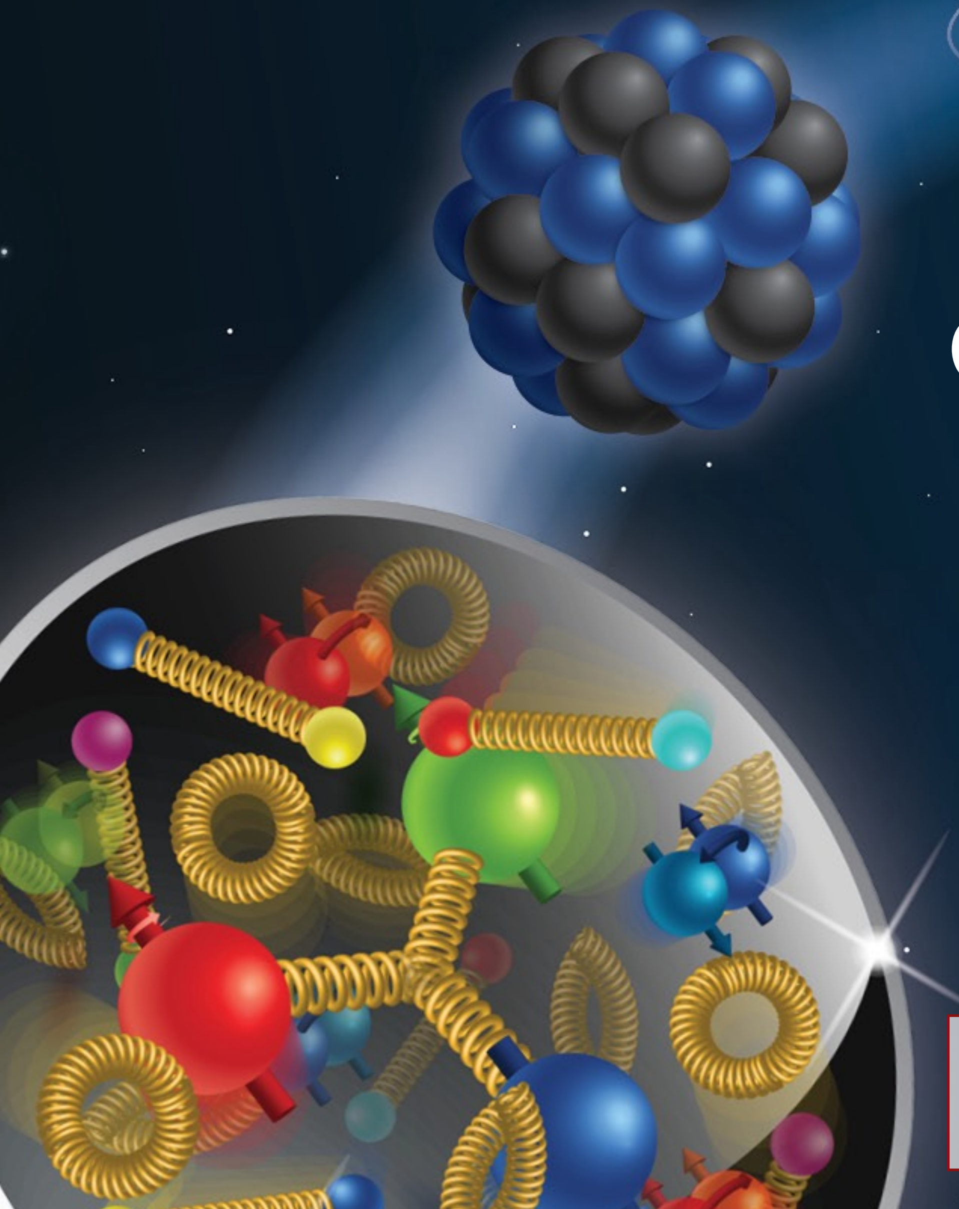
Version 11.1-p02

Geometry IV

Makoto Asai (Jefferson Lab)

Geant4 Tutorial Course



Contents

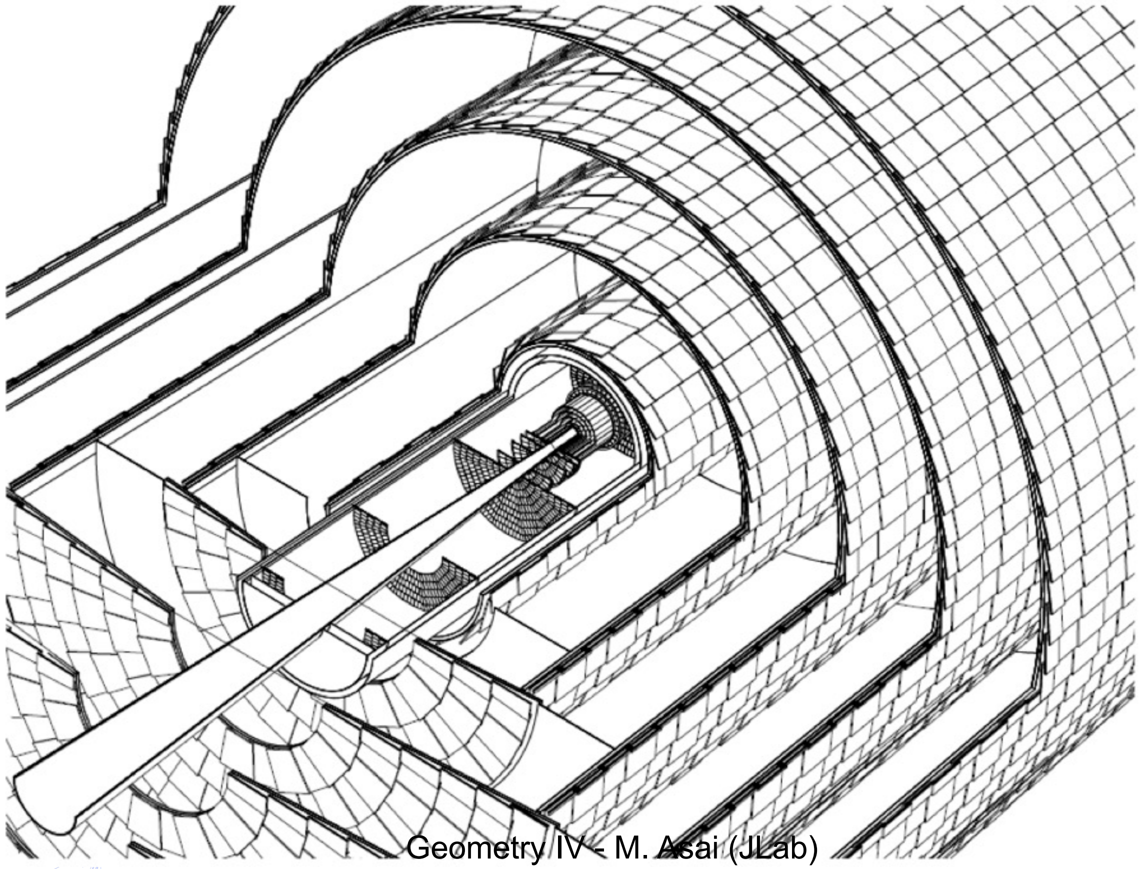
- GDML/CAD interfaces
- Geometry checking tools
- Geometry optimization
- Parallel geometry
- Moving objects

Contents

- GDML/CAD interfaces
- Geometry checking tools
- Geometry optimization
- Parallel geometry
- Moving objects

GDML and CAD Interfaces

- Up to now, the course has shown how to define materials and volumes from C++.
- This part of slides shows some alternate ways to define geometry at runtime by providing a file-based detector description.



Silicon Pixel & Microstrip Tracker for Collider Detector
N. Graf, J. McCormick, LCDD, SLAC

GDML : Geometrical Description Modeling Language

- An XML-based language designed as an application independent persistent format for describing the geometries of detectors.
  - Implements “geometry trees” which correspond to the hierarchy of volumes a detector geometry can be composed of
  - Allows materials to be defined and solids to be positioned
- Because it is pure XML, GDML can be used universally
  - Not just for Geant4
  - Can be format for interchanging geometries among different applications.
  - Can be used to translate CAD geometries to Geant4
- XML is simple
  - Rigid set of rules, self-describing data validated against schema
- XML is extensible
  - Easy to add custom features, data types
- XML is Interoperable to OS’s, languages, applications
- XML has hierarchical structure
  - Appropriate for Object-Oriented programming
  - Detector/sub-detector relationships

GDML structure

- Contains numerical values of constants, positions, rotations and scales that will be used later on in the geometry construction.
  - Uses CLHEP expressions
- Constants

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

- Variables

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

  - Once defined, can be used anywhere later, e.g.

```
<variable name="y" value="x/2"/>
<box name="my_box" x="x" y="y" z="x+y"/>
```

- Positions

```
<position name="P1" x="25.0" y="50.0" z="75.0" unit="cm"/>
```

- Rotations

```
<rotation name="RotateZ" z="30" unit="deg"/>
```

- Matrices

```
<matrix name="m" coldim="3" values=" 0.4 9 126

                                    8.5 7 21

                                    34.6 7 9"/>
```

