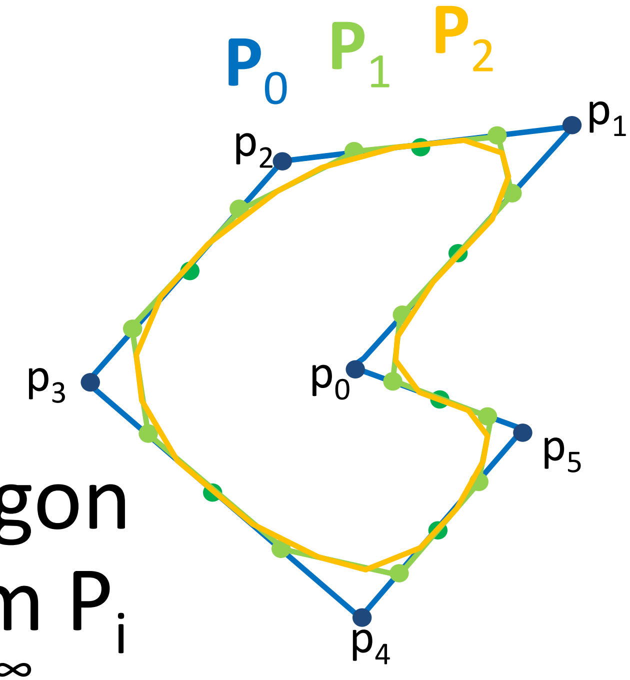
Subdivision Curves and Surfaces

COMP 557
Paul Kry

Subdivision Curves



$\mathbf{P} = \{p_i,\ i=0..n\text{-}1\}$

- Repeatedly refine a control polygon
  $\mathbf{P} = \mathbf{P}_0 \rightarrow \mathbf{P}_1 \rightarrow \mathbf{P}_2 \rightarrow \mathbf{P}_3$ $\qquad \mathbf{C} = \lim_{i\rightarrow\infty} P_i$
- In the limit we get a smooth curve

Chaikin's algorithm (1974)… corner cutting

- Given piecewise linear curve
  - Insert new vertices at midpoints
  - Average each with *next* neighbour
  - Repeat!

(the limit produces a piecewise quadratic polynomial curve)

Subdivision Curves

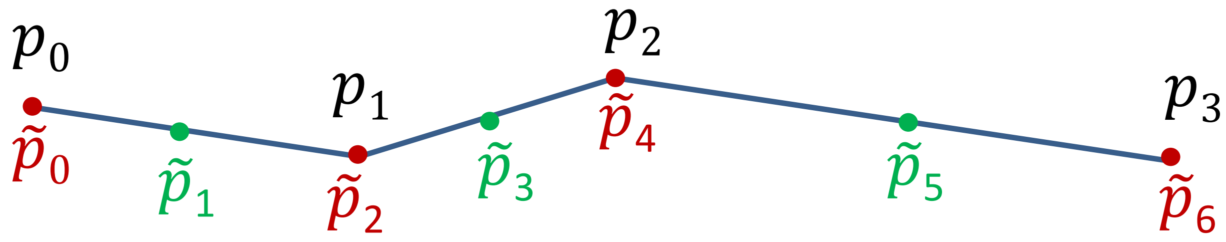
- Instead of neighbours, we can apply some other ***averaging mask*** (..., r_{-1}, r_0, r_1, ...)
  - Chaikin uses **r** = (r_0, r_1) = (0.5, 0.5)
  - Lane-Risenfeld algorithm (1980) uses masks from Pascal's triangle

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

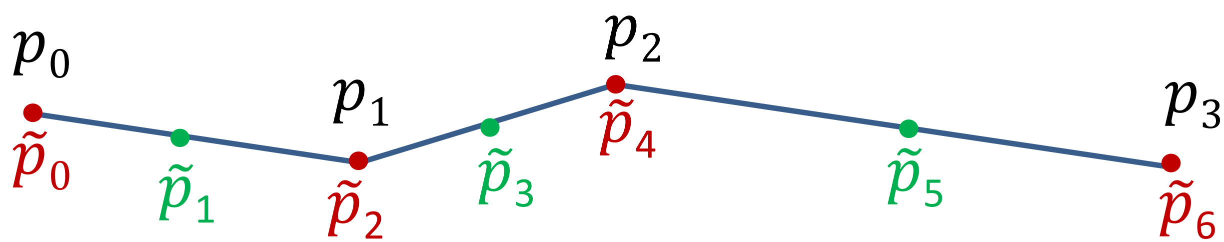
    - This mask produces piecewise degree n+1 polynomial curves in the limit.

Subdivision Curves

- How many steps until converged?
  - Flat? No subdivision needed
  - Curved? Need to subdivide, at least until sufficiently flat *locally*
    - Can test for this by looking at neighbours
- Even and odd vertices
  - We call new vertices we insert ***odd*** vertices and the old vertices are ***even*** vertices.



Combined Splitting and Averaging Steps



Split/ Refine

$$\tilde{p}_{2i+1}^{j+1} = \frac{1}{2}p_i^j + \frac{1}{2}p_{i+1}^j,$$
$$\tilde{p}_{2i}^{j+1} = p_i^j.$$

Subscript denotes index

Superscript denotes subdivision level

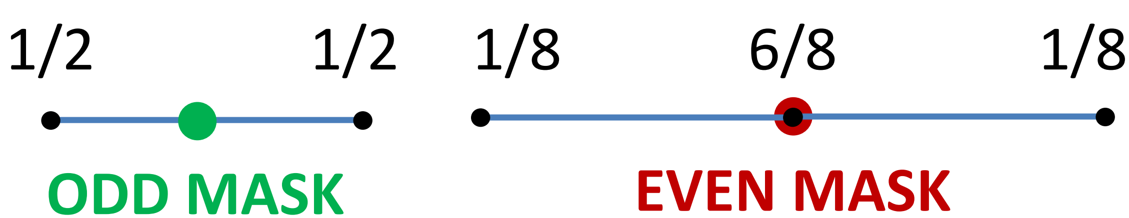
Lets look at Lane-Risenfeld scheme with n = 2

Apply mask

$$p_{2i+1}^{j+1} = \frac{1}{4}\tilde{p}_{2i}^{j+1} + \frac{1}{2}\tilde{p}_{2i+1}^{j+1} + \frac{1}{4}\tilde{p}_{2i+2}^{j+1},$$
$$p_{2i}^{j+1} = \frac{1}{4}\tilde{p}_{2i-1}^{j+1} + \frac{1}{2}\tilde{p}_{2i}^{j+1} + \frac{1}{4}\tilde{p}_{2i+1}^{j+1}.$$

$\Rightarrow$

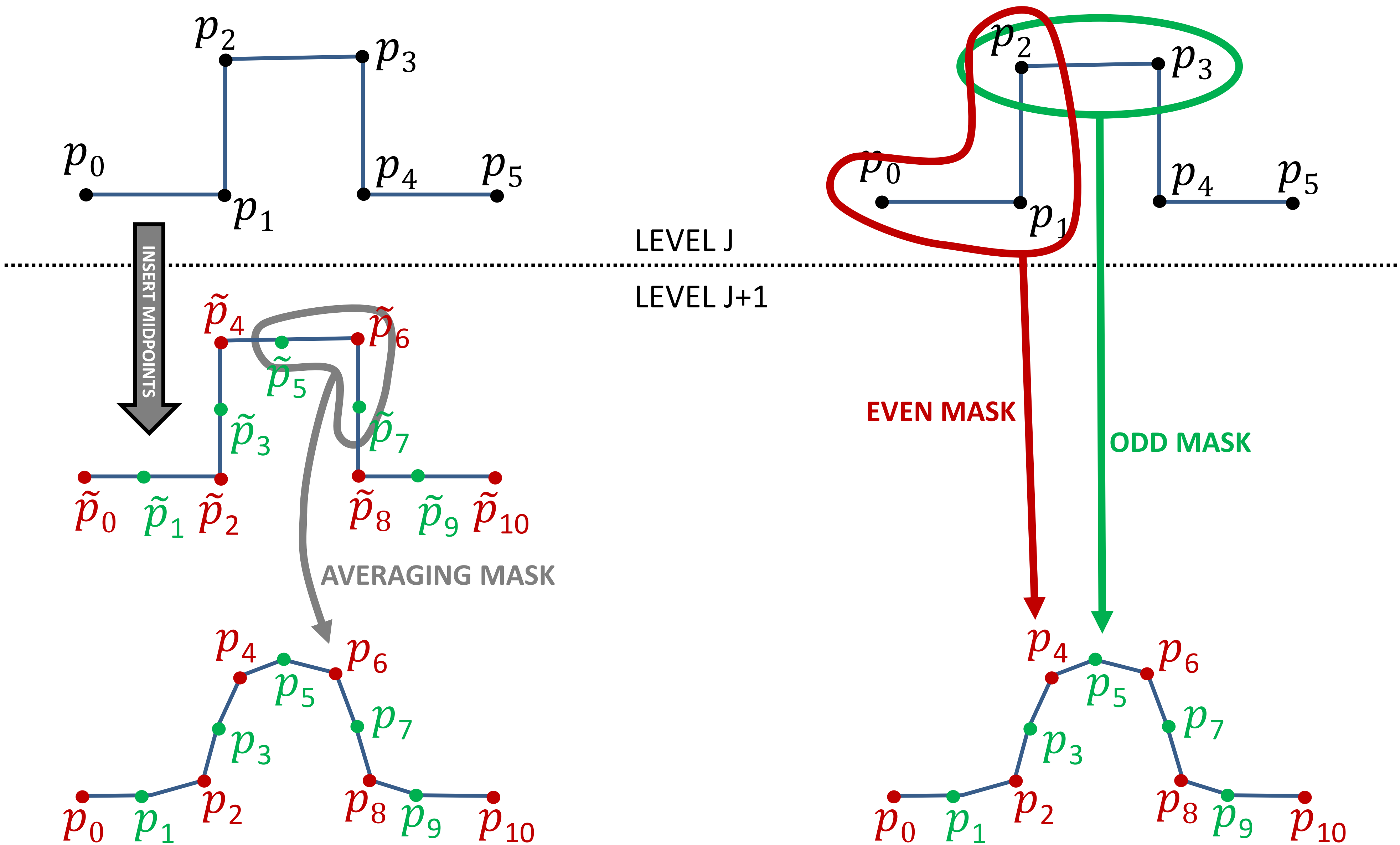
$$p_{2i+1}^{j+1} = \frac{1}{2}p_i^j + \frac{1}{2}p_{i+1}^j,$$
$$p_{2i}^{j+1} = \frac{1}{8}p_{i-1}^j + \frac{6}{8}p_i^j + \frac{1}{8}p_{i+1}^j.$$



Sub one into the other, and collect terms to identify rules!

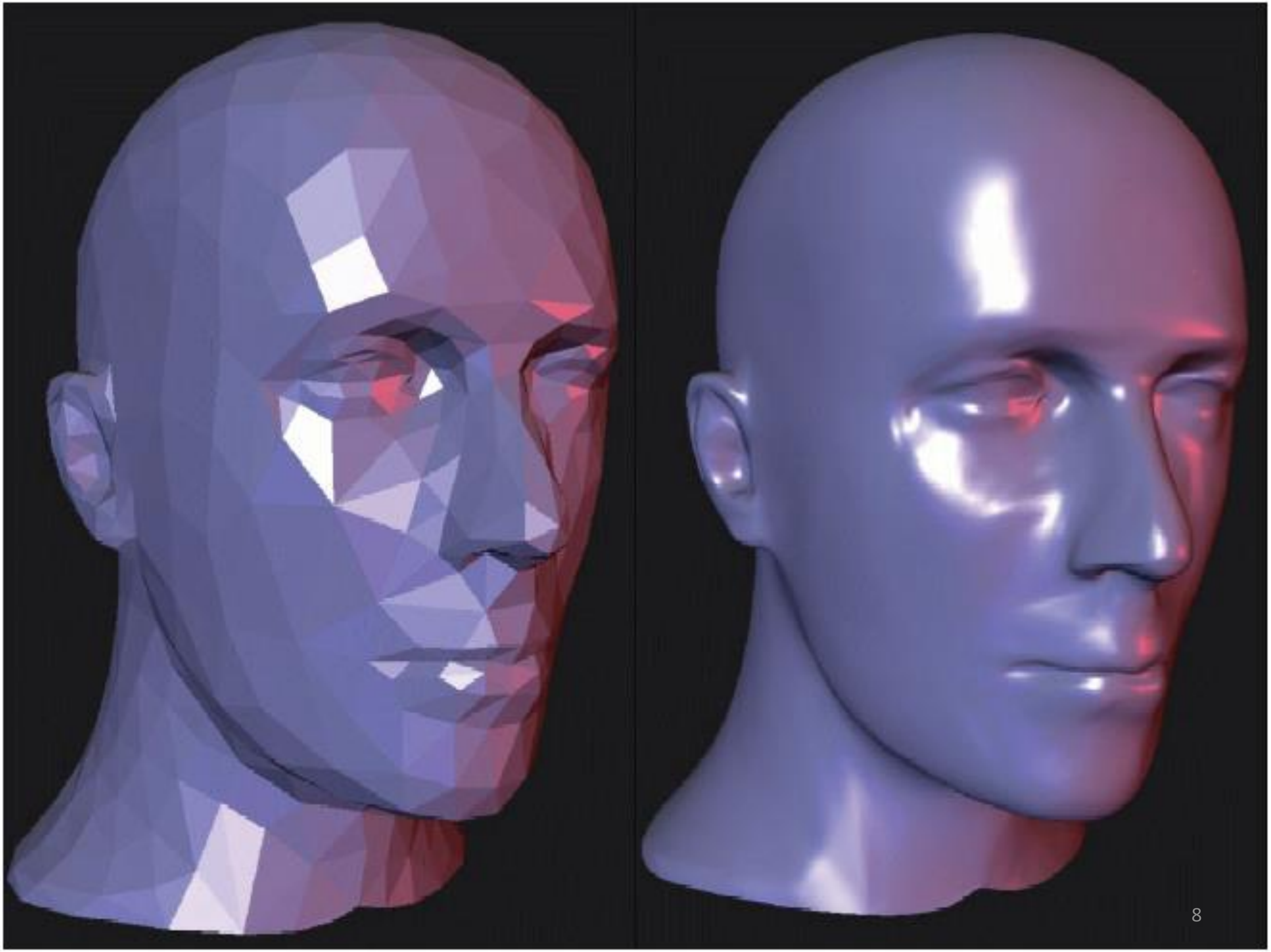
TWO STEP VIEW

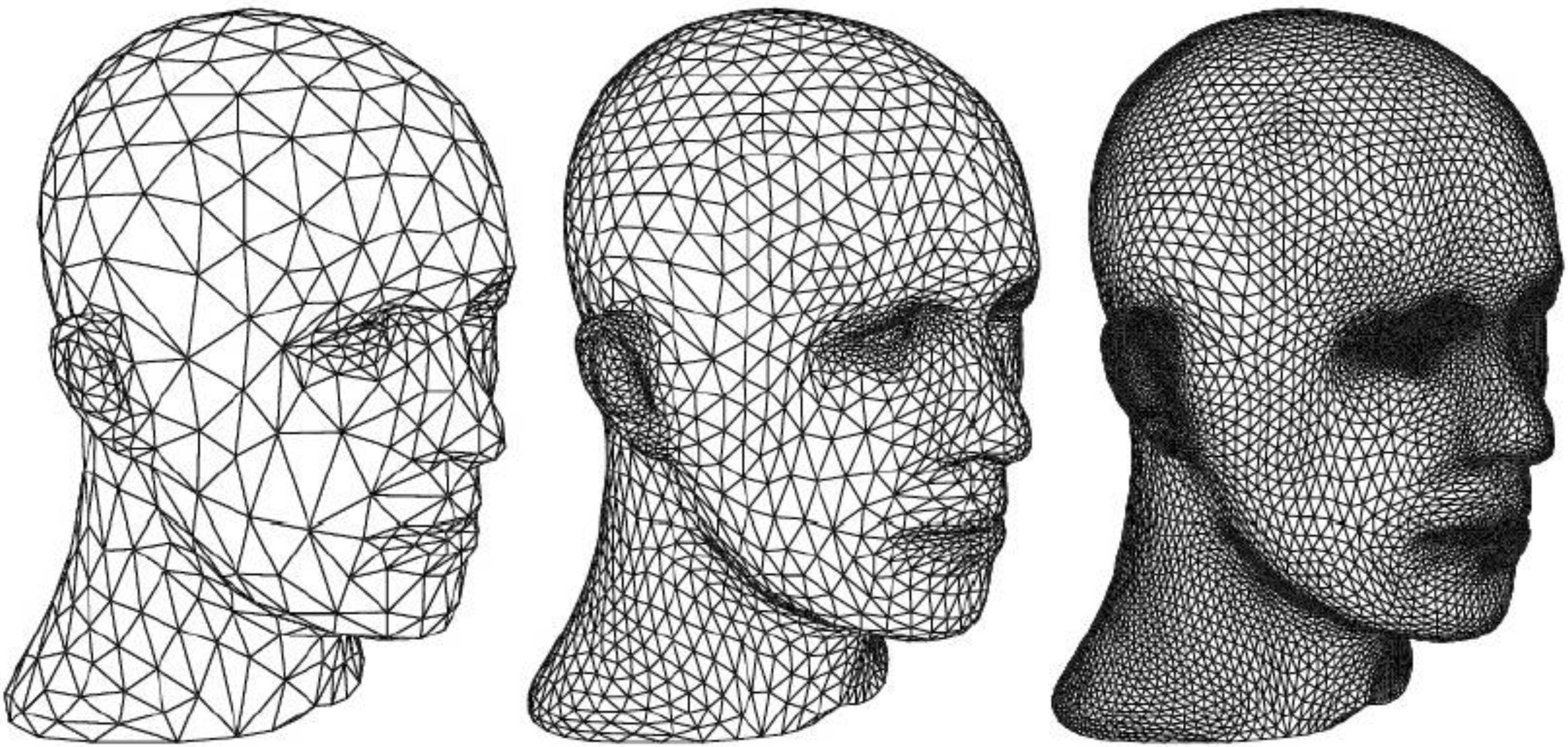
COMBINED / SINGLE STEP VIEW



Subdivision Features

- ***Efficiency***: location of new points computed with small number of FP operations.
- ***Compact support***: region over which a point influences final shape is small and finite.
- ***Affine invariance***: transform of the original set of points followed by subdivision is the same as the transform on the limit shape.
- ***Simplicity***: determining the rules is an offline process and only a small set of rules required.
- ***Continuity***: resulting curves and surfaces are generally nice and smooth, and differentiability nice too!



