3D Mesh Streaming and Rendering – An approach based on Predictive Modeling

V.Vani [1], R. Pradeep Kumar [2], and Mohan.S [3]

1. **Dr.N.G.P. Institute of Technology, Anna University, Coimbatore, India – 641048**
2. **Adithya Institute of Technology, Anna University, Coimbatore, India– 641048**
3. **Dr.N.G.P. Institute of Technology, Anna University, Coimbatore, India– 641048**

Abstract
The complexity in 3D environment over the web is growing rapidly. This 3D environment comprises of set of structured scenes and each scene has multiple 3D objects/meshes. In such 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 server to the client by utilizing the limited network bandwidth and the limited client memory space. Further to reduce the time and effectively utilize the bandwidth and memory space, in proposed work, an attempt is made to exploit the user interactions on 3D object and built a Predictive Agent(PA) which would minimize the latency in rendering of the 3D mesh that is being streamed.
Keywords: *3D Environment, Predictive Modeling (P_rM), 3D Streaming, 3D Rendering, 3D mesh, Visibility Culler*

1. Introduction

In recent times, 3D modeling and rendering has gained attention over the internet and most of the multiuser 3D 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 paper, 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 to 22, from engineering institutions with good visual and computer senses). As part of the analysis, the speed of the key press and pattern of the keys pressed are taken across various 3D models. For experimentation purpose, 3D models of various sizes ranging from few KBs to few MBs are considered with various shapes.

2. Related Works

2.1 Progressive Mesh(PM)

Progressive Mesh (PM), a method proposed by Hoppes [1] & [9], 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 representation 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.

2.2 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 [2].

2.3 Streaming of 3D progressive meshes

Streaming of progressive meshes [4] 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 paper 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 lossy network and also proposed a receiver driven protocol to stream progressive meshes based on the user view point in a scalable way.

2.4 Efficient and Feature Preserving Triangular Mesh Decimation

The method proposed by [3] 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.

2.5 Quadric based polygon surface simplification

Automatic simplification [5] 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.

2.6 View dependent mesh streaming with minimal latency

The work proposed by [6] 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.

1. **Proposed Work**

1.1 Predictive Model (P_rM)

The proposed predictive model is based on understanding user navigation in the 3D 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.

1.2 Analytical Model

The main objective of the proposed work 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 & response time can be reduced. To construct the predictive model (***Predictive Agent: PA***), the following notations have been used:
Let $\boldsymbol{S_v}$ *be a set of mesh vertices in the server and* $\boldsymbol{S_f}$ *be a set of corresponding mesh faces in the server for the selected 3D mesh and Let* $\boldsymbol{C_v}$ *be the set of mesh vertices in the client where,* $\boldsymbol{C_v} \subseteq \boldsymbol{S_v}$ *and* $\boldsymbol{C_f}$ *be the set of corresponding mesh faces in the client where* $\boldsymbol{C_f} \subseteq \boldsymbol{S_f}$*.*

On an Operation $\boldsymbol{O_i}$ *, which can be an arbitrary rotation*$(\Theta_x, \Theta_y, \Theta_z)$ *,* $\boldsymbol{C_v}$ *and* $\boldsymbol{C_f}$ *can undergo a change* $\pm\Delta\{V_i\}$ & $\pm\Delta\{F_i\}$*. For* $\pm\Delta\{V_i\}$ *:* $+\Delta\{V_i\} \subseteq S_v$ *,* $-\Delta\{V_i\} \subseteq Cv$ *where,* $+\Delta\{Vi\}$ *is the set of vertices chosen from* ***Sv*** *and* $-\Delta\{V_i\}$ *is the set of vertices chosen out from* ***Cv.*** *For* $\pm\Delta\{F_i\}$*:* $+\Delta\{F_i\} \subseteq S_f$ *,* $-\Delta\{F_i\} \subseteq C_f$ *where,* $+\Delta\{F_i\}$ *is the set of faces chosen from* $\boldsymbol{S_f}$ *and* $-\Delta\{F_i\}$ *is the set of faces chosen out from* $\boldsymbol{C_f}$*.*

1.2.1 Operation Profiling

To profile the interaction performed by the user, basically the Rotation operation $\mathbf{R_{\Theta}}$ in any one of the directions: $+\mathbf{\Theta_x}$/- $\mathbf{\Theta_x}$, $+\mathbf{\Theta_y}$/- $\mathbf{\Theta_y}$, $+\mathbf{\Theta_z}$/- $\mathbf{\Theta_z}$ is considered.
For every key press during the rotation, a fixed angle of rotation is applied on the 3D object and outcome of the rotation generates updated eye position and eye orientation (*eye refers to the camera position, which is the view point of the user in 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 ($\mathbf{+\Delta\{V_i\}}$**and** $\mathbf{+\Delta\{F_i\}}$) 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.

