Visualization Viewpoints

Editor: Theresa-Marie Rhyne

Adaptive Graphics

Ioana M. Boier-Martin

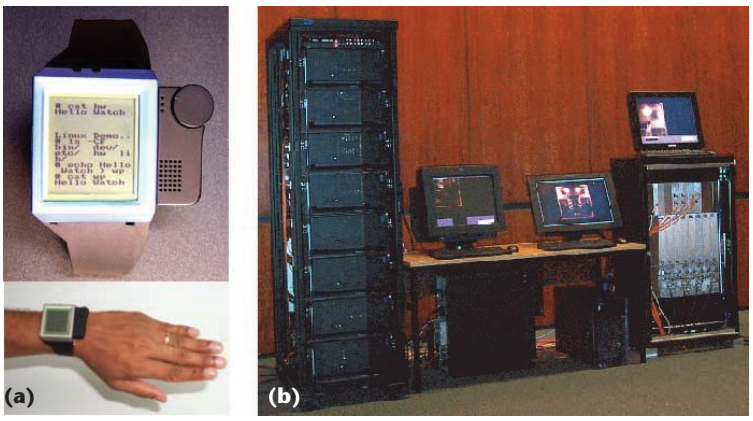
IBM T.J. Watson Research Center

The old man tallies up his colored beads;
He fits a blue one here, a white one there,
Makes sure a large one, or a small, precedes,
And shapes his Game ring with devoted care.
Hermann Hesse, *The Glass Bead Game*

In 1959, in a talk to the American Physical Society, Richard Feynman talked about amassing the contents of Encyclopedia Britannica on the surface of a pin head. Inspired by the ability of biological systems to encode information on a small scale, the Nobel laureate described his vision of a miniature, atom-size computer with the processing ability of the human brain.[1] In 1998, in his capstone address at the IEEE Visualization conference, Turner Whitted envisioned projecting information on large surfaces. Inspired by artistic drawings on cave walls, he advanced the idea of large-format computer displays that surround us and facilitate collaboration.[2] Today we are in the privileged position of witnessing these visionary predictions come true. Although there is still "plenty of room at the bottom," as Feynman once said, and not every man-made surface is a display yet, the recent explosion of computing devices large and small is proof that good progress is under way toward both ends of the scale spectrum.

When small, large, and everything in between coexist in the same networked environment, we are faced with the challenge of providing customized access to information, depending on the resources available. While the hardware technology that makes feasible creating small computers or driving large displays has known considerable advances in recent years, software applications have remained for the most part rather generic and insensitive to device diversity. Figure 1a shows a prototype Linux-based wristwatch computer featuring a 320 × 240 dot monochrome VGA display hardly bigger than a postal stamp, a wireless link, and 8 Mbytes of RAM.[3] Figure 1b shows a 3840 × 2400 pixel display connected to a network-attached frame buffer capable of double buffering up to 16 million pixels and a cluster of eight workstations each featuring dual 866-MHz processors, 1 Gbyte of RAM, and high-speed Ethernet connections.[4] Imagine both of these devices attempting to access and visualize stock market data over the network in real time. While the brute-force approach of downloading all the data and rendering it locally might arguably work for the cluster, it is clearly inappropriate for the wristwatch. In fact, given the sheer volume of stock market data and its dynamic nature, both devices could benefit from an intelligent selection mechanism that accounts for the needs of the application running on each device and its resources.

This article presents the idea of a unifying framework that allows visual representations of information to be customized and mixed together into new ones. The net result is a fine-grained approach to representing data, better suited to accessing and rendering it over networks. Although my focus is on geometric models and 3D shape representations, many issues I discuss here are relevant to network-based visualization in general.



1 (a) Wristwatch computer weighing 44 grams and featuring a monochrome 320 × 240 VGA display and wireless connectivity. (Image courtesy of Chandra Narayanaswami.) (b) DeepView system including a cluster of Linux workstations (left), a high-resolution T221 display (middle-right), and a Scalable Graphics Engine frame buffer (right). (Image courtesy of James Klosowski.) Both prototypes have been developed at IBM Research.

