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 text-based
- It is easy for machines to parse and generate
- It is originally based on a subset of JavaScript (but is now in most languages)

```
{
  "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 (`a={}`)
   - C++ `std::map`
   - also called a “hash” or “hashmap”
2. arrays
   - Python lists (`a=[]`)
   - C++ `std::vector` / `std::array`

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
}
```

Key-value pair

“name”: “Hugo”

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

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

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

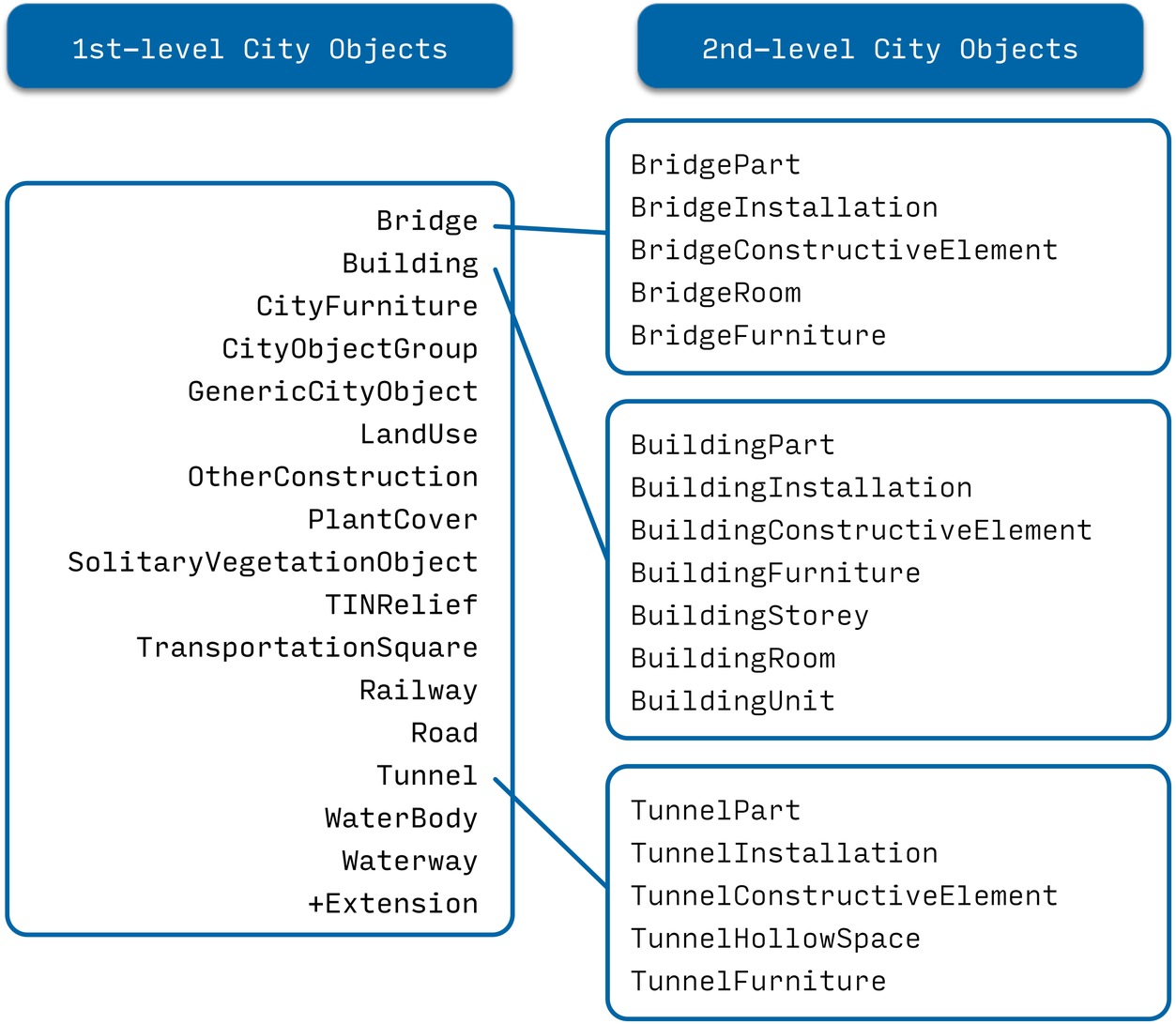
MSc Geomatics students when trying to parse a CityGML file in Python

