www.cafetinnova.org

Indexed in
Scopus Compendex and Geobase Elsevier,
Geo-Ref Information Services-USA, List B of Scientific Journals, Poland, Directory of Research Journals

ISSN 0974-5904, Volume 09, No. 05

International Journal of Earth Sciences and Engineering

October 2016, P.P.2315-2320

Three-Dimensional Simulation on Urban Waterlogging and Seeper Based on WebGL

ZHANG CHENGCA, ZHANG LINGJIE AND ZHU ZULE
School of Water Conservancy & Environment, Zhengzhou University, Zhengzhou 450001, China
Email: *zhangcc2000@163.com*

Abstract: Urban waterlogging refers to the seeper disaster caused by heavy or continuous precipitation that exceeds the city's drainage capacity. A heavy and focused rainfall, together with large hardening area, can make the rainwater difficult to infiltrate and easily form urban waterlogging. This article achieved the network visualization of three-dimensional (3D) scenes of city's main flooded roads based on WebGL. Owing to its a series of advantages, such as being free from browse plug-ins, acceleration by directly calling GPU and Javascript-based strong interoperability, WebGL can make the browse more smooth, the rendering more colorful and the operation more convenient and truly reconstruct the city's waterlogging scene by conducting 3D dynamic simulation on the submerging process. The combination between 3D visualization and urban rainfall and flood simulation can reproduce urban waterlogging process and thus provide a new way of investigating urban waterlogging and seeper.

Keywords: *WebGL; urban waterlogging; terrain rendering; three-dimensional (3D) simulation*

1. Introduction

In recent years, urban waterlogging and seeper issue has become increasingly prominent, which greatly affected urban residents' daily work. More seriously, it can cause damage to urban infrastructure and threaten people's lives and property. Three-dimensional (3D) visualization technology can visually display, reconstruct and simulate urban waterlogging and seeper in all directions. Thus, the combination of 3D visualization technology and urban rainfall flood simulation is of great significance.

In 1971, with support from U.S. Environment Protection Agency, the University of Florida (UFL), Metcalf & Eddy Company (M&E), Water Resource Company (WRE) jointly developed SWMM model, which marked the advent of computer modeling era of urban drainage network. Afterwards, various urban drainage models such as the drainage simulation model for Illinois (ILLUDAS), sheetflood storage and processing model developed by U.S.

Army Corps of Engineers (STORM) and urban runoff model developed by the University of Cincinnati (UCURM) appeared successively. In the meantime, the researchers also developed many urban flood hygrograph and rainfall flood simulation models, such as ISS, HSP, DR3M-QUAL, LAVRENSON, CAREPAS, QQS, RATIONAL and WFP. These in-depth studies on the simulation model of urban drainage system have laid the foundation for the subsequent development of urban flood simulation software.

On October 29, 2014, the World Wide Web Consortium (WSC) announced that HTML5 standard were finally formulated and successfully published. Some technologies of HTML5 have been put to use in succession. Based on 3D function of Canvas, WebGL, SVG and CSS3, the visual effects presented in the browser-side are amazing. Specifically, based on OpenGL ES 2.0 standard, WebGL (abbreviation of Web-based Graphics Language) is a cross-platform and free Web standard of underlying 3D plotting application program interface (API). WebGL, included in HTML5 standard, is a kind of 3D image rendering technology on the web page. Using the exposed DOM programming interface of HTML5 Canvas, WebGL provides the method of interaction between JavaScript and GPU, makes the browser get free of the trouble of using plug-ins and can finish web rendering by directly invoking the graphics processing unit (GPU) in graphics card. Thus, not only the web pages can be displayed faster; but also the unified, standard and cross-platform OpenGL programming interface is used. WebGL aroused the interests and attention all over the world as it emerged, and many international mainstream IT companies such as Apple, Google and Microsoft expressed their supports for WebGL, thus promoting its development.

