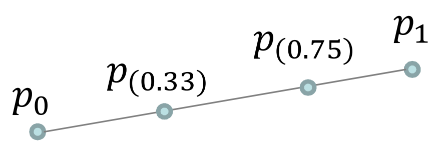
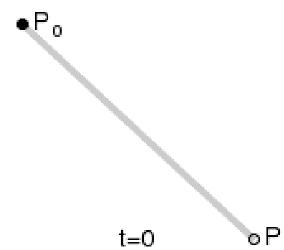
LEI / LCC
Departamento de Informática
Universidade do Minho

Curves and Surfaces

Bezier Curves of degree 1

- Curves of degree 1 are straight lines
- To define a straight line between two points we can use the following equation:
  - $p(t) = (1 - t)p_0 + tp_1, with\ \ 0 \leq t \leq 1$
- Varying t between 0 and 1 we can get all points in the line



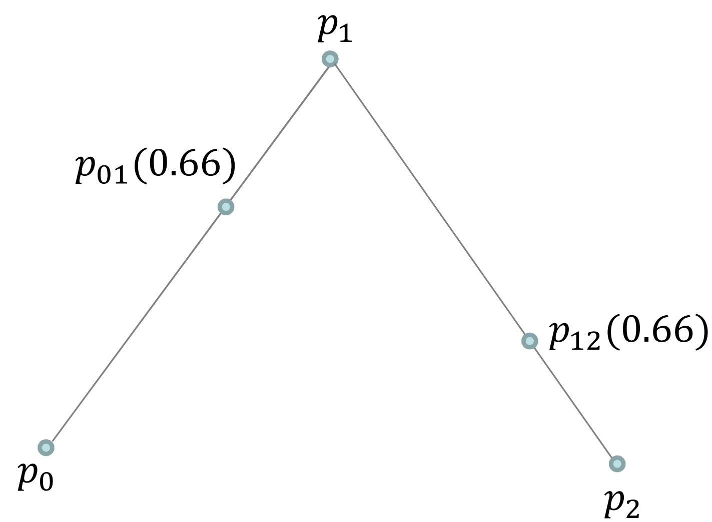


Phil Tregoning (https://commons.wikimedia.org/wiki/File:B%C3%A9zier_1_big.gif)

- Can we extend this reasoning to higher degree curves?

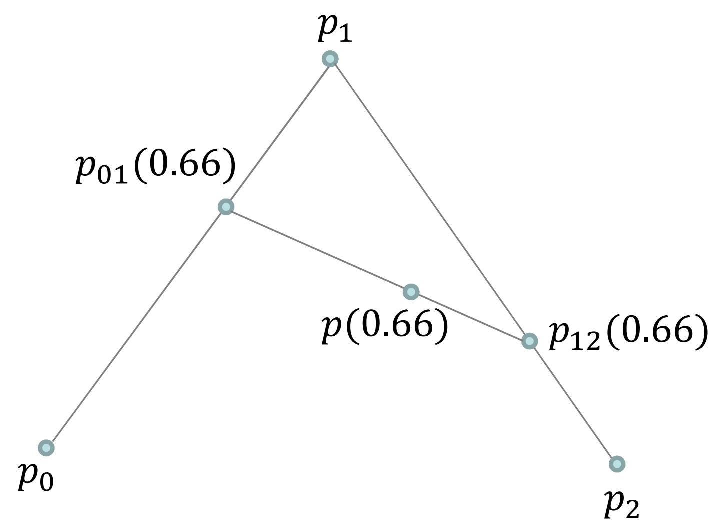
Bezier Curves of degree 2

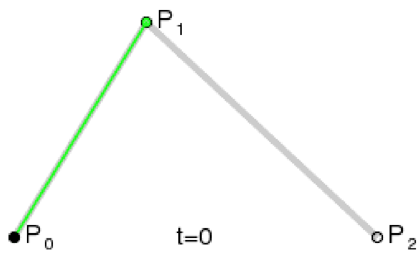
- Do the same process for each line segment and get p_{01} and p_{02}
- Three points are required (control points)



Bezier Curves of degree 2

- Do the same process for each line segment and get $p_{01}$and p_{02}
- Now connect $p_{01}$to p_{02} and repeat the process in this line segment
  - $p_{01}(t) = (1-t)p_0 + tp_1$
  - $p_{12}(t) = (1-t)p_1 + tp_2$
  - $p(t) = (1-t)p_{01} + tp_{12}$

