Point-Based Multi-Resolution Rendering
Overview:

- Introduction
- Forward Mapping
- Animated Scenes
- Raytracing
- Extensions
Overview:

- Introduction
- Forward Mapping
- Animated Scenes
- Raytracing
- Extensions
Motivation

Highly Complex Scenes

- Billions of primitives
- Interactive rendering (CAD, visualization, games...)
- No general solution
Mesh Simplification

Related Work: Mesh Simplification

- [Schröder et al. 92], [Hoppe et al. 93], [Hoppe 96]
- Works well for smooth meshes
- Problems with irregular topology
Related Work –
Image Based Rendering:

- [Regan and Pose 94],
  [Maciel and Shirley 95],
  [Gortler et al. 96],
  [Levoy and Hanrahan 96],
  [Shade 96], [Schauffler 98]
- Parallax problems vs. memory demands
Point-Based Approach...

Point-Based Multi-Resolution Rendering:

- Select surface sample points
- Distributed uniformly in image
- Reconstruct image out of sample points
Point-Based Approach...

Advantages:

• Sample size independent of scene complexity
• No topological constraints
• Avoids parallax errors
Related Work

Some Earlier Work in Point-Based Rendering:

- *Particle Systems* [Reeves 83], [Reeves and Blau 85]
- *Surface Rendering* [Levoy and Whitted 85]
- *Volume Rendering* [Westover 90]
- *Voxel Space* [Novalogic 92]
- *Hierarchical Simplification* [Chamberlain et. al 96]
- *Image-Based Rendering* [Shade 98], [Lischinski and Rappoport 98]
- *Simplification* [Max 96, Grossman and Dally 98]
Recent Work

Some Recent Related Work

• *Surfels* [Pfister et al. 2000]
  *QSplat* [Rusinkiewicz and Levoy 2000]

• *Surface Splatting* [Zwicker 2001]


• *Raytracing* [Schaufler and Jensen 2000, Adamson and Alexa 2001]
Overview:

- Introduction
- **Forward Mapping**
- Animated Scenes
- Raytracing
- Extensions
A Dynamic Sampling
- “Randomized z-Buffer”
- Developed in parallel to Surfels / QSplat (mid 1999 - 2000)
- Dynamic random sampling

B Static Sampling
- Precomputed sample sets
- Based on Surfels & Rnd.-z-Buffer
A – Dynamic Sampling

Dynamic Sampling
Dynamic Sampling Data Structure

- 1. Spatial octree: Fix sampling density for regions
- 2. Distribution lists: Random sampling in $O(\log n)$
- Piecewise constant sampling density
First...

Sample Set Selection
Perspective Mapping

Perspective Mapping

Projection factor \( prj(x) \)

- area scale factor of perspective projection
- sampling density \( \sim prj \), conservative approximation
Perspective Mapping

Projection factor $prj(x)$

- area scale factor of perspective projection
- sampling density $\sim prj$, conservative approximation

$$prj(x) = \frac{1}{z^2} \cdot \cos \beta \cdot \frac{1}{\cos \alpha}$$

- depth factor
- orientation factor
- distortion factor
Depth Approximation

Approximating the Depth Factor:

- $\varepsilon$-approximation
- Octree traversal
- Subdivide if still
  \[ \frac{z_{\text{max}}^2}{z_{\text{min}}^2} > 1 + \varepsilon \]

Efficiency:

- Running time $O(\log \tau + h)$
- Relative depth range $\tau$, octree height $h$
Orientation, Distortion

Other Factors:

- **Distortion factor**: No problem in practice
- **Orientation factor**: Two options...

\[ prj(x) = \frac{1}{z^2} \cdot \cos \beta \cdot \frac{1}{\cos \alpha} \]
Orientation Factor

Orientation Classes

• Adapt sampling density to orientation
• Minor speedup in practice
• No strict $\varepsilon$-approximation possible
• Ignore orientation $\Rightarrow$ Average oversampling factor $2\times$
Image Reconstruction
Reconstr. of Occlusion

Two problems:

1. Reconstruction of occlusion
   - remove adjacent points with larger depth

2. Filling
   - scattered data interpolation

Sample points
Interpolation Options

Per-pixel reconstruction

Gaussian reconstruction
Analysis
How many random sample points are necessary?

