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

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
- 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
- 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

Density of Points

- Foldover Points

Figure 26: Foldover

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

Figure 3a: Unit vectors in u-v space; perturbed but not transformed
Figure 3b: Unit vectors in perturbed and transformed u-v space

Figure 3c: Tangent plane to surface in small neighborhood

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

\[ A = \left| \det \begin{bmatrix} J_F(p') \\ x' - x_0 \\ y' - y_0 \end{bmatrix} \right| = \left| \det \begin{bmatrix} x' - x_0 & x' - x_0 \\ y' - y_0 & y' - y_0 \end{bmatrix} \right| \]

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

• Error in Density

• Leads to artifacts

\[ \varepsilon = \left| \frac{\det \left[ J_F(p') \right]}{\det \left[ J_F(q_0) \right]} - 1 \right| \]

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

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
- 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
    \[ \text{color}_{\text{new}} = \text{color}_{\text{old}} + (\text{color}_{\text{incoming}} \times \text{weight}_{\text{incoming}}) \]
  - Visibility
    \[ \text{color}_{\text{new}} = \text{color}_{\text{old}} \times (1 - \alpha_{\text{incoming}}) + \text{color}_{\text{incoming}} \times \alpha_{\text{incoming}} \]
  - Only if blending already done
  - Blending computations more frequent
Must check normals before blending

Figure 4b: Visibility calculations

Contents of bins are merged in bottom-up order at the end.
Figure 5: Depth comparisons with tolerance
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
- Surfels
- Neighborhood data representation
- Forward Warping = Perspective Projection
- Filtering and Shading
- Last two done simultaneously

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