DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

An evaluation of culling methods in VR? Rendering optimizations for Vulkan-based VR applications?

Paul Preißner

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

An evaluation of culling methods in VR? Rendering optimizations for Vulkan-based VR applications?

. . .

Author:	Paul Preißner
Supervisor:	Sven Liedtke
Advisor:	Advisor
Submission Date:	Submission date

Garching bei München, Submission date   Paul Preißner

Acknowledgments

Abstract

Kurzfassung

Contents

1 Introduction

1.1 Culling general research

1. Experiments in GPU-based occlusion culling (Anagnostou, Interplay of Light) [1]:
tldr CPU occlusion culling is very coarse as the CPU isn't great at rasterising. GPU can rasterise efficiently and supports hardware occlusion queries. Problem: "drawcall level" granularity, needs query and fence around every drawcall, thus can't handle instancing very well. Other problem: readback to CPU necessary, to avoid stalling use previous frame occlusion info for current frame. Thus popping likely for fast moving objects or camera. Other solution: render low res occlusion buffer on GPU, produce mip chain, determine screen-space size of each prop and compare vs appropriate mip-level of buffer. Still needs CPU readback though, unless... compute shader in modern API: "In this case I will be using a compute shader to perform the occlusion tests, producing a list of visible props that I will then be consuming on the GPU, avoiding the CPU roundtrip."

2. Why Frustum Culling Matters, and Why It's Not Important [2]:
tldr disregardm, oversimplified explanation of frustum culling.

3. Overview on popular occlusion culling techniques [3]:
tldr temporal culling methods are prone to artifacting with fast object or camera movement, CPU culling "is the most efficient, forward-looking approach", then again the article is written by someone from Umbra3D, a CPU occlusion culling middleware. Still, precomputed visibility sets for weak devices are brought up, low res HI-Z depth buffer rasterised on the CPU with SIMD units for dynamic culling, see Intel's 2015/2016 occlusion culling article.

4. Frustum Culling [4]:
tldr code samples and brief performance numbers for bounding sphere, AABB, OBB based culling, SSE support, MT support, somewhat simple GPU culling. Sphere or AABB with SSE + MT support quite fast, 0.1-0.2ms for 100k objects on unspecified i5 quadcore.

5. Dual-Cone View Culling for Virtual Reality Applications [5]:
tldr not discernibly faster than frustum pyramid culling plus stencil mesh. Needs careful tuning for each HMD to avoid performing worse when a larger image is rendered and warped to view. With SIMD, more accurate cones etc may be more efficient.

6. Culling Techniques [6]:
tldr general explanation. Of interest maybe hierarchical bounding volumes for frustum

culling. As the goal is a fast rendering of extreme number of objects, hierarchical ordering seems obviously necessary.

7. A Survey of Visibility for Walkthrough Applications [7]:
tldr .

8. Occlusion Culling Methods [8]:
tldr overview of methods anno 2001 (duh). Hierarchical frustum culling, occluder fusion, etc.

9. Math for Game Developers - Frustum Culling[9]:
tldr as expected, brief mathematical explanation on how to compute whether a prop is inside the view frustum (based on position and prop radius).

10. Wie Sie einen Einstieg in das Frustum Culling finden [10]:
tldr D3D sample code for view frustum culling. Very brief, not very useful. But raises the questions how to compute (and store) bounding sphere radius for each prop, and how to compute and update frustum planes each frame (aka reconstruct from view + proj matrices or set up at start and then move all planes according to camera movement).

11. Superfrustum culling [11]:
tldr instead of one frustum per eye, construct one superfrustum covering both. On paper means somewhat fewer objects are culled, but the reduced overhead still comes out on top. In their testing, shaved off 1ms on lower end CPUs. Also monoscopic far-field for lower end devices and far draw distances: past a certain distance, stereo separation becomes very small, nigh indistinguishable, so it may be faster to render the far field monoscopic and then composit with the near-field stereo image.

12. OpenGL sample for shader-based occlusion culling [12]:
tldr shader-based batched occlusion culling system: leverages multi_draw_indirect and works well with setups where all geo is stored in one big buffer.

13. GPU occlusion culling using compute shader with Vulkan [13]:
tldr follows [1] to showcase GPU occlusion culling.

14. Understanding Culling Methods | Live Training | Unreal Engine [14]:
tldr basic explanation of methods employed in UE4.

15. Occlusion Culling Algorithms [15]:
tldr .

Thus, things up in the air: frustum culling, occlusion culling, detail culling, distance culling, hierarchical culling, cpu culling vs gpu culling, dual-view frustum culling vs superfrustum culling, stencil meshing, ...center-distance culling for foveated rendering, SIMD and MT optimization, ...

