Lesson 4.1
Semantic 3D city models

GEO1004:
3D modelling of the built environment

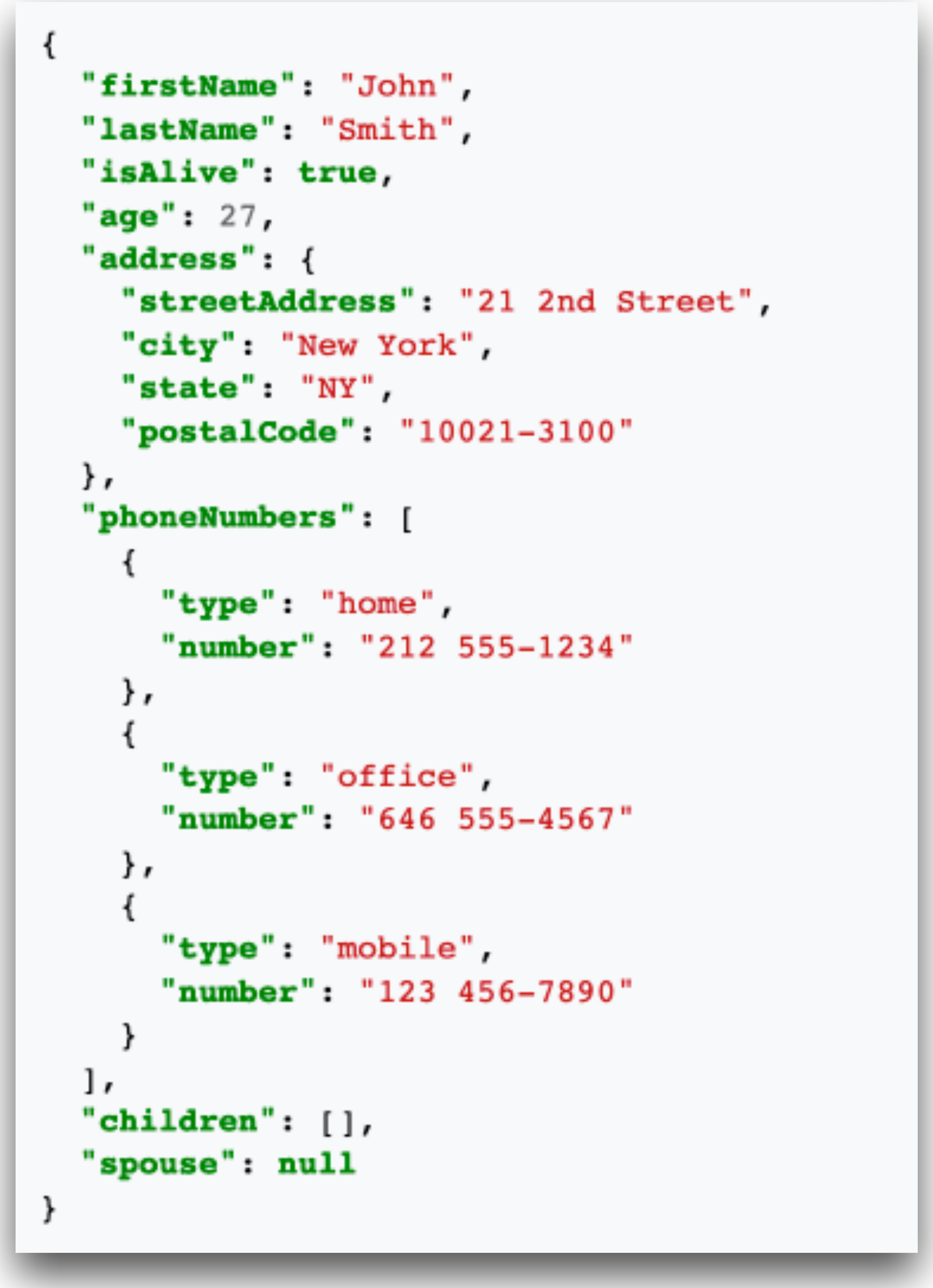
https://3d.bk.tudelft.nl/courses/geo1004

3D geoinformation
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

JSON

JSON

- **J**ava**S**cript **O**bject **N**otation
- Lightweight data-interchange format
- It is easy for humans to read and write.
- It is easy for machines to parse and generate.
- It is originally based on a subset of JavaScript (but is now in most languages)
- It is text-based



```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

Virtually all languages have native support for it

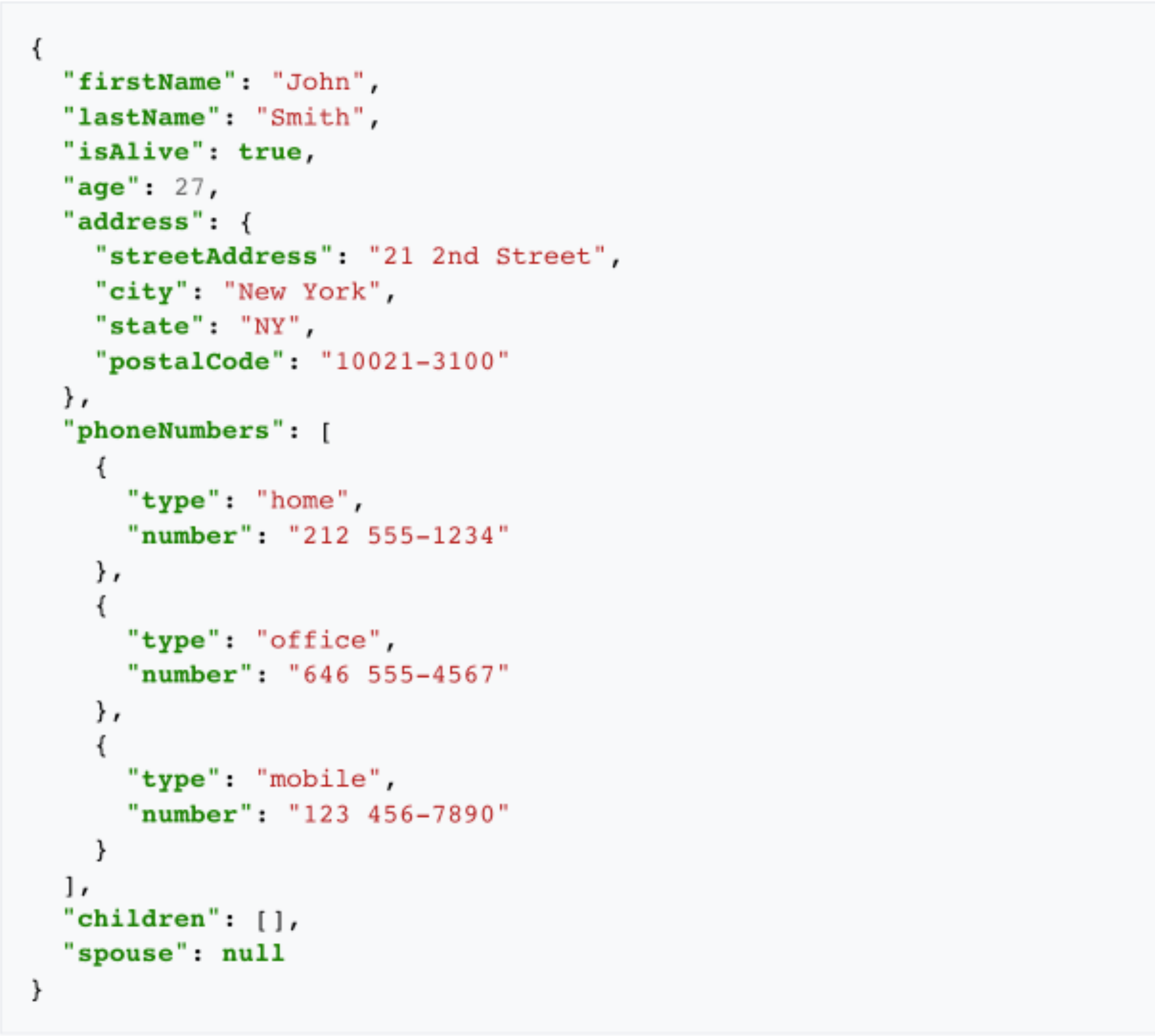
It is built on two (very common) **data structures**:

1. collection of name-value pairs
   - Python dictionaries
   - C++ `std::map`
   - `also called a “hash”`
2. arrays
   - Python lists
   - C++ `std::vector` / `std::array`

Data types:

- number (float and double)
- integer
- string
- Boolean
- null
- *(no support for datetime type)*

To represent a person



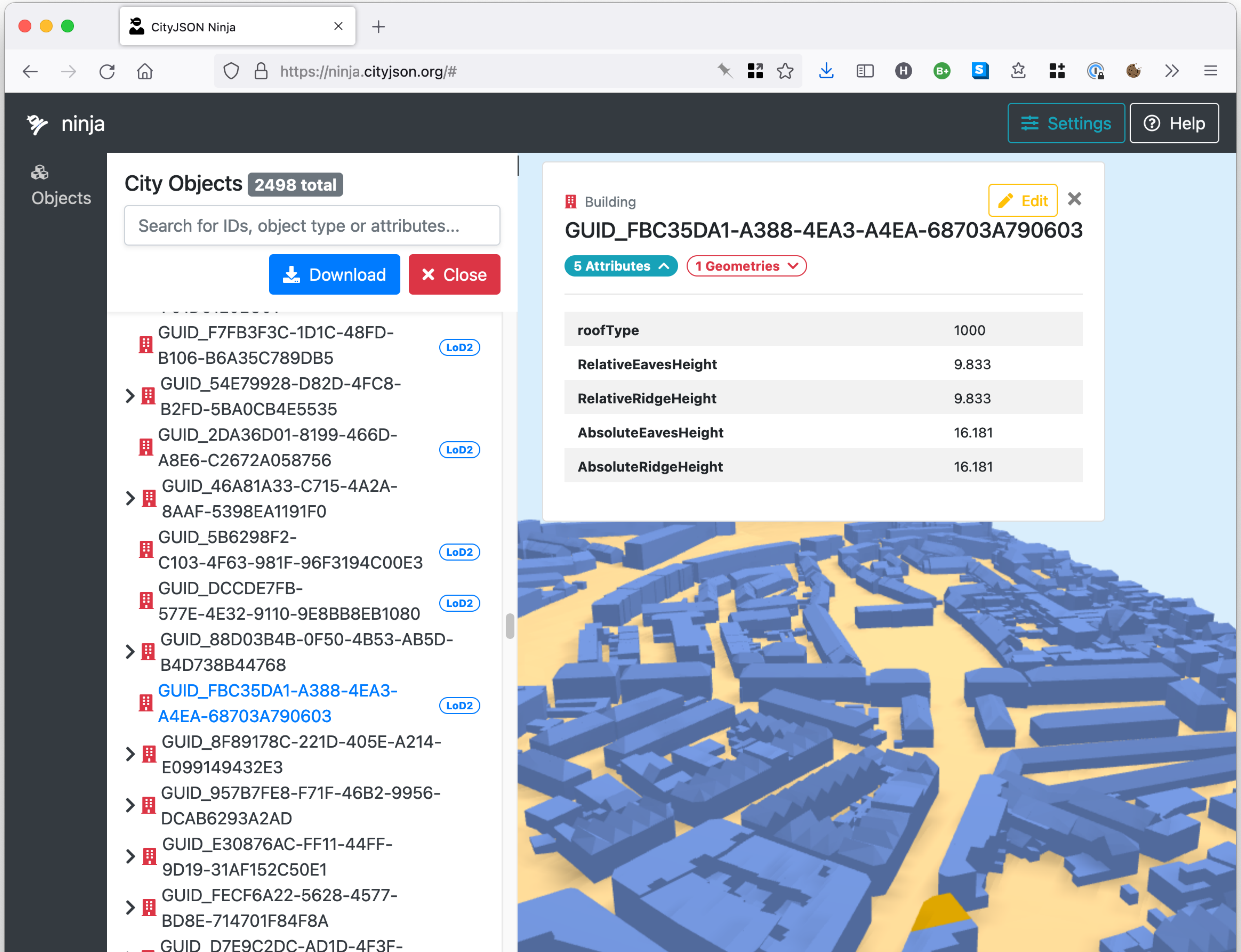
```json
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

A CityJSON file

```
{
  ”type”: “CityJSON”,
  ”version”: “1.1”,
  ”metadata”: {
    ”referenceSystem”: ”https://www.opengis.net/def/crs/EPSG/0/7415”
  },
  ”transform”: {…}
  ”CityObjects”: {
    ”id-1”: {
      ”type”: ”Building”,
      ”attributes”: {
        ”measuredHeight”: 22.3,
        ”roofType”: ”gable”,
        ”owner”: “Elvis Presley”
      },
      ”geometry”: [
        {
          ”type”: ”MultiSurface”,
          ”boundaries”: [
            [[0, 3, 2, 1]], [[4, 5, 6, 7]], [[0, 1, 5, 4]]
          ]
        }
      ]
    }
  },
  ”vertices”: [
    [231, 23212, 110],
    [1111, 3211, 120],
    ...
  ],
  ”appearance”: {
    ”materials”: [],
    ”textures”:[],
    ”vertices-texture”: []
  }
}
```

CityJSON software

web-viewer: ninja.cityjson.org



City Objects 2498 total

Search for IDs, object type or attributes...

