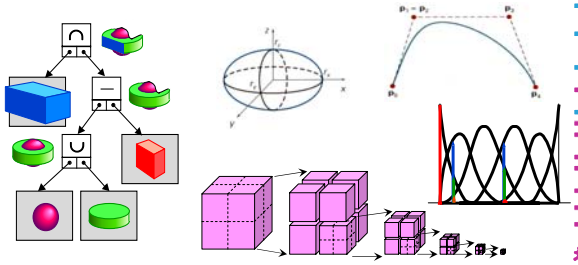
3D Object Representations

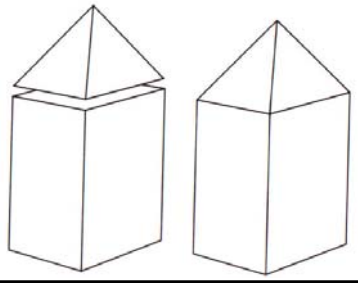


3D Object Representations

- graphics scenes contain
  - solid geometric objects
  - trees, flowers, clouds, rocks, water
- representations
  - surface ↔ interior models
  - procedural models
  - physically based models
- boundary representations (B-reps)
- space-partitioning representations

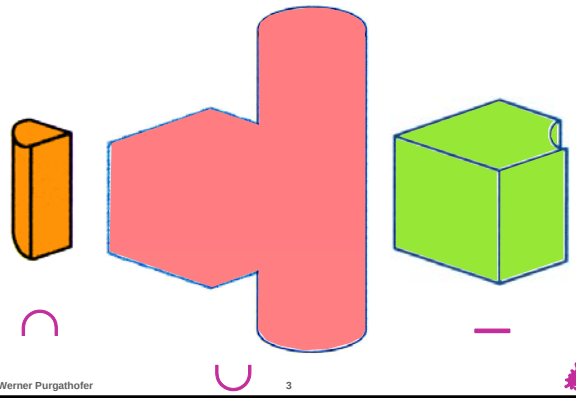
Constructive Solid Geometry

- **C**onstructive **S**olid **G**eometry (CSG)
  - boolean set operations on 3D objects
  - union, intersection, difference operation



combining 2 objects with a union operation, producing a single composite object

CSG: Different Set Operations

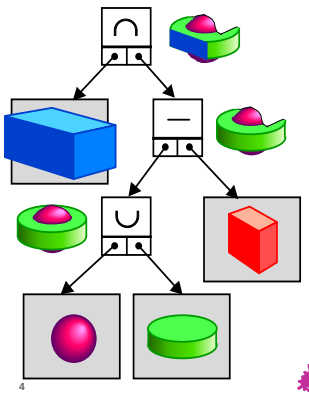


CSG Data Structure

Every object is assembled from simple solids with **set operations**

data structure: **binary tree**

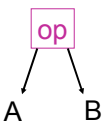
recursive evaluation



Operations with CSG Trees

- **transformations**
  - multiplication of all transformation matrices with the matrix of this transformation
- **combinations**
  - generate a new node with the desired operator and link the operands as subtrees to it

A op B:



Rendering of CSG Trees

- transform into B-Rep and use normal hidden surface algorithm

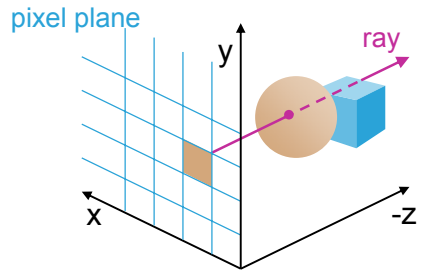
  or
- render directly with ray tracing

Properties of CSG

- advantages
  - exact representation
  - low memory cost
  - combinations and transformations trivial
- disadvantages
  - rendering effort is high

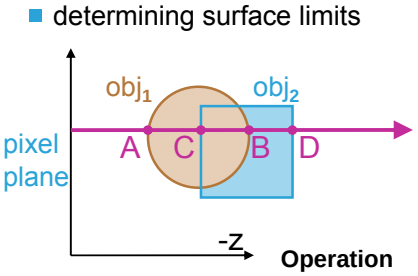
Ray-Casting Methods for CSG (1)

