Geometry Persistency

I. Hrivnacova, IJCLab Orsay

Geant4 IN2P3 and ED PHENIICS Tutorial,
22 – 26 May 2023

Outline

- Text files (ASCII)
- Geometry Description Markup Language (GDML)
- Exchanging geometries with ROOT
- Importing geometries from CAD

ASCII Text Models

ASCII File Format

- Well defined syntax for identifying the different geometrical entities:
  - materials, solids, volumes and volume attributes
- Dedicated manual:
  - https://geant4.web.cern.ch/collaboration/working_groups/persistency/docs/textgeom.pdf (link)
- Example of use of ASCII text model:
  - extended/persistency/P03

ASCII File Format - Example

g4geom_simple.txt

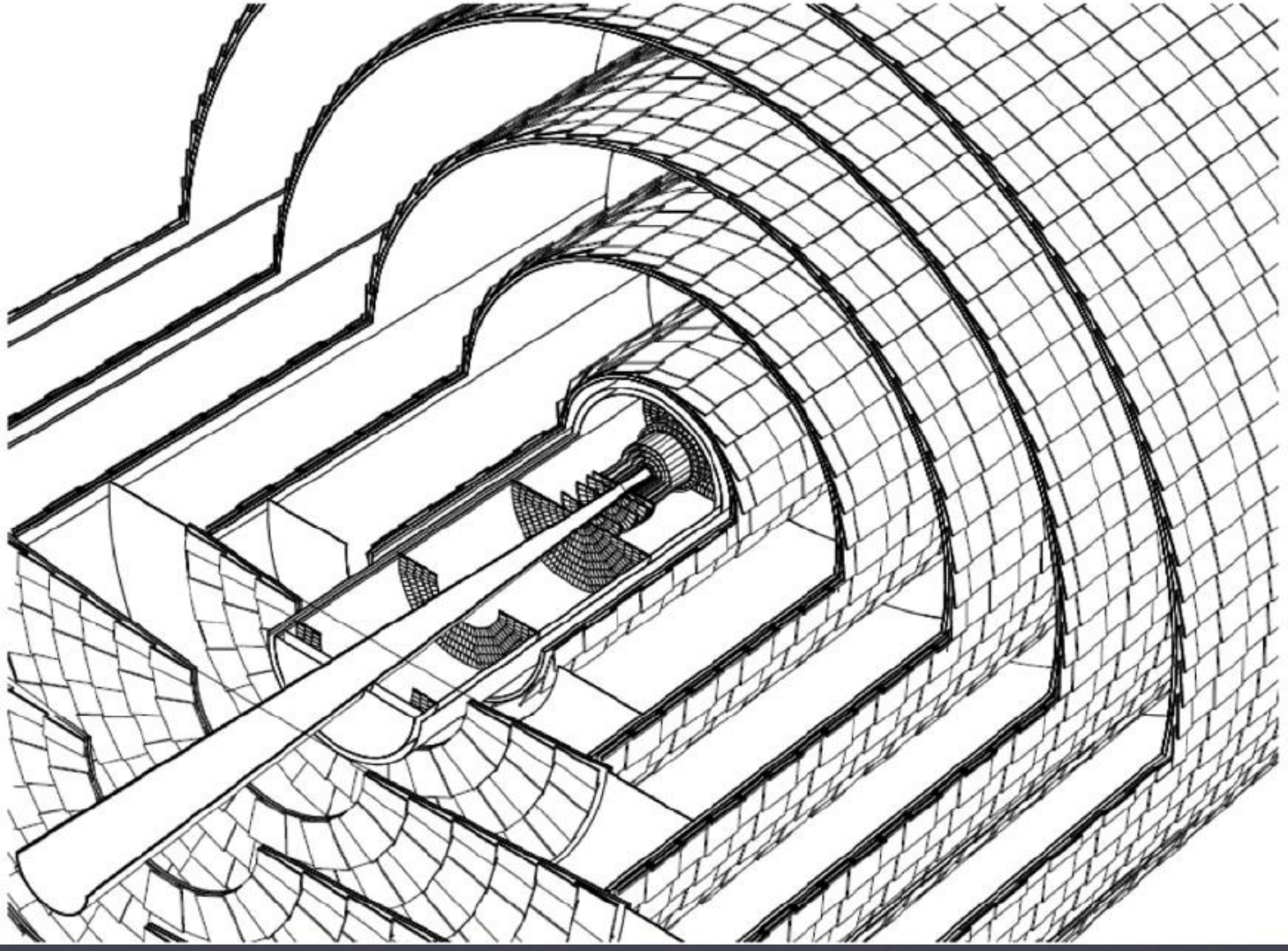
```
// Define a parameter for later
// use
:P POSZ 5.

// Define materials
:ELEM Hydrogen H 1. 1.
:ELEM Oxygen O 8 16.
:ELEM Nitrogen N 7 14.
:MIXT Air 1.214E-03 2
      Nitrogen   0.75
      Oxygen     0.25

// Define rotation matrix
:ROTM R00 90. 0. 90. 90. 0. 0.
// unit matrix
```

```
// Define volumes and place them
:VOLU world BOX 30. 30. 30. Air


:VOLU "my tube" TUBS 0. 10. 20.
0. 360. G4_WATER
:PLACE "my tube" 1 world R00 0.
0. $POSZ


:VOLU sphere ORB 5.  G4_AIR
:PLACE sphere 1 "my tube" R00 0.
1. 10.
```

