

Lesson 4.1

Three-dimensional geometries in geoinformation

GEO1004:
3D modelling of the built environment

<https://3d.bk.tudelft.nl/courses/geo1004>



3D geoinformation

Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

About GIS standards

Standardisation organisations

- **ISO:** International Standardisation Organisation
- **OGC:** Open Geospatial Consortium
- **INSPIRE:** Infrastructure for Spatial Information in Europe
- **NEN:** *normalisatie en normen* (“Dutch ISO”)

ISO 191xx series of geographic information standards

- A main goal of the ISO 191xx series is to facilitate the interoperability of geographic information systems, i.e. ability to discover, access, understand and use the information and tools, independently from the platform supporting them
- Structured set of standards specifying methods, tools and services for the management of geographic information, including:
 - Definition of data,
 - Access to data
 - Presentation of data
 - Transfer between users
- weirdly, and annoyingly, one has to pay to read these 😞 (but free if TUDelft student!)

ISO 191xx series of geographic information standards

6709 - Standard representation of latitude, longitude and altitude for geographic point locations	19122 - Qualifications and Certification of personnel
19101 - Reference model	19123 - Schema for coverage geometry and functions
19101-2 - Reference model - Part 2: Imagery	19124 - Imagery and gridded data components
19103 - Conceptual schema language	19125-1 - Simple feature access - Part 1: Common architecture
19104 - Terminology Introduction	19125-2 - Simple feature access - Part 2: SQL option
19105 - Conformance and testing	19126 - Profile - FACC Data Dictionary
19106 - Profiles	19127 - Geodetic codes and parameters
19107 - Spatial schema	19128 - Web Map server interface
19108 - Temporal schema	19129 - Imagery, gridded and coverage data framework
19109 - Rules for applicaiton schema	19130 - Sensor and data models for imagery and gridded data
19110 - Methodology for feature cataloguing	19131 - Data product specifications
19111 - Spatial referencing by coordinates	19132 - Location based services - Reference model
19112 - Spatial referencing by geographic identifiers	19133 - Location based services - Tracking and navigation
19113 - Quality principles	19134 - Multimodal location based services for routing and navigation
19114 - Quality evaluation procedures	19135 - Procedures for registration of geographical information items
19115 - Metadata	19136 - Geography Markup Language

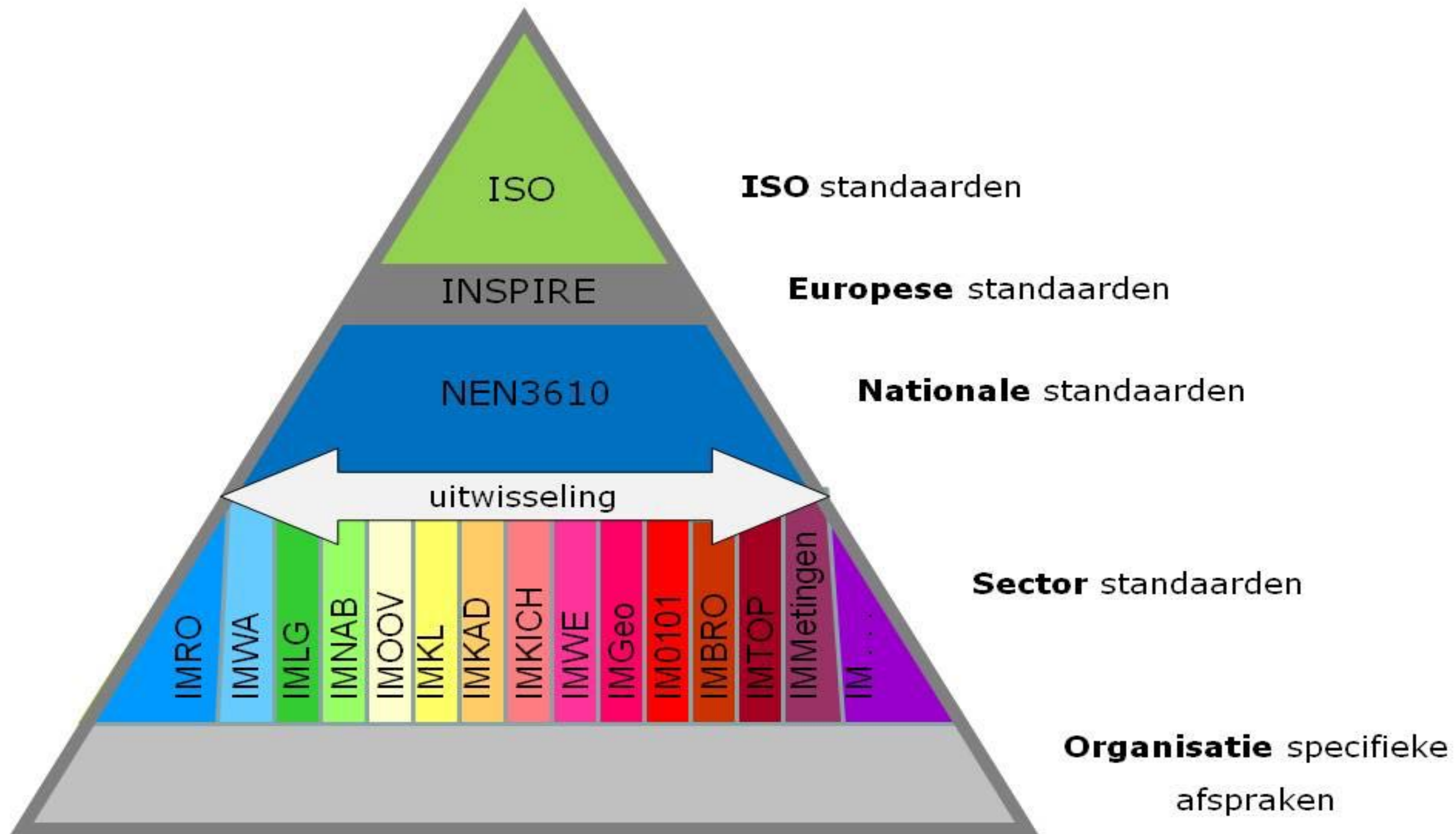
19115-2 - Metadata - Part 2: Extensions for imagery and gridded data	19137 - Generally used profiles of the spatial schema and of similar important other schemas
19116 - Positioning services	19138 - Data quality measures
19117 - Portrayal	19139 - Metadata - Implementation specification
19118 - Encoding	19140 - Technical amendment to the ISO 191** Geographic information series of standards for harmonization and enhancements
19119 - Services	19141 - Schema for moving features
19120 - Functional standards	
19121 - Imagery and gridded data	

OGC: Open Geospatial Consortium®



- many standards are aligned with ISO's (ie, they are exactly the same)
- OGC usually standards closer to implementations than ISO
 - GML
 - KML
 - netCDF
- abstract specifications = ISO
- implementation specification = OGC
- unlike ISO's, OGC's documents are free for everyone 😊

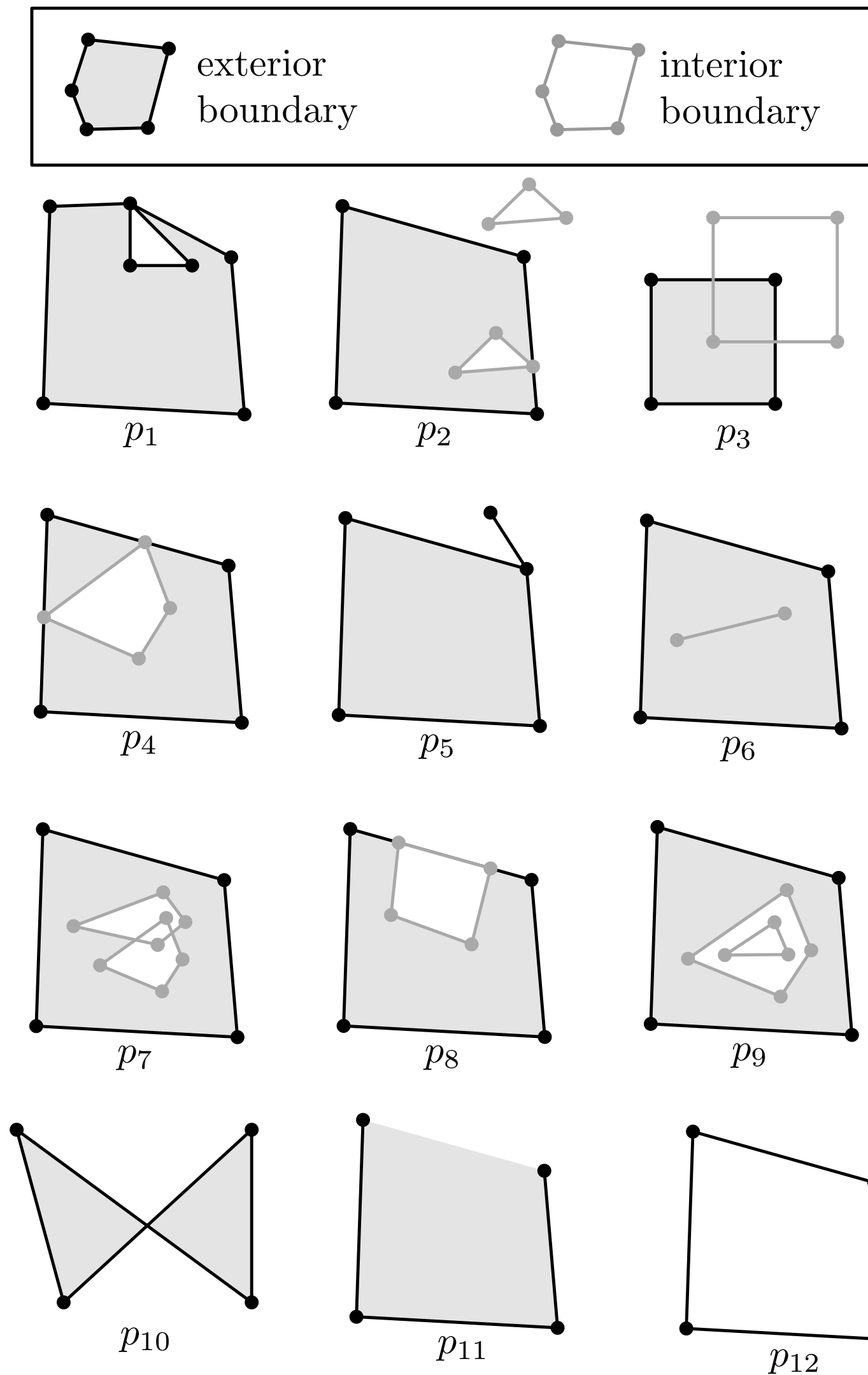
The standards from a Dutch point-of-view



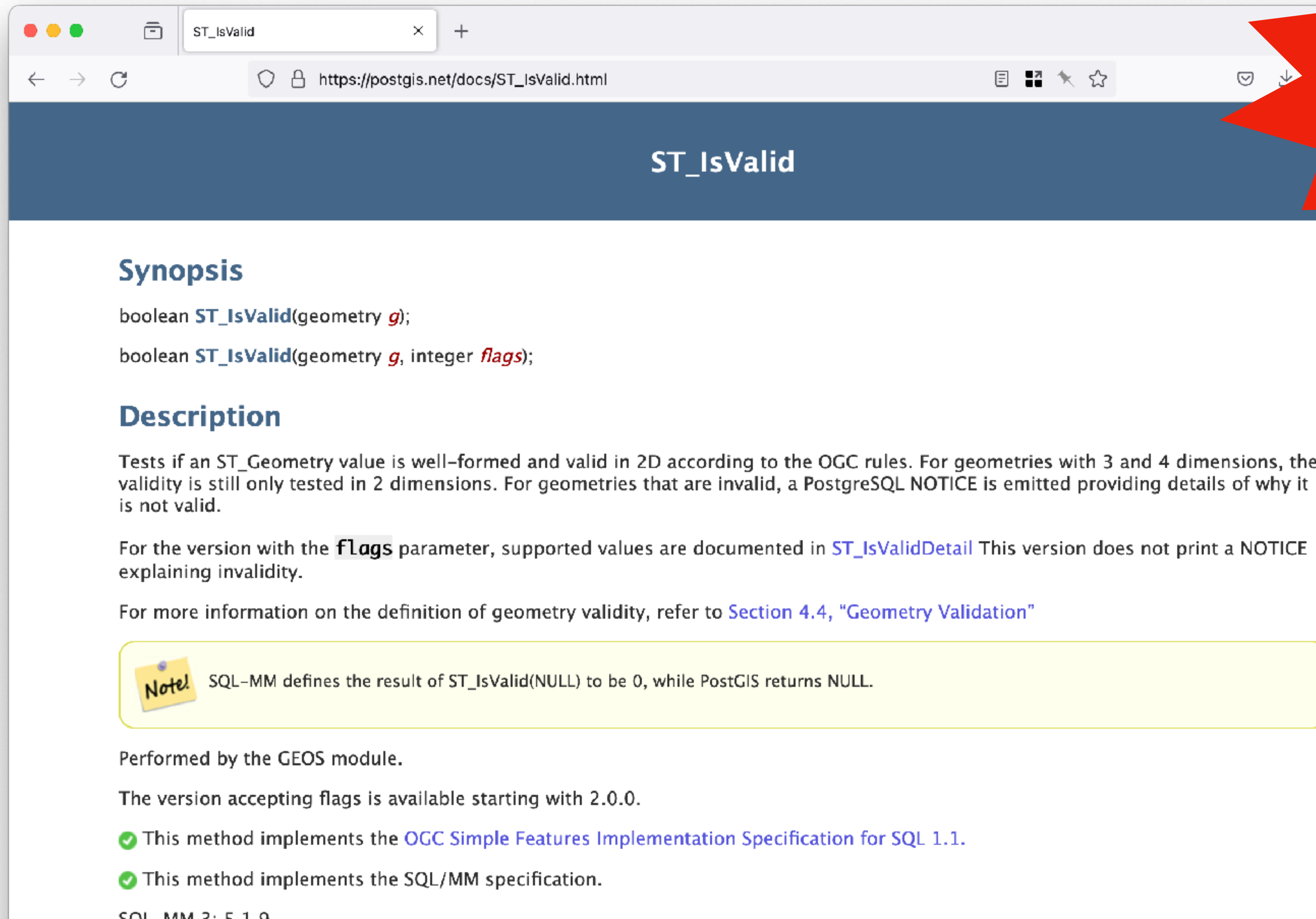
Rules for validation in 2D

OGC Simple Features and
ISO19107 rules:

- 1 no self-intersection
- 2 closed boundaries
- 3 rings can touch but not overlap
- 4 no duplicate points
- 5 no dangling edges
- 6 connected interior
- 7 etc



Rules for validation in 2D: standard implementations available



The screenshot shows a web browser window with the URL `https://postgis.net/docs/ST_IsValid.html`. The page title is `ST_IsValid`. The content includes a synopsis, a description, and a note. The synopsis shows two function signatures: `boolean ST_IsValid(geometry g);` and `boolean ST_IsValid(geometry g, integer flags);`. The description explains that the function tests if an ST_Geometry value is well-formed and valid in 2D according to the OGC rules. It also mentions that for geometries with 3 and 4 dimensions, the validity is still only tested in 2 dimensions. A note states that SQL-MM defines the result of `ST_IsValid(NULL)` to be 0, while PostGIS returns NULL. The page is performed by the GEOS module and the version accepting flags is available starting with 2.0.0. It also mentions that this method implements the OGC Simple Features Implementation Specification for SQL 1.1 and the SQL/MM specification.

Synopsis

```
boolean ST_IsValid(geometry g);  
boolean ST_IsValid(geometry g, integer flags);
```

Description

Tests if an ST_Geometry value is well-formed and valid in 2D according to the OGC rules. For geometries with 3 and 4 dimensions, the validity is still only tested in 2 dimensions. For geometries that are invalid, a PostgreSQL NOTICE is emitted providing details of why it is not valid.

For the version with the `flags` parameter, supported values are documented in [ST_IsValidDetail](#) This version does not print a NOTICE explaining invalidity.

For more information on the definition of geometry validity, refer to [Section 4.4, "Geometry Validation"](#)

Note! SQL-MM defines the result of `ST_IsValid(NULL)` to be 0, while PostGIS returns NULL.

Performed by the GEOS module.

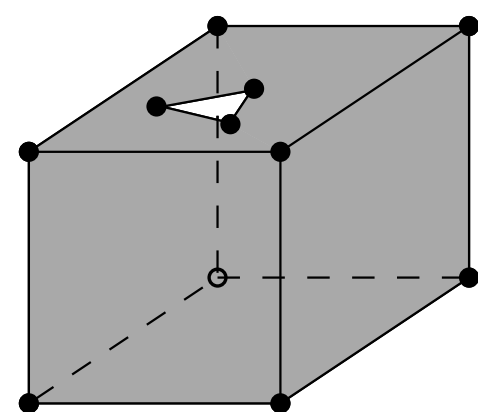
The version accepting flags is available starting with 2.0.0.

- ✓ This method implements the [OGC Simple Features Implementation Specification for SQL 1.1](#).
- ✓ This method implements the SQL/MM specification.

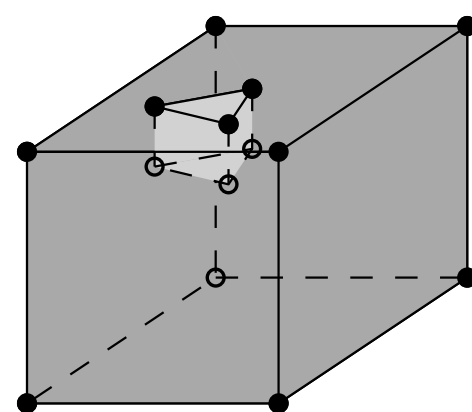
SQL-MM 2: 5.1.9

Everyone uses GEOS
implementation

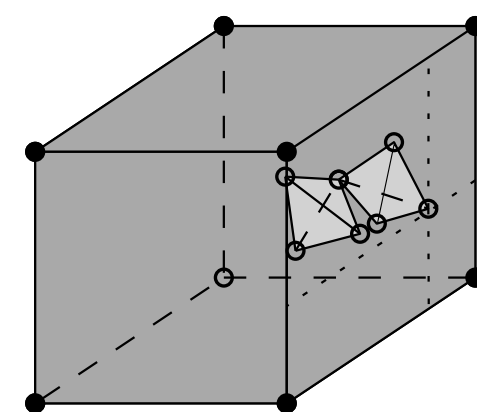
In 3D it's way way way more complex...