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

version 1.1 = current version

2.0 **(since September 2023)**

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



Specifications give all the gory details

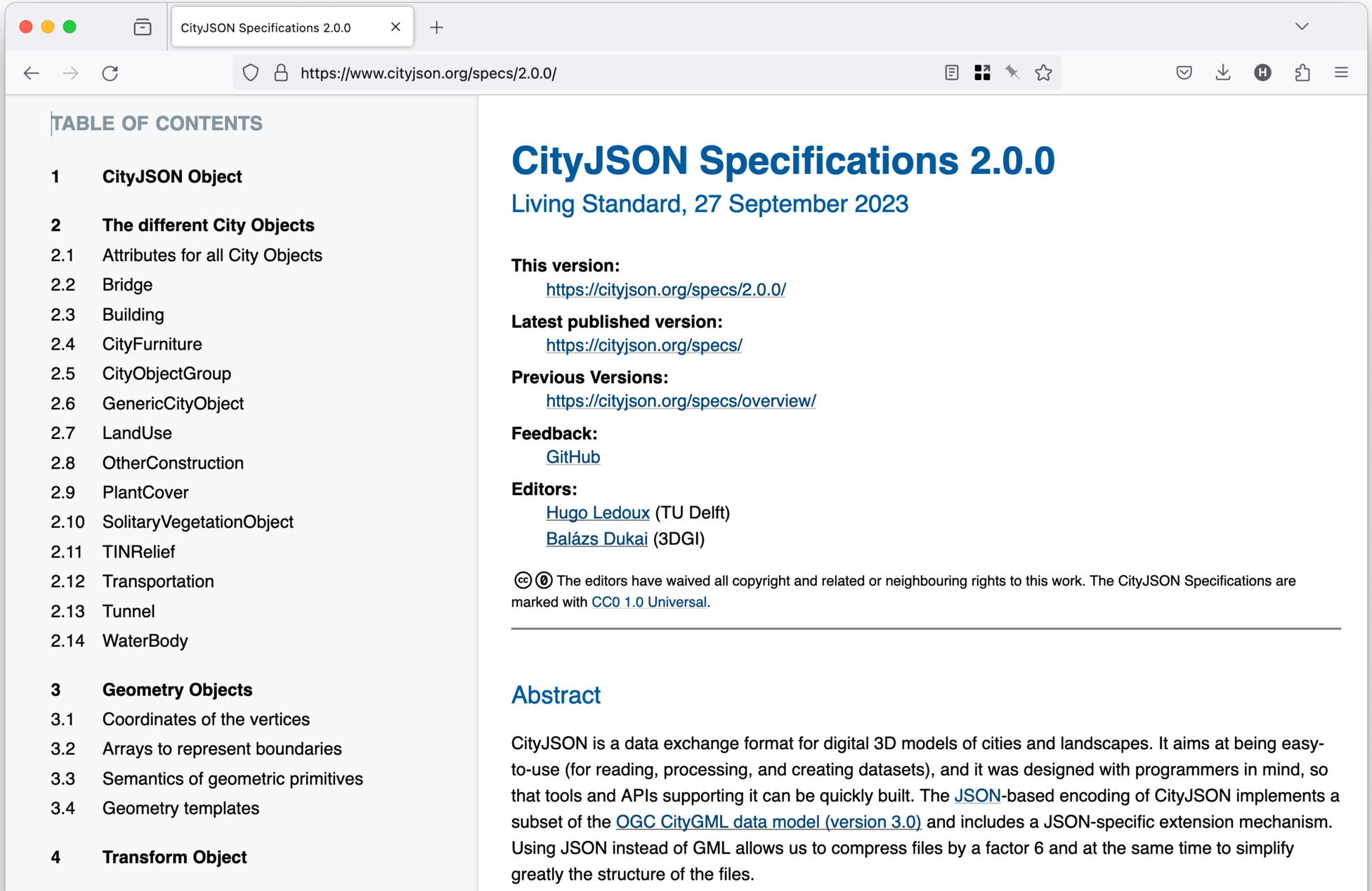


TABLE OF CONTENTS

CityJSON Specifications 2.0.0

Living Standard, 27 September 2023

This version:
https://cityjson.org/specs/2.0.0/

Latest published version:
https://cityjson.org/specs/

Previous Versions:
https://cityjson.org/specs/overview/

Feedback:
GitHub

Editors:
Hugo Ledoux (TU Delft)
Balázs Dukai (3DGI)

Abstract

CityJSON is a data exchange format for digital 3D models of cities and landscapes. It aims at being easy-to-use (for reading, processing, and creating datasets), and it was designed with programmers in mind, so that tools and APIs supporting it can be quickly built. The JSON-based encoding of CityJSON implements a subset of the OGC CityGML data model (version 3.0) and includes a JSON-specific extension mechanism. Using JSON instead of GML allows us to compress files by a factor 6 and at the same time to simplify greatly the structure of the files.

A CityJSON file

```
{
  "type": "CityJSON",
  "version": "2.0",
  "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": "2.0",
  "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": "2.0",
  "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": "2.0",
  "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": "2.0",
  "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": []
  }
}
```

