# Curves and Surfaces

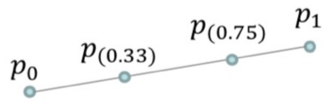
**Bezier curves:** polynomial curve being the linear interpolation between some representative points, called control points.

→ **Degree 1**: straight lines between 2 points → $p(t) = p_0 + t\vec{v}$

$p(t) = (1-t)p_0 + tp_1$ with $0 \le t \le 1$

varying $t$ we can get all points in the line.

ganhamos curvatura

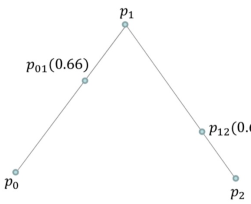


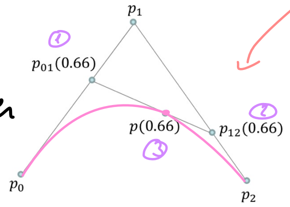

→ **Degree 2**: three points are required and it is the same process for each line segment to get $p_{01}$ and $p_{02}$. Connect $p_{01}$ to $p_{02}$ and repeat the process in this line segment

$p_{01}(t) = (1-t)p_0 + tp_1$

$p_{12}(t) = (1-t)p_1 + tp_2$

podemos ter curvas mais complexas por exemplo:




1 - ①
2 - ②
3 - ligar ① e ②
4 - ③

→ **Degree 3**: the process is exactly the same but with 4 control points.

degree x
↳ x+1 control points

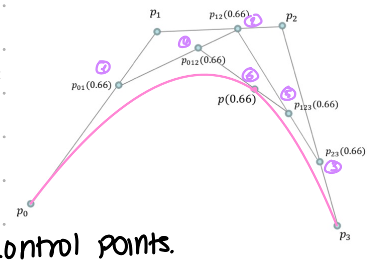
$p_{01}(t) = (1-t)p_0 + tp_1$

$p_{12}(t) = (1-t)p_1 + tp_2$

