Scientific Research Publishing

Journal of Geographic Information System, 2023, 15, 440-457
https://www.scirp.org/journal/jgis
ISSN Online: 2151-1969
ISSN Print: 2151-1950

Implementation of a 3D WebGIS for Dynamic Geo-Referencing of 3D Tiles on the Virtual Globe

Kyongil Woo, Adrian Onsen, Wonsok Kim

Institute of Remote Sensing and Geoinformatics, Academy of Science, Pyongyang, Democratic People's Republic of Korea
Email: wugis1219@gmail.com

How to cite this paper: Woo, K., Onsen, A. and 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://doi.org/10.4236/jgis.2023.155022

Received: July 3, 2023
Accepted: October 6, 2023
Published: October 9, 2023

Abstract

Needs for real-time interactive visualization of 3D Tiles for massive 3D content on the web-based virtual globe is rapidly increasing, and to achieve this goal, 3D Tiles needs to be correctly geo-referenced to other geospatial data on a web-based virtual globe. It is possible to generate 3D Tiles from different kinds of spatial data through various software tools. However, due to various factors the 3D Tile datasets are often poorly or not at all geo-referenced. To tackle this issue, we propose a new 3D WebGIS framework that facilitates dynamic geo-referencing 3D Tiles on the CesiumJS virtual globe.

Keywords

3D Tiles, 3D WebGIS, CesiumJS, Geo-Referencing, Virtual Globe

1. Introduction

3D Tiles is the open specification for streaming massive heterogeneous 3D geospatial datasets [1]. It is a widely adopted OGC Community Standard designed to improve the streaming and rendering performance of massive 3D datasets [2], and more and more software platforms have added support for the open standard.

3D Tiles has gained widespread adoption as the preferred way for streaming massive 3D data on the web [3] and received a lot of attention in a wide range of scientific and application domains such as architecture, engineering, and construction (AEC) industry, aerospace, military mission planning, agriculture, real estate, smart cities, BIM/GIS integration, sports, entertainment, tourism, and cultural heritage management.

There are some web-based 3D Tiles rendering frameworks and libraries such as CesiumJS [4], iTowns [5], deck.gl [6], 3DTilesRendererJS [7], three-loader-

3dtiles [8], 3d-tiles [9], and mapbox-3dtiles [10]. However, CesiumJS is the most suitable to visualize 3D Tiles on the web. CesiumJS is an open-source JavaScript library for creating world-class 3D globes and maps with the best possible performance, precision, visual quality, and ease of use [4], and it has grown into a leading three-dimensional 3D WebGIS platform adopting state-of-the-art WebGL technology [11].

The virtual globe of the CesiumJS is modeled accurately using the WGS84 Ellipsoid [12], and all kinds of spatial data that the CesiumJs supports are finally given in the earth-centered, earth-fixed (ECEF) [13] reference frame (EPSG:4978) [14] and rendered on the 3D virtual globe. Therefore, to visualize 3D Tiles on the CesiumJS virtual globe, all coordinates of 3D Tiles need to be transformed into the ECEF, *i.e.*, 3D Tiles should be geo-referenced.

The term geo-referencing is originally from the area of GIS and remote sensing (RS), which refers to the process of assigning geodetic coordinates to pixels of remotely sensed images, such as satellite images and aerial photo images. Before the raw non-georeferenced spatial datasets can be managed and analyzed in a GIS, their local coordinate systems (LCSs) must first be transformed into a proper CRS. This process is referred to as the traditional geo-referencing approach [15].

Geographic information systems (GISs) integrate, manage, and analyze heterogeneous geospatial datasets using coordinate reference systems (CRSs), and all GIS data formats have the capability to define CRSs.

However, the specification for 3D Tiles does not constrain to explicitly declare the CRS of the tileset. According to the specification for version 1.0, 3D Tiles uses a right-handed Cartesian coordinate system; that is, the cross product of x and y yields z. 3D Tiles defines the z-axis as up for local Cartesian coordinate systems. A tileset's global coordinate system will often be in a WGS 84 earth-centered, earth-fixed (ECEF) reference frame (EPSG:4978), but it doesn't have to be, e.g., a power plant may be defined fully in its local coordinate system for use with a modeling tool without a geospatial context [16]. At version 1.1 the CRS of a tileset may be defined explicitly, as part of the tileset metadata. The metadata for the tileset can contain a property that has the TILE-SET_CRS_GEOCENTRIC semantic, which is a string that represents the EPSG Geodetic Parameter Dataset identifier [17].

In 3D Tiles, each tile is defined in its own cartesian coordinate system and has an optional transform property [16]. The transform property is a 4×4 affine transformation matrix, stored in column-major order, which transforms from the tile's local coordinate system to the parent tile's coordinate system—or the tileset's coordinate system in the case of the root tile [16]. The transformation from each tile's local coordinate system to the tileset's global coordinate system is computed by a top-down traversal of the tileset and by post-multiplying a child's transform with its parent's transform like a traditional scene graph or node hierarchy in computer graphics [16].

Therefore, in order to establish a spatial reference for 3D Tiles, the local coordinate systems of each tile should be transformed to a local reference frame of a

three-dimensional cartesian spatial reference coordinate such as EPSG:1046 [18] and ECEF by the transform property of it and its parents. An east-north-up (ENU) system and a north-east-down (NED) system are good examples of local reference frames of the ECEF [19].

Although it is believed that CesiumJS is the most suitable for visualizing 3D Tiles on the web, geo-referencing which is indispensable for visualizing 3D Tiles on the CesiumJS virtual globe has rarely been studied systematically.

The objective of this study is to implement a 3D WebGIS framework that allows users to geo-reference 3D Tiles dynamically and flexibly on the CesiumJS virtual globe.

The remainder of this paper is organized as follows. Section 2 summarizes previous works related to the generation of 3D Tiles and geo-referencing them. Section 3 describes the proposed methods for geo-referencing 3D Tiles. Section 4 explores the validation of the proposed geo-referencing approach. Section 5 contains a discussion on the topic.

2. Previous Works

Generating 3D Tiles from heterogeneous kinds of spatial data such as photogrammetry or LiDAR-derived meshes, BIM, CAD, and point clouds has drawn much attention since the strength of 3D Tiles was known.

