# 3DTiler: A tool to georeference 3D Models and generate 3D Tiles

Rijin Shaji[1], Théo Bouveyron[2], Christian Willmes[1]

[1] Institute of Geography, University of Cologne, Albertus-Magnus-Platz, 50923 Cologne, Germany - (rshaji, cwillmes)@uni-koeln.de
[2] Department for Digital Humanities, University of Cologne, Albertus-Magnus-Platz, 50923 Cologne, Germany - tbouvey1@uni-koeln.de

**Keywords:** OGC 3D Tiles, 3D Geoinformation, Georeferencing, Digital Twin.

**Abstract**

As 3D geoinformation and city models are increasingly used across domains like urban planning, education, indoor navigation, asset management, smart cities, augmented reality and digital twins, the demand for standardized 3D geospatial formats that are optimized for fast and efficient streaming, processing and rendering is growing. The here presented 3DTiler tool was developed in the context of the Virtual Campus project at the University of Cologne, where a georeferenced 3D model of the University of Cologne is developed and created, primarily to facilitate geometry matching based indoor positioning, and based on this allow for indoor navigation and geospatial augmented reality across the University Campus infrastructure and its buildings. 3DTiler was designed and developed to georeference 3D building models and generate according interoperable OGC 3D Tiles, to use the georeferenced 3D models in CesiumJS based web and game engine applications, as well as to provide our 3D data in a standardized and interoperable format for use in any client software, that supports the OGC 3D Tiles standard.

## 1. Introduction, Context, Related Work

As 3D city models are increasingly used across domains like urban planning, education, and digital twins (Biljecki et al., 2015), the demand for standardized 3D geospatial formats is growing. The OGC 3D Tiles format addresses this by enabling the scalable streaming and rendering of massive 3D datasets (Cozzi and Lilley, 2023).

In the context of the CampusGIS2[1] project, that was funded by Gender and Diversity Management department of the University of Cologne (UoC) with a focus on accessibility and features for mobility impaired persons between 2020 and 2024, we developed an interactive web map and a comprehensive and detailed Spatial Data Infrastructure (SDI) for the University of Cologne Campus, Buildings and its Infrastructure. The final product of that project was the UoCMaps[2] interactive web map application, that is now the official site map of the UoC. In the second half of the CampusGIS2 project we successfully obtained funds for an additional project, the Virtual Campus[3] of the University of Cologne. This project extends the comprehensive spatial database developed for the CampusGIS2 project to the third dimension and includes detailed indoor information and 3D data of the UoC buildings and infrastructure. One major focus of the Virtueller Campus Project is on developing a geometry matching based indoor positioning system, based on the 3D indoor geometries of the UoC buildings (Willmes et al., 2024).

The development of the 3DTiler application presented here started in the context of the CampusGIS2 project as a BSc thesis project (Ganser, 2024). It resulted in the tilesetGenerator tool (Ganser, 2025), which allows users to create hierarchically structured 3D Tile tilesets including metadata by georeferencing multiple 3D models in an intuitive, client-based WebGIS, while giving real-time visual feedback. The automatically generated tilesets are used in the CesiumJS based 3D view, as part of the UoCmaps application. tilesetGenerator, however, is highly customized to the requirements and data structure of the Virtual Campus and CampusGIS2 projects. To make the tool usable for a broader audience, it was decided to develop it further to be independent of our custom project infrastructure, and renamed the project to 3DTiler.

### 1.1 Related work

The Virtual Campus project is directly linked to CampusGIS2, a project focused on creating a Spatial Data Infrastructure (SDI) and Geographic Information System (GIS) for the UoC campus. This includes the interactive web map application UoCMaps specifically designed to enhance inclusivity and accessibility for individuals with disabilities or other disadvantages (Willmes et al., 2024). The standout feature is its routing engine, which enables accessible and barrier-free navigation throughout the UoC campus. To support this highly data-intensive routing, the UoC campus is meticulously mapped in detail by project members and by students participating in Geography field mapping courses. The CampusGIS2 project is a follow up to the original CampusGIS project (Baaser, 2010), which was funded from 2005 to 2010, developing an interactive Web Map that already had routing and navigation functionality for wheelchairs, with accessibility in mind. As part of the previous CampusGIS project mentioned above, a complete LoD2 model of the UoC Campus, and several LoD3 models of selected UoC buildings were developed (Willmes et al., 2010). This project already included a web application based on the degree3(Kiehle, 2010) implementation of the OGC Web Perspective View Service (WPVS) for accessing and visualizing the first 3D model of the University of Cologne campus.

