Geant-4's Architecture

John Apostolakis, CERN

for Geant4 collaboration

Contents

- *Geant4's goals and scope*
- *Geant4's overall design*
  - *The kernel class categories*
  - *Geant4's interfaces*
- *Dependencies*

GEANT 4 Context

- Software Engineering and OO technology
- Detector simulation tool-kit for HEP
- Requirements from:
  - LHC
  - heavy ions, CP violation, cosmic rays
  - medical and space science applications
- World-wide collaboration

Geant4 Capabilities

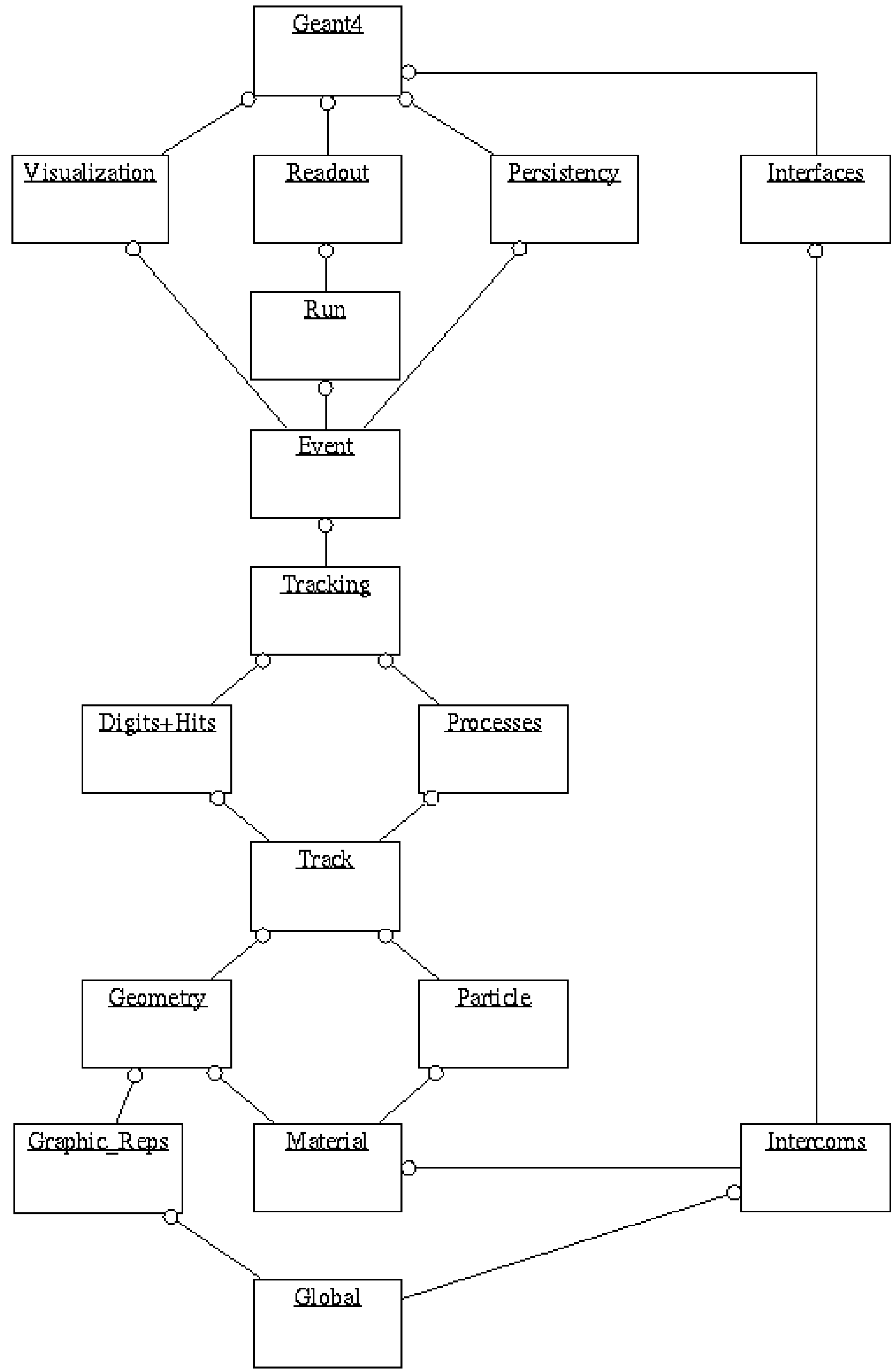
- *Very powerful Geant4 kernel*
  - *tracking, stacks, geometry, hits, ..*
- *Extensive & transparent physics models*
  - *electromagnetic, hadronic, …*
- *Additional*
  - *persistency, visualization, ...*
- *Surpasses Geant-3*
  - *in nearly every respect*

