Lesson 6.1
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 & XML/GML

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 based on a subset of JavaScript
- 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
   - also called a “hash”
   - C++ std::map
2. arrays

Data types:

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

To represent a person

```
{
  "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
}
```

XML (e**X**tensible **M**arkup **L**anguage)

You'll learn this in
GEO1007 + GEO5014

- a markup language that defines a set of rules for encoding information and documents
- both human-readable and machine-readable
- designed to store and transport data
- it is self-descriptive (what data "is")
- designed to focus on documents, but widely used for the representation of arbitrary data structures
- hierarchical structure (a tree)
- examples of XML? ...

Sources:
https://en.wikipedia.org/wiki/XML
https://www.w3schools.com/xml/xml_whatis.asp

XML (e**X**tensible **M**arkup **L**anguage)

- a markup language that defines a set of rules for encoding information and documents
- both human-readable and machine-readable
- designed to store and transport data
- it is self-descriptive (what data “is”)
- designed to focus on documents, but widely used for the representation of arbitrary data structures
- hierarchical structure (a tree)
- examples of XML? HTML, GML, KML, DOCX, RSS, SVG, COLLADA, GPX, etc.

Sources:
https://en.wikipedia.org/wiki/XML
https://www.w3schools.com/xml/xml_whatis.asp

XML (e**X**tensible **M**arkup **L**anguage)

```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- no predefined tags
- tags are <note>, <from>, etc.
- authors must define both the tags and the document structure, the latter is done with a *schema*

Sources:
https://en.wikipedia.org/wiki/XML
https://www.w3schools.com/xml/xml_whatis.asp

XML elements (or nodes)

```xml
<note>
  <to>Tove</to>
  <from>Jani</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

- start-tag: <from>
- end-tag: </from>
- empty-element tag: <from/>

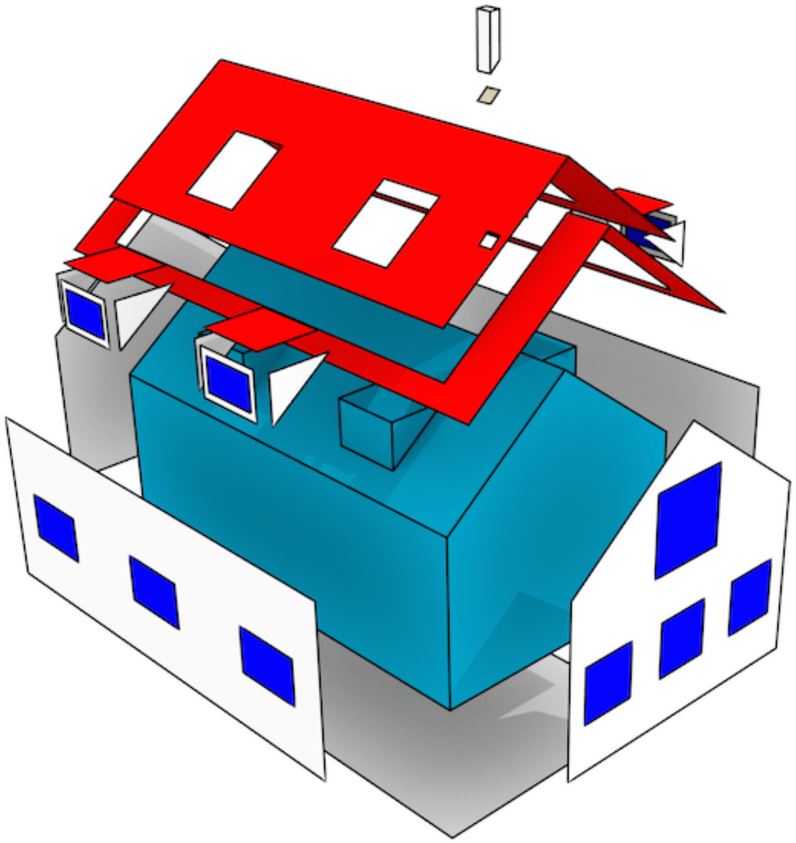
- element can be either:
  - string
  - another element (hierarchy)

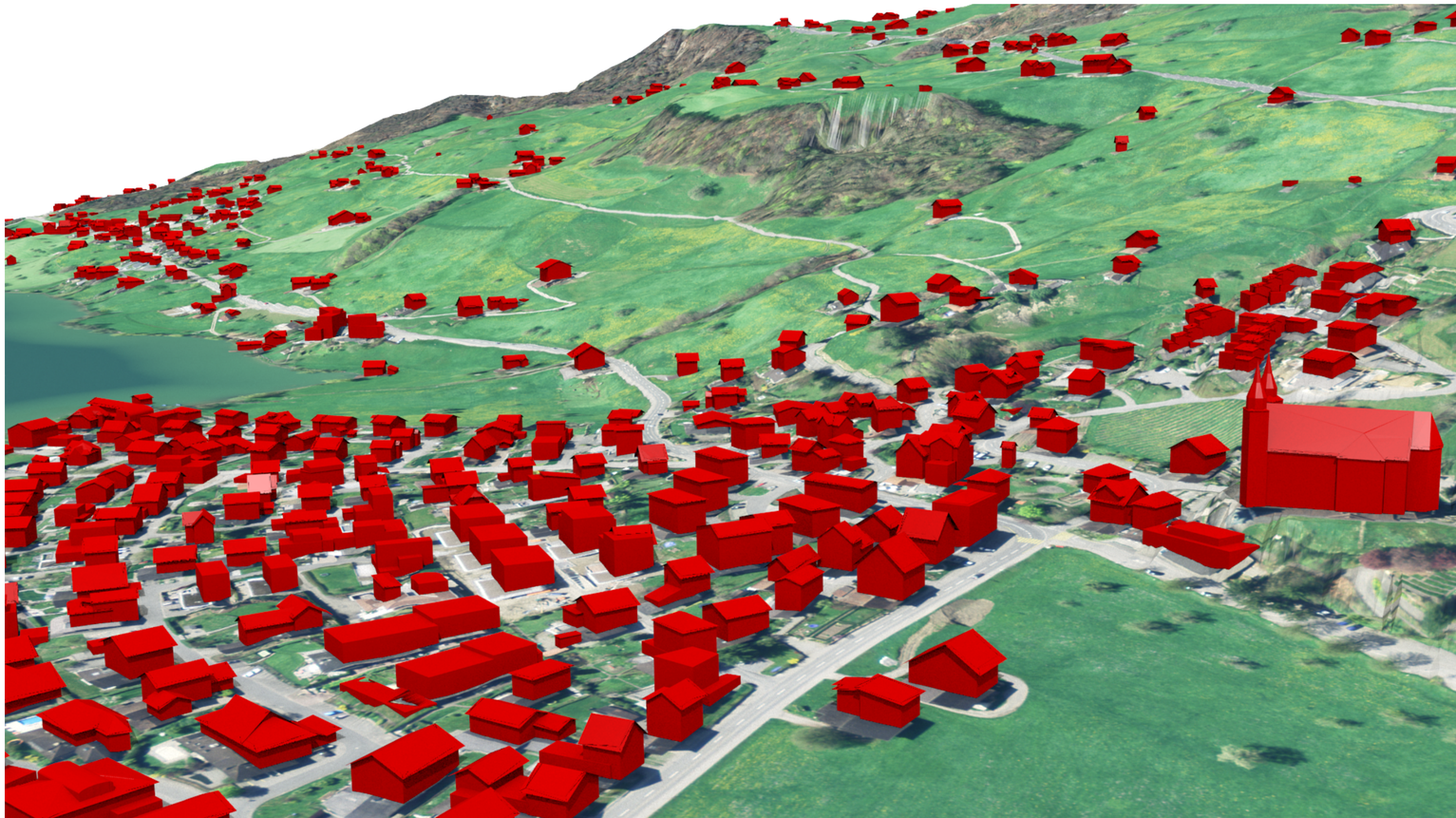
Sources:
https://en.wikipedia.org/wiki/XML
https://www.w3schools.com/xml/xml_whatis.asp

