CS 563 Advanced Topics in Computer Graphics
The Use of Points as a Display Primitive

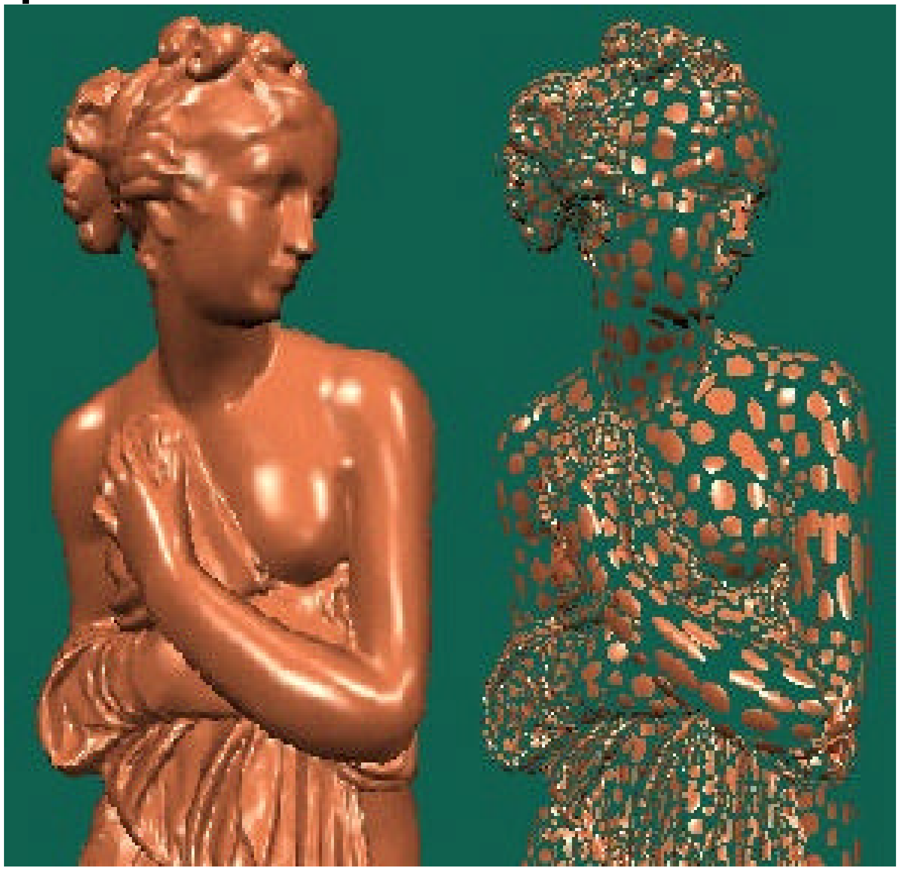
by Jared Krechko

Overview

- What it is
- Why we do it
- How to do it
- Examples and Advances

What is point based rendering?

- Simply, using points to display objects
- First proposed in 1985, recent resurgence



http://www-i8.informatik.rwth-aachen.de/teaching/ws04/seminar/seminar_ws04.html

Why?

- Separates geometry
- Fewer overall points to handle
- Lower memory requirement
- Accurate displays

Contributing theories

- Smoke, trees, clouds and fire already modeled
- Texture mapping
- Bump mapping
- Tabular arrays for terrain
- Object order rendering
- Image order rendering

Problems to solve

- How to render
  - New primitive means new modeling and rendering algorithm
- Model and render at the same time?
- Rendering is then converting from geometric description to new primitive
- Display using standard format