The appearance of WebGL exactly solves the above-described two questions. Firstly, WebGL can achieve the creation of Web interactive 3D graphics programs through JavaScript and need no supports from browser plug-in. Secondly, WebGL's graphics rendering using the underlying graphics hardware acceleration function can be achieved through unified, standard and cross-platform OpenGL interfaces. It means that, without the use of any browser plug-ins, Web interactive 3D graphics applications can be created only using HTML and Javascript, while the applications' visual effects and performance can be comparable to those made using Flash3D and Silverlight; moreover, the applications can operates on any platforms in a same way, which can be regarded as a revolutionary change.

Based on WebGL, this article achieved the terrain rendering, texture mapping and building model loading for the city's main flooded roads and 3D simulation on the urban waterlogging, which can serve as a new approach to investigating urban waterlogging process.

2. Construction of 3D web scene

This study constructed the 3D scene of some main flooded roads using Three.js (the Javascript 3D library for WebGL). The local Web server was first constructed and some external resources such as 3D models and texture images were loaded. Then, the scene was constructed using the source codes such as THREE. Scene,THREE. Perspective Camera, THREE. WebGL Render and THREE. Lightsource.

2.1 Terrain rendering based on Three.js

In this study, the .ASCII file in the format of GRID (Esri Company) was used for representing digital elevation model (DEM), which mainly consists of data head and data body. Data heads are listed line by line according to the format of attribute value pair, as shown in Table 1.

Data bodies are arranged line by line from left to right in accordance with row and column of DEM, and the values of DEM are separated by a spacing.

The key to terrain rendering is to determine the correspondence between the coordinate (X, Y) of each vertice on the 3D terrain surface and the column/line number of DEM grid (M, N), and then express the elevation value of each vertice on the 3D terrain surface using the elevation value of DEM grid.

Three.js provides THREE.PlaneGeometry for 3D plane plotting. THREE.PlaneGeometry inherited from THREE.Geometry and possesses the attribute of .vertices. Thus, the size of the plotted surface and the number of vertices can be controlled by setting the length and width of THREE. Geometry and the number of segments, while the topographic characteristics can be reconstructed by setting the .vertices attributes of the objects created by PlaneGeometry. Class.

The coordinates and elevation values of each vertice are required before terrain rendering. Since DEM exhibits great numbers of column and row, the rendering should be conducted by means of diluting and interpolation (see Fig. 1). The rendering process is described in detail below.

Table 1 *Definitions of each attribute value pair of .asc files*

Attribute	Value	Comment
ncols	921	Column number of DEM grid
nrows	795	Row number of DEM grid
xllcorner	12591782.194844	X-coordinate of the lower-left corner of DEM grid
yllcorner	4111893.9287671	Y-coordinate in the lower-left corner of DEM grid
cellsize	10.290469083668	Spacing lengths between DEM grids
NODATA_value	-9999	Null in DEM

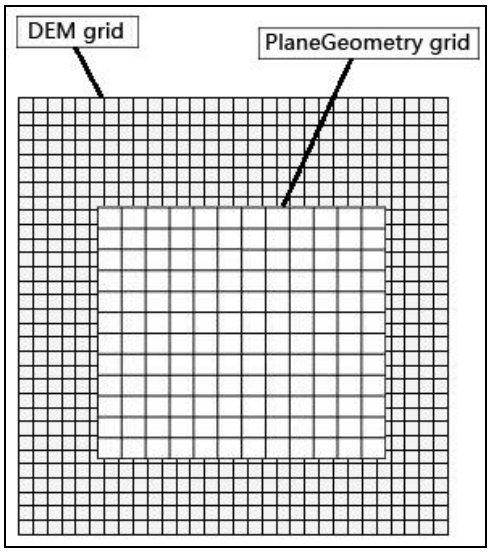

Fig. 1 *Correspondence between DEM grid and 3D surface*

Firstly, the positions of each vertice of PlaneGeometry were acquired. As stated above, given the following information, xllcorner, yllcorner, cellsize, nrows, ncols, length, width (as defined in Table 1), length and width of the known PlaneGeometry (denoted as length and width), the number of sections along horizontal and vertical directions (denoted as xSegs and ySegs), the coordinates of upper-left corner of DEM grid (X(0,0), Y(0,0)), the coordinate of any a vertice (the i-th row, the j-th column) in PlaneGeometry can be calculated by Formula (2.1).

