International Journal of Modelling and Simulation

ISSN: 0228-6203 (Print) 1925-7082 (Online) Journal homepage: https://www.tandfonline.com/loi/tjms20

Marching-Cube-and-Octree-Based Level-of-Detail Modelling of 3D Objects

H. Lee & H.S. Yang

To cite this article: H. Lee & H.S. Yang (2009) Marching-Cube-and-Octree-Based Level-of-Detail Modelling of 3D Objects, International Journal of Modelling and Simulation, 29:2, 121-126, DOI: 10.1080/02286203.2009.11442517

To link to this article: https://doi.org/10.1080/02286203.2009.11442517

Published online: 15 Jul 2015.

International Journal of Modelling and Simulation, Vol. 29, No. 2, 2009

MARCHING-CUBE-AND-OCTREE-BASED LEVEL-OF-DETAIL MODELLING OF 3D OBJECTS

H. Lee* and H.S. Yang*

Abstract

The marching cube octree data structure is proposed as a scheme for representing and generating the mesh of various level-of-details (LODs). We suggest a solution to a problem of modelling partially complex objects, too. The marching cube octree is based on the data structure of the Marching Cube algorithm [1] and the octree structure. The LOD meshes are generated at run-time using proposed efficient algorithm. It triangulates only needed nodes of the marching cube octree to cover up the whole surface of mesh. Using priority numbers on nodes and flagging, the LOD mesh can be generated by only referencing without floating point operations which the other LOD models need.

Key Words

Virtual reality, mesh, surface representation, level-of-detail modelling, multi-resolution modelling

1. Introduction

In 3D graphic systems, the system performance is changed continuously because the number of objects or the complexity of the background scene is changed. If we can control the level-of-details (LoDs) of the object, we can construct an effective computer graphics system. LOD modelling consists of the representation of the mesh and the algorithm to generate a certain LOD. This paper proposes a new data structure called marching cube octree, which is based on the data structure of a Marching Cube algorithm used to generate mesh from range data. We used this data structure to make the new LOD model. Using the sampling paradigm, our algorithm turned out to be faster than previous methods.

* Dept. of EECS, KAIST, 373-1 Guseong-dong, Yuseong-gu, Daejeon, 305-701, Republic of Korea; e-mail: hasups @kaist.ac.kr, hsyang@cs.kaist.ac.kr

Recommended by Dr. Nazanin Mansouri
(paper no. 205-4527)

1.1 Related Work

There is well organized book "Level of detail for 3D graphics" [2]. According to the survey paper [3], there are three main categories of researches on LOD modelling by geometric criteria: adaptive subdivision, geometry removal and sampling.

An adaptive subdivision and analysis method that applies wavelet-based multi-resolution analysis to an arbitrary topology surface was proposed [4, 5]. This method can perform smooth parameterization at any LOD and can be applied to adaptive simplification, compression, progressive transmission and editing [6–8]. But this method renders the making of the base mesh expensive and slow. To overcome these drawbacks, a new algorithm, MAPS, was proposed [9]. The MAPS algorithm uses hierarchical simplification, defined by vertex removal, flattening and retriangulation, to induce a parameterization of the original mesh over a base mesh.

In the geometry removal category, an algorithm called "progressive mesh" was proposed that makes the new mesh by defining the edge collapse and the vertex split operation and applying these to the detailed mesh [10]. In addition, a new format was developed for saving and transmitting the triangulated geometric model [11]. The Progressive mesh method can be applied to adaptive simplification, compression, progressive transmission and view dependency rendering [12]. The model generation is relatively slow, however, because the simplification is based on the energy function. Triangle strips with edge collapse operator is used for real-time LOD mesh generation in [13]. The view-dependent meshing of part of the whole data is proposed in [14]. The whole data is in external memory and only part of it is loaded.