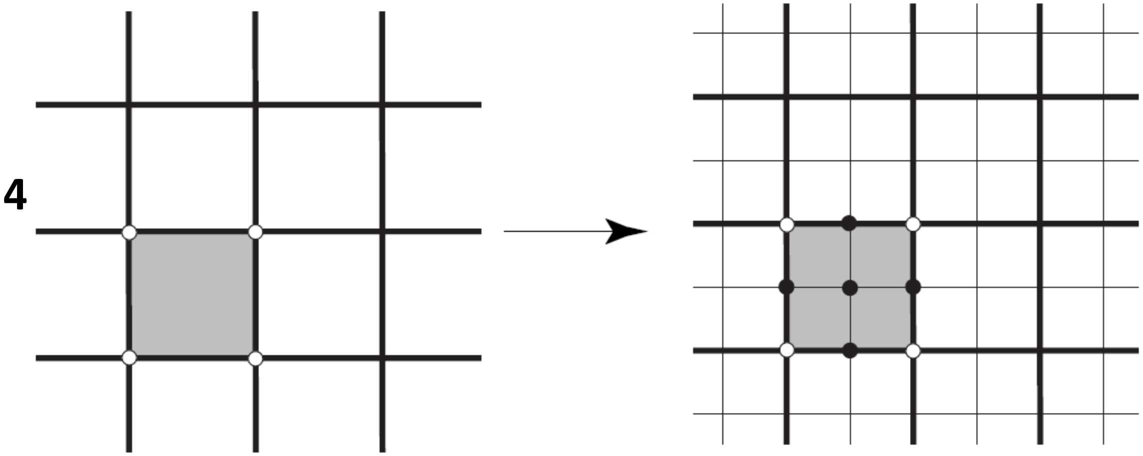


Subdivision of Meshes

- Quadrilaterals

  Regular **vertex has degree 4**
  Extraordinary **vertex has degree ≠ 4**

  – Catmull-Clark (1978)

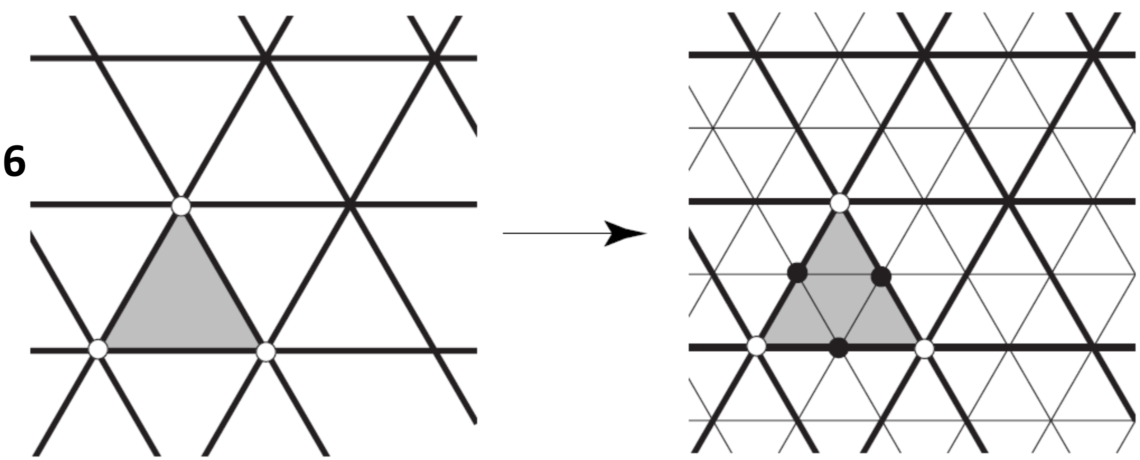


Face split for quads

- Triangles

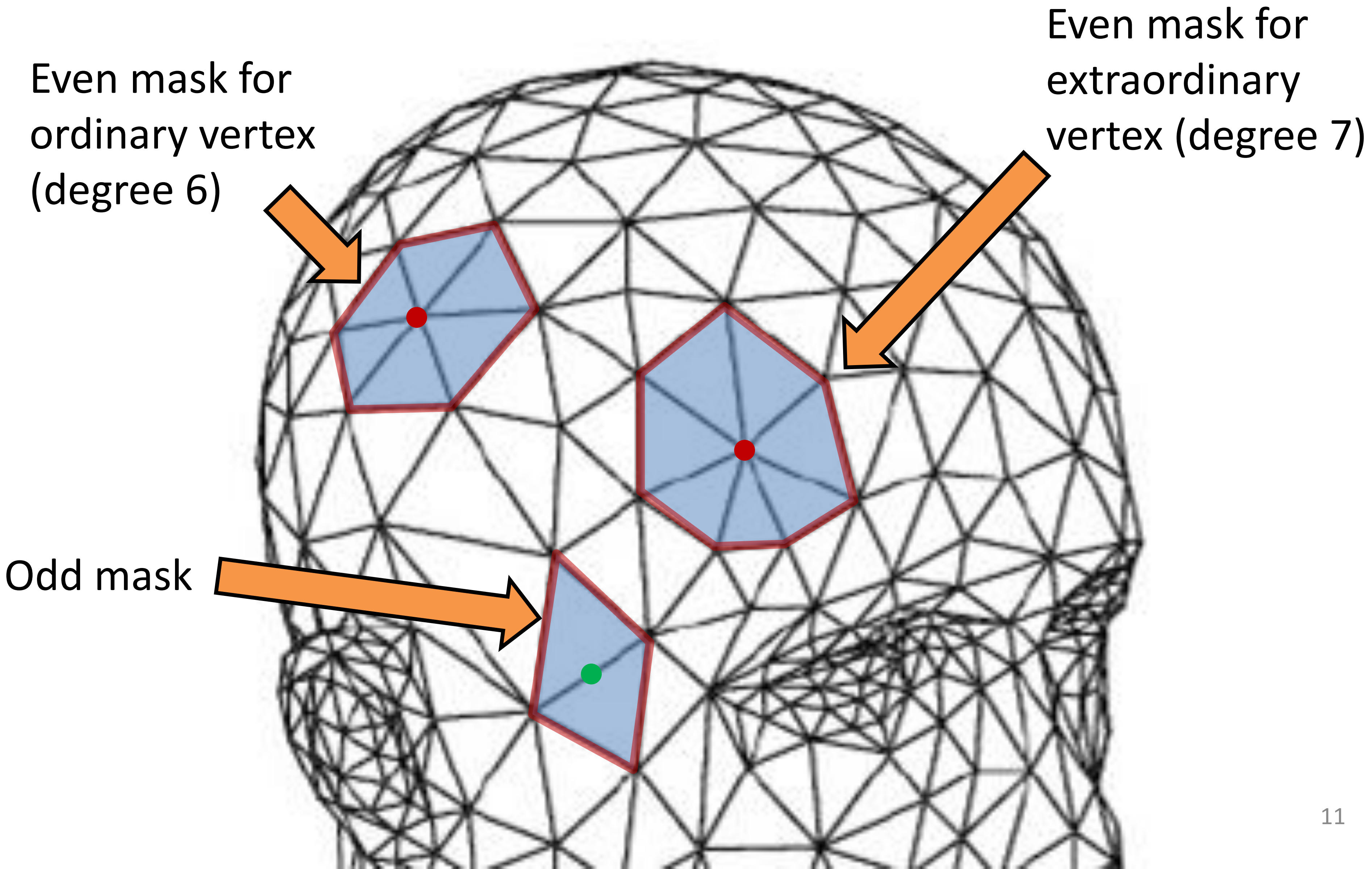
  Regular **vertex has degree 6**
  Extraordinary **vertex has degree ≠ 6**

  – Loop (1987)

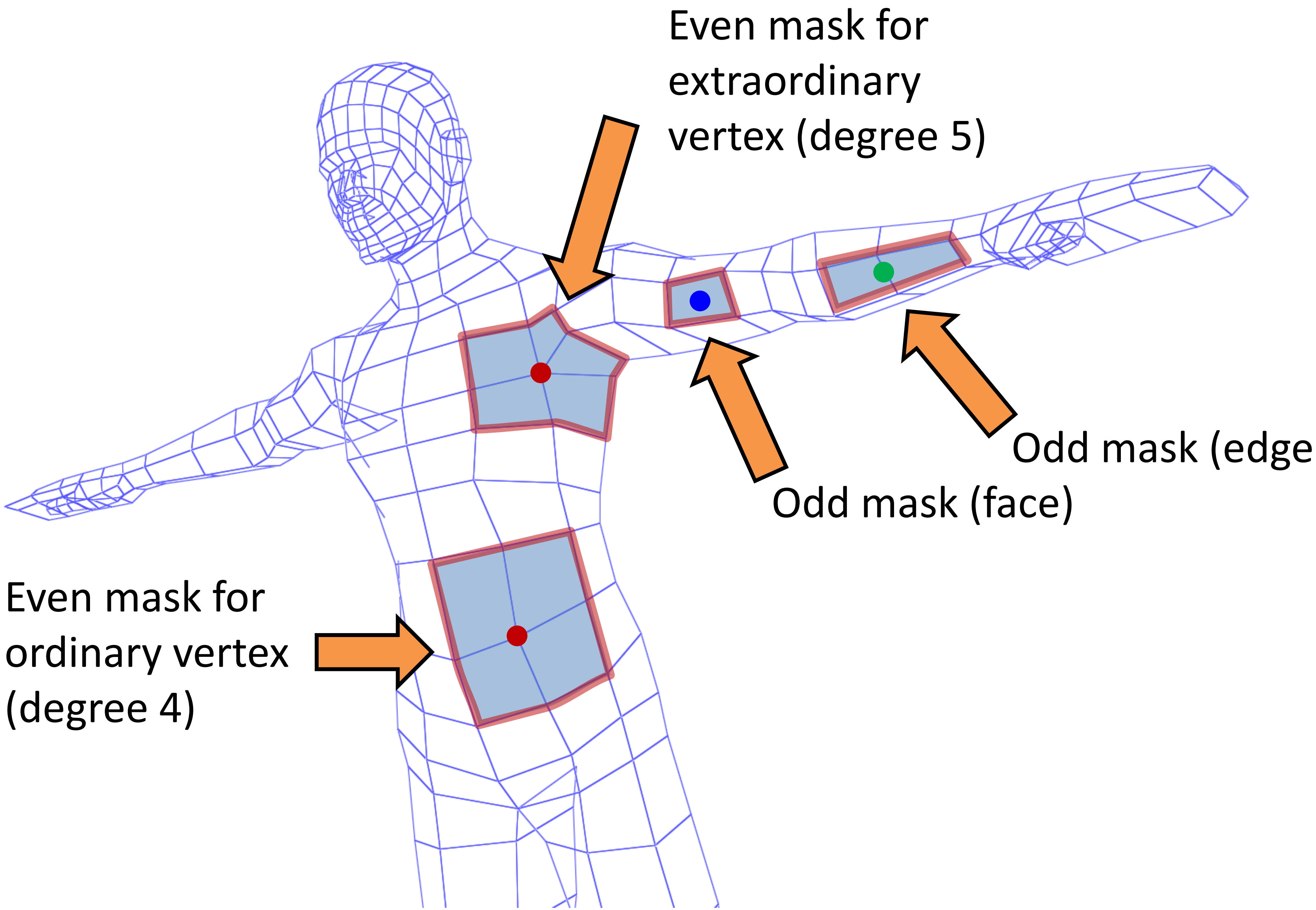


Face split for triangles

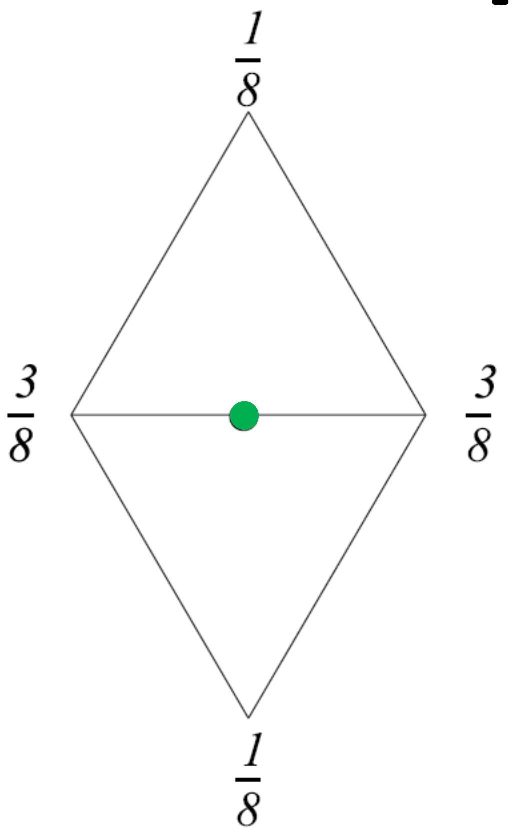
Even and Odd Masks



Even and Odd Masks

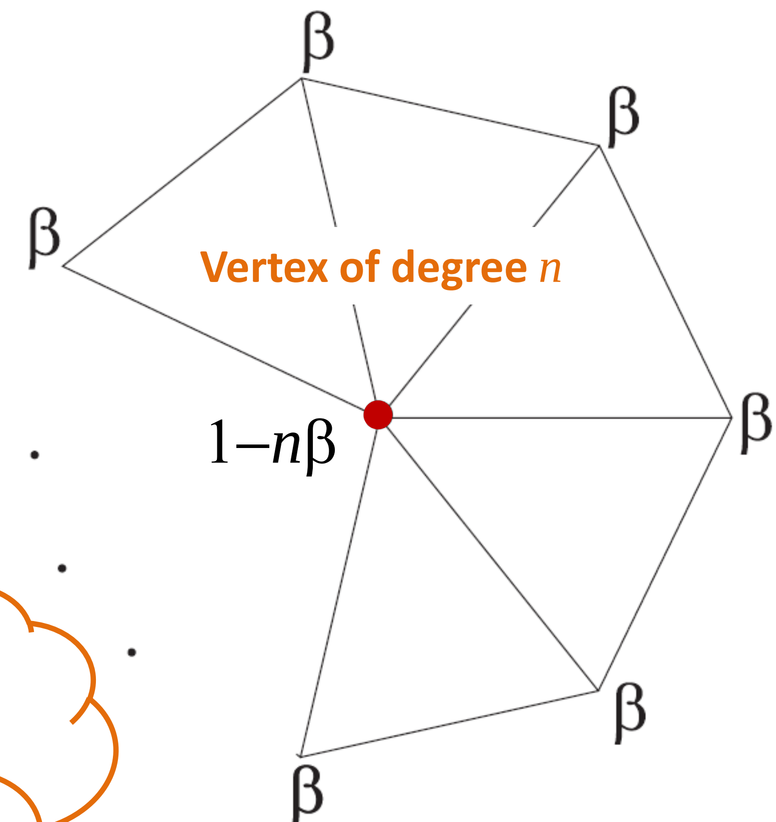


Full Loop Rules (triangle mesh)



Mask for an edge vertex

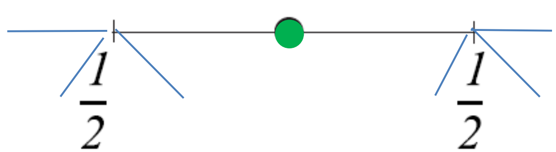
Interior

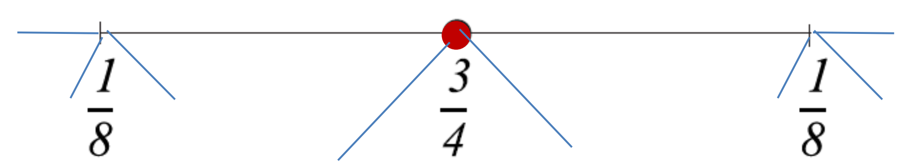


Can also use rules proposed by Warren:
$\beta = 3/(8n)$ for $n > 3$,
$\beta = 3/16$ for $n = 3$.

$$\boxed{\beta = \frac{1}{n}\left(5/8 - \left(\frac{3}{8} + \frac{1}{4}\cos\frac{2\pi}{n}\right)^2\right)}$$

Crease and boundary

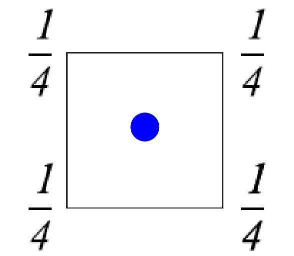




a. Masks for odd vertices

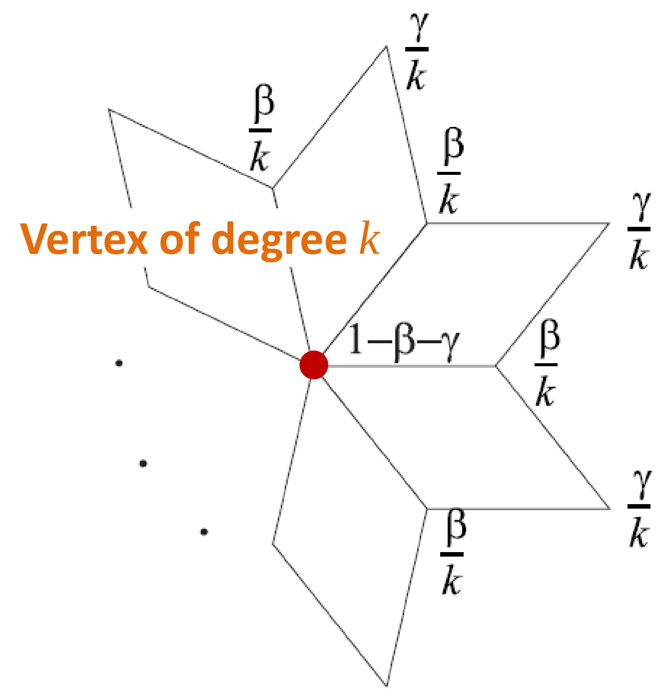
b. Masks for even vertices

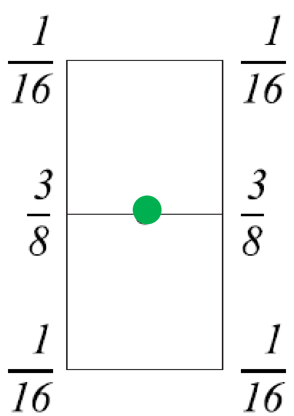
Full Catmull-Clark Rules (quad mesh)



Mask for a face vertex

Interior

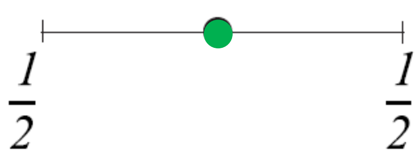


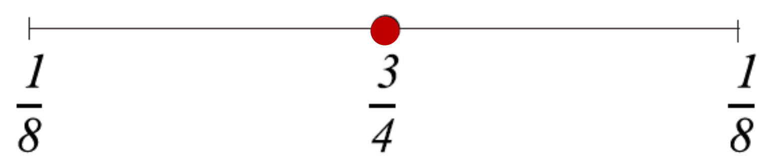


Mask for an edge vertex

$$\boxed{\beta = \frac{3}{2k} \text{ and } \gamma = \frac{1}{4k}}$$

Crease and boundary



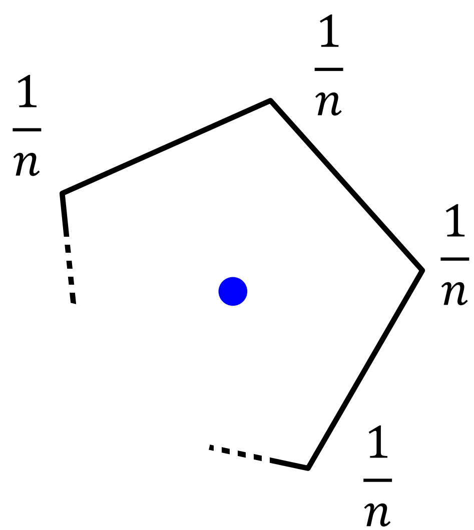


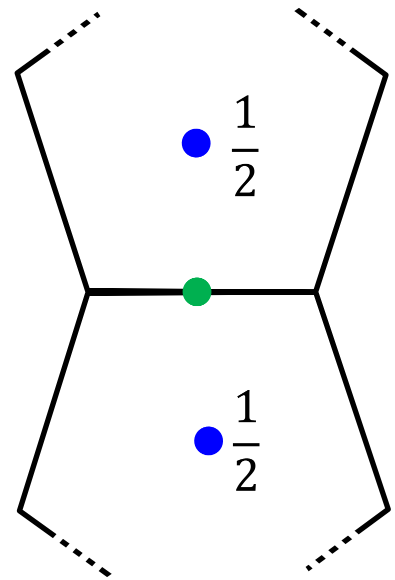
a. Masks for odd vertices

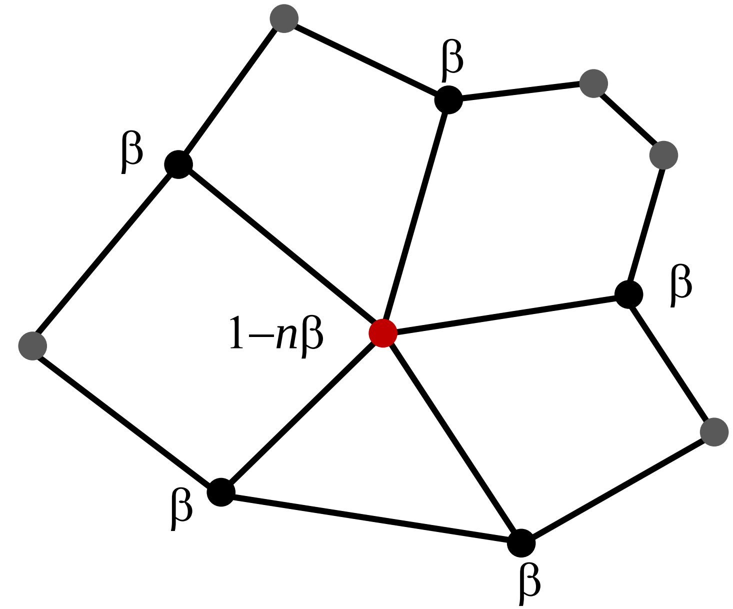
b. Masks for even vertices

Catmull-Clark Rules (n-gons)

Only one subdivision necessary to convert all n-gons into quads







Mask for a n-gon vertex

Average all face vertices to compute the new odd vertex

Mask for an edge vertex

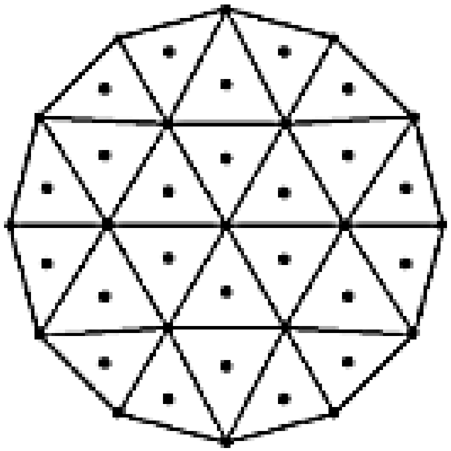
Use the face child vertices (this is equivalent to the normal rule with adjacent quads)

Mask for an even vertex

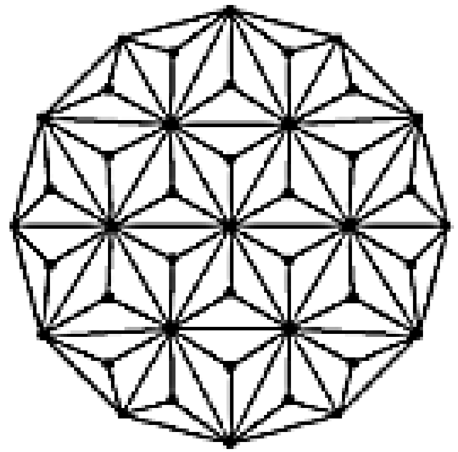
Many options...
But must only use special rule for the first subdivision!

