Geometric Theory, Algorithms, and Techniques

Hong Qin

Department of Computer Science

State University of New York at Stony Brook

Stony Brook, New York 11794--4400

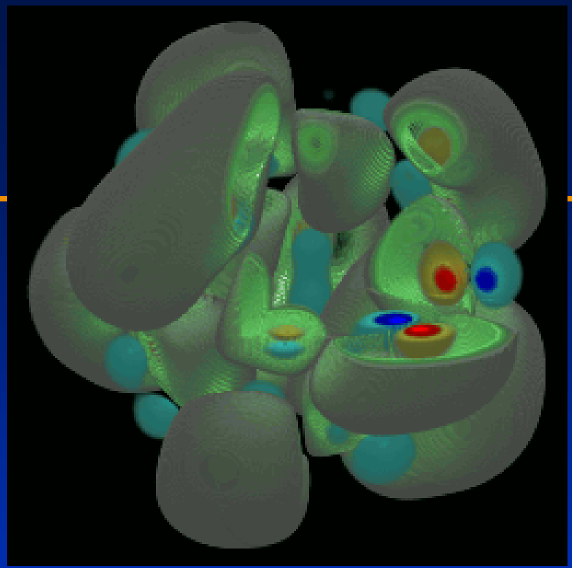
Tel: (631)632-8450; Fax: (631)632-8334

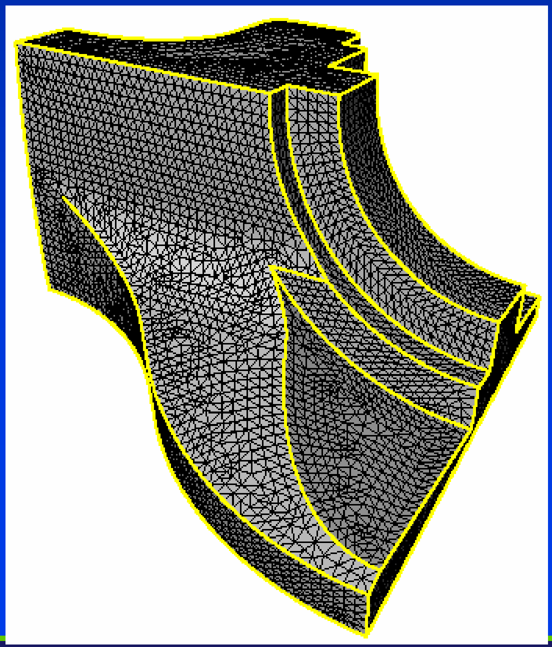
qin@cs.sunysb.edu

http://www.cs.sunysb.edu/~qin

Introduction

- Geometric modeling and visual computing
  - Computer graphics
    - Visualization, animation, virtual reality
  - CAD/CAM
    - Engineering, manufacturing
  - Computer vision
  - Physical simulation
  - Natural phenomena

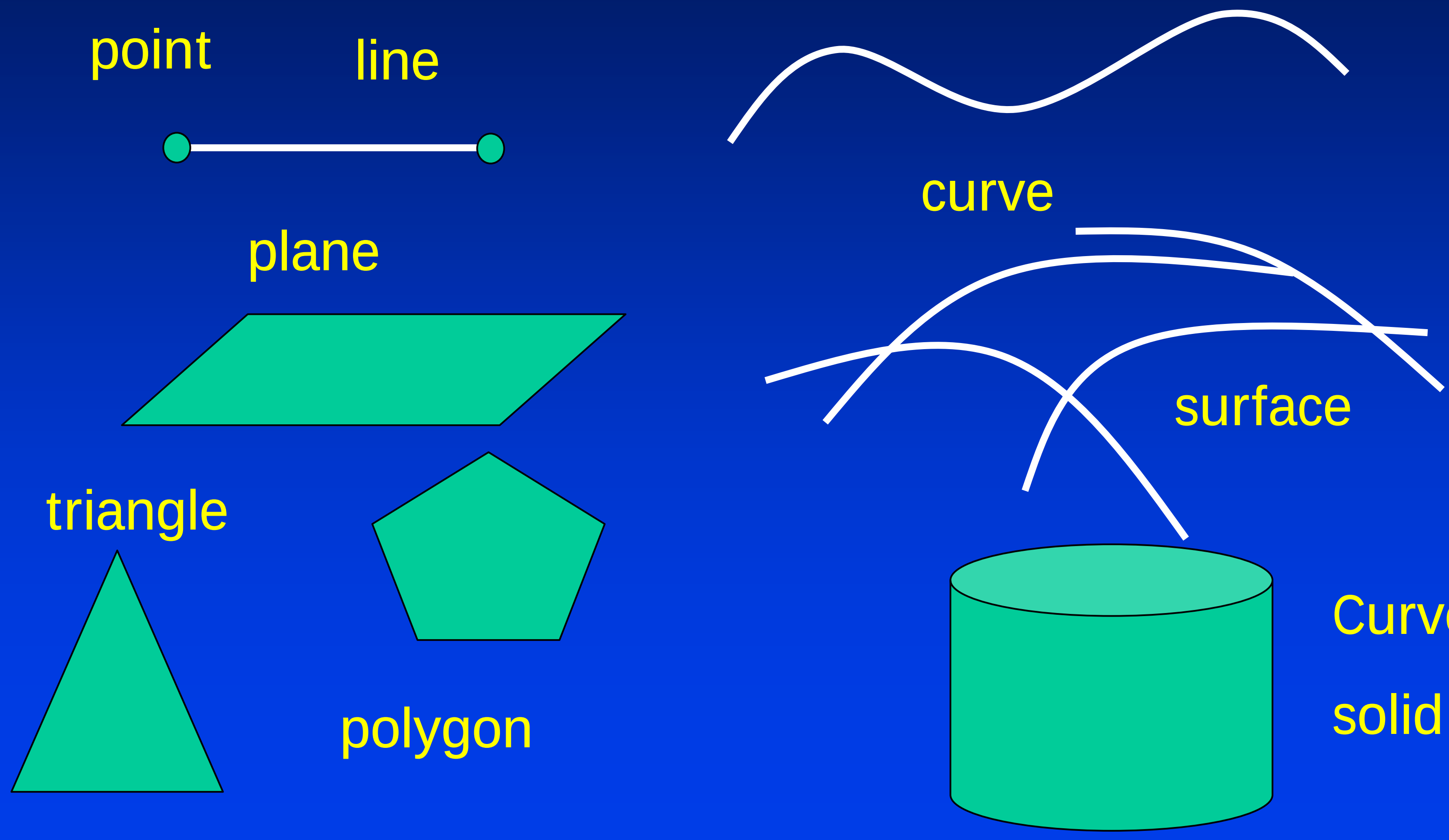




3D Shape Representation

- Points (vertices), a set of points
- Lines, polylines, curve
- Triangles, polygons
- Triangular meshes, polygonal meshes
- Analytic (commonly-used) shape
- Quadric surfaces, sphere, ellipsoid, torus
- Superquadric surfaces, superellipse, superellipsoid
- Blobby models

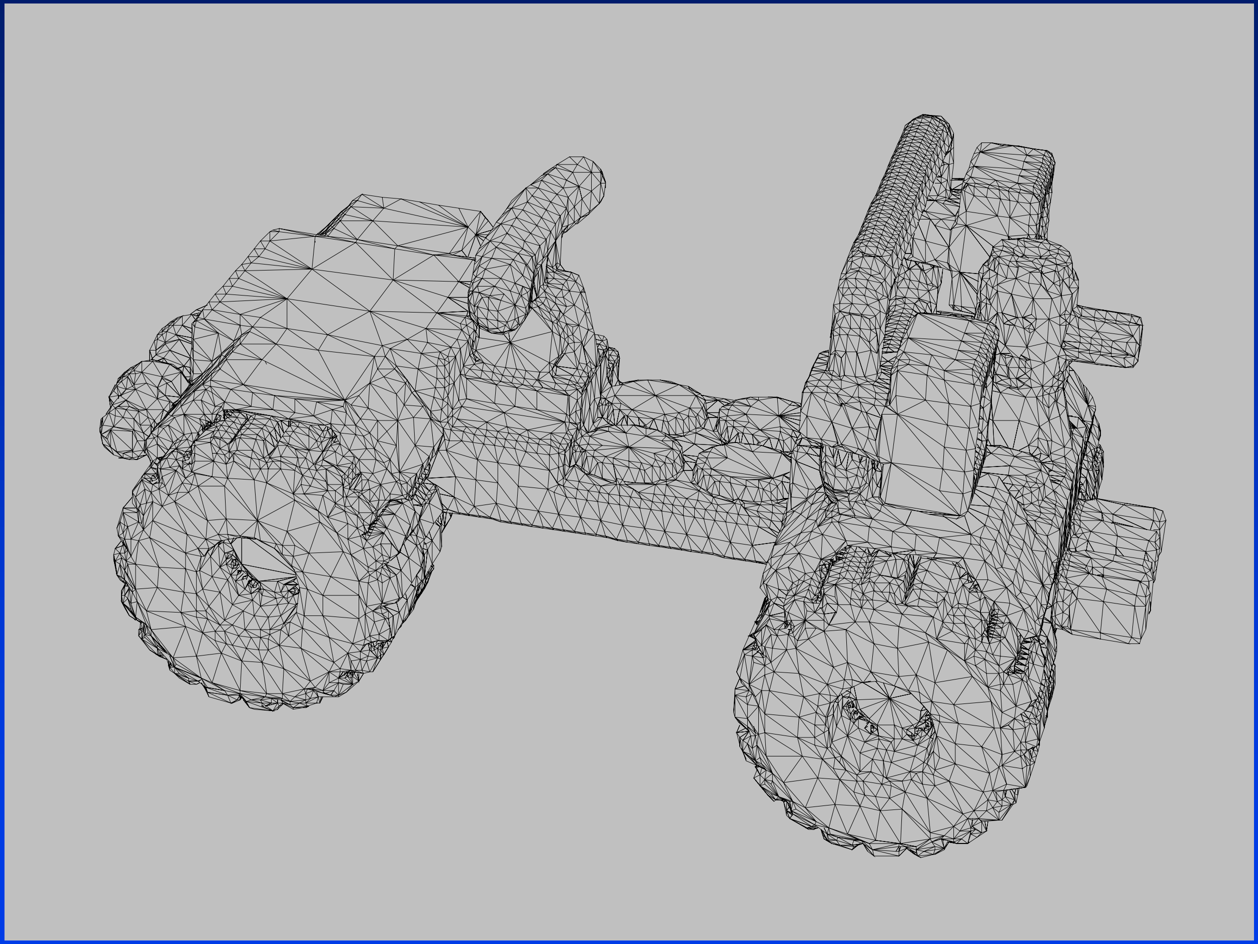
Basic Shapes



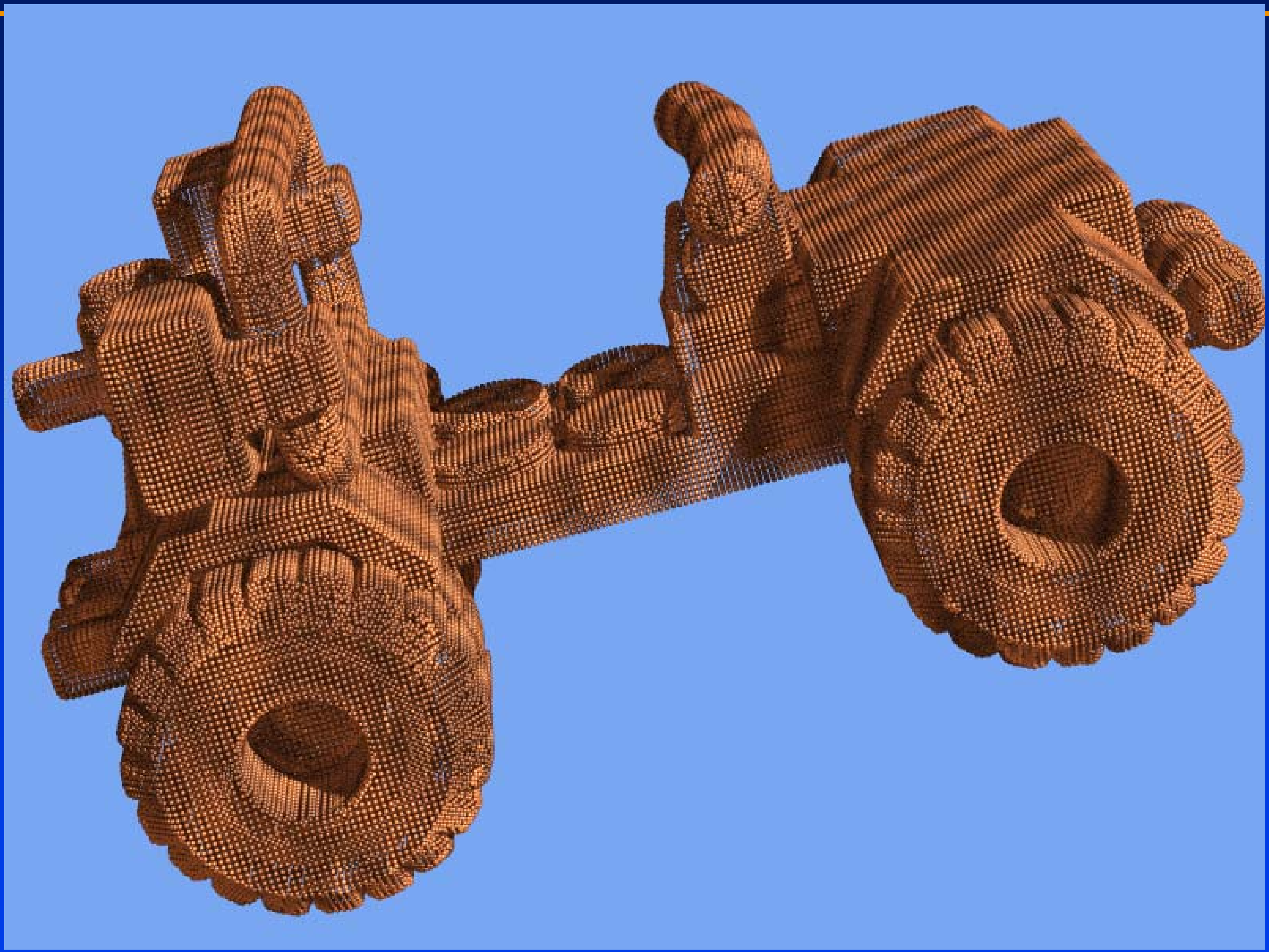
Fundamental Shapes

- Vertex (vertices)
- Line segments
- Triangle, triangular meshes
- Quadrilateral
- Polygon
- Curved object
- Tetrahedron, pyramid, hexahedron
- Many more…

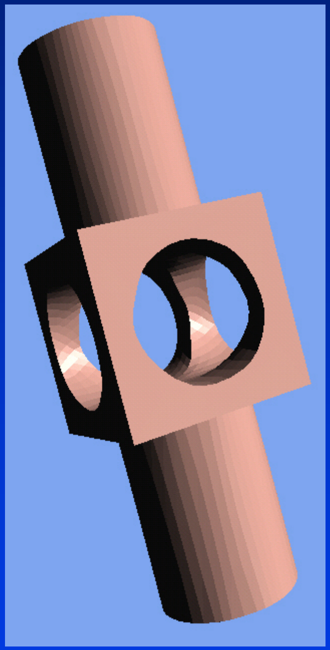
Polygonal Meshes

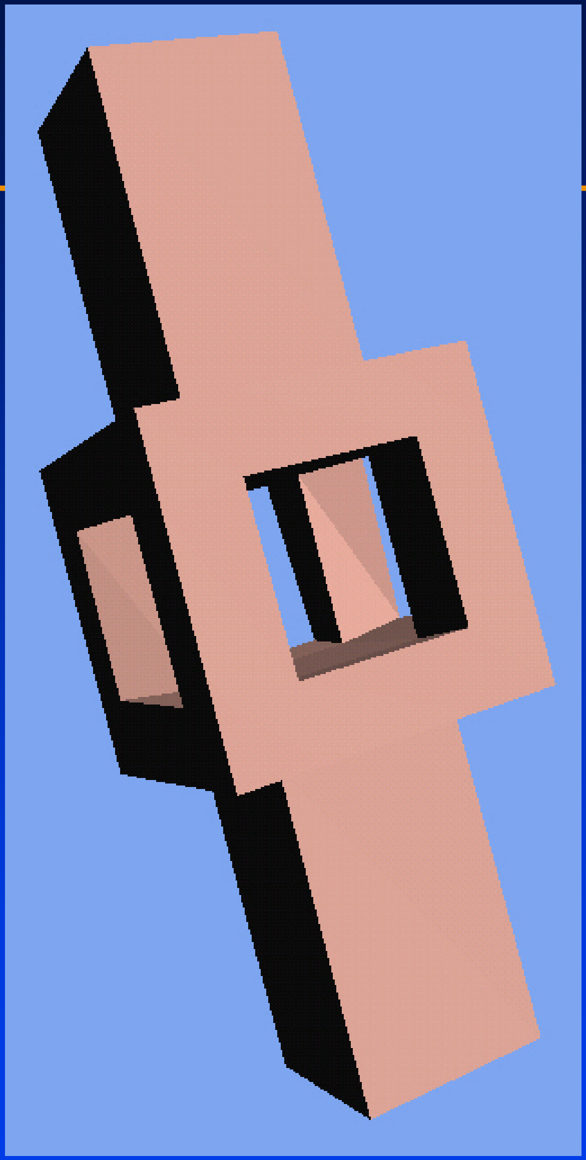


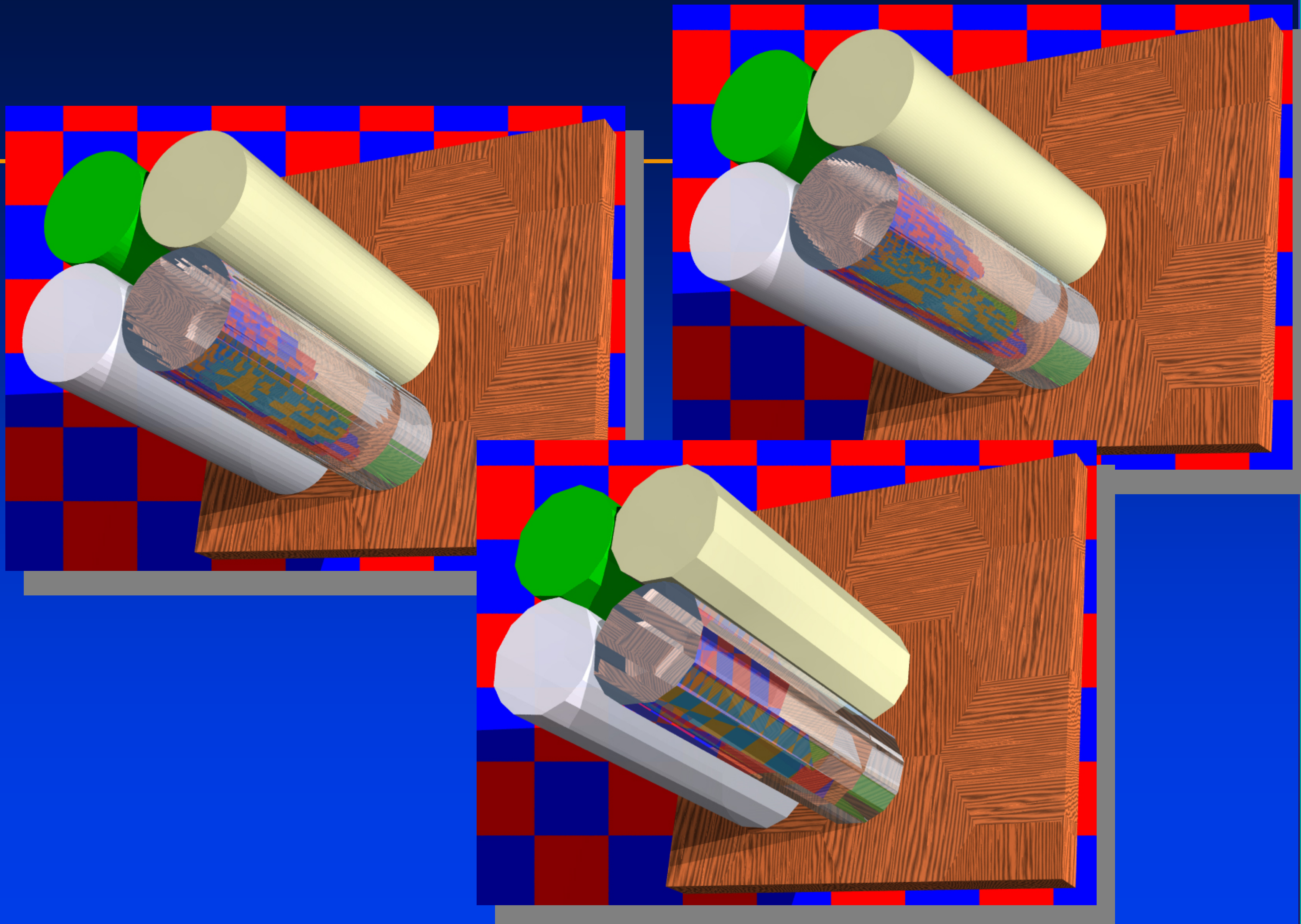
Shaded Model

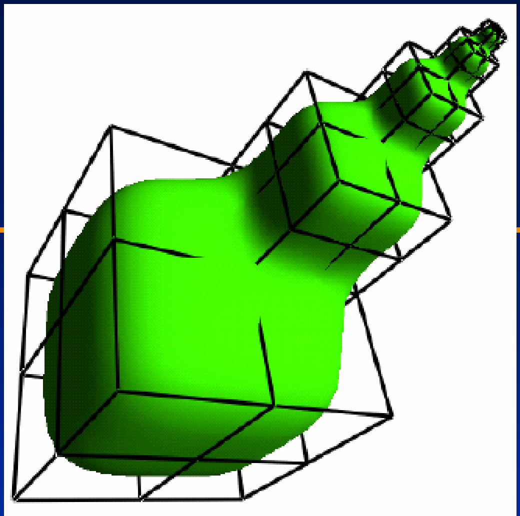


Mechanical Part



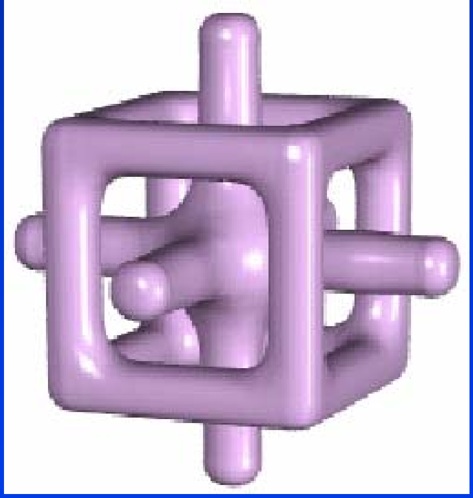






Subdivision model

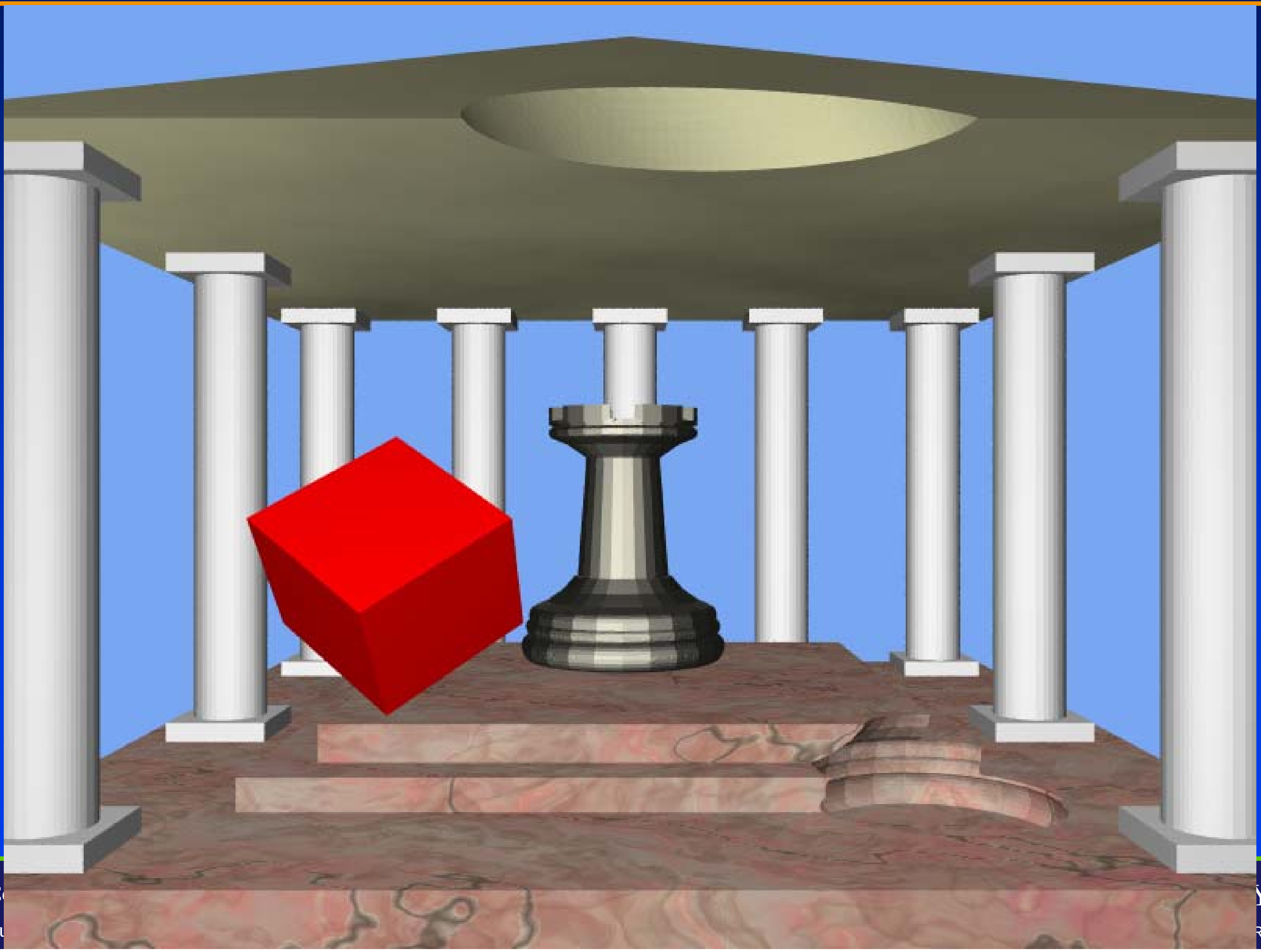
NURBS model

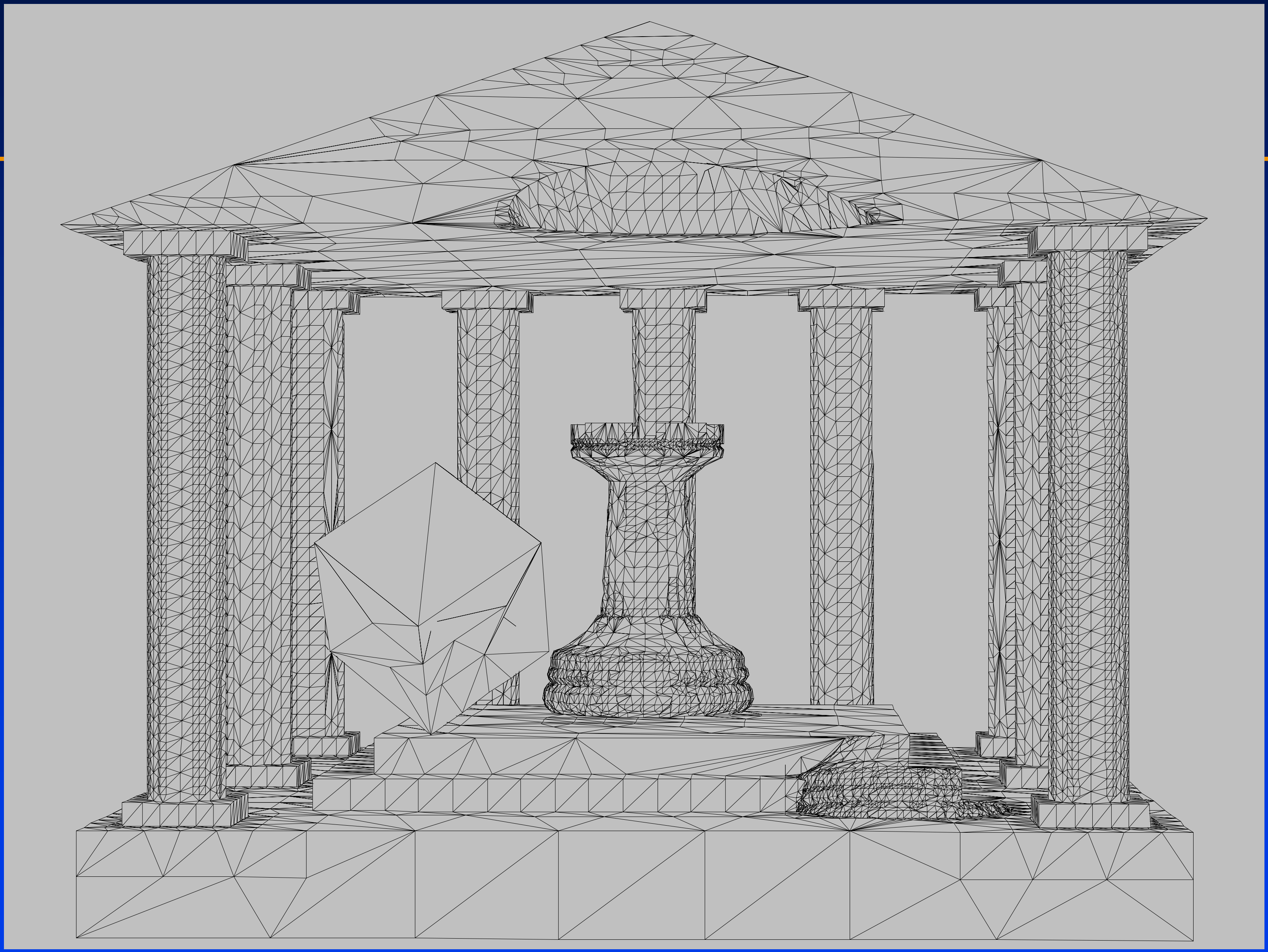


Implicit model

PDE models

Building Structure





Mathematical Tools

- Parametric curves and surfaces
- Spline-based objects (piecewise polynomials)
- Explicit, implicit, and parametric representations
- The integrated way to look at the shape:
  - Object can be considered as a set of faces, each face can be further decomposed into a set of edges, each edge can be decomposed into vertices
- Subdivision models
- Other procedure-based models
- Sweeping
- Surfaces of revolution
- Volumetric models

Line Equation

- Parametric representation

$$\mathbf{l}(\mathbf{p}_0, \mathbf{p}_1) = \mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u$$
$$u \in [0,1]$$

- Parametric representation is not unique

$$\mathbf{l}(\mathbf{p}_0, \mathbf{p}_1) = \mathbf{0.5}(\mathbf{p}_1 + \mathbf{p}_0) + \mathbf{0.5}(\mathbf{p}_1 - \mathbf{p}_0)v$$
$$v \in [-1,1]$$

- In general

$$\mathbf{p}(u),$$
$$u \in [a, b]$$

- Re-parameterization (variable transformation)

$$v = (u - a)/(b - a)$$
$$u = (b - a)v + a$$
$$\mathbf{q}(v) = \mathbf{p}((b - a)v + a)$$
$$v \in [0,1]$$

Basic Concepts

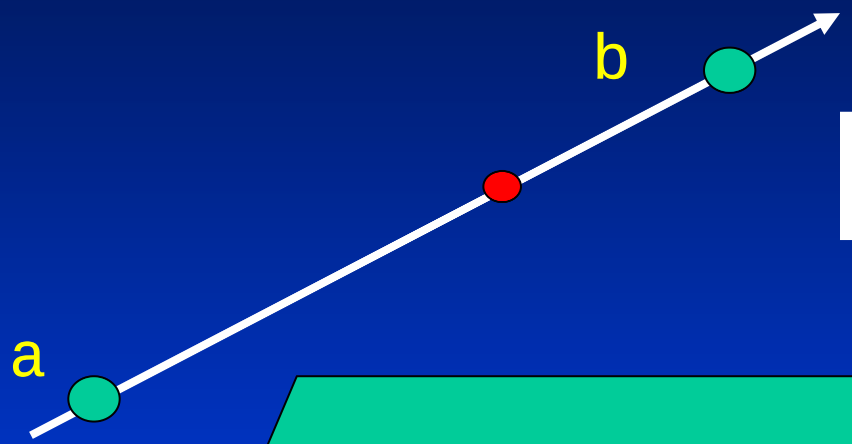
- Linear interpolation: $\mathbf{v} = \mathbf{v}_0(1-t) + \mathbf{v}_1(t)$
- Local coordinates: $\mathbf{v} \in [\mathbf{v}_0, \mathbf{v}_1], t \in [0,1]$
- Reparameterization: $f(u), u = g(v), f(g(v)) = h(v)$
- Affine transformation:

$$f(ax+by) = af(x) + bf(y)$$

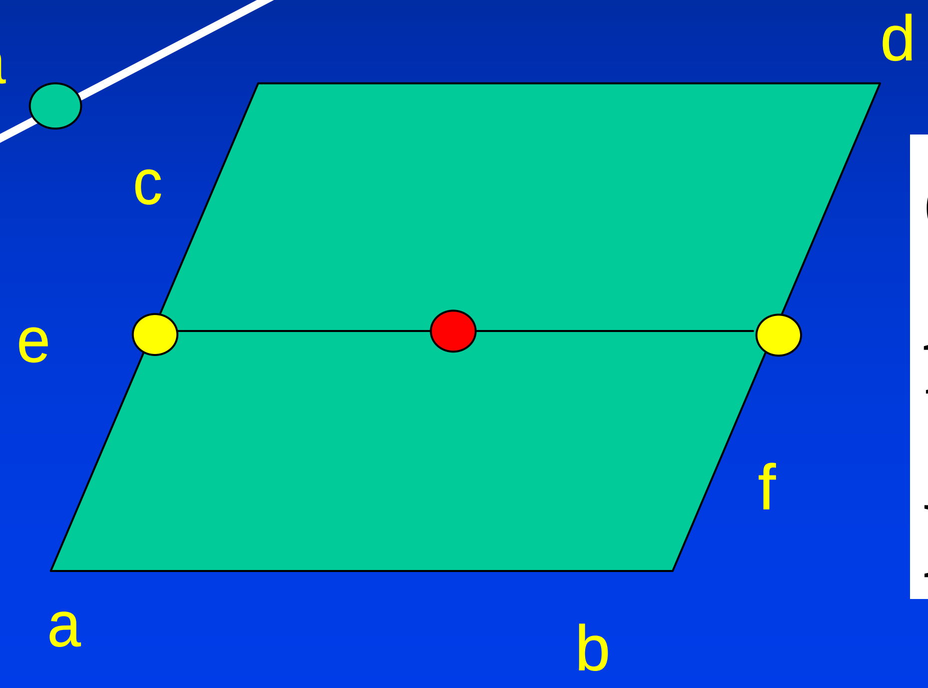
$$a + b = 1$$

- Polynomials
- Continuity

Linear and Bilinear Interpolation



$$\mathbf{p} = (1-u)\mathbf{a} + u\mathbf{b}$$



$$\mathbf{e} = (1-u)\mathbf{a} + u\mathbf{c}$$

$$\mathbf{f} = (1-u)\mathbf{b} + u\mathbf{d}$$

$$\mathbf{p} = (1-v)\mathbf{e} + v\mathbf{f}$$

Fundamental Features

- Geometry
  - Position, direction, length, area, normal, tangent, etc.
- Interaction
  - Size, continuity, collision, intersection
- Topology
- Differential properties
  - Curvature, arc-length
- Physical attributes
- Computer representation & data structure
- Others!

Mathematical Formulations

- Point:

$$\mathbf{p} = \begin{bmatrix} \mathbf{a}_x \\ \mathbf{a}_y \\ \mathbf{a}_z \end{bmatrix}$$

- Line:

$$\mathbf{l}(u) = [\mathbf{a} \quad \mathbf{a} \quad \mathbf{a}]^T u + [\mathbf{b} \quad \mathbf{b} \quad \mathbf{b}]^T$$

- Quadratic curve:

$$\mathbf{q}(u) = [\mathbf{a}_x \quad \mathbf{a}_y \quad \mathbf{a}_z]^T u^2 + [\mathbf{b}_x \quad \mathbf{b}_y \quad \mathbf{b}_z]^T u + [\mathbf{c}_x \quad \mathbf{c}_y \quad \mathbf{c}_z]^T$$

- Parametric domain and reparameterization:

$$u \in [u_s, u_e]; \; v \in [0,1]; \; v = (u - u_s)/(u_e - u_s)$$

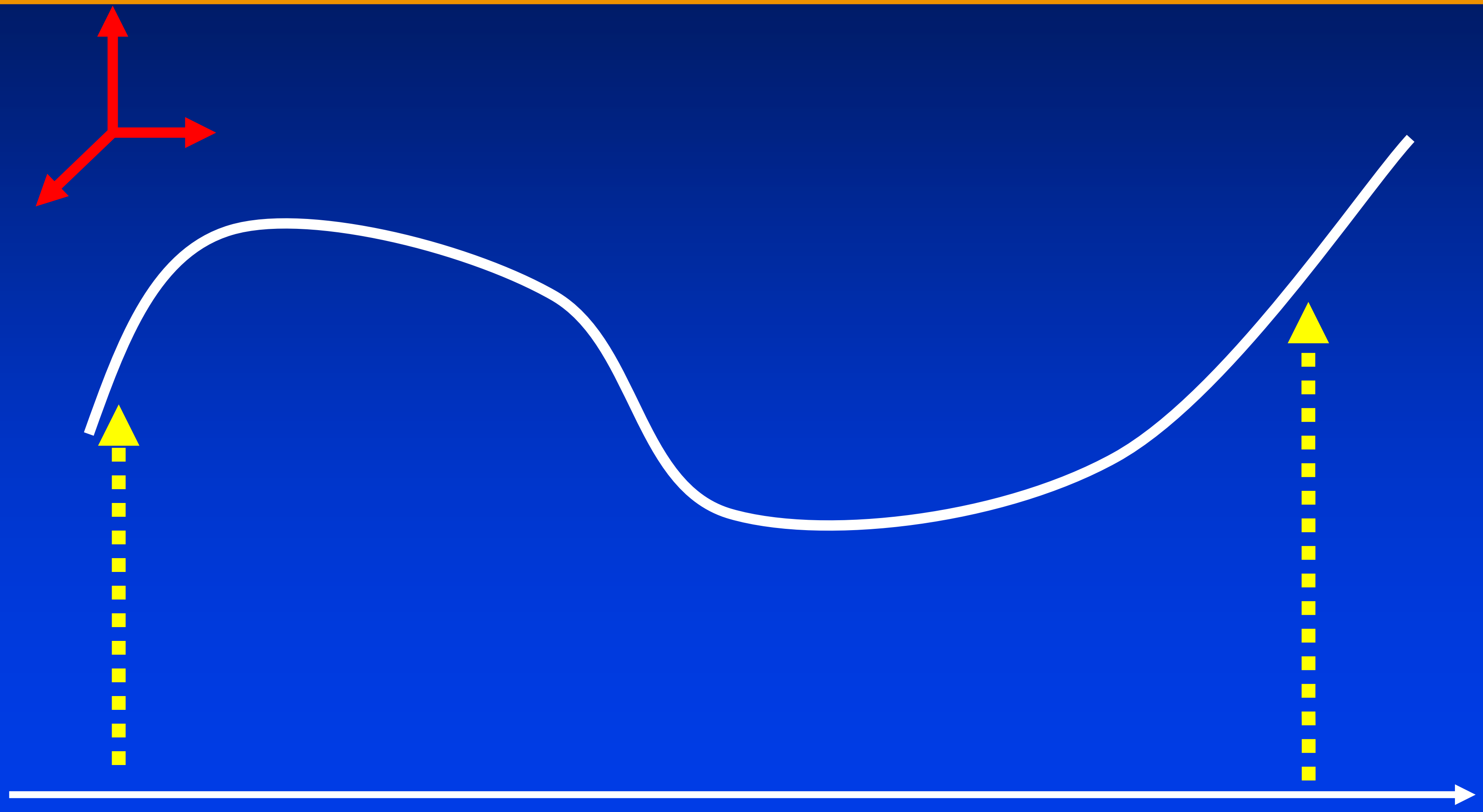
Parametric Polynomials

- High-order polynomials

$$\mathbf{c}(u) = \begin{bmatrix} \mathbf{a}_{0,x} \\ \mathbf{a}_{0,y} \\ \mathbf{a}_{0,z} \end{bmatrix} + \ldots + \begin{bmatrix} \mathbf{a}_{i,x} \\ \mathbf{a}_{i,y} \\ \mathbf{a}_{i,z} \end{bmatrix} u^i + \ldots + \begin{bmatrix} \mathbf{a}_{n,x} \\ \mathbf{a}_{n,y} \\ \mathbf{a}_{n,z} \end{bmatrix} u^n$$

- No intuitive insight for the curved shape
- Difficult for piecewise smooth curves

Parametric Polynomials



How to Define a Curve?

