Journal of Computer Science 8 (7): 1123-1133, 2012
ISSN 1549-3636

3D Mesh Streaming based on Predictive Modeling

[1]V. Vani, [2]R. Pradeep Kumar and [3]S. Mohan
[1]Department of Information Technology, Dr. N.G.P. Institute of Technology,
Anna University of Technology, Coimbatore, 641048, India
[2]Department of Computer Science and Engineering,
Adithya Institute of Technology, Anna University of Technology, Coimbatore, India
[3]Department of Computer Science and Engineering,
Dr. N.G.P. Institute of Technology, Anna University of Technology, Coimbatore, 641048, India

Abstract: The complexity in 3D virtual environment over the web is growing rapidly every day. This 3D virtual environment comprises a set of structured scenes and each scene has multiple 3D objects/meshes. Therefore the granular level of the block in a virtual environment is the object. In a virtual environment, it is required to give user interactions for every 3D object and at any point of time, it is enough if the system streams and brings in only the visible portion of the object from the server to the client by utilizing the limited network bandwidth and the limited client memory space. This streaming would reduce the time to present the rendered object to the requested clients. Further to reduce the time and effectively utilize the bandwidth and memory space, in the proposed study, an attempt is made to exploit the user interaction on 3D object and built a predictive agent which would minimize the latency in the rendering of the 3D mesh that is being streamed. The experiment result shows that the rendering time and cache miss rates are significantly reduced with the predictive agent.

Keywords: 3D virtual environment, Predictive Modeling (PrM), 3D streaming, 3D rendering, 3D mesh, visibility culler, user profiling, operation profiling

INTRODUCTION

In recent times, 3D modeling and rendering has gained attention over the internet and most of the multiuser virtual environment renders the entire world once it is fully downloaded from the server. Therefore, to get the first response from the server, the clients/users ought to wait till the entire model is downloaded and rendered. Due to the increased complexity of the 3D model, even with the high bandwidth user has to wait for a longer time to view the model. To reduce the waiting time of the user 3D streaming technique is made available to the users. Based on the user camera/eye position and orientation, the visible portion of the model is made available to the user by culling the invisible portions. In this study, an attempt is made towards reducing the waiting time of the user further by predicting the operation that would be performed by the users. A Predictive Agent (PA) is built after successful offline analysis carried out on profiles collected from 50 different users (aged 18-22, from engineering institutions with good visual and computer senses). As part of the analysis, the speed of the key press and the pattern of the keys pressed are taken for analysis across various 3D models. For experimentation purpose, 3D models of various sizes ranging from a few KBs to few MBs are considered with various shapes. The PA contains the typical key press and patterns of all users which will be used further to predict their navigation. This in turn helps to optimize the 3D streaming and rendering over web by reducing the time delay between user request and response.

Related works:

Progressive Mesh (PM): Progressive Mesh (PM), a method proposed by Hoppe (1996) and Cheng *et al*. (2011), shows how an arbitrary mesh is stored as a much coarser mesh together with a sequence of N detail records that indicate how to incrementally refine exactly back into the original mesh. Each of these records stores the information associated with a vertex split, an elementary mesh transformation that adds an additional vertex to the mesh. The PM representatives thus defines a continuous sequence of meshes of increasing accuracy, from which Level of Details (LOD) approximations of any desired complexity can be efficiently retrieved.

Decimation of triangular meshes: The goal of the decimation algorithm (Schroeder *et al*., 1992) is to

Corresponding Author: Vani V., Department of Information Technology, Dr. N.G.P Institute of Technology,
Anna University of Technology, Coimbatore, 641048, India

reduce the total number of triangles in a triangle (polygon) mesh, preserving the original topology and a good approximation to the original geometry.

Streaming of 3D progressive meshes: Streaming of progressive meshes (Cheng, 2008) enable users to view 3D meshes with increasing level of details, by sending a coarse version of a mesh initially, followed by a sequence of refinements to incrementally improve the quality. This study concentrates on how to send refinements to quickly improve the quality. An analytical model is developed to investigate the effects of dependency when the progressive meshes are sent over a lossy network and also proposed a receiver driven protocol to stream progressive meshes based on the user viewpoint in a scalable way.

Efficient and feature preserving:
Triangular mesh decimation: The method proposed by (Hussain *et al*., 2004) deals with a new automatic method for the decimation of triangular meshes in which at low levels of detail the system preserves visually important parts of the mesh and thus keeps the semantic or high level meaning of the model. The algorithm followed is based on greedy approach and exploits a new method of measuring geometric error employing a form of vertex visual importance that helps to keep visually important vertices even at low levels of detail and causes to remove other kinds of vertices, which do not profoundly influence the overall shape of the model.

3D model streaming based on JPEG 2000: 3D mesh (El-Leithy and Sheta, 2009) streaming method based on JPEG 2000 standard was proposed with the integration into an existed multimedia streaming server (Lin *et al*., 2007). In this method, the mesh data of a 3D model were first converted into a JPEG 2000 image and then based on the JPEG 2000 streaming technique, the mesh data were then transmitted over the Internet as a mesh streaming.

View dependent mesh streaming with minimal latency: The study proposed by (Kim, *et al*., 2004) presents a framework for view-dependent streaming of multi-resolution meshes. Here, the server dynamically adjusts the transmission order of the detail data with respect to the client's current viewpoint. By extending the truly selective refinement scheme for progressive meshes to client-server architecture, it accomplishes an efficient view-dependent streaming framework that minimizes network communication overhead to facilitate minimal latency of mesh updates for varying viewpoints.

Design of geometric streaming systems: A system was designed to stream large graphics environments from a central server to multiple clients. The streaming is transparent to the user who can treat remote models just like local ones. The streaming system automatically adapts to the rendering capabilities, network bandwidth and latency of the client and transmits an optimized model (Deb and Narayanan, 2004).

General and Automated Polygon Simplification (GAPS): The method uses an adaptive distance threshold and surface area preservation (Erikson and Manocha, 1998) along with a quadric error metric to join unconnected regions of an object. Its name comes from this ability to "fill in the gaps" of an object. The algorithm combines approximations of geometric and surface attribute error to produce a unified object space error metric.

Quadric based polygon surface simplification: Automatic simplification (Garland, 1999) of highly detailed polygonal surface models into faithful approximations containing fewer polygons. The system, examines the hierarchical structure that is induced on the surface as a result of simplification. This resulting hierarchy can be used as a multi-resolution model-a surface representation which supports the reconstruction of a wide range of approximations to the original surface model.

