# Subdivision Curves & Surfaces

*Bridge the gap between discrete surfaces (polygonal meshes) and continuous surfaces (e.g. collection of spline patches)*

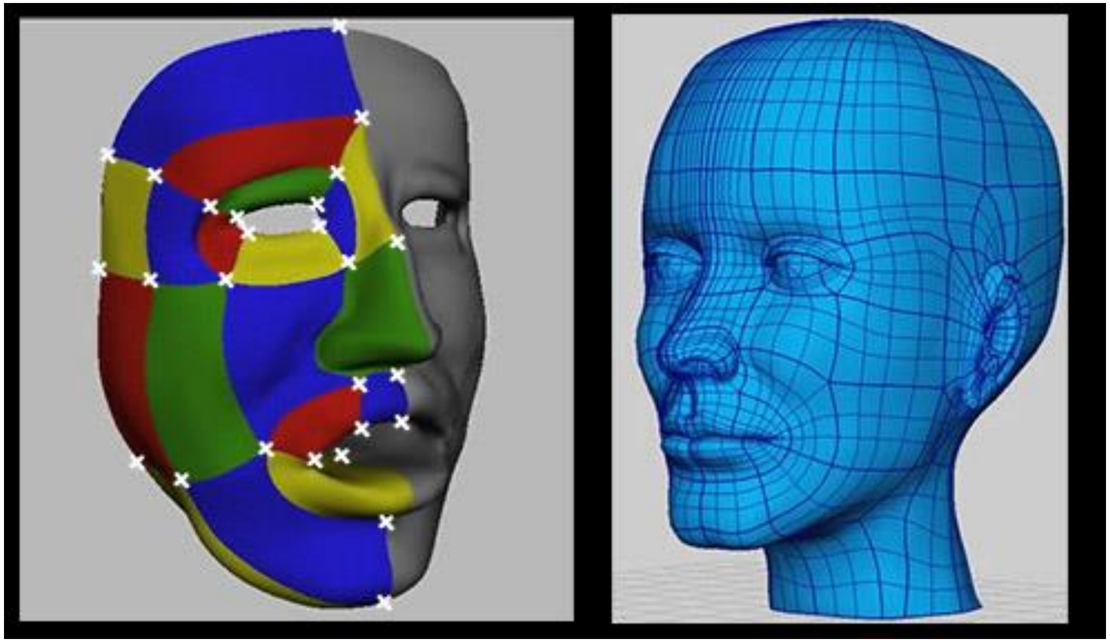

Geri’s Game (1989) : Pixar Animation Studios
http://mrl.nyu.edu/~dzorin/sig99/derose/index.htm

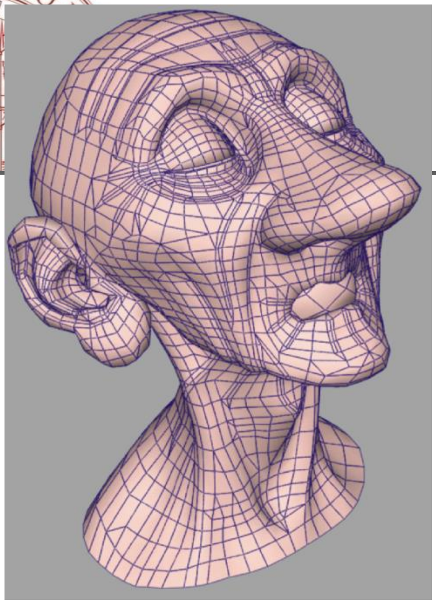
# Sometimes need more than polygon meshes…

- Traditional geometric modeling used NURBS
- Problems with NURBS
  - A single NURBS patch has quadrilateral topology

Must use many NURBS patches to model complex geometry

When deforming a surface made of NURBS patches, cracks arise at the seams





- Traditionally spline patches (NURBS) have been used in production for character animation.

- Difficult to control spline patch density in character modelling.

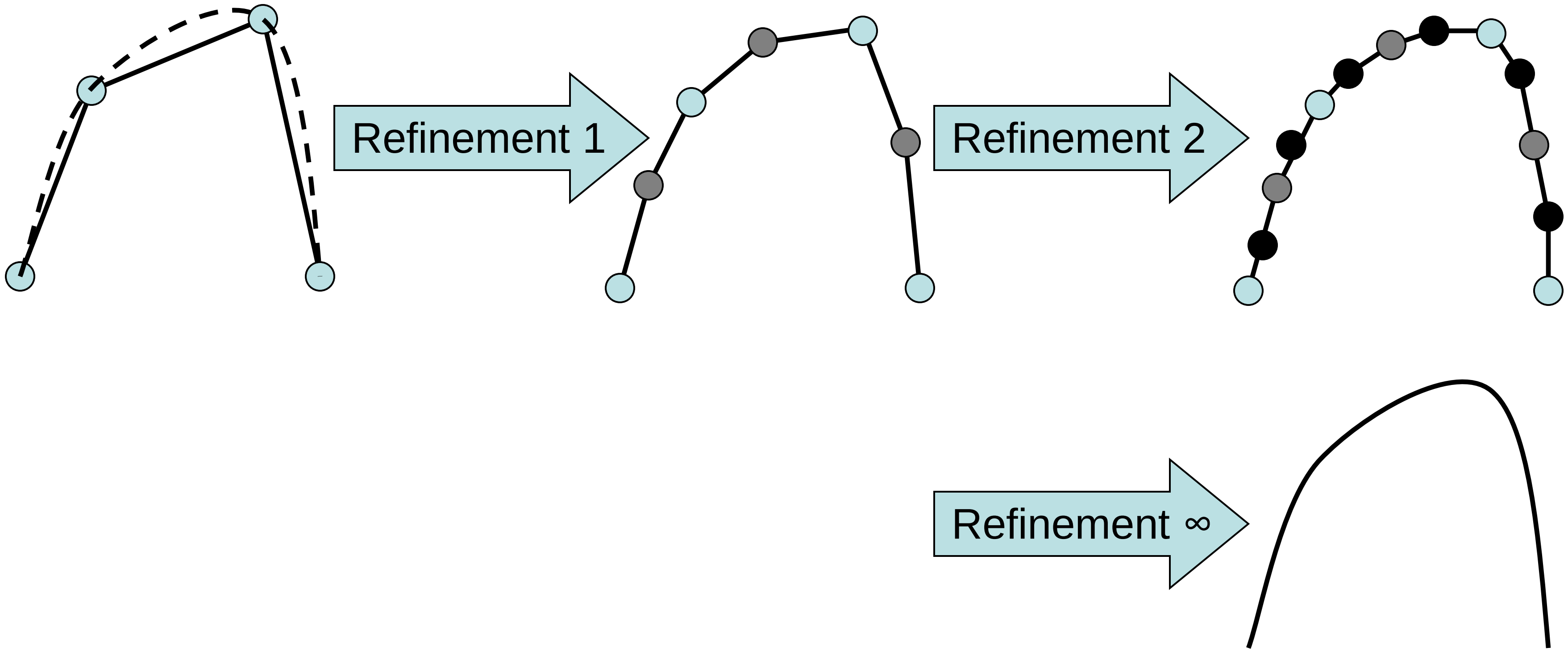
**Subdivision in Character Animation**
Tony Derose, Michael Kass, Tien Troung
(SIGGRAPH ’98)

(Geri’s Game, Pixar 1998)


(c) Disney/Pixar

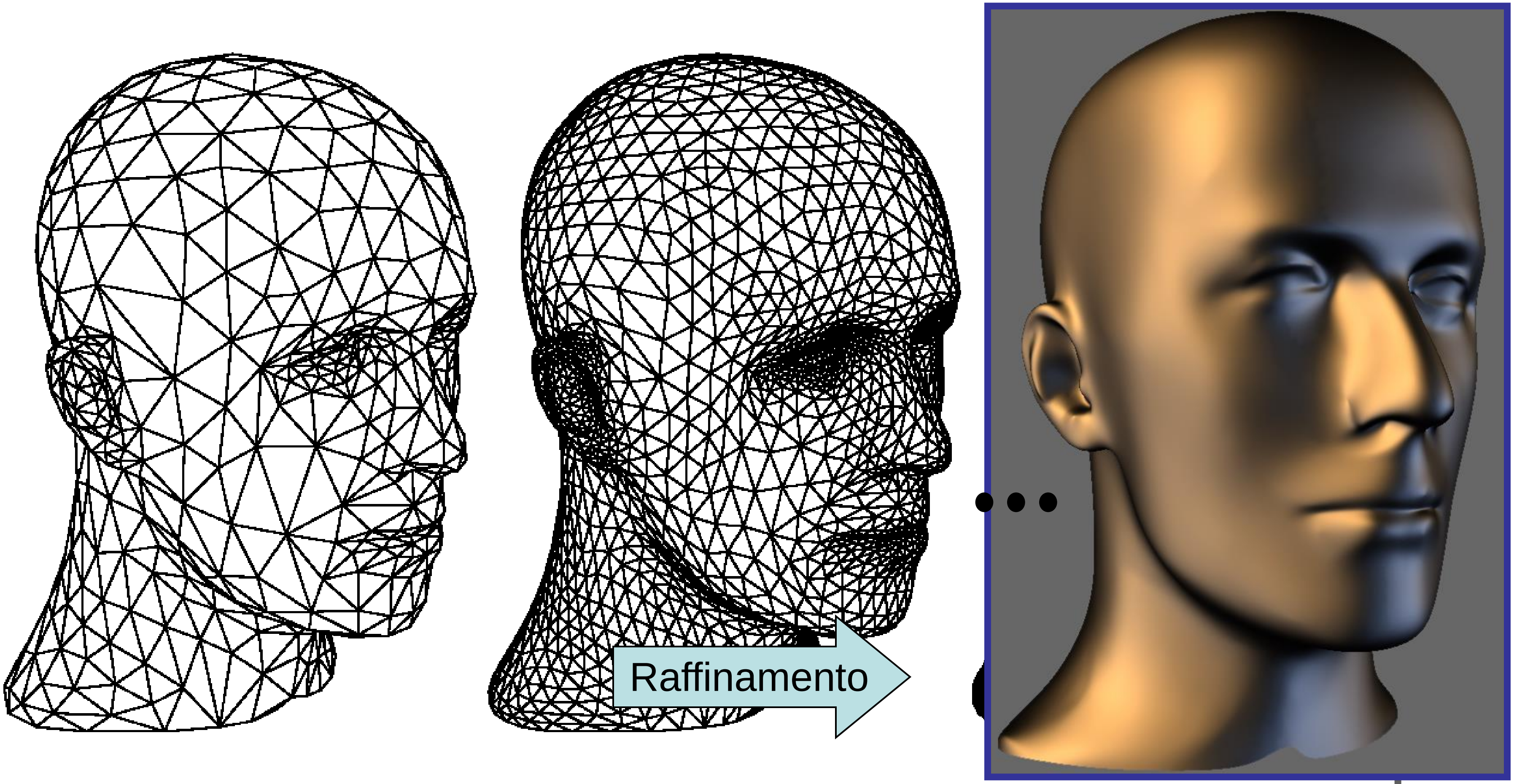
# Subdivision Curves

- Bézier curves, spline e subdivision are based on an algorithm which takes a control polygon in input and constructs a smooth curve.
- Approach Limit Curve through an **Iterative Refinement Process**.

