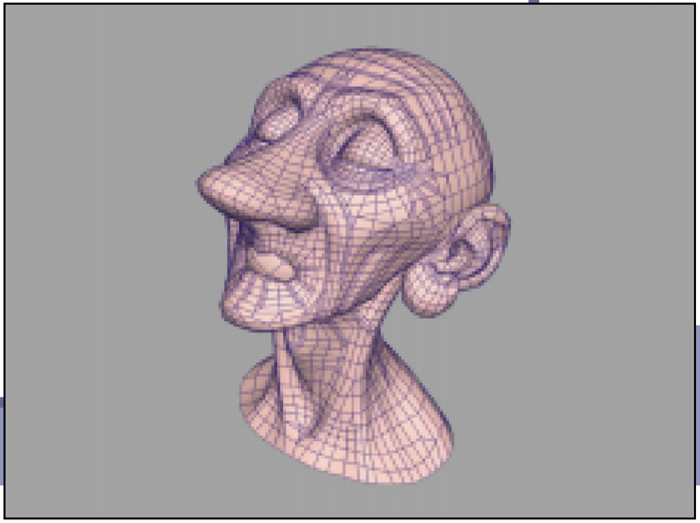
Advanced Graphics

Subdivision Surfaces



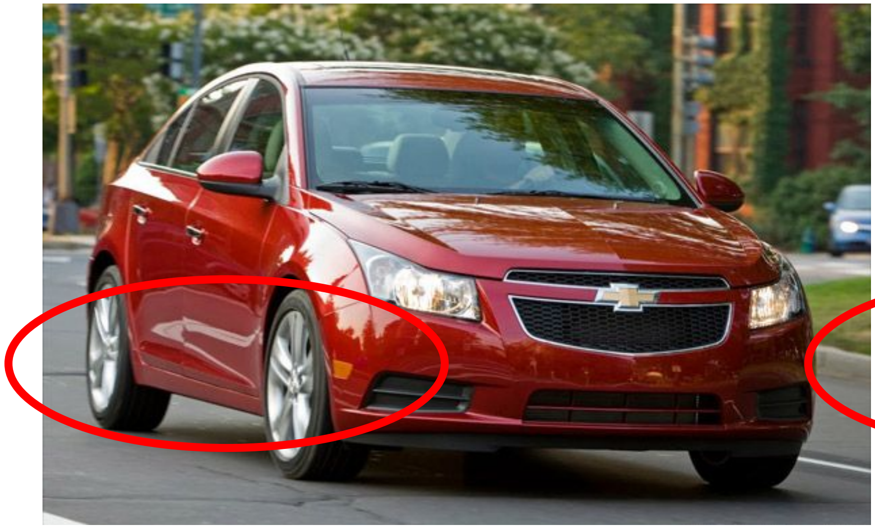
Alex Benton, University of Cambridge – A.Benton@damtp.cam.ac.uk

Supported in part by Google UK, Ltd

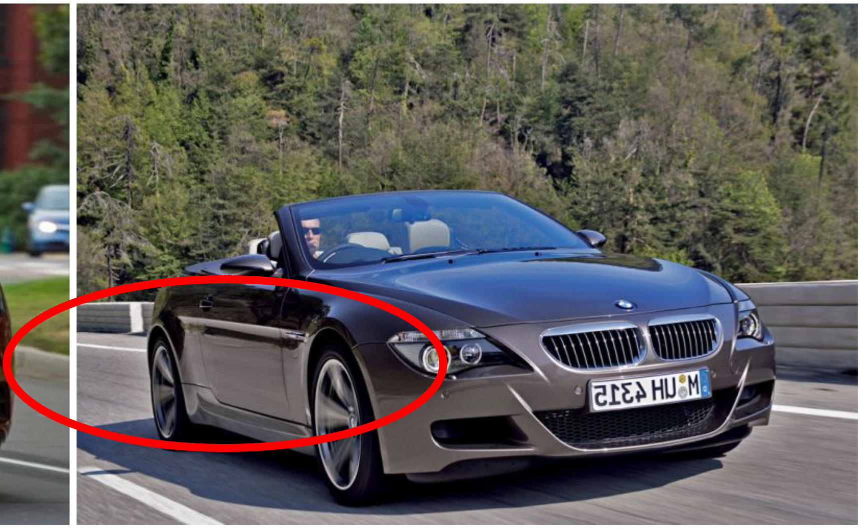
CAD, CAM, and a new motivation: *shiny things*

Expensive products are sleek and smooth.

→ Expensive products are C2 continuous.



Shiny, but reflections are warped



Shiny, and reflections are perfect

History

The term *spline* comes from the shipbuilding industry: long, thin strips of wood or metal would be bent and held in place by heavy 'ducks', lead weights which acted as control points of the curve.

Wooden splines can be described by C_n-continuous Hermite polynomials which interpolate n+1 control points.

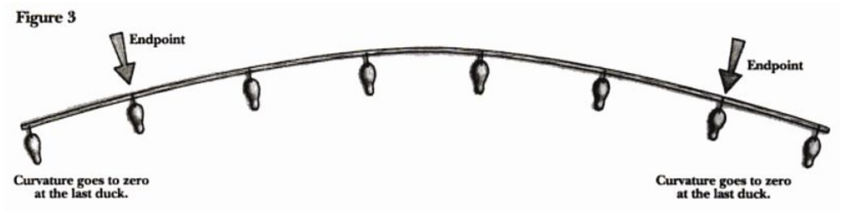
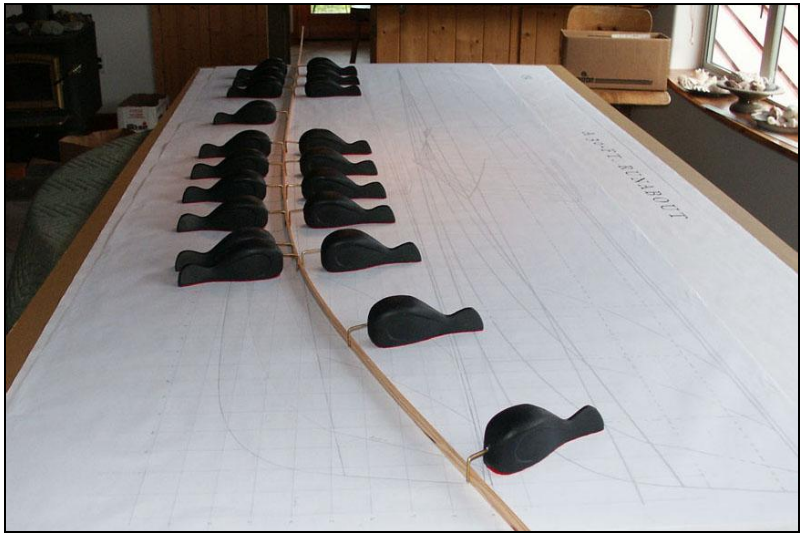


Figure 3
Endpoint
Endpoint
Curvature goes to zero at the last duck.
Curvature goes to zero at the last duck.



Top: Fig 3, P.7, Bray and Spectre, *Planking and Fastening*, Wooden Boat Pub (1996)

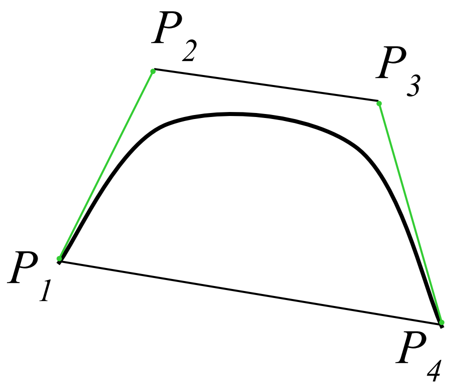
Bottom: http://www.pranos.com/boatsofwood/lofting%20ducks/lofting_ducks.htm

The drive for smooth CAD/CAM

- *Continuity* (smooth curves) can be essential to the perception of *quality*.
- The automotive industry wanted to design cars which were aerodynamic, but also visibly of high quality.
- Bezier (Renault) and de Casteljau (Citroen) invented Bezier curves in the 1960s. de Boor (GM) generalized them to B-splines.

