Subdivision curves

Reading

Stollnitz, DeRose, and Salesin. *Wavelets for Computer Graphics: Theory and Applications*, 1996, section 6.1-6.3, A.5.

Subdivision curves

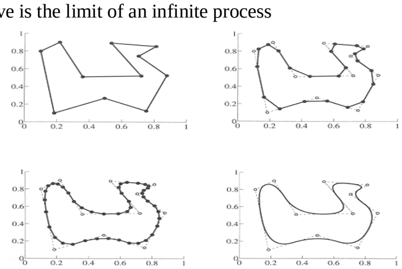
Idea:

- repeatedly refine the control polygon

$$P_0 \rightarrow P_1 \rightarrow P_2 \rightarrow \text{☹}$$

$$C = \lim_{i \to \infty} P_i$$

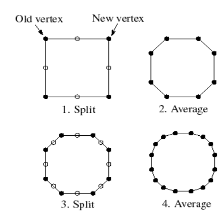
- curve is the limit of an infinite process



Chaikin's algorithm

Chakin introduced the following "corner-cutting" scheme in 1974:

- Start with a piecewise linear curve
- Insert new vertices at the midpoints (the **splitting step**)
- Average each vertex with the "next" neighbor (the **averaging step**)
- Go to the splitting step



Averaging masks

The limit curve is a quadratic B-spline!

Instead of averaging with the nearest neighbor, we can generalize by applying an **averaging mask** during the averaging step:

$$r = (\ldots, r_{-1}, r_0, r_1, \ldots)$$

In the case of Chaikin's algorithm:

$$r =$$

Lane-Riesenfeld algorithm (1980)

Use averaging masks from Pascal's triangle:

$$r = \frac{1}{2^n}\left(\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}\right)$$

Gives B-splines of degree *n*+1.

n=0:

n=1:

n=2:

Subdivide ad nauseum?

After each split-average step, we are closer to the limit surface.

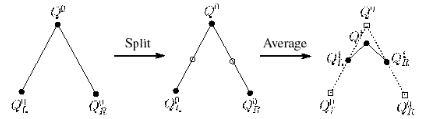
How many steps until we reach the final (limit) position?

Can we push a vertex to its limit position without infinite subdivision? Yes!

Local subdivision matrix

Consider the cubic B-spline subdivision mask: $\frac{1}{4}(1 \quad 2 \quad 1)$

Now consider what happens during splitting and averaging:



Relating points at one subdivision level to points at the previous:

$$Q_L^1 = \frac{1}{2}\left(Q_L^0 + Q^0\right) = \frac{1}{8}\left(4Q_L^0 + 4Q^0\right)$$

$$Q^1 = \frac{1}{8}\left(Q_L^0 + 6Q^0 + Q_R^0\right)$$

$$Q_R^1 = \frac{1}{2}\left(Q^0 + Q_R^0\right) = \frac{1}{8}\left(4Q^0 + 4Q_R^0\right)$$

Local subdivision matrix

We can write this as a recurrence relation in matrix form:

$$\begin{pmatrix} Q_L^j \\ Q^j \\ Q_R^j \end{pmatrix} = \frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}\begin{pmatrix} Q_L^{j-1} \\ Q^{j-1} \\ Q_R^{j-1} \end{pmatrix}$$

$$\mathbf{Q}^j = \mathbf{S}\mathbf{Q}^{j-1}$$

Q's are row vectors and **S** is the **local subdivision matrix**.

Looking at the x-coordinate independently:

$$\begin{pmatrix} x_L^j \\ x^j \\ x_R^j \end{pmatrix} = \frac{1}{8}\begin{pmatrix} 4 & 4 & 0 \\ 1 & 6 & 1 \\ 0 & 4 & 4 \end{pmatrix}\begin{pmatrix} x_L^{j-1} \\ x^{j-1} \\ x_R^{j-1} \end{pmatrix}$$

$$\mathbf{X}^j = \mathbf{S}\mathbf{X}^{j-1}$$

Local subdivision matrix, cont'd

Tracking just the *x* components through subdivision:

$$\mathbf{X}^j = \mathbf{S}\mathbf{X}^{j-1} = \mathbf{S}\cdot\mathbf{S}\mathbf{X}^{j-2} = \mathbf{S}\cdot\mathbf{S}\cdot\mathbf{S}\mathbf{X}^{j-3} = \ldots$$
$$= \mathbf{S}^j\mathbf{X}^0$$

The limit position of the x's is then:

$$X^\infty = \lim_{j\to\infty} S^j X^0$$

OK, so how do we apply a matrix an infinite number of times??

