# Computational topology: Lecture 12

Alberto Paoluzzi

April 26, 2019

1. Software debugging
2. Debugging `planar_arrangement()`
3. Debugging `spatial_arrangement()`: it loops! :-(

# Software debugging

# Definition

Debugging is the routine process of locating and removing computer program bugs, errors or abnormalities, which is methodically handled by software programmers via debugging tools.

Debugging checks, detects and corrects errors or bugs to allow proper program operation according to set specifications.

**Definition**

Debugging is the process of finding and resolving defects or problems within a computer program that prevent correct operation of computer software or a system.

# A debugging strategy

From 5 Steps to a Bullet-Proof Debugging Strategy

1. Get into the Debugging Mindset
2. Prioritize Which Bug to Fix
3. So Where Is the Bug, Exactly?
4. Which Line of Code Is the Bug On, Exactly?
5. What's the Solution?

# Debugger for julia

A Julia interpreter and debugger

# Juno GUI (on top of Atom)

## Some examples

# Debugging planar_arrangement()

# Prepare parametric test examples

- randomarrangement2d.jl
- randombubbles.jl
- randomlines.jl
- randomshapes.jl
- stresstest2d.jl

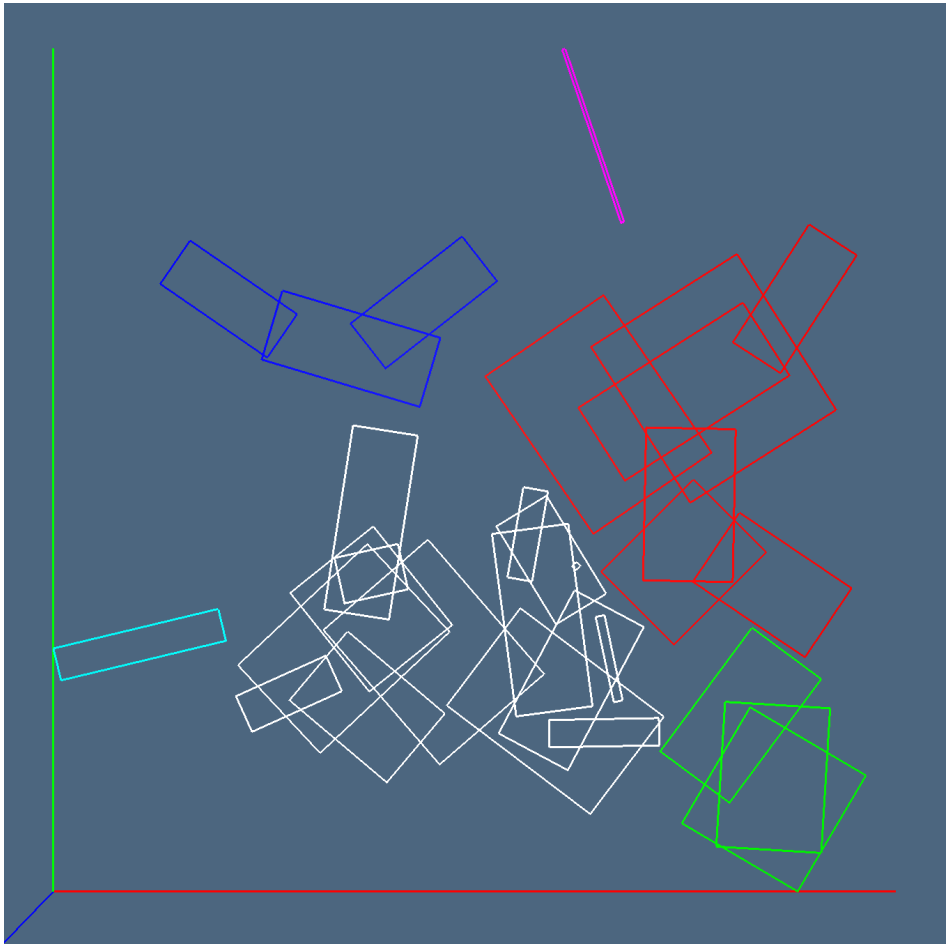

Figure 1: Some examples

# Test for containment of holes in holes

Can be done by preparing the input as an .SVG file

