# Subdivision curves and surfaces

Brian Curless
CSE 557
Fall 2014

## Reading

Required:

- Stollnitz, DeRose, and Salesin. *Wavelets for Computer Graphics: Theory and Applications,* 1996, section 6.1-6.3, 10.2, A.5.

Note: there is an error in Stollnitz, et al., section A.5. Equation A.3 should read:

$$\mathbf{MV} = \mathbf{V}\Lambda$$

This is already fixed in the handout.

# Subdivision curves

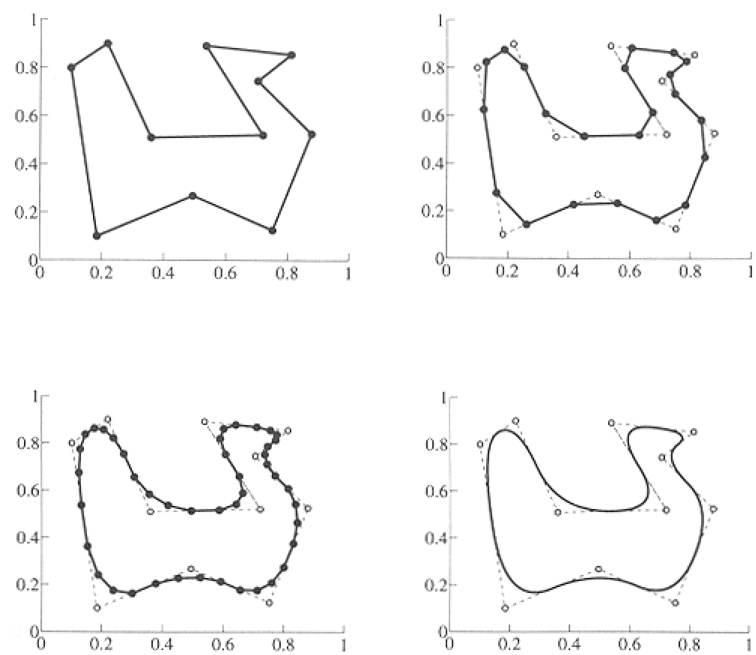
Idea:

- repeatedly refine the control polygon

$$P_1 \to P_2 \to P_3 \to \cdots$$

- curve is the limit of an infinite process

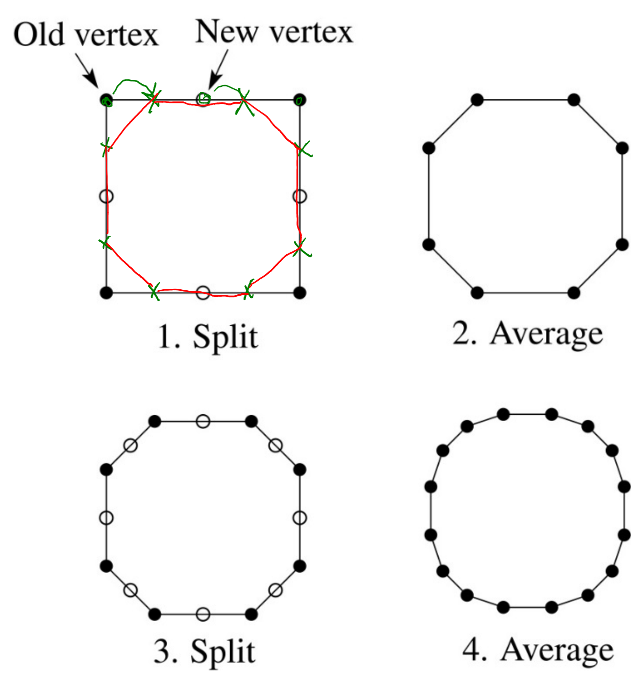
$$Q = \lim_{j \to \infty} P_j$$



## 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" (clockwise) 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:

$$r = \begin{bmatrix} \ldots & r_{-1} & r_0 & r_1 & \end{bmatrix}$$

In the case of Chaikin's algorithm:

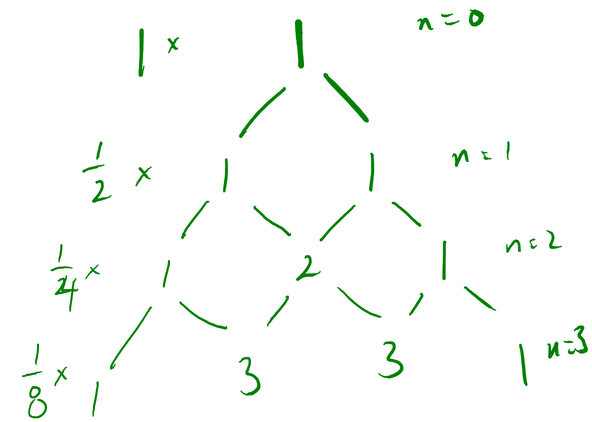
$$r = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}$$