GUID_F7FB3F3C-1D1C-48FD-B106-B6A35C789DB5
GUID_54E79928-D82D-4FC8-B2FD-5BA0CB4E5535
GUID_2DA36D01-8199-466D-A8E6-C2672A058756
GUID_46A81A33-C715-4A2A-8AAF-5398EA1191F0
GUID_5B6298F2-C103-4F63-981F-96F3194C00E3
GUID_DCCDE7FB-577E-4E32-9110-9E8BB8EB1080
GUID_88D03B4B-0F50-4B53-AB5D-B4D738B44768
GUID_FBC35DA1-A388-4EA3-A4EA-68703A790603
GUID_8F89178C-221D-405E-A214-E099149432E3
GUID_957B7FE8-F71F-46B2-9956-DCAB6293A2AD
GUID_E30876AC-FF11-44FF-9D19-31AF152C50E1
GUID_FECF6A22-5628-4577-BD8E-714701F84F8A
GUID_D7E9C2DC-AD1D-4F3F-

Building

GUID_FBC35DA1-A388-4EA3-A4EA-68703A790603

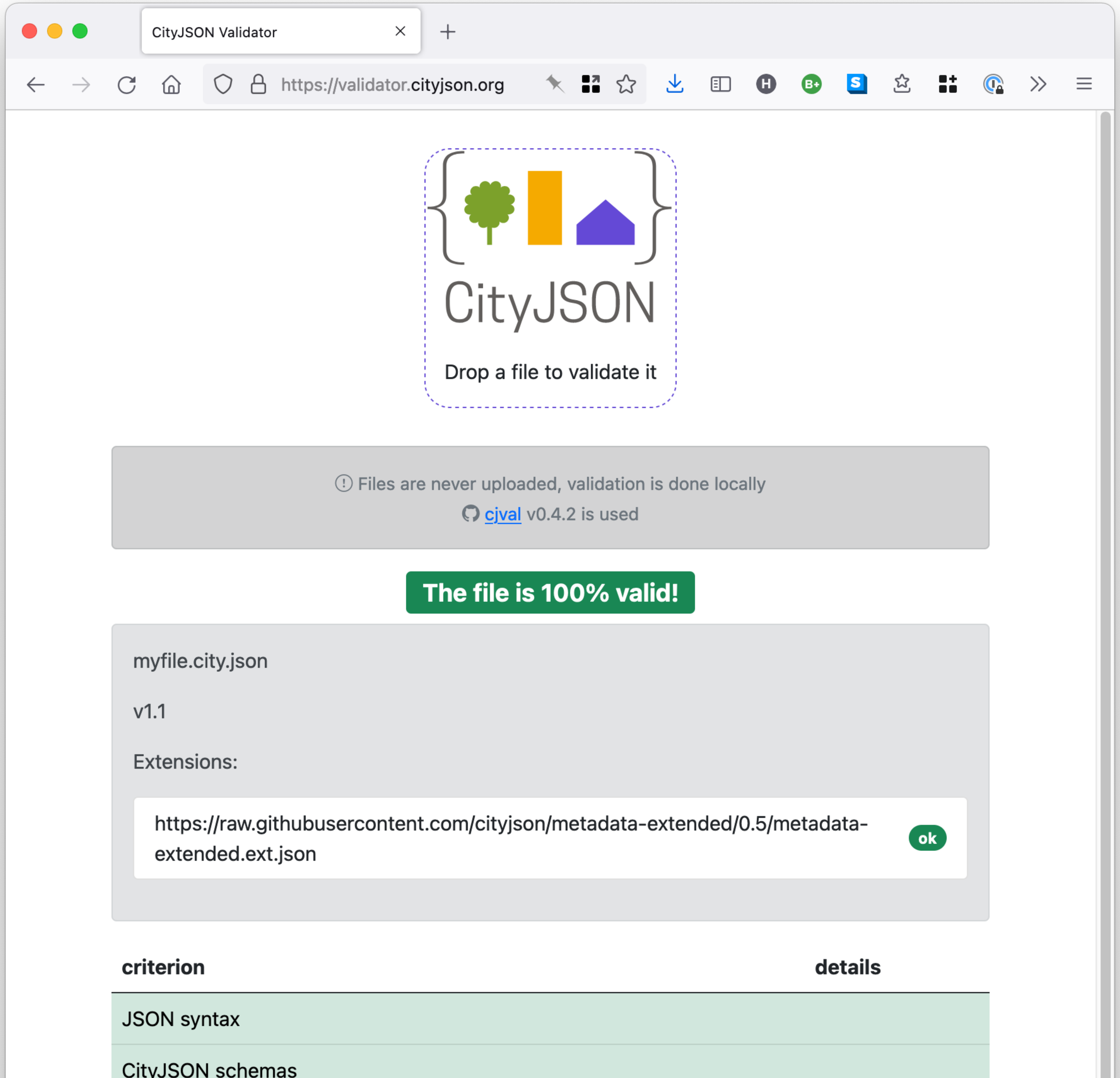
5 Attributes 1 Geometries

roofType	1000
RelativeEavesHeight	9.833
RelativeRidgeHeight	9.833
AbsoluteEavesHeight	16.181
AbsoluteRidgeHeight	16.181

Cool fact: programmed mostly by 2 Geomatics students, as projects for GEO5010

You can do the same (or similar!)

Validation of the syntax of a file: cjval



Drop a file to validate it

ⓘ Files are never uploaded, validation is done locally
cjval v0.4.2 is used

The file is 100% valid!

myfile.city.json

v1.1

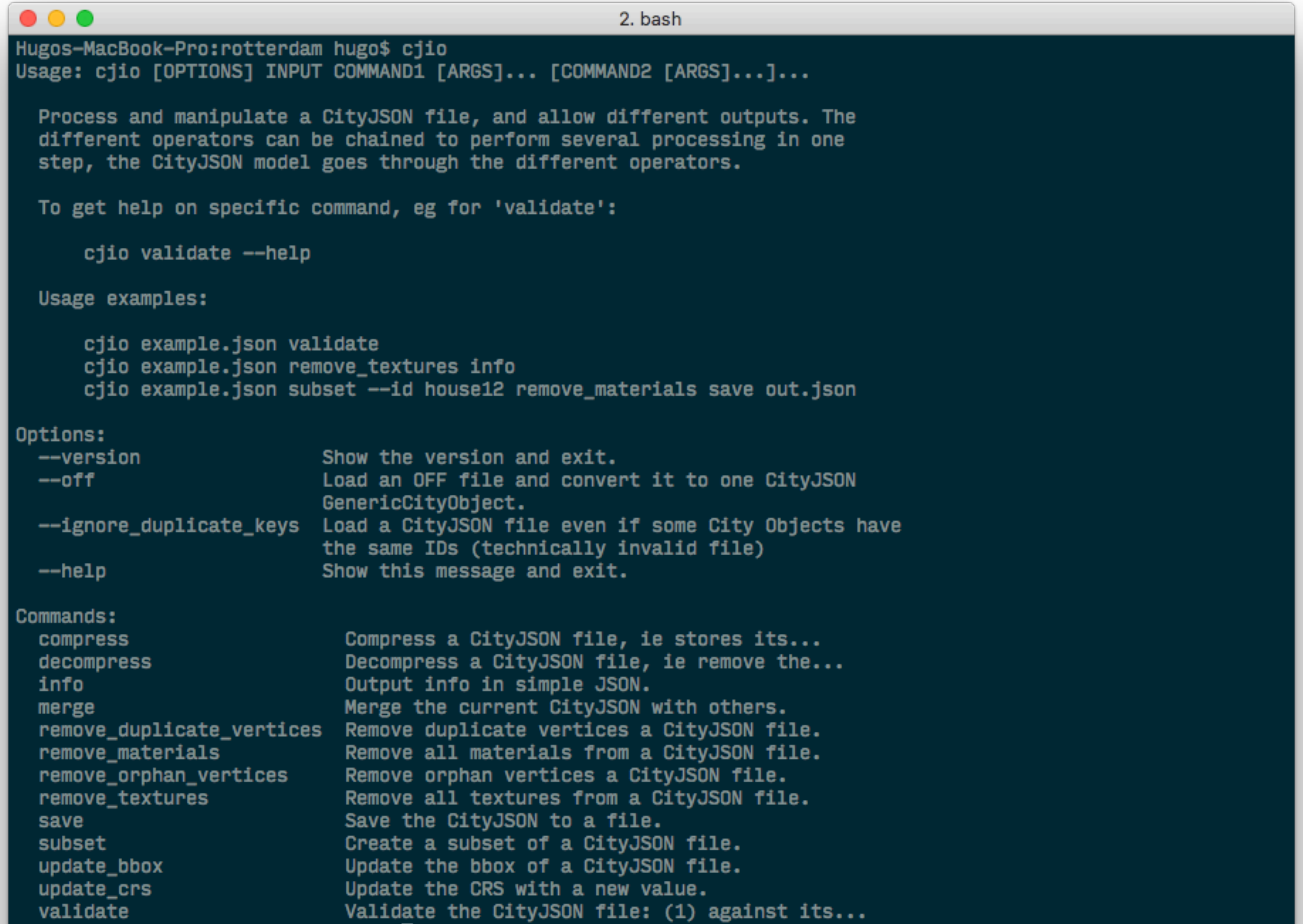
Extensions:

https://raw.githubusercontent.com/cityjson/metadata-extended/0.5/metadata-extended.ext.json ok

criterion	details
JSON syntax	
CityJSON schemas	

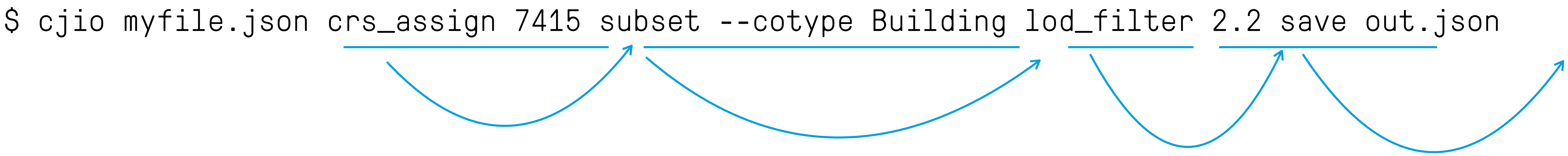
```
{
  ”type”: “CityJSON”,
  ”version”: “1.1”,
  ”metadata”: {
    ”referenceSystem”: ”https://www.opengis.net/def/crs/EPSG/0/7415”
  },
  ”transform”: {…}
  ”CityObjects”: {
    ”id-1”: {
      ”type”: ”Buildnig”,
      ”attributes”: {
        ”measuredHeight”: 22.3,
        ”roofType”: “gable”,
        ”owner”: “Elvis Presley”
      },
      “geometry”: [
        {
          ”type”: “MultiSurface”,
          ”boundaries”: [
            [[0, 3, 2, 1]], [[4, 5, 6, 7], [[0, 1, 5, 4]]
          ]
        }
      ]
    }
  },
  ”vertices”: [
    [231, 23212, 110.223],
    [1111, 3211, 120],
    ...
  ],
  ”appearance”: {
    ”materials”: [],
    ”textures”:[],
    ”vertices-texture”: []
  }
}
```

cjio (CityJSON/io) => pip install cjio



```
Hugos-MacBook-Pro:rotterdam hugo$ cjio
Usage: cjio [OPTIONS] INPUT COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  Process and manipulate a CityJSON file, and allow different outputs. The
  different operators can be chained to perform several processing in one
  step, the CityJSON model goes through the different operators.

  To get help on specific command, eg for 'validate':

      cjio validate --help

  Usage examples:

      cjio example.json validate
      cjio example.json remove_textures info
      cjio example.json subset --id house12 remove_materials save out.json

Options:
  --version                Show the version and exit.
  --off                    Load an OFF file and convert it to one CityJSON
                           GenericCityObject.
  --ignore_duplicate_keys  Load a CityJSON file even if some City Objects have
                           the same IDs (technically invalid file)
  --help                   Show this message and exit.

Commands:
  compress                   Compress a CityJSON file, ie stores its...
  decompress                 Decompress a CityJSON file, ie remove the...
  info                       Output info in simple JSON.
  merge                      Merge the current CityJSON with others.
  remove_duplicate_vertices  Remove duplicate vertices a CityJSON file.
  remove_materials           Remove all materials from a CityJSON file.
  remove_orphan_vertices     Remove orphan vertices a CityJSON file.
  remove_textures            Remove all textures from a CityJSON file.
  save                       Save the CityJSON to a file.
  subset                     Create a subset of a CityJSON file.
  update_bbox                Update the bbox of a CityJSON file.
  update_crs                 Update the CRS with a new value.
  validate                   Validate the CityJSON file: (1) against its...
```

