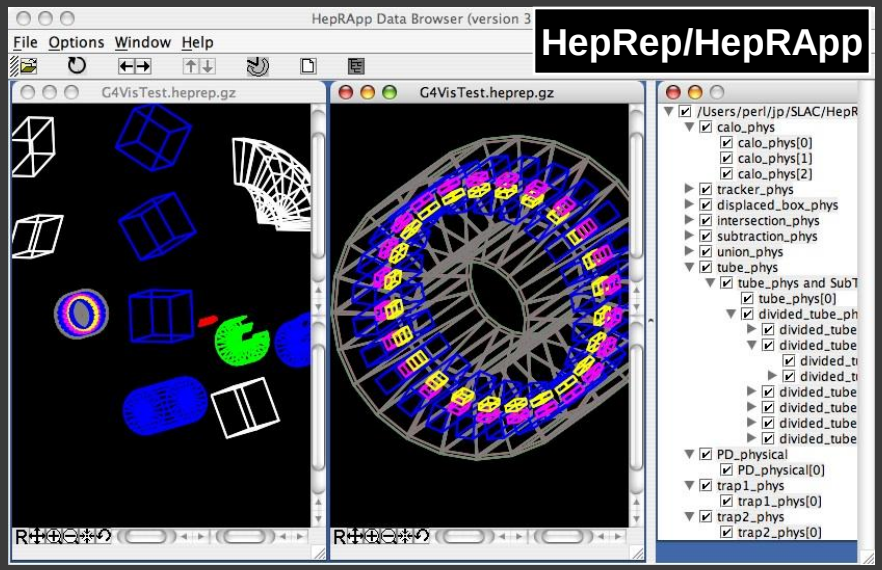
OpenGL



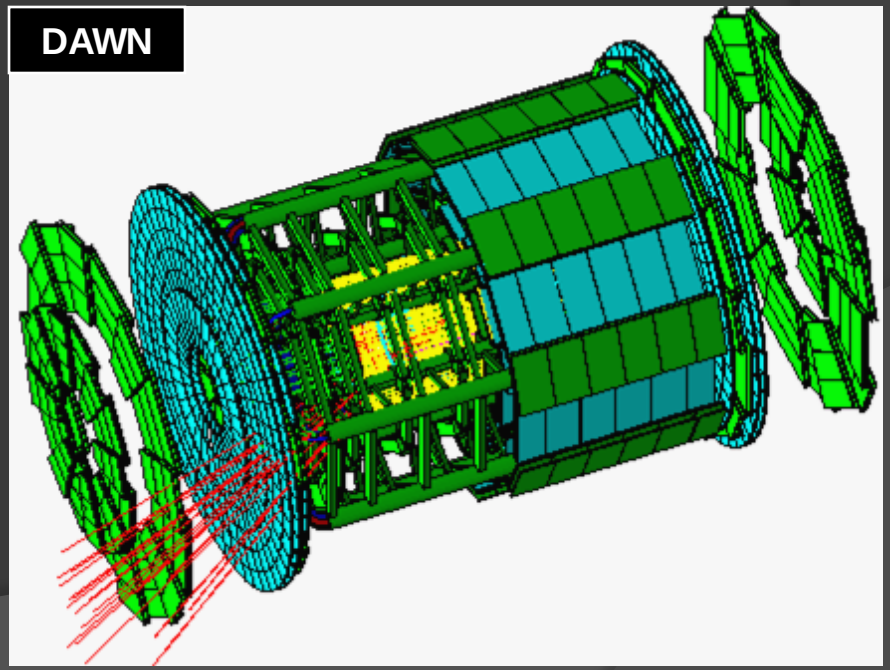
OpenInventor

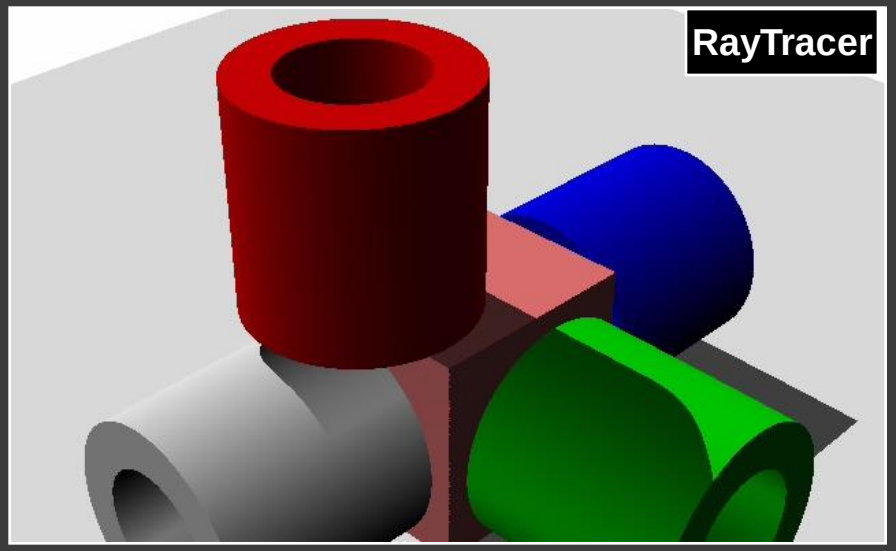


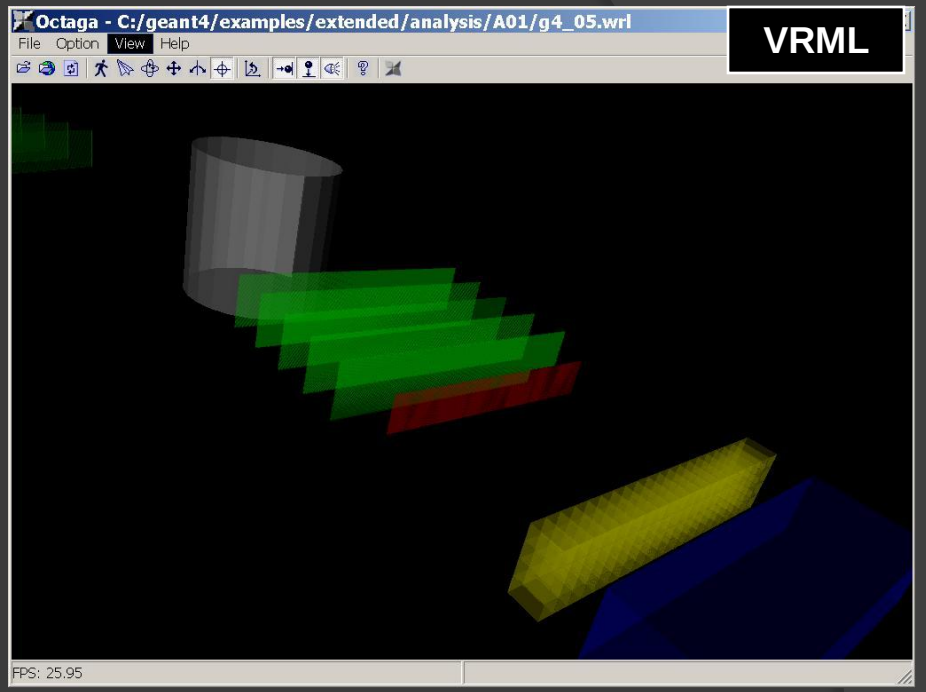
HepRep/HepRApp

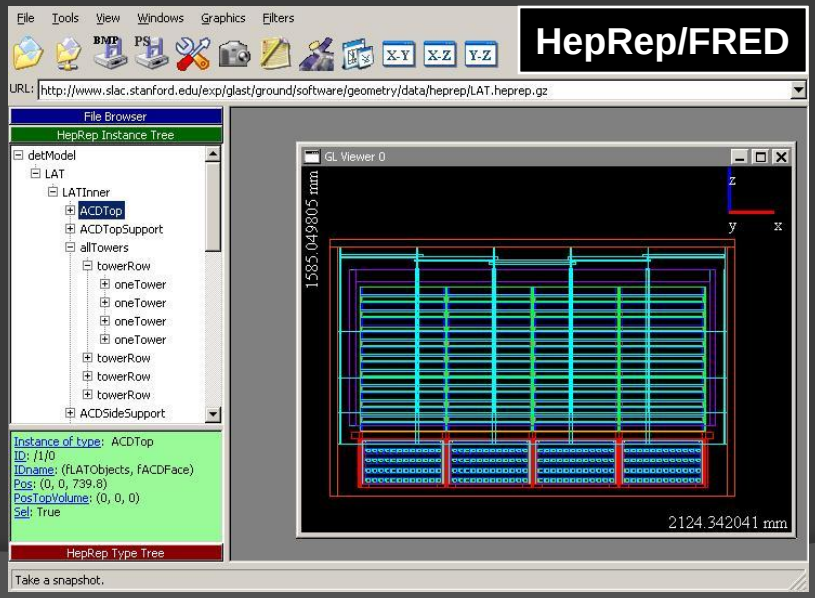


DAWN




RayTracer


Octaga - C:/geant4/examples/extended/analysis/A01/g4_05.wrl
File Option View Help
VRML
FPS: 25.95


File Tools View Windows Graphics Filters
HepRep/FRED
URL: http://www.slac.stanford.edu/exp/glast/ground/software/geometry/data/heprep/LAT.heprep.gz
File Browser
HepRep Instance Tree
detModel
LAT
LATInner
ACDTop
ACDTopSupport
allTowers
towerRow
oneTower
oneTower
oneTower
oneTower
towerRow
towerRow
towerRow
ACDSideSupport
Instance of type: ACDTop
ID: /1/0
IDname: (fLATObjects, fACDFace)
Pos: (0, 0, 739.8)
PosTopVolume: (0, 0, 0)
Sel: True
HepRep Type Tree
GL Viewer 0
1585.049805 mm
2124.342041 mm
Take a snapshot.