Published by the IEEE Computer Society

Graphics in a networked world

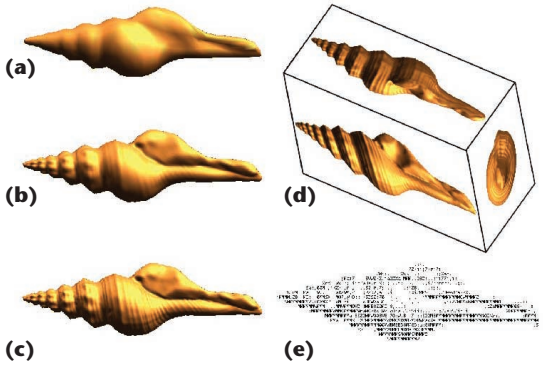
An increasing number of applications are beginning to use 3D graphics to provide visual information. In addition to games and industrial design where 3D has been the norm for a while, we have begun to notice its presence in domains like e-commerce, medicine, education, and the arts. "Will that suit make me look fat? Can I see the pianist's hands from section D of the concert hall?" Surprisingly, we can find answers to questions like these on a number of Web sites offering interactive 3D technologies. In the automotive and aerospace industries, entire manufacturing processes are being digitally planned with the help of 3D. In medicine, devices used to diagnose organs such as the heart and liver are shifting from 2D imaging to 3D reconstructions. People are able to look at 3D models of their babies' smiling faces before they are born. And on stage, virtual dancers envelop live performers creating surreal mixtures of picture planes and clear space.

Yet, 3D is not a mainstream medium. Companies trying to make a business out of providing 3D over the Internet have either gone out of business or are gaining little traction in today's market. Part of the reason is that creating 3D content remains rather complex. On any given day, I can capture hundreds of pictures and video clips with my digital camera I bought for less than $300. Three-dimensional cameras are not quite there yet. Portable shape capture devices have become more affordable in recent years, however, their interfaces still need work. Even after the content creation problem is solved, a number of challenging issues remain to be addressed. Key ones such as network delivery and rendering are the subject of this article.

While in some instances 3D data may be locally available, in many cases, it is stored at remote locations from where it has to be retrieved by applications. Due to the increasing diversity of client devices, efficient transfer technologies must support access to appropriate representations of 3D models, enabling clients to view these models regardless of their graphics capabilities and network connections. So far, the predominant form of "adaptivity" has been progressive transmission and streaming. Although this may be suitable in certain scenarios, it does not always work. If I am looking to buy a car, for example, and I cannot afford to download and render the corresponding data on my computer, then I would prefer to inspect a high-resolution image-based impostor (for example, a billboard cloud[5]) to a severely and often arbitrarily decimated version of the original mesh. However, accommodating multiple representations of the same model and selecting optimal ones with respect to given constraints is not trivial. Bandwidth (or lack thereof), interactivity, scalability, error resilience, and security are concerns that have to be considered in order to bring 3D to the mass market.

Motivations

An extensive number of optimization methods for both transmission and rendering of 3D models have been developed. Such methods include 3D model compression for reduced storage requirements and faster delivery, streaming techniques for progressively downloading data, model simplification and level-of-detail management for improved graphics performance, as well as various image-based methods that replace all or parts of a model with images, thus trading freedom of interaction for reduced geometric complexity. Different styles of rendering have also been pursued, from traditional shading to photorealistic approaches, to the abstract look of nonphotorealism.



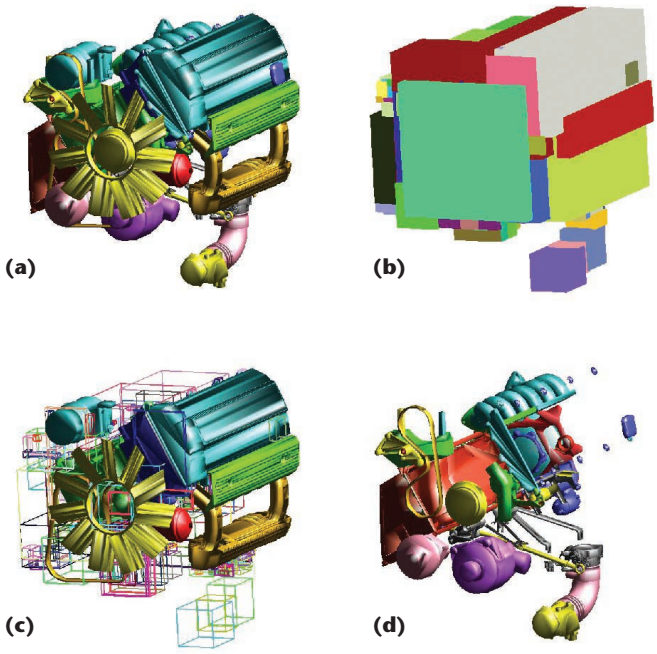
2 Different representations of a 3D shell model: (a)–(c) The shell is represented as a multiresolution subdivision surface; the meshes are progressively finer (from top to bottom) and correspond to three different levels of detail in the multiresolution hierarchy. (d) A box textured with six views of the model along the principal axes. (e) ASCII art representation of the shell (2D).

