Novel Robust Multilevel 3D Visualization Technique for Web Based GIS

Mohamed Sherif[1] and Hatem Abdul-Kader[2]
[1]Faculty of Computers and Informatics, Suez Canal University, Egypt
[2]Faculty of Computers and Information, Menofiya University, Egypt

Abstract: *Number of recent technologies take Geographic Information Systems to new levels of power and usability. One of the most promising technologies that empower Geographic Information Systems is the 3D GIS modeling. In this paper, a novel robust multilevel data structure model called EBOT model (block octree tetrahedron mode Geographic Information Systems l) is presented on the basis of BOT visualization model. This model combines octree and Tetrahedral Network structures. In this paper a performance simulation for implementing EBOT (Enhanced block octree tetrahedron model) algorithm into the browser using X3D Visualization is presented. A Simulation results is introduced to demonstrate the robustness of the proposed algorithm.*

Index Terms: *Geographic information systems, Geometric modeling, octrees, visualization.*

Received December 13, 2008; accepted August 3, 2009

1. Introduction

The challenge of 3D Web Geographic Information Systems (GIS) lies in creating software systems that are platform independent and run on any computer capable of connecting to the Internet and running a Web browser [1, 8, 9].

X3D (Extensible 3D); is a powerful language for describing interactive 3D worlds. The nodes contained in X3D describe the geometry of the 3D world, and a powerful messaging system that can generate animation and allow the world to respond to user input. Now what about 3D GIS modeling techniques? 3D GIS have been studied thoroughly on the basis of the successful application of the two-dimensional GIS [7]. How to organize and manage the 3D data efficiently and to construct 3D GIS spatial model has become the key to the successful application of 3D GIS. In present 3D GIS studies, the spatial data models fall into two categories: surface data model and solid data model. The former category is mainly used to represent the boundary of the spatial object, while the latter one is to represent the spatial object with solid information. In the solid data model, the spatial entity is abstracted into the collection of a series of adjacent but non-intersect 3D basic units. In the field of geology, the commonly used entity models include the block model, octree, TEtrahedral Network (TEN) and so on. However, each method has its own advantages and disadvantages. For example, the block model, adopting implicit locating technique, saves the memory space and runtime but makes model edge in contradiction with partition granularity at the same time; while TEN model can keep the original observation data and has the ability to precisely represent spatial objects and complicated topological relations, but its structure is too complicated; octree [5] is obviously simple and easy to use, but with the increase of the partition granularity, model data will multiply greatly. Besides, what octree represents is always approximate rather than accurate. Therefore, on the basis of analyzing octree model and TEN model structure, a hybrid data structure model (BOT) based on geological block model is used, which combines octree and TEN structures. The block model is subdivided with octree as general shape description and TEN as partial precise description. With the merits of both octree and TEN, BOT model can represent 3D objects more effectively and precisely.

In section II related work is presented where different relative simulation technologies is descried; section II.A is about X3D programming language, II.B Octree model structure 3D visualization algorithm, II.C TEN representation 3D visualization algorithm, then section II.D is about Data structure design and realization for BOT model where a description and flowchart of the BOT algorithm is presented . In section III simulation result is introduced where an Enhancement over BOT algorithm (EBOT) is proposed, described, flowcharted and implemented to gain better performance time. In section IV Conclusion and future work is proposed based on simulation results. Lastly different references introduced.

2. Related Work

Number of technologies cooperated to generate our 3D web GIS simulation (X3D, Octree model structure, TEN representation and BOT mode), therefore a description of each technology is presented in this section prior to introducing our simulation and results.

2.1. X3D

X3D is a browser based high level object-oriented language for the description of scenes and the behavior of objects. The language has passed through several stages, i.e., VRML 1.0, VRML 2.0 VRML97, before its endorsement as a Web standard. The syntax of X3D is based on objects (nodes) with parameters (fields) [7].

A number of nodes are responsible for the design of the scene: description of geometry (regular and irregular shapes, grids, and text), illumination of the model (directional, spot, point and ambient lights), materials and textures (draping and mapping of JPEG, GIF, PNG image file formats). Combinations of other nodes, i.e., sensors, routes and interpolators introduce dynamics. Sensors detect viewer actions (e.g., mouse move, click, and drag), time changes and viewer positions (visibility, proximity, collision). Routes direct captured events to interpolators to alter some fields (color, position, orientation, and scale). While appropriate for direct animations, the mechanism is insufficient for descriptions of complex actions, e.g. the control of sequential clicks with the mouse on an object. In case of complicated movements and manipulations, the script node referring to Java applets and Java Scripts, may be employed. The proto node supplies the user with a tool to design his/her own sensors and interpolators.