GDML structure

- Simple Elements

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

- Material 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>
```

- Material as a fractional mixture of elements or materials, (“compound”):

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

Importing GDML file

- GDML files can be directly imported into Geant4 geometry, using the GDML plug-in facility:

  ```
  #include “G4GDMLParser.hh”
  ```

- Generally you will want to put the following lines into your DetectorConstruction class:

  ```
  G4GDMLParser parser;
  parser.Read(“geometryFile.gdml”);
  G4VphysicalVolume* W = parser.GetWorldVolume();
  ```

- To include the Geant4 module for GDML,
  - Install the XercesC parser (version 2.8.0 or 3.0.0)
    http://xerces.apache.org/xerces-c/download.cgi
  - Set appropriate environment variables when G4 libraries are built

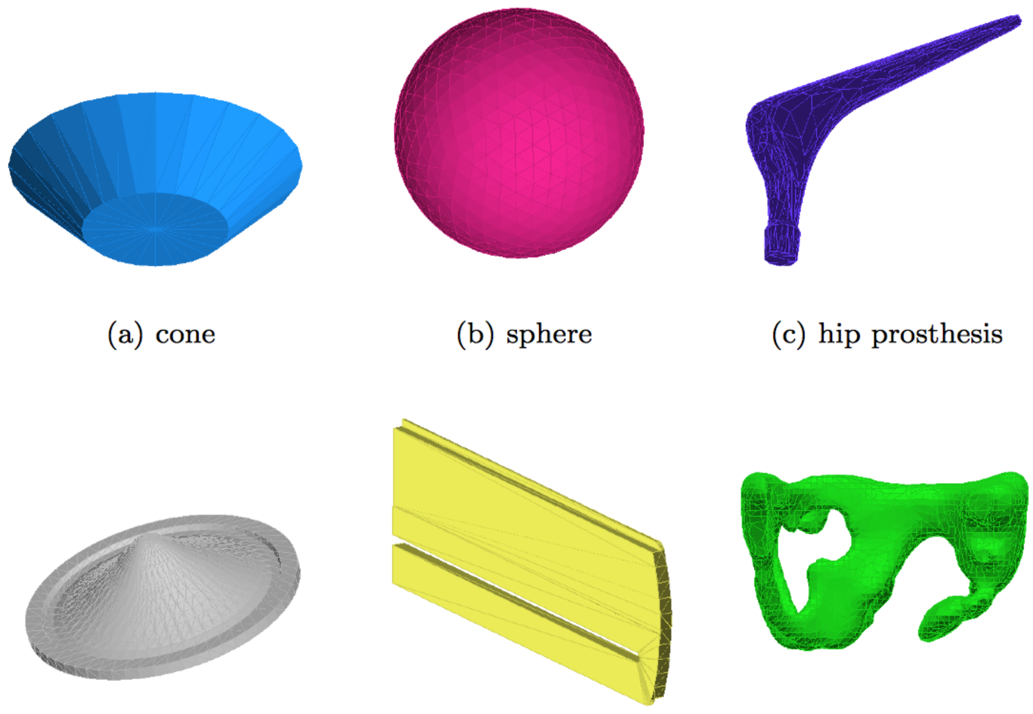
- Examples available in:
  $G4INSTALL/examples/extended/persistency/gdml

Importing CAD geometry

- Users with 3D engineering drawings may want to incorporate these into their Geant4 simulation as directly as possible
- Difficulties include:
  - Proprietary, undocumented or changing CAD formats
  - Usually no connection between geometry and materials
  - Mismatch in level of detail required to machine a part and that required to transport particles in that part
- CAD is never as easy as you might think (if the geometry is complex enough to require CAD in the first place)
- CADMesh is a direct CAD model import interface for GEANT4 optionally leveraging VCGLIB, and ASSIMP by default. Currently it supports the import of triangular facet surface meshes defined in formats such as STL and PLY. A G4TessellatedSolid object is returned and can be included in a standard user detector constructor.
  - https://code.google.com/p/cadmesh/
- One output format most CAD programs do support is STEP
  - Not a complete solution, in particular does not contain material information
  - There are movements under way to get new formats that contain additional information, but none yet widely adopted.

CADMesh

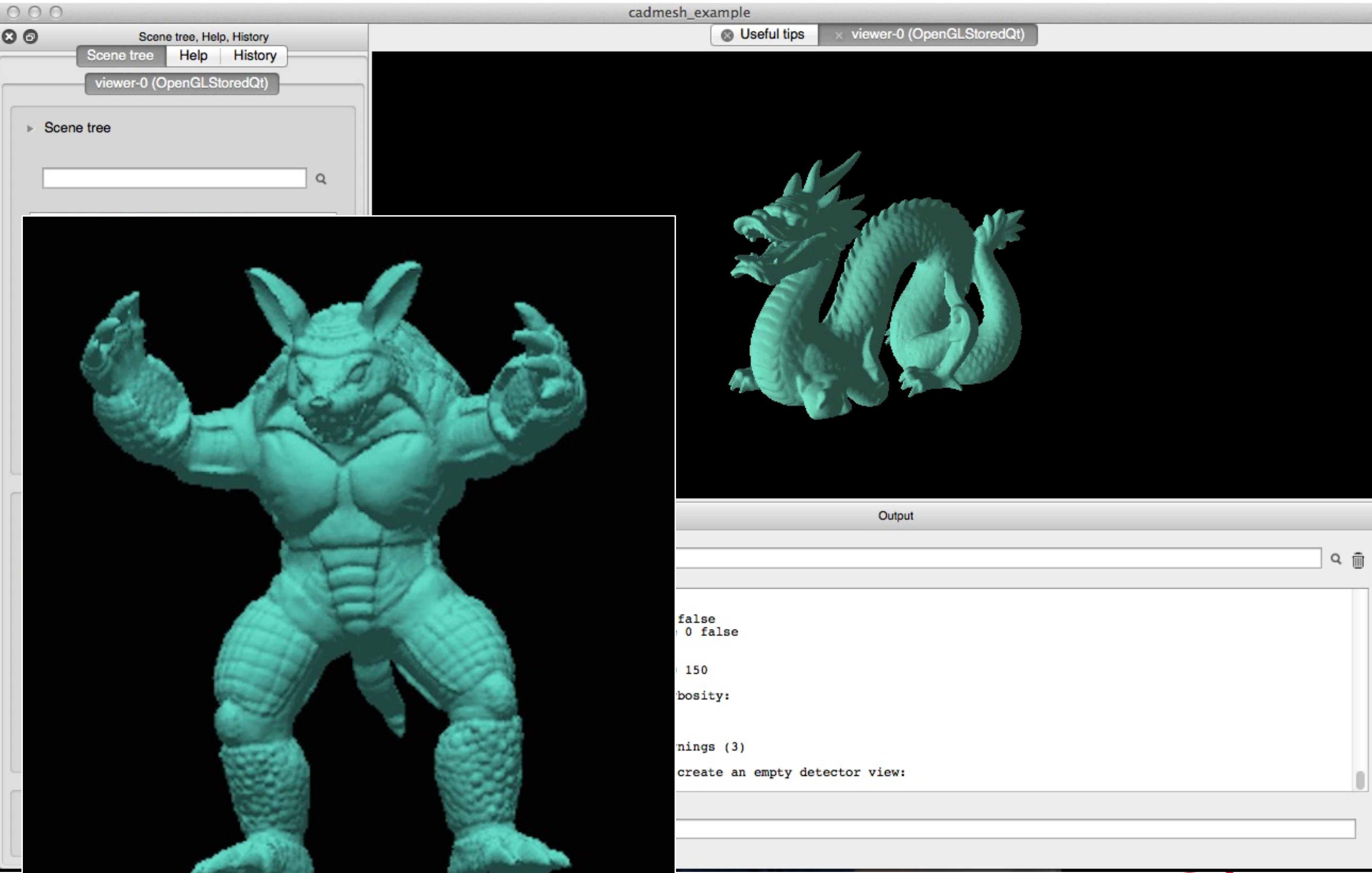
- CADMesh is a direct CAD model import interface for GEANT4 optionally leveraging VCGLIB, and ASSIMP by default. Currently it supports the import of triangular facet surface meshes defined in formats such as STL and PLY. A G4TessellatedSolid is returned and can be included in a standard user detector construction.
  - https://code.google.com/p/cadmesh/
  - http://arxiv.org/pdf/1105.0963.pdf


(a) cone (b) sphere (c) hip prosthesis

(d) flattening filter (e) MLC leaf (f) pelvis model

Fig. 3 Six test geometries loaded directly into GEANT4 using the proposed CAD interface and visualised using the GEANT4 OpenGL viewer.

CADMesh


cadmesh_example
Scene tree, Help, History
Scene tree
Help
History
viewer-0 (OpenGLStoredQt)
Scene tree
Useful tips
viewer-0 (OpenGLStoredQt)
Output