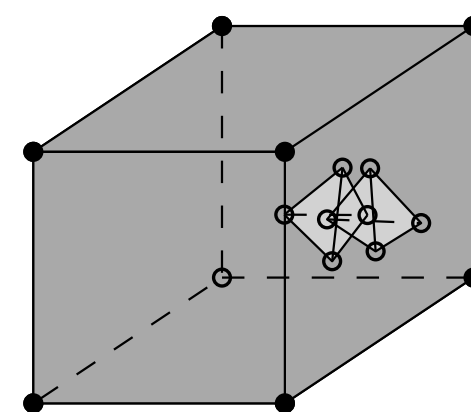
s_1
invalid (1)



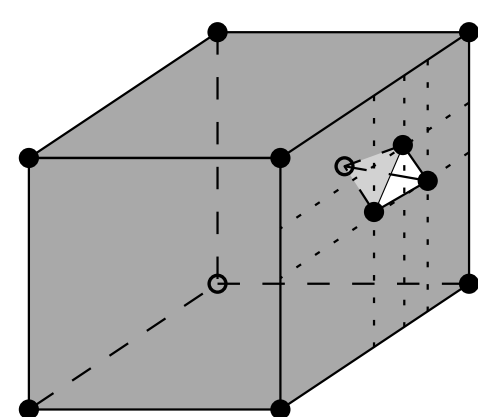
s_2
valid



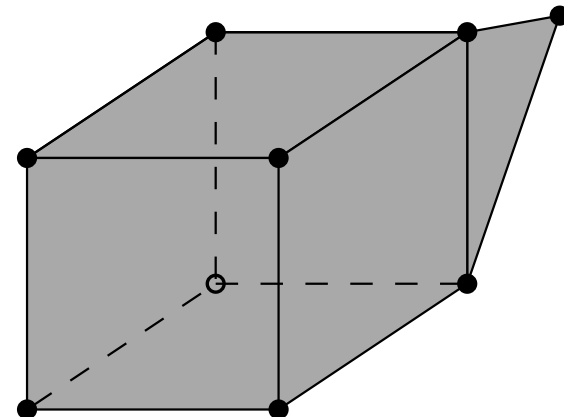
s_3
valid



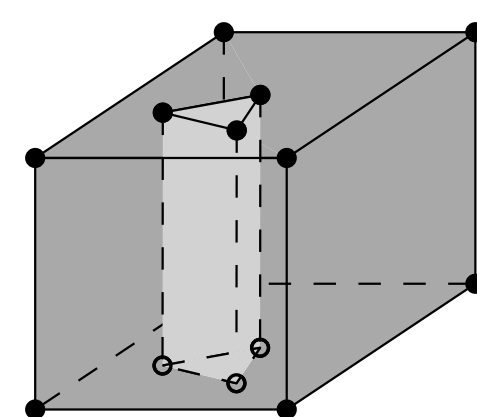
s_4
invalid (3, 6)



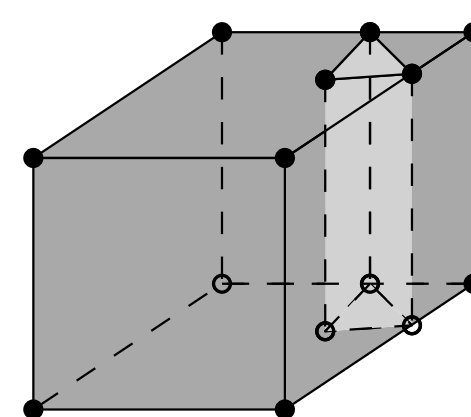
s_5
invalid (6)



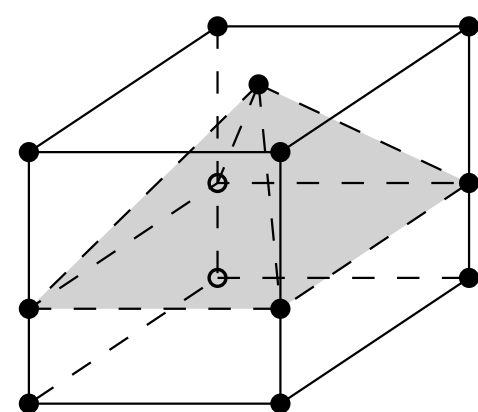
s_6
invalid (4)



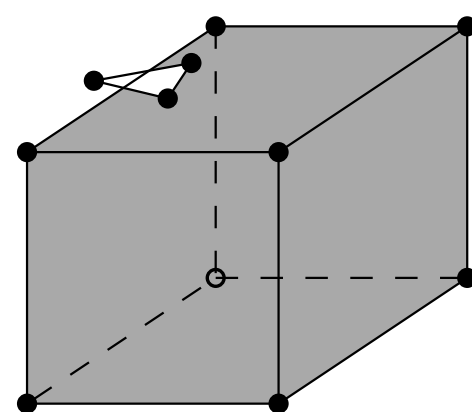
s_7
valid



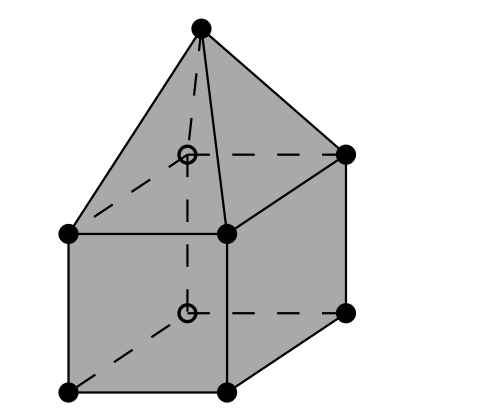
s_8
invalid (2, 5)



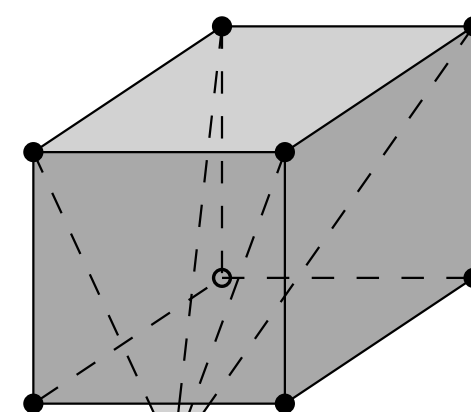
s_9
invalid (5)



s_{10}
invalid (3 in 2D)



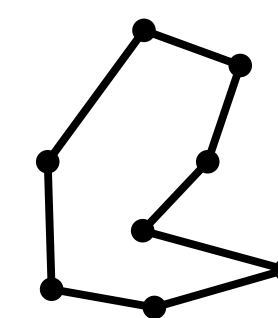
s_{11}
valid



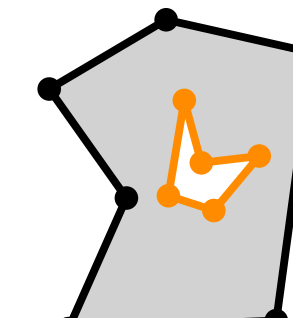
s_{12}
invalid (2)



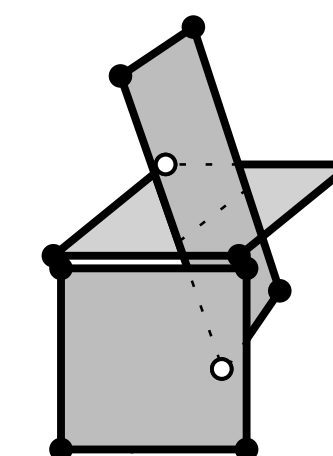
Point



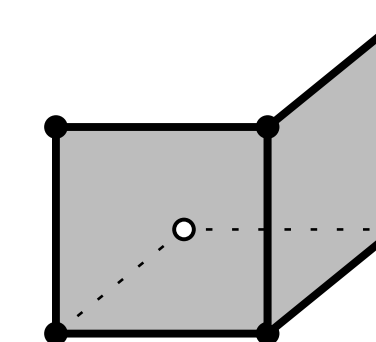
LinearRing



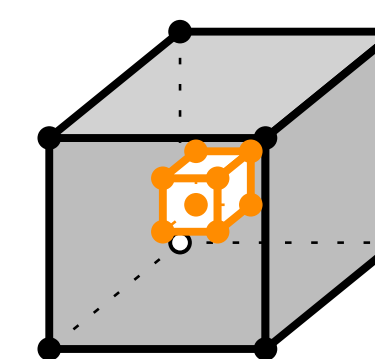
Polygon



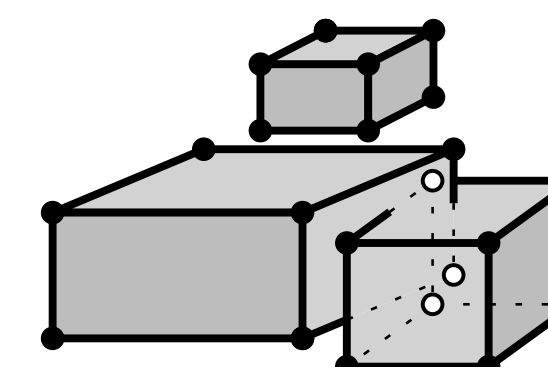
MultiSurface



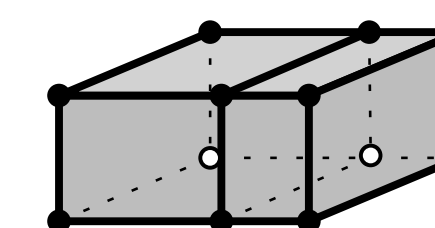
CompositeSurface



Solid



MultiSolid



CompositeSolid

In 3D it's way more complex... and no GEOS/PostGIS implementation

The screenshot shows the GitHub repository page for `tudelft3d/val3dity`. The repository is public and has 86 stars, 22 forks, and 14 watchers. The main branch is `main`, and there are 4 branches and 21 tags. The repository description is "Validation of 3D primitives according to the international standard ISO19107". The repository includes a README, GPL-3.0 license, and a list of tags: `validation`, `gis`, `cgal`, `gml`, `iso19107`, `3d-gis`, `indoorgml`, and `cityjson`. The repository also has 19 releases, with the latest being `2.4.0`.

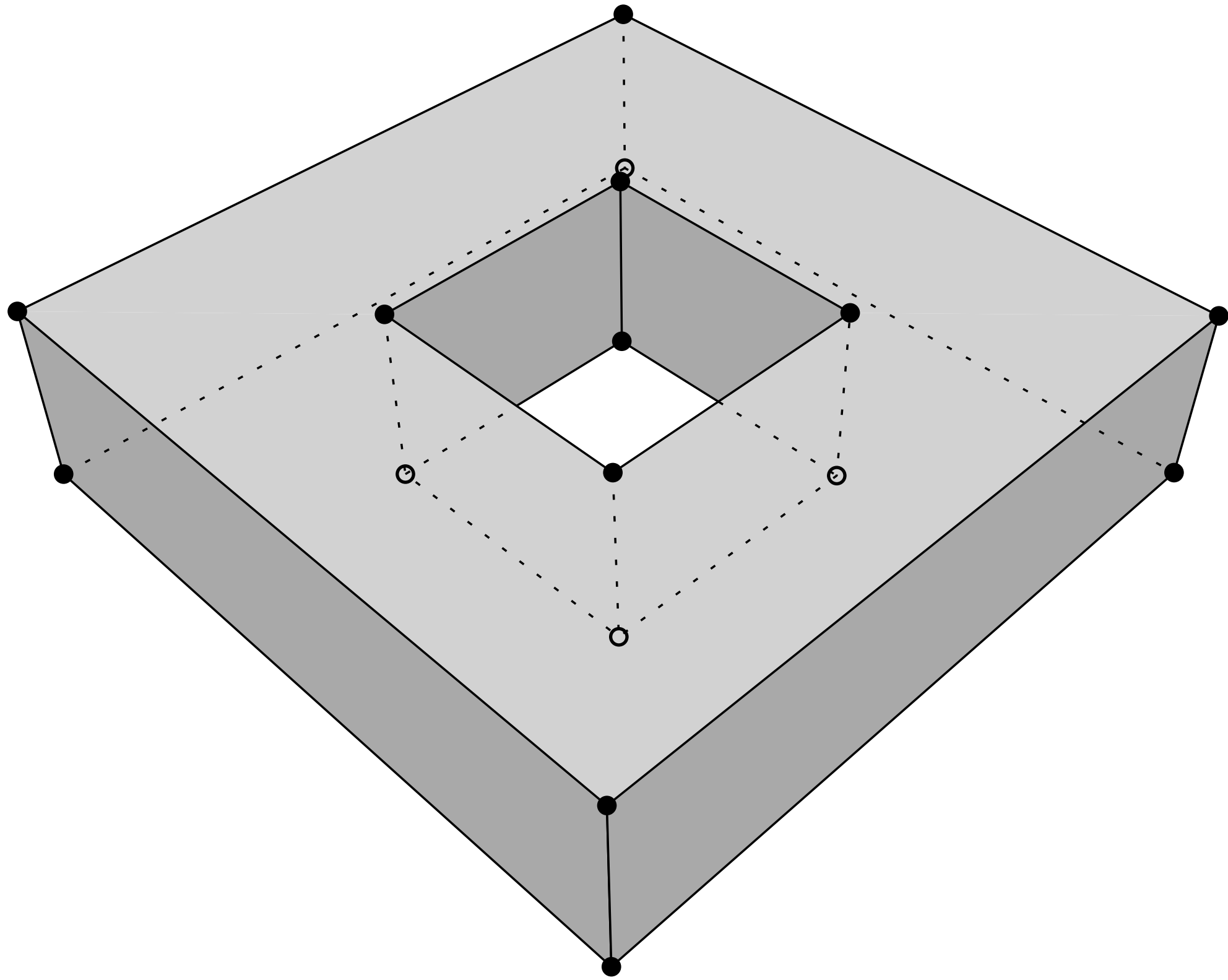
The repository's file structure is as follows:

File/Folder	Commit Message	Commit Time
<code>.github/workflows</code>	Revert "Fix wrong branch in homebrew yml update file"	2 years ago
<code>cmake</code>	Update the cmake for using Eigen (new way imposed ...)	3 years ago
<code>data</code>	Fix the yml file for the pytest: 2 files were not described	8 months ago
<code>demo_lib</code>	Fix the readme in the demo_lib, old functions were still...	8 months ago
<code>docs</code>	Bump to version 2.4.0	8 months ago
<code>misc</code>	Add termcast in svg, looks neater, innit?	4 years ago
<code>report_browser</code>	Add link to val3dity webapp in the report browser	4 years ago
<code>src</code>	Bump to version 2.4.0	8 months ago
<code>tests</code>	Add unit test for diff versions of cityjson v1.0 + v1.1)	9 months ago
<code>thirdparty</code>	Upgrade pugixml to v1.13	8 months ago

Quiz time



Quiz time!

