


IBM Research Report

Adaptive Rendering of 3D Models Over Networks Using Multiple Modalities

Ioana M. Martin

IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

 Research Division
Almaden · T.J. Watson · Tokyo · Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties). Copies may be requested from IBM T.J. Watson Research Center, 16-220, P.O. Box 218, Yorktown Heights, NY 10598, USA (email: reports@us.ibm.com). Some reports are available at <http://domino.watson.ibm.com/library/CyberDig.nsf/home>.

Adaptive Rendering of 3D Models Over Networks Using Multiple Modalities

Ioana M. Martin

IBM T. J. Watson Research Center
30 Saw Mill River Road, Rm. H4-B32
Hawthorne, NY 10532
Tel: +1-914-784-6108

ioana@us.ibm.com

ABSTRACT

Content adaptation for the delivery of multimedia data such as text, images, audio, and video has been extensively studied in recent years. In contrast, techniques for adaptive delivery of 3D models to clients with different rendering capabilities and network connections are just beginning to emerge. In this paper we describe the principles involved in the design and development of an adaptive environment for rendering of 3D models over networks. This environment monitors the resources available and selects the appropriate transmission and representation modalities to match these resources. We illustrate the functionality of this environment through a prototype client-server implementation and we report experimental results obtained with this prototype.

Keywords

3D graphics over networks, hybrid rendering, adaptive selection, transcoding, universal access.

1. INTRODUCTION

In the scientific and medical domains an increasing number of applications are beginning to take advantage of remote visualization and analysis. In the industrial sector, companies have already begun to migrate towards fully electronic design and e-commerce. In this context, community boundaries are being redefined and new models of organization and interaction emerge. Networking infrastructures struggle to accommodate an ever-increasing variety of clients and servers inter-connected by communication links of various types and capabilities. This heterogeneity makes it difficult for servers to provide a level of service that is appropriate for every client that requests access to three-dimensional graphics content.

So far, a significant body of work has been dedicated to the challenges of universal access related to the delivery of traditional multimedia content such as text, images, audio, and video. However, less attention has been paid to 3D digital content, as true market opportunities for 3D graphics over networks have just recently begun to emerge. In this paper, we describe a set of design principles that we believe are fundamental for the efficient transmission of 3D models to clients spanning a wide range of graphics capabilities and connection types. We present an adaptive environment embodying these principles, as well as experiments with a prototype software we developed.

The main underlying idea is that *adaptive service* both increases the quality of service and reduces the end-to-end latency perceived by clients. For traditional multimedia types, *transcoding* has proven successful in serving variations of the

same object at different sizes and using different modalities. Transcoding, which is defined as a transformation that is used to convert multimedia content from one form to another [1], can be naturally extended to 3D data. By their very nature, 3D models are amenable to access through various representation modalities, that typically imply trade-offs between complexity, interaction, and download times.

This paper is organized as follows: section 2 contains a survey of existing literature pertaining to various aspects of transmission and rendering of complex 3D models, as well as content-adaptation techniques used for other types of multimedia. In section 3 we formulate a set of design principles for adaptive delivery of 3D data and we present the logical organization of a client-server platform built upon these principles. The tools necessary to monitor the characteristics of the transmission environment, the performance model used, and our adaptive selection algorithm are described in sections 4 and 5, respectively. Architectural details regarding the development and implementation of a prototype system, as well as experimental results are summarized in section 6.

2. BACKGROUND

Transcoding of text, images, audio and video has been extensively studied in recent years. For example, in the TranSend project [2], [3], adaptation to client variability is termed “distillation” and is achieved by image compression, reduction of image size and color space, and video conversion to different frame rates and encodings. In the InfoPyramid [4], a multiresolution multimodal hierarchy is used to represent content items on a Web page. This hierarchy is used to drive the selection of the best version of an item that delivers the most value for a given set of client resources.

The delivery of 3D data over networks has been traditionally investigated mainly in the context of networked virtual environments (see [5] for a survey). Such environments have emerged as simulation applications for the Department of Defense running on supercomputers and as networked games and workstation demonstrations running on graphics workstations and game machines. They are targeted towards the creation of telepresence, i.e., the illusion that other users are visible from remote locations. The main challenge in such environments is maintaining a consistent state among a large number of clients that share and access portions of a 3D scene at interactive rates [6]. At the network level, a protocol was proposed in [7] in response to this challenge (see also [8]). At the application level, a number of strategies developed to meet this challenge have evolved into technologies routinely used for optimization of 3D transmission and rendering in domains beyond high-performance

simulations and games. Such technologies include 3D model compression [9]-[11] for reduced storage requirements and faster delivery, model simplification (see [12] for a survey) and level-of-detail management [13], [14] for improved graphics performance, streaming techniques [15]-[17] for progressive downloading of 3D data, as well as various image-based methods [18]-[22] that replace all or parts of a model with images, thus trading freedom of interaction for reduced geometric complexity.

In general, methods for rendering 3D models in client-server environments can be classified into three major categories, according to where the rendering takes place. *Client-side methods* do not involve any rendering on the part of the server. The geometry is downloaded to each client that requests it and the client is responsible for rendering it. A number of transformations can be applied to the model before it is downloaded, including simplification or conversion to a progressive representation consisting of a simplified model and a sequence of refinement records that are downloaded gradually. Client-side methods are well suited for applications for which real-time interaction is paramount to viewing the model, assuming that the client has the ability to store and render the corresponding data.

Using *server-side methods*, the model is fully rendered on the server and the resulting image or set of images is transmitted to clients. Such methods are useful when clients with limited resources and graphics capabilities require high-quality images of large models. The main tradeoffs are the loss of real-time interaction with the model (i.e., due to the inherent latency of the network, images are delivered to clients at non-interactive rates) and a potential increase in server response times as the number of clients concurrently accessing the server increases.

