electronics

MDPI

Article

Spatial Information Platform with VWorld for Improving User Experience in Limited Web Environment

Ahyun Lee * and Insung Jang

City & Geospatial ICT Research Section, Electronics and Telecommunications Research Institute (ETRI), 218 Gajeong-ro, Yuseong-gu, Daejeon 34129, Korea; e4dol2@etri.re.kr

* Correspondence: ahyun@etri.re.kr

Received: 25 October 2019; Accepted: 24 November 2019; Published: 26 November 2019

Abstract: VWorld Data Center (VDC), operated by the Ministry of Land, Infrastructure, and Transport of South of Korea, provides the latest high-quality spatial information of South Korea. VWorld data include aerial images, digital elevation models, 3D buildings, 3D bridges, roads, administrative information, etc. We present a 3D spatial information platform that uses VWorld data. This platform is based on the web browser environment and supports cross platform by complying with WebGL and web standards. The 3D-terrain tile structure has a quadtree-based detail level for high-resolution rendering. When rendering different tile levels or sizes simultaneously, abnormal gaps occur between the tiles. We propose the use of a specially shaped 3D terrain object model to minimize these gaps. This model can be used and verified using the service site of the WebGL-based VWorld spatial information platform. Our platform has a limited environment and runs in a web browser while requesting real-time spatial information data from VDC. VWorld data cannot be stored locally by policy. We improve the frame rate in such way that it can easily be used in a limited web browser environment or even a computing environment without a graphics processing unit. We intend to officially present the proposed method to the VDC and apply it. We expect our platform to be used for various web-based geospatial applications.

Keywords: GIS; 3D map; WebGL; VWorld

1. Introduction

VWorld Data Center (VDC) of the Ministry of Land, Infrastructure, and Transport of South of Korea provides up-to-date high-quality national spatial information [1]. The provided national spatial information is terrain-based data such as aerial images, digital elevation models (DEMs), and 3D structures. VWorld spatial information provides aerial images of areas around the world. In particular, aerial images of the Korean Peninsula (South and North Korea) are provided with 12-cm accuracy. It provides the data of 3D buildings and bridges for major cities such as Seoul, Busan, Incheon, and Daejeon in the South of Korea. In addition, it provides the 3D data of Pyongyang's representative structures such as Ryugyong Hotel and Rungnado Stadium. Due to the peculiarity of South Korea as a divided country, Google Earth [2] and other web 3D maps only service 3D terrain except buildings. The VDC enables to provide 3D terrains and buildings that have been modified in accordance with the Korean government's security policy.

VDC provides approximately 30 TB or more through DataAPI to users [1,3]. Users can use DataAPI to download VWorld data; however, there are limitations. South Korea is in a special situation of a truce with North Korea. The government of South Korea limits the unauthorized use of measurable spatial information such as longitude, latitude, or height of a specific location in a building. Since

the misuse of VWorld data could cause a security threat, editing and redistributing it for proprietary purposes are prohibited by law.

Figure 1 shows the proposed WebGL-based VWorld spatial information platform service site [4,5]. Users can control the camera to render 3D terrains and buildings from various angles. The previous VWorld map service [6] was an ActiveX-based program that ran only in Internet Explorer. VWorld map service site needed a cross-platform service for more users to easily access it. WebGL is a cross-platform, royalty-free web standard for a low-level 3D graphics application programming interface (API) based on OpenGL embedded systems (ES) exposed to ECMAScript via the HTML5 Canvas element. Developers familiar with OpenGL ES 2.0 will recognize WebGL as a Shader-based API that uses GLSL with constructs that are semantically similar to those of the underlying OpenGL ES API. WebGL brings plugin-free 3D to the web for implementation in the browser. Major browser vendors Apple (Safari), Google (Chrome), Microsoft (Edge), and Mozilla (Firefox) are members of the WebGL Working Group [7–11]. The proposed WebGL-based platform is cross-platform compatible and has been tested on at least nine web browsers and four or more operating systems [5].

Figure 1. The proposed WebGL-based VWorld spatial information platform.

This paper contributes to solving two major problems through the implementation of a 3D map platform based on the web browser environment. The first problem is the implementation of the real-time web-based platform under a limited computing environment. A platform designed for a wide range of people should be able to guarantee at least 30 FPS in a computing environment without a graphics processing unit (GPU). We present a method for speeding up request and download times and finding the data to be rendered from large amounts of data, over 30 TB.

The second problem is the determination of a method for expressing the 3D terrain object model of VWorld data. VWorld spatial information data does not provide 3D terrain object models; it only provides aerial images and DEMs. We propose a method for generating 3D terrain object models using the VWorld data. In geospatial applications, quadtree-based level of detail (LOD) approach is widely used for streaming images, terrains, vectors, and other data. Figure 2 shows an example of a complete quadtree-based LOD structure. All the tiles are defined {level, x, y} and have the same resolution. The root is at level zero, its children are at level one, and so on [12–15]. Since all the tiles have the same resolution, when the lower level tiles are represented, more tiles are used for rendering. As more tiles are used, the resolution of the image sources in a particular area can be increased.

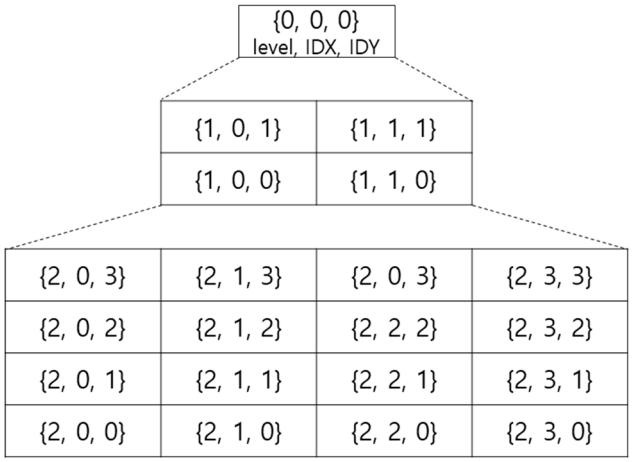
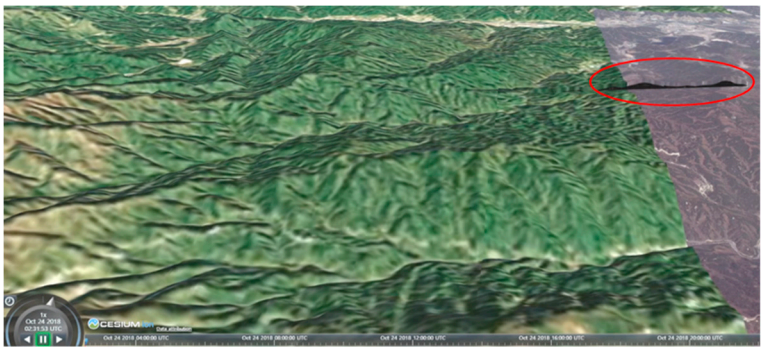


Figure 2. Example of quadtree structure. All the tiles have the same resolution.

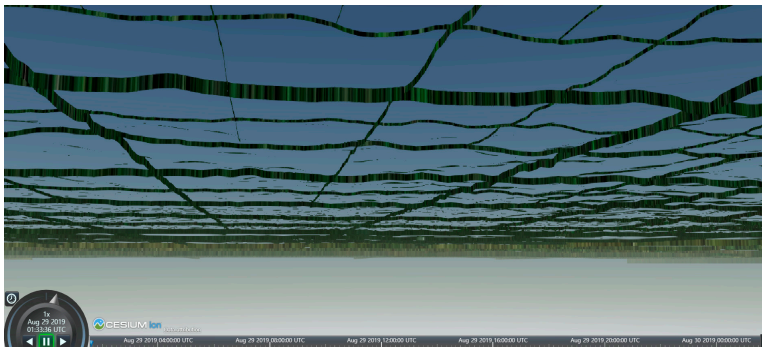
The 3D map platforms with quadtree-based LOD make use of the position and direction of the camera to determine the LOD of the tiles to be rendered on the screen. Depending on the position and direction of the camera, various LODs of tiles may be adjacent to each other and rendered in a frame. In such a case, gaps such as a hollow state may occur. Cesium is a representative spatial information platform with quadtree-based LOD [16]. The formats and methods used in Cesium are widely used as standards in the field of 3D mapping. Figure 3 shows the gaps that occur when another LOD of tiles are adjacent in the Cesium platform. This gap occurrence is common in 3D map platforms with the quadtree-based LOD; 3D map platforms make use of a variety of approaches to minimize gaps. Cesium creates additional meshes similar to curtains in the terrain model. Figure 3c shows the result obtained when the camera is moved to a level below the ground.



(**a**)

(**b**)

Figure 3. *Cont.*



(**c**)

Figure 3. Cesium 3D map: (**a**) gaps in mountainous terrain, (**b**) gaps in the edges of unclear tiles owing to some tiles not having loaded, and (**c**) use of curtain-like 3D terrain object models for minimizing gaps.

When different LODs are adjacent to each other, the gap can be minimized by changing the shape of the meshes of a terrain model. In previous studies, the shape of the outer meshes was changed according to the shape of the outer meshes of the adjacent LOD tile [17,18]. In such a case, it is necessary to know in advance the information of all the tiles rendered before the meshes of the 3D tiles are generated. When the position or direction of the camera is changed, the tile LOD should also be changed. As a result, a procedure is required to change or confirm the shape of the 3D terrain object models in every frame. The tiles to be rendered can be changed according to the data request result in the web-based 3D map platforms that obtain geospatial data from the server. Therefore, it is difficult to predict the relationship between adjacent tiles in advance. Furthermore, depending on the server state, if the data requests fail, the tile continuity can become irregular. Owing to the limited computing environment in a web browser, changing or confirming the shape of the meshes of a 3D terrain object model in every frame can affect the performance.