As described in section 2.1, a complete toolchain for modeling, georeferencing and streaming or distributing the UoC virtual campus 3D models, is developed within the project. Because we make use of the Cesium (CesiumJS, 2025a) platform and virtual globe for rendering and browsing our 3D campus model, we need to provide our 3D data in OGC 3D Tiles format. The 3DTiler takes care of georeferencing the 3D models and gener-

[1] `https://campusgis2.uni-koeln.de/`
[2] `https://maps.uni-koeln.de/`
[3] `https://virtueller-campus.uni-koeln.de/`

ating the according OGC 3DTiles (Cozzi and Lilley, 2023) for streaming the data to the Cesium virtual globe.

OGC 3D Tiles is a format specifically designed for the efficient streaming and rendering of massive 3D geospatial datasets. Developed by Cesium and later standardized by the Open Geospatial Consortium (OGC), it supports a wide range of 3D data types including buildings, terrain, point clouds, and photogrammetric meshes. One of its key features is its hierarchical level-of-detail (HLOD) structure, which dynamically loads tiles based on camera distance, thereby improving performance in web, mobile, AR and VR environments. This makes it particularly well-suited for digital twins, urban planning, and other interactive geospatial applications.

Cesium offers the Cesium Ion (CesiumJS, 2025b) SaaS product, for hosting 3D models and creating according OGC 3D Tiles for integration into Cesium based applications. Mainly because of financial but also because of digital sovereignty and securing the sustainability of the project infrastructure considerations, we decided against relying on buying the Cesium Ion SaaS product and developed our own tool, which was first tilesetGenerator and is now 3DTiler, for the georeferencing and generation of OGC 3D Tiles part in our 3D project tool chain.

At the time (in late 2023) we were looking for an open-source alternative to Cesium Ion, we did not find any tool that facilitated the georeferencing of 3D models on an interactive map in a user-friendly GUI application. Now there are several good open source tools for creating 3D Tiles available, see the "Awesome 3D Tiles"[4] list by Pirmin Kalberer, but we are still not aware of any tool that allows the georeferencing of the models on a map, like Cesium Ion or now also 3D Tiler allows.

The 3D Tiler implementation is mainly influenced by the works of (Woo et al., 2022) and (Woo et al., 2023), who implemented a tool that is quite similar to 3D Tiler, but was never open sourced, or even offered as a product for licensing it, so we had to develop 3D Tiler. But we will publish a free to use website with access to the 3D Tiler web application and open source the code base accordingly.

## 2. Method, Technology, System Architecture

The 3DTiler tool presented here is a browser-based WebGIS tool developed to simplify the georeferencing of 3D models and generate open-standard OGC 3D Tiles (Cozzi and Lilley, 2023) format files including the transformation parameters for the 3D models. It allows users to upload 3D models in .glTF or .glb format and place them on the Earth's surface using an intuitive user interface. Through direct interaction with a 3D globe, users can set geographic coordinates (latitude, longitude), elevation, and rotation parameters (heading, pitch, and roll). Once positioned, the model is transformed based on these inputs and packaged into a valid 3D Tileset, making it ready for integration into web-based 3D viewers or immersive environments.

The primary role of the 3DTiler is to produce a tileset.json file that includes georeferencing information and transformation matrices in accordance with the OGC 3D Tiles specification. The output has all necessary files arranged in a conventional tileset folder structure along with the original model. This allows seamless use in CesiumJS-based viewers or integration into platforms like Unity through Cesium for Unity.

[4] `https://github.com/pka/awesome-3d-tiles`

As shown in Figure 1, the workflow represents a generalized pipeline that begins with CAD floor plans, continues through 3D model creation and georeferencing, and finally results in a deployable 3D Tileset for use in interactive 3D rendering platforms, frameworks and applications such as Cesium(CesiumJS, 2025a).

The technical implementation of 3DTiler is entirely client-side, built using ReactJS for interface logic and CesiumJS for 3D visualization and globe rendering. Model handling is performed using glTF utilities that parse and display uploaded geometry. All data processing is done locally in the browser, ensuring both ease of deployment and user privacy. No server backend is required, which lowers the barrier to entry and makes the tool accessible for users in academic research, or public sector settings.