VII. How to visualize

3. Choose the Driver that Meets your Current Needs

- If you want responsive
  - OpenGL
- If you want interactivity
  - OpenInventor, HepRep/WIRED
- If you want highest quality
  - DAWN
- If you want to render to a 3D format that others can view in a variety of commodity browsers
  - VRML
- If you want photo-realistic high quality
  - RayTrace
- If you want to quickly check the geometry hierarchy
  - ASCIITree

VII. How to visualize

4. Adding Visualization to Your Executable

- Visualization Drivers must be installed
- Environmental variables may be needed
  - G4VIS_BUILD_DRIVERNAME_DRIVER and G4VIS_USE_DRIVERNAME
- How to write the main()

```
…
#ifdef G4VIS_USE
#include "G4VisExecutive.hh"
#endif
…
#ifdef G4VIS_USE
    G4VisManager* visManager = new G4VisExecutive;
    visManager->Initialize();
#endif
    …
#ifdef G4VIS_USE
    delete visManager;
#endif
    …
```

VII. How to visualize
5. Visualization commands

- Visualize a detector using DAWN

  /vis/open DAWNFILE

  /vis/drawVolume

  /vis/viewer/flush

- Visualize trajectories for 10 events using OpenGL

  /vis/open OGLIX

  /vis/viewer/set/viewpointThetaPhi 70 20

  /vis/viewer/zoom 2

  /vis/drawVolume

  /vis/scene/add/trajectories

  /vis/scene/endOfEventAction accumulate

  /run/beamOn 10

Create DAWN file

Draw detector

Execute the visualization

Camera control

Draw trajectories of particles

Accumulate trajectories in a figure

VII. How to visualize

6. Controlling visualization attributes

- In Construct() of DetectorConstruction

```
…
logicCalor = new G4LogicalVolume(…);
…
G4VisAttributes* simpleBoxVisAtt =
    new G4VisAttributes(G4Colour(1.0, 0, 0));
logicCalor->SetVisAttributes(simpleBoxVisAtt);

logicWorld
    ->SetVisAttributes(G4VisAttributes::Invisible);
…
```