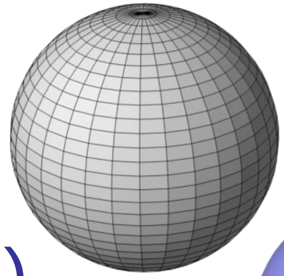
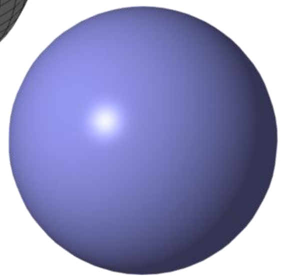
Computer Graphics

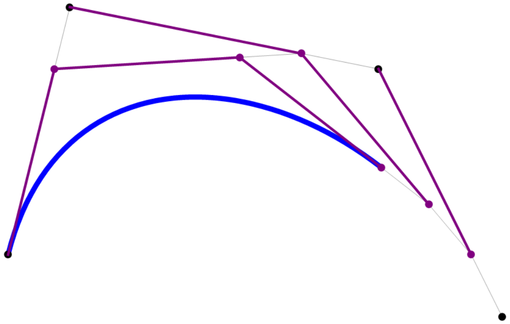
Curves and
Smooth Surfaces

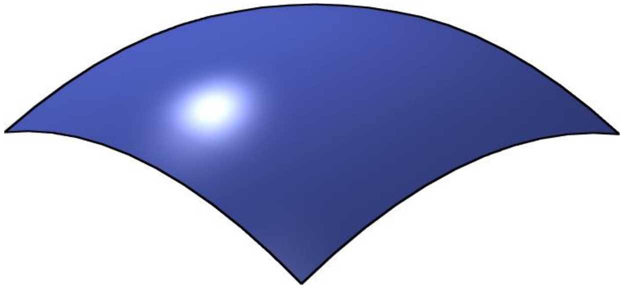
Curves and Smooth Surfaces





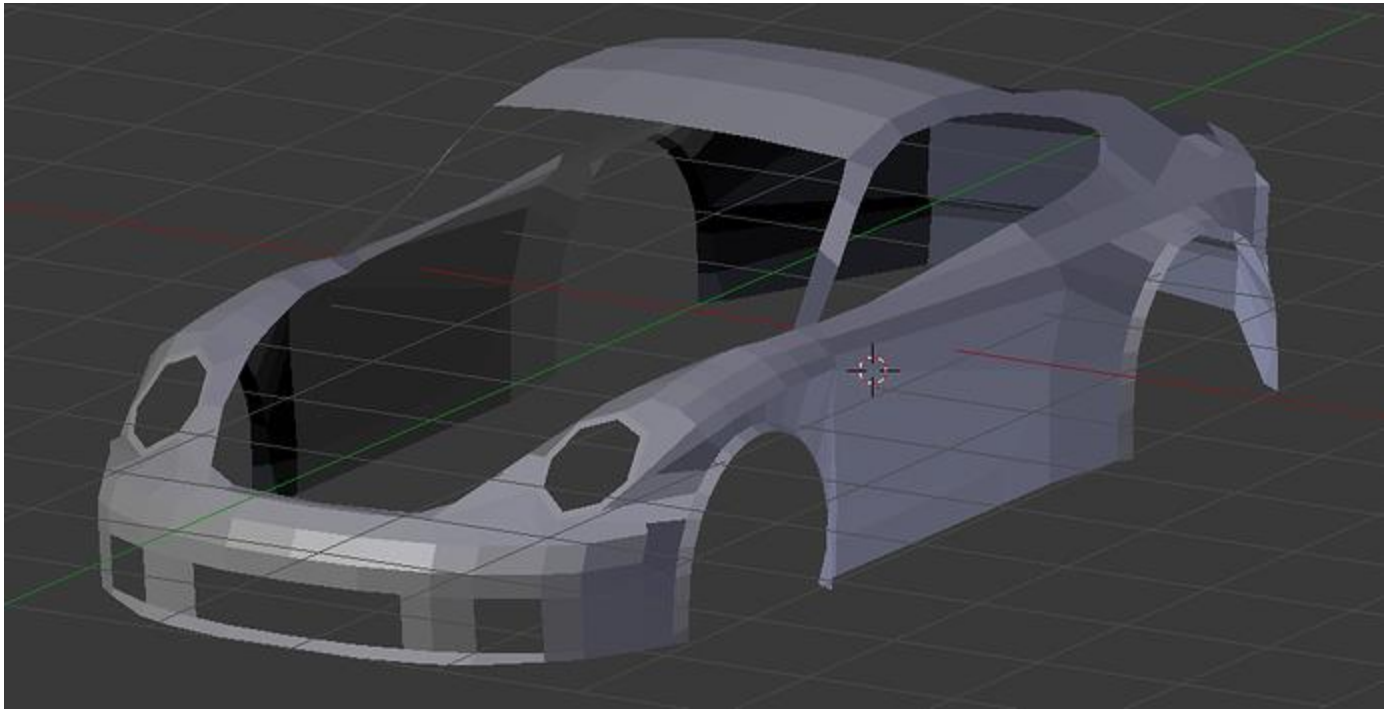
- object representations so far
  - polygonal meshes (shading)
  - analytical descriptions (raytracing)
- flexibility vs. accuracy
- now: flexible yet accurate representations
  - piecewise smooth curves: *Bézier curves, splines*
  - smooth (freeform) surfaces
  - subdivision surfaces





Again, why do we need all this?

- not only representation, but also **modeling**
- and it's all about cars! shiny cars! ☺



Again, why do we need all this?

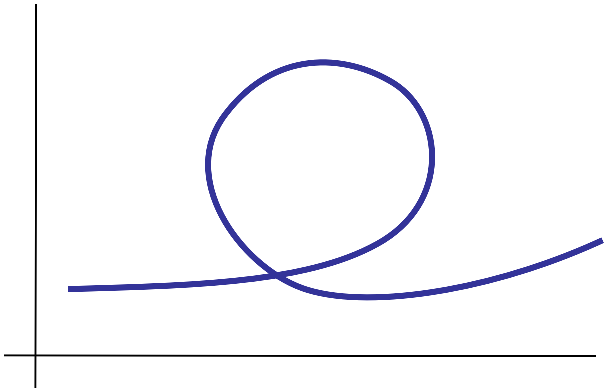
- not only representation, but also **modeling**
- and it's all about cars! shiny cars! ☺

Curves and Smooth Surfaces

Curves

Specifying Curves

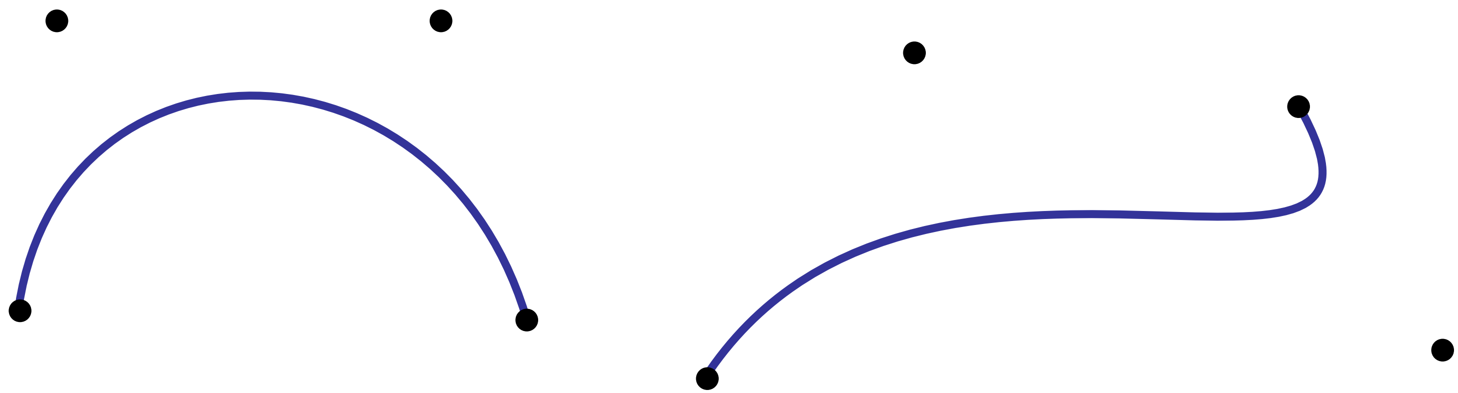
- functional descriptions
  - $y = f(x)$ in 2D;
    for 3D also $z = f(x)$
  - cannot have loops
  - functions only return one scalar, bad for 3D
  - difficult handling if it needs to be adapted



- parametric descriptions
  - independent scalar parameter $t \in \Re$
  - typically $t \in [0, 1]$, mapping into $\Re^2$ / $\Re^3$
  - point on the curve: $P(t) = (x(t), y(t), z(t))$

Polynomial Parametric Curves

- use **control points** to specify curves

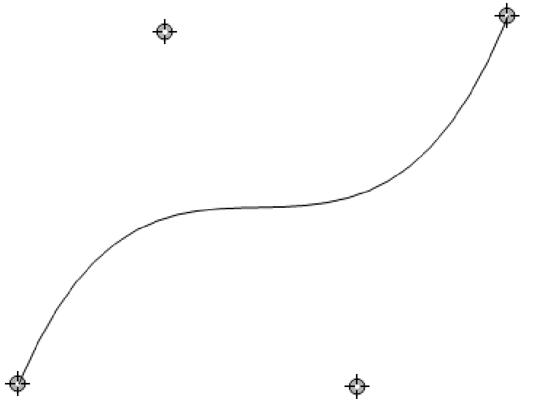


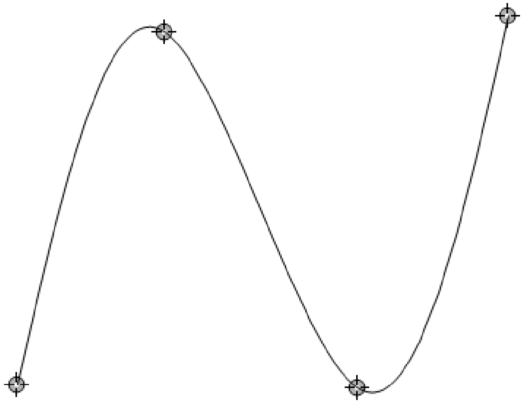
- n+1 control points for a curve segment
- set of **basis** or **blending functions**:

$$P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$$

Interpolating vs. Approximating

- two different curves schemes: curves do not always go through all control points
  - **approximating curves** not all control points are on the resulting curve
  - **interpolating curves** all control points are on the resulting curve





Curves and Smooth Surfaces

Bézier Curves

Bézier Curves: Blending Functions

- formulation of curve: $P(t) = \sum_{i=0}^{n} P_i B_{i,n}(t)$
- $B_{i,n}$ – **Bernstein polynomials** (control point weights, depend on t):

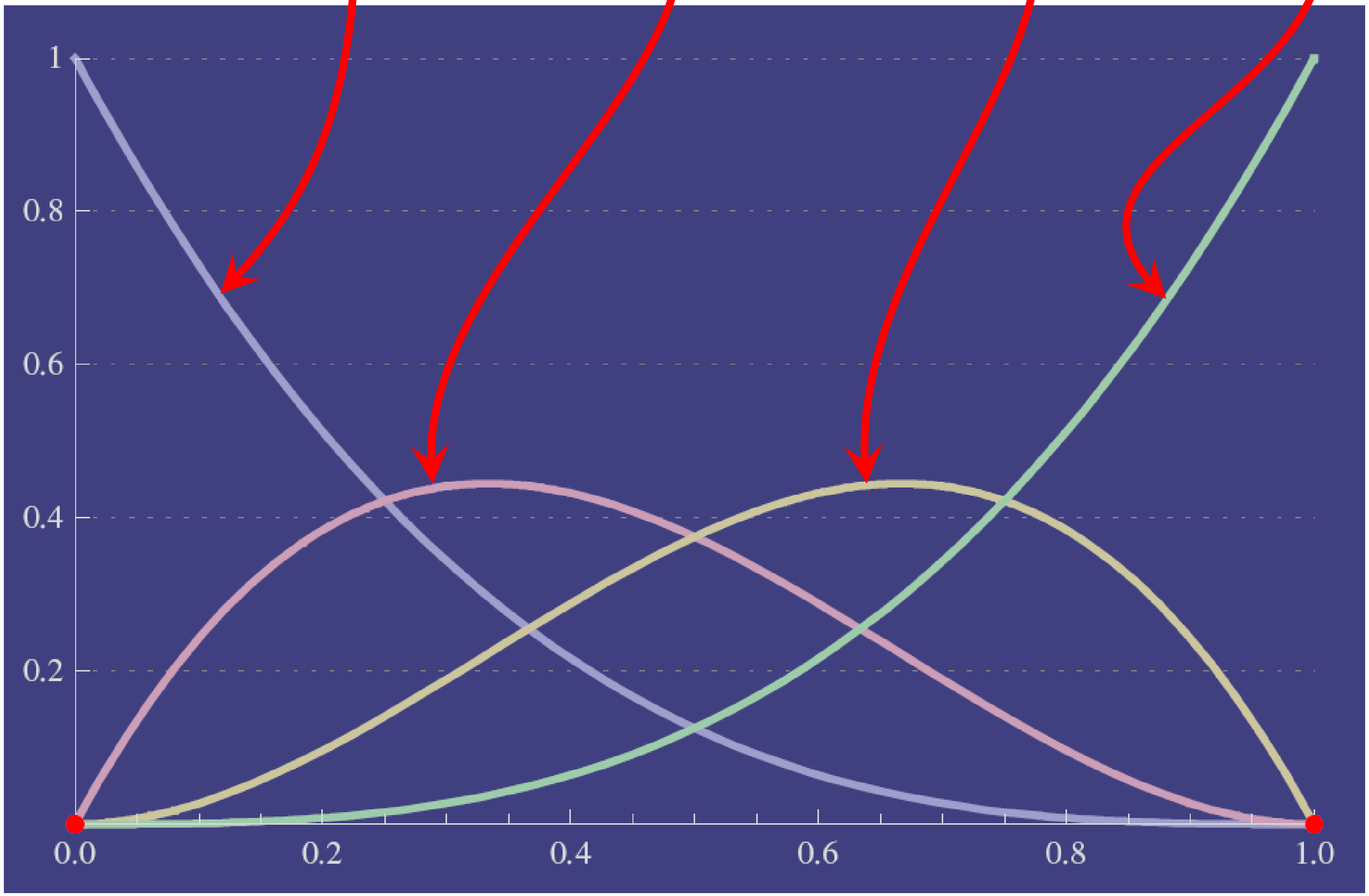
$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i} = \frac{n!}{i!\,(n-i)!} t^i (1-t)^{n-i}$$

- Bézier curve example for n = 3:

$$\begin{aligned} P(t) &= P_0 B_{0,3}(t) + P_1 B_{1,3}(t) + P_2 B_{2,3}(t) + P_3 B_{3,3}(t) \\ &= P_0\,(1-t)^3 + P_1\,3t(1-t)^2 + P_2\,3t^2(1-t) + P_3\,t^3 \end{aligned}$$

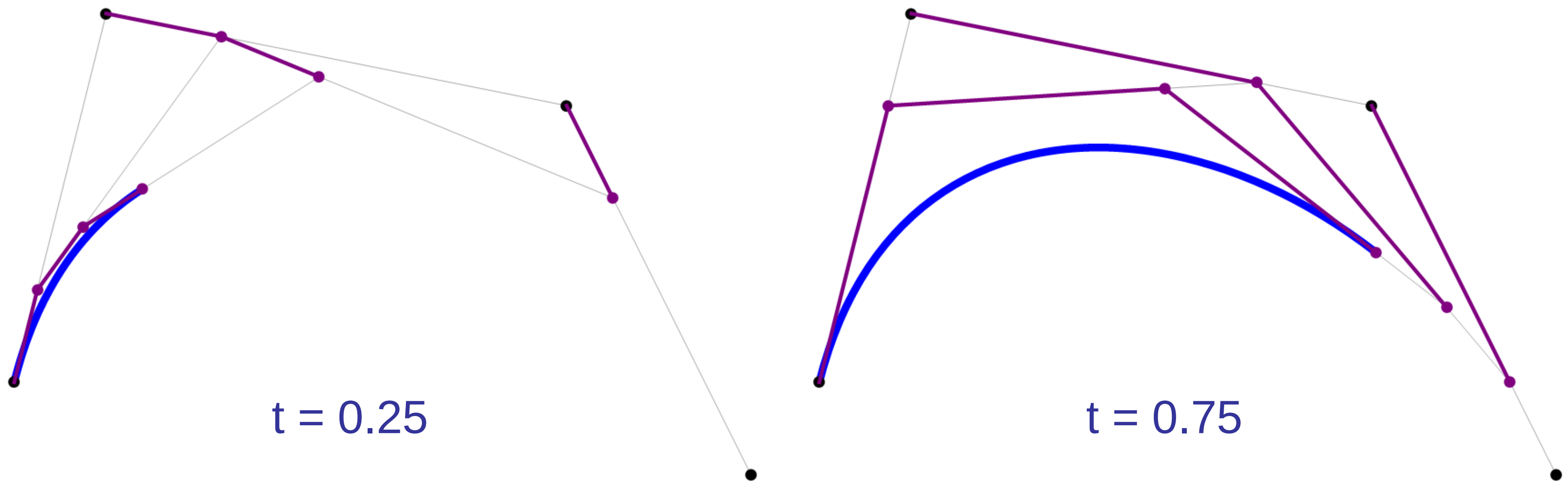
Bernstein Polynomials Visualized

$$P(t) = P_0 \boxed{(1-t)^3} + P_1 \boxed{3t(1-t)^2} + P_2 \boxed{3t^2(1-t)} + P_3 \boxed{t^3}$$