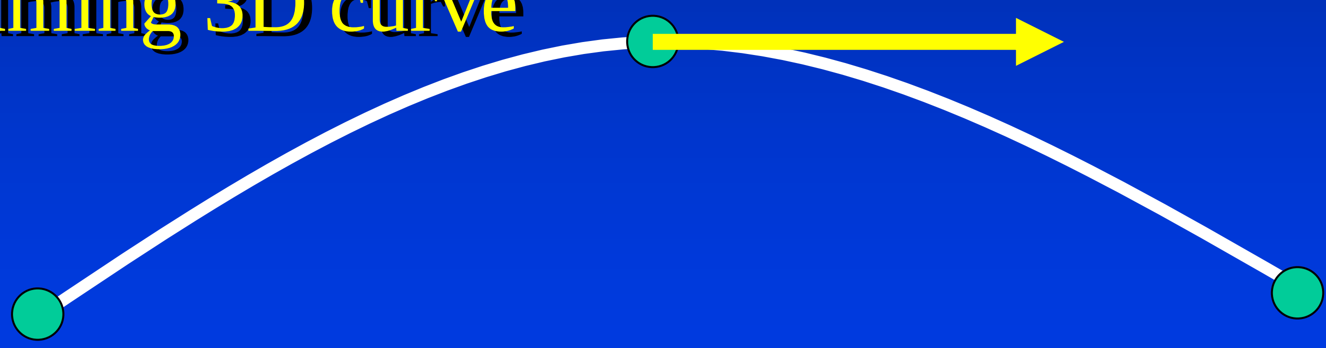
- Specify a set of points for interpolation and/or approximation with fixed or unfixed parameterization

$$\begin{bmatrix} x(u_i) \\ y(u_i) \\ z(u_i) \end{bmatrix} \qquad \begin{bmatrix} x'(u_i) \\ y'(u_i) \\ z'(u_i) \end{bmatrix}$$

- Specify the derivatives at some locations
- What is the geometric meaning to specify derivatives?
- A set of constraints
- Solve constraint equations

One Example

- Two end-vertices: c(0) and c(1)
- One mid-point: c(0.5)
- Tangent at the mid-point: c'(0.5)
- Assuming 3D curve



Cubic Polynomials

- Parametric representation (u is in [0,1])

$$\begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \begin{bmatrix} a_3 \\ b_3 \\ c_3 \end{bmatrix} u^3 + \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix} u^2 + \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} u + \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix}$$

- Each components are treated independently
- High-dimension curves can be easily defined
- Alternatively

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_3 & a_2 & a_1 & a_0 \end{bmatrix}^T = UA$$

$$y(u) = UB$$

$$z(u) = UC$$

Cubic Polynomial Example

- Constraints: two end-points, one mid-point, and tangent at the mid-point

$$x(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} A$$
$$x(0.5) = \begin{bmatrix} 0.5^3 & 0.5^2 & 0.5^1 & 1 \end{bmatrix} A$$
$$x'(0.5) = \begin{bmatrix} 3(0.5)^2 & 2(0.5) & 1 & 0 \end{bmatrix} A$$
$$x(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} A$$

- In matrix form

$$\begin{bmatrix} x(0) \\ x(0.5) \\ x'(0.5) \\ x(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0.125 & 0.25 & 0.5 & 1 \\ 0.75 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} A$$

Solve this Linear Equation

- Invert the matrix

$$A = \begin{bmatrix} -4 & 0 & -4 & 4 \\ 8 & -4 & 6 & -4 \\ -5 & 4 & -2 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(0.5) \\ x'(0.5) \\ x(1) \end{bmatrix}$$

- Rewrite the curve expression

$$x(u) = UM\begin{bmatrix} x(0) & x(0.5) & x'(0.5) & x(1) \end{bmatrix}^T$$

$$y(u) = UM\begin{bmatrix} y(0) & y(0.5) & y'(0.5) & y(1) \end{bmatrix}^T$$

$$z(u) = UM\begin{bmatrix} z(0) & z(0.5) & z'(0.5) & z(1) \end{bmatrix}^T$$

Basis Functions

- Special polynomials

$$f_1(u) = -4u^3 + 8u^2 - 5u + 1$$
$$f_2(u) = -4u^2 + 4u$$
$$f_3(u) = -4u^3 + 6u^2 - 2u$$
$$f_4(u) = 4u^3 - 4u^2 + 1$$

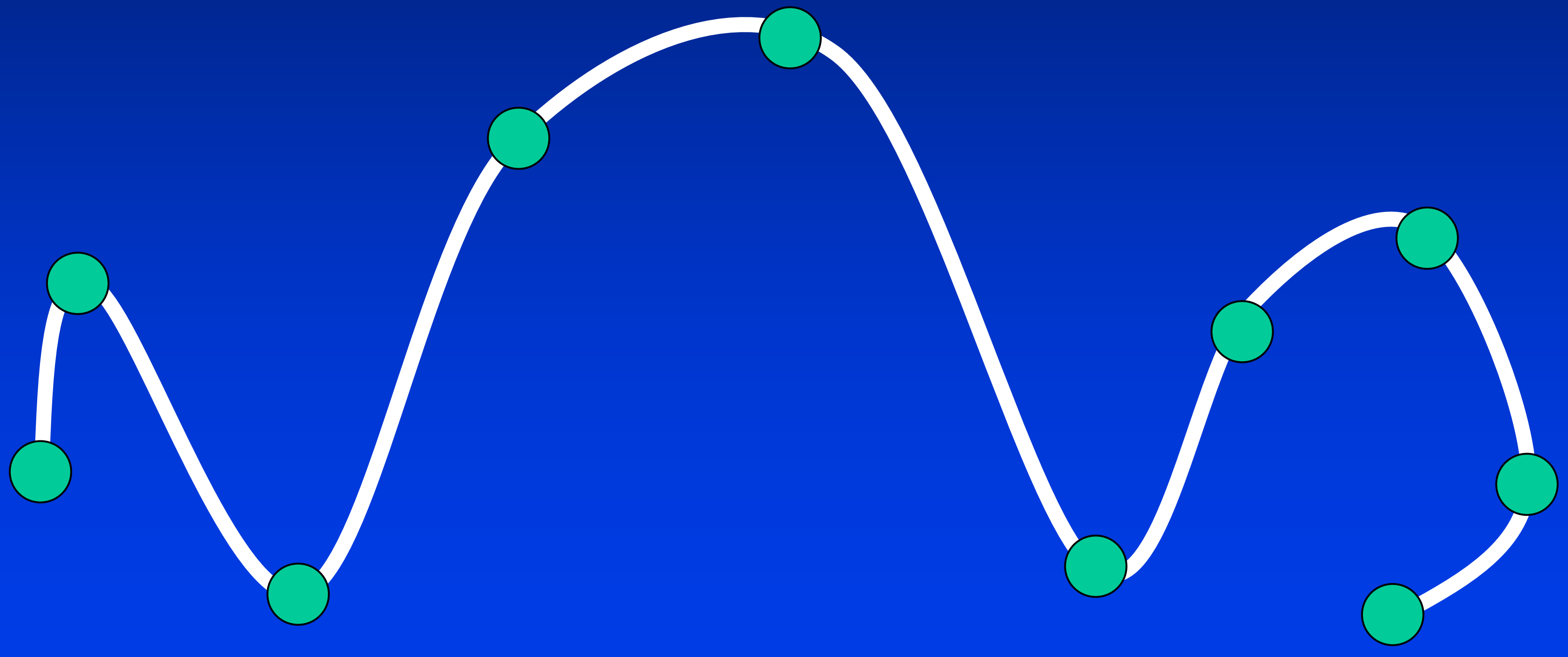
- What is the image of these basis functions?
- Polynomial curve can be defined by

$$\mathbf{c}(u) = \mathbf{c}(0) f_1(u) + \mathbf{c}(0.5) f_2(u) + \mathbf{c}'(0.5) f_3(u) + \mathbf{c}(1) f_4(u)$$

- Observations
  - More intuitive, easy to control, polynomials

Lagrange Curve

- Point interpolation



Lagrange Curves

- Curve

$$\mathbf{c}(u) = \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \mathbf{a} \end{bmatrix} L_0^n(u) + \ldots + \begin{bmatrix} \mathbf{a} \\ \mathbf{a} \\ \mathbf{a} \end{bmatrix} L_n^n(u)$$

- Lagrange polynomials of degree n: $L_i^n(u)$
- Knot sequence: $u_0, \ldots, u_n$
- Kronecker delta: $L_i^n(u_j) = \delta_{ij}$
- The curve interpolate all the data point, but unwanted oscillation

Lagrange Basis Functions

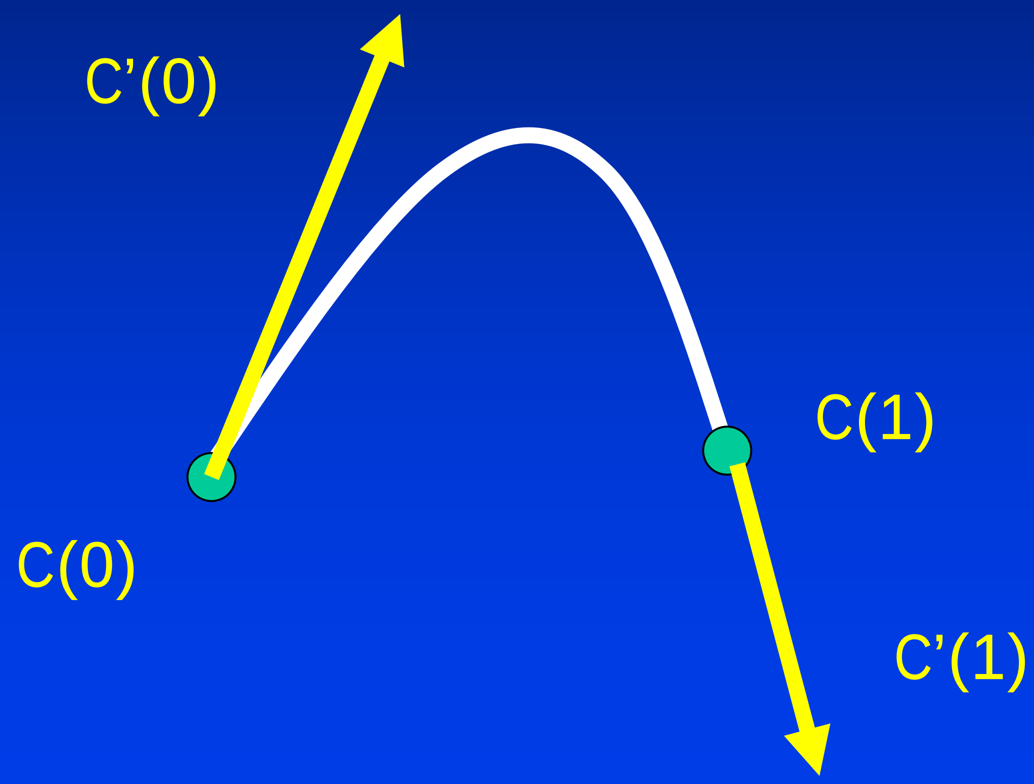
$$L_i^n(u_j) = \begin{cases} 1 & i = j(i, j = 0,1,\ldots, n) \\ 0 & Otherwise \end{cases}$$

$$L_0^n(u) = \frac{(u-u_1)(u-u_2)\ldots(u-u_n)}{(u_0-u_1)(u_0-u_2)\ldots(u_0-u_n)}$$

$$L_i^n(u) = \frac{(u-u_0)\ldots(u-u_{i-1})(u-u_{i+1})\ldots(u-u_n)}{(u_i-u_0)\ldots(u_i-u_{i-1})(u_i-u_{i+1})\ldots(u_i-u_n)}$$

$$L_n^n(u) = \frac{(u-u_0)\ldots(u-u_{n-2})(u-u_{n-1})}{(u_n-u_0)\ldots(u_n-u_{n-2})(u_n-u_{n-1})}$$

Cubic Hermite Splines



Cubic Hermite Curve

- Hermite curve

$$\mathbf{c}(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$

- Two end-points and two tangents at end-points

$$\begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} A$$

- Matrix inversion

$$x(u) = U \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ x'(0) \\ x'(1) \end{bmatrix}$$

$$y(u) = UM \begin{bmatrix} y(0) & y(1) & y'(0) & y'(1) \end{bmatrix}^T$$

$$z(u) = UM \begin{bmatrix} z(0) & z(1) & z'(0) & z'(1) \end{bmatrix}^T$$

Hermite Curve

- Basis functions

$$f_1(u) = 2u^3 - 3u^2 + 1$$
$$f_2(u) = -2u^3 + 3u^2$$
$$f_3(u) = u^3 - 2u^2 + u$$
$$f_4(u) = u^3 - u^2$$

- Display the image of these basis functions and the Hermite curve itself

$$\mathbf{c}(u) = \mathbf{c}(0) f_1(u) + \mathbf{c}(1) f_2(u) + \mathbf{c}'(0) f_3(u) + \mathbf{c}'(1) f_4(u)$$

Cubic Hermite Splines

- Two vertices and two tangent vectors:

$$\mathbf{c}(0) = \mathbf{v}_0, \mathbf{c}(1) = \mathbf{v}_1;$$
$$\mathbf{c}^{(1)}(0) = \mathbf{d}_0, \mathbf{c}^{(1)}(1) = \mathbf{d}_1;$$

- Hermite curve

$$\mathbf{c}(u) = \mathbf{v}_0 H_0^3(u) + \mathbf{v}_1 H_1^3(u) + \mathbf{d}_0 H_2^3(u) + \mathbf{d}_1 H_3^3(u);$$
$$H_0^3(u) = f_1(u), H_1^3(u) = f_2(u), H_2^3(u) = f_3(u), H_3^3(u) = f_4(u)$$

Hermite Splines

- Higher-order polynomials

$$\mathbf{c}(u) = \mathbf{v}_0^0 H_0^n(u) + \mathbf{v}_0^1 H_1^n(u) + \ldots + \mathbf{v}_0^{(n-1)/2} H_{(n-1)/2}^n(u)$$

$$+ \mathbf{v}_1^{(n-1)/2} H_{(n+1)/2}^n(u) + \ldots + \mathbf{v}_1^1 H_{(n-1)}^n(u) + \mathbf{v}_1^0 H_n^n(u);$$

$$\mathbf{v}_0^i = \mathbf{c}^{(i)}(0), \mathbf{v}_1^i = \mathbf{c}^{(i)}(1), i = 0, \ldots (n-1)/2;$$

- Note that, n is odd!
- Geometric intuition
- Higher-order derivatives are required

Why Cubic Polynomials

- Lowest degree for specifying curve in space
- Lowest degree for specifying points to interpolate and tangents to interpolate
- Commonly used in computer graphics
- Lower degree has too little flexibility
- Higher degree is unnecessarily complex, exhibit undesired wiggles

Variations of Hermite Curve

- Variations of Hermite curves

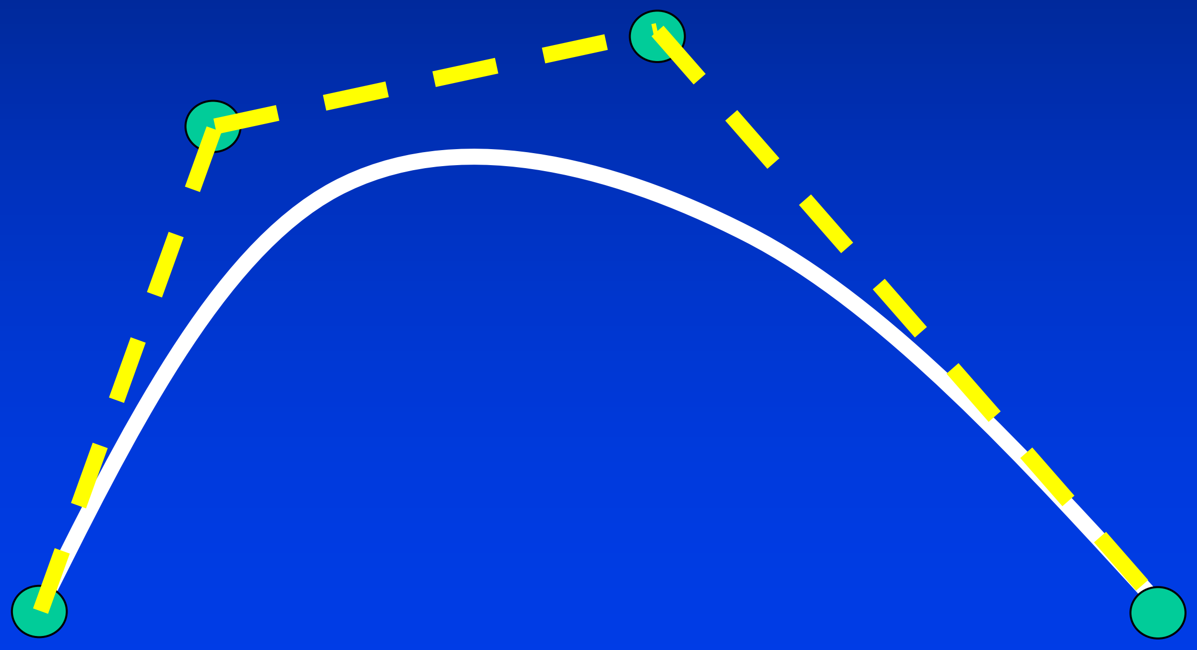
$$\mathbf{p}_0 = \mathbf{c}(0)$$
$$\mathbf{p}_3 = \mathbf{c}(1)$$
$$\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0), \mathbf{p}_1 = \mathbf{p}_0 + \mathbf{c}'(0)/3$$
$$\mathbf{c}'(1) = 3(\mathbf{p}_3 - \mathbf{p}_2), \mathbf{p}_2 = \mathbf{p}_3 - \mathbf{c}'(1)/3$$

- In matrix form (x-component only)

$$\begin{bmatrix} \mathbf{c}(0)_x \\ \mathbf{c}(1)_x \\ \mathbf{c}'(0)_x \\ \mathbf{c}'(1)_x \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{0,x} \\ \mathbf{p}_{0,x} \\ \mathbf{p}_{0,x} \\ \mathbf{p}_{0,x} \end{bmatrix}$$

Cubic Bezier Curves

- Four control points
- Curve geometry



Curve Mathematics (Cubic)

- Bezier curve

$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(u)$$

- Control points and basis functions

$$B_0^3(u) = (1-u)^3$$

$$B_1^3(u) = 3u(1-u)^2$$

$$B_2^3(u) = 3u^2(1-u)$$

$$B_3^3(u) = u^3$$

- Image and properties of basis functions

Recursive Evaluation

- Recursive linear interpolation

$$(1-u) \quad (u)$$

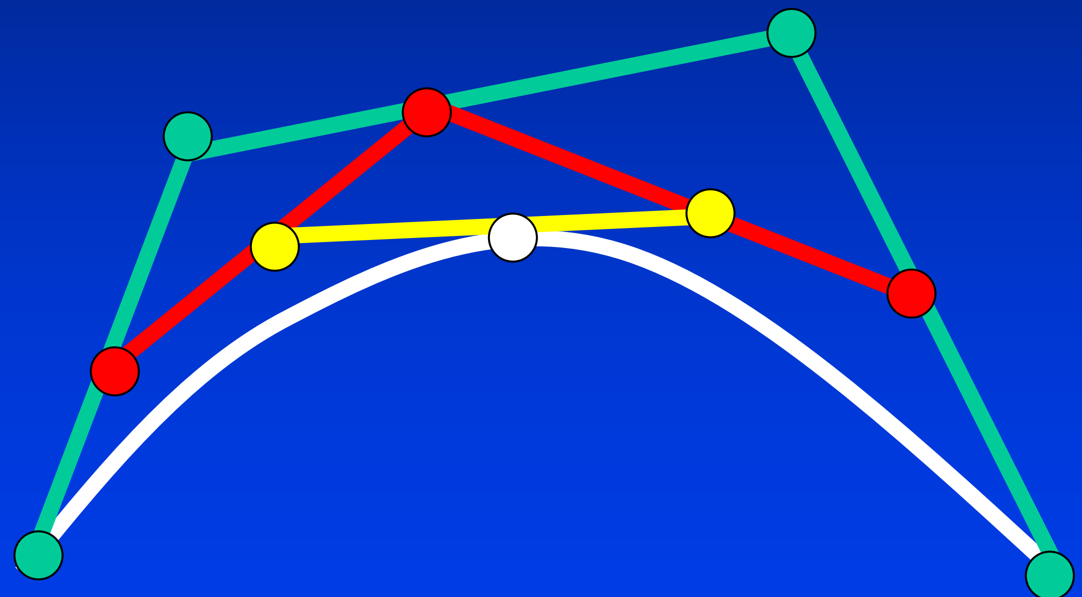
$$\mathbf{p}_0^0 \quad \mathbf{p}_1^0 \quad \mathbf{p}_2^0 \quad \mathbf{p}_3^0$$

$$\mathbf{p}_0^1 \quad \mathbf{p}_1^1 \quad \mathbf{p}_2^1$$

$$\mathbf{p}_0^2 \quad \mathbf{p}_1^2$$

$$\mathbf{p}_0^3 = \mathbf{c}(u)$$

Recursive Subdivision Algorithm



Basic Properties (Cubic)

- The curve passes through the first and the last points (end-point interpolation)
- Linear combination of control points and basis functions
- Basis functions are all polynomials
- Basis functions sum to one (partition of unity)
- All basis functions are non-negative
- Convex hull (both necessary and sufficient)
- Predictability

Derivatives

- Tangent vectors can easily be evaluated at the end-points $\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0); \mathbf{c}'(1) = (\mathbf{p}_3 - \mathbf{p}_2)$
- Second derivatives at end-points can also be easily computed:

$$\mathbf{c}^{(2)}(0) = 2 \times 3((\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)) = 6(\mathbf{p}_2 - 2\mathbf{p}_1 + \mathbf{p}_0)$$
$$\mathbf{c}^{(2)}(1) = 2 \times 3((\mathbf{p}_3 - \mathbf{p}_2) - (\mathbf{p}_2 - \mathbf{p}_1)) = 6(\mathbf{p}_3 - 2\mathbf{p}_2 + \mathbf{p}_1)$$

Derivative Curve

- The derivative of a cubic Bezier curve is a quadratic Bezier curve

$$\mathbf{c}'(u) = -3(1-u)^2\mathbf{p}_0 + 3\left((1-u)^2 - 2u(1-u)\right)\mathbf{p}_1 + 3\left(2u(1-u) - u^2\right)\mathbf{p}_2 + 3u^2\mathbf{p}_3 =$$
$$3(\mathbf{p}_1 - \mathbf{p}_0)(1-u)^2 + 3(\mathbf{p}_2 - \mathbf{p}_1)2u(1-u) + 3(\mathbf{p}_3 - \mathbf{p}_2)u^2$$

More Properties (Cubic)

- Two curve spans are obtained, and both of them are standard Bezier curves (through reparameterization)

$$\mathbf{c}(v),\ v\in[0,u]$$
$$\mathbf{c}(v),\ v\in[u,1]$$
$$\mathbf{c}_l(u),\ u\in[0,1]$$
$$\mathbf{c}_r(u),\ u\in[0,1]$$

- The control points for the left and the right are

$$\mathbf{p}_0^0,\ \mathbf{p}_0^1,\ \mathbf{p}_0^2,\ \mathbf{p}_0^3$$
$$\mathbf{p}_0^3,\ \mathbf{p}_1^2,\ \mathbf{p}_2^1,\ \mathbf{p}_3^0$$

High-Degree Curves

- Generalizing to high-degree curves

$$\begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix} = \sum_{i=0}^{n} \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix} u^i$$

- Advantages:
  - Easy to compute, Infinitely differentiable
- Disadvantages:
  - Computationally complex, undulation, undesired wiggles
- How about high-order Hermite? Not natural!!!

Bezier Splines

- Bezier curves of degree n

$$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_i^n(u)$$

- Control points and basis functions (Bernstein polynomials of degree n):

$$B_i^n(u) = \binom{n}{i}(1-u)^{n-i}u^i$$

$$\binom{n}{i} = \frac{n!}{(n-i)!\,i!}$$

Recursive Computation

$$\mathbf{p}_i^0 = \mathbf{p}_i, i = 0,1,2,\ldots n$$

$$\mathbf{p}_i^j = (1-u)\mathbf{p}_i^{j-1} + u\mathbf{p}_{i+1}^{j-1}$$

$$\mathbf{c}(u) = \mathbf{p}_0^n(u)$$

Recursive Computation

- N+1 levels

$$
\begin{array}{ccccccc}
 & (1-u) & & & (u) & & \\
\mathbf{p}_0^0 & & \cdots & & \cdots & & \mathbf{p}_n^0 \\
 & \mathbf{p}_0^1 & & \cdots & & \mathbf{p}_{n-1}^1 & \\
 & \cdots & & \cdots & & \cdots & \\
 & & \mathbf{p}_0^{n-1} & & \mathbf{p}_1^{n-1} & & \\
 & & & \mathbf{p}_0^n = \mathbf{c}(u) & & &
\end{array}
$$

Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $\mathbf{c}(0) = \mathbf{p}_0, \mathbf{c}(1) = \mathbf{p}_n$
- Binomial expansion theorem

$$((1-u)+u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by control points

More Properties

- Recursive subdivision and evaluation
- Symmetry: c(u) and c(1-u) are defined by the same set of point points, but different ordering

$$\mathbf{p}_0, \ldots, \mathbf{p}_n;$$
$$\mathbf{p}_n, \ldots, \mathbf{p}_0$$

Tangents and Derivatives

- End-point tangents: $$\mathbf{c}'(0) = n(\mathbf{p}_1 - \mathbf{p}_0)$$ $$\mathbf{c}'(1) = n(\mathbf{p}_n - \mathbf{p}_{n-1})$$
- I-th derivatives at two end-points depend on

$$\mathbf{p}_0, \ldots, \mathbf{p}_i;$$
$$\mathbf{p}_n, \ldots, \mathbf{p}_{n-i}$$

- Derivatives at non-end-points involve all control points

Other Advanced Topics

- Efficient evaluation algorithm
- Differentiation and integration
- Degree elevation
  - Use a polynomial of degree (n+1) to express that of degree (n)
- Composite curves
- Geometric continuity
- Display of curve

Bezier Curve Rendering

- Use its control polygon to approximate the curve
- Recursive subdivision till the tolerance is satisfied
- Algorithm go here
  - If the current control polygon is flat (with tolerance), then output the line segments, else subdivide the curve at u=0.5
  - Compute control points for the left half and the right half, respectively
  - Recursively call the same procedure for the left one and the right one

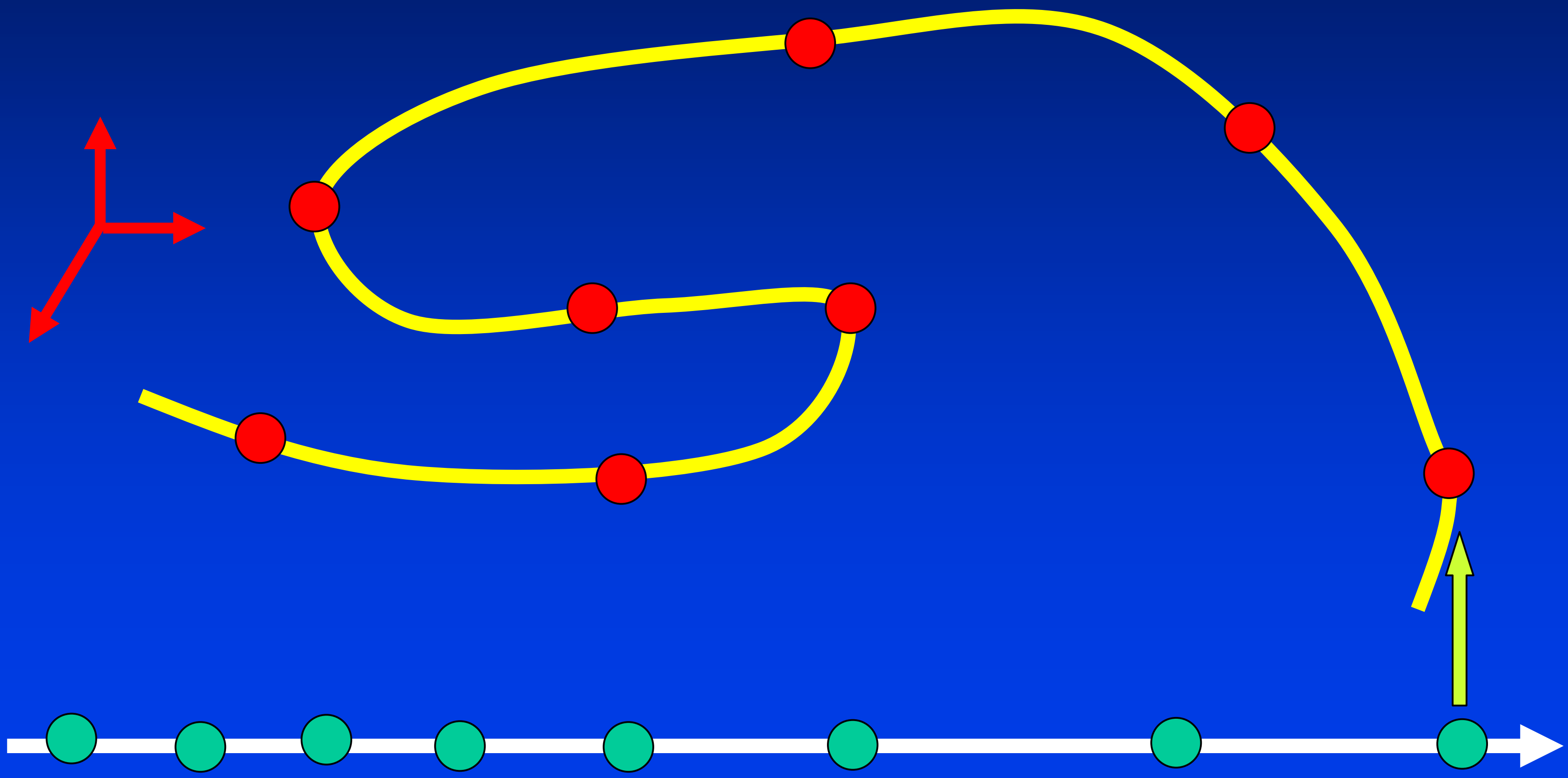
High-Degree Polynomials

- More degrees of freedom
- Easy to compute
- Infinitely differentiable
- Drawbacks:
  – High-order
  – Global control
  – Expensive to compute, complex
  – undulation

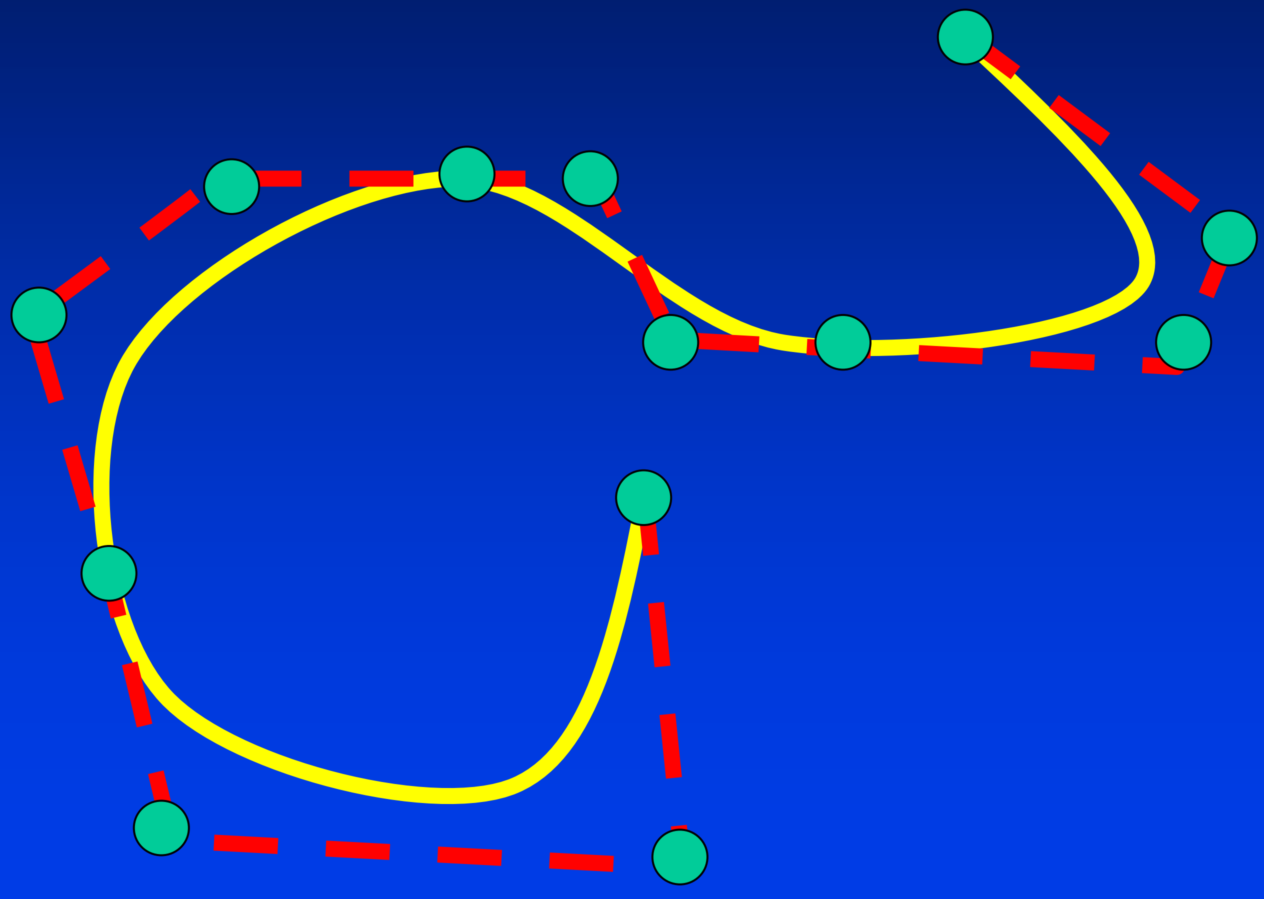
Piecewise Polynomials

- Piecewise --- different polynomials for different parts of the curve
- Advantages --- flexible, low-degree
- Disadvantages --- how to ensure smoothness at the joints (continuity)

Piecewise Curves



Piecewise Bezier Curves

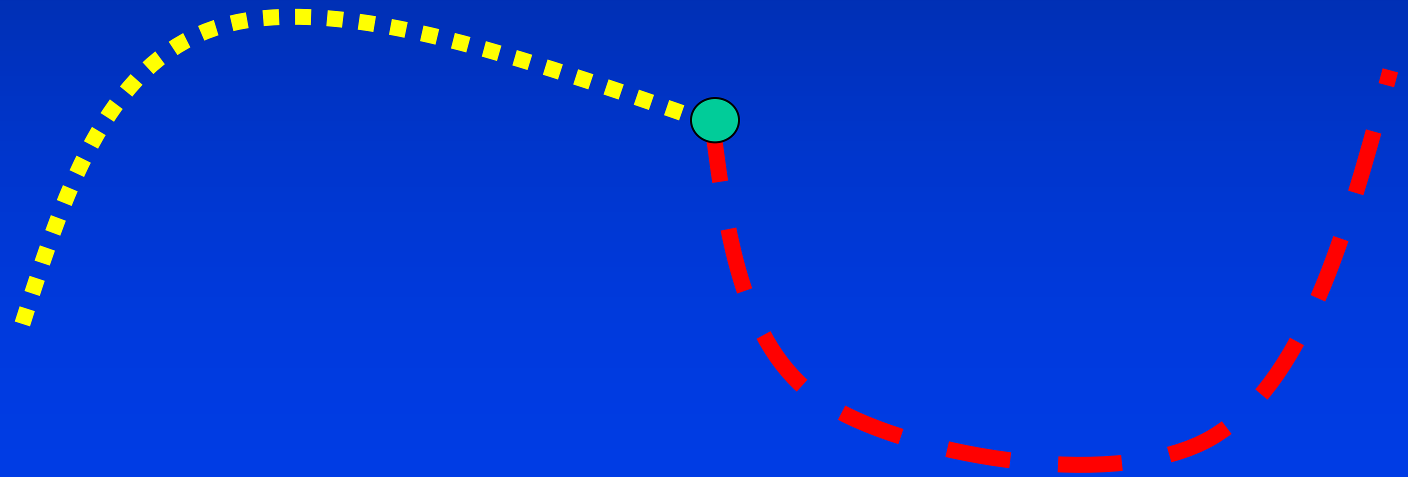


Continuity

- One of the fundamental concepts
- Commonly used cases: C^0, C^1, C^2
- Consider two curves: a(u) and b(u) (u is in [0,1])

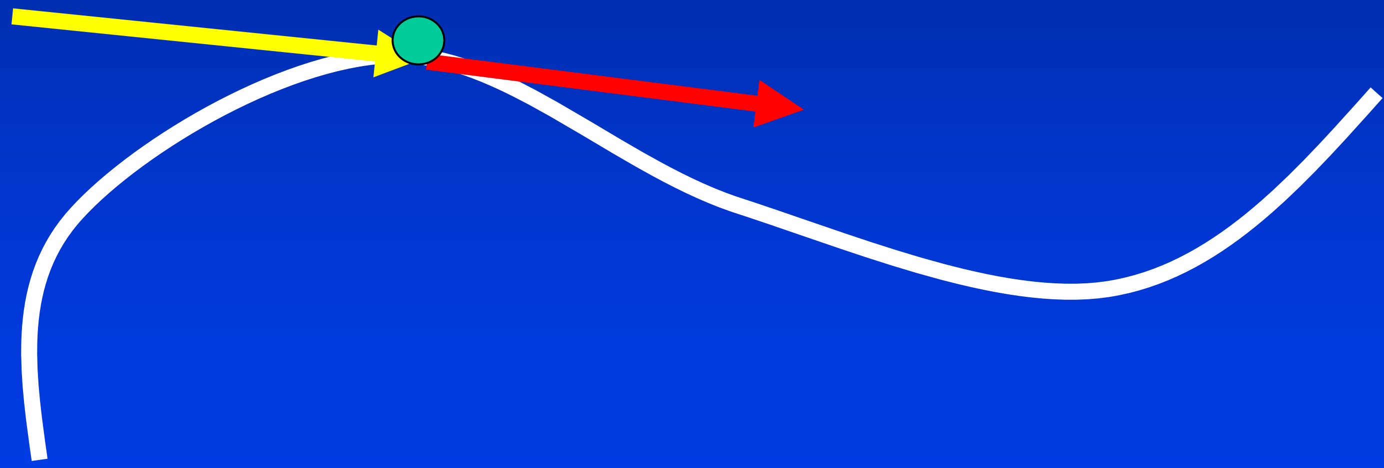
Positional Continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$



Derivative Continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$
$$\mathbf{a}'(1) = \mathbf{b}'(0)$$



General Continuity

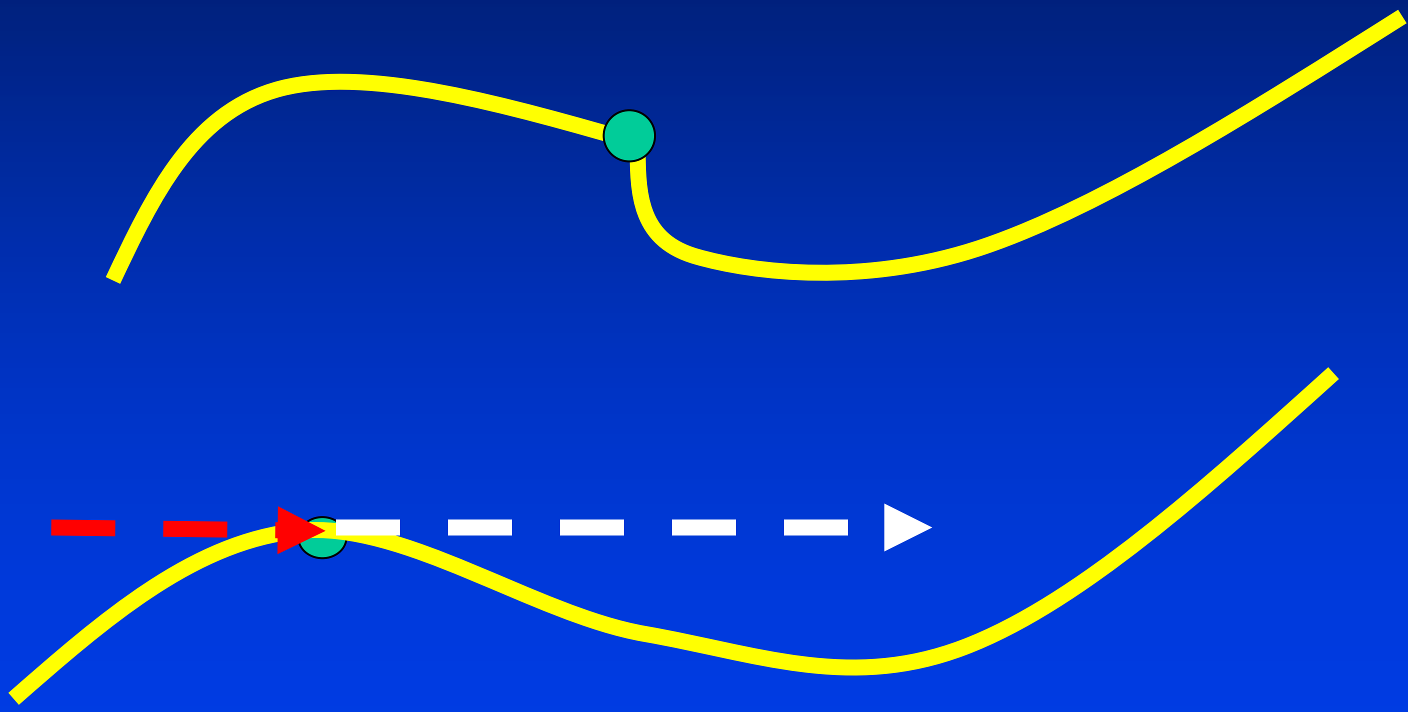
- Cn continuity: derivatives (up to n-th) are the same at the joining point

$$\mathbf{a}^{(i)}(1) = \mathbf{b}^{(i)}(0)$$
$$i = 0, 1, 2, \ldots, n$$

- The prior definition is for parametric continuity
- Parametric continuity depends of parameterization! But, parameterization is not unique!
- Different parametric representations may express the same geometry
- Re-parameterization can be easily implemented
- Another type of continuity: geometric continuity, or Gn

Geometric Continuity

- G0 and G1



Geometric Continuity

- Depend on the curve geometry
- DO NOT depend on the underlying parameterization
- G0: the same joint
- G1: two curve tangents at the joint align, but may (or may not) have the same magnitude
- G1: it is C1 after the reparameterization
- Which condition is stronger???
- Examples

Piecewise Hermite Curves

- How to build an interactive system to satisfy various constraints
- C0 continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$

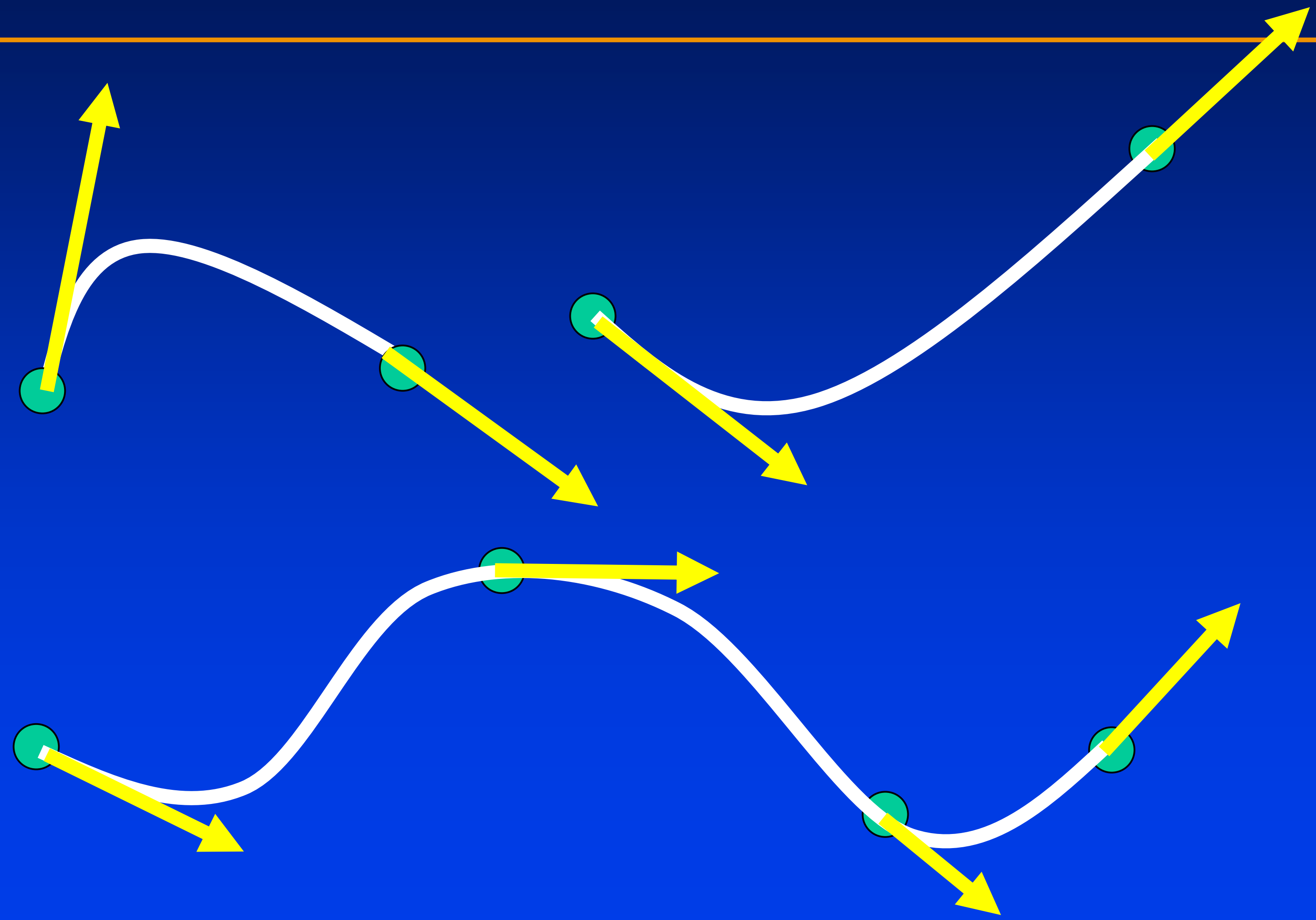
- C1 continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$
$$\mathbf{a}'(1) = \mathbf{b}'(0)$$

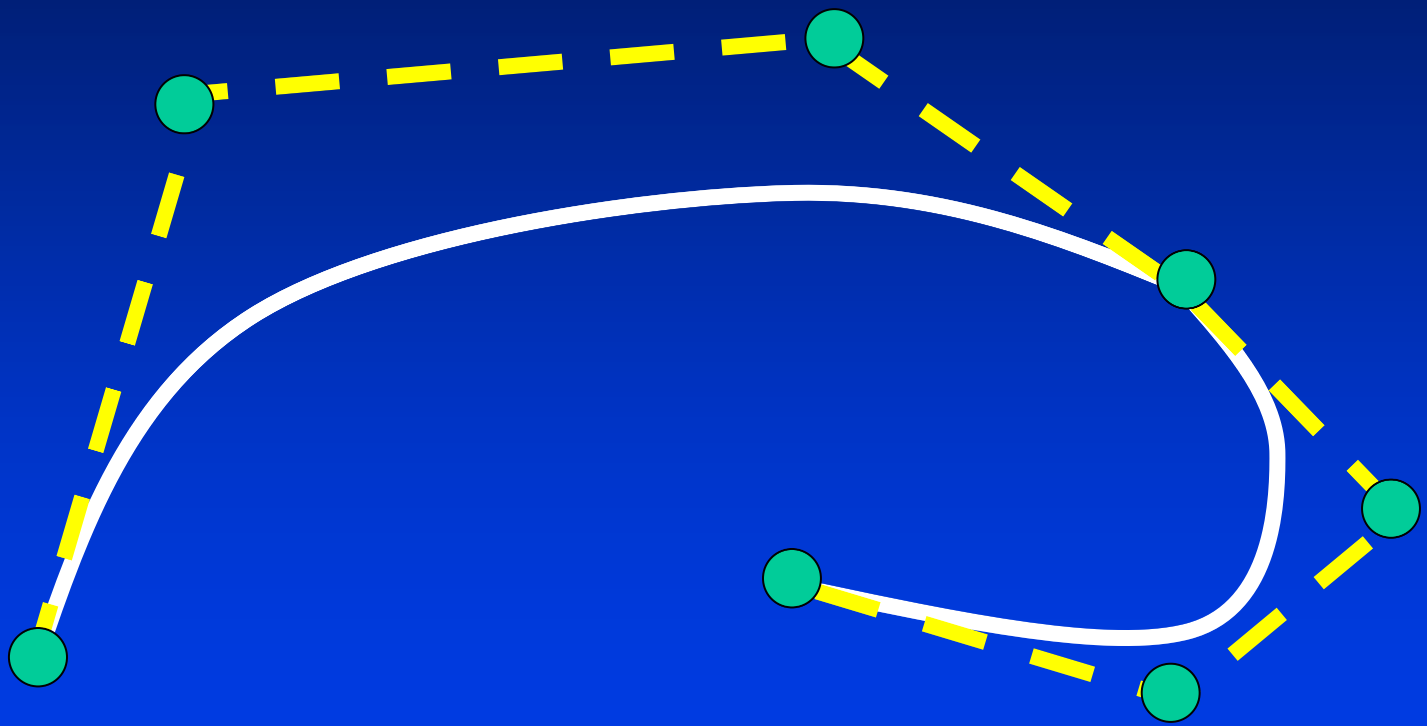
- G1 continuity

$$\mathbf{a}(1) = \mathbf{b}(0)$$
$$\mathbf{a}'(1) = \alpha\mathbf{b}'(0)$$

Piecewise Hermite Curves



Piecewise Bezier Curves



Piecewise Bezier Curves

- C0 continuity
- C1 continuity
- G1 continuity
- C2 continuity
- Geometric interpretation
- G2 continuity

