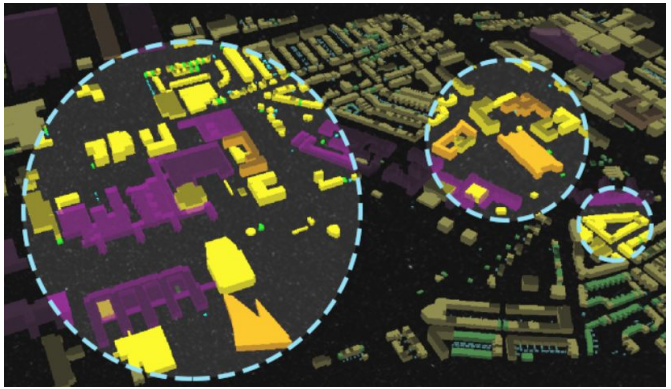
Directly Serving 3D Tiles From A Geo-DBMS

Yue Yang 5516862

Supervisors:
Dr.ir. B.M. Meijers,
Prof.dr.ir. P.J.M. van Oosterom

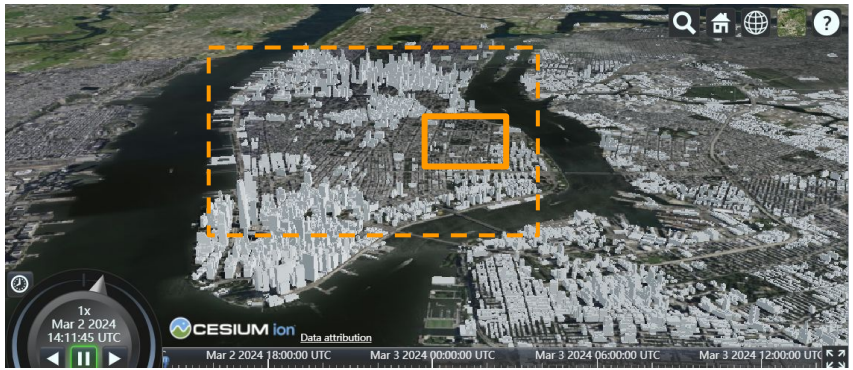
TU Delft



Content

1. Introduction
2. Background
3. Methodology
    - Storage database model
    - Web service workflow
4. Results
    - Datasets
    - Use case
    - Benchmarking
5. Conclusion & Discussion

Introduction



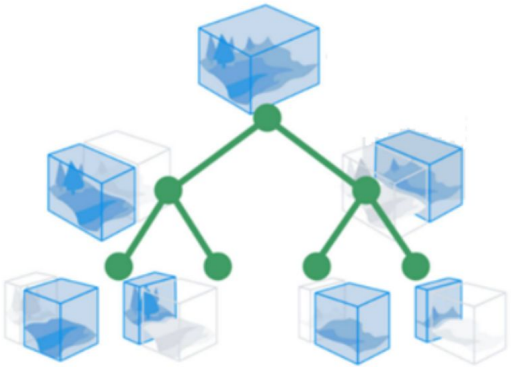
- Scalability: 3D Tiles divides massive geospatial data into chunks, and it supports progressively loading.
- Interoperability: 3D Tiles is as an open specification, widely adopted within the geoinformation and web communities.

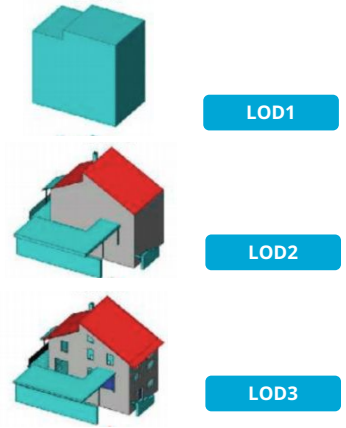
Introduction

- Multiple content

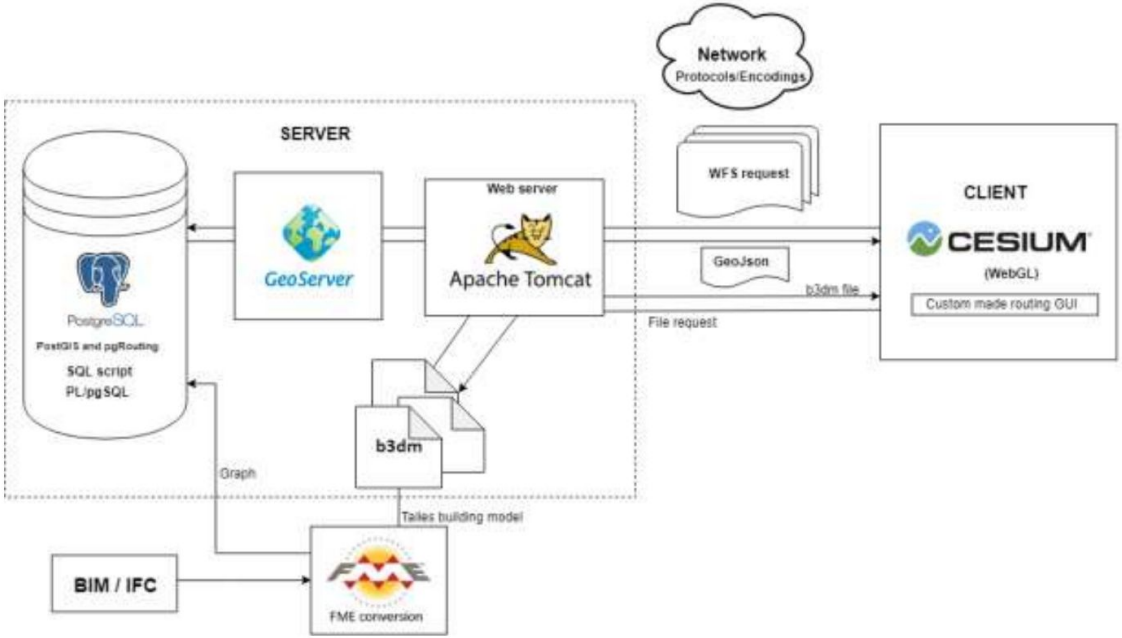
Format	Data
Batch 3D Model	3D buildings, BIM, Photogrammetry
Instance 3D Model	Multiple-rendered instanced features, such as rees, chairs, etc.
PointCloud	Pointcloud
Component	A combination of the first three

- Hierarchical level of detail (HLOD)





A typical approach to serve 3D Tiles



The system architecture [Alattas et al., 2021]

Background - typical approach

Dissemination of 3D Tiles with a file-based system:

- Issues:
  - Redundancy → data copies
  - Inconsistency
  - Fixed-size files → Flexibility

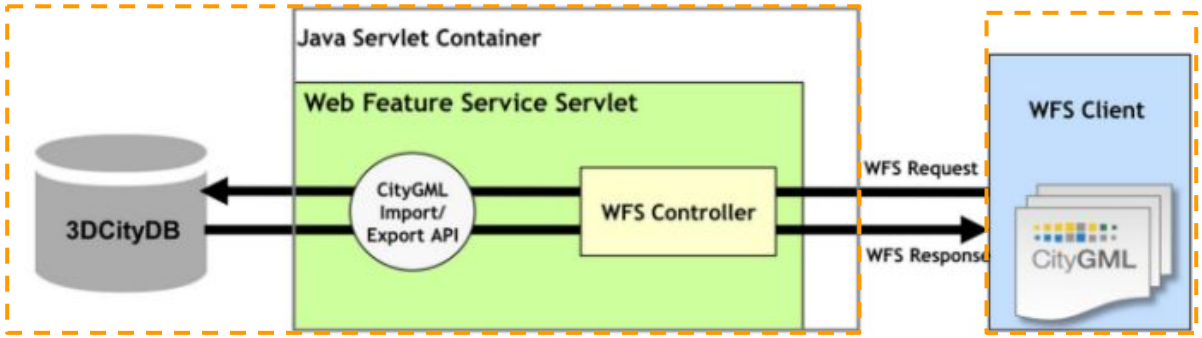
Can we directly serve 3D Tiles from a DBMS (Database management system)?

- Implementation not found

How to directly serve 3D models?

- Example: Implementation of the 3DCityDB Web Feature Service

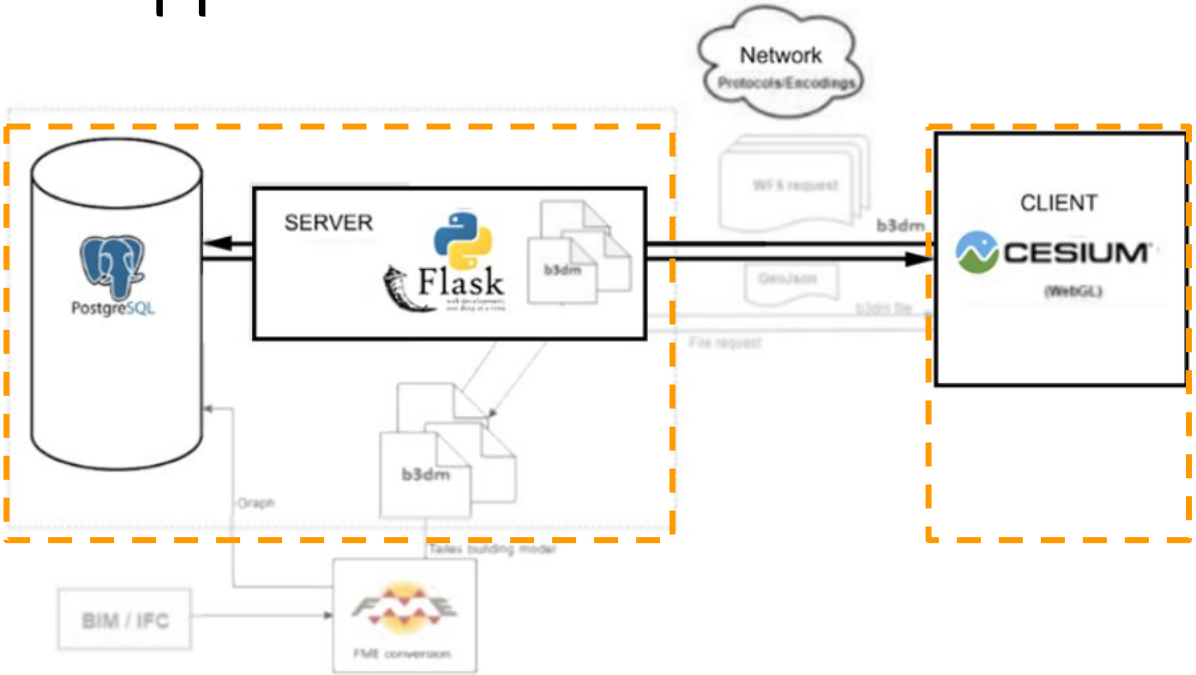
Example: 3DCityDB-Web-Feature-Service Workflow



The 3D City Database (3DCityDB) is an Open Source software suite which allows the im port, management, analysis, visualisation, and export of virtual 3D city models [Yao et al., 2018]. 3DCityDB-Web-Feature-Service compliant with OGC Web Feature Service 2.0, **extends the CityGML import/export tool to enable web-based access to city objects**.