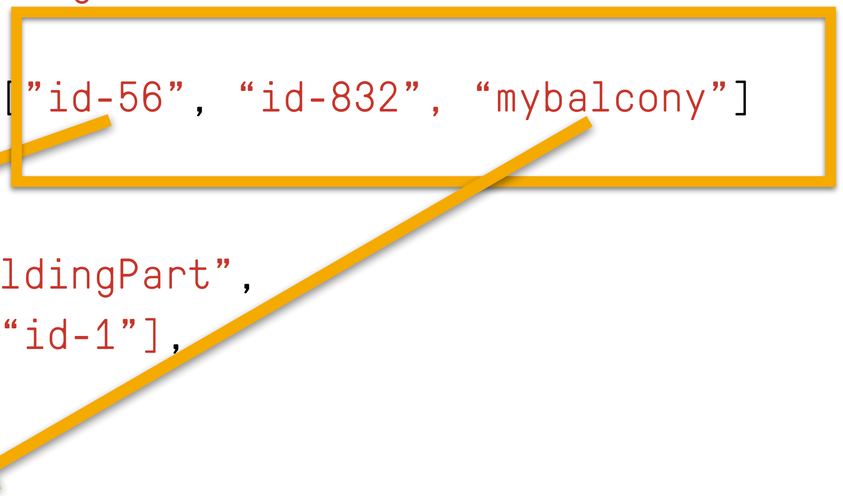
material + texture possible

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

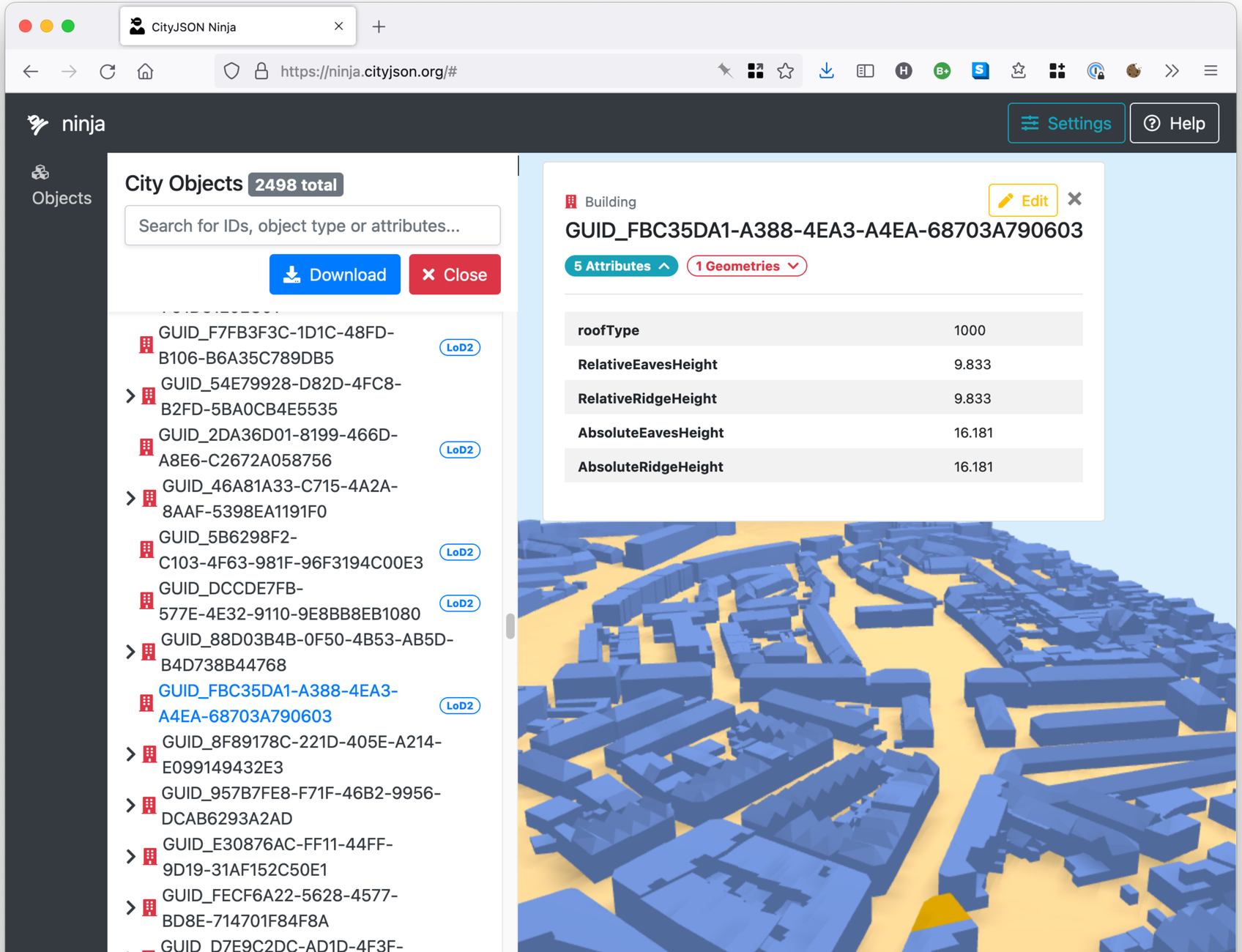
```
"CityObjects": {
  "id-1": {
    "type": "Building",
    "attributes": {
      "roofType": "gable"
    },
    "children": ["id-56", "id-832", "mybalcony"]
  },
  "id-56": {
    "type": "BuildingPart",
    "parents": ["id-1"],
    ...
  },
  "mybalcony": {
    "type": "BuildingInstallation",
    "parents":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



goal == a flat schema

CityJSON software

web-viewer: ninja.cityjson.org

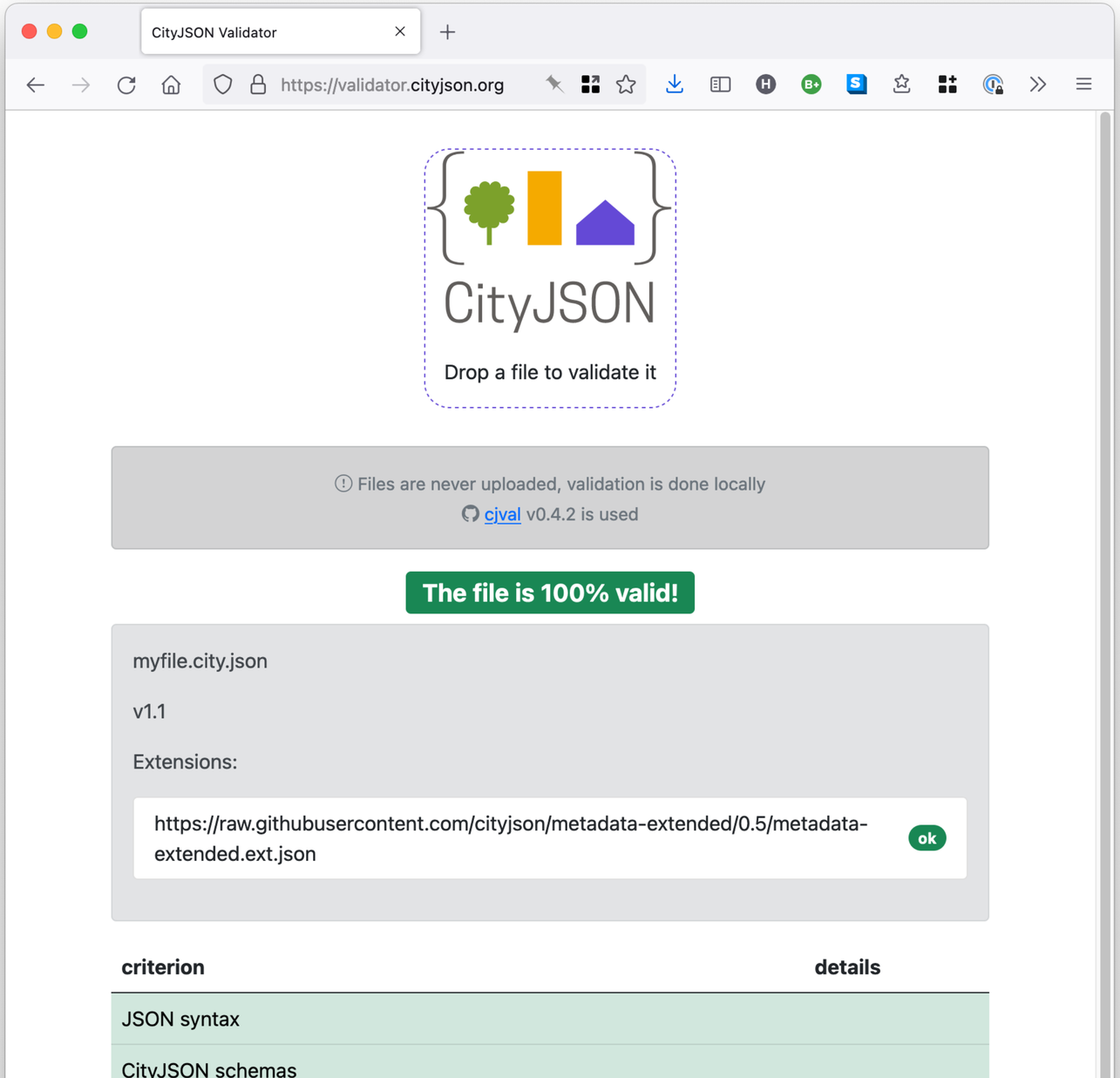


Nice fact: programmed mostly by 2 Geomatics students, as projects for GEO5010

You can do the same (or similar!)

Validation of the syntax of a file: cjval

A GEO5010 project from a student also kick-started this!




