Interruptible Rendering

J. Cliff Woolley
University of Virginia
jwoolley@virginia.edu

David Luebke
University of Virginia
luebke@virginia.edu

Ben Watson
Northwestern University
watsonb@cs.nwu.edu

Abstract

Interruptible rendering is a novel approach to the fidelity-versus-performance tradeoff ubiquitous in real-time rendering. Interruptible rendering unifies *spatial error*, caused by rendering coarse approximations for speed, and *temporal error*, caused by the delay imposed by rendering, into a single image-space error metric. The heart of this approach is a progressive rendering framework that renders a coarse image into the back buffer and continuously refines it, while tracking the temporal error. When the temporal error exceeds the spatial error caused by coarse rendering, further refinement is pointless and the image is displayed. We discuss the requirements for a rendering algorithm to be suitable for interruptible use, and describe one such algorithm based on hierarchical splatting. Interruptible rendering provides a low-latency, self-tuning approach to interactive rendering. Interestingly, it also leads to a "one-and-a-half buffered" approach that renders sometimes to the back buffer and sometimes to the front buffer.

1 Overview

Computer graphics practitioners have long recognized the tradeoff between spatial detail and frame rates. Given additional resources, which is more important: more detailed models or higher frame rates? Answers to date have been ad hoc. Conventional wisdom simply dictates that high, constant frame rates are best; applications tend to target 30-60 Hz frame rates.

Interruptible rendering provides a principled approach to this tradeoff by unifying spatial and temporal error in a single metric: screenspace distance (Figure 1). A coarse image is generated and continuously refined, decreasing spatial error while temporal error grows with the passage of time. When temporal error exceeds spatial error, there is no longer any reason to refine further: any improvement to the appearance of objects in the image will be overwhelmed by their wrong position and/or size. In other words, when the error due to the image being *late* is greater than the error due to the image being *coarse*, taking time for further refinement is pointless. The front and back buffers are then swapped and rendering begins again into the back buffer for the most recent viewpoint. A system which minimizes combined spatial-temporal error quite intuitively results in coarse, high frame rate display when input is changing rapidly, and finely detailed, low frame rate display when input is static.

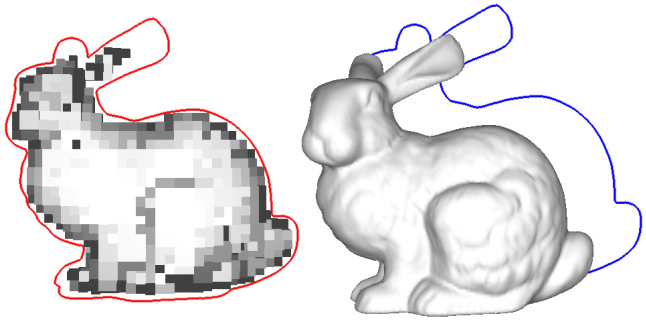

Figure 0: Spatial and temporal error. The ideal instantaneous image that reflects the most up-to-date input is shown in silhouette (colored outlines). The left image is coarsely sampled, representing some spatial error. The right image is finely sampled, but as a result is quite late. The resulting temporal error, manifested as image-space distance from the ideal instantaneous image, is larger than the spatial error of the coarse approximation. In this sense the coarsely sampled bunny actually represents lower visual error.

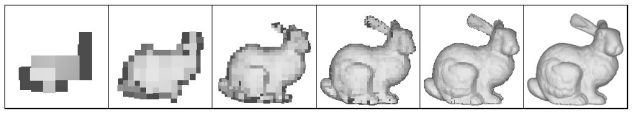

Figure 2: A sequence of progressively refined, splatted volumetric models. Each finer model contains and covers all coarser models.

Interruptible Image Generation: As the name implies, interruptible rendering requires an interruptible image generation process – interruptible because a sudden motion by the user can drive up temporal error at any time. The back buffer should always contain a complete image ready to be displayed, with the rendering process incrementally refining the image until further refinement is pointless. This requires an interruptible, progressive image generation process. Depth buffered rendering requires a continuous LOD algorithm that preserves *containment* – refined versions of an object completely contain more simplified versions already rendered. We use a hierarchical splatting algorithm similar to QSplat [Rusinkiewicz 2000], but using *bounded* rather bounding volumes to ensure containment (Figure 2). Other possibilities include Sander's *progressive hull* structure [Sander 2000] or adaptive ray tracing.