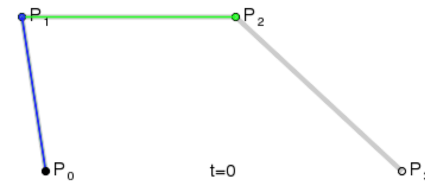




Phil Tregoning (https://commons.wikimedia.org/wiki/File:B%C3%A9zier_2_big.gif)

Bezier Curves of degree 3

Phil Tregoning (https://commons.wikimedia.org/wiki/File:B%C3%A9zier_3_big.gif)



- Exactly same process but now we start we 4 control points

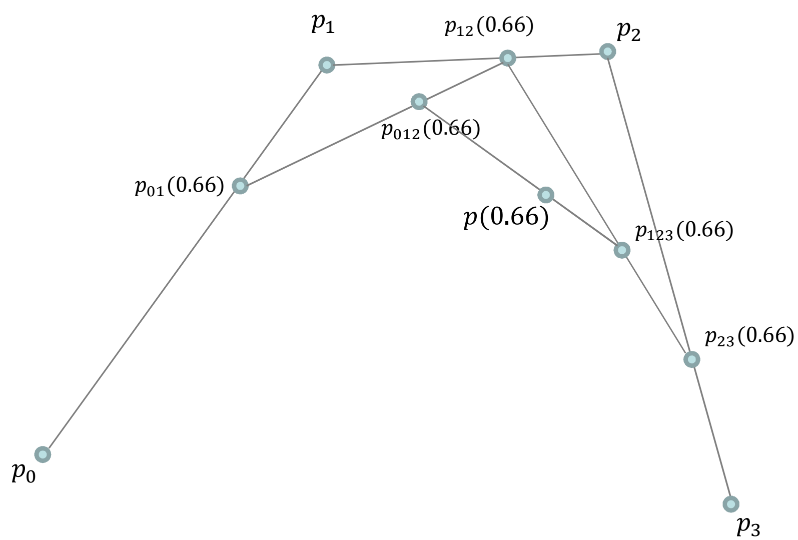
  - $p_{01}(t) = (1-t)p_0 + tp_1$
  - $p_{12}(t) = (1-t)p_1 + tp_2$
  - $p_{23}(t) = (1-t)p_2 + tp_3$

