Environmental Modelling and Software 114 (2019) 188–194

Contents lists available at ScienceDirect

Environmental Modelling & Software

journal homepage: www.elsevier.com/locate/envsoft

Implementation of interactive three-dimensional visualization of air pollutants using WebGL

Dongwei Liu[a], Jie Peng[a], Yanyu Wang[b], Meijin Huang[c], Qianshan He[a,*], Yafei Yan[a], Bingxin Ma[a], Caijun Yue[a], Ying Xie[a]

[a] Shanghai Meteorological Service, Shanghai, China
[b] Shanghai Key Laboratory of Atmospheric Particle Pollution and Prevention (LAP3), Department of Environmental Science and Engineering, Institute of Atmospheric Sciences, Fudan University, Shanghai, China
[c] Fujian Meteorological Observatory, Fuzhou, China

ARTICLE INFO

Keywords:
Three-dimension
Visualization
Pollutant
WebGL
Interaction

ABSTRACT

The visualization of a three-dimensional (3D) distribution of high-density data for scientific computations with limited computing resources under a network environment is difficult. This paper describes the implementation of an interactive 3D display of air pollutants in a network environment using the open-source WebGL class library, Three. js. A kd-tree is used to find the nearest point data to interpolate the 3D grid data of pollutants because the source data are scattered, which helps locate the data conveniently at the client's end. The binary file format is used to save and transfer data to reduce the number of network transmissions and to improve the speed of the image display. The results show that the visual effects and rendering speed are significantly improved on the web with the use of the proposed technology.

1. Introduction

3D visualization is commonly used to fully explore the 3D characteristics of natural phenomena. Because a computer screen is a 2D surface, correctly displaying the data beneath the surface layer for a 3D data package is challenging. As an alternative approach, a cross-section slice can display the exact value of each location, but with a lack of 3D visual effects. Recently, 3D visualization technology of scientific data has been widely used in medical science, geologic exploration and engineering designs (Velayutham et al., 2016; Zhou et al., 2015; Wang et al., 2014). The application of 3D visualization in the early stage mostly takes place in non-web environments and requires a high-performance computer or purchased/developed professional 3D visualization software to obtain an acceptable performance; moreover, third-party plug-ins are needed for web environments (Liu et al., 2012). Owing to the huge differences in client environments, 3D visualization for the web is likely to be blocked or difficult to upgrade; therefore, it is very inconvenient to use. Although there are currently plenty of application scenarios for 3D visualization technology for web environments (Alder et al., 2015; Jones et al., 2016), interactive 3D for web environments were only integrated into the main browsers as part of the Graphics Processing Unit (GPU) in 2011, but has since become popular (Devaux et al., 2012). Such visualization can be achieved in a web environment using WebGL and HTML5 (Parisi, 2012), without the application of plug-ins. WebGL is a cross-platform, open-source, and plug-in-free 3D graphic JavaScript application programming interface (API). It can produce interactive 3D animation on the web based on an HTML script itself, and achieve graphic rendering using low-level graphics hardware acceleration; WebGL is therefore supported by many different hardware products and browsers. The applications of WebGL-based 3D visualization have greatly increased in many different fields (e.g., medicine, construction, and geospatial application) during the past years (Li et al., 2014; Chaturvedi et al., 2015; Krämer et al., 2015; Chen & Luo, 2016).

Atmospheric pollutants in China caused by urbanization have becomes extremely serious, and the related environmental problems have received a significant amount of attention. The spatial distributions of various pollutants are quite different (Cogliani, 2001; Guttikunda et al., 2012) as the distributions depend on their chemical compositions and weather conditions. The 3D distribution of pollutants features a large amount of information, rapid variations over time, and poor regularity, and is therefore significantly different from former 3D applications. Former applications of WebGL mainly included the display of subjects with well-defined geometrical forms, and only few studies have focused on atmospheric pollutants, which have no fixed geometric forms and a highly variable distribution. Currently, commonly used 3D

* Corresponding author. Shanghai Meteorological Service, Shanghai, 200030, China.
E-mail address: oxeye75@163.com (Q. He).

https://doi.org/10.1016/j.envsoft.2019.01.019
Received 6 May 2018; Received in revised form 18 January 2019; Accepted 22 January 2019
Available online 23 January 2019

visualization schemes for scientific data include a triangulated irregular network (TIN), digital elevation model (DEM), cloud representation, spatial interpolation, and mesh grid. Because they are still mainly used in non-web environments (Gan et al., 2013), the 3D visualization of pollutants for the web remains a challenge. The first challenge for Web-based 3D visualization of pollutants comes from the limitation of local network bandwidth. Generally, the large size of 3D-data will cause inevitably delays in the uploading and browsing of 3D-data, even sometimes prevent the executing functions. The compression of 3D data in such situation becomes quite necessary, which would lead to the loss of information. Therefore, a compromise between compression rate and data integrity is needed. The second challenge is that before the WebGL standard is made and generalized, third party plug-ins are often needed for the Web-based 3D visualization, which would bring high costs due to the large variations in user's software and hardware conditions. Although the WebGL standard partly solved the problem, but the differences in user's hardware conditions still require a compromised solution between performance and display.