Based on the coordinates of the lower-left corner of DEM, the coordinates of the upper-left corner can be written as (xllcornery+(nrows-1)*cellsize), and the coordinates corresponding to (X(i,j), Y(i,j)) in DEM can be calculated by Formula (2.2).

In Eq. (2.2), m and n are the numbers of rows and columns of DEM grid. By combining Eq. (2.1) and Eq. (2.2), m and n can be solved and written as Formula (2.3).

Then, m and n were rounded up and off and the integers M and N were acquired, which were the column and row number of the vertice closest to the upper-left corner of DEM.

The elevation values of each DEM vertice were taken out and stored in the two-dimensional array demArr[ ][ ], and the elevation values of the surrounding four vertices of (X(i,j), Y(i,j)) were calculated and listed in Table 2.

$$\begin{aligned} X_{(i,j)} &= X_{(0,0)} + j*(width/xSegs) \\ Y_{(i,j)} &= Y_{(0,0)} + i*(hight/xSegs) \end{aligned} \quad \text{where } 0 <= i <= ySegs, 0 <= j <= xSegs \qquad (2.1)$$

$$\begin{cases} X_{(i,j)} = xllcorner + n*cellsize \\ Y_{(i,j)} = yllcorner + (nrows - m)*cellsize \end{cases} \quad \text{where } \begin{array}{l} 0 <= m <= nrows, 0 <= n <= ncols; \\ 0 <= i <= ySegs, 0 <= j <= xSegs; \end{array} \qquad (2.2)$$

$$\begin{cases} n = (X_{(0,0)} + j*(width/xSegs) - xllcorner)/cellsize \\ m = nrows - (Y_{(0,0)} + i*(height/ySegs) - yllcorner)/cellsize \end{cases} \quad \text{where } \begin{array}{l} 0 <= i <= ySegs, \\ 0 <= j <= xSegs; \end{array} \qquad (2.3)$$

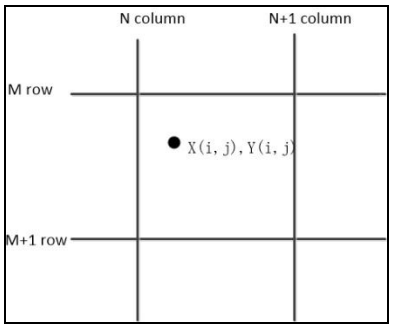

Fig. 2 *Relation between (X(i,j), Y(i,j)) and the corresponding DEM grid*

Table 2 *Elevation values of the surrounding four vertices of (X(i,j), Y(i,j)*

Position	Elevation
Upper-left	demArr[M][N]
Lower-left	demArr[M+1][N]
Upper-right	demArr[M][N+1]
Lower-right	demArr[M+1][N+1]

Then, Z (i, j) was assigned using bilinear interpolation. Bilinear interpolation refers to the linear interpolation of a two-variable interpolation function, whose core idea is to conduct linear interpolation in two directions. The interpolation of Z (i, j) can be described as Formula (2.4).

Finally, the Z values of all vertices of PlaneGeometry were calculated through circular calculation, which were then assigned to the z values of the vertices in THREE.PlaneGeometry for rendering. The number of vertices in PlaneGeometry is (xSegs+1)*(ySegs+1). The detailed rendering process is shown in Fig. 3.Fig. 4 displays the results of terrain rendering on the browser.

$$\begin{aligned} Z_{(i,j)} = {} & demArr[M][N]*(1-dx)*(1-dy) + demArr[M+1][N]*dx*(1-dy) \\ & + demArr[M][N+1]*(1-dx)*dy + demArr[M+1][N+1]*dx*dy \end{aligned}, \text{ where } \begin{cases} dx = n - N \\ dy = M - m \end{cases}; \qquad (2.4)$$

