What are Subdivision Methods?

- **Hot Topic** in computer graphics!
- Subdivision methods refine a (simple) control polygon such that in the limit it converges to a smooth curve.
- A multitude of different curve and surface types can be described by subdivision.
- Prediction: Subdivision methods will become the most important tool to model smooth curves and surfaces.

Flexible Modeling...

... and Smooth Surfaces

Basic Idea

- Subdivide the faces of a polyhedron until it converges to a smooth surface.

Example: *Geri’s Game*

- Subdivision surfaces are used for:
  - Geri’s hands and head.
  - Clothes: Jacket, Pants, Shirt.
  - Tie and Shoes.

*Movie*
Sharp Transitions

- Edges and creases of variable crispness.
- Detailed representation of surface features.
- Only possible with substantive NURBs modeling!

(DeRose et al., SIGGRAPH '98.)

---

Sharp Creases

---

Why Subdivision?

- Subdivision methods have a series of interesting properties:
  - Applicable to meshes of arbitrary topology (non-manifold meshes).
  - Scalability, level-of-detail.
  - Numerical stability.
  - Simple implementation.
  - Compact support.
  - Affine invariance.
  - Continuity.

---

Subdivision in 1-D

Simplest example: piecewise linear subdivision

\[ x_2 = \frac{1}{2} (x_1 + x_3) \]

\[ y_2 = \frac{1}{2} (y_1 + y_3) \]
Subdivision in 1-D

- More interesting example: 4pt scheme
  \[ p_{i,j}^{4pt} = \frac{1}{16}(-p_{i+1,j} + 9p_i^j + 9p_{i-1,j}^{} - p_{i-1,j}^{}) \]
- Idee: 4 points determine an interpolating polynomial of degree 3.
- Insert a new point at the parametric midpoint.

Chaikin’s Algorithm

- Piecewise linear interpolation of points.
- Cutting of polygon corners.
- Converges to a quadratic B-spline!

Review: Splines

- Splines are piecewise polynomial curves of some chose degree.
- For example, each polynomial segment of the curve can be written as:
  \[
  \begin{align*}
  x(t) &= a_0t^3 + a_1t^2 + a_2t + a_3 \\
  y(t) &= b_0t^3 + b_1t^2 + b_2t + b_3 
  \end{align*}
  \]

Splines

- \((a, b)\) are constant coefficients which control the shape of the curve over the associated segment.
- This representation uses monomials \((t^3, t^2, t^1, t^0)\) as basis functions.
- In the case of cubic splines one would typically want \(C^2\) continuity.

Parametric Continuity

- \(C^0\): curves are joined
- \(C^1\): first derivatives are equal
  \[ \frac{d}{dt} Q(t) = \text{velocity is the same} \]
- \(C^2\): first and second derivatives are equal
  \[ \frac{d^2}{dt^2} Q(t) = \text{acceleration is the same (important for animation)} \]
**B-Splines**

- Each B-spline has a coefficient known as a control point:
  \[
  x(t) = \sum x_i B_i(t) \\
  y(t) = \sum y_i B_i(t)
  \]
- The new basis function \( B(t) \) is chosen such that the resulting curves are always continuous and that the influence of each control point is local.

**B-Splines Requirements**

- Differentiable to the appropriate order.
- Influence of a control point should be maximal over the region of the curve which is closest to the control point.
- Its influence should decrease as we move away along the curve and disappear at some point.
- Want the basis functions to be piecewise polynomials so that we can represent any piecewise polynomial of a given degree with the associated basis function.

---

**B-Splines of Degree 1**

- Construction through repeated convolution with box function.
  \[
  B_i(t) = \int B_i(s) B_j(t-s) ds = (B_i \otimes B_j)(t)
  \]
  \[
  B_i = \begin{cases}
  1 & t \in (0,1) \\
  0 & t \not\in (0,1)
  \end{cases}
  \]

**B-Spline of Higher Degree**

- A B-spline of degree \( n \) can be obtained by convolving a B-spline of degree \( n-1 \) with the box \( B_0(t) \).
  \[
  B_n(t) = \int B_{n-1}(s) B_0(t-s) ds = (B_{n-1} \otimes B_0)(t)
  \]

---

**B-Splines Features**

- General definition of degree \( n \).
- Piecewise polynomial.
- Local support.
- B-Splines of degree \( n \) are \( C^{n-1} \) continuous.