svg2lar.jl

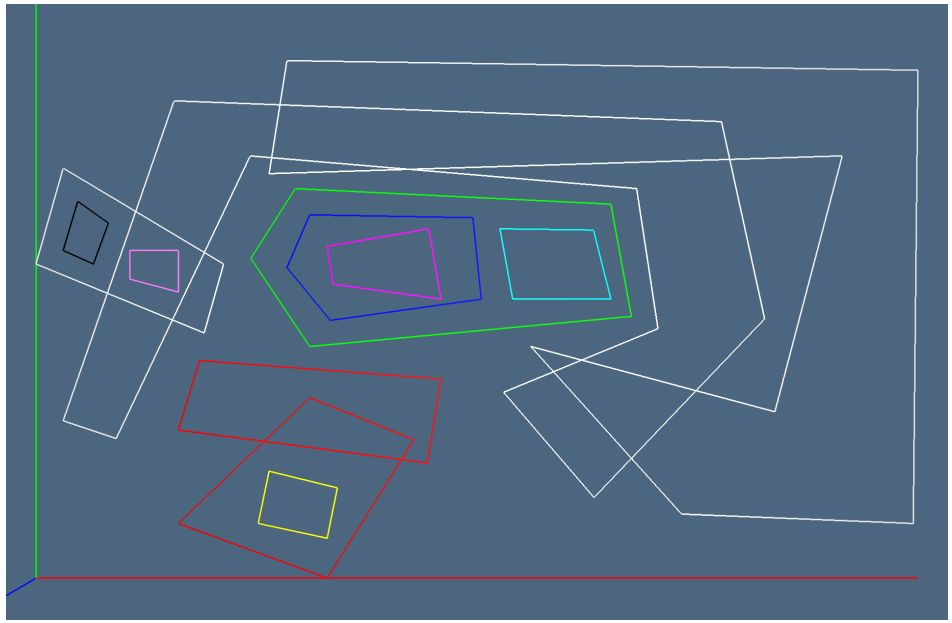

Figure 2: Some examples

# Test for containment of holes in holes

svg2lar.jl

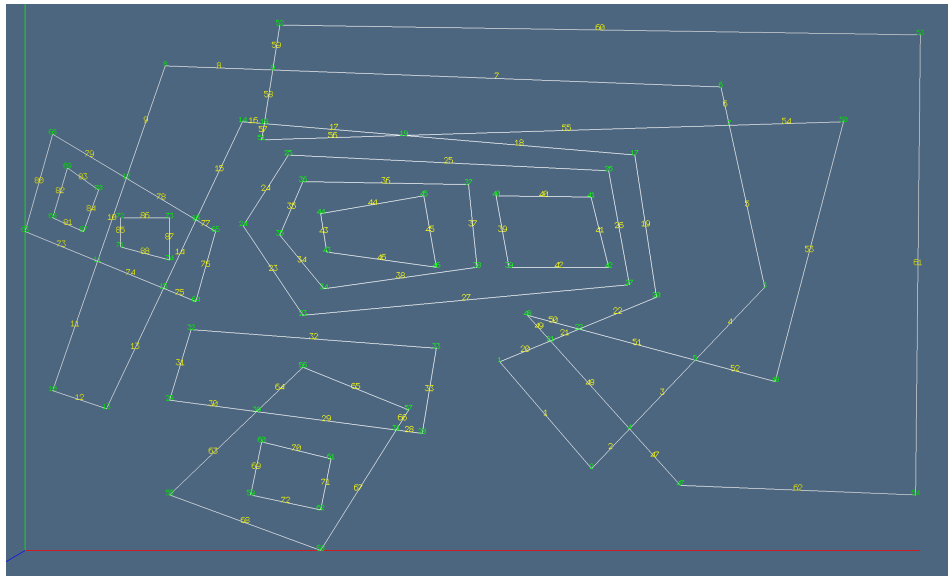

Figure 3: Some examples

# 2D space arrangement generated by random circles

randombubbles.jl

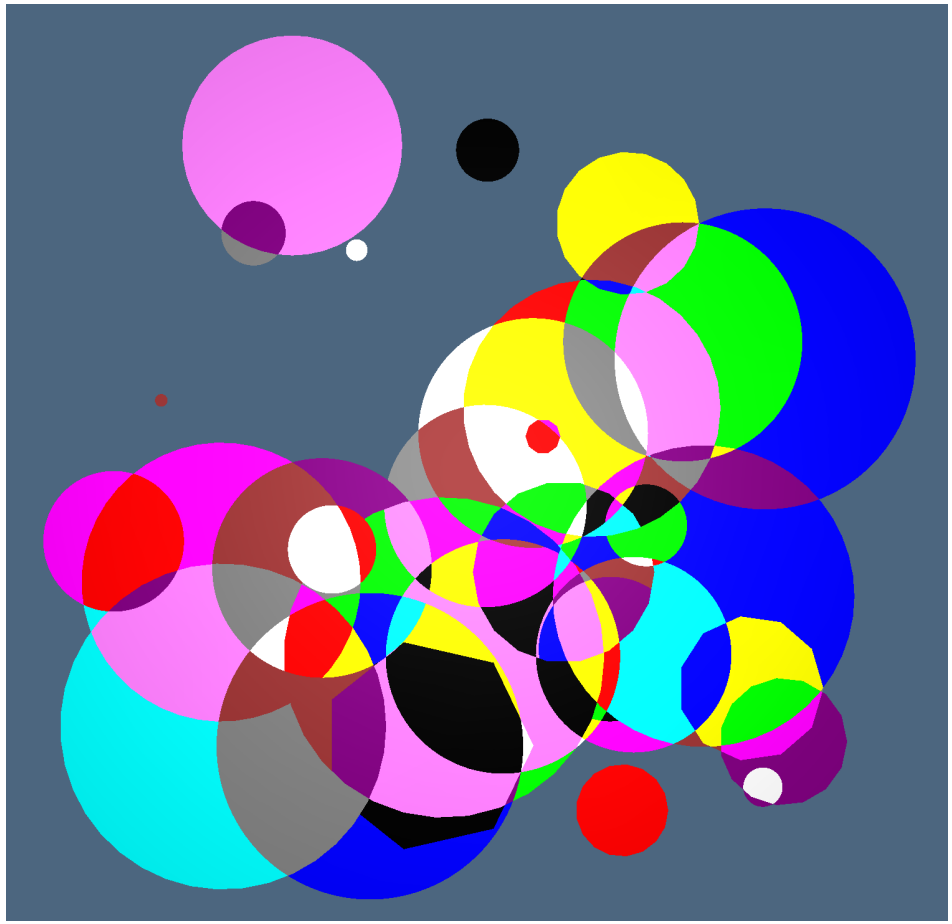

Figure 4: Some examples

# Exploded (non convex) polygons

randombubbles.jl

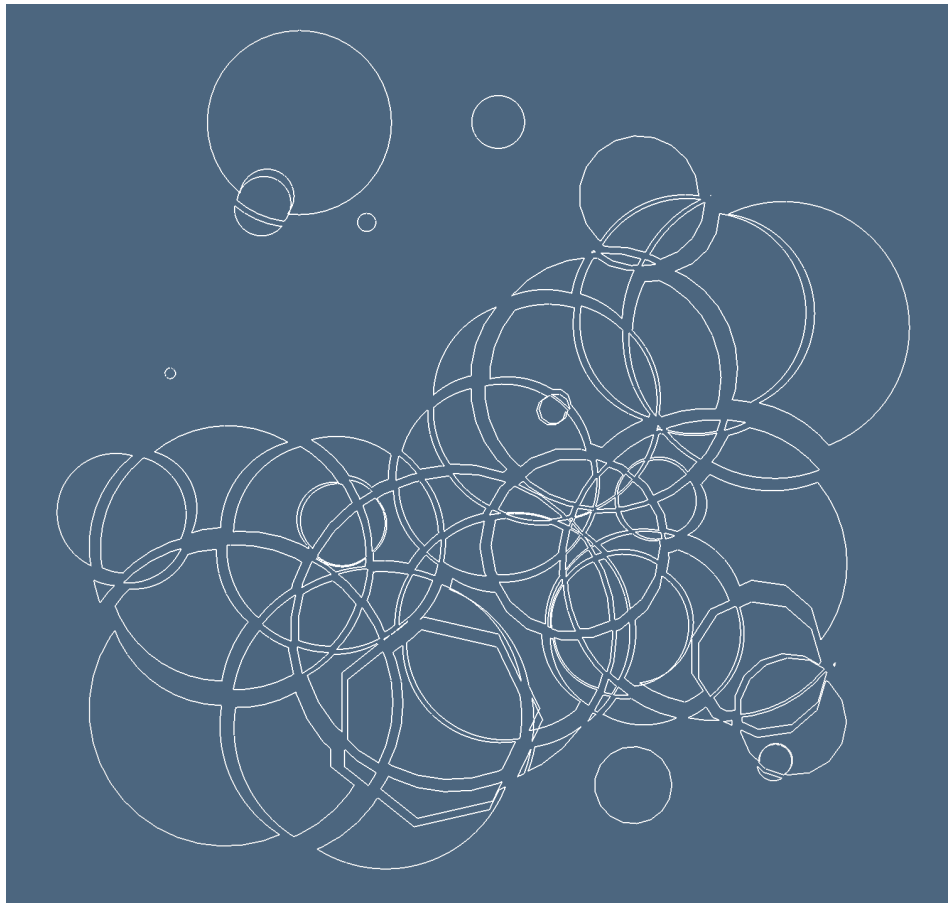

Figure 5: Some examples

# Debugging spatial_arrangement(): it loops! :-(

# Start with a single 3D cube

## twocubes.jl

comment (partially) the generating expression `twocubes`

```
using LinearAlgebraicRepresentation
using Plasm, SparseArrays
Lar = LinearAlgebraicRepresentation

V,(VV,EV,FV,CV) = Lar.cuboid([0.5,0.5,0.5],true,[-0.5,-0.5,-0.5])
mybox = (V,CV,FV,EV)

twocubes = Lar.Struct([ mybox ])#, Lar.t(0.3,0.4,0.5), Lar.r(pi/3,0,0), l
V,CV,FV,EV = Lar.struct2lar(twocubes)
Plasm.view(V,CV)

cop_EV = Lar.coboundary_0(EV::Lar.Cells);
cop_EW = convert(Lar.ChainOp, cop_EV);
cop_FE = Lar.coboundary_1(V, FV::Lar.Cells, EV::Lar.Cells);
W = convert(Lar.Points, V');

#V, copEV, copFE, copCF = Lar.Arrangement.spatial_arrangement(
    W::Lar.Points, cop_EW::Lar.ChainOp, cop_FE::Lar.ChainOp)

Plasm.view(Plasm.numbering(0.25)((V,[[[k] for k=1:size(V,2)],EV,FV])))
```