```
false
 0 false

 150

bosity:


nings (3)

create an empty detector view:
```

Loading CADMesh solid

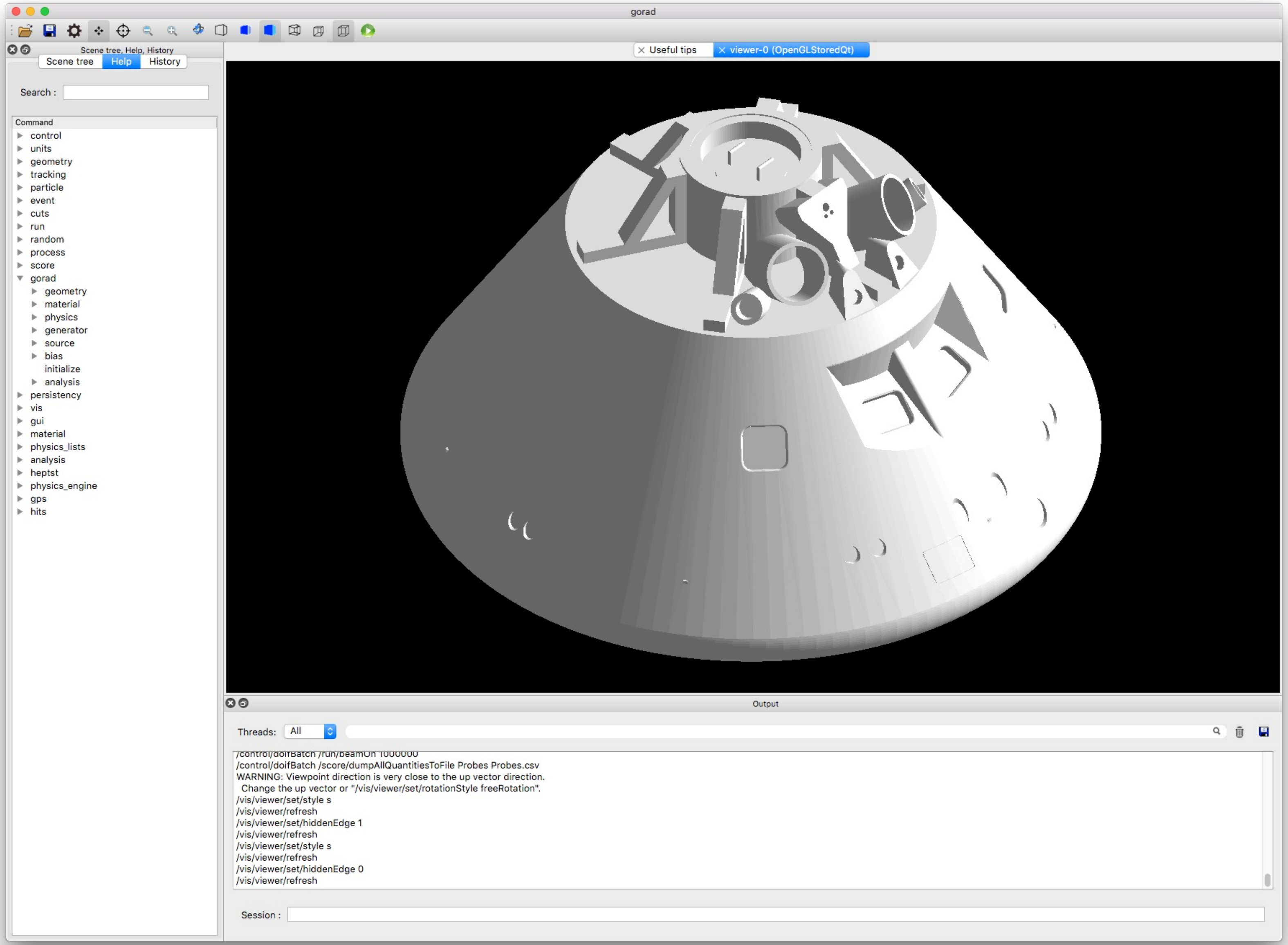
- CADMesh provides a G4TessellatedSolid object (a kind of G4VSolid).

```
#include “CADMesh.hh”
G4VPhysicalVolume* MyDetectorConstruction::Construct()
{
  …
  CADMesh cadMesh;
  G4VSolid* aSolid = cadMesh.LoadMesh( file_name, file_type );
  G4LogicalVolume* aLV = new G4LoficalVolume( aSolid, material, “name” );
  G4VPhysicalVolume* aPV = new G4PVPlacement( 0,
                    G4ThreeVector(x,y,z), aLV, “name”, motheLV, 0, 0 );
  …
```

Importing CAD geometries: more solutions …

- InStep: supporting import/export of various formats including STL, STEP and GDML
  https://www.solveering.com/InStep/instep.aspx
- SALOME: can import STEP BREP/IGES/STEP/ACIS, mesh it, export to STL (then STL2GDML)
  http://www.salome-platform.org
- ESABASE2: space environment analysis CAD, basic modules are free for academic non-commercial use. Exports to GDML shapes or complete geometry. Imports STEP.
  https://esabase2.net
- Blender GDML exporter: GDML plug-in for Blender tool
  http://projects.blender.org/tracker/index.php?func=detail&aid=30578&group_id=153&atid=467
- FASTRAD: 3D tool for radiation shielding analysis, exports meshes to GDML
  http://www.fastrad.net
- STEP Solutions: commercial, exports meshes GDML
  http://www.steptools.com/products/stdev/
- Cogenda TCAD: for the case of 3D meshes. Module Gds2Mesh exports to GDML
  http://www.cogenda.com/article/products#VTCAD
- SW2GDML: converts SolidWorks descriptions (using its API) to real primitives in GDML, including material
  https://github.com/cvuosalo/SW2GDMLconverter
- CadMC: tool to convert FreeCAD geometry to Geant4 (tessellated and CGS shapes)
  http://polar.psi.ch/cadmc/
- 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
- EDGE: CAD with direct editing/importing/exporting GDML with material definition
  http://www.space-suite.com/
- MRADSIM: CAD with direct editing/importing/exporting GDML with material definition
  https://www.mradsim.com/mradsim-converter/

Advanced example GORAD



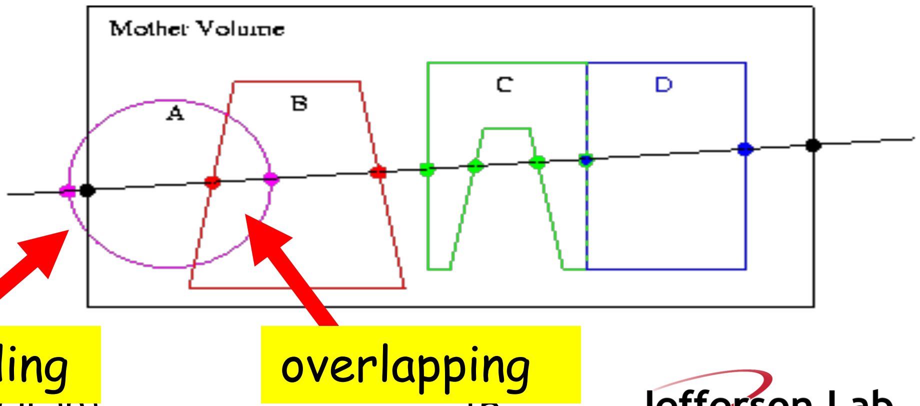
```
/control/doifBatch /run/beamOn 1000000
/control/doifBatch /score/dumpAllQuantitiesToFile Probes Probes.csv
WARNING: Viewpoint direction is very close to the up vector direction.
  Change the up vector or "/vis/viewer/set/rotationStyle freeRotation".
/vis/viewer/set/style s
/vis/viewer/refresh
/vis/viewer/set/hiddenEdge 1
/vis/viewer/refresh
/vis/viewer/set/style s
/vis/viewer/refresh
/vis/viewer/set/hiddenEdge 0
/vis/viewer/refresh
```

Contents

- GDML/CAD interfaces
- Geometry checking tools
- Geometry optimization
- Parallel geometry
- Moving objects

Debugging geometries

- An protruding volume is a contained daughter volume which actually protrudes from its mother volume.
- Volumes are also often positioned in a same volume with the intent of not provoking intersections between themselves. When volumes in a common mother actually intersect themselves are defined as overlapping.
- Geant4 does not allow for malformed geometries, neither protruding nor overlapping.
  - The behavior of navigation is unpredictable for such cases.
- The problem of detecting overlaps between volumes is bounded by the complexity of the solid models description.
- Utilities are provided for detecting wrong positioning
  - Optional checks at construction
  - Kernel run-time commands
  - Graphical tools (DAVID, OLAP)



Optional checks at construction

- Constructors of G4PVPlacement and G4PVParameterised have an optional argument "pSurfChk".

