REAL TIME VISIBILITY CULLING WITH HARDWARE OCCLUSION QUERIES AND UNIFORM GRIDS

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

by

Ilya Seletsky

June 2013

COMMITTEE MEMBERSHIP

TITLE:	Real Time Visibility Culling With Hardware Occlusion Queries and Uniform Grids
AUTHOR:	Ilya Seletsky
DATE SUBMITTED:	June 2013
COMMITTEE CHAIR:	Zoë Wood, Ph.D.
COMMITTEE MEMBER:	Foaad Khosmood, Ph.D.
COMMITTEE MEMBER:	Aaron Keen, Ph.D.

Abstract

Real Time Visibility Culling With Hardware Occlusion Queries and Uniform Grids

Ilya Seletsky

Culling out non-visible portions of 3D scenes is important for rendering large complex worlds at an interactive frame rate. Past 3D engines used static prebaked visibility data which was generated using complex algorithms. Hardware Occlusion Queries are a modern feature that allows engines to determine if objects are invisible on the fly. This allows for fully dynamic destructible and editable environments as opposed to static prebaked environments of the past. This paper presents an algorithm that uses Hardware Occlusion Queries to cull fully dynamic scenes in real time. This algorithm is relatively simple in comparison to other real time occlusion culling techniques, making it possible for the average developer to render large detailed scenes. It also requires very little work from the artists who design the scenes since no portals, occluders, or other special objects need to be used.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Real Time Graphics

Real time computer graphics are needed for CAD applications, simulations, video games, and even for just displaying the user interface of an operating system. Computer graphics can be non-real time, like when doing CGI for a movie. Each frame can take hours to render, and the result is a very photorealistic image that looks like it's part of the movie itself. Millions of frames are generated only for each still frame to be visible for a fraction of a second until the next frame to give the illusion of motion. Real time graphics also display a frame for only a fraction of a second but are rendered on the fly right as the user is controlling the application. This means the computer can't sit there for hours rendering a frame, it has 33.33 milliseconds if trying to render at 30 frames per second. Movies are typically shown at 25 FPS (frames per second), but 30 FPS is about the threshold until us humans start to notice slightly unresponsive input and jerky movement when directly controlling a real time graphics application. Most computer monitors update at 60 Hz so real time graphics typically strive for 60

FPS giving an even smoother experience, and this means the computer now only has 16.66 milliseconds to render a frame.

1.2 Visibility Culling

Modern graphics hardware is becoming better and better at rendering large detailed environments in real time. Games like Rage, Crysis, Skyrim, Battlefield 3, Just Cause 2, and Grand Theft Auto have maps on an epic scale with drawing distances of several kilometers. These kinds of environments are achievable by avoiding drawing as much of the world as possible. This is the job of a visibility culling algorithm. It's relatively easy to avoid drawing portions of the scene that aren't even in the camera's field of view. If a camera is looking at a wall or a mountain in the distance, it's also best to avoid drawing things behind those objects. With large scenes, it's possible that only about 1% of it has to be rendered.

The GPU is very good at drawing geometry. Modern hardware can render about X triangles with relative ease. It's perfectly fine for the algorithm to have some false positives when rendering a few objects that are actually not visible. This would just cause a negligible performance drop that the hardware can just brush off. The starts to suffer if objects go missing that should actually be there, so false negatives on visibility should be avoided. Most of the visibility culling work happens on the CPU while the GPU does all the rendering. If the CPU is doing too much work finding which objects the GPU shouldn't render, the performance may be worse than if there is no occlusion culling algorithm in the first place. A good algorithm must strike the perfect balance where the CPU isn't working too hard but is still saving the GPU a lot of work, giving a considerable

performance increase.

If the GPU is so powerful compared to the CPU, why not have the GPU run the occlusion culling algorithm instead? The architecture of the GPU makes it well suited for running the same operation on large amounts of data in parallel, such as all the vertices, triangles, and pixels that will show up on screen. The GPU isn't good at branching and other sequential algorithms that the CPU is good at. The CPU is the one who must do the actual work of checking if an object is visible, and then issuing the draw calls to the GPU.

1.3 Dynamic Environments

Static environments of the past used prebaking steps to compute things like lighting and visibility. Nowadays more is possible with fully destructible environments and WYSIWYG level editors built right into the game. Levels nowadays are typically made out of modular set pieces and put together like lego blocks. A wall, column, railing, ladder, or tree may be reused many times in the level, saving artists a lot of work as well as keeping memory down due to loading in one resource and reusing it many times. Past 3D engines would use polygon soups made out of static Binary Space Partitioning geometry brushes. Now 3D engines rely on mesh soups of these modular objects.

