

## 14. Subdivision curves

1

## Reading

Recommended:

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

2

## Subdivision curves

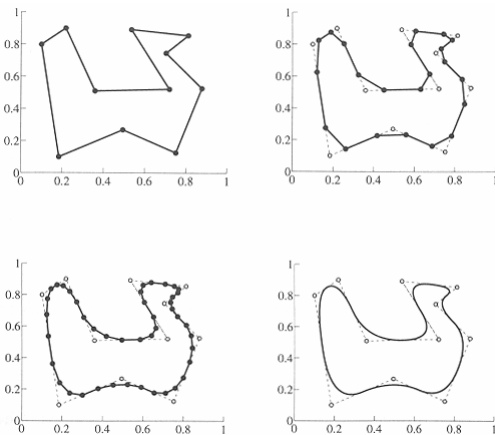
Idea:

- ♦ repeatedly refine the control polygon

$$P^1 \rightarrow P^2 \rightarrow P^3 \rightarrow \dots$$

- ♦ curve is the limit of an infinite process

$$Q = \lim_{i \rightarrow \infty} P^i$$

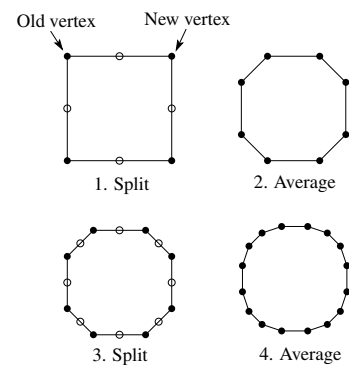


3

## Chaikin's algorithm

Chaikin 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



4

## 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:

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

In the case of Chaikin's algorithm:

$$r =$$

5

## Lane-Riesenfeld algorithm (1980)

Use averaging masks from Pascal's triangle:

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

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

$n=0$ :

$n=1$ :

$n=2$ :

6

## Subdivide ad nauseum?

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

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

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

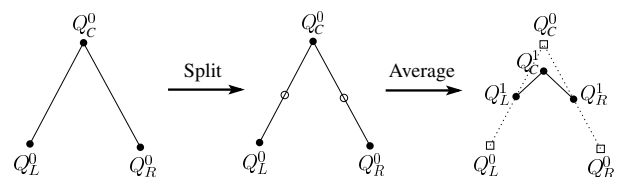
7

## Local subdivision matrix

Consider the cubic B-spline subdivision mask:

$$\frac{1}{4} \begin{pmatrix} 1 & 2 & 1 \end{pmatrix}$$

Now consider what happens during splitting and averaging:



We can write equations that relate points at one subdivision level to points at the previous:

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

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

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

8

## Local subdivision matrix

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

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

$$Q^j = SQ^{j-1}$$

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

We can think about the behavior of each coordinate independently. For example, the x-coordinate:

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

$$X^j = SX^{j-1}$$

9

## Local subdivision matrix, cont'd

Tracking just the x components through subdivision:

$$X^j = SX^{j-1} = S \cdot SX^{j-2} = S \cdot S \cdot SX^{j-3} = \dots = S^j X^0$$

The limit position of the x's is then:

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

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

10

## 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  $\lambda$ .

An  $n \times n$  matrix can have  $n$  eigenvalues and eigenvectors:

$$\begin{aligned} Sv_1 &= \lambda_1 v_1 \\ &\vdots \\ Sv_n &= \lambda_n v_n \end{aligned}$$

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

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

11

## To infinity, but not beyond...

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

$$X^1 = SX^0 = S \sum_i^n a_i v_i = \sum_i^n a_i S v_i = \sum_i^n a_i \lambda_i v_i$$

Applying it  $j$  times:

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

Let's assume the eigenvalues are sorted so that:

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

Now let  $j$  go to infinity:

$$X^\infty = \lim_{j \rightarrow \infty} S^j X^0 = \lim_{j \rightarrow \infty} \sum_i^n a_i \lambda_i^j v_i$$

If  $\lambda_1 > 1$ , then:

If  $\lambda_1 < 1$ , then:

If  $\lambda_1 = 1$ , then:

12

## Evaluation masks

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

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

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

We're OK!

But where did the x-coordinates end up?

What about the y-coordinates?

13

## Evaluation masks, cont'd

To finish up, we need to compute  $a_j$ . First, we can reorganize the expansion of  $X$  into the eigenbasis:

$$X^0 = a_1 v_1 + a_2 v_2 + \dots + a_n v_n = \begin{bmatrix} \vdots & \vdots & \dots & \vdots \\ v_1 & v_2 & \dots & v_n \\ \vdots & \vdots & \dots & \vdots \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \mathbf{V} \mathbf{A}$$

We can then solve for the coefficients in this new basis:

$$\mathbf{A} = \mathbf{V}^{-1} X^0$$

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \dots & u_1^T & \dots \\ \dots & u_2^T & \dots \\ \vdots & \vdots & \vdots \\ \dots & u_n^T & \dots \end{bmatrix} X^0$$

Now we can compute the limit position of the x-coordinate:

$$x^\infty = a_1 = u_1^T X^0$$

We call  $u_j$  the **evaluation mask**.

14

## Evaluation masks, cont'd

It's easy to show that the  $u$ -vectors obey:

$$u_i^T S = \lambda_i u_i^T$$

These are the "left eigenvectors" of  $S$ .