## Lane-Riesenfeld algorithm (1980)

Use averaging masks from Pascal's triangle:

$$r = \frac{1}{2^n}\left[\binom{n}{0} \quad \binom{n}{1} \quad \cdots \quad \binom{n}{n}\right]$$




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

n=0: $[\,1\,]$ linear

n=1: $[\,\frac{1}{2} \quad \frac{1}{2}\,]$ quadratic

n=2: $[\,\frac{1}{4} \quad \frac{1}{2} \quad \frac{1}{4}\,]$ cubic

## 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?

$\infty$

Can we push a vertex to its limit position in one step?

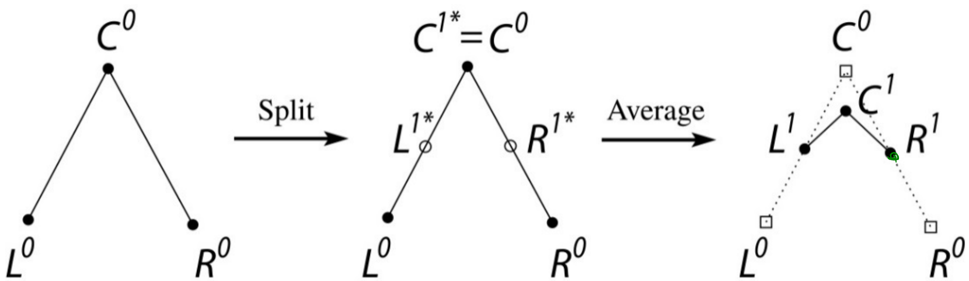
Y

# Local subdivision matrix

Consider the cubic B-spline subdivision mask:

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

Now consider what happens during splitting and averaging in a small neighborhood:




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

$L^{1*} = \frac{1}{2}L^0 + \frac{1}{2}C^0 \qquad C^{1*} = C^0 \qquad R^{1*} = \frac{1}{2}C^0 + \frac{1}{2}R^0$

$$L^1 = \frac{1}{4}L^0 + \frac{1}{2}L^{1*} + \frac{1}{4}C^0$$
$$= \frac{1}{4}L^0 + \frac{1}{4}L^0 + \frac{1}{4}C^0 + \frac{1}{4}C^0$$
$$= \frac{1}{2}L^0 + \frac{1}{2}C^0$$

$$R^1 = \frac{1}{2}C^0 + \frac{1}{2}R^0$$

$$C^1 = \frac{1}{4}L^{1*} + \frac{1}{2}C^0 + \frac{1}{4}R^{1*}$$
$$= \frac{1}{4}\left(\frac{1}{2}L^0 + \frac{1}{2}C^0\right) + \frac{1}{2}C^0 + \frac{1}{4}\left(\frac{1}{2}C^0 + \frac{1}{2}R^0\right)$$
$$= \frac{1}{8}L^0 + \frac{3}{4}C^0 + \frac{1}{8}R^0$$

# Local subdivision matrix

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

$$\begin{bmatrix} L_j \\ C_j \\ R_j \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/8 & 3/4 & 1/8 \\ 0 & 1/2 & 1/2 \end{bmatrix} \begin{bmatrix} L_{j-1} \\ C_{j-1} \\ R_{j-1} \end{bmatrix}$$

where the *L, R, C*'s are (for convenience) *row* vectors.

In 2D, we can write out all the elements as follows:

$$\underbrace{\begin{bmatrix} L_j^x & L_j^y & 1 \\ C_j^x & C_j^y & 1 \\ R_j^x & R_j^y & 1 \end{bmatrix}}_{A_j} = \underbrace{\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/8 & 3/4 & 1/8 \\ 0 & 1/2 & 1/2 \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} L_{j-1}^x & L_{j-1}^y & 1 \\ C_{j-1}^x & C_{j-1}^y & 1 \\ R_{j-1}^x & R_{j-1}^y & 1 \end{bmatrix}}_{A_{j-1}}$$

We can re-write this as:

$$\mathbf{A}_j = \mathbf{M}\mathbf{A}_{j-1}$$

and **M** is the **local subdivision matrix**.

