Multi-Frame Rate Rendering and Display

Jan P. Springer[1] Stephan Beck[1] Felix Weiszig[1] Dirk Reiners[2] Bernd Froehlich[1]

[1] Bauhaus-Universität Weimar [2] University of Louisiana at Lafayette

Abstract

We introduce a new concept for improved interaction with complex scenes: multi-frame rate rendering and display. Multi-frame rate rendering produces a multi-frame rate display by optically or digitally compositing the results of asynchronously running image generators. Interactive parts of a scene are rendered at the highest possible frame rates while the rest of the scene is rendered at regular frame rates. The composition of image components generated with different update rates may cause certain visual artifacts, which can be partially overcome with our rendering techniques. The results of a user study confirm that multi-frame rate rendering can significantly improve the interaction performance while slight visual artifacts are either not even recognized or gladly tolerated by users. Overall, digital composition shows the most promising results, since it introduces the least artifacts while requiring the transfer of frame buffer content between different image generators.

Keywords: Multi-Frame Rate Rendering, Multi-Frame Rate Display, 3D Interaction, Projection-based Display Systems

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Virtual Reality

1 Introduction

The interactive and high-quality visualization of large models is still a challenging problem even though the capabilities of graphics processing units (GPU) have been dramatically improved over the past years. Unfortunately the expectations on *visual quality* have increased even more, which in general affects the interactivity of applications or *interaction quality*. These two qualities seem to be at opposite ends of a continuum. Visual quality is mainly dependent on the scene complexity (e. g. the number of primitives), the rendering method, the illumination and shading model, and the display resolution. While all of these factors might also improve the interaction quality, they often lead to low frame rates when excellent visual quality is desired. Interaction quality heavily depends on immediately incorporating user actions into the simulation and image generation process which demands high frame rates.

Our multi-frame rate approach uses multiple image generators to render interactive parts of a scene, e. g. menus, cursor, interaction-related visualizations as well as scene objects currently manipulated by the user, with the highest possible frame rates while the rest of the scene is rendered at regular frame rates. The results of individual image generators are optically or digitally combined into a multi-frame rate image. The optical combination can be achieved by using multiple projectors displaying completely overlapped images on the same screen. Digital image composition requires either dedicated hardware like the Lightning-2 system [Stoll et al. 2001] or HP's Sepia/Sepia-2 systems [Moll et al. 1999; Lombeyda et al. 2001] or the exchange of color and depth information between different image generators. Our approach for digital composition of asynchronously generated images can be seen as an unconventional case of the Sort-Last technique [Molnar et al. 1994], which commonly focuses on balancing workload between multiple image generators to improve the overall frame rate. Instead we purposely generate a highly unbalanced load for the image generators to improve the interactivity and responsiveness of an application considerably.

Our approach is motivated by a number of observations made with different application prototypes in the automotive as well as in the oil and gas industry, where often highly complex scenes are explored and manipulated on large projection-based displays:

- Scenes are mostly static and only small parts of the scene are manipulated, like an oil well or an engine part.
- System control is quite often used, but menus, sliders, etc. are difficult to manipulate at low frame rates.
- Head tracking is rarely used. Even if it is used head-tracked users move around very little in most cases. Head tracking seems to work quite well at low frame rates while selection, object manipulation and system control become increasingly difficult.
- Navigation often involves only the coarse adjustment of view point positions, which can be achieved at relatively low frame rates.

Based on these observations we realized that object selection, object manipulation, and system control require higher frame rates to work reasonably well than navigation and head tracking for the described scenarios. This insight is either completely ignored in current systems and thus interactivity is sacrificed or it is resolved by rendering the whole scene with an appropriate frame rate by reducing visual quality, e. g. by sending fewer polygons into the graphics pipeline.

We introduce the concept of multi-frame rate rendering to improve the interaction for complex scenes while maintaining visual quality. Multi-frame rate rendering introduces certain visual artifacts for some situations, which are caused by the different update rates of scene parts. We discuss how different approaches deal with this problem, when it occurs, and how it can be ameliorated. Our user study confirms that users benefit from the improved interaction quality while certain visual artifacts are either not even recognized or gladly tolerated. Overall, digital composition shows the most promising results, since it introduces the least artifacts at the expense of transferring frame buffer content between different image generators. With the recent ubiquitous availability of single system multi-GPU configurations our approach becomes easy and efficient to implement. For some application types it may be favorable over commonly used Sort-First-based load balancing approaches for multi-GPU environments.

2 Related Work

Bergman et al. [1986] present a mechanism for adaptively refining the image presentation to the viewer. They propose to initially show the vertices only, followed by adding the edges between vertices, which in turn is followed by flat shading, shadow generation, Gouraud shading, Phong shading, and finally anti-aliasing. They reason that as long as the viewer does not change parameters this successive refinement provides a good combination of rendering speed, user convenience, and image quality. They also suggest the possibility of a *golden thread*, a single step that, repeated a few times, will generate a coarse image, and when repeated further will result in incrementally higher quality images.

Bishop et al. [1994] propose a rendering technique that allows smooth updates of an image from a scene. Instead of using a double-buffered approach, where the new image is being generated while the previous one is shown on the display, they propose pixel computation based on the most recent user input and immediately updating the pixel on the display. The resulting images would converge to a final high-quality display when user motion stops. During input changes the image display may become blurry since intermediate images contain pixels from different temporal samplings. Watson et al. [2002] follow up on this work by introducing a visual error metric, consisting of a spatial and a temporal error to control the image refinement. The proposed techniques by Bishop et al. [1994] and Watson et al. [2002] of updating single pixels on a display are practically limited to ray tracing render systems only. It is also worth noting that no display hardware currently exists capable of efficiently updating single pixels, cf. [Ferwerda 2003], making frameless rendering a more conceptual approach rather than a practical method.

Woolley et al. [2003] introduce the concept of interruptible rendering. A single image-space error measure is used to unify spatial error caused by rendering coarse representations and temporal error caused by rendering delay. A progressively refined rendering of a coarse image into the back buffer is used. During this process the temporal error is monitored and once it exceeds the spatial error, further refinement is stopped and the image is displayed. Their rendering system uses LOD-based techniques combined with real-time ray tracing.

