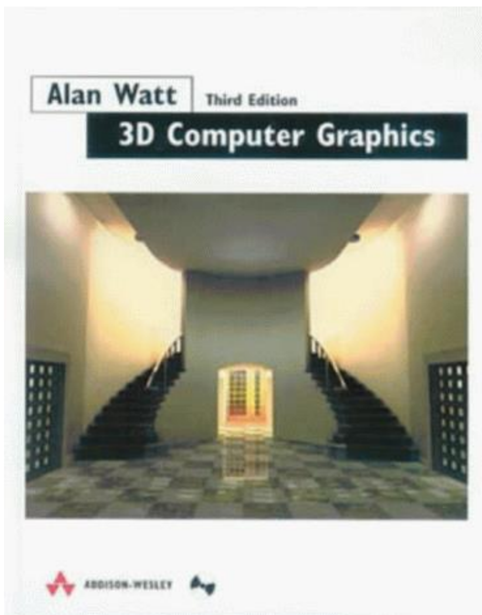
The University Of Sheffield.

COM3503/4503/6503: 3D Computer Graphics

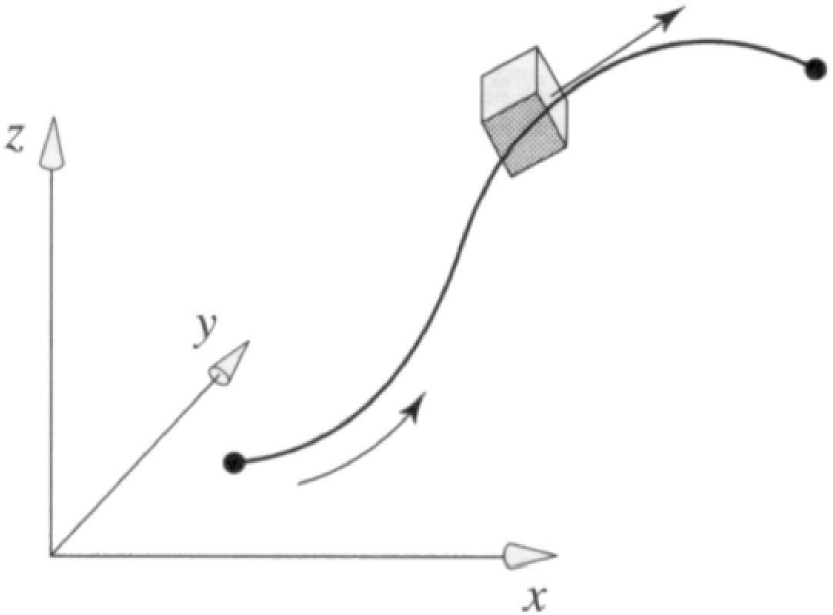
Lecture 14: Parametric curves



Dr. Steve Maddock
Room G011, Regent Court
s.maddock@sheffield.ac.uk

1. Introduction

- Parametric curves are used in many applications:
  - Drawing packages;
  - Scaleable fonts (outline fonts);
  - Flight paths;
  - Inbetween paths
- For animation, we can use the curve to give the value of a parameter (e.g. x position or y position or z position or other) as it changes over time, i.e. as we step along the curve

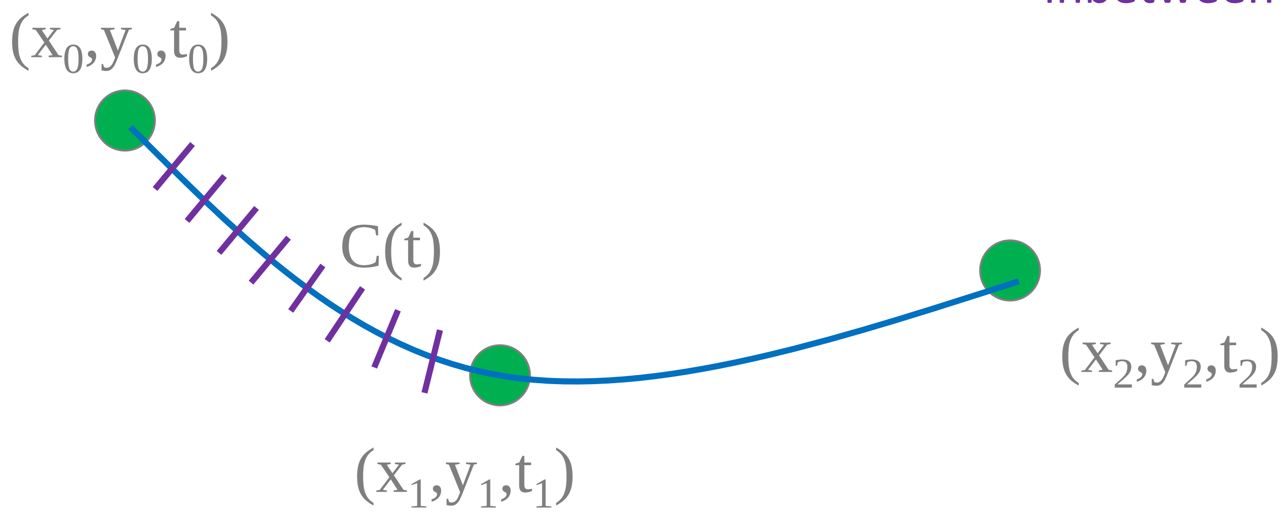


1. Introduction

- The parametric curve may need to be generated from limited information values – use interpolation
- Example:
  - Given positions: (x_i, y_i, t_i), i = 0,...,n
  - Find curve $C(t) = (x(t), y(t))$ such that $C(t_i) = (x_i, y_i)$

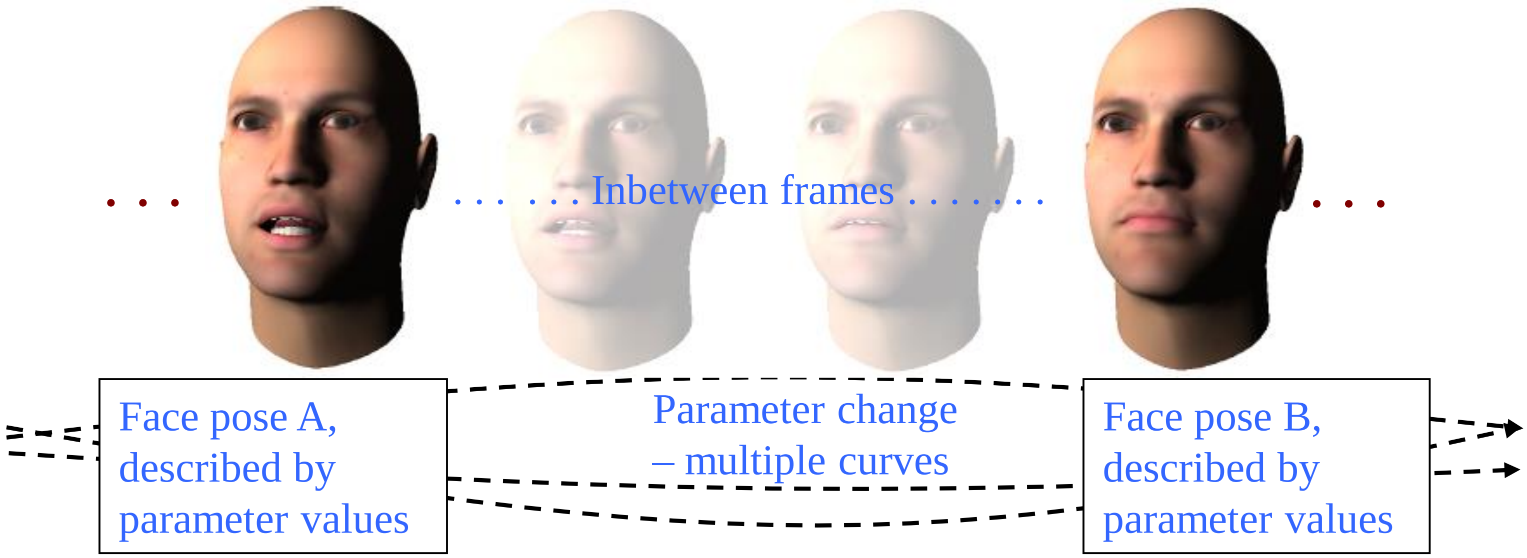
[keyframes]

[use curve to generate inbetween positions]