3D city modelling

CityGML

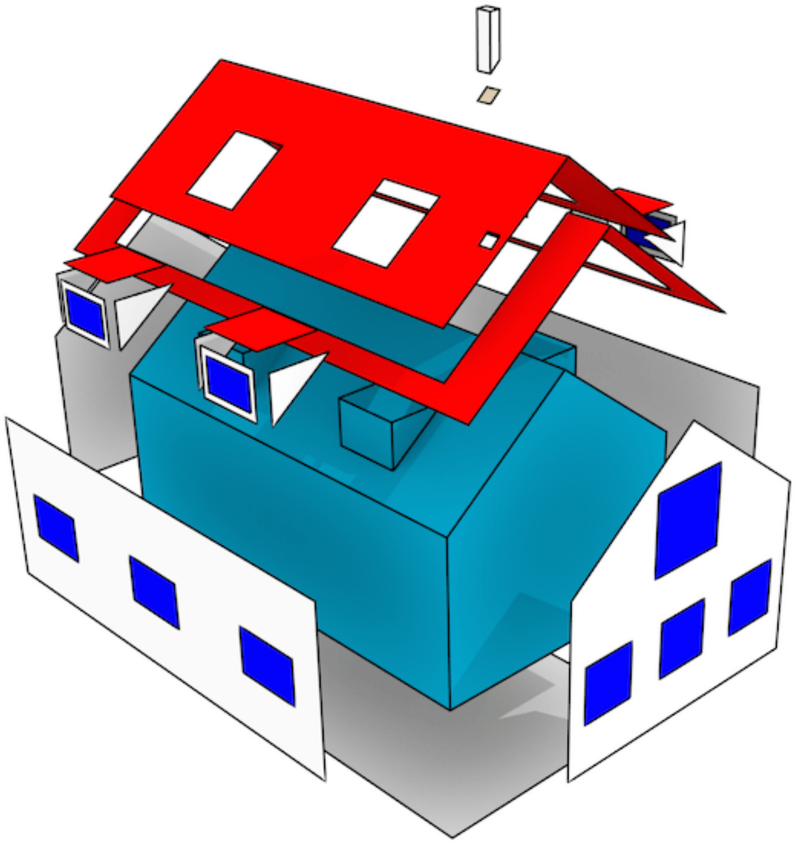
- International standard (from OGC) for representing and storing 3D city models
- Currently at version 3.0
- Both a data model & an encoding (confusing, huh?)

