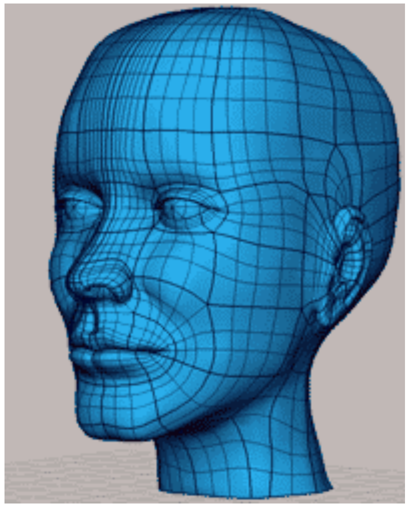
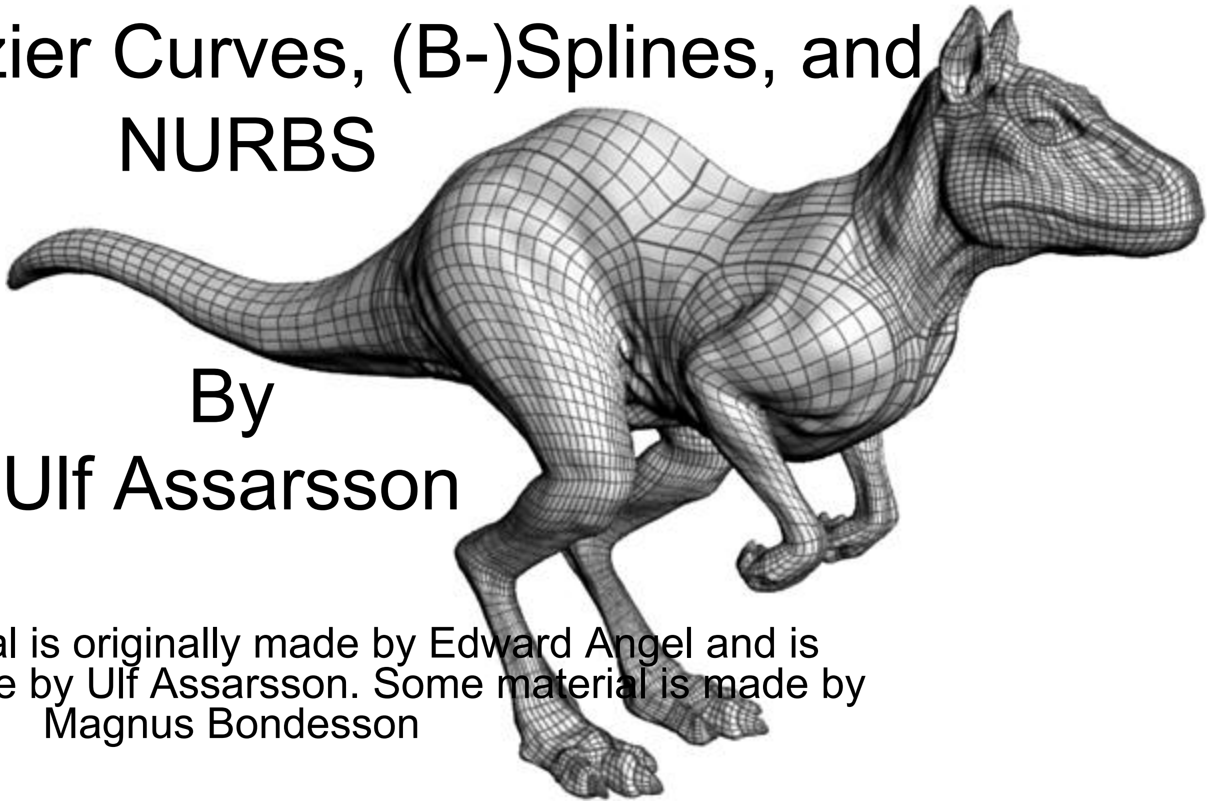
Computer Graphics

Curves and Surfaces

Hermite/Bezier Curves, (B-)Splines, and NURBS



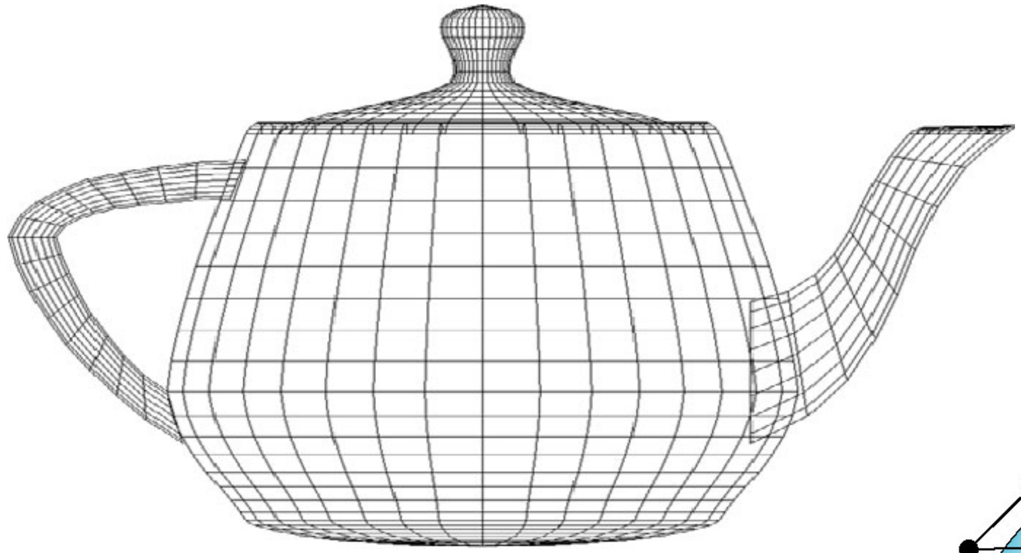


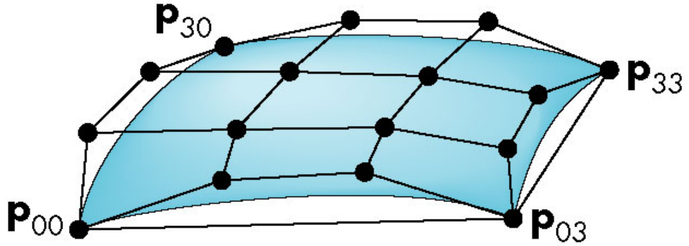
By
Ulf Assarsson

Most of the material is originally made by Edward Angel and is adapted to this course by Ulf Assarsson. Some material is made by Magnus Bondesson

Utah Teapot

- Most famous data set in computer graphics
- Widely available as a list of 306 3D vertices and the indices that define 32 Bezier patches





A Bezier patch

If you want, you can find more information here:

- ”KURV- OCH YTAPPROXIMATION MED POLYNOM” by Magnus Bondesson:
  - http://www.ce.chalmers.se/edu/course/TDA361/DG_KURV2004.pdf

- See also book, Angel ”Interactive Computer Graphics – A Top-Down Approach Using OpenGL” chapter 11, pages 569-624

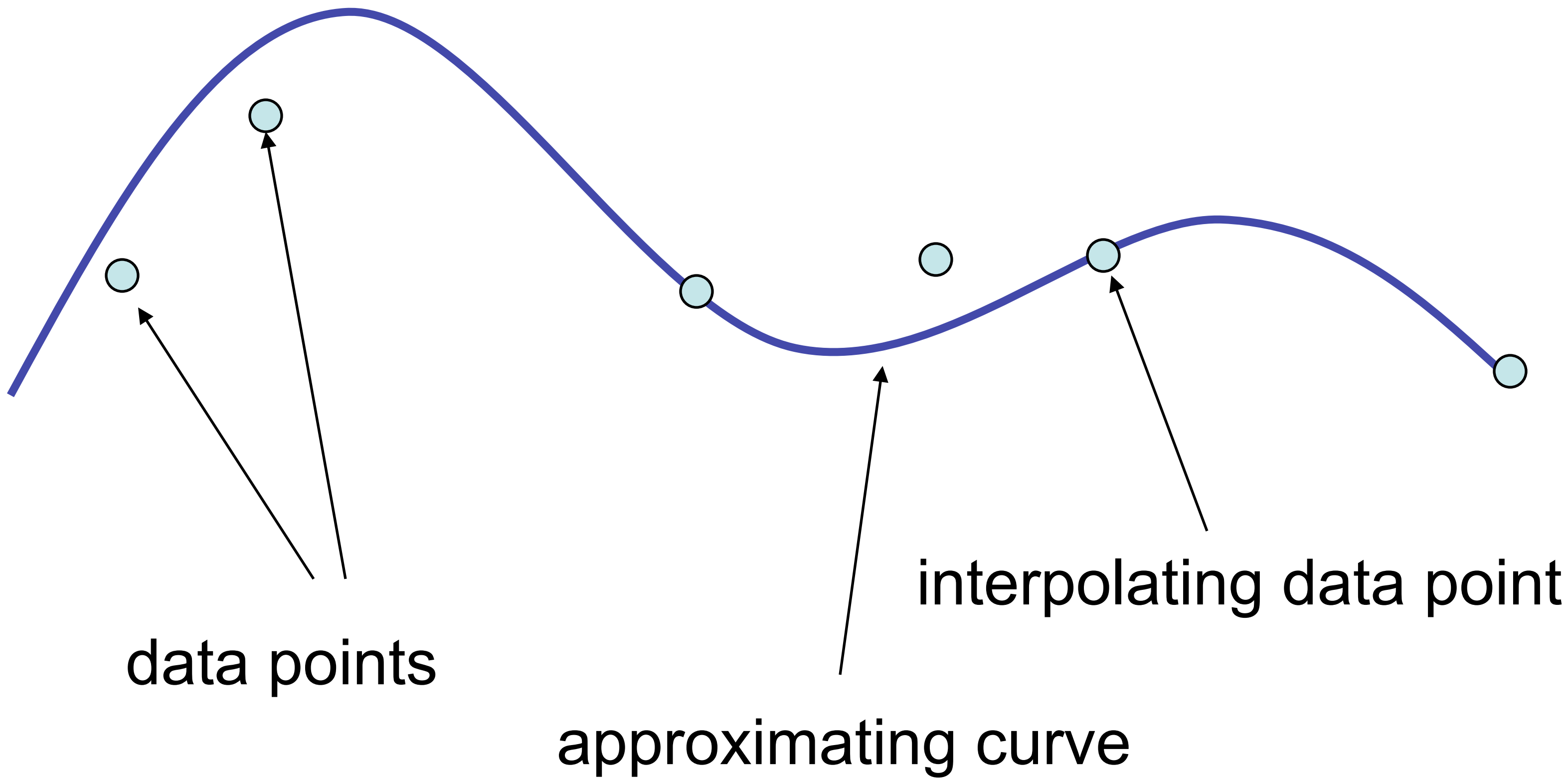
- In Swedish: Chapter 24, page 34-38 in ”Introduktion till OpenGL”
  - http://www.ce.chalmers.se/edu/course/TDA361/Introduktion%20till%20OpenGL.pdf

- OH 114-138
  - Available at bottom of page, here:
  - http://www.cse.chalmers.se/edu/course/TDA361/schedule.html

Objectives

- Introduce types of curves and surfaces
  - Explicit
  - Implicit
  - Parametric

Modeling with Curves



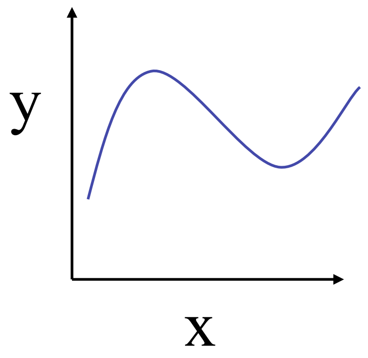
What Makes a Good Representation?

- There are many ways to represent curves and surfaces
- Want a representation that is
  - Stable
  - Smooth
  - Easy to evaluate
  - Must we interpolate or can we just come close to data?
  - Do we need derivatives?