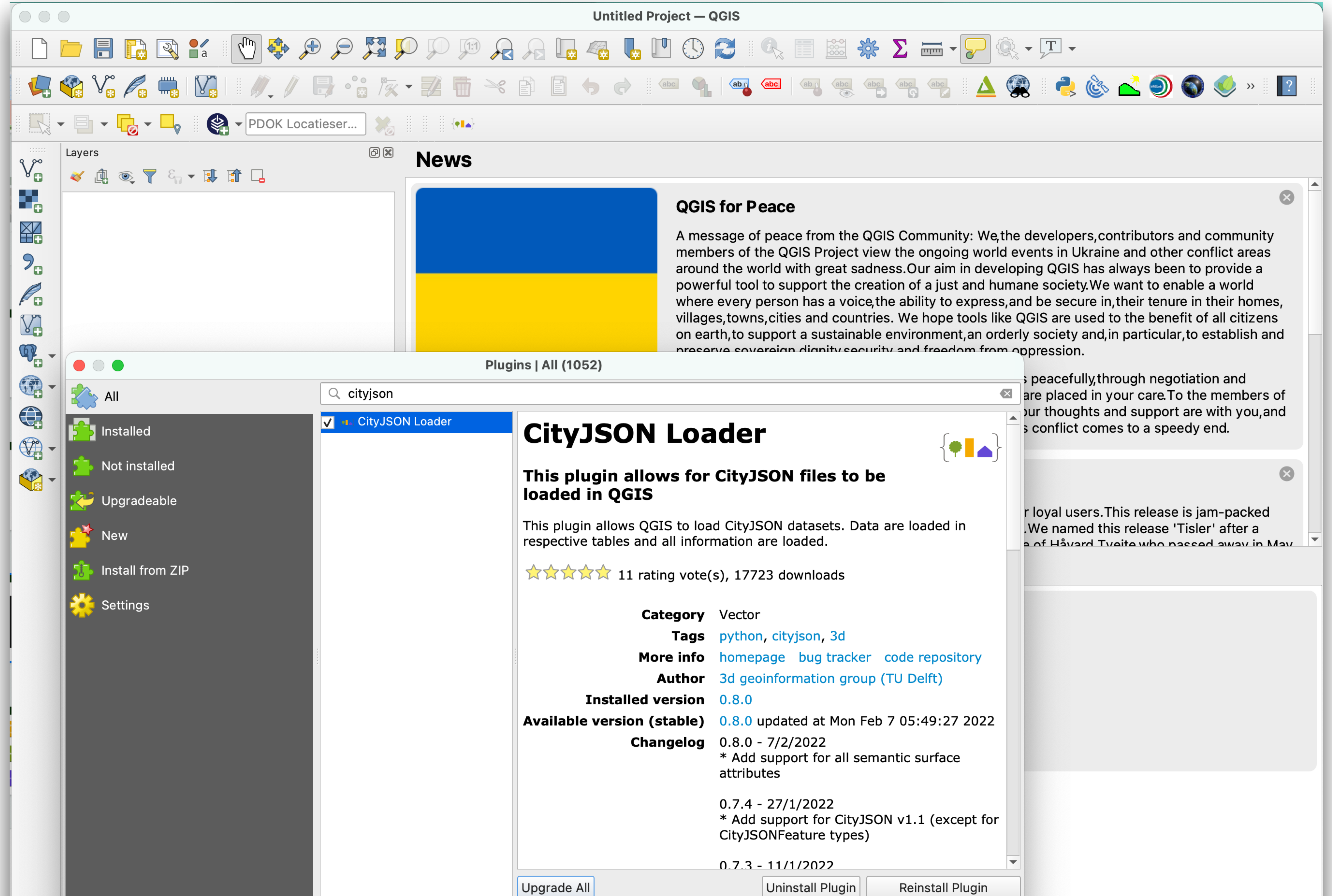
cjio (CityJSON/io)



```
$ cjio myfile.json crs_assign 7415 subset --cotype Building lod_filter 2.2 save out.json
```

No need to save temp files between operators: pipeline is used

QGIS plugin



QGIS for Peace

A message of peace from the QGIS Community: We,the developers,contributors and community members of the QGIS Project view the ongoing world events in Ukraine and other conflict areas around the world with great sadness.Our aim in developing QGIS has always been to provide a powerful tool to support the creation of a just and humane society.We want to enable a world where every person has a voice,the ability to express,and be secure in,their tenure in their homes, villages,towns,cities and countries. We hope tools like QGIS are used to the benefit of all citizens on earth,to support a sustainable environment,an orderly society and,in particular,to establish and preserve sovereign dignity,security and freedom from oppression.

s peacefully,through negotiation and
are placed in your care.To the members of
our thoughts and support are with you,and
s conflict comes to a speedy end.

r loyal users.This release is jam-packed
.We named this release 'Tisler' after a
e of Håvard Tveite who passed away in May

CityJSON Loader

This plugin allows for CityJSON files to be loaded in QGIS

This plugin allows QGIS to load CityJSON datasets. Data are loaded in respective tables and all information are loaded.

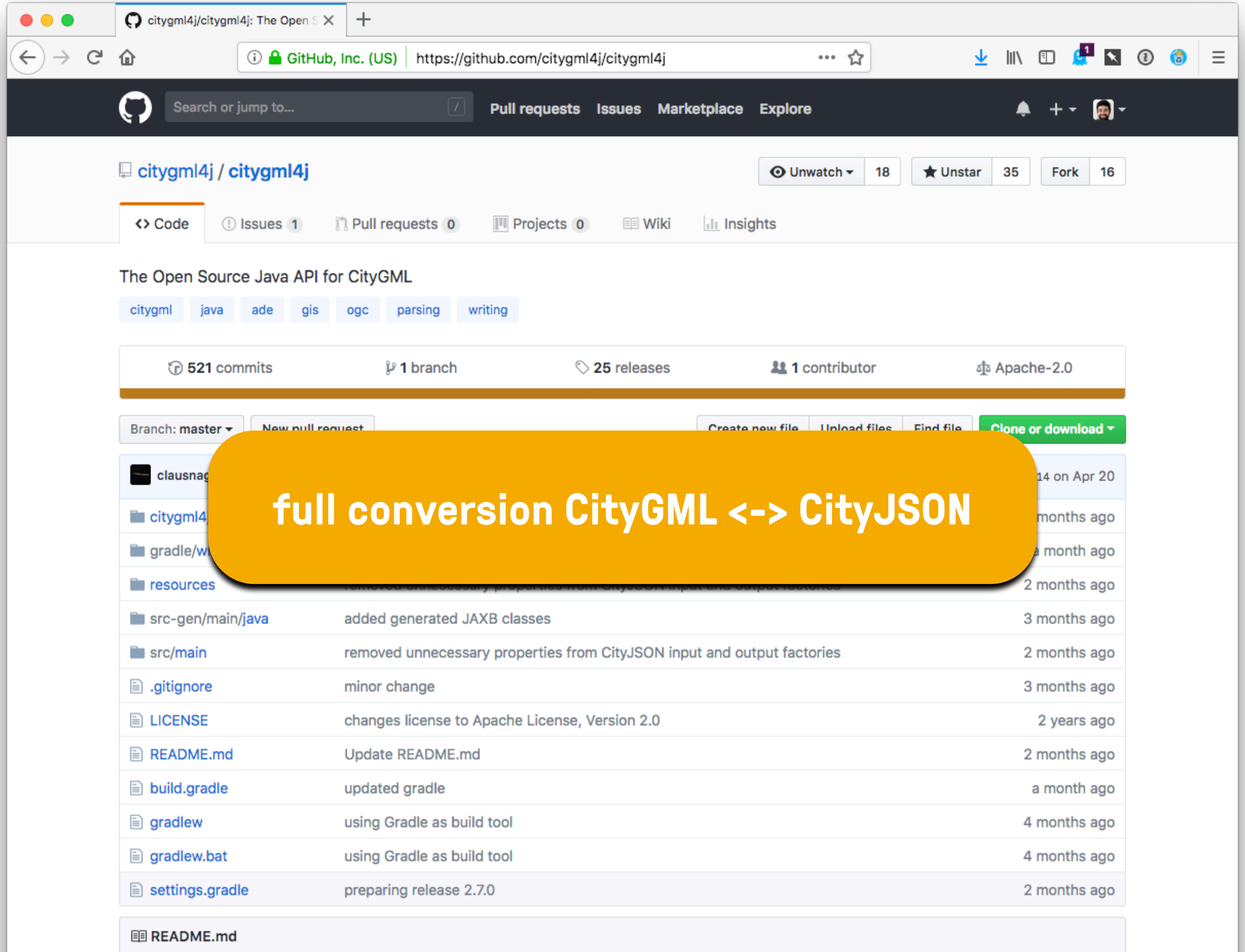
★★★★★ 11 rating vote(s), 17723 downloads

Category Vector
Tags python, cityjson, 3d
More info homepage bug tracker code repository
Author 3d geoinformation group (TU Delft)
Installed version 0.8.0
Available version (stable) 0.8.0 updated at Mon Feb 7 05:49:27 2022
Changelog 0.8.0 - 7/2/2022
* Add support for all semantic surface attributes

0.7.4 - 27/1/2022
* Add support for CityJSON v1.1 (except for CityJSONFeature types)

0.7.3 - 11/1/2022

citygml4j



The Open Source Java API for CityGML

citygml java ade gis ogc parsing writing

521 commits | 1 branch | 25 releases | 1 contributor | Apache-2.0

clausnag ... 14 on Apr 20

citygml4		months ago
gradle/w		a month ago
resources		2 months ago
src-gen/main/java	added generated JAXB classes	3 months ago
src/main	removed unnecessary properties from CityJSON input and output factories	2 months ago
.gitignore	minor change	3 months ago
LICENSE	changes license to Apache License, Version 2.0	2 years ago
README.md	Update README.md	2 months ago
build.gradle	updated gradle	a month ago
gradlew	using Gradle as build tool	4 months ago
gradlew.bat	using Gradle as build tool	4 months ago
settings.gradle	preparing release 2.7.0	2 months ago

