

# Dynamic Adaptive Shadow Maps on Graphics Hardware (sketches\_0263)

Aaron Lefohn  
University of California, Davis

Robert Strzodka  
Caesar Research Institute, Bonn

Shubhbrata Sengupta  
University of California, Davis

John D. Owens  
University of California, Davis

Joe Kniss  
University of Utah

We present a novel implementation of adaptive shadow maps (ASMs) that performs all shadow lookups and scene analysis on the GPU, enabling interactive rendering with ASMs while moving both the light and camera. Adaptive shadow maps [Fernando et al. 2001] offer a rigorous solution to projective and perspective shadow map aliasing while maintaining the simplicity of a purely image-based technique. The complexity of the ASM data structure, however, has prevented full GPU-based implementations until now. Our approach uses an entirely GPU-based data structure and a blend of graphics and GPU stream programming. We support shadow map effective resolutions up to  $131,072^2$  and, unlike previous implementations, provide smooth transitions between resolution levels by trilinearly filtering (mipmapping) the shadow lookups.

## Implementation

Shadow maps, which are depth images rendered from the light position, offer an attractive solution to real-time shadowing because of their simplicity. Their use is plagued, however, by projective aliasing, perspective aliasing, and false self-shadowing [Wimmer et al. 2004]. ASMs nearly eliminate shadow map aliasing by ensuring that the projected area of a screen-space pixel into light space matches the shadow map sample area. Figure 1 and the included movie compare our ASM results to a traditional shadow map.

An ASM stores depth data in a quadtree of small shadow map pages. Our structure represents a quadtree using two GPU textures, representing a mipmap hierarchy of page tables and a physical memory buffer. The page tables adaptively map shadow coordinates into physical memory pages, and the mipmaps allow direct access to arbitrary resolution levels in the quadtree. We use the Glift template library for GPU data structures [Lefohn et al. 2005] to build the ASM. The library makes it possible to quickly build this structure and use it in a way similar to a conventional texture. Below is an example of a Cg shader that performs an ASM lookup:

```
float4 main( uniform VMem2D asm,  
            float3 shadowCoord ) : COLOR {  
    return asm.vTex2Ds( shadowCoord );  
}
```

Our ASM algorithm is faithful to Fernando et al.'s original approach [2001] and is summarized below.

```
refineASM {  
    AnalyzeScene : Identify shadow pixels with resol. mismatch  
    StreamCompaction : Pack these pixels into small stream  
    CpuReadback : Read refinement request stream  
    AllocPages : Draw new PTEs into mipmap page tables  
    CreatePages : Draw depth into ASM for each new page  
}
```

The algorithm begins by performing a scene analysis to determine which camera-space pixels require refinement. A pixel requires refinement if it lies on a shadow boundary and the required resolution is not in the current ASM. We use a Sobel edge detector to identify shadow boundaries and compute required resolutions using derivatives of the shadow coordinates. We then pack the pixels needing refinement into a small contiguous stream using the algorithm described by Horn et al. [2005]. This small image (usually tens of pixels) is read back to the CPU to initiate new shadow data generation. The CPU adds new shadow data to the ASM by first rendering new page allocations into the GPU-based page tables,

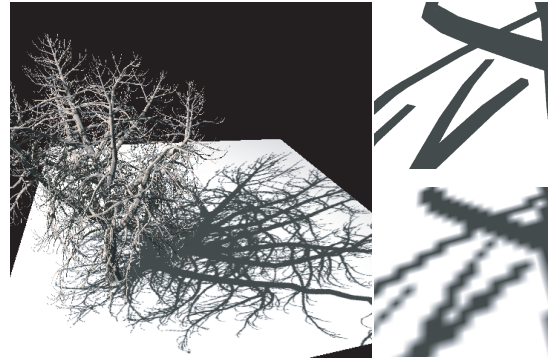


Figure 1: This GPU-based adaptive shadow map has an effective shadow map resolution of  $131,072^2$ , uses 37 MB of GPU memory, and supports trilinear (mipmapped) filtering. The detail at top right shows the ASM, and the detail at bottom right shows a  $2048^2$  standard shadow map.

then rendering the scene geometry from the light into the new physical pages. We repeat this refinement algorithm to convergence, but implementations could interrupt it to maintain a target frame rate.

## Performance Results

We implemented our ASM algorithm on an NVIDIA GeForce 6800 GT using a window size of  $512^2$ . For a 45k polygon model and effective shadow map resolution of  $131,072^2$ , our implementation achieves 13–16 frames per second while the camera is moving. We achieve 5–10 fps while interactively moving the light for the same model, thus rebuilding the entire ASM each frame. Our ASM lookup performance is between 73–91% of a traditional  $2048^2$  shadow map. The total frame rate is dominated by the stream compaction step of the refinement algorithm. This computation reduces CPU readback cost at the expense of  $\log n$  render passes, where  $n$  is the number of pixels in the camera image. Hardware support for read-modify-write programmable blending or variable-length output streams would make this operation possible with a single pass.

## Conclusions and Future Work

We show that adaptive shadow maps can be used at interactive rates on moderately complex models, making them an effective option for high-quality GPU shadows. We are currently extending our system to provide a guaranteed frame rate option (instead of guaranteed shadow quality) that limits the number of refinement requests serviced per frame.

## References

- FERNANDO, R., FERNANDEZ, S., BALA, K., AND GREENBERG, D. P. 2001. Adaptive shadow maps. In *Proceedings of ACM SIGGRAPH 2001*, Computer Graphics Proceedings, Annual Conference Series, 387–390.
- HORN, D. 2005. *GPU Gems 2: Stream Reduction Operations for GPGPU Applications*. Addison Wesley, Apr., ch. 36, 575–591.
- LEFOHN, A. E., KNISS, J., STRZODKA, R., SENGUPTA, S., AND OWENS, J. D. 2005. Glift: An abstraction for generic, efficient GPU data structures. *ACM Transactions on Graphics*. To appear.
- WIMMER, M., SCHERZER, D., AND PURGATHOFER, W. 2004. Light space perspective shadow maps. In *Eurographics Symposium on Rendering*, 143–151.