What’s new in Geometry

Makoto Asai (SLAC PPA/SCA)
November 12th, 2014
Geant4 tutorial @ ANS Winter Meeting 2014
Geometry updates – New solid library

• An important effort was begun in the last couple of years to write a new solid library, reviewing at the algorithmic level most of the primitives and provides an enhanced, optimized and well-tested implementation.

• AIDA Unified Solids library - As optional component, for replacing the original solids (G4GEOM_USE_USOLIDS flag).

• It provides optimized implementation for a large number of geometrical primitives.
  – box, orb, sphere (+sphere section), tube (+cylindrical section), cone (+conical section), simple trapezoid, tetrahedron, polycone, polyhedra

• A shape can also be adopted individually by directly using provided G4Uxxx wrapper classes
New tessellated solid implementation

- In most cases considerable performance improvement was achieved.
  - For example, the time required to compute intersections with the tessellated solid was dramatically reduced with the adoption of spatial partitioning for composing facets into a 3D grid of voxels.

- Such techniques allow speedup factors of a few thousand for relatively complex structures having of order 100k to millions of facets, which is typical for geometry descriptions imported from CAD drawings.
  - Consequently, it is now possible to use tessellated geometries for tuning the precision in simulation by increasing the mesh resolution, something that was not possible before.

<table>
<thead>
<tr>
<th>Method</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inside</td>
<td>2423x</td>
</tr>
<tr>
<td>DistanceToIn</td>
<td>1334x</td>
</tr>
<tr>
<td>DistanceToOut</td>
<td>1976x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Information</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of facets</td>
<td>164.149</td>
</tr>
<tr>
<td>Number of voxels</td>
<td>100.000</td>
</tr>
<tr>
<td>Memory saved compared with original Geant4</td>
<td>22% (51MB)</td>
</tr>
</tbody>
</table>
Geometry updates – New “multi-union” solid

- In addition to a full set of highly optimized primitives and a tessellated solid, the library includes a new "multi-union” structure implementing a composite set of many solids to be placed in 3D space.

- This differs from the simple technique based on Boolean unions, with the aim of providing excellent scalability on the number of constituent solids.

- The multi-union adopts a similar voxelization technique to partition 3D space, allowing dramatically improved speed and scalability over the original implementation based on Boolean unions.
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
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 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.
Converting STEP to GDML

• Imperfect, but still helpful solutions are tools to convert STEP to GDML and provide the user a way to add materials information
• There are two cases where existing CAD programs have added GDML export features. Since these CAD programs can also read in STEP, they can be used as STEP to GDML converters.
  – Neither option is free, neither option works perfectly
  – ST-Viewer
    • [http://www.steptools.com](http://www.steptools.com)
  – FastRad
    • GDML export extension was funded by European Space Agency
    • Not free except for limited, trial mode that can handle only a small number of volumes
    • [http://www.fastrad.net/](http://www.fastrad.net/)
• Discussion of these solutions takes place in the Geant4 Persistency forum:
• Useful technical note:
Parallel layered mass geometry

• In the past, material is considered only if it appears in the mass (tracking) world. The user might define parallel world(s) for artificial purposes, i.e. shower parameterization envelopes, readout/scoring geometry, dedicated stepping action.

• Now, the user may define a material in parallel world(s) which is also seen by physics processes.
Layered mass geometries in parallel world

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

• Dig a hole in water…

• Or, chop a brick into two and place them separately…
How it works

• A step is limited on the boundary of any volume of any world.
• The step (and all physics processes) sees the material defined in the top-most layer. If the top-most layer has null pointer to material, material in next layer is used.
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
    - Avoid complicated Boolean operations
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
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.
How to use it – Parallel world

Tst1ParallelWorldConstruction::
    Tst1ParallelWorldConstruction(G4String& parallelWorldName)
    :G4VUserParallelWorld(parallelWorldName)
    {};
void Tst1ParallelWorldConstruction::Construct()
{
    // World
    G4VPhysicalVolume* ghostWorld = GetWorld();
    G4LogicalVolume* worldLogical = ghostWorld->GetLogicalVolume();
    G4Material* water = G4Material::GetMaterial("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);

In the parallel world, define valid material pointers only for the volumes you want to overlay. Material pointers of other volumes must be set to null.
int main(int argc, char** argv)
{
    G4MTRunManager * runManager = new G4MTRunManager;
    G4String paraWorldName = "ParallelWorld";
    // Geometry
    MyDetectorConstruction* realWorld = new MyDetectorConstruction;
    MyParallelWorldConstruction* parallelWorld =
        new MyParallelWorldConstruction(paraWorldName);
    realWorld->RegisterParallelWorld(parallelWorld);
    runManager->SetUserInitialization(realWorld);
    // Physics list
    G4VModularPhysicsList* physicsList = new FTFP_BERT;
    physicsList->RegisterPhysics(
        (new G4ParallelWorldPhysics(paraWorldName, true)));
    runManager->SetUserInitialization(physicsList);