A few commercial software tools and platforms like Cesium ion [20], FME [21], Bently ContextCapture [22], Agisoft Metashape [23], TerraExplorer for Desktop [24], LuciadFusion Studio [25] to support generating 3D Tiles were reported, and among them, Cesium ion is widely used because of the excellent quality of generated data and supporting different kinds of input spatial data, high performance, convenient hosting, and ease of use. Therefore, many of the leading platforms for creating photogrammetry from drone captures, including ContextCapture, Agisoft Metashape [23], RealityCapture [26], Correlator3D [27], and OpenDroneMap [28] have added integrations to Cesium ion, allowing their users to export their data directly to Cesium ion where it can be tiled into 3D Tiles, hosted in the cloud, and combined with Cesium's global 3D datasets to be shared online [3].

However, Cesium ion does not support tiling georeferenced 3D models into 3D Tiles [29], and it assumes that the vertices of the uploaded 3D model are given in a local reference frame centered around the (0, 0, 0).

To set position, orientation, and scale for 3D Tiles generated from non-georeferenced data, Cesium ion provides a tool called "3D Tiles Location Editor" [30], however, it is not easy to position 3D Tiles into desired location accurately because it controls ENU system established on the center of the bounding sphere of 3D Tiles.

Obviously, it is inevitable that there are some limitations in customizing the geo-referencing of 3D Tiles in cases using commercial software tools and platforms because the workflow of tiling is a black box for users. Moreover, using

them is infeasible in the case the data are sensitive to the restricted public access requirement.

There are some open sources for generating 3D Tiles like Obj2Tiles [31], ObjTo3d-tiles [32], py3dtiles [33], py3dtilers [34], 3dtiles [35], osm-cesium-3d-tiles [36]. However, in practice, generated 3D Tiles are often poorly or not at all geo-referenced, therefore it is unavoidable to readjust the geo-referencing information by trial and error. For example, to set up geospatial information Obj2Tiles requires that users specify the longitude, latitude, and altitude of the OBJ 3D model. It isn't easy to get correct values of those command line parameters in practice. Similarly, ObjTo3d-tiles requires users to prepare a data file called "customTilesetOptions.json" containing geo-referencing information.

So far, it is generally agreed that no standard, a wholesome open-source solution exists for converting geospatial data into 3D Tiles. Therefore, many researchers have studied different tiling methods according to the characteristics of their data and the purpose of their project.

Schilling *et al.* [37] processed CityGML data set to prepare 3D Tiles for large and highly detailed 3D city models, and all CityGML coordinates given in a Coordinate Reference System (CRS) were reprojected into the ECEF and then transformed into a required local cartesian coordinate system, and combined with a global reference point in 3D Tiles using CESIUM_RTC [38] extension of glTF. Gan *et al.* [39] generated 3D Tiles from the digital surface model (DSM) derived from oblique photogrammetry automatic modeling and fine building model however, did not clearly explain how to integrate DSM with non-georeferenced data, manual 3D building model. Song *et al.* [40] converted oblique photogrammetry models with OSGB format into 3D Tiles however did not explain the detailed tiling method. Chen *et al.* [11] proposed a workflow to convert Building Information Models (BIM) in the Industry Foundation Classes (IFC) into 3D Tiles using open-source libraries, however, did not describe how to give geospatial information to converted 3D Tiles. Visuri *et al.* [41] used Cesium ion to tile CityGML and stream as 3D Tiles using FME and CesiumIonConnector plugin. Kolarić *et al.* [42] generated 3D Tiles from OpenStreetMap (OSM) data and BIM model but did not explain deeply how to geo-reference 3D Tiles. Xu *et al.* [43] created 3D Tiles from IFC using Obj2gltf [44] and described how to transform the local coordinate system of IFC into a global coordinate system of the 3D digital earth used in CesiumJS. Lu *et al.* [45] described data transformation steps to generate 3D Tiles from WRCD (weather radar composing data) in detail. They converted the latitude and longitude coordinates of the WRCD into Cartesian coordinates, moved the coordinate origin of the Cartesian coordinate system to the center point of the surface data, and determined the transform property of tileset.json using the geographic coordinates of the WRCD center point as the origin. Mao *et al.* [46] converted 3D geometry data of 3DCityDB to OBJ 3D models, and generated 3D Tiles from OBJ 3D models based on the Quadtree LOD algorithm using ObjTo3d-tiles but did not mention geo-referencing of 3D Tiles on the CesiumJS virtual

globe. Li *et al.* [47] generated 3D Tiles with OBJ 3D models using ObjTo3d-tiles and did not also clarify how generated 3D Tiles was geo-referenced from OBJ 3D models which do not have any geospatial information. Jaillot *et al.* [48] converted CityGML data into 3D Tiles extended with 3DTiles_temporal extension using their own tiler software, py3dtilers [34], and implemented the visualization in UD-Viz [49] but did not explain how to integrate coordinates of CityGML into 3D Tiles. Zhan *et al.* [50] generated 3D Tiles from complex BIM models based on degraded R-tree without simplifying BIM models, however, did not mention whether or not generated 3D tiles is geo-referenced.

In previous studies geo-referencing information of 3D Tiles was established during the generation process of 3D Tiles, therefore such geo-referencing can be considered static. In order to precisely place 3D Tiles at a desired position and an orientation on the virtual globe, it is necessary to flexibly change the geo-referencing information of 3D Tiles in the background of other geospatial data of the virtual globe such as terrain, satellite imagery, oblique aerial images, 3D model, and other vector data. Thus, 3D Tiles need to be geo-referenced in a dynamic mode. To address this issue, we propose a new 3D WebGIS framework that facilitates dynamic geo-referencing of 3D Tiles on the CesiumJS virtual globe.

3. Methodology

In the context of this paper, geo-referencing refers to the process of establishing a spatial reference for a tileset so that the global coordinate system of the tileset is transformed into the ECEF coordinate system.

We classify 3D Tiles into three categories in terms of geo-referencing as below.

Non-georeferenced 3D Tiles is one in which the position of vertices in all tiles is fully defined in the local Cartesian coordinate system. It is often that the center of the bounding sphere of the tileset coincides with the origin of the local coordinate system and the transform property of the root tile is an identity matrix.

Geo-referenced 3D Tiles is defined as one in which positions of vertices have meaning in a three-dimensional Cartesian spatial reference coordinate system such as EPSG:1046 and the ECEF. It is not possible to visualize 3D Tiles geo-referenced at a 3D Cartesian spatial reference coordinate system except ECEF on the CesiumJS virtual globe because the transformation from such coordinate system into the ECEF cannot be expressed as a 4×4 matrix.

