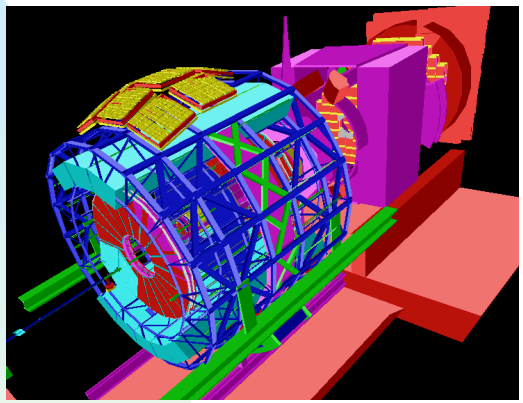
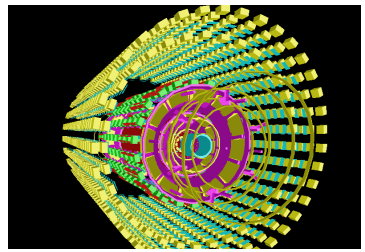
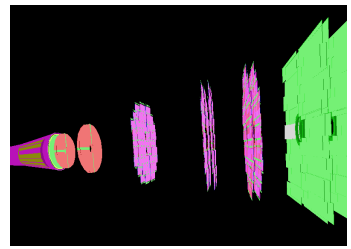
The new ROOT geometry package

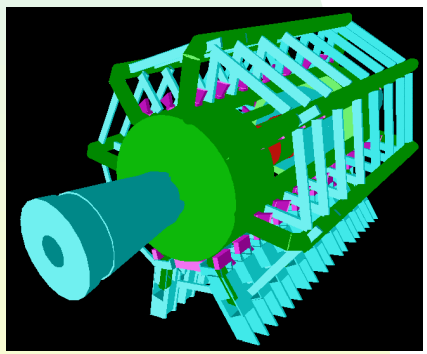
Andrei Gheata - ALICE

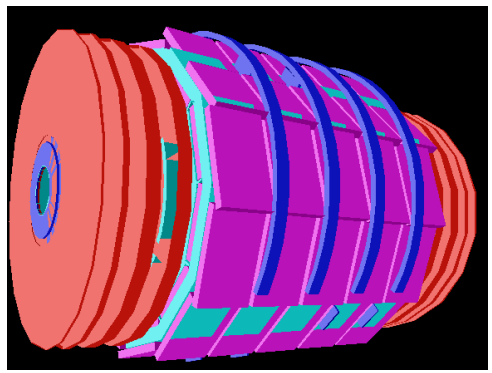
Institute of Space Sciences, Bucharest

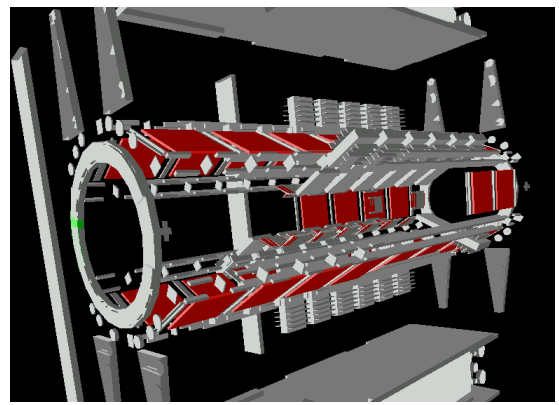


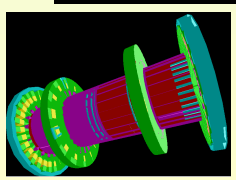


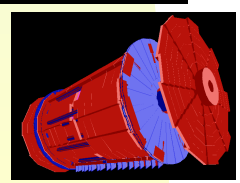


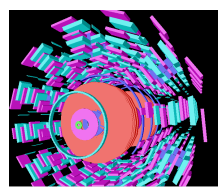




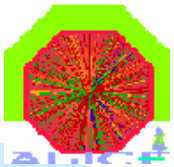








Jump to first page



Topics

- Mission statement
- Architecture description
- Current status
- Tracking optimizations
- Benchmarks
- Latest developments
- Next steps
  - Demo (10')