GDML

https://gdml.web.cern.ch/GDML

Defining Geometry in GDML



Silicon Pixel & Microstrip Tracker for Collider Detector
Norman Graf, LCDD Collaboration, SLAC

GDML Geometry

- An XML-based language designed as an application-independent persistent format for describing geometries of detectors
  - Allows to define hierarchy of volumes, their materials and solids
- As pure XML, GDML can be used universally
  - Not just for Geant4
  - Can be used for interchanging geometries among different applications, used also to translate CAD geometries to Geant4
- XML (Extensible Markup Language) = a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.
- XML is simple
  - Rigid set of rules, self-describing data validated against schema
- XML is extensible
  - Easy to add custom features, data type

GDML Format - Example

```
<box name="worldBox" x="1"  y="1"  z="1"  unit="m" />
<box name="boxA"     x="10" y="10" z="10" unit="cm" />

<position name="pos1" x="25.0" y="50.0" z="75.0" unit="cm" />
<rotation name="rotZ" z="30.0" unit="deg" />

<volume name="World"/>
   <material ref="Air" />
   <solid ref="WorldBox" />
   <physvol >
       <volumeref ref="boxA"  />
       <positionref ref="pos1"  />
       <rotationref ref="rot1"  />
    </physvol >
</volume >
```

- Examples of use of GDML with Geant4:
  - extended/persistency/gdml

Geant4 <-> GDML

- G4GDMLParser class provides import/export of GDML files into/from Geant4
  - Import:

```
#include “G4GDMLParser.hh”

G4GDMLParser* parser;
parser.Read(”geometryFile.gdml”);
G4VPhysicalVolume* world = parser.GetWorldVolume();
```

  - Export

```
G4VPhysicalVolume* worldPV = ...;   // Get world
physical volume
G4GDMLParser* parser;
parser.Write(”geometryFile.gdml”, worldPV);
```

