

INTERACTIVE TRANSFORMATION OF LAYERS IN 3D TILES FORMAT WITHIN A WEB GIS APPLICATION

Markéta Nedělová¹, Ivo Pisařovic¹

¹Department of Informatics, Faculty of Business and Economics, Mendel University in Brno, Zemědělská 1, 613 00 Brno, Czech Republic

DOI: https://doi.org/10.11118/ecos2026_025

ABSTRACT

The paper deals with the design and implementation of an extension to the Spatial Lens web application, which is used to prepare 3D educational materials displayed in virtual reality. The subject of the extension is editing spatial layers stored in the 3D Tiles format, specifically their displacement, rotation and scaling directly in the scene preview, which eliminates the need to use external tools. Three-dimensional geodata is not always perfectly prepared for a specific scene and may have a different axis orientation or incorrect scale. The entire 4×4 transformation matrix is not transferred between the frontend and backend, but a more economical data structure containing the position (longitude, latitude and altitude), rotation and scale. This solution reduces cumulative rounding errors, increases accuracy for large numbers, and facilitates input validation. The backend recalculates the resulting matrix each time from the beginning from the saved original matrix and a stable pivot point, thus ensuring idempotence of repeated saves. The usability of the extension was verified through scenario-based user testing with ten participants. The results confirm the usability of the tool while identifying areas for further improvement of the user experience.

Keywords: 3d tiles, cesiumjs, web gis application, spatial data transformation

JEL Code: C88, L86

1 INTRODUCTION

The development of geospatial technologies has led to the wider use of three-dimensional geographic data in a number of areas, such as spatial planning, environmental monitoring and navigation systems (Biljecki *et al.*, 2015). The ability to visualize and manipulate 3D geographic data allows for a more realistic and detailed representation of the real world, which supports decision-making and analysis (Ketzler *et al.*, 2020). This is accompanied by the growing importance of web geographic information systems (Web GIS), which make complex spatial datasets accessible directly in the browser without the need to install specialized desktop software; map creation on the web has undergone rapid development in recent decades from simple static map services to fully-fledged interactive systems (Veenendaal *et al.*, 2017).



This article is licensed under CC BY-SA 4.0 Attribution-ShareAlike 4.0 International license. <https://creativecommons.org/licenses/by-sa/4.0/deed.en>

One such platform is Spatial Lens, formerly known as Spatial Data Service, designed for preparing 3D educational courses that are subsequently displayed in virtual reality (Chylíková, 2025). The system builds on the work of Spatial Data Browser for Metaverse, which was created as part of a bachelor's thesis by Mitter (2022) as a client-server application for visualizing 3D spatial data in virtual reality using the 3D Tiles format and open web standards. Chylíková (2025) subsequently expanded this system to support the creation of educational materials.

Three-dimensional geodata and 3D models are not always perfectly prepared for a specific scene. For example, they may have different axis orientations or incorrect scales. If editing is not available directly in the application, the user must use an external tool, which complicates the entire workflow. The ability to move, rotate, and scale a layer allows you to adapt the scene to a specific learning goal. The 3D Tiles standard is designed primarily for streaming and visualization, not for editing geometry (Open Geospatial Consortium, 2023). However, it is possible to move, rotate, or scale an entire tileset by editing the root transformation matrix in the `tileset.json` file.

The main objective of the paper is to describe the design and implementation of an extension to the Spatial Lens application that allows users to perform editing operations with spatial layers (shift, rotate, and scale 3D objects) directly in the application, with a live preview of the scene. The usability of the proposed solution was subsequently evaluated through user testing.

2 METHODOLOGY AND DATA

The CesiumJS library (CESIUM, 2026) uses several coordinate systems to work with the position of objects. The geodetic system determines the position of a point using longitude, latitude, and altitude, taking into account the fact that the Earth is not a perfect sphere. The Earth-Centered, Earth-Fixed (ECEF) system describes the position in Cartesian coordinates relative to the center of a reference ellipsoid and rotates with the Earth, so that a point fixed on the surface has constant coordinates in this system (Van Sickle, 2017). Because the global Z axis points to the North Pole and does not coincide with the local vertical at any point, the East-North-Up (ENU) system is introduced for local orientations and transformations. In CesiumJS, it is generated by the `Transforms.east-NorthUpToFixedFrame` function, which returns a 4×4 transformation matrix.