Mission statement

Modeling

Visualisation

Interactivity

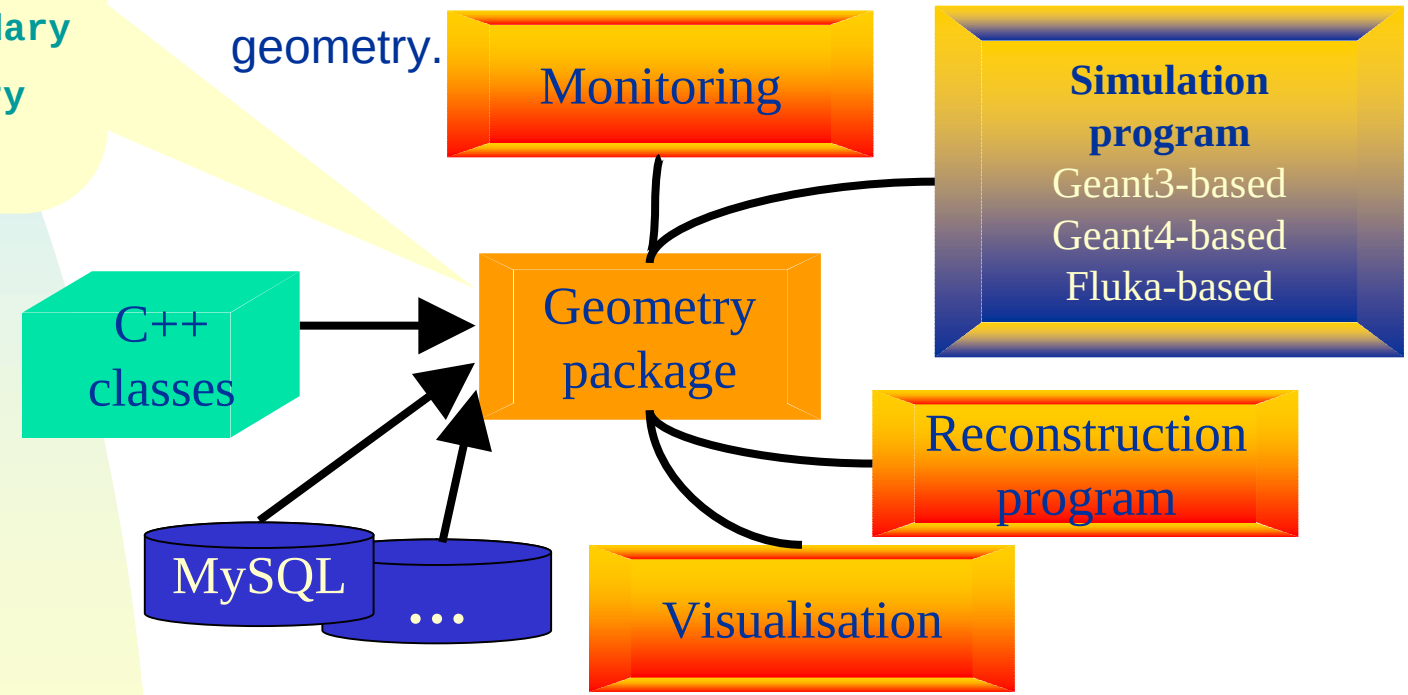
Where am I?

Distance to boundary

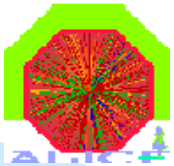
Closest boundary

Persistency

- The ROOT geometrical package is intended as a toolkit to provide the geometrical description of an experiment, full tracking functionality + additional tools to ease-up building, checking and debugging a geometry.