Common Gateway Interface (CGI) scripts embedded in the body of the X3D document allow establishment of connections to any application on the server. All the X3D nodes can be aggregated in various complex hierarchical composites and altered together. The scene designed according to X3D is stored in an ASCII file. Specific visualization software, i.e., Virtual Reality (VR) browser is necessary to display data on the screen. The role of X3D document and VR browsers is different. The VRML document supplies the parameters for scene design and the dynamics of objects while the VR browser takes care of scene rendering and the interface to navigate through and interact with the model. Initially, the basic function of the VR browser, besides visualization, was only real time navigation through the model, i.e., provision of virtual reality techniques: examine fly-over, walk-trough, pan, zoom, and detection of user interactions with objects.

X3D can integrate to the 3D Geospatial component, providing support for geographic and geospatial applications. This support includes the ability to embed geospatial coordinates in certain X3D nodes, to support high-precision geospatial modeling, and to handle large multi-resolution terrain databases.

2.2. Octree Model Structure

Octree representation is a hierarchical form of tessellation in which a volume occupied by a 3D model is subdivided and approximated with cubes of various sizes [4]. These cubes are known as octants and each of them is identified by a location code that gives its exact location in the octree as shown in Figure 1(a, b). The relationship among octants can also be viewed as a hierarchical tree structure as shown in Figure 1(c) where each branch is identified by the relative position of the octants in its parent node according to the six orientations R(right), L (left), U (up), D (down), F (front), and B (back).

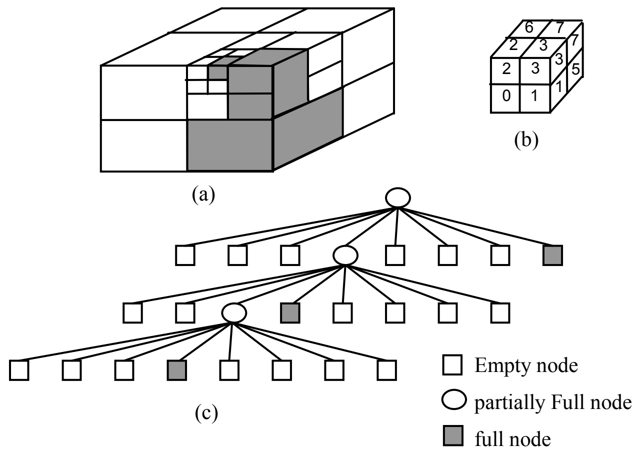

Figure 1. Octree model.

Given a 3D solid model, the first step to create its octree decomposition is to find the object's minimal bounding cube. This cube is then subdivided into eight octants by halving it along each axis direction, and the resultant octants are known as level-one octants. The octants are classified as full, empty or partially full, depending on their relative location in the 3D model.

The main advantage of octree lies in that it can be adopted to realize complex set operation that is just what the other methods can hardly deal with or need more computation resources to handle. With octree structure, it is easy to redivide the above-mentioned block model, and then to realize efficient data rearrangement. The blocks can be classified in terms of attribute value of each block: The block whose attribute value is more than the upper limit corresponds to the full node in octree structure; Otherwise, the block whose attribute value is less than the lower limit corresponds to the empty node; And the block whose attribute value is between the two limits parallels to the partially full node which needs subdivision. In the process of data rearrangement, the tetrahedral network constructed with original observation data can be used to realize partial precise description for the area where the structures (or attribute value) are complicated and changeful [3].

2.3. Tetrahedral Network Representation

TEN represents the 3D objects with tightly arrayed but not overlapped irregular tetrahedrons, essentially being the extension of two-dimensional Triangle Irregular Network (TIN) in three dimensional space. In concept, the two dimensional Voronoi network is developed to the three dimensional Voronoi polyhedron, then the TIN structure to the tetrahedral network in three dimension.

