INF2110 - I2G

Surface Models

Caroline Larboulette

Fall 2017

Outline

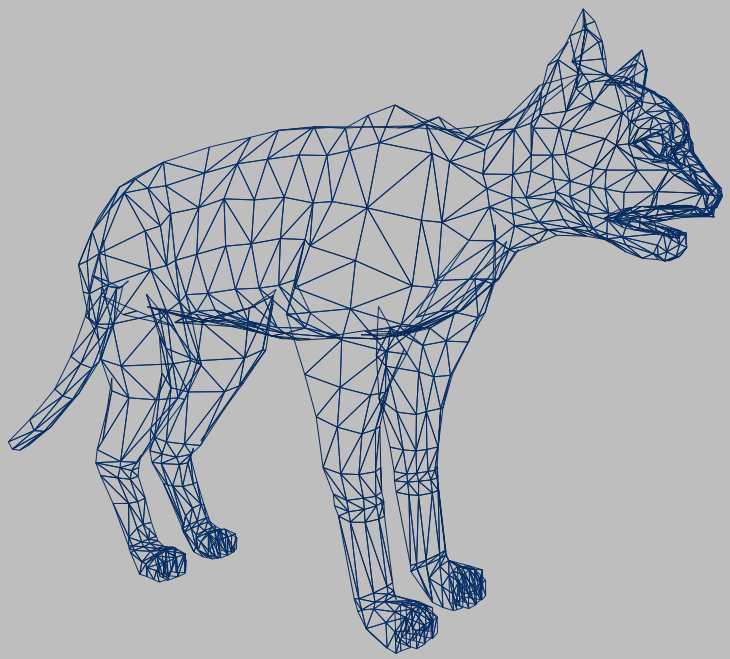
Surface Models

- Definition
  Mathematical representation of a $3D$ object
- Depends on the objet
  What should my object look like ? (round, square, smooth, sharp...)
- Depends on the application
  What kind of deformation do I want on my objet ?
  What rendering algorithm will I use ?
  What are the costs in modeling/animation/rendering ?
  - time
  - memory

Surface Models

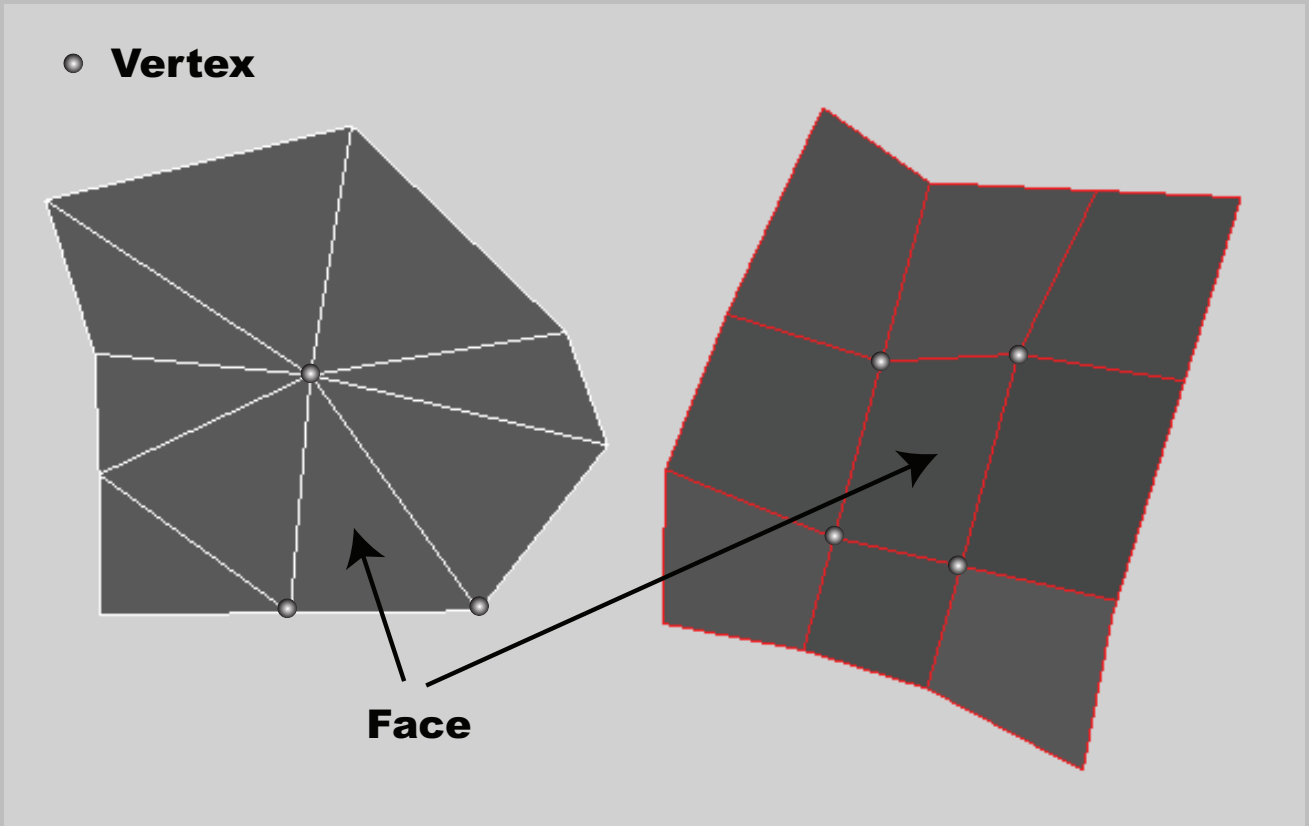
- Surfaces well-suited for deformable solids
  - Polygonal Surfaces
  - Parametric Surfaces
  - Subdivision Surfaces
  - Implicit Surfaces
- Other representations
  - Solid Modeling (CSG)
  - Particle Systems
  - Point Based Graphics

Polygonal Surfaces



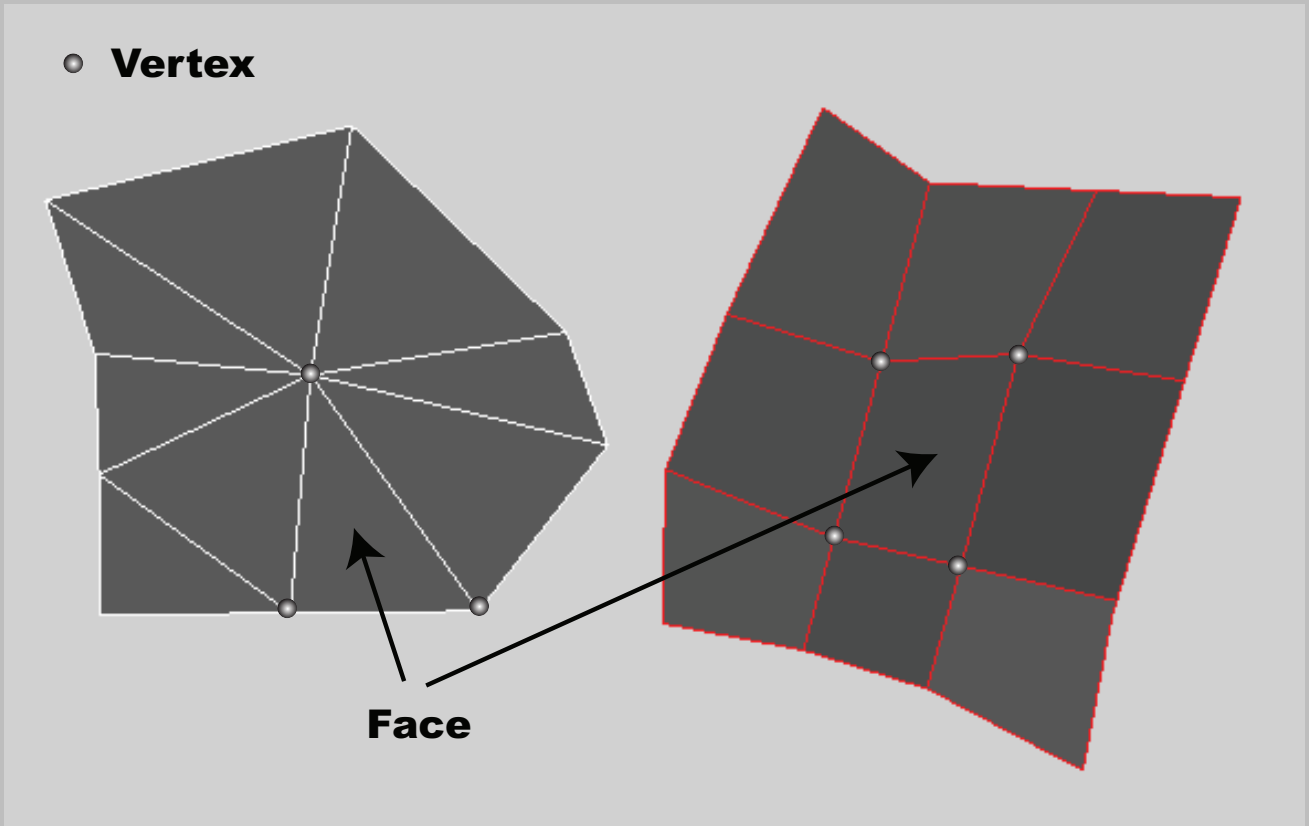
Polygonal Surfaces

Set of flat geometric primitives (quads, triangles)



Polygonal Surfaces

Set of flat geometric primitives (quads, triangles)



3 vertices define a plane $ax + by + cz + d = 0$

Polygonal Surfaces

Set of flat geometric primitives (quads, triangles)
3 vertices define a plane $ax + by + cz + d = 0$
Good for rendering:

- Easy/fast ray-triangle intersection
- Surface normal = cross product of 2 consecutive edges
- Normals per point
- Graphics boards render triangles (fast)

Polygonal Surfaces

Set of flat geometric primitives (quads, triangles)
3 vertices define a plane $ax + by + cz + d = 0$
Not as good for modeling

- Only approximates the surface
- Modeling is a tedius task
- Good for flat surfaces: planes, cubes, tables...
- Not so good for curved surfaces (more triangles)
- More polygons = better approximation of the surface

Polygonal Surfaces

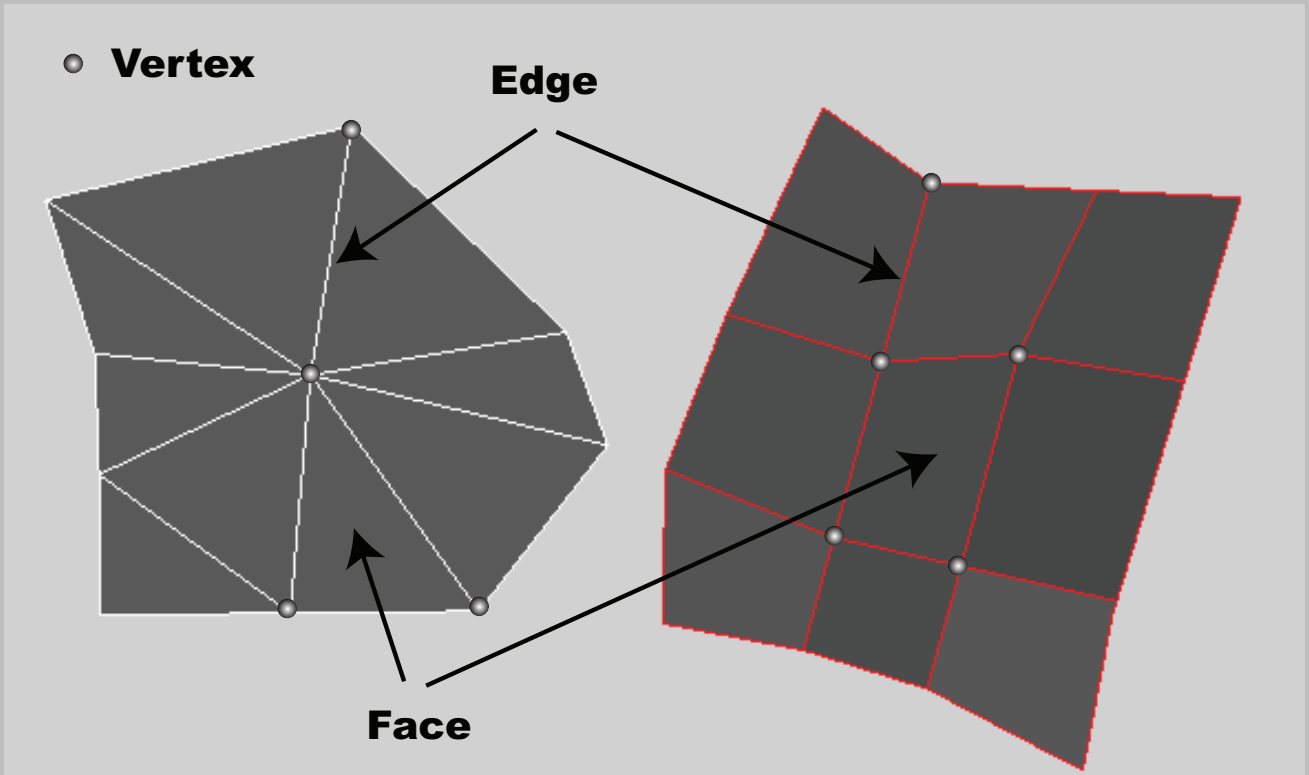
Set of flat geometric primitives (quads, triangles)
3 vertices define a plane $ax + by + cz + d = 0$
Usually good for animation

- Manipulation of vertices
- Animation might be slow if too many triangles

Mesh

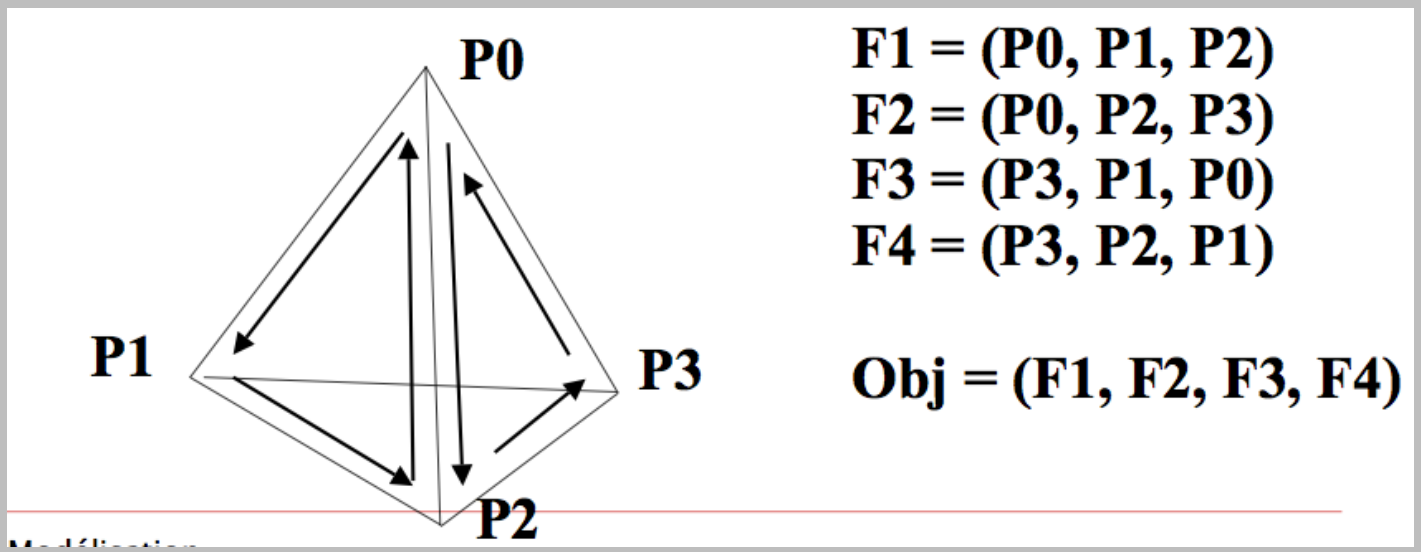
A mesh is a collection of primitives

- edges + vertices OR
- vertices + faces



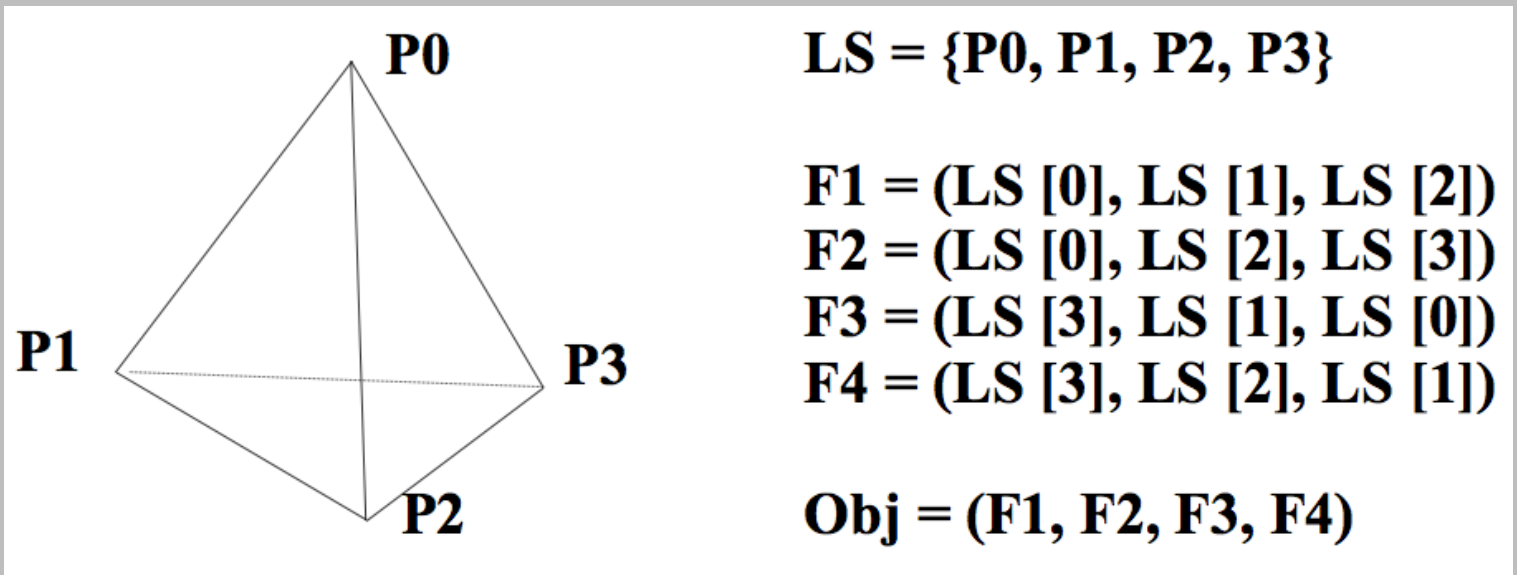
Mesh

- Easy to set up
- ⚠Duplicates (using an index solves the problem)
- ⚠Coherency (there is an order !)



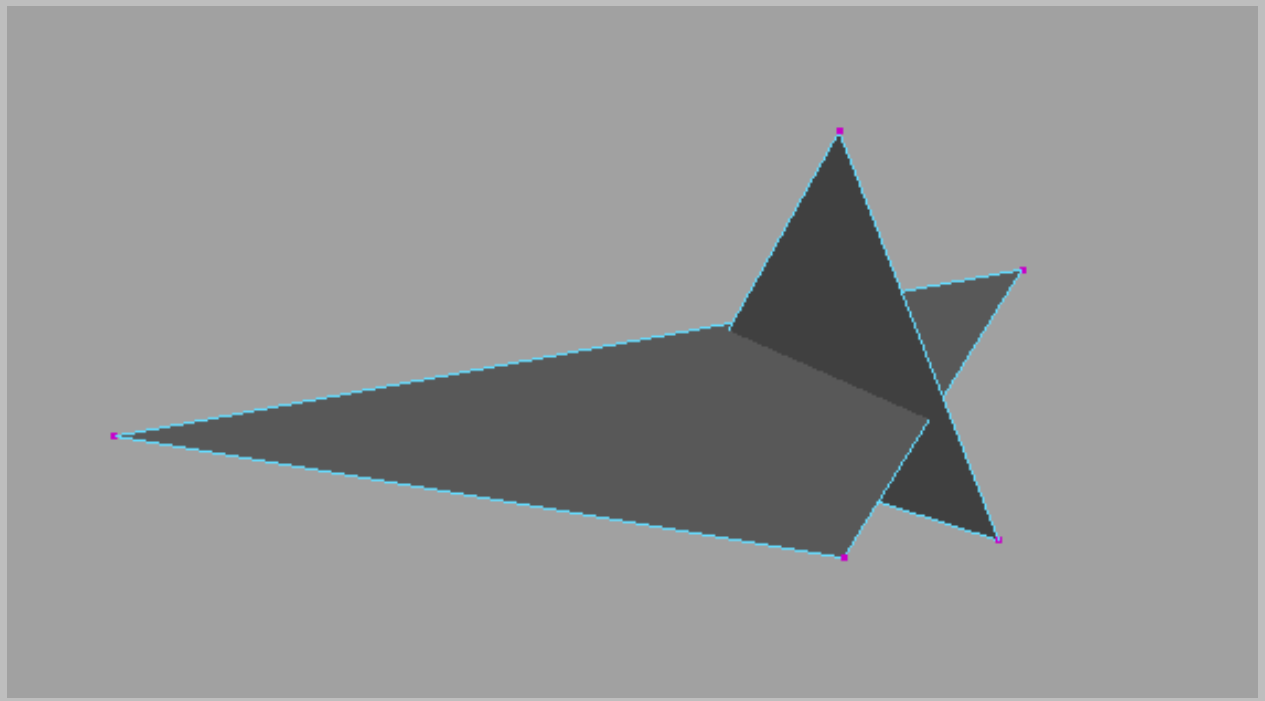
Mesh

- Easy to set up
- ⚠Duplicates (using an index solves the problem)
- ⚠Coherency (there is an order !)