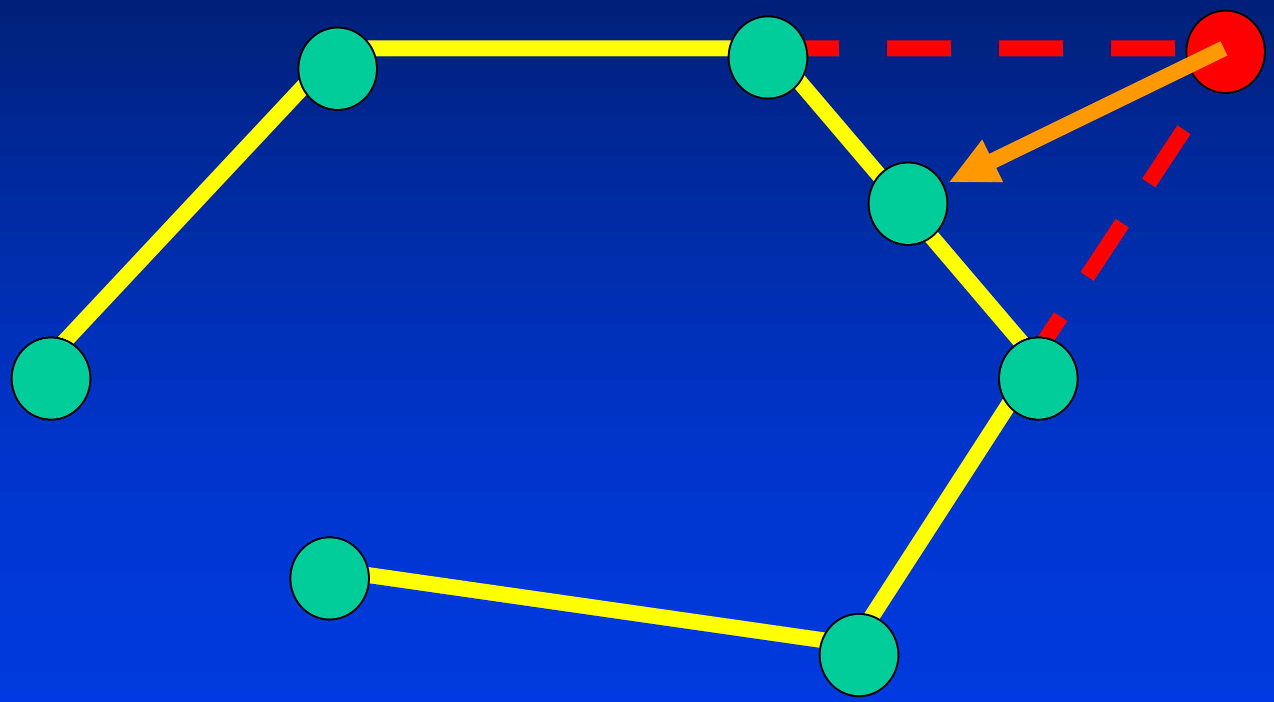
$$\mathbf{p}_3 = \mathbf{q}_0$$

$$\mathbf{p}_3 = \mathbf{q}_0$$
$$(\mathbf{p}_3 - \mathbf{p}_2) = (\mathbf{q}_1 - \mathbf{q}_0)$$

$$\mathbf{p}_3 = \mathbf{q}_0$$
$$(\mathbf{p}_3 - \mathbf{p}_2) = \alpha(\mathbf{q}_1 - \mathbf{q}_0)$$

$$\mathbf{p}_3 = \mathbf{q}_0$$
$$(\mathbf{p}_3 - \mathbf{p}_2) = (\mathbf{q}_1 - \mathbf{q}_0)$$
$$\mathbf{p}_3 - 2\mathbf{p}_2 + \mathbf{p}_1 = \mathbf{q}_2 - 2\mathbf{q}_1 + \mathbf{q}_0$$

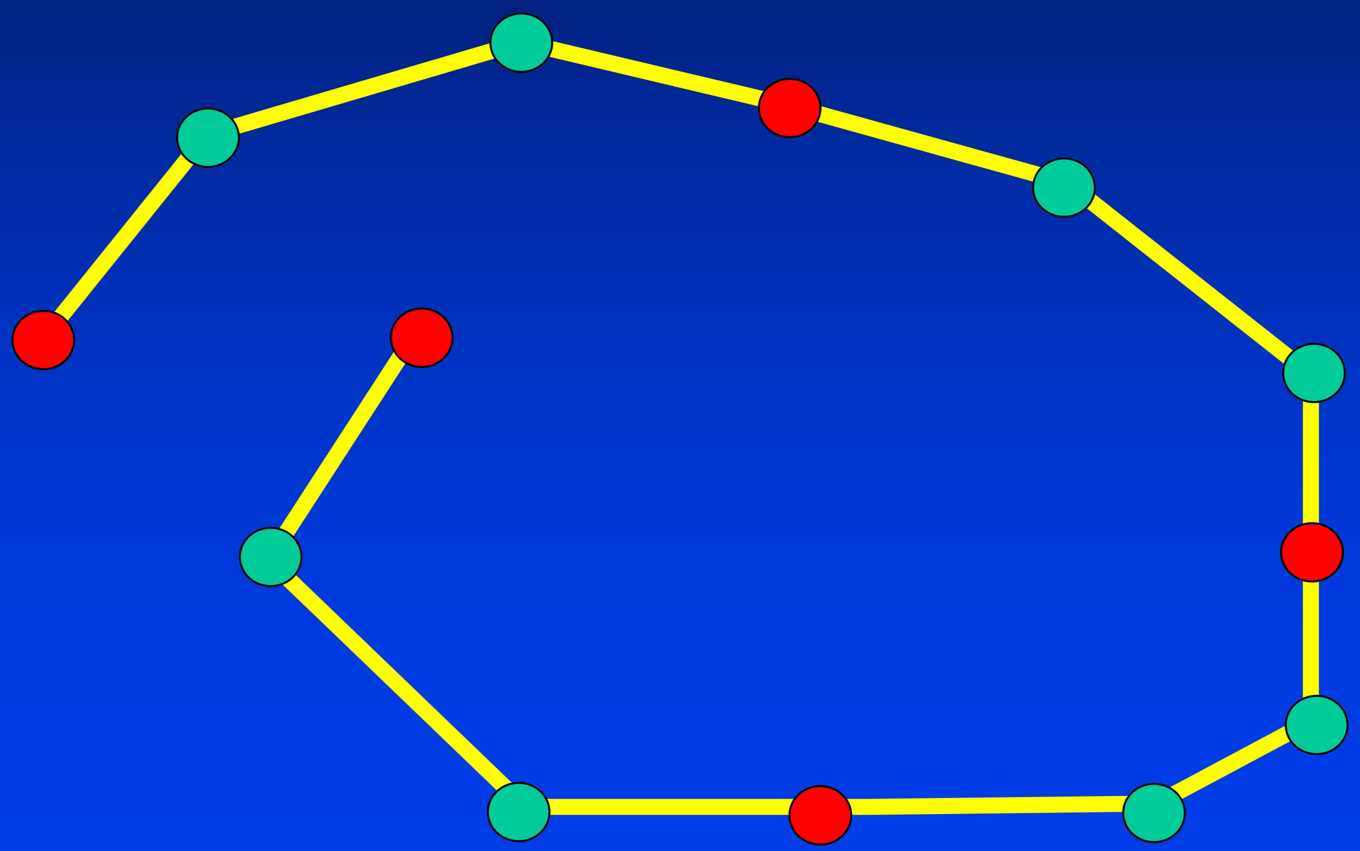
Piecewise C2 Bezier Curves



Continuity Summary

- C0: straightforward, but not enough
- C3: too constrained
- Piecewise curves with Hermite and Bezier representations satisfying various continuity conditions
- Interactive system for C2 interpolating splines using piecewise Bezier curves
- Advantages and disadvantages

C2 Interpolating Splines



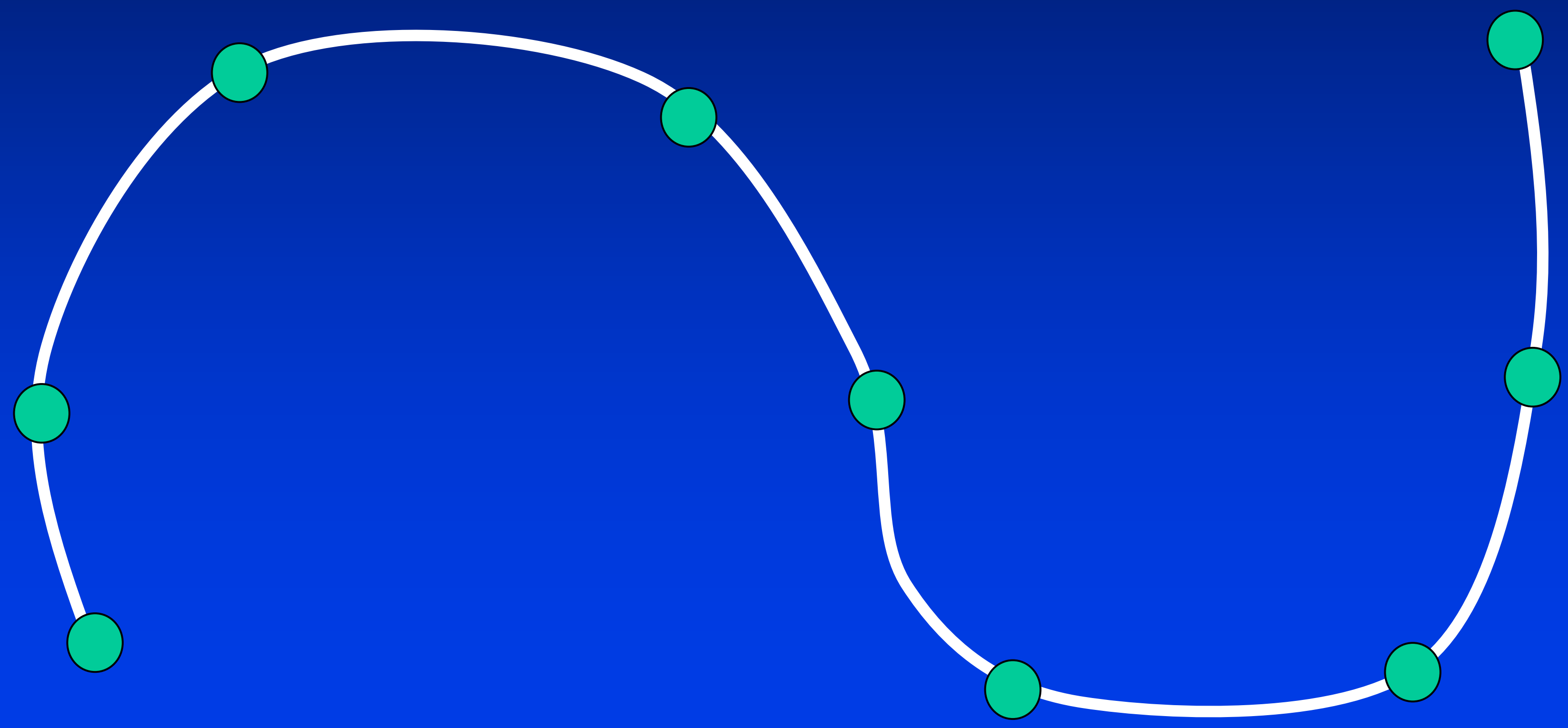
Natural C2 Cubic Splines

- A set of piecewise cubic polynomials

$$\mathbf{c}_i(u) = \begin{bmatrix} x(u) \\ y(u) \\ z(u) \end{bmatrix}$$

- C2 continuity at each vertex

Natural C2 Cubic Splines



Natural Splines

- Interpolate all control points
- Equivalent to a thin strip of metal in a physical sense
- Forced to pass through a set of desired points
- No local control (global control)
- N+1 control points
- N pieces
- 2(n-1) conditions
- We need two additional conditions

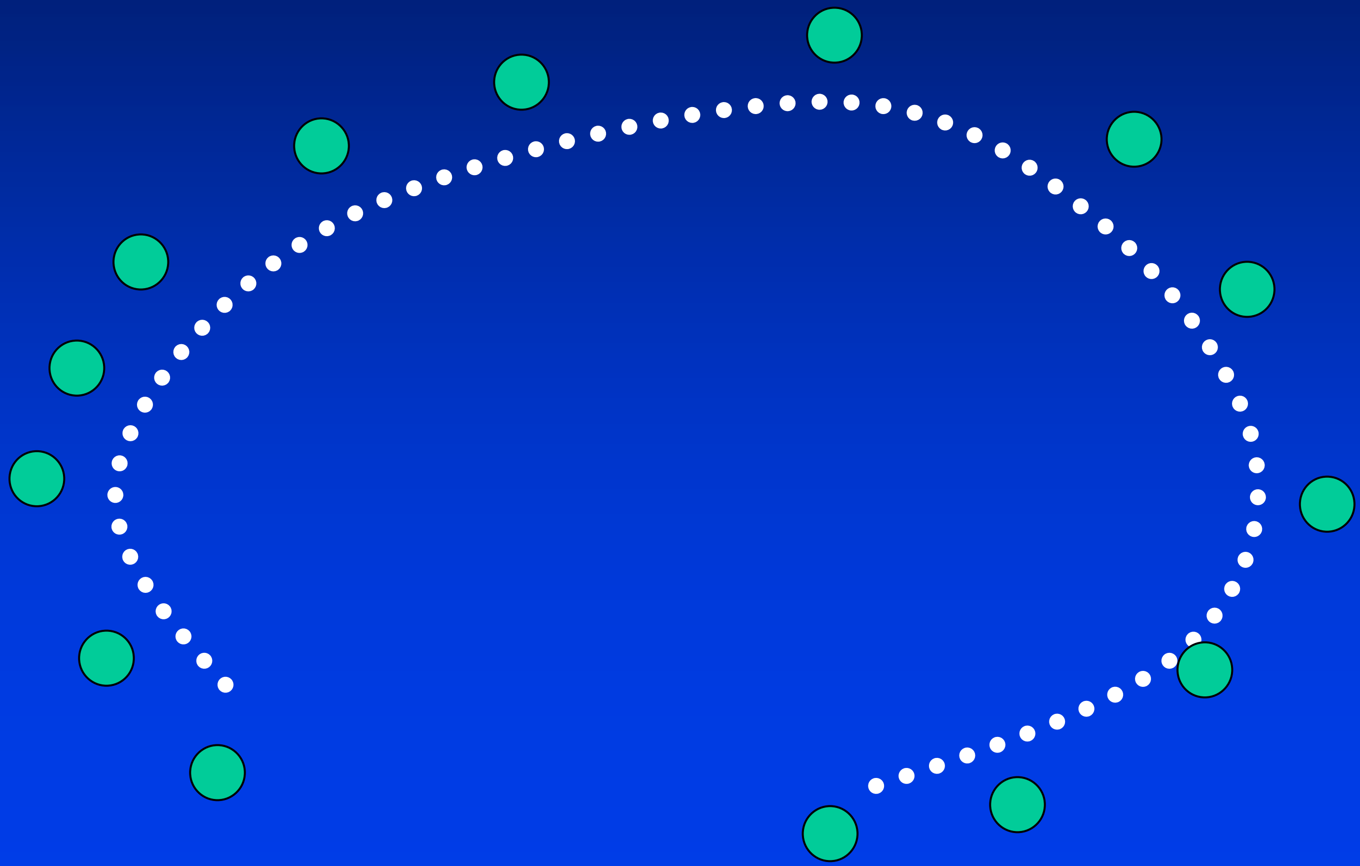
Natural Splines

- Interactive design system
  - Specify derivatives at two end-points
  - Specify the two internal control points that define the first curve span
  - Natural end conditions: second-order derivatives at two end points are defined to be zero
- Advantages: interpolation, C2
- Disadvantages: no local control (if one point is changed, the entire curve will move)
- How to overcome this drawback: B-Splines

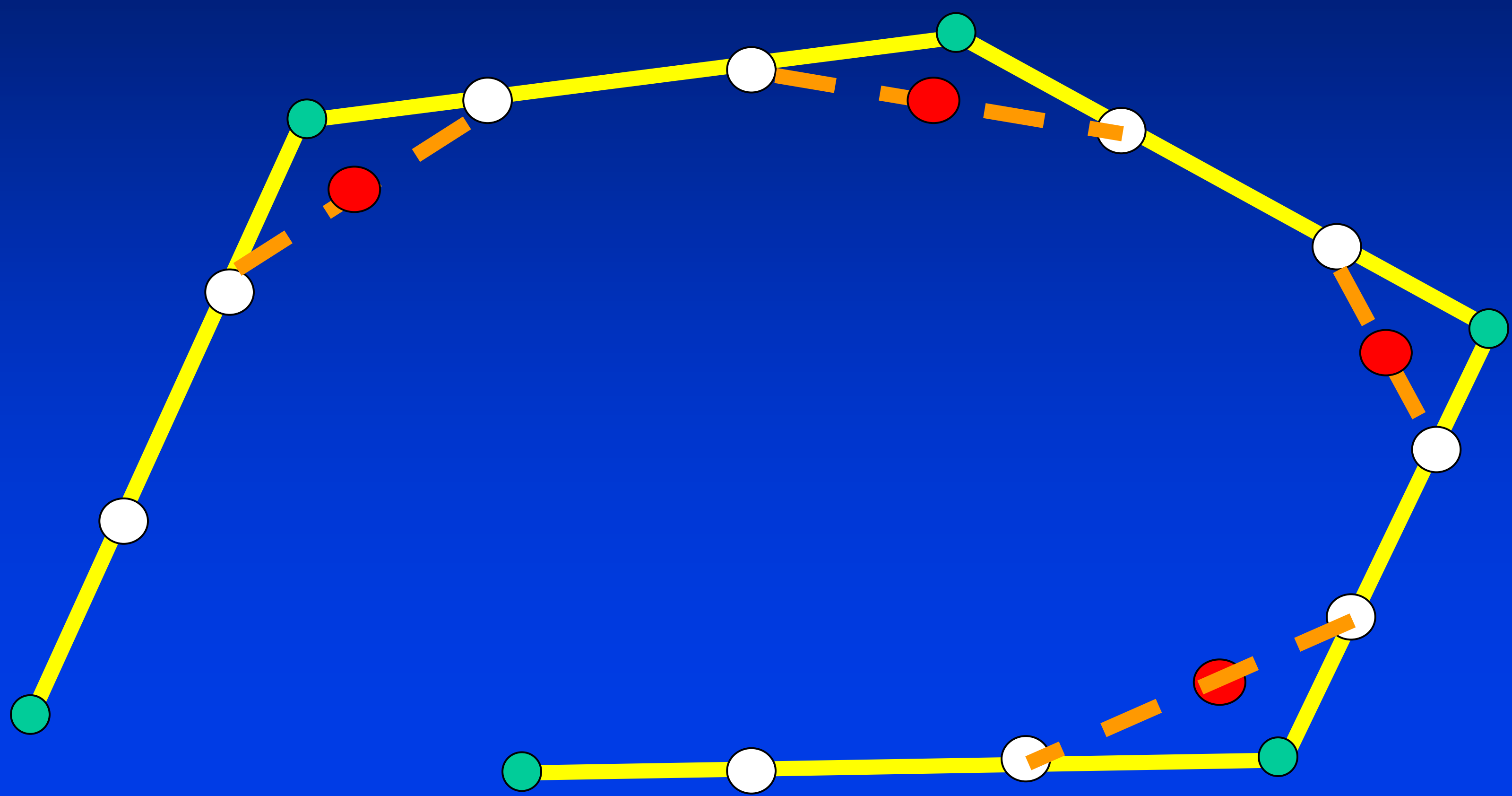
B-Splines Motivation

- The goal is local control!!!
- B-splines provide local control
- Do not interpolate control points
- C2 continuity
- Alternatively
  - Catmull-Rom Splines
  - Keep interpolations
  - Give up C2 continuity (only C1 is achieved)
  - Will be discussed later!!!

C2 Approximating Splines



From B-Splines to Bezier



Uniform B-Splines

- B-spline control points: $\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n$
- Piecewise Bezier curves with C2 continuity at joints
- Bezier control points:

$$\mathbf{v}_0 = \mathbf{p}_0$$
$$\mathbf{v}_1 = \frac{2\mathbf{p}_1 + \mathbf{p}_2}{3}$$
$$\mathbf{v}_2 = \frac{\mathbf{p}_1 + 2\mathbf{p}_2}{3}$$
$$\mathbf{v}_0 = \frac{1}{2}\left(\frac{\mathbf{p}_0 + 2\mathbf{p}_1}{3} + \frac{2\mathbf{p}_1 + \mathbf{p}_2}{3}\right) = \frac{1}{6}(\mathbf{p}_0 + 4\mathbf{p}_1 + \mathbf{p}_2)$$
$$\mathbf{v}_3 = \frac{1}{6}(\mathbf{p}_1 + 4\mathbf{p}_2 + \mathbf{p}_3)$$

Uniform B-Splines

- In general, I-th segment of B-splines is determined by four consecutive B-spline control points

$$\mathbf{v}_1 = \frac{2\mathbf{p}_{i+1} + \mathbf{p}_{i+2}}{3}$$

$$\mathbf{v}_2 = \frac{\mathbf{p}_{i+1} + 2\mathbf{p}_{i+2}}{3}$$

$$\mathbf{v}_0 = \frac{1}{6}(\mathbf{p}_i + 4\mathbf{p}_{i+1} + \mathbf{p}_{i+2})$$

$$\mathbf{v}_3 = \frac{1}{6}(\mathbf{p}_{i+1} + 4\mathbf{p}_{i+2} + \mathbf{p}_{i+3})$$

Uniform B-Splines

- In matrix form

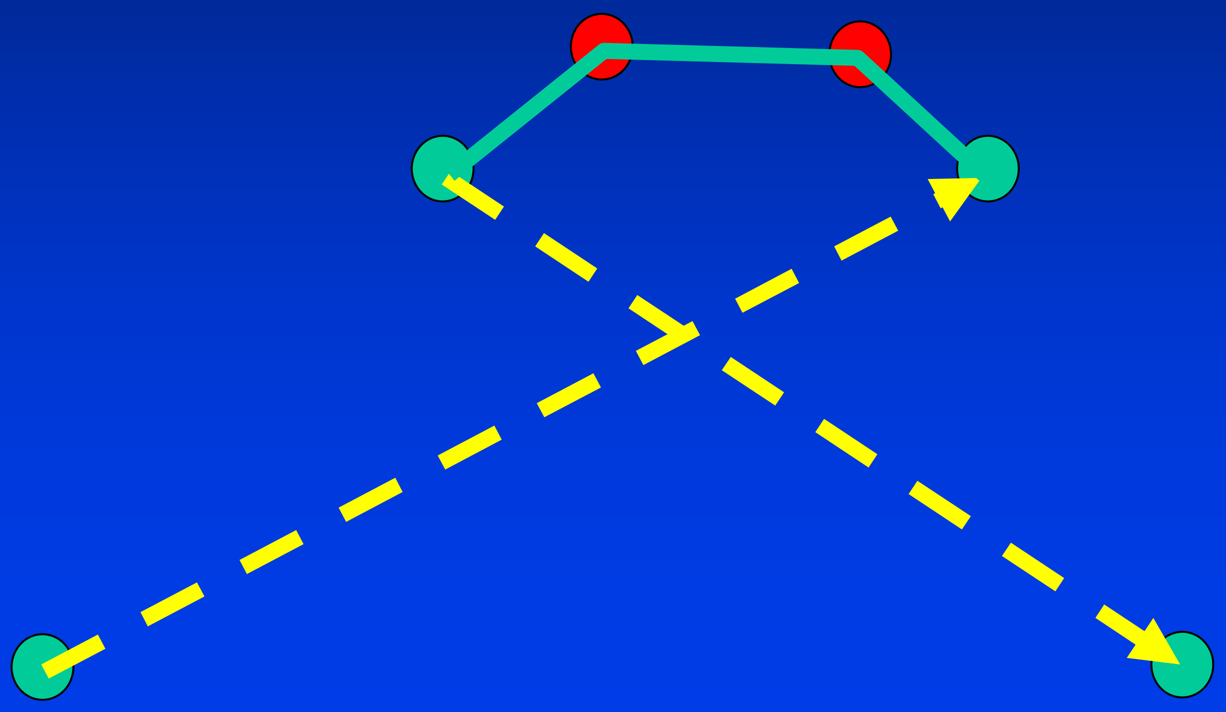
$$\begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 1 & 4 & 1 & 0 \\ 0 & 4 & 2 & 0 \\ 0 & 2 & 4 & 0 \\ 0 & 1 & 4 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \\ \mathbf{p}_{i+3} \end{bmatrix}$$

- Question: how many Bezier segments???

B-Spline Properties

- C2 continuity, Approximation, Local control, convex hull
- Each segment is determined by four control points
- Questions: what happens if we put more than one control points in the same location???
  - Double vertices, triple vertices, collinear vertices
- End conditions
  - Double endpoints: curve will be tangent to line between first distinct points
  - Triple endpoint: curve interpolate endpoint, start with a line segment
- B-spline display: transform it to Bezier curves

Catmull-Rom Splines



Catmull-Rom Splines

- Keep interpolation
- Give up C2 continuity
- Control tangents locally
- Idea: Bezier curve between successive points
- How to determine two internal vertices

$$\mathbf{c}(0) = \mathbf{p}_i = \mathbf{v}_0, \mathbf{c}(1) = \mathbf{p}_{i+1} = \mathbf{v}_3$$

$$\mathbf{c}'(0) = \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2} = 3(\mathbf{v}_1 - \mathbf{v}_0)$$

$$\mathbf{c}'(1) = \frac{\mathbf{p}_{i+2} - \mathbf{p}_i}{2} = 3(\mathbf{v}_3 - \mathbf{v}_2)$$

$$\mathbf{v}_1 = \frac{\mathbf{p}_{i+1} + 6\mathbf{p}_i - \mathbf{p}_{i-1}}{6}$$

$$\mathbf{v}_2 = \frac{-\mathbf{p}_{i+2} + 6\mathbf{p}_{i+1} + \mathbf{p}_i}{6}$$

Catmull-Rom Splines

- In matrix form

$$\begin{bmatrix} \mathbf{v}_0 \\ \mathbf{v}_1 \\ \mathbf{v}_2 \\ \mathbf{v}_3 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 0 & 6 & 0 & 0 \\ -1 & 6 & 1 & 0 \\ 0 & 1 & 6 & -1 \\ 0 & 0 & 6 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}$$

- Problem: boundary conditions
- Properties: C1, interpolation, local control, non-convex-hull

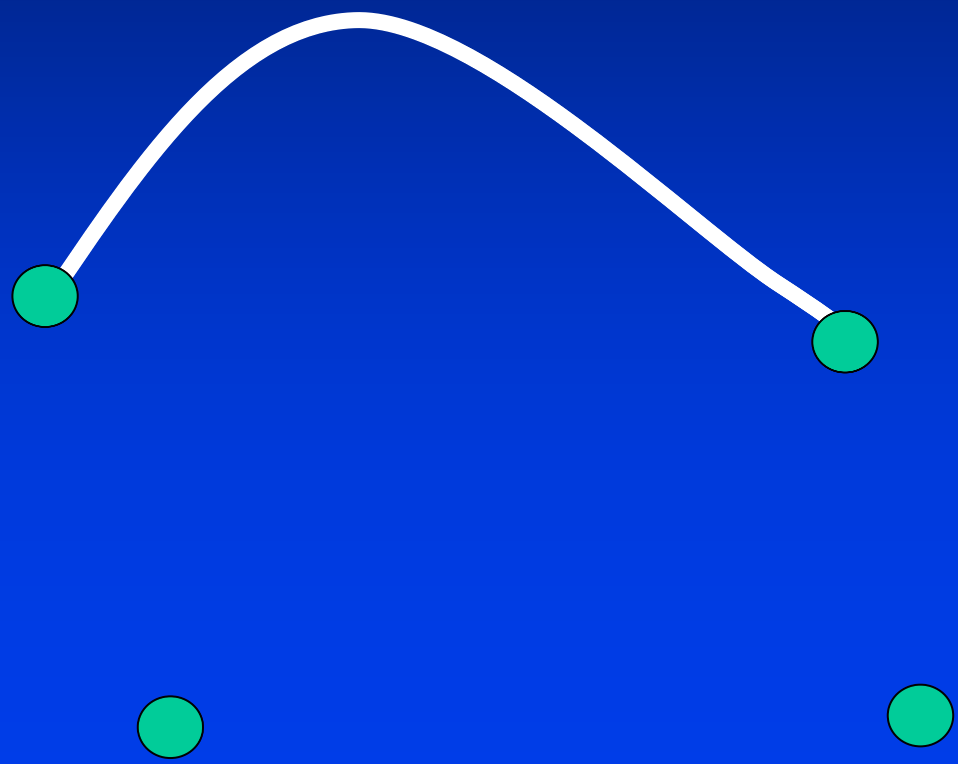
Cardinal Splines

- Four vertices define end-points and their associated tangents

$$\mathbf{c}(0) = \mathbf{v}_1, \mathbf{c}(1) = \mathbf{v}_2$$
$$\mathbf{c}^{(1)}(0) = \frac{1}{2}(1-\alpha)(\mathbf{v}_2 - \mathbf{v}_0)$$
$$\mathbf{c}^{(1)}(1) = \frac{1}{2}(1-\alpha)(\mathbf{v}_3 - \mathbf{v}_1)$$

- Special case: Catmull-Rom splines when $\alpha = 0$
- More general case: Kochanek-Bartels splines
  - Tension, bias, continuity parameters

Cardinal Splines



Kochanek-Bartels Splines

- Four vertices to define four conditions

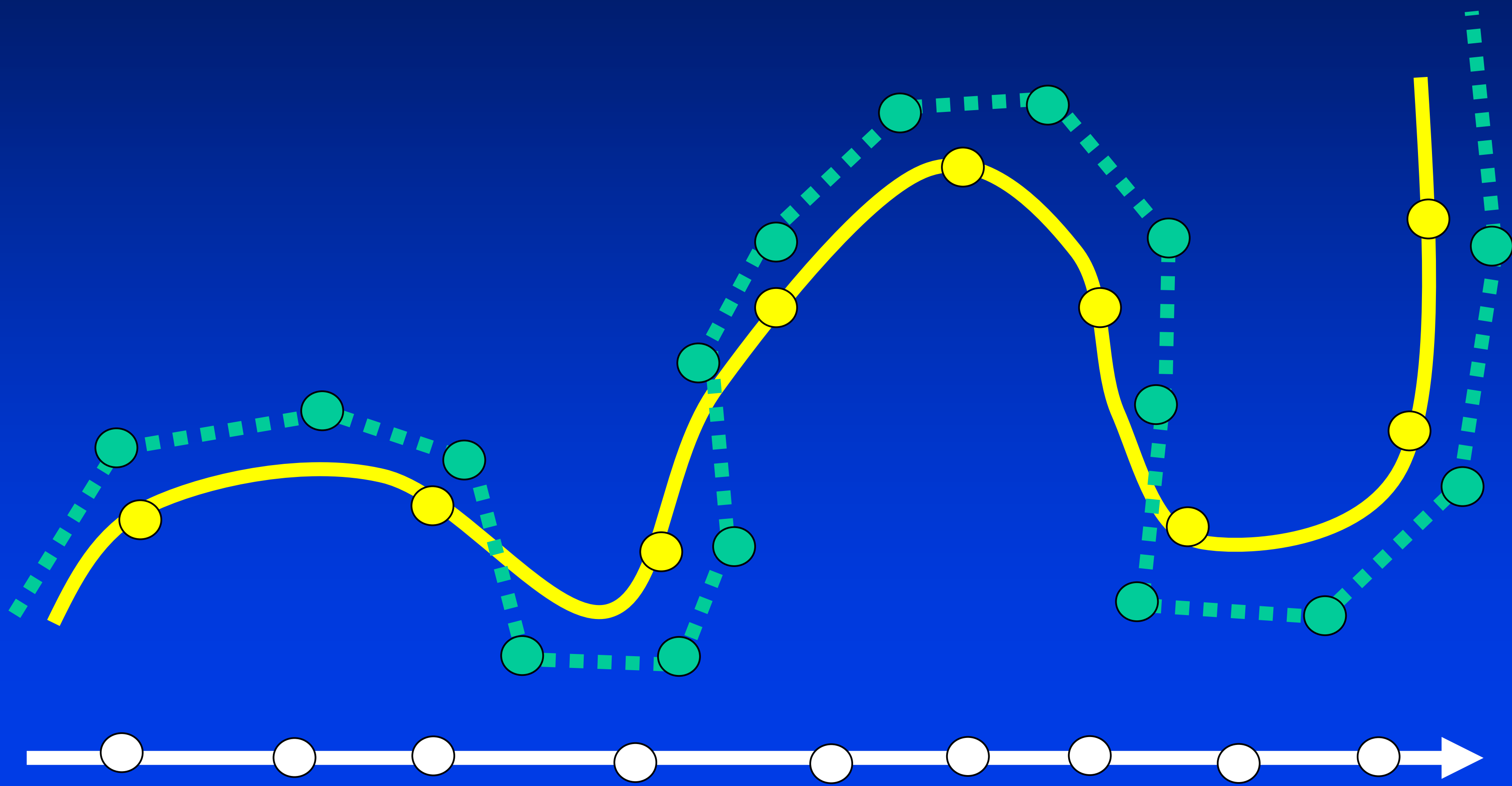
$$\mathbf{c}(0)=\mathbf{v}_1,\mathbf{c}(1)=\mathbf{v}_2$$

$$\mathbf{c}^{(1)}(0)=\frac{1}{2}(1-\alpha)((1+\beta)(1-\gamma)(\mathbf{v}_1-\mathbf{v}_0)+(1-\beta)(1+\gamma)(\mathbf{v}_2-\mathbf{v}_1))$$

$$\mathbf{c}^{(1)}(1)=\frac{1}{2}(1-\alpha)((1+\beta)(1+\gamma)(\mathbf{v}_2-\mathbf{v}_1)+(1-\beta)(1-\gamma)(\mathbf{v}_3-\mathbf{v}_2))$$

  - Tension parameter: α
  - Bias parameter: β
  - Continuity parameter: γ

Piecewise B-Splines



B-Spline Basis Functions

$$B_{i,1}(u) = \begin{cases} 1 & u_i <= u < u_{i+1} \\ 0 & otherwise \end{cases}$$

$$B_{i,k}(u) = \frac{u - u_i}{u_{i+k-1} - u_i} B_{i,k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} B_{i+1,k-1}(u)$$

Basis Functions

- Linear examples

$$B_{0,2}(u) = \begin{cases} u & u \in [0,1] \\ 2-u & u \in [1,2] \end{cases}$$

$$B_{1,2}(u) = \begin{cases} u-1 & u \in [1,2] \\ 3-u & u \in [2,3] \end{cases}$$

$$B_{2,2}(u) = \begin{cases} u-2 & u \in [2,3] \\ 4-u & u \in [3,4] \end{cases}$$

- How does it look like???

Basis Functions

- Quadratic cases (knot vector is [0,1,2,3,4,5,6])

$$B_{0,3}(u) = \begin{cases} \frac{1}{2}u^2, & 0 <= u < 1 \\ \frac{1}{2}u(2-u) + \frac{1}{2}(u-1)(3-u), & 1 <= u < 2 \\ \frac{1}{2}(3-u)^2, & 2 <= u < 3 \end{cases}$$

$$B_{1,3}(u) = \begin{cases} \frac{1}{2}(u-1)^2, & 1 <= u < 2 \\ \frac{1}{2}(u-1)(3-u) + \frac{1}{2}(u-2)(4-u), & 2 <= u < 3 \\ \frac{1}{2}(4-u)^2, & 3 <= u < 4 \end{cases}$$

$$B_{2,3}(u) = \ldots\ldots$$

$$B_{3,3}(u) = \ldots\ldots$$

- Cubic example

B-Spline Basis Function Image

B-Splines

- Mathematics $$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(u)$$
- Control points and basis functions of degree (k-1)
- Piecewise polynomials
- Basis functions are defined recursively
- We also have to introduce a knot sequence (n+k+1) in a non-decreasing order

$$u_0, u_1, u_2, u_3, \ldots\ldots, \quad u_{n+k}$$

- Note that, the parametric domain: $u \in [u_{k-1}, u_{n+1}]$

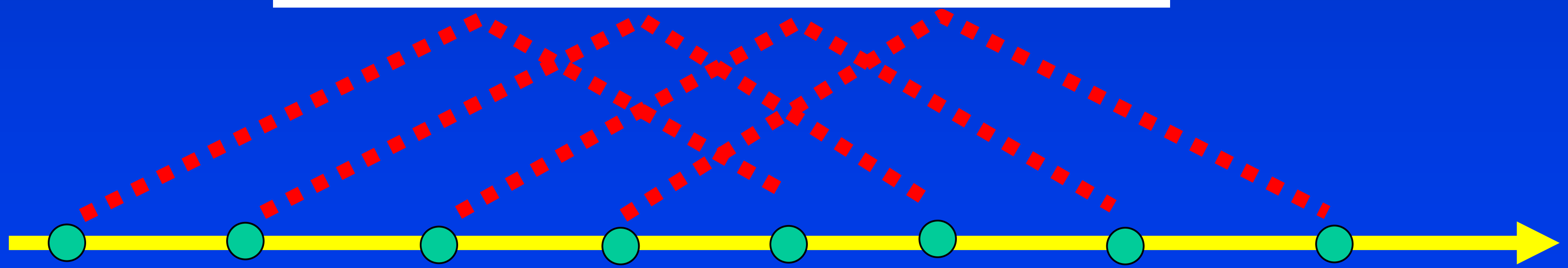
Basis Functions

$$B_{0,1} \quad B_{1,1} \quad B_{2,1} \quad B_{3,1} \quad B_{4,1} \quad B_{5,1} \quad B_{6,1}$$

$$B_{0,2} \quad B_{1,2} \quad B_{2,2} \quad B_{3,2} \quad B_{4,2} \quad B_{5,2}$$

$$B_{0,3} \quad B_{1,3} \quad B_{2,3} \quad B_{3,3} \quad B_{4,3}$$

$$B_{0,4} \quad B_{1,4} \quad B_{2,4} \quad B_{3,4}$$



B-Spline Facts

- The curve is a linear combination of control points and their associated basis functions ((n+1) control points and basis functions, respectively)
- Basis functions are piecewise polynomials defined (recursively) over a set of non-decreasing knots

$$\{u_0,\ldots\ldots,u_{k-1},\ldots\ldots,u_{n+1},\ldots\ldots,u_{n+k}\}$$

- The degree of basis functions is independent of the number of control points (note that, I is index, k is the order, k-1 is the degree)
- The first k and last k knots do NOT contribute to the parametric domain. Parametric domain is only defined by a subset of knots

B-Spline Properties

- C(u): piecewise polynomial of degree (k-1)
- Continuity at joints: C(k-2)
- The number of control points and basis functions: (n+1)
- One typical basis function is defined over k sub-intervals which are specified by k+1 knots ([u(k),u(I+k)])
- There are n+k+1 knots in total, knot sequence divides the parametric axis into n+k sub-intervals
- There are (n+1)-(k-1)=n-k+2 sub-intervals within the parametric domain ([u(k-1),u(n+1)])

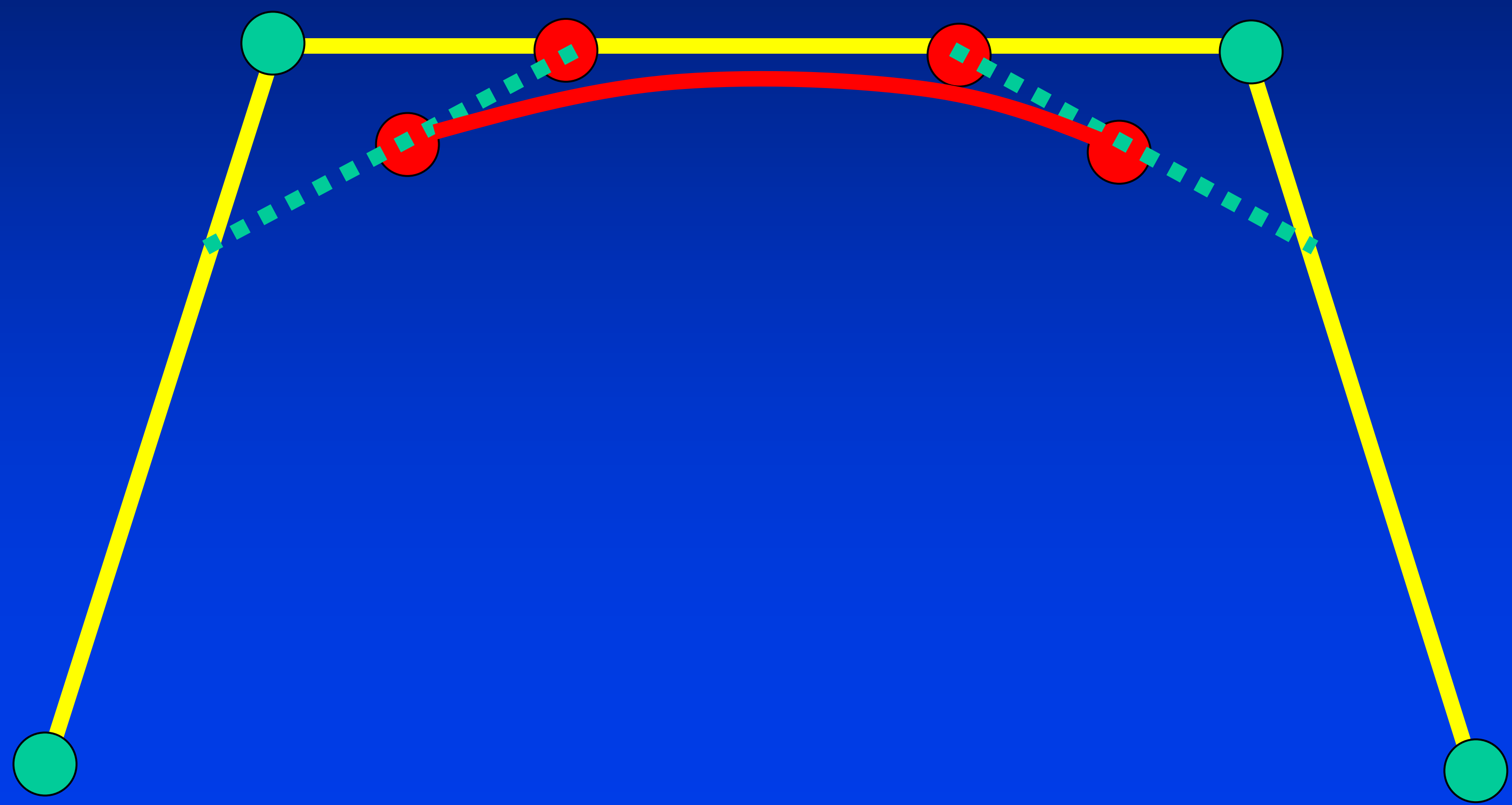
B-Spline Properties

- There are n-k+2 piecewise polynomials
- Each curve span is influenced by k control points
- Each control points at most affects k curve spans
- Local control!!!
- Convex hull
- The degree of B-spline polynomial can be independent from the number of control points
- Compare B-spline with Bezier!!!
- Key components: control points, basis functions, knots, parametric domain, local vs. global control, continuity

B-Spline Properties

- Partition of unity, positivity, and recursive evaluation of basis functions
- Special cases: Bezier splines
- Efficient algorithms and tools
  - Evaluation, knot insertion, degree elevation, derivative, integration, continuity
- Composite Bezier curves for B-splines

Uniform B-Spline



Another Formulation

- Uniform B-spline
- Parameter normalization (u is in [0,1])
- End-point positions and tangents

$$\mathbf{c}(0) = \frac{1}{6}(\mathbf{p}_0 + 4\mathbf{p}_1 + \mathbf{p}_2)$$
$$\mathbf{c}(1) = \frac{1}{6}(\mathbf{p}_1 + 4\mathbf{p}_2 + \mathbf{p}_3)$$
$$\mathbf{c}'(0) = \frac{1}{2}(\mathbf{p}_2 - \mathbf{p}_0)$$
$$\mathbf{c}'(1) = \frac{1}{2}(\mathbf{p}_3 - \mathbf{p}_1)$$

Another Formulation

- Matrix representation

$$\mathbf{c}(u) = UM_h \begin{bmatrix} \mathbf{c}(0) \\ \mathbf{c}(1) \\ \mathbf{c}'(0) \\ \mathbf{c}'(1) \end{bmatrix} = UM_h M' \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix} = UM\mathbf{p}$$

- Basis matrix

$$M = \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}$$

Basis Functions

- Note that, u is now in [0,1]

$$B_{0,4}(u) = \frac{1}{6}(1-u)^3$$

$$B_{1,4}(u) = \frac{1}{6}(3u^3 - 6u^2 + 4)$$

$$B_{2,4}(u) = \frac{1}{6}(-3u^3 + 3u^2 + 3u + 1)$$

$$B_{3,4}(u) = \frac{1}{6}(u)^3$$

B-Spline Rendering

- Transform it to a set of Bezier curves
- Convert the I-th span into a Bezier representation

$$\mathbf{p}_{i}, \mathbf{p}_{i+1}, \ldots\ldots, \mathbf{p}_{i+k-1}$$
$$\mathbf{v}_{0}, \mathbf{v}_{1}, \ldots\ldots, \mathbf{v}_{k-1}$$

- Consider the entire B-spline curve

$$\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots\ldots, \mathbf{p}_n$$
$$\mathbf{v}_0, \ldots\ldots, \mathbf{v}_3, \mathbf{v}_4, \ldots\ldots, \mathbf{v}_7, \ldots\ldots, \mathbf{v}_{4(n-3)}, \ldots\ldots, \mathbf{v}_{4(n-3)+3}$$

Matrix Expression

$$\begin{bmatrix} \mathbf{v}_0 \\ \vdots \\ \mathbf{v}_{4(n-3)+3} \end{bmatrix} = \mathbf{B} \begin{bmatrix} \mathbf{p}_0 \\ \vdots \\ \mathbf{p}_n \end{bmatrix}$$

- The matrix structure and components of B?

$$\mathbf{q} = \mathbf{A}\mathbf{v} = \mathbf{A}\mathbf{B}$$

- The matrix structure and components of A?