In the competition between network applications and desktop applications, interactive visualization of scientific data has always been a shortcoming of network applications (Congote et al., 2011). Many efforts have been devoted to develop interactive graphical applications through the internet. WebGL provides a solution for web interactive 3D visualization. Similar to WebGL, Three. js is a third-party open-source library compiled using JavaScript, and provides a number of 3D visualization functions. With the enclosed graphic interface, users can display complex 3D scenes using simple codes (Dirksen, 2013) without an understanding of complicated graphics, technical details, and the complex language of WebGL. WebGL is thus quite suitable for solving the 3D visualization problem in a web environment. Using 3D visualization of the extinction coefficient retrieved by an FX 3 satellite applying Three. js, this study successfully demonstrates the interactive 3D visualization of atmospheric pollutants in a web environment, and discusses the associated key technologies.

2. Data and methodology

2.1. Data

The simulated data are from the extinction coefficients (EC) at a height of between zero and 3 km over the East China area (23°N-39°N, 113°E − 123°E). The data represent the spatial distributions of atmospheric pollutants in East China. Our data include the latitude, longitude, height, and EC values, and the valid range of EC is 0–2 km^{-1}. The horizontal and vertical resolutions of the data are 1 km and 30 m, respectively. The actual data points change substantially owing to the weather conditions: approximately 1.6 million on a clear day, and zero under cloudy conditions.

2.2. Methodology

The display system (DS) is a single page web site based on HTML5/CSS3, the interactive UI mainly works via the mouse and keyboard. There is also a “help” label on the web, which would show the Keys and function instructions when clicked. Fig. 1 shows a flowchart of DS, which includes the data processing layer, model display layer (in-client), and data storage layer (in-server). First, the DS interpolates the scattered EC into the grid data and graphic files, and stores them in the server to receive calls of the client, which is an offline process. When the client sends out a request for data, the server will search the data based on the date and client requests, and load the data into the client's computer memory for caching through asynchronous JavaScript and XML (AJAX) (Garrett, 2007), and therefore, the model build and control of the client can be read directly from the cache. The scene control display module can adjust the observed angle, distance, and number, color, and display range of the particles. To reduce the computing time

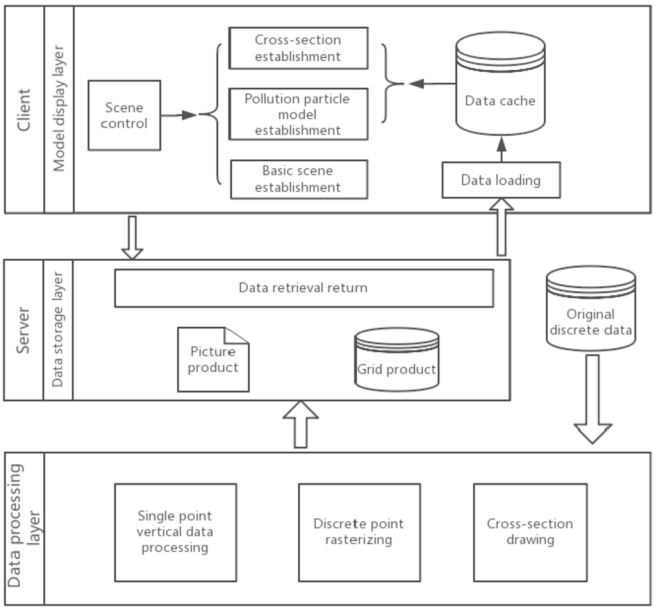

Fig. 1. Structural flowchart of display system.

of the server, the scattered data are gridded offline. Without the middle layer processes (grid), the client can obtain the data from the server in much less time. The web can only request information for the needed grid without refreshing the entire page using AJAX technology, and therefore the DS is much easier and faster to use.

2.2.1. Data processing layer

The data processing layer is responsible for the pre-processing and restructuring of the data. Because the raw dataset is scattered with a large number of points, it will take a much longer time to search. Therefore, before the display at the client, the scattered data need to be interpolated into regular grid data for a fast search of the particle location during the web-page display. To do so, the nearest neighbor interpolation is used. The spatial resolution of the raw satellite data is approximately 1 km if there are no missing data and the point density is sufficient. The nearest point is searched using kd-tree method (Bentley, 1975). The kd-tree is used for the offline interpolation, and therefore has nothing to do with the server and the client. The size of generated grid file is determined based on the type of data file and the number of grids with a valid value. The standard format of a JavaScript exchange file is JavaScript object notation (JSON, or JS), which is a light format exchange file (Gao et al., 2011) that is easy to write and read, and is therefore commonly used on the Internet. However, the data format is still a text type, and will have large size when storing grid data. To reduce the size of the file for storage and transfer, an unsigned byte format with a valid value range of 0–255 is used to store and transfer the data. Because the value range of EC is 0–2 km^{-1}, with one decimal place reserved, we multiply the EC by 10, and then store the integer part in the file and mark the missing data with 255.