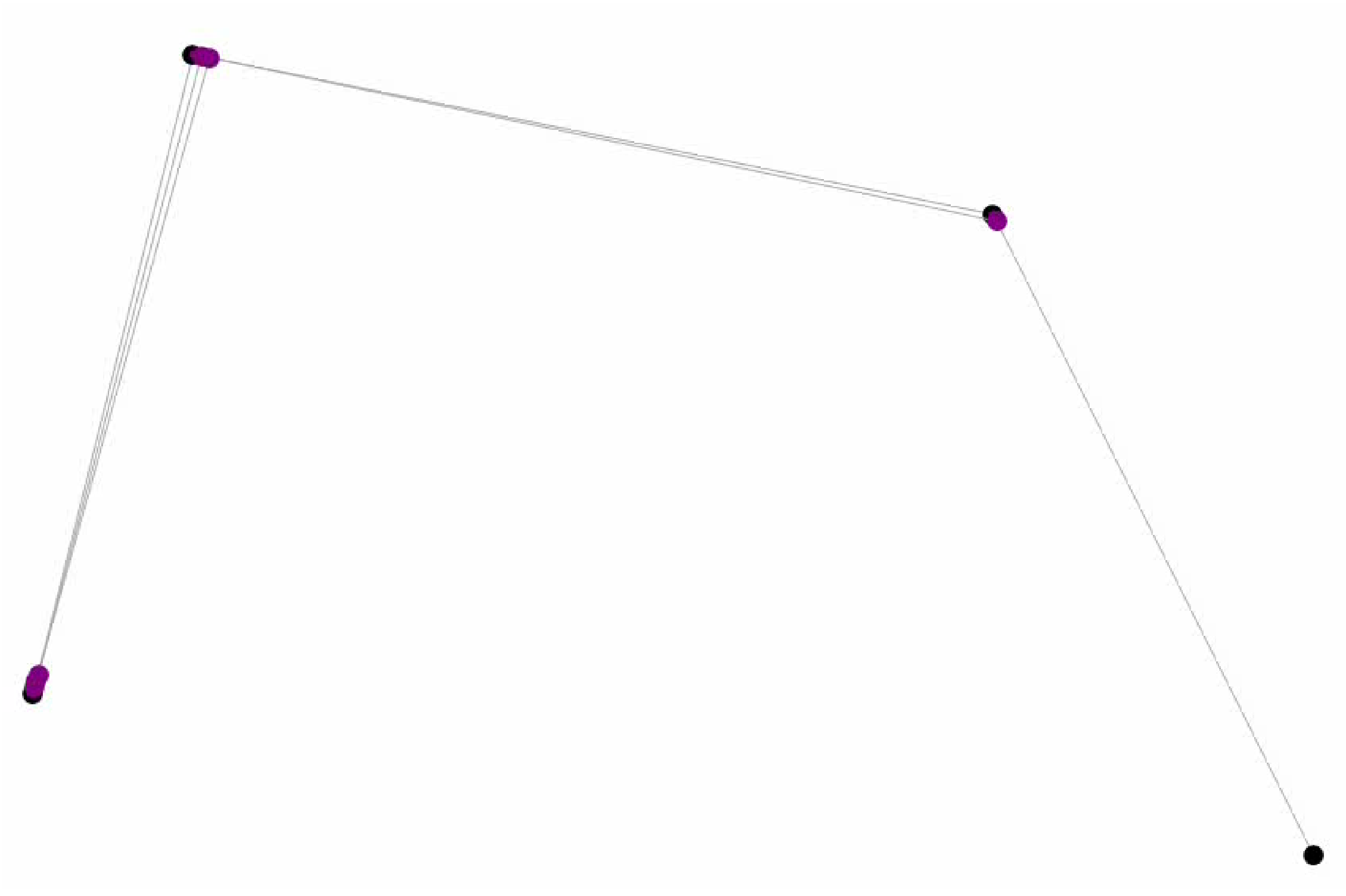


De Casteljau's Algorithm

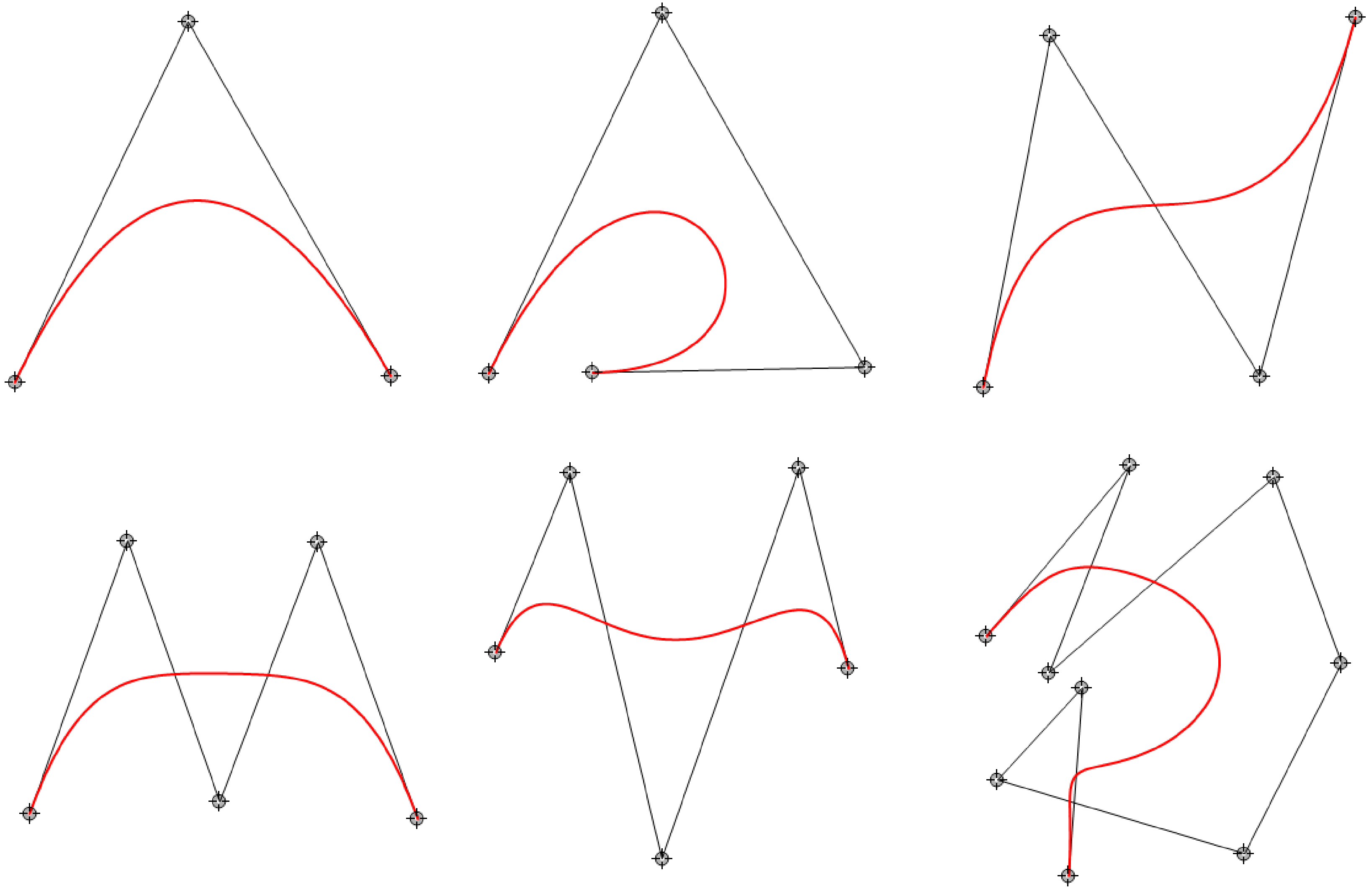
- algorithm by Paul de Casteljau
  trivia: original inventor of Bézier curves (in 1959);
  Pierre Bézier just publicized them widely in 1962;
  both working for French car makers (Citroën & Renault)
- geometric & numerically stable way to evaluate the polynomials in Bézier curves



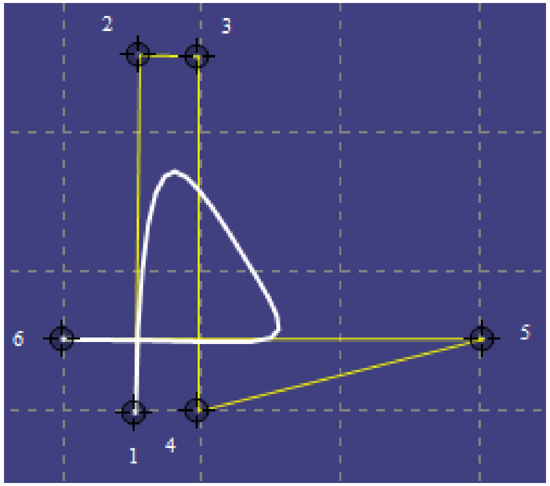
De Casteljau's Algorithm

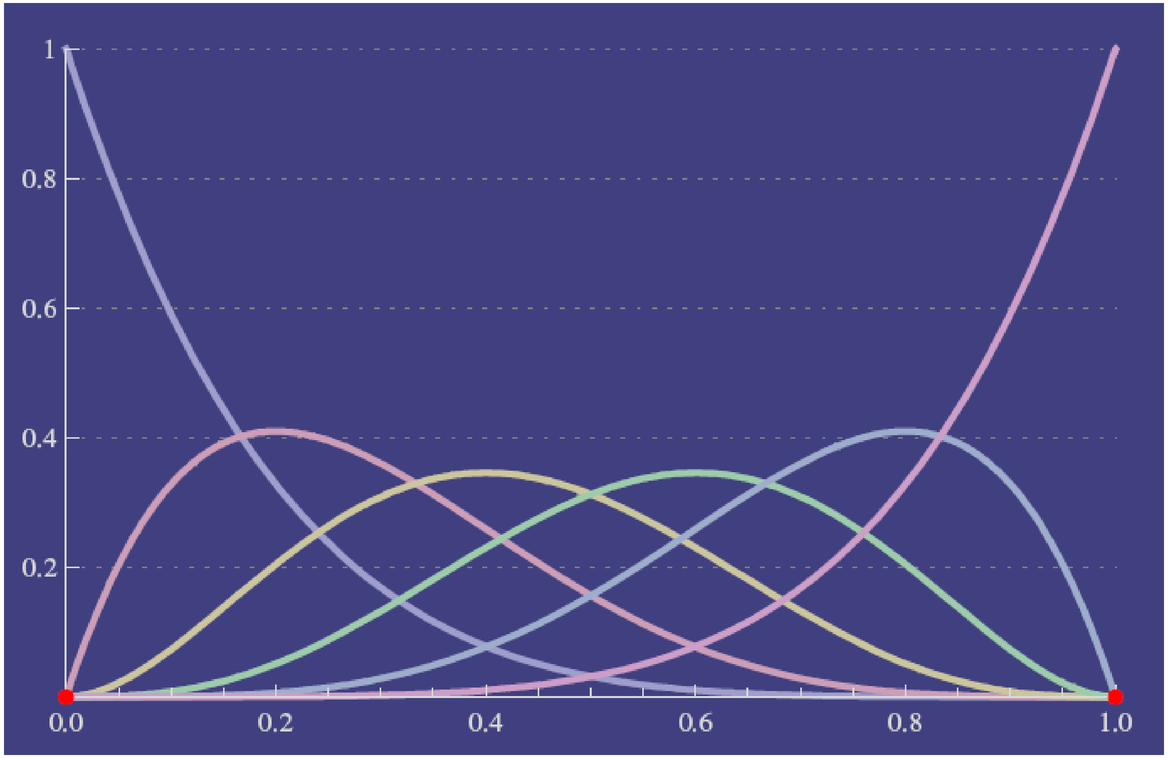


Bézier Curves: Examples



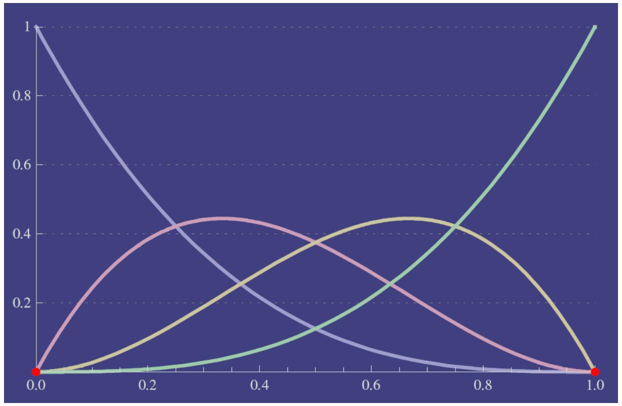
Bézier Curves: Example & $B_{i,n}$





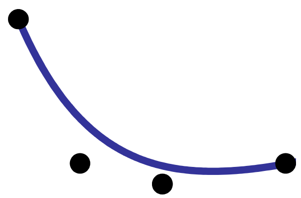
Bézier Curves: Properties

- curve always inside the **convex hull** of the control polygon – why? $\sum_{i=0}^{n} B_{i,n}(t) = 1 \quad \forall t \in [0,1]$
- **approximating** curve: only first & last control points are interpolated – why?
- each control point affects the entire curve, **limited local control** $\rightarrow$ problem for modeling



Piecewise Smooth Curves

- low order curves give sufficient control
- *idea*: connect segments together
  - each segment only affected
    by its own control points $\rightarrow$ local control
  - make sure that segments connect smoothly

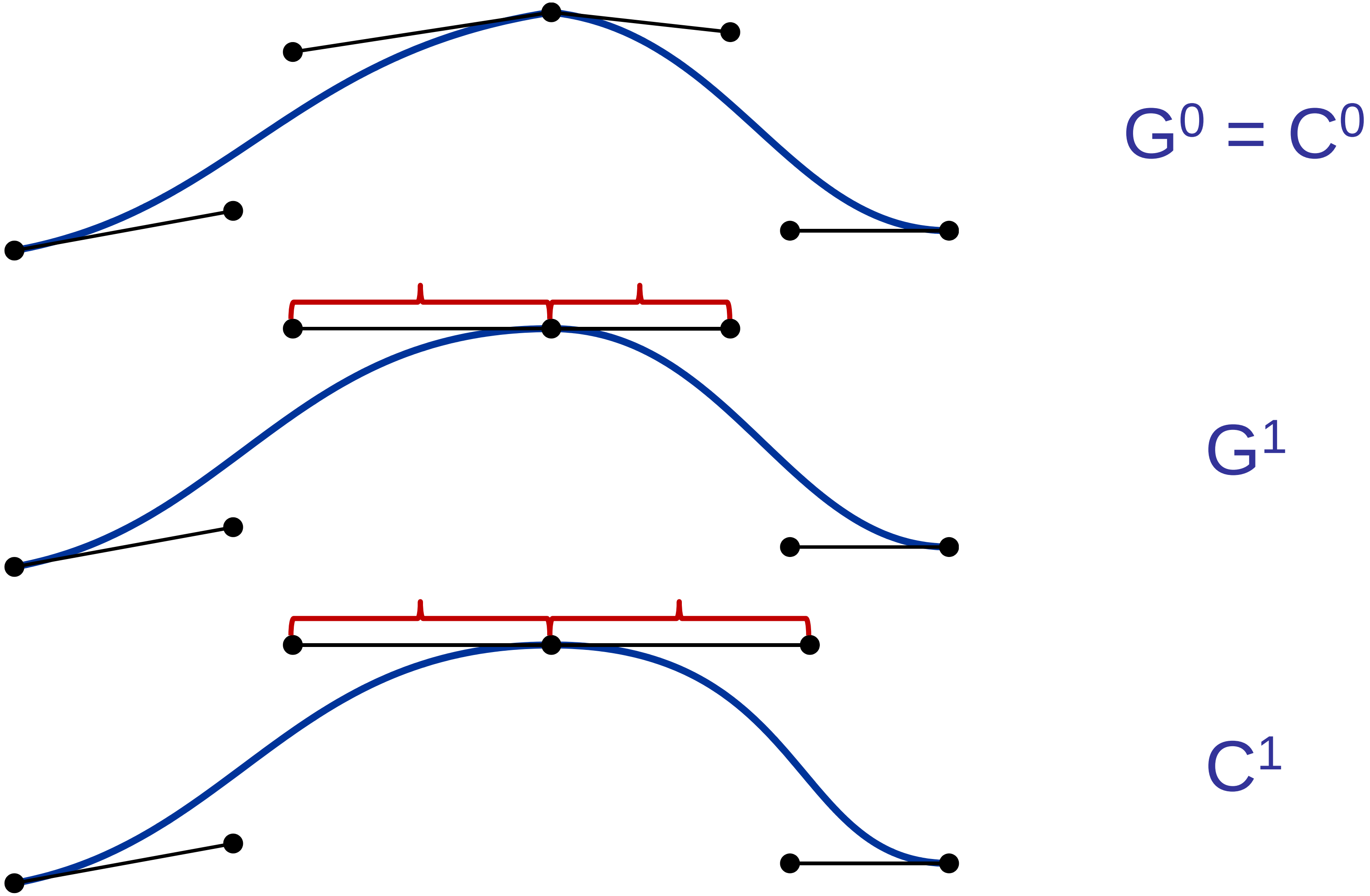


- *problem*: what are smooth connections?