- How to render

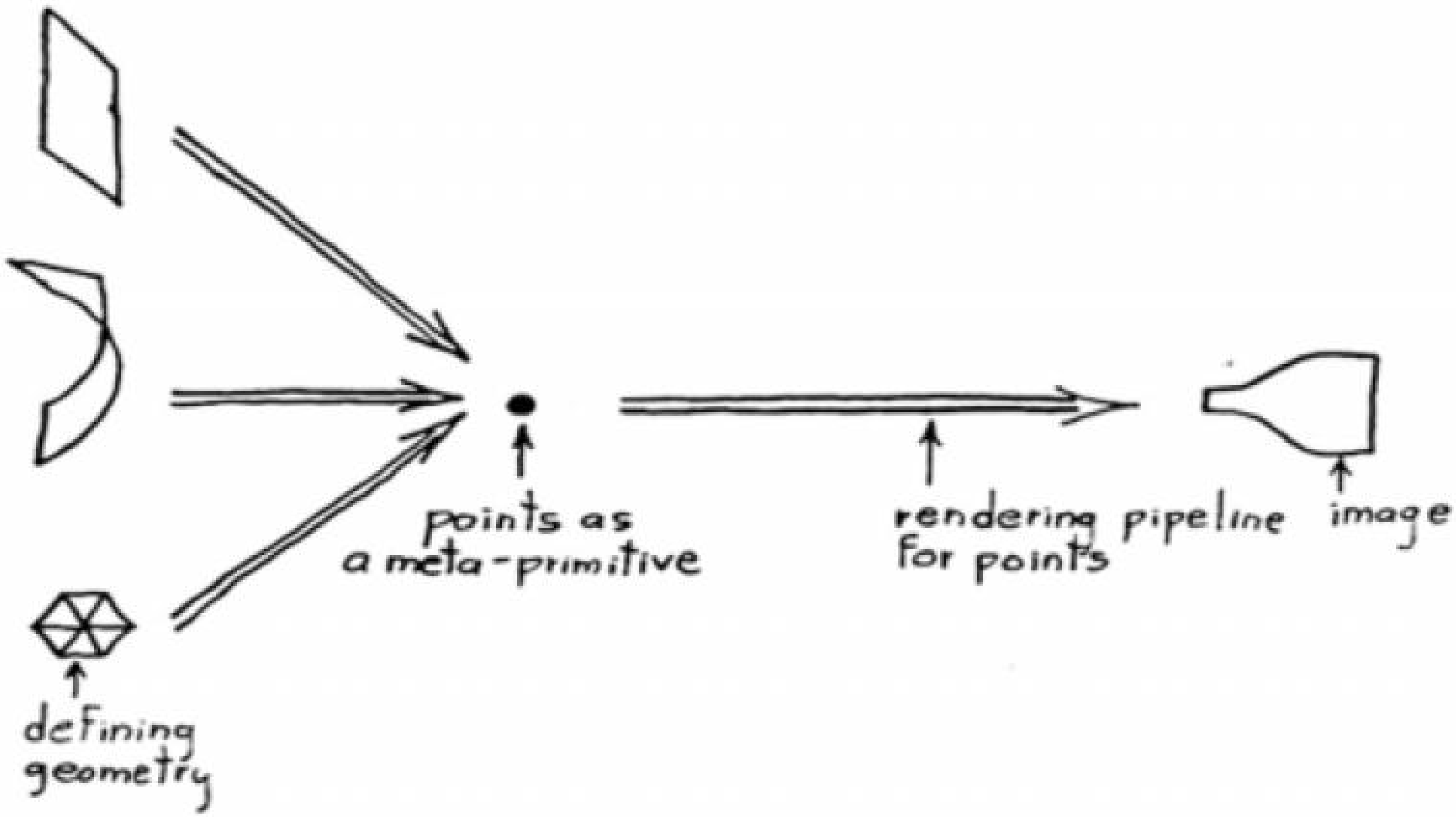


Figure 1: Overview of algorithm

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Object order or image order?

- Object Order
  - Render objects in order in which they are computed
- Image order
  - Construct image pixel-by-pixel
  - Objects contribute to a pixel computed at rendering time
- Which to choose?
  - Object order
    - Correct visibility and filtering

Complexity vs. Coherence

- Geometry = coherence
- Expensive coherence
- Why track extra coherence?
- PBR = no coherence

Overview of How

- Geometry -> points, then render
- Rendering complicated
- Goal: take array of points and display them so they appear continuous
- Texture in interior of point array properly filled
- Edge of array anti-aliased
- Array must obscure its background

Problems?

- No constraint on spatial perturbation
  - Points within array could move anywhere
- Must be able to render randomly

Example

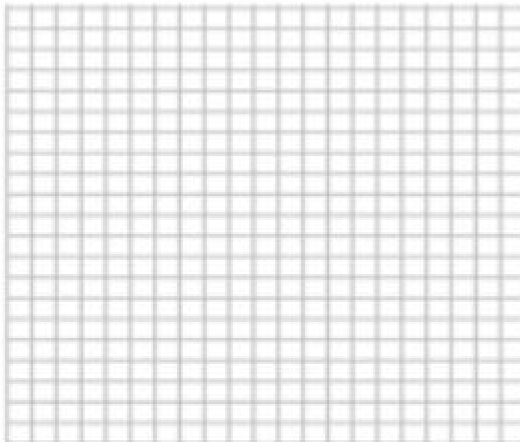


Fig 6a

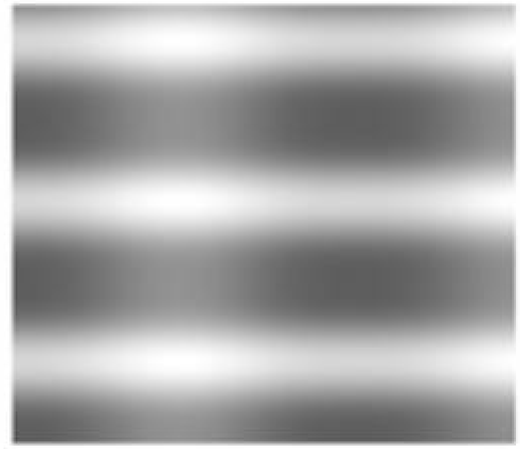


Fig 6b

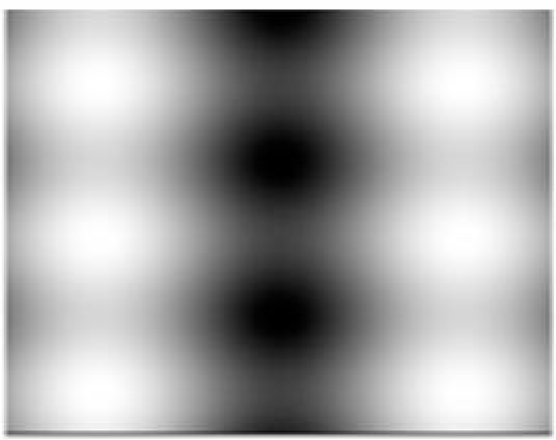


Fig 6c

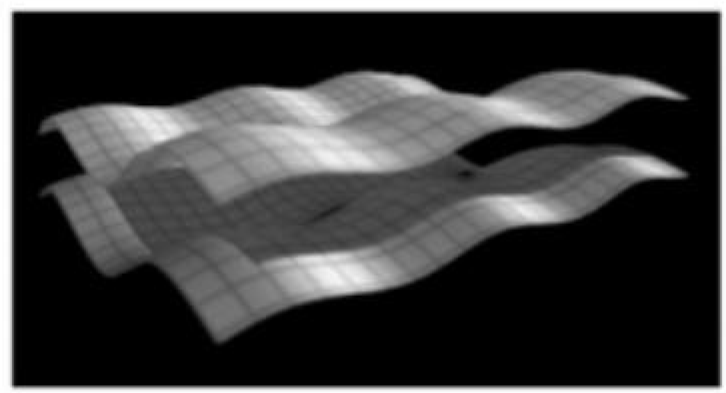


Fig 7

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Point defined

- A *source point* is defined by:
  - (x, y, z, r, g, b, a)
- x, y, and z are spatial attributes
- Any attribute can be perturbed
- *initial grid* - parametric coordinates
- For now, u=x, v=y
- Initial grid is a texture

Selecting points to render

- Each iteration a point is sent through the rendering pipeline
- May choose:
  - Sequentially based on parametric space
  - Procedurally
  - Randomly
- This algorithm uses random

Perturbation

- Any operation which changes an attribute
- Limits
  - Non-spatial attributes - computer
  - Spatial attributes - discontinuous