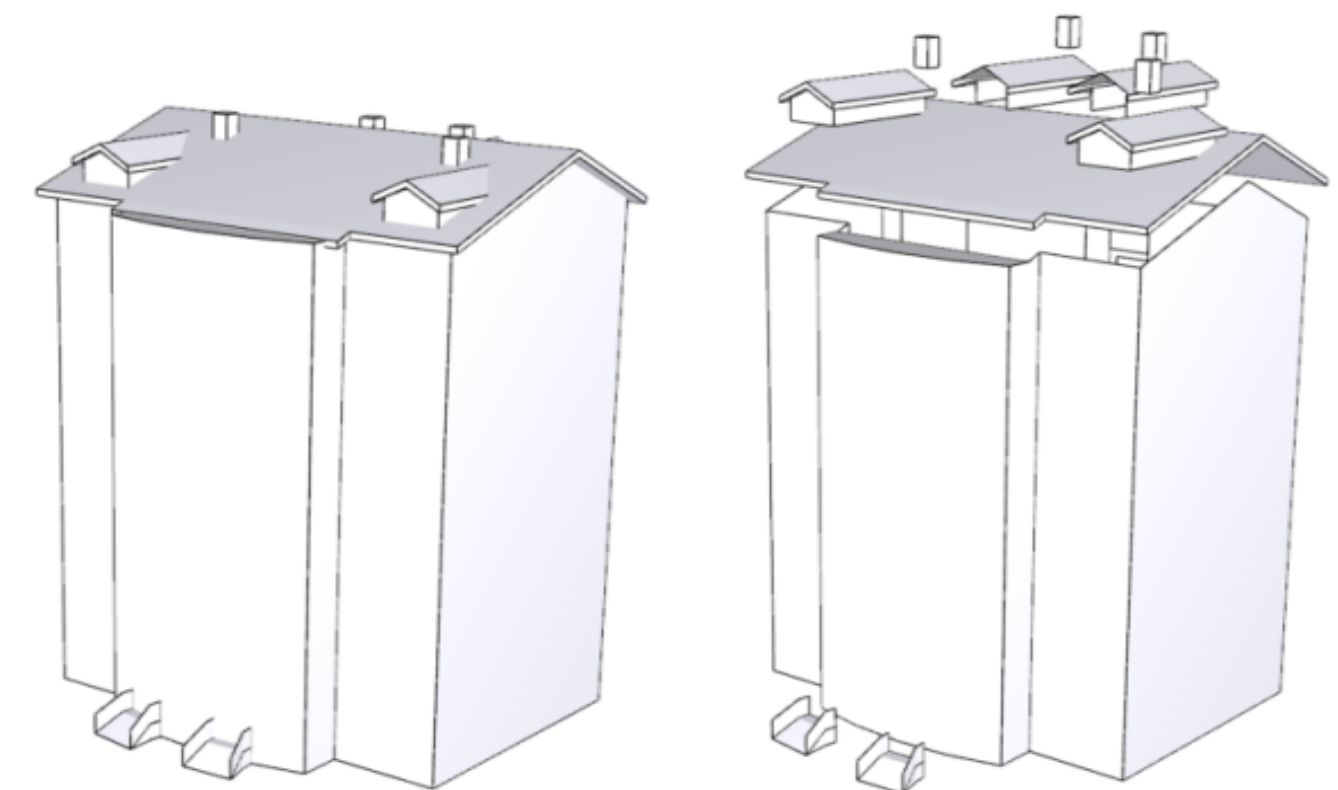
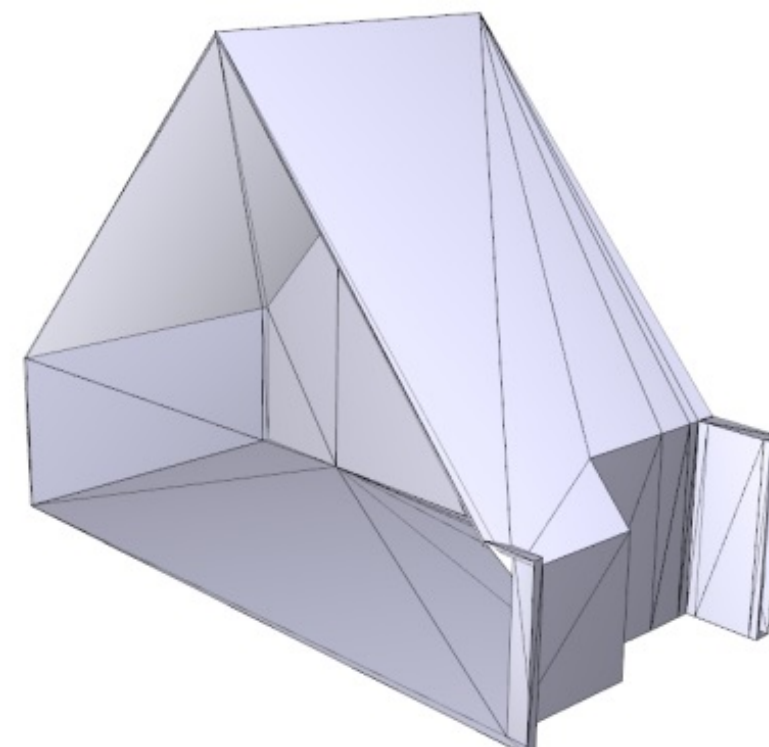
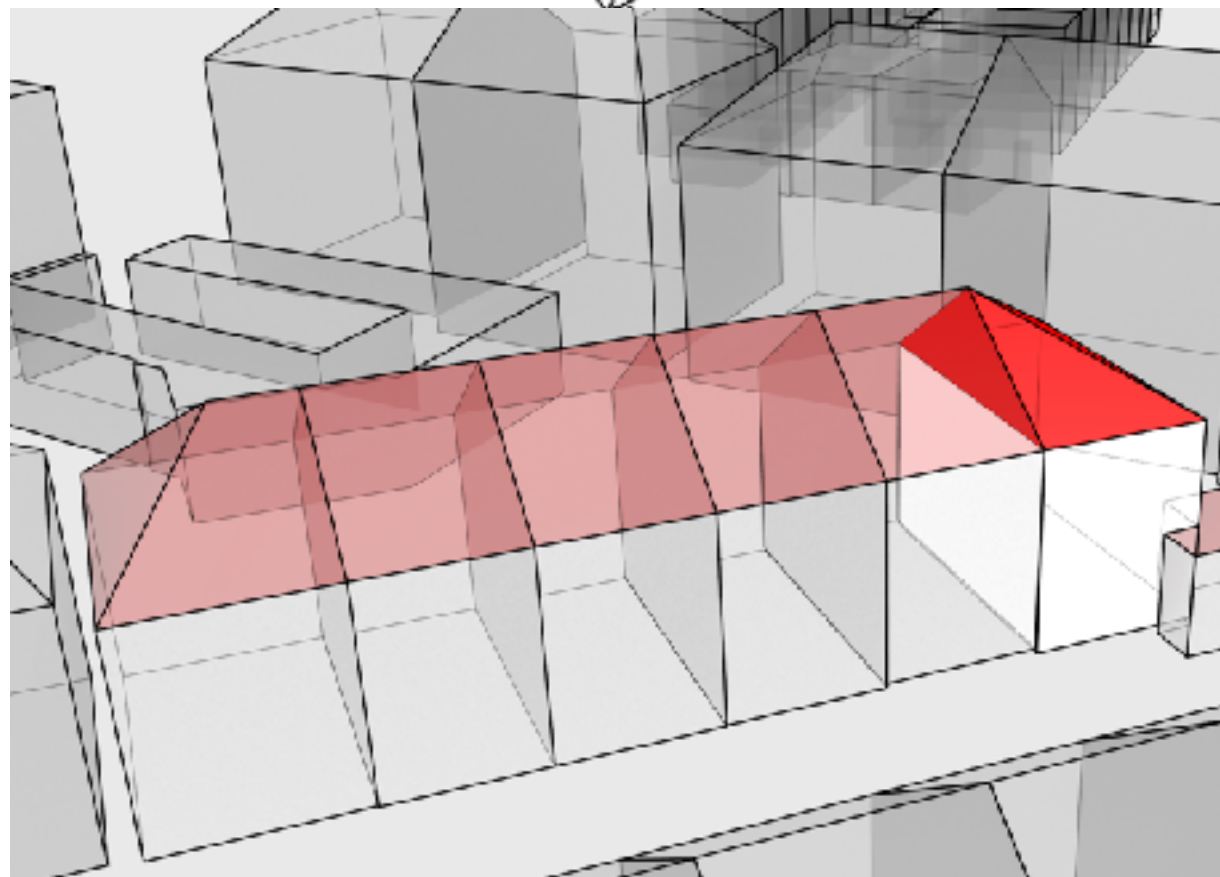
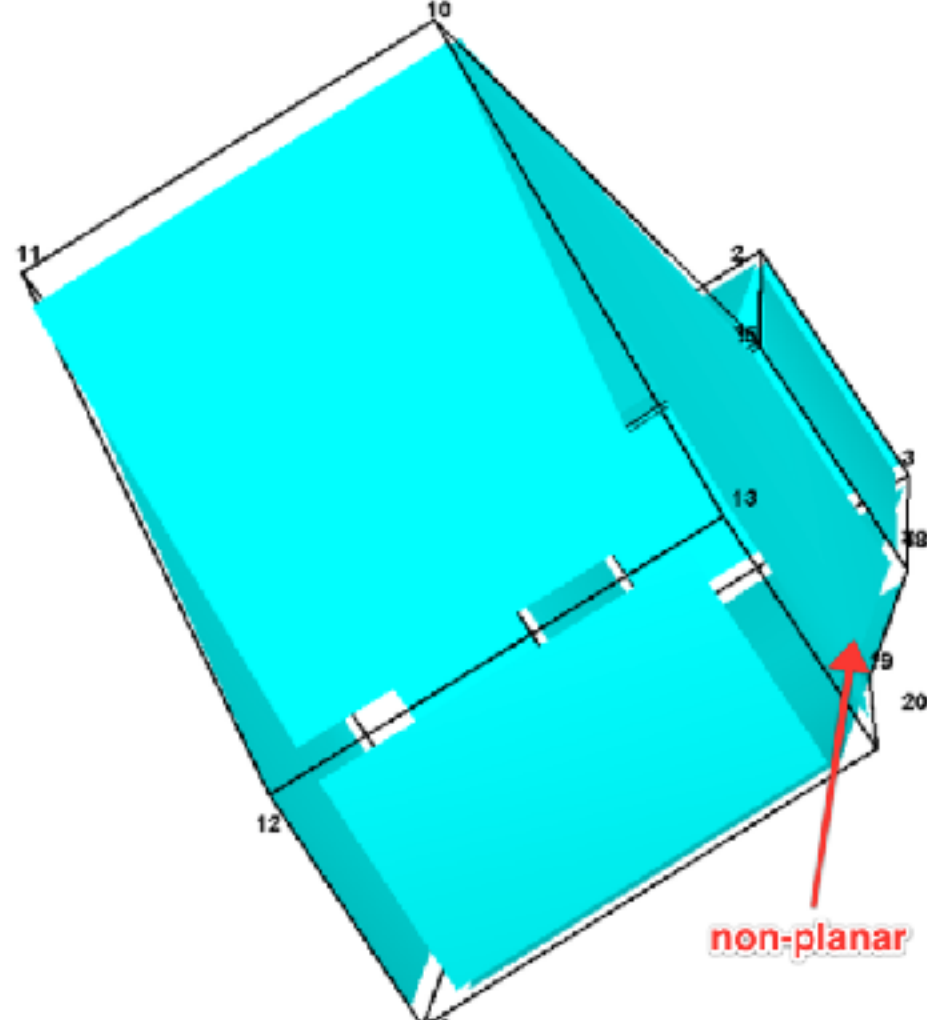
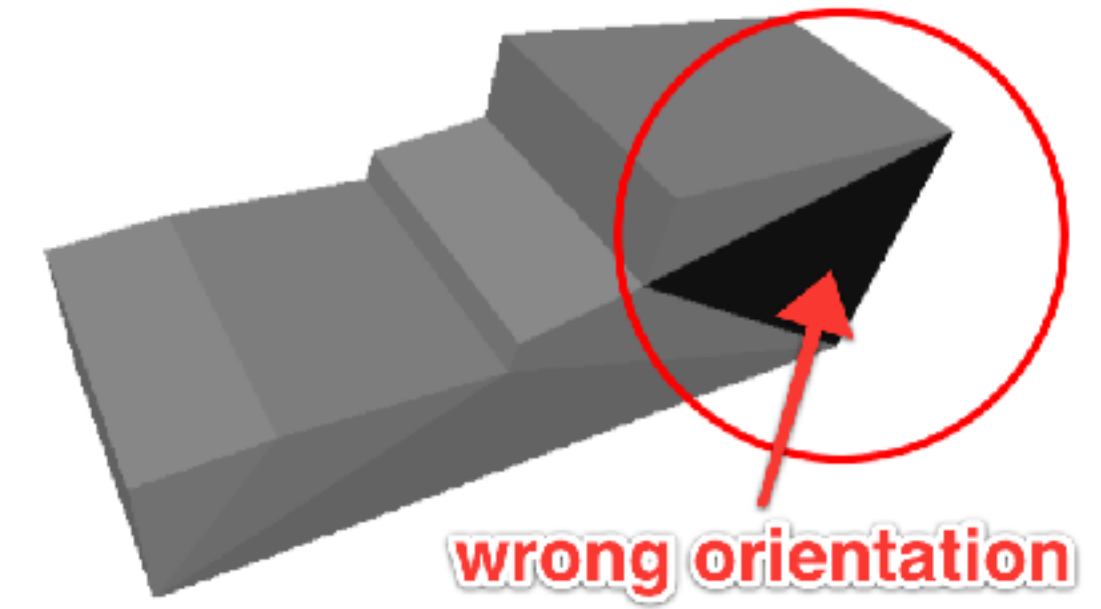
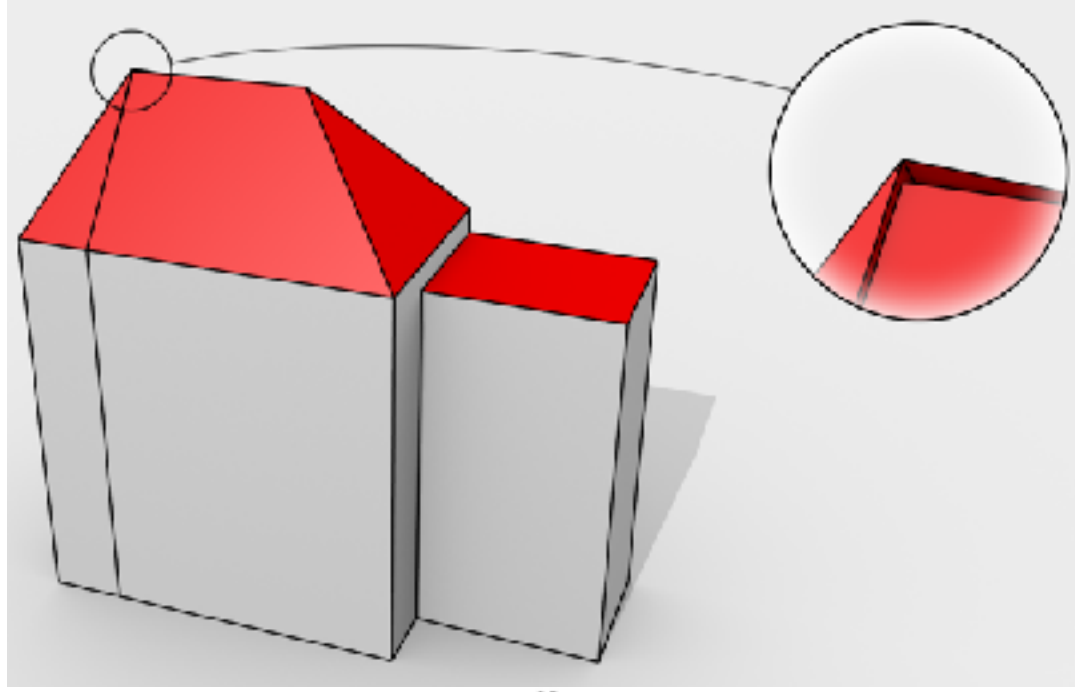


1. How many interior shells does this “squared torus” have?
2. How many surfaces form its outer shell?
3. List them

**Do current 3D city models
often contain these
geometric errors?**

tl;dl: **YES.**

Errors are **very** common in 3D city models: my wall of shame

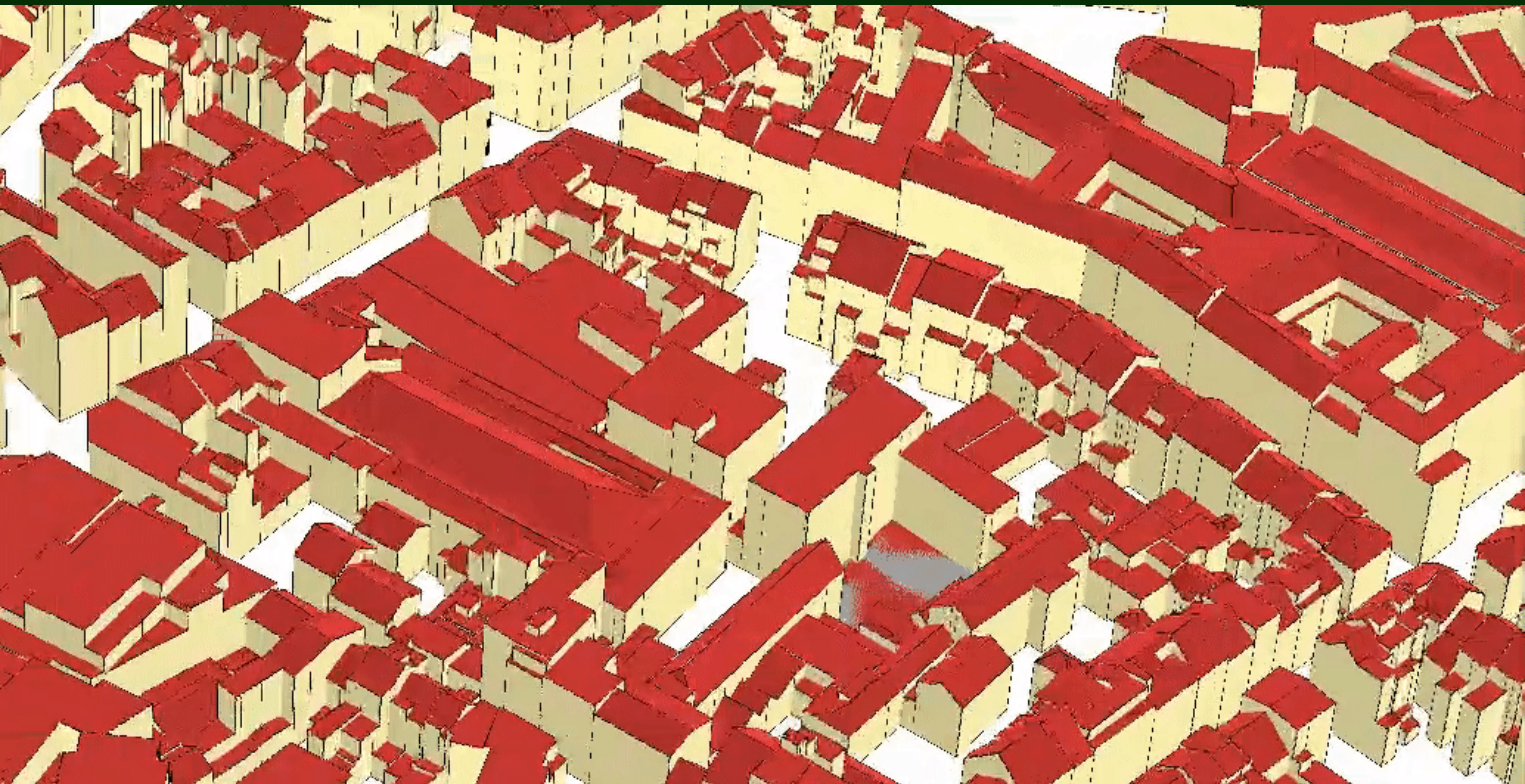


Should we care?

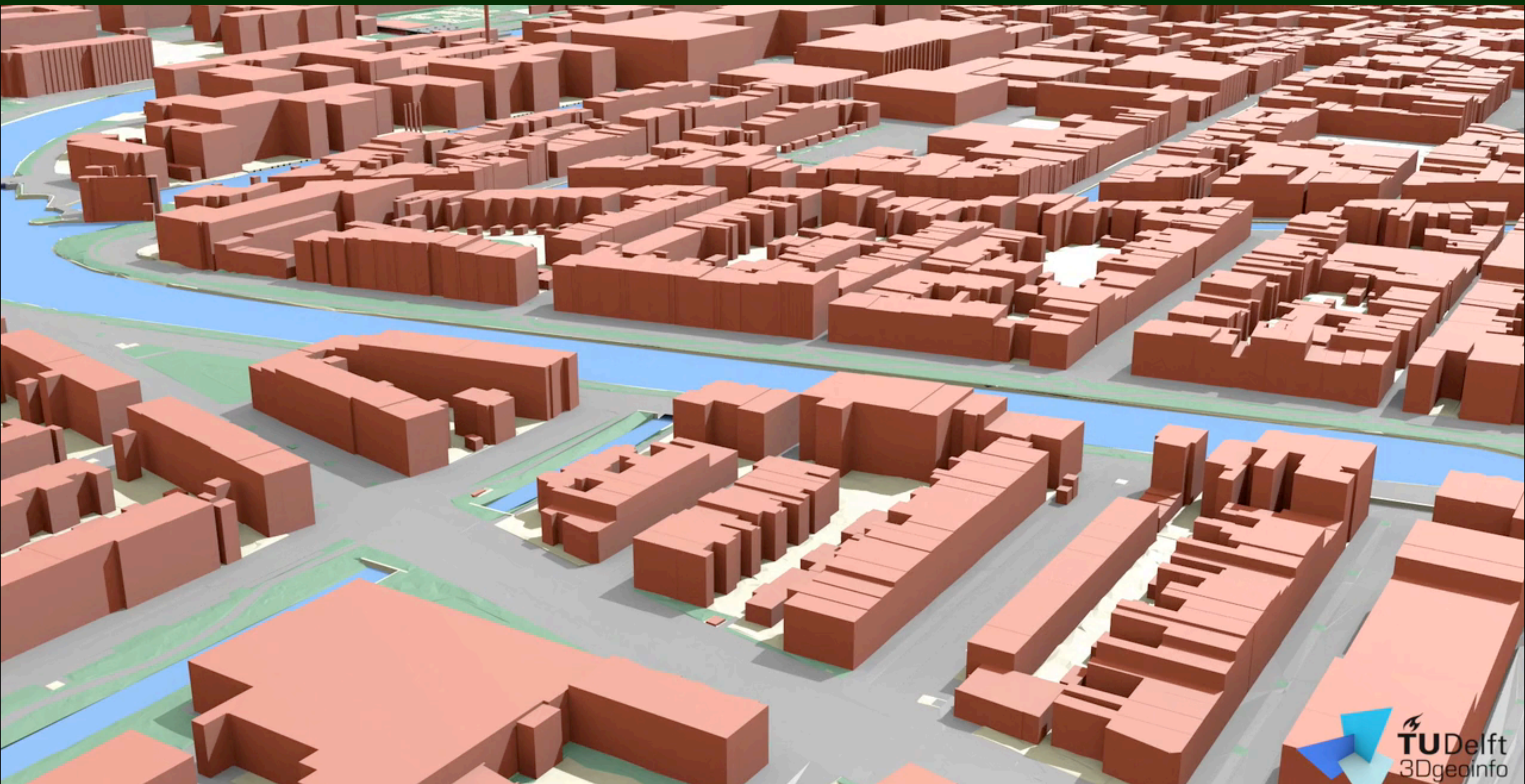
YES!