In this paper, we present a method for rendering 3D terrain object models effectively in the aforementioned limited environment based on a web browser. In contrast to the existing research methods, we propose a method for minimizing the gap that can occur when different LOD tiles are adjacent even when the information of the neighboring tiles is unknown. The proposed platform has a rendering frame rate of more than 50 FPS even in a computing environment without a GPU, and it can thus be easily used. Anyone can use the proposed platform through the service site [4].

2. VWorld Spatial Information Data

Vworld spatial information data comprises a variety of the latest high-definition data such as aerial images, DEMs, 3D buildings, administrative information, roads, and facility names. The data is indexed in units of tiles and has a quadtree-based LOD tile structure. A layer is used to distinguish various spatial information data in a tile. A layer is a unit used for classifying spatial information data according to attributes. The spatial information data with the same attributes can be managed collectively using layers [19–21]. A tile is defined as {level, IDX, IDY}, and one tile includes a plurality of layers. In this paper, a tile that has various layers is defined as a sector. A sector is a minimum unit for storing Vworld spatial information data and a basic unit displayed on the screen. If a sector is rendered on the screen, the spatial information of the activated layer in this sector is rendered.

2.1. Sector

Vworld data constitutes all sectors based on the latitude and longitude coordinates. It has six LOD steps (0–5) throughout the world. For South Korea and North Korea, it comprises 13 to 17 steps of LOD (0–12 or 16). The sectors at LOD 0 consist of 50 sectors divided into 5 steps of latitude and 10 steps of longitude. They are square with a latitude and longitude of 36° each. Figure 4a shows the 50 sectors at LOD 0. It is based on the X-axis in the longitude direction and Y-axis in the latitude

direction from the lower left origin. Sectors are defined as {level, IDX, IDY}. For example, the lower left sector of (a) is defined as {0, 0, 0}, and the upper right sector is defined as {0, 9, 4}. Figure 4b shows the sectors at LOD 1 that consist of 10 steps of latitude and 20 steps of longitude, which result in a total of 100 sectors. A sector at LOD 0 is divided into four sectors at LOD 1. In this paper, we implemented the 3D spherical earth, and Figure 4c shows the result thus obtained.

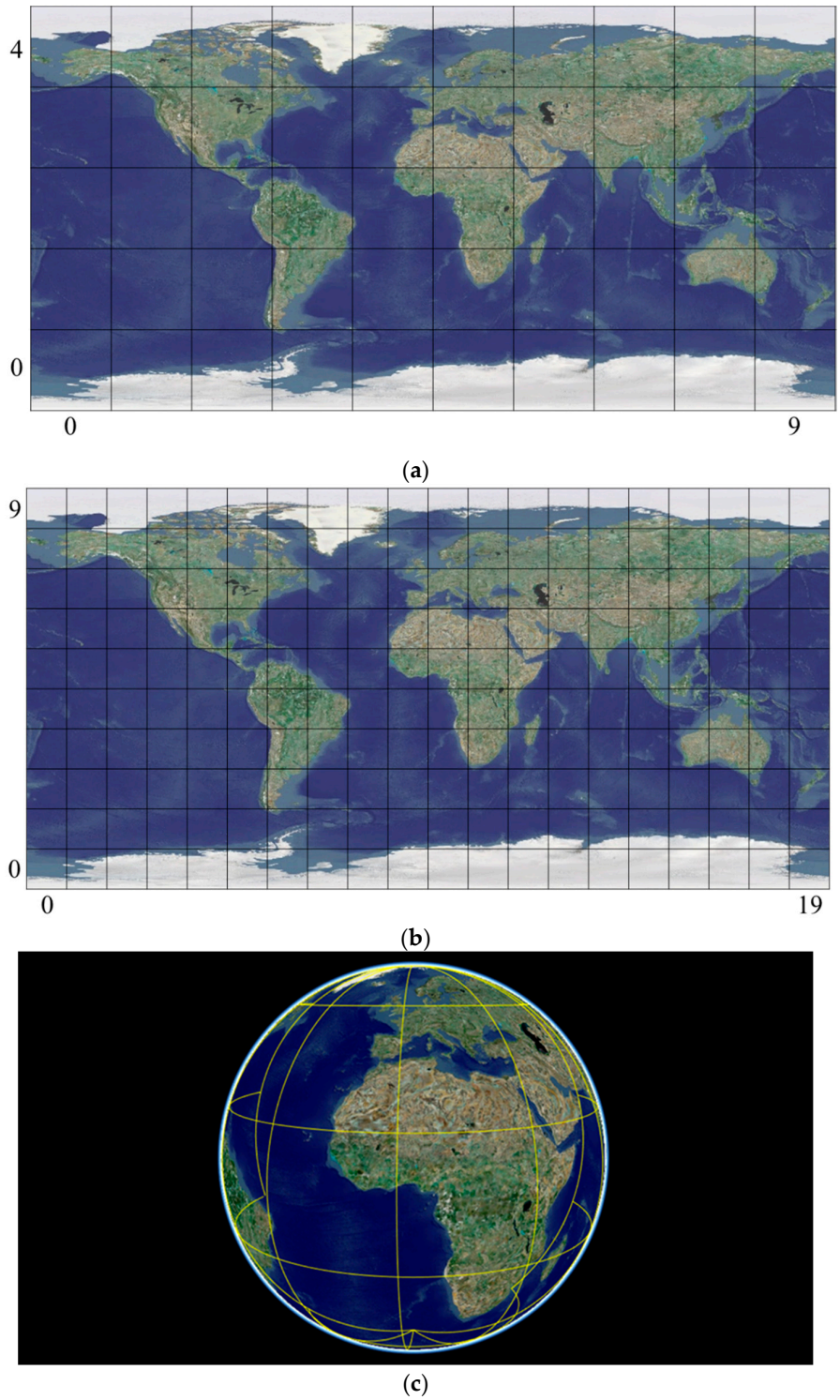


Figure 4. VWorld tile system: (**a**) sectors at LOD 0, (**b**) sectors at LOD 1, and (**c**) sectors at LOD 0 are represented by the sphere, and the yellow lines are the outlines of sectors.

Figure 5 shows an example of spatial information presentation in a sector. At the bottom, an aerial image is placed, and additional images with transparency are superimposed thereon. Figure 5b shows an orthogonal image. The images taken from the aircraft are distorted, as shown in Figure 5a. To correct the distortion of the images caused by the altitude, an orthogonal image similar to that taken from the vertical of the building is generated. The overlapping result is presented in Figure 5f. A plurality of layers is superimposed on a sector to express the final rendering result.

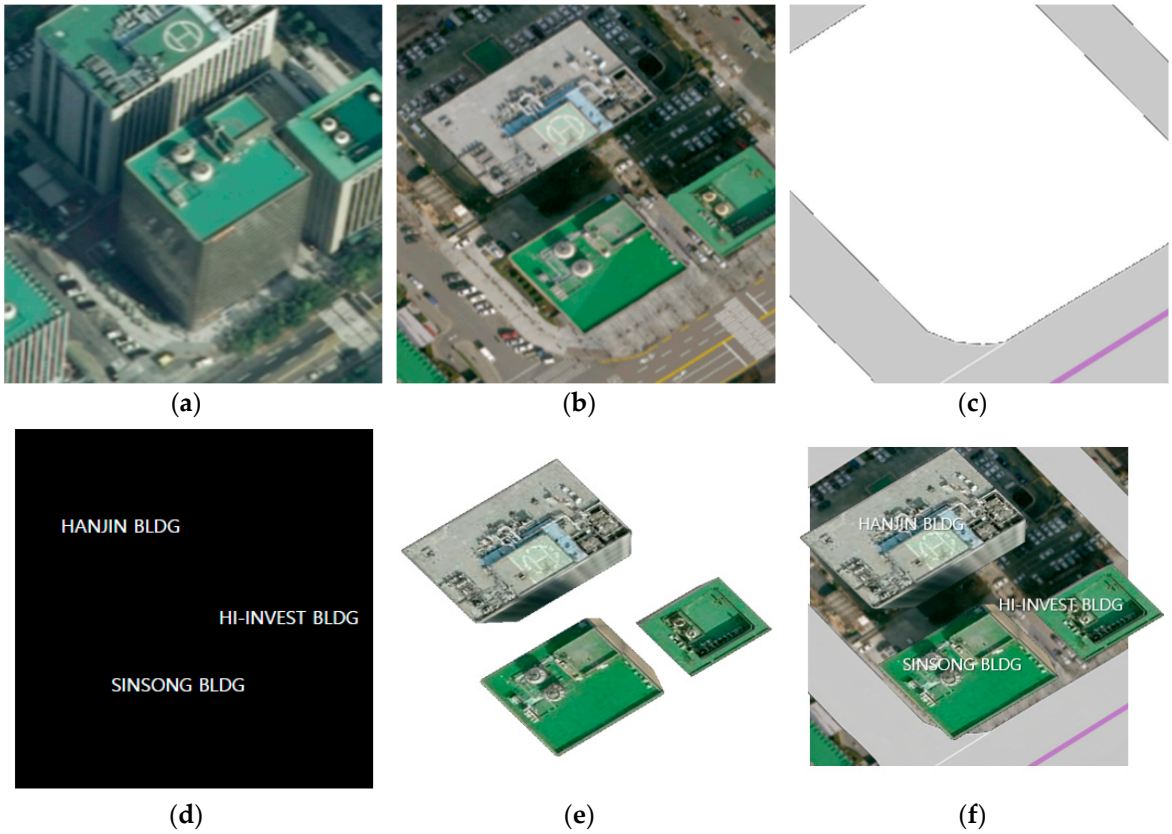


Figure 5. Examples of spatial information layers: (**a**) oblique aerial image, (**b**) orthogonal image, (**c**) road, (**d**) facility name, (**e**) 3D building models, and (**f**) superimposed result of layers (**a**)–(**e**).