1.2.2 User Profiling

To construct the PA, an offline analysis has been carried out by considering 50 user profiles taken from a range of novice to professionals in interacting with 3D world. The user profile includes, 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 world. 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.

1.3 Representation of 3D streaming system

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

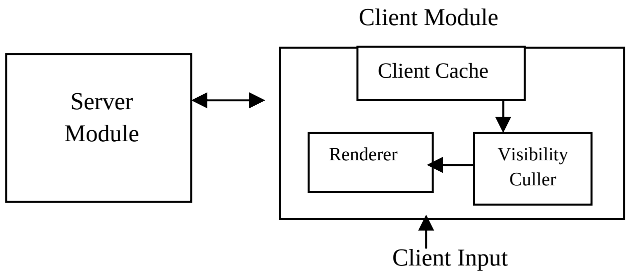


Figure 1: Client Module

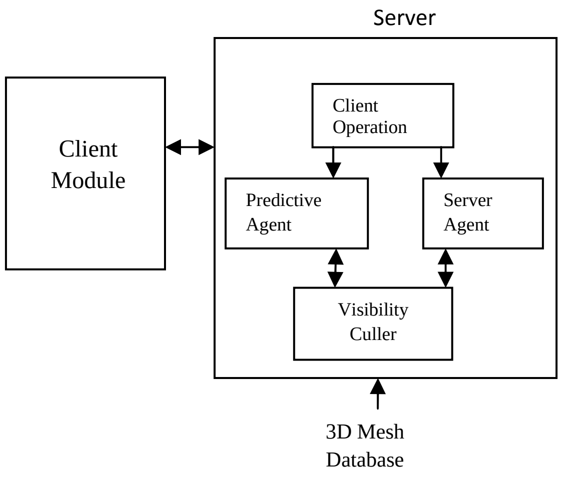


Figure 2: Server Module

1.3.1 Client Module

The client module comprises of 3 components namely *Client Cache, Renderer* and *Visibility Culler*. Also, the module receives a key press and 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 1.

Table 1: 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	PgUP	Rotate +Z
6	PgDOWN	Rotate -Z
7	PLUS (+)	Zoom In
8	MINUS(-)	Zoom Out

The operation performed as indicated in Table 1 would update the client's eye position and eye orientation for every key press during navigation into the world.

Client Cache

Inspired by the cache memory model [7], a cache is built on the client side during the rendering. Initially, 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* in 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.

Visibility Culler (for Client)

The *Visibility Culler* is implemented using back face culling algorithm [8]. Initially, based on the user key press, eye position and eye orientation is 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 brought to the client.

Renderer

The *Renderer* [8] 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.

1.3.2 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*.

Visibility Culler (for Server)

Visibility Culler implements Backface culling algorithm [8] 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.

