This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the WCNC 2009 proceedings.

Progressive Display Method for Interactive Mobile 3D Graphics

Mouguang Lin
School of Information Science and Technology
Sun Yat-Sen University
Guangzhou, China
garnettlam@gmail.com

Xiaonan Luo
Key Laboratory of Digital Life (Sun Yat-Sen University)
Ministry of Education
Guangzhou, China
lnslxn@mail.sysu.edu.cn

***Abstract*—3D graphics on mobile devices is significant ongoing research. By considering the limitations of the mobile devices, the investigation of reducing geometry complexity and progressive display is highly necessary. In this paper a progressive display method based on a client-server system is proposed to provide a solution of displaying interactive mobile 3D graphics. A progressive reconstruction and display method based on a unique detail file structure is introduced. The results on PDA illustrate that this new progressive display method implements progressive display of interactive mobile 3D graphics efficiently, and is suitable for mobile devices which are powered by batteries. This research will have a wider prospect in mobile 3D graphics field, such as real-time interactive application on mobile devices.**

Keywords- mobile devices; progressive reconstruction; progressive display; mobile 3D graphics

I. INTRODUCTION

Since the great advancement of wireless communications and mobile computing technology in recent years, displaying 3D graphics on mobile devices has become a significant ongoing research. The characteristics of mobile devices are the following: the display panel is small and there is no need for very high detail graphics, the performance is relatively low and it is very difficult to process the high detail 3D models, the bandwidth of wireless network is narrow and it needs special transmission method, the battery is limited and it needs lightweight operation. To display interactive 3D graphics on mobile devices, all of these characteristics must be considered.

Two approaches are adopted to address mobile 3D graphics: local rendering and remote rendering. The former uses local device resources to display the 3D scene, while the latter uses remote hardware to render the scene to be displayed on the screen of the mobile device [1]. The remote rendering is an image-based rendering (IBR). In [2] an IBR for mobile 3D graphics based on depth image and warping is proposed. The authors indicate that the running time of IBR depends mainly on display resolution instead of the number of polygons. Hence IBR is well suited for mobile devices [2]. Lots of work do discuss image-based remote rendering, such as [1, 3, 4, 5].

Image-based remote rendering is suitable for mobile graphics, but usually producing of artifacts is inevitable and special methods to reduce artifacts are needed, therefore some interactive applications might require local rendering for more intuitional display and better interaction. A multi-resolution rendering method for PDA is introduced in [6]. It reduces the meshes of models that are far from the viewer. In [7] an X3D and H-Anim player is implemented by OpenGL ES [8]. Reference [9] and [10] propose a local rendering method based on client-server system. The server extracts the geometry that is visible for the client and sends it to the client for displaying. Reference [11] introduces an interactive walkthrough method of large 3D buildings on mobile devices. Culling algorithms are used to reduce 3D buildings complexity. In summary, because of the processors performance and batteries issues, local rendering for interactive mobile graphics mainly focus on reducing geometry complexity.

In this paper we present a new local rendering display method to address mobile 3D graphics. Our work introduces a client-server framework of progressive display for mobile graphics. This framework adopts mesh simplification on the server-side to reduce geometry complexity, while a unique detail file structure is proposed for extracting details on the server for progressive display. In addition, an efficient progressive reconstruction and display method based on this detail file structure is presented for progressive display on mobile devices. This display method consumes limited processing and memory resources on mobile devices.

II. PROGRESSIVE DISPLAY FRAMEWORK

This progressive display framework adopts the client-server architecture. First, as mentioned previously, the display panel of a mobile device is small, and the performance is relatively low. It is not feasible to render high detail 3D graphics for interactive application. Therefore geometry complexity must be reduced. Second, since mobile devices run in wireless network environment, the relatively narrow bandwidth and unreliability of wireless networks cause that transmission of a whole 3D model from the server to the devices consumes a lot of time and resource, and lead the users to wait for a long time before navigating the whole 3D model or 3D scene. For these reasons, progressive transmission and display are necessary. The progressive display means that a server transmits a base model to a mobile device for displaying. Then if necessary, this server transmits model details to the device for reconstructing an upper level model, and finally the mobile device can progressively reconstruct an original model if the user needs it.

978-1-4244-2948-6/09/$25.00 ©2009 IEEE

A. *Progressive simplification and details saving on server*

In this framework, a mesh simplification on the server is necessary. This operation comprises simplifying a high details model to a sparse model, which has fewer details, and saving the deleted vertices and triangles to a detail file. Base on this simplification, the server progressively simplifies the 3D model to generate a base model and a series of detail files. This is showed in Fig. 1.