Randomly new vertices are inserted and then moved on the mesh surface to be displaced over maximal curvature areas. The old vertices are deleted and the new retiled mesh is generated from the new vertices in [15]. An intermediate octree representation is used for producing simplified boundary representation of mesh [16]. An intermediate hierarchical representation based on voxels with adaptive surface fitting was proposed in [17, 18].

2. Marching Cube Octree Representation

2.1 Overview

The Marching cube mesh-generation algorithm for medical images like MRI and CT was proposed [1]. To generate triangles, we use the sign and the ratio converted from the signed distance [19] of the cube's vertices. We can generate triangles naturally using a low resolution by choosing the proportional point instead of the midpoint for the vertex of the triangles. The isosurface generation using marching cubes and octree traversal was proposed [20]. This paper describes efficient creation of octree-based representation. But we applied modified marching cube configuration to octree structure, thus can construct hierarchies and implement LOD.

We define a marching cube in this paper as the set of signs and colors for each of the 8 vertices, and of ratios for the 12 edges. A marching cube octree is defined as a spatial octree whose nodes are marching cubes. As the dimension of the root node is known, we can determine the relative position and the size of any node in the octree. A null node is defined for all edges of the corresponding marching cube that do not intersect with the surface. The null node has no child nodes. Finally, the nodes of the marching cube octree correspond only to the region of the surface.

2.2 Creation of the Marching Cube Octree

The marching cube octree is created from the marching cubes of the Marching cube algorithm. The creation algorithm consists of two operations: parent node creation and marching cube conversion. The spatial octree is created using a bottom-up approach. We make the parent node from adjacent nodes and then convert it into the marching cube (Fig. 1).

A parent node can be made into the marching cube by referencing its child nodes. The decision on the sign of the vertex is classified into four cases (Fig. 2): if a corresponding (A) or adjacent (B) or diagonal (C) or center (D) vertex exists in the child node, the sign is the same as that vertex.

In case (B), several adjacent vertices can exist in the child nodes, but their signs will all be the same. The color value is copied in the same manner. The ratio is calculated easily by extending the vertex that has the triangle's vertex in it (Fig. 2). The intermediate data structure of our algorithm is shown in Fig. 3.

2.3 Node Priority Numbering

To detail the LOD of the model, we assign a priority number to all the nodes. The priority numbering determines which node expands first in the same level. We use the area difference as the LOD metric. The area difference is defined as the difference between the area of the triangles generated from one node and the area of the triangles generated from the child nodes. The higher the area difference is, the more detailed is the description of the local feature and the higher the priority is. To describe the 3D object

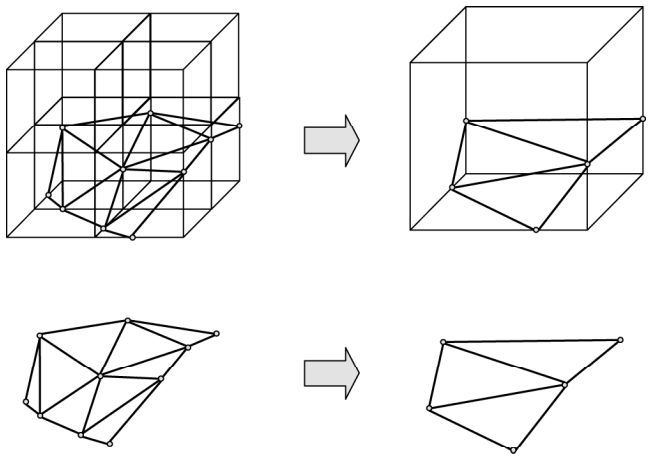

Figure 1. Creation of the parent node.

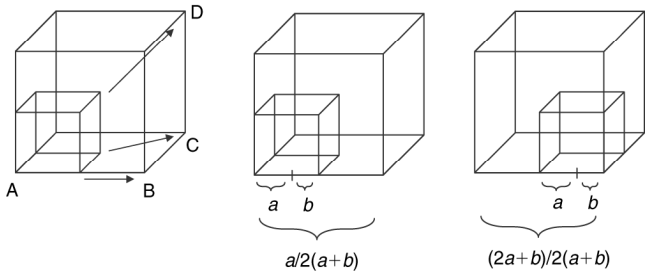

Figure 2. Conversion of the marching cube.

with fewer triangles, the higher-priority node expands first in the same level.

