IEEE information theory: coding and communication (ITCC 2000), pp. 104-108, Las Vegas, NV, Mar. 2000.

OctMesh - Interactive Mesh Browsing Over the Internet

Hongwu Wang and Jin Li
Microsoft Research China
5F, Sigma Center, No. 49 Zhichun Road, Haidian District, Beijing 100080, P. R. China
Email: {i-hwwang, jinl}@microsoft.com

Abstract

This paper proposes OctMesh, an interactive mesh browsing scheme over the Internet. In OctMesh, a large-scale mesh is encoded into one bit-stream, which is organized with respect to spatial locality and resolution. When the mesh is viewed over the Internet, by examining the user's viewpoint, the compressed mesh with desired locality and resolution is streamed to the client. OctMesh enables large-scale mesh to be viewed quickly over the Internet.
Keywords: *mesh coding, view-dependent, progressive, Internet*

1. Introduction

Polygonal models, especially triangular meshes, are widely used in CAD and interactive 3D computer graphics. With the improvement of modeling tools and 3D scanning technologies, mesh models become more and more complex. The model of a real industry part usually involves hundreds and thousands of triangles; and the model of a large product, such as a new airplane, may reach the size of terabytes. Transmitting and sharing such large models over the Internet in remote client-server applications, such as collaborative working environment, e-commerce, is a daunting task.

To deal with the problem, mesh compression schemes have been proposed. Schemes such as the generalized triangle mesh proposed by Deering [1] and the topological surgery by Taubin [2] reduce the file size of a mesh representation by exploring correlation and redundancy in the mesh structure. Progressive transmission [2][3][4][5] is another effective approach to speed up mesh streaming over the Internet. A progressive mesh-coding scheme encodes the mesh into a bit-stream with decreasing importance. When the bitstream is streamed over the Internet, at first a coarse model with reduced polygon count and coarse vertex position is reconstructed, as more and more bits are received, the model is gradually refined. The full resolution mesh model may be reconstructed when the entire bit-stream has been received.

However, real industry model is usually so large that neither the existing mesh compression nor mesh progressive streaming scheme can handle them effectively. In this work, we propose the interactive browsing of mesh model. For a user dealing with the large-scale mesh, not all the details of the mesh structure are required all the time. When the mesh is viewed globally at a relative far away distance, we only need the global structure of the mesh and are not interest in local details. Alternatively, when the mesh is viewed at a closeby distance, high-resolution details are needed but only for a small region. Hence, the mesh may be browsed interactively, with only a certain mesh region viewable at a specific resolution at any time. If the compressed mesh bitstream can be organized in a structure with locality to the space and resolution, during the interactive browsing, only the mesh structure that is visible at the current view point and resolution level needs to be accessed. The compressed bitstream of the mesh region may be delivered in a priority way over the network so that we can quickly get a coarse view, and gradually refines the model when more and more bits are arrived. The accessed mesh model of the interactive browsing bears resemblance to the mesh model generated by view-dependent mesh-rendering schemes ([6][7]), which reduce the rendering complexity by rendering close-by mesh at high resolution, and far-away mesh at low resolution. However, in the view-dependent mesh-rendering schemes, it is assumed that the entire mesh structure can be accessed at local. Such is obviously not the case for the interactive browsing application.

OctMesh is an interactive mesh browsing scheme. OctMesh encodes a large-scale mesh into one single bit-stream, with a structure that localizes the bitstream element with regard to the space, resolution and quality. When the mesh is viewed over the Internet, the viewable mesh is determined and only the viewable mesh at desired resolution is streamed to the viewer. With OctMesh, large-scale mesh can be interactively browsed quickly over the Internet.

2. OctMesh Representation

We only deal with triangle-based meshes in this paper. A triangle mesh consists of two parts: the geometry, which is described by a set of vertices in 3D space, and the connectivity, which refers the set of triangles built on the vertices. The challenge is to develop a scheme which organized the bitstream with a localized structure with regard to the space and resolution.

We select an oct-tree based structure, as the oct-tree itself provides localization in space and resolution. The mesh represented by oct-tree can be accessed with difference resolution at different space locations; we call the mesh representation the OctMesh. It takes two steps to build an OctMesh. We first deal with the geometry. An oct-tree structure is built according to the coordinates of vertices of the original mesh. Then, the connectivity of the mesh is handled and the triangles of the mesh are clustered and assigned to different nodes of the oct-tree.