The spatial information provided by VDC comprises over 30 TB, and the amount of data increases as the spatial data is updated every year. In this paper, we classify dozens of types of spatial information data into tiles with latitude and longitude and define them as sectors. The basic function of the VWorld platform is to search and request for the sectors rendered on the screen according to the position and direction of the camera as controlled by a user. Every frame of the downloaded data is then rendered and displayed on the screen [5]. By classifying and constructing the data based on sectors having geometric information, it is possible to efficiently manage various types of large-capacity spatial information data.

2.2. *Aerial Images*

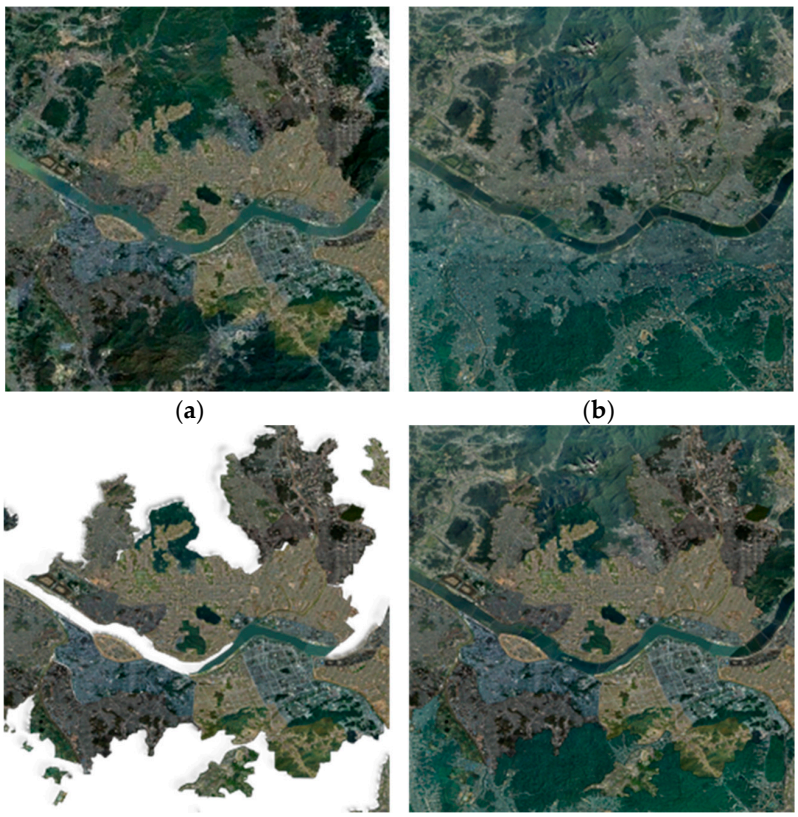
Aerial images provided by VDC have a 256 × 256-pixel resolution and comprise LOD 0 to LOD 16. The area displayed by a single aerial image at LOD 13 can be represented by four aerial images at LOD 14 or by 16 aerial images at LOD 15, as shown in Figure 6. At this time, because the resolution of each aerial image is the same, it is expressed in high quality as it is expressed at a lower level [22,23].

VWorld aerial images are provided in three layers: *earth*, *2018*, and *orthogonal image*. *Earth layer* is an aerial image covering the whole world. Meanwhile, the areas of South Korea and North Korea are updated extensively every year. Although the VDC comprises updated aerial image data, owing to

the peculiarity of the divided country, it provides only the latest year's aerial images with the latest security rules. Currently, the *2018 layer* is the latest aerial image layer. *The orthogonal image layer* is obtained or edited in the shape obtained on looking down at the building vertically from the sky. In the case of images taken by an aircraft, tilting occurs according to the height of the terrain and the building. The *orthogonal image layer* is used to improve this distortion, as shown in Figure 7c. The *earth layer* and *2018 layer* are default layers that are available at the start, and a user can select additional aerial image layers on VWorld service site [4]. If three aerial images are used simultaneously, they are superimposed and displayed as shown in Figure 7d. The *earth layer* is a jpeg file, and the *2018 layer* and *orthogonal image layer* are blended using alpha channel values in the png format.

(**a**) (**b**) (**c**)

Figure 6. Quadtree-based aerial image structure: (**a**) LOD 13, for which the total resolution is 256 × 256, (**b**) LOD 14, for which the total resolution is 512 × 512, and (**c**) LOD 15, for which the total resolution is 1024 × 1024.



(**a**) (**b**)

(**c**) (**d**)

Figure 7. Aerial image layer types: (**a**) *earth layer*, (**b**) *2018 layer*, (**c**) *orthogonal image layer*, and (**d**) superimposed results of (**a**–**c**).

2.3. Digital Elevation Model

A DEM is a data model used to represent terrain elevation [24,25]. A sector has a DEM. A DEM consists of 65 × 65 with a row-major order, and the top left corner is the origin (0, 0), as shown in Figure 8. Figures 8 and 9 show the scaled-down DEM from 65 × 65 to 5 × 5 for illustrative purposes. Aerial images have LODs up to LOD 16, but DEMs have LODs up to LOD 15. The upper LODs are generated based on the lowest LOD, which is LOD 15, as shown in Figure 9. Therefore, the DEMs with the same longitude and position have the same value regardless of the LOD. For example, the size of the DEM at LOD 15 is 65 × 65, and the size of the DEM at LOD 14 is also 65 × 65. The DEM at LOD 14 is configured by considering a DEM size of only 33 × 33 from four DEMs at LOD 15.

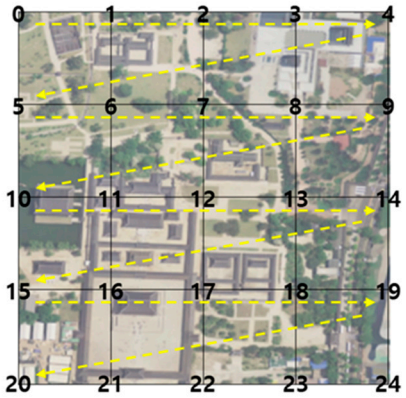


Figure 8. DEM with row-major order. DEM is scaled down to 5 × 5 size.

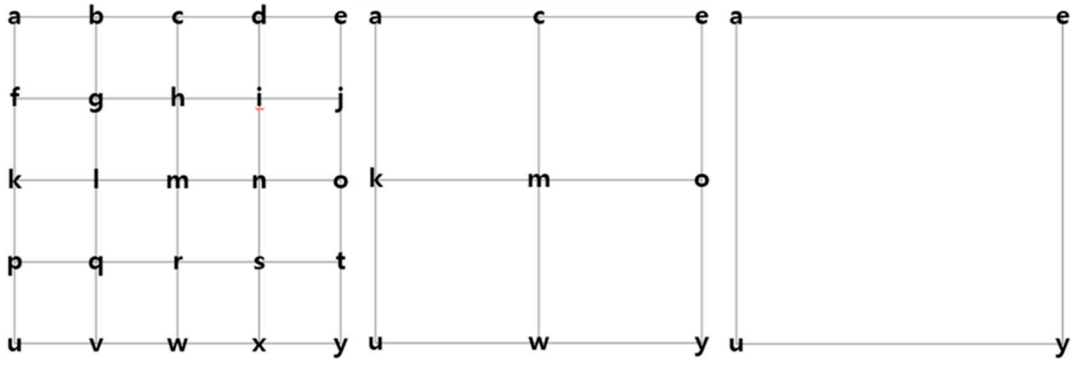


Figure 9. LOD of DEM: (**a**) DEM at LOD *n*, (**b**) DEM at LOD *n-1*, 1/4 area overlapping with (**a**), and (**c**) DEM at LOD *n-2*, 1/16 area overlapping with (**a**).

3. Visualization of 3D Terrain Object Model

The VDC does not provide 3D terrain object model objects. A 3D terrain object model has to be created by the client using an aerial image and a DEM. A sector has at least an aerial image and a DEM. In some regions, sectors may not have DEMs, in which case the DEM value of the parent sector is used. The parent sector includes a reference sector (it is called the child sector), which has one LOD less than the parent sector. This chapter describes how to request for Vworld data and create a 3D terrain object model.

3.1. Method of Requesting for Vworld Data

The VDC provides a variety of up-to-date and high-quality spatial information. APIKeys are issued to users and a method is provided for using Vworld data via DataAPIs. Even if the APIKey is issued, the act of privatizing, editing, modifying, and selling Vworld data is regulated by law.

The use of DataAPI is a method of requesting for the uniform resource locator (URL) generated based on a rule. Table 1 shows an example of the use of DataAPI based on data type. As the majority of the spatial information data is classified and managed based on sectors, the data layer at a desired position can be requested using the sector level, IDX, and IDY. For the Key value, instead of *, the private APIkey that is provided by VDC is used. IDX and IDY are determined based on sectors at LOD 0, as shown in Figure 4.

