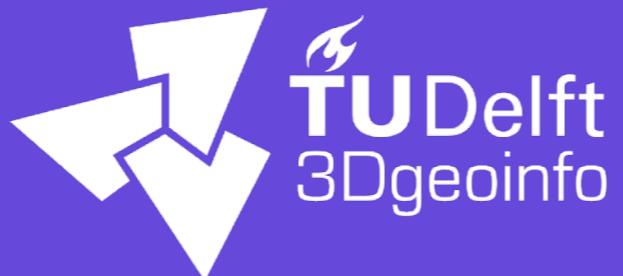




A compact and developer-friendly  
JSON-based encoding of the  
CityGML data model

Hugo Ledoux

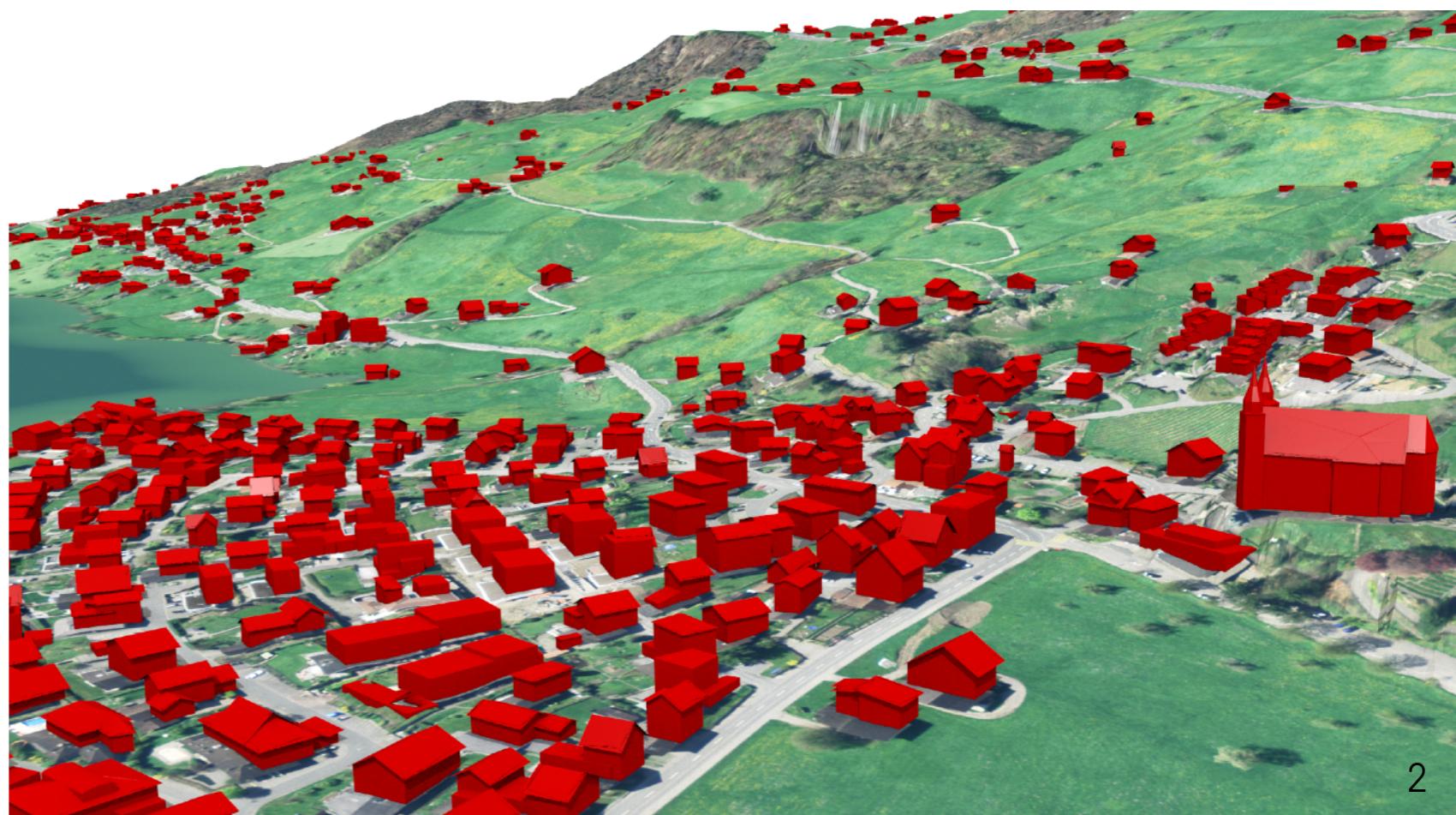
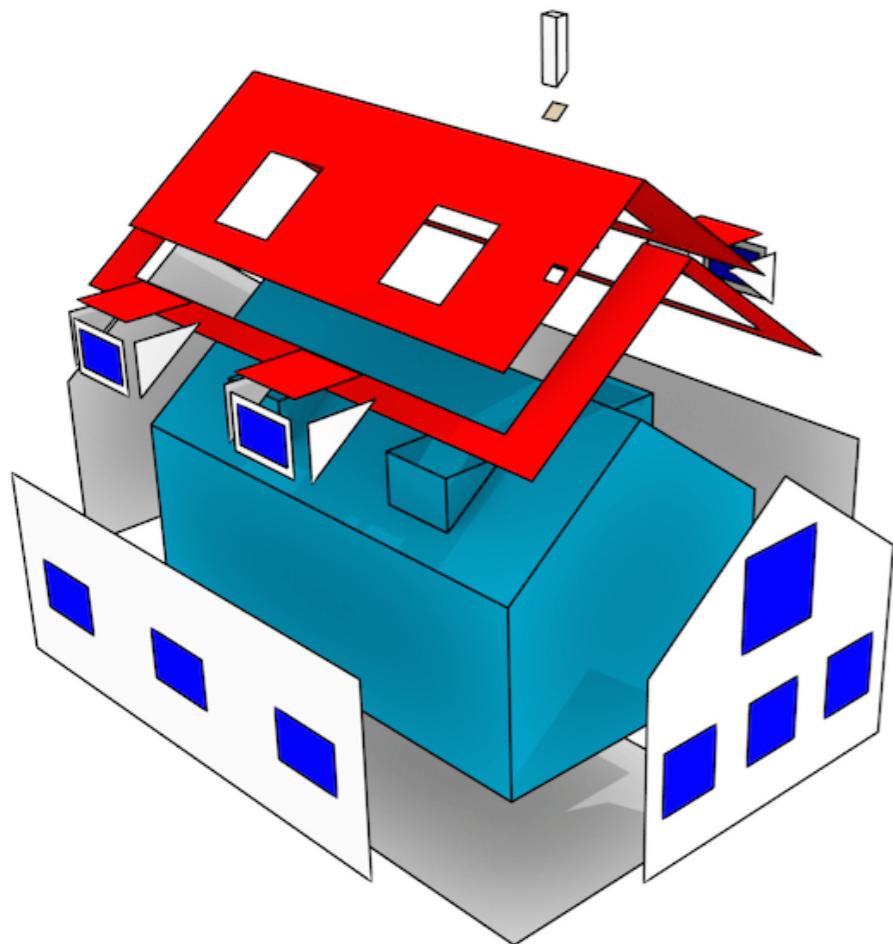
FOSS4GNL  
Delft, the Netherlands  
2019-06-20





