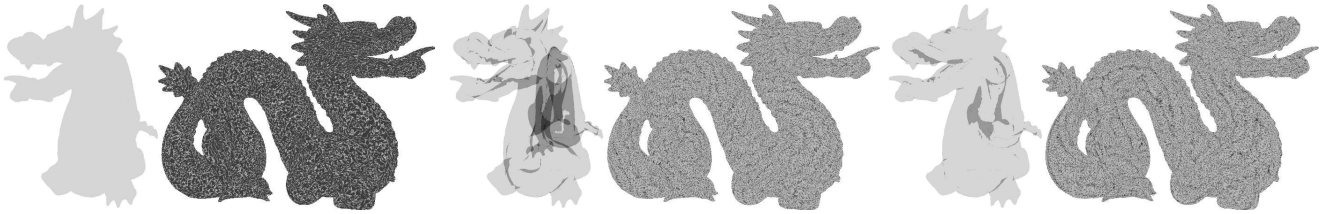
Triangle Order Optimization for Graphics Hardware Computation Culling

Diego Nehab
Princeton University

Joshua Barczak
University of Maryland, Baltimore County

Pedro V. Sander
ATI Research



(a) front-to-back (b) vertex cache optimized (c) our method

Figure 1: Illustration of overdraw and vertex cache efficiency. In the front views, darker regions represent high-overdraw. In the side views, darker regions represent cache misses.

Abstract

In Nehab et al. [2006], we describe an automatic preprocessing algorithm that reorders triangles in a mesh so as to enable the graphics hardware to efficiently cull both vertex and pixel processing at rendering time. Our method brings the overdraw rates of a wide range of models close to that of front-to-back order, while preserving state-of-the-art vertex cache performance. This results in higher frame rates for pixel-bound applications with no penalty to vertex-bound applications.

1 Exposition

In real-time rendering, vertex processing costs can be greatly reduced by mesh locality optimizations [Hoppe 1999]. On the other hand, pixel processing can be minimized by drawing the triangles in front-to-back order (i.e. eliminating overdraw). These two goals can be contradictory. Besides being fundamentally view-dependent (thus creating run-time overhead), depth sorting can ruin vertex cache performance (figure 1a). Conversely, mesh locality optimizations can lead to high overdraw under certain viewpoints (figure 1b). Our goal is to find a view-independent order that minimizes overdraw while preserving mesh locality optimizations (figure 1c).

Due to back-face culling, the relative order of any two triangles is irrelevant if their normals point in opposite directions (as far as overdraw is concerned). This freedom is what allows us to find an ordering that meets our goals. Our triangle order optimization algorithm proceeds in three steps.

Clustering First, we cluster the model into planar regions. We adapted the method of Sander et al. [2003] to produce clusters suitable for our application. More specifically, since we are not concerned with boundary compactness, we use a metric solely based on normals. We favor planar clusters because those are less likely to self-occlude. Resulting clusters are optimized individually for mesh locality, with the method described in Hoppe [1999].

Partial order graph Given the clustered mesh, we generate a partial ordering graph that captures the relative overdraw between every pair of clusters. For each pair, we determine whether to add an arc between them and, if so, the direction and weight it should have. Consider clusters a and b. We add an arc from a to b if drawing a before b generates less pixels than the opposite order, when rendered under a variety of viewpoints. The weight of the arc is the net difference in the number of generated pixels. The rationale is that if a can occlude large portions of b, then it should be rendered first. If only a small part of b can be occluded, the relative order matters less.

Ordering the clusters Finally, we produce a cluster ordering from the graph that attempts to minimize total overdraw. If the graph has no cycles, a Topological Sort solves the ordering problem in $O(|V| + |E|)$. However, some models are likely to produce graphs with cycles. In those cases, we are interested in finding the ordering that violates the minimum number of pairwise orderings, or rather the ordering that minimizes the sum of the weights of all violations. This problem is known as the Minimum Feedback Arc Set problem, and it is NP-complete. A greedy heuristic which we have found to work well in practice is described in Skiena [1997]. It has the advantages of being easy to implement, efficient to execute (it also runs in $O(|V|+|E|)$), and of agreeing with the Topological Sort whenever possible.

2 Results and conclusion

Our results indicate that our clustering has little or no impact on vertex cache efficiency. Furthermore, our final ordering significantly reduces overdraw when compared with global mesh locality optimization (figure 1). This results in substantial improvements in rendering time for pixel bound applications.

In addition, we have observed a considerable reduction in the variance of the overdraw rates with regard to viewpoint change (when compared to the global mesh locality optimization). This leads to more consistent frame rates.

References

HOPPE, H. 1999. Optimization of mesh locality for transparent vertex caching. In *Proc. of ACM SIGGRAPH 99*, ACM Press, pages 269–276.

NEHAB, D., BARCZAK, J., and SANDER, P. V. 2006. Triangle order optimization for graphics hardware computation culling. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pages 207–211.

SANDER, P. V., WOOD, Z. J., GORTLER, S. J., SNYDER, J., and HOPPE, H. 2003. Multi-chart geometry images. In *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, Eurographics Association, pages 146–155.

SKIENA, S. S. 1997. *The Algorithm Design Manual*. Springer.