TRACER/CORE

THE CROSS-EXPERIMENTAL INTERACTIVE DETECTOR DISPLAY

SHARMAZANASHVILI Alexander
Georgian Technical University

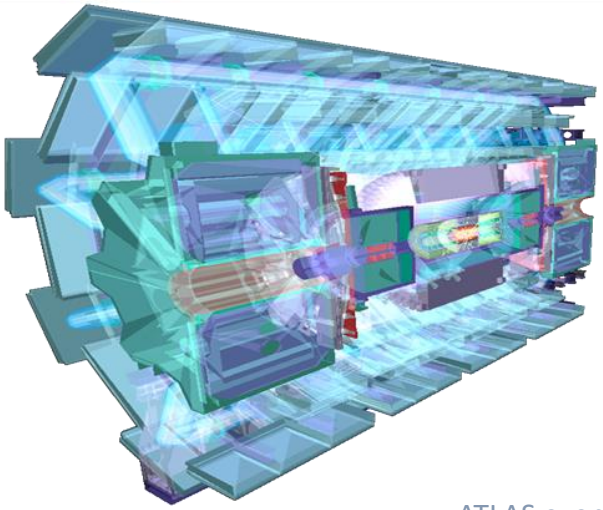
ATLAS Tracer Team:
KHELASHVILI Levan
KOBAKHIDZE Shota
UDZILAURI Nikoloz
PHATARIDZE Lasha

About

- Tracer is Interactive Detector Display (IDD) Application (called by Kate Shaw, many thanks to her!)
- This is not the event display!
- The main purpose of Tracer is to play the cognitive role of experiments and facilities for the education and outreach and also be the extension for physics and engineers
- Tracer can visualize events – tracks, jets, MET, energy deposits, but this is not professional event display and can not be considered as a replacement of Atlandis or VP1

About

- Tracer can be used for the other experiments or HEP projects as well, without changing of source code. This is a cross-experimental software application
- Tracer loads the events from the standard JiveXML and geometries from the Jason. So, the application can be adapted to the other detector display by putting the corresponding data into the JiveXML and Jason

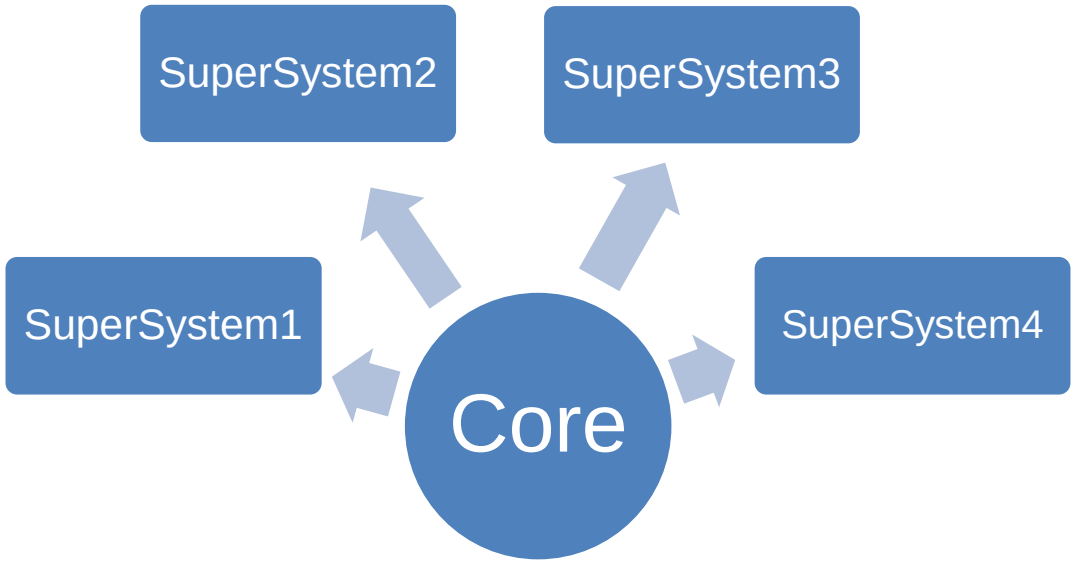


Concept

- Tracer intended for the very large domain of users, which brings special requirements to the application:
  1. Cross-platform compatibility – for ensuring application workability on all type of hardware – desktop PC, notebooks, tablets, phones and all type of software platforms – MacOS, Win, Linux, Android
  2. The application should not require any installations. It should be a click-and-go concept
  3. The application should have a very well developed and intuitive user interface

Architecture

- Tracer has a flexible architecture for easy adaptation to the user-specific tasks
- The architecture contains core and number of super systems, developed on the base of core



Architecture

- Tracer/Core has basic functionalities, less depending on user-specific tasks:
  1. Geometry loaders
  2. Event loaders
  3. 3D scene organizer
  4. Grid
  5. Haptic controls
  6. Geometry cuts
- Super systems are using basic functionality of core (not necessarily all) + functionality of user-specific tasks

Architecture

- The Tracer/Core is now ready and fully developed. It can be reached from here http://cadcamge.ch/at/r3.3/#
- The application is open-source on the Gitlab. Can be reached here https://gitlab.cern.ch/asharmaz/tracer-r3.3
- It took 3 years of 5FTE team work
- Tracer/Core consists of 155 modules and ¬7’000 code line
- Tracer/Core has detailed geometries of all ATLAS detector components coming partly from Geant and partly from CATIA CAD DB

II. Developers View

- The development of the Tracer/Core has been finished. All components are on the place and all functionalities have been done
- Tracer is a web-based application developed on the base of WebGL/three.js library
- Development of IDD as a three.js web-based application is the very tough project requires, well-qualified js programming, good expertise in geometry descriptions and discovering original solutions in many particular cases
- Success is possible in case of the existence of a strong team and long term team contribution. Single, or part-time student involvement will never bring high-quality results

II. Developers View

- There are several main ingredients decide successful application development results at the end:
    I. Performance
    II. Geometry development
    III. Development of geometry cuts
    IV. User framework

I. Performance

- There is no binary code and browser interpret the source code on-line. Therefore, as much application grow the functionality, as much it becomes slower. Finally, it comes to the point when the scene is ‘freeze’ – low fps, or the browser kills the process because it is very slow and application crashes
- The development of a high performing engine, including high efficient and fast object loaders is absolutely necessary
- Parameters of scene have to express the good balance between the quality (rendering types, lighting, shading, texturing, etc.) and performance