```
G4PVPlacement(G4RotationMatrix* pRot,
  const G4ThreeVector &tlate,
  G4LogicalVolume *pDaughterLogical,
  const G4String &pName,
  G4LogicalVolume *pMotherLogical,
  G4bool pMany, G4int pCopyNo,
  G4bool pSurfChk=false);
```

- If this flag is true, overlap check is done at the construction.
  - Some number of points are randomly sampled on the surface of creating volume.
  - Each of these points are examined
    - If it is outside of the mother volume, or
    - If it is inside of already existing other volumes in the same mother volume.
- This check requires lots of CPU time, but it is worth to try at least once when you implement your geometry of some complexity.

Debugging run-time commands

- Built-in run-time commands to activate verification tests for the user geometry are defined
  - to start verification of geometry for overlapping regions based on a standard grid setup, limited to the first depth level

    `geometry/test/run` or **`geometry/test/grid_test`**
  - applies the grid test to all depth levels (may require lots of CPU time!)

    `geometry/test/recursive_test`
  - shoots lines according to a cylindrical pattern

    `geometry/test/cylinder_test`
  - to shoot a line along a specified direction and position

    `geometry/test/line_test`
  - to specify position for the **`line_test`**

    `geometry/test/position`
  - to specify direction for the **`line_test`**

    `geometry/test/direction`

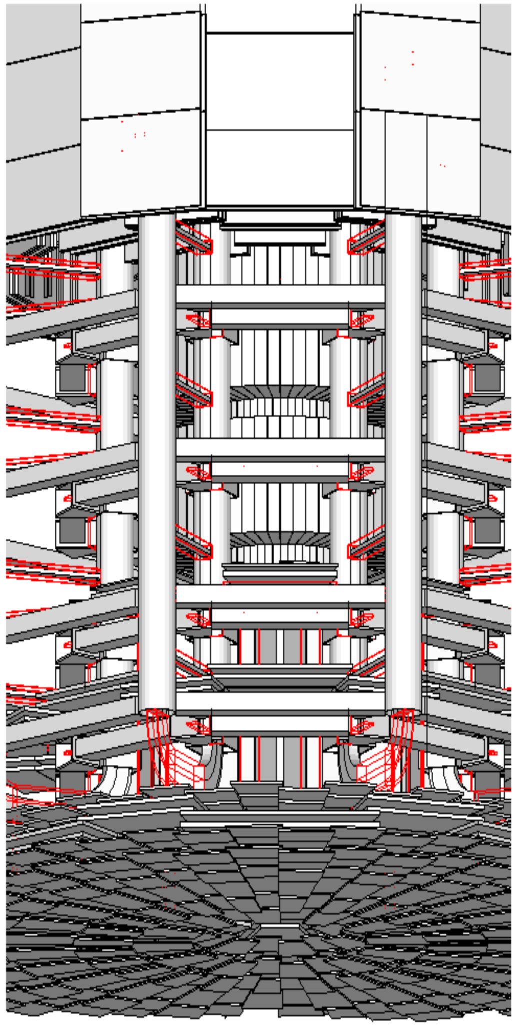
Debugging run-time commands

- Example layout:

```
GeomTest: no daughter volume extending outside mother detected.
GeomTest Error: Overlapping daughter volumes
    The volumes Tracker[0] and Overlap[0],
    both daughters of volume World[0],
    appear to overlap at the following points in global coordinates: (list
  truncated)
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    240            -240       -145.5      -145.5     0        -145.5      -145.5
Which in the mother coordinate system are:
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    . . .
Which in the coordinate system of Tracker[0] are:
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    . . .
Which in the coordinate system of Overlap[0] are:
  length (cm)    ----- start position (cm) -----  ----- end position (cm) -----
    . . .
```

Debugging tools: DAVID

- DAVID is a graphical debugging tool for detecting potential intersections of volumes
- Accuracy of the graphical representation car be tuned to the exact geometrical descriptior
  - physical-volume surfaces are automatically decomposed into 3D polygons
  - intersections of the generated polygons are parsed.
  - If a polygon intersects with another one, the physical volumes associated to these polygons are highlighted in color (red is the default).
- DAVID can be downloaded from the Web as external tool for Geant4
  - http://geant4.kek.jp/~tanaka/

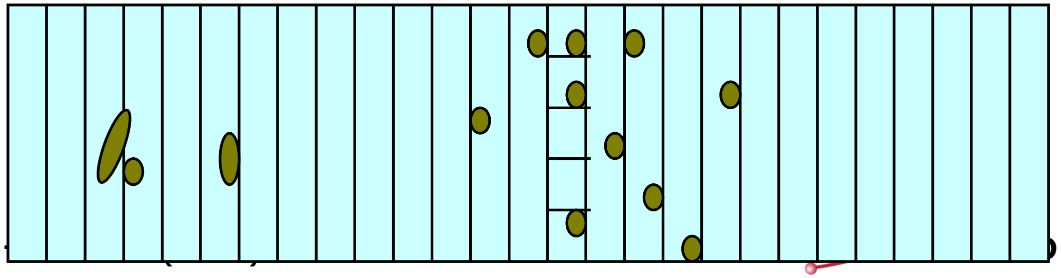


Contents

- GDML/CAD interfaces
- Geometry checking tools
- Geometry optimization
- Parallel geometry
- Moving objects

Smart voxelization

- In Geant4, user's geometry is automatically optimized to be most suitable to the navigation. - "Voxelization"
  - For each mother volume, one-dimensional virtual division is performed.
  - Subdivisions (slices) containing same volumes are gathered into one.
  - Additional division again using second and/or third Cartesian axes, if needed.
- *"Smart voxels"* are computed at initialisation time
  - When the detector geometry is *closed*
  - Does not require large memory or computing resources
  - At tracking time, searching is done in a hierarchy of virtual divisions



Detector description tuning

- Some geometry topologies may require 'special' tuning for ideal and efficient optimisation
  - for example: a dense nucleus of volumes included in very large mother volume
- Granularity of voxelization can be explicitly set
  - Methods **`SetSmartless()`** from **`G4LogicalVolume`**
- Critical regions for optimisation can be detected
  - Helper class **`G4SmartVoxelStat`** for monitoring time spent in detector geometry optimisation
    - Automatically activated if **`/run/verbose`** greater than `1`

