The Geant4 Geometry Modeler

G. Cosmo, CERN

***Abstract*— The Geometry modeler in Geant4 (a software toolkit for the simulation of the interactions of particles with matter) is a key component of the Geant4 software. It offers the ability to describe the geometrical structure of a detector in a natural way, and has been designed for allowing efficient propagation of particles in the geometrical detector model and exploit at the best the features provided by the Geant4 simulation toolkit. Advanced techniques for optimizing tracking in the geometrical model have been seriously taken into consideration in the design, both in order to optimize the run-time performance and reduce the physical memory consumption when dealing with complex geometrical setups. A great variety of geometrical shapes are defined, including the possibility of combining them with Boolean operations to achieve nearly any possible detector geometry configuration. The major concepts of the Geant4 geometry modeler are here reviewed, with emphasis on recent features introduced in the last releases of the software.**

***Index Terms*— Geant4, Logical volume, Physical volume, Voxels.**

I. Modeling a Detector

The Geometry module of Geant4 offers to the simulation user the ability to describe the geometrical structure of a detector and allows to the system to propagate particles efficiently in the detector model. The Geant4 geometry modeler includes many advanced concepts in key areas in order to cope with the great number and different organisation of detector volumes now possible.

Important considerations in the design were needed for performant navigation and for facilitating as much as possible also exchange conversion of simple detector geometries developed with CAD systems.

A detector's geometry is described by listing the different elements it contains and specifying their positions and orientations. Key concepts for modelling geometries in Geant4 are those of *logical* and *physical* volume. A logical volume represents a detector element of a certain shape, can hold other volumes inside it and can have different attributes associated; it manages the information associated to detector elements represented by a given solid and material, independently from its physical position in the detector. A logical volume can also represent the envelope for other logical volumes included, holding attributes affecting the physics which will take place in its elements: a specific *electromagnetic field* can be associated, or specific production cuts dependent on the particle interacting with matter inside it (*region*).

A physical volume represents the spatial positioning of the volumes describing the detector elements, as positioned with respect to an enclosing (logical) volume.

Structures of the detector that are repeated can be described as one logical volume and placed in several places and great memory saving for complex structures can be achieved. Volumes can be replicated according to a regular structure or can be parameterized according to a precise mathematical formula applied to their shape, positioning or attributes, such that only one instance of the physical volume will be created in memory.

Irregular combined geometry structures can be *assembled* together and easily placed multiple times. A whole hierarchy of volumes can be *reflected*, by specifying an axis plane.

The concept of a logical volume's shape and dimensions is separated and represented in a separate entity, the *solid*. Solids with simple shapes, like rectilinear boxes, trapezoids, spherical and cylindrical sections or shells, are available directly as solid objects, according to the Constructed Solid Geometry (CSG) specifications. Other more specific solids are provided, like elliptical tubes, hyperbolic tubes and twisted shapes (tube, box, trapezoid). More complex solids can be defined by their bounding surfaces, which can be planes or second order surfaces [4]: *Boundary REPresentations (BREPs)*.

Another way to obtain solids is by combining simple solids using boolean operations, like unions, intersections and subtractions. Creating such a *boolean solid* may also require an optional transformation for one of the two solids involved. The solids used should be either CSG solids (for examples a box, a spherical shell, or a tube) or another boolean solid (the product of a previous boolean operation). By properly using boolean operations one can describe boolean solids with particular shapes in a simple and natural way with very efficient geometrical navigation inside them.

For all shapes it is possible to compute the geometrical volume of the solid, either if a simple solid or a boolean composition; for the case of complex compositions or complex shapes, the value is estimated adopting a specific Monte Carlo technique.

As described above, a detector can be described by utilising a hierarchy of volumes, each of which can contain smaller volumes. Or it can be specified by listing all the volumes one by one, with no volume containing another. The former way can be utilised successfully to provide performant simulations.

Geant4 also provides built-in tools for helping a user modelling his/her geometry in detecting potential mistakes, like volume overlaps for which Geant4 tracking algorithms are quite sensitive and in general not tolerant. Volumes are defined as overlapping when they actually protrude from their mother-volume or when volumes in a common mother-volume actually intersect themselves. The tools implemented in the geometry modeler allow to detect most of these erroneous setups.

II. Tolerant Geometry

An important task of the geometry module in Geant4 is also to handle the geometrical propagation of tracks and find

their intersections with volume boundaries even in the case of charged particles in a magnetic field. This geometrical or "pure" tracking consists of three steps:

1) Identify the solid in which the particle is inside.
2) Calculate the intersection distance to volumes inside of the current solid, and to the current solid's boundary (exiting distance).
3) Calculate the minimum step, and move the particle.

Geant4's propagation methods were designed to overcome limitations present in previous simulation packages (where particles were sometimes moved by small steps in order to cross a volume boundary) and to provide accuracy and efficiency. To minimise the number of geometrical calculations, tracks are propagated exactly to boundaries and their state is stored: whether it is on a boundary, whether it is exiting the current volume, etc.