2.1 Oct-tree based vertex clustering

The first step of building an OctMesh is to cluster the vertices of the original mesh and build a hierarchical space-partitioning oct-tree. The clustered vertex forms an oct-vertex tree. We first bound the entire mesh structure with a bounding box, which forms the root node of the oct-tree. Then, the bounding box of the root node is subdivided exactly in half along the X, Y, and Z directions into 8 cubical cells, which are the direct child nodes of the root node. Each node may be split recursively again if there are at least two vertices falling within the node. If there is only one vertex falling within a node, the node is marked as a terminal node and is not split any more. The nodes with children nodes are marked as non-terminal node. There is exactly one vertex within each terminal node; and for any vertex, there is a terminal node that envelops the vertex. For each non-terminal node, we find a representative vertex of all the vertices falling within the node. This representative vertex serves a low-resolution representation for the oct-tree area if the mesh is viewed at a far away distance.

There are many ways to determine the position of the representative vertex of a node. One strategy is to position the representative vertex at the centroid of all the vertices within the node. Though getting better intermediate mesh, the approach spends bits on the intermediate representative vertex that may not be useful at a fine level. An alternative representation is to place the representative vertex at the center of the node. The advantage of the approach is that no additional location information is needed for the intermediate representative vertex. To obtain a more accurate vertex location, it is necessary to go down the vertex tree structure. The disadvantage is that the location of the intermediate representative vertex of the non-terminal node may not be accurate, resulting in a poor quality intermediate mesh representation.

By encoding the vertex tree, we encode all the vertices of the mesh model. Moreover, the vertex tree localizes the vertices with respect to space and resolution. By accessing a subtree lead from a child of the root node, we access $1/8^{th}$ of the original mesh. Further space localization can be achieved by going down the vertex tree. Alternatively, by truncating the leaf of the vertex tree, we reduce the details of the mesh, and thus reduce its resolution.

The OctMesh structure bears resemblance to the hierarchical dynamic simplification (HDS) scheme proposed by Luebke[7]. However, OctMesh is able not only to form a view dependent mesh with changing resolution in different part of the mesh, but also to store the mesh according to the locality of space and resolution. The mesh can thus be interactively browsed over the Internet according to the viewing angle and resolution.

2.2 Triangle representation

With all the vertices encoded, a triangle can be represented by the indexes to its three vertices in the global vertex-table. In OctMesh, because vertices are managed by the oct-tree, we use the branching information of a node to identify the vertex enveloped by the node. For the oct-tree structure, 3 bits is enough to code the branching information at each level. Thus, all the vertices of the mesh structure, including the original vertex of a terminal node and the representative vertex of a non-terminal node can be represented by a 3-bit triplet string, in which each 3-bit triplet indicates the correct branching for one level of the oct-tree. We call the 3-bit triplet string of a vertex the vertex branching code (VBC). Apparently, each 3-bit triple string uniquely identifies a vertex; and truncating the triple string yields a coarse representation of the space location of the vertex, or a representative vertex whose cluster envelops the vertex. Figure 1 shows a vertex-tree (a) and a list of VBCs of the vertices in the vertex-tree (b). The deeper the vertex tree, the longer the VBC code and the more accurate the vertex position. A triangle in the original mesh can be represented by the three VBCs that identify the three vertices of the triangle. We use a progressive refined VBC table to manage the VBCs of an OctMesh. The detail will be described in the encoding process.

For the purpose of interactive browsing, we need to localize the triangles, i.e., the connectivity, in the vertex-tree representation. As we move up the vertex-tree, vertices clusters into representative vertices, and the associated triangle collapses first to a line if more than two of its vertices merge, and then to a single point if all three vertices merge. We define the location node of a triangle to be the node that the triangle collapses to a single node. The trian-

gle are recorded, or localized, at its location node. And the triangle is a generated triangle, and its vertices the generated vertices of the location node. Many triangles may be introduced at the same node, in such a case all the newly introduced triangles form a triangle stripe that can be efficiently encoded.