What does a 3D Tiles web service need?

Proposed approach - 3D Tiles web service workflow



A more clean system architecture with geometry stored in the DBMS
It manages both geometry and attributes in the DBMS, and creates 3D Tiles on the fly

Proposed approach

- Reduce data copies (file representation) on the disk
- Consistency
- Flexibility! Deliver a subset of the available models with attribute filtering or spatial filtering
- Functionality ! geometric computation
- Indexing and clustering! Fast query executions

Research question

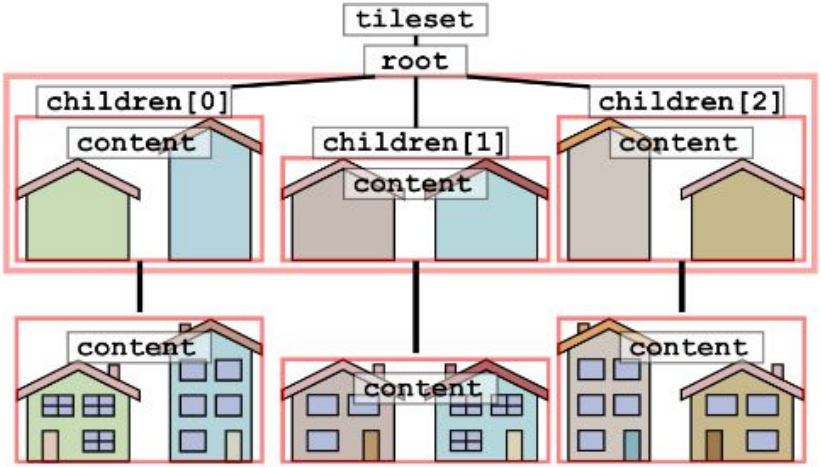
How to compactly store geometries and attributes in the database and effectively serve 3D Tiles directly to the client?

Scope:
Multiple levels of detail → Single LoD
3D Tiles → b3dm

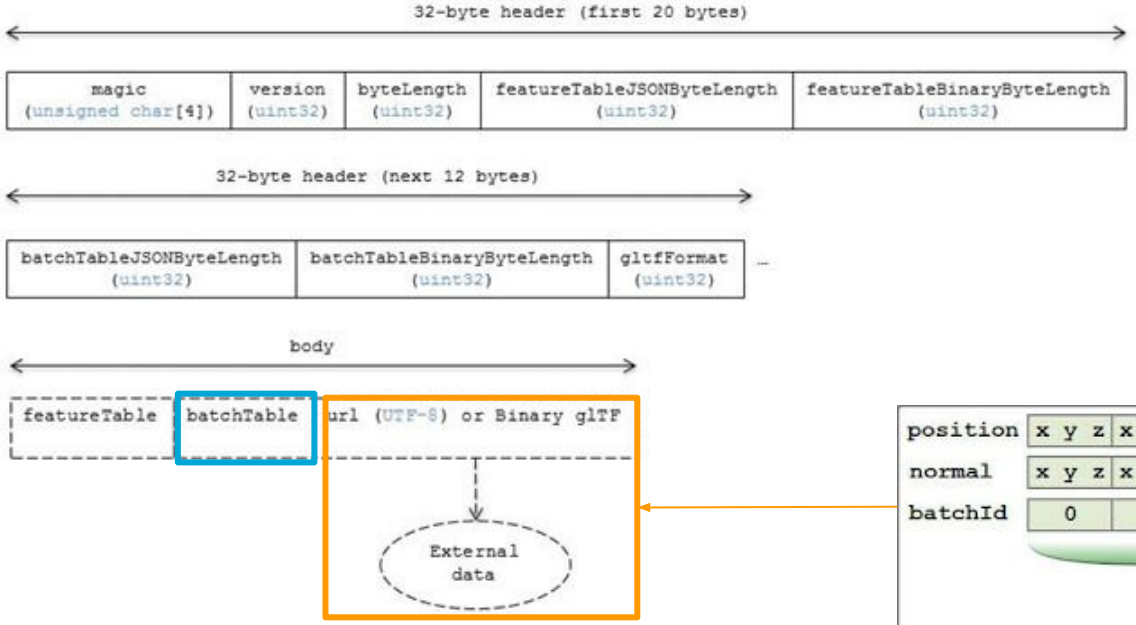
3D Tiles

- Hierarchical structure
- Tileset (JSON) + Content (a b3dm tile)

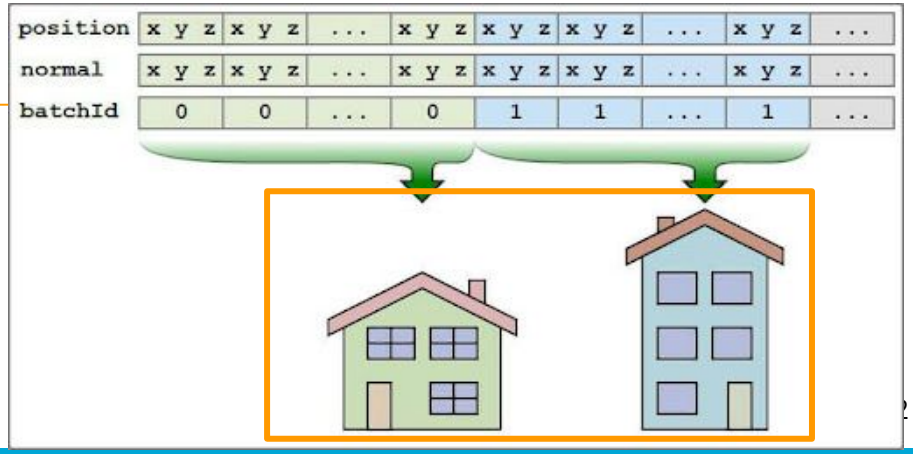
- It does not record:
  - geometry data (vertices, normals etc).
- It only records :
  - the logical relationship among tiles
  - Tile information (house name, construction year, height. etc.)



What is b3dm - Batched 3D Model

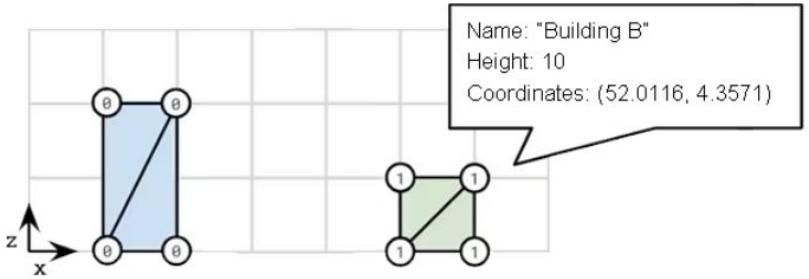


What is batched?
"batched" refers to grouping multiple 3D models into larger batches.

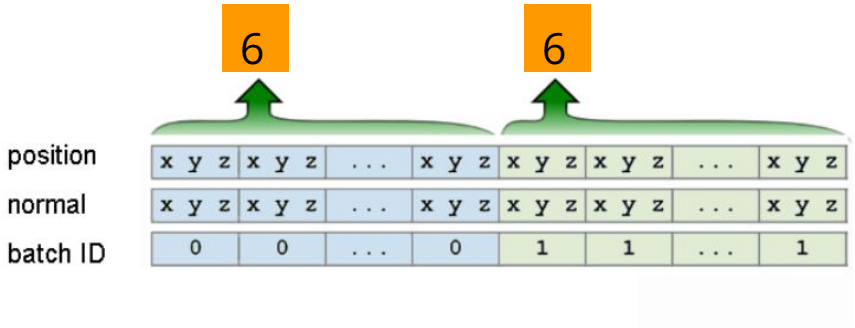


- The b3dm file format stores a **batch table** containing attribute information for each model, identified by unique batch IDs.
- In 3D Tiles, the geometry data is embedded in the format of glb (**binary glTF**)

What is b3dm – Batched 3D Model

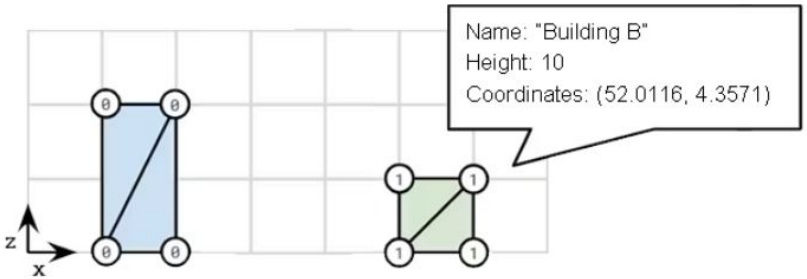


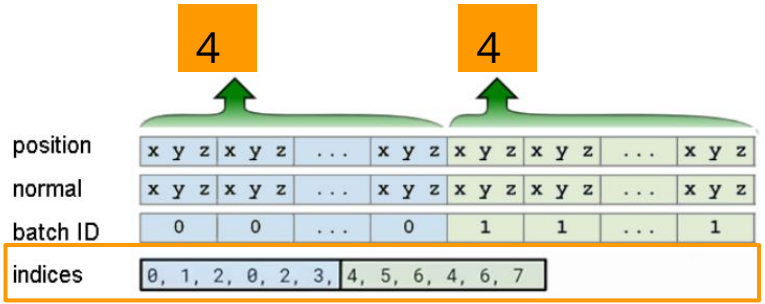
Feature ID	Name	Height	Coordinates
0	"Building A"	24	52.0112, 4.3576
1	"Building B"	12	52.0116, 4.3571