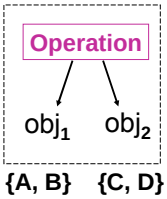
- visibility processing



Ray-Casting Methods for CSG (2)

- determining surface limits



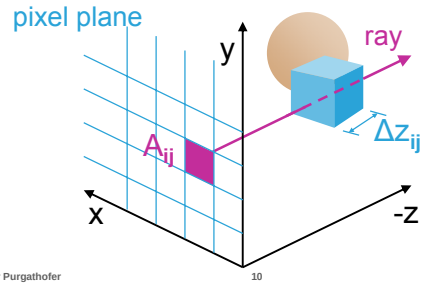


Operation	Result
Union	{A, D}
Intersection	{C, B}
Difference	{A, C}

Ray-Casting Methods for CSG (3)

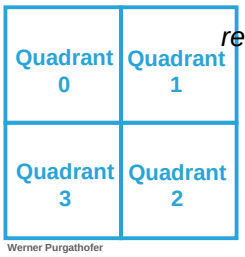
- volume determination

$V_{ij} \approx A_{ij} \cdot \Delta z_{ij}$ $V \approx \Sigma V_{ij}$



Quadtrees

- hierarchical enumeration of objects
- in 2D: quadtree
  - hierarchical subdivision until a region is homogeneous

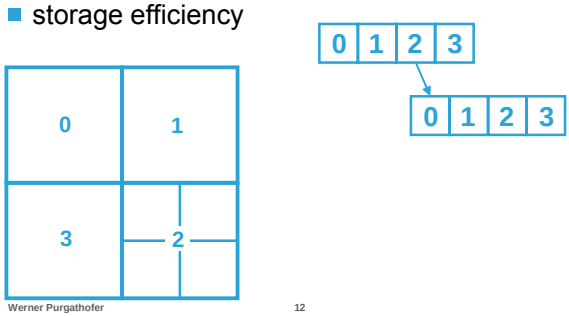


region of a 2-dim. space

data elements in the representative quadtree node

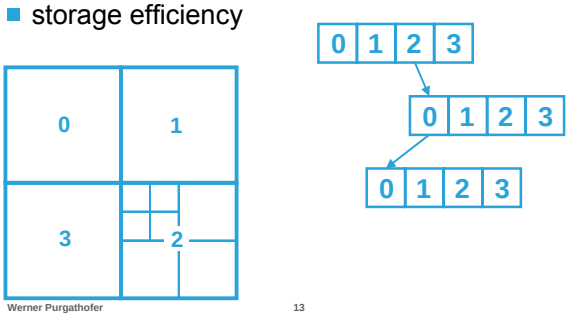
Quadtrees

- area with 2^n by 2^n pixels ⇒ quadtree with n levels
- storage efficiency



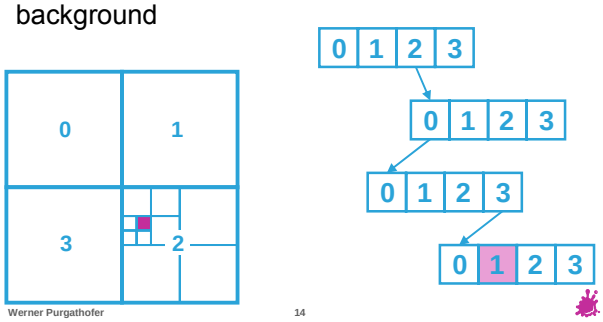
Quadtrees

- area with 2^n by 2^n pixels ⇒ quadtree with n levels
- storage efficiency

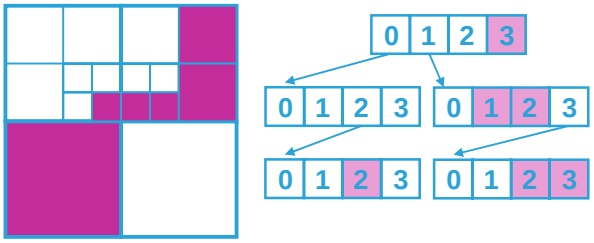


Quadtrees

quadtree representation for a region containing one foreground-color pixel on a solid background

