Subdivision curves and surfaces

Brian Curless
CSE 457
Spring 2011

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:

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

This is already fixed in the handout.

Subdivision curves

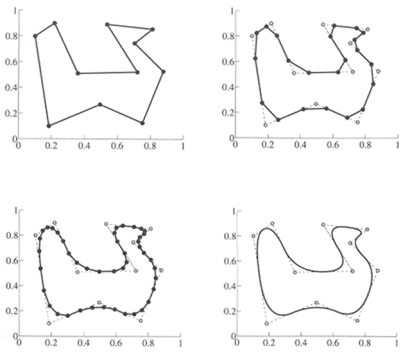
Idea:

- repeatedly refine the control polygon

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

- curve is the limit of an infinite process

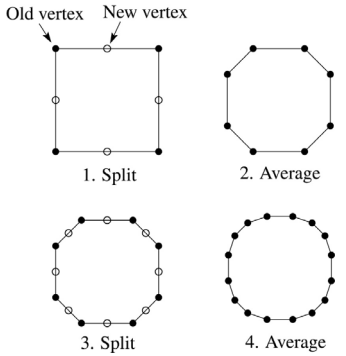
$$Q = \lim_{j \to \infty} P^j$$



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:

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

In the case of Chaikin's algorithm:

$$r =$$

Different averaging masks lead to different curves.

For example,

$$r = \left(\frac{1}{4} \quad \frac{1}{2} \quad \frac{1}{4} \right)$$

Leads to ***cubic*** B-spline curves.

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?

Recipe for subdivision curves

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

After subdividing and averaging a few times, we can push each vertex to its limit position by applying an **evaluation mask**.

Each subdivision scheme has its own evaluation mask, mathematically determined by analyzing the subdivision and averaging rules.

For cubic B-spline subdivision, we get:

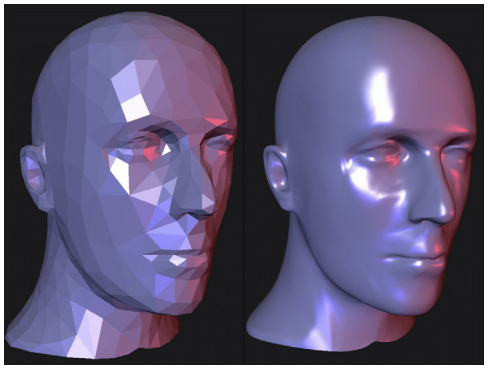
$$r = \left(\frac{1}{6} \quad \frac{2}{3} \quad \frac{1}{6} \right)$$

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.

Building complex models

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



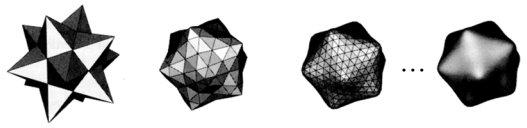
Subdivision surfaces

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

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

using splitting and averaging steps.

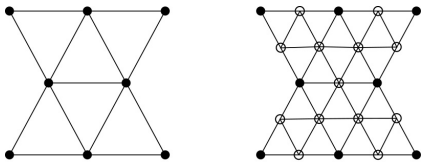


M^0 M^1 M^2 $\cdots$ M^∞

Triangular subdivision

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

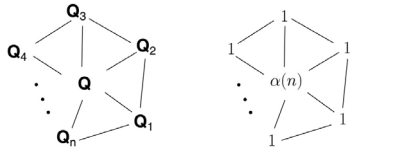
A common choice for triangle meshes is 4:1 subdivision – each triangular face is split into four smaller triangles:



Original After splitting

Loop averaging step

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



Vertex neighorhood Averaging mask (before affine normalization)

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

where

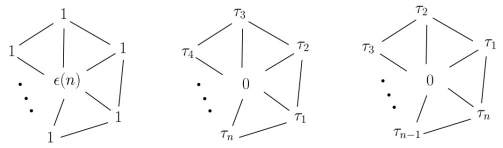
$$\alpha(n) = \frac{n(1-\beta(n))}{\beta(n)} \qquad \beta(n) = \frac{5}{4} - \frac{(3+2\cos(2\pi/n))^2}{32}$$

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 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?

Recipe for subdivision surfaces

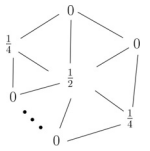
As with subdivision curves, we can now describe a recipe for creating and rendering subdivision surfaces:

- 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

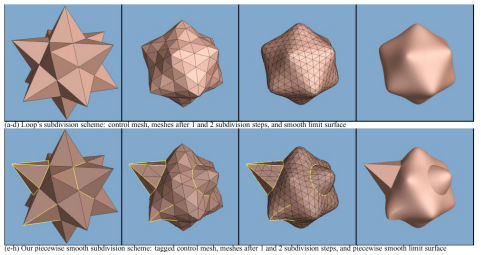
For NURBS surfaces, adding sharp features like creases required the use of trim curves.

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).

[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

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.