B-Spline Discretization

- Parametric domain: [u(k-1),u(n+1)]
- There are n+2-k curve spans (pieces)
- Assuming m+1 points per span (uniform sampling)
- Total sampling points m(n+2-k)+1=l
- B-spline discretization with corresponding parametric values:

$$\mathbf{q}_0, \ldots\ldots, \quad \mathbf{q}_{l-1}$$

$$\mathbf{v}_0, \ldots\ldots, \quad \mathbf{v}_{l-1}$$

$$\mathbf{q}_i = \mathbf{c}(v_i) = \sum_{j=0}^{n} \mathbf{p}_j B_{j,k}(v_i)$$

B-Spline Discretization

- Matrix equation

$$\begin{bmatrix} \mathbf{q}_0 \\ M \\ \mathbf{q}_{l-1} \end{bmatrix} = \begin{bmatrix} B_{0,k}(v_0) & \Lambda & B_{n,k}(v_0) \\ M & O & M \\ B_{0,k}(v_{l-1}) & \Lambda & B_{n,k}(v_{l-1}) \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ M \\ \mathbf{p}_n \end{bmatrix}$$

- A is (l)x(n+1) matrix, in general (l) is much larger than (n+1), so A is sparse
- The linear discretization for both modeling and rendering

From B-Splines to NURBS

- What are NURBS???
- Non Uniform Rational B-Splines (NURBS)
- Rational curve motivation
- Polynomial-based splines can not represent commonly-used analytic shapes such as conic sections (e.g., circles, ellipses, parabolas)
- Rational splines can achieve this goal
- NURBS are a unified representation
  - Polynomial, conic section, etc.
  - Industry standard

From B-Splines to NURBS

- B-splines

$$\mathbf{c}(u) = \sum_{i=0}^{n} \begin{bmatrix} \mathbf{p}_{i,x} w_i \\ \mathbf{p}_{i,y} w_i \\ \mathbf{p}_{i,z} w_i \\ w_i \end{bmatrix} B_{i,k}(u)$$

- NURBS (curve)

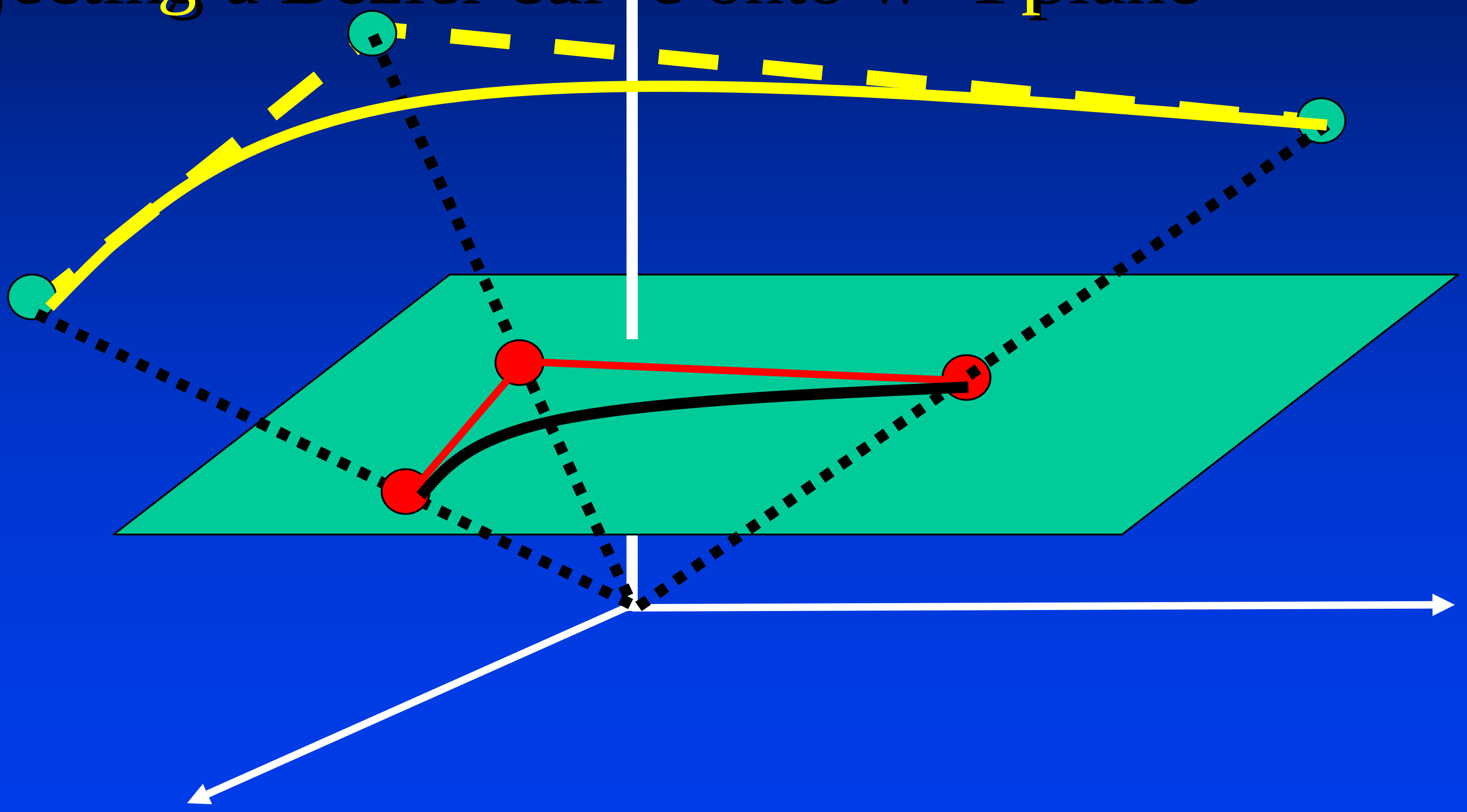
$$\mathbf{c}(u) = \frac{\sum_{i=0}^{n} \mathbf{p}_i w_i B_{i,k}(u)}{\sum_{i=0}^{n} w_i B_{i,k}(u)}$$

Geometric NURBS

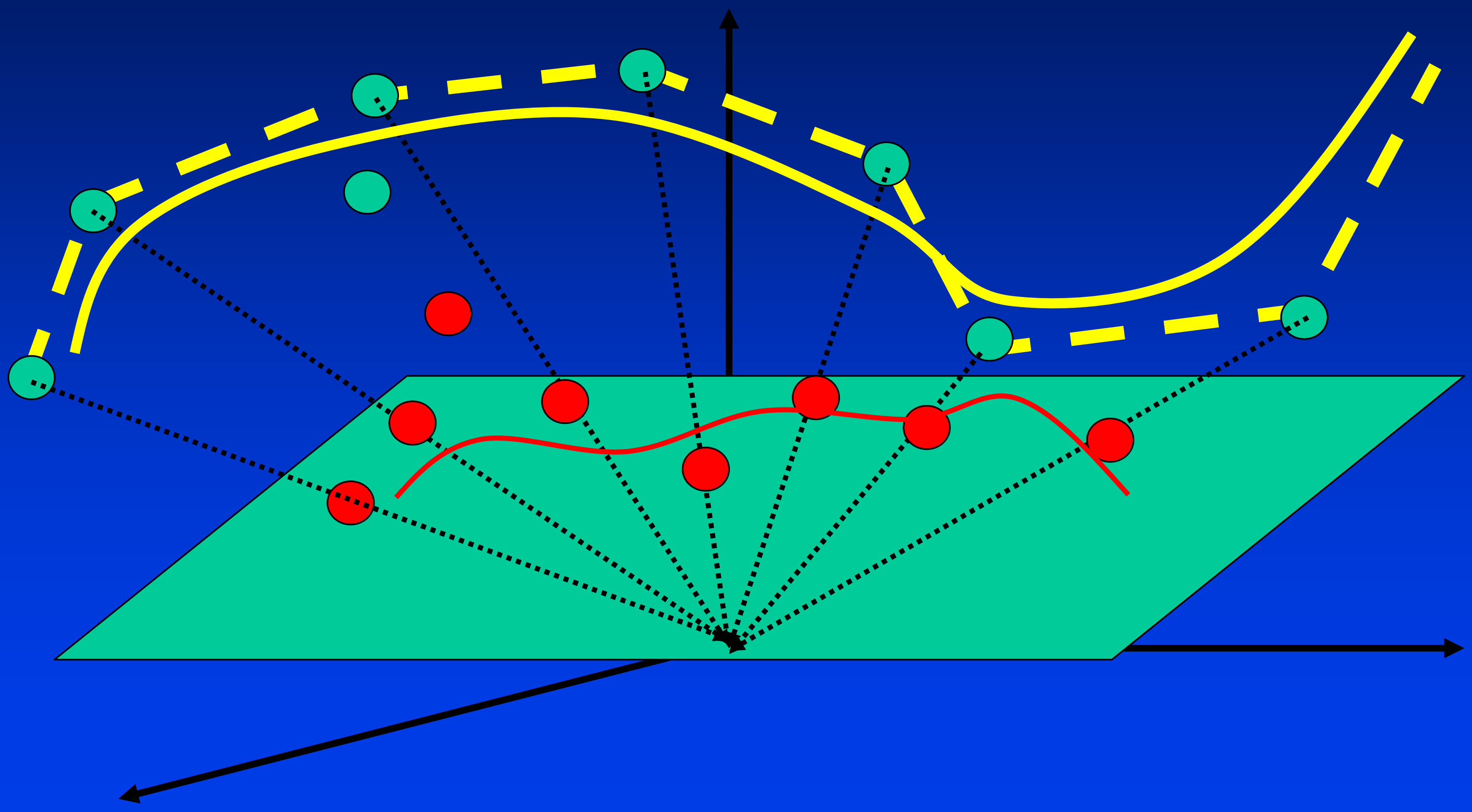
- Non-Uniform Rational B-Splines
- CAGD industry standard --- useful properties
- Degrees of freedom
  - Control points
  - Weights

Rational Bezier Curve

- Projecting a Bezier curve onto w=1 plane



From B-Splines to NURBS



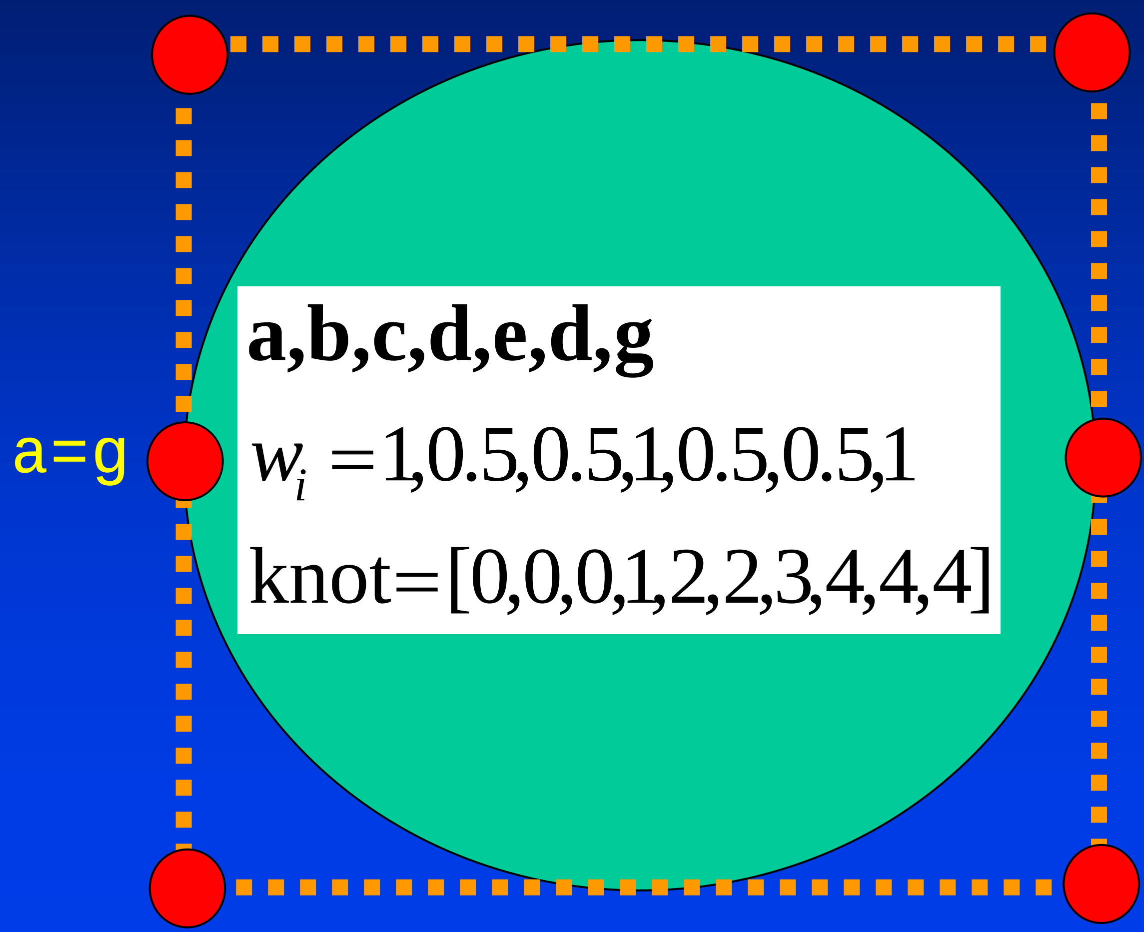
NURBS Weights

- Weight increase “attracts” the curve towards the associated control point
- Weight decrease “pushes away” the curve from the associated control point

NURBS for Analytic Shapes

- Conic sections
- Natural quadrics
- Extruded surfaces
- Ruled surfaces
- Surfaces of revolution

NURBS Circle



NURBS Curve

- Geometric components
  - Control points, parametric domain, weights, knots
- Homogeneous representation of B-splines
- Geometric meaning --- obtained from projection
- Properties of NURBS
  - Represent standard shapes, invariant under perspective projection, B-spline is a special case, weights as extra degrees of freedom, common analytic shapes such as circles, clear geometric meaning of weights

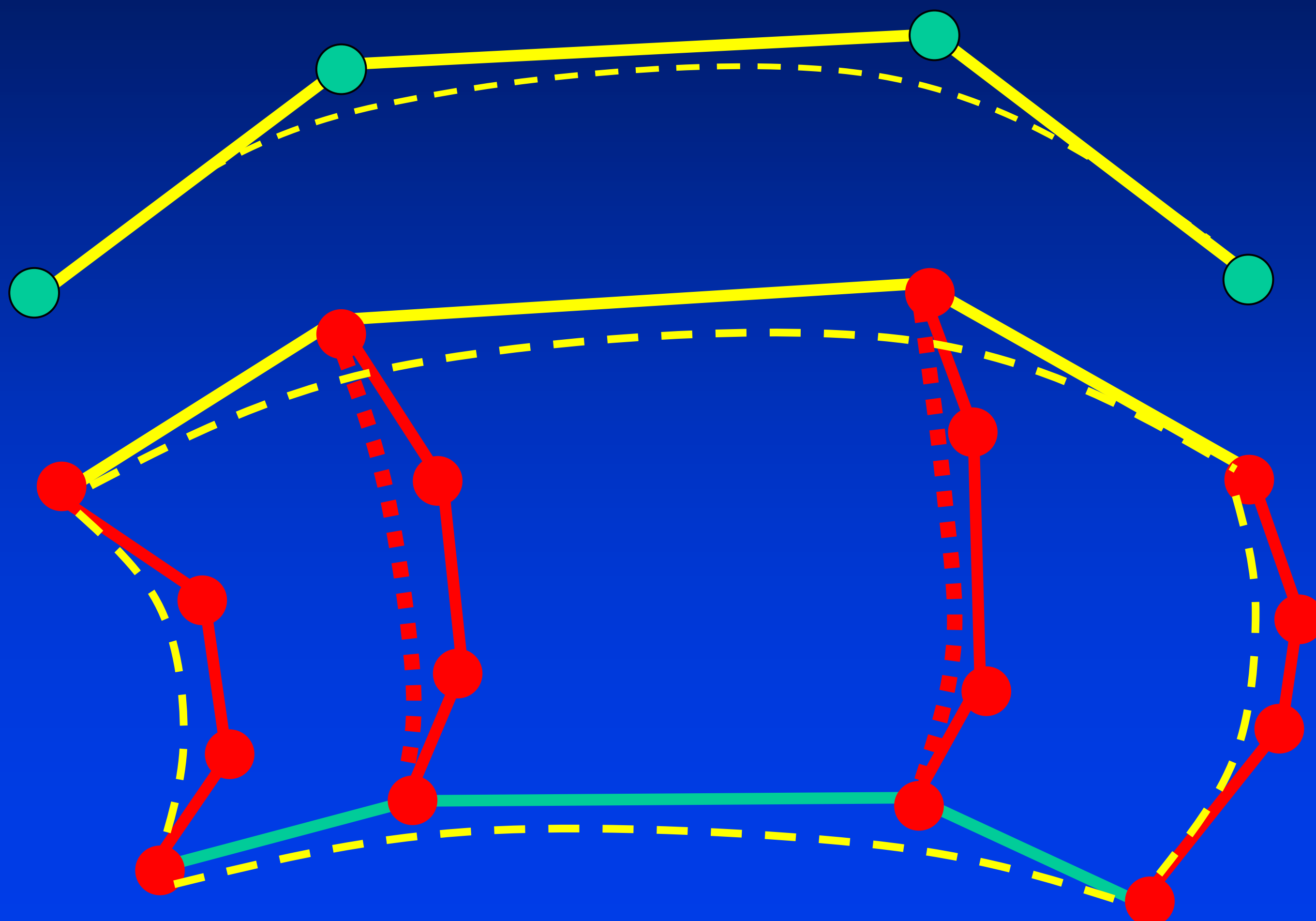
NURBS Properties

- Generalization of B-splines and Bezier splines
- Unified formulation for free-form and analytic shape
- Weights as extra DOFs
- Various smoothness requirements
- Powerful geometric toolkits
- Efficient and fast evaluation algorithm
- Invariance under standard transformations
- Composite curves
- Continuity conditions

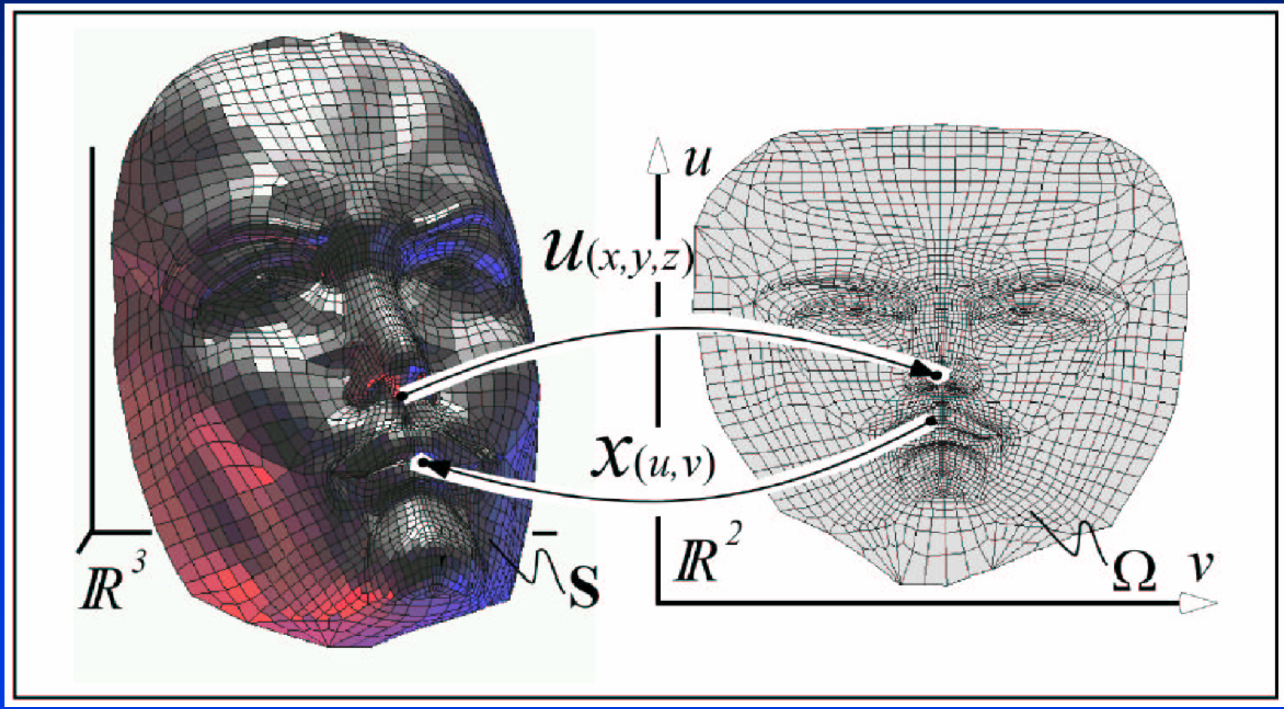
Geometric Modeling

- Why geometric modeling
- Fundamental for visual computing
  - Graphics, visualization
  - Computer aided design and manufacturing
  - Imaging
  - Entertainment, etc.
- Critical for virtual engineering
- Interaction
- Geometric information for decision making

From Curve to Surface



Parameterization



Surfaces

- From curves to surfaces
- A simple curve example (Bezier)

$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i(u)$$
$$u \in [0,1]$$

- Consider each control point now becoming a Bezier curve

$$\mathbf{p}_i = \sum_{j=0}^{3} \mathbf{p}_{i,j} B_j(v)$$
$$v \in [0,1]$$

Surfaces

- Then, we have

$$\mathbf{s}(u,v)=\sum_{i=0}^{3}(\sum_{j=0}^{3}\mathbf{p}_{i,j}B_j(v))B(u)=\sum_{i=0}^{3}\sum_{j=0}^{3}\mathbf{p}_{i,j}B_i(u)B_j(v)$$

- Matrix form

$$\mathbf{s}(u,v)=\begin{bmatrix}B_0(u) & B_1(u) & B_2(u) & B_3(u)\end{bmatrix}\begin{bmatrix}\mathbf{p}_{0,0} & \mathbf{p}_{0,1} & \mathbf{p}_{0,2} & \mathbf{p}_{0,3}\\ \mathbf{p}_{1,0} & \mathbf{p}_{1,1} & \mathbf{p}_{1,2} & \mathbf{p}_{1,3}\\ \mathbf{p}_{2,0} & \mathbf{p}_{2,1} & \mathbf{p}_{2,2} & \mathbf{p}_{2,3}\\ \mathbf{p}_{3,0} & \mathbf{p}_{3,1} & \mathbf{p}_{3,2} & \mathbf{p}_{3,3}\end{bmatrix}\begin{bmatrix}B_0(u)\\ B_1(u)\\ B_2(u)\\ B_3(u)\end{bmatrix}$$

$$=UMPM^TV^T$$

Surfaces

- Further generalize to degree of n and m along two parametric directions

$$\mathbf{s}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} \mathbf{p}_{i,j} B_i^n(u) B_j^m(v)$$

- Question: which control points are interpolated?
- How about B-spline surfaces???

Tensor Product Surfaces

- Where are they from?
- Monomial form

$$\mathbf{s}(u,v) = \sum_i \sum_j \mathbf{a}_{i,j} u^i v^j$$

- Bezier surface

$$\mathbf{s}(u,v) = \sum_i \sum_j \mathbf{p}_{i,j} B_i^m(u) B_j^n(v)$$

- B-spline surface

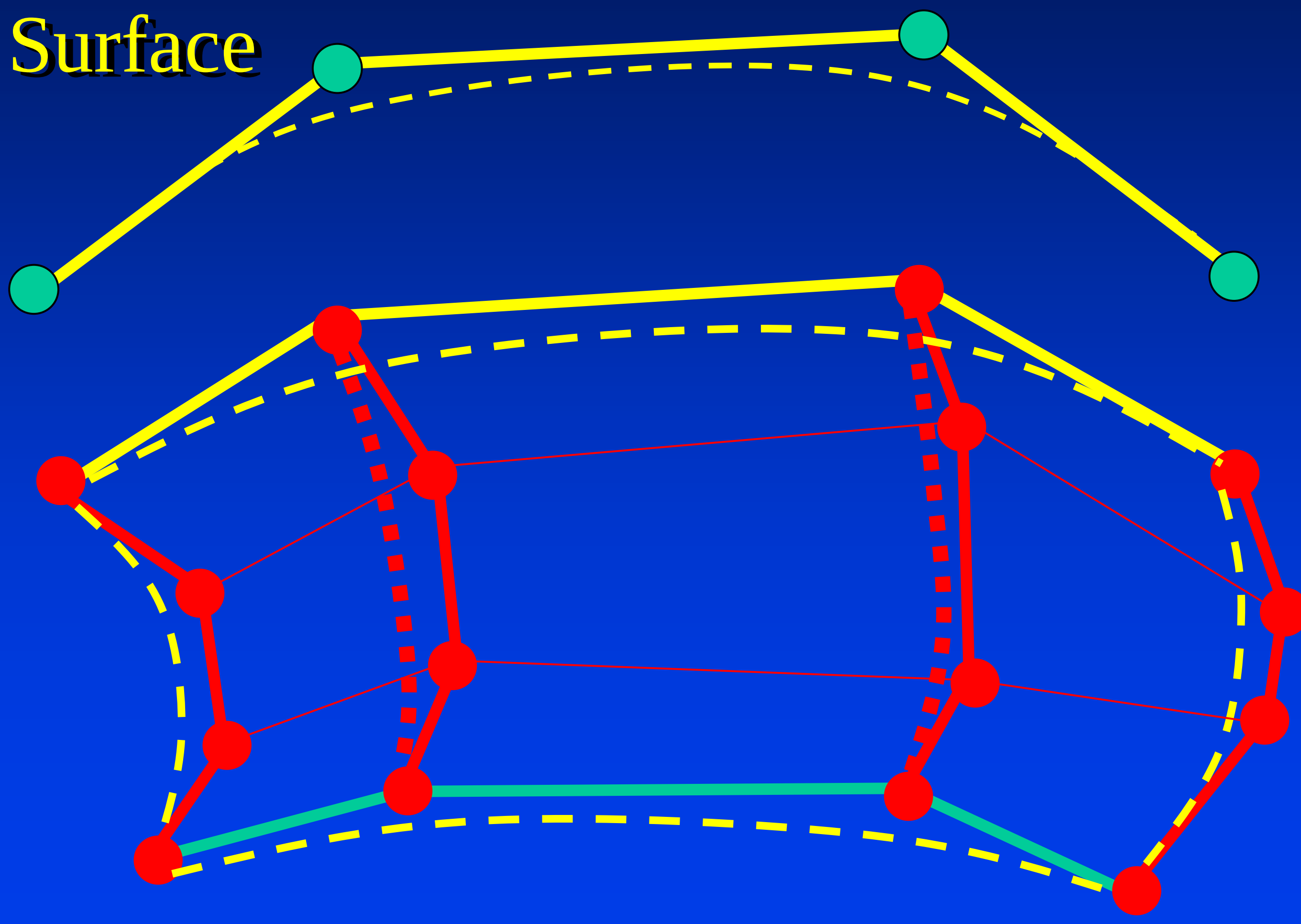
$$\mathbf{s}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{i,j} B_{i,k}(u) B_{j,l}(v)$$

- General case

$$\mathbf{s}(u,v) = \sum_i \sum_j \mathbf{v}_{i,j} F_i(u) G_j(v)$$

Tensor Product Surface

- Bezier Surface



B-Splines

- B-spline curves

$$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,k}(u)$$

- Tensor product B-splines

$$\mathbf{s}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{p}_{i,j} B_{i,k}(u) B_{j,l}(v)$$

- Question again: which control points are interpolated???
- Another question: can we get NURBS surface this way???
- Answer: NO!!! NURBS are not tensor-product surfaces
- Another question: can we have NURBS surface?
- YES!!!

NURBS Surface

- NURBS surface mathematics

$$\mathbf{s}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} \mathbf{p}_{i,j} w_{i,j} B_{i,k}(u) B_{j,l}(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m} w_{i,j} B_{i,k}(u) B_{j,l}(v)}$$

- Understand this geometric construction
- Question: why is it not the tensor-product formulation??? Compare it with Bezier and B-spline construction

NURBS Surface

- Parametric variables: u and v
- Control points and their associated weights: (m+1)(n+1)
- Degrees of basis functions: (k-1) and (l-1)
- Knot sequence:

$$u_0 <= u_1 <= \ldots\ldots <= u_{m+k}$$
$$v_0 <= v_1 <= \ldots\ldots <= v_{n+l}$$

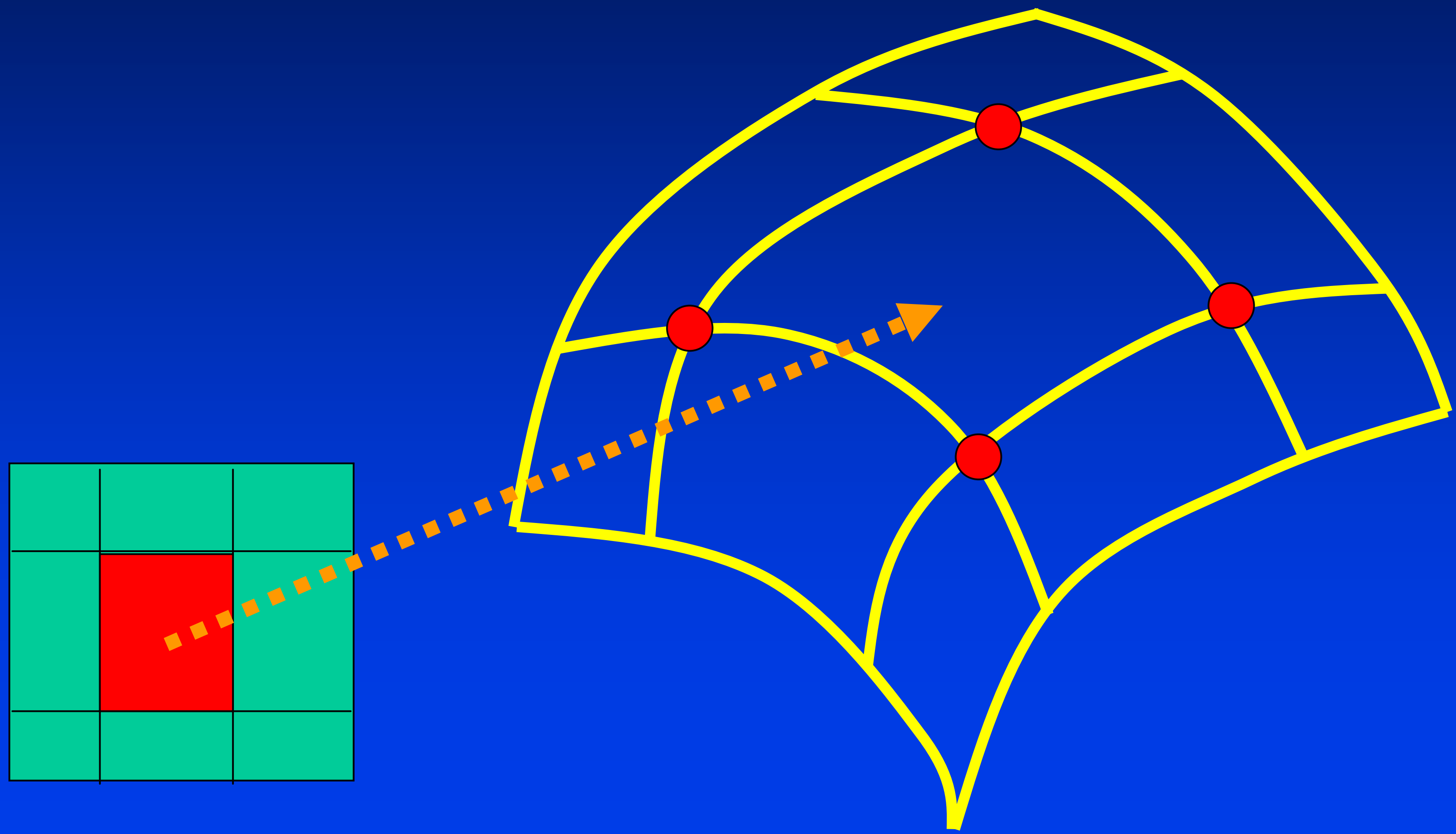
- Parametric domain:

$$u_{k-1} <= u <= u_{m+1}$$
$$v_{l-1} <= v <= v_{n+1}$$

NURBS Surface

- The same principle to generate curves via projection
- Idea: associate weights with control points
- Generalization of B-spline surface

Rectangular Surface



Hermite Surfaces

- How about Hermite surfaces???
- Hermite Curve

$$\mathbf{c}(u) = \begin{bmatrix} H_0(u) & H_1(u) & H_2(u) & H_3(u) \end{bmatrix} \begin{bmatrix} \mathbf{c}(0) \\ \mathbf{c}(1) \\ \mathbf{c}'(0) \\ \mathbf{c}'(1) \end{bmatrix}$$

- C(0) is not a curve s(0,v) which is also a Hermite Curve:

$$s(0,v) = \begin{bmatrix} H_0(v) & H_1(v) & H_2(v) & H_3(v) \end{bmatrix} \begin{bmatrix} \mathbf{s}(0,0) \\ \mathbf{s}(0,1) \\ \mathbf{s}_v(0,0) \\ \mathbf{s}_v(0,1) \end{bmatrix}$$

Hermite Surfaces

- Similarly, c(1) is now a curve s(1,v) which is also a Hermite curve:

$$s(1,v) = \begin{bmatrix} H_0(v) & H_1(v) & H_2(v) & H_3(v) \end{bmatrix} \begin{bmatrix} \mathbf{s}(1,0) \\ \mathbf{s}(1,1) \\ \mathbf{s}_v(1,0) \\ \mathbf{s}_v(1,1) \end{bmatrix}$$

- The same are for c'(0) and c'(1):

$$\mathbf{s}_u(0,v) = H(v) \begin{bmatrix} \mathbf{s}_u(0,0) \\ \mathbf{s}_u(0,1) \\ \mathbf{s}_{uv}(0,0) \\ \mathbf{s}_{uv}(0,1) \end{bmatrix}$$

$$\mathbf{s}_u(1,v) = H(v) \begin{bmatrix} \mathbf{s}_u(1,0) \\ \mathbf{s}_u(1,1) \\ \mathbf{s}_{uv}(1,0) \\ \mathbf{s}_{uv}(1,1) \end{bmatrix}$$

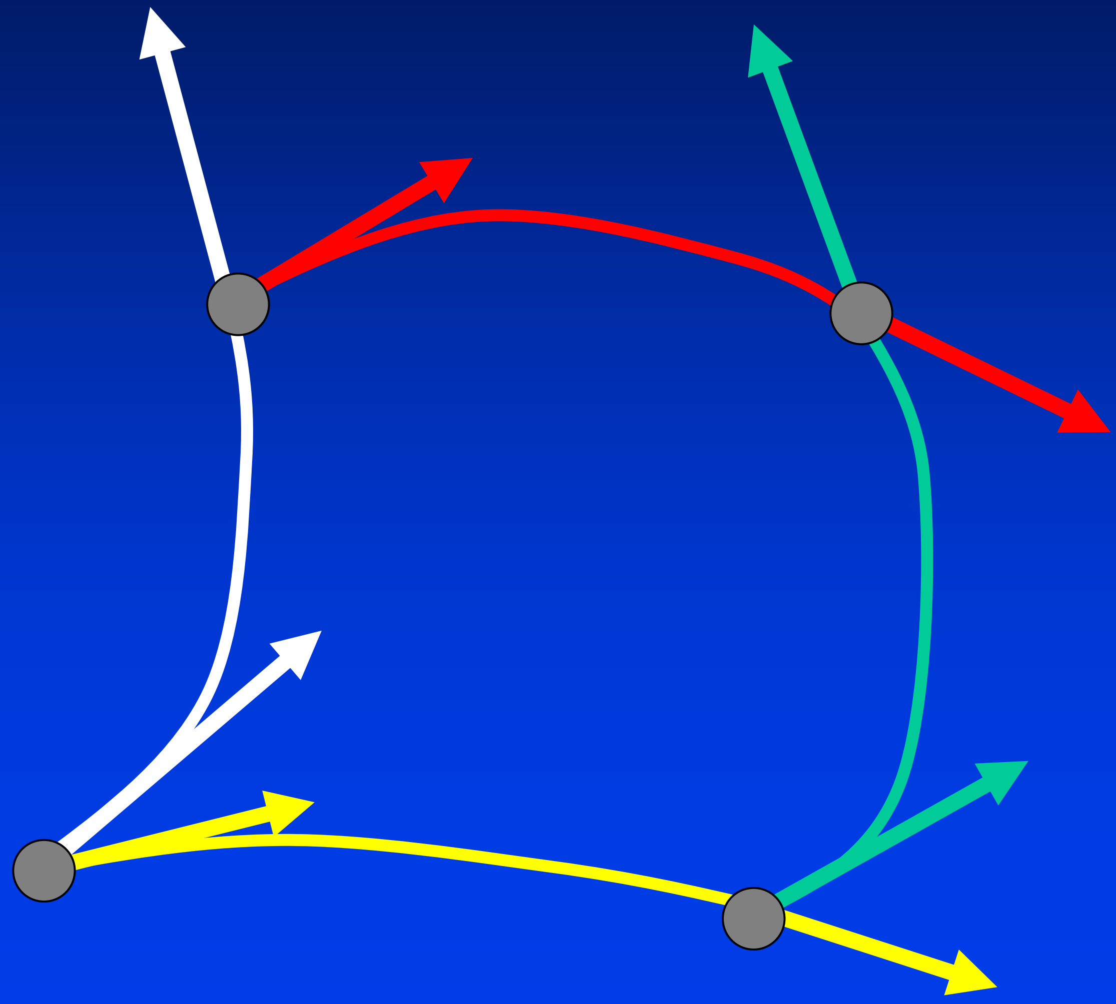
Hermite Surfaces

- It is time to put them together!