# Local subdivision matrix, cont'd

Starting from the initial control polygon, we can track the original vertex and its original neighborhood through subdivision:

$$\mathbf{A}_j = \mathbf{M}\mathbf{A}_{j-1} = M(MA_{j-2}) = M(M(MA_{j-3})) = \cdots = M^j A_0$$

$$M^2 A_{j-2} \qquad M^3 A_{j-3}$$

The limit position of the neighborhood is then:

$$\mathbf{A}_\infty = \lim_{j\to\infty} \mathbf{M}^j \mathbf{A}_0$$

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

## Eigenvectors and eigenvalues

We now need to look at the eigenvectors and eigenvalues of **M**. Let **v** be a vector such that:

$$\mathbf{Mv} = \lambda \mathbf{v}$$

We say that **v** is an eigenvector of **M** with eigenvalue $\lambda$.

A 3x3 matrix can have 3 eigenvalues and eigenvectors:

$$\mathbf{Mv}_1 = \lambda_1 \mathbf{v}_1$$
$$\mathbf{Mv}_2 = \lambda_2 \mathbf{v}_2$$
$$\mathbf{Mv}_3 = \lambda_3 \mathbf{v}_3$$

In matrix form:

$$\mathbf{M}\begin{bmatrix} & & \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ & & \end{bmatrix} = \begin{bmatrix} & & \\ \lambda_1\mathbf{v}_1 & \lambda_2\mathbf{v}_2 & \lambda_3\mathbf{v}_3 \\ & & \end{bmatrix}$$

$$= \begin{bmatrix} & & \\ \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \\ & & \end{bmatrix}\begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix}$$

$$\mathbf{MV} = \mathbf{V}\Lambda$$

# To infinity, but not beyond…

$$\Lambda^2 = \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} = \begin{bmatrix} \lambda_1^2 & 0 & 0 \\ 0 & \lambda_2^2 & 0 \\ 0 & 0 & \lambda_3^2 \end{bmatrix}$$

Now let's apply **M** to original neigborhood $\mathbf{A}_0$:

$$\mathbf{A}_1 = \mathbf{M}\mathbf{A}_0 = MVV^{-1}A_0 = V\Lambda V^{-1}A_0$$

Now let's advance another subdivision:

$$\mathbf{A}_2 = \mathbf{M}^2\mathbf{A}_0 = \mathbf{M}\mathbf{A}_1 = MV\Lambda V^{-1}A_0 = V\Lambda\Lambda V^{-1}A_0 = V\Lambda^2 V^{-1}A_0$$

Do it j times:

$$\mathbf{A}_j = \mathbf{M}^j\mathbf{A}_0 = V\Lambda^j V^{-1}A_0 = V\begin{bmatrix} \lambda_1^j & 0 & 0 \\ 0 & \lambda_2^j & 0 \\ 0 & 0 & \lambda_3^j \end{bmatrix} V^{-1}A_0$$

What if we do this an infinite number of times?

$$\mathbf{A}_\infty = \mathbf{M}^\infty\mathbf{A}_0 = V\begin{bmatrix} \lambda_1^\infty & 0 & 0 \\ 0 & \lambda_2^\infty & 0 \\ 0 & 0 & \lambda_3^\infty \end{bmatrix} V^{-1}A_0$$

Let's assume the eigenvalues are non-negative and sorted so that:

$$\lambda_1 > \lambda_2 > \lambda_3 \geq \cdots \geq \lambda_n \geq 0$$

If $\lambda_1 > 1$, then: blows up

If $\lambda_1 < 1$, then: implodes

If $\lambda_1 = 1$, then: stable

## Evaluation masks

For cubic B-splines, the local subdivision matrix **M** is:

$$\mathbf{M} = \begin{pmatrix} 1/2 & 1/2 & 0 \\ 1/8 & 3/4 & 1/8 \\ 0 & 1/2 & 1/2 \end{pmatrix}$$

It's eigenvalues and eigenvectors are:

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

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \qquad \mathbf{v}_2 = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \qquad \mathbf{v}_3 = \begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix}$$

$\lambda_1 = 1 > \lambda_2 > \lambda_3$, so we're OK!