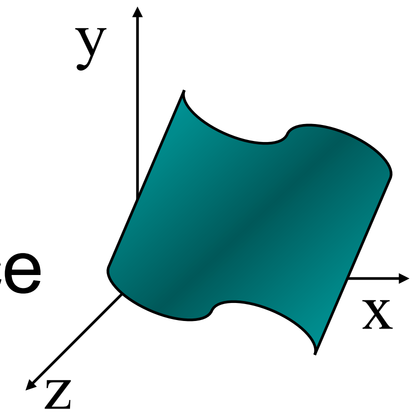
Explicit Representation

- Most familiar form of curve in 2D

  $y=f(x)$



- Cannot represent all curves
  - Vertical lines
  - Circles
- Extension to 3D
  - $y=f(x)$, $z=g(x)$ – gives a curve
  - The form $z = f(x,y)$ defines a surface



Implicit Representation

- Two dimensional curve(s)

  $g(x,y)=0$

- Much more robust
  - All lines $ax+by+c=0$
  - Circles $x^2+y^2-r^2=0$
- Three dimensions $g(x,y,z)=0$ defines a surface
  - (we could intersect two surfaces to get a curve)

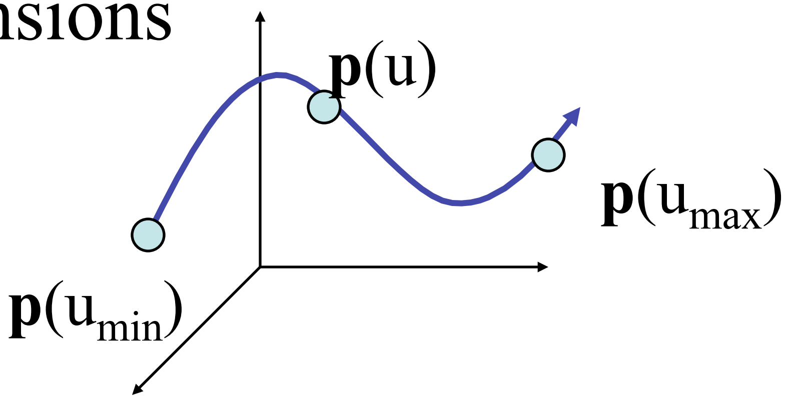
Parametric Curves

- Separate equation for each spatial variable

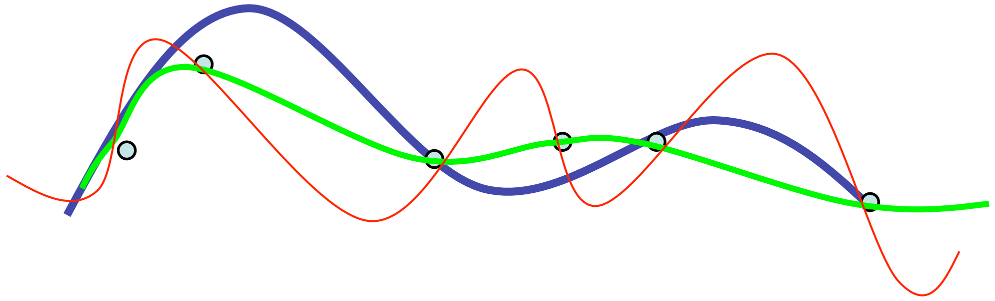
$$x=x(u)$$
$$y=y(u)$$
$$z=z(u)$$

$$\mathbf{p}(u)=[x(u), y(u), z(u)]^T$$

- For $u_{max} \geq u \geq u_{min}$ we trace out a curve in two or three dimensions



Selecting Functions



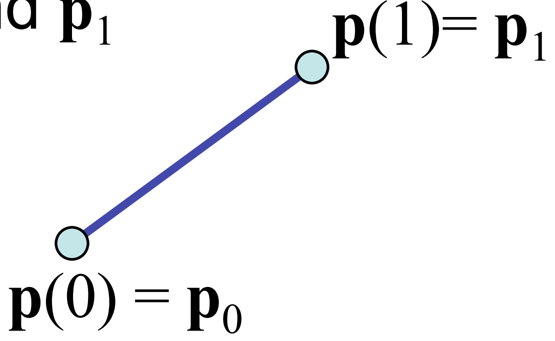
- Usually we can select “good” functions
  - not unique for a given spatial curve
  - Approximate or interpolate known data
  - Want functions which are easy to evaluate
  - Want functions which are easy to differentiate
    - Computation of normals
    - Connecting pieces (segments)
  - Want functions which are smooth

Parametric Lines

We can let u be over the interval (0,1)

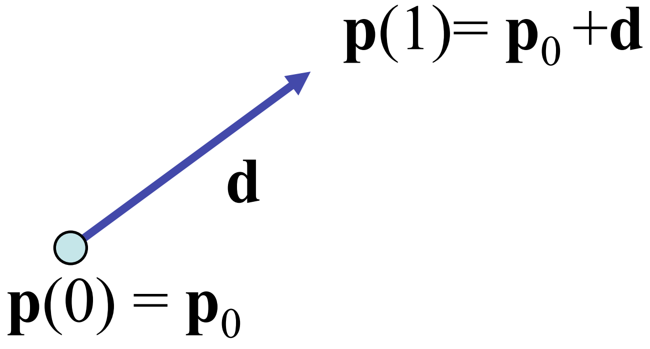
Line connecting two points $\mathbf{p}_0$ and $\mathbf{p}_1$

$$\mathbf{p}(u)=(1-u)\mathbf{p}_0+u\mathbf{p}_1$$



Ray from $\mathbf{p}_0$ in the direction $\mathbf{d}$

$$\mathbf{p}(u)=\mathbf{p}_0+u\mathbf{d}$$

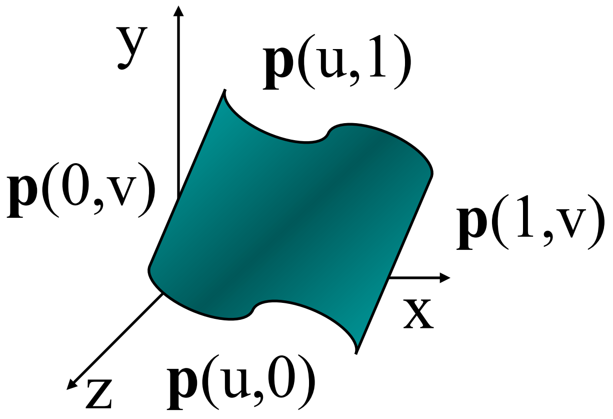


$\mathbf{p}(1)= \mathbf{p}_0 +\mathbf{d}$
$\mathbf{d}$
$\mathbf{p}(0) = \mathbf{p}_0$

Parametric Surfaces

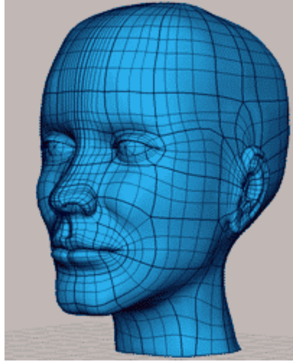
- Surfaces require 2 parameters

$$x=x(u,v)$$
$$y=y(u,v)$$
$$z=z(u,v)$$



$$\mathbf{p}(u,v) = [x(u,v), y(u,v), z(u,v)]^T$$

- Want same properties as curves:
  - Smoothness
  - Differentiability
  - Ease of evaluation



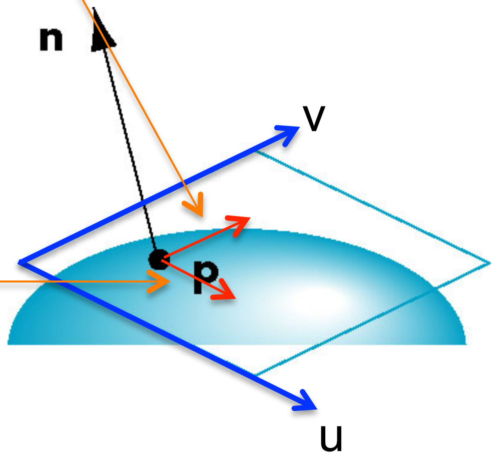
Normals

We can differentiate with respect to u and v to obtain the normal at any point $\mathbf{p}$

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

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

$$\mathbf{n} = \frac{\partial \mathbf{p}(u,v)}{\partial u} \times \frac{\partial \mathbf{p}(u,v)}{\partial v}$$



Parametric Planes

point-vector form

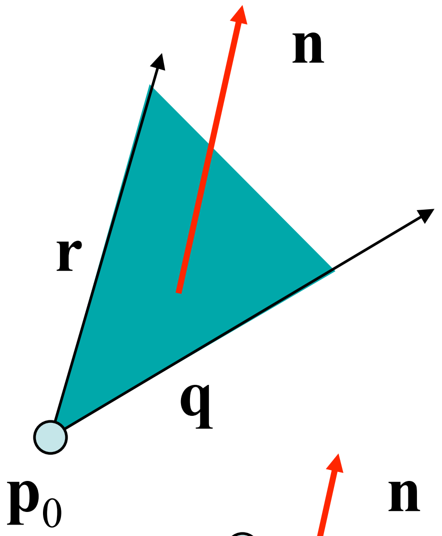
$$\mathbf{p}(u,v)=\mathbf{p}_0+u\mathbf{q}+v\mathbf{r}$$

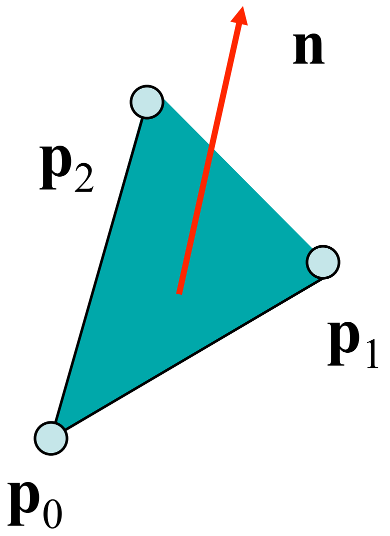
$$\frac{\partial \mathbf{p}(u,v)}{\partial u} \times \frac{\partial \mathbf{p}(u,v)}{\partial v}$$

$$\mathbf{n} = \mathbf{q} \text{ x } \mathbf{r}$$

(three-point form

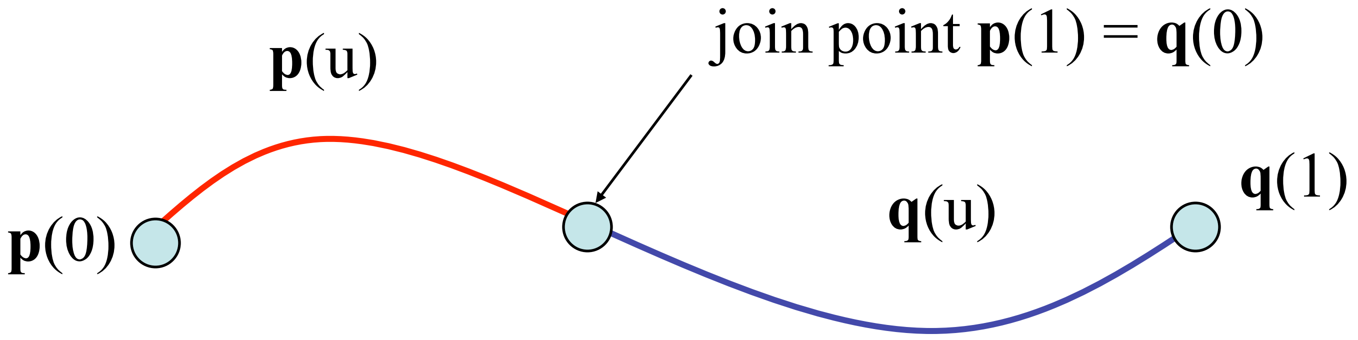
$$\mathbf{q} = \mathbf{p}_1 - \mathbf{p}_0$$
$$\mathbf{r} = \mathbf{p}_2 - \mathbf{p}_0 \quad)$$