Finally, *hybrid methods* partition a model into parts that are rendered on the server (e.g., background geometry) and parts that are downloaded and rendered on the client (e.g., foreground objects). Such methods have the advantage that they reduce the geometric complexity of the data being transmitted by replacing parts of it with images. However, determining whether a part of a model should be rendered on the server or on the client is not a trivial task and user interaction with the hybrid representation may be limited. Alternatively, the server could render high and low-resolution versions of a 3D model and send the residual error image and the low-resolution geometry to the client [23] (see also [24]). In this case, the task of the client is to render the coarse model and to add the residual image to restore a full quality rendering. Such a method is typically adequate for use in low-latency environments in which the servers have high-performance graphics capabilities.

Although all of the methods described perform well for certain types of 3D models and system configurations, they have not been designed to deal with client variation. We reported a first effort to combine existing techniques into an adaptive framework for transmission of 3D data in [25]. The Network Graphics Framework (NGF) provides improved service by dividing the transmission of a model into segments corresponding to several levels of detail and by selecting the most appropriate delivery mechanism for each level. While it proves the feasibility of adaptive transfer, the NGF is limited in that models are viewed as monolithic entities that are delivered at various resolutions using the best of several transmission methods available and the final image is rendered using a single representation for the entire

model. The design principles outlined in this paper are based in part on lessons learned from this early work, and are at the basis of a new prototype system that takes the idea of adaptive 3D transfer a step further, as described next.

3. PRINCIPLES OF ADAPTIVE DESIGN

Adaptive delivery of 3D models over networks must take into account information about the characteristics of the 3D data, the capabilities of various client platforms to which the data is being delivered, the load and capabilities of the servers, as well as user preferences. We have identified several design principles that we believe are fundamental to adaptive transfer and rendering of 3D models:

1. Models should be regarded as collections of one or more *components*, thus allowing for efficient management and refined schemes for perceptual measurements. A component defines an atomic part of a model that corresponds to a visually meaningful entity and can be individually transmitted from a server to a client. The partitioning of the model into components may be defined at model creation time or on the fly, and may range from a simple decomposition into connected components to an organization into a complex data structure reflecting semantic or spatial grouping criteria.

2. Combining different modalities for representing model components can achieve better transmission and rendering performance than the use of a “one-size-fits-all” strategy. Examples of modalities include polygonal mesh representations using indexed face sets, progressive meshes, 2D images, depth images, bounding boxes with or without textures, and basic shapes (e.g., sphere, cylinder, cone, box). The modalities may be computed and delivered at various levels of resolution.

3. The selection of the most appropriate modality for each component should account for the importance of that component to the final rendering of the model.

4. The selection mechanism should also take into account the resources available to predict the performance of a modality in a given context.

5. Last but not least, the selection should accommodate user preferences, in terms of specific modalities to be used for certain model components, resource budget allocation, and trade-off resolution.

The work described in this paper uses these principles to facilitate accessing and rendering of 3D models over networks to match the attributes of diverse clients. Ultimately, it provides an environment for automatically generating and delivering hybrid representations that combine 2D and 3D modalities. In addition to a multi-component model organization scheme and an adaptive selection mechanism, this environment includes a monitoring tool that keeps track of the dynamically changing characteristics of the transmission and rendering contexts, as well as a transcoding engine that drives the actual conversion of the 3D data to modalities selected by the selection mechanism. Figure 1 illustrates the logical organization of these components within the environment.

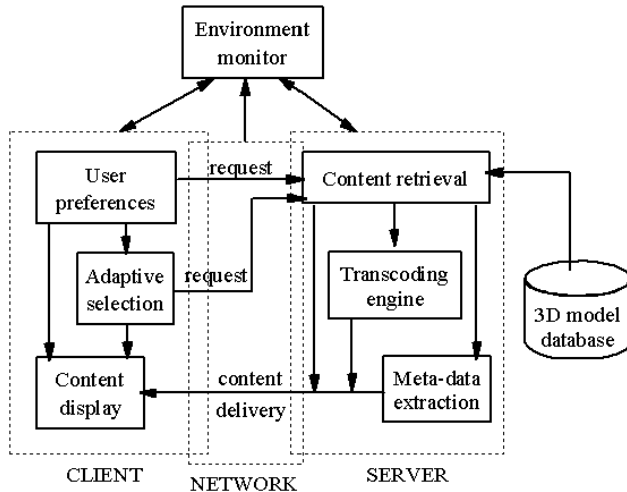


Figure 1. Logical flow of control in an adaptive client-server setup: a monitoring tool records the characteristics of the environment, such as server load, network delay, client and server rendering capabilities. This data is used in conjunction with information about the model to select suitable modalities for transmission and rendering of the model components.

When a client makes a request for a 3D model to a server, it first receives *meta-data* information about the requested model. This information allows the framework to estimate the performance characteristics for each of the modalities available. The *perceptual importance* of each model component, i.e., its contribution to the perception of a final rendering of the model, is also estimated. An adaptive mechanism selects the best modality available for each component that fits into the budget and delivers it to the client. The components are processed in decreasing order of their importance.

4. ENVIRONMENT MONITORING

A monitoring tool is necessary to record the events taking place in the environment in which model transfer and rendering occurs, and based on these, to provide quantitative information to the selection process.

We characterize the state of a given client-server setup in terms of four *state parameters*:

- the rendering capability of the client,
- the rendering capability of the server,
- the load on the server, and
- the performance of the communication link between the two.

Instead of attempting to map values of low-level performance counters describing CPU, memory, I/O, and network behavior to values corresponding to the four state parameters considered, we record measurements that are easier to interpret from an application’s perspective. Hence, we use the average frame rate (in frames per second) to describe the rendering capabilities of a machine (client or server) for a discrete range of model sizes and several types of rendering. We measure server load in terms of the average time between the receipt of a request by the server and the processing of that request. We measure network performance in terms of latency and bandwidth.

For each of the state parameters we maintain *history* information. The history is initialized from benchmark data once for each state parameter, when the server is set up and at the beginning of each client session. The history is subsequently updated dynamically with actual measurements taken as data is transferred. In the remainder of this section we describe the creation, updating, and usage of history data for the client rendering capability and the network performance. The server rendering capability and the server load are monitored in a similar fashion and therefore are not described separately.