We can write out $\Lambda$ and **V**:

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} \qquad \mathbf{V} = \begin{bmatrix} 1 & -1 & 2 \\ 1 & 0 & -1 \\ 1 & 1 & 2 \end{bmatrix}$$

We will also need $\mathbf{V}^{-1}$, which turns out to be:

$$\mathbf{V}^{-1} = \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ -1/2 & 0 & 1/2 \\ 1/6 & -1/3 & 1/6 \end{bmatrix}$$

# Evaluation masks (cont'd)

So, we have:

$$\Lambda = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} \qquad \mathbf{V} = \begin{bmatrix} 1 & -1 & 2 \\ 1 & 0 & -1 \\ 1 & 1 & 2 \end{bmatrix} \qquad \mathbf{V}^{-1} = \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ -1/2 & 0 & 1/2 \\ 1/6 & -1/3 & 1/6 \end{bmatrix}$$

We can now compute the limit position of the neighborhood $\mathbf{A}_0$ :

$$\mathbf{A}_\infty = \mathbf{M}^\infty \mathbf{A}_0 = \mathbf{V}\Lambda^\infty \mathbf{V}^{-1}\mathbf{A}_0$$

$$d_{center}^T = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}$$

$$d_{tangent}^T = \begin{bmatrix} 0 & -1 & 1 \end{bmatrix}$$

$$d_{2nd\ deriv}^T = \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & -1 & 2 \\ 1 & 0 & -1 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ -1/2 & 0 & 1/2 \\ 1/6 & -1/3 & 1/6 \end{bmatrix} A_0$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ -1/2 & 0 & 1/2 \\ 1/6 & -1/3 & 1/6 \end{bmatrix} A_0$$

$$= \begin{bmatrix} 1/6 & 2/3 & 1/6 \\ 1/6 & 2/3 & 1/6 \\ 1/6 & 2/3 & 1/6 \end{bmatrix} \begin{bmatrix} L^0 \\ C^0 \\ R^0 \end{bmatrix}$$

$$C^\infty = \frac{1}{6} L^0 + \frac{2}{3} C^0 + \frac{1}{6} R^0$$

## Recipe for subdivision curves

The row vector $\mathbf{u}_1^T$ that pushes the original vertex to the limit position is called the **evaluation mask**:

$$\mathbf{u}_1^T = \begin{bmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{bmatrix}$$

Note that we do ***not*** need start with the 0th 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:

$$\mathbf{A}_\infty = \mathbf{M}^\infty \mathbf{A}_j = \mathbf{u}_1^T \mathbf{A}_j$$

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.

$$U = V^{-1} = \begin{bmatrix} u_1^T \\ u_2^T \\ u_3^T \end{bmatrix}$$

$$MV = V\Lambda$$

$$V^{-1}MV = V^{-1}V\Lambda$$

$$V^{-1}MV = \Lambda$$

$$V^{-1}MVV^{-1} = \Lambda V^{-1}$$

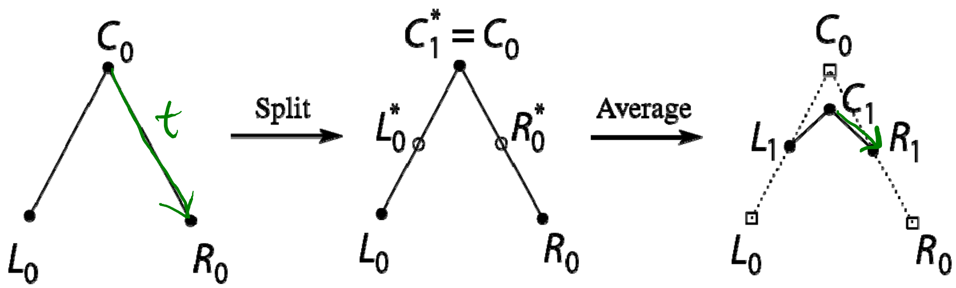
$$V^{-1}M = \Lambda V^{-1}$$

$$UM = \Lambda U$$

$$\begin{bmatrix} u_1^T \\ u_2^T \\ u_3^T \end{bmatrix} M = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \lambda_3 \end{bmatrix} \begin{bmatrix} u_1^T \\ u_2^T \\ u_3^T \end{bmatrix}$$

$$u_1^T M = \lambda_1 u_1^T$$

## Tangent analysis




$$A_j = \begin{bmatrix} L_j^x & L_j^y & 1 \\ C_j^x & C_j^y & 1 \\ R_j^x & R_j^y & 1 \end{bmatrix}$$

$$t_j = R_j - C_j = \begin{bmatrix} 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} L_j^x & L_j^y & 1 \\ C_j^x & C_j^y & 1 \\ R_j^x & R_j^y & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 1 \end{bmatrix} A_j = d^T A_j$$