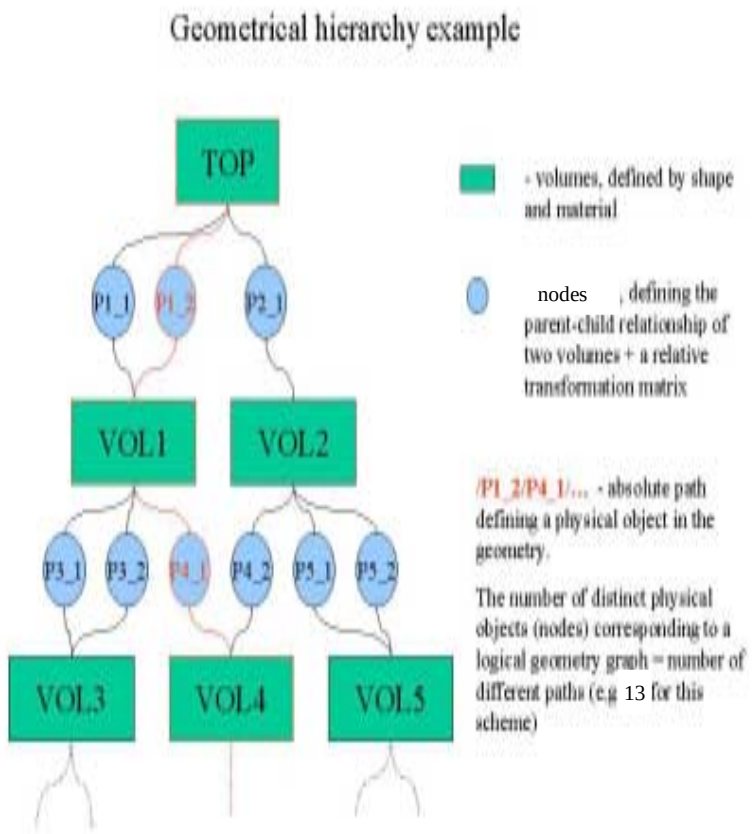
- Started ~1 year ago as a common ALICE/ROOT effort having in mind the idea to run several MC's with the same user code.



Requirements

- Tracking-related functionality - ("*Where am I*?", "*Where next and how far*?", …)
- Backward compatibility - to be able to reuse Geant3 geometries
- Scalability - LHC geometries are big, we have to be able to represent them
- Performance - it should be faster than existing modelers
- Interactivity - users should be able to access easily all relevant geometry information + checking tools, builder interface, ...

Architecture description



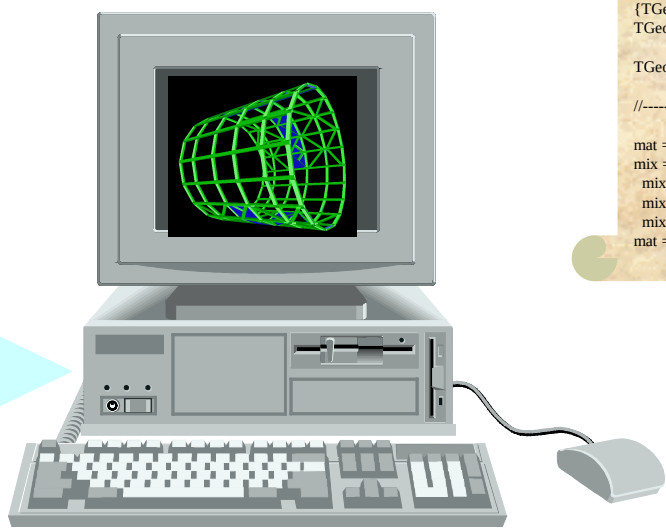
- G3-like : CSG with constrains imposed by container-containee concept
- Supporting an extended set of primitive shapes
- Supporting full CSG architecture at shape level (composite boolean shapes)
- Partially expanding full hierarchy in memory (for performance reasons) - using a cache mechanism and matrix compression.

