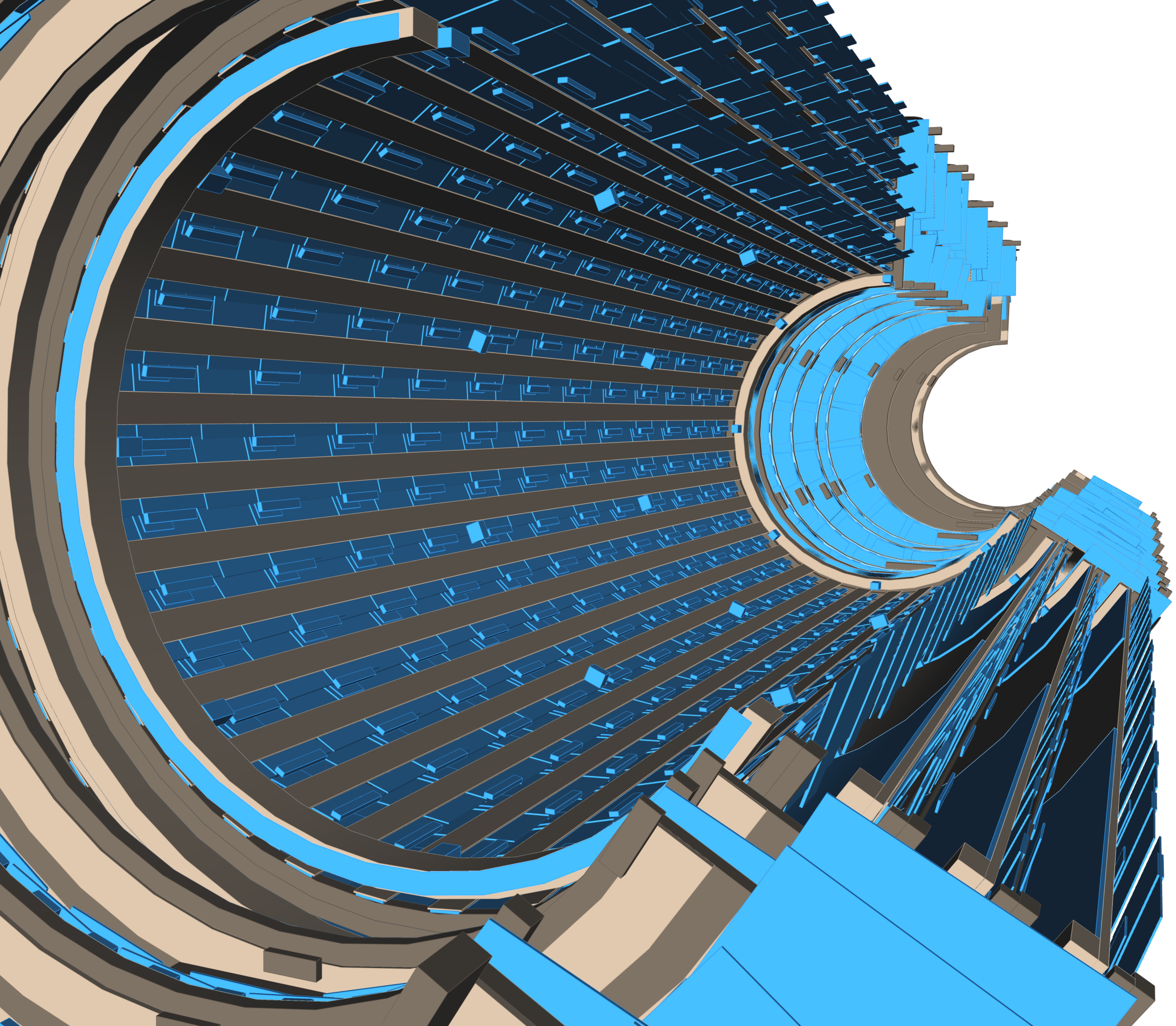


The new GeoModel suite

A lightweight detector description and visualization toolkit for HEP

*Marilena **BANDIERAMONTE** (Pittsburgh)*
*Riccardo Maria **BIANCHI** (Pittsburgh)*
*Joseph **BOUDREAU** (Pittsburgh)*
*Andrea **DELL'ACQUA** (CERN)*
*Vakhtang **TSULAIA** (LBNL)*

ATLAS EXPERIMENT

ACAT 2021
Track 1: Computing Technology for Physics Research
1 Dec 2021

Detector Description in HEP

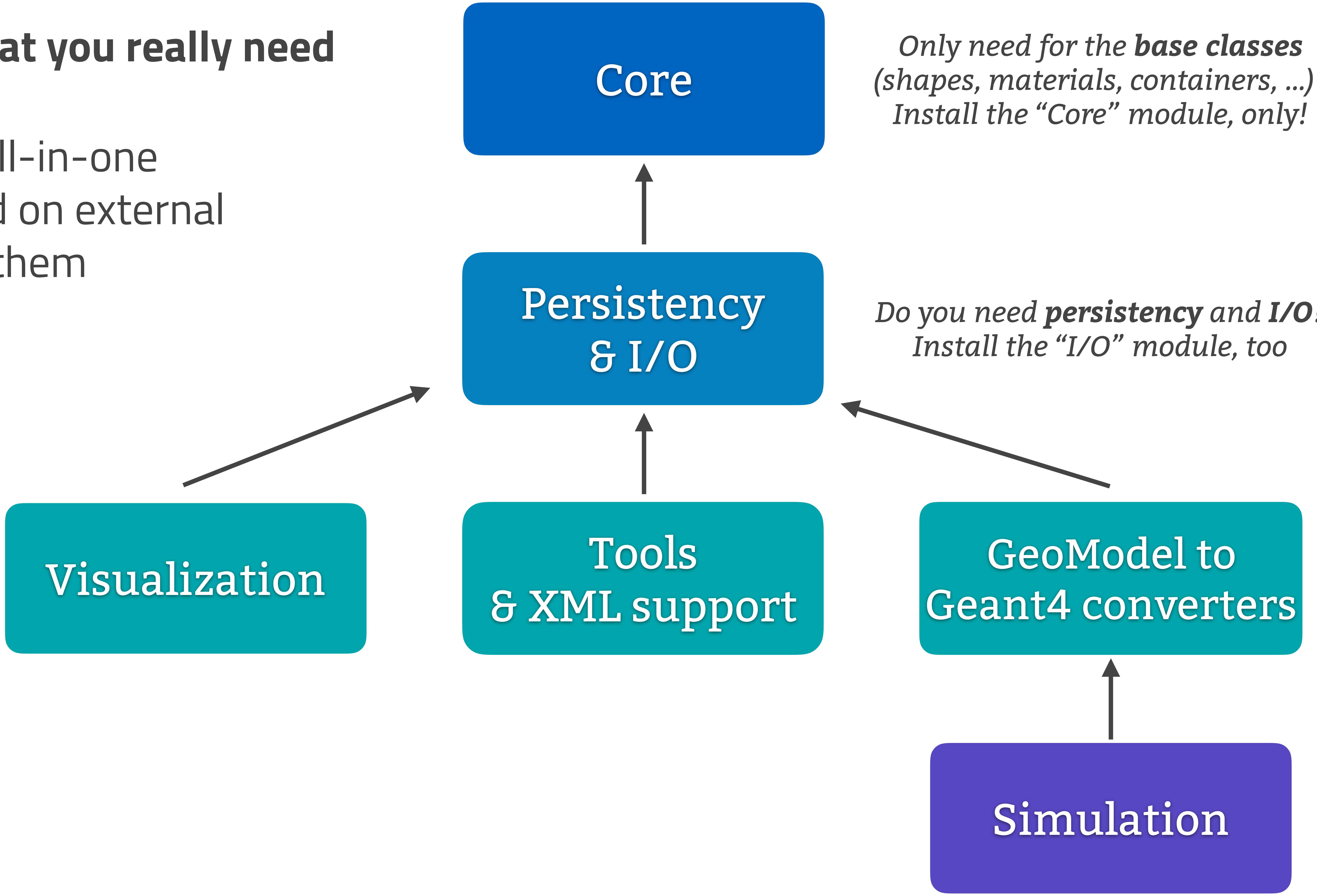
- Detector Description is a **key component** of the High Energy Physics (HEP) workflow:
  - Geometry information is used when reconstructing raw measurements — ***Reconstruction***
  - When simulating the **detector response**, geometry shapes and materials are used in the calculations for the particle transport — ***Simulation***
  - The position of the geometry volumes needs to be **corrected on-the-fly** to assess for misalignments during data taking — ***Alignment***
  - When **analyzing data**, events are inspected visually to check the right objects have been selected, and correctly reconstructed — ***Performance & Analysis***
  - A nicely rendered detector geometry is the foundation of **engaging images** for events and publications — ***Outreach & Education***

Why GeoModel?

- The new GeoModel toolkit is **our answer** to detector description developers looking for:
  - a **lightweight toolkit**, with minimal external dependencies and free from large and cumbersome all-in-one frameworks
  - a **modular architecture**, to let developers use only the modules and tools they need
  - a **choice of input formats**, to better integrate the detector description to their workflow
  - **quick development cycle**, with quick feedback upon changes to the detector description
  - a suite of **standalone tools**, for interactive visualization and to perform checks
  - a **standalone simulation** application, for particle transport, clash detection, mass calculation, and material & radiation length maps

Modular architecture

- You only **build/install what you really need**
- No need to install large, all-in-one frameworks, or to depend on external libraries if you won't use them



*Only need for the **base classes** (shapes, materials, containers, …)? Install the "Core" module, only!*

*Do you need **persistency** and **I/O**? Install the "I/O" module, too*

*Do you need **visualization** but not XML support and other tools? Then, install the "Visualization" module, only*

*Do you need **simulation** but not visualization? Just install the "Simulation" module and skip the "Visualization" one!*

Minimal dependencies

- GeoModel was designed to have a **minimal** set of external **dependencies**
- That **ease** the developers' day-by-day **work**: GeoModel can be installed and used on any Unix-like platform (macOS, Linux, …)
- That also **helps the integration** of GeoModel in existing workflows, being **free** of large HEP frameworks

Core	I/O	Tools / XML	Visualization	Simulation
Eigen3 (maths)	*SQLite (persistency)*	*XercesC (XML)* *nlohman-json (JSON)*	*Coin & SoQt (3D)* *Qt5 (GUI)*	*Geant4*

Multiple possible ways to use GeoModel

- Developers can **choose** to implement their detector description in any of these **three ways**:
  - native GeoModel code in **bare C++ classes / applications**
  - use the **GeoModel C++ plugin mechanism**, to embed/pack native GeoModel code in precompiled libraries (.so/.dylib) and read/store/load it at **runtime**
  - **XML files**, then parsed at runtime to automatically create the GeoModel tree
- Developers can freely use **any combination** of the options above, to stick to their needs and **optimize their workflow**
  - *E.g., one part of the geometry can be described through native code, while another part through XML files*

Fast Development Cycle & Easy Installation

- We designed the new GeoModel suite to let developers experience a **fast development cycle**:
  - Developers can **modify** the geometry and then visualize it in a matter of seconds
  - Standalone command-line tools provide **prompt checks** of the geometry
  - **Visualization** is optimized for geometry, offering a set of tools to easily inspect it
  - Standalone **simulation tools** give quick feedback on the geometry correctness
  - The **persistent representation** in SQLite files is **easy to integrate** in any workflow, and **quick to inspect and query** with standard tools

- Developers can **install** the latest stable version from their preferred **package managers***:
  - E.g., for macOS:

```
brew tap atlas/geomodel https://gitlab.cern.ch/GeoModelDev/packaging/homebrew-geomodel.git
brew install geomodel-visualization
```

- GeoModel can be built* on all Unix-like platforms: e.g., macOS, Ubuntu, Fedora, Centos7