As a special case of geo-referenced 3D Tiles, ECEF geo-referenced 3D Tiles is one in which positions of vertices are explicitly defined in the EPSG:4978, therefore it can be visualized on the CesiumJS virtual globe without any process. In ECEF geo-referenced 3D Tiles the transform property of the root tile is usually not an identity matrix, and each tile's local coordinate system is transformed into a local reference frame of the ECEF through the transform property of the root tile.

In this paper, we aim to establish a geo-referencing way for only non-georeferenced 3D Tiles and ECEF geo-referenced 3D Tiles.

Geo-referencing approaches for 3D Tiles have been implemented as a 3D WebGIS, namely Construkted Reality at https://construkted.com, which is the platform to stream and visualize massive photogrammetry and point cloud datasets with ease on the web. The front-end of the system has been developed with CesiumJS, which allows for the positioning of 3D Tiles on a virtual 3D globe viewed inside of a web browser.

In the system, 3D Tiles are managed in units of assets. An asset is a representation of one tileset that is related to several metadata including "geolocation" metadata for geo-referencing. An asset is created from a 3D model or a tileset, or a point cloud.

The "geolocation" metadata is expressed as JSON structure as shown in **Figure 1**. Longitude, latitude, and height determine the position of the asset, the origin of the local reference frame. Heading, pitch, and roll determine the orientation of the asset, and scale represents the scale of the asset. localGeoRefX, localGeoRefY, and localGeoRefZ are coordinates of "Geo-Location Reference" in the local coordinate system, which are explained in the following section.

On the Construkted Reality platform, geo-referencing is performed through two methods: the Visual Geo-Location Editor, and the Multi GCP Location Editor.

3.1. Visual Geo-Location Editor

Figure 2 shows the user interface of the Visual Geo-Location Editor. The UI provides a gizmo that allows users to translate and rotate 3D Tiles so that users can geo-reference 3D Tiles based on visual recognition. The center of the gizmo initially coincides with the center of the bounding sphere of the tileset but can be relocated to any point on the asset. Several text fields are also available to precisely adjust geo-referencing parameters.


```
{
    "longitude": 18.29078142,
    "latitude": 49.02953391,
    "height": 298.88561779,
    "heading": -0.00224372,
    "pitch": -0.00041056,
    "roll": -0.00214566,
    "localGeoRefX": 0,
    "localGeoRefY": 0,
    "localGeoRefZ": 0
    "scale": {
        "x": 1,
        "y": 1,
        "z": 1
    }
}
```


Figure 1. "geolocation" metadata.

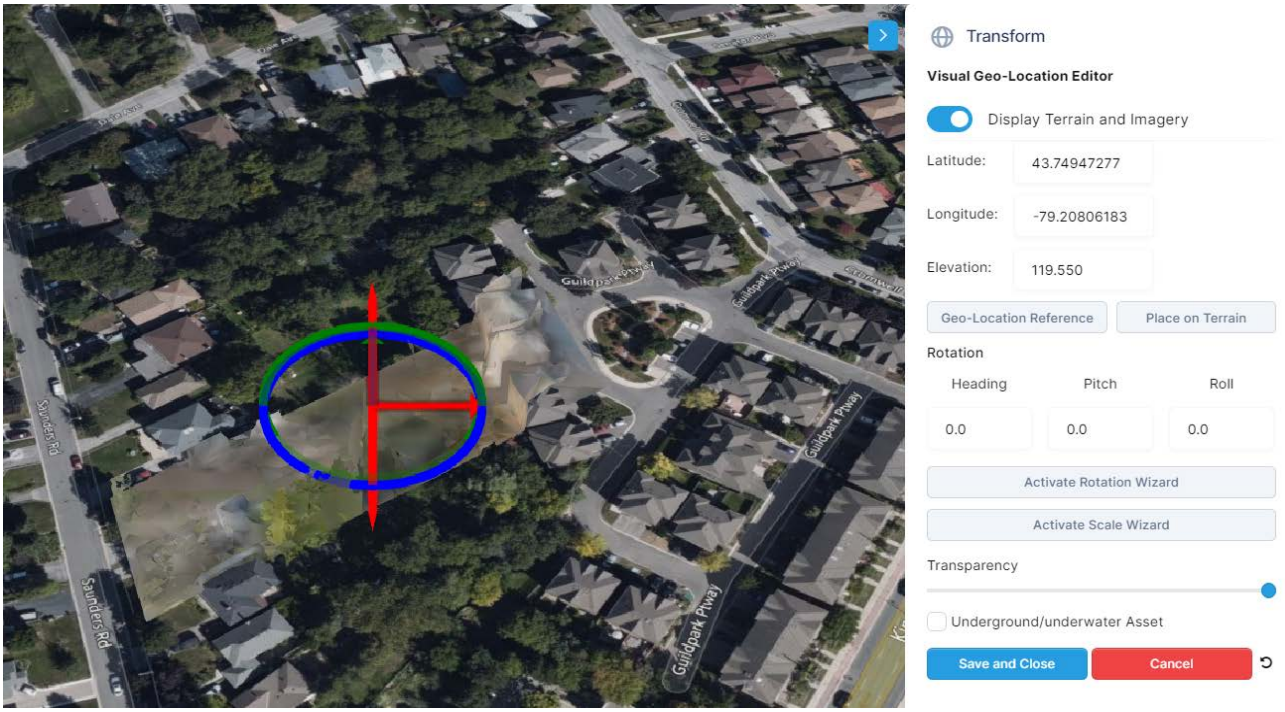

Figure 2. Visual geo-location editor.

To geo-reference 3D Tiles the Visual Geo-Location Editor uses the following 7 parameters: latitude, longitude, height, heading, pitch, roll, and scale which would be used to transform the local coordinate system of the root tile to a local reference frame of the ECEF. With these 7 parameters, it calculates the modelMatrix property of Cesium3DTileset [51], which is a 4 × 4 transformation matrix that transforms the tileset's root tile.

For non-georeferenced 3D Tiles, the modelMatrix is given by:

$$\boldsymbol{M}_{Local} = \boldsymbol{T}_{ENU}^{p} \cdot \boldsymbol{R} \cdot \boldsymbol{S} \tag{1}$$

where $\boldsymbol{S}$ is a 4 × 4 scaling matrix determined by the scale parameter, and $\boldsymbol{R}$ is a 4 × 4 rotation matrix determined by heading, pitch, and roll. p is a cartesian 3D point of the ECEF determined by longitude, latitude, and height. $\boldsymbol{T}_{ENU}^{p}$ is a 4 × 4 transformation matrix from an ENU reference frame centered at the position p, which can be easily obtained using a function of CesiumJS, Transforms.eastNorthUpToFixedFrame [52]. According to Equation (1), rotation and scaling are done in the local coordinate system of the tileset, and the origin of the local coordinate system is transformed into p. If the center of the bounding sphere of the tileset coincided with the origin of the local coordinate system before geo-referencing, p would become the center of the bounding sphere of the tileset after geo-referencing.

