The new GeoModel suite, a lightweight detector description and visualization toolkit for HEP

M Bandieramonte[1†], R M Bianchi[1†*], J Boudreau[1†], A Dell'Acqua[2†], V Tsulaia[3†]

[1] University of Pittsburgh, Pittsburgh, PA 15260, USA
[2] CERN, EP Department, Meyrin, 1211, Switzerland
[3] Lawrence Berkeley National Laboratory, Berkeley, CA, 94720, USA

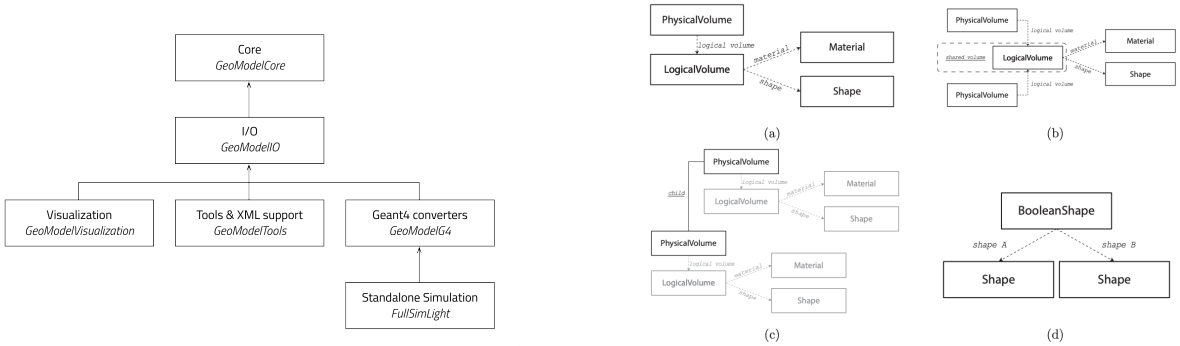
E-mail: riccardo.maria.bianchi@cern.ch

Abstract. The GeoModel toolkit is an open-source suite of standalone tools that provides the user with lightweight tools to describe, visualize, test, and debug detector descriptions and geometries for HEP standalone studies and experiments. GeoModel has been designed with independence and responsiveness in mind and offers a development environment free of other large HEP tools and frameworks, and with a very quick development cycle. With very few and lightweight dependencies, GeoModel is easy to install on all systems, in a modular way; and pre-compiled binaries are provided for the major platforms, for a quick and easy installation. Coded entirely in C++, GeoModel offers the user tools to describe geometries inside C++ code or in external XML files, create persistent representation with a low disk footprint and interactively visualize and inspect the geometry in a 3D view. It also offers a plugin mechanism and an optional Geant4 application to simulate the described geometry in a standalone environment. GeoModel has been developed as part of the software for the ATLAS experiment at the LHC, and evolved towards an experiment-independent toolkit. In this contribution, we describe all the available tools, with a focus on the latest additions, which provide users with more visualization, debug, and simulation tools.

1. Introduction

Detector description is a key component of the High Energy Physics (HEP) workflow, at all steps. At reconstruction level, when reconstructing the event from the raw measurements, geometry information is used to know the position of the sensitive material. When simulating the detector response, the geometry shapes and their materials' composition are used in the calculations for particle transport. When processing the raw measurements from the different detectors, the position of the geometry volumes containing the sensitive material is used in the "Alignment step", to account for misalignment during data taking. When analyzing the final event data, events are sometimes subjected to visual inspection. to check that the right objects have been selected and correctly reconstructed in the different layers of detectors. Ultimately, when publishing the results or engaging with the general public (e.g., in *"Outreach & Education"* activities), nicely rendered detector geometry is the foundation of appealing and

* Corresponding author
† On behalf of the ATLAS Collaboration



(a) The GeoModel modular architecture

(b) Example of GeoModel nodes

Figure 1: On the left, is the modular architecture of the GeoModel tool suite: developers can pick and use the modules that provide the functionality they need. On the right, diagrams showing examples of different GeoModel nodes: a) the implementation of a physical volume; b) a logical volume shared between two physical volumes; c) the parent-child relationship between two physical volumes; d) a boolean shape. More details can be found in [2, 8].

interesting images for publications, presentations, or public events. Therefore, an accurate detector description is key in all HEP experiments.