Transformation and Clipping

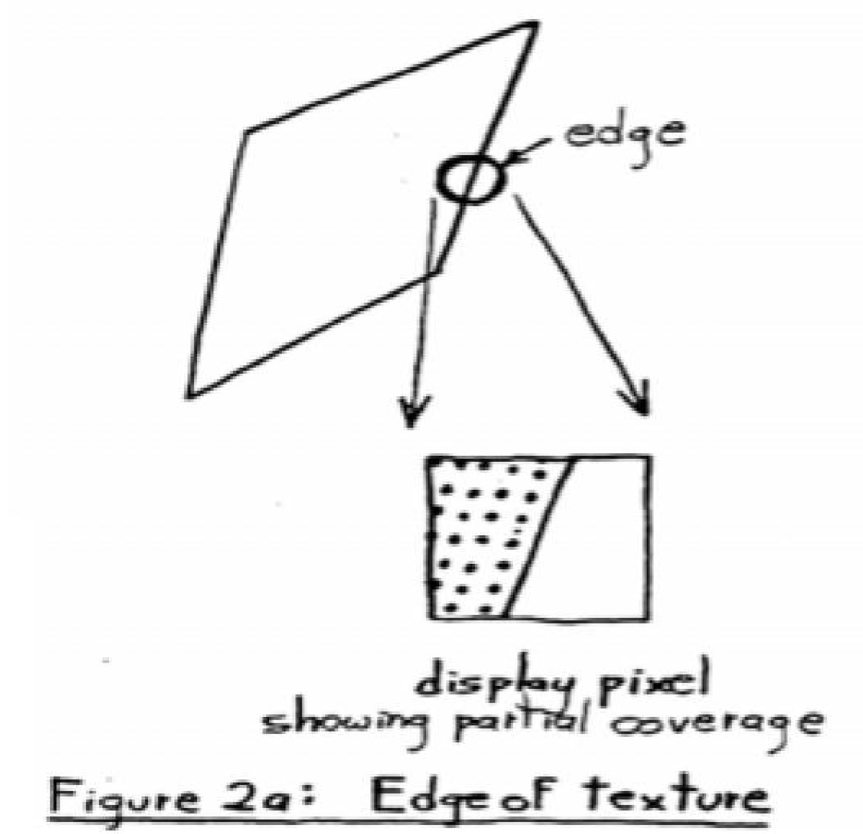
- Transform:
  - Multiply [x, y, z, 1] by 4x4 transform matrix followed by perspective divide
  - Don’t divide z by w so z-clipping can be done
- Clipping:
  - Compare transformed x, y, and z coordinates against a frustum of vision

Density of Points

- Contribution of each source to each pixel proportional to distance from pixel center
- Filter function at each pixel, highest at center
- Radially symmetric Gaussian here
- Contribution computed – distance to pixel weighted

Density of Points

- Edge of Texture



http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Density of Points

- Foldover Points

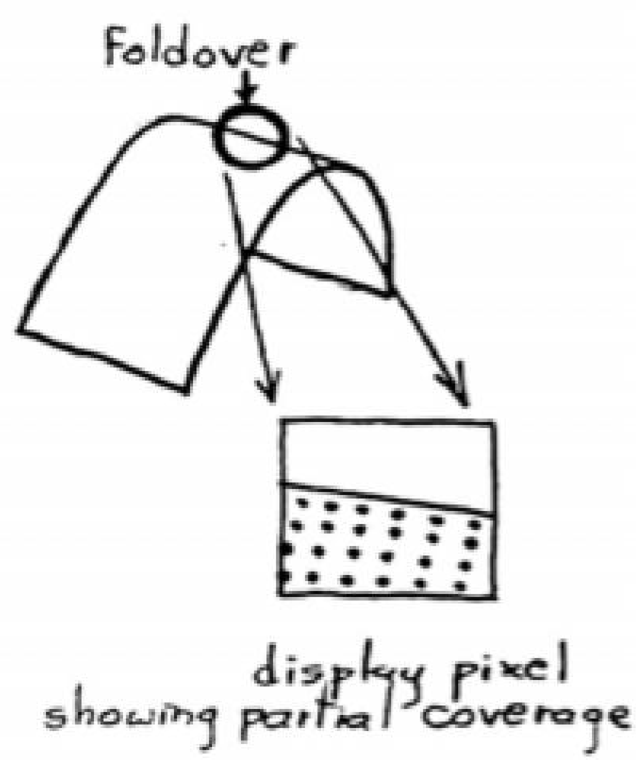


Figure 2b: Foldover

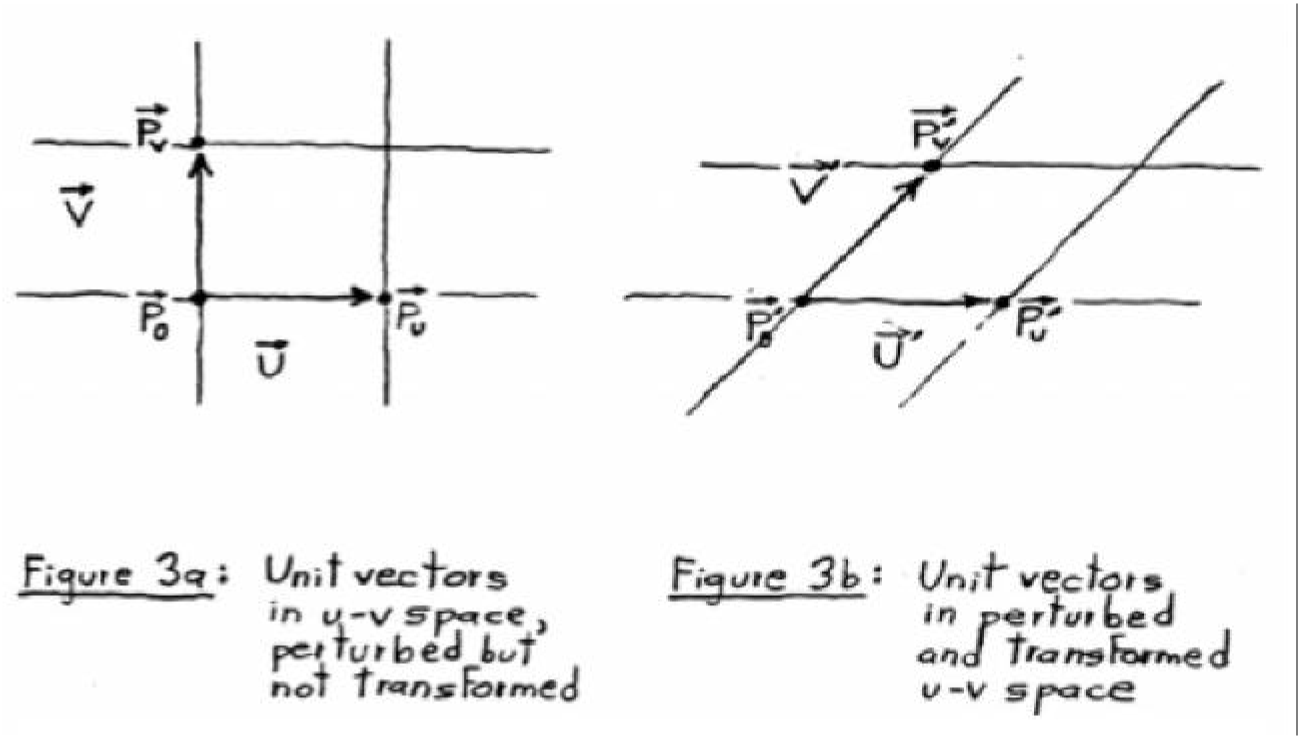
http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Density of Points

