Exercise 09: Boolean Operations with Solid Geometries

2014-11-30: Tim C. Lueth, MIMED - Technische Universität München, Germany (URL: http://www.mimed.de)

Contents

- 1. Summary of the already explained aspects of the VLFL-Toolbox
- 2. Remarks on boolean operations for solid geometries based on surfaces
- 3. Creating two solids for showing the boolean operations
- 4. Boolean operator: Substraction A-B or A\B
- 5. Boolean operator: Substraction A+B
- 6. Boolean operator: Substraction B\A
- 7. Boolean operator: A xor B
- 8. Analyzing the results and comparision with additive design.
- Final remarks on toolbox version and execution date

1. Summary of the already explained aspects of the VLFL-Toolbox

In examples 1-6 we've learned about:

- *Point lists* (PL), *Vertex lists* (VL), *Facet lists* (VL), *Edge lists* (EL)
- Import, and export of *STL-File* data, Conversion of text strings into solid volumes
- *Closed polygon lists* (CPL) for boolesche operations of contours
- 2.5D extrusion of closed polyogon list (CPL) to create *Solid Geometries* (SG)
- 2.5D rotation of closed polyogon list (CPL) to create *Solid Geometries* (SG)
- Spatial transformation, positioning, alignment and merging of solid geometry sets

2. Remarks on boolean operations for solid geometries based on surfaces

The implementation of the boolean operations based on STL geometries took the author several years. The reason was not only the complexity of the numerous special cases but also the numerical accuracy or resolution of the required geometrical calucations. So even if a normal position is cacluated with 12 digits accuracy, the cross product often has a just an accuracy of 6 digits. Unfortunatly crossing triangles with position errors of 6 digits can easily lead to phantom triangles, phantom edges wich either do not really exist or are just doubles of already existing lines. Since normally all edges must have a second edge with the opposite direction, doubled lines/egdes with the same direction make trouble. So to be successful with the boolean operations you should make sure that

1. No facet should be in the same plane as or overlap another facet or cross with almost parallel edges to the plane of another facet. This is always valid for one solid, but in case of a second solid for boolean operations, it is quite difficult to guarantee this.
2. It is fact that, the more boolean operations took place, to create a new solid, the more vertices and facets were created. The removement of dispensable vertices and facets is possible but is a boring non productive pieve of source code. So the motivation to programm such a procedure is not high.
3. No edge of a triangle should be in the sample plane or crossing but almost parallel to a plane of a facet.
4. It is fact that a normal user just want to use the boolean operator without thinking about those problems. The normal user will just be disappointed if the way to design a physical solid object finally fails because of the limitations of the crossing

5. Make definitly sure that after all boolean operations you use SGchecker to analyze the solid geoemtry do detect errors immedeatly.
6. May be the only solution is to use a fixed coordiate grid during all calculations to make shure that two vertices are either definitly separated or definitly the same. #

3. Creating two solids for showing the boolean operations

```
VLFLfigure; view(-30,30); grid on;

A=SGofCPLz(PLcircle(10,4),10);  A=SGtrans0(A);
B=SGofCPLz(PLcircle(5,10),30);  B=SGtrans0(B); B=SGtransR(B,rotdeg(45,5,0));

SGplot(A,'b');
SGplot(B,'r');
VLFLplotlight (0,0.9);
A=SGstripfields(A)
B=SGstripfields(B)
```

```
A =

  struct with fields:

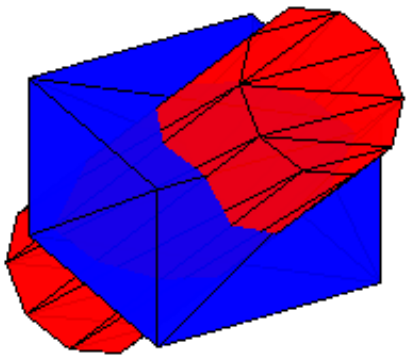
    VL: [8×3 double]
    FL: [12×3 double]


B =

  struct with fields:

    VL: [20×3 double]
    FL: [36×3 double]
```