Properties

- Non self-intersecting = no edge passes through a polygon

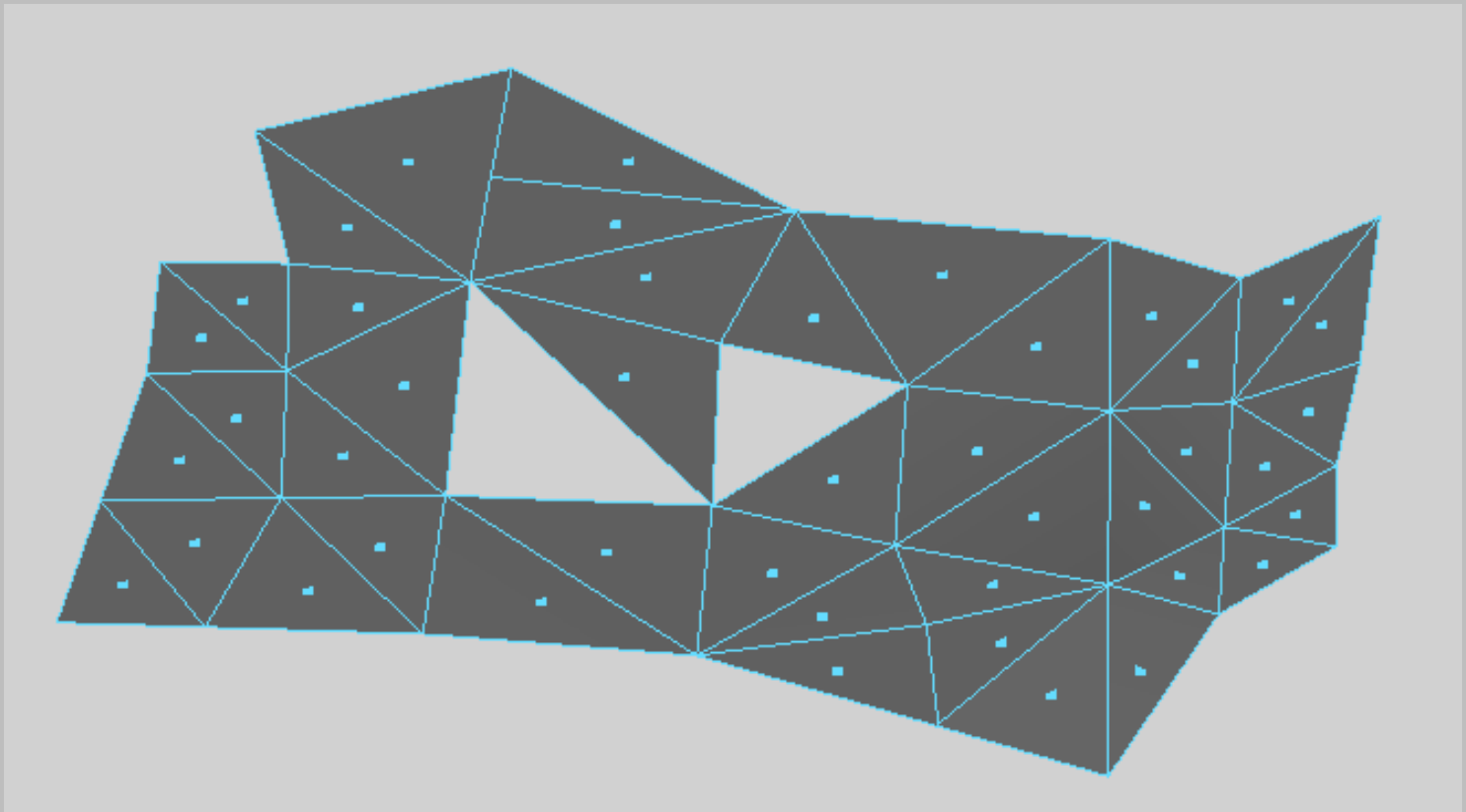


Properties

- Non self-intersecting
- Manifold = no hole and no singularity

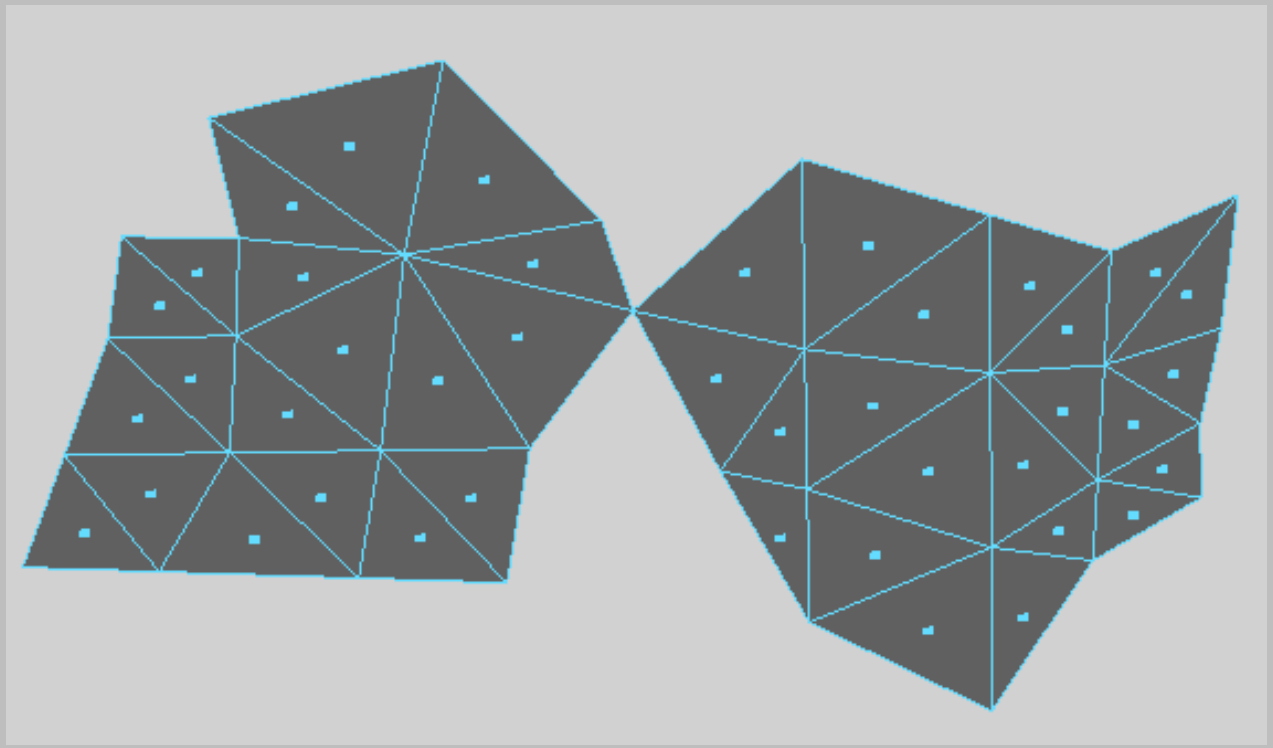
Properties

- Non self-intersecting
- Manifold = no hole and no singularity
  Hole =



Properties

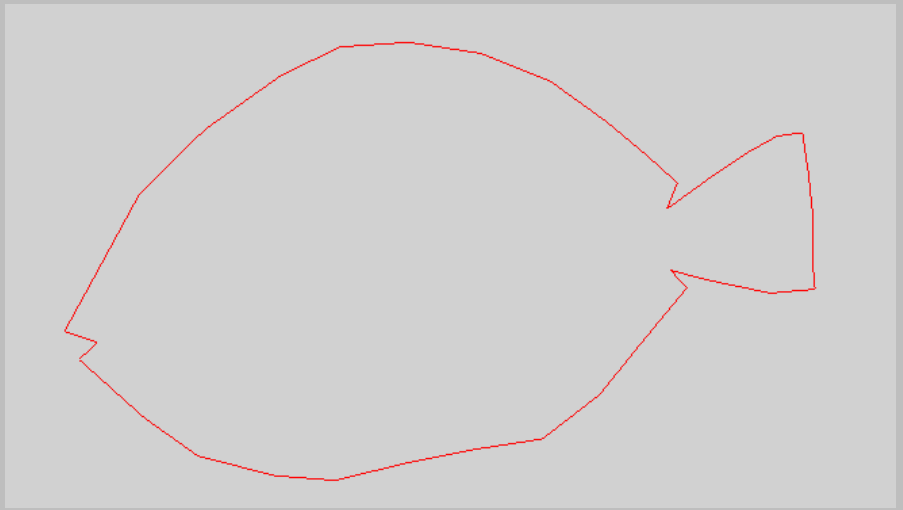
- Non self-intersecting
- Manifold = no hole and no singularity
  Singularity = two sections of a mesh connected with a single vertex



Creating an object

Under Maya (by hand):

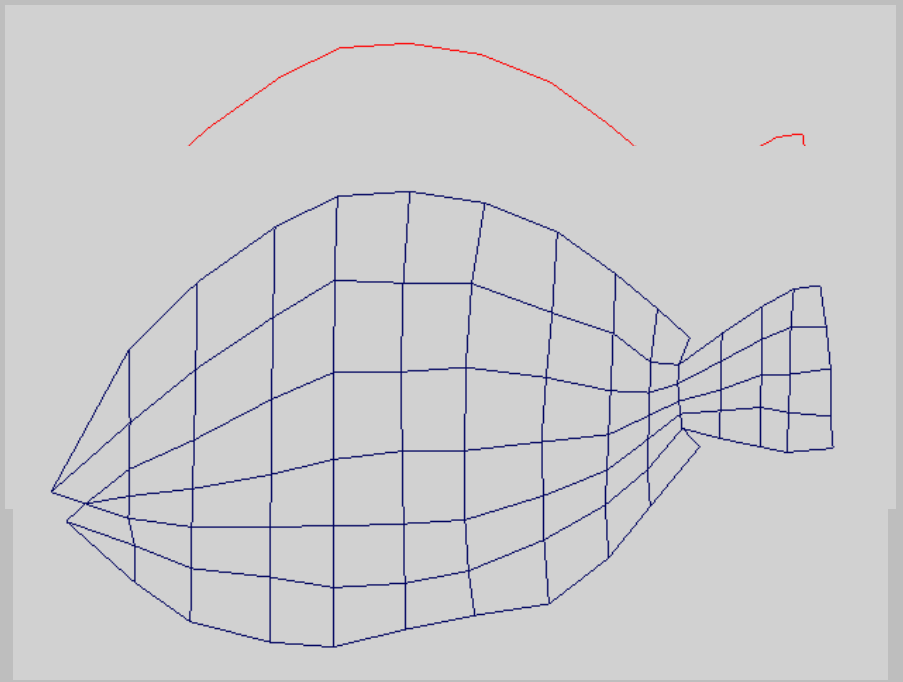
- Create Polygon Tool
- Multi-Cut Tool
- Translate / Rotate / Scale
- (Mirror Geometry)
- (Smooth (⚠adds polygons))



Creating an object

Under Maya (by hand):

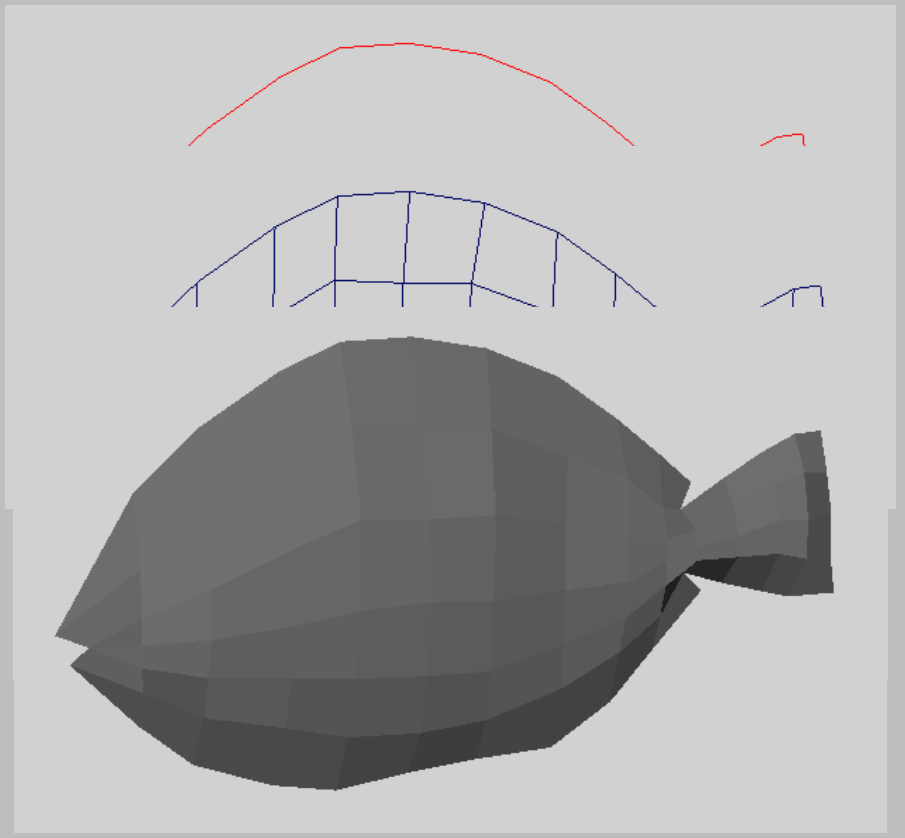
- Create Polygon Tool
- Multi-Cut Tool
- Translate / Rotate / Scale
- (Mirror Geometry)
- (Smooth (⚠adds polygons))



Creating an object

Under Maya (by hand):

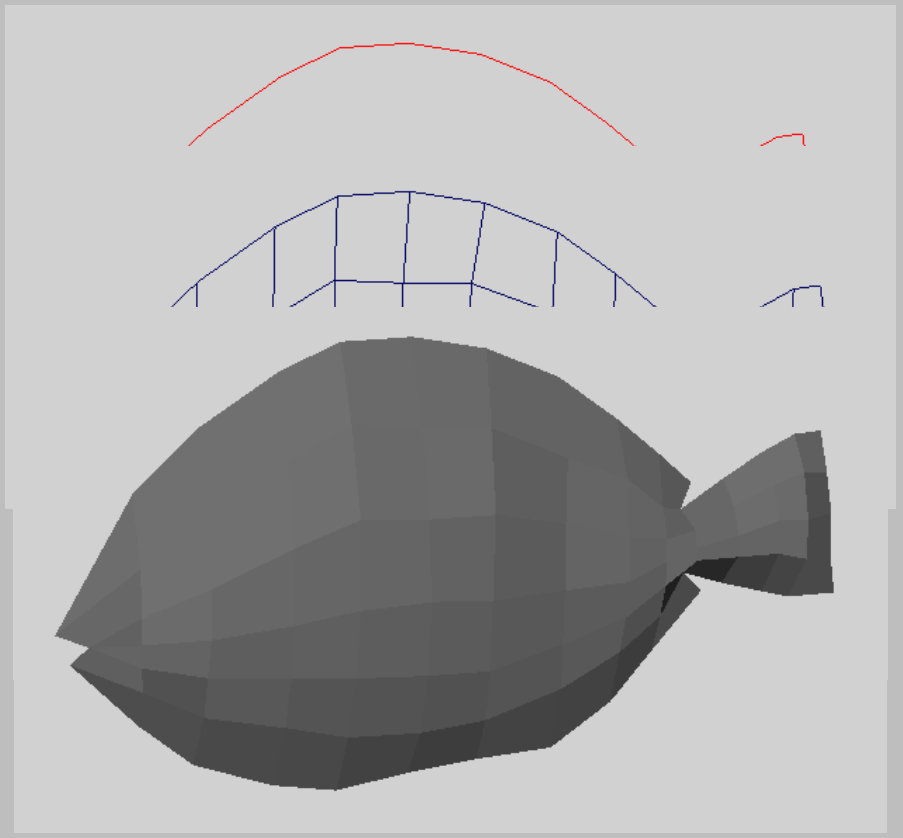
- Create Polygon Tool
- Multi-Cut Tool
- Translate / Rotate / Scale
- (Mirror Geometry)
- (Smooth (⚠adds polygons))



Creating an object

Under Maya (by hand):

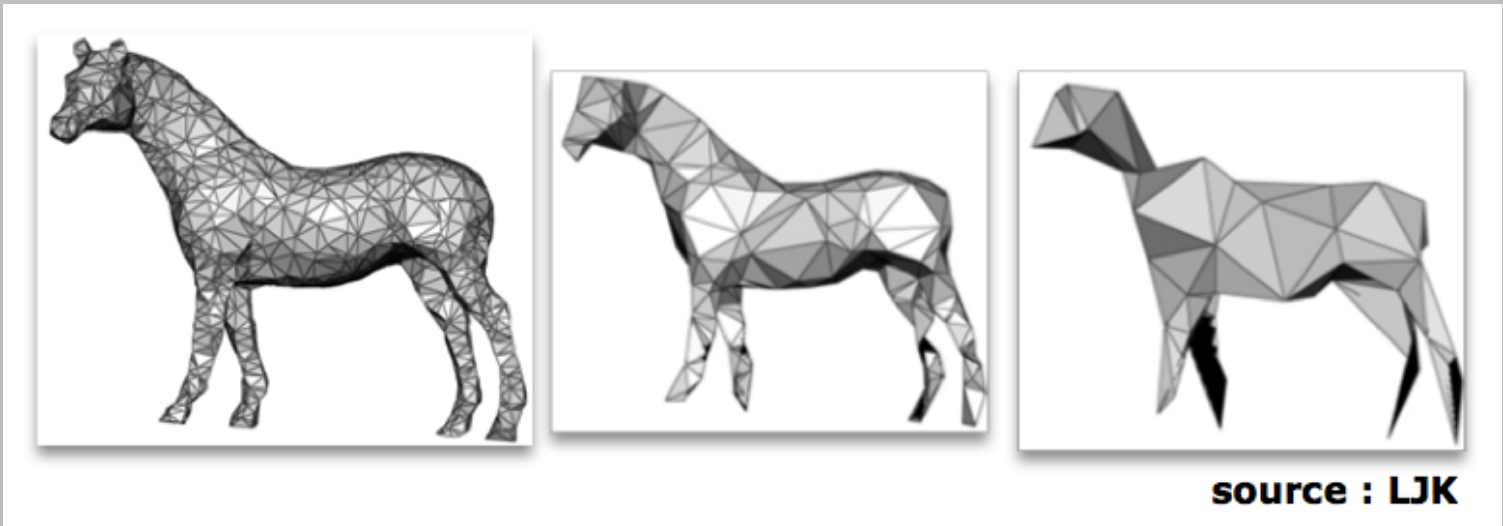
- Create Polygon Tool
- Multi-Cut Tool
- Translate / Rotate / Scale
- (Mirror Geometry)
- (Smooth (⚠adds polygons))



http://zabador.free.fr/maya/fish/fish_dory_animationFR.html

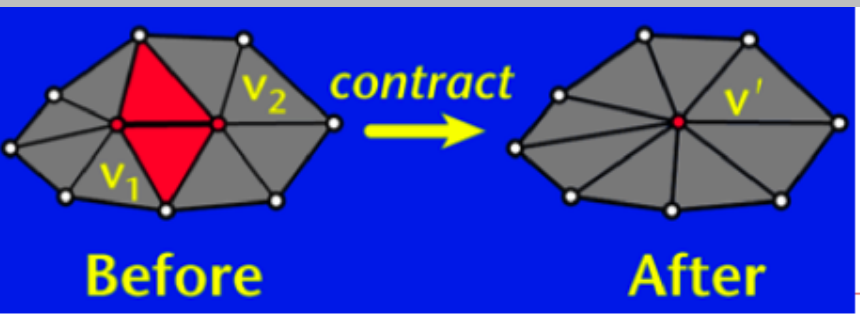
Mesh Simplification

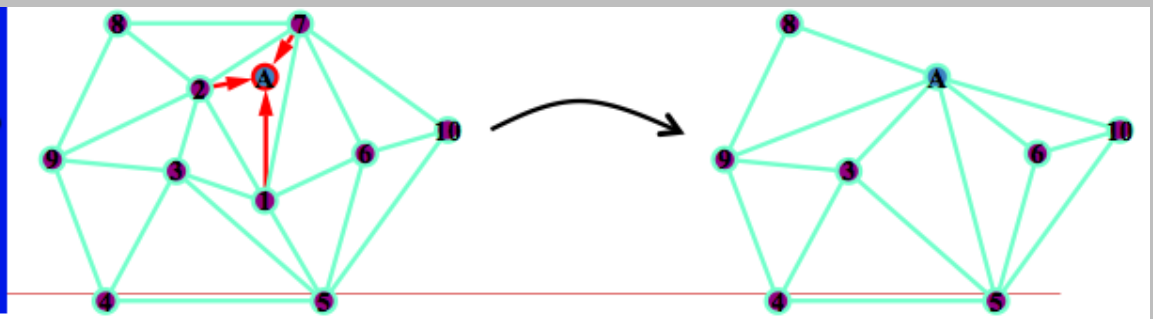
Idea: reduce the number of polygons while keeping the visual appearance