- Could use β from Loop rules (either Loops' or Warren's) with closest adjacent vertices.
- Could use an affine combination of the new n-gon face vertices and even vertex position.
- Could use "original" formula on page 77 from the SIGGRAPH 2000 subdivision course notes

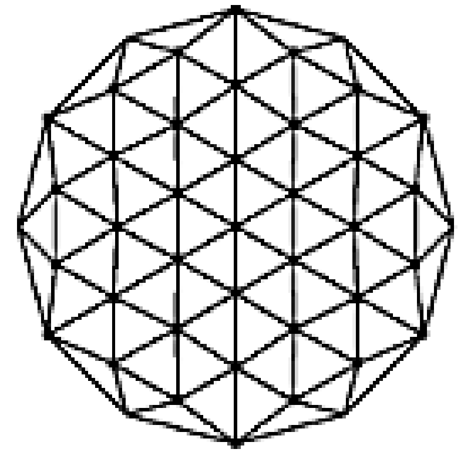
Sqrt(3) Subdivision Rules (triangles)



The split operation places a midvertex at the centre of each triangle



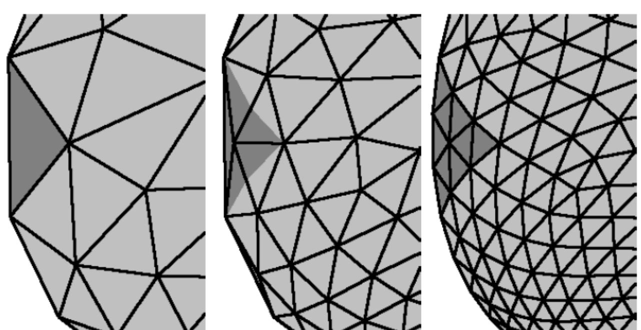
Joining the midvertex to the vertices of the triangle realizes the 1 to 3 split



After smoothing each old vertex, edges are flipped to connect pairs of midvertices

$$\mathbf{q} := \frac{1}{3}\left(\mathbf{p}_i + \mathbf{p}_j + \mathbf{p}_k\right), \qquad S(\mathbf{p}) := (1-\alpha_n)\,\mathbf{p} + \alpha_n \frac{1}{n}\sum_{i=0}^{n-1} \mathbf{p}_i. \qquad \alpha_n = \frac{4 - 2\cos\left(\frac{2\pi}{n}\right)}{9}$$

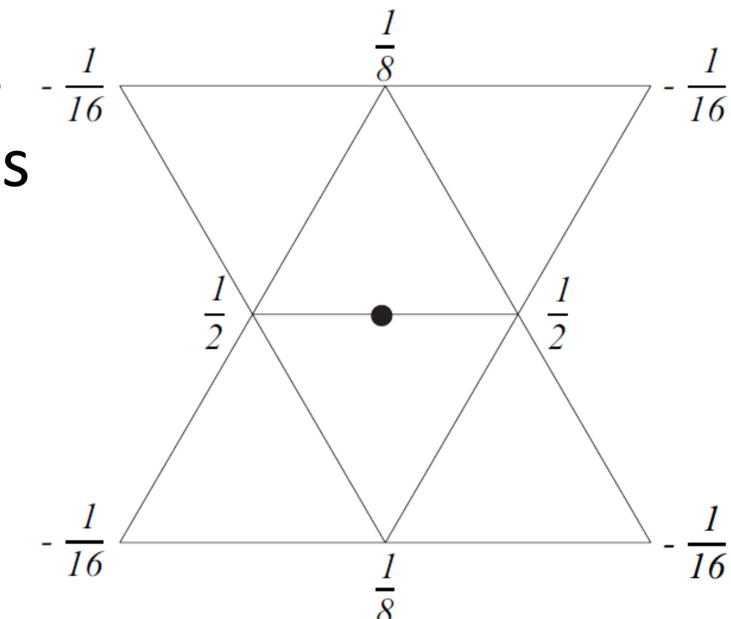
Boundary rules are a bit tricky, involving two vertices inserted every 2nd subdivision



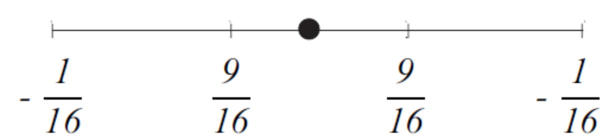
$$\begin{aligned}
\mathbf{p}'_{3i-1} &= \frac{1}{27}\left(10\,\mathbf{p}_{i-1} + 16\,\mathbf{p}_i + \mathbf{p}_{i+1}\right) \\
\mathbf{p}'_{3i} &= \frac{1}{27}\left(4\,\mathbf{p}_{i-1} + 19\,\mathbf{p}_i + 4\,\mathbf{p}_{i+1}\right) \\
\mathbf{p}'_{3i+1} &= \frac{1}{27}\left(\mathbf{p}_{i-1} + 16\,\mathbf{p}_i + 10\,\mathbf{p}_{i+1}\right).
\end{aligned}$$

Interpolating Subdivision Schemes

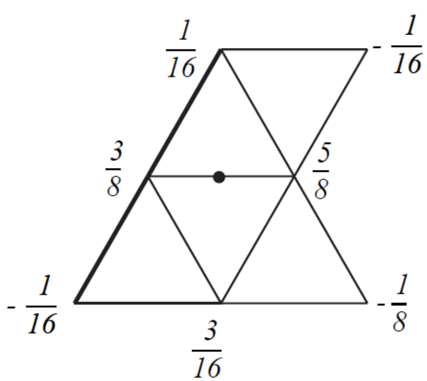
- Loop and sqrt(3) subdivision schemes do not interpolate the original vertices
- Butterfly subdivision Interpolates original mesh vertices!
  - Odd masks only… no even masks!
  - Use loop style rules to deal with extraordinary vertices
  - Use a collection of rules to deal with boundaries and creases



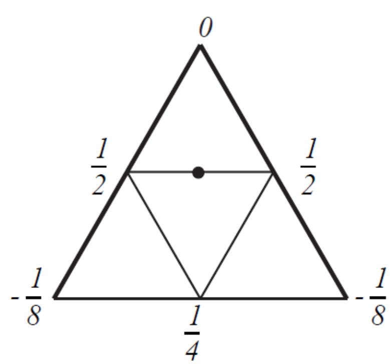
Mask for interior odd vertices with regular neighbors