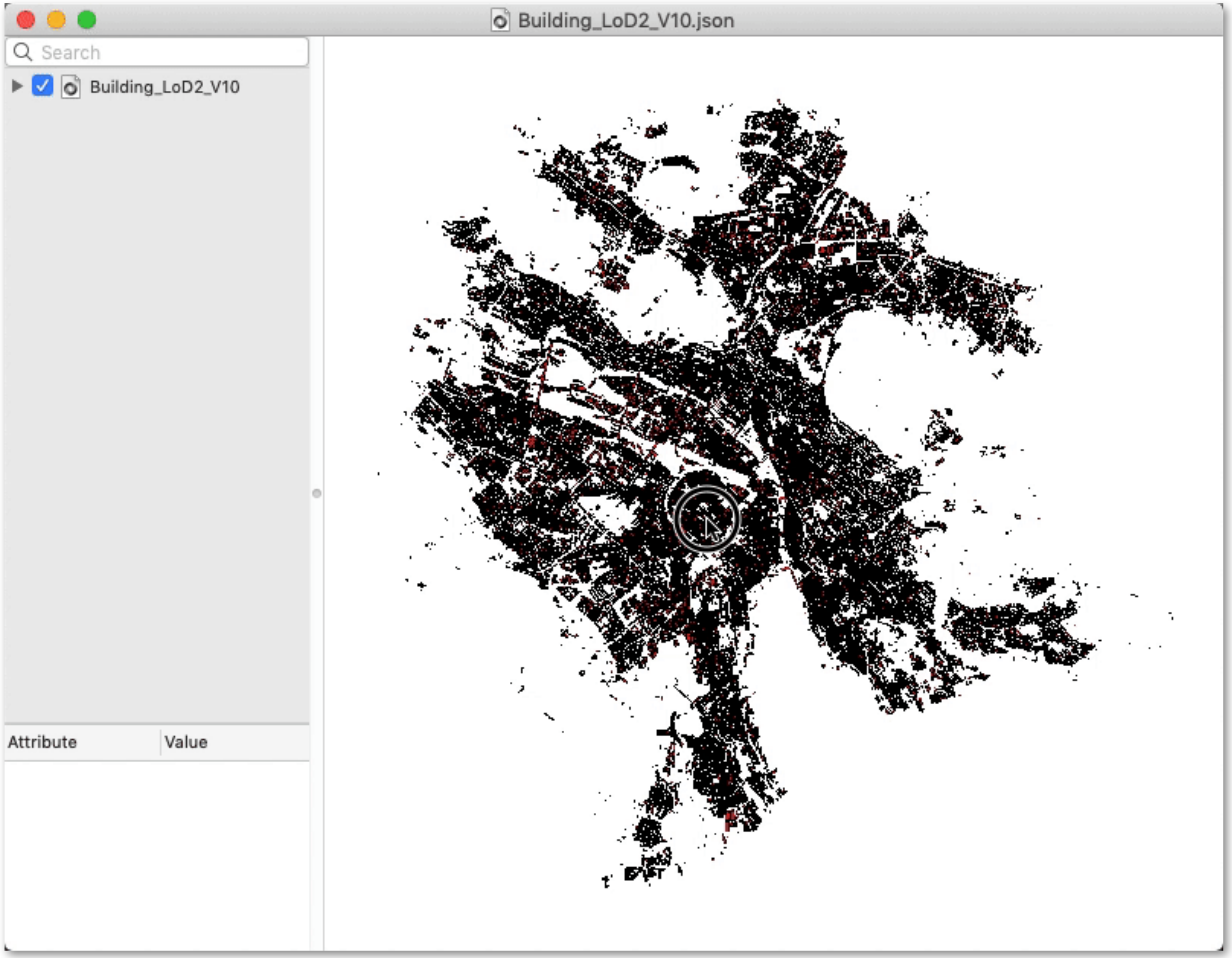
README.md

full conversion CityGML <-> CityJSON

Compression factor == ~6X

file	CityGML size (original)	CityGML size (w/o spaces)	textures?	CityJSON	CityJSON compressed	compression factor
CityGML demo "GeoRes"	4.3MB	4.1MB	yes	582KB	524KB	8.0
CityGML v2 demo "Railway"	45MB	34MB	yes	4.5MB	4.3MB	8.1
Den Haag "tile 01"	23MB	18MB	no, material	3.1MB	2.9MB	6.2
Montréal VM05	56MB	42MB	yes	5.7MB	5.4MB	7.8
New York LoD2 (DA13)	590MB	574MB	no	110MB	105MB	5.5
Rotterdam Delfshaven	16MB	15MB	yes	2.8MB	2.6MB	5.4
Vienna	37MB	36MB	no	5.6MB	5.3MB	6.8

One example: Zürich LoD2 buildings



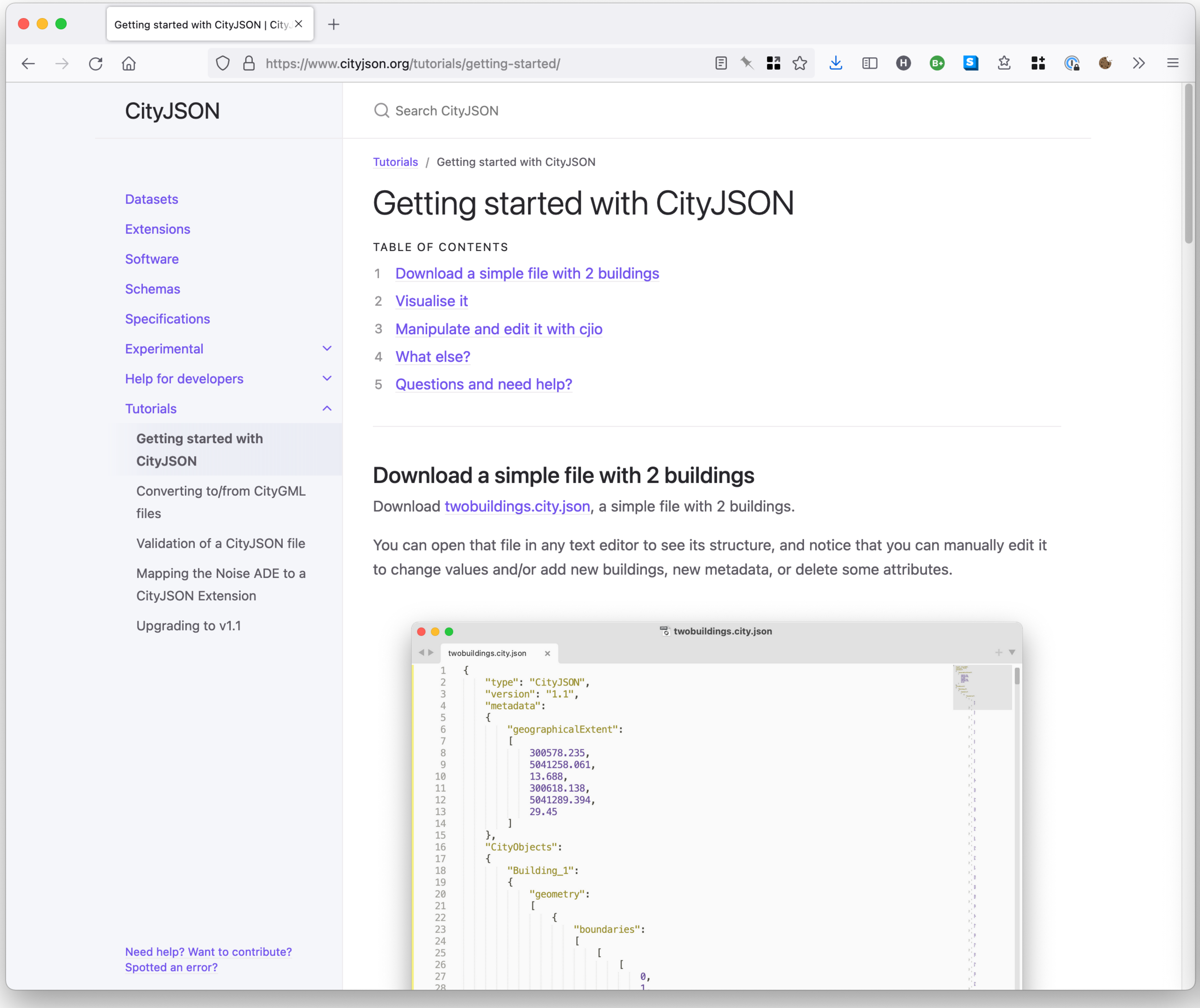
CityGML = 3.0GB

(but 1GB of spaces/CRs/tabs!)

CityJSON = 292MB

Compression == 7.1X

Getting started?



Getting started with CityJSON | City
https://www.cityjson.org/tutorials/getting-started/
CityJSON
Search CityJSON
Datasets
Extensions
Software
Schemas
Specifications
Experimental
Help for developers
Tutorials
Getting started with CityJSON
Converting to/from CityGML files
Validation of a CityJSON file
Mapping the Noise ADE to a CityJSON Extension
Upgrading to v1.1
Need help? Want to contribute? Spotted an error?
Tutorials / Getting started with CityJSON

Getting started with CityJSON

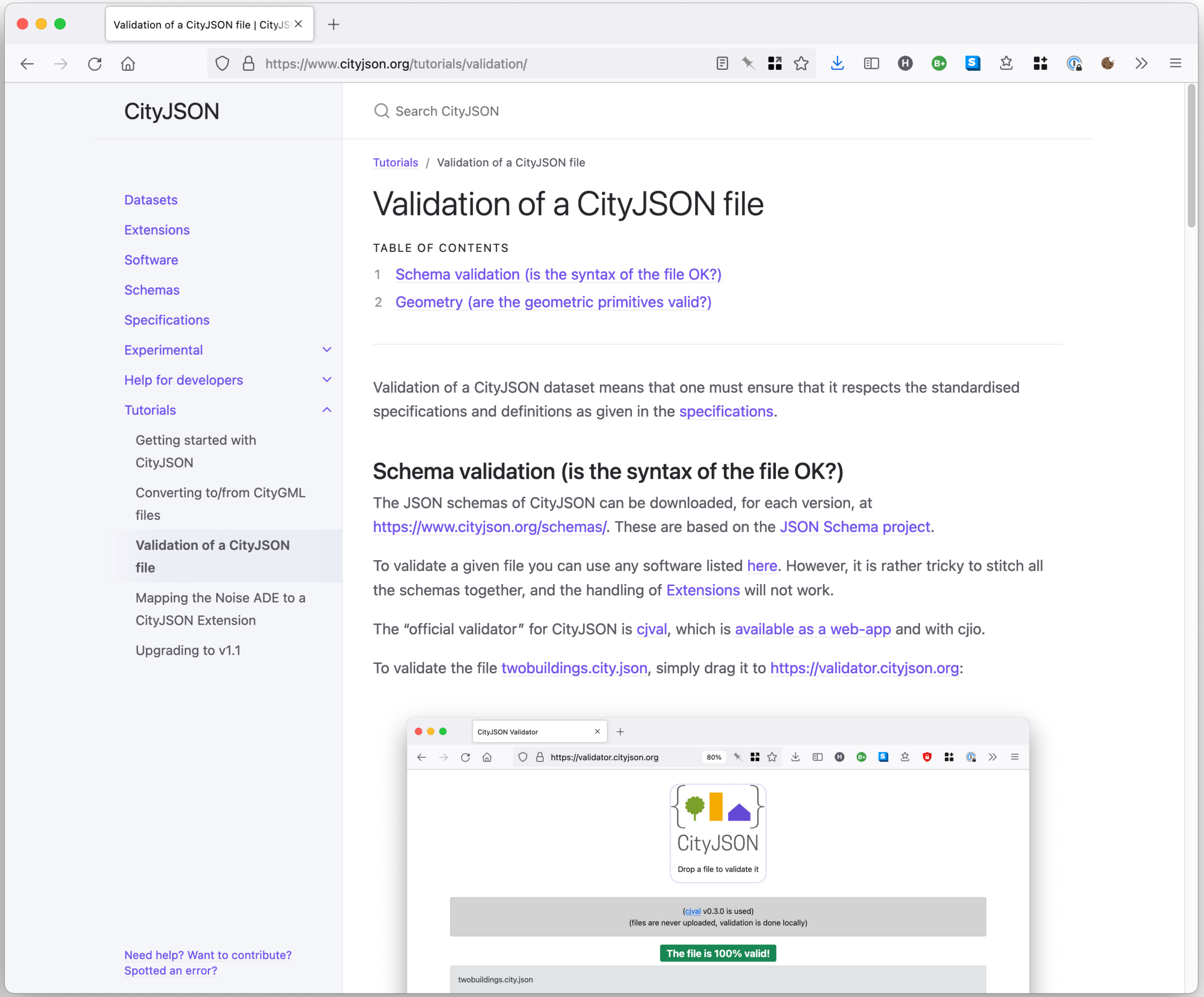
TABLE OF CONTENTS

Download a simple file with 2 buildings

Download twobuildings.city.json, a simple file with 2 buildings.

You can open that file in any text editor to see its structure, and notice that you can manually edit it to change values and/or add new buildings, new metadata, or delete some attributes.

```
{
    "type": "CityJSON",
    "version": "1.1",
    "metadata":
    {
        "geographicalExtent":
        [
            300578.235,
            5041258.061,
            13.688,
            300618.138,
            5041289.394,
            29.45
        ]
    },
    "CityObjects":
    {
        "Building_1":
        {
            "geometry":
            [
                {
                    "boundaries":
                    [
                        [
                            [
                                0,
```



Validation of a CityJSON file | CityJS
https://www.cityjson.org/tutorials/validation/
CityJSON
Search CityJSON
Datasets
Extensions
Software
Schemas
Specifications
Experimental
Help for developers
Tutorials
Getting started with CityJSON
Converting to/from CityGML files
Validation of a CityJSON file
Mapping the Noise ADE to a CityJSON Extension
Upgrading to v1.1
Need help? Want to contribute? Spotted an error?
Tutorials / Validation of a CityJSON file

Validation of a CityJSON file

TABLE OF CONTENTS

Validation of a CityJSON dataset means that one must ensure that it respects the standardised specifications and definitions as given in the specifications.

