Visual debugging with SimpleGeo

Chris Theis

Overview

- SimpleGeo’s visual debugger

- Running a few examples

Geometry errors

Two types of errors

- Overlapping regions → FLUKA won’t crash but the results may be not what you had intended.

- Undefined regions → program crash

SimpleGeo’s approach

Basic idea is similar to FLUKA:

Test if particles at specified positions encounter an unambiguous and well-defined region description

The “black magic”:

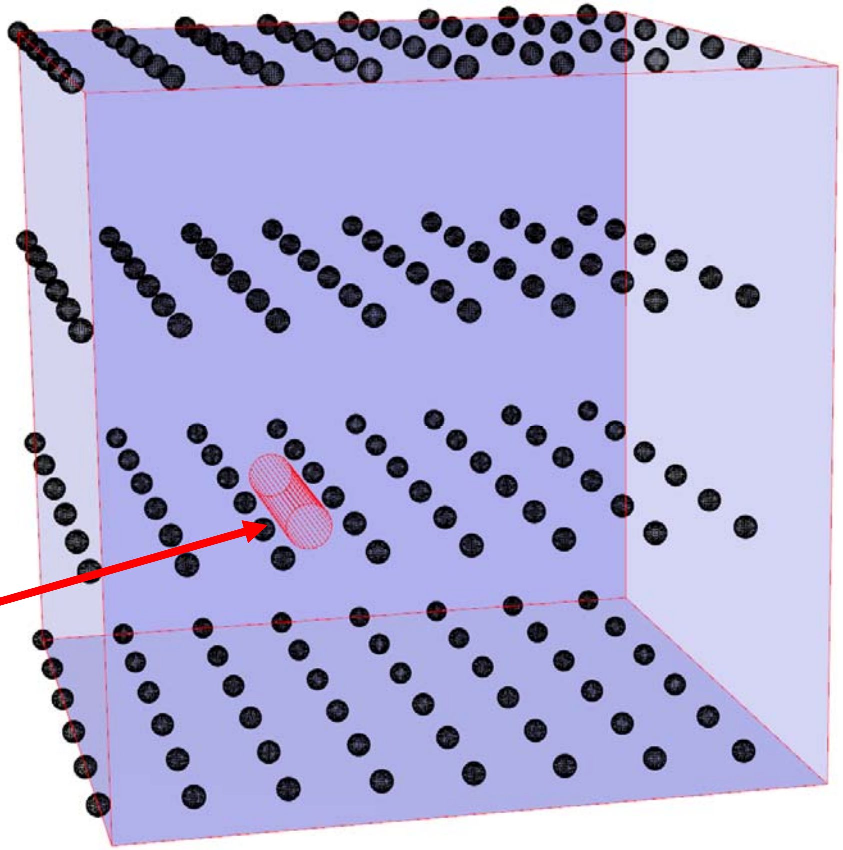
How do you define those positions that should be tested?

ATTENTION: The debugger checks the geometry only!!! It does not include diagnostics for misaligned values in the input file!

Grid debugging

1. Definition of a region of interest (ROI) via XMin, XMax, YMin, YMax, ZMin, ZMax

2. Specification of how many steps should be taken in each direction



The problem:

Errors located in between the equidistant step sizes cannot be found by definition!

Other methods

- ## Monte Carlo sampling of the region

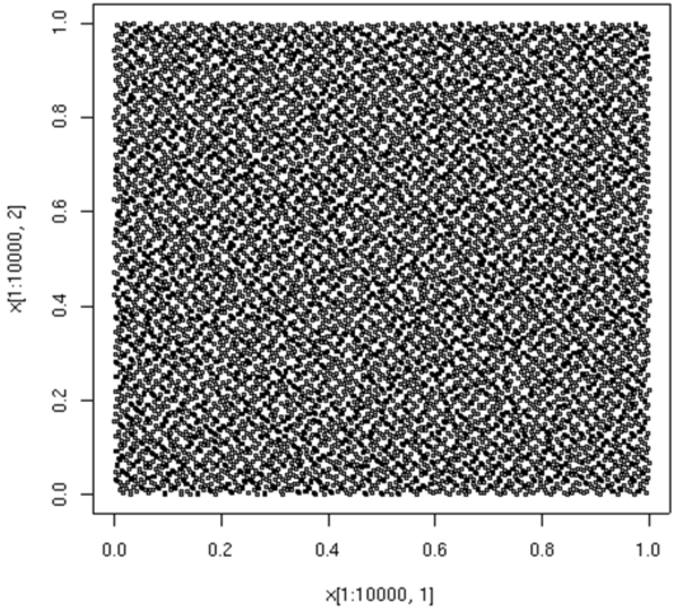
  Due to the randomness the probability to find very small errors exists.