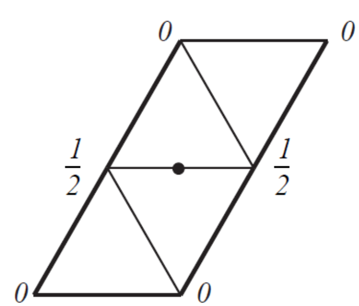
Mask for crease and boundary vertices



interior-crease rule



crease-crease rule 1

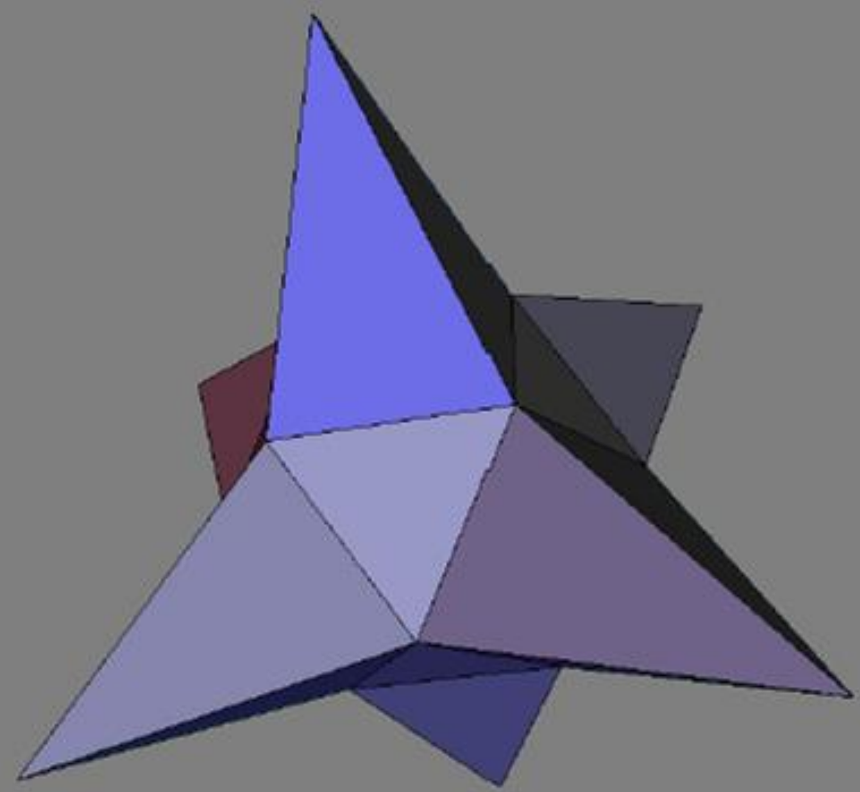


crease-crease rule 2

a. Masks for odd vertices

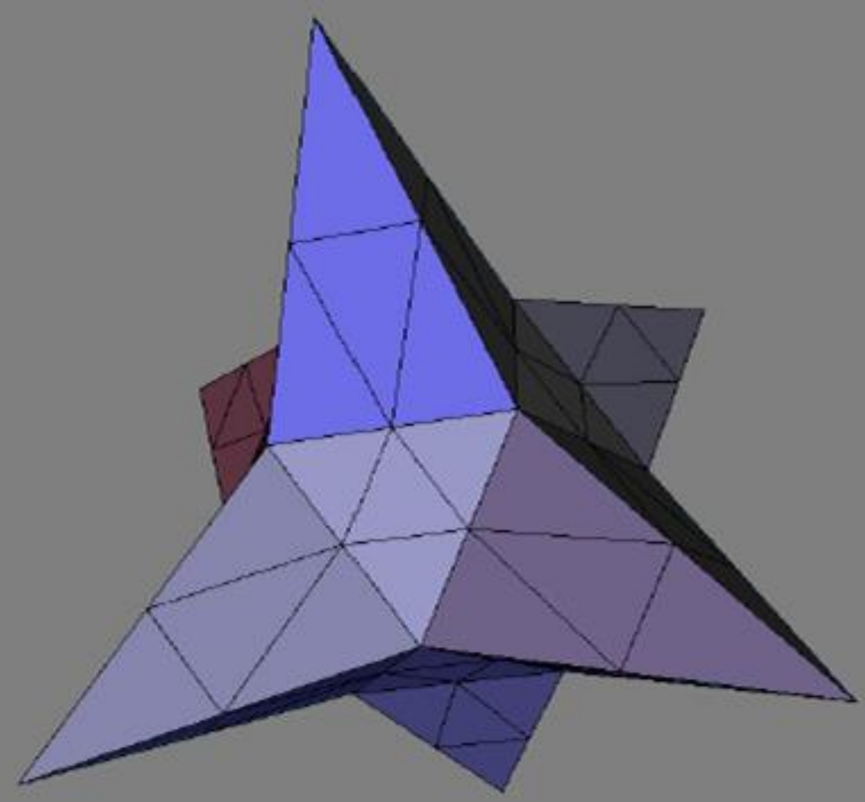
Loop Subdivision Example

Control polyhedron



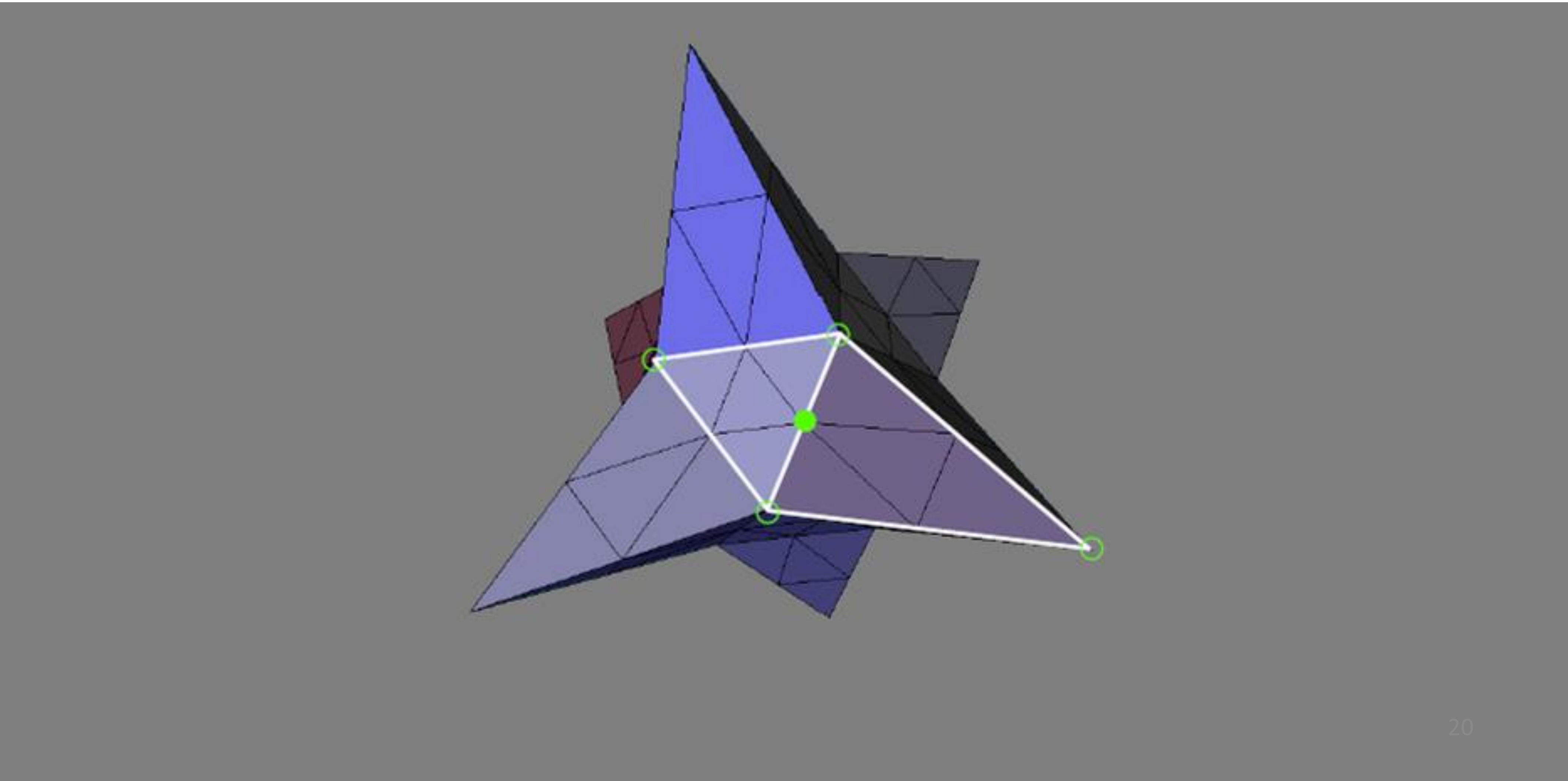
Loop Subdivision Example

Refined Control Polygon



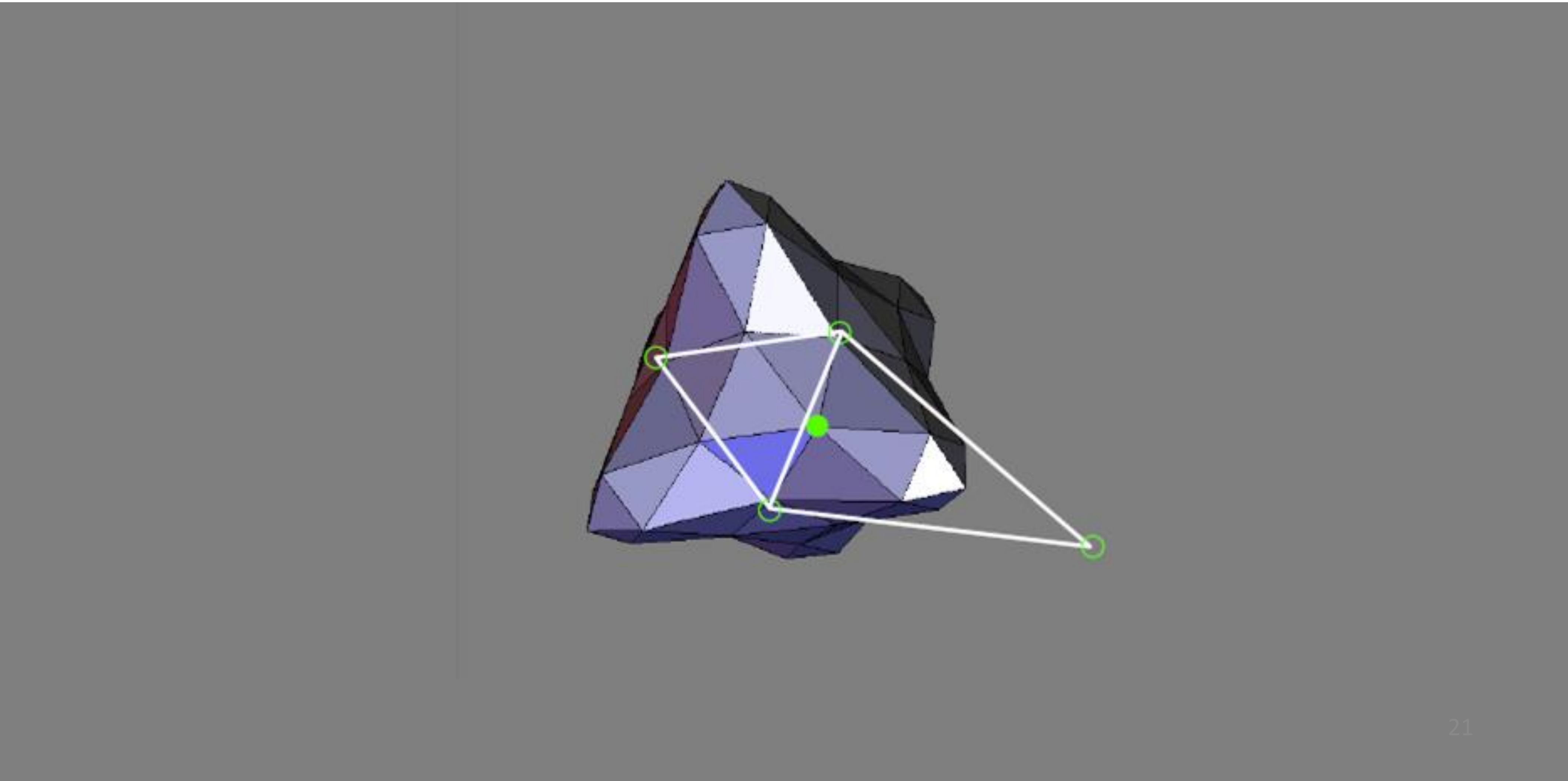
Loop Subdivision Example

Odd Subdivision Mask



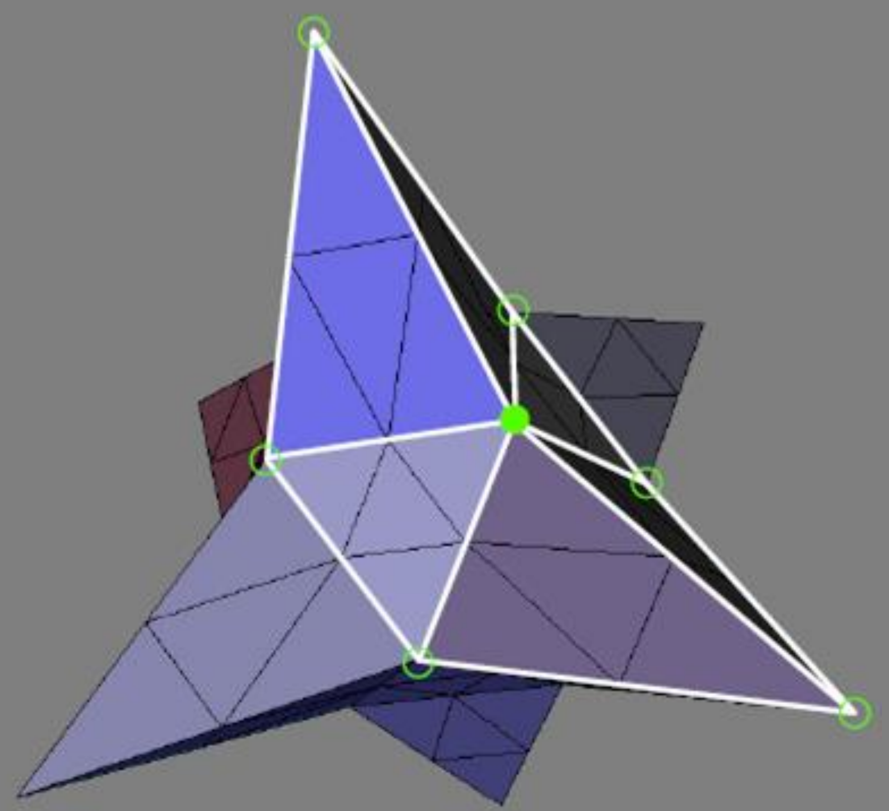
Loop Subdivision Example

Subdivision Level 1



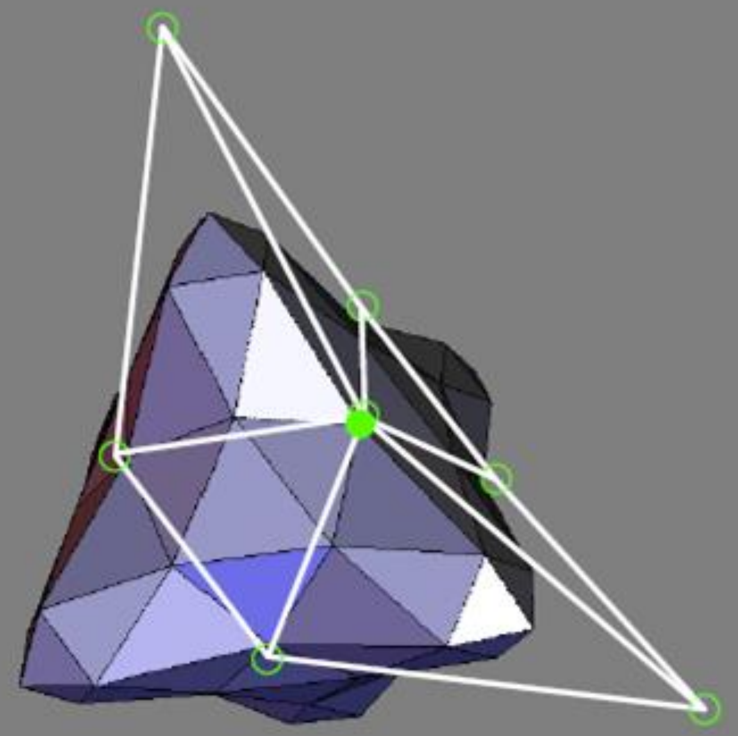
Loop Subdivision Example

Even Subdivision Mask (Ordinary Vertex)



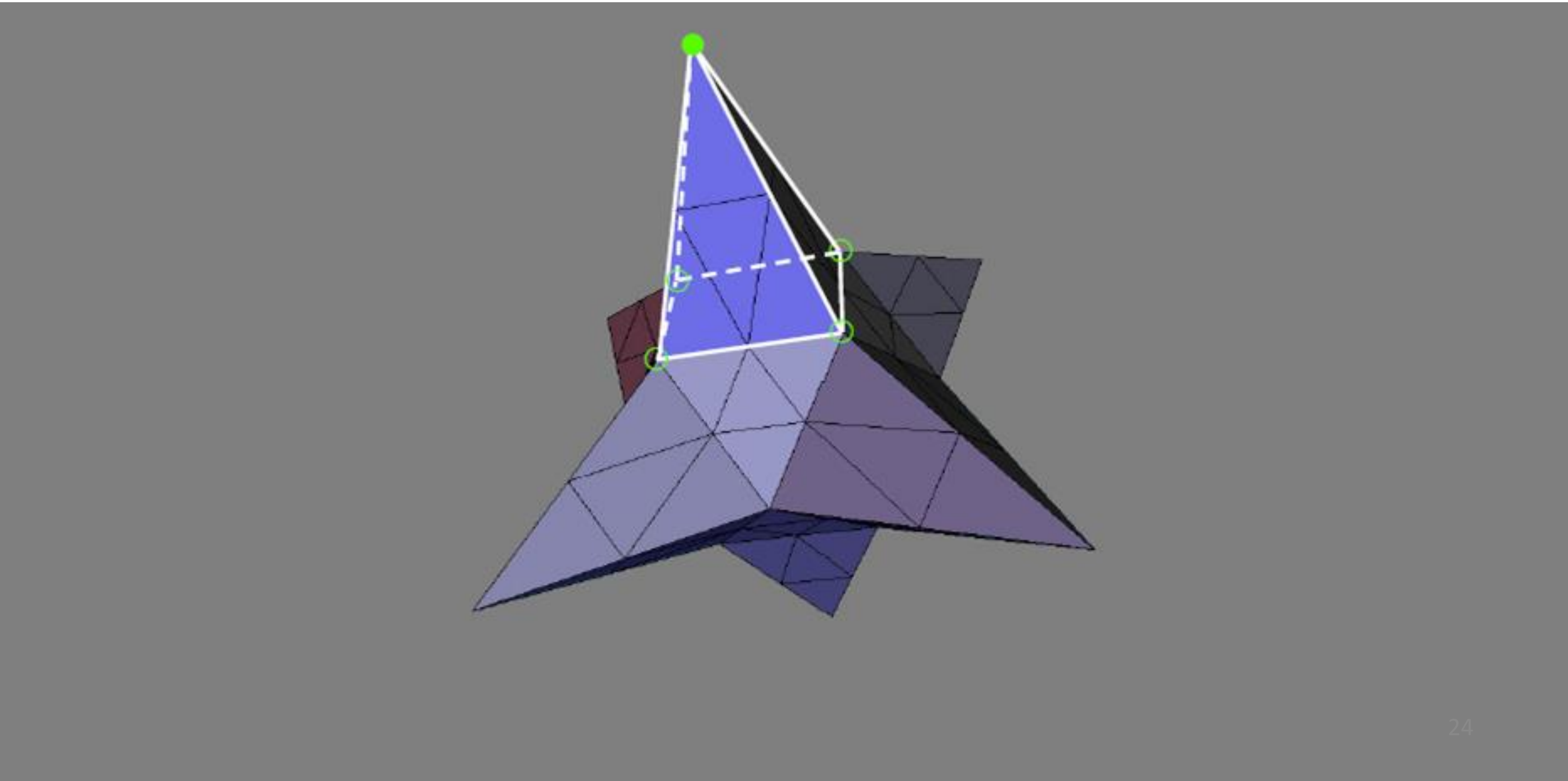
Loop Subdivision Example

Subdivision Level 1



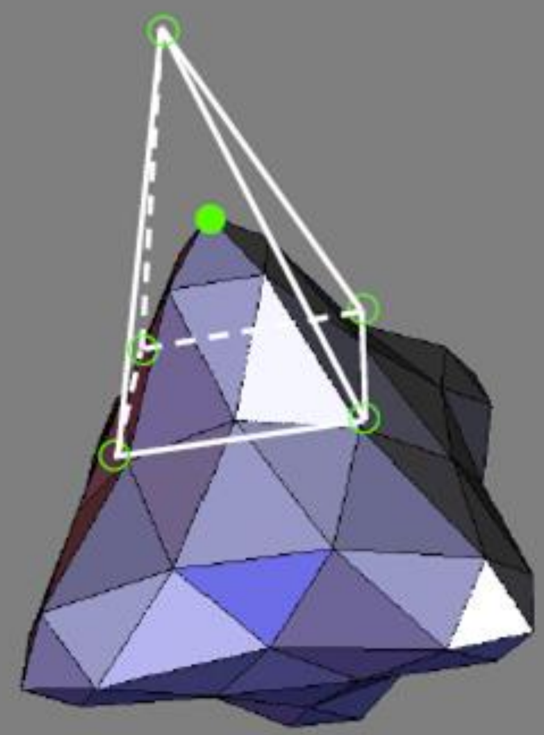
Loop Subdivision Example

Even Subdivision Mask (Extraordinary Vertex)