(especially if you want to pass this course 😅)

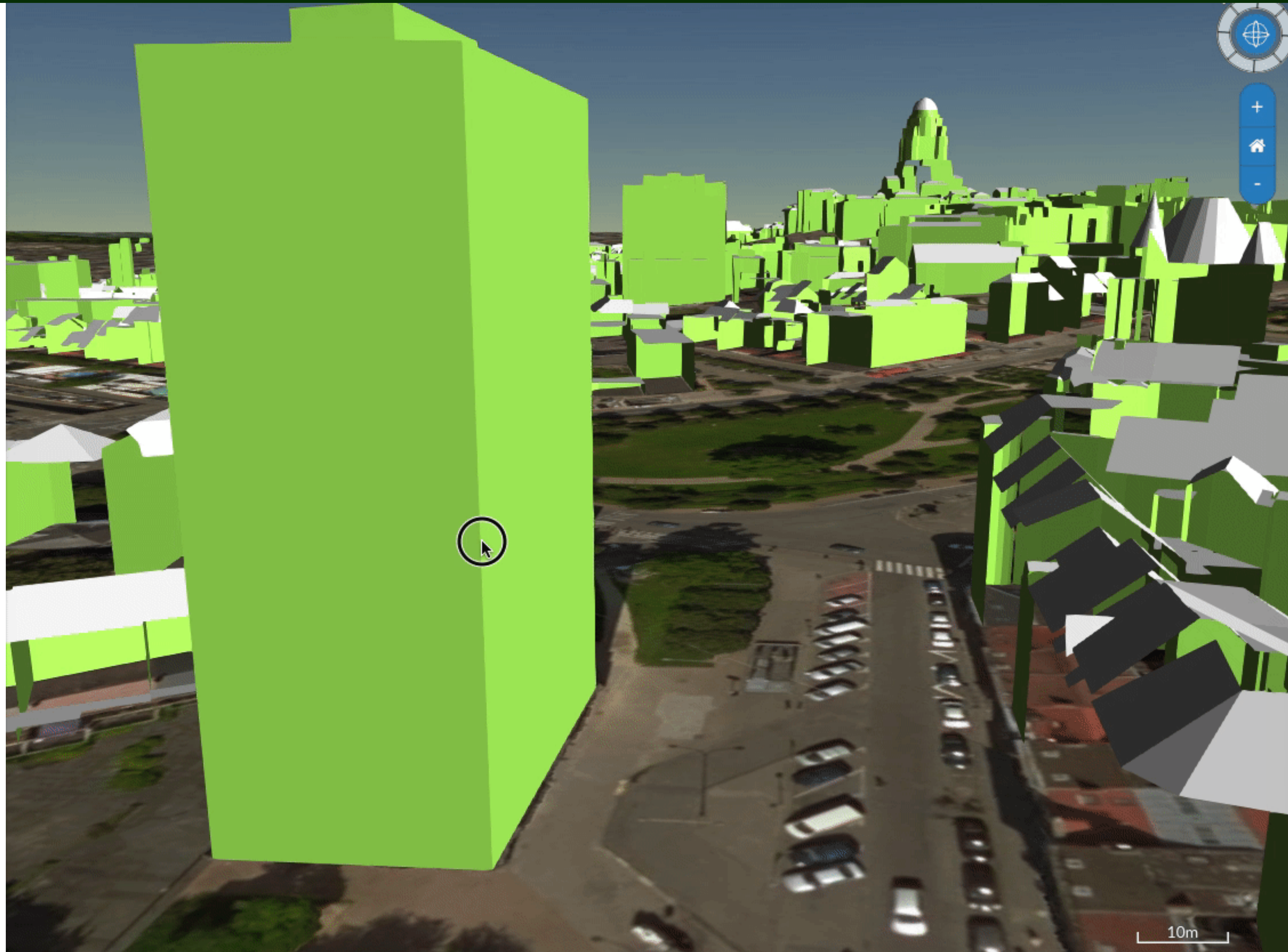
Visualisation – duplicated surfaces == annoying



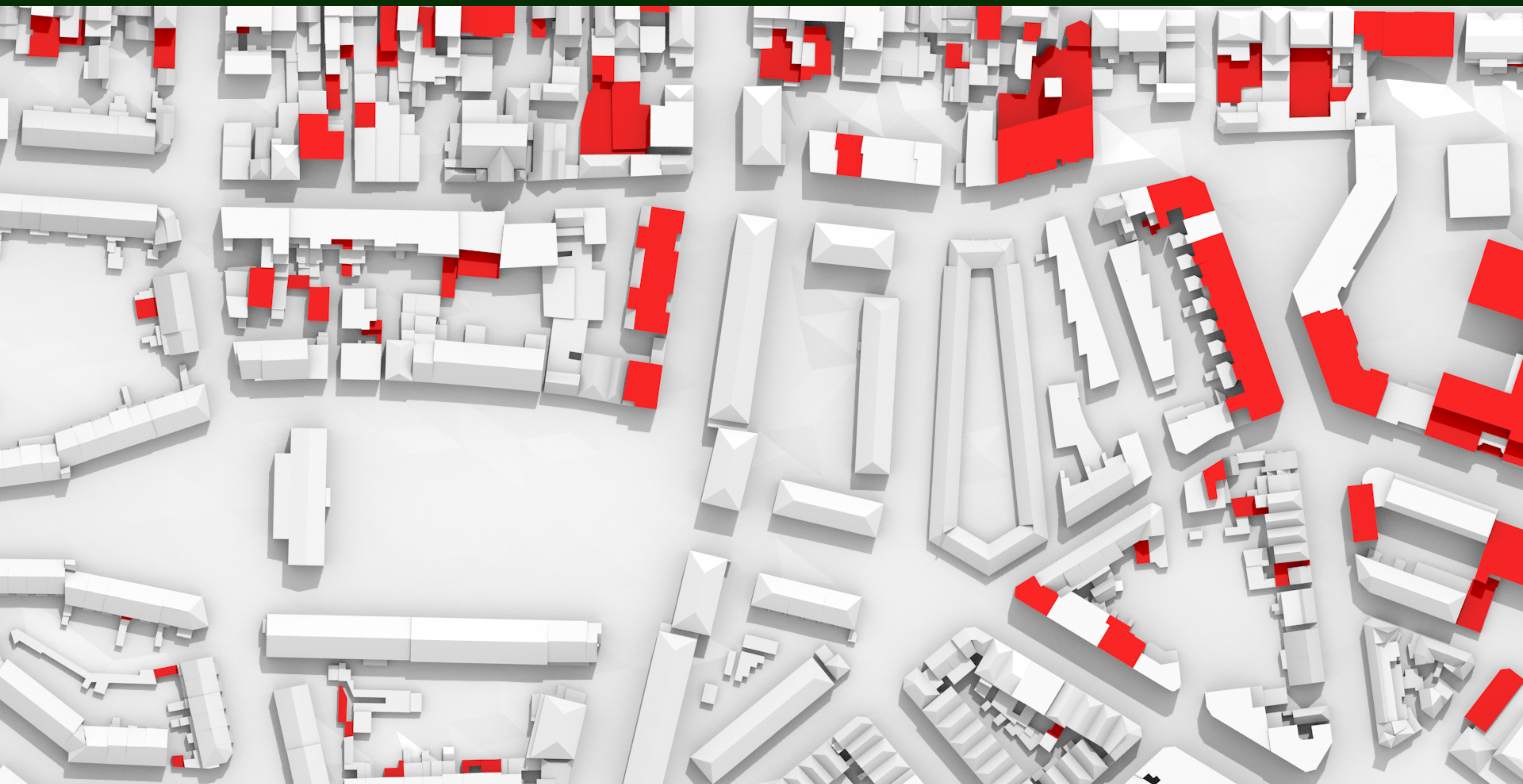
Visualisation – duplicated surfaces == distracting



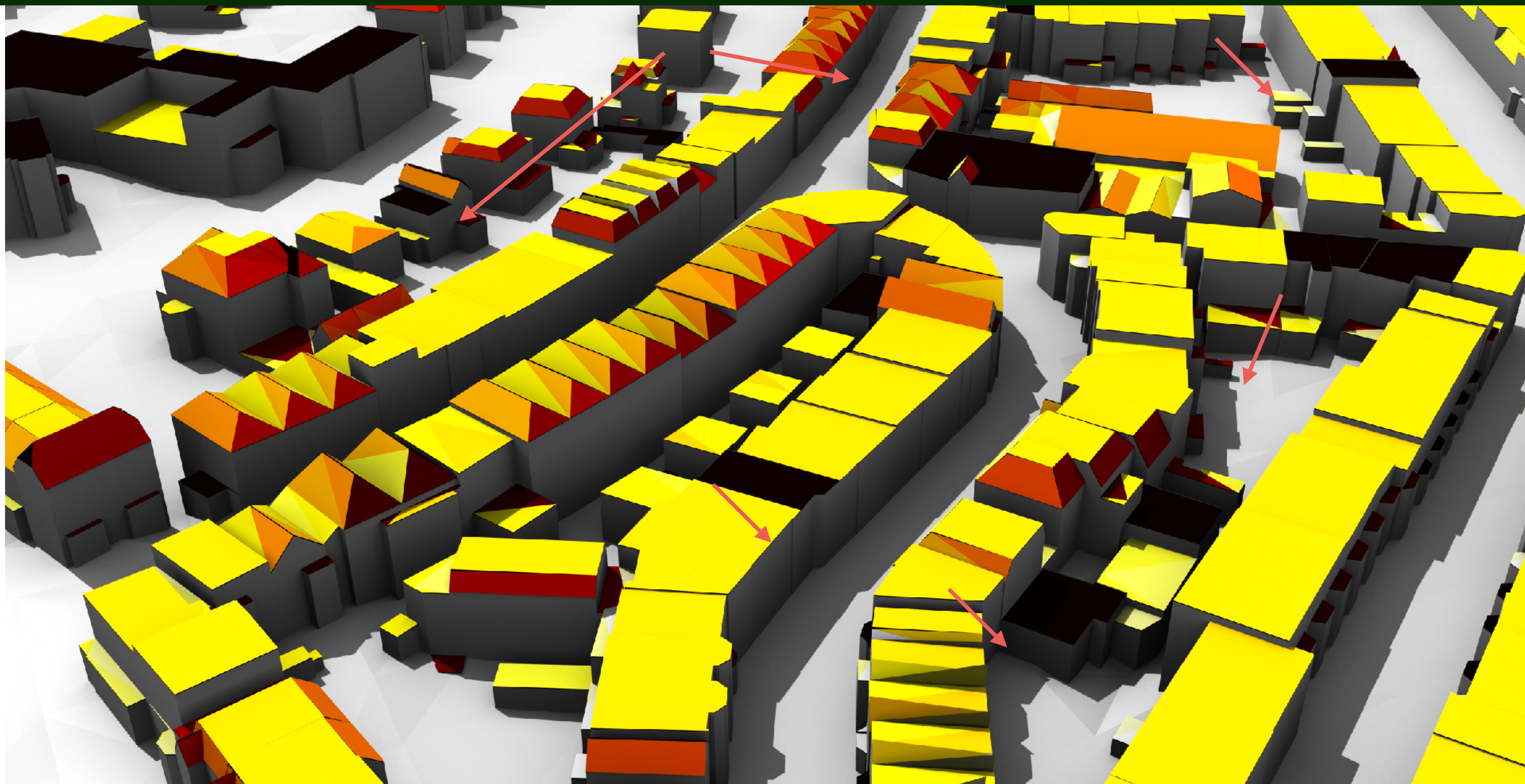
Visualisation – wrong orientation == missing faces



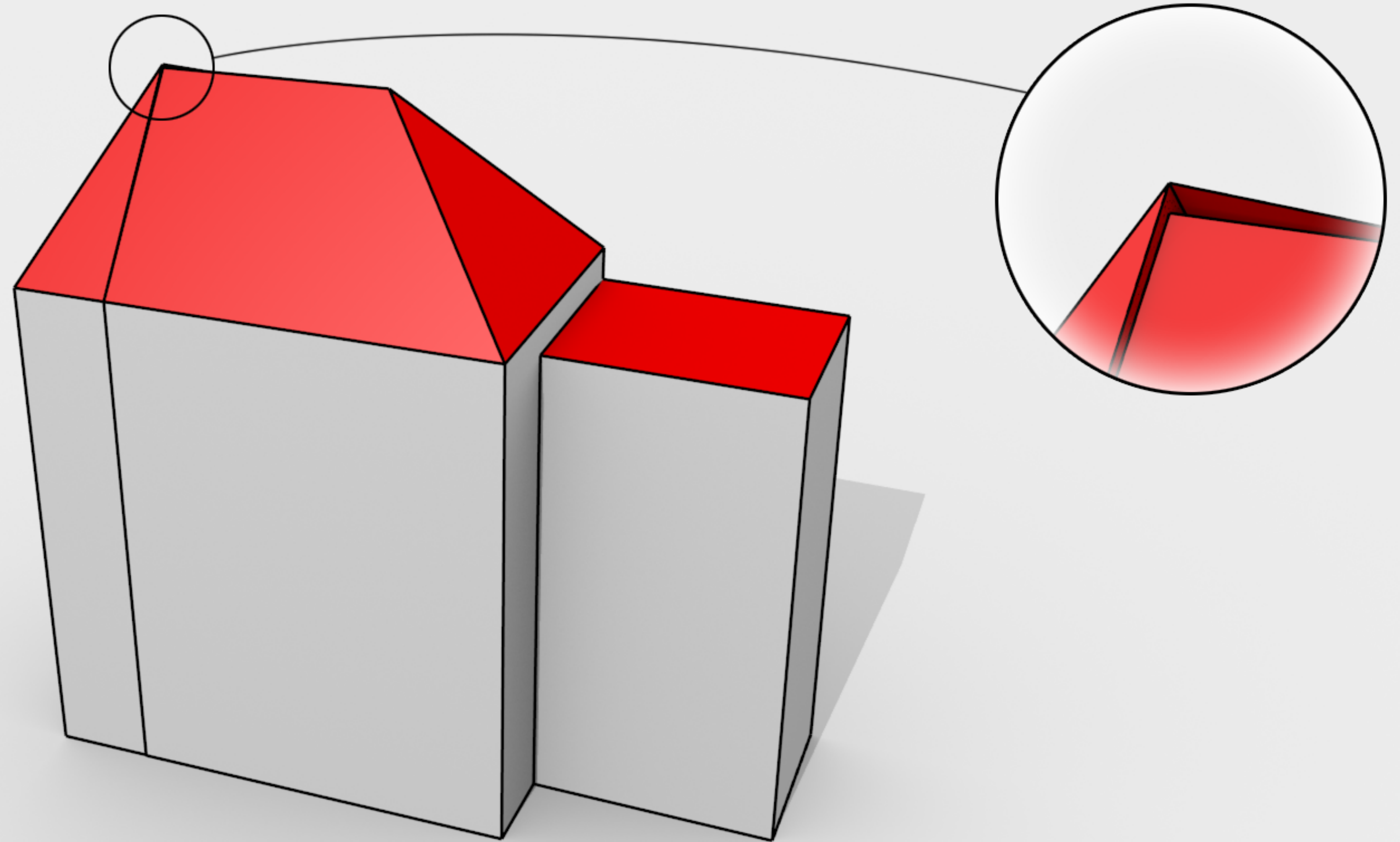
Visualisation – wrong orientation of surfaces (red ones)



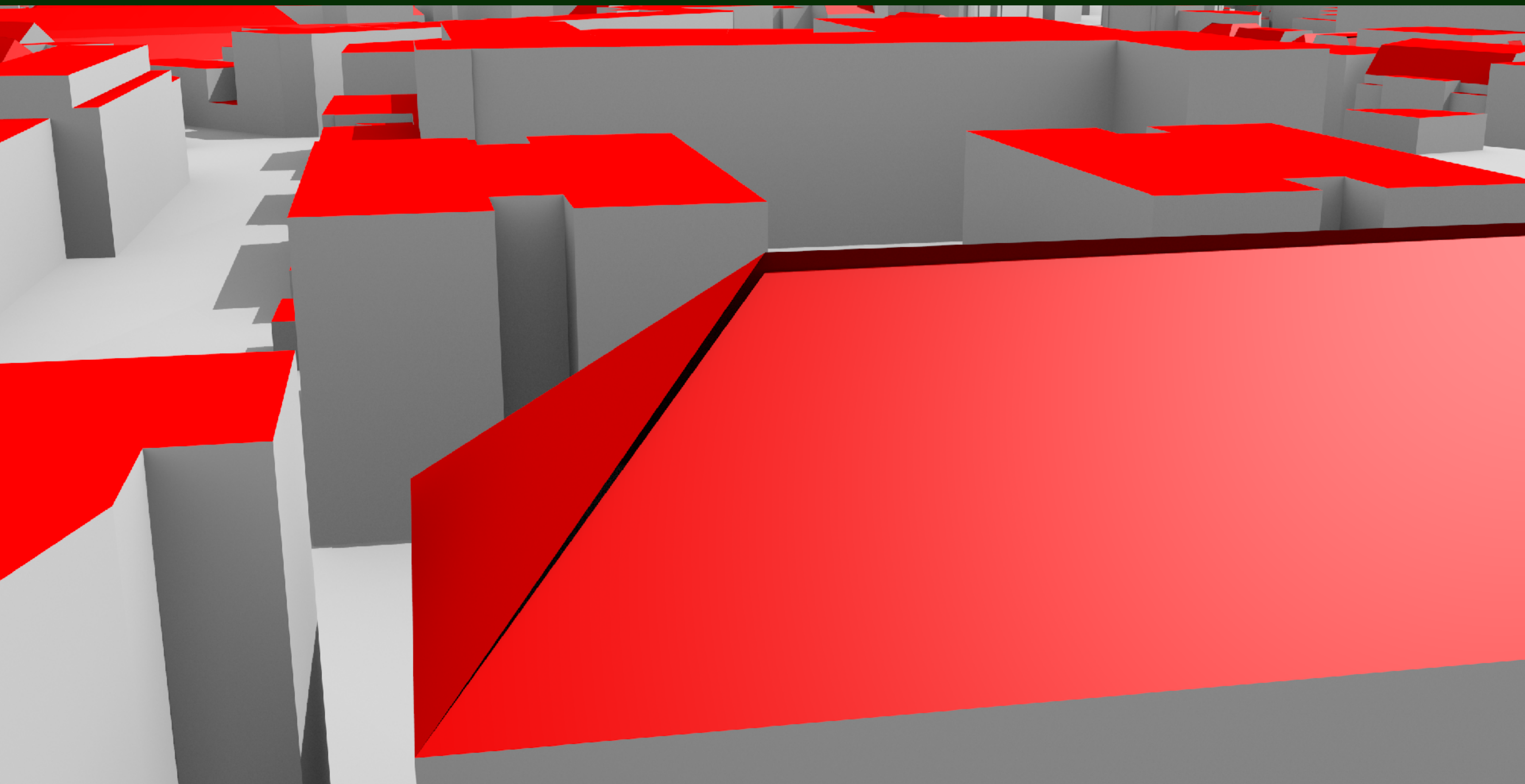
Solar potential – wrong orientation == no potential assigned