2.2.2. Data storage layer

Storage of the processed data in the server is based on the data type and date in a non-structural manner, including graphs and binary files. After the client's request is received, the server will return the data storage catalogue and the data file back to the client.

2.2.3. Model display layer

The model display layer is responsible for the display of the model to the user. Specific tasks include the following:

(1) Establishment of a basic scene: This includes the map of East China,

and the grid lines of the latitude, longitude, and height. The map is generated using a map file in SVG format. The grid lines of the latitude, longitude, and height are created using a Three. Line object, the title of which is loaded and created in the scene using Three. FontLoader.

(2) Visualization of pollutant density (PD): The random ball system (RBS) based on Three. SphereGeometry is used to illustrate the PD, and the EC of any location is represented based on the color of the corresponding ball. Here, Three. SphereGeometry instead of Three. sprites is used mainly because that Three. sprites is a two-dimensional image, which is always facing the camera and could not produce effects like shadow and lighting. In contrast, Three. SphereGeometry has 3D effect, and therefore we can manually chose metal effect for the ball, and achieves a better performance under a lighting environment.

(3) Visualization of section slice: The detailed EC distribution can be presented well using the section slice of the RBS. The section slices are usually generated in two ways: (a) The section slices are created at different locations and directions using professional graphing software, stored according to certain rules, and represented when needed. Because a section slice is generated beforehand, the loading speed will be very fast using this approach. However, a slice cannot be generated at any given location. (b) Generate a section slice at a user-chosen location in real time, in which a slice can be obtained at any given location; however, the computational load of the client will increase. Both solutions are included in the DS and can be selected by the user. The location and color for the vertex of each grid are needed to apply a section slice generated in real time. With Three. js, a surface is a triangle formed by three vertices, and a rectangle is the combination of two triangles.

The process is shown in Fig. 2, which takes a horizontal surface as an example:

(a) Divide the section slice into an area of n (the number in the latitude direction) × m (the number in longitude direction).
(b) Set up the location for each vertex in the "position" parameter of BufferGeometry: Three elements are needed to represent the (x, y, z) location in the scene, and nine elements are needed to form a surface.
(c) Set up the color for each vertex in the "color" parameter of BufferGeometry: Three elements are needed to represent (r, g, b), and nine elements are needed to form a surface.
(d) Calculate the position of each vertex based on (x, y, z) and draw the color based on (r, b, g).

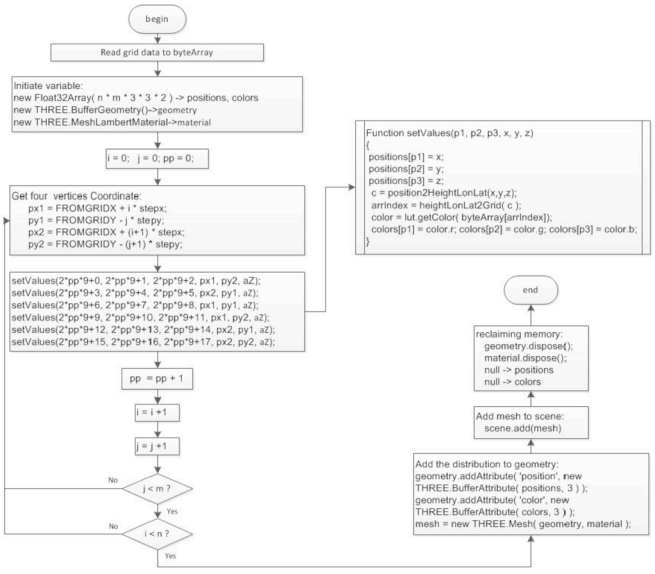

Fig. 3. Flow chart used to generate cutting section images in real time, where "aZ" represents the height of the section. The function "setValues" is used to set the position and color attributes. The function "position2HeightLonLat" is used to convert the position coordinates in the scene into the height, latitude, and longitude. The function "heightLonLat2Grid" is used to obtain the index in the byteArray, which contains the serial number according to the height, latitude, and longitude, and byteArray is stored in the background using ready grid data through interpolation processing. (4) The data load: The client's browser uses AJAX to read the grid data, which are cached in the format of ArrayBuffer, and are used to create and apply the model.

A flowchart of this is provided in Fig. 3.