ROOT <-> GDML

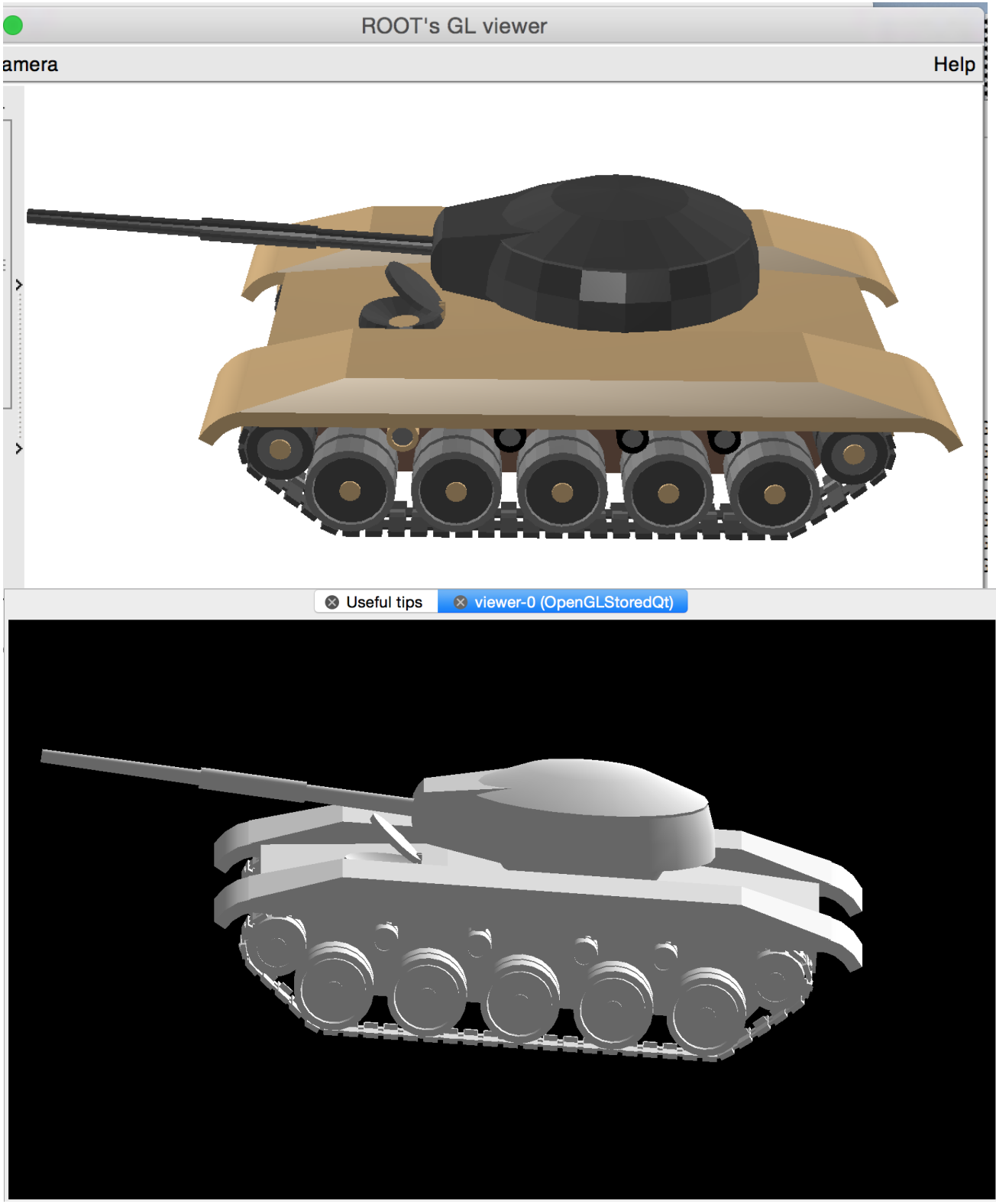
- ROOT geometry model can also import/export GDML
  - Import:

```
root[0] TGeoManager::Import("geometryFile.gdml");
```

  - Export

```
root[0] gGeoManager->Export("geometryFile.gdml");
```

- GDML can be used to exchange geometry between ROOT and Geant4
  - The converted geometry may be incomplete if the source geometry is using solids not supported in the destination geometry model
  - ROOT geometry allows overlapping volumes, **not allowed** in Geant4