Schema validation (is the syntax of the file OK?)

The JSON schemas of CityJSON can be downloaded, for each version, at https://www.cityjson.org/schemas/. These are based on the JSON Schema project.

To validate a given file you can use any software listed here. However, it is rather tricky to stitch all the schemas together, and the handling of Extensions will not work.

The “official validator” for CityJSON is cjval, which is available as a web-app and with cjio.

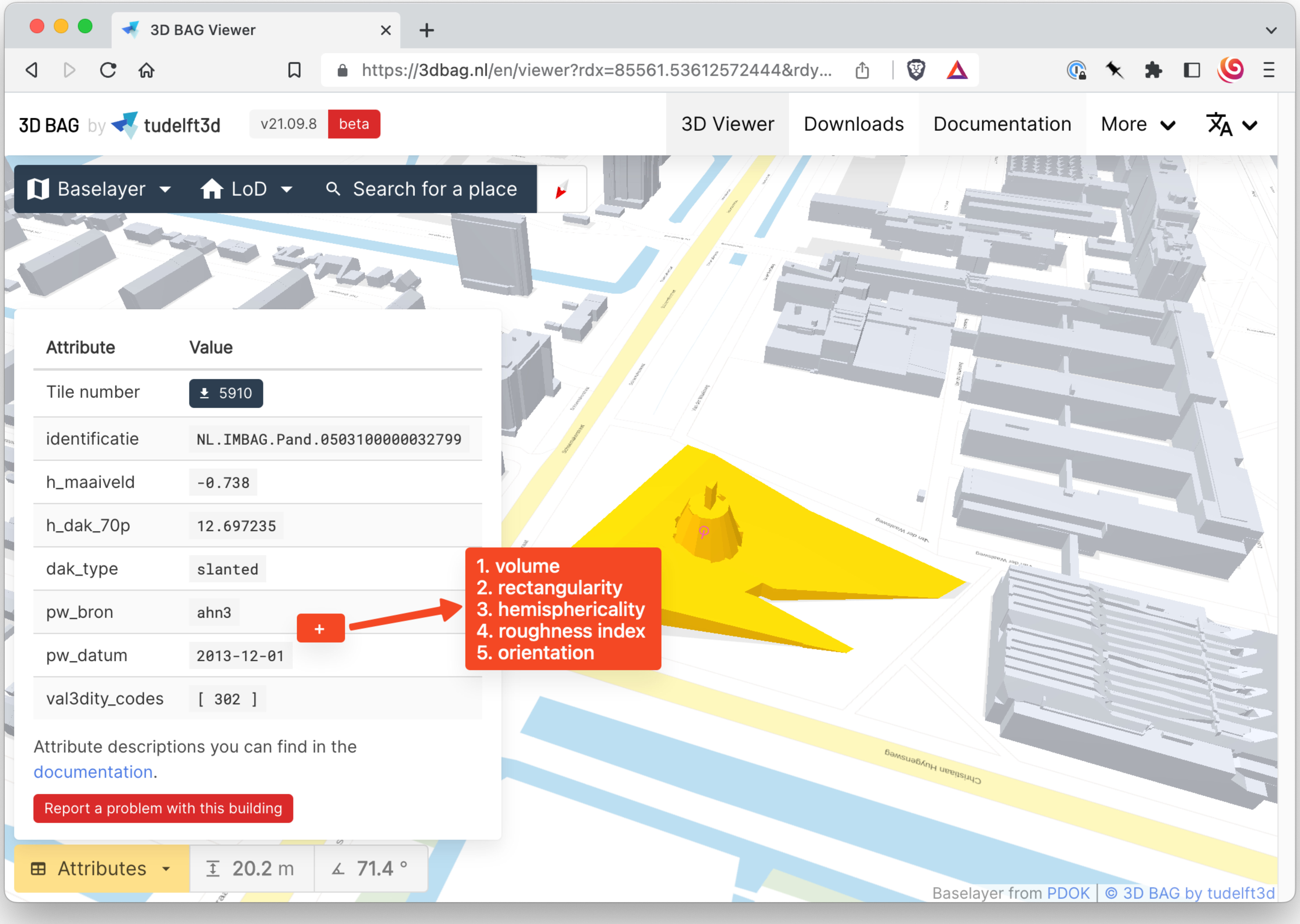
To validate the file twobuildings.city.json, simply drag it to https://validator.cityjson.org:

(cjval v0.3.0 is used)
(files are never uploaded, validation is done locally)

The file is 100% valid!

twobuildings.city.json

hw02


3D BAG Viewer
https://3dbag.nl/en/viewer?rdx=85561.53612572444&rdy...
3D BAG by tudelft3d
v21.09.8
beta
3D Viewer
Downloads
Documentation
More
Baselayer
LoD
Search for a place
+
Report a problem with this building
Attributes
20.2 m
71.4 °
Baselayer from PDOK | © 3D BAG by tudelft3d

Attribute	Value
Tile number	5910
identificatie	NL.IMBAG.Pand.0503100000032799
h_maaiveld	-0.738
h_dak_70p	12.697235
dak_type	slanted
pw_bron	ahn3
pw_datum	2013-12-01
val3dity_codes	[302]

Attribute descriptions you can find in the documentation.

1. volume
2. rectangularity
3. hemisphericality
4. roughness index
5. orientation

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
2023, VOL. 37, NO. 1, 36–67
https://doi.org/10.1080/13658816.2022.2103818

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

OPEN ACCESS Check for updates

3D building metrics for urban morphology

Anna Labetski[a], Stelios Vitalis[a], Filip Biljecki[b,c], Ken Arroyo Ohori[a] and Jantien Stoter[a]

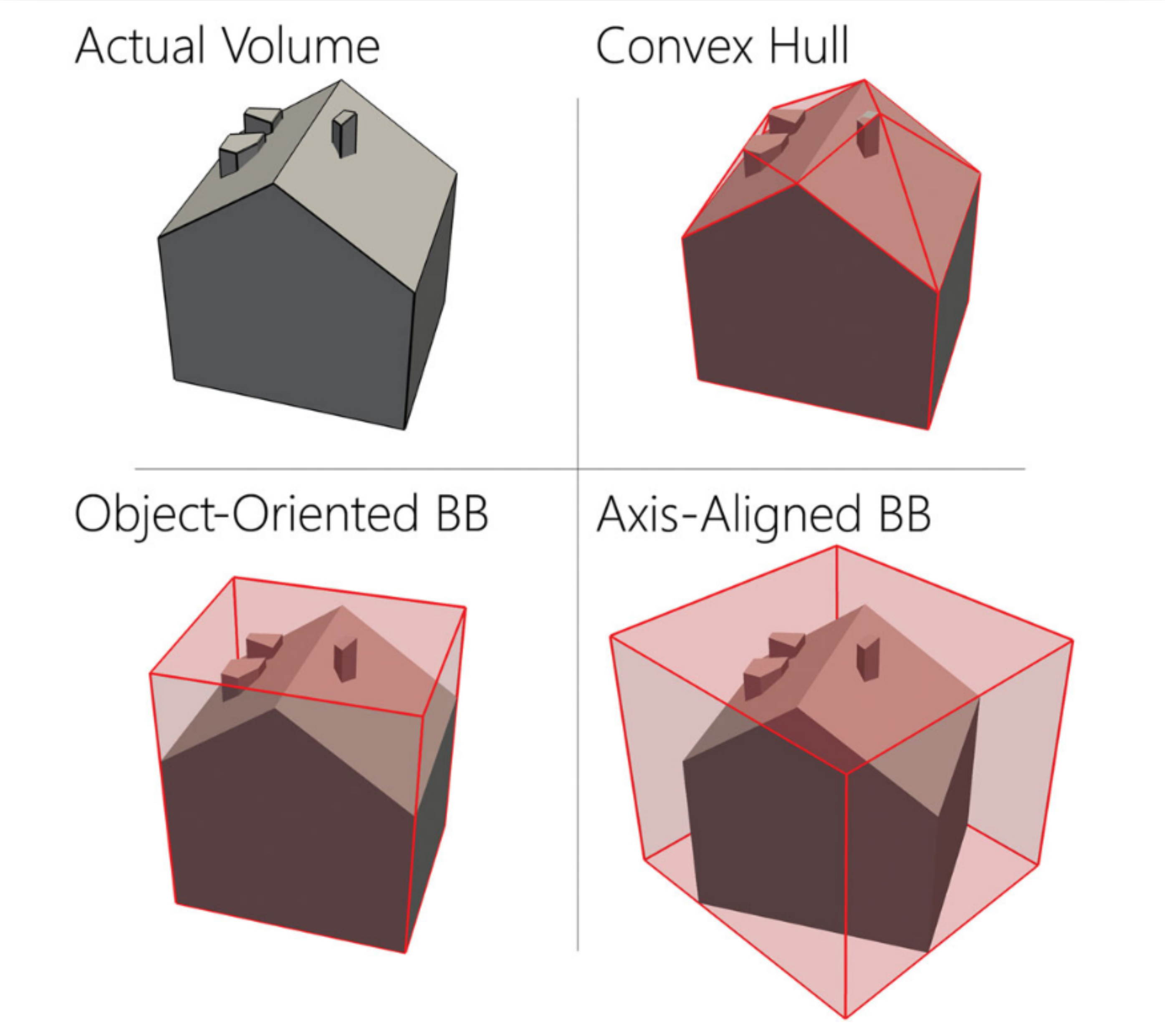
[a]3D Geoinformation Group, Delft University of Technology, Delft, The Netherlands; [b]Department of Architecture, National University of Singapore, Singapore, Singapore; [c]Department of Real Estate, National University of Singapore, Singapore, Singapore