Automatic conversion from Geant3

Zebra memory
Data structure
JVOLUM
JMATE,etc

Root > .x mygeom.C

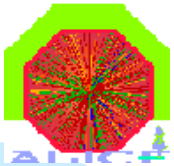
```
void gexam1()
{TGeoMaterial *mat;
TGeoMixture  *mix;

TGeoManager *gexam1 = new TGeoManager("gexam1","gexam1.C");

//----------List of Materials and Mixtures----------

mat = new TGeoMaterial("mat9","ALUMINIUM",26.98,13,2.7);
mix = new TGeoMixture("mix10","IRON(COMPOUND)",3);
  mix->DefineElement(0,55.847,26,0.703964);
  mix->DefineElement(1,58.71,28,0.9900000E-01);
  mix->DefineElement(2,51.998,24,0.197);
mat = new TGeoMaterial("mat11","COPPER",63.54,29,8.96);
```



Geant > RZ/File 21
mygeom.geom on

Zebra RZ
mygeom.geom

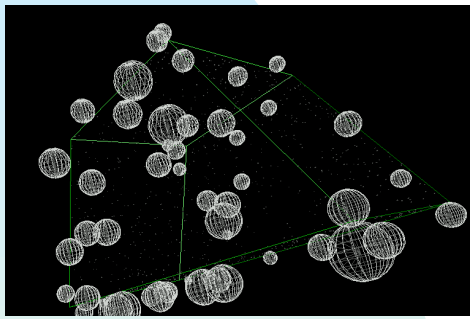
g2root mygeom.geom
mygeom.C

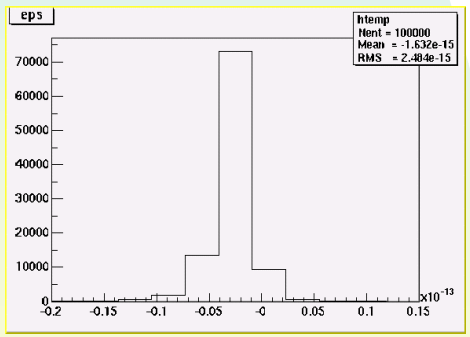


Current status

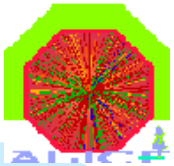
- The code is inside ROOT since v3-03-08
- Html classes, description, quick start guide at : http://root.cern.ch/root/htmldoc/TGeoManager.html
- We can map practically any geometry described by Geant3 + more flexibility due to composite shapes
- Tracking functionality mostly implemented : still work to be done in computing safety and normals for some shapes
- Several benchmarks were done with respect to Geant3 – better performance
- Composite shapes implemented – still work to be done for visualization
- Several ongoing tests to provide stability of the code