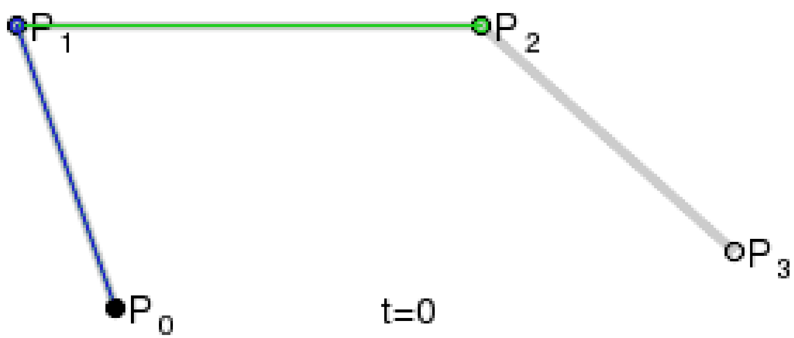
1. Introduction

- A parametric curve can be used to control any parameter as it changes over time
  - E.g. camera position, object size, object colour, joint angles for a hierarchical object, facial poses



1. Introduction

- Lots of different kinds of parametric curve
  - Hermite, Bezier, Catmull-rom, B-splines, non-uniform rational B-splines (NURBS), …
- Join segments together to create longer curves
  - E.g. piecewise cubic splines
- We'll focus on cubic Bezier curves, which can be used to interpolate keyframe values



http://en.wikipedia.org/wiki/Bezier_curves

2. Lines

- Parametric form

  $Q(t) = at + b$

- As three components (3D):

  $x(t) = a_x t + b_x$
  $y(t) = a_y t + b_y$
  $z(t) = a_z t + b_z$

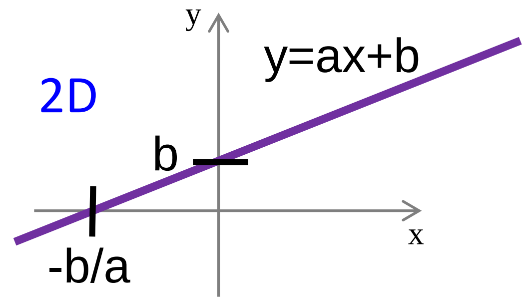
- Rearrange for intuitive control:

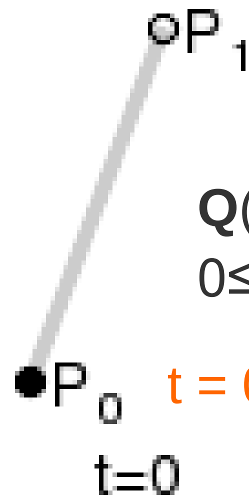
  $\mathbf{Q}(t) = \mathbf{P}_0(1-t) + \mathbf{P}_1 t \quad 0 \le t \le 1$

- where

  $a = P_1 - P_0$

  $b = P_0$





$t = 1;\ P_1 = (x_1, y_1, z_1)$

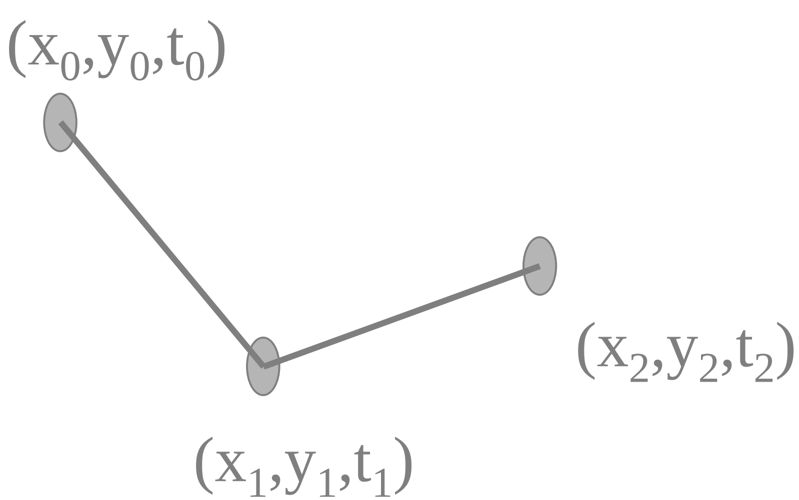
$\mathbf{Q}(t) = \mathbf{P}_0(1-t) + \mathbf{P}_1 t$
$0 \le t \le 1$

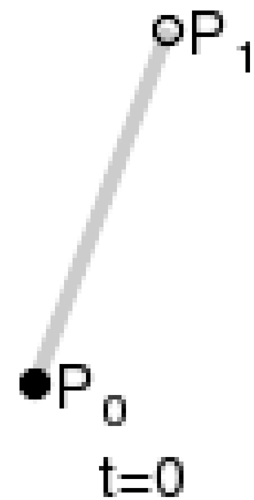
$t = 0;\ P_0 = (x_0, y_0, z_0)$

http://en.wikipedia.org/wiki/Bezier_curves

2.1 Linear interpolation

- For first two points, assuming $t_0 = 0$ and $t_1 = 1$
- X coordinate: $x(t) = x_0(1-t) + x_1 t$





$\mathbf{Q}(t) = \mathbf{P}_0(1-t) + \mathbf{P}_1 t \quad 0 \le t \le 1$

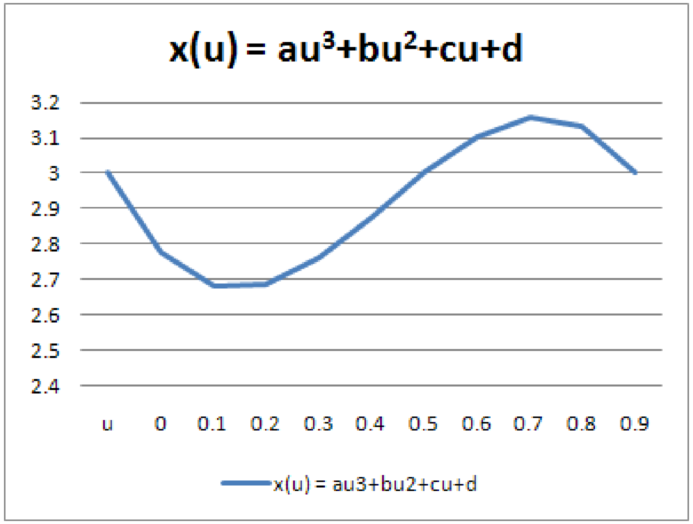
http://en.wikipedia.org

3. General parametric curves

- Linear $Q(u) = au + b$
- Quadratic $Q(u) = au^2 + bu + c$
- Cubic $Q(u) = au^3 + bu^2 + cu + d$

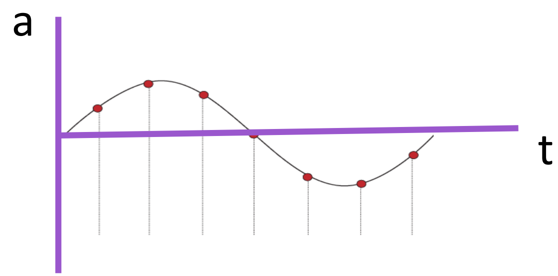
$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

- Plot x(u), 0<=u<=1

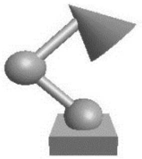


a	b	c	d	
-5	8	-3	3	
				x(u) =
	u	u^2	u^3	au^3+bu^2+cu+d
	0	0	0	3
	0.1	0.01	0.001	2.775
	0.2	0.04	0.008	2.68
	0.3	0.09	0.027	2.685
	0.4	0.16	0.064	2.76
	0.5	0.25	0.125	2.875
	0.6	0.36	0.216	3
	0.7	0.49	0.343	3.105
	0.8	0.64	0.512	3.16
	0.9	0.81	0.729	3.135
	1	1	1	3

3.1 Useful parametric curves



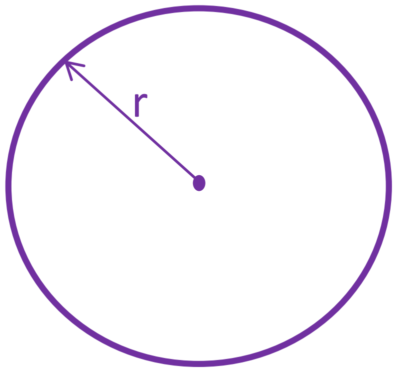
- A sine wave: `a = sin(t)`
  - For 0 <= t <= 1,
    scale to 0 <= t <= 2π,
    `sin(t)` gives repeating, smoothly-varying `a`
    between -1 and +1



