

# New software library of geometrical primitives for modelling of solids used in Monte Carlo detector simulations

**Marek Gayer, John Apostolakis, Gabriele Cosmo, Andrei Gheata, Jean-Marie Guyader, Tatiana Nikitina**  
CERN PH/SFT

The International Conference on Computing in High Energy  
and Nuclear Physics (CHEP), New York, May 21-25, 2012

# Motivations for a common solids library

- Optimize and guarantee better long-term maintenance of Root and Geant4 solids libraries
  - A rough estimation indicates that about 70-80% of code investment for the geometry modeler concerns solids, to guarantee the required precision and efficiency in a huge variety of combinations
- Create a single library of high quality implementations
  - Starting from what exists today in Geant4 and Root
  - Adopt a single type for each shape
  - Create a new Multi-Union solid
  - Aims to replace solid libraries in Geant4 and Root
  - Allowing to reach complete conformance to GDML solids schema
- Create extensive testing suite

# Navigation functionality and library services for each solid

- **Performance critical methods:**
  - Location of point either inside, outside or on surface
  - Shortest distance to surface for outside points
  - Shortest distance to surface for inside points
  - Distance to surface for inside points with given direction
  - Distance to surface for outside points with given direction
  - Normal vector for closest surface from given point
- **Additional methods:** Bounding Box, Capacity, Volume, Generating points on surface/edge/inside of solid, creating mesh / polyhedra for visualization

# Topics presented next:

...

- Testing suite
- New Multi Union Solid

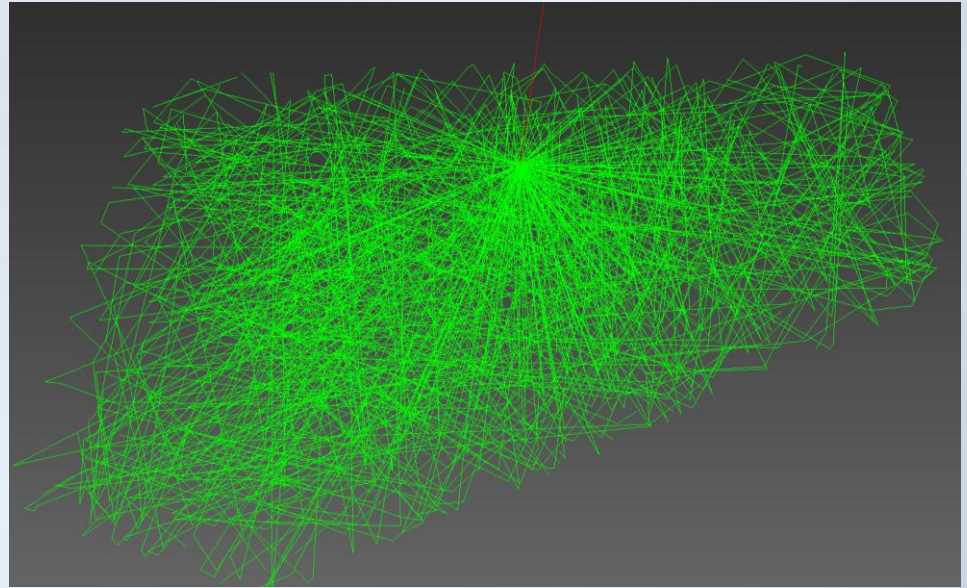
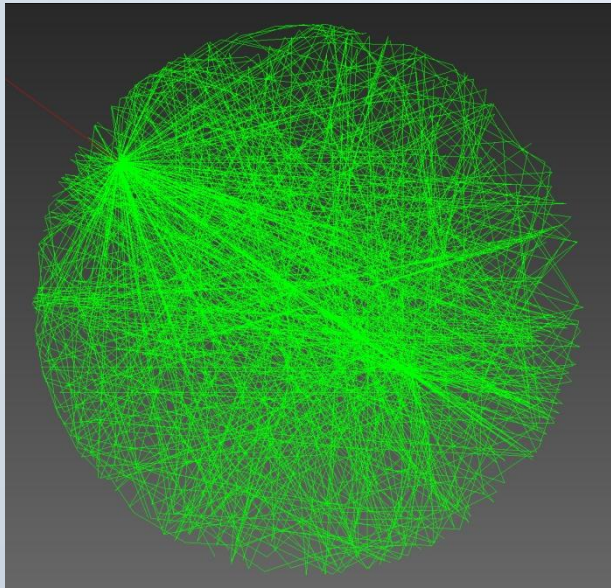
# Testing Suite

...

- Solid Batch Test
- Optical Escape
- Data analysis and performance (SBT DAP)
- Specialized tests (e.g. quick performance scalability test for multi-union)

# Optical Escape Test

- Optical photons are generated inside a solid
- Repeatedly bounce on the reflecting inner surface
- Particles must not escape the solid



# Solids Batch Test (SBT)

- Points and vectors test
  - Generating groups of inside, outside and surface points
  - Testing all distance methods with numerous checks
    - E.g. for each inside random point  $p$ ,  $SafetyFromInside(p)$  must be  $> 0$
- Voxels tests
  - Randomly sized voxels with random inside points
- Scriptable application, creates logs
- Extendible C++ framework
  - Allowing easy addition of new tests

# Data Analysis and Performance (DAP)

...



# DAP features

- Extension of the SBT framework
- Centred around testing USolids together with existing Geant4 and Root solids
- Values and their differences from different codes can be compared
- Constrain: aim to reach similar or better performance in each method
- The core part of USolids testing
- Portable: Windows, Linux, Mac
- Two phases
  - Sampling phase (generation of data sets, implemented as C++ app.)
  - Analysis phase (data post-processing, implemented as MATLAB scripts)

# DAP - Sampling phase

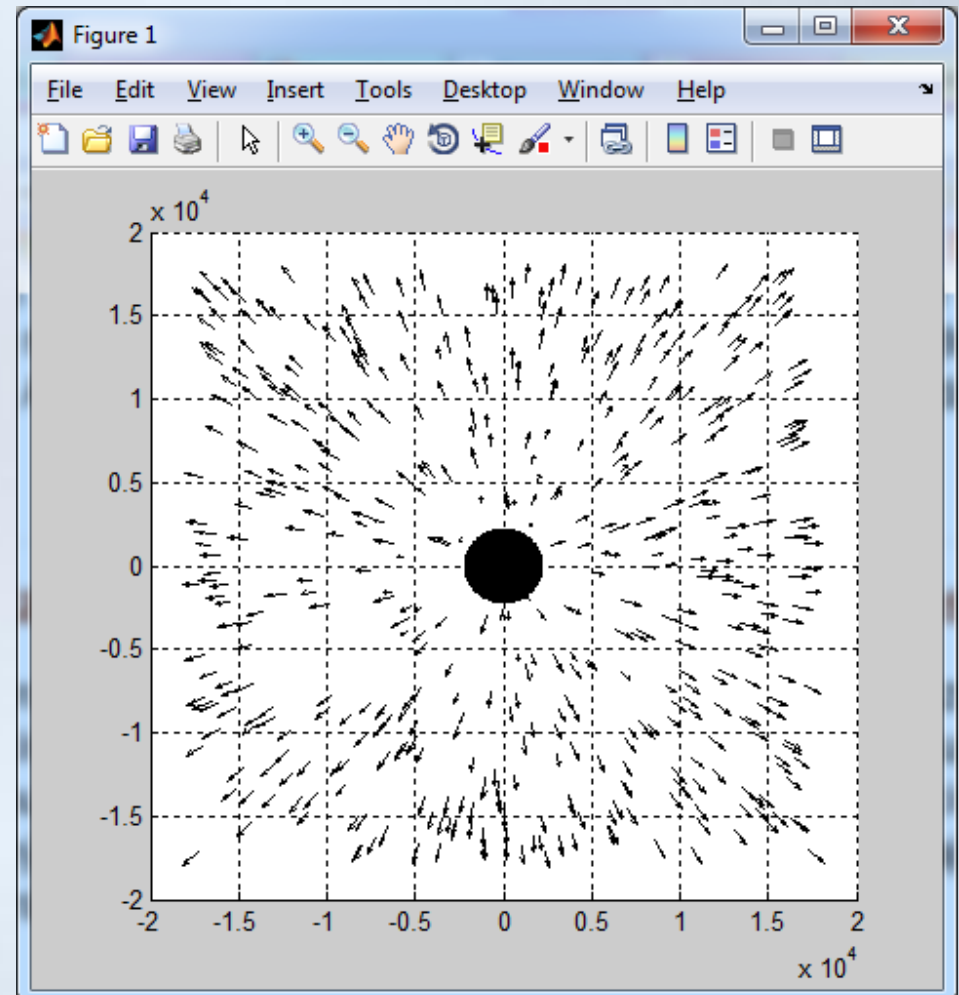
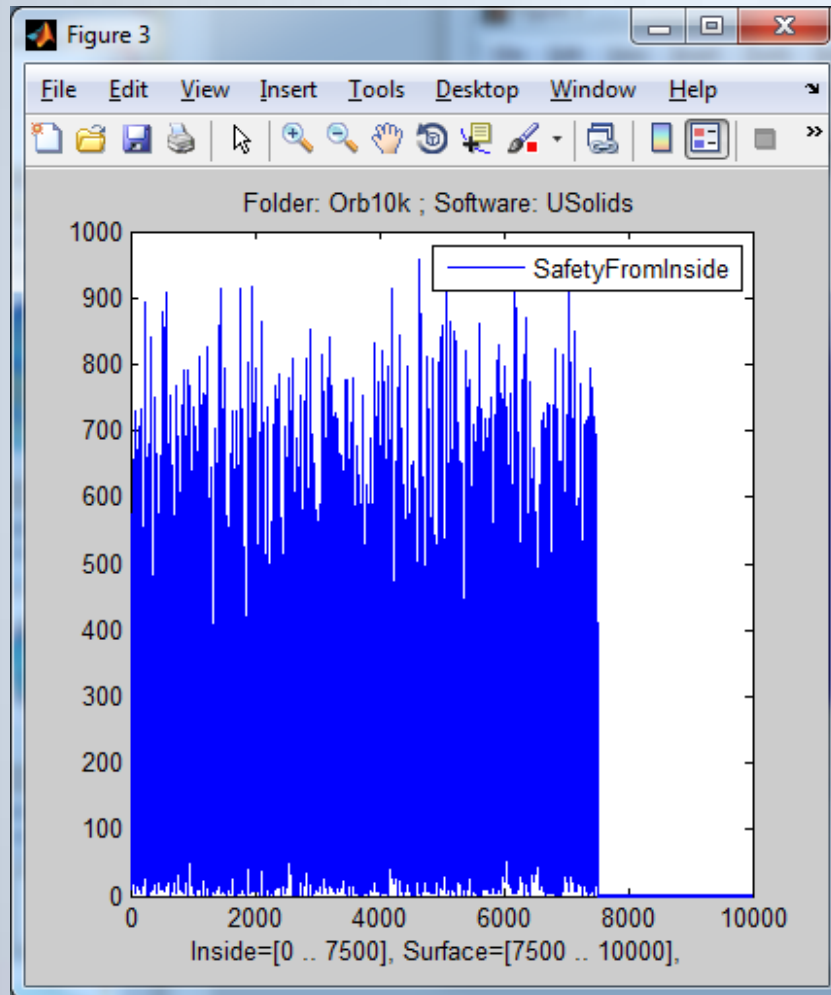
- Tests with solids from three libraries: Geant4, Root and USolids
- Tests with pre-calculated, randomly generated sets of points and vectors
- Storing of results data sets to disk
- Measurement of performance
- Support for batch scripting
  - Detailed configuration of conditions in the tests
  - Invoking several tests sequentially
- Rich debugging possibilities in Visual Studio