In computer graphics, linear transformations in three-dimensional space are usually represented by 4×4 matrices in homogeneous coordinates and are divided into three basic types: translation, rotation, and scaling. Rotation can be expressed in several ways — using a transformation matrix, a triple of Euler angles, or quaternions. Euler angles are intuitive for users, but they can lead to a phenomenon called gimbal lock, in which one degree of freedom is lost. Quaternions prevent this problem and serve as a robust intermediate step in the calculations (Žára *et al.*, 2005). However, for the user interface, Euler angles were chosen because they provide an intuitive way for users to input precise numerical values via sliders.

The entry point for each tileset is the `tileset.json` file, whose `root.transform` property contains a 4×4 matrix stored in column order that places the local coordinate system of the model in the ECEF system (Open Geospatial Consortium, 2023). The primary format for the content of the tiles is the native glTF 2.0 in the 3D Tiles 1.1 specification (The Khronos Group, 2021). Due to the enormous scale of the virtual globe, 32-bit floating-point numbers cannot be reliably used for coordinates; therefore, CesiumJS operates at the processor level with 64-bit precision and employs techniques such as Relative-to-Center and Relative-to-Eye (Cozzi and Ring, 2011).

The above features lead to a key design decision: the backend does not receive the entire 4×4 matrix from the frontend, but the more efficient data structure shown in Listing 1, which contains position, scale, and rotation.

```
"transform": {
  "position": {
    "longitude": 16.613117421768894,
    "latitude": 49.19078110708189,
    "altitude": 220.1149262056975
  },
  "scale": { "x": 1, "y": 1, "z": 1 },
  "rotation": { "yaw": 90, "pitch": 0, "roll": 0 }
}
```

Listing 1 Transform data structure designed for communication between frontend and backend.

There are four reasons for this solution. The structure can be easily extended with future features, such as pivot point selection. It also prevents the accumulation of errors: if the frontend were to repeatedly multiply the current matrix and send it for storage, this would lead to gradual distortion (skew or rotation drift) after several saves. The third reason is accuracy for large numbers — ECEF coordinates are in the millions of meters, and transmitting a 4×4 matrix in JSON could lead to a loss of decimal precision, while transmitting geographic coordinates is more secure. The last reason is easier validation and security of inputs on the backend side.

The application is built on the Vue.js framework (frontend) and FastAPI (backend), the MongoDB database is used for data storage. Data validation on the backend is provided by the Pydantic library, and the vue-cesium component library is used on the frontend to work with CesiumJS.

The usability of the extension was verified by a scenario usability test combined with the think-aloud method. The method is based on cognitive psychology (Ericsson and Simon, 1980) and has become a standard in the field of usability testing (Nielsen, 1994). Evaluating the usability of web mapping applications through a combination of expert judgment and user testing is an established procedure in this domain (Nivala *et al.*, 2008). Ten people participated in the testing as part of our research; the total time to complete the task was measured, and success was monitored qualitatively and supplemented with a questionnaire.

3 RESULTS

The proposed solution consists of two tightly coupled parts. On the backend, a dedicated computation pipeline translates the compact Transform structure into a precise 4×4 root matrix. On the frontend, the same transformation parameters drive a real-time scene preview, giving the user immediate visual feedback. The following subsections describe both layers in detail, followed by the results of the usability evaluation.

3.1 Calculating the transformation on the backend

The main computational logic is located on the backend in the `geo_utils.py` module, whose functions are called when the `PUT /layers/{id}` endpoint is queried. On the first save, the root matrix `root.transform` generated by the processing pipeline is stored in the `asset.extras.originalRoot`

array and is never overwritten on subsequent saves. This value serves as a stable reference point representing the original position of the model. Each subsequent save recalculates the entire transformation matrix from the beginning from this reference, rather than incrementally modifying the previously saved matrix. This prevents the accumulation of rounding errors that would otherwise manifest as gradual geometric deformations.

A prerequisite for proper rotation and scaling is the determination of a stable pivot point. Without it, rotation around an arbitrary origin would cause the model to “orbit” through space, rather than rotating in place. The pivot point is defined as the center of the enclosing body of the tile set expressed in ECEF coordinates, calculated once from **originalRoot** and constant throughout the lifetime of the set. The resulting matrix is then composed according to relation (see Listing 2), where **ENU(pos)** denotes the ENU matrix at the target position, **HPR** the rotation matrix, **Scale** the scale matrix, and **bsCenter** the center of the enclosing body:

```
new_root = ENU(pos) × HPR × Scale × inv(ENU(bsCenter)) × originalRoot
```

Listing 2 Composing the resulting matrix.