Dependencies

- *"Dictionary"*
  - *CLHEP*
    - *mostly G4 exports: RNGs, 3-vect.*
  - *Class library for containers*
    - *this was Rogue Wave Tools.h++*
    - *Since version 4.0.1 we also support STL*
      - *next version is expected to use only STL*
    - *RW used directly in code*
      - *so currently we implement this interface in STL*

Design

- Class category diagram:
  - Clear dependencies



The kernel and interfaces

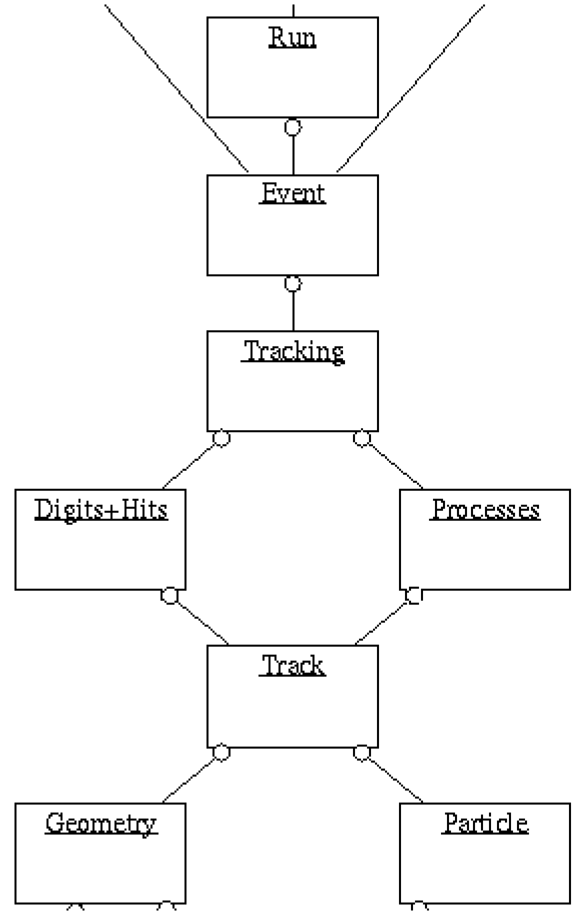
- *Kernel classes*
  - *designed for simulation*
  - *careful adherence to class categories*
- *Interface classes*
  - *designed to decouple Geant4 from solution*
  - *anyone can plug-in their choice*
    - *eg user interface, visualization*

Required user definitions

- *Detector Construction*
  - *code, GGE, persistent*
- *Primary Generator*
  - *simple or event generator interface*
- *Physics List*
  - *chooses particles*
  - *chooses list of processes*
    - *for each particle*

Geant4 kernel

- *Tracking*
  - – *general & flexible*
- *Event*
  - • *powerful stacking*
    - – *at no cost*
- *Geometry*
  - – *hierarchy or flat; *
  - – *voxels for speed.*



Geant4 kernel

- *Includes categories for run, event, track*
- *One computing process can have many runs*

- *Run*
  - *each run has a fixed geometry & event-generator*
- *Event:*
  - *Manages track creation*
  - *Stacks for inactive tracks*
    - *3 default stacks*
    - *very powerful*
    - *no cost!*

Geant4 kernel: tracking

- *Tracking is general*
  - *same for all particle types*
  - *different list of processes for each particle*
- *It messages*
  - *sensitive detectors and user actions*
- *Users choose which processes to register*
  - *for each particle*

Geant4 kernel

- Hits & digitization
  - Experiment specific hits

  Handles event pileup
  - using new readout category
- Materials
  - isotopes, elements,
  - compounds, ...
- Particles
  - properties from PDG
- Intercoms:

  communicate
  - between categories,
  - from UI to kernel

Processes design

- Processes can be added by anyone
- Each process must define
  - Get Physical Interaction Length
    - proposes a limit to the step due to interaction
  - Do It
    - that proposes modification in particle's state

for each appropriate type of action :

at a point, along the step or in time.