Beziers—a quick review

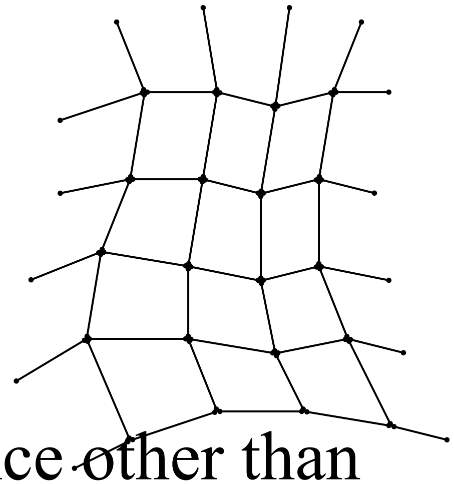
- A Bezier cubic is a function P(t) defined by four control points:
  - P_1 and P_4 are the endpoints of the curve
  - P_2 and P_3 define the other two corners of the bounding polygon.
- The curve fits entirely within the convex hull of $P_1 \ldots P_4$.
- Beziers are a subset of a broader class of splines and surfaces called *NURBS*: *Non Uniform Rational B-Splines*.
- For decades, NURBS patches have been the bedrock of CAD/CAM.



Cubic: $P(t) = (1-t)^3P_1 + 3t(1-t)^2P_2 + 3t^2(1-t)P_3 + t^3P_4$

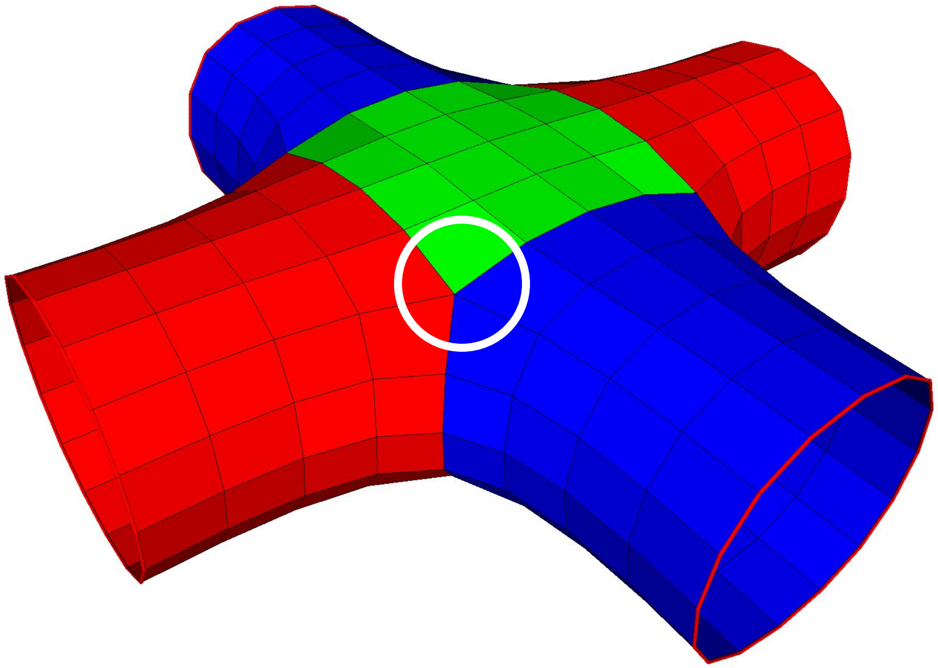
Bezier (NURBS) patches aren't the greatest

- NURBS patches are $n \times m$, forming a mesh of quadrilaterals.
  - What if you wanted triangles or pentagons?
    - A NURBS dodecahedron?
  - What if you wanted vertices of valence other than four?
- NURBS expressions for triangular patches, and more, do exist; but they're cumbersome.



Problems with NURBS patches

- Joining NURBS patches with C_n continuity across an edge is challenging.
- What happens to continuity at corners where the number of patches meeting isn't exactly four?
- Animation is tricky: bending and blending are doable, but not easy.



Sadly, the world isn't made up of shapes that can always be made from one smoothly-deformed rectangular surface.

Subdivision surfaces

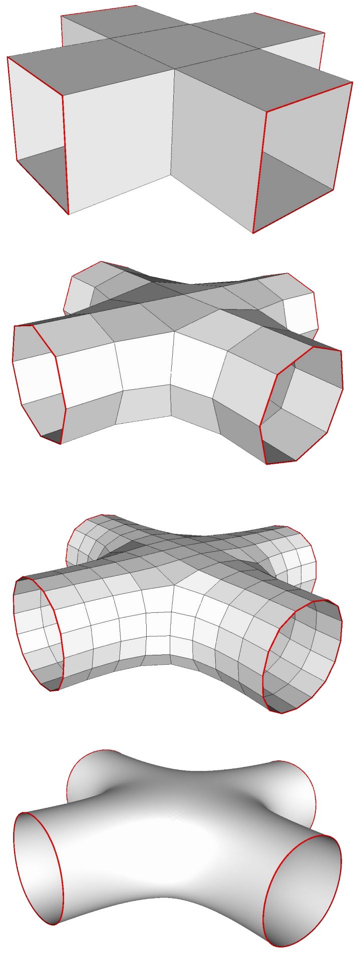
- Beyond shipbuilding: we want guaranteed continuity, without having to build everything out of rectangular patches.
    - Applications include CAD/CAM, 3D printing, museums and scanning, medicine, movies…

- The solution: *subdivision surfaces*.

Geri's Game, by Pixar (1997)

Subdivision surfaces

- Instead of ticking a parameter *t* along a parametric curve (or the parameters *u*,*v* over a parametric grid), subdivision surfaces repeatedly refine from a coarse set of *control points*.
- Each step of refinement adds new faces and vertices.
- The process converges to a smooth *limit surface*.



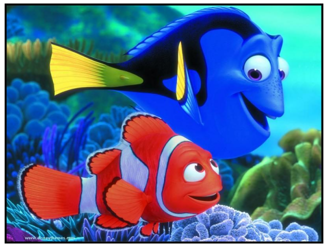
(Catmull-Clark in action)

Subdivision surfaces – History

- de Rahm described a 2D (curve) subdivision scheme in 1947; rediscovered in 1974 by Chaikin
- Concept extended to 3D (surface) schemes by two separate groups during 1978:
  - Doo and Sabin found a biquadratic surface
  - Catmull and Clark found a bicubic surface
- Subsequent work in the 1980s (Loop, 1987; Dyn [Butterfly subdivision], 1990) led to tools suitable for CAD/CAM and animation

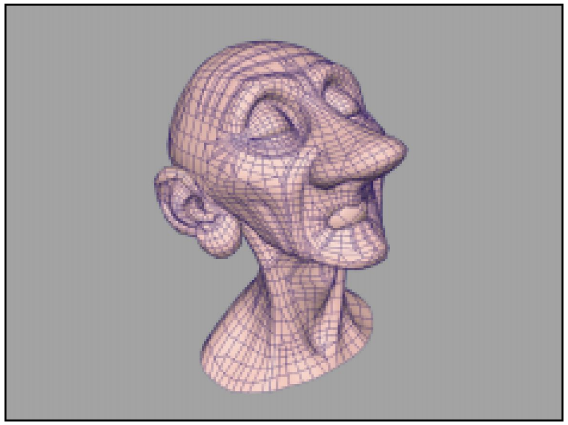
Subdivision surfaces and the movies

- Pixar first demonstrated subdivision surfaces in 1997 with Geri's Game.
  - Up until then they'd done everything in NURBS (Toy Story, A Bug's Life.)
  - From 1999 onwards everything they did was with subdivision surfaces (Toy Story 2, Monsters Inc, Finding Nemo...)
  - Two decades on, it's all heavily customized.
- It's not clear what Dreamworks uses, but they have recent patents on subdivision techniques.