Proposed Method:
Predictive Model (PrM): The proposed predictive model is based on understanding user navigation in the virtual world. It is built based on the current camera position and orientation. Therefore only the visible vertices and faces of the selected triangular meshes are brought to the client. Simultaneously, based on the previous history collated from various user inputs, the next set of predicted vertices and faces are also pushed to the client with the help of the Predictive Agent (PA). This would reduce the time delay between the user request and response.

Analytical model: The main objective of the proposed study is to develop an analytical model based on the user interaction while viewing the 3D models over the network. The central idea is to predict the user navigation and construct an analytical model for every 3D object (3D meshes) using the PA. This predictive model hence would be useful in bringing the necessary surfaces during streaming so that rendering and response time can be reduced. To construct the predictive model (Predictive Agent: PA), the following notations have been used: Let S_v be a set of mesh vertices in the server and S_f be a set of corresponding mesh faces in the server for the selected 3D mesh and

Let C_v be the set of mesh vertices in the client where, $C_v \subseteq S_v$ and C_f be the set of corresponding mesh faces in the client where $C_f \subseteq S_f$.

On an Operation O_i, which can be an arbitrary rotation (θ_x, θ_y, θ_z), C_v and C_f can undergo change $\pm\Delta\{V_i\}$ and $\pm\Delta\{F_i\}$.

For $\pm\Delta\{V_i\}$:

$$+\Delta\{V_i\} \subseteq S_v, -\Delta\{Vi\} \subseteq Cv.$$

Where:

$+\Delta\{V_i\}$ = The set of vertices chosen from S_v

$-\Delta\{V_i\}$ = The set of vertices chosen out from C_v

For $\pm\Delta\{F_i\}$:

$+\Delta\{F_i\} \subseteq S_f, -\Delta\{F_i\} \subseteq C_f$

Where:

$+\Delta\{F_i\}$ = The set of faces chosen from S_f

$-\Delta\{F_i\}$ = The set of faces chosen out from C_f

Table 1 summarizes the notations used in our model.

Operation profiling: To profile the interaction performed by the user, basically the Rotation operation R_θ in any one of the directions: $+\theta_x/-\theta_x$, $+\theta_y/-\theta_y$, $+\theta_z/-\theta_z$ is considered.

For every key press during the rotation, a fixed angle of rotation is applied to the 3D object and outcome of the rotation generates updated eye position and eye orientation (eye refers to the camera position, which is the viewpoint of the user in the 3D world). Based on this operation, the speed of rotation is estimated which directly depends on the number of key pressed per second. The key presses would determine the amount of angle being rotated per second.

Based on the rotation output, the amount of change in the vertices and faces ($+\Delta\{V_i\}$ and $+\Delta\{F_i\}$) that ought to be transmitted to the client is predicted. The predicted faces and vertices only are transmitted to the client. The prediction, hence, would reduce the response time taken for rendering the visible portion of the 3D mesh based on the client input.

User profiling: To construct the predictive agent, an offline analysis has been carried out by considering 50 user profiles taken from a range of novice to professionals in interacting with 3D virtual world. The user profiles include, rate at which the key is pressed and the actual key that is pressed per user session on various 3D meshes considered for analysis. Using the collated user profiles, operation patterns are determined and predictive model is built. This process is considered to be a training session for the users before they actually navigate the virtual world.

Table 1: Notations

S_v	Server vertex set
S_f	Server face set
C_y	Client vertex set
Cf	Client face set
O_i	ith operation
$\Delta\{V_i\}$	Vertices changes
$\Delta\{F_i\}$	Faces changes
$R\theta$	Rotation
θ_x	Rotation about x axis
θ_y	Rotation about y axis
θ_z	Rotation about z axis

Table 2: User key press

S. No	Key pressed	Operation performed
1	Up	Rotate +Y
2	Down	Rotate –Y
3	Left	Rotate +X
4	Right	Rotate –X
5	Pg up	Rotate +Z
6	Pg down	Rotate -Z
7	Plus (+)	Zoom In
8	Minus (-)	Zoom Out

Once trained, the users would be able to get the rendered 3D models with a better response time across the network while interacting with the 3D web.

Representation of 3D streaming system: The proposed predictive model is used to stream the 3D data from the server to the requested clients in an effective manner. The implementation details of streaming system are discussed here. Figure 1 and 2 shows the 3D streaming system and its components as schematic diagram.

Client module: The client module comprises of 3 components namely Client Cache, Renderer and Visibility Culler. Also, the module receives a key press and the name of the 3D object to be viewed as the Client Input. The key pressed and the operation performed based on the key press is specified in the Table 2.

The operation performed as indicated in Table 2 would update the client's eye position and eye orientation for every key press during navigation into the virtual world.

Client cache: Inspired by the cache memory model (Hennessy and Patterson, 2007), a cache is built on the client side during the rendering. Initially, a client module receives the 3D mesh data from the server based on the current client's eye position and navigation. Once the 3D mesh data is brought to the client, it is set as referred data at the server end. In the client side, the data are stored in the Client Cache. Further, based on the client input, the 3D mesh data is received only when it is referred for the first time. Otherwise, data will be fetched from the Client Cache. In this case, retrieval is made from local and thus the transmission time and bandwidth is saved.

