CS-184: Computer Graphics

Lecture #25: Modeling w/ Points

Prof. James O'Brien
University of California, Berkeley

Today

- Points as a graphics primitive
A Thought Experiment

- Laser scanners
  - Millions to billions of points
- Typical image
  - At most a few million pixels
- More points than pixels...

“Point-Based Graphics”

- Surfaces represented only by points
  - Maybe normals also
  - No topology
- How can we do
  - Rendering
  - Modeling operations
  - Simulation
Rendering

- For each point draw a little “splat”
  - Use associated normal for shading
  - Possibly apply texture

If “splats” are small compared to spacing then gaps result
Splatting too many points would waste time

Ohtake, et al., SIGGRAPH 2003

Rendering

- “QSplat” algorithm
  - Build hierarchical tree of the points
  - Use bounding spheres to estimate size of clusters
  - Render clusters based on screen size
  - Use cluster-normals for internal nodes

From Rusinkiewicz and Levoy, SIGGRAPH 2000.
Rendering

From Rusinkiewicz and Levoy, SIGGRAPH 2000.
Rendering

From Rusinkiewicz and Levoy, SIGGRAPH 2000.

Rendering

From Rusinkiewicz and Levoy, SIGGRAPH 2000.
Defining a Surface

- Two related methods
  - Surface is a point attractor
    - Point-set surfaces
  - Implicit surface
    - Multi-level Partition of Unity Implicits
    - Implicit Moving Least-Squares

Point-Set Surfaces

- Surface is the attractor of a repeated projection process
  - Find nearby points
  - Fit plane (weighted)
  - Project into plane
  - Repeat
- Does it converge?
- How to weight points?
Does this give us a good surface? 
New “robust” methods exist for sharp features
Point-Set Surfaces

- Some examples

Note shrinkage

Implicit Moving Least-Squares

- Define a scalar function that is zero passing through all the points
Implicit Moving Least-Squares

Function is zero on boundary
Decreases in outward direction


Moving Least-Square Interpolation

Standard Least Square

$$\begin{bmatrix} b^T(p_1) \\ \vdots \\ b^T(p_N) \end{bmatrix} c = \begin{bmatrix} \phi_1 \\ \vdots \\ \phi_N \end{bmatrix}$$
Moving Least-Square Interpolation

Moving Least Square

\[
\begin{bmatrix}
  w(x, p_1) & \cdots & w(x, p_N)
  \\
  b^T(p_1) & \cdots & b^T(p_N)
\end{bmatrix}
\begin{bmatrix}
  c_1 \\
  \vdots \\
  c_N
\end{bmatrix}
= \begin{bmatrix}
  w(x, p_1) & \cdots & w(x, p_N)
  \\
  b^T(p_1) & \cdots & b^T(p_N)
\end{bmatrix}
\begin{bmatrix}
  \phi_1 \\
  \vdots \\
  \phi_N
\end{bmatrix}
\]

\[w(x) = \frac{1}{(x - x_i)^2 + \epsilon^2}\]

\[B^T (W(x))^2 B c(x) = B^T (W(x))^2 \phi\]

Moving Least-Square Interpolation

Least Square

Moving Least Square

Interpolating

Approximating
Interpolating Functions

\[ S_k(x) = \phi_k + (x - p_k)^T \hat{\beta}_k \]
\[ = \psi_{0k} + \psi_{1k} x + \psi_{2k} y + \psi_{3k} z \]
Implicit Moving Least-Squares


Proof of good behavior in Kolluri SODA 2005

(Actually based on polygon constraints... but same idea.)
Partition of Unity Method

Ohtake, et al., SIGGRAPH 2003

Partition of Unity is a special case of Moving Least-Squares
Partition of Unity Method

Editing Operations

- Implicit function can be
  - Combined w/ booleans
  - Warped
  - Offset
  - Composed
  - And more...
Editing Operations

- Implicit function can be
  - Combined w/ booleans
  - Warped
  - Offset
  - Composed
  - And more...

Ohtake, et al., SIGGRAPH 2003
**Smoothing**

Simple Smoothing

Adjustment Smoothing


---

**Point-Based Simulation**

- MLS originated in mechanics literature
- Natural use in graphics for animation

Point-Based Simulation


Point Based Animation of Elastic, Plastic and Melting Objects

Suggested Reading

"QSplat: A Multiresolution Point Rendering System for Large Meshes" by Szymon Rusinkiewicz and Marc Levoy, SIGGRAPH 2000.

"Multi-level Partition of Unity Implicit"s by Yutaka Ohtake and colleagues, SIGGRAPH 2003.

"Point Based Animation of Elastic, Plastic and Melting Objects" by Mueller and colleagues, SCA 2004.

"Defining point-set surfaces" by Nina Amenta and Yong Joo Kil, SIGGRAPH 2004.

"Interpolating and Approximating Implicit Surfaces from Polygon Soup" by Chen Shen, James O’Brien, and Jonathan Shewchuk, SIGGRAPH 2004.