Continuity Criteria

- a curve *s* is said to be C^n-continuous if its n^{th} derivative $d^n s/dt^n$ is continuous of value $\rightarrow$ **parametric continuity**: shape & speed
- not only for individual curves, but also *and in particular* for where segments connect
- **geometric continuity**: two curves are G^n-continuous if they have proportional n^{th} derivatives (same direction, speed can differ)
- G^n follows from C^n, but not the other way
- car bodies need at least G^2-continuity

Continuity Criteria: Examples

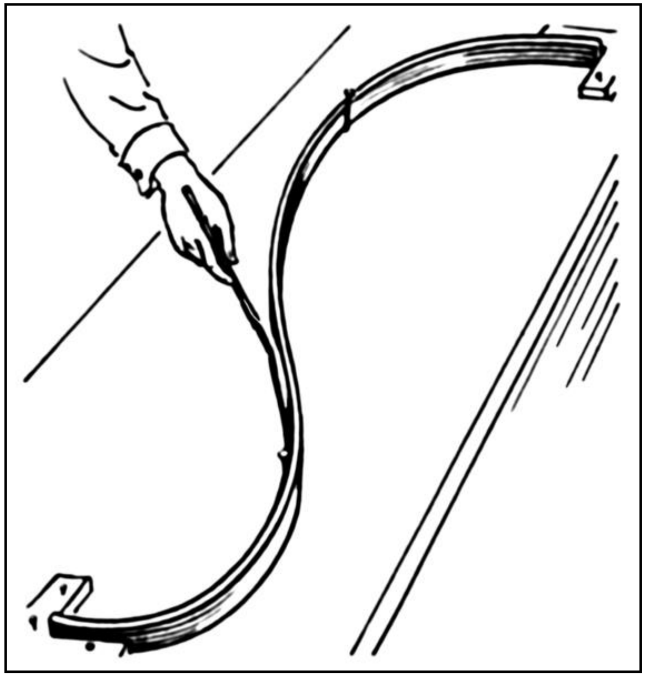


Curves and Smooth Surfaces

Splines

Splines

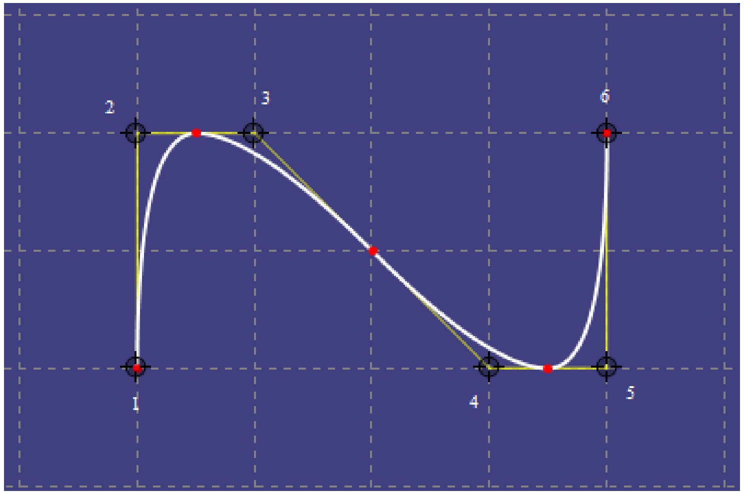
- term from manufacturing (cars, planes, ships, etc.): metal strips with weights or similar attached

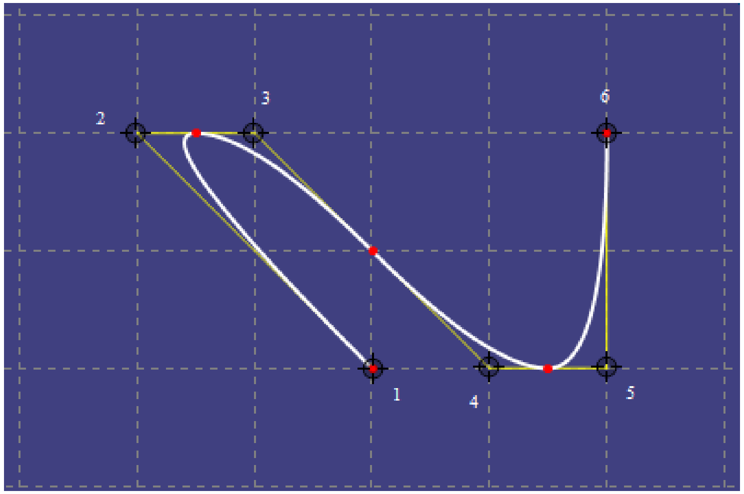


- mathematically in cg: composite curves that are composed of polynomial sections and that satisfy specified continuity conditions
- Bézier curves are one class of splines

B-Splines

- Bézier curves: global reaction to change
- *goal*: find curve that provides local control
- *idea*: approximating curve with many control points where only a few consecutive control points have local influence:





B-Splines

- mathematical formulation (n+1 control pts):

$$P(t) = \sum_{i=0}^{n} P_i B_{i,d}(t) \qquad 1 \leq d \leq n$$

degree

$$B_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \leq t \leq t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

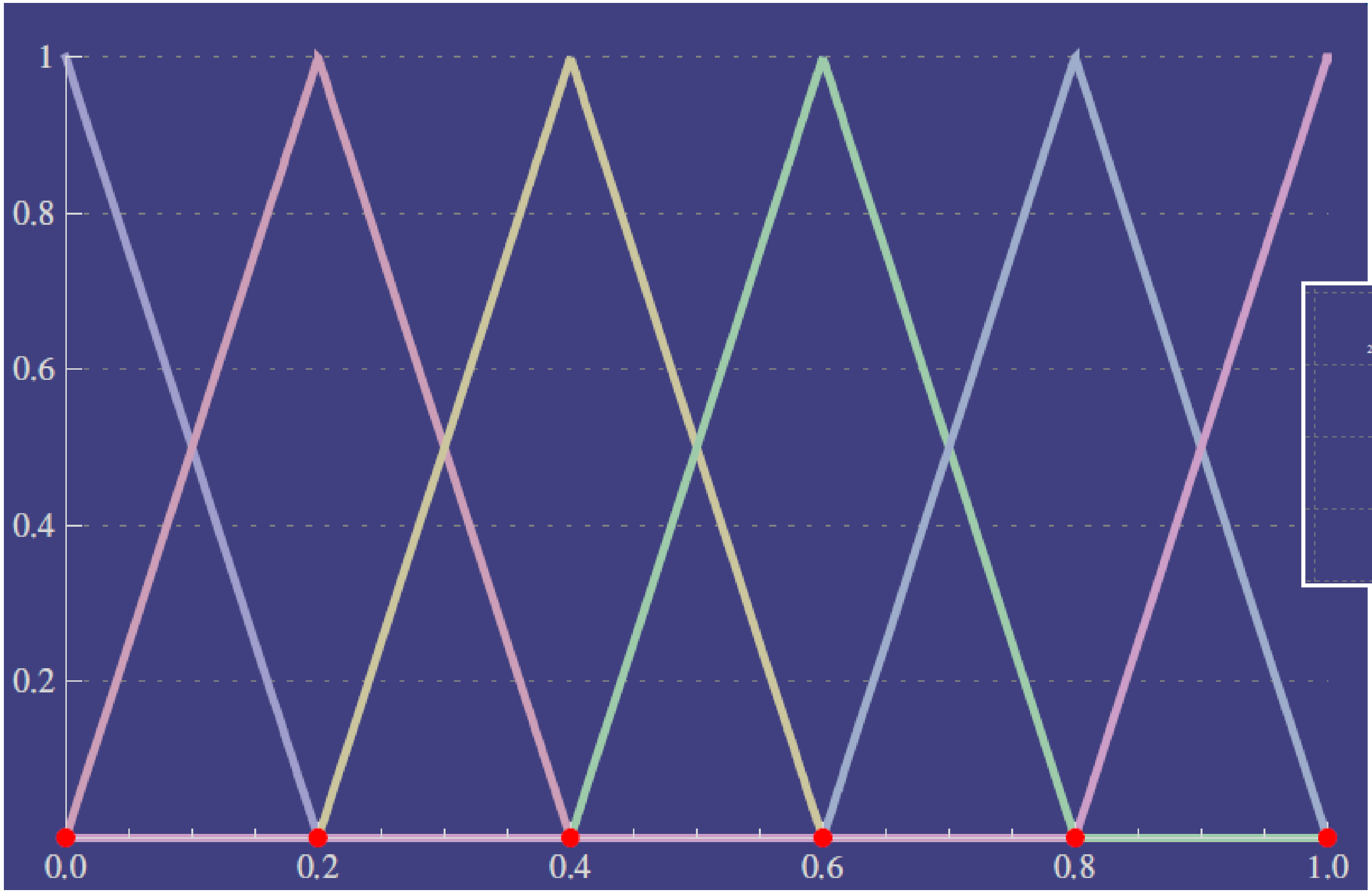
knots

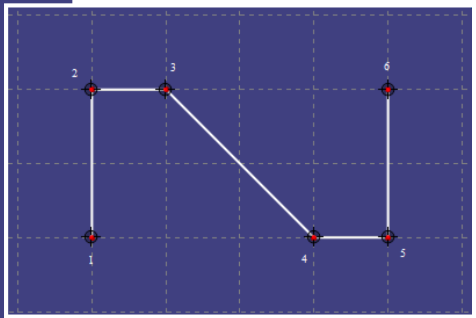
$$B_{i,d}(t) = \frac{t - t_i}{t_{i+d-1} - t_i} B_{i,d-1}(t) + \frac{t_{i+d} - t}{t_{i+d} - t_{i+1}} B_{i+1,d-1}(t)$$

- recursive definition of $B_{i,d}$
- $B_{i,d}$ only non-zero for certain range (knots)
- range of each $B_{i,d}$ grows with degree