This algorithm starts from the root node. The expanded nodes are marked as "marked". When the level differences of some node and all surrounding nodes are less than 2, that node can be expanded. The node that has the biggest area difference in the expandable nodes set is inserted in the LOD array. Then this node is marked as "marked". This algorithm can be implemented effectively using priority queue (sorted list). To generate the LOD mesh rapidly, we save this priority number in a referencing array, the LOD array (Fig. 4).

3. Direct Generation of Level-of-Detail Mesh

In the LOD array, there is a triangulated node sequence of all nodes in the marching cube octree. When the next node is triangulated, the number of triangles in the mesh is increased by d $(0 \le d \le 4)$. The pseudo code is written as follows:

For all node n (n's priority number is i, $i-1$, ... , 2, 1)
    If all n's child nodes are 'expanded',
        exit loop;
    Otherwise,
        triangulate n's child nodes except 'expanded'
        mark n as 'expanded'
end of loop;

The final data structure of our algorithm shows as followed (Fig. 5).

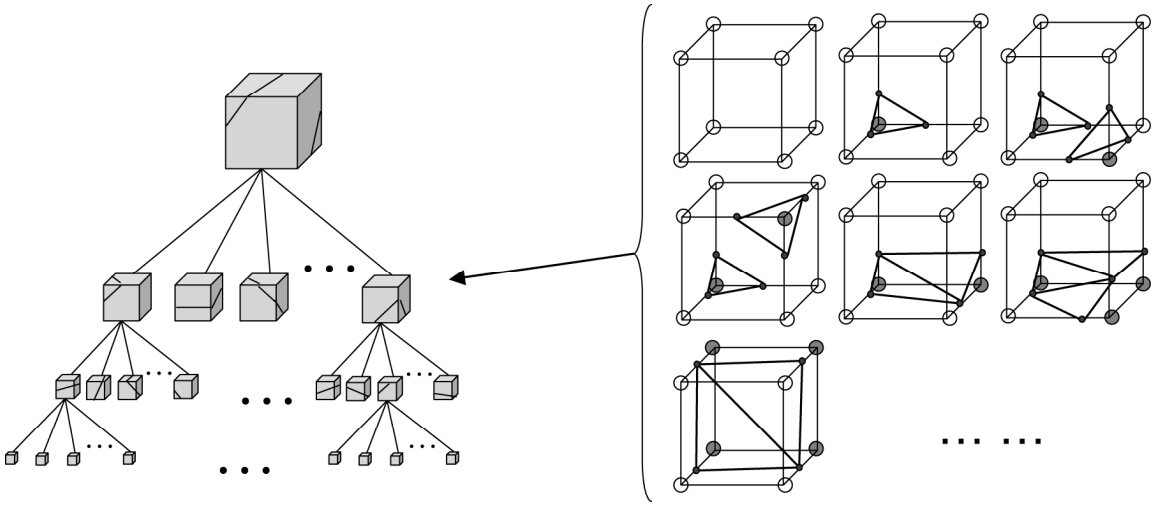

Figure 3. Marching cube octree.