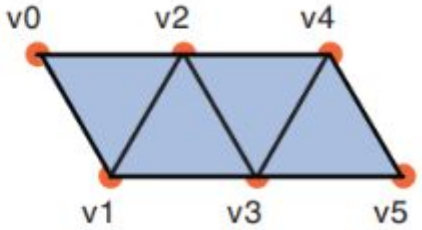


A binary glTF - It is a triangle mesh

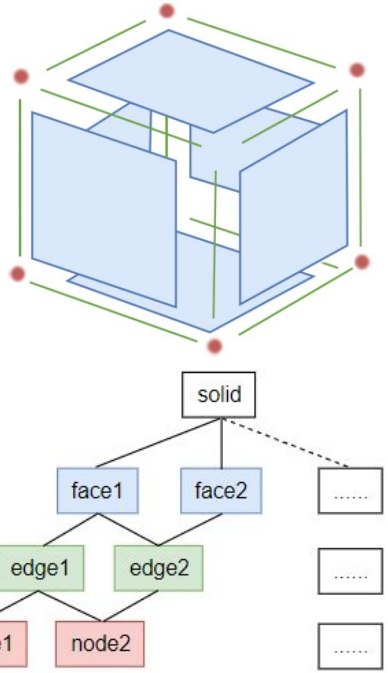
binary glTF – Triangle mesh



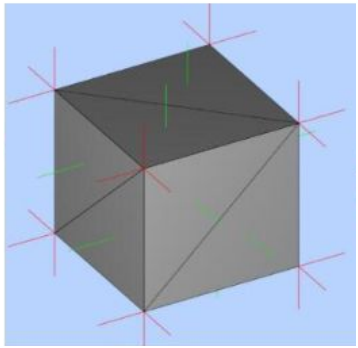




A Cube in Geomatics
(a boundary representation)



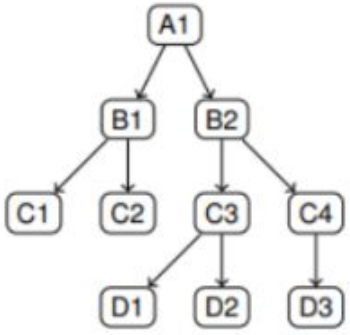
A Cube in Computer Graphics
(the binary glTF)



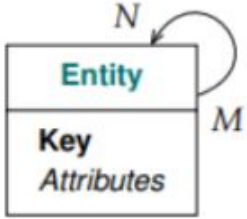
- It does not have the concept of face, edge, node.
- A GPU-friendly design.
- Vertex, normal and other information used by graphics chips for real-time rendering

Map hierarchy in the database

Hierarchical 3D Tiles → Parent-child relationship



(a)



(b)

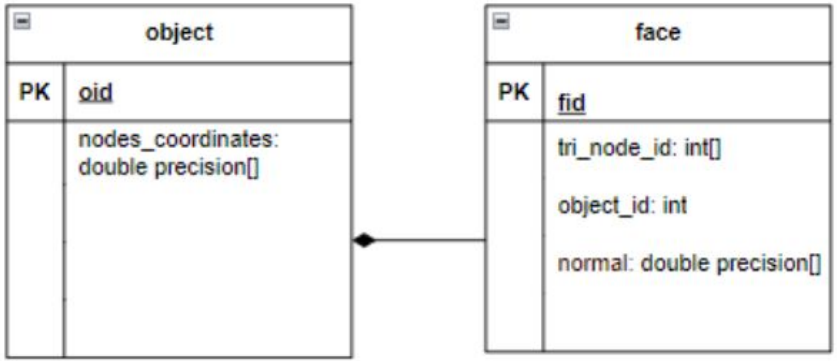
Entity		
ID	PID	*Attributes*
'A1'	NULL	...
'B1'	'A1'	...
'B2'	'A1'	...
'C1'	'B1'	...
'C2'	'B1'	...
'C3'	'B2'	...
'C4'	'B2'	...
'D1'	'C3'	...
'D2'	'C3'	...
'D3'	'C4'	...

(c)

Methodology

Methodology – Storage database model

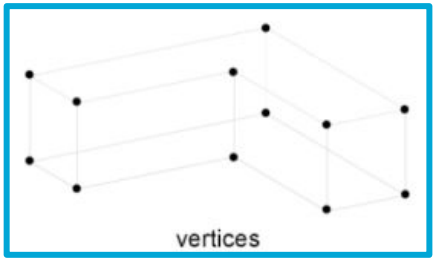
- How is the b3dm organised in DB?
- Coordinates of object, normals, and triangulated topology of faces



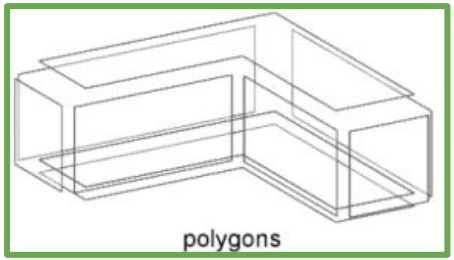
- why?
- Saving storage space.
- Normals shared by triangles on the same face. Vertices shared by multiple faces in the same object.