root/tutorials/geom/tank.C

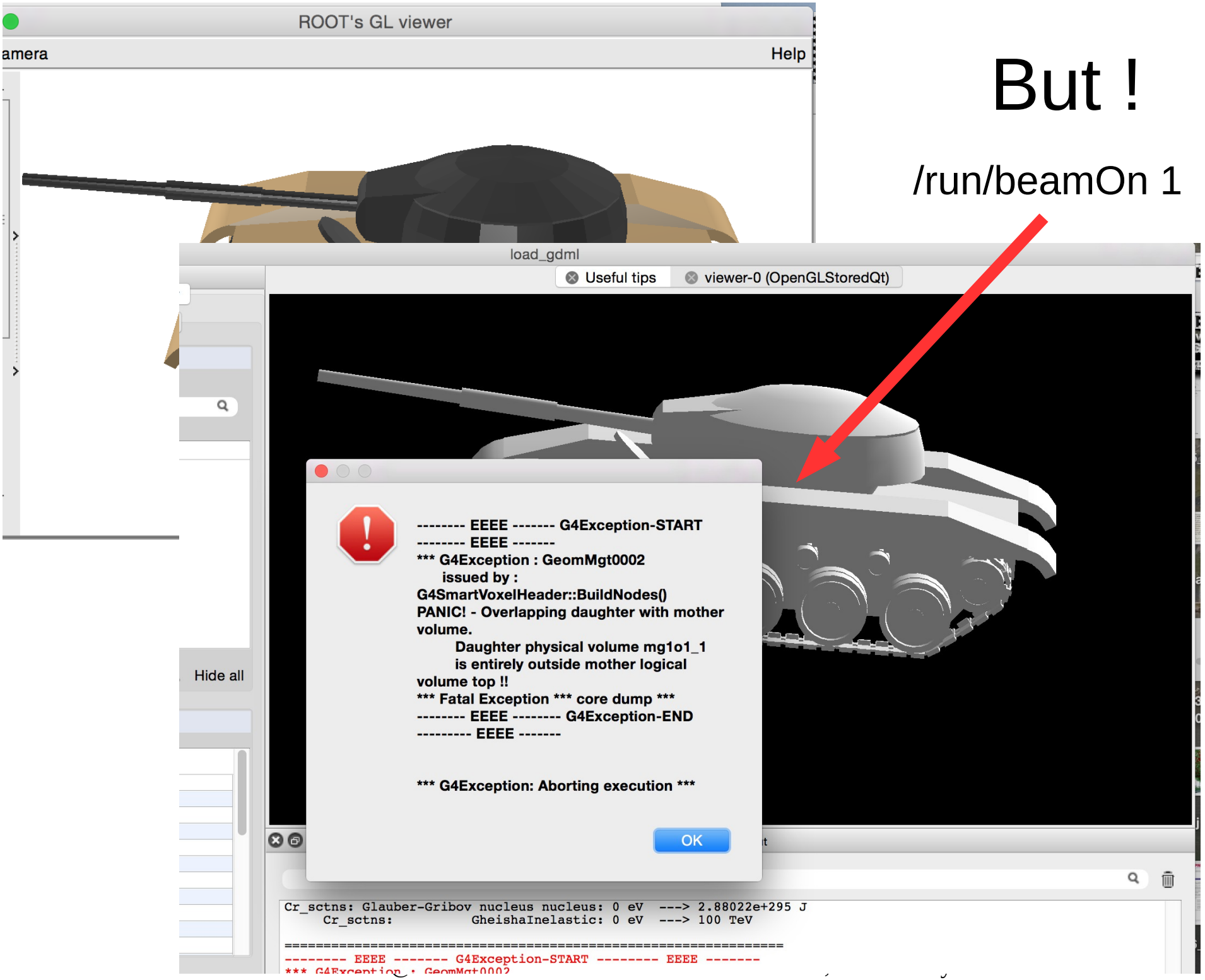
tank.gdml

geant4/examples/
extended/persistency/
gdml/G01

./load_gdml

But !

/run/beamOn 1



CAD
(Computer-aided design)

Importing CAD Geometries

- Use case: 3D engineering drawings to be incorporated into simulation as directly as possible
- Existing difficulties:
  - Proprietary, undocumented [or changing] CAD formats
  - In general, no connection between geometrical parts and materials
  - Level of details mismatch in the geometry
    - As required for engineering vs. particles transport for simulation
- CAD is never as easy as you might think
  - If the geometry is complex enough to require CAD in the first place
- Most CAD programs do support the STEP format, but…
  - Not a complete solution: does not contain material information
  - There are developments under way to define extensions for treatment of additional information, but none are widely adopted (i.e. not part of a Standard)

Importing CAD Geometries (2)

- **CADMesh:** **https://code.google.com/p/cadmesh/**
  http://arxiv.org/pdf/1105.0963.pdf

  - A direct CAD model import interface for Geant4 optionally based on VCGLIB, and ASSIM
  - It supports the import of **triangular facet surface meshes** defined in formats such as STL and PLY
  - A G4TessellatedSolid is returned and can be included in a standard user detector constructor