- Density of source or partial coverage along edges
- Pre-normalize the contributions
- Sum to unity
- No unity = partial coverage
- Sum of contributions = coverage

Density of Points

- Predicting the density of source points
  - Do it before rendering
  - Use to compute normalizing divisor for weight



http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

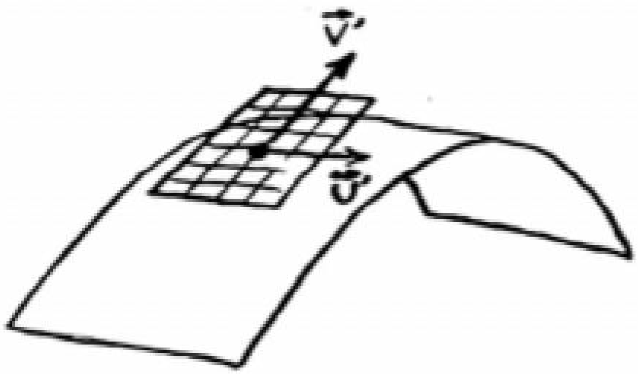


Figure 3c: Tangent plane to surface in small neighborhood

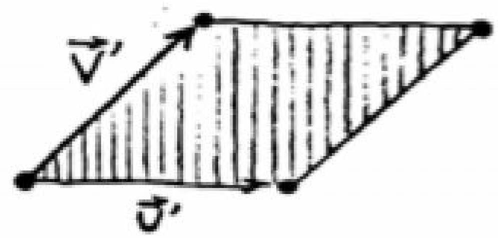


Figure 3d: Area of parallelogram gives density of source points

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Density of Points

$$A = \left| \det \left[J_{\mathbf{F}}(\mathbf{p}_0') \right] \right| = \left| \det \begin{bmatrix} x_u' - x_0' & x_v' - x_0' \\ y_u' - y_0' & y_v' - y_0' \end{bmatrix} \right|$$

- Gives density of source points
- Normalizing divisor for any source point given any view transform
- Interior sum to unity
- Edges sum to coverage

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Error in Density

- Leads to artifacts

$$\varepsilon = \left| \frac{\det\left[J_{\mathbf{F}}(\mathbf{p}_0')\right]}{\det\left[J_{\mathbf{F}}(\mathbf{q}_0')\right]} - 1 \right|$$

- Large E = artifacts
- Really large E = initial resolution insufficient
  - Low pass filter perturbation function
  - Increase spatial resolution of initial grid

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Where we are

- Point and tangent plane -> image space
- Point in image space
- Area point would cover if surface element
- Position in image space separate from display sample points in image plane

Filter Radius

- Function of
  - Source density
  - Display sample density
- Minification
  - Avoid aliasing of source function
- Magnification
  - Avoid aliasing of reconstruction
- Radius decreases as source density increases

Filter Radius

- Function zero beyond small neighborhood
- Cutoff makes contributions vary slightly
- Computed as partial coverage
- Fix by extend Gaussian

Hidden Surface Removal

- Contribution of source points
  - Blending

$$color_{new} = color_{old} + (color_{incoming} \times weight_{incoming})$$

  - Visibility

$$color_{new} = color_{old} \times (1 - \alpha_{incoming}) + color_{incoming} \times \alpha_{incoming}$$