Table 1. Example of the use of Vworld DataAPI.

Type	Layer	Level	IDX, Y	URL
Aerial image	tile_mo_HD	0	0, 0	Request=GetLayer&Layer=**tile_mo_HD**&Level=**0**&IDX=**0**&IDY=**0**&Key=*
Aerial image	2018	0	0, 0	Request=GetLayer&Layer=**2018**&Level=**0**&IDX=**0**&IDY=**0**&Key=*
Aerial image	hybrid _silgam	0	0, 0	Request=GetLayer&Layer=**hybrid_silgam**&Level=**0**&IDX=**0**&IDY=**0**&Key=*
DEM	DEM	0	0, 0	Request=GetLayer&Layer=**dem**&Level=**0**&IDX=**0**&IDY=**0**&Key=*

3.2. Generation of 3D Terrain Object Model

The mesh of the 3D terrain object model is created using the latitude and longitude coordinates and the DEM in a sector. All the sectors are square shaped, and the same-level sectors have the same size. For example, in the case of a sector at LOD 0, the latitude and longitude are 36° apart, and the sectors at LOD 1 are 18° apart. Therefore, it is possible to calculate the latitude and longitude position corresponding to a DEM of size 65 × 65 that is configured at regular intervals with only the lower left corner position in a sector.

The vertex, index, and UV texture are the basic elements required to create a 3D object model. The vertex is the smallest unit that makes up a triangular mesh. Three vertices are required to define a mesh. In the implemented spatial information platform, the vertex coordinates have a 3D coordinate system based on the XDO coordinate system, as shown in Figure 10a [5,26].

XDO is a unique 3D model format that stores Vworld 3D structure models. The XDO format is a left-handed coordinate system, and the z-axis is the up vector of the camera. The WebGL coordinate system is a right-handed coordinate system, and the y-axis is the up vector of the camera. Therefore, in order to perform rendering using the WebGL Library, the transformation matrix from the XDO to WebGL coordinate system is included in the model-view matrix. The model-view matrix is an essential parameter for performing rendering using the WebGL library [8,26].

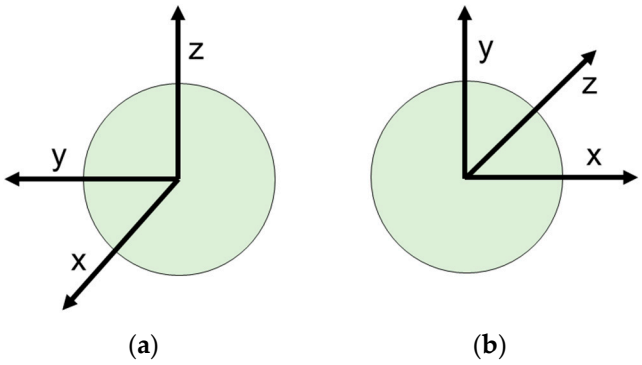


Figure 10. Coordinates system: (**a**) XDO and (**b**) WebGL (left-handed coordinates system).

In this paper, the sectors are represented by a mesh size of 4 × 4 × 2 for the purpose of explaining briefly. A sector can be used to create a 3D terrain object model of up to the size of 64 × 64 × 2 based on a DEM size of 65 × 65. Figure 11 shows an example of the vertex coordinates consisting of the latitude, longitude, and DEM for the sectors defined as LOD: 0, IDX: 5, and IDY: 2. If only the latitude and longitude values at the bottom left are determined, all the vertex coordinates can be calculated as the interval of the sector at LOD 0 is 36°.

$d_{col,row}$ is used to represent the column and row of the DEM. In the case of the sectors at LOD 0, the longitude of a sector is divided into ten from −180° to 180°, and the sixth sector from the left is from 0° to 36 °. Furthermore, the latitude of a sector is divided into five from −90° to 90°, and the third sector from the bottom is from −18° to 18°. The interval degree of the latitude and longitude is 9°.

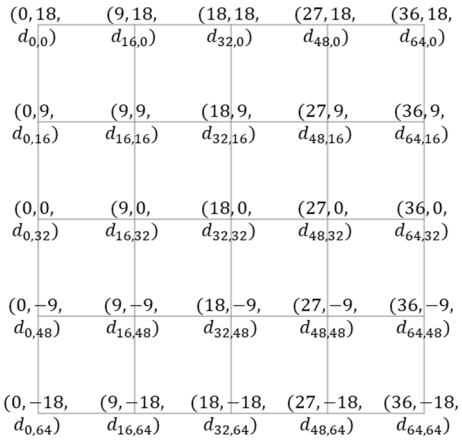
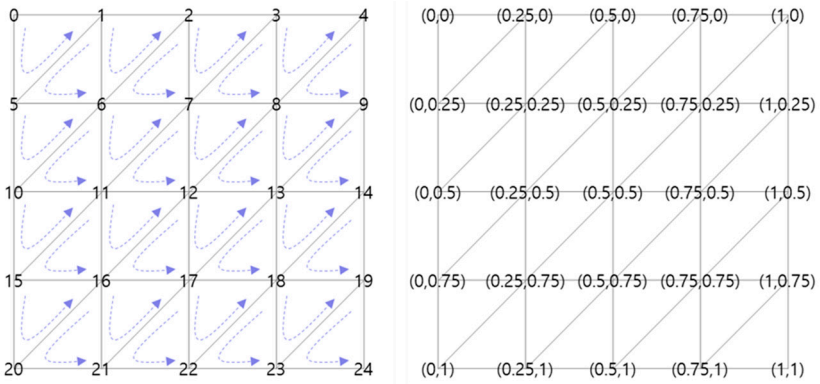


Figure 11. Example of the 3D vertex coordinates (longitude, latitude, DEM).

The latitude and longitude coordinates are transformed into the 3D coordinate vector v based on the XDO coordinates system [5]. Earth is assumed to be a sphere in the XDO coordinate system. The radius of the earth R is considered to be 6378137 m. The elevation of the terrain is the addition of R and the DEM of each location.

$$v = \begin{bmatrix} (R+d) * \cos lat * \cos lon & (R+d) * \cos lat * \sin lon & (R+d) * \sin lat & 1 \end{bmatrix}^{\mathrm{T}} \quad (1)$$

The index defines the triangular mesh as the indexing value of the vertex array. The 3D coordinates transformed from the vertex are stored in the vertex array in the order shown in Figure 12a. The corner number represents the indexing value of the vertex array, which defines a triangular mesh with the indexing values of the three vertices. For example, (0, 5, 1) indicates the triangle in the upper left corner of Figure 12a. In this case, the order of the index array is obtained in the counterclockwise direction. When the mesh size of a sector is 4 × 4 × 2, a total of 32 triangles is used. To define this model using the index array, a total of 96 index values are required to define the 32 triangular meshes.



(**a**) (**b**)

Figure 12. Example of index and UV texture of the 3D terrain object model: (**a**) index, number means indexing values stored in the vertex array and (**b**) UV texture.

The UV texture array is used for mapping a texture image into the meshes of the 3D terrain object model. The resolution of all the texture images where 3D terrain object model generation is used is

256 × 256. However, because multiple overlapping texture images can be used, the values of the UV texture array have a value between 0 and 1 without the use of the pixel coordinates. The upper left point is defined as (0, 0), and the lower right point has a value of (1, 1). In the case of the UV texture and index arrays, all the sectors use the same array when the mesh size of the 3D terrain object model is the same, in contrast to vertex arrays that are changed based on the latitude and longitude coordinates.

Box 1 lists the vertex, index, and UV texture arrays. v is the 3D coordinate vector that is transformed to the XDO coordinate system. The vertex array stores values in the order of the x, y, and z axes. Therefore, a total of 75 values are stored for 25 vertices. Ninety-six values are defined for the index array, and 50 values are defined for the UV texture array. The parentheses () in Box 1 are used to distinguish a mesh or a point. It is actually stored in the form of a 1D array.

Box 1. Examples of vertex, index, and UV texture arrays. The parenthesis () is used to group a mesh or a point unit. It is actually a 1D array.

$$\text{Vertex} = \left\{v_{0_x}, v_{0_y}, v_{0_z}, v_{1_x}, v_{1_y}, v_{1_z}, v_{2_x}, v_{2_y}, v_{2_z}, \ldots, v_{24_x}, v_{24_y}, v_{24_z}\right\}$$

Index = { (0, 5, 1), (1, 5, 6), (1, 6, 2), (2, 6, 7), . . . , (18, 23, 19), (19, 23, 24)}

UV texture = {(0, 0), (0.25,0), (0.5, 0), . . . , (0.75, 1), (1, 1)}

$$v_i = \left\{ \begin{array}{cccc} v_{i_x} & v_{i_y} & v_{i_z} & 1 \end{array} \right\}^{\mathrm{T}} \tag{2}$$

3.3. Minimizing the Gap Between Sectors at Different LODs

In the case of spatial information tiles with quadtree-based LODs, some gaps may occur when the tiles at different LODs are adjacent to each other. The main cause of these gaps is the mesh size. Figure 13 shows an example of gaps that can be observed when sectors at different LODs are adjacent. The length of one side of the green sector at LOD n is twice that of the red sector at LOD $n + 1$. In the case of a mesh size of 4 × 4 × 2, the sector at LOD n uses four meshes on the outer side, and the two red sectors at LOD $n + 1$ use a total of eight meshes. Because the DEM of the entire LOD is generated based on the sectors at LOD 15, the change in the values of the DEM is usually small in adjacent terrains. Large changes in the DEM often occur in areas such as mountainous terrains, as shown in Figure 13.

