National Research Tomsk State University

Investigations in the ATLAS geometry

Evgueni Tcherniaev

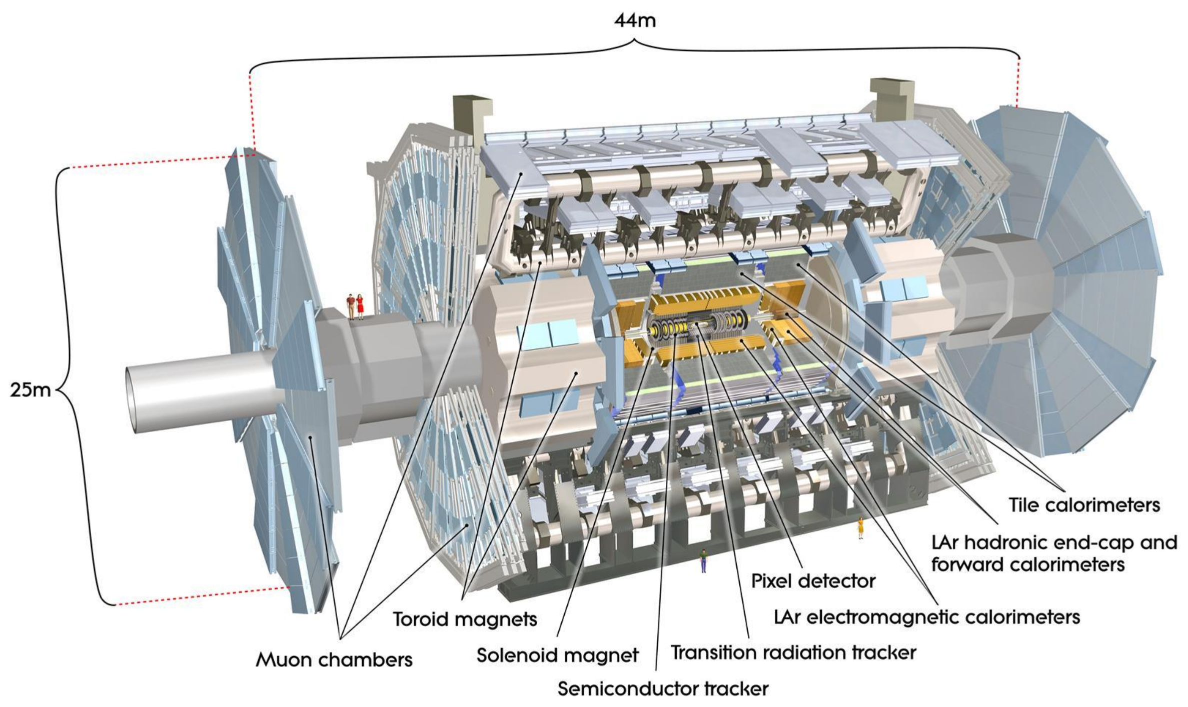
Geant4 development team

Outline

- Introduction
- Side effect of some GeoModel features
- Statistics of Geant4 solid methods
- Usage of Boolean solids
- Usage of G4Polycone solids
- CPU consumption by Solids & Navigator
- Miscellaneous
- Summary

Introduction

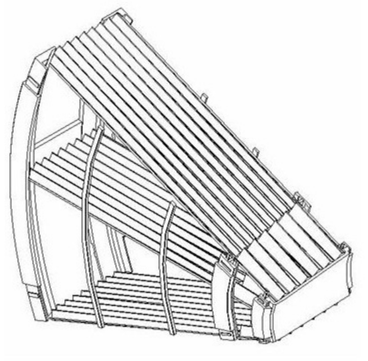
Analysis of the ATLAS geometry

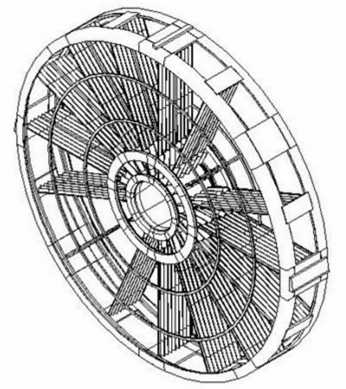

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

- The purpose of the analysis was to gather *quantitative* information on usage of Geant4 solids in the simulation of very complex detector.
- Gathered information was used:
  - To enhance of Geant4/Geometry code by taking into account the specifics of the shapes used in the detector description
  - To spot places, where the ATLAS geometry description can be optimized
- It also has improved the ability to predict the impact on the performance of various modifications

Specifics of the ATLAS geometry

- ATLAS geometry description is very complex
  - It contains several millions of physical volumes
- ATLAS geometry description is very detailed, for example:
  - 7756 G4Box solids have volume < 1 cm^3, of those 472 have volume < 1 mm^3
  - 798 G4Tubs solids have volume < 1 cm^3, of those 123 have volume < 1 mm^3
- There is a custom solid used for navigation inside EMEC (Electromagnetic End-cup Calorimeter). Simulation inside EMEC takes >35% of CPU.