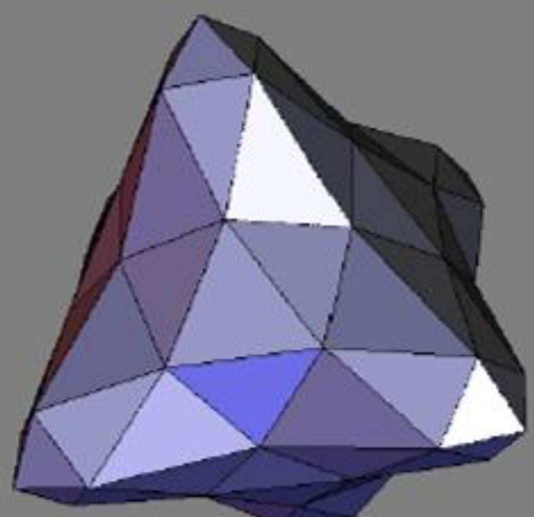
Loop Subdivision Example

Subdivision Level 1



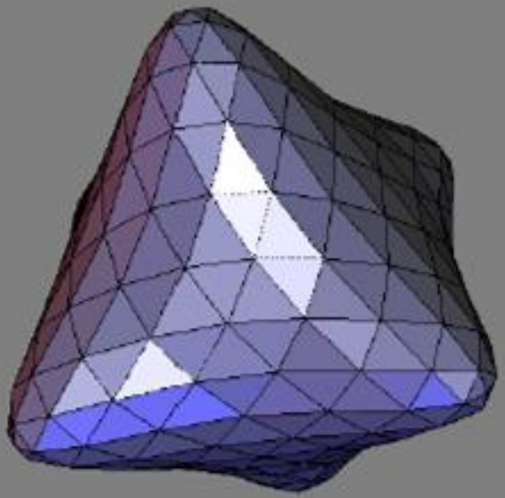
Loop Subdivision Example

Subdivision Level 1



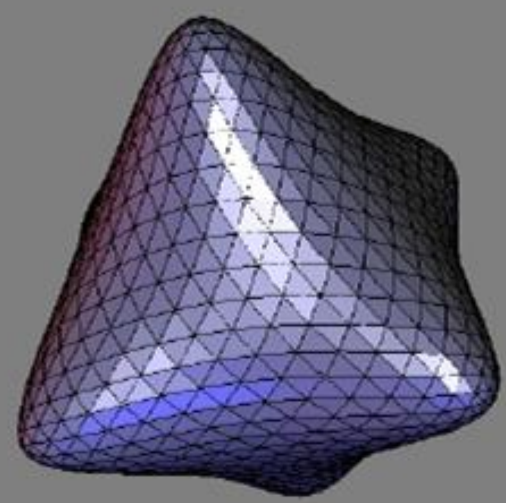
Loop Subdivision Example

Subdivision Level 2



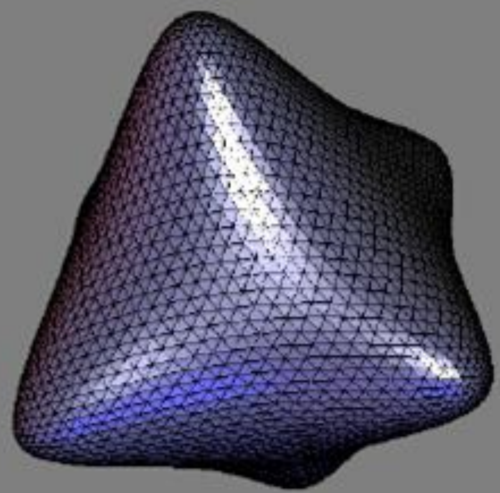
Loop Subdivision Example

Subdivision Level 3



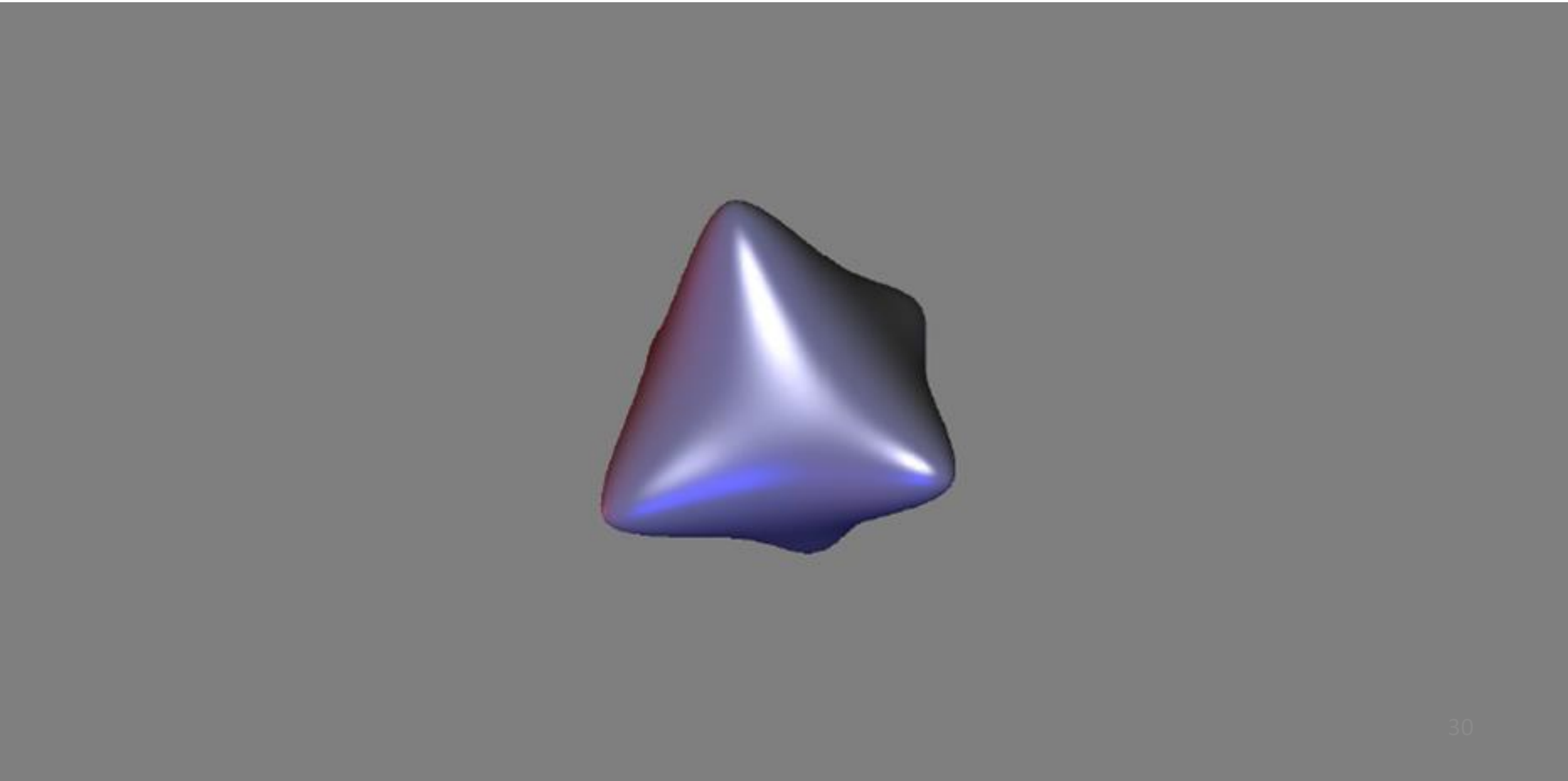
Loop Subdivision Example

Subdivision Level 4

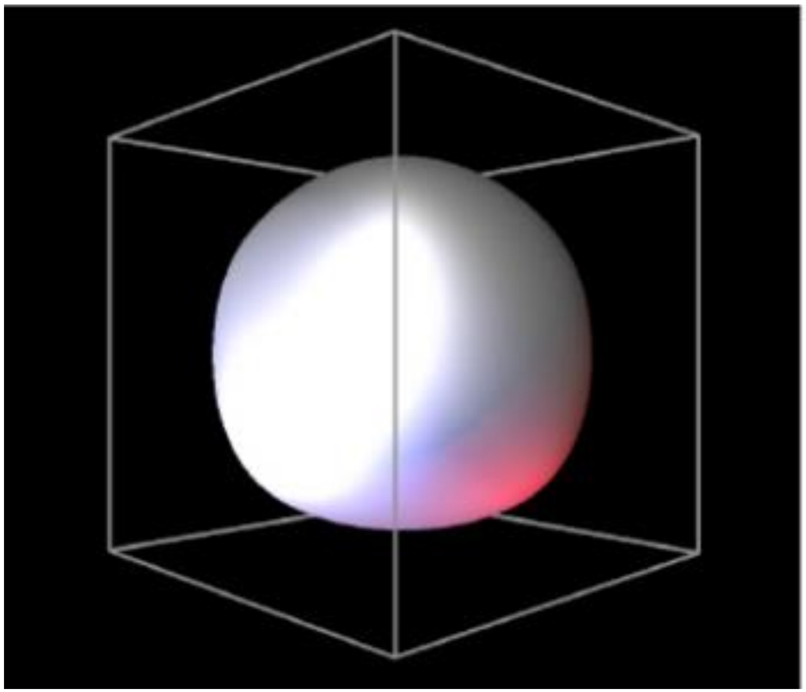


Loop Subdivision Example

Limit Surface

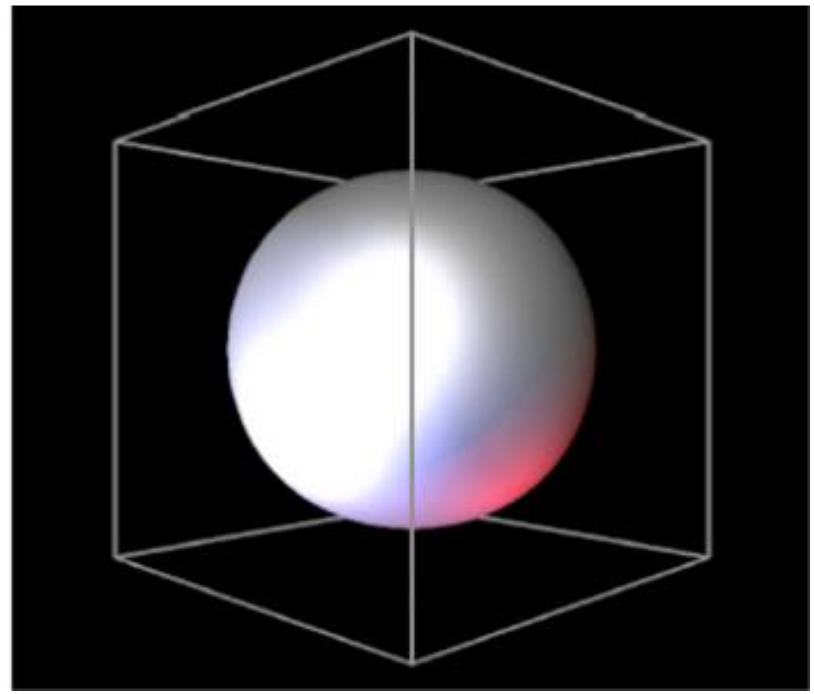


Loop vs Catmull-Clark



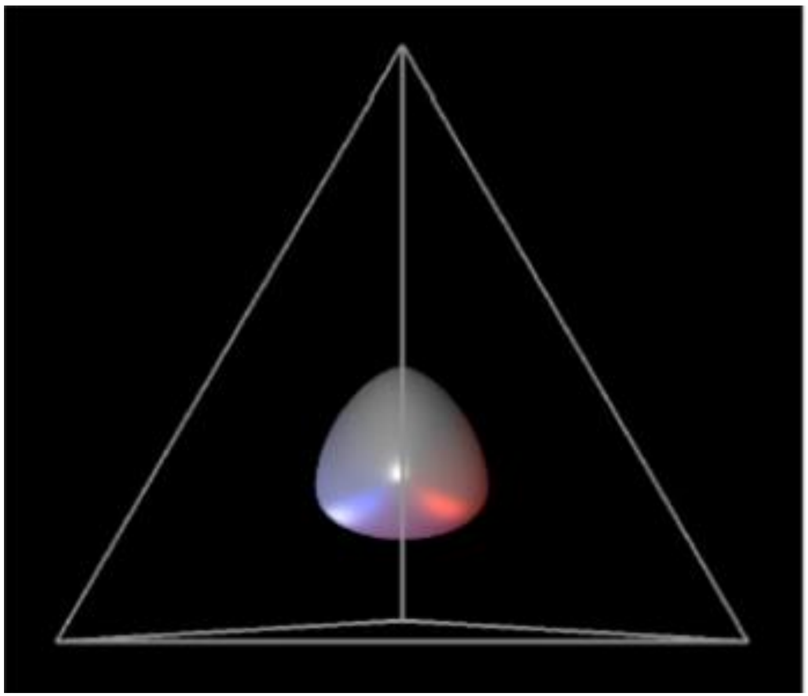
Loop

(split faces before first subdivision step)

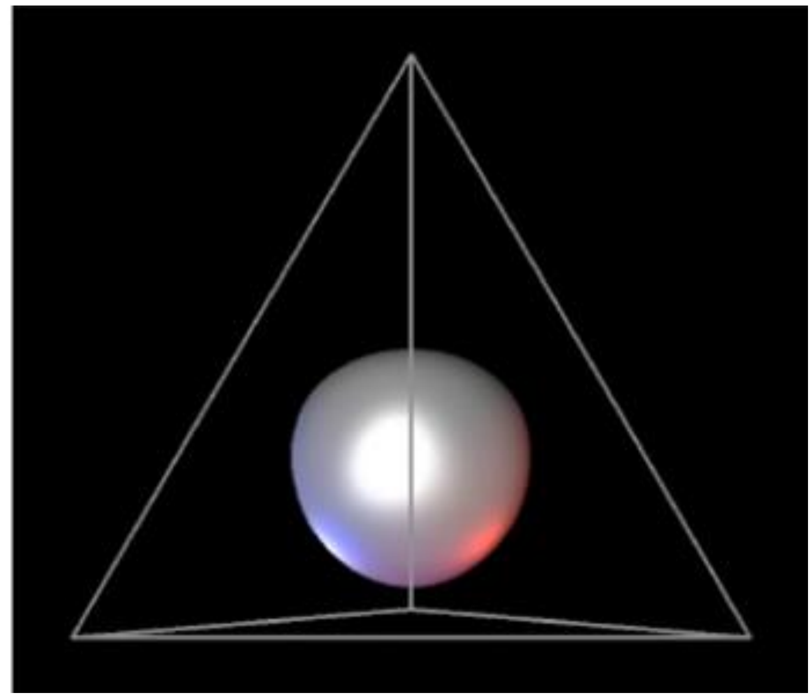


Catmull-Clark

Loop vs Catmull-Clark

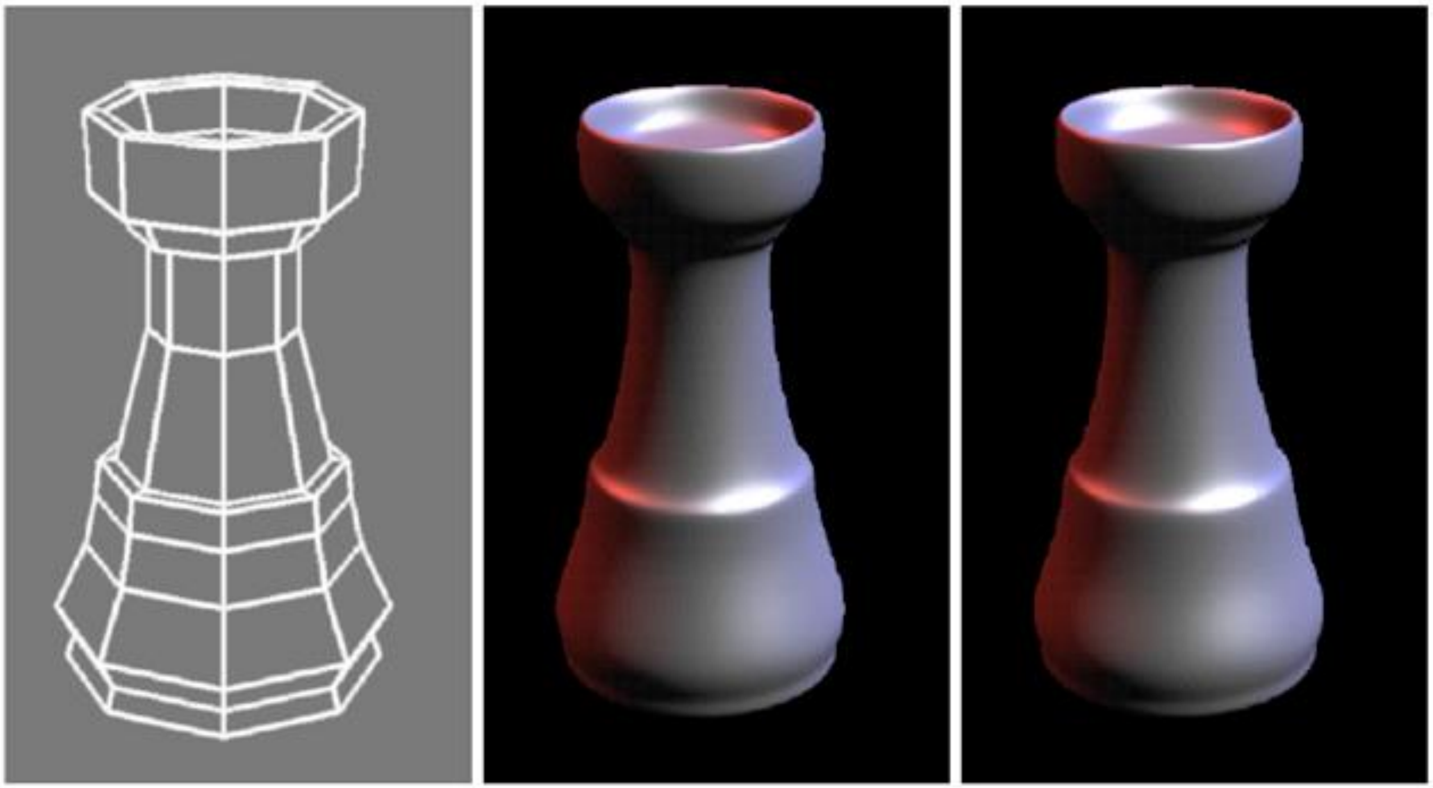


Loop



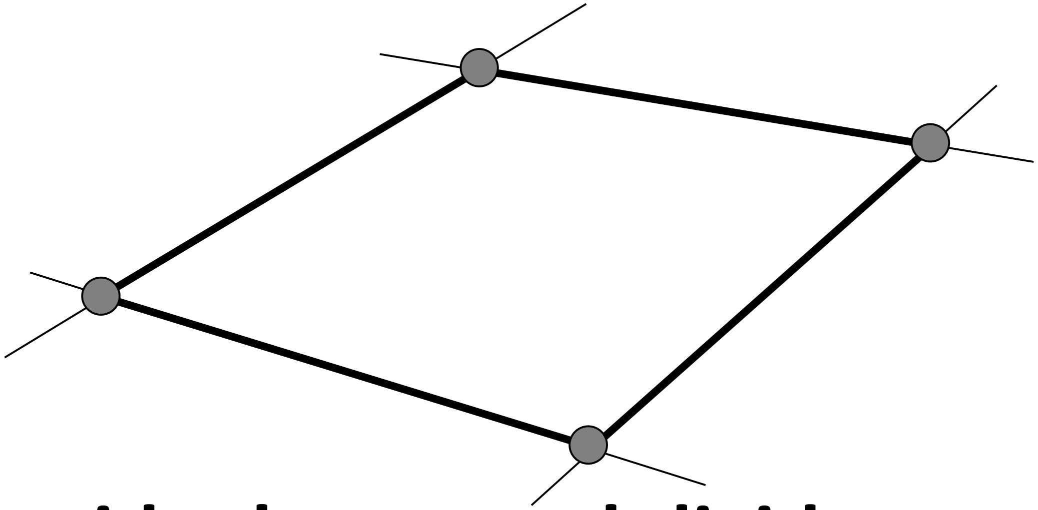
Catmull-Clark
(special first subdivision step to produce quads)

Loop vs Catmull-Clark



Loop
(after splitting faces)

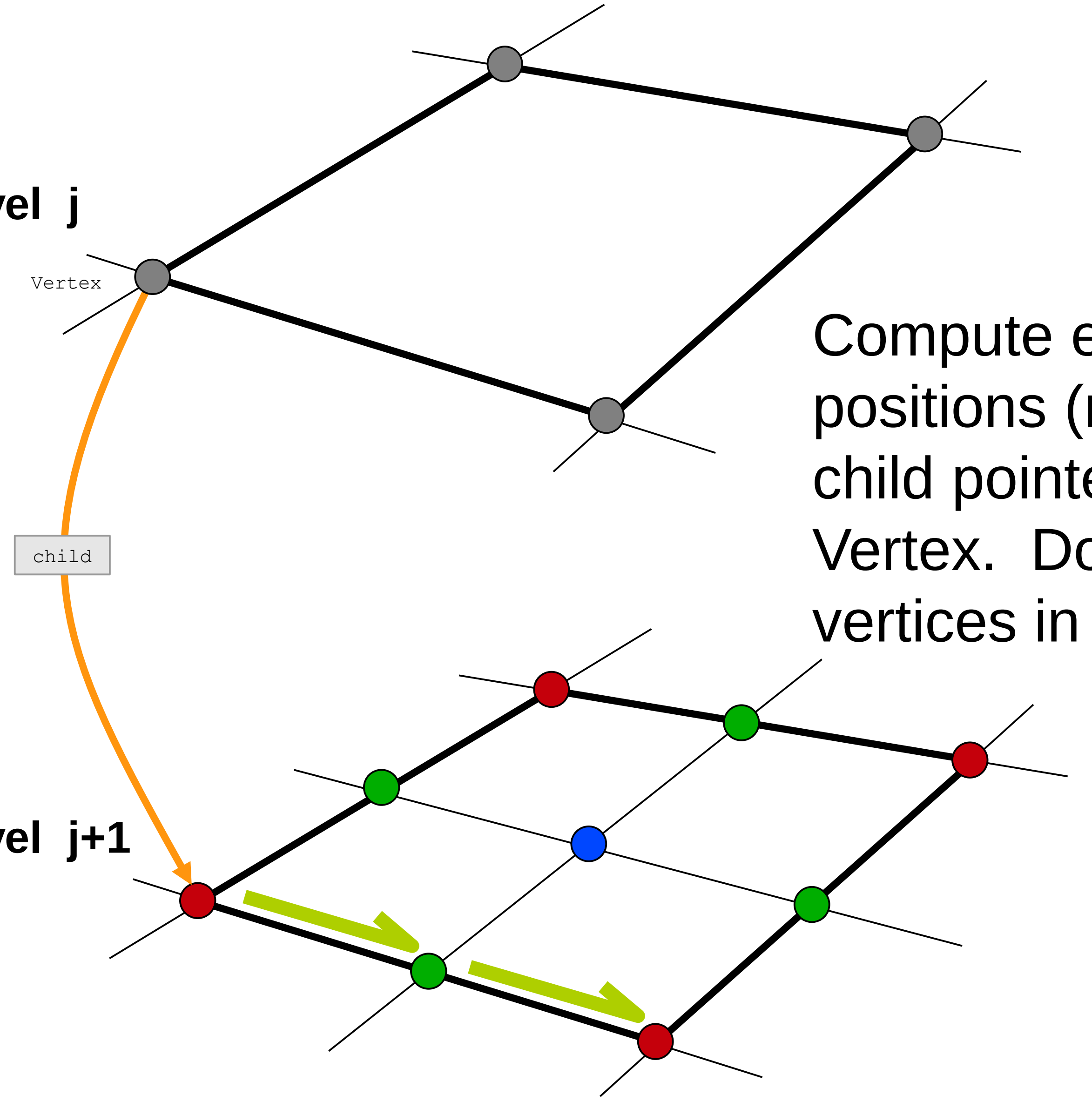
Catmull-Clark



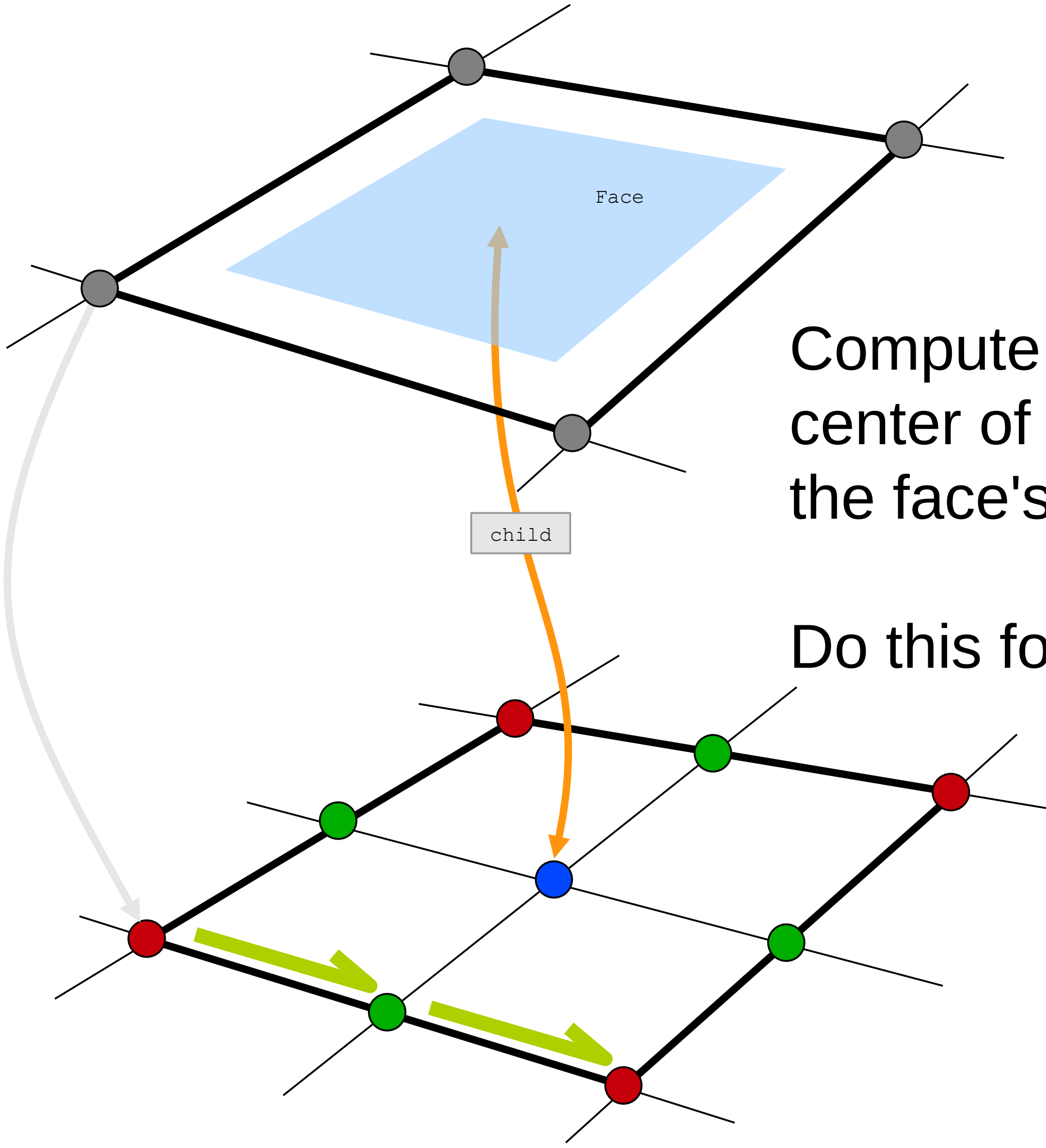
Let us consider how to subdivide a quadrilateral in a *half edge data structure*, for instance, using the Catmull-Clark subdivision scheme.