3. Results

3.1. Advantage of the data format for storage and exchange

Fig. 4 indicates the storage and exchange of binary data. There are 101 blocks in height, and each block includes 161 and 121 points in the latitude and longitude directions, respectively. Each point is a 1-byte integer (0–255). Therefore, there are 101, 161, and 121 steps for the interpolation in the vertical, longitude, and latitude directions,

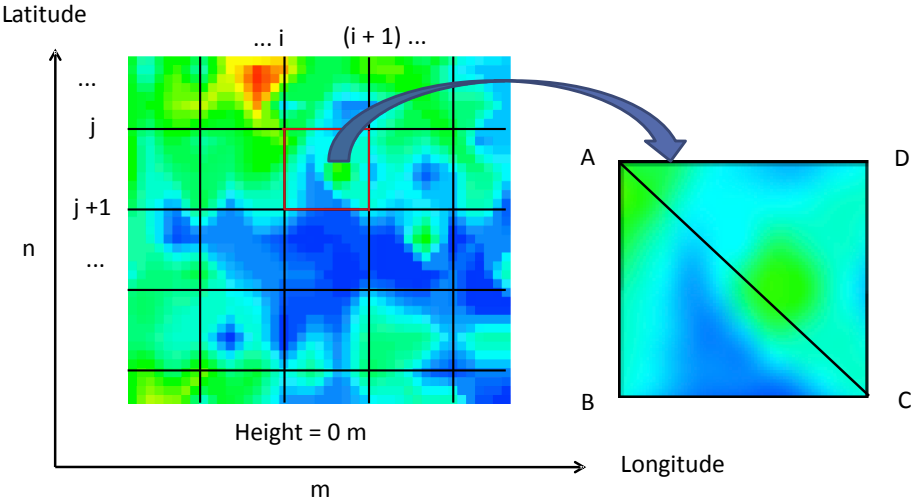

Fig. 2. Schematic diagram used to generate real-time cutting section images. Each level of the cutting section images is divided into n × m rectangular regions, each of which consists of two triangular regions (A, B, C) and (A, C, D) in WebGL.

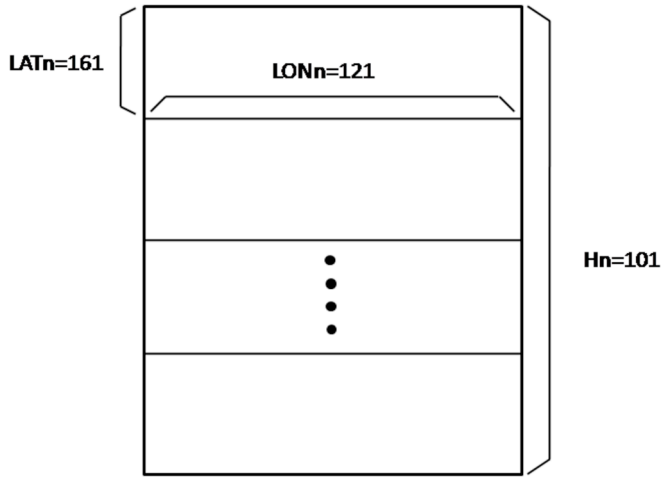

Fig. 4. Binary data file storage structure (Hn is the height block number, LATn is the number in latitude, and LONn is the number in longitude).

respectively, which generate a binary file with a size of approximate 1.88 M. If the data are stored in JSON format, all data require at least 4 bytes, leading to a binary file of 7.51 M. Apparently, the current format of the data file for storage and transfer will reduce the amount of transferred data by at least 75%, and will therefore significantly increase the web-page loading speed.

3.2. Advantage of the selected visualization method

Because the particles in the back layer are blocked by the particles in the front layer when using the perspective camera in any direction, the particle density is not too large in the RBS. Therefore, many locations will be easily missed. However, using the random location and interactive rotation, the observation depth will be significantly enhanced. Theoretically, no location will be missed when applying multiple refreshments of a random location. Fig. 5 illustrates the difference in display between a regular and random location generation. The result clearly shows that, random location could display much deeper location and more highly polluted regions.

3.3. Advantage of interactive method

In this study, the server hardware is an Intel(r) e5-2630 V2 2.60 G Xeon(r) CPU, and Windows Server 2008 R2 Enterprise Edition and Apache HTTP Server are installed in the server, using HTML5 plus JavaScript for the development. The section slice figure (PNG files) is created using the NCAR command language (NCL). Because too many

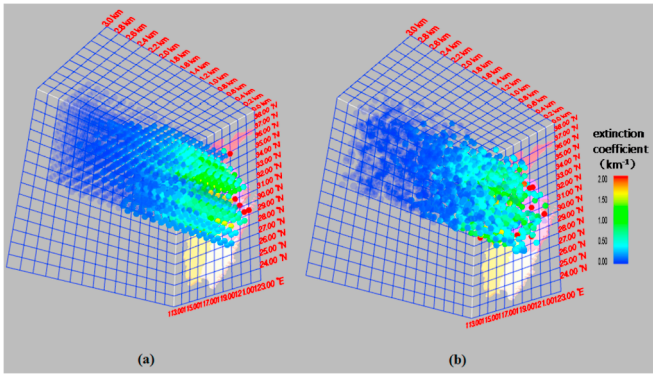

Fig. 5. Effect of 3D environmental data display for (a) regular position arrangement (2032 particles) and (b) random placement (1996 particles) in "particle system".