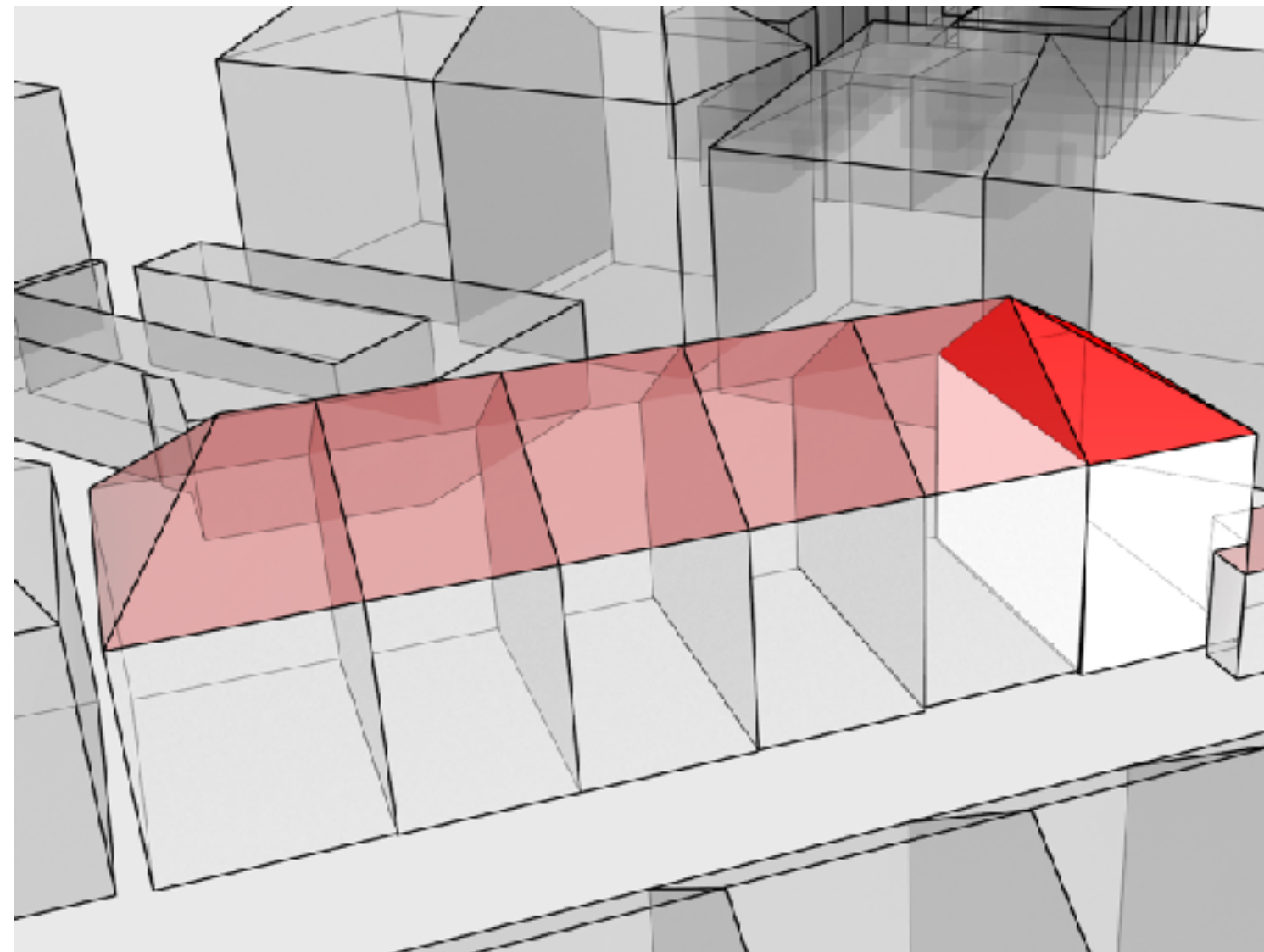
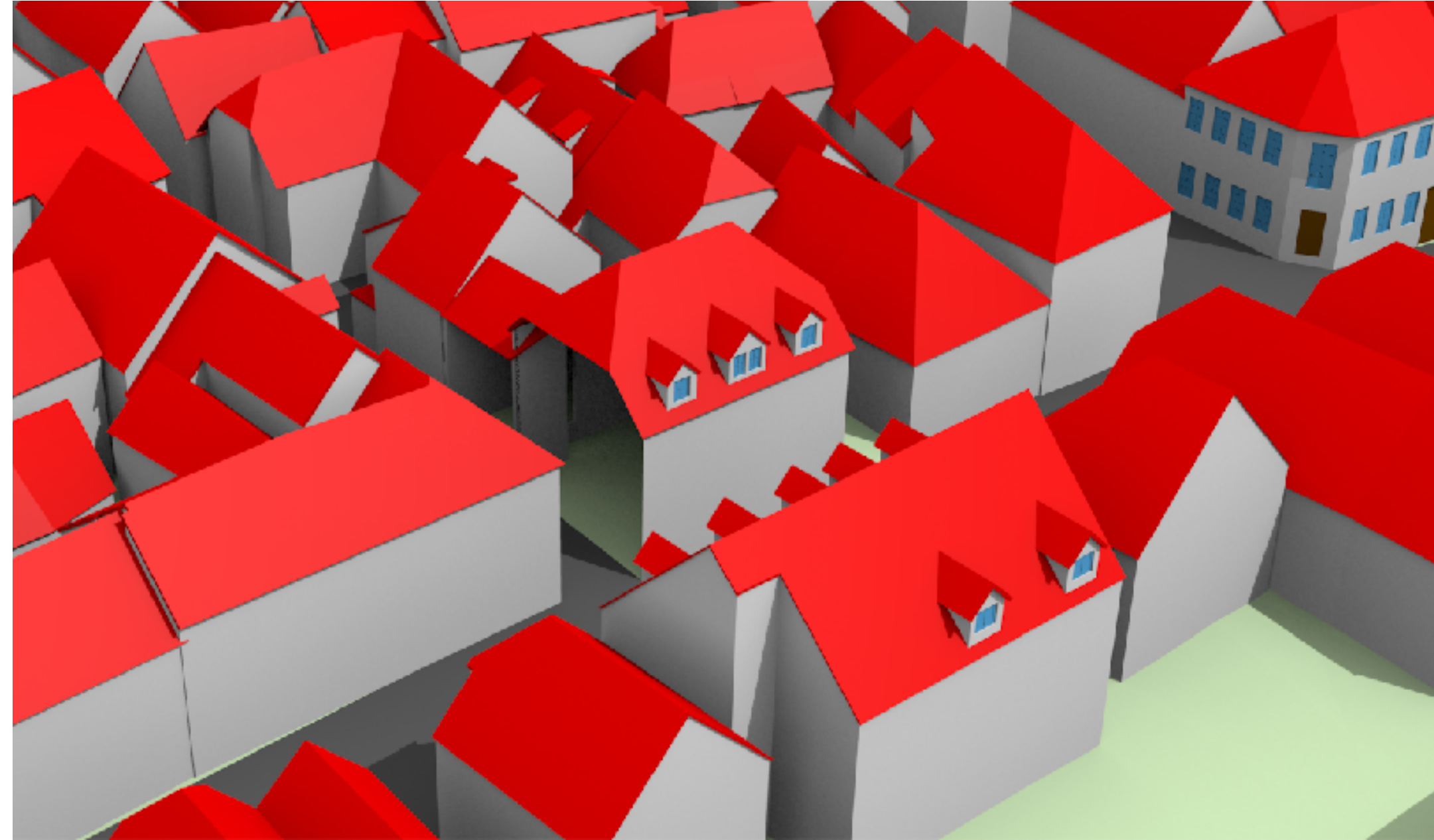
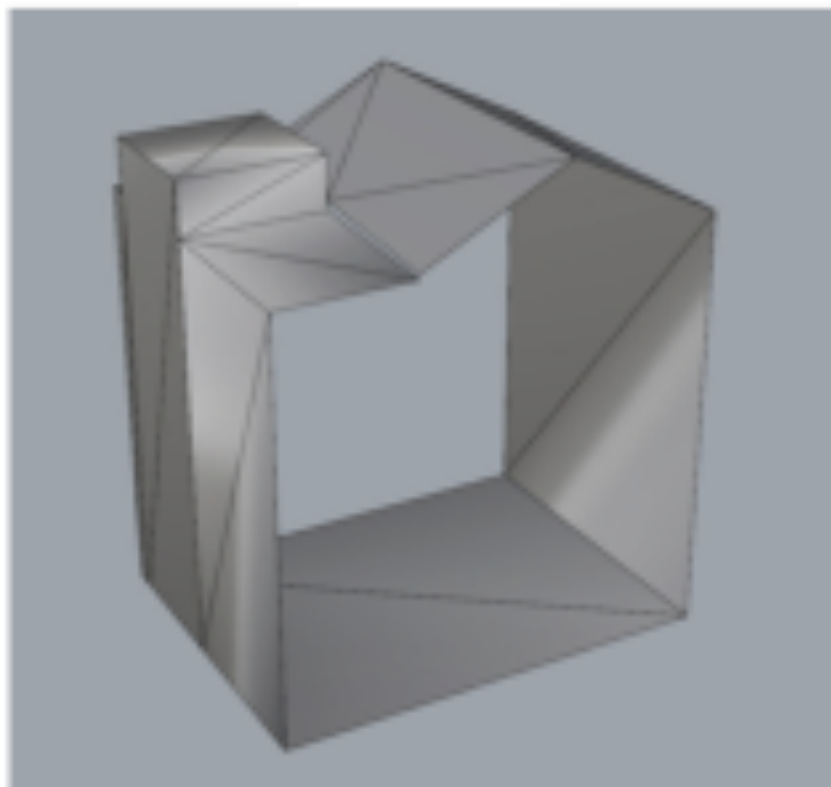
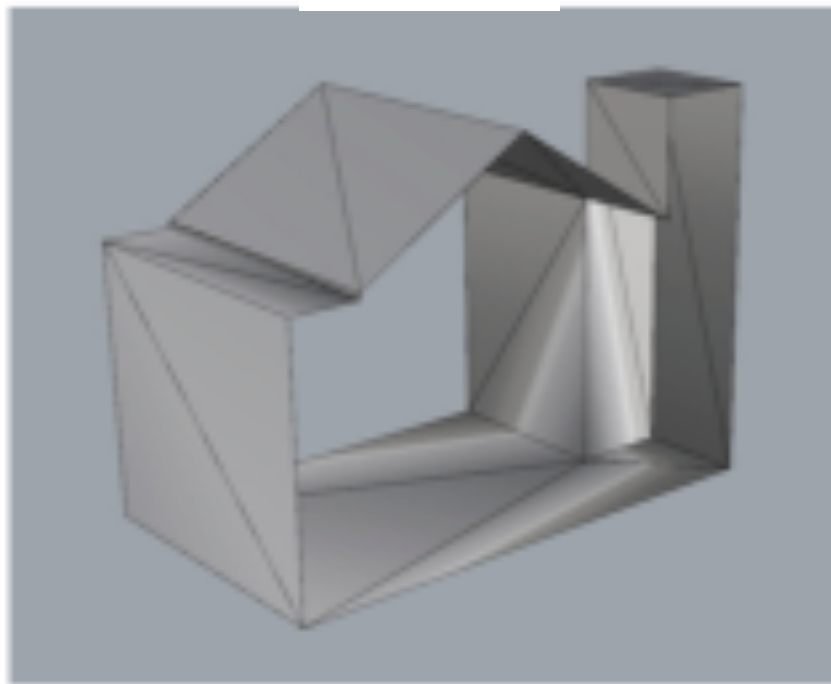
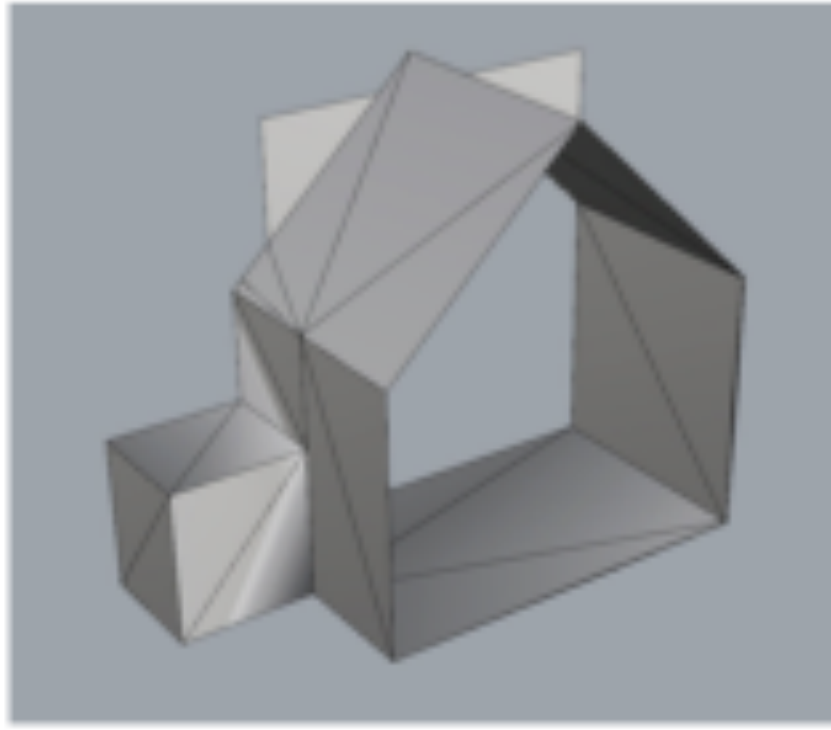
Volume calculation – tiny problems == some methods do not work



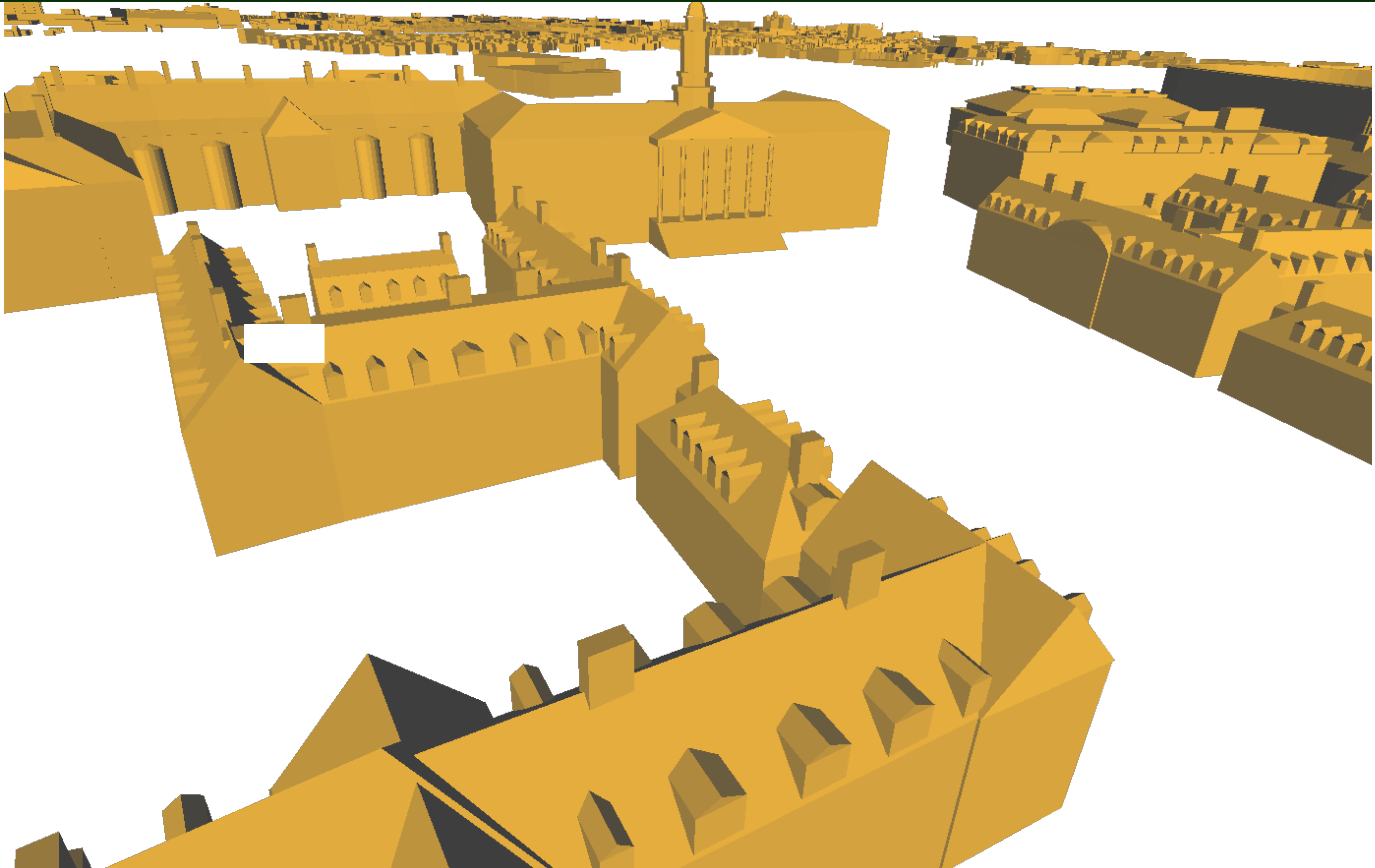
Volume calculation – tiny problems == some methods do no work



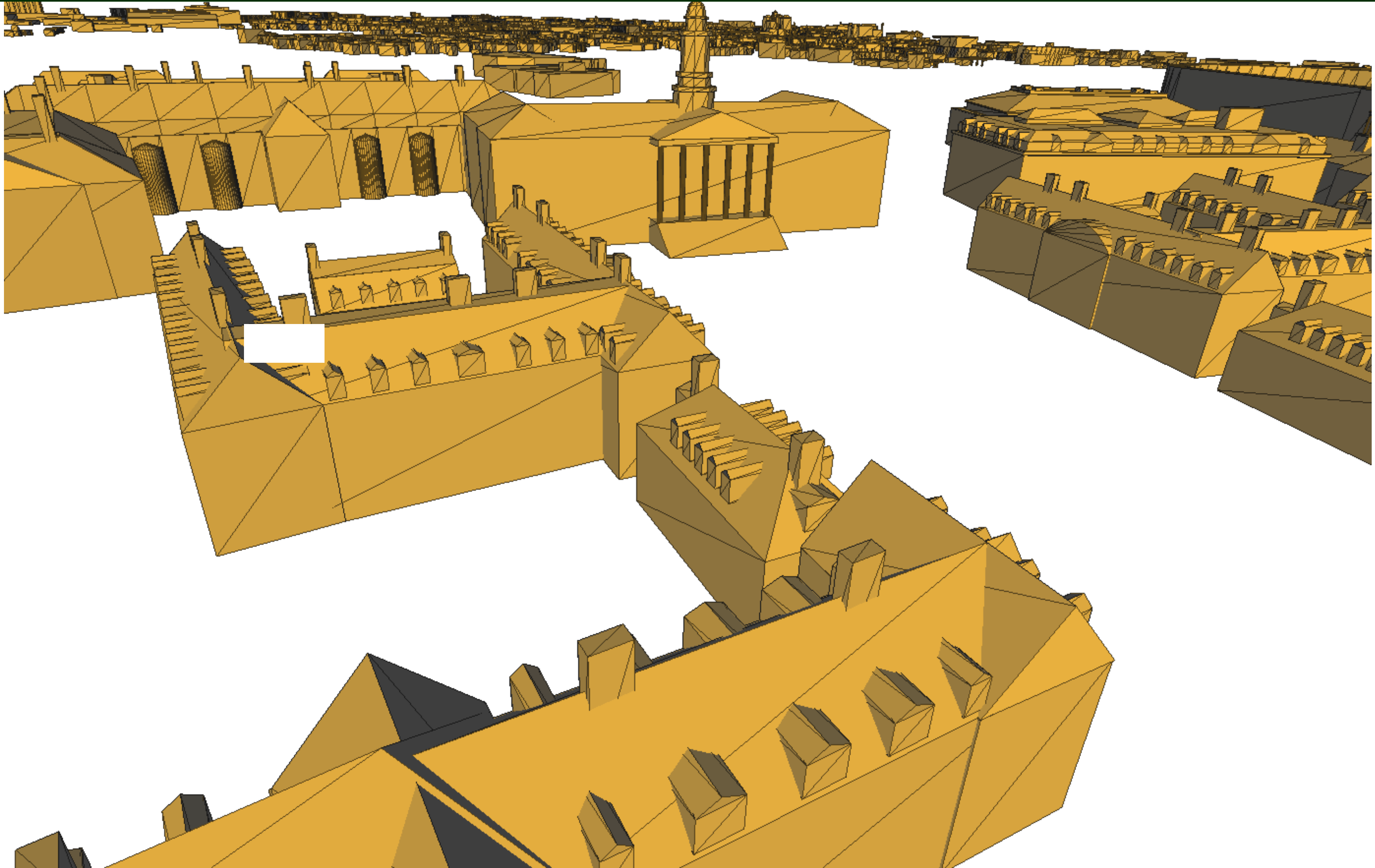
Volume calculation – big problems == some methods do no work



