WebTerrain: A Web-Based Multi-Functional Tool for Terrain Data

Huaxuan Gao
The Hong Kong University of Science and Technology
hgaoab@connect.ust.hk

Raymond Chi-Wing Wong
The Hong Kong University of Science and Technology
raywong@cse.ust.hk

***Abstract*—Terrain data play an important role on human activities. With the recent development of remote sensing technologies, the demand for gathering, managing, visualizing and computing of terrain data has attracted attention from both the academia and the industry. We observed that terrain computation features are missing in most of the terrain visualization applications, while existing terrain computing tools requires programming environment setup and strong computing power on the machine, which makes it hard to access by normal user. Motivated by the limitations, we proposed WebTerrain, a terrain toolkit that supports both visualization and computation for terrain data. It also requires minimal user-side setup and enables task allocation on the server.**

I. Introduction

Terrain data is crucial in our daily life. For centuries, terrain data has been the irreplaceable information for decision making in many areas such as military, agriculture, civil engineering, aviation and hydrology. Motivated by this demand, remote sensing techniques such as airborne laser scanning has been developed for collecting terrain information. The expanding accessibility of terrain data is drawing increasing interest from both the industry and academia to the field of terrain data analysis and application. There are several tools available for terrain visualization, such as Terragen [3], Google Earth [1] and NASA World Win [2].

Terragen is commercial software for photo-realistic terrain visualization, terrain creation, special effects and artistic designs.

Google Earth is a free "geo-browser" that enables the user to traverse the surface on earth based on images collected by satellites. Its large collection of images and wide coverage range of the globe surface outweigh its inaccuracy in the terrain model, making it one of the most popular terrain visualization applications.

NASA WorldWin is an open source virtual globe API. It aims at providing developers with the capability of creating 3D globe visualization easily.

For a more comprehensive review on terrain visualization tools, please refer to [8]. Besides terrain visualization tools, a terrain computation tool called Terrain Tool-Kit was proposed in [9].

Terrain Tool-kit is a multi-functional tool for terrain data. It is designed to support many different formats for terrain data, such as OFF, OBJ and PLY. It also provides algorithms for shortest surface query and terrain simplification.

Although the aforementioned terrain visualization applications provide comprehensive features for artistic design, wide range coverage of the globe and programming APIs respectively, there exists certain common limitations from the perspective of terrain data computation.

First, each application alone does not support the extensive variety of data format for the terrain models. This will cause inconvenience when users try to use their own model because if the format is not supported, they need to perform format conversion on their own. This will sometimes lead to missing, collapsed or inverted faces. Moreover, if the model needs to be used across different software, then several format conversion processes needs to be carried out.

Second, to the best of our knowledge, no terrain visualization applications provide terrain simplification and shortest surface path query functionalities. Shortest surface path query is vital in terrain data processing because it provides foundation for other queries such as nearest neighbor search, range queries, motion planning and navigation. Terrain simplification can significantly decrease the complexity of the terrain so that other computational expensive tasks, such as path finding and rendering can be performed more efficiently.

Terrain computation features like terrain simplification and shortest surface path are provided in Terrain Tool-kit. However, it requires the user side to manually install many supporting software and packages (e.g. Qt, Visual Studio, Boost and CGAL), which make it less accessible to a normal user without professional knowledge. Besides, the rendering and terrain data computation are performed on the same machine, which may lead to slow response due to the high computation power required.

Contributions. Based on the limitations discussed above, in this paper, we propose WebTerrain, a web-based multi-functional terrain toolkit. WebTerrain is designed to support a wide variety of terrain model formats and the conversion among them, provide terrain computation functionalities, require minimum user-side installation and allocate client computation tasks to the server.

The remaining of this paper is organized as follows: Section II introduces the system design of WebTerrain. Section III describes the algorithm for terrain simplification and geodesic path query. Section IV discusses the demonstration plan. Section V concludes this paper and provides some future

directions.

