Subdivision curves and surfaces

Reading

Recommended:

- 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:

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

Subdivision curves

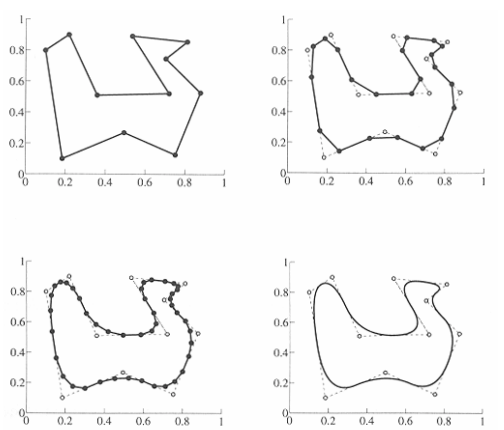
Idea:

- repeatedly refine the control polygon

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

- curve is the limit of an infinite process

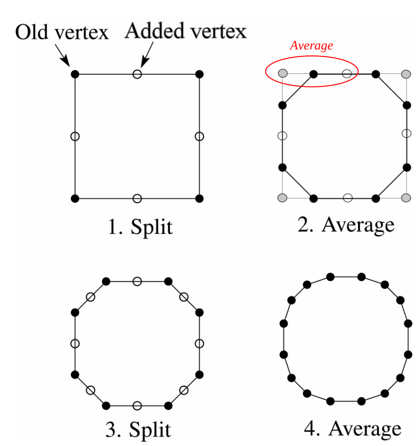
$$Q = \lim_{i \to \infty} P^i$$



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 next 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:

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

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

n=0:

n=1:

n=2:

Subdivide ad infinitum?

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

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

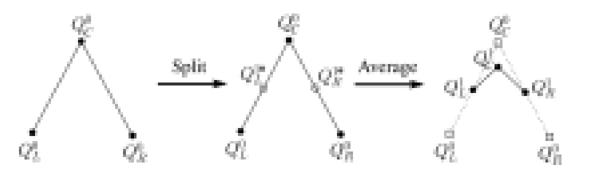
Can we push a vertex to its limit position without infinite subdivision? Yes!

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:

$$Q_L^{1*} = \frac{1}{2}\left(Q_L^0 + Q_C^0\right)$$

$$Q_R^{1*} = \frac{1}{2}\left(Q_C^0 + Q_R^0\right)$$

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

$$Q_C^1 = \frac{1}{4}\left(Q_L^{1*} + 2Q_C^0 + Q_R^{1*}\right) = \frac{1}{8}\left(Q_L^0 + 6Q_C^0 + Q_R^0\right)$$

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

Local subdivision matrix

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

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

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

Where the Q's are (for convenience) *row* vectors and S is the **local subdivision matrix**.

Expanding this relation we get

$$Q^j = SQ^{j-1} = SSQ^{j-2} = SSSQ^{j-3} = \cdots = S^jQ^0$$

and so the limit position for Q^0 is

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

Recipe for subdivision curves

Each subdivision scheme has its own **evaluation mask**, determined by eigenanalysis of the subdivision and averaging rules.

After subdividing and averaging a few times to get a fine enough mesh, we can push each vertex in the mesh to its limit position by applying the evaluation mask.

For Lane-Riesenfeld cubic B-spline subdivision, the evaluation mask 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.

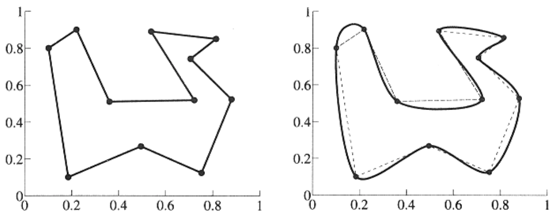
DLG interpolating scheme (1987)

Slight modification to subdivision algorithm:

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

For DLG (Dyn-Levin-Gregory), the averaging mask is:

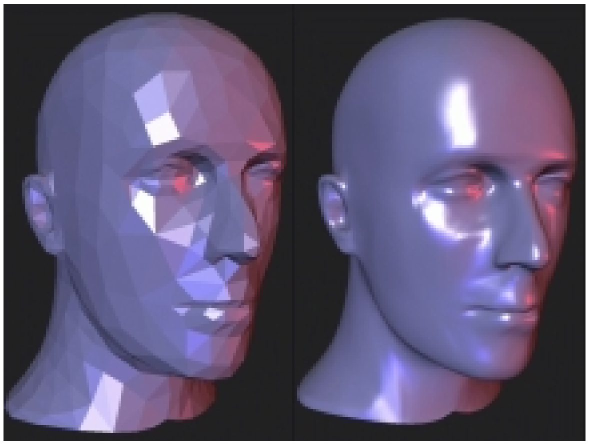
$$r = \frac{1}{16}(-2, 5, 10, 5, -2)$$



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…



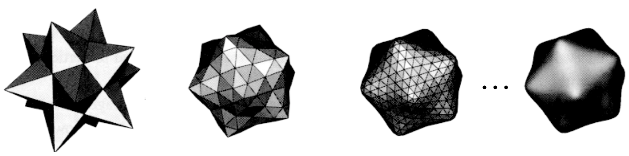
Subdivision surfaces

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

$$\sigma = \lim_{j \to \infty} M^j$$

using splitting and averaging steps.

