*SEARIS Workshop at IEEE VR 2008.*

# From Single-Frame Rate to Multi-Frame Rate Systems

Jan P. Springer Bernd Froehlich

Bauhaus-Universität Weimar

## Abstract

System response and thus interactivity is strictly tied to the frame rate in most virtual reality systems. For complex scenes either the visual quality needs to be sacrificed (e. g. by using a lower level of detail) or the system becomes non-interactive. We suggest that for standard manipulation tasks in virtual environments parting from such a single-frame rate approach is beneficial.

Multi-frame rate rendering uses two (or more) graphics cards to render the interactive parts of the scene on one card and the rest on the other. The results are optically or digitally combined in a single image displayed to the user. This approach significantly improves the interaction fidelity for medium and large scenes without compromising visual quality. We provide a detailed motivation and an overview of our approach. We argue that modern virtual reality system benefit greatly from multi-frame rate rendering.

**Keywords:** Interaction Fidelity, Visual Quality, Single-Frame Rate and Multi-Frame Rate Systems, System Responsiveness

**Index Terms:** F.1.2 [Computation by Abstract Devices]: Modes of Computation—Interactive and Reactive Computation; I.3.2 [Computer Graphics]: Graphics Systems—Distributed/Network Graphics; 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 Motivation

The interactive and high-quality visualization of large models and scenes 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 or *interaction fidelity* of applications. 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 interaction fidelity, they often lead to low frame rates if excellent visual quality is desired.

Interaction fidelity describes how users perceive their interactive input being incorporated into the image generation process. In contrast to high-quality imagery shown in movies, computer graphics used in visualization and simulation contexts is expected to be interactive. Users want to be able to manipulate the elements of the 3D world and perceive these changes immediately on the display.

Incorporating interaction responses into an image is not a very time consuming process, i. e. usually the computational resources required are low compared to the resources involved for generating high-quality imagery. For interaction it is more important to minimize the time from real-world sensor acquisition to incorporating the resulting changes into the image. Conventional single-frame rate systems exhibit a tightly coupled visualization and interaction architecture, i. e. sensor data is evaluated as part of the visualization loop. In such systems interaction response is limited by the time it takes to render an image. Earlier research found a dependency between the frame rate of an application and the corresponding interaction fidelity. For sufficient interaction fidelity it is recommended that the frame rate should be $\geq 6$ Hz [Airey et al. 1990], $> 7$ Hz [Pausch 1991], or $> 10$ Hz [Card et al. 1991; McKenna and Zeltzer 1992; Bryson 1993]; these values are mostly rules of thumb derived from experience or extrapolated from 2D GUI experiments.

More formal studies investigated the interaction-fidelity to frame-rate relationship for task performance in VR applications [Tharp et al. 1992; Ware and Balakrishnan 1994; Reddy 1997; Watson et al. 1998]. They all arrive at the conclusion that sufficient interaction fidelity in a single-frame rate system requires a frame rate above 10 Hz. Tharp et al. [1992] and Reddy [1997] found that for frame rates above 20 Hz and 15 Hz, respectively, no dramatic performance improvement can be seen, though both agree that the higher the frame rate the better the task performance. Watson et al. [1998] focused on studying system response for (open-loop) grasp and (closed-loop) placement tasks [Wickens 1992]. They found that open-loop tasks (e. g. grasping) seem to be less sensitive to system responsiveness than closed-loop tasks (e. g. placement) because the latter require continuous visual feedback for appropriate control. They describe system response as the time elapsed from a user's action to the display of that action. In contrast, frame time is the time an image is presented on the display while system latency describes the age of a sample presented in the image. Frame time and system latency are the two main influences on system response. The relationship between these entities is depicted in figure 1.

The age of a sample displayed in an image consists mainly of the time it takes to process the data from a sensor and sending it to the application. This sensor data latency is typically constant for a certain setup (e. g. a tracking system). Once the sensor data arrives at the application it is incorporated into the application state (e. g. matrices of a scene graph). Based on the modified application state a new frame is rendered. Rendering time is typically equal or similar to the time an image is displayed—the frame time. In low frame rate applications system response and system latency are thus dominated by frame time. System latency affects how well the response on the display is connected to the action of the user. The frame time also contributes to this perception of action and reaction, but additionally affects the smoothness of object movements and view changes.