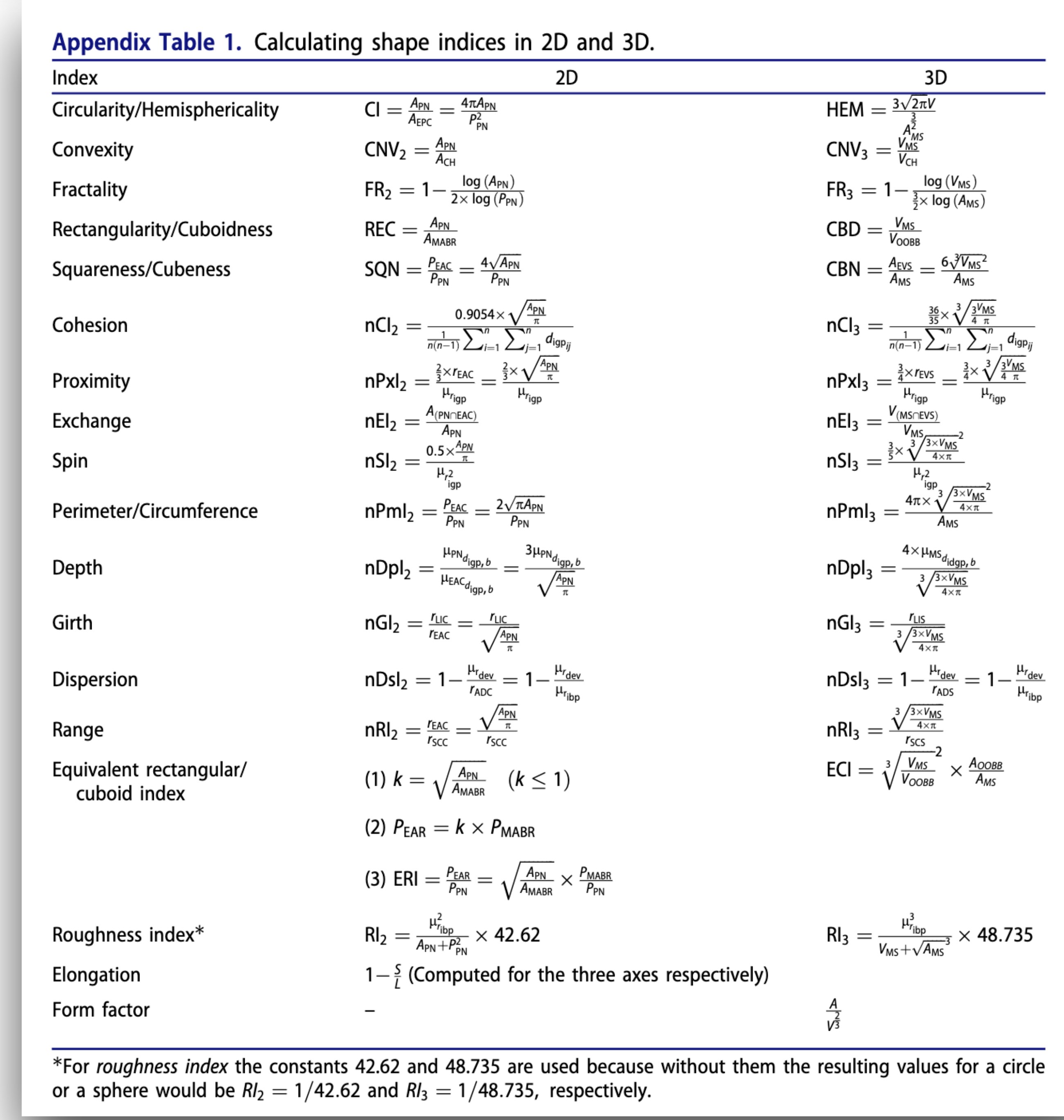
ABSTRACT

Urban morphology is important in a broad range of investigations across the fields of city planning, transportation, climate, energy, and urban data science. Characterising buildings with a set of numerical metrics is fundamental to studying the urban form. Despite the rapid developments in 3D geoinformation science, and the growing 3D data availability, most studies simplify buildings to their 2D footprint, and when taking their height into account, they at most assume one height value per building, i.e. simple 3D. We take the first step in elevating building metrics into full/true 3D, uncovering the use of higher levels of detail, and taking into account the detailed shape of a building. We set the foundation of the new research line on 3D urban morphology by providing a comprehensive set of 3D metrics, implementing them in openly released software, generating an open dataset containing 2D and 3D metrics for 823,000 buildings in the Netherlands, and demonstrating a use case where clusters and architectural patterns are analysed through time. Our experiments suggest the added value of 3D metrics to complement existing counterparts, reducing ambiguity, and providing advanced insights. Furthermore, we provide a comparative analysis using different levels of detail of 3D building models.

ARTICLE HISTORY
Received 18 January 2022
Accepted 17 July 2022

KEYWORDS
3D GIS; urban fabric; urban modelling; morphometrics; urban indicators



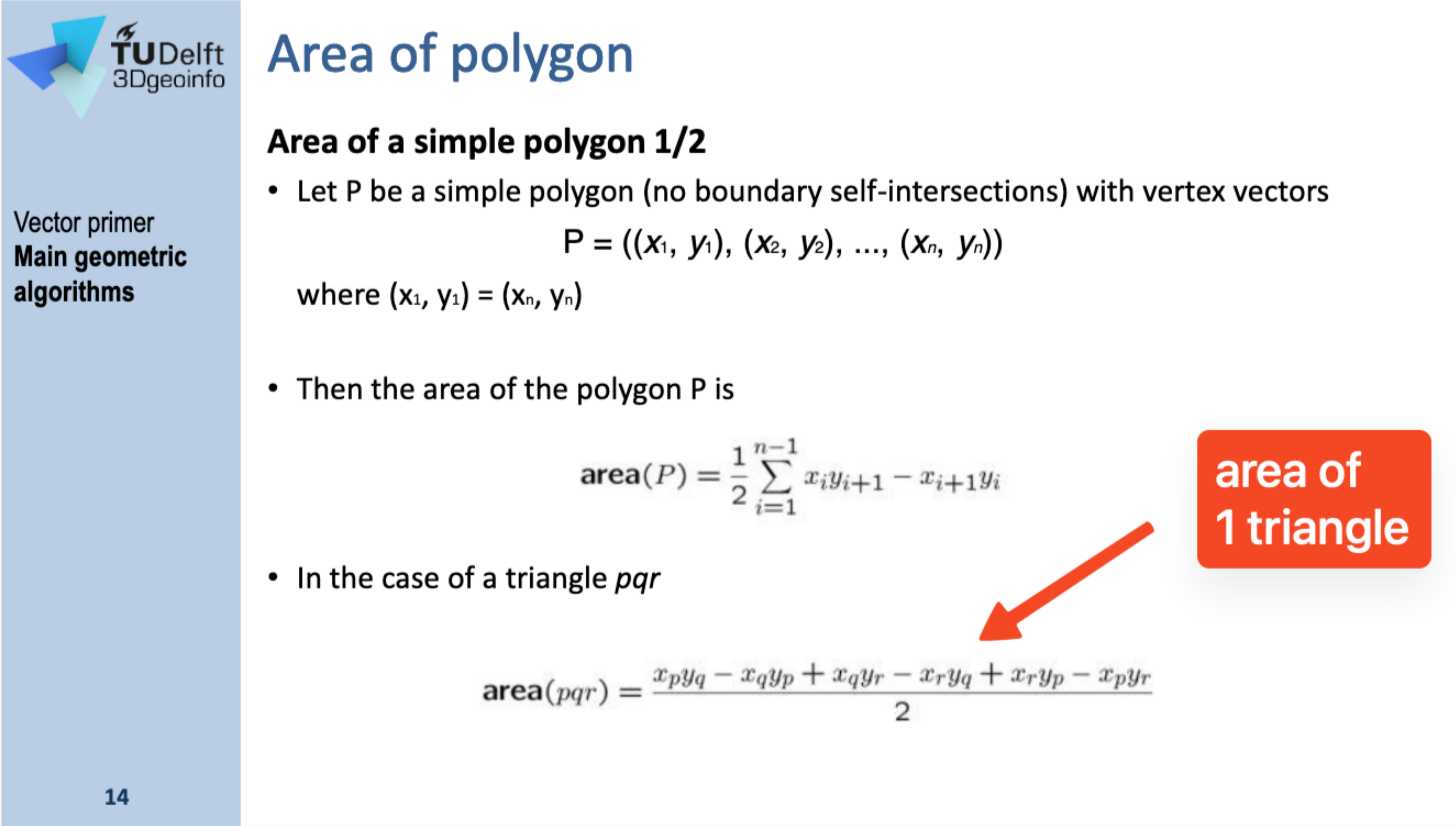


Appendix Table 1. Calculating shape indices in 2D and 3D.

Index	2D	3D
Circularity/Hemisphericality	$\mathrm{CI}=\frac{A_{PN}}{A_{EPC}}=\frac{4\pi A_{PN}}{P_{PN}^2}$	$\mathrm{HEM}=\frac{3\sqrt{2\pi}V}{A_{MS}^{\frac{3}{2}}}$
Convexity	$\mathrm{CNV}_2=\frac{A_{PN}}{A_{CH}}$	$\mathrm{CNV}_3=\frac{V_{MS}}{V_{CH}}$
Fractality	$\mathrm{FR}_2=1-\frac{\log(A_{PN})}{2\times\log(P_{PN})}$	$\mathrm{FR}_3=1-\frac{\log(V_{MS})}{\frac{3}{2}\times\log(A_{MS})}$
Rectangularity/Cuboidness	$\mathrm{REC}=\frac{A_{PN}}{A_{MABR}}$	$\mathrm{CBD}=\frac{V_{MS}}{V_{OOBB}}$
Squareness/Cubeness	$\mathrm{SQN}=\frac{P_{EAC}}{P_{PN}}=\frac{4\sqrt{A_{PN}}}{P_{PN}}$	$\mathrm{CBN}=\frac{A_{EVS}}{A_{MS}}=\frac{6\sqrt[3]{V_{MS}}^2}{A_{MS}}$
Cohesion	$\mathrm{nCI}_2=\frac{0.9054\times\sqrt{\frac{A_{PN}}{\pi}}}{\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}d_{igp_{ij}}}$	$\mathrm{nCI}_3=\frac{\frac{36}{35}\times\sqrt[3]{\frac{3}{4}\frac{V_{MS}}{\pi}}}{\frac{1}{n(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n}d_{igp_{ij}}}$
Proximity	$\mathrm{nPxI}_2=\frac{\frac{2}{3}\times r_{EAC}}{\mu_{r_{igp}}}=\frac{\frac{2}{3}\times\sqrt{\frac{A_{PN}}{\pi}}}{\mu_{r_{igp}}}$	$\mathrm{nPxI}_3=\frac{\frac{3}{4}\times r_{EVS}}{\mu_{r_{igp}}}=\frac{\frac{3}{4}\times\sqrt[3]{\frac{3}{4}\frac{V_{MS}}{\pi}}}{\mu_{r_{igp}}}$
Exchange	$\mathrm{nEI}_2=\frac{A_{(PN\cap EAC)}}{A_{PN}}$	$\mathrm{nEI}_3=\frac{V_{(MS\cap EVS)}}{V_{MS}}$
Spin	$\mathrm{nSI}_2=\frac{0.5\times\frac{A_{PN}}{\pi}}{\mu_{r^2_{igp}}}$	$\mathrm{nSI}_3=\frac{\frac{3}{5}\times\sqrt[3]{\frac{3\times V_{MS}}{4\times\pi}}^2}{\mu_{r^2_{igp}}}$
Perimeter/Circumference	$\mathrm{nPmI}_2=\frac{P_{EAC}}{P_{PN}}=\frac{2\sqrt{\pi A_{PN}}}{P_{PN}}$	$\mathrm{nPmI}_3=\frac{4\pi\times\sqrt[3]{\frac{3\times V_{MS}}{4\times\pi}}^2}{A_{MS}}$
Depth	$\mathrm{nDpI}_2=\frac{\mu_{PN_{d_{igp,b}}}}{\mu_{EAC_{d_{igp,b}}}}=\frac{3\mu_{PN_{d_{igp,b}}}}{\sqrt{\frac{A_{PN}}{\pi}}}$	$\mathrm{nDpI}_3=\frac{4\times\mu_{MS_{d_{idgp,b}}}}{\sqrt[3]{\frac{3\times V_{MS}}{4\times\pi}}}$
Girth	$\mathrm{nGI}_2=\frac{r_{LIC}}{r_{EAC}}=\frac{r_{LIC}}{\sqrt{\frac{A_{PN}}{\pi}}}$	$\mathrm{nGI}_3=\frac{r_{LIS}}{\sqrt[3]{\frac{3\times V_{MS}}{4\times\pi}}}$
Dispersion	$\mathrm{nDsI}_2=1-\frac{\mu_{r_{dev}}}{r_{ADC}}=1-\frac{\mu_{r_{dev}}}{\mu_{r_{ibp}}}$	$\mathrm{nDsI}_3=1-\frac{\mu_{r_{dev}}}{r_{ADS}}=1-\frac{\mu_{r_{dev}}}{\mu_{r_{ibp}}}$
Range	$\mathrm{nRI}_2=\frac{r_{EAC}}{r_{SCC}}=\frac{\sqrt{\frac{A_{PN}}{\pi}}}{r_{SCC}}$	$\mathrm{nRI}_3=\frac{\sqrt[3]{\frac{3\times V_{MS}}{4\times\pi}}}{r_{SCS}}$
Equivalent rectangular/ cuboid index	(1) $k=\sqrt{\frac{A_{PN}}{A_{MABR}}}\quad(k\le 1)$	$\mathrm{ECI}=\sqrt[3]{\frac{V_{MS}}{V_{OOBB}}}^2\times\frac{A_{OOBB}}{A_{MS}}$
	(2) $P_{EAR}=k\times P_{MABR}$	
	(3) $\mathrm{ERI}=\frac{P_{EAR}}{P_{PN}}=\sqrt{\frac{A_{PN}}{A_{MABR}}}\times\frac{P_{MABR}}{P_{PN}}$	
Roughness index*	$\mathrm{RI}_2=\frac{\mu^2_{r_{ibp}}}{A_{PN}+P^2_{PN}}\times 42.62$	$\mathrm{RI}_3=\frac{\mu^3_{r_{ibp}}}{V_{MS}+\sqrt{A_{MS}}^3}\times 48.735$
Elongation	$1-\frac{S}{L}$ (Computed for the three axes respectively)	
Form factor	–	$\frac{A}{V^{\frac{2}{3}}}$

*For *roughness index* the constants 42.62 and 48.735 are used because without them the resulting values for a circle or a sphere would be $RI_2=1/42.62$ and $RI_3=1/48.735$, respectively.