- ATLAS geometry description is based on GeoModel:
  - GeoModel is a library of C++ classes that can be used to describe detector geometries
  - GeoModel often means also a structure of geometrical primitives that, in principle, can be transferred into the structures of various simulation packages
  - To transfer the GeoModel structure into the Geant4 structure a tool Geo2G4 is used

Side effect of some GeoModel features

- Issue with TRT mother volume
- Fake overlaps

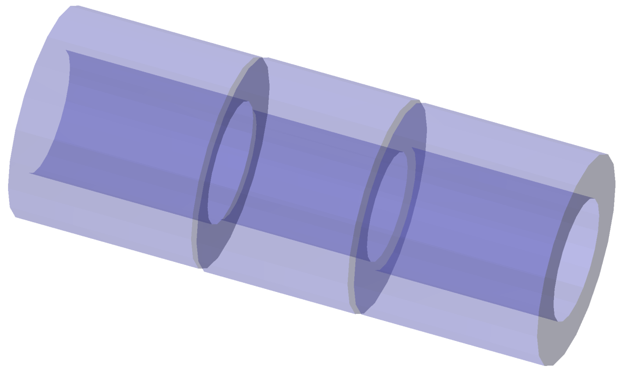
Analysis of ATLAS GDML dump

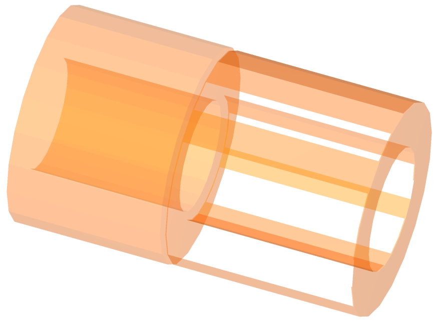
- Geant4 has a number of useful tests which work with GDML files: overlap check, Geantino test, real simulation test
- When running those tests with the GDML dump, provided by the ATLAS simulation group, we have faced a couple of unexpected issues:
  - Original GDML dump did not loaded:

```
PANIC! - Overlapping daughter with mother volume.
         Daughter physical volume TRT
         is entirely outside mother logical volume TRT::TRT !!
```

  - After a manual fix of the GDML file, the overlap check has discovered several rather big, suspicious overlaps

Issue with TRT mother volume in GDML





- The mother volume of TRT (Transition Radiation Tracker) is described as a union of its daughter volumes
- In the original GDML file both end cups were placed on the same side relative to the central barrel (problems with rendering are caused by that)
- A bit of history: the issue with TRT in the GDML dump has been discovered in November 2014. It was considered as a bug in the GDML writer. Indeed, a bug in the GDML writer has been found and fixed. However...

  The same issue has been re-discovered three years later, in October 2017 – the bug fixed in 2014 was not the reason of the issue with TRT
- Finally, during ATLAS Hackathon in February 2018 the issue with TRT has been understood and fixed

Use of G4DisplacedSolid objects

- The reason of the issue with TRT was a construction of a G4DisplacedSolid object from another G4DisplacedSolid object.
- Although G4DisplacedSolid is meant to be used as an internal class for Boolean solids, it has all methods required for the navigation and is used in ATLAS/GeoModel as an ordinary solid.
- To fix the issue, the constructors of G4DisplacedSolid have been enhanced to handle the case of nested displaced objects.
- The modification has also improved the performance of Boolean operations in such a case.

Overlap check

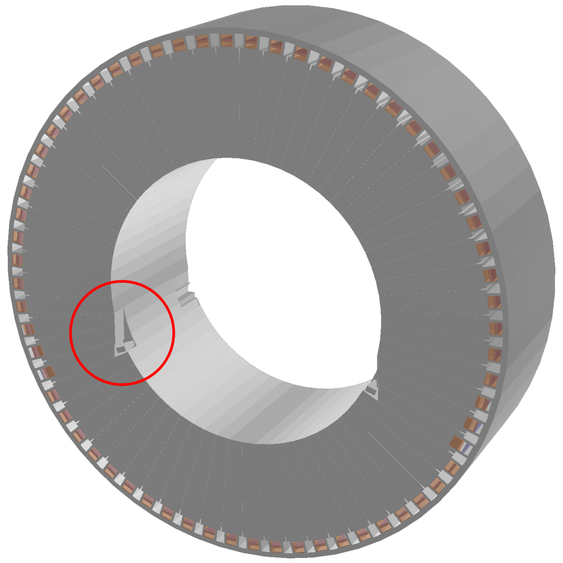
- The existence of overlaps in the detector geometry description means that there are some issues in the geometry definition that may lead to incorrect result of the simulation
- Geant4 tracking algorithms are quite sensitive to overlaps, which can be the cause of various issues during tracking, for example skipping of material (and eventually sensitive material)
- Below is the result of overlap check in the GDML dumps:

  Inner Detector – 1983 overlaps

  Calorimeters – 172 overlaps

  Muon System – 332 overlaps

Propagation of Boolean subtraction to daughter volumes. Fake overlaps



- A number of fake overlaps (20-40 cm) have been observed in the end cups of the Tile calorimeter
- The fake overlaps have appeared as a result of automatic propagation by GeoModel of Boolean subtraction, applied to a sector of the end cup (see the image), to its daughter volumes
- If the result of a Boolean operation (subtraction or intersection) is a null object, the Geant4 overlap check may report an overlap. In such a case the warning message on overlap will be preceded by a warning message on a failure to pick up a random point on the surface of the object:

```
All attempts to generate a point on the surface have failed!
The solid created may be an invalid Boolean construct!
```

- Null objects do not create problems for tracking, however they consume execution time

Statistics of Geant4 solid methods

Distribution of calls by Solids and Methods

Number of calls for simulation of 10 ttbar events, Geant4 10.01.patch03.atlas05

	Inside x 10^6	**Normal** x 10^6	**SafetyToIn** x 10^6	**SafetyToOut** x 10^6	**DistanceToIn** x 10^6	**DistanceToOut** x 10^6	**TOTAL** x 10^6
G4Box	**421**	1	19	5	9	4	**459**
G4Trap	**978**	1	111	42	38	23	**1193**
G4Polycone	336	1	21	120	9	41	**528**
G4Tubs	172	9	168	124	66	46	**585**
G4Trd	34		16	14	6	9	**79**
G4Extruded	3		3				**6**
G4Union	**1287**	3	23	6	14	22	**1355**
G4Subtraction	38		13	1	2		**54**
G4Displaced	**1385**	1	34	4	19	5	**1448**

- It took quite a bit of time to understand why there is so many calls to `Inside()` for G4Union, G4DisplacedSolid, G4Box and G4Trap
- The reason was just in three identical Boolean solids that had 42 components

Usage of Boolean solids

- Bad practices in usage of Boolean solids
- CPU consumption by TMT

Booleans solids in ATLAS geometry

- Boolean solids, except G4MultiUnion, do not have internal optimization. The time required for calculations in Boolean solids scales at best with the number of components, often worse than this because of rather complicated logic of the calculations.
- Below is a list of Boolean volumes in different parts of the Atlas detector:
  - Inner Detector – 465 Boolean volumes
    3 volumes are composed of 42 components
    others are composed of 2 – 11 components
  - Tile Calorimeter – 15863 Boolean volumes
    Almost all Boolean solids can be replaced with Extruded Solids
  - Muon System – 2869 Boolean volumes
    >170 volumes have 30 - 82 components

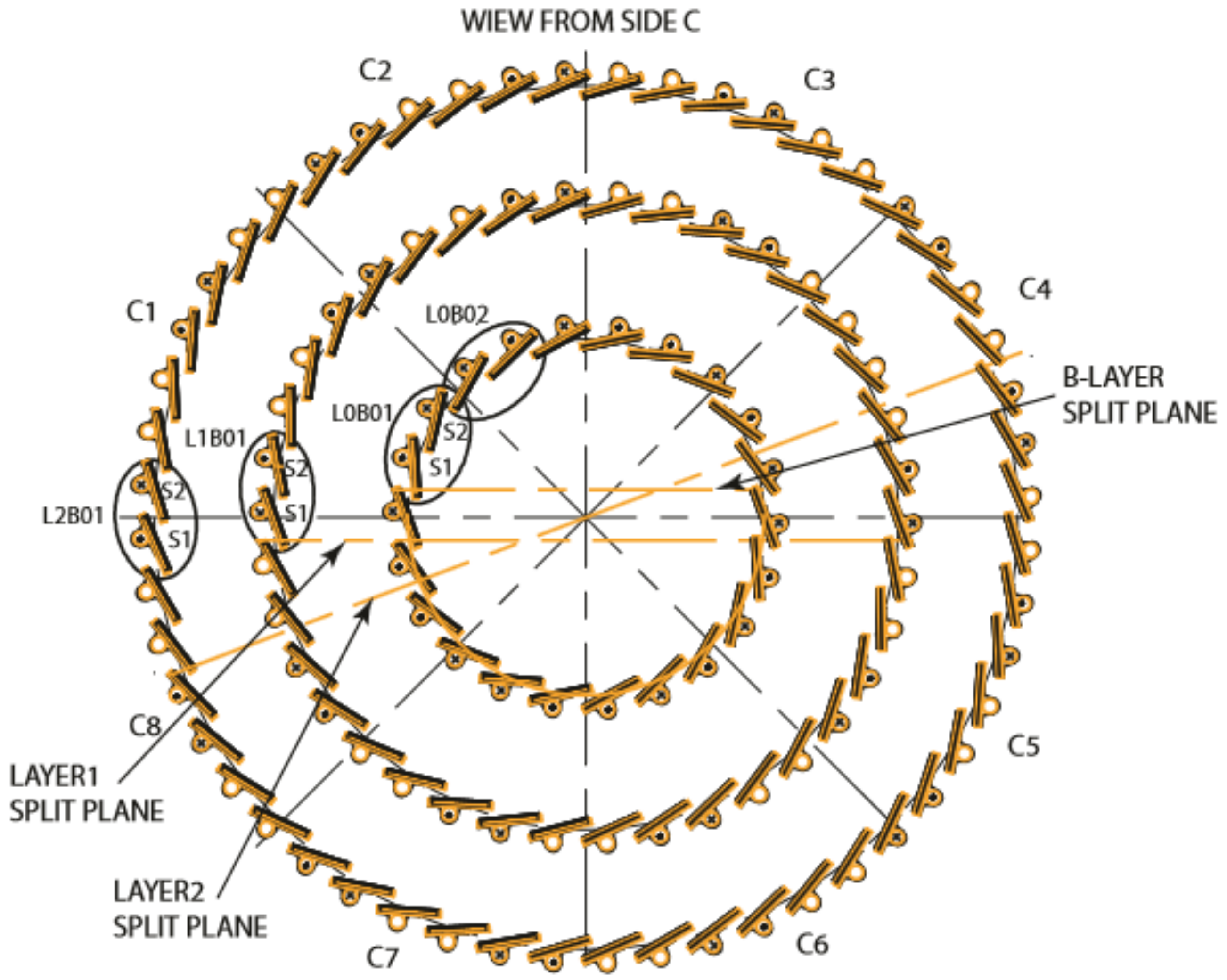
Typical “bad practices” in using Boolean solids