4.1 Measuring Client Rendering Performance

A benchmark test is optionally performed at the beginning of each client session to determine the frame rates achievable on a given client for a preset number of rendering types (e.g., wireframe, shaded, with or without textures). By performing this benchmark at the beginning of each client session, we aim to capture more accurate initial measurements that reflect the client load. Alternatively, if the benchmark is not performed at the beginning of a session, the history is initialized with default values (e.g., measurements recorded when the client was installed). We have designed our benchmarks to render different size data sets for a preset number of frames, and we average the measurements collected for each set.

As models are downloaded, rendered, and manipulated by users, the frame rate history is updated with measurements performed on these models. To limit the size of the history and to ensure that only most recent data is used for performance estimation, we maintain a relatively small, fixed number of samples for each entry, and we overwrite old samples based on a least-recently used (LRU) policy.

The goal of the selection process is to identify for each model component the modality that brings the most value to the final rendering of the model for a given resource budget. The value of a particular modality is estimated in terms of visual quality and degree of interaction. The time necessary to render a modality is estimated based on the history information, taking into account modalities already downloaded for other components.

4.2 Measuring Network Performance

In general, network performance is measured in two fundamental ways: bandwidth (i.e., throughput) and latency (i.e., delay). The former is measured by the number of bits that can be transmitted over the network in a certain period of time, whereas the latter corresponds to how long it takes a message to travel from one end of the network to the other.

In our framework, we are interested in estimating the time necessary to transfer a given modality from a server to a client. When the connection is set up, a benchmark is performed to measure transfer times for different size packets. To reduce inaccuracies due to path asymmetries and differences in the source and destination clocks, we measure the round-trip delays of messages, rather than one-way latency. According to [26], the relationship between transfer time, bandwidth, and data size can be expressed as:

$$TransferTime = RTT + TransferSize / Bandwidth,$$

where *RTT* represents the round-trip time of the network and is used to account for a request message being sent across the network and the data being sent back.

Figure 2 illustrates the benchmark measurements taken for two client-server configurations over 10Mbps and 100Mbps connections, respectively. For small data sizes, the transfer time is latency bound, and the bandwidth available does not significantly influence the transfer time. In contrast, for large data, the more bandwidth there is, the faster the data is delivered. As in the case of the frame rate measurements, the delay history is used to estimate the transfer time for modalities during the selection process. We approximate the transfer time of a modality of a given size with the shortest time recorded for that size [27]. If the actual size does not have a corresponding entry in the history, we approximate it by the next largest size for which history data is available.

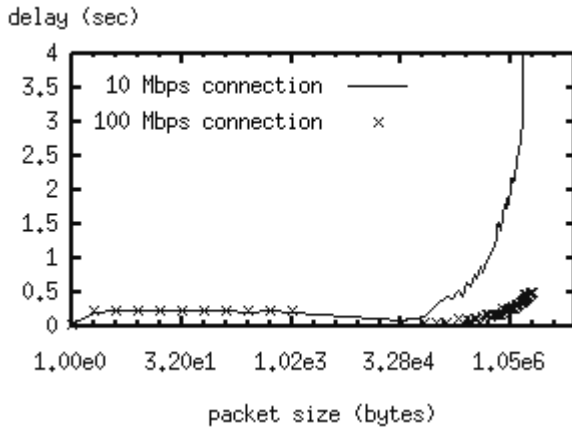


Figure 2. Benchmark measurements for two client-server configurations with different bandwidth connections. We use a logarithmic scale to show relative performance.

5. TRANSCODING OF 3D CONTENT

The transcoding engine consists of a collection of transcoding modules that convert 3D content to various 2D and 3D modalities that are combined to deliver models to clients under various conditions. The content conversion may be done either on-line, upon selection of a desired modality or off-line, at model creation time.

5.1 The Performance Model

We define a set of performance parameters that allow various modalities to be compared in terms of the resources they require and the values they offer. If m denotes a modality associated with a model component, the following performance parameters are considered:

1. $T(m)$: the total estimated time to deliver m to a client,
2. $Q(m)$: the estimated quality associated with rendering m ,
3. $I(m)$: the degree of interaction supported by m .

The *estimated delivery time* T is defined as the sum of estimates of the time T_g it takes the transcoding engine to generate a modality, the time T_t to transfer it over the network, and the time T_r to render it for the first time on the client: $T(m) = T_g(m) + T_t(m) + T_r(m)$. For modalities generated off-line and cached with the model, the generation time is equivalent to the time to retrieve them from the database, which we approximate with a constant. For modalities generated on the fly, the transcoding time is estimated based on information about the rendering capabilities of

the server (if the generation involves rendering on the part of the server) and/or the worst-case complexity of the transcoding method as a function of modality size.

The *estimated quality* Q reflects how closely the rendering of a model component using a particular modality resembles the rendering of the full-detail data. In general, it is difficult to find a common measure of fidelity to be used for a variety of 2D and 3D modalities without actually rendering them and comparing the resulting images. We define quality as a dimensionless number between 0.0 and 1.0 that is modality specific. For instance, the modality corresponding to the full-detail representation of a model component has a quality of 1.0; the quality of a level-of-detail representation is expressed as a percentage of the number of vertices in a level with respect to the number of vertices in the full-detail representation; the quality of a depth image can be 1.0 if the image is rendered at the same (or higher) resolution as on the client, or proportionally less, otherwise.

The *degree of interaction* I represents the number of degrees of freedom when interacting with a particular modality. It can assume values between 0 and 7 for the three principal axes of rotation, translation, and the field-of-view. Geometric modalities typically have all degrees of freedom, whereas image-based representations usually allow for restricted forms of interaction only (e.g., panoramas typically allow rotation about a point, but no translation).