** complete instructions at: https://geomodel.web.cern.ch*

Core

Geometry as a tree of nodes

- GeoModelKernel provides the **base geometry classes**

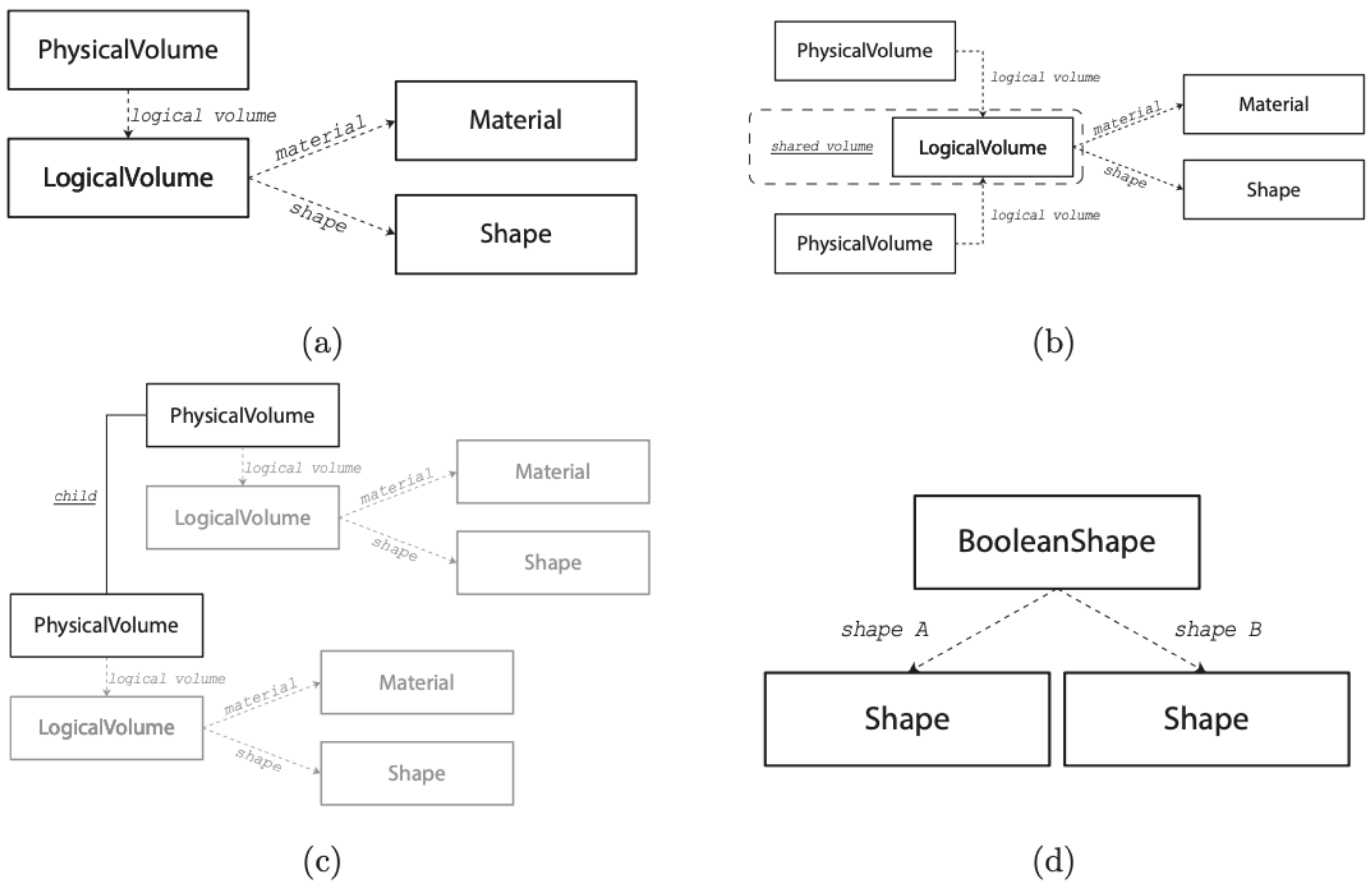
- Geometry entities are **nodes**, each node representing a shape, a material, a volume, a transformation, etc…

  a. Some nodes **reference others**
  (E.g., a logical volume node references a material and a shape node)

  b. Nodes are **shared**, to lower the geometry memory footprint
  E.g., material used by multiple nodes is only created once, then shared

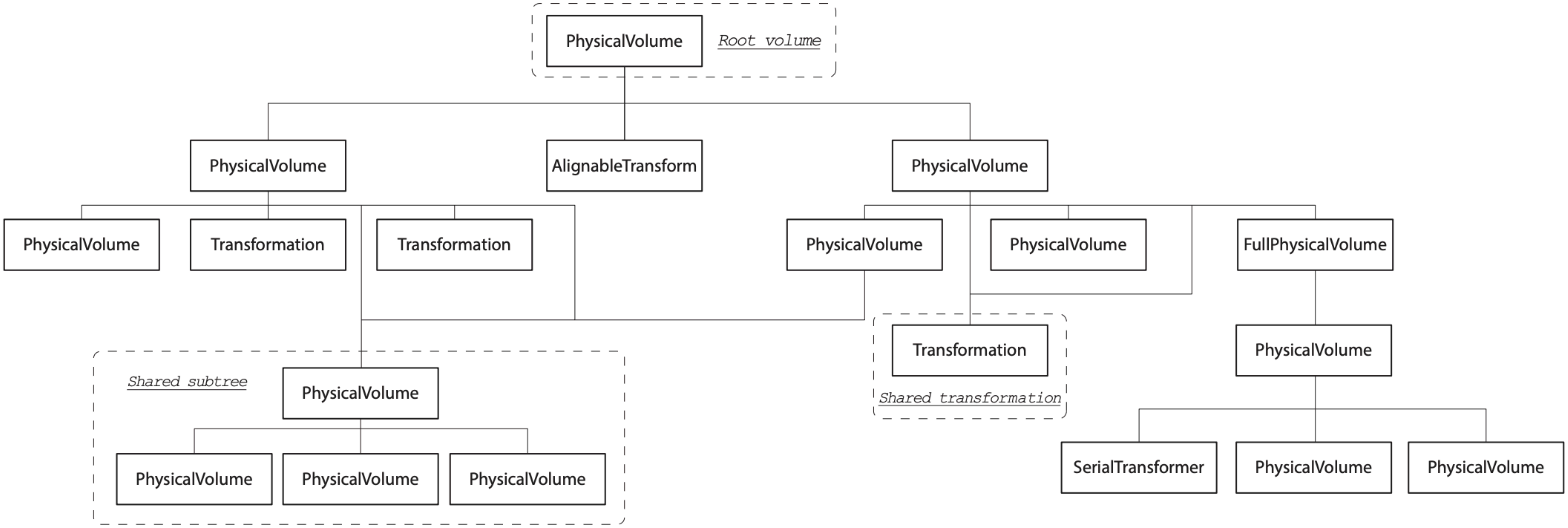
  c. Some nodes acts as **containers** of child nodes
  (E.g., a calorimeter mother volume containing the absorber slices as children)

  d. Some nodes acts as **operators**
  E.g., a boolean shape operates over two shape nodes



Geometry as a tree of nodes

- The whole detector **geometry** is organized as a **tree of nodes**



Persistency & I/O

Persistification & I/O

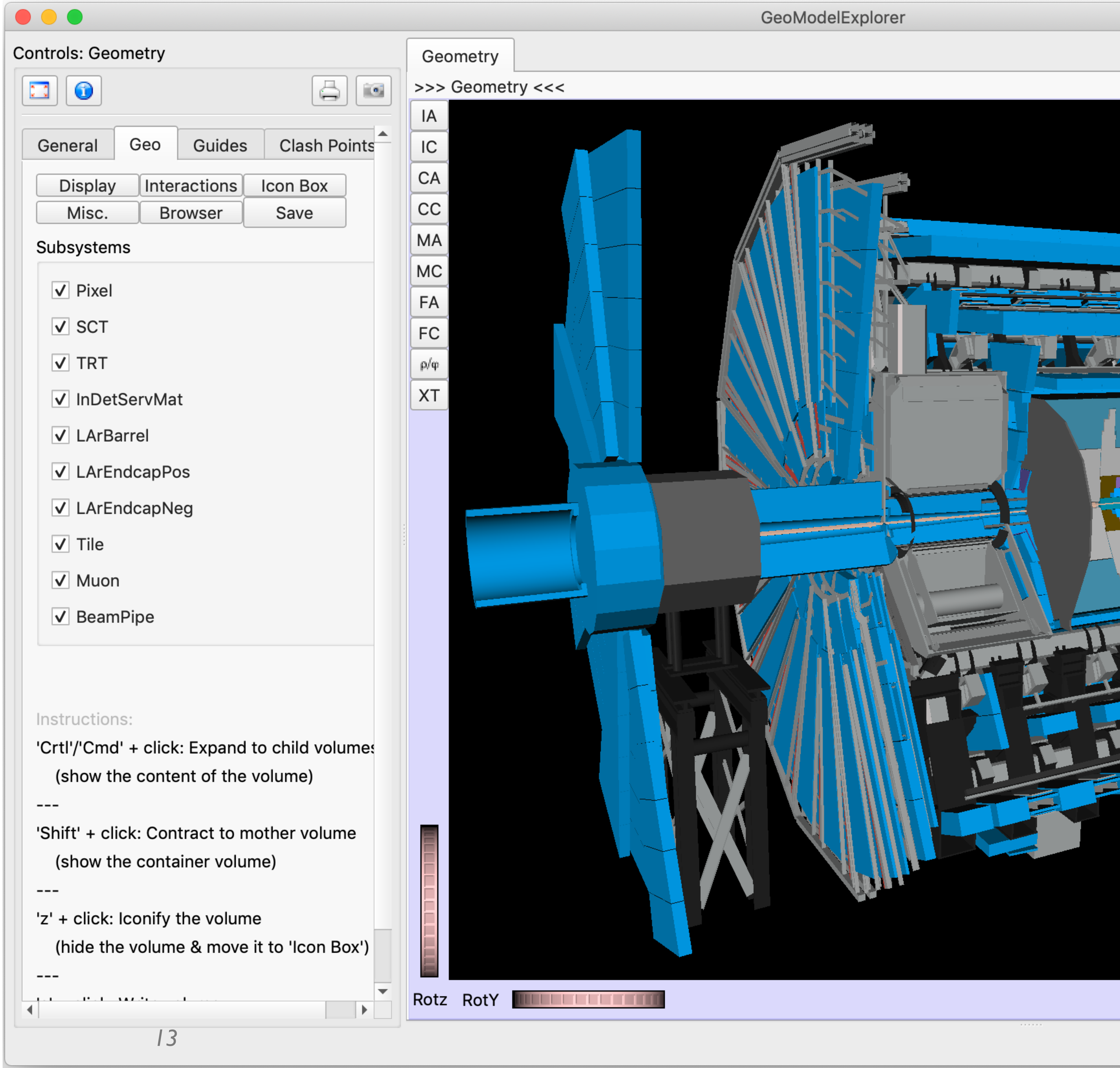
- The **full geometry** can be **persistified** into a **SQLite file**
  - SQLite files can be inspected/modified with standard DB tools
  - SQLite is ubiquitous (no need to install it) and lightweight (no need for large frameworks)
- We **optimized the data model** to optimize the file size and disk footprint
  - Only IDs of the nodes are stored in tables
  - Shared nodes are stored once only, and *copy numbers* are stored to ensure reproducibility
- The “Read" operation of GeoModel SQLite files is fully parallelized, for a fast loading time

*As an example, the **full** geometry of the **ATLAS** experiment (composed of ~**530’000 nodes**) is **stored** in a **~48 Mb SQLite file**, and it is **loaded** and restored in memory in **~7 seconds on a laptop***

- Geometry **can be** also **exported to GDML**, through the use of our GM—>GDML **converter***

* see slide on "Simulation tools" for details

Visualization



Visualizing & Inspecting geometry

- The new GeoModel suite offers a **complete 3D visualization package** optimized for the interactive **inspection** and **debugging** of **detector geometry**: GeoModelExplorer
- The visualization tool **can handle a very large number of volumes on screen**, letting users visualize very **complex geometries, responsively**
- Users can pick **volumes** to display information, can hide them, can expand them to show their child volumes
- A **browser** lets users **inspect the hierarchy** of the loaded GeoModel tree
- GeoModelExplorer *(gmex)* can export the 3D view to the SVG graphic format, letting users open it in external applications (like Adobe Illustrator)

See the Extra Material slides at the end, for additional information!

** an upgrade to Qt6 is foreseen*

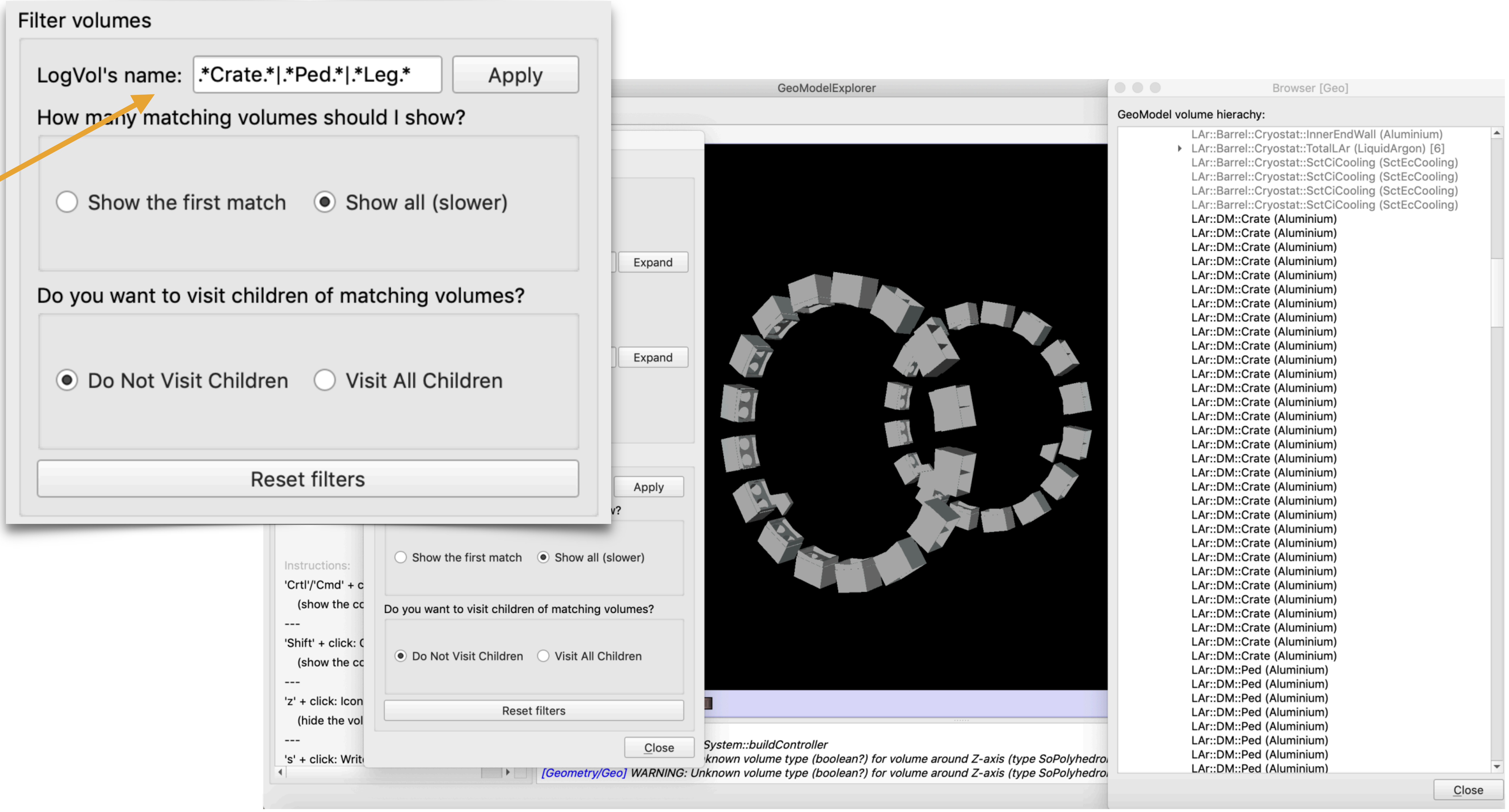
New feature: Search patterns

- The **latest addition** to the visualization tool is the “Volume Filter”. Developers can now **visualize** volumes based on **search patterns**