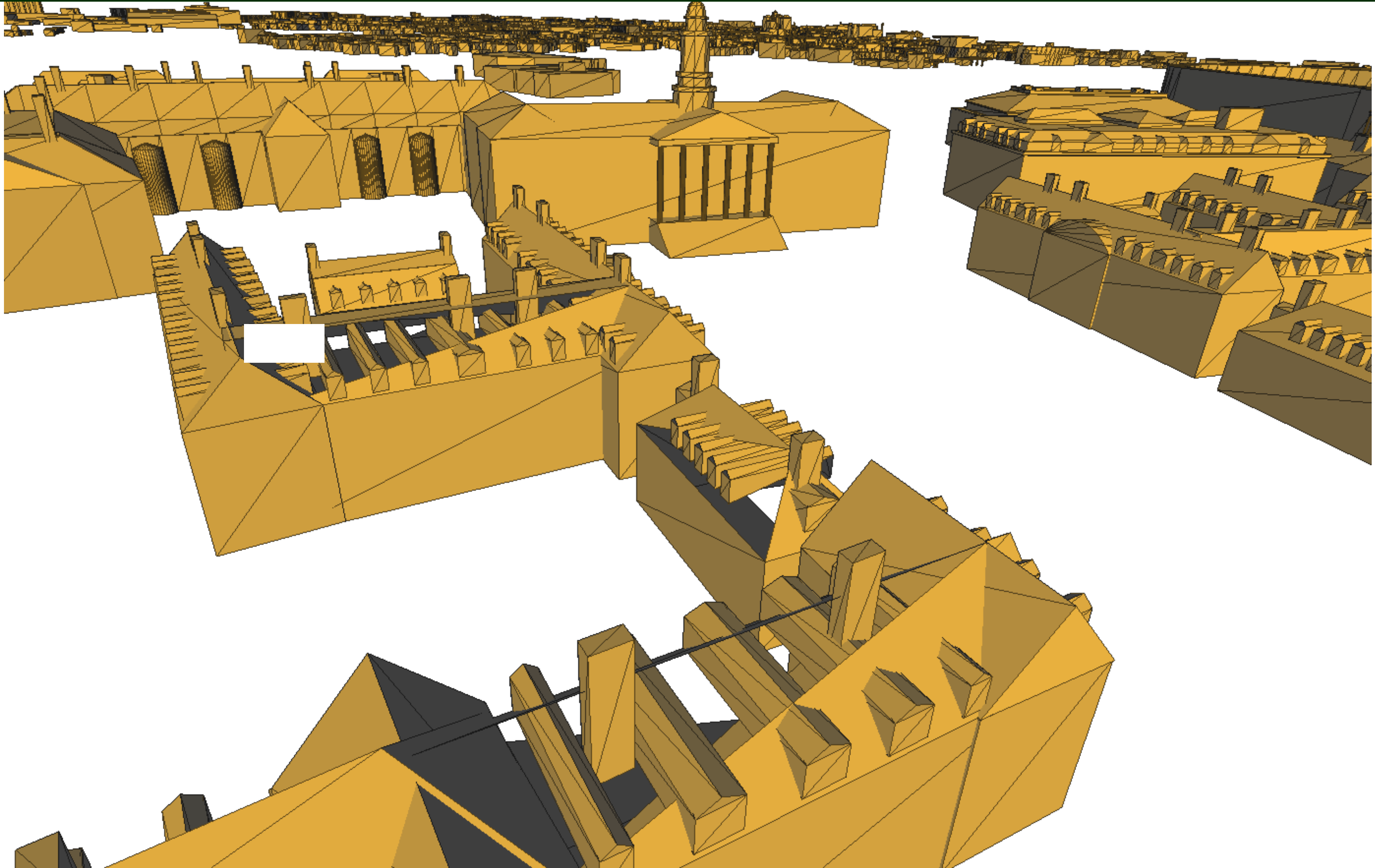
Volume calculation – superstructures not topologically connected



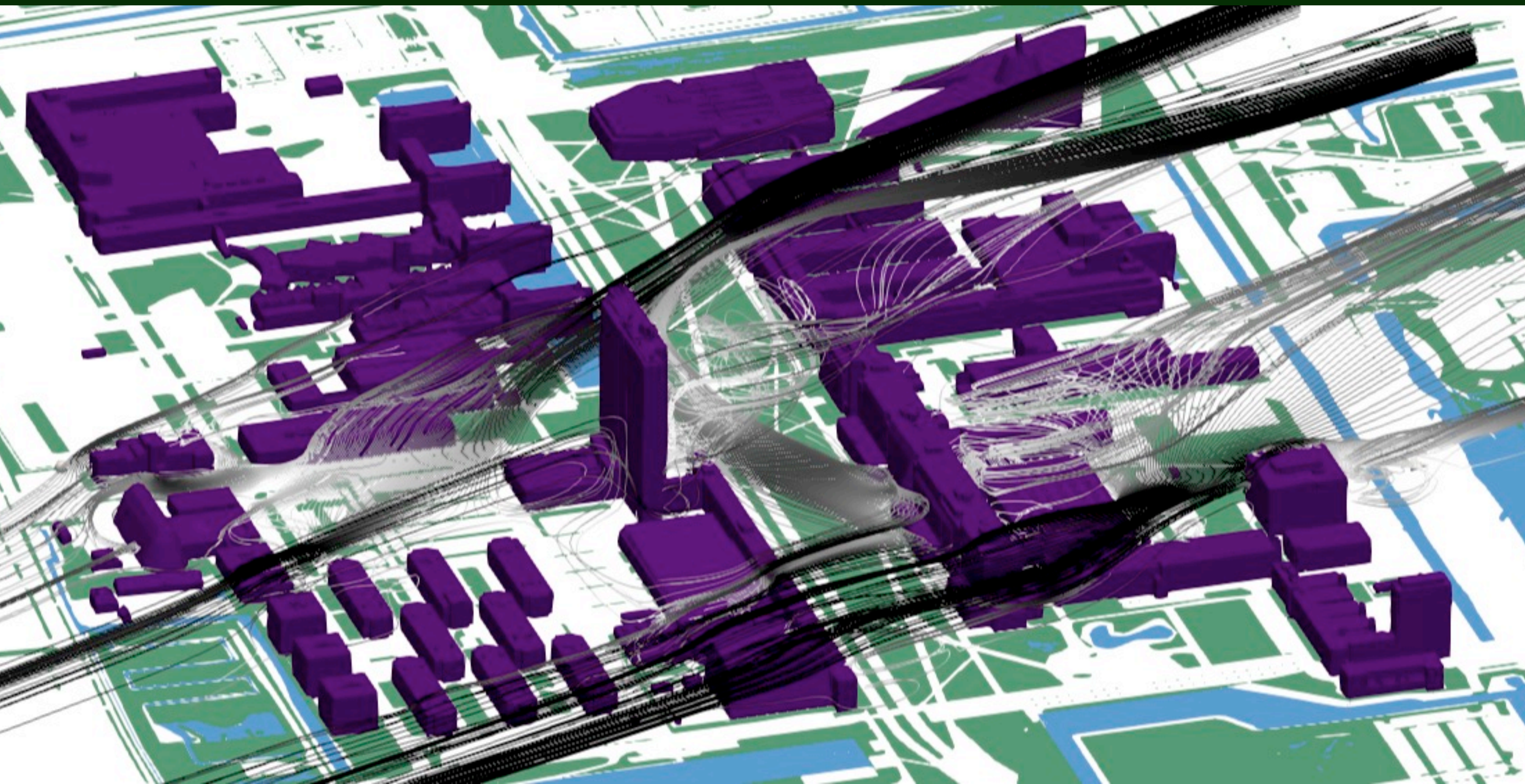
Volume calculation – superstructures not topologically connected



Volume calculation – superstructures not topologically connected



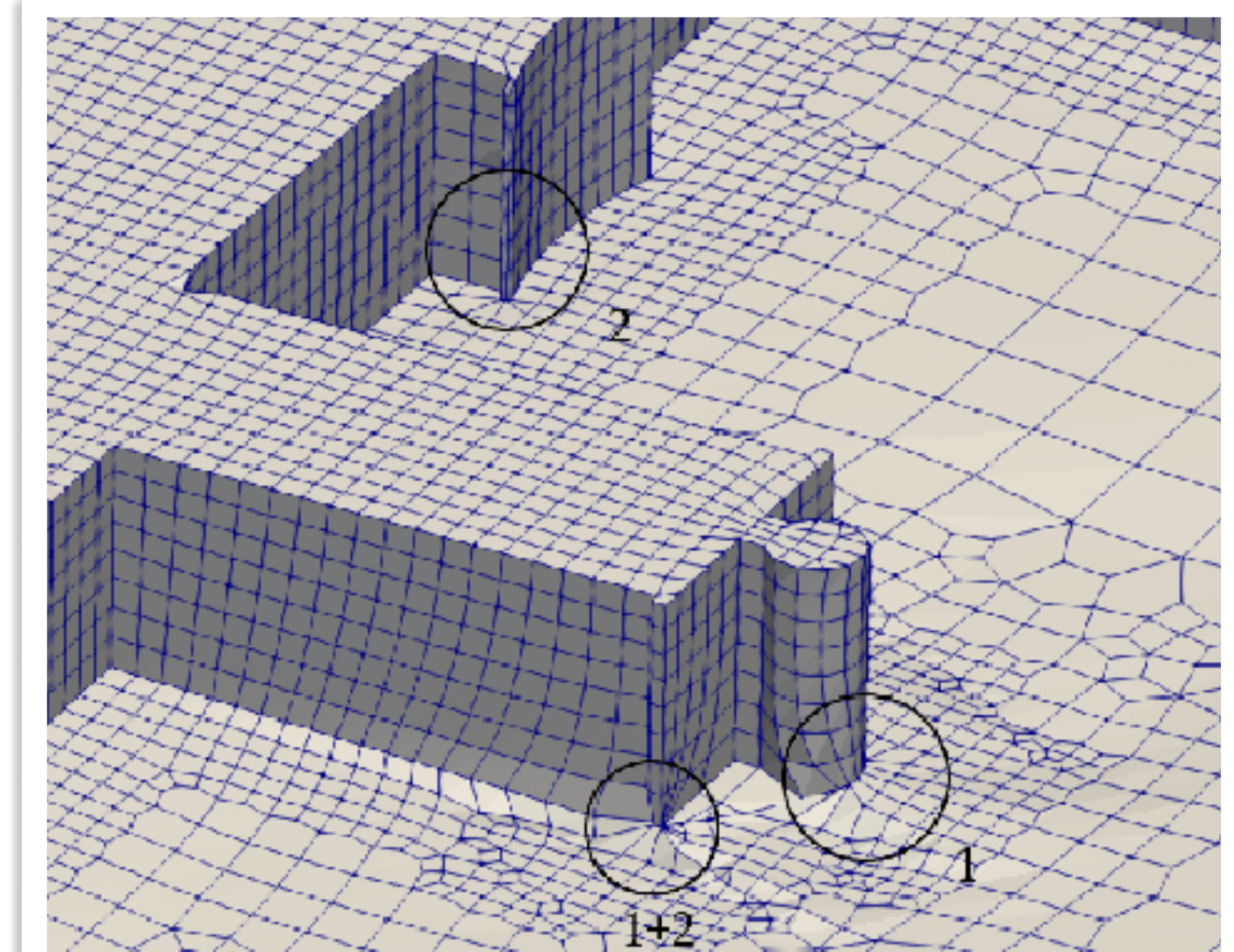
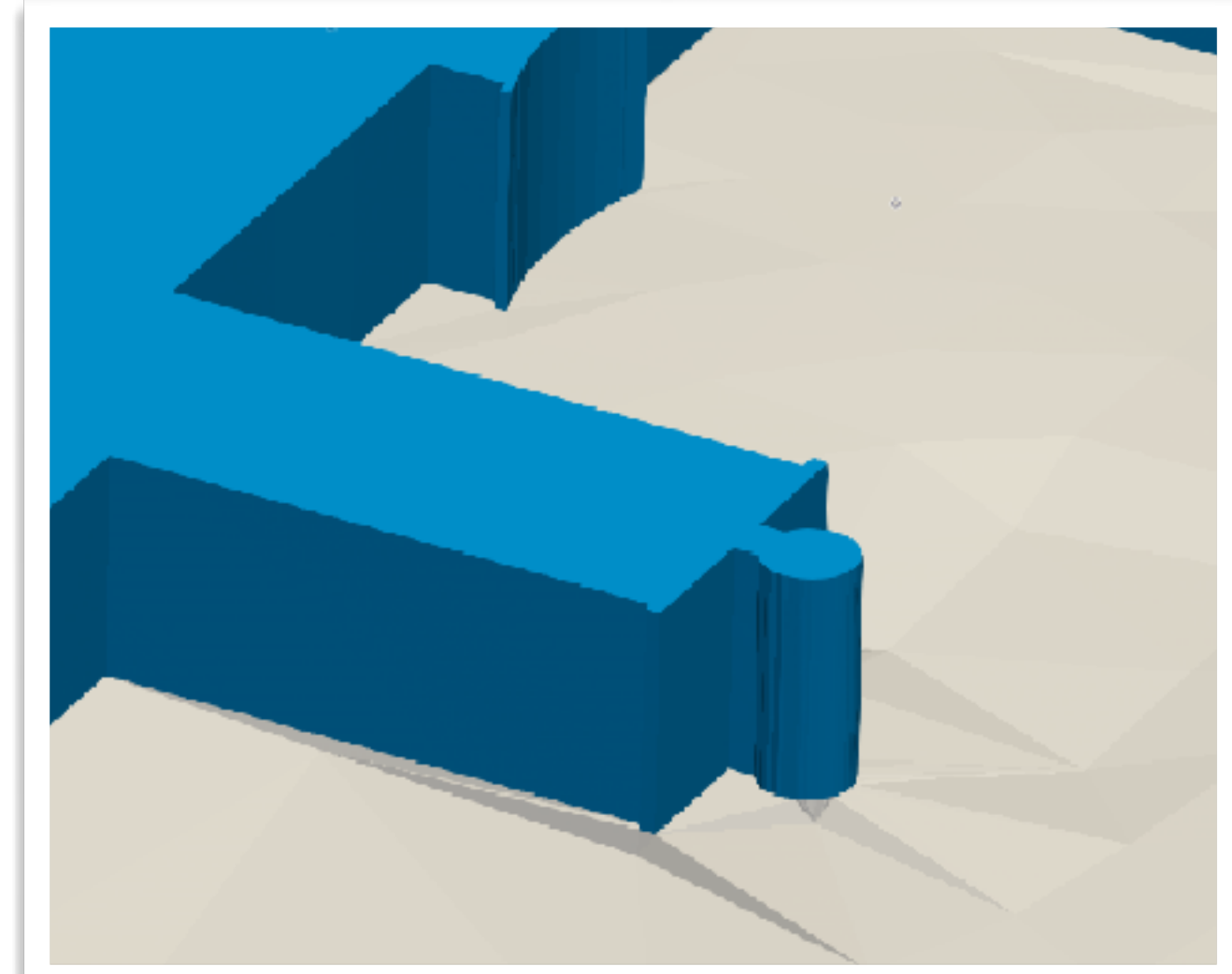
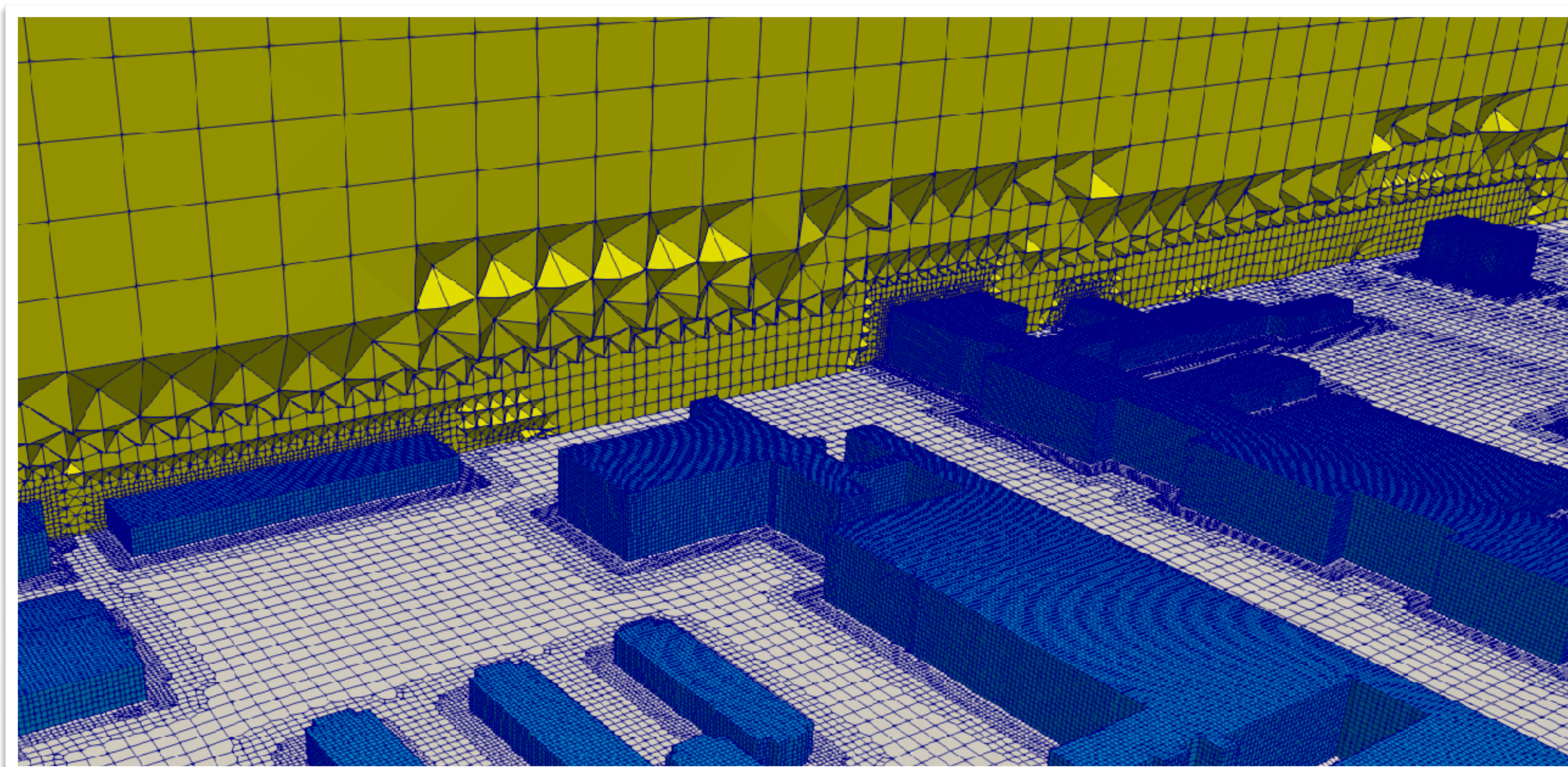
Advanced simulations – wind comfort for pedestrians



Advanced simulations – CFD methods: subdivision of volume (mesh)

computational fluid dynamics

GE05021: Modelling wind and dispersion in urban environments (Q5!)