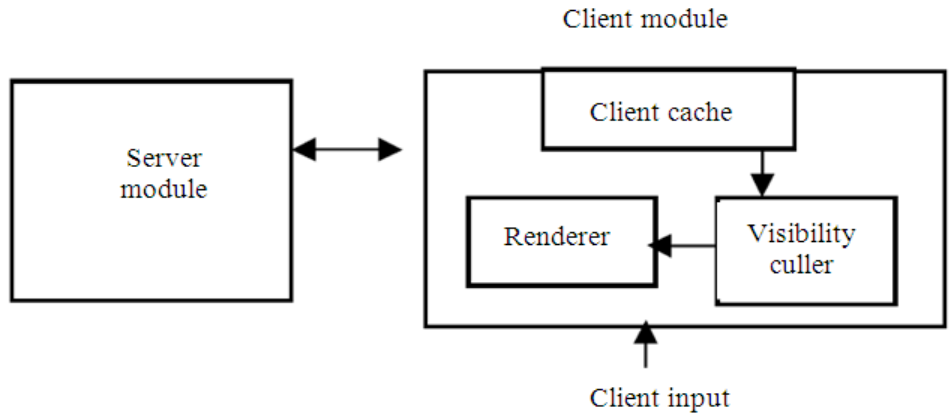

Fig. 1: Client module

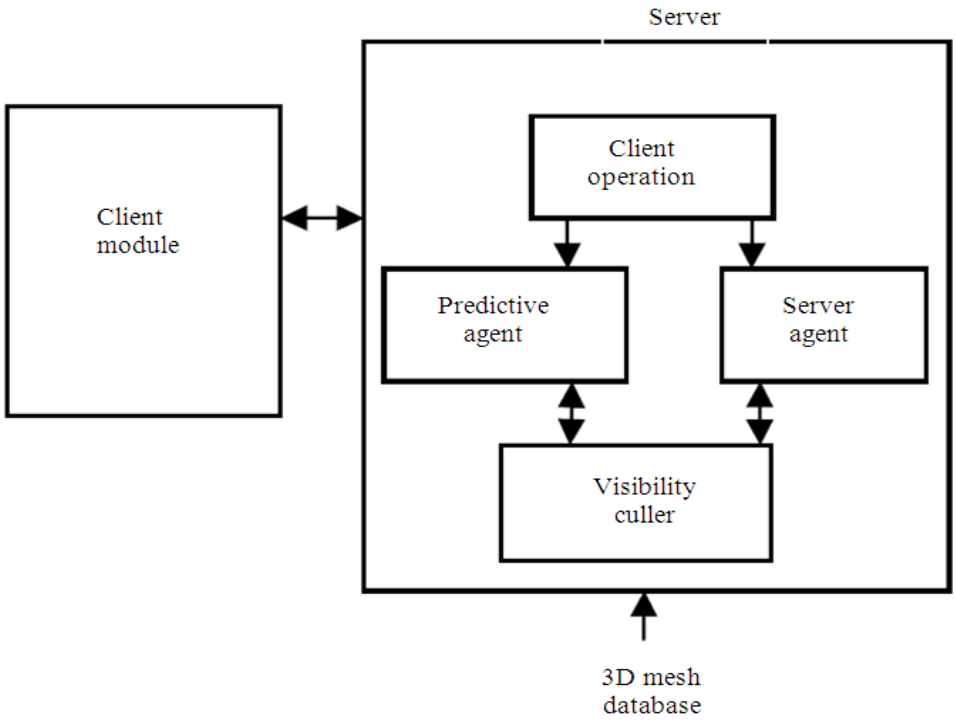

Fig. 2: Server module

Visibility culler (for client): The Visibility Culler is implemented using back face culling algorithm (Moller *et al*., 2008). Initially, based on the user key press, eye position and eye orientation are calculated. With the updated eye position and orientation, visible portion of the object is determined with the help of back face culling/hidden surface elimination algorithm. The client side visibility culler algorithm is activated when the required vertices and faces are already bringing to the client.

Renderer: The Renderer (Moller *et al*., 2008) is implemented to render the 3D data which is visible to the user at that particular eye position and orientation. The rendering speed is maintained with the help of predictive agent that rests in the server.

Server module: The server module comprises of 3 components namely Server Agent, Predictive Agent and Visibility Culler. Also, the module retrieves the 3D mesh data based on the client input from the underlying 3D Mesh database.

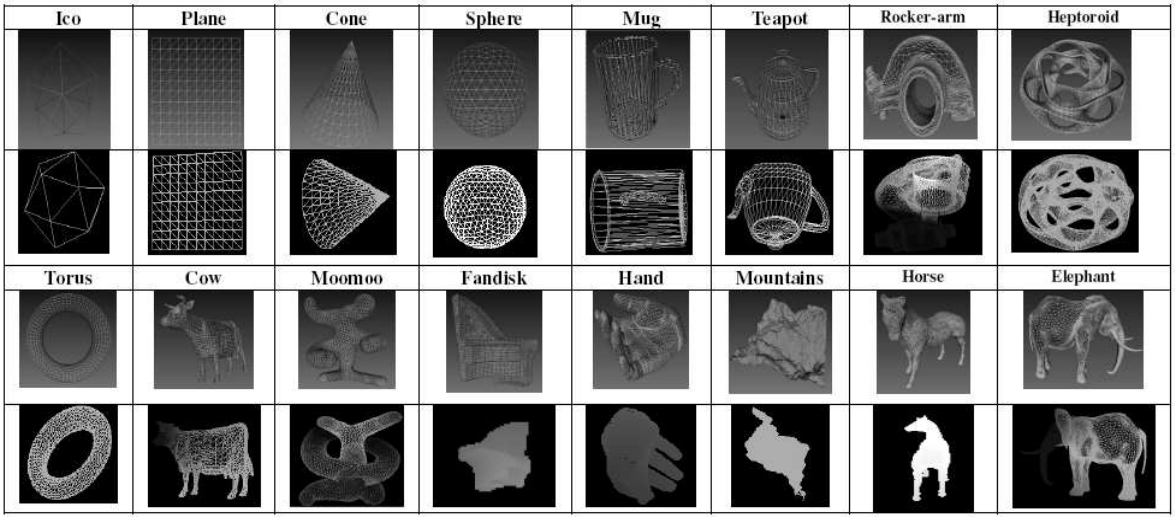

Fig. 3: Thumbnails of actual and Rotated/Zoomed 3D meshes (#http://www1.cs.columbia.edu/~cs4162/models)