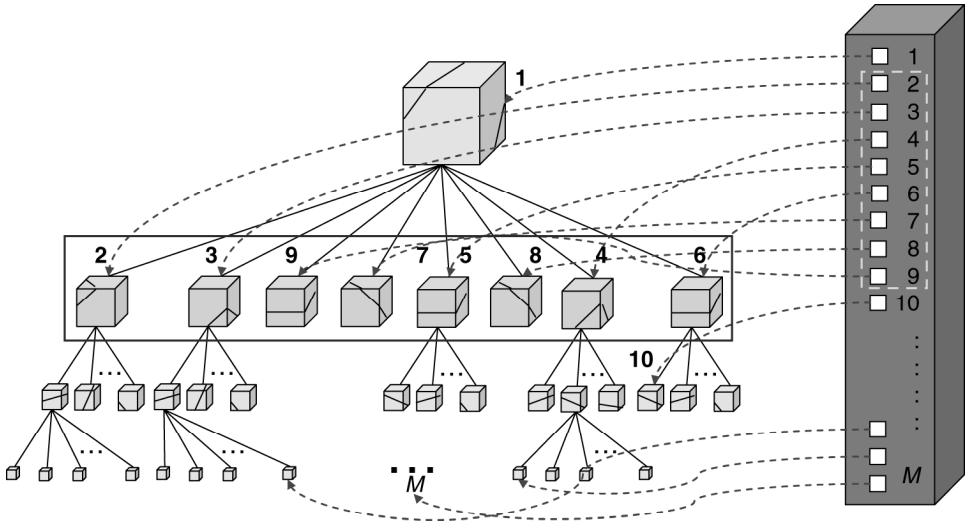

Figure 4. Marching cube octree with LOD array.

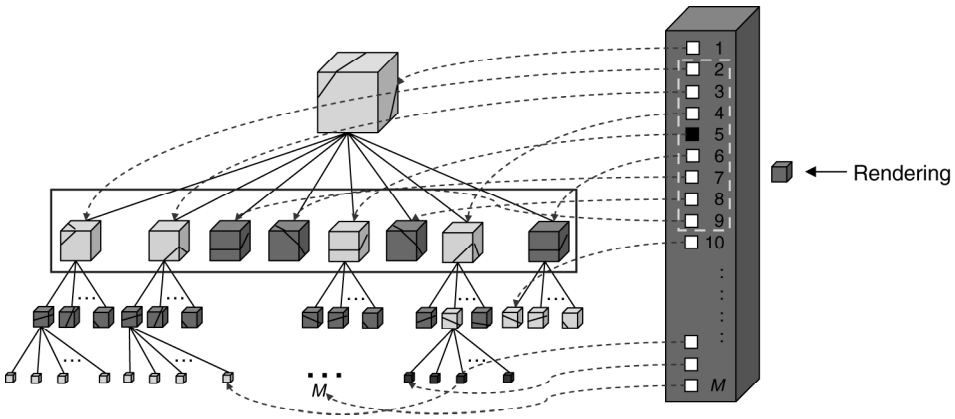

Figure 5. Direct generation of the LOD mesh.

4. Results

We use the "ball joint bone" and the "Venus statue" meshes as experimental data downloaded from Cyberware™ homepage. The results are showed by 40% of LOD. The meshes of the left side are wire frames and of the right side, rendered meshes in Fig. 6. The numbers of triangles and vertices are shown in Table 1.