- Use this to control, say, change of an angle between a maximum and a minimum value

- A circle:

$x = f(t) = r \sin(t)$
$y = g(t) = r \cos(t)$
$0 <= t <= 2\pi$



- (Circle with centre (a,b): $(x-a)^2 + (y-b)^2 = r^2$)

4. Bezier cubic curves

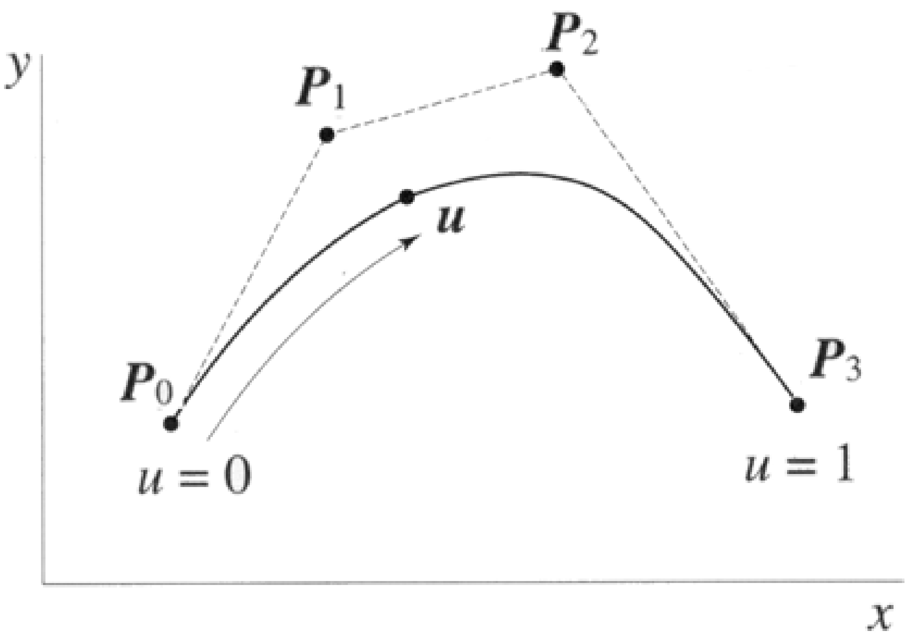
$$Q(u) = au^3 + bu^2 + cu + d$$

- Rearrange general cubic parametric curve:

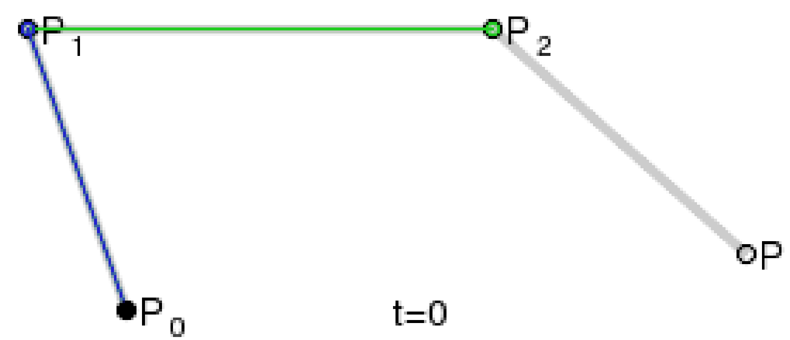
  $$\mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + \mathbf{P}_1 3u(1-u)^2 + \mathbf{P}_2 3u^2(1-u) + \mathbf{P}_3 u^3$$

- $\mathbf{P}_0$ and $\mathbf{P}_3$ are the endpoints of the curve
- $\mathbf{P}_1$ and $\mathbf{P}_2$ control the shape of the curve

$a=-\mathbf{P_0}+3\mathbf{P_1}-3\mathbf{P_2}+\mathbf{P_3}$
$b=3\mathbf{P_0}-6\mathbf{P_1}+3\mathbf{P_2}$
$c=-3\mathbf{P_0}+3\mathbf{P_1}$
$d=\mathbf{P_0}$



Space of curve



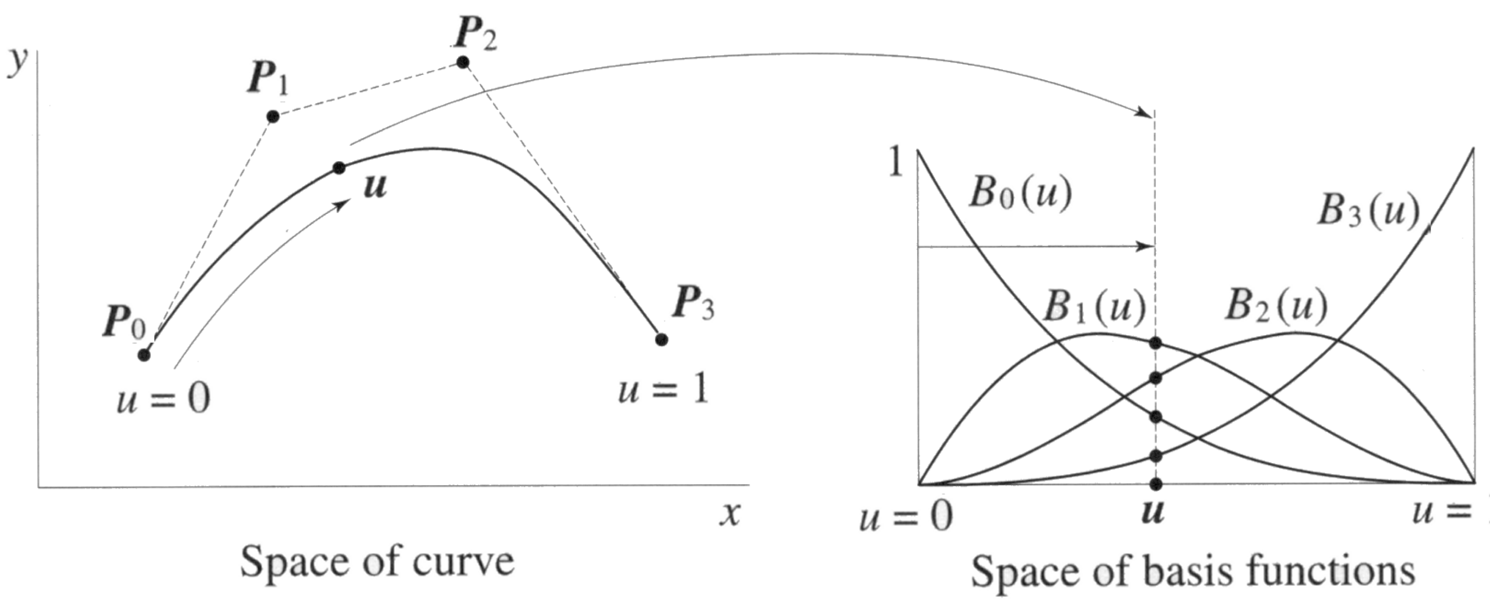
http://en.wikipedia.org/wiki/Bezier_curves