Sher Zagier [1997] proposes the incorporation of known behavior as well as limitations of the human visual system. Coupled with camera input a personalized display could be created and combined with a refined frameless rendering approach. Dumont et al. [2003] present a perceptually-driven decision theory for interactive rendering and demonstrate the approach for various applications such as diffuse texture caching, environment map prioritization and radiosity mesh simplification.

All of the techniques mentioned so far strive for realistic image generation in the presence of user interaction. They sacrifice image quality in one way or another to preserve interactive response to the user. Our goal is the preservation of the visual quality for a given image generation process; we do not want to alter what the application developer thinks is appropriate for the intended audience. Instead we provide a means to allocate and manage a dedicated resource for the interactive parts of a scene under the assumption that it will exhibit less computational cost than rendering the whole scene.

Our work on optical multi-frame rate image composition was inspired by Majumder and Welch [2001]. They suggest the use of completely overlapped projections from multiple projectors for creating interactive depth of field effects by optical blurring, for greater parallelism and flexibility in rendering, and for generating higher-fidelity imagery. Also separating lighting components or multi-pass rendering is suggested. Multi-frame rate rendering and display also makes use of the greater flexibility when using multiple overlapping projectors, but primarily for improving user interactivity.

Another way of improving the overall rendering performance is the use of parallelized graphics, e. g. [Humphreys et al. 2002; Deering and Naegle 2002; Yang et al. 2002; Ogata et al. 2003; Bethel et al. 2003]. Molnar et al. [1994] introduce a classification scheme for reasoning about parallel rendering. It is based on where the graphical primitives are distributed to a particular screen, frame buffer, or image generator. This leads to the observation that rendering can be viewed as a problem of sorting primitives to a given screen, which was first noted by Sutherland et al. [1974]. This sort may happen anywhere in the rendering pipeline: during geometry processing (Sort-First), between geometry processing and rasterization (Sort-Middle), or during rasterization (Sort-Last). Sort-First redistributes primitives before transformation into screen space. Sort-Middle means redistributing screen-space transformed primitives. Sort-Last is the redistribution of pixels, samples, or fragments. The classification scheme allows for computational and communication costs to be analyzed. These approaches focus mostly on advanced load balancing strategies to improve the overall frame rate. This is in contrast to our approach, because we only try to improve the frame rate of the highly interactive parts of the scene.

3 Overview

We propose techniques which combine the output of several image generators (IG) in an asynchronous way using the superposition of video projectors to create an optical buffer or by digitally merging the color buffers and depth buffers of several render clients. Each IG runs at its own frame rate dependent on the workload, which is typically quite unbalanced with our approach. The combination of the output of multiple IGs leads to a multi-frame rate display or multi-frame rate images. The underlying technique for assigning data to the respective IGs and the image combination technique is called multi-frame rate rendering.

Optical Buffer The superposition of images from multiple projectors requires precise geometric and optical calibration as discussed in [Raskar 2002; Majumder 2003]. The superimposed projectors create an *optical buffer* as shown in figure 1.

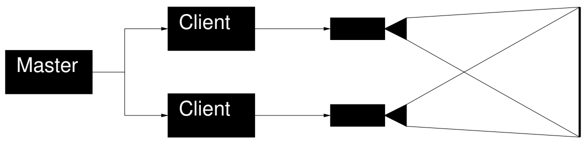


Figure 1: Optical superposition of two projectors creating an optical (output) buffer.

Digital Buffer The alternative to optical superposition of multiple images is digital composition. This can be achieved by using dedicated hardware such as the Lightning-2 system [Stoll et al. 2001] or the Sepia-2 system [Lombeyda et al. 2001]. Since these systems are not widely available, the exchange of color and depth buffers between IGs is often used—in our implementation as well. We combine the output of asynchronously running IGs in a *digital buffer* (cf. figure 2), which is a variation of the Sort-Last technique.

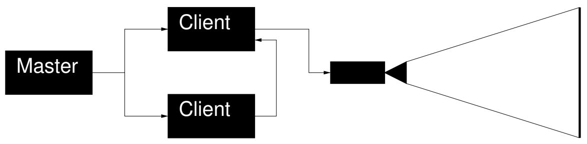


Figure 2: Digital composition of color buffer and depth buffer from two render nodes creating an digital (output) buffer.

Multi-Frame Rate Display We call the visual result of digitally combined or optically superimposed outputs from multiple asynchronously running IGs a *multi-frame rate display*. An individual image at a certain instant in time is called a *multi-frame rate image*.

Scene Superposition We assume that a scene is represented as a scene graph. Parts of the scene graph are distributed to different clients and the resulting outputs are digitally or optically combined. The sum of the parts of the scene graph results in the complete scene graph and the combination of the outputs from the different IGs results in the complete image. Figure 3 shows this process schematically. The partial scene graphs are rendered on their respective nodes and the combination of the nodes' outputs results in an image of the complete scene graph with the different parts potentially rendered at different instants in time.

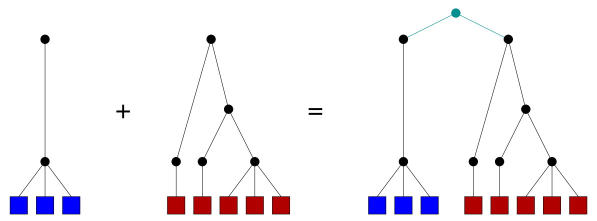


Figure 3: Scene superposition; the partial scene graphs (blue on the left and red in the middle) form the final scene graph (on the right) for the interacting user when rendered on a multi-frame rate display.

Graphics Cluster and Distributed Scene Graph To explain the different multi-frame rate rendering techniques we need to introduce some details of the hardware and software setup. The asynchronous generation of multiple images in parallel requires the use of multiple IGs. This can be achieved by using a single computer with multiple IGs or a cluster setup with at least one IG in each machine or a combination of both approaches. We use a cluster setup with a single IG in each machine. The IGs are *not* required to be synchronized in any way. For the generation of stereoscopic multi-frame rate displays some synchronization between the IGs for the left and right eye and of the overall video refresh is generally necessary and can be achieved by hardware vendor specific means.

The minimal cluster setup used for our implementation consists of a master node, a high frame rate client (*fast client*, *FC*), and a low frame rate client (*slow client*, *SC*). The master node executes an application based on a scene graph toolkit with cluster support and communicates changes in the scene to the client nodes over the network.