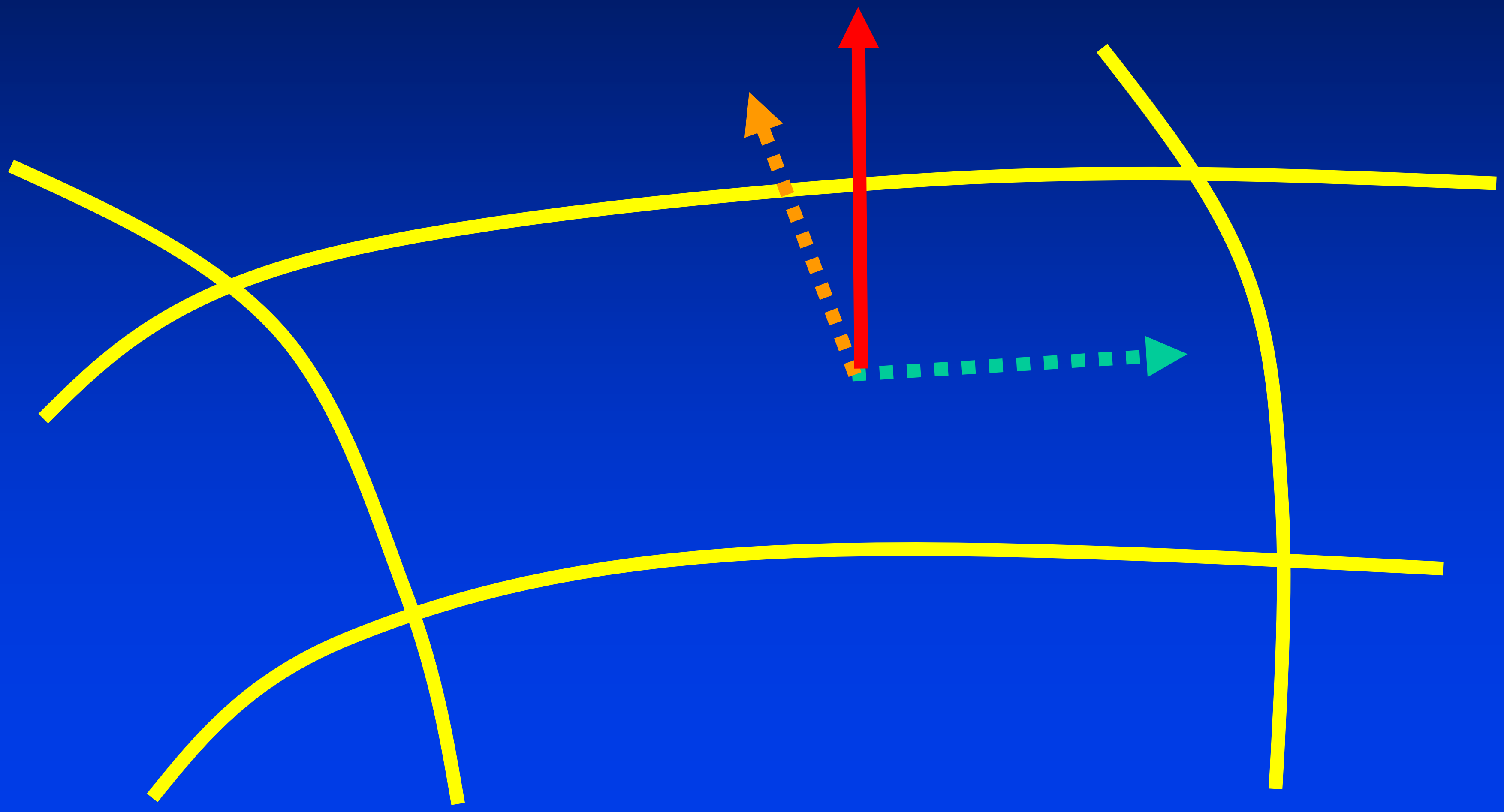
$$\mathbf{s}(u,v) = H(u)\begin{bmatrix} \mathbf{s}(0,0) & \mathbf{s}(0,1) & \mathbf{s}_v(0,0) & \mathbf{s}_v(0,1) \\ \mathbf{s}(1,0) & \mathbf{s}(1,1) & \mathbf{s}_v(1,0) & \mathbf{s}_v(1,1) \\ \mathbf{s}_u(0,0) & \mathbf{s}_u(0,1) & \mathbf{s}_{uv}(0,0) & \mathbf{s}_{uv}(0,1) \\ \mathbf{s}_u(1,0) & \mathbf{s}_u(1,1) & \mathbf{s}_{uv}(1,0) & \mathbf{s}_{uv}(1,1) \end{bmatrix} H(v)^T$$

- Continuity conditions for surfaces
- Bezier surfaces, B-splines, NURBS, Hermite surfaces
- C1 and G1 continuity

Hermite Surfaces

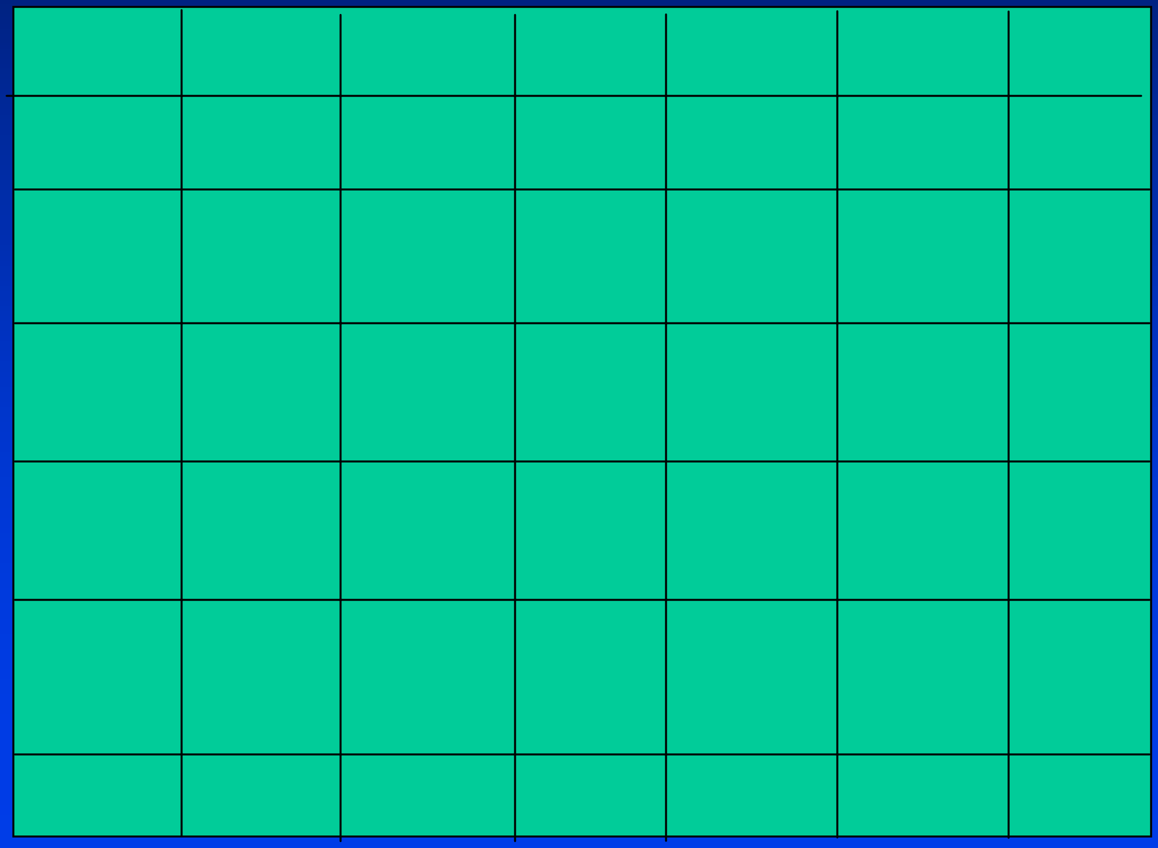


Surface Normal



Surface Rendering

- Parametric grids ([0,1]X[0,1]) as a set of rectangles

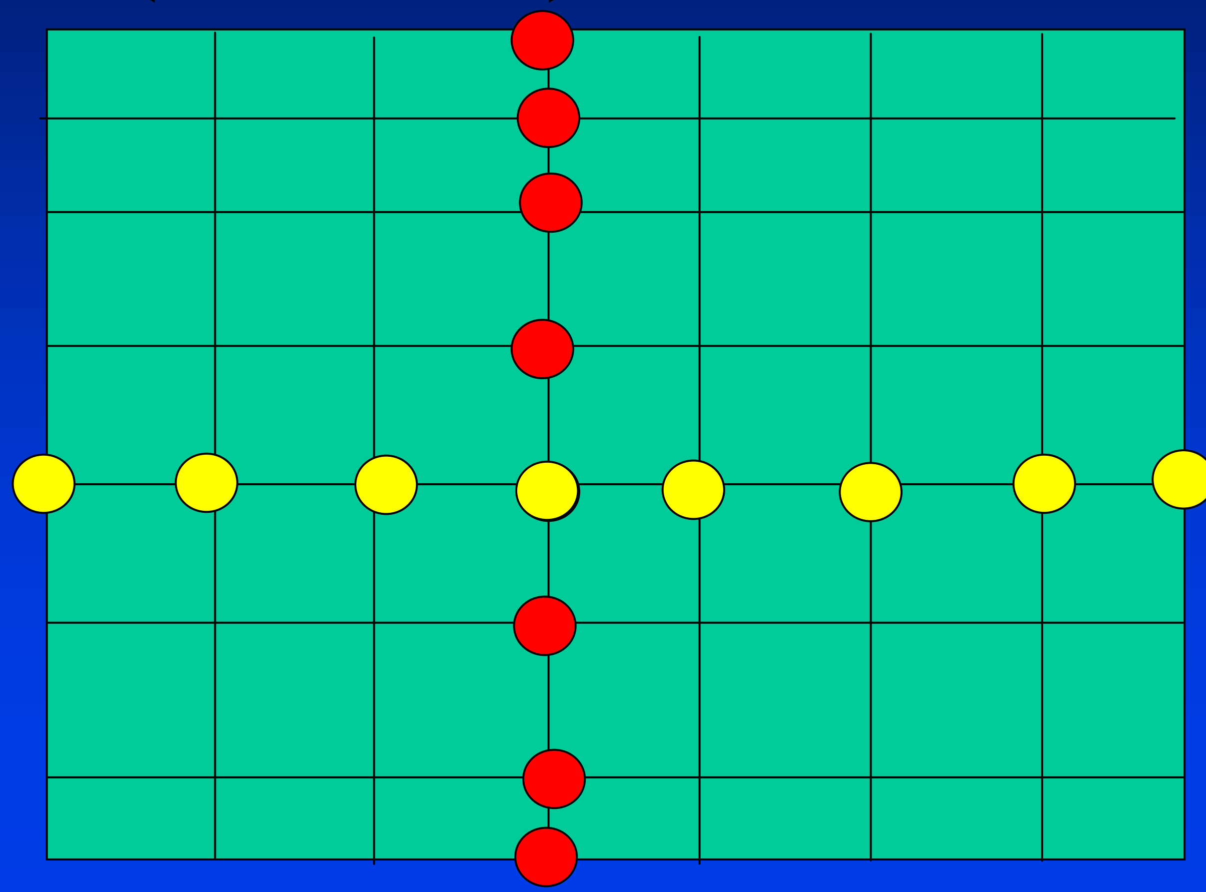


Surface (Patch) Rendering

- We use bicubic as an example
- The simplest (naïve): convert curved patches into primitives that we always know how to render
- From curved surfaces to polygon quadrilaterals (non-planar) and/or triangles (planar)
- Surface evaluation at grid points
- This is straight forward but inefficient, because it requires many times of evaluation of s(u,v)
- The total number is $3 \frac{1}{\delta u} \frac{1}{\delta v}$

Surface Rendering

- Parametric grids ([0,1]X[0,1]) as a set of rectangles



Surface Rendering

- Better approach: precomputation

$$\mathbf{s}(u,v) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} M \begin{bmatrix} v^3 \\ v^2 \\ v^2 \\ 1 \end{bmatrix}$$

- M is constant throughout the entire patch. The followings are the same along isoparametric lines

$$\begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$
$$\begin{bmatrix} v^3 & v^2 & v & 1 \end{bmatrix}$$

- Use one dimensional array to compute and store (evaluation only once)

Surface Rendering

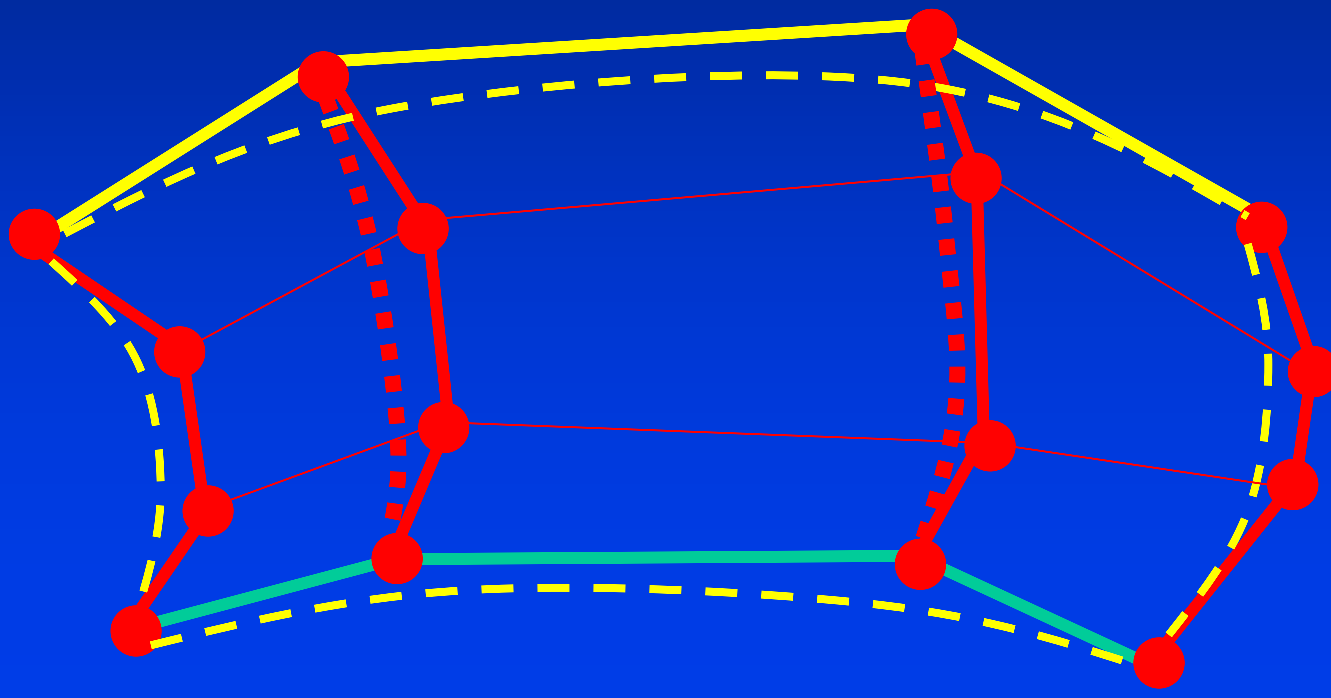
- How about many patches: the array is unchanged, its sampling rate is the same, this is more useful
- How about adaptive sampling based on curvature information!!!
- How to computer normal at any grid point (approximation)

$$\mathbf{s}_u(u,v) \times \mathbf{s}_v(u,v)$$

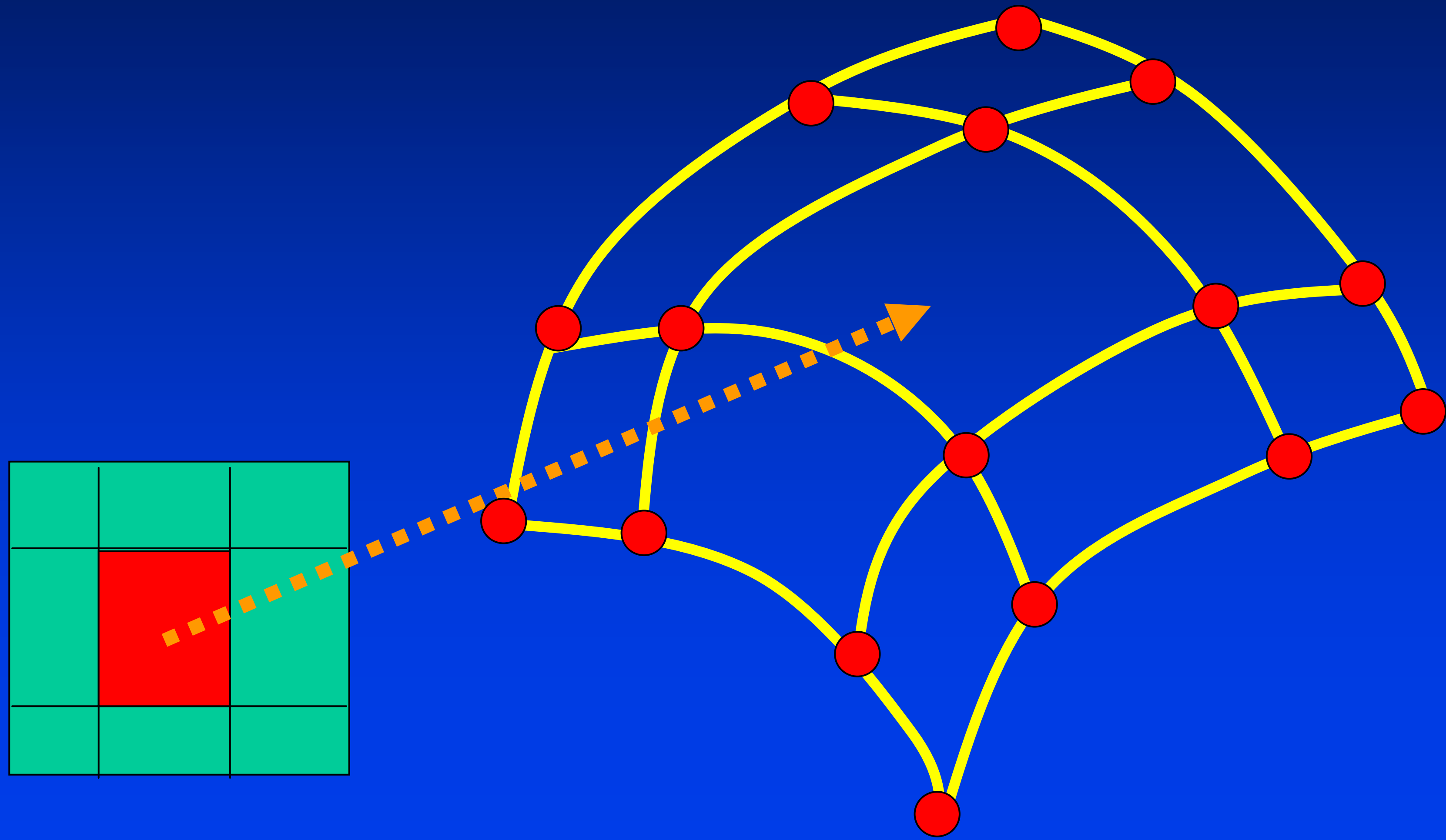
$$(\mathbf{s}(u+\delta u, v) - \mathbf{s}(u,v)) \times (\mathbf{s}(u, v+\delta v) - \mathbf{s}(u,v))$$

Regular Surface

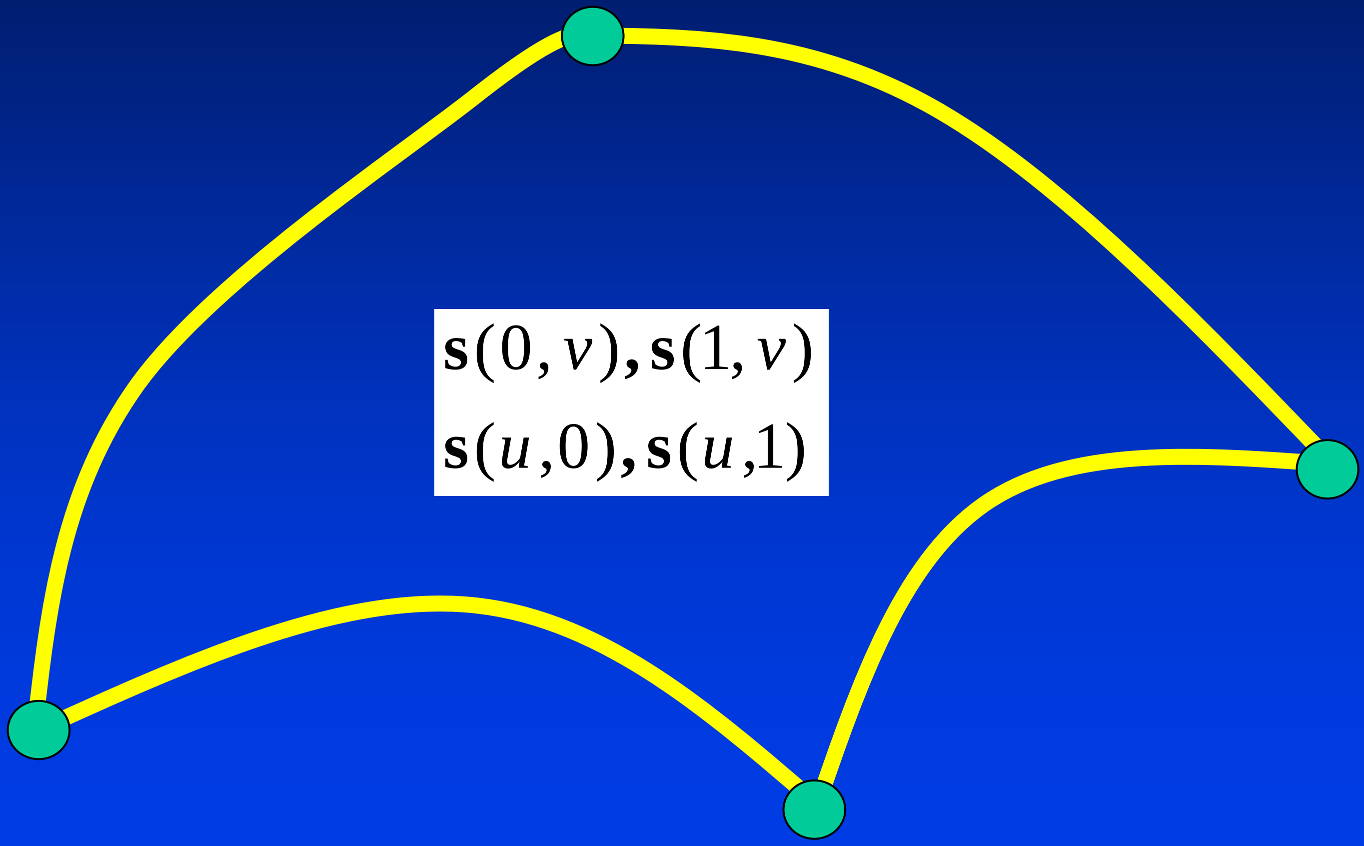
- Generated from a set of control points.



Curve Network

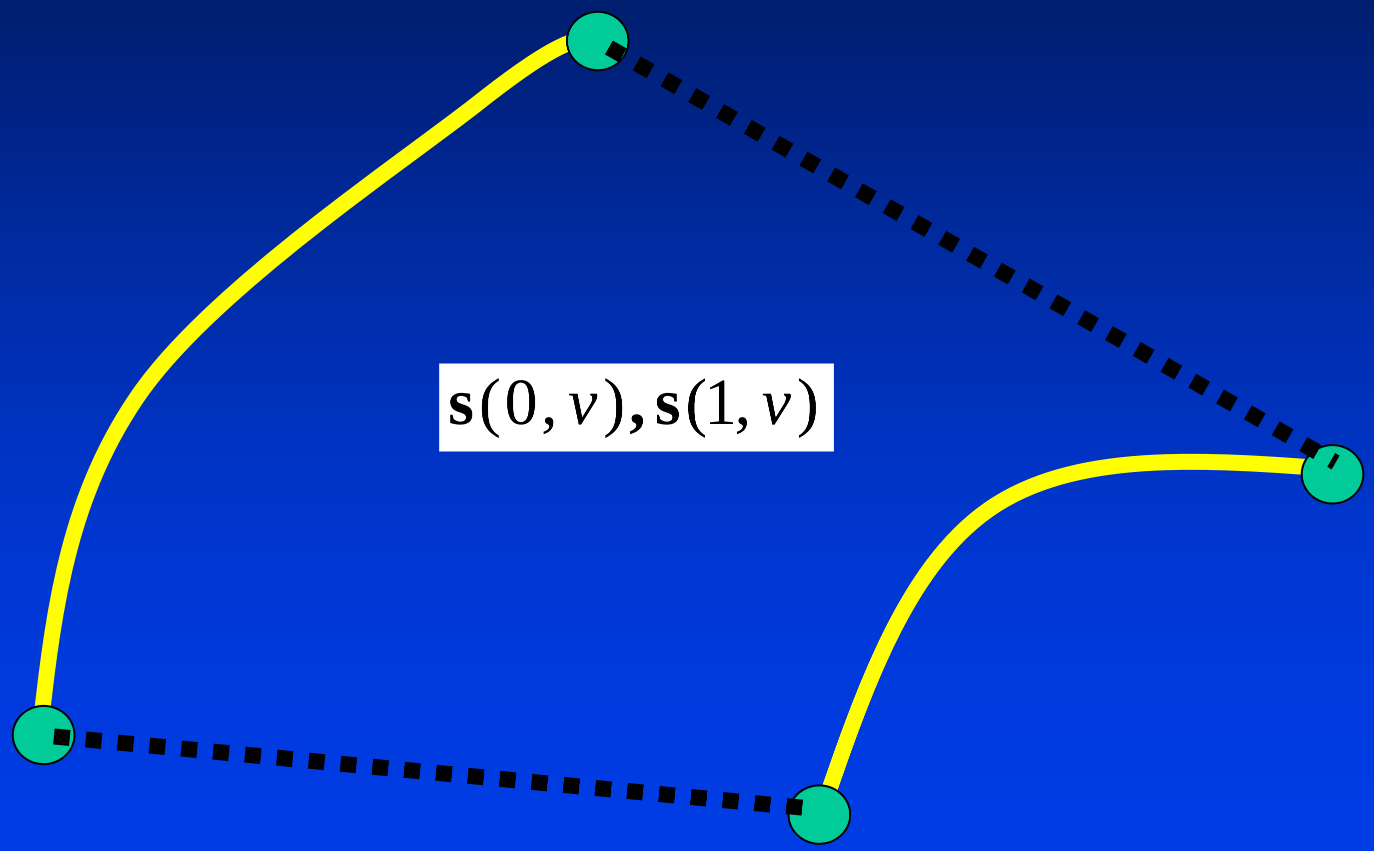


Coons Patch

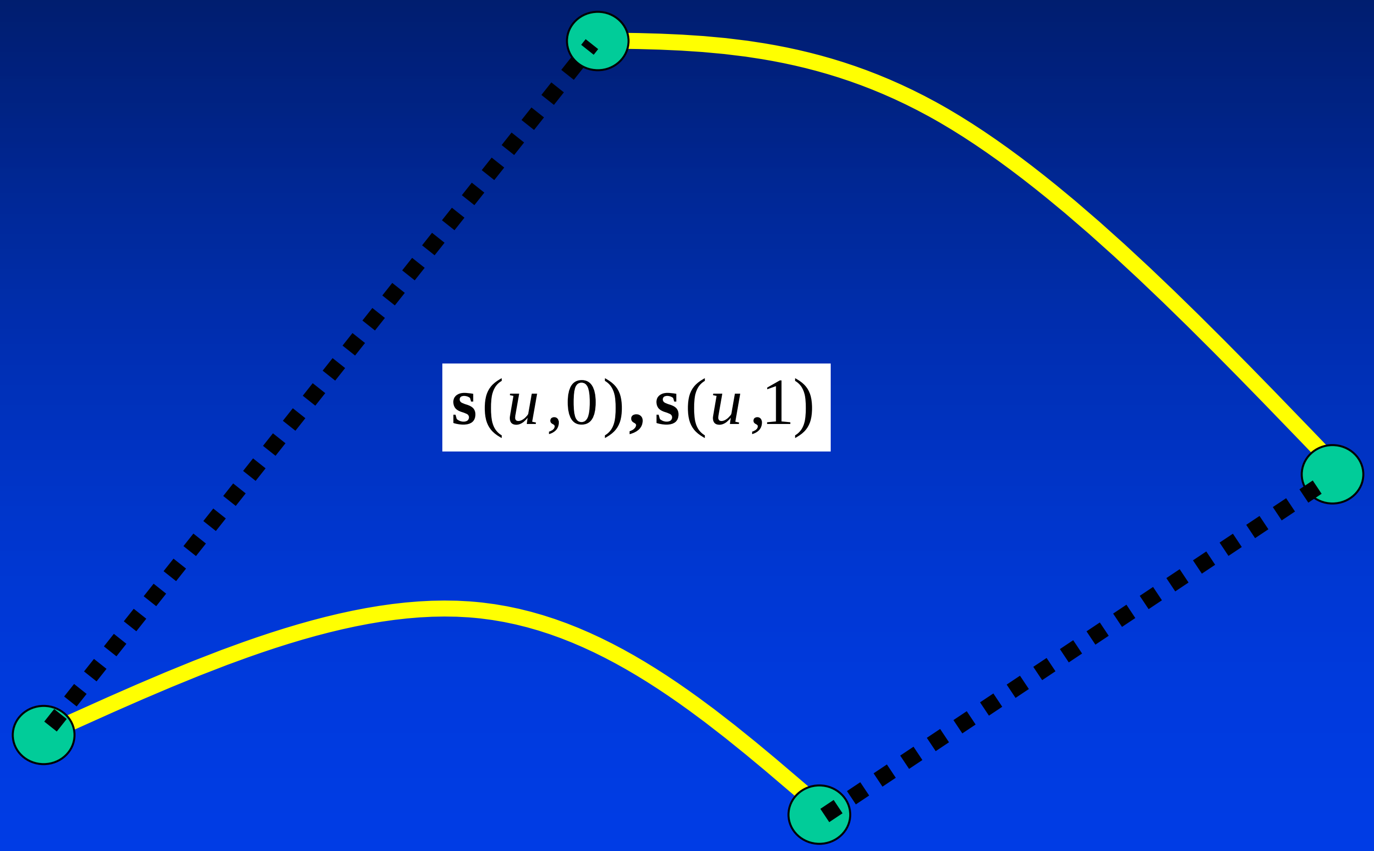


$$\mathbf{s}(0,v),\mathbf{s}(1,v)$$
$$\mathbf{s}(u,0),\mathbf{s}(u,1)$$

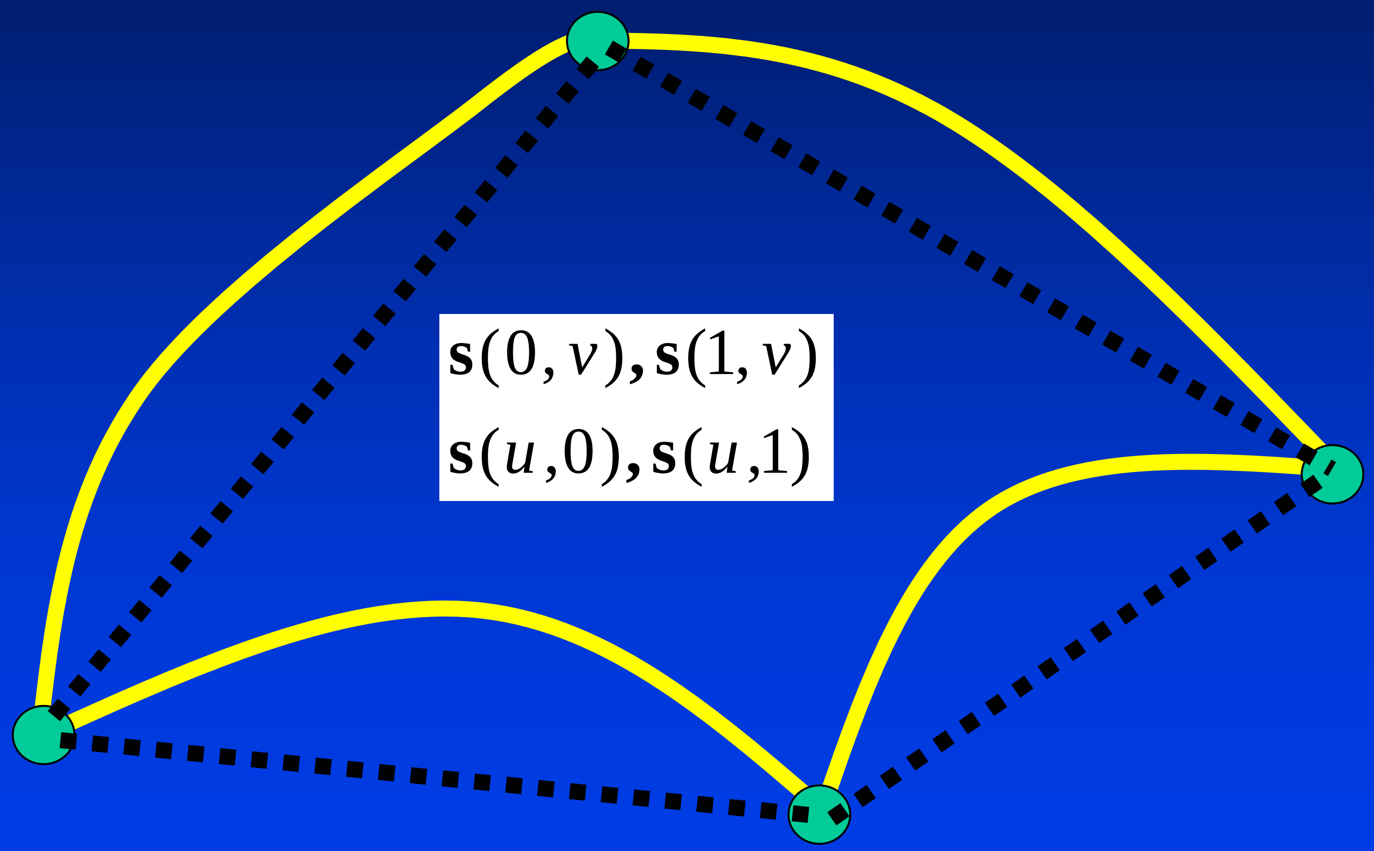
Coons Patch



Coons Patch



Coons Patch



Coons Patch

- Bilinearly blended Coons patch

$$(P)\mathbf{f} = (P_1 \oplus P_2)\mathbf{f} = (P_1 + P_2 - P_1 P_2)\mathbf{f}$$

$$(P_1)\mathbf{f} = \mathbf{f}(0,v)L_0^1(u) + \mathbf{f}(1,v)L_1^1(u)$$

$$(P_2)\mathbf{f} = \mathbf{f}(u,0)L_0^1(v) + \mathbf{f}(u,1)L_1^1(v)$$

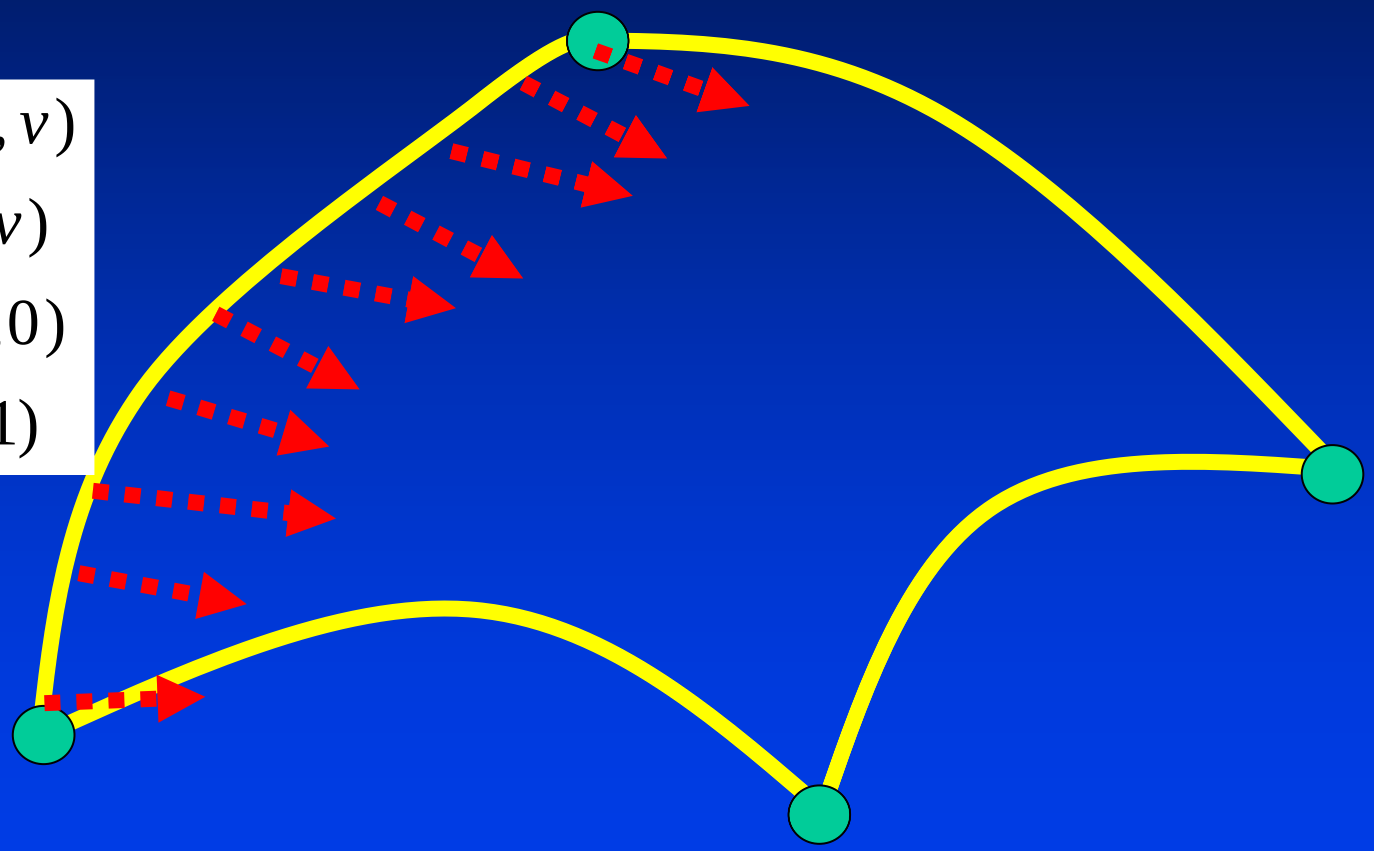
- Bicubically blended Coons patch

$$(P_1)\mathbf{f} = \mathbf{f}(0,v)H_0^3(u) + \mathbf{f}_u(0,v)H_1^3(u) + \mathbf{f}_u(1,v)H_2^3(u) + \mathbf{f}(1,v)H_3^3(u)$$

$$(P_2)\mathbf{f} = \mathbf{f}(u,0)H_0^3(v) + \mathbf{f}_v(u,0)H_1^3(v) + \mathbf{f}_v(u,1)H_2^3(v) + \mathbf{f}(u,1)H_3^3(v)$$

Coons Patch

$$\mathbf{s}(0,v),\mathbf{s}_u(0,v)$$
$$\mathbf{s}(1,v),\mathbf{s}_u(1,v)$$
$$\mathbf{s}(u,0),\mathbf{s}_v(u,0)$$
$$\mathbf{s}(u,1),\mathbf{s}_v(u,1)$$



Gordon Surfaces

- Generalization of Coons techniques
- A set of curves

$$\mathbf{f}(u_i, v),\ i = 0, \ldots\ldots, n$$
$$\mathbf{f}(u, v_j),\ j = 0, \ldots\ldots, m$$

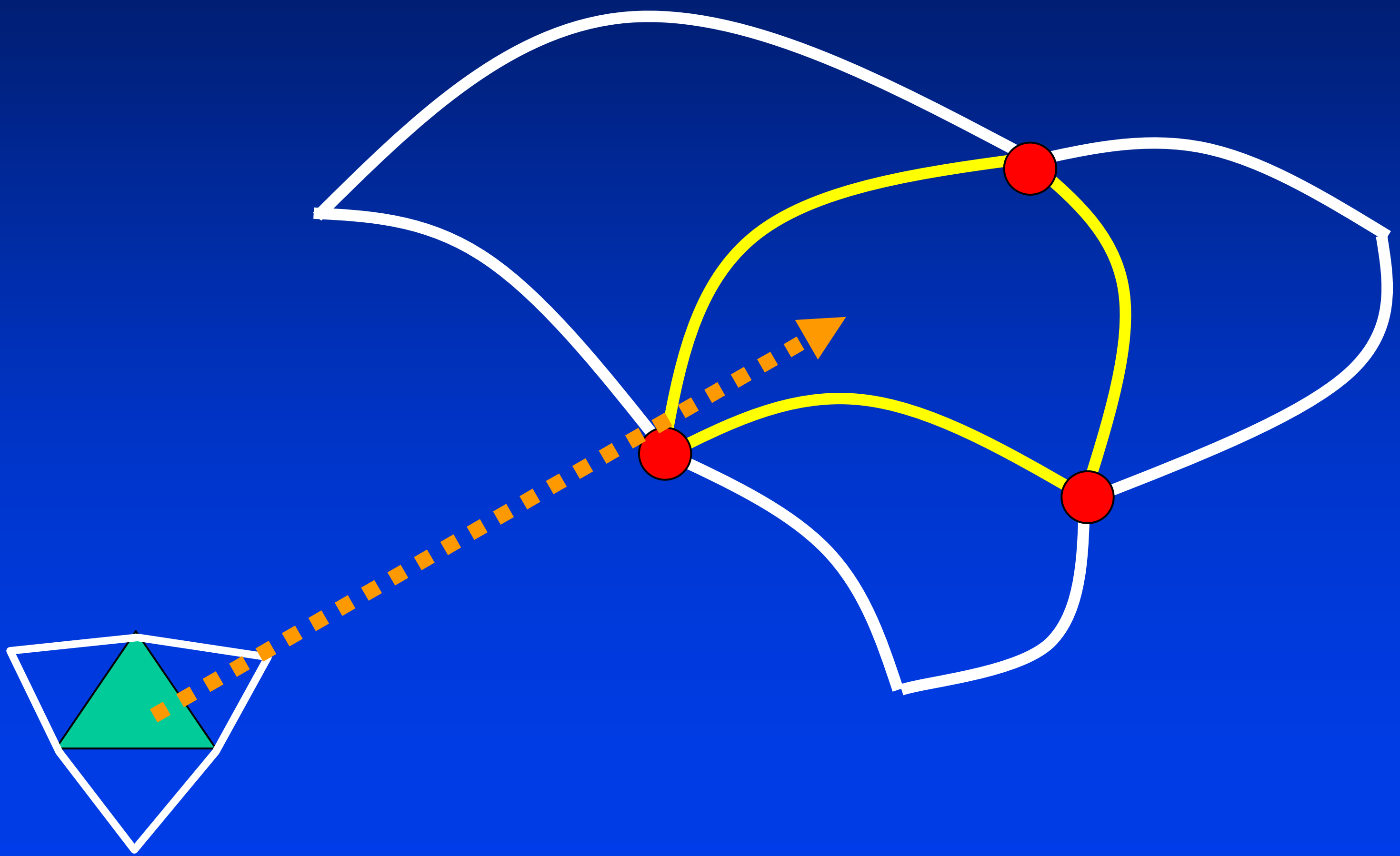
- Boolean sum using Lagrange polynomials

$$(P_1)\mathbf{f} = \sum_{i=0}^{n} \mathbf{f}(u_i, v) L_i^n(u)$$
$$(P_2)\mathbf{f} = \sum_{j=0}^{m} \mathbf{f}(u, v_j) L_j^m(v)$$
$$(P)\mathbf{f} = (P_1 \oplus P_2)\mathbf{f} = (P_1 + P_2 - P_1 P_2)\mathbf{f}$$

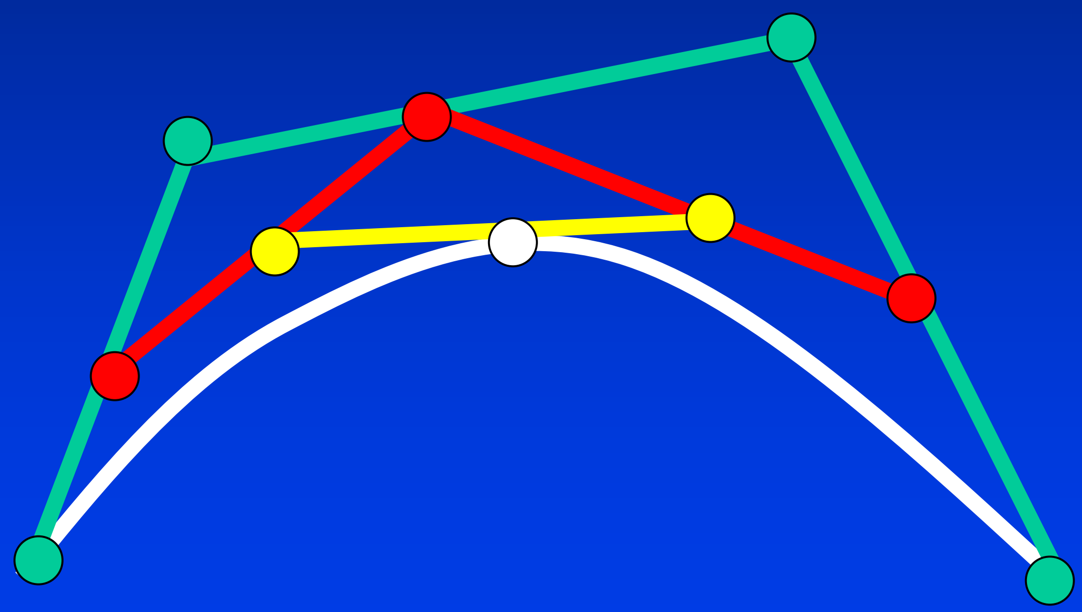
Transfinite Methods

- Bilinearly blended Coons patch
  – Interpolate four boundary curves
- Bicubically blended Coons patch
  – Interpolate curves and their derivatives
- Gordon surfaces
  – Interpolate a curve-network
- Triangular extension
  – Interpolate over triangles

Triangular Surfaces



Recursive Subdivision Algorithm



Curve Mathematics (Cubic)

- Bezier curve

$$\mathbf{c}(u) = \sum_{i=0}^{3} \mathbf{p}_i B_i^3(u)$$

- Control points and basis functions

$$B_0^3(u) = (1-u)^3$$

$$B_1^3(u) = 3u(1-u)^2$$

$$B_2^3(u) = 3u^2(1-u)$$

$$B_3^3(u) = u^3$$

- Image and properties of basis functions

Recursive Evaluation

- Recursive linear interpolation

$$(1-u) \quad (u)$$

$$\mathbf{p}_0^0 \quad \mathbf{p}_1^0 \quad \mathbf{p}_2^0 \quad \mathbf{p}_3^0$$

$$\mathbf{p}_0^1 \quad \mathbf{p}_1^1 \quad \mathbf{p}_2^1$$

$$\mathbf{p}_0^2 \quad \mathbf{p}_1^2$$

$$\mathbf{p}_0^3 = \mathbf{c}(u)$$

Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $\mathbf{c}(0) = \mathbf{p}_0, \mathbf{c}(1) = \mathbf{p}_n$
- Binomial expansion theorem

$$((1-u)+u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by control points

Properties

- Basis functions are non-negative
- The summation of all basis functions is unity
- End-point interpolation $\mathbf{c}(0) = \mathbf{p}_0, \mathbf{c}(1) = \mathbf{p}_n$
- Binomial expansion theorem

$$((1-u)+u)^n = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-i}$$

- Convex hull: the curve is bounded by the convex hull defined by control points

Derivatives

- Tangent vectors can easily evaluated at the end-points $\mathbf{c}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0); \mathbf{c}'(1) = (\mathbf{p}_3 - \mathbf{p}_2)$
- Second derivatives at end-points can also be easily computed:

$$\mathbf{c}^{(2)}(0) = 2 \times 3((\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)) = 6(\mathbf{p}_2 - 2\mathbf{p}_1 + \mathbf{p}_0)$$
$$\mathbf{c}^{(2)}(1) = 2 \times 3((\mathbf{p}_3 - \mathbf{p}_2) - (\mathbf{p}_2 - \mathbf{p}_1)) = 6(\mathbf{p}_3 - 2\mathbf{p}_2 + \mathbf{p}_1)$$

Derivative Curve

- The derivative of a cubic Bezier curve is a quadratic Bezier curve

$$\mathbf{c}'(u) = -3(1-u)^2\mathbf{p}_0 + 3\left((1-u)^2 - 2u(1-u)\right)\mathbf{p}_1 + 3\left(2u(1-u) - u^2\right)\mathbf{p}_2 + 3u^2\mathbf{p}_3 =$$

$$3(\mathbf{p}_1 - \mathbf{p}_0)(1-u)^2 + 3(\mathbf{p}_2 - \mathbf{p}_1)2u(1-u) + 3(\mathbf{p}_3 - \mathbf{p}_2)u^2$$

More Properties (Cubic)

- Two curve spans are obtained, and both of them are standard Bezier curves (through reparameterization)

$$\mathbf{c}(v),\ v \in [0,u]$$
$$\mathbf{c}(v),\ v \in [u,1]$$
$$\mathbf{c}_l(u),\ u \in [0,1]$$
$$\mathbf{c}_r(u),\ u \in [0,1]$$