1.2 Another angle: Monoscopic far-field rendering

Slides 14-31 in [16], [17], [18]: Basic idea is to render near-field (e.g. first 30ft) in stereo for good separation effect, anything beyond that in mono, then mask the background and composite the two images. This leverages the fact that stereo separation becomes very minimal beyond a certain distance, even so small that it's less than 1px shift. When tuned correctly and used in scenes with many objects beyond the threshold distance, this can yield good performance improvements of up to 25%. The savings from only drawing far objects once must outweigh the cost of adding another camera and more render passes. Another question is how savings scale when multiview rendering is used, as it already avoids a lot of redundant vertex shader work. While Epic Games seem to have dropped the feature from Unreal Engine in 4.22 for yet unknown reasons, I couldn't find much further research about it. One curious extension would be to use the far-field depth buffer to shift the image slightly in post-processing to create a fake stereo separation effect. Or perhaps to render not a centered far-field image, but to use the far-field image of each eye in alternating fashion. This may allow a further reduction in near-field far plane while still being a convincing approximation.

List of Figures

List of Tables

Bibliography

[1]   K. Anagnostou. *EXPERIMENTS IN GPU-BASED OCCLUSION CULLING*. 2017. URL: https://interplayoflight.wordpress.com/2017/11/15/experiments-in-gpu-based-occlusion-culling/.

[2]   S. Barrett. *Why Frustum Culling Matters, and Why It's Not Important*. 2017. URL: https://gist.github.com/nothings/913056601b56e5719cc987684a16544e.

[3]   *Overview on popular occlusion culling techniques: The experts at Umbra share their knowledge on solving the 3D visibility problem*. 2016. URL: https://www.gamesindustry.biz/articles/2016-12-07-overview-on-popular-occlusion-culling-techniques.

[4]   A. Gerlits. *Frustum Culling*. 2017. URL: https://www.gamedev.net/articles/programming/general-and-gameplay-programming/frustum-culling-r4613/.

[5]   J. Hale. *Dual-Cone View Culling for Virtual Reality Applications*. 2018. URL: https://blog.squareys.de/dual-cone-view-culling-for-vr/.

[6]   Dr. K.R. Subramanian. *Culling Techniques*. Charlotte, NC. URL: https://webpages.uncc.edu/krs/courses/5010/ged/lectures/cull_lod2.pdf.

[7]   D. Cohen-Or, Y. L. Chrysanthou, C. T. Silva, and F. Durand. "A Survey of Visibility for Walkthrough Applications". In: *IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS* 9.3 (2003), pp. 412–431. URL: https://www.researchgate.net/publication/2440562_A_Survey_of_Visibility_for_Walkthrough_Applications.

[8]   H. Hey and W. Purgathofer. "Occlusion Culling Methods". In: *EUROGRAPHICS STAR* (2001). URL: https://pdfs.semanticscholar.org/c1a0/aa9000f62bb9b1a60f27182e23872c375e1e.pdf.

[9]   J. Rodriguez. *Math for Game Developers - Frustum Culling*. 2013. URL: https://www.youtube.com/watch?v=4p-E_31XOPM.

[10]   *Wie Sie einen Einstieg in das Frustum Culling finden*. URL: https://viscircle.de/wie-sie-einen-einstieg-in-das-frustum-culling-finden/.

[11]   N. Whiting. *Oculus Connect 4 | The Road to Shipping: Technical Postmortem for Robo Recall: Superfrustum culling*. 2017. URL: https://www.youtube.com/watch?v=BZhOUGG45_o&feature=youtu.be&t=46m12s.

[12]   C. Kubisch. *OpenGL sample for shader-based occlusion culling*. 2014. URL: https://github.com/nvpro-samples/gl_occlusion_culling.

[13]   sydneyzh. *GPU occlusion culling using compute shader with Vulkan*. 2018. URL: https://github.com/sydneyzh/gpu_occlusion_culling_vk.

[14]   T. Hobson. *Understanding Culling Methods | Live Training | Unreal Engine*. 2019. URL: https://www.youtube.com/watch?v=6WtE3CoFMXU.

[15]   E. Haines and T. Müller. *Occlusion Culling Algorithms*. 1999. URL: https://www.gamasutra.com/view/feature/131801/occlusion_culling_algorithms.php.

[16]   D. Di Donato, R. Palandri, and R. Vance. *High quality mobile VR with Unreal Engine and Oculus*. 1.03.2017.

[17]   *Monoscopic Far Field Rendering*. 2016. URL: https://docs.unrealengine.com/en-US/Platforms/VR/DevelopVR/MonoFarFieldRendering/index.html.

[18]   *Hybrid Monoscopic Rendering*. 2016. URL: https://developer.oculus.com/documentation/unreal/1.14/concepts/unreal-hybrid-monoscopic/.