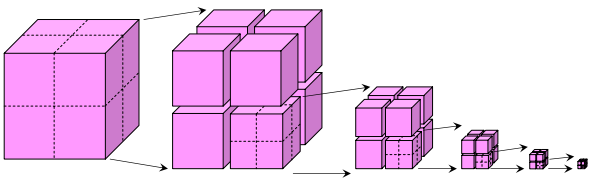


Quadtree Example



suitable for representing (2D) images

Octree

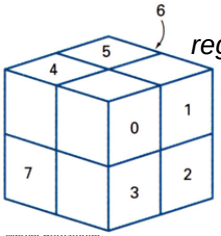


regular space subdivision:

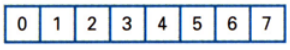
- simple (empty or uniform) ⇒ leaf node
- complex (other cases) ⇒ divide further

Octrees

- octree divides 3D cube into octants
- volume elements (voxels)
- set operations easy on octrees
- geometric transformations difficult on octrees

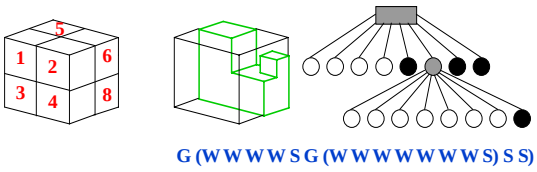


region of a 3-dim. space



data elements in the representative octree node

Octree Simple Example



G (W W W W S G (W W W W W W W W S) S S)

Operations with Octrees

- **transformations**
  - very complicated except for a few special cases,
    e.g. rotation by 90°, mirroring at a subdivision plane, scalation by 2^n
- **combinations**
  - very simple:
    if A or B homogeneous ⇒ simple rules
    else combine recursively all 8 octants of A and B

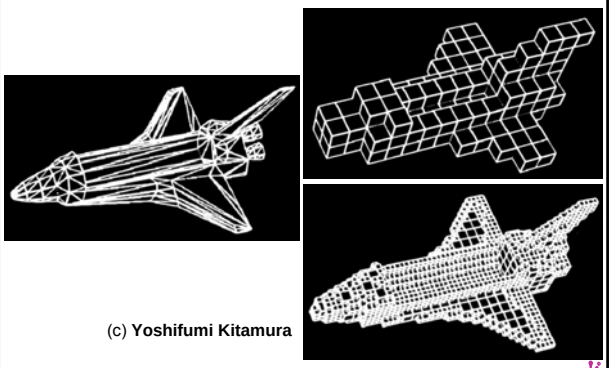
Rendering of Octrees

- rendering algorithm:
  - if octree node is *full:* draw the cube
  - if octree node is *empty:* do nothing
  - if octree node is *inhomogeneous:*
    render the 8 octants from back to front

Properties of Octrees

- **advantages**
  - every geometry can be represented
  - combinations very simple
  - fast rendering
  - spatial search possible
- **disadvantages**
  - inexact representation
  - low image quality
  - restricted transformations
  - high memory cost

Octree Example



(c) **Yoshifumi Kitamura**

Other 3D Object Representations

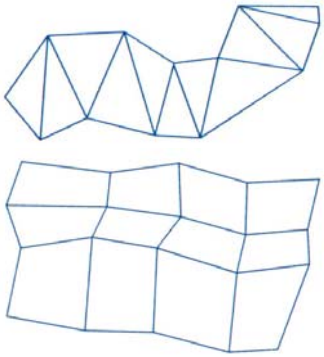
- BSP trees
- fractal geometry methods
- shape grammars, procedural models
- particle systems
- physically based modeling
- visualization of data sets
- ...

Curved Lines and Surfaces

- defined by
  - mathematical functions (implicit, explicit, parametrically)
  - set of data points (surface fitting)
- tesselation to get polygon mesh approximation
  - triangles
  - quadrilaterals ... (planar?!)

Polygon Meshes

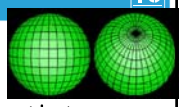
- efficient data structures for tiled surfaces
- triangle strip
  - n-2 triangles for n vertices
- quadrilateral mesh
  - (n-1)x(m-1) quadrilaterals



Quadric Surfaces

- defined by second degree equations (quadrics)
  - sphere
  - ellipsoid
  - torus
  - paraboloid
  - hyperboloid
  - ...