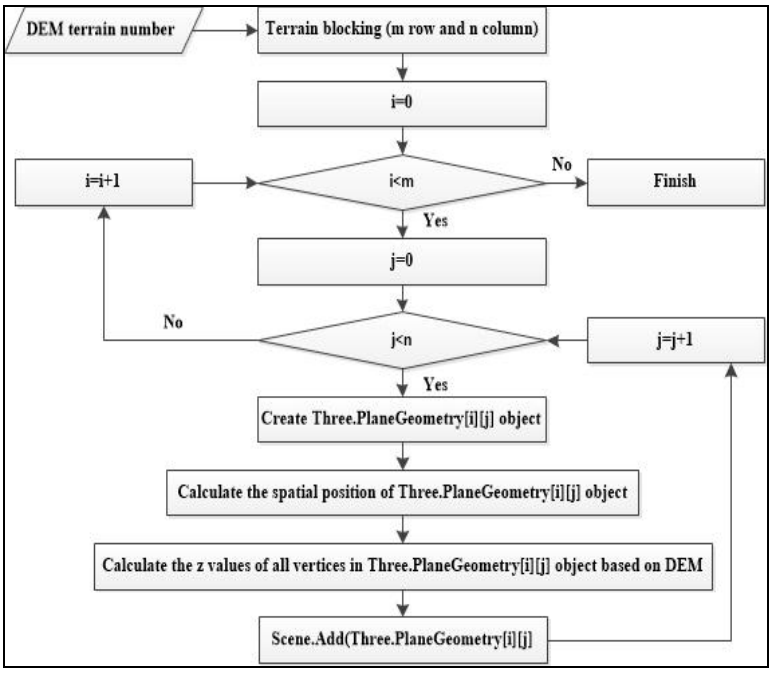

Fig. 3 *Flowchart of road terrain rendering*

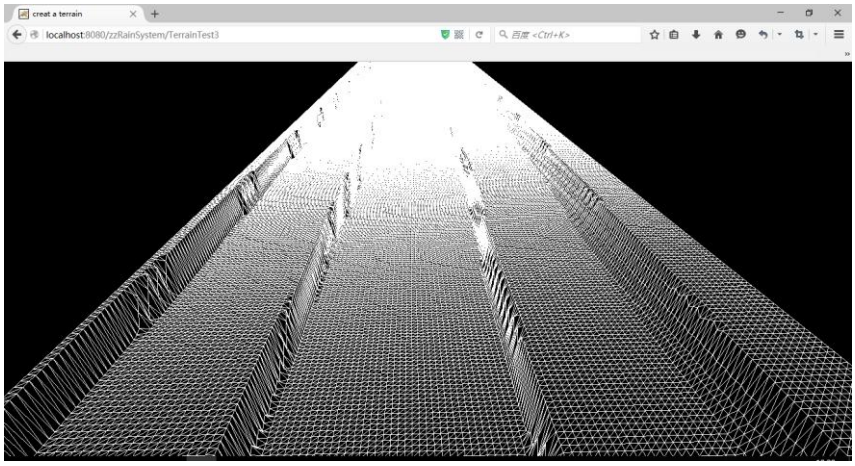

Fig. 4 *Displays the results of terrain*

2.2 Texture mapping and the loading of building model

'Material' is an attribute connected with rendering effects but independent of the object's vertice information. The object's color, texture and lighting model can be changed by setting different values of 'material'. Three.js provides the following several basic materials: MeshBasicMaterial, MeshLambertMaterial, MeshPhongMaterial and MeshNormalMaterial.

Lambert material conforms to Lambert lighting model, i.e., only the diffuse reflection effect is considered while the specular reflection effect is neglected. Thus, this model is not applicable to the objects that require specular reflection effect such as metal and mirror, but is suitable for describing the diffuse reflection effects of most objects.