Below is a list of typical usage of Boolean solids that can degrade the performance:

1) Usage of Boolean subtraction to define holes
   A better way to define holes is to define them as daughter volumes
2) Describing a mother volume as a union of its daughter volumes
   A better way to group volumes is to use G4AssemblyVolume
3) Usage of a Boolean solid with complex shape for very low level volume
   A better way is to split the solid into simple parts and, if required, group the parts using G4AssemblyVolume

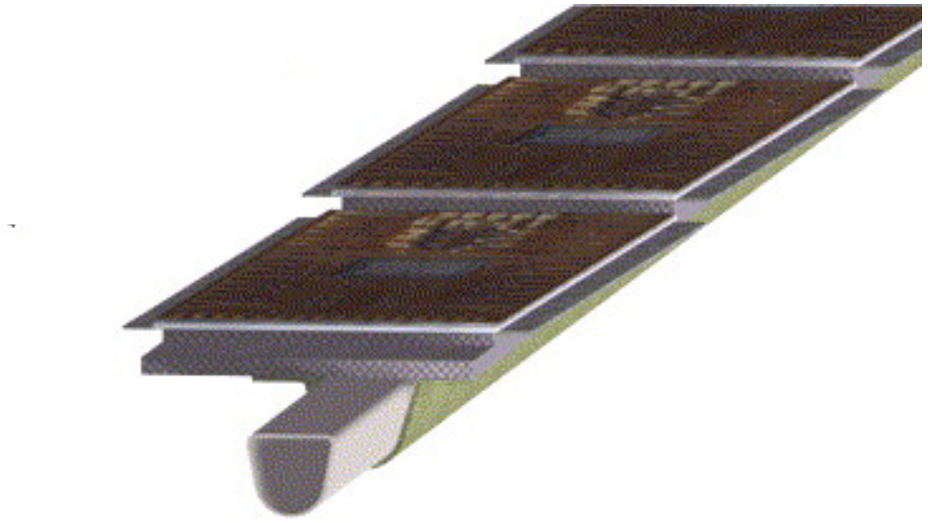
A solid defined for Thermal Management Tile (TMT) is an extreme example of the case 3) – the TMT solid has been constructed as a union of 6 G4Box solids and 36 G4Trap solids

