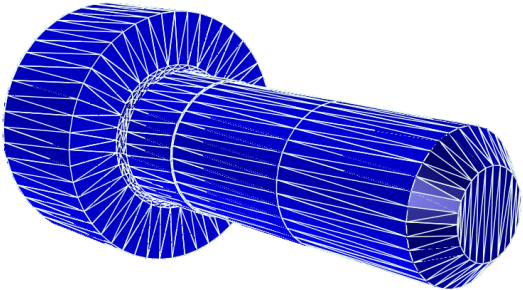
# TGeoArbN and how we use it

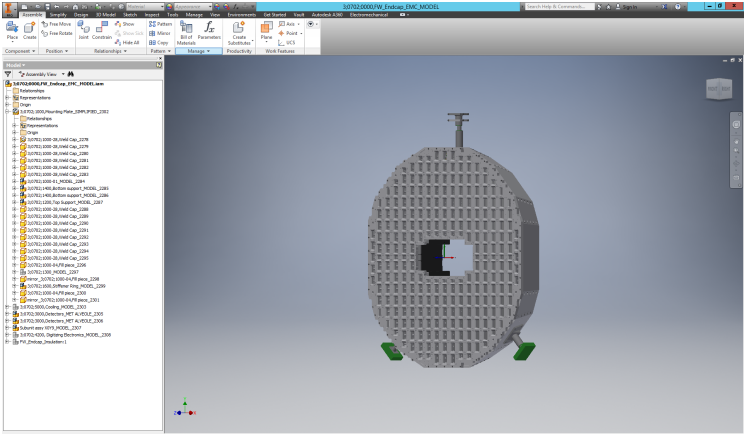
Ben William Salisbury

HISKP Universität Bonn

June 25, 2019

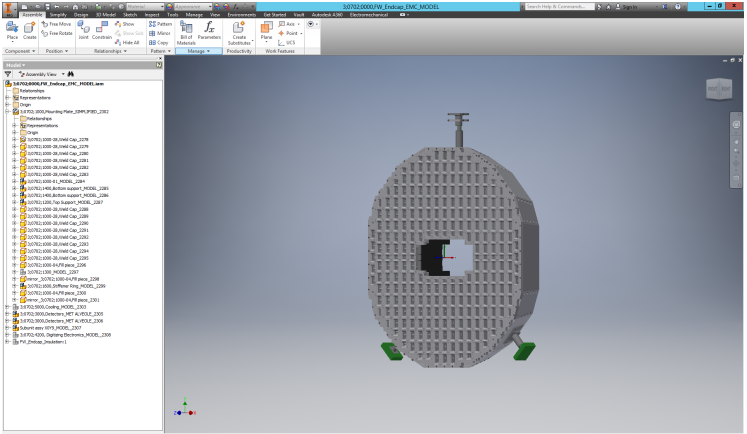


# From CAD to simulatable ROOT representations



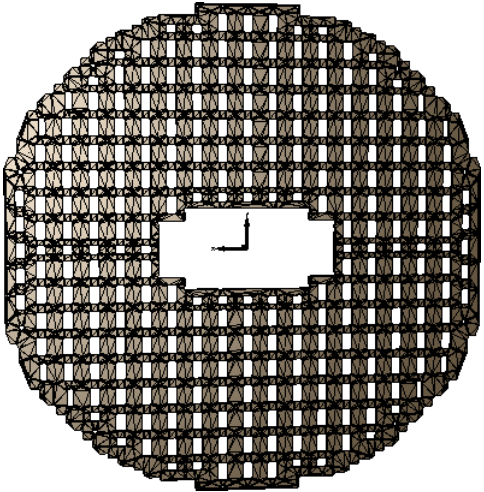
- Detectorcomponents are designed with **CAD**, but simulation studies need **ROOT**-geometries
- Complex geometries need to be **manually** built using ROOT-geometry primitives

