Surfels: Surface Elements as Rendering Primitives

Hanspeter Pfister, Matthias Zwicker, Jeroen van Baar and Markus Gross

SIGGRAPH 2000: International Conference on Computer Graphics and Interactive Techniques, pages 335-242

Problems with Current Graphics Rendering Systems

- Interactive computer graphics has not reached the level of realism that allows true immersion into a virtual world
- Rendering realistic, organic looking models requires highly complex shapes with a huge number of triangles
- Processing many small triangles leads to bandwidth bottlenecks and excessive floating point and rasterization requirements

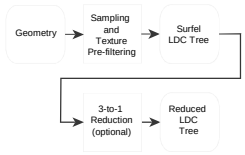
New Solution: Surfels

A Surfel is a zero-dimensional n-tuple + attributes.

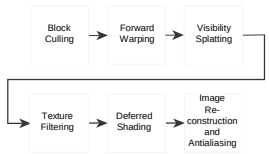
- Local approximation of an object surface
- Attributes: depth, texture color, normal and others

Conceptual Overview

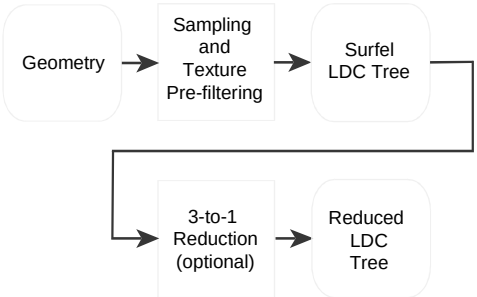
Pre-processing - Surfels generation:



Rendering of Surfels:



Preprocessing



Sampling

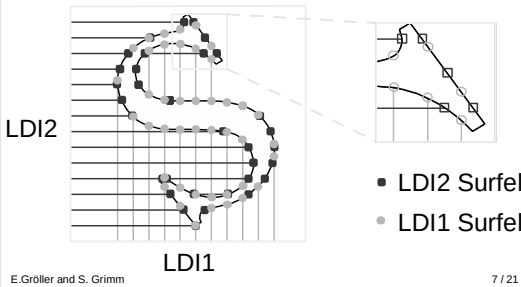
Goal: Optimal Surfel representation of the geometry

→ Layered Depth Cube (Lischinsky et al.)

- Create three orthogonal Layered Depth Images → LDC

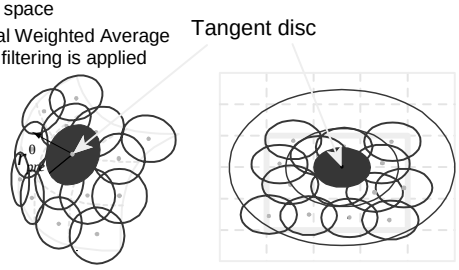
LDC Sampling

- Cast rays from three different directions
- Generate Surfels at intersection points with object

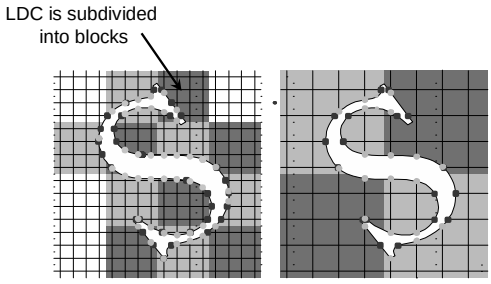


Texture Pre-filtering

- Mapping of tangent disc to texture space
- Elliptical Weighted Average (EWA) filtering is applied



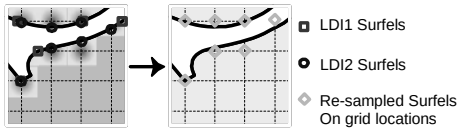
Data Structure/LCD Tree



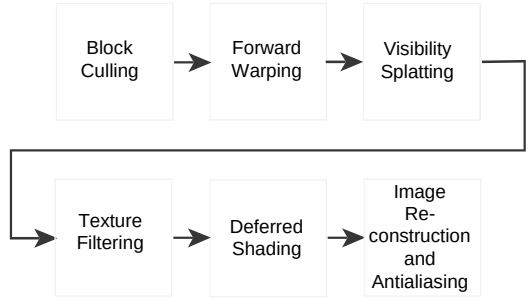
Two different Levels of a LCD Tree

3-to-1 Reduction

- Re-sampling of Surfels on rectilinear grid
- Reduction triples warping speed → 3-1