While designing levels, the artists would work with these objects by throwing in new ones, moving them around, and deleting them. The level editor's performance can be greatly increased if it uses a real time visibility culling algorithm. In a game with destructible environments, these same objects can now be moved around and deleted in real time. With all of this happening, visibility data needs to update automatically and in real time. In fact, the level editor can be either

built right into the game or just use the same 3D engine that that game uses. This lets artists see exactly how the environment looks as if they were currently playing the game unlike the engines of the 90s.

1.4 Artist Placed Hint Objects

Sometimes an algorithm can't figure out the best way to perform visibility culling in every situation so artists need to place special objects. This creates a lot of extra work for the artists and can be really tedious. Artists now need to be educated on the technical inner workings of the engine so they can place the hint objects in the best way. Even then, artists may still not do it the best way.

Portals are objects that are placed in windows and doorways linking different rooms or zones of an indoor environment. They help the engine render objects only visible to the camera through the portal geometry and not through the walls. They are effective in static environments where the artists know a door or window will always be around. However if someone takes a rocket launcher and blows up a hole in a wall, there is no artist placed portal there to help the engine. The engine won't necessarily be smart enough to know how to best place a portal there since any kind of arbitrary destruction can take place. The zones divided by portals are often static as well and the engine would need to somehow figure out how to create new zones on the fly. Overall, portals become pretty useless in non-static environments. Portals can also be very tedious for the level designers to keep track of as they are constantly iterating on the design of a level. The last thing an artist wants to do after all the work of making a room look pretty before seeing it in action is to go in and add a bunch of portal objects.

Occluder geometry is also often used in modern game engines. Modelers

might make an extremely low detail version of a mesh. The engine can then do ray intersection tests against all triangles in this geometry to test if some object is behind the geometry. Engines that do software occlusion culling render these meshes CPU side rather than rendering the detailed geometry the GPU powered hardware occlusion queries would render. Making these low resolution meshes that are used specifically for occlusion culling create extra work for modelers. If an object is destructible, the low resolution mesh must somehow be deformable along with the high resolution mesh it represents.

1.5 Our Contribution

This paper presents a simple high performing visibility culling algorithm that any 3D engine can integrate. It allows rendering of large outdoor worlds that are seamlessly combined with detailed indoor architecture. It requires hardware that supports occlusion queries, which have been available since about 2003.

This algorithm avoids the added complexity of a software rasterizer. Similar algorithms use software occlusion queries in order to avoid the delay when retrieving hardware occlusion query results. This would require software rasterizers to be written that run on the CPU instead of making use of the hardware that performs this task far more efficiently. The paper also presents a method for combining the use of queries with effective render state sorting.

There is very minimal artist intervention required when designing environments using this algorithm. The high detail objects that are rendered are also automatically occluding parts of the environment without the need for artists to manually tweak things with portals or special occluding geometry.

A front to back view frustum scene traversal algorithm is presented in order to make the query results optimal. If the objects in front are drawn before the objects behind, the objects behind can be tested against those already drawn objects. This reduces false positives in visibility tests.

Chapter 2

Previous Work

LaTeX is a document markup language and document preparation system for the TeX typesetting program.

It is widely used by mathematicians, scientists, philosophers, engineers, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language to access the power of TeX. LaTeX essentially comprises a collection of TeX macros, and a program to process LaTeX documents. Since TeX's formatting commands are very low-level, it is usually much simpler for end-users to use LaTeX.

LaTeX was originally written in 1984 by Leslie Lamport at SRI International and has become the dominant method for using TeXfew people write in plain

TeX anymore.

LaTeX is based on the idea that authors should be able to focus on the meaning of what they are writing, without being distracted by the visual presentation of the information. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. It therefore encourages the separation of layout from content, while still allowing manual typesetting adjustments where needed. This is similar to the mechanism by which many word processors allow styles to be defined globally for an entire document, or the CSS mechanism used by HTML.

2.1 This is a new section

LaTeX is a document markup language and document preparation system for the TeX typesetting program.

It is widely used by mathematicians, scientists, philosophers, engineers, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language to access the power of TeX. LaTeX essentially comprises a collection of TeX macros, and a program to process LaTeX documents. Since TeX's formatting commands are very low-level,

$$\begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} = \begin{bmatrix} p'_x \\ p'_y \\ p'_z \end{bmatrix}$$

Figure 2.1: This is a sample matrix equation.

it is usually much simpler for end-users to use LaTeX.

LaTeX was originally written in 1984 by Leslie Lamport at SRI International and has become the dominant method for using TeXfew people write in plain TeX anymore.