# DAP - Analysis phase

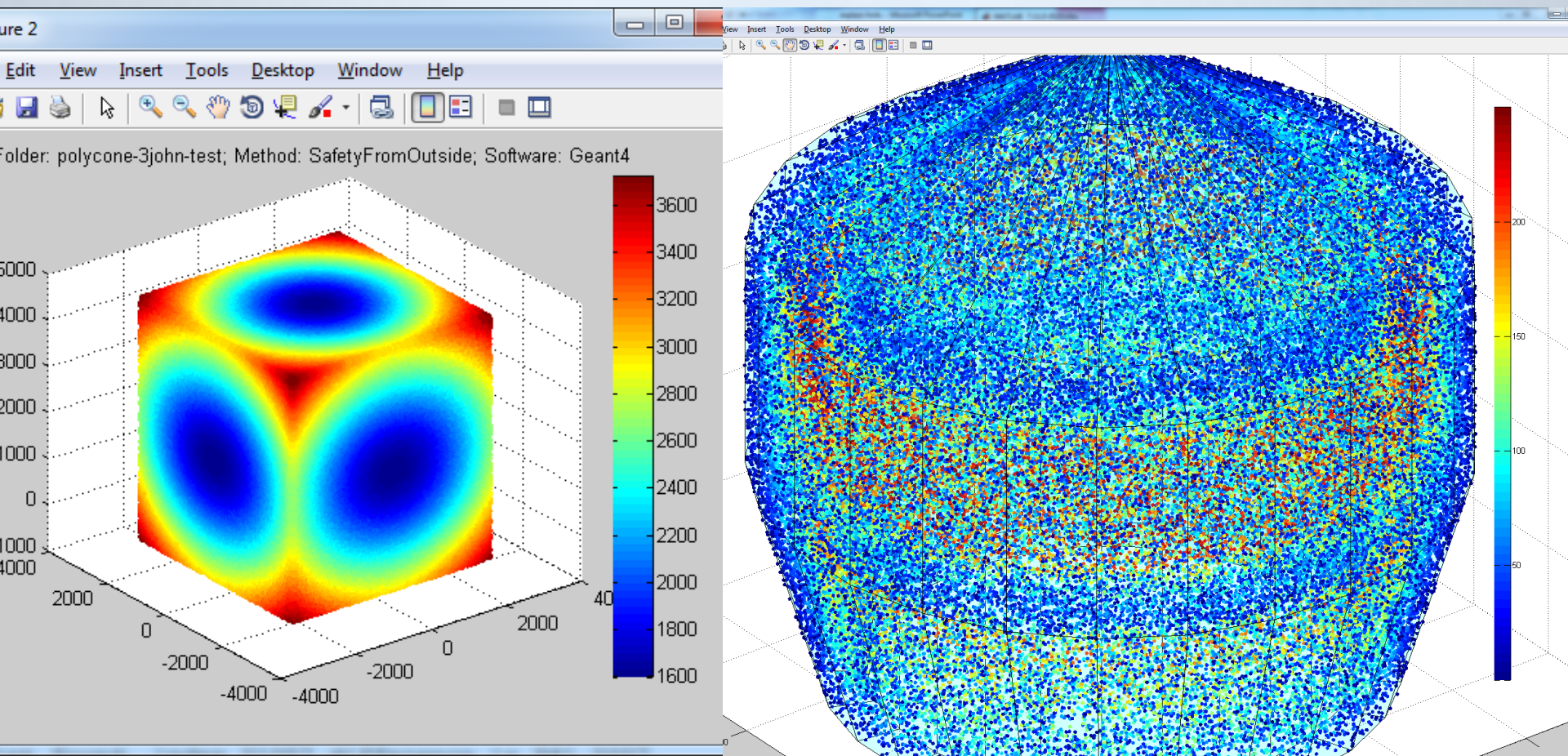


- Visualization of scalar and vector data sets and shapes
- Visual analysis of differences
- Graphs with comparison of performance and scalability
- Inspection of values and differences of data sets

# Visualization of scalar and vector data sets

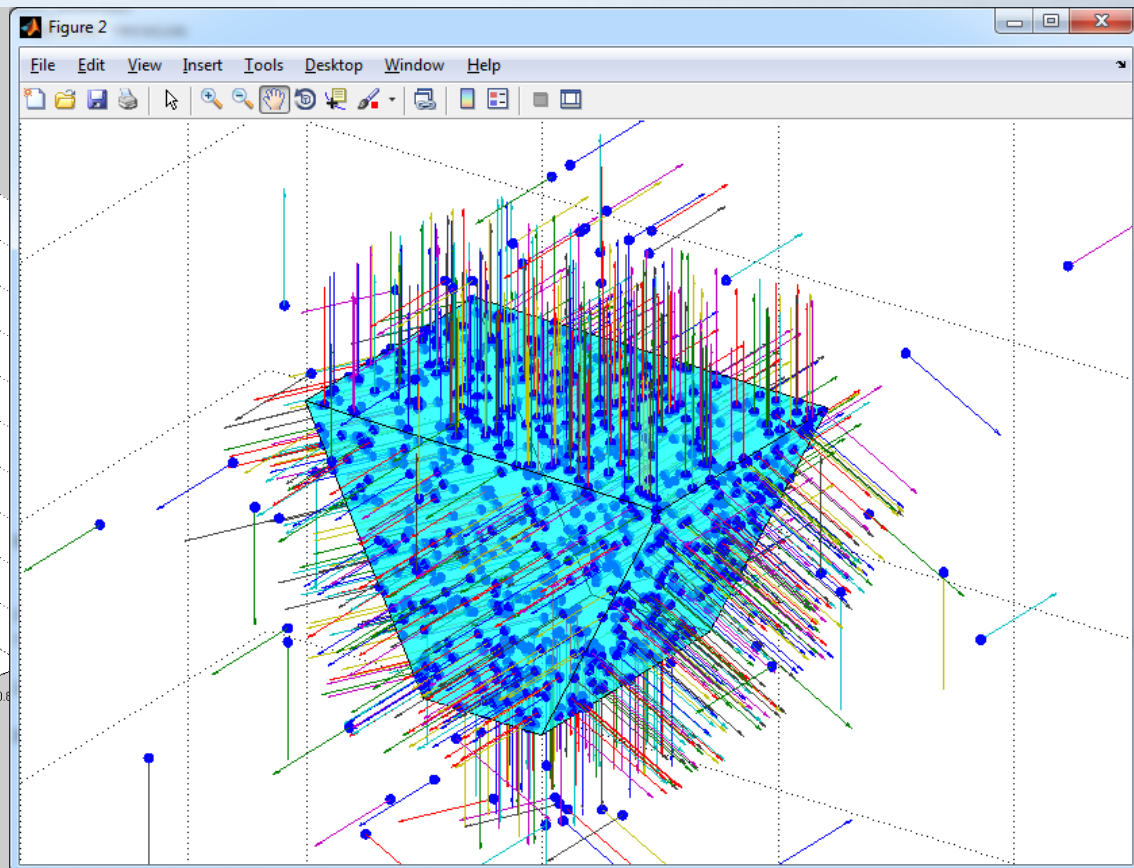
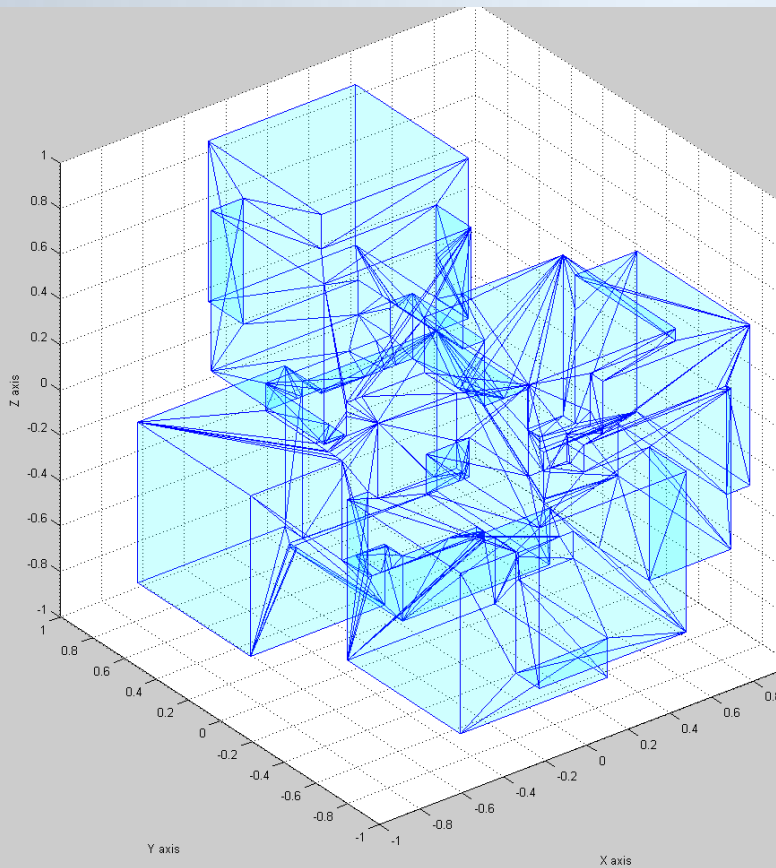