B-Splines

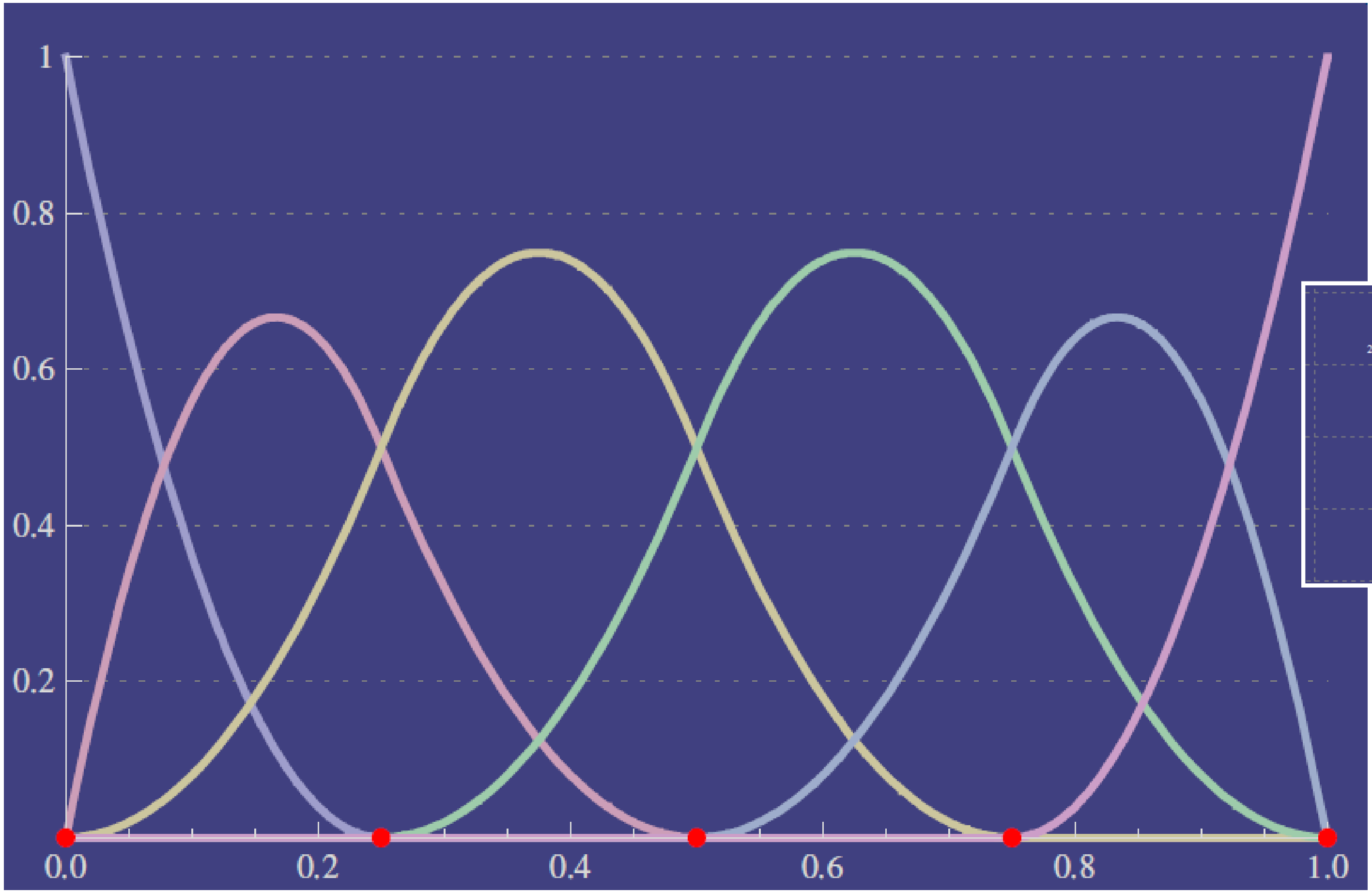
degree: 1

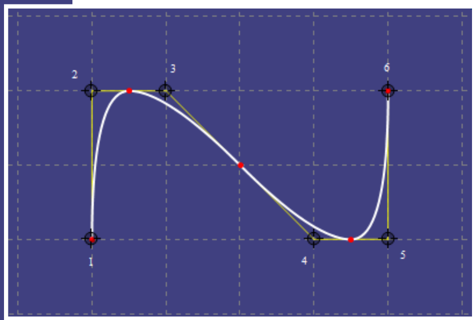




B-Splines

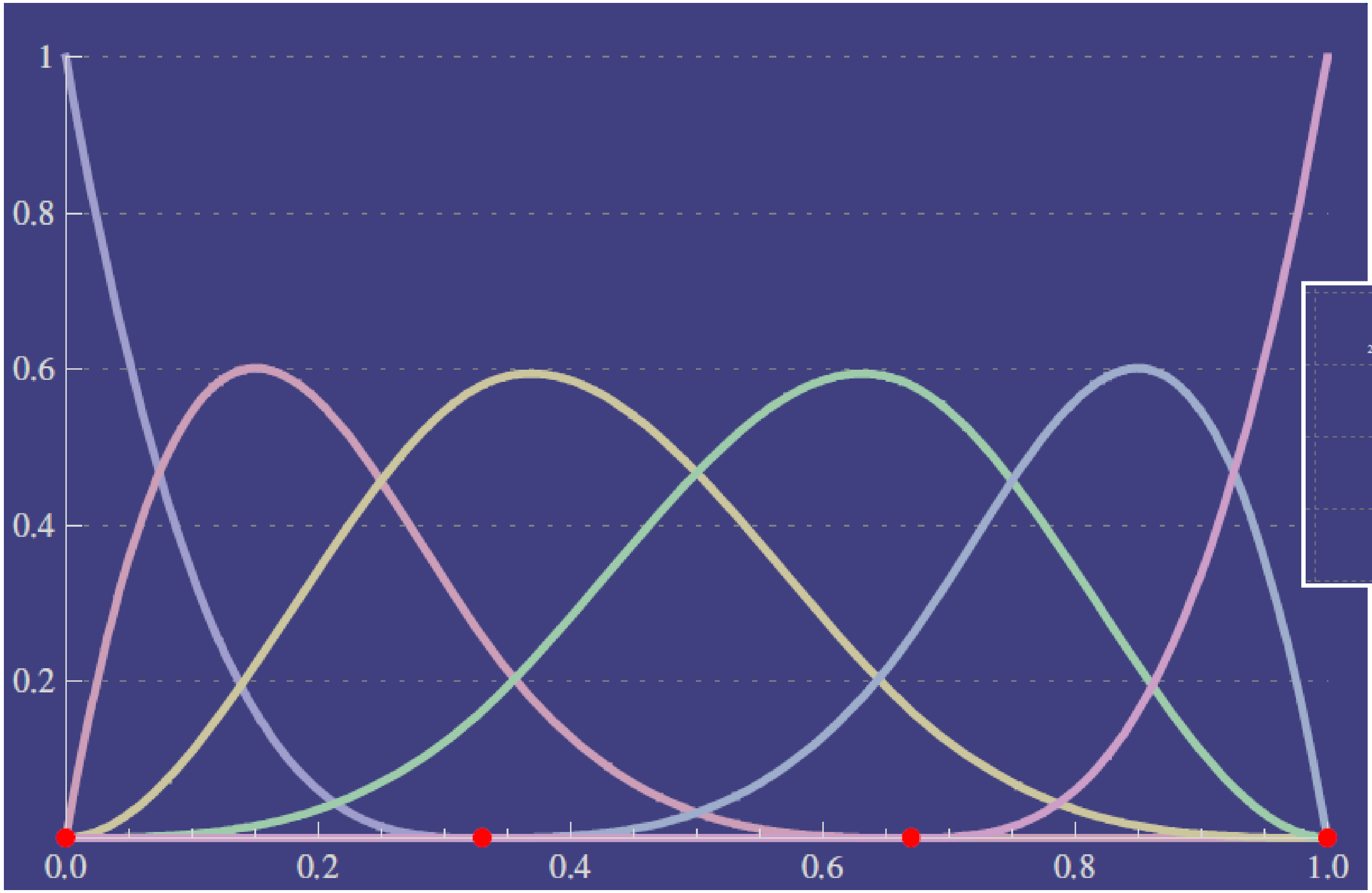
degree: 2

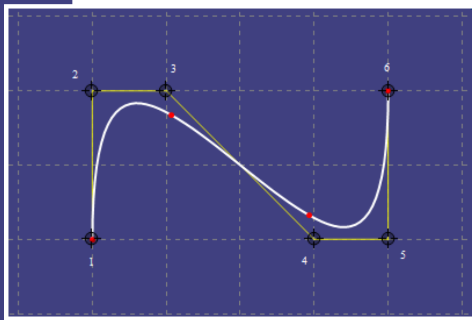




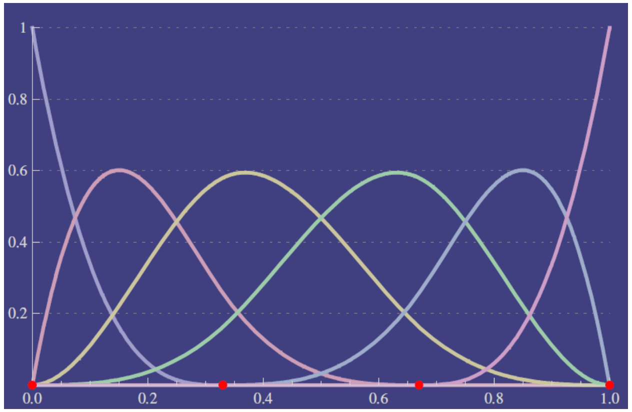
B-Splines

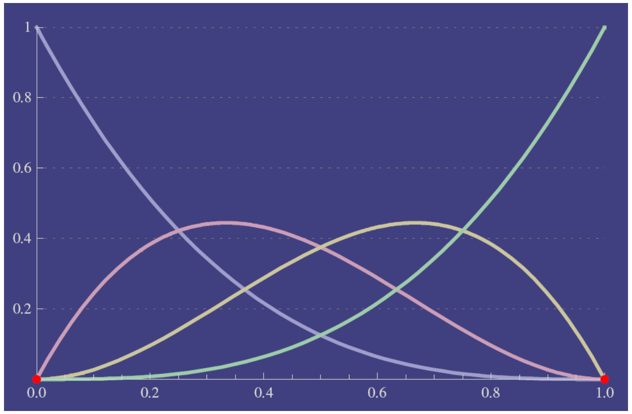
degree: 3

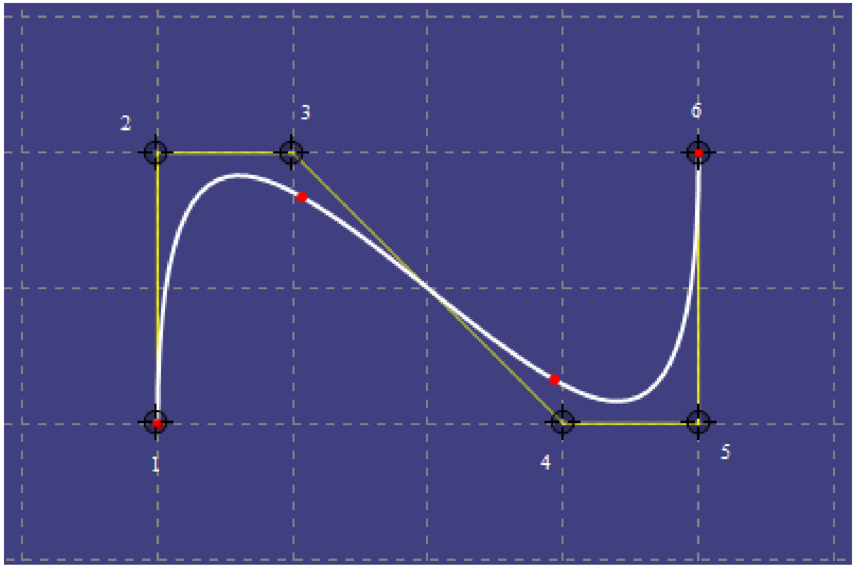


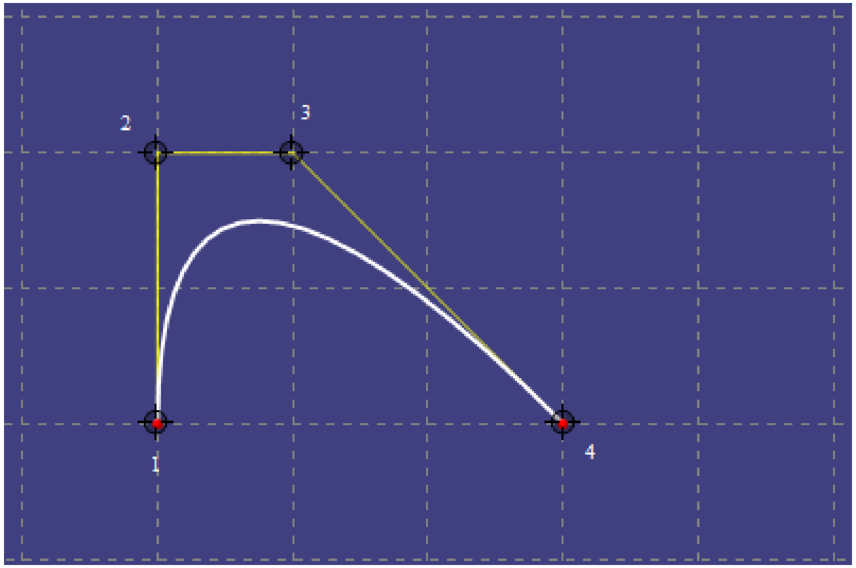


B-Splines vs. Bézier Curves

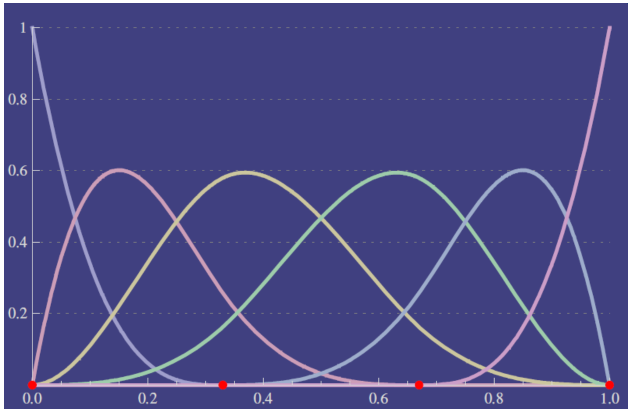


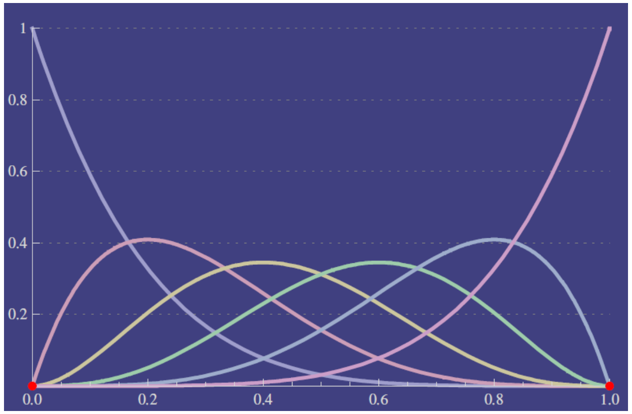


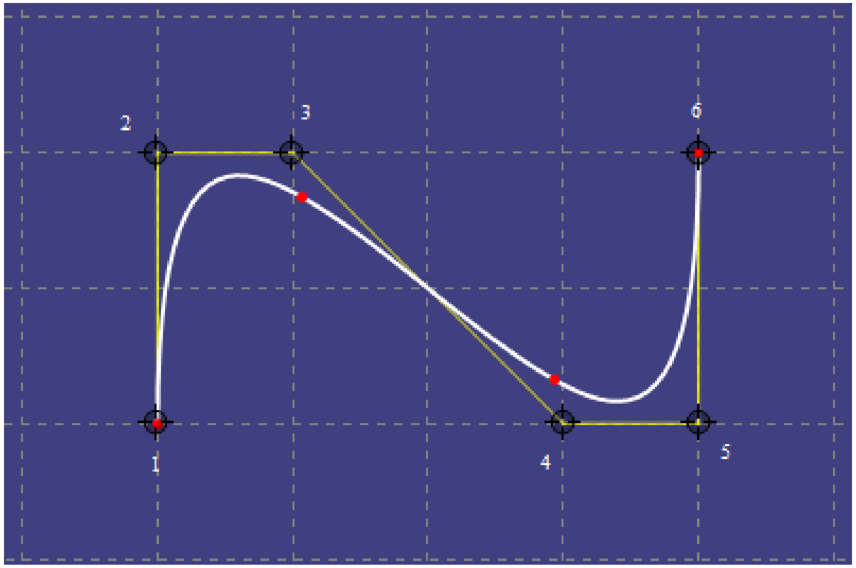




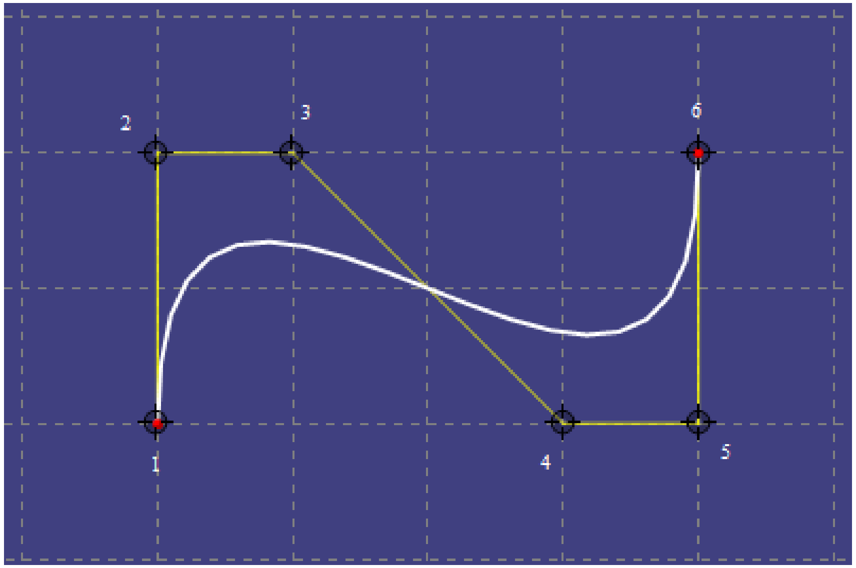
B-Splines vs. Bézier Curves







cubic B-spline



degree 5 B-spline and Bézier curve

NURBS

- knots can be non-uniformly spaced in the parameter space
- additional skalar weights for control points
- Non Uniform Rational Basis Spline:

$$P(t) = \frac{\sum_{i=0}^{n} h_i \, P_i \, B_{i,d,k}(t)}{\sum_{i=0}^{n} h_i \, B_{i,d,k}(t)}$$

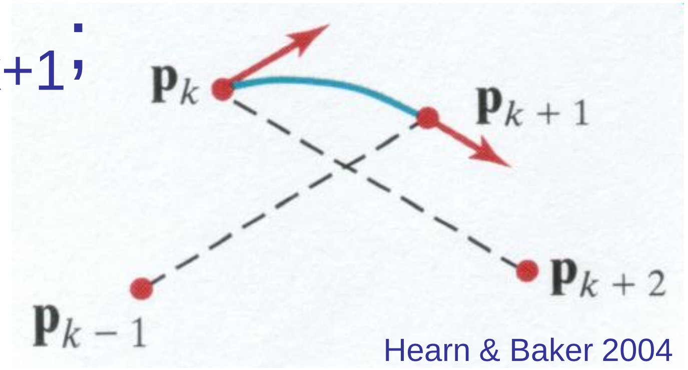
- “rational” refers to ration, i.e., a quotient
- can also represent, e.g., conic sections

Interpolating Curves

- how to specify smooth curves that interpolate control points?
- *idea*: use 4 control points to specify an interpolating curve between the middle 2
- *example*: **Cardinal splines**:

$$P(t) = P_{k-1}Car_0(t) + P_kCar_1(t) + P_{k+1}Car_2(t) + P_{k+2}Car_3(t)$$

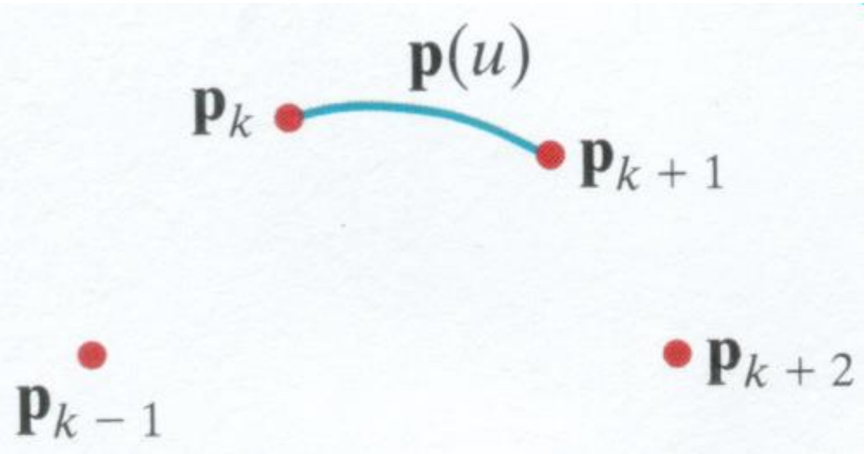
- curve defined from P_k to P_{k+1}; P_{k-1} & P_{k+1} as well as P_k & P_{k+2} define tangents:

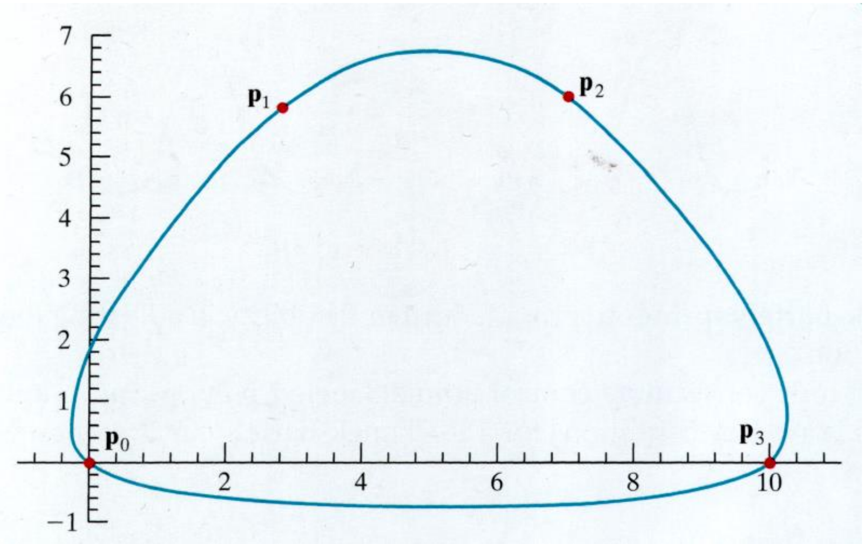


Hearn & Baker 2004

Open vs. Closed Cardinal Splines

- **open curves** need extra control points to specify the boundary conditions
- for **closed curves** no boundary conditions necessary, treat as never-ending curve





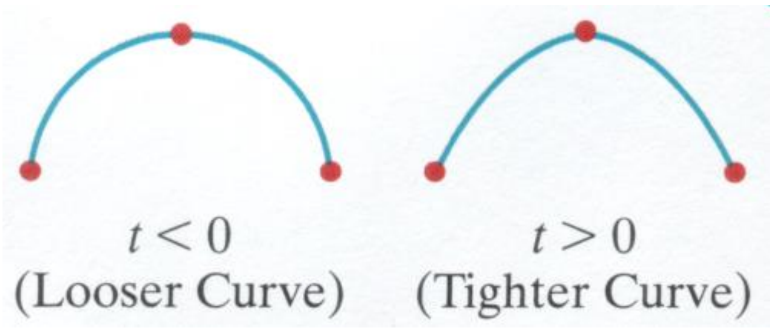
Hearn & Baker 2004

Cardinal Splines: Definition

- Car_i – cubic polynomial blending functions:

$$
\begin{aligned}
P(t) = {} & P_{k-1}(-s\,t^3 + 2s\,t^2 - s\,t) + \\
& P_k((2-s)\,t^3 + 2(s-3)\,t^2 + 1) + \\
& P_{k+1}((s-2)\,t^3 + (3-2s)\,t^2 + s\,t) + \\
& P_{k+2}(s\,t^3 - s\,t^2)
\end{aligned}
$$