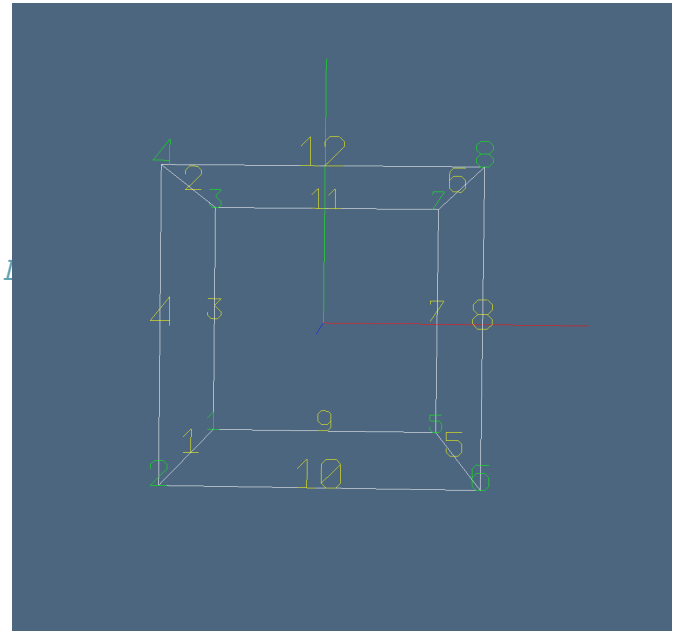


Figure 6: Some examples

# With TWO cubes it loops !!

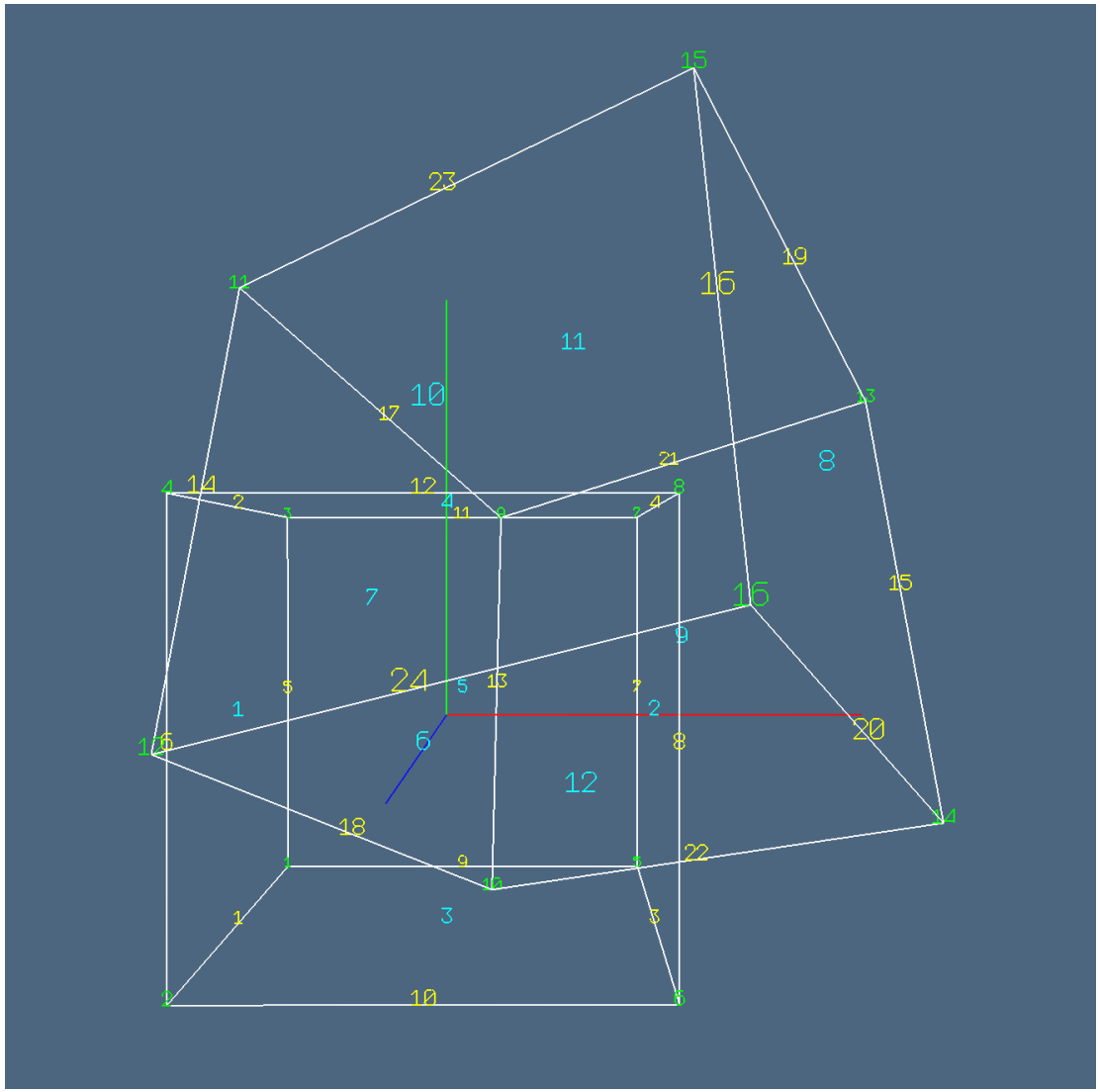


Figure 7: Some examples

# First clue: look for evidence

Notice that the numbering of decomposed edges do not correspond to the expected ones on the right-front edge of the cube centered on the origin . . .

Why . . . ?!?