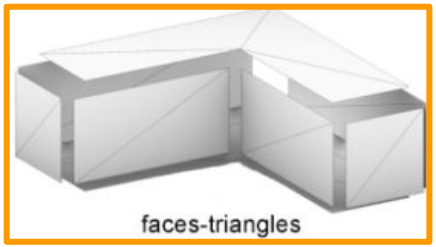
An L-shaped Polyhedron example

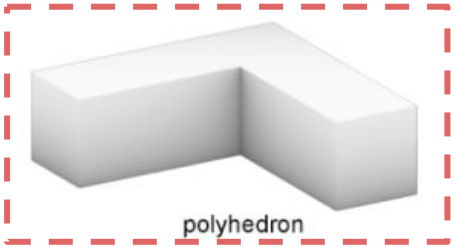
12



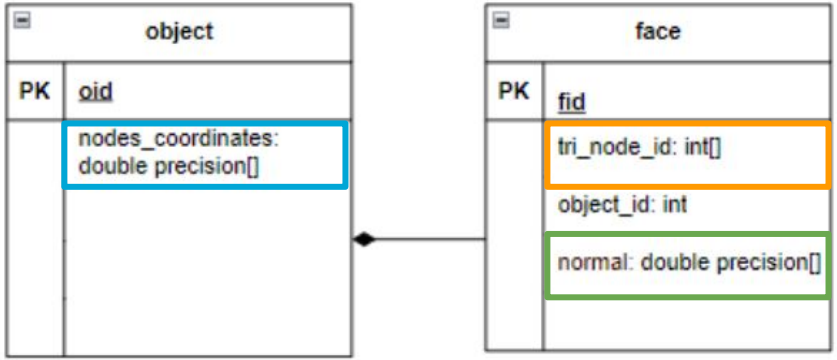
8

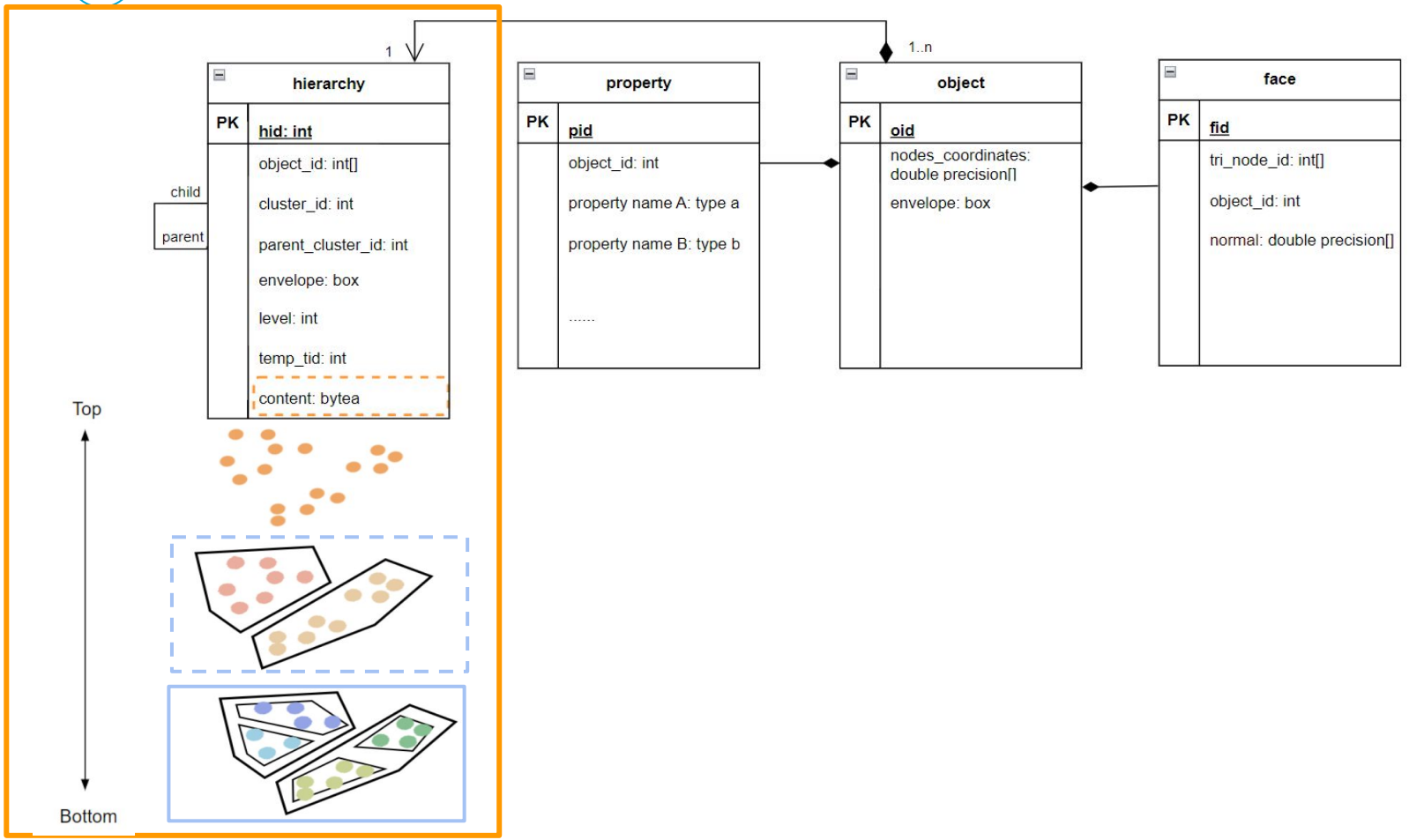




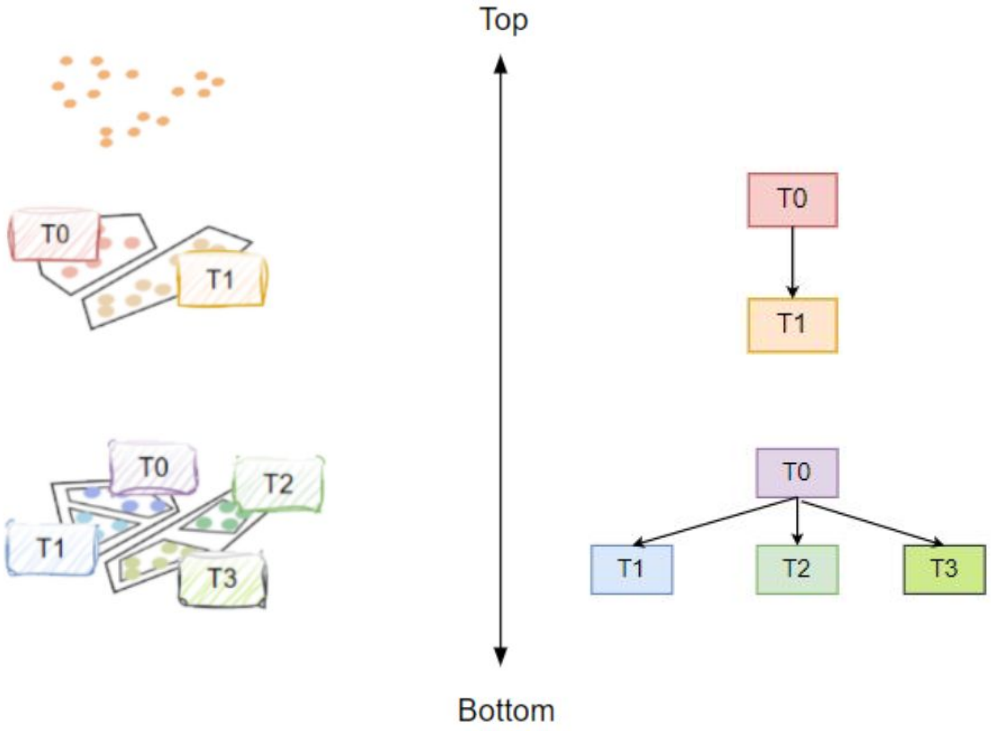


20



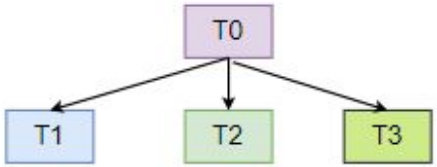

1
1..n
hierarchy
PK
hid: int
object_id: int[]
child
cluster_id: int
parent
parent_cluster_id: int
envelope: box
level: int
temp_tid: int
content: bytea
property
PK
pid
object_id: int
property name A: type a
property name B: type b
.....
object
PK
oid
nodes_coordinates: double precision[]
envelope: box
face
PK
fid
tri_node_id: int[]
object_id: int
normal: double precision[]
Top
Bottom

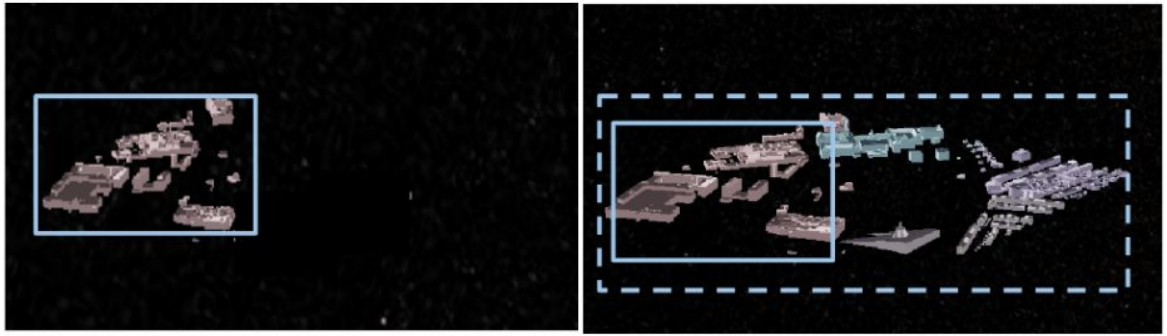
Methodology – Tiling



Organise 3D Tiles from the storage model, and create a **Tileset.**

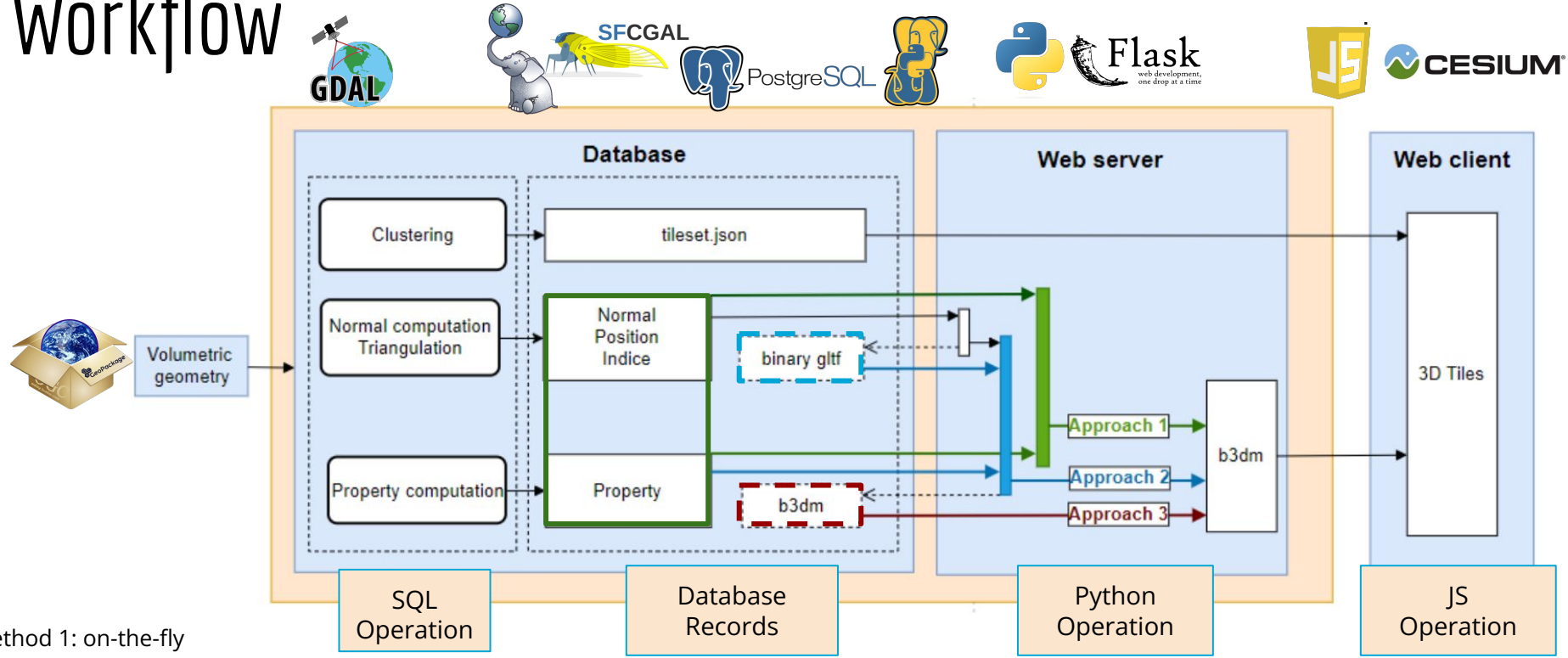
Methodology – Tiling





(a) zoom out (b) zoom in

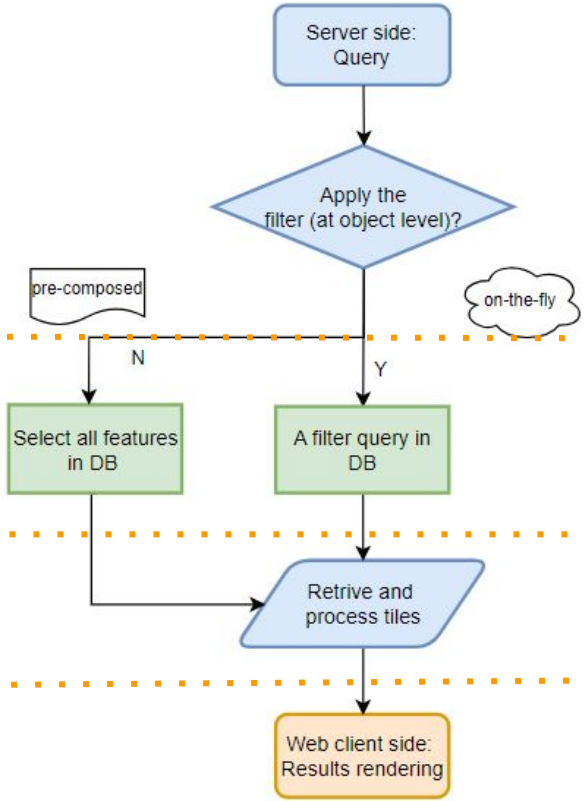
Workflow



Method 1: on-the-fly
Method 2: pre-composed geometry (binary glTF)
Method 3: pre-composed b3dm

The overview of the 3D Tile approach (Simplified)

Web service procedure - b3dm generation



Query in the DB (b3dm)

Complete 3D Tile generation in the web server

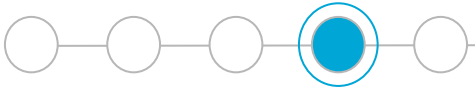
Display on the web client

on-the-fly or pre-composed ?

I: fetches precomputed information to compose a b3dm (Attribute/ Spatial Filter)

II: fetch precomposed binary glTF and properties to compose a b3dm (Different Attributes)

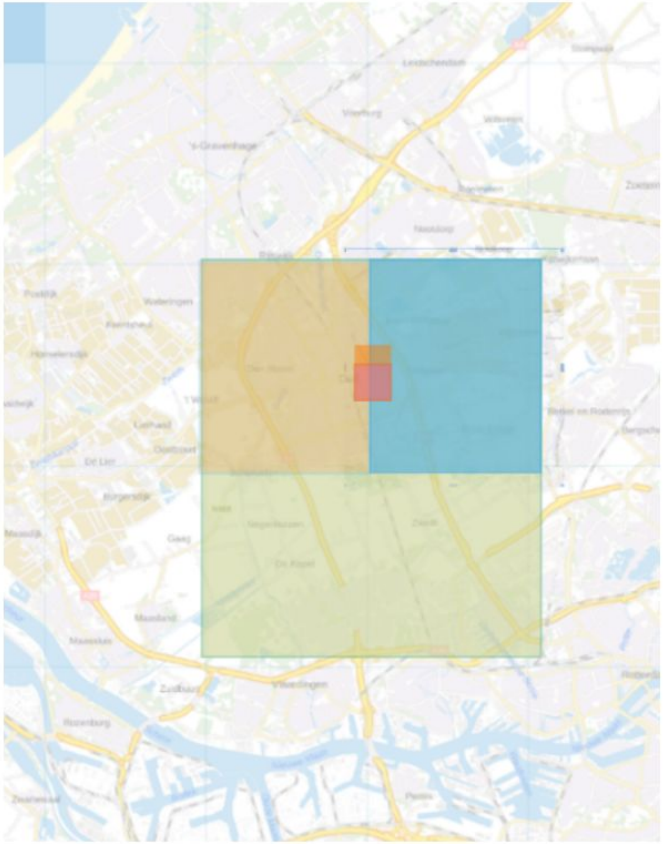
III: fetch precomposed b3dm stored in DB