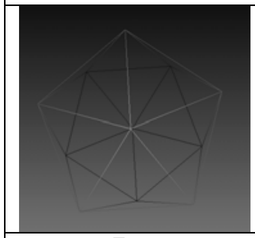
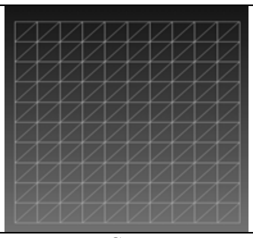
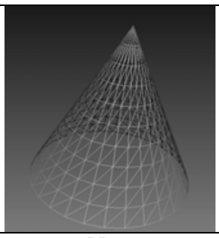
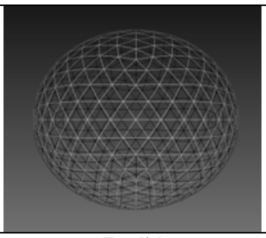
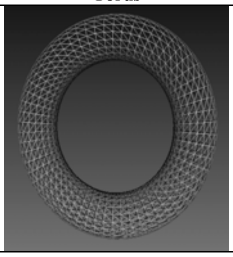
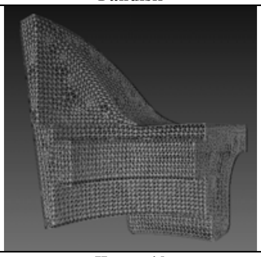
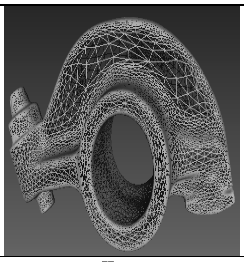
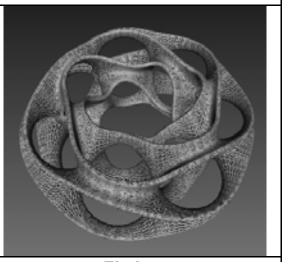
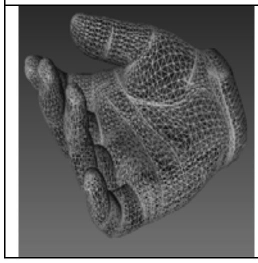
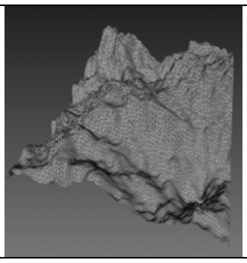
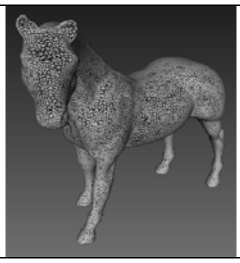
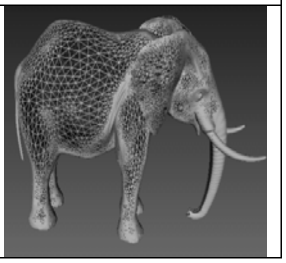
Predictive Agent

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 is retrieved. 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 attributes of the 3D objects are given in Table 2.The 3D meshes[#] used are tabulated in Table 3.

Table 2: 3D mesh objects and its attributes

S.No	**Model**	**Total No. Vertices**	**Total No. of Faces**	**File Size**
1.	Ico	12	20	1 KB
2.	Plane	121	200	4 KB
3.	Cone	301	570	15 KB
4.	Sphere	482	960	27 KB
5.	Mug	1088	572	33 KB
6.	Teapot	726	1452	37 KB
7.	Torus	1200	2400	69 KB
8.	Cow	2903	5804	161 KB
9.	Moomoo	3890	7776	218 KB
10	Fandisk	6475	12946	369 KB
11	Hand	7609	15214	437 KB
12	Mountains	10201	20000	523 KB
13	Rocker-arm	10000	20000	603 KB
14	Heptoroid	17878	35840	1.15 MB
15	Horse	19851	39698	1.36 MB
16	Elephant	78792	157160	5.21 MB

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

Ico	Plane	Cone	Sphere
			
Torus	**Cow**	**Moomoo**	**Fandisk**
			
Mug	**Teapot**	**Rocker-arm**	**Heptoroid**
			
Hand	**Mountains**	**Horse**	**Elephant**
			

2. Experimental Results & Implications

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 number of vertices and faces starting from simple 3D mesh model to the complex one are considered. Table 4 to Table 8 and Figure 3 – Figure 8 illustrates the various experimental results which indicates that the streaming using predictive agent is advantageous than downloading entire 3D model to the client.

It is found from these results that, the client can quickly view the first response received from the server without much delay. Also, before the client requests for next chunk of data by changing his view point, the predictive agent would determine the probable move the client might make and client cache is updated if it is the demanded data. 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, Figure 3 and Figure 4 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 and Figure 5 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 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