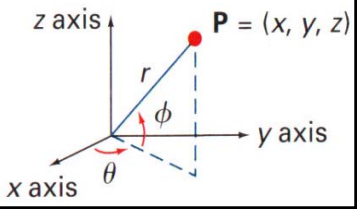
Quadric Surfaces: Sphere



- implicit: $x^2 + y^2 + z^2 = r^2$
- parametric:

$$x = r\cos\phi\cos\theta, \quad -\pi/2 \le \phi \le \pi/2$$
$$y = r\cos\phi\sin\theta, \quad -\pi \le \theta \le \pi$$
$$z = r\sin\phi$$

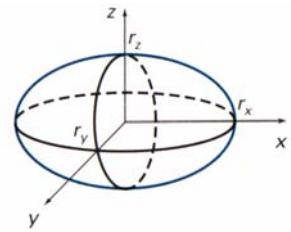
parametric coordinate position (r, θ, ϕ) on the surface of a sphere with radius r



Quadric Surfaces: Ellipsoid

- implicit:

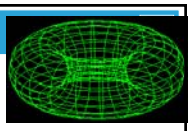
$$\left(\frac{x}{r_x}\right)^2 + \left(\frac{y}{r_y}\right)^2 + \left(\frac{z}{r_z}\right)^2 = 1$$

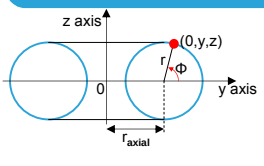


- parametric:

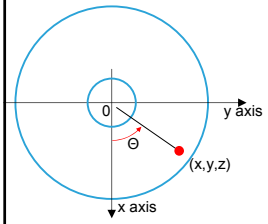
$$x = r_x\cos\phi\cos\theta, \quad -\pi/2 \le \phi \le \pi/2$$
$$y = r_y\cos\phi\sin\theta, \quad -\pi \le \theta \le \pi$$
$$z = r_z\sin\phi$$

Quadric Surfaces: Torus

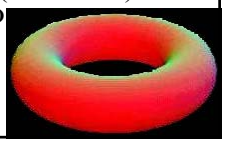




$$(\sqrt{x^2 + y^2} - r_{axial})^2 + z^2 = r^2$$



$$x = (r_{axial} + r\cos\Phi)\cos\Theta \quad (-\pi \le \Phi \le \pi)$$
$$y = (r_{axial} + r\cos\Phi)\sin\Theta \quad (-\pi \le \Theta \le \pi)$$
$$z = r\sin\Phi$$

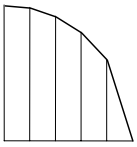


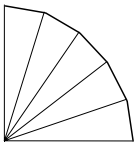
Free Form Surfaces

- can be represented by
  - **huge number of points (or polygons)**
    - \+ arbitrary shapes possible
    - – large memory requirements
    - – changes cause much work
    - – corners after scaling!
    - – modeling?
  - **mathematical functions**
    - – only for some shape categories
    - \+ marginal memory requirements
    - \+ changes are rather simple
    - \+ definition arbitrarily exact
    - • modeling!

Nonparametric ↔ Parametric

$y = f(x)$	$x = f(t)$ $y = g(t)$
axis dependent	*axis independent*
example: $y = \sqrt{1-x^2}$	$x=\cos(t)$ $y=\sin(t)$





Properties of Curves

interpolating or approximating control points?

degree of continuity at concatenations (C, G)

oscillatory behaviour compact or overswinging?

global or local influence of control points

axis (in)dependence (does the curve change when the coordinate system is rotated?)

multiple points possible? (for closed curves and corners)

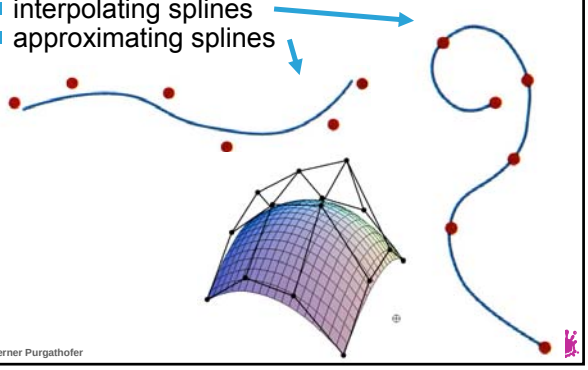
possible curve forms

Spline Representations

- spline curve
  - composite curve
  - polynomial sections, piecewise continuous
  - continuity conditions
- spline surface
  - two sets of orthogonal spline curves