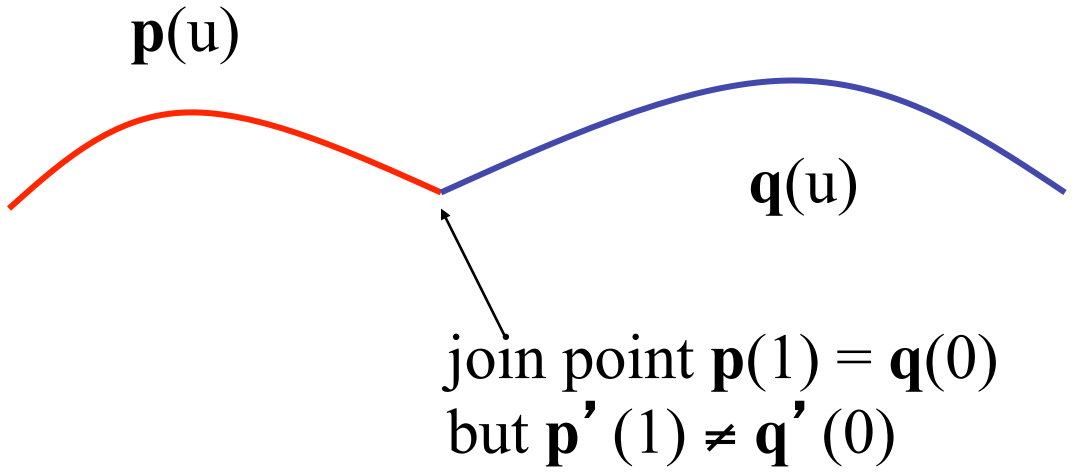
Curve Segments

- After normalizing u, each curve is written
  $\mathbf{p}(u)=[x(u), y(u), z(u)]^T, \quad 1 \geq u \geq 0$
- In classical numerical methods, we design a single global curve
- In computer graphics and CAD, it is better to design small connected curve *segments*



We choose Polynomials

- Easy to evaluate
- Continuous and differentiable everywhere
  - Must worry about continuity at join points including continuity of derivatives



$\mathbf{p}(u)$
$\mathbf{q}(u)$
join point $\mathbf{p}(1) = \mathbf{q}(0)$
but $\mathbf{p}'(1) \neq \mathbf{q}'(0)$

Parametric Polynomial Curves

$$x(u) = \sum_{i=0}^{N} c_{xi} u^i \quad y(u) = \sum_{j=0}^{M} c_{yj} u^j \quad z(u) = \sum_{k=0}^{L} c_{zk} u^k$$

•Cubic polynomials gives N=M=L=3

•Noting that the curves for x, y and z are independent, we can define each independently in an identical manner

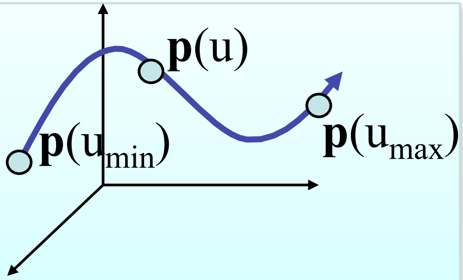
•We will use the form $\mathrm{p}(u) = \sum_{k=0}^{L} c_k u^k$
where p can be any of x, y, z

Remember:

x=x(u)

y=y(u)

z=z(u)



where $\mathbf{p}(u)=[x(u), y(u), z(u)]^T$

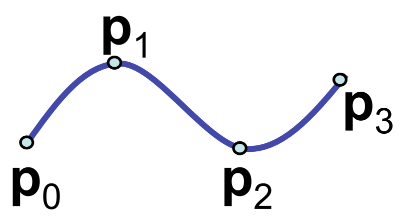
Cubic Parametric Polynomials

- Cubic polynomials give balance between ease of evaluation and flexibility in design

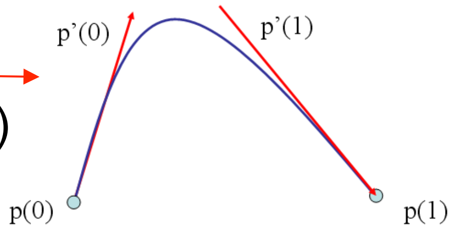
$$\mathrm{p}(u) = \sum_{k=0}^{3} c_k u^k$$

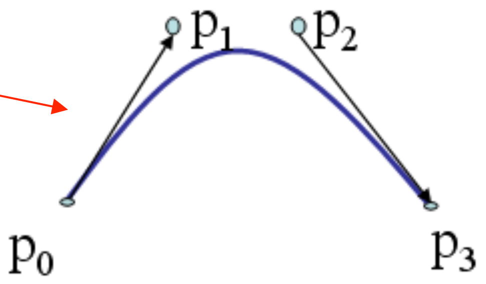
- Four coefficients to determine for each of x, y and z
- Seek four independent conditions for various values of u resulting in 4 equations in 4 unknowns for each of x, y and z
  - Conditions are a mixture of continuity requirements at the join points and conditions for fitting the data

Objectives



- Introduce the types of curves
  - Interpolating
    - Blending polynomials for interpolation of 4 control points (fit curve to 4 control points)
  - Hermite
    - fit curve to 2 control points + 2 derivatives (tangents)
  - Bezier
    - 2 interpolating control points + 2 intermediate points to define the tangents
  - B-spline
    - To get C^1 and C^2 continuity
  - NURBS
    - Different weights of the control points
- Analyze them





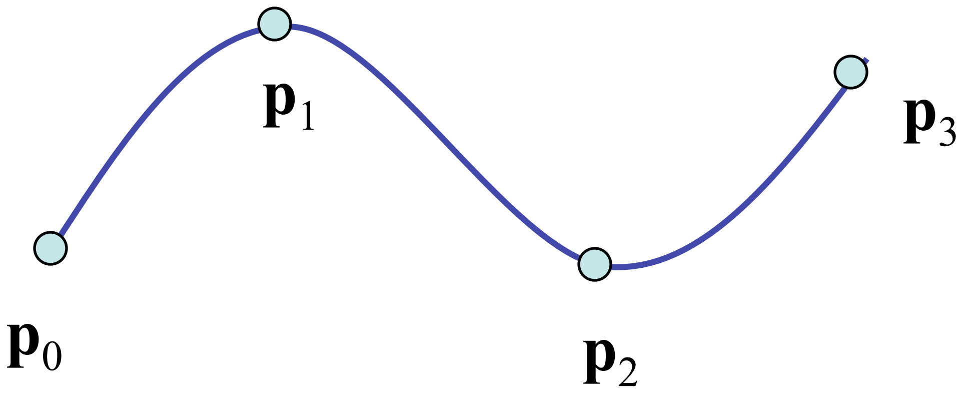
Matrix-Vector Form

$$\mathrm{p}(u) = \sum_{k=0}^{3} c_k u^k$$

define $\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$ $\mathbf{u} = \begin{bmatrix} 1 \\ u \\ u^2 \\ u^3 \end{bmatrix}$

then $\mathrm{p}(u) = \mathbf{u}^T \mathbf{c} = \mathbf{c}^T \mathbf{u}$

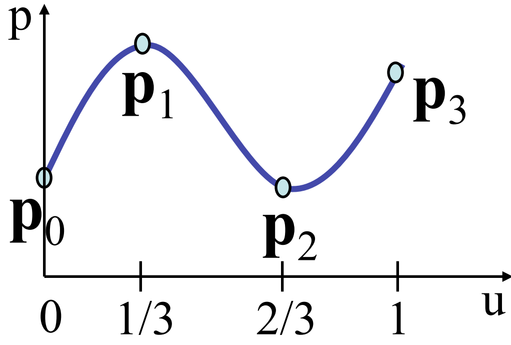
Interpolating Curve



Given four data (control) points $\mathbf{p}_0$, $\mathbf{p}_1$,$\mathbf{p}_2$, $\mathbf{p}_3$ determine cubic $\mathbf{p}(u)$ which passes through them

Must find $\mathbf{c}_0$,$\mathbf{c}_1$,$\mathbf{c}_2$, $\mathbf{c}_3$

Interpolation Equations



$p(u) = c_0 + c_1u + c_2u^2 + c_3u^3$

apply the interpolating conditions at $u=0, 1/3, 2/3, 1$

$$\left\{\begin{aligned} p_0 &= p(0) = c_0 \\ p_1 &= p(1/3) = c_0+(1/3)c_1+(1/3)^2c_2+(1/3)^3c_3 \\ p_2 &= p(2/3) = c_0+(2/3)c_1+(2/3)^2c_2+(2/3)^3c_3 \\ p_3 &= p(1) = c_0+c_1+c_2+c_3 \end{aligned}\right.$$

or in matrix form with $\mathbf{p} = [p_0\ p_1\ p_2\ p_3]^T$

$$\mathbf{p=Ac}$$

I.e., $\mathbf{c=A^{-1}p}$

$$\mathbf{p} = \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} = \mathbf{Ac} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & \left(\frac{1}{3}\right) & \left(\frac{1}{3}\right)^2 & \left(\frac{1}{3}\right)^3 \\ 1 & \left(\frac{2}{3}\right) & \left(\frac{2}{3}\right)^2 & \left(\frac{2}{3}\right)^3 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Interpolation Matrix

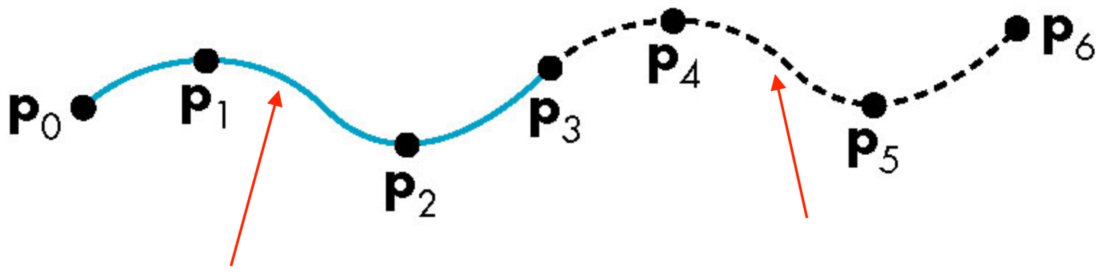
Solving for $\mathbf{c}$ we find the *interpolation matrix*

$$\mathbf{M}_I = \mathbf{A}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -5.5 & 9 & -4.5 & 1 \\ 9 & -22.5 & 18 & -4.5 \\ -4.5 & 13.5 & -13.5 & 4.5 \end{bmatrix}$$

$$\mathbf{c} = \mathbf{M}_I \mathbf{p}$$

Note that $\mathbf{M}_I$ does not depend on input data and can be used for each segment in x, y, and z

Interpolating Multiple Segments

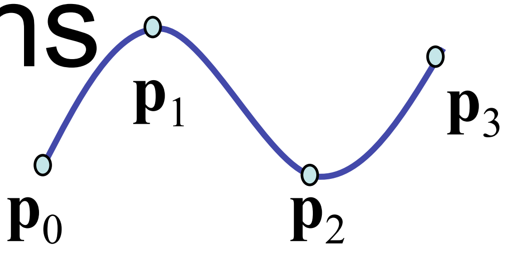


use $\mathbf{p} = [p_0\ p_1\ p_2\ p_3]^T$

use $\mathbf{p} = [p_3\ p_4\ p_5\ p_6]^T$

Get continuity at join points but not continuity of derivatives

Blending Functions



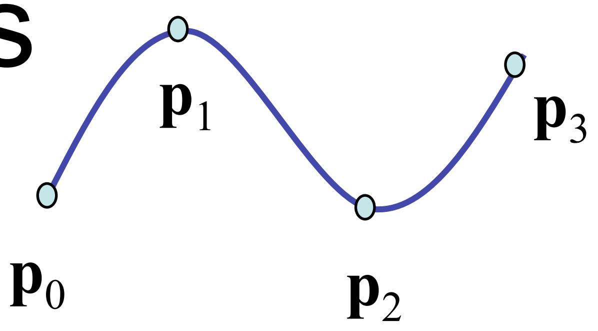
Rewriting the equation for p(u)