Tracking optimizations



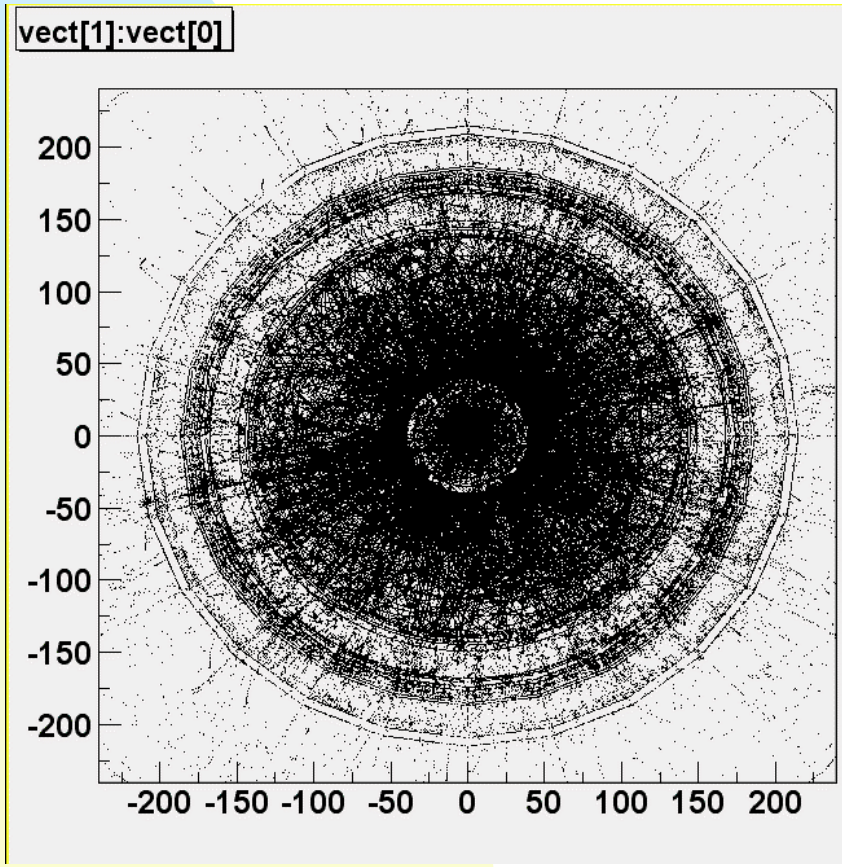


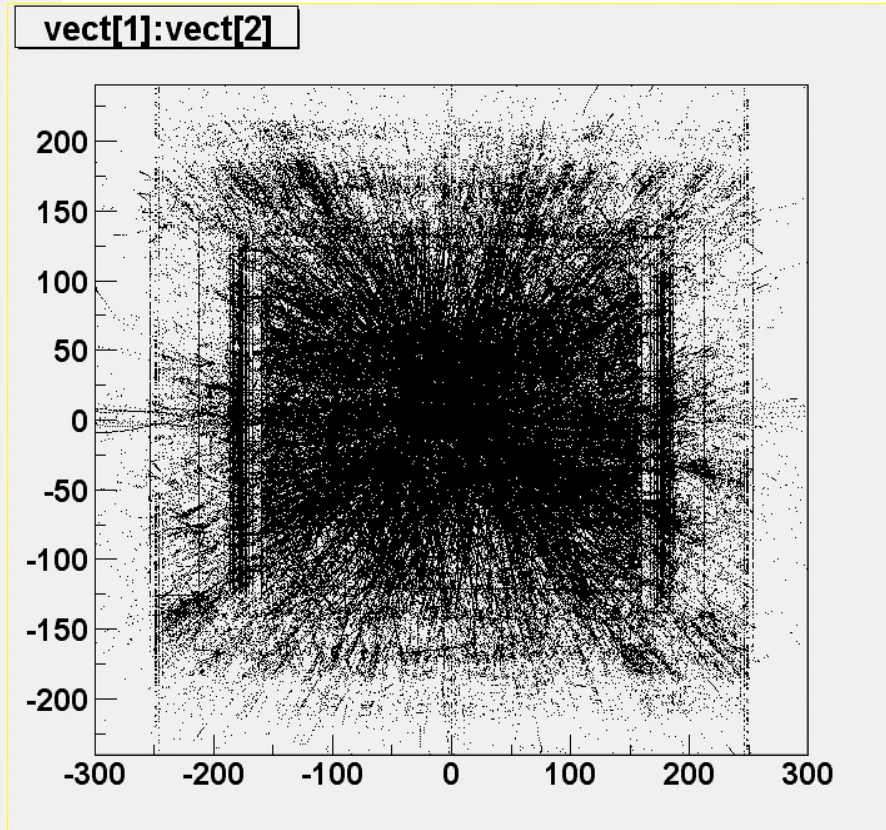
- Several optimizations have been done to improve performance :
  - Bounding boxes for all shapes
  - Voxelization used to minimize the number of candidates to be checked (voxels = *volumetric pixels*) – both cartesian and cylindrical coordinates
  - *Safety* computed for most shapes
  - Physical geometry tree partially expanded with a cache mechanism
  - "Smart" transformations – we make the difference between translations, rotations or general transformations for instance for computing point and vector transformations
- Boundary diffusion due to floating point precision have been analyzed for some shapes : ~10^{-12}
- Introduced several tracking flags, to control : boundary crossing, entering/exiting