Spline Curves

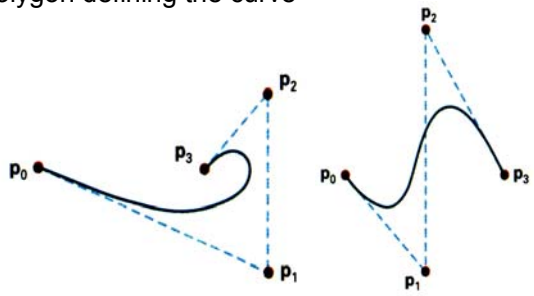
- spline specification with control points
- interpolating splines
- approximating splines



Splines: Control Polygon

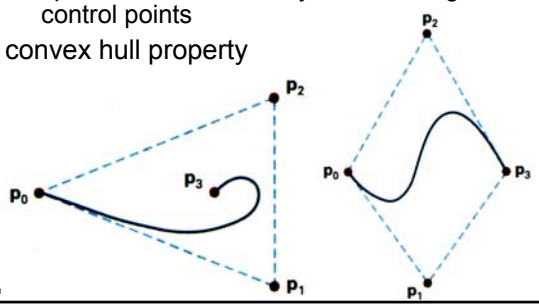
(also called „Characteristic Polygon“)

polygon defining the curve



Spline Properties

- operations on splines
  - move, insert control points
  - spline transformation by transforming all control points
- convex hull property

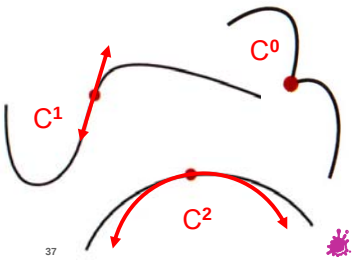


Spline: Continuity Conditions (1)

- *parametric* continuity conditions (C^n)
  - derivations at section joints are equal

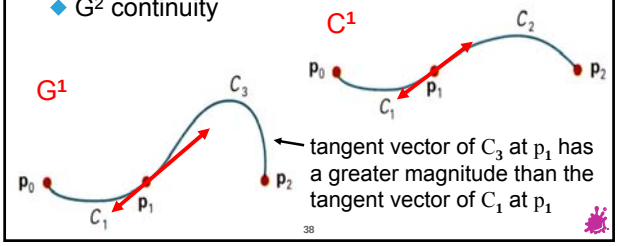
$$x = x(u) \quad y = y(u) \quad z = z(u) \quad u_1 \le u \le u_2$$

  - C^0 continuity
  - C^1 continuity
  - C^2 continuity



Spline: Continuity Conditions (2)

- *geometric* continuity conditions (G^n)
  - derivations at section joints are proportional
  - G^0 ($=C^0$) continuity
  - G^1 continuity (tangent vectors are collinear)
  - G^2 continuity



tangent vector of C_3 at p_1 has a greater magnitude than the tangent vector of C_1 at p_1

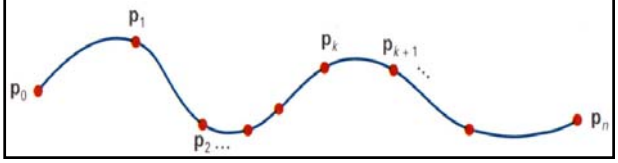
Cubic Spline Interpolation

- n+1 control points

$$p_k = (x_k, y_k, z_k) \quad k = 0,1,2,\ldots,n$$

- cubic polynomial $P_k(u)$ between each pair of control points

$$P_k(u) = a_k u^3 + b_k u^2 + c_k u + d_k$$
$$k = 0, 1, 2, \ldots, n-1, \qquad 0 \le u \le 1$$