Let guess

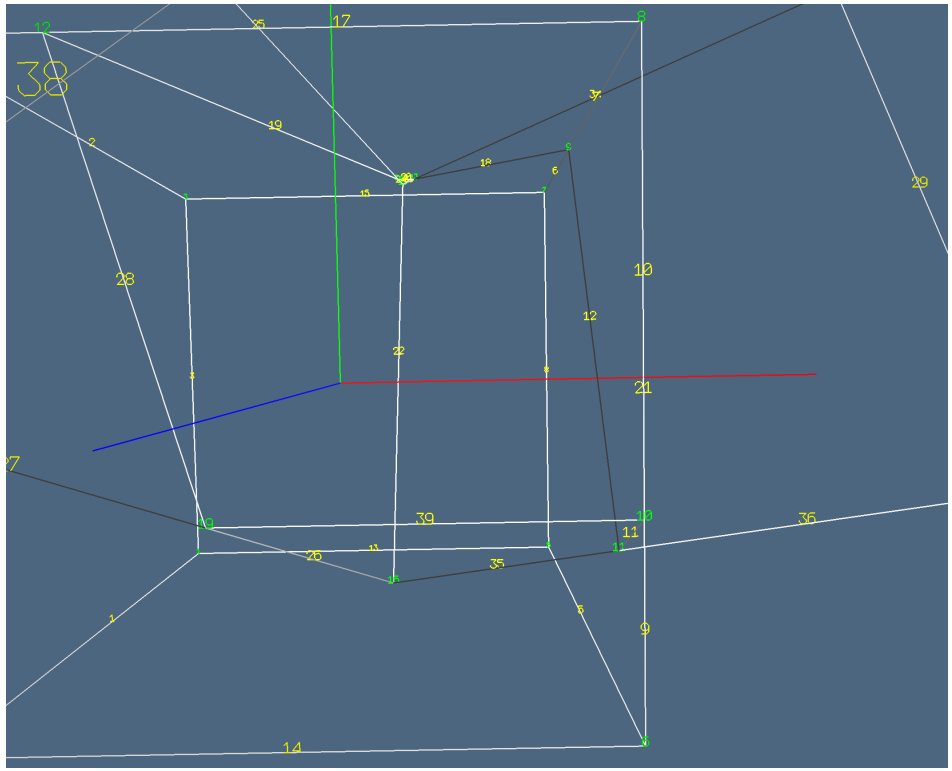


Figure 8: Some examples

# Going further deep

The smaller cube (centered on the origin) is the more distant from the observer.

Just look at edges 41 and 73 on its front face, and to edges 9, 10 and 31 on its rightmost edge.

Why . . . ?!?

Let guess

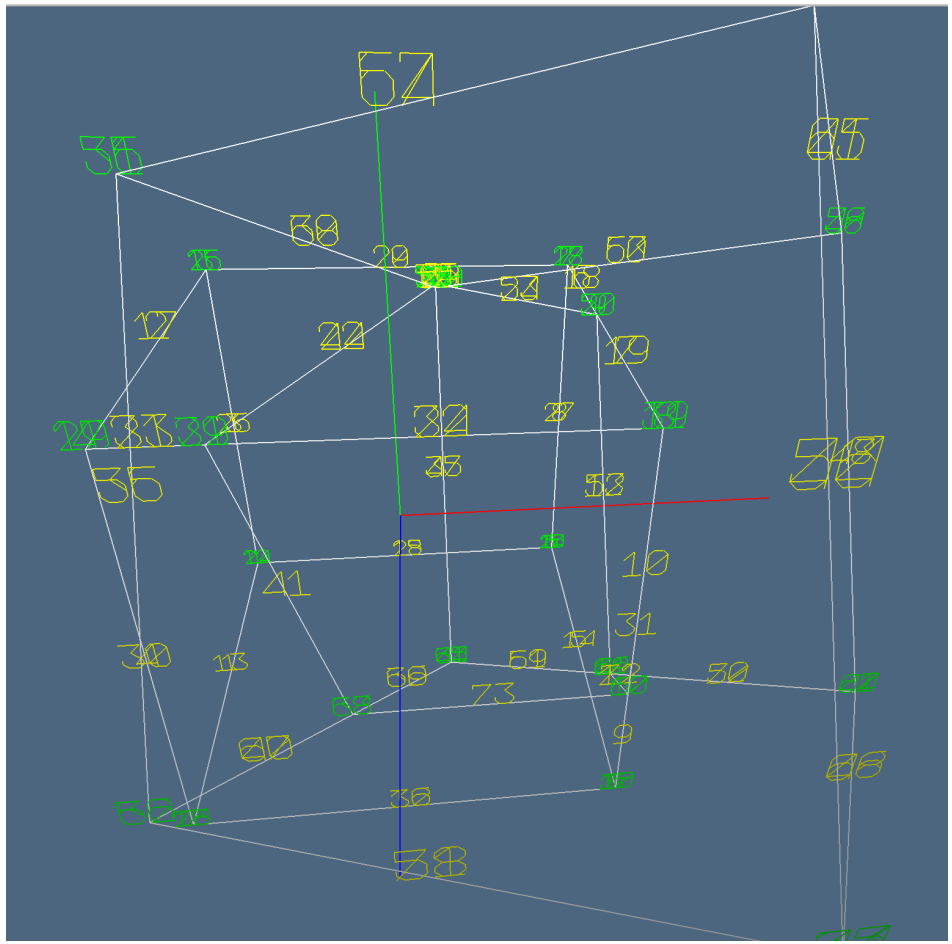


Figure 9: Some examples

# aaaaaaaaa

# aaaaaaaaa

# aaaaaaaaa

# aaaaaaaaa