# CityGML

international standard (from OGC) for representing  
and storing 3D city models



# CityGML files are very complex

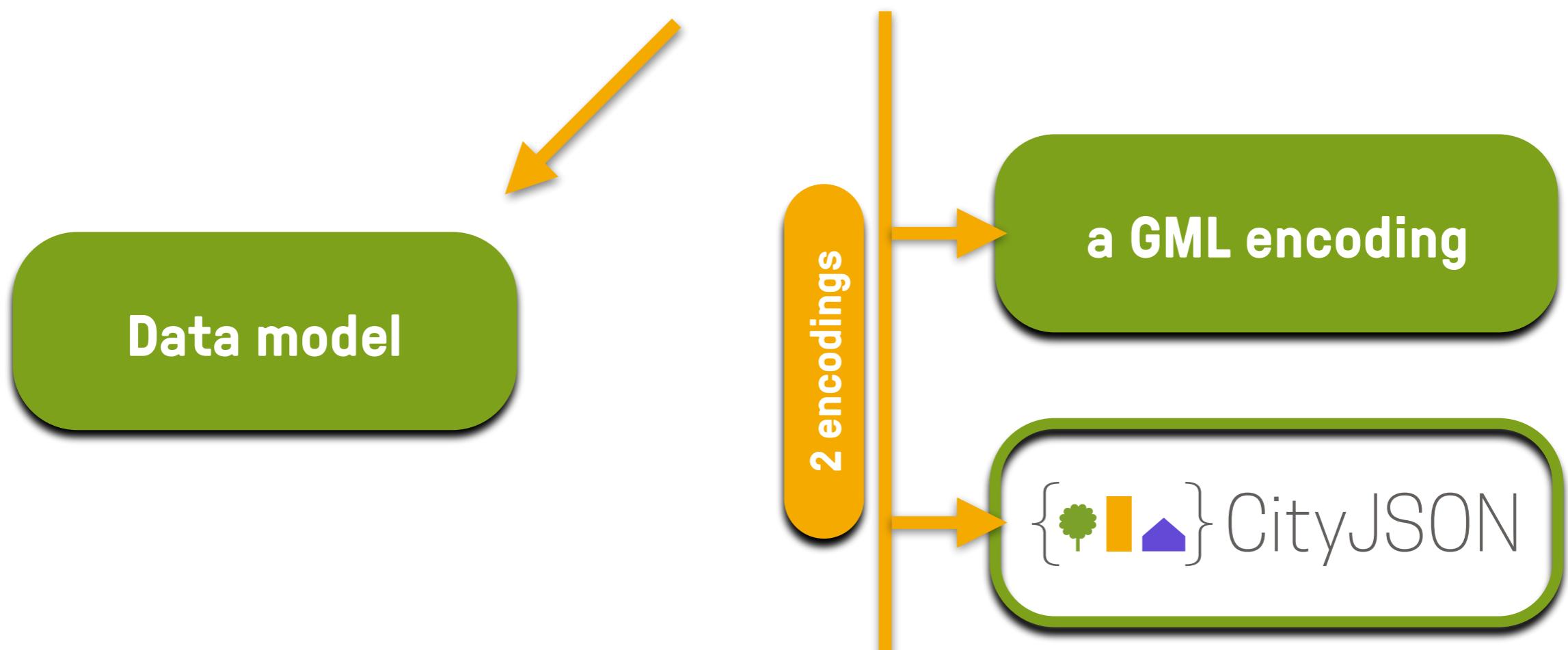
- files are deeply nested, and large
- many “points of entry”
- many diff ways to do one thing