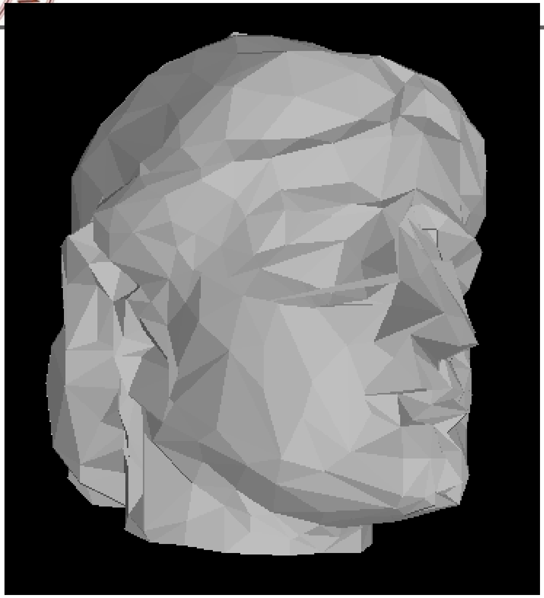


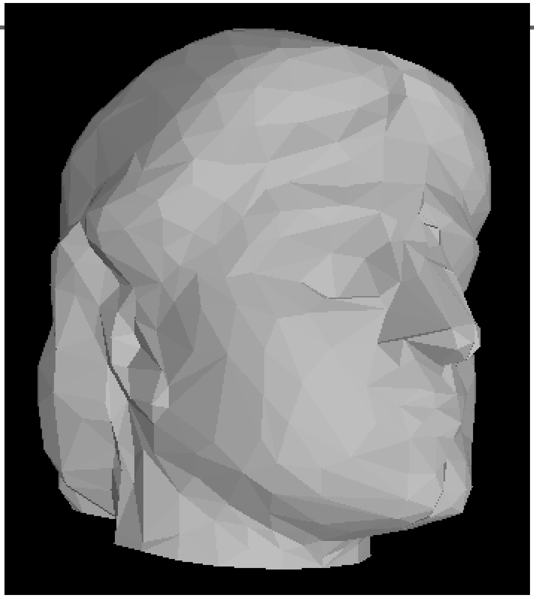
# Subdivision surfaces

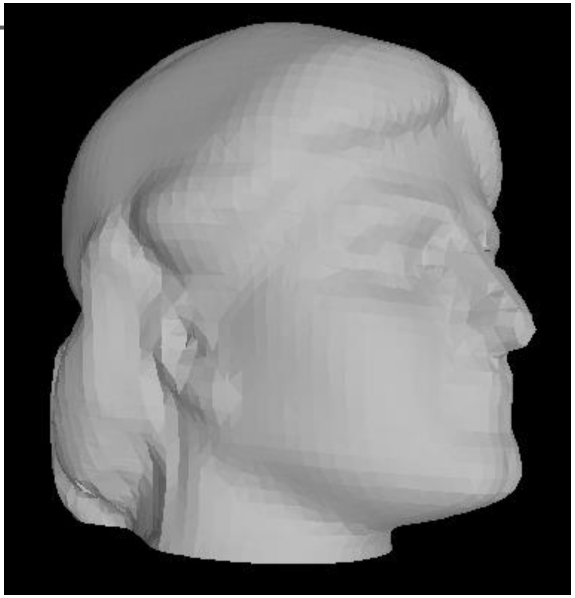
- Same approach works in 3D



# Example

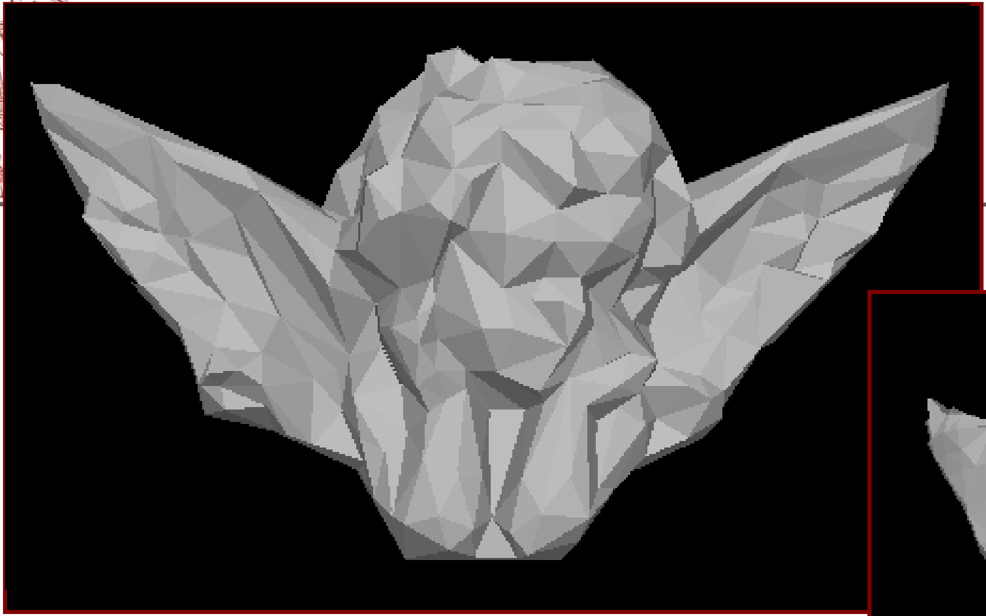


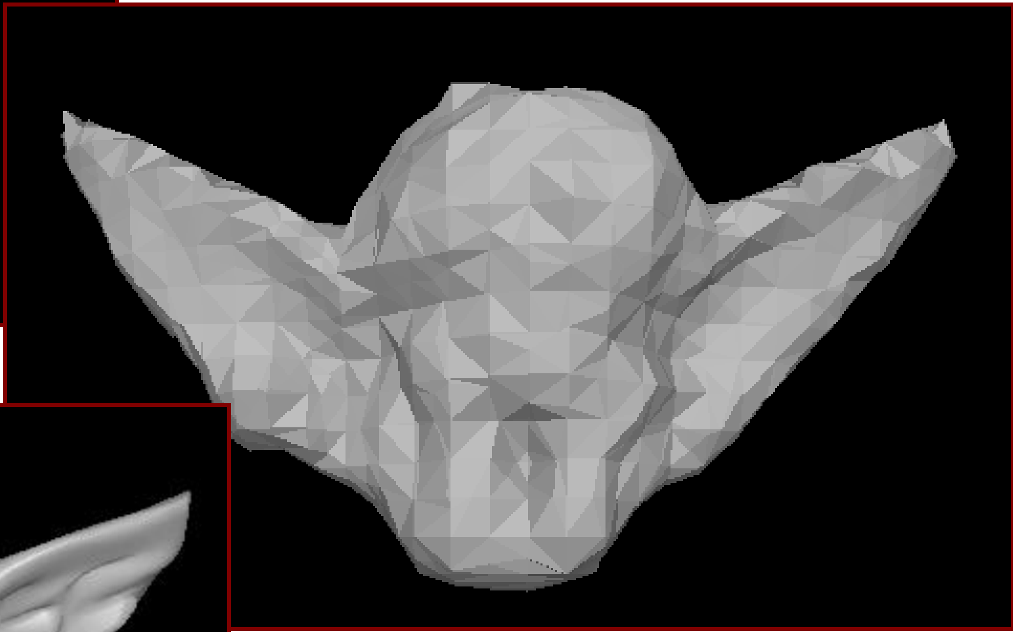


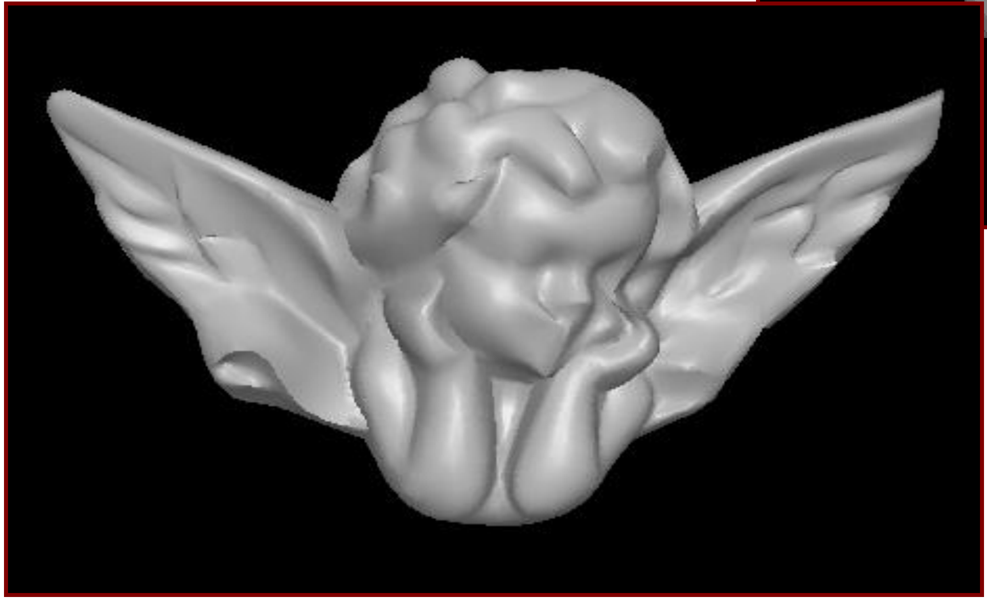


8 RANGE IMAGE, point cloud 98503

# Example

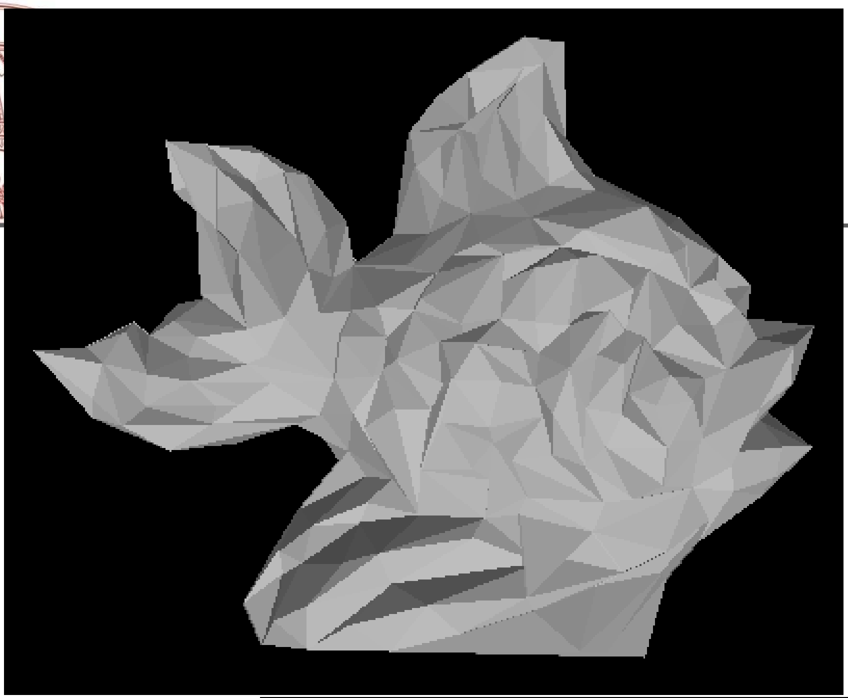


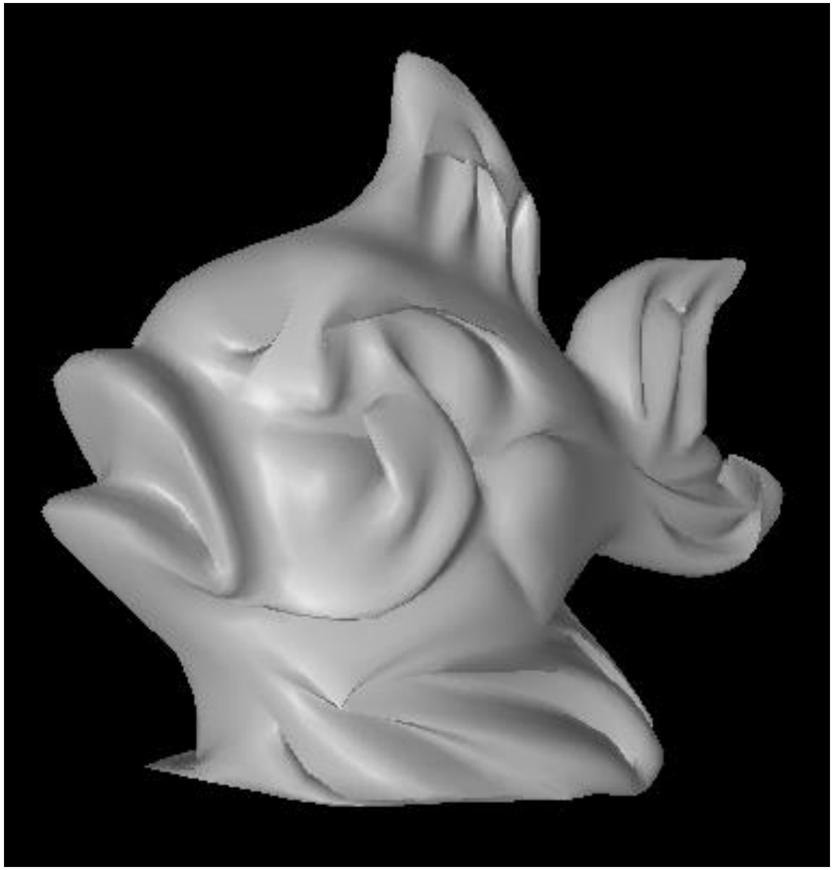




1 RANGE IMAGE, point cloud 13903

# Example





2 RANGE IMAGE, point coud 13166

# Goals of Subdivision Surfaces

- Represent arbitrary topology surfaces
- How do we represent curved surfaces in the computer?
  - Efficiency of Representation
  - Continuity
  - Affine Invariance
  - Efficiency of Rendering
- How do they relate to splines/patches?
- Why use subdivision rather than patches?

# Types of Subdivision

- Interpolating Schemes
  - Limit Surfaces/Curve will pass through original set of data points.
- Approximating Schemes
  - Limit Surface will not necessarily pass through the original set of data points.