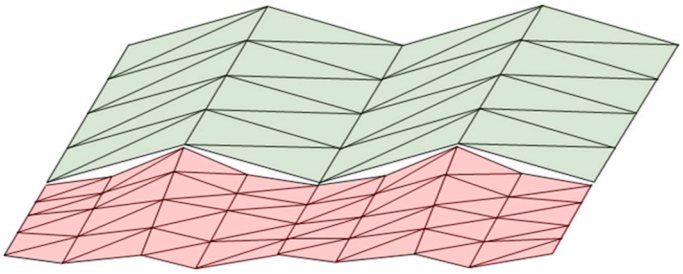


Figure 13. Example of gaps, a green sector at LOD n, and two red sectors at LOD $n + 1$.

In this paper, we use the same outer mesh size of the 3D terrain object model of the green sector at LOD n as the red sector at LOD $n + 1$ for minimizing gaps. Figure 14 shows the result obtained using the additional mesh created. Two vertices are added to represent four meshes at the four corners, while the outer corner consists of three meshes with one added vertex. Therefore, the outer mesh size of the green sector at LOD n can be the same size as that of the sector at LOD $n + 1$. In the proposed VWorld platform, the size of the DEM is 65 × 65, and up to 64 × 64 × 2 meshes can be configured. However, we limit the maximum size of the created outer meshes to 32 × 32 × 2.

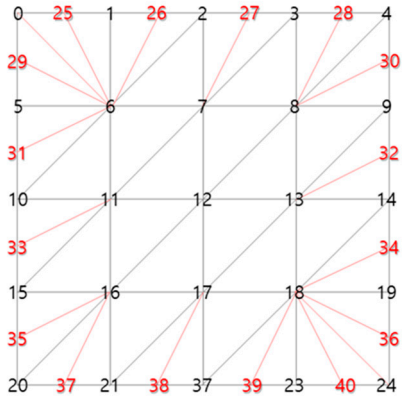


Figure 14. Added vertices (red) of the 3D terrain object model with double the number of meshes in the outer area.

Figure 15 shows an example of the use of an added 3D mesh model with the red vertices as shown in Figure 14. Because the outer mesh size of the green sector at LOD n is the same as that of the red sectors, the gaps can be minimized.

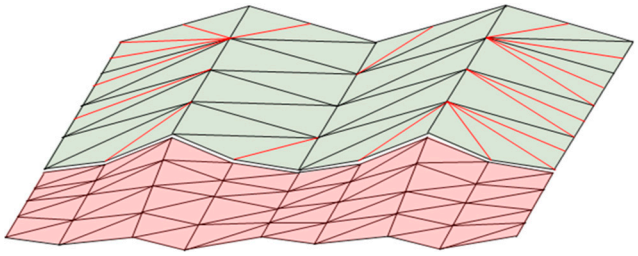


Figure 15. Example of the use of the added mesh model with the red vertices as shown in Figure 14.

The gaps still occur if the difference in the LOD of the adjacent sectors is greater than one. In this paper, we use a curtain-shaped terrain model to minimize the gaps that can occur when the difference in the LOD of the adjacent sectors is greater than two. The additional curtain-shaped meshes are lowered downwards in a manner similar to a curtain on the outer line of the existing terrain model, as shown in Figure 16.

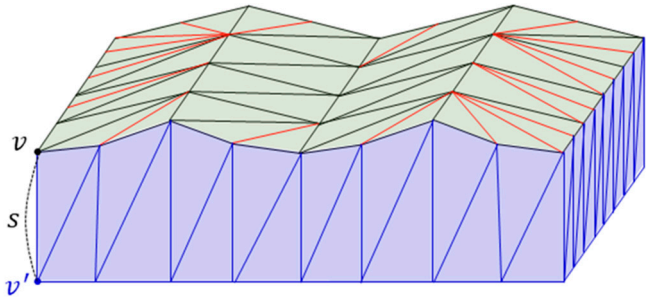


Figure 16. Example of the curtain-shaped meshes in blue.

The additional vertices of the curtain-shaped meshes are the coordinates at which the outer vertices have moved down in the original terrain. Because the 3D terrain object model corresponds to a spherical surface of the earth, the direction of the axis under the terrain cannot be referenced to a particular axis of the coordinate system. The downward direction of the terrain is opposite to the existing outer vertex vector. v' is the vector whose scale of the v is modified to $(l - s)$ as follows:

$$l = \sqrt{{v_x}^2 + {v_y}^2 + {v_z}^2} \tag{3}$$

$$v' = \begin{bmatrix} (l-s) * \frac{v_x}{l} & (l-s) * \frac{v_y}{l} & (l-s) * \frac{v_z}{l} & 1 \end{bmatrix} \tag{4}$$

Figure 17 shows the results obtained before and after using the proposed gap-minimization method in our WebGL-based platform [4]. Figure 17a,b presents the rendering result obtained with the mesh size of $16 \times 16 \times 2$ as a line-string. Figure 17c,d presents the texture rendering result obtained with the same mesh size. The 3D curvature of the original terrain model is represented by a cube-like shape without the bottom face as per our proposed method.

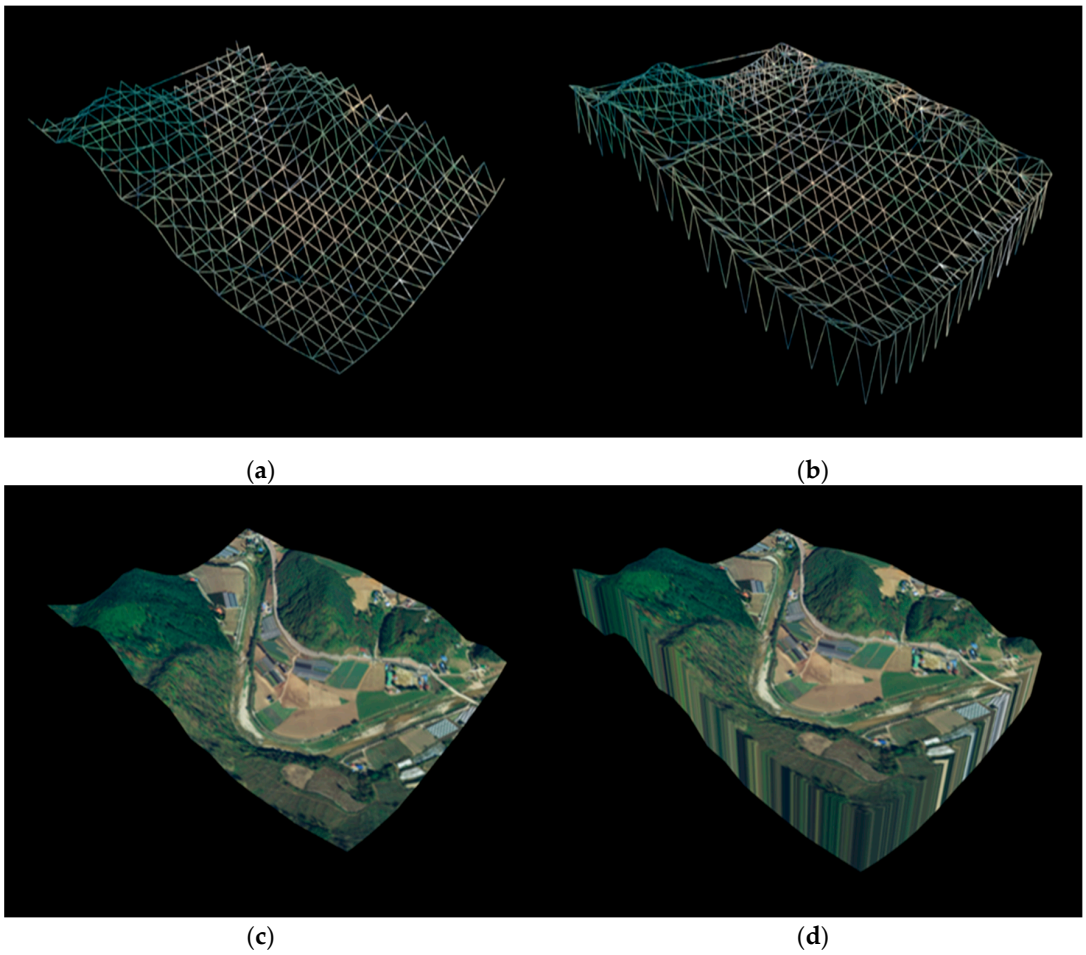
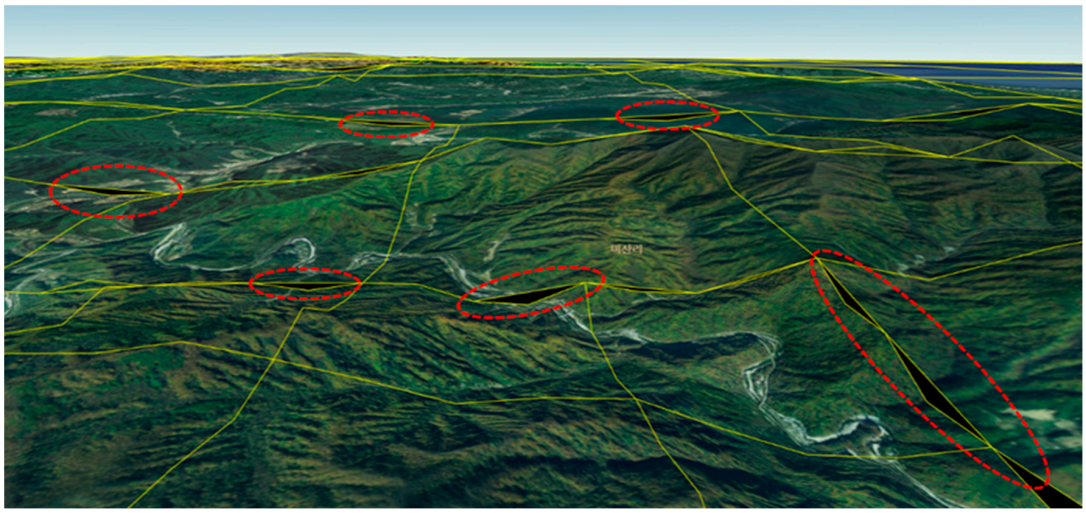