5.2 The Selection Process

The information provided in the meta-data describes the basic characteristics of a model and it is customized according to the capabilities of the client. In this section we focus on adaptive delivery of 3D models to clients with varying degrees of support for 3D rendering (either in software or in hardware) for which the main challenge is to determine an optimal mixture of modalities to represent a model. Clients that can display only text or 2D images and text are not discussed in this paper.

For 3D capable clients, the meta-data information includes the model structure (e.g., relationships between components or a model hierarchy), bounding box information for each of the model components, and the number, types, and characteristics of all modalities available for each component. The characteristics of a modality are the data necessary to evaluate the performance parameters T , Q , and I for that modality and include size (e.g., number of geometric primitives or pixel dimensions), and number of degrees of freedom supported.

5.2.1 Estimating Perceptual Importance

Initially, the collection of bounding boxes can be used to display an outline of the requested model (see Figures 3 (a)-(b)) and to allow users on the client side to define camera parameters. Next, to determine the order in which model components should be considered for modality selection and downloading, the visibility and perceptual importance of each component are estimated. The perceptual importance is a measure of the contribution of a component to the perception of a final rendering of the model. Good heuristics to evaluate importance are difficult to design in a way that closely mimics the partition made by a human eye into what is important and what is not. We found that the benefit heuristics proposed in [14] for interactive navigation of large 3D data sets are also suitable for predicting importance in the context of adaptive transmission. In this section we describe a simple

approach to estimate perceptual importance and visibility of a component simultaneously.

Visibility and perceptual importance can be derived from the bounding box information. A simple way of estimating visibility is to perform view-frustum culling at the bounding box level. Perceptual importance can be roughly approximated by the projected area of the bounding box. We use an image-based method to estimate visibility and perceptual importance in one pass. For a given viewing position, the collection of bounding boxes representing a 3D model is rendered into an off-screen buffer (e.g., the back buffer). Each box is rendered using a color that uniquely identifies it (see Figure 3 (c)). The resulting image is processed to compute a histogram of the colors in it. Model components corresponding to the colors found, i.e., those whose bounding boxes are visible from the current viewpoint, are sorted in decreasing order of their histogram levels and are considered by the adaptive selection algorithm in this order. Components left out by this algorithm due to having bounding boxes that are not visible from a given viewpoint are considered to be *context data* and are processed last.

This approach has the advantage that it is fast to compute and can be later augmented to incorporate additional parameters [13] such as distance from the center of the screen (i.e., the focus of attention) to more accurately predict perceptual importance.

5.2.2 Automatic Selection

User preferences and/or information provided by the environment monitor and meta-data determine the time budget T_B available for transmission and rendering. Starting with the component with the highest importance value, the selection algorithm proceeds to identify the most suitable modality for a component C , as follows. The performance parameters T , Q , and I are evaluated for all modalities available for C , based upon the characteristics of C defined in its meta-data. Among the modalities with $T \leq T_B$, the one with the highest quality Q is selected. If several modalities have the highest quality, the one that supports the highest degree of interaction I is chosen. This type of selection assumes that timely delivery of the model data is most important to the client, followed by the quality of the modalities received, and lastly by the degree of interaction they offer. However, depending on the application, alternative prioritization schemes may be more suitable. For instance, clients may be willing to concede on the waiting time, as long as the quality level of all components received is larger than a threshold value Q_B . In this case, modalities are first selected based on the quality level they provide, and if several modalities offer the same quality they are differentiated based on their associated delivery times, and lastly, based on their corresponding degrees of interaction. In total six prioritization schemes corresponding to the permutations of T , Q , and I are considered.

6. IMPLEMENTATION DETAILS

The design ideas described in this paper have been incorporated into an Adaptive Rendering and Transmission Environment (ARTE) that supports adaptive downloading of 3D models between a server and multiple clients. For transmission and storage purposes, 3D data is encoded in compressed format using the Topological Surgery compression scheme described in [10]. The modalities currently available for representing each model component are the indexed face set polygonal representation and

the depth image, which can be delivered at various resolutions. These modalities determine an implicit partitioning of model components for hybrid rendering on the clients.

Listening Thread

While connection is active **do**

Listen for incoming data

If new packet has arrived **then**

Decode packet header and extract packet info

Copy payload into frame in the decoder buffer

If frame is full **then**

Place frame into decoder queue

Resume decoder thread

End if

End if

End do

Decoder Thread

While connection is active **do**

If decoder queue is empty **then**

Suspend decoder thread

Else

Decode frame at the front of the queue

Send message to rendering thread to update display

End if

End do

Figure 4. The steps involved in the processing of the data packets from their receipt by the client to the display of their contents.

6.1 The Communication Architecture

Both the client and the server are designed as multithreaded applications, in which communication, rendering, and modality generation/decoding are performed in separate threads. An application-level protocol supports efficient data transfer of various modalities between servers and clients. This protocol enables on-demand transmission of 2D and 3D data and allows for a choice of the delivery channel (UDP or TCP). Since modalities generated by the transcoding engine may be view-dependent, the protocol also supports transmission of control information (e.g., viewing camera parameters) over a back channel from clients to servers.

Content to be transferred from a server to a client is partitioned into variable-size *packets*. In ARTE the size of a packet is tailored to semantic units of content (e.g., the compressed geometry of a single model component or a fixed number of “lines” in a depth image). As packets are received by a client, they are grouped into *decoder frames*, for efficient decoding. A *frame* defines an atomic unit that can be individually decoded by a decoder on the client. As soon as all packets in a frame are received, the frame is placed in a *decoder queue* waiting for its turn to be decoded. If the packets are delivered via an unreliable transport protocol such as

UDP, it may happen that not all packets in a frame are received, in which case the entire frame is omitted from decoding, and ultimately, from being rendered. The processing of the packets from their receipt by the client to their display is described in pseudocode in Figure 4.

Client	Model	OS	CPU (MHz)	Mem (MB)	Display adapter
#1	ThinkPad 770 ED	Win 95	266	96	Trident Cyber 9397
#2	ThinkPad 600 E	Win 98	300	192	NeoMagic Magic Media
#3	Intellistation M Pro	Win NT	400	256	Intense 3D Pro 3400-T
#4	Intellistation Z Pro	Win NT	400	1024	Intense 3D WildCat 4000
#5	Running on the same machine as the server				

Table 1. Platform configurations used in the adaptive experiment reported in Table 1.