LaTeX is based on the idea that authors should be able to focus on the meaning of what they are writing, without being distracted by the visual presentation of the information. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. It therefore encourages the separation of layout from content, while still allowing manual typesetting adjustments where needed. This is similar to the mechanism by which many word processors allow styles to be defined globally for an entire document, or the CSS mechanism used by HTML.

1. Here is a list item.

    (a) Here is a sub list item.

        i. Here is a sub sub list item.

Chapter 3

Results

Here is the results section.

LaTeX is a document markup language and document preparation system for the TeX typesetting program.

It is widely used by mathematicians, scientists, philosophers, engineers, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language to access the power of TeX. LaTeX essentially comprises a collection of TeX macros, and a program to process LaTeX documents. Since TeX's formatting commands are very low-level, it is usually much simpler for end-users to use LaTeX.

LaTeX was originally written in 1984 by Leslie Lamport at SRI International

and has become the dominant method for using TeXfew people write in plain TeX anymore.

LaTeX is based on the idea that authors should be able to focus on the meaning of what they are writing, without being distracted by the visual presentation of the information. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures. It therefore encourages the separation of layout from content, while still allowing manual typesetting adjustments where needed. This is similar to the mechanism by which many word processors allow styles to be defined globally for an entire document, or the CSS mechanism used by HTML.

LaTeX can be arbitrarily extended by using the underlying macro language to develop custom formats. Such macros are often collected into packages which are available to address special formatting issues such as complicated mathematical content or graphics. In addition, there are numerous commercial implementations of the entire TeX system, including LaTeX, to which vendors may add extra features like additional typefaces and telephone support. LyX is a free visual document processor that uses LaTeX for a back-end. TeXmacs is a free, WYSIWYG editor with similar functionalities as LaTeX, but a different typesetting engine.

A number of popular commercial desktop publishing systems use modified versions of the original TeX typesetting engine. The recent rise in popularity of XML systems and the demand for large-scale batch production of publication-quality typesetting from such sources has seen a steady increase in the use of LaTeX.

LaTeX is a document markup language and document preparation system for

	Some Data			Some More Data		
	Hi-Res	Lo-Res	Reduction	Hi-Res	Lo-Res	Speedup
Row Data A	225,467	43,850	80.6%	360	90	4.0
Row Data B	225,467	16,388	92.7%	360	26	13.8

Table 3.1: Here is some performance data for the system.

the TeX typesetting program.

It is widely used by mathematicians, scientists, philosophers, engineers, and scholars in academia and the commercial world, and by others as a primary or intermediate format (e.g. translating DocBook and other XML-based formats to PDF) because of the quality of typesetting achievable by TeX. It offers programmable desktop publishing features and extensive facilities for automating most aspects of typesetting and desktop publishing, including numbering and cross-referencing, tables and figures, page layout and bibliographies.

LaTeX is intended to provide a high-level language to access the power of TeX. LaTeX essentially comprises a collection of TeX macros, and a program to process LaTeX documents. Since TeX's formatting commands are very low-level, it is usually much simpler for end-users to use LaTeX.

LaTeX was originally written in 1984 by Leslie Lamport at SRI International and has become the dominant method for using TeXfew people write in plain TeX anymore.

LaTeX is based on the idea that authors should be able to focus on the meaning of what they are writing, without being distracted by the visual presentation of the information. In preparing a LaTeX document, the author specifies the logical structure using familiar concepts such as chapter, section, table, figure, etc., and lets the LaTeX system worry about the presentation of these structures.

It therefore encourages the separation of layout from content, while still allowing manual typesetting adjustments where needed. This is similar to the mechanism by which many word processors allow styles to be defined globally for an entire document, or the CSS mechanism used by HTML.

LaTeX can be arbitrarily extended by using the underlying macro language to develop custom formats. Such macros are often collected into packages which are available to address special formatting issues such as complicated mathematical content or graphics. In addition, there are numerous commercial implementations of the entire TeX system, including LaTeX, to which vendors may add extra features like additional typefaces and telephone support. LyX is a free visual document processor that uses LaTeX for a back-end. TeXmacs is a free, WYSIWYG editor with similar functionalities as LaTeX, but a different typesetting engine.

A number of popular commercial desktop publishing systems use modified versions of the original TeX typesetting engine. The recent rise in popularity of XML systems and the demand for large-scale batch production of publication-quality typesetting from such sources has seen a steady increase in the use of LaTeX.

Bibliography

[1] ATI NormalMapper. http://ati.amd.com/developer/tools.html.

[2] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[3] T. Ju, F. Losasso, S. Schaefer, and J. Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM Press.

[4] nVidia Melody. http://developer.nvidia.com/object/melody_home.html.