```
{
  "type": "CityJSON",
  "version": "2.0",
  "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": []
  }
}
```

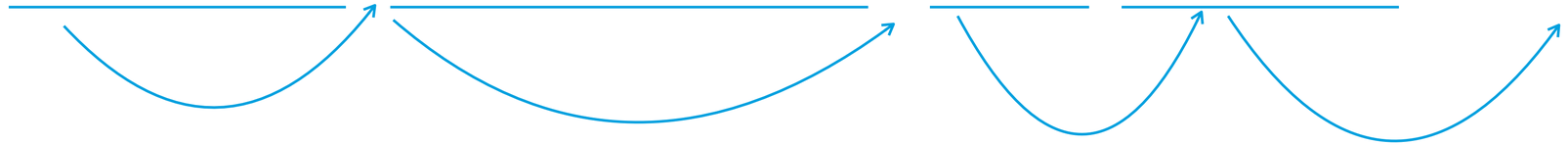
cjio (CityJSON/io) => pip install cjio

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...
```

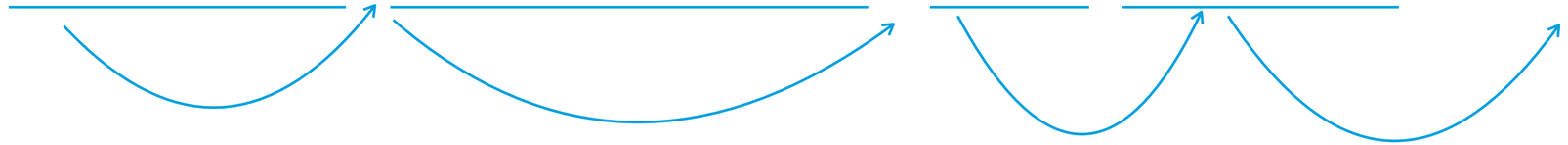
cjio (CityJSON/io)

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



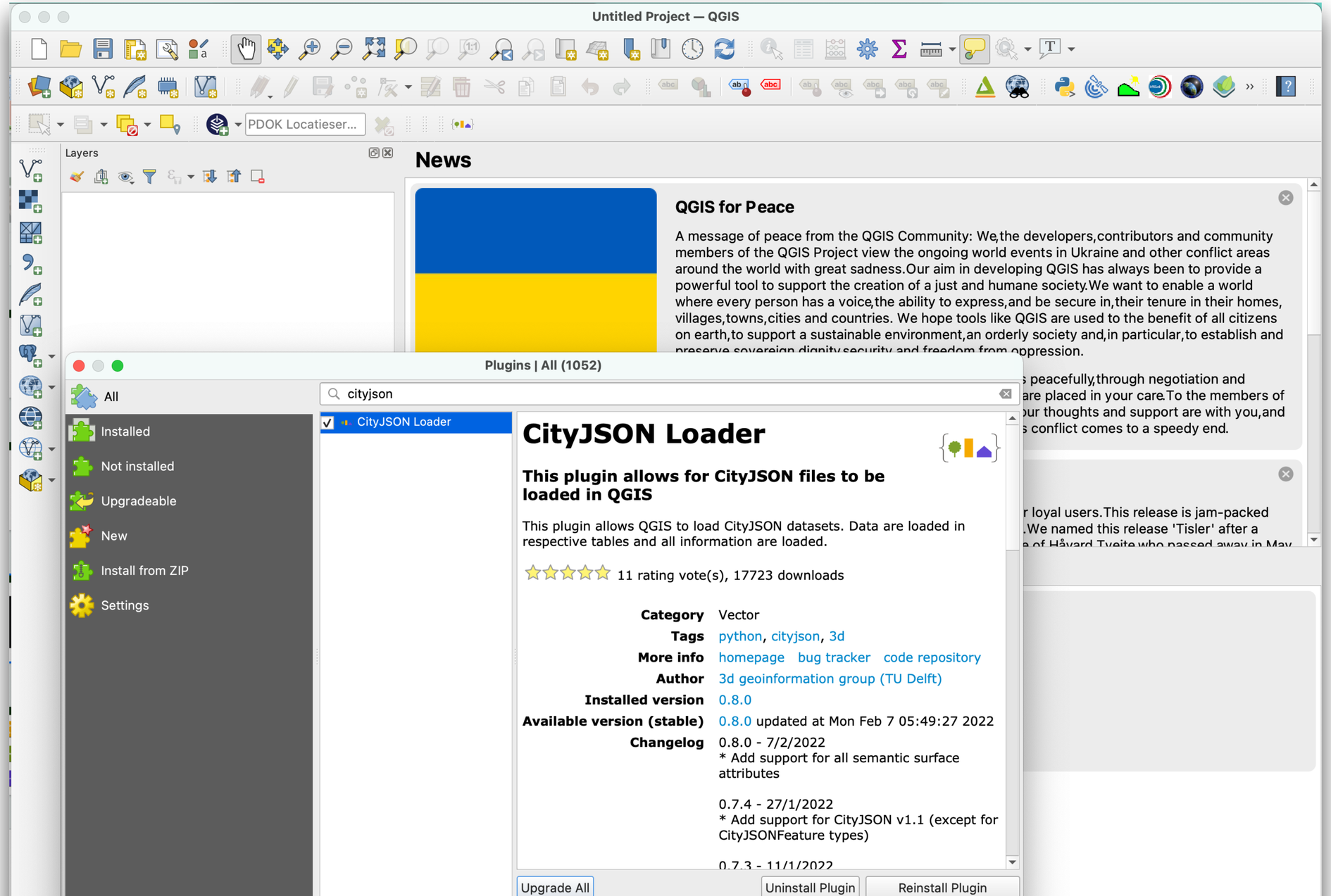
cjio (CityJSON/io)