We illustrate the OctMesh representation in Figure 2(a). Since it is difficult to show the vertex-tree in 3D, we show a 2D mesh instead. The oct-vertex tree of the 3D reduces to a quad-vertex tree in this case. The mesh in Figure 2(a) is consisted of 13 nodes and 12 triangles. The generated triangles of the root node R are: Δbfe, Δbed, Δedh, Δdhg, Δgkh, Δklh, Δehi and Δhij. The generated nodes of the root node R are: b, d, e, f, g, h, i, j, k and l. The root node is split into four non-terminal nodes A, B, C and D. And the generated triangles and nodes of node A are: Δacd, Δacd, node a and c. The generated triangles and nodes of node D are: Δhlj and Δljm and node m.

2.3 Multiresolution representation of OctMesh

By managing vertices and triangles in an oct-tree based structure, OctMesh offers a multiresolution mesh representation, in which we may extract a specific region of the mesh at a certain resolution. We define a multiresolution representation of the original mesh as a sub-tree of the full vertex-tree of the OctMesh, provided that the sub-tree meets the following requirements:

- The sub-tree shares the same root node with the OctMesh.
- A node has all its ancestor nodes in the sub-tree. This guarantees that the sub-tree is indeed a tree, and ensures the connectivity of the sub-tree.
- A node has all its sibling nodes (nodes sharing the same parent node) in the sub-tree. This guarantees the completeness of the lower resolution mesh.

A mesh represented by the sub-tree may have different resolutions for various space areas. The resolution is higher for the part of mesh where the sub-tree is expanded deeper, and lower where the sub-tree is collapsed in shallow levels. Two sample sub-trees of the OctMesh are shown in Figure 2(b) and (c), respectively. The sub-tree in Figure 2 (b) has a full resolution at node A, and a reduced resolution at node B, C, and D. Alternatively, the sub-tree in Figure 2 (c) has a full resolution at node B, C, and D and a reduced resolution at node A.

2.4 Interactive browsing

The OctMesh encoded model may be interactively browsed by the user. At a certain moment, a viewing point and a viewing angle are specified by the user. The browser then downloads a sub-tree of the Oct-mesh which corresponds to the viewable portion of the mesh with sufficient resolution. The root node of the OctMesh is downloaded firstly together with its generated triangles and vertices. According to the viewpoint of the user, the child nodes of the root node in the OctMesh are sorted. The node close to the viewpoint and region of interest is assigned with a high priority, and the node faraway to the viewpoint and the region of interest is assigned with a low priority. The streaming of the OctMesh can thus be performed in a prioritized fashion. High priority nodes are further expanded. At any moments, the children nodes of all the existing nodes are examined and sorted by priority. As a rule of thumb, close-by node has a higher priority than the far away node, and low hierarchy node has a higher priority than high hierarchy node. As more and more nodes are received, a sub-tree of the OctMesh is built up gradually. The mesh is refined and its quality becomes better and better. The user may change the viewpoint and viewing angle during the browsing process, in such a case, the priority of the children nodes will be changed accordingly corresponds to the new viewpoint and viewing angle.

2.5 Coding and localization of OctMesh

The compressed bitstream of the OctMesh can be classified into four categories: the global head, the generated vertices of a location node, the generated triangles of a location node, and the refinement VBC of the vertices. The information is localized in OctMesh through five information blocks:

1. H-block: Header of OctMesh stream, containing global head, e.g. the location and size of the mesh bounding box.
2. N-block: Node information block, including child mask information, coordinate offset information, etc. (Only for terminal node).
3. T-block: Sub-triangle strip block.
4. I-block: Introduction block of VBCs.
5. R-block: Refinement blocks of VBCs.

We split the bit stream of a VBC into two parts: one introduction block and a series of refinement blocks. When a new vertex is introduced, a 3-bit triple is used to indicate the branching information to the next tree level. The 3-bit triples of all the vertices introduced in a node form its introduction block, i.e., the I-block. Once introduced, the vertex should be refined in the descendants of the node that introduces the vertex. In other words, a node should refine all the vertices that introduced by its ancestors and falling within the bounding box of the node. For any previous introduced vertex, a 3-bit triple, which indicates the vertex's branching information in the next level of the vertex-tree, refines the vertex's VBC. All the refining 3-bit triples of a node form the refinement block of the node.