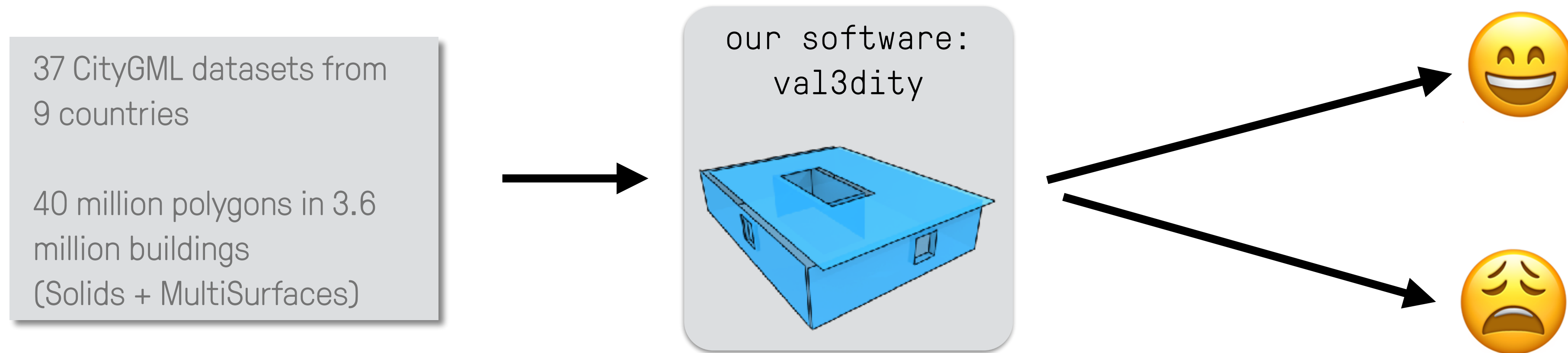


Advanced simulations – very strict input requirements

- no holes
- no intersection
- triangles must have certain shape (no *slivers*)
- a “perfect” model is thus required
- in practice, that usually means days or even weeks of (semi-)manual repair for an area like the TUDelft campus



An old-ish study we made



- 37 datasets (3D cities) from 9 countries
- 40 million polygons in 3.6 million buildings

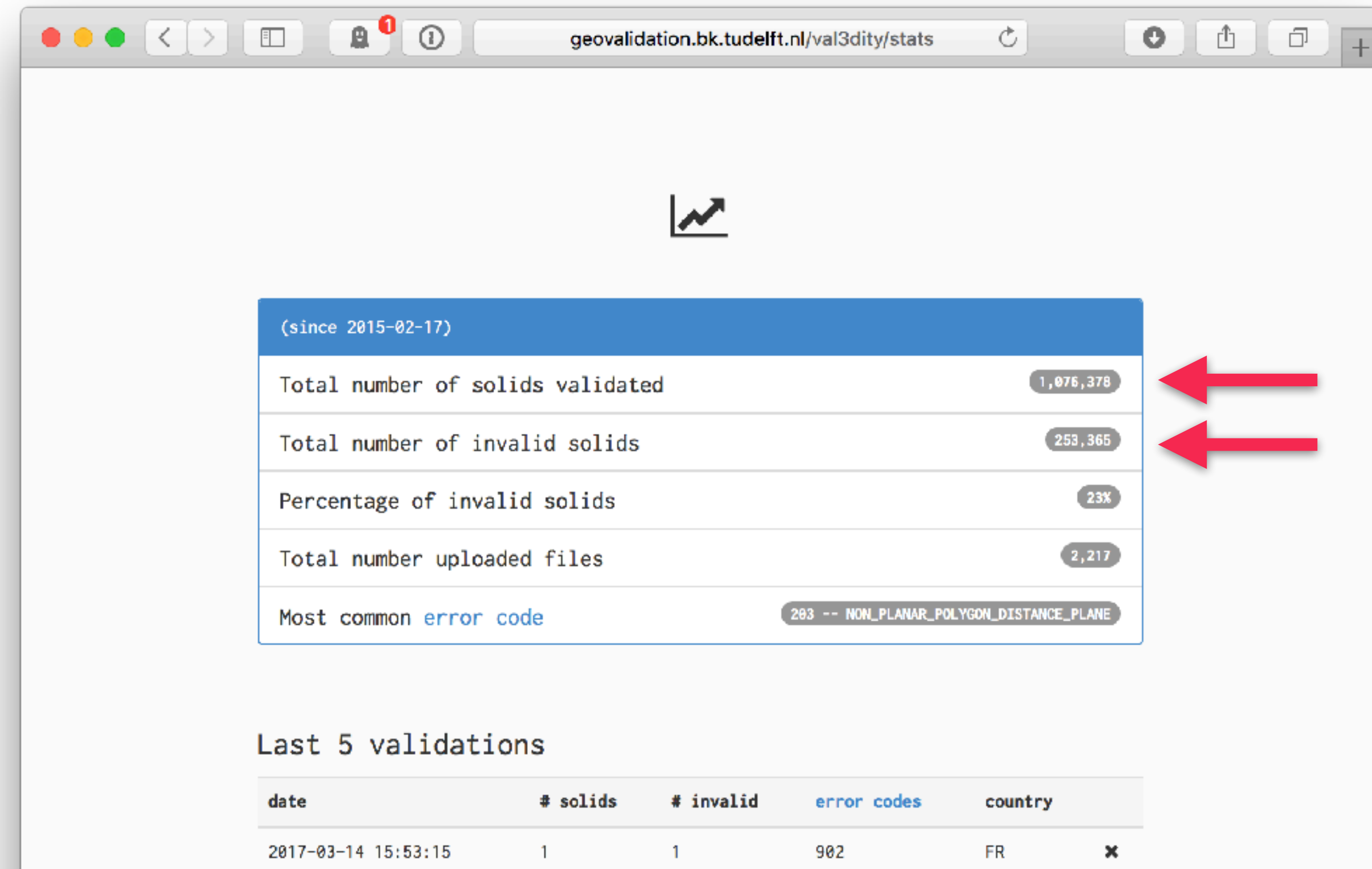
Many cities around the world have 3D models openly available

Cities/regions around the world with open datasets									
dataset	country	year	building LoD	other classes	textures	acquisition	formats	notes	
Adelaide	Australia	2015	LoD1/LoD2	Terrain	true		.3DS with JPEG textures, Blender, FBX		
American cities	USA	2019	LoD1		false		CityJSON, CityGML	125 million buildings, separated by state	
Austin	USA	2013	LoD2		false		KMZ		
Berlin	Germany	2013	LoD2		true		CityGML, 2D Shape, 3D Shape - PolygonZ, 3D Shape - Multipatch, KMZ, DXF, DWG, 3DS, ESRI FGDB	Released in 2015	
Bogotá	Colombia	2019	LoD1		false		GeoPackage, PostgreSQL, GeoDataBasc		
Bordeaux	France	2012	LoD2		true		3DS	Version without textures available here	
Boston	USA	2017	LoD1+LoD2	Terrain	false		OBJ, DWG, MAX, Shapefile		
Brussels	Belgium	2014	LoD2		false		CityGML, DWG, DGN, GML, Shapefile, SketchUp		
Calgary	Canada	2023	LoD2		false			GDB, KMZ, Shapefile	
Cambridge, MA	USA		LoD1		false			DXF, Shapefile, COLLADA	
Dresden	Germany	2009	LoD1/LoD2/LoD3		partially			CityGML	
Espoo	Finland	2019	LoD1-3		true			CityGML Many classes from CityGML: water body, terrain, landuse, etc.	
Estonia	Estonia	2021	LoD1+LoD2		false			Geodatabase, CityGML, OBJ Updated frequently	
Fredericton	Canada	2016	LoD2		true			KMZ	
Greater Geelong	Australia	2016	LoD1/LoD2	Terrain	false			KML	
Hamburg	Germany	2017	LoD1 and LoD2			Cadastral footprints + LiDAR		CityGML	
Helsinki	Finland	2016	LoD2		true			CityGML, 2D Shape, 3D Shape - PolygonZ, 3D Shape - Multipatch, KMZ, DXF, DWG, 3DS, ESRI FGDB	
Ingolstadt (by research project)	Germany	2020	LoD3	mainly buildings	true	manually modeled based on MLS		CityGML LoD3 Modeling Guide	
Japan (PLATEAU)	Japan	2021	LoD1+LoD2		false			Shapefile, CityGML, 3Dtile/geopackage in total 56 cities in Japan	
Linz	Austria	2011	LoD2		false			CityGML	

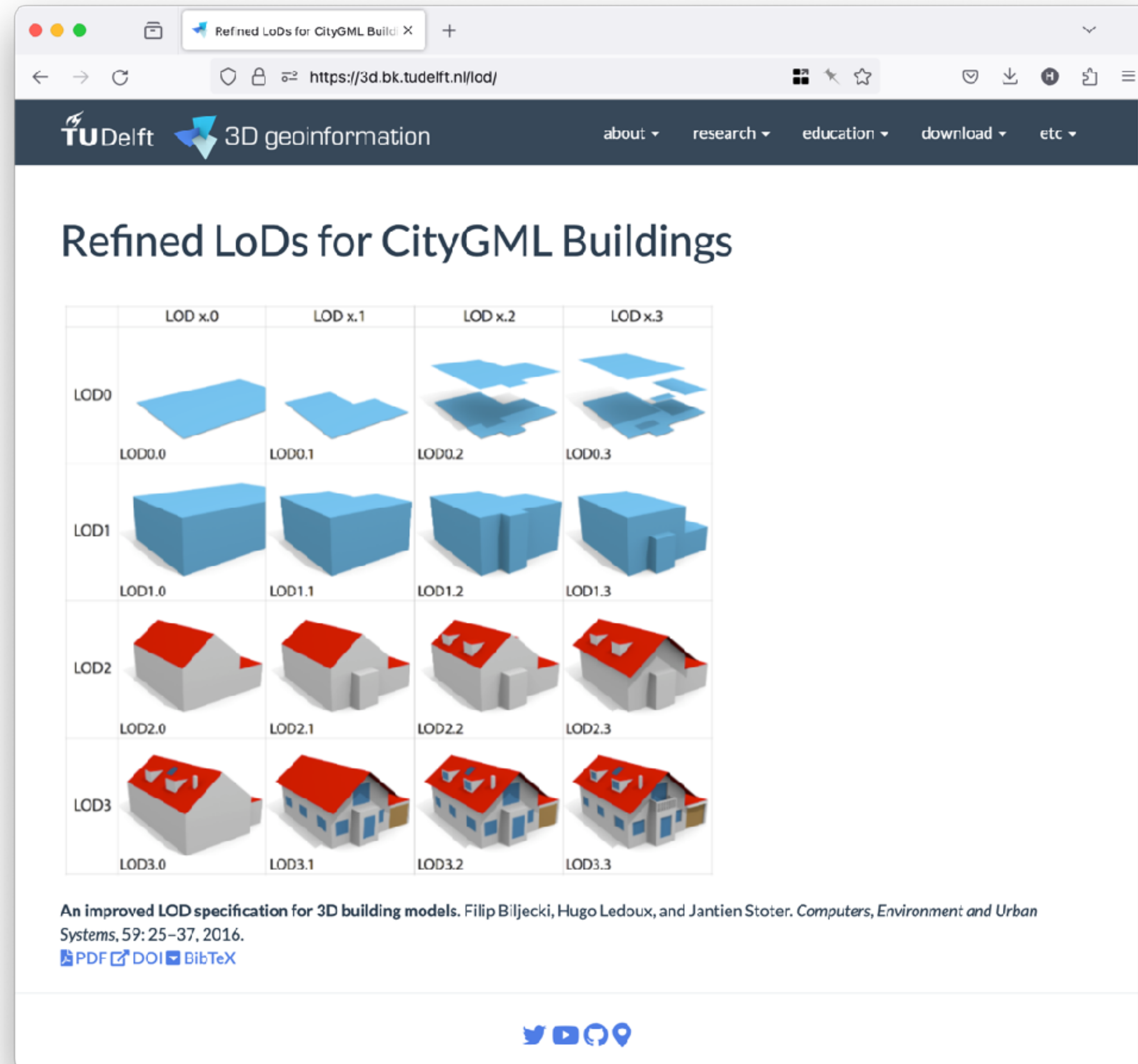
Summary of results

- No 3D city dataset is 100% valid, not even LoD1.2 😞
- Many simple errors, eg:
 - repeated vertices
 - non-planar polygons (most common error!)

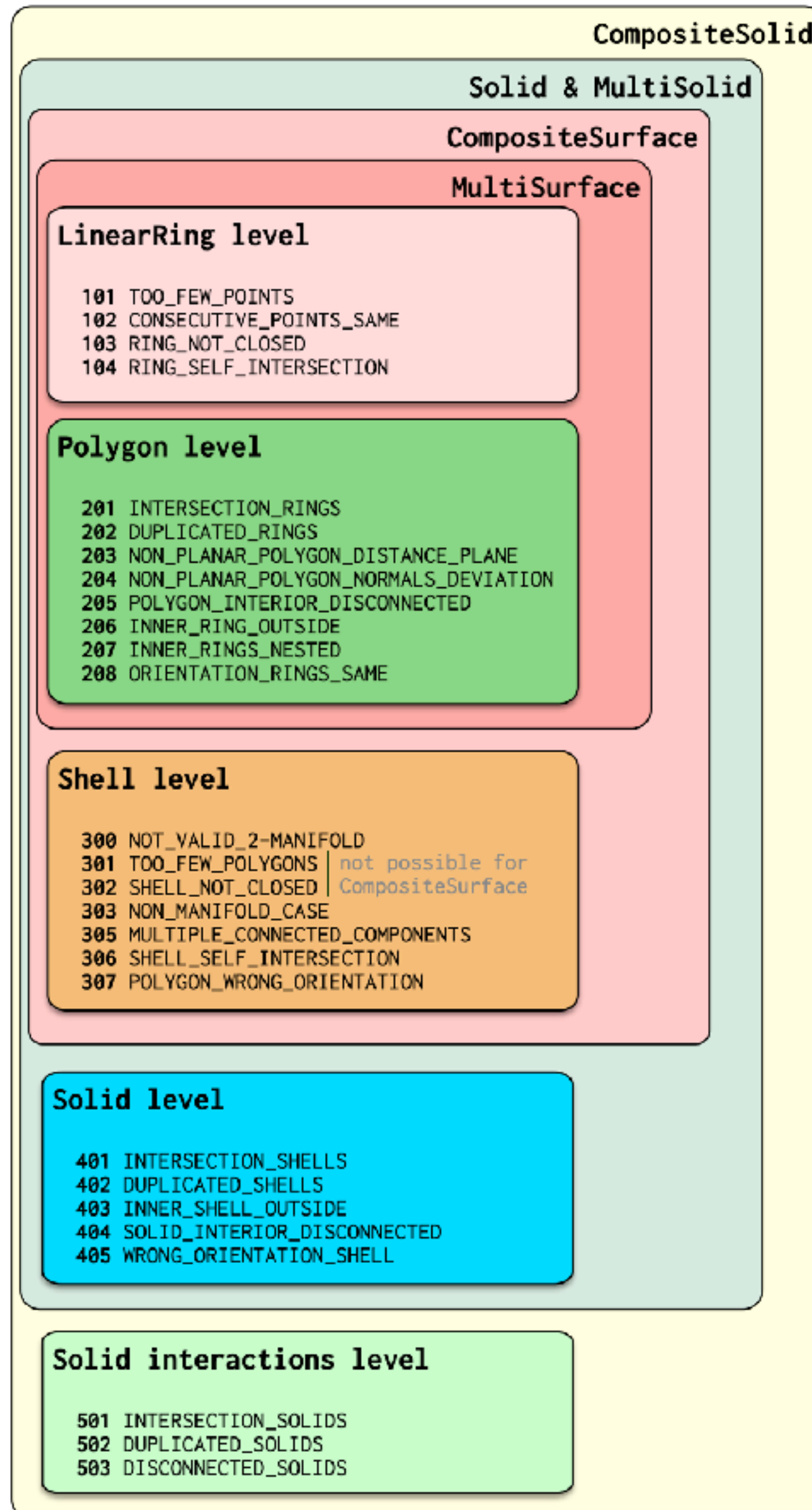
Notice that these are
often not visible



The Levels-of-details (LoDs)



val3dity reports with a specific error code



CityGML Objects

- 601 BUILDINGPARTS_OVERLAP

IndoorGML Objects

- 701 PRIMAL_CELLS_OVERLAP
- 702 DUAL_VERTEX_OUTSIDE_PRIMAL_CELL
- 703 PRIMAL_DUAL_XLINKS_ERROR
- 704 PRIMAL_DUAL_ADJACENCIES_INCONSISTENT

Others

- 901 INVALID_INPUT_FILE
- 902 EMPTY_PRIMITIVE
- 903 WRONG_INPUT_PARAMETERS
- 904 FORMAT_NOT_SUPPORTED
- 906 PRIMITIVE_NO_GEOMETRY
- 999 UNKNOWN_ERROR

```
~/stacktmp/teaching/2024/iso19107 (0.041s)
val3dity cube_flipped.obj --report r.json
validation of 1 feature(s):
[=====] 100%

+++++ SUMMARY +++++
INVALID :(
++++
Input file type:
  OBJ
++++
Total # of Features:      1
  # valid:                0 (0.0%)
  # invalid:              1 (100.0%)
Types:
  GenericObject
++++
Total # of primitives:    1
  # valid:                0 (0.0%)
  # invalid:              1 (100.0%)
Types:
  Solid
++++
Errors present:
  303 -- NON_MANIFOLD_CASE
        1 primitive(s)
  307 -- POLYGON_WRONG_ORIENTATION
        1 primitive(s)
++++

Validation report saved to "/Users/hugo/stacktmp/teaching/2024/iso19107/r.json"
Browse its content:
==>http://geovalidation.bk.tudelft.nl/val3dity/browser/

~/stacktmp/teaching/2024/iso19107
```


Summary of results

ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-2/W1, 2016
11th 3D Geoinfo Conference, 20–21 October 2016, Athens, Greece

THE MOST COMMON GEOMETRIC AND SEMANTIC ERRORS IN CITYGML DATASETS

F. Biljecki ^{a,*}, H. Ledoux ^{a,*}, X. Du ^a, J. Stoter ^a, K. H. Soon ^b, V. H. S. Khoo ^b

^a 3D Geoinformation, Delft University of Technology, Delft, The Netherlands — (f.biljecki, h.ledoux, x.du-2, j.e.stoter)@tudelft.nl

^b Singapore Land Authority, Singapore — (soon_KEAN_huat, victor_khoo)@sla.gov.sg

KEY WORDS: CityGML, 3D city models, Quality control, Validation, Error

ABSTRACT:

To be used as input in most simulation and modelling software, 3D city models should be geometrically and topologically valid, and semantically rich. We investigate in this paper what is the quality of currently available CityGML datasets, i.e. we validate the geometry/topology of the 3D primitives (*Solid* and *MultiSurface*), and we validate whether the semantics of the boundary surfaces of buildings is correct or not. We have analysed all the CityGML datasets we could find, both from portals of cities and on different websites, plus a few that were made available to us. We have thus validated 40M surfaces in 16M 3D primitives and 3.6M buildings found in 37 CityGML datasets originating from 9 countries, and produced by several companies with diverse software and acquisition techniques. The results indicate that CityGML datasets without errors are rare, and those that are nearly valid are mostly simple LOD1 models. We report on the most common errors we have found, and analyse them. One main observation is that many of these errors could be automatically fixed or prevented with simple modifications to the modelling software. Our principal aim is to highlight the most common errors so that these are not repeated in the future. We hope that our paper and the open-source software we have developed will help raise awareness for data quality among data providers and 3D GIS software producers.