How to calculate the volume of a (invalid) solid



TUDelft 3Dgeoinfo

Vector primer
Main geometric algorithms

Area of polygon

Area of a simple polygon 1/2

- Let P be a simple polygon (no boundary self-intersections) with vertex vectors

  $$P = ((x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n))$$

  where $(x_1, y_1) = (x_n, y_n)$

- Then the area of the polygon P is

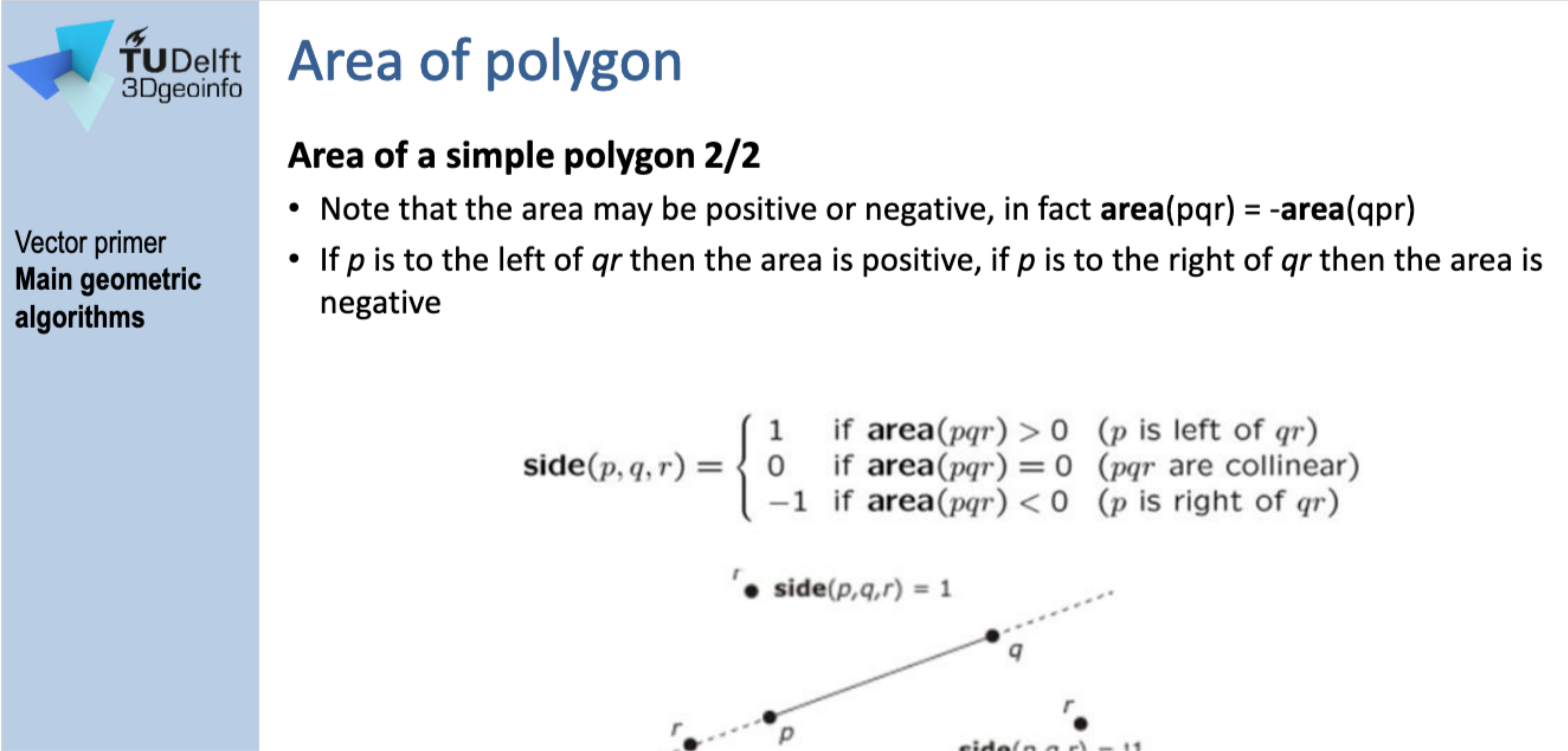
$$\text{area}(P) = \frac{1}{2}\sum_{i=1}^{n-1} x_i y_{i+1} - x_{i+1} y_i$$

- In the case of a triangle *pqr*

$$\text{area}(pqr) = \frac{x_p y_q - x_q y_p + x_q y_r - x_r y_q + x_r y_p - x_p y_r}{2}$$

area of 1 triangle

14



TUDelft 3Dgeoinfo

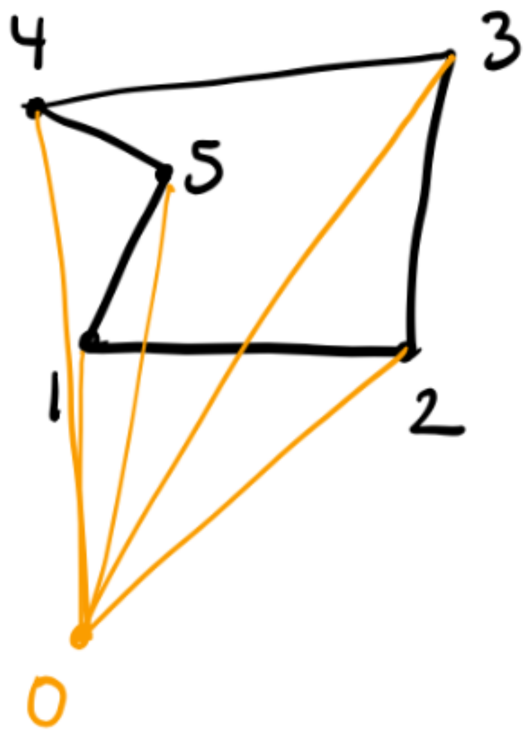
Vector primer
Main geometric algorithms

Area of polygon

Area of a simple polygon 2/2

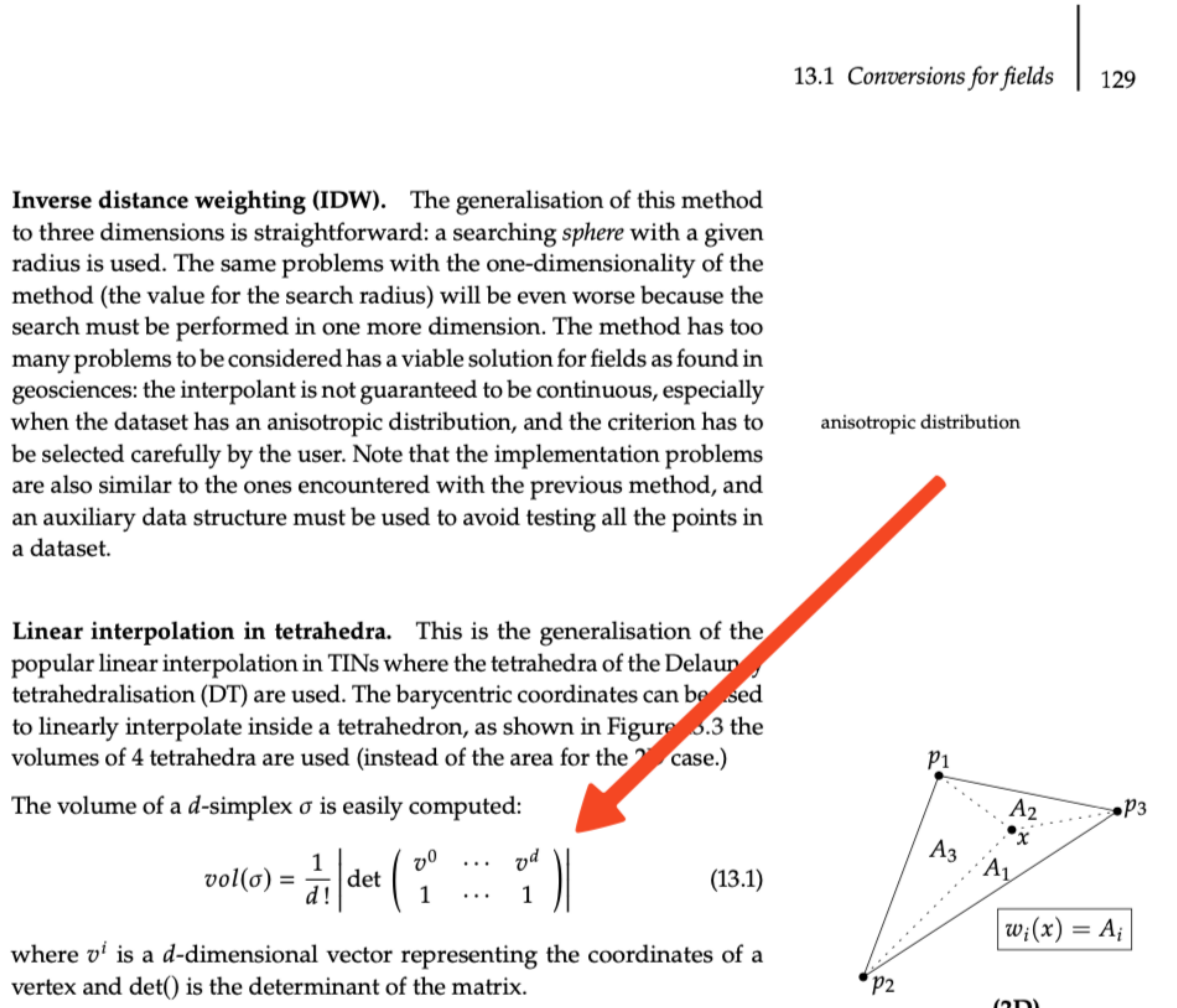
- Note that the area may be positive or negative, in fact **area**(pqr) = -**area**(qpr)
- If *p* is to the left of *qr* then the area is positive, if *p* is to the right of *qr* then the area is negative

$$\text{side}(p,q,r) = \begin{cases} 1 & \text{if } \text{area}(pqr) > 0 \quad (p \text{ is left of } qr) \\ 0 & \text{if } \text{area}(pqr) = 0 \quad (pqr \text{ are collinear}) \\ -1 & \text{if } \text{area}(pqr) < 0 \quad (p \text{ is right of } qr) \end{cases}$$



Area → signed area

	Δ 012	–	negative
	Δ 023	–	positive
+	Δ 034	–	positive
	Δ 045	–	negative
	Δ 051	–	positive



13.1 *Conversions for fields* | 129

Inverse distance weighting (IDW). The generalisation of this method to three dimensions is straightforward: a searching *sphere* with a given radius is used. The same problems with the one-dimensionality of the method (the value for the search radius) will be even worse because the search must be performed in one more dimension. The method has too many problems to be considered has a viable solution for fields as found in geosciences: the interpolant is not guaranteed to be continuous, especially when the dataset has an anisotropic distribution, and the criterion has to be selected carefully by the user. Note that the implementation problems are also similar to the ones encountered with the previous method, and an auxiliary data structure must be used to avoid testing all the points in a dataset.