In Fig. 1 M_0 is the original model, while M_n is considered the base model. The process of M_0 being simplified to M_n with a series of detail files is expressed as following:

$$M_0 \xrightarrow{S} (M_1, detail_1)\ldots \xrightarrow{S} (M_n, detail_n, detail_{n-1}, \ldots detail_1) \quad (1)$$

B. *Progressive reconstruction and display on mobile device*

Correspondingly, the mobile device receives the base model M_n and displays it firstly. Since this M_n has a small number of vertices and triangles, the user can navigate it interactively. Then if the user needs more details, the server transmits $detail_n$ to the device to reconstruct an upper level model M_{n-1}. Because of the new added vertices and triangles, the interactive performance ought to be slightly lower. If M_{n-1} is good enough or the performance is low enough, the user can just use this model for application. Otherwise, the system would progressively transmit and reconstruct more details models until they fulfill the user requirements, and even reconstruct the original model finally. It all depends on the users requirements between interactive performance and 3D model details. This process is showed in Fig. 2.

The graphics reconstruction operation is expressed as following:

$$(M_{i+1} \oplus detail_{i+1}) \xrightarrow{R} M_i \ , \quad i = 0,1,\ldots n-1 \quad (2)$$

The base model M_n together with all detail files can be used to reconstruct the original model M_0:

$$\underbrace{\underbrace{M_n}_{detail_n} \quad M_{n-1} \quad \ldots \quad \underbrace{M_i}_{detail_i} \quad M_{i-1} \quad \ldots \quad \underbrace{M_1}_{detail_1}}_{M_0} \quad (3)$$

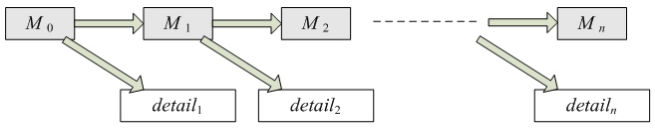

Figure 1. Progressive simplification and details saving on server.

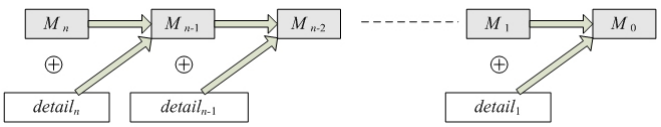

Figure 2. Progressive reconstruction and displaying on mobile device.

III. PROGRESSIVE RECONSTRUCTION AND DISPLAY METHOD ON MOBILE DEVICE

There are lots of papers do discuss mesh simplification, and we use half-edge collapse to simplify the mesh models progressively on the server. This part focuses on a progressive reconstruction and display method on mobile device. A classical progressive level of detail (LOD) display method is the Progressive Meshes (PM), which is first introduced and developed by Hoppe [12, 13, 14]. PM is lossless and can reconstruct the original meshes by vertex-split operations. The authors of [15] adopt the vertex-split operations to reconstruct meshes on desktop clients. But for mobile graphics, if PM is adopted to reconstruct meshes on mobile devices, this complicated vertex-split operation may not be suited for this kind of machines, which have limited computation and batteries capacity. Therefore a more efficient reconstruction method is needed for mobile devices. In this part a unique detail file structure and a very efficient progressive reconstruction and display method are proposed.

A. *Detail file structure for efficient reconstruction*

A detail file structure is very important because the details saving operation must be guided by this file structure, and the complexity of the reconstruction method on mobile devices depends on this file structure directly. During simplification the server simplifies M_i to M_{i+1}, and saving the deleted vertices and triangles to $detail_{i+1}$. Fig. 3 shows an example of half-edge collapse simplification, in which vertex 9 moves to vertex 4's position to merge with vertex 4.

The contents of detail files are read directly by mobile device for reconstruction. There are 3 lists in file $detail_{i+1}$:

List 1 records the deleted vertices of M_i being simplified to M_{i+1}. Thanks to the half-edge collapse, the vertices in M_{i+1} are subset of vertices in M_i. This means all vertices will remain the same or be deleted during simplification. In Fig. 3, vertex 9 is the deleted vertex of M_i. List 1 is for adding deleted vertices into M_{i+1} to reconstruct M_i.