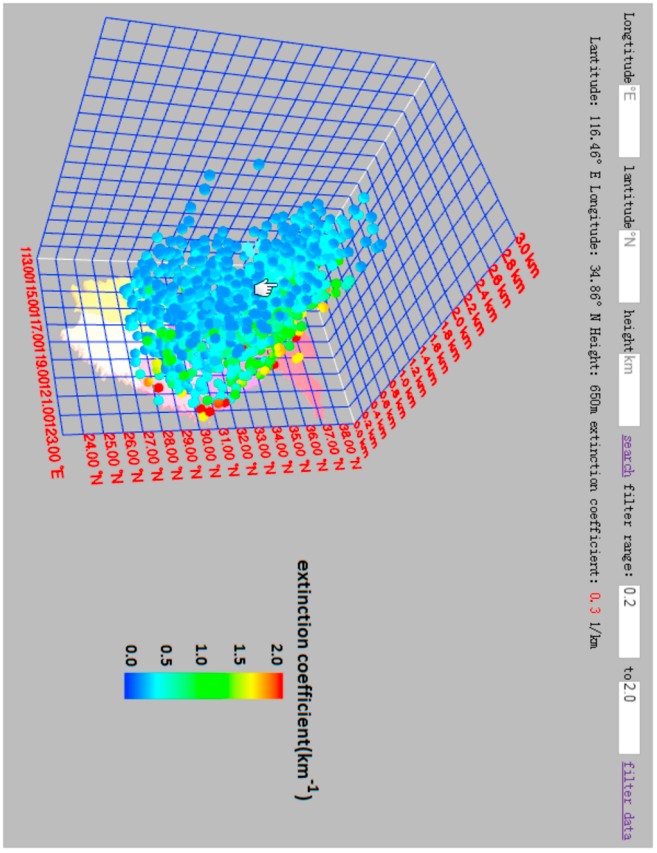

Fig. 6. Example display of filter extinction coefficient data.

small-sized PNG files will reduce the performance of the server, we combine the files into a large PNG file based on the direction, and use UV mapping to select the needed part for the display. To enhance the experience of user interaction and the adoption of different hardware, the following design is used:

(1) A particle in the RBS will be given a "value" parameter to store its EC, and the range of EC for display can be chosen and changed in the display module. The location information (latitude, longitude, and height) and EC for each particle will be displayed when pointed at by the mouse (Fig. 6). {{}}

(2) The number of randomly generated balls in the RBS can be determined by the users according to their client performance and experience. Because the number of parts in the longitude (widthSegments) and latitude (heightSegments) directions for the selection of the basic elements (Three.SphereGeometry) will impact the number of triangular surfaces, the users should be given a suitable number (a ball will appear insufficiently round with too small a number, whereas the drawing speed will be too slow if the number is too large). Using a bandwidth of 50 M, an AMD Phenom(tm) II X4 B97 3.2G CPU, 4 GB of DDR3 memory, an NVIDIA GeForce 405, and Google Chrome 58.0.3029.110 as the web browser, the relationship between the response time and the particle number is as shown in Fig. 7. The display time for a particle number of up to 16,000 can be completed within 2 s when setting widthSegments and heightSegments to 8 and 4, respectively. With 32,000 particles, however, the time increases sharply to 43 s, and therefore the approximate number should be smaller than 16,000.

(3) The rotation and zoom-in/out of camera, as well as the increase/reduction of the particle number and the display of the section slice, are all achieved using the keyboard and mouse for a concise web page. The functions of the keyboard and mouse are listed in Table 1.

(4) Reorganize the RBS, longitude/latitude, and grid into one array

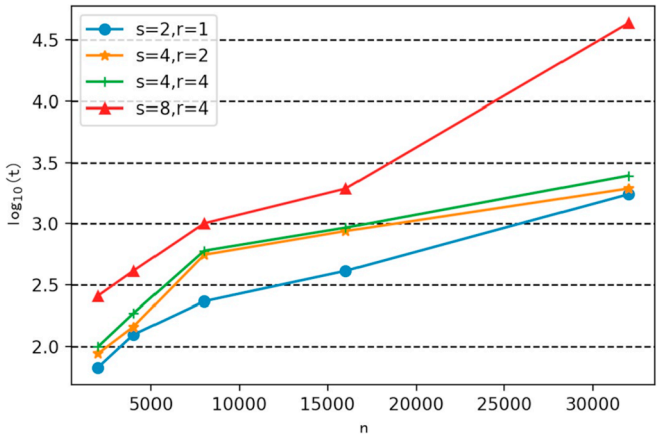

Fig. 7. Relationship between the number of particles and the display response time, where t is the response time in milliseconds, n is the number of particles, s is the section number in the longitude direction, and r is the section number in the latitude direction.

Table 1
Keys and function instructions.

Keys	Functions
The direction key	Adjust the position of the camera
f/n	Pulls camera farther/closer
Left mouse button + move	Adjusts the angle of the model
Spacebar	Resets to original position of the model
a/d	Increases/decreases the number of particles
s	Hide/display particles
x/y/z	Show the warp/weft/horizontal sections
c	Hidden slice map
v	Changes the color table

(group subject in Three. js). When selecting the section slice, the entire RBS can be rotated by rotating the group along the x-y axis. The user can easily observe any angle simply by rotating the RBS with the mouse and adjusting the angle of the camera to find a suitable angle to observe any section slice (Fig. 8).