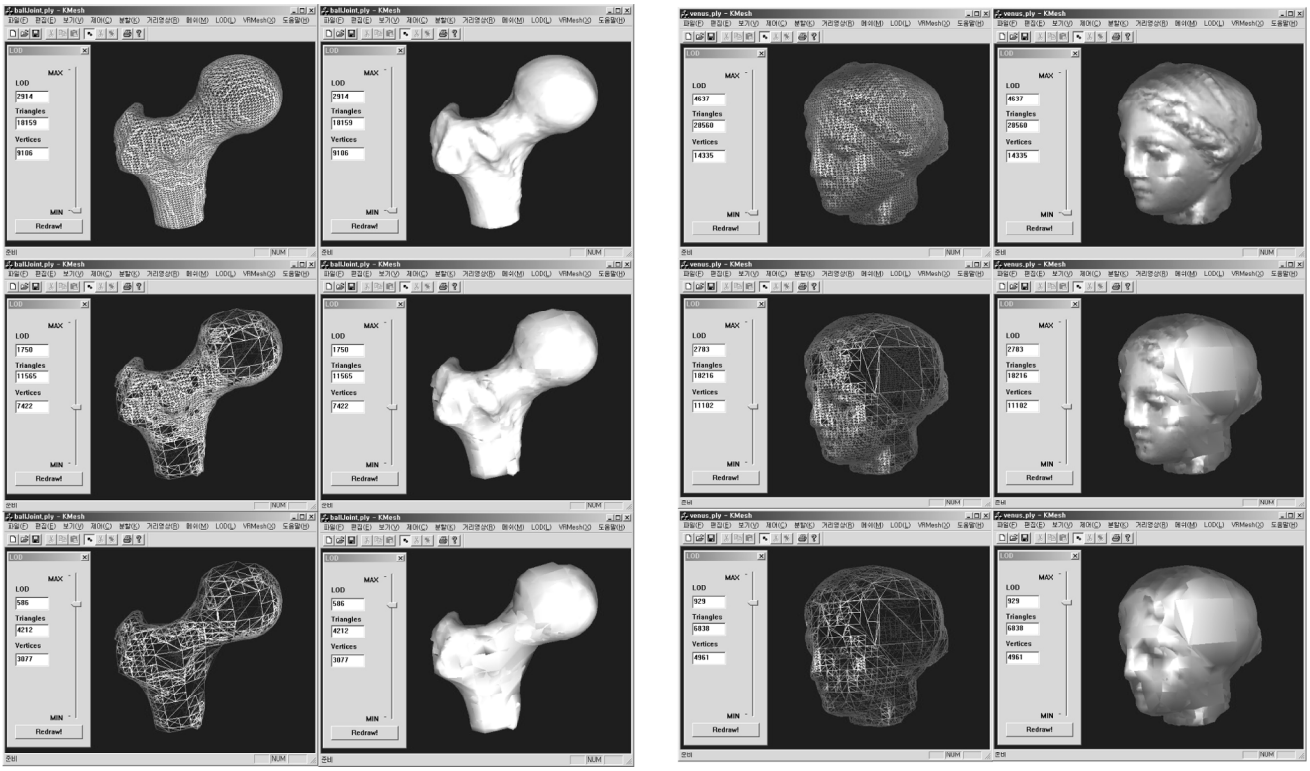

Figure 6. Examples of ball joint and Venus (data set from of Cyberware™ (http://www.cyberware.com/)).

Table 1
Numbers of Triangles, Vertices in Examples of Ball Joint and Venus

LOD (%)	Ball Joint		Venus	
	Triangles	Vertices	Triangles	Vertices
100	18,159	9,106	28,560	14,335
60	11,565	7,422	18,216	11,102
20	4,212	3,077	6,838	4,961

These results show that the marching cube octree model is feasible for a LOD representation. In flat area, the number of polygons is fewer. And in the area having curvatures, the number is more relatively. The upper right part (ball part) of the ball joint bone and the head side of the Venus statue are coarse in low LOD. But the neck-like part of the ball joint bone and the eye, nose and mouth part of the Venus statue are detailed.

LOD metric – the difference between the sum of a node's triangles area and the sum of its all child node's triangles area – data are used for the priority numbering. The graph in (Fig. 7) is LOD metric data of "ball joint bone" mesh after the priority numbering. The left graph is the case of previous level-based algorithm [21] and the right graph is of improved. The left graph shows level discontinuity. They mean our method is beyond level constraint and more continuous model. The accumulation graph is shown in Fig. 8. This shows the improved algorithm is more robust about level discontinuity. But the quality of LOD modelling is the problem of a human perception. So the results are not judged easily. The "visual fidelity" is another key issue of the LOD modelling.