  - $p_{012}(t) = (1-t)p_{01} + tp_{12}$
  - $p_{123}(t) = (1-t)p_{12} + tp_{23}$

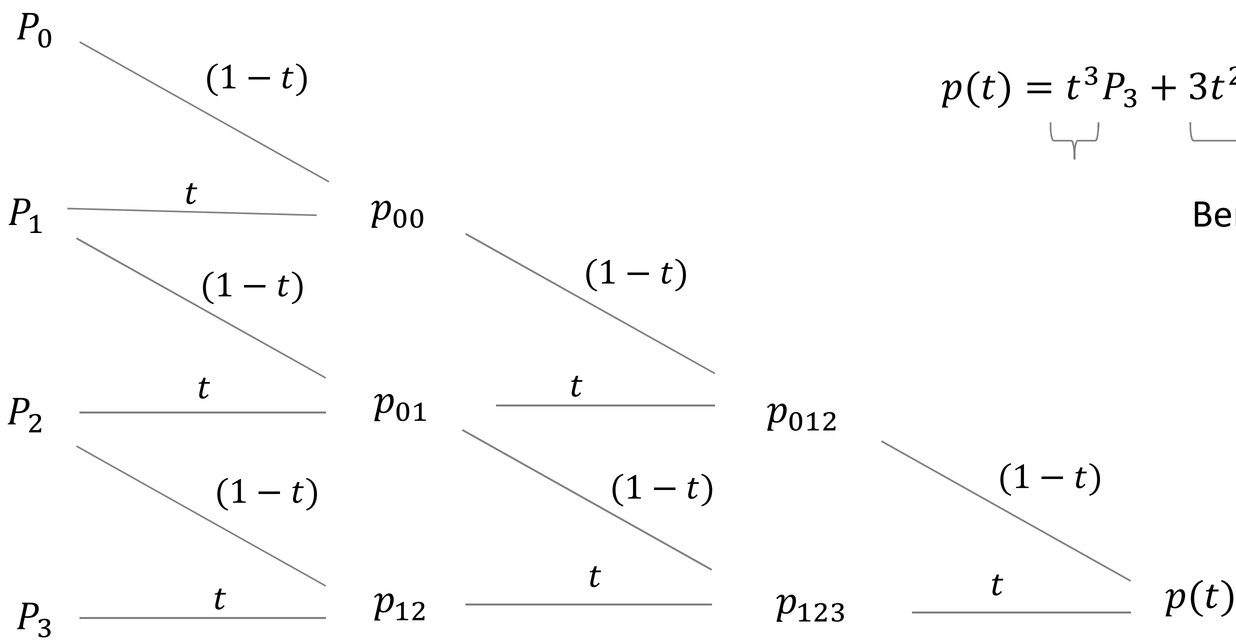
  - $p(t) = (1-t)p_{012} + tp_{123}$

- The process can go on for any degree

- With degree n we need $n+1$ control points



Bezier Curves of degree 3

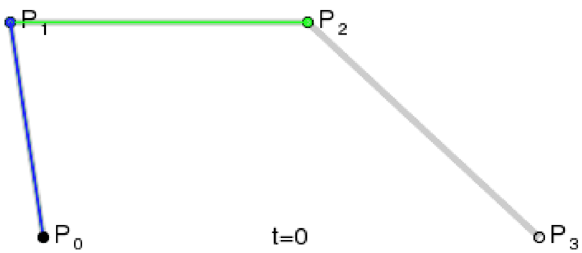


$$p(t) = \underbrace{t^3}P_3 + \underbrace{3t^2(1-t)}P_2 + \underbrace{3t(1-t)^2}P_1 + \underbrace{(1-t)^3}P_0$$

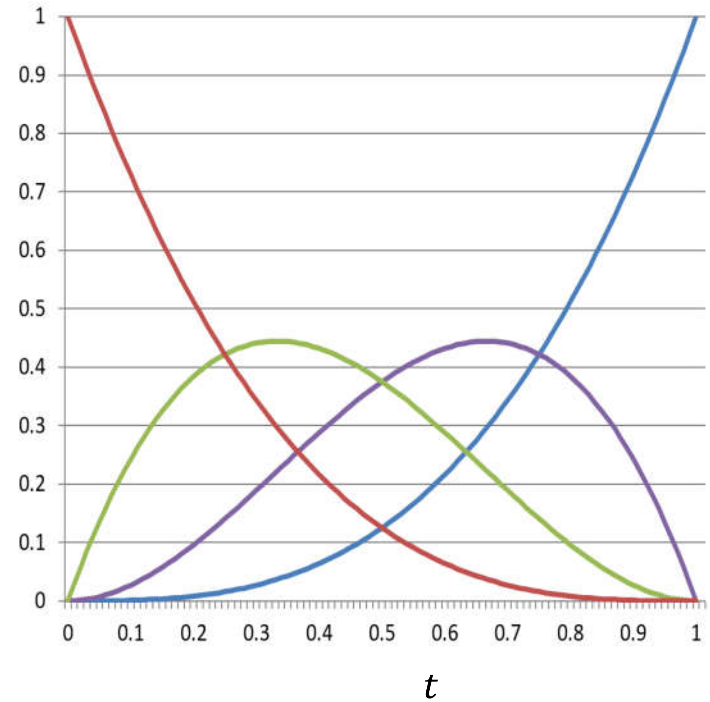
Bernstein Polinomials

Bezier Curves of degree 3

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



$$B_{0,3}(t) = (1-t)^3$$
$$B_{1,3}(t) = 3t(1-t)^2$$
$$B_{2,3}(t) = 3t^2(1-t)$$
$$B_{3,3}(t) = t^3$$