List 2 records the deleted triangles of M_i being simplified to M_{i+1}. All triangles will remain the same, or be changed, or be deleted during simplification. A triangle is changed means that some of the vertices belong to this triangle are changed. In Fig. 3, triangles $T3$ and $T6$ are deleted, and $T1$, $T5$, $T7$, $T8$ are changed, because vertex 9 is merged with vertex 4. The merge operations of vertices will be recorded in list 3. List 2 is for adding deleted triangles into M_{i+1} to reconstruct M_i.

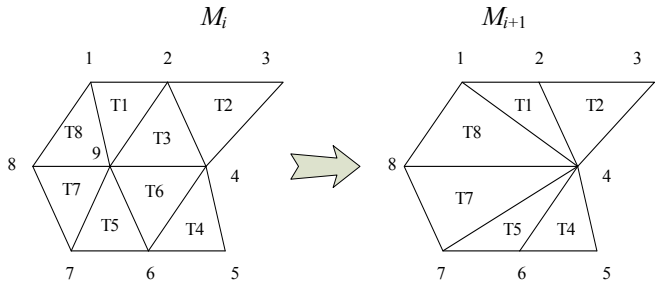

Figure 3. Half-edge collapse simplification.

List 3 is very special. It records the merge operations during simplification. But these are not the merge operations of M_i being simplified to M_{i+1}. It is a subset of all the merge operations of M_0 being simplified to M_1, and M_1 being simplified to M_2 ... M_{i-1} being simplified to M_i. All merge operations are recorded as "$A \rightarrow B$", which means vertex A moves to vertex B's position to merge with B. For all merge operations, there is a recursion process that the server finding all the proper ones to generate this subset:

The server checks all merge operations one after another. If the first one is "$A \rightarrow B$", then:

Step 1: B is the input element. If B is in list 1, "$A \rightarrow B$" will be added to list 3, and then the process jumps to Step 2. Else if B is not in list 1, the server checks another merge operation until all merge operations are checked.

Step 2: A is considered the affected one. The server finds all the merge operations "$x \rightarrow A$". Then A is the input element of Step 1.

During progressive simplification, all the indexes of vertices and triangles remain the same as in M_0. List 1 contains the vertex index of M_0 and 3D values of each vertex. List 2 contains the triangle index of M_0 and 3 vertices indexes of each triangle. List 3 contains the merge operations in the reverse order of progressive simplification, which means it is in the order of M_i, M_{i-1} ... M_0. All of these 3 lists in detail file are designed for efficient reconstruction on mobile devices. The reconstruction method will be given in the next part. Fig. 4 is the $detail_{i+1}$ file structure.

During progressive displaying, the mobile device has to display the base model first. The base model file structure is almost the same as the detail files, but slightly different. List 1 in base model is all the remaining vertices, while list 2 is all the remaining triangles. List 3 contains all the merge operations of every simplification. Fig. 5 is the base model file structure.

To reduce the file size of the detail files and the base model file, we use the Zlib library [16] to compress files on the server during the detail and base model saving operation, and to read files on the client during reconstruction and display.

List 1	Deleted vertices of M_i being simplified to M_{i+1} (Contains the vertex index of M_0 and 3D values of each vertex)
List 2	Deleted triangles of M_i being simplified to M_{i+1} (Contains the triangle index of M_0 and 3 vertices indexes of each triangle)
List 3	A subset of all the merge operations from M_i to M_0 (Contains the merge operations in the reverse order of progressive simplification)

Figure 4. Detail file structure of $detail_{i+1}$.

List 1	All the remaining vertices
List 2	All the remaining triangles
List 3	All the merge operations of every simplification

Figure 5. Base model file structure.

B. *Reconstruction and display method on mobile device*

The mobile device read the base model file and detail files to reconstruct a 3D model progressively. We assume that all the files have been transmitted to the mobile client already.

The method needs only 2 arrays in the mobile device for progressive display: vertices and triangles array. The vertices array length is equal to the number of M_0's vertices, while the triangles array length is equal to the number of M_0's triangles. Initially, all vertices and triangles in the arrays are marked as deleted, and they will be marked as non-deleted progressively during reconstruction. In this progressive display method only the non-deleted elements can be displayed.

The mobile device displays the base model first. After reading the list 1 and list 2 in base model file, the mobile device add all the remaining vertices and triangles into the correct position in those 2 arrays by the vertices indexes and triangles indexes. New added elements are marked as non-deleted. The mobile device can display the remaining vertices, but cannot display the remaining triangles at this moment. This is because some of the vertices belong to the remaining triangles are empty in the vertices array, since these vertices had been merged with some other vertices during simplification. Hence list 3 in base model is needed for assigning a value to the empty elements of the vertices array.