TEN composes four basic factors of dot, line, facet and body, and it is able to precisely represent 3D objects In practical application, the key problem is automatic creating algorithm of the TEN structure. In this paper, a tetrahedralization Algorithm is proposed based on the theory of Delaunay triangulation. This theory is featured by that the decomposition result is in consistency with empty sphere principle, that is to say, each tetrahedron's circumcircle includes no point in order to make the tetrahedron as likely to be regular tetrahedron as possible. Thus the existence of long and narrow tetrahedron can be avoided at the utmost. The principle of TEN generation algorithm is to create the first tetrahedron in the data field firstly, and then to create new tetrahedrons with one of the first tetrahedron's facets as one of their own facets until all the discrete points are connected into the network in this way. The detailed algorithm is described as follows:

1) Select two nearest points and draw a line segment as one side of a triangle;
2) Select a third point to form the first triangle;
3) select the fourth point to form the first tetrahedron;
4) Let I=1, J=1 (I is the number of the tetrahedrons formed already and J is for the number of the tetrahedrons being formed);
5) Extend the Jth tetrahedron to form 0 to 4 tetrahedrons;
6) I=I+K (K=0, 1, 2, 4), J=J+1;
7) If I is bigger than J, go to step 5;
8) End.

In step 2 of the algorithm above, the rules for choosing a third point are based on two features of Delaunay. First, the center of the circle made by the original two points and the chosen point is nearest to the line formed by the two original points. Second, the angle formed by the two lines from the chosen point to each of the two original points is the biggest. In step 3, when choosing a fourth point, one must meet a requirement that the center of the sphere formed by the chosen fourth point and the three points above should be nearest to the plane decided by the triangle[6].

2.4. Data Structure Design for BOT Model

In BOT model, the octree is taken as general shape description and TEN as partial precise description where the structures (or attribute value) are complicated and changeful. First of all, with the principle as what is stated above, the geological block model can be created with the original observation data (drill whole data). With the consideration of the purpose for modeling, the block size is generally defined as small granularity and the attribute value are calculated through distance weighted mean and preponderance principle, and then the model space is subdivided with octree structure. The data structure of block model and octree can be well matched during data rearrangement because the number of sub-blocks in every side is 2 to the power of an integer as well as the number of nodes created by subdivision.

Secondly, every sub divisional octant is classified into three types according to the weighted average of the attribute value of the sub-blocks in the octant. The octants whose weighted average are bigger than the upper limit correspond to full node; the octants whose weighted average are smaller than lower limit are empty node, and between the two limits correspond to partially full node which need subdivision. At the same time, the attribute value variances of the octants are calculated to decide when to construct TEN. If the variance is bigger than the set value, it means that the structure (or attribute value) of this octant is complicated and changeful and it is necessary to make partial precise description with TEN model. Finally, the nodes of BOT model are encoded and assigned with attribute value. To save memory space, a linear encoding technology is proposed based on the octree encoding principles, in which only the address code, attribute value and pointer code of the full nodes are recorded. In the BOT model, the partial TEN is equal to a full node which is located by a pointer.

For general full nodes, the attribute value can directly reproduce the attribute value of the corresponding sub-blocks in the block model, while for the full nodes formed by TEN structures; the attribute value can be obtained by distance weighted mean and preponderance principle. The linear BOT encoding technology, owing to leaving out large numbers of empty nodes and pointers, saves memory space effectively and reduces the complexity of this algorithm [3]. The algorithm diagram of BOT model is shown in Figure 2.

3. The Proposed Enhanced BOT Algorithm

Before EBOT visualization algorithm starts both the sub block size and the resolution-factor (r) must be predetermined by the user. Witch affects the resolution of the generated visualization by the algorithm; the sub block size and resolution-factor (r) affects the redivision-factor (μ) where: $\mu = (1/r) * (\text{sub-block size})$ BOT model needs further improvement for better time performance[3]. The Octree block modeling has a very good modeling time compared with TEN [6] modeling. On the contrary, Octree modeling has a bad visualization in its sparse generality and a better visualization in its dense generality; but we know that the more dense generality the more data to be saved, transferred and visualized.