6.2 Selection of Model Components

ARTE supports two modes for the selection of components to be transmitted as geometric data: *user-defined* and *automatic*. In user-defined mode, the user selects specific components from a model hierarchy, based on meta-data information. Alternatively, in automatic mode, the system derives the components to be downloaded and the resolution of the modalities based on the current view position, the resource budget, the estimated resources available, and the priorities assigned by the user to performance parameters, as described in Section 5.2.2. The remaining components (i.e., those that are not selected for downloading) are rendered on-demand into a depth image on the server, which is subsequently delivered to the requesting client as context. Figure 3 shows a hybrid-rendering example in which model components were selected automatically and downloaded in decreasing order of their projected screen size.

We experimented with two implementations of the component importance estimation algorithm presented in section 5.2.1. Using the first implementation, the entire collection of bounding boxes representing a model is rendered off-screen for every view position. This approach ensures that only components with bounding boxes visible from that position are considered. However, due to the error with which a box approximates the actual component, visible components may be omitted, as shown in Figure 3 (f). Our second implementation uses a “layered” approach in which only bounding boxes corresponding to components that have not yet been downloaded are rendered off-screen. While this method guarantees that, if the budget allows it, all components are eventually downloaded as the model is manipulated, large components occluded by already downloaded data may be considered for processing sooner than small, unoccluded components (see Figure 5). In practice, if the viewing

position is modified frequently, small changes in the camera parameters typically reveal the larger components, and their out-of-order delivery does not impact negatively on the final rendering.

6.3 Experimental Results

Data pertaining to content adaptation is shown in Figures 7 and 8 for a configuration consisting of a server and five clients. The clients made requests for two models of different complexities: an engine with 218 components (70275 vertices) and an automobile with 48 components (12733 vertices). The server was installed on an Intellistation Z Pro, running Windows NT 4.0, with a dual Pentium II 450 MHz processor, 1.2 GB RAM, and an Intergraph Intense 3D Pro 3400-GT graphics adapter. The clients were connected to this server over combinations of 16Mbps Token Ring, 10Mbps Ethernet, and 100Mbps Ethernet connections. Table 1 summarizes the configurations of the five clients. The benchmark measurements reflecting the rendering and network performance of each of these clients are shown in Figure 6.

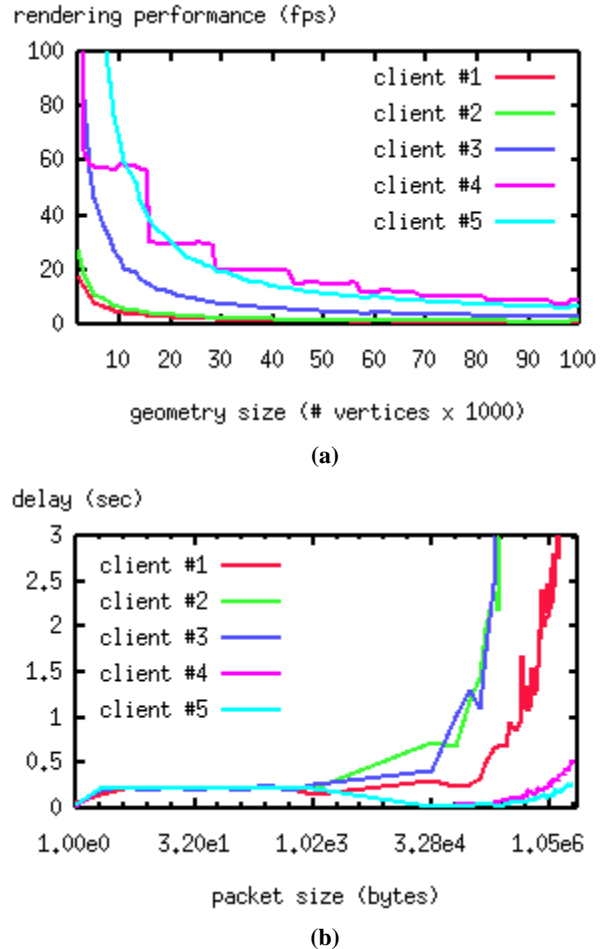
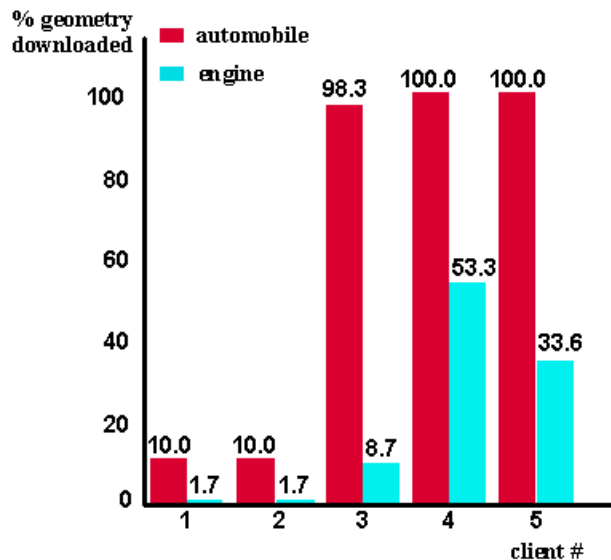
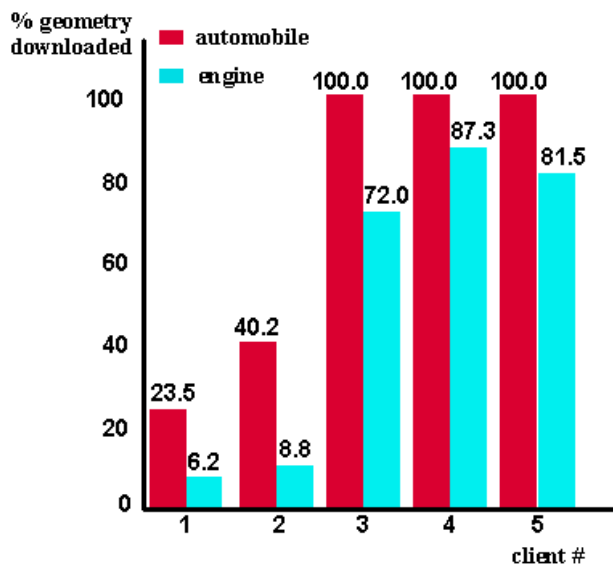


Figure 6. Snapshot of the benchmark measurements recorded for the five clients listed in Table 1. (a) Rendering performance measured for smooth shading, one light, and no triangle strips. (b) Network delay (shown on a logarithmic scale).