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



# CityGML



# Why an alternative encoding? Read this.

Ledoux et al. *Open Geospatial Data, Software and Standards*  
<https://doi.org/10.1186/s40965-019-0064-0>

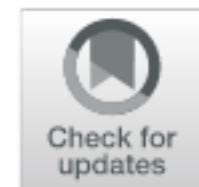
(2019) 4:4

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 present the different software packages to parse/view/edit/create files (including one to automatically convert between the JSON and GML encodings), analyse how real-world datasets compare to those of CityGML, and we also introduce *Extensions*, which allow us to extend the core data model in a documented manner.

**Keywords:** 3D city modelling, CityGML, Data modelling

# CityJSON

Home - CityJSON    +

https://www.cityjson.org

CityJSON    Search CityJSON

Home  
What is CityJSON?  
News  
Specifications  
Schemas  
Datasets  
Extensions  
Software  
Tutorials

{} CityJSON

A compact and developer-friendly JSON-based encoding of the CityGML data model

[Getting started](#)    [Specifications \(v1.0.0\)](#)

---

CityJSON is a [JSON-based](#) encoding for a subset of the [OGC CityGML](#) data model (version 2.0.0), which is an open standardised data model and exchange format (in [GML](#)) to store digital 3D models of cities and landscapes.

The aim of CityJSON is to offer an alternative to the GML encoding of CityGML, which can be verbose and complex (and thus rather frustrating to work with). CityJSON aims at being easy-to-use, both for reading datasets, and for creating them. It was designed with programmers in mind, so that tools and APIs supporting it can be quickly built, and [several](#) have been created already.

We believe that you should use CityJSON because:

- 1 its simplicity means that it is already supported by [several software](#)
- 2 you can in one-click convert CityGML files to CityJSON files, and vice versa, with the open-source tool [citygml-tools](#); we even have a [tutorial](#)
- 3 files are on average [6X more compact](#) than their CityGML equivalent
- 4 there is a [web-viewer](#) where you can drag-and-drop a file
- 5 you can easily manipulate files with [cijo](#), you can for instance merge files, remove/filter objects,

# version 1.0.0 released recently

- all modules mapped 💪
- software for full conversion  
CityGML <-> CityJSON
- several software already
- Extensions (ADEs) are possible

1. "Building"
2. "BuildingPart"
3. "BuildingInstallation"
4. "Road"
5. "Railway"
6. "TransportSquare"
7. "TINRelief"
8. "WaterBody"
9. "PlantCover"
10. "SolitaryVegetationObject"
11. "LandUse"
12. "CityFurniture"
13. "GenericCityObject"
14. "Bridge"
15. "BridgePart"
16. "BridgeInstallation"
17. "BridgeConstructionElement"
18. "Tunnel"
19. "TunnelPart"
20. "TunnelInstallation"
21. "CityObjectGroup"

# Same information as CityGML, but in JSON format

```
{  
  "type": "CityJSON",  
  "version": "1.0",  
  "metadata": {  
    "referenceSystem": "urn:ogc:def:crs:EPSG::7415",  
  },  
  "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": [  
      [23.1, 2321.2, 11.0],  
      [111.1, 321.1, 12.0],  
      ...  
    ],  
    "appearance": {  
      "materials": [],  
      "textures": [],  
      "vertices-texture": []  
    }  
  }  
}
```

