OO model of STAR detector for simulation, visualization and reconstruction

V. Fine[1,2], *P. Nevski*[1]

[1] Brookhaven National Laboratory, USA
[2] Joint Institute for Nuclear Research, Dubna,Russia

Abstract

The Solenoidal Tracker At RHIC (STAR) is a large acceptance collider detector, at Brookhaven National Laboratory. Most of the detailed knowledge on the STAR detector is implemented into a GEANT3 based simulation model. This knowledge certainly is valuable for the new STAR OO software.

STAR ROOT-based framework was upgraded to provide tools to access this model via a set of **TVolume** classes. In this paper we present our experience with migration of the GEANT3 based detector simulation for STAR to an OO model.

Keywords: OO, Fortran, C++, ROOT, GEANT3

1 Introduction

The Solenoidal Tracker At RHIC (STAR), commissioned at Brookhaven National Laboratory in 1999, contains a set of Time Projection Chambers (TPCs) for charged particle tracking over almost six units around central rapidity, a silicon detector for vertexing, electro-magnetic calorimeters, and a number of other systems.

STAR is designed to measure the momentum and identify several thousand particles per event from the collision of heavy nuclei at ultra-relativistic energy. Over 300 Terabytes of real and simulated data will be generated each year. The unprecedented complexity of the events and the total data volume present a formidable software challenge.

STAR has developed a software framework supporting simulation, reconstruction and analysis in offline production, interactive physics analysis and online monitoring environments that is well matched both to STAR present transitional status between Fortran and C++ based software and to a future evolution into a fully OO-based software [1].

To benefit from a transition from Fortran to C++ STAR needs an OO geometry model. On the other hand, most of the detailed knowledge on the STAR detector is implemented into a GEANT3 based simulation model. At the same time, we also want to be able to continue to use the well understood simulation tools based on GEANT3.

This paper presents our experience with using a GEANT3 based detector model as the basis for an OO model of STAR geometry.

2 Basics of STAR geometry model

STAR Detector is described in AGI - Advance Geometry Interface language [2]. This language is an OO FORTRAN extension developed for the GEANT application. In order to be compiled the AGI source code is translated into FORTRAN by a preprocessor.

AGI includes several “GEANT operators” supported by a dedicated Advance GEANT Interface library. Maintaining the GEANT specific tables of materials, volumes, hit descriptors etc, and automatically ensuring the internal consistency of most of the actual parameters of the GEANT routines, it significantly reduce the amount of informations that the user should take care of and provide the necessary robustness of the program.

To ensure the database access and tight control of data integrity, AGI includes data access operators, which together with C-like structure definition allows to control the data transfer between program modules.

Many important implementation details - such as memory management, built-in data structure documentation and database access, coding rule reinforcement etc. - are done by AGI in a way fully transparent for user.

3 ROOT access to geometry objects

Our first approach to the problem was to convert the complete GEANT volume hierarchy into a set of ROOT [3] **TObject** classes. However ROOT model with only **TNode / TShape** classes is not flexible enough, because it does not allow to position the same object many times. A detector model build with only ROOT **TNode/TShape** takes significantly more memory than the original GEANT model. This size may become even prohibitive in case one wants to save the geometry as a persistent object.

To extend the ROOT classes functionality we have introduced a **TVolume** class, which corresponds to a logical volume not assigned to any position in space, as it is used in GEANT.