Key features of 3DTiler include 3D model positioning, real-time visual feedback during placement, automated generation of the tileset.json structure, and the ability to download it. The interface is designed to be lightweight and user-friendly, supporting both technical and non-technical users. While it was originally developed with focus on university campus 3D models, its general-purpose design allows for broader applications.

### 2.1 System Architecture

The 3DTiler application is tightly integrated with the overall Virtual Campus system architecture and technology stack, described in (Willmes et al., 2024). 3DTiler takes care of the georeferencing and 3D Tiles generation part of the overall project pipeline, which was first implemented in a bachelor thesis (Ganser, 2024) in the context of the CampusGIS2 project, of which the code base was later published on GitHub and Zenodo (Ganser, 2025).

**2.1.1 3D Workflow** The generation of 3D building models for the virtual campus environment relies on a clearly defined pipeline, as seen in Figure 1, that transforms 2D architectural data into spatially accurate and semantically annotated 3D assets.

The input data consists of CAD-based floor plans provided by the University of Cologne's facilities management (Dezernat 5), which are subsequently georeferenced, processed into GIS layers, and modeled as multi-floor 3D objects using Blender. The aim of this process is to create consistent, lightweight, and extensible 3D representations of campus buildings that can be reused in simulation, visualization, or spatial analytics.

The floor plans are georeferenced with the help of a GIS application and reduced to four thematic layers per floor: rooms, doors, circulation elements (e.g. stairs), and floor outlines. Attributes are cleaned, simplified and mapped to a (internal project) schema before being exported into a PostgreSQL/PostGIS database. Only geometry and room IDs are retained for modeling purposes, and later used for referencing of various attribute data, for the according features like rooms, floor levels, building parts and so on from the project databse, from the glTF-based 3D models.

For each floor, the four layers are exported from QGIS as Shapefiles, reprojected to EPSG:25832, and named according to a strict schema (e.g., `303__0__rooms.shp`). Door features are filtered to retain only true thresholds. Export settings are optimized to minimize file size and ensure interoperability with modeling tools.

Modeling is performed in Blender using the BlenderGIS plugin. The outer outline is extruded to full floor height; room geometries are extruded separately and subtracted via Boolean operations to create interior spaces. Door lines are vertically extruded and thickened with a Solidify modifier to cut doorways into the walls. Room IDs remain attached to individual meshes for semantic referencing. Final cleanup includes recalculating normals and merging redundant geometry.

Each floor is exported as a .gltf or .glb file, ensuring compatibility with real-time engines and web-based platforms. The glTF format preserves geometry, object structure, and metadata. Although modeling rules (e.g., height and thickness parameters) are internally defined, the output follows an open standard and is suitable for reuse.

Most steps are automatable via Python scripting in Blender. However, architectural irregularities—such as height offsets or complex stair designs—still require manual refinement. This hybrid approach balances scalability with the flexibility needed for diverse building typologies.

The resulting files can then be loaded into the 3DTiler to position the model and generate 3D tilesets for further use in applications as described in chapter 4.

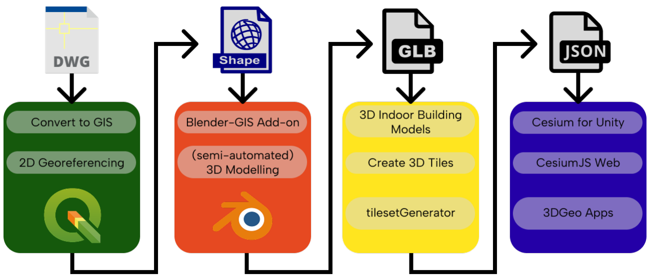

Figure 1. 3D geodata workflow of the Virtual Campus project, from 2D floor plans to interactive 3D geospatial applications.

,

#### 2.1.2 Development of the web application

The technical implementation of 3DTiler is entirely client-side, built using ReactJS (ReactJS, 2025) for interface logic and CesiumJS for 3D visualization and globe rendering. Model handling is performed using glTF utilities, which are part of CesiumJS (CesiumJS, 2025a) that parse and display uploaded geometry. All data processing is done locally in the browser, ensuring both ease of deployment and user privacy. No server backend is required, which lowers the barrier to entry and makes the tool accessible for users without complex server infrastructure at hand, e.g. in private, academic research, small companies or public sector settings.