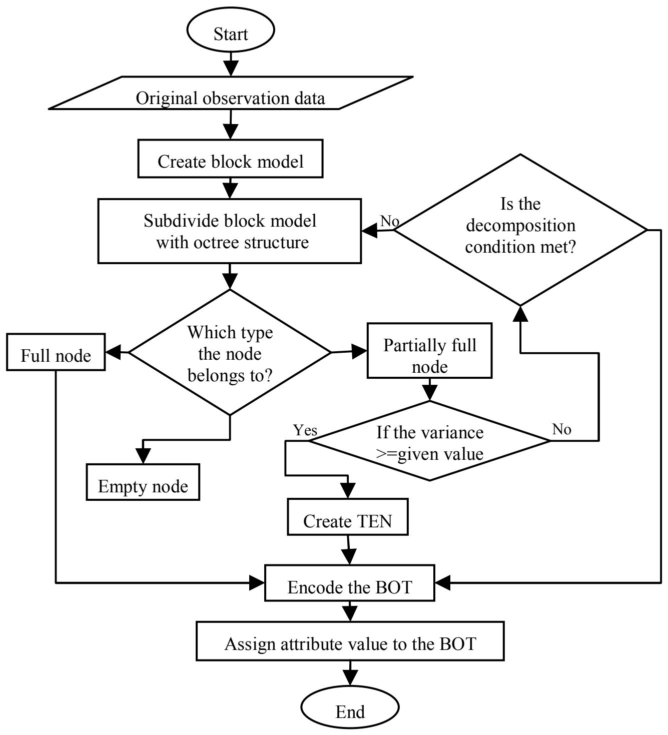
Start
Original observation data
Create block model
Subdivide block model with octree structure
No
Is the decomposition condition met?
Which type the node belongs to?
Full node
Partially full node
Empty node
Yes
If the variance >=given value
No
Create TEN
Encode the BOT
Assign attribute value to the BOT
End

Figure 2. Algorithm diagram for BOT model.

TEN modeling in the other hand have a bad modeling time compared with Octree Block modeling; but TEN has a very good visualization result compared with Octree Block modeling even in its dense generality. Now how can we solve such a problem? If we use Octree with very sparse granularly, we face the problem of low visualization quality but high performance time. Octree with very dense granularly result in low performance time, but high visualization quality. TEN modeling has a quality high performance time, Also very high visualization quality. Our enhancement of BOT performance basically rely on reducing the use of TEN modeling as possible, and using Octree as a general shape descriptor with a low generality.

The EBOT algorithm have a main check to determine the type of EBOT node (empty, full or partially full) empty node generated when no points are in current block. The other two types were depending on the computed coefficient of variance of that block. Then comparing it with threshold (full_threshold), if it is grater than full_threshold we have a full node. Else we have a partially full node which mean it ether have a dense or sparse details, we use TEN_threshold to differentiate between more block subdivision and model current block as TIN. If variance is grater than TEN_threshold then we model the current block as a TEN (as TEN preserve dense details) else it require more subdivision. Figure 3 summarize the EBOT algorithm in flow chart form.

3.1. Terrain Models Visualization Using EBOT

When to model terrain GIS. We observe that terrain GIS has a main special characteristic: "Terrain shape determined by the shape of its surface (the terrain top)". From this observation, we suppose our enhancement of BOT algorithm by increasing the TEN_threshold to all blocks lower than a TEN block (a block modeled as a TEN) as shown in Figure 1 (b); TEN modeling blocks 2,3,6,7 increase TEN_threshold for blocks 0,1,4,5 respectively. In other words; Reducing the probability of the lower blocks to be modeled as TIN as possible which reduce the whole performance time of BOT algorithm and grantee a good visualization quality. A full flowchart of EBOT algorithm is presented in Figure 3

