Conservative Z-Prepass for Frustum-Traced Irregular Z-Buffers

Yusuke Tokuyoshi
SQUARE ENIX CO., LTD.

Tomohiro Mizokuchi
SQUARE ENIX CO., LTD.

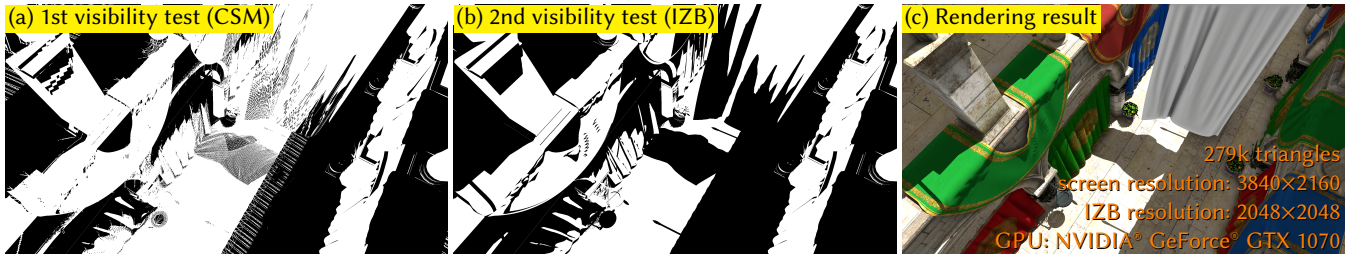

Figure 1: Visibility mask buffer (a, b) and rendering result (c) using a two-pass visibility test for hard shadows. The visibility is first roughly tested using a conservative shadow map (CSM) (a), and then an accurate visibility test using an IZB is performed only for the remaining shading points (b). Using this pipeline, the shadow performance is improved from 8.52 ms to 3.59 ms.

ABSTRACT

This paper presents a pipeline to accelerate frustum traced irregular z-buffers (IZBs). The IZB proposed by Wyman et al. is used to render accurate hard shadows for real-time applications such as video games, while it is expensive compared to shadow mapping. To improve the performance of hard shadows, we use a two-pass visibility test by integrating a *conservative shadow map* into the pipeline of the IZB. This paper also presents a more precise implementation of the conservative shadow map than the previous implementation. In our experiments for 4K screen resolution, the performance of the hard shadow computation is improved by more than double on average using the two-pass visibility test, though there is still room for optimization.

CCS CONCEPTS

• **Computing methodologies** → **Visibility**;

KEYWORDS

hard shadows, irregular z-buffers, conservative shadow maps

ACM Reference Format:
Yusuke Tokuyoshi and Tomohiro Mizokuchi. 2018. Conservative Z-Prepass for Frustum-Traced Irregular Z-Buffers. In *Proceedings of SIGGRAPH '18 Posters.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3230744.3230755

1 INTRODUCTION

An accurate hard shadow algorithm using frustum traced irregular z-buffers (IZBs) [Wyman et al. 2016] is used for real-time applications such as video games [Story and Wyman 2016]. In this paper, we present a pipeline to accelerate the IZB shadow algorithm. The IZB algorithm first constructs per-texel linked lists of shading points in light space, and then the visibility is tested for each node (i.e., shading point) in those lists using frustum tracing. This algorithm can be expensive when many shading points are projected into a texel of the IZB and are occluded by multiple triangles in light space. Therefore, we employ a two-pass visibility test using a *conservative shadow map* [Hertel et al. 2009] which stores a conservative depth to detect fully shadowed shading points. The conservative shadow map was developed to alleviate the cost of classic ray-traced hard shadows, while this paper integrates it into the modern IZB pipeline. In our pipeline, fully shadowed nodes are detected using the conservative shadow map before constructing the lists, and then they are culled to reduce the cost of the IZB creation and frustum tracing (Fig. 1). Since the efficiency depends on the precision and computation cost of the conservative shadow map, this paper also presents a more precise implementation of conservative shadow mapping than the previous implementation. Using our method, we are able to improve the performance of hard shadows especially when shading points are occluded by multiple triangles.

2 CONSERVATIVE SHADOW MAPS

Our pipeline first renders a conservative shadow map (CSM) with the same resolution as the IZB. Unlike general shadow maps, a conservative depth is stored only into a texel fully covered by a triangle to detect fully shadowed shading points. This conservative depth must be more distant than the quadrilateral created by the plane of the triangle and the texel (Fig. 2). By setting the *slope scaled depth bias* by 1, the GPU rasterizer automatically outputs this conservative depth for a distant light source. Whether the texel is fully covered or not is tested in a pixel shader.

Fully Covered Texel. Hertel et al. [2009]'s implementation was too conservative to detect fully covered texels, because they tested whether the circumscribed circle of the texel is inside the triangle or not in the image space. To improve the precision of this detection, this paper uses barycentric coordinates at the four corners of the

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGGRAPH '18 Posters, August 12-16, 2018, Vancouver, BC, Canada
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5817-0/18/08.
https://doi.org/10.1145/3230744.3230755

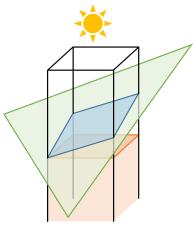

Figure 2: Conservative shadow maps store a depth more distant than the blue quadrilateral created by the triangle and texel. This depth gives the fully shadowed volume (orange) in each texel.