For ECEF geo-referenced 3D Tiles, the modelMatrix is calculated by:

$$\boldsymbol{M}_{ECEF} = \boldsymbol{T}_{ENU}^{p} \cdot \boldsymbol{R} \cdot \boldsymbol{S} \cdot \left(\boldsymbol{T}_{ENU}^{o}\right)^{-1} \tag{2}$$

where $\boldsymbol{T}_{ENU}^{o}$ is a 4 × 4 transformation matrix from an ENU reference frame

centered at the position *o*. *o* is an original center of the bounding sphere of the tileset when the modelMatrix property has not yet been modified from an identity matrix. By $\left(\boldsymbol{T}_{ENU}^{o}\right)^{-1}$ coordinates of the tileset are transformed from the ECEF into a local coordinate system. Rotation and scaling are still done with respect to the local coordinate system, and *o* is transformed into *p*, which is a new center of the bounding sphere of the tileset once geo-referencing is done.

Finally, *p* obtained by longitude, latitude, and height determines the center of the bounding sphere of the tileset, therefore, it represents the position of the asset.

Setting the position of the asset in terms of the center of the bounding sphere might be inconvenient, therefore, we introduce the concept of "Geo-Location Reference" (Figure 3). "Geo-Location Reference" is a point at which geographic coordinates are already determined by survey or can be easily guessed on the asset by the user. Once a "Geo-Location Reference" is specified on the asset by clicking the mouse, values in Latitude, Longitude, and Elevation text fields in the user interface are used to adjust the "Geo-Location Reference" instead of the center of the bounding sphere of the asset. Furthermore, the center of the gizmo is changed to the "Geo-Location Reference", and rotation and scaling are done with respect to the "Geo-Location Reference".

When a "Geo-Location Reference" is given, the modelMatrix of non-georeferenced 3D Tiles is calculated by

$$\boldsymbol{M}_{Local} = \boldsymbol{T}_{ENU}^{p} \cdot \boldsymbol{R} \cdot \boldsymbol{S} \cdot \boldsymbol{T} \tag{3}$$

where $\boldsymbol{T}$ is given by

$$\boldsymbol{M}_{Local} = \boldsymbol{T}_{ENU}^{p} \cdot \boldsymbol{R} \cdot \boldsymbol{S} \cdot \boldsymbol{T} \tag{4}$$

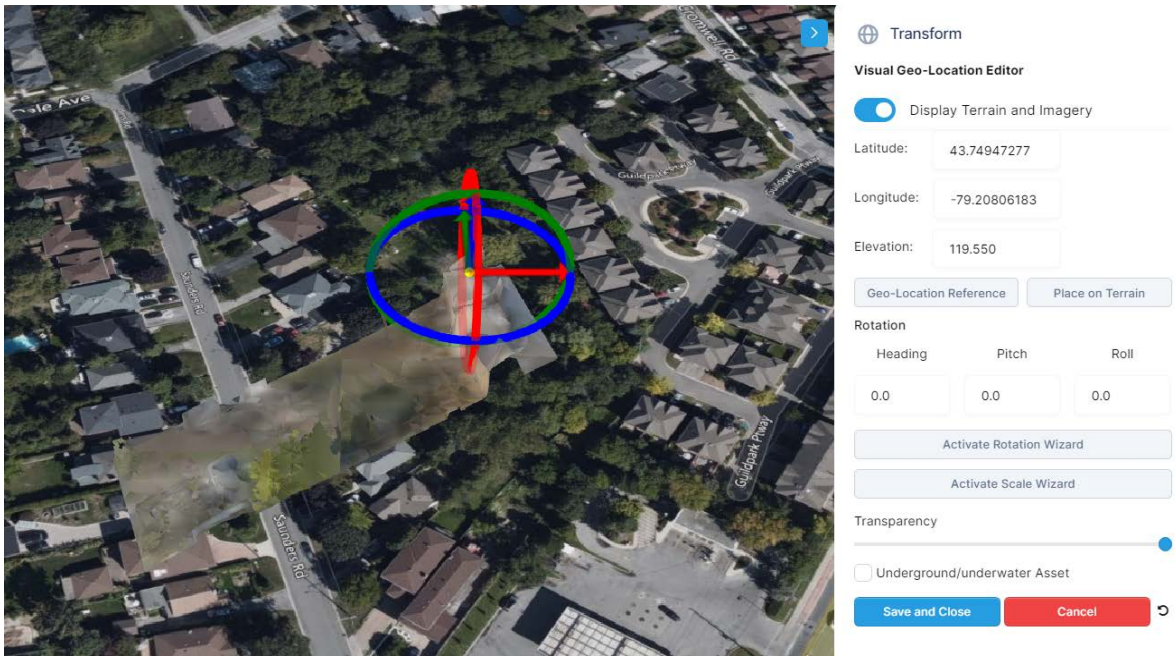

Figure 3. Geo-location reference (yellow dot).

In Equation (4), *x*, *y*, *z* are coordinates of "Geo-Location Reference" in the local coordinate system of the tileset.

For ECEF geo-referenced 3D Tiles, the modelMatrix is calculated by

$$\boldsymbol{M}_{ECEF} = \boldsymbol{T}_{ENU}^{p} \cdot \boldsymbol{R} \cdot \boldsymbol{S} \cdot \boldsymbol{T} \cdot \left(\boldsymbol{T}_{ENU}^{o}\right)^{-1} \tag{5}$$

The TypeScript codes for calculating modelMatrix of non-georeferenced 3D Tiles and ECEF geo-referenced 3D Tiles are presented in **Appendix A**.

The "Place on Terrain" function is used for perfectly clamping a given asset to the terrain by adjusting the height parameter with the correct height value of the terrain at given longitude and latitude. Using the "Rotation Wizard" function, users can define a local tangent plane by specifying three points on the asset. The "Scale Wizard" function allows users to scale the asset based on a real distance between two points on the asset.

3.2. Multi GCP Location Editor

When ground control points (GCPs) have been measured with the traditional surveying method or obtained by other sources, 3D Tiles can be precisely geo-referenced with these GCPs using the Multi GCP Location Editor. The user interface of the Multi GCP Location Editor allows users to specify points corresponding to the given GCPs on the asset (Figure 4).