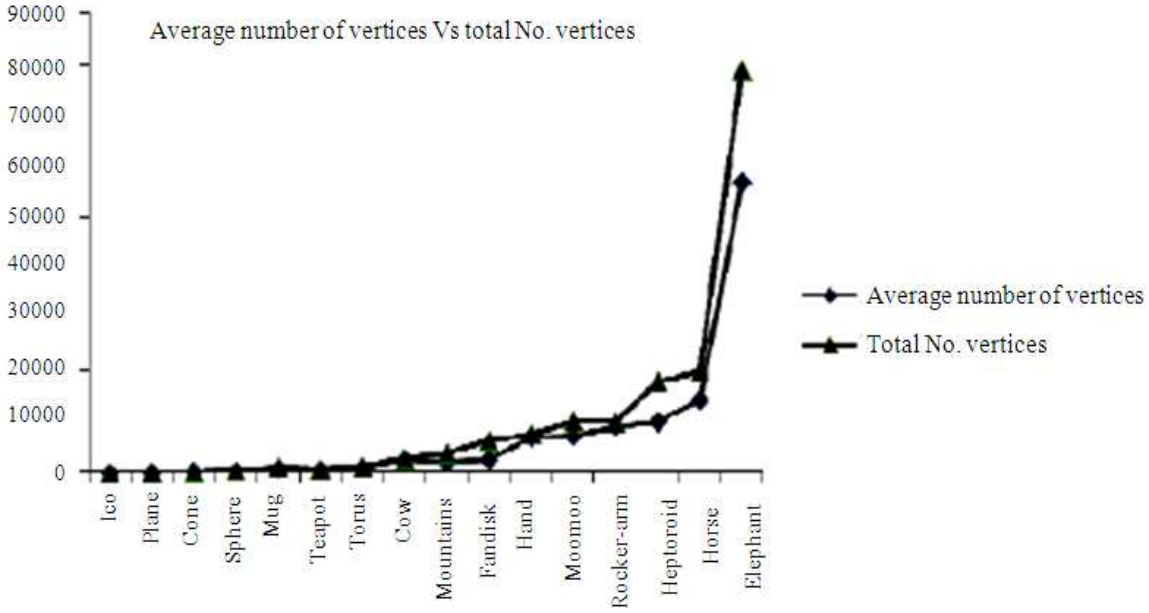

Fig. 4: Average Number of Vertices Cached for Multiple Access

Visibility culler (for server): Visibility Culler implements Back face culling algorithm (Moller *et al*., 2008) that determines the set of vertices and faces that has to be sent to the client whenever there is a request corresponding to those faces and vertices through user navigation.

Server agent: The Server Agent receives the client input and output from the visibility culler and store into the dynamic data structure with the reference bit is set against the corresponding vertices and faces that have to be sent to the client. Also, the server agent keeps track of the no. of times each vertices and faces have been referred.

Predictive agent: The Predictive Agent in parallel with the server agent also receives the client input and the output from the visibility culler. Based on the user profiling analysis carried out offline by collating the user interactions of 50 users across various models, the next key press is predicted and the corresponding 3D data are retrieved.

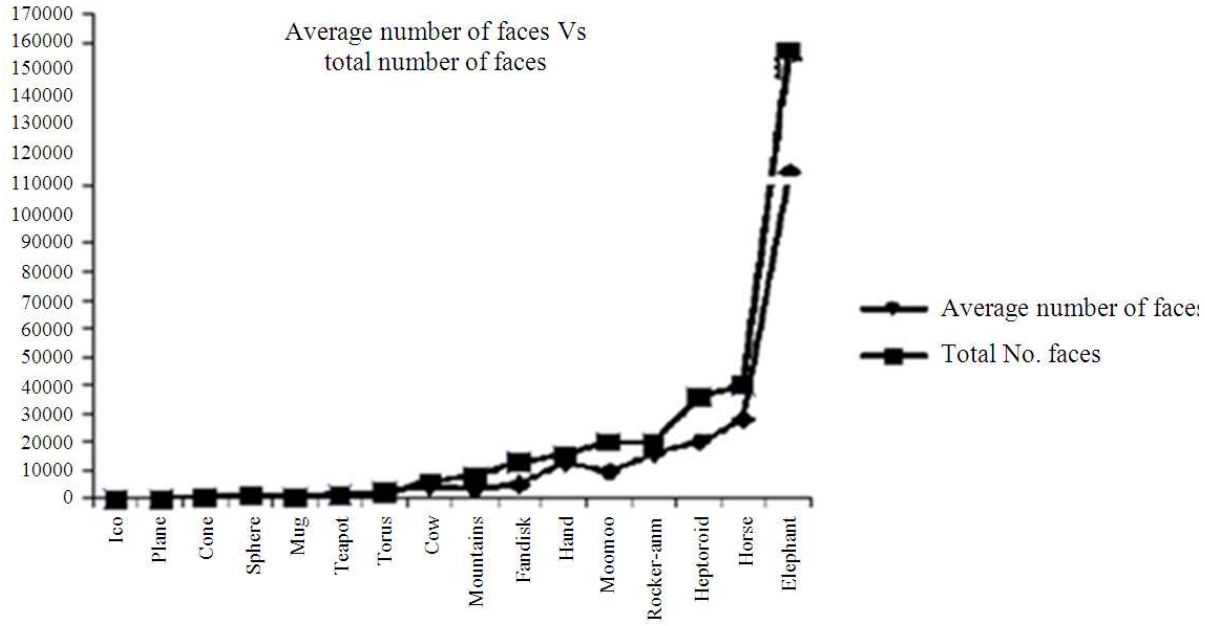

Fig. 5: Average number of faces cached for multiple accesses

Table 3: 3D mesh objects and its attributes

Model	Total No. vertices	Total No. of faces	File size
Ico	12	20	1 KB
Plane	121	200	4 KB
Cone	301	570	15 KB
Sphere	482	960	27 KB
Mug	1088	572	33 KB
Teapot	726	1452	37 KB
Torus	1200	2400	69 KB
Cow	2903	5804	161 KB
Moomoo	3890	7776	218 KB
Fandisk	6475	12946	369 KB
Hand	7609	15214	437 KB
Mountains	10201	20000	523 KB
Rocker-arm	10000	20000	603 KB
Heptoroid	17878	35840	1.15 MB
Horse	19851	39698	1.36 MB
Elephant	78792	157160	5.21 MB

Table 4: Average no. Of vertices and faces stored in cache for multiple Accesses across models

Model	Average number of vertices	Average number of faces
Ico	11	16
Plane	37	61
Cone	229	420
Sphere	410	794
Mug	891	463
Teapot	636	1180
Torus	999	1900
Cow	2670	4551
Moomoo	1992	3624
Fandisk	2585	4907
Hand	7027	12783
Mountains	7219	9470
Rocker-arm	9100	16000
Heptoroid	10011	20070
Horse	14268	27788
Elephant	56730	114240

These 3D data's reference bits are also set and it is sent to the client along with the requested data. This prediction would minimize the rendering latency and increase the cache hits. The 3D meshes[#] used are tabulated in Fig. 3. This table also shows the rotated or zoomed (in/out) 3D meshes. In each row, top one shows the actual and bottom one shows one of the screen shots of 3D mesh during the user interaction. The attributes of the 3D objects are given in Table 3.