# A CityJSON file

```
{  
  "type": "CityJSON",  
  "version": "1.0",  
  "metadata": {  
    "referenceSystem": "urn:ogc:def:crs:EPSG::7415",  
  },  
  "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": [  
      [23.1, 2321.2, 11.0],  
      [111.1, 321.1, 12.0],  
      ...  
    ],  
    "appearance": {  
      "materials": [],  
      "textures": [],  
      "vertices-texture": []  
    }  
  }  
}
```

version CityJSON

# A CityJSON file

```
{  
  "type": "CityJSON",  
  "version": "1.0",  
  "metadata": {  
    "referenceSystem": "urn:ogc:def:crs:EPSG::7415",  
  },  
  "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": [  
      [23.1, 2321.2, 11.0],  
      [111.1, 321.1, 12.0],  
      ...  
    ],  
    "appearance": {  
      "materials": [],  
      "textures": [],  
      "vertices-texture": []  
    }  
  }  
}
```

metadata, ISO19115 “compliant”

CityGML has no mechanism in v2.0  
but we thought it's important

ALL geometries have the same CRS,  
unlike CityGML

# A CityJSON file

```
{  
  "type": "CityJSON",  
  "version": "1.0",  
  "metadata": {  
    "referenceSystem": "urn:ogc:def:crs:EPSG::7415",  
  },  
  "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": [  
      [23.1, 2321.2, 11.0],  
      [111.1, 321.1, 12.0],  
      ...  
    ],  
    "appearance": {  
      "materials": [],  
      "textures": [],  
      "vertices-texture": []  
    }  
  }
```

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

Each have geometries + attributes

# A CityJSON file

```
{  
  "type": "CityJSON",  
  "version": "1.0",  
  "metadata": {  
    "referenceSystem": "urn:ogc:def:crs:EPSG::7415",  
  },  
  "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": [  
      [23.1, 2321.2, 11.0],  
      [111.1, 321.1, 12.0],  
      ...  
    ],  
    "appearance": {  
      "materials": [],  
      "textures": [],  
      "vertices-texture": []  
    }  
  }
```

Geometry is ID of the vertex, global list

compression + more “topology”

# A CityJSON file

```
{  
  "type": "CityJSON",  
  "version": "1.0",  
  "metadata": {  
    "referenceSystem": "urn:ogc:def:crs:EPSG::7415",  
  },  
  "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": [  
      [23.1, 2321.2, 11.0],  
      [111.1, 321.1, 12.0],  
      ...  
    ],  
    "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

The screenshot shows a web browser window titled "CityJSON Specifications 1.0.0". The URL in the address bar is <https://www.cityjson.org/specs/1.0.0/#attributes>. The left sidebar contains a "TABLE OF CONTENTS" with sections 1 through 7. Section 2.1, "Attributes", is currently selected and highlighted with a blue underline. The main content area is titled "§ 2.1. Attributes". It contains text about attributes prescribed by CityGML and lists three possible attributes: "class", "function", and "usage". It also mentions a codelist maintained by SIG 3D. A code snippet shows a JSON object for "CityObjects" with two entries, "id-1" and "id-2", each representing a different type of City Object (LandUse and WaterBody) with their respective attributes and geometries.

**TABLE OF CONTENTS**

- 1 CityJSON Object
- 2 City Object types
  - 2.1 Attributes
  - 2.2 Building
  - 2.3 Transportation
  - 2.4 TINRelief
  - 2.5 WaterBody
  - 2.6 LandUse
  - 2.7 PlantCover
  - 2.8 SolitaryVegetationObject
  - 2.9 CityFurniture
  - 2.10 GenericCityObject
  - 2.11 Bridge
  - 2.12 Tunnel
  - 2.13 CityObjectGroup
- 3 Geometry Objects
  - 3.1 The coordinates of the vertices
  - 3.2 Arrays to represent boundaries
  - 3.3 Semantic Surface Object
- 4 Metadata
  - 4.1 CRS
  - 4.2 Geographic Extent (bbox)
  - 4.3 Geographic location
  - 4.4 Topic Category
  - 4.5 Lineage
- 5 Transform Object
- 6 Geometry templates
- 7 Appearance Object
  - 7.1 Geometry Object having material(s)

**§ 2.1. Attributes**

The attributes prescribed by CityGML differ per City Object, and can be seen either in the [official CityGML documentation](#) or in the schemas of CityJSON.

In CityJSON, any other attributes not prescribed by the CityGML data model can be added with a JSON key-value pair ("owner" in the example above is one such attribute) in the "attributes" of a City Object.

All the City Objects have the following 3 possible attributes:

1. "class"
2. "function"
3. "usage"

While CityGML does not prescribe the values for these, the [SIG 3D maintains a codelist](#) that can be used. In CityJSON, as can be seen in the schemas, the values should be a string, thus either the name of the values should be used, or the code as a string:

```
"CityObjects": {  
    "id-1": {  
        "type": "LandUse",  
        "attributes": {  
            "function": "Industry and Business"  
        },  
        "geometry": [...]  
    },  
    "id-2": {  
        "type": "WaterBody",  
        "attributes": {  
            "class": "1010"  
        },  
        "geometry": [...]  
    }  
}
```

**§ 2.2. Building**

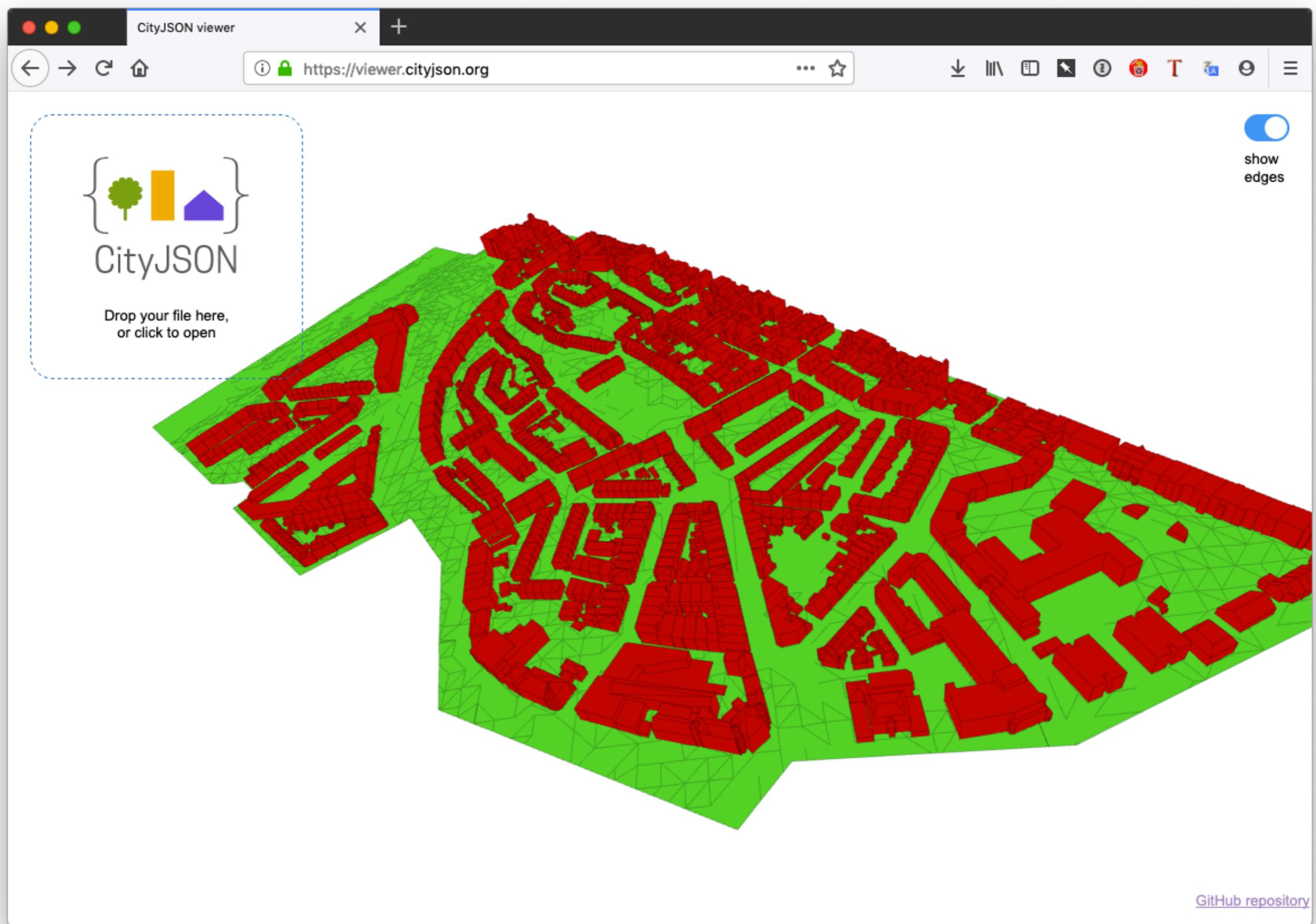
Three City Objects are related to buildings: "Building", "BuildingPart", and "BuildingInstallation".

The geometry of both "Building" and "BuildingPart" can only be represented with these Geometry Objects:  
(1) "Solid", (2) "CompositeSolid", (3) "MultiSurface".

The geometry of a "BuildingInstallation" object can be represented with any of the Geometry Objects.

**CityJSON software?  
We have that.**

# web-viewer



Featured Top Charts Categories Purchased Updates

Search

Azul 4+

azul is a 3D viewer for 3D city models in (City)GML, CityJSON, OBJ, OFF and POLY. It supports loading multiple files, selecting objects by clicking them or selecting them in the sidebar, and browsing their attributes.

What's New in Version 0.8.1

Support for CityJSON 0.5

Install ▾

Azul Support >

Information

Category: Utilities  
Updated: 19 December 2017  
Version: 0.8.1  
Price: Free  
Size: 9.6 MB  
Family Sharing: Yes  