Physics processes

- *EM physics*
  - *Several physical processes have different implementations:*
    - *differential vs. integral approach*
    - *adapted, eg for thin absorbers*
- *Hadronics*
  - *Distinguish process and model*
  - *Separate model designs*
    - *for parameterized, data and theory driven*

GEANT4 Physics Processes Design

- *The GEANT4 physics processes exploit Object-Oriented Technology to make transparent how physics results are produced.*
- *The way **cross sections** are calculated (via formulas, data files, etc. and using different data-sets with applicability by particle, energy, material) is clearly exposed via OO design and separated from the way they are accessed and used in the algorithms.*
- *The way the **final state** is computed is separated from the tracking and is split into alternative or complementary models, according to the energy range, the particle type, the material. Multiple implementations of physics processes and models are available.*

Units, data libraries

- *No numbers are hard-coded in formulas and algorithms. Instead variables and constants are used.*
- *An extensive set of units is defined in GEANT4 and* **all the numerical quantities are expressed through units explicitly.** *Users are free to choose any units.*
- *Data libraries and evaluations:*
  - ***ENDF/B, JENDL, FENDL, CENDL, ENSDF,JEF, BROND, EFF***
  - ***MENDL, IRDF, SAID, EPDL, EEDL, EADL, SANDIA, ...***
- *Distribution centers*
  - *NEA (also for HERMES-KFA), LLNL, BNL, KEK,*
  - *IAEA, IHEP, Helsinki, TRIUMF, FNAL (for MARS)........*

Parameterization/Fast Simulation

- *Fast Simulation Manager*
  - *Framework for parameterization*
  - *Takes over from detailed simulation*
  - *can return to detailed simulation (eg cracks)*
- *Can trigger on particle, volume, ..*
  - *Parallel geometrical description*
- *BaBar is developing Bogus based on this.*