In itself, each method makes for a one-size-fits-all strategy that works well for certain models and system configurations and is less applicable and efficient for others. The idea of adaptive graphics is to include such methods into a single system that allows optimal combinations to be selected and applied, depending on the specifics of each application. In multimedia jargon, the process of converting between different representations to adapt to different client capabilities is known as *transcoding*. I maintain that using transcoders for 3D data considerably improves the quality and efficiency of its transmission in heterogeneous environments. Successful transcoding, however, is hardly straightforward. Once clients have been differentiated based on their capabilities, the challenge lies in extracting the most meaningful part of the data for those clients that cannot afford to receive it all. Along these lines, text and video summarization have been the topic of intense research in recent years. In contrast, work on 3D shape summarization is barely beginning to emerge.[6]

In my earlier work,[7] I described the design and implementation of an adaptive rendering and transmission environment (ARTE), which I view as a first step toward addressing some of these issues. Here, I synthesize the ideas and lessons learned during the development of ARTE, and I present my vision for future exploration on this topic.

Adaptively transmitting 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. Figure 2 shows the data cor-



3 **Hybrid rendering of an engine model adaptively downloaded to a client. (a) The model data received and rendered by the client. It consists of a combination of geometry and a depth image. (b) The importance value of each component in the view shown is estimated based on metadata (in this case, the projected size of its bounding box). (c) For the most important components, full-resolution geometry is downloaded. A rendering budget of 15 frames per second was specified by the user. (d) The remaining components were rendered on the server into a depth image and sent to the client as context data to create the image in Figure 3a.**

responding to a 3D shell model being represented and rendered using different modalities. Depending on the resources available, either one of these representations may be used to deliver the model to a requesting client. A high-end PC with a fast network connection would likely be able to receive the entire data set and render it locally at full resolution; a personal digital assistant with wireless access may receive the lower resolution data or just the textured box; finally, a cell phone with a text-only display would get just the ASCII drawing.

Combining representations. In the previous example (Figure 2), it suffices to select a single representation to deliver the model to a client. For more complex data sets, it becomes useful to partition them into several components and to allow different representations for each component. Thus, the data received by a client requesting a complex model may consist of a combination of representations with different characteristics. The engine model in Figure 3 has 208 components. The final rendering in Figure 3a combines geometry rendered on the client (corresponding to the most important components as Figure 3c shows) with a context depth image rendered on the server (see Figure 3d).

Two aspects are important to note here. The first is the partitioning of the model into components. For some models, this is done at creation time and the corresponding information is stored with the models (this is the case, for example, for CAD assemblies). For others, components must be generated on-the-fly, using some form of semantic or spatial partitioning (for example, octrees). Note that some decomposition schemes may lead to data duplication. Depending on the underlying organization, it might be possible to tag and exploit such redundancy to increase performance and improve error resilience during transfer. The second aspect relates to associating an importance value to each model component to characterize its worth to the client. This value may be implicitly defined in terms of the contribution of that component to the final rendering or it may be explicitly stated by the application (for example, "download the aircraft's landing gear at full resolution").

Metadata. When accessing complex models, it is often convenient to preview them before downloading. Such capability is routinely provided by applications with support for image browsing, letting clients view thumbnails before opening full-resolution images. For 3D models, the preview data or metadata plays several roles. In addition to quickly providing basic information about a model, it also helps clients establish selection criteria that are subsequently used to steer the downloading process. Metadata can also serve as calibration data for the monitoring tools. Last but not least, it can be regarded as a form of summarization that provides a minimal representation for use in environments with limited resources.

Creating metadata constitutes an important and difficult problem in itself. Automatically generating it for a given data set is largely a data mining issue. Often times, a mixture of automatic and user-driven metadata creation works best. Examples of metadata might include information about the organization of a model (that is, relationships between various components or model hierarchy), reduced geometric information (for example, bounding boxes or coarse meshes), and representation information concerning the modalities available to deliver the model. Figure 3b illustrates the geometric information included in the metadata for the engine model. In this case, the metadata consists of a collection of component bounding boxes. As explained further on, the bounding-box information is subsequently used to prioritize components during the downloading of the model.