# From CAD to simulatable ROOT representations



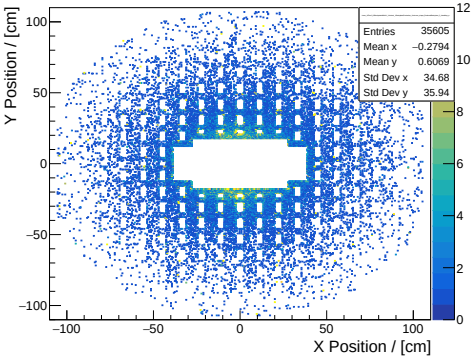
- We want quick reaction times during the design
- We need an accurate ROOT representation
- We must know, what the cost of approximating instead of replicating is

# TGeoArbN



- Every CAD-object exportable as **triangle mesh**.
- TGeoArbN derives from TGeoShape allowing use in simulation framework
- **Triangle mesh + ray casting** → simulatable geometry

# TGeoArbN

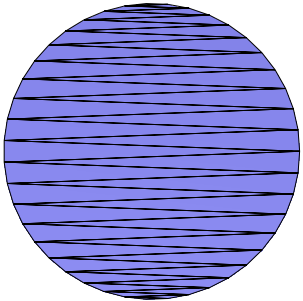



Hit map of electrons on the Backplate

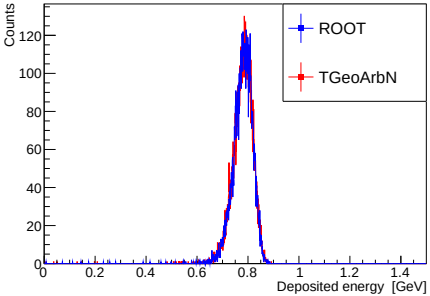
- **TGeoArbN geometry works!**

# **Triangle mesh** issue: Precision

Curves can only be approximated. This leads to differences in the volume for curved shapes.



Tessellated base of a cylinder.

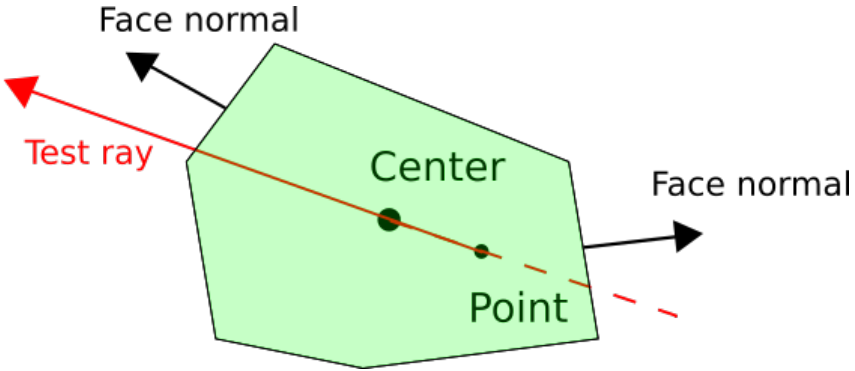



Deposited energy of a 1 GeV photon in a cylindric form.

Deviations depend on **triangle mesh** and geometry

# ray casting: `Contains()` implementation

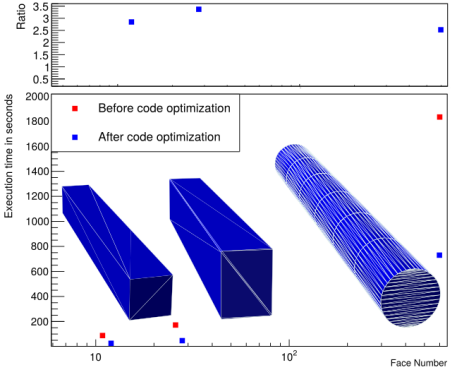
`Contains()`: Is a point (particle) contained by the geometry?




2D example of ray-casting to test, if a point is inside or outside a shape.

# The main issue: Performance

Execution time increases with the number of triangles in the mesh.