Volume boundaries are effectively given a very small but finite 'thickness' to take into account the roundoff and accumulated errors of floating point arithmetic. Thanks to this "tolerant" geometry [3], intersections with boundaries less than the tolerance from the point will be ignored if the direction of the particle is away from the boundary. The thickness (or tolerance) is chosen to be very small compared to detector features but much larger than the expected arithmetic errors.

The number of steps a particle must take to traverse a detector is therefore much reduced. However, in order to efficiently traverse a detector model geometry, it is critical to reduce the number of intersections of candidate volumes made in each step. As intersections are potentially costly operations, we seek an optimisation method to minimise the number of them required.

III. Optimisation Techniques

While tracking through the detector, a particle may encounter any one of several detector parts at each step. Calculating the intersection of a track with every daughter volume at each tree level would be extremely inefficient. Different methods also inspired by techniques used in ray tracing, have been studied to optimise the process by lowering the number of candidate volumes to be tested for intersection.

A possible technique is the one adopting *virtual divisions*, which consists of having mother volumes sliced evenly along one axis into sections. Each section stores pointers to the sub-volumes it contains, compressed by bunching together common lists. This scheme works well in a hierarchical detector description, where the number of structured daughter volumes is small. It will fail though for 'flat' geometries, where a large number of sub-volumes may exist and the level of detail in different regions of the model detector varies greatly.

Other techniques based on *fixed* or *variable grids* foresee storing for each cell of the grid a list of pointers to the volumes intersected. The contents of each cell can be determined entirely at initialisation time, based on the bounding box of each volume. Fixed size grids have the disadvantage of memory consumption versus finer granularity dictated by small detector components. The usage of variable grids would overcome this problem but would make more complex intersection tests and determining within which cell a particle is inside.

In Geant4 a different optimisation technique is adopted. The technique is derived from the *voxels*-based method also adopted in ray tracing, where space is subdivided into cubical volume elements (voxels) and a tree based map is created by recursively dividing the detector into octants. A traditional voxel based technique retains the disadvantage of grid based methods in that every voxel intersected along the particle's trajectory must be tested for intersection. In the *smart voxels* technique [2] adopted in Geant4, for each mother volume, a one dimensional virtual division is performed. The best axis for the virtual division is chosen by using a heuristic where equal subdivisions are gathered into single bins. Each division containing too many volumes is then refined by applying virtual division again, but using a second Cartesian axis. If the resultant subdivisions contain too many volumes, a further refiniment can be performed by dividing again along a third Cartesian axis.

If a hierarchy of volumes exists, the local axes introduced typically aid in producing efficient space decomposition. For a "flat hierarchy" smart voxels can produce either a simple virtual division if volumes are regular and suitably placed, or a deep tree if it contains many volumes of different sizes and placements. Such technique allows to greatly improve runtime performance for "pure" tracking, up to 20 times for "flat" geometries compared to the traditional techniques adopted in previous simulation packages.

Smart voxels are computed at initialisation time, and require small memory and computing resources. At tracking time searching is done in the hierarchy of virtual divisions. This method for pure tracking was found to be very well performant and does not require the need to tune detector description parameters.

IV. Persistency of Geometry Models

There are several way by which a user can define the actual geometry description in Geant4: either by directly coding it in pure C++, making use of the APIs defined in the Geant4 toolkit; either defining it by means of the GAG Java visual tool [5] and then directly generating the related C++ code to be integrated; either by importing/exporting the geometry model with GDML (Geometry Description Markup Language) [6].

GDML is a markup language based on XML and is aimed to define a common approach for sharing and exchanging of geometry description data. Through GDML it is possible in Geant4 to store persistently on an ASCII file all the information related to a geometry model: material definitions, solids and volumes compositions, including replications and size/positioning parameterisations.

V. Conclusion

The geometry modeler of Geant4 is capable of describing complex geometries made of a combination of a large variety of shapes. Geometrical shapes are represented as separate entities (*solids*) which define the *logical* structure of the *volumes* positioned in the model.

It provides means by which memory consumption can be greatly reduced, and regular or irregular patterns can be easily replicated, assembled or reflected. Efficient navigation and optimisation techniques are provided to allow for exact geometrical propagation of tracks in the simulation, and fast detection of intersections in generic geometries.

Geometry models can be easily exchanged by importing/exporting their GDML descriptions.

REFERENCES

[1] S. Agostinelli et al, Geant4 Collaboration, Nuclear Instruments and Methods in Physics Research, A 506 (2003) 250.
See also the Geant4 web page: *cern.ch/geant4* .

[2] Pure Tracking and Geometry in Geant4, P.Kent, April 1995, CERN note.

[3] Minimising Precision Problems in Geant4 Geometry, P.Kent, April 1995, CERN note.

[4] GEREP, a Boundary Representation Modeler proposal for Geant4, J.Sulkimo and J.Vuoskoski, IT Division Internal Report, CERN.

[5] M. Nagamatsu, T. Kodama, H. Uno, H. Yoshida, K. Ohtsubo, S. Tanaka, M. Asai, Proceedings of the CHEP '98 Conference, Chicago, September 1998.

[6] R.Chytracek, Proceedings of the CHEP '01 Conference, Bejing, September 2001. See also the GDML web page: *cern.ch/gdml* .