```
Percent      Memory     Heads    Nodes   Pointers    Total CPU    Volume
-------      ------     -----    -----   --------    ---------    -----------
  91.70          1k         1       50         50         0.00    Calorimeter
   8.30          0k         1        3          4         0.00    Layer
```

Visualising voxel structure

- The computed voxel structure can be visualized with the final detector geometry
  - Helper class **`G4DrawVoxels`**
  - Visualize voxels given a logical volume

    ```
    G4DrawVoxels::DrawVoxels(const G4LogicalVolume*)
    ```
  - Allows setting of visualization attributes for voxels

    ```
    G4DrawVoxels::SetVoxelsVisAttributes(…)
    ```
  - useful for debugging purposes

Contents

- GDML/CAD interfaces
- Geometry checking tools
- Geometry optimization
- Parallel geometry
- Moving objects

Parallel navigation

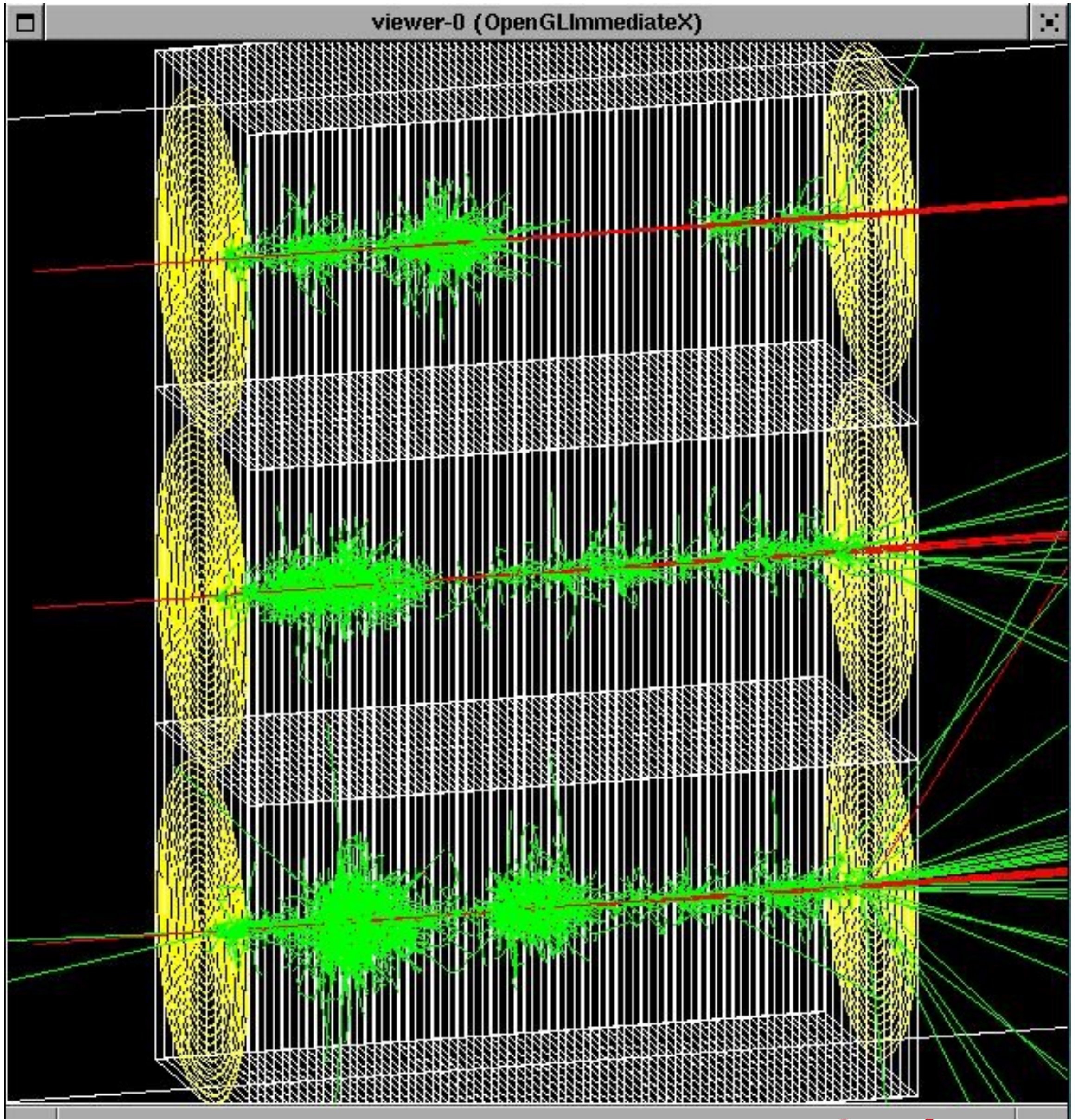
- Occasionally, it is not straightforward to define sensitivity, importance or envelope to be assigned to volumes in the mass geometry.
  - Typically, a geometry that is built machinery by CAD, GDML, DICOM, etc. has this difficulty.
- Parallel navigation functionality allows the user to define more than one worlds simultaneously.
  - G4CoupledTransportation process sees all worlds simultaneously.
  - A step is limited not only by the boundary of the mass geometry but also by the boundaries of parallel geometries.
  - Materials, production thresholds and EM field are used only from the mass geometry.
    - Exception for materials in “layered mass geometry”
  - In a parallel world, the user can define volumes in arbitrary manner with sensitivity, regions with shower parameterization, and/or importance field for biasing.
    - Volumes in different worlds may overlap.

Parallel navigation

- G4VUserParrallelWorld is the base class where the user implements a parallel world.
  - The world physical volume of the parallel world is provided by G4RunManager as a clone of the mass geometry.
  - All UserParallelWorlds must be registered to UserDetectorConstruction.
  - Each parallel world has its dedicated G4Navigator object, that is automatically assigned when it is constructed.
- Though all worlds will be comprehensively taken care by G4CoupledTransportation process for their navigations, each parallel world must have its own process to achieve its purpose.
  - For example, in case the user defines a sensitive detector to a parallel world, a process dedicated to this world is responsible to invoke this detector. G4SteppingManager sees only the detectors in the mass geometry. The user has to have G4ParallelWorldProcess in his physics list.

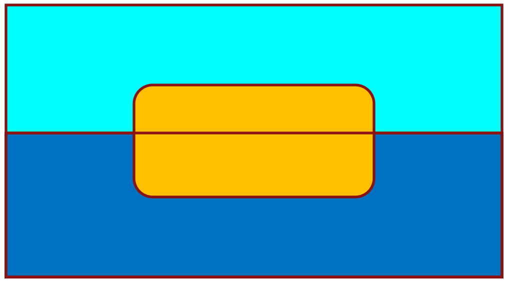