    - Only if blending already done
    - Blending computations more frequent

Bins

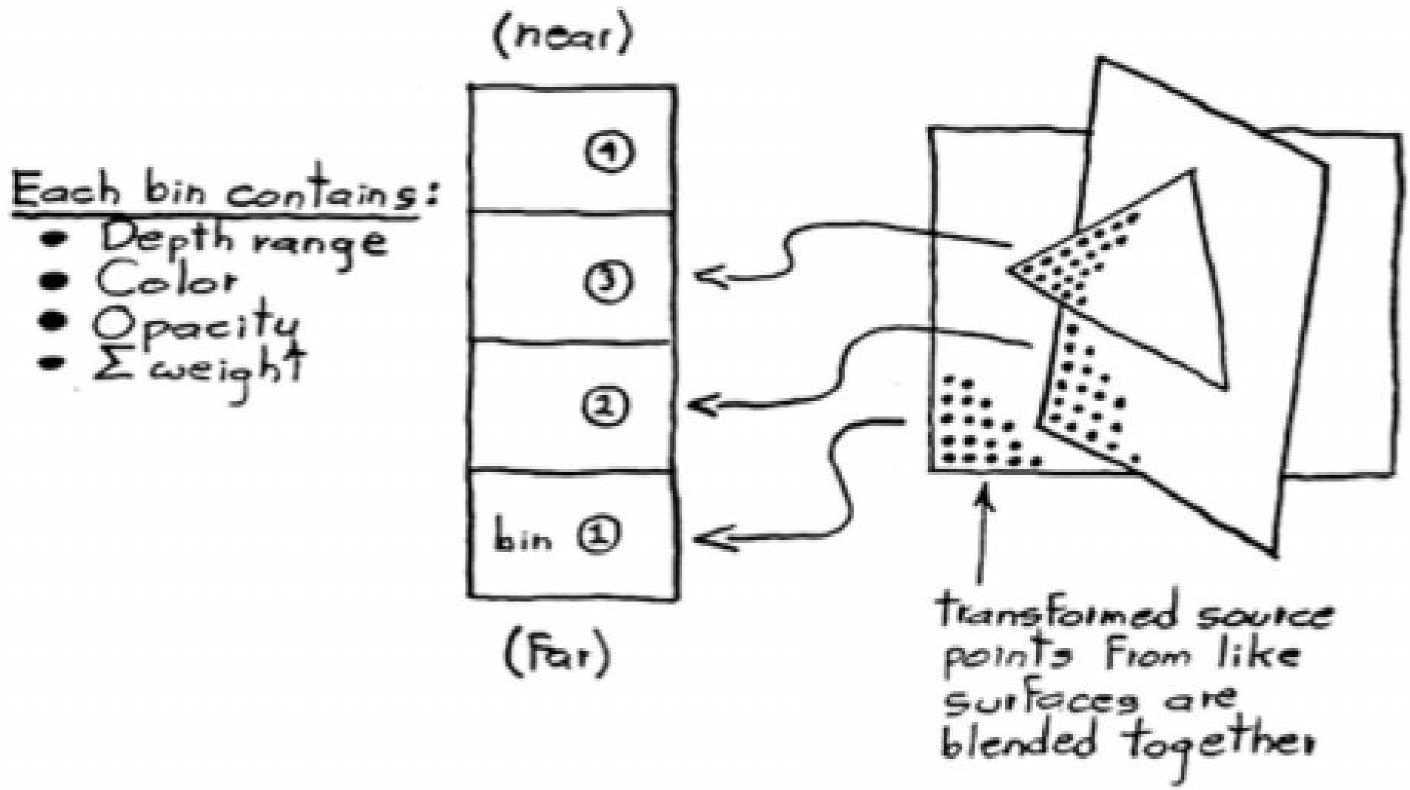


Figure 4a: Blending calculations

- Must check normals before blending

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Bins

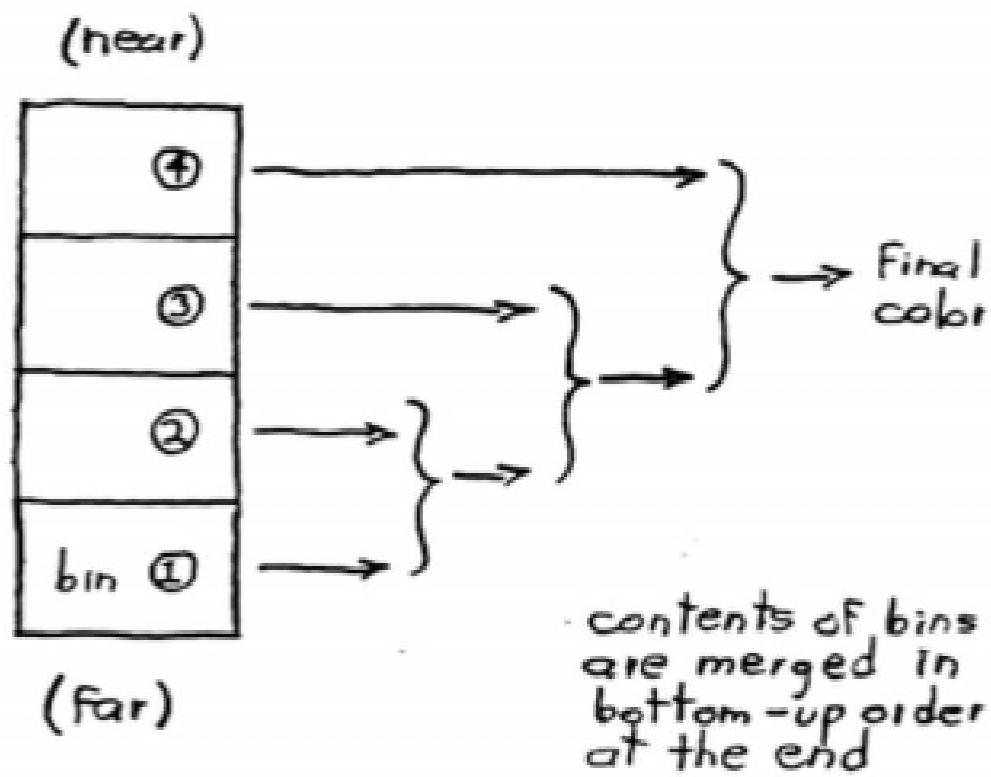
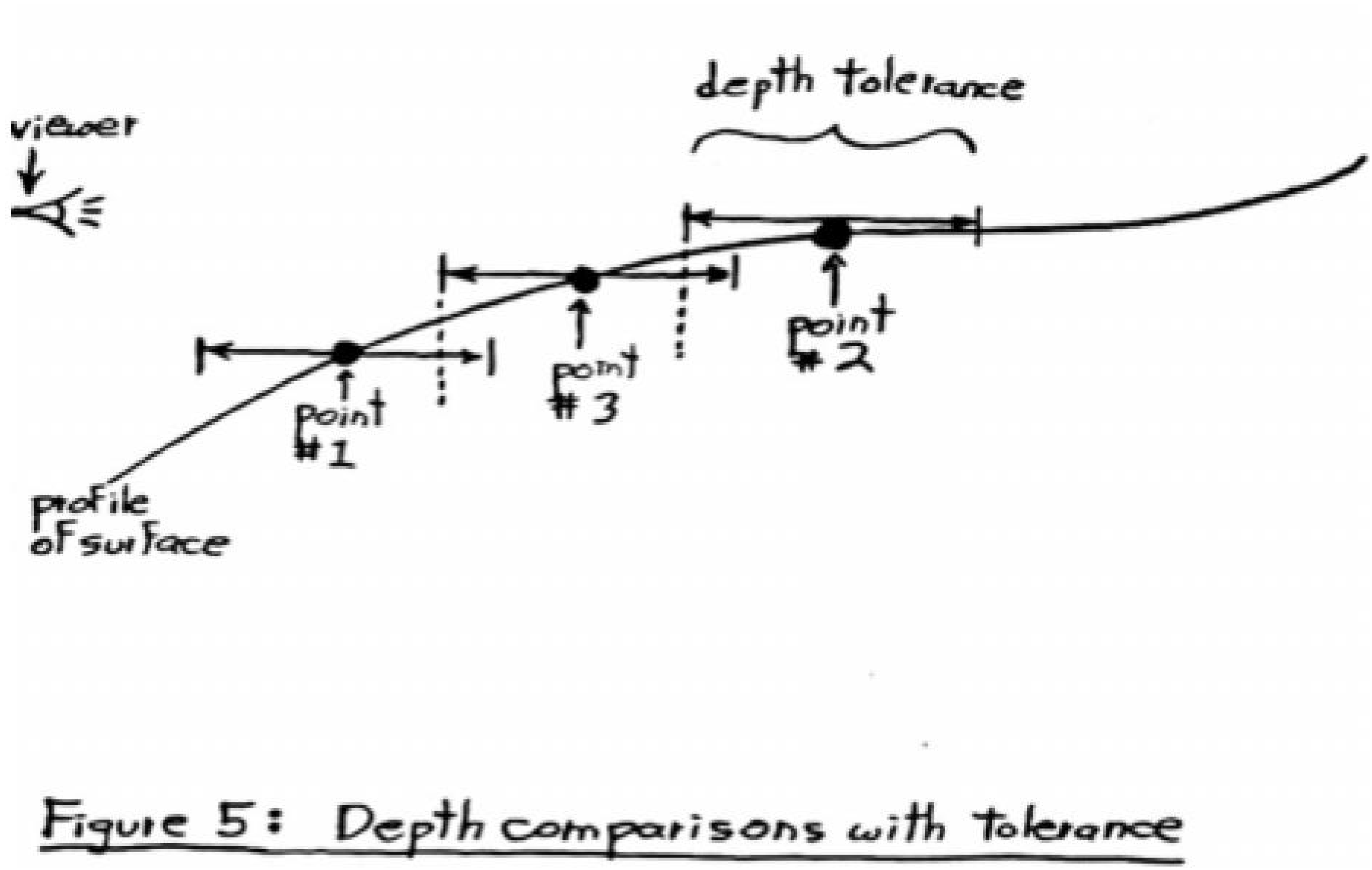