3. Results

Both the OctMesh encoder and browser have been implemented. A triangular mesh model is compressed through OctMesh and is then stored on a server. The compressed bitstream may then be accessed interactively by an OctMesh browser. We select a partial region of interest which is the center of the browse window. At first, a global view of the coarse resolution mesh is rendered; it is then gradually refined according to the viewpoint of the user. The user may change the view point and view angle by rotating and zooming in/out of the mesh.

We show an example of the view-dependent refinement of a cow model in figure 3. The cow with full resolution is shown in figure 3(a), with 2904 vertices and 5804 triangles. In figure 3(b), we assume that the viewer is focusing at the head of cow, which is highlighted by a square. In this view, 97% of the vertices and 95% of the triangles have been generated, but only 31% of the data amount has been downloaded because many vertices are not refined. In figure 3(c), the viewer is focusing at the back legs of the cow. 98% of the vertices and 95% of triangles have been introduced in this view; however, only 33% of the data has been downloaded.

4. Conclusion

In this paper we proposed OctMesh, an interactive mesh browsing scheme. Any arbitrary mesh can be encoded with an OctMesh and interactively browsed by the user. Unlike local view-dependent rendering schemes, view-dependent refinement of OctMesh does not need the access of the entire mesh structure.

5. Reference

[1] M. Deering, “Geometry compression”, Computer Graphics (SIGGRAPH’95), pp. 13-20, Aug. 1995.

[2] G. Taubin and J. Rossignac, “Geometric compression through topological surgery”, Tech. Rep. RC-20340, IBM Watson Research Center, Jan. 1996.

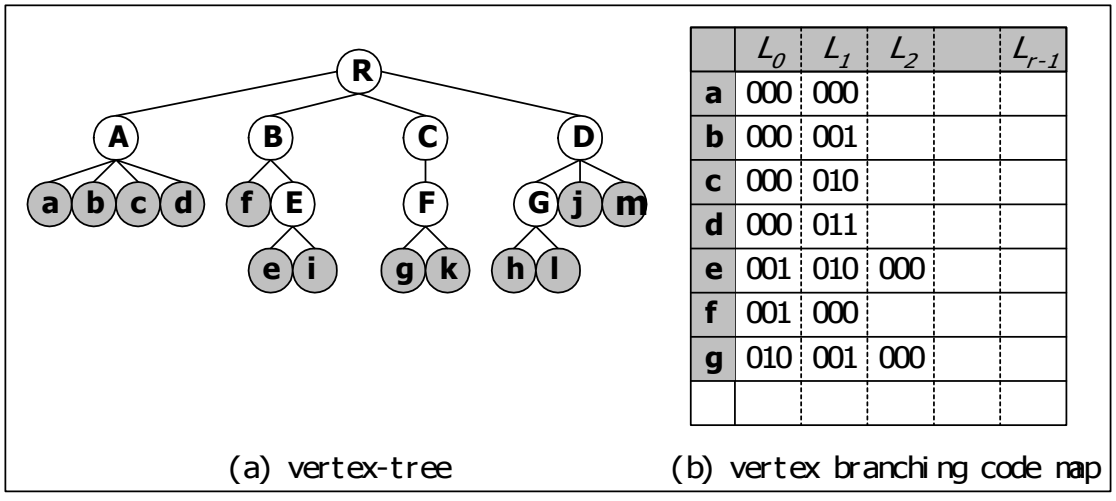
[3] J. Li and C. C. Kuo, “Progressive coding of 3D graphics models”, Proc. Of the IEEE Vol. 86 No. 6 pp. 1052-1063, June 1998.

[4] H. Hoppe, “Progressive meshes”, Computer Graphics (SIGGRAPH’96), pp. 99-108, Aug. 1996.

[5] J. Popovic and H. Hoppe, “Progressive simplicial complexes”, Computer Graphics (SIGGRAPH’97), pp. 217-225, Aug. 1997.

[6] H. Hoppe, “View-dependent refinement of progressive mesh”, Computer Graphics (SIGGRAPH’97), pp.189-198, Aug. 1997.

[7] D. Leubke and Carl Erikson, “View-dependent simplification of arbitrary polygonal environment”, Computer Graphics (SIGGRAPH’97), pp.199-208, Aug. 1997.