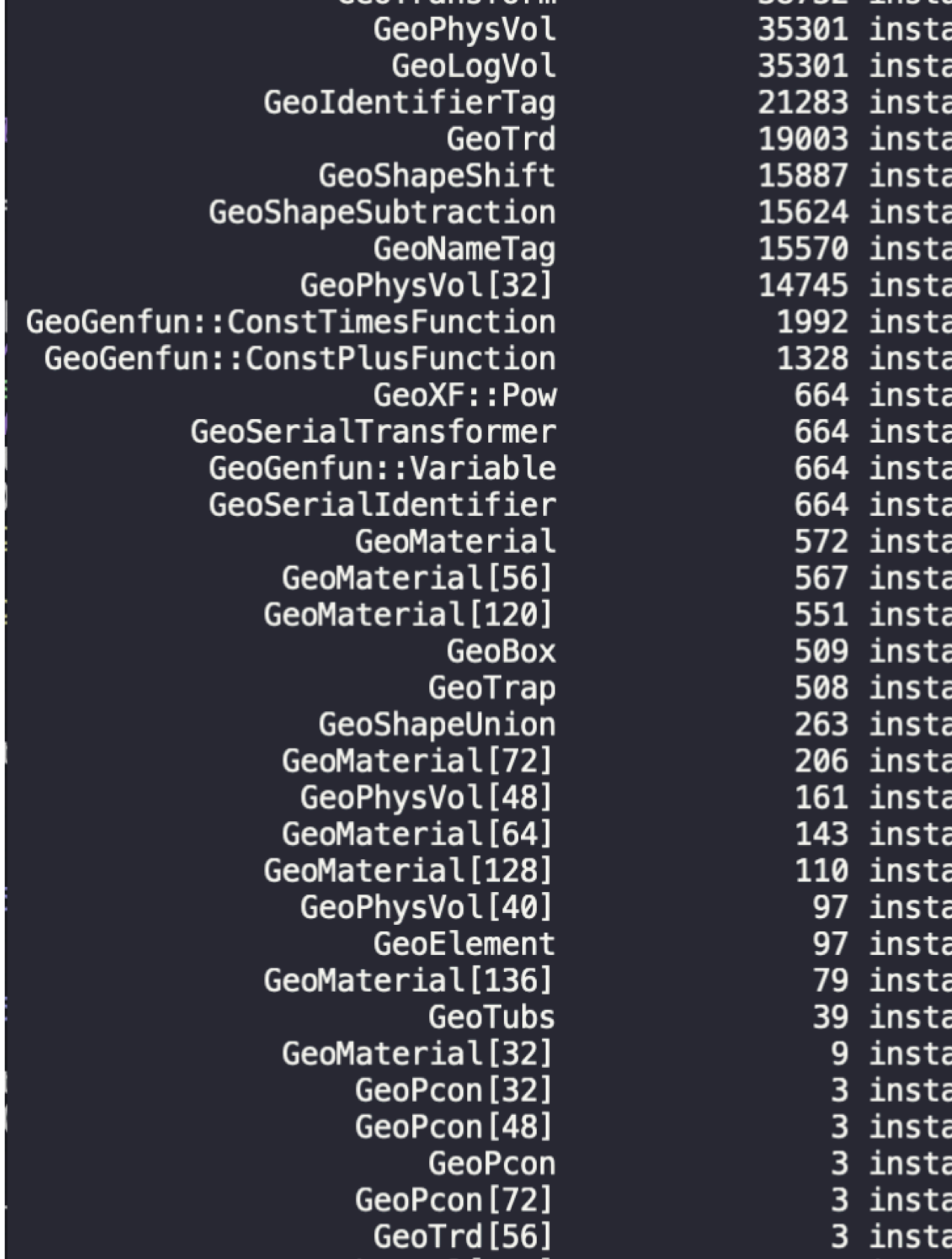
- No need to dig down a complex geometry by hand

- You can use **regular expressions** to create smart search patterns

- Search patterns can be combined with **boolean operators** to only show the relevant volumes



Command-line Tools & XML support



Other command line tools

- The GeoModelTools package offers a set of **command-line tools** to handling detector descriptions and perform test:
  - **NEW! gmstatistics**: it prints out the memory footprint *(see next slide)*
  - **NEW FEATURES! XML Parser**: it lets users describe their detectors in XML files *(see next slides)*
  - **gmcat**: it lets users concatenate SQLite files (.db) and/or plugins (.so/.dylib) *(see backup slides)*
  - **gdml2gm**: it converts input GDML* geometry into GeoModel *(see backup slides)*

See the "Extra Material" slides at the end, for additional information!

gmstatistics - Memory footprint

- The **new tool** GMSTATISTICS *(gmstatistics)* lets user **print out** the **detailed memory footprint** of a **given geometry**

- A technical note: the tool behaves differently on macOS and on linux:
  - **macOS** traces the memory as it is allocated
  - **Linux** traces memory as it is deallocated
  - Also, **macOS** “heap” **gives a lot of details**: which objects are allocated, how many, and how much memory they occupy

GeoTransform	38732 instances	5577408 bytes
GeoPhysVol	35301 instances	4518528 bytes
GeoLogVol	35301 instances	2259264 bytes
GeoIdentifierTag	21283 instances	340528 bytes
GeoTrd	19003 instances	1216192 bytes
GeoShapeShift	15887 instances	2541920 bytes
GeoShapeSubtraction	15624 instances	499968 bytes
GeoNameTag	15570 instances	747360 bytes
GeoPhysVol[32]	14745 instances	1167072 bytes
GeoGenfun::ConstTimesFunction	1992 instances	63744 bytes
GeoGenfun::ConstPlusFunction	1328 instances	42496 bytes
GeoXF::Pow	664 instances	106240 bytes
GeoSerialTransformer	664 instances	21248 bytes
GeoGenfun::Variable	664 instances	10624 bytes
GeoSerialIdentifier	664 instances	10624 bytes
GeoMaterial	572 instances	82368 bytes
GeoMaterial[56]	567 instances	33024 bytes
GeoMaterial[120]	551 instances	32720 bytes
GeoBox	509 instances	24432 bytes
GeoTrap	508 instances	56896 bytes
GeoShapeUnion	263 instances	8416 bytes
GeoMaterial[72]	206 instances	6000 bytes
GeoPhysVol[48]	161 instances	24048 bytes
GeoMaterial[64]	143 instances	3584 bytes
GeoMaterial[128]	110 instances	2304 bytes
GeoPhysVol[40]	97 instances	18816 bytes
GeoElement	97 instances	7760 bytes
GeoMaterial[136]	79 instances	1392 bytes
GeoTubs	39 instances	2496 bytes
GeoMaterial[32]	9 instances	288 bytes
GeoPcon[32]	3 instances	384 bytes
GeoPcon[48]	3 instances	384 bytes
GeoPcon	3 instances	336 bytes
GeoPcon[72]	3 instances	304 bytes
GeoTrd[56]	3 instances	48 bytes
GeoPhysVol[120]	2 instances	80 bytes
GeoPcon[80]	1 instances	128 bytes
GeoLogVol[56]	1 instances	16 bytes

Total GeoModel object allocation: 19.4294MB

XML Parser

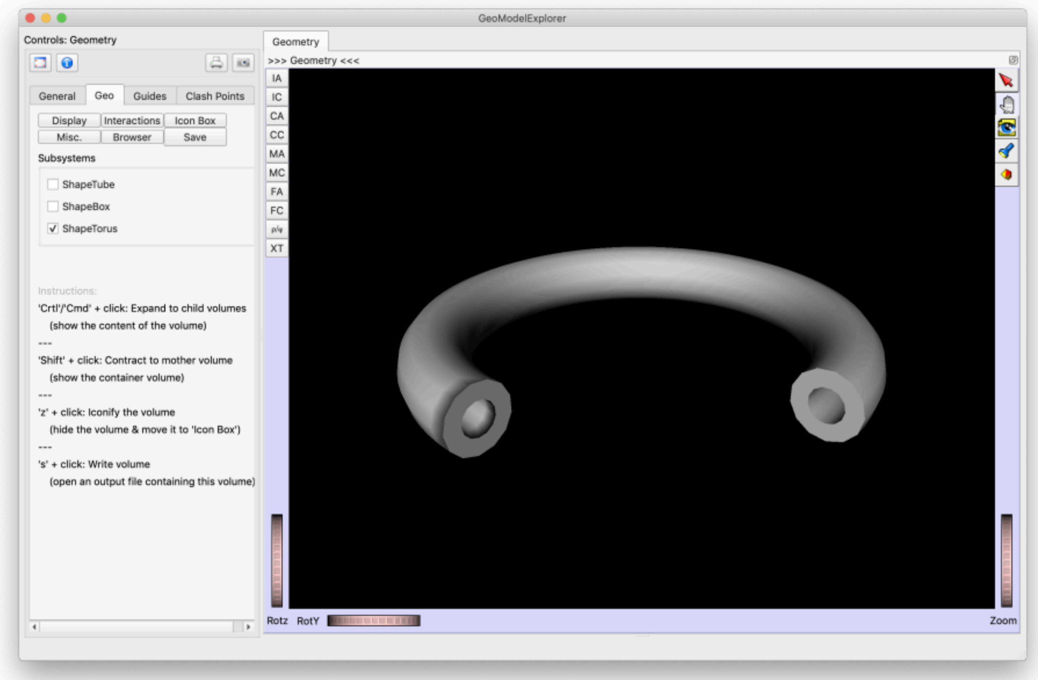
- The GeoModelXML package is part of GeoModelTools and lets users describe their geometry with **XML instructions** and get translated in **GeoModel nodes**
- The new GeoModelXML package is a standalone and **updated version** of the original package earlier developed for an ATLAS experiment sub-detector*
- The use of XML files allow for:
  - **fast updates** to the geometries ==> No need to recompile when modifying
  - **versioning** & storage on Git** ==> XML can be tagged to created versions/revisions
- GeoModelXML now offers **new** convenient ***positioners***, XML entities to place volumes in the 3D space
- **New shapes** are being added

```xml
<shapes>
    <torus name="Torus" rmin="TubeInnerRadius"
                        rmax="TubeOuterRadius"
                        rtor="TorusRadius"
                        sphi="0"
                        dphi="0.5" />
</shapes>
```