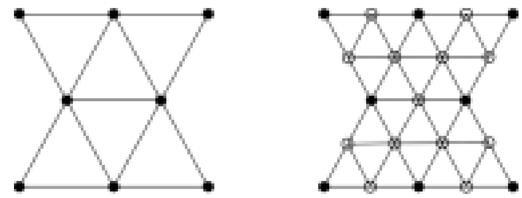


M^0 $\quad M^1 \quad M^2 \quad \cdots \quad M^\infty$

Triangular subdivision

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

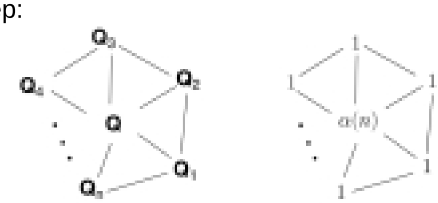
A common choice for triangle meshes is 4:1 subdivision – each triangular face is split into four subfaces:



Original After splitting

Loop's subdivision scheme

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



Vertex neighborhood Averaging mask (before affine normalization)

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

where

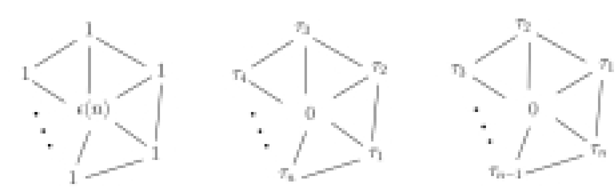
$$\alpha(n) = \frac{n(1-\beta(n))}{\beta(n)} \qquad \beta(n) = \frac{5}{4} - \frac{(3 + 2\cos(2\pi/n))^2}{32}$$

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's evaluation and tangent masks

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



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? Why would we want to?

Recipe for subdivision surfaces

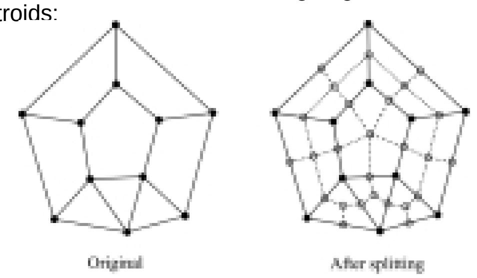
As with subdivision curves, we can now describe a recipe for creating and rendering subdivision surfaces:

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

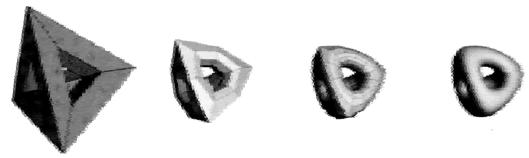
Catmull-Clark subdivision

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:

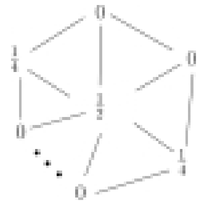


Note: after the first subdivision, all polygons are quadilaterals in this scheme.

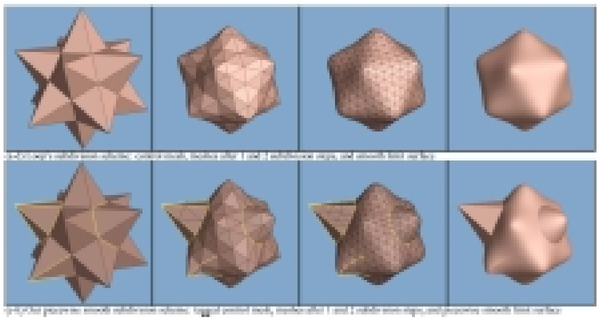
Adding creases

In some cases, we want a particular feature such as a crease to be preserved.

For subdivision surfaces, we can just modify the subdivision mask:



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



Creases

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

Summary

What to take home:

- The meanings of all the **boldfaced** terms.
- How to perform the splitting and averaging steps on subdivision curves.
- How to perform mesh splitting steps for subdivision surfaces, especially Loop.
- How to construct and render subdivision surfaces from their averaging masks, evaluation masks, and tangent masks.