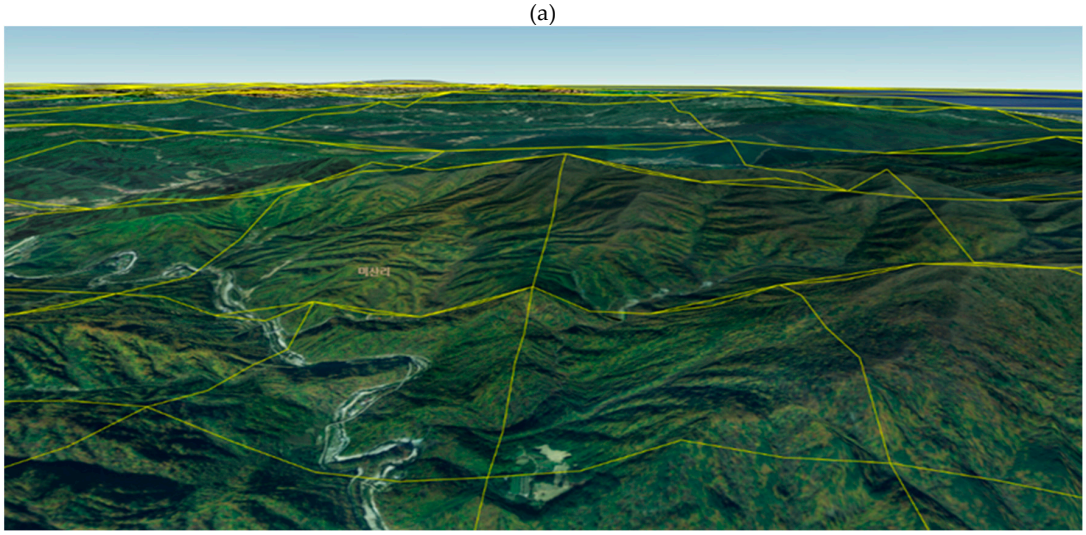
Figure 17. Result obtained using the proposed method for minimizing gaps; the mesh size is $16 \times 16 \times 2$: (**a**) line-string rendering results before improvement, (**b**) result of the method proposed in (**a**), (**c**) texture mapping results before improvement, and (**d**) result of the method proposed in (**c**).

Figure 18a,c shows the results obtained when the sectors at different LODs are adjacent to each other in the case of mountainous terrains with relatively large DEM changes. The yellow line is the outer line of the sectors. The rendering target area is Seorak Mountain, which is located in Gangwon-do Province, South Korea. The LOD of the sector rendered on the screen is determined based on the distance between the center point of the sector and the camera. Therefore, various LOD sectors can be displayed adjacent to each other. In a mountainous terrain, where DEM changes are large, frequent gaps can be observed between the sectors. When the mesh size of the 3D terrain object model is large, the occurrence of gaps is more frequent, and the size of the gaps is larger. The experimental results show that the mesh size of $16 \times 16 \times 2$ results in relatively smaller gap than those in the case of a $4 \times 4 \times 2$ mesh size. Figure 18b,d shows the results obtained using the 3D terrain object models with

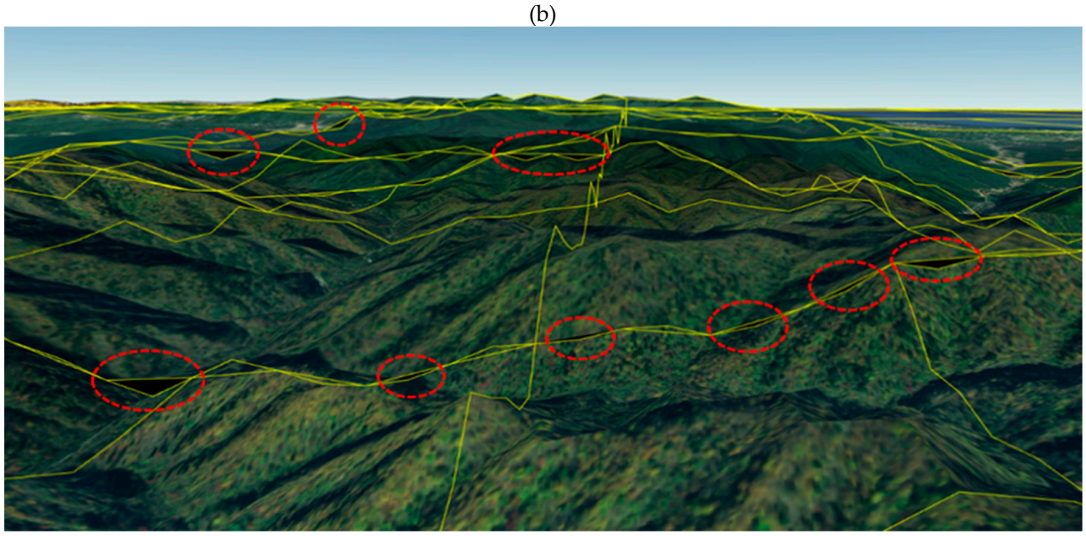
the proposed gap-minimization method. It is difficult to find gaps that occur even in a mountainous terrain where the DEM changes are large.



(a)

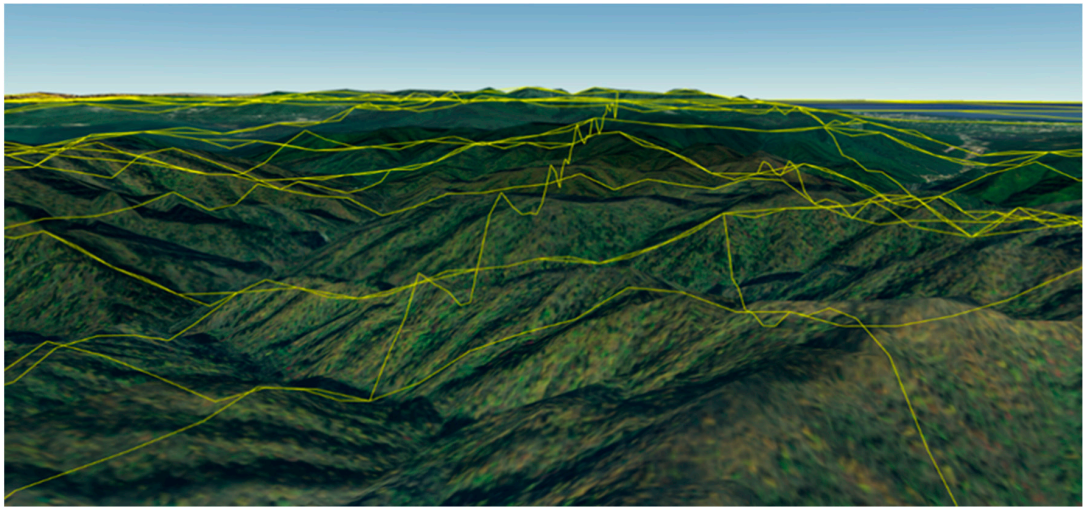


(b)



(c)

Figure 18. *Cont*.



(d)

Figure 18. Experimental results of the gap-minimization method in Seorak Mountain: (**a**) the mesh size used is $4 \times 4 \times 2$, (**b**) result of the application of the method proposed in (**a**), (**c**) the mesh size used is $16 \times 16 \times 2$, and (**d**) result of application of the method proposed in (**c**).

4. Frame-Rate Improvement for Limited Web-Based Environments

The primary objective of the proposed VWorld spatial information platform is to make high-quality, up-to-date spatial information easily available to anyone. Therefore, we used a web browser environment that does not require installation. The proposed platform also supports cross-platform with WebGL and Web standards to run on a variety of operating systems and web browsers. The rendering of a large amount of data in 3D requires a certain computing environment level. In previous research, the use of a platform that maintains 30 FPS or more even in the environment without a GPU has been proposed [5]. The processes performed in one frame can be categorized into three process categories: finding sectors to be rendered, data request, and rendering. The most time-consuming process is finding the sector to be rendered. In this section, we describe an algorithm for finding sectors to be rendered, and we propose the application of subdomains for speeding up the data request process.