Execution time in seconds for 1000 1 GeV photons shot at one point against the triangle/face number

Bad: We want complex geometries → More faces

# Conceptual modification - Octree

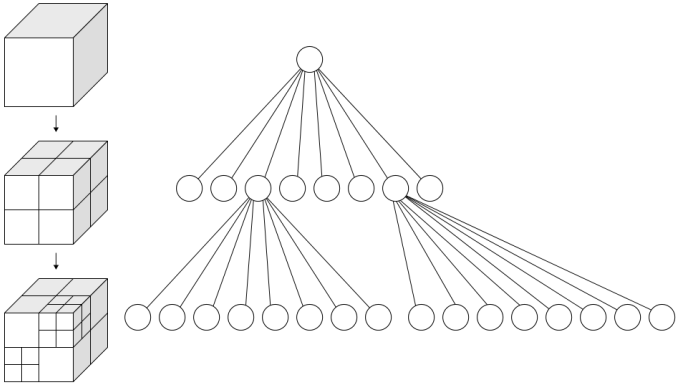
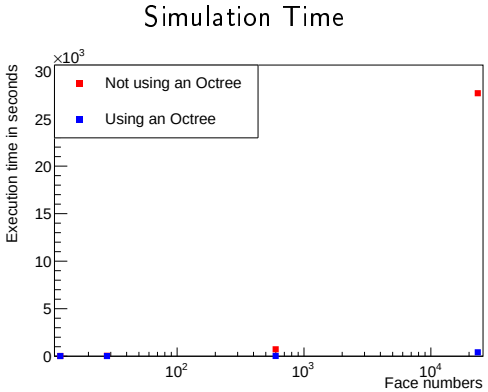


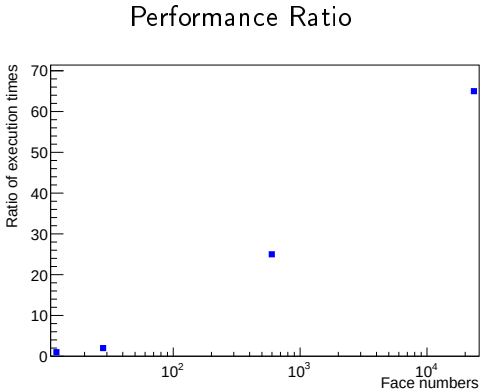
Figure: Octree-concept: picture taken from WhiteTimberwolf, *Schematic drawing of an octree, a data structure of computer science.*

Bounding box recursively split into eight identical sub-boxes (octants). Splitting occurs if a box contains more than $x$ triangles.

# Overall simulation time measurement

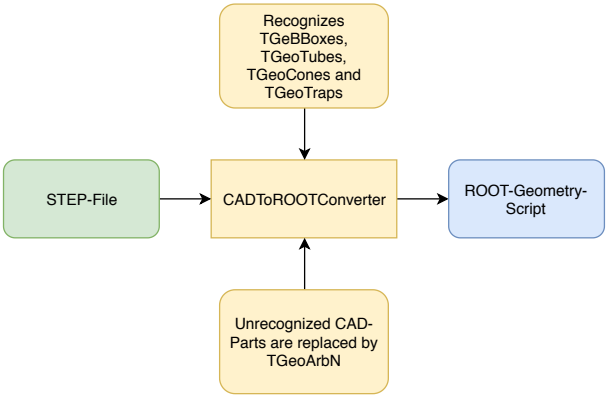



Execution time for 1000 1 GeV photons shot at one point for different geometries using an Octree of depth 5



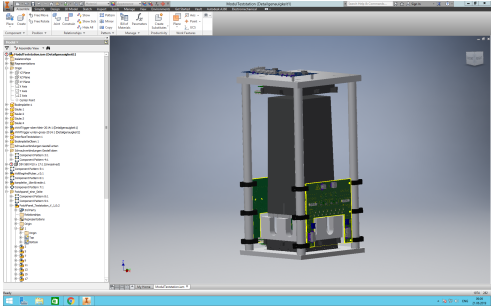

Ratio of execution times for different geometries using an Octree of depth 5

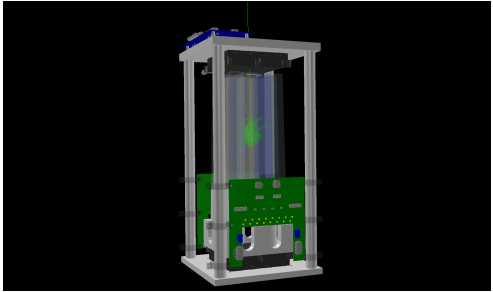
# How do we use it?