The 3DTiler web application was developed with the idea of not having any server-side logic (middleware) or software dependencies other than a standard web server. This decision was taken to guarantee sustainability of the application after the end of funding for its development. Thus, the website was designed to consist solely of static files. The 3DTiler georeferencing application can also be deployed within a Docker[5] container, locally or compiled to a PWA and simply uploaded to any webspace.

[5] https://www.docker.com/

## 3. Methodology

In this work, georeferencing refers to positioning 3D building models originally defined in a local coordinate system onto the Earth's surface using global coordinates. Each model starts in its own local metric coordinate system, typically centered at the origin (0,0,0), and is then georeferenced interactively by the user through manual placement on the map.

The interactive map in the 3DTiler application is based on the WGS84 geographic coordinate system (EPSG:4326), where users specify longitude, latitude, and height. These geographic coordinates, along with user-defined orientation values (heading, pitch, and roll), are used to calculate a transformation matrix that places the model accurately on the globe.

This transformation matrix is computed using CesiumJS functions, which convert the geographic coordinates into Earth-Centered, Earth-Fixed (ECEF) space and combine them with the specified orientation. The resulting matrix is then applied to the 3D Tileset, aligning it correctly for visualization in the CesiumJS viewer.

The goal of the methodology is to assign global spatial reference to unreferenced 3D models using intuitive, interactive placement tools without requiring scaling, least-squares adjustment, or multiple ground control points. All changes are updated in real time, providing immediate visual feedback to the user.

The following subsections detail the assumptions, transformation steps, and implementation logic used in the 3DTiler WebGIS application to achieve accurate georeferencing of 3D models.

### 3.1 Input Assumptions

This method operates on 3D models that have been exported at true scale and organized into 3D Tiles format, but whose spatial position and orientation in the global coordinate system have not yet been defined. The dataset must include:

- Models are assumed to be defined in a local coordinate system centered at (0,0,0).
- Users manually specify geographic position (longitude, latitude, height) and orientation (heading, pitch, roll).
- All models are assumed to be exported at true scale; only rigid transformation is required (no scaling).

### 3.2 Coordinate System Transformation

The GCPs, defined in geographic coordinates, are converted to *Earth-Centered, Earth-Fixed (ECEF)* coordinates, also known as EPSG:4978. The corresponding points selected on the 3D Tiles remain in the local coordinate system.

Let:

- $P = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ be the set of corresponding local points.
- $Q = \{\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_n\}$ be the transformed GCPs in ECEF coordinates.

An optional parameter called *altitude offset* can be applied to the Z-component (altitude) of the GCPs to adjust for vertical alignment, particularly for clamping to terrain in platforms like Cesium World Terrain.

### 3.3 Kabsch Algorithm for Optimal Alignment

The Kabsch algorithm, which is typically used to compute the optimal rigid alignment between two 3D point sets, is not implemented in its classical numerical form. Instead, its conceptual steps are implicitly realized in the `3DTiler` WebGIS application through CesiumJS transformation logic and React based components. The implementation follows the algorithm's spirit while leveraging CesiumJS's spatial computation tools.

1. Centroid Approximation: The application uses CesiumJS to convert geographic coordinates (longitude, latitude, height) into Cartesian coordinates. This effectively positions each model in 3D space relative to the globe.

Listing 1. Conversion to Cartesian3 (centroid-like positioning)

```
Cesium.Cartesian3.fromDegrees(longitude,
    latitude, height)
```

2. Relative Centering and Alignment: The 3DTiler computes a relative transformation by applying the inverse of the root tileset's transformation matrix to the model's local transformation. This ensures correct placement of the building within the tileset hierarchy.

Listing 2. Relative transformation calculation

```
const inverseMainTransform = Cesium.Matrix4.
    inverse(
mainTileset.root.transform,
new Cesium.Matrix4()
);
const relativeMatrix = Cesium.Matrix4.multiply(
inverseMainTransform,
modelMatrix,
new Cesium.Matrix4()
);
```

3. Rotation Matrix via Heading-Pitch-Roll: In the application, rotation is defined through user input for heading, pitch, and roll. These values are used to construct a transformation matrix via CesiumJS functions, which determines the model's orientation on the globe.