  Drawback: Tendency to cluster at low sampling rates. Uniformity is reached only by a large number of samples.

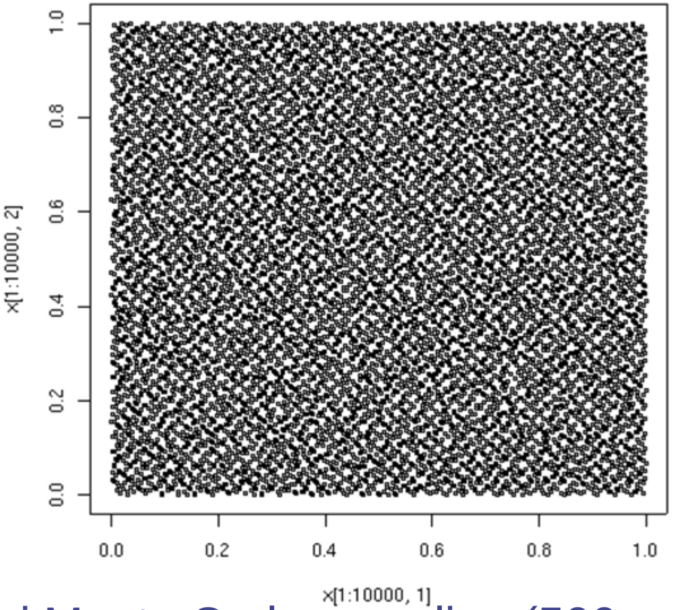
- ## Quasi-Monte Carlo sampling of the region

  Faster convergence towards uniformity.

  Drawback: Very complex mathematics involved (but this can be compensated for by highly optimized sampling routines).



Monte Carlo sampling (500 points)



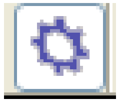
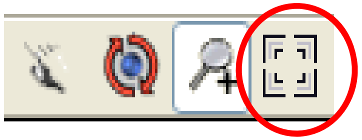
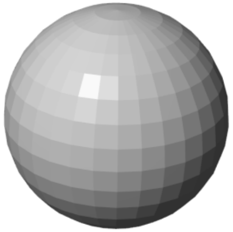
Quasi Monte Carlo sampling (500 points)

* Images courtesy of R. Dodier, Wikipedia
http://en.wikipedia.org/wiki/Low-discrepancy_sequence

Algorithms available in SimpleGeo

- Grid sampling
  - Equidistant points in each direction X, Y, Z
- Stochastic sampling (Monte Carlo method)
  - Uniform distribution of points in a box
- Importance sampling
  - Exponential distribution of points in a sphere
- Quasi Monte Carlo (recommended)
  (low discrepancy, jittered stratified, latin hypercube)
  - Quasi-random distribution of points in a box

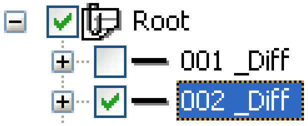
A first example

1. Import ngen_e1.inp from the folder data/Debugging
2. Activate the automatic build on the toolbar 
3. Reset the camera clicking on the toolbar 
4. You will now see the surrounding blackhole 
5. Turn off the visibility of the blackhole

   (Click on the checkbox next to the region named 001_Diff or click on the gray sphere and press the Space key)

A first example

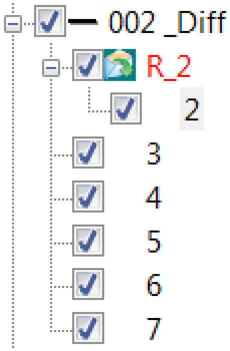
6. Select region 002_Diff in the CSG tree



☑ Root
□ 001 _Diff
☑ 002 _Diff

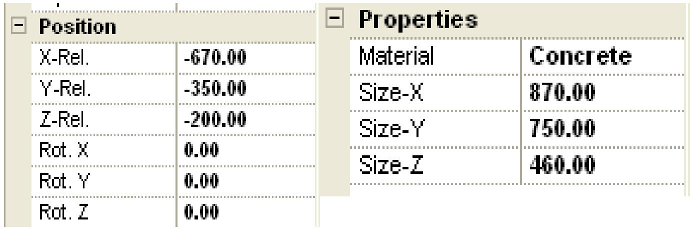
7. We want to debug everything inside this region