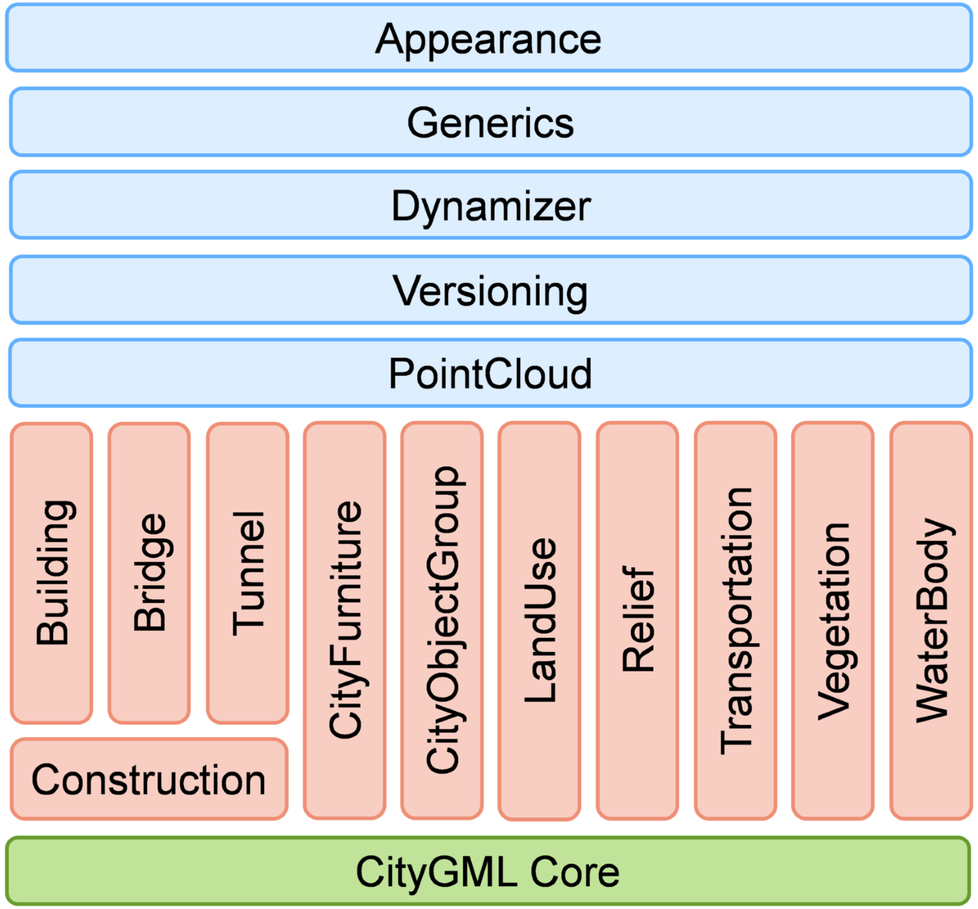




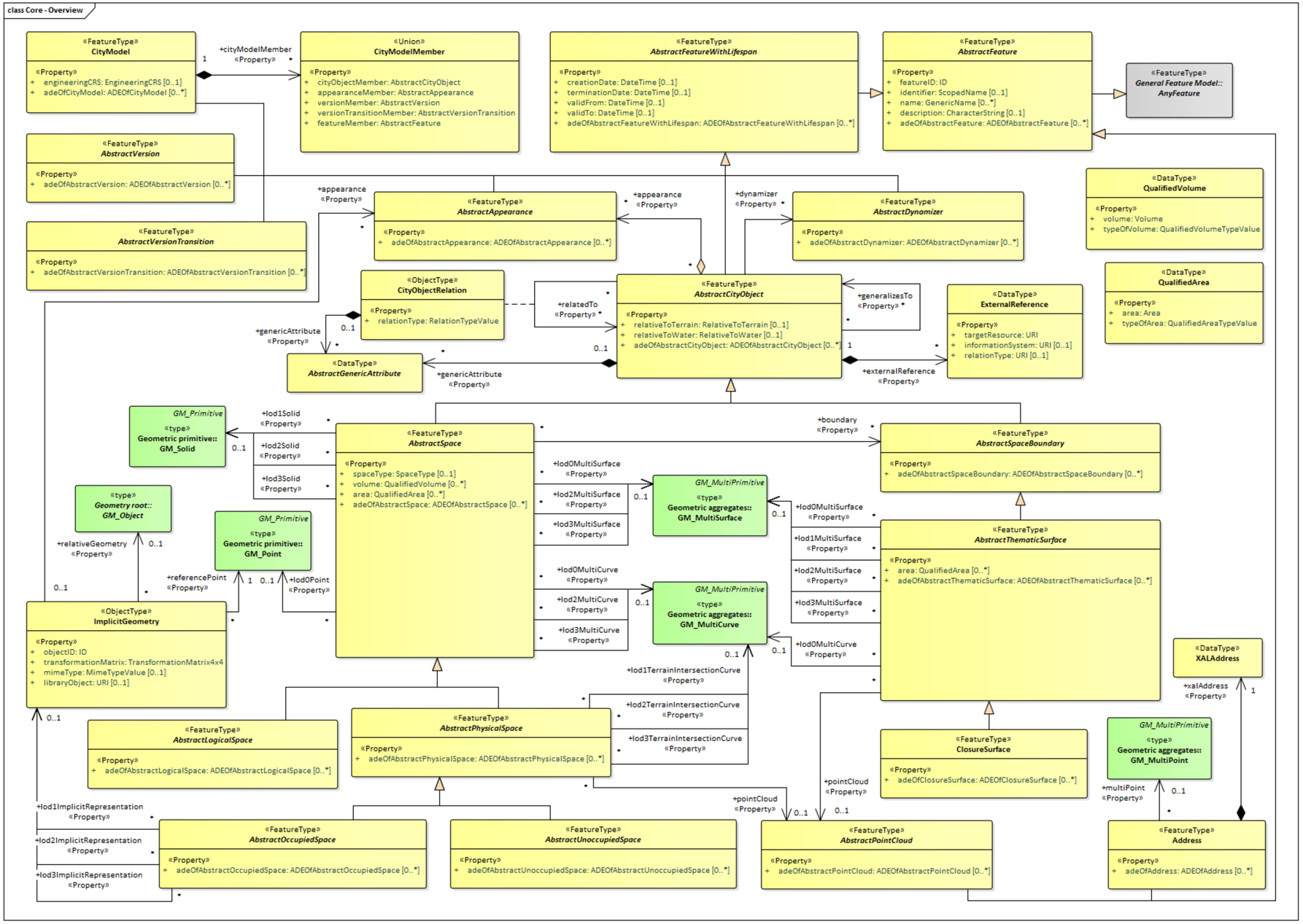
CityGML

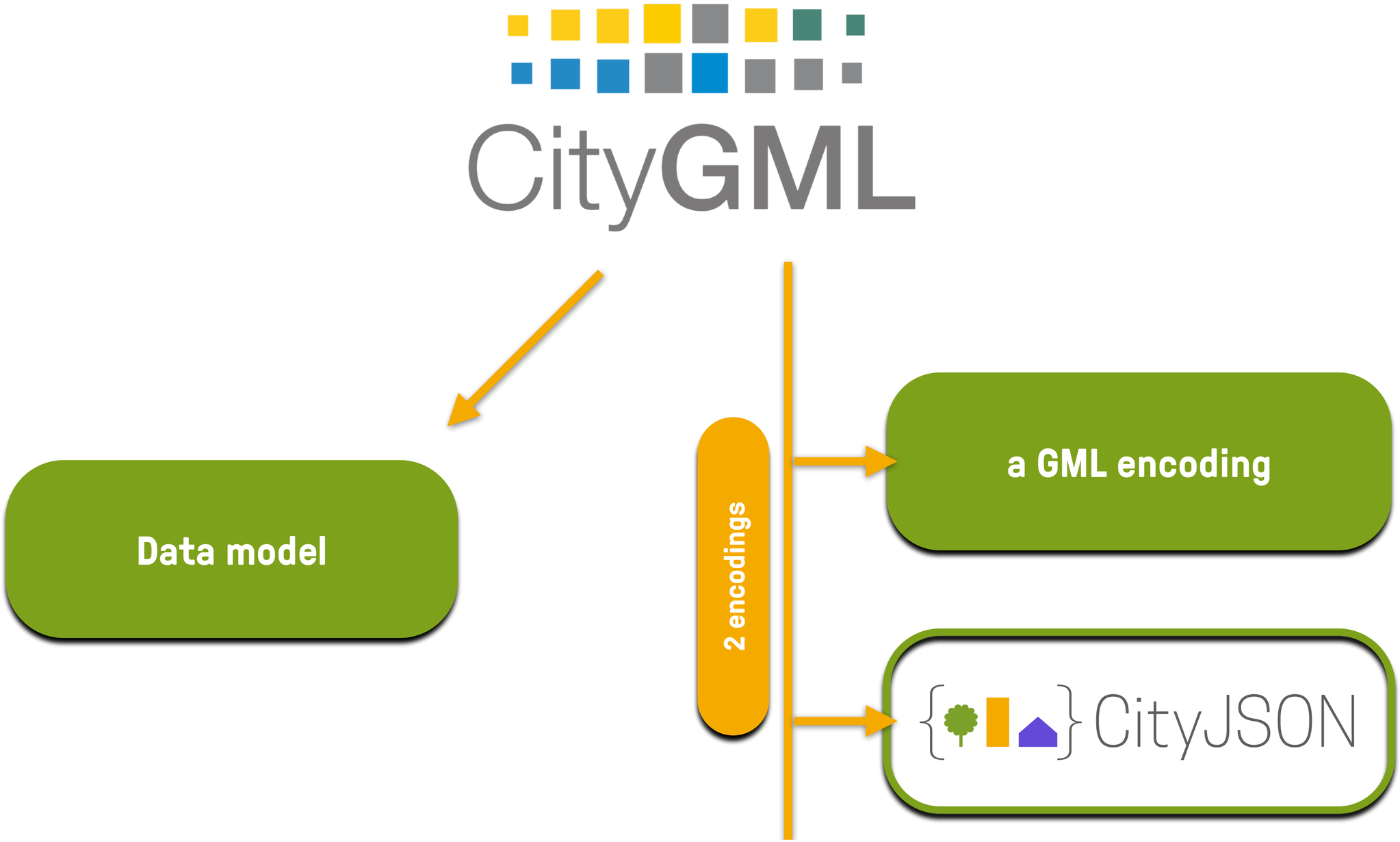
- International standard (from OGC) for representing and storing 3D city models
- Currently at version 3.0
- Both a data model & an encoding (confusing, huh?)





CityGML == a data model == UML models




CityGML
Data model
2 encodings
a GML encoding
CityJSON

CityGML-XML files are very complex

- files are deeply nested, and large
- many “points of entry”
- many diff ways to do one thing

➡ few software packages use CityGML
➡ no parsers in JavaScript
➡ I personally get 😩 each time I get a new file

MSc Geomatics students when trying to parse a CityGML file in Python

Why an alternative encoding? Read this.