Measuring Temporal Error: We currently measure temporal error by tracking the corners of an object's bounding box to quickly estimate how far it may have moved since rendering began on the current image. For a more accurate, scalable approach we plan to investigate randomized algorithms that avoid bias by continuously picking a random set of vertices to track from the model.

2 Benefits and Implications

Interruptible rendering is by nature a low-latency algorithm: temporal error is continuously monitored (we check once per millisecond in our current implementation), so the system can respond almost instantly if the user makes a sudden move that invalidates the current image. By addressing temporal error, interruptible rendering also intrinsically tunes spatial detail to the rendering platform performance. On a low-end platform, less detail can be drawn before temporal error dominates spatial error and a new image is started. Thus when input is changing rapidly, the system will render less detail than a high-end platform but be just as responsive. Finally, an interesting corollary of unifying spatial and temporal error is that sometimes the image being refined in the back buffer may achieve lower combined error than the image in the front buffer. This implies that we should not continue to display the front image, but instead should swap buffers and continue refining the image *in the front buffer*.

References

RUSINKIEWICZ, S., AND LEVOY, M. 2000. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of ACM SIGGRAPH 2000*.

SANDER, P., GU, X., GORTLER, S., HOPPE, H., SNYDER, J. 2000. Silhouette Clipping. In *Proceedings of ACM SIGGRAPH 2000*.