	L_0	L_1	L_2		L_{r-1}
a	000	000			
b	000	001			
c	000	010			
d	000	011			
e	001	010	000		
f	001	000			
g	010	001	000		

Figure 1. A vertex-tree and the VBCs of the terminal nodes in the vertex-tree

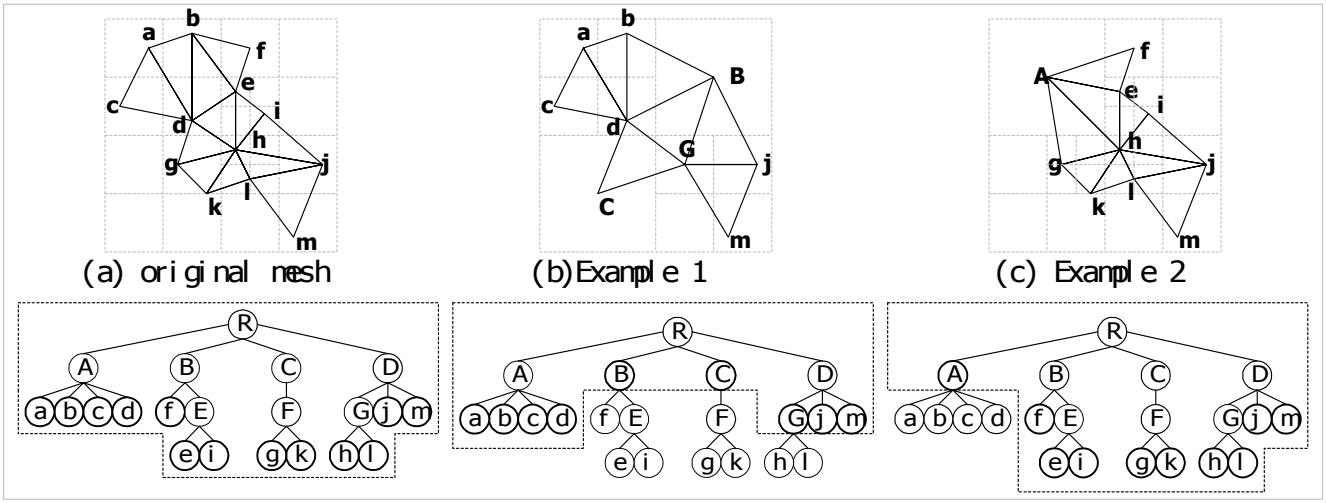
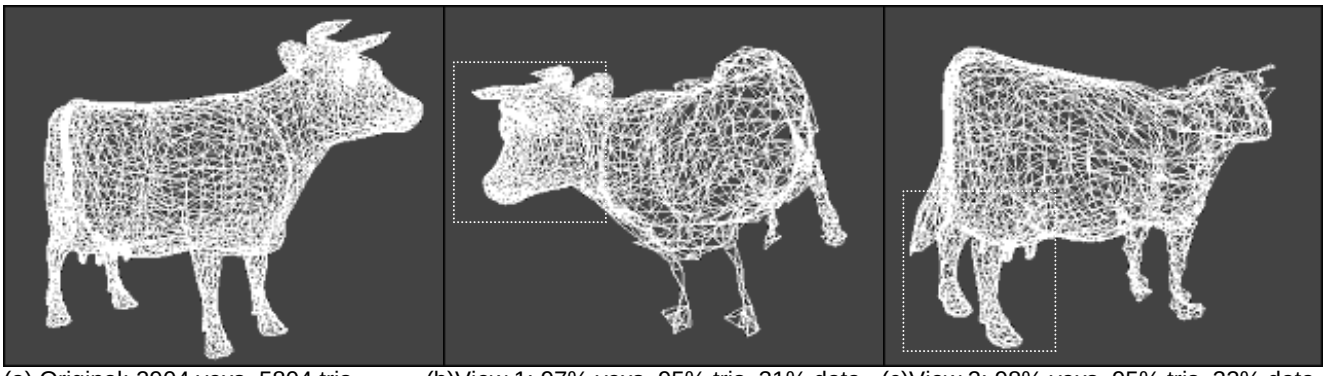


Figure 2. Example of multiresolution representation



(a) Original: 2904 vexs, 5804 tris, (b)View 1: 97% vexs, 95% tris, 31% data (c)View 2: 98% vexs, 95% tris, 33% data

Figure 3. Examples of view-dependent refinement of cow model.