Geant 4's interfaces

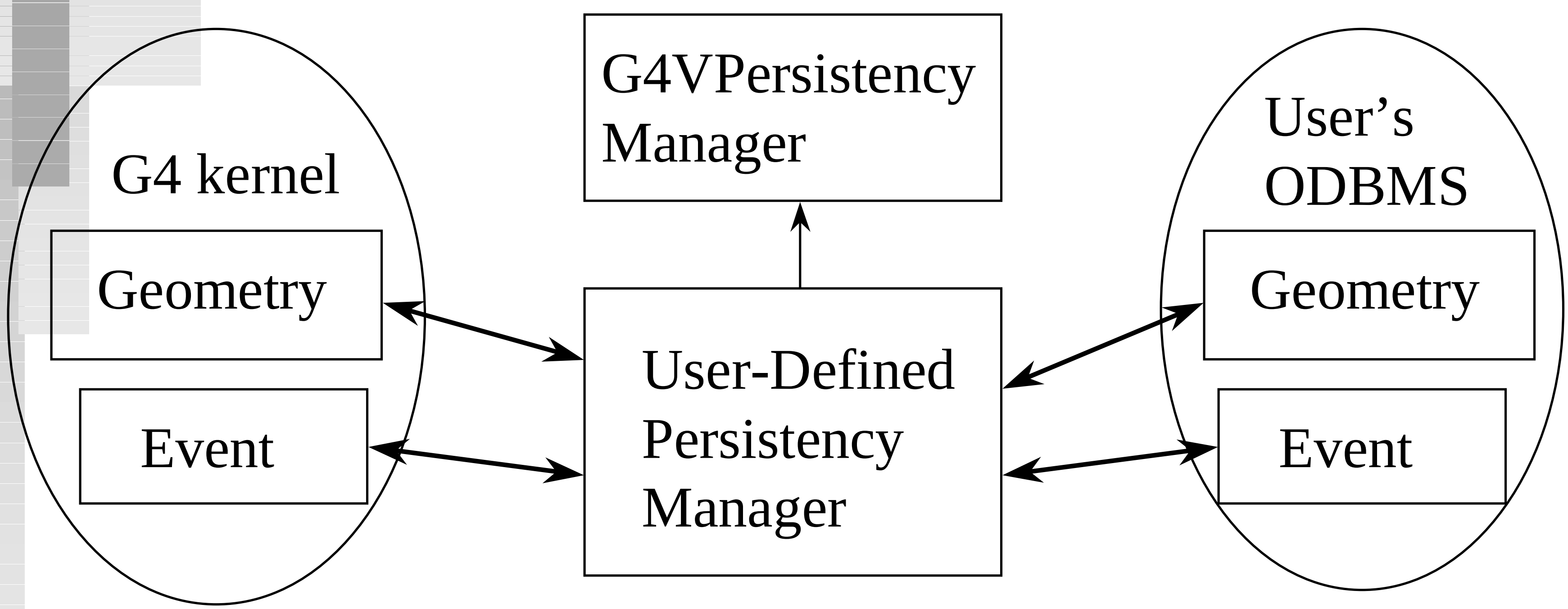
- *Abstract interfaces*
  - *used to decouple Geant4 from most possible dependencies*
- *In particular*
  - *User Interface, Visualisation*
  - *Hits*
  - *Persistency*

Object Persistency: Hits & other

- *Requirements:*
  - *storage for generated hits, runs, events, geometry ("in/out") and trajectories*
  - *decoupling from kernel*
    - *applications can be transient*
      - *Geant4 kernel does not know about persistency*
- *ODBMS solution via HepODBMS*
  - *in cooperation with RD45, testing with Objectivity*
- *Minimal modifications of implementation*

Persistency

- *To avoid the dependencies between G4 and the user's experimental data structure(s), we adopted parallel persistent class scheme:* *(M.Asai)*