4.1. Finding Sectors to Be Rendered

The process that takes the longest time on one frame is the process of finding the sector to be rendered. It is necessary to decide which sector is to be rendered from among the 71.5 billion target sectors, which have 16 steps of LODs in total. On a global basis, the LODs have 6 steps, and only South Korea and North Korea have 13 or 16 steps. The actual number of sectors is approximately 200 million sectors. However, in order to determine whether a sector is actually constructed, it is necessary to request the sector from the VDC. In the previous research [5], they used two culling tests in the quad tree-based LOD structure. Only the child sector of the sector that passed the culling test could be the test target. By reducing the number of inspection targets with only sectors that have passed, the frame speed could be improved. In this paper, we can improve upon the previous approach by adding the center-culling test and specifying the conditions for the sector to be tested. In addition, we could reduce the number of target sectors to be rendered by adjusting the *scaleFactor* in (5) that determines the *drawLevel*.

$$drawLevel = \frac{\log(R/dist)}{\log 2} \times scaleFactor \tag{5}$$

By default, the *scaleFactor* is set as 1. When the frame rate slows down, the *scaleFactor* can be adjusted to at least 0.92 to reduce the number of sectors to be rendered in a frame by 30%. R represents the radius of the earth, and *dist* represents the distance between the camera and the center of the

sector. Figure 19 shows the flow chart of the render loop function. When the frame starts, the position and direction of the camera under user control are calculated. Three culling tests are performed on 50 sectors with LOD 0.

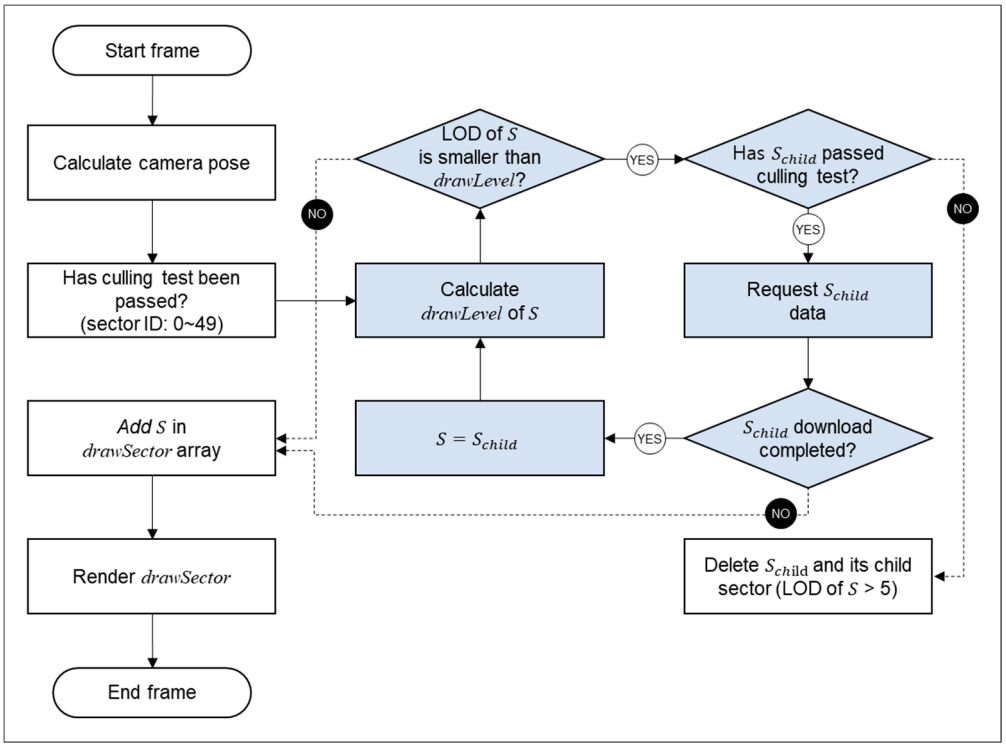


Figure 19. Flow chart of render loop function.

Figure 20 shows the flow chart of the culling test. The first center culling test calculates whether the sector is located in the center of the screen. The central sector is the sector that contains the longitude and latitude through which the camera's principal line passes. If a sector is determined as the center sector, the curling test is ended. In other cases, a sector may be rendered only if the two tests are passed.

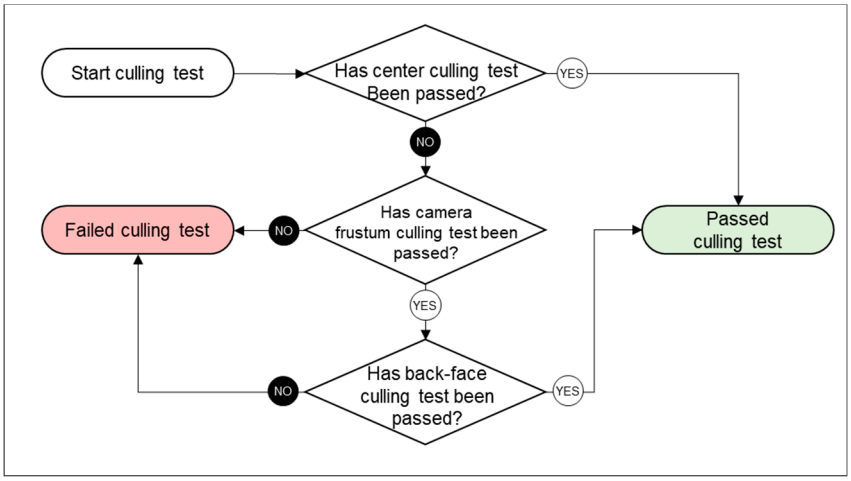


Figure 20. Flow chart of culling test.

A child sector search, marked in blue in Figure 19, is performed on sectors that pass the culling test. First, the *drawLevel* of the sector *S* is determined. When the LOD of the sector is lower than the

drawLevel, a culling test is performed on the four sectors S_{child} of S. The child sectors S_{child} are four sectors of LOD + 1 located in the region of the parent sector S. If S_{child} passes the culling test, the S_{child} data is requested from the VDC. Because the download is not completed after the request in the same frame, the parent sector S is added in the *drawSector*. If the download is completed, the target sector S is changed to the child sector S_{child}. The procedures marked in blue are repeated in a manner similar to the recursive function to the lowest LOD.

If S_{child} does not pass the culling test, S_{child} and all its child sectors are deleted. Only sectors above LOD 5 are deleted, so that the camera suddenly moves to another location. In the previous approach, the platform occupied up to 3 GB of web-browser memory. Our proposed platform deletes the sector above LOD 5 under certain conditions, and the memory occupancy is reduced to less than 1.5 GB.

In the frame rate experiment, two types of computers are used. The desktop computer with a GPU has an Intel®4.20-GHz Core™ i7 CPU, an NVIDIA GeForce GTX 1080Ti GPU, and a wired network. The notebook without a GPU has an Intel®2.70 GHz Core™ i7 CPU and a wireless network. The desktop has an external GPU, whereas the notebook does not. The web browser used is Google Chrome (version 77.0.3865.120, 64 bit). Arithmetically, there are 71.5 billion sectors, but the proposed method tests only approximately 80 sectors or less per frame. Therefore, a frame rate of more than 50 FPS can be maintained even in a computing environment without a GPU, as shown in Table 2. If the frame rate is less than 16.7 ms, the maximum speed supported by the web browser is 60 FPS. The proposed method used with a GPU was measured at 5.34 ms per frame and recorded a maximum FPS of 60.

Table 2. Result of frame rate experiment: 47 sectors, 2,560 meshes in each sector, and a total of 120,320 meshes. A sector has two textures (256 × 256 pixels). The number of culling test target sectors is 78.

	GPU (ms)	**Without GPU (ms)**
Our platform	5.34 (60 FPS)	19.01 (52.6 FPS)
Previous research [5]	6.93 (60 FPS)	25.06 (39.9 FPS)

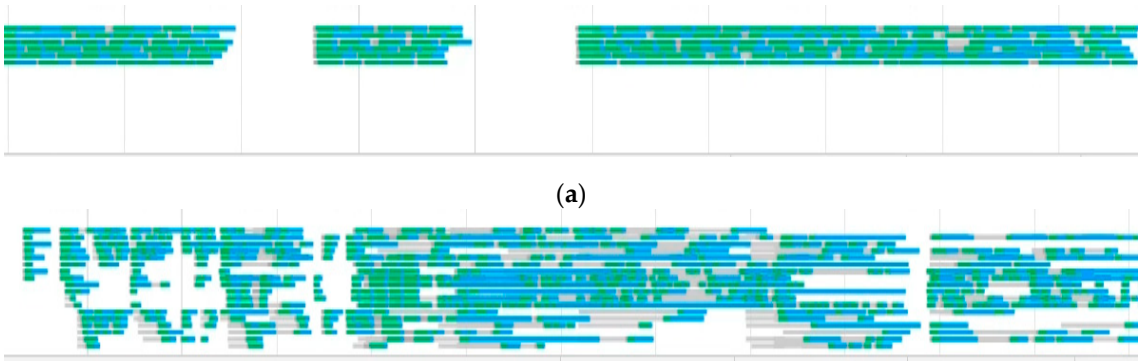
4.2. Data Request through Subdomain

Web browsers restrict the amount of data that can be simultaneously requested on a single domain. In Google Chrome, you can make up to six requests to a single domain at the same time. We use subdomains to increase the number of data requests and speed up the downloads. Subdomains are created by distributing multiple URLs on the same site. For example, if there exists a site called google.com, then scholar.google.com is a subdomain. Web browsers limit the number of simultaneous requests from one domain, but if we use subdomains, the web browser can recognize them as different domains and increase the number of concurrent requests. This is a simple method, but the experiment result demonstrated that it is effective.