II. Geometry Development

- It is impossible to draw a full detector geometry with three.js internal methods. So, geometry should be imported in the form of facets (triangles)
- It is absolutely necessary to simplify facet-based geometry before the import – 'unnecessary' triangles should be deleted. Otherwise, it becomes impossible to open all detector components in the scene – the application will be crashed
- The number of triangles in the scene for three.js is limited by 2'000'000. The un-simplified weight of the Pixel detector from Geant is 353'000 triangles and the full Inner detector Geant model has 1'4 Mln triangles. So without simplification, all detector components can not be visualized

II. Geometry Development

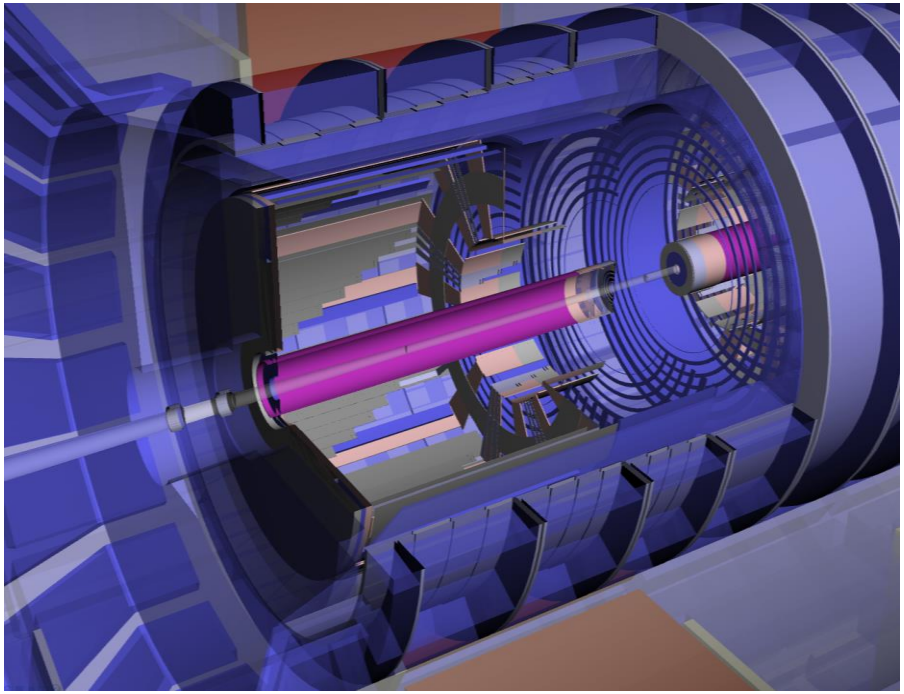
- In some cases very detailed geometries are necessary to visualize, coming from the CATIA CAD engineering database, which makes simplification more important
- We have developed our own life cycle for geometry simplification. The table below describes the results of simplification of ATLAS detector components

	Components		Orginal	Simplified	
1	Inner Detector	Pixel	353,015	101,908	71%
2	Inner Detector	SCT	283,042	89,772	68%
3	Inner Detector	TRT	707,072	66,944	91%
4	Lar EM	Barrel	102,086	37,000	64%
5	Lar EMEC	EndCap			
6	Lar HEC	EndCap			
7	Lar Fcal	EndCap	344,800	15,184	96%
8	Tile	Barrel	11,600	1,600	86%
9	Tile	Extended	81,000	40,232	50%
10	Muon Small Wheel	EndCap	73,000	29,168	60%
11	Muon Extra Wheel	EndCap	52,243	2,456	95%
12	Toroid Magnet	EndCap	40,137	20,384	49%
13	Feets		672,000	30,780	95%
14	Warm Structure		128,000	27,136	79%
15	Toroid Magnet	Barrel	24,000,000	86,752	100%
16	Muon Inner	Barrel	122,000	35,000	71%
17	Muon Middle	Barrel	384,000	50,000	87%
18	Muon Outer	Barrel	791,000	20,000	97%
19	Muon Big Wheel TGS1	EndCap	584,785	108,776	81%
20	Muon Big Wheel TGS2	EndCap	392,984	23,500	94%
21	Muon Big Wheel TGS3	EndCap	392,984	23,500	94%
22	Muon Big Wheel MDT	EndCap	250,000	40,000	84%
23	Muon Outer Wheel	EndCap	274,912	2,304	99%
Total			30,040,660	852,396	

III. Geometry Cuts

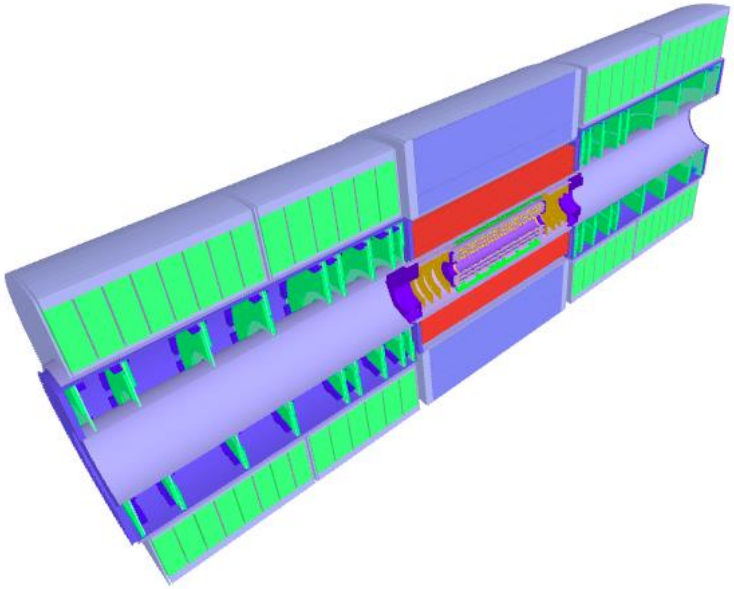
- Geometry cuts needed to visualize internal content of facilities
- Three.js has several methods for cut:

1. `Clipping/intersection` – analyze screen pixel matrix and hide ‘everything’ which is in front or behind the cutting plane. However, cutting surfaces are omitted and geometry has ‘empty’ spaces which Bring very poor visualization

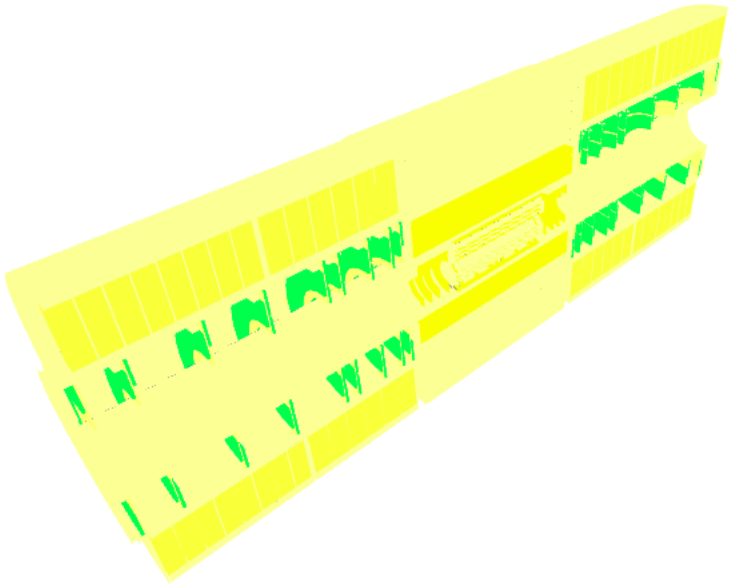


III. Geometry Cuts

2. `Clipping/stencil` – doing the same and fill in addition cutting surfaces, but in reality, it fills cutting planes and method can not separate cutting surfaces. So, everything is filled by one single colour



Pixel detector cut in Tracer



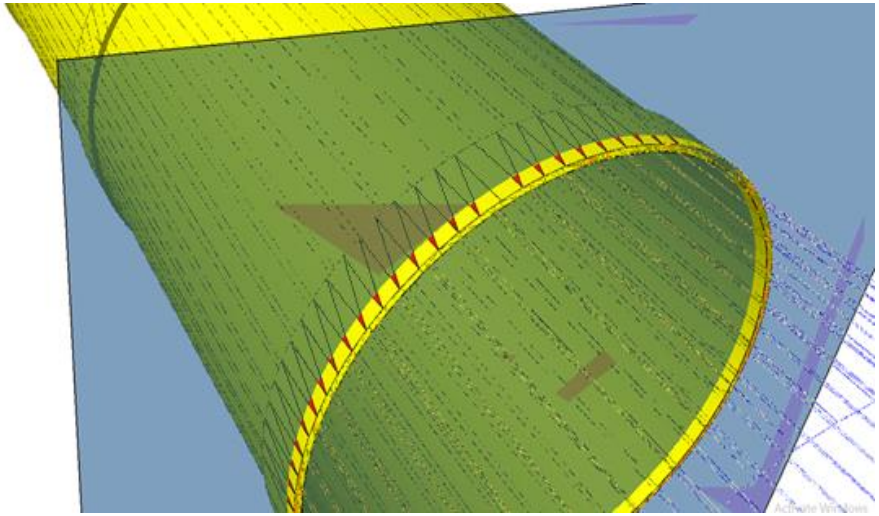
Pixel detector cut in three.js by `clipping/stencil` method

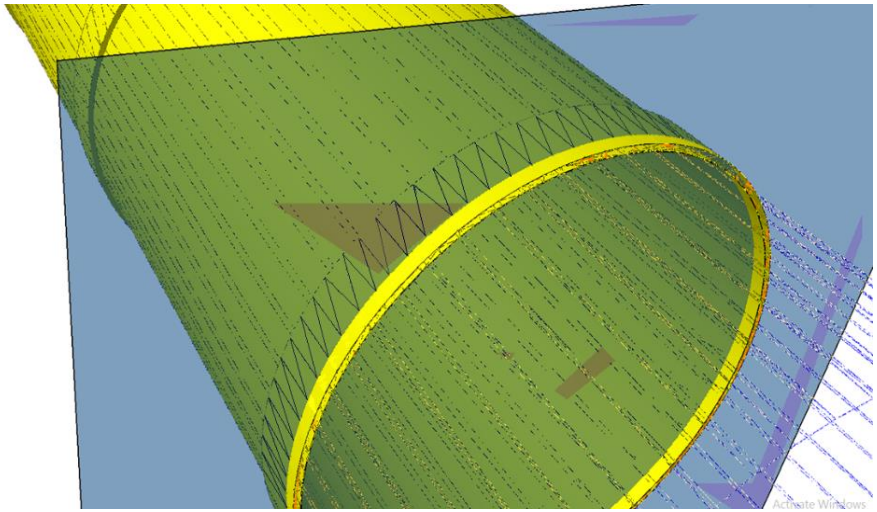
III. Geometry Cuts

- Real cuts are doing by Boolean processors – methods:

  `cub.substract; cub.union; cub.intersect`

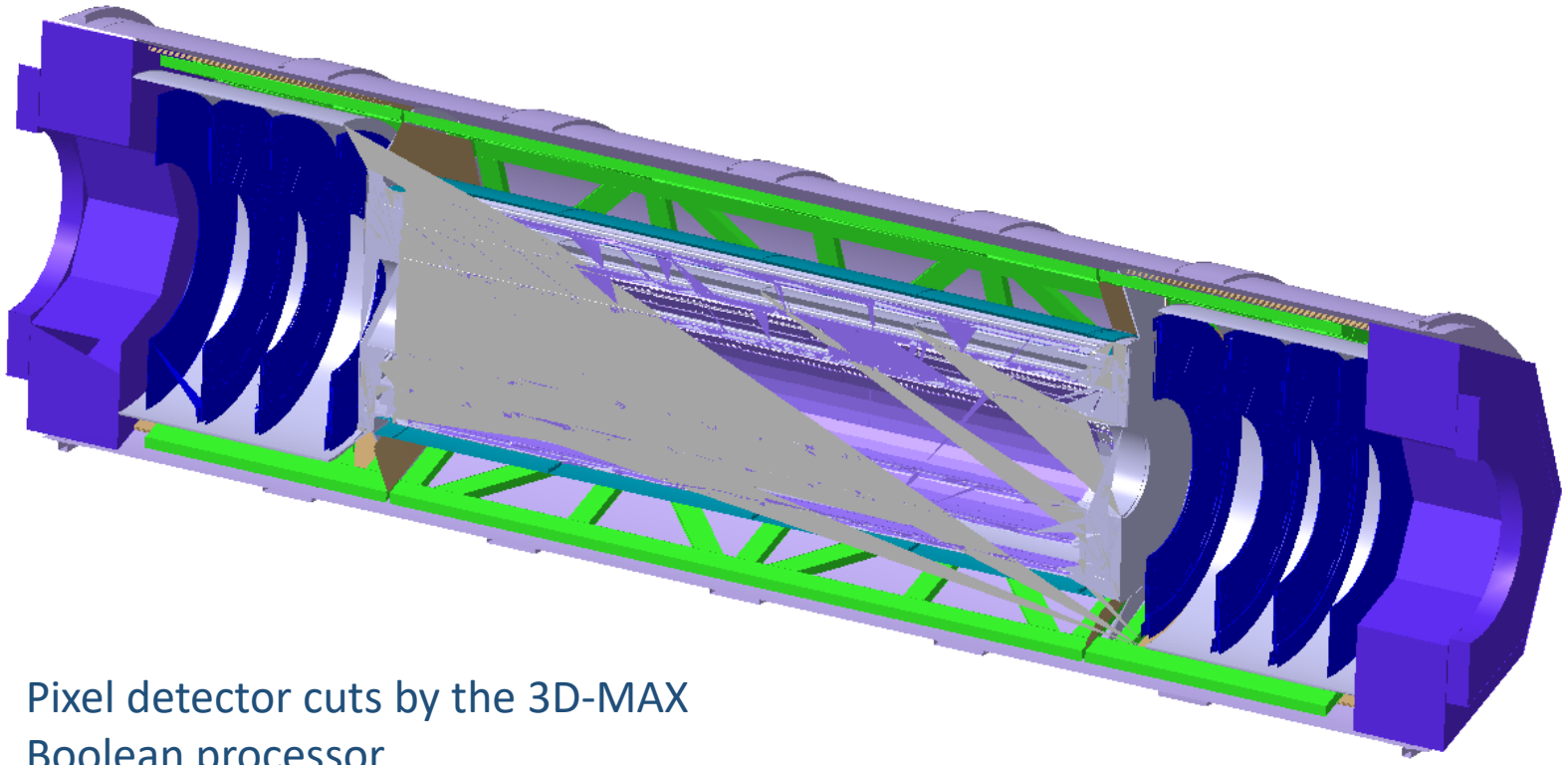
- For facet-based models, method identifies all triangles touching with cutting plane -> delete them and build the new triangles





III. Geometry Cuts

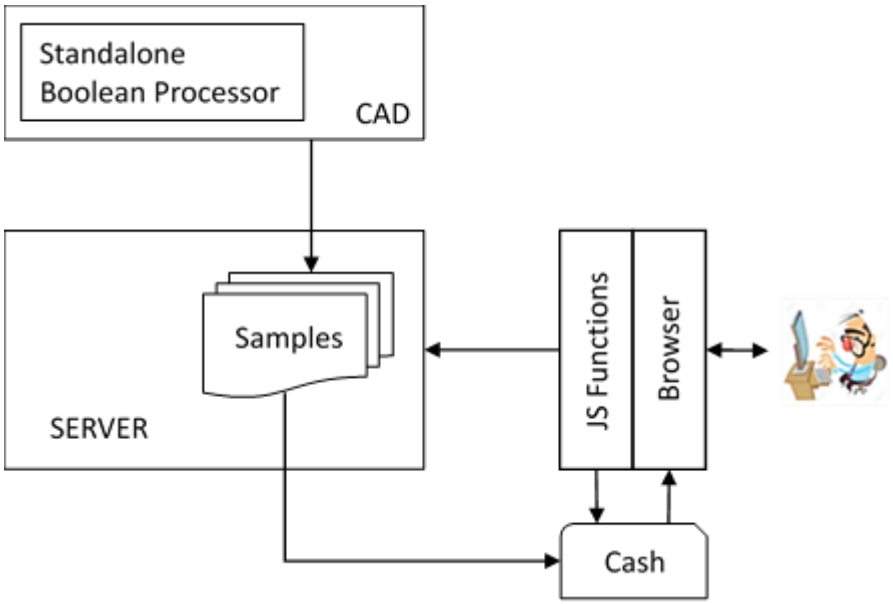
- However, for the complex facet-based geometries, like detector geometries, those methods doesn't works. For the Pixel detector cuts, application was crashed after the 23 minutes of calculations



Pixel detector cuts by the 3D-MAX Boolean processor

III. Geometry Cuts

- Tracer/Core use external Boolean processors and cutting samples for geometry cuts
- Therefore, Tracer/Core ensure high quality geometry cuts in 3D scene by downloading the cut samples instead of activation of three.js internal Boolean processors in real-time