$$d = \begin{bmatrix} 0 \\ -1 \\ 1 \end{bmatrix}$$

$$t_j = d^T A_j = d^T V \Lambda^j V^{-1} A_0 = d^T \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} \Lambda^j V^{-1} A_0$$

$$= \begin{bmatrix} \underbrace{d^T v_1}_{w_1} & \underbrace{d^T v_2}_{w_2} & \underbrace{d^T v_3}_{w_3} \end{bmatrix} \begin{bmatrix} \lambda_1^j & & \\ & \lambda_2^j & \\ & & \lambda_3^j \end{bmatrix} \underbrace{V^{-1}}_{U^T} A_0$$

$$w_1 = d^T v_1 = \begin{bmatrix} 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = 0$$

$$= \begin{bmatrix} \lambda_1^j w_1 & \lambda_2^j w_2 & \lambda_3^j w_3 \end{bmatrix} \begin{bmatrix} u_1^T \\ u_2^T \\ u_3^T \end{bmatrix} A_0$$

$$= \left( \lambda_1^j w_1 u_1^T + \lambda_2^j w_2 u_2^T + \lambda_3^j w_3 u_3^T \right) A_0$$

## Tangent analysis (cont'd)

$$\mathbf{t}_j = \left(\cancel{\lambda_1^j w_1 \mathbf{u}_1^T} + \lambda_2^j w_2 \mathbf{u}_2^T + \lambda_3^j w_3 \mathbf{u}_3^T\right) A_0 \qquad \lambda_1 > \underline{\lambda_2 > \lambda_3} \geq \cdots \geq \lambda_n \geq 0$$

$$\hat{t}_j = \frac{\left(\lambda_2^j w_2 u_2^T + \lambda_3^j w_3 u_3^T\right) A_0 \cdot 1/\lambda_2^j}{\|\left(\lambda_2^j w_2 u_2^T + \lambda_3^j w_3 u_3^T\right) A_0\| \cdot 1/\lambda_2^j}$$

$$= \frac{\left(w_2 u_2^T + (\lambda_3/\lambda_2)^j w_3 u_3^T\right) A_0}{\|\left(w_2 u_2^T + (\lambda_3/\lambda_2)^j w_3 u_3^T\right) A_0\|} \qquad \frac{\lambda_3}{\lambda_2} < 1$$

$$\lim_{j \to \infty} \hat{t}_j = \frac{w_2 u_2^T A_0}{\|(w_2 u_2^T A_0\|}$$

$$= \frac{u_2^T A_0}{\|u_2^T A_0\|}$$

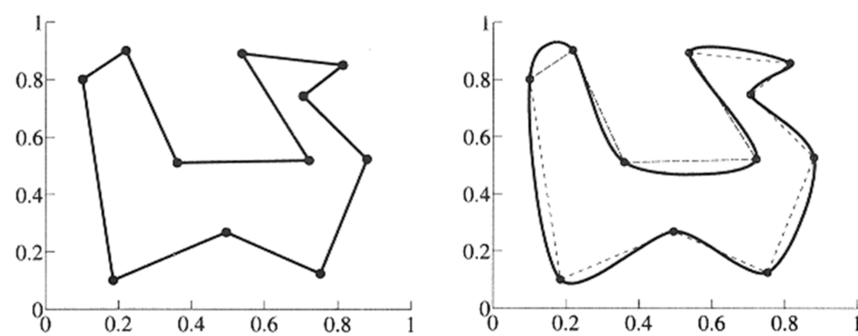
## DLG interpolating scheme (1987)

Slight modification to subdivision algorithm:

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

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

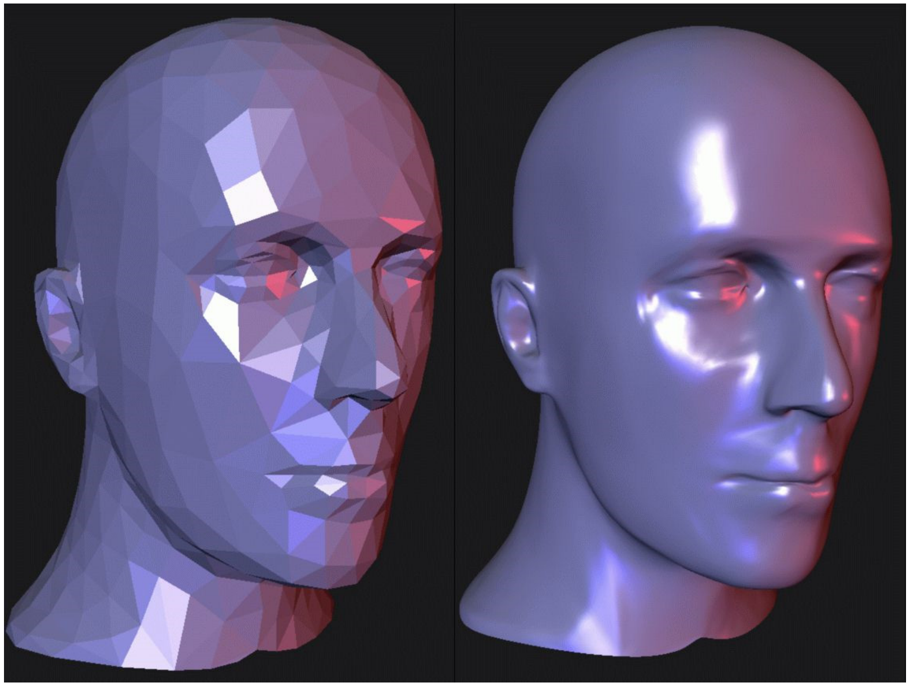
$$r_{\text{old}} = (1) \qquad r_{\text{new}} = \frac{1}{16}(-2,5,10,5,-2)$$




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

## Building complex models

We can extend the idea of subdivision from curves to surfaces…



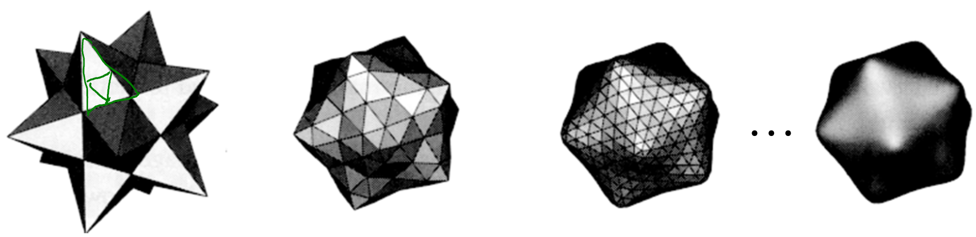
# Subdivision surfaces

Chaikin's use of subdivision for curves inspired similar techniques for subdivision surfaces.

Iteratively refine a **control polyhedron** (or **control mesh**) to produce the limit surface

$$S = \lim_{j \to \infty} P_j$$

using splitting and averaging steps.

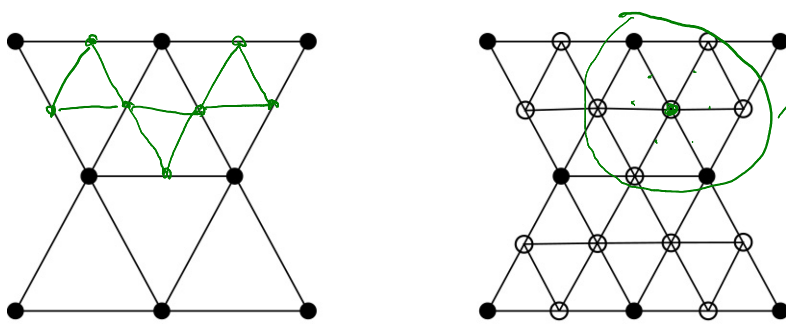


$P_0$ $P_1$ $P_2$ $\cdots$ $P_\infty$

## Triangular subdivision

There are a variety of ways to subdivide a poylgon mesh.

A common choice for triangle meshes is 4:1 subdivision – each triangular face is split into four smaller triangles:



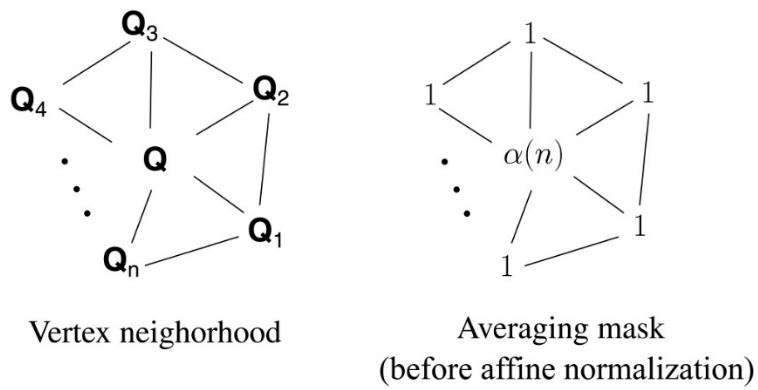
Original After splitting

1-ring neighborhood

valence = 6

# Loop averaging step

Once again we can use **masks** for the averaging step:




$$\mathbf{Q} \leftarrow \frac{\alpha(n)\mathbf{Q} + \mathbf{Q}_1 + \cdots + \mathbf{Q}_n}{\alpha(n) + n}$$

where

$$\alpha(n) = \frac{n(1-\beta(n))}{\beta(n)} \qquad \beta(n) = \frac{5}{4} - \frac{(3 + 2\cos(2\pi / n))^2}{32}$$

These values, due to Charles Loop, are carefully chosen to ensure smoothness – namely, tangent plane or normal continuity.