'Tim C. Lueth:' : 22-Mar-2017 22:59:10



4. Boolean operator: Substraction A-B or A\B

```
close all;

C=SGbool ('-',A,B);

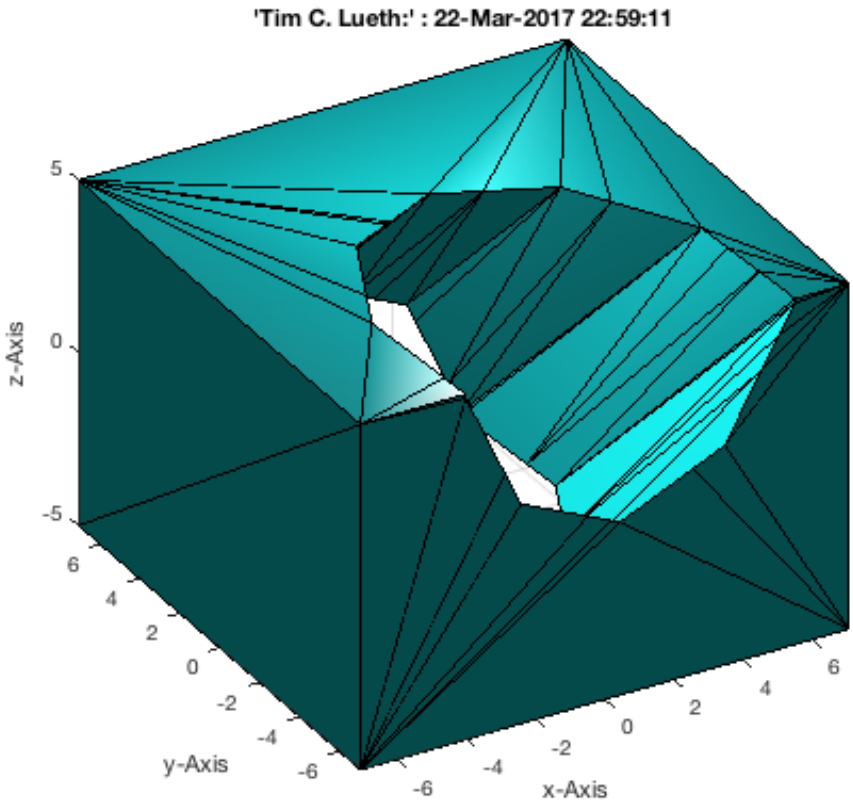
close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
```

```
C =

  struct with fields:

    VL: [60×3 double]
    FL: [120×3 double]
```



5. Boolean operator: Substraction A+B

```
close all;

C=SGbool ('+',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

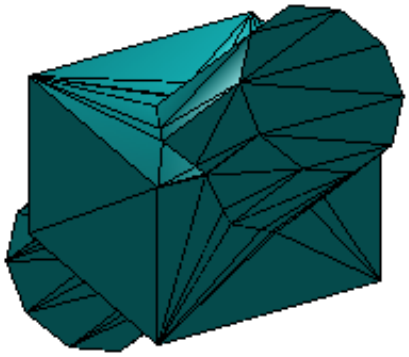
SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
return
```

```
C =

  struct with fields:

    VL: [80×3 double]
    FL: [156×3 double]
```

'Tim C. Lueth:' : 22-Mar-2017 22:59:12



6. Boolean operator: Substraction B\A

```
close all;

C=SGbool ('B',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
```

7. Boolean operator: A xor B

```
close all;

C=SGbool ('x',A,B);

close all;
VLFLfigure; view(-30,30); grid on;

SGplot(C,'c');
VLFLplotlight (1,0.9);
SGchecker(C);
C=SGstripfields(C)
```

8. Analyzing the results and comparision with additive design.

Analyzing the number of vertices and facets of the results of a boolean operation shows clearly that there are much more vertices and facets than the sum of the vertices and facets. In general it makes for STL more sense to add simple solids to a more complex by attaching them together by simply pushing them into another. In this case the final number of vertices and facets is the sum of the individual facets and vertices.

Final remarks on toolbox version and execution date

```
VLFLlicense
```

- *Tim Lueth, tested and compiled on OSX 10.7.5 with Matlab 2014b on 2015-06-07*
- *Christina Friedrich, executed and published on 64 Bit PC using Windows with Matlab 2015a on 2015-06-17*

Published with MATLAB® R2016b