Occluder Simplification using Planar Sections

Ari Silvennoinen
Remedy Entertainment
Aalto University

Hannu Saransaari
Umbra Software

Samuli Laine
NVIDIA

Jaakko Lehtinen
NVIDIA
Aalto University
Coping with Scene Complexity

We want to render more than we can afford
Coping with Scene Complexity

We want to render more than we can afford

Need to use visibility culling and level-of-detail techniques
Visibility Culling
Visibility Culling
Visibility Culling
Visibility Culling
Problem

Visibility algorithm needs to be **faster** than processing the whole scene.
Problem

Visibility algorithm needs to be faster than processing the whole scene

Need simplified occluders; currently mostly manual work
Goal

Automatic occluder simplification

Given a display mesh and a budget of N triangles, generate simplified model with similar occlusion characteristics
Goal

Automatic occluder simplification

Given a **display mesh** and a **budget** of N triangles, generate simplified model with **similar occlusion characteristics**

Works with **general 3D triangle soups**
Previous Work

Special cases for occlusion

- Subset of the input [Coorg and Teller 1997], [Wonka and Schmalstieg 1999]
- 2.5D urban scenes [Germs and Jensen 2001]
- Valid from small region only (e.g., hoops) [Brunet 2001]
- Simple, axis-aligned 3D scenes [Darnell 2011]

Difficult to generalize
Previous Work

General mesh simplification methods

− Simplification envelopes [Cohen 1996]
− Quadratic error metrics [Garland and Heckbert 1997]
− Voxel-based [Nooruddin and Turk 2003]
− Textured tangent planes [Decoret 2003]

Focus on visual similarity which is not the same as occlusion
Main Idea: Focus on Occlusion
Main Idea: Focus on Occlusion
Main Idea: Focus on Occlusion
Main Idea: Focus on Occlusion
Main Idea: Focus on Occlusion
Main Idea: Focus on Occlusion

Interior slice is a conservative occluder
Main Idea: Focus on Occlusion

Line soup and polygon have similar occlusion characteristics.
Main Idea: Focus on Occlusion

Line soup and polygon have similar occlusion characteristics.
Main Idea: Focus on Occlusion

Focusing on occlusion gives us more freedom
Algorithm Sketch

1. Cut the model using a large number of planes
2. Assemble the cuts such they satisfy our budget and \textbf{maximise occlusion similarity}
How to Quantify Occlusion?
Surface Area?
Surface Area?
Why is a sphere a better occluder than a torus?
Topological Erosion

Why is a sphere a better occluder than a torus?
Topological Erosion

Why is a sphere a **better occluder** than a torus?
Topological Erosion

Why is a sphere a **better occluder** than a torus?
Topological Erosion

Why is a sphere a better occluder than a torus?
Topological Erosion

Why is a sphere a better occluder than a torus?
Measuring Occlusion

Surface area after erosion
Measuring Occlusion

\[
\int_{\Omega}^{\infty} \int_{0}^{\infty} \text{Surface area after erosion } \, dr \, d\omega
\]
Algorithm Outline

1. **Voxelize** the input model and build a closed model with a well defined interior

2. **Generate** a large number of planar polygons by sampling the interior of the voxel model

3. **Assemble** the polygons such that they satisfy our budget and maximise total occlusion measure
1. **Voxelize** the input model and build a closed model with a well defined interior

2. **Generate** a large number of planar polygons by sampling the interior of the voxel model

3. **Assemble** the polygons such that they satisfy our budget and maximise total occlusion measure
Algorithm Outline

1. **Voxelize** the input model and build a closed model with a well defined interior

2. **Generate** a large number of planar polygons by sampling the interior of the voxel model

3. **Assemble** the polygons such that they satisfy our budget and maximise total occlusion measure
1. Voxelization
2. Interior Sampling
2. Interior Sampling
2. Interior Sampling

Rasterized interior slice → Polygons
3. Occluder Assembly
3. Occluder Assembly

Use a **greedy** method to pick one polygon at a time

Evaluate **total occlusion measure** for each possible choice and choose the **maximal** one

**Iterate** the process until the occluder is complete
3. Occluder Assembly

The resulting occluder might still exceed our triangle budget

Remove triangles one-by-one until we reach our budget
Bunny 5K
Buddha

Input 1087474 tris

Output 64 tris
Machine

Input 394452 tris

Output 64 tris
Dragon

Input 871306 tris

Output 64 tris
Surface Area vs. Occlusion Measure

Input

Surface Area

Occlusion Measure
Comparison against Oxel

Input 28 tris
Ours 6 tris
Oxel 509 tris

[Darnell 2011]
Difficult Input

Input 100K tris

Ours 64 tris

Comparison
Hierarchical extension

Straightforward extension to handle **large scenes**

Build a BVH over the input geometry and apply the algorithm to **each** node separately
Occluders

1095 occluder triangles, 0.10% of all triangles
Conclusions and Future Work

**Principled way to quantify occlusion**
— Fast evaluation based on Euclidean Distance Transform

**Scalable method for occluder simplification**
— Works with general triangle soups
Conclusions and Future Work

Principled way to quantify occlusion
— Fast evaluation based on Euclidean Distance Transform

Scalable method for occluder simplification
— Works with general triangle soups

Could we apply the occlusion measure / simplified occluders to other problems?
— Light ray connections in path tracing? BVH build heuristics?
Thank You!

Acknowledgments

Anonymous reviewers for constructive and thorough feedback.
Bounded Approximation Error

Sources of error

— Voxelization error (bounded by voxel size)
— Rasterization error (bounded by raster resolution)
— Edge loop simplification error (bounded by tolerance)

Single user parameter: Voxel resolution

— Raster resolution and edge loop simplification set accordingly
Effect of Scale-Sensitive Discretization

16x16x16  32x32x32  64x64x64  128x128x128
Comparison to Other Methods

- **Input**
- **Ours**
- **Oxel**
- **Max**

Models: Bunny, Dragon, Terrain, Machine, Buddha
Fast Evaluation of the Occlusion Measure

Directional occlusion is connected to **Euclidean Distance Transform**

Gives a practical way to calculate the occlusion measure
Hierarchical extension

Input

Hierarchical Occluder