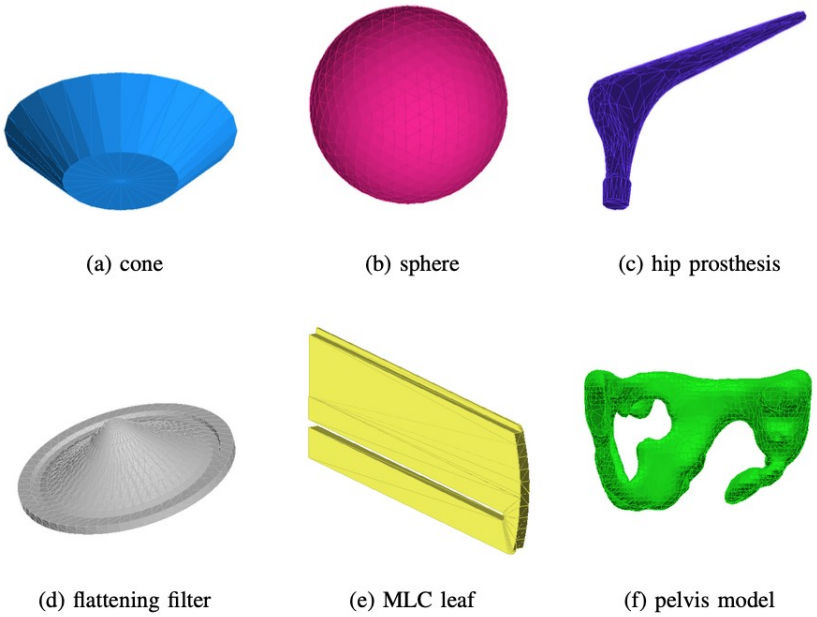
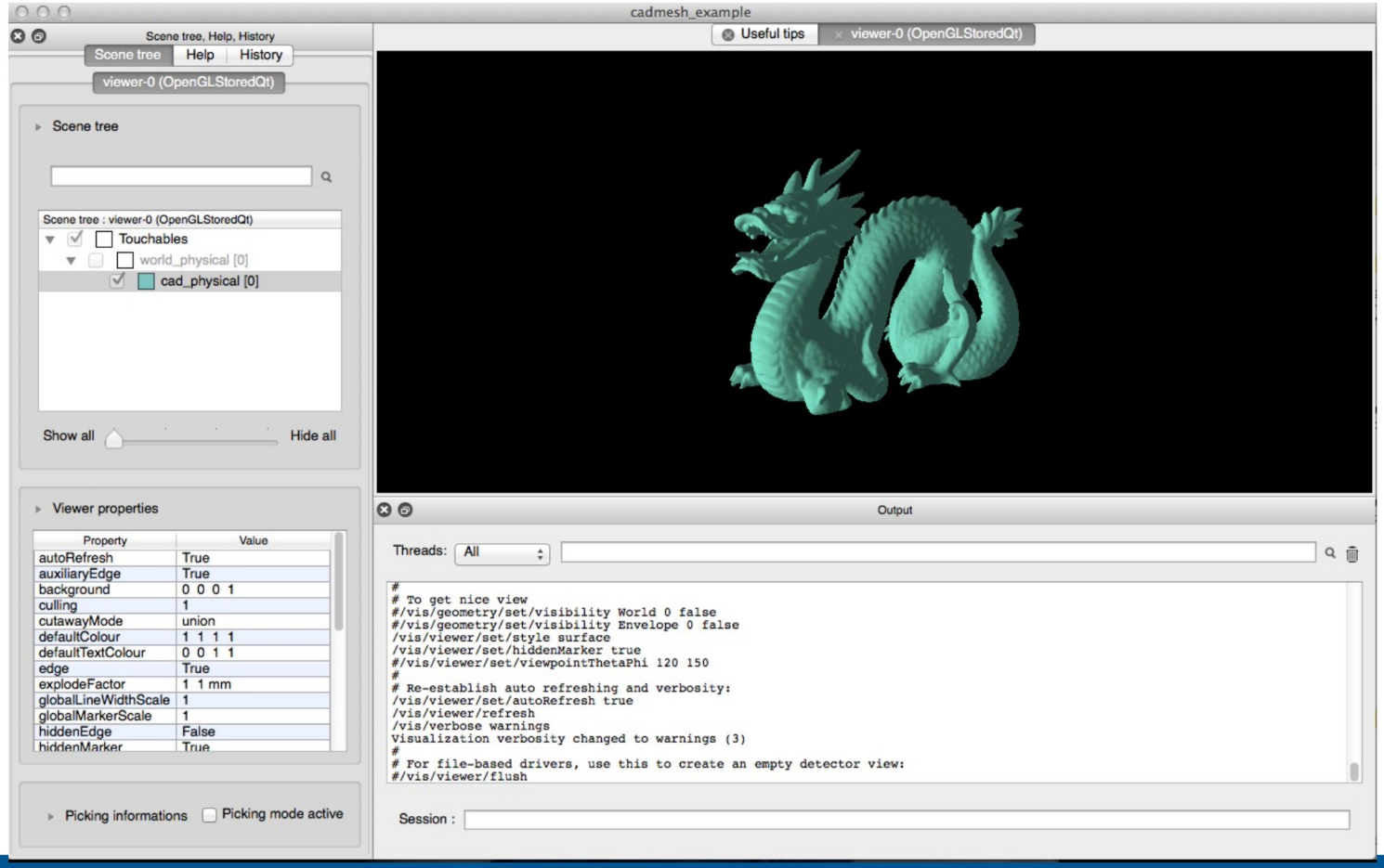