Language: English  
Developer: Ken Arroyo Ohori  
© 2016-2017 Ken Arroyo Ohori

Rated 4+  
Compatibility:  
macOS 10.13 or later, 64-bit processor



# 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

Branch: master New pull request Create new file Upload files Find file Clone or download

clausnagel · 14 on Apr 20 · 1 month ago · full conversion CityGML <-> CityJSON

citygml4j · removed unnecessary properties from CityJSON input and output factories · 2 months ago

gradle/wrapper · added generated JAXB classes · 3 months ago

resources · removed unnecessary properties from CityJSON input and output factories · 2 months ago

src-gen/main/java · minor change · 3 months ago

src/main · changes license to Apache License, Version 2.0 · 2 years ago

.gitignore · Update README.md · 2 months ago

LICENSE · updated gradle · a month ago

README.md · using Gradle as build tool · 4 months ago

build.gradle · using Gradle as build tool · 4 months ago

gradlew · preparing release 2.7.0 · 2 months ago

gradlew.bat

settings.gradle

README.md

citygml4j / citygml4j

Pull requests Issues Marketplace Explore

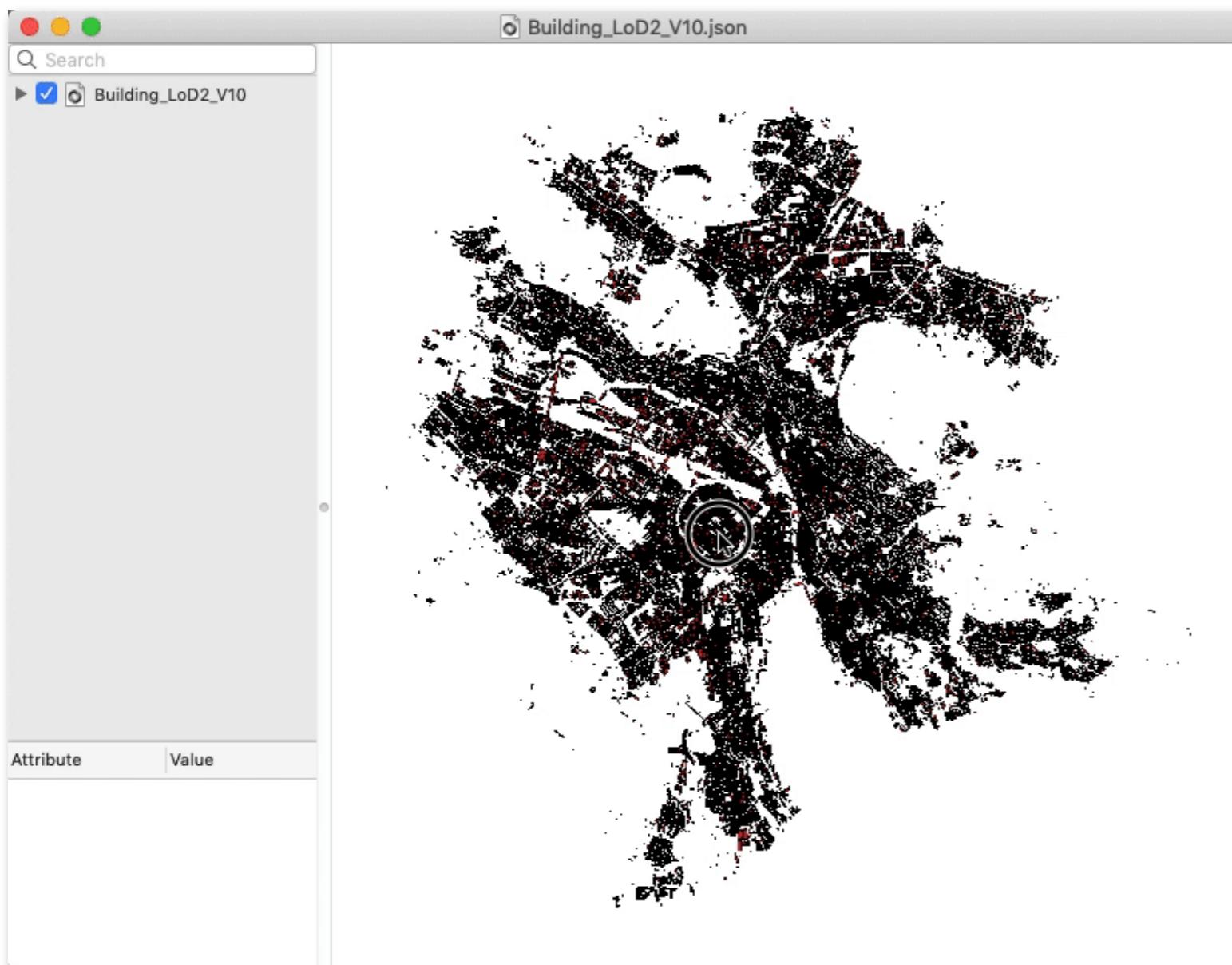
Unwatch 18 Unstar 35 Fork 16

Search or jump to...

# 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

# 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
```