# 3D plots allowing to overview data sets

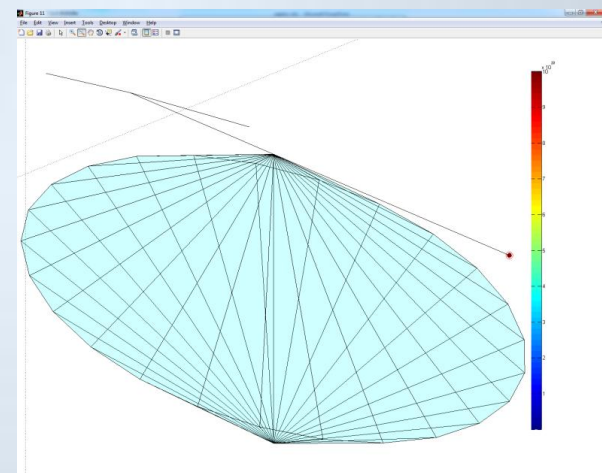
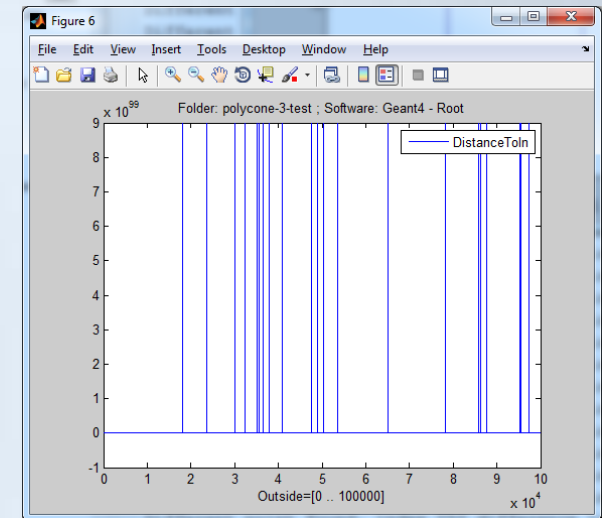
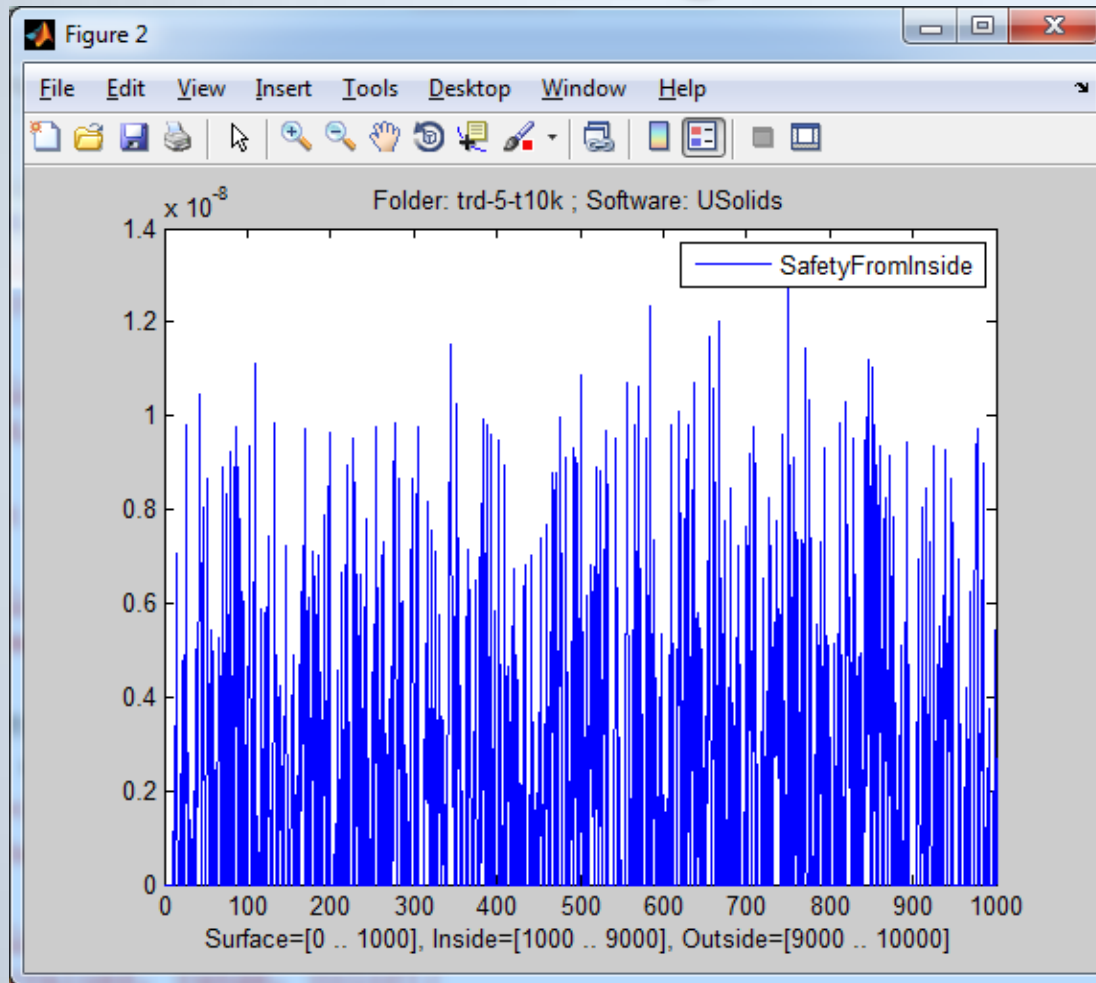