Fig. 2: Six test geometries loaded directly into GEANT4 using the proposed CAD interface.

CADMesh


cadmesh_example
Scene tree, Help, History
Scene tree
Help
History
viewer-0 (OpenGLStoredQt)
Scene tree
Scene tree : viewer-0 (OpenGLStoredQt)
Touchables
world_physical [0]
cad_physical [0]
Show all
Hide all
Viewer properties
Property
Value
autoRefresh
True
auxiliaryEdge
True
background
0 0 0 1
culling
1
cutawayMode
union
defaultColour
1 1 1 1
defaultTextColour
0 0 1 1
edge
True
explodeFactor
1 1 mm
globalLineWidthScale
1
globalMarkerScale
1
hiddenEdge
False
hiddenMarker
True
Picking informations
Picking mode active
Useful tips
viewer-0 (OpenGLStoredQt)
Output
Threads:
All
#
To get nice view
#/vis/geometry/set/visibility World 0 false
#/vis/geometry/set/visibility Envelope 0 false
/vis/viewer/set/style surface
/vis/viewer/set/hiddenMarker true
#/vis/viewer/set/viewpointThetaPhi 120 150
#
Re-establish auto refreshing and verbosity:
/vis/viewer/set/autoRefresh true
/vis/viewer/refresh
/vis/verbose warnings
Visualization verbosity changed to warnings (3)
#
For file-based drivers, use this to create an empty detector view:
#/vis/viewer/flush
Session :

Importing CAD geometries: more solutions ...

1. InStep, supporting import/export of different format, including STL, STEP and GDML:
   - https://www.solveering.com/InStep/instep.aspx
2. SALOME, can import STEP BREP/IGES/STEP/ACIS, mesh it, export to STL than use STL2GDML to export to GDML:
   - http://www.salome-platform.org
3. ESABASE2, space environment analysis CAD, basic modules are free for academic non-commercial use. Exports to GDML shapes or complete geometries. Imports STEP: https://esabase2.net
4. Blender GDML exporter, GDML plugin for the Blender tool:
   - http://projects.blender.org/tracker/index.php?func=detail&aid=30578&group_id=153&atid=467
5. FASTRAD, 3D tool for radiation shielding analysis; exports meshes to GDML: http://www.fastrad.net
6. STEP Solutions, commercial, exports meshes to then import as GDML: http://www.steptools.com/products/stdev/
7. Cogenda TCAD, for case of 3D meshes. Module Gds2Mesh exports to GDML:
   - http://www.cogenda.com/article/products#VTCAD
8. SW2GDML convert SolidWorks descriptions (using its API) to real primitives in GDML, including materials:
   - https://github.com/cvuosalo/SW2GDMLconverter
9. CadMC, tool to convert FreeCAD geometries to Geant4 (tessellated and CSG shapes): http://polar.psi.ch/cadmc/
10. EDGE, a commercial GDML editor, able to import/export STEP/STL geometries: https://www.space-suite.com/edge/
11. McCAD tool ‘integrated approach’ by KIT group, using half-space solids extensions to Geant4:
    - http://indico.cern.ch/event/400576/contributions/1841503/attachments/802327/1099598/CERN_Visit_YQIU_V0.2.pdf

Backup

GDML Overview

- Definitions
- Materials
- Solids
- Structures
- Setup

GDML Definitions

- Allow to define numerical values of constants, positions, rotations and scales that can be used later on in the detector description
  - Use of CLHEP expressions
- Constants

```
<constant name="length" value="6.25"/>
```