8. Expand the region and select the primitive named 2, which is the source primitive of reference node R_2 and surrounds our region of interest (ROI)



☑ 002 _Diff
☑ R_2
☑ 2
☑ 3
☑ 4
☑ 5
☑ 6
☑ 7

9. Note the position & extension – they form our ROI



Position	
X-Rel.	-670.00
Y-Rel.	-350.00
Z-Rel.	-200.00
Rot. X	0.00
Rot. Y	0.00
Rot. Z	0.00

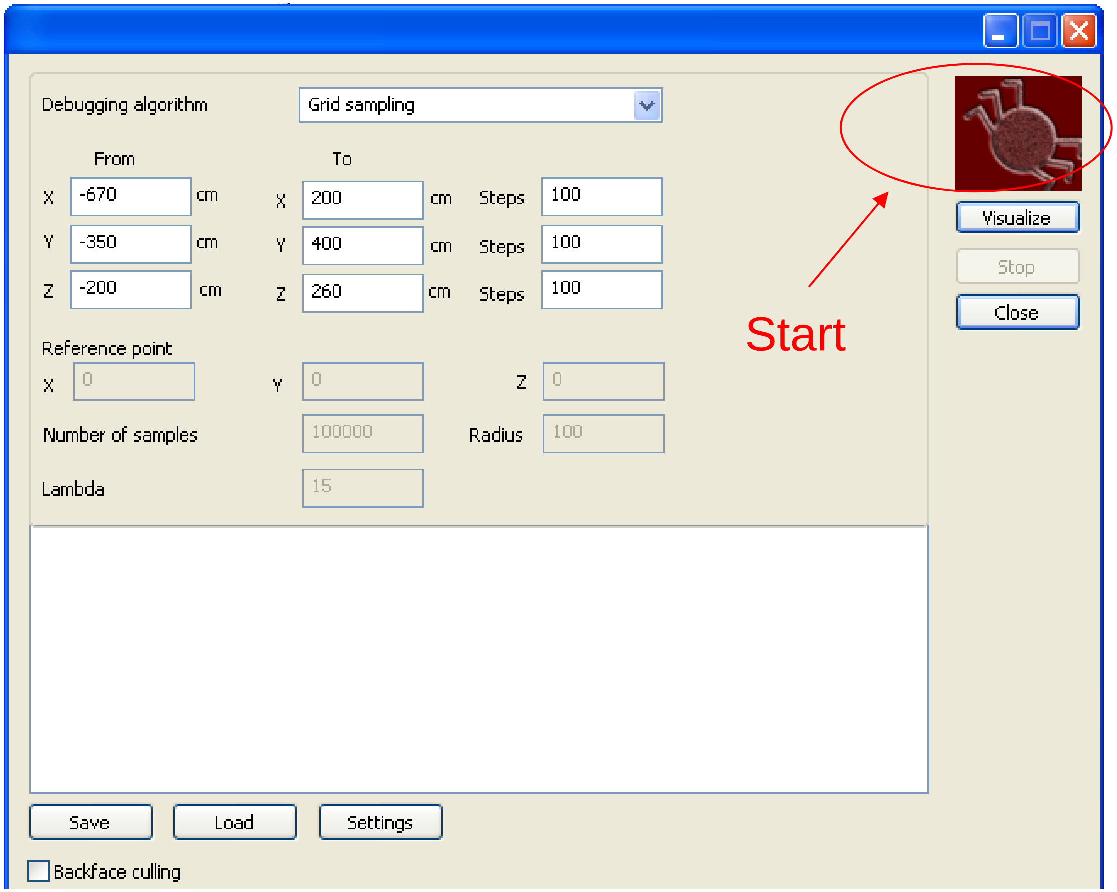
Properties	
Material	Concrete
Size-X	870.00
Size-Y	750.00
Size-Z	460.00