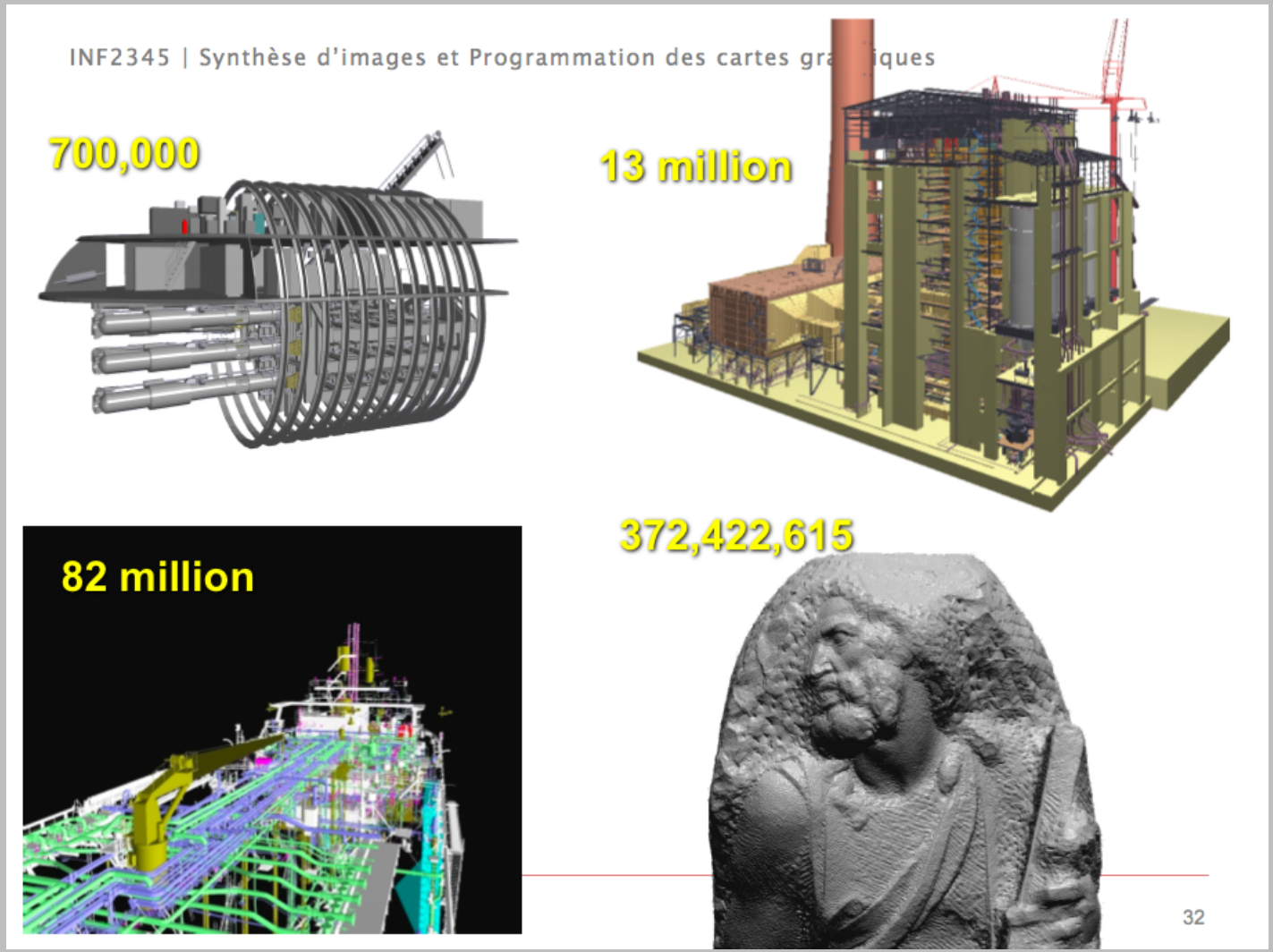
Mesh Simplification

- Various strategies
  - Edge contraction
  - Vertices removal
  - Triangles removal + re-triangulation
- Common problems
  - How to keep the shape (and topology) ?
  - How to evaluate the error induced ?



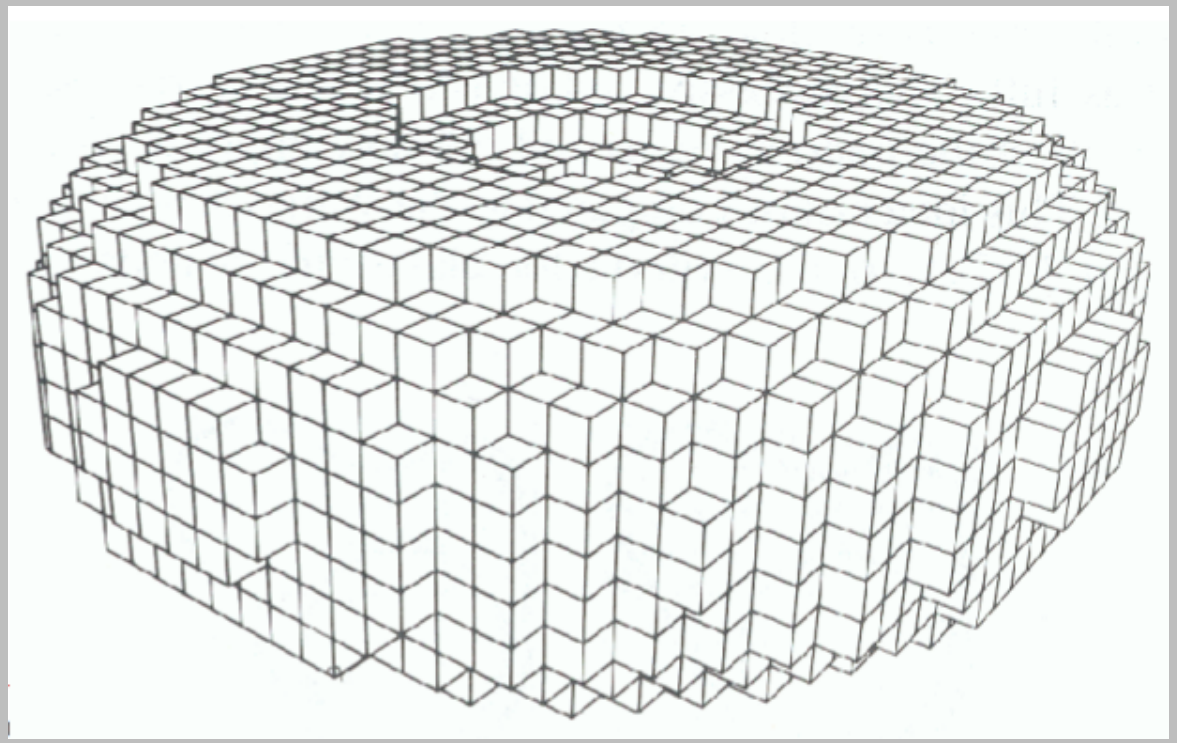


Polygonal Meshes Examples



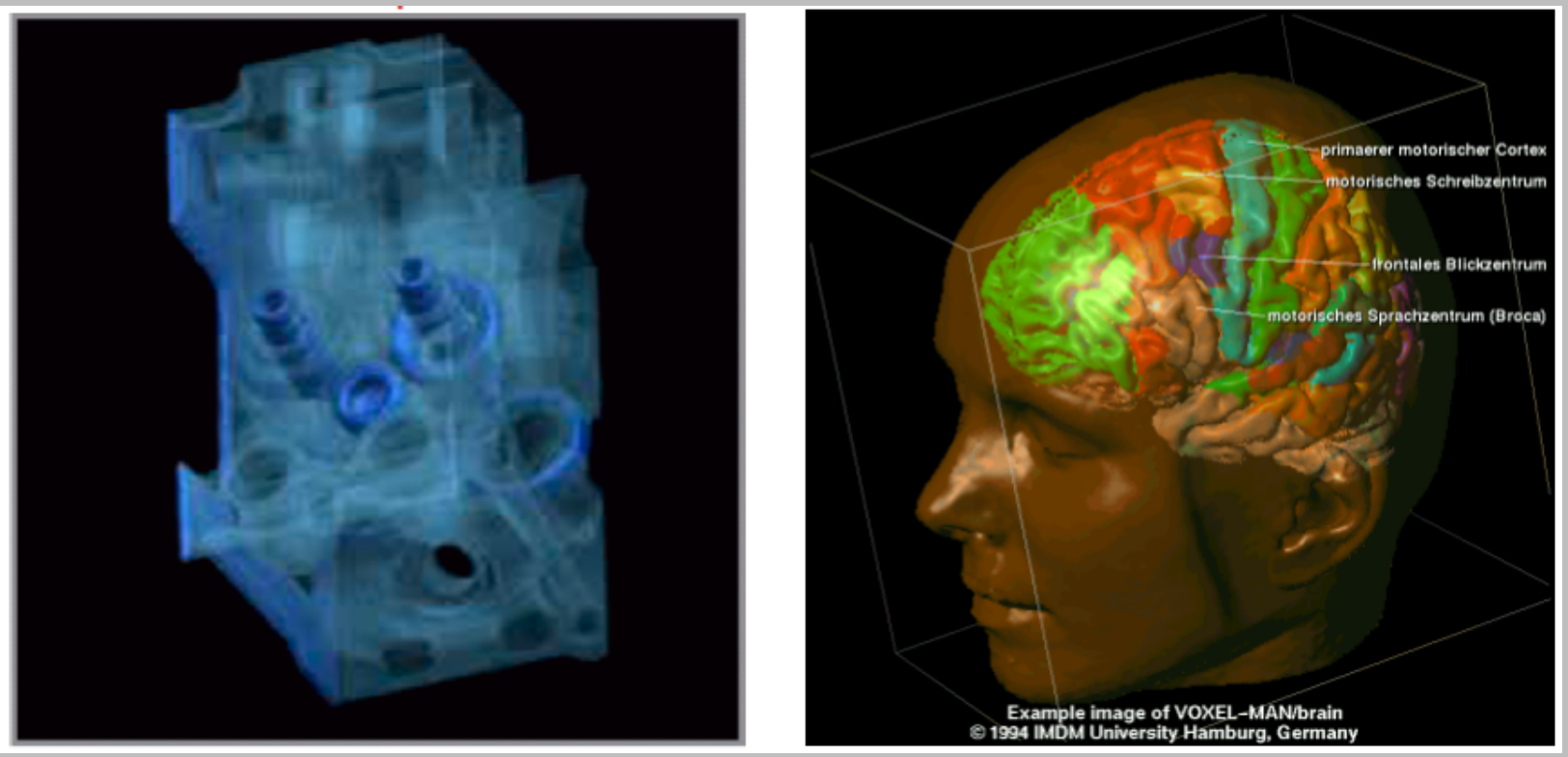
Voxels

Describe the volume of the object by (regularly) subdividing the space



Voxels

- Approximate the volume (same as meshes)
- Expensive in terms of memory
- Efficient visualization if combined with an octree

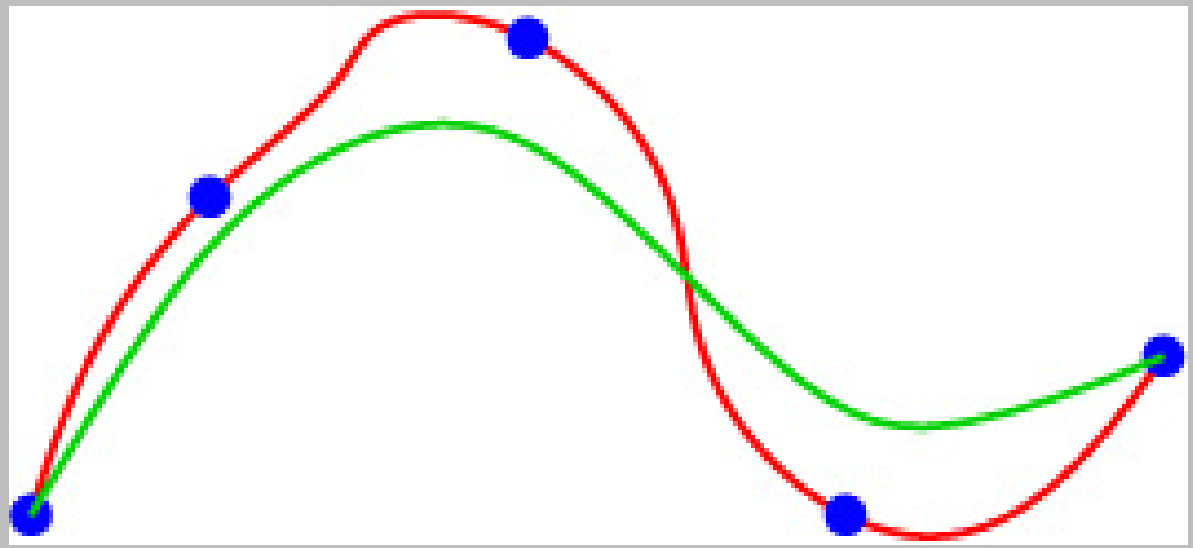


Parametric Surfaces

Parametric Curves (1D)

Defined by a few *Control Points/Vertices*
2 types

- Approximating (go around): Cardinal splines, Bezier
- Interpolating (go through): cubic B-Splines, Hermite

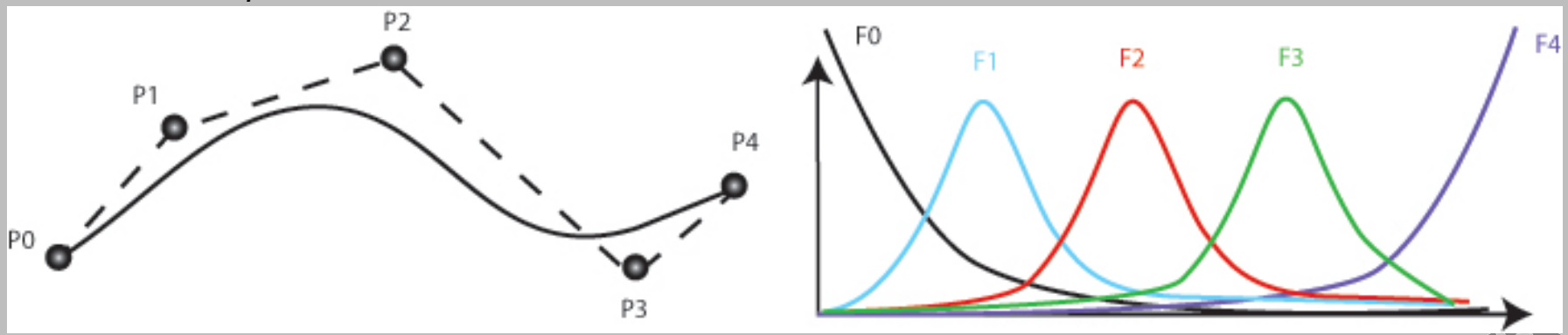


Mathematical Definition

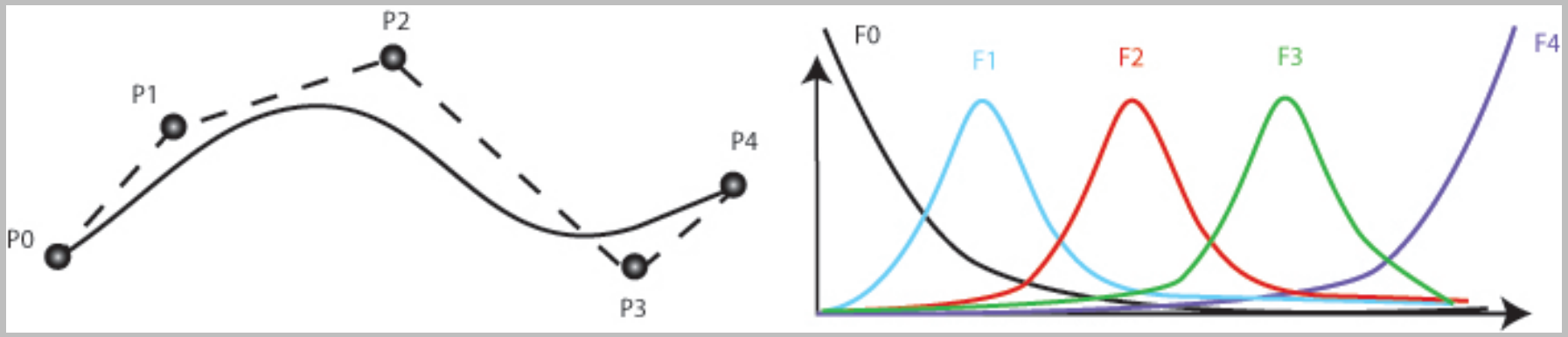
Let $\mathbb{P} = \{P_1, \ldots, P_k\}$ be a set of k points in the plane. Let $\mathbb{F} = \{F_1, \ldots, F_k\}$ be a set of k functions defined on $[0,1]$ to $\mathbb{R}$. We call *spline curve* generated by the couples (P_i, F_i), $1 \leq i \leq k$, the curve C which parametric equation is:

$$\forall t \in [0,1], C(t) = \sum_{i=1}^{k} F_i(t) P_i \tag{1}$$

Points P_i are the *control points* of C.
Functions F_i are called *influence* or *basis functions* of C.