anisotropic distribution

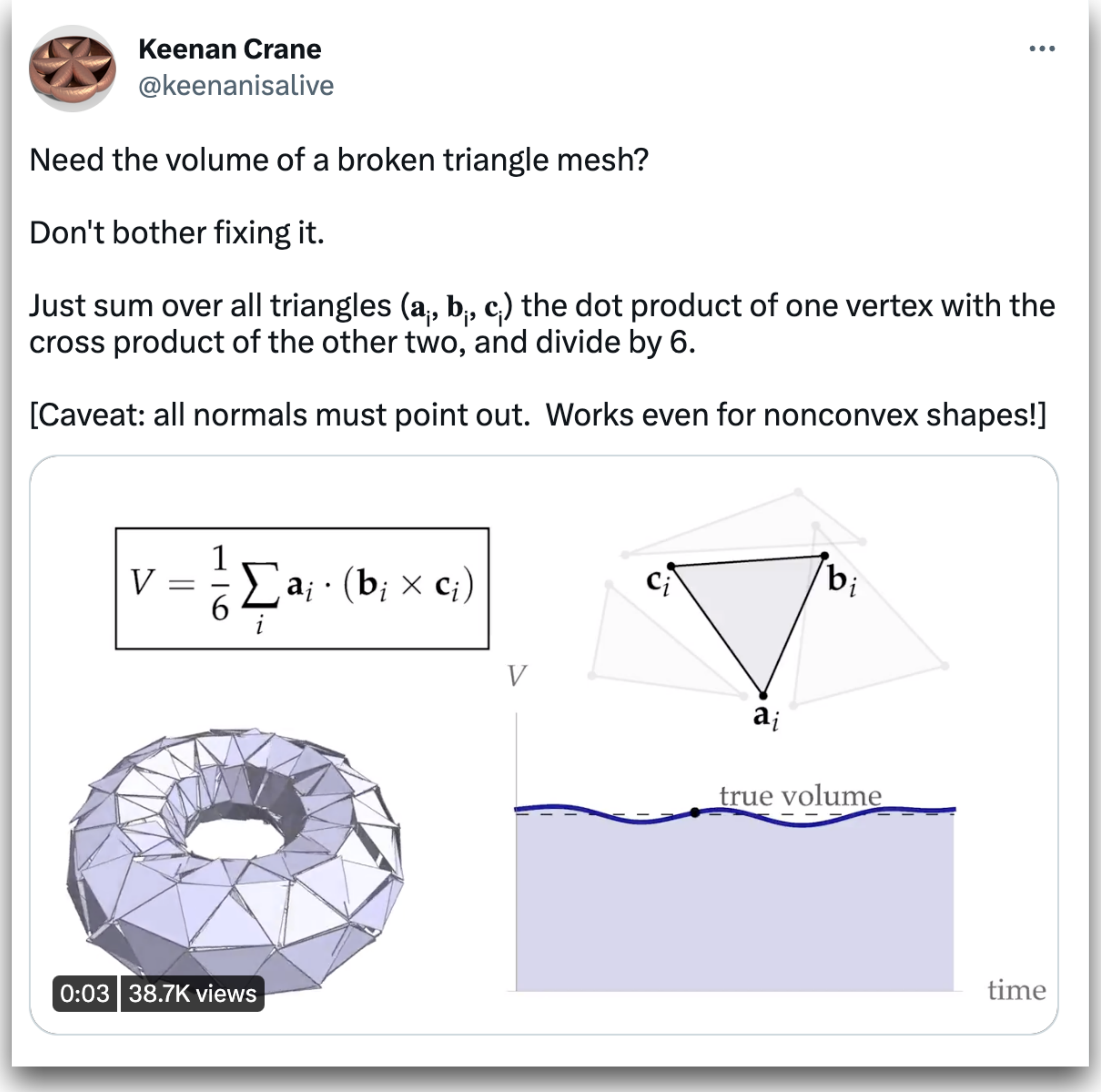
Linear interpolation in tetrahedra. This is the generalisation of the popular linear interpolation in TINs where the tetrahedra of the Delaunay tetrahedralisation (DT) are used. The barycentric coordinates can be used to linearly interpolate inside a tetrahedron, as shown in Figure 13.3 the volumes of 4 tetrahedra are used (instead of the area for the 2D case.)

The volume of a d-simplex σ is easily computed:

$$vol(\sigma) = \frac{1}{d\,!}\left|\det\begin{pmatrix} v^0 & \cdots & v^d \\ 1 & \cdots & 1 \end{pmatrix}\right| \tag{13.1}$$

where v^i is a d-dimensional vector representing the coordinates of a vertex and det() is the determinant of the matrix.

How to calculate the volume of a (invalid) solid



Keenan Crane
@keenanisalive

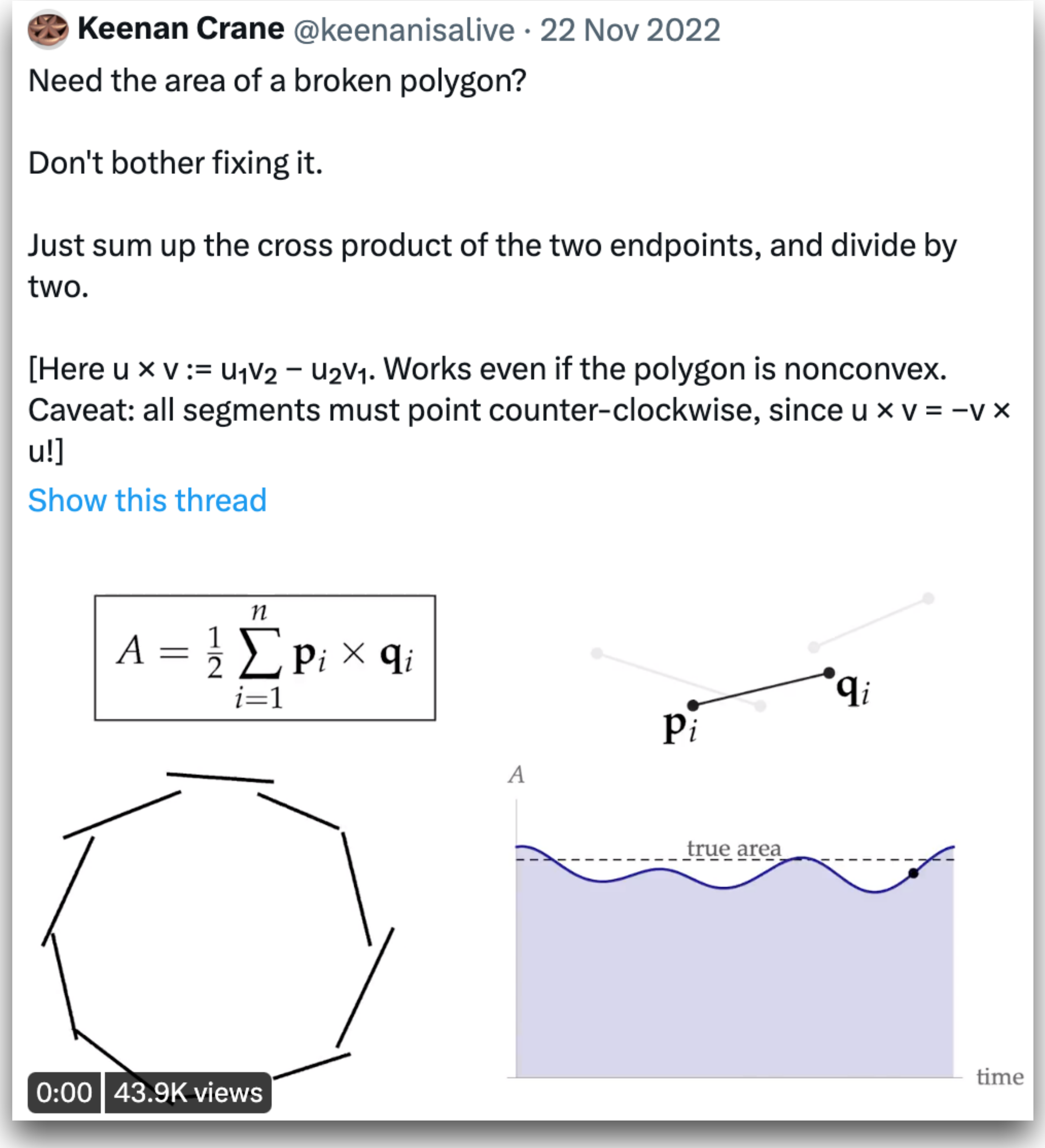
Need the volume of a broken triangle mesh?

Don't bother fixing it.

Just sum over all triangles ($\mathbf{a}_i$, $\mathbf{b}_i$, $\mathbf{c}_i$) the dot product of one vertex with the cross product of the other two, and divide by 6.

[Caveat: all normals must point out. Works even for nonconvex shapes!]

$$V = \frac{1}{6}\sum_i \mathbf{a}_i \cdot (\mathbf{b}_i \times \mathbf{c}_i)$$



Keenan Crane @keenanisalive · 22 Nov 2022

Need the area of a broken polygon?

Don't bother fixing it.

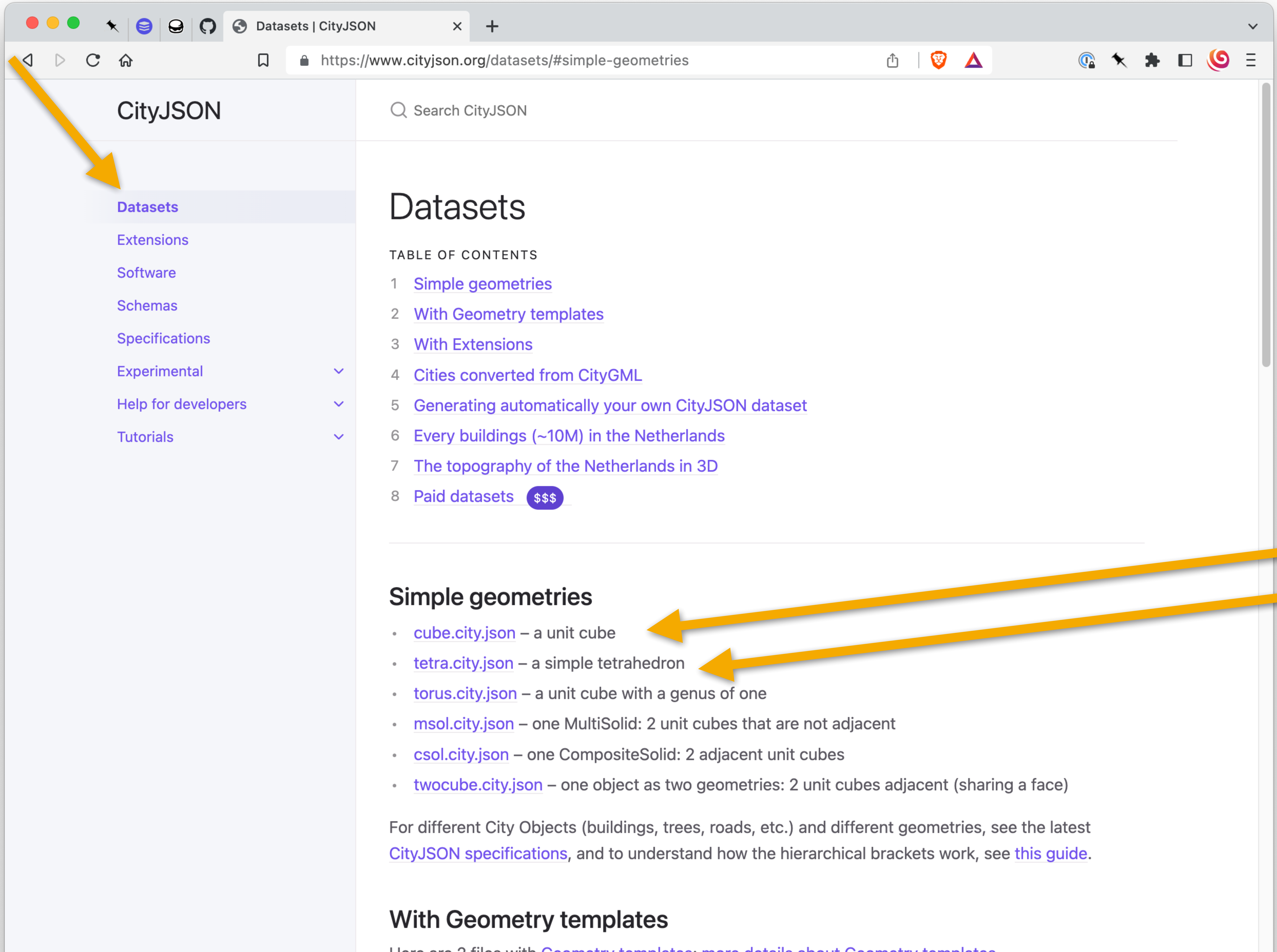
Just sum up the cross product of the two endpoints, and divide by two.

[Here u × v := $u_1v_2 - u_2v_1$. Works even if the polygon is nonconvex. Caveat: all segments must point counter-clockwise, since u × v = −v × u!]

Show this thread

$$A = \frac{1}{2}\sum_{i=1}^{n} \mathbf{p}_i \times \mathbf{q}_i$$

Tip: start with simple shapes, not BK-City...



Datasets

TABLE OF CONTENTS

Simple geometries

- cube.city.json – a unit cube
- tetra.city.json – a simple tetrahedron
- torus.city.json – a unit cube with a genus of one
- msol.city.json – one MultiSolid: 2 unit cubes that are not adjacent
- csol.city.json – one CompositeSolid: 2 adjacent unit cubes
- twocube.city.json – one object as two geometries: 2 unit cubes adjacent (sharing a face)

For different City Objects (buildings, trees, roads, etc.) and different geometries, see the latest CityJSON specifications, and to understand how the hierarchical brackets work, see this guide.

With Geometry templates

Here are 2 files with Geometry templates; more details about Geometry templates

Download and modify those