Figure 4b : Visibility calculations

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf



http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf

Finally, Geometry

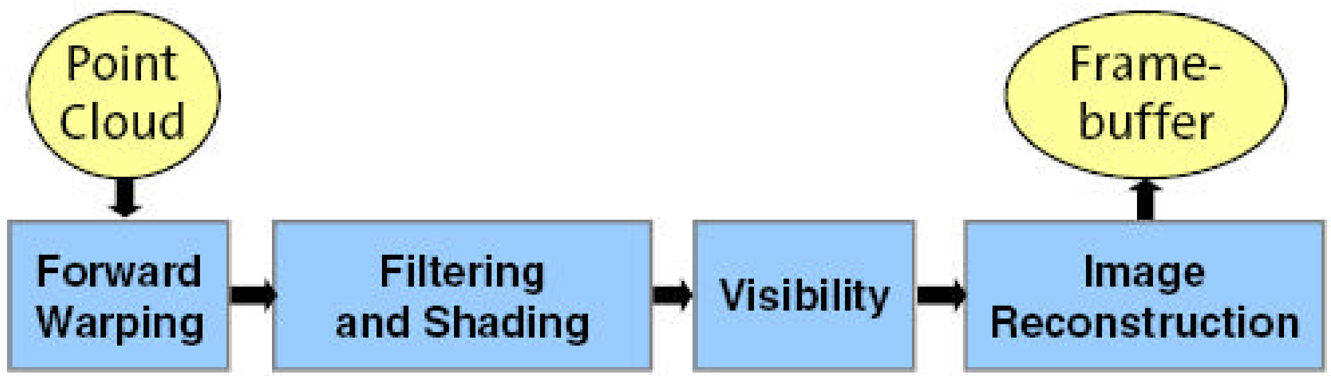
- Valid geometry
  - Break surface into points
  - Continuous and differentiable in small neighborhood around each point
  - Find two non-collinear on a tangent plane approximating surface at point

Valid Geometry

- Allows
  - Polygons
  - Spheres
  - Conic sections
  - Any parametrically defined surface

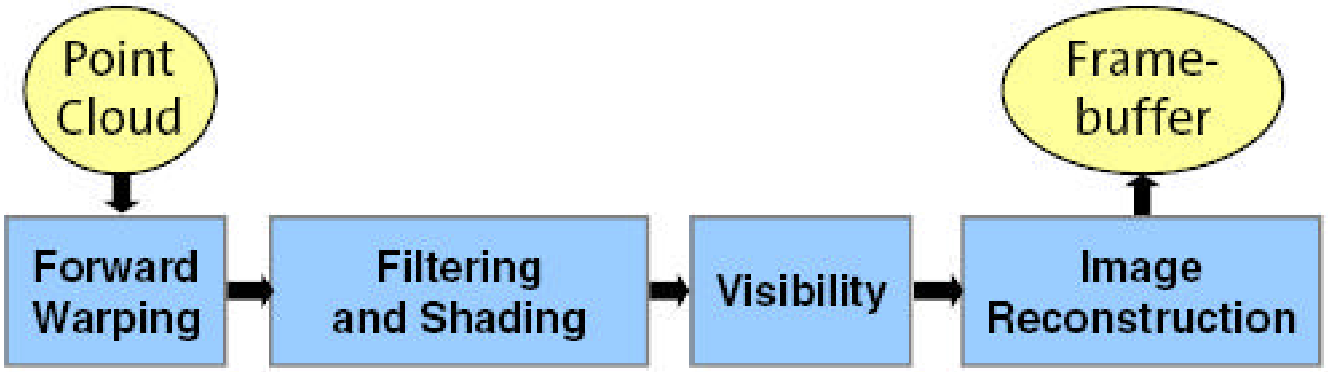
More simple

- Surfels
- Neighborhood data representation



More simple

- Forward Warping = Perspective Projection
- Filtering and Shading
- Last two done simultaneously



http://graphics.ethz.ch/publications/tutorials/eg2002/powerpoint/Rendering.Zwicker.pdf