Mathematical Definition



- P_i and P_iP_{i+1} is the *control network* of C
- P_iP_{i+1} (dashed line) is the *hull* of C

Advantages

- Shape influenced by control points (easier for modeling)
- Deformation is local
  - If one CP moves, only a portion of the curve will deform
  - To change a whole portion of the curve, only 1 CP needs to be moved
- A curve can be closed (first CP re-used) or open
- Behavior defined by the properties

Properties - Normality

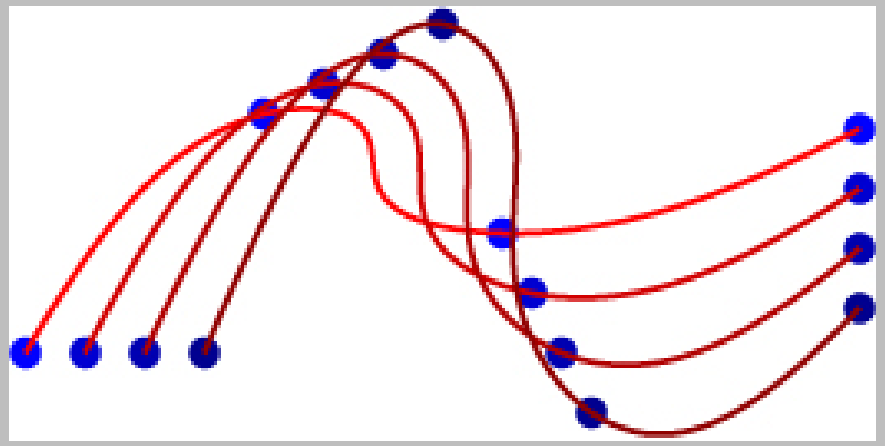
A spline curve C is *normal* if

$$\forall t \in [0,1], \sum_{i=0}^{k} F_i(t) = 1$$

- Invariant to affine transformations (affine transformation is applied to the CPs)
- Invariant to barycentric transformations (weighted mean of several splines can be computed from the weighted mean of their CPs)

Normality - Consequences

- The shape of a normal spline is independent of the frame (coordinate system) in which the CPs are expressed
- Translation, rotation and homotecy move the curve but do not change its shape
- Morphing realized by interpolation of the CPs



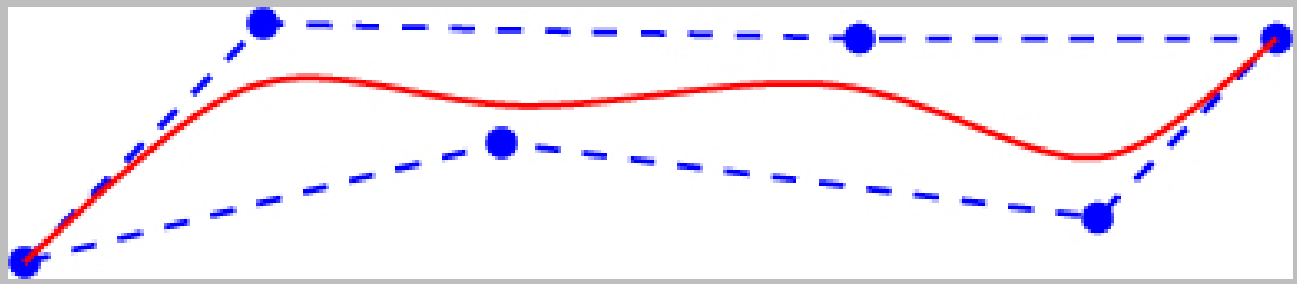
Properties - Positivity

A spline curve C is *positive* if

$$\forall t \in [0,1], \forall i = 1 \ldots k, F_i(t) \geq 0$$

A *normal* and *positive* curve is entirely enclosed in its **convex hull**

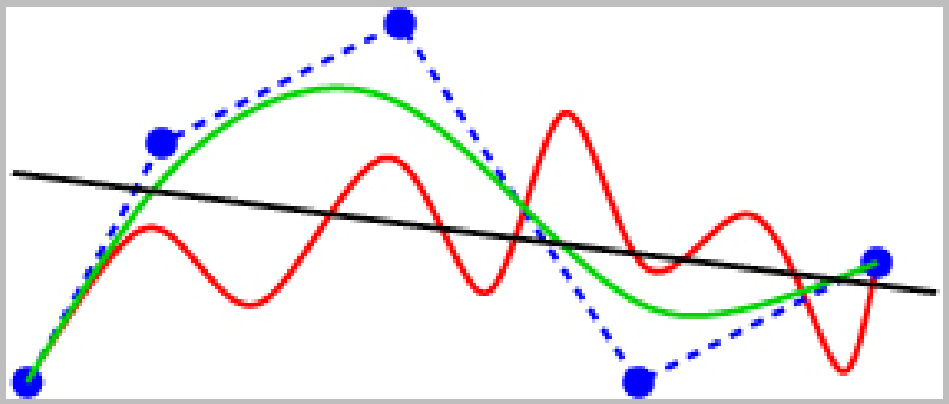
- Bounding box (collisions detection)
- Straight line if CPs are aligned



Properties - Regularity

A curve is *regular* if the number of intersections between a plane and the curve is at most the number of intersections between this plane and the control network of the curve

- Useful to control the *oscillations* of the curve



Properties - Locality

The *locality* controls the influence of the CPs on the curve

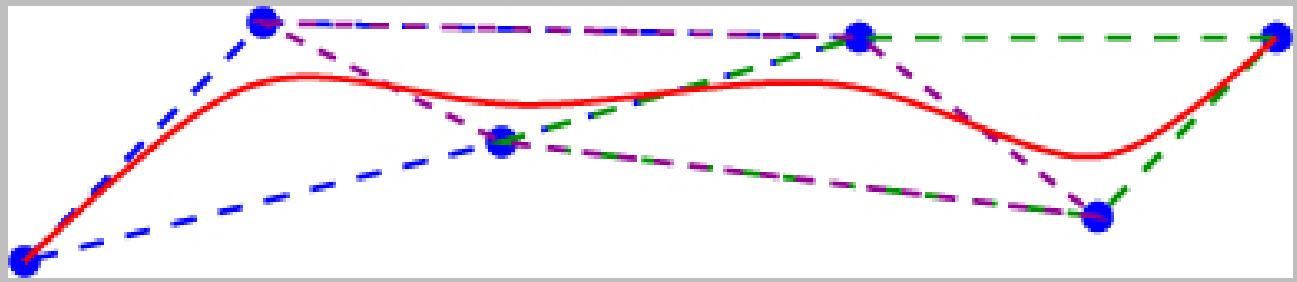
- Each F_i is defined by segment
  $F_i(t) = polynome_i(t)$ for $t \in [T_i^-, T_i^+]$, 0 otherwise
  $\forall t < T_i^-, F_i(t) = 0$ and $\forall t > T_i^+, F_i(t) = 0$
- The set of all t for which the same CPs affect $C(t)$ is a **curve segment**
- A spline curve $C(t)$ is *local of order m* if each CP affects at most m curve segments
- or... A curve segment is computed thanks to m control points

Locality - Consequences

- +/- easy modeling
- Effects +/- local
- +/- fast to compute

Properties - Strict Convex Hull

A *normal*, *positive*, *regular* and *local* spline curve is entirely enclosed in the union of the convex hulls of its control network



Properties - C Continuity

Parametric Continuity

A spline curve C has a *parametric continuity of order n* (noted as C^n) if the function $t \mapsto \frac{d^n C}{dt^n}(t)$ is defined and continuous on $[0,1]$

- C continuity
  - Easy to verify
  - Depends on the parameterization
  - Does not correspond to visual regularity
  - Useful to model **trajectories**

Properties - G Continuity

Geometric Continuity

A spline curve C has a *geometric continuity of order n* (noted as G^n) if the function $s \mapsto \frac{d^n C}{ds^n}(s)$ is defined and continuous on $[0,1]$ (s is the arc length of the curve)

- G continuity
  - Harder to verify
  - Independant on the parameterization
  - Corresponds to visual regularity

Continuity Example

$q(u) = (2u, u)$
$r(t) = (4t + 2, 2t + 1)$

Continuity Example - C Continuity

Parametric Continuity

A spline curve C has a *parametric continuity of order n* (noted as C^n) if the function $t \mapsto \frac{d^n C}{dt^n}(t)$ is defined and continuous on $[0,1]$

Continuity Example - G Continuity

Geometric Continuity

A spline curve C has a *geometric continuity of order n* (noted as G^n) if the function $s \mapsto \frac{d^n C}{ds^n}(s)$ is defined and continuous on $[0,1]$ (s is the arc length of the curve)

Interpolating Splines

- Examples
  - Cardinal Spline
  - Hermite Spline
- Advantages
  - Goes through the CPs
- Drawbacks
  - May be hard to create a smooth curve

Hermite Spline

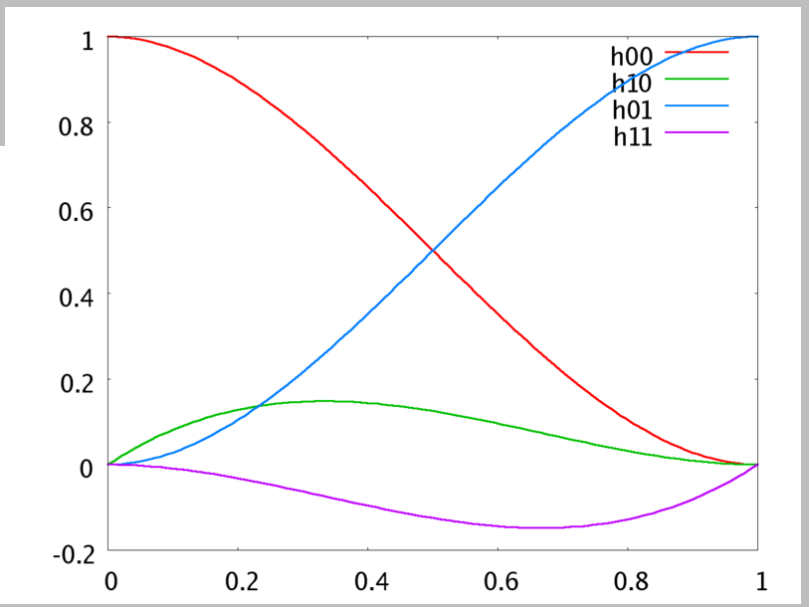
Defined by

- 2 Control Points p_0, p_1
- 2 tangents m_0, m_1

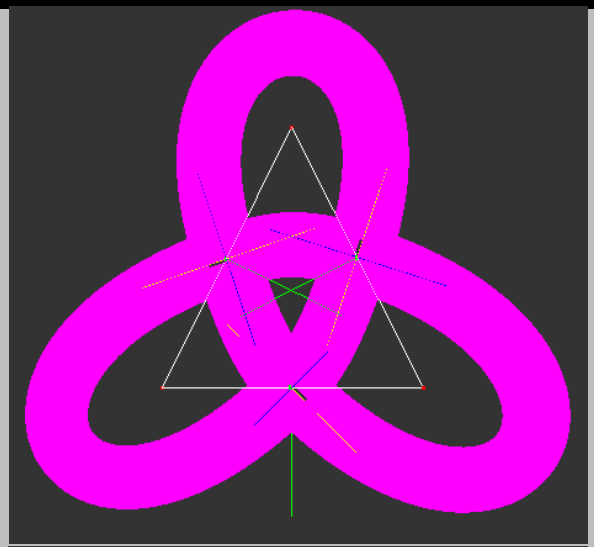
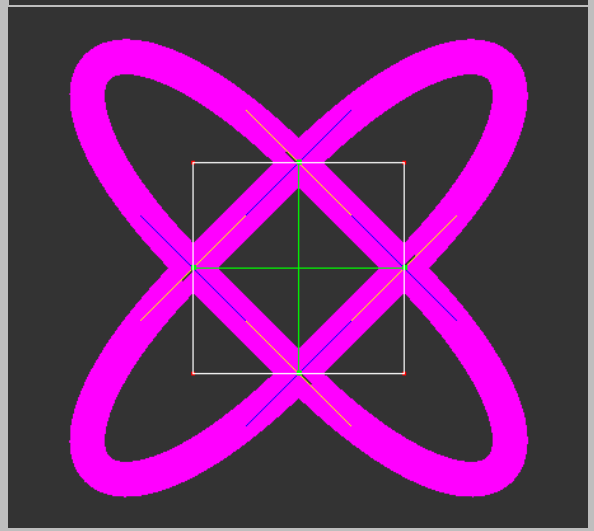
$$p(t) = (2t^3 - 3t^2 + 1)p_0 + (t^3 - 2t^2 + t)m_0 + (-2t^3 + 3t^2)p_1 + (t^3 - t^2)m_1$$

- Basis Functions

$$h_{00}(t) = 2t^3 - 3t^2 + 1$$
$$h_{10}(t) = t^3 - 2t^2 + t$$
$$h_{01}(t) = -2t^3 + 3t^2$$
$$h_{11}(t) = t^3 - t^2$$



Hermite Spline

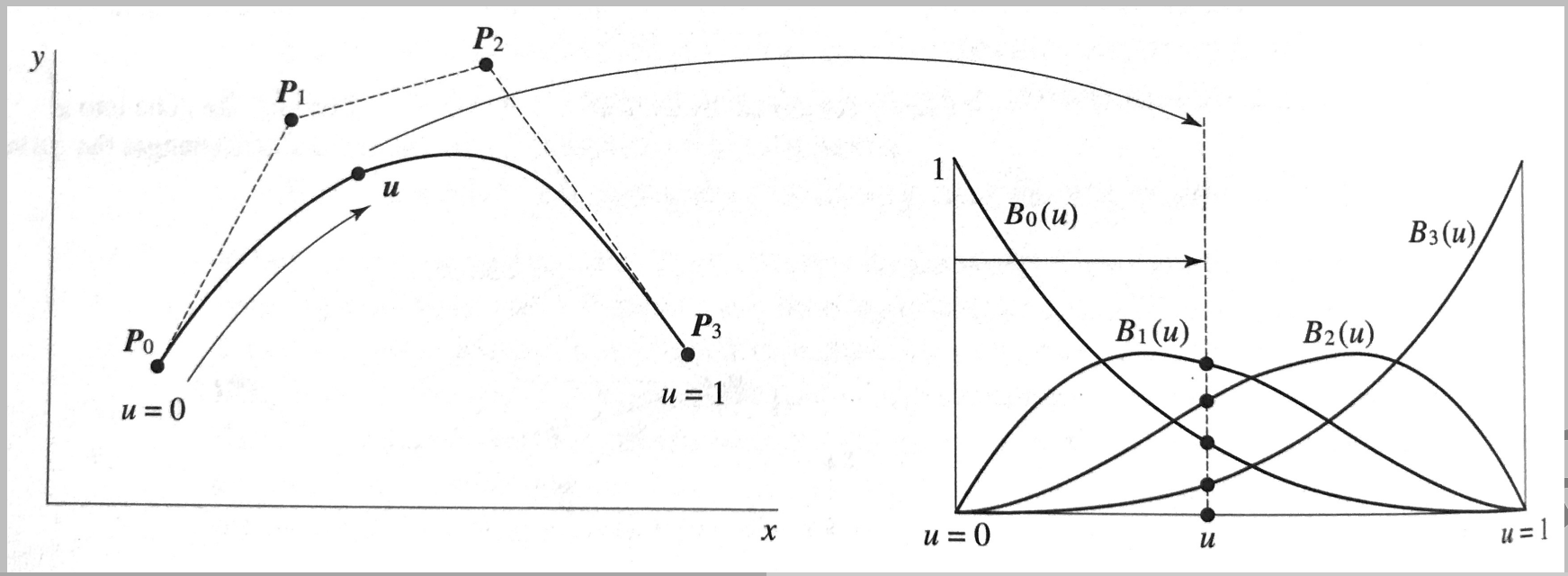




Approximating Splines

- Examples
  - B-Spline
  - Bézier Spline
  - NURBS (see Maya)
- Advantages
  - Curves are smooth
  - Weights of the NURBS
  - Use of tangents (Bézier)

Bézier

$$B_0(t) = (1-t)^3$$
$$B_1(t) = 3t(1-t)^2$$
$$B_2(t) = 3t^2(1-t)$$
$$B_3(t) = t^3$$



NURBS

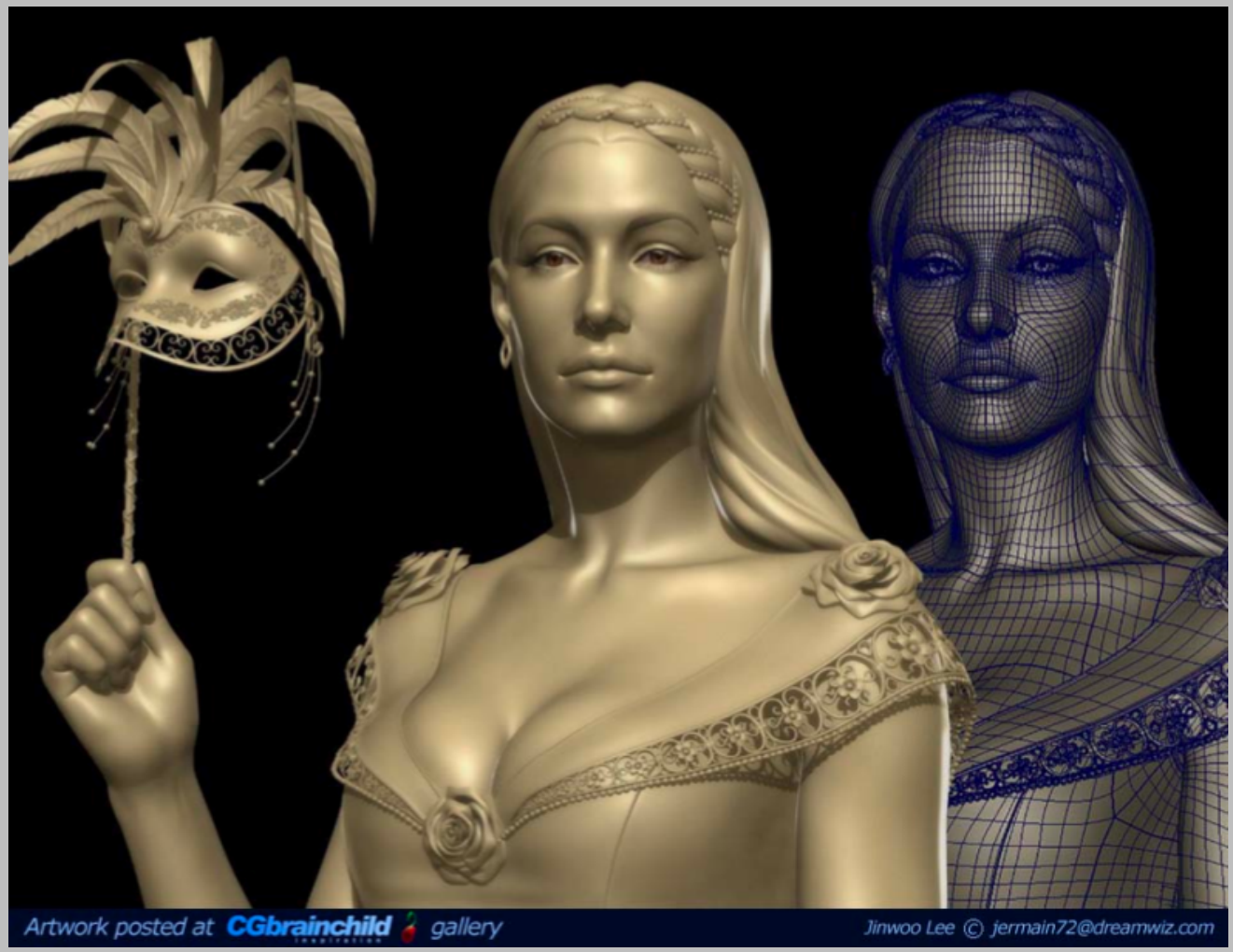
NURBS = Non-Uniform Rational B-Spline

$$C(t) = \frac{\sum_{i=0}^{n} w_i B_i(t) P_i}{\sum_{i=0}^{n} w_i B_i(t)}$$

- Non-Uniform: weights w_i
- Rational: fraction
- B-Spline: uses functions $B_i(t)$ from the B-Spline basis (usually cubic functions)

Used in Maya

NURBS Example

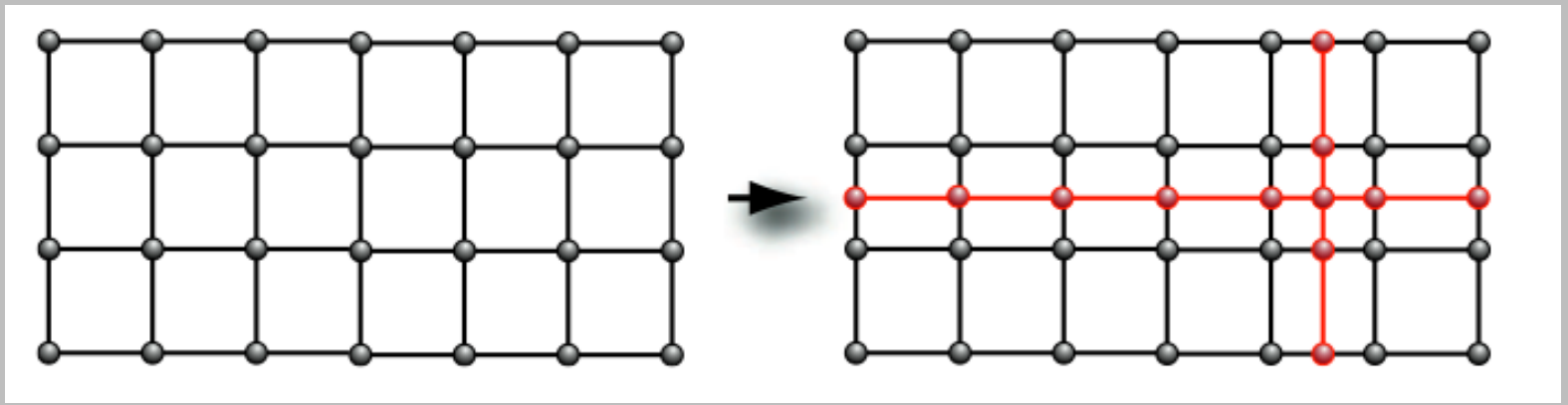


Parametric Surfaces (2D) - Patches

Combination of 2 splines (product)

$$\forall u \in [0,1], \forall v \in [0,1], S(u,v) = \sum_{i=1}^{k} \sum_{j=1}^{l} F_i(u) F_j(v) P_{ij}$$