Results

Datasets

Name	Files	LOD	Geometry representation	Disk size (MB)	Description
Delft	4	LOD1. 2	2D polygon	67.4	City Delft and surroundings
Delft_NE	1	LOD1.2	2D polygon	15.4	Northeastern part of City Delft
Delft_NW	1	LOD1. 2	2D polygon	39.4	Northwestern part of City Delft
Campus_LOD1	3	LOD1. 2	MultiPolygonZ	32.4	TU Delft Campus
Campus_LOD2	3	LOD2.2	MultiPolygonZ		
BK_LOD1	2	LOD1.2	MultiPolygon Z	15.13	BK and surrounding areas
BK_LOD2	2	LOD2.2	MultiPolygon Z		
Aula_LOD1	1	LOD1.2	MultiPolygon Z	17.3	Aula and surrounding areas
Aula_LOD2	1	LOD2.2	MultiPolygon Z		
Aula_Extrusion	1	LOD1.2	2D polygon		



Benchmarking:

- **Serving methodology (Datasets from 3DBAG, different LODs)**
  - Database storage
  - Web retrieval
- **Tiling methodology (Datasets from PDOK, large spatial extent)**

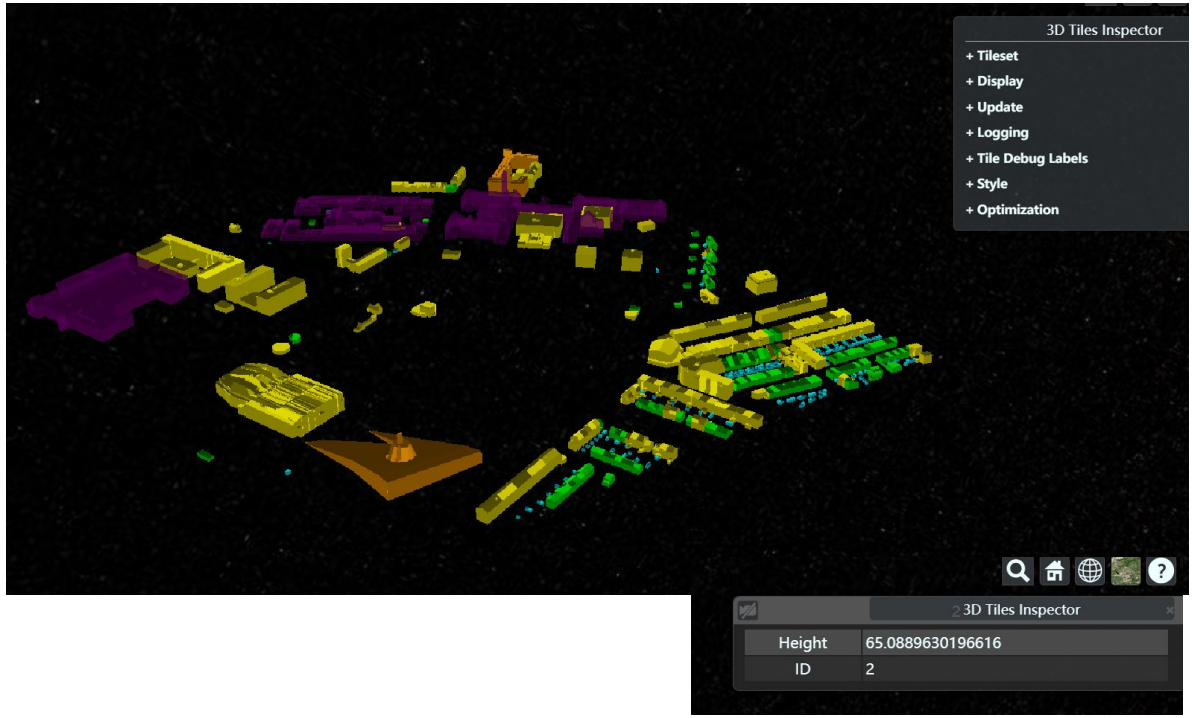
Use case

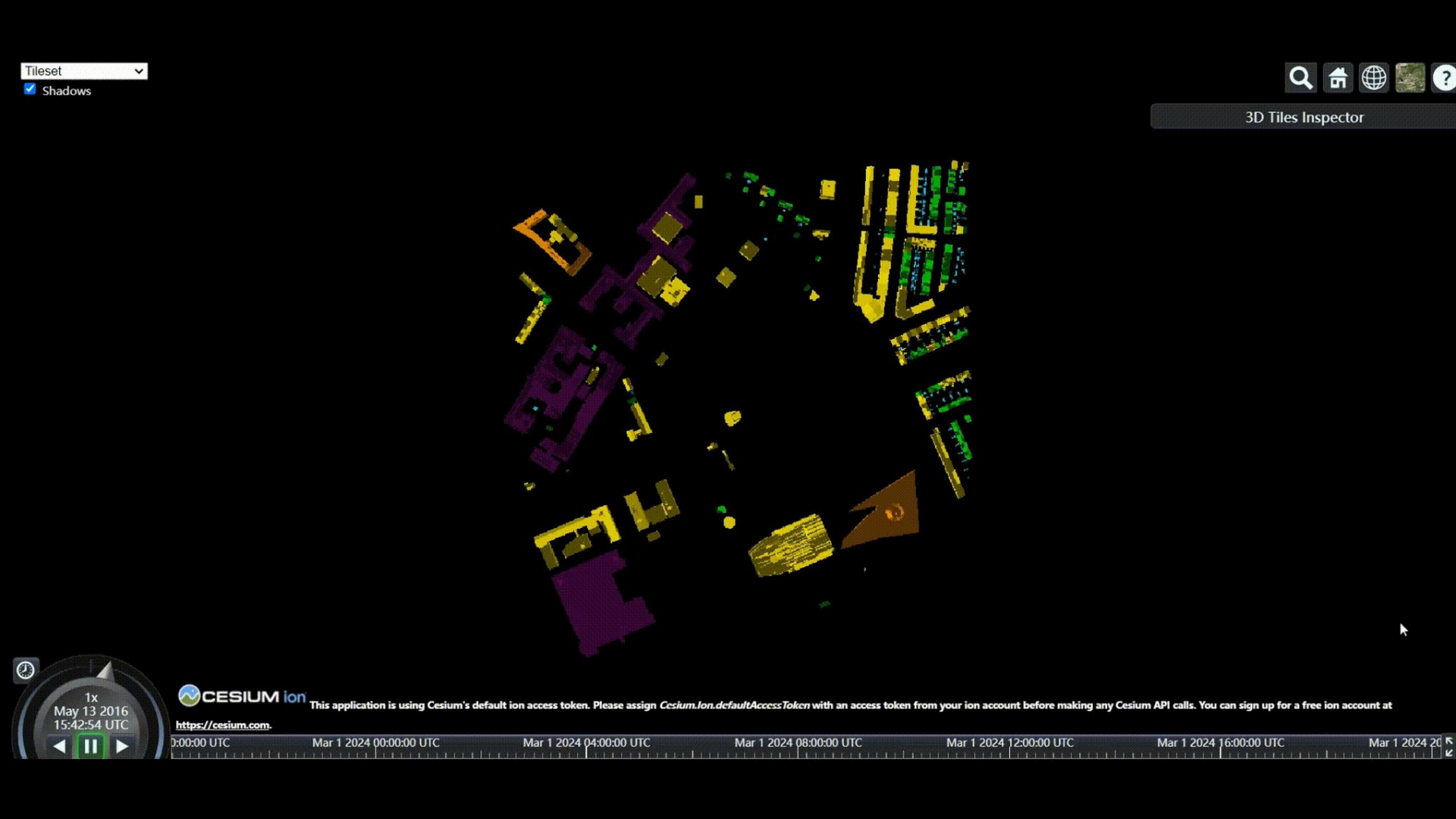
Resistance to sea level rise - TU Delft campus

When the sea level rises to 10m, where are temporary safe destinations?

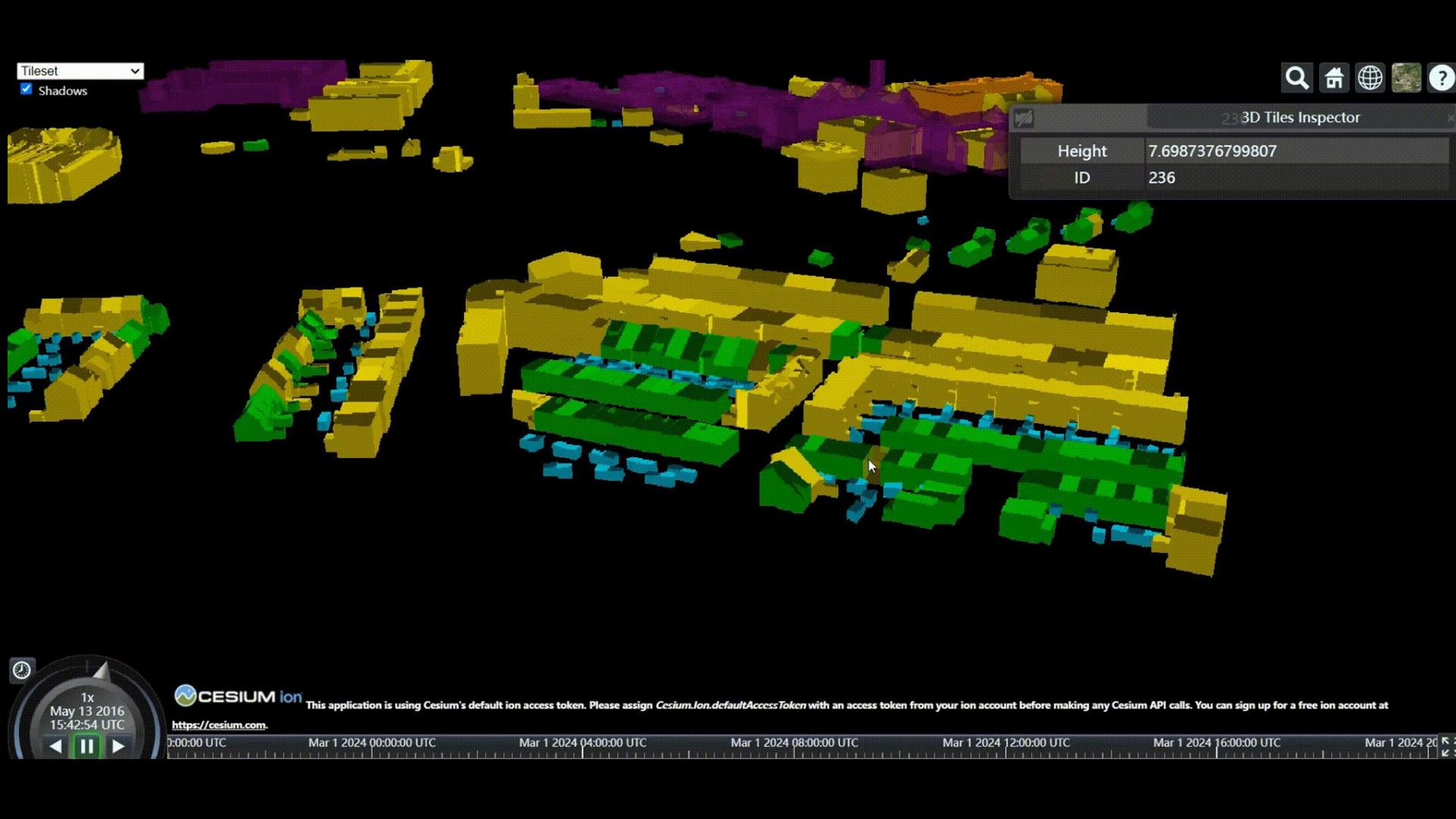
1. visualisation based on height
2. retrieve buildings under water after sea level rises
3. only retrieve buildings above 10m