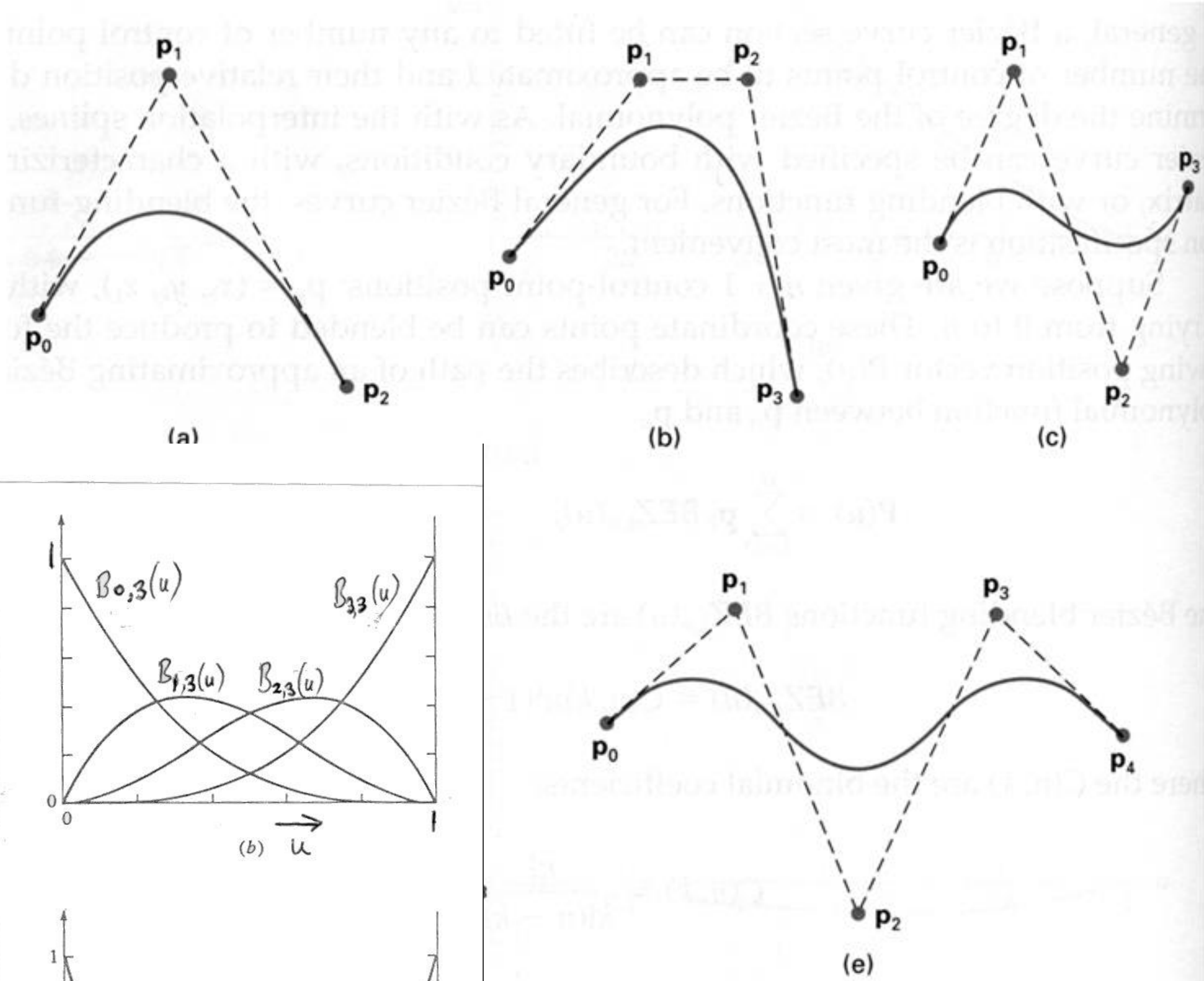
4.1 Bernstein blending functions

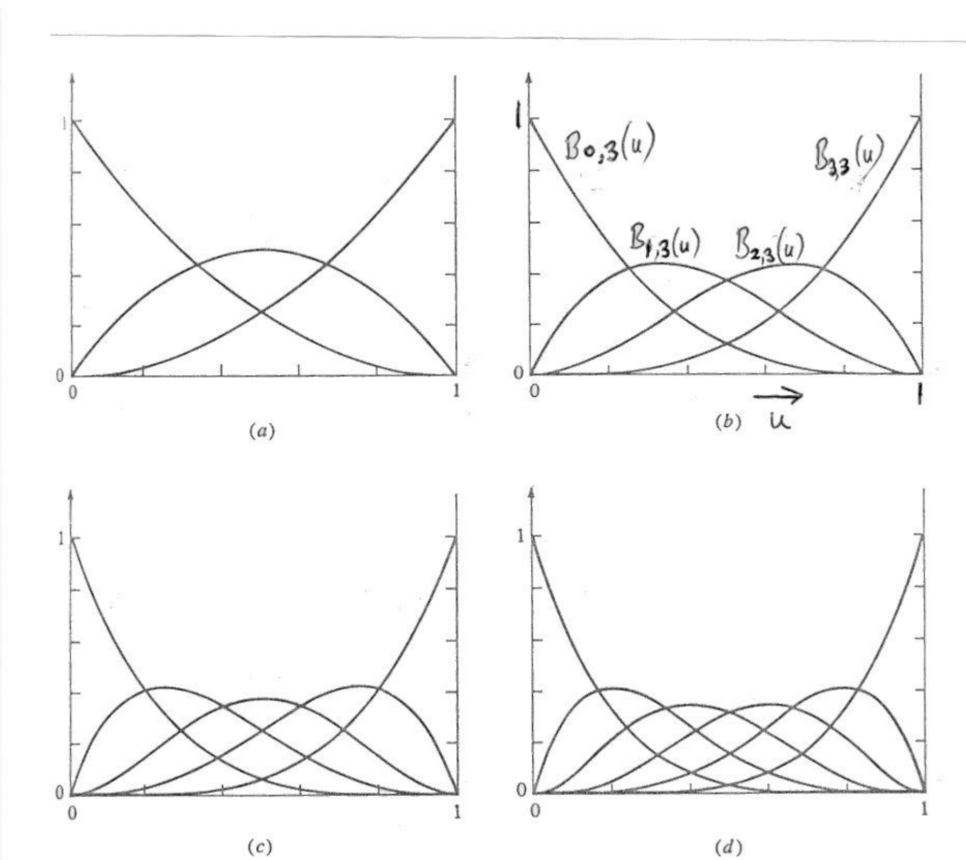
- Rewrite: $\mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + \mathbf{P}_1 3u(1-u)^2 + \mathbf{P}_2 3u^2(1-u) + \mathbf{P}_3 u^3$
- as:

$$\mathbf{Q}(u) = \sum_{i=0}^{3} \mathbf{P}_i \, B_i(u)$$

- $B_i(u)$ are the Bernstein blending functions, which sum to 1 for $0 \le u \le 1$




p_0 p_1 p_2
(a)
p_0 p_1 p_2 p_3
(b)
p_0 p_1 p_2 p_3
(c)
p_0 p_1 p_2 p_3 p_4
(e)


$B_{0,3}(u)$
$B_{1,3}(u)$
$B_{2,3}(u)$
$B_{3,3}(u)$
u
0
1
(a)
(b)
(c)
(d)

4.2 Matrix notation

- Rewrite $\mathbf{Q}(u) = \mathbf{P}_0(1-u)^3 + \mathbf{P}_1 3u(1-u)^2 + \mathbf{P}_2 3u^2(1-u) + \mathbf{P}_3 u^3$
- as $\mathbf{Q}(u) = \mathbf{U}\, B\, \mathbf{P}$

$$= \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}$$

4.3 Tangents

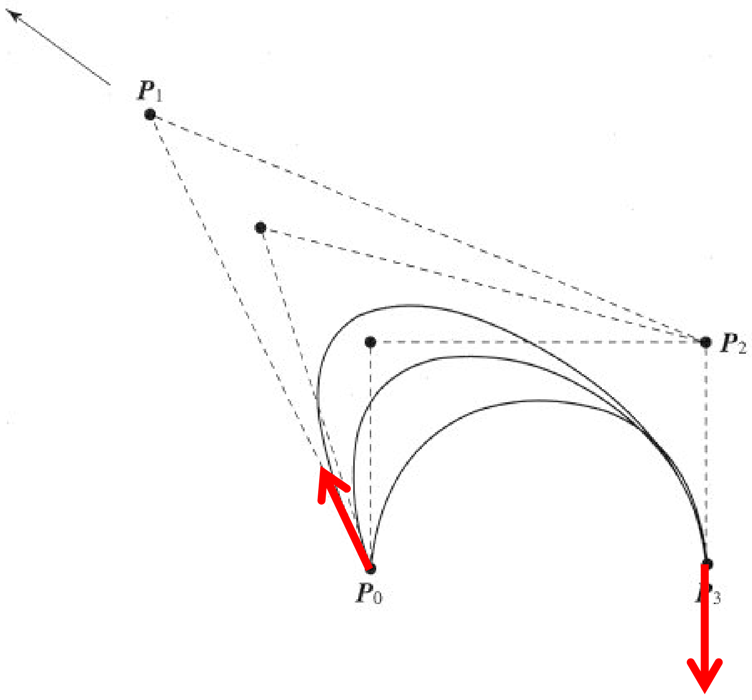
- $\mathbf{Q}(u) = au^3+bu^2+cu+d$

  where $a=-\mathbf{P_0}+3\mathbf{P_1}-3\mathbf{P_2}+\mathbf{P_3}$, $b=3\mathbf{P_0}-6\mathbf{P_1}+3\mathbf{P_2}$, $c=-3\mathbf{P_0}+3\mathbf{P_1}$, $d=\mathbf{P_0}$