From the above discussion it can be concluded that improving the frame time for the interactive parts of a scene will result in significantly improved interaction fidelity. Closed loop tasks such as place-

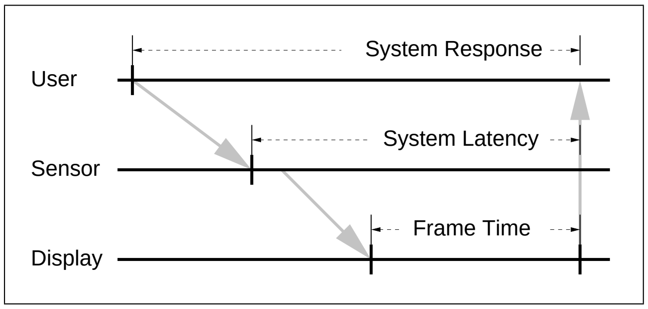

**Figure 1:** System response, system latency, and frame time relationship (after Watson et al. [1998]).

ment tasks are particularly affected by overall system responsiveness. Because grasping and placement are common manipulations in VR applications we concluded that parting from the single-frame rate approach for image generation might be beneficial. In a single-frame rate system interactive frame rates can only be guaranteed by controlling parameters affecting visual quality. Our goal was to achieve frame rates sufficient for interaction with the 3D world while also supporting high-quality visualization methods exhibiting very low frame rates. For this approach two requirements must be fulfilled. First, it must be possible to designate parts of the scene's content as relevant to the current interaction. This might be inferred automatically (e. g. by interest prediction) or explicitly by state changes with respect to the interaction framework used (e. g. objects must be selected before being manipulated). Second, scene parts relevant for the interaction will be rendered on a dedicated GPU. By assuming that usually only small parts of the scene are relevant to the current interaction this should allow for sufficiently high frame rates of interaction responses to the user, i. e. minimizing the time for system response. The rest of the scene is rendered on a separate GPU as well but typically with much lower frame rates. Both of these graphics sub-systems are assumed to run asynchronously, i. e. they are not expected to be synchronized in any way. Finally, the partial images must be combined in some way for displaying the complete scene to the user.

## 2 Multi-Frame Rate Systems

Before actually explaining multi-frame rate methods in more detail some related work trying to move beyond single-frame rate systems will be discussed.

Bishop et al. [1994] propose a *frameless* 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 per-pixel computation based on the most recent user input and immediately updating individual pixels on the display. The resulting images would converge eventually to a high-quality image once 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] and later Dayal et al. [2005] follow up on this work by introducing a visual error metric, consisting of a spatial and a temporal error controlling the image refinement. All of the proposed techniques are practically limited to ray-tracing render systems.

Woolley et al. [2003] introduce the concept of interruptible rendering. A single image-space error measure is used to unify the spatial error caused by rendering coarse representations and the 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.

It is interesting to note that these approaches, as well as many others that follow a single-frame rate approach, will trade visual quality for interaction fidelity in some way. The conceptually interesting approach of frameless rendering still trades visual quality for interactivity, but the granularity of the update is on a per-pixel level. We argue that updating on a per-object level is much more efficient since typically only few objects are affected by an interaction in a virtual environment. We will see that multi-frame rate rendering also trades interactivity for visual quality in the form of artifacts, which fortunately can be mostly hidden.

### 2.1 Multi-Frame Rate Display

The multi-frame rate image generation process produces more than one image (e. g. one for the non-interactive scene parts and one for the interactive objects). Basically two methods exist for displaying the combined result on a single display: optical superposition and digital composition [Springer et al. 2007]. In both cases it is assumed that the images are produced by multiple asynchronously running image generators.

By using two completely overlapping projection devices the partial images from the asynchronous image generation processes can be combined in an easy way. The rendering processes must be configured to output images for identical view projection setups. Also, the projectors should be precisely calibrated, geometrically as well as optically, for the images to be perceived as one by the user. Unfortunately, since the images are merged by optical blending, no correct depth occlusion can be produced. While this may not be a problem for system control elements such as menus, objects in the scene are difficult to manipulate without correct occlusion. Nevertheless, for simple interaction schemes that do not rely on depth-correct occlusion optical superposition is easy to accomplish on a current PC system using a dual-GPU setup. This method is also related and was inspired by *Computer Graphics Optique* [Majumder and Welch 2001].

Digital composition involves merging of multiple images, which are then displayed using a single graphics output. In contrast to optical superposition this approach supports not only projection-based systems but also conventional monitor displays or HMDs. The mechanism as such is straightforward. Every time the slow rendering process finishes a frame the color and depth information of that image is transferred to the fast rendering process. There it is used at the beginning of each frame to setup the frame buffer. Then the scene parts relevant to user interaction are rendered afterward. Since the depth information from the slow image's content is already available correct depth occlusion is guaranteed.

