EMC-geometries
A first step for special shapes

C. Schmidt, A. Wilson, C. Hammann, U. Thoma

8-12 June 2015, Computing Session, Uppsala, Sweden

Helmholtz Institut für
Strahlen- und Kernphysik

universitätbonn

Outline

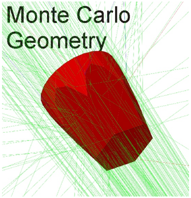
- **Motivation**
  - **Idea how to move complex shapes from CAD to ROOT**
- **Implementation**
  - **Example `Contains` function**
- **Performance aspects**
  - **First quick tests**
  - **Comparison to existing root geo object**
- **Outlook**

From Drawings to Root Objects

universitätbonn

Workflow

- Use drawings/cad to extract position/dimension/shape
- Construct geometry using basic root shapes (redo the work of the engineer)
- Transfer position data to root scene



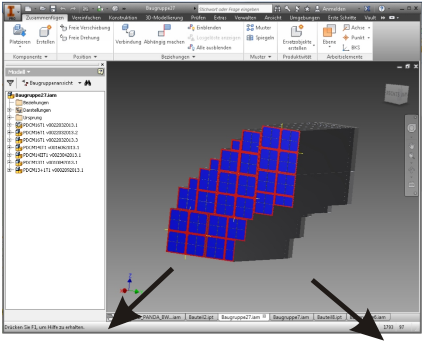
Drawbacks

- Very time comsuming
- Error-prone procedure
- If things change, task may start all over again
- To avoid costly work, shapes get simplified

From Drawings to root objects

Nice to have

Export a component from CAD into files, which contain positions and shapes.



```
crystal.001 = (10.23,2.3,12.00)
crystal.002 = (.......)
FrontStopper = (10.3,4.5,14.5)
.....
```

From Drawings to root objects

What is available to achieve this?

- **File format**
  Standard for the Exchange of Product Model Data (STEP)
  - **Contains detailed information**
    (e.g. shapes, positions, names, logical structure)
  - **not easy to use, helper libs needed**
    (e.g. STEP Tools, OpenCascade,...)
- **Converter**
  CADconverter (T. Stockmanns)
  - **Uses STEP-file as input**
  - **Extracts positions or simple shapes (boxes, cones)**
  - **Replaces objects by names with predefined root objects**

$\rightarrow$ Combination helps a lot for transfering CAD informations
(e.g. used in the MVD)

But still:
Complex shapes must be constructed by hand using root shapes.

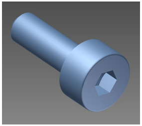
Idea - Import Complex Shapes

Problems in automatic conversions

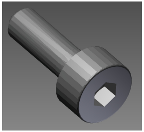
- Root geo objects are restricted to a few simple shapes.
- No way to convert CAD objects to basic root geo objects.

Way out, approximate objects/componentes by tesselation.

- Standard: The Standard Tessellation Language (STL)
- File format is widely used in the 3D printing business
- All CAD programs can export tesselated objects
- Precision of approximation can be defined on export



Tesselation
(STL)



$\rightarrow$ **Perfectly suited for usage.**

Problem:
There is NO root geo object, which supports tesselated shapes

New Root Geometry Object - TGeoArbN

Introduce a new geometry object

- **Take advantage of existing tessellation algorithms**
- **Build a Root Geometry object which can be defined by reading in the tessellated information.**

TGeoArbN

Starting point

- **Use TGeoArb8 as template**
- **List of points and faces (e.g. triangles)**
- **Each object is a collection of planar faces**
- **No restriction on number of points, which define a face (flexibility)**

universitätbonn

User function to add tesselated data

Add Points, returns index of point in internal array

```
Int_t AddPoint(Double_t x, Double_t y, Double_t z)
```

Add Face, index is list of point indexes, returns index of face in internal array

```
Int_t AddFace( Int_t index[], Int_t Ncorners = 3)
```

Order of indexes is important! Adjacent indexes define an edge and order defines the normal vector (consistent with tessellation algorithms).