For example, Fig. 6 shows displaying a simplified mesh by a base model file. During simplification triangle $T2$ is deleted, while $T1$ is the remaining triangle and is changed by vertex 3 moving to vertex 4 as "$A \rightarrow B$". The remaining triangle "$T1$: 1-2-3" in list 2 is to be displayed, but only vertex 1, 2, 4 is the remaining vertex and vertex 3 in vertices array is empty. The merge operation "$A \rightarrow B$" in list 3 is used as "$A=B$". Hence we use "$A=B$" to calculate "3 = 4", which means the 3D values of vertex 4 is assigned to vertex 3. Hence the mobile device is displaying the triangle "1-2-4". During generation of the base file, we prefer recording the original "$T1$: 1-2-3" rather than the changed "$T1$: 1-2-4". Suppose we record the changed ones, the triangles array has to be updated frequently because when a new vertex is added during reconstruction, every triangle in the triangles array that has this vertex needs to be updated, and the mobile device has to search every element in triangles array to update them. Therefore we record the original ones, and just the vertices array needs to be updated during reconstruction.

If there are "$A \rightarrow B$" and then "$B \rightarrow C$" in list 3, we will calculate the "$B=C$" firstly and "$A=B$" subsequently to assign the 3D values of vertex C to vertex A. Therefore the merge operations in list 3 are in the reverse order of progressive simplification. We track the progressive simplification reversely for reconstruction.

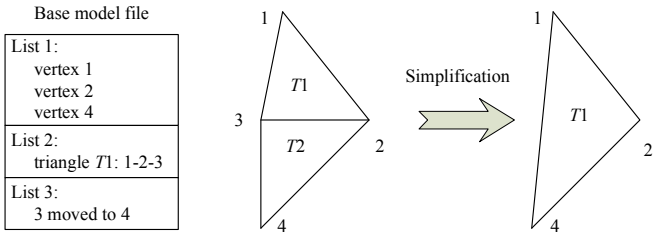

Figure 6. A base model file and a simplified mesh to be displayed.

Because all merge operations are included in list 3 in the base model file, all the elements in the vertices array will have 3D values after all merge operations in list 3 are read. But the vertices to which are assigned values by list 3 are still marked as deleted. The triangles array only has the remaining triangle, while the rest elements are empty and marked as deleted.

After displayed the base model, the mobile device can display more details progressively by reading the detail files. The process is the same as displaying the base model. Firstly the mobile device read list 1 in detail file to replace old values in vertices array by the new vertices, and read list 2 to add new triangles into triangles array. New added vertices and triangles are marked as non-deleted. List 3 is used to update the vertices values which are affected by new added vertices. For example, if B is a new added vertex, the operation "A=B" is needed for updating the value of A. But if B is not included in list 1, the operation "A=B" is not necessary since it has already been calculated during displaying the base model. Therefore list 3 in detail file is a subset of all the merge operations. And the means we build list 3 in detail file is based on this.

In our framework, since lots of operations are performed by the server, such as mesh simplification and detail files construction, during reconstruction the only one operation the mobile client needs to perform is reading the files and assigning values to the elements in those 2 arrays. In summary, the reconstruction method on mobile device can be expressed as following:

Step 1: Read list 1 in base model or detail files to add new vertices into the vertices array and mark them as non-deleted;

Step 2: Read list 2 in base model or detail files to add new triangles into the triangles array and mark them as non-deleted;

Step 3: Read list 3 in base model or detail files to update the values of the affected vertices in the vertices array.

This reconstruction method on mobile devices is extremely fast and run in O(n) time. And there is no extra data structure except 2 arrays. The processing and memory consumption are well limited. This feature makes this method very suitable for mobile devices which are powered by batteries.

IV. EXPERIMENTAL RESULTS

A Notebook with Intel Pentium M 1.5GHz is used as the server, and a PDA Mio 336 with Intel PXA255 300MHz is used as the mobile client. OpenGL ES is adopted for local rendering on the PDA.

A 3D dragon model with 5002 vertices and 10000 triangles is used as an original model. After 5 simplifications, a base model and 5 detail files are generated. We assume that all the files have been transmitted to the PDA already.

Table I lists the file size of the base model and 5 detail files. The original model's file size is 333 KB. The total file size of the base model and 5 detail files is less than the original one's.