- Differentiate: $Q'(u) = 3au^2+2bu+c$
- Then, set u to 0 and 1 respectively:

  $$\mathbf{Q'}(0) = 3(\mathbf{P_1}-\mathbf{P_0})$$

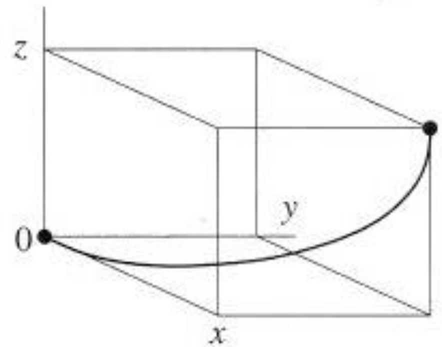
  $$\mathbf{Q'}(1) = 3(\mathbf{P_3}-\mathbf{P_2})$$



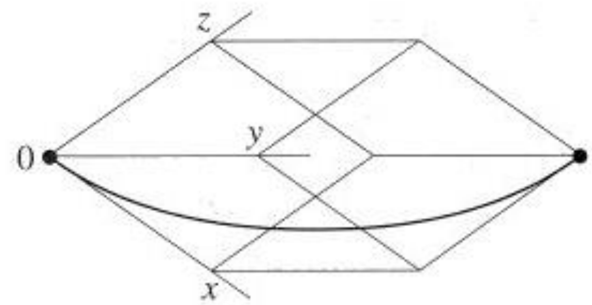
- This gives us the tangent directions at the endpoints of the curve
- Normalise to give unit tangents
- Important when joining curve segments

4.4 Bezier cubic curves: 3D

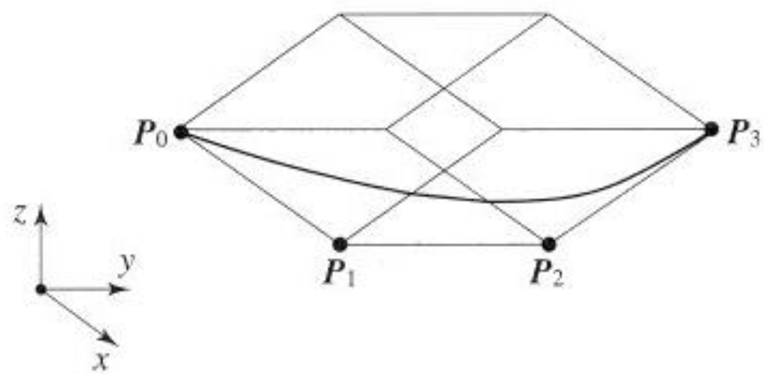
- The start and end vertices of the curve are located at opposite vertices of a parallelepiped formed by the control points



Curve 'contained' by a cube



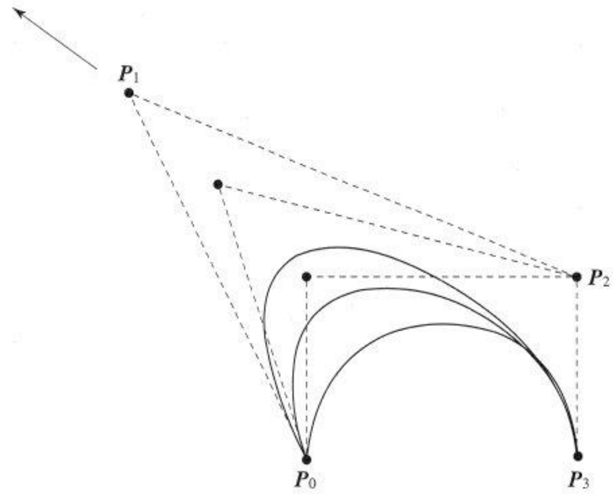
Drawing the cube into a parallelepiped changes the curve

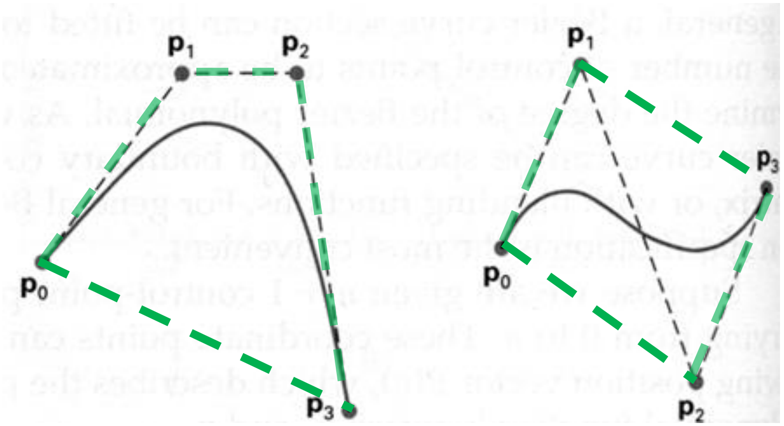


Vertices used as control points

4.5 Bezier cubic curves: properties

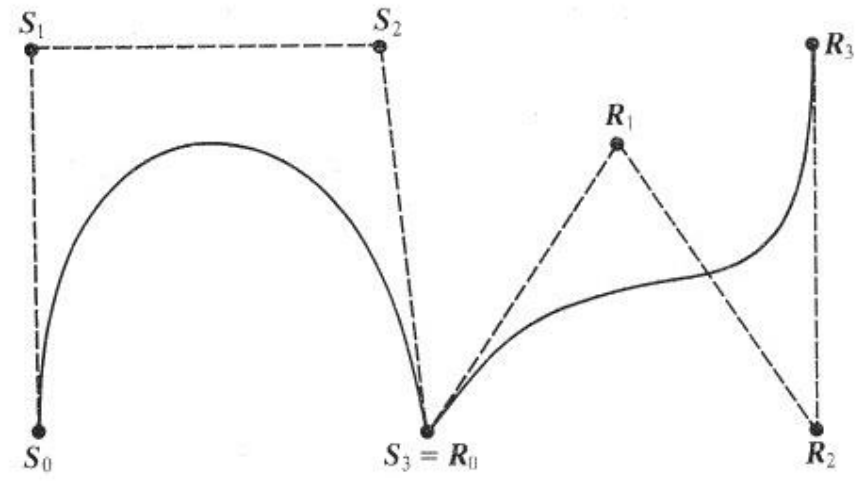
- The number of control points is always 1 larger than the degree
- Global control
  - Move a control point and affect whole curve
- Convex hull property
  - Curve contained within **convex hull** formed around the control points
  - 3D: parallelepiped
- Invariant under affine transformations
  - Transform the control points and then evaluate the curve
    = evaluate curve and transform it



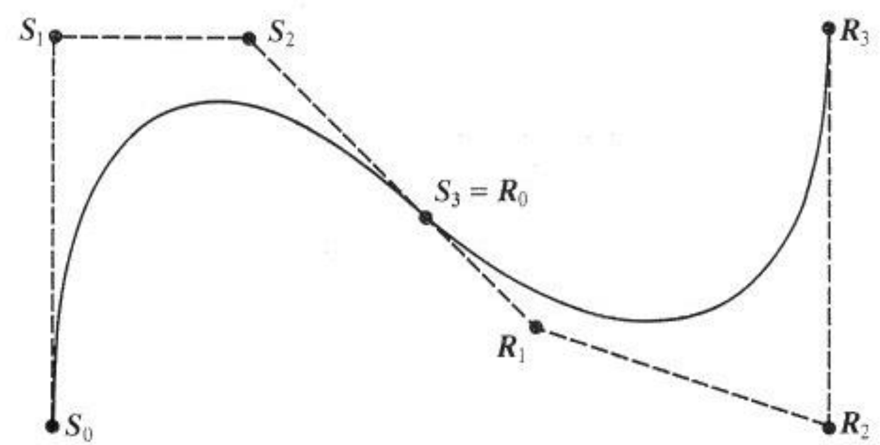


4.6 Joining Bezier curve segments

- Given two curves $\mathbf{S}_i$, $\mathbf{R}_i$
  - C^0/G^0 continuity: $\mathbf{S}_3 = \mathbf{R}_0$
  - C^1 continuity: $(\mathbf{S}_3\text{-}\mathbf{S}_2) = (\mathbf{R}_1\text{-}\mathbf{R}_0)$
  - G^1 continuity: $(\mathbf{S}_3\text{-}\mathbf{S}_2) = k(\mathbf{R}_1\text{-}\mathbf{R}_0)$
- Derivative continuity (C^1) is important for animation:
  - If an object moves along the piecewise curve with constant parametric speed, there should be no sudden jump at the joins
- For other applications, *tangent continuity* (G^1) may be enough