Debug the geometry

10. Open the debugger via the menu “File” - “Debug geometry”

11. Enter the coordinates of our ROI or load them from the file ngen_e1.sdb

12. Click on the bug to start



Debugging algorithm: Grid sampling

	From			To		
X	-670	cm	X	200	cm	Steps 100
Y	-350	cm	Y	400	cm	Steps 100
Z	-200	cm	Z	260	cm	Steps 100

Reference point: X 0, Y 0, Z 0

Number of samples: 100000 Radius: 100

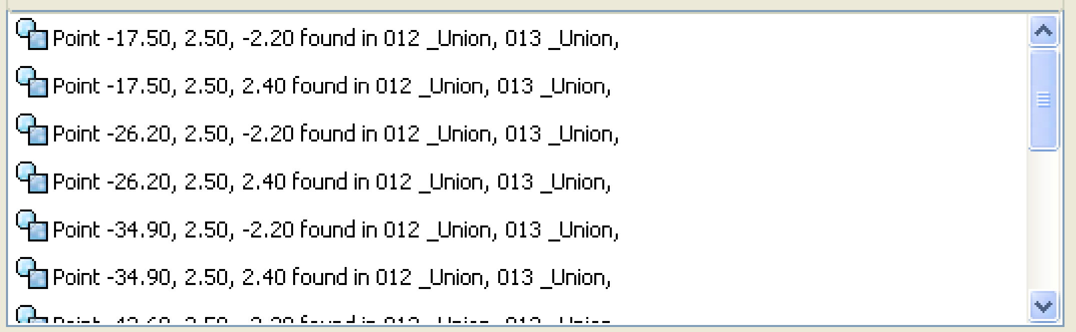
Lambda: 15

□ Backface culling

The result

- Debugging can be stopped anytime. All errors found so far will be displayed.

- After some time the following overlaps will be found:

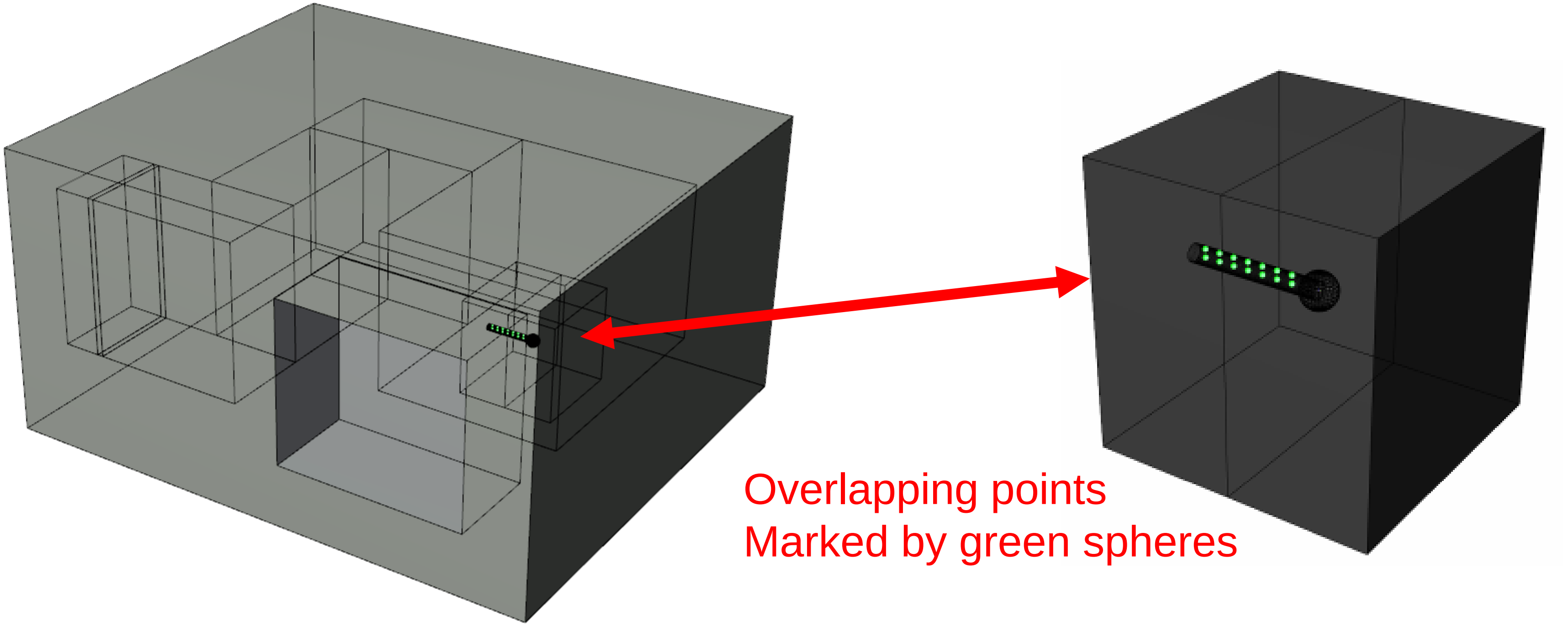


```
Point -17.50, 2.50, -2.20 found in 012 _Union, 013 _Union,
Point -17.50, 2.50, 2.40 found in 012 _Union, 013 _Union,
Point -26.20, 2.50, -2.20 found in 012 _Union, 013 _Union,
Point -26.20, 2.50, 2.40 found in 012 _Union, 013 _Union,
Point -34.90, 2.50, -2.20 found in 012 _Union, 013 _Union,
Point -34.90, 2.50, 2.40 found in 012 _Union, 013 _Union,
Point -43.60, 2.50, -2.20 found in 012 _Union, 013 _Union,
```