The following sections briefly present the GeoModel [1] geometry tool kit and tool suite, which lets detector description developers design and implement a detector geometry in a straightforward way, with minimal external dependencies and without the burden of heavy HEP software frameworks. After introducing the available tools, we focus on the most recent additions to the tool suite.

2. The GeoModel Tool Suite

The GeoModel toolkit [2, 3] lets developers implement and manage detector descriptions with a complete set of tools. GeoModel offers a very fast development cycle, with immediate feedback to developers' changes. GeoModel has minimal external dependencies and it is free of the burden brought in by large HEP software frameworks; that lets developers include GeoModel in any existing workflow or framework easily. It also offers a modular architecture: developers can pick all and only the modules that provide the functionality they need (see Figure 1a).

The GeoModel core libraries are currently used by the ATLAS [4] and FASER [5] experiments on the Large Hadron Collider [6] at CERN to describe their detector geometry. In addition to this, the complete GeoModel tool suite is being used by ATLAS as the foundation of its upgraded Detector Description architecture for the forthcoming HL-LHC data taking period [7]. The GeoModel toolkit is actively maintained and developed by a dedicated team of developers supported by the ATLAS experiment.

2.1. Core

The core libraries (the *kernel*) offer the base objects to build a detector description, such as shape primitives (boxes, poly-cones, trapezoids, and so forth), chemical elements, materials, logical volumes, physical volumes, transformations, alignable transforms, functions, parametric positioners, naming tags, and others. These items compose the so-called GeoModel *tree*, which completely describes the detector geometry. The architecture is designed so as to make this description as compact as possible. It allows building a directed, acyclic graph (DAG) of *nodes* which is interpreted as a DAG of volumes (see Figure 1b) [8, 9].

2.2. I/O & Persistency

The I/O module allows to store a persistent copy of the detector description (the *GeoModel tree*) to an SQLite file and restore from it. The data model has been optimized for a minimal footprint on disk and quick I/O operations. As an example, the full geometry of the ATLAS experiment (composed of about 530'000 nodes) is stored in an SQLite file of around 48 Mb, and it is loaded and restored in memory in about 7 seconds on a standard laptop computer.

SQLite has been chosen as the I/O layer because of its ubiquity and lightness: being an industry standard, and depending on very few system packages, it is easy for the final users to integrate it in any existing workflow with minimal changes.

GeoModel also offers an exporter to the GDML [10] format: it is part of the standalone simulation tools (see Section 2.5) and converts each GeoModel node to a correspondent GDML entity; this helps HEP developers to integrate GeoModel in existing workflows. Moreover, a tool to convert GDML-based geometries to GeoModel is under development within the GeoModelTools module, with new GDML entities being added. That lets detector developers use the GeoModel visualization and inspection tools with geometries that are already implemented in GDML.

2.3. Tools & Parsers

GeoModel provides a set of libraries and command-line standalone tools to implement, import, handle, and check detector description data. In particular, a plugin system has been implemented, to let detector developers describe their detector in a self-contained C++ plugin, which is then loaded at run-time. This allows the developers to use different plugins for different parts of the detector, separating the development cycle and optimizing the build time. If needed by the user's workflow, the plugins can also interact with the GeoModelIO layer, to store auxiliary data (data that are used later in the experiment's framework; for instance, when building the so-called *readout geometry*[1]). In addition, an optional XML parser (GeoModelXML) can be used by the detector developers to store geometry data or to describe repetitive parts of their geometry in XML files and to build the GeoModel tree from there. Recently, new positioning entities have been added to GeoModelXML; that will be discussed more in detail in Section 3.

The latest release of GeoModel also offers a new command-line tool to compute the memory footprint of a given piece of the geometry, at different levels of detail. This gives detector developers a way to measure the efficiency of the implementation geometry description.

