

Occlusion culling in z-buffer systems has drawn a lot of attention over the last few years.<sup>1,3</sup> Due to the high computational costs of occlusion queries, hardware support is required to accelerate interactive rendering. This requires mechanisms that fit nicely into an available graphics subsystem without greedily consuming chip real estate (one reason why hardware support for occlusion culling is rare).

One of the few available dedicated hardware mechanisms for occlusion culling renders objects in an occlusion mode and sets a flag in case parts of the geometry would have altered the frame buffer.<sup>2</sup> Using a scene hierarchy, bounding primitives can be tested to discard the contained geometry if the flag has not been set. Hence, the bandwidth of the front bus (geometry transfer) and geometry transformation can be significantly reduced, while sacrificing rendering bandwidth during the occlusion test itself. Despite its efficiency, the main drawback of this approach is the costly query, due to latency in the graphics subsystem, which allows only a limited amount of visibility queries performed per frame, and therefore limits the overall possible efficiency.

Visibility-driven rasterization represents a new scheme, which extends rasterization in two ways. First, during frame generation, screen-space-oriented visibility information is generated several times and stored within the rasterizer in a visibility mask. Second, if contributions to the visibility mask were detected, the established visibility information is exploited to discard all triangles and groups of pixels of subsequently rendered polygons that belong to non-visible areas (see Figure 1(a,b)).

Visibility-driven rasterization achieves significantly higher cull-rates than the existing hardware mechanisms. Instead of having a single flag indicating only overall visibility (occlusion culling prior to geometry transformation), we use a frame buffer-oriented visibility mask containing a flag for each tile of pixels to enable culling prior to and during rasterization, in addition to previous culling stages. This tremendously reduces the cost of the unsuccessful visibility queries, by exploiting information during rendering of the geometry, which is at least partially visible. Consequently, more visibility queries can be performed per frame, increasing the overall frame-rate. Visibility-driven rasterization will gain even more importance using multi-texturing, which requires iterating over pixels multiple times, wasting valuable rendering bandwidth while processing occluded pixels.

The performance of visibility-driven rasterization depends on tile size, which again depends on the size of triangles sent down the graphics subsystem. To overcome this dependency, we organize the visibility

mask as register file or on-chip SRAM of 16-bit entries. Hence, a natural tile hierarchy is obtained grouping 4x4 tiles (four 2x2 tiles) into each entry. By using basic bit masking operations (AND/OR), triangles residing across multiple occluded tiles can be easily culled. This reduces the influence of the actual tile size on the culling performance, allowing for a generic tile size selection. We evaluated the quantitative efficiency of visibility-driven rasterization for different tile sizes (extending the MESA 3D graphics library) applied to a set of four real-world scenes: A CAGD cotton picker (10M tris, see Figure 2), a CAGD cathedral (416K tris), a CAGD screwdriver (150K tris), and an extracted iso-surface of the ventricular system of the human brain (270K tris).

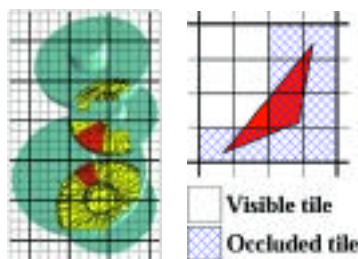
On average, best results could be obtained for tiles of 16x16 pixels while the peak culling performance varied from 8x8 to 32x32 pixels. The necessary visibility mask can be stored in as little as 4 Kbit of storage, which can easily be integrated on-chip. For the set of test scenes, this allows for additionally culling between 20 to 40 percent of the remaining triangles prior to rasterization and another 35 to 55 percent of all remaining pixels during rasterization. The latter additional pixel culling is achieved by discarding chunks of pixels of individual triangles that reside across multiple tiles, where at least one tile has been visible (see Figure 1(b)).

One of the advantages of visibility-driven rasterization is its compactness which, in contrast to other approaches, does not require storing depth values (32 bit), since it only contains visibility information. The depth test is inherent to the approach of testing bounding primitives first before rendering the contained geometry. Other approaches might achieve little higher cull-rate, but require major changes to the existing graphics subsystem and consume significantly more chip area to store depth values.

The measured results show that visibility-driven rasterization could extend real-time rendering performance of existing graphics subsystems. Last but not least, visibility-driven rasterization is an efficient culling scheme that fits nicely into the present state-of-the-art, stamp-based rasterization scheme, which is also tile oriented.

## References

- Greene, N. (1996). Hierarchical polygon tiling with coverage masks. In *Computer Graphics, Proc. of ACM SIGGRAPH 96* 65-74.
- Scott, N., Olsen, D., & Gannett, E. (1998, May). An overview of the VISUALIZE fx graphics accelerator hardware. *The Hewlett-Packard Journal*, 28-34.
- Zhang, H., Manocha, D., Hudson, T., & Hoff, K. E. (1997). Visibility culling using hierarchical occlusion maps. In *Computer Graphics, Proc. of ACM SIGGRAPH 97*, 77-88.



Visibility-driven rasterization: (a) Wheel hubs illustrating triangles that can be culled prior to rasterization. (b) Culling of chunks of pixels during rasterization



Cotton-picker dataset containing 10 million triangles.