Requirements for simulation

Point Contained

```
Bool_t Contains(const Double_t *point[3])
```

Distances

```
Double_t Safety(const Double_t *point[3], Bool_t inside)
```

Directional Distances

```
Double_t DistFromInside(Double_t *point[3], Double_t *dir[3],
      Int_t iact, Double_t step, Double_t *safe)
Double_t DistFromOutside(Double_t *point[3], Double_t *dir[3],
      Int_t iact, Double_t step, Double_t *safe)
```

Normal vector

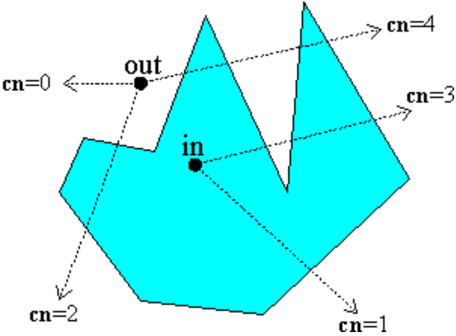
```
void ComputeNormal(Double_t *point[3], Double_t *dir[3],
      Double_t* normal[3])
```

Defining the Contains function

```
Bool_t TGeoShape::Contains(const Double_t *point[3])
```

Strategy

- **Pick a ray in any direction, originating at point**
- **If the number of crossed faces is odd, the point is inside**
- **if the number of crossed faces is even, the point is outside**




```
http://geomalgorithms.com/a03-_inclusion.html
```

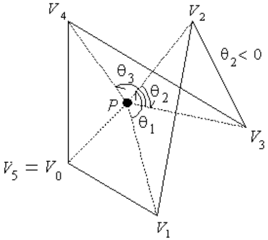
Defining the Contains function

```
Bool_t TGeoShape::Contains(const Double_t *point[3])
```

Finding Number of Ray-Crossing Faces

- **Pick the ray to be through the center of the first face.**
- **Calculate the intersection point between the ray and the plane which contains a face.**
- **Test whether intersection point is inside the face. (winding number algorithm)**

Winding Number Algorithm **While following the edges of the face, how many times does the shape “winds” around the point?**

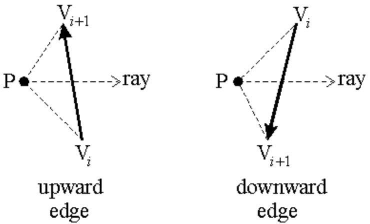


Defining the Contains function

```
Bool_t TGeoShape::Contains(const Double_t *point[3])
```

Winding Number Algorithm

Reduces to: how many edges cross a ray and in which direction?



- **upward edge : windingnumber++**
- **downward edge: windingnumber –**

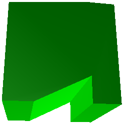
After summing over all edges

if winding number is zero then intersection point is outside face

Simple test procedure

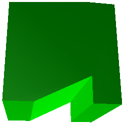
Definition of a simple test object

TGeoArbN



14 points and 9 faces

TGeoXtru



Extruded face with 7 points

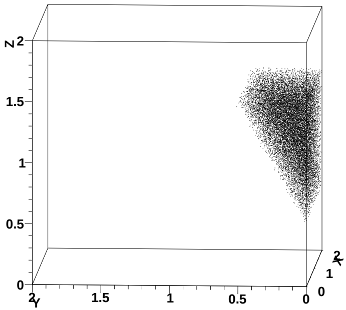
Testing with Monte Carlo method

- **A cube which contains the whole object in space**
- **Evaluate the contains function at random points within the cube**

$$Volume = V_{BoundingBox} * \frac{\#contained\ points}{\#generated\ points}$$

Result of Monte Carlo of test object

- **40000 random points generated inside the bounding box.**

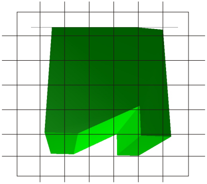


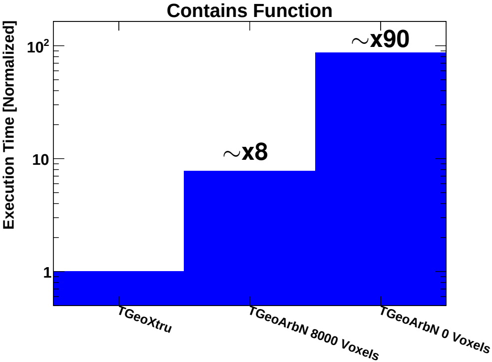
- **Monte Carlo method $\rightarrow$ V = 7.48714 $\pm$ 0.0374 (by defintion 7.5)**
- **BUT: compared to TGeoXtru $\sim$90 times slower**

Implemented Optimizations

Optimization by pre-evaluation

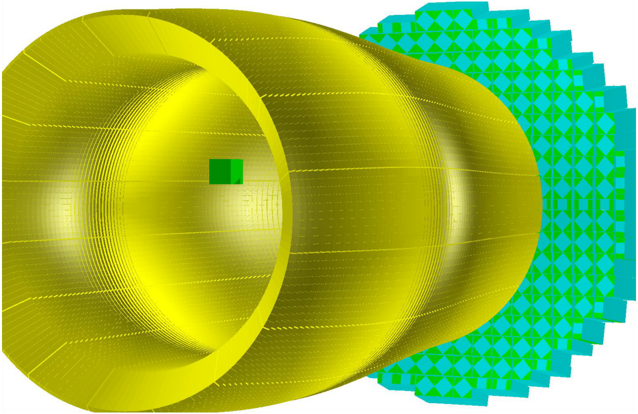
- **Implemented internal voxelation**
- **Predict `Contains` function for voxels**
  - **complete inside/outside $\rightarrow$ distinct**
  - **contains surface call `Contains`**
- **Pre-estimate and store closest Face**