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

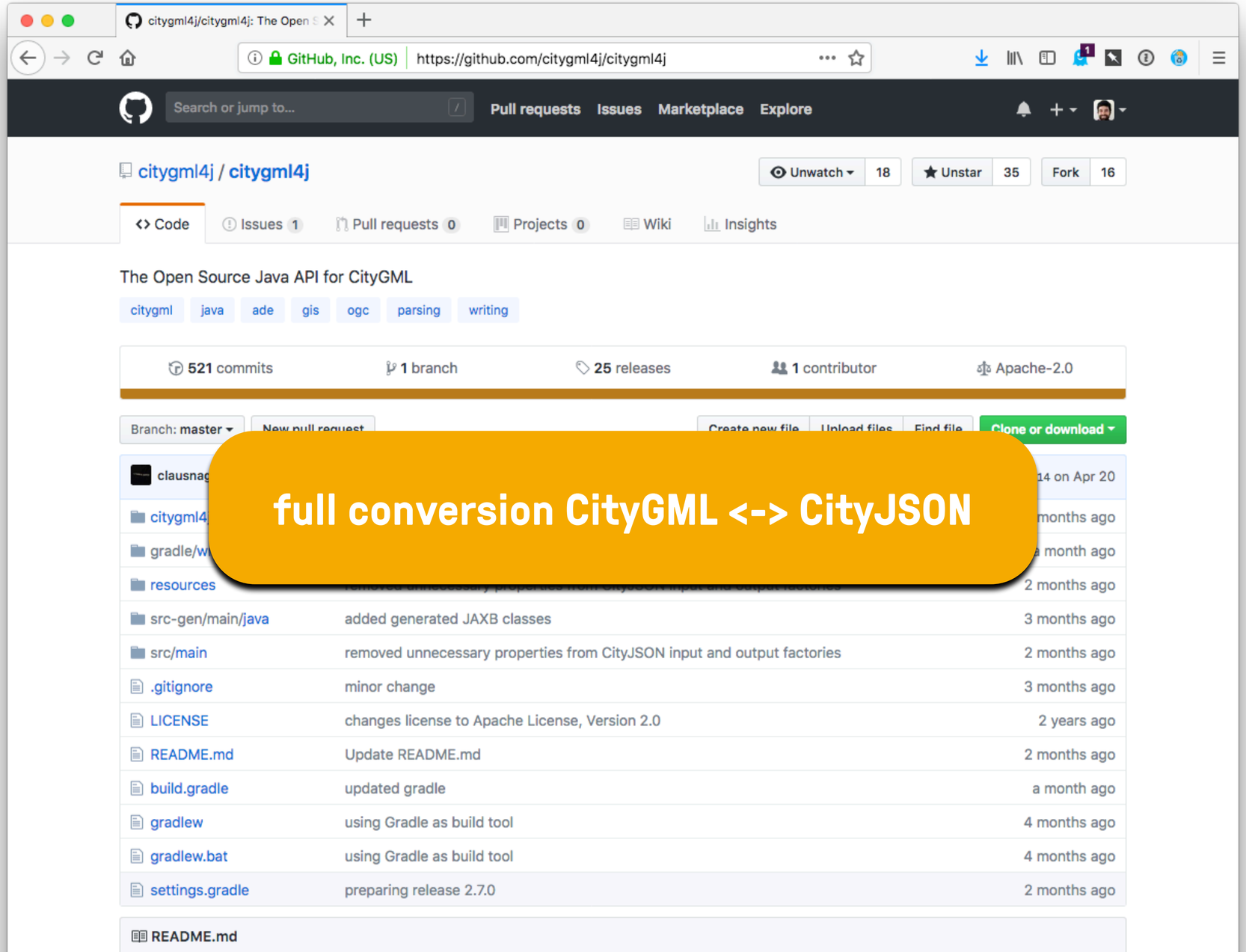


No need to save temp files between operators: pipeline is used

QGIS plugin


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

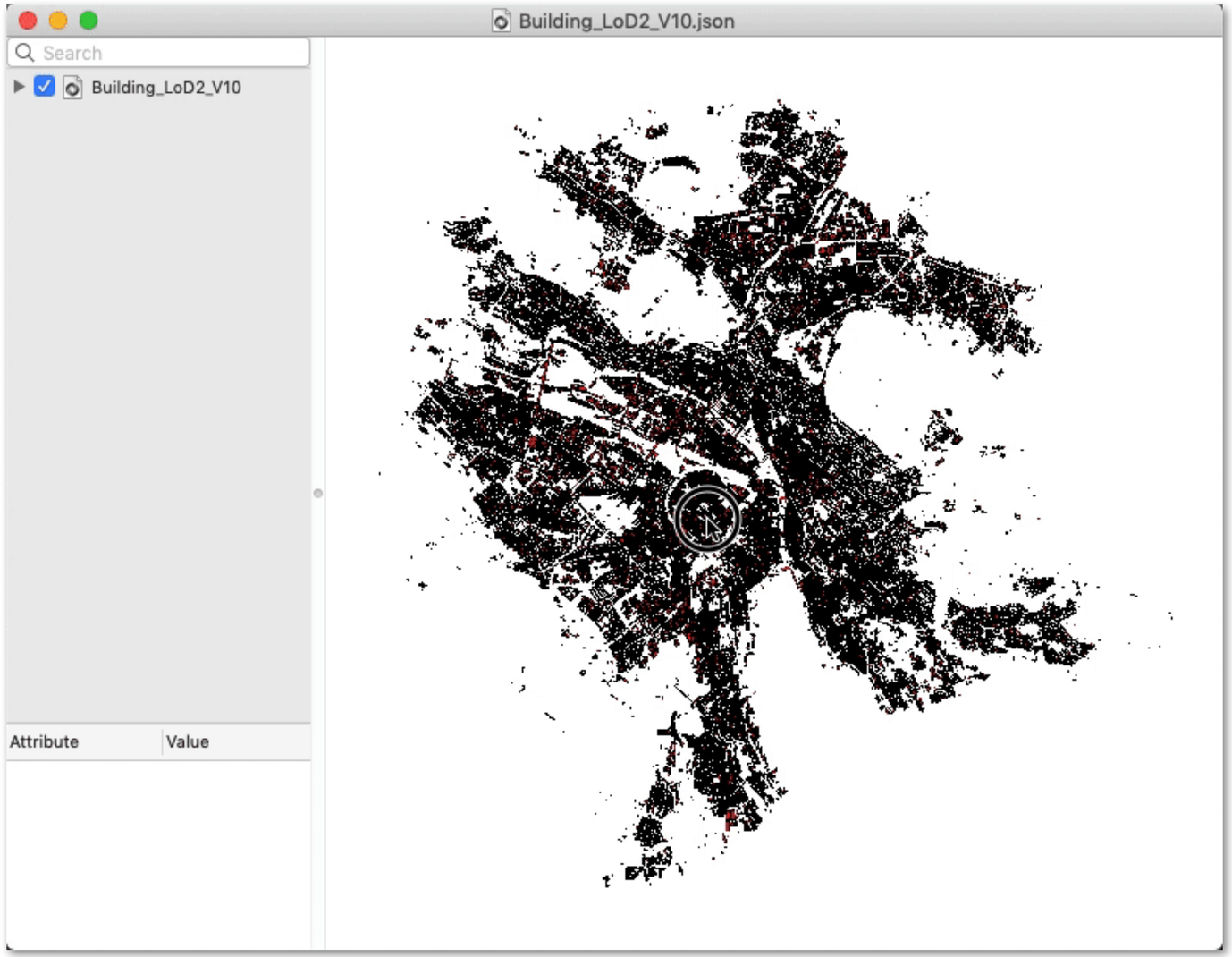
citygml4j


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



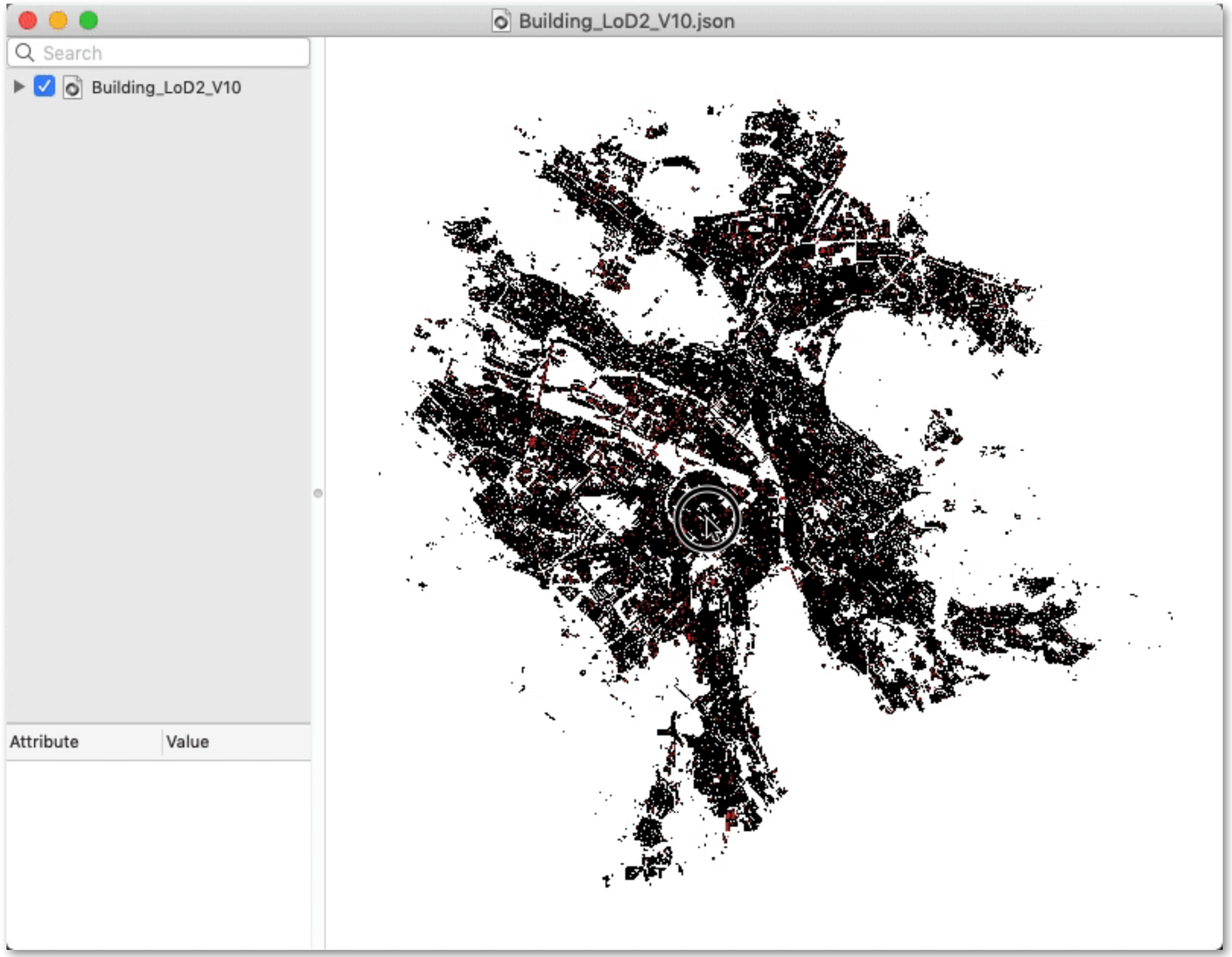
CityGML = 3.0GB

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

CityJSON = 292MB

Compression == 7.1X

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,
                                1
```

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.