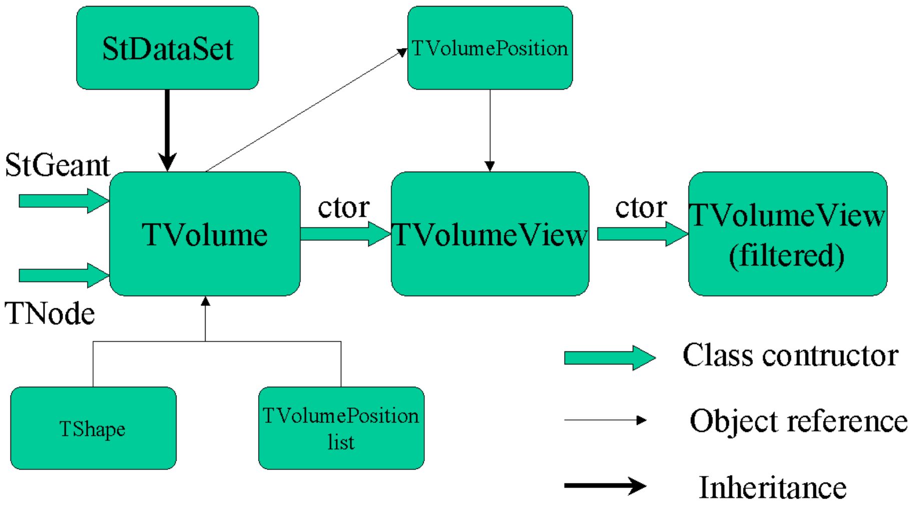

Figure 1: TVolume Class Diagram

TVolume contains a single**TShape**, but contrary to the **TNode** it does not know its own position in space. At the same time **TVolume** inherits from **StDataset** class, developed in the STAR framework as a named collection of objects [2], thus providing a functionality similar to that of a GEANT3 volume.

Using the new class in conjunction with a universal GEANT3 volume decoder, developed in STAR, we are able to project any GEANT3 geometry into a set of **TVolume** classes. The conversion is done inside STAR ROOT-based framework "in flight". This allows to run simulations using GEANT3 while having a C++ presentation of the detector geometry.

The resulting geometry model is almost as compact as the original GEANT model and makes it possible to create and export a persistent detector geometry description.

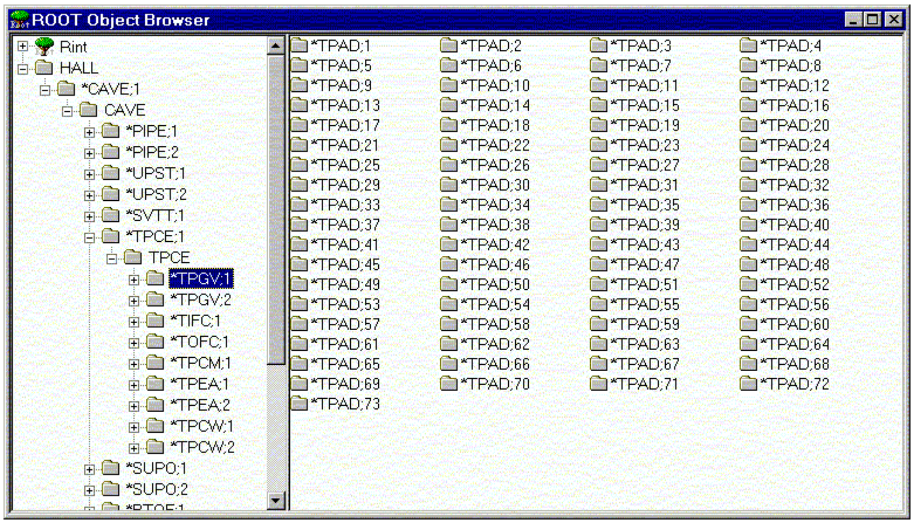

Figure 2: STAR detector model in ROOT browser// A name prefixed with an asterix stands for a **TVolume-Position** object, while other names stand for **TVolume** objects.

However, its disadvantage for graphics is a need for many position recalculations for an interactive display. To support fast graphics we convert this *compact* geometry into a *full* geometry, where each **TVolumeView** represents an individual object, knowing its own position. A **TVolumeView** volume is similar to a **TNode** object, but does not have its own **TShape** object. Instead, been derived from the **StDataSet** class, it can contain a collection of other **TVolumeView** objects, positioned inside.

Since each object in this geometry model has its pre-calculated position, the presentation is very efficient for graphics and graphical navigation.

In case of a complicated geometry and a limited memory this conversion is done only for a part of the detector a user is interesting in.

Since **TVolumeView** does not introduce new shapes and features, **TVolumeView** is a utility visitor class with a pointer to its parent **TVolume** object.

To navigate in both geometries one can use a generic **StDataSetIter** as well as a special **TVolumeView** iterator. The last is derived from the generic one and inherits all features of the **StDataSetIter**. In addition it provides some extra functionality, like a coordinate transformation between any pair of nodes. It can be used for geometrical calculation, coordinate transformation and geometry navigation. In particular, is is possible to delete a number of uninteresting intermediate **TVolumeView** classes from the original hierarchy without loosing the position information for the remaining volumes. This allows to create a compact, but still a complete **filtered TVolumeView** hierarchy which is easier to draw and faster to navigate.