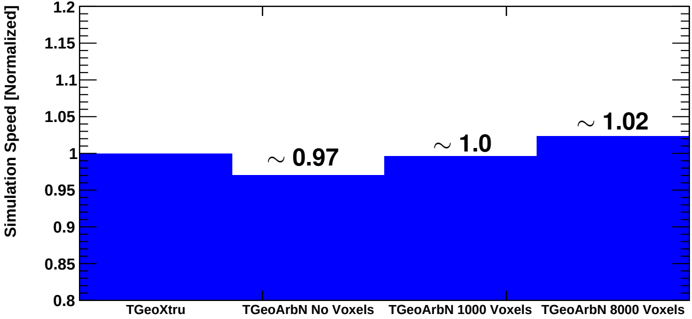


Simulation Performance

- **Included test object into a PANDAroot simulation along with EMC**
- **Box Simulation 5x 2-GeV photons, isotropic: 1000 Events**



- **Included test object into a PANDAroot simulation along with EMC**
- **Box Simulation 5x 2-GeV photons, isotropic: 1000 Events**



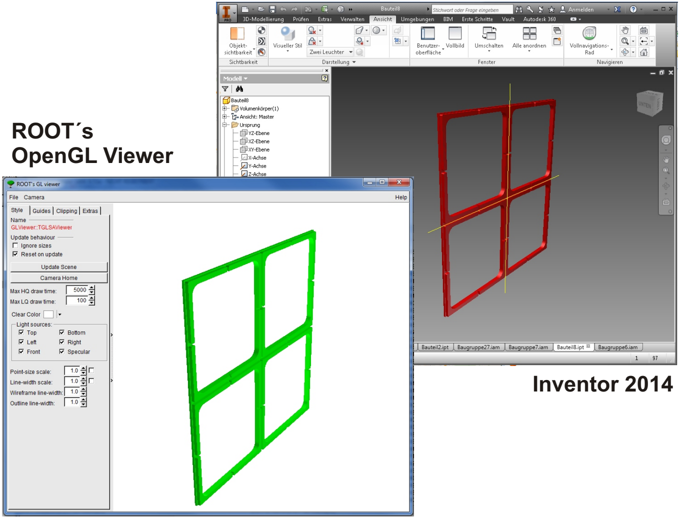
In this case

- **No slow down of simulation with voxelation**
- **Very basic test → tests with complex shapes needed**

universitätbonn

Example of exported component from Inventor to root

ROOT´s OpenGL Viewer

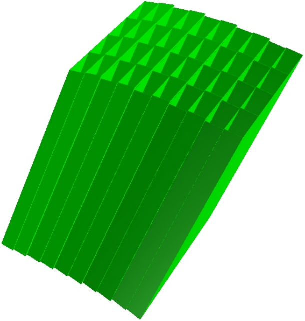


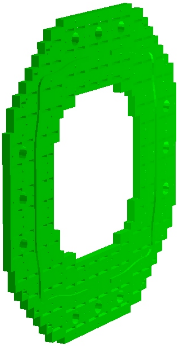
Inventor 2014

universitätbonn

Example of exported component from Inventor to root

Alveole
Barrel slice





Backplate
BW-Endcap

TGeoArbN

- New Root Geometry Object created to read in tessellated objects
- Defined and running with PANDAroot
- In test case, if fraction of total volume is small
  $\rightarrow$ simulation does not significantly slow down
- Should be used wisely
- Very useful for short ”time to simulation”, if speed is less important
- A way to compare exact geometry with a simplification

Next steps

- More detailed performance tests (complexer objects, many faces > 500)
- Test it together with CADconverter (positioning)
- Compare simulation results to basic root objects
- Export CAD objects and test with reasonable geometry

TGeoArbN

- New Root Geometry Object created to read in tessellated objects
- Defined and running with PANDAroot
- In test case, if fraction of total volume is small
  $\rightarrow$ simulation does not significantly slow down
- Should be used wisely
- Very useful for short "time to simulation", if speed is less important
- A way to compare exact geometry with a simplification

Next steps

- More detailed performance tests (complexer objects, many faces $>$ 500)
- Test it together with CADconverter (positioning)
- Compare simulation results to basic root objects
- Export CAD objects and test with reasonable geometry

Thank you for your attention