Hybrid Rendering: Enabling Interactivity in a Distributed Post-Processing Environment

Markus Flatken and Andreas Gerndt

German Aerospace Center, Simulation and Software Technology,
Lilienthalplatz 7, 38108 Braunschweig, Germany
{markus.flatken,andreas.gerndt}@dlr.de
http://www.dlr.de/sc/

1 Introduction

The ever increasing compute capacity of high performance computing (HPC) systems enables scientists to simulate and explore physical phenomena with an enormous spatial and temporal accuracy. On the other hand, this accuracy leads to datasets of many terabytes, petabytes, and even exabytes envisioning the upcoming exascale area projected for 2018 [1]. To understand complex physical coherences behind such a simulation, an efficient analysis and visualization is essential but also difficult, since the challenges concern all stages of the visualization pipeline [2]. With this presentation we set the focus on distributed and hybrid rendering.

2 Parallel Rendering

The amount of data to be rendered from a large-scale simulation on a supercomputer can easily exceed the required main memory size and rendering performance of a single machine. Therefore, parallel rendering techniques have been developed. Using these techniques each processor only renders a subset of the total data. Molnar et al. have given an early classification of these parallel rendering techniques [3]. The sort-last rendering approach e.g. distributes the data over a defined number of rendering machines. Each of these rendering machines determines whether their chunk of data is projected onto screen. In that case the rendering machine calculates the correct color and depth value per pixel and stores them in buffers. In a conclusive step all generated buffers are composed via z-value comparison to a final image (depicted in the middle of Figure 1). The major advantage is that sort-last rendering scales with respect to the size of the data. The achieved frame rate, however, is limited by network bandwidth due to costly communication for image composition and therefore disrupts user interaction. Thus, decoupling rendering from the display stage is useful.

3 Hybrid Rendering

Hybrid rendering (depicted in Figure 1) aims at splitting render load to local and remote resources [5]. Remote resources are generally supercomputers or graphics

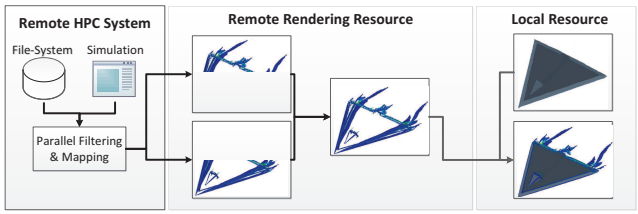
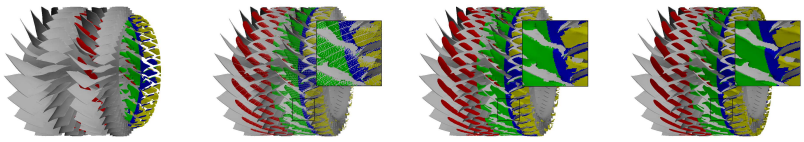


Fig. 1. Remotely rendered images are transfered to the local resource where they are combined with the local context image according to the pixels depth values.

clusters while local resources may be simple workstations. Complex features of the simulation are rendered remotely while the context geometry, e.g. the shape of an airplane, is rendered locally to guarantee interactivity. Therefore, images are generated with different frame rates on these resources. Since remote images are generated with lower frame rates and cause latency, they might not fit with the current local view (depicted in Figure 2(a)). To avoid this problem image-based rendering (IBR) techniques are exploited. With these IBR techniques it's possible to align an obsolete 2.5D remote image to the current local view. A pixel is transformed from screen space into object space by calculating its inverse projection. This 3D point is then transformed according to the current local projection. However, this approach leads to artifacts inside the presentation depicted in Figure 2(b). To deal with these artifacts we use an adaptive point size adjustment. The required point size is determined during point transformations and depends on the derivation in x- and y-axis. Therefore, we calculate the screen-space distance to the transformed neighbor pixels with the same z-values. This adaptive point size rendering approach successfully fills holes inside a surface (depicted in Figure 2(c)). Boundaries, however, tend to be a bit blurred compared to correct rendering in Figure 2(d). Nevertheless, this hybrid rendering approach combined with IBR hides the varying frame rates and latencies and thus guarantees interactivity on the local resource. This approach is also suitable for tiled displays. To further optimize communication and latency, the image stream can be encoded using video and depth compression algorithms [6].



(a) rotated local image (b) fixed point-size adjustment (c) adaptive point-size adjustment (d) correct remote image

Fig. 2. In the left image the user is rotating the scene leading to diverged images. The second image shows the re-projected version with fixed point sizes. The third image is generated with adaptive point sizes and the right image depicts a correct rendering. [4]

References

1. Ashby, S. et al.:The opportunities and challenges of exascale computing. Summary report of the advanced scientific computing advisory computing subcommittee, U.S Department of Energy (2010)
2. Moreland, K.: A Survey of Visualization Pipelines. IEEE Transactions on Visualization and Computer Graphics. vol. 19, 367–378 (2013)
3. Molnar, S., Cox, M., Ellsworth, D., Fuchs, H.: A sorting classification of parallel rendering. IEEE Comput. Graph. Appl. vol. 14, 23–32 (1994)
4. Wagner, C., Flatken, M., Chen, F., Gerndt, A., Hansen, C., Hagen, H.: Interactive Hybrid Remote Rendering for Multi-pipe Powerwall Systems. In: VR/AR, pp. 155–166. Shaker, Herzogenrath (2012)
5. Noguera, J. M., Segura, R. J., Ogyar, C. J., Joan-Arinyo, R.: Navigating large terrains using commodity mobile devices. Computers and Geosciences. vol. 37, 1218–1233 (2011)
6. Pajak, D., Herzog, R., Eisemann, E., Myszkowski, K., Seidel, H.: Scalable Remote Rendering with Depth and Motion-flow Augmented Streaming. Computer Graphics Forum. vol 30, 415–424 (2011)