Eigenvectors and eigenvalues

To solve this problem, we need to look at the eigenvectors and eigenvalues of *S*. First, a review…

Let *v* be a vector such that:

$$Sv = \lambda v$$

We say that *v* is an eigenvector with eigenvalue λ.

An *nxn* matrix can have *n* eigenvalues and eigenvectors:

$$Sv_1 = \lambda_1 v_1$$
$$\vdots$$
$$Sv_n = \lambda_n v_n$$

$$X = \sum^n a_i v_i$$

For *non-defective* matrices, the eigenvectors form a basis, which means we can re-write *X* in terms of the eigenvectors:

To infinity, but not beyond…

Now let's apply the matrix to the vector X:

$$SX = S\sum^n a_i v_i = \sum^n a_i S v_i = \sum^n a_i \lambda_i v_i$$

Applying it *j* times:

$$S^j X = S^j \sum^n a_i v_i = \sum^n a_i S^j v_i = \sum^n a_i \lambda_i^j v_i$$

Let's assume the eigenvalues are sorted so that:

$$\lambda_1 > \lambda_2 > \lambda_3 \geq \ldots \geq \lambda_n$$

Now let *j* go to infinity.

If $\lambda_1 > 1$, then…

If $\lambda_1 < 1$, then…

If $\lambda_1 = 1$, then:

$$S^\infty X = \sum^n a_i \lambda_i^\infty v_i = a_1 v_1$$

Evaluation masks

What are the eigenvalues and eigenvectors of our cubic B-spline subdivision matrix?

$$\lambda_1 = 1 \qquad \lambda_2 = \frac{1}{2} \qquad \lambda_3 = \frac{1}{4}$$

$$v_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \qquad v_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \qquad v_3 = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix}$$

We're OK!

But where did the x-coordinates end up?

Evaluation masks, cont'd

To finish up, we need to compute a_1.

It turns out that, if we call v_i the "right eigenvectors" then there are a corresponding set of "left eigenvectors" with the same eigenvalues such that:

$$u_1^T S = \lambda_1 u_1^T$$

M

$$u_n^T S = \lambda_n u_n^T$$

Using the first left eigenvector, we can compute: $x^\infty = a_1 = u_1^T X^0$

In fact, this works at any subdivision level: $x^\infty = S^\infty X^j = u_1^T X^j$

The same result obtains for the y-coordinate: $y^\infty = S^\infty Y^j = u_1^T Y^j$

We call u_i an **evaluation mask**.

Recipe for subdivision curves

The evaluation mask for the cubic B-spline is:

$$\frac{1}{6}(1 \quad 4 \quad 1)$$

Now we can cook up a simple procedure for creating subdivision curves:

- Subdivide (split+average) the control polygon a few times. Use the averaging mask.
- Push the resulting points to the limit positions. Use the evaluation mask.

Question: what is the tangent to the curve?

Answer: apply the second left eigenvector, u_2, as a tangent mask.

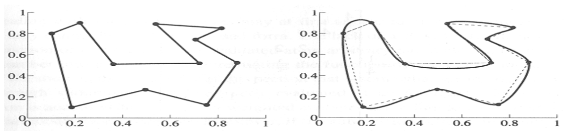
DLG interpolating scheme (1987)

Slight modification to algorithm:

- splitting step introduces midpoints
- averaging step *only changes midpoints*

For DLG (Dyn-Levin-Gregory), use:

$$r = \frac{1}{16}(-2, 6, 10, 6, -2)$$



Since we are only changing the midpoints, the points after the averaging step do not move.

Summary

What to take home:

- How to perform the splitting and averaging steps
- What an evaluation mask is and how to use it
- An appreciation for the mathematics behind subdivision curves