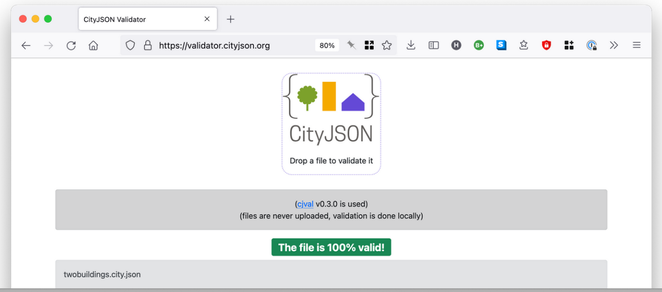
Schema validation (is the syntax of the file OK?)

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.

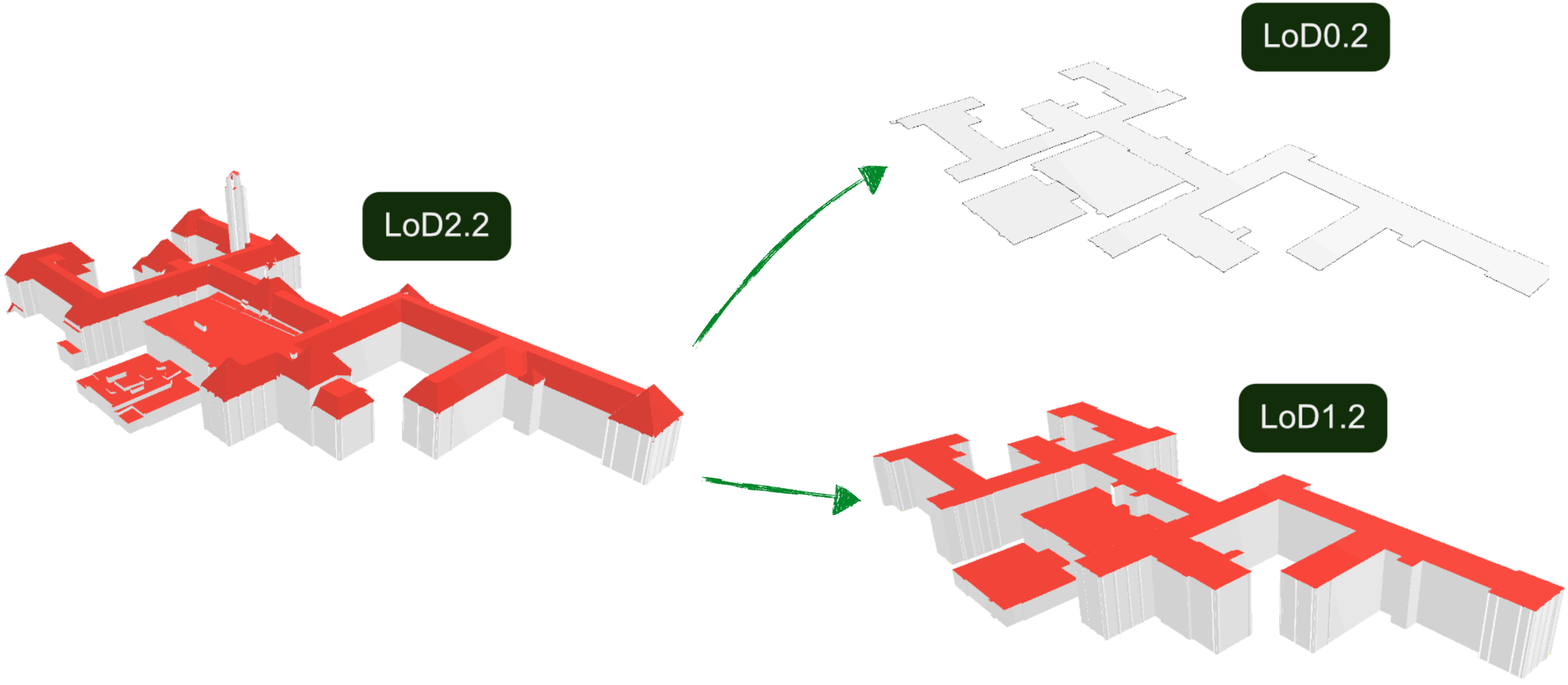
To validate a given file you can use any software listed here. However, it is rather tricky to stitch all the schemas together, and the handling of Extensions will not work.

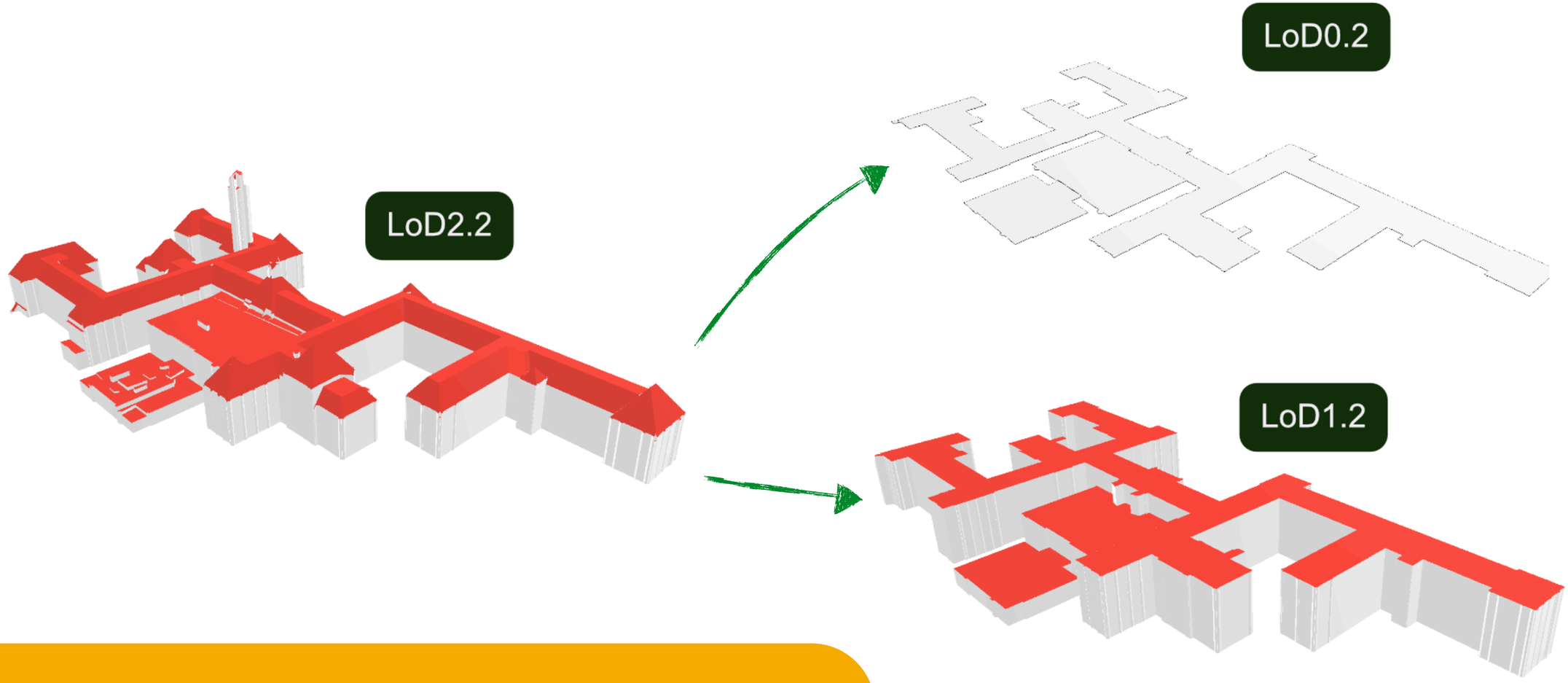
The "official validator" for CityJSON is cjval, which is available as a web-app and with cjio.

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



hw02

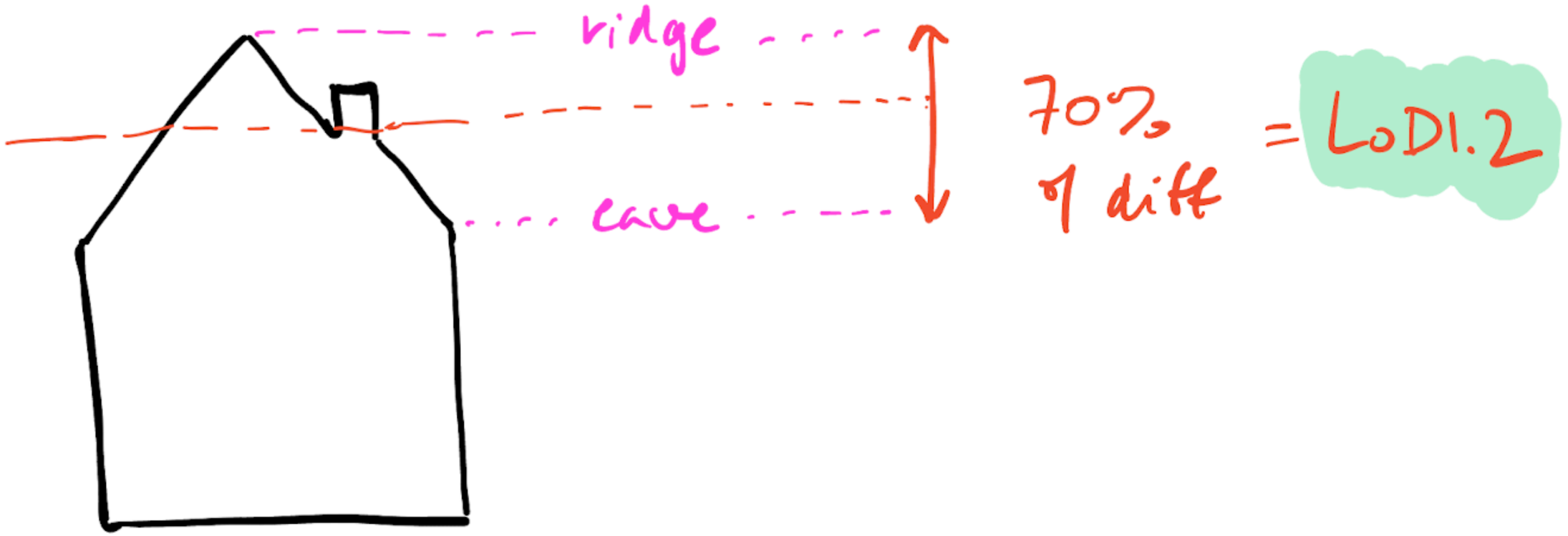

LoD0.2
LoD2.2
LoD1.2