2.4. Visualization

The GeoModel tool suite offers an embedded graphical tool for prompt 3D interactive visualization. With minimal dependencies (the open-source Qt C++ GUI framework [11] and Coin3D/SoQt libraries [12, 13]), the *Visualization* module can be built on any standard platform and provides the users with a full set of tools to interactively visualize, inspect, and debug the detector geometry while they are implementing it. The detector description developers can therefore work with an environment allowing very quick feedback: as soon as they modify the detector geometry, they can promptly check it visually, without the need of running cumbersome frameworks or external tools. One of the latest addition to the visualization module is the possibility to query the GeoModel tree with regular expression; this will be the focus of Section 4.

[1] In this paper, we mainly refer to the so-called *raw* geometry. In HEP experiments, this describes the detector in terms of physical volumes, each of them featuring a particular shape and composed by a given material, and space transformations. On top of that, the *readout* geometry is built, which takes into account geometrical features not modelled as separate volumes, such as silicon strips or calorimeter cells whose boundaries are determined by electronic coupling

```
<replicaXYArrays name="Quadrant_1" n="106" zValue="2.995"
                 xCoordinates="x1" yCoordinates="y1">
    <logVolRef ref="Module" />
</replicaXYArrays>

<replicaXYArrays name="Quadrant_2" n="144" zValue="2.995"
                 xCoordinates="x2" yCoordinates="y2">
    <transform name="transformation2">
        <transformation name="rot2">
            <rotation zcos="1." angle="PI/2" />
        </transformation>
        <logVolRef ref="Module" />
    </transform>
</replicaXYArrays>
```

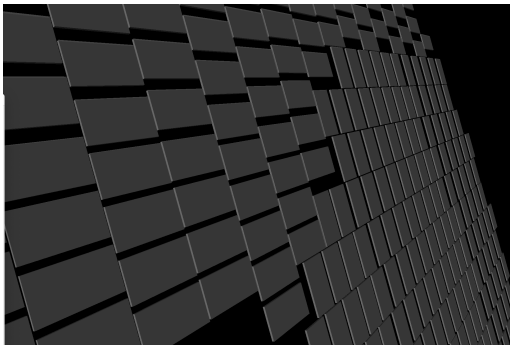


Figure 2: The new `replicaXYArrays` positioner: a helper XML entity, which lets developers replicate a given geometry volume on the X-Y plane, by specifying the number of copies and two arrays storing their coordinates. By using a combination of positioners and transformations (such as rotations, as in the example on the left) it is possible for the developers to efficiently describe a very complex piece of detector geometry, as shown in the image on the right.

2.5. Standalone Simulation Tools

As part of the GeoModel tool suite, a Geant4-based [14] application helps the detector developers to promptly test the implemented geometry for particle transport in a lightweight and standalone environment. The suite also offers Geant4-based command-line tools to detect clashes between geometry volumes, compute material and radiation length maps, and calculate the mass of a given piece of the geometry. The latter will be the focus of Section 5.

3. New parametric volume positioners in GeoModelXML

Describing the detector geometry within XML files lets developers easily organize the data in a hierarchical way and allows for a quick development cycle: applying changes to a piece of geometry is only a matter of seconds because no compilation is needed, the new geometry is loaded and visualized instantly. However, XML is a markup language and lacks all commands and structures for conditional execution and loops, making it difficult for geometry developers to describe pieces of geometry featuring, for example, repetitive patterns.

GeoModel features parametric positioners in the C++ interface (for instance, the `GeoSerialTransformers` node), to help developers place a number of copies of a given volume according to a parametric mathematical function[2].

However, such helper entities were not present in the XML interface so far. Therefore, to ease the task of describing, in the XML description, a same geometry volume repetitively placed at different positions, an additional series of new parametric positioners have been introduced in the XML interface, GeoModelXML [15]. The new `replicaXYArrays` XML entity lets developers replicate a given piece of geometry by specifying the number of copies and two arrays containing their coordinates on the X and Y axes (see Figure 2). The new XML `replicaRPhi` positioner allows for replication of a piece of geometry over a radial surface, by specifying an angle and a radius. The `replicaX`, `replicaY`, and `replicaZ` XML entities let developers easily replicate a given volume along an axis. All the new XML positioners allow for offset and step values, letting the developers describe complex detector geometries in a few lines. See Figure 3 for an example of the new XML positioning entities.