# Refinement scheme

A refinement process defines a sequence of control polygons

$$P_0, P_1, \dots, P_{n1}$$
$$P_0^1, P_1^1, \dots, P_{n2}^1$$
$$P_0^2, P_1^2, \dots, P_{n3}^2$$
$$\dots\dots$$
$$P_0^k, P_1^k, \dots, P_{nk}^k$$

Where for each k each control point is given by

Linear combination of the control points $\left\{P_0^{k-1}, P_1^{k-1}, \dots, P_{n_k-1}^{k-1}\right\}$ of the control polygon at the previous step

$$P_j^k = \sum_{i=0}^{n_k-1} \alpha_{i,j,k} P_i^{k-1}$$

# Refinement scheme

Mask: $$\alpha_{i,j,k}, \quad \forall i, j, k$$

- The number of CP can be either increased (eg. Chaikin’s curve) or decreased (eg. de Casteljau for Bézier curves)

- Uniform Scheme:

  the alfa values are independent on the refinement level k

- Stationary Scheme: the mask is the same for each CP

# Subdivision as Matrices

$$P_j^k = \sum_{i=0}^{n_k-1} \alpha_{i,j,k} P_i^{k-1}$$

For each control point j=0,..,$n_k$:

**New CP**

$$P_j^k = \begin{bmatrix} \alpha_{0,j,k} & \alpha_{1,j,k} & \cdots & \alpha_{n_k-1,j,k} \end{bmatrix} \begin{bmatrix} P_0^{k-1} \\ P_1^{k-1} \\ \cdots \\ P_{n_k}^{k-1} \end{bmatrix}$$

**Old CP**

$$\begin{bmatrix} P_0^k \\ P_1^k \\ \cdots \\ P_{n_k}^k \end{bmatrix} = S_k \begin{bmatrix} P_0^{k-1} \\ P_1^{k-1} \\ \cdots \\ P_{n_k}^{k-1} \end{bmatrix} \qquad S_k = \begin{bmatrix} \alpha_{0.0.k} & \alpha_{1.0.k} & \cdots & \alpha_{n_k-1.0.k} \\ \alpha_{0.1.k} & \alpha_{1.1.k} & \cdots & \alpha_{n_k-1.1.k} \\ \cdots & \cdots & \cdots & \cdots \\ \alpha_{0.n_k.k} & \alpha_{1.n_k.k} & \cdots & \alpha_{n_k-1.n_k.k} \end{bmatrix}$$

*$S_{mask}$ refinement matrix*

# Subdivision as Matrices

- Subdivision can be expressed as a matrix $S_{mask}$ of weights *w*.
  - *$S_{mask}$ is very sparse*
  - *Never Implement this way!*
  - Allows for analysis
    - Curvature
    - Limit Surface

$$P^k = S_k P^{k-1}$$

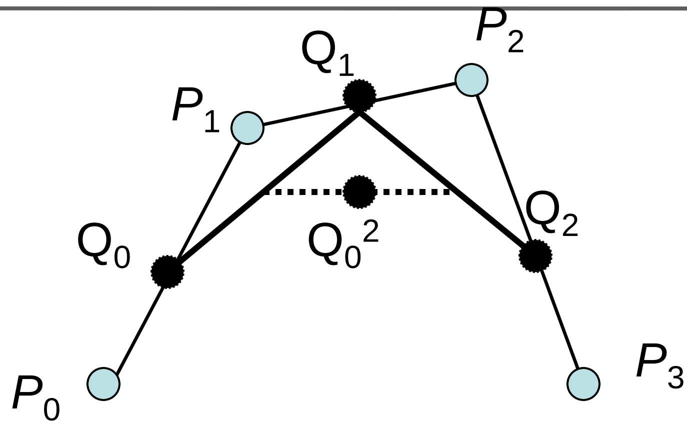
$$\begin{bmatrix} P_0^k \\ P_1^k \\ \dots \\ P_{n_k}^k \end{bmatrix} = \begin{bmatrix} \alpha_{0.0.k} & \alpha_{1.0.k} & \dots & \alpha_{n_k-1.0.k} \\ \alpha_{0.1.k} & \alpha_{1.1.k} & \dots & \alpha_{n_k-1.1.k} \\ \dots & \dots & \dots & \dots \\ \alpha_{0.n_k.k} & \alpha_{1.n_k.k} & \dots & \alpha_{n_k-1.n_k.k} \end{bmatrix} \begin{bmatrix} P_0^{k-1} \\ P_1^{k-1} \\ \dots \\ P_{n_k}^{k-1} \end{bmatrix}$$

New Points

$S_{mask}$ Weights

Old Control Points

# Example: de Casteljau's algorithm




$$Q_0 = \frac{1}{2}P_0^0 + \frac{1}{2}P_1^0 \qquad Q_1 = \frac{1}{2}P_1^0 + \frac{1}{2}P_2^0$$

$$Q_2 = \frac{1}{2}P_2^0 + \frac{1}{2}P_3^0$$

$$Q_0^1 = \frac{1}{2}Q_1^0 + \frac{1}{2}Q_2^0 \qquad Q_1^1 = \frac{1}{2}Q_1^0 + \frac{1}{2}Q_2^0$$

$$Q_0^2 = \frac{1}{2}Q_0^1 + \frac{1}{2}Q_1^1$$

Refinement scheme

Each new CP is the average of the edge
Between two points of the old control polygon

In general:

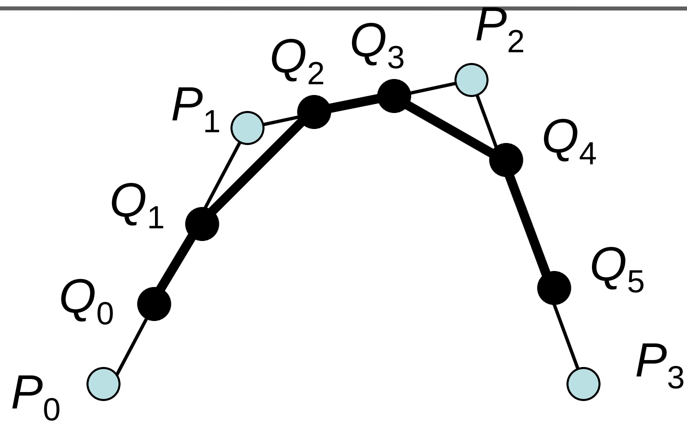
$$P_j^k = \frac{1}{2}P_j^{k-1} + \frac{1}{2}P_{j+1}^{k-1}$$

$$k = 0,1,2,..,n-1, \; 0 \le j \le n-k$$

Old control polygon has n+1 CP
New control polygon has n CP.
The final poly has 1 point

*Uniform – Stationary*

# Chaiken’s Algorithm (1974)




$$Q_0 = \frac{1}{4}P_0 + \frac{3}{4}P_1$$

$$Q_1 = \frac{3}{4}P_0 + \frac{1}{4}P_1$$

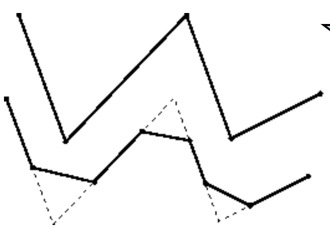
$$Q_2 = \frac{1}{4}P_1 + \frac{3}{4}P_2$$

$$Q_3 = \frac{3}{4}P_1 + \frac{1}{4}P_2$$

$$Q_4 = \frac{1}{4}P_2 + \frac{3}{4}P_3$$

$$Q_5 = \frac{3}{4}P_2 + \frac{1}{4}P_3$$

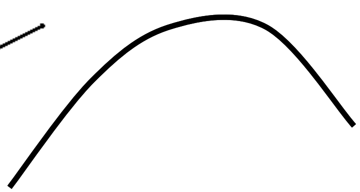
Apply Iterated Refinement scheme



$$Q_{2i} = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}$$

$$Q_{2i+1} = \frac{3}{4}P_i + \frac{1}{4}P_{i+1}$$

Old control poly with n+1 CP
New control poly with 2n CP.

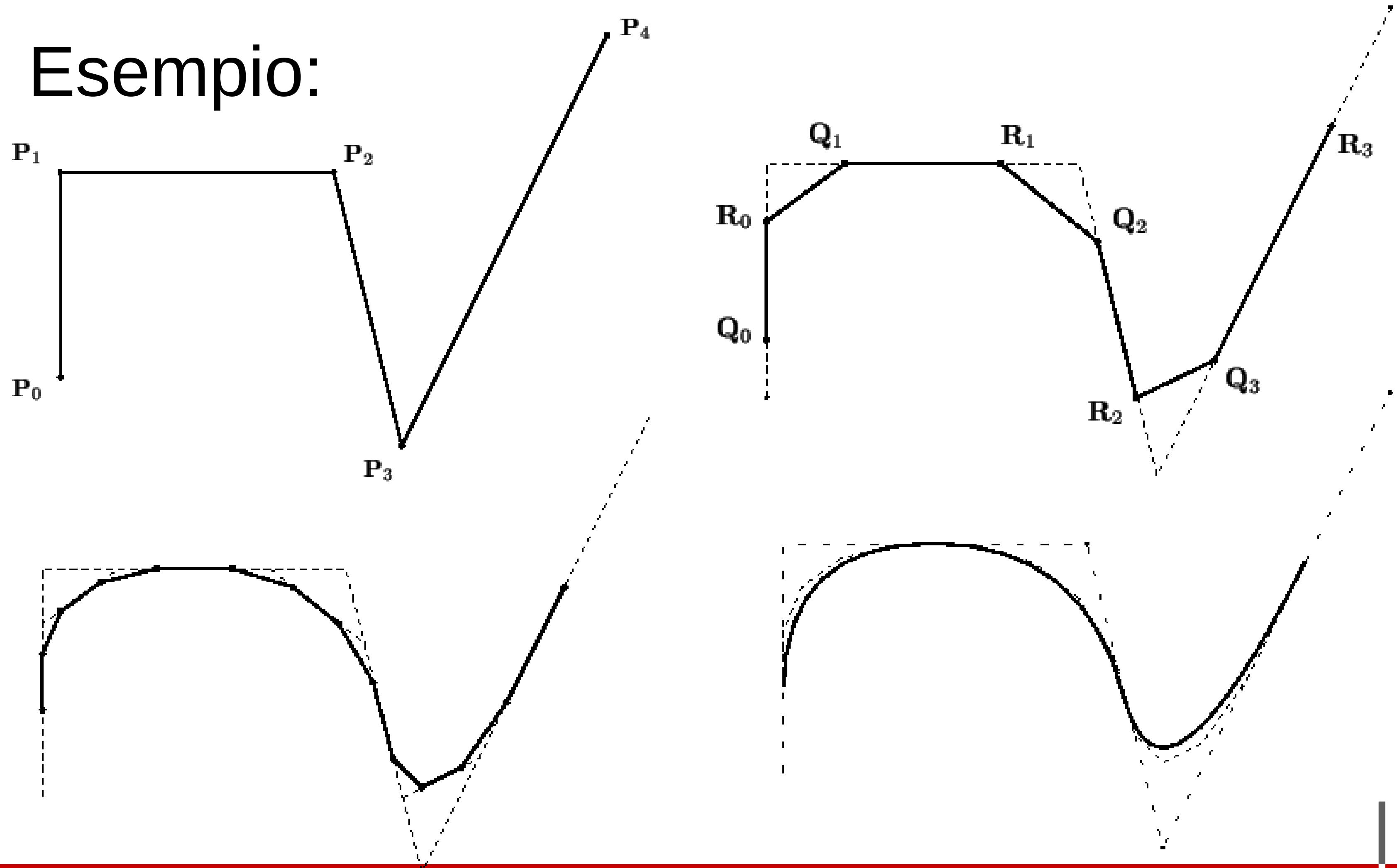


Limit Curve Surface

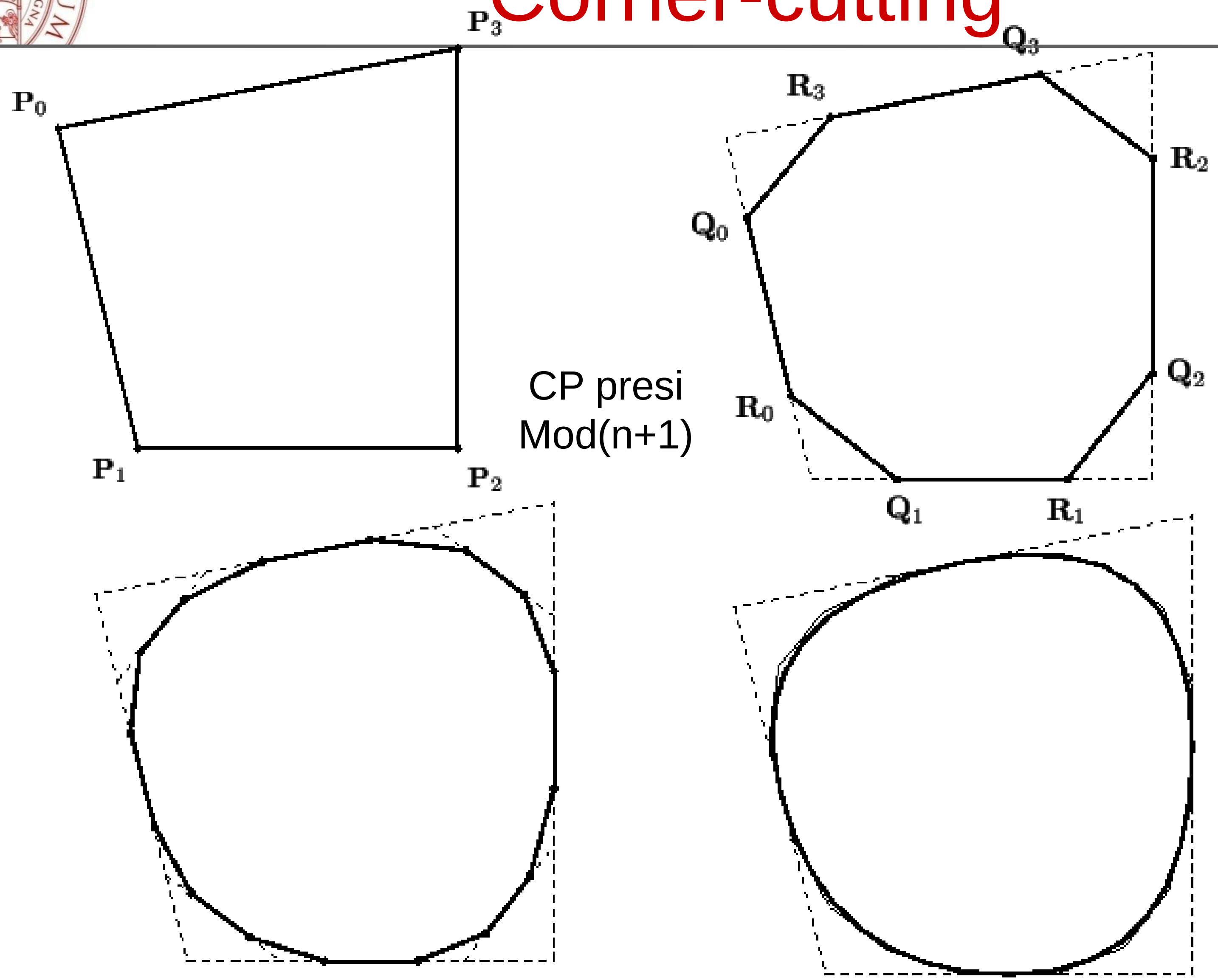
***Uniform –Non stationary***

# Chaiken's Algorithm : Corner-cutting

Esempio:



# Chaiken's Algorithm : Corner-cutting




CP presi Mod(n+1)

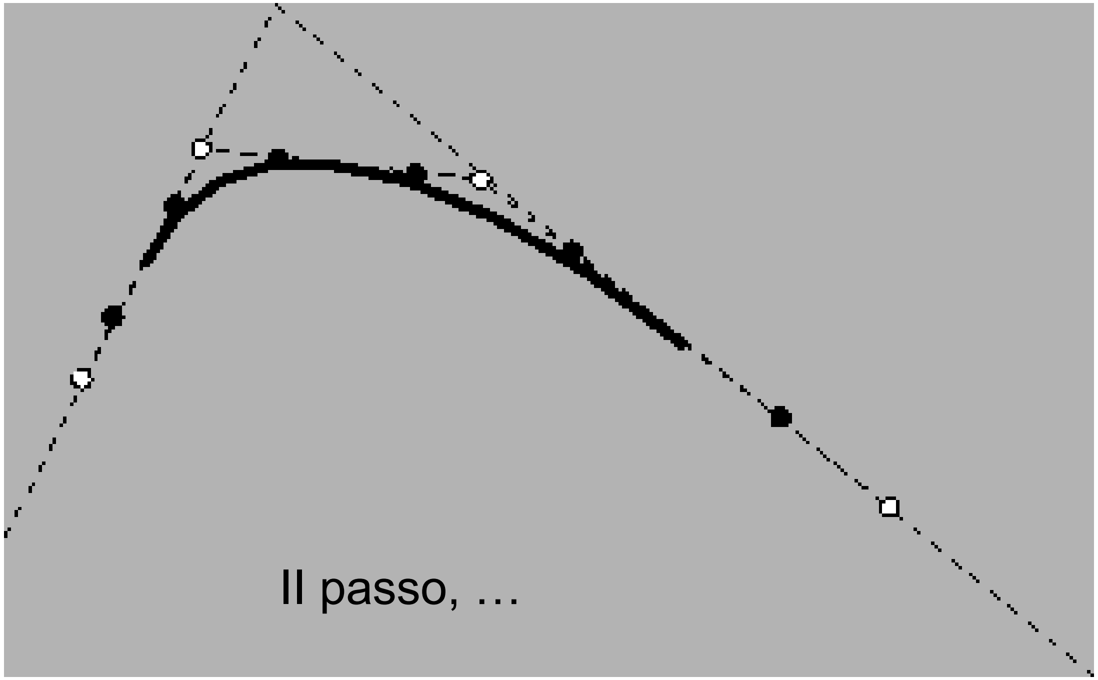
# Subdivision curves/surfaces

- Convergence: given a subdivision operator and a control polygon, does the refinement process converge?

- Continuity: the refinement process converges to a continuous curve/surface? Which continuity order?

# Convergence to a quadratic uniform spline curve

- The curve obtained by Chaikin ‘s subdivision scheme is a unform, quadratic spline




At the limit, the refined CPs converge to the spline curve

# Subdivision scheme for surfaces

- INPUT:

  control mesh of vertices, edges, faces.
- ITERATE SUBDIVISION OPERATOR:

  refine the control mesh by increasing the number of vertices
  - **Refine** the mesh
  - **Smooth** the mesh moving vertices

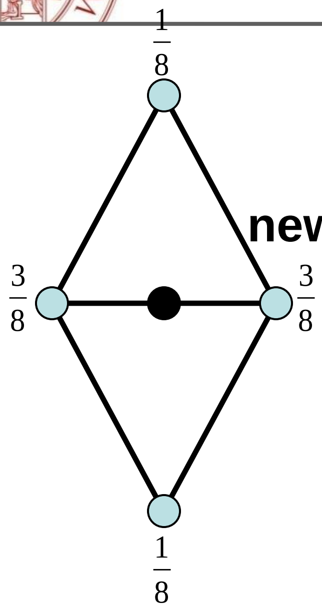
- At limit, the vertices of the control mesh converge to a limit smooth surface

# Loop Subdivisions

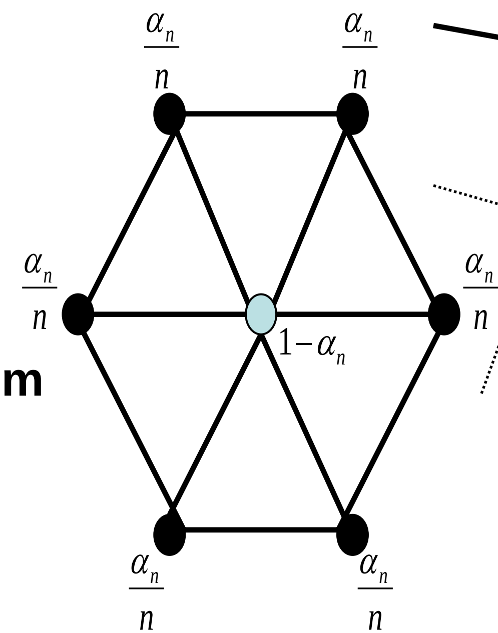
- Works on triangular meshes
- Is an Approximating Scheme
- Guaranteed to be smooth everywhere except at ***extraordinary*** vertices (valence ≠6).

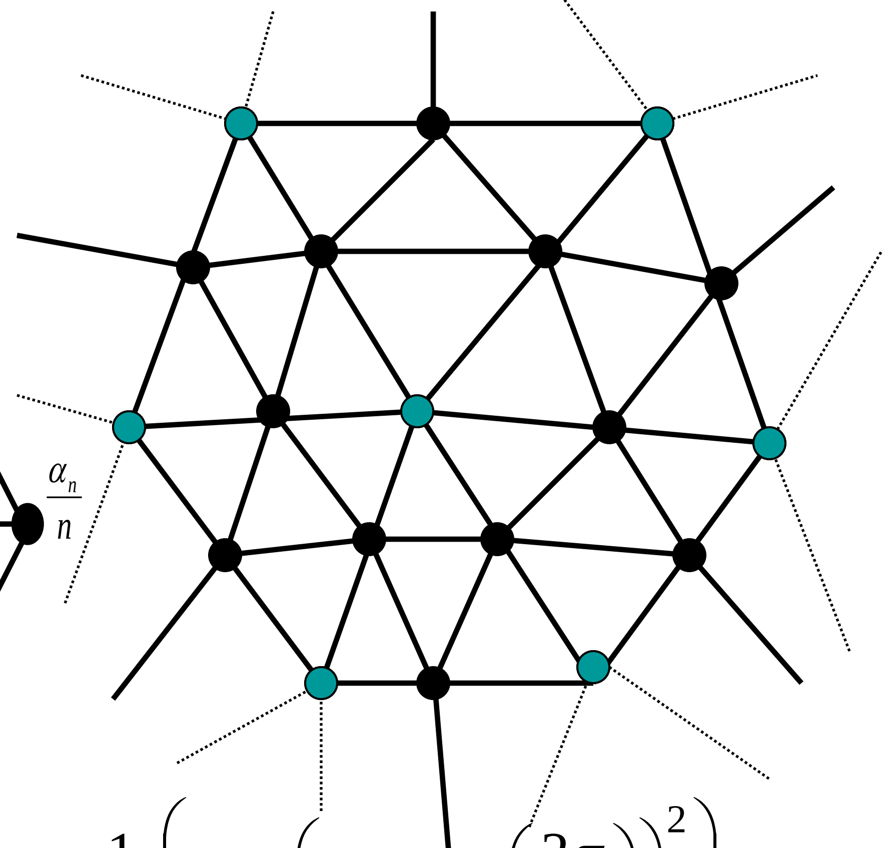
- Two refinement rules:
  - **Odd rule**: add new control points
  - **Even rule**: modify the existing control points

# Loop Subdivision Mask: valence n




**Odd rule:**
**new vertices on edges**



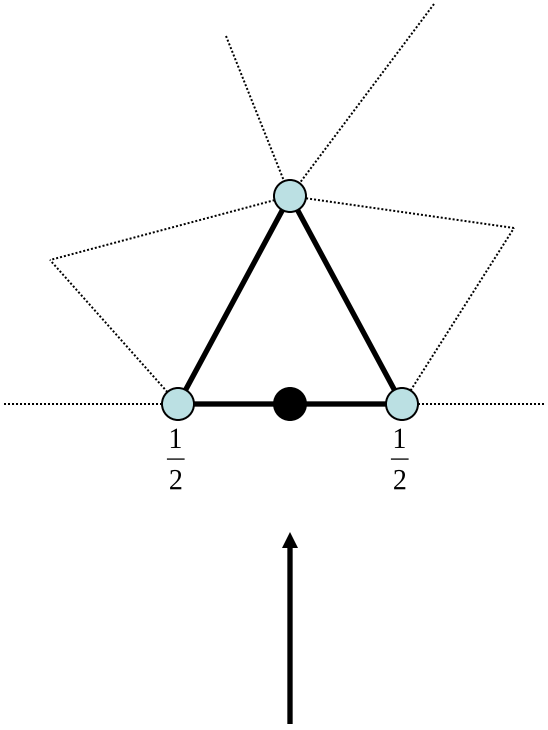
**Even rule:**
**Modify vertices from previous step**

$$V^k = (1 - n\alpha) V^{k-1} + \alpha \sum_{i \in N(V)} V_i$$

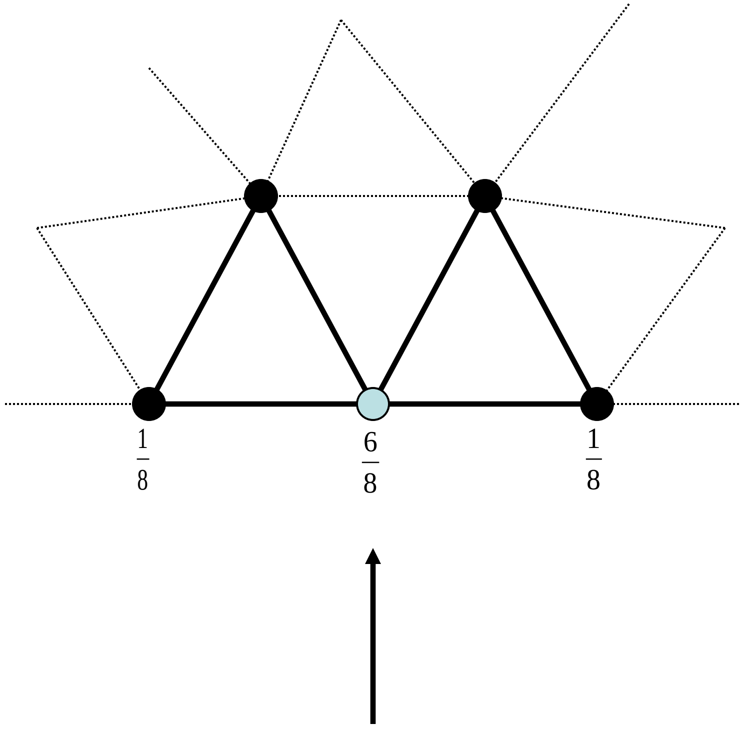
$$\alpha_n = \frac{1}{64}\left(40 - \left(3 + 2\cos\left(\frac{2\pi}{n}\right)\right)^2\right) \qquad \alpha_6 = \frac{1}{16}$$

# Loop Subdivision Boundaries

- Subdivision Mask for Boundary Conditions




Edge Rule




Vertex Rule

# What About Continuity and Curvature..

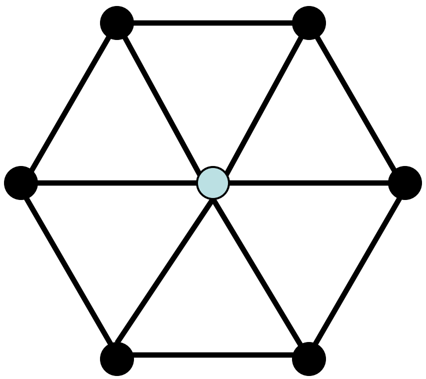
- Subdivision mask weights *w* are derived from splines, such as B-Splines.
  - Subdivision surfaces converge to spline surfaces with $C^2$ continuity everywhere.**
  - Too lengthy to cover here, but there is lots of literature.