### 2.2 Multi-Frame Rate Rendering

Multi-frame rate rendering is the process of producing images for a multi-frame rate display. Independently of the chosen display method, optical superposition or digital composition, the scene must be divided *somehow* in parts relevant for the interaction and the rest. This division however is not static. User interaction may affect any object in the scene, i. e. objects must able to migrate from one rendering process to the other upon request. Also, when combining resulting images from the divided scene parts the displayed image should be as much as possible similar to rendering the whole scene in a single process.

An easy way to divide the scene for the participating render processes would be the use of object lists. User input may affect any object by selection and release. Once an object is selected it has to migrate to the object list relevant for interaction and upon release it is migrated back to the object list containing the rest of the scene. This allows for implementations running in a single process on the same host machine as well as in distributed setups. However, most of the current infrastructure for visualization and VR systems is build upon a scene graph API. Springer et al. [2007] describe a mechanism where the assignment of an object to a render process is achieved by comparing node masks on objects and traversal masks on the render traversal process. Scene graphs are used to describe the structure of a scene [Clark 1976]. As such they do not prescribe a particular render method. The standard method in a scene graph environment for creating images is to traverse the graph and accumulate necessary information for a low-level rendering API (e. g. OpenGL). During traversal the traversal mask of the render process is tested against the node mask of each scene graph node visited, usually by a bit-wise `AND` operation, and upon evaluating to `true` the traversal descends further down. If the test evaluates to `false` the node and its sub-graph are excluded from further consideration. By assigning disjoint traversal masks for the fast and slow render processes changes to node masks, describing selection and release, will automatically assign objects to the appropriate render process. Again, this object migration method is also suitable for both single process as well as distributed rendering setups provided the changes in the scene graph are communicated across the participating cluster nodes by the scene graph API (e. g. OpenSG [Reiners 2002]) or VR system (e. g. Avango [Tramberend 2003]).

Multi-frame rate rendering by optical-superposition display simply uses the described object migration method for deciding to which render process each object currently belongs to. Each render process then produces its own image and the final image composition is achieved by overlapping the images of two projectors.

In the case of a digital composition display for multi-frame rate rendering the color and depth information from the slow render process must be transferred to the fast render process. This may be achieved in several ways: by network transfer between nodes in a graphics cluster, by local network device transfer between processes on the same host but running on different graphics devices, or by host memory transfer between either processes or within the same process [Springer et al. 2008]. Also, Springer et al. [2008] note that reading and writing of frame buffer sized image data from current graphics devices is at least one magnitude faster than transferring the same data on current Gigabit network configurations. Even faster transfer times can be expected if the image data could be transferred from one graphics device to the other over a dedicated link. Unfortunately this support does not yet exist.

### 2.3 Temporal Artifacts

The image creation by multiple asynchronously running render processes may create artifacts in the final image that can be perceived by the user. The problem is a result of combining different temporal samplings of a scene through the participating render processes. For example *popping artifacts* may occur during object migration [Springer et al. 2008]. Depending on frame rate differences as well as the latency between the slow and the fast render processes the situation can be dealt with in several ways. Springer et al. [2008] describe a solution for ray-based selection. Instead of migrating objects upon actual selection or release an over status similar to 2D GUIs is used. Upon intersecting an object the migration of that object is triggered by node mask changes. The same process is executed in reverse once the object is no longer intersected by the ray. Entering or leaving an object with the selection ray does not actually select or release the object but changes only its status as being relevant for interaction, i. e. its render process assignment. Because it usually takes some time from the enter event to the actual selection, this time is in almost all cases sufficient to update the render processes, transfer the slow image content to the fast render process and incorporate it. This heuristic works well up to where the update rate differences between the render processes become very large. To ensure consistency between the participating render processes Springer et al. [2008] also describe a state management approach that additionally sends the object identification of all objects contained in the current image from the slow rendering process. This information is used in the fast render process to decide if a certain object encountered during render traversal must be included in the final image or not. This state management also works in a conservative setup where pressing a button would actually trigger the selection process. The result of this simple state management is that temporal artifacts from object migration can be almost always avoided except for the case where users change from over status to selection and object movement so quickly that the new image from the slow render process has not yet arrived. In this case the object might be shown twice for a short amount of time. It is important to note that there is still no synchronization between the render processes necessary, which would inevitably stall the rendering and could potentially degrade system responsiveness and therefore interaction fidelity.