In our implementation the master node loads the complete scene and distributes the resulting scene graph to all clients. Thus each client holds the complete scene graph and we use the concept of traversal masks to select a certain part of the scene graph for rendering on each individual client. The traversal mask concept consists of two types of masks. Nodes in the scene graph have bit masks assigned, which we call node masks. Scene graph traversal processes, such as the rendering process, have also bit masks assigned, which are commonly called traversal masks. During traversal of the scene graph the current node mask is evaluated against the traversal mask using a bit-wise AND operation and if the result is zero, the sub graph below the current node is not further considered. Each client has a different traversal mask assigned for its rendering traversal as shown in figure 4. The master node assigns different node masks to the respective parts of the scene graph, such that these parts are rendered only on the corresponding clients. These node mask changes as well as any other changes inside the scene graph are communicated to the clients using the cluster support of the scene graph toolkit. Note that distributing the complete scene to all clients greatly simplifies the process of switching parts of the scene graph on and off on individual clients.

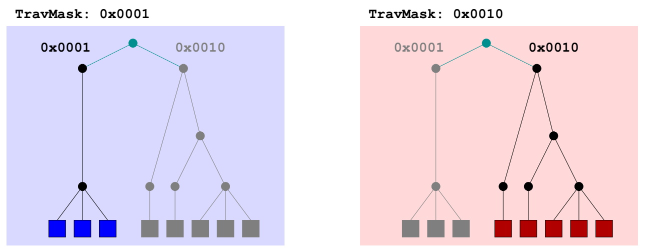


Figure 4: Traversal mask update on different cluster nodes. On the left only the node mask of the left partial graph (in blue) is compatible with the traversal mask; the partial graph on the right (in gray) is excluded from the traversal. On the right the inverse situation for another cluster node is shown.

For the remainder of this paper we are mostly focusing on the selection and spatial manipulation of single objects in the scene graph. System control techniques will work in a similar way. Navigation and head tracking will be discussed later. Object selection and manipulation is only considered in the master application. A user selecting an object in the scene would trigger a node mask re-assignment on the manipulated node. The node mask change is then communicated to all clients and thus the corresponding sub tree will be rendered only on the client with a compatible traversal mask.

4 Multi-Frame Rate Rendering by Optical Superposition

We developed two techniques for multi-frame rate rendering through optical superposition: basic asynchronous rendering and LOD-based depth testing. These techniques require the setup of a graphics cluster consisting of at least three nodes as described before.

4.1 Basic Asynchronous Rendering

Figure 5 shows the actual setup of the rendering cluster consisting of a master and two client nodes. The scene is distributed to both clients from the master. *SC* and *FC* have different traversal masks for their render traversal processes. Initially the nodes in the scene have traversal masks so that they are only rendered by *SC*. If a part of the scene is selected by the user on the master its node mask is changed to a node mask that is compatible with the traversal mask of the *FC*. This affects the actual scene content rendered by the clients. The *SC* will exclude the selected part of the scene graph from its traversal process while the *FC* will include this part. This way any selected object is exclusively rendered at full frame rate on the *FC*. The node mask is reversed at the end of the selection.

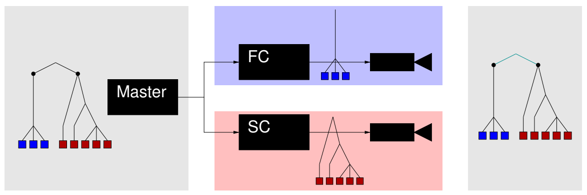


Figure 5: Basic asynchronous rendering setup sketch. *SC* is the "slow client" while *FC* is the "fast client."

Figure 6 shows digital photographs taken from a multi-frame rate display running in basic asynchronous rendering mode. Figure 6a shows the final image produced by two fully overlapped projectors while figures 6b and 6c show the images from the projectors attached to *SC* and *FC*, respectively. Figure 6a clearly shows that no correct depth relation can be resolved in the optical buffer leading to half-transparency effects. This is discussed in more detail in section 6.

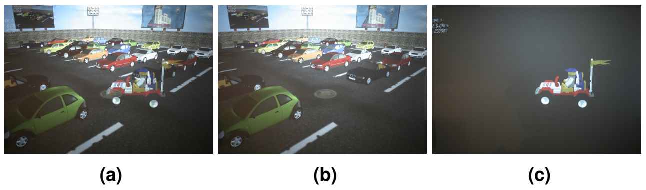


Figure 6: Basic asynchronous rendering and image composition in the optical buffer. (a) Combined output of the overlapped projectors. (b) Projector showing scene (*SC*). (c) Projector showing selection (*FC*).

4.2 LOD-Based Depth Testing

The goal for developing LOD-based depth testing was to avoid the half-transparency effects exhibited in the basic asynchronous rendering technique (section 4.1) and to generate correct depth relationships. The idea is that both clients, *SC* and *FC*, render the whole scene, but the parts with incompatible node masks are rendered to the depth buffer only. This allows the generation of correct depth relationships for two optically blended images—as long as there is no interaction. Figure 7 shows the schematic setup for LOD-based depth testing.

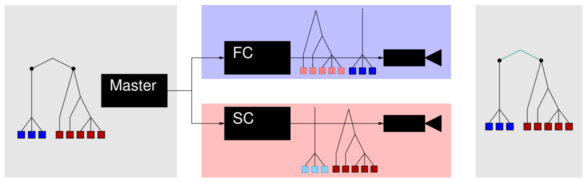
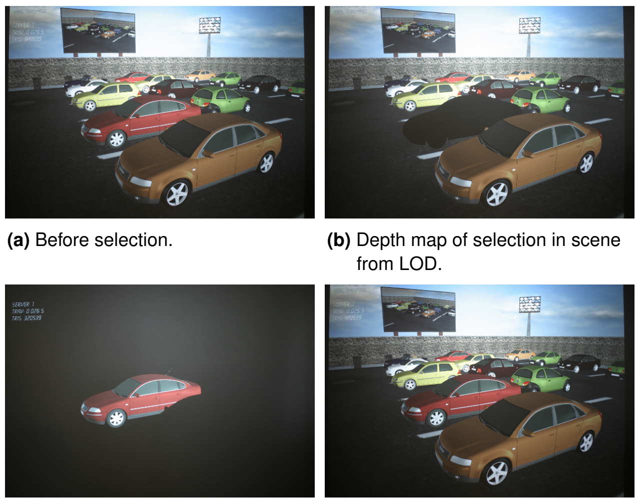


Figure 7: LOD-based depth testing setup sketch. *SC* is the "slow client" while *FC* is the "fast client". The *FC* first renders a lower resolution representation of the scene assigned to *SC* before rendering its own scene part. *SC* proceeds inversely.