(Assume the world is flat)

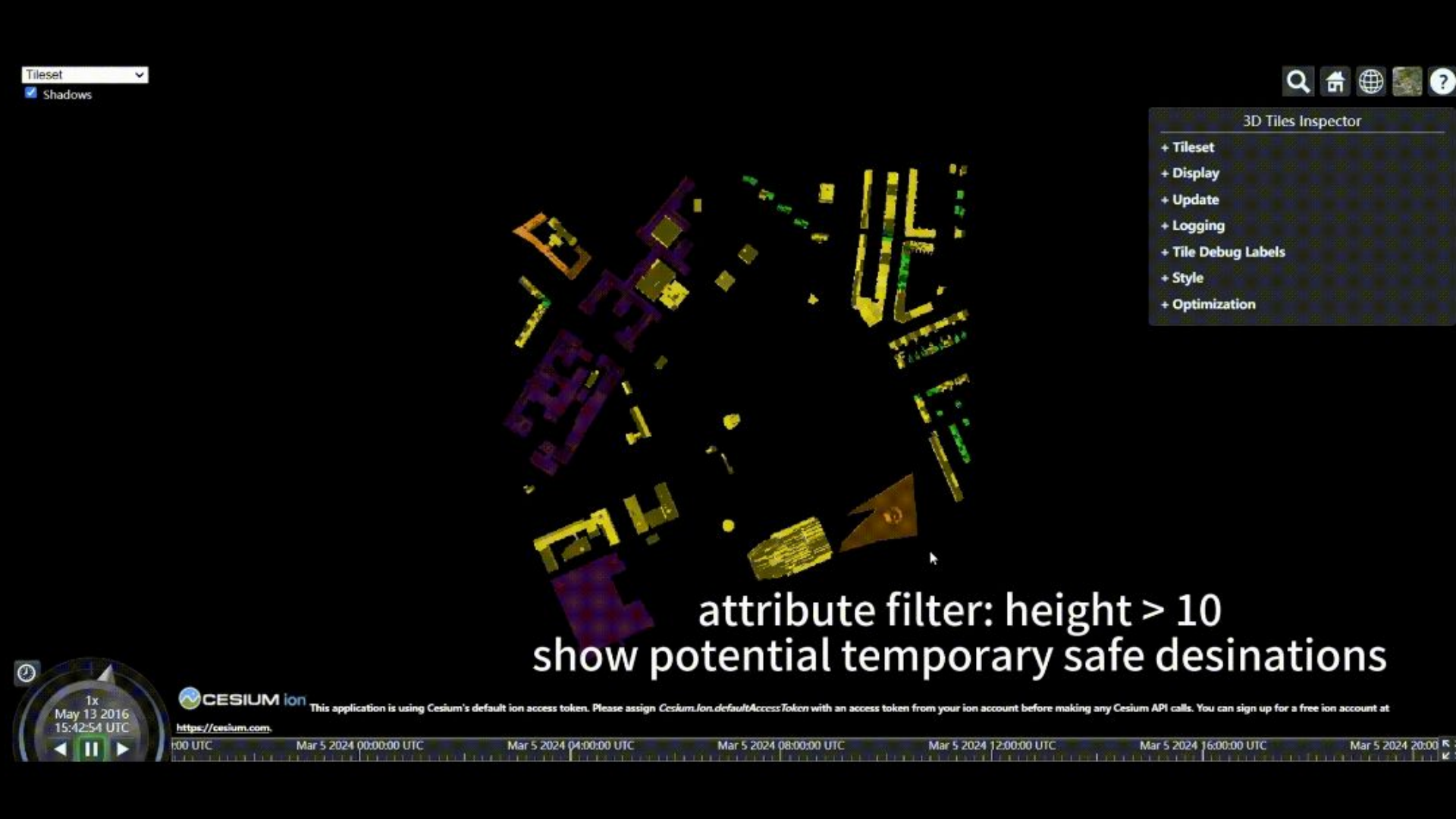


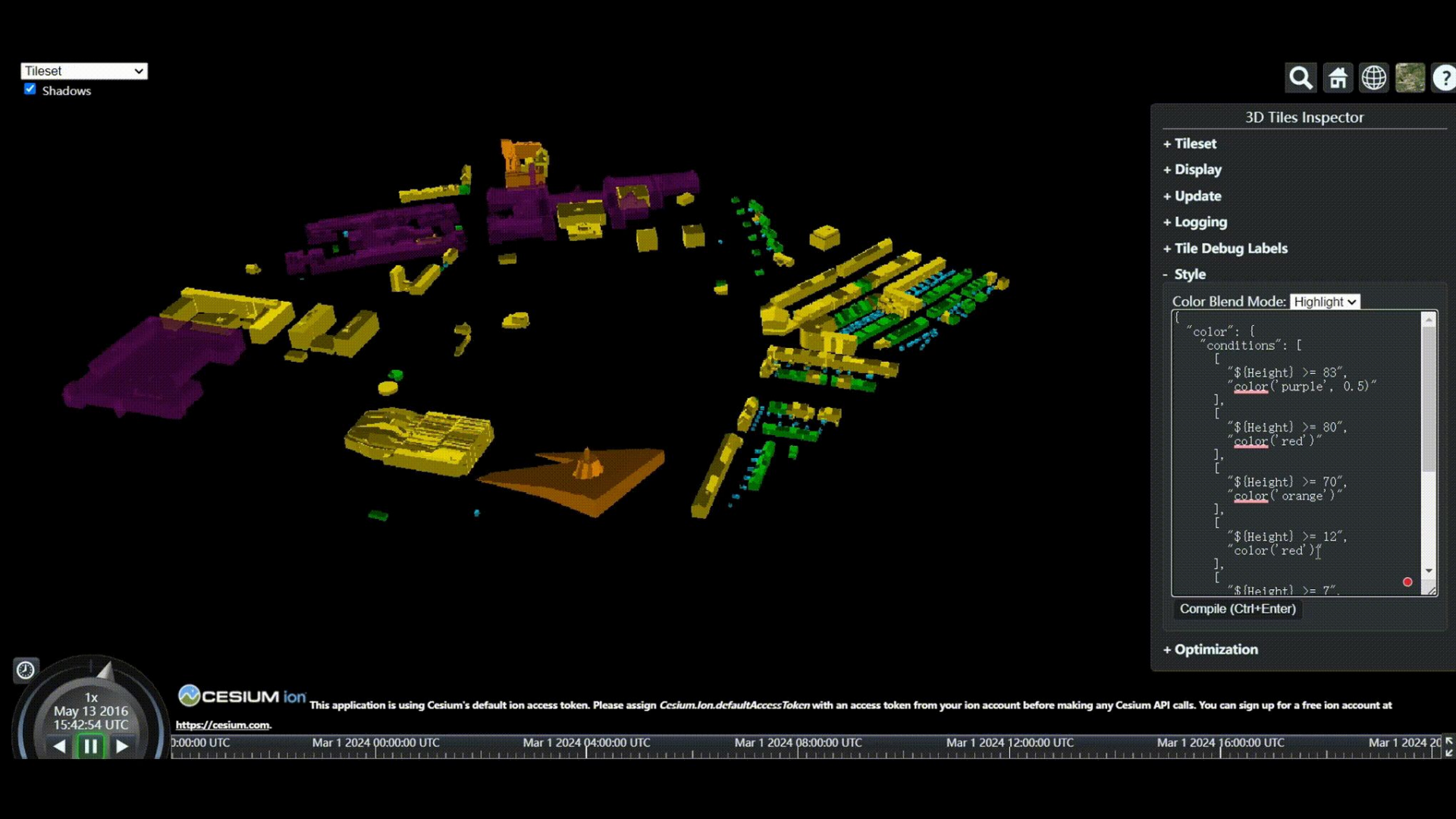

Tileset
Shadows
3D Tiles Inspector
1x
May 13 2016
15:42:54 UTC
CESIUM ion
This application is using Cesium's default ion access token. Please assign Cesium.Ion.defaultAccessToken with an access token from your ion account before making any Cesium API calls. You can sign up for a free ion account at https://cesium.com.
:00:00 UTC
Mar 1 2024 00:00:00 UTC
Mar 1 2024 04:00:00 UTC
Mar 1 2024 08:00:00 UTC
Mar 1 2024 12:00:00 UTC
Mar 1 2024 16:00:00 UTC
Mar 1 2024 20


943 3D Tiles Inspector
Height 48.4551131399348
Building_type Residential
Construction_year 1973
City Delft
ID 943