- Problem: hard to add details

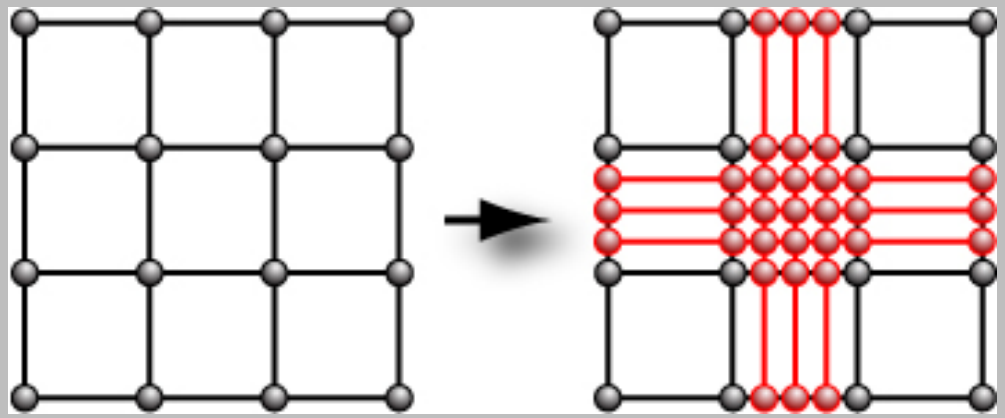


Parametric Surfaces (2D) - Patches

Combination of 2 splines (product)

$$\forall u \in [0,1], \forall v \in [0,1], S(u,v) = \sum_{i=1}^{k} \sum_{j=1}^{l} F_i(u) F_j(v) P_{ij}$$

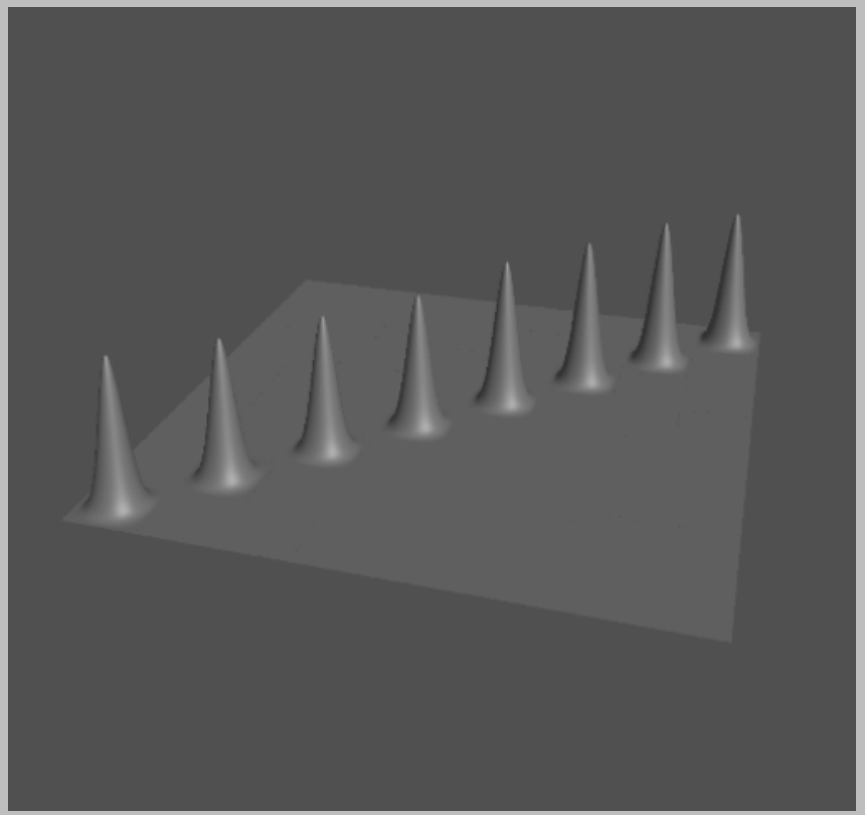
- Problem: hard to add details



Hierarchical B-Splines

Hierarchical B-Spline Refinement, Forsey et al., SIGGRAPH 1988

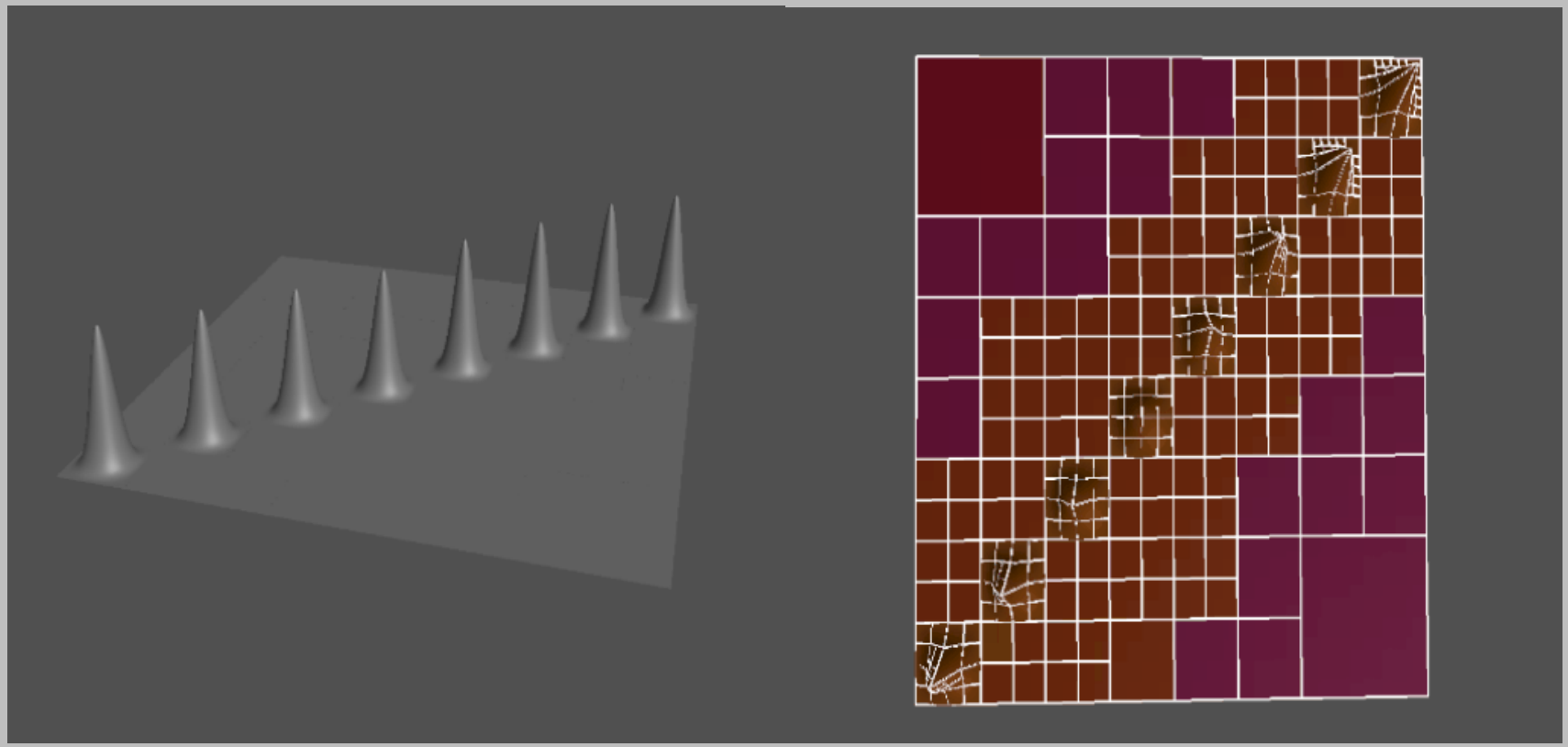
Worst case !



Hierarchical B-Splines

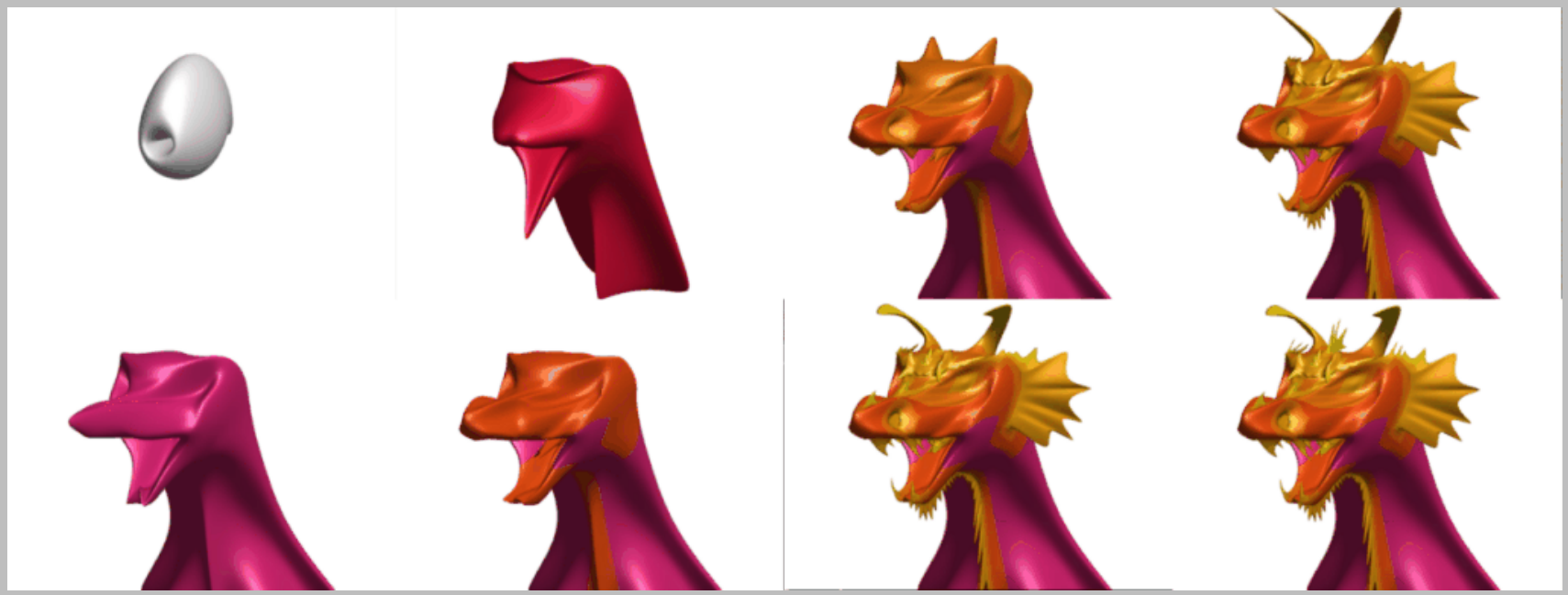
Hierarchical B-Spline Refinement, Forsey et al., SIGGRAPH 1988

Worst case !



Hierarchical B-Splines

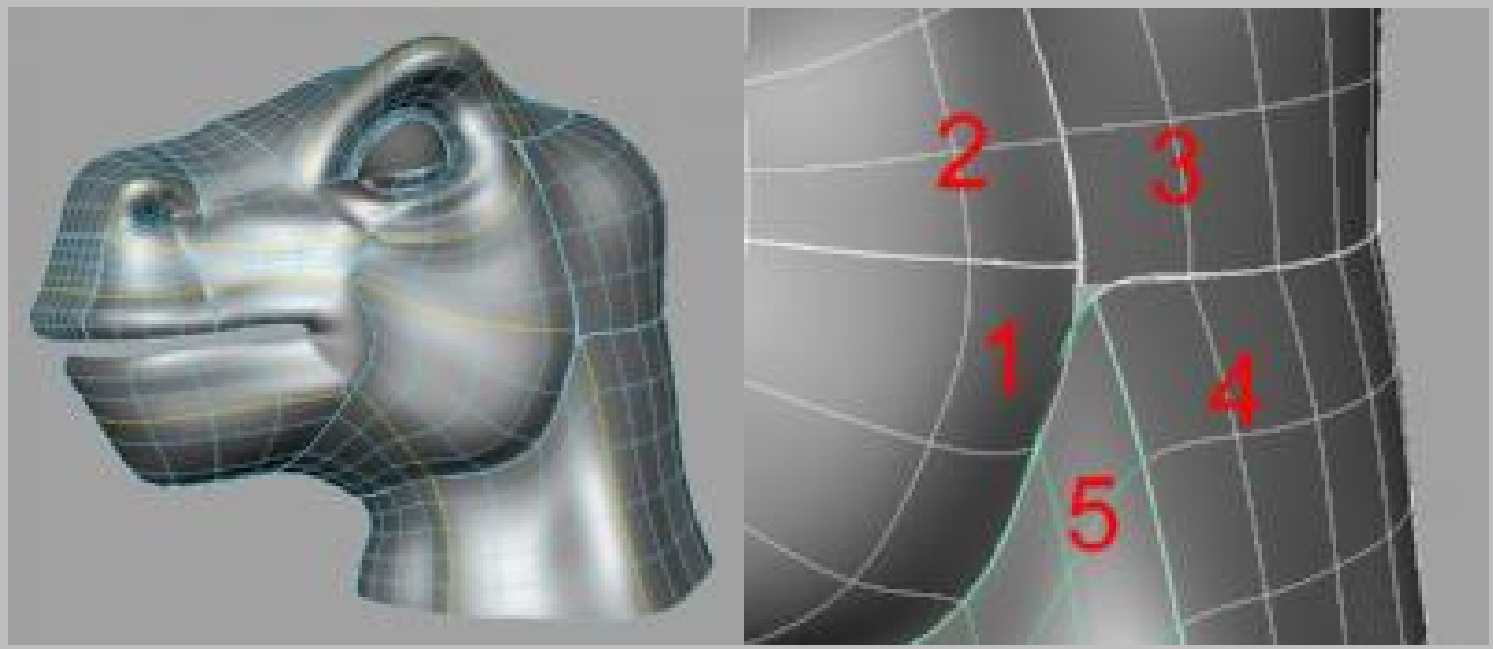
Hierarchical B-Spline Refinement, Forsey et al., SIGGRAPH 1988



Patch Stitching

Aim: join several patches together

- Each patch has its own topology
- Edges need to be connected



Subdivision Surfaces

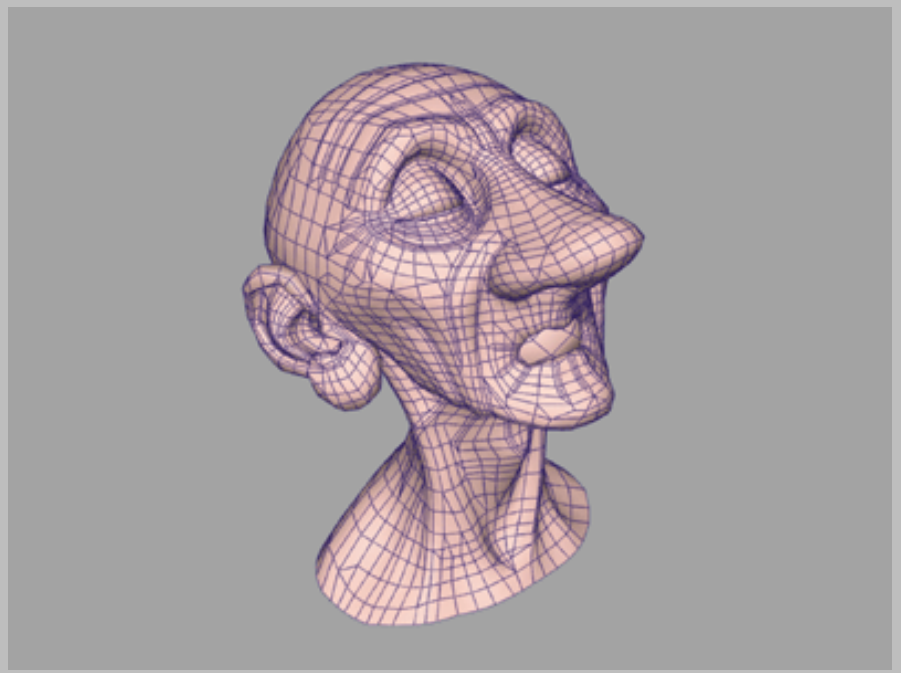
Idea:

- Create a model with only a few polygons
- Subdivide the polygons to obtain a surface with more details

Subdivision Surfaces

Idea:

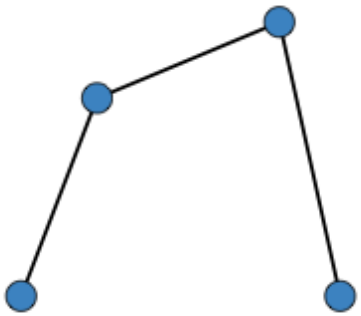
- Create a model with only a few pol
- Subdivide the polygons to obtain a



Subdivision Curves

Principle

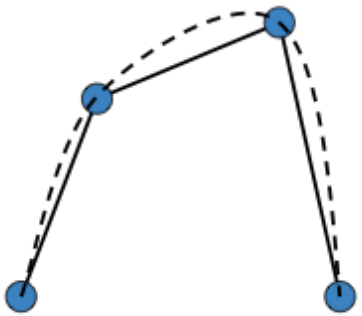
Approximate a spline curve as the limit of a recursive process of subdividing a polyline



Subdivision Curves

Principle

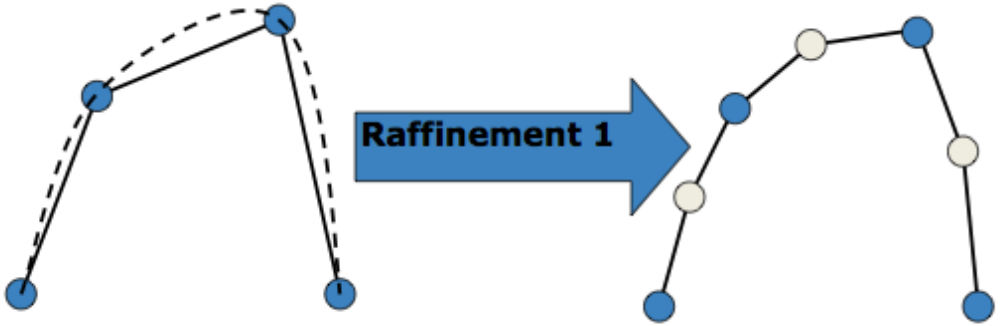
Approximate a spline curve as the limit of a recursive process of subdividing a polyline



Subdivision Curves

Principle

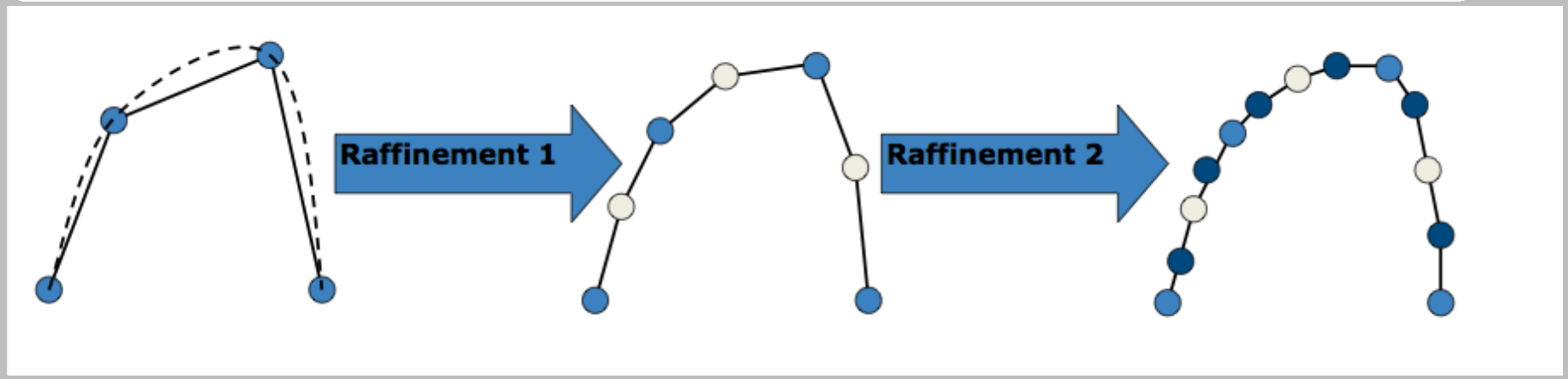
Approximate a spline curve as the limit of a recursive process of subdividing a polyline