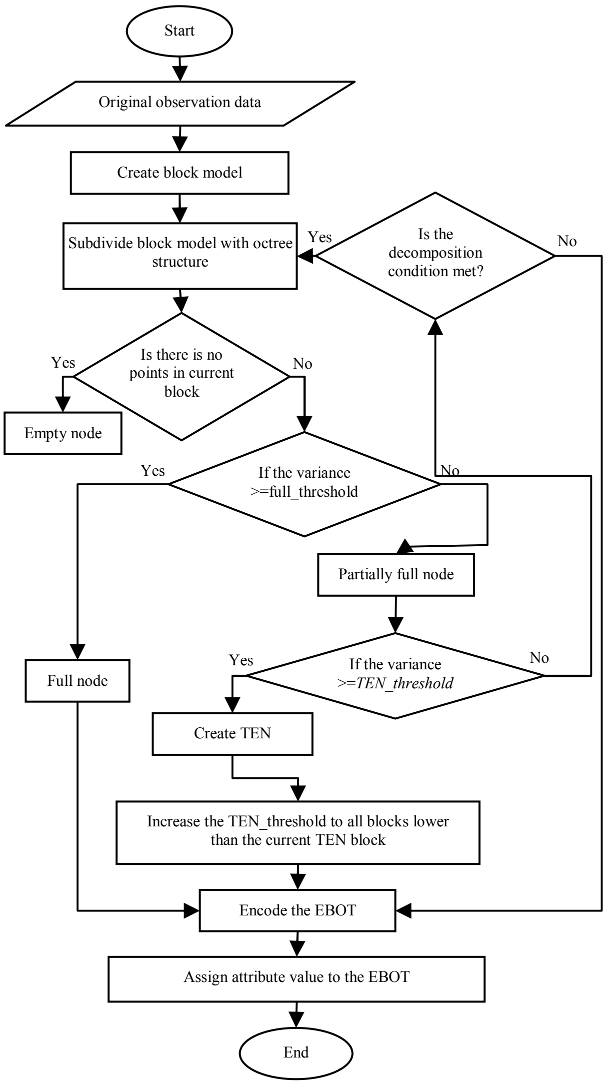
Start
Original observation data
Create block model
Subdivide block model with octree structure
Yes
Is the decomposition condition met?
No
Is there is no points in current block
Yes
No
Empty node
Yes
If the variance >=full_threshold
No
Partially full node
Full node
Yes
If the variance >=TEN_threshold
No
Create TEN
Increase the TEN_threshold to all blocks lower than the current TEN block
Encode the EBOT
Assign attribute value to the EBOT
End

Figure 3. Algorithm diagram for EBOT model.

3.1. EBOT Visualization of a Mountain (Low Resolution)

In the first mountain simulation; a cuboid volume (160m x 160m x 160m) is used as model space; simple data set of 256 points, the geological block model is set up with a block size of 10m x 10m x 10m, resolution factor (r = 16). With the generation algorithm for EBOT model, the block model is then re-divided until all the blocks are smaller than μ where: μ = (1/16)*(10*10*10) =62.5 m3, material attribute and normal vector are assigned to the EBOT model for visualization, as well as light position and lighting condition to the modeling scene (as described at Table 1), the EBOT visualization model of the low-resolution mountain is shown at Figure 4.

Table 1. BOT-EBOT simulations parameters.'

Operating System	windows XP
Web Browser	Mozilla Firefox
X3D Player	Octaga Player
Processor	Pentium IV
Ram	1 mega
Video Graphic Adaptor Card	128 RAM
Lighting Condition	Ambient lighting
Material Attribute	Solid material
Color Attribute	Gray scale coloring

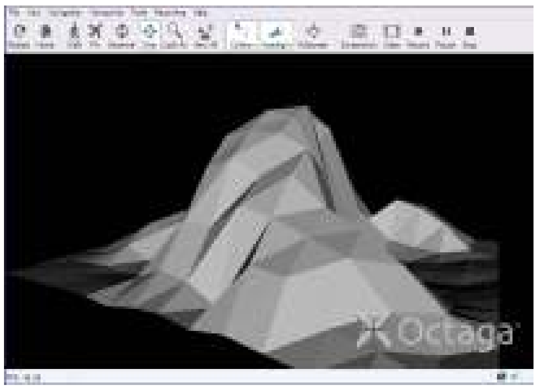

Figure 4. EBOT visualization of a mountain (low resolution).

3.2. EBOT Visualization of a Mountain (Medium Resolution)

In the second mountain simulation; a cuboid volume (320m x 320m x 320m) is used as model space; data set of 1024 points, the geological block model is set up with a block size of 10m x 10m x 10m resolution factor(r = 32). With the generation algorithm for EBOT model, the block model is then re-divided until all the blocks are smaller than μ where: μ = (1/32)*(10*10*10) =31.25 m3

Material attribute and normal vector are assigned to the EBOT model for visualization, as well as light position and lighting condition to the modeling scene as described at Table 1, the EBOT visualization model of the medium resolution mountain is shown at Figure 5 .