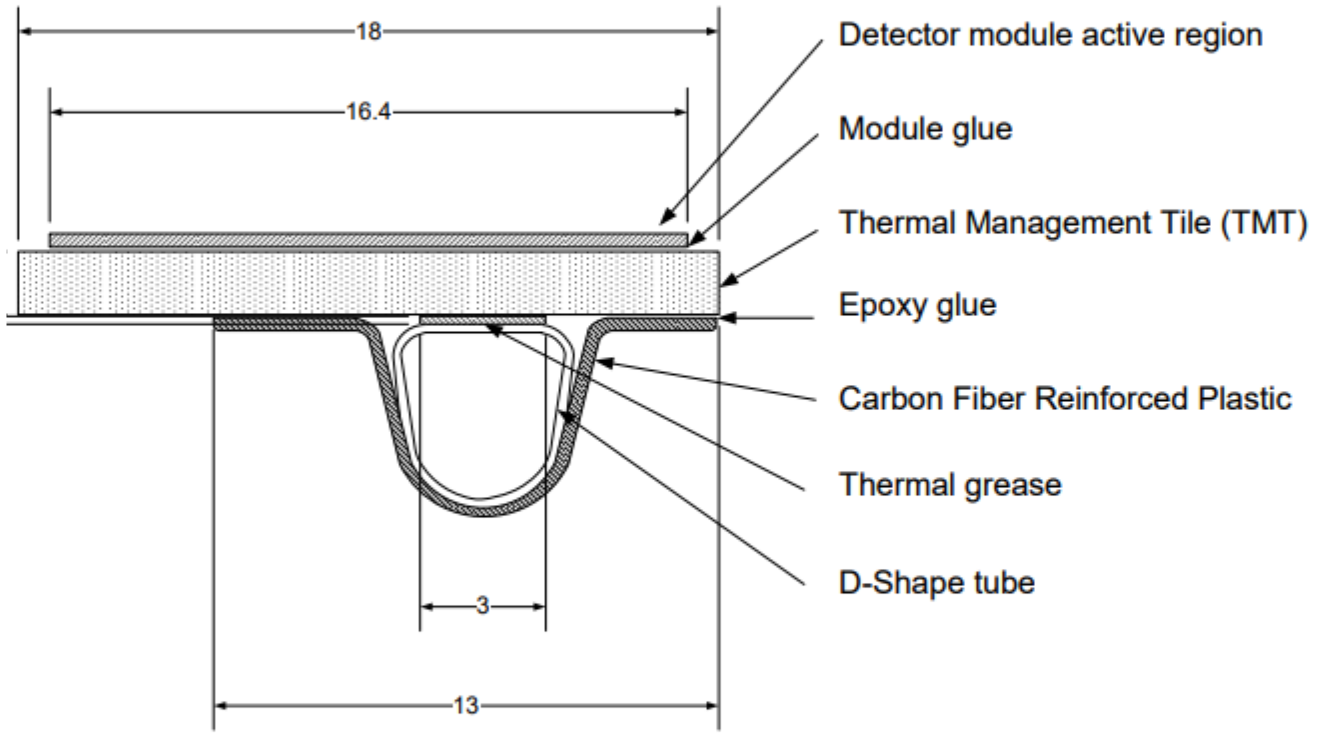
ATLAS Pixel detector – 112 staves



No. of staves: B-layer – 22, Layer 1 – 38, Layer 2 – 52

Pixel Stave Profile. Thermal Management Tile





18
16.4
3
13
Detector module active region
Module glue
Thermal Management Tile (TMT)
Epoxy glue
Carbon Fiber Reinforced Plastic
Thermal grease
D-Shape tube

Redefinition of TMT: old vs new

Profile comparison of two geometries:
aibuild027.cern.ch, SLC6, **1.20 GHz**
FullG4 simulation, Geant4 10.01.patch03.atlas07, 10 events
Average time per event: **238.0 s** vs **232.9 s** (**~2%** less)

	Inside **x 10^6**	**Normal** **x 10^6**	**SafetyToIn** **x 10^6**	**SafetyToOut** **x 10^6**	**DistToIn** **x 10^6**	**DistToOut** **x 10^6**
G4Box	72.9 | 37.6	1.3 | 1.2	21.2 | 18.6	5.1 | 4.9	9.8 | 8.7	4.6 | 4.2
G4Trap	124.2 | 58.0	1.0 | 1.0	110.3 | 107.3	43.4 | 44.2	38.0 | 34.2	23.6 | 23.0
G4Trd	32.8 | 30.9	0.5 | 0.5	18.5 | 17.7	15.4 | 15.0	7.3 | 7.0	10.3 | 10.0
G4Tubs	150.4 | 146.9	9.1 | 8.9	165.9 | 162.4	122.5 | 121.0	65.2 | 64.4	45.3 | 44.7
G4Extruded	3.9 | 3.6		4.0 | 3.5	0.5 | 0.5	0.6 | 0.5	0.1 | 0.1
G4Polycone	317.0 | 314.8	1.6 | 1.6	21.6 | 21.3	115.7 | 114.8	9.0 | 8.9	39.0 | 38.8
G4Union	1168.7 | 33.9	3.4 | 1.8	23.0 | 16.6	6.4 | 4.7	13.8 | 8.7	22.7 | 3.2
G4Subtraction	62.9 | 42.4	0.4 | 0.3	17.8 | 15.1	2.8 | 2.2	3.4 | 2.9	1.2 | 0.9
G4Displaced	1171.4 | 73.2	1.4 | 1.3	36.4 | 28.5	4.7 | 4.8	19.9 | 14.1	5.3 | 4.0
Total	**3104 | 741**	**19 | 17**	**419 | 391**	**316 | 312**	**167 | 149**	**152 | 128**
	4177 vs 1738 (G4Tubs – 548, G4Polycone – 500)					

Redefinition of TMT. Summary

- Redefinition of TMT has reduced the number of calls to `Inside()` by several times:
  - more than 4 times in comparison Geant4 10.01.patch03.atlas07 (741 x10^6 instead of 3104 x10^6)
  - more than 6 times in comparison with Geant4 10.01.patch03.atlas02 (741 x10^6 instead of 4700 x10^6)
- Combination of *Geant4 10.01.patch03.atlas07 & new TMT* may give 5% improvement in the performance of FullATLASG4 simulation
- After redefinition of TMT, 60% of calls are to the methods of G4Tubs and G4Polycone (31.5% + 28.8%)