- Click on one of the entries and the respective region will be automatically selected in the CSG tree

The result

- Select “View” - “Overlay sketch” to render all regions inside our ROI
- Click on the “Visualize” button in the debugger dialog

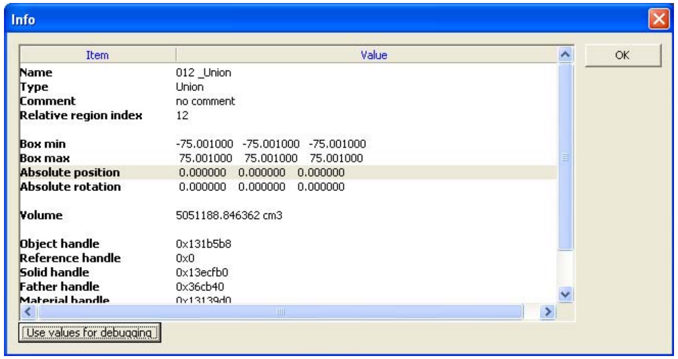


Overlapping points
Marked by green spheres

Taking a closer look

- Regions #12 and #13 are problematic
  ➡ refine the ROI for debugging

1. Select 012_Union in the CSG tree
2. Double click on the right mouse button to open the context menu, and select Node-info.

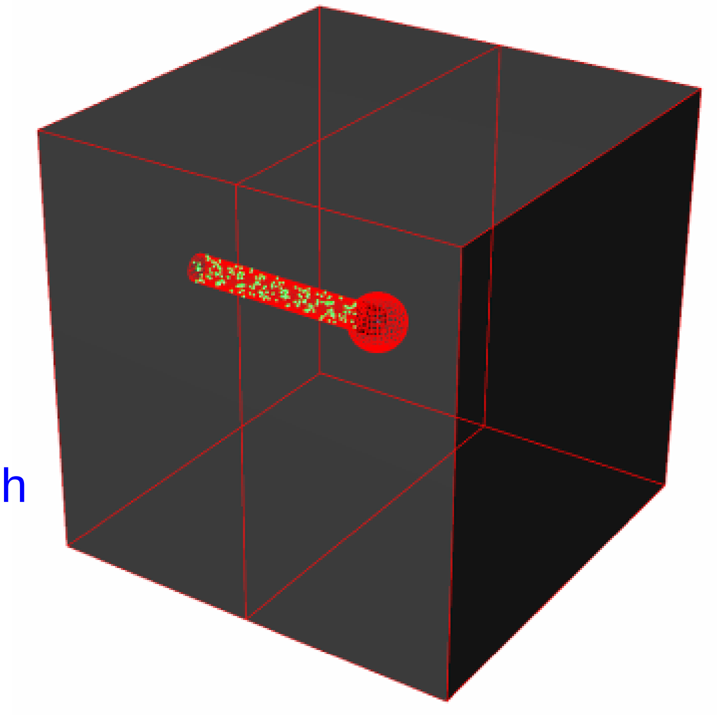


Item	Value
Name	012 _Union
Type	Union
Comment	no comment
Relative region index	12
Box min	-75.001000 -75.001000 -75.001000
Box max	75.001000 75.001000 75.001000
Absolute position	0.000000 0.000000 0.000000
Absolute rotation	0.000000 0.000000 0.000000
Volume	5051188.846362 cm3
Object handle	0x131b5b8
Reference handle	0x0
Solid handle	0x13ecfb0
Father handle	0x36cb40
Material handle	0x13139d0

1. The values of Box min & Box max are the extensions of the bounding box.
2. Click on “Use values for debugging” to transfer the bounding box values to the debugger.