5. Discussion

5.1 Modelling Time

Adaptive subdivision methods start from base mesh which is the simplest. When one triangle is divided and new vertices are moved, it calculates the moving points through whole initial mesh. So its complexity is $O(n^2)$. The overhead of crack patching is through all vertices. So an overall complexity is not changed. Geometry removal method reference all edges whenever it picks the edge to be collapsed. Geometry removal method use energy function optimization over whole vertices in this step, so its time complexity is $O(n^2)$. The edge collapse operator is a reverse of the vertex split, thus one vertex is added to one operation. The proposed algorithm references each range datum just once and calculates the configuration of octree. The crack patching is local calculation through whole node. In the node priority step, the priority queue is always sorted. The insertion to the priority queue is the same problem as "insert one entity to sorted list." The complexity of the insertion is $O(\log n)$. Then this operation is proceeded over whole nodes. Thus, our algorithm's time complexity is $O(n \log n)$. But the unit of the operation is not a vertex, a node which contains several vertices. So the time taken is slightly lesser than a vertex-based operation.

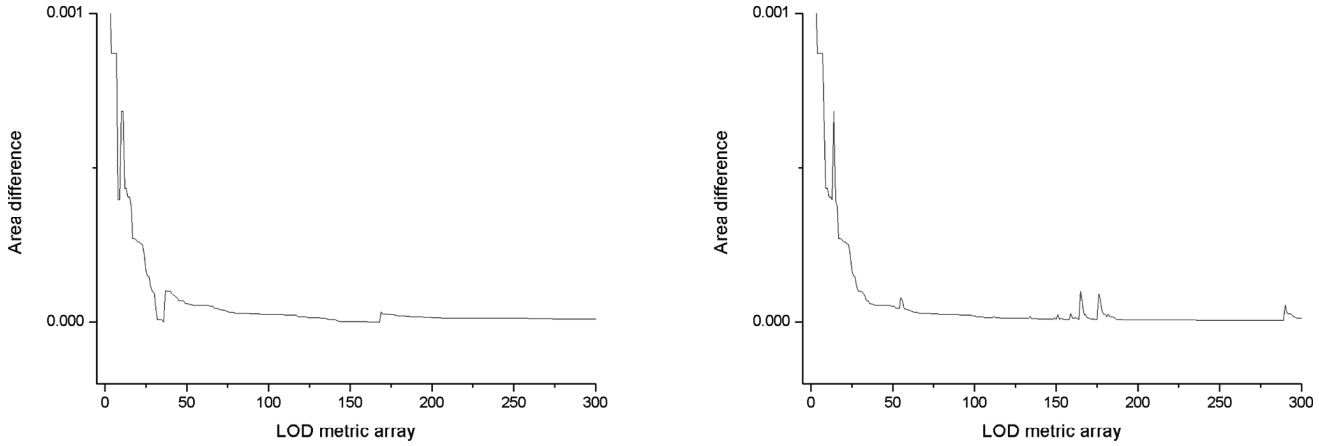

Figure 7. LOD metric graph.