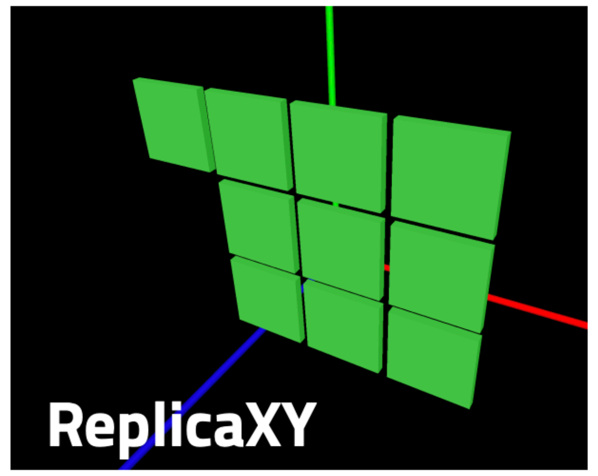
XML code

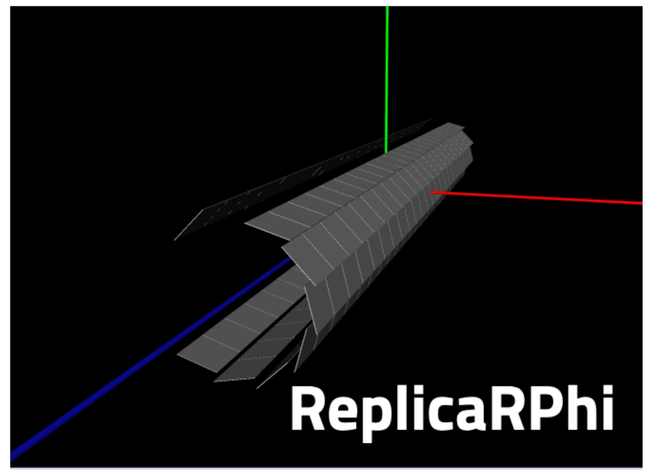
GeoModel tree

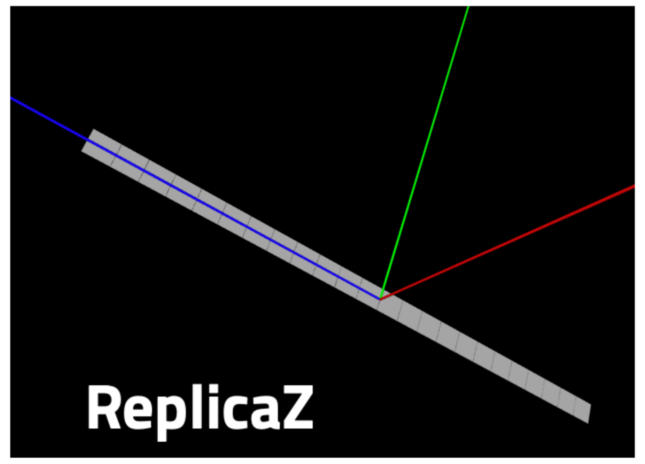
GeoModel conversion



Visualization



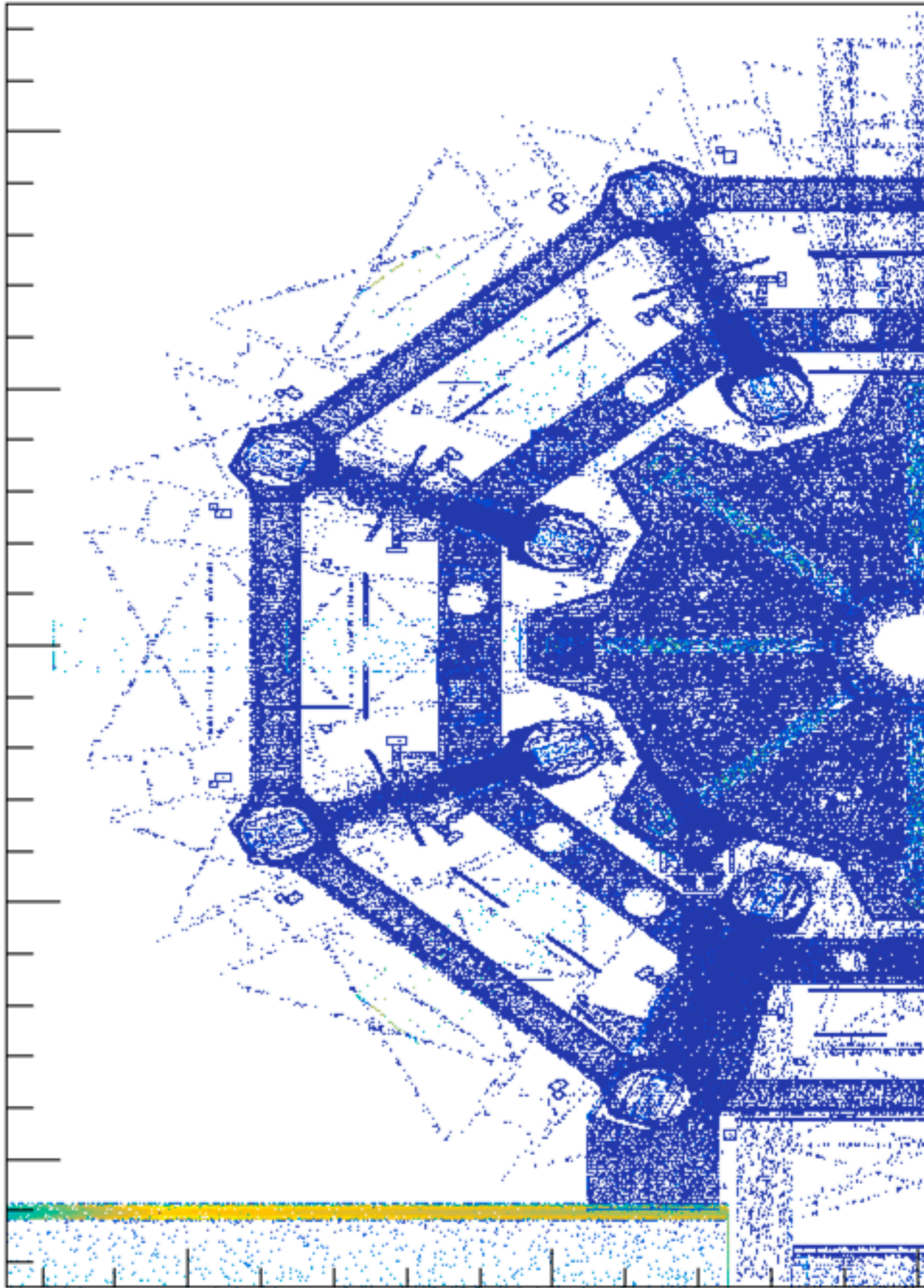




* see backup slides for refs

** or any other SCM/VCS, of course

Standalone Simulation Tools



Standalone Geant4 simulation tools

- The FullSimLight sub-package implements a **Geant4-based application** to run a **standalone simulation on any GeoModel geometry**
- Users can **choose what to simulate**: a whole detector, a part of it, or a single volume only
- Supported **geometry input**: SQLite (.db), GDML (.gdml) and GeoModel plugins (.dylib/.so)
- FullSimLight consists of **different tools**:
  - **NEW! gmmasscalc**: it calculates the inclusive and exclusive mass of a geometry *(see next slide)*
  - **fullSimLight**: a light particle transport simulation (geometry, transport in magnetic field and basic physics scoring) *(see backup slides)*
  - **gmgeantino**: a tool to generate *geantino* maps from your input geometry *(see backup slides)*
  - **gm2gdml**: a tool to convert GeoModel-based geometries into the GDML format *(see backup)*
  - **gmclash**: it runs clash detection on your input geometry, producing a JSON file report *(see backup slides)*

See the "Extra Material" slides at the end, for additional information!

New: gmmasscalc - Mass calculation

- The **new** GeoModelMassCalculator (*gmmasscalc*) is a command-line tool that **calculates** the inclusive and exclusive **mass of a given geometry:**
  - **exclusiveMass** is the mass of the considered volume only (from which the volumes occupied by the daughters volumes have been subtracted)
  - **inclusiveMass** is the mass of the considered volume, comprehensive of the masses of the respective daughters (propagated in an iterative way to their daughter volumes).

- The mass report is given in an **output JSON file**

- By default (if the optional flag are not used) *gmmasscalc*, takes the main *World Volume*, and calculates the inclusive and exclusive masses of the respective daughters

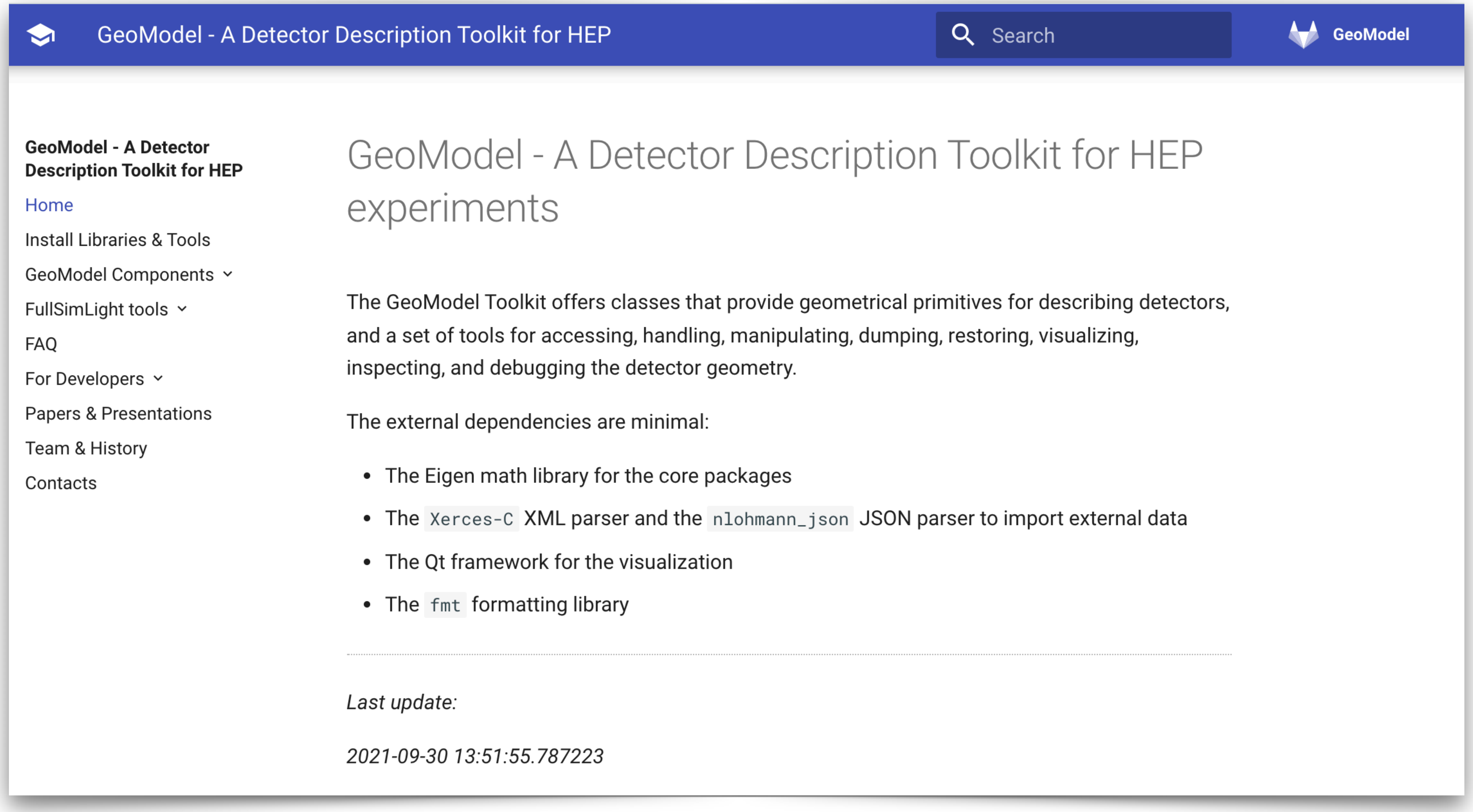

```
{
"exclusiveMass[kg]": 1.3358800280858636,
"inclusiveMass[kg]": 133.8127326258405 4,
"logicalVolumeName": "SCT_ForwardC",
"material": "Air",
"physicalVolumeName": "SCT",
"volumeCopyNo": 16969,
"volumeEntityType": "G4Tubs"
}
```


```
./gmmasscalc -g caloBarrel.db -o mass_caloBarrel.json
```

Documentation, Source code, License, History

Documentation

- Users can find **documentation** on GeoModel modules and tools at:
  - https://geomodel.web.cern.ch/home/
- Documentation is written in **Markdown** with **MKDocs**, to **let users contribute** to it through Git Merge Requests
- It is **built and published automatically** (with CI jobs) on new additions and changes

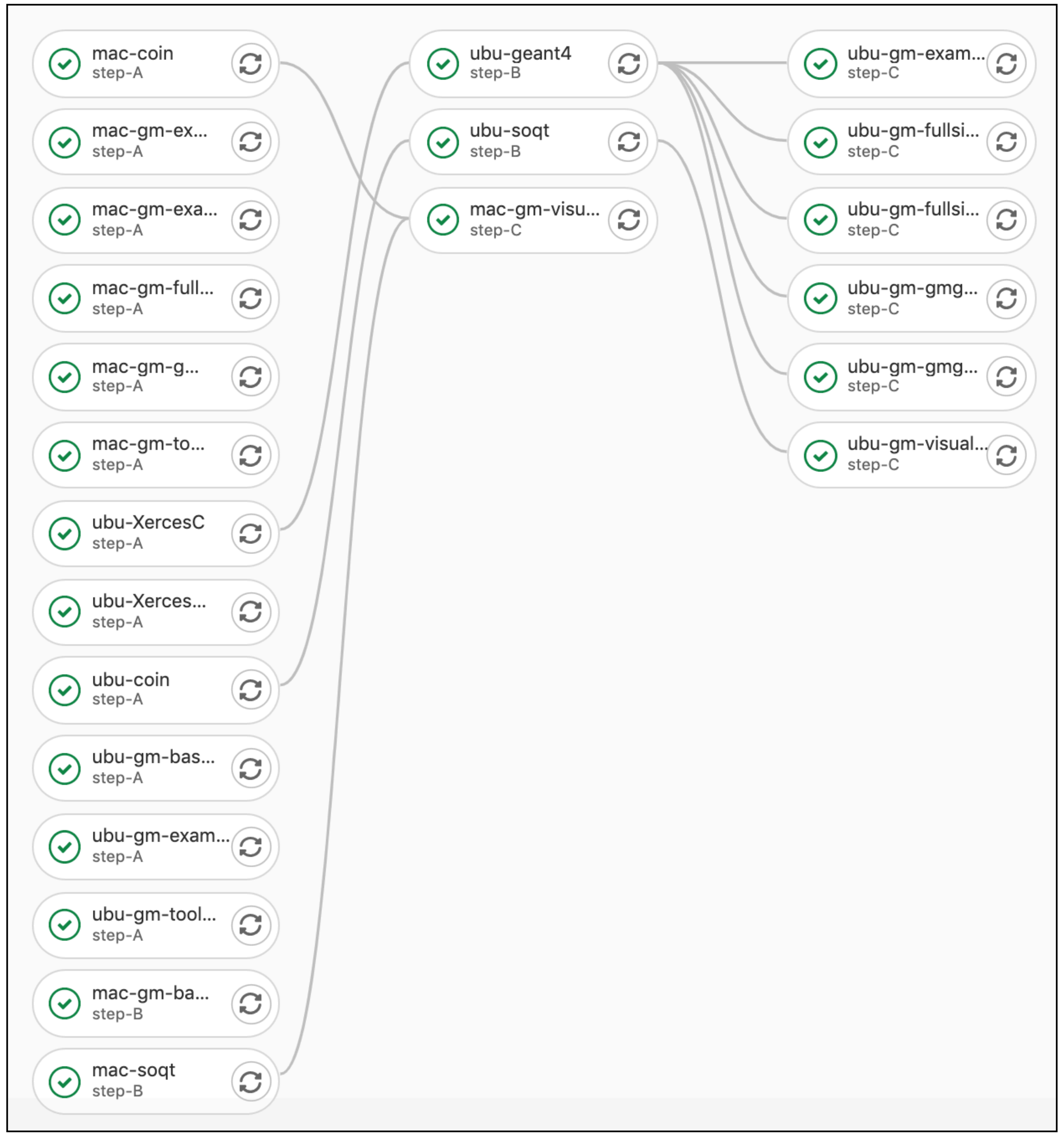


Source code & License

- The source code is **fully public**, and it is hosted on the CERN **GitLab** servers:
  - https://gitlab.cern.ch/GeoModelDev/GeoModel