Additional Material for Reviewers

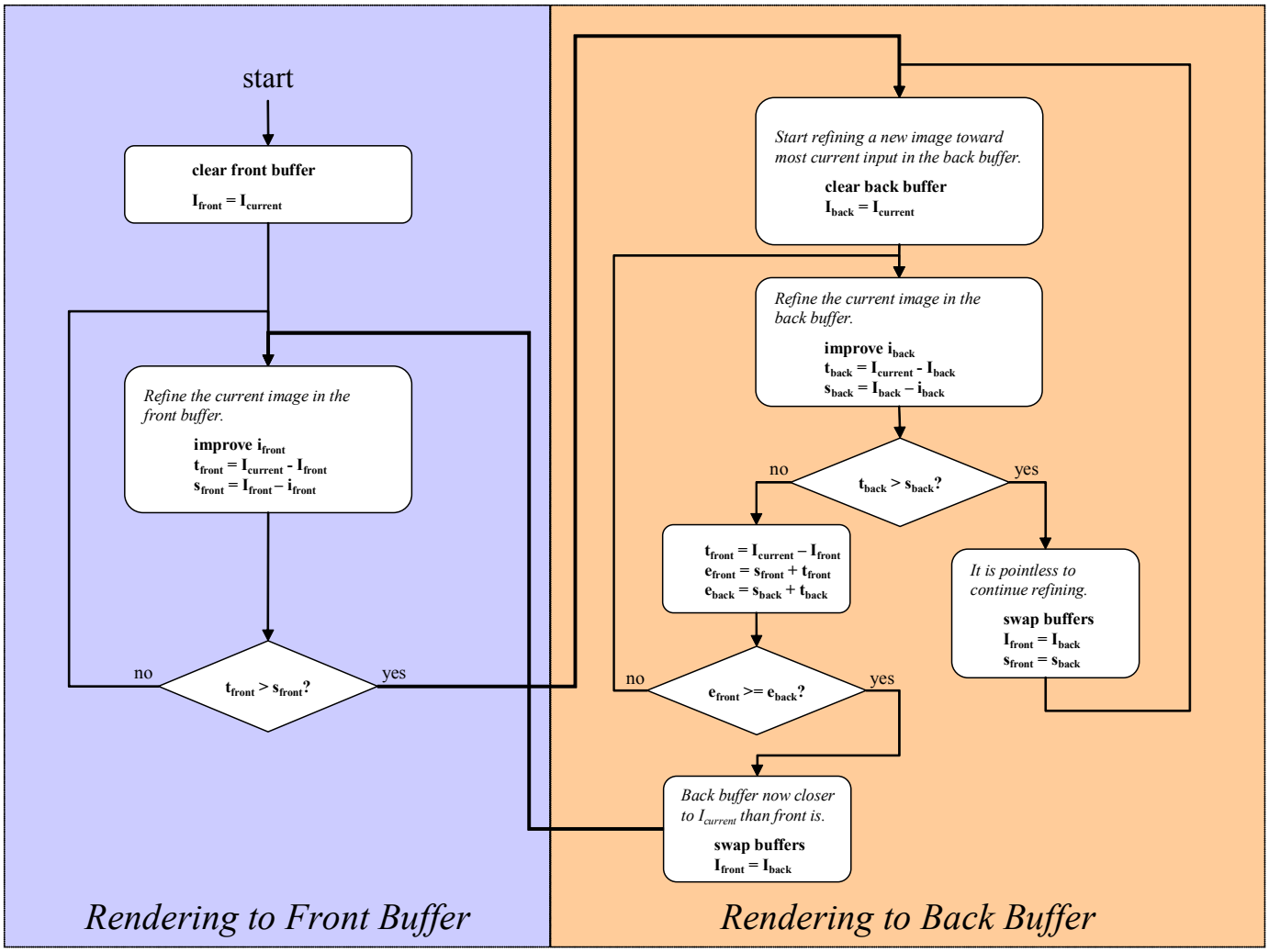

Figure 3: The interruptible rendering state machine. $I_{current}$ represents the instantaneous image, I_{back} and I_{front} represent the images currently in the front and back buffers, and s and t represent estimates of the spatial and temporal error. An interesting consequence of considering dynamic visual error in a framed context is that rendering sometimes begins refining in the back buffer and switches to refining in the front buffer.

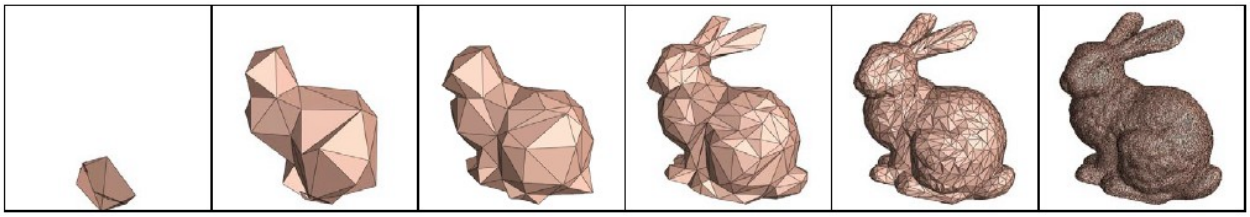

Figure 4: A sequence of contained progressively refined polygonal models (a *progressive hull*) suitable for interruptible rendering. Each finer model contains and covers the preceding coarser models. This figure is from Sander [2000].

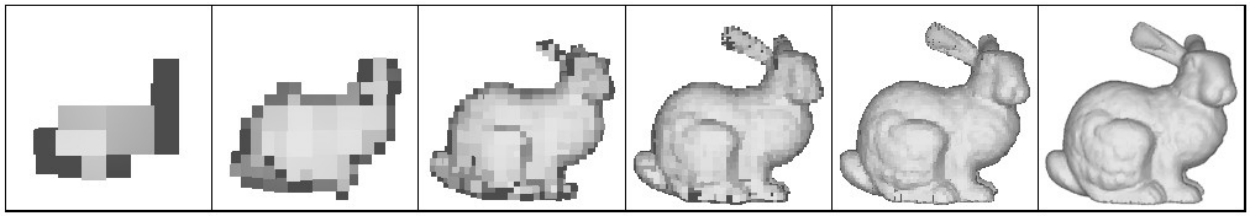

Figure 5: (larger version of Figure 2) A sequence of progressively refined, splatted volumetric models. Each finer model contains and covers the preceding coarser models.