Experimental Results and Discussions: To conduct the experiment and affirm that the predictive model for 3D mesh streaming and rendering would lessen the response time in rendering and reduce the cache miss rate, 16 standard 3D mesh models with various numbers of vertices and faces starting from simple 3D mesh model to the complex one are considered. Table 4-10 and Fig. 4-12 illustrates the various experimental results which indicate that the streaming using predictive agent is advantageous than downloading entire 3D model of the client.

It is found from these results that, by exploiting the viewpoint of the client, the visible portion of 3D meshes are streamed and rendered, instead of downloading the entire object to the client. This avoids the initial waiting time of the client. The client can quickly view the first response received from the server without much delay. Also, before the client requests for the next chunk of data by changing his viewpoint, the predictive agent would determine the probable move the client might make and client cache is updated if it is the demanded data.

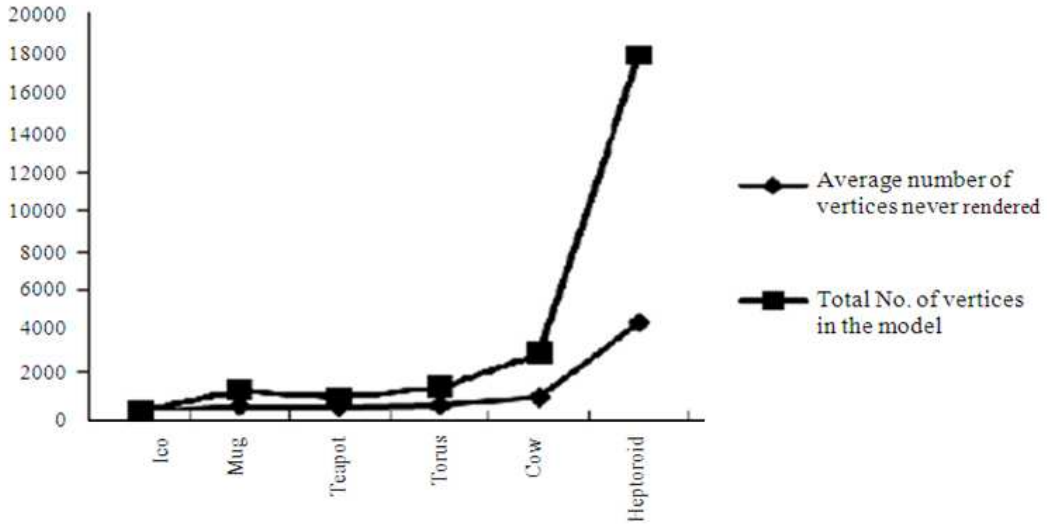

Fig. 6: Average number of vertices never rendered for multiple accesses by multiple users

Table 5: User interaction analysis of various models for multiple accesses by various users

Model	Rendered Vertices and Faces (RVF) /Never Rendered Vertices and Faces (NRVF)	User 1	User 2	User 3	User 4	User 5	Avg.
Ico	RVF	12	10	12	11	11	11
		16	12	17	13	14	14
	NRVF	0	2	0	1	1	1
		4	8	3	7	6	6
Mug	RVF	891	704	892	716	800	801
		459	380	460	372	428	420
	NRVF	197	384	196	372	288	287
		113	192	112	200	144	152
Teapot	RVF	701	407	685	550	491	567
		1246	658	1196	852	773	945
	NRVF	25	319	41	176	235	159
		206	794	256	599	679	507
Torus	RVF	979	797	1012	840	881	902
		1814	1442	1900	1526	1610	1658
	NRVF	221	403	188	360	319	298
		586	958	500	874	790	742
Cow	RVF	2303	2116	2465	2327	2184	2279
		3851	3434	4335	3055	3621	3659
	NRVF	600	787	438	576	719	624
		1953	2370	1469	1849	2183	1965
Heptoroid	RVF	14183	13032	15181	14331	13451	14036
		23781	21206	26769	24423	22360	23708
	NRVF	3695	4846	2697	3547	4427	3842
		12059	14634	9071	11417	13480	13480

This prediction reduces the cache miss rate and also the rendering time as the future data is made available in the cache before it is requested.

Table 4, Fig. 4 and 5 highlights the average number of vertices and faces brought to the client after multiple accesses across various models. It clearly shows that none of the instances all the vertices and faces is referred by the client. Therefore, we shall conclude that mesh saving and bandwidth saving can be achieved through streaming.

Table 5, Fig. 6 and 7 shows the results for multiple user interactions across various models after multiple simultaneous accesses. Once again the results prove that mesh saving and bandwidth reduction is possible through streaming.

Table 6 and Fig. 8 shows that number of meshes brought to the client initially and the result shows that on an average only about 40% of the meshes are saved in the server end itself and is not brought to the client. Table 8 and Fig. 10 shows the results of average user speed across various models in seconds for multiple degrees or key presses. This result is used to determine the time gap between the user interactions. This is studied for pushing the predicted 3D data to the client before it is requested.

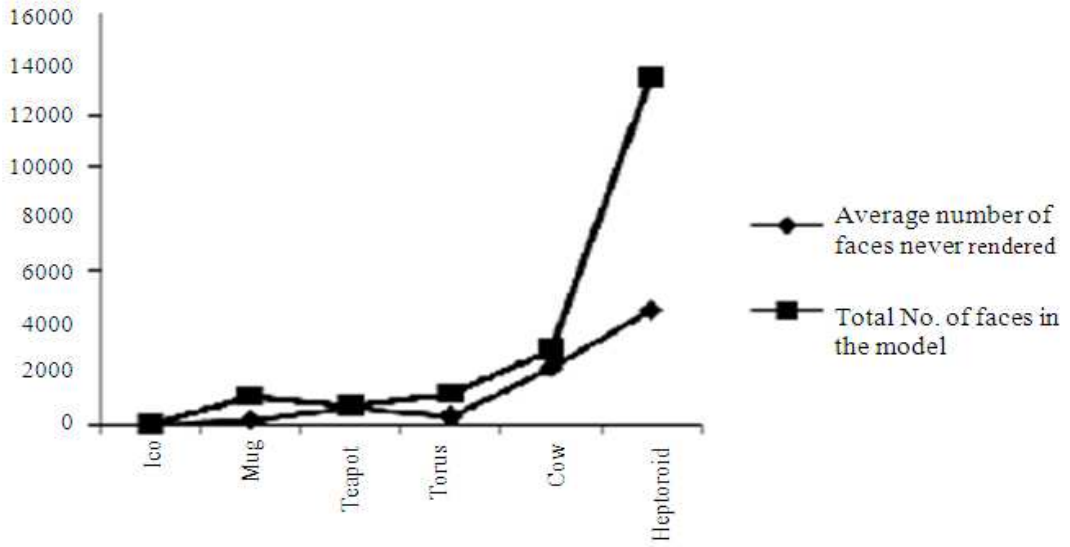

Fig. 7: Average number of faces never rendered for multiple accesses by multiple users