# 3D visualization of investigated shapes

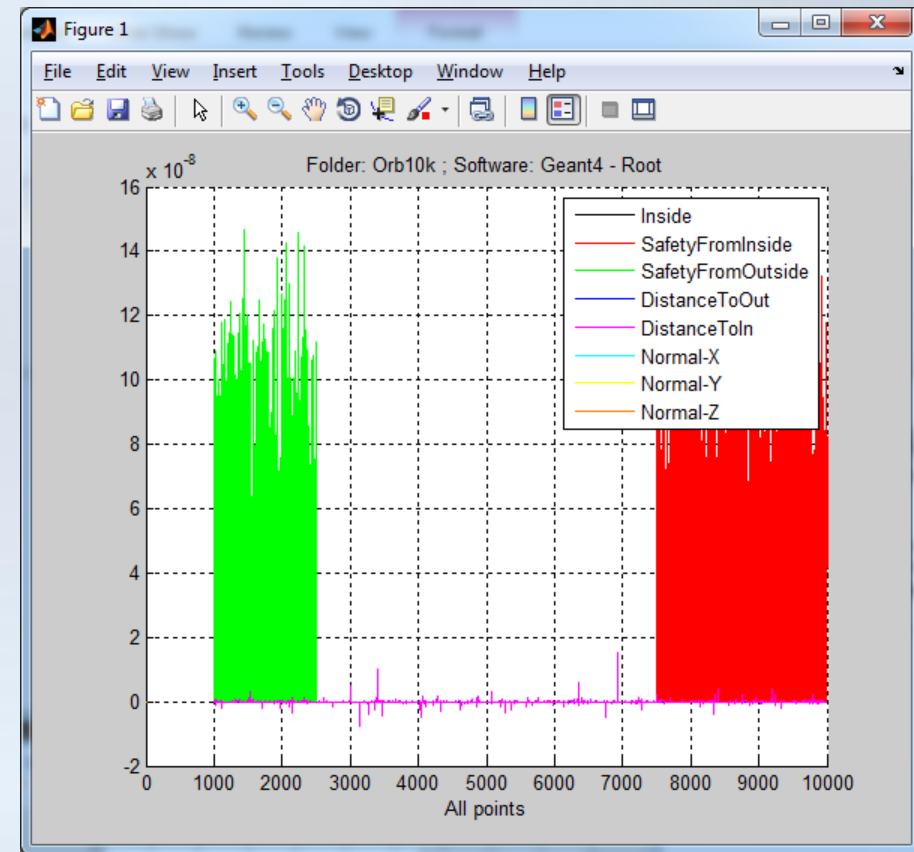
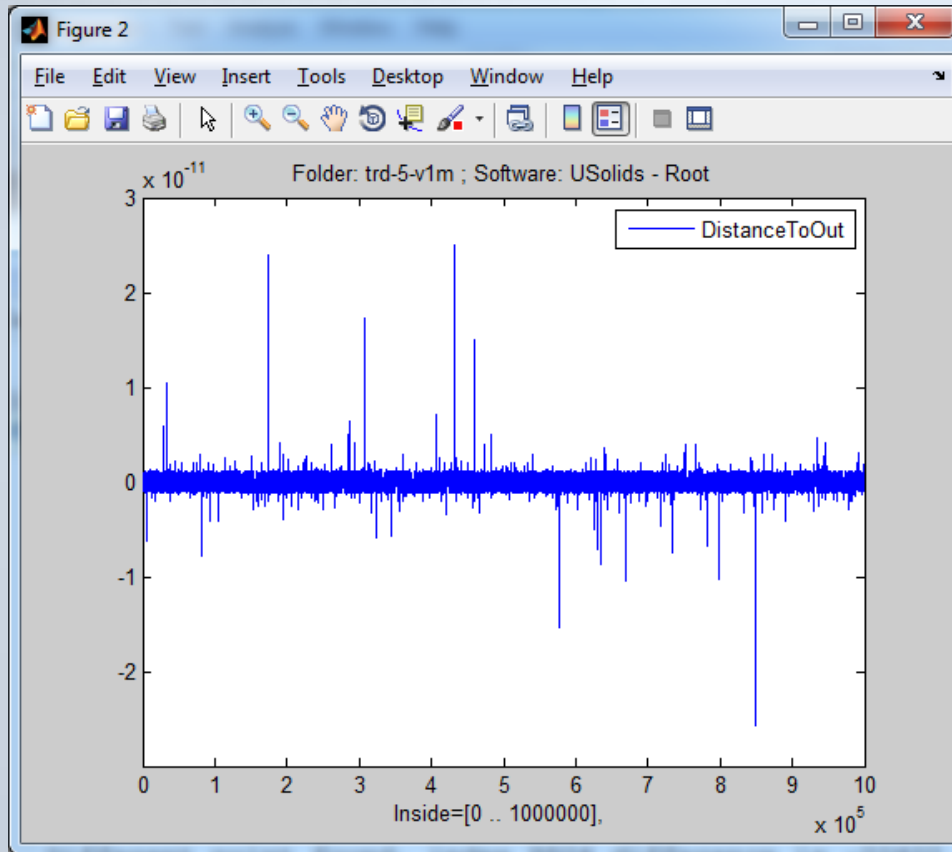


# Support for regions of data, focusing on sub-parts



- Marek Gayer - New software library of geometrical primitives for modelling of solids used in Monte Carlo detector simulations

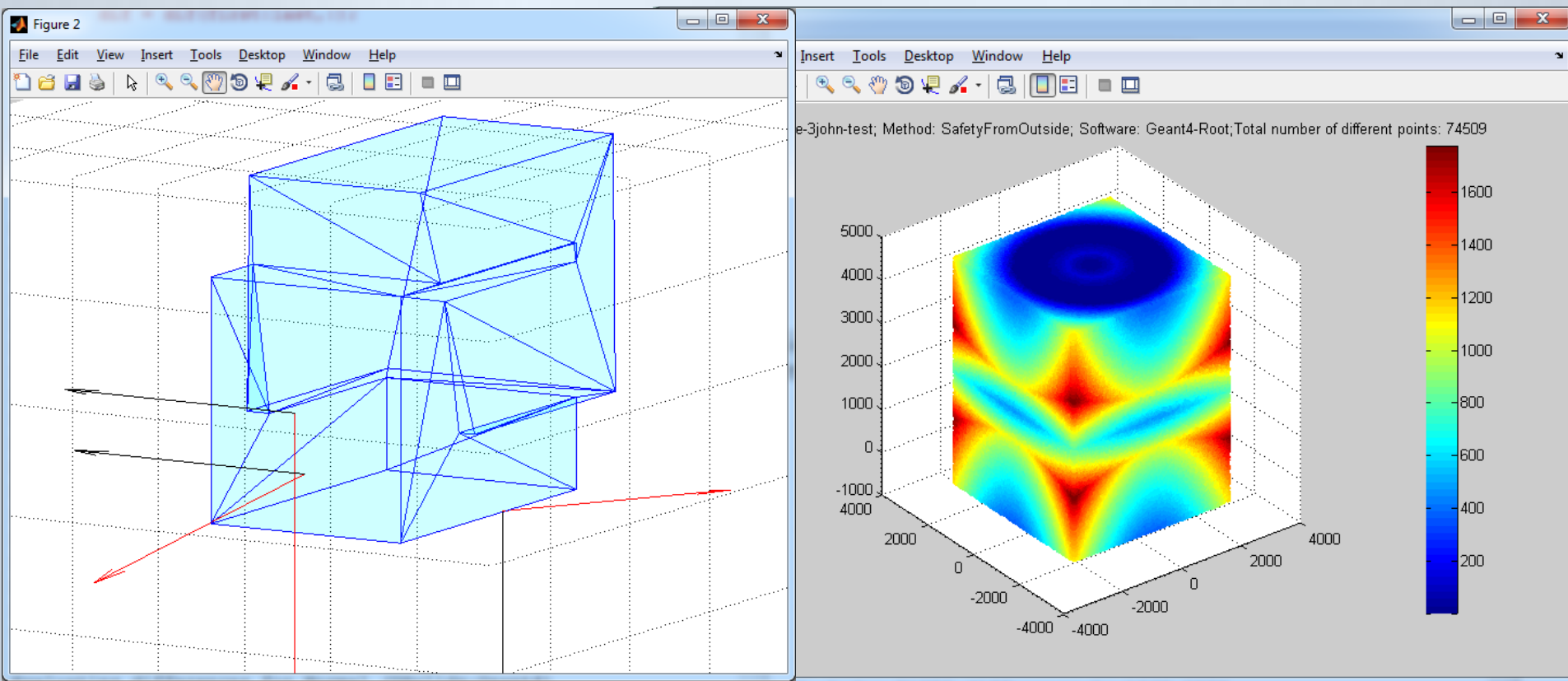
# Visual analysis of differences



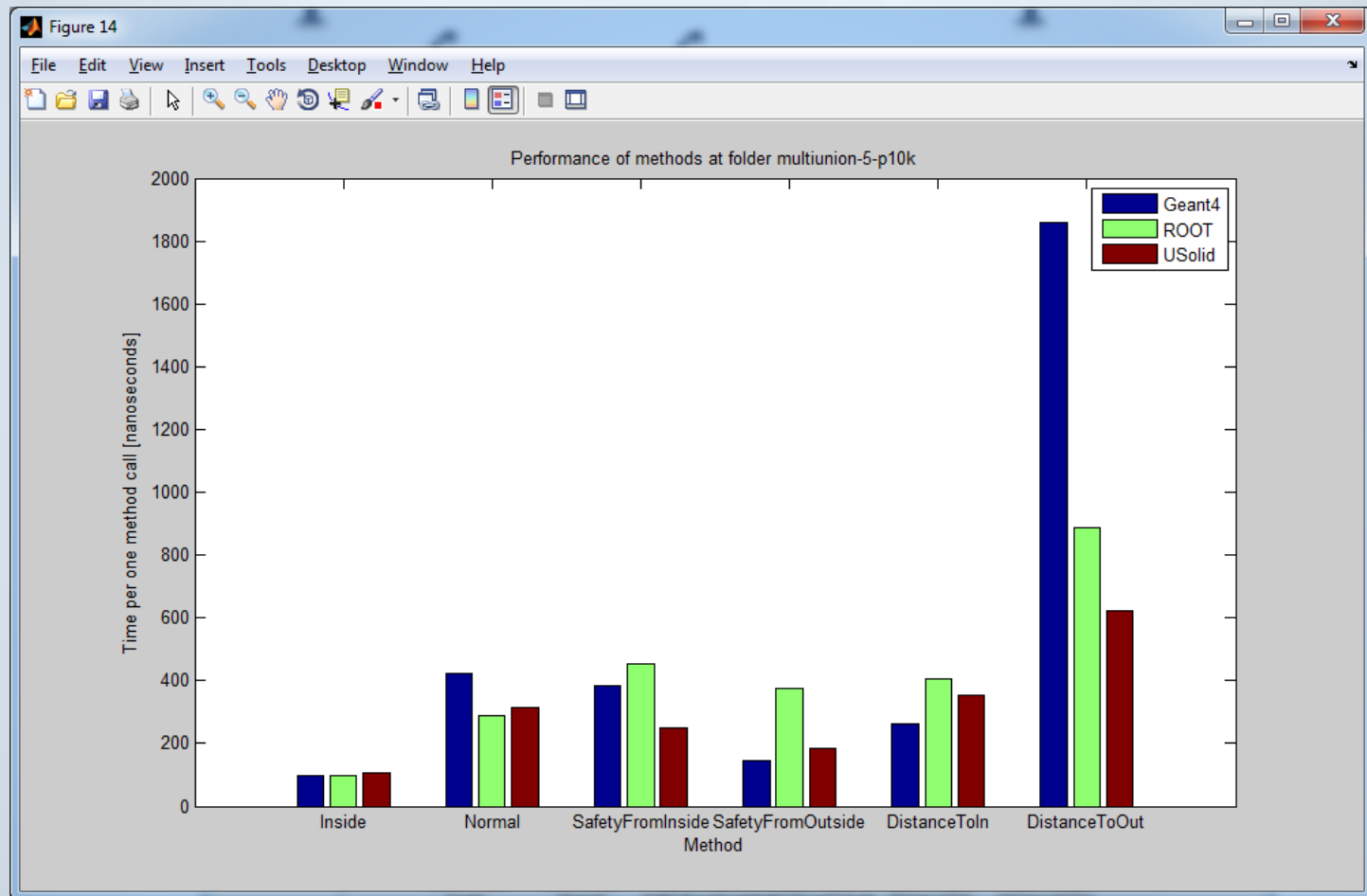
- Marek Gayer - New software library of geometrical primitives for modelling of solids used in Monte Carlo detector simulations