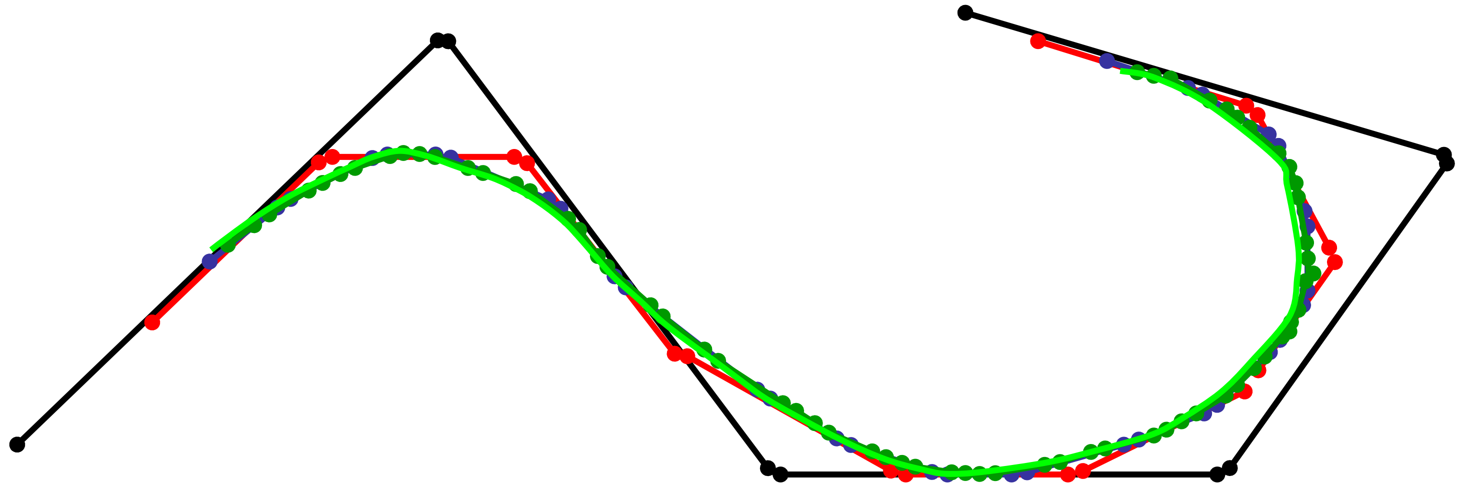
Useful terms

- A scheme which describes a 1D curve (even if that curve is travelling in 3D space, or higher) is called *univariate*, referring to the fact that the limit curve can be approximated by a polynomial in one variable (t).
- A scheme which describes a 2D surface is called *bivariate*, the limit surface can be approximated by a u,v parameterization.
- A scheme which retains and passes through its original control points is called an *interpolating* scheme.
- A scheme which moves away from its original control points, converging to a limit curve or surface nearby, is called an *approximating* scheme.


Control surface for Geri's head

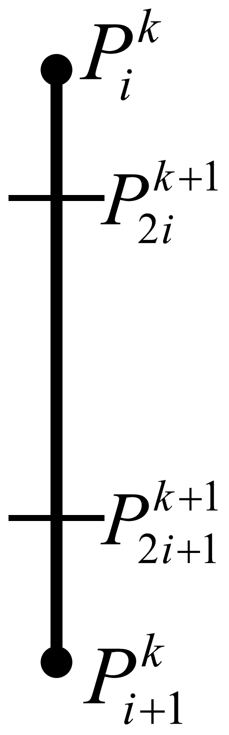
How it works

- Example: *Chaikin* curve subdivision (2D)
  - On each edge, insert new control points at ¼ and ¾ between old vertices; delete the old points
  - The *limit curve* is C1 everywhere (despite the poor figure.)



Notation

Chaikin can be written programmatically as:



$$P_{2i}^{k+1} = (3/4)P_i^k + (1/4)P_{i+1}^k \quad \leftarrow \textit{Even}$$

$$P_{2i+1}^{k+1} = (1/4)P_i^k + (3/4)P_{i+1}^k \quad \leftarrow \textit{Odd}$$

…where k is the ‘generation’; each generation will have twice as many control points as before.

Notice the different treatment of generating odd and even control points.

Borders (terminal points) are a special case.

Notation

Chaikin can be written in vector notation as:

$$\begin{bmatrix} \vdots \\ P_{2i-2}^{k+1} \\ P_{2i-1}^{k+1} \\ P_{2i}^{k+1} \\ P_{2i+1}^{k+1} \\ P_{2i+2}^{k+1} \\ P_{2i+3}^{k+1} \\ \vdots \end{bmatrix} = \frac{1}{4} \begin{bmatrix} & & & \vdots & & & & \\ & 0 & 3 & 1 & 0 & 0 & 0 & \\ & 0 & 1 & 3 & 0 & 0 & 0 & \\ \cdots & 0 & 0 & 3 & 1 & 0 & 0 & \cdots \\ & 0 & 0 & 1 & 3 & 0 & 0 & \\ & 0 & 0 & 0 & 3 & 1 & 0 & \\ & 0 & 0 & 0 & 1 & 3 & 0 & \\ & & & \vdots & & & & \end{bmatrix} \begin{bmatrix} \vdots \\ P_{i-2}^{k} \\ P_{i-1}^{k} \\ P_{i}^{k} \\ P_{i+1}^{k} \\ P_{i+2}^{k} \\ P_{i+3}^{k} \\ \vdots \end{bmatrix}$$

Notation

- The standard notation compresses the scheme to a *kernel*:
  - $h = (1/4)[\dots,0,0,\boxed{1},\boxed{3},\boxed{3},\boxed{1},0,0,\dots]$
- The kernel interlaces the odd and even rules.
- It also makes matrix analysis possible: eigenanalysis of the matrix form can be used to prove the continuity of the subdivision limit surface.
  - The details of analysis are fascinating, lengthy, and sadly beyond the scope of this course
- The limit curve of Chaikin is a quadratic B-spline!

Reading the kernel

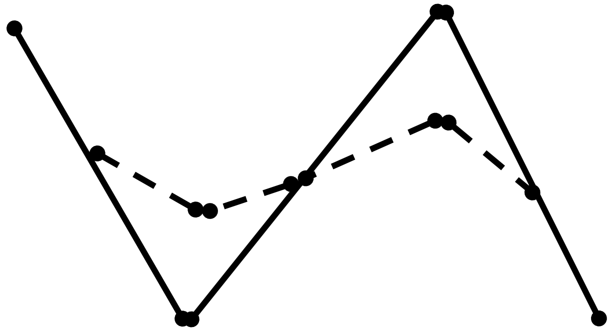
Consider the kernel

$$h=(1/8)[\ldots,0,0,1,4,6,4,1,0,0,\ldots]$$

You would read this as

$$P_{2i}^{k+1} = (\tfrac{1}{8})(P_{i-1}^{k} + 6P_{i}^{k} + P_{i+1}^{k})$$

$$P_{2i+1}^{k+1} = (\tfrac{1}{8})(4P_{i}^{k} + 4P_{i+1}^{k})$$



The limit curve is provably C2-continuous.

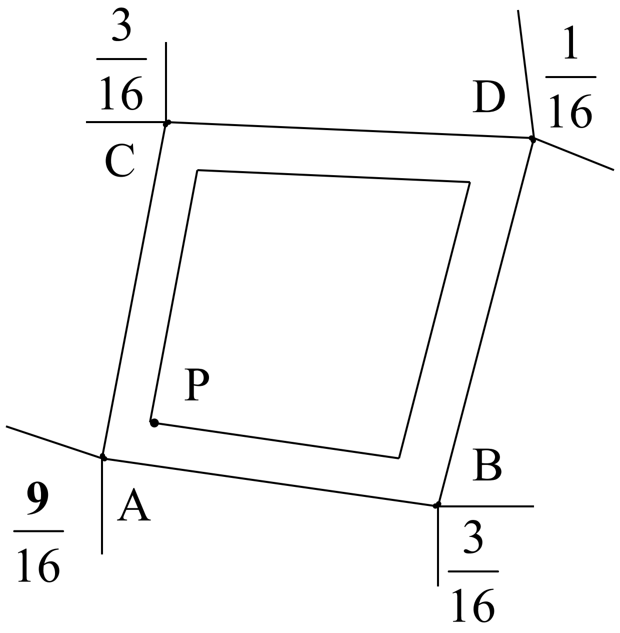
Making the jump to 3D: Doo-Sabin

Doo-Sabin takes Chaikin to 3D:

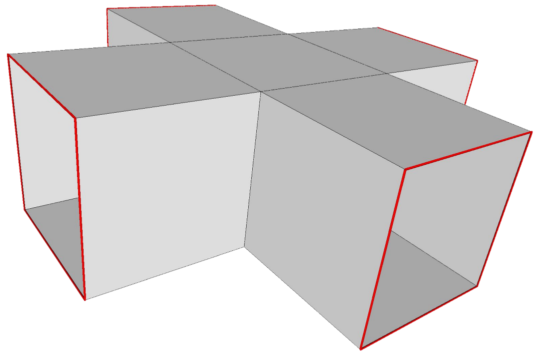
$$P = (9/16)\,A + (3/16)\,B + (3/16)\,C + (1/16)\,D$$

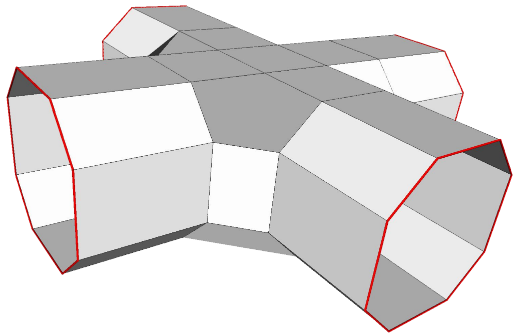
This replaces every old vertex with four new vertices.