TABLE I. THE FILE SIZE OF THE ORIGINAL MODELS, BASE MODELS AND THEIR DETAIL FILES

File Size of Each Files					
base.gz	***detail_5.gz***	***detail_4.gz***	***detail_3.gz***	***detail_2.gz***	***detail_1.gz***
44.2KB	9.8KB	15KB	22.4KB	35.5KB	59.7KB

Fig. 7 demonstrates the progressive display of the dragon model on the PDA. Fig. 7 (a) is the base model M_5, while Fig. 7 (f) is the original model M_0. It illustrates that details are added at each level during progressive display.

Fig. 8 demonstrates 6 snapshots of LOD dragon models token from PDA. Fig. 8 (a) is the base model M_5, while Fig. 8 (f) is the original model M_0. It illustrates that more details can be seen at each level during progressive display.

Table II lists the number of vertices, number of triangles, and the display frames per second (FPS) on PDA of all LOD models. FPS drops when new elements are added.

Table III lists the reconstructions running time on the PDA. The running time includes running time of file reading and reconstruction. It indicates that the reconstruction method in this paper is very efficient for mobile devices.

The statistic of FPS indicates that the original model is not suitable for real time interaction on mobile devices because of low FPS. Acceptable FPS for interactive mobile graphics can be gained by displaying the sparsest simplified models. If a user navigates this base model and the interaction is smooth enough, he/she can efficiently see more details by the progressive display method, because the reconstructions are extremely fast running on the PDA. Otherwise, the user would just use the base model for navigation if the FPS is barely enough. It all depends on the user requirements between interactive performance and 3D model details.

TABLE II. THE STATISTIC OF 6 LOD DRAGON MODELS

Elements	**Statistic of Each Dragon Models**					
	M_5 (base)	***M_4***	***M_3***	***M_2***	***M_1***	***M_0***
Vertices	887	1078	1419	2005	3049	5002
Triangles	1770	2152	2834	4006	6094	10000
FPS on PDA	8.6	7.4	5.9	4.5	3.1	2.1

TABLE III. THE RECONSTRUCTIONS RUNNING TIME ON THE PDA

Reconstruction Running Time				
M_5 to M_4	***M_4 to M_3***	***M_3 to M_2***	***M_2 to M_1***	***M_1 to M_0***
215 ms	343 ms	512 ms	810 ms	1404 ms

Figure 7. Progressive display of the dragon model on the PDA.

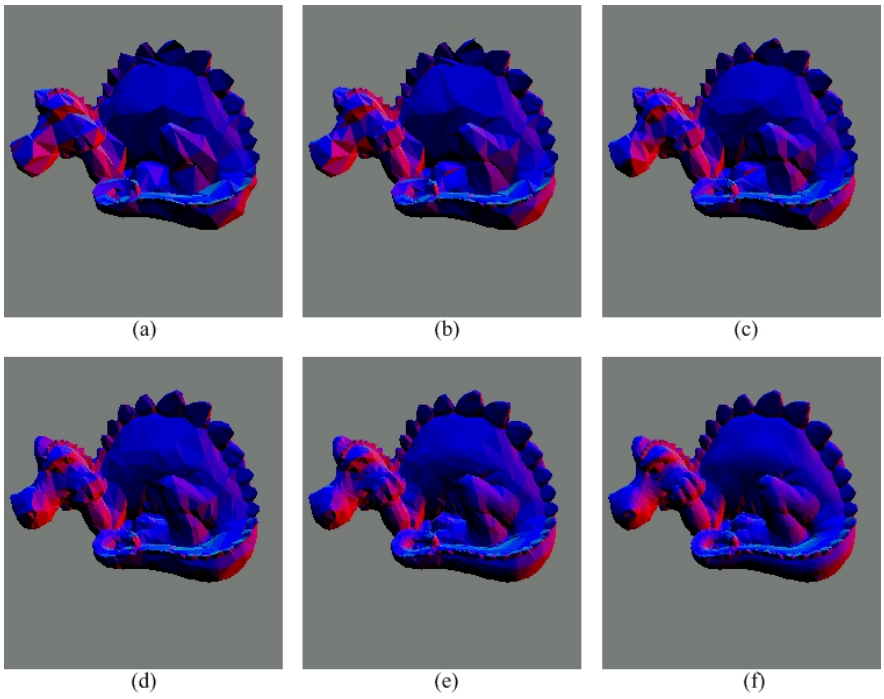

Figure 8. 6 snapshots of LOD dragon models token from PDA.

