THE GEOMODEL TOOLKIT FOR DETECTOR DESCRIPTION

J. Boudreau, V. Tsulaia, University of Pittsburgh, PA 15260, USA

Abstract

The GeoModel toolkit is a library of geometrical primitives that can be used to describe detector geometries. The toolkit is designed as a data layer, and especially optimized in order to be able to describe large and complex detector systems with minimum memory consumption. Some of the techniques used to minimize the memory consumption are: shared instancing with reference counting, compressed representations of Euclidean transformations, special nodes which encode the naming of volumes without storing name-strings and, especially, parameterization though embedded symbolic expressions of transformation fields. A faithful representation of a GeoModel description can be transferred to Geant4, and, we predict, to other engines that simulate the interaction of particles with matter. Native capabilities for geometry clash detection and for material integration are foreseen for the near future. GeoModel's only external dependency is CLHEP.

INTRODUCTION

The GeoModel toolkit provides application developers with a complete set of mechanisms for the description of large and complex detector geometries with minimal memory consumption. The main purpose of GeoModel is to support a central store for detector description information that can be accessed by two main clients – Simulation and Reconstruction programs. GeoModel provides a scheme for accessing both the raw geometry of a detector and arbitrary subsystem-specific geometrical services synched to the raw geometry, while incorporating time-dependent alignments.

GeoModel-based applications can build the description of detector geometry by reading primary numbers from a single database. In case the relational database schema allows versioning of primary numbers, GeoModel descriptions can also be versioned. An automatic version detection system has also been developed.

In ATLAS, visual debugging tools developed with the use of Open Inventor toolkit complement the GeoModel toolkit. These debugging tools include an interactive Geometry Browser that allows various manipulations with volumes: navigation of volume hierarchy, iconization, printing of volume characteristics such as mass, shape dimensions, name and copy number. When used together with simulated data the Browser can also visualize tracks and hits on top of raw geometry.

In this paper we describe design principles of the GeoModel toolkit, memory optimization techniques and also the mechanism of converting GeoModel description to GEANT4.

DESIGN PRINCIPLES

Material geometry

Material geometry consists of a set of classes that bears a large resemblance to the GEANT4 material geometry classes. These classes, however, are designed to take a minimal size in memory. This requirement determines the basic data structure used to hold the data for the geometry description. That structure is a graph of nodes consisting of both physical volumes and their properties (name, identifier, transformation in local coordinate system). The tree is built directly and accessed in a way that provides users with access to volumes and, simultaneously, to the properties accumulated during graph traversal that apply to the volumes.

Physical volumes are the main building blocks of the geometry graph. The structure of a physical volume consists of an associated logical volume (describing just shape and material) and a set of child physical volumes and their properties. In GeoModel we distinguish two types of physical volumes:

- Regular Physical Volumes, designed to be small;
- Full Physical Volumes, designed to hold in cache complete information about how the volume is located with respect to the world volume, its formatted name string and other important information.

When one adds a transformation in the graph, it changes the position of the subsequent volume with respect to the parent. If one adds more than one transformation to the volume before adding a parent, they will be multiplied. The last transformation to be added is applied first to the child. Like physical volumes, transformations come in two types:

- Regular transformations, designed to be small;
- Alignable transformations, which allow one to add a misalignment to the system. Misaligning a transformation changes the position of all volumes "under" the transformation and clears the absolute location caches of all full physical volumes.

GeoModel includes also three mechanisms for giving names to physical volumes:

1. Do nothing, the volume will be called "ANON";
2. Add a **Name Tag** object to the graph before adding a volume, the next volume to be added will be given the Tag's name;
3. Add a **Serial Denominator** object to the graph before adding more volumes. The volumes will be named according to the base name of the Denominator, plus given a serial number 0, 1, 2 …

Readout geometry

The readout geometry layer consists of geometrical information that is not declared directly to the tracing engines (GEANT4), for example: projective towers within a calorimeter, or the boundaries of ion implant layers in silicon detectors. Information such as the position of the boundaries of these regions is not required in the simulation of basic physics processes, though it certainly is required in the digitization, and possibly hit-making phase of simulation (Sensitive Detectors).

Detector-specific geometrical services can and should include services that derive from the basic raw and readout geometry of the detector. Such services could include point-of-closest-approach calculations, global-to-local coordinate transformations, calculations that compute the total number of radiation lengths within a cell etc.

The description of detector readout system can be realized with the use of Detector Elements. The detector element has a required association with a piece of material geometry (full physical volume), and has access to that piece. The rest of the interface – all of the geometrical services discussed above, such as the boundaries of implant layers, strip pitches, or whatever, can be placed in the detector element.

Interface to Detector Description clients

In GeoModel applications the Detector Manager objects play a central role in providing an interface to Detector Description clients. Detector Managers manage all raw and readout geometry and should provide a fast mechanism for accessing the detector elements in a detector-specific way.

So in general the subsystems developers have a lot of flexibility, but need to devise an interface to both the detector manager and the detector element that satisfies their needs. The basic framework requires only that

1. Special Detector Factory objects create a physical volume tree;
2. They associate readout elements to certain physical volumes;
3. Additional readout information appear in the interface to the detector manager and the detector element.

MEMORY OPTIMIZATION TECHNIQUES

The requirement of minimizing the memory consumption has led us to foresee a system in which objects in the detector description can be re-used. This is called shared instancing. It essentially means that an element, compound, volume, or entire tree of volumes may be referenced by more than one object in the detector description. We can list following use cases for shared instancing: few physical volumes can share the same logical volume, the same name tags and identifier tags can label different volumes, and the transformations can be used more than once in the geometry graph.