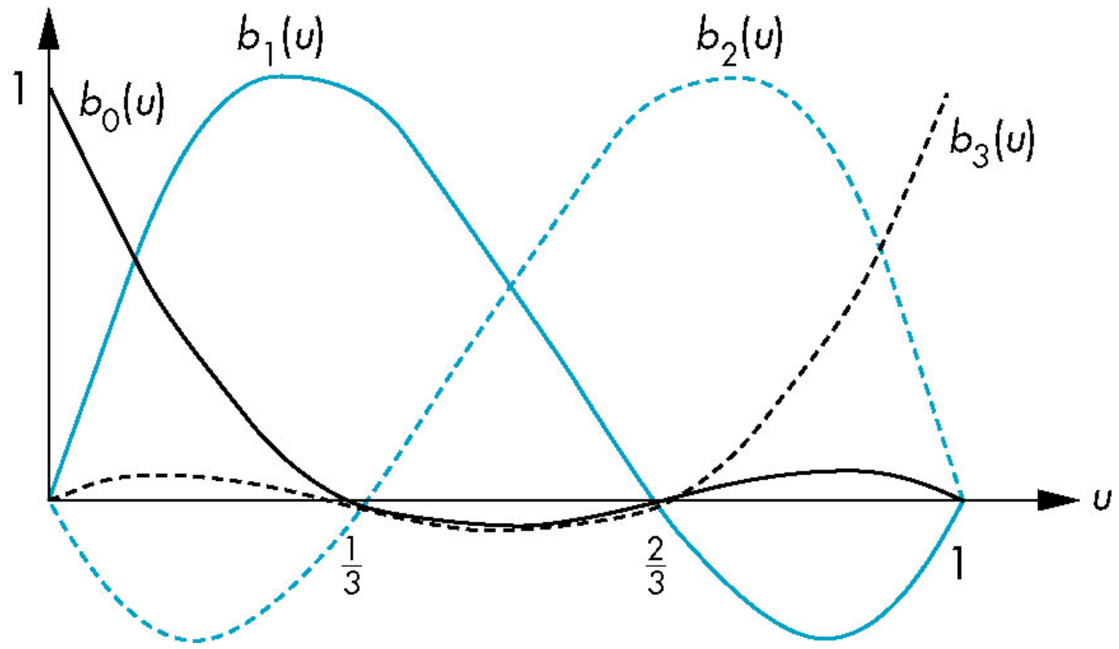
$$p(u)=\mathbf{u}^T\mathbf{c}=\mathbf{u}^T\mathbf{M}_I\mathbf{p} = \mathbf{b}(u)^T\mathbf{p}$$

where $b(u) = [b_0(u)\ b_1(u)\ b_2(u)\ b_3(u)]^T$ is an array of *blending polynomials* such that $p(u) = b_0(u)p_0 + b_1(u)p_1 + b_2(u)p_2 + b_3(u)p_3$

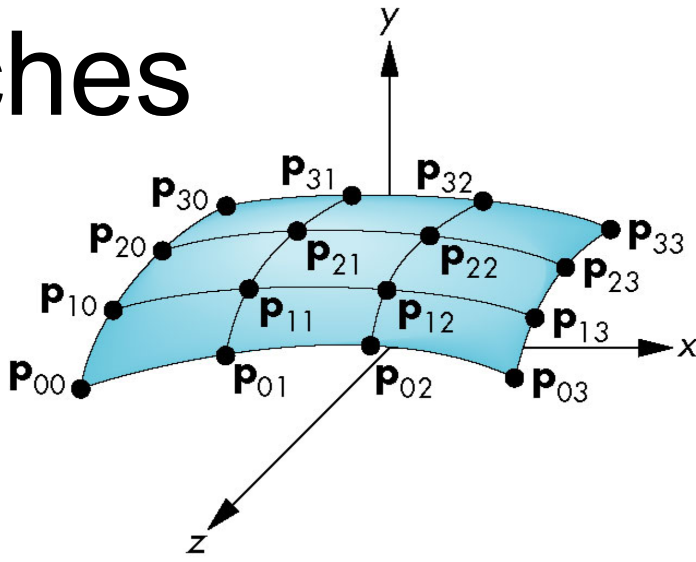
$$b_0(u) = -4.5(u-1/3)(u-2/3)(u-1)$$
$$b_1(u) = 13.5u\ (u-2/3)(u-1)$$
$$b_2(u) = -13.5u\ (u-1/3)(u-1)$$
$$b_3(u) = 4.5u\ (u-1/3)(u-2/3)$$

Blending Functions





Blending Patches



$$p(u,v) = \sum_{i=o}^{3} \sum_{j=0}^{3} c_{ij} u^i v^j$$

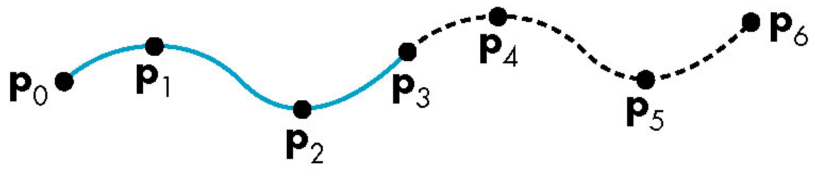
$$p(u,v) = \sum_{i=o}^{3} \sum_{j=0}^{3} b_i(u) b_j(v) p_{ij} = u^T \mathbf{M}_I \mathbf{P} \mathbf{M}_I^T v$$

Each $b_i(u)b_j(v)$ is a blending patch

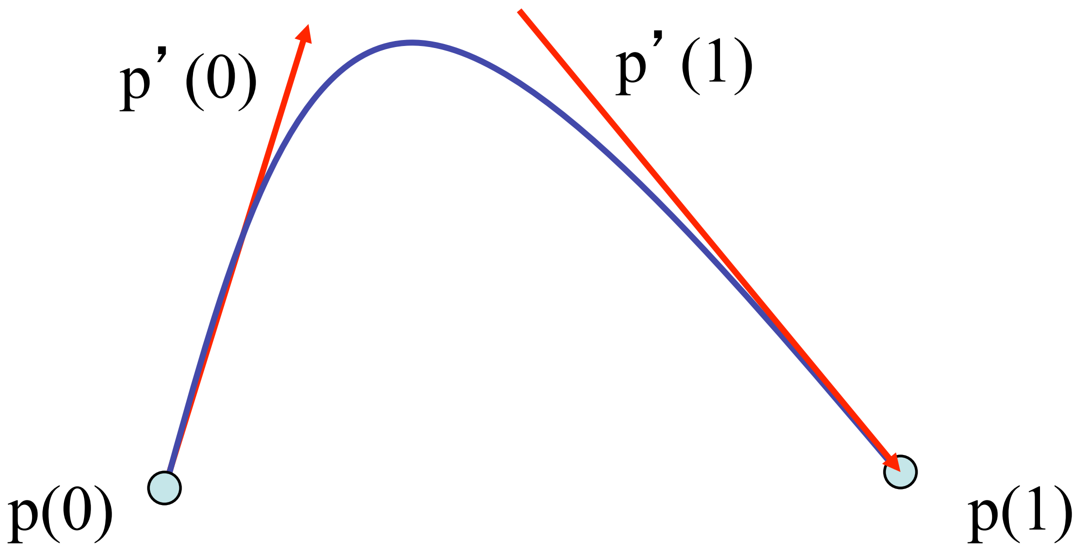
Shows that we can build and analyze surfaces from our knowledge of curves

Hermite Curves and Surfaces

- How can we get around the limitations of the interpolating form
  - –Lack of smoothness
  - –Discontinuous derivatives at join points
- We have four conditions (for cubics) that we can apply to each segment
  - –Use them other than for interpolation
  - –Need only come close to the data



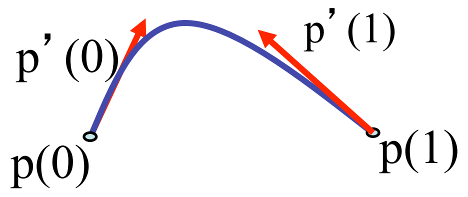
Hermite Form



Use two interpolating conditions and two derivative conditions per segment

Ensures continuity and first derivative continuity between segments

Equations



$p(u) = c_0+uc_1+u^2c_2+u^3c_3$

Interpolating conditions are the same at ends

$$p(0) = p_0 = c_0$$
$$p(1) = p_1 = c_0+c_1+c_2+c_3$$

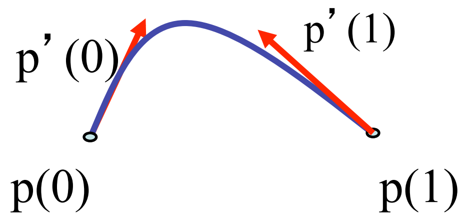
Differentiating we find $p'(u) = c_1+2uc_2+3u^2c_3$

Evaluating at end points

$$p'(0) = p'_0 = c_1$$
$$p'(1) = p'_1 = c_1+2c_2+3c_3$$

$$\mathbf{q} = \begin{bmatrix} p_0 \\ p_1 \\ p'_0 \\ p'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c}$$

Matrix Form



$$\mathbf{q} = \begin{bmatrix} p_0 \\ p_3 \\ p'_0 \\ p'_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \mathbf{c}$$

Solving, we find $\mathbf{c}=\mathbf{M}_H\mathbf{q}$ where $\mathbf{M}_H$ is the Hermite matrix

$$\mathbf{M}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

$p(u) = u^T\mathbf{c}$ =>

$p(u) = u^T\mathbf{M}_H\mathbf{q}$

Blending Polynomials

$$p(u) = \mathbf{b(u)}^T\mathbf{q}$$

$$p(u) = u^T\mathbf{M}_H\mathbf{q}$$

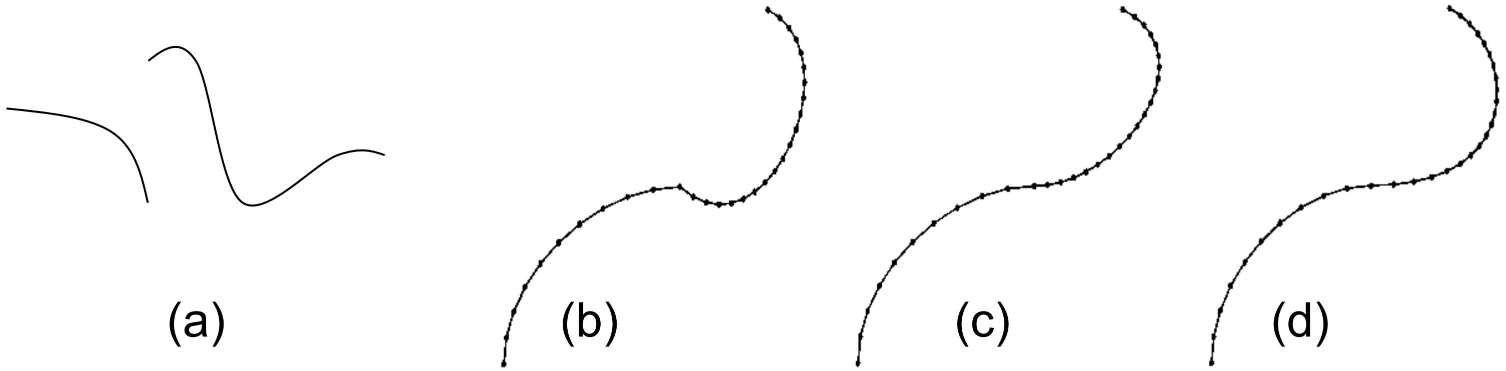
$$\mathbf{b}(u) = \begin{bmatrix} 2u^3 - 3u^2 + 1 \\ -2u^3 + 3u^2 \\ u^3 - 2u^2 + u \\ u^3 - u^2 \end{bmatrix}$$

$$\mathbf{M}_H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix}$$

Although these functions are smooth, the Hermite form is not used directly in Computer Graphics and CAD because we usually have control points but not derivatives

However, the Hermite form is the basis of the Bezier form

Continuity



- A) Non-continuous
- B) C^0-continuous
- C) G^1-continuous
- D) C^1-continuous
- (C^2-continuous)

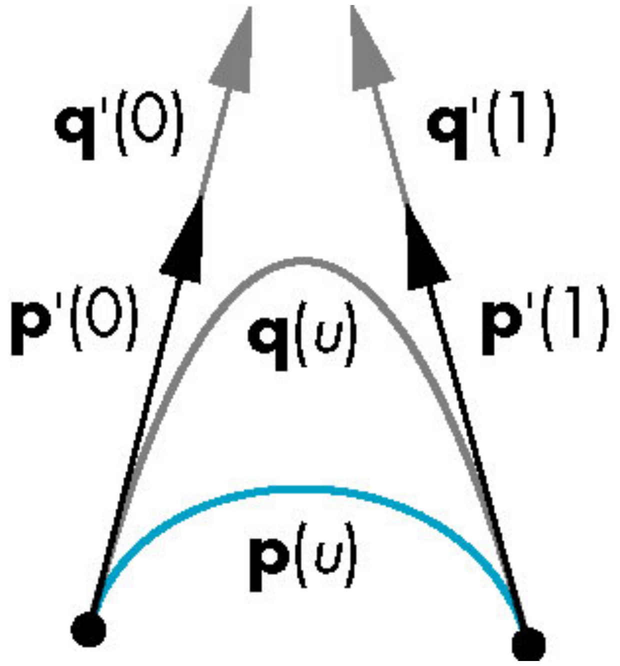
See page 585-587 in Real-time Rendering, 3rd edition.