II. Design

A. Preliminaries

*1) **Terrain Model**:* Terrain data can be represented in either digital elevation model (DEM) or triangulated irregular network (TIN). A DEM model rasterizes the terrain surface into equally spaces grids, and stores the elevation value of each grid cell, whereas a TIN model is composed of a set of irregular 3D points and the connectivity between these points. In our research, we adopt the TIN representation, because compared to DEM, TIN models can clearly define boundaries, such as valley floors or mountain tops. TIN models are also suitable for terrain simplification when vertex removal is applied. WebTerrain provides support to several kinds of mainstream TIN models (e.g. OFF, OBJ and PLY).

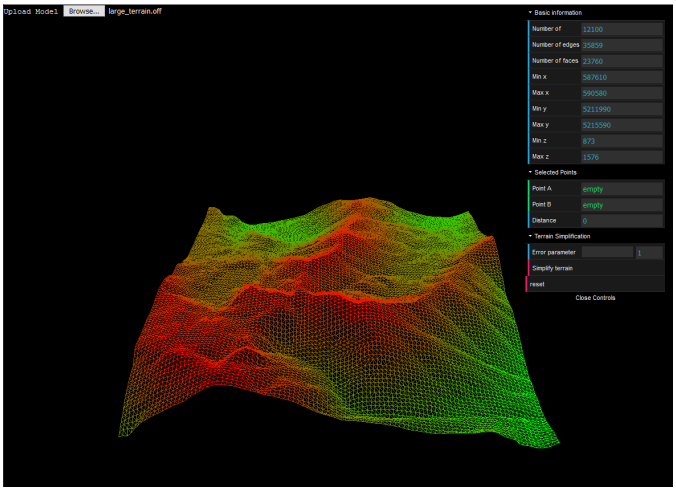

Fig. 1. Example of the terrain visualizaion using custom GLSL shader.

*2) **GLSL Shader**:* WebTerrain uses the OpenGL Shading Language (GLSL) that is executed by the graphics pipeline. GLSL was designed to provide developers direct control of the graphics pipeline without having to write hardware-specific languages. GLSL shader consists of two parts: Vertex Shader and Fragment Shader.

Vertex Shader generates a matrix indicating how to project a vertex's position in 3D space onto a 2D screen.

Fragment Shader computes the renderings of the RGBA (red, blue, green, alpha) colors for each pixel being processed. The vertex color in WebTerrain is defined based on the z value of the vertex. As shown in Figure 1, the vertices with the highest z coordinate are rendered in red, whereas the lowest vertices are in green.

We define the color of a vertex to be rgb(red, green, blue), each value ranges in [0,1]. Let p denote a point in 3D space and $p.z$ denotes its z coordinate. Then

$$r = \frac{p.z - \min\limits_{v \in V}\{v.z\}}{\max\limits_{v \in V}\{v.z\} - \min\limits_{v \in V}\{v.z\}} \quad (1)$$

where V is the set of all vertices.

The color of vertex p is then defined as *rgb(r, 1-r, 0)*.

*3) **gRPC**:* gRPC Remote Procedure Call was developed by Google to provide support for client-server communication across various programming languages including C++, python, Java, Go, .etc. gRPC adopts Protocol buffers, a language-neutral mechanism for serializing structured data to inter-change message. gRPC allows the client application to call a function defined on server side directly. For example, in WebTerrain, the client side is implemented in JavaScript whereas the geodesic path finding and terrain simplification algorithms are implemented in C++. Several tests were carried out by [4] to evaluate the performance of gRPC and REST under different settings. It was shown that gRPC outperforms REST in terms of smaller payload from serialized data and long-lasting client connection.

B. System Architecture

The architecture of WebTerrain is shown in Figure 2. Similar to most web applications, our system consists of three parts: Front-end, Middleware and Back-end. The front-end is where user interaction and rendering happens. The main process defines the user interface and user interaction logic. The render process utilize the threeJS library. The middleware is a NodeJS process which receives requests from the front-end via HTTPS and sends the request to the back-end gRPC server using proto buffers. After the server replies with the response through proto buffer, it will then forward the reply to the front-end. The back-end is a C++ process of the gRPC server. The server is listening for requests from the middleware. Upon receiving the request, it will run different algorithms based on the input and then return the output. With this architecture design, the front-end, middleware and back-end can be deployed on different machine with different levels of computation power. Thus, it achieves the efficient usage of resources. It also provides strong scalability.