- The control points for the left and the right are

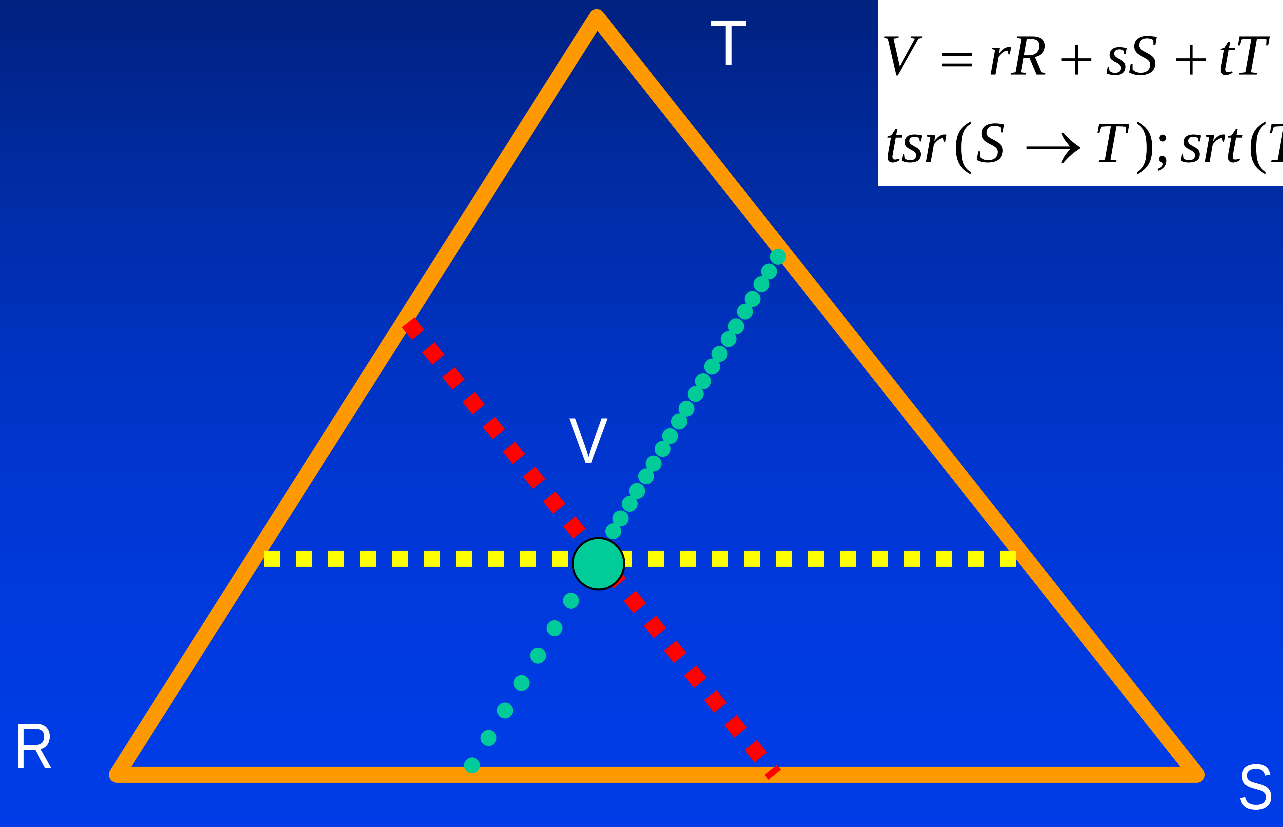
$$\mathbf{p}_0^0, \mathbf{p}_0^1, \mathbf{p}_0^2, \mathbf{p}_0^3$$
$$\mathbf{p}_0^3, \mathbf{p}_1^2, \mathbf{p}_2^1, \mathbf{p}_3^0$$

Barycentric Coordinates

$$r + s + t = 1$$

$$V = rR + sS + tT$$

$$tsr(S \to T); srt(T \to S); rts(S \to R)$$



Triangular Bezier Patch

- Triangular Bezier surface

$$\mathbf{s}(u,v) = \sum_{i,j,k>=0}^{i+j+k=n} \mathbf{p}_{i,j,k} B^{n}_{i,j,k}(r,s,t)$$

- Where r+s+t=1, and they are local barycentric coordinates
- Basis functions are Bernstein polynomials of degree n

$$B^{n}_{i,j,k}(r,s,t) = \frac{n!}{i!\,j!\,k!} r^{i} s^{j} t^{k}$$

Triangular Bezier Patch

- How many control points and basis functions:

$$\frac{1}{2}(n+1)(n+2)$$

- Partition of unity

$$\sum_{i,j,k>=0} B^n_{i,j,k}(r,s,t) = 1$$

- Positivity

$$B^n_{i,j,k}(r,s,t) >= 0; r,s,t \in [0,1]$$

Recursive Evaluation

$$\mathbf{p}_{i,j,k}^{0} = \mathbf{p}_{i,j,k}$$

$$\mathbf{p}_{i,j,k}^{l} = r\mathbf{p}_{i+1,j,k}^{l-1} + s\mathbf{p}_{i,j+1,k}^{l-1} + t\mathbf{p}_{i,j,k+1}^{l-1}; i+j+k = n-l, i, j, k >= 0$$

$$\mathbf{s}(u,v) = \mathbf{p}_{0,0,0}^{n}$$

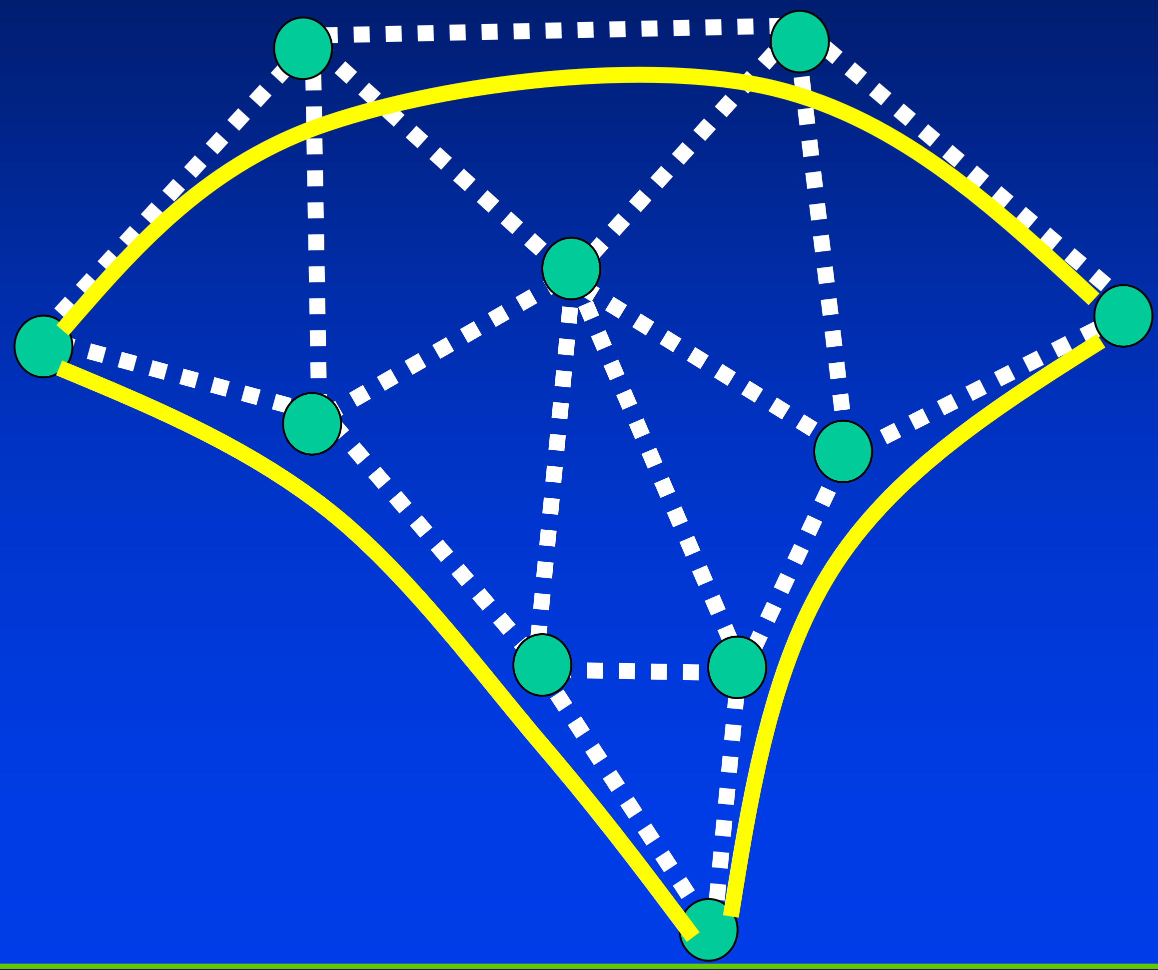
Properties

- Efficient algorithms
- Recursive evaluation
- Directional derivatives
- Degree elevation
- Subdivision
- Composite surfaces

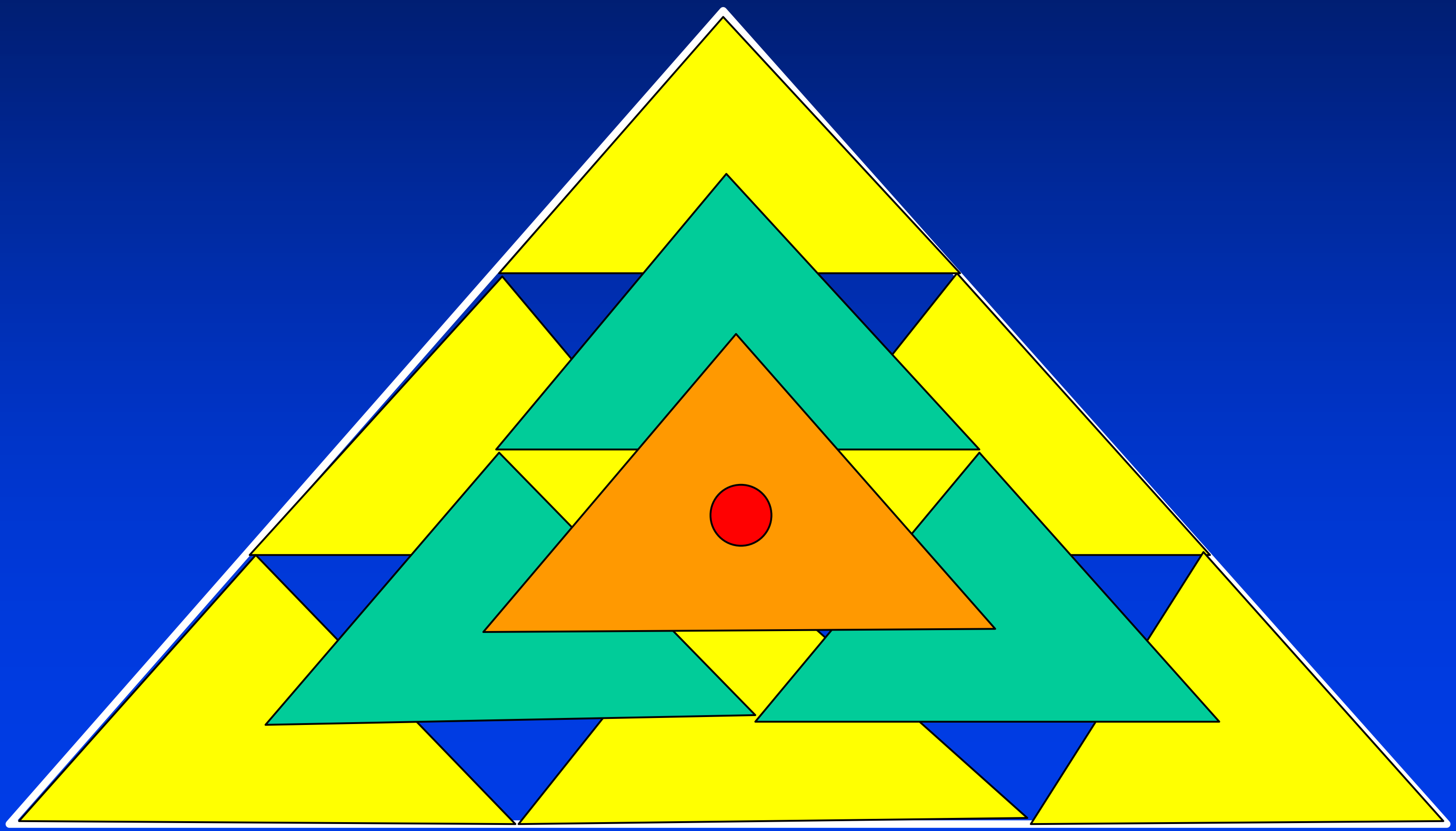
Research Issues

- Continuity across adjacent patches
- Integral computation
- Triangular splines over regular triangulation
- Transform triangular splines to a set of piecewise triangular Bezier patches
- Interpolation/approximation using triangular splines

Triangular Bezier Surface



Recursive Evaluation



Control points (Cubic)

$$\mathbf{p}_{0,3,0}$$

$$\mathbf{p}_{0,2,1} \quad \mathbf{p}_{1,2,0}$$

$$\mathbf{p}_{0,1,2} \quad \mathbf{p}_{1,1,1} \quad \mathbf{p}_{2,1,0}$$

$$\mathbf{p}_{0,0,3} \quad \mathbf{p}_{1,0,2} \quad \mathbf{p}_{2,0,1} \quad \mathbf{p}_{3,0,0}$$

Basis Functions (Cubic)

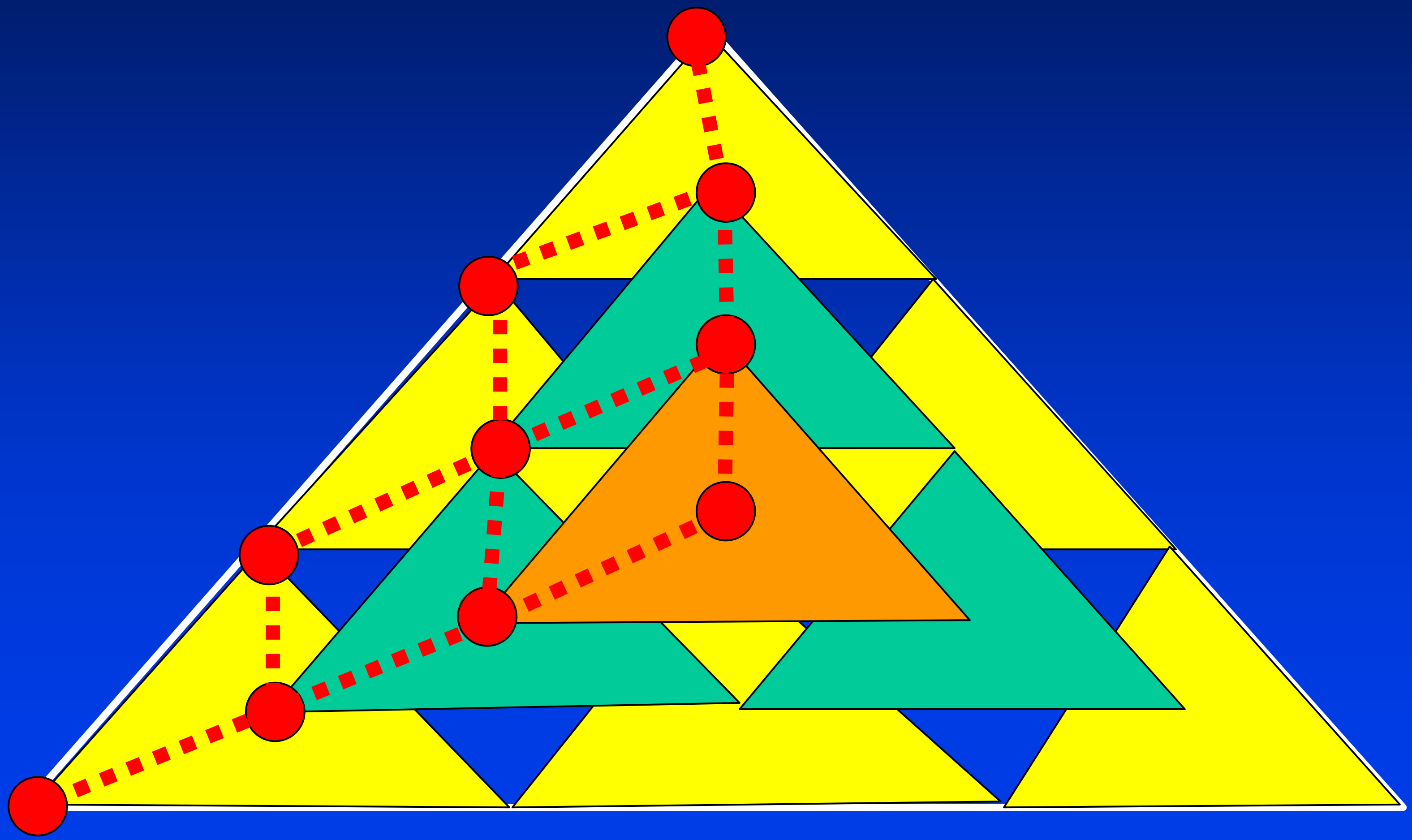
$$sss$$

$$3sst \quad 3rss$$

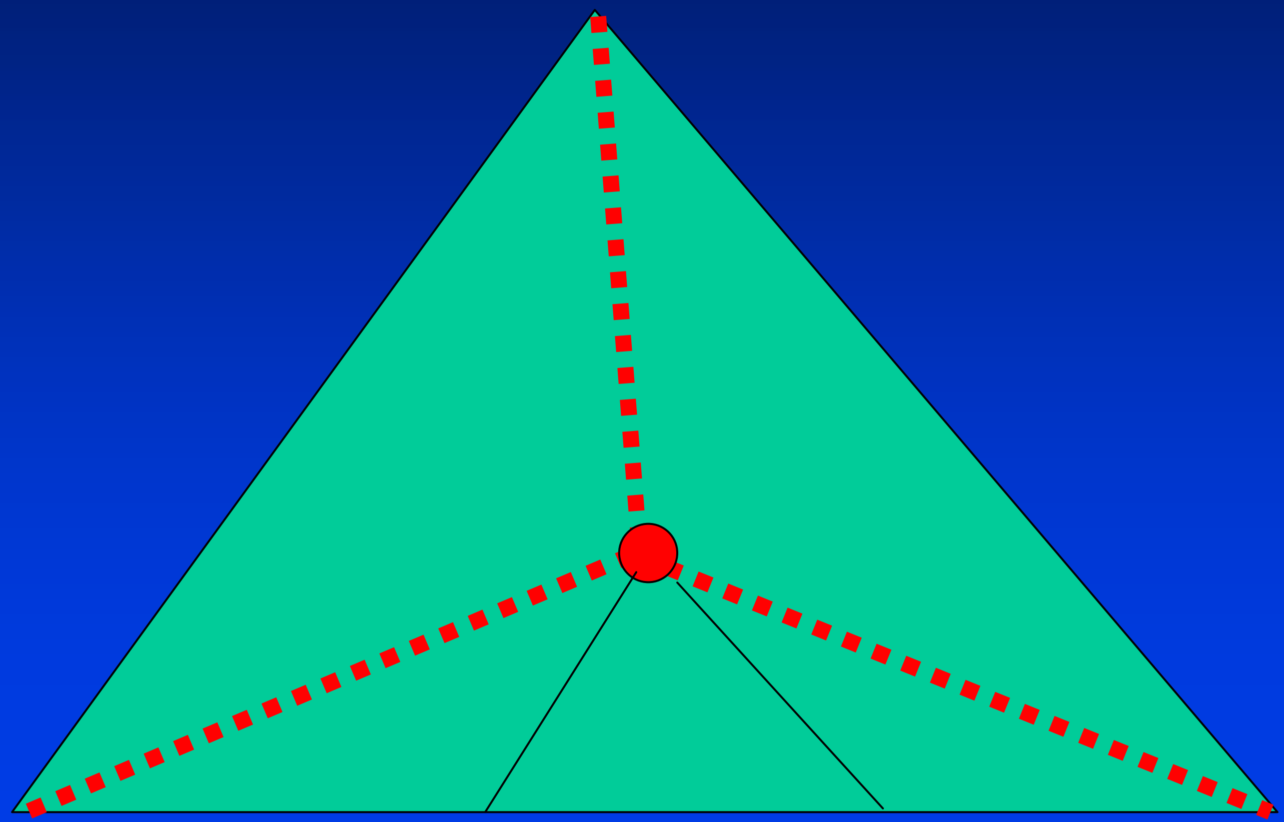
$$3stt \quad 6rst \quad 3rrs$$

$$ttt \quad 3rtt \quad 3rrt \quad rrr$$

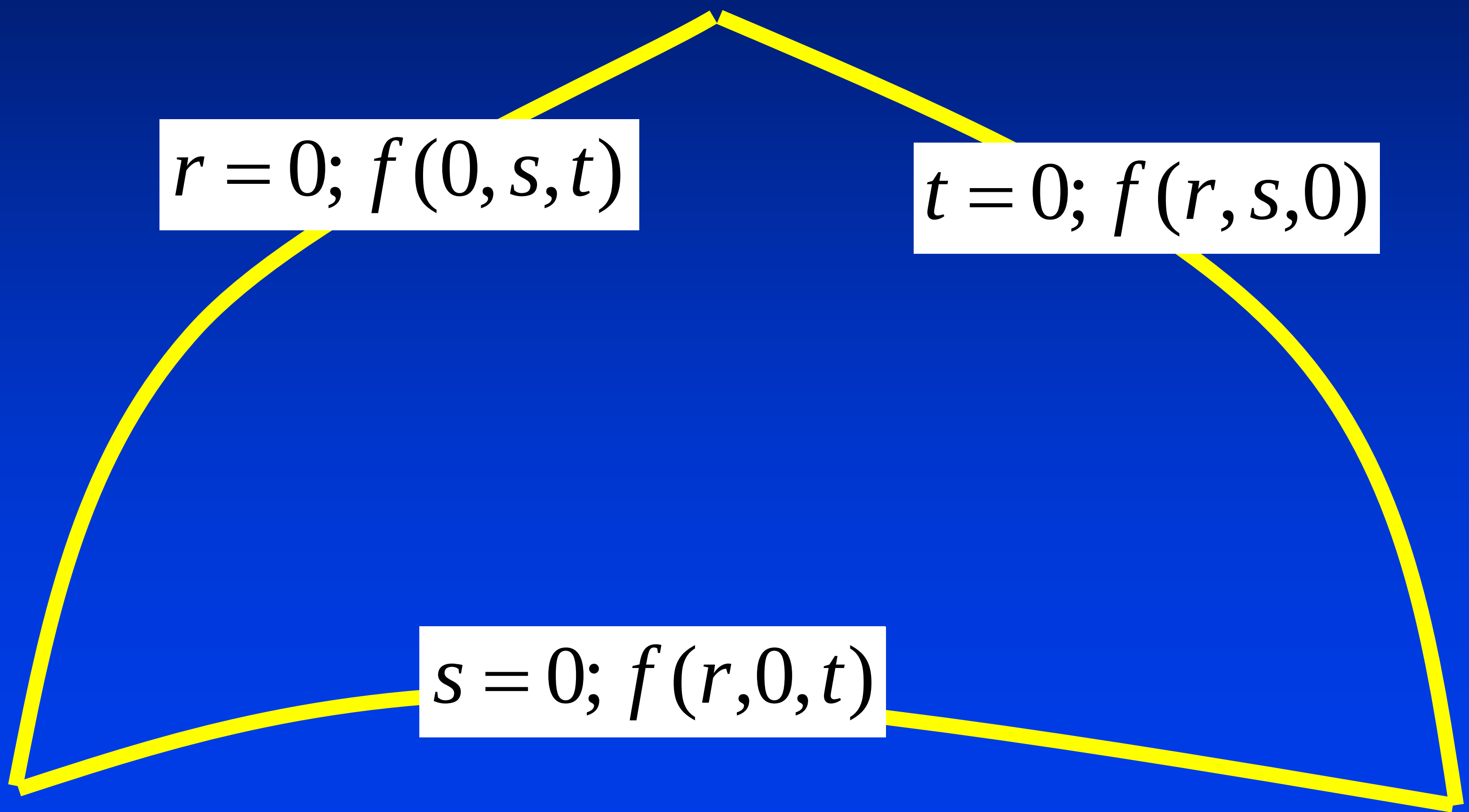
Triangular Patch Subdivision



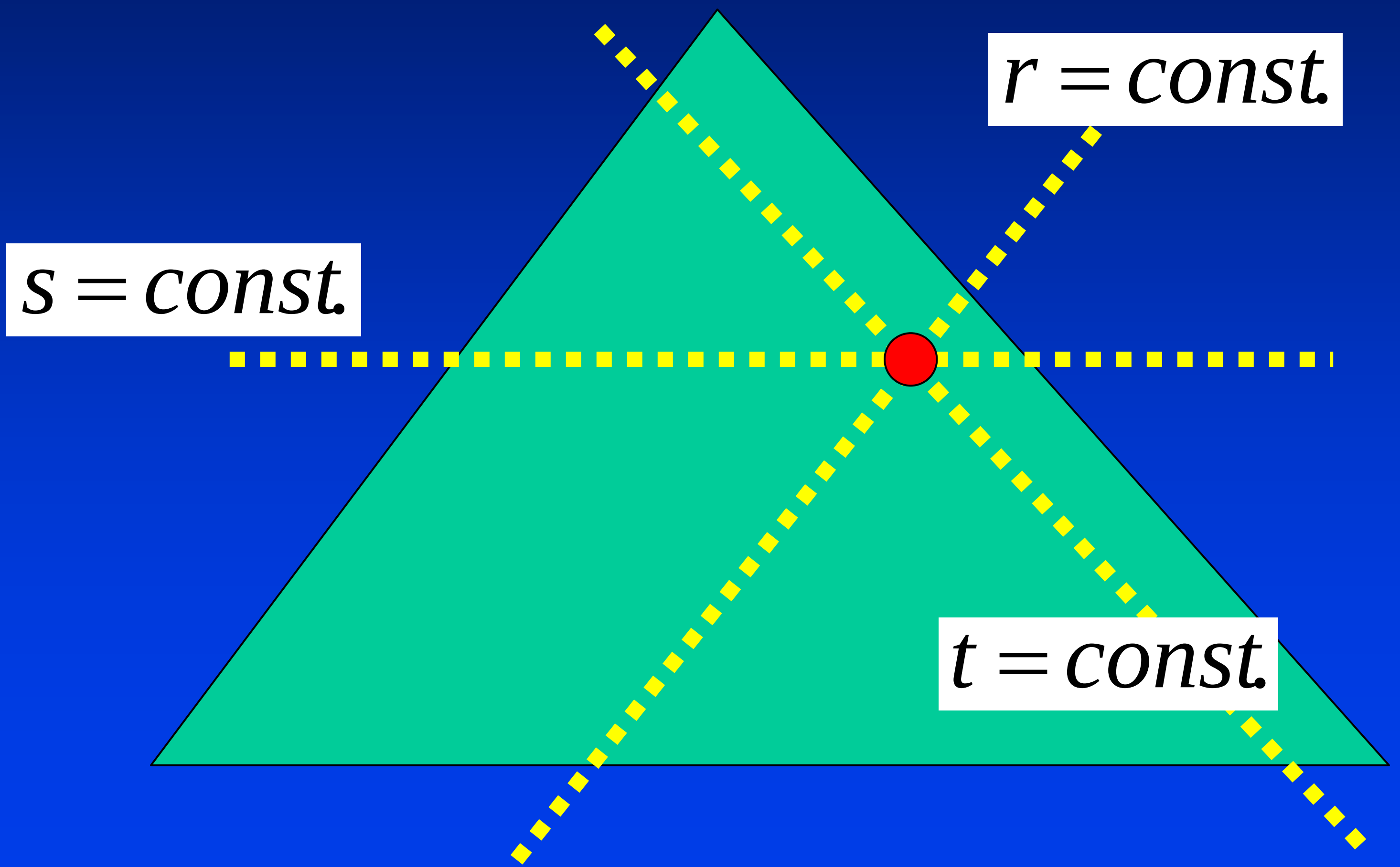
Triangular Domain



Triangular Coons-Gordon Surface



Triangular Coons-Gordon Surface



Triangular Interpolation

$$(P_1)\mathbf{f} = \mathbf{f}(r,0,t)L_0^1(\alpha) + \mathbf{f}(r,s,0)L_1^1(\alpha)$$

$$\alpha = \frac{s}{s+t}$$

$$(P_2)\mathbf{f} = \mathbf{f}(r,s,0)L_0^1(\beta) + \mathbf{f}(0,s,t)L_1^1(\beta)$$

$$\alpha = \frac{r}{r+t}$$

$$(P_3)\mathbf{f} = \mathbf{f}(0,s,t)L_0^1(\gamma) + \mathbf{f}(r,0,t)L_1^1(\gamma)$$

$$\alpha = \frac{r}{r+s}$$

Triangular Interpolation

- The Boolean sum of any two operators results the same!

$$(P_{12})\mathbf{f} = (P_1 \oplus P_2)\mathbf{f}$$
$$(P_{13})\mathbf{f} = (P_1 \oplus P_3)\mathbf{f}$$
$$(P_{23})\mathbf{f} = (P_2 \oplus P_3)\mathbf{f}$$

- Use cubic blending functions for C1 interpolation!

$$(Q_1)\mathbf{f} = \mathbf{f}(r,0,t)H_0^3(\alpha) + D_\alpha \mathbf{f}(r,0,t)H_1^3(\alpha) + D_\alpha \mathbf{f}(r,s,0)H_2^3(\alpha) + \mathbf{f}(r,s,0)H_3^3(\alpha)$$
$$(Q_2)\mathbf{f} = \ldots\ldots\ldots$$
$$(Q_3)\mathbf{f} = \ldots\ldots\ldots$$

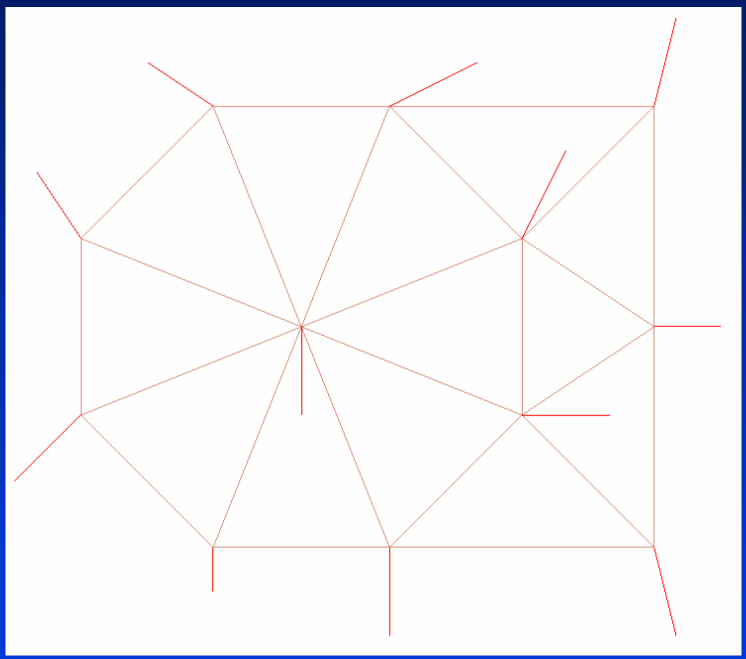
Gregory's Method

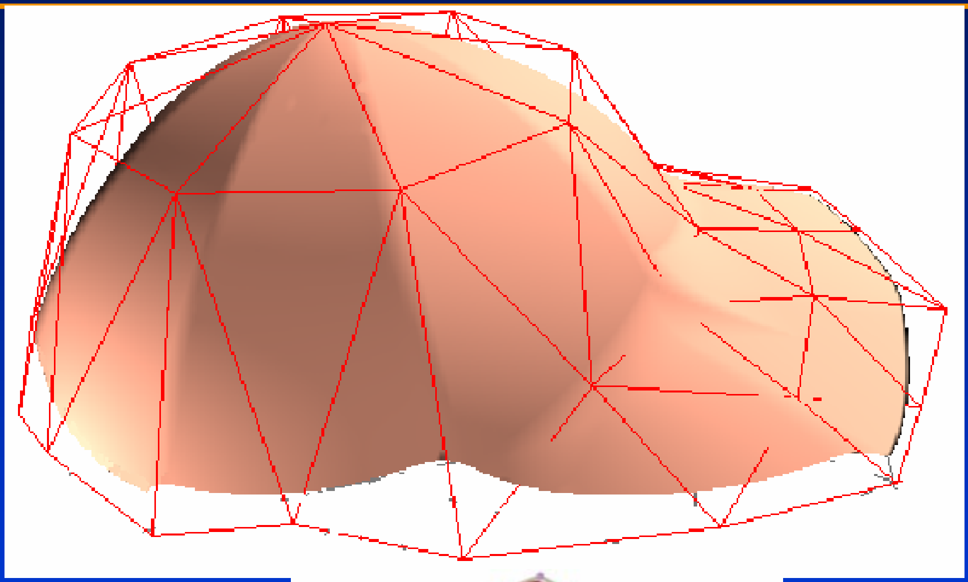
- Convex combination

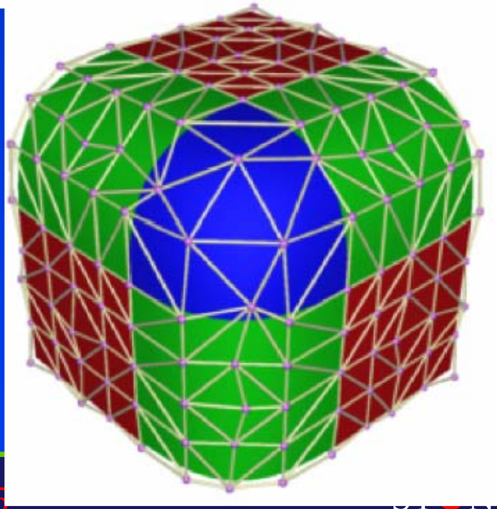
$$
\begin{aligned}
(T_1)\mathbf{f} &= \mathbf{f}(r,0,t) + \alpha D_\alpha \mathbf{f}(r,0,t) \\
(T_2)\mathbf{f} &= \dots\dots\dots \\
(T_3)\mathbf{f} &= \dots\dots\dots \\
(T_{12})\mathbf{f} &= (T_1 \oplus T_2)\mathbf{f} \\
(T_{13})\mathbf{f} &= (T_1 \oplus T_3)\mathbf{f} \\
(T_{23})\mathbf{f} &= (T_2 \oplus T_3)\mathbf{f} \\
(T)\mathbf{f} &= (a_1 T_{23} + a_2 T_{13} + a_3 T_{12})\mathbf{f} \\
a_1 &= \frac{s^2}{r^2 + s^2 + t^2} \\
a_2 &= \dots\dots\dots \\
a_3 &= \dots\dots\dots
\end{aligned}
$$

- Generalize to pentagonal patch!

Triangular B-splines



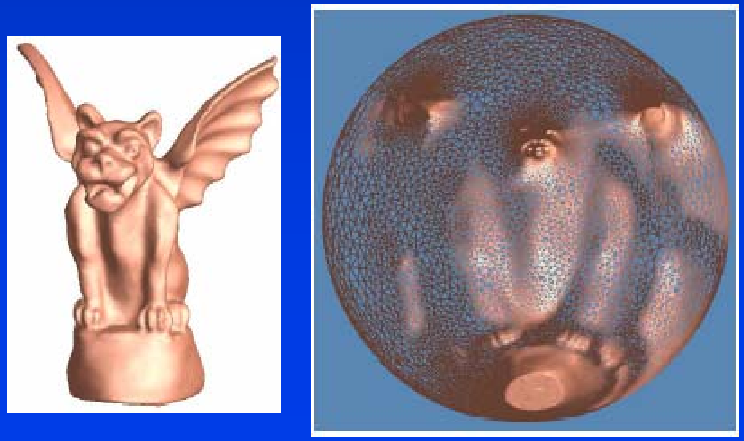


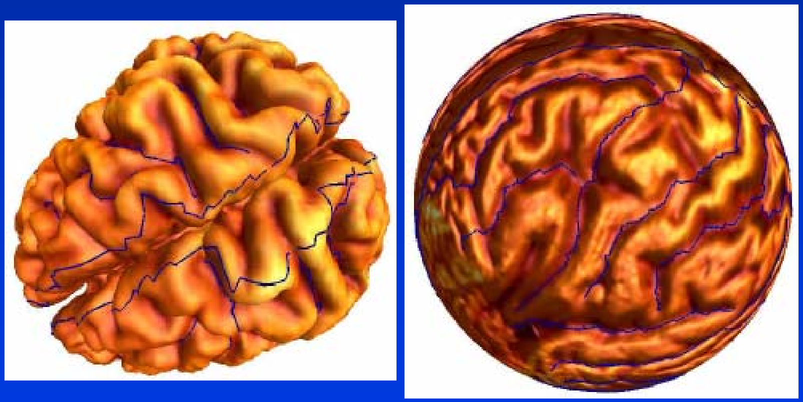


Surface Properties

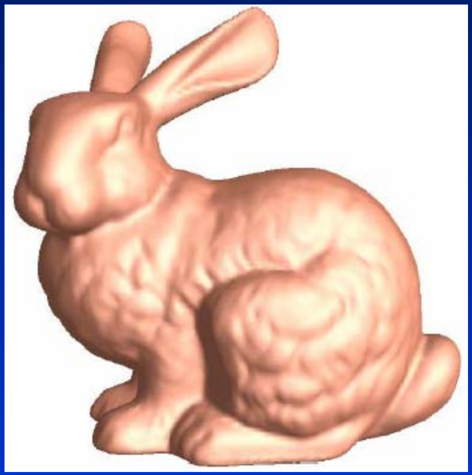
- Inherit from their curve generators
- More!
- Efficient algorithms
- Continuity across boundaries
- Interpolation and approximation tools