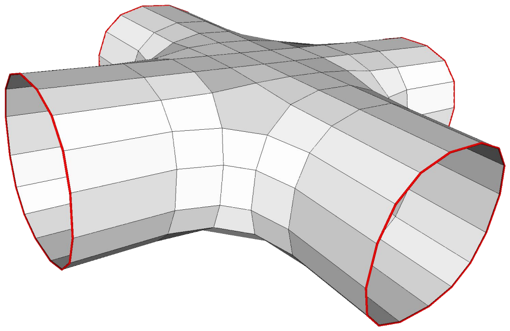
The limit surface is biquadratic, C1 continuous everywhere.

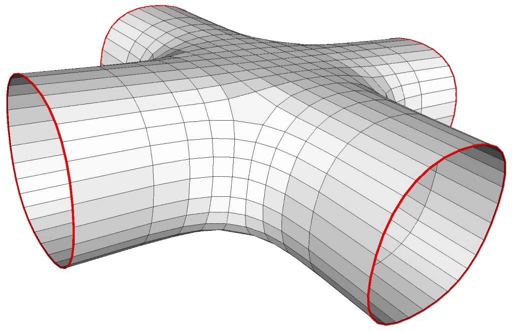


Doo-Sabin in action


(0) 18 faces

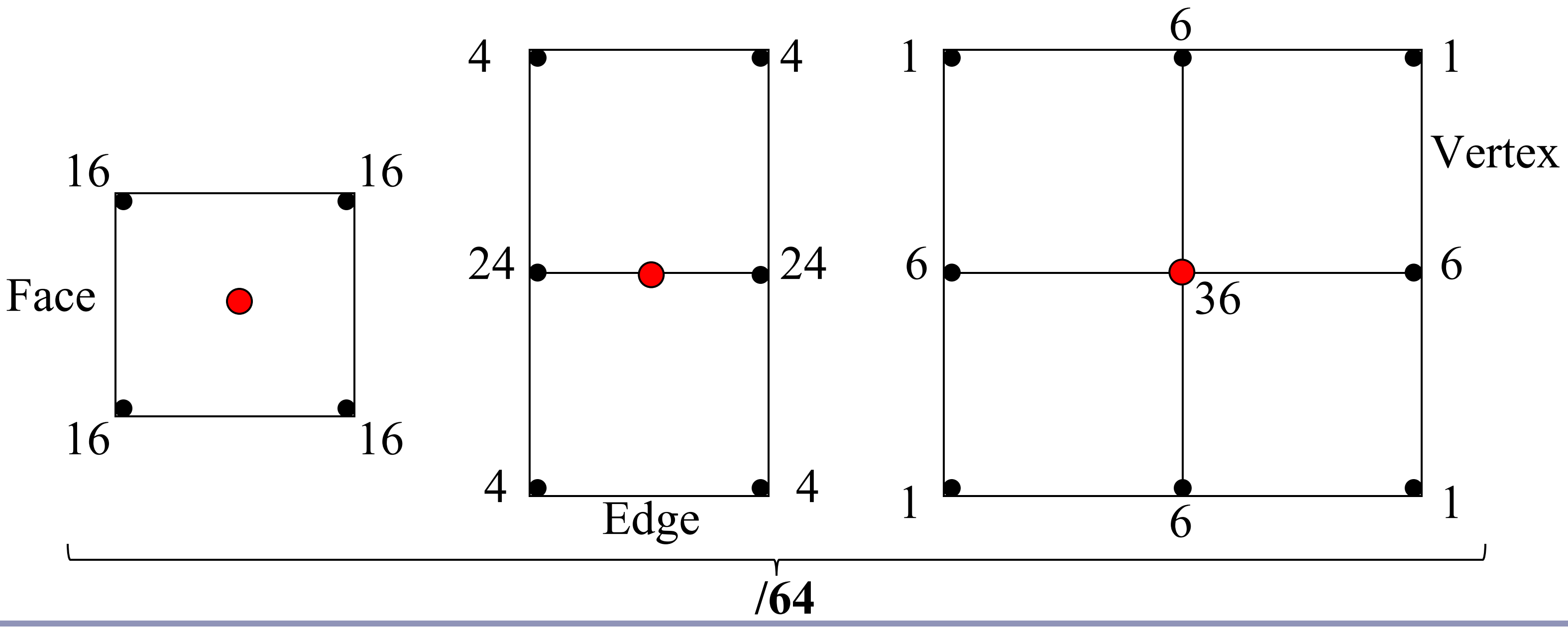

(1) 54 faces


(2) 190 faces

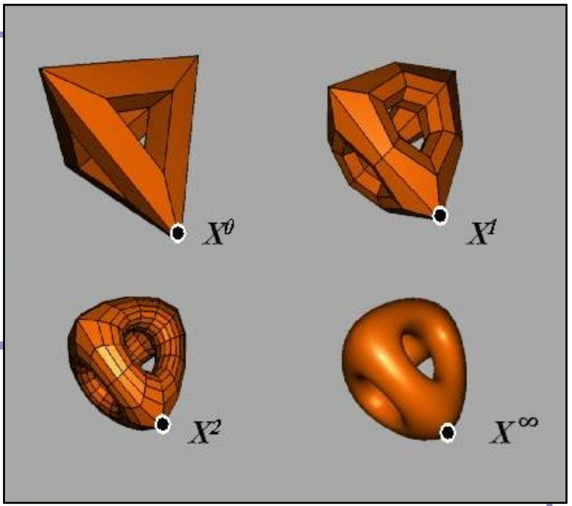

(3) 702 faces

Catmull-Clark

- *Catmull-Clark* is a bivariate approximating scheme with kernel h=(1/8)[1,4,6,4,1].
  - Limit surface is bicubic, C2-continuous.