The scene graph is distributed by the master node to the client nodes in the same way as for the basic asynchronous rendering approach. The *SC* renders the currently selected object into the depth buffer only and the rest of the scene with regular level-of-detail and shading afterward. The *FC* renders the inverse configuration, i. e. the rest of the scene is rendered into the depth buffer only and then the selected object is rendered as usual. When an object is selected its node mask is changed so that it will create depth values only on the *SC* but will be rendered normally on the *FC*. The *FC* renders the parts of the scene graph with incompatible node masks, i. e. the whole scene excluding the currently selected object, with low level-of-detail to preserve high frame rates. In addition texturing and shading is turned off for this part of the scene.

Figure 8 shows digital photographs taken from a multi-frame rate display running in LOD-based depth testing mode. Figure 8a shows the combined view before any selection, i. e. the image is generated solely by the *SC*. Figure 8b shows the part rendered by the *SC* after selection occurred; a black shape can be seen, from rendering into the depth buffer only, where the formerly selected object was located. Figure 8c shows the image from the *FC* at the same instant as figure 8b; only the selected object is visible but partly covered by the depth values of an object located nearer to the viewer (and thus rendered by the *SC*). Finally figure 8d shows the optical superposition of both figure 8b and 8c.



(a) Before selection. **(b)** Depth map of selection in scene from LOD.

(c) Selection occluded by other object. **(d)** Combined view.

Figure 8: LOD-based depth testing and image composition in the optical buffer. (b) shows the projection from *SC* while (c) shows the part projected by *FC*; (a) and (d) show the combined view of both projectors.

5 Multi-Frame Rate Rendering by Digital Composition

We also developed a technique for multi-frame rate rendering through digital composition using frame buffer exchange. With this approach only the fast client displays the final image. The clients share a common viewing setup such that the resulting images can be digitally merged without perspective inconsistencies. As a consequence head tracking is also limited to the frame rate of the slow client.

The digital composition approach transfers the slow client's depth and color buffers to the fast client. The *FC* initializes its depth and color buffer with this pre-rendered information followed by rendering the currently selected objects. This approach can be seen as a variation of the Sort-Last approach [Molnar et al. 1994], which gathers images of a subset of the scene using an image composition node or a dedicated hardware compositor. Traditional Sort-Last has to wait until all the images arrive, so the slowest rendering node determines the frame rate. Instead we do not wait for the slow client. We always render at the frame rate of the *FC* and incorporate new color and depth information from the slow client as it becomes available. The schematic setup is shown in figure 9.

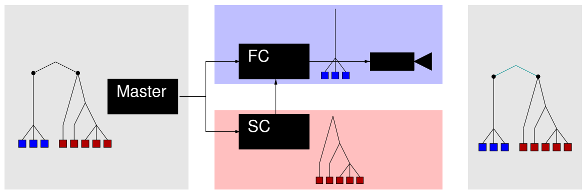


Figure 9: Frame buffer exchange setup sketch. *SC* is a "slow client" while *FC* is a "fast client". Only *FC* is connected to a projection device and responsible for generating the image. *SC* is responsible for updating periodically the depth and color information of its associated *FC*.

Digital compositing introduces frame buffer transfer overhead to the rendering process. After the *SC* has rendered the parts of the scene graph with a compatible node mask, the generated color and depth buffers are read back into main memory and send through the network to the corresponding *FC*. Upon arrival of a new frame buffer at the *FC* these depth and color buffers are transferred into the graphics subsystem. Any objects assigned to *FC* are rendered afterward. Because the depth buffer from *SC* was already applied objects rendered by *FC* will be correctly occluded if the *SC* does not render animated objects. *SC*'s depth and color buffers will be re-used each frame on the *FC* until updated versions from the *SC* arrive. Note that the *SC* does not need to render and send frame buffers in a fixed cycle. It is sufficient to perform this process only when changes in the scene graph occur or the view transformation is updated, e. g. because of navigation or head tracking.

Figure 10 shows digital photographs taken from a multi-frame rate display running in frame buffer exchange mode. Figures 10a and 10c show the scene as perceived by the user with an object selected in figure 10a. Figures 10b and 10d show the same scene with the color buffer content from the *SC* converted to gray scale for emphasis. Note that the formerly selected object in figure 10b is shown in gray scale in figure 10d because of object deselection.

6 Rendering Artifacts

Multi-frame rendering introduces various artifacts due to the asynchronous combination of images from different image generators. For spatial object manipulation tasks these artifacts mostly appear during selection of an object and while an object is moving or interpenetrating other objects. The suggested multi-frame rate rendering techniques deal in different ways with these problems.

Basic asynchronous rendering in combination with optical superposition is the most simple way to drive a multi-frame rate

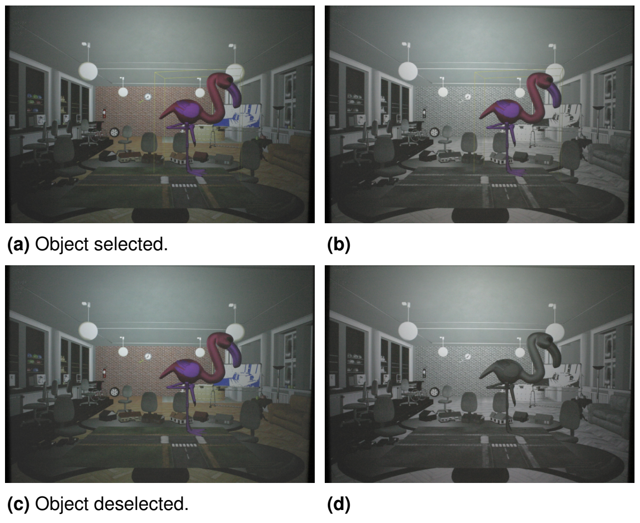


Figure 10: Frame buffer exchange image composition. (a) and (c) show an selected and deselected object, respectively, as perceived by the user; (b) and (d) show the same process with the contents from the *SC* post-processed to gray scale and the contents rendered by the *FC* in color.

display. As explained in section 4.1 the overhead only consists of the one-time distribution of the scene graph and the occasional update of node mask values for a small number of scene graph nodes.