- The code is tested with **continuous integration** (CI) jobs, covering the **major platforms** used by our users:
  - macOS
  - Ubuntu Linux

- The whole GeoModel code is **released** under the **permissive Apache 2.0 license***

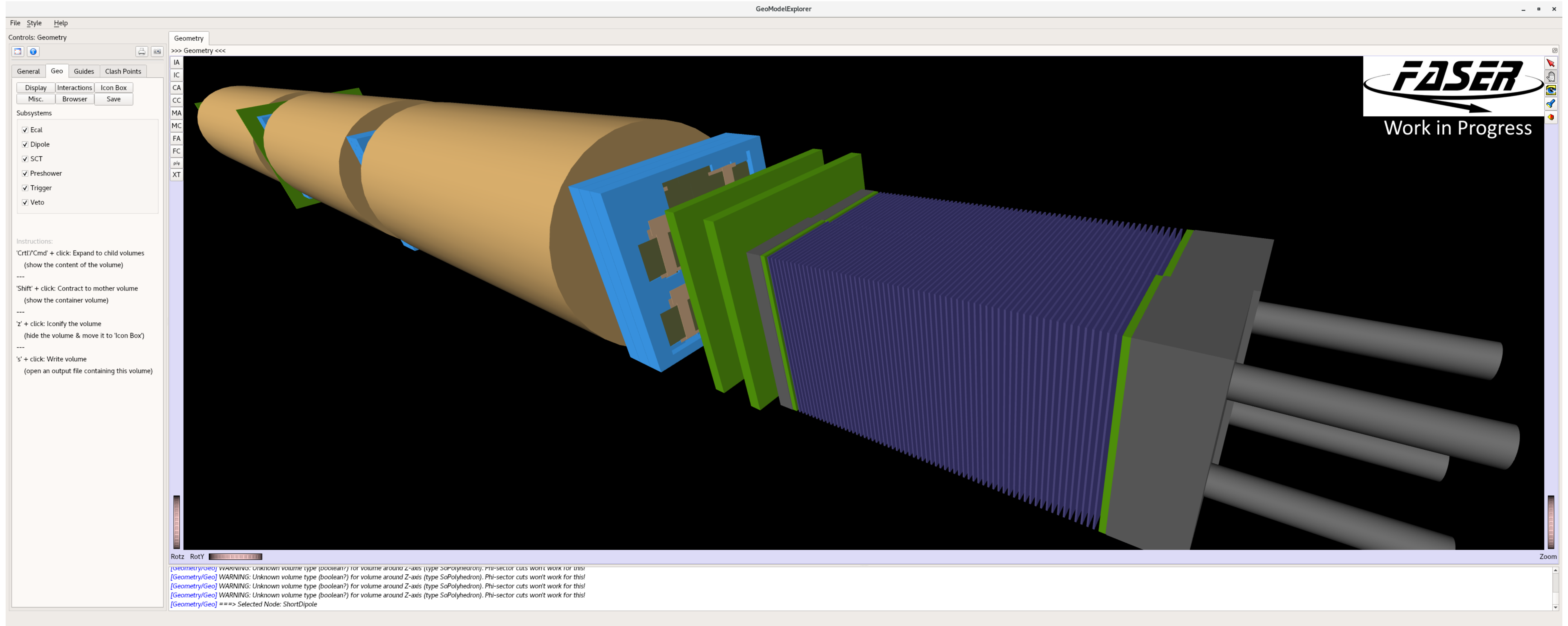


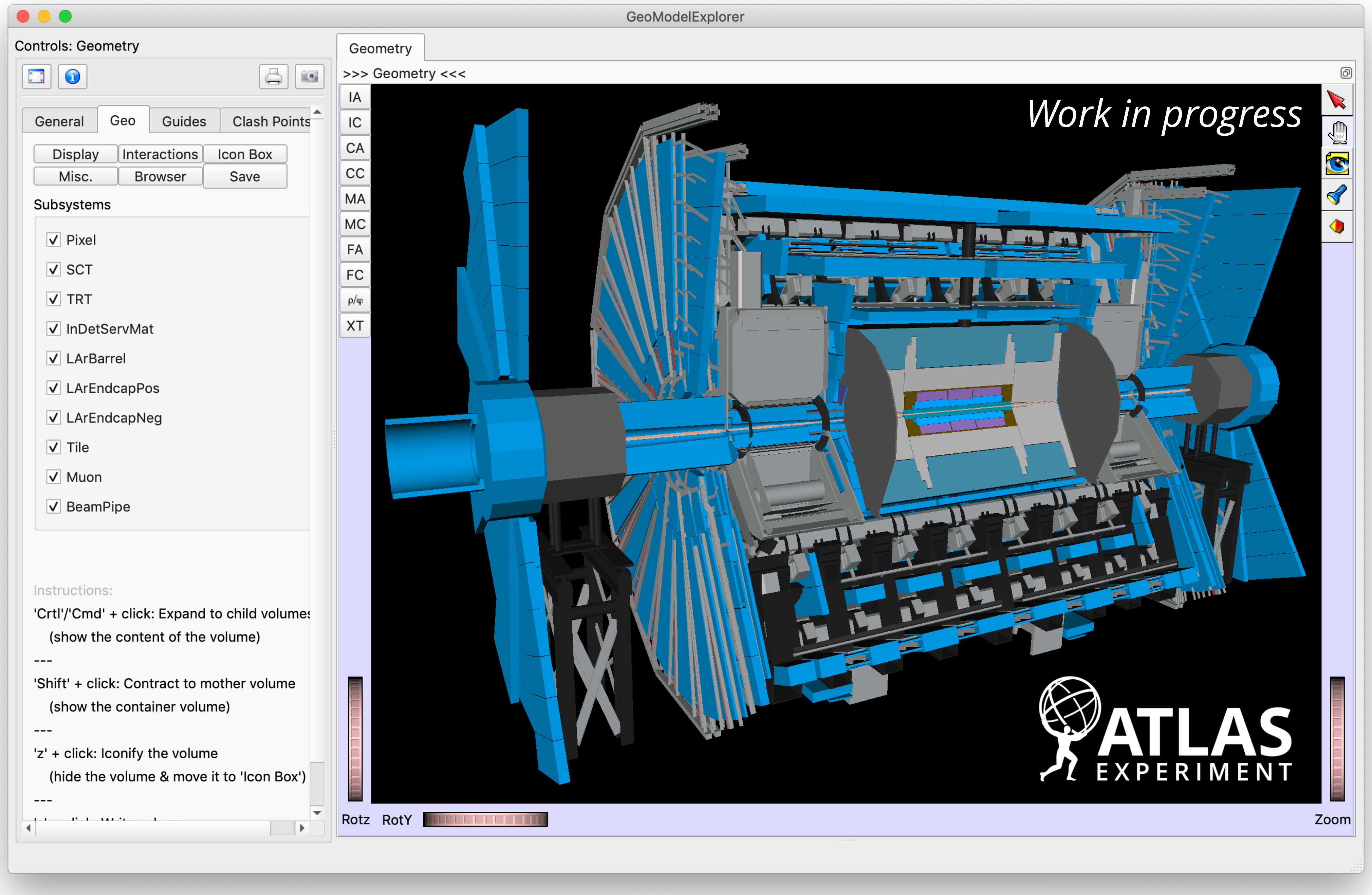
** this is the license suggested by the HSF Licensing Group and today used by many HEP experiments for their code*

GeoModel in HEP

- The **ATLAS experiment adopted** the new GeoModel suite as the foundation of the whole **new** Detector Description **architecture** for the **LHC *Run4*** data taking period

- The **FASER experiment adopted** GeoModel for **its** new Detector Description **architecture**





*The **FASER** and **ATLAS** detectors described with GeoModel and visualized in GeoModelExplorer*

GeoModel: 20 years experience

- The **GeoModelKernel** package —containing the core classes— has been part of the ATLAS framework and the foundation of the ATLAS Detector Description for 20 years

- The **new toolkit** made the core classes **standalone and experiment-independent**, and added a whole set of **new functionalities** and tools.
  Its **development is backed and supported by the ATLAS experiment**

- During the last few years, the new GeoModel tool suite **has steadily grown into a full-featured, yet lightweight, tool-suite**, and we have **presented** the new tools and packages at the major HEP conferences. *See backup slides for links to the past presentations*

GeoModel

a lightweight & powerful
detector description toolkit

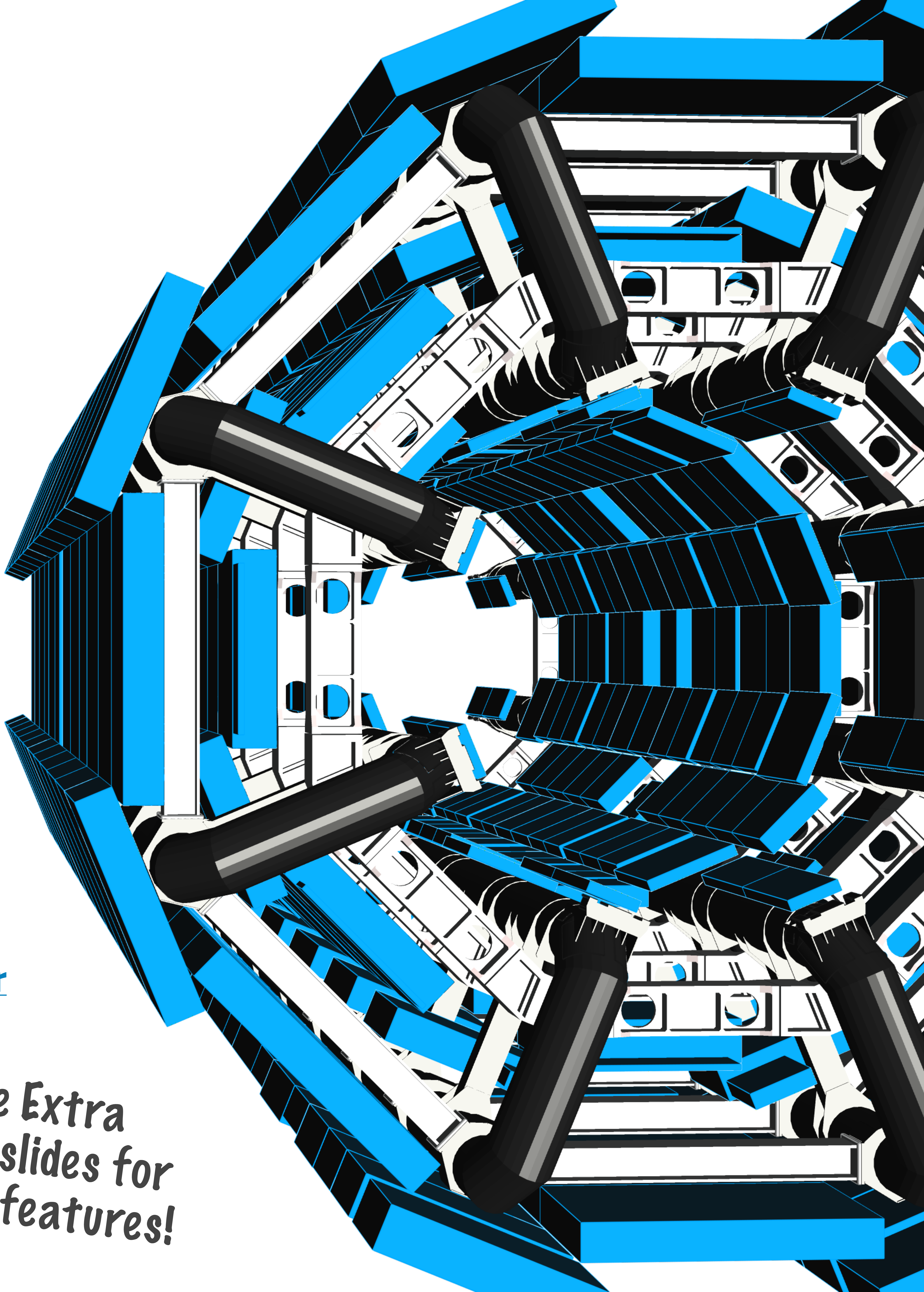
made for every experiment

based on 20 years experience

backed by ATLAS.