Catmull-Clark

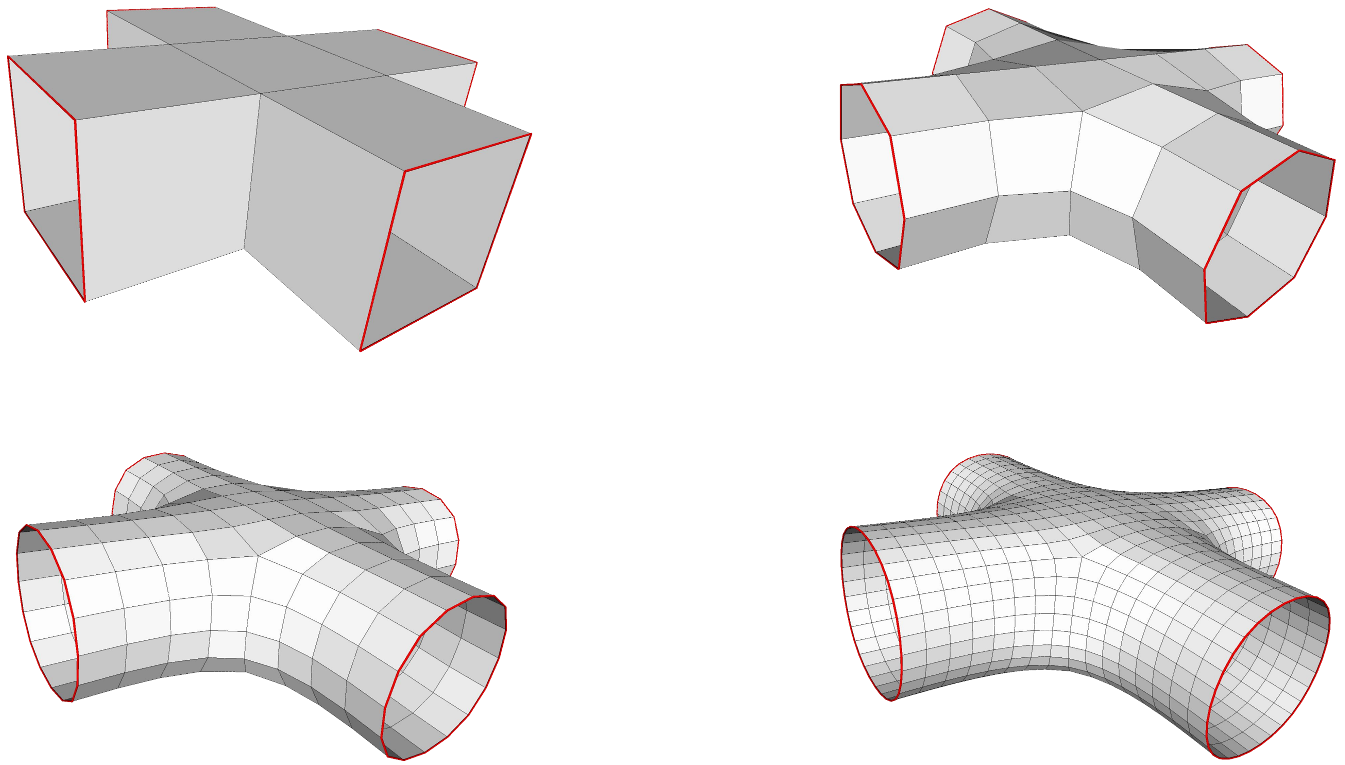


Getting tensor again:

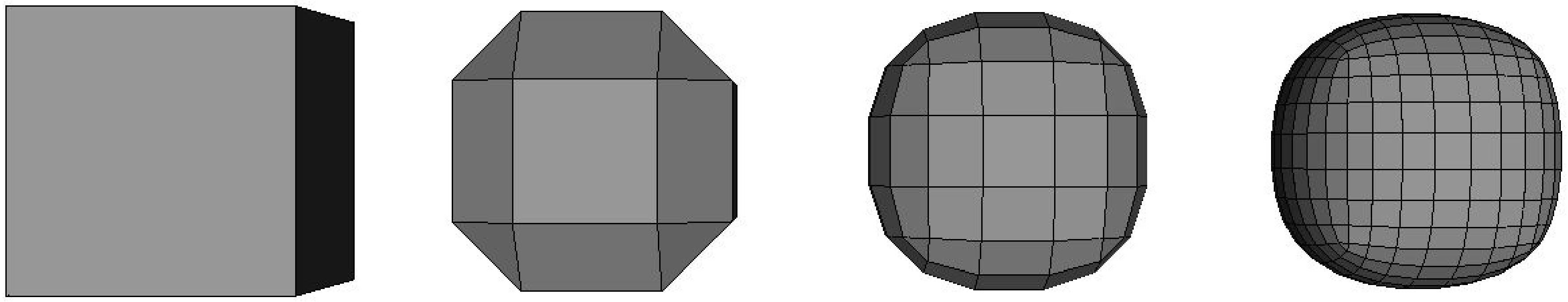
$$\frac{1}{8}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix}\otimes\frac{1}{8}\begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix}=\frac{1}{64}\begin{bmatrix}\boxed{1}&4&\boxed{6}&4&\boxed{1}\\ \boxed{4}&\boxed{16}&\boxed{24}&\boxed{16}&\boxed{4}\\ \boxed{6}&24&\boxed{36}&24&\boxed{6}\\ \boxed{4}&\boxed{16}&\boxed{24}&\boxed{16}&\boxed{4}\\ \boxed{1}&4&\boxed{6}&4&\boxed{1}\end{bmatrix}$$

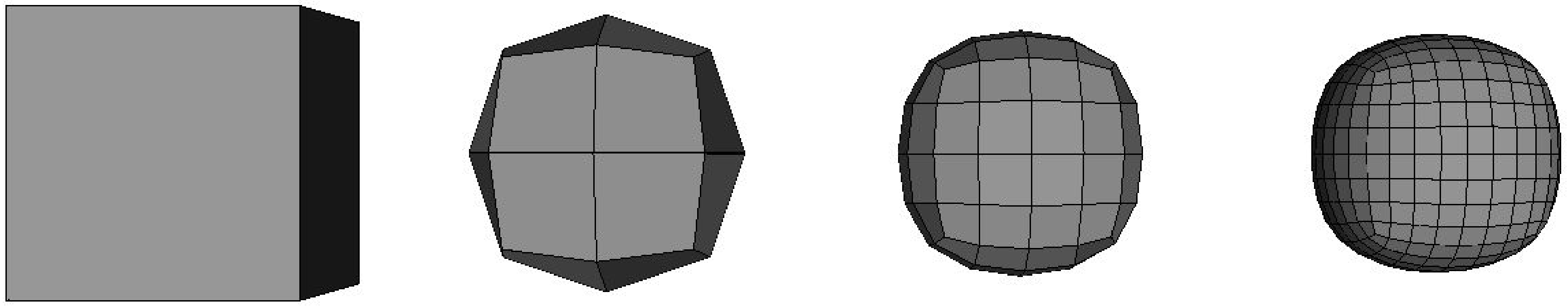
Vertex rule Face rule Edge rule

Catmull-Clark in action



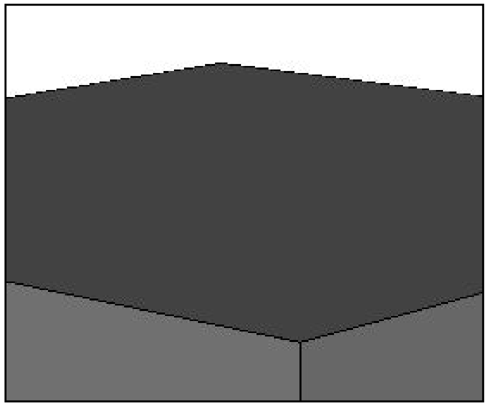
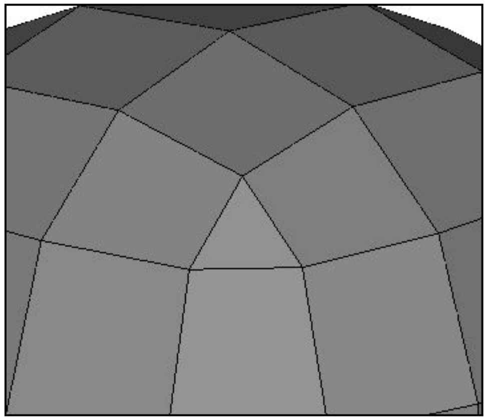
Catmull-Clark vs Doo-Sabin


Doo-Sabin


Catmull-Clark

Extraordinary vertices

- Catmull-Clark and Doo-Sabin both operate on quadrilateral meshes.
  - All faces have four boundary edges
  - All vertices have four incident edges
- What happens when the mesh contains *extraordinary* vertices or faces?
  - For many schemes, adaptive weights exist which can continue to guarantee at least some (non-zero) degree of continuity, but not always the best possible.
- CC replaces extraordinary faces with extraordinary vertices; DS replaces extraordinary vertices with extraordinary faces.



Detail of Doo-Sabin at cube corner

Extraordinary vertices: Catmull-Clark

Catmull-Clark vertex rules generalized for extraordinary vertices:

- Original vertex:
  $(4n\text{-}7) / 4n$
- Immediate neighbors in the one-ring:
  $3/2n^2$
- Interleaved neighbors in the one-ring:
  $1/4n^2$