The main artifact for this technique is caused by the fact that there is no depth relationship computed between the slow and fast client's objects and the respective images are simply projected on top of each other. Thus occlusion of objects is not properly resolved and objects from *SC* and *FC* displayed on top of each other appear translucent.

Another effect is the visual perception of popping artifacts. When a user selects an object it takes one frame of the *SC* to update its node mask and the changes to be actually considered by the render traversal afterward. The associated *FC* also needs only one frame to update the node mask of the respective node(s) in its scene graph copy but this happens much faster. Therefore while *SC* is still displaying an image with the selected object included *FC* will also show the selected object. This double appearance in the optical buffer creates a double bright object, which is perceivable by the user. Depending on the difference of actual frame rates between *SC* and *FC* this results in a short flashing on the display for small differences in frame rates up to a prolonged local brightness increase for larger differences in frame rates. The inverse effect happens in the case of object deselection. Here, while *FC* already removed the object *SC* needs more time for the update, which results in a time period where the deselected object is not displayed at all.

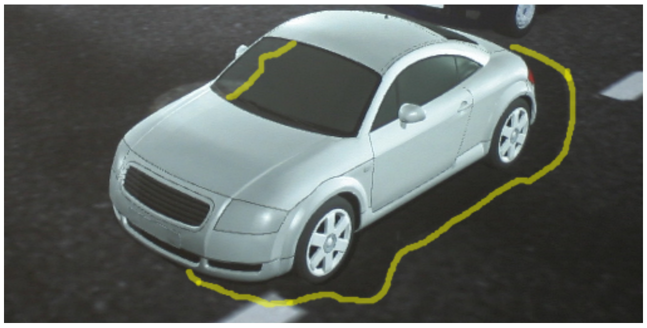


Figure 11: LOD-based depth testing artifact due to frame rate differences being too large between *SC* and *FC*. The shadow-like outline (marked-up) is the depth map of the object in the *SC* not yet updated to the final object position.

LOD-based depth testing in combination with optical superposition was primarily developed to avoid half-transparencies. This is traded for a *dragging* depth shadow if the frame rates differ substantially between *FC* and *SC*. In figure 11 a detail from a digital photograph of a multi-frame rate display running in LOD-based depth testing mode is shown where the difference in frame rates is large enough, resulting in a delayed dragging of the depth mask for a selected object in the depth buffer of the *SC* (marked outline in figure 11). This shape boundary discrepancy is resolved when user interaction is stopped or becomes very slow, which enables fine grained positioning with correct depth relations while coarser interaction patterns will exhibit a dragging depth shadow.

Even though LOD-based depth testing solves the half-transparency problem to some degree the popping artifacts as discussed for the basic asynchronous rendering method still remain.

Digital composition as a rendering method for a multi-frame rate display appears to be most promising. It completely avoids half-transparencies in comparison to optical superpositioning techniques. This comes at a price though, since it requires a more sophisticated software and hardware infrastructure. The combination of *SC* and *FC* need to be able to exchange data of a low-latency high-bandwidth nature through a peer-to-peer connection. The (assumed) cluster distribution mechanism of the scene graph is almost always inappropriate for such client-to-client transfers.

The low-latency high-bandwidth data transfer will also restrict the maximum achievable update rate. Assuming a frame buffer resolution of 1280×1024, a 32 bit color buffer, and a 24 bit depth buffer one frame buffer update would amount to roughly nine megabytes of data. Using a dedicated gigabit network a bandwidth of 90 megabytes per second can be realized in practice. Thus, sending frame buffer updates from *SC* to *FC* would let us achieve a change rate of ten frame buffers per second on the *FC* for newly arrived updates from the *SC*. This would determine the frame rate at which a user could navigate using head tracking. Thus the network bandwidth is the limiting factor since reading and writing color and depth buffers is around four times faster than the network transfer itself on current graphics hardware; if only taking into account writing of the buffers on the *FC* the network transfer is slower by an order of magnitude!

Finally, the data transfer mechanism also influences user interaction. The selection process is coupled to the frame buffer exchange rate between *SC* and *FC*, while the object manipulation will run at full frame rate on the *FC* once the selected object appeared. The need to create a frame buffer update on the *SC*, sending it over the network, and applying it on the *FC* may create a temporarily inconsistent display, if the user already started moving the selected object. This is due to the fact, that the selected object is still contained in the old frame buffer from the *SC*, but also rendered by the *FC*. Only when an updated frame buffer from the *SC* arrives this double rendering artifact will be resolved. This is the corresponding effect to the flashing or popping in the optical superposition of images.

7 Implementation

Since the introduction of a distributed graphics API by MacIntyre and Feiner [1998] several approaches have been realized and presented to the research community, e. g. [Hesina et al. 1999; Tramberend 1999; Voß et al. 2002; Naef et al. 2003]. Our implementations are based on OpenSG [Reiners 2002] which is a very sophisticated scene graph API providing inherent support for concurrent scene graph traversal and distributed graphics applications.

OpenSG uses a multi-aspect mechanism to support multi-threaded graphics applications by minimizing lock delays. Conventional OpenSG applications consist of a set of server programs and a client program. The server programs are started on their respective nodes. The client program configures the display setup of each server allowing for tiled walls as well as composition schemes. Changes to the scene graph on the client program are communicated on a frame by frame basis and the frame boundaries of all participating programs are locked to a network barrier by default. To achieve our asynchronous render management we employ a local per client thread scheme where each client employs a thread that listens to the changes from the network and another thread that is responsible solely for rendering. Changes from the network are applied to one aspect of the local scene graph copy which are merged at the end of the render thread's frame. The frame buffer exchange technique (section 5) uses the regular gigabit network connection between *SC* and *FC* for sending frame buffer updates. In our implementations all participating clients are decoupled from the frame rate of any other client. Note that in this nomenclature an OpenSG server is a slow or fast client, while the OpenSG client is our master application.