Ledoux *et al. Open Geospatial Data, Software and Standards* (2019) 4:4
https://doi.org/10.1186/s40965-019-0064-0

Open Geospatial Data, Software and Standards

ORIGINAL ARTICLE Open Access

CityJSON: a compact and easy-to-use encoding of the CityGML data model

Hugo Ledoux*, Ken Arroyo Ohori, Kavisha Kumar, Balázs Dukai, Anna Labetski and Stelios Vitalis

Abstract

The international standard CityGML is both a data model and an exchange format to store digital 3D models of cities. While the data model is used by several cities, companies, and governments, in this paper we argue that its XML-based exchange format has several drawbacks. These drawbacks mean that it is difficult for developers to implement parsers for CityGML, and that practitioners have, as a consequence, to convert their data to other formats if they want to exchange them with others. We present CityJSON, a new JSON-based exchange format for the CityGML data model (version 2.0.0). CityJSON was designed with programmers in mind, so that software and APIs supporting it can be quickly built. It was also designed to be compact (a compression factor of around six with real-world datasets), and to be friendly for web and mobile development. We argue that it is considerably easier to use than the CityGML format, both for reading and for creating datasets. We discuss in this paper the main features of CityJSON, briefly

Home | CityJSON

https://www.cityjson.org

CityJSON

Search CityJSON

- Datasets
- Extensions
- Software
- Schemas
- Specifications
- Experimental
- Help for developers
- Tutorials

CityJSON

A JSON-based encoding for 3D city models

Getting started | NEW! Specs (v1.1.1) | Web-viewer | Validator

Latest news

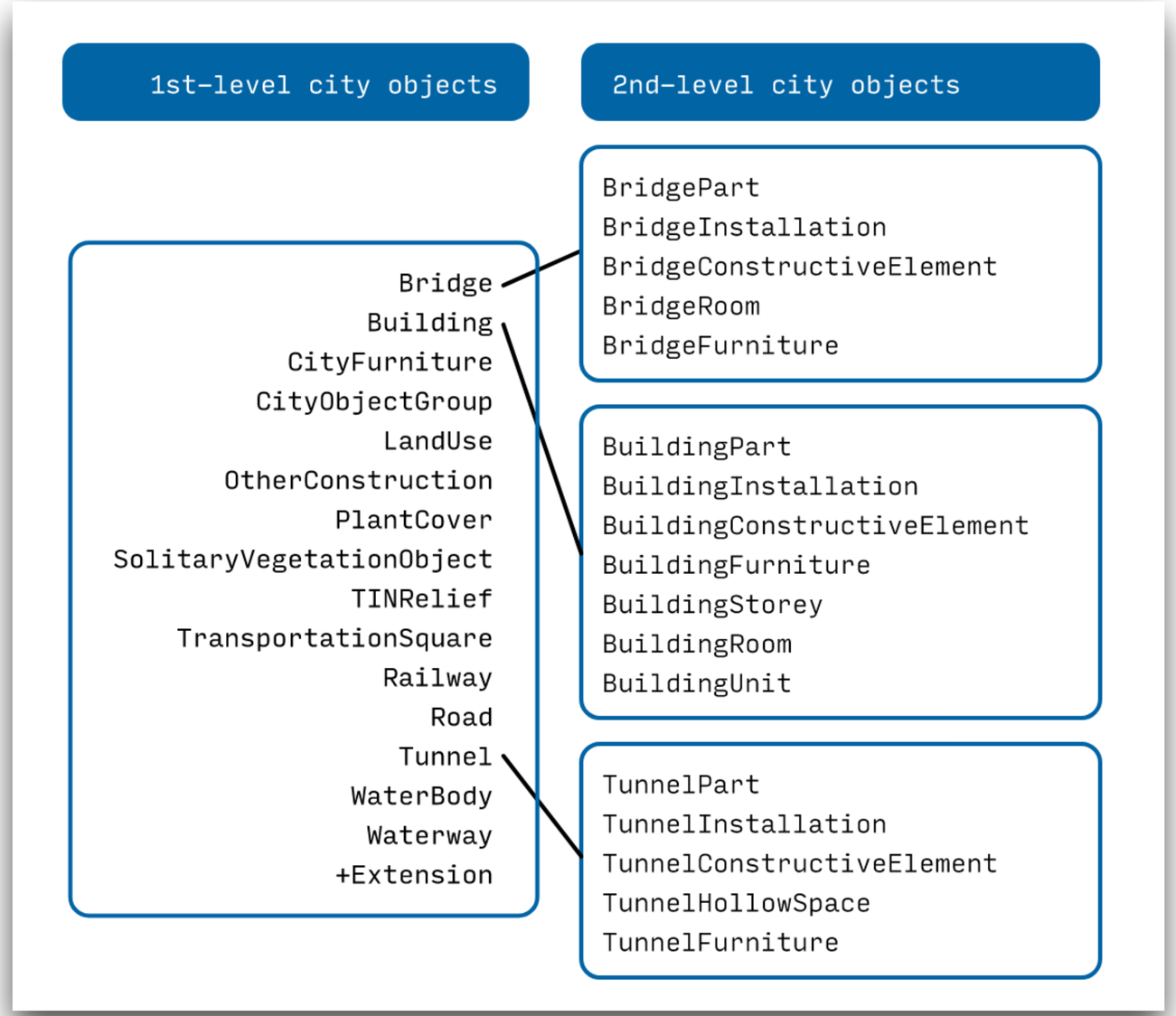
17 JAN 2022 CityJSON now has its own media type

01 DEC 2021 CityJSON v1.1 is released!

16 AUG 2021 CityJSON v1.0 is now an OGC standard!

version 1.1 = current version

- OGC® Making location count. community standard
- compliant with CityGML v3.0
- subset of CityGML (~90% of features)
- software for full conversion CityGML <-> CityJSON
- several software support CityJSON



Same information as CityGML, but in JSON format

```
{
  "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": []
  }
}
```

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": []
  }
}
```

version CityJSON

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": []
  }
}
```

metadata possible, eg:

- CRS
- Point of contact
- bbox of dataset

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": []
  }
}
```

All City Objects listed here, *indexed* by their ID

Each have geometries + attributes

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": []
  }
}
```

Geometry is ID of the vertex, global list

compression + more “topology”

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": []
  }
}
```

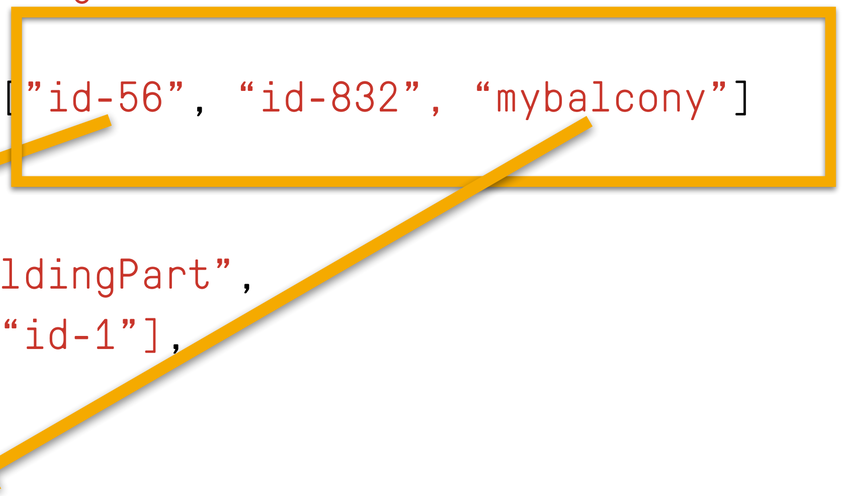
material + texture possible

BuildingParts: links between City Objects

```
"CityObjects": {
  "id-1": {
    "type": "Building",
    "attributes": {
      "roofType": "gable"
    },
    "children": ["id-56", "id-832", "mybalcony"]
  },
  "id-56": {
    "type": "BuildingPart",
    "parents": ["id-1"],
    ...
  },
  "mybalcony": {
    "type": "BuildingInstallation",
    "parent": ["id-1"],
    ...
  }
}
```

BuildingParts: links between City Objects