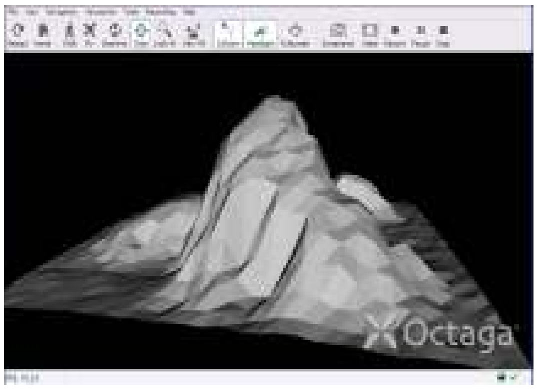

Figure 5. EBOT visualization of a mountain (medium resolution).

3.3. EBOT Visualization of Mountain (High Resolution)

In the third mountain simulation; a cuboid volume (1280m x 1280m x 1280m) is used as model space; using large data set of 16,384 points, the geological block model is set up with a block size of 10m x 10m x 10m resolution factor(r = 128). With the generation algorithm for EBOT model, the block model is then re-divided until all the blocks are smaller than μ where:
μ = (1/128)*(10*10*10) =7.8 m3

Material attribute and normal vector are assigned to the EBOT model for visualization, as well as light position and lighting condition to the modeling scene as described at Table 1, the EBOT visualization model of the medium resolution mountain is shown at Figure 3.

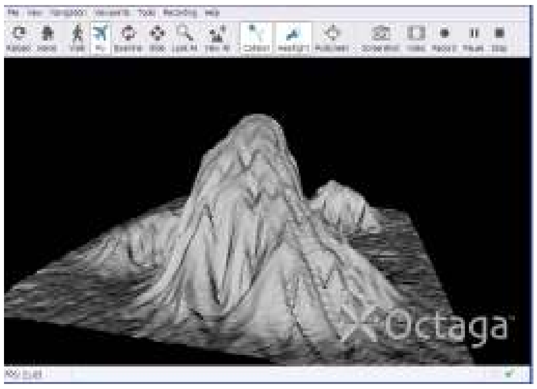

Figure 6. EBOT visualization of mountain (high resolution).

4. EBOT Performance Evaluation

Comparing 3D Web-based GISs is difficult because their functionality varies significantly. For example, native Web-based 3D viewers are faster because they run directly from the client, whereas Java applets are slightly slower because they run from a Java virtual machine The JavaVM runs on the client, resulting in a three-tier model instead of the two-tier native model. Since EBOT simulations are ran directly by native X3D players through web browsers which achieve faster visualization result [7].

In order to determine whether the proposed algorithm offers acceptable system performance, the evaluation is based on the widely accepted 10-second limit for keeping users' attention on a task [2]. The evaluation aimed to show that EBOT performs tasks within this 10-second limit. Because data load directly affects system performance, we also evaluated EBOT with different sized data sets. The data sets included a range of both terrain and urban geospatial data, varying in size and extent, to simulate complete geospatial visualization. As an additional performance indicator, the evaluation also included monitoring the frame rate as the user navigated through the visualization.

The client machines were Windows PCs with 3.0 GHz Intel processors. The simulations are visualized using Octaga Player plug-in imbedded in Mozilla Firefox web-browser. We separated the tasks into data upload and visualization operations. The upload timer started at the moment a client sent a data set request to the server. The visualization timer started when the user told the system to start creating the 3D geometry (EBOT). Figure 7 shows the results from the upload and visualization of geospatial data using EBOT. The data file sizes ranged from 0.4 Mbytes for 1,000 points to 15.2 Mbytes for 14,000 points.

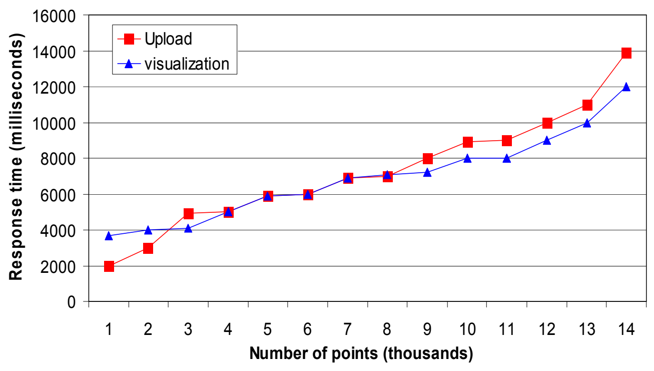

Figure 7. Upload and Visualization time chart.

Table 2. BOT-EBOT Comparison between simulations results.