As shown in Figure 21, the increase in the number of requests available simultaneously has the effect of aggregating the available networks. One consecutive row of the same color indicates that the downloading of one piece of data is in progress, while a row of the same column position indicates that a simultaneous download is in progress. Up to six pieces of data can be downloaded concurrently as shown in Figure 21a, but in case of Figure 21b, up to approximately 20 can be requested simultaneously.

The 3D terrain representation was considered to be the top priority for a natural user environment. Therefore, the DEMs and aerial images, which make up the terrain model, are requested from the newly operated subdomains, and other data, such as buildings, roads, and facility names, are used in the existing single domain. The speed improvement realized as a result of using the local server is presented in Table 3. This effect may be limited in a limited network environment. However, as the number of data that can be requested simultaneously is greatly increased from 6 to 30, this effect will improve as the speed of the network technology advances.

- During the experiment, the web browser cache was initialized each try.
- Thirty simultaneous requests were made while using four subdomains and an original single domain.
- Because the time of speed is determined by the network environment, only relative time comparisons are meaningful. The network environment changes according to the experiment time, and the experiments are alternately performed in the same order as once for the subdomain and once if not.



(**a**)

(**b**)

Figure 21. Dart request result obtained on inspecting network activity in Chrome DevTools: (**a**) with the use of a single domain and (**b**) with the use of subdomains.

Table 3. Data request completion time by number of domains: a single domain means one domain, and the case of subdomains means five domains (request data 3618, size 38.29 MB).

	Subdomain (ms)	Single Domain (ms)
Try 1	2137.66	5303.46
Try 2	1460.88	4475.67
Try 3	1340.15	4460.35

As the result of the experiment, the data download completion time was reduced by approximately 2.5 to 3 times compared to when a single domain is used. We officially requested the VDC operated by the Ministry of Land, Infrastructure and Transport for the use of subdomains. Our proposal has been accepted, and VDC now has four subdomains in addition to its existing single domain. Previously, only one domain of *http://xdworld.vworld.kr* was operated. However, a total of four subdomains within the same server are being operated as *http://xdworld0.vworld.kr*, *http://xdworld1.vworld.kr*, *http://xdworld2.vworld.kr*, and *http://xdworld3.vworld.kr*.

5. Conclusions

In this paper, we presented a web-based spatial information platform for use with VWorld. VWorld spatial information comprises a quadtree-based tile structure. Different levels of tiles may be simultaneously rendered according to the position and direction of the camera. As the mesh size of the terrain model is constant, gaps occur when different levels of tiles are adjacent to each other. We proposed the use of a method of 3D terrain object model generation to minimize these gaps. On using the proposed method, the occurrence of gaps was reduced. The proposed platform also supports real-time rendering in GPU-less computing environments. It facilitates the finding of sectors to be rendered on the screen quickly from 30 TB or more of VWorld data and render it at a rate of at least 50 FPS even in a computing environment without a GPU. Our platform facilitates the finding of all sectors to be rendered while testing only approximately 80 sectors per frame out of 71.5 billion

target sectors. Sector management algorithms have also improved web browser memory from 3.0 GB to approximately 1.5 GB as compared to the previous research [5]. In particular, we officially proposed the use of the subdomain method and applied it to VDC. Anyone can use the proposed WebGL-based spatial information platform through the VWorld map service site [4].

In our future work, we intend to develop a WebGL-based spatial information platform using Unity3D. Unity3D is a 3D development platform that anyone can easily access, and it also supports WebGL. Globe 3D map assets will be developed and distributed. We also intend to research an improved real-time rendering method for the relatively large size of point cloud data.

Author Contributions: Conceptualization, I.J., A.L.; methodology, I.J., A.L.; software, A.L.; validation, A.L., I.J.; writing—original draft preparation, A.L.; writing—review and editing, I.J.

Acknowledgments: This work was supported by a grant (19DRMS-B147287-02) for the development of customized realistic 3D geospatial information update and utilization technology based on consumer demand, funded by MOLIT, Korea.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. VWorld Data Center, operated by the Ministry of Land, Transport and Maritime Affairs of South Korea. Available online: http://data.vworld.kr/data/v4dc_usrmain.do (accessed on 23 October 2019).
2. Yu, L.; Gong, P. Google Earth as a virtual globe tool for Earth science applications at the global scale: Progress and perspectives. *Int. J. Remote Sens.* **2012**, *33*, 3966–3986. [CrossRef]
3. Jang, H.S.; Go, J.H.; Jung, W.R.; Jang, I.S. Performance evaluation of CDN method in V-World web service as spatial information open platform service. *Spat. Inf. Res.* **2016**, *24*, 355–364. [CrossRef]
4. VWorld 3D Spatial Information Open Platform Base on WebGL, 2017, Operated by the Ministry of Land, Transport and Maritime Affairs of South Korea. Available online: http://map.vworld.kr/map/wcmaps.do (accessed on 23 October 2019).
5. Lee, A.; Jang, I. Implementation of an open platform for 3D spatial information based on WebGL. *ETRI J.* **2019**, *41*, 277–288. [CrossRef]
6. VWorld 3D Spatial Information Open Platform Base on ActiveX, 2015, Operated by the Ministry of Land, Transport and Maritime Affairs of South Korea. Available online: http://map.vworld.kr/map/maps.do (accessed on 23 October 2019).
7. WebGL Overview. *Khronos WebGL Working Group*, 2019. Available online: https://www.khronos.org/webgl (accessed on 23 October 2019).
8. Cantor, D.; Jones, B. *WebGL Beginner's Guide*, 1st ed.; Packt Publishing Ltd.: Birmingham, UK, 2012; pp. 23–58.
9. Lavoué, G.; Chevalier, L.; Dupont, F. Streaming Compressed 3D Data on the Web using JavaScript and WebGL. In Proceedings of the 18th International Conference on 3D Web Technology, San Sebastian, Spain, 20–22 June 2013; ACM: New York, NY, USA; pp. 19–27.
10. Parisi, T. *WebGL: Up and Running*, 1st ed.; O'eilly Media, Inc.: Sebastopol, NY, USA, 2012; pp. 137–164.
11. Angel, E.; Shreiner, D. *Interactive Computer Graphics with WebGL, Global Edition*, 7th ed.; Addison-Wesley Professional: Boston, NY, USA, 2014; pp. 191–223.
12. Zebedin, L.; Klaus, A.; Gruber-Geymayer, B.; Karner, K. Towards 3D map generation from digital aerial images. *ISPRS J. Photogramm. Remote Sens.* **2006**, *60*, 413–427. [CrossRef]
13. Min, K.; An, K.; Jang, I.; Jin, S. A system framework for map air update navigation service. *ETRI J.* **2011**, *33*, 476–486. [CrossRef]
14. Wang, J.A.; Ma, H.T.; Wang, C.M.; He, Y.J. Fast 3D reconstruction method based on UAV photography. *ETRI J.* **2018**, *40*, 788–793. [CrossRef]
15. Samet, H. The quadtree and related hierarchical data structures. *ACM Comput. Surv.* **1984**, *16*, 184–260. [CrossRef]
16. *Cesium: An Open-Source JavaScript Library for World-Class 3D Globes and Maps*; Analytical Graphics, Inc. and Bentley Systems, 2012. Available online: https://cesiumjs.org (accessed on 23 October 2019).

17. Agrawal, A.; Radhakrishna, M.; Joshi, R.C. Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models. In Proceedings of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, Czech Republic, 31 January–2 February 2006; pp. 1–9.
18. Polack, T. *Focus on 3D Terrain Programming*, 1st ed.; Premier Press: Cincinnati, OH, USA, 2003; pp. 106–126.
19. Tang, J.; Gong, G.H. Modeling and real-time rendering technology of real large area terrain database. *J. Syst. Simul.* **2006**, *18*, 453–456.
20. Koller, D.; Lindstrom, P.; Ribarsky, W.; Hodges, L.F.; Faust, N.; Turner, G. Virtual GIS: A real-time 3D geographic information system. In Proceedings of the 6th IEEE Visualization Conference, Atlanta, NY, USA, 29 October–3 November 1995; pp. 94–100.
21. Krivokuća, M.; Abdulla, W.H.; Wünsche, B.C. Progressive compression of 3D mesh geometry using sparse approximations from redundant frame dictionaries. *ETRI J.* **2017**, *39*, 1–12. [CrossRef]
22. Haala, N.; Rothermel, M.; Cavegn, S. Extracting 3D urban models from oblique aerial images. In Proceedings of the 2015 Joint Urban Remote Sensing Event (JURSE), Lausanne, Switzerland, 30 March–1 April 2015; pp. 1–4.
23. Kim, M.; Jang, I.S. Efficient in-memory processing for huge amounts of heterogeneous geo-sensor data. *Spat. Inf. Res.* **2016**, *24*, 313–322. [CrossRef]
24. Podobnikar, T. Production of integrated digital terrain model from multiple datasets of different quality. *Int. J. Geogr. Inf. Sci.* **2005**, *19*, 69–89. [CrossRef]
25. Susaki, J. Adaptive slope filtering of airborne LiDAR data in urban areas for digital terrain model (DTM) generation. *Remote Sens.* **2012**, *4*, 1804–1819. [CrossRef]
26. Woo, M.; Neider, J.; Davis, T.; Shreiner, D. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*; Addison-Wesley Longman Publishing Co., Inc.: Boston, NY, USA, 1999; pp. 85–139.