This logic is written in the **geo_utils.py** module as shown in Listing 3.

```
new_root = ( +
    @ enu_target
    @ rot_4x4
    @ scale_4x4
    @ np.linalg.inv(enu_bs_center)
    @ original_root
)
data["root"]["transform"] = new_root.flatten(order='F').tolist()
```

Listing 3 Folding the resulting root matrix on the backend (“full-bake” method).

A less obvious complication arises from the region type wrapping body, which uses geographic coordinates instead of Cartesian coordinates local to the tiles. CesiumJS derives tile visibility from these coordinates independently of **root.transform**, so after saving a new position, the culling system continues to evaluate visibility at the original position and the model sometimes disappears from view. The solution is to replace the region definition with an oriented wrapping box expressed in the local tile space, which is correctly transformed using **root.transform** according to the 3D Tiles specification (Open Geospatial Consortium, 2023). This conversion only happens once, on the first save, when the region type wrapping body is detected.

3.2 Real-time preview on the frontend

The frontend dynamically updates the 3D preview of the layer based on how the user interacts with the controls. When the tileset is finished loading, the **onReady** callback explicitly downloads the **tileset.json** file from the server. This additional request is necessary because the CesiumJS rendering engine optimizes the tile hierarchy internally, often omitting its own extras array, which holds the original transformation parameters.

To make the operations idempotent, the model matrix is recomputed from scratch from the full stored transformation state (position, rotation, and scale) for each change using the

`calculateModelMatrix` function (Listing 4). Because the rendered state is always a pure function of the stored values, rather than an accumulation of incremental changes, repeated save-load cycles produce the same visual result. This approach eliminates the initial problem of errors gradually accumulating as the scale changes.

```
export const calculateModelMatrix=(transform: Transform): Matrix4 | null => {
  const targetPos=Cartesian3.fromDegrees(
    transform.position.longitude,
    transform.position.latitude,
    transform.position.altitude
  );
  const newENU=Transforms.eastNorthUpToFixedFrame(targetPos);
  const scaleMatrix=Matrix4.fromScale(
    new Cartesian3(transform.scale.x, transform.scale.y, transform.scale.z),
    new Matrix4()
  );
  const hpr=new HeadingPitchRoll(
    CesiumMath.toRadians(transform.rotation.yaw),
    CesiumMath.toRadians(transform.rotation.pitch),
    CesiumMath.toRadians(transform.rotation.roll)
  );
  const rotationMatrix4=Matrix4.fromRotation(
    Matrix3.fromHeadingPitchRoll(hpr, new Matrix3()),
    new Matrix4()
  );
  const localTransform=Matrix4.multiply(
    rotationMatrix4, scaleMatrix, new Matrix4()
  );
  return Matrix4.multiply(newENU, localTransform, new Matrix4());
};
```

Listing 4 Calculating the model matrix on the frontend for real-time preview.