The aim of this assignment is to generalise a 3D city model, from buildings in LoD2 you need to generate two extra LoDs: LoD1.2 and LoD0.2.

You will have to write a software that reads a CityJSON file that contains buildings in LoD2.x, and you will need to add the two LoDs and outputs a few file containing 3 LoDs for each building.

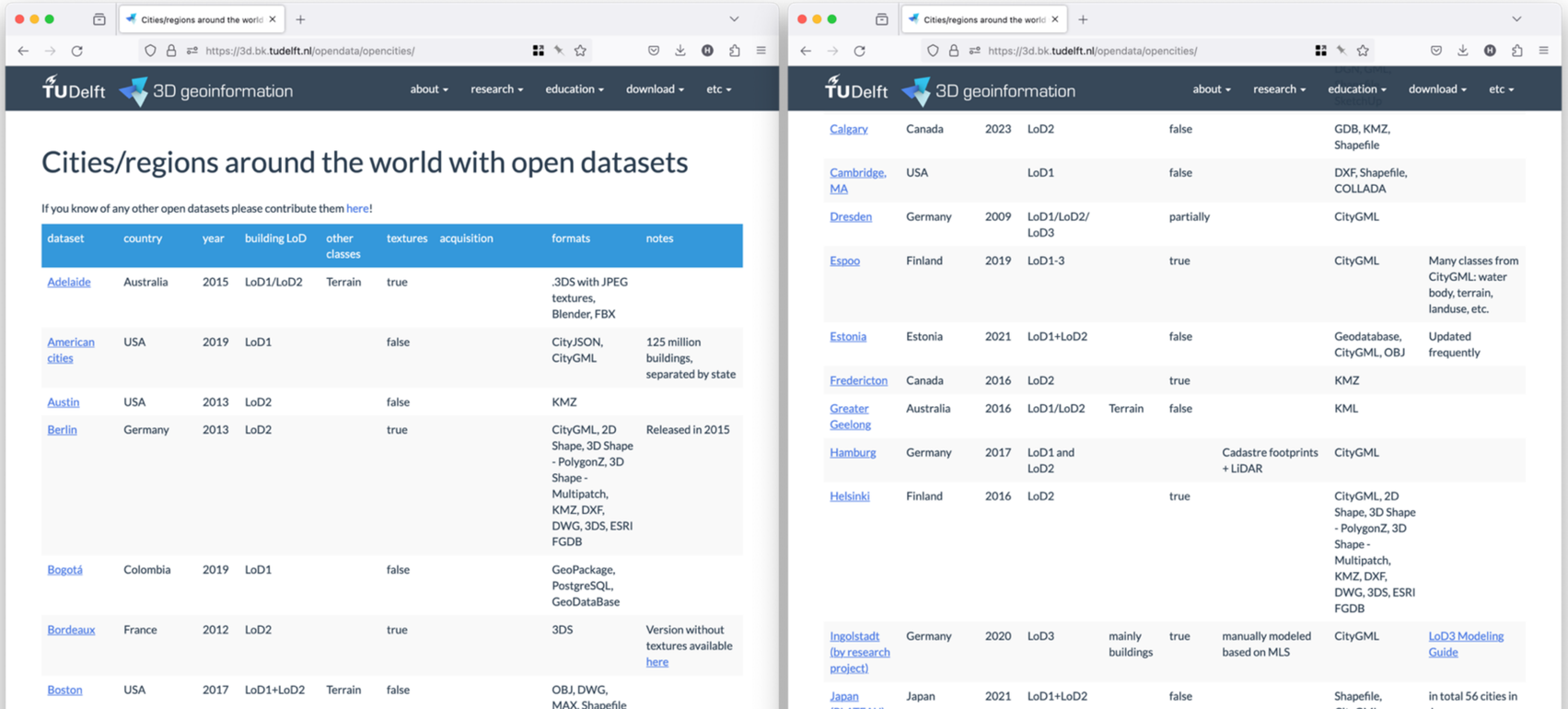
LoD1.2 specifications



Your C++ program should work with any CityJSON input file

- There are a few datasets in `/data`
- But browse the 3DBAG and others
- Convert from CityGML -> CityJSON if needed

Many cities around the world have 3D models openly available



Cities/regions around the world

https://3d.bk.tudelft.nl/opendata/opencities/

TUDelft 3D geoinformation — about · research · education · download · etc

Cities/regions around the world with open datasets

If you know of any other open datasets please contribute them here!

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, GeoDataBase	
Bordeaux	France	2012	LoD2		true		3DS	Version without textures available here
Boston	USA	2017	LoD1+LoD2	Terrain	false		OBJ, DWG, MAX, Shapefile	
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			Cadastre 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	Japan	2021	LoD1+LoD2		false		Shapefile,	in total 56 cities in