Natural Cubic Splines

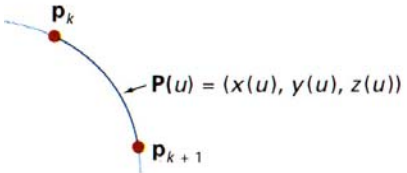
- adjacent curve segments have the same first and second derivative (C^2 continuity)
- solving an equation system with 4n variables
- two extra conditions required
  (e.g., $P_0''(0) = 0, \quad P_{n-1}''(1) = 0$)
- global influence of control points

$$P_k(0) = p_k, \qquad k = 0,\ldots,n-1$$
$$P_k(1) = p_{k+1}, \qquad k = 0,\ldots,n-1$$
$$P_k'(1) = P_{k+1}'(0), \qquad k = 0,\ldots,n-2$$
$$P_k''(1) = P_{k+1}''(0), \qquad k = 0,\ldots,n-2$$

Hermite Interpolation (1)

- tangent Dp_{k+1} specified at each control point
- local influence of control points

$$P_k(0) = p_k \qquad k = 0, \ldots, n-1$$
$$P_k(1) = p_{k+1}$$
$$P'_k(0) = Dp_k$$
$$P'_k(1) = Dp_{k+1}$$



Hermite Interpolation (2)

$$P_k(u) = a_k u^3 + b_k u^2 + c_k u + d_k \qquad 0 \le u \le 1$$

$$P_k(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix} \qquad P_k'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix}$$

$$\begin{aligned} P_k(0) &= p_k \\ P_k(1) &= p_{k+1} \\ P_k'(0) &= Dp_k \\ P_k'(1) &= Dp_{k+1} \end{aligned} \qquad \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix}$$

Hermite Interpolation (3)

$$\begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

$$\begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix} = \mathbf{M_H} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

Hermite matrix

Hermite Interpolation (4)

$$P_k(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix} \qquad \begin{bmatrix} a_k \\ b_k \\ c_k \\ d_k \end{bmatrix} = M_H \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

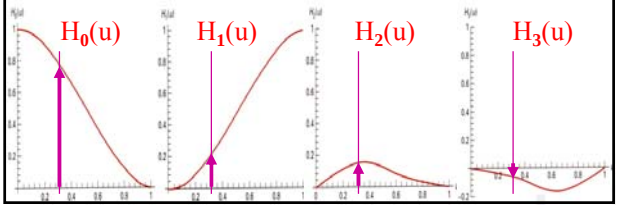
$$P_k(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \mathbf{M_H} \cdot \begin{bmatrix} p_k \\ p_{k+1} \\ Dp_k \\ Dp_{k+1} \end{bmatrix}$$

Hermite Interpolation (5)

$$P_k(u) = p_k(2u^3 - 3u^2 + 1) + p_{k+1}(-2u^3 + 3u^2) + \\ + Dp_k(u^3 - 2u^2 + u) + Dp_{k+1}(u^3 - u^2)$$

$$P_k(u) = p_k H_0(u) + p_{k+1} H_1(u) + Dp_k H_2(u) + Dp_{k+1} H_3(u)$$

- $H_k(u)$ blending functions:



Bézier Curves and Surfaces

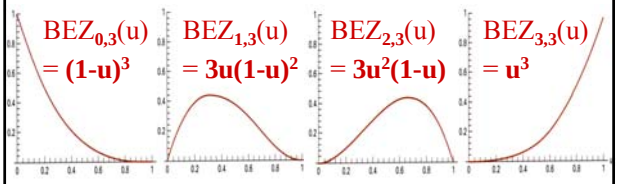
- spline approximation for points p_i , i=0,..., n

$$P(u) = \sum_{k=0}^{n} p_k BEZ_{k,n}(u) \qquad 0 \le u \le 1$$

- Bernstein polynomials

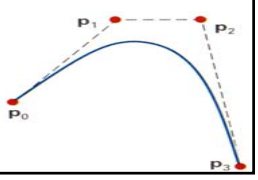
$$BEZ_{k,n}(u) = \binom{n}{k} u^k (1-u)^{n-k}$$