Note: tangent plane continuity is also know as $G^1$ continuity for surfaces.

# Loop evaluation and tangent masks

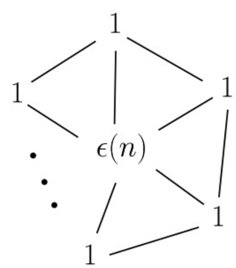
Curves

$\lambda_1 > \lambda_2 > \lambda_3 \cdots \geq 0$

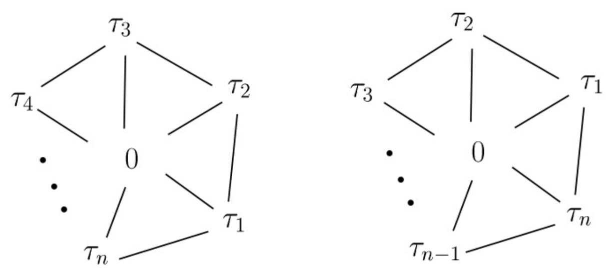
As with subdivision curves, we can split and average a number of times and then push the points to their limit positions.

Surfaces

$\lambda_1 > \lambda_2 = \lambda_3 > \cdots$




Evaluation mask
(before affine normalization)




Tangent masks

$$\mathbf{Q}^\infty = \frac{\varepsilon(n)\mathbf{Q} + \mathbf{Q}_1 + \cdots + \mathbf{Q}_n}{\varepsilon(n) + n}$$

$$\mathbf{T}_1^\infty = \tau_1(n)\mathbf{Q}_1 + \tau_2(n)\mathbf{Q}_2 + \cdots + \tau_n(n)\mathbf{Q}_n$$

$$\mathbf{T}_2^\infty = \tau_n(n)\mathbf{Q}_1 + \tau_1(n)\mathbf{Q}_2 + \cdots + \tau_{n-1}(n)\mathbf{Q}_n$$

where

$$\varepsilon(n) = \frac{3n}{\beta(n)} \qquad \tau_i(n) = \cos(2\pi i/n)$$

How do we compute the normal?

$T_1^\infty \times T_2^\infty$

## Recipe for subdivision surfaces

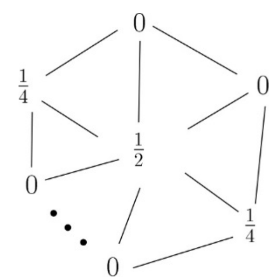
As with subdivision curves, we can now describe a recipe for creating and rendering subdivision surfaces:

- Subdivide (split+average) the control polyhedron a few times. Use the averaging mask.
- Compute two tangent vectors using the tangent masks.
- Compute the normal from the tangent vectors.
- Push the resulting points to the limit positions. Use the evaluation mask.
- Render!

# Adding creases without trim curves

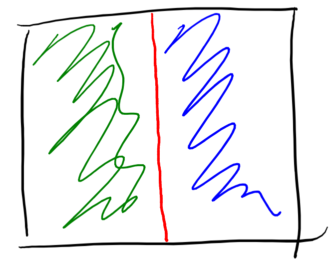
For NURBS surfaces, adding sharp features like creases required the use of trim curves.

For subdivision surfaces, we can just modify the subdivision masks.  E.g., we can mark some edges and vertices as “creases” and modify the subdivision mask for them (and their children):