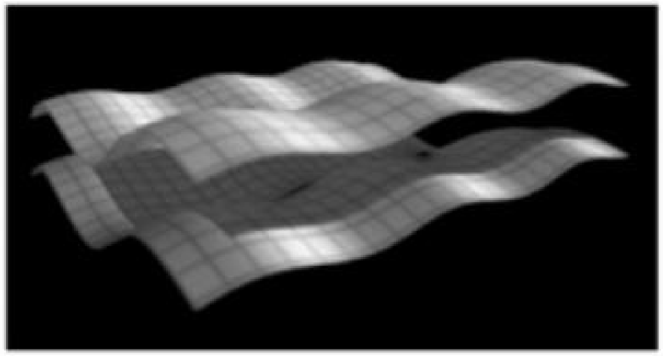


Fig 7

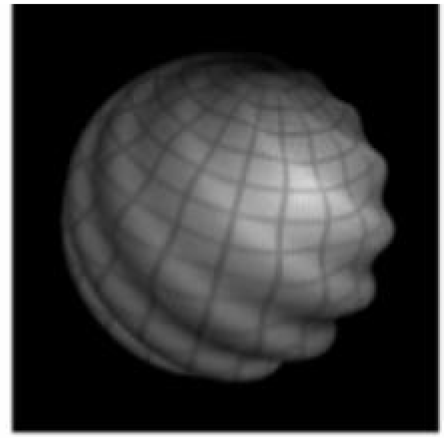


Figure 8

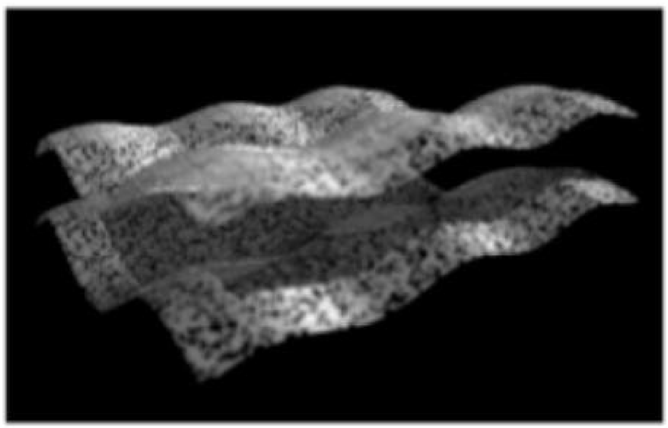


Figure 9b

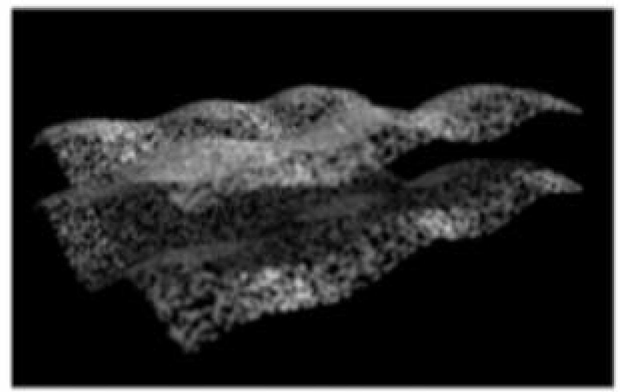
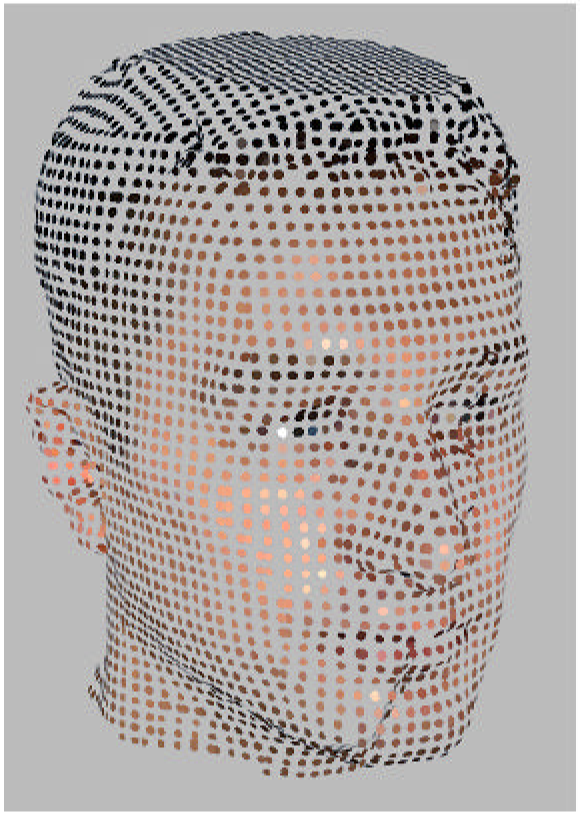
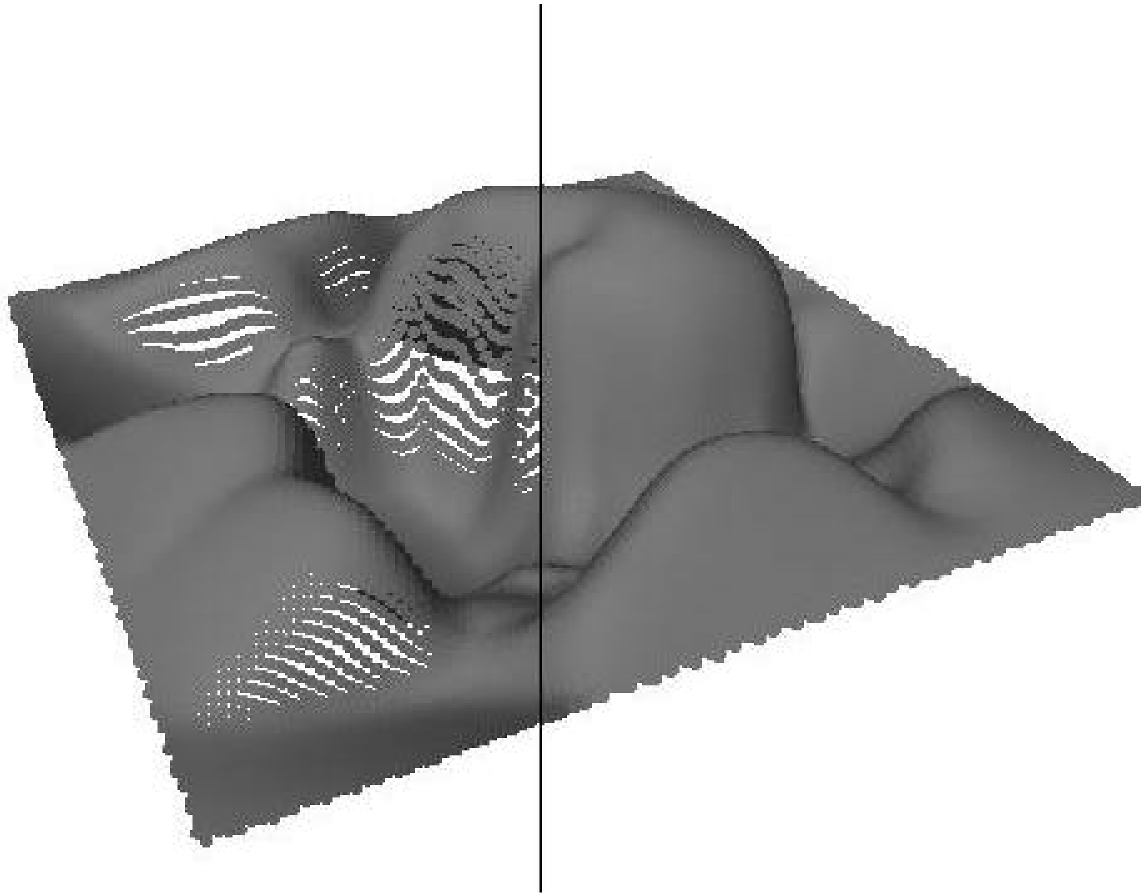