Cubic Bézier Blending Functions



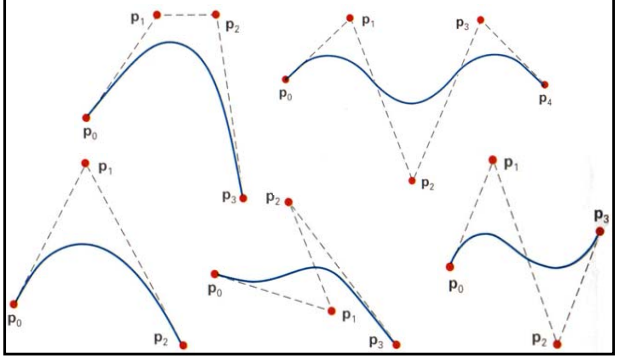
$$P(u) = (1-u)^3 \cdot p_0 + 3u(1-u)^2 \cdot p_1 + 3u^2(1-u) \cdot p_2 + u^3 \cdot p_3$$

the 4 Bézier blending functions for cubic curves (n=3)



2-Dim. Bézier Curves Examples

generated from 3, 4, and 5 control points



Bézier Curves Properties

- P(u) polynomial of degree n, global influence
- P(u) interpolates start and endpoint

$$P(0) = p_0, \quad P(1) = p_n$$

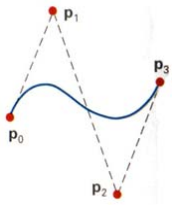
- tangents at start and endpoint

$$P'(0) = -np_0 + np_1$$

$$P'(1) = -np_{n-1} + np_n$$

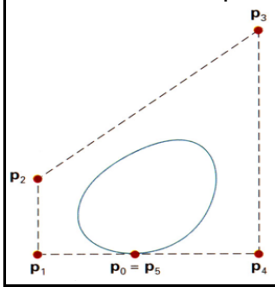
- convex hull property

$$\sum_{k=0}^{n} BEZ_{k,n}(u) = 1$$

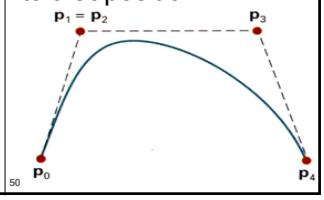


Bézier Curves Design Techniques (1)

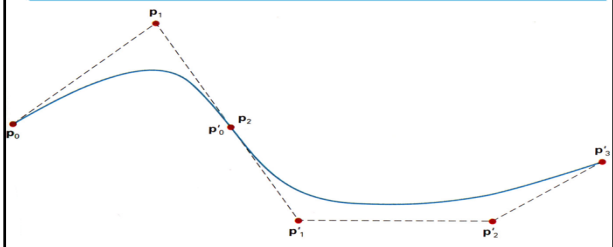
a ***closed Bézier curve*** generated by setting: first = last control point



a Bézier curve can be made to pass closer to a given coordinate position by assigning ***multiple control points*** to that position



Bézier Curves Design Techniques (2)



piecewise approximation curve formed with 2 Bézier sections. 0-order and 1st-order continuity (C^0, C^1 continuity) are attained by setting p0' = p2 and by making p1, p2, and p1' collinear.

Cubic Bézier Curve Matrix Notation

$$P(u) = (1-u)^3 \cdot p_0 + 3u(1-u)^2 \cdot p_1 + 3u^2(1-u) \cdot p_2 + u^3 \cdot p_3$$

$$P(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot M_{Bez} \cdot \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix}$$

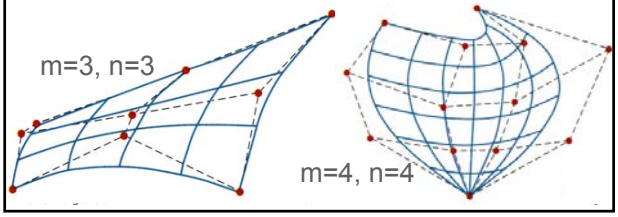
$$M_{Bez} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bézier Surfaces Definition

- Cartesian product of two Bézier curve bundles

$$P(u,v) = \sum_{j=0}^{m} \sum_{k=0}^{n} p_{j,k} BEZ_{j,m}(v) BEZ_{k,n}(u)$$

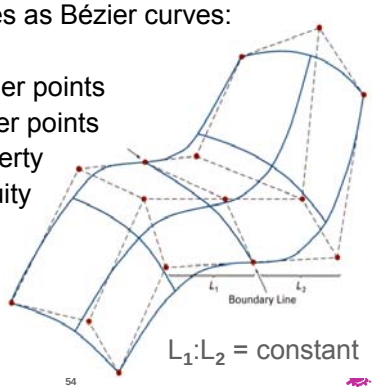
- $p_{j,k}$ grid of (m+1)x(n+1) control points