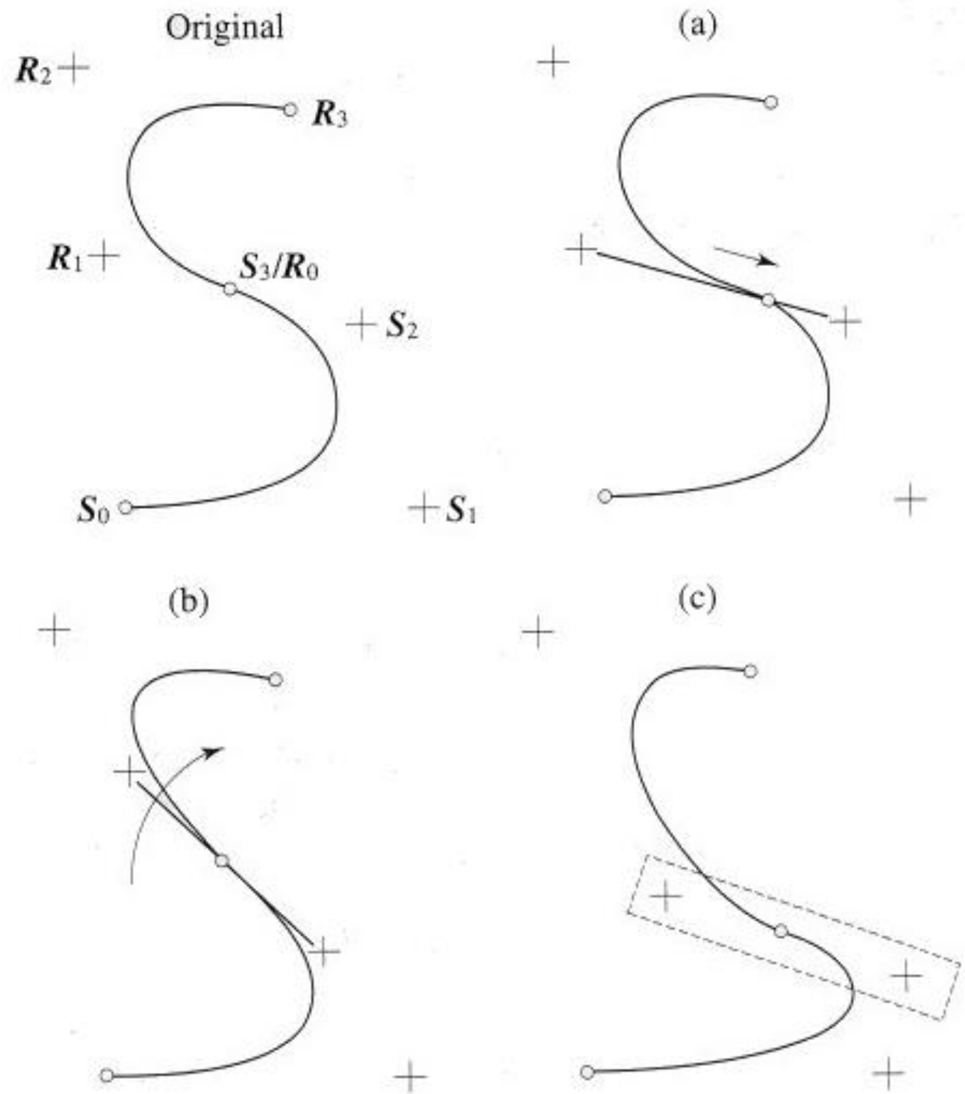
(a)



(b)

4.7 Editing joined Bezier curve segments

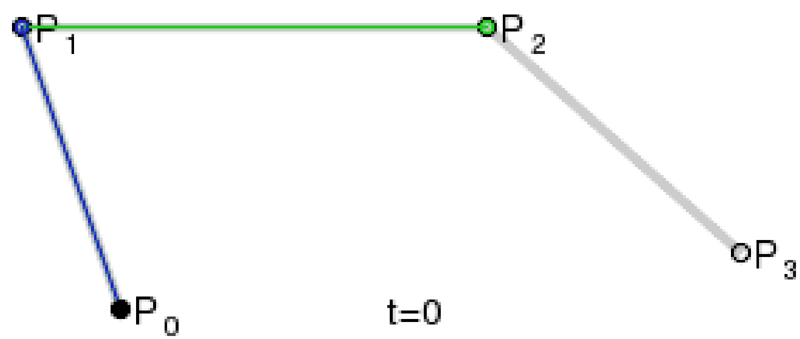
- Constraints on control point movement for a complex piecewise curve
- (a) maintains G1 continuity
- (b) and (c) maintain C1 continuity



4.8 Drawing Bezier curves

Iterative evaluation:

- Incremental steps in u and evaluate Q(u) at each iteration
  - Loop:
    u+=step; evaluate q(u); draw line
- Advantage: very simple
- Disadvantage: No easy way of knowing how fine to sample points