Subdivision Curves

Principle

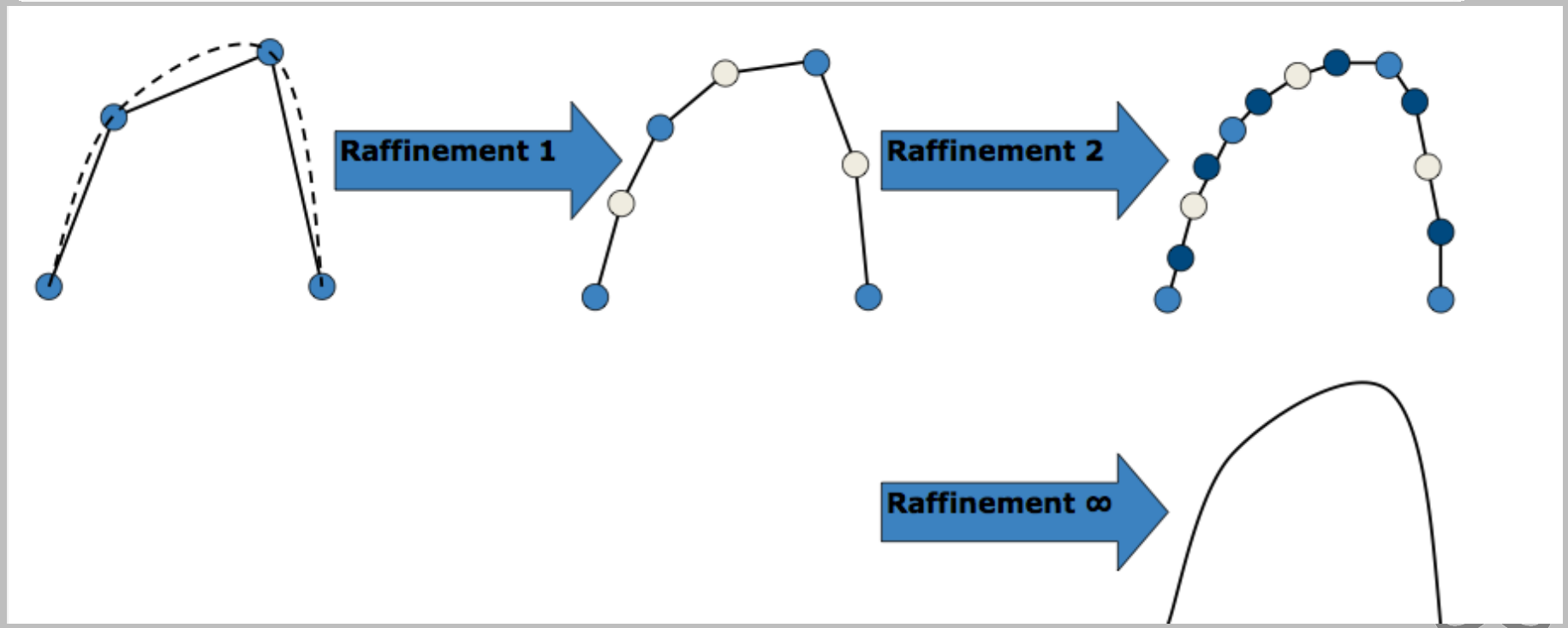
Approximate a spline curve as the limit of a recursive process of subdividing a polyline



Subdivision Curves

Principle

Approximate a spline curve as the limit of a recursive process of subdividing a polyline

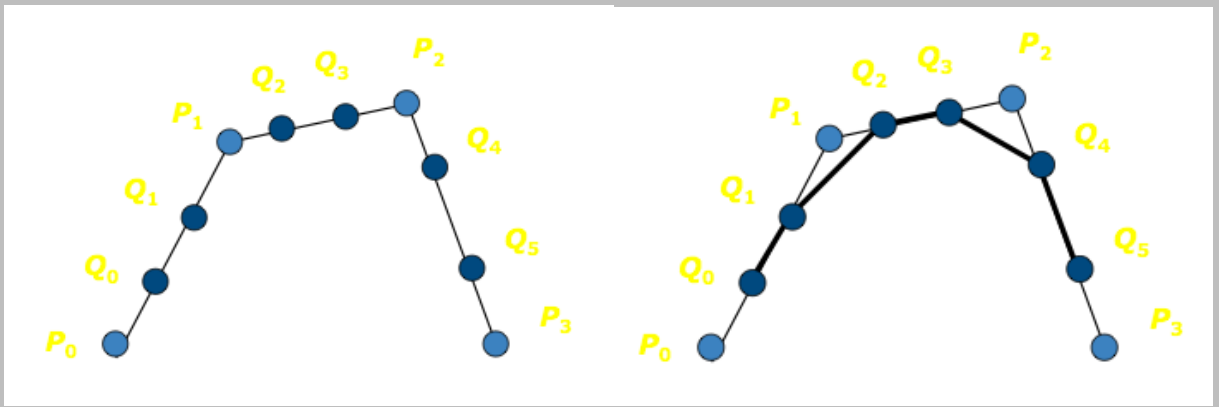


Subdivision Curves
Chaikin Algorithm (1974)

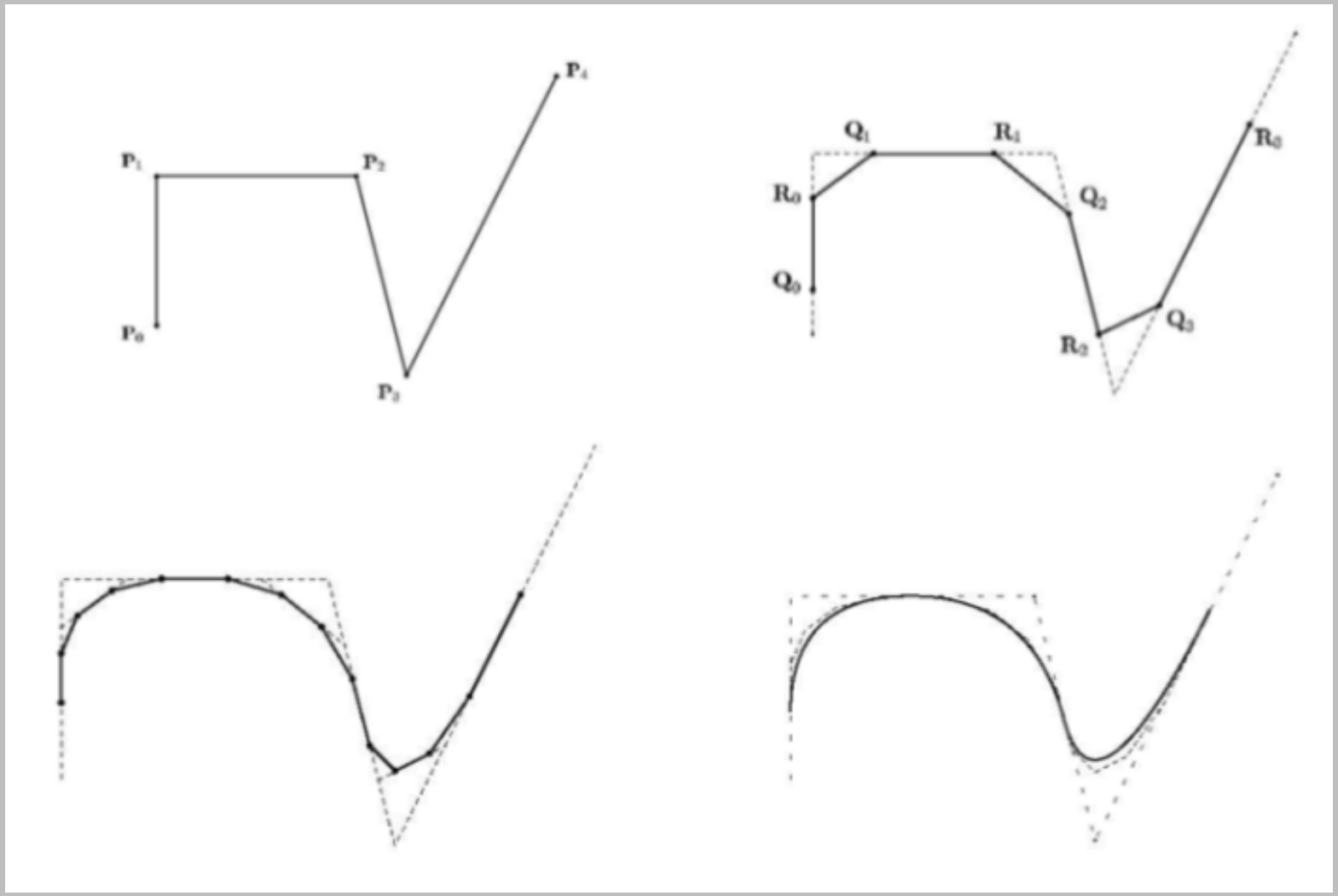
- Based on *corner cutting*
- End points are preserved
- Subdivision scheme (uniform quadric B-Spline)

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

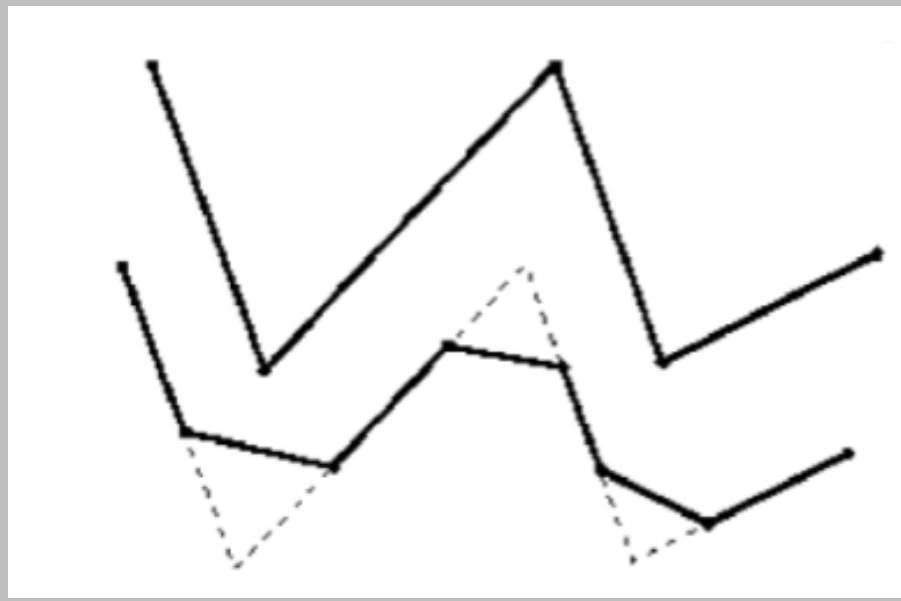
$$R_i = \frac{1}{4}P_i + \frac{3}{4}P_{i+1}$$

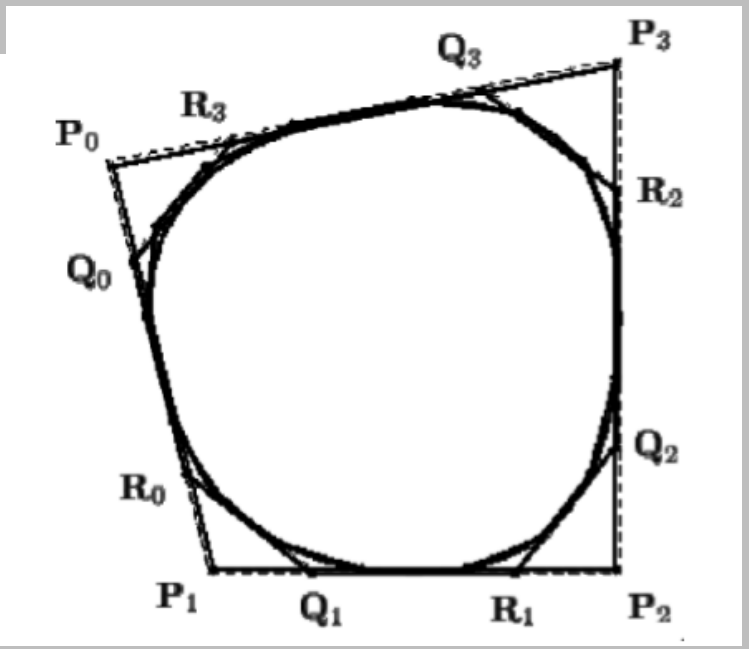


Subdivision Curves
Chaikin Algorithm (1974)

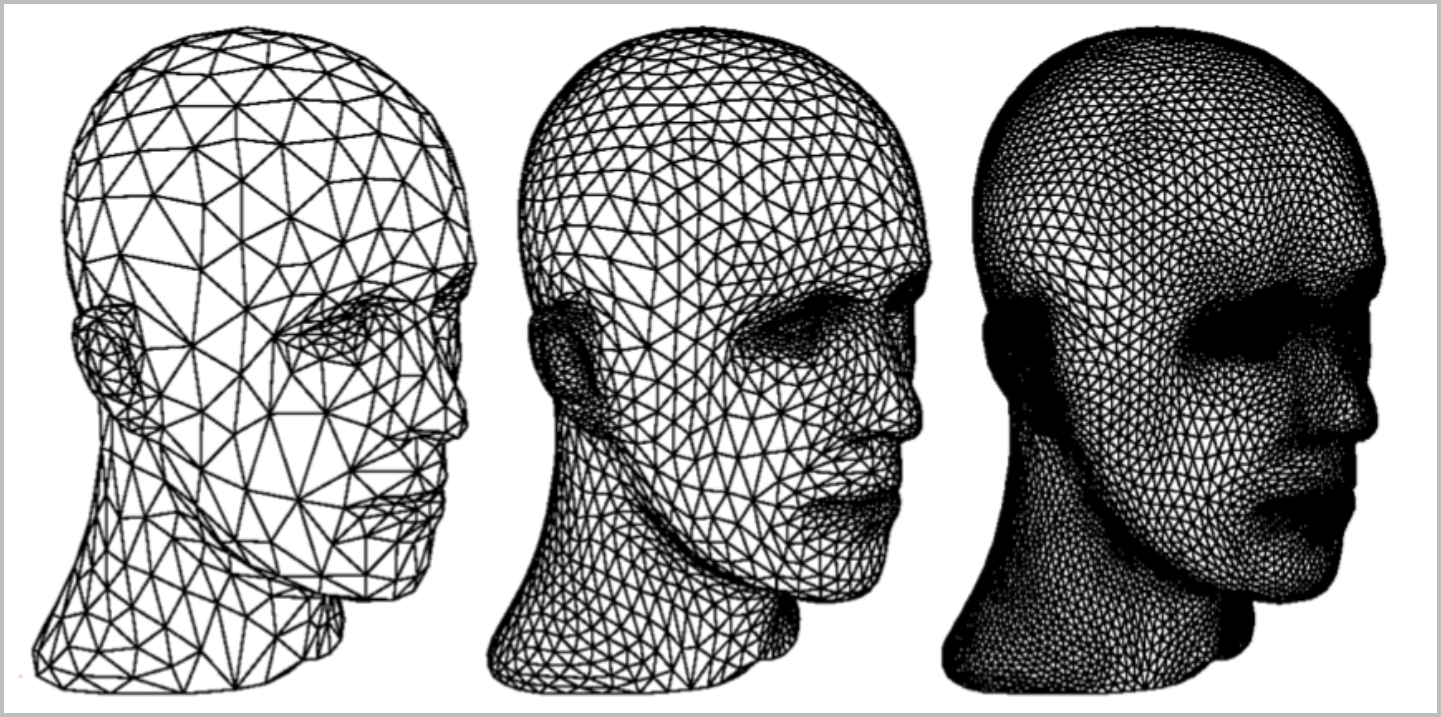


Subdivision Curves
Chaikin Algorithm (1974)





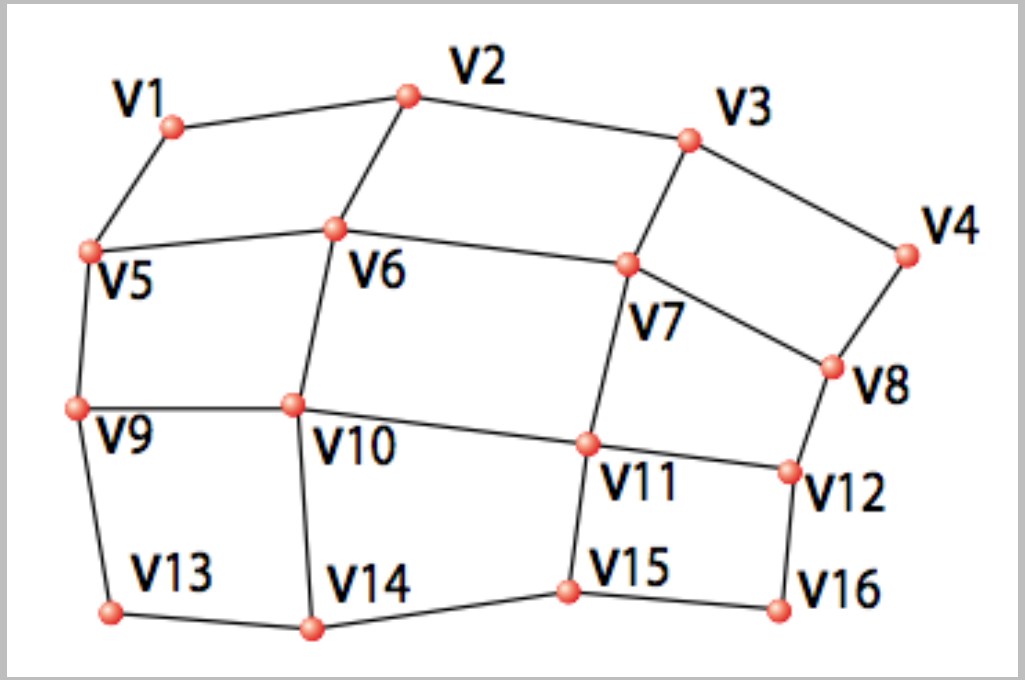
Subdivision Surfaces



https://www.khanacademy.org/partner-content/pixar/modeling-character/modeling-subdivision/v/cm-start

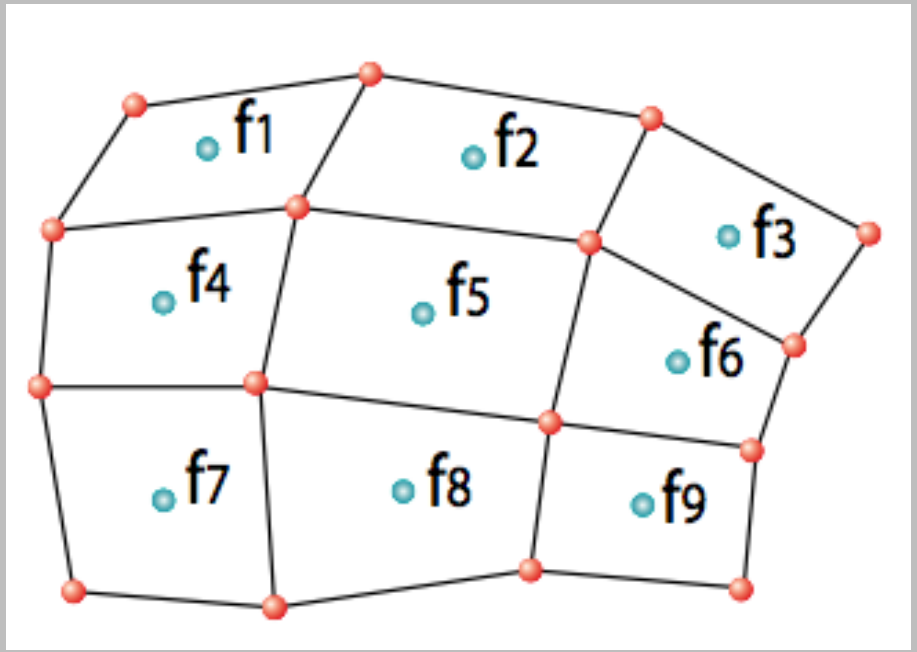
Subdivision Process
Example of Catmull-Clark (1978)

1. Each vertex is labelled v_i



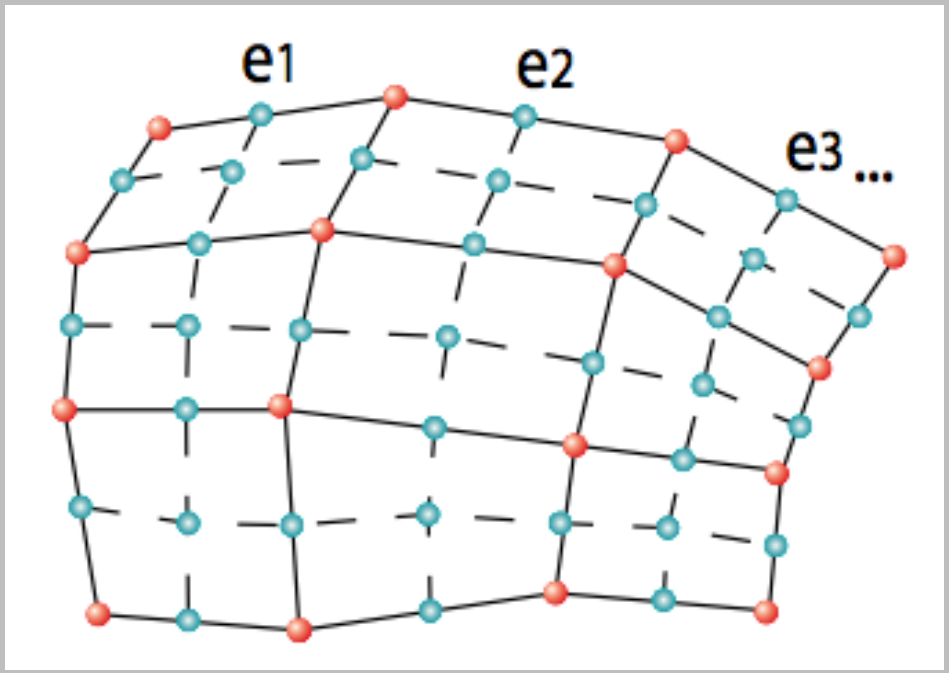
Subdivision Process
Example of Catmull-Clark (1978)