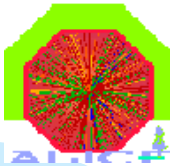


Validation procedure

Use one million points generated by Geant3 with full physics switched on

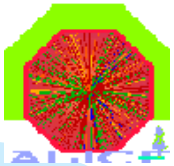






Validation procedure

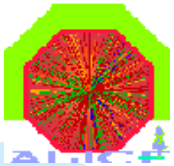
- Using points generated by Geant3 applications
  - grecord.f --> mygeom.geom, mygeom.hbook
- Play-back these points in Geant3 using only the Geant3 geometry package. Compute g3path, snext, safety in myresults.hbook
- h2root myresults.hbook myresults.root
- Same operation with TGeo classes. Compare g3path with tgeopath, same for snext, safety
- Some discrepancies with Geant3 :
  - Due to precision problems in Geant3 (single precision geometries collected), points recorded at the volume boundaries may be found on the other side by TGeo.
  - When volumes declared MANY in Geant3 overlap and have sub-volumes also MANY, Geant3 is not always reporting the right answer. In general it does not matter, sometimes it does
  - In this exercise, we also found original errors in the detector description, e.g. wrong paramers



ROOT modeler benchmarks vs. GEANT3 for "Where am I" task

	Number nodes	gtmedi physics	Root physics	Geant3/ Root	gtmedi random	Root random	Geant3/ Root
Gexam1	425	3.08	1.84	1.67	6.60	4.39	1.50
Gexam3	86	2.87	2.15	1.33	3.47	2.50	1.38
Gexam4	12781	2.51	2.20	1.14	12.09	11.18	1.08
Brahms	2649	5.82	3.04	1.91	4.17	1.93	2.16
Tesla	15370	6.56	5.58	1.17	12.95	7.15	1.81
CDF	24422	14.81	4.31	3.43	20.94	5.85	3.57
Minos_near	30988	30.93	20.99	1.47	21.57	13.70	1.57
BTeVEcal	52	1.57	1.08	1.45	1.78	0.73	2.43
BTeV	295310	45.27	25.88	1.75	197.06	26.83	7.34
CMSEcal	251713	5.60	1.81	3.09	5.69	1.74	3.27
CMS	1166310	33.57	8.76	3.83	39.09	24.98	1.56
LHCb	1533488	7.98	6.75	1.18	12.58	2.89	4.35
Alice	3080198	11.50	8.63	1.33	11.45	7.28	1.57
Atlas	29046966	8.90	9.94	0.89	32.48	23.39	1.38

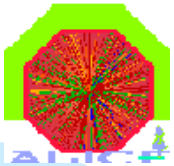
- 1 mil. Points extracted from G3 simulation (full physics)
- Point re-injected both in G3 and ROOT modeler, timings/track [microseconds] presented in the table.



I/O - file size

	Number nodes	RZ file (kBytes)	text file(.C) (kBytes)	root file (kBytes)
Alicenew	3080198	->mem	2714	3070
Atlas	29046966	9863	5872	3554
Brahms	2649	688	149	101
CDF	24422	2412	1350	1252
CMS	1166310	13296	1930	1284
LHCb	1533488	2121	1053	1215

- The size of the .root file and time to read are very important when developing a geometry.
- .root files includes full voxelization



Composite shapes

- Union (+), intersection (*) and subtraction (-) operations fully supported at shape level.
- Composite shapes are defined like :

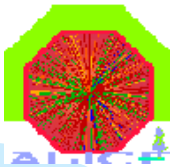
TGeoCompositeShape(const char *name, const char *expression)

- Expressions are made by identifiers (shape_name:transf_name), e.g:

  (tube+tube:rot1)*box:t1 which means: "*union of a tube with the same tube rotated, then intersect the result with a translated box*"

- Internally a *CSG binary tree* is built : any query is posted at top level, while answers are collected from final leaves upwards

(see demo)



Next steps

- The main functionality is there, but it has to be further tested and improved
- Checking tools to be developed (we already have good ideas how)
- Concept of *module* to be introduced : this should be the link to any *detector model*
- Visualization for composite shapes to be done : this might take a while (polyedron vs. polyedron algorithms for boolean operations missing)
- G3 tracking based on the new modeler need to be implemented (as a first stage)
- The user interface can always be discussed and improved