Bezier Cubic Curves

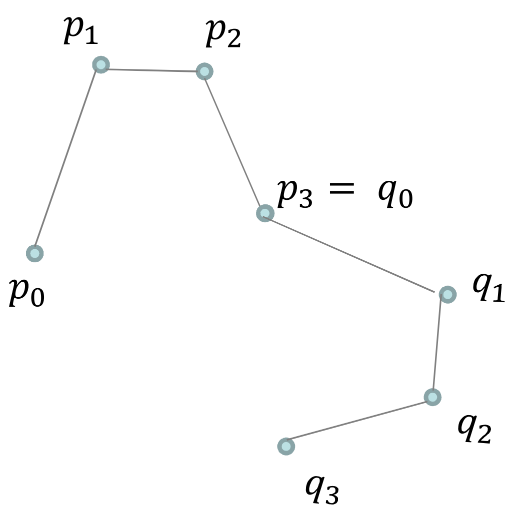
- Matrix form

$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

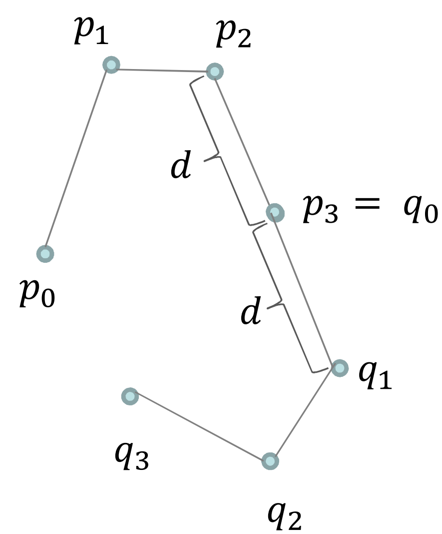
$$p'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Continuity when joining Bezier curves

Continuity of position (C_0)

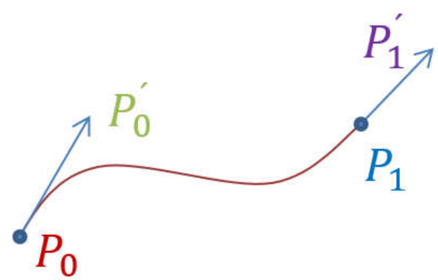


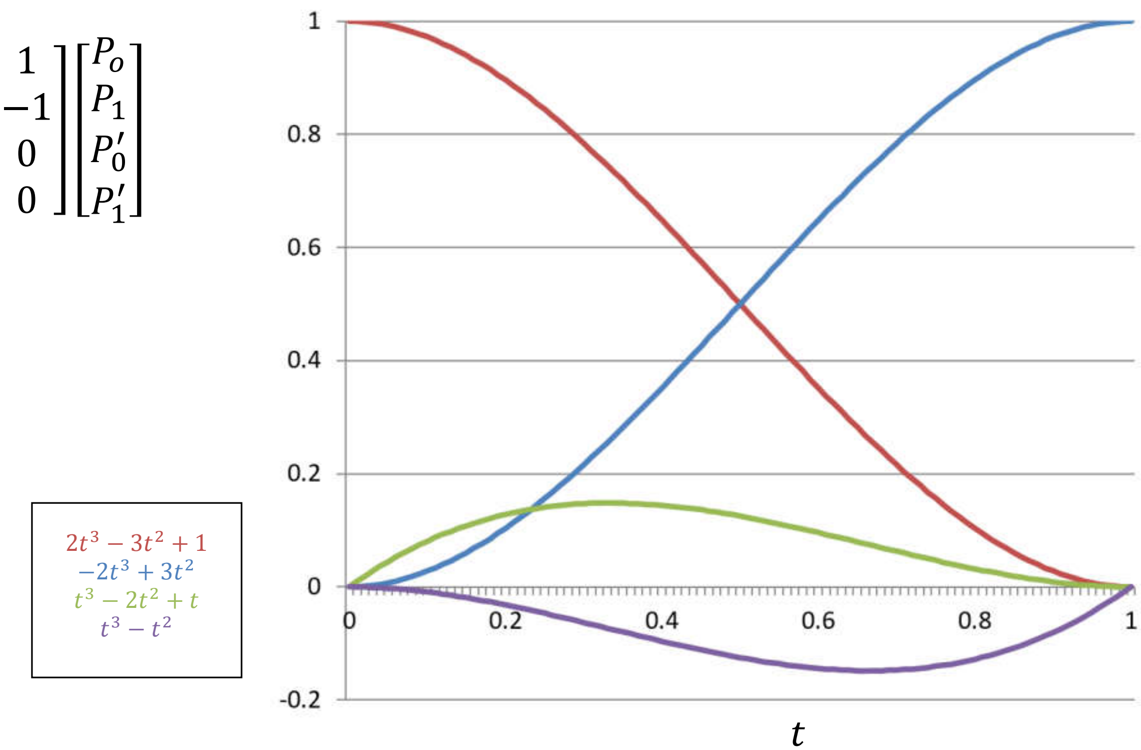
Continuity of first derivative (C_1) $\Rightarrow$ $p_3 - p_2 = q_1 - q_0$