Rendering



Block Culling

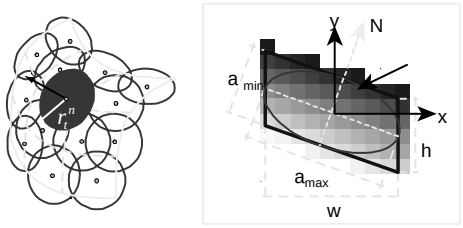
- LDC tree is traversed from top (lowest resolution) to bottom (highest resolution)
  - For each block view frustum culling is performed
  - Visibility cones to perform back-face culling.

Block Warping

- During rendering LDC tree is traversed from top to bottom.
- Octree level is determined by Surfels per pixel
  - Fast rendering: One Surfel per Pixel
  - High Quality rendering: Multiple Surfels per pixel (super-sampling)
- Appropriate octree level is determined by the distance between two Surfels in image space

Visibility Splatting

Z-Buffer projection of Surfel

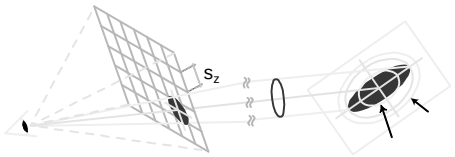


Object Space Z-Buffer

Texture Filtering

Projected Pixel Coverage

Image Space Object Tangent Space

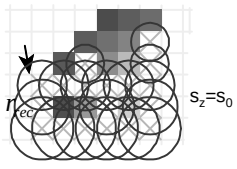


- Surfel color determination by linear interpolation of Surfel mipmaps
- Mayor axis of the projection determines Surfel mipmap level

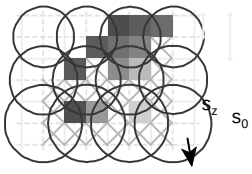
Shading

- Shading after visibility testing: avoiding unnecessary work
- Per-surfel Phong illumination using cube reflectance and environmental maps
- High quality specular highlights: Shading with per-Surfel normals

Image Reconstruction and Antialiasing

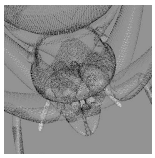


- Surfels are mapped to pixel centers
- Output pixel size == z-Buffer pixel size
- Holes are marked by green crosses → Filled by using symmetric Gaussian filter

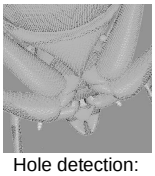


- Super-sampling
- Output pixel size is multiple of z-Buffer size
- Reconstruction by putting a Gaussian filter at the centers of output pixels

Results (1/3)



Surfel object with holes



Hole detection: Green pixels are holes

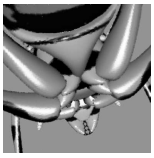
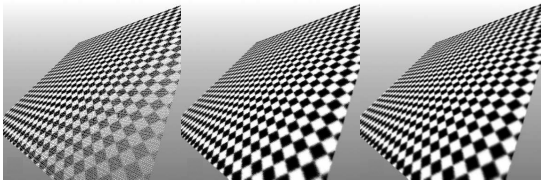


Image reconstruction with Gaussian filter

Results (2/3)

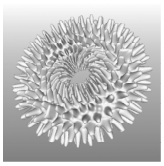
Reconstruction filters



Nearest Neighbor Gaussian Filter Supersampling

There is an increasing number of Surfels per pixel towards the top of the image.

Results (3/3)



Conclusion

- Surfels ideal for: models with very high shape and shade complexity
- Rendering costs are reduced: Moving rasterization and texturing to a preprocessing step
- Rendering performance depends on: warping, shading, and image reconstruction (could be optimized by exploiting vectorization, parallelism, and pipelining)
- Antialiasing and supersampling are naturally integrated into the surfel system
- Surfel rendering is capable of high image quality at interactive framerates