Example

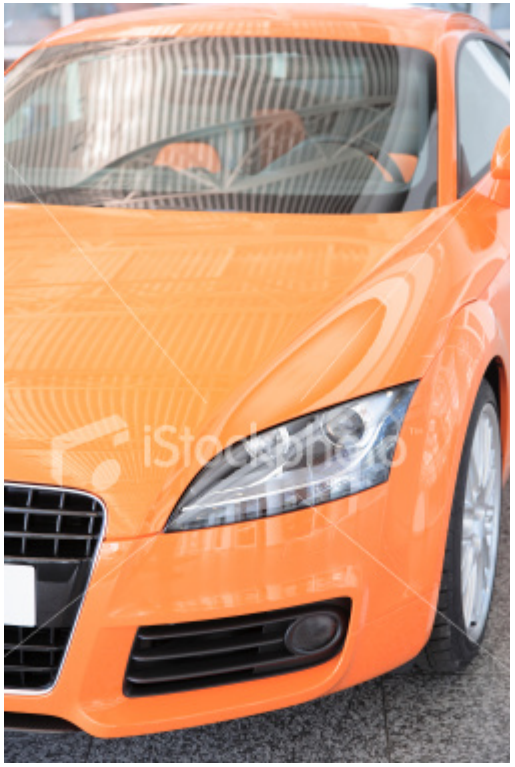
- Here the p and q have the same tangents at the ends of the segment but different derivatives
- Generate different

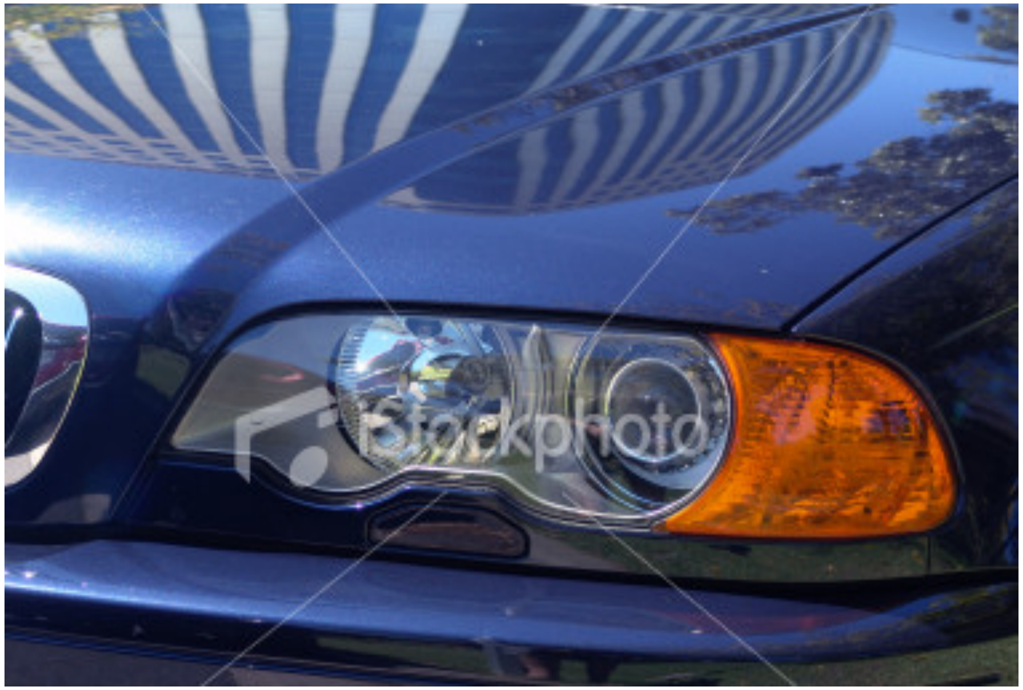
  Hermite curves
- This techniques is used

in drawing applications



Reflections should be at least C^1





Bezier Curves

- In graphics and CAD, we do not usually have derivative data
- Bezier suggested using the same 4 data points as with the cubic interpolating curve to approximate the derivatives in the Hermite form

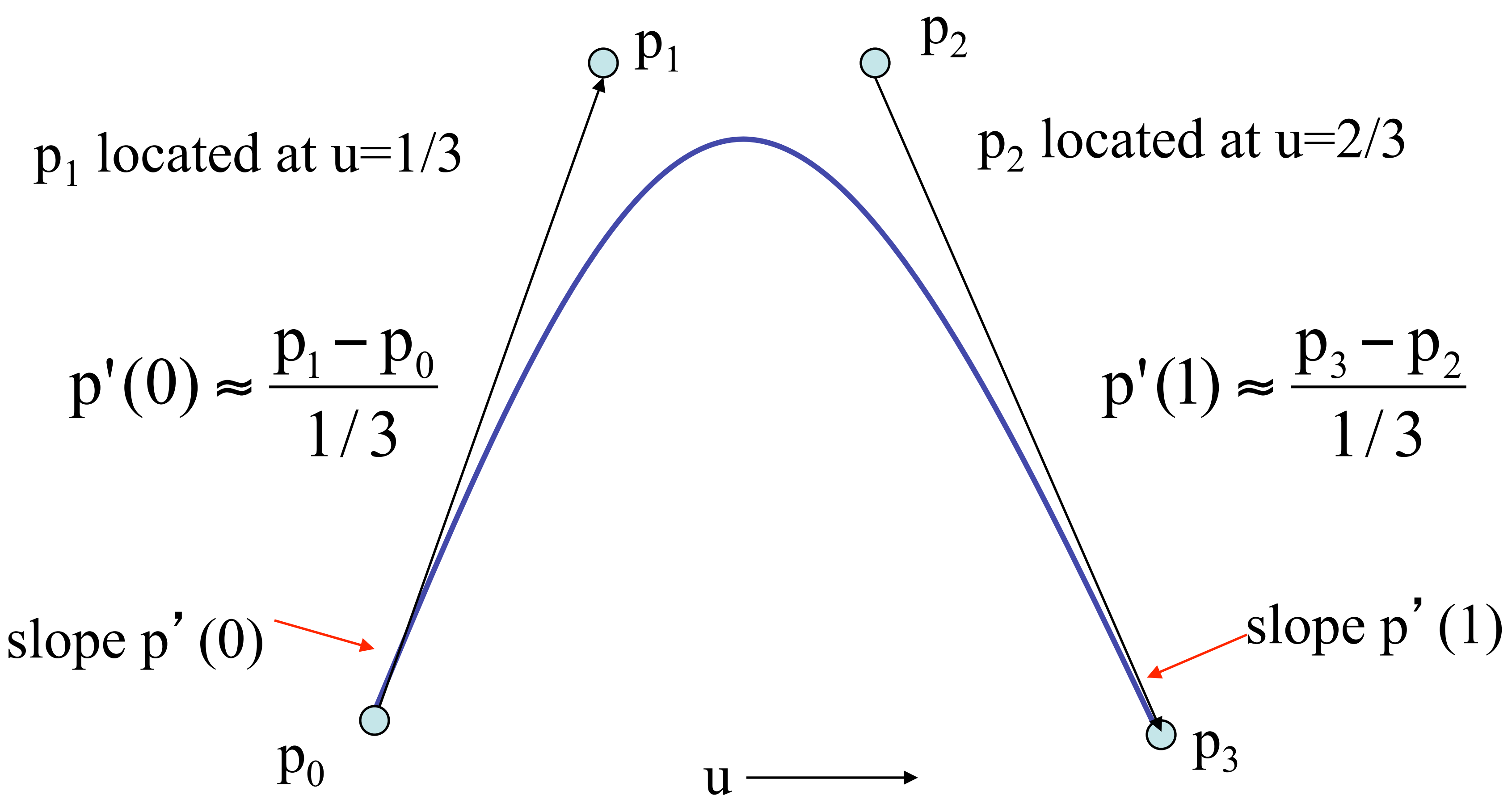
Approximating Derivatives

p_1 located at u=1/3

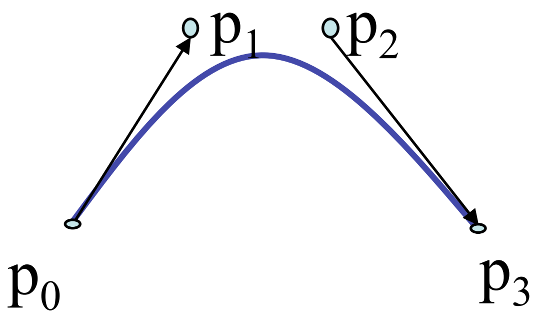
p_2 located at u=2/3

$$p'(0) \approx \frac{p_1 - p_0}{1/3}$$

$$p'(1) \approx \frac{p_3 - p_2}{1/3}$$



Equations



$p(u) = c_0+uc_1+u^2c_2+u^3c_3$

$p'(u) = c_1+2uc_2+3u^2c_3$

Interpolating conditions are the same

$$p(0) = p_0 = c_0$$
$$p(1) = p_3 = c_0+c_1+c_2+c_3$$

Approximating derivative conditions

$$\left.\begin{array}{l} p'(0) \approx \dfrac{p_1 - p_0}{1/3} \\ p'(1) \approx \dfrac{p_3 - p_2}{1/3} \end{array}\right]$$

$$p'(0) = 3(p_1 - p_0) = c_1$$
$$p'(1) = 3(p_3 - p_2) = c_1+2c_2+3c_3$$

Solve four linear equations for $\mathbf{c}=\mathbf{M}_B\mathbf{p}$

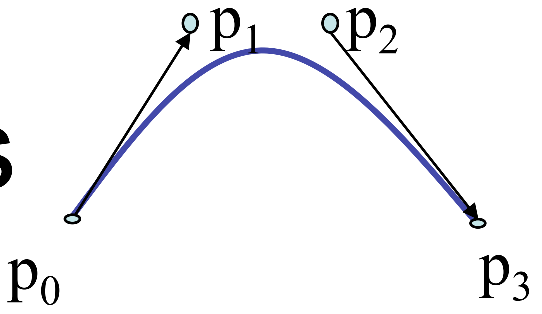
Bezier Matrix

$$\mathbf{M}_B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

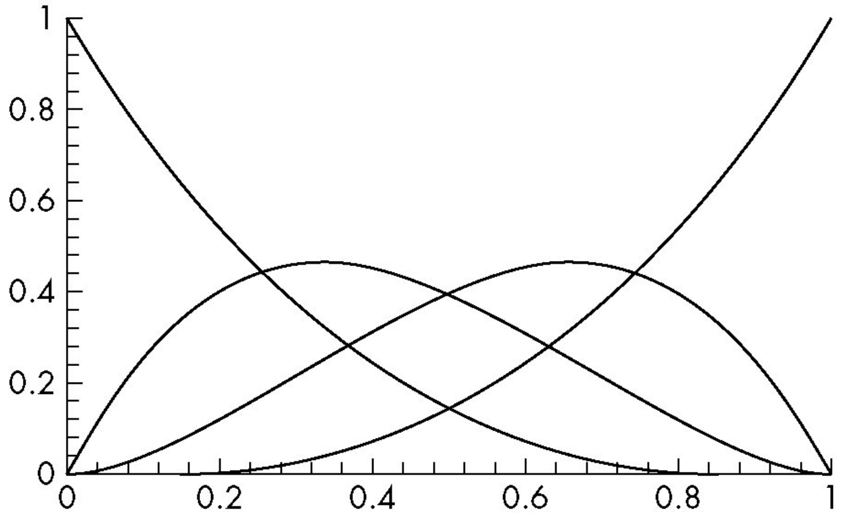
$$p(u) = \mathbf{u}^T \mathbf{M}_B \mathbf{p} = \mathbf{b(u)}^T \mathbf{p}$$

blending functions

Blending Functions



$$\mathbf{b}(u) = \begin{bmatrix} (1-u)^3 \\ 3u(1-u)^2 \\ 2u^2(1-u) \\ u^3 \end{bmatrix}$$