example/extended/runAndEvent/RE06

- Mass geometry
  - sandwich of rectangular absorbers and scintilators
- Parallel scoring geometry
  - Cylindrical layers

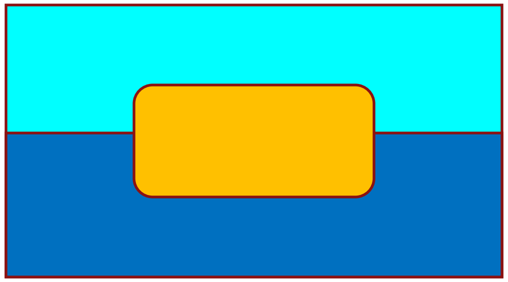


Layered mass geometries in parallel world

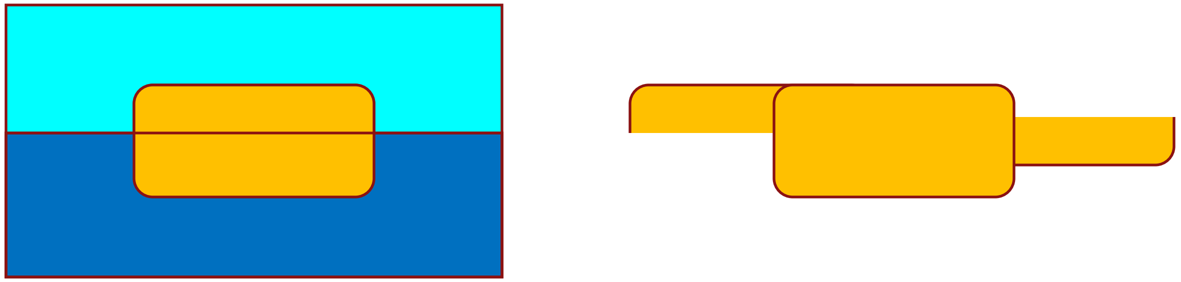
- Suppose you implement a wooden brick floating on the water.



- Dig a hole in water by a Boolean operation…

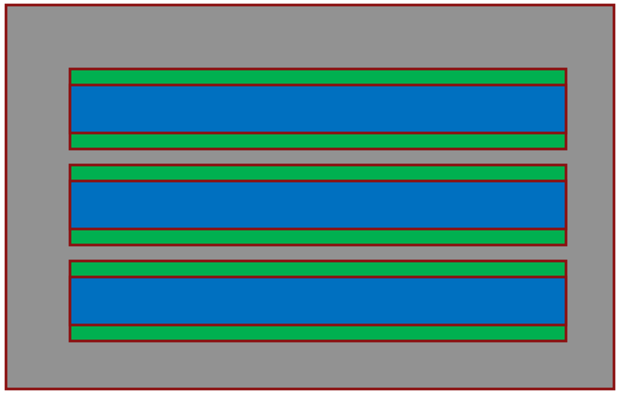


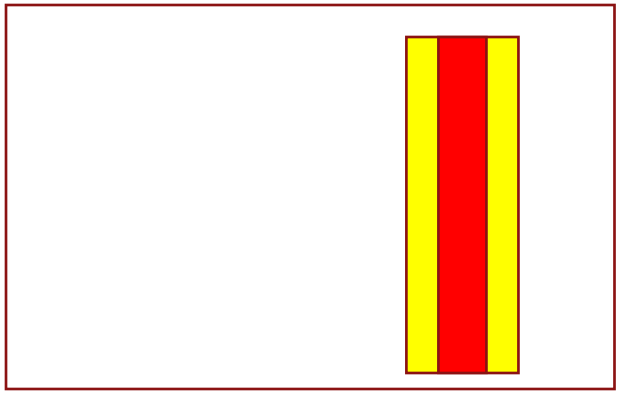
- Or, chop a brick into two and place them separately…



Layered mass geometries in parallel worlds

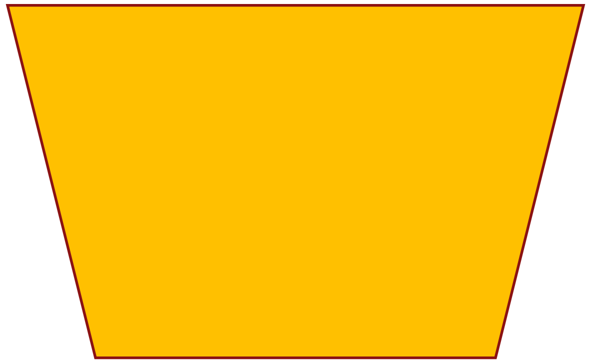
- Parallel geometry may be stacked on top of mass geometry or other parallel world geometry, allowing a user to define more than one worlds with materials (and region/cuts).
  - Track will see the material of top-layer, if it is null, then one layer beneath.
  - Alternative way of implementing a complicated geometry
    - Rapid prototyping
    - Safer, more flexible and powerful extension of the concept of "many" in Geant3

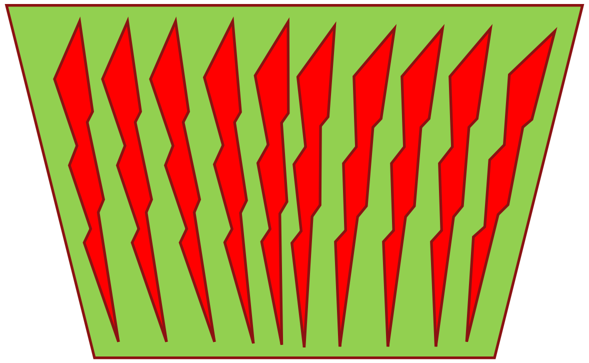

Mass world


Parallel world

Layered mass geometries in parallel worlds - continued

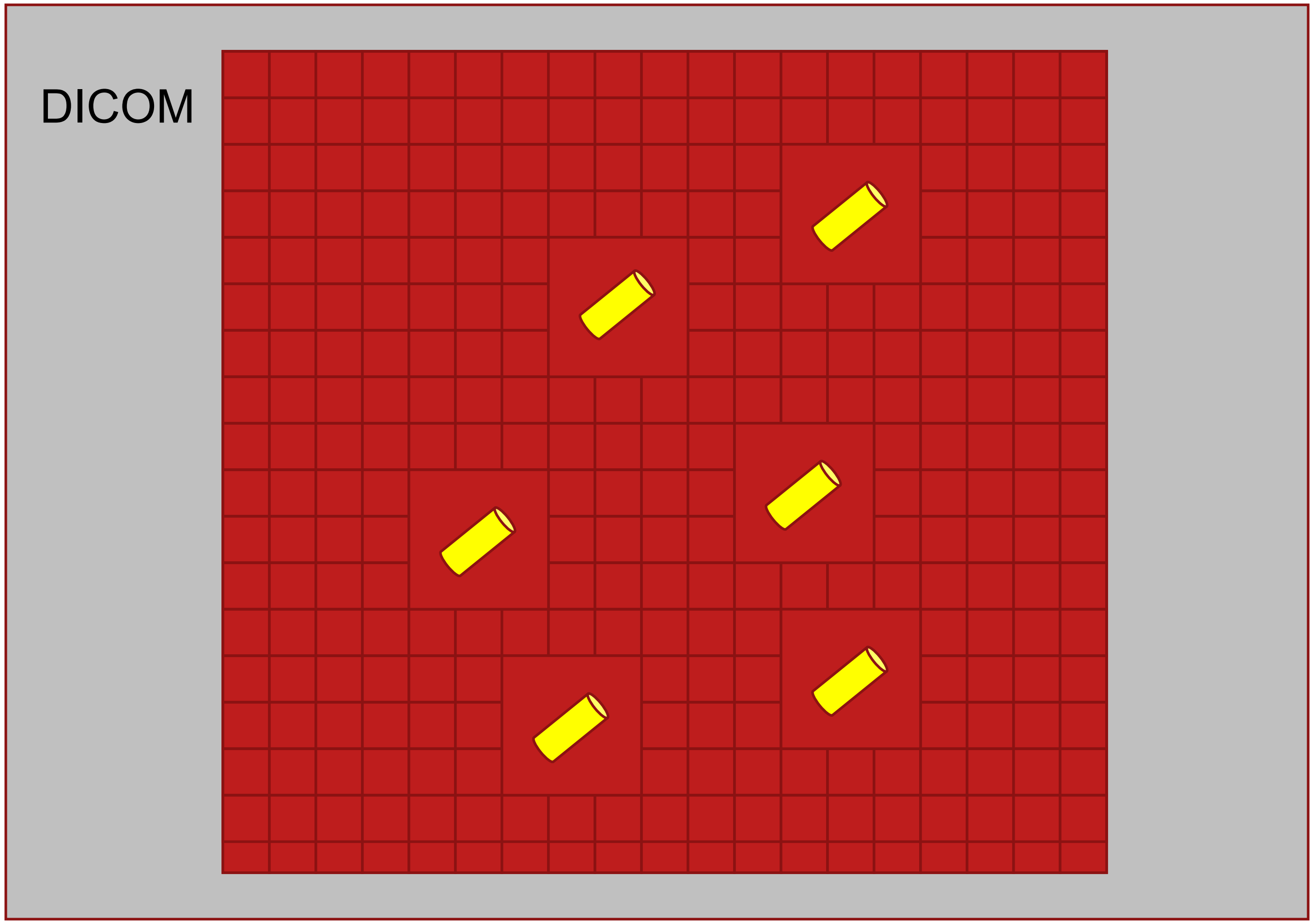
- A parallel world may be associated only to some limited types of particles.
  - May define geometries of different levels of detail for different particle types
  - Example for sampling calorimeter: the mass world defines only the crude geometry with averaged material, while a parallel world with all the detailed geometry. Real materials in detailed parallel world geometry are associated with all particle types except e+, e- and gamma.
    - e+, e- and gamma do not see volume boundaries defined in the parallel world, i.e. their steps won't be limited
  - Shower parameterization such as GFLASH may have its own geometry


Geometry seen by e+, e-, γ


Geometry seen by other particles

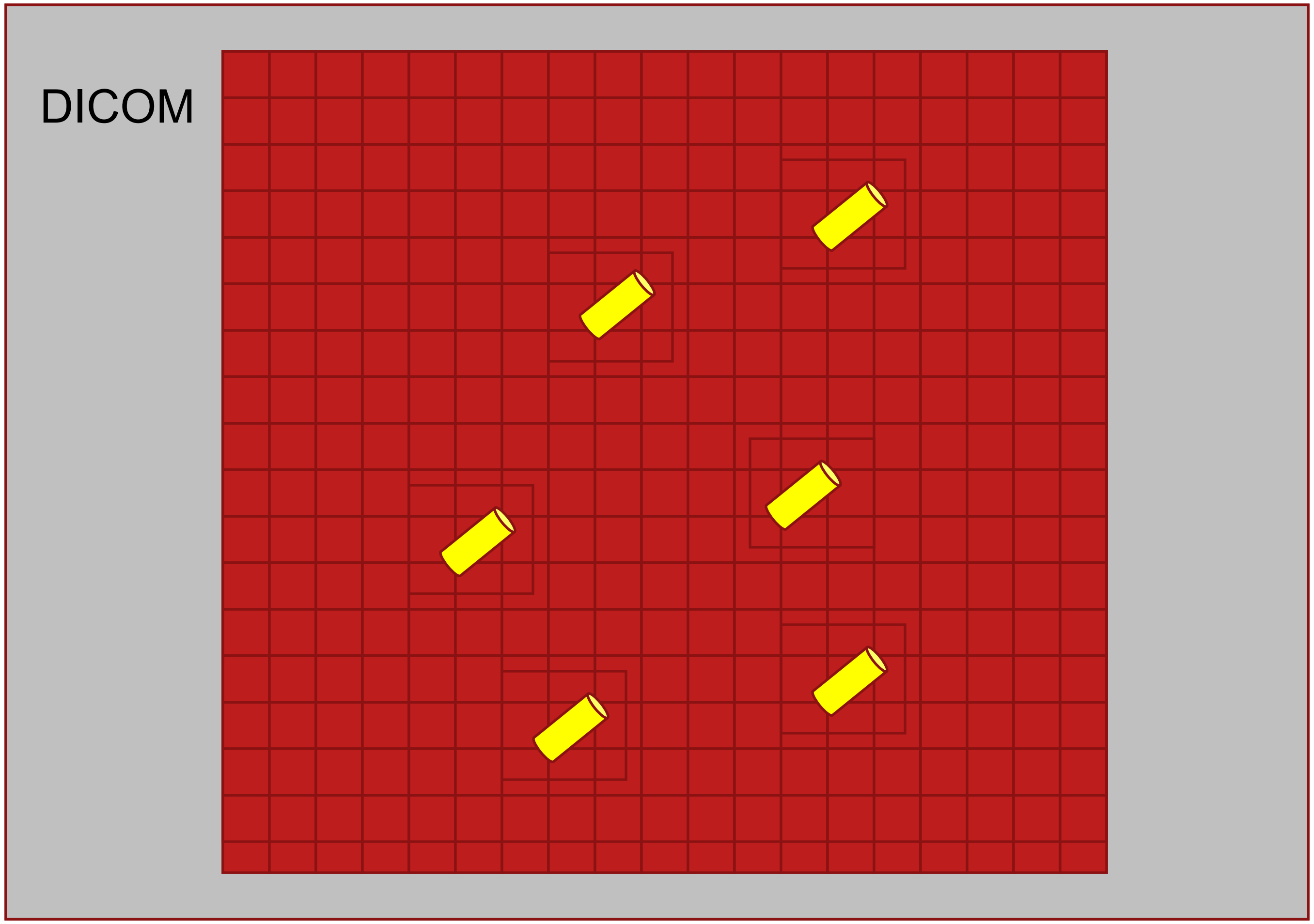
A medical use case

- Brachytherapy treatment for prostate cancer.



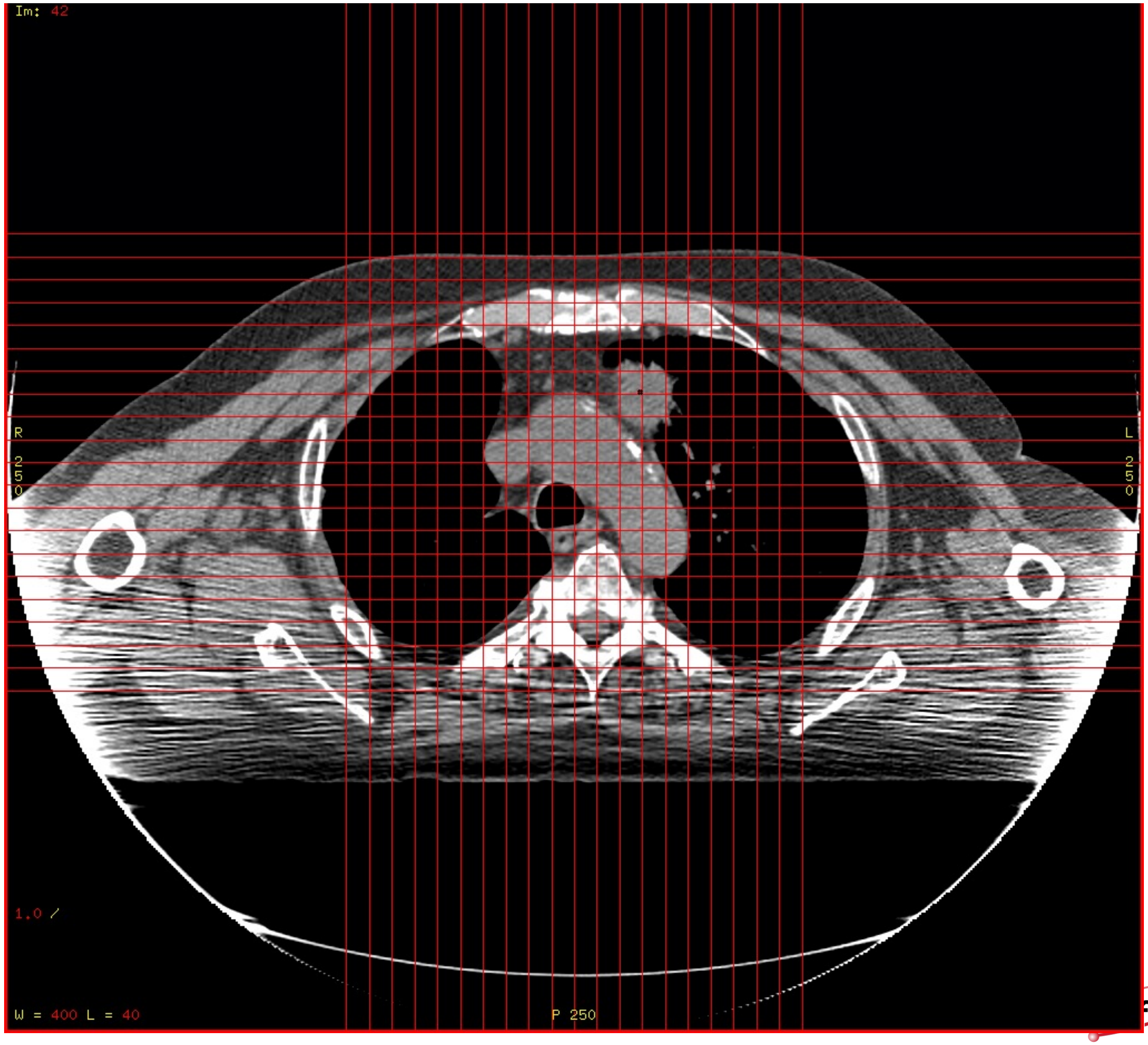
A medical use case

- Instead, seeds could be implemented in an empty parallel world.
  - Seeds in the parallel world would be encapsulated in empty boxes for faster navigation



Another important use case in medicine

- DICOM data contain void air region outside of the patient, while the treatment head should be placed as close as patient's body.

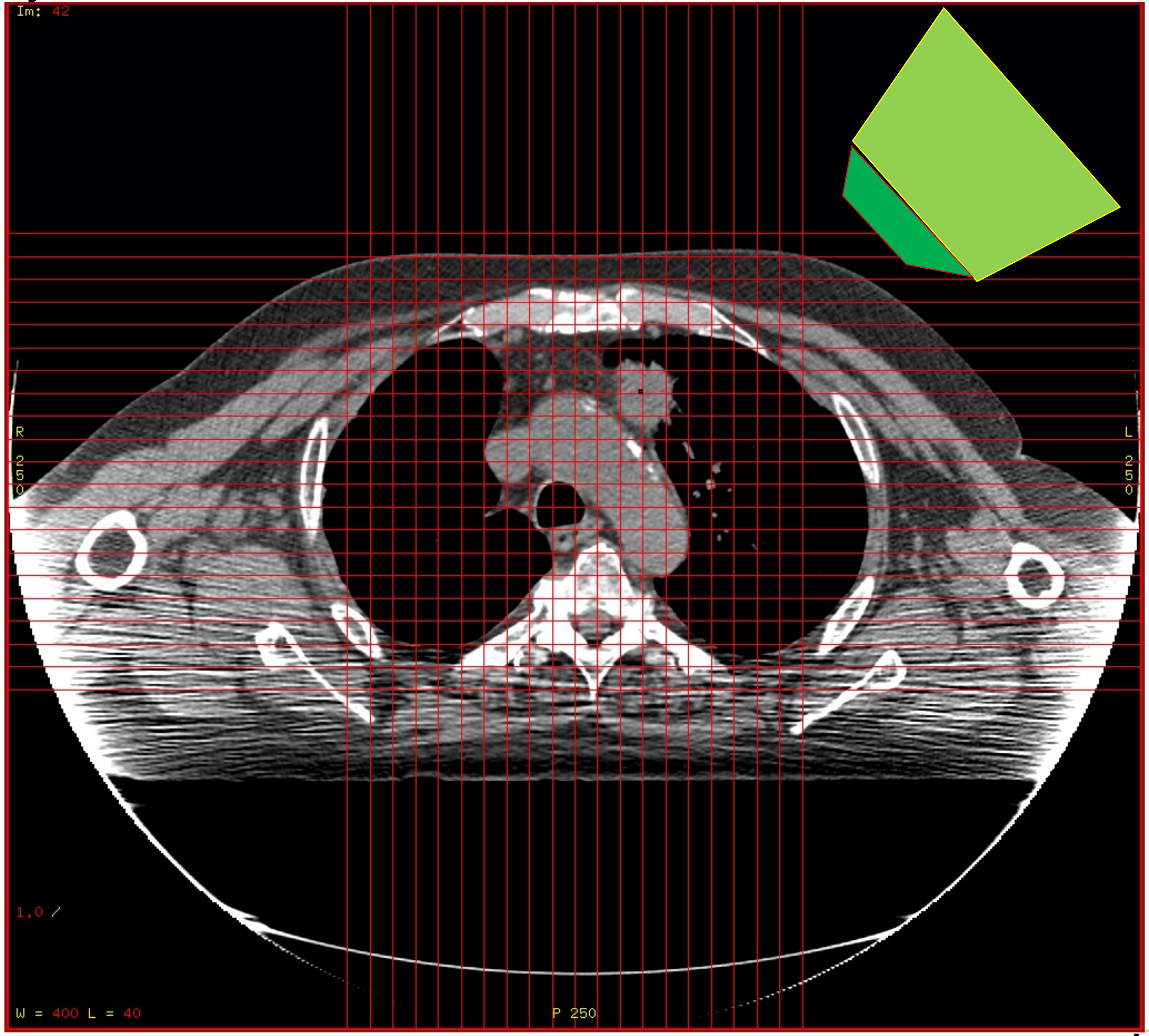


Another important use case in medicine

- Implement the treatment head in a parallel world.

Another important use case in medicine

- And overlay.



Defining a parallel world with layered mass geometry

main() (RE04.cc)