[2] The parametric placement also helps in optimizing the memory footprint of the detector description. The placed volume, in fact, is created only once in memory; then, the same instance is used for all occurrences in the 3D space.

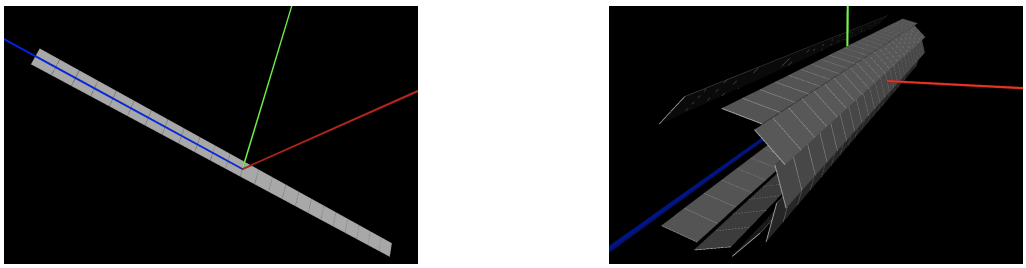


Figure 3: The new `replicaZ` and `replicaRPhi` positioners are helper XML entities that let developers replicate a given piece of geometry along the Z-axis (on the left) and on a radial surface (on the right). In the images, the Z, X, and Y axes are represented by the blue, red, and green axis, respectively.

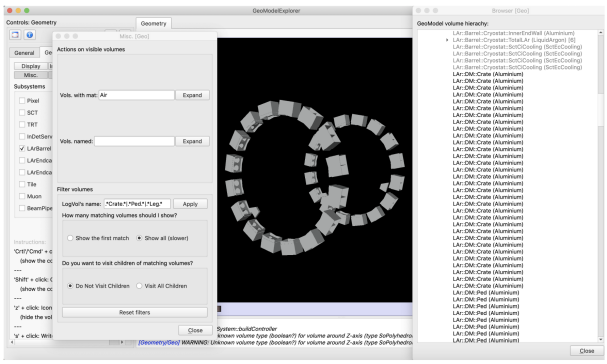


Figure 4: The new filtering engine added to `GeoModelExplorer` lets users interactively filter on volumes based on the name of the associated logical volume. Regular expressions can be used and complex searches can be built by combining multiple patterns with boolean operators.

4. Querying the Geometry Tree with Regular Expressions

Detector geometries can be extremely complex, with hundreds of thousands of volumes. Therefore, it is often hard to visualize, highlight, or inspect the desired volumes. Because of that, a new filtering mechanism has been added to the interactive 3D visualization tool, the `GeoModelExplorer` [16] (also known as `gmex`).

The new tool lets detector developers search for volumes by the use of regular expressions. The algorithm searches over the "name" of the logical volume (`GeoLogVol`) associated with a physical volume (`GeoPhysVol`) [8]. Developers can search for a specific volume, or for a set of volumes whose name matches the same search pattern. The search patterns can be combined through the use of boolean operators and regular expressions, to build complex and more refined searches, as shown in Figure 4. The number of geometry layers considered in the search can be customized by the users: the *depth* value can be used to exclude layers of geometry or can be used for a more accurate search among the nested volumes.

5. Computing the mass of a piece of geometry

A new tool has been recently introduced, the `gmmasscalculator`. The new tool is part of the `FullSimLight` [17] sub-package gathering all tools that depend on the Geant4 toolkit [14]. The new tool calculates the *inclusive* and *exclusive mass* of a given piece of the geometry, and its *apparent weight* in air. The *exclusive mass* is the mass of the considered volume, from which the volumes occupied by the daughters' volumes have been subtracted (see Equation 1). The *inclusive mass* is the mass of the considered volume, comprehensive of the masses of the

respective daughters, propagated in an iterative way to their daughter volumes (see Equation 2). The *apparent weight in Air* is the weight of a body as affected by the buoyancy of the fluid (such as air) in which it is immersed (see Equation 3).