We experimented with our techniques on our passive stereo projection setups. Each projector was connected to a PC running an AMD Athlon 64 3000+ processor, 2 GB main memory, and an nVidia FX 3400 PCIe-based graphics board. A third PC, using the same configuration, was assigned to run the master application responsible for device input and scene graph event distribution. All PCs were connected by a switched gigabit network.

8 User Study and Discussion

Our hypothesis is that the improved interaction performance using our multi-frame rate rendering techniques facilitates common selection and object manipulation tasks in virtual environments. While there are many parameters that can be studied with our setup, we focused on a head-tracked 3D docking task running at 10 Hz on the *SC* and at 30 Hz on the *FC*. We compared the basic optical superpositioning approach (MF_{opt}) and our digital compositing technique (MF_{dig}) to the conventional single-frame rate rendering (SF_{10}) of the whole scene at 10 Hz on a single graphics card. The SF_{10} scenario served as our lower baseline. Since we perform the interaction at 30 Hz, while head tracking and rendering the rest of the scene happens at 10 Hz, the upper baseline is provided by conventional single-frame rate rendering at 30 Hz (SF_{30}). The optical superposition technique supporting LOD-based depth testing was excluded from the study since the fast object manipulation during a docking task would strongly pronounce the artifacts of this technique. We selected a scene that could be rendered at 30 Hz on a single graphics card and limited the frame rate for the different techniques appropriately.

8.1 User Study

We recruited sixteen volunteers for our study, who were all daily users of computer technology and most of them had worked with 3D graphics before. All participants had stereo vision capabilities and could interact with stereoscopically displayed objects which were positioned in front of the screen.

Our 3D docking task required participants to select a small ball-shaped object (3 cm diameter) from a randomly selected location on one side of the screen and move it to the other side, where the object had to be dropped off in a ring-shaped target location as can be seen in figure 12 and 13. The pick-up and drop-off location were randomly chosen from a set of predefined positions such that their distance was always equal. In addition the ball-shaped object at the pick-up location was occluded when viewed from the center position in front of the screen to enforce the use of head tracking. Selection required some head movement toward the appropriate side of the screen, followed by further head movement toward the drop-off location. The task was performed with an optically tracked input device using a ray-casting metaphor and required three degrees of freedom. The orientation of the manipulated object was not considered. The displayed image was restricted to 1080×440 pixels for all tests to allow a conservative 10 Hz network transfer rate for the frame buffer updates in the MF_{dig} method. We used passive stereo and linear polarization.

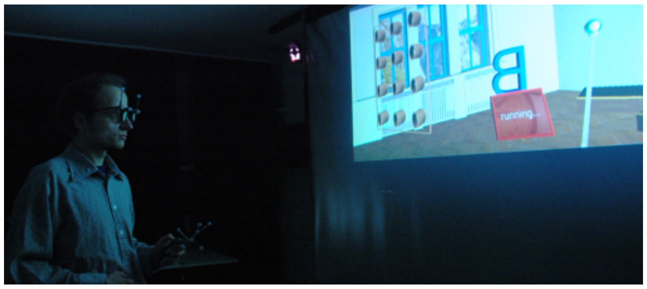


Figure 12: Setup used during the user study.

All the participants performed the docking task with each of the techniques. The order of techniques (SF_{10}, MF_{opt}, MF_{dig}, and SF_{30}) was balanced across participants using a Latin Square design. The task was explained to each participant followed by a practice run with each of the four techniques in the same order as the subsequent trials. A trial was started by pointing at a start button in the middle of the screen and pressing a button on the input device. The trial was finished once the ball-shaped object was dropped off at the target location. See figure 13 for screenshots from the running application. The required precision was 1.5 cm, which was half the diameter of the manipulated ball-shaped object. Each participant performed 15 trials with each method. At the end of each method participants filled out a written questionnaire asking about problems with each method. After the participants were finished with all the techniques they reported about their preference with respect to each rendering technique on a one to five scale.

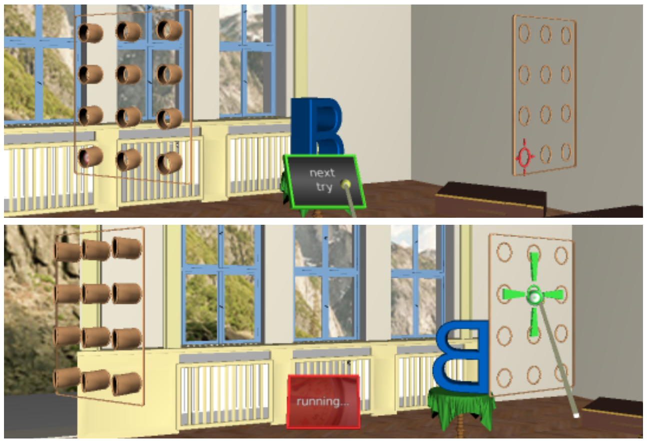


Figure 13: Screenshots of the application prototype used in the user study; Upper image shows the start of a trial and the lower image trial finish.

The task completion times (TCT) were entered in a 4×1 analysis of variance for repeated measures with the within-factors rendering techniques (SF_{10}, MF_{opt}, MF_{dig}, and SF_{30}) as well as the between-factor order of techniques. The order of the techniques did not produce a main effect nor did it interact with the technique indicating that there was no transfer between the techniques probably due to the simplicity of the chosen task and the extended practice runs.

The performance of the four techniques differed significantly $F_{3,188} = 10.01, p < .001$. Docking with SF_{10} took 5.75 seconds (standard error $se = .26$), while MF_{opt} produced even longer TCTs of 6.47 seconds ($se = .49$). The shortest TCTs were obtained with SF_{30} 4.26 seconds ($se = .19$) followed by our MF_{dig} technique with 4.75 seconds ($se = .23$). Figure 14 shows these results. Post-hoc comparisons revealed that all techniques differed from each other ($p < .05$) except the pair SF_{10} and MF_{opt} as well as the pair MF_{dig} and SF_{30} did not produce significant differences. User preference ratings (cf. figure 14b) of the techniques differed significantly as well $F_{3,60} = 16.91, p < 0.05$. Post-hoc comparisons showed the same results as for the TCTs. Thus the subjective measures are exactly coincident with the results of the performance measures.