- Variables

```
<variable name="x" value="6"/>
<variable name="y" value="x/2"/>
```

  - Once defined, can be used anywhere later, eg.

```
<box name="my_box" x="x" y="y" z="z"/>
```

GDML Definitions (2)

- Positions

```
<position name="pos1" x="25.0" y="50.0" z="75.0" unit="cm" />
```

- Rotations

```
<rotation name="rotZ" z="30.0" unit="deg" />
```

- Scales

```
<scale name="reflectionZ" x="1.0" y="1.0" z="-1.0" />
```

- Matrices

```
<matrix name="m1" coldim="3" values="  0.4    9   126
                                       8.5    7    21
                                      34.6    7     9  />
```

GDML Materials

- Isotopes

```
<isotope name="U235" Z="92" N="235" >
   <atom type="A" value="235.04"  />
</isotope >
```

- Simple Elements

```
<element name="Oxygen" formula="O" Z="8" >
   <atom value="16"  />
</element >
```

  - Compare to:

```
G4Isotope* U235
   = new G4Isotope("U235", 92, 235, 235.04*g/mole);
G4Element* O
   = new G4Element("Oxygen", "O", 8.0, 16.0*g/mole);
```

GDML Materials (2)

- Elements with user-defined isotopic abundances

```
<element name="Enriched_uranium" >
   <fraction ref="U235" n="0.9"  />
   <fraction ref="U238" n="0.1"  />
</element >
```

  - Compare to:

```
G4Element* enU
   = new G4Element("enU", "U", 2)
enU->AddIsotope(isoU235, 90.*perCent);;
enU->AddIsotope(isoU238, 10.*perCent);;
```

GDML Materials (3)

- Material created directly from an element

```
<material name="Al" Z="13.0" >
   <D value="2.70" />
   <atom value="26.98" />
</material >
```

- Material created from previously defined elements or materials by number of atoms ("molecule"):

```
<material name="Water" formula="H2O" >
   <D value="1.0" />
   <composite n="2" ref="Hydrogen" />
   <composite n="1" ref="Oxygen" />
</material >
```

GDML Materials (4)

- Material created as a fractional mixture of previously defined elements or materials by number of atoms (“compound”):

```
<material name=”Air” formula=”air” >
   <D value=”0.00129” />
   <fraction n=”0.7” ref=”Nitrrogen” />
   <fraction n=”0.3” ref=”Oxygen” />
</material >
```

- Material via NIST ?

GDML Solids

- Collection of all Geant4 solid definitions
- Box, cone segment, ellipsoid, elliptical tube, elliptical cone, orb, paraboloid, parallelepiped, polycone. polyhedron, sphere, torus segment, trapezoid, general trapezoid, tube with hyperbolic profile, cut tube, tube segment, twisted box, twisted trapezoid, twisted general trapezoid, twisted tube segment, extruded solid, tessellated solid, tetrahedron
- Example of box:

```
<box name="my_box" x="x" y="y" z="z"/>
```

GDML Boolean Solids

- Supported boolean operations: *union, subtraction and intersection*

```
<box name="box_A"  x="1" y="5" z="20"/>
<box name="box_B"  x="4" y="4.5" z="18"/>
...
<union name="union"  >
    <first ref="box_A"  />
    <second ref="box_B"  />
    <position ref="union_position"  />
    <rotation ref="union_rotation"  />
</union >
```

GDML Volumes

- Volumes are created from solids and materials that were previously defined in this or a linked GDML files
- Both logical and physical volumes are defined in one structure:

```
<volume name="WorldBox "/>
  <material ref="Air" />
  <solid ref="WorldBox" />
  <physvol >
    <volumeref ref="box_B"  />
    <positionref ref="union_position"  />
    <rotationref ref="union_rotation"  />
  </physvol >
</volume >
```

- Replicated and parameterised volumes are also supported
  - See the documentation and examples

GDML Setup

- The top volume is defined in “setup” element:

```
<setup name=”TestI”  version=”1.0” />
  <world ref=”World” />
</setup >
```

- Multiple geometry setups can be defined choosing different volumes as world volumes
- Geometry description can be split over multiple files, allowing more granular and/or distributed development