- Written a simplified version of Tobias' CADToROOTConverter in python
- Major difference: unrecognized CAD-Parts get tessellated and replaced by a TGeoArbN instance

# Using CADToROOTConverter to convert Alveole-TestStation

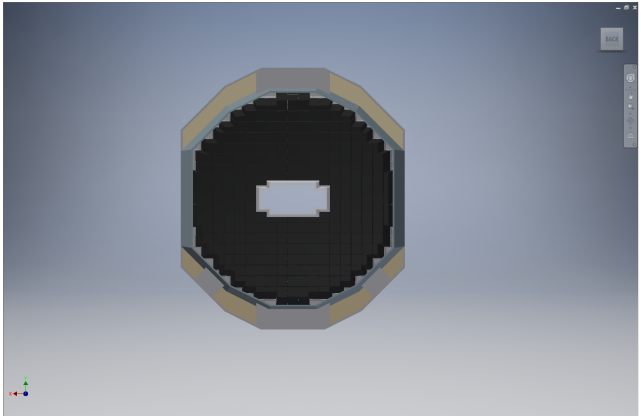


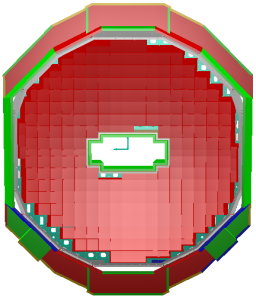


Conversion process took me roughly 3 hours (I had to split up the CAD-Model into two groups due to naming clashes)

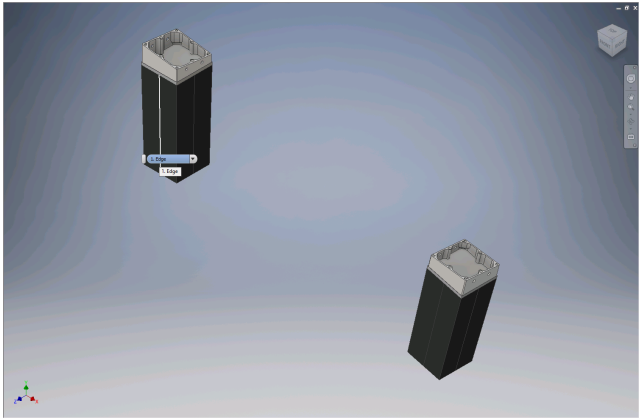
# Progress on a new FwEndcap geometry version

Looks good but, sadly, is not usable for simulation.





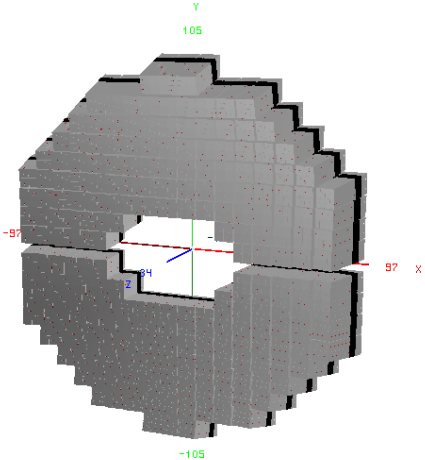
# Progress on a new Forward-Endcap geometry version



- The CAD-Model is built using placeholder-objects as Alveoles.
- They do not include Crystals, etc.
- Wrong relative placement

→ Have to build a new CAD-Model using Alveole-Models

# Summary




- The combination of TGeoArbN and the CADToROOTConverter can allow for quick geometry creation
- We still need to test TGeoArbN in a full simulation (performance, precision, intersection of geometries)

# Bibliography I

WhiteTimberwolf. *Schematic drawing of an octree, a data structure of computer science.* 27 March 2010. URL: https://commons.wikimedia.org/wiki/File:Octree2.svg.