Resource monitoring. Tools are necessary to keep track of the resources available. Their role is to feed quantitative information to the adaptive selection process by recording the events that occur in the environment. The choices of resources to be monitored in

complex networked setups are multiple. We can focus our attention on four general state parameters:

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

Two alternatives for measuring values of these parameters become obvious. The first is to map values of performance counters available at the operating system level (for example, CPU and memory usage, I/O rates, and network behavior) to values corresponding to the four state parameters. The main drawbacks of this approach are having to design suitable mappings and that the set of performance counters varies between platforms. The second option is to record measurements that are easier to interpret from an application's perspective. For instance, the average frame rate (in frames-per-second) could be used to characterize the rendering capabilities of a machine (client or server); the load on the server could be estimated in terms of the average time between the receipt of a request by the server and the processing of that request; lastly, network performance could be quantified as a function of latency and bandwidth. In ARTE, I have experimented with the latter approach by maintaining history information for each parameter. In this case, I chose to record measurements on the models being transferred and rendered into the history and to use this data to predict the resource budget for future requests.

Differentiated service. The main challenge of adaptive delivery is determining and generating optimal combinations of representations for different client platforms. One approach is to prioritize model components based on their importance and to select for each of them the best representation that satisfies the resource budget constraints. Typically, a constraint is a limit on the time available for transmission and rendering. However, quality and interaction requirements may also have to be considered, depending on the application.

Once a priority scheme is in place, the selection proceeds to identify a suitable representation for components in decreasing order of their importance. The difficulty lies in devising good heuristics that capture the semantics of "most important component" and "best representation."

Heuristics for selection. When the importance value of a component is not explicitly specified, it has to be inferred. In a purely visual system like the one considered in this article, an obvious approach is to define importance as visual contribution in the context of the final rendering. Perceptual experiments have shown that it is difficult to design good heuristics in a way that closely mimics the partition made by a human eye into what is important and what is not. In the case of 3D graphics, such heuristics have been proposed for interactive navigation of large data sets.[8,9] In ARTE, I have shown that they are also applicable to predicting importance for adaptive transmission.

Figure 3b illustrates a simple example. In this case, importance is proportional to a component's projected screen size. An approximate projected area is derived based on the bounding-box information in the metadata. For a given viewing direction, the collection of bounding boxes is rendered using a unique color for each component. The histogram of the image thus generated gives us a quick scheme for prioritizing components according to their screen size. The larger components are downloaded first (see Figure 3c), and the ones that do not fit into the transmission budget are rendered on the server and sent to the client in the form of a context image (see Figure 3d).

As I previously mentioned, performance metrics are not only needed for importance estimation but also to compare representations of the same component. I have identified three main characteristics of a representation:

- the estimated time T it would take to deliver it from the server to the client (including time to generate it, if not cached);
- its quality Q, which defines how closely a rendering of this representation resembles the rendering of the full-resolution data; and
- the type of interaction I it supports.

For example, a coarse mesh representation offers all degrees of freedom for interaction, however, its quality depends on how much geometry was removed during simplification. In contrast, images rendered on the server at the client resolution have high quality but cannot be interactively manipulated in 3D.

During selection, for a given component C, the performance parameters T, Q, and I are evaluated for each of the representations available. Among the representations with T less than the budget, the one with the highest quality Q is selected. If several representations have the highest quality, the one that supports the highest degree of interaction I is chosen.

Support for interaction. When multiple representations are combined as described to render a model, some of them may be view-independent whereas others may be generated with respect to a particular viewing position. In an interactive client session, the viewpoint may change often, requiring the view-dependent representations to be updated accordingly. Nevertheless, generating all these representations at every frame is typically impractical. Alternatives to this brute-force approach include

- displaying only the view-independent representations as the object moves and requesting the view-dependent ones to be generated only for the model's final position after the motion has stopped;
- using view-dependent data previously downloaded to synthesize new views;
- using a dead reckoning technique[10] to predict the model's position at the time the representation would arrive to the client (when the viewpoint changes according to a known trajectory).

A related and more difficult problem is supporting dynamically changing content typical of real-time visu-

alizations, simulations, and network games. In this case, we must weigh methods of locally updating a previously downloaded representation against generating a completely new one.

Practical considerations. Implementing an adaptive system for delivering 3D models over networks is not trivial. In addition to the points I have already discussed, several aspects outside the graphics domain are pertinent. At the networking level, error resilience is important if lossy transmission protocols like User Datagram Protocol (UDP) are used. Scalability is a concern for applications with a potentially large number of users. Some of the work done by the server might have to be delegated to proxies. Security is paramount to the integration of the adaptive system into the real world, as it constitutes a major concern for all networked graphics applications. Tasks like encryption and watermarking might have to be factored into the performance model. Finally, database management issues such as search, retrieval, and access to 3D models must be addressed.