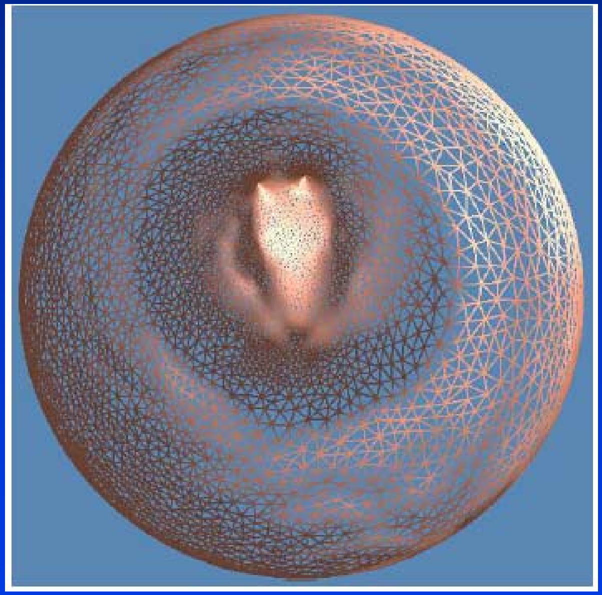
Spherical Parameterization



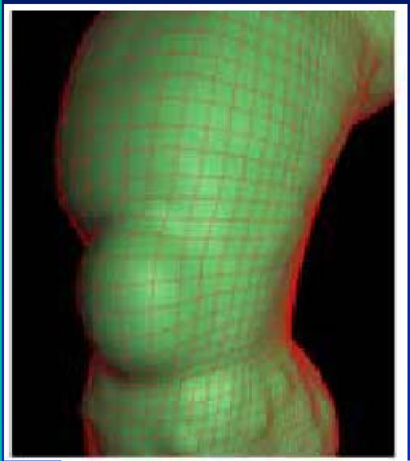


Spherical Parameterization





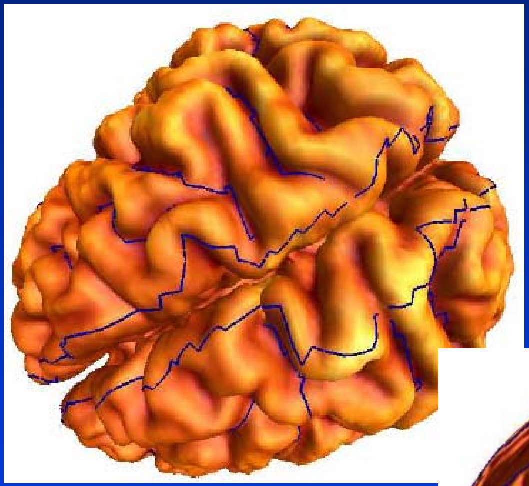
Possible Applications

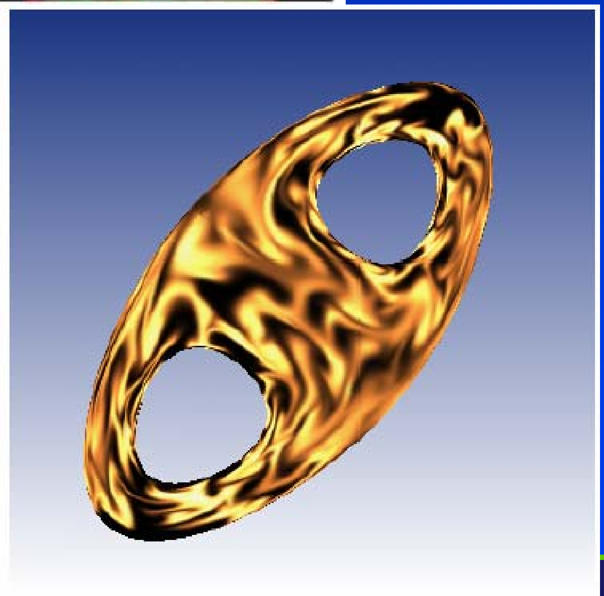


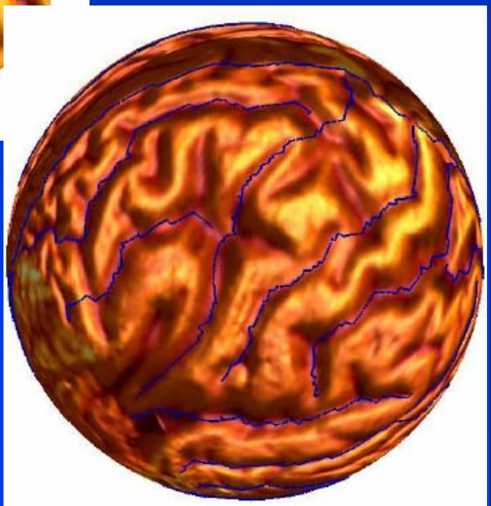
Smooth surface fitting

Shape classification

Medical registration

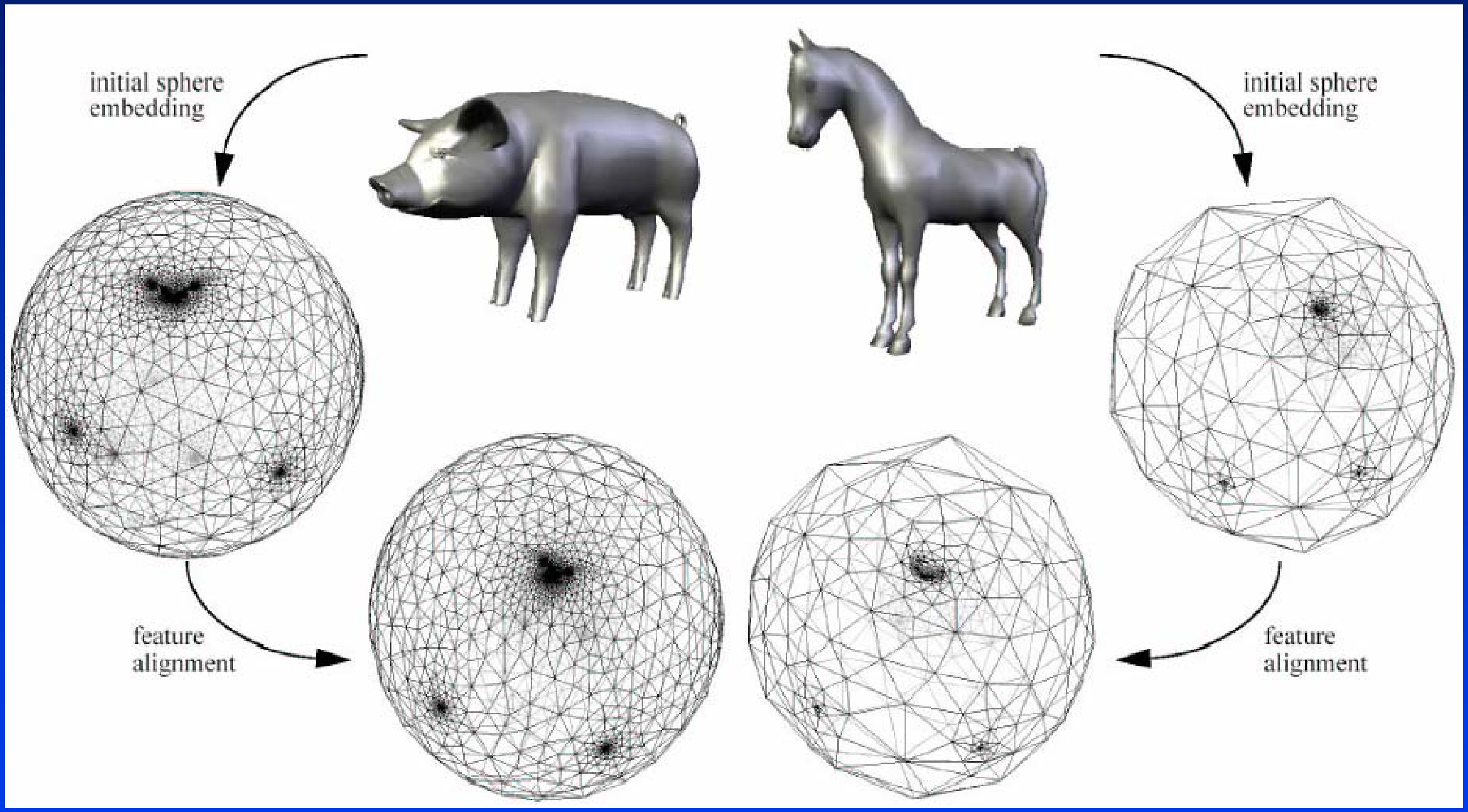




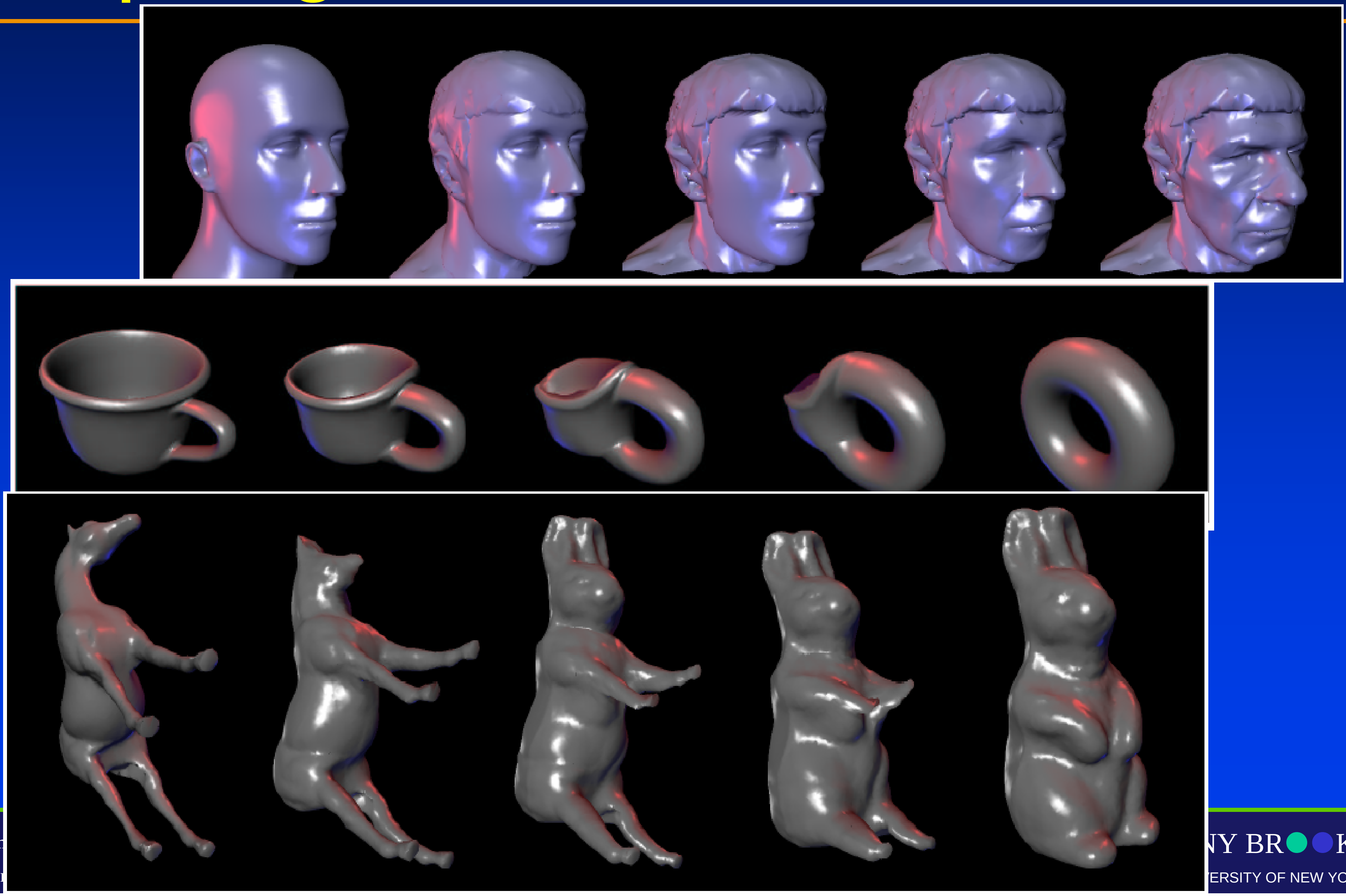


Solving PDEs on surfaces

Shape Morphing



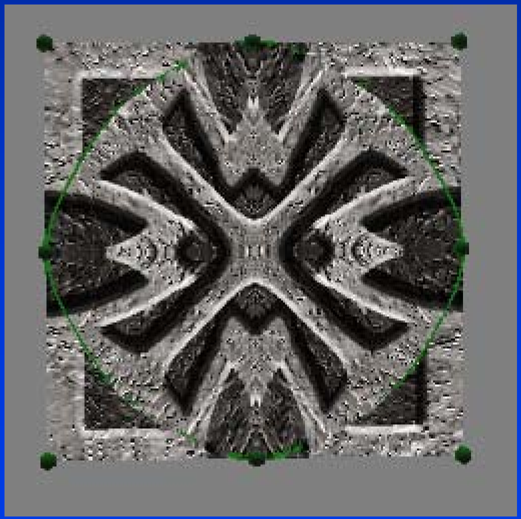
Morphing

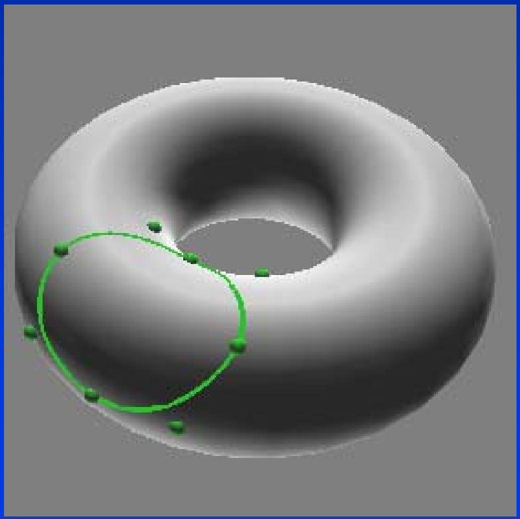


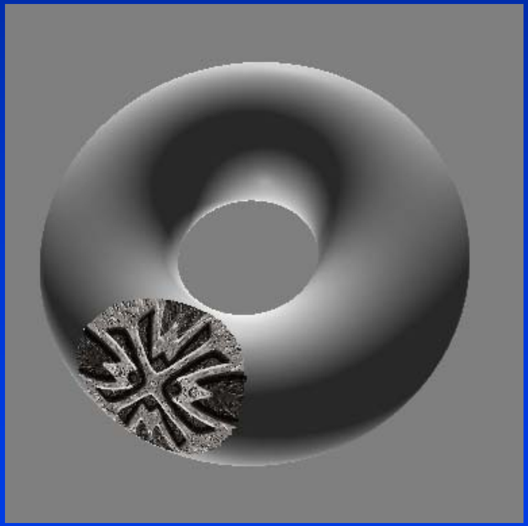
Multiresolution Mapping

- Multiresolution morphing

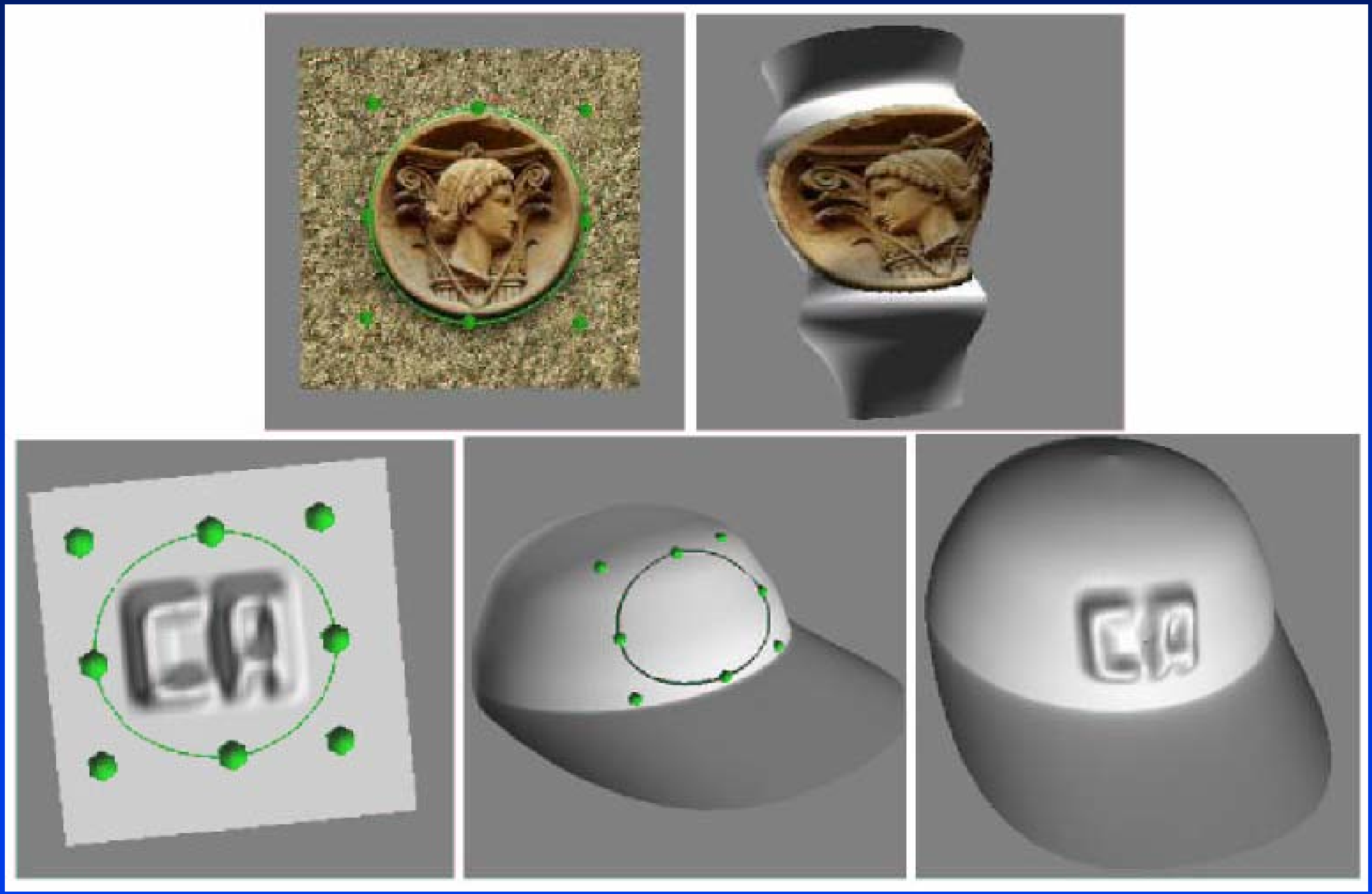
Feature Mapping



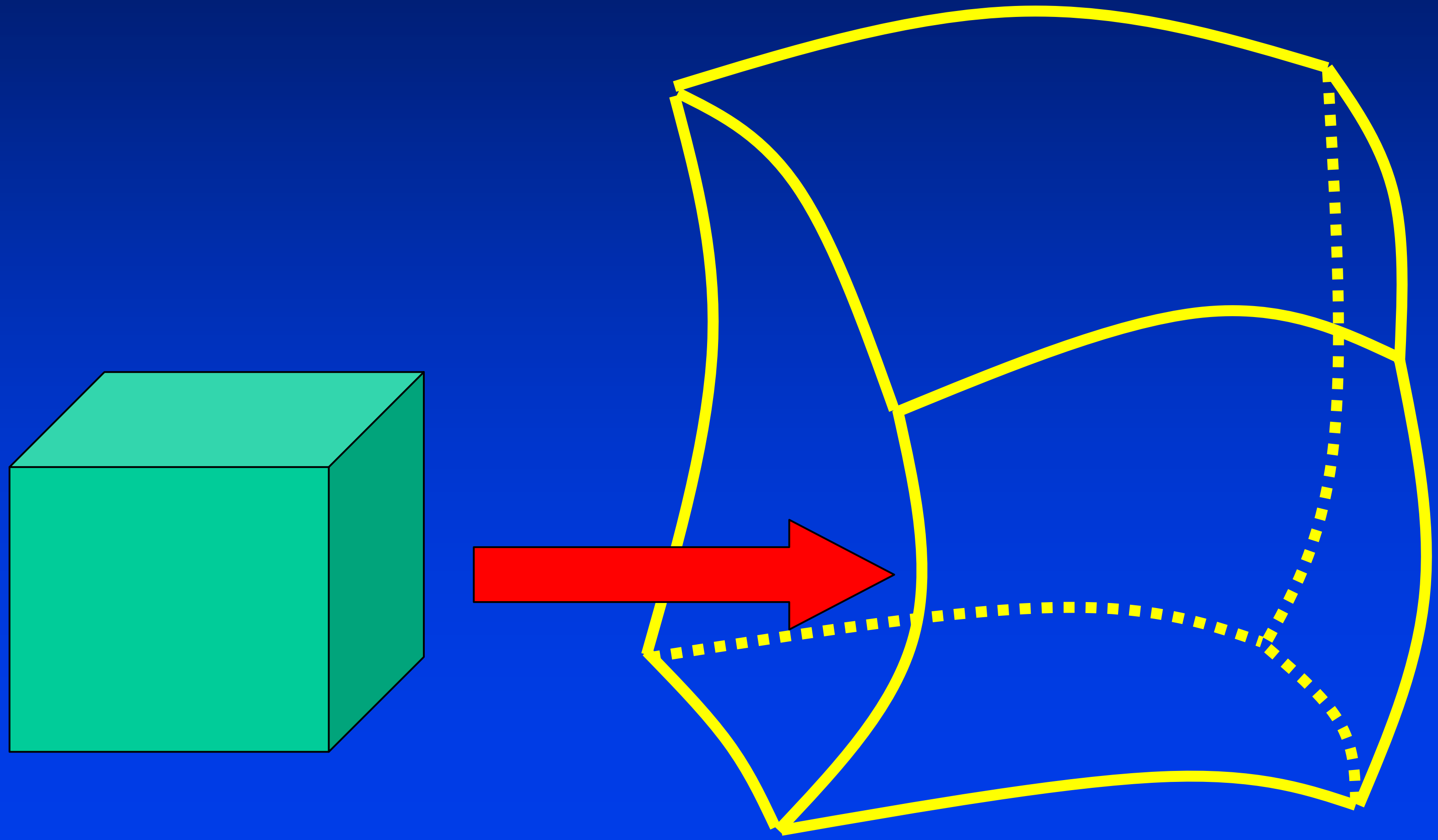




Texture Mapping



Solid



Parametric Solids

- Tricubic solid

$$\mathbf{p}(u,v,w) = \sum_{i=0}^{3}\sum_{j=0}^{3}\sum_{k=0}^{3}\mathbf{a}_{ijk}u^i v^j w^k$$

$$u,v,w \in [0,1]$$

- Bezier solid

$$\mathbf{p}(u,v,w) = \sum_i\sum_j\sum_k \mathbf{p}_{ijk}B_i(u)B_j(v)B_k(w)$$

- B-spline solid

$$\mathbf{p}(u,v,w) = \sum_i\sum_j\sum_k \mathbf{p}_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)$$

- NURBS solid

$$\mathbf{p}(u,v,w) = \frac{\sum_i\sum_j\sum_k \mathbf{p}_{ijk}q_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)}{\sum_i\sum_j\sum_k q_{ijk}B_{i,I}(u)B_{j,J}(v)B_{k,K}(w)}$$

Parametric Solids

- Tricubic Hermite solid
- In general

$$\mathbf{p}(u,v,w) = \begin{bmatrix} x(u,v,w) \\ y(u,v,w) \\ z(u,v,w) \end{bmatrix}$$
$$u,v,w \in [0,1]$$

- Also known as "hyperpatch"
- Parametric solids represent both exterior and interior
- Examples
  - A rectangular sold, a trilinear solid
- Boundary elements
  - 8 corner points, 12 curved edges, and 6 curved faces

Curves, Surfaces, and Solids

- Isoparametric curves for surfaces

$$\mathbf{s}(u,v), \mathbf{s}(u_i,v), \mathbf{s}(u,v_j)$$
$$u_i = const.;\ v_j = const.$$

- Isoparametric curves for solids

$$\mathbf{s}(u,v,w), \mathbf{s}(u_i,v_j,w), \mathbf{s}(u_i,v,w_k), \mathbf{s}(u,v_j,w_k)$$

- Isoparametric surfaces for solids

$$\mathbf{s}(u,v,w), \mathbf{s}(u_i,v,w), \mathbf{s}(u,v_j,w), \mathbf{s}(u,v,w_k)$$

Curves, Surfaces, and Solids

- Non-isoparametric curves for surfaces

$$\mathbf{s}(u,v)$$
$$\mathbf{c}(t) = \begin{bmatrix} u(t) \\ v(t) \end{bmatrix}$$
$$\mathbf{s}(u(t), v(t))$$

- Non-isoparametric curves for solids

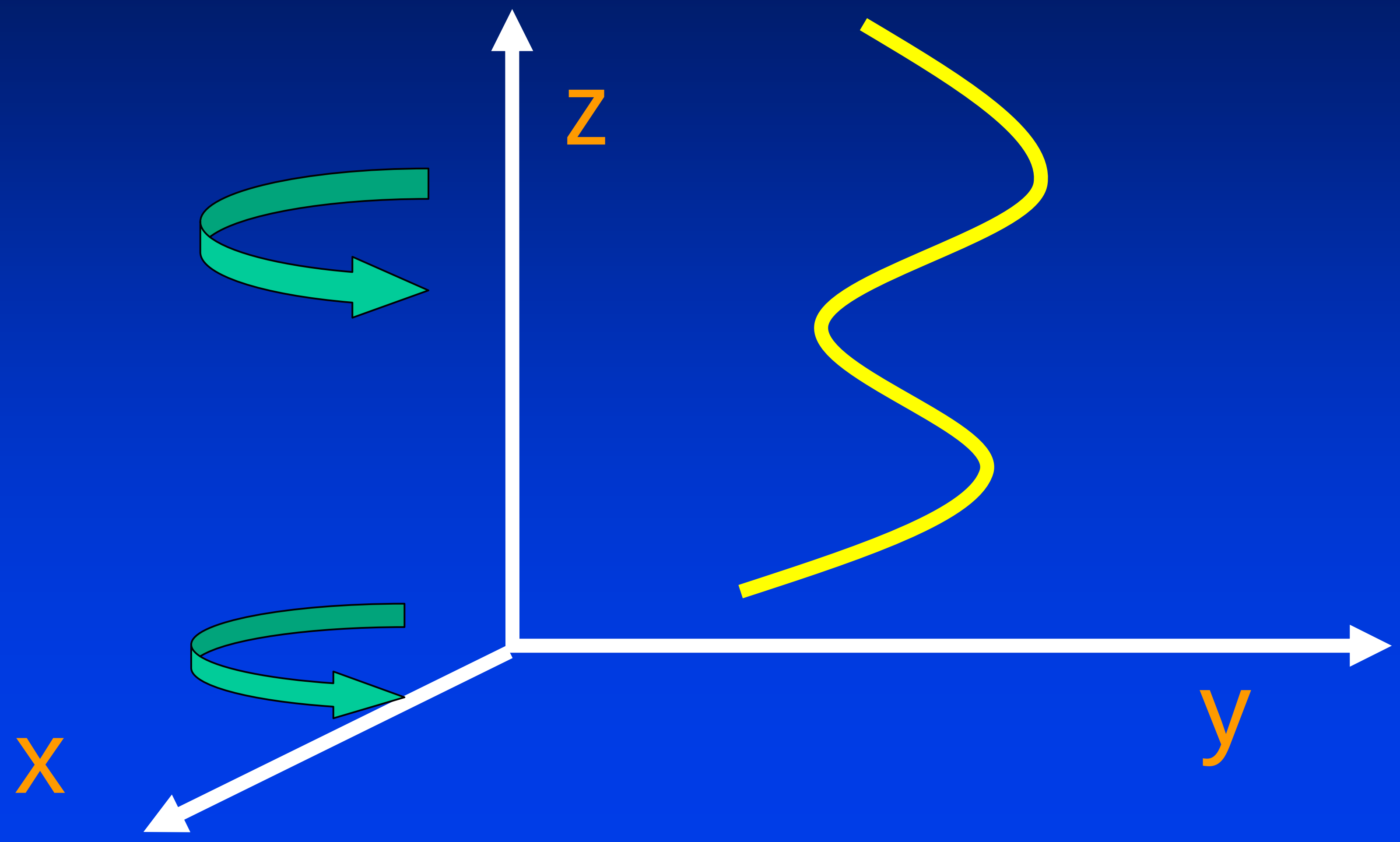
$$\mathbf{s}(u,v,w)$$
$$\mathbf{c}(t) = \begin{bmatrix} u(t) \\ v(t) \\ w(t) \end{bmatrix}$$
$$\mathbf{s}(u(t), v(t), w(t))$$

- Non-isoparametric surfaces for solids

$$\mathbf{s}(u,v,w) = \mathbf{s}(u(a,b), v(a,b), w(a,b))$$

CSE530-11

Surfaces of Revolution



Surfaces of Revolution

- Geometric construction
  - Specify a planar curve profile on y-z plane
  - Rotate this profile with respect to z-axis
- Procedure-based model
- What kinds of shape can we model?
- Review: three dimensional rotation w.r.t. z-axis

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

Surfaces of Revolution

- Mathematics: surfaces of revolution

$$\mathbf{c}(u) = \begin{bmatrix} 0 \\ y(u) \\ z(u) \end{bmatrix}$$

$$\mathbf{s}(u,v) = \begin{bmatrix} -y(u)\sin(v) \\ y(u)\cos(v) \\ z(u) \end{bmatrix}$$

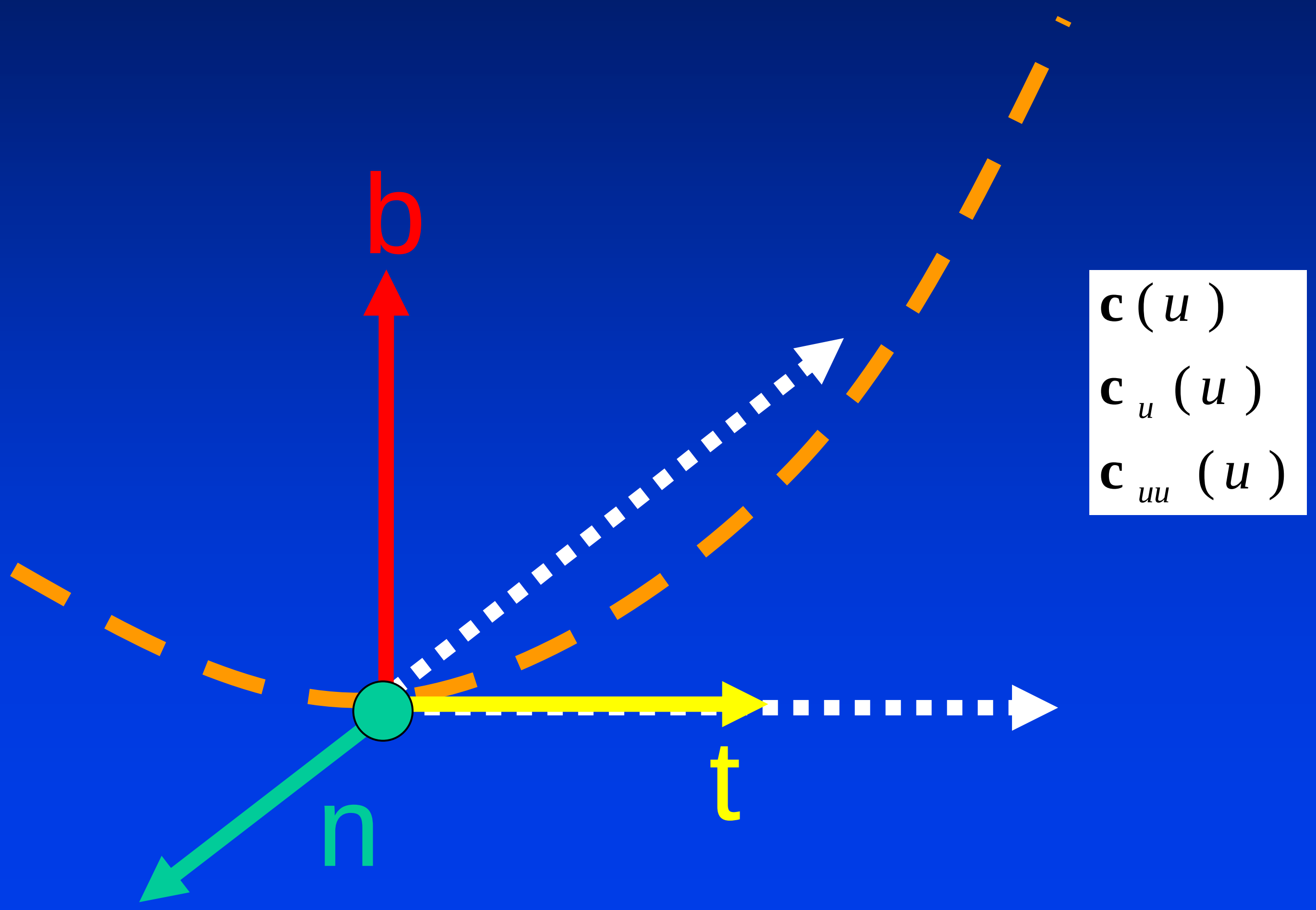
Frenet Frames

- Motivation: attach a smoothly-varying coordinate system to any location of a curve
- Three independent direction vectors for a 3D coordinate system: (1) tangent; (2) bi-normal; (3) normal

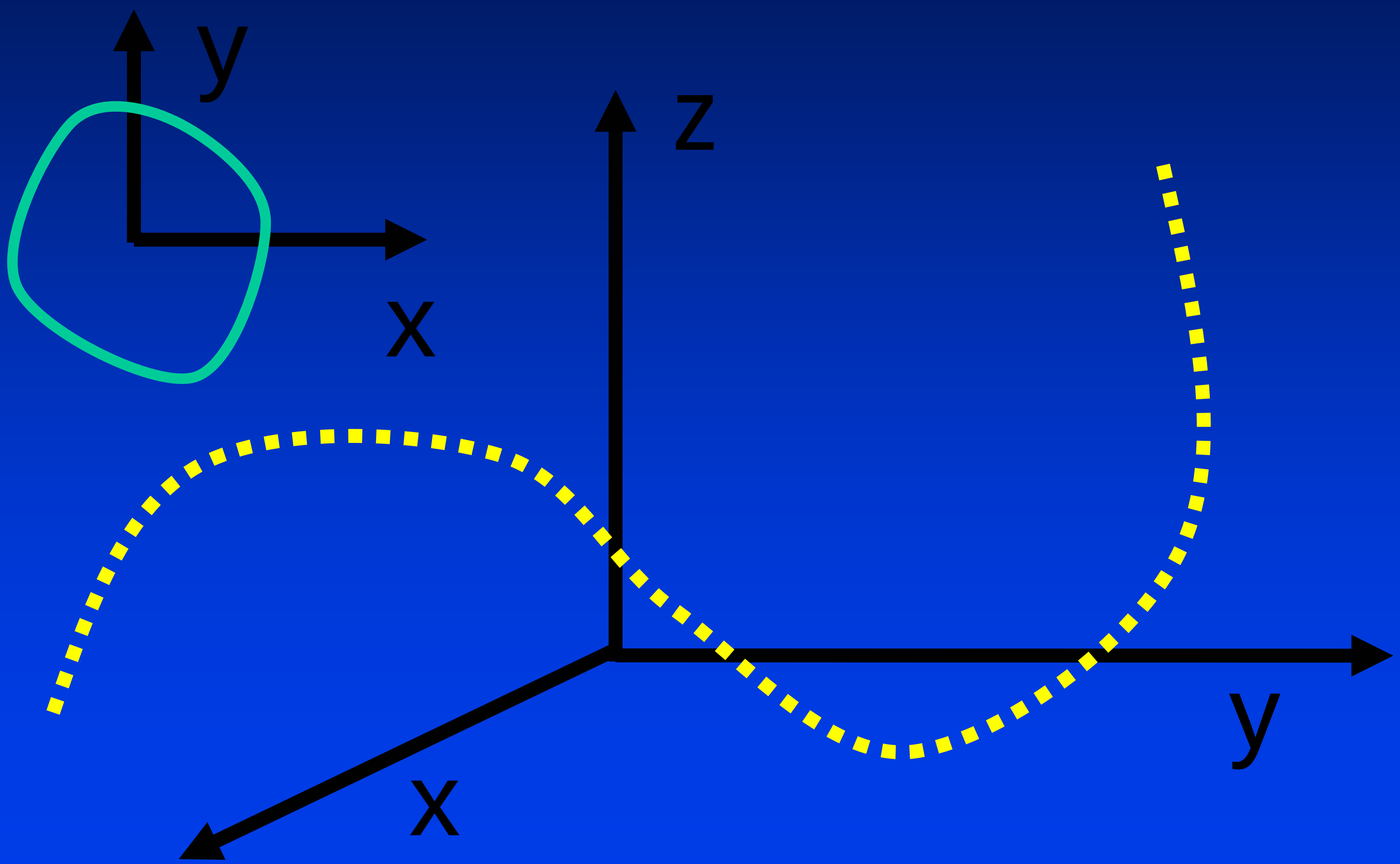
$$
\begin{aligned}
\mathbf{t}(u) &= normalize\ (\mathbf{c}_u(u)) \\
\mathbf{b}(u) &= normalize\ (\mathbf{c}_u(u) \times \mathbf{c}_{uu}(u)) \\
\mathbf{n}(u) &= normalize\ (\mathbf{b}(u) \times \mathbf{t}(u))
\end{aligned}
$$

- Frenet coordinate system (frame) (t,b,n) varies smoothly, as we move along the curve c(u)

Frenet Coordinate System



Sweeping Surface



General Sweeping Surfaces

- Surface of revolution is a special case of a sweeping surface
- Idea: a profile curve and a trajectory curve

$$\mathbf{c}_1(u)$$
$$\mathbf{c}_2(v)$$

- Move a profile curve along a trajectory curve to generate a sweeping surface
- Question: how to orient the profile curve as it moves along the trajectory curve?
- Answer: various options

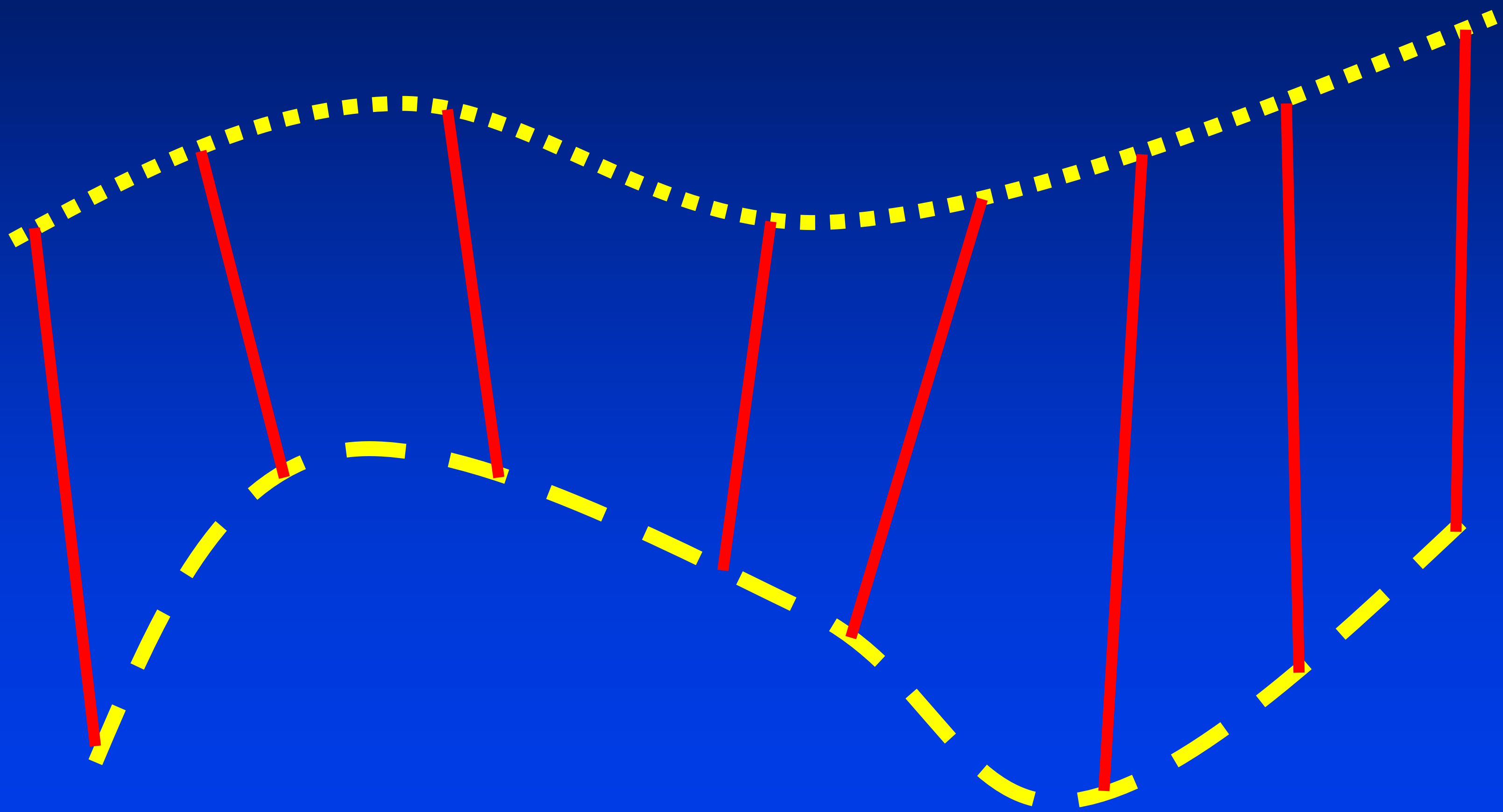
General Sweeping Surfaces

- Fixed orientation, simple translation of the coordinate system of the profile curve along the trajectory curve
- Rotation: if the trajectory curve is a circle
- Move using the “Frenet Frame” of the trajectory curve, smoothly varying orientation
- Example: surface of revolution
- Differential geometry fundamentals: Frenet frame

Frenet Swept Surfaces

- Orient the profile Curve (C1(u)) using the Frenet frame of C2(v)
  - Put C1(u) on the normal plane (n,b)
  - Place the original of C1(u) on C2(v)
  - Align the x-axis of C1(u) with –n
  - Align the y-axis of C1(u) with b
- Example: if C2(v) is a circle
- Variation (generalization)
- Scale C1(u) as it moves
- Morph C1(u) into C3(u) as it moves
- Use your own imagination!

Ruled Surfaces



Ruled Surfaces

- Move one straight line along a curve
- Example: plane, cone, cylinder
- Cylindrical surface
- Surface equation

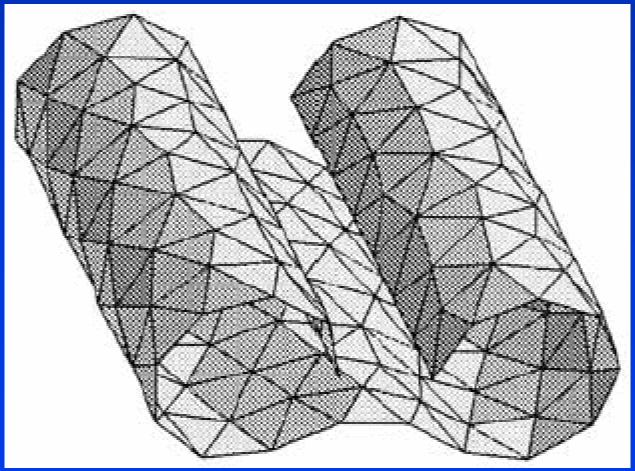
$$\mathbf{s}(u,v) = (1-v)\mathbf{a}(u) + v\mathbf{b}(u)$$
$$\mathbf{s}(u,v) = (1-v)\mathbf{s}(u,0) + v\mathbf{s}(u,1)$$
$$\mathbf{s}(u,v) = \mathbf{p}(u) + v\mathbf{q}(u)$$

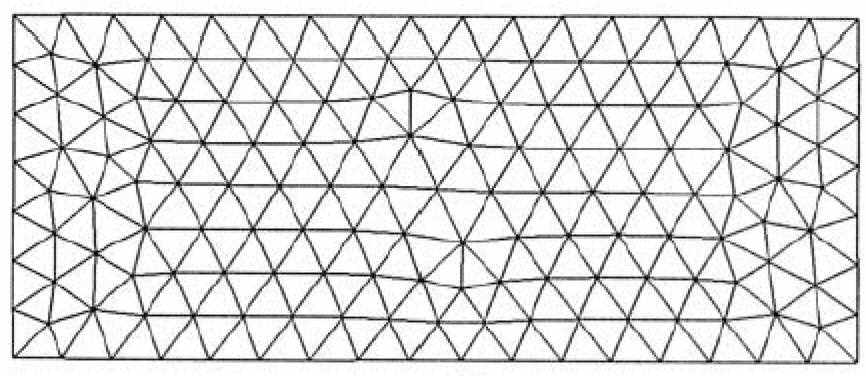
- Isoparametric lines
- More examples

Developable Surfaces

- Deform a surface to planar shape without length/area changes
- Unroll a surface to a plane without stretching/distorting
- Example: cone, cylinder
- Developable surfaces vs. Ruled surfaces
- More examples???

Developable Surface

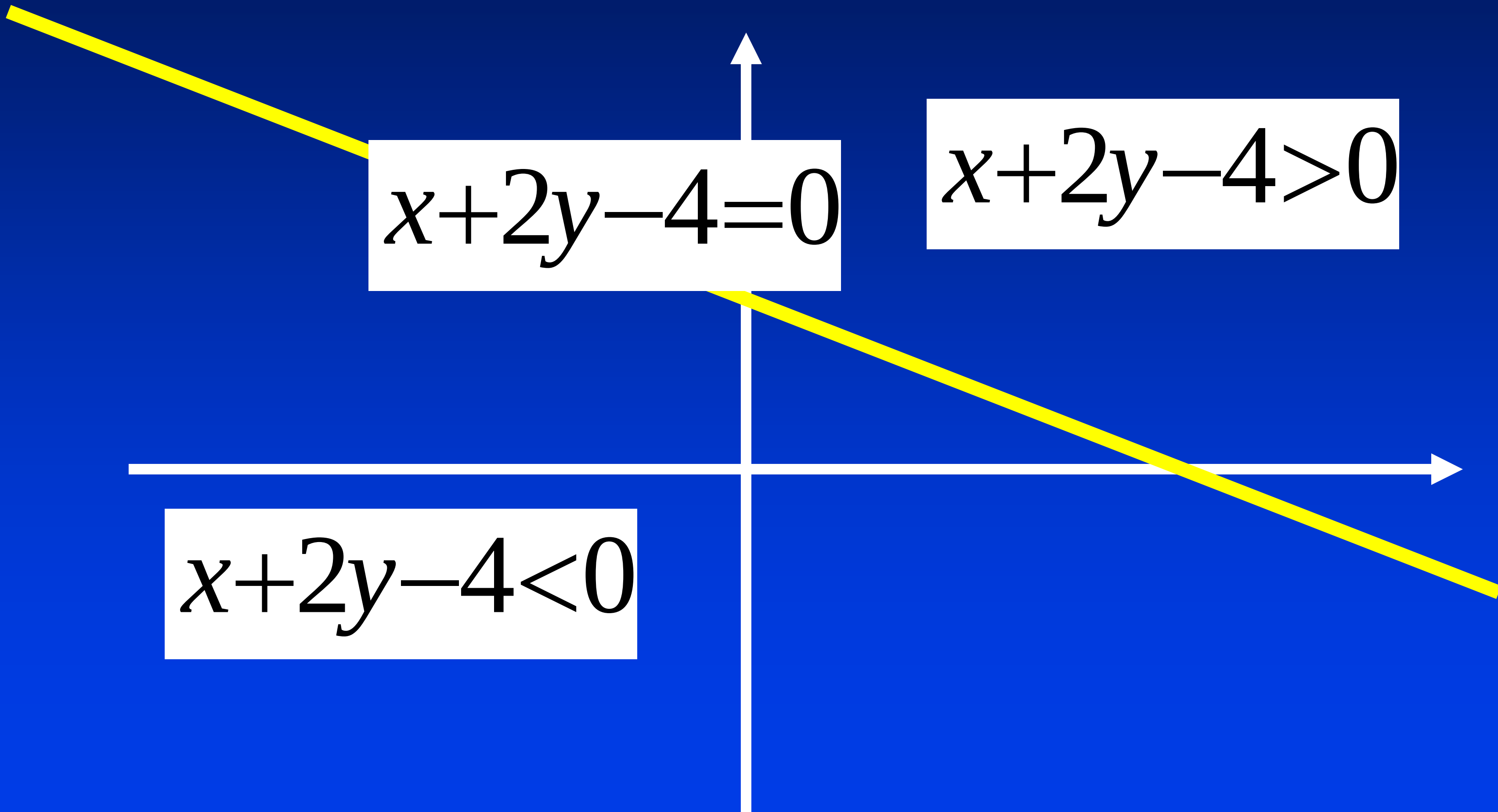




Summary

- Parametric curves and surfaces
- Polynomials and rational polynomials
- Free-form curves and surfaces
- Other commonly-used geometric primitives (e.g., sphere, ellipsoid, torus, superquadrics, blobby, etc.)
- Motivation:
  - Fewer degrees of freedom
  - More geometric coverage