V. CONCLUSION

In this paper a new progressive display method for interactive mobile 3D graphics is presented. By displaying simplified mesh models, this method introduces real time interaction into mobile 3D graphics. And by progressively reconstructing models on the mobile devices, this method introduces progressive display into mobile 3D graphics. Users can obtain more 3D scene details by reconstruction if they need to. The reconstruction method is very efficient and lightweight, and it is well suited for the mobile devices which have limited computation and batteries capacity. This new method implements progressive display of interactive mobile 3D graphics efficiently. Interactive applications such as mobile products 3D demonstration, interactive walkthrough of 3D environment, etc. will benefit from this new method.

ACKNOWLEDGMENT

This work is supported by the projects (No. 60525213, No. U0735001) of NSFC, the Cultivation Fund of the Key Scientific and Technical Innovation Project (No. 706045) and RFDP (No. 20060558078) of Ministry of Education of China, and 863 Program (No. 2007AA01Z236).

The 3D model in this paper is a modification of the Phlegmatic Dragon model provided by EuroGraphics 2007 (http://www.cgg.cvut.cz/eg07/index.php?page=dragon).

REFERENCES

[1] F.Lamberti and A.Sanna, "A streaming-based solution for remote visualization of 3D graphics on mobile devices," *IEEE Trans. Visualization and Computer Graphics*, vol.13, no.2, 2007, pp. 247-260.

[2] C.F.Chang and S.H.Ger., "Enhancing 3D graphics on mobile devices by image-based rendering," in *Proc. 3rd IEEE Pacific Rim Conf. Multimedia*, 2002, pp. 1105-1111.

[3] K.S.Banerjee and E.Agu, "Remote execution for 3D graphics on mobile devices," in *Proc. IEEE Int. Conf. Wireless Networks, Communications and Mobile Computing*, 2005, pp. 1154-1159.

[4] A.Boukerche and R.W.N.Pazzi, "Remote rendering and streaming of progressive panoramas for mobile devices," in *Proc. 14th Annual ACM Int. Conf. Multimedia*, 2006, pp. 691-694.

[5] P.Bao and D.Gourlay, "A framework for remote rendering of 3-D scenes on limited mobile devices," *IEEE Trans. Multimedia*, vol.8, no.2, 2006, pp. 382-389.

[6] C.Zunino, F.Lamberti and A.Sanna, "A 3D multiresolution rendering engine for PDA devices," in *Proc. 7th World Multiconference on Systemics, Cybernetics and Informatics - Computer Science and Engineering*, 2003, pp. 538-542.

[7] D.Nadalutti, L.Chittaro and F.Buttussi, "Rendering of X3D content on mobile devices with OpenGL ES," in *Proc. 11th ACM Int. Conf. 3D Web Technology*, 2006, pp. 19-26.

[8] *OpenGL ES*, http://www.khronos.org/opengles/, 2008.

[9] J.Lluch, R.Gaitan, E.Camahort and R.Viv, "Interactive three-dimensional rendering on mobile computer devices," in *Proc. ACM SIGCHI Int. Conf. Advances in Computer Entertainment Technology*, 2005, pp. 254-257.

[10] J.Lluch, R.Gaitan, M.Escriva and E.Camahort, "Multiresolution 3D rendering on mobile devices," in *Proc. Computational Science - ICCS*, 2006, pp. 287-294.

[11] A.Mulloni, D.Nadalutti and L.Chittaro, "Interactive walkthrough of large 3D models of buildings on mobile devices," in *Proc. 12th ACM Int. Conf. 3D web technology*, 2007, pp. 17-25.

[12] H.Hoppe, "Progressive meshes," in *Proc. SIGGRAPH '96*, 1996, pp. 99-108.

[13] H.Hoppe, "View-dependent refinement of progressive meshes," in *Proc. SIGGRAPH '97*, 1997, pp. 189-198.

[14] H.Hoppe, "Efficient implementation of progressive meshes," *Computers & Graphics*, vol.22, no.1, 1998, pp. 27-36.

[15] B.Y.Chen and T.Nishita, "Multiresolution streaming mesh with shape preserving and QoS-like controlling," in *Proc. 7th ACM Int. Conf. 3D Web technology*, 2002, pp. 35-42.

[16] *Zlib*, http://www.zlib.net/, 2005.