Subdivision level j

Subdivision level j+1

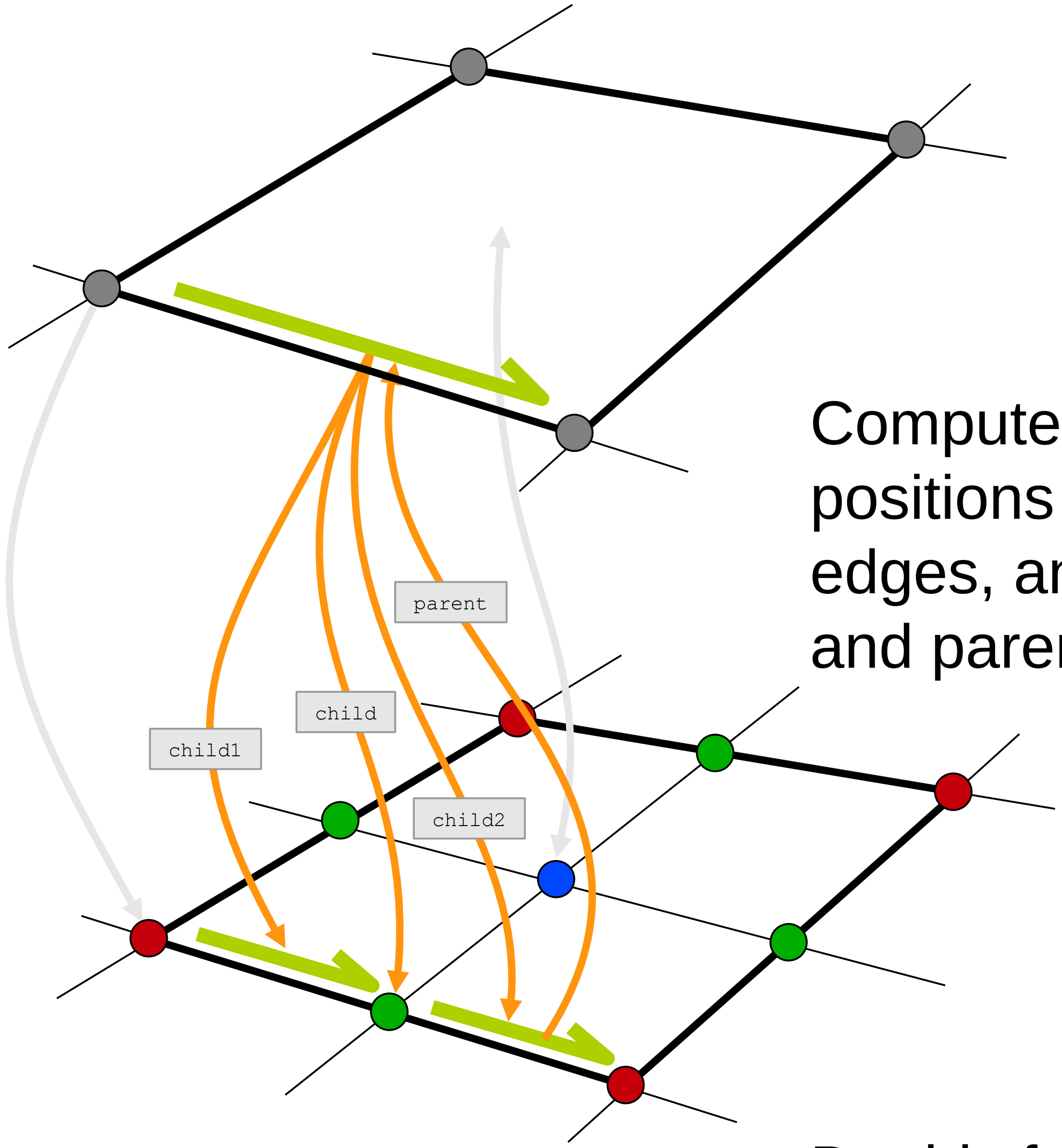


Compute even vertex positions (red), and set child pointers for each Vertex. Do this for all vertices in the face.



Compute odd vertex at center of a Face and set the face's child.

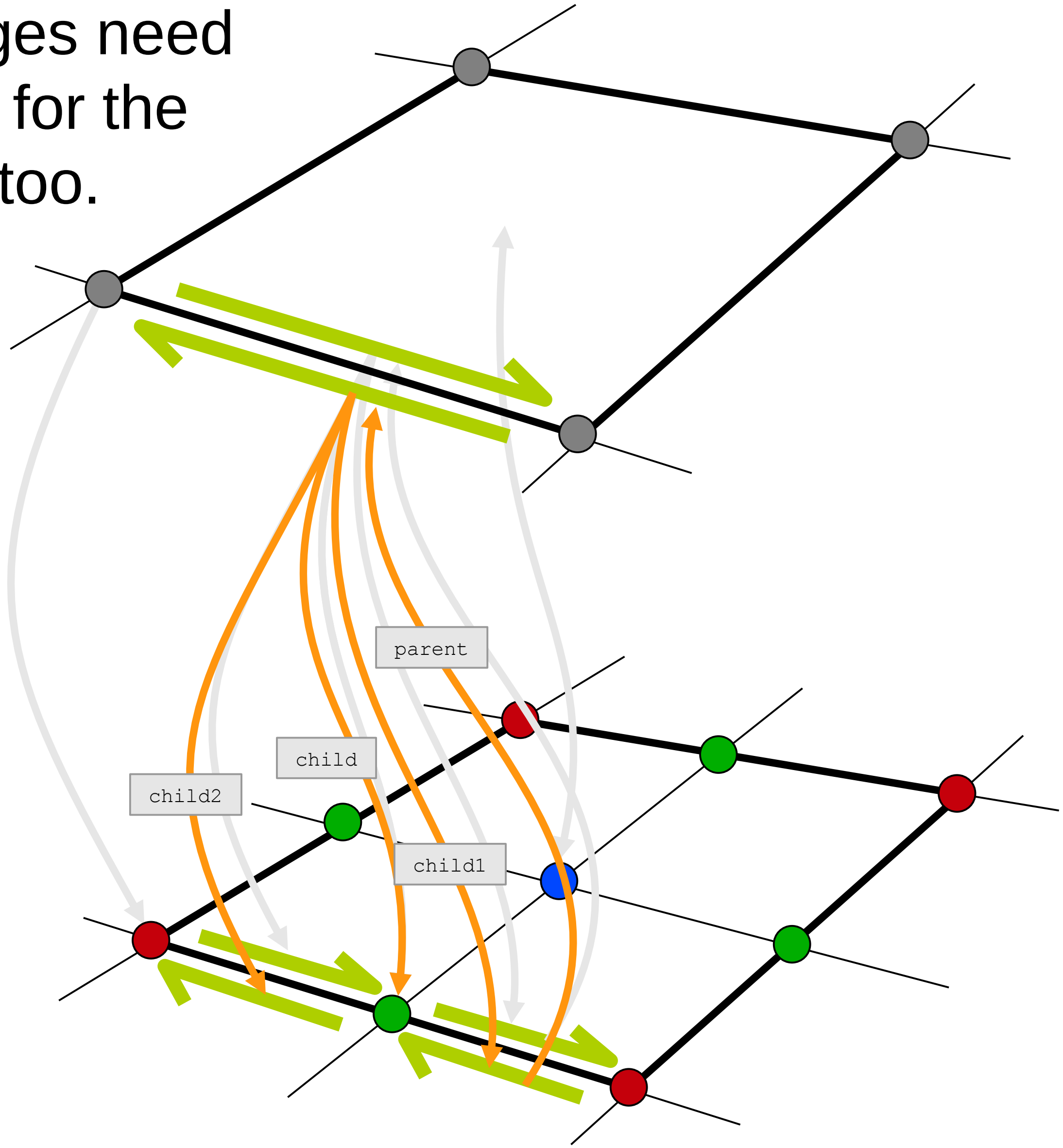
Do this for all faces.

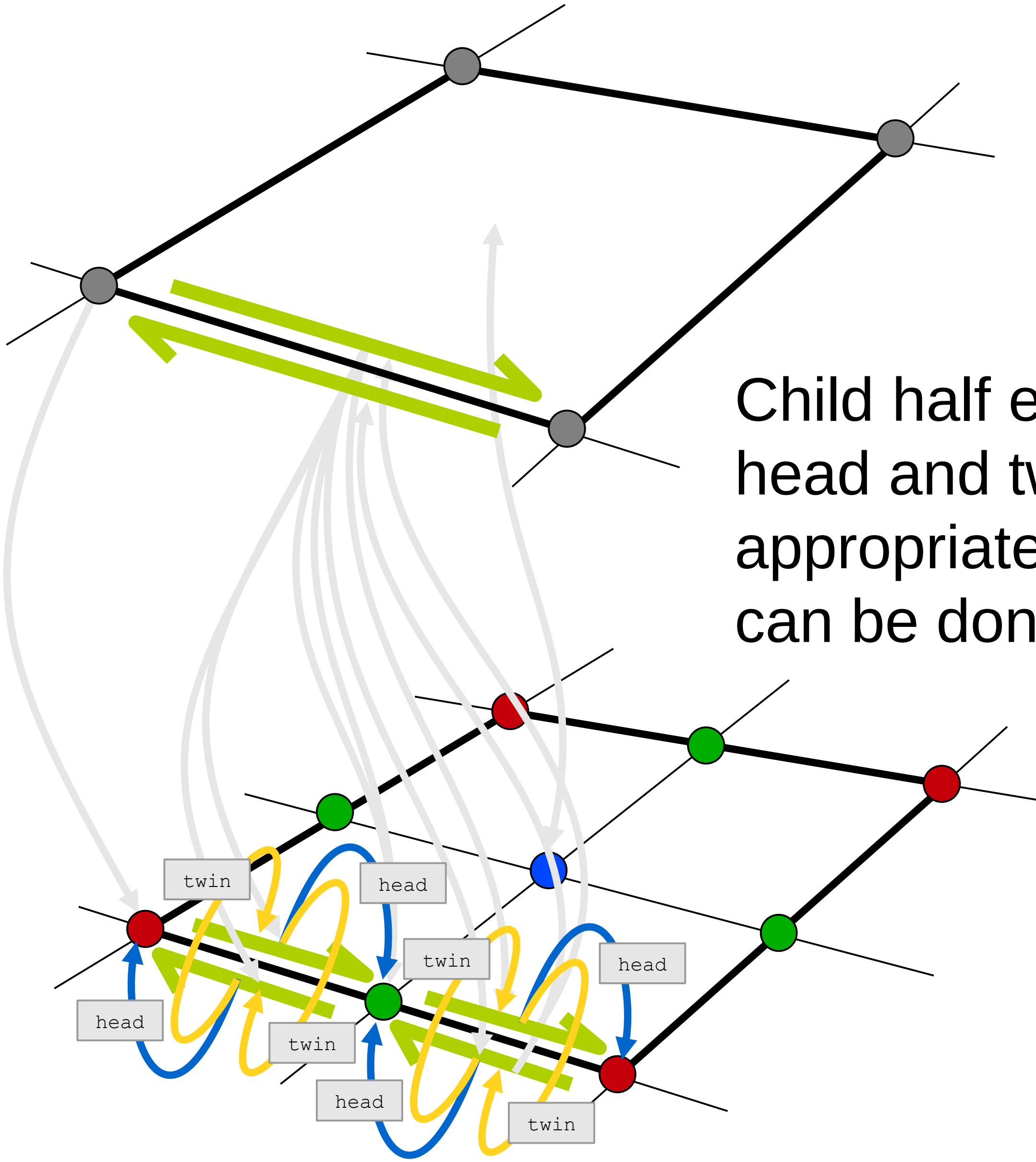


Compute odd vertex positions (green) for edges, and and set child and parent pointers.

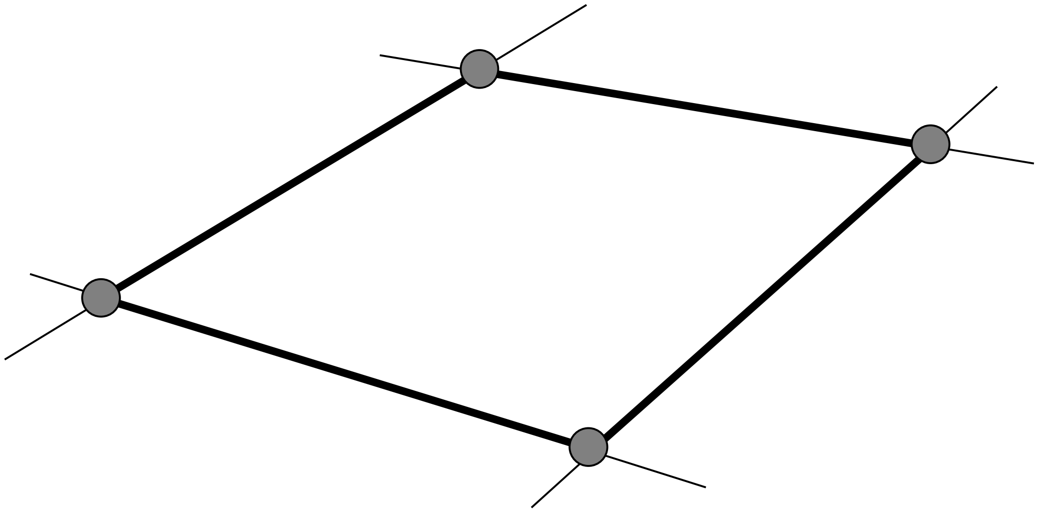
Do this for all half edges.

Child half edges need to be created for the parent's twin too.

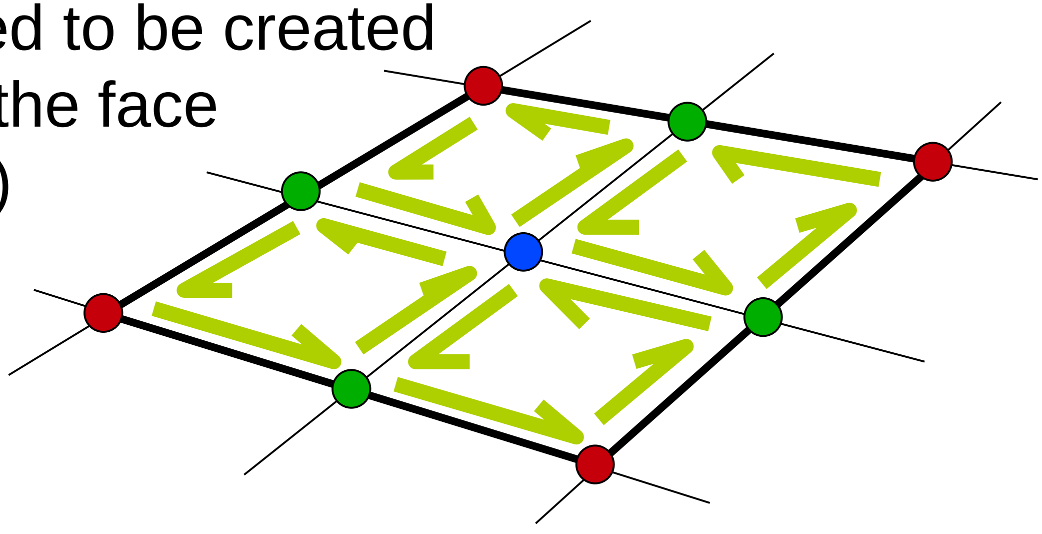


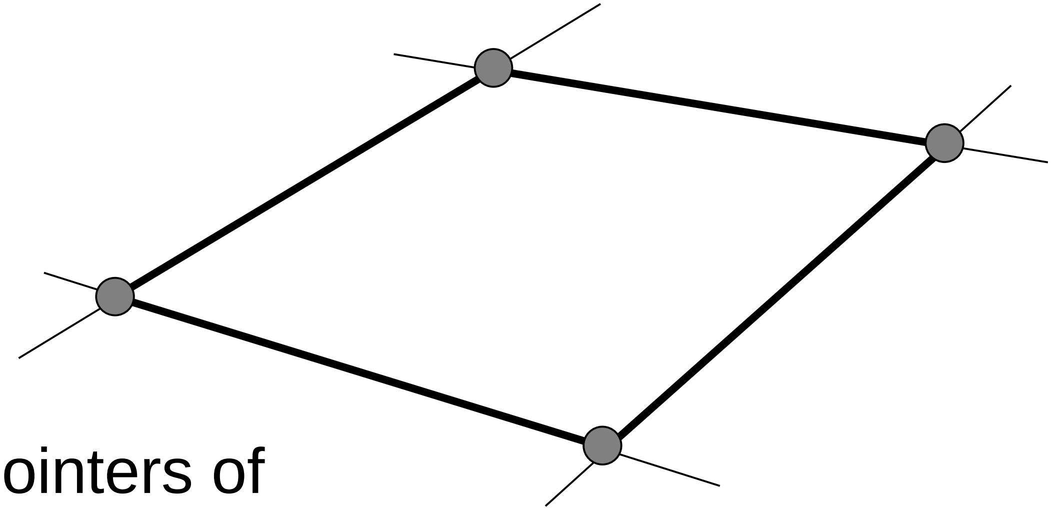


Child half edges need their head and twin pointers set appropriately (next pointers can be done at the very end)

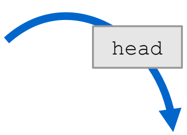


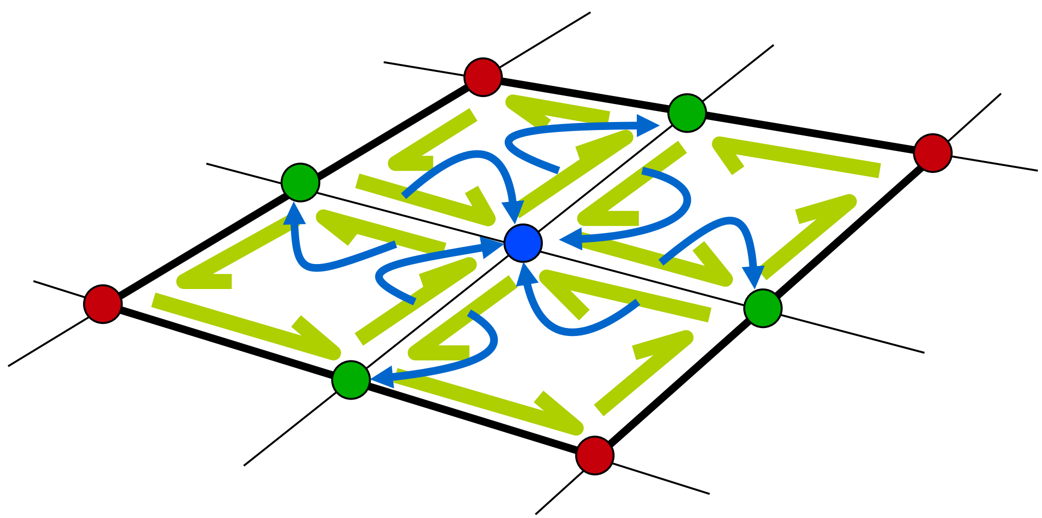
With all of the above complete,
8 half edges need to be created
at the middle of the face
(2n for an n-gon)

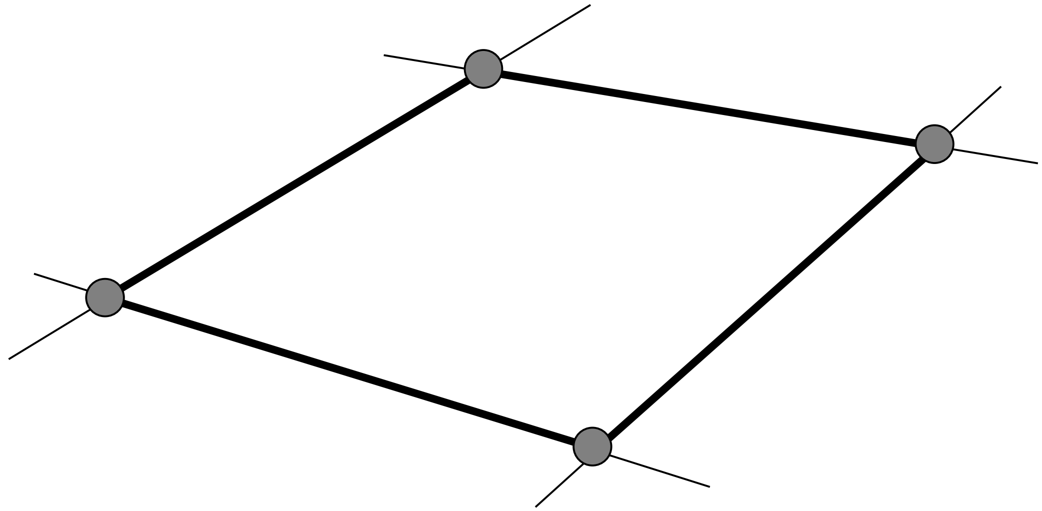




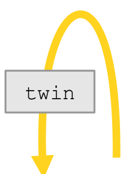
Set the head pointers of these new half edges to the appropriate vertices