The Multi GCP Location Editor calculates the best modelMatrix of the Cesium3DTileset, by which transformed corresponding points optimally match GCPs (Figure 5). In order to find the optimal/best rotation and translation between two sets of corresponding 3D points data, we use the Kabsch algorithm [53] which is the method for calculating the optimal translation and rotation that minimizes the Root Mean Squared Deviation (RMSD) between two paired sets of points.

Let $B_i, L_i, H_i \left(i = 1, 2, \cdots, n\right)$ be latitude, longitude, and height of GCPs and $p_i \left(i = 1, 2, \cdots, n\right)$ a point corresponding to ith GCP on the asset.

Figure 4. Multi GCP location editor.

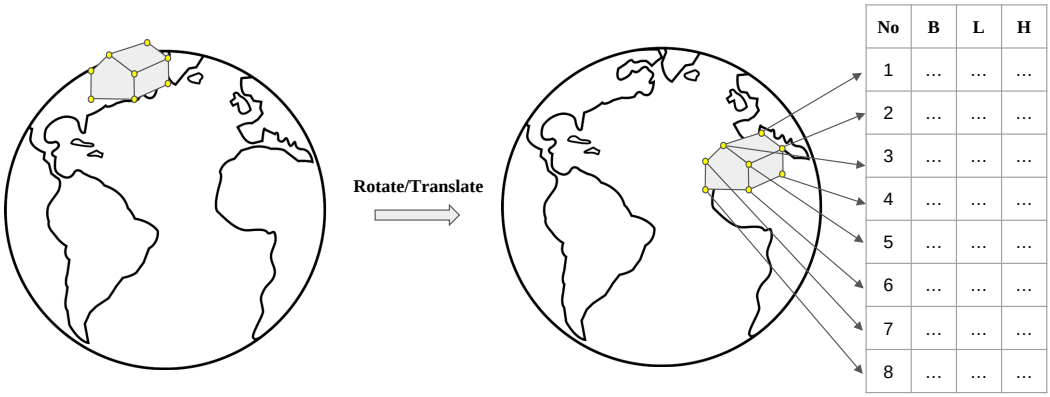

Figure 5. Geo-referencing of 3D Tiles using GCPs.

Then, $\boldsymbol{P}$ of the Kabsch algorithm can be directly constructed with p_i

$$\boldsymbol{P} = \begin{pmatrix} p_{x1} & p_{y1} & p_{z1} \\ p_{x2} & p_{y2} & p_{z2} \\ \vdots & \vdots & \vdots \\ p_{xn} & p_{yn} & p_{zn} \end{pmatrix} \tag{6}$$

$\boldsymbol{Q}$ of the Kabsch algorithm can be constructed by converting the geographic coordinate of GCPs into the ECEF coordinate system.

$$\boldsymbol{Q} = \begin{pmatrix} q_{x1} & q_{y1} & q_{z1} \\ q_{x2} & q_{y2} & q_{z2} \\ \vdots & \vdots & \vdots \\ q_{xn} & q_{yn} & q_{zn} \end{pmatrix} \tag{7}$$

where $\left(q_{xi}, q_{yi}, q_{zi}\right)$ is a cartesian coordinate in the ECEF for $B_i, L_i, H_i - H_{offset} \left(i = 1, 2, \cdots, n\right)$. H_{offset} is a constant called "altitudeOffset". The "altitudeOffset" is the offset value of altitude which is added to all ground control points' altitude. Surveyed altitude of GCPs may be different than the altitude of the terrain on the CesiumJS virtual globe, therefore this is very useful in case users hope to correctly clamp 3D Tiles to the scene's terrain, for example, Cesium World Terrain [54].

Let $\boldsymbol{C}_p, \boldsymbol{C}_q$ are centroids of GCPs and corresponding points.

$$\boldsymbol{C}_p = \frac{1}{n}\sum_{i=1}^{n}\left(p_{xi}, p_{yi}, p_{zi}\right) \tag{8}$$

$$\boldsymbol{C}_q = \frac{1}{n}\sum_{i=1}^{n}\left(q_{xi}, q_{yi}, q_{zi}\right) \tag{9}$$

Then the Translation step of the Kabsch algorithm is done using $\boldsymbol{C}_p, \boldsymbol{C}_q$, and $\boldsymbol{P}$, $\boldsymbol{Q}$ are updated so that their centroid coincides with $\boldsymbol{C}_p, \boldsymbol{C}_q$.

$$\boldsymbol{P} = \begin{pmatrix} p_{x1} - C_{px} & p_{y1} - C_{py} & p_{z1} - C_{pz} \\ p_{x2} - C_{px} & p_{y2} - C_{py} & p_{z2} - C_{pz} \\ \vdots & \vdots & \vdots \\ p_{xn} - C_{px} & p_{yn} - C_{py} & p_{zn} - C_{pz} \end{pmatrix} \tag{10}$$

$$\boldsymbol{Q} = \begin{pmatrix} q_{x1} - C_{qx} & q_{y1} - C_{qy} & q_{z1} - C_{qz} \\ q_{x2} - C_{qx} & q_{y2} - C_{qy} & q_{z2} - C_{qz} \\ \vdots & \vdots & \vdots \\ q_{xn} - C_{qx} & q_{yn} - C_{qy} & q_{zn} - C_{qz} \end{pmatrix} \tag{11}$$

Then, the optimal rotation matrix $\boldsymbol{R}$ that turns $\boldsymbol{P}$ as close as possible to $\boldsymbol{Q}$ is calculated using the Singular Value Decomposition [55] module as follows.

$$\boldsymbol{H} = \boldsymbol{Q}^{\mathrm{T}} \boldsymbol{P} \tag{12}$$

$$\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V} = SVD(\boldsymbol{H}) \tag{13}$$

$$\boldsymbol{R} = \boldsymbol{V}\boldsymbol{U}^{\mathrm{T}} \tag{14}$$

As a result, we get a 3 × 3 matrix $\boldsymbol{R}$ and then, we convert it to a 4 × 4 matrix:

$$\boldsymbol{R} = \begin{pmatrix} \boldsymbol{R}_{00} & \boldsymbol{R}_{10} & \boldsymbol{R}_{20} & 0 \\ \boldsymbol{R}_{01} & \boldsymbol{R}_{11} & \boldsymbol{R}_{21} & 0 \\ \boldsymbol{R}_{02} & \boldsymbol{R}_{12} & \boldsymbol{R}_{22} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{15}$$

Finally, the transformation is determined:

$$\boldsymbol{M}_{ECEF} = \boldsymbol{T}_{ENU}^{p} \cdot \boldsymbol{R} \cdot \boldsymbol{S} \cdot \boldsymbol{T} \cdot \left(\boldsymbol{T}_{ENU}^{o}\right)^{-1} \tag{16}$$

where,

$$\boldsymbol{T}_{p} = \begin{pmatrix} 1 & 0 & 0 & -C_{px} \\ 0 & 1 & 0 & -C_{py} \\ 0 & 0 & 1 & -C_{pz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{17}$$

$$\boldsymbol{T}_{q} = \begin{pmatrix} 1 & 0 & 0 & C_{qx} \\ 0 & 1 & 0 & C_{qy} \\ 0 & 0 & 1 & C_{qz} \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{18}$$

Error for each GCP is estimated as the distance between a GCP and the corresponding point transformed by the calculated $\boldsymbol{M}$.

