Occlusion Culling in Alan Wake

Ari Silvennoinen∗
Umbra Software

Teppo Soininen†
Umbra Software

Markus Mäki‡
Remedy Entertainment

Olli Tervo§
Remedy Entertainment

1 Introduction

The combination of large outdoor environments and dynamic, shadow casting light sources posed a rendering performance challenge Remedy had to tackle during the production of Alan Wake.

This talk presents the evolution of the occlusion culling system used in Alan Wake from early concepts and manually placed portals to a fully automatic online occlusion culling solution developed together with Umbra Software.

In addition to using this system to cull invisible objects and light sources from the camera’s point of view, Alan Wake uses the visibility information in a novel way to efficiently cull shadow casters from the light’s point of view to speed up shadow map generation. This extension uses a mask of potential shadow receivers to cull shadow casters using a hierarchical occlusion culling algorithm.

We will demonstrate the speedup in rendering performance in Alan Wake and conclude the talk with a discussion of the trade-offs between hardware and software based occlusion culling systems.

2 Culling Algorithms

For solving visibility from the camera’s point of view, Alan Wake used a hierarchical hardware based occlusion query algorithm [Bittenner et al. 2004], [Guthe et al. 2006]. The advantages of using a hardware based culling algorithm outweighed the content pipeline issues associated with using a typical software occlusion system based on either precomputed potentially visible sets (PVS) or artist generated occlusion geometry and software rasterizer [Aila and Miettinen 2004]. Occlusion culling minimized the overhead and the bottleneck shifted to shadow map generation.

In order to speed up shadow map rendering, Alan Wake used a prototypical implementation of a shadow caster culling algorithm for efficient shadow mapping [Bittner et al. 2011]. A simple approximation to the receiver mask is obtained by spanning a light space scissor rectangle for each shadow cascade based on camera visibility. This light space AABB can be also used for accurate shadow frustum culling and it automatically culls empty cascades.

3 Conclusion

The main benefit of a hardware based occlusion system is that it does not require any modifications to the game content or the content pipeline, whereas typical software based approaches require customized occlusion models. This can lead to increased content generation times and the dependencies between renderable models and their respective occlusion models can be tedious to maintain.

There are basically two major drawbacks in using a hardware based solution. First, a GPU assisted solution requires the application to synchronize CPU and GPU in order to know which draw calls need to be submitted to the GPU. Second, processing the hardware occlusion queries consumes valuable GPU time, which can be an issue on some target platforms.

The choice between a hardware and software based occlusion system requires a careful thought and is dependent on the application and target platform. However, we expect that the content pipeline issues related to software based systems will be solved in the near future, making it a more viable option in a wider array of scenarios.

References


∗e-mail: ari@umbrasoftware.com
†e-mail: teppo@umbasoftware.com
‡e-mail: markus@remedygames.com
§e-mail: olli.tervo@remedygames.com