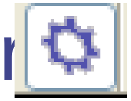
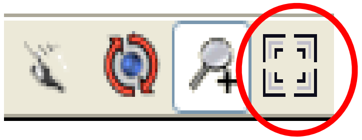
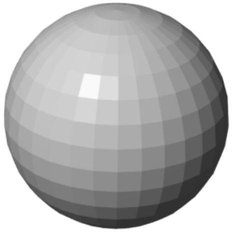
Taking a closer look

5. Turn off the visibility of all regions except 012_Union and 013_Union
6. Select Low discrepancy or Latin Hypercube as the debugging algorithm and start the debugger.



There is an error in the region description of region # 12, which lacks a cut plane.

A second example

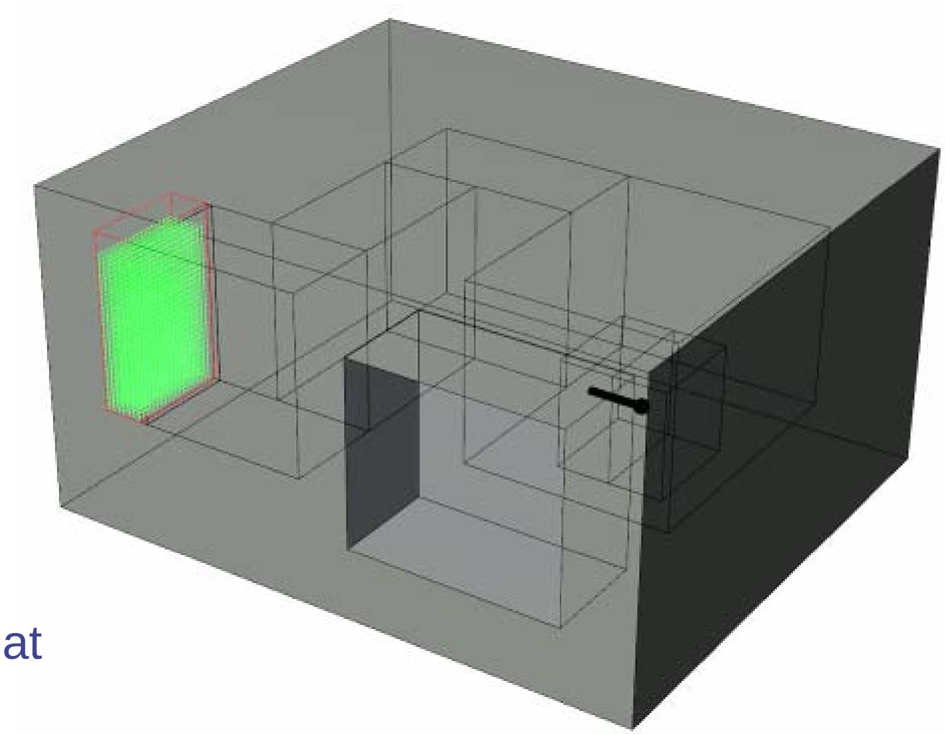
1. Close the debugger, create a new scene and import ngen_e2.inp from the folder data/Debugging
2. Activate the automatic build on the toolbar 
3. Reset the camera clicking on the toolbar 
4. You will now see the surrounding blackhole 
5. Turn off the visibility of the blackhole

   (Click on the checkbox next to the region named 001_Diff or click on the gray sphere and press the Space key)

A second example

6. Load the debug parameters from ngen_e2.sdb
7. Start the debugger

A huge amount of overlaps is found in 007_Int and 008_Int.



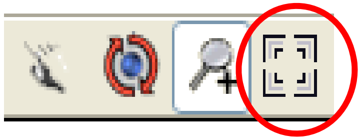
Hint: Select 007_Int in the CSG tree and then select 008_Int. You will see that both have the same red contours. This means that they are identical!