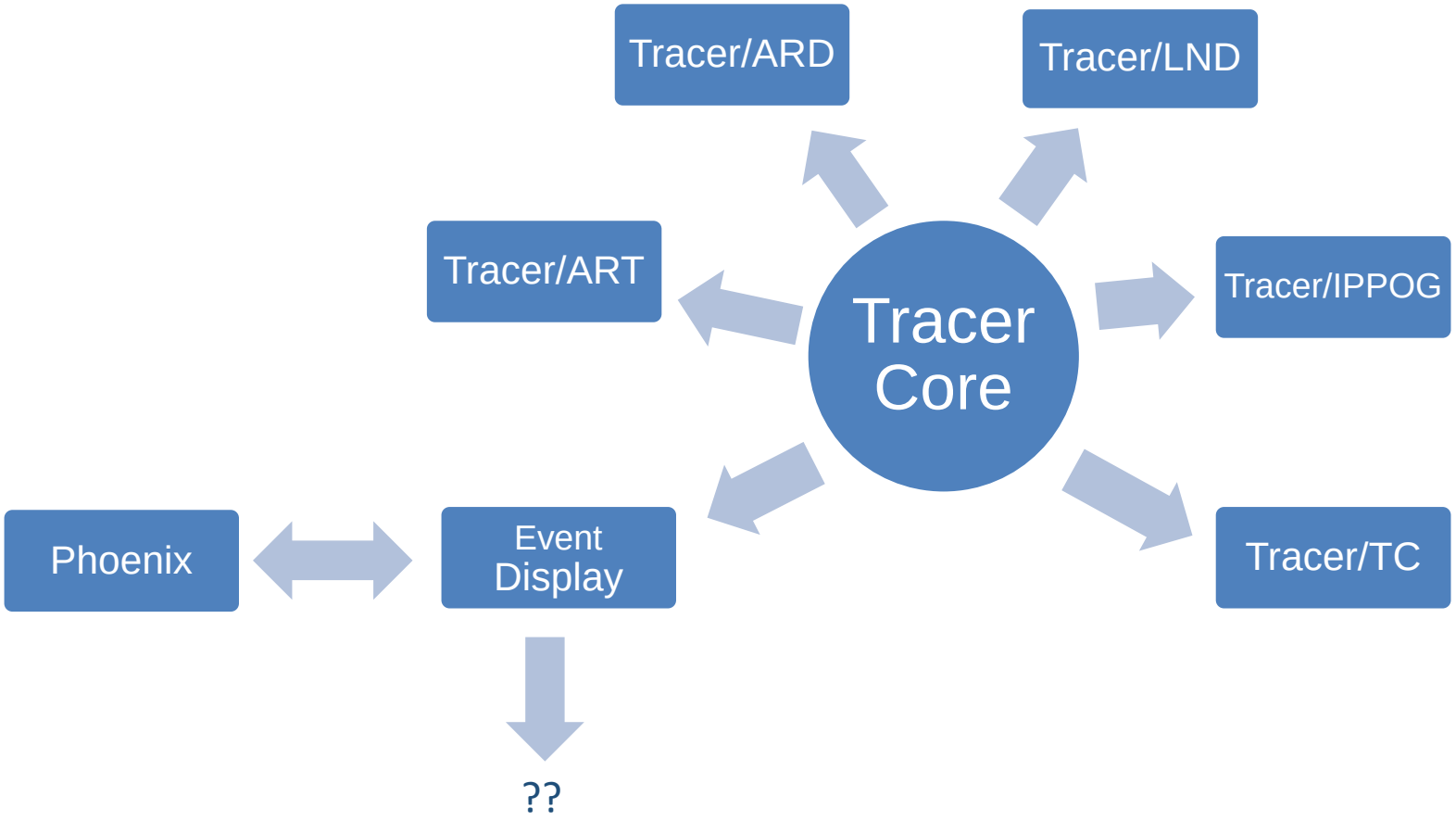
IV. User Framework

- Tracer/Core has a responsive web interface
- Interactive project tree ensure representation of full anatomy of the detector and shows the topology of a current 3D scene
- It is possible to reach objects properties through the tree and control separately the visualization modes, like transparency, geometry cuts, brightness, etc.
- CSS3/html5 functions ensure adaptation to different screen sizes and resolutions
- Tracer/Core has friendly mouse control functionality