(a)



(b)

Figure 7. Results of adaptive delivery of 3D content from a server to five clients with different capabilities (see Table 1). (a) The percentage of the total geometry downloaded to the clients for a target frame rate of 15 frames-per-second for two models of different complexity. (b) Same as (a) for a target frame rate of 5 frames-per-second.

7. SUMMARY

In this paper, we addressed issues related to access to 3D graphics content via client devices with varying degrees of support for 3D rendering and different types of connections to servers that manage this content. In particular, we emphasized the importance of integrating existing techniques for transmission and

representation of 3D models into a common framework, aware of the context in which transfer and rendering take place, and capable of selecting the most appropriate combination of modalities to deliver models to clients requesting them.

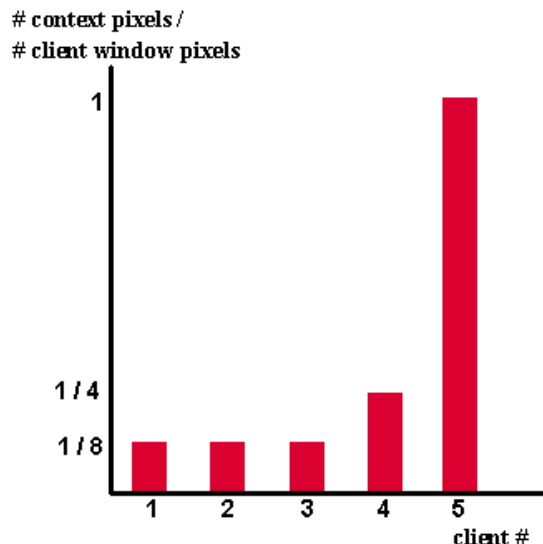


Figure 8. The resolution of the context image received by the five clients listed in Table 1 was adjusted adaptively, given a user-specified transfer budget of 0.3 seconds. The context image was generated on the server by rendering the components of the engine model that were not downloaded to the clients for the scenario shown in Figure 6(a).

We presented the design principles used in the development of ARTE, an adaptive environment that attempts to provide increased quality of service by monitoring the resources available and by automatically selecting among multiple modalities associated with model components.

ARTE is a proof of concept and a prototype under implementation. Future work will include enhancing the accuracy of the monitoring tool to provide better input for the performance estimators and developing better quality measures. We plan to generalize our current definition of the degree of interaction associated with a modality to reflect more general actions (e.g., making a component transparent) and to support editing operations. We also plan to add transcoding modules to enable selection from a larger number of modalities. This will allow us to experiment with the order in which modalities are considered for selection (e.g., delivering the “best” modality versus delivering the “first” modality that satisfies the requirements).

8. ACKNOWLEDGMENTS

I am grateful to Bill Horn for our discussions and his support of this project. I owe special thanks to Jim Klosowski for his contribution to early versions of the code, to Gabriel Taubin for providing the geometric compression library, to Holly Rushmeier and Fausto Bernardini for reviewing the paper, to Claudio Silva for pointing out valuable references, and to Bengt-Olaf Schneider for steering my interests into this area of research. I would also like to thank all of my colleagues in the Visual Technologies Department at IBM T. J. Watson for their helpful comments and suggestions during presentations of my prototype.