# Python API (work-in-progress)

The screenshot shows a web browser window with the title "cpio API tutorial — cpio 0.5.2 documentation". The URL in the address bar is [https://tudelft3d.github.io/cpio/cpio\\_tutorial.html](https://tudelft3d.github.io/cpio/cpio_tutorial.html). The page content is the "cpio API tutorial".

**Left sidebar:**

- Search docs
- cpio, or CityJSON/io
  - Load the city model
  - Using the CLI commands in the API
  - Explore the city model
  - Working with the objects in the model
  - Save or Export
- Reference

**Main content area:**

## cpio API tutorial

In this tutorial we explore what is possible with `cpio`'s API. I refer to `cpio`'s command line interface as *CLI*.

The CLI is what you use when you invoke `cpio` from the command line, such as:

```
$ cpio some_city_model.json validate
```

The API is what you use from `cpio` when working with a city model in a Python script.

You can play with the executable version of this notebook on Binder [here](#)

```
[1]: import os  
from copy import deepcopy  
from cpio import cityjson
```

Set up the paths for the tutorial.

```
[6]: package_dir = os.path.dirname('.../...')  
schema_dir = os.path.join(package_dir, 'cpio', 'schemas', '1.0.0')  
data_dir = os.path.join(package_dir, 'example_data')
```

## Load the city model

# 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.

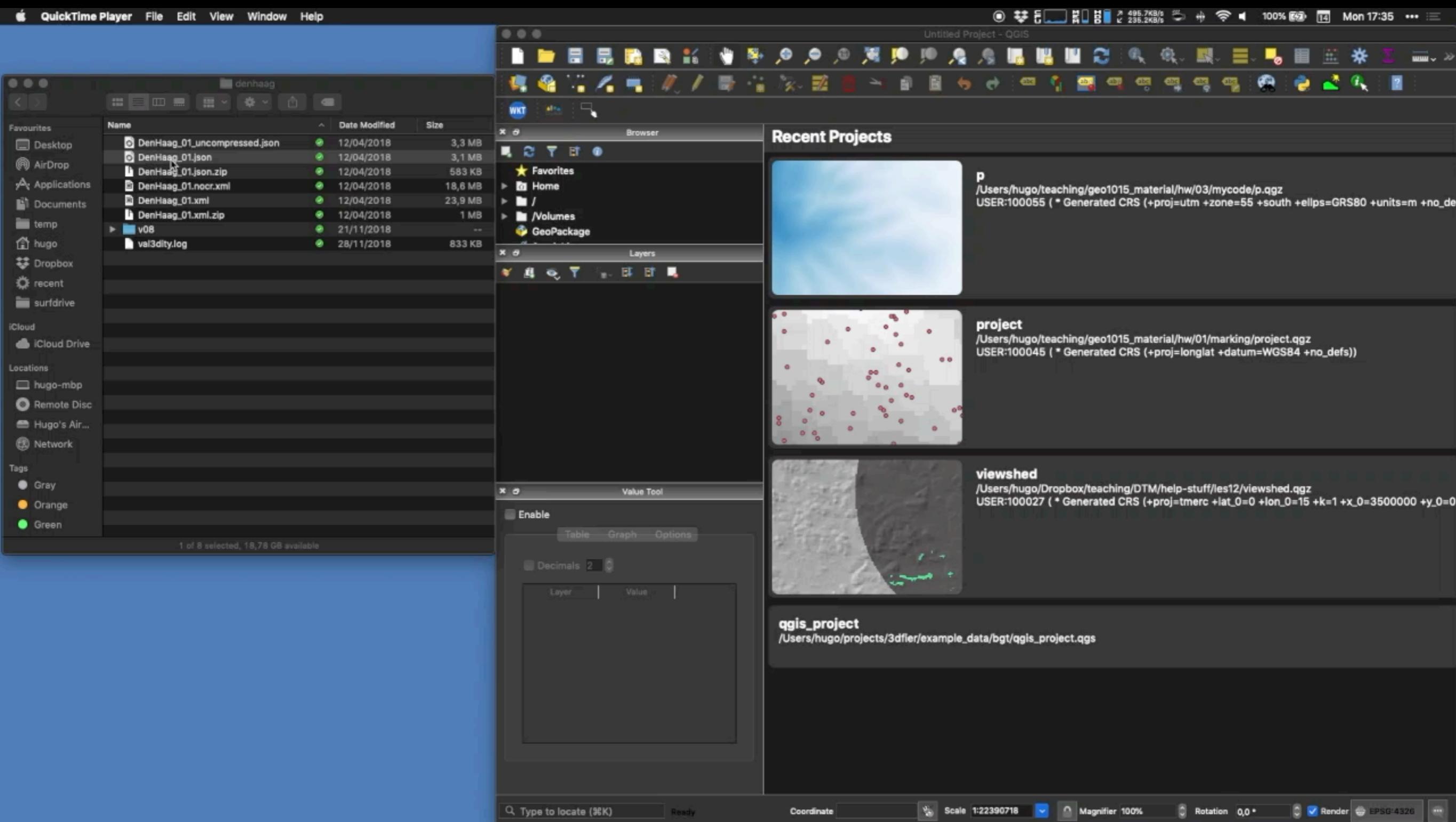
Commands:
  compress            Compress a CityJSON file, ie stores its...
  decompress          Decompress a CityJSON file, ie remove the...
  info                Output info in simple JSON.
  merge               Merge the current CityJSON with others.
  remove_duplicate_vertices Remove duplicate vertices 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)

```
$ cjio myfile.json assign_epsg 7415 subset -cotype Building compress save out.json
```