Usage of G4Polycone solids

- EMEC Wheels
- Beam pipe
- World and Muon system volumes

General statistics

- After redefinition of TMT the distribution of calls to Geant4 Solids is as follows:
  - ~30% to the methods of G4Polycone
  - ~30% to the methods of G4Tubs
  - ~40% to the methods of other G4 solids
- **So, it is clear that revision of G4Tubs and, especially, G4Polycone has sense**
- There are 135 different G4Polycone solids, below is the summary of their usage

EMEC Wheels

Muon System

Beam Pipe

N. of Z-sections	N. of solids	% of calls	% of time	N. of Z-sections	N. of solids	% of calls	% of time
Nz = 2	25 solids	32.6%	10.1%	**Nz = 12**	4 solids	2.0%	6.7%
Nz = 3	17 solids	54.4%	33.7%	**Nz = 17**	1 solid	0.1%	0.5%
Nz = 4	31 solids	2.1%	1.9%	**Nz = 18**	3 solid	1.7%	8.7%
Nz = 5	5 solids	0.4%	0.5%	**Nz = 24**	1 solid	0.3%	2.4%
Nz = 6	38 solids	2.3%	3.6%	**Nz = 26**	1 solid	0.7%	5.3%
Nz = 7	4 solids	0.1%	0.2%	**Nz = 32**	1 solid	0.4%	3.8%
Nz = 10	1 solid	0.3%	0.7%	**Nz = 34**	2 solids	0.7%	7.0%
				Nz = 38	1 solid	1.3%	14.9%

Potential optimization

There are three places where relatively simple modifications in the geometry can improve the performance:

- EMEC Inner and Outer Wheels (estimated percentage of time is ~40% or ~4% of CPU)
- Beam Pipe mother volume (estimated percentage of time is ~15% or ~1.5% of CPU)
- World volume + Muon System mother volume (estimated percentage of time is ~9% or ~0.9% of CPU)

Remark: (% of CPU time) ~= (% of time spent in G4Polycone) / 10

EMEC Inner Wheel

All solids below have exactly the same parameters:

G4Polycone : Nz=2 Sphi=0^o Dphi=360^o (can be replaced with G4Cons)

z, r_{min}, r_{max} = 0, 302.3, 610.4

z, r_{min}, r_{max} = 514, 344.3, 695.3

FullG4 simulation, 10 events, 500 x10^6 calls to G4Polycone methods

	Inside x 10^6	**Normal x 10^6**	**SafetyToIn x 10^6**	**SafetyToOut x 10^6**	**DistToIn x 10^6**	**DistToOut x 10^6**
LAr::EMEC::Neg::InnerWheel	4.65		0.28	5.31	0.06	0.11
::Absorber-BoundingPolycone	19.69			4.93		2.59
::Electrode-BoundingPolycone	14.65			2.32		1.36
::Glue-BoundingPolycone	17.35			3.42		2.57
::Lead-BoundingPolycone	18.68			7.05		1.93
Total	**106.5 x10^6 (21%)**					
LAr::EMEC::Pos::InnerWheel	2.20		0.17	2.47	0.04	0.06
::Absorber-BoundingPolycone	9.33			2.28		1.25
::Electrode-BoundingPolycone	6.95			1.08		0.66
::Glue-BoundingPolycone	8.26			1.62		1.24
::Lead-BoundingPolycone	8.72			3.18		0.91
Total	**49.5 x10^6 (10%)**					

EMEC Outer Wheel

All solids below have exactly the same parameters:

G4Polycone : Nz=3 Sphi=0^o Dphi=360^o (a union of two hollow cones)

z, r_{min}, r_{max} = 0, 613.4, 1999.9

z, r_{min}, r_{max} = 63.2, 623.8, 2034

z, r_{min}, r_{max} = 514.0, 698.3, 2034

	Inside x 10^6	Normal x 10^6	SafetyToIn x 10^6	SafetyToOut x 10^6	DistToIn x 10^6	DistToOut x 10^6
LAr::EMEC::Neg::OuterWheel	7.95		0.27	8.99	0.04	0.10
::Absorber-BoundingPolycone	35.26			8.67		4.88
::Electrode-BoundingPolycone	26.12			4.14		2.57
::Glue-BoundingPolycone	31.87			6.24		4.84
::Lead-BoundingPolycone	31.15			10.49		3.33
Total	**186 x10^6 (37%)**					
LAr::EMEC::Pos::OuterWheel	2.12		0.15	2.29	0.01	0.03
::Absorber-BoundingPolycone	9.65			2.24		1.42
::Electrode-BoundingPolycone	7.12			1.08		0.74
::Glue-BoundingPolycone	8.94			1.72		1.41
::Lead-BoundingPolycone	8.20			2.42		0.91
Total	**50 x10^6 (10%)**					

EMEC Wheels. Summary

- EMEC Inner and Outer Wheels take 78% of calls to G4Polycone and approximately 40% of the time spent in G4Polycone
- The G4Polycone solids used in the EMEC Wheels are auxiliary objects, they are not used in any logical or physical volumes
- Suggestions:
  - Replace the G4Polycone solids in EMEC Inner Wheel (Nz=2) with G4Cons
  - Consider replacement of the G4Polycone solids in EMEC Outer Wheel (Nz=3) with simpler shapes