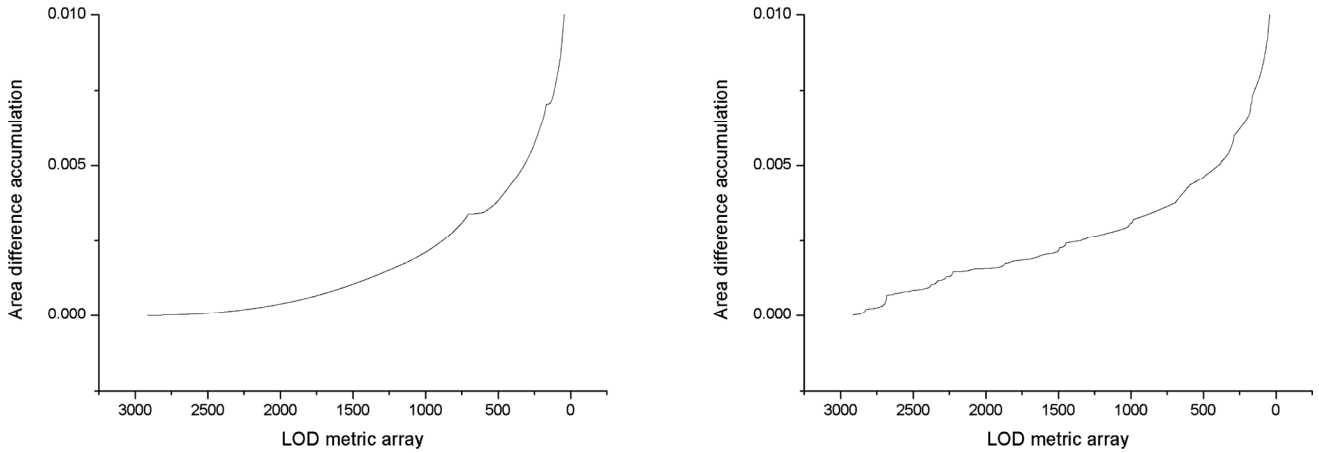

Figure 8. LOD metric graph (accumulation).

5.2 Mesh Generation Time

In online mesh generation phase, adaptive subdivision methods start from base mesh. To reach a certain LOD it divides triangles and moves the new vertices serially. Its time complexity is $O(n)$ but each dividing and moving step needs floating point operation. Geometry removal method uses edge collapse or vertex split operation serially, too. Thus, its complexity is also $O(n)$ and these operations need floating point operation, too. Our model references only needed nodes to cover the whole surface of the mesh, so its complexity is $O(n)$. But there's no floating point calculation because we need only referencing operations to generate triangles. For all three models, the traversal time is regardless compared to a triangulation. All models need to follow the links or pointers.

6. Conclusion

In this paper, the improved method of LOD modelling using the marching cube octree was proposed. We create the representation easily and efficiently by using the marching cube features. We can take advantage of the octree representation in a 3D graphics system. Our LOD model can support adaptive simplification, progressive transmission, view dependency rendering and collision detection. By using the marching cubes, our LOD mesh generation algorithm becomes efficient without floating point operations.

The LOD meshes are generated at run-time using proposed efficient algorithm. It triangulates only needed nodes of the marching cube octree to cover up the whole surface of mesh. Using the priority numbers on nodes and flagging, the LOD mesh can be generated by only referencing without floating point operations which the other LOD models need.

The contributions of this paper are as follows: supporting a continuous LOD, modifying the marching cube octree (vertex ratio on edge of marching cube) and priority numbering the nodes with an algorithm that can simplify flat area more. An efficient LOD mesh generation refers only needed nodes without floating point operations. The proposed model can perform collision detection efficiently in the complexity of $O(\log n)$.

7. Future Work

In the proposed model, the marching cube is divided into halves in the next level. If we control the size of the next level's cube and formulate it, we can construct a more detailed continuous LOD model. In that case, the fastness

and simplicity of the modelling may be lost. And the advantage of sampling simplification is no longer available.

Acknowledgments

This research was supported as a Brain Neuroinformatics Research Program sponsored by the Ministry of Commerce, Industry and Energy, and the Ubiquitous Autonomic Computing and Network Project, the Ministry of Information and Communication 21st Century Frontier R&D Program in Korea.

References

[1] W.E. Lorensen & H.E. Cline, Marching Cubes: A High Resolution 3D Surface Construction Algorithm, *Computer Graphics*, *21*(4), 1987, 163–170.

[2] D. Luebke, B. Watson, J.D. Cohen, & M. Reddy, A. Varshney, *Level of detail for 3D graphics* (Elsevier Science Inc. 2002).

[3] C. Erikson, Polygonal Simplification: An Overview. Technical Report TR-96-016, University of North Carolina – Chapel Hill, 1996.

[4] M. Lounsbery, T. Derose, & J. Warren, Multiresolution Analysis for Surfaces of Arbitrary Topological Type. *Transactions on Graphics*, *16*(1), 1997, 34–73.