III. Upcoming Development

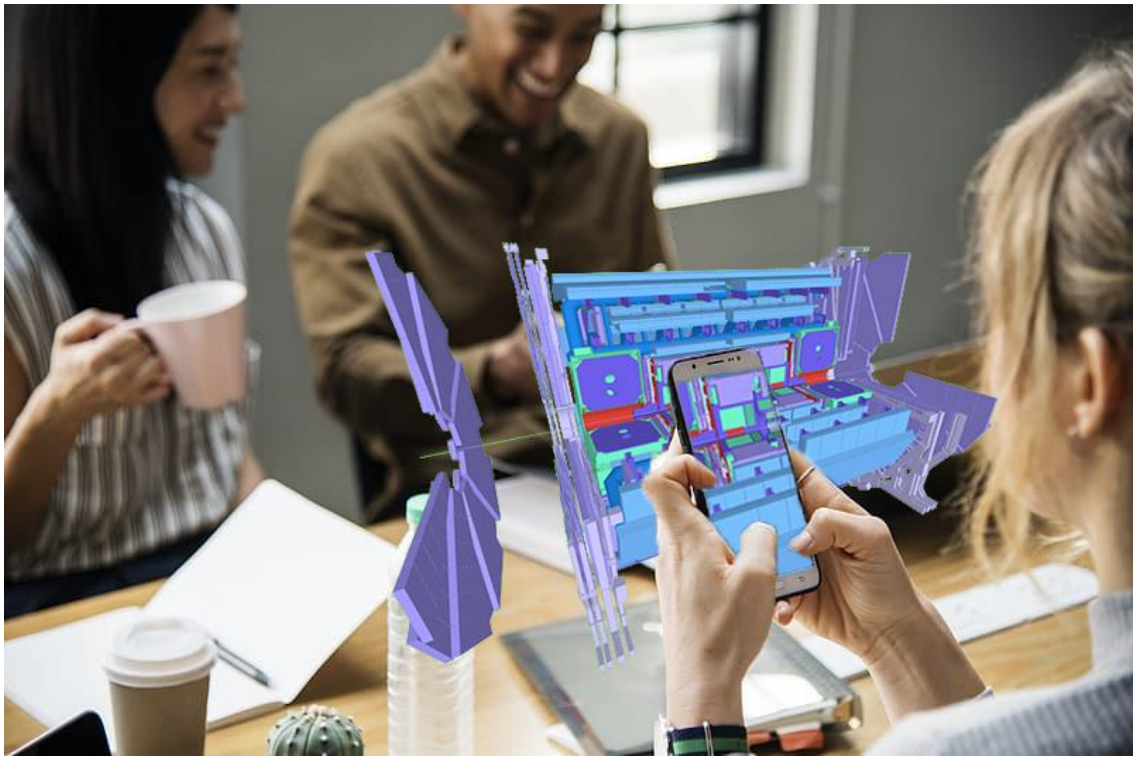
III. Upcoming Development

- We are developing several packages on the base of Tracer/Core



Tracer/ART

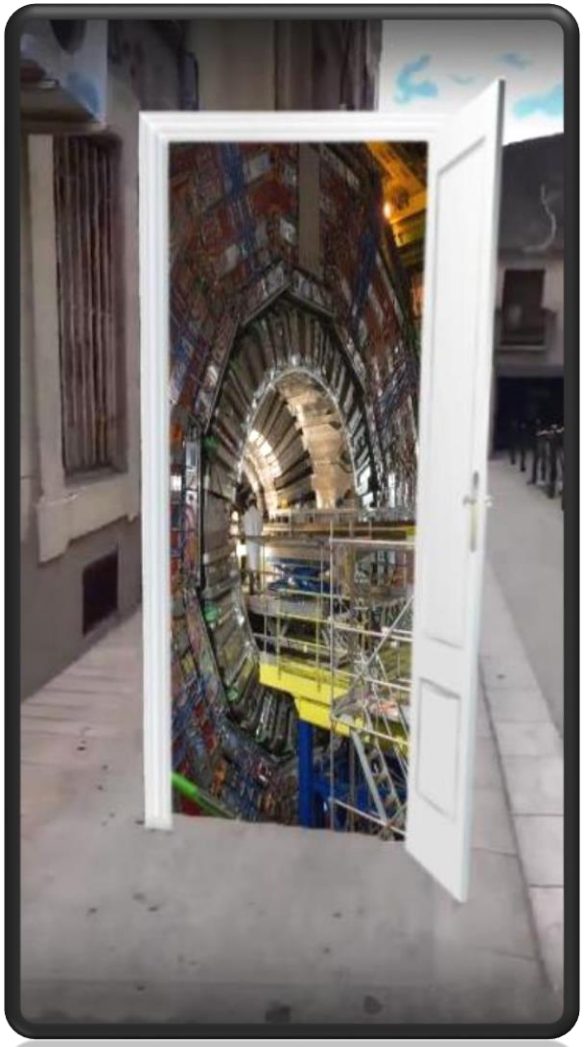
- Tracer/ART (Augmented Reality Table) enables to ‘put’ detector on the discussion table and make the interaction with components



- 1st draft of Tracer/ART already developed

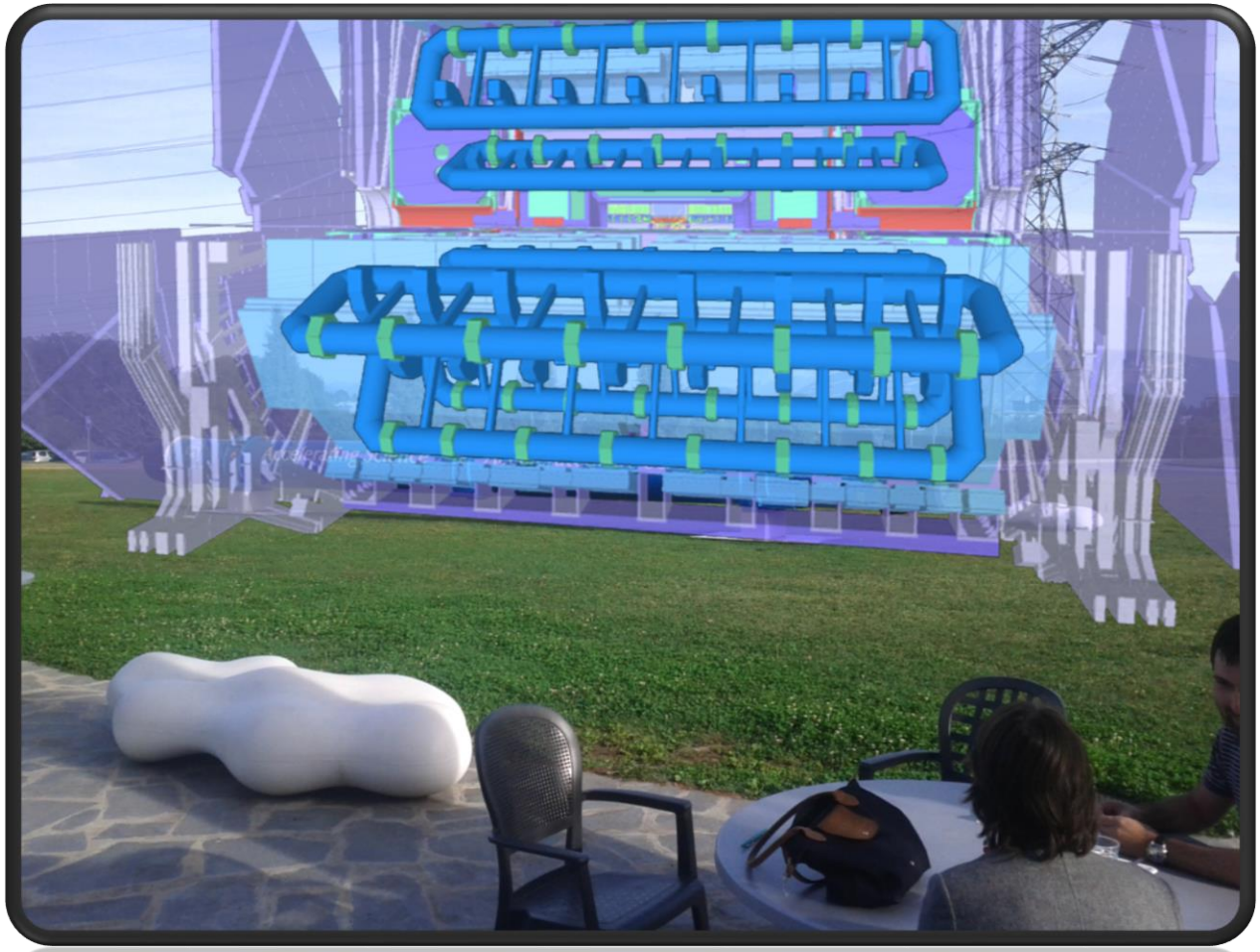
Tracer/ARD

- Tracer/ARD (Augmented Reality Door) enables to navigate inside the detector through the virtual doors
- It is possible to pass through the door and go in a virtual room, for instance in atlas cavern and walk there
- Also possible to ‘put’ another door in virtual room and pass through there
- Back door is always visible in virtual room. So, we can come back to reality