Listing 3. HPR to rotation matrix

```
const hpr = new Cesium.HeadingPitchRoll(heading
    , pitch, roll);
const modelMatrix = Cesium.Transforms.
    headingPitchRollToFixedFrame(
position,
hpr
);
```

4. Applying the Rotation Matrix: Rotation is directly created through the 'HeadingPitchRoll' object, reflecting user input from the GUI.

Listing 4. Constructing HPR object

```
new Cesium.HeadingPitchRoll(heading, pitch,
    roll)
```

5. Applying the Full Transformation Matrix: The final transformation combining both rotation and translation is applied to the tileset as a relative matrix.

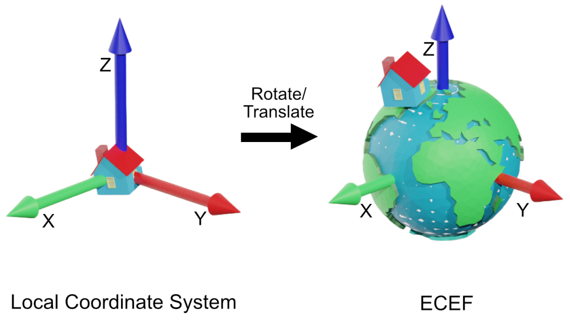

Figure 2. Georeferencing 3D Tiles by transformation. Figure adapted after (Woo et al., 2022).

Listing 5. Applying the transformation

```
tileset.transform = relativeMatrix;
```

All transformation logic is encapsulated in the function `createTransformationMatrixRelativeToMainTileset()` found in the `ModelPositioning.jsx` file. This function is called automatically whenever a user adjusts spatial parameters via the interface, ensuring that changes are reflected immediately within the CesiumJS viewer.

### 3.4 Limitations and Design Decisions

The transformation implemented in the 3DTiler is a rigid-body transformation, consisting of six parameters (three for translation and three for rotation). Scaling is not applied, as the 3D models are assumed to be exported at true scale from the modeling environment (e.g., Blender or Rhino), as per the input assumptions.

At present, the coordinate validity is not programmatically validated, instead, the application relies on real-time visual feedback within the CesiumJS viewer to guide and verify correct placement. Users can interactively adjust positioning and visually confirm alignment.

While the method is inspired by the basic ideas of the Kabsch algorithm, it does not support using more than three Ground Control Points (GCPs) to calculate a more precise fit, such as with a least squares adjustment. The current design favors simplicity and intuitive usability in a browser-based interface.

In future versions, we plan to improve the system by allowing users to use more control points and apply a least squares approach. This would help increase accuracy and give users information about how well the model fits, similar to what has been done in more advanced systems like the one presented in (Hakim et al., 2024).

### 3.5 Embedding the Transformation in 3D Tiles

The computed rotation $R$ and translation $T$ are merged into a $4 \times 4$ homogeneous transformation matrix. This matrix is inserted as the `transform` property of the root tile in the `tileset.json` file. This ensures that all vertex coordinates are correctly reprojected from the local coordinate system into the global ECEF system. As shown in Figure 2, this transformation repositions the 3D model from its local origin to its georeferenced global location.

The transformation matrix has the form:

$$M = T_q \cdot R \cdot T_p$$

Where:

- $T_p$ and $T_q$ are the translation matrices used to shift the local and global point sets to their respective origins and then back after rotation.

### 3.6 Accuracy Assessment

After transformation, the error for each GCP by calculating the Euclidean distance between the transformed local point and the corresponding ECEF GCP is calculated. This serves as a validation metric for the quality of the georeferencing process.

## 4. Results

In this section the GUI and the main features of the 3DTiler web app will be explained.

### 4.1 Web Application

The WebGIS application developed in this work referred to as the *3DTiler* serves as an interactive tool for creating, positioning, and exporting georeferenced 3D Tiles based on 3D building models in `.glTF` or `.glb` format. Built with `React` and `CesiumJS`, it offers a graphical user interface (GUI) that facilitates intuitive manipulation of building models and generation of hierarchical tileset structures for WebGIS platforms.