Object manipulation may also trigger changes that affect the scene as a whole (e. g. viewpoint changes or changing light parameters). Certain global effects can be properly handled within the multi-frame rate framework. For example changes in lighting parameters can be evaluated in the fast render process if the slow render process transfers the necessary information for final lighting instead of an already finished image. Springer et al. [2008] describe a variant of deferred shading [Deering et al. 1988; Hargreaves and Harris 2004] where the slow render process generates per-pixel information for normal, depth, material, etc. The fast render process combines this information with the lighting stage allowing fast user interactions with lights (e. g. re-positioning lights by manipulating a proxy geometry as well as changing lighting parameters or even an object's material). Since the original rendering information (color and depth) from the slow process is replaced by more parameters (normal, depth, material) more bandwidth is necessary for buffer transfer. Current network technology, such as Gigabit Ethernet, will introduce too much lag to the buffer transfer, so this method is practically limited to standalone multi-GPU setups. Also, computational load is shifted from the slow to the fast render process which may lead to an increase of frame time for rendering the interaction-related image. At the expense of one additional buffer to be transferred, containing an image of the fully shaded scene by any non-interactive light, this computational shift can be re-balanced by the application as needed.

## 3 Discussion and Future Work

Systems based on multi-frame rate rendering and display purposely create a highly unbalanced load on the image generators to improve the interactivity of object manipulations while maintaining most of the overall visual quality.

A user study by Springer et al. [2007] compared task performance for a 3D placement task for single-frame rate rendering at 10 Hz and 30 Hz with multi-frame rate rendering at 10/30 Hz, i. e.10 Hz for the slow and 30 Hz for the fast render process. It showed that task performance for multi-frame rate rendering using digital composition was almost as good as single-frame rate rendering at 30 Hz. While the single-frame rate setup also updated head-tracking data at 30 Hz the multi-frame rate setup updated it at 10 Hz. Although users noticed this lower update frequency it did not degrade their task performance. This observation is in tune with the findings by Watson et al. [1998]. On the other hand, multi-frame rate rendering using optical superposition exhibited a significant lower task performance than single-frame rate rendering at 10 Hz, which in turn has a lower task performance than single-frame rate rendering at 30 Hz as expected. This confirms that the missing depth occlusion of the optical superposition display method poses a serious problem for 3D placement tasks.

Multi-frame rate systems can be build from a variety of existing technology. By using known features and behavior of existing scene graph APIs multi-frame rate rendering is also supported for a variety of hardware setups. It is also conceivable to combine multi-frame rate rendering using digital composition with other techniques such as cluster rendering. Here, either the fast or the slow render process, or even both, may consist of a graphics cluster of their own. The possibility of dynamically re-assigning graphics processors between the sets of fast and slow clients may provide great potential to trade interactivity for overall rendering performance on demand. Future graphics hardware may provide control of frame buffer transfer between interconnected devices via dedicated links. The achievable bandwidth would be far greater than using intermediate host memory. Even the use of a single graphics device is supported if GPUs provide an efficient render task scheduling mechanism. In such an interleaved mechanism the remaining time of a fast frame could be used for rendering parts of the slow frame's content. Over several fast frames the slow frame's image would emerge and be ready for use on the GPU for the next fast frame. This approach enables the use of multi-frame rate rendering on low-end graphics systems such as laptops, mobile phones, and PDAs.

There is a potential problem with our proposed approach with respect to navigation. Decoupling the image generation rate from the interaction response rate will inevitably create temporally inconsistent images. Due to potentially large differences in update rates the partial images are created at different times and using different states from the underlying data as well as user input. In setups where no viewpoint changes occur this rarely leads to visible artifacts [Springer et al. 2008] but incorporating support for navigation is difficult. Since the update rate for interaction happens in the fast rendering process it is tempting to incorporate viewpoint changes here as well. The problem with this approach is that the high-quality image for the non-interactive scene parts will be then rendered from a different perspective; even if users are willing to cope with that in a monoscopic setup they will certainly not in a stereoscopic setup. Coupling the navigation update to the image update for the non-interactive scene parts is conservative but safe. Upon merging the images the faster interaction update process will use the viewpoint for generating the slower image for as long as no new image for the non-interactive scene parts has been created. This way users perceive a perspectively correct and coherent final image but with an navigation update rate as good as the update rate for the slow image part. One solution for the navigation problem is to predict viewpoint changes and use depth-image warping, an image-based rendering technique to generate new views from reference images considering per-pixel color and depth information (e. g. [Mark et al. 1997]).