---

**B-Splines Refinement**

- A B-Spline can be represented as a combination of scaled and translated copies of itself.
- Example: Linear B-Spline.
B-Splines Refinement

- B-splines of degree 0 refinement.
  \[ B_0(t) = B_s(2t) + B_s(2t - 1) \]

- Generally, following binomial theorem:
  \[ b_n(0) = \sum_{k=0}^{n} \binom{n+1}{k} b_0(2t-k) \]

B-Spline Curves

- Linear combination of B-Spline basis functions:
  \[ \gamma(t) = \sum p_i B(t - i) \]

- Refinement of B-Spline curves as summation:
  \[ \gamma(t) = \sum p_i B(t - i) \]

B-Spline Curves: Refinement

- Refinement of B-Spline curves as summation of basis functions:
  \[ \gamma(t) = \sum p_i B(t - i) \]
  \[ = \sum p_i \left( \sum s_j B(2t - 2i - k) \right) \]
  \[ = \sum B(2t - i) \left( \sum s_j p_i \right) \]

Subdivision Vectors

- P = vector of control points:
  \[ P = \begin{bmatrix} \vdots \\ p_2 \\ p_1 \\ p_0 \\ p_1 \\ p_2 \\ \vdots \end{bmatrix} \]

- B(t) = vector of translates of B:
  \[ B(t) = [ \ldots B(t+2) B(t+1) B(t) B(t-1) B(t-2) \ldots ] \]
Subdivision Operator

- Example: Cubic B-Splines

\[
\begin{pmatrix}
1 & 1 & 1 & 1 \\
8 & 4 & 1 & 0 \\
2 & 2 & 1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0
\end{pmatrix}
\]

B-Spline Curves: Refinement

- B-Splines fulfill the refinement relation.
- Basis refinement corresponds to control point refinement.
- Instead of drawing the curve we draw the refined control polygon.
- Subdivision is the refinement of a control polygon!

Continuous Limit Curve

Difference scheme:
- Intuitive definition of convergence: The difference between successive control points has to diminish fast enough.

\[
(\Delta p_{i+1}^j) = p_{i+1}^{j+1} - p_i^j
\]

Convergence (without Proof)

Lemma
- If \(\|D\| < \epsilon \beta^j\) for a constant \(\epsilon > 0\) and a factor \(0 < \beta < 1\), then \(P(t)\) converges to a continuous limit function \(P(t)\).
- A difference scheme exists for a „sensible“ subdivision method (affine invariance).

\[
\Delta p_{i+1}^j = D\Delta p_i^j \\
\|D^n\| = \beta < 1 \\
m > 0
\]
Subdivision Surfaces

Classification criteria:
- Kind of refinement (primal, dual).
- Type of mesh cells (triangles, quadrilaterals, hexagons).
- Approximating or interpolating.

Refinement Rules

- Primal (vertex insertion)
- Dual (corner cutting)

Approximating vs. Interpolating

- Approximating methods
  - Based mostly on splines.
  - Kompact support (small subdivision mask).
- Interpolating methods
  - Control points lie on the subdivision surface.
  - “In-place” implementation.

Important Methods

<table>
<thead>
<tr>
<th></th>
<th>Primal</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Approximating</td>
<td>Loop</td>
<td>Catmull-Clark</td>
</tr>
<tr>
<td>Interpolating</td>
<td>Butterfly</td>
<td>Kobbelt</td>
</tr>
</tbody>
</table>

Subdivison Masks

- “Graphical” representation of subdivision schemes.
- Graph of control points and weights that influence new vertices.
- Different masks for all vertex types.

\[ p''(v) = \frac{3}{8} p'(v_1) + \frac{3}{8} p'(v_2) + \frac{1}{8} p'(v_3) + \frac{1}{8} p'(v_4) \]
Loop I

- Charles Loop, 1987 (M.S. thesis!)
- $C^2$-continuous surfaces on regular meshes.
- $C^1$-continuous for extraordinary vertices.
- In practice, no extra masks are necessary for extraordinary vertices (no guarantees for continuity!).

Loop II

\[
\beta = \frac{3}{8k}, \quad k > 3; \quad \beta = \frac{3}{16}, \quad k = 3
\]

Loop: Example

Catmull-Clark I

- Ed Catmull and Jim Clark, 1978
- Based on bicubic tensor-product B-Spline.
- $C^2$-continuous.
- $C^1$-continuous on extraordinary vertices.
- Arbitrary polygon meshes can be converted to quadrilateral meshes using generalized Catmull-Clark rules.

Catmull-Clark II

\[
\beta = \frac{3}{2k}, \quad \gamma = \frac{1}{4k}
\]

Catmull-Clark: Beispiel
Butterfly I

- Dyn et al., 1990
- $C^1$-continuous on regular meshes.
- Not $C^1$-continuous on extraordinary vertices with valence $k = 3$ and $k > 7$.

Butterfly II

- $\alpha_i = 5/12$
- $\alpha_i = 3/8$
- $\alpha_i = 0$
- $\alpha_i = -1/8$

Doo-Sabin I

- Donald Doo and Malcolm Sabin, 1978
- Based on bi-quadratic tensor-product B-Splines.
- No difference between new and old vertices.
- Only one extra mask for boundary:

Doo-Sabin II

- $d_j' = \frac{d_j + E_j + E_{j+1} + V}{4}$

Doo-Sabin: Beispiel
Implementation

- Algorithm
  - Uniform
  - Adaptive
- Datastructure
  - Quadtree
  - Array
  - Winged-edge

Adaptive Subdivision

- Refinement only if a certain criteria is fulfilled.
- Advantage
  - Runtime
  - Memory footprint
- Disadvantage
  - More complicated implementation
  - Overhead for almost uniform subdivision

Top-level Data

- Store the connectivity (e.g., winged-edge)

Quadtrees I

Quadtrees II