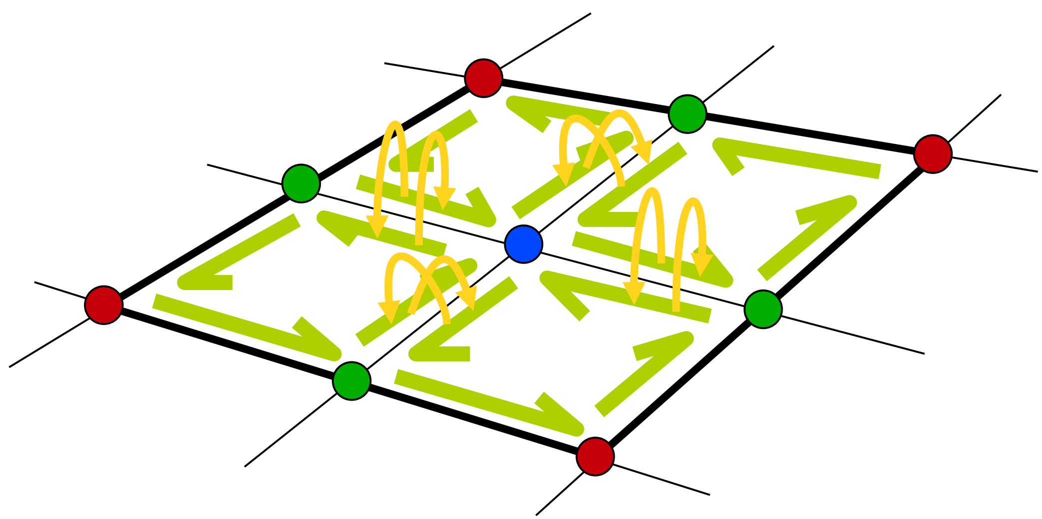


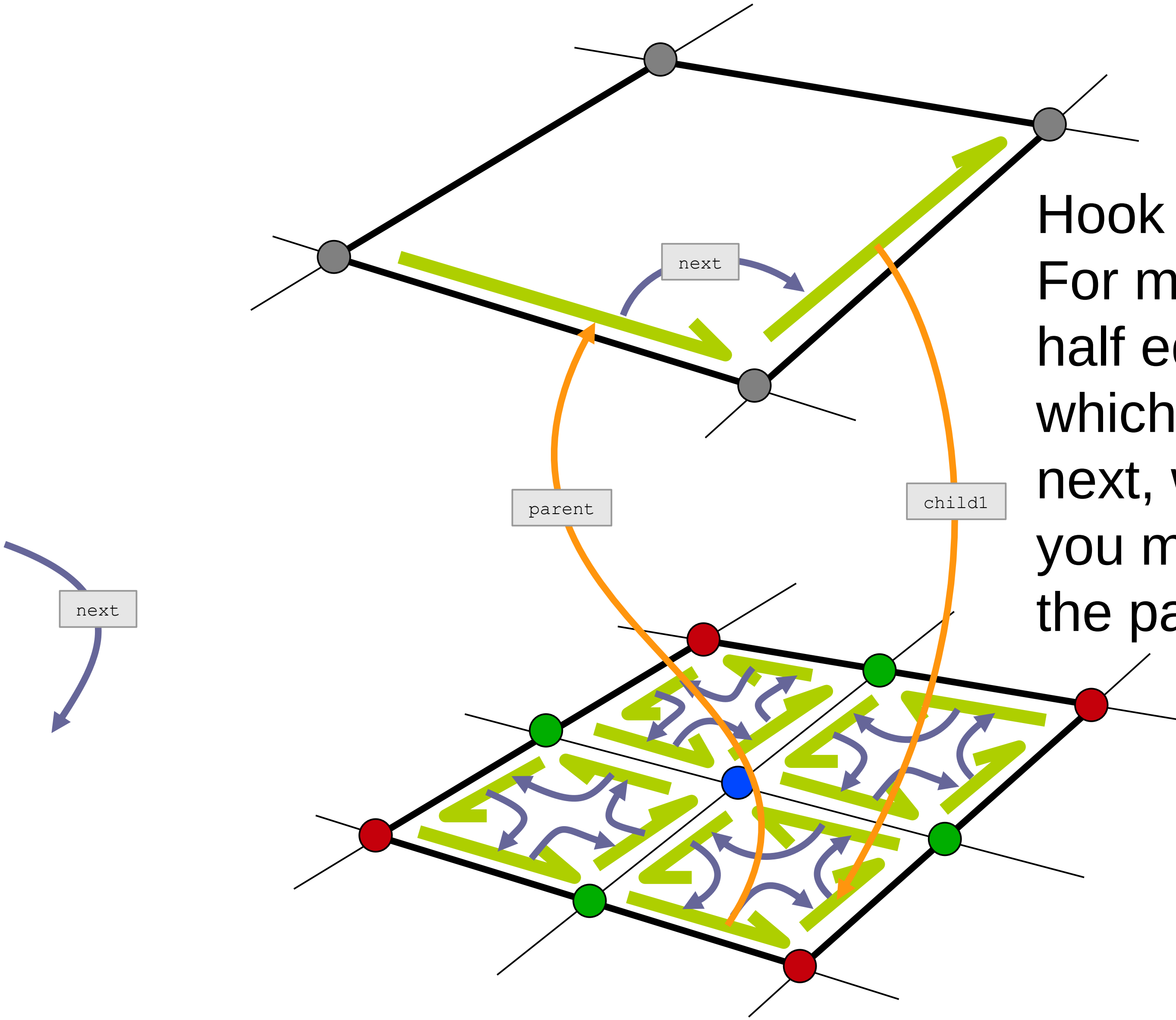




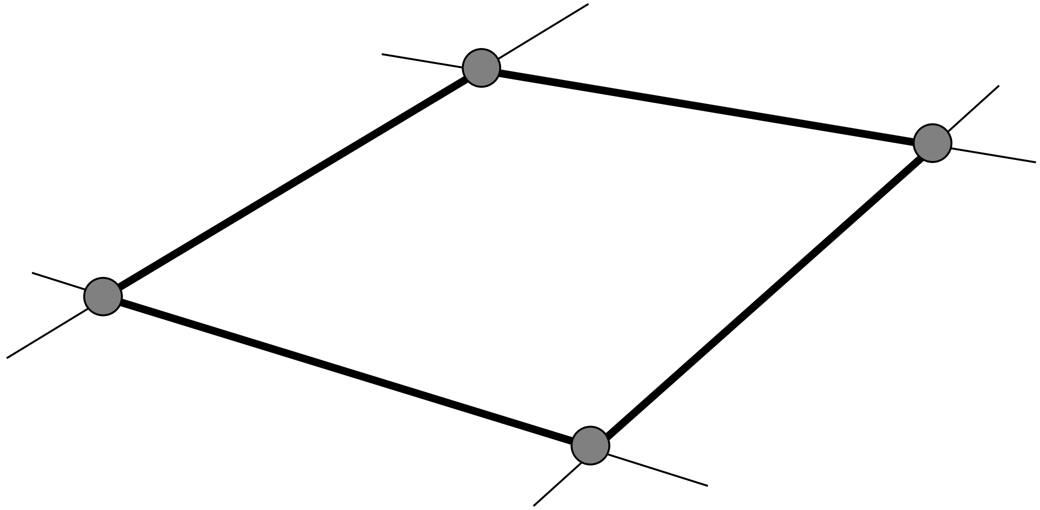
Twin pointers need to be set

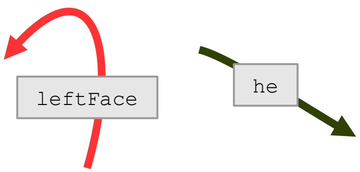


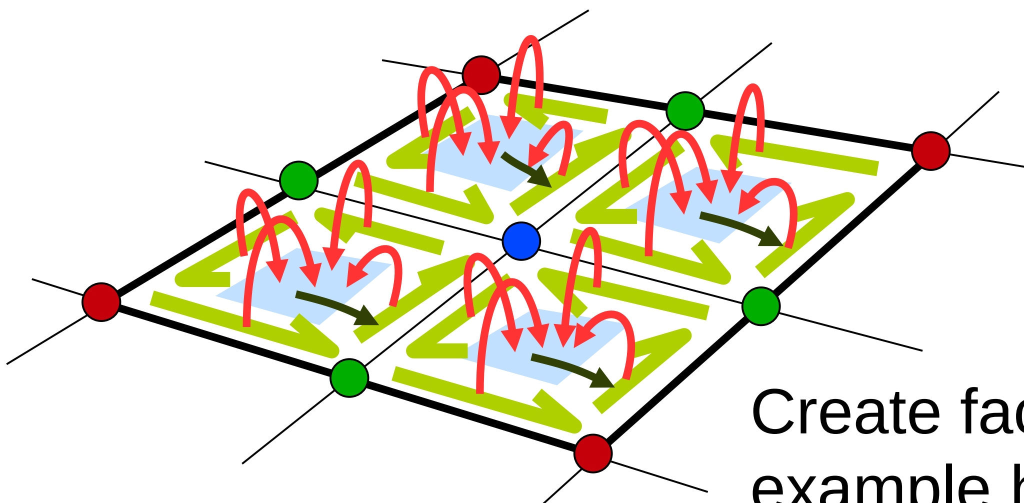




Hook up next pointers. For most of the new half edges you know which half edge is next, while for some you may need to use the parent pointers.

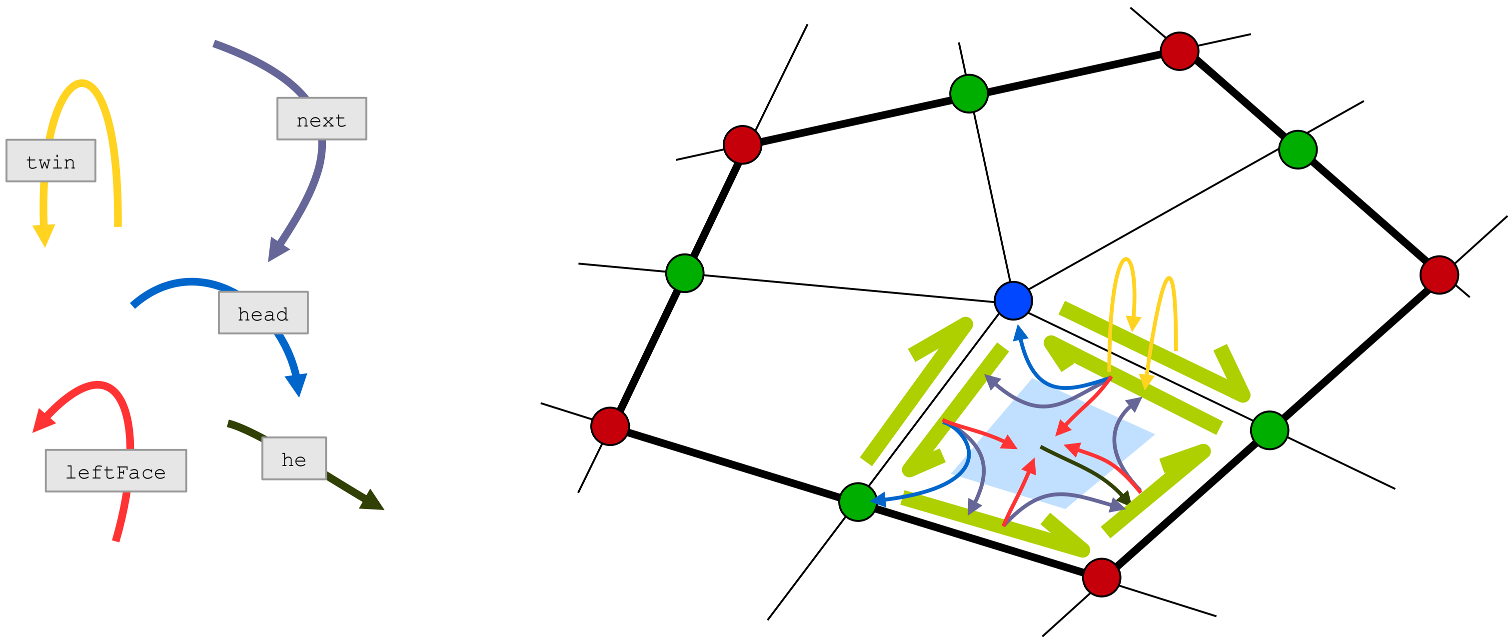






Create faces and set an example half edge for each. Also set all left face pointers of the half edge to point to the face

You find some way to do this in a loop to handle n-gons
(i.e., building the new faces, and setting all the necessary pointers).

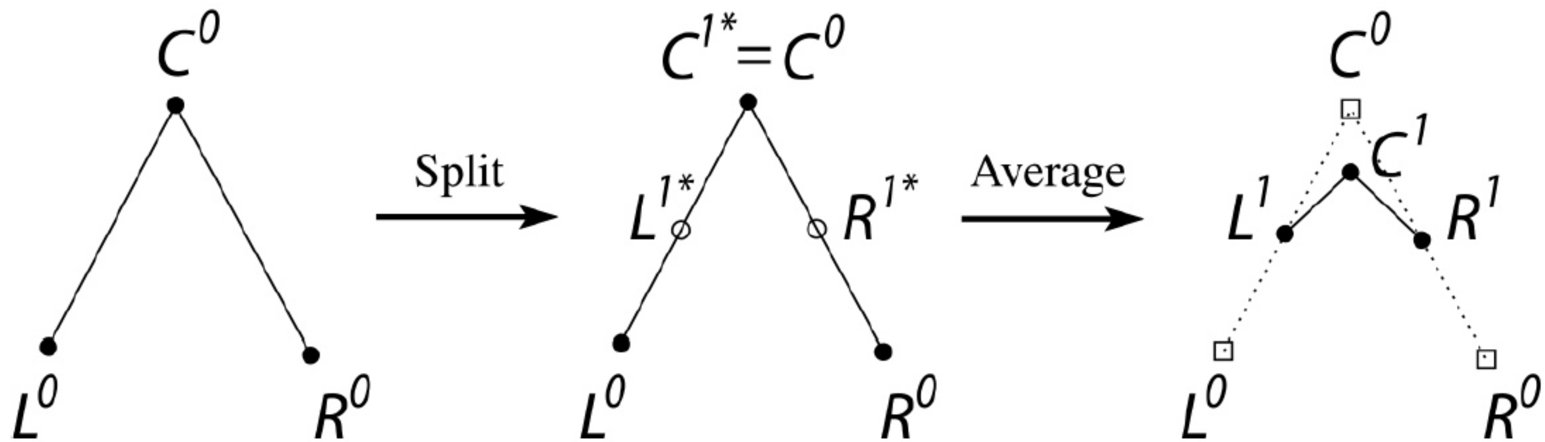


While there are many approaches, one way would be to set the13 pointers in each iteration.

Subdivision Analysis

(Lane-Risenfeld sheme with n = 2)

- Where does a point go to in the limit?
- What is the curve tangent (surface normal?)



Examine what happens in a local neighbourhood and write down the subdivision matrix.

$$\begin{pmatrix} L^j \\ C^j \\ R^j \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ \frac{1}{8} & \frac{6}{8} & \frac{1}{8} \\ 0 & \frac{1}{2} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} L^{j-1} \\ C^{j-1} \\ R^{j-1} \end{pmatrix}$$

$$\text{Let } X^j = (L^j, C^j, R^j)^T \qquad X^{j+1} = SX^j$$

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

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

$$= \begin{pmatrix} \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{2}} & -\frac{2}{3} \\ \frac{1}{\sqrt{3}} & 0 & \frac{1}{3} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & -\frac{2}{3} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{4} \end{pmatrix} \begin{pmatrix} \frac{1}{2\sqrt{3}} & \frac{2}{\sqrt{3}} & \frac{1}{2\sqrt{3}} \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{2} & 1 & -\frac{1}{2} \end{pmatrix}$$

$$S^{\infty} = V\Lambda^{\infty} V^{-1}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{3}} & * & * \\ \frac{1}{\sqrt{3}} & * & * \\ \frac{1}{\sqrt{3}} & * & * \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{2\sqrt{3}} & \frac{2}{\sqrt{3}} & \frac{1}{2\sqrt{3}} \\ * & * & * \\ * & * & * \end{pmatrix}$$

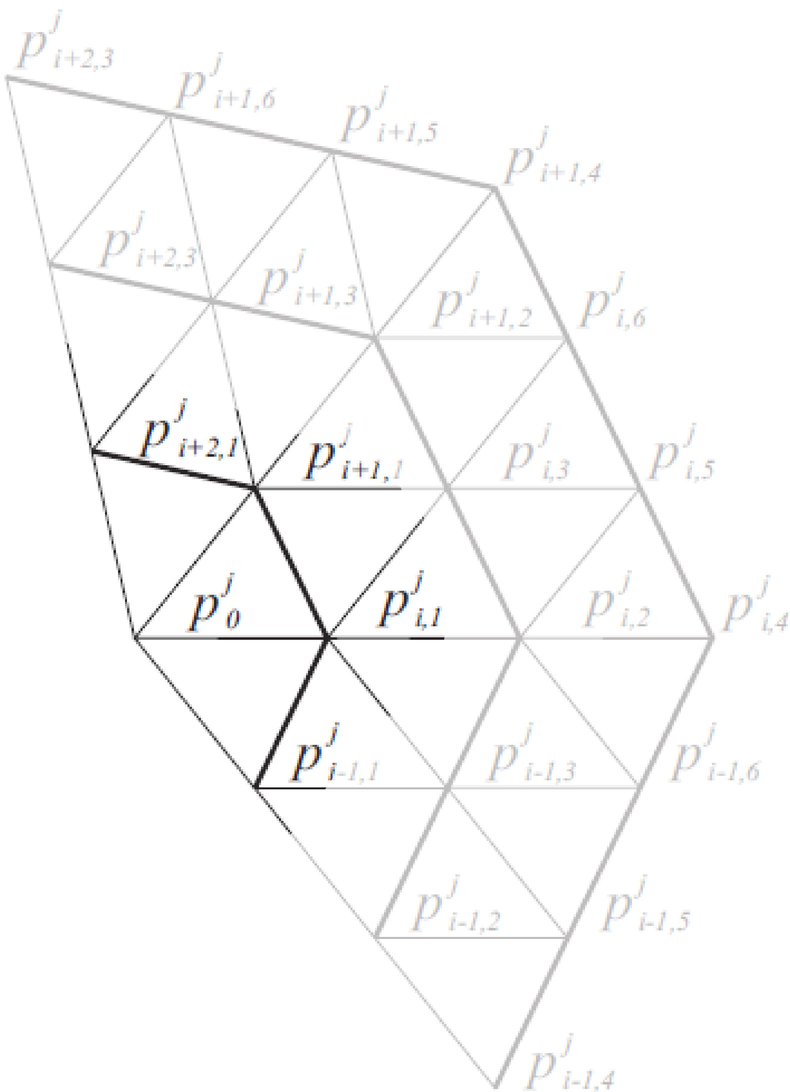
$$C^{\infty} = \begin{pmatrix} \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{pmatrix} \begin{pmatrix} L^0 \\ C^0 \\ R^0 \end{pmatrix}$$

Tangent Analysis

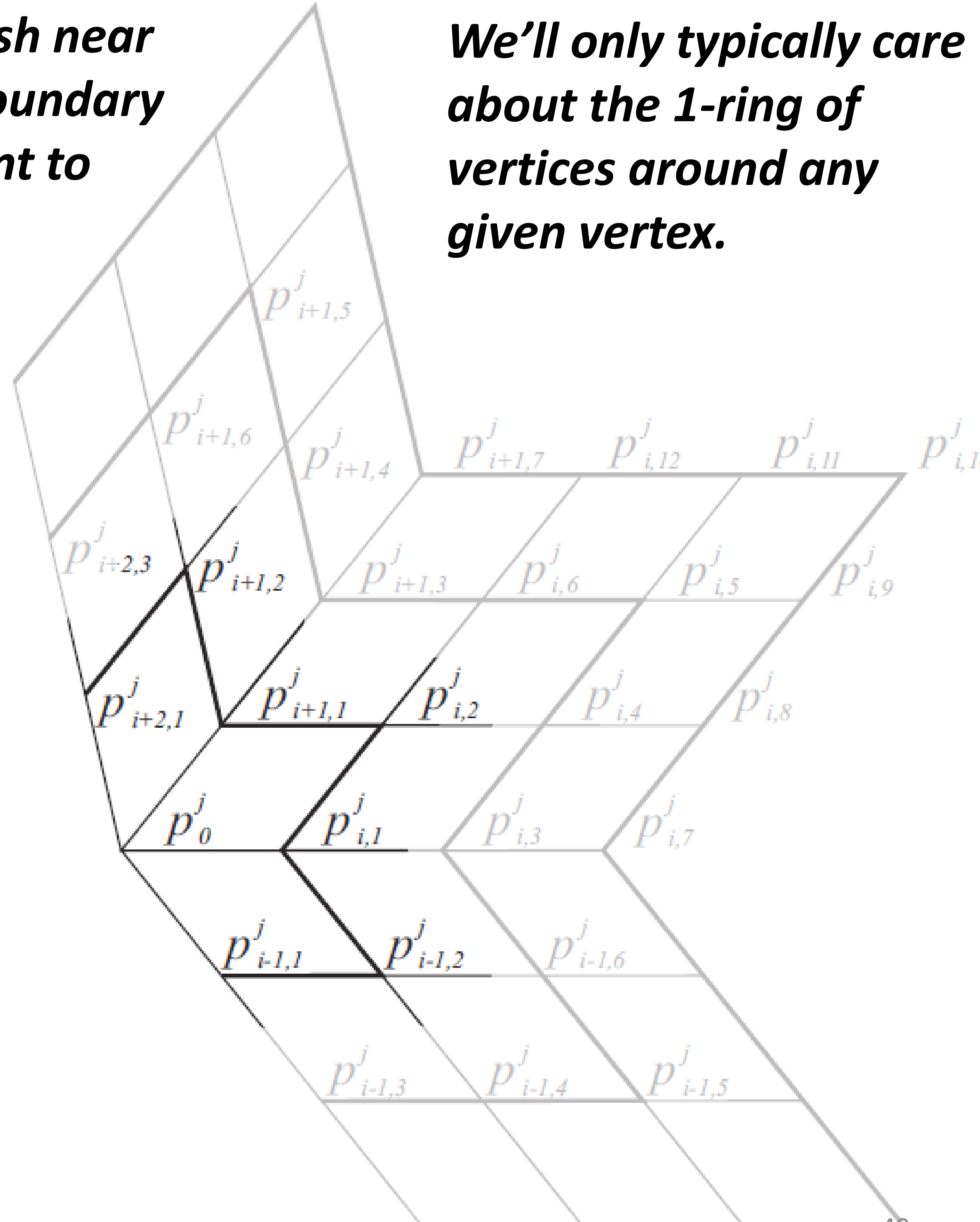
$$T^{\infty} = \lim_{j \to \infty} \frac{R^j - C^j}{||R^j - C^j||}$$

- Can also look at right minus left (easier)

Enumeration of vertices of a mesh near an extraordinary vertex; for a boundary vertex, the 0–th sector is adjacent to the boundary.