```
G4String paraWorldName = "ParallelWorld";
G4VUserDetectorConstruction* realWorld = new
                                  RE04DetectorConstruction;
G4VUserParallelWorldConstruction* parallelWorld
  = new RE04ParallelWorldConstruction(paraWorldName);
realWorld->RegisterParallelWorld(parallelWorld);
runManager->SetUserInitialization(realWorld);
//
G4VModularPhysicsList* physicsList = new FTFP_BERT;
physicsList->RegisterPhysics
    (new G4ParallelWorldPhysics(paraWorldName, true));
runManager->SetUserInitialization(physicsList);
```

Switch of layered mass geometry

– The name defined in the G4VUserParallelWorld constructor is used as the physical volume name of the parallel world, and must be given to G4ParallelWorldPhysics.

Defining a parallel world

```
void RE04ParallelWorldConstruction::Construct()
{
  // World
  G4VPhysicalVolume* ghostWorld = GetWorld();
  G4LogicalVolume* worldLogical = ghostWorld->GetLogicalVolume();
  // material defined in the mass world
  G4Material* water = G4Material::GetMaterial("G4_WATER");
  // parallel world placement box
  G4VSolid* paraBox = new G4Box("paraBox",5.0*cm,30.0*cm,5.0*cm);
  G4LogicalVolume* paraBoxLogical
                         = new G4LogicalVolume(paraBox, water, "paraBox");
  new G4PVPlacement(0,G4ThreeVector(-25.0*cm,0.,0.),paraBoxLogical,
                          "paraBox",worldLogical,false,0);
```

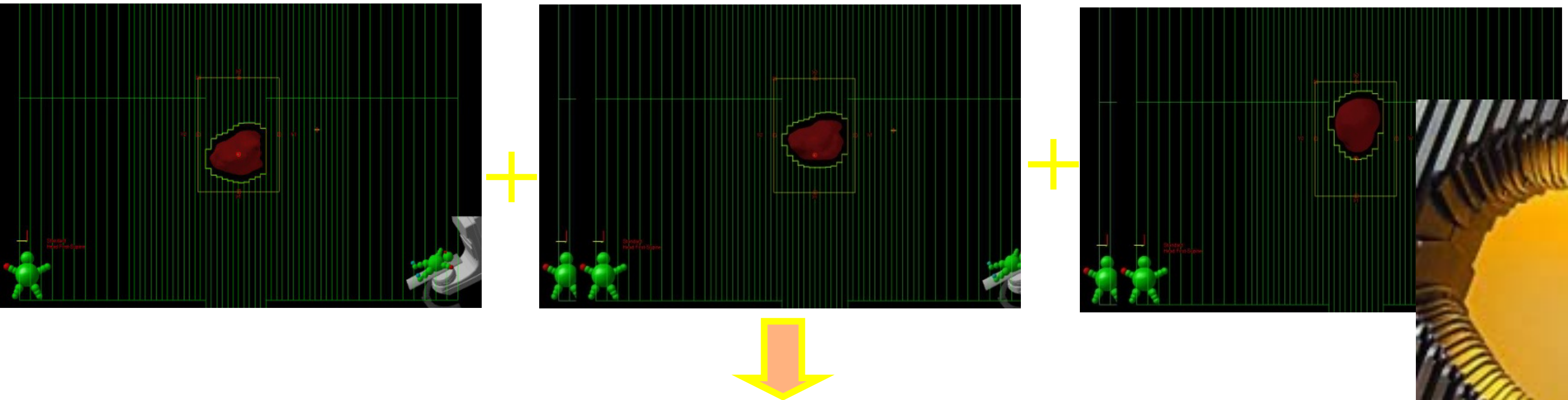
- The world physical volume of the parallel is provided as a clone of the world volume of the mass geometry. The user cannot create it.
- You can fill contents regardless of the volumes in the mass geometry.
- Logical volumes in a parallel world that do not overlay materials should have null material pointer.

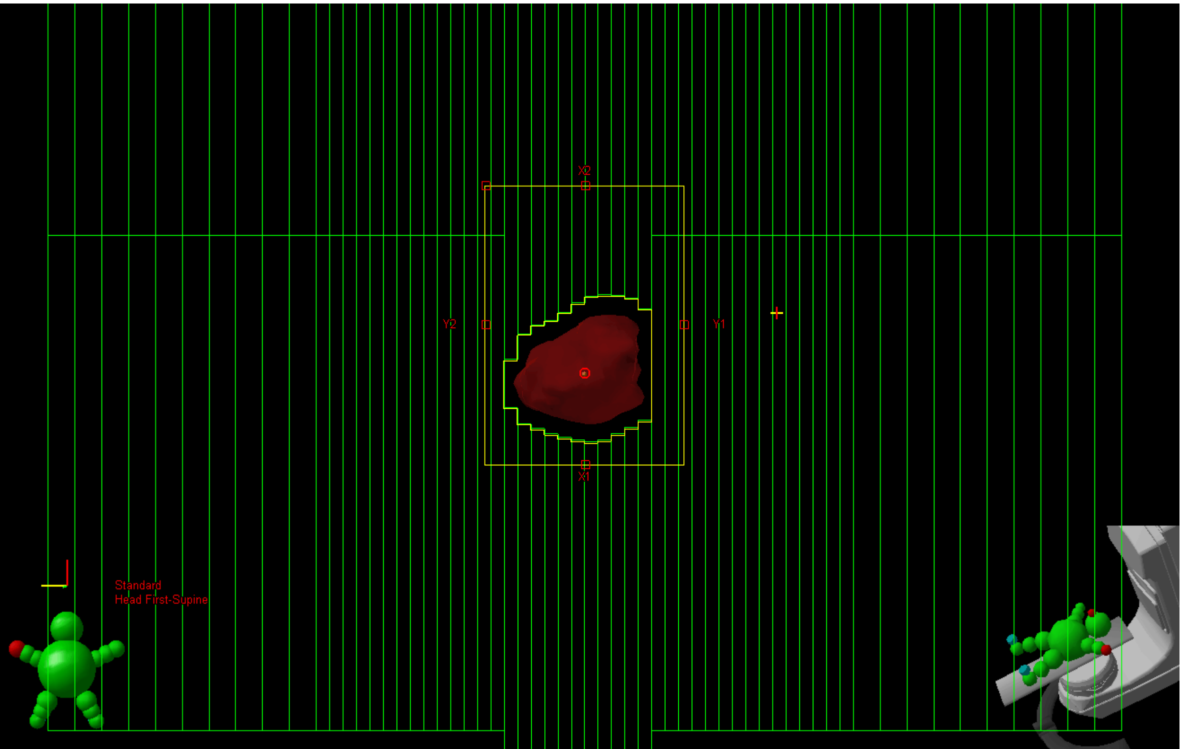
Contents

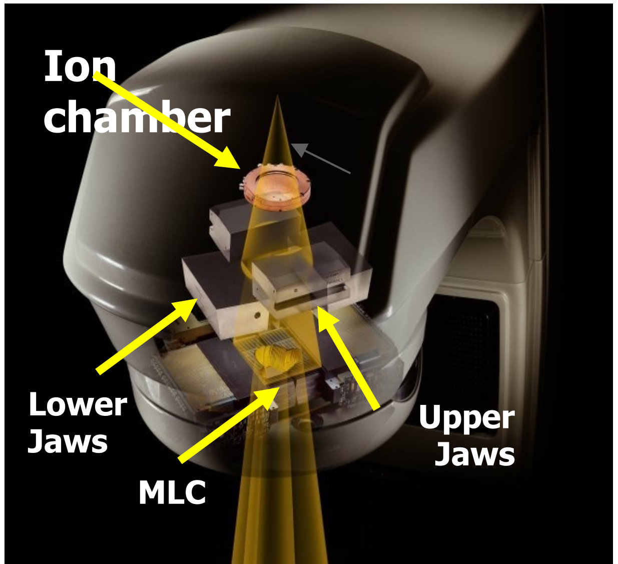
- GDML/CAD interfaces
- Geometry checking tools
- Geometry optimization
- Parallel geometry
- Moving objects

4D RT Treatment Plan

Source: Lei Xing, Stanford University



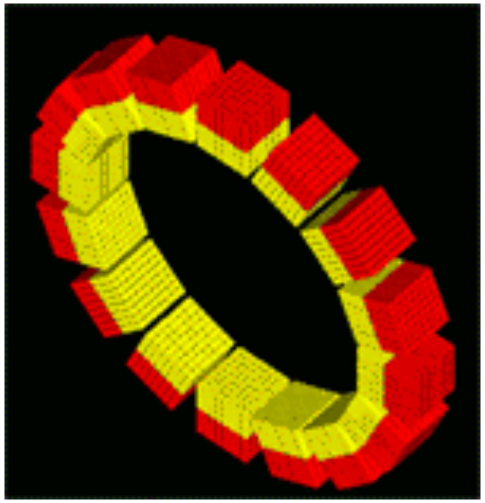




Moving objects

- In some applications, it is essential to simulate the movement of some volumes.
  - E.g. particle therapy simulation
- Geant4 can deal with moving volume
  - Speed of the moving volume is slow enough compared to speed of elementary particles, so that you can assume the position of moving volume is constant within one event.
- Two tips to simulate moving objects :

1. Use parameterized volume to represent the moving volume.
2. Do not optimize (voxelize) the mother volume of the moving volume(s).



Moving objects - tip 1

- Use parameterized volume to represent the moving volume.
  - Use event number as a time stamp and calculate position/rotation of the volume as a function of event number.