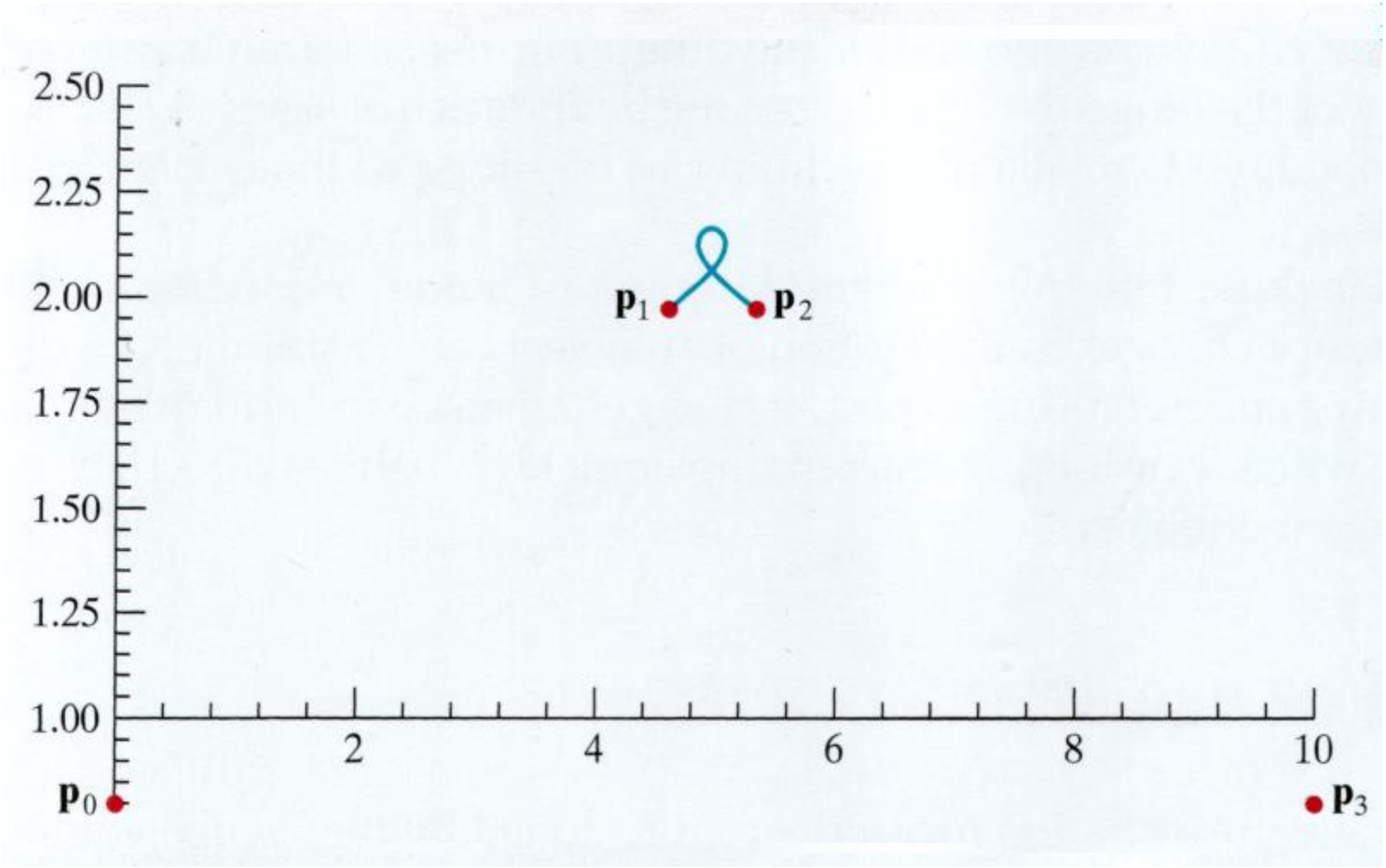
$$
s = \frac{1 - tension}{2}
$$



Hearn & Baker 2004

- tension parameter
  to control curve path and overshooting

Cardinal Splines: Examples



Hearn & Baker 2004

Smooth Curves: Summary

- parametric definition using parameter t
- flexible control points to control path
- blending functions compute each control point's contribution for a given parameter t
- works for 2D and 3D curves alike:
  just use 2D or 3D control points
- two ways to gain local control:
  - stitching low-degree curves together
  - using b-splines with degree parameter

Curves and Smooth Surfaces

Freeform Surfaces

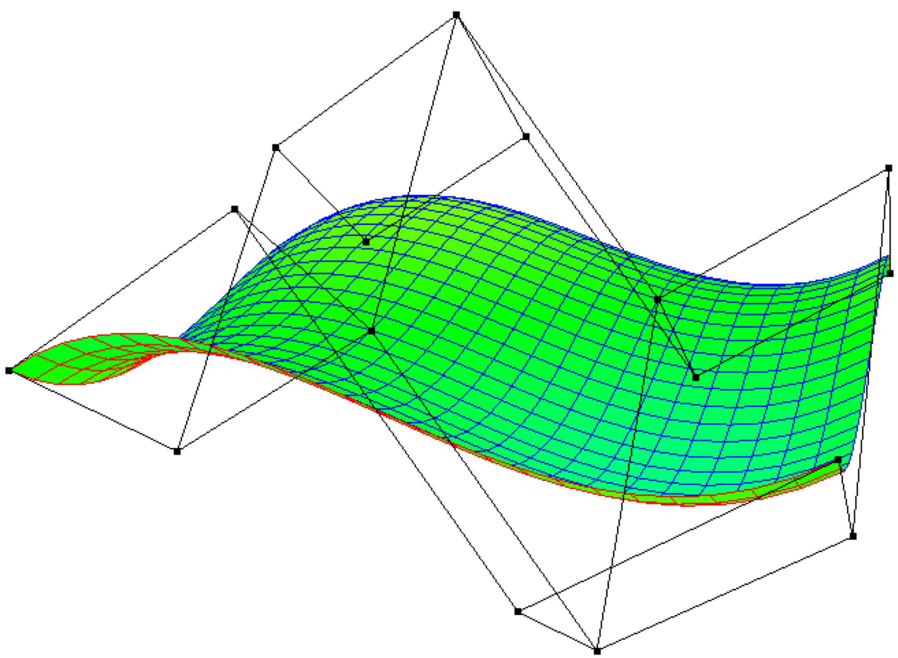
Freeform Surfaces

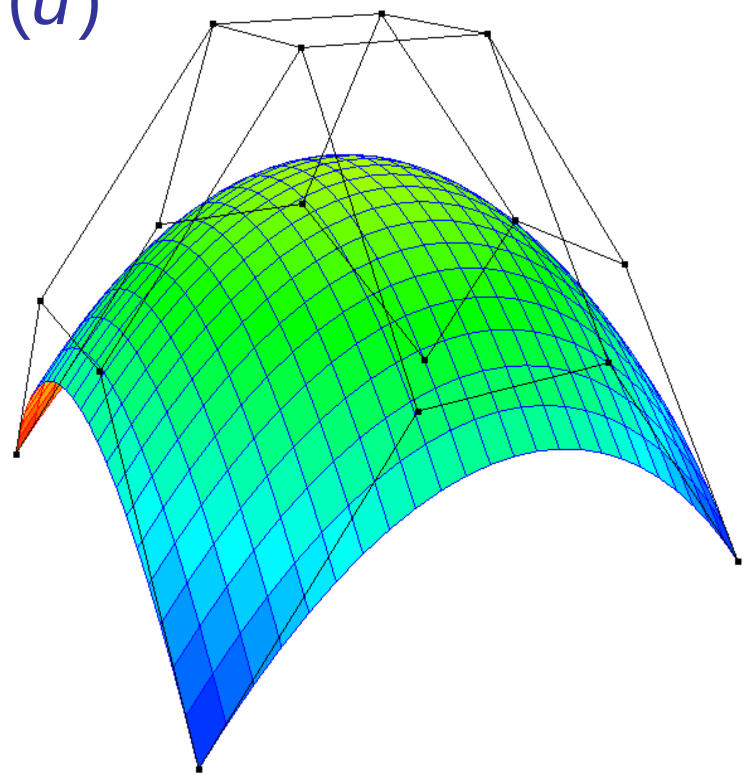
- base surfaces on parametric curves
- Bézier curves $\rightarrow$ Bézier surfaces/patches
- spline curves $\rightarrow$ spline surfaces/patches
- mathematically:
  application of curve formulations
  along two parametric directions

Freeform Surfaces: Principle

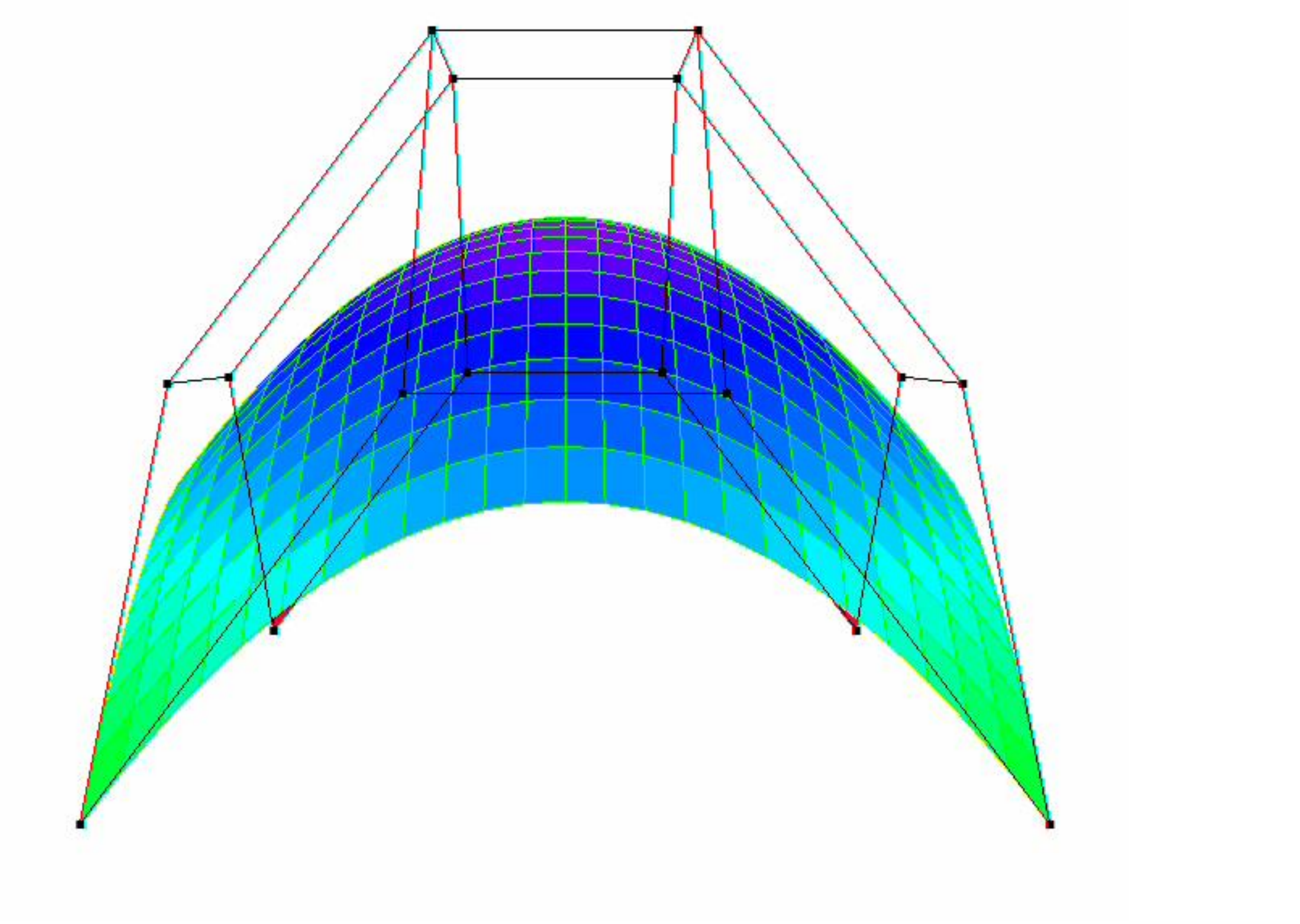
- Bézier surface: control mesh with m × n control points now specifies the surface:

$$P(u,v) = \sum_{j=0}^{m} \sum_{i=0}^{n} P_{j,i}\, B_{j,m}(v)\, B_{i,n}(u)$$

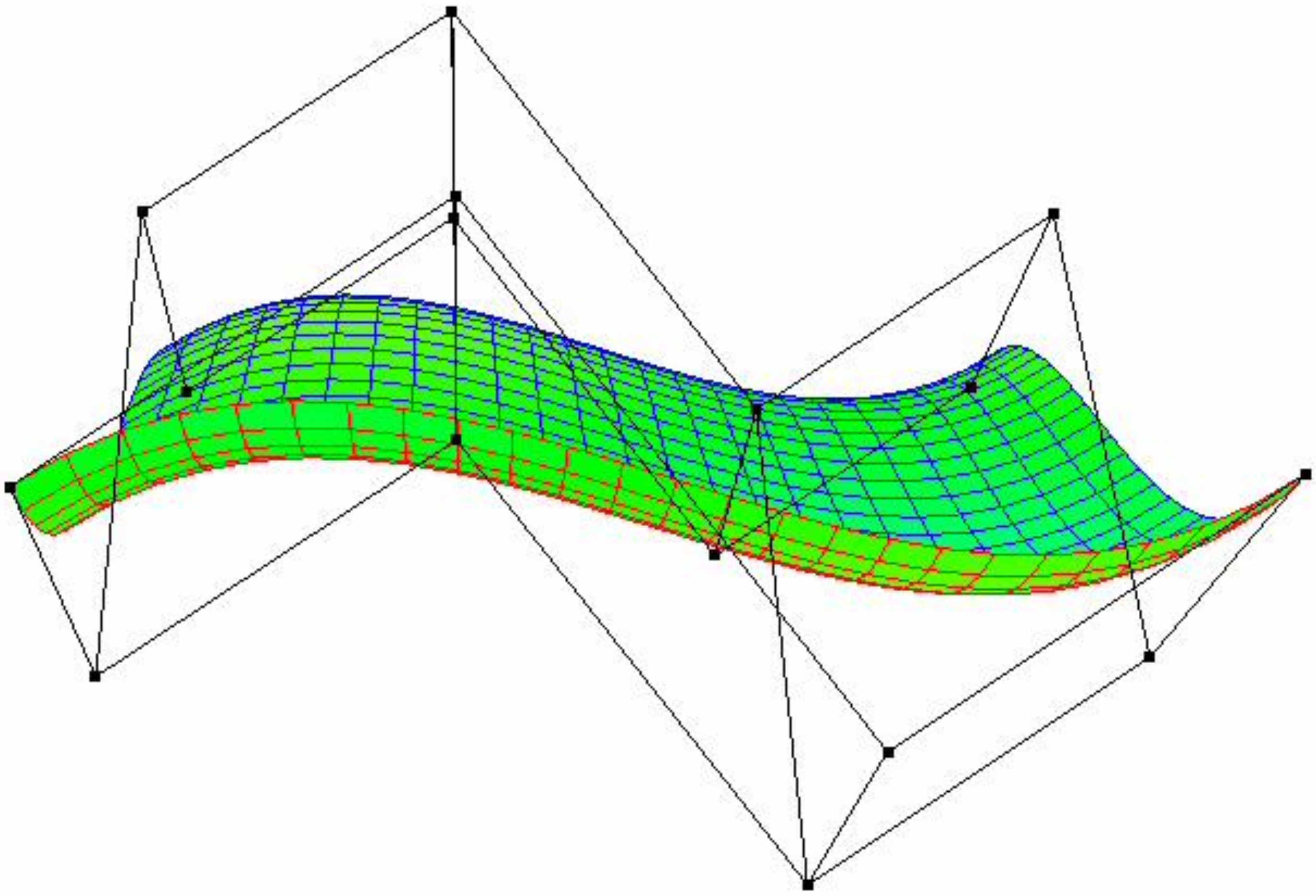




Freeform Surfaces: Examples



Freeform Surfaces: Examples



Trivia: The Utah Teapot

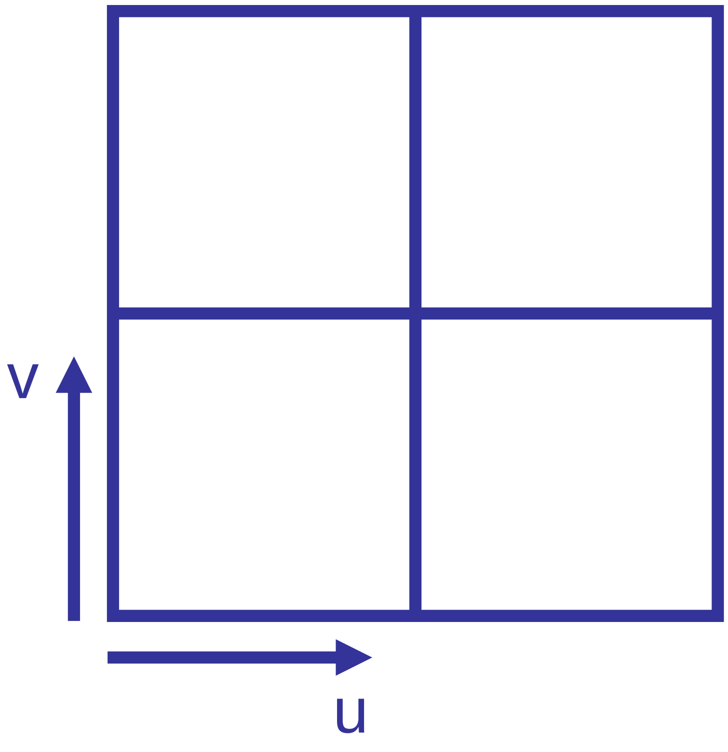
- famous model used early in CG
- modeled from Bézier patches in 1975
- is even available in GLUT
- used frequently in CG techniques as an example along with other “famous” models like the Stanford bunny

