Parallel and Distributed Visualization Advances

Huabing Zhu
National University of Singapore, Singapore

Lizhe Wang
Institute of Scientific Computing, Forschungszentrum Karlsruhe, Germany

Tony K. Y. Chan
Nanyang Technological University, Singapore

INTRODUCTION

Visualization is the process of mapping numerical values into perceptual dimensions and conveying insight into visible phenomena. With the visible phenomena, the human visual system can recognize and interpret complex patterns. One can detect meaning and anomalies in scientific data sets. Another role of visualization is to display new data in order to uncover new knowledge. Hence, visualization has emerged as an important tool widely used in science, medicine, and engineering.

As a consequence of our increased ability to model and measure a wide variety of phenomena, data generated for visualization are far beyond the capability of desktop systems. In the near future, we anticipate collecting data at the rate of terabytes per day from numerous classes of applications. These applications can process a huge size of data, which are produced by more sensitive and accurate instruments, for example, telescopes, microscopes, particle accelerators, and satellites (Foster, Insley, Laszewski, Kesselman, & Thiebaux, 1999). Furthermore, the speed of the generation of data is still increasing.

Therefore, to visualize large data sets, visualization systems impose more requirements on a variety of resources. For most users, it becomes more difficult to address all requirements on a single computing platform, or for that matter, in a single location. In a distributed computing environment, various resources are available, for example, large volume data storage, supercomputers, video equipment, and so on. At the same time, high speed networks and the advent of multi-disciplinary science mean that the use of remote resources becomes both necessary and feasible (Foster et al., 1999).

BACKGROUND

Visualization Process

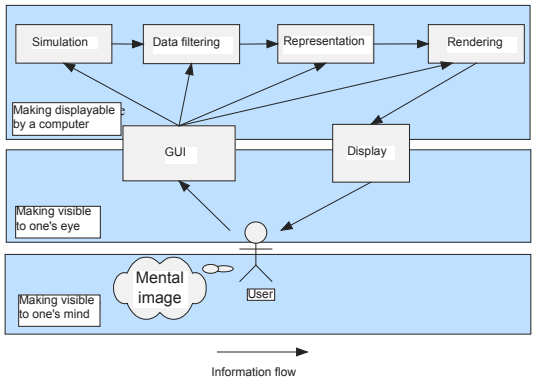
The process of visualization is decided by the choice of representation. The process of visualization with 3-D, real-time interaction is much more complicated than one with still images. No matter how a specific process behaves, it can be considered in three different but interrelated semantic contexts (see Figure 1). They are *making displayable by a computer*, *making visible to one's eyes* and *making visible to one's mind* (Brodlie et al., 2004). This section provides an overview of the process of creating a 3-D, real-time, interactive visualization.

Interactive means that the system responds quickly enough for users to adjust the controlling inputs in rapid response to the output (Mueller, 2001). *Real time* means that the system must always keep updating outputs within a certain small fixed amount of time (Mueller, 2001). Generally, *real time interactive visualization* is defined as the process of creating images at rates between approximately 1 and 100 frames per second. In particular, if the latency exceeds beyond approximately 1 second, humans would feel that the computer's responses is too slow to interact continuously. However, increasing the frame rate from 100 frames per second to 1000 frames per second is of no use due to perceptual characteristics of the human brain (Igehy, 2000).

Figure 1 shows the visualization flow chart of real-time, interactive visualization. In this diagram, information flow follows the arrows. Firstly, data are generated from some simulation systems such as a mathematically based computational model or a collection of observed values. Data filtering involves a wide range of operations, such as removing noise, replacing missing values, and so on, and makes data readable to visualization software. These operations are also used to refine the data by sifting the most relevant aspects and removing unnecessary values.

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

Figure 1. Interactive visualization process



After the data are filtered, the representation procedure maps data to some geometric form. At this point, a geometric scene graph will be setup. The scene graph involves the geometric objects, the color value of the objects, the materials, and so on. These parameters can also be driven by computational models within the constraints imposed by visualization software.

The rendering procedure takes the information of scene graph and computes the 2-D image for human eyes to see. These images will be stored in a color buffer temporal for display. The resultant images will be displayed on computer screens or other output devices, such as project wall, Head Mounted Display (HMD).

Using graphical user interfaces (GUI), users can steer visualization systems. One can adjust the visualization system by modifying variables and data in the simulation, data filtering, representation, and rendering stages and get real-time response. It is important for real-time interaction that both simulations and graphics systems should provide real-time performance and allow for user input.