# Visual analysis of differences in 3D



# Graphs with comparison of performance



# Inspection of values and differences of scalar and vector data sets

The image shows the MATLAB 7.11.0 (R2010b) interface. The main window displays two variable editors side-by-side, comparing data from 'NormalUSolids' and 'NormalGeant4'. Both variables are of type '<10000x3 double>'. The data is presented in a table with columns 1, 2, and 3. The values for both variables are identical for the first 15 rows, but differ for the last row (index 16).

The Workspace window on the right shows a list of variables and their dimensions. The Command History window at the bottom right shows the commands executed in the MATLAB session.

**Variable Editor: NormalUSolids**

	1	2	3
1	-0.4084	-0.5636	0.7180
2	-0.9957	0.0704	0.0598
3	0.4470	-0.8910	-0.0797
4	-0.3520	0.7533	-0.5555
5	-0.0066	0.5707	0.8211
6	0.7121	-0.7021	-0.0017
7	-0.6749	0.2703	0.6866
8	0.4903	-0.4064	0.7710
9	0.7487	0.2883	-0.5969
10	0.9404	0.2728	-0.2033
11	-0.4545	-0.6434	-0.6160
12	-0.9539	-0.2938	-0.0611
13	0.8185	-0.5680	0.0861
14	-0.0035	-0.9790	-0.2039
15	-0.0594	0.7941	0.6049
16	0.7175	0.4430	0.5377

**Variable Editor: NormalGeant4**

	1	2	3
1	-0.4084	-0.5636	0.7180
2	-0.9957	0.0704	0.0598
3	0.4470	-0.8910	-0.0797
4	-0.3520	0.7533	-0.5555
5	-0.0066	0.5707	0.8211
6	0.7121	-0.7021	-0.0017
7	-0.6749	0.2703	0.6866
8	0.4903	-0.4064	0.7710
9	0.7487	0.2883	-0.5969
10	0.9404	0.2728	-0.2033
11	-0.4545	-0.6434	-0.6160
12	-0.9539	-0.2938	-0.0611
13	0.8185	-0.5680	0.0861
14	-0.0035	-0.9790	-0.2039
15	-0.0594	0.7941	0.6049
16	0.7175	0.4430	0.5377

**Workspace**

Name	Value	Min	Max
NormalDirections	<1000x3 double>	-0.9994	0.9993
NormalGeant4	<10000x3 double>	-1	1
NormalPoints	<1000x3 double>	-1.2816	3.7816
NormalQuads	<12x4 double>	1	16
NormalRoot	<1000x3 double>	-1	1
NormalUSolids	<10000x3 double>	-1	1
NormalVertices	<267x3 double>	-1000	1000

**Command History**

```
%-- 12/03/2012 15:35 --%
sbtgenpolycones
sbtyscale
sbtperfb
sbtplot3d(Inside, USolids);
sbtperfb
sbtyscale
sbtplot(SafetyFromOutside, USolids);
sbtplot(SafetyFromOutside, Geant4);
sbtplot(SafetyFromOutside, Geant4);
sbtplot(Normal, Geant4, USolids);
```

**Command Window**

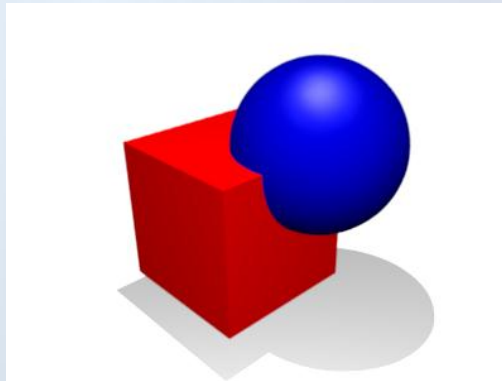
```
New to MATLAB? Watch this Video, see Demos, or read Getting Started.
Different point found, index 988 difference is -1
Different point found, index 989 difference is -1
Different point found, index 991 difference is 1
Different point found, index 992 difference is 1
Total number of different points: 190
fx >>
```

# New Multi-Union solid

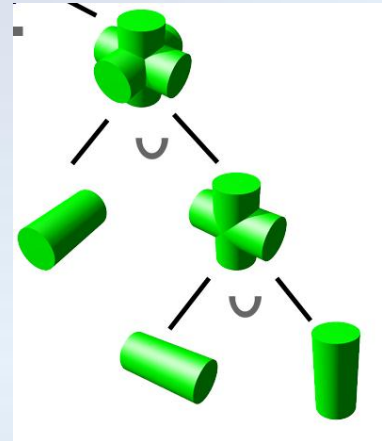
...

# Boolean Union solids

- Existing CSG Boolean solids (Root and Geant4) represented as binary trees
  - To solve navigation requests, most of the solids composing a complex one have to be checked
  - Scalability is typically linear => low performance for solids composed of many parts



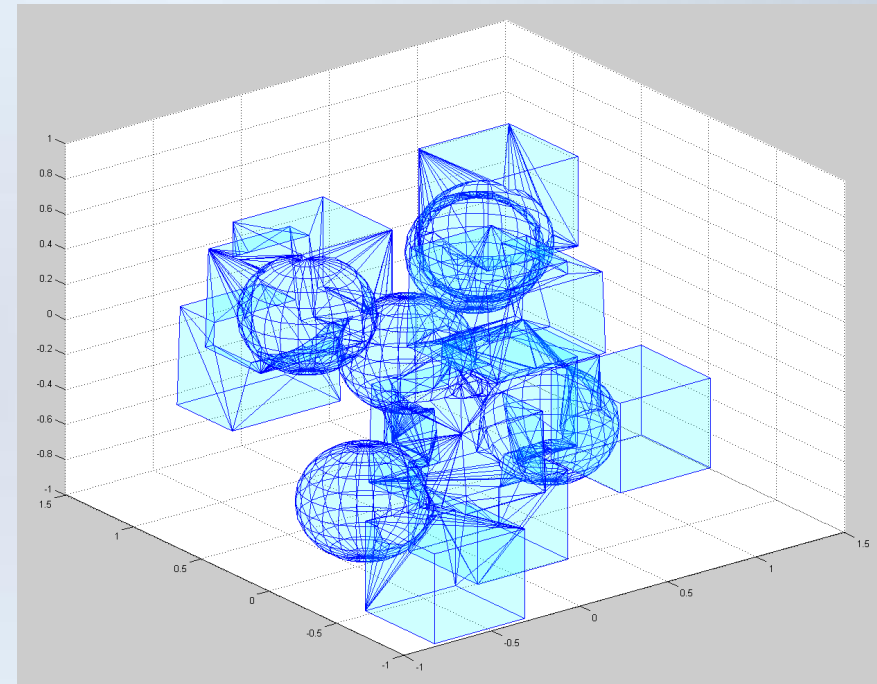
Boolean Union solid:  
is composite of two solids, either primitive or Boolean