**Subdivision Methods for Geometric Design**

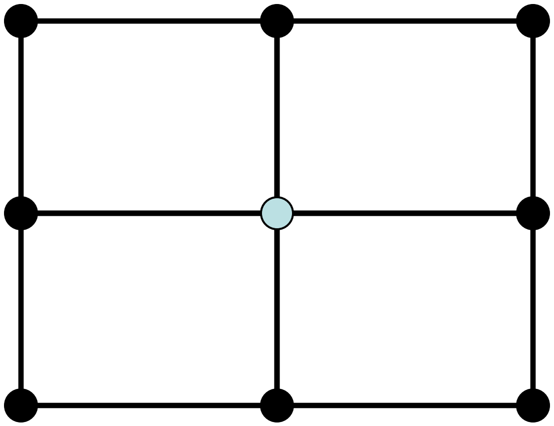
Joe Warren, Henrik Weimer. (2002)

**Math works out except at "**Extraordinary Vertices**". Most Subdivision Schemes have and "ideal" valence for which it can be shown that the limit surface will converge to a spline surface.

# *Ordinary* and *Extraordinary*



Loop Subdivision
Valence 6



Catmull-Clark Subdivision
Valence 4

•Subdividing a mesh does not add ***extraordinary*** vertices.

•Subdividing a mesh does not remove ***extraordinary*** vertices.

How should ***extraordinary*** vertices be handled?

•Make up rules for ***extraordinary*** vertices that keep the surface "smooth".

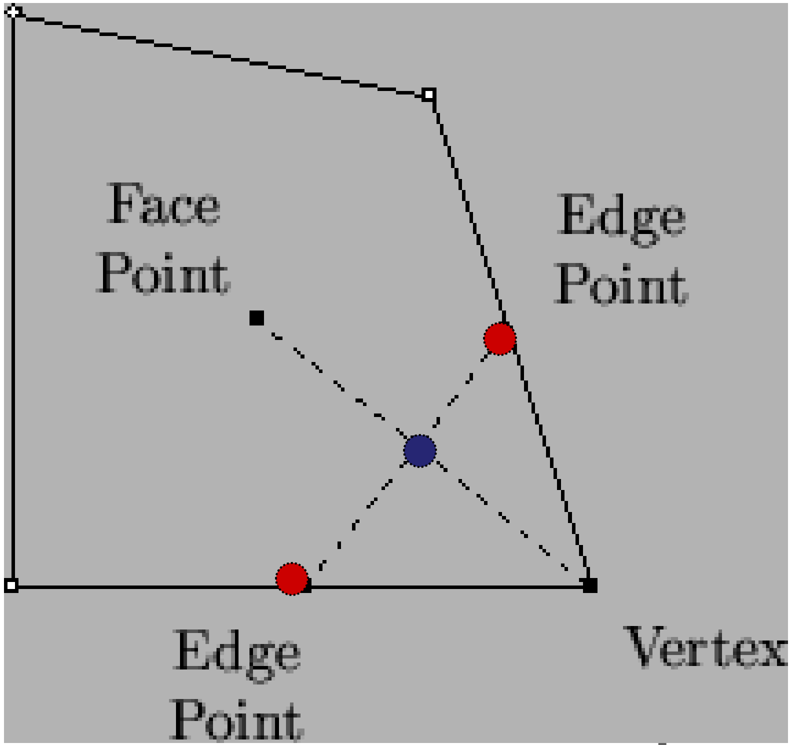
# Doo-Sabin subdivision surfaces

Extend Chaikin's algorithm to generate uniform bi-quadratic spline surfaces

**Face point: average of the 4 vertices**

**Edge point: average of the edge adjacent to the vertex**

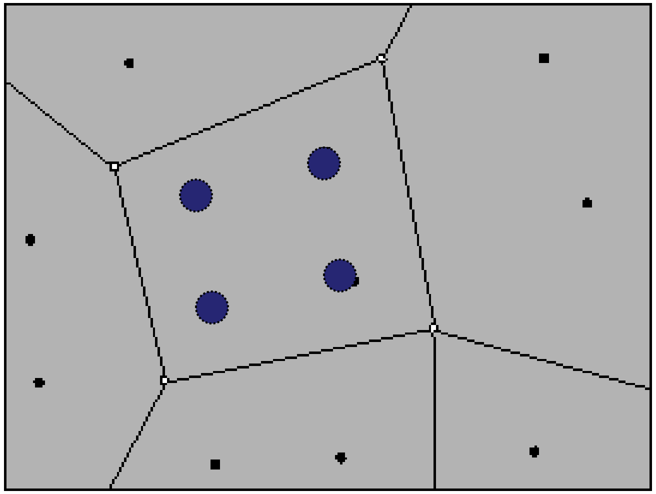
For each **Vertex** of a face generate a new point **P** as average of the 4 points:
(Face, Edge,Edge,Vertex)

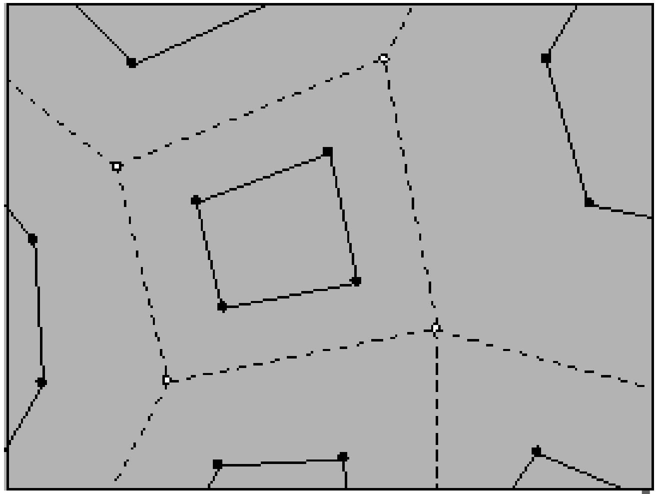


# Doo-Sabin subdivision surfaces

- For each face:

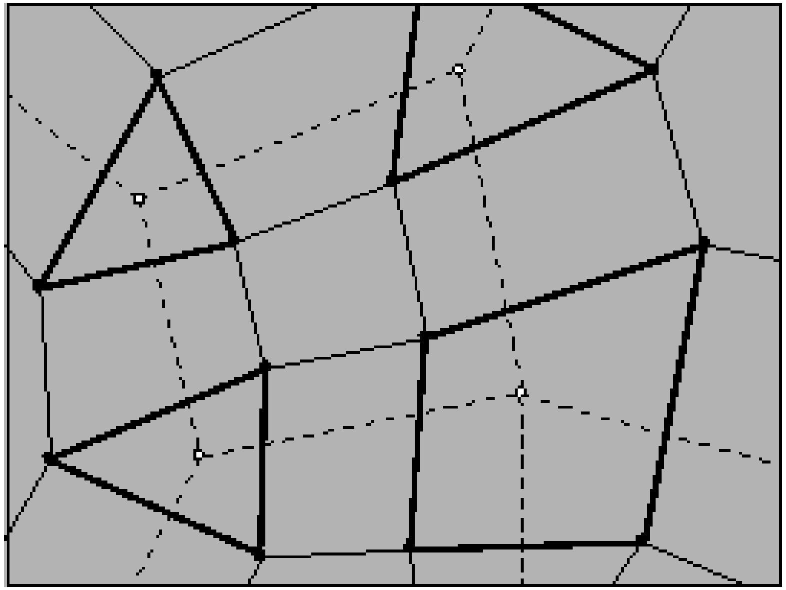
Connect the new points **P generated for each vertex of the face**





# Doo-Sabin subdivision surfaces

- For each vertex, connect the new cp **P** with the new points in adjacent faces

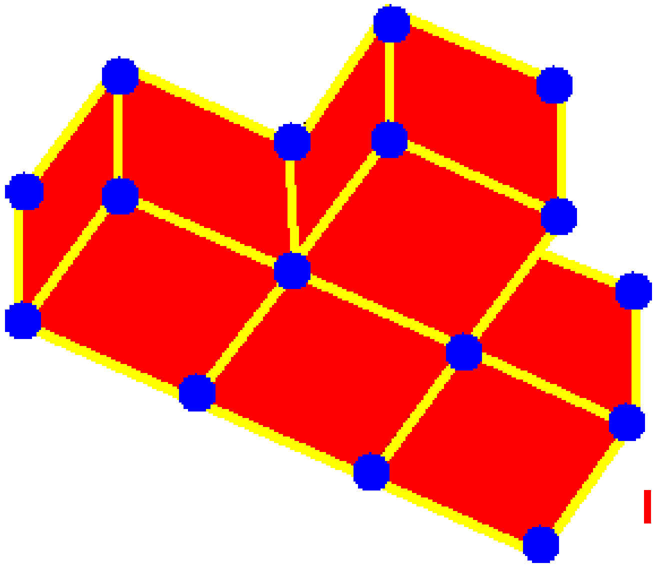


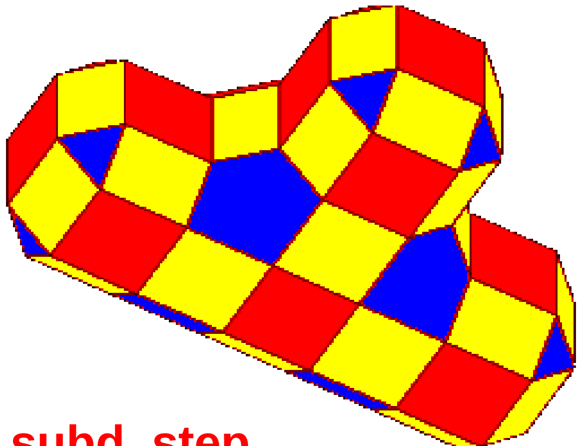
- For each edge, connect the new CP generated for the faces sharing the edge
- The new generated polygons define the new control mesh

# Example: Doo-Sabin ('78) subdivision surfaces

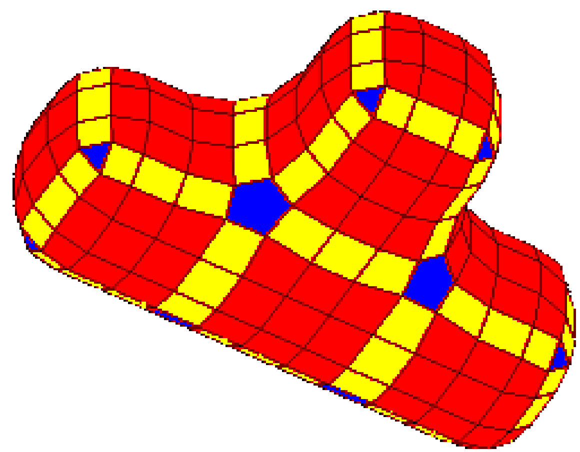
This process generates one new face at each original vertex, n new faces along each original edge, and n x n new faces at each original face.

Triangular Faces converge to extraordinary points





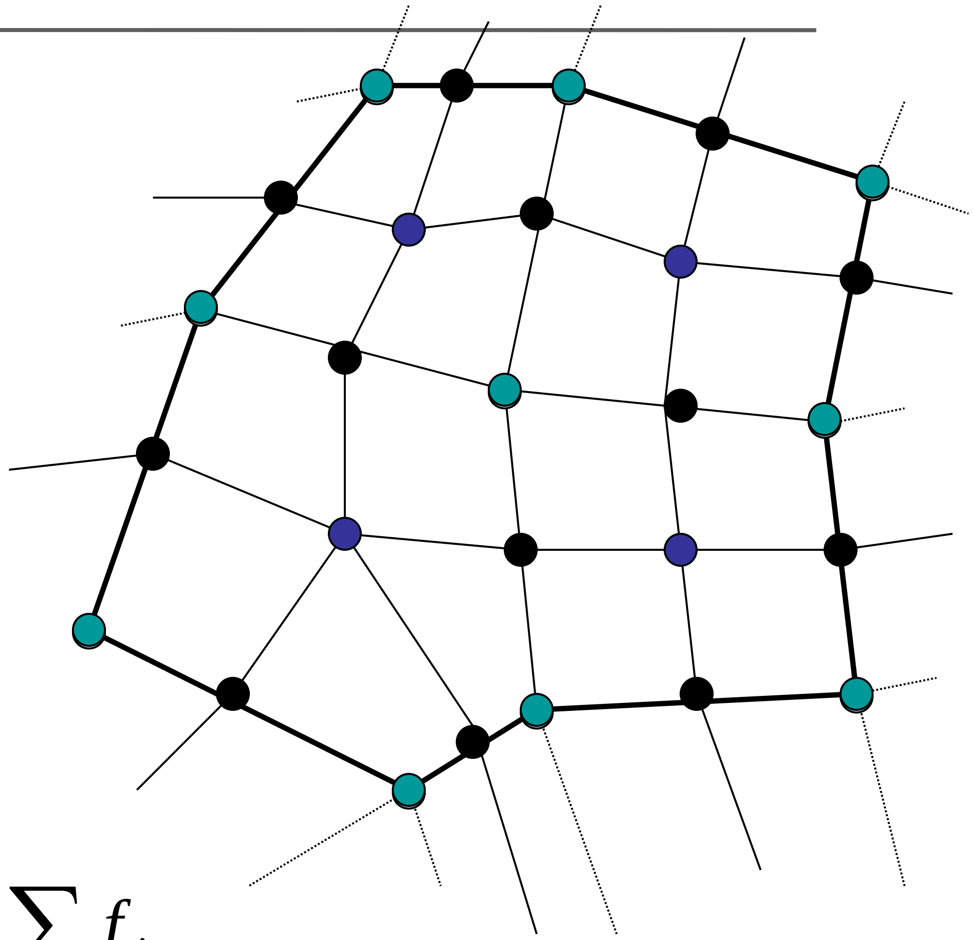
I subd. step



Generate limit surfaces $C^1$, $C^0$ in extraordinary points

All vertices has valence 4

# Catmull-Clark Subdivision (1978)



- FACE

$$f = \frac{1}{n}\sum_{1}^{n} v_i$$

- EDGE

$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

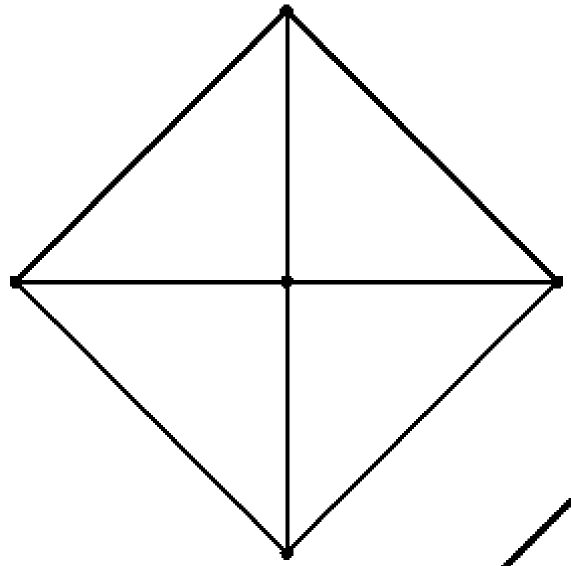
- VERTEX

$$v_{i+1} = \frac{n-2}{n} v_i + \frac{1}{n^2}\sum_j e_j + \frac{1}{n^2}\sum_j f_j$$