[5] M. Lounsbery, Multiresolution Analysis for Surfaces of Arbitrary Topological Type. Doctoral dissertation. University of Washington, Seattle, WA, 1994.

[6] M. Eck, T. Derose, T. Duchamp, H. Hoppe, M. Lounsbery, & W. Stuetzle, Multiresolution Analysis of Arbitrary Meshes. *Computer Graphics*, *29*, 1995, 173–182.

[7] A. Certain, J. Popovic, T. Derose, T. Duchamp, D. Salesin, & W. Stuetzle, Interactive Multiresolution Surface Viewing. *Computer Graphics*, *30*, 1996, 91–98.

[8] D. Zorin, P. Schroder, & W. Sweldens, Interactive Multiresolution Mesh Editing. *Computer Graphics*, *31*, 1997, 259–268.

[9] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, & D. Dobkin, MAPS: Multiresolution Adaptive Parameterization of Surfaces. *Computer Graphics*, *32*, 1998, 95–104.

[10] H. Hoppe, Progressive Meshes. *Computer Graphics*, *30*, 1996, 99–108.

[11] J. Popovic & H. Hoppe, Progressive simplicial complexes. *Computer Graphics*, *31*, 1997, 217–224.

[12] H. Hoppe, View-Dependent Refinement of Progressive Meshes. *Computer Graphics*, *31*, 1997, 189–198.

[13] M. Shafae & R. Pajarola, DStrips: Dynamic Triangle Strips for Real-Time Mesh Simplification and Rendering, *Proc. Pacific Graphics 2003*, 2003, 271–280.

[14] C. DeCoro & R. Pajarola, XFastMesh: Fast View-Dependent Meshing from External Memory, *Proc. IEEE Visualization 2002*, 2002, 363–370.

[15] C. Andujar, D. Ayala, P. Brunet, R.J.-Arinyo, & J. Sole', Automatic generation of multiresolution boundary representations. *Computer Graphics Forum (Eurographics'96 Proc.)*, *15*(3), 1996, 87–96.

[16] T. He, L. Hong, A. Kaufman, A. Varshney, & S. Wang, Voxel-based object simplification. *IEEE Visualization '95 Proceedings*, 1995, 296–303.

[17] T. He, L. Hong, A. Varshney, & S. Wang, Controlled topology simplification. *IEEE Trans. on Visualization & Computer Graphics*, *2*(2), 1996, 171–183.

[18] G. Turk, Re-tiling polygonal surfaces. In Edwin E. Catmull, editor, *Computer Graphics*, *26*, 1992, 55–64.

[19] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, & W. Stuetzle, Surface Reconstruction from Unorganized Points, *Computer Graphics*, *26*(2), 1992, 71–78.

[20] J. Wilhelms & A.V. Gelder, Octrees for Faster Isosurface Generation, *ACM Transactions on Graphics*, *11*(3), 1992, 201–227.

[21] H. Lee, J. Lee, & H.S. Yang, Real-time LOD: Marching-cube-and-octree-based 3D Object Level-of-detail Modeling, *The 8th International Conference on Virtual Systems and MultiMedia Proceedings*, 2002, 634–643.

Biographies

Hasup Lee received his B.S., M.S. and Ph.D. degrees from the School of Electrical Engineering and Computer Science, Division of Computer Science in Korea and Advanced Institute of Science and Technology in Korea in 1995, 1997 and 2007, respectively. His current research interests include computer graphics, computer vision, medical imaging, scientific visualization and virtual reality.

Hyun S. Yang received his B.S. degree in 1976 from the Department of Electronics Engineering at Seoul National University in South Korea. He also received his M.S.E.E. and Ph.D. degrees from the School of Electrical Engineering at Purdue University in West Lafayette, IN, in 1983 and 1986, respectively. Since 1988, he has been with the Department of Computer Science at KAIST in Daejeon, South Korea where he is currently a full-time professor. His current research interests include humanoid robotics, computer vision, interactive media art and multimedia.