Beam pipe

- The BeamPipe mother volume is a Boolean union of its daughter volumes. It consists of three parts: one G4Tubs solid (BeamPipeCentral) and two G4Polycone solids (BeamPipeFrw)

```
BeamPipeFrw & BeamPipeCentral & BeamPipeFrw
```

- BeamPipeFrw has 38 Z-sections (37 tubes), takes 1.3% of calls to G4Polycone and approximately 15% of the time spent in G4Polycone
- Suggestions:
  - Eliminate the BeamPipe mother volume - BeamPipeCentral and BeamPipeFrw can be included directly to the World volume
  - In addition BeamPipeFrw can be simplified

World volume and Muon System mother volume

- The structure of the top volumes:

```
Atlas -> MUONQ02 -> MuonSys
      -> CALO
      -> IDET
      -> BeamPipe
```

- World volume (Atlas) has 18 Z-sections (17 cylinders) and takes 0.39% of calls to G4Polycone
- MUONQ02 has 34 Z-sections (33 tubes) and takes 0.36% of calls to G4Polycone
- MuonSys has very similar shape to the shape of MUONQ02 (34 Z-sections) and takes 0.32% of calls to G4Polycone
- Atlas + MUONQ02 + MuonSys take approximately 9% of the time spent in G4Polycone
- Suggestions:
  - Simplify the World volume
  - Revise (simplify or, possibly, entirely eliminate) the bounding volume for Muon system

CPU consumption by Geometry & Navigator

Profile of Geantino test

- Comparison of the CPU time spent by the Geantino test and the CPU time of real simulation has allowed to estimate CPU consumption by the Geometry & Navigation code.
- Below is distribution of calls by Solids and Methods for the Geatino test run on the GDML dump, where 112 physical volumes with TMT have been removed

MacBook Pro, Intel Core i5, **2.7 GHz**
100k events, No. of steps: 159.739.590, CPU time: **186 s**

	Inside x 10^6	**Normal x 10^6**	**SafetyToIn x 10^6**	**SafetyToOut x 10^6**	**DistToIn x 10^6**	**DistToOut x 10^6**
G4Box	236	4	38	15	34	21
G4Trap	30		28	25	25	25
G4Trd	264	2	54	62	50	65
G4Tubs	640	0.6	84	44	76	41
G4Extruded	17	0.6	2	1	2	1
G4Polycone	28	0.3	5	13	5	6
G4Union	164	7	23	19	21	21
G4Subtraction	656	6	15	26	13	25
G4Displaced	865	4	54	8	52	15
Total	**2900**	**25**	**303**	**213**	**278**	**220**
		All – 1039 (G4Tubs – 246, G4Polycone – 29)				

Profile of real simulation

aibuild027.cern.ch, SLC6, **1.20 GHz**
Simulation of 10 ttbar events, Geant4 10.01.patch03.atlas07 (currently under verification)
No. of steps: unknown, CPU time: **~250 s / event** (>35% in EMEC)

	Inside x 10^6	**Normal x 10^6**	**SafetyToIn x 10^6**	**SafetyToOut x 10^6**	**DistToIn x 10^6**	**DistToOut x 10^6**
G4Box	38	1	19	5	9	4
G4Trap	101	1	111	42	38	23
G4Trd	29		16	14	6	9
G4Tubs	146	9	168	124	66	46
G4Extruded	2		3			
G4Polycone	332	1	21	120	9	41
G4Union	1162	3	23	6	14	22
G4Subtraction	35		13	1	2	
G4Displaced	1149	1	34	4	19	5
Total	**2994**	**16**	**408**	**316**	**163**	**150**
		All – 1053 (G4Tubs – 413, G4Polycone – 192)				

CPU consumption by Solids & Navigator

- Quantitative characteristics of two tests are very similar:

  Number of calls to `Inside()`: 2994 vs 2900 (x10^6)

  Number of calls to other functions: 1053 vs 1039 (x10^6)

- So, the execution time of the Geantino test can be used as a good estimation of the CPU time consumed by Geant4 Solids & Navigator in real simulation:

  Conversion factor between the platforms: 2.70 GHz / 1.20 GHz = 2.25

  Estimation of the execution time of the Geantino test on aibuild027.cern.ch: 186 * 2.25 >= 420 s

  CPU time that would be consumed by Geant4 Solids & Navigator in one event: 420 / 10 >= 42 s

  Estimated percentage: 42 / 250 >= 17%

  Estimated percentage if to exclude the time spent in EMEC (>35%): 42 / (250 * 0.65) >= 26%

- Taking into account that in real simulation there were more calls to G4Tubs and G4Polycone, 413 vs 246 (x10^6) and 192 vs 29 (x10^6) respectively, it will be correct to say that the percentage of