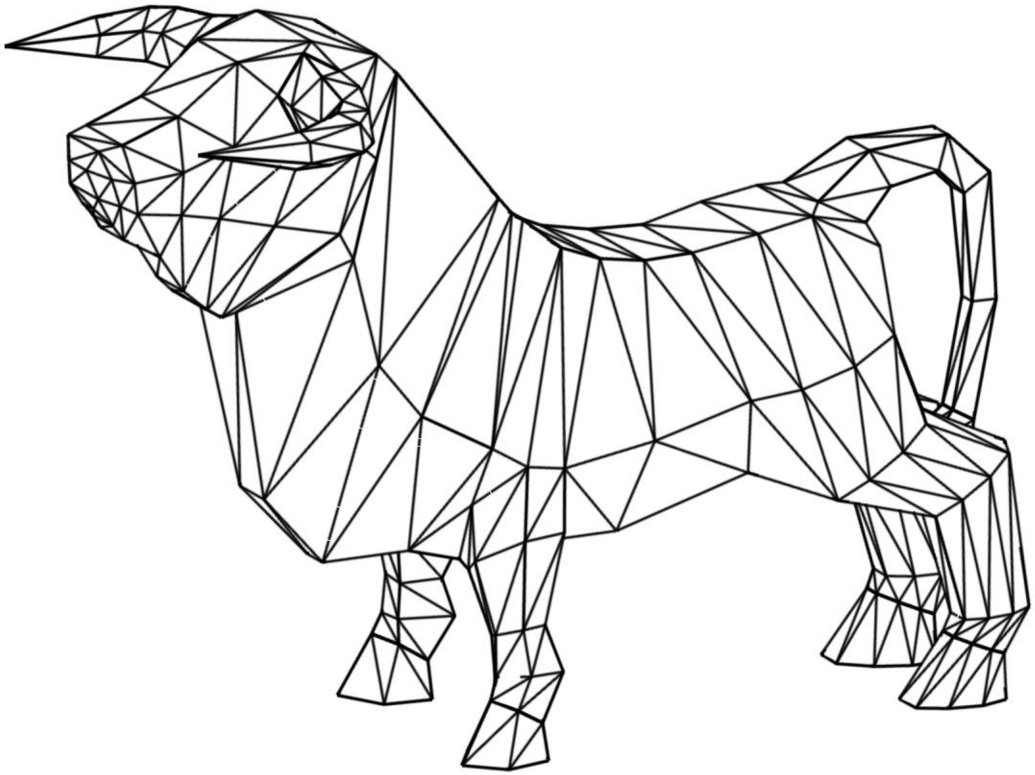
The Utah Teapot

Freeform Surfaces: How to Render?

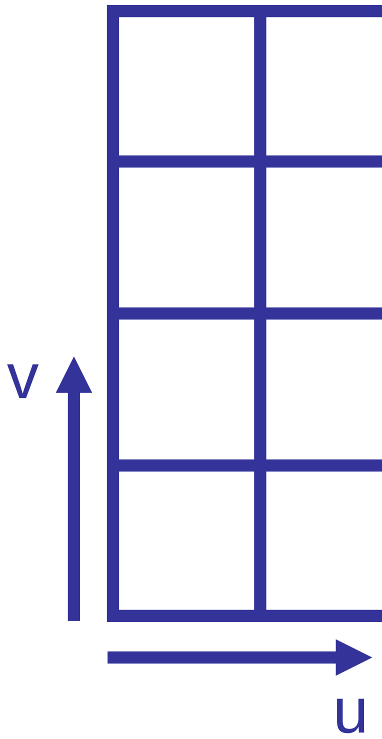
- freeform surface specification yields
  - points on the surface (evaluating the sums)
  - order of points (through parameter order)
- extraction of approximate polygon mesh
  - chose parameter stepping size in u and v
  - compute the points for each of the steps
  - create polygon mesh using the inherent order
- can be created as detailed as necessary

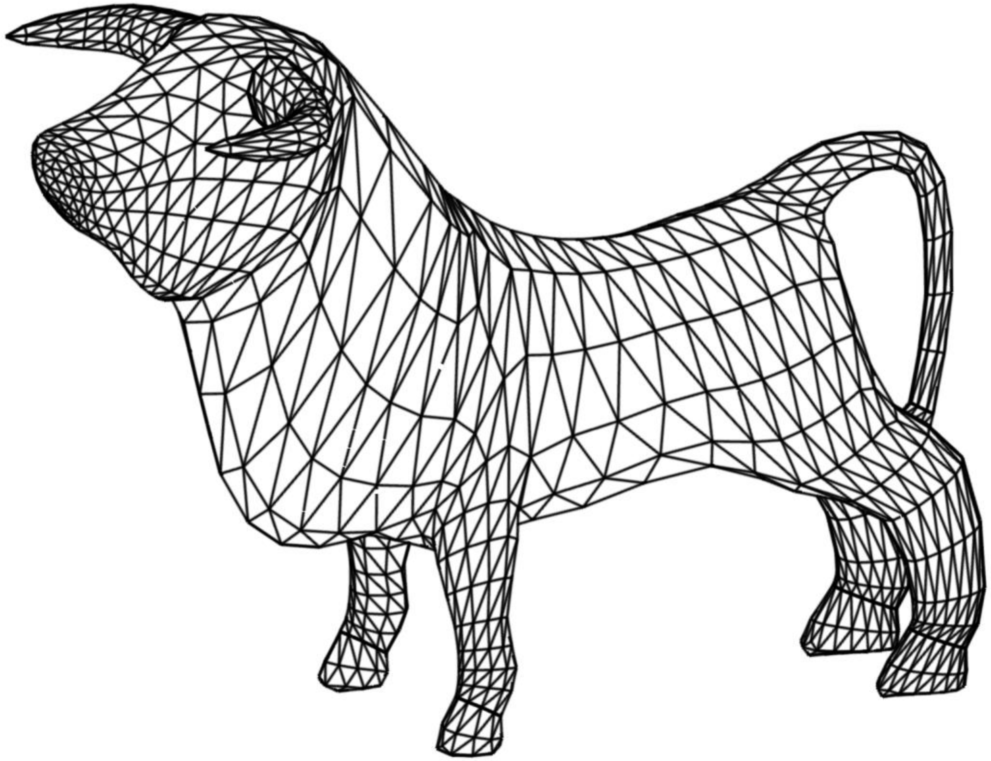
Tessellation (parameter space sampling)



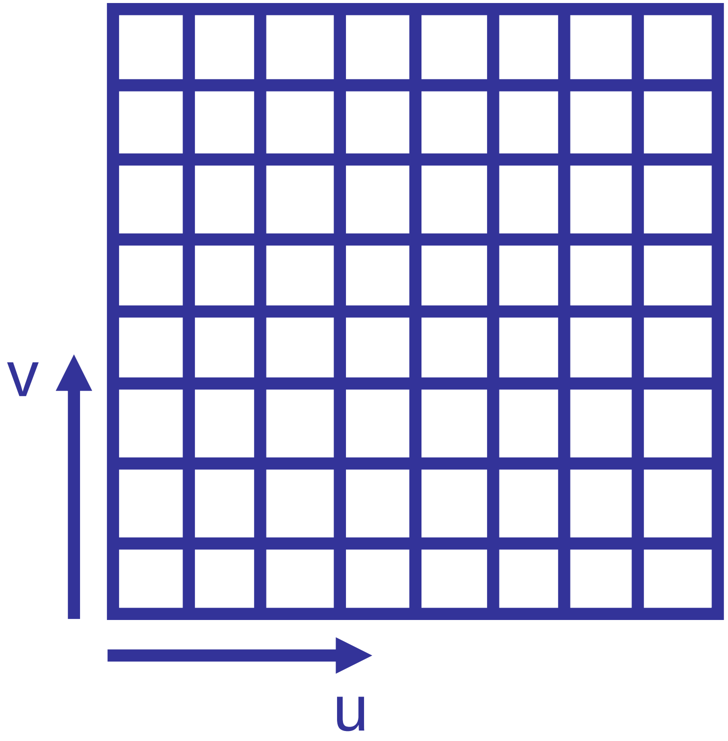


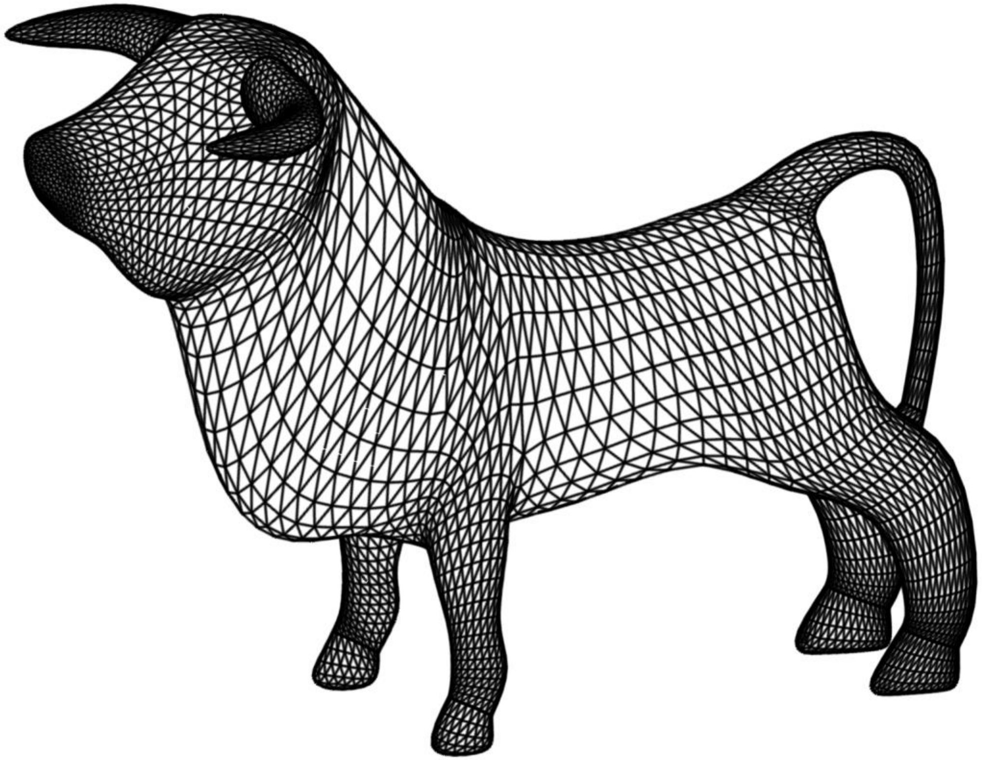
Tessellation (parameter space sampling)





Tessellation (parameter space sampling)





Curves and Smooth Surfaces

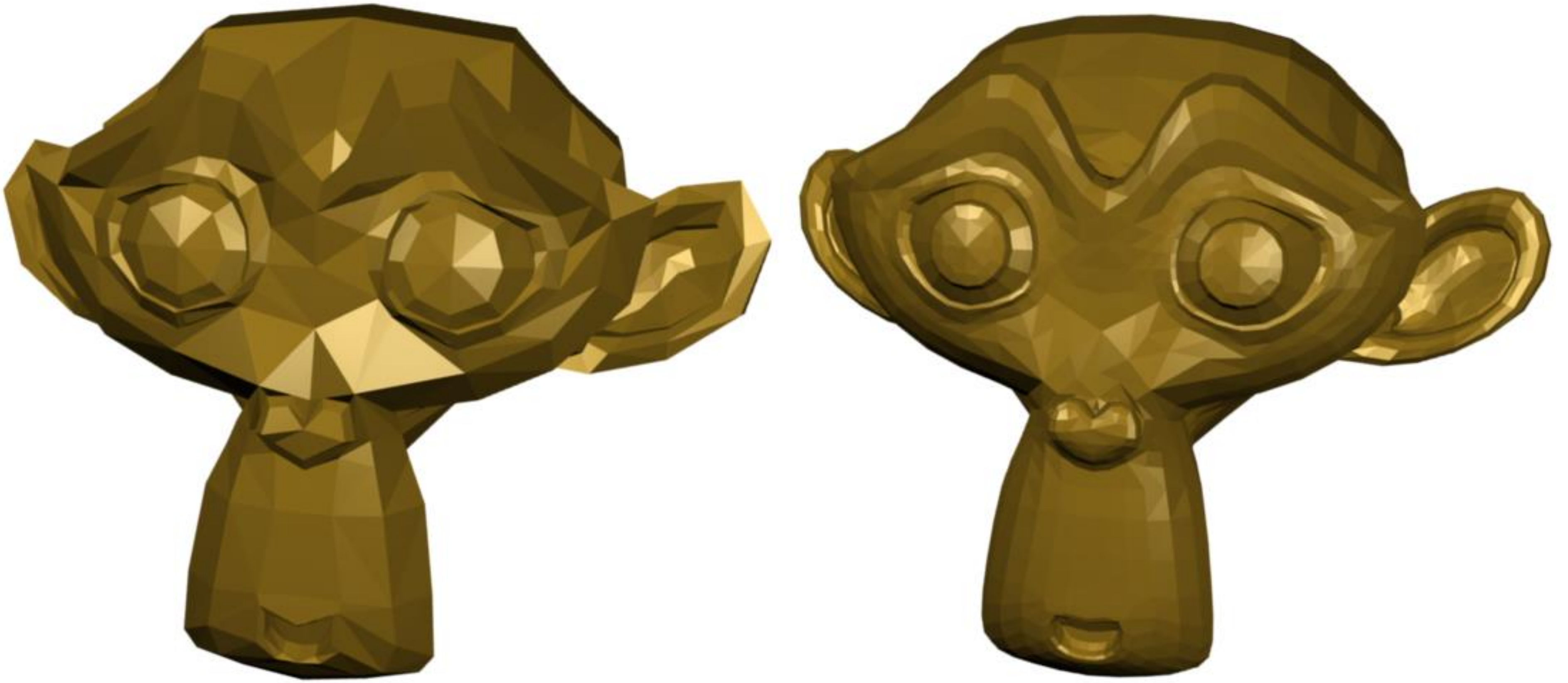
Subdivision Surfaces

Subdivision Surfaces

- but we already have so many polygon models, is there anything we can do?
- sure there is: **subdivision surfaces**!
- basic idea:
  - model coarse, low-resolution mesh of object
  - recursively refine the mesh using rules
  - use high-resolution mesh for rendering
  - limit surface should have continuity properties and is typically one of the freeform surfaces

Subdivision Surfaces: Example

Subdivision Surfaces: Example



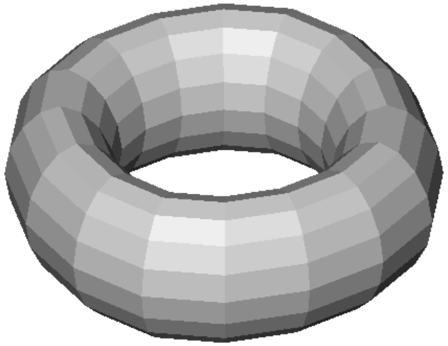
Subdivision Surfaces: Example

Subdivision Surfaces: Example

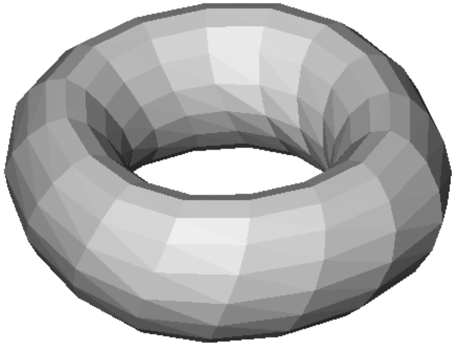
Subdivision Surfaces: Example

Subdivision Schemes for Surfaces

- quad-based vs. triangle-based subdivision
- quad-based subdivision
  - Doo-Sabin
  - Catmull-Clark
  - Kobbelt