Straight Line



Straight Line

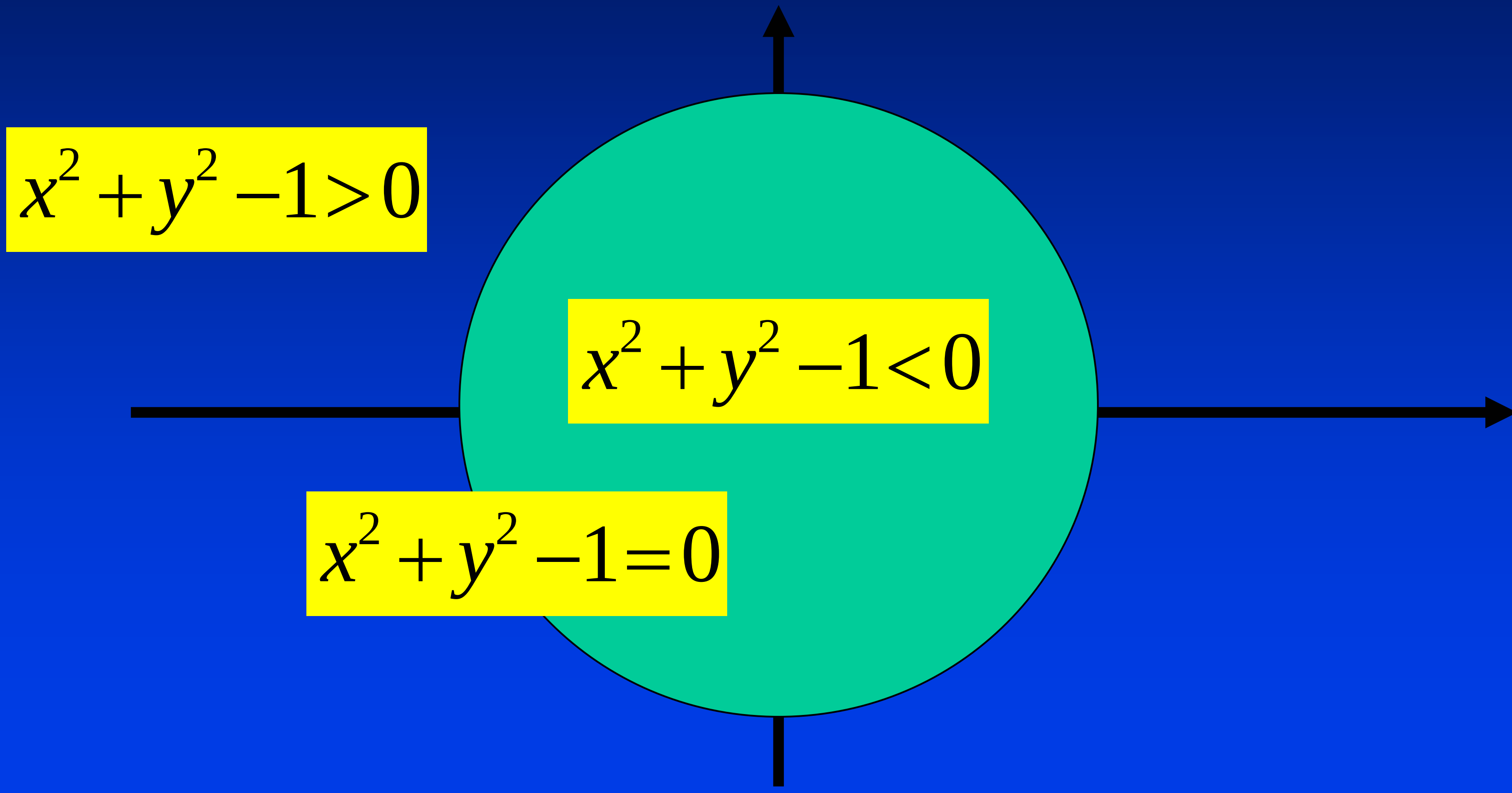
- Mathematics

$$ax + by + c = 0$$
$$+\alpha(ax + by + c) = 0$$
$$-\alpha(ax + y + c) = 0$$

- Example

$$x+2y-4=0$$

Circle



Conic Sections

- Mathematics

$$ax^2 + 2bxy + cy^2 + dx + ey + f = 0$$

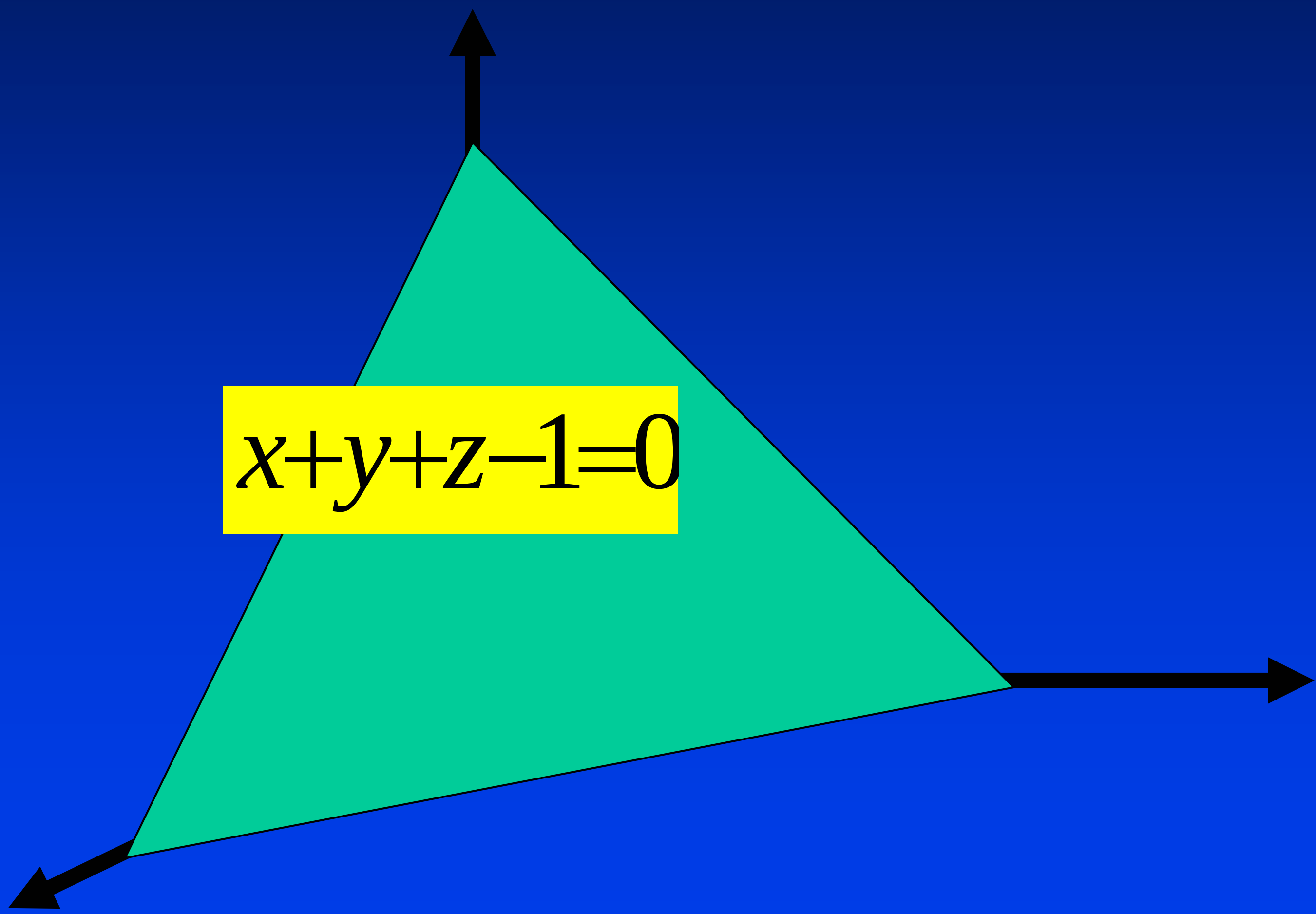
- Examples
  - Ellipse
  - Hyperbola
  - Parabola
  - Empty set
  - Point
  - Pair of lines
  - Parallel lines
  - Repeated lines

$$2x^2 + 3y^2 - 5 = 0$$
$$2x^2 - 3y^2 - 5 = 0$$
$$2x^2 + 3y = 0$$
$$2x^2 + 3y^2 + 1 = 0$$
$$2x^2 + 3y^2 = 0$$
$$2x^2 - 3y^2 = 0$$
$$2x^2 - 7 = 0$$
$$2x^2 = 0$$

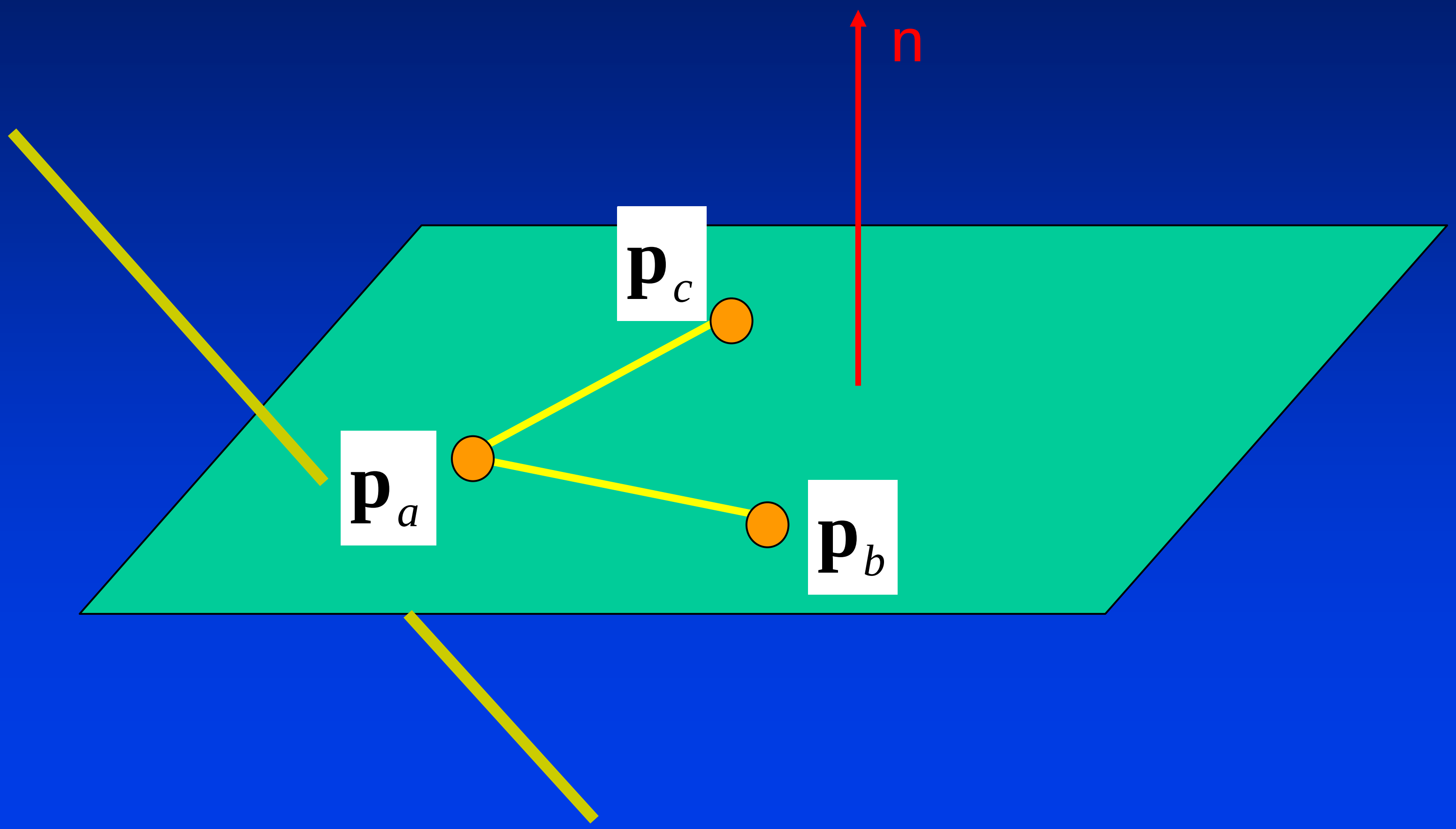
Conics

- Parametric equations of conics
- Generalization to higher-degree curves
- How about non-planar (spatial) curves

Plane



Plane and Intersection



Plane

- Example $x + y + z - 1 = 0$
- General plane equation $ax + by + cz + y = 0$
- Normal of the plane $$\mathbf{n} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$
- Arbitrary point on the plane

$$\mathbf{p}_a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}$$

Plane

- Plane equation derivation

$$(x-a_x)a+(y-a_y)b+(z-a_z)c=0$$
$$ax+by+cz-(a_xa+a_yb+a_zc)=0$$

- Parametric representation (given three points on the plane and they are non-collinear!)

$$\mathbf{p}(u,v)=\mathbf{p}_a+(\mathbf{p}_b-\mathbf{p}_a)u+(\mathbf{p}_c-\mathbf{p}_a)v$$

Plane

- Explicit expression (if c is non-zero)

$$z = -\frac{1}{c}(ax + by + d)$$

- Line-Plane intersection

$$\mathbf{l}(u) = \mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u$$

$$(\mathbf{n})(\mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u) + d = 0$$

$$u = -\frac{\mathbf{np}_0}{\mathbf{np}_1 - \mathbf{np}_0} = -\frac{plane(\mathbf{p}_0)}{plane(\mathbf{p}_1) - plane(\mathbf{p}_0)}$$

Circle

- Implicit equation $x^2 + y^2 - 1 = 0$
- Parametric function

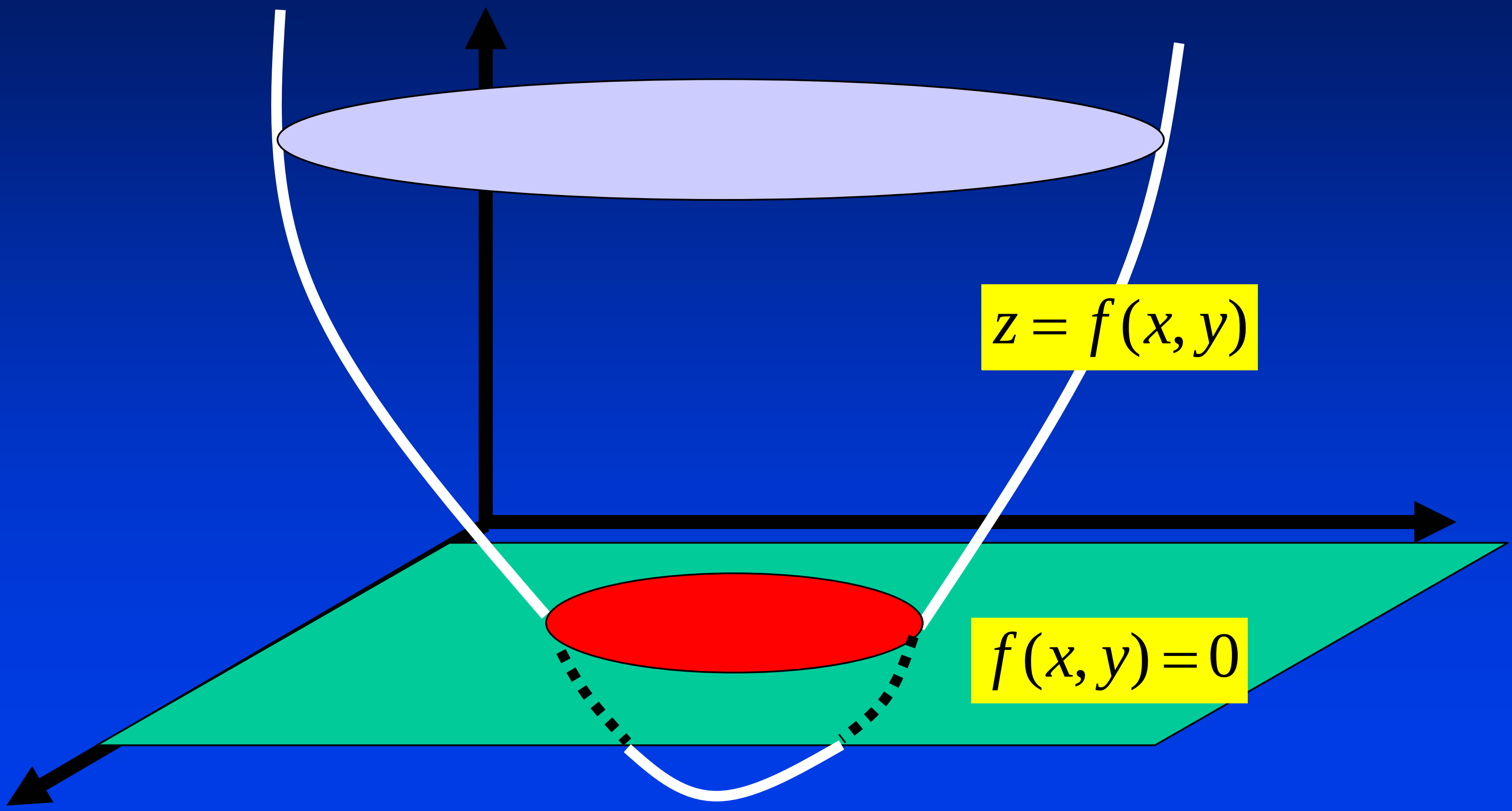
$$\mathbf{c}(\theta) = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$
$$0 <= \theta <= 2\pi$$

- Parametric representation using rational polynomials (the first quadrant)

$$x(u) = \frac{1 - u^2}{1 + u^2}$$
$$y(u) = \frac{2u}{1 + u^2}$$
$$u \in [0,1]$$

- Parametric representation is not unique!

CSE530-13


$z = f(x, y)$
$f(x, y) = 0$

Implicit Equations for Curves

- Describe an implicit relationship
- Planar curve (point set) $\{(x, y) \mid f(x, y) = 0\}$
- The implicit function is not unique

$$\{(x, y) \mid +\alpha f(x, y) = 0\}$$
$$\{(x, y) \mid -\alpha f(x, y) = 0\}$$

- Comparison with parametric representation

$$\mathbf{p}(u) = \begin{bmatrix} x(u) \\ y(u) \end{bmatrix}$$

Implicit Equations for Curves

- Implicit function is a level-set

$$\begin{cases} z = f(x, y) \\ z = 0 \end{cases}$$

- Examples (straight line and conic sections)

$$ax + by + c = 0$$

$$ax^2 + 2bxy + cy^2 + dx + ey + f = 0$$

- Other examples
  - Parabola, two parallel lines, ellipse, hyperbola, two intersection lines

Implicit Functions for Curves

- Parametric equations of conics
- Generalization to higher-degree curves
- How about non-planar (spatial) curves

Implicit Equations for Surfaces

- Surface mathematics $\{(x,y,z) \mid f(x,y,z)=0\}$
- Again, the implicit function for surfaces is not unique

$$\{(x,y,z) \mid +\alpha f(x,y,z)=0\}$$
$$\{(x,y,z) \mid -\alpha f(x,y,z)=0\}$$

- Comparison with parametric representation

$$\mathbf{p}(u,v)=\begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}$$

Implicit Equations for Surfaces

- Surface defined by implicit function is a level-set

$$\begin{cases} w = f(x, y, z) \\ w = 0 \end{cases}$$

- Examples
  - Plane, quadric surfaces, tori, superquadrics, blobby objects
- Parametric representation of quadric surfaces
- Generalization to higher-degree surfaces

Quadric Surfaces

- Implicit functions
- Examples

$$ax^2 + by^2 + cz^2 + dxy + exz + fyz + gx + hy + jz + k = 0$$

  - Sphere
  - Cylinder
  - Cone
  - Paraboloid
  - Ellipsoid
  - Hyperboloid

$$x^2 + y^2 + z^2 - 1 = 0$$
$$x^2 + y^2 - 1 = 0$$
$$x^2 + y^2 - z^2 = 0$$
$$x^2 + y^2 + z = 0$$
$$2x^2 + 3y^2 + 4z^2 - 5 = 0$$
$$x^2 + y^2 - z^2 + 4 = 0$$

- More
  - Two parallel planes, two intersecting planes, single plane, line, point

Quadrics: Parametric Rep.

- Sphere

$$x^2 + y^2 + z^2 - r^2 = 0$$
$$x = r\cos(\alpha)\cos(\beta)$$
$$y = r\cos(\alpha)\sin(\beta)$$
$$z = r\sin(\alpha)$$
$$\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}];\ \beta \in [-\pi, \pi]$$

- Ellipsoid

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0$$
$$x = a\cos(\alpha)\cos(\beta)$$
$$y = b\cos(\alpha)\sin(\beta)$$
$$z = c\sin(\alpha)$$
$$\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}];\ \beta \in [-\pi, \pi]$$

- Geometric meaning of these parameters

Generalization

- Higher-degree polynomials

$$\sum_i \sum_j \sum_k a_{ijk} x^i y^j z^k = 0$$

- Non polynomials

Superquadrics

- Geometry (generalization of quadrics)
- Superellipse

$$\left(\frac{x}{a_1}\right)^{\frac{2}{s}} + \left(\frac{y}{a_2}\right)^{\frac{2}{s}} - 1 = 0$$

- Superellipsoid

$$\left(\left(\frac{x}{a_1}\right)^{\frac{2}{s_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{s_2}}\right)^{\frac{s_2}{s_1}} + \left(\frac{z}{a_3}\right) - 1 = 0$$

- Parametric representation

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 \cos^{s_1}(\alpha) \sin^{s_2}(\beta) \\ a_2 \cos^{s_1}(\alpha) \sin^{s_2}(\beta) \\ a_3 \sin^{s_2}(\alpha) \end{bmatrix}$$

$$\alpha \in [-\frac{\pi}{2}, \frac{\pi}{2}]; \ \beta \in [-\pi, \pi)$$

- What is the meaning of these control parameters?

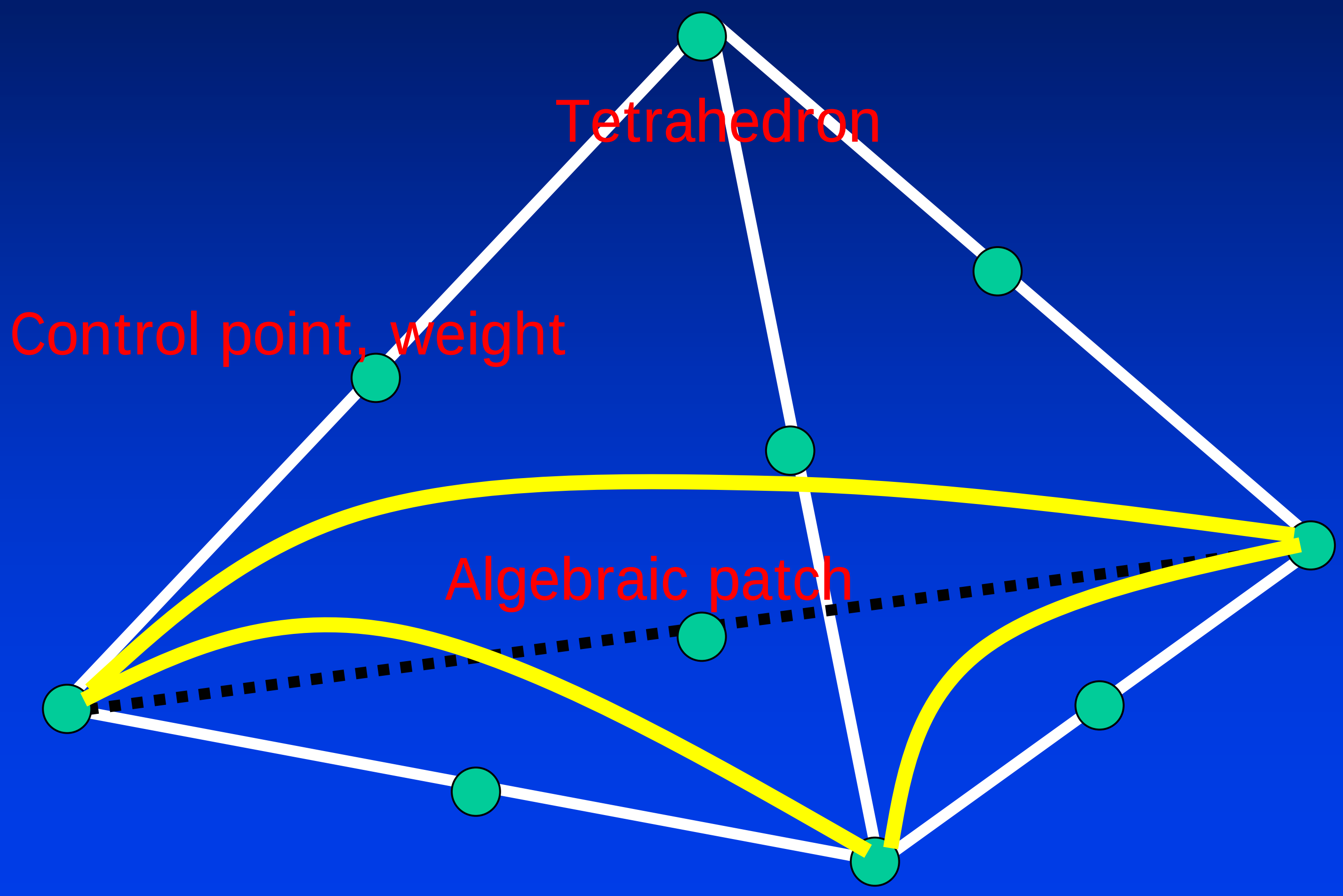
Algebraic Function

- Parametric representation is popular, but…
- Formulation

$$\sum_i \sum_j \sum_k a_{ijk} x^i y^j z^k = 0$$

- Properties…
  - Powerful, but lack of modeling tools

Algebraic Patch



Algebraic Patch

- A tetrahedron with non-planar vertices

$$\mathbf{v}_{n000}, \mathbf{v}_{0n00}, \mathbf{v}_{00n0}, \mathbf{v}_{000n}$$

- Trivariate barycentric coordinate (r,s,t,u) for p

$$\mathbf{p} = r\mathbf{v}_{n000} + s\mathbf{v}_{0n00} + t\mathbf{v}_{00n0} + u\mathbf{v}_{000n}$$
$$r + s + t + u = 1$$

- A regular lattice of control points and weights

$$\mathbf{p}_{ijkl} = \frac{i\mathbf{v}_{n000} + j\mathbf{v}_{0n00} + k\mathbf{v}_{00n0} + l\mathbf{v}_{000n}}{n}$$
$$i, j, k, l >= 0; i + j + k + l = n$$

Algebraic Patch

- There are (n+1)(n+2)(n+3)/6 control points. A weight w(I,j,k,l) is also assigned to each control point
- Algebraic patch formulation

$$\sum_i \sum_j \sum_k \sum_{l=n-i-j-k} w_{ijkl} \frac{n!}{i!\,j!\,k!\,l!} r^i s^j t^k u^l = 0$$

- Properties
  – Meaningful control, local control, boundary interpolation, gradient control, self-intersection avoidance, continuity condition across the boundaries, subdivision

Spatial Curves

- Intersection of two surfaces

$$\begin{cases} f(x, y, z) = 0 \\ g(x, y, z) = 0 \end{cases}$$

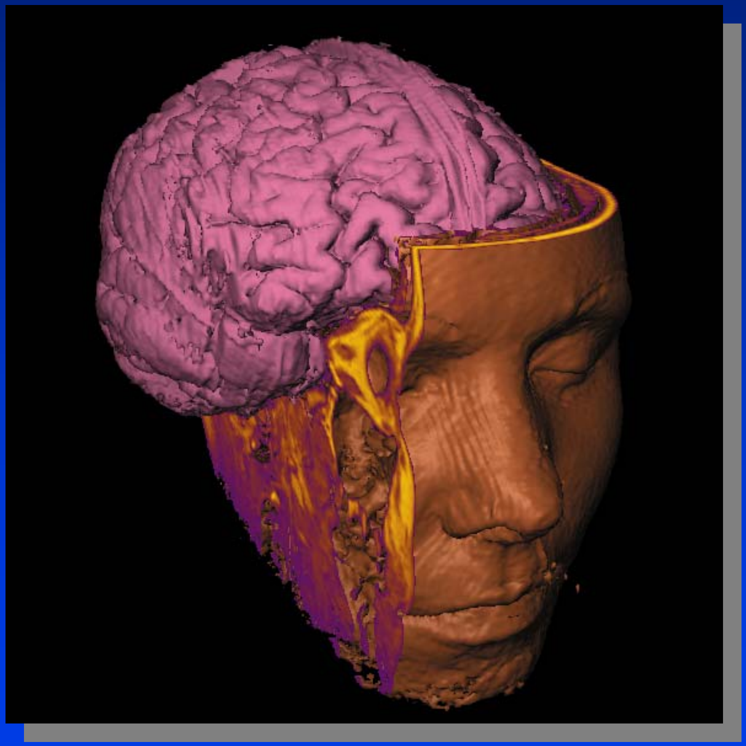
Algebraic Solid

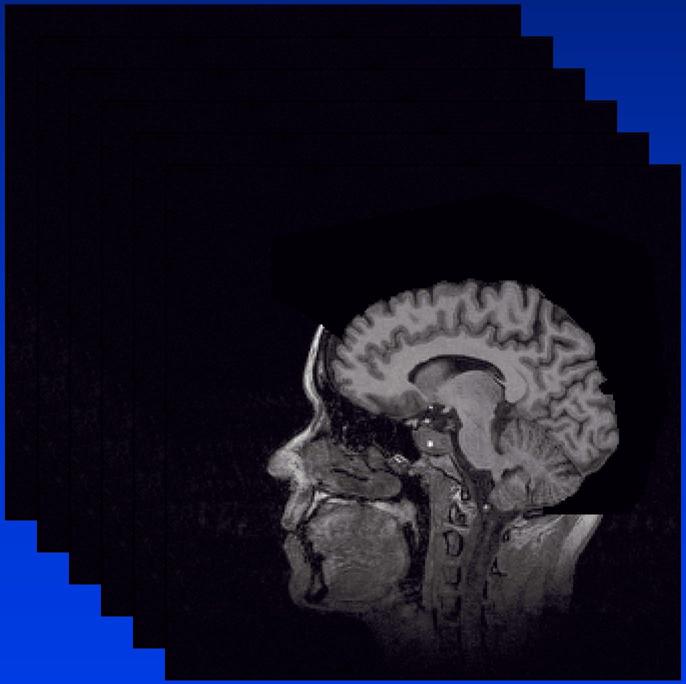
- Half space $\{(x,y,z) \mid f(x,y,z) <= 0\}; or$
  $\{(x,y,z) \mid f(x,y,z) >= 0\}$

- Useful for complex objects (refer to notes on solid modeling)

$$\mathbf{f}(x,y,z) = \begin{bmatrix} f_1(x,y,z) \\ f_2(x,y,z) \\ f_3(x,y,z) \\ \Lambda \end{bmatrix} = \mathbf{0}$$

Volume Datasets

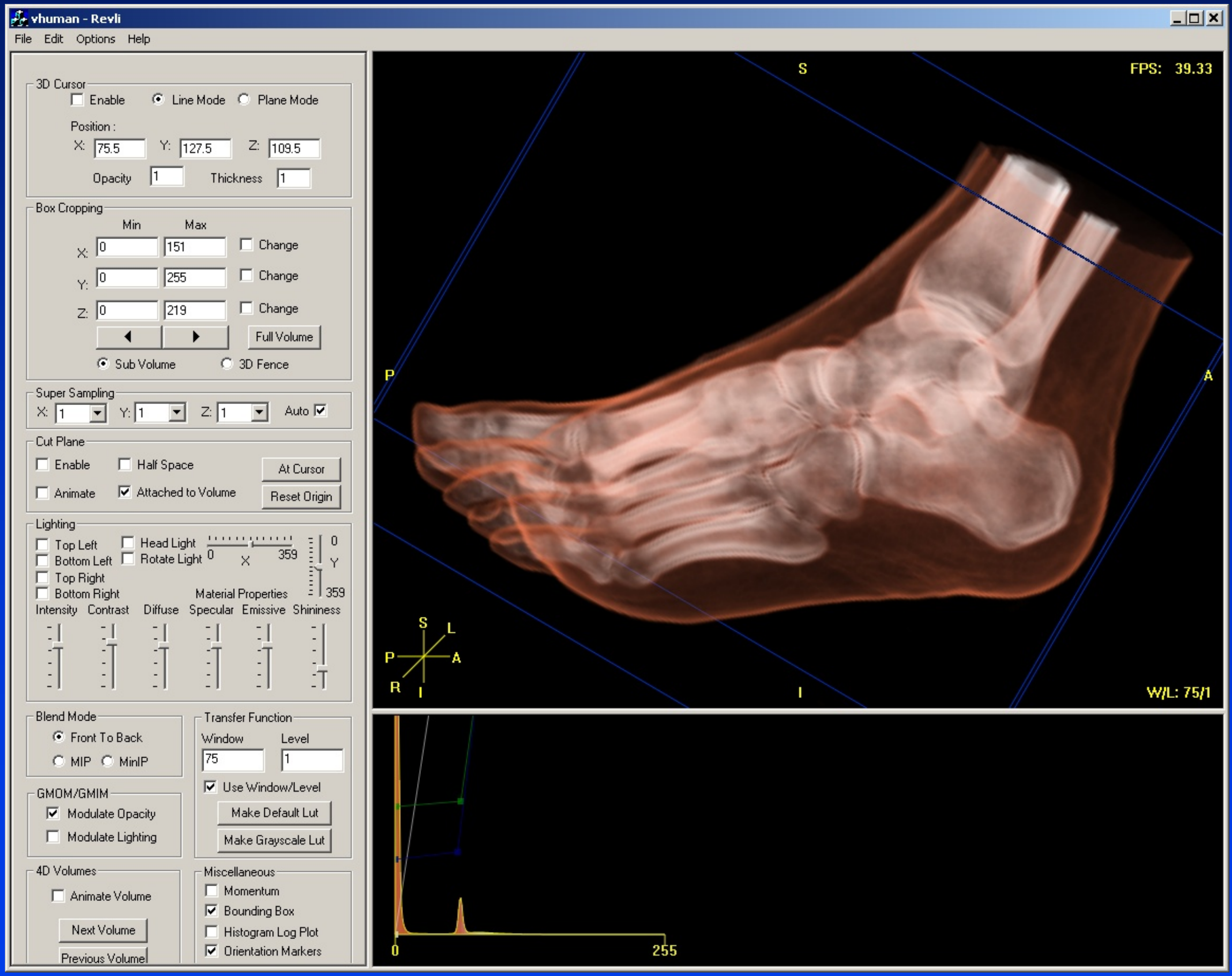




Isosurface Rendering

Isovalue =30 Isovalue =100 Isovalue =200

Direct Volume Rendering



Implicit Functions

- Long history: classical algebraic geometry
- Implicit and parametric forms
  – Advantages
  – Disadvantages
- Curves, surfaces, solids in higher-dimension
- Intersection computation
- Point classification
- Larger than parameter-based modeling
- Unbounded geometry
- Object traversal
- Evaluation

Implicit Functions

- Efficient algorithms, toolkits,software
- Computer-based shape modeling and design
- Geometric degeneracy and anomaly
- Algebraic and geometric operations are often closed
- Mathematics: algebraic geometry
- Symbolic computation
- Deformation and transformation
- Shape editing, rendering, and control

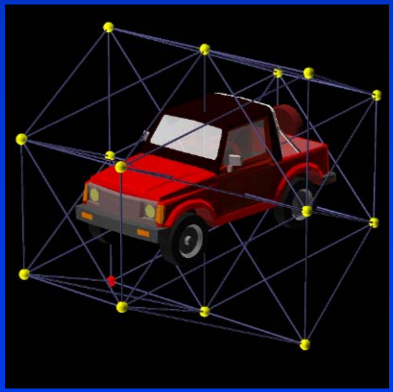
Implicit Functions

- Conversion between parametric and implicit forms
- Implicititization vs. parameterization
- Strategy: integration of both techniques
- Approximation using parametric models

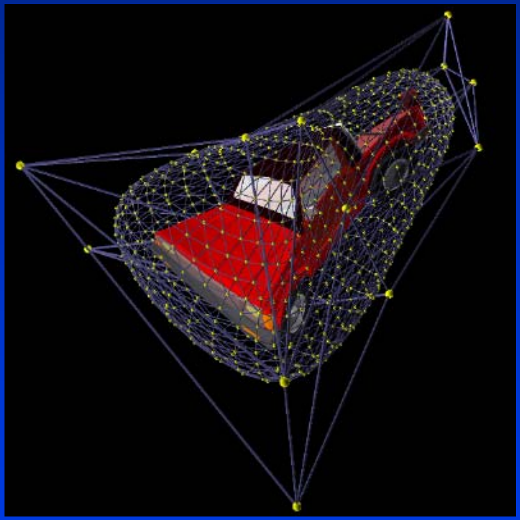
Free-Form Deformation

- Free-Form Deformation Example

Original Model



Solid Mesh



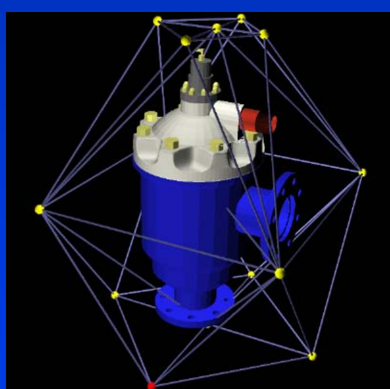
Deformed Mesh

Result

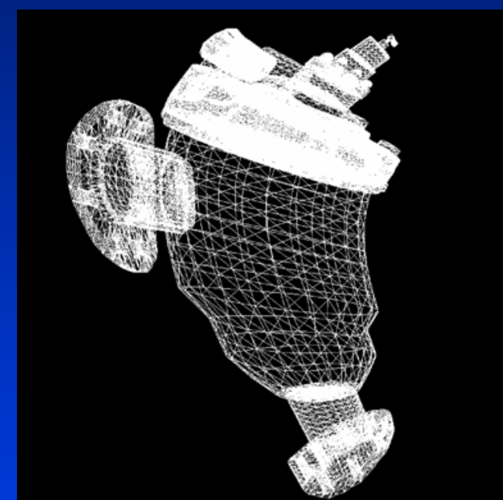
Free-Form Deformation

- Free-Form Deformation Example (Complex >> 49000 faces)

Original Model



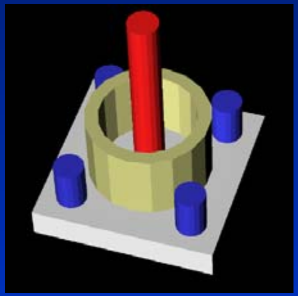
Solid Mesh



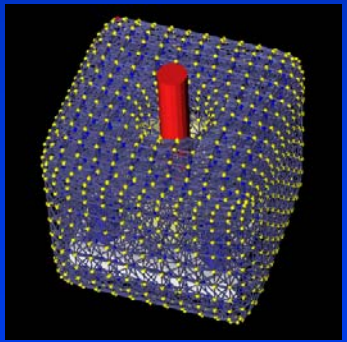
Deformed (Results in both surface rendered and wireframe)

Free-Form Deformation

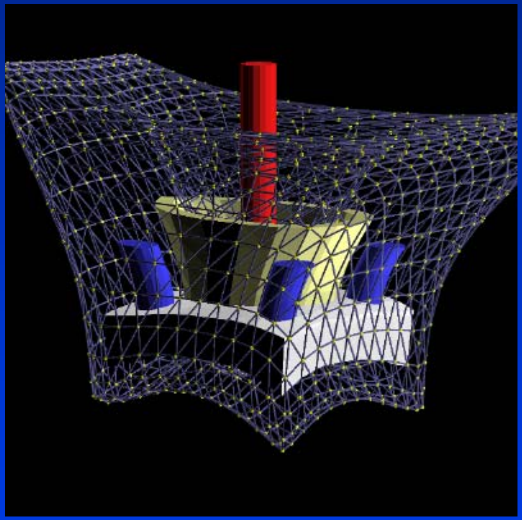
- Free-Form Deformation Example (Non-trivial topology)



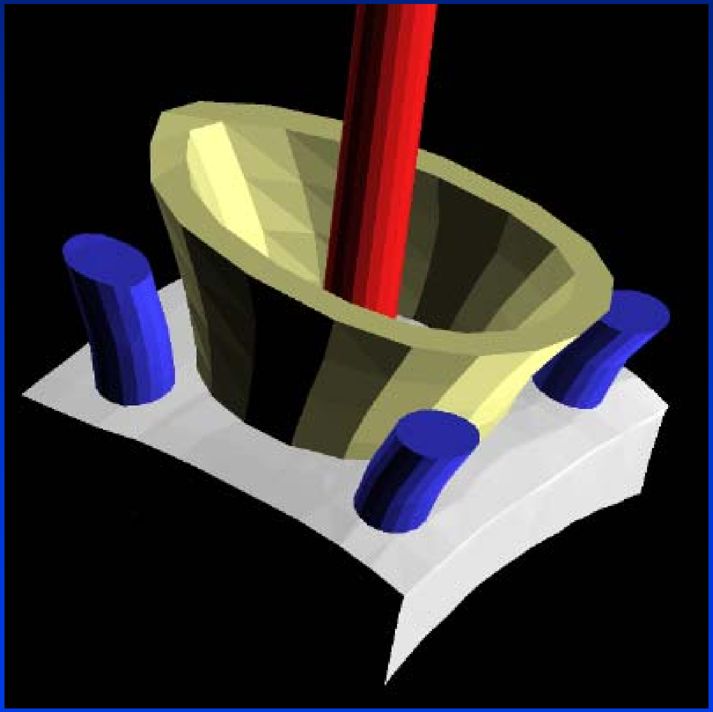
Original Model



Solid Mesh with a hole



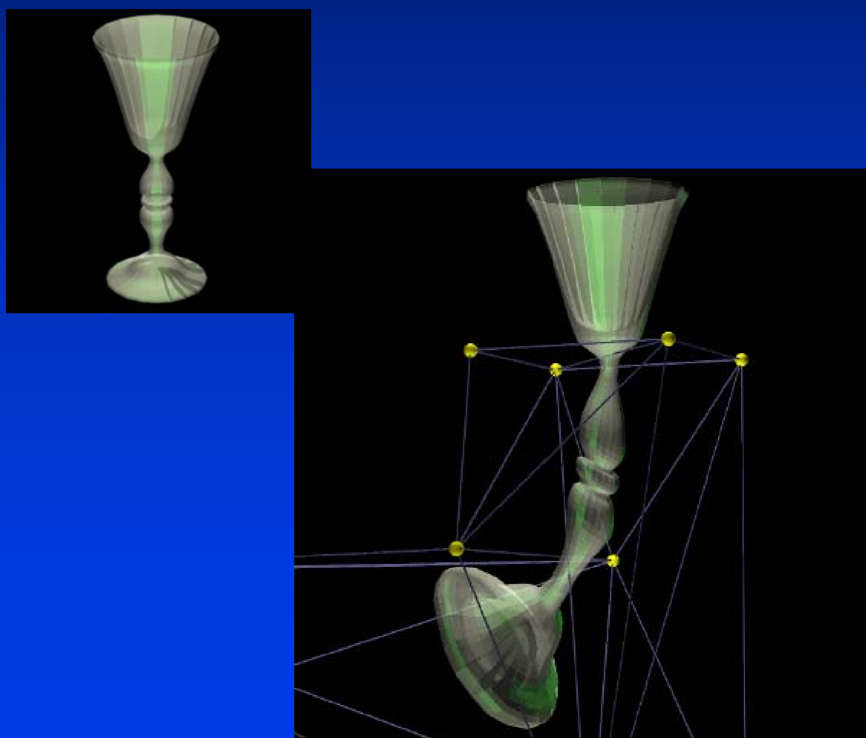
Deformed Mesh

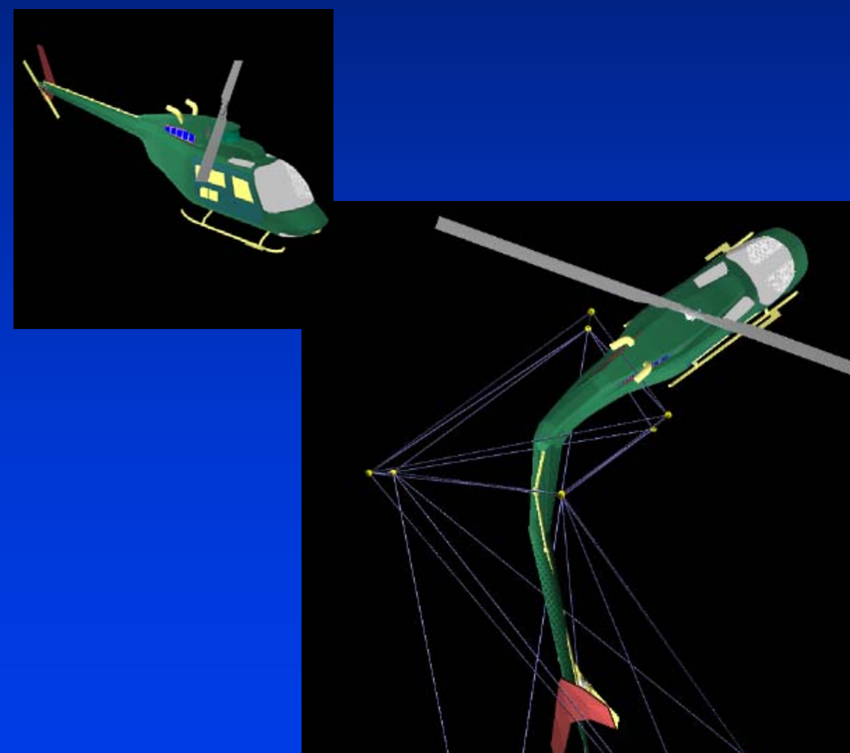


Result (no change in central cylinder)

Free-Form Deformation

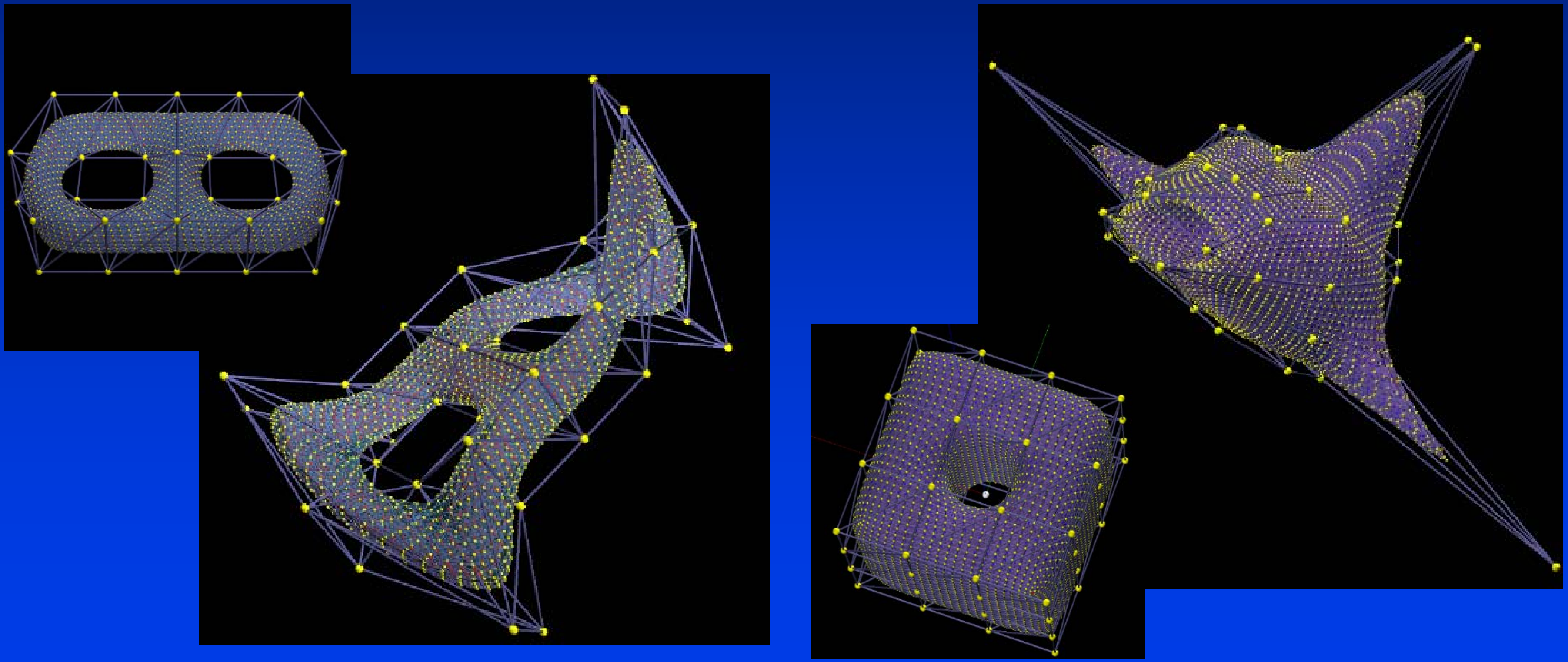
- Free-Form Deformation Example (Localized)





Shape Modeling

- Direct Modeling / Manipulation



Material Modeling

- Material Representation (Non-homogeneous)