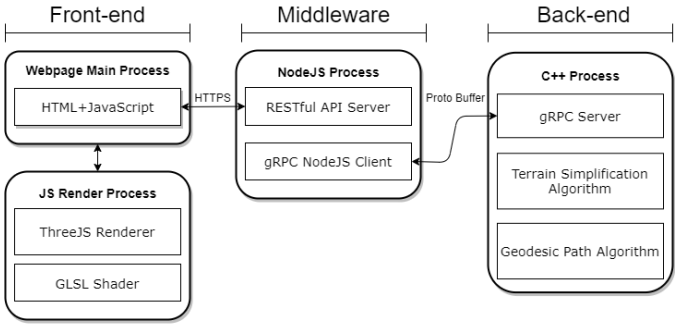

Fig. 2. System architecture of WebTerrain.

III. Algorithm

A. Shortest Surface Path

Finding and calculating the shortest surface path or geodesic path is a fundamental problem in terrain data computation. It is the basic operator in many fields such as computational

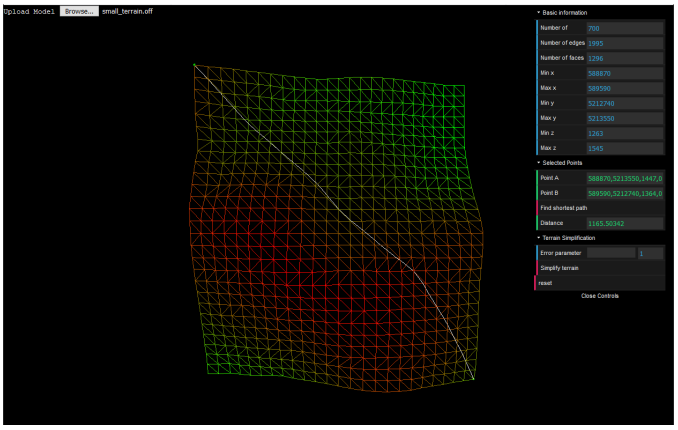

Fig. 3. Demonstration: Shortest surface path query

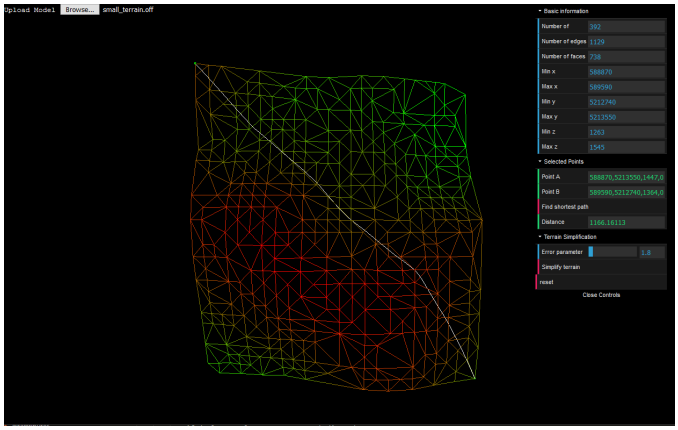

Fig. 4. Demonstration: Shortest surface query after terrain simplification with error parameter=1.8

geometry, computer graphics and computer vision. It is defined as the shortest path between two points on the 3D surface. Compared to Euclidean distance or network path distance, which defines the path strictly on edges, geodesic path query is more complex due to the curved nature of terrain surface. Theoretically, the Chen-Han (CH) algorithm [5] is the state-of-the-art exact algorithm for geodesic path finding with overall time complexity of $O(n^2)$ and space cost of $O(n)$. The algorithm is based on the assumption that each face in the terrain is flat after triangulation, which enables them to be unfolded to transform the 3D surfaces into a 2D planes. The local shortest path can then be computed as straight-line segments. The global shortest path can be computed by examining all related local shortest paths between the source point and destination point. To efficiently evaluate the local paths, the "continuous Dijkstra" method was proposed, which is an extension of the Dijkstra algorithm [6] from graph to terrain surface. The local path information on the terrain is computed and wave-front propagation in the Dijkstra algorithm can be applied. In this paper, we adopt the improvement on CH algorithm proposed by Xin et al. [10] which accelerates the algorithm by trimming redundant nodes in the graph.