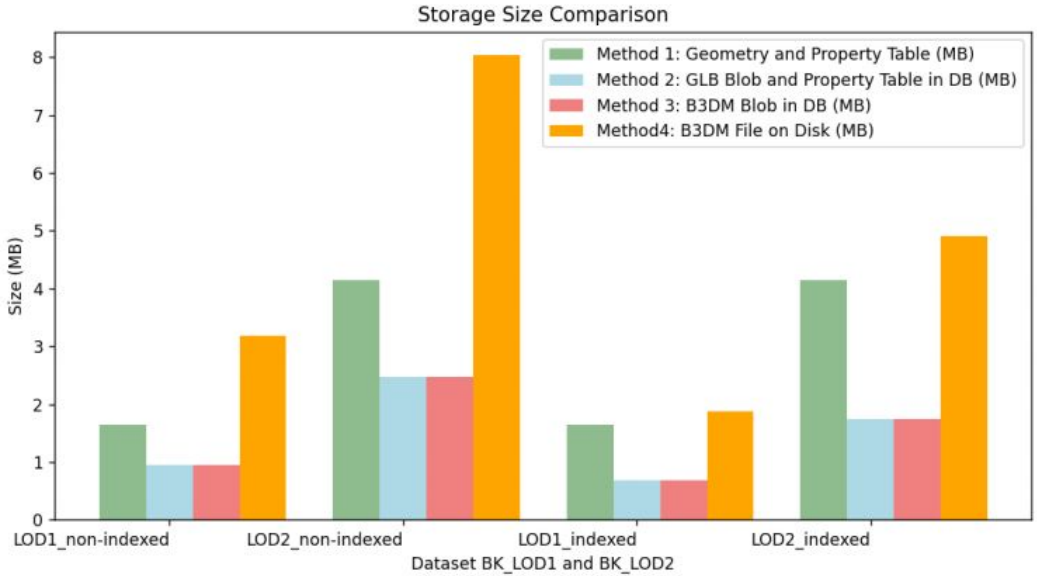

Tileset
Shadows
3D Tiles Inspector
Height
7.6987376799807
ID
236
1x
May 13 2016
15:42:54 UTC
CESIUM ion
This application is using Cesium's default ion access token. Please assign Cesium.Ion.defaultAccessToken with an access token from your ion account before making any Cesium API calls. You can sign up for a free ion account at
https://cesium.com.
0:00:00 UTC
Mar 1 2024 00:00:00 UTC
Mar 1 2024 04:00:00 UTC
Mar 1 2024 08:00:00 UTC
Mar 1 2024 12:00:00 UTC
Mar 1 2024 16:00:00 UTC
Mar 1 2024 20


Tileset
Shadows
3D Tiles Inspector
+ Tileset
+ Display
+ Update
+ Logging
+ Tile Debug Labels
+ Style
+ Optimization
Attribute filter: height <= 10
1x
May 13 2016
15:42:54 UTC
CESIUM ion
This application is using Cesium's default ion access token. Please assign Cesium.Ion.defaultAccessToken with an access token from your ion account before making any Cesium API calls. You can sign up for a free ion account at https://cesium.com.
:00 UTC
Mar 5 2024 00:00:00 UTC
Mar 5 2024 04:00:00 UTC
Mar 5 2024 08:00:00 UTC
Mar 5 2024 12:00:00 UTC
Mar 5 2024 16:00:00 UTC
Mar 5 2024 20:00


Tileset
Shadows
3D Tiles Inspector
+ Tileset
+ Display
+ Update
+ Logging
+ Tile Debug Labels
+ Style
+ Optimization
attribute filter: height > 10
show potential temporary safe desinations
1x
May 13 2016
15:42:54 UTC
CESIUM ion
This application is using Cesium's default ion access token. Please assign Cesium.Ion.defaultAccessToken with an access token from your ion account before making any Cesium API calls. You can sign up for a free ion account at
https://cesium.com.
:00 UTC
Mar 5 2024 00:00:00 UTC
Mar 5 2024 04:00:00 UTC
Mar 5 2024 08:00:00 UTC
Mar 5 2024 12:00:00 UTC
Mar 5 2024 16:00:00 UTC
Mar 5 2024 20:00


Tileset
Shadows
3D Tiles Inspector
+ Tileset
+ Display
+ Update
+ Logging
+ Tile Debug Labels
- Style
Color Blend Mode: Highlight
{
"color": {
"conditions": [
[
"${Height} >= 83",
"color('purple', 0.5)"
],
[
"${Height} >= 80",
"color('red')"
],
[
"${Height} >= 70",
"color('orange')"
],
[
"${Height} >= 12",
"color('red')"
],
[
"${Height} >= 7",
Compile (Ctrl+Enter)
+ Optimization
1x
May 13 2016
15:42:54 UTC
CESIUM ion
This application is using Cesium's default ion access token. Please assign Cesium.Ion.defaultAccessToken with an access token from your ion account before making any Cesium API calls. You can sign up for a free ion account at https://cesium.com.
:00:00 UTC
Mar 1 2024 00:00:00 UTC
Mar 1 2024 04:00:00 UTC
Mar 1 2024 08:00:00 UTC
Mar 1 2024 12:00:00 UTC
Mar 1 2024 16:00:00 UTC
Mar 1 2024 20

Benchmarking storage system



Method 1: on-the-fly

Method 2: pre-composed geometry (binary glTF)

Method 3: pre-composed b3dm

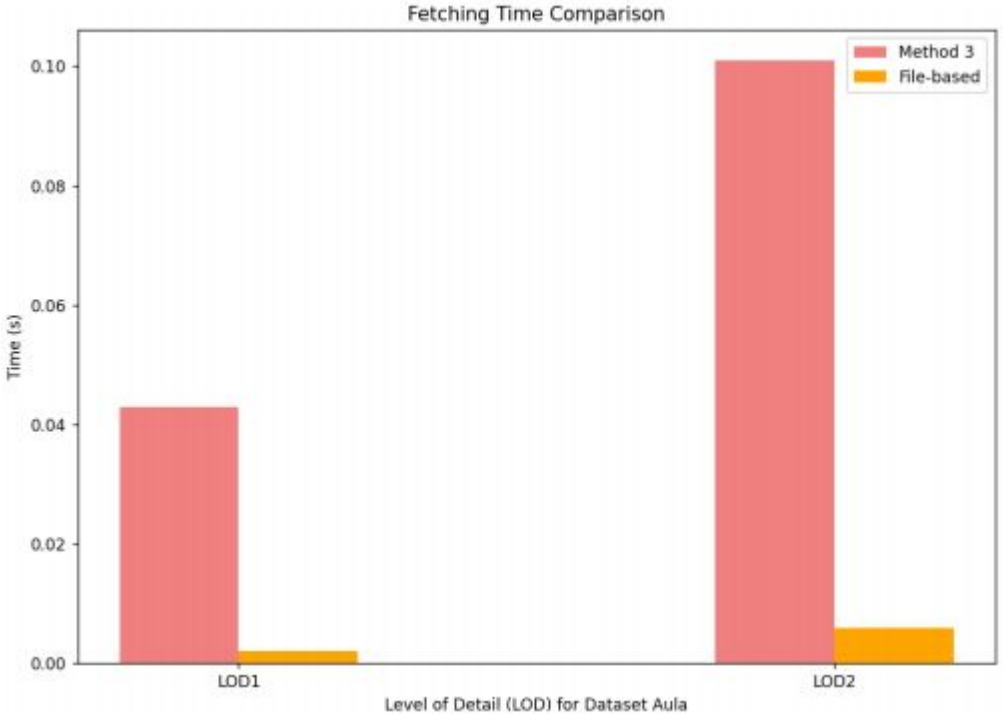
Method 4: file-based

Content(b3dm) size performance:

on-the-fly generation takes up less disk storage than file-based approach.

pre-composed approaches takes least space. However, if there are only fixed b3dm or glb blobs, it lacks flexibility.

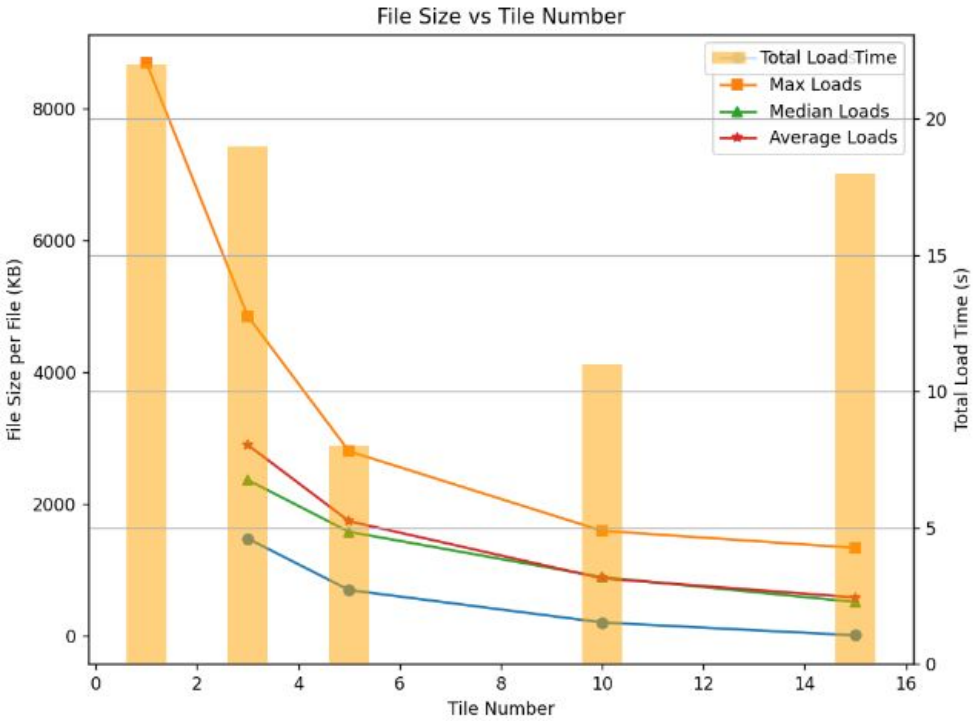
Benchmarking web retrieval



- Timing
  - **File-based** is the fast
  - DBMS approaches:
    - **Methode 3** < Methode 2 < Methode 1
    - Two pre-composed approaches outweighs on-the-fly approach (Method 1)
    - Precomposed b3dm (Method 3) is faster than precomposed geometry (Method 2)

Figure: Time performance for fetching one specific tile

Benchmarking tiling methodology



- The variation between tile sizes is big.

Maximum size

Minimum size

- Overall transmission performance is affected by tile number and tile size.
- As the tile number increases, the overall loading time is reduced. However, a turning point occurs when the tile number continues increasing.

Figure: Relationship between tile size and fetching time (bar chart: tile number; line chart: tile sizes)

Conclusion and suggestions

Does a system architecture with geometry stored in the DBMS that supports directly serving b3dm to the client outweighs the geometry stored in files?