9. REFERENCES

- [1] Chandra, S., Schlatter-Ellis, C., and Vahdat, A., "Multimedia Web services for mobile clients using quality aware transcoding", in *Proceedings of the 2nd ACM International Workshop on Wireless Mobile Multimedia*, 99-108, 1999.
- [2] Fox, A., and Brewer, E.A., "Reducing WWW latency and bandwidth requirements by real-time distillation", in *Proceedings of 5th International WWW Conference*, Paris, France, 1996.
- [3] Fox, A., Gribble, S.D., Brewer, E.A., and Amir, E., "Adapting to network and client variation using infrastructural proxies: lessons and perspectives", *IEEE Personal Communications*, 40, 10-19, 1998.
- [4] Mohan, R., Smith, J. R., and Li, C.-S., "Adapting multimedia Internet content for universal access", *IEEE Transactions on Multimedia*, 1(1), 1999.
- [5] Singhal, S., and Zyda, M., "Networked virtual environments", Addison-Wesley, 1999.
- [6] Funkhouser, T.A., "RING: A client-server system for multi-user virtual environments", in *Proceedings of 1995 Symposium on Interactive 3D Graphics*, 85-209, 1995.
- [7] Brutzman, D., Zyda, M., Watsen, K., and Macedonia, M., "Virtual reality transfer protocol (vrtp) design rationale", in *Workshops on Enabling Technology: Infrastructure for Collaborative Enterprises (WET ICE): Sharing a Distributed Virtual Reality*, MIT 1997 (http://www.web3d.org/WorkingGroups/vrtp/docs/vrtp_design.pdf).
- [8] Rhyne, T.-M., Barton, B., Brutzman, D., and Macedonia, M., "Internetworked 3D computer graphics: overcoming bottlenecks and supporting collaboration", in *ACM SIGGRAPH'99 Course Notes*, course # 21, 1999.
- [9] Deering, M.F., "Geometry compression", in *Proceedings of ACM SIGGRAPH'95*, 13-20, 1995.
- [10] Taubin, G., and Rossignac, J., "Geometric compression through topological surgery", in *ACM Transactions on Graphics*, 17(2), 84-115, 1998.
- [11] Gumhold, S., and Strasser, W., "Real-time compression of triangle mesh connectivity", in *Proceedings of ACM SIGGRAPH'98*, 133-140, 1998.
- [12] Cignoni, P., Montani, C., Scopigno, R., "A comparison of mesh simplification algorithms", in *Computers and Graphics*, 22, 37-54, 1998.
- [13] Funkhouser, T.A., and Sequin, C.H., "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments", in *Proceedings of ACM SIGGRAPH'93*, 247-254, 1993.
- [14] Maciel, P.W.C., and Shirley, P., "Visual navigation of large environments using textured clusters", in *Proceedings of 1995 Symposium on Interactive 3D Graphics*, 95-102, 1995.
- [15] Hoppe, H., "Progressive meshes", in *Proceedings of ACM SIGGRAPH'96*, 99-108, 1996.
- [16] Hoppe, H., "View-dependent refinement of progressive meshes", in *Proceedings of ACM SIGGRAPH'97*, 189-198, 1997.
- [17] Gueziec, A., Taubin, G., Lazarus, F., and Horn, W., "Simplicial maps for progressive transmission of polygonal surfaces", in *Proceedings of ACM VRML'98*, 1998.
- [18] Chen, S.E., "QuickTime VR – an image-based approach to virtual environment navigation", in *Proceedings of ACM SIGGRAPH'95*, 29-38, 1995.
- [19] Aliaga, D.G., "Visualization of complex models using dynamic texture-based simplification", in *Proceedings of IEEE Visualization'96*, 101-106, 1996.
- [20] Shade, J., Lischinski, D., Salein, H., DeRose, T., and Snyder, J., "Hierarchical image caching for accelerated walkthroughs of complex environments", in *Proceedings of ACM Siggraph'96*, 75-82, 1996.
- [21] Shade, J., Gortler, S., He, L., and Szeliski, R., "Layered depth images", in *Proceedings of ACM SIGGRAPH'98*, 231-242, 1998.
- [22] Aliaga, D.G., Lastra, A., "Automatic image placement to provide a guaranteed frame rate", in *Proceedings of ACM SIGGRAPH'99*, 307-316, 1999.
- [23] Levoy, M., "Polygon-assisted JPEG and MPEG compression of synthetic images", in *Proceedings of ACM SIGGRAPH'95*, 21-28, 1995.
- [24] Mann, Y., and Cohen-Or, D., "Selective pixel transmission for navigating in remote virtual environments", in *Computer Graphics Forum (proceedings of Eurographics'97)*, 16(3), 201-206, 1997.
- [25] Schneider, B.-O., and Martin, I., "An adaptive framework for 3D graphics over networks", in *Computers and Graphics*, 23, 867-874, 1999.
- [26] Peterson, L.L. and Davie, B.S., "Computer networks: a systems approach", Morgan Kaufmann, 2000.
- [27] Snell, Q.O., Mikler, A.R., and Gustafson, J.I., "NetPIPE: A network protocol independent performance evaluator", in *Proceedings of the International Conference on Intelligent Information Management Systems*, 1996.