4. Results

Several hundreds of assets were created and geo-referenced in the system by a growing number of users. In order to validate the Visual Geo-Location Editor, five assets geo-referenced by the Visual Geo-Location Editor were selected in Table 1. Links to assets are provided in **Appendix B**.

Qualities of geo-referencing of selected assets were assessed through visual comparison with Google Earth Pro, which provides 3D Buildings for selected assets. The elevation accuracy was not estimated because it cannot be guaranteed that terrain data are the same in Google Earth Pro and CesiumJS. Figure 6 shows the geo-referenced asset, Mexico City Metropolitan Cathedral in the system, which illustrates an accurate geo-referencing. Other assets in Table 1 also

Table 1. Selected assets for validating the visual geo-location editor.

Title	Position (longitude, latitude, height)	Orientation (heading, pitch, roll)
Mexico City Metropolitan Cathedral	(−99.13299675, 19.43416307, 2219.4)	(99.80, 0.00, −0.00)
Mausoleum of Petar II Petrovic Njegos, Montenegro	(18.83790629, 42.39985857, 1620.0)	(5.71, −0.68, −0.52)
Palace of Fine Arts, Mexico	(−99.14141637, 19.43528203, 2227.9)	(2.28, 0.00, 0.00)
Thomas Jefferson Memorial	(−77.03648737, 38.88097772, −39.0)	(0.00, −0.00, 0.00)
Pigeon Forge Margaritaville RV Resort	(−83.53535968, 35.77708751, 308.4)	(−0.01, 1.58, 0.35)

(a)

(b)

Figure 6. (a) Geo-referenced 3D tiles on Construkted Reality, and (b) 3D buildings on google earth. (a) Mexico City Metropolitan Cathedral (Construkted Reality); (b) Mexico City Metropolitan Cathedral (google earth)

show good geo-referencing results, which indicates that the Visual Geo-Location Editor is capable of placing dynamically and flexibly 3D Tiles at any desired position and orientation.

In order to evaluate the feasibility of the Multi GCP Location Editor, three assets were created from 3D models generated from sample datasets of OpenDroneMap [28], Pix4DMapper [56], and RealityCapture [26] that have GCPs, as shown in **Table 2**. Download links to sample datasets are given in **Appendix C**, and links to assets are provided in **Appendix D**.

The accuracy of geo-referencing was visually accessed through Microsoft BingMaps imagery draped on the Cesium World Terrain [54]. Different values of Altitude Offset were used to clamp assets to the Cesium World Terrain. **Figure 7** shows that the geo-referenced asset, shelffield_cross matches well the footprint of Microsoft BingMaps imagery.

Table 2. Created assets for validating the multi GCP location editor.

Title	Number of GCPs	Position (longitude, latitude, height)	Orientation (heading, pitch, roll)	Altitude offset	Average error
Sheffield_cross	5	(−82.69705665, 28.04120802, −28.07)	(−0.81, −0.33, 0.17)	−28	0.48
Quarry	9	(6.53680493, 46.65523103, 594.94)	(−0.62, 0.01, 0.01)	60	0.33
Drone Imagery + Ground control points	13	(18.29078143, 49.02953391, 289.88)	(−0.00, −0.00, −0.00)	121	0.01

Figure 7. Geo-referenced sheffield_cross.

Table 2 shows that centimeter accuracy can be reached in the geo-referencing process by Multi GCP Location Editor.

5. Conclusions

Geo-referencing is a vital step for visualizing 3D Tiles on the CesiumJS virtual globe, and it has drawn considerable attention from the Cesium community. Even if there exist some commercial and open-source tools for generating 3D Tiles, and they are capable of embedding spatial reference information into 3D Tiles, the required spatial reference information may not be properly assigned during the generation or not assigned at all.

In order to precisely place 3D Tiles at a desired position and an orientation on the CesiumJS virtual globe, this paper presents the implementation of a 3D Web-GIS which facilitates dynamic geo-referencing of 3D Tiles on the background of other spatial data of the CesiumJS virtual globe.

Geo-referencing of 3D Tiles is performed through the Visual Geo-Location Editor and the Multi GCP Location Editor that calculates the modelMatrix property of Cesium3DTileset from "geolocation" metadata so that the local coordinate system of the root tile of 3D Tiles is transformed into a local reference frame of the ECEF. This makes sure the global coordinate system of 3D Tiles is ECEF.

The feasibility of the Visual Geo-Location Editor was tested by comparing several assets created and geo-referenced by different users with 3D models of Google Earth Pro. Also, three assets were created with 3D models generated from

sample datasets of different photogrammetry software, and geo-referenced with Multi GCP Location Editor, and then the error was gauged for each asset. Experimental results indicate that implemented 3D WebGIS can facilitate dynamic geo-referencing of 3D Tiles.

Though the current version of Construkted Reality has the potential to be used for georeferencing 3D Tiles, there are some considerable things in using it. The system simply changes the transform property of the root tile of the tileset to adjust the position and the orientation of the 3D Tiles at runtime, therefore, during georeferencing, any part of the 3D Tiles itself is not warped.