Bézier Surfaces Properties

the same properties as Bézier curves:

- global influence
- interpolates corner points
- tangents at corner points
- convex hull property
- 1st-order continuity connections



L_1:L_2 = constant

B-Spline Curves and Surfaces

- spline approximation for points p_i , i=0,..., n

$$P(u) = \sum_{k=0}^{n} p_k B_{k,d}(u) \qquad u_{min} \le u \le u_{max}$$

$$2 \le d \le n+1$$

- B-Spline blending functions
  - recursive Cox-deBoor formulas

B-Spline Basis Functions

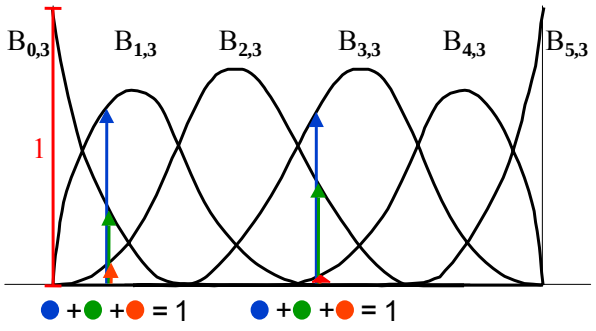
$$B_{k,1}(u) = \begin{cases} 1 & \text{if } u_k \le u \le u_{k+1} \\ 0 & \text{otherwise} \end{cases}$$

$$B_{k,d}(u) = \frac{(u-u_k)\cdot B_{k,d-1}(u)}{u_{k+d-1} - u_k} + \frac{(u_{k+d}-u)\cdot B_{k+1,d-1}(u)}{u_{k+d} - u_{k+1}}$$

$$\text{for } 0 \le u \le n-d+2$$

$$u_k = \begin{cases} 0 & \text{for } k < d \\ k-d+1 & \text{for } d \le k \le n \\ n-d+1 & \text{for } k > n \end{cases}$$ global, do not change

B-Spline Functions for d=3



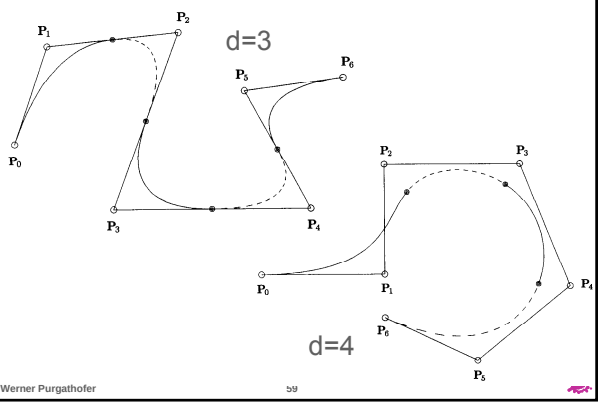
Important Property of the $B_{k,d}$

- for all B-Spline basis functions the following property holds:

$$\sum_{k=0}^{n} B_{k,d}(u) = 1 \qquad \text{for all } u$$

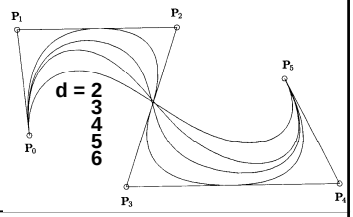
⇒ every curve point is a weighted mean of the control points

2-Dim. B-Spline Examples



Influence of d

- d describes, how many control points influence every point on the curve
  - d = 2 linear
  - d = 3 quadratic
  - d = 4 cubic
  - ...



- for **d=n+1** you get **Bézier** curves!

Differences B-Spline ↔ Bézier

- control points have **local influence** ⟶
- effort only **linearly dependent on n**, therefore splitting of huge point sets not necessary
- further extension: **N**on**U**niform **R**ational **B**-**S**plines = "NURBS"