Hermite Cubic Curves

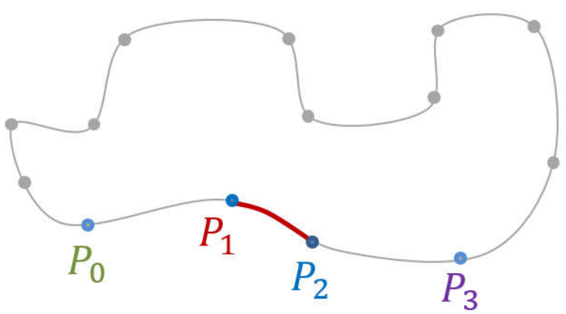
$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_o \\ P_1 \\ P'_0 \\ P'_1 \end{bmatrix}$$



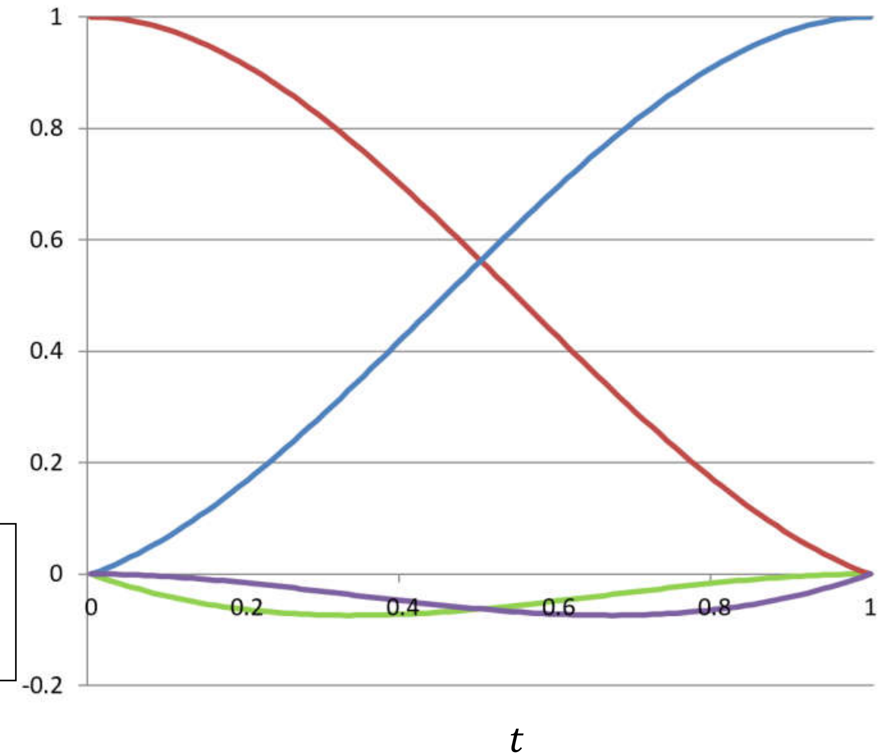


Catmull-Rom Curves

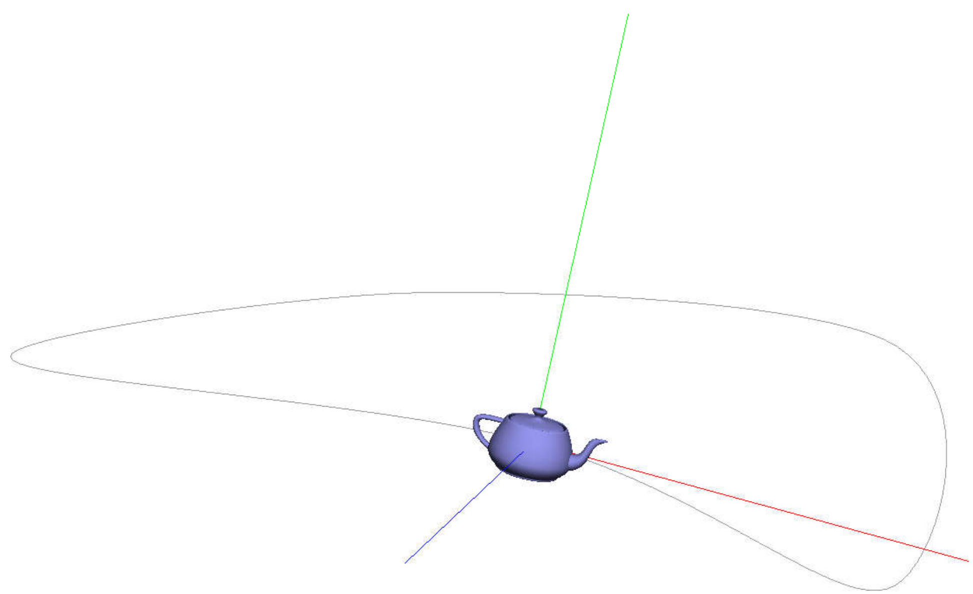
$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -0.5 & 1.5 & -1.5 & 0.5 \\ 1 & -2.5 & 2 & -0.5 \\ -0.5 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_o \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$