Non-georeferenced 3D Tiles can be generated from a 3D model defined at a local Cartesian coordinate system at which the Earth's surface is modeled as an infinite, flat surface. In this case, it might be difficult to precisely clamp 3D Tiles on the curved surface of the Earth, and this issue would become clear in large, linear objects such as roads, and railways. Also, Geo-referencing of 3D Tiles defined at a Cartesian spatial coordinate system except ECEF is not yet supported, because the transformation into a local reference frame of the ECEF cannot be expressed as a 4×4 matrix. In future research, we plan to focus on these issues to further improve the applicability of our system.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Cozzi, P. (2015) Introducing 3D Tiles. https://cesium.com/blog/2015/08/10/introducing-3d-tiles/

[2] 3D Tiling with Cesium. Cesium GS, Inc. https://cesium.com/platform/cesium-ion/3d-tiling-pipeline/

[3] Sampath, M. (2021) Agisoft Metashape Now Directly Supports Sharing 3D Models with Cesium Ion. https://cesium.com/blog/2021/09/10/agisoft-metashape-integrated-with-cesium-ion/

[4] CesiumJS. Cesium GS, Inc. https://cesium.com/platform/cesiumjs/

[5] iTowns. https://github.com/iTowns/itowns

[6] visgl. deck.gl. https://github.com/visgl/deck.gl

[7] NASA-AMMOS. 3DTilesRendererJS. https://github.com/NASA-AMMOS/3DTilesRendererJS

[8] Nytimes. Three-Loader-3dtiles. https://github.com/nytimes/three-loader-3dtiles

[9] Ebeaufay. Threedtiles. https://github.com/ebeaufay/threedtiles

[10] Geodan. Mapbox-3dtiles. https://github.com/Geodan/mapbox-3dtiles

[11] Yiqun, C., Erfan, S., Abbas, R. and Soheil, S. (2018) From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium. *ISPRS International Journal of Geo-Information*, **7**, Article No. 393. https://doi.org/10.3390/ijgi7100393

[12] World Geodetic System. Wikimedia. https://en.wikipedia.org/wiki/World_Geodetic_System

[13] Earth-Centered, Earth-Fixed Coordinate System. Wikimedia. https://en.wikipedia.org/wiki/Earth-centered,_Earth-fixed_coordinate_system

[14] Klokan Technologies. EPSG:4978. https://epsg.io/4978

[15] Zhu, J. and Wu, P. (2021) A Common Approach to Geo-Referencing Building Models in Industry Foundation Classes for BIM/GIS Integration. *ISPRS International Journal of Geo-Information*, **10**, Article No. 362. https://doi.org/10.3390/ijgi10060362

[16] 3D Tiles Specification 1.0. Open Geospatial Consortium. https://docs.opengeospatial.org/cs/18-053r2/18-053r2.html

[17] Cozzi, P. and Lilley, S. (2022) 3D Tiles Specification 1.1. Open Geospatial Consortium. https://docs.ogc.org/cs/22-025r4/22-025r4.html

[18] Klokan Technologies. EPSG:1046. https://epsg.io/1046-cs

[19] Local Tangent Plane Coordinates. Wikimedia. https://en.wikipedia.org/wiki/Local_tangent_plane_coordinates

[20] Cesium Ion. Cesium GS, Inc. https://cesium.com/platform/cesium-ion/

[21] FME. Safe Software Inc. https://fme.safe.com/platform/

[22] Bently Context Capture. Bently Systems Inc. https://www.bentley.com/software/contextcapture/

[23] Agisoft Metashape. Agisoft. https://www.agisoft.com/downloads/installer/

[24] TerraExplorer for Desktop. Skyline Software Systems Inc. https://www.skylinesoft.com/terraexplorer-for-desktop/

[25] LuciadFusion Studio. https://dev.luciad.com/portal/productDocumentation/LuciadFusion/docs/articles/userguide/studio/UserGuide.html

[26] RealityCapture. CapturingReality. https://www.capturingreality.com/

[27] Correlator3D. SimActive. https://www.simactive.com/

[28] Awesome. Drone. Software. OpenDroneMap. https://www.opendronemap.org/

[29] Shehata, O. (2020) Support Tiling Georeferenced 3D Models into 3D Tiles. https://community.cesium.com/t/support-tiling-georeferenced-3d-models-into-3d-tiles/10524

[30] 3D Tiles Location Editor. Cesium GS, Inc. https://cesium.com/learn/3d-tiling/ion-tile-set-location/

[31] OpenDroneMap. Obj2Tiles. https://github.com/OpenDroneMap/Obj2Tiles

[32] PrincessGod. ObjTo3d-Tiles. https://github.com/PrincessGod/objTo3d-tiles

[33] Oslandia. Py3dtiles. https://github.com/Oslandia/py3dtiles

[34] VCityTeam. Py3dtilers. https://github.com/VCityTeam/py3dtilers

[35] Fanvanzh. 3dtiles. https://github.com/fanvanzh/3dtiles

[36] Kiselev-dv. Osm-Cesium-3D-Tiles. https://github.com/kiselev-dv/osm-cesium-3d-tiles

[37] Arne, S., Jannes, B. and Claus, N. (2016) Using glTF for Streaming CityGML 3D City Models. *Proceedings of the* 21*st International Conference on Web3D Technology*, Anaheim, 22-24 July 2016, 109-116.

[38] KhronosGroup. CESIUM_RTC. https://github.com/KhronosGroup/glTF/blob/main/extensions/1.0/Vendor/CESIUM_RTC/README.md

[39] Gan, L., Li, J. and Jing, N. (2017) Hybrid Organization and Visualization of the DSM Combined with 3D Building Model. *Proceedings of the* 2017 2*nd Internation-*

al Conference on Image, Vision and Computing, Chengdu, 2-4 June 2017, 566-571. https://doi.org/10.1109/ICIVC.2017.7984619

[40] Song, Z. and Li, J. (2018) A Dynamic Tiles Loading and Scheduling Strategy for Massive Oblique Photogrammetry Models. 2018 *IEEE* 3*rd International Conference on Image*, *Vision and Computing* (*ICIVC*), 27-29 June 2018, Chongqing, 648-652. https://doi.org/10.1109/ICIVC.2018.8492731

[41] Visuri, H., Jokela, J., Mesterton, N., Latvala, P., and Aarnio, T. (2019) Producing and Visualizing a Country-Wide 3d Data Repository in Finland. *The International Archives of the Photogrammetry*, *Remote Sensing and Spatial Information Sciences*, *Volum*e XLII-4/*W*15, 2019 14*th* 3*D GeoInfo Conference*, 24-27 September 2019, Singapore, 105-110. https://doi.org/10.5194/isprs-archives-XLII-4-W15-105-2019

[42] Kolaric, S. and Shelden, D. (2019) DBL SmartCity: An Open-Source IoT Platform for Managing Large BIM and 3D Geo-Referenced Datasets. *Proceedings of the* 52*nd Hawaii International Conference on System Sciences*, Hawaii, 8-11 January 2019.