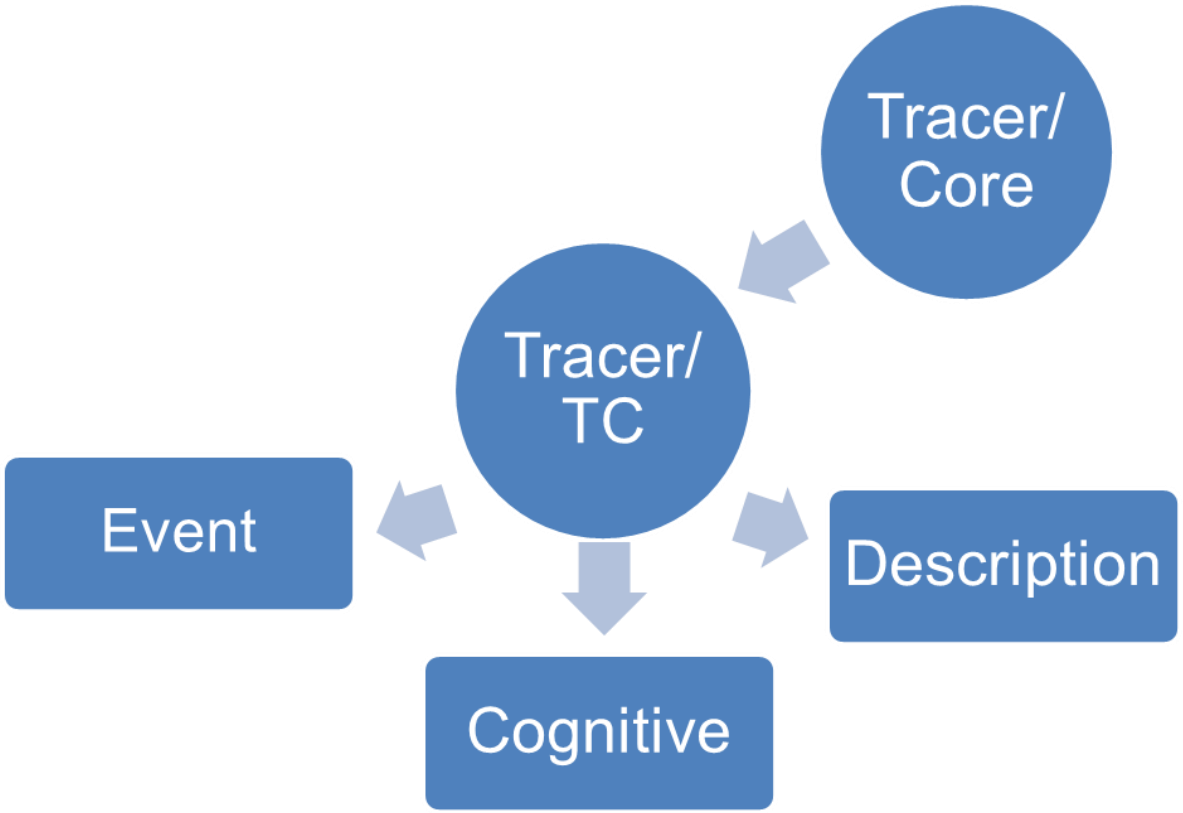
Tracer/LND

- Tracer/LND (Landscape) to put full detector with natural or scaled size on any landscape



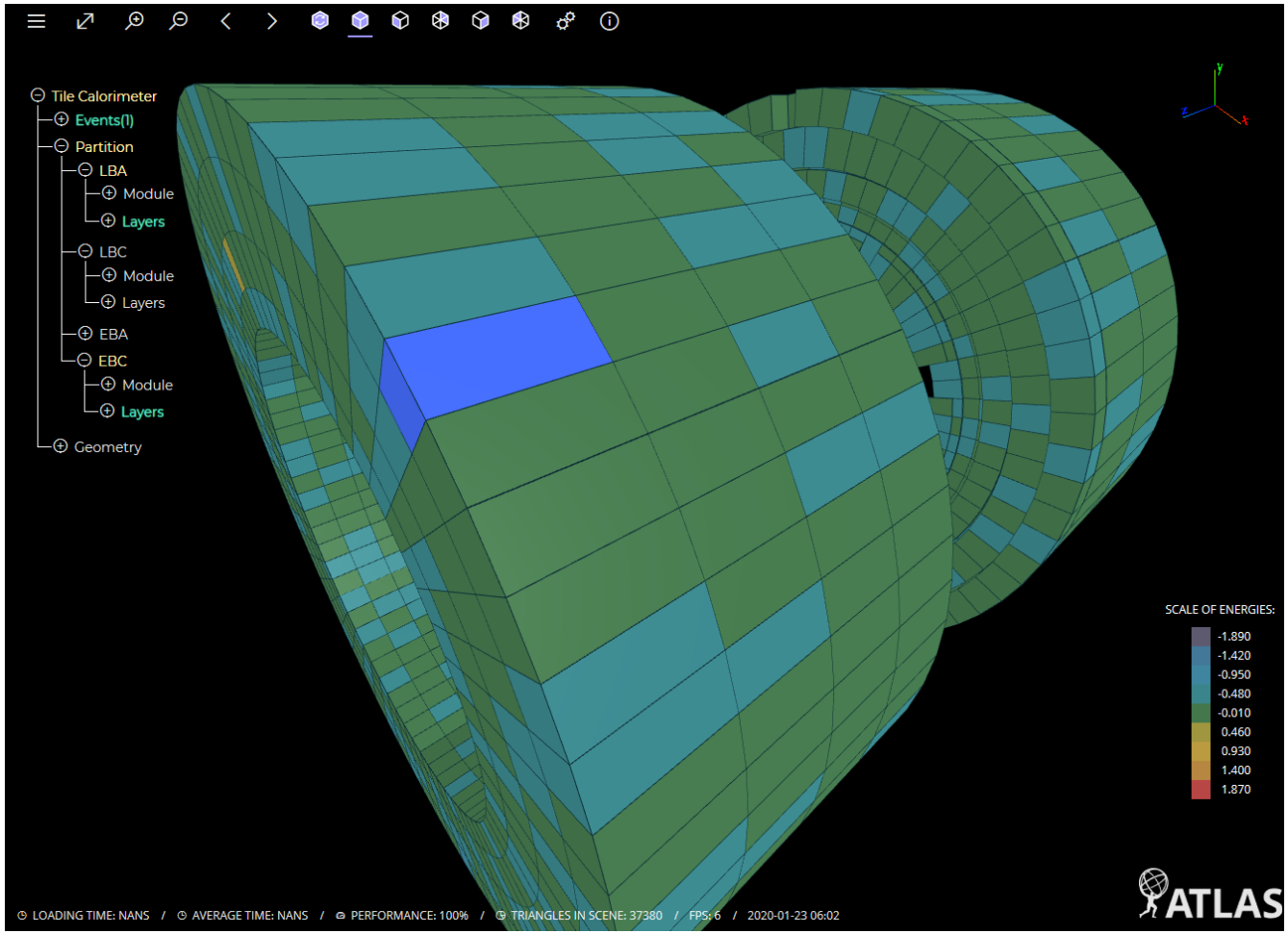
Tracer/TC

- We are working together with ATLAS TileCal team – contact persons Oleg Solovyanov, Alexander Solodkov