$-0.5t^3 + t^2 - 0.5t$
$1.5t^3 - 2.5t^2 + 1$
$-1.5t^3 + 2t^2 + 0.5t$
$0.5t^3 - 0.5t^2$

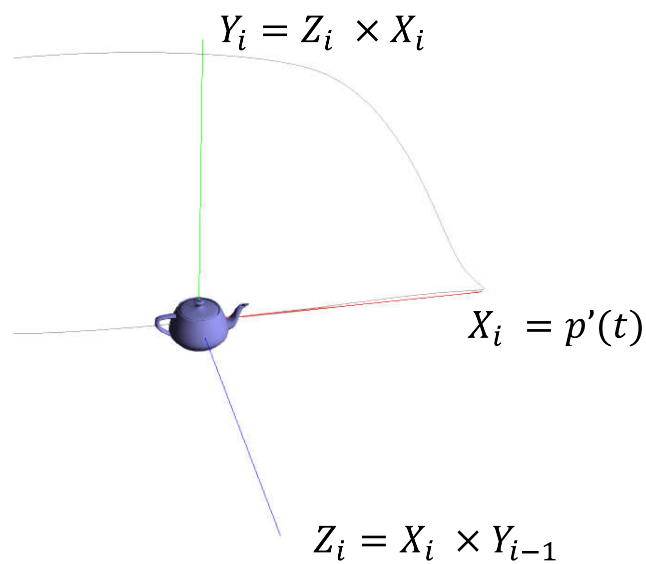


Animation with Catmull-Rom Curves



Animation with Catmull-Rom Curves

- Axis for Rotation Matrix
  - Available data at instant t
    - $p(t)$ - position of an object “walking” along the curve
    - $p'(t)$ - vector tangent to the curve
  - Transform for teapot
    - Translation to place teapot
    - Rotation to align with curve
    - $Y_0 = (0,1,0)$



Cubic Curves – Catmull-Rom

- Assuming an initial specification of an $\overrightarrow{Y_0}$ vector, to align the object with the curve, we need to build a rotation matrix for the object:

$$
\begin{aligned}
\overrightarrow{X_i} &= p'(t) \\
\overrightarrow{Z_i} &= X_i \times \overrightarrow{Y_{i-1}} \\
\overrightarrow{Y_i} &= \overrightarrow{Z_i} \times \overrightarrow{X_i}
\end{aligned}
\qquad
M = \begin{bmatrix} X_x & Y_x & Z_x & 0 \\ X_y & Y_y & Z_y & 0 \\ X_z & Y_z & Z_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

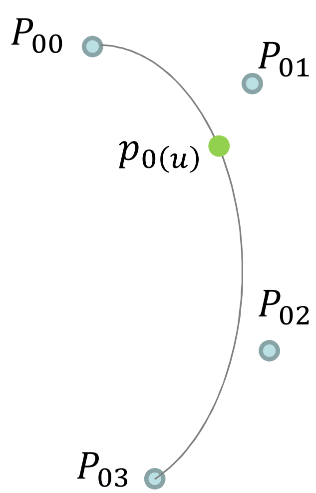
Note: All vectors need to be normalized

```
glMultMatrixf(float *m)
```