Note that we need not start with the 0<sup>th</sup> level control points and push them to the limit. If we subdivide and average the control polygon  $j$  times, we can push the vertices of the refined polygon to the limit as well:

$$x^\infty = S^\infty X^j = u_1^T X^j$$

The same result obtains for the y-coordinate:

$$y^\infty = S^\infty Y^j = u_1^T Y^j$$

15

## Recipe for subdivision curves

The evaluation mask for the cubic B-spline is:

$$\frac{1}{6} \begin{pmatrix} 1 & 4 & 1 \end{pmatrix}$$

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.

16

## Tangent analysis

What is the tangent to the cubic B-spline curve?

First, let's consider how we represent the x and y coordinate neighborhoods:

$$X^0 = a_1 v_1 + a_2 v_2 + a_3 v_3$$

$$Y^0 = b_1 v_1 + b_2 v_2 + b_3 v_3$$

We can view the point neighborhoods then as:

$$\begin{bmatrix} Q_L^0 \\ Q_C^0 \\ Q_R^0 \end{bmatrix} = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} v_1 + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} v_2 + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} v_3$$

or:

$$\begin{bmatrix} Q_L^0 \\ Q_C^0 \\ Q_R^0 \end{bmatrix} = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \begin{bmatrix} v_{1,L} \\ v_{1,C} \\ v_{1,R} \end{bmatrix} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \begin{bmatrix} v_{2,L} \\ v_{2,C} \\ v_{2,R} \end{bmatrix} + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \begin{bmatrix} v_{3,L} \\ v_{3,C} \\ v_{3,R} \end{bmatrix}$$

The tangent to the curve is along the direction:

$$\mathbf{t} = \lim_{j \rightarrow \infty} (Q_R^j - Q_C^j)$$

17

## Tangent analysis (cont'd)

But, this vector is vanishing as the "right" point approaches the center point. However, the *normalized* tangent direction should be defined:

$$\mathbf{t} = \lim_{j \rightarrow \infty} \frac{Q_R^j - Q_C^j}{\|Q_R^j - Q_C^j\|}$$

Now, let's look at the "right" point in isolation:

$$Q_R^0 = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} v_{1,R} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} v_{2,R} + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} v_{3,R}$$

After  $j$  applications of the subdivision matrix:

$$Q_R^j = \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \lambda_1^j v_{1,R} + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \lambda_2^j v_{2,R} + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \lambda_3^j v_{3,R}$$

Subtracting out the center point:

$$\begin{aligned} Q_R^j - Q_C^j &= \begin{bmatrix} a_1 \\ b_1 \end{bmatrix} \lambda_1^j (v_{1,R} - v_{1,C}) + \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \lambda_2^j (v_{2,R} - v_{2,C}) + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \lambda_3^j (v_{3,R} - v_{3,C}) \\ &= \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \lambda_2^j (v_{2,R} - v_{2,C}) + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \lambda_3^j (v_{3,R} - v_{3,C}) \end{aligned}$$

18

## The tangent mask

And now computing the tangent:

$$\begin{aligned} \lim_{j \rightarrow \infty} \frac{Q_R^j - Q_C^j}{\|Q_R^j - Q_C^j\|} &= \lim_{j \rightarrow \infty} \frac{\begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \lambda_2^j (v_{2,R} - v_{2,C}) + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \lambda_3^j (v_{3,R} - v_{3,C})}{\left\| \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \lambda_2^j (v_{2,R} - v_{2,C}) + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \lambda_3^j (v_{3,R} - v_{3,C}) \right\|}} \\ &= \lim_{j \rightarrow \infty} \frac{\begin{bmatrix} a_2 \\ b_2 \end{bmatrix} (v_{2,R} - v_{2,C}) + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \left( \frac{\lambda_3}{\lambda_2} \right)^j (v_{3,R} - v_{3,C})}{\left\| \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} (v_{2,R} - v_{2,C}) + \begin{bmatrix} a_3 \\ b_3 \end{bmatrix} \left( \frac{\lambda_3}{\lambda_2} \right)^j (v_{3,R} - v_{3,C}) \right\|}} \\ &= \frac{\begin{bmatrix} a_2 \\ b_2 \end{bmatrix} (v_{2,R} - v_{2,C})}{\left\| \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} (v_{2,R} - v_{2,C}) \right\|}} \\ &= \frac{\begin{bmatrix} a_2 \\ b_2 \end{bmatrix}}{\left\| \begin{bmatrix} a_2 \\ b_2 \end{bmatrix} \right\|}} \\ &= \frac{(u_2^T Q^0)^T}{\|u_2^T Q^0\|} \end{aligned}$$

Thus, we can compute the tangent using the *second* left eigenvector! This analysis holds for general subdivision curves gives us the **tangent mask**.

19

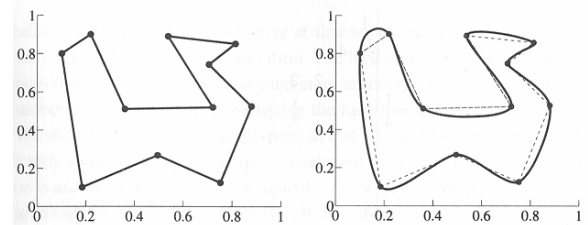
## DLG interpolating scheme (1987)

Slight modification to subdivision 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.

20