Questions?

Riccardo Maria Bianchi / riccardo.maria.bianchi@cern.ch / @SubatomicHunter



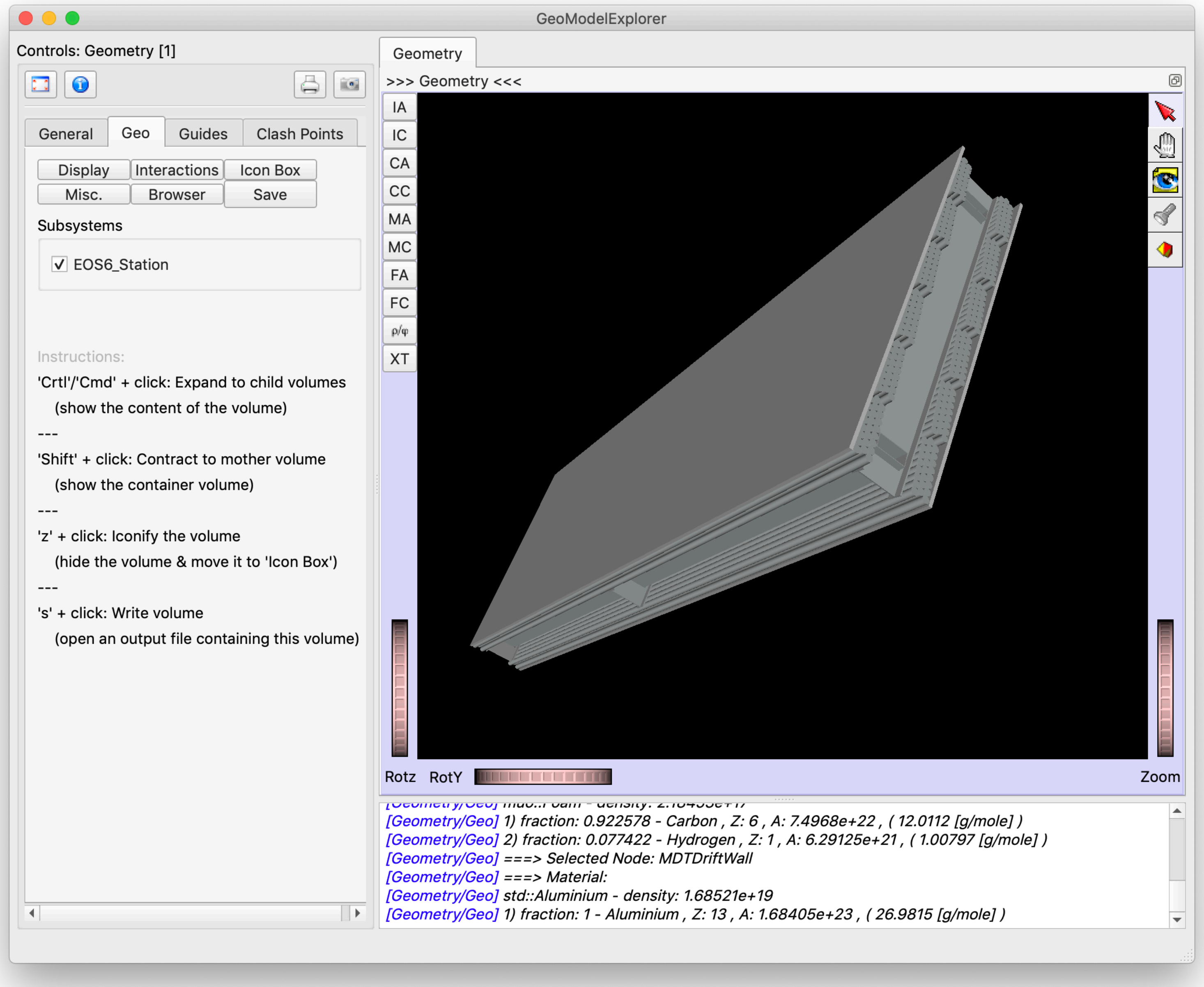
See the Extra Material slides for additional features!

EXTRA MATERIAL

EXTRA - Visualization

Visualization - Customizable visualization parameters in JSON

- GeoModelExplorer visually represents materials and volumes in the geometry, based on visualization parameters defined by the user and stored **in JSON files**
- That means that the **appearance** of materials and volumes **can be modified on-the-fly**, with **no need to recompile** GeoModel



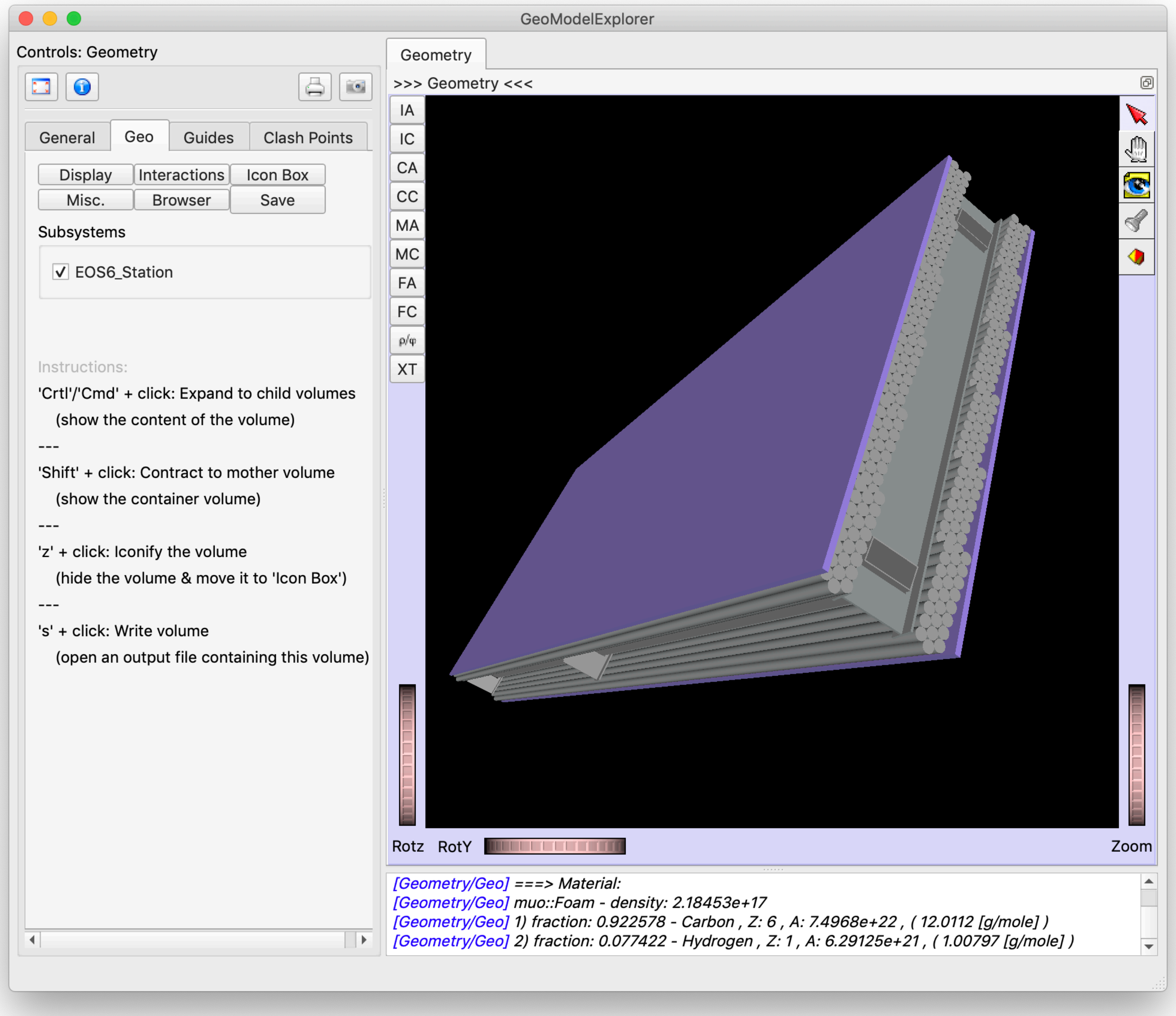
A detector, with the default colors

+

```json
{
    "diffuse": {
        "B": 1.0,
        "G": 0.5249999761581421,
        "R": 0.6000000238418579
    },
    "name": [
        "muo::Foam"
    ],
    "shininess": 0.800000011920929,
    "specular": {
        "B": 0.27000001072883606,
        "G": 0.27000001072883606,
        "R": 0.27000001072883606
    },
    "transparency": 0.0
}
```

New user-defined colors

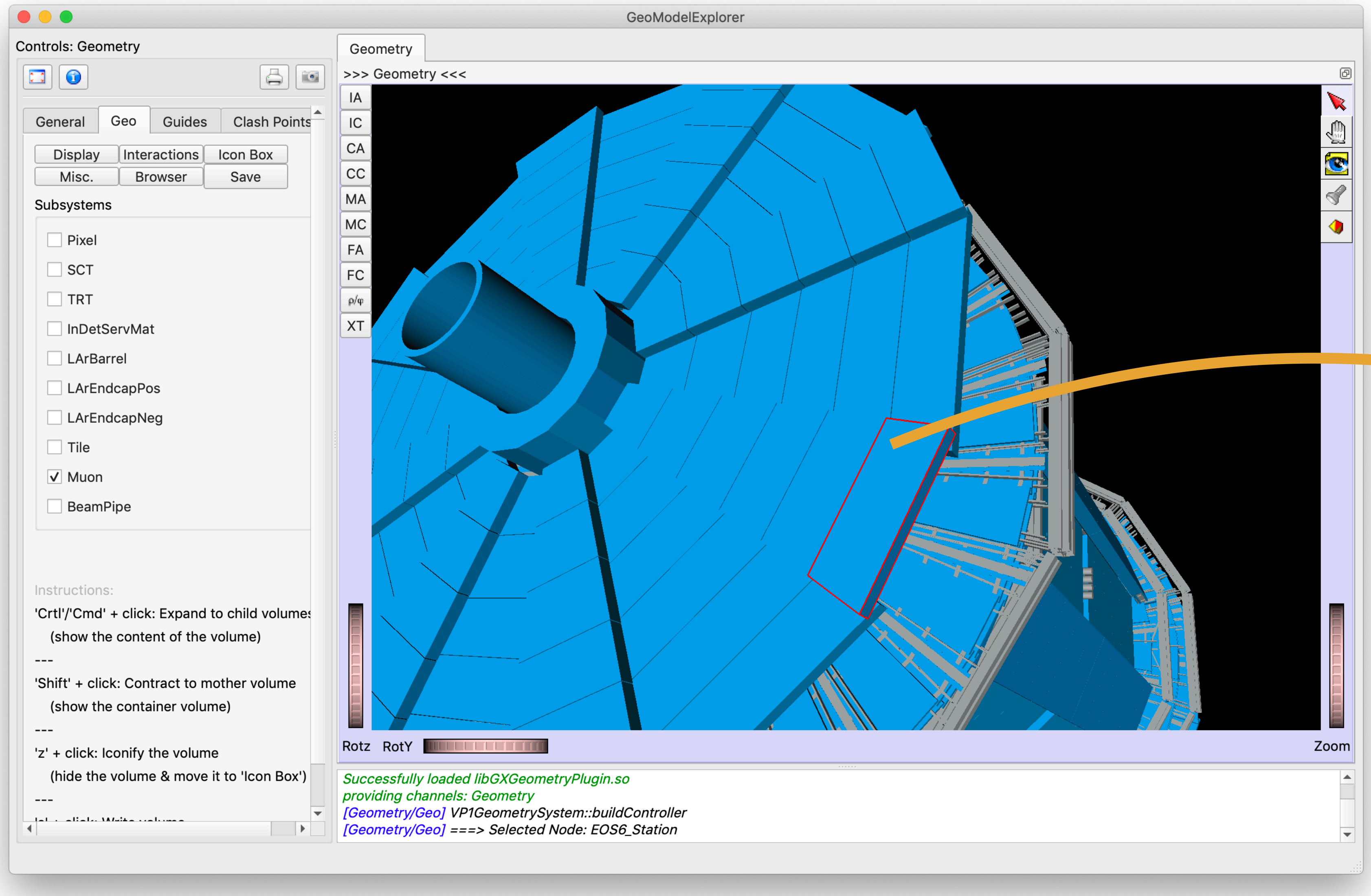
=



Same detector, with the new colors

Visualization - Saving parts to sub-trees

- Users can select a **specific detector volume** on screen and **save it and** all its **child** sub-volumes into a separated GeoModel file
- The selected piece of detector can be re-opened in an **isolated environment** for additional **inspection**, checked with test tools, **separately simulated**, or **be used as a sub-tree** into another tree
- Users can also save into a GeoModel SQLite file the **entire 3D scene**, to get a persistent copy of the geometry tree visualized on the screen