**pip install cjio**



# Help? Feedback? All development is open

Issues · tudelft3d/cityjson

GitHub, Inc. (US) https://github.com/tudelft3d/cityjson/issues

Search or jump to... Pull requests Issues Marketplace Explore

tudelft3d / cityjson

Code Issues 13 Pull requests 1 Actions Projects 0 Wiki Security Insights Settings

Label issues and pull requests for new contributors

Now, GitHub will help potential first-time contributors discover issues labeled with [help wanted](#) or [good first issue](#)

Go to Labels

Filters is:issue is:open Labels 7 Milestones 0 New issue

13 Open 29 Closed

Author Labels Projects Milestones Assignee Sort

- Clarify how to nest children of CityObjects question #49 opened on 7 May by balazsdukai 3 comments
- CityObjects in sorted order #47 opened on 30 Apr by clausnagel 6 comments
- offer a "minified" schema enhancement #46 opened on 16 Apr by hugoledoux 2 comments
- Parent Property Energy Extension #45 opened on 16 Apr by hugoledoux 2 comments
- Why using an indexed scheme instead of a "feature-approach"? question #20 opened on 3 Apr 2018 by hugoledoux 1 comment

it's not an OGC standard, and there are no concrete plans for it to become one

# Getting started?

The screenshot shows a web browser window with the title "Getting started with CityJSON - City". The URL in the address bar is <https://www.cityjson.org/tutorials/getting-started/>. The page content is titled "Getting started with CityJSON". On the left, there is a sidebar with links to "Home", "What is CityJSON?", "News", "Specifications", "Schemas", "Datasets", "Extensions", "Software", "Tutorials" (which is currently selected), and a list under "- Getting started with CityJSON" including "Converting to/from CityGML files", "Validation of a CityJSON file", "Mapping the CityGML Noise ADE to a CityJSON Extension", and "Modelling guide for CityJSON". The main content area has a search bar and a "TABLE OF CONTENTS" section with four items: "Download a simple file with 2 buildings", "Visualise it", "Manipulate and edit it with cjo", and "What else?". Below this is a section titled "Download a simple file with 2 buildings" with a link to "twobuilding.json". A note says you can open the file in a text editor to see its structure and edit it. At the bottom, there is a code editor window showing the JSON structure of "twobuildings.json".

Getting started with CityJSON

Search CityJSON

Tutorials / Getting started with CityJSON

Home

What is CityJSON?

News

Specifications

Schemas

Datasets

Extensions

Software

Tutorials

- Getting started with CityJSON

- Converting to/from CityGML files
- Validation of a CityJSON file
- Mapping the CityGML Noise ADE to a CityJSON Extension
- Modelling guide for CityJSON

TABLE OF CONTENTS

- 1 [Download a simple file with 2 buildings](#)
- 2 [Visualise it](#)
- 3 [Manipulate and edit it with cjo](#)
- 4 [What else?](#)

## Download a simple file with 2 buildings

Download [twobuilding.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.

```
1 {  
2   "type": "CityJSON",  
3   "version": "1.0",  
4   "metadata": {  
5     "geographicalExtent": [  
6       300578.235,  
7       5041258.061,  
8       13.688,  
9       300618.138,  
10      5041289.394,  
11      29.45  
12    ]  
13  },  
14  "CityObjects": {  
15    "Building_1": {  
16      "id": "Building_1",  
17      "type": "Building",  
18      "name": "Building 1",  
19      "x": 300578.235,  
20      "y": 5041258.061,  
21      "z": 13.688,  
22      "x2": 300618.138,  
23      "y2": 5041289.394,  
24      "z2": 29.45  
25    }  
26  }  
27}
```

# thank you.

Hugo Ledoux

h.ledoux@tudelft.nl  
3d.bk.tudelft.nl/hledoux



<https://cityjson.org>