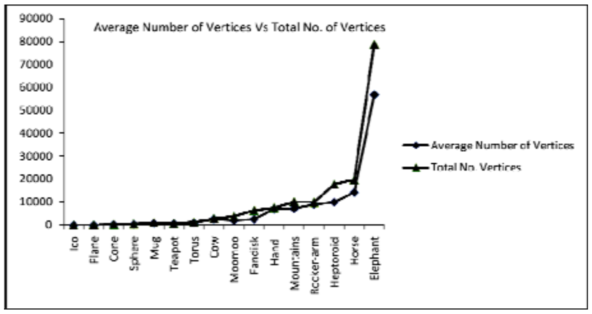


Figure 3: Average Number of Vertices Cached for Multiple Access

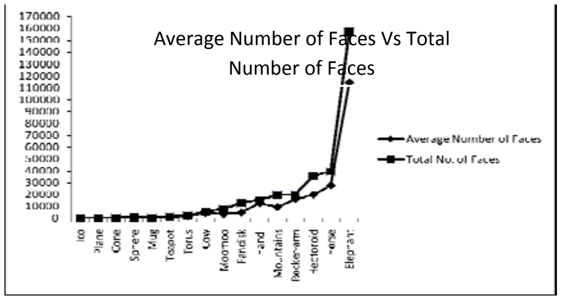


Figure 4: Average Number of Faces Cached for Multiple Accesses

Table 5: Server Initial Mesh Saving

Mesh Savings Initially (%)		
Model Name	% of Faces Saved	%.of Vertices Saved
Ico	45	16.67
Cone	50	46.51
Sphere	44.79	39.83
Mug	41.26	40.81
Teapot	56.2	45.32
Torus	48.17	41.5
Cow	43.69	29.25
Heptoroid	49.31	44.08

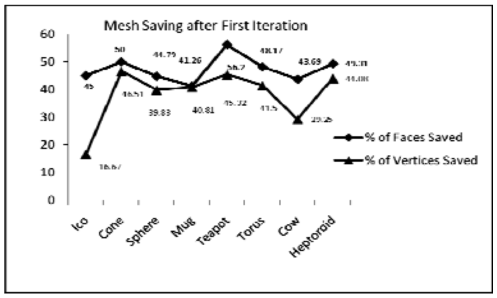


Figure 5: Server Initial Mesh Saving

Table 6 and Figure 6 results shows that considerable amount of meshes are saved after multiple accesses by a user.

Table 6: Server Mesh Saving after Multiple Accesses

Mesh Savings after Multiple Access		
Model Name	**% of Faces Saved**	**%.of Vertices Saved**
Ico	20	16.67
Cone	50	46,51
Sphere	17.6	39.83
Mug	19.76	40.81
Teapot	14.19	45.32
Torus	24.42	41.5
Cow	21.21	29.25
Heptoroid	44.87	44.08

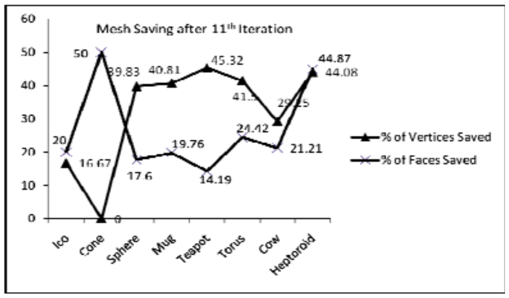


Figure 6: Server Mesh Saving After Multiple Accesses

Table 7 and Figure 7 show the client cache hit/miss without including predictive agent. It clearly highlights when the complex model is accessed from multiple view points, all the vertices and faces viewed already are not referred as a whole after quite a large number of accesses.

Table 7: Client Cache Hit/Miss without Prediction for multiple accesses

Without Prediction (%)				
Model Name	**Vertex Hit**	**Vertex Miss**	**Face Hit**	**Face Miss**
Ico	85.71	14.29	57.14	42.86
Cone	71.43	28.57	66.67	33.33
Sphere	19.05	80.95	9.52	90.48
Mug	9.52	90.48	9.52	90.48
Teapot	28.57	71.43	4.76	95.24
Torus	0	100	0	100
Cow	0	100	0	100

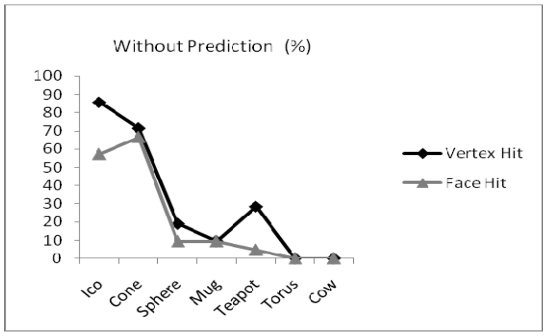


Figure 7: Client Cache Hit without Prediction for multiple accesses

Table 8 and Figure 8 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 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 non predictive approach.