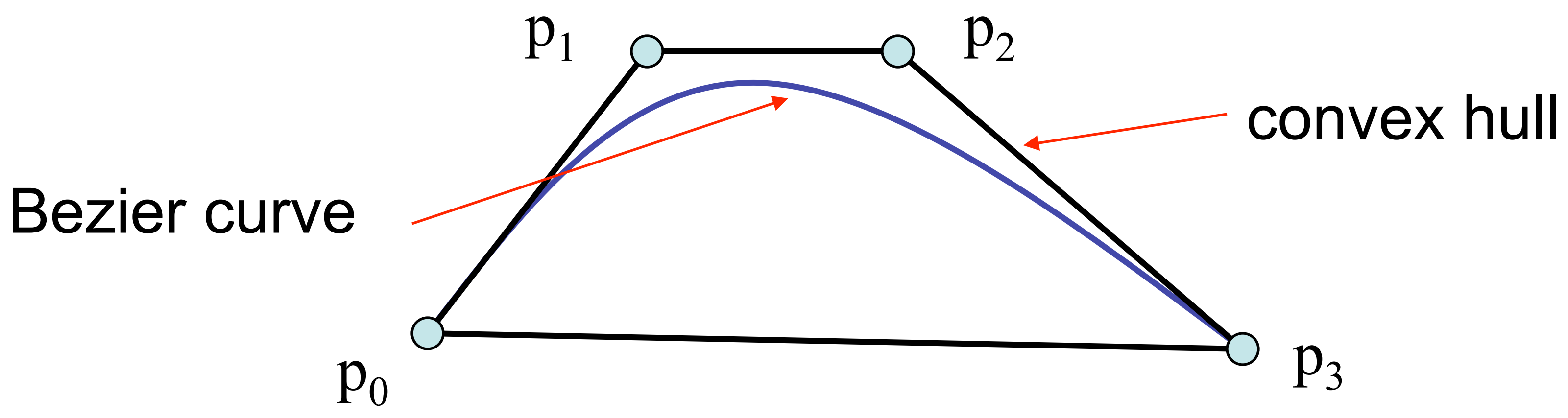


Note that all zeros are at 0 and 1 which forces the functions to be smoother over (0,1)

Smoother because the curve stays inside the convex hull, and therefore does not have room to fluctuate so much.

Convex Hull Property

- All weights within [0,1] and sum of all weights = 1 (at given u) ensures that all Bezier curves lie in the convex hull of their control points
- Hence, even though we do not interpolate all the data, we cannot be too far away

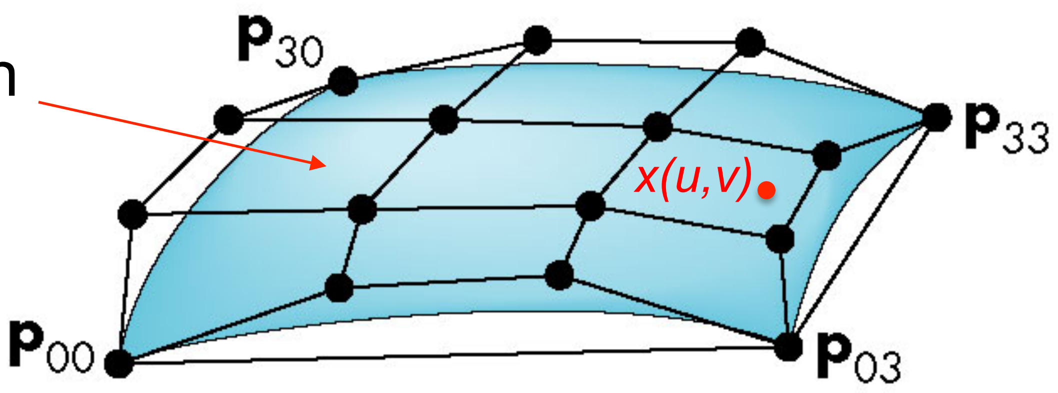


Bezier Patches

Using same data array $\mathbf{P}=[p_{ij}]$ as with interpolating form

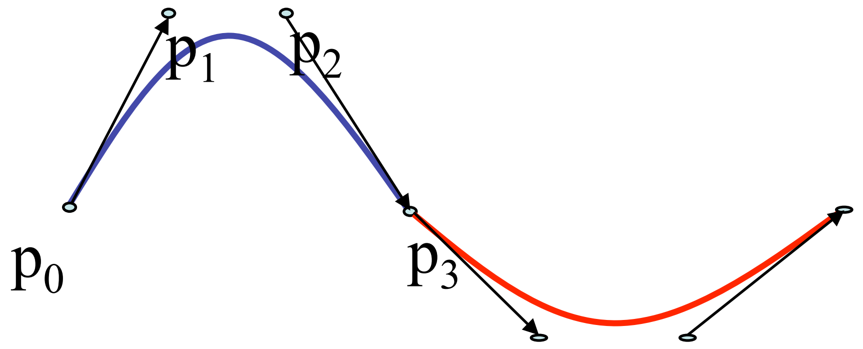
$$\underset{x(u,v)}{p(u,v)} = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\, b_j(v)\, p_{ij} = u^T \mathbf{M}_B \underset{x}{\mathbf{P}} \mathbf{M}_B^T v$$

Patch lies in convex hull



Analysis

- Although the Bezier form is much better than the interpolating form, the derivatives are not continuous at join points



- What shall we do to solve this?

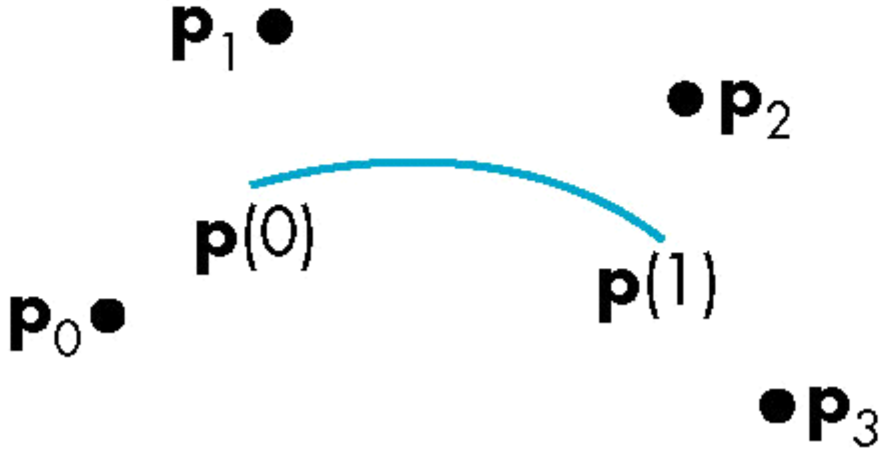
B-Splines

- Basis splines: use the data at
  $\mathbf{p}=[p_{i-2}\ p_{i-1}\ p_i\ p_{i+1}]^T$ to define curve only between p_{i-1} and p_i
- Allows us to apply more continuity conditions to each segment
- For cubics, we can have continuity of the function, and first and second derivatives at join points

Cubic B-spline

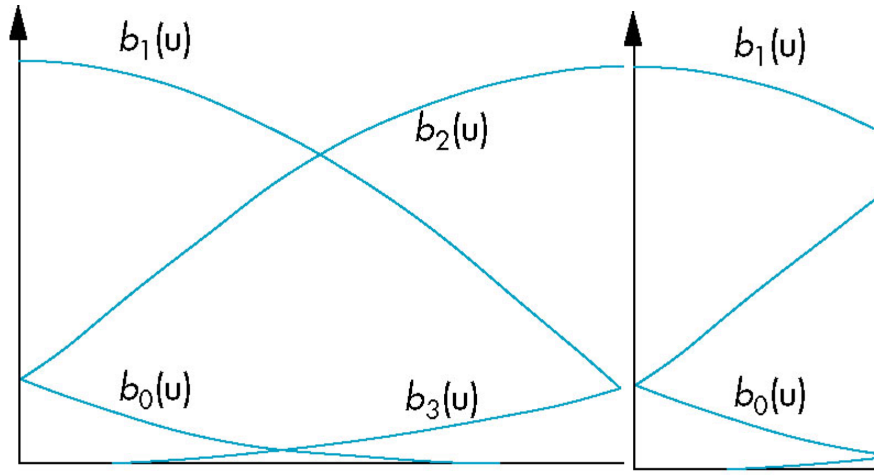
$$\mathrm{p}(\mathrm{u}) = \mathbf{u}^{\mathrm{T}}\mathbf{M}_{S}\mathbf{p} = \mathbf{b(u)}^{\mathrm{T}}\mathbf{p}$$

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

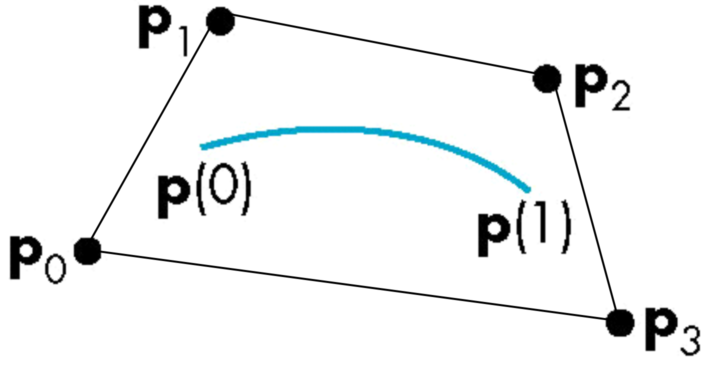


Blending Functions

$$\mathbf{b}(u) = \frac{1}{6}\begin{bmatrix} (1-u)^3 \\ 4-6u^2+3u^3 \\ 1+3u+3u^2-3u^3 \\ u^3 \end{bmatrix}$$



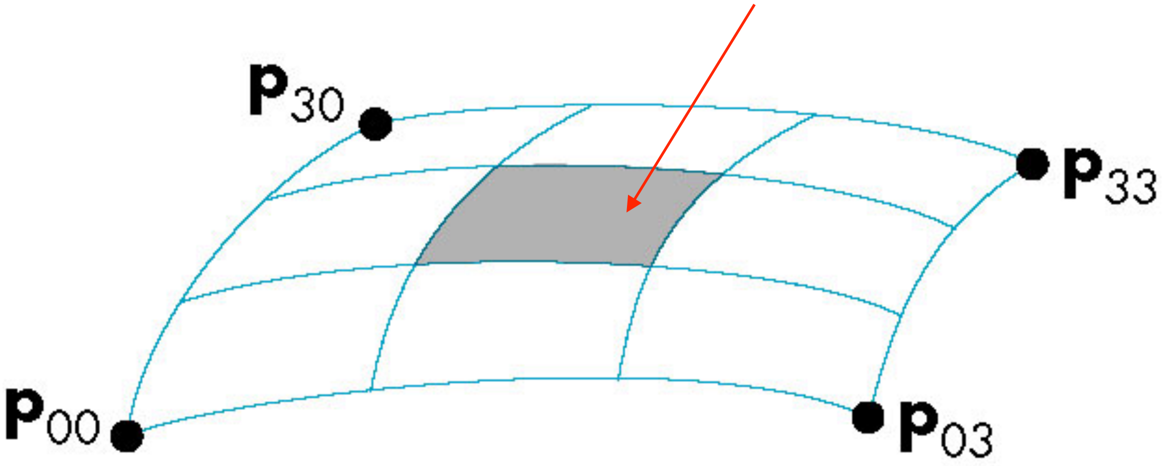
convex hull property



B-Spline Patches

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} b_i(u)\, b_j(v)\, p_{ij} = u^T \mathbf{M}_S \mathbf{P} \mathbf{M}_S^T v$$

defined over only 1/9 of region



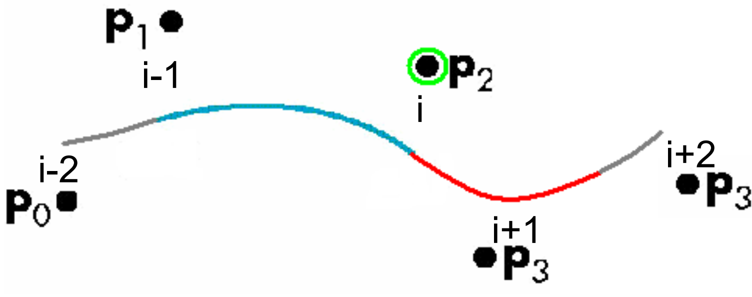
Splines and Basis

- If we examine the cubic B-spline from the perspective of each control (data) point, each interior point contributes (through the blending functions) to four segments
- We can rewrite p(u) in terms of the data points as

$$p(u) = \sum B_i(u)\, p_i$$

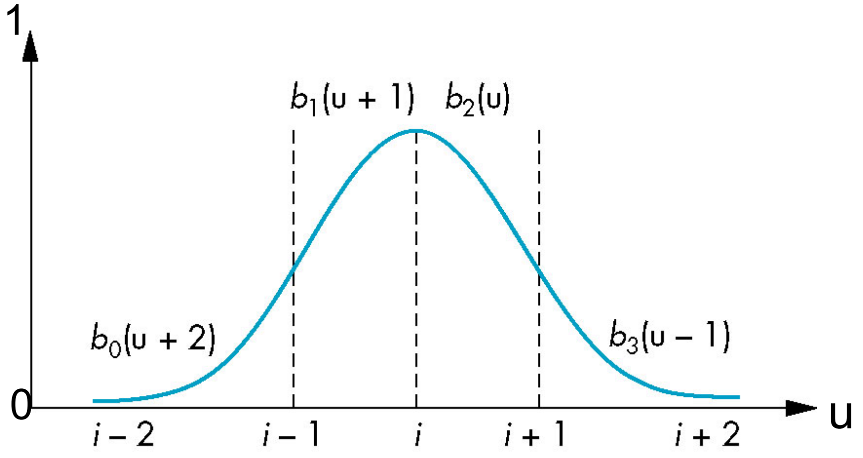
defining the basis functions $\{B_i(u)\}$