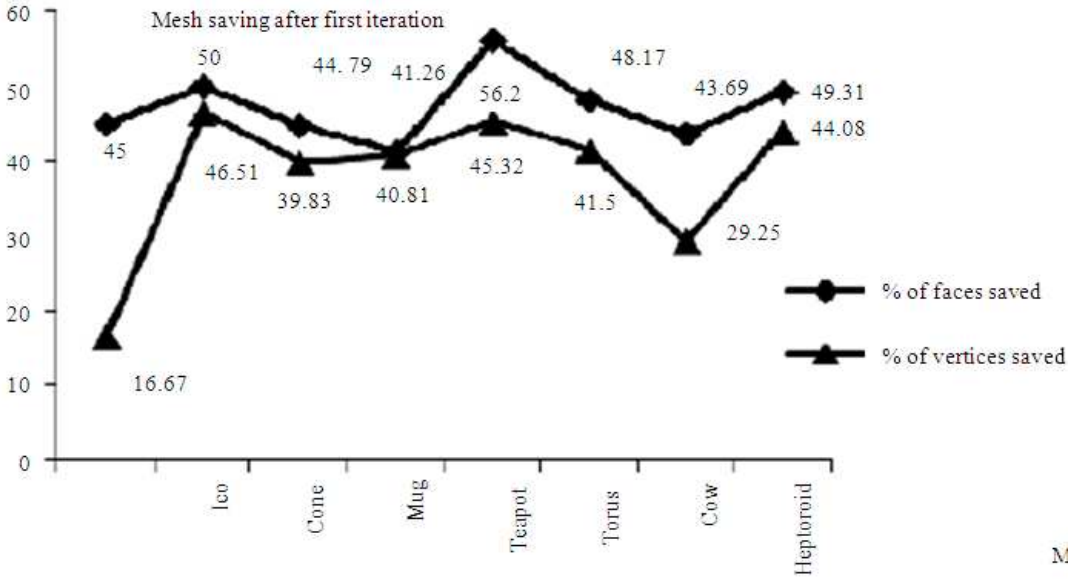

Fig. 8: Server Initial Mesh Saving

Table 6: Server initial mesh saving

Mesh savings initially (%)		
Model name	% of faces saved	%. of vertices saved
Ico	45.00	16.67
Cone	50.00	46.51
Sphere	44.79	39.83
Mug	41.26	40.81
Teapot	56.20	45.32
Torus	48.17	41.50
Cow	43.69	29.25
Heptoroid	49.31	44.08

Table 7: Server mesh saving after multiple accesses

Mesh savings after multiple accesses		
Model name	% of faces saved	%.of vertices saved
Ico	20.00	16.67
Cone	50.00	46,51
Sphere	17.60	39.83
Mug	19.76	40.81
Teapot	14.19	45.32
Torus	24.42	41.50
Cow	21.21	29.25
Heptoroid	44.87	44.08