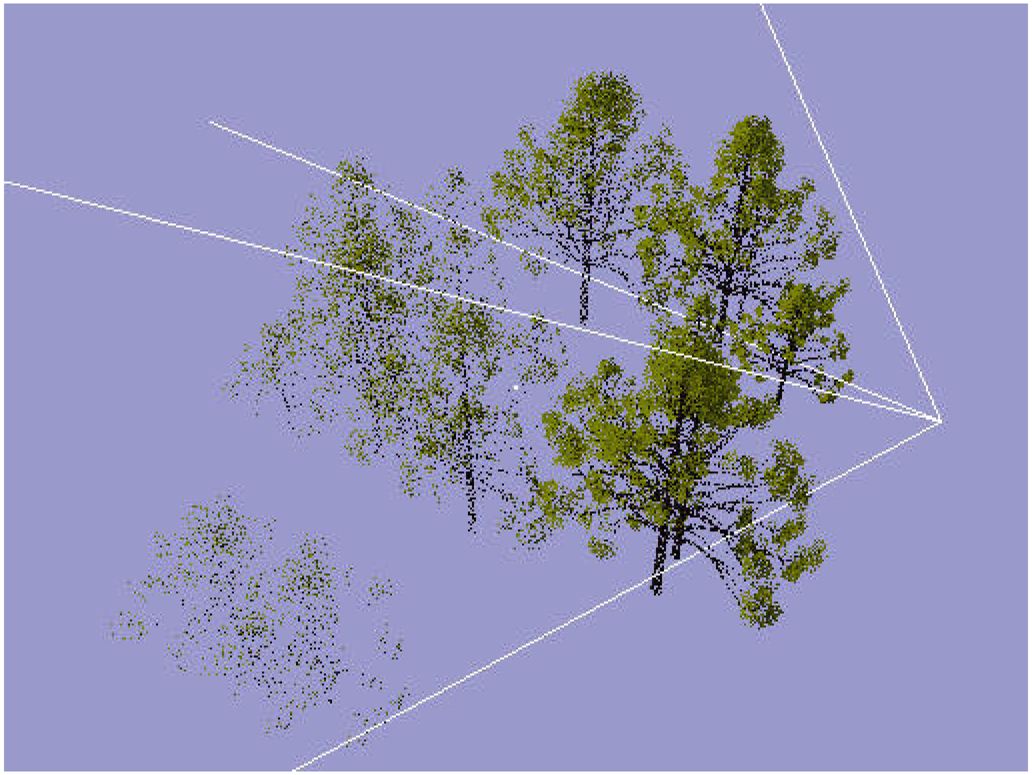
Figure 9a

http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf





http://www-sop.inria.fr/reves/publications/data/2001/SD01/?LANG=gb



Advances

- Splatting (QSplat)
- Depth of Field
- LOD Changes
- Mobile Devices
- More Hardware Support
- Polygon/Point rendering
- Taking advantage of other new algorithms
- Virtual Reality

Conclusions

- Standard rendering algorithm for any geometry
- Rendering in object order
- Arrays of points with no underlying geometry
- Simple primitive, no coherence

Questions?

Questions?

- Marc Stamminger, http://www-sop.inria.fr/reves/Marc.Stamminger/pbr/
- Matthias Zwicker, http://graphics.ethz.ch/publications/tutorials/eg2002/powerpoint/Rendering.Zwicker.pdf
- Marc Levoy, Turner Whitted, http://graphics.stanford.edu/papers/points/point-with-scanned-figs.pdf
- J. Krivanek, http://www.cgg.cvut.cz/~xkrivanj/papers/workshop2003/workshop2003-abstract.pdf
- Liviu Coconu, Hans-Christian Hege, http://delivery.acm.org/10.1145/590000/581903/p43-coconu.pdf?key1=581903&key2=9082482111&coll=GUIDE&dl=GUIDE&CFID=41482871&CFTOKEN=71845390
- Miguel Sainz, Renato Pajarola, Roberto Lario, http://www.ics.uci.edu/~graphics/pub/PointsReloaded.pdf