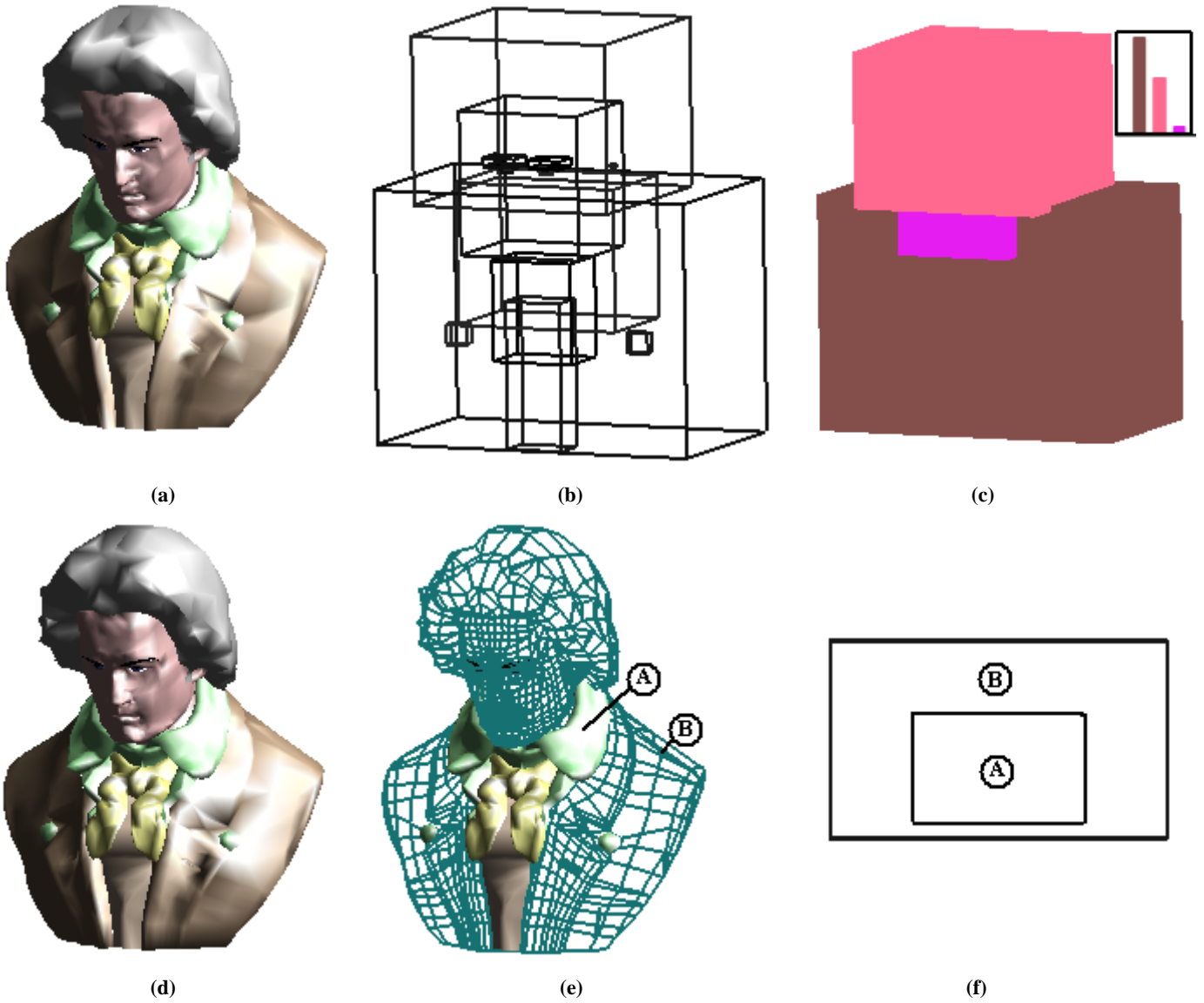


Figure 3. Example of adaptive selection of components to be downloaded to a client based on estimated perceptual importance and visibility from the current viewing position. (a) Original model. (b) Meta-data information includes a collection of bounding boxes that can be manipulated in 3D to select a viewing position. (c) Once a view is selected, the bounding boxes of the components are rendered into an off-screen buffer. The histogram levels of the resulting image (shown in the upper right corner) determine the components that will be downloaded and the order in which they are processed. In this example, three components are downloaded. (d) Hybrid rendering on the client combining the components downloaded with a depth image generated on the server. (e) Same as (d), except the three components downloaded are shown as wireframe. (f) Top view of the bounding boxes corresponding to components A and B. Using a non-layered approach, component A is never considered by the selection algorithm due to occlusion by the larger box corresponding to component B.

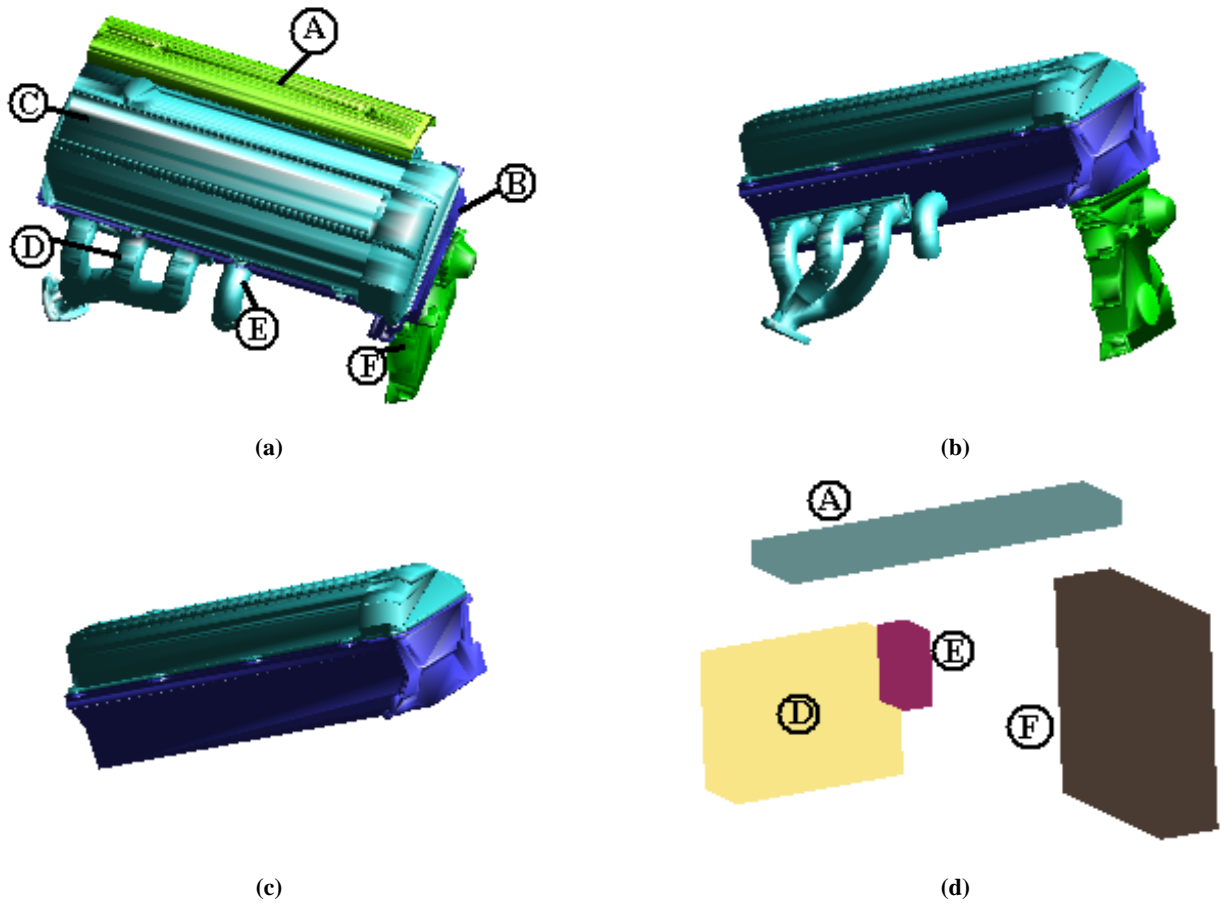


Figure 5. Estimation of visibility and perceptual importance using a layered approach: (a) portion of an engine model. (b) The same model, viewed from a position in which component A is completely occluded. (c) Components B and C have already been downloaded to the client (e.g., by user selection). (d) The order in which the remaining components are processed is determined by the histogram of the image shown. By “peeling off” the layer of components already on the client, the bounding box corresponding to component A becomes unoccluded and since its projected area is larger than that of component E, A will be processed before E.