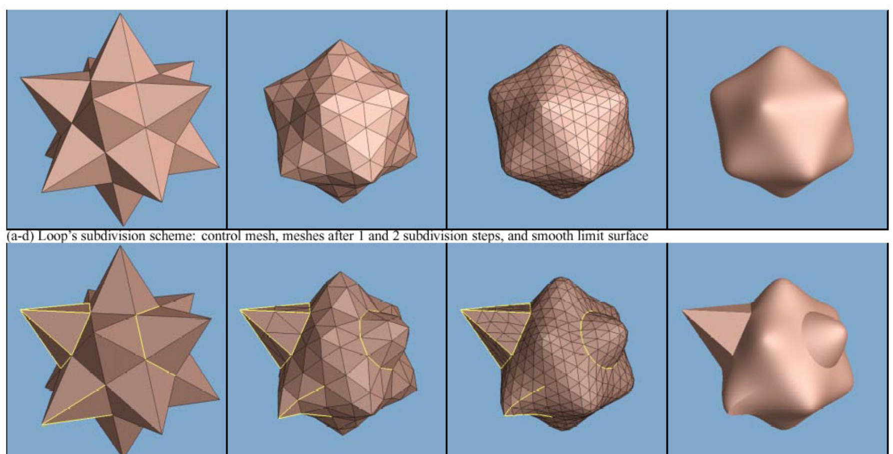

This gives rise to $G^0$ continuous surfaces (i.e., having positional but not tangent plane continuity).

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$



$$\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$$

[Hoppe, SIGGRAPH 1994]



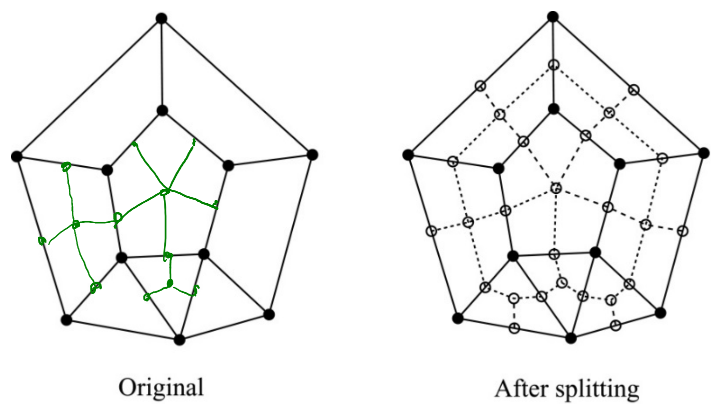
(a-d) Loop's subdivision scheme: control mesh, meshes after 1 and 2 subdivision steps, and smooth limit surface

(e-h) Our piecewise smooth subdivision scheme: tagged control mesh, meshes after 1 and 2 subdivision steps, and piecewise smooth limit surface

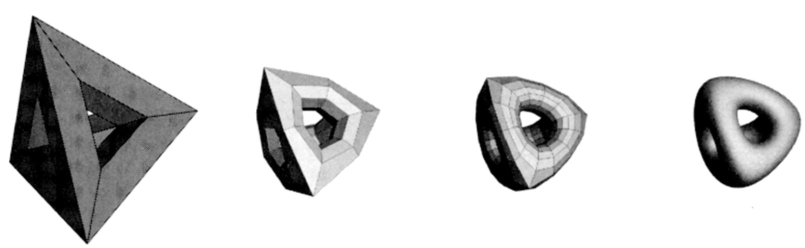
## Catmull-Clark subdivision

4:1 subdivision of triangles is sometimes called a **face scheme** for subdivision, as each face begets more faces.

An alternative face scheme starts with arbitrary polygon meshes and inserts vertices along edges and at face centroids:




**Catmull-Clark subdivision**:



Note: after the first subdivision, all polygons are quadilaterals in this scheme.

## Creases without trim curves, cont.

Here's an example using Catmull-Clark surfaces (based on subdividing quadrilateral meshes):

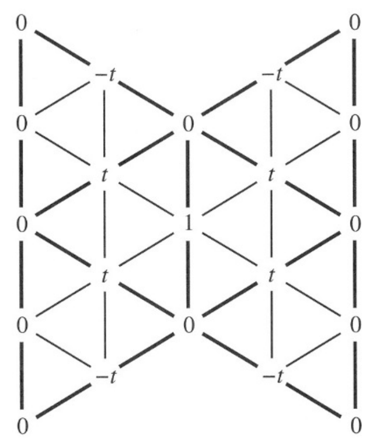
This particular example uses the hybrid technique of DeRose, et al., which applies sharp subdivision rules at some creases for a finite number of steps, and then switches to smooth subdivision, giving more gentle creases. This technique was used in Geri's Game.

## Interpolating subdivision surfaces

Interpolating schemes are defined by

- splitting
- averaging only new vertices

The following averaging mask is used in **butterfly subdivision**:




Setting $t$=0 gives the original polyhedron, and increasing small values of $t$ makes the surface smoother, until $t$=1/8 when the surface is provably $G^1$.