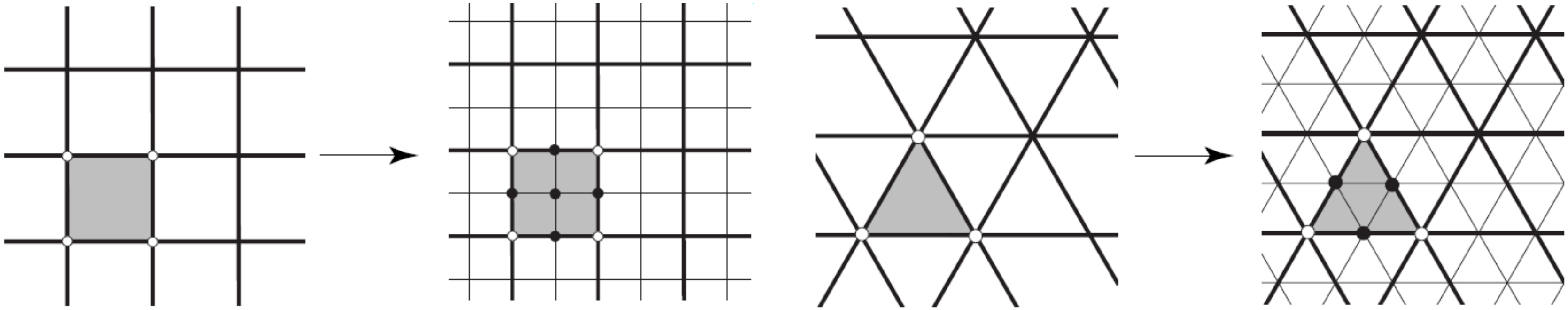


- triangle-based subdivision
  - Loop
  - (modified) butterfly
  - √3

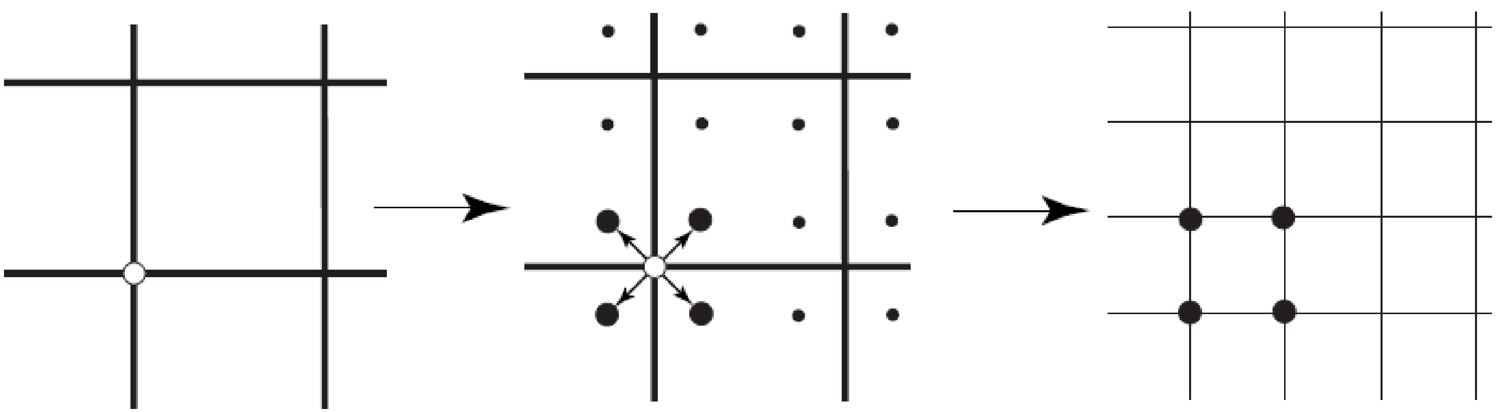


Face Splitting vs. Vertex Splitting

- face splitting: faces directly subdivided:



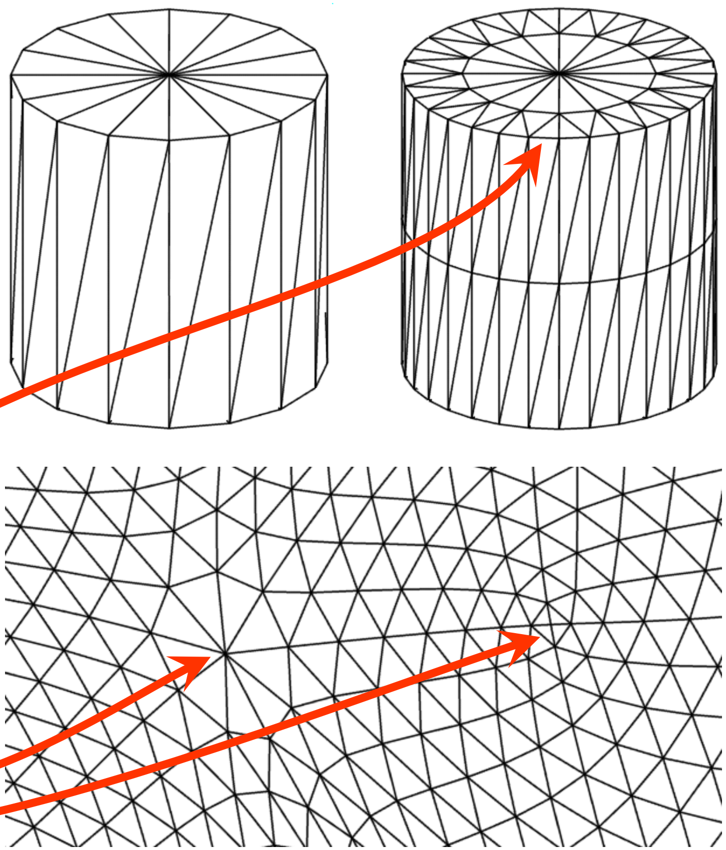
- vertex splitting: vertices are “split”



Zorin et al., 2000

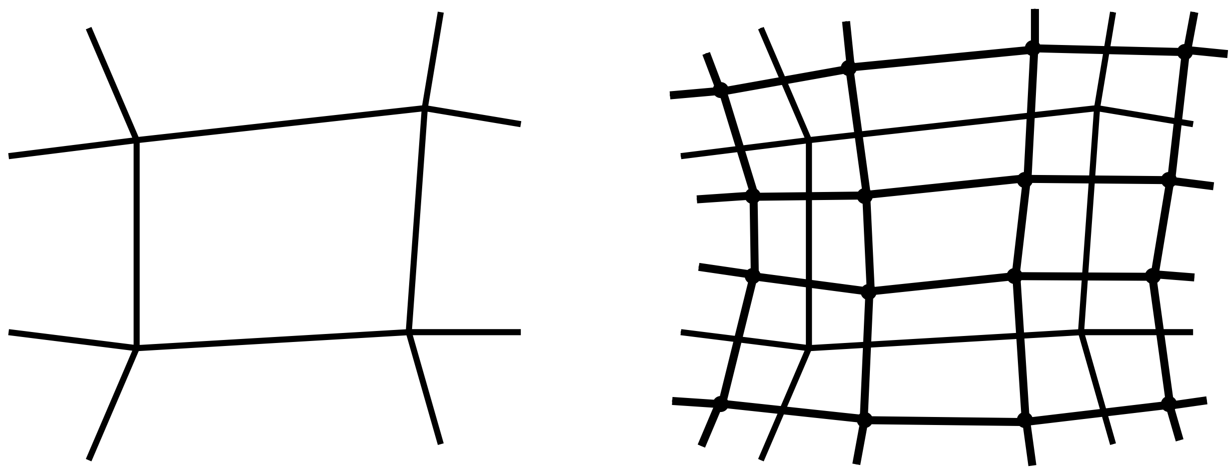
Position of New Vertices

- positions computed based on weighted averages from neighbouring original vertices or new vertices
- each scheme has its own weights (look up for implementation)
- special weights for sharp edges or borders
- extraordinary vertices

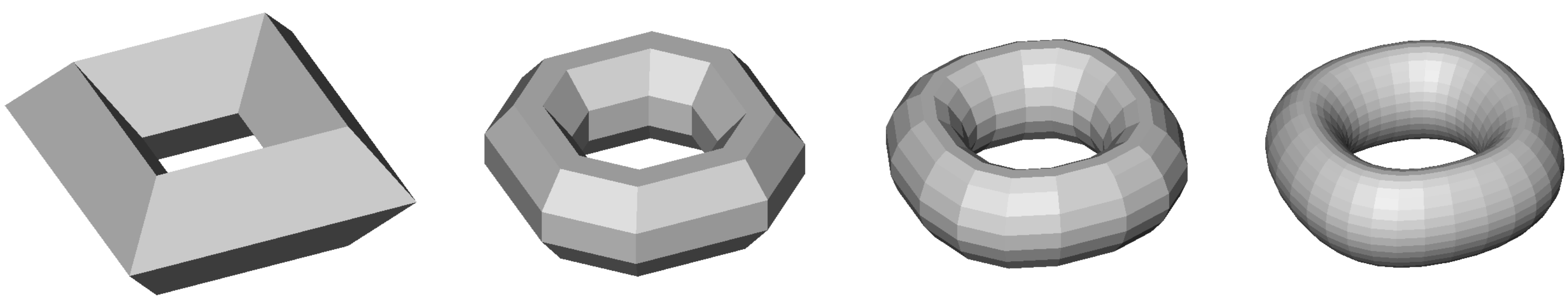


Doo-Sabin Subdivision

- approximating (quad mesh) vertex split

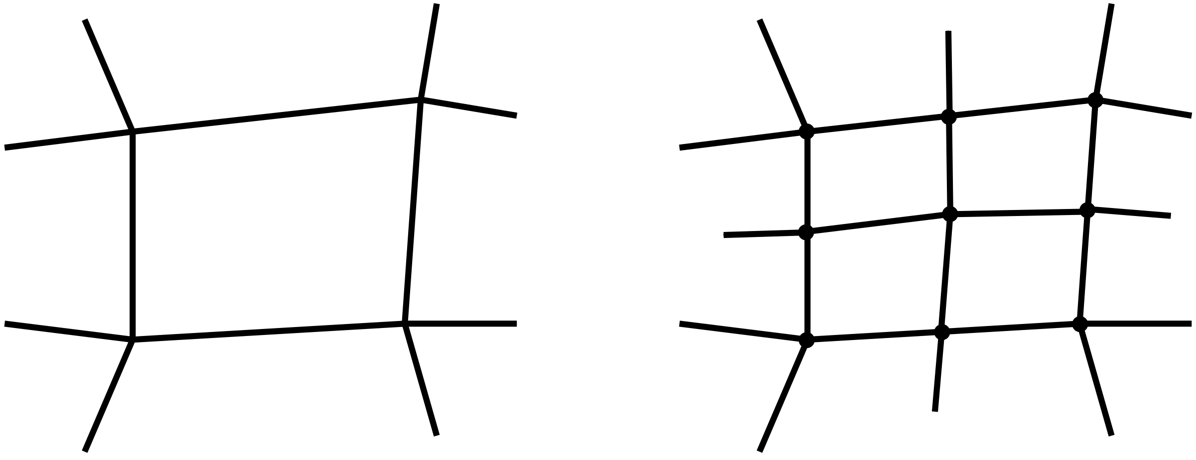


- example:

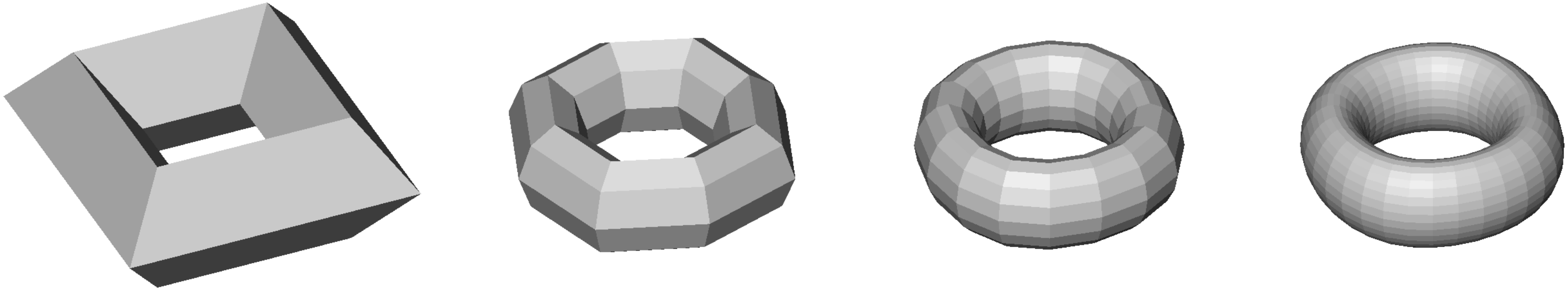


Catmul-Clark Subdivision

- approximating quad mesh face-split

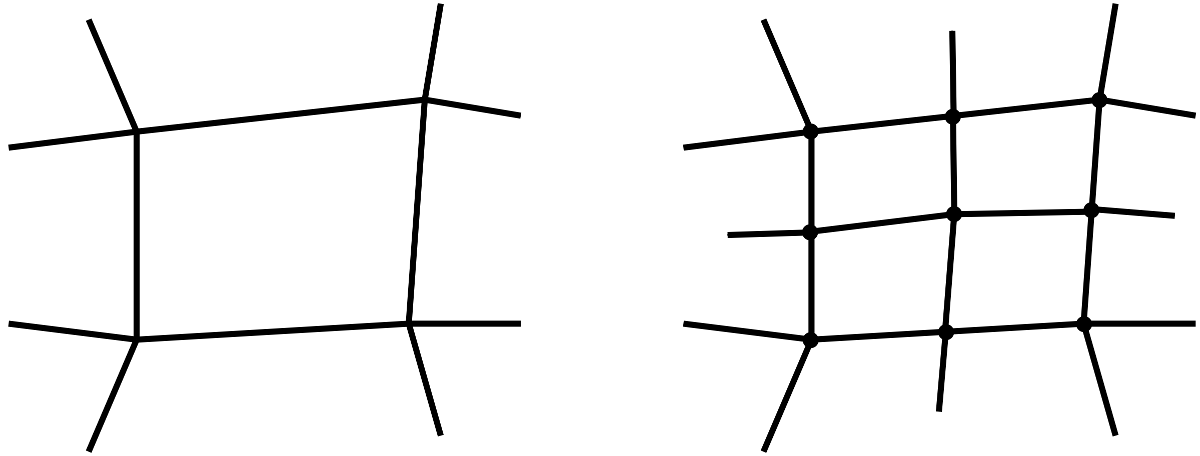


- example:



Kobbelt Subdivision

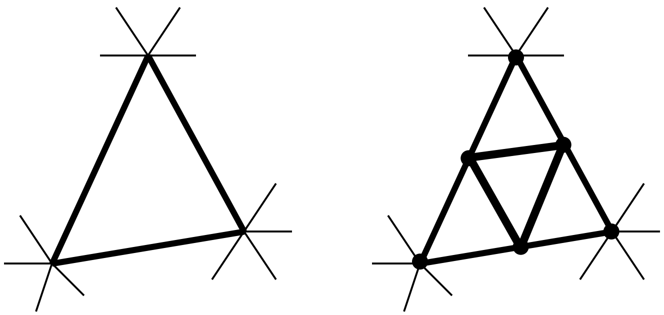
- interpolating quad-mesh face-split

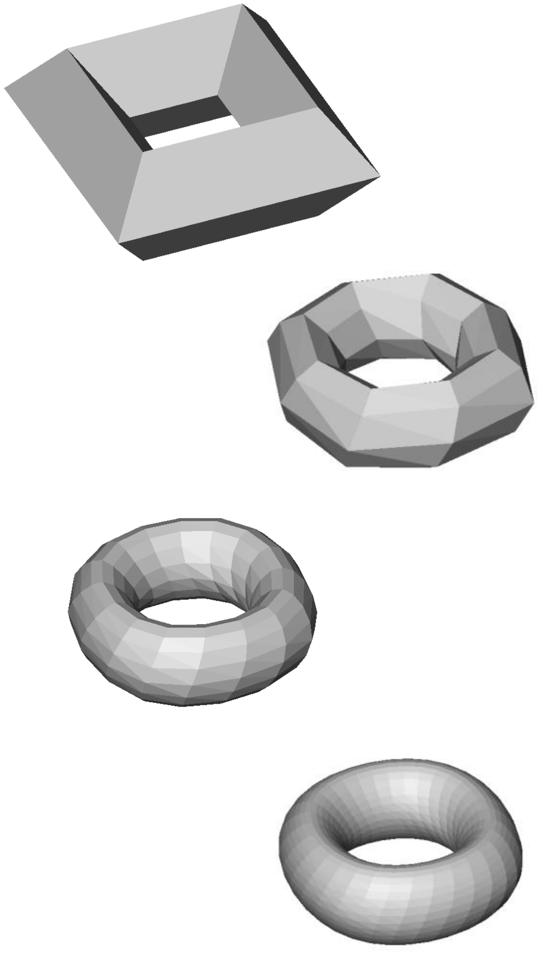


- using different weights than the Catmull-Clark scheme

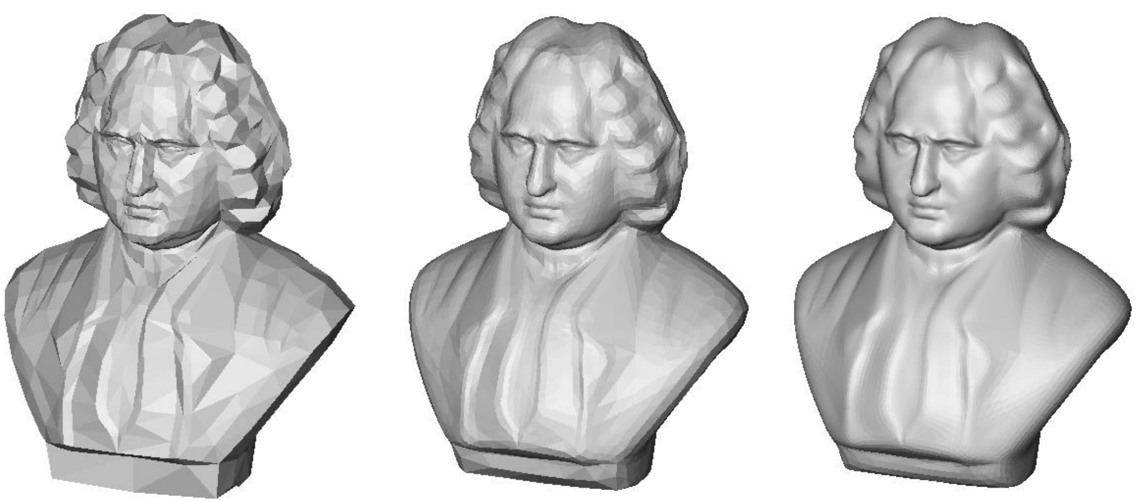
Loop Subdivision

- approximating triangle mesh face-split



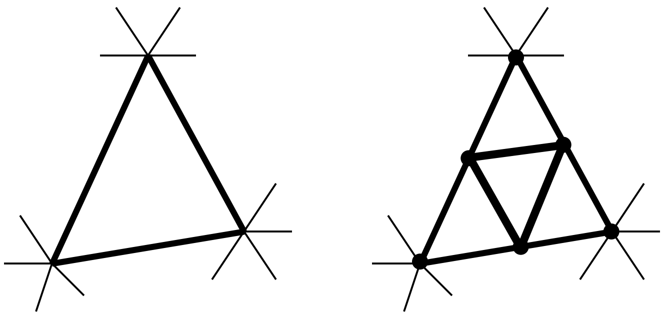


- example:



Modified Butterfly Subdivision

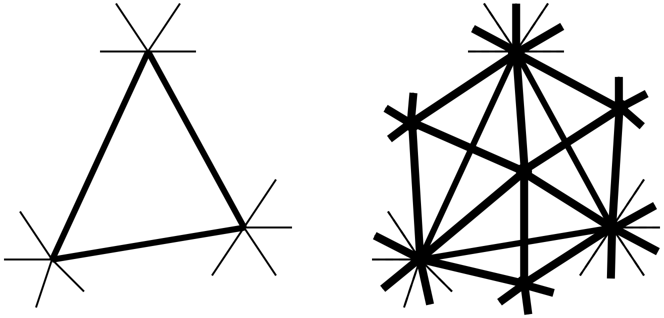
- interpolating triangle mesh face-split, using different weights compared to Loop scheme



- example:

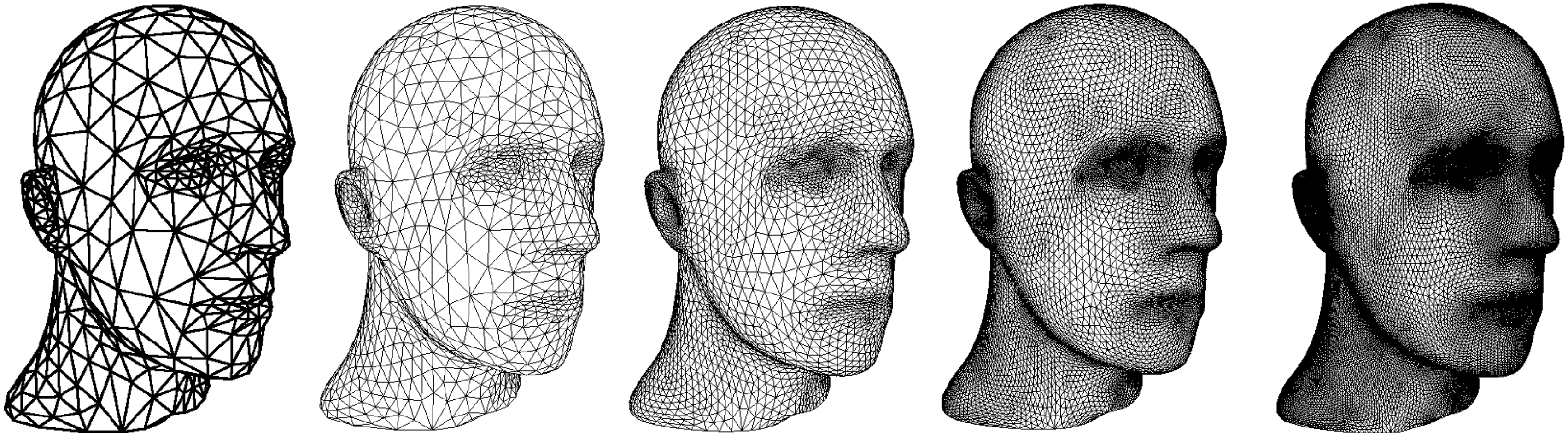
√3 subdivision

- approximating triangle mesh face-split



- only 1:3 triangle increase, not 1:4

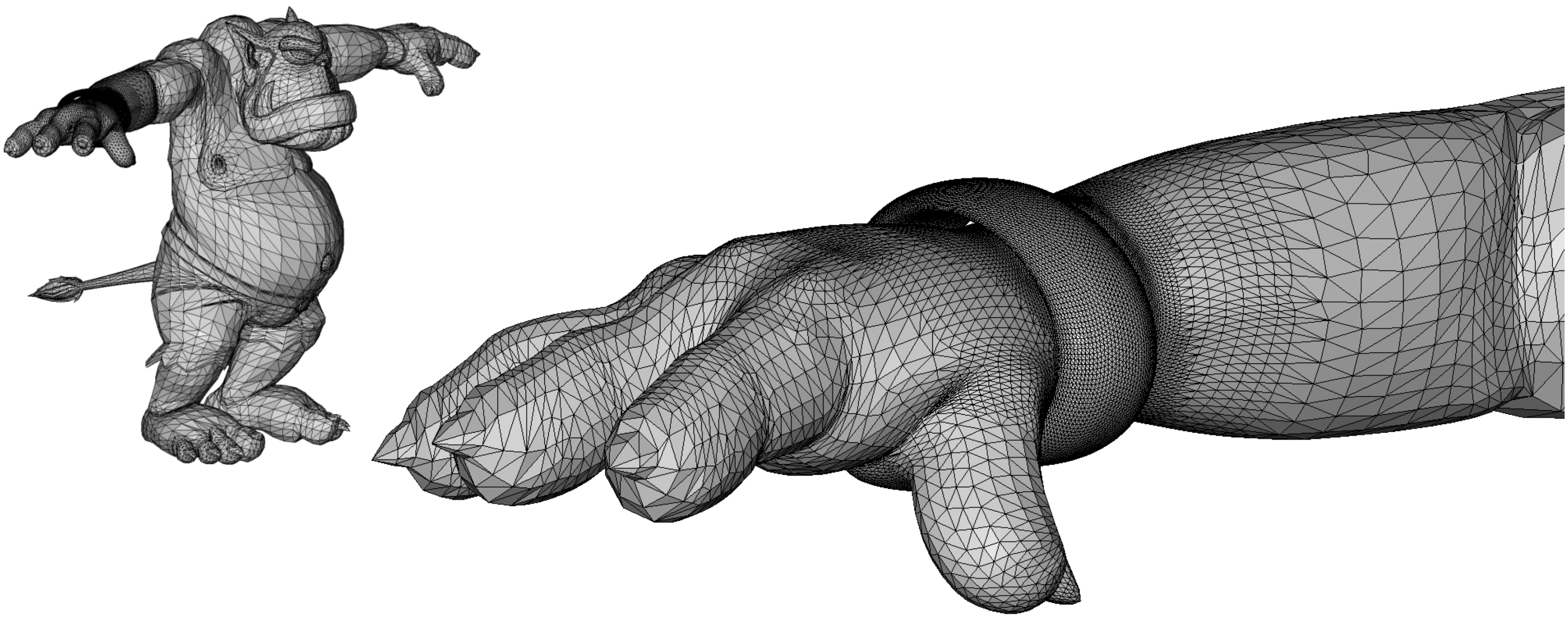
Loop scheme:



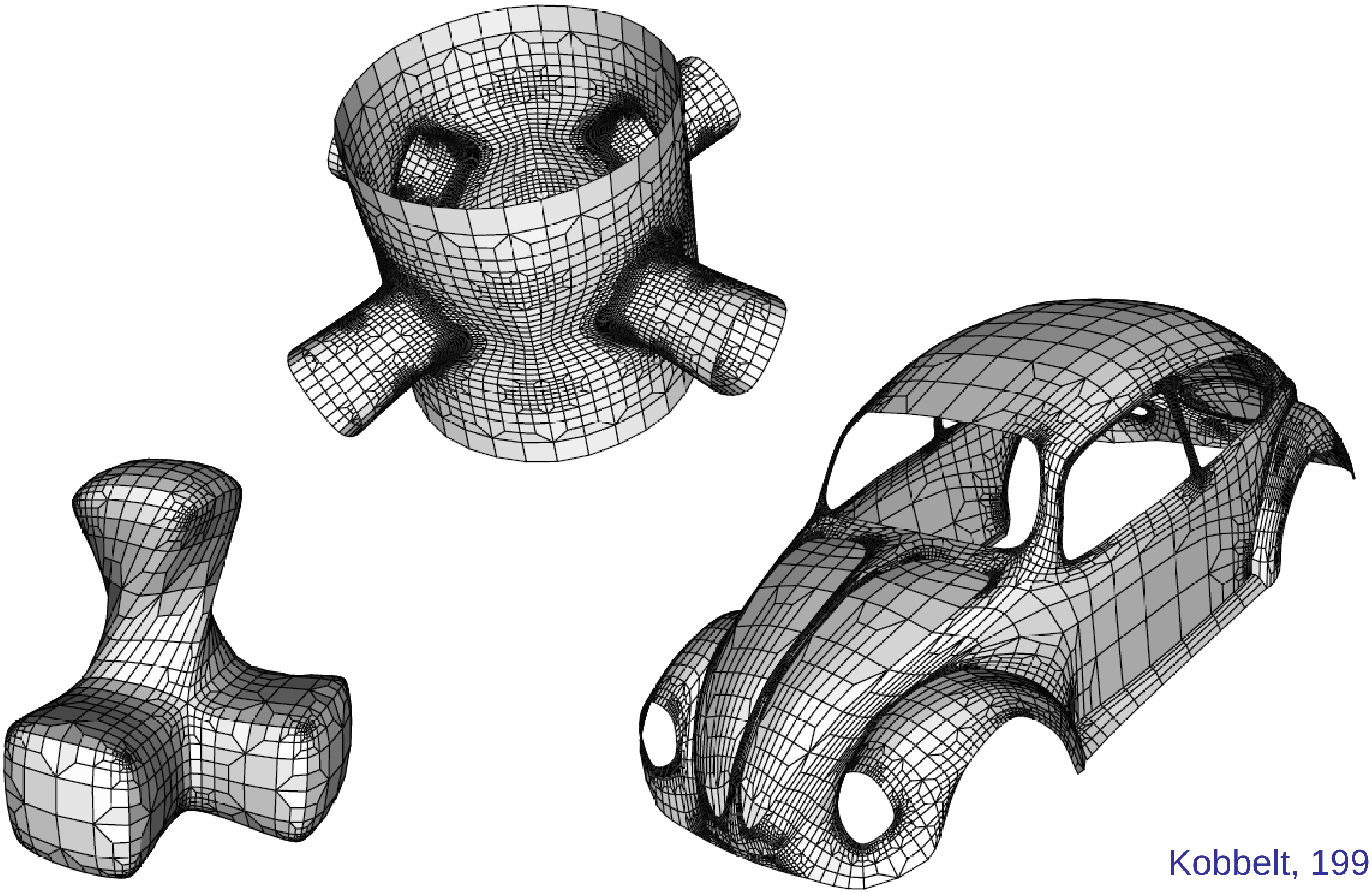
Kobbelt, 2000

Adaptive Subdivision

- subdivide only where detail is needed
- special care for boundary of subdivided region to maintain smooth transition



Adaptive Subdivision



Kobbelt, 1996

Subdivision and Freeform Surfaces

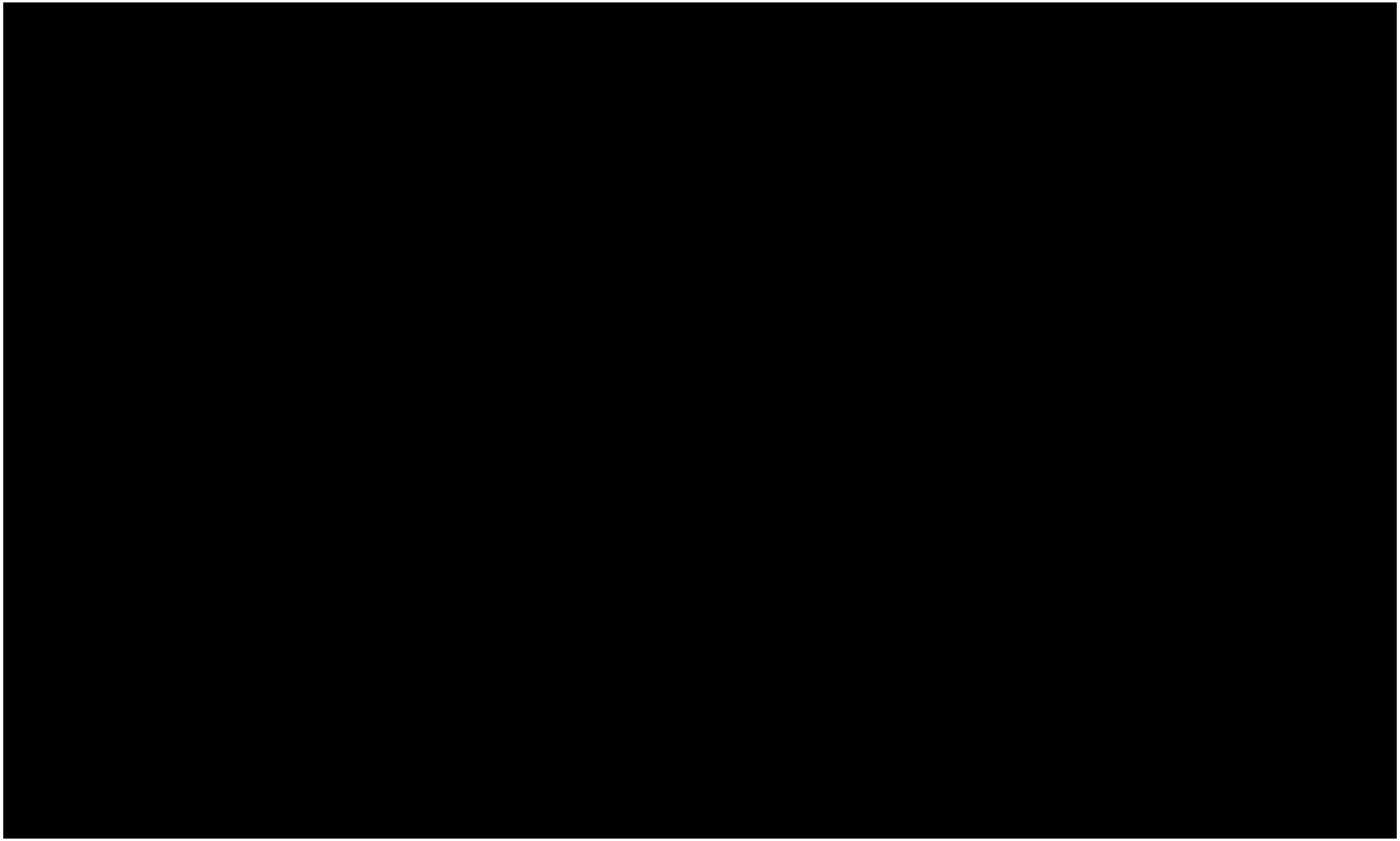
- limit surfaces of subdivision have also certain continuity properties:
  - C^1: Doo-Sabin, Kobbelt, Modified Butterfly
  - C^2: Loop, $\sqrt{3}$, Catmull-Clark
- for some schemes, the limit surfaces are Bézier/spline surfaces

Application: Subdivision Modeling

- model coarse meshes as usual
- apply subdivision to get smooth surfaces
- now used often in animated features to aid the modeling of characters and objects

Pixar, 1997 / DeRose et al., 1998

Intermission: Geri’s Game



Curves and Surfaces: Summary

- need to model smooth curves & surfaces
- use of control points
- polynomial descriptions
- continuity constraints C^n/G^n, important both for curves and surfaces
- surfaces from curves
- subdivision surfaces