$$m^{\text{excl}}_{\text{mother}} = (V_{\text{mother}} - \sum_i V_{\text{daughter}_i}) \cdot \rho_{\text{mother}} \tag{1}$$

$$\begin{aligned} m^{\text{incl}}_{\text{mother}} &= m^{\text{excl}}_{\text{mother}} + \sum_i ((V_{\text{daughter}_i} - \sum_j V_{\text{grandaughter}_j}) \cdot \rho_{\text{daughter}}) + ... \\ &= m^{\text{excl}}_{\text{mother}} + m^{\text{excl}}_{\text{daughters}} + m^{\text{excl}}_{\text{granddaughters}} + ... \end{aligned} \tag{2}$$

$$w^{\text{apparent}} = \sum_i (m^{\text{incl}}_{\text{daughter}_i} - \sum_j (V_{\text{daughter}_j} \cdot \rho_{\text{Air}}) \tag{3}$$

By default, the new mass calculator tool takes the main *"World"* (*top*) volume as the *mother volume*, and calculates the inclusive and exclusive masses of the respective daughters, saving the calculated quantities in the output JSON file. The total masses and the apparent weight in Air are reported for the whole *"World"* volume. Two optional parameters let the users steer the tool, one to specify a set of volume names to be considered, the other to only run on volumes made of a given material. In the end, a mass report is given in an output JSON file.

The new `gmmasscalculator` tool supports different geometry formats: the GeoModel native SQLite data format; a GeoModel plugin describing the geometry; or a GDML geometry file.

6. Conclusions

New tools have been added to the GeoModel toolkit, letting detector developers describe their geometry more easily and perform new and more complete checks on it, as well as to measure the memory footprint, compute the masses of the volumes, and debug the overall geometry implementation in a more efficient way by using interactive filters within the visualization tool. The whole GeoModel tool suite is being used as the foundation of the new, optimized, and more efficient ATLAS detector description architecture for the HL-LHC data taking period. More tools will be added in the near future, towards an even more lightweight, portable, and efficient detector description toolkit for HEP experiments.

References

[1] GeoModel, A C++ Toolkit for Detector Description URL `https://gitlab.cern.ch/GeoModelDev/GeoModel`
[2] Bandieramonte M, Bianchi R M, Boudreau J, Dell'Acqua A and Tsulaia V 2021 *EPJ Web Conf.* **251** 03007
[3] GeoModel Tool Suite Documentation Website URL `http://cern.ch/geomodel`
[4] ATLAS Collaboration 2008 *Journal of Instrumentation* **3** s08003
[5] FASER Collaboration 2018 Technical Proposal URL `http://cds.cern.ch/record/2651328`
[6] Evans L and (editors) P B 2008 *JINST* **3** s08003
[7] HL-LHC Collaboration 2020 (Geneva: CERN) URL `https://cds.cern.ch/record/2749422`
[8] Bianchi R M, Boudreau J and Vukotic I 2017 *Journal of Physics: Conference Series* **898** 072015
[9] Bianchi R M and Vukotic I 2018 *Journal of Physics: Conference Series* **1085** 032035
[10] Chytracek R, Mccormick J, Pokorski W and Santin G 2006 *IEEE Transactions on Nuclear Science* **53**
[11] Qt, the cross-platform software development framework URL `https://doc.qt.io/`
[12] Coin, an OpenGL-based, 3D graphics library URL `https://github.com/coin3d/coin`
[13] SoQt, a Qt GUI toolkit library for Coin URL `https://github.com/coin3d/soqt`
[14] Agostinelli S *et al.* (GEANT4) 2003 *Nuclear Instruments and Methods in Physics Research Section A* **A506**
[15] GeoModelXML, an XML interface for GeoModel URL `https://gitlab.cern.ch/GeoModelDev/GeoModel/-/tree/master/GeoModelTools/GeoModelXML`
[16] GeoModelVisualization, a 3D visualization module for GeoModel URL `https://gitlab.cern.ch/GeoModelDev/GeoModel/-/tree/master/GeoModelVisualization`
[17] FullSimLight, a light, standalone, Geant4-based simulation toolkit URL `https://gitlab.cern.ch/GeoModelDev/GeoModel/-/tree/master/FullSimLight`