```
"CityObjects": {
  "id-1": {
    "type": "Building",
    "attributes": {
      "roofType": "gable"
    },
    "children": ["id-56", "id-832", "mybalcony"]
  },
  "id-56": {
    "type": "BuildingPart",
    "parents": ["id-1"],
    ...
  },
  "mybalcony": {
    "type": "BuildingInstallation",
    "parents": ["id-1"],
    ...
  }
}
```



goal == a flat schema

Specifications give all the gory details

CityJSON Specifications 1.1.0

https://www.cityjson.org/specs/1.1.0/#the-coordinates-of-the-vertices

TABLE OF CONTENTS

§ 3.1. The coordinates of the vertices

A CityJSON must have one member named "`vertices`", whose value is an array of coordinates of each vertex of the city model. Their position in this array (0-based) is used to represent the Geometry Objects.

- one vertex **must** be an array with exactly 3 values, representing the *(x,y,z)* location of the vertex.
- the array of vertices may be empty.
- vertices may be repeated

```
"vertices": [
  [0.0, 0.0, 0.0],
  [1.0, 0.0, 0.0],
  [0.0, 0.0, 0.0],
  ...
  [1.0, 0.0, 0.0],
  [8523.134, 487625.134, 2.03]
]
```

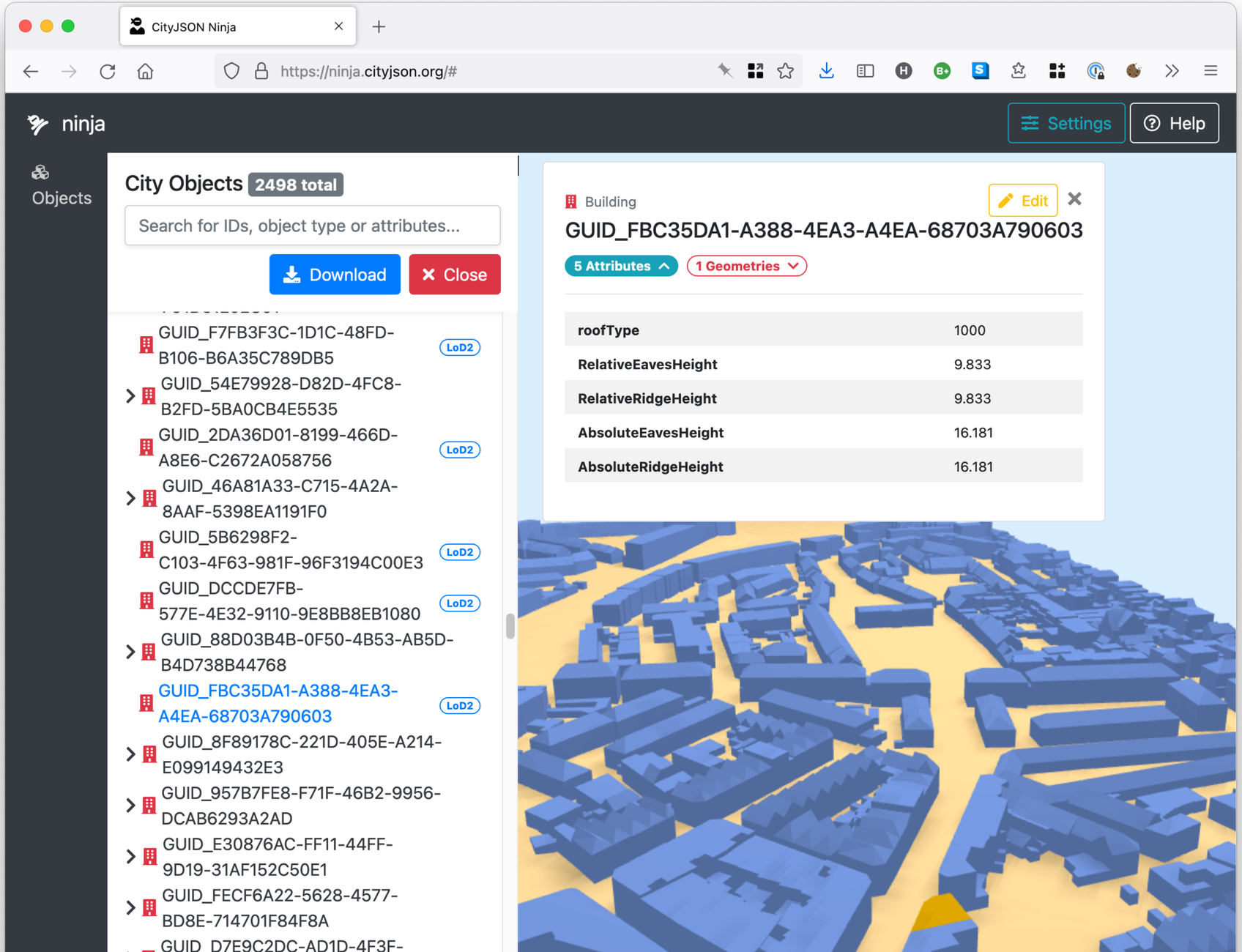
§ 3.2. Arrays to represent boundaries

The depth of the hierarchy of arrays depends on the Geometry object, and is as follows.

- A "`MultiPoint`" has an array with the indices of the vertices; this array can be empty.
- A "`MultiLineString`" has an array of arrays, each containing the indices of a LineString
- A "`MultiSurface`", or a "`CompositeSurface`", has an array containing surfaces, each surface is modelled by an array of array, the first array being the exterior boundary of the surface, and the others the interior boundaries.
- A "`Solid`" has an array of shells, the first array being the exterior shell of the solid, and the others the interior shells. Each shell has an array of surfaces, modelled in the exact same way as a MultiSurface/CompositeSurface.

CityJSON software

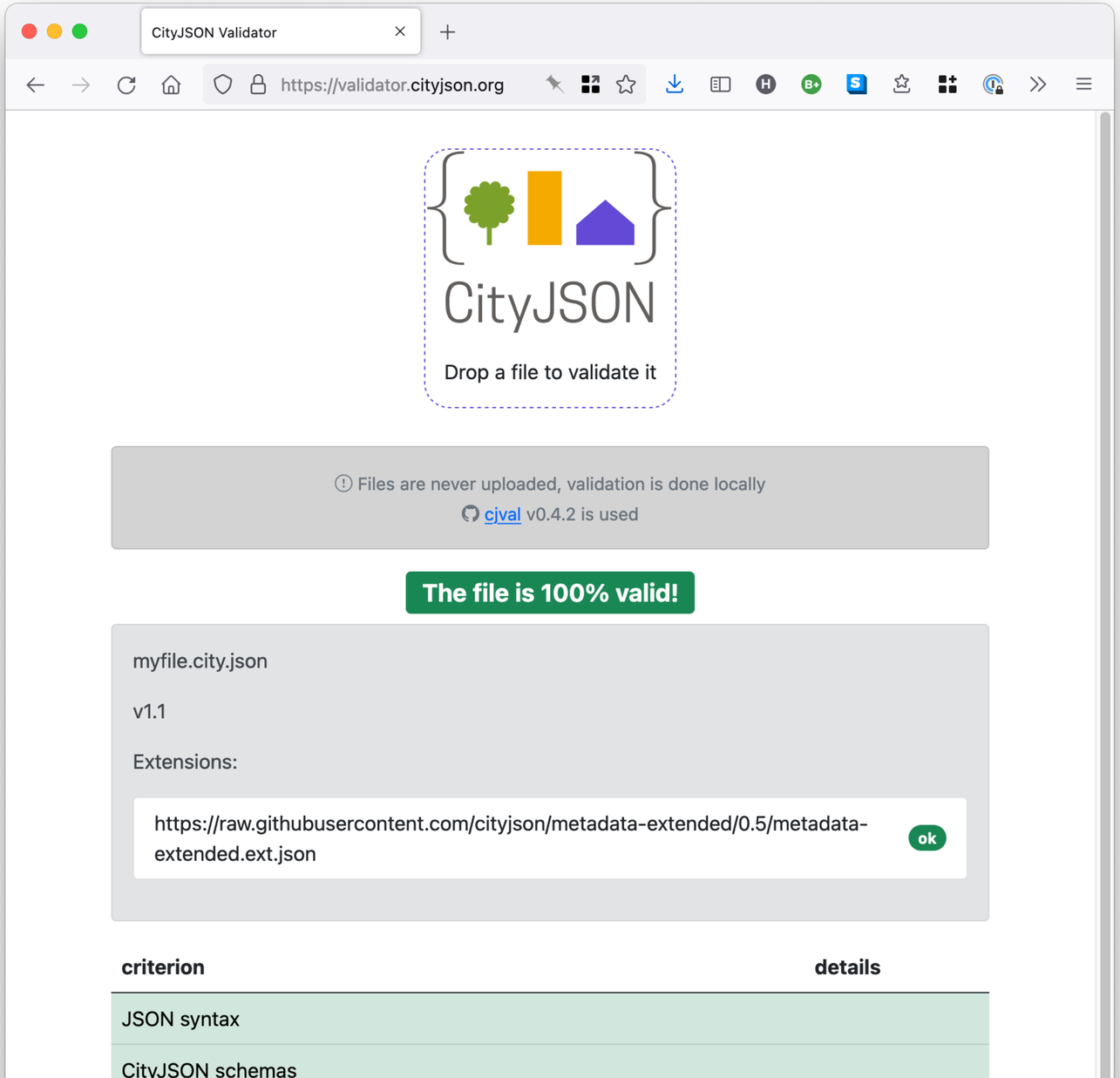
web-viewer: ninja.cityjson.org



Cool fact: programmed mostly by 2 Geomatics students, as projects for GE05010

You can do the same (or similar!)

Validation of the syntax of a file: cjval

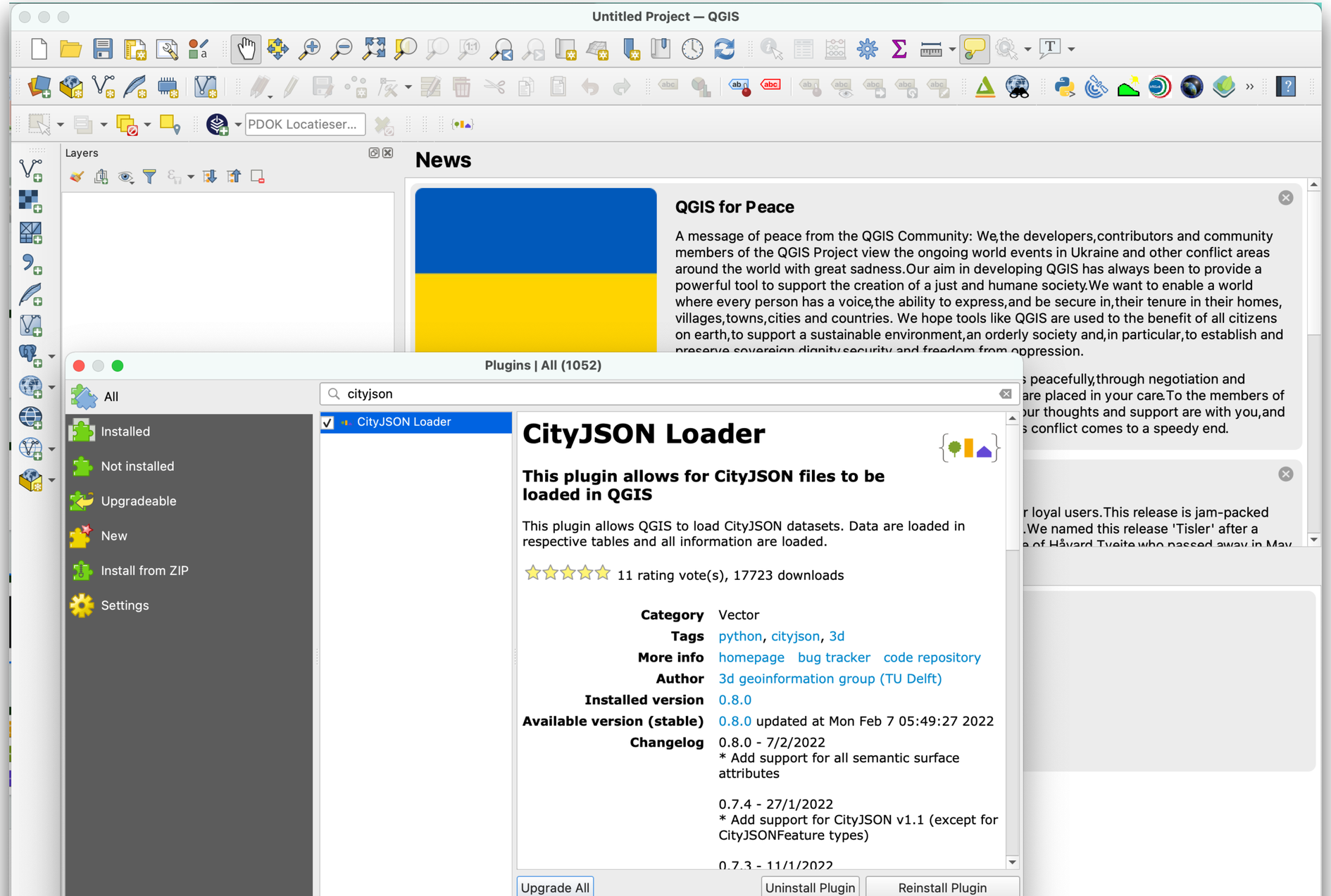