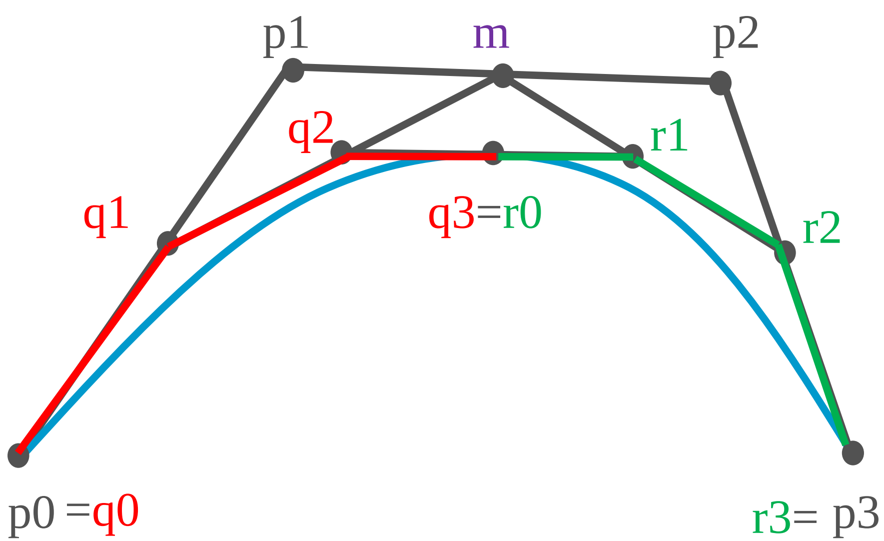


http://en.wikipedia.org – Bezier curves

4.8.1 Subdivision – de Castlejau algorithm

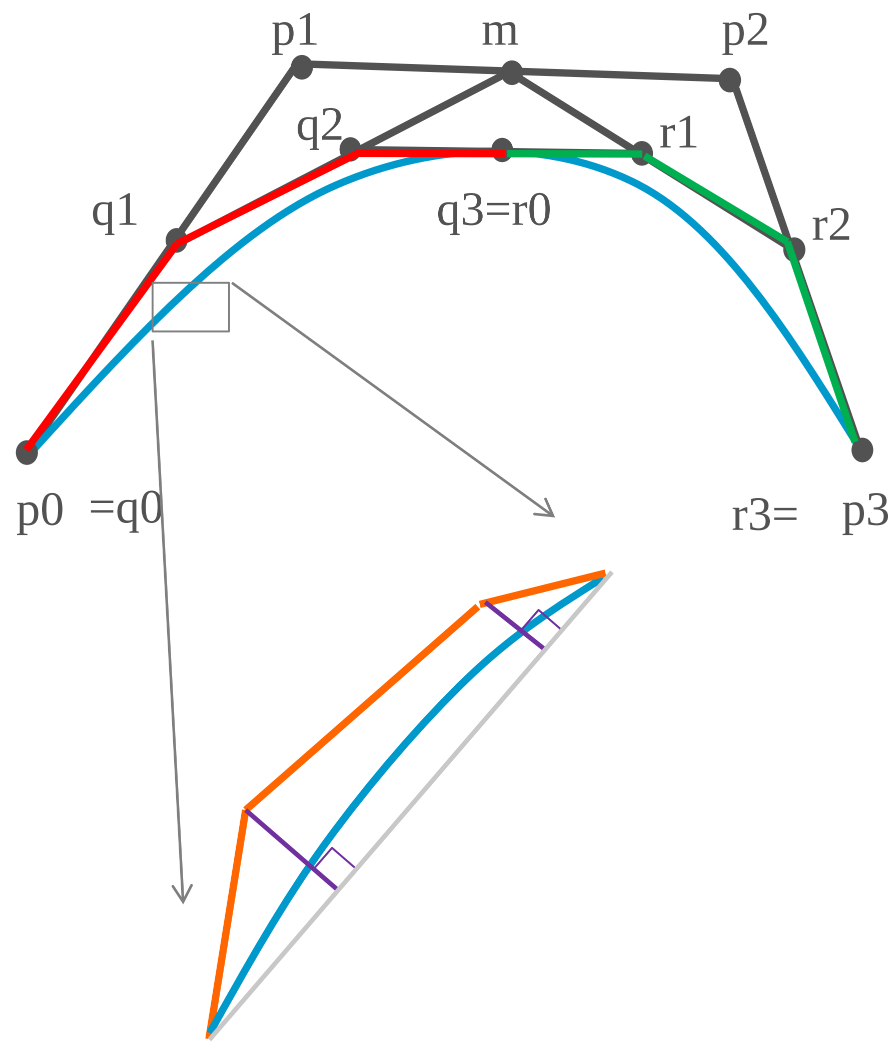
- Keep subdividing a curve into two curves
- Split one curve p_i into two curves q_i and r_i which are continuous across the join

q0 = p0
q1 = (p0+p1)/2
m = (p1+p2)/2
r2 = (p2+p3)/2
r3 = p3
q2 = (q1+m)/2
r1 = (m+r2)/2
q3 = (q2+r1)/2
r0 = q3



4.8.1 Subdivision – de Castlejau algorithm

- Alternative stopping conditions:
  a) Uniform depth
  b) The curve matches a linearity criterion
- At the limit, draw the control polygon for each subcurve
  - If control vertices are nearly collinear, then convex hull is a good approximation to the curve
- *An inefficient way to draw a curve*



5. Computer facial animation

- The first three-dimensional facial animation was created by Parke in 1972.

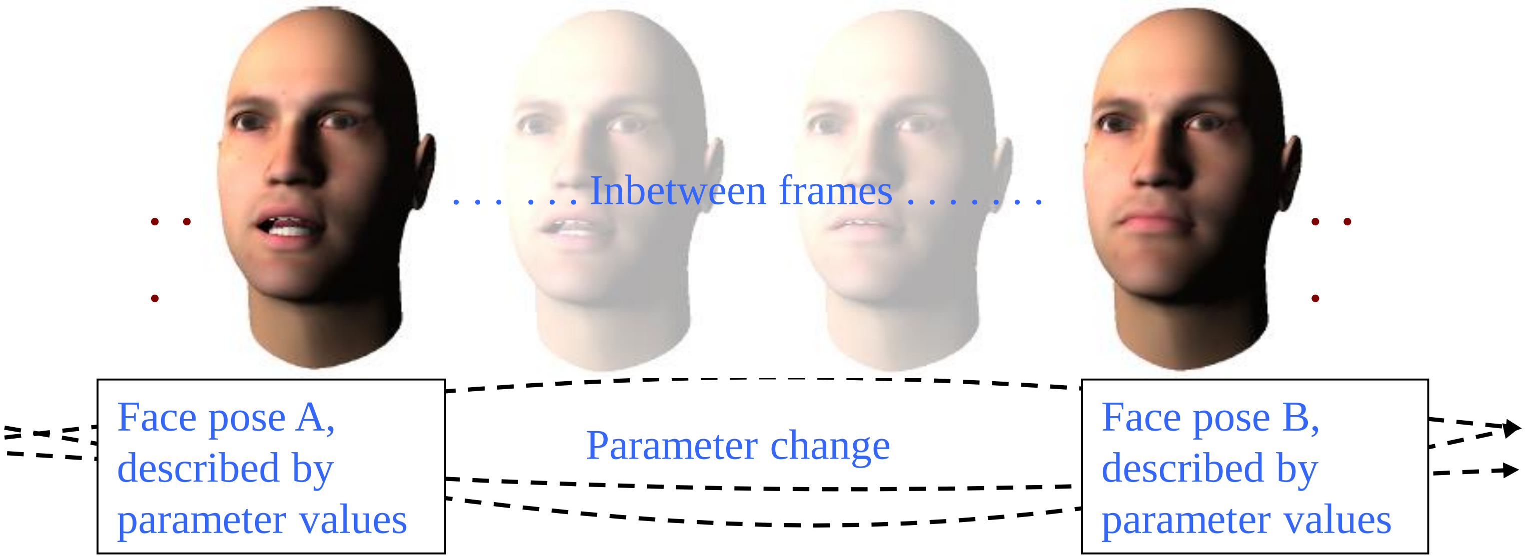
"A Computer Generated Hand", Ed Catmull and Fred Parke, University of Utah, 1972 http://pixartimes.com/2011/12/28/ed-catmulls-computer-animated-hand-added-to-national-film-registry/

Jump to 4:49

- Lots of different approaches since then
  - Blendshapes/face poses/morph targets; Motion-capture; Data-driven approaches; Physics-based approaches

5.1 Interpolate between face poses

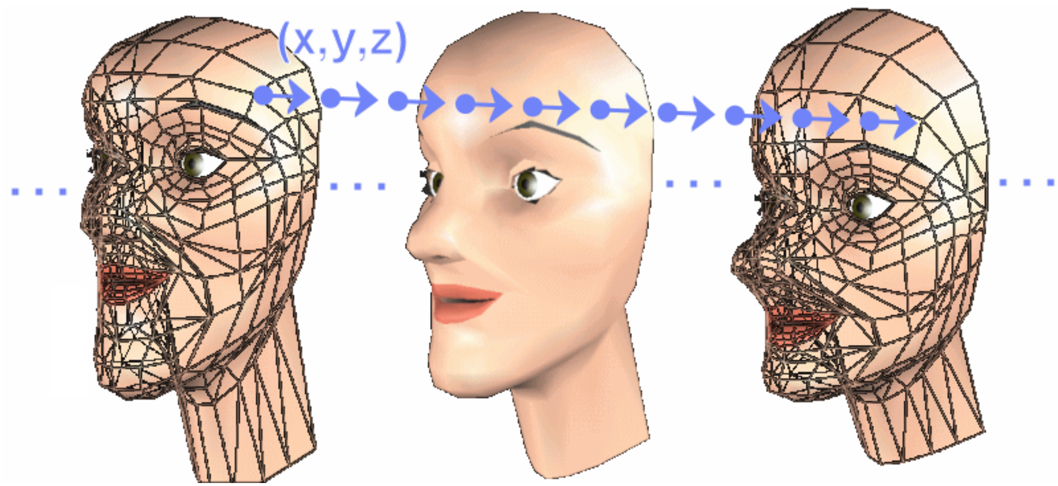
- Create pose targets at discrete time locations and interpolate



Interpolate vertex positions

- Polygon mesh created for a face in X poses
- Vertex correspondence between poses
- Intermediate poses generated by interpolation of vertex positions
  - E.g: Linear: $p_i = p_1(1\text{-}t) + p_2(t)$, $0<=t<=1$

- Low-level animation control technique
  - Very labour intensive
- Still a common approach today