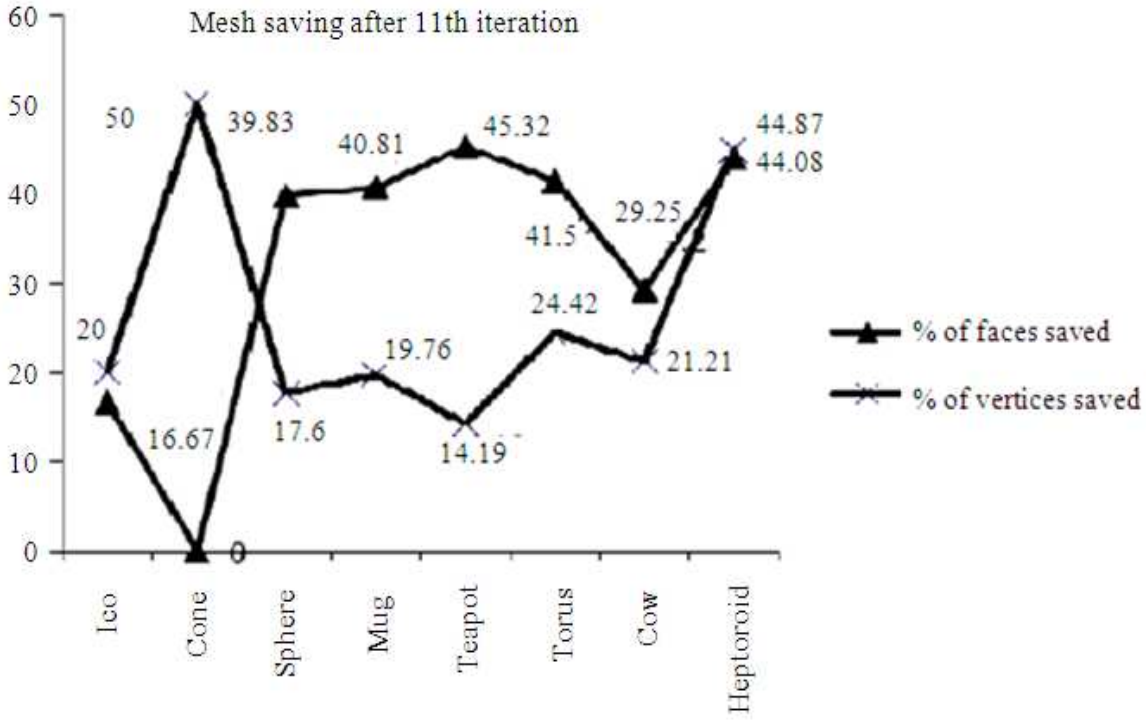

Fig. 9: Server mesh saving after multiple accesses

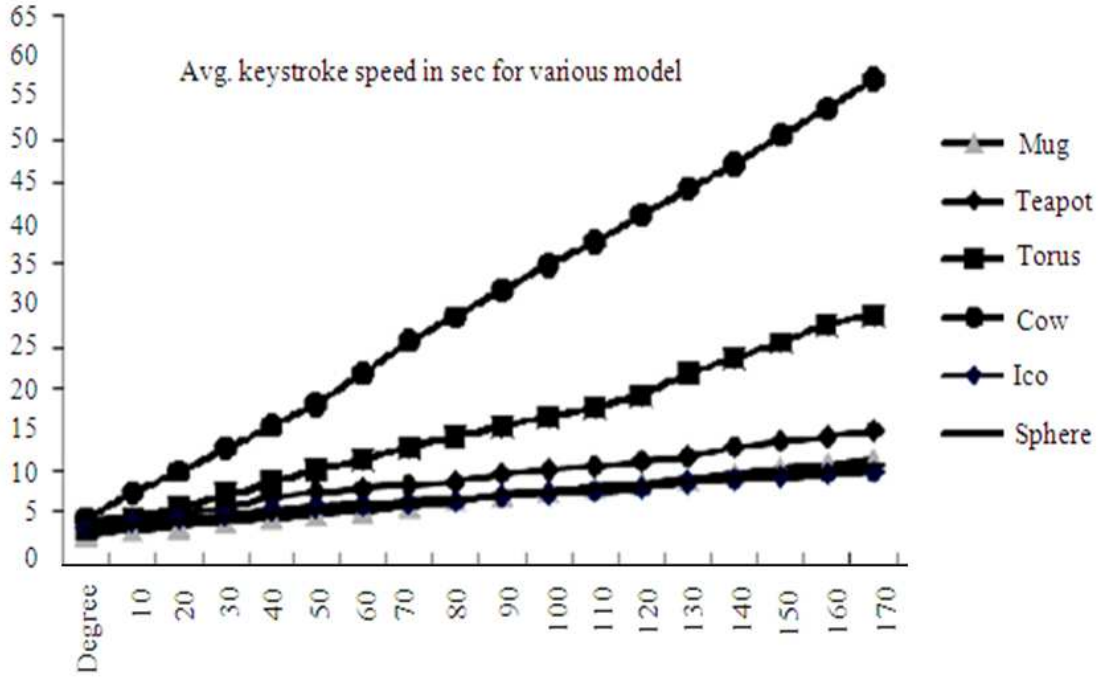

Fig. 10: Average user speed for various models

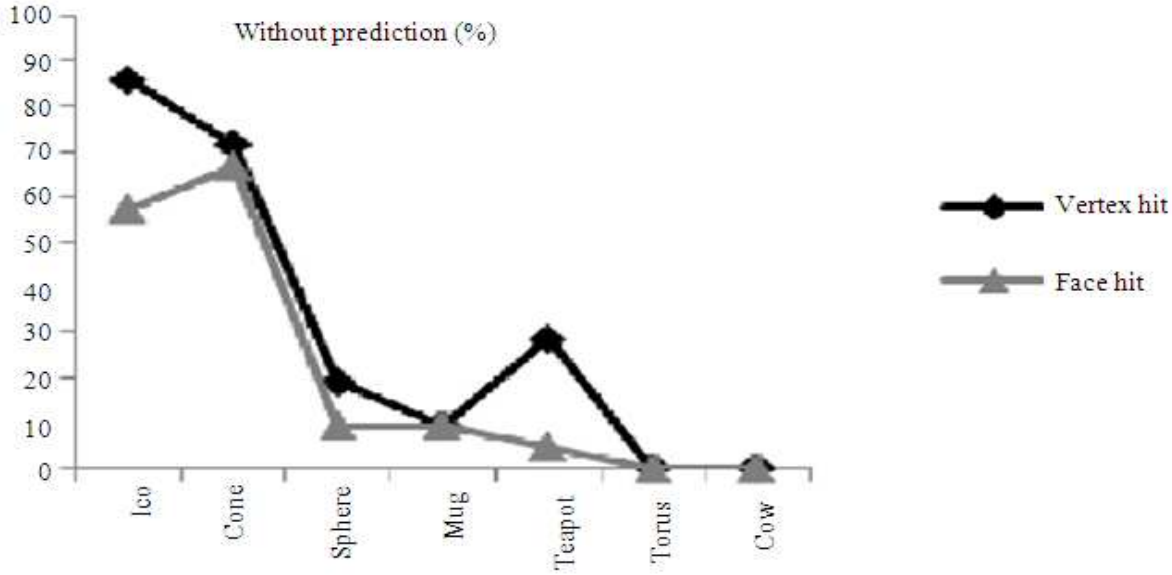

Fig. 11: Client cache hit without prediction for multiple accesses

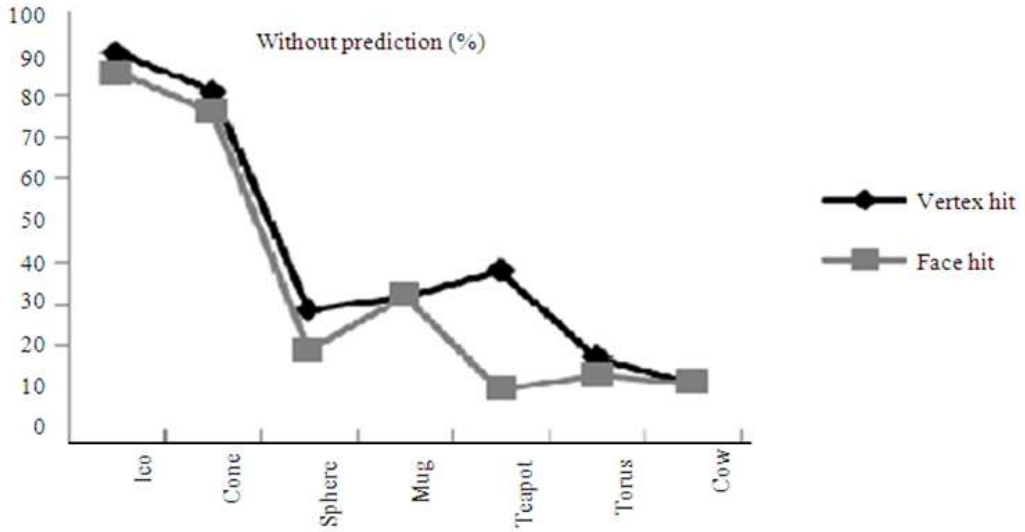

Fig. 12: Client cache hit with prediction for multiple accesses

Table 9 and Fig. 11 show the client cache hit/miss without including predictive agent. It clearly highlights when the complex model is accessed from multiple viewpoints, all the vertices and faces viewed already are not referred as a whole after quite a large number of accesses.