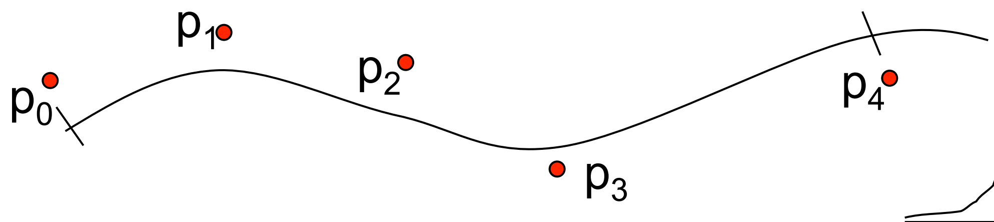
Basis Functions

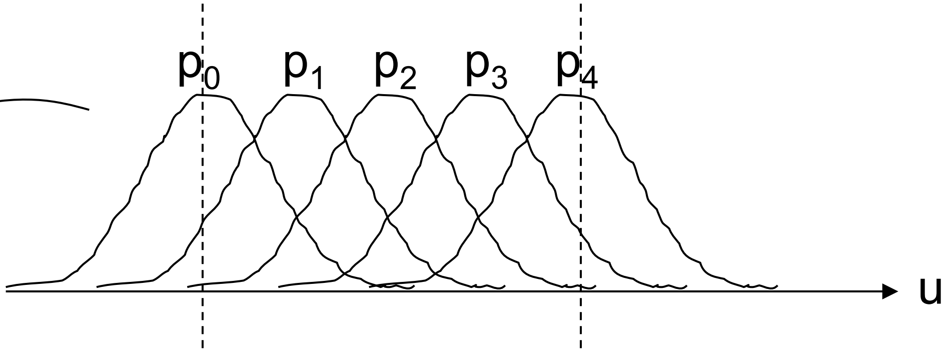


In terms of the blending polynomials

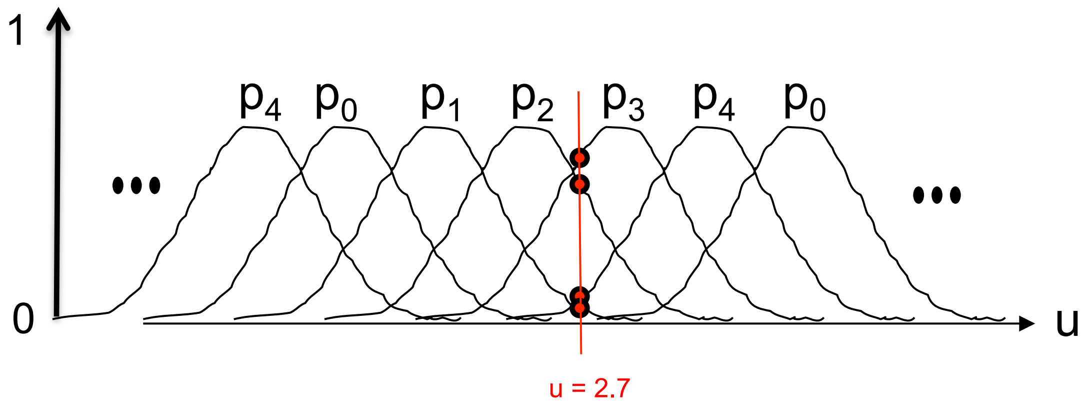
$$B_i(u) = \begin{cases} 0 & u < i-2 \\ b_0(u+2) & i-2 \le u < i-1 \\ b_1(u+1) & i-1 \le u < i \\ b_2(u) & i \le u < i+1 \\ b_3(u-1) & i+1 \le u < i+2 \\ 0 & u \ge i+2 \end{cases}$$

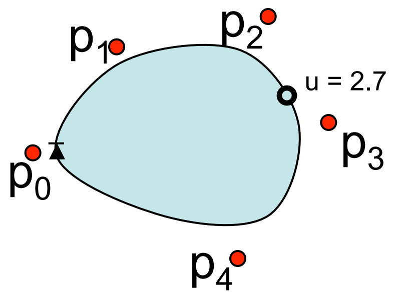






One more example





$$p(u) = B_0(u)p_0 + B_1(u)p_1 + B_2(u)p_2 + B_3(u)p_3 + B_4(u)p_4$$

I.e.,: $p(u) = \sum B_i(u)\, p_i$

SUMMARY

B-Splines

These are our control points, p_0-p_8, to which we want to approximate a curve

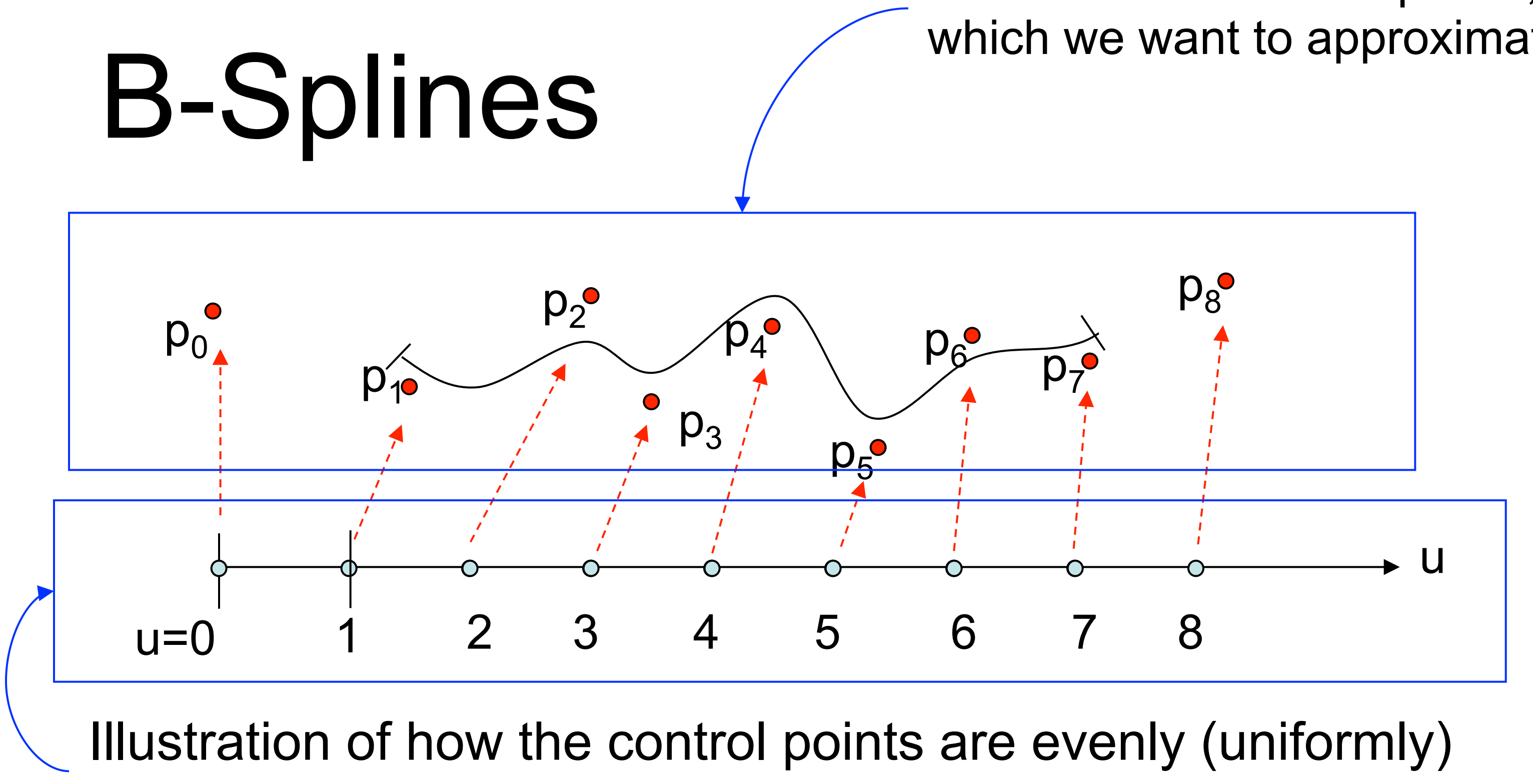
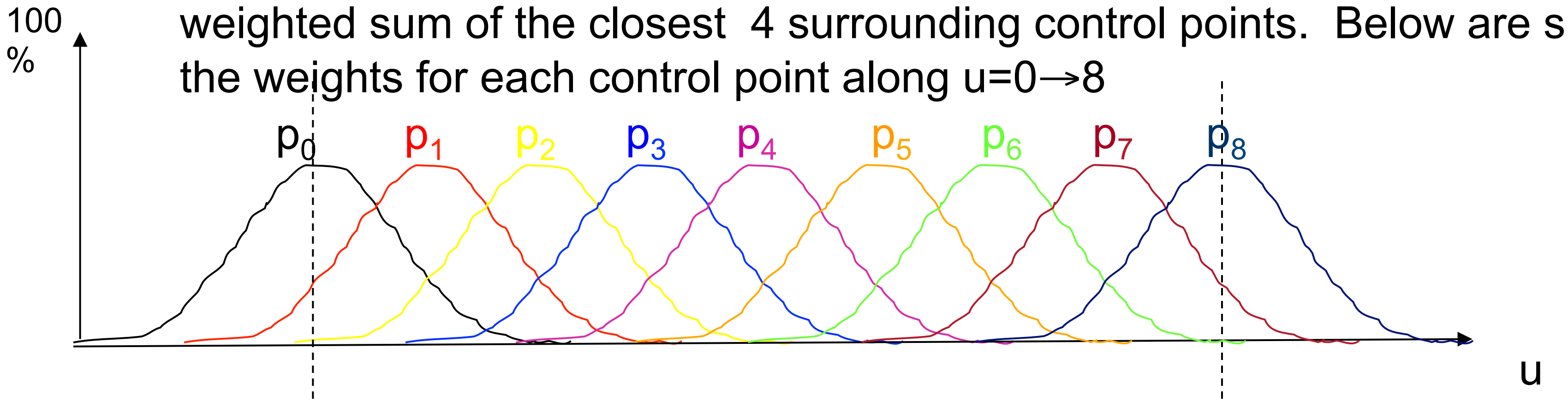


Illustration of how the control points are evenly (uniformly) distributed along the parameterisation u of the curve p(u).