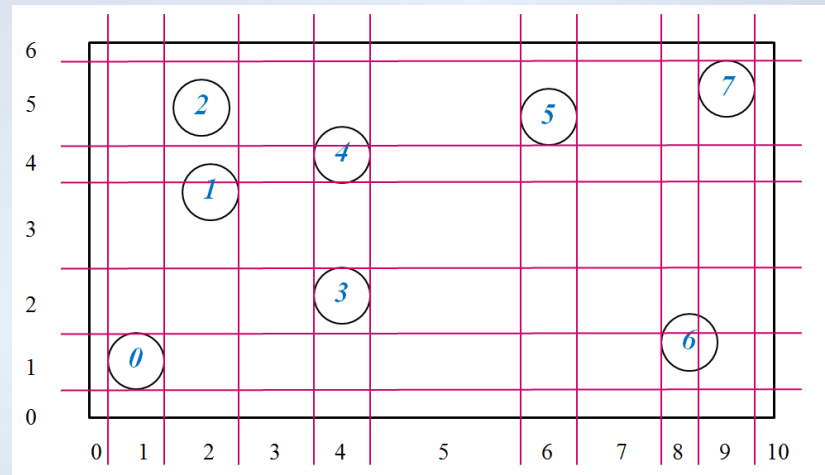
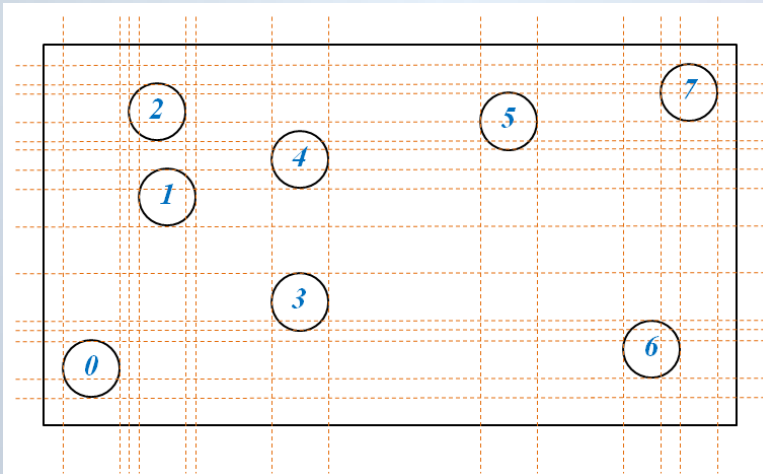
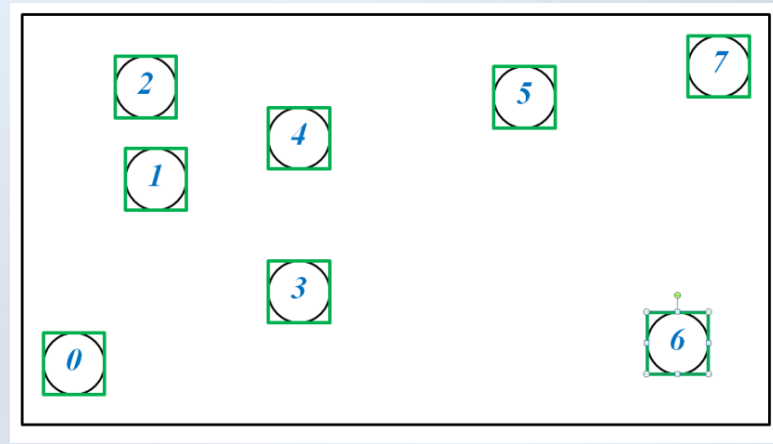
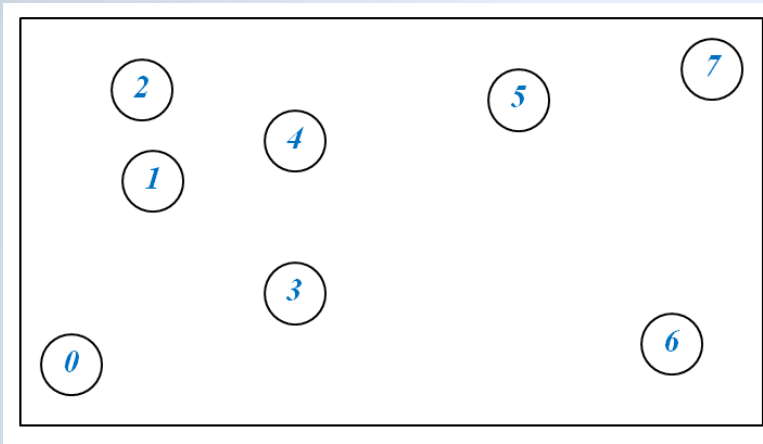
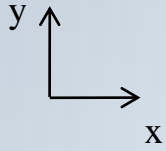
[The pictures were produced by users of Wikipedia "Captain Sprite" and "Zottie" and are available under Creative Commons Attribution-Share Alike 3.0 Unported license ]

# Multi-Union solid

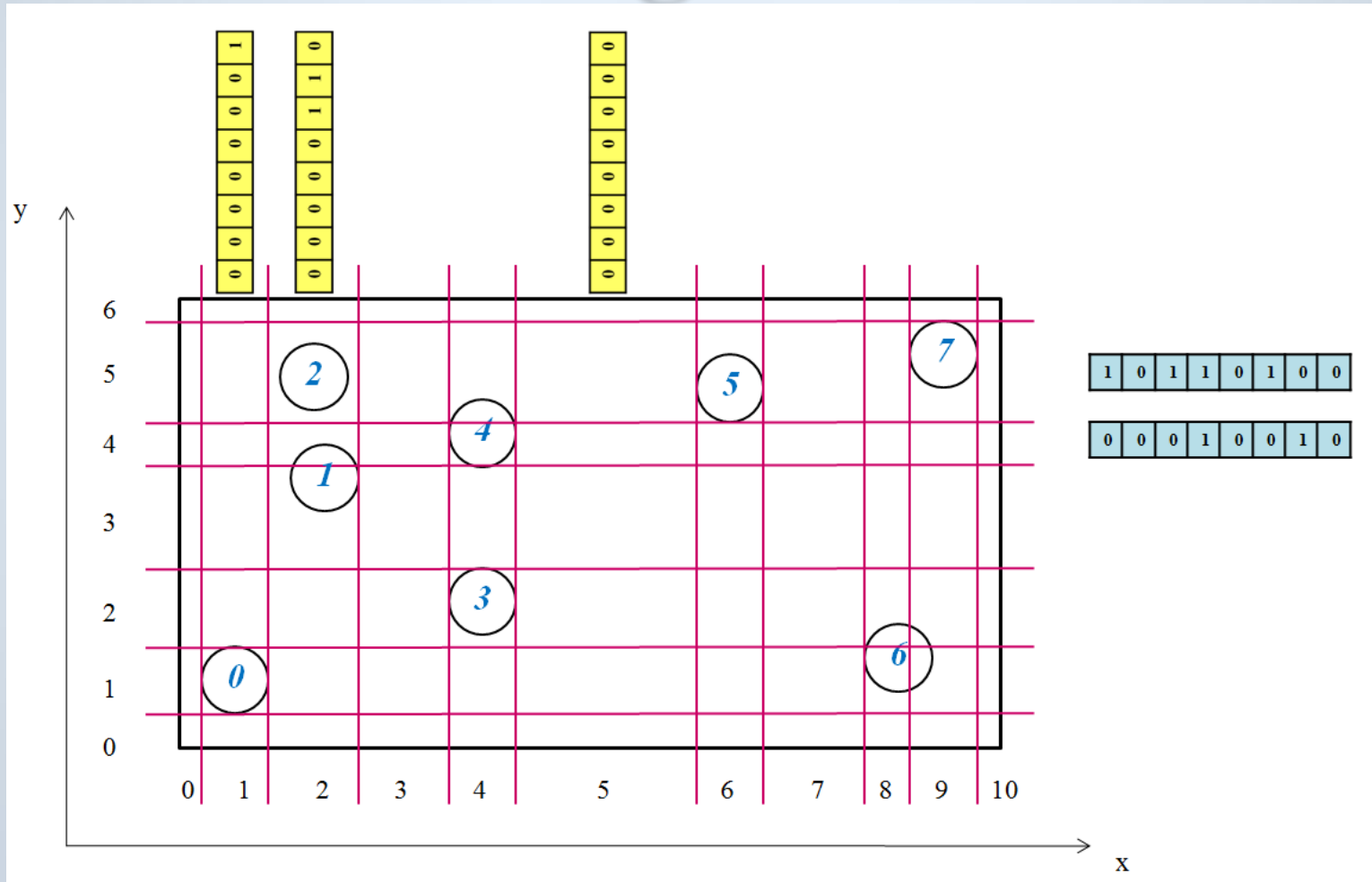
- We implemented a new solid as a union of many solids using voxelization technique to optimize the speed
  - 3D space partition for fast localization of components
  - Aiming for a  $\log(n)$  scalability
- Useful also for several complex composites made of many solids with regular patterns