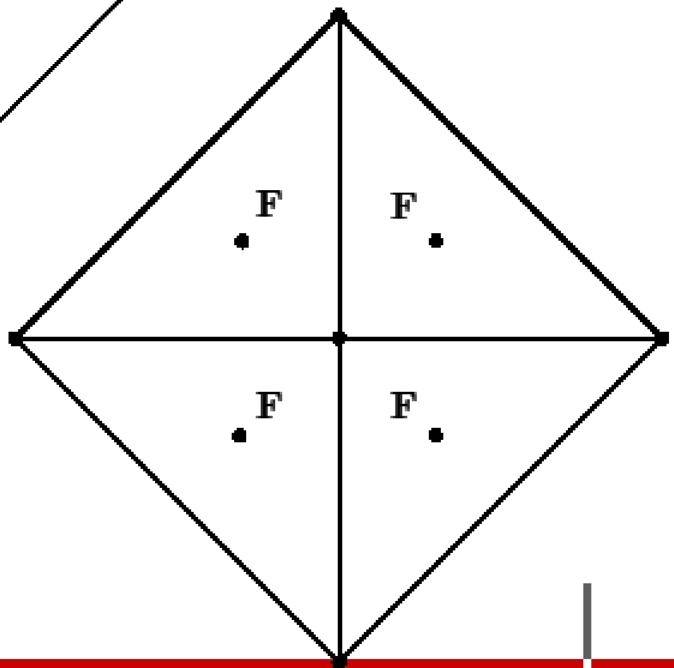
We get uniform bi-cubic spline surafces

# Catmull Clark subdivision surfaces

- Initial mesh:

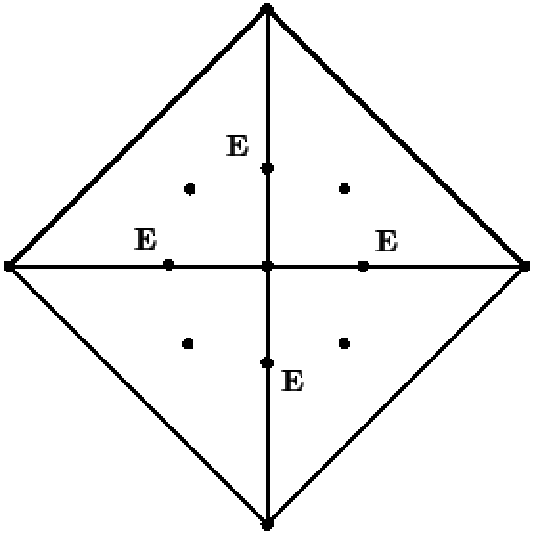


- Compute the face point as mean of the vertices of the face

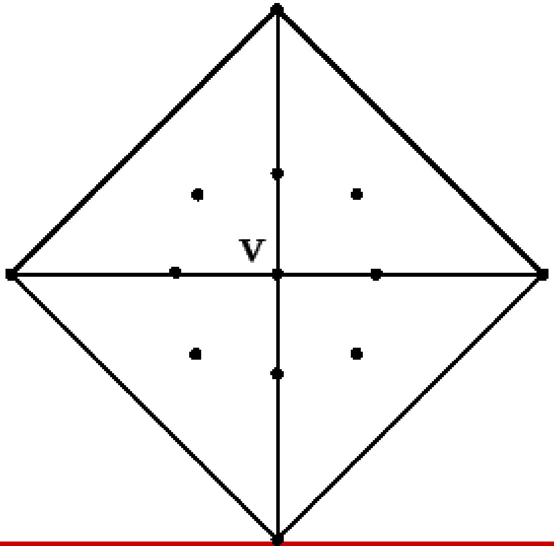


# Catmull Clark subdivision surfaces

- Compute the edge point as average of 4 points: the 2 vertices of the edge, the 2 new face points of the adjacent faces




- Update the vertex point:

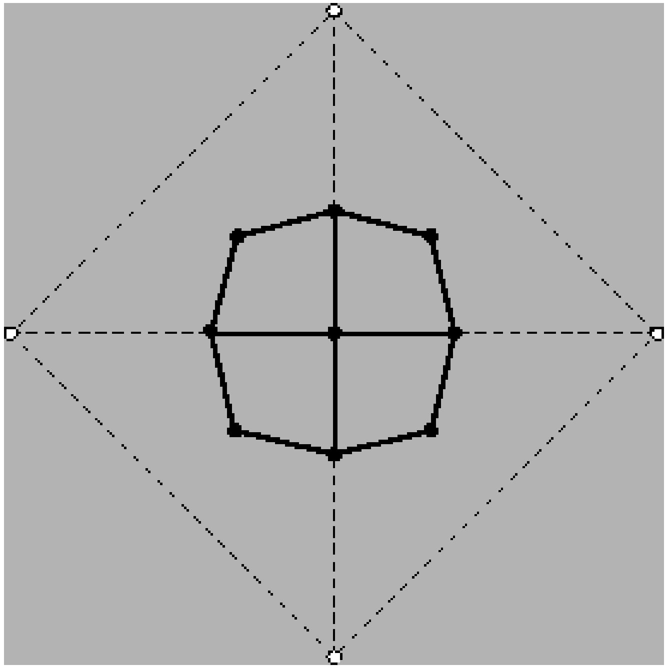


# Catmull Clark subdivision surfaces

- New refined mesh:

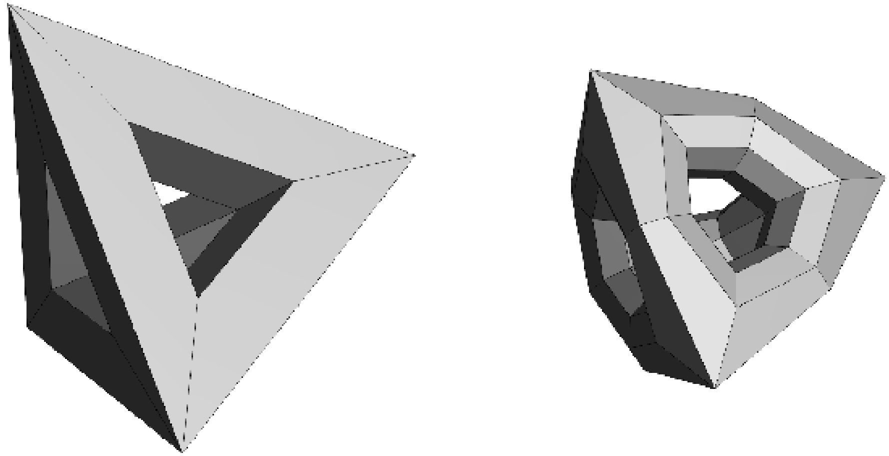
  - connect the new face points to the new edge points,

  - connect the vertex point to the edge points



- **After the first refinement all the polygons are quadrilaterals**

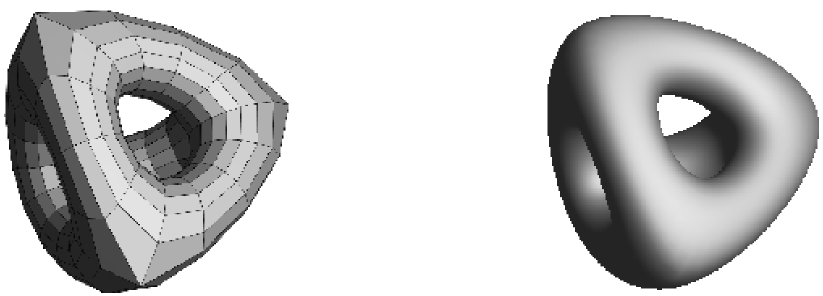
# Example: Catmull-Clark SS



(a) (b)

•The vertices of the original mesh maintain the same valence

•Extraordinary vertices have valence ≠ 4



• **Generate limit surface $C^2$, $C^1$ at extraordinary points**

•Each patch of 4x4 CPs with rectangular topology (valence 4) represents a uniform bi-cubic spline surface

# Modeling with Catmull-Clark

- Subdivision produces smooth continuous surfaces.
- How can "sharpness" and creases be controlled in a modeling environment?

ANSWER: Define new subdivision rules for "creased" edges and vertices.

1. Tag Edges sharp edges.
2. If an edge is sharp, apply new sharp subdivision rules.
3. Otherwise subdivide with normal rules.

CC surfaces in Toy story 2 and Geri's game

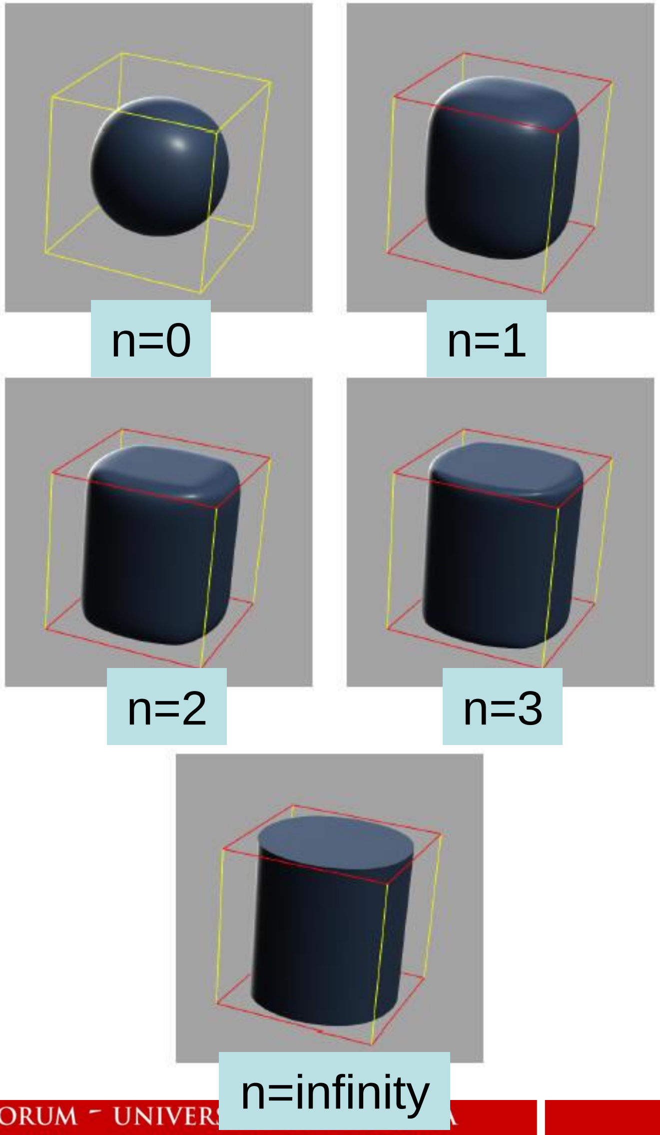
# Sharp edges…

1. Tag Edges as "**sharp**" or "not-sharp"
   - n = 0 – "**not sharp**"
   - n > 0 – sharp

During Subdivision,

2. if an edge is "sharp", use sharp subdivision rules. Newly created edges, are assigned a sharpness of n-1.
3. If an edge is "not-sharp", use normal smooth subdivision rules.

IDEA: Edges with a sharpness of "n" do not get subdivided smoothly for "n" iterations of the algorithm.