Program 1: Our pixel shader for CSMs (HLSL).

```
void main(float4 p : SV_Position, float2 c : BARYCENTRICS) {
  float2 dx = ddx(c) * 0.5;
  float2 dy = ddy(c) * 0.5;
  float2 a = dx + dy;
  float2 b = dx − dy;
  if(c.x < max(abs(a.x), abs(b.x)) || c.y < max(abs(a.y), abs(b.y)) ||
    1.0 − c.x − c.y < max(abs(a.x + a.y), abs(b.x + b.y))) {
    discard;
  }
}
```

Table 1: Rendering time of a CSM and culling rate.

	262k triangle scene		12.8M triangle scene	
	time	rate	time	rate
Hertel et al. [2009]	0.48 ms	55.5%	4.75 ms	35.8%
Ours	0.47 ms	56.0%	4.72 ms	37.4%

texel instead of the circumscribed circle. If these barycentric coordinates are all positive, this texel is fully covered. For a distant light source, this is simply implemented in a pixel shader as shown in Program 1. This precise implementation has almost the same computation cost as the previous implementation (Table 1). Although our current implementation uses a geometry shader to generate barycentrics, they can also be obtained using *SV_Barycentrics* which will be available in HLSL 6.1. Thus, our implementation can be further accelerated in the future.

3 TWO-PASS VISIBILITY TEST

In our IZB creation pass, the visibility of each shading point is roughly tested using the above conservative shadow map. If the shading point is shadowed, the shadow is outputted into a visibility mask buffer (Fig. 1a). Otherwise, the shading point is inserted into the IZB as a new node of the list. Although our IZB creation pass has an overhead for this visibility test and node culling, the computation time of this pass is reduced in practice (Fig. 3) because our node culling alleviates the cost of linked-list construction. After this IZB creation pass, the frustum tracing pass rasterizes the scene geometry in light space, and a pixel shader performs the accurate visibility test for the list (Fig. 1b).

Triangle Fragment Culling. For the frustum tracing pass, triangle fragments are culled using another depth buffer, similar to Wyman et al. [2016]. This depth buffer is also created in the IZB creation pass, by writing the depth at the farthest node into each texel. For our pipeline, this depth buffer has depth values smaller than the previous method because of node culling. Thus, fragments are also further culled in the frustum tracing pass. Since the depth buffer is written by the rasterizer, our IZB creation and depth buffer creation are implemented in a single pass using a vertex shader by treating each shading point as a point primitive.

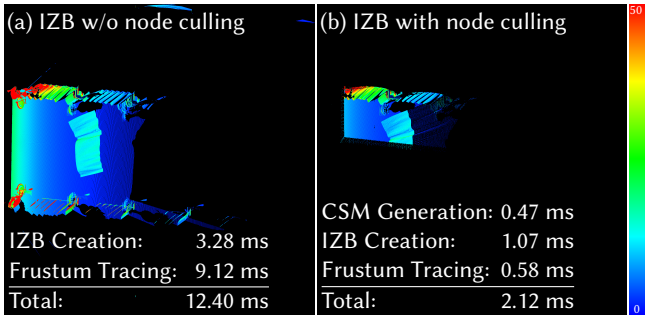

Figure 3: Visualizations of the length of the list in each IZB texel and computation time for each pass. Our pipeline culls fully shadowed candidate nodes for the list, and thus the performance is improved.

4 RESULTS AND FUTURE WORK

In our experiments for 4K screen resolution on an NVIDIA® GeForce® GTX 1070 GPU, our pipeline improves the computation time of hard shadows as shown in Fig. 1. The additional experiments for more complex scenes are shown in the supplementary material. The two-pass visibility test is especially effective when many shading points are occluded by multiple triangles (Fig. 3). On the other hand, there is still room for further improvement in performance.

Light-Space Partitioning. In our experiment, light-space partitioning [Story and Wyman 2016] (which performs load balancing for IZBs) is not used. Since partitions are determined based on the distribution of nodes, the efficiency can be further improved by combining the proposed node culling with partitioning.

Small Triangles. The conservative shadow map is not efficient for triangles smaller than the texel. However, it is not necessary to rasterize all the triangles in the scene to generate conservative shadow maps. Therefore, only objects containing large triangles should be drawn to reduce the overhead.

SV_Barycentrics vs. SV_InnerCoverage. A fully covered texel can also be detected using an HLSL system value *SV_InnerCoverage* for *conservative rasterization tier 3* capable GPUs[1]. Although this approach is simple, it has to use expensive conservative rasterization unlike our implementation. We would like to compare the performance between our implementation using SV_Barycentrics and this conservative rasterization-based approach in the future.

REFERENCES

S. Hertel, K. Hormann, and R. Westermann. 2009. A Hybrid GPU Rendering Pipeline for Alias-Free Hard Shadows. In *EG '09 Areas Papers*.

J. Story and C. Wyman. 2016. HFTS: Hybrid Frustum-traced Shadows in "the Division". In *SIGGRAPH '16 Talks*. Article 13, 2 pages.

C. Wyman, R. Hoetzlein, and A. Lefohn. 2016. Frustum-Traced Irregular Z-Buffers: Fast, Sub-Pixel Accurate Hard Shadows. *IEEE Trans. Vis. Comput. Graph.* 22, 10 (2016), 2249–2261.

[1]NVIDIA® GeForce® GTX 10 series are tier 2 capable.