- Current OpenGL MODEL_VIEW matrix gets multiplied by `m`

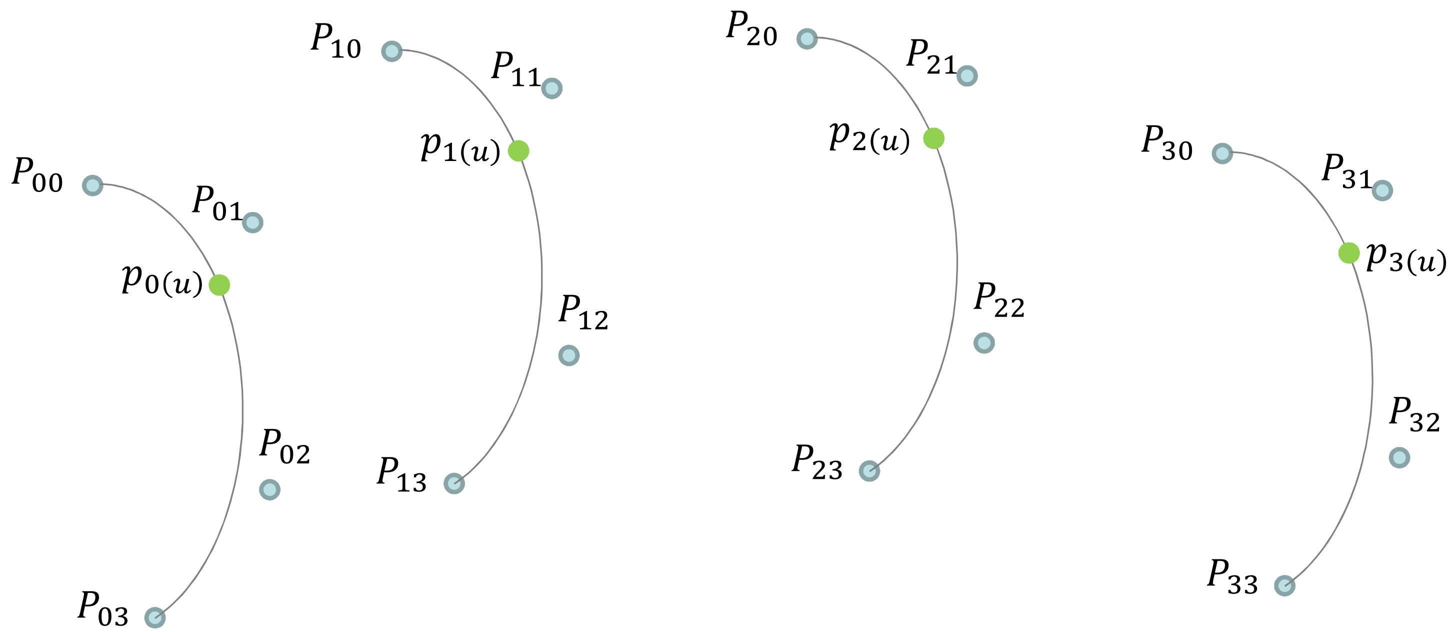
Note: OpenGL matrices are column major => compute the transpose instead