Table 10 and Fig. 12 show the client cache hit/miss with predictive agent. Since the next move is predicted and the corresponding faces and vertices are brought to the client well in advance before it is requested by the client, it is considered as a cache hit. The result proves that predictive agent could bring in the probable vertices and faces that would be referred by the client in comparison with nonproductive approach.

CONCLUSION

The proposed study addresses the need for streaming with predictive agent. The system attempts to stream the 3D data from the server to the client based on the viewpoint of the client by predicting the user's next move.

Table 8: Average user speed for various models

Average user speed for various models (in sec)						
Degree	Ico	Sphere	Mug	Teapot	Torus	Cow
10	3.0	2.9	2.3	4.0	2.9	4.2
20	3.9	3.5	2.8	4.4	4.3	7.3
30	4.2	3.9	3.2	5.0	5.6	9.9
40	4.6	4.3	3.8	5.4	7.0	12.7
50	5.3	4.8	4.3	6.8	8.5	15.4
60	5.6	5.3	4.7	7.4	10.1	18.1
70	5.8	5.8	5.1	7.9	11.3	21.8
80	6.1	6.2	5.7	8.3	12.9	25.7
90	6.3	6.6	6.5	8.6	14.1	28.6
100	6.9	7.1	7.0	9.6	15.3	31.9
110	7.2	7.5	7.5	10.1	16.5	34.9
120	7.5	7.9	8.0	10.5	17.6	37.7
130	7.9	8.3	8.5	11.1	19.1	40.9
140	8.6	8.7	9.0	11.7	21.8	44.1
150	8.9	9.3	9.7	12.8	23.6	47.1
160	9.3	9.7	10.3	13.5	25.5	50.5
170	9.6	10.2	10.8	14.0	27.6	53.7
180	9.8	10.7	11.3	14.8	28.8	57.3

Table 9: Client cache hit/miss without prediction for multiple accesses

Without prediction (%)				
Model name	Vertex hit	Vertex miss	Face hit	Face miss
Ico	85.710	14.29	57.14	42.86
Cone	71.430	28.57	66.67	33.33
Sphere	19.050	80.95	9.52	90.48
Mug	9.520	90.48	9.52	90.48
Teapot	28.570	71.43	4.76	95.24
Torus	0.000	100.00	0.00	100.00
Cow	0.000	100.00	0.00	100.00

Table 10: Client cache hit/miss with a prediction for multiple access

With prediction (%)				
Model name	Vertex hit	Vertex miss	Face hit	Face miss
Ico	90.48	9.52	85.71	38.09
Cone	80.95	19.05	76.19	23.81
Sphere	28.57	71.43	19.05	80.95
Mug	32.00	68.00	32.00	68.00
Teapot	38.10	61.90	9.52	90.48
Torus	17.39	82.61	13.04	86.96
Cow	11.11	88.89	11.11	88.89

It is proved that the predictive model reduces the waiting time of the client and he/she can see the first response quickly when it is compared with the full download of the model from the server to the client. Once the initial model is streamed and rendered on the client side, as per the client's further interactions, the referred 3D data are transmitted to the client from the server. If the required data is already in the client then the rendering process is carried out without streaming.

In this working model, an additional flavor is added to predict the probable move of the client across models by profiling multiple user interactions. A predictive agent is constructed and the result shows that the rendering time and cache miss rates are significantly reduced. The study can be further extended for a scene.

REFERENCES

Cheng, W., 2008. Streaming of 3D progressive meshes. Proceedings of the 16th ACM International Conference on Multimedia, (ICM' 08), ACM, New York, pp: 1047-1050. DOI: 10.1145/1459359.1459570

Cheng, W., W.T. Ooi, S. Mondet, R. Grigoras and G. Morin, 2011. Modeling progressive mesh streaming: Does data dependency matter? ACM Trans. Multimedia Comput. Commun. Appli. DOI: 10.1145/1925101.1925105

Deb, S. and P.J. Narayanan, 2004. Design of a geometry streaming system. International Institute of Information Technology.

Erikson, C. and D. Manocha, 1998. GAPS: General and automatic polygonal simplification. Proceedings of the 1999 Symposium on Interactive 3D Graphics, (IG' 99), ACM, USA, pp: 79-88. DOI: 10.1145/300523.300532

El-Leithy, S.T. and W.M. Sheta, 2008. Wavelet-based geometry coding for 3D mesh using space frequency quantization. Proceedings of the IEEE Symposium on Computers and Communications, Jul. 6-9, IEEE Xplore Press, Marrakech, pp: 1034-1039.

Garland, M., 1999. Quadric-based polygonal surface simplification. 1st Edn., Carnegie Mellon University, Pittsburgh, pp: 200.

Hennessy, J.L. and D.A. Patterson, 2007. Computer Architecture: A Quantitative Approach. 4th Edn., Morgan Kaufmann, Burlington, ISBN-10: 0123704901, pp: 704.

Hoppe, H., 1996. Progressive meshes. Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, (GIT' 07), ACM, USA, pp: 99-108. DOI: 10.1145/237170.237216

Hussain, M., Y. Okada and K. Niijima, 2004. Efficient and feature-preserving triangular mesh decimation. J. WSCG, 12: 1-3.

Kim, J., S. Lee and L. Kobbelt, 2004. View-dependent streaming of progressive meshes. Proceedings of the Shape Modeling Applications, Jun. 7-9, IEEE Xplore Press, South Korea, pp: 209-220. DOI: 10.1109/SMI.2004.1314508

Lin, N.H., T.H. Huang and B.Y. Chen, 2007. 3D model streaming based on JPEG 2000. IEEE Trans. Consumer Elect.ronics, 53: 182-190. DOI: 10.1109/TCE.2007.339523

Moller, T., E. Haines and N. Hoffman, 2008. Real-Time Rendering. 1st Edn., CRC Press, ISBN-10: 1568814240 pp: 1027.

Schroeder, W.J., J.A. Zarge and W.E. Lorensen, 1992. Decimation of triangle meshes. ACM, SIGGRAPH Comput. Graph., 26: 65-70. DOI: 10.1145/142920.134010