		Simulation 3 Ection (3.3)		
		Low Res.	Mediu m Res.	High Res.
BOT	No. of Points	256	1024	16,384
	Cuboid Volume	160m x 160m x 160m	320m x 320m x 320m	1280m x 1280m x 1280m
	Re-division Factor (µ)	62.5 m3	31.25 m3	7.8 m3
	No. of Tetrahedrons	316	1264	5056
	No. of Octrees	32	129	519
EBOT	No. of Points	256	1024	16,384
	Cuboid Volume	160m x 160m x 160m	320m x 320m x 320m	1280m x 1280m x 1280m
	Re-Division Factor (µ)	62.5 m3	31.25 m3	7.8 m3
	No. of Tetrahedrons	216	864	3456
	No. of Octrees	37	149	598

5. Conclusions

Simulation results imply that the proposed EBOT algorithm give and enhanced performance that the classical BOT algorithm. Figure 7 illustrates that for data files up to 12,500 points (10 Mbytes in size), the EBOT algorithm visualization perform within the accepted 10-second (10,000 millisecond) limit. The performance of EBOT file uploading is also slightly faster than visualization, particularly as the number of points increases (the uploading stage exceeds the limit specified only when the number of points is greater than 13,000 points, or 11 Mbytes). Expectedly in complex polygons that can took longer to upload or visualize because of its higher numbers of vertices.
The frame rate at the previous simulations is set during data rendering ranged from 19 to 34 frames per second (fps). The lower frame rate is recorded only when displaying whole scene at once. The higher value is recorded when flying through the scene. This range in frame rate is acceptable because, as Selman notes in Java 3D Programming.

Tetrahedrons modeling are most costly part of BOT/EBOT processing, so the proposed EBOT model is trying to minimize number of tetrahedrons as possible to gain the less performance profit than the original BOT. ***Error! Reference source not found.*** Summarizes different simulations parameters' along with both BOT and EBOT algorithms. This table shows that the more points to be visualized the less the EBOT processing time over the original BOT, as the number of tetrahedrons of EBOT to model is less than the alternative tetrahedrons number of BOT.

References

[1] Del V., Paolino L., and Pittarello F., "A Usability-Driven Approach to the Development of A 3D Web-GIS Environment," *Computer Journal of Visual Languages and Computing*, vol. 18, no. 3, pp. 280-314, 2007.

[2] Geoffrey R. and Colin A., "Introspective Approach to Marking Graphical User Interfaces," *in Proceedings of The 11th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*, pp. 43-47, 2006.

[3] Huixin W. and Huifeng X., "A New Hybrid Data Structure for 3D GIS," *in Proceedings of the First International Conference on Innovative Computing, Information and Control*, pp. 256-259, 2006.

[4] Jiange T., "3D GIS Integrated Model Simulation Algorithm Based on Block Model," *Pacific-Asia Workshop*, 2008.

[5] Medellin H. and Corney J., "Algorithms for the Physical Rendering and Assembly of Octree Models," *Journal of Computer-Aided Design*, vol. 38, no. 7, pp. 69-85, 2006.

[6] Qingquan L. and Deren L., "Algorithms for Tetrahedral Network Generation," *Computer Journal of Geo-Spatial Information Science*, vol. 3, no. 1, pp. 11-16, 2000.

[7] Rong L., Yan L., Yufeng Z., and Zhuguo X., "Research on Application of VRML in Virtual City Construction," *Computer Journal of Information Technology and Applications*, vol. 1, no. 5, pp. 532-535, 2009.

[8] Wang M., "A 3D Web GIS System Based on VRML and X3D," *in Proceedings of 2nd International Conference of Genetic and Evolutionary Computing*, pp. 197-200, 2008.

[9] Zlatanova S., Rahman A., and Pilouk M., "3D GIS: Current Status and Perspectives," *in Proceedings of the Joint Conference on Geo-Spatial Theory, Processing and Applications*, Ottawa, pp. 6-10, 2002.

Mohamed Sherif is a PhD student and teaching assistant in Information System Department, Faculty of Computers and Informatics, Suez Canal University, Egypt. He holds MSc degree in geograpic information systems from Menofiya University, Egypt in 2009.

Hatem Abdul-Kader obtained his BS and MSc both in electrical engineering from the Alexandria University, Faculty of Engineering, Egypt in 1990 and 1995, respectively. He obtained his PhD degree in electrical engineering also from Alexandria University, Faculty of Engineering, Egypt in 2001.