B. Terrain Simplification

Due to the complexity of terrain model, simplification algorithms have been studied extensively for the purpose of efficient rendering and computation. The demand for efficient simplification algorithm is more significant in real-time rendering scenarios such as virtual reality. For the purpose of preserving the geodesic path distance between vertices, we adopt the simplification algorithm proposed in [7] which guarantees an error bound in shortest surface path and provides the flexibility to specify a scale parameter indicating the level of simplification. The algorithm adopts the vertex decimation approach for simplification. It iteratively removes vertices on the surface and check whether the resulting terrain satisfies *intra-distance property* and the *inter-distance property*.

The intra-distance property states that the distance between any two adjacent vertices of a vertex v does not change much after the removal of v.

The inter-distance property states that the pairwise distance between any neighbor of a vertex v and a previously removed vertex adjacent to v does not change much after the removal of v.

For a detailed description of the algorithm, please refer to [7].

IV. Demonstration

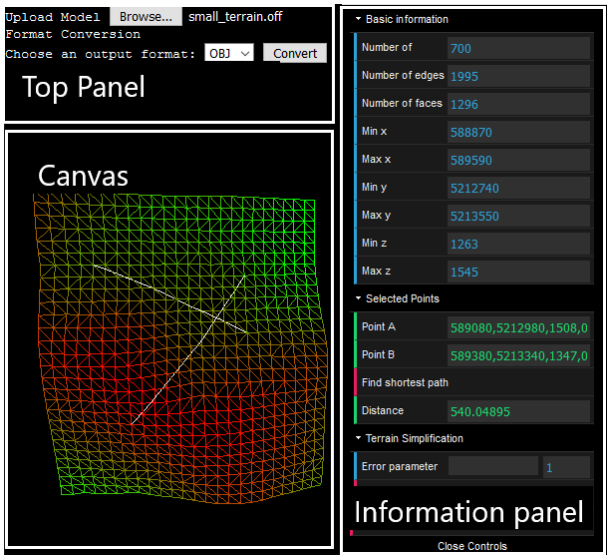

Fig. 5. The three components of the UI

A. User Interface

The User Interface (UI) of WebTerrain consists of three parts: top panel, main canvas and control panel. The three components are shown in Figure 5. The top panel provides functionalities for upload model and format conversion. The user can upload their custom terrain models by clicking the upload button and select a model file. After upload is

finished, the user can select one of the supported convertion formats from the dropdown menu: OBJ, PLY or OFF, and the converted model can be downloaded by the user. The canvas is used for rendering the terrain model. With the orbit control utility, the user can rotate, zoom and pane the model by changing the view point using their mouse. The user can also select a vertex by double clicking at it. The selected vertex will be highlighted and enlarged. The control panel contains the basic information of the terrain such as the number of vertices, edges, faces and the terrain bounding box. The coordinates of the selected vertices will also be shown upon selection. When the user has selected two vertices, the button for finding geodesic path will show up. Clicking the find path button will trigger the find path algorithm, and the shortest path will be rendered in the canvas when the computation is done. The user can specify the error parameter using the slide bar in range [1,2] for terrain simplification. After selecting the error parameter, the user can click on the simplify terrain button, and the simplified terrain will be rendered in the canvas. The terrain information will also be updated.