Because the **StDataSet** is supported by a generic STAR general data model [4], geometry information can be recalled form any maker, save and restored by I/O “makers” in ROOT files, MySQL database and a plain ASCII format.

Because of the inheritance from the ROOT classes, different graphic viewer can be used to draw the resulting geometry. OpenGL, ROOT TPad and X3D are particular examples of such viewers. This also allows to export geometry in postscript and GIF formats.

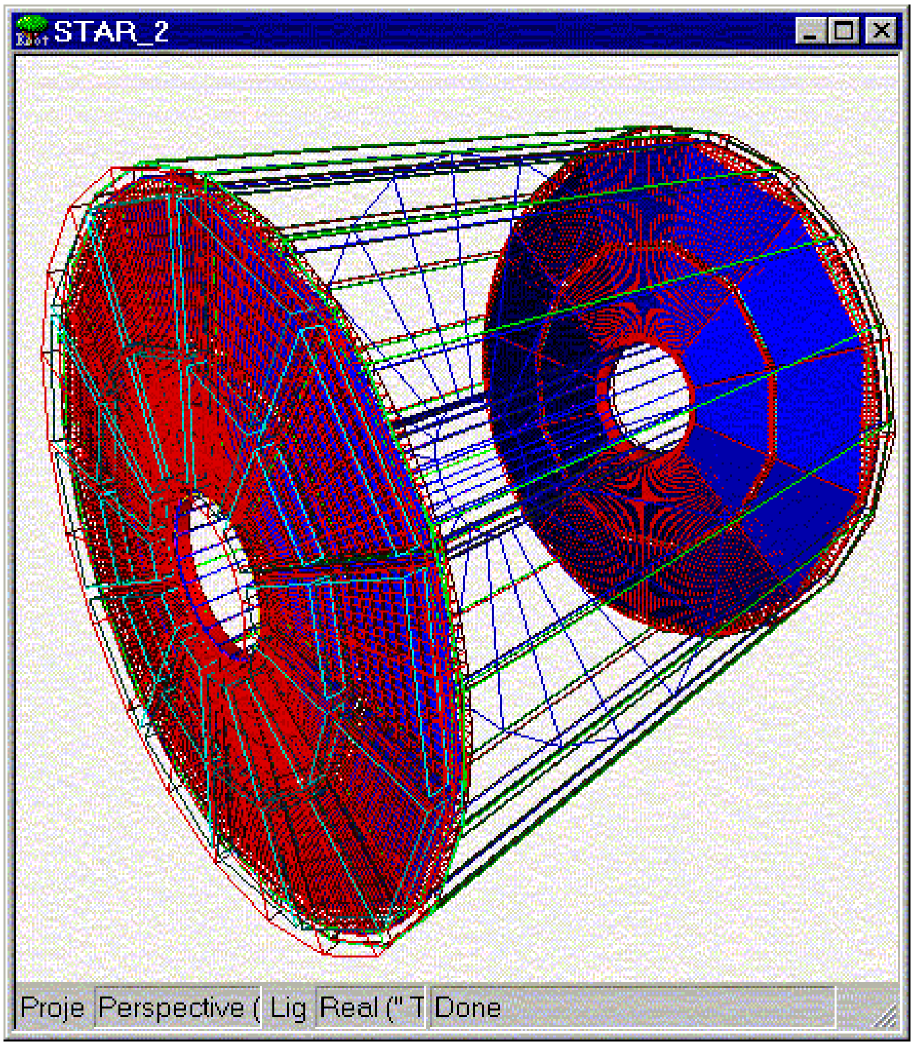

Figure 3: a part of the STAR detector geometry in ROOT

References

1 V.Fine at al, “Steps Towards C++/OO Offline Software in STAR”, CHEP’98, Chicago, Autumn 1998.

2 A. Artamonov et al, “DICE-95”, internal note ATLAS-SOFT/95-14, CERN, 1995.

3 R. Brun et al, “The ROOT Framework”, AIHEPN-96, Lausanne, August 1996.

4 V. Fine et al, “STAR offline framework”, CHEP’2000