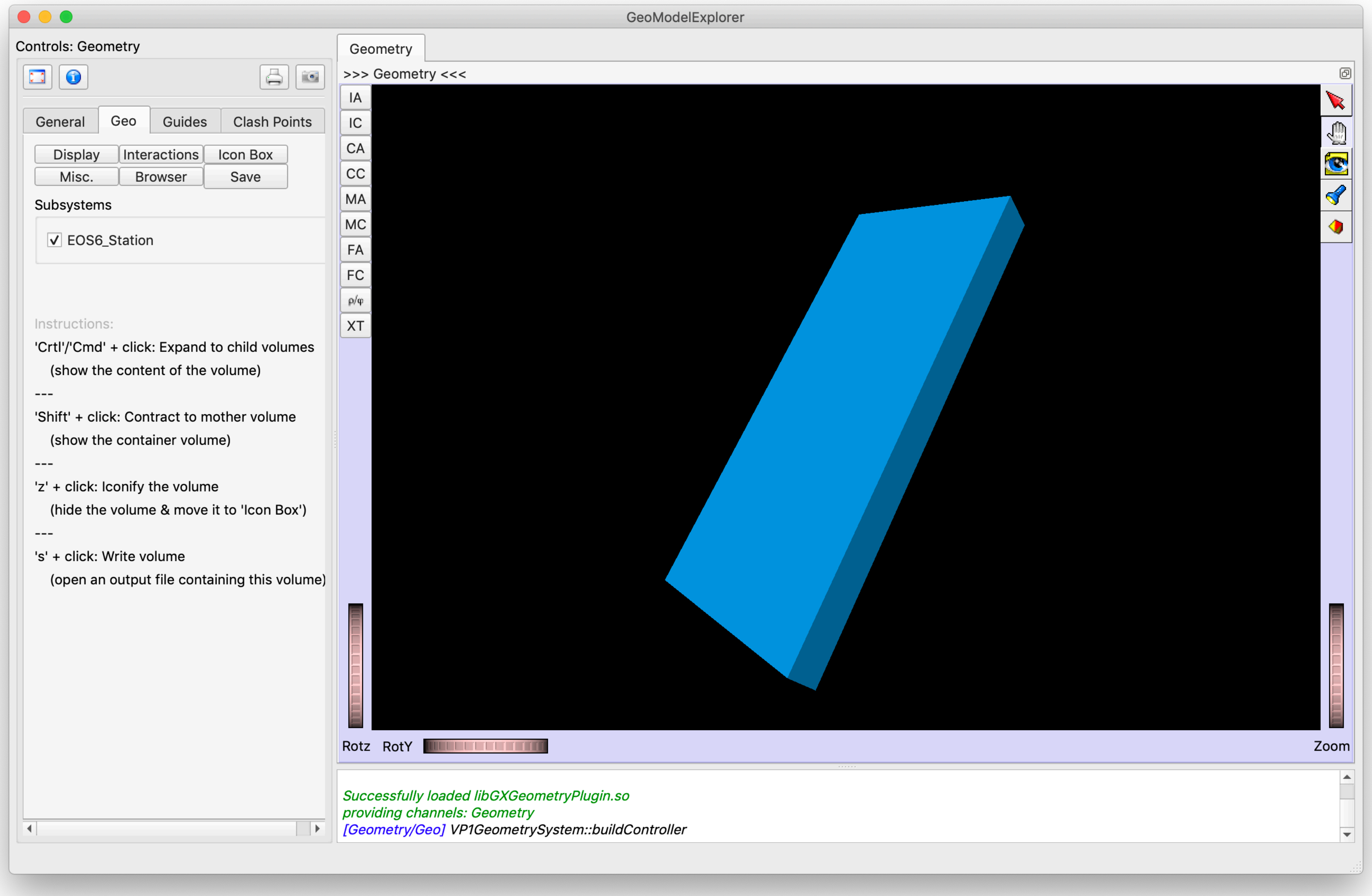
SAVE

Here, we dump the geometry of a single muon chamber into a file...

SQLite file

LOAD

...then we load it, to inspect/visualize it in an isolated environment



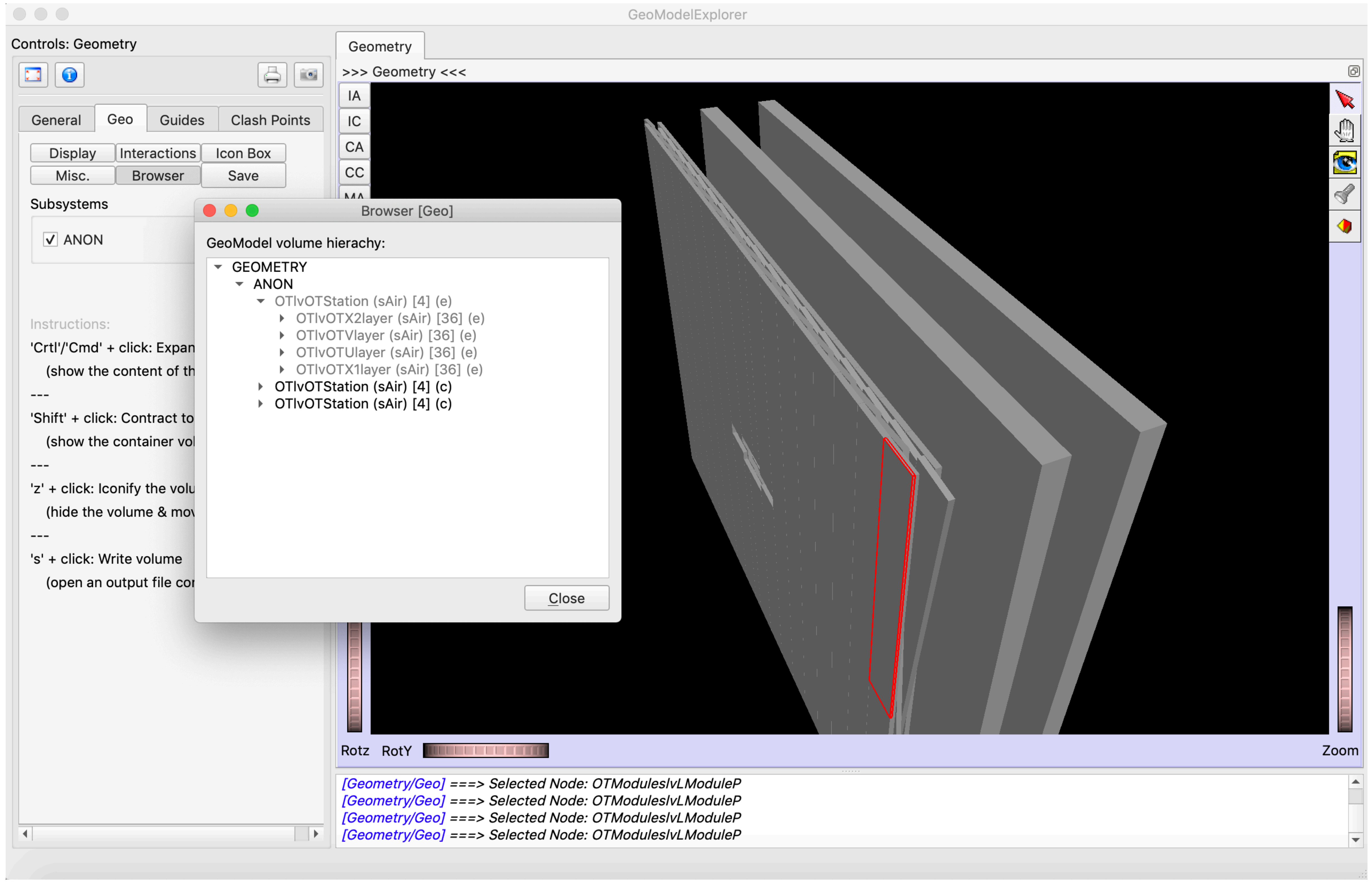
Visualization - The graphics SW stack

- The GeoModelExplorer visualization tool uses a few software external libraries:
  - Coin3D + OpenGL —> for the 3D graphics
  - Qt5 —> for the graphical user interface (GUI)
  - SoQt —> as a glue package between the 3D graphics layer and the GUI
- An upgrade to Qt 6 is foreseen for the near future
- An R&D sub-project is also ongoing about the replacement of Coin/SoQt with Qt3D

EXTRA - Command-line Tools

Tools - *gdml2gm* - Convert GDML to GeoModel

- The GDML2GM package offers the *gdml2gm* command-ine tool to **convert** input **GDML** detector descriptions **into GeoModel**
- That lets **users use the feature-rich & standalone GeoModelExplorer to inspect** existing geometries implemented in **GDML**
- In the figure, the Geant4 GDML test file *(test.gdml)* is visualized in GeoModelExplorer
- The GDML-GeoModel converters are work -in-progress, not all GDML entities are supported yet; but more entities are being added as they are needed by users



EXTRA - Standalone Simulation Tools

Simulation - *fullSimLight* - Standalone particle transport simulation

- FullSimLight (*fullSimLight*) is a **Geant4-based** lightweight application that offers **particle transport** simulation **through a given piece of geometry**
- It can use both sequential and multithreaded Geant4 builds
- it uses the **Geant4 particle gun** as default primary generator, but **also supports *Pythia***
- A constant **magnetic field** can be set through a macro command
- It supports, as **input formats**, GeoModel SQLite files, GeoModel plugins and GDML files

```
./fullSimLight -m ../share/FullSimLight/macro.g4 -g myGeometry.db
```
on a GeoModel SQLite file

```
./fullSimLight -m ../share/FullSimLight/macro.g4  -g libMyGeoPlugin.dylib
```
on a GeoModel plugin

```
./fullSimLight -m ../share/FullSimLight/macro.g4 -g myGeometry.gdml
```
on a GDML file

Simulation - *gmclash* - clash detection

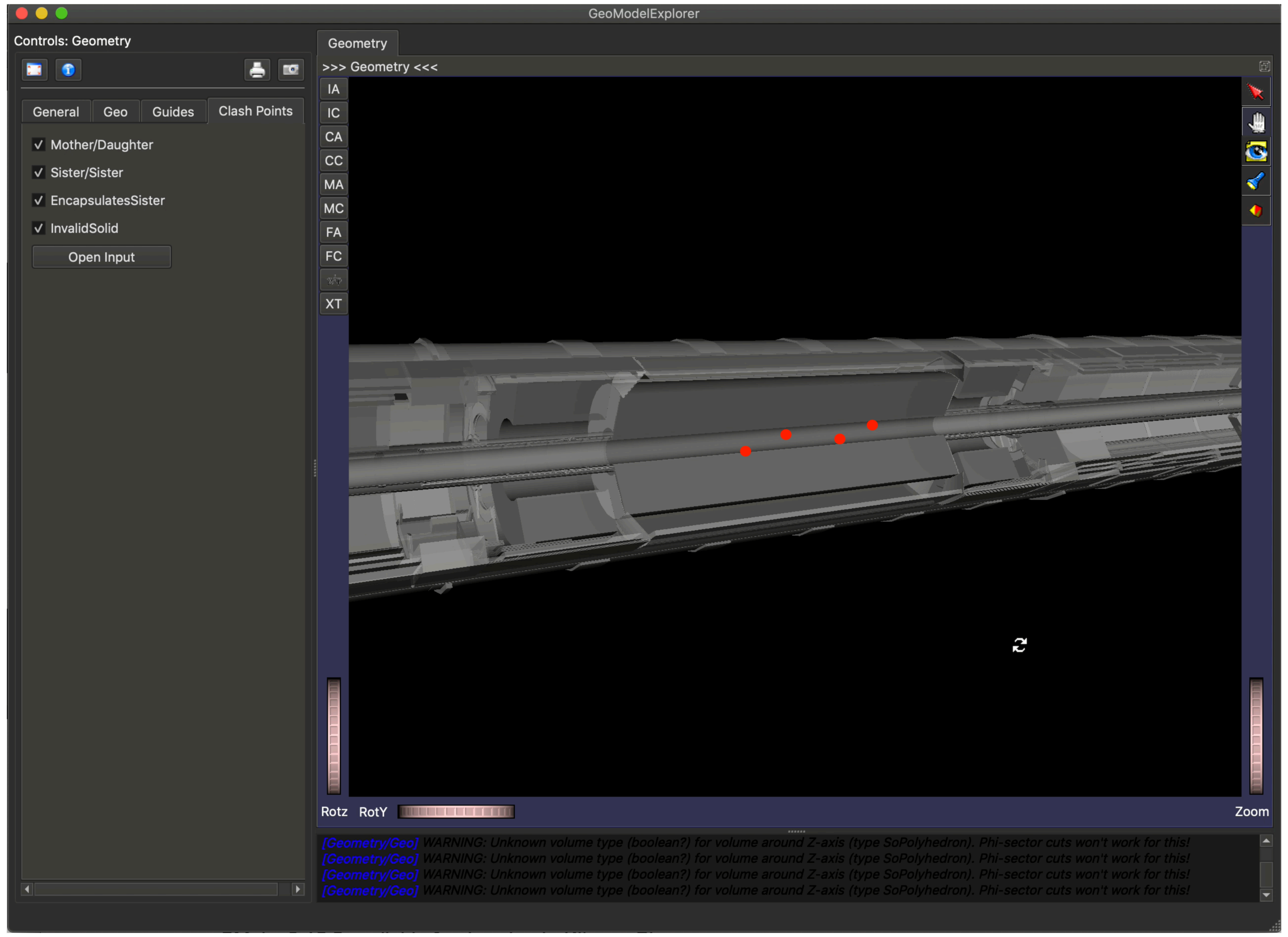
- GeoModelClash (*gmclash*) runs Geant4-based geometry overlap checks on a given geometry

```
./gmclash -g LArBarrel.db -o cr_LArBarrel.json
```