The 3DTiler application is accessible for public use under:
`https://3dtiler.uni-koeln.de`

To support the visualization and interactive georeferencing of 3D building models, 3DTiler automatically generates a main tileset representing the entire campus. Each building added by the user is inserted as a child tile. To ensure each building is uniquely identifiable, the application embeds the building ID as metadata in the child tile, using a predefined file naming convention. The root tileset is placed at the campus center, and building positions are calculated relative to this origin.

The graphical user interface (see Figure 3) enables users to upload 3D models in `.glTF` or `.glb` format. Initially, models appear at the center of the campus, as they lack spatial transformation. Bounding volumes are automatically calculated for rendering. The user then specifies the geographic location (latitude, longitude, height) and orientation (heading, pitch, roll) for each model. Cardinal direction buttons are also provided to adjust positioning without entering numeric values.

Any change to these spatial parameters triggers a real-time update of the transformation matrix. This matrix is calculated using CesiumJS and applied to the tile, ensuring immediate visual feedback. According to the 3D Tiles specification (Cozzi and Lilley, 2023), the transformation is stored as a 4×4 affine matrix in column-major order, converting local tile coordinates to the parent or global coordinate system.

To support more complex scenarios, such as multi-level buildings, users can upload individual floors as separate files. Based on file naming conventions, floors are automatically grouped under the correct building tile. Each floor is added as a child tile, and floor-level metadata is stored to maintain distinct representation and future filtering capabilities.

#### 4.1.1 User Interface Overview

Upon launching the application, the user is presented with:

- A 3D CesiumJS globe.
- A control panel in the top-left corner for file upload and building management.
- A toggle button in the top-right to switch between OSM and satellite basemaps.

#### 4.1.2 Location Search and Repositioning Root Tileset

The WebGIS interface includes a location search field that allows users to quickly navigate to any place name. Upon entering a valid query, the CesiumJS viewer smoothly transitions the camera to the specified location. Additionally, a dedicated button enables users to reposition the root tileset so that it aligns with the center of the current view. The root tileset needs to be repositioned before selecting the models to upload.

#### 4.1.3 Model Upload and Classification

Models can be uploaded using the `Select file/s` button. The application automatically distinguishes between full buildings and floors based on file naming conventions such as `building_303.glb` or `building_303_floor_1.glb`. The files are grouped accordingly into building objects and their associated levels (floors).

#### 4.1.4 Building List Panel

After upload, each detected building appears in the control panel with:

- A unique identifier (e.g., 303).
- A `Select` button for opening the transformation tools.
- A `Delete` button to remove the building from the session.

#### 4.1.5 Model Positioning Interface

Clicking the `Select` button opens a panel containing input fields for spatial transformation:

| Input Field | Description |
|---|---|
| Latitude / Longitude | Geographic coordinates for model placement on the globe. |
| Height | Altitude in meters. |
| Heading | Rotation around the vertical (Z) axis. |
| Pitch | Rotation around the east-west axis (Y). |
| Roll | Rotation around the north-south axis (X). |

Table 1. Transformation input fields for model positioning

Each input can be modified via mouse scroll, keyboard arrows, or manual typing. Four additional buttons representing cardinal directions (N, S, E, W) enable intuitive movement without numerical input. Changes to any of the positioning inputs such as latitude, longitude, or orientation parameters immediately trigger a recalculation of the transformation matrix using CesiumJS. These updates are applied in real time, providing immediate feedback within the 3D viewer and allowing precise interactive positioning of models relative to the root tileset.