Towards Distributed Visualization

There are trends to increase not only the complexity of the models that are displayed, but also the resolution of the images. However, users still require that the system must always respond with updated output within a certain small fixed amount of time (Mueller, 2001). Then hundreds of megaFLOPS of performance and memory bandwidth of gigabytes per second are demanded. These requirements are far beyond capabilities of a single desktop system (Molnar, Cox, Ellsworth, & Fuchs, 1994). Therefore, research efforts have been made to build high performance architectures for visualization. Several recent developments have demonstrated how graphics hardware of a PC cluster can accelerate a graphics and visualization task (Muraki, Lum, Ma, Ogata, & Liu, 2003). In order to speed up the development of distributed visualization systems, some toolkits are developed for distributed parallel visualization.

WireGL (Humphreys, Buck, Eldridge, & Hanrahan, 2000; Humphreys et al., 2001) is the first sort-first cluster rendering system. It includes an efficient network protocol, a geometry bucketing scheme, and an OpenGL (http://www.opengl.org) state tracking algorithm. It supports heterogeneous hardware platforms. WireGL divides the computational nodes into clients and rendering servers. It replaces the OpenGL driver on client machines, intercepts the OpenGL calls, and sends the calls over a high-speed network to servers, which render the geometry. WireGL classifies each OpenGL call into one of three categories: (1) geometry, (2) state, or (3) special. Each rendering server receives OpenGL commands from remote clients and renders a portion of the screen space. Chromium (Humphreys et al., 2002), the new version of WireGL, is a stream-oriented framework for processing streams of OpenGL commands on parallel architectures, for example, clusters. It can support sort-first, sort-last, and hybrid parallelization strategies by the use of stream processing units. Some researches, such as VTK and OpenRM (Bethel, Humphreys, Paul, & Brederson, 2003), integrated Chromium with visualization software.

Mesa is an open source software implementation of OpenGL without using a hardware accelerator. PMesa (Mitra & Chiueh, 1998) is a parallel version of Mesa. PMesa is built with sort-last architecture. The geometry data are arbitrarily assigned to rendering processors. Mitra and Chiueh developed a general parallel compositing algorithm, which is integrated with both binary swapping composition (Ma, Painter, Hansen, & Krogh, 1994) and parallel pipeline composition (Lee, Raghavendra, & Nicholas, 1996). According to the number of composition processors, it divides the processors into several groups. Inside one group, it performs parallel pipeline composition. Between groups, it runs binary sweeping composition. In this way, it not only keeps all processors at high utilization efficiency, but also cuts down the step number for composition to optimize the performance.

AnyGL (Yang, Shi, Jin, & Zhang, 2002) is a software implementation of a sort-anywhere architecture.

Sort-Both (Zhu, Chan, Wang, & Jegathese, 2004) is a hybrid architecture which includes both sort-first and sort-last stages in the rendering pipeline. Other systems, such as OpenSG (http://www.opensg.org), Aura (Van der Schaaf, Renambot, Germans, Spoelder, & Bal, 2002), and PGL (Crockett & Orloff, 1993), have achieved a set of valuable results in parallel rendering.

6 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage: www.igi-global.com/chapter/parallel-distributed-visualization-advances/14020

Related Content

The Application of the Theory of Reasoned Action to Senior Management and Strategic Information Systems
Peter P. Mykytyn Jr. and David A. Harrison (1993). *Information Resources Management Journal (pp. 15-26).*
www.irma-international.org/article/application-theory-reasoned-action-senior/50976

Toy or Useful Technology?: The Challenge of Diffusing Telemedicine in Three Boston Hospitals
Hüseyin Tanriverdi and C. Suzanne Iacono (1999). *Success and Pitfalls of Information Technology Management (pp. 1-13).*
www.irma-international.org/article/toy-useful-technology/33475

Semantic Business Process Management: A Case Study
Sebastian Stein, Christian Stamber, Marwane El Kharbili and Pawel Rubach (2010). *Information Resources Management: Concepts, Methodologies, Tools and Applications (pp. 1144-1166).*
www.irma-international.org/chapter/semantic-business-process-management/54536

Particle Swarm Optimization Research Base on Quantum Q-Learning Behavior
Lu Li and Shuyue Wu (2017). *Journal of Information Technology Research (pp. 29-38).*
www.irma-international.org/article/particle-swarm-optimization-research-base-on-quantum-q-learning-behavior/176372

Evidence of Compensatory Adaptation to Unnatural Media in a Field Study of Process Redesign Dyads
Ned Kock (2007). *Emerging Information Resources Management and Technologies (pp. 123-156).*
www.irma-international.org/chapter/evidence-compensatory-adaptation-unnatural-media/10097