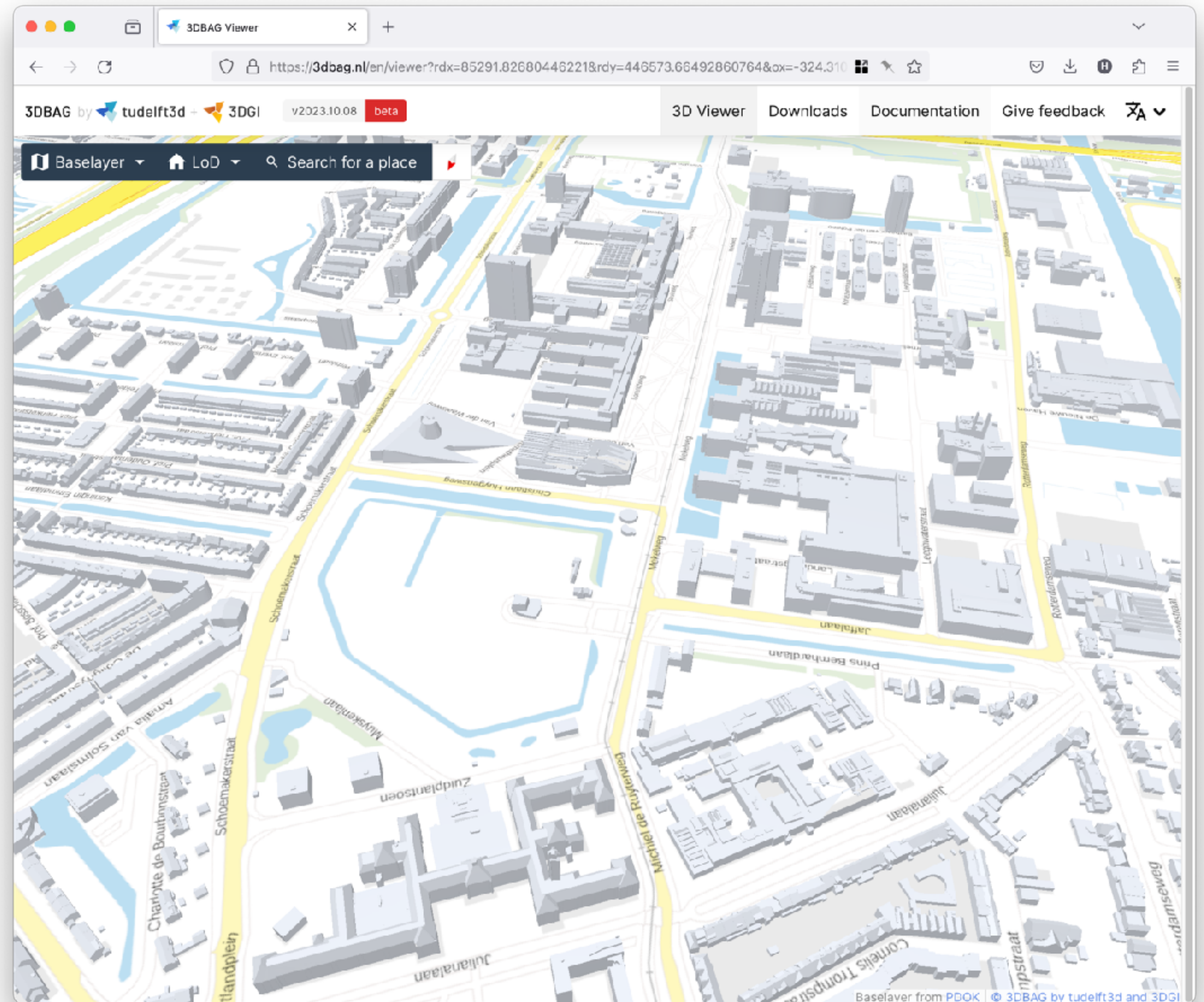
1. INTRODUCTION

In recent years, several cities around the world have released their 3D city models (3DGM).

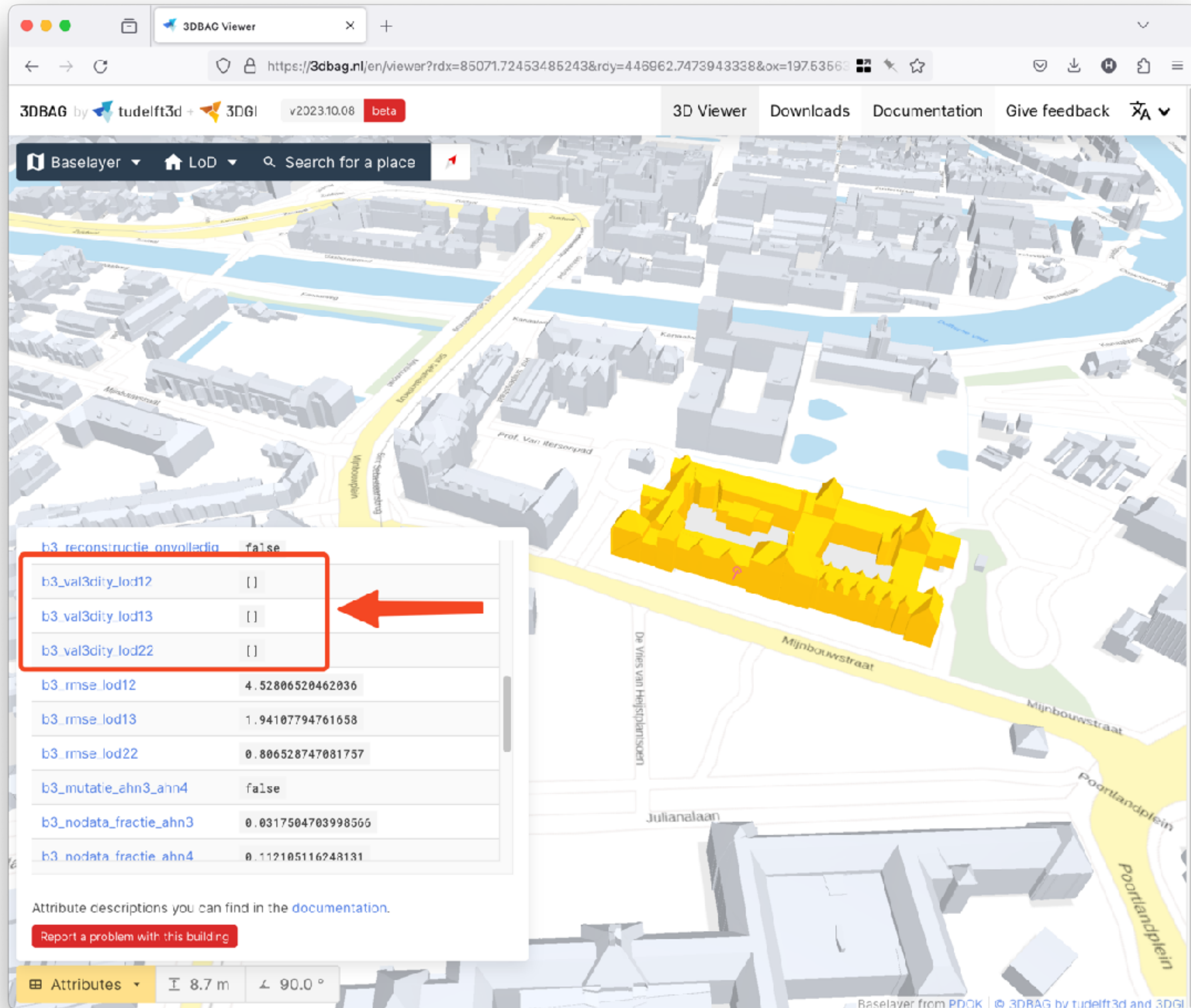
Related work considers additional aspects, e.g. resolvability of *XLinks*; application-specific rules, such as that a building is required to have a ground floor to form a volume (which is, surprisingly, not mandatory in CityGML); and the redundancy of the

3DBAG is ~99% geometrically valid 🦾🚀

Lesson 7.2
on 2026-06-03

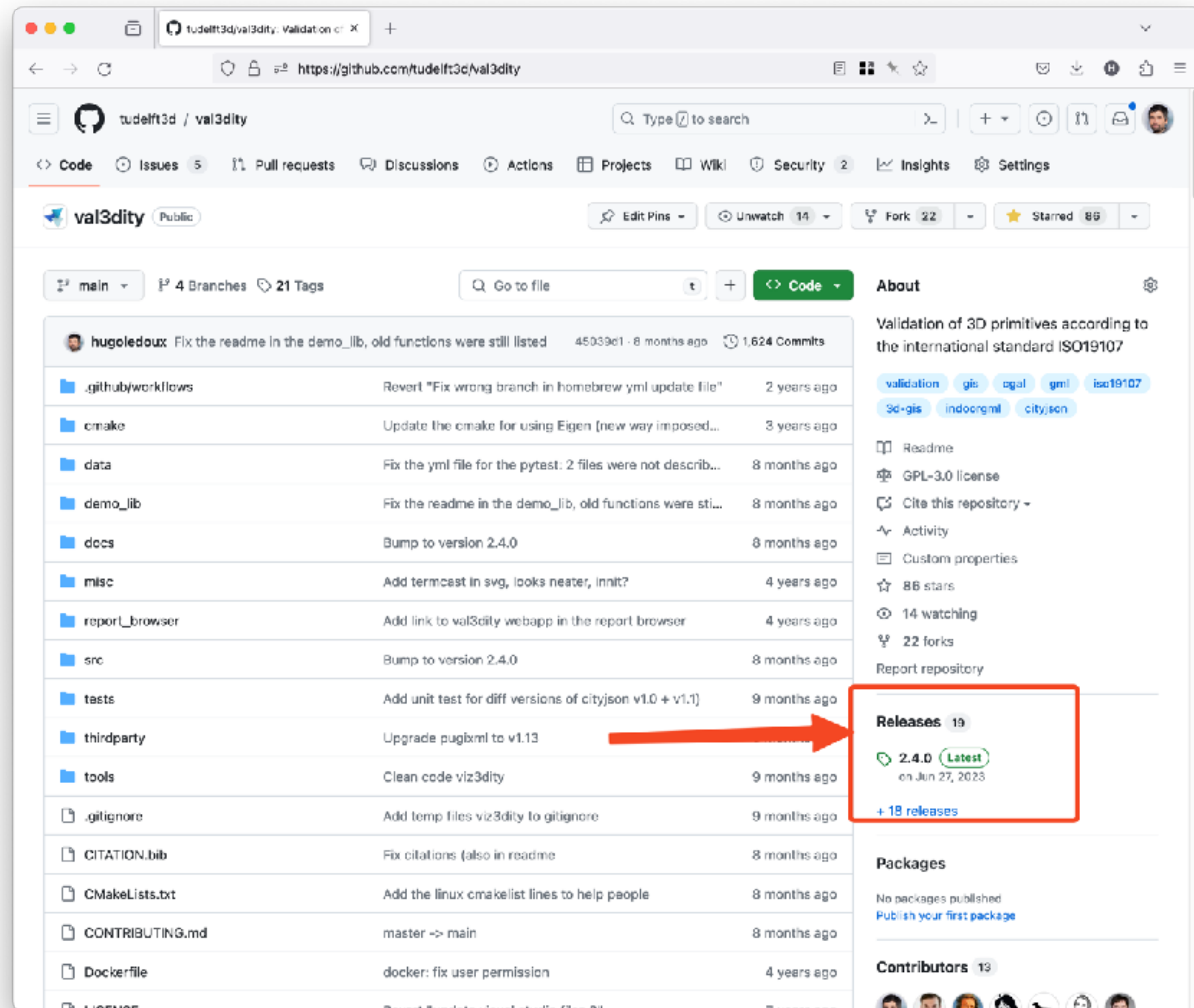


3DBAG: you can see the validity

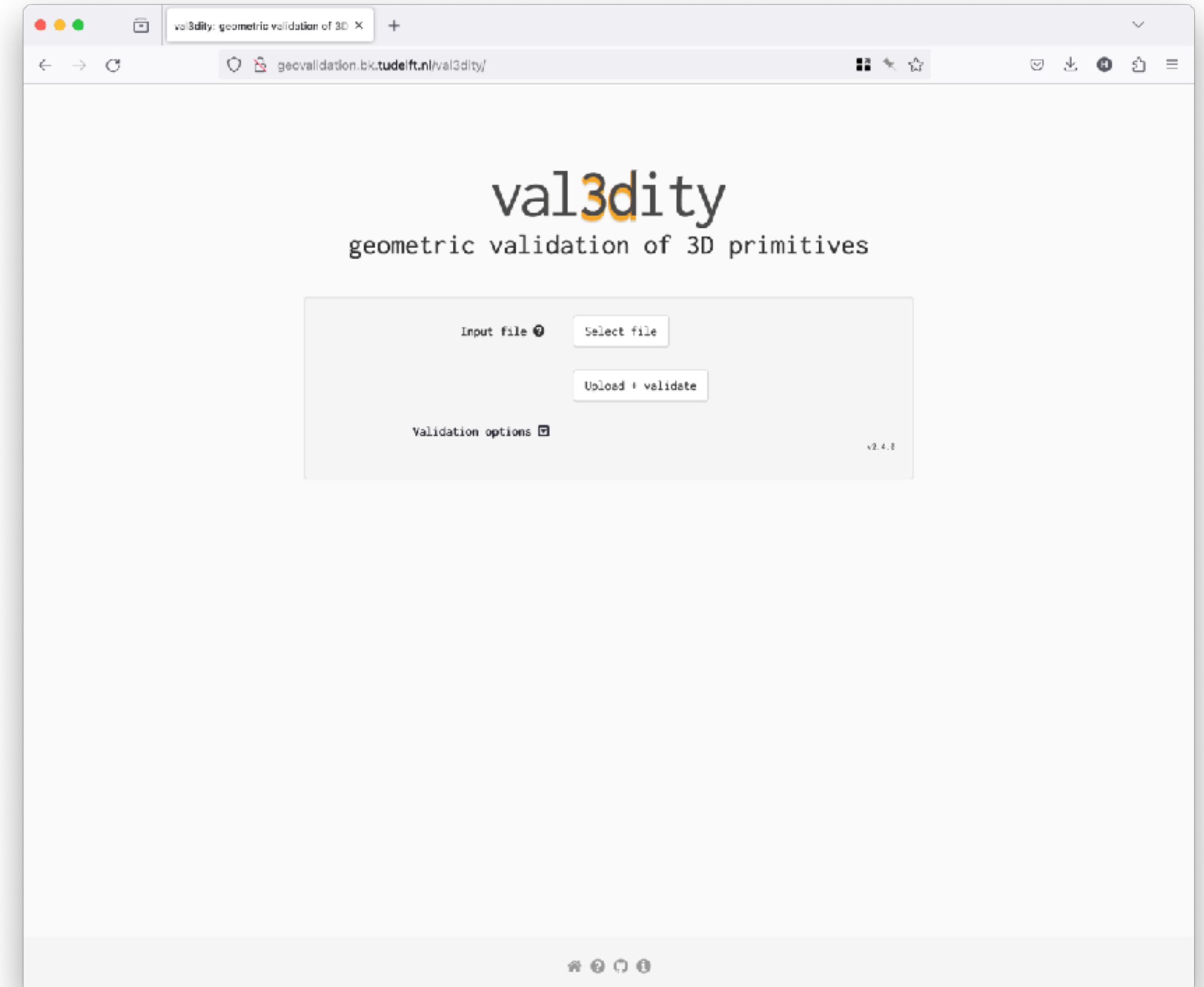


val3dity is integrated in the process

How to use val3dity?

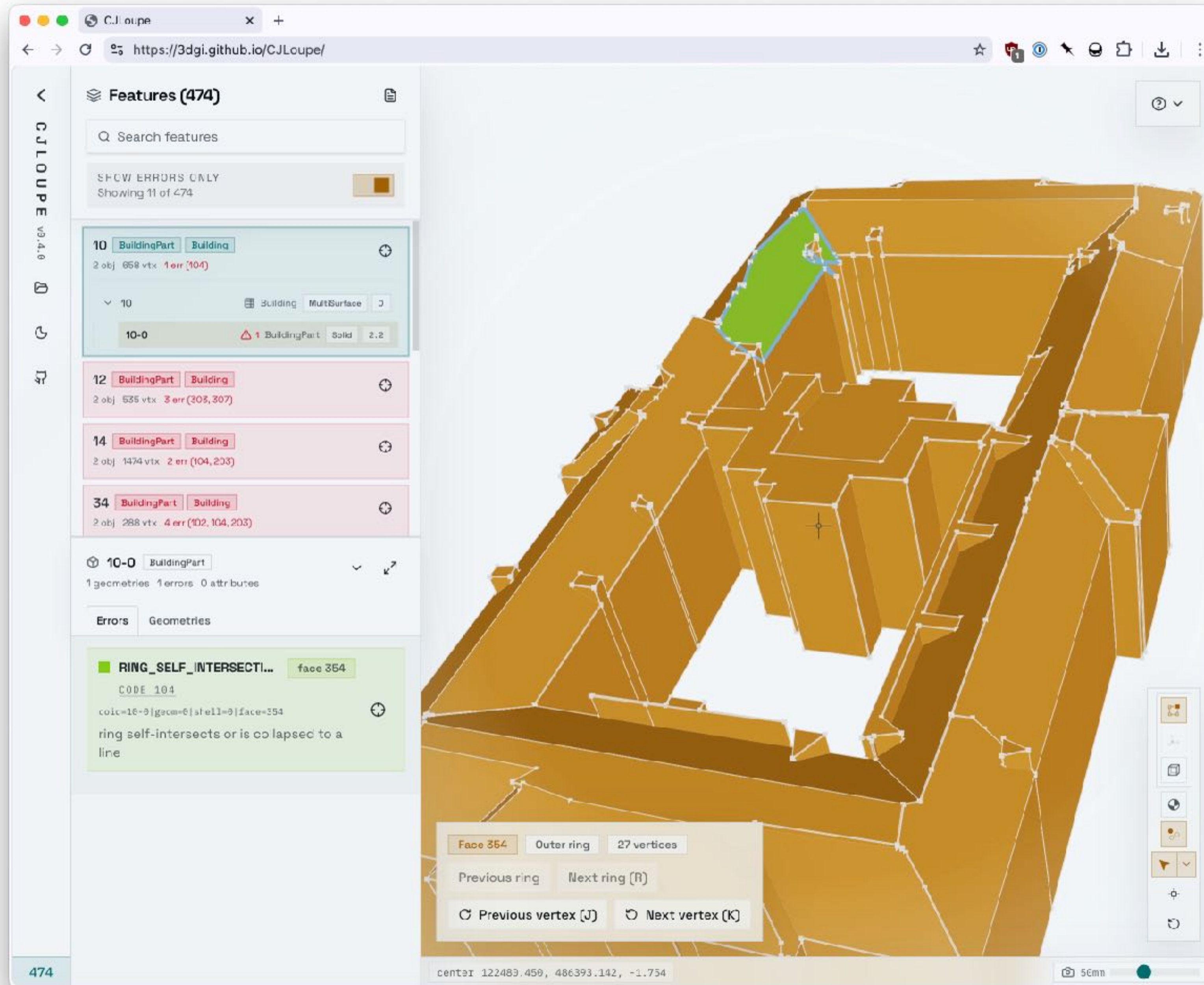


CLI
(compile or download Window binary)



web-application

How to use val3dity?



1. Run val3dity with option `-report`
2. <https://3dgi.github.io/CJLoupe/>
3. Open both *.city.json and report.json
4. explore

Using val3dity as a library in C++

The screenshot shows the GitHub repository page for `val3dity/demo_lib` at the `main` branch. The repository is owned by `tudelft3d` and is public. The page displays the repository's navigation bar, including links to `Code`, `Issues` (7), `Pull requests`, `Discussions`, `Actions`, `Projects`, `Wiki`, `Security`, `Insights`, and `Settings`. The repository has 12 unwatchers, 22 forks, and 68 stars. The file list shows three files: `CMakeLists.txt` (misc, 2 years ago), `main.cpp` (Update demo of API, last year), and `readme.md` (Add warning about relative path in cmake for compiling as library, 8 months ago). The `readme.md` file is selected, showing its content. The title of the README is `how to use val3dity as a library in your code`. The text in the README explains that the API of the library is `val3dity.h`, which needs to be included in the source. It also mentions that the functions to validate read/return `nlohmann JSON objects`, so the `nlohmann-json` library needs to be included. Finally, it states that the API reads `const char*` for XML data (`IndoorGML` input), because it doesn't want to enforce a specific XML library. The code snippet at the bottom shows the required includes:

```
#include "val3dity.h"
#include "nlohmann-json/json.hpp"
#include <fstream>
```