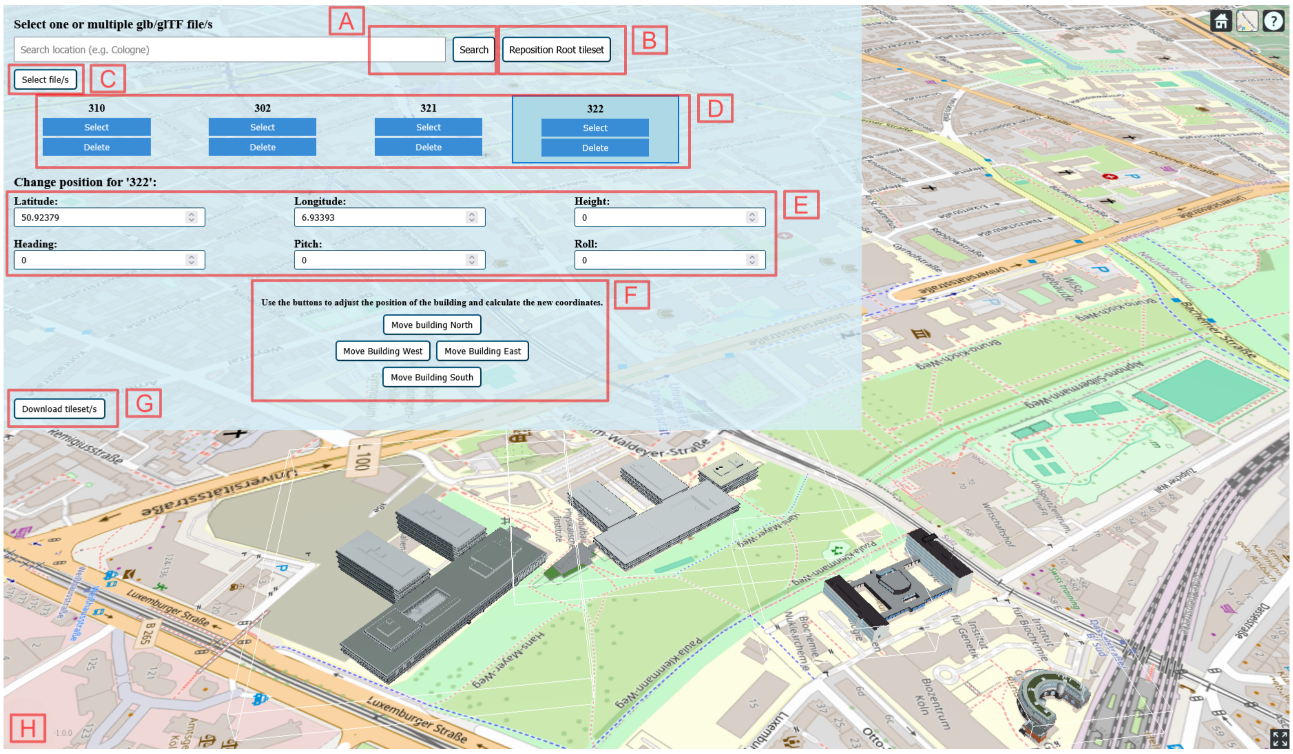

Figure 3. Screenshot of the 3DTiler web application, georeferencing four 3D building models on the UoC Campus. Labeled GUI elements (A–H) correspond to the components described in Table 2.

**4.1.6 Tileset Generation and Export** Clicking the `Download tileset/s` button initiates the tileset export process. The application first replaces internal temporary object URLs (used for local model previews) with persistent, relative filenames. It then generates a main campus tileset file (`tileset_campus.json`) along with individual tileset files for each building (e.g., `tileset_303.json`).

| Label | Component | Functionality |
|---|---|---|
| A | Location Search Field | Centers the view on the specified geographic location. |
| B | Reposition Root Tileset Button | Aligns the root tileset to the center of the current view. |
| C | File Upload Button | Imports `.glTF` or `.glb` models. |
| D | Building List Panel | Displays uploaded models with select/delete options. |
| E | Positioning Panel | Inputs for lat/lon/height/heading/pitch/roll. |
| F | Cardinal Direction Tools | Move model with N/S/E/W buttons. |
| G | Download Button | Exports tilesets in 3D Tiles format. |
| H | CesiumJS Viewer | Interactive 3D globe for positioning buildings. |

Table 2. GUI components of the 3DTiler web application, labeled according to Figure 3

**4.1.7 Summary of GUI Features** The presented GUI provides an intuitive and efficient workflow for creating georeferenced 3D Tilesets suitable for WebGIS platforms such as the UoCMaps application. Figure 3 shows a screenshot of the graphical user interface (GUI) of the application. The labeled elements in Figure 3 correspond to the components summarized in Table 2.

## 5. Conclusion and Outlook

This paper presents the development and application of 3DTiler, a web-based tool designed to georeference and convert 3D building models into OGC-compliant 3D Tiles. The tool allows users to interactively position .glb or .glTF models on a map and automatically generate structured 3D Tilesets for use in platforms like CesiumJS. The main contribution of this work is a reusable, user-friendly, and open workflow for creating standardized 3D geospatial content—streamlining what was previously a project-specific, semi-manual process. 3DTiler provides an accessible alternative to complex GIS pipelines and contributes to broader goals in 3D geoinformation and digital twin development.