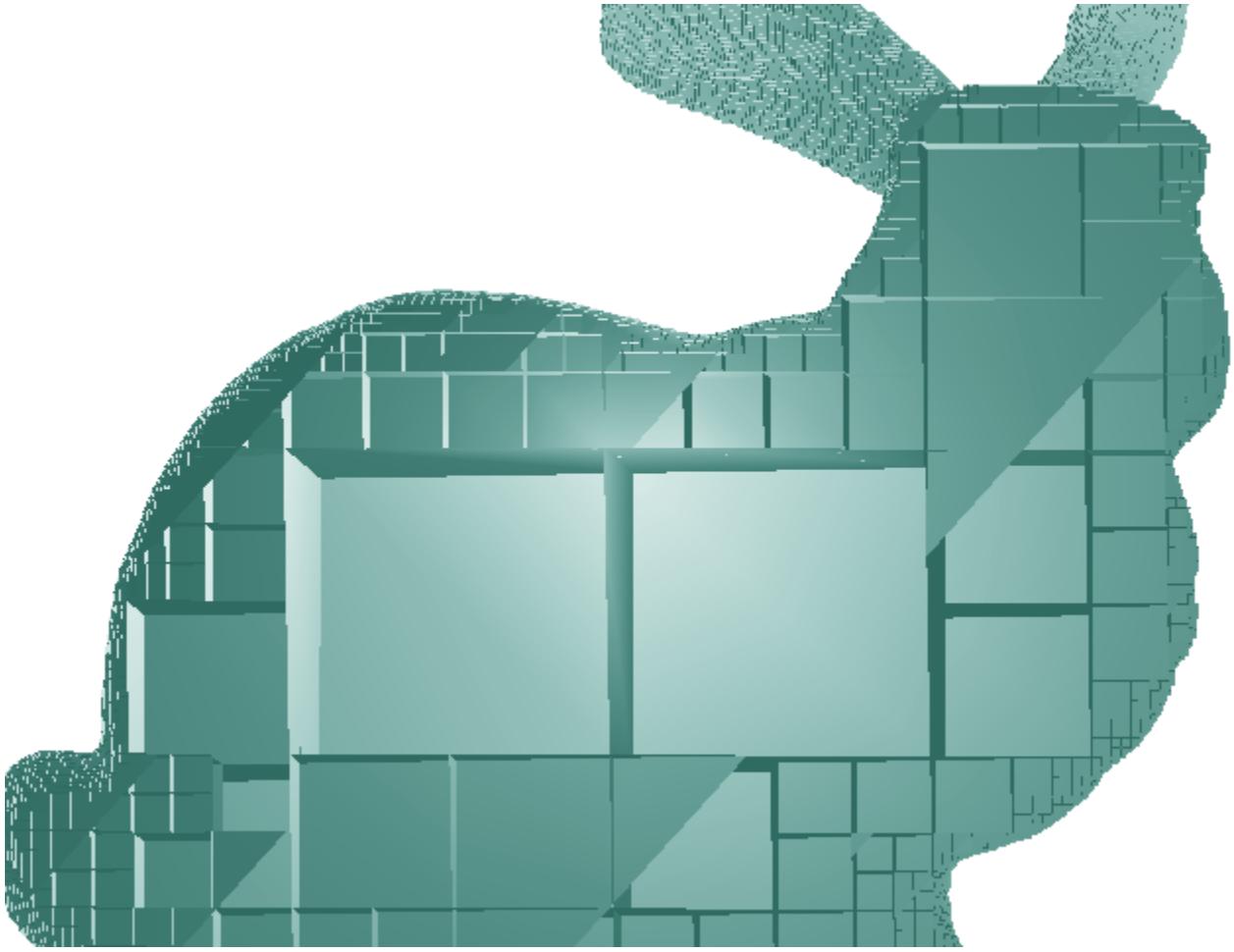

Figure 6: A cutaway of the full-resolution bunny in our system illustrating the multiple levels of the octree used to generate splats. In the interruptible system, the higher levels of the tree (the bigger nodes in the middle) are rendered first as simple splats, and then successive lower levels (progressively smaller nodes) are rendered on top of them in a streaming fashion. This process can be interrupted at any time.