```
{
  "type": "CityJSON",
  "version": "1.1",
  "metadata": {
    "referenceSystem": "https://www.opengis.net/def/crs/EPSG/0/7415"
  },
  "transform": {…}
  "CityObjects": {
    "id-1": {
      "type": "Buildnig",
      "attributes": {
        "measuredHeight": 22.3,
        "roofType": "gable",
        "owner": "Elvis Presley"
      },
      "geometry": [
        {
          "type": "MultiSurfaec",
          "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": []
  }
}
```

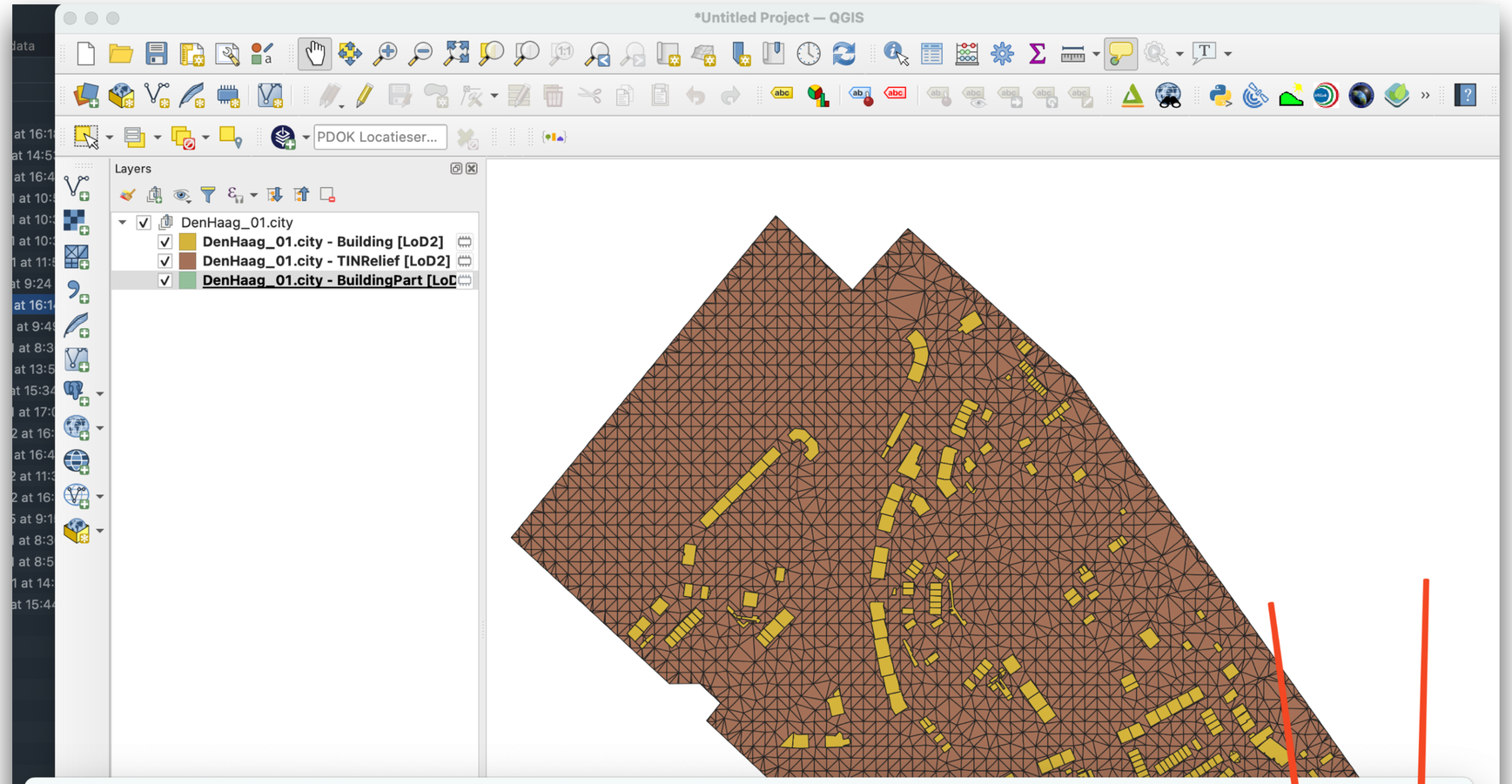
Python parser is simple