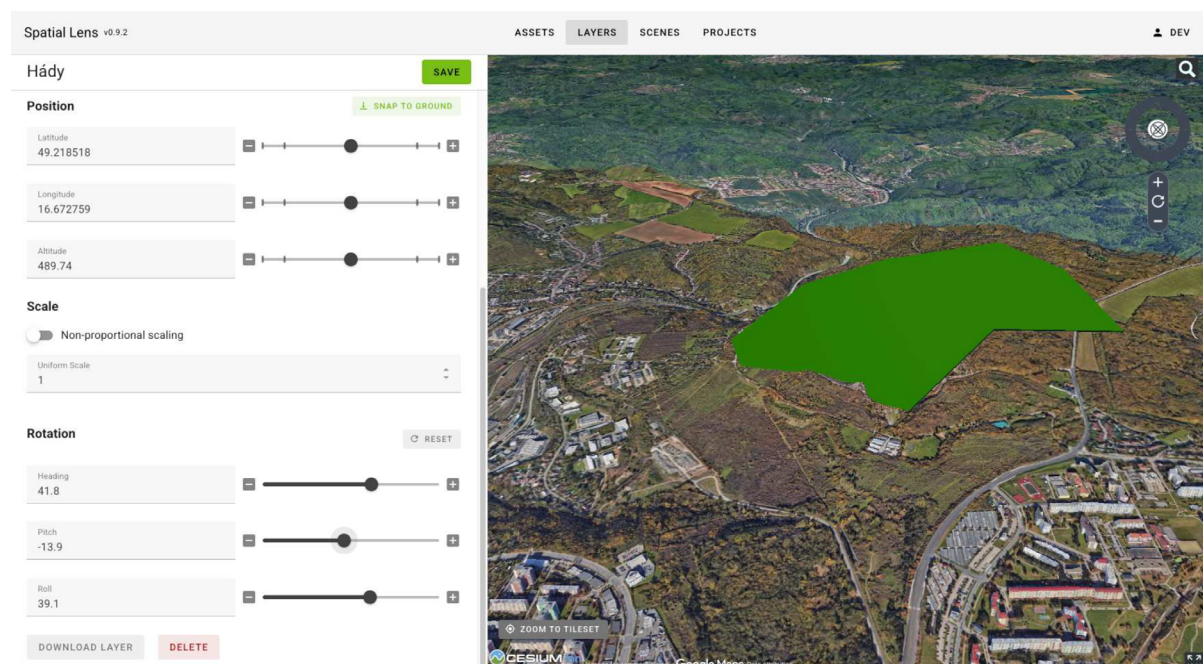


Fig. 1: The resulting implementation of a layer detail with position, scale and rotation settings and a 3D scene preview. (Nedělová, 2026)

The transformation control user interface (see Figure 1) includes separate components for translation, rotation, and scaling. For precise adjustments, a custom **VariableSlider** component has been created that supports dynamic step sizes — both linear and logarithmic behavior based on the distance of the slider from the center. This allows the user to quickly make coarse adjustments as well as fine micro-adjustments. The **flyToLayer** function dynamically calculates a new bounding sphere for objects outside the originally defined area from the transformed coordinates and the changed radius to correctly point the camera at the current spatial extent.

3.3 User testing results

Ten people (aged 26 to 40) participated in the test. Six of them were educators from primary school to university. The remaining four were technically proficient users with experience with GIS applications. All participants successfully completed the assigned tasks, in some cases with minor assistance. The duration of the testing phase ranged from 15.22 to 42.31 minutes, with an average of 27.33 minutes and a median of 25.90 minutes. The upper limit fell on the first session, during which several critical problems were identified and subsequently eliminated.

The evaluation of transformation functions and related interface aspects from the questionnaire is summarized in Table 1. Columns 1 to 5 indicate the number of participants who chose a given evaluation on a five-point Likert scale (1 = strongly disagree, 5 = strongly agree).

The results show that the best rated were rotation adjustment (4.3) and confidence in future work efficiency (4.5). Scale adjustment received 3.8 points; position adjustment (3.6) and interface orientation (3.4) received the lowest ratings with a greater variance of responses. Qualitative findings indicated that participants were unsure of which direction the layer would move, as it depended on the current camera orientation; most had to move the slider and observe the result. Three participants preferred the increase and decrease buttons to dragging the slider. Three participants experienced incorrect caching behavior, where the layer did not reflect the changes after saving, and a hard page reload was required. The most frequently mentioned wish was the ability to edit layers directly in the scene preview.

Tab. 1 Evaluation of questionnaires related to transformations (selection from results).

Questionnaire item	1	2	3	4	5	Average
Future work efficiency	0	0	1	3	6	4.5
Rotation adjustment	0	0	0	7	3	4.3
Scaling	1	1	1	3	4	3.8
Position adjustment	0	2	2	4	2	3.6
Interface orientation	0	3	2	3	2	3.4

4 DISCUSSION AND CONCLUSIONS

The aim of the paper was to describe the design and implementation of an extension to the Spatial Lens application that allows moving, rotating and scaling layers in 3D Tiles format directly in the scene preview. The key design decision was to transfer the economical Transform structure instead of the entire 4×4 matrix and to recalculate the resulting matrix on the backend from scratch, always from the saved original matrix and a stable pivot point. This “full-bake” approach has proven itself in practice: repeated save-load cycles led to identical visual results, thus confirming the idempotence of operations and eliminating the accumulation of round-off errors and initial distortion when changing the scale. At the same time, it was necessary to solve a less obvious complication with the region-type wrapping body, which led to the model disappearing from the view after its movement.

The results of the user testing confirm the usability of the tool. Of the transformation functions, rotation was rated best (4.3), followed by scaling (3.8) and position adjustment (3.6). The high rating of confidence in future work efficiency (4.5) indicates that participants expected to complete the task significantly faster in subsequent sessions, which indicates a manageable learning curve. However, the lower and more scattered ratings of position adjustment and orientation in the interface indicate room for improvement of the user experience, especially regarding the dependence of the direction of movement on the camera orientation. The finding that interactive work with 3D objects places different demands on users than working with static displays is consistent with research on the differences between interactive and static 3D maps (Herman *et al.*, 2018).

The testing resulted in several recommendations for further development. It would be beneficial to add an interface with a manipulator (gizmo) for direct manipulation of the object, preset camera modes, support for keyboard shortcuts for placing layers, and a snap to terrain function. Although the sample of ten participants was sufficient for identifying key usability issues (Nielsen, 1994), a larger and more diverse group would strengthen the generalizability of the findings. The developed extension is functional, meets the set goals, and represents a solution that can be built on; future development should therefore focus on implementing improvements based on feedback from testing.

Acknowledgements

This paper was supported by the project CZ.02.1.01/0.0/0.0/16_017/0002334 Research Infrastructure for Young Scientists, this is co-financed from Operational Programme Research, Development and Education, and also supported by grant No. IGA24-PEF-TP-003 (Using metaverse technologies for working with spatial data) of the Internal Grant Agency FBE MENDELÚ.

Anthropic Claude was used to summarize a part of the final thesis (Nedělová, 2026) and transform it into the form of this compact paper. The paper was then reviewed and fine-tuned by the author. The author is responsible for the correctness of the final text.

REFERENCES

- BILJECKI, F., STOTER, J., LEDOUX, H., ZLATANOVA, S., ÇÖLTEKİN, A. 2015. Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*. 4(4), 2842–2889.
- CESIUM. 2026. *CesiumJS Documentation*. Exton, PA: Cesium GS, Inc. <https://cesium.com/learn/cesiumjs/ref-doc/>
- CHYLÍKOVÁ, E. 2025. *Supporting education through virtual reality and geospatial data*. Diploma thesis. Brno: Mendel University in Brno, Faculty of Business and Economy.
- COZZI, P., RING, K. 2011. *3D Engine Design for Virtual Globes*. 1st Edition. Boca Raton, FL: CRC Press.
- ERICSSON, K. A., SIMON, H. A. 1980. Verbal reports as data. *Psychological Review*. 87(3), 215–251.
- HERMAN, L., JUŘÍK, V., STACHOŇ, Z., VRBÍK, D., RUSSNÁK, J., ŘEZNÍK, T. 2018. Evaluation of User Performance in Interactive and Static 3D Maps. *ISPRS International Journal of Geo-Information*. 7(11), 415. <https://doi.org/10.3390/ijgi7110415>
- KETZLER, B., NASERENTIN, V., LATINO, F., ZANGELIDIS, C., THUVANDER, L., LOGG, A. 2020. Digital Twins for Cities: A State of the Art Review. *Built Environment*. 46(4), 547–573.
- MITTER, M. 2022. *Spatial Data Browser for Metaverse*. Bachelor thesis. Brno: Mendel University in Brno, Faculty of Business and Economy.
- NEDĚLOVÁ, Markéta. 2026. *Editing Spatial Data Using a Web Application*. Diploma thesis. Brno: Mendel University in Brno.
- NIELSEN, J. 1994. *Usability Engineering*. San Francisco, CA: Morgan Kaufmann.
- NIVALA, A.-M., BREWSTER, S., SARJAKOSKI, LT 2008. Usability Evaluation of Web Mapping Sites. *The Cartographic Journal*. 45(2), 129–138. <https://doi.org/10.1179/174327708X305120>
- OPEN GEOSPATIAL CONSORTIUM. 2023. *3D Tiles Specification 1.1*. Arlington, VA: Open Geospatial Consortium. <http://www.opengis.net/doc/cs/3D-Tiles/1.1>.
- THE KHRONOS GROUP. 2021. *glTF 2.0 Specification: Version 2.0.1*. Beaverton, OR: The Khronos Group. <https://registry.khronos.org/glTF/specs/2.0/glTF-2.0.html>
- VAN SICKLE, J. 2017. Two Coordinate Systems. In: *Basic GIS Coordinates*. 3rd Edition. Boca Raton, FL: CRC Press, pp. 47–68.
- VEENENDAAL, B., BROVELLI, M. A., LI, S. 2017. Review of Web Mapping: Eras, Trends and Directions. *ISPRS International Journal of Geo-Information*. 6(10), 317. <https://doi.org/10.3390/ijgi6100317>
- ŽÁRA, J., BENEŠ, B., SOCHOR, J., FELKEL, P. 2005. *Modern computer graphics*. 2nd Edition. Brno: Computer Press.

Contact information

Markéta Nedělová: e-mail: xnedelov@mendelu.cz

Ivo Pisařovic: e-mail: research@pisarovic.cz