# Sharp Rules (CC)

FACE (unchanged)

$$f = \frac{1}{n}\sum_{1}^{n} v_i$$

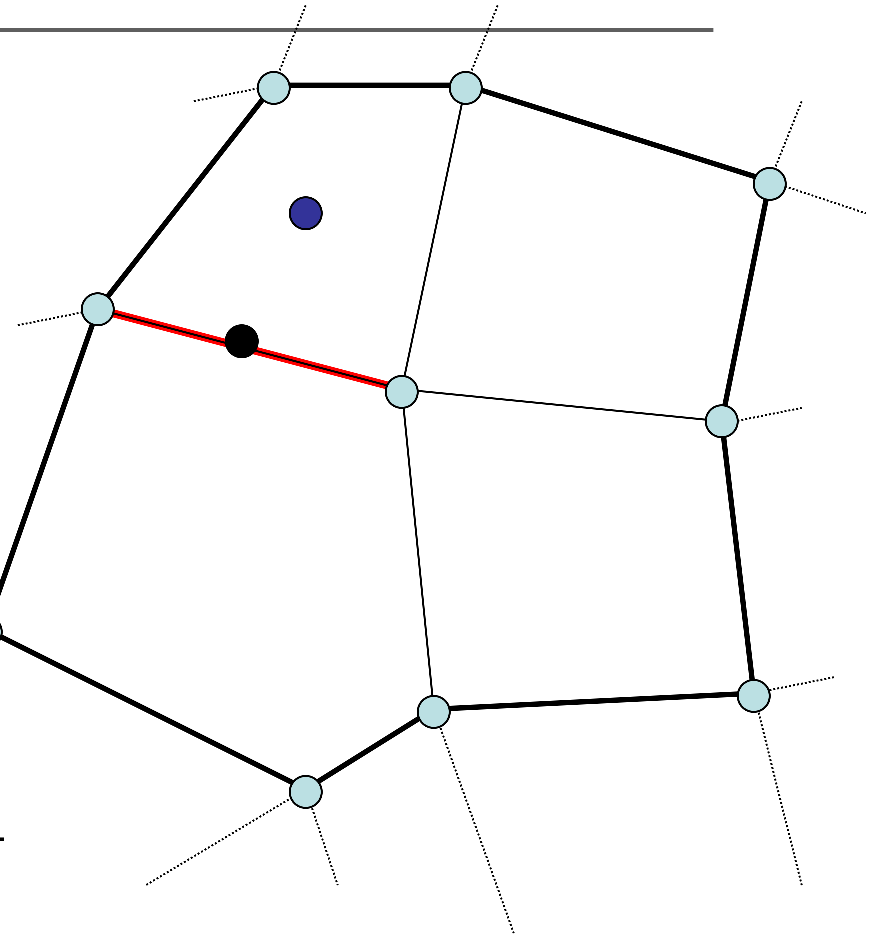
EDGE

$$e = \frac{v_1 + v_2}{2}$$

VERTEX

# adj. Sharp edges

| | | |
|---|---|---|
| dart | >2 | $v_{i+1} = v_i$ |
| crease | 2 | $v_{i+1} = \frac{e_1 + 6v_i + e_2}{8}$ |
| corner | 0,1 | $v_{i+1} = \frac{n-2}{n} v_i + \frac{1}{n^2}\sum_j e_j + \frac{1}{n^2}\sum_j f_j$ |

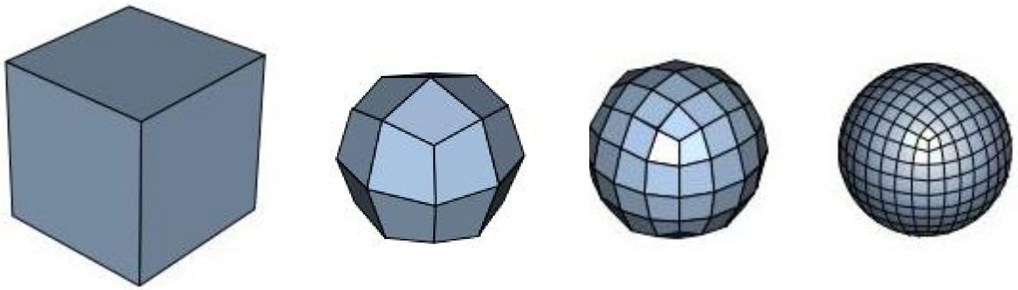


# Subdivision rendering

All the shown surfaces are piecewise flat approximations of the corresponding limit surfaces

# Refinement vs Exact Evaluation

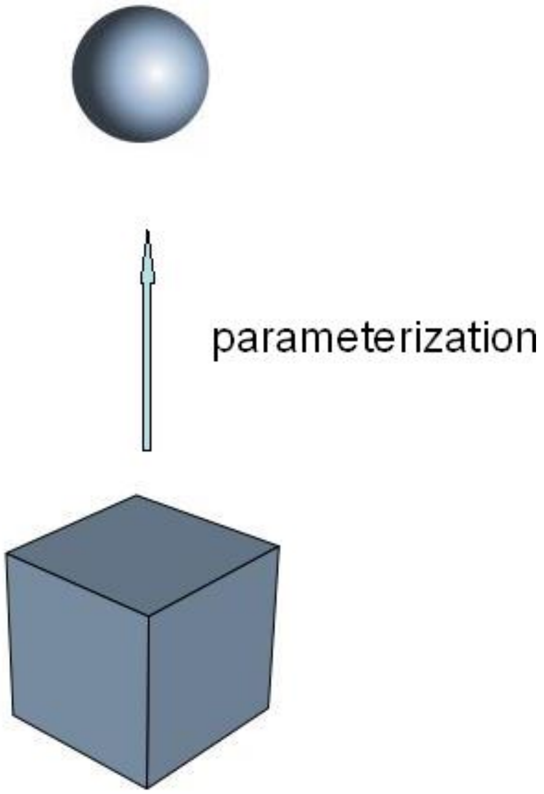
- **Refinement of a coarse mesh only approximates the smooth limit surface**
  - **this produces a huge amount of faces that have** to be stored, manipulated and rendered by the graphics pipeline
  - Their use in real-time interactive graphics applications is even more computational demanding and slows down the entire rendering process



# Refinement vs Exact Evaluation

- **Exact evaluation (Stam for CC):**

provides a direct way to render a subdivision surface by exact evaluation of the limit surface in a suitable parametric space associated to each primitive

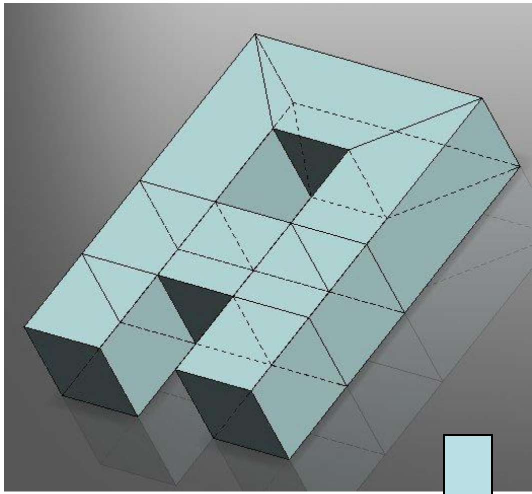



**Both schemes offer a natural parallelization**

**S**TAM J.: Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values. In SIGGRAPH '98

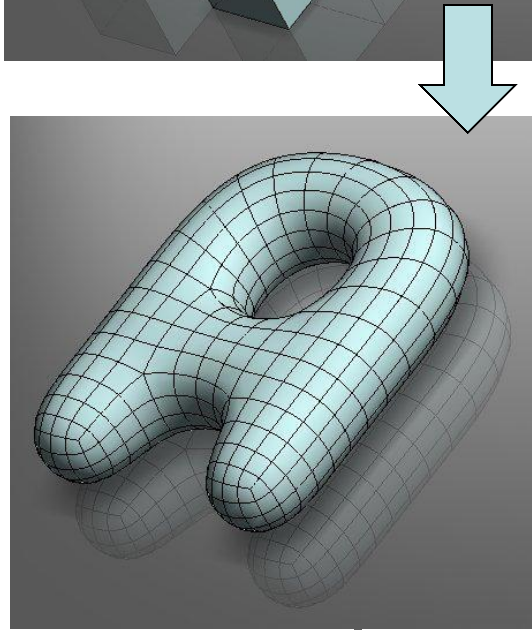
# Exact CC Subdivision Surfaces inside a CAD system

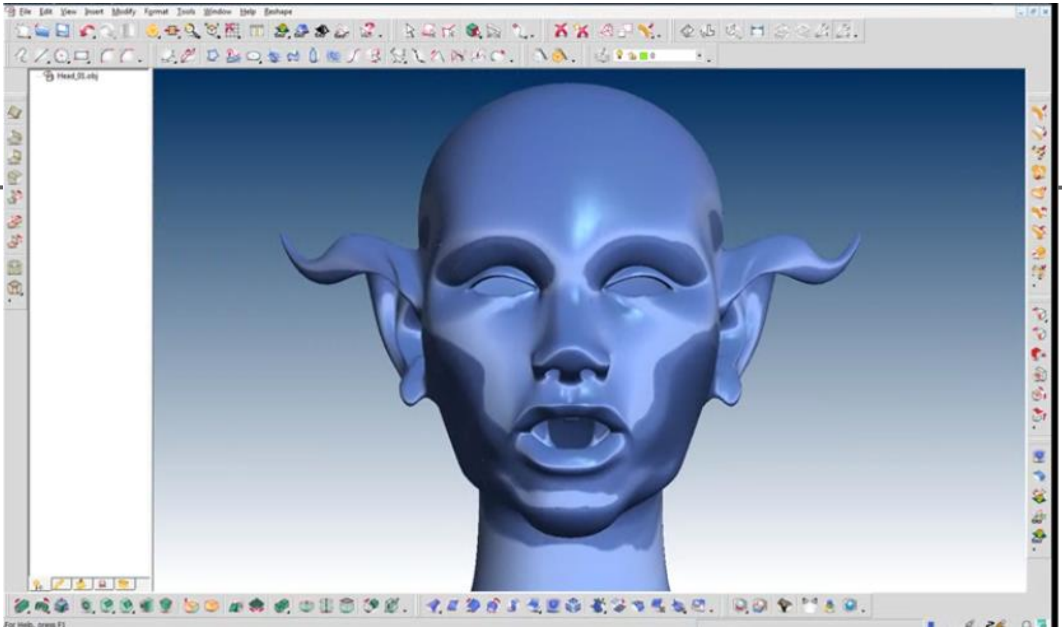
**[Think3 & Univ.of Bologna,** New Interactive Technologies for CAD- **EUROSTARS Project 2010-2012]**

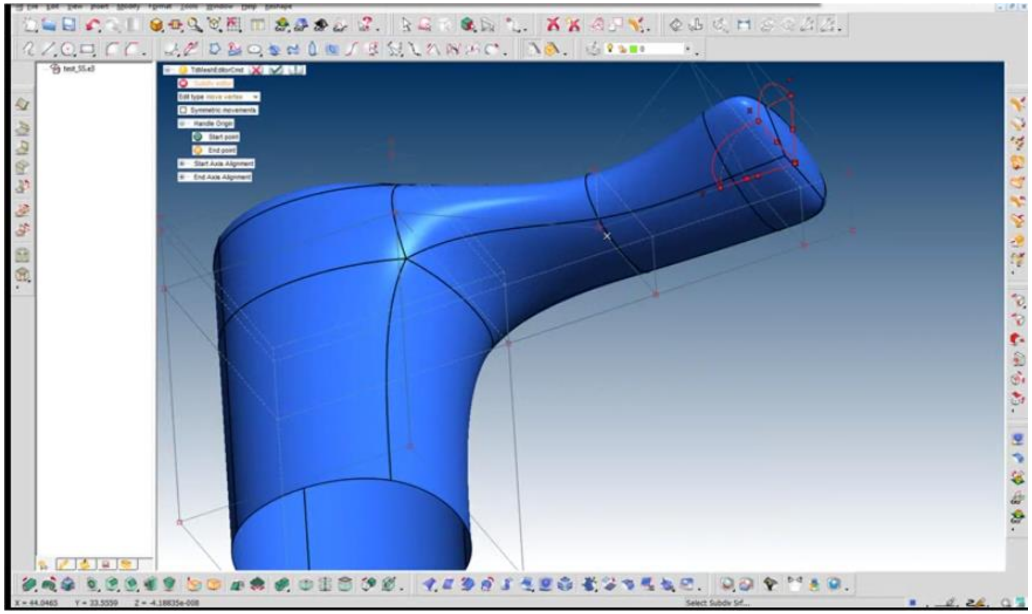


- Design and development of a software module for subdivision surfaces inside **thinkdesign** geometric kernel considering the following issues:
  - –Algorithm for **exact evaluation** (use Jos Stam algorithm to have an F(u,v))
  - –B-rep representation for solids made of subd surfaces.

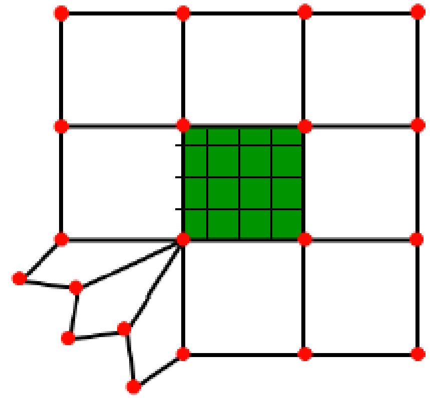
    A mesh is converted to a B-rep solid where each face corresponds to a mesh face. Each face is evaluated as a Catmull-Clark surface with the original mesh as control points.
  - –Tool to create and edit subd shapes.