- Criterion: Save surface coverage
- \( k \leq a(\ln a + \ln f^{-1}) \) points necessary
- \( a \) – estimated projected area \textit{in pixel} (incl. overestimation and occlusion)
- Average scenes: \( a \) linear in true projected area
- \( f = \) failure probability
Preprocessing Costs:

- Preprocessing: $O(n)$ memory, $O(n \log n)$ time.
- Dynamic updates (insert, delete triangles): $O(h)$
- In practice: 8 seconds for 90K triangles & instances
Rendering

- Running time $O(\log \tau + h + \alpha \frac{\log \alpha}{\log n})$
- $\tau$ – relative depth range
- $h$ – octree height
- $\alpha$ – estimated projected area
  (incl. overestimation and occlusion)

Applicable to highly complex scenes
Examples...

0.8 - 3.9 sec.
(4 \cdot 10^8 \text{ triangles, } 640 \times 480 \text{ pixel})

0.4 - 81 sec.
(6 \cdot 10^9 \text{ triangles, } 640 \times 480 \text{ pixel})
Static Sampling
Static Sampling: Precomputed Sample Sets (cf. Surfels)

- Octree Hierarchy
- Sample spacing: Fraction of box side length
- Store large triangles (say > 3 points) “as-is”
Which sampling strategy is best?

- Candidates:
  - Random sampling
  - Jittered grid
  - Quantized grid
  - Neighborhood based sampling ($\approx$ Poisson disc)

- Criterion: Oversampling
  - Analytical upper bounds
  - Average case (empirical)
Sampling

Which sampling strategy is best?

• Candidates:
  • Random sampling
  • Jittered grid
  • Quantized grid flexible & efficient
  • Neighborhood based sampling (≈ Poisson disc)

• Criterion: Oversampling
  • Analytical upper bounds
  • Average case (empirical)
Sampling

Neighborhood-Based Point Removal:

- Two step approach
- Candidate set, stratification

First Step: Candidate Set

- Random candidate set
- Uniformly distributed on triangle area
- $O(\alpha \log \alpha + \log f^{-1})$ points
  $$(\alpha = \text{area} / \text{sampling distance}^2, f = \text{failure probability})$$
Second Step: Stratification

- Delete points that are still covered by other points
- Greedy strategy
- Typically $O(n \log^2 n)$ processing time, $n$ points output
Oversampling

random

8-50x

grid

13.4x
(upper bound: 28.3x)

neighbors

1.61x
(upper bound: 3.6x)

quantized

3.45x
(upper bound: 7.1x)

Optimum: 1.21x
Efficiency

Static Sampling

- Overall rendering time $O(\log \tau + h + a \log \log n)$
- $\tau$ – Relative depth range
- $h$ – octree height
- $a$ – estimated projected area
  (incl. overestimation and occlusion)
- Preprocessing: $O(n)$ memory, $O(hn)$ time.

Allows real-time rendering
Rendering Performance

4 frames / sec.
(4·10^8 triangles,
640 \times 480 pixel)

5-10 frames / sec.
(10^{15} triangles,
640 \times 480 pixel)
Overview:

• Introduction
• Forward Mapping
• Animated Scenes
• Raytracing
• Extensions
Modeling Animations
Keyframe Animations

Modeling of Animations:

- Keyframe animations
- Linear interpolation
- Arbitrary topology
Animation

Generalization to Keyframe Animations:

- Separate hierarchy between consecutive keyframes
- Hierarchy interpolation
Interpolated Point Hierarchies
Interpolated Hierarchies:

- *Bounding boxes* can be interpolated linearly (upper bound)
- *Triangle vertices* are interpolated linearly (incl. attributes)
- *Point samples* are interpolated linearly (incl. attributes)
Animated Sampling

Problem: How to define sample sets?