GeoModel includes few other memory optimization techniques: Serial Denominators can generate name strings for physical volumes so that the memory does not fill up with nearly identifical ASCII name tags; tiny" transforms (where the footprint for a simple translation along z, for example, is the size of one floating point number) reduce the size requirement for most transformations; finally, volumes can be parameterized (this mechanism is discussed in next section).

```
double a, radius, z;
Variable icpy;
GENFUNCTION gama = Gama0+a*icpy;
TRANSFUNCTION xf = Pow(HepRotateZ3D(1.0),gama)*
                   HepTranslateX3D(radius)*
                   HepTranslateZ3D(z);

GeoSerialTransformer *st= new GeoSerialTransformer(vol, &xf, N);
parent->add(st);
```

Figure 1: Example of the usage of parameterized volumes in GeoModel application

Parameterizations

Parameterizations are mathematical recipes for creating volumes. There are three main ingredients to these recipes:

- **GENFUNCTIONS**, which are mathematical function-objects; they allow one to perform function arithmetic in the same way that one performs floating point arithmetic.
- **TRANSFUNCTIONS**, which, together with `GENFUNCTIONS` and HepTransform3D, allow one to expand and parameterize elements of the Euclidean group (i.e., rigid body transformations).
- **Serial Transformers**, a kind of GeoModel graph nodes, which allow a particular physical volume to

be placed according to a TRANSFUNCTION expansion of a rigid body transformation

The example presented on Figure 1 demonstrates the usage of parameterizations in GeoModel applications.

The first step is to define a GENFUNCTION. Then a TRANSFUNCTION is constructed, which parameterizes the rigid body transformation. The expansion of the TRANSFUNCTION is as follows. Let X_i (i = 1, 2 … N) represent any transformation. Furthermore, let us denote by $f_i(x)$ a function of a single variable. Then, the expansion of an arbitrary function is:

$$T(x) = X_1^{f1(x)} * X_2^{f2(x)} * X_3^{f3(x)\ldots} X_n^{fn(x)}$$

In this expression, $T(x)$ is the resulting transformation, which is now a function of the single input parameter, x. The expansion is both simple, and completely general. A single term in this expansion (for example $X_2^{f2(x)}$), will be referred to as an *exponentiated transformation.* Exponentiated transformations are simple TRANSFUNCTIONs, and can be composed to make other TRANSFUNCTIONS. The TRANSFUNCTION interface also allows one to compose fixed transformations with exponentiated transformations.

Once one has a TRANSFUNCTION in hand, it can be used together with a Serial Transformer object to repeatedly place a physical volume. The Serial Transformer can then be added to the geometry graph. During any subsequent volume traversal, the geometry graph will appear to contain multiple physical volumes at different locations. However, only the memory of a single physical volume and a TRANSFUNCTION is actually allocated.

INTERFACE TO GEANT4, GEO2G4 TRANSLATOR

To run GEANT4 simulation of a detector described in GeoModel it is necessary to build a GEANT4 specific raw geometry based on a GeoModel description. In ATLAS collaboration for this purpose we have developed a tool called Geo2G4, which automatically translates the GeoModel description to GEANT4 by navigating the GeoModel raw geometry graph. To make the resulting geometry memory-efficient we have implemented a set of memory optimization techniques, which are applied during the translation:

1. If some logical volume is shared by *leaf* physical volumes in GeoModel tree, the corresponding GEANT4 logical volume will also be shared.
2. If some physical volume is shared by several parents in GeoModel tree the corresponding branch in GEANT4 tree should also be shared.
3. There is a possibility to translate GeoModel serial transformers into GEANT4 parameterizations. This feature can be switched on/off. For this purpose we have developed a class G4STParameterisation, which can also be used in GEANT4 applications directly, for the purpose of creating G4 parameterizations *without subclassing.*

```
// Constructor
G4STParameterisation(const GeoXF::Function*, unsigned int);

// Compute Transformation method
void G4STParameterisation::ComputeTransformation(const G4int copyNo,
                           G4VPhysicalVolume* physVol)  const
{
  HepTransform3D transform = (*function)(copyNo);
  G4ThreeVector translation = transform.getTranslation();
  G4RotationMatrix* rotation = new
  G4RotationMatrix(transform.getRotation().inverse());
  physVol->SetTranslation(translation);
  physVol->SetRotation(rotation);
}
```

Figure 2: Implementation of GEANT4 parameterisations with G4STParameterisation class

G4STParameterisation objects perform volume parameterizations as in GeoModel, in particular with the use of TRANSFUNCTIONS. The code example presented on Figure 2 demonstrates a mechanism implemented by G4STParameterisation class. We believe that this mechanism could be of major benefit to G4 users whether or not they also use GeoModel.

The Geo2G4 converter is simple and reliable tool, which has been used in ATLAS experiment to translate GeoModel descriptions of all subsystems to GEANT4 simulation.

USAGE OF GEOMODEL TOOLKIT IN ATLAS SOFTWARE

The ATLAS GeoModel project started nearly two years ago. Since that the geometries of all ATLAS detector subsystems have been described using GeoModel mechanisms. Reference[1] describes the way in which all of ATLAS is described in terms of GeoModel. The GeoModel descriptions of tracking detectors are presently used for simulation part of ATLAS Data Challenge 2. Reference [2] describes the simulation of ATLAS and the operational experience in DC2. It is also planned to develop all subsequent versions of ATLAS detector (Initial Layout, Realistic Geometry etc.) with the use of GeoModel and recently developed Geometry Versioning System.

REFERENCES

1. Adele Rimoldi *et. al*, these proceedings.
2. S. Spagnolo *et. al*, these proceedings.