```
import json

fin = open('mycity.json')
cm = json.loads(fin.read())

print "There are", len(cm['CityObjects']), "CityObjects"

# list all ids
for id in cm['CityObjects']:
    print "\t", id
```

QGIS plugin


Untitled Project — QGIS
PDOK Locatieser...
Layers
News
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
peacefully,through negotiation and
are placed in your care.To the members of
our thoughts and support are with you,and
conflict comes to a speedy end.
loyal users.This release is jam-packed
.We named this release 'Tisler' after a
Plugins | All (1052)
All
Installed
Not installed
Upgradeable
New
Install from ZIP
Settings
cityjson
CityJSON Loader
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
Upgrade All
Uninstall Plugin
Reinstall Plugin

QGIS plugin



DenHaag_01.city - BuildingPart [LoD2] — Features Total: 13059, Filtered: 13059, Selected: 0

	uid	type	parents	children	attribute.roofType	ute.RelativeEaves	ute.RelativeRidge	ute.AbsoluteEaves	ute.AbsoluteRidge	lod	surface.type	surface.Direction	surface.Slope
1	GUID_DBDA...	BuildingPart	GUID_DBDA...	NULL	1000	3.133	3.133	7.717	7.717	2	WallSurface	NULL	NULL
2	GUID_DBDA...	BuildingPart	GUID_DBDA...	NULL	1000	3.133	3.133	7.717	7.717	2	WallSurface	NULL	NULL
3	GUID_DBDA...	BuildingPart	GUID_DBDA...	NULL	1000	3.133	3.133	7.717	7.717	2	WallSurface	NULL	NULL
4	GUID_DBDA...	BuildingPart	GUID_DBDA...	NULL	1000	3.133	3.133	7.717	7.717	2	WallSurface	NULL	NULL
5	GUID_DBDA...	BuildingPart	GUID_DBDA...	NULL	1000	3.133	3.133	7.717	7.717	2	RoofSurface	0	90
6	GUID_DBDA...	BuildingPart	GUID_DBDA...	NULL	1000	3.133	3.133	7.717	7.717	2	GroundSurface	NULL	NULL
7	GUID_8CE5...	BuildingPart	GUID_8CE5...	NULL	1000	6.392	6.392	10.075	10.075	2	WallSurface	NULL	NULL
8	GUID_8CE5...	BuildingPart	GUID_8CE5...	NULL	1000	6.392	6.392	10.075	10.075	2	WallSurface	NULL	NULL

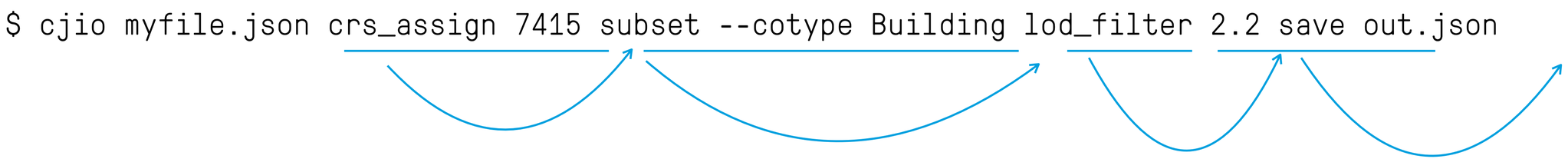
To visualise the attributes of Semantic Surfaces (essential for hw02!)

cjio (CityJSON/io)

2. bash

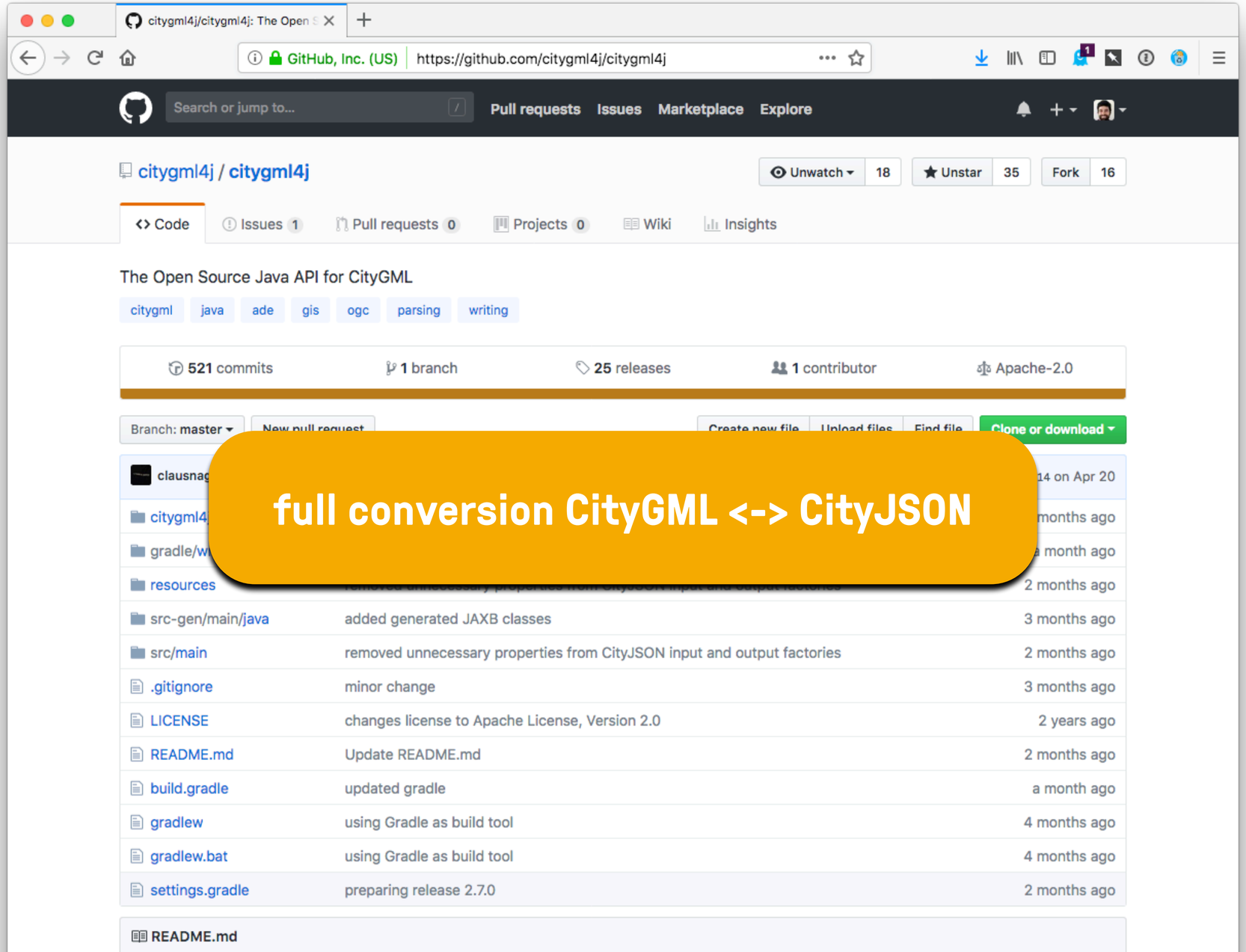
```
Hugos-MacBook-Pro:rotterdam hugo$ cjio
Usage: cjio [OPTIONS] INPUT COMMAND1 [ARGS]... [COMMAND2 [ARGS]...]...