# 1. Create voxel space (2D simplification)

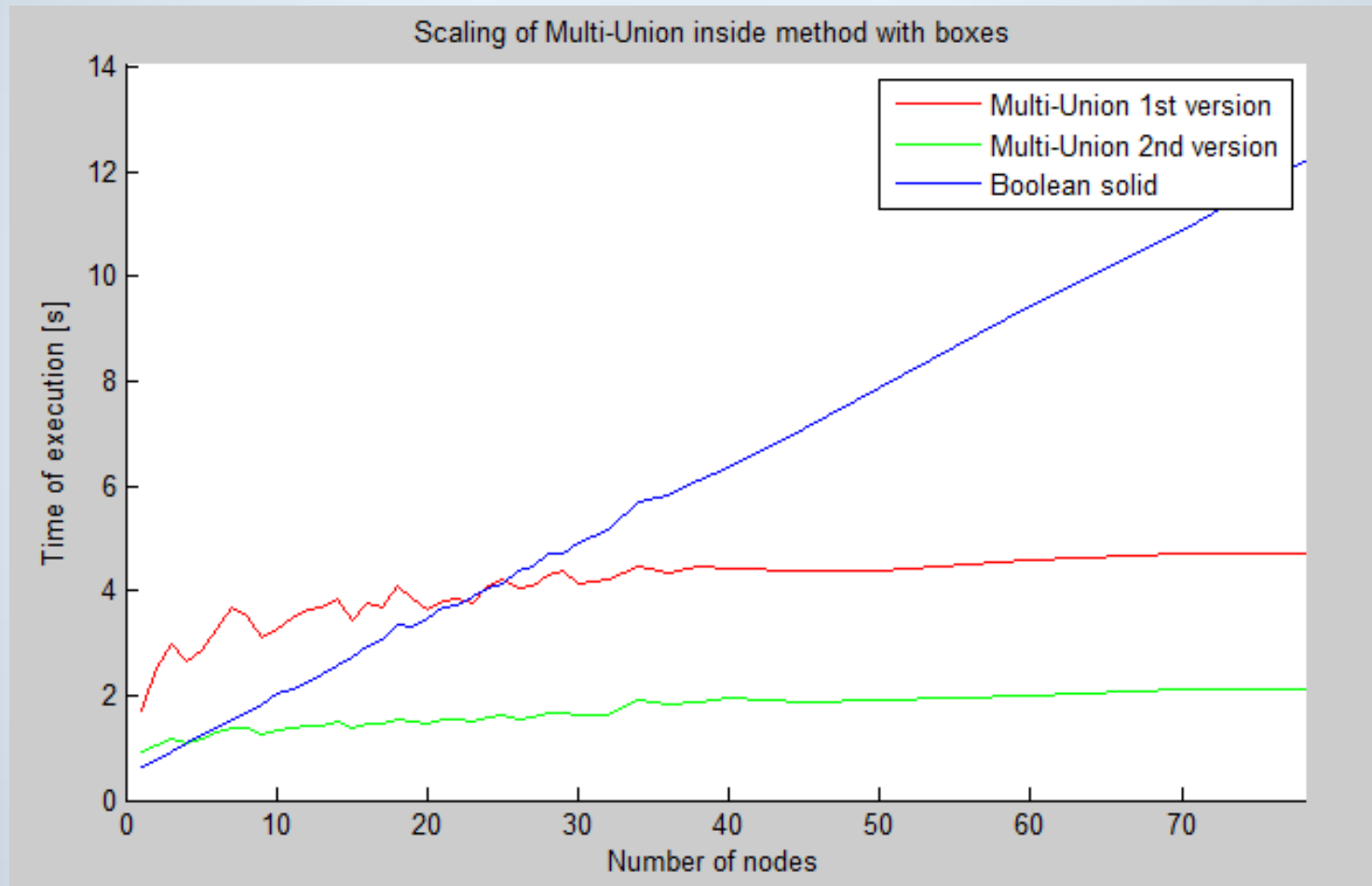


## 2. Usage of bit masks for storing voxels

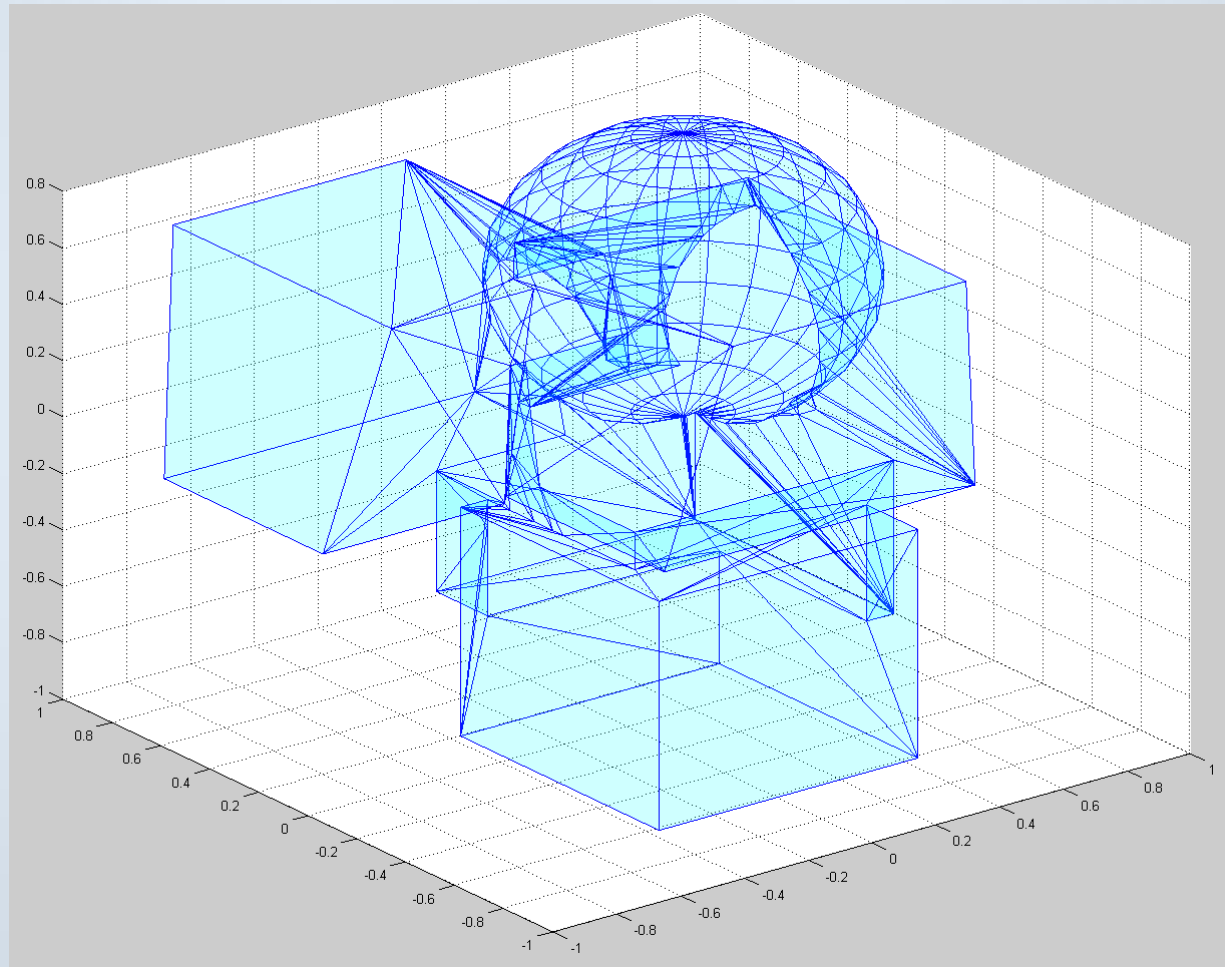




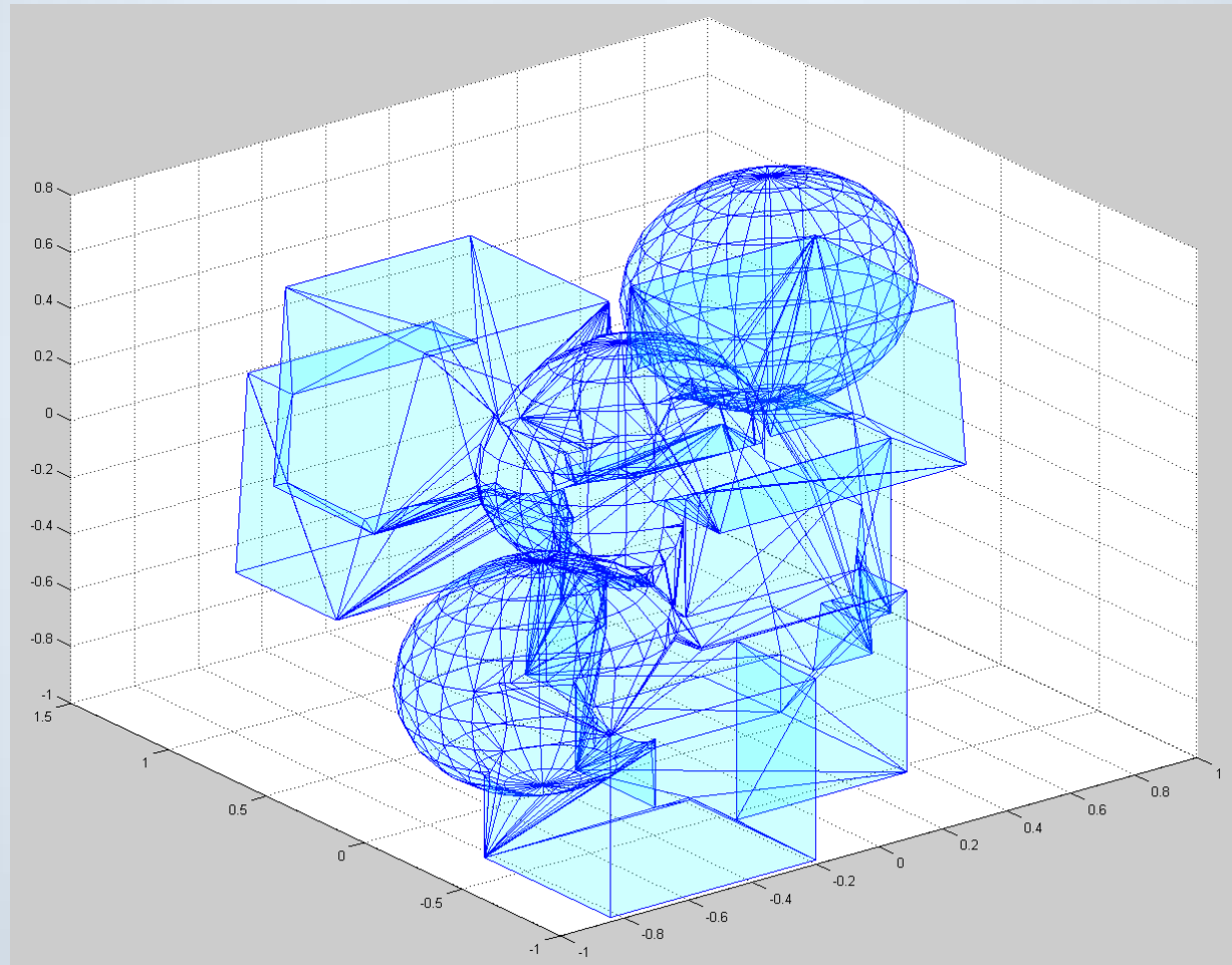
# Scaling of Multi-Union vs. Boolean solid