B. Use Case

The user can upload their models through the WebTerrain UI. The model will be parsed into a geometry object containing position information and connection information. The position information stores the x,y,z coordinates of the vertices in the model. The connection information records how different vertices are connected with each other to form triangular faces. After that, a custom shader material is generated based on the maximum and minimum z coordinates of the model. With the geometry and the material, we can generate a mesh object that can be passed into the threeJS renderer. On the other hand, the model will be transferred to the back-end gRPC server for a pre-load via the nodejs gRPC client, which is also a RESTful API server. The user can perform a shortest surface path query by picking two vertices. The selected verteices are passed to the RESTful API server using Post request in HTTP protocol. After the nodejs server receives the form data, it then passes the data to the gRPC server using proto buffer. The server runs the find path function and returns a list of points on the surface which form the shortest path. After receiving the path, the front-end will render the path as shown in Figure 3. The user can also perform terrain simplification by specifying the error parameter and click on the simplify button. The simplified terrain will be shown in the system as shown in Figure 4. Compared to the terrain before simplification, when the parameter is set to 1.8, the number of vertices decreases from 700 to 392 in the simplified terrain. By selecting the same vertices as before, i.e. the top left corner and the bottom right corner, we can again calculate the distance of the shortest path between them. According to the result, the distance before simplification is 1165.50, while the distance after simplification is 1166.16, resulting in a difference of only 0.05%. In this case, the terrain simplification algorithm can not only decrease the complexity of the terrain, but also maintain the vertex-to-vertex distance information.

WebTerrain is hosted at http://rwcpu1.cse.ust.hk/WebTerrain/. Interested readers are welcome to visit the URL and experience the functionalities we provide. We have also shot a video for demonstrating the UI of WebTerrain which could be found at https://youtu.be/zy4IAc6PPUg.

V. Conclusion

Observing that terrain shortest path query and simplification functions are missing in most visualization software, and the difficult installation process causes terrain computation software less accessible, we proposed WebTerrain, a web-based multi-functional terrain toolkit that provides both visualization and computation utilities. WebTerrain provides terrain visualization alongside with the surface shortest path query between vertices and terrain simplification functionality. No installation other than a web browser is required to use WebTerrain, which makes it easy to accessed by users without prior knowledge of terrain packages. By default, the terrain computation tasks are carried out on the server, and the client machine is only responsible for rendering, which can significantly reduce the workload on the client side. In this project, we implemented a custom GLSL shader that visualizes terrain with different color according to their altitude. We also demonstrated the use case of finding the shortest path between two vertices which confirmed that the terrain simplification algorithm is distance-preserving. There are several other features that can be added to the demo. For example, we can extend the find path algorithm from the current single-source-single-destination design to multiple-source-single-destination or single-source-multiple-destination. WebTerrain can also serves as a platform to compare the performance for different terrain computation algorithms. Besides the current two terrain computation functions (i.e. geodesic path query and terrain simplification), other terrain algorithms like nearest neighbor query can also be added to the toolkit.

References

[1] Google earth. https://www.google.com/earth/. Accessed: 2020-09-24.

[2] Nasa. nasa world win: Java sdk. https://worldwind.arc.nasa.gov/. Accessed: 2020-09-24.

[3] Terragen. photorealistic scenery rendering software. https://planetside.co.uk/terragen-overview/. Accessed: 2020-09-24.

[4] Why milliseconds matter. https://www.yonego.com/nl/why-milliseconds-matter/. Accessed: 2020-09-24.

[5] Jindong Chen and Yijie Han. Shortest paths on a polyhedron. In *Proceedings of the sixth annual symposium on Computational geometry*, pages 360–369, 1990.

[6] Dijkstra and Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[7] Manohar Kaul, Raymond Chi-Wing Wong, Bin Yang, and Christian S Jensen. Finding shortest paths on terrains by killing two birds with one stone. *Proceedings of the VLDB Endowment*, 7(1):73–84, 2013.

[8] Che Mat Ruzinoor, Abdul Rashid Mohamed Shariff, Biswajeet Pradhan, Mahmud RODZI AHMAD, and Mohd Shafry Mohd Rahim. A review on 3d terrain visualization of gis data: techniques and software. *Geospatial Information Science*, 15(2):105–115, 2012.

[9] Qi Wang, Manohar Kaul, Cheng Long, and Raymond Chi-Wing Wong. Terrain-toolkit: A multi-functional tool for terrain data. *Proceedings of the VLDB Endowment*, 7(13):1645–1648, 2014.

[10] Shi-Qing Xin and Guo-Jin Wang. Improving chen and han's algorithm on the discrete geodesic problem. *ACM Transactions on Graphics (TOG)*, 28(4):1–8, 2009.