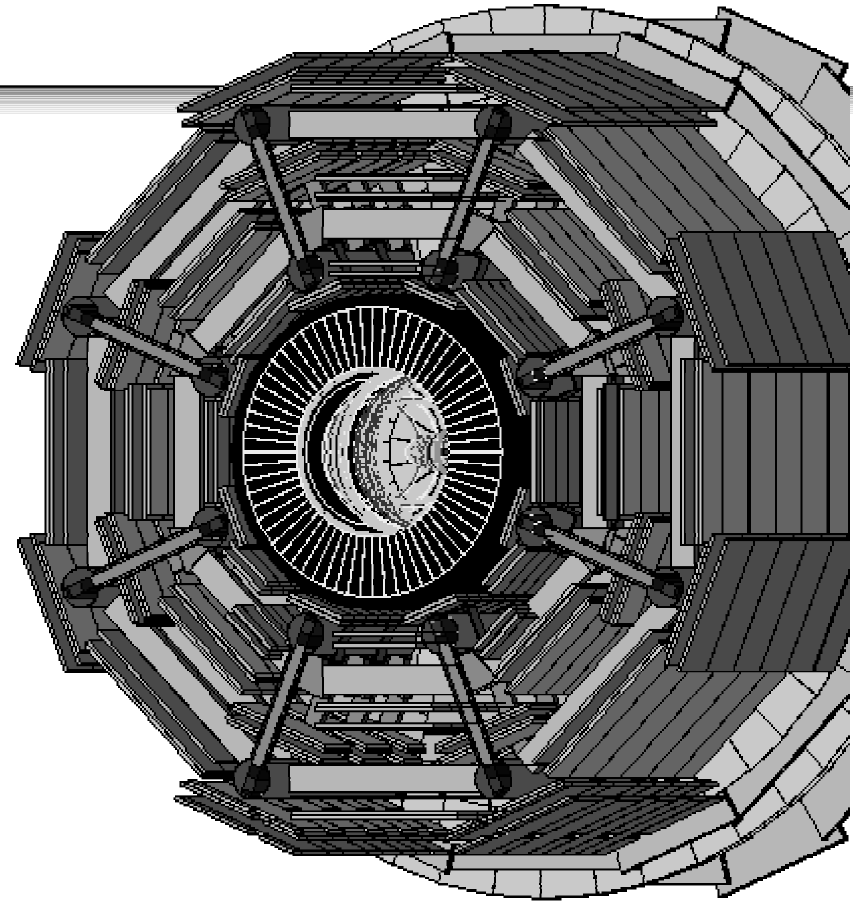


Visualization

- *Abstract base class for G4VVisManager*
- *Several drivers developed*
  - *Abstract interface*
    - *OpenGL, Open Inventor*
    - **DAWN renderer**
- *DAVID detects of volume intersection*
- *Same strategy for User Interfaces*
  - *Terminal, Momo, OPACS*