1. Each vertex is labelled v_i
2. Add f_i in the middle of each face



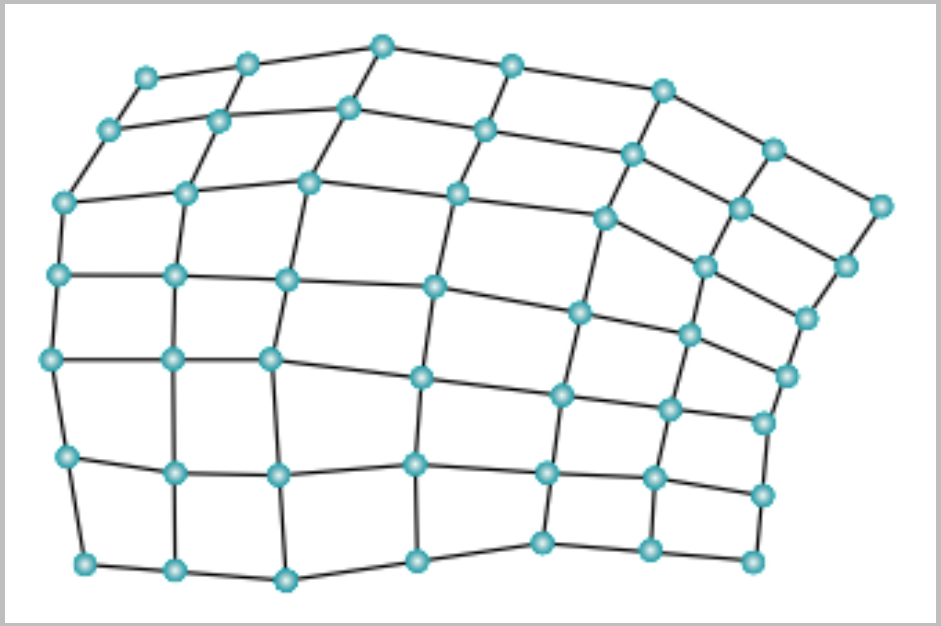
Subdivision Process
Example of Catmull-Clark (1978)

1. Each vertex is labelled v_i
2. Add f_i in the middle of each face
3. Cut each edge in two with e_i = avg (center of edge + f_i of the 2 faces sharing that edge)



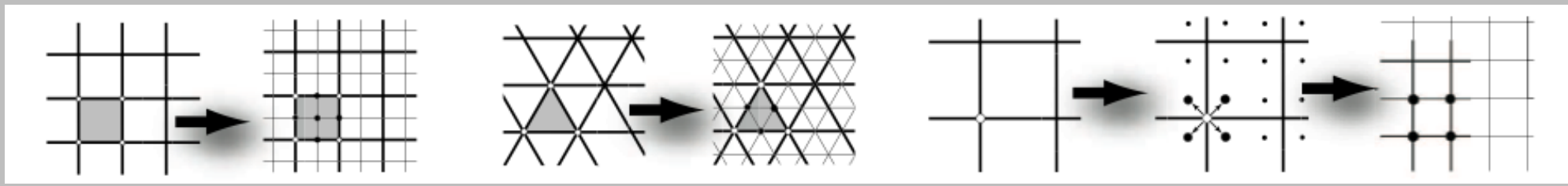
Subdivision Process
Example of Catmull-Clark (1978)

1. Each vertex is labelled v_i
2. Add f_i in the middle of each face
3. Cut each edge in two
4. Move v_i to $v_i' = (Q + 2R + S)/4$



Subdivision Algorithms

- Different types of algorithms
  - Triangles or Quads
  - Splitting Vertices or Edges
  - Interpolating or Approximating

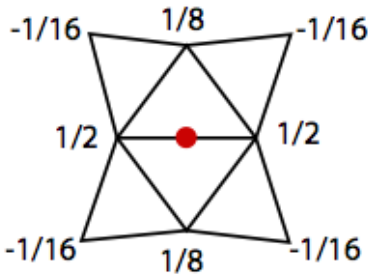
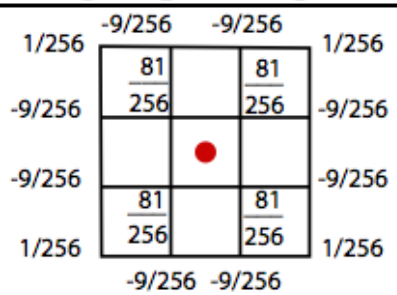
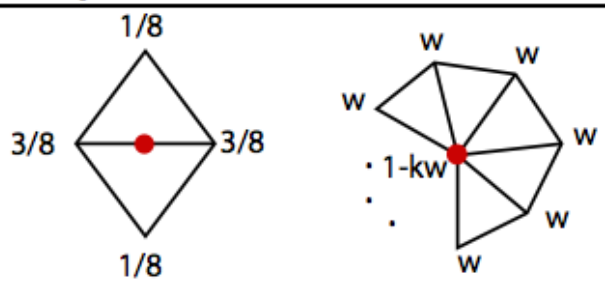
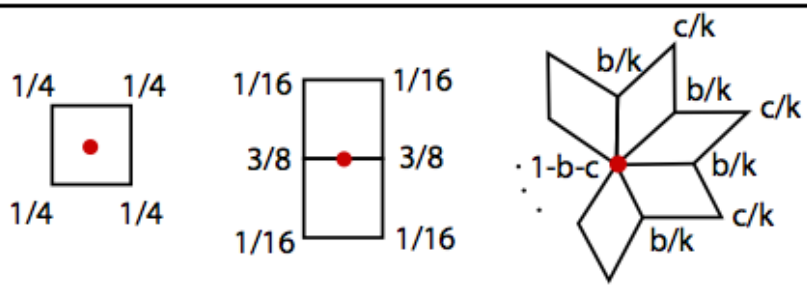


- Butterfly (interpolating, triangles)
- Kobbelt (interpolating, quads)
- Loop (approximating, triangles)
- Catmull-Clark (approximating, quads)

Subdivision Masks

Algorithm summarized through a mask showing

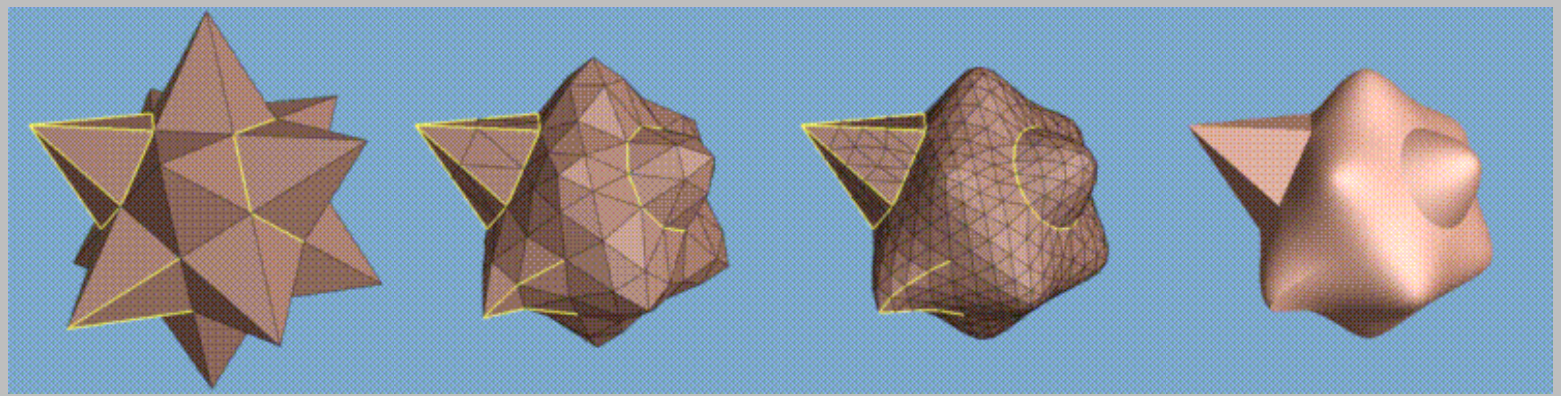
- General case
- Extraordinary vertices

	Maillages triangulaires	Maillages quadrangulaires
Interpolation	 Butterfly modifié [DLG90, ZSS96] (C^1)	 Kobbelt [Kob96](C^1)
Approximation	 Loop [Loo87] (C^2)	 Catmull-Clark [CC78] (C^2)

Subdivision Surfaces

Advantages:

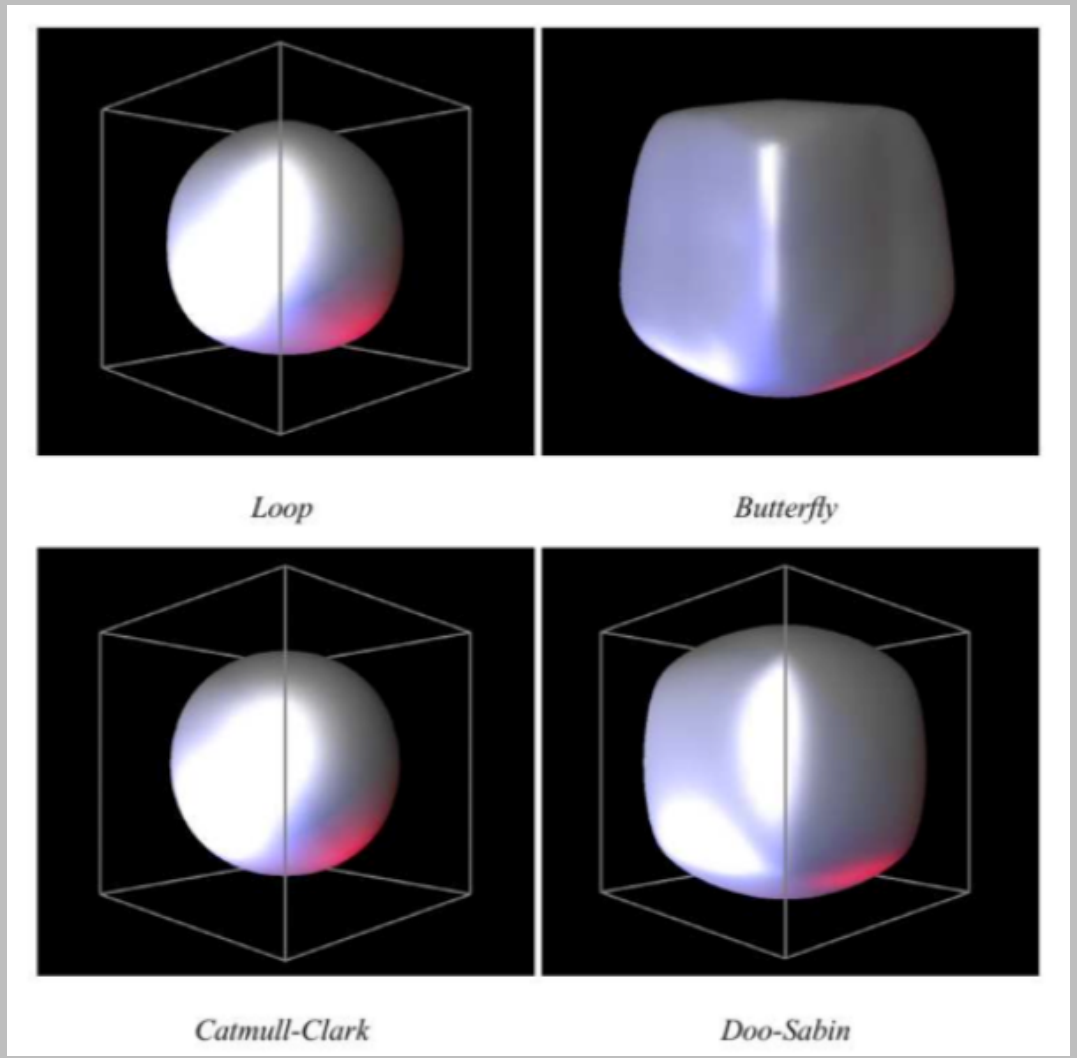
- Simple implementation
- Arbitrary topology
- Animation of the coarse mesh
- Local control of the surface continuity



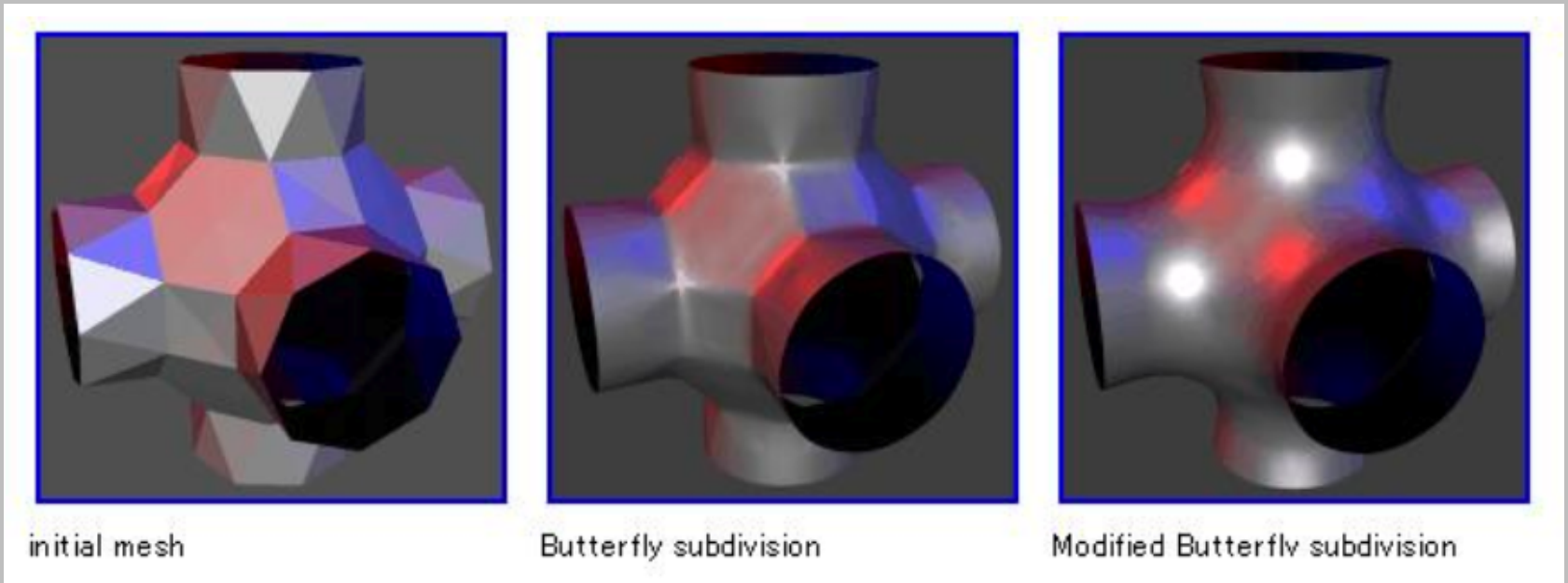
Drawbacks:

- Need to know the connectivity (problem with polygon soup)
- The smoother, the more memory consumption and rendering time !

Subdivision Surfaces
Comparison on a cube

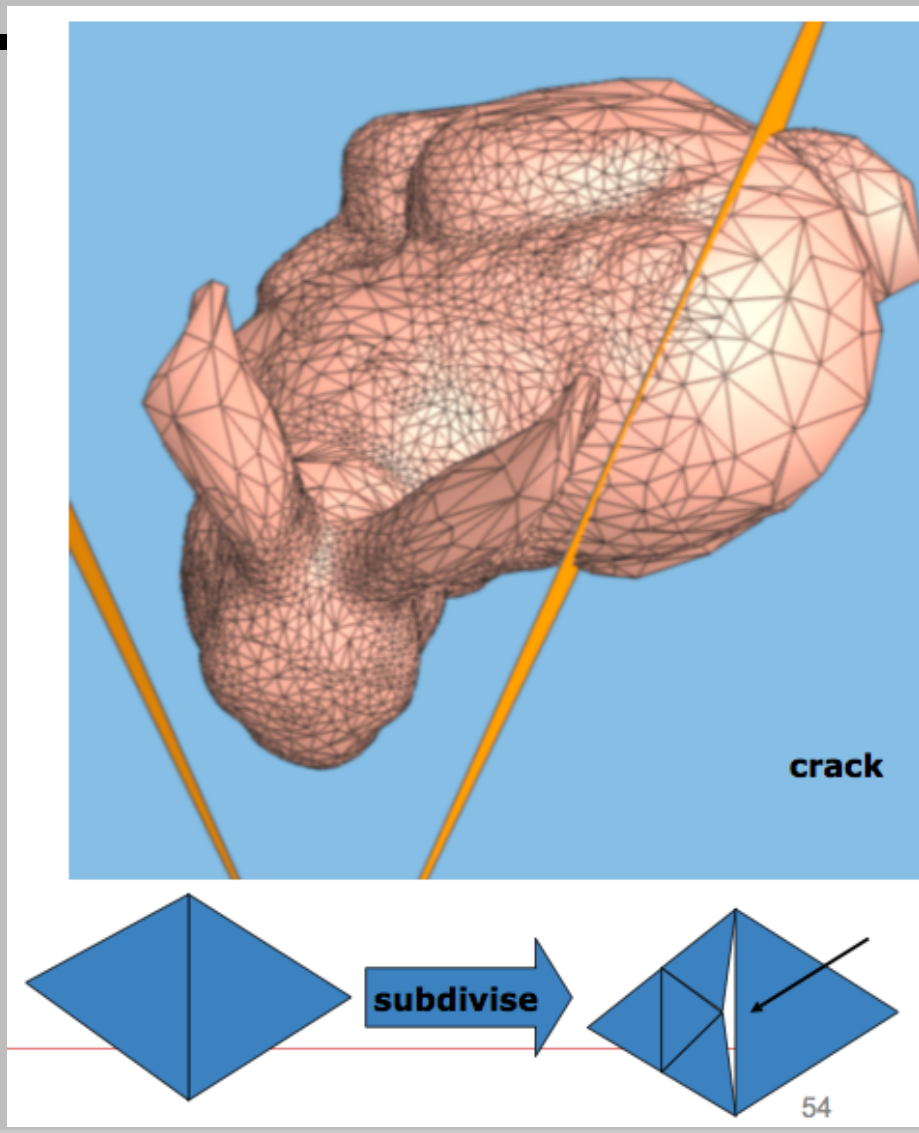


Subdivision Surfaces
Butterfly (1990) and Modified Butterfly (1996)



Adaptive Subdivision

- Non-uniform subdivision
  - Mesh curvature
  - Screen resolution
  - Viewpoint (distance, silhouette)
  - Stiffness / Deformation zones
- Problem: cracks appear if we "simply" do not subdivide some polygons



Implicit Surfaces

Definition

An implicit surface is a collection of points $p(x,y,z)$ that satisfy $F(p) = d$

- Commun functions: spheres, ellipsoids

$$f(x,y,z) = \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2}$$

- Good for volume preservation $V = \frac{4}{3}\pi * a * b * c$
- The equation $f(x,y,z) \geq d$ defines the volume
  - Collision detection / raytracing
  - Is a point inside or outside ?