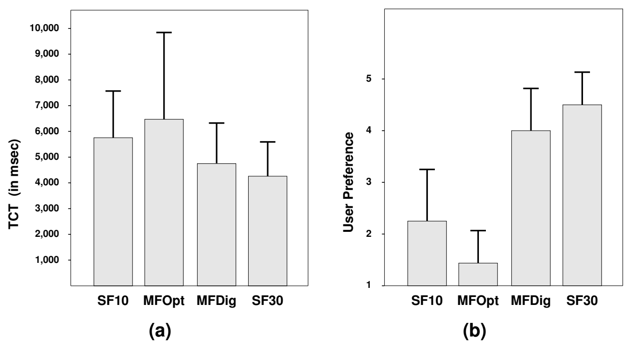


Figure 14: User study results. (a) Task completion times (mean values and standard deviation, in milliseconds) for single-frame rate rendering at 10 Hz (SF_{10}) and 30 Hz (SF_{30}) as well as multi-frame rate rendering using optical superposition (MF_{opt}) and digital composition (MF_{dig}) at a frame-rate ratio 10/30 Hz for *SC* and *FC*, respectively; (b) User preference for the methods tested (mean values and standard deviation, from 1 = dislike to 5 = prefer).

8.2 Discussion

In summary our digital image composition approach MF_{dig} performs significantly better than simply rendering at 10 Hz frame rate (SF_{10}). Even more the MF_{dig} TCTs and the user preferences are almost at the level of rendering the whole scene at 30 Hz (SF_{30}), which shows the potential of this approach. At first sight the performance of our optical superposition technique MF_{opt} seems to be disappointing, since it is the overall slowest technique and the participants also did not like the technique. Further investigation of the questionnaires revealed that the depth perception during selection and drop-off of the manipulated object was problematic. This is most likely due to the two conflicting depth cues: occlusion and stereopsis. The simple optical superposition does not provide any occlusion information. This information was particularly important during the selection process, since we used a selection ray of only 1.5 meters length. The end of the ray had to pierce the ball-shaped object, which was difficult to judge just from the stereoscopic parallax. Thus the basic optical superposition technique is not well suited for these types of tasks, where objects intersect and precise manipulation is required. However it is very easy to implement.

Multi-frame rate techniques trade rendering artifacts for interaction performance. A particular selection artifact occurs since the slow client needs to remove the selected object from its frame buffer, which happens with a delay depending only on the frame rate of the *SC*. The selected object is just activated on the *FC* and can be manipulated right away. However, while the *SC* has not yet updated its frame buffer (and sent to the *FC*) the selected object is still contained in the *SC* and *FC* renderings. This results in different visual artifacts for the multi-frame rate rendering techniques as described in section 6. For dropping off objects a similar artifact occurs. However, the participants of our study did not mention these artifacts in the questionnaire indicating that they are not very prominent at 10 Hz. If the *SC* and the *FC* would sync their swap buffer operations every N frames and their video signals accordingly, the visual artifacts could be even avoided at the expense of introducing selection lag. In addition the hardware and software overhead to guarantee a swap-locked and video-synced system is also considerable. Another option would be the use of shadow objects on the fast client as they are suggested for distributed VR systems [Benford and Mariani 1993]. In combination with the synchronized systems scenario shadow objects would generate a controlled transition from *SC* to *FC*.

Multi-frame rate techniques using the digital composition method introduce extra lag into an application. Color buffer and depth buffer contents at the *SC* needs to be read back from the graphics board into host memory, send over the network to the *FC*, and there these buffers must be fed back into the graphics sub-system. We found that current graphics hardware can upload color and depth buffers for a resolution of 1280×1024 pixels in less than 20 milliseconds, downloading into the graphics card takes less than 10 milliseconds. Transferring the data over a switched gigabit network may take up to 120 milliseconds, which is our main bottleneck. However 10 gigabit network adapters and switches are becoming readily available, which will reduce the total lag considerably.

9 Conclusions and Future Work

We have introduced multi-frame rate rendering techniques which optically or digitally combine the results from two asynchronously running image generators. We have used this approach to assign interactive parts of a scene to one GPU and rendering the rest of the scene on the other GPU. This purposely creates a highly unbalanced load on the image generators to improve the interactivity of the application while maintaining the overall visual quality. Our user study confirms the increased object manipulation performance and reveals that visual artifacts for the digital image composition technique are in most cases acceptable or not even recognized. Based on these observations we believe that our multi-frame rate approach has the potential to be used in various complex graphics applications, where the focus is often on interaction with only small parts of the whole scene.

Our research into multi-frame rate rendering and display methods is just at the beginning. We have only implemented basic navigation tools which simply run on the slow client. Since navigation and object manipulation do happen in sequence only and not in parallel, this is a possible solution. On the other hand head tracking does happen in parallel with object manipulation. In our user study scenario head tracking at 10 Hz did not seem to have a strong negative impact on user performance. The question is which frame rate ratios between the slow and fast client still provide a usable system and what are the limits for head-tracked scenarios? The many parameters of our multi-frame rate rendering techniques need to be investigated and evaluated in further user studies to identify the best trade-offs between artifacts, overall visual quality, and interactivity for different application scenarios.

The recent availability of common off-the-shelf solutions for single-system multi-GPU configurations makes the implementation of multi-frame rate rendering methods easier and much more efficient. In particular a single-system solution allows the use of a conventional scene graph API instead of distributing the scene graph to a cluster of graphics nodes. An application would dedicate one GPU to rendering the non-interactive parts of the scene while the other GPU handles the interactive parts. The GPUs would run asynchronously. Frame buffer content should be ideally exchanged without intermediate transfer through host memory.

Multi-frame rate rendering can be combined with common parallel graphics approaches such as Sort-First, Sort-Middle, and Sort-Last. The possibility of dynamically re-assigning graphics processors between the sets of fast and slow clients is a great way to trade interactivity for overall rendering performance on demand.

Acknowledgments

We thank Anke Huckauf, André Kunert, Andreas Simon, and the members of our VR systems group for help, discussion, and encouragement. We also thank Roland Blach and Matthias Bues for initial discussions on the multi-frame rate idea.