DB:

- Yes
- face-object approach: triangulated topology and unique coordinates of the object
- compared with file-based approach, the on-the-fly and precomposed approaches take up less disk storage
    - Instead of producing a b3dm file (data copy) on the disk, it calls memory and generate content on server-side

Web:

- It depends
- Regarding fetching time, pre-composed approaches outweigh the on-the-fly approach.
    - Both are much slower than serving from disk
- However, DBMS approach maintains spatial and temporal consistency, offers more flexibility to users.
- It is fair to believe on-the-fly a promising approach with the generation process improved (eg: C++)

Triangulation comparison

	ST_DelaunayTriangles	ST_Tesselate
Convex polygon	√	√
Concave polygon	×	√
Vertical polygon	×	√
tilted polygon	Depends on tilted degree	Depends on coordinates precision
	😐	🙂

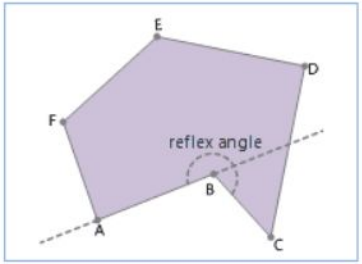


Figure 1 Concave Polygon

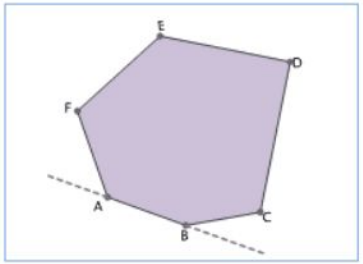
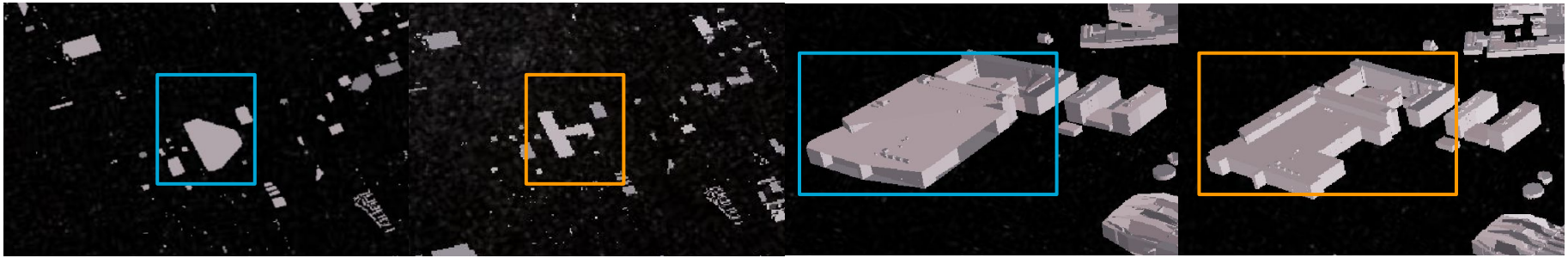


Figure 2 Convex Polygon

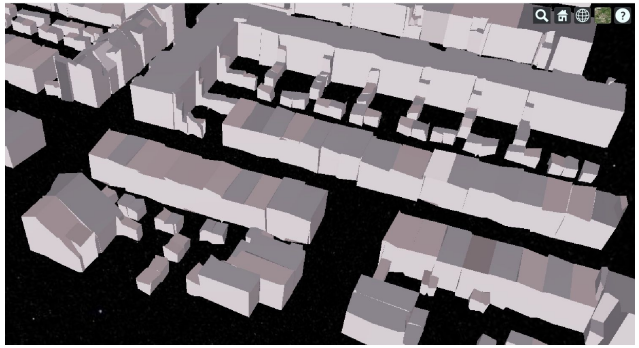


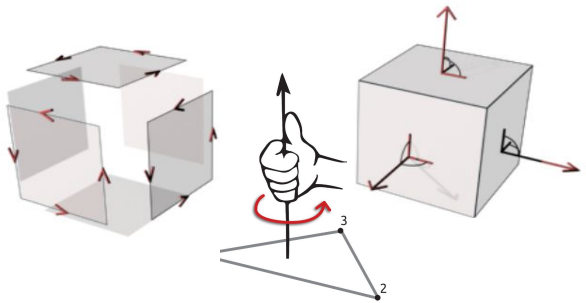
Limitation - Geometric operations

- **Triangulation**
- A valid polyhedron is not ensured. Triangulation can yield incorrect orientation with respect to the polyhedron.

- Faces are expected to keep an counterclockwise orientation when viewed from the outside of an three-dimensional enclosure.

- Precision is not confirmed. A set of **geometric operations introduced floating-point errors**, leading to inconsistency between the resulting coordinates and the raw geometry.



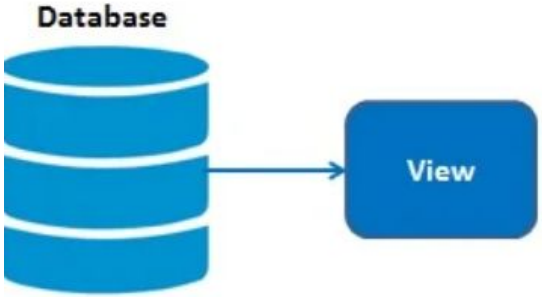


Limitation - serving and tiling

- Serving 3D Tiles on-the-fly approach is slow.
- Fetching precomposed 3D Tiles is faster than on-the-fly, but slower than file-based. The BLOB storage in the table is a bit redundant.
- Hierarchical k-means tiling does not ensure an even distribution.
- Data transfer back and forth between DB and Flask API occurs.

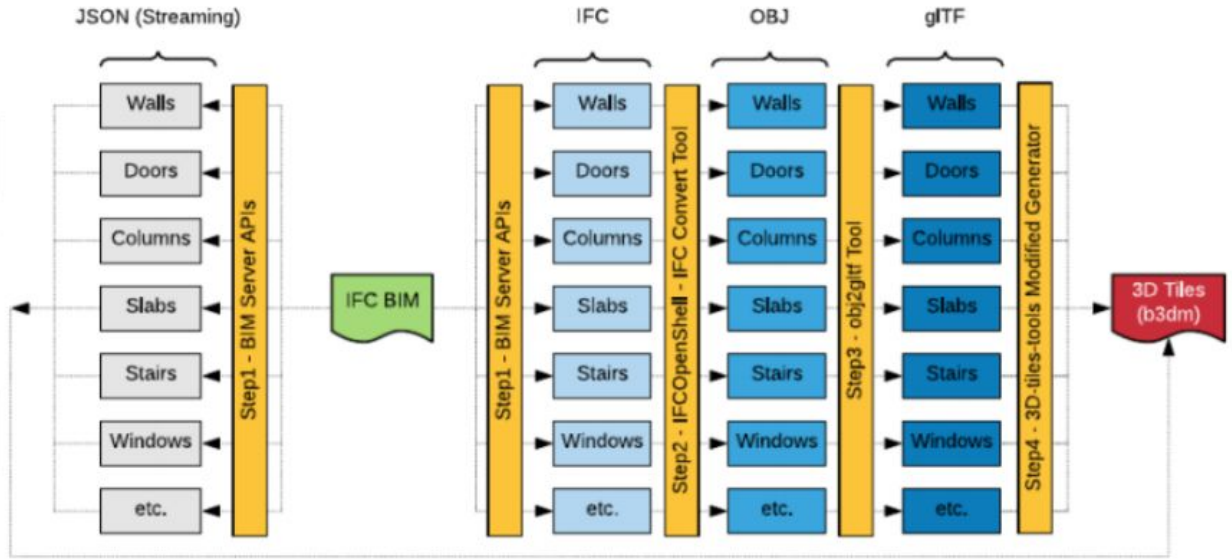
Future development

- More native database functionality, support create a set of views based on raw geometries.
- Reduce redundancy, faster



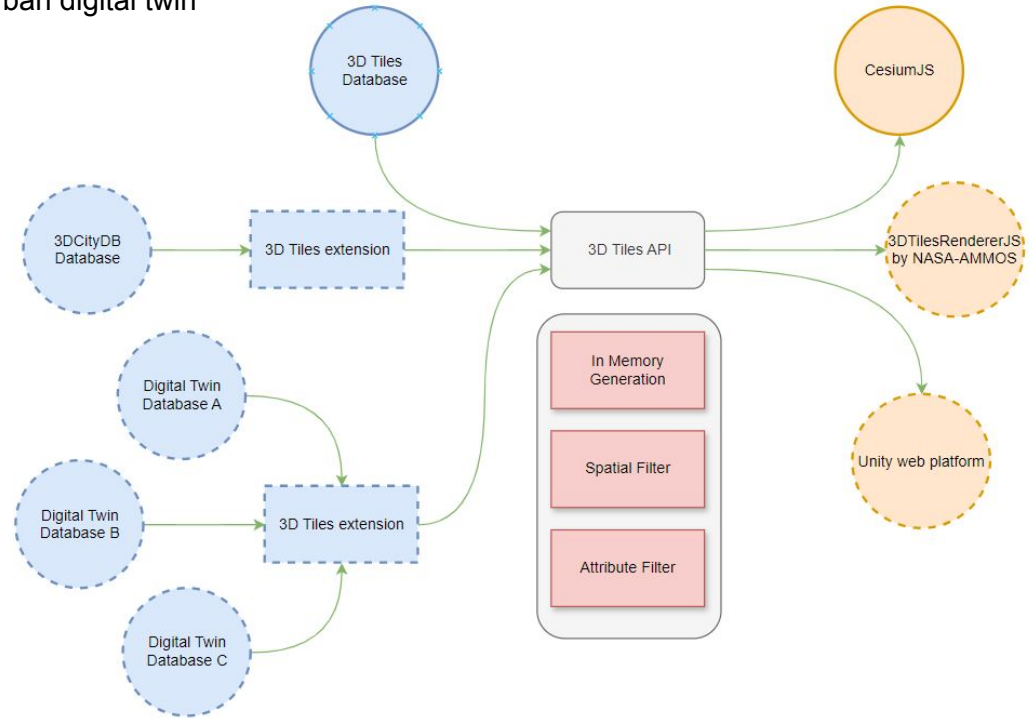
Future development

- Collaborating with more 3D data formats



Future development

- Interoperability with other existing databases and web clients
- Urban digital twin

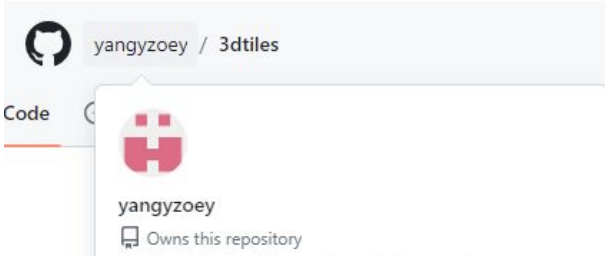


Future development

Others:

- Adapt to multiple levels of details
- Improve indexing and clustering
- 3D Tiles generation on the web client side
- ……

Github link



Feel free to play with it.

Your feedback and suggestions are more than welcomed! ☺

Thank you for attention!