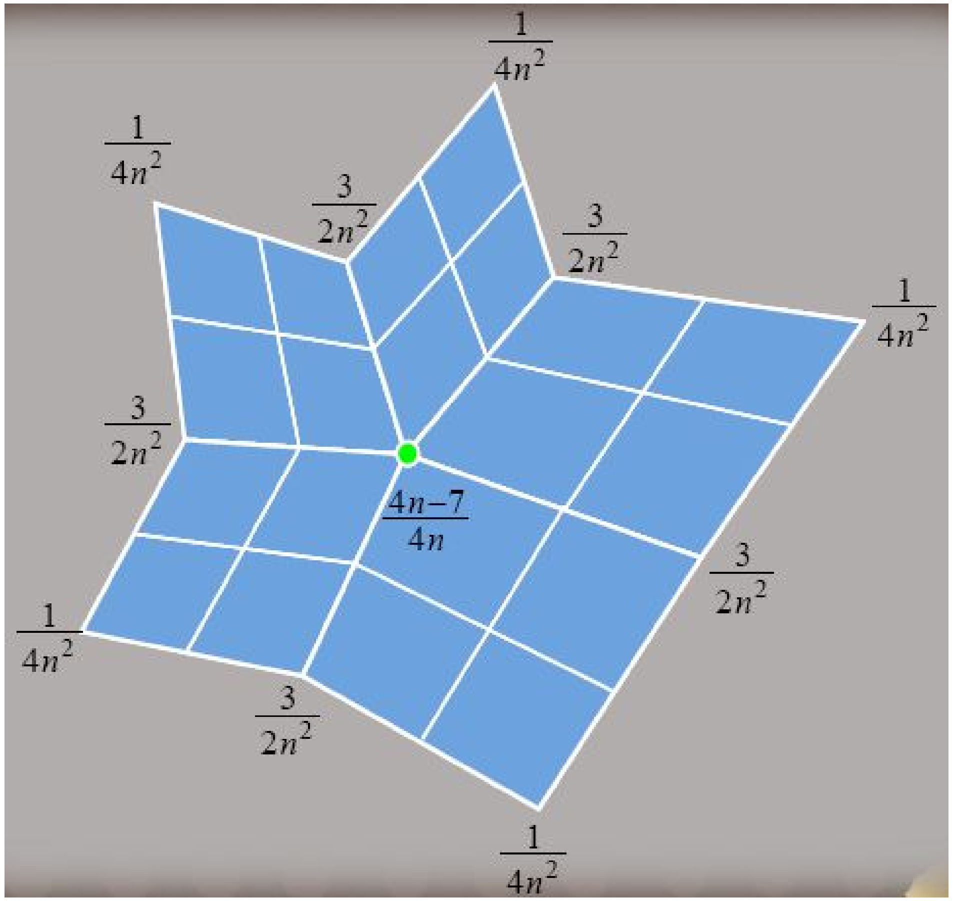
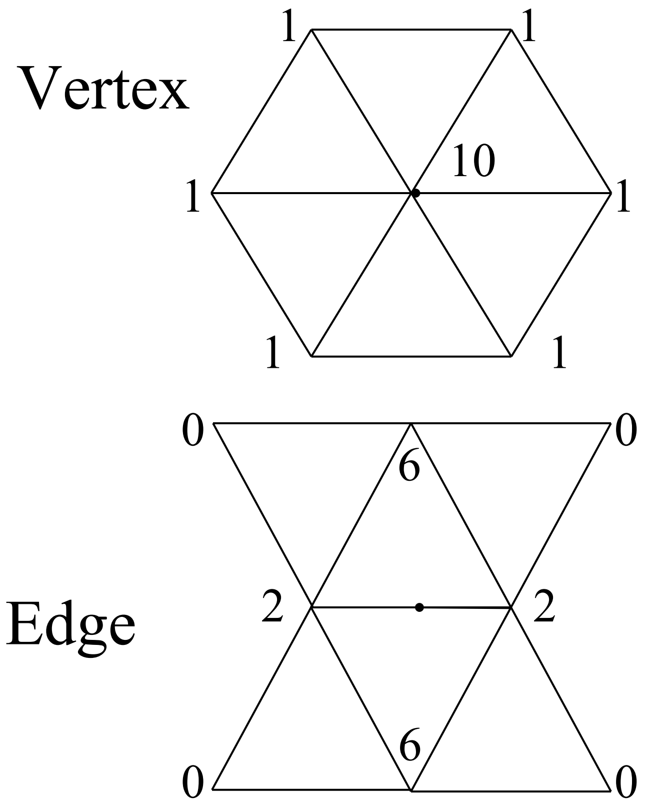
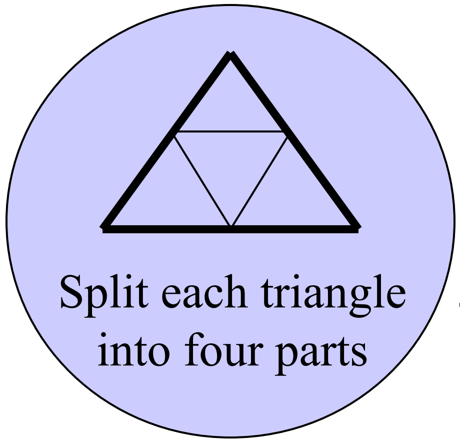


Image source: "*Next-Generation Rendering of Subdivision Surfaces*", Ignacio Castaño, SIGGRAPH 2008

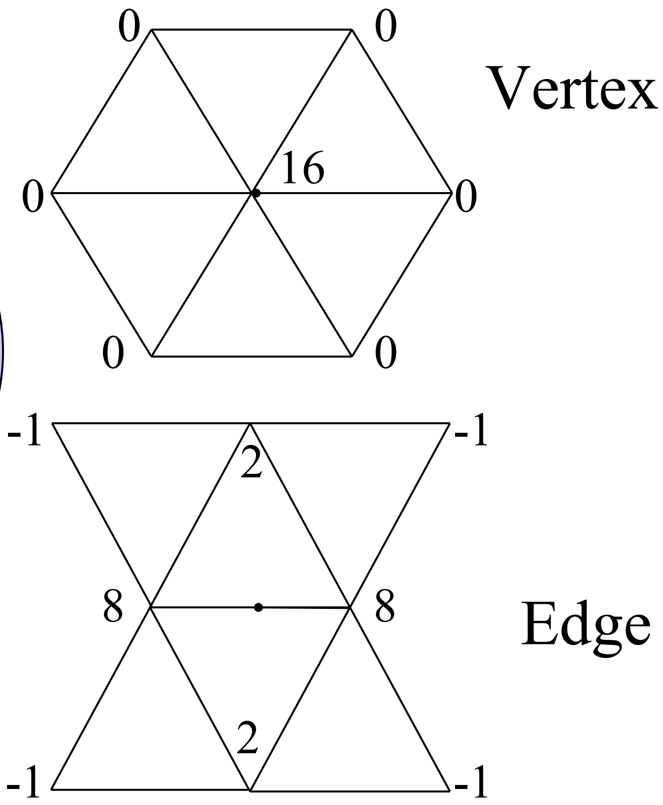
Schemes for simplicial (triangular) meshes

- *Loop* scheme:
- *Butterfly* scheme:

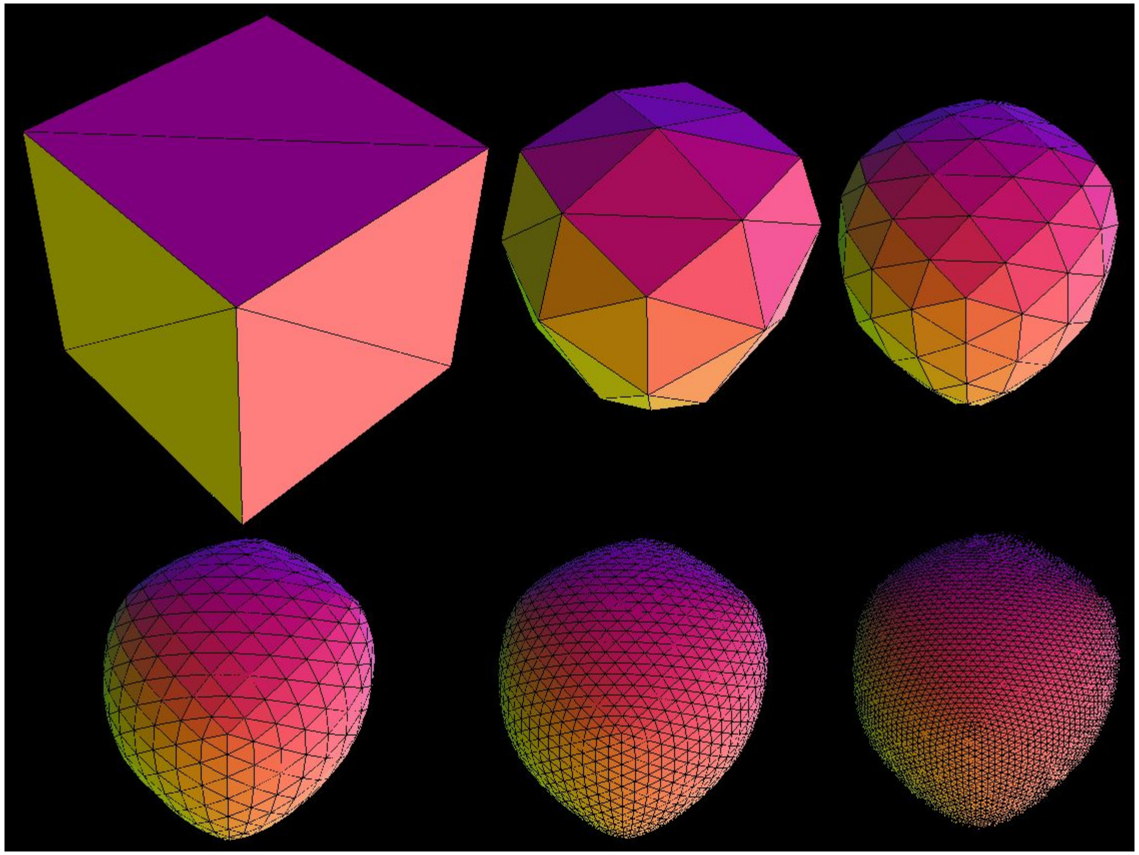




(All weights are /16)