  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)



pip install cjio

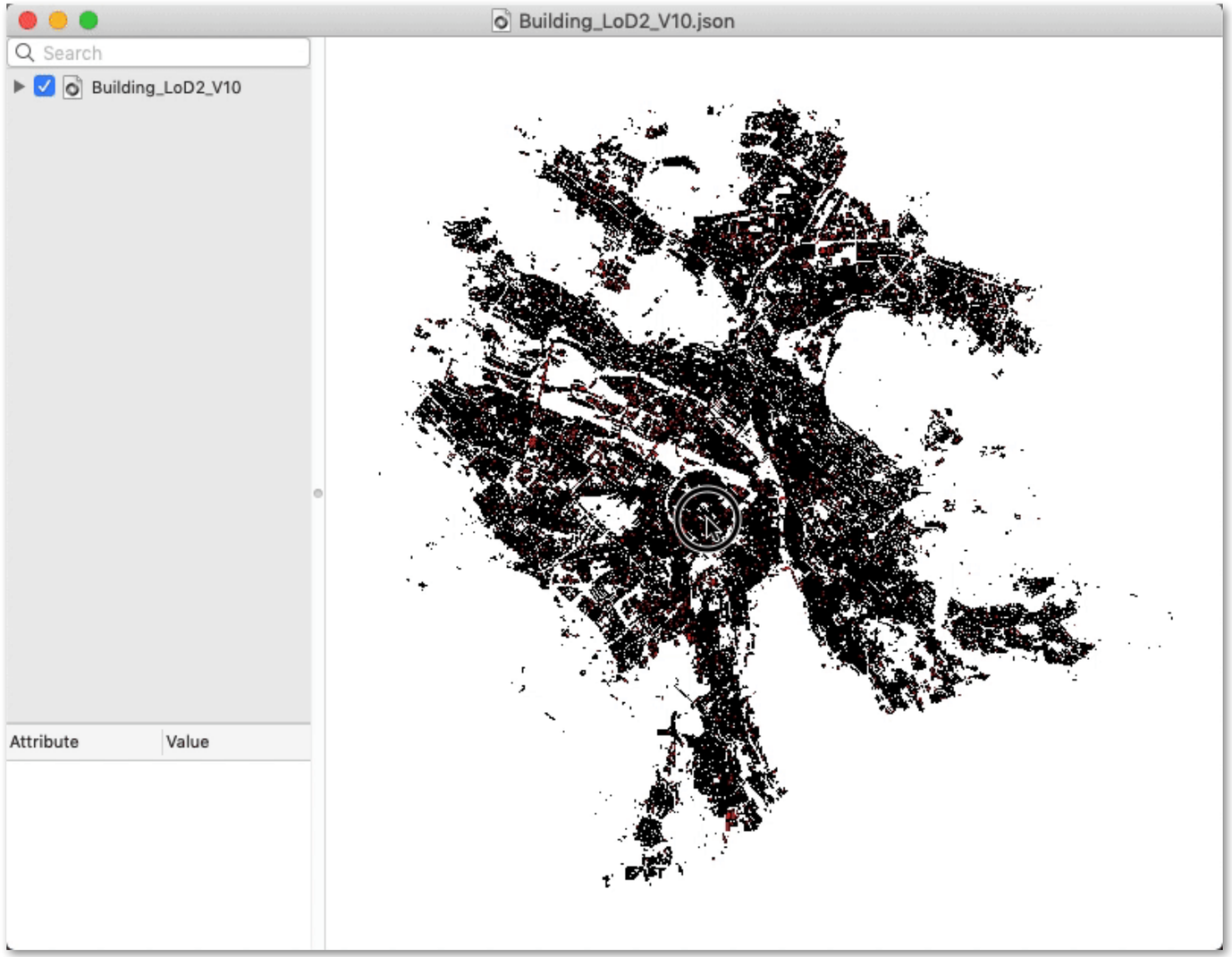
citygml4j


citygml4j/citygml4j: The Open
GitHub, Inc. (US)
https://github.com/citygml4j/citygml4j
Search or jump to...
Pull requests
Issues
Marketplace
Explore
citygml4j / citygml4j
Unwatch 18
Unstar 35
Fork 16
Code
Issues 1
Pull requests 0
Projects 0
Wiki
Insights
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
Branch: master
Clone or download
full conversion CityGML <-> CityJSON
resources
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

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

1 Download a simple file with 2 buildings
2 Visualise it
3 Manipulate and edit it with cjio
4 What else?
5 Questions and need help?

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.

twobuildings.city.json

```
{
    "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

1 Schema validation (is the syntax of the file OK?)
2 Geometry (are the geometric primitives valid?)

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:

CityJSON Validator

https://validator.cityjson.org

CityJSON

Drop a file to validate it

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

The file is 100% valid!

twobuildings.city.json