CPU time consumed by Geant4 Solids & Navigator varies between 20% and 30%

Miscellaneous

- Resolution of G4Exception messages
- Enhancement of G4AffineTransform
- Optimizations in navigation code

Resolution of G4Exception warnings

- There were 72 G4Exeption messages during simulation of 500 events with current production version Geant4 10.01.patch03.atlas02:
  - 63 messages - *Track stuck or not moving.*
  - 8 messages - *Proposed step is zero; hstep = 0 !*
  - 1 message - *Expected normal-global-frame to be valid, i.e. a unit vector!*
- All G4Exeption messages have disappeared in Geant4 10.01.patch03.atlas07 which at present time is under verification
  - The disappearance of *Track stuck or not moving* is explained by the revision of G4Trd (code taken from Gean4 10.04)
  - The disappearance of *Proposed step is zero* is explained by the bug fix in the calculation of the normal at the endpoint of current step in G4Navigator::GetLocalExitNormal() (code taken from Gean4 10.04.p02)
  - It looks that the message on invalid normal was due to calculation errors. It has disappeared after last modifications in G4DisplacedSolid (Gean4 10.04.p02)

Enhancement of G4AffineTransform (1)

- G4AffineTransform is a class that is used to define the position of solids and to make transformation of points and vectors
- It is implemented as a 4x3 transformation matrix – a 3x3 rotation matrix and a 3-vector shift
- All methods are inlined
- The methods that are used at navigation:

  `Inverse()` – it returns the inverse transformation matrix

  `Invert()` – it inverts the transformation matrix

  `InvertProduct(t1,t2)` – it calculates `t1*(t2`$^{-1}$`)`

  `TransformPoint(p)` – it transforms a point

  `TransformAxis(v)` – it transforms a vector

Enhancement of G4AffineTransform (2)

- `G4AffineTransform::InvertProduct(t1,t2)` has been optimized:
  - now it does 36 multiplications instead of 45 as before
  - A specialization for the case where `t2` is a pure translation has been added.
    In such a case the computation of the product is very light:
    3 subtractions, 12 assignments, no multiplications
- Four new methods have been added:

  `InverseTransformPoint(p)` – it makes inverse transformation of a point

  `InverseTransformAxis(v)` – it makes inverse transformation of a vector

  `InverseNetRotation()` – it returns inverse rotation matrix

  `InverseNetTranslation()` – it returns the shift vector of the inverse transformation

Optimizations in the navigation code

- Inverse transformation of points and vectors does not require addition computations in comparison with direct transformation:

```
G4ThreeVector G4AffineTransform::TransformPoint(const G4ThreeVector& vec) const
{
   G4double vecx = vec.x(), vecy = vec.y(), vecz = vec.z();
   return G4ThreeVector( vecx*rxx + vecy*ryx + vecz*rzx + tx,
                         vecx*rxy + vecy*ryy + vecz*rzy + ty,
                         vecx*rxz + vecy*ryz + vecz*rzz + tz );
}


G4ThreeVector G4AffineTransform::InverseTransformPoint(const G4ThreeVector& vec) const
{
   G4double vecx = vec.x() - tx, vecy = vec.y() - ty, vecz = vec.z() - tz;
   return G4ThreeVector( vecx*rxx + vecy*rxy + vecz*rxz,
                         vecx*ryx + vecy*ryy + vecz*ryz,
                         vecx*rzx + vecy*rzy + vecz*rzz );
}
```

- New methods in G4AffineTransform allowed to eliminate Calculation/Instantiation of the inverse transformation in several navigation classes, for example:

```
G4ThreeVector md = Td.Inverse().TransformPoint(mp);
```

has been replaced with:

```
G4ThreeVector md = Td.InverseTransformPoint(mp);
```

Summary

- A detailed quantitative statistics on the use of Geant4 in ATLAS has been gathered. Jira tickets dedicated to specific findings can been found at the Epic collection ATLASSIM-3678 *Improve the description of ATLAS in Geant4.*
  In particular:
  - ATLASSIM-3556 *Overlaps detected in MC16 geometry when running in standalone Geant4*
  - ATLASSIM-3557 *Optimizing Boolean solid use in ATLAS G4 geometry*
  - ATLASSIM-3680 *Revision of TMT*
  - ATLASSIM-3778 *Optimizing G4Polycone usage in ATLAS G4 geometry*
  - ATLASSIM-3761 *Performance comparison of four geant4 patches*
- A number of improvements have been done in Geant4, namely in G4Box, G4DisplacedSolid, G4UnionSolid, G4AffineTransform and navigation code (10.01.patch03.atlas07).
  The improvements will be included in Geant4 10.05
- The issue with the description of the TRT mother volume in GDML has been fixed
- The G4Exception warnings observed during simulation of 500 events have disappeared
- The combination of Geant4 10.01.patch03.atlas07 & optimization of TMT has shown **5%** speed-up in the performance for FullATLASG4 simulation and **11%** speed-up for ATLFASTII
- Further improvement of the performance can be achieved by optimizing G4Polycone usage in the ATLAS geometry and revising G4Polycone code in Geant4

Thank you!