Loop subdivision

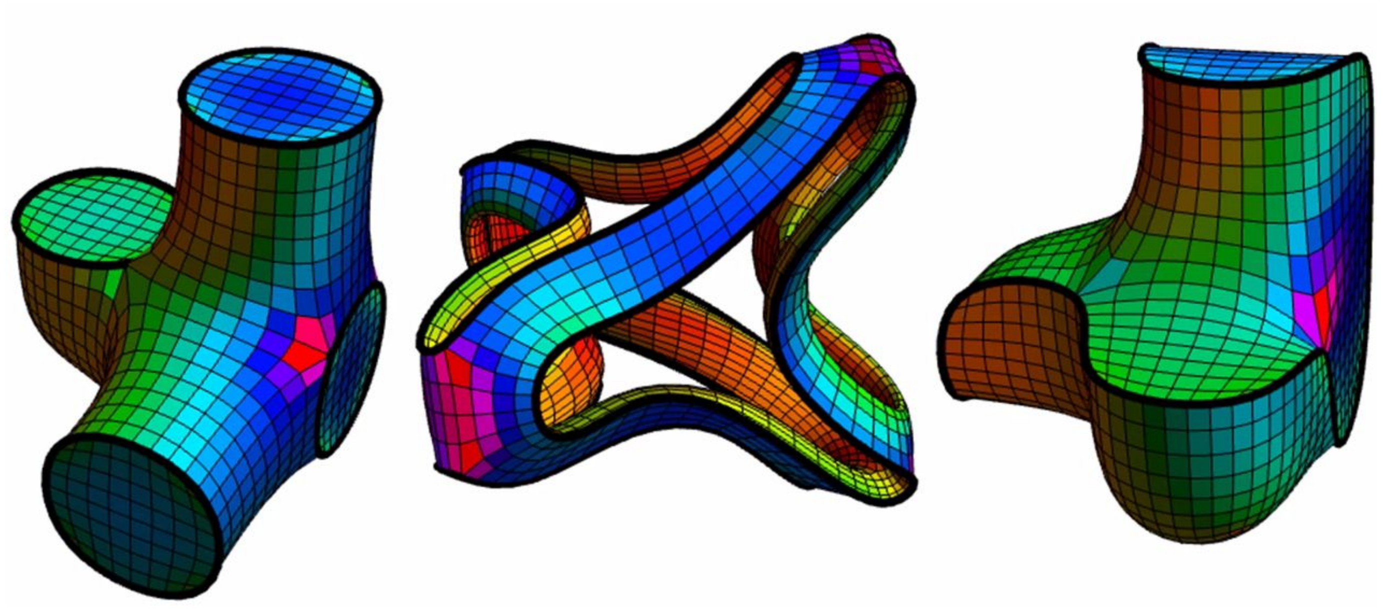


Loop subdivision in action. The asymmetry is due to the choice of face diagonals.

Image by Matt Fisher, http://www.its.caltech.edu/~matthewf/Chatter/Subdivision.html

Creases

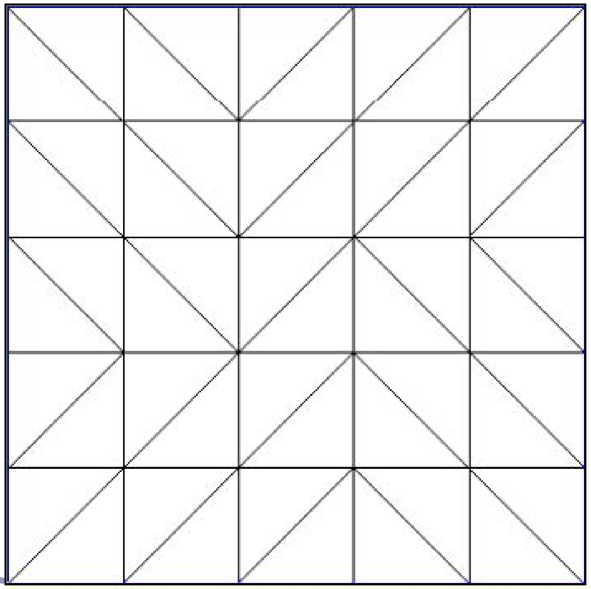
Extensions exist for most schemes to support *creases*, vertices and edges flagged for partial or hybrid subdivision.

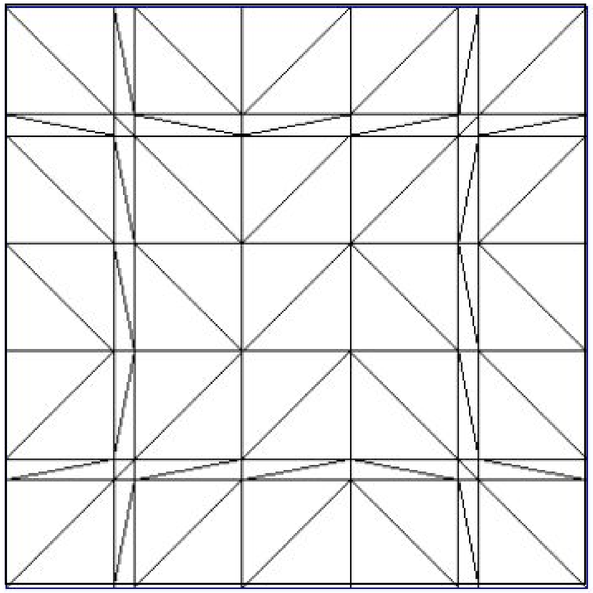


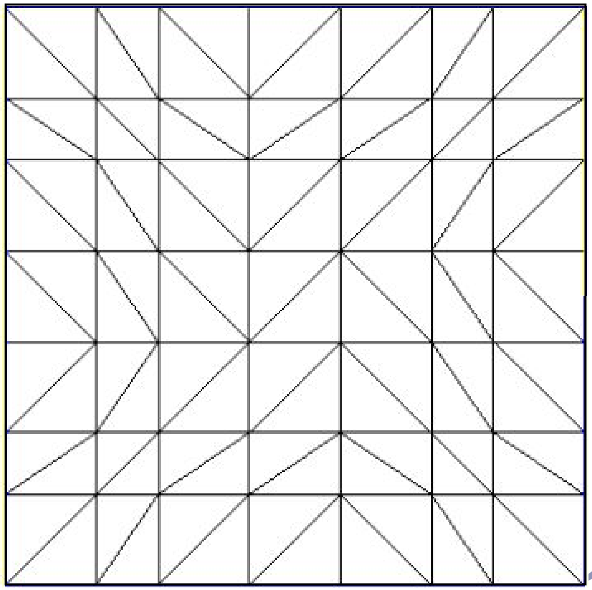
Still from "Volume Enclosed by Subdivision Surfaces with Sharp Creases" by Jan Hakenberg, Ulrich Reif, Scott Schaefer, Joe Warren http://vixra.org/pdf/1406.0060v1.pdf

Continuous level of detail

For live applications (e.g. games) can compute *continuous* level of detail, e.g. as a function of distance:


Level 5


Level 5.2

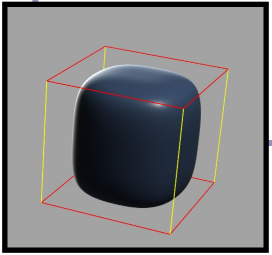

Level 5.8

Direct evaluation of the limit surface

- In the 1999 paper *Exact Evaluation Of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values*, Jos Stam (now at Alias|Wavefront) describes a method for finding the exact final positions of the CC limit surface.
  - His method is based on calculating the tangent and normal vectors to the limit surface and then shifting the control points out to their final positions.
  - What's particularly clever is that he gives exact evaluation at the extraordinary vertices. (Non-trivial.)