```
void MyMovingVolumeParameterisation::ComputeTransformation
    (const G4int copyNo, G4VPhysicalVolume *physVol) const
{
  G4RotationMatrix rMat;
  G4int eID = 0;
  const G4Event* evt =
      G4RunManager::GetRunManager()->GetCurren
  if(evt) eID = evt->GetEventID();
  G4double t = 0.1*s*eID;
  G4double r = rotSpeed*t;
  G4double z = velocity*t + orig;
  while(z>0.*m) { z -= 8.*m; }
  rMat.set(CLHEP::HepRotationX(-r));
  physVol->SetTranslation(G4ThreeVector(0.,0.,
  physVol->SetRotation(&rMat);
}
```

Null pointer must be protected.
This method is also invoked while
ed at
i.e.

You are responsible not to make
the moving volume get out of
(protr
e.

Here, event number is converted
to time.
(0.1 sec/event)

Position and rotation
are set as the function
of event number.

son Lab

Moving objects - tip 2

- Do not optimize (voxelize) the mother volume of the moving volume(s).
  - If moving volume gets out of the original optimized voxel, the navigator gets lost.

```
motherLogical -> SetSmartless( number_of_daughters );
```

  - With this method invocation, the one-and-only optimized voxel has all daughter volumes.
  - For the best performance, use hierarchal geometry so that each mother volume has least number of daughters.