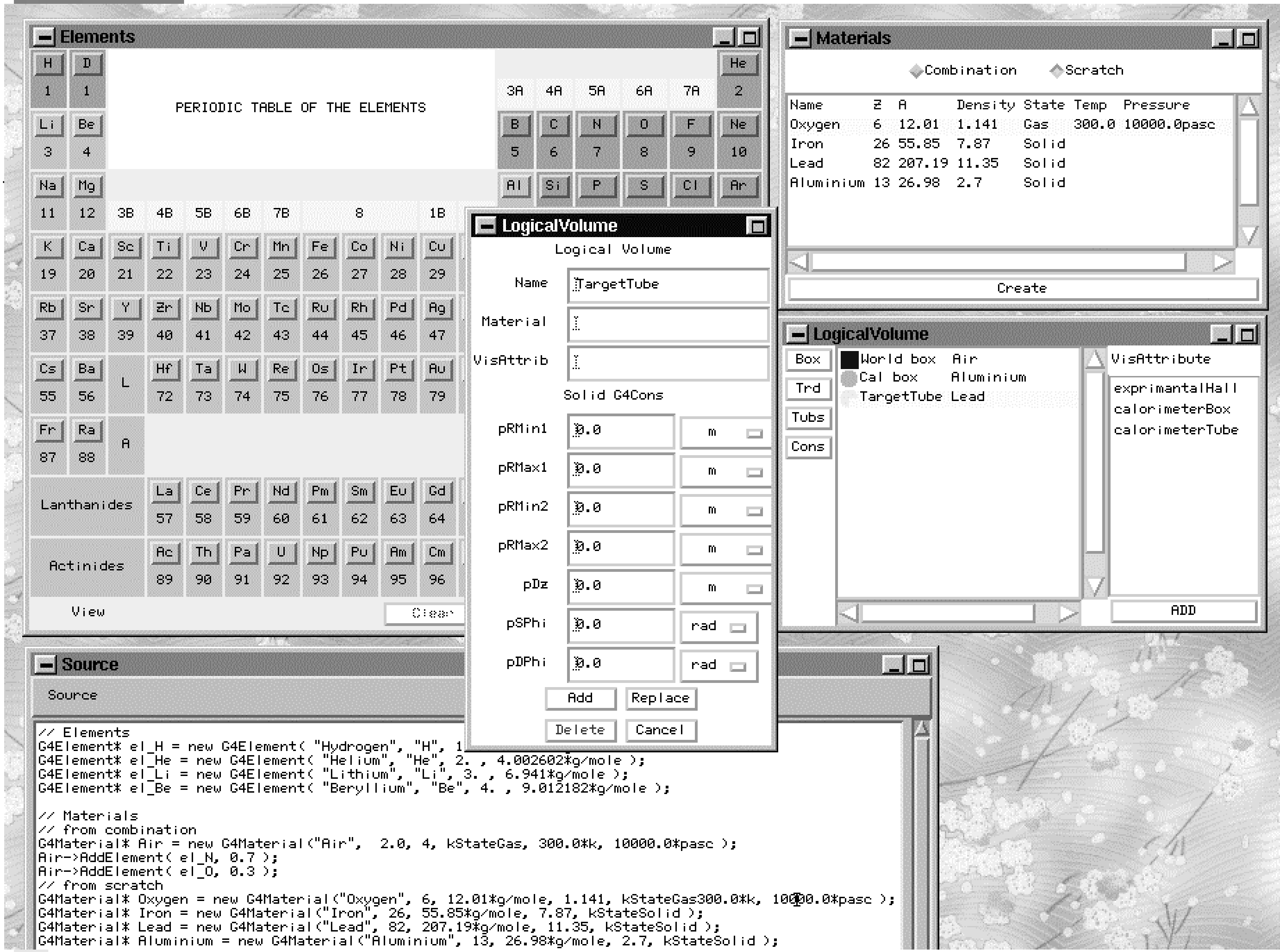
Example

- G4VVisManager
  - Abstract Base Class
- driver for DAWN. Output:
  - vector Postscript
  - Atlas detector
    - 100’s of volumes



Interfaced applications

- *Geometry editor*
  - *1st version: tcl/tk*
  - *now: Java / SWING*
- *Geometry checker*
  - *DAVID: implemented as Viz. driver*


Elements
PERIODIC TABLE OF THE ELEMENTS
H D He
3A 4A 5A 6A 7A
Li Be B C N O F Ne
Na Mg Al Si P S Cl Ar
3B 4B 5B 6B 7B 8 1B
K Ca Sc Ti V Cr Mn Fe Co Ni Cu
Rb Sr Y Zr Nb Mo Tc Ru Rh Pd Ag
Cs Ba L Hf Ta W Re Os Ir Pt Au
Fr Ra A
Lanthanides La Ce Pr Nd Pm Sm Eu Gd
Actinides Ac Th Pa U Np Pu Am Cm
View
Clear
Materials
Combination
Scratch
Name Z A Density State Temp Pressure
Oxygen 6 12.01 1.141 Gas 300.0 10000.0pasc
Iron 26 55.85 7.87 Solid
Lead 82 207.19 11.35 Solid
Aluminium 13 26.98 2.7 Solid
Create
LogicalVolume
Logical Volume
Name TargetTube
Material
VisAttrib
Solid G4Cons
pRMin1 0.0 m
pRMax1 0.0 m
pRMin2 0.0 m
pRMax2 0.0 m
pDz 0.0 m
pSPhi 0.0 rad
pDPhi 0.0 rad
Add
Replace
Delete
Cancel
LogicalVolume
Box
Trd
Tubs
Cons
World box Air
Cal box Aluminium
TargetTube Lead
VisAttribute
exprimantalHall
calorimeterBox
calorimeterTube
ADD
Source
Source
// Elements
G4Element* el_H = new G4Element("Hydrogen", "H", 1
G4Element* el_He = new G4Element("Helium", "He", 2. , 4.002602*g/mole);
G4Element* el_Li = new G4Element("Lithium", "Li", 3. , 6.941*g/mole);
G4Element* el_Be = new G4Element("Beryllium", "Be", 4. , 9.012182*g/mole);
// Materials
// from combination
G4Material* Air = new G4Material("Air", 2.0, 4, kStateGas, 300.0*k, 10000.0*pasc);
Air->AddElement(el_N, 0.7);
Air->AddElement(el_O, 0.3);
// from scratch
G4Material* Oxygen = new G4Material("Oxygen", 6, 12.01*g/mole, 1.141, kStateGas300.0*k, 10000.0*pasc);
G4Material* Iron = new G4Material("Iron", 26, 55.85*g/mole, 7.87, kStateSolid);
G4Material* Lead = new G4Material("Lead", 82, 207.19*g/mole, 11.35, kStateSolid);
G4Material* Aluminium = new G4Material("Aluminium", 13, 26.98*g/mole, 2.7, kStateSolid);

Help checking detector geometry

- After constructing a geometry
- DAVID suggests intersections between volumes
- Outputs
  - list of intersections
  - interactive visuals