Conclusions

Most of the ideas in this article emerged during the implementation of the ARTE system. I have presented them in this article with the goals of emphasizing the advantages of adaptive delivery of 3D models and pointing out some of the challenges involved. Although our experiments have led us to solutions to some of the problems, many issues remain open. For example, in the scenario described, representations are chosen and transmitted based on their performance characteristics and the budget available. However, each component is independently considered, which may lead to combinations of different representations in a single image. Combining a depth image with geometry is relatively pleasing to the eye, especially if the resolution of the image is good. However, other combinations (for example, a textured box with coarse geometry) may cause visual discontinuities that degrade the quality of the final image. Hence, a more elaborate selection scheme including a more sophisticated perceptual model (similar to the one proposed in Horvitz and Lengyel[11]) that accounts for intercomponent dependencies and methods for seamless integration of different representations is necessary.

A related issue is augmenting 3D scenes with other perceptual cues that can help convey shape.[12,13] The main question is determining if such an enriched representation balances out the need for identifying additional resources (for example, a tactile interface may be present on some clients and not on others).

The problem of 3D summarization is largely unsolved. Summarization of geometric information barely scratches the surface of what is really needed. Models typically come with more than geometric information. Textures and vector-field attributes commonly accompany 3D data. Any successful summarization technique would have to consider these as well. More complex yet is the extension of summarization and the other aspects discussed to visualizations of scientific and business data. Extracting the meaningful core from terabytes of data and determining the best ways to represent it constitutes a research subject all by itself.

With increasing bandwidth, some of the issues in this article may be alleviated but not entirely solved. At any given moment, more information is generated than any computer can handle. Intelligent transcoders will be necessary to exchange it over networks, visualize it, and ultimately, make sense of it. ■

Acknowledgments

I thank the anonymous reviewers for their valuable and constructive suggestions, Jim Klosowski and Chandra Narayanaswami for providing images of their prototype systems, and Holly Rushmeier for proofreading the article.

References

1. R.P. Feynman, "There's Plenty of Room at the Bottom," *Proc. Ann. Meeting American Physical Society*, Engineering and Science (California Institute of Technology), 1960, pp. 22 (also at http://www.zyvex.com/nanotech/feynman.html).
2. T. Whitted, "Draw on the Wall," *IEEE Computer Graphics and Applications*, vol. 19, no. 4, July/Aug. 1999, pp. 6-9.
3. C. Narayanaswami and M.T. Raghunath, "Application Design for a Smart Watch with a High-Resolution Display," *Proc. Symp. Wearable Computers* (ISWC'00), IEEE CS Press, 2000, pp. 7-14 (also at http://www.research.ibm.com/WearableComputing/).
4. J.T. Klosowski et al., "Deep View: High-Resolution Virtual Reality," *IEEE Computer Graphics and Applications*, vol. 22, no. 3, May/June 2002, pp. 12-15 (also at http://www.research.ibm.com/deepview/index.html).
5. X. Decoret et al., *Billboard Clouds*, Rapport de Recherche 4485, Inria Rhones-Alpes, 2002.
6. I. Herman and D. Duke, "Minimal Graphics," *IEEE Computer Graphics and Applications*, vol. 21, no. 6, Nov./Dec. 2001, pp. 18-21.
7. I.M. Martin, "Hybrid Transcoding for Adaptive Transmission of 3D Content," *Proc. IEEE Int'l Conf. Multimedia and Expo* (ICME), IEEE CS Press, 2002, pp. 373-376.
8. T.A. Funkhouser and C.H. Sequin, "Adaptive Display Algorithm for Interactive Frame Rates During Visualization of Complex Virtual Environments," *Computer Graphics* (Proc. Siggraph 93), ACM Press, 1993, pp. 247-254.
9. P.W.C. Maciel and P. Shirley, "Visual Navigation of Large Environments Using Textured Clusters," *Proc. Symp. Interactive 3D Graphics*, ACM Press, 1995, pp. 95-102.
10. S. Singhal and M. Zyda, *Networked Virtual Environments*, Addison-Wesley, 1999.
11. E. Horvitz and J. Lengyel, "Perception, Attention, and Resources: A Decision-Theoretic Approach to Graphics Rendering," *Proc. 13th Conf. Uncertainty in Artificial Intelligence*, Morgan-Kaufmann, 1997, pp. 238-249.
12. A.J. Hollander, *An Exploration of Virtual Auditory Shape Perception*, masters thesis, Univ. of Washington, 1994.
13. D.K. Pai et al., "Scanning Physical Interaction Behavior of 3D Objects," *Computer Graphics* (Proc. Siggraph 2001), ACM Press, 2001, pp. 87-96.

Readers may contact Ioana M. Boier-Martin at IBM T.J. Watson Research Center, Hawthorne, NY 10532, email ioana@us.ibm.com.