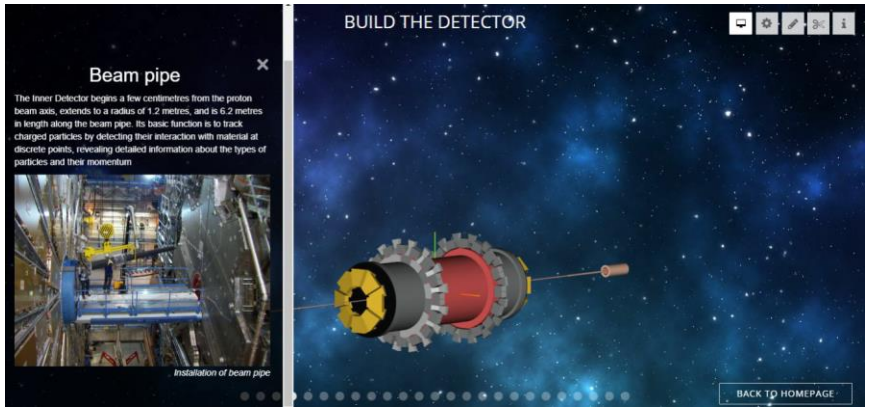
Tracer/TC

- R1.2 - first release already developed. It is here http://cadcamge.ch/tc/r1.1/#



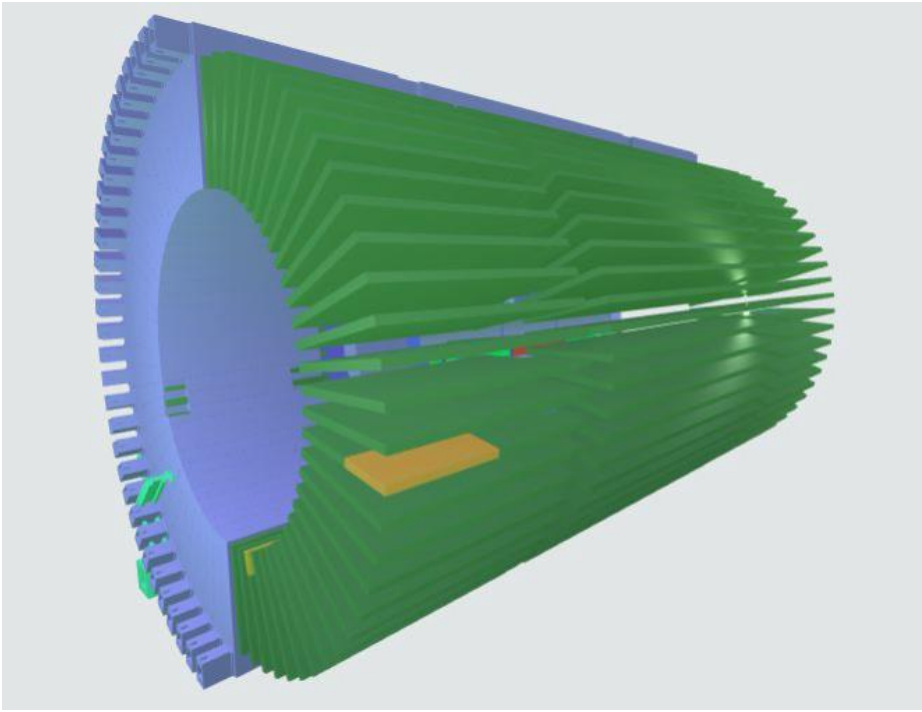
Tracer/TileCal Technical Specification

- **Tracks:** Improve precision of track drawing algorithm, visualize hits, display muon tracks, detect TilCal cells, which this muons went through, sum energies of muon cells, calculate muon isolation
- **Cognitive:** display educational videos explaining how different parts of TileCal works, installation videos, photos, TileCal geometries, links to the pages that give more information about TileCal; TileCal virtual reality application, including: fly mode, controls (hands on experience)



Tracer/TC

- **Technical Description:** Development of TileCal technical description, including electronics, supports, RoD etc. Development of detailed geometric descriptions of TileCal; Display "dead" cells from DB



Tracer/IPPOG

- Tracer/IPPOG is 3D detector display application for IPPOG masterclasses
- We have high interest from IPPOG community
- IPPOG did 6 sessions of masterclasses in Georgia with participation of more then 500 teenagers
- Tracer/IPPOG was successfully used for the events in 2018, 2019

References on Tracer

- Draft of application was presented on HEP Software Foundation Workshop at San Diego 23-26 January, 2017
  http://indico.cern.ch/event/570249/
- Final version of tracer R1 was presented on HSF Visualization Workshop at CERN 28-30 March, 2017
  https://indico.cern.ch/event/617054/
- Trace was recognized as an efficient tool for ATLAS Outreach & Education https://cds.cern.ch/record/2243749/

Acknowledgements

to Tracer team:

- Lasha PHATARIDZE – Concept, Events programmer
- Levan KHELASHVILI – WebGL engine programmer
- Nikoloz UDZILAURI – UI, Web designer programmer
- Shota KOBAKHIDZE – Geometry responsible
- Mariam PIRTSKHALAVA – Augmented Reality

to Physics Consultants:

- Felix SOCHER
- Giorgi ARABIDZE

Comments are welcome,
Thanks

Lasha.Sharmazanashvili@cern.ch
www.cadcam.ge