An interesting question that frequently emerges during discussion concerns the relationship between the update rates of the slow and fast render processes. We need to investigate if update ratios exist that work better than others. Also, the influence of buffer-transfer latency from the slow to the interactive render process needs to be explored.

Multi-frame rate systems promise high visual quality and improved interaction fidelity at the expense of a few, often imperceptible visual artifacts. We argue that multi-frame rate capabilities should be built into most virtual reality systems to increase the acceptance of 3D user interfaces, which can now run at 30 Hz to 60 Hz in most cases.

## References

J. Airey, J. Rohlf, and P. Frederick. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. In *SI3D '90: Proceedings of the 1990 Symposium on Interactive 3D Graphics*, pages 41–50. ACM, 1990.

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

S. Bryson. Implementing Virtual Reality. In *Proceedings of ACM SIGGRAPH 93*, Computer Graphics Proceedings, Annual Conference Series, pages 1.1.1–1.6.6, 16.1–16.12. ACM, 1993.

S. K. Card, G. G. Robertson, and J. D. Mackinlay. The Information Visualizer, an Information Workspace. In *CHI '91: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 181–186. ACM, 1991.

J. H. Clark. Hierarchical Geometric Models for Visible Surface Algorithms. *Commun. ACM*, 19(10):547–554, 1976.

A. Dayal, C. Woolley, B. Watson, and D. Luebke. Adaptive Frameless Rendering. In *Proceedings of the 2005 Eurographics Symposium on Rendering*, Rendering Techniques 2005, pages 265–275. Eurographics, 2005.

M. Deering, S. Winner, B. Schediwy, C. Duffy, and N. Hunt. The Triangle Processor and Normal Vector Shader: A VLSI System for High Performance Graphics. In *Computer Graphics (Proceedings of ACM SIGGRAPH 88)*, volume 22(4), pages 21–30. ACM, 1988.

S. Hargreaves and M. Harris. Deferred Shading. URL http://developer.nvidia.com/object/6800_leagues_deferred_shading.html. 2004.

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.

W. R. Mark, L. McMillan, and G. Bishop. Post-Rendering 3D Warping. In *SI3D '97: Proceedings of the 1997 Symposium on Interactive 3D Graphics*, pages 7–16. ACM, 1997.

M. D. McKenna and D. Zeltzer. Three Dimensional Visual Display Systems for Virtual Environments. *Presence: Teleoperators & Virtual Environments*, 1(4):421–458, 1992.

R. Pausch. Virtual Reality on Five Dollars a Day. In *CHI '91: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 265–270. ACM, 1991.

M. Reddy. The Effects of Low Frame Rate on a Measure for User Performance in Virtual Environments. Technical Report ECS-CSG-36-97, University of Edinburgh, Department of Computer Science, 1997.

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.

J. P. Springer, S. Beck, F. Weiszig, D. Reiners, and B. Froehlich. Multi-Frame Rate Rendering and Display. In *Proceedings IEEE Virtual Reality 2007 Conference*, pages 195–202. IEEE, 2007.

J. P. Springer, C. Lux, D. Reiners, and B. Froehlich. Advanced Multi-Frame Rate Rendering Techniques. In *Proceedings IEEE Virtual Reality 2008 Conference*. IEEE, 2008. accepted.

G. Tharp, A. Liu, L. French, and L. Stark. Timing Considerations of Helmet Mounted Display Performance. *Proceedings of SPIE*, 1666:570–576, 1992.

H. Tramberend. *Avocado: A Distributed Virtual Reality Framework*. PhD thesis, Universität Bielefeld, Technische Fakultät, 2003.

C. Ware and R. Balakrishnan. Reaching for Objects in VR Displays: Lag and Frame Rate. *Trans. on Computer-Human Interaction*, 1 (4):331–356, 1994.

B. Watson, N. Walker, W. Ribarsky, and V. Spaulding. Effects of Variation in System Responsiveness on User Performance in Virtual Environments. *Human Factors*, 40(3):403–414, 1998.

B. A. Watson, D. Luebke, and A. Dayal. Breaking the Frame: A New Approach to Temporal Sampling. Technical Sketch, ACM SIGGRAPH, 2002.

C. D. Wickens. *Engineering Psychology and Human Performance*. HarperCollins, New York, NY, USA, 2nd edition, 1992.

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