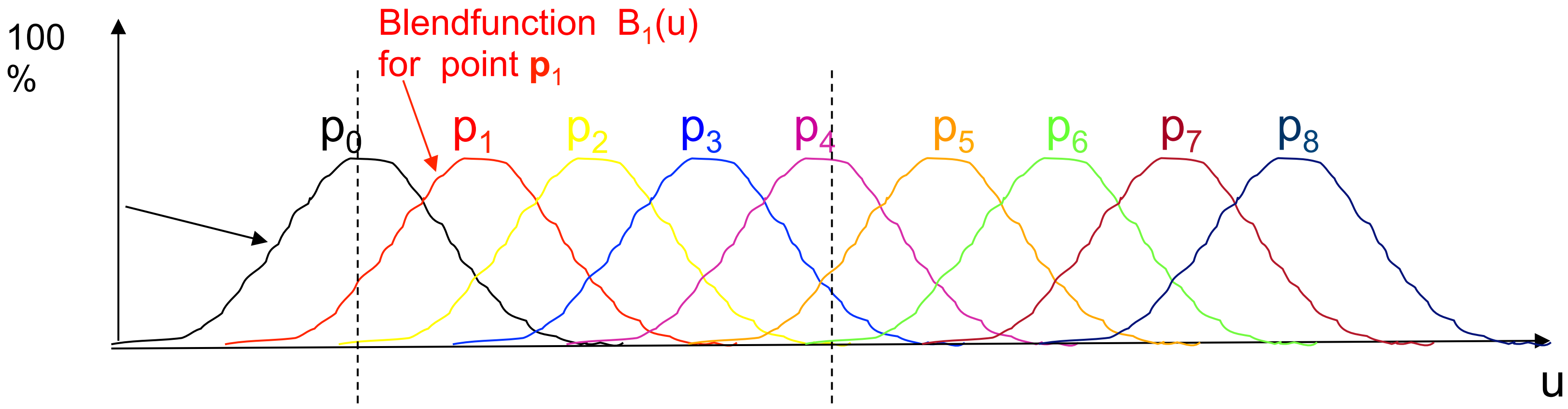
In each point p(u) of the curve, for a given u, the point is defined as a weighted sum of the closest 4 surrounding control points. Below are shown the weights for each control point along u=0→8



SUMMARY

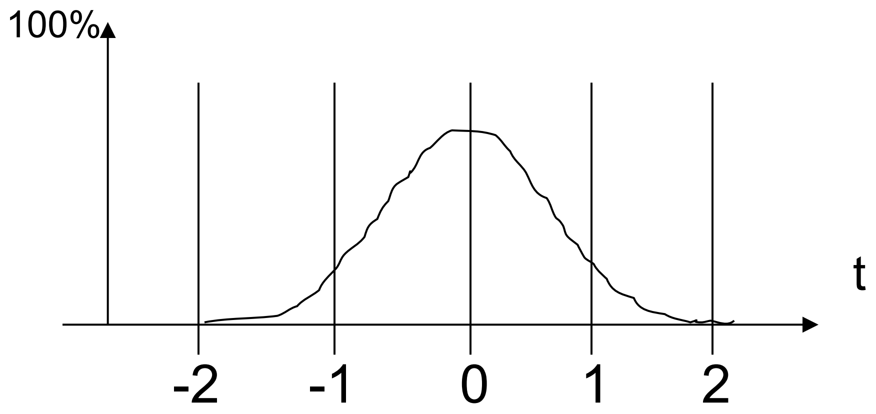
B-Splines

In each point p(u) of the curve, for a given u, the point is defined as a weighted sum of the closest 4 surrounding points. Below are shown the weights for each point along u=0→8



The weight function (blend function) $B_{\mathbf{p}i}$ (u) for a point $\mathbf{p}_i$ can thus be written as a translation of a basis function B(t). $B_{\mathbf{p}i}(u) = B(u-i)$

B(t):



Our complete B-spline curve p(u) can thus be written as:

$$p(u) = \sum B_i(u)\, p_i$$

Generalizing Splines

- We can extend to splines of any degree
- Data and conditions do not have to be given at equally spaced values (the *knots*)
  - –Nonuniform and uniform splines
  - –Can have repeated knots
    - Can force spline to interpolate points
- (Cox-deBoor recursion gives method of evaluation (also known as deCasteljau-recursion, see page 579, RTR 3:rd Ed. for details))

DEMO of B-Spline curve: (make duplicate knots)

NURBS

- Nonuniform Rational B-Spline curves and surfaces add a fourth variable w to x,y,z
  - –Can interpret as weight to give more importance to some control data
  - –Can also interpret as moving to homogeneous coordinate

- (Requires a perspective division
  - –NURBS act correctly for perspective viewing
- Quadrics are a special case of NURBS)

NURBS

NURBS is similar to B-Splines except that:

1. The control points can have different weights, w_i, (heigher weight makes the curve go closer to that control point)
2. The control points do not have to be at uniform distances (u=0,1,2,3...) along the parameterisation u. E.g.: u=0, 0.5, 0.9, 4, 14,…

NURBS = Non-Uniform Rational B-Splines

The NURBS-curve is thus defined as:

$$\mathbf{p}(u) = \frac{\sum_{i=0}^{n} B_i(u)\, w_i\, \mathbf{p}(i)}{\boxed{\sum_{i=0}^{n} B_i(u)\, w_i}}$$

Division with the sum of the weights, to make the combined weights sum up to 1, at each position along the curve. Otherwise, a translation of the curve is introduced (which is not desirable).

NURBS

- Concider a control point in 3 dimensions:

$$\mathbf{p}_i = \left[x_i, y_i, z_i\right]$$

- The weighted homogeneous-coordinate is:

$$\mathbf{q}_i = w_i \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}$$

- The idea is to use the weights w_i to increase or decrease the importance of a particular control point

NURBS

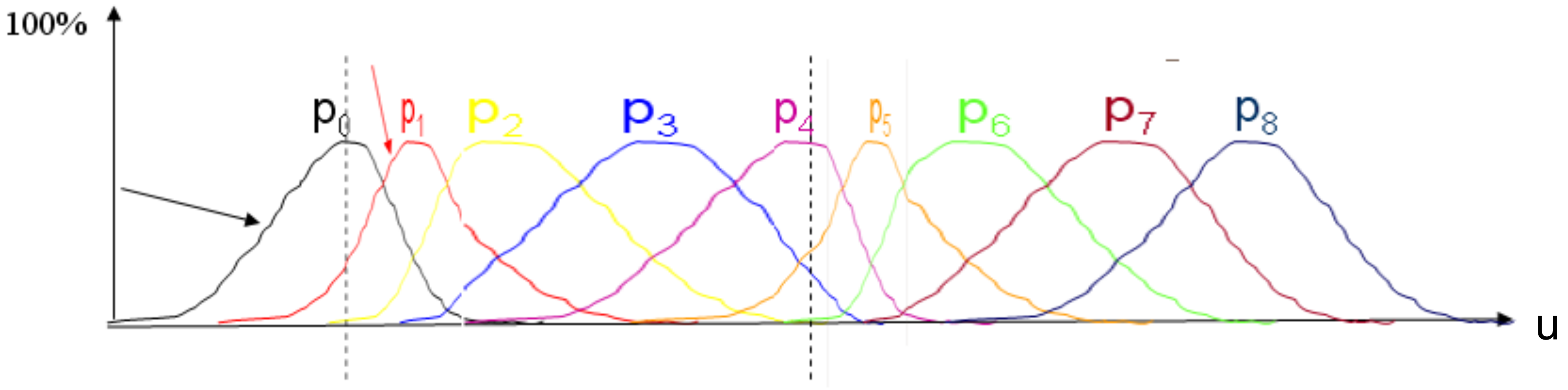
- The w-component may not be equal to 1.
- Thus we must do a perspective division to get the three-dimensional points:

$$\mathbf{p}(u) = \frac{1}{w(u)}\mathbf{q}(u) = \frac{\sum_{i=0}^{n} B_{i,d} w_i \mathbf{p}(i)}{\sum_{i=0}^{n} B_{i,d} w_i}$$

- Each component of **p**(u) is now a rational function in *u,* and because we have not restricted the knots (the knots does not have to be uniformly distributed), we have derived a **nonuniform rational B-spline (NURBS)** curve

NURBS

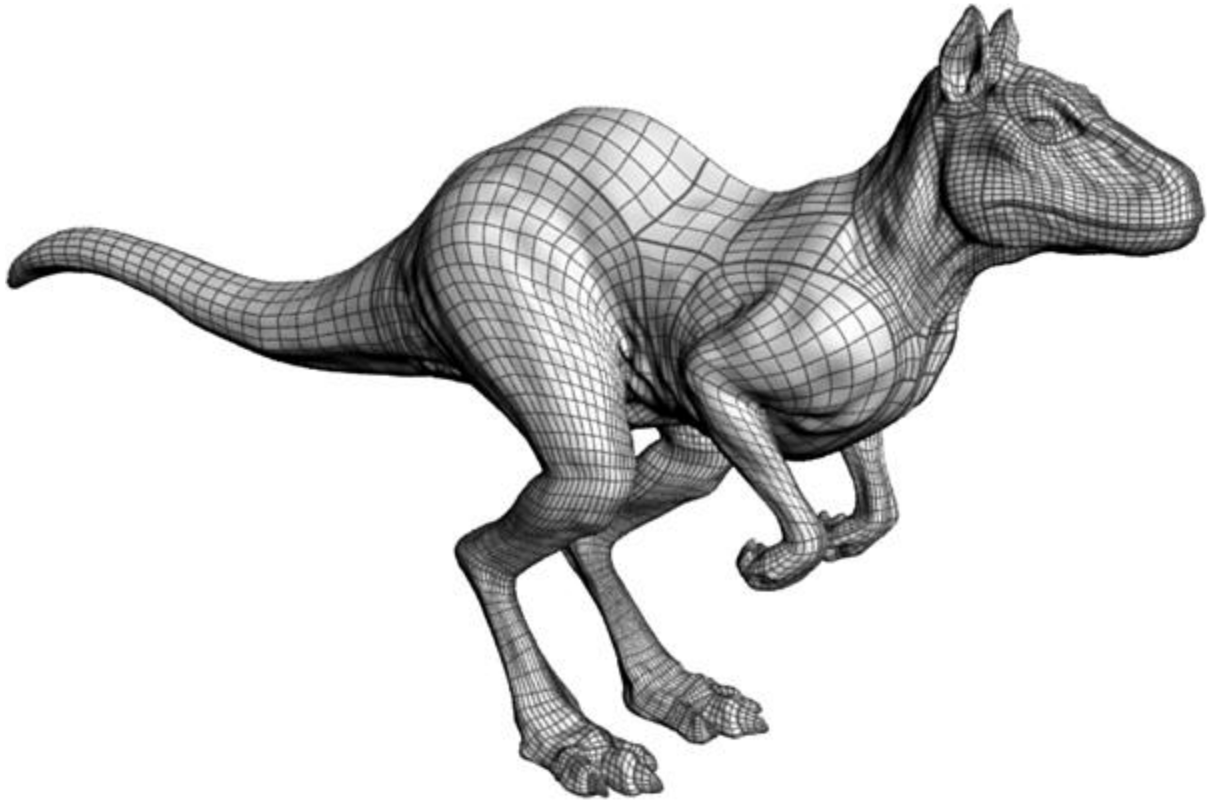
- Allowing control points at non-uniform distances means that the basis functions $B_{pi}()$ are being streched and non-uniformly located.
- E.g.:

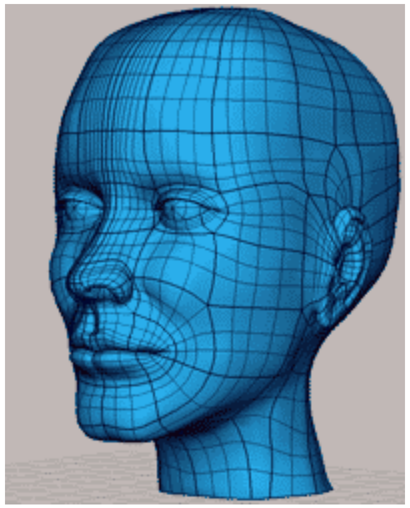


Each curve $B_{pi}()$ should of course look smooth and C^2 –continuous. But it is not so easy to draw smoothly by hand...

(The sum of the weights are still =1 due to the division in previous slide)

NURBS Surfaces - examples





NURBS

- If we apply an affine transformation to a B-spline curve or surface, we get the same function as the B-spline derived from the transformed control points.
- Because perspective transformations are not affine, most splines will not be handled correctly in perspective viewing.
- However, the perspective division embedded in the NURBS ensures that NURBS curves are handled correctly in perspective views.
- Quadrics can be shown to be a special case of quadratic NURBS curve; thus, we can use a single modeling method, NURBS curves, for the most widely used curves and surfaces

THE END