Bezier Patches



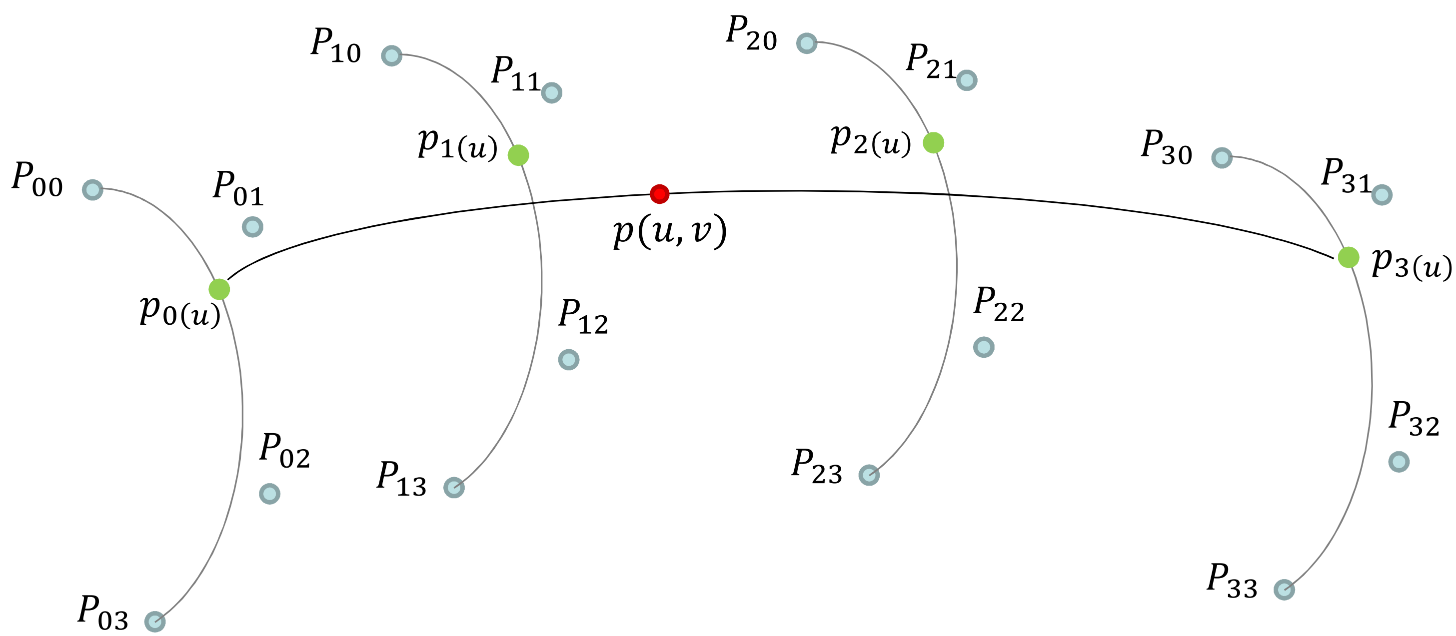
A Bezier curve of degree 3

Bezier Patches



Consider 4 distinct Bezier curves, select a value for parameter u (in patches we use u instead of t), equal for all curves, and compute a point in each curve.

Bezier Patches



Consider the resulting 4 points (green dots) as the control points of a new Bezier Curve. Now select a value for parameter v and the result is a point in the patch $p(u,v)$, the red dot.

Bezier Patches

$$p(u,v) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

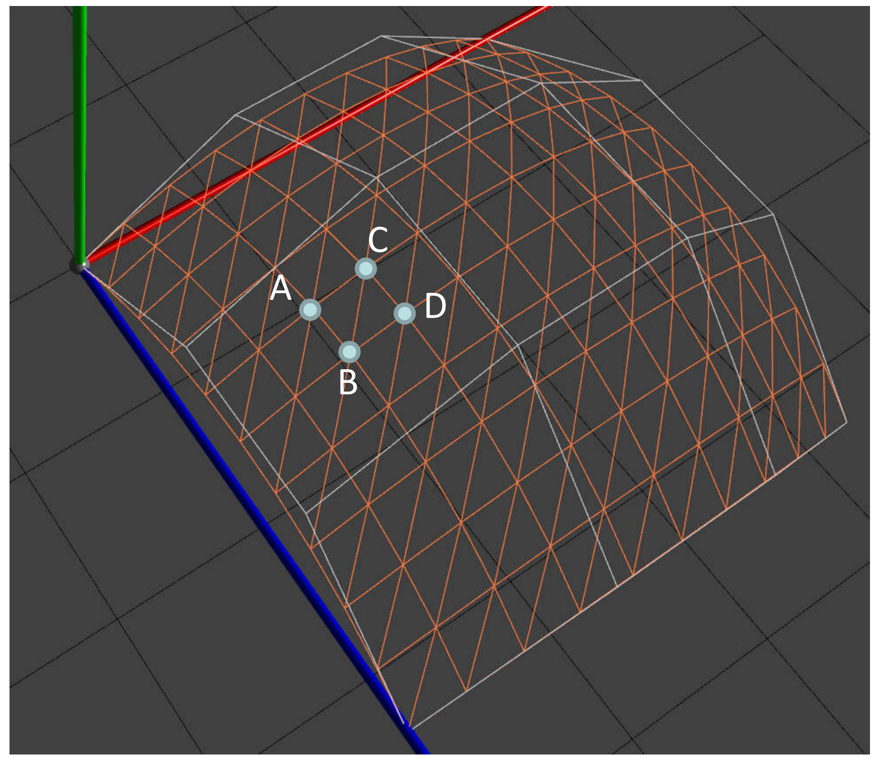
$$\frac{\partial p(u,v)}{\partial u} = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T V^T$$

$$\frac{\partial p(u,v)}{\partial v} = UM \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} 3v^2 \\ 2v \\ 1 \\ 0 \end{bmatrix}$$

Normal
The normal vector at any point of the surface is defined as the normalized result of the cross product of the tangent vectors.

Bezier patches

- Building a triangulation for the patch
  - The white line vertices are the control points
  - The Bezier patch is in orange
    - Level of tessellation (divisions) = 10
    - A = $p(0.2, 0,4)$
    - B = $p(0.2, 0.5)$
    - C = $p(0.3, 0.4)$
    - D = $p(0.3, 0.5)$



Bezier patches

- Computing the normals for the patch
  - $A = (0.3, 0.4)$
  - $\vec{u} = \frac{\partial p(0.3{,}0{,}4)}{\partial u}$
  - $\vec{v} = \frac{\partial p(0.3{,}0.4)}{\partial v}$
  - $\vec{n} = \vec{v} \times \vec{u}$