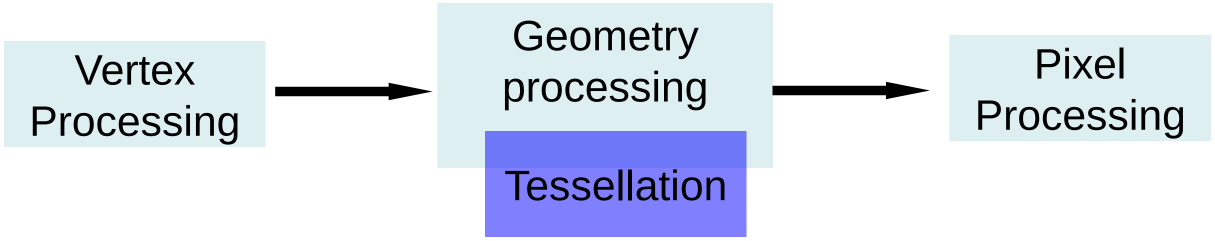




# Patch-based Geometry Shader Tesselation in GPU



Given an input multi-sided patch, the geometry shader tessellates the main face of the patch and directly invokes the rasterizer for rendering

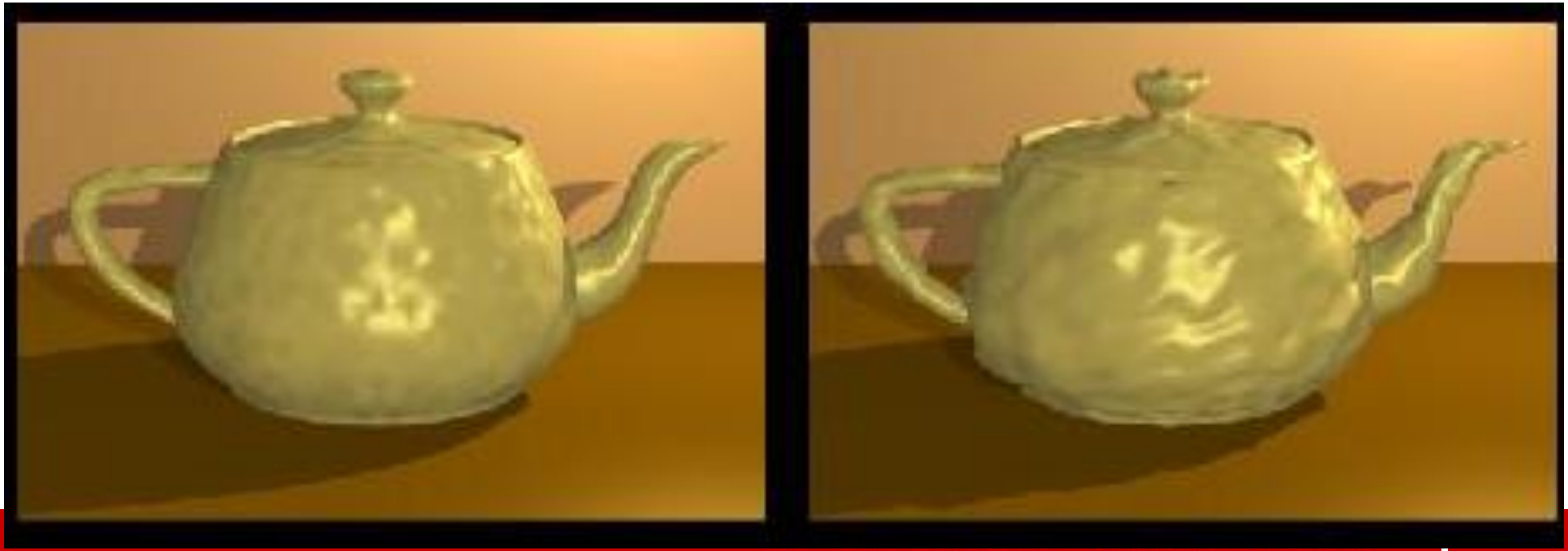



- Data from the control mesh is collected into vertex and patch streams, and passed to the GPU for the evaluation and rendering steps.

- ***Subdivision kernel (Geometry shader):*** each patch is either *refined* by the CC scheme at a given depth *d, or exactly* evaluated.

# Displacement Mapping

- Bump mapping provides normals to simulate an alterated geometry ( problems with shadows, silhouettes)
- Displacemente mapping: alterate the geometry of the surface
- Use height field to perturb a point on the surface along the normal vector.



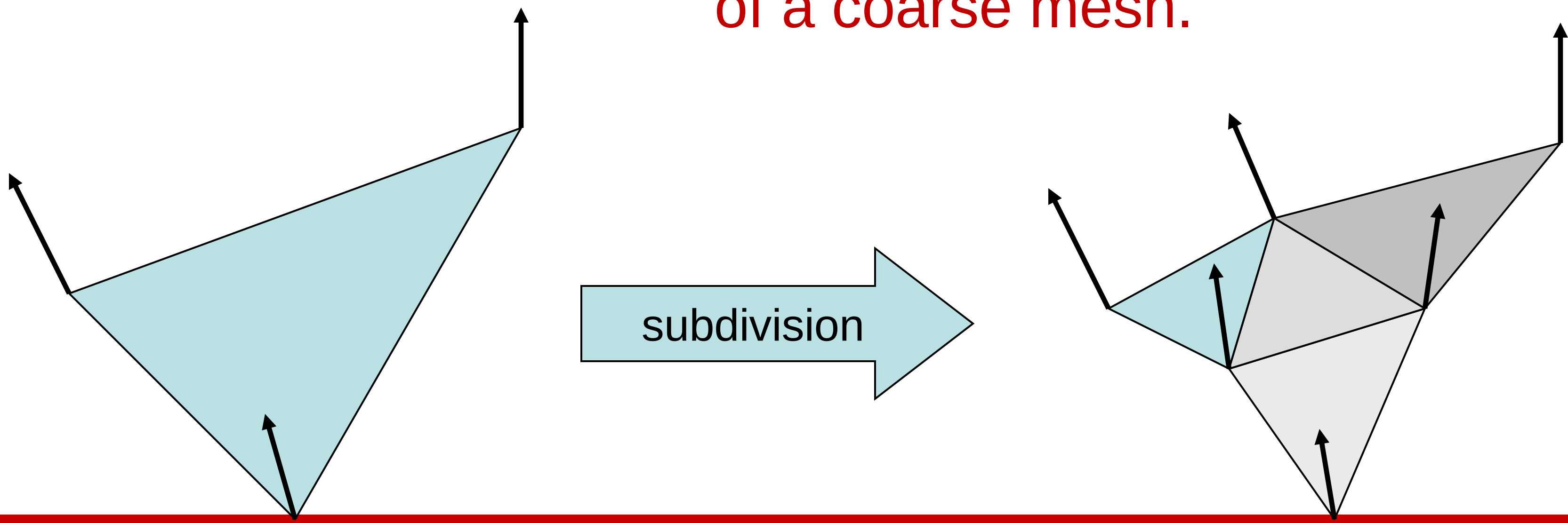
# Displacement Mapping for subdivision surfaces

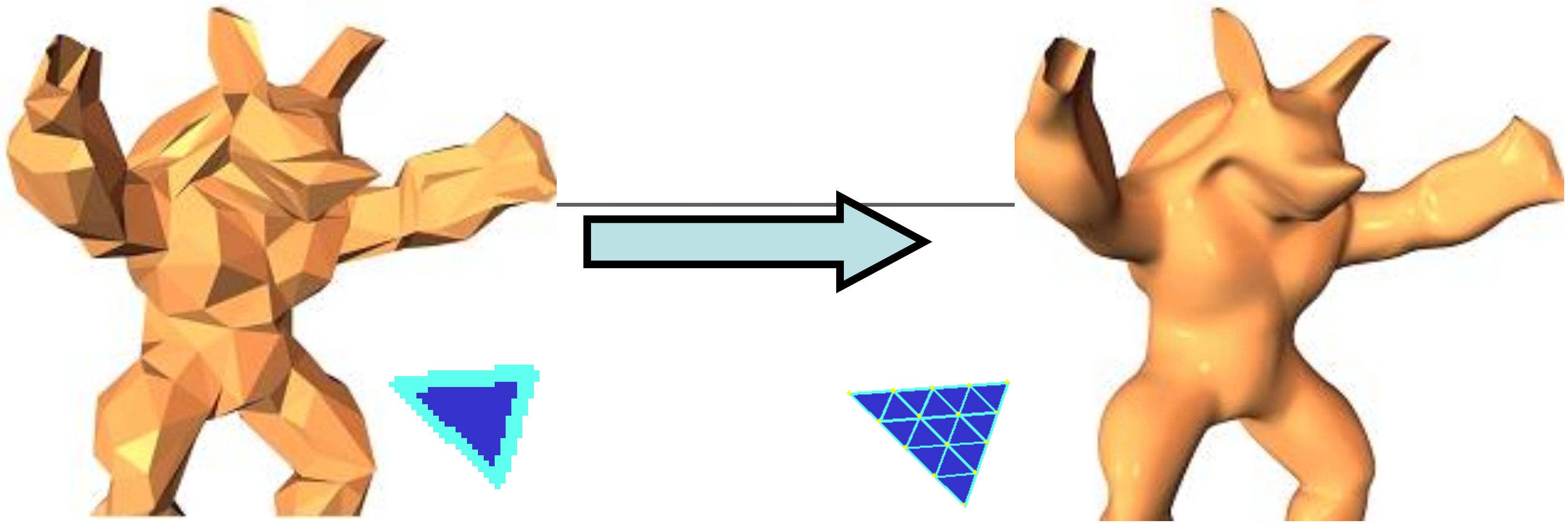
Let **p** be a point on surface and **n** its normal, then the point on the displaced surface is given by

$$\mathbf{s} = \mathbf{p} + d\mathbf{n}$$

with **d** scalar value that represnets the displacement of the point **p**

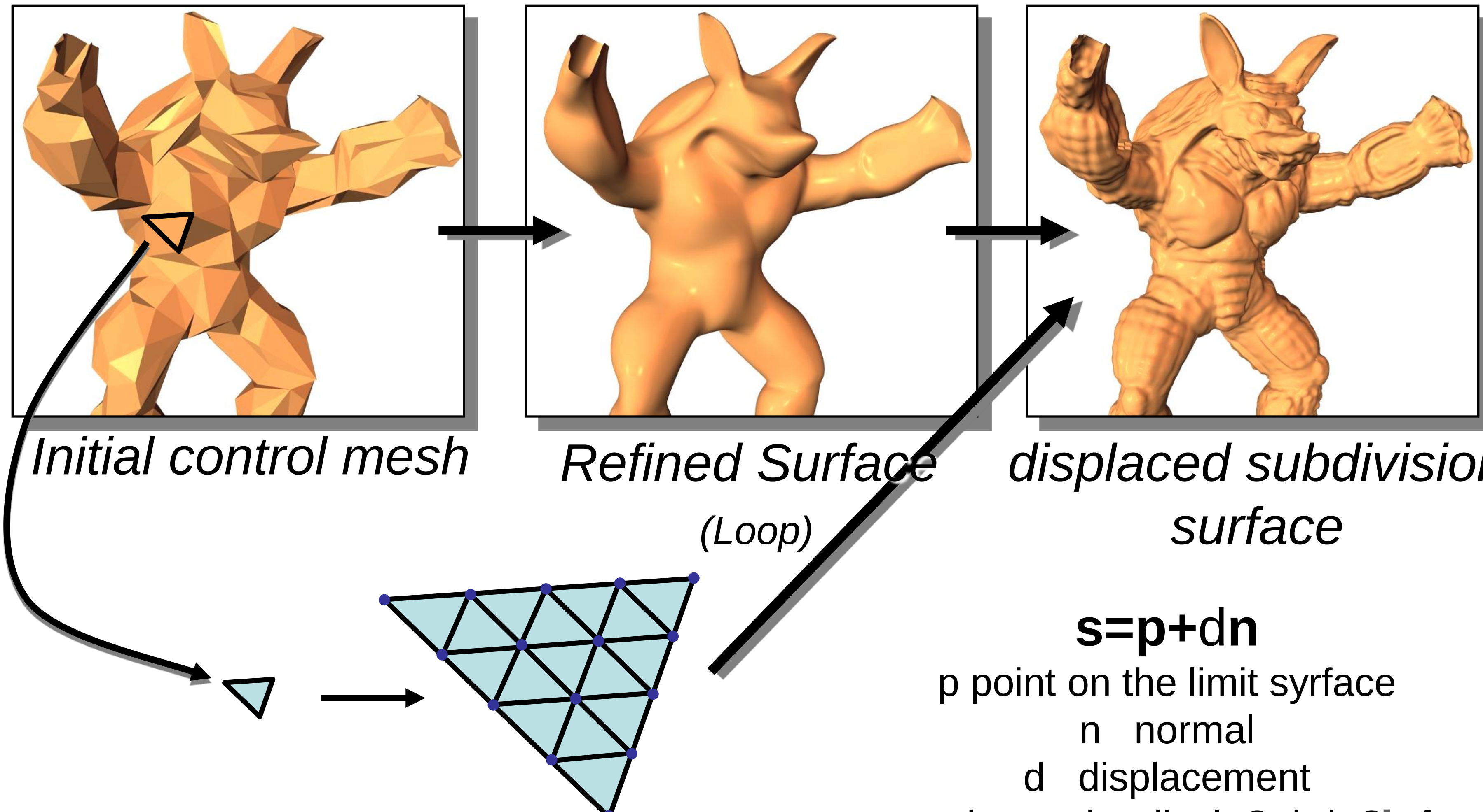
Define a displacement map (height field) for each triangle of a coarse mesh.





1. From a coarse mesh M0 apply a subdivision scheme to get a smooth surface M1
2. Apply a displacement along normal vector at each vertex of M1

# Displacement Mapping for subdivision surfaces




scalar displacements

**s=p+dn**

p point on the limit syrface

n normal

d displacement

s point on the displ. Subd. Surf.