Table 8: Client Cache Hit/Miss With 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	68	32	68
Teapot	38.1	61.9	9.52	90.48
Torus	17.39	82.61	13.04	86.96
Cow	11.11	88.89	11.11	88.89

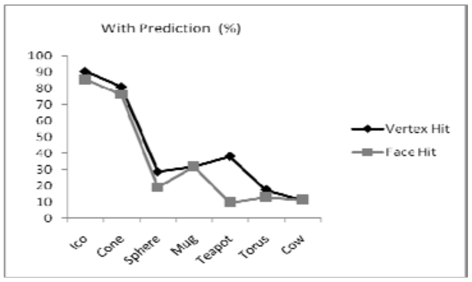


Figure 8: Client Cache Hit with Prediction for multiple accesses

3. Conclusion

The proposed work 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 view point of the client by predicting the user's next move. 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 at 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 work can be further extended for a scene.

4. References

1. Hugues Hoppe.1996. Progressive meshes. In *Proc. SIGGRAPH*, pages 99-108.
2. Soumyajit Deb and P. J. Narayanan. 2004. Design of a geometry streaming system. In *Proc. ICVGIP*, pages 296-301.
3. Muhammad Hussain, Yoshihiro Okada, Koichi Niijima. *2004.* Efficient and Feature-Preserving Triangular Mesh Decimation. *Journal of WSCG, Vol.12, No.1-3., ISSN 1213-6972 WSCG'2004. Plzen, Czech Republic.* Copyright UNION Agency – Science Press
4. Wei Cheng. 2008. Streaming of 3D progressive meshes. In *Proceedings of the 16th ACM international conference on Multimedia* (MM '08). ACM, New York, NY, USA, 1047-1050. DOI=10.1145/1459359.1459570 http://doi.acm.org/10.1145/1459359.1459570
5. Michael Garland. 1999. Quadric-Based Polygonal Surface Simplification. Ph.D. Dissertation. CMU-CS-99-105.
6. Junho Kim, Seungyong Lee, and Leif Kobbelt.2004.View-dependent streaming of progressive meshes. In *Proc. SMI'04*, pages 209-220.
7. John L. Hennessy and David Patterson. 2007. Computer Architecture – A Quantitative Approach. Fourth Edition.Morgan Kaufmann publishers.
8. Tomas Akenine-Moller, Eric Haines, Naty Hoffman**.** 2008. Real-Time Rendering. Third Edition. A.K.Peters Ltd.Welesley, Massachusetts.
9. Wei Cheng, Wei Tsang Ooi, Sebastien Mondet, Romulus Grigoras, and Geraldine Morin. 2011. Modeling progressive mesh streaming: Does data dependency matter?. *ACM Trans. Multimedia Comput. Commun. Appl.* 7, 2, Article 10 (March 2011), 24 pages. DOI=10.1145/1925101.1925105 http://doi.acm.org/10.1145/1925101.1925105

Vani.V, is a Professor & Head in the Department of Information Technology, Dr.N.G.P. Institute of Technology, Coimbatore, India. She holds PG degree in Computer Cognition Technology from University of Mysore and her graduation is on Computer science and Engg. Currently she is pursuing her PhD in Anna University of Technology, Coimbatore, India. She has over 13 years of teaching, research and industry experience. Her areas of research include Computer Vision and Graphics, 3D computing.

R.Pradeep Kumar is a Professor & Head, Department of Computer Science and Engg, Adithya Institute of Technology, Coimbatore, India. He holds PhD in computer science from University of Mysore. His areas of research include Video Analytics, Symbolic data, Knowledge Engineering. He is currently supervising 7 PhD candidates under Anna University.

Mohan.S is a Professor & Head, Department of Computer Science & Enng, Dr.N.G.P. Institute of Technology, Coimbatore, India. He holds PhD in Computer Science from University of Mysore. Over 13 years of experience and having more than 25 publications and his areas of research include Video Analytics, Computer Vision and Graphics. He is currently supervising 4 PhD candidates under Anna University.