In the context of the CampusGIS2 and Virtual Campus projects at the University of Cologne, the tool plays a crucial role in enabling the integration of 3D building models into a web-based site plan. The tilesets produced using 3DTiler support metadata-driven functions such as dynamic zooming to specific buildings, interaction with user-selected elements, and structured loading of sub-tilesets based on view-dependent logic. It replaces the need for Cesium ion in the current workflow by offering full control over tileset structure, georeferencing, and deployment. This supports not only visualization but also deeper semantic interaction within the WebGIS environment.

Looking ahead, the tool is intended to be released as open source, making it available for other institutions and developers. Future enhancements may include support for instanced models, batch processing, and hierarchical LOD within buildings.

By incorporating configurable parameters and modular export logic, 3DTiler can evolve into a general-purpose geospatial authoring tool—bridging the gap between 3D modeling and standardized geo-web deployment. Its development marks a step toward more transparent, open, and interoperable 3D geoinformation workflows.

## References

Baaser, U. V., 2010. Das CampusGIS der Universität zu Köln - webgestützte Geodatendienste für raumbezogene Anwendungen. PhD thesis, Universität zu Köln.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., Çöltekin, A., 2015. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4), 2842–2889. https://www.mdpi.com/2220-9964/4/4/2842.

CesiumJS, 2025a. 3D geospatial visualization for the web. `https://cesium.com/platform/cesiumjs/`.

CesiumJS, 2025b. Create and host 3D content in the cloud. `https://cesium.com/platform/cesium-ion/`.

Cozzi, P., Lilley, S., 2023. 3D Tiles Specification. Technical report, Open Geospatial Consortium. `http://www.opengis.net/doc/cs/3D-Tiles/1.1`.

Ganser, C. S., 2024. 3D-Gebäudemodelle für den Lageplan der Universität zu Köln - Entwicklung einer WebGIS-Applikation zur Erstellung von 3D Tiles. BSc Thesis. University of Cologne, unpublished.

Ganser, C. S., 2025. tilesetGenerator - WebGIS-App to georeference 3D building models and create 3D Tiles tilesets. `https://zenodo.org/records/15189767`.

Hakim, A., Arroyo Ohori, K., van der Vaart, J., El Yamani, S., Stoter, J., 2024. Enhancing Georeferencing of IFC Models through Surveyed Points Integration. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W11-2024, 41–47. https://isprs-archives.copernicus.org/articles/XLVIII-4-W11-2024/41/2024/.

Kiehle, Christian; Fitzke, J. S. M., 2010. deegree 3 - architektur und anwendungsfelder. *Angewandte Geoinformatik 2010 - 22. AGIT-Symposium*, VDE Verlag.

ReactJS, 2025. A JavaScript library for building user interfaces. `https://reactjs.org/`.

Willmes, C., Baaser, U., Volland, K., Bareth, G., 2010. Internet based distribution and visualization of a 3D model of the University of Cologne Campus. *3rd ISDE Digital Earth Summit, 12-14 June, 2010, Nessebar, Bulgaria*. `https://cwillmes.de/media/CampusGIS-3D_Willmes_DigitalEarth2010.pdf`.

Willmes, C., Canals, L., Ganser, C., Mennecke, D., Phillipps, F. L., Pietsch, A.-K., Reichenau, T., Reuhl, E., Schildkamp, P., Stempel, M., Wickeroth, D., Wieners, J., Eide, Ø., 2024. The Technology Stack and System Architecture of the University of Cologne Virtual Campus. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVIII-4/W11-2024, 153–160. https://isprs-archives.copernicus.org/articles/XLVIII-4-W11-2024/153/2024/.

Woo, K., Onsen, A., Kim, W., 2022. Georeferencing 3D Tiles Generated from Photogrammetry-Derived Mesh Using Ground Control Points. *Journal of Geographic Information System*, 14(5), 430–443. https://dx.doi.org/10.4236/jgis.2022.145023. Number: 5 Publisher: Scientific Research Publishing.

Woo, K., Onsen, A., Kim, W., 2023. Implementation of a 3D WebGIS for Dynamic Geo-Referencing of 3D Tiles on the Virtual Globe. *Journal of Geographic Information System*, 15, 440-457. https://dx.doi.org/10.4236/jgis.2023.155022.