A third example

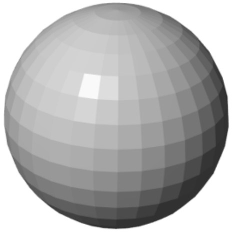
1. Close the debugger, create a new scene and import ngen_e4.inp from the folder data/Debugging

2. Activate the automatic build on the toolbar

3. Reset the camera clicking on the toolbar



4. You will now see the surrounding blackhole



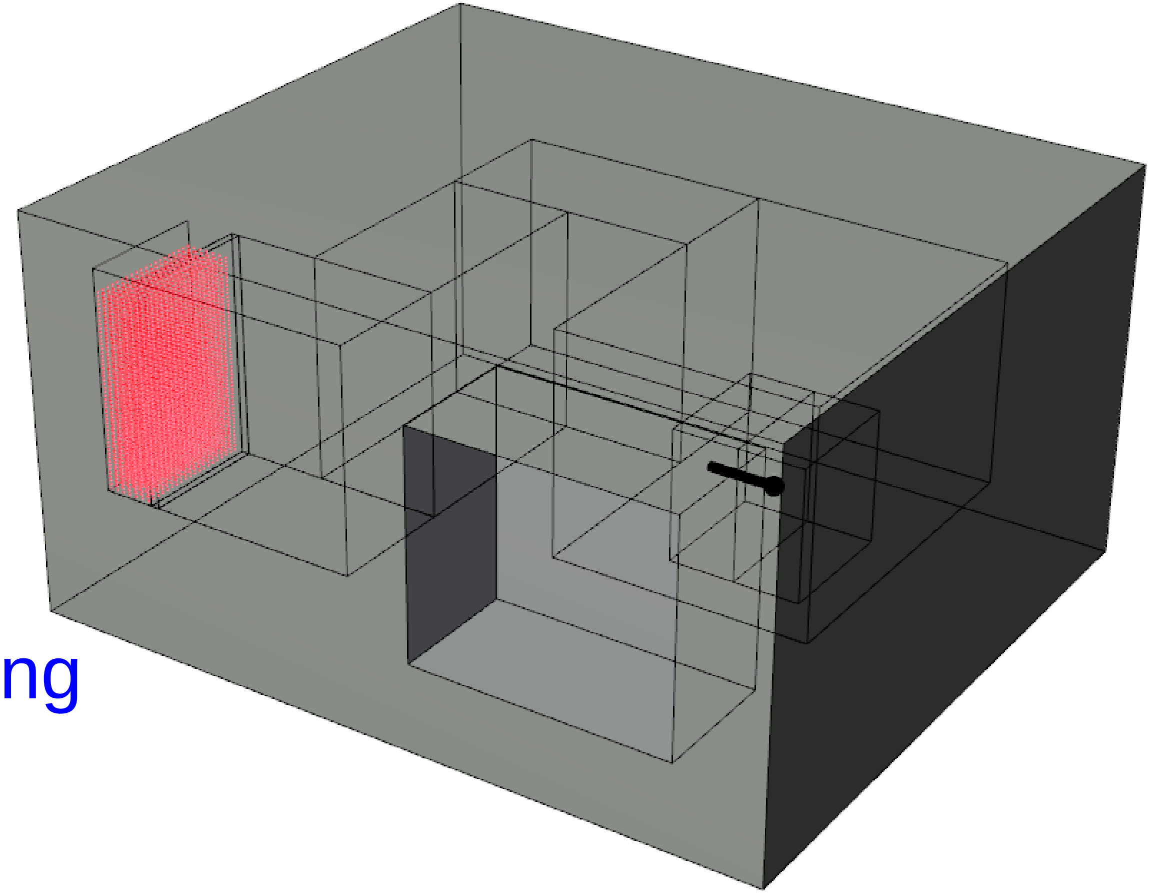
5. Turn off the visibility of the blackhole

   (Click on the checkbox next to the region named 001_Diff or click on the gray sphere and press the Space key)

A third example

6. Load the debug parameters from ngen_e3.sdb
7. Start the debugger

A huge amount of undefined points was found and marked by red spheres.



➡ A region is missing

Final remarks

Which algorithm is the best?

- The best is a total, uniform coverage of the region.

➡ possible only with infinitely many steps

➡ **Combine and try various methods.**

1.) Low discrepancy/Latin Hypercube/Jittered stratified
(recommended in the first place)
2.) Stochastic sampling
3.) Grid sampling