References

S. Benford and L. Mariani. Requirements and Metaphors of Shared Interaction. COMIC Project, Esprit Basic Research Project 6225, Deliverable D4.1, Lancaster University, October 1993. ISBN 0-901800-31-7.

L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image Rendering by Adaptive Refinement. In D. C. Evans and R. J. Athay, editors, *Computer Graphics (Proceedings of ACM SIGGRAPH 86)*, volume 20(4), pages 29–37. ACM, 1986.

E. W. Bethel, G. Humphreys, B. Paul, and J. D. Brederson. Sort-First, Distributed Memory Parallel Visualization and Rendering. In *PVG '03: Proceedings of the 2003 IEEE Symposium on Parallel and Large-Data Visualization and Graphics*, page 7. IEEE Computer Society, 2003.

G. Bishop, H. Fuchs, L. McMillan, and E. J. Scher Zagier. Frameless Rendering: Double Buffering Considered Harmful. In A. Glassner, editor, *Proceedings of ACM SIGGRAPH 94*, Computer Graphics Proceedings, Annual Conference Series, pages 175–176. ACM, 1994.

M. Deering and D. Naegle. The SAGE Graphics Architecture. In J. F. Hughes, editor, *Proceedings of ACM SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 683–692. ACM, 2002.

R. Dumont, F. Pellacini, and J. A. Ferwerda. Perceptually-Driven Decision Theory for Interactive Realistic Rendering. *Trans. on Graphics*, 22(2):152–181, 2003.

J. A. Ferwerda. Three Levels of Realism in Computer Graphics. In *SPIE Human Vision and Electronic Imaging '03*, pages 290–297. SPIE, 2003.

G. Hesina, D. Schmalstieg, A. Furhmann, and W. Purgathofer. Distributed Open Inventor: A Practical Approach to Distributed 3D Graphics. In *VRST '99: Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 74–81. ACM, 1999.

G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters. In J. F. Hughes, editor, *Proceedings of ACM SIGGRAPH 2002*, Computer Graphics Proceedings, Annual Conference Series, pages 693–702. ACM, 2002.

S. Lombeyda, L. Moll, M. Shand, D. Breen, and A. Heinrich. Scalable Interactive Volume Rendering Using Off-the-Shelf Components. In *IEEE 2001 Symposium on Parallel and Large-Data Visualization and Graphics*, pages 115–121. IEEE Computer Society, 2001.

B. MacIntyre and S. Feiner. A Distributed 3D Graphics Library. In M. F. Cohen, editor, *Proceedings of ACM SIGGRAPH 98*, Computer Graphics Proceedings, Annual Conference Series, pages 361–370. ACM, 1998.

A. Majumder. *A Practical Framework to Achieve Perceptually Seamless Multi-Projector Displays*. PhD thesis, University of North Carolina at Chapel Hill, Department of Computer Science, 2003.

A. Majumder and G. Welch. Computer Graphics Optique: Optical Superposition of Projected Computer Graphics. In *EGVE/IPT 2001: 7th EG Workshop on Virtual Environments & 5th Immersive Projection Technology Workshop*, pages 209–218. Center of the Fraunhofer Society Stuttgart IZS, 2001.

L. Moll, A. Heinrich, and M. Shand. Sepia: Scalable 3D Compositing Using PCI Pamette. In K. L. Pocek and J. Arnold, editors, *IEEE Symposium on FPGAs for Custom Computing Machines*, pages 146–155, 1999.

S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A Sorting Classification of Parallel Rendering. *IEEE Comput. Graph. Appl.*, 14(4): 23–32, 1994.

M. Naef, E. Lamboray, O. Staadt, and M. Gross. The Blue-C Distributed Scene Graph. In J. Deisinger and A. Kunz, editors, *IPT/EGVE 2003: 7th International Workshop on Immersive Projection Technology, 9th Eurographics Workshop on Virtual Enviroments*, pages 125–133. Center of the Fraunhofer Society Stuttgart IZS, 2003.

M. Ogata, S. Muraki, X. Liu, and K.-L. Ma. The Design and Evaluation of a Pipelined Image Compositing Device for Massively Parallel Volume Rendering. In *VG '03: Proceedings of the 2003 Eurographics/IEEE TVCG Workshop on Volume Graphics*, pages 61–68. ACM, 2003.

R. Raskar. *Projector-Based Three Dimensional Graphics*. PhD thesis, University of North Carolina at Chapel Hill, Department of Computer Science, 2002.

D. Reiners. *OpenSG: A Scene Graph System for Flexible and Efficient Realtime Rendering for Virtual and Augmented Reality Applications*. PhD thesis, Technische Universität Darmstadt, Fachbereich Informatik, 2002.

E. J. Sher Zagier. Perceptually-Driven Graphics. Technical Report TR97-017, University of North Carolina, Computer Science Department, 1997.

G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan. Lightning-2: A High-Performance Display Subsystem for PC Clusters. In E. Fiume, editor, *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, pages 141–148. ACM, 2001.

I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A Characterization of Ten Hidden-Surface Algorithms. *Computing Surveys*, 6 (1):1–55, 1974.

H. Tramberend. Avocado: A Distributed Virtual Reality Framework. In *Proceedings IEEE Virtual Reality '99 Conference*, pages 14–21. IEEE Computer Society, 1999.

G. Voß, J. Behr, D. Reiners, and M. Roth. A Multi-Thread Safe Foundation for Scene Graphs and its Extension to Clusters. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 33–37. Eurographics, 2002.

B. A. Watson, D. Luebke, and A. Dayal. Breaking the Frame: A New Approach to Temporal Sampling. URL http://cde.ncsu.edu:16080/watson/docs/sig02.frameless.pdf. Technical Sketch, ACM SIGGRAPH, 2002.

C. Woolley, D. Luebke, B. Watson, and D. Dayal. Interruptible Rendering. In *SI3D '03: Proceedings of the 2003 Symposium on Interactive 3D Graphics*, pages 143–151. ACM, 2003.

J. Yang, J. Shi, Z. Jin, and H. Zhang. Design and Implementation of a Large-Scale Hybrid Distributed Graphics System. In *EGPGV '02: Proceedings of the Fourth Eurographics Workshop on Parallel Graphics and Visualization*, pages 39–49. Eurographics, 2002.