3.4. Performance evaluation

In order to evaluate the performance of WebGL in term of its interactivity, tests have been performed with different hardware configurations. The hardware configurations of the computers used in evaluations are:

Hardware configuration A(HA): Intel(R) Core (TM) i7-6700 Processor (8CPUS), 3.40 GHz, 19.6G of RAM and Intel(R) HD Graphics 530; Hardware configuration B(HB): Intel(R) Core (TM) i5-5200U Processor (4CPUS), 2.20 GHz, 4G of RAM and Intel(R) HD Graphics 5500; Hardware configuration C(HC): AMD Phenom (tm) II X4 B97 Processor (4CPUs), 3.2GHZ, 4G of RAM and NVIDIA GeForce 405; The client browser is Google Chrome (v58.0.3029.110). The frame rate was recorded by Chrome browser's development tool “Rendering- > FPS Meter”. For the interaction with the web, we take 24 FPS as the acceptable frame rate. As seen in Table 2, HA hardware configuration showed the best performance, which can support over 6000 particles. HB and HC can support up to about 4000 particles. The user can adjust the number of particles to fit the different hardware configurations.

Generally, there are two rendering methods on the HTML5 web page: Canvas 2D rendering and WebGL rendering. Canvas 2D provides many functions to draw 2D objects, including a set of process-based rendering interface that start and end with “beginPath” and “closePath”. With low development costs, it is easy to use. But when there are too many objects to render, or the real-time interaction is

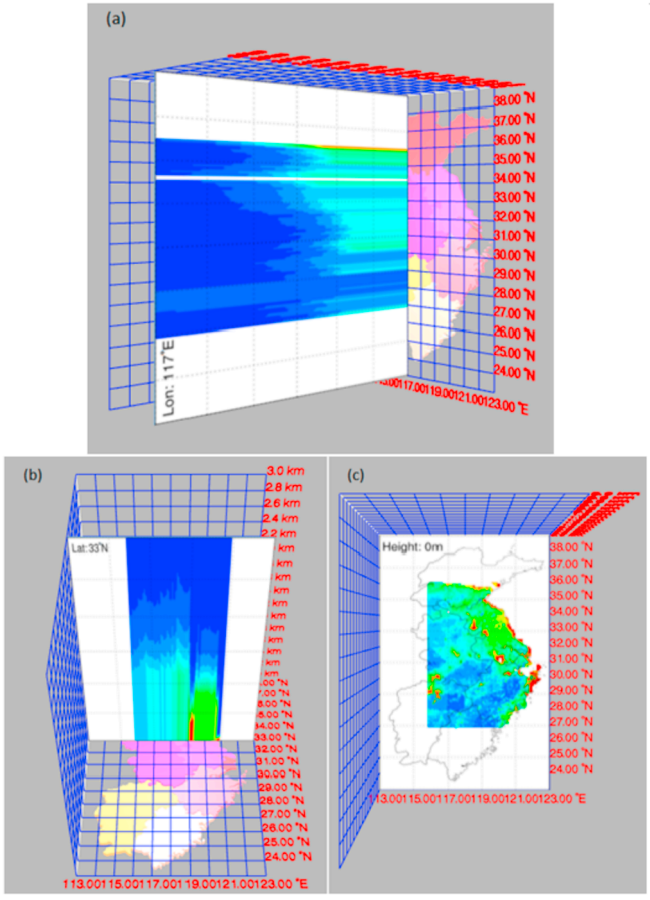

Fig. 8. Display effect in different directions: (a) meridional, (b) latitudinal, and (c) horizontal display.

Table 2
Frame rate (FPS) changes with particle number in different hardware configurations.

Hardware configuration	Particle number								
	1000	2000	3000	4000	5000	6000	7000	8000	9000
HA	60	60	60	52	37	28	21	16	14
HB	60	60	35	28	21	14	11	8	7
HC	60	60	39	31	22	15	12	9	7

required, Canvas 2D rendering method is unable to meet the requirements. This is illustrated here by comparing the performance of the two rendering methods. We apply the two rendering methods on Three. JS with hardware configuration HA. Different display effects by the two method with test interface from the tool “Rendering- > FPS Meter” are shown in Fig. 9. The WebGL rendering method shows better three-dimensional effects (e.g. light and shadow) than Canvas 2D rendering method. And when displaying 1000 particles, the frame rate of the WebGL rendering method is kept at 60 FPS, while that of the Canvas 2D rendering method drops to 1.7 FPS, which is unable to satisfy the real-time interaction requirement. Fig. 10 shows changes in the frame rate of the two rendering methods corresponding to the increase of number of particles. The WebGL method does not fall lower than the standard requirement line until the number of particles exceeds 6000. In contrast, the canvas 2D cannot meet the animation standard, and it is always below 1.7 FPS.

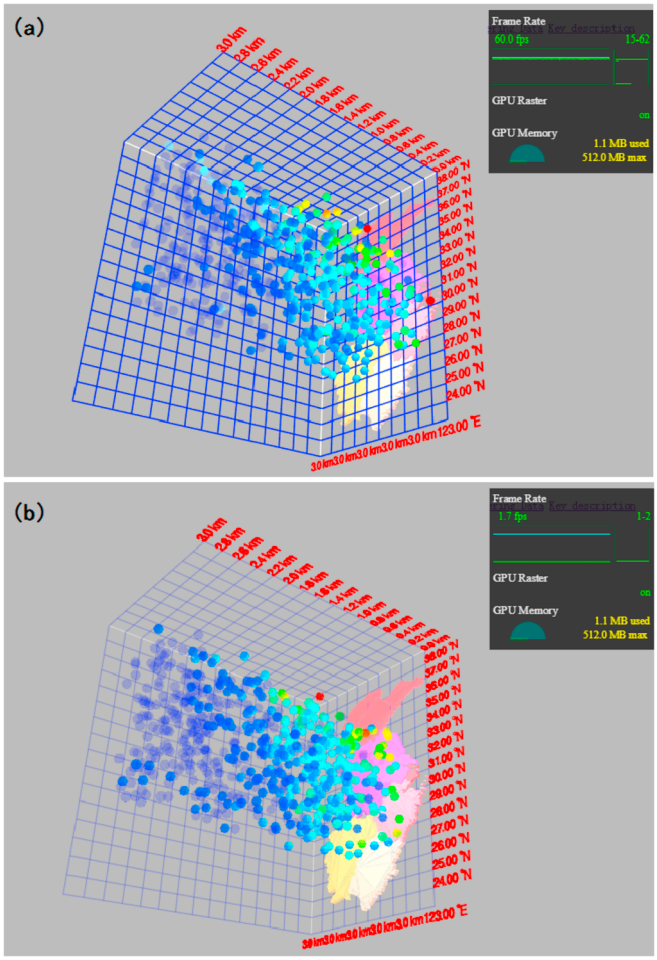

Fig. 9. Display effects using the (a) WebGL rendering method and (b) Canvas 2D rendering method.

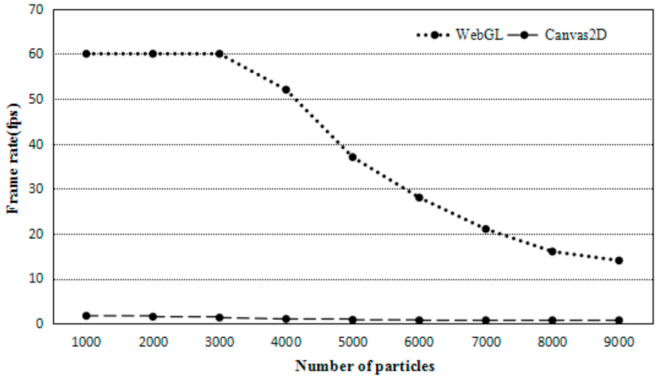

Fig. 10. Change of the frame rate with the number of particles by WebGL rendering method and Canvas2D rendering method.

4. Conclusion and discussion

4.1. Conclusion

This paper presented a method for the interactive 3D visualization of atmospheric pollutants in web environments based on Three. js. Through the introduction and analysis of the method and key technology, we have the following:

(1) Three. js can realize the interactive visualization of the 3D distribution of atmospheric pollutants, and the visual effect and display speed can satisfy the user requirements.
(2) An option is included for the users to change the number of particles for display in the system design, allowing the users to adjust and enhance their experience based on the hardware conditions of the client.
(3) Data reorganization is very important in the process of 3D visualization; the amount of data exchange between the server and front display, and the calculation of data in the front display, should be reduced as much as possible to increase the display speed.

5. Discussion

Using the requestAnimationFrame function and setInterval function in Three. js, or a third-part library (e.g., Tween. js), and adjusting the location of the camera or model, an animation display and better 3D visual effects can be achieved. If the datasets is large and the model is complex, level of detail (LOD) model (Hussain et al., 2003) can be used to optimize the performance of animation using the object THREE. LOD in package Three. js Such technology can also be used in the 3D display of pollutants in water, meteorological data, or satellite retrieval, as well as other types of scientific data. After setup in the server, only a browser supporting WebGL is required for use by the client. Because WebGL is already supported by many different web browsers, it can be easily applied. In addition, because GPU is used by WebGL to speed up the display speed, a high-performance graphics card can greatly enhance the display effect.

This paper provides a new method for three-dimensional data displaying. Further improvements are required in displaying multi-elements that directly rendered in the foreground and better handling of large amount of data. Until now, there are only few available toolkits based on WebGL for scientific data rendering. If more such toolkit is available, it could save the development cost of three-dimensional scientific data displaying greatly, and therefore, it could be the work to do next. WebGL can directly control the hardware devices of the client, therefore how to improve the security of WebGL is another important research topic.

Acknowledgements

This study was partially supported by the National Natural Science Foundation of China (NSFC, Grant Nos. 91637101), the Shanghai Science and Technology Committee Research Special Funds (Grant No. 16ZR1431700) and the National key research and development plan (Grant 2017YFC1501701). We would like to thank Elsevier Language Editing Services for their language modification.

Software availability

Name of software 3D air pollutant visualization for East China
Developers Dongwei Liu, Qianshan He, Yafei Yan
Contact Dongwei Liu: liudongwei2000@163.com, +86,021 54892930
Qianshan He oxeye75@163.com
Year first available 2017
Hardware required NA
Software required Browsers that support webGL
Operation system required OS independent
Program language HTML5/Javascript
Program size ~1 MB
Availability and cost Open source (no cost)
Repository https://github.com/liusir2000/visAirPollutant
Webpage https://github.com/liusir2000/visAirPollutant

References

Alder, J.R., Hostetler, S.W., 2015. Web based visualization of large climate data sets. Environ. Model. Softw 68, 175–180.

Bentley, J.L., 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18 (9), 509–517.
Congote, J., Segura, A., Kabongo, L., et al., 2011. Interactive visualization of volumetric data with WebGL in real-time, 3d technologies for the world wide web. In: Proceedings of the, International Conference on Web 3d Technology, Web3d 2011, Paris, France, June. DBLP, pp. 137–146.
Chaturvedi, K., Yao, Z., Kolbe, T.H., 2015. Web-based Exploration of and interaction with large and deeply structured semantic 3D city models using HTML5 and WebGL.Bridging Scales-Skalenübergreifende Nah-und Fernerkundungsmethoden, 35. Wissenschaftlich-Technische Jahrestagung der DGPF.
Chen, Z.Y., Luo, F., 2016. Revit three-dimensional building model reconstruction based on WebGL. Journal of Zhejiang University of Technology (in Chinese) 44 (6), 608–613 2015.
Cogliani, E., 2001. Air pollution forecast in cities by an air pollution index highly correlated with meteorological variables. Atmos. Environ. 35 (16), 2871–2877.
Devaux, A., Brédif, M., Paparoditis, N., 2012. A web-based 3D mapping application using WebGL allowing interaction with images, point clouds and models. International Conference on Advances in Geographic Information Systems. ACM 586–588.
Dirksen, J., 2013. Learning Three.Js: the JavaScript 3D Library for WebGL. Packt Publishing Ltd, pp. 14–17.
Gan, Z.Y., Cai, J.F., Hu, J.G., 2013. Analysis of the advantages and disadvantages of several visualization schemes of carbon dioxide based on OpenGL and ArcEngine. Modern Computer 7, 3–7 (in Chinese).
Gao, J., Duan, H.C., 2011. Research on data transmission efficiency of JSON. Computer Engineering and Design (in Chinese) 32 (7), 2267–2270.
Garrett, J.J., 2007. Ajax: a new approach to web applications. Online. Available: http://www.adaptivepath.com/publications/essays/archives/000385.php.
Guttikunda, S.K., Gurjar, B.R., 2012. Role of meteorology in seasonality of air pollution in megacity Delhi, India. Environ. Monit. Assess. 184 (5), 3199–3211.
Hussain, Muhammad, Okada, Y., Niijima, K., 2003. A fast and memory-efficient method for LOD modeling of polygonal models. In: International Conference on Geometric Modeling & Graphics IEEE.
Jones, A.S., Horsburgh, J.S., Jackson-Smith, D., et al., 2016. A web-based, interactive visualization tool for social environmental survey data. Environ. Model. Softw 84, 412–426.
Krämer, M., Gutbell, R., 2015. A case study on 3D geospatial applications in the web using state-of-the-art WebGL frameworks. In: Proceedings of the 20th International Conference on 3D Web Technology. ACM,2015, pp. 189–197.
Li, H., Leung, K.S., Nakane, T., et al., 2014. iview: an interactive WebGL visualizer for protein-ligand complex. BMC Bioinf. 15 (1), 56.
Liu, A.H., Han, Y., Zhang, X.L., et al., 2012. Research and implementation of network 3D visualization based on WebGL Technology. Geospatial Information (in Chinese) 10 (5), 79–81.
Parisi, T., 2012. WebGL: up and Running. O'Reilly Media, Inc.
Velayutham, V., Fuks, D., Nomi, T., et al., 2016. 3D visualization reduces operating time when compared to high-definition 2D in laparoscopic liver resection: a case-matched study. Surg. Endosc. 30 (1), 147–153.
Wang, Peng, 2014. Three-dimensional visualization of colliery TEM data base on MATLAB. Prog. Geophys. 29 (3), 1277–1283.
Zhou, Y., Dao, T.H.D., Thill, J.C., et al., 2015. Enhanced 3D visualization techniques in support of indoor location planning. Comput. Environ. Urban Syst. 50, 15–29.