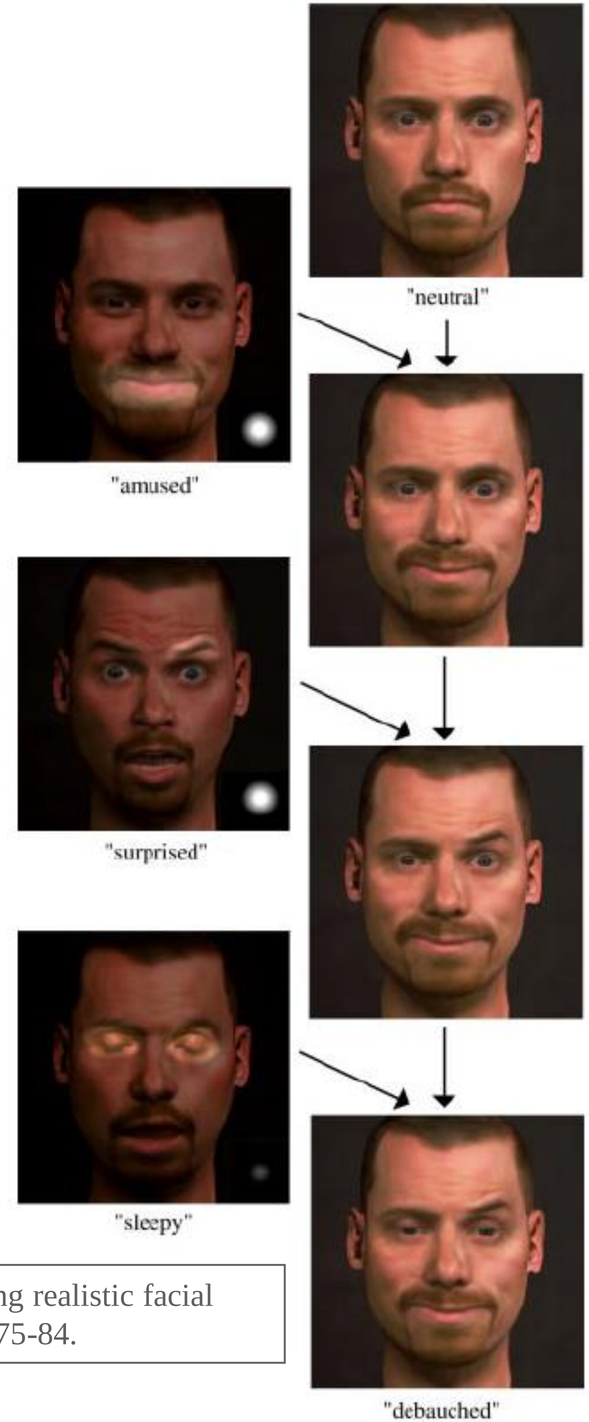


5.2 Regionalisation to create poses/blendshapes/morph targets

- (Pighin et al, 98) uses regional blends to combine faces (geometry and texture) in a localised way. Weights are assigned to regional extremes.

Figure 4 A global blend between "surprised" (left) and "sad" (center) produces a "worried" expression (right).

Figure 5 Combining the upper part of a "neutral" expression (left) with the lower part of a "happy" expression (center) produces a "fake smile" (right).



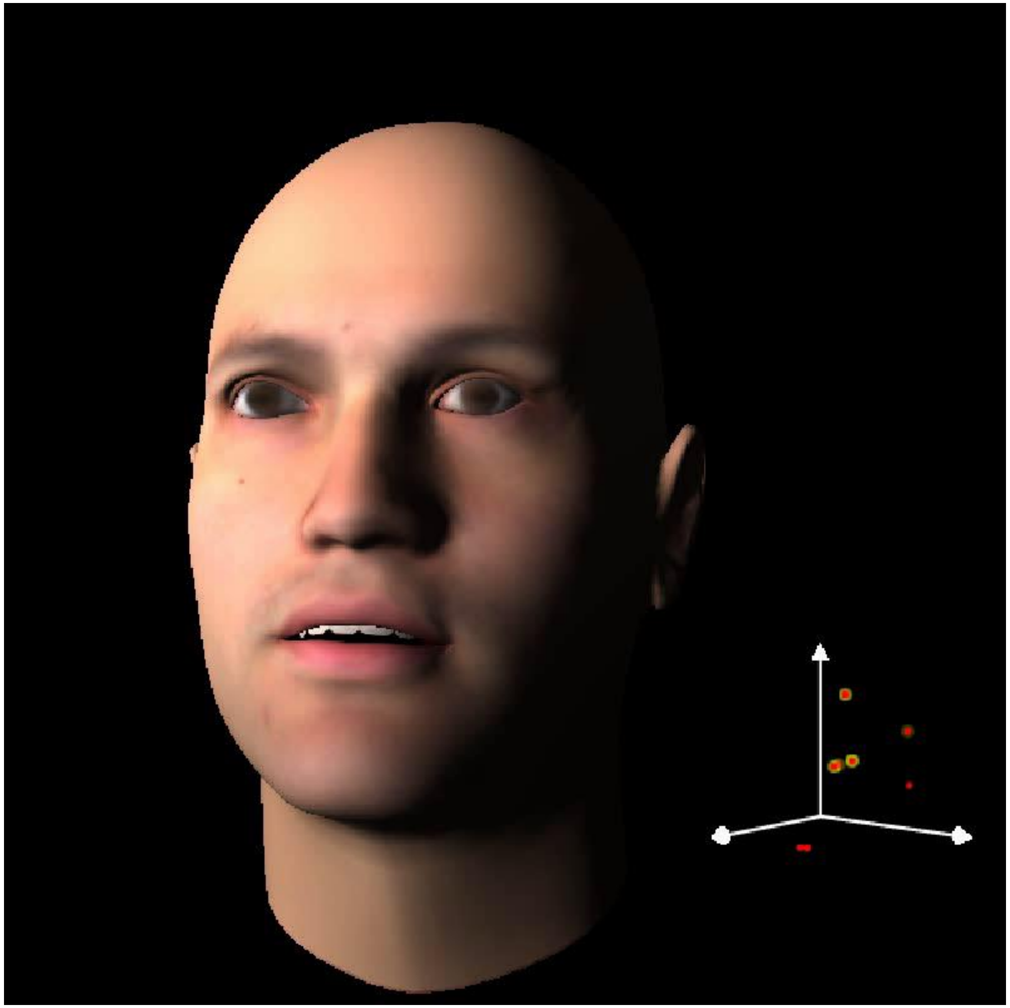
Pighin, F., J. Hecker, D. Lischinski, R. Szeliski and D.H. Salesin, "Synthesizing realistic facial expressions from photographs", SIGGRAPH 98 Conference Proceedings, pp.75-84.

5.3 Tools to create facial poses

- Facial rigs: control techniques to deform a localised region of space
  - map to descriptive parameters: e.g. raise eyebrow or contract muscle x
- Use to create poses which can then be interpolated
- Tools to set up rigs, e.g. "Rigging a Face in Maya", Jason Baskin, Lynda.com (via MUSE) or "3ds Max: Character rigging", George Maestri, Lynda.com (via MUSE)

5.4 Using a different coordinate system

- Use Principal component analysis to reduce the number of parameters (vertex positions) to a few parameters that best explain the variance in the data set
- Interpolate (parametrically) the Principal components
- Reconstruct vertex positions from the principal components

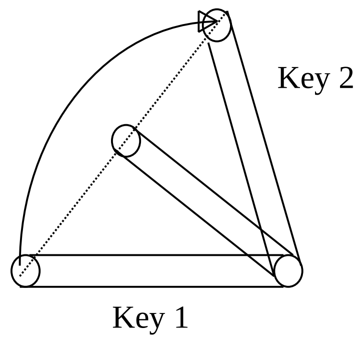


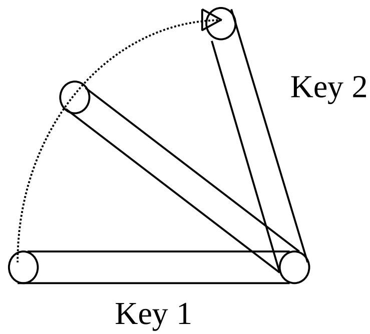
5.5. Commercial example: Gollum

- Interpolation (based on keyframes) is a commonly-used CFA approach
  - 946 blend shapes for Gollum in Lord of the Rings (Lewis et al, 2005)

Lewis, J.P., Mooser, J., Deng, Z. and Neumann, U., Reducing blendshape interference by selected motion attenuation. In Proc. of ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D), 25–29

6. Summary





- Parametric curves are useful in animation
  - Be careful to choose the correct parameter for interpolation, e.g. angle instead of position
- Cubic Bezier curves
  - Intuitive shape editing control
  - Convex hull property
  - Invariant under affine transformations
  - Join curve segments to make longer curves
- There are other parametric curves
  - Catmull-rom – interpolating spline for sequences of parameter values
  - B-spline – approximating spline for sequences of parameter values
  - B-splines and non-uniform rational B-splines (NURBS) more popular in CAD work