Implicit Surfaces

Definition

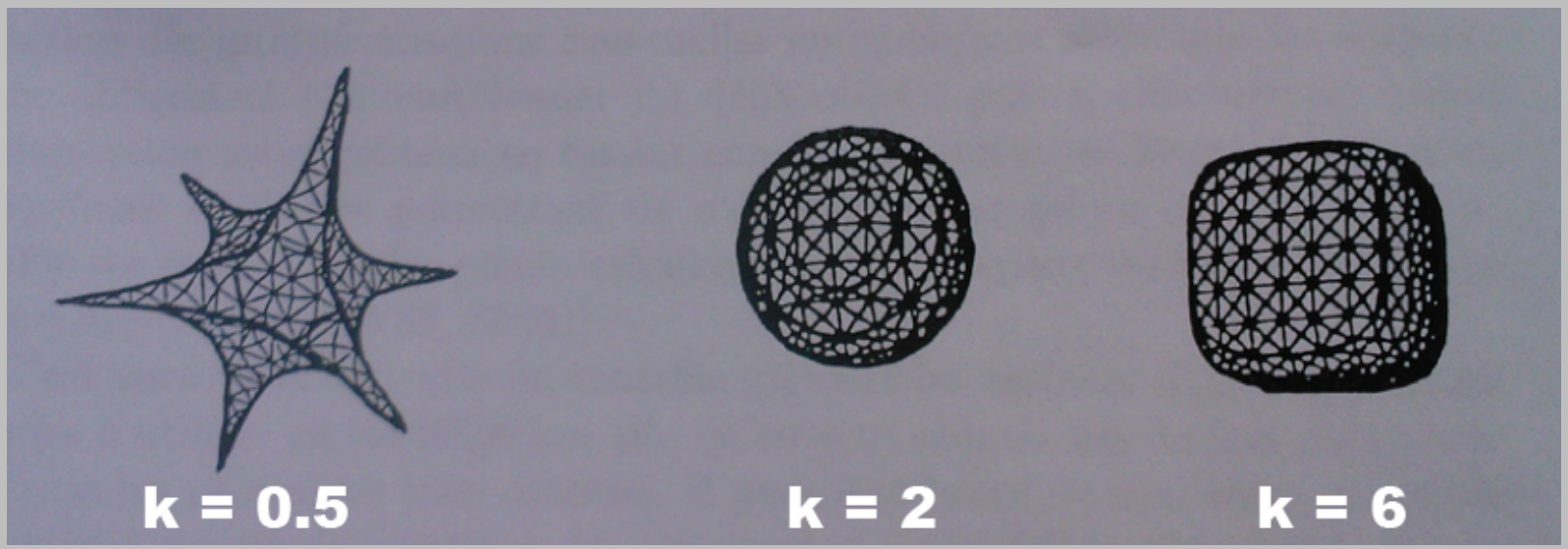
An implicit surface is a collection of points $p(x, y, z)$ that satisfy $F(p) = d$

- CSG operations
  - union: $\max(F_1, F_2)$
  - intersection: $\min(F_1, F_2)$
  - blending: $F_1 + F_2$

Implicit Surfaces

- Super-ellipsoids

$$f(x,y,z) = \frac{x^k}{a^k} + \frac{y^k}{b^k} + \frac{z^k}{c^k}$$



Implicit Surfaces - Deformation

Generalized Implicit Functions for Computer Graphics, Sclaroff et al., SIGGRAPH'91

- Let R be a rotation matrix
- Let t be a translation vector

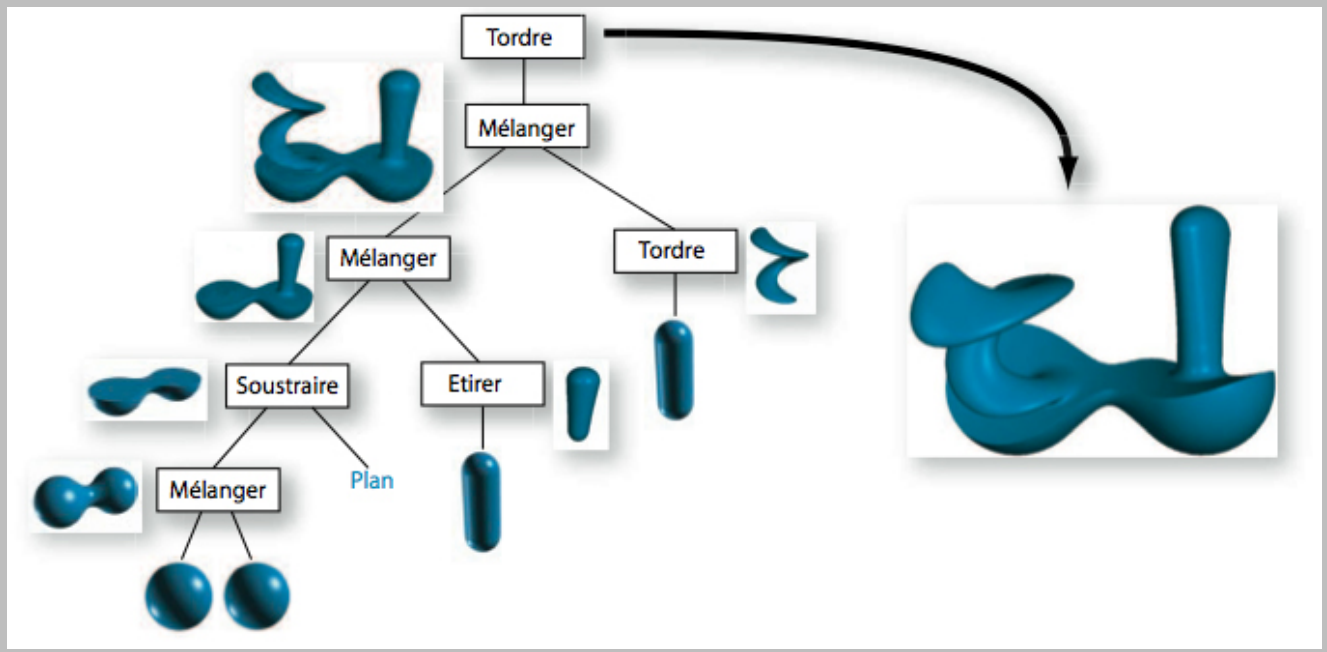
$$p' = Rp + t$$

$$f(R^{-1}(p' - t)) = d$$

Implicit Surfaces - The Blob Tree

Extending The CSG Tree. (...), Wyvill et al., Computer Graphics Forum 99

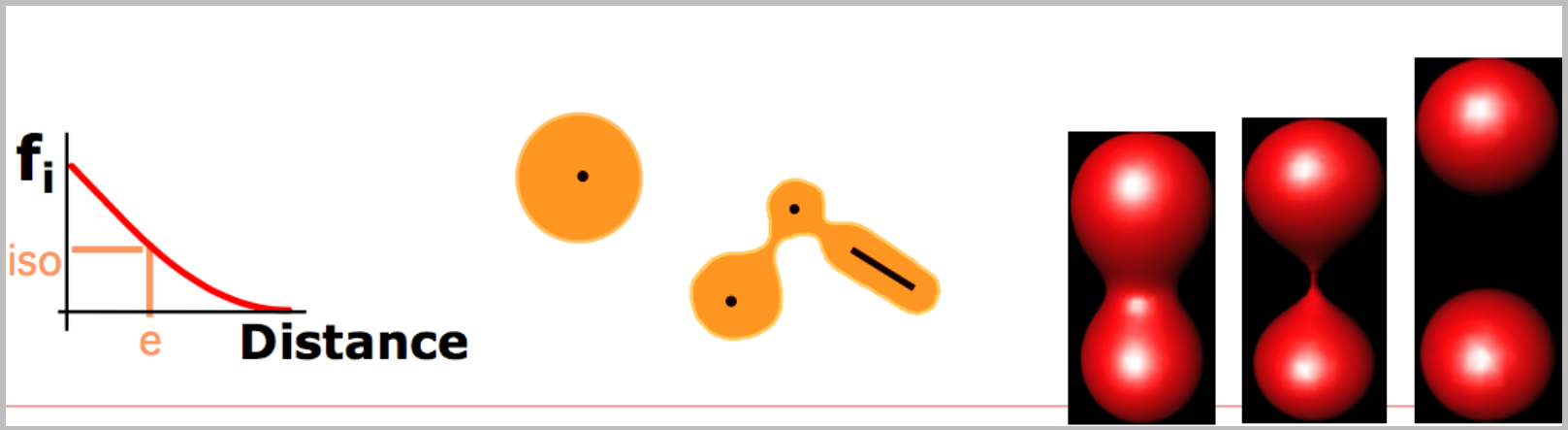
To create various shapes, it is possible to add, blend, substract and twist the primitive functions



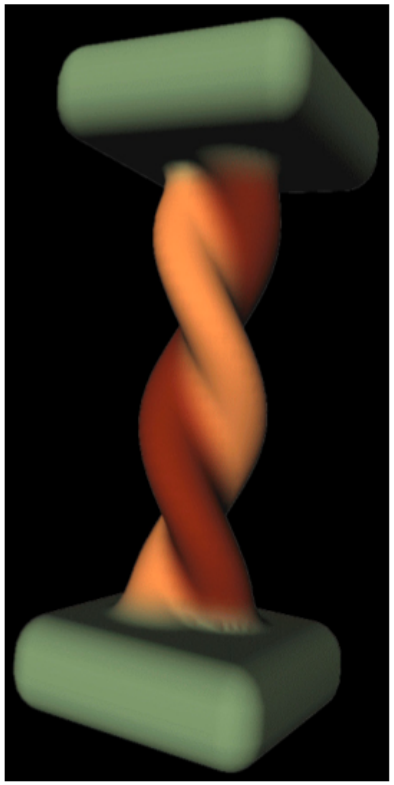
Skeletal Implicit Surfaces

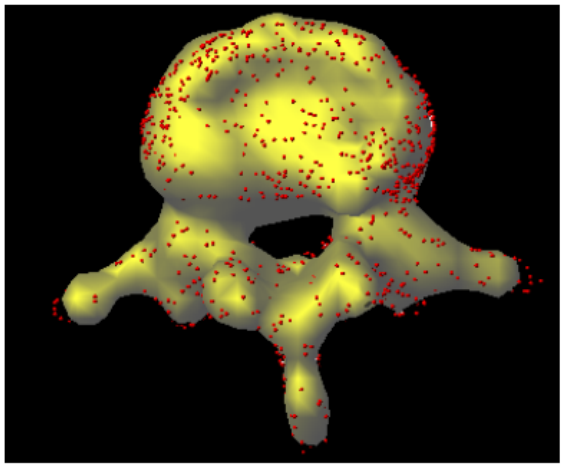
$$F = \sum F_i = d$$

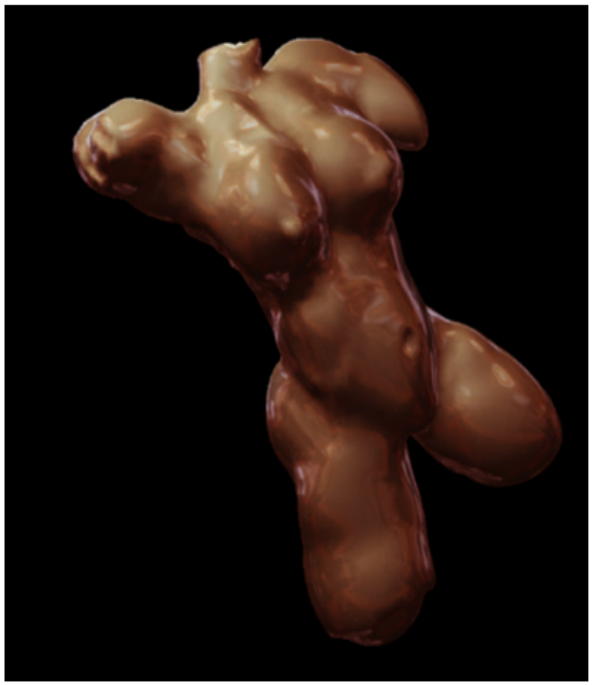
F_i: distance function $dist(p, S_i)$, S_i being the skeleton of the shape



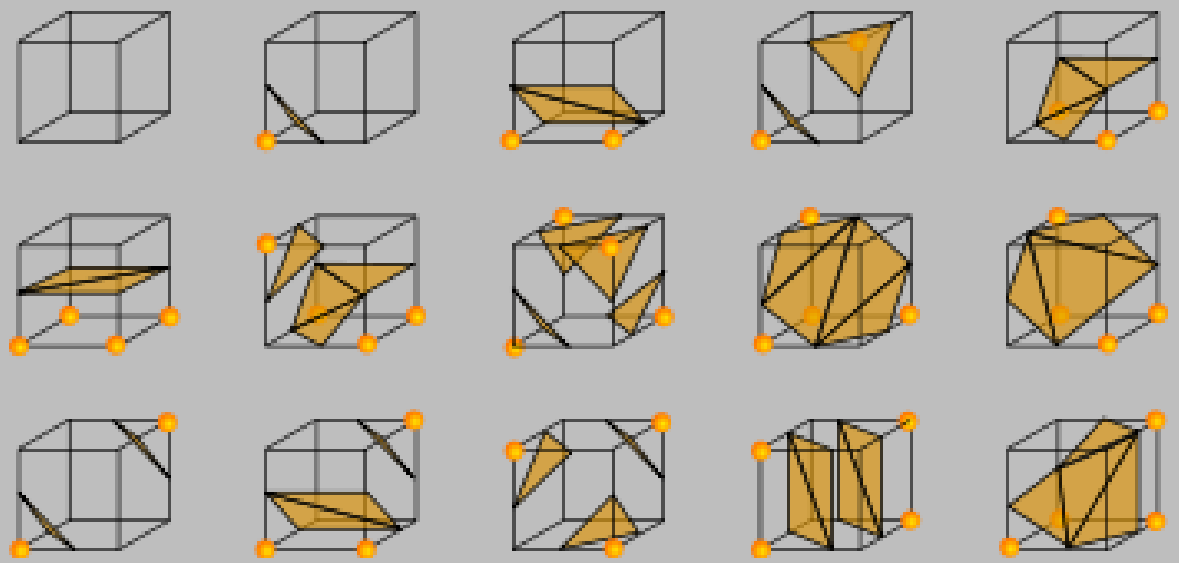
Implicit Surfaces - Examples



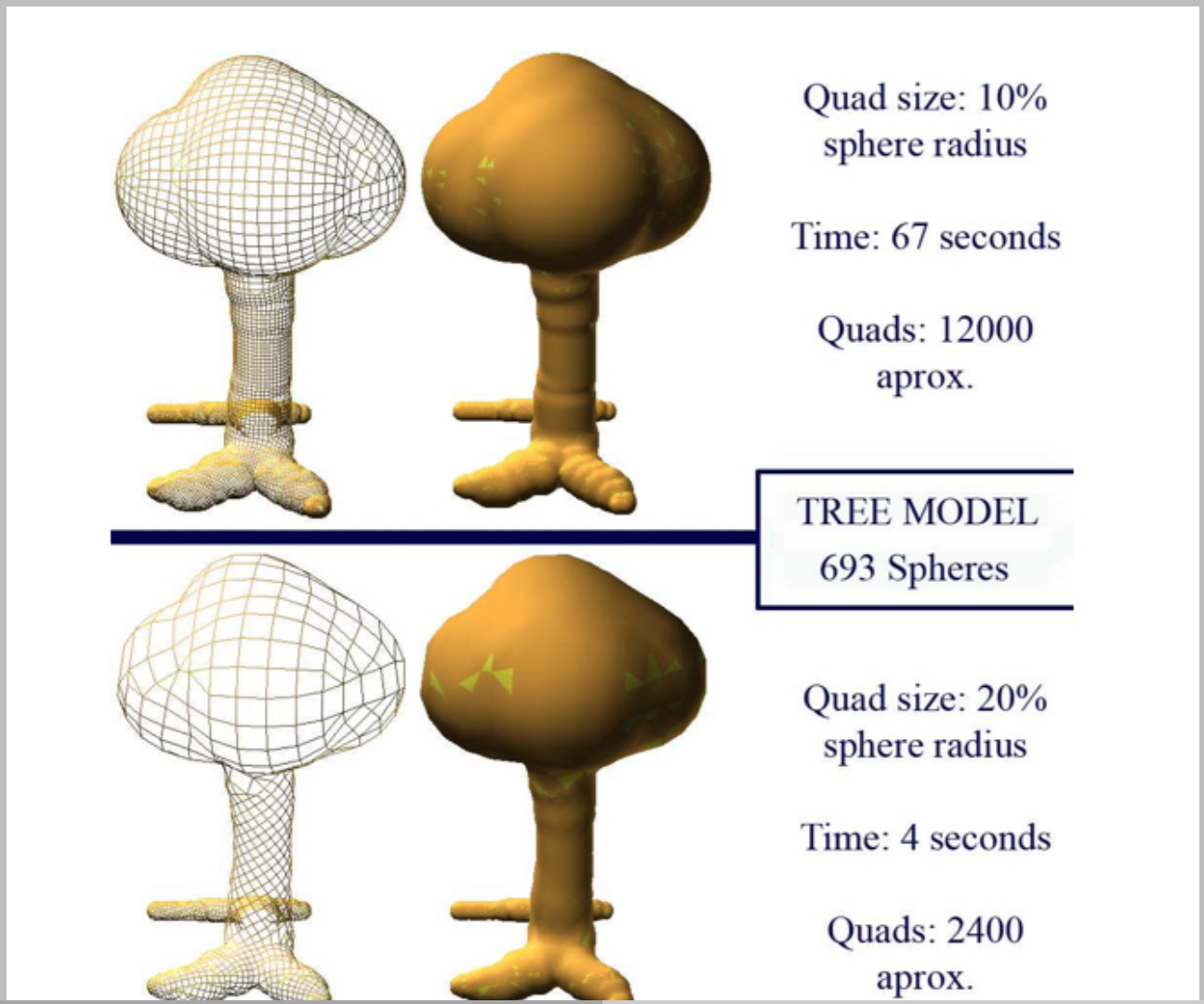




Implicit Surfaces - Rendering
Marching Cubes



Implicit Surfaces - Rendering
Marching Quads



Implicit Surfaces

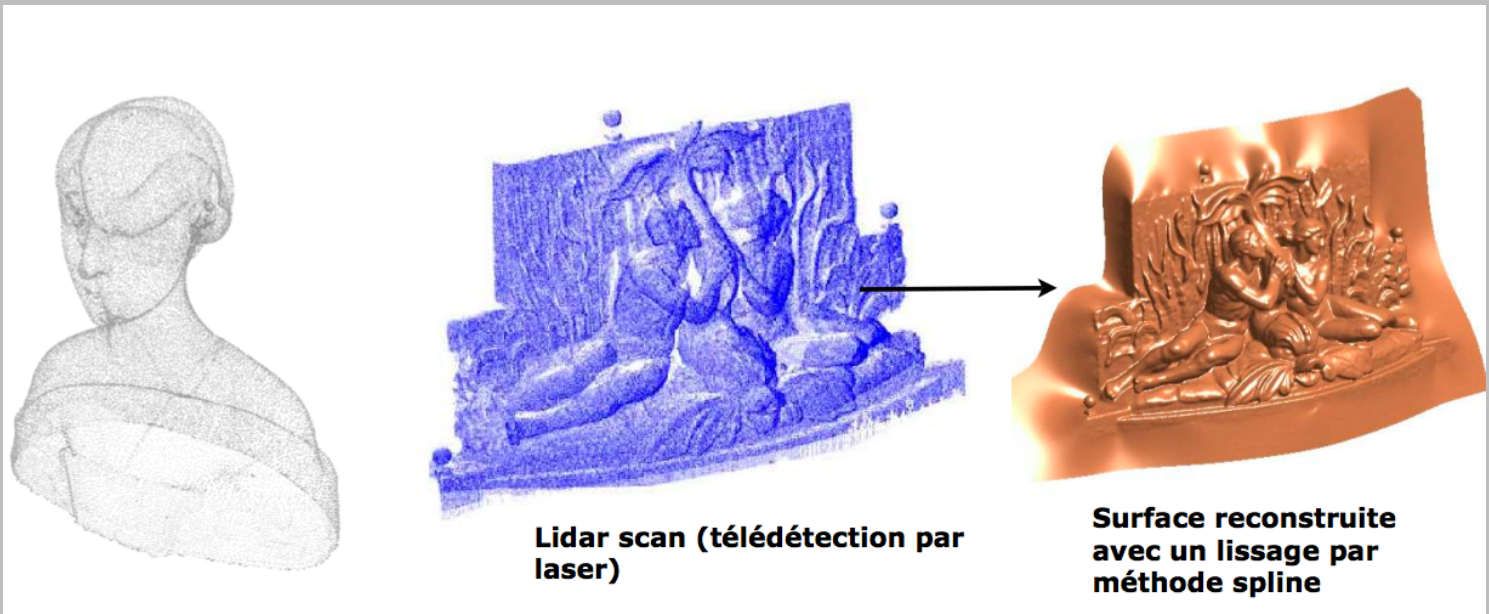
- Modeling sometimes difficult
  - Blobs with skeleton
  - Potential function
- Slow to render
  - Marching Cubes
  - Marching Triangles / Quads
  - Volumetric rendering
  - Techniques using particles

Outline

Unstructured Data

Unstructured data is usually produced through scanning devices or photos and is difficult to use

- Point Clouds
- Polygon Soup



Point Clouds

- Memory consuming
  - Need data simplification
- $3D$ reconstruction
  - Polygonizer
  - Inside / Outside ?
  - Coherency, topology, connexity ...
- Visualization: point-based rendering

Polygon Soup

Unstructured set of polygons (as opposition to meshes)

- + Adapted to OpenGL (ie not a problem for drawing the geometry)
- - Normals unknown (flat shading, reversed polygons)
- - Often inappropriate for animation