Implementing Efficient Virtual Shadow Maps for Many Lights

Ola Olsson, Erik Sintorn, Viktor Kämpe, Markus Billeter and Ulf Assarsson
Chalmers University of Technology

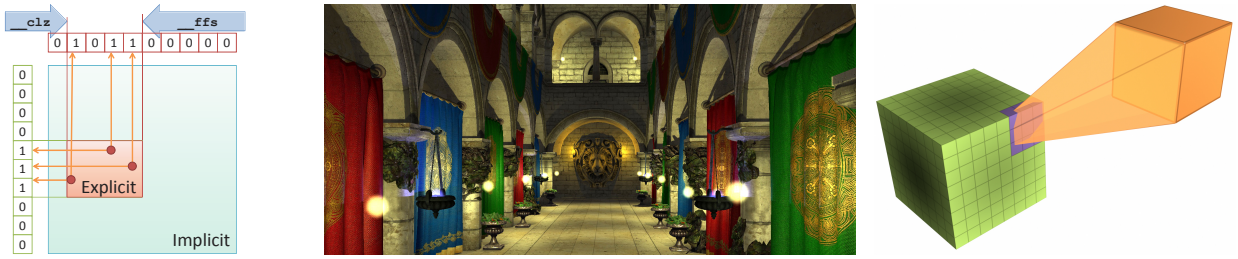

Figure 1: *View of the Crytek Sponza scene with 65 shadow-casting lights, rendered with the system in 16ms at a resolution of* 1920×1080 *on a Geforce Titan. Left: explicit cluster bounds using bitwise operations. Right: AABB projected onto the virtual cube map.*

1 Abstract

In the past few years, several techniques have been presented that enable real-time shading using many hundreds or thousands of lights [Harada et al. 2013]. However, only recently has a comprehensive study including shadows been presented by Olsson et al. [2014], where real-time performance is achieved for several hundred light sources with high quality and controllable memory footprint. The new algorithm uses many modern features of OpenGL and contains many design choices only described very briefly in the paper. We present additional details and focus on the practical implementation aspects of the system, in order to facilitate the implementation of the algorithm for the game development community.

Bounding Box Re-projection The quick re-projection of bounding boxes onto the virtual cube map surrounding the light is one of the corner stones of the algorithm. To compute the cube-face mask, virtual-page mask, and the projection map, we designed a coarse but very cheap method to project an AABB to a cube-map face, as shown in Listing 1. Applications already using the clustered shading algorithm, can potentially use this cheap re-projection for many other applications.

```
Rect xPlus(Aabb aabb)
{
  float2 one = { 1.0f, 1.0f };

  float rdMin = 1.0f / max(Epsilon, aabb.min.x);
  float rdMax = 1.0f / max(Epsilon, aabb.max.x);

  float sMin = min(-aabb.max.z * rdMin,
                   -aabb.max.z * rdMax);
  float sMax = max(-aabb.min.z * rdMin,
                   -aabb.min.z * rdMax);

  float tMin = min(-aabb.max.y * rdMin,
                   -aabb.max.y * rdMax);
  float tMax = max(-aabb.min.y * rdMin,
                   -aabb.min.y * rdMax);

  Rect r;
  r.min = clamp(float2(sMin, tMin), -one, one);
  r.max = clamp(float2(sMax, tMax), -one, one);
  return r;
}
```

Listing 1: *Bounding box projection on the* $+X$ *cube map face.*

Virtual Shadow Map Management We also give a practical overview of how we allocate and manage virtual shadow maps, presenting the new OpenGL 4.3 extension `ARB_sparse_texture`.

Resolution Selection The calculation of the correct resolution of each shadow map is illustrated with CUDA code samples to show how the atomic operations are used to establish the maximum needed resolution. Additionally, the implementation of the extension enabling several shadow maps per light is detailed.

Explicit Cluster Bounds The explicit cluster bounds calculation introduced in the paper is also presented with example code. This novel optimization is a useful technique that uses bitwise logic to pack several operations into one atomic instruction (Figure 1, left).

GPU-based Culling A crucial step in the algorithm is quickly culling shadow casting geometry. We therefore explain implementation details of the culling process and acceleration structure used.

OpenGL Extensions Achieving high performance when rendering to several hundred virtual cube maps requires careful attention to details. The paper shows that the combination of `glMultiDrawElementsIndirect`, layered rendering, and geometry shader routing is a viable solution. We present additional details to show how these steps are connected into a functioning system.

References

HARADA, T., MCKEE, J., AND YANG, J. C. 2013. Forward+: A step toward film-style shading in real time. In *GPU Pro 4: Advanced Rendering Techniques*, W. Engel, Ed. 115–134.

OLSSON, O., SINTORN, E., KÄMPE, V., BILLETER, M., AND ASSARSSON, U. 2014. Efficient virtual shadow maps for many lights. In *Proc. I3D '14*, ACM, 87–96.