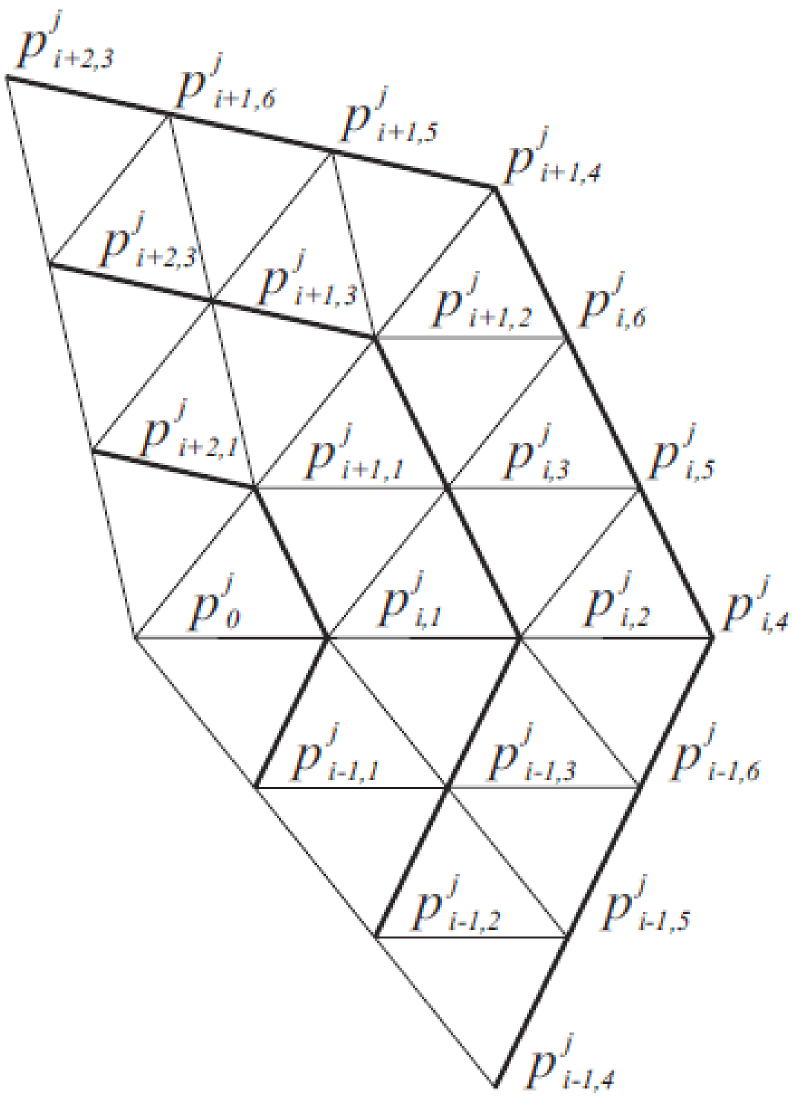


We'll only typically care about the 1-ring of vertices around any given vertex.



Loop Surface Tangent Vectors

(see page 72 of subdivision notes)



The rules for computing tangent vectors for the Loop scheme are especially simple at an interior vertex,

$$t_1 = \sum_{i=0}^{k-1} \cos \frac{2\pi i}{k} p_{i,1}$$

$$t_2 = \sum_{i=0}^{k-1} \sin \frac{2\pi i}{k} p_{i,1}.$$

Loop Boundary Tangent Vectors

(see page 71 of subdivision notes)

At a boundary vertex, tangents are computed ***along*** and ***across*** the boundary (same for creases)

$$t_{along} = p_{0,1} - p_{k-1,1}$$

$$t_{across} = p_{0,1} + p_{1,1} - 2p_0 \quad \text{for } k = 2$$

$$t_{across} = p_{2,1} - p_0 \quad \text{for } k = 3$$

TYPO!

$$t_{across} = \sin\theta\,(p_{0,1} + p_{k-1,1}) + (2\cos\theta - 2)\sum_{i=1}^{k-2} \sin i\theta\, p_{i,1} \quad \text{for } k \geq 4$$

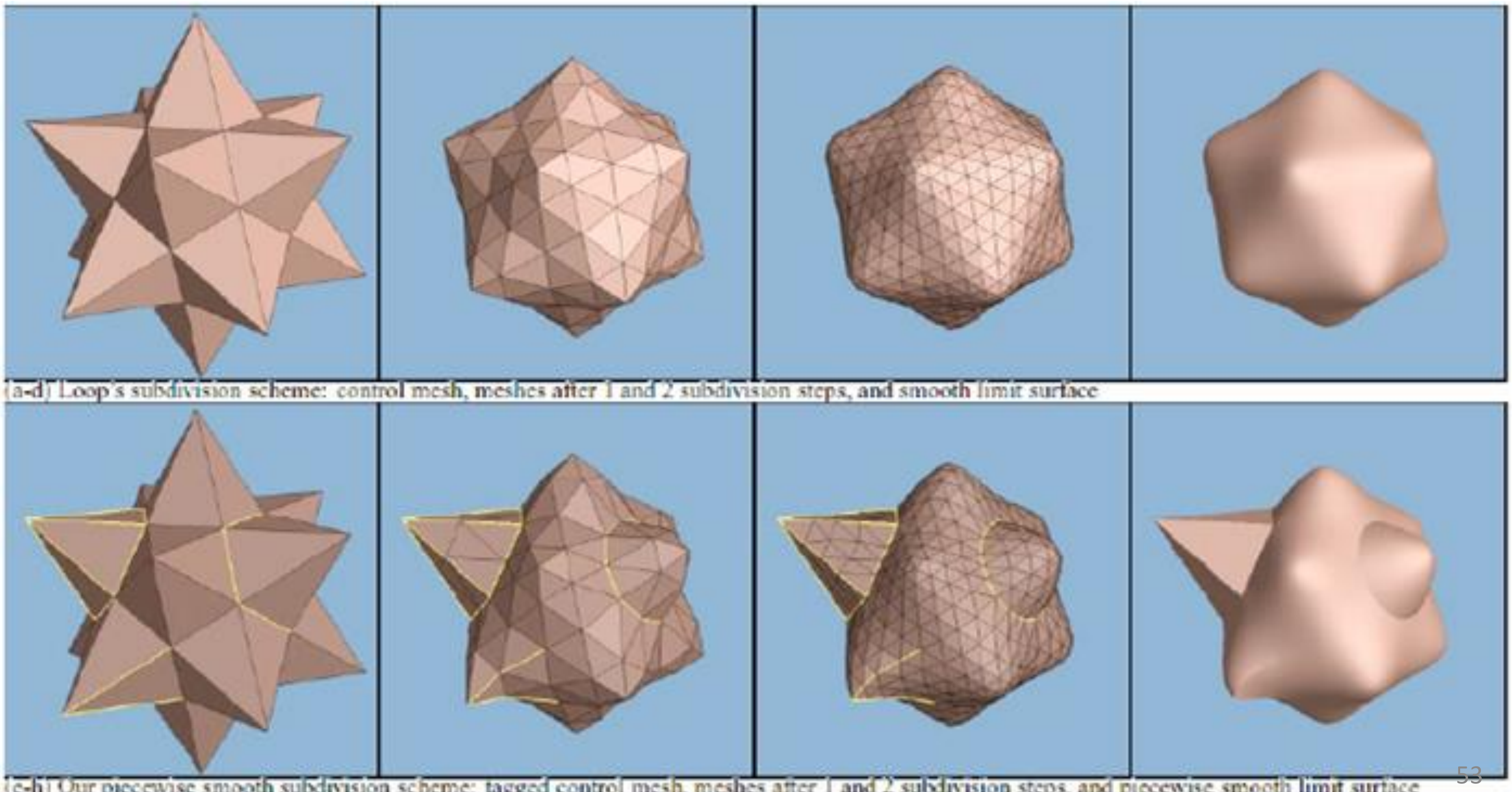
Catmull-Clark tangents computed with the same formulas!

Relationship to Splines (aside)

- In regular regions, behaviour is identical
- At extraordinary vertices, achieve C1 continuity
  - Behaviour near extraordinary vertices is different from splines
- Linear everywhere
  - Mapping from parameter space to 3D is a linear combination of the control points
  - “emergent” basis functions per control point
    - Match the splines in regular regions
    - “custom” basis functions around extraordinary vertices
- Parametric evaluation possible through eigen-analysis of subdivision rules.

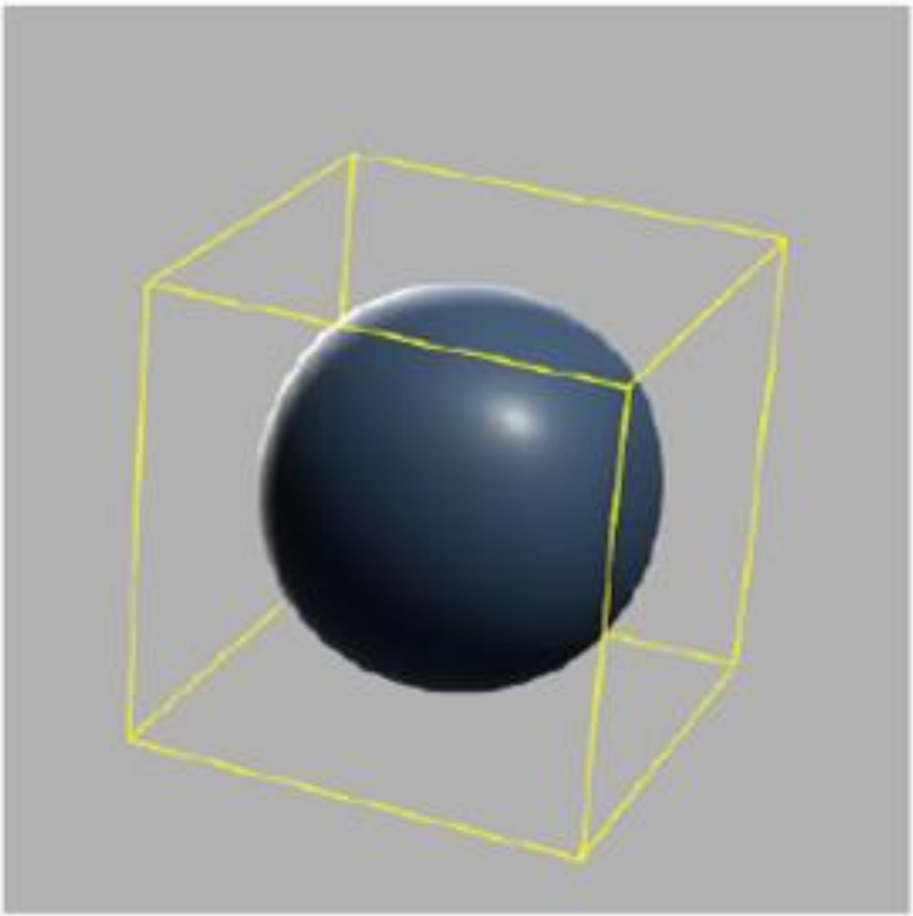
Loop with Creases

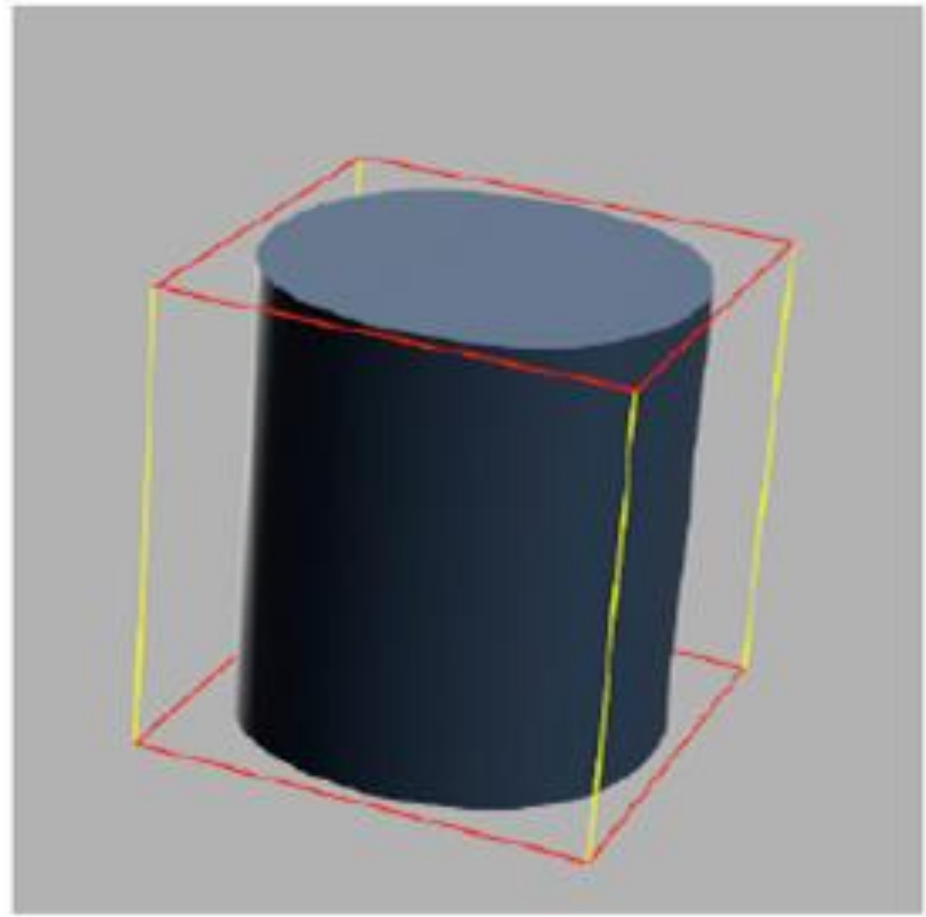
Treating interior edges as a boundary results in creases. Edges can also be marked as partially creased (treat as boundary for only a finite number of subdivisions).

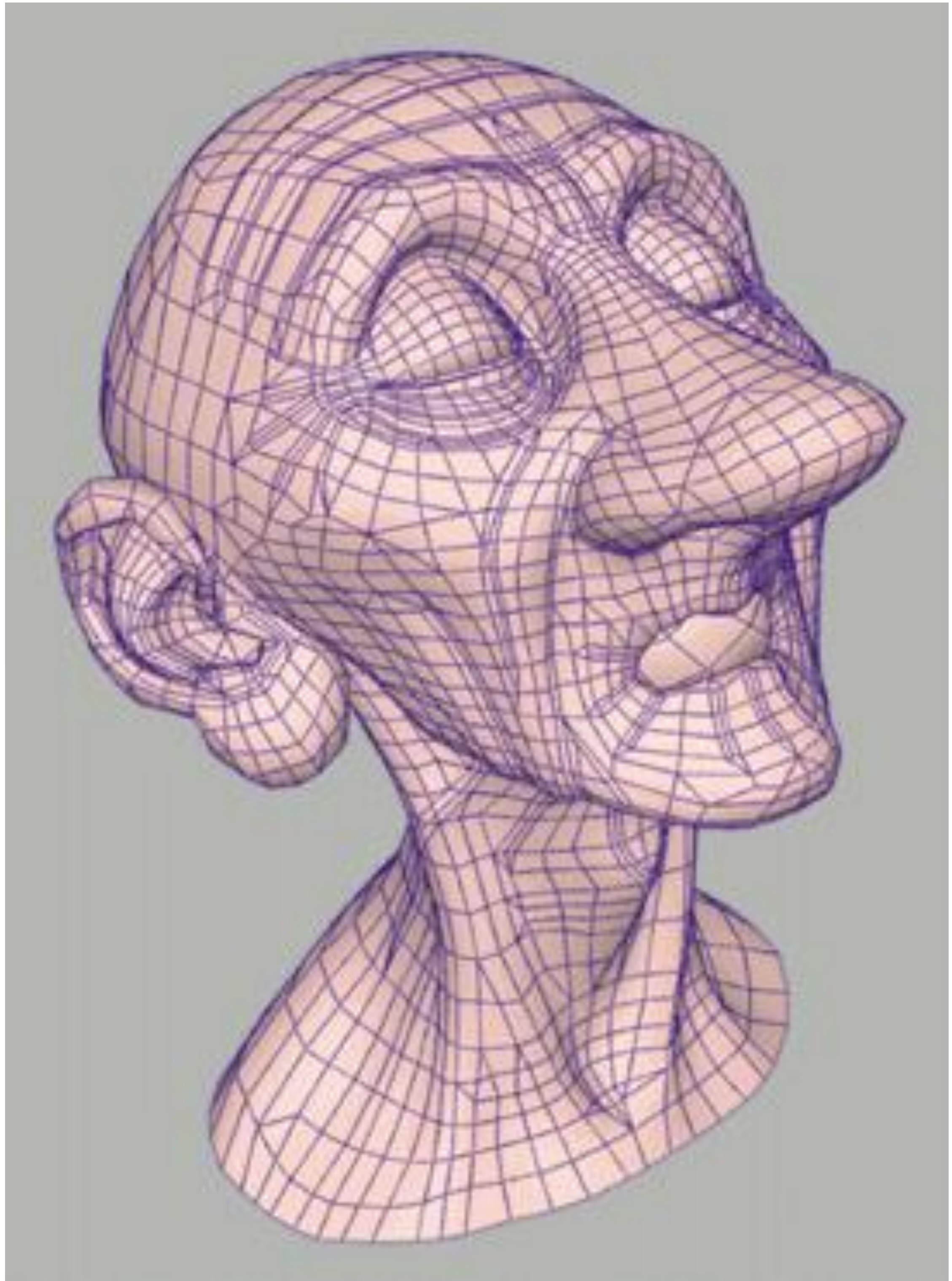


[Huges Hoppe]

Catmull-Clark with creases















Geri's Game

- Pixar short film to test subdivision in production
  - Catmull-Clark (quad mesh) surfaces
  - Complex geometry
  - Extensive use of creases
  - Subdivision surfaces to support cloth dynamics