• Grid-based sampling not applicable
  ⇒ Neighborhood-based sampling

First Step: Random Sampling

• Use maximum area per triangle, coverage still guaranteed

Second Step: Stratification

• Points cover each other if:
  • sufficiently small distance at start...
  • ...and end time.
Application Examples
Instantiation

16,416 football fans à 6,400 triangles, 640 × 480 pixel
⇒ 105 million triangles

Rendering speed: 10-20 frames / sec
Simulation

1.300 horses, 200 trees
⇒ 42 million triangles

Rendering speed: up to 8-10 frames / sec
Sound Rendering

Extension: Point-Sampling Based Audio Synthesis

• Same principle – observer dependent sampling
2.000 Tonquellen
Overview:

- Introduction
- Forward Mapping
- Animated Scenes
- Raytracing
- Extensions
Multi-Resolution Raytracing

Point-Based Multi-Resolution Raytracing:

- Antialiasing, soft shadows, blurry reflections etc...
- 1 primary ray per Pixel

Algorithm: [cf. Amanatides 84]

- Shoot extended ray cones
- Prefiltered (static) point hierarchy
- Select points that match ray footprint
- Local surface reconstruction

Extension of Surface Splatting [Zwicker et al. 2001]
First...

Ray Model
Extended Ray Volumes

Anisotropic Ray Volumes:
1st order approximation (cf. [Igehy 99])

\[ fp(t) = \text{startDev} + t \cdot \text{incrDev} \]
Extended Ray Volumes (2)

Gaussian Filter:

- \( -(u,v) \cdot f_p(t)^{-2} \cdot \begin{pmatrix} u \\ v \end{pmatrix} \)

- weight\((u,v) = e\)

- Eigenspace: \( f_p(t)^2 = U \begin{pmatrix} \lambda_1^2 & 0 \\ 0 & \lambda_2^2 \end{pmatrix} U^T \)

\((U \text{ orthogonal})\)
Surface Interaction

primary rays

reflection

depth-of-field

shadow rays

[Kalaiah et al. ’01]
Intersection Tests
Intersection Tests

Intersection with:
- Bounding volumes, points, triangles

General Technique:
- Project vertices into ray coordinates (Eigensystem)
- Calculate distance
- Evaluate Gaussian filter
Ray Compositing: “Ray A-Buffer”  (cf. [Zwicker et al. ‘01])

- Merge points with overlapping depth (same surface)
- Blending for different surfaces:
  - (a) Alpha blending (weight sum)
  - (b) Subpixel masks
Merging

Fragment Merging:

- Weighted sum of point attributes
- Normalize by weight sum

point $p_i$:

attribute: $a_i(u,v)$

weight: $w_i(u,v)$

reconstructed attribute: \[
\frac{\sum_{i=1}^{n} w_i \cdot a_i}{\sum_{i=1}^{n} w_i}
\]
Conventional Raytracing

Conventional Raytracing: 215 sec

MR Raytracing: 1332 sec
Distributed Raytracing: 1334 sec

MR Raytracing: 1332 sec
Subpixel Masks

MR Raytracing, $\alpha$-blending:
1332 sec

MR Raytracing, subpixel masks:
7244 sec
Special Effects

- soft shadows
- depth-of-field
- blurry reflections
Overview:

• Introduction
• Forward Mapping
• Animated Scenes
• Raytracing
• Extensions
Extensions...

Multi-Resolution Volume Rendering
(joint work with Stefan Guthe)

Volume Rendering: \(O(\log n)\) rendering time for \(n^3\) voxels
Visible Human (8GB) at 5-10 fps

Sound Rendering: Real-time auralization of scenes with a large numbers of sound sources