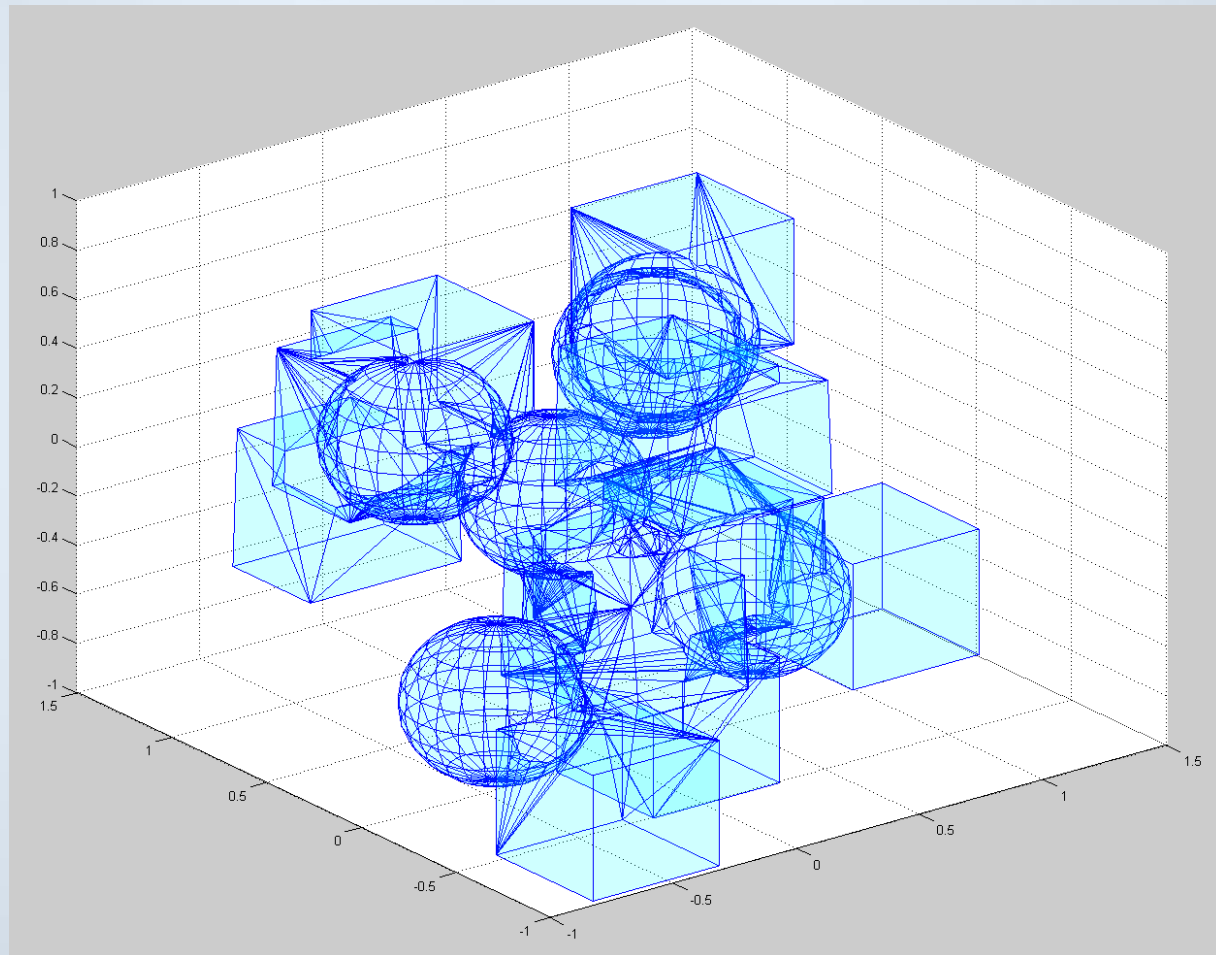
# Test union solids for scalability measurements



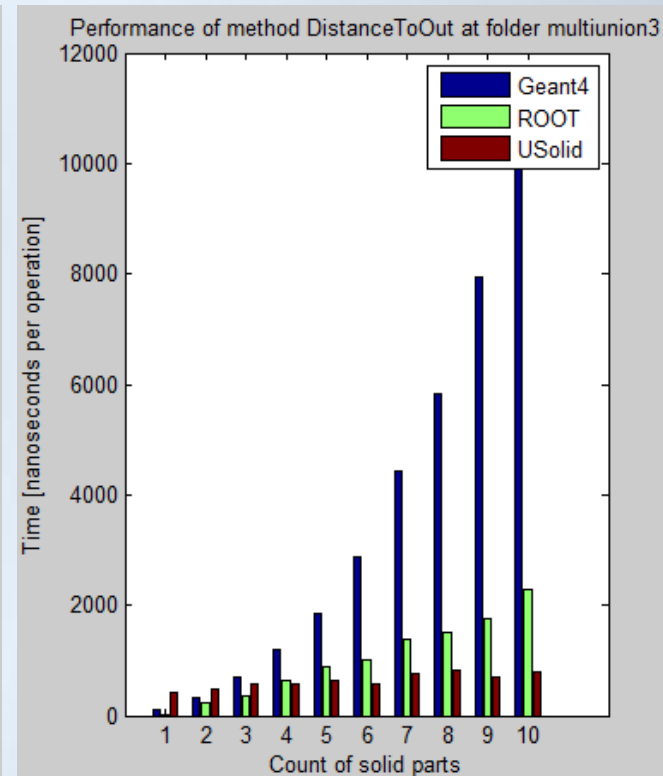
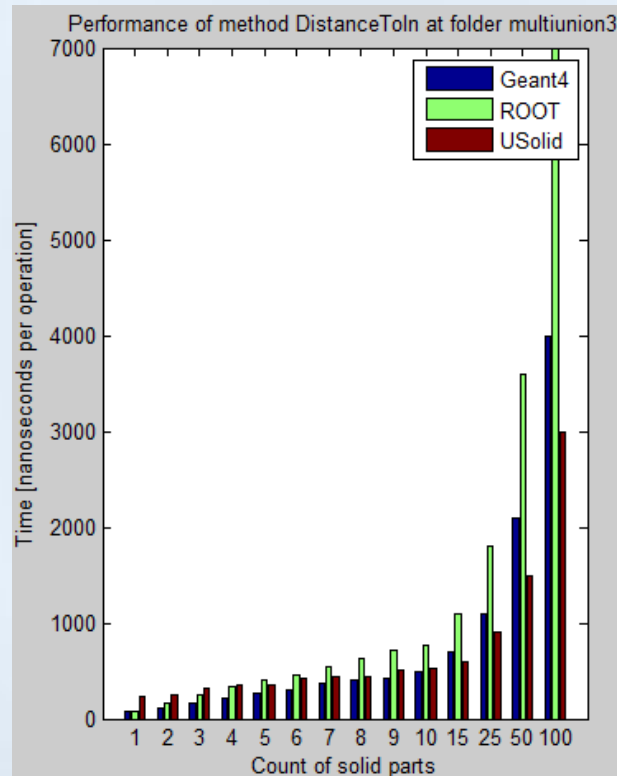
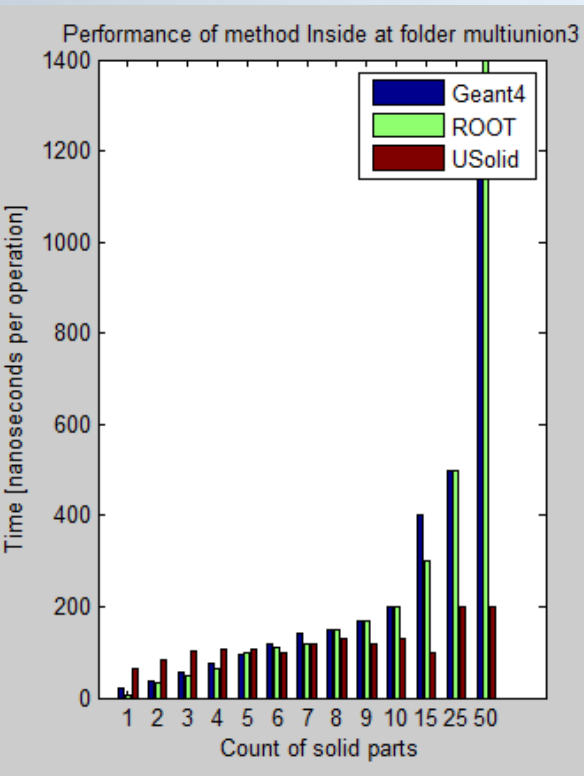
# Test union solids for scalability measurements



# Test union solids for scalability measurements



# The most performance critical methods



# Status of work

- ✓ Types and USolid interface are defined
- ✓ Bridge classes defined and implemented for both Geant4 and Root
- ✓ Testing suite defined and deployed
- ✓ Implementation of Multi-Union solid completed and performance optimized
- ✓ Started implementation of primitives:
  - ✓ First implementation of Box, Orb (simple full sphere) and Trd (simple trapezoid)

# Future work

- Give priority to the most critical solids and those where room for improvement can be easily identified
- Systematically analyze and implement remaining solids in the new library

# Thank you for your attention.



## Questions ?