Bounding boxes and convex hulls for subdivision surfaces

- The limit surface is (the weighted average of (the weighted averages of (the weighted averages of (repeat for eternity…)))) the original control points.
- This implies that for any scheme where all weights are positive and sum to one, the limit surface lies entirely within the convex hull of the original control points.
- For schemes with negative weights:
  - Let $L=max_t \Sigma_i |N_i(t)|$ be the greatest sum throughout parameter space of the absolute values of the weights.
  - For a scheme with negative weights, L will exceed 1.
  - Then the limit surface must lie within the convex hull of the original control points, expanded unilaterally by a ratio of (L-1).



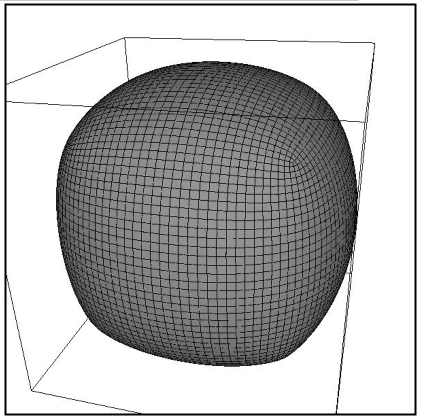
Splitting a subdivision surface

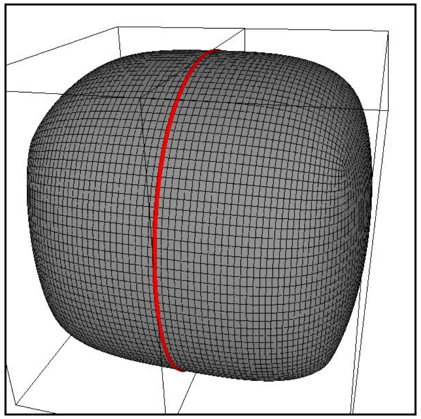
Many algorithms rely on subdividing a surface and examining the bounding boxes of smaller facets.

- Rendering, ray/surface intersections…

It's not enough just to delete half your control points: the limit surface will change (see right)

- Need to include all control points from the previous generation, which influence the limit surface in this smaller part.

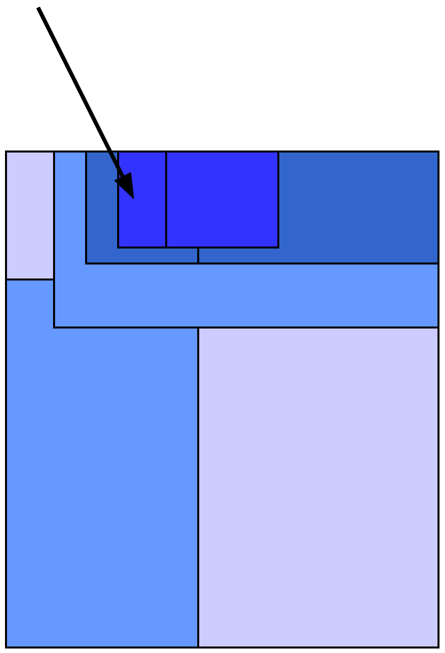




(Top) 5x Catmull-Clark subdivision of a cube
(Bottom) 5x Catmull-Clark subdivision of two halves of a cube; the limit surfaces are clearly different.

Ray/surface intersection

- To intersect a ray with a subdivision surface, we recursively split and split again, discarding all portions of the surface whose bounding boxes / convex hulls do not lie on the line of the ray.
- Any subsection of the surface which is ‘close enough’ to flat is treated as planar and the ray/plane intersection test is used.
- This is essentially a binary tree search for the nearest point of intersection.
  - You can optimize by sorting your list of subsurfaces in increasing order of distance from the origin of the ray.

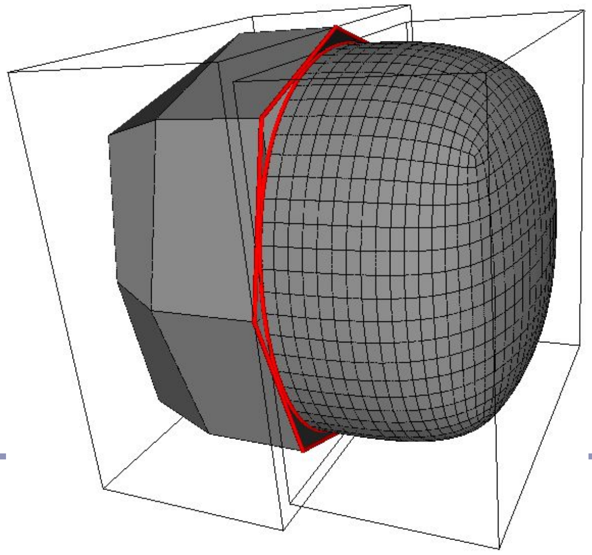


Rendering subdivision surfaces

- The algorithm to render any subdivision surface is exactly the same as for Bezier curves:

  “If the surface is simple enough, render it directly; otherwise split it and recurse.”
- One fast test for “simple enough” is,

  “Is the convex hull of the limit surface sufficiently close to flat?”
- Caveat: splitting a surface and subdividing one half but not the other can lead to tears where the different resolutions meet. →



Rendering subdivision surfaces on the GPU

- Subdivision algorithms have been ported to the GPU using geometry (tesselation) shaders.
  - This subdivision can be done completely independently of geometry, imposing no demands on the CPU.
  - Uses a complex blend of precalculated weights and shader logic
  - Impressive effects in use at id, Valve, et al

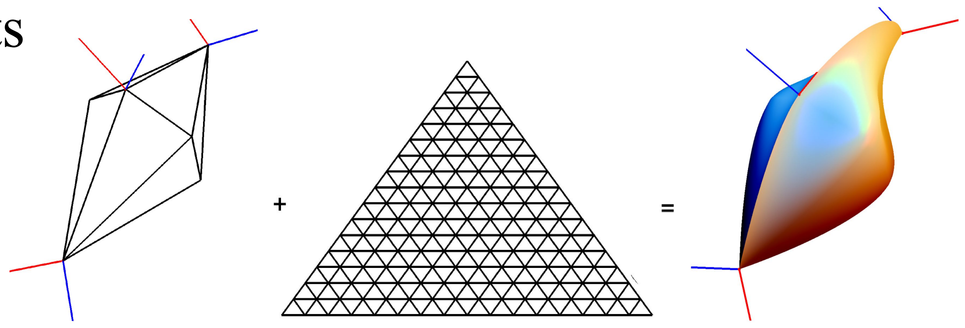


Figure from *Generic Mesh Renement on GPU,*
Tamy Boubekeur & Christophe Schlick (2005)
LaBRI INRIA CNRS University of Bordeaux, France

Subdivision Schemes—A partial list

- Approximating
  - Quadrilateral
    - (1/2)[1,2,1]
    - (1/4)[1,3,3,1] (Doo-Sabin)
    - (1/8)[1,4,6,4,1] (Catmull-Clark)
    - *Mid-Edge*
  - Triangles
    - Loop

- Interpolating
  - Quadrilateral
    - *Kobbelt*
  - Triangle
    - Butterfly
    - *"√3" Subdivision*

Many more exist, some much more complex

This is a major topic of ongoing research

References

Catmull, E., and J. Clark. "Recursively Generated B-Spline Surfaces on Arbitrary Topological Meshes." *Computer Aided Design*, 1978.

Dyn, N., J. A. Gregory, and D. A. Levin. "Butterfly Subdivision Scheme for Surface Interpolation with Tension Control." *ACM Transactions on Graphics*. Vol. 9, No. 2 (April 1990): pp. 160–169.

Halstead, M., M. Kass, and T. DeRose. "Efficient, Fair Interpolation Using Catmull-Clark Surfaces." *Siggraph '93*. p. 35.

Zorin, D. "Stationary Subdivision and Multiresolution Surface Representations." Ph.D. diss., California Institute of Technology, 1997

Ignacio Castano, "Next-Generation Rendering of Subdivision Surfaces." Siggraph '08, http://developer.nvidia.com/object/siggraph-2008-Subdiv.html

Dennis Zorin's SIGGRAPH course, "Subdivision for Modeling and Animation", http://www.mrl.nyu.edu/publications/subdiv-course2000/