- The **clash report** is given in an output **JSON file**

```
"distance[mm]": 51.21328179620343,
"typeOfClash": 1,
"volume1CopyNo": 16969,
"volume1EntityType": "G4Tubs",
"volume1Name": "LAr::Barrel::Cryostat::Cylinder::#13Phys",
"volume2CopyNo": 16969,
"volume2EntityType": "G4UnionSolid",
"volume2Name": "LAr::DM::SectorEnvelopes2r",
"x": -1.736718203796568,
"y": -1263.348806272393,
"z": -166.75403155804725
```

- The clash **report** can also be **loaded** in GeoModelExplorer, **to visualize the points of clash**



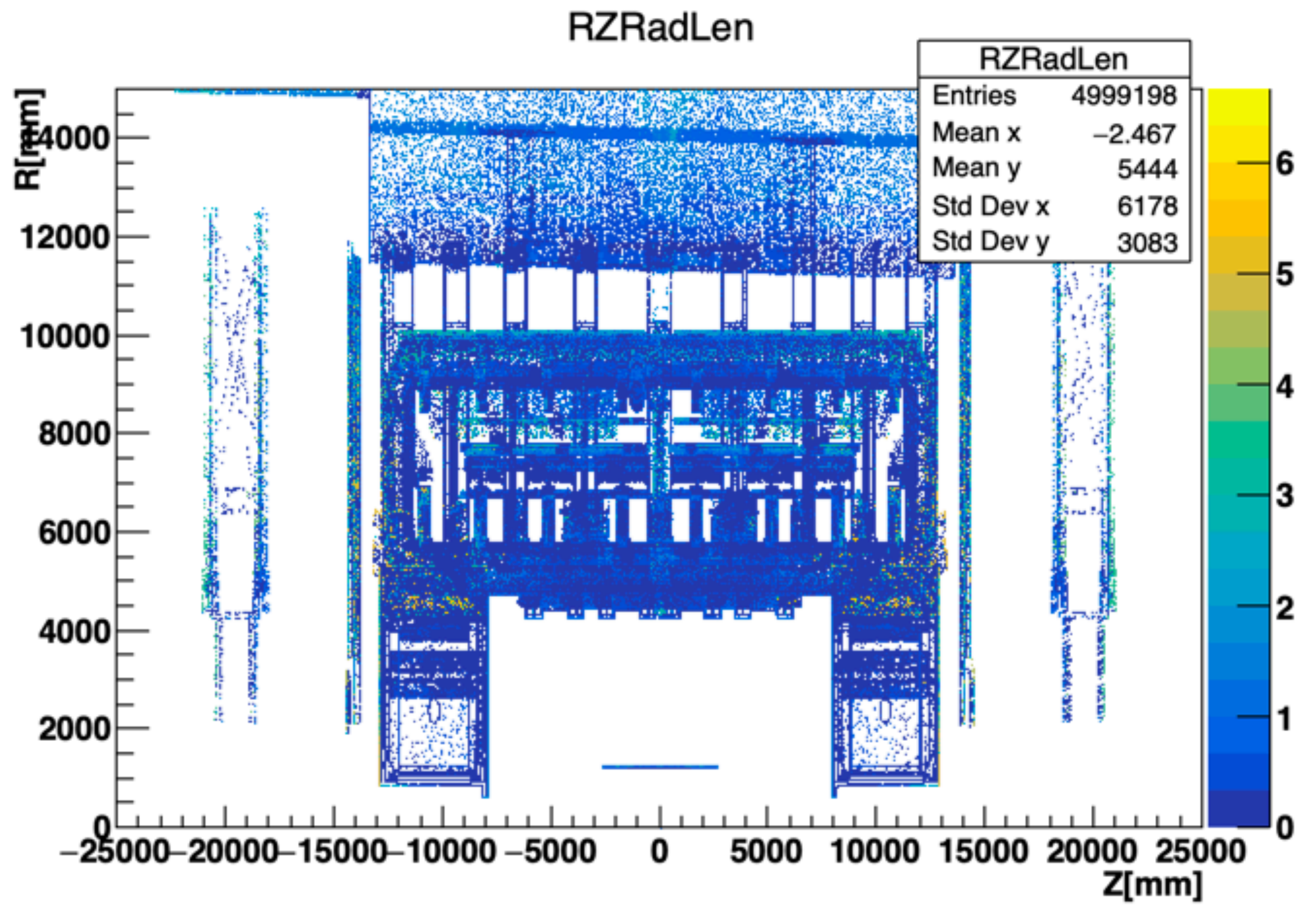
clash points highlighted in red in GeoModelExplorer

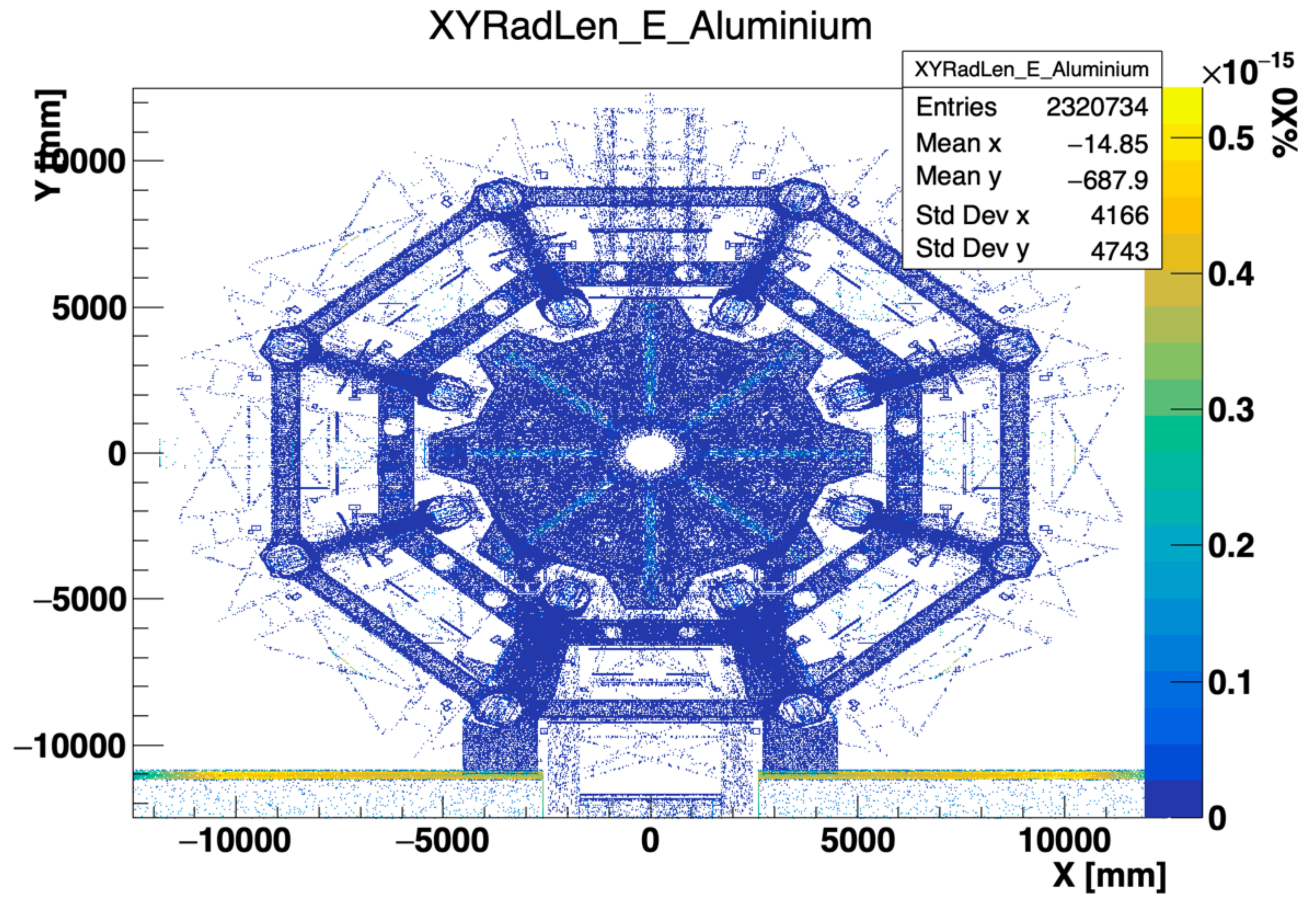
Simulation - *gmgeantino* - Standalone geantino scans

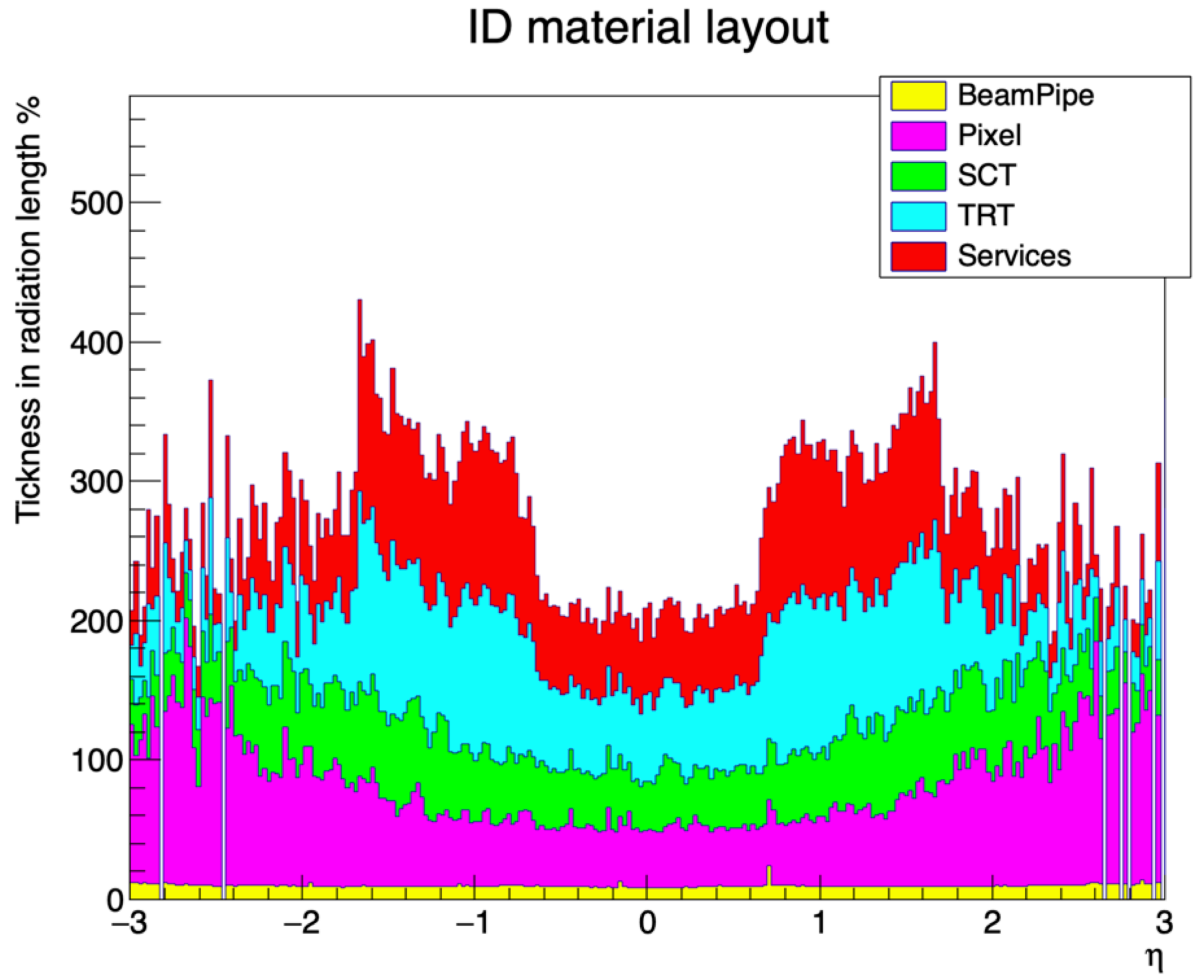
- GeoModelGeantino (gmgeantino) is a Geant4-based application that **produces geantino maps for a given geometry**. It supports .db/.gdml/.dylib/.so geometry formats

```
./gmgeantino -m ../share/FullSimLight/geantino.g4 -g LArBarrel.db -e -d
```

- it writes out the **geantino maps** in a ROOT file. However, **it does not depend on ROOT**, because it uses the *G4AnalysisManager* to create/fill/write 1D and 2D Profiles







Past GeoModel presentations

- Past GeoModel presentations at major HEP computing&software conferences:
  - **vCHEP 2021**: https://doi.org/10.1051/epjconf/202125103007
  - **CHEP 2019**: https://doi.org/10.1051/epjconf/202024502029
  - **CHEP 2018**: https://doi.org/10.1051/epjconf/201921402035
  - **ACAT 2017**: https://doi.org/10.1088/1742-6596/1085/3/032035
  - **CHEP 2016**: https://inspirehep.net/literature/1638122

References

- The **original GeoModelXml** package had developed as part of the software framework of the ATLAS experiment, on top of which the **new**, **standalone**, **updated GeoModelXML** is built. See, for example:

  - N.Hessey, *"User Manual and Guide to developing geometries with the GeoModelXml Package"*, ATLAS Public Note, https://twiki.atlas-canada.ca/pub/AtlasCanada/ITk/gmx2geo.pdf