Lambert lighting model can be described as:

$$Idiffuse = Kd * Id * \cos(theta) \quad (5.5)$$

where Idiffuse denotes the light intensity of diffuse reflection, Kd denotes the diffuse reflection property of the object surface, Id denotes the light intensity and theta denotes the radian of the incident/incidence angle.

The Lambert materials in Three.js can be directly used without a thorough understanding of the above formula. For creating a yellow Lambert material, we can use the following command:

```
new THREE.MeshLambertMaterial
({color: 0xffff00});
```

Since the roads and buildings in the city can be roughly described using Lambert light model, Lambert material was adopted for texture mapping. The geometry of terrain grid PlaneGeometry[i][j] one-to-one corresponds to texture image in the tile file. Firstly, texture image should be imported to texture object:

```
var texture =
THREE.ImageUtils.loadTexture('../img/0.png');
```

Then, the material's 'map' attribute should be set as 'texture':

```
var material = new
THREE.MeshLambertMaterial({map:texture});
```

Finally, material was assigned to the corresponding PlaneGeometry[i][j], and thus texture mapping was finished.

Next, the building model was loaded. Three.js provides abundant model input interfaces and can conveniently import the files in the formats of JSON, OBJ, Collada, PLY, STL and VTK. This study used the .obj model file exported from 3dmax and the texture mapping file .mtl, which can be well understood by Three. js. Moreover, Three. Js provides two different loaders. If only the geometry is loaded, OBJLoader can be used. This study used OBJMTLLoader for model loading and material assignment. The detailed loading code is described below:

```
var loader = new THREE.OBJMTLLoader();
loader.load( 'model/export/22/22.obj',
'model/export/22/22.mtl', function ( object )  {
object.position.y = 0;
object.position.x = 0;
object.position.z = 0;
scene.add( object );}, onProgress, onError );
```

After being loaded, the model was added to the scene using a callback function and the position was set. Fig. 5 shows the loading result.

Fig. 5 *3D scene renderin*

3. 3D simulation on urban waterlogging and seeper process

According to the calculated road submerged area and depth, 3D visualization was performed in 3D scenes. Then, all PlaneGeometrys were judged. If the submerged mark is FLood[i][j] of PlaneGeometry[i][j] was 'TRUE', the submerged water surface was added to the grid surface; else, the submerged water surface was not added. After all the judgment finished, the submerged area was acquired and the browsing effect of the submerged area in the browser is shown in Fig. 6.

Then, the submerged range and depth within the whole rainfall time series were simulated. By adding the dynamic water surface to the 3D scene, the waterlogging depth and area within the corresponding time series were acquired, i.e., the submerged condition of the whole rainfall process was simulated. Fig. 7 shows the dynamic simulation results of rainfall and flood process according to the calculation results