[43] Xu, Z., Zhang, L., Li, H., Lin, Y.-H. and Yin, S. (2020) Combining IFC and 3D Tiles to Create 3D Visualization for Building Information Modeling. *Automation in Construction*, **109**, Article ID: 102995. https://doi.org/10.1016/j.autcon.2019.102995

[44] Obj2gltf. Cesium GS, Inc. https://github.com/CesiumGS/obj2gltf

[45] Lu, M., Wang, X., Liu, X., Chen, M., Bi, S., Zhang, Y. and Lao, T. (2021) Web-Based Real-Time Visualization of Large-Scale Weather Radar Data Using 3d Tiles. *Transactions in GIS*, **25**, 25-43. https://doi.org/10.1111/tgis.12638 https://onlinelibrary.wiley.com/doi/epdf/10.1111/tgis.12638

[46] Mao, B., Ban, Y. and Laumert, B. (2020) Dynamic Online 3D Visualization Framework for Real-Time Energy Simulation Based on 3D Tiles. *ISPRS International Journal of Geo-Information*, **9**, Article No. 166. https://doi.org/10.3390/ijgi9030166

[47] Li, Y., Zhang, H. and Zhang, Q. (2021) A Framework for Interactive Online 3D Visualization of Electric Information. *Journal of Physics*: *Conference Series*, **1757**, Article ID: 012170. https://doi.org/10.1088/1742-6596/1757/1/012170

[48] Jaillot, V., Servigne, S. and Gesquière, G. (2020) Delivering Time-Evolving 3D City Models for Web Visualization. *International Journal of Geographical Information Science*, **34**, 2030-2052. https://doi.org/10.1080/13658816.2020.1749637

[49] VCityTeam. UD-Viz. https://github.com/VCityTeam/UD-Viz

[50] Zhan, W, Chen, Y. and Chen, J. (2021) 3D Tiles-Based High-Efficiency Visualization Method for Complex BIM Models on the Web. *ISPRS International Journal of Geo-Information*, **10**, Article No. 476. https://doi.org/10.3390/ijgi10070476

[51] Cesium3DTileset. Cesium GS, Inc. https://cesium.com/learn/cesiumjs/ref-doc/Cesium3DTileset.html

[52] Transforms. Cesium GS, Inc. https://cesium.com/learn/cesiumjs/ref-doc/Transforms.html

[53] Kabsch Algorithm. Wikimedia. https://en.m.wikipedia.org/wiki/Kabsch_algorithm

[54] Cesium World Terrain. Cesium GS, Inc. https://cesium.com/platform/cesium-ion/content/cesium-world-terrain/

[55] Singular Value Decomposition. Wikimedia. https://en.wikipedia.org/wiki/Singular_value_decomposition

[56] Pix4D. PIX4Dmapper. https://www.pix4d.com/product/pix4dmapper-photogrammetry-software

Supplementary Materials

Demo videos of dynamic geo-referencing functions of Construkted Reality are available online: https://youtu.be/KWRq5yVCrVs.

Appendix A

TypeScript codes for calculating the modelMatrix of non-georeferenced 3D Tiles and ECEF Geo-referenced 3D Tiles

```
export function updateModelMatrixOfTilesetDefinedAtLocal(
  tileset: Cesium3DTileset,
  position: Cartesian3,
  refInLocal: Cartesian3,
  hpr: HeadingPitchRoll,
  scale: Cartesian3
) {
  const toWorld = Transforms.eastNorthUpToFixedFrame(position);
  const T = Matrix4.fromTranslation(Cartesian3.negate(refInLocal, new Cartesian3()));
  const R = Matrix4.fromRotation(Matrix3.fromQuaternion(Quaternion.fromHeading
PitchRoll(hpr)));
  const S = Matrix4.fromScale(scale, new Matrix4());
  const modelMatrix = toWorld;
  Matrix4.multiply(modelMatrix, R, modelMatrix);
  Matrix4.multiply(modelMatrix, S, modelMatrix);
  Matrix4.multiply(toWorld, T, modelMatrix);
  tileset.modelMatrix = modelMatrix;
}
export function updateModelMatrixOfTilesetDefinedAtECEF(
  tileset: Cesium3DTileset,
  position: Cartesian3,
  geoRefAtLocal: Cartesian3,
  hpr: HeadingPitchRoll,
  scale: Cartesian3
) {
  tileset.modelMatrix = Matrix4.IDENTITY;
  const origin = Cartesian3.clone(tileset.boundingSphere.center);
  const referenceFrame = Transforms.eastNorthUpToFixedFrame(origin);
  const toLocal = Matrix4.inverseTransformation(referenceFrame, new Matrix4());
  const T = Matrix4.fromTranslation(Cartesian3.negate(geoRefAtLocal, new Carte-
sian3()));
  const R = Matrix4.fromRotation(Matrix3.fromQuaternion(Quaternion.fromHeading
PitchRoll(hpr)));
  const S = Matrix4.fromScale(scale, new Matrix4());
  const toWorld = Transforms.eastNorthUpToFixedFrame(position);
  const modelMatrix = toWorld;
```

```
    Matrix4.multiply(modelMatrix, R, modelMatrix);
    Matrix4.multiply(modelMatrix, S, modelMatrix);
    Matrix4.multiply(modelMatrix, T, modelMatrix);
    Matrix4.multiply(modelMatrix, toLocal, modelMatrix);
    tileset.modelMatrix = modelMatrix;
  }
```

Appendix B

Links of assets selected for validation the Visual Geo-Location Editor.

Mexico City Metropolitan Cathedral: https://construkted.com/asset/arkcnfuk9fw/

Mausoleum of Petar II Petrovic Njegos, Montenegro: https://construkted.com/asset/ank1b5x1jrp/

Palace of Fine Arts, Mexico: https://construkted.com/asset/a73faxnydqk/

Thomas Jefferson Memorial 02: https://construkted.com/asset/qpnur8sroi/

Pigeon Forge Margaritaville RV Resort: https://construkted.com/asset/ayooitlvgg9/

Appendix C

Sample datasets used for evaluating the Multi GCP Location Editor are available at the following links.

https://github.com/pierotofy/drone_dataset_sheffield_cross/

https://s3.amazonaws.com/mics.pix4d.com/example_datasets/example_quarry_2.0.zip

https://www.capturingreality.com/download/files/GCP-Drone-Sample-Dataset

Appendix D

Links of assets selected for validation the Multi GCP Location Editor.

sheffield_cross: https://construkted.com/asset/a1fg3o0r69r/

Quarry: https://construkted.com/asset/avajuri6r3g/

Drone Imagery + Ground control points: https://construkted.com/asset/avplbp844b3/