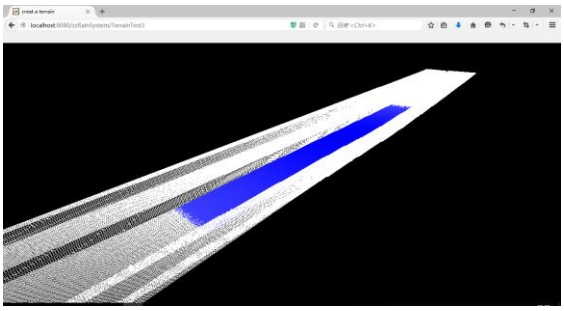

Fig.6 *Submerging calculation renderings*

Fig.7 *Dynamic simulation of rainfall and flood process*

4. Conclusions

It is of great significance to conduct 3D dynamic simulations on urban waterlogging processes and analyze urban seeper conditions under different rainstorm scenes for urban disaster prevention and reduction, disaster evaluation, post-disaster reconstruction and economic and social developments. WebGL technical standard can use the no-rendering plug-ins to create the web pages with complex 3D structures and even design 3D web games. Owing to its several advantages such as cross-language and cross-platform, WebGL can make one-development and multiple-use possible, get easy access to and interact with the fantastic 3D scenes on both PCs and mobiles only with the aid of Web browser. WebGL technique can provide technological base for 3D visualization of the mass data concerning urban waterlogging using the network client. Three.js is one of the best graphics library and can create simple 3D scenes, cartoons and even interactive games. Three.js supports WebGL, SVG, Canvas and CSS3D rendering. This article used WebGL's open source library Three.js to create 3D scenes, conducted 3D surface drawing and texture mapping on road DEM, completed 3D scene rendering and finally carried out 3D dynamic simulations on urban waterlogging and seeper process.

Acknowledgements

This work was supported by the Henan Fundamental and Frontier Technology Research, Project (Grand No. 152300410044), Key University Science Research Project of Henan Province((Grand No. 16A420005). Our gratitude is also extended to reviewers for their efforts in reviewing the manuscript and their very encouraging, insightful and constructive comments.

References

[1] YANU Xiaodong, HU Litang, TANG Zhonghua. 3D modeling and visualization of stratum based on Java/Java 3D [J]. Acta Geodaetica ET Cartographical Sinica, 2006, 35(2):166-169.

[2] LV Zhihan, FEND Shengzhong. Multi-dimensional WebGIS based 3D interactive network virtual community [J]. Journal of System Simulation, 2013, 25(9):2109-2114.

[3] CHEN Ge, QI Yongyang, CHEN Yong, MA Chunyong. Design and Implementation of Urban Simulation Oriented VRGIS [J]. Journal of System Simulation (S1004-731X), 2009, 21(2): 457-460.

[4] LIN Hui, GONG Jianhua. On Virtual Geographic Environments[J]. Acta Geodaetica ET Cartographical Sinica, 2002, 31(1):1-7.

[5] HAN Yi. The new development trend in Web3D and Web visualization-taking WebGL and O3D as example [J]. Science Mosaic, 2010(5):81-86.

[6] MA Ruina, LV Zhihan, HAN Yong, CHEN Ge. Research and Implementation of Geocoding Searching and Lambert Projection Transformation Based on WebGIS [J].Geospatial Information, 2009, 7(5):31-34.

[7] Nobuyuki Bannai, Robert B Fisher, Alexander Agathos. Multiple color texture map fusion for 3D models [J]. Pattern Recognition Letters (S0167-8655), 2007, 28(6): 748-758.
[8] A.H. Elliott, S.A. Trowsdale. A review of models for low impact urban stormwater drainage[J]. Environmental Modelling and Software, 2006, 22(3): 394-405.
[9] Christopher Zoppou. Review of urban storm water models [J]. Environmental Modelling and Stoftware, 2001, 16(3): 195-231.
[10] Alun Evans, Marco Romeo, Arash Bahrehmand, et al .3D graphics on the web: A survey [J]. Computers Graphics-UK, 2014, 41: 43-61.
[11] Dean Jackson, Jeff Gilbert. WebGL 2 Specification Editoe's Draft 2 February 2016.https://www.khronos.org/registry/webgl/specs/latest/2.0/
[12] Kyung-sook Choi, James E. Ball. Parameter estimation for urban runoff modelling [J]. Urban Water, 2002(4):31-41.
[13] Michael Bruen, Jianqing Yang. Combined hydraulic and black-box models for flood forecasting in urban drainage systems [J]. Journal of Hydrologic Engineering, ASCE, 2006, 11(6):589-596.
[14] Sing Hariom, Garg R.D. Web 3D GIS Application for Flood Simulation and Querying Through Open Source Technology [J]. Journal of the Indian Society of Remote Senaing, 2016, 44(4):485-494.
[15] Hunter Jane, Brooking Charles, Reading Lucy, Vink Sue. A Web-based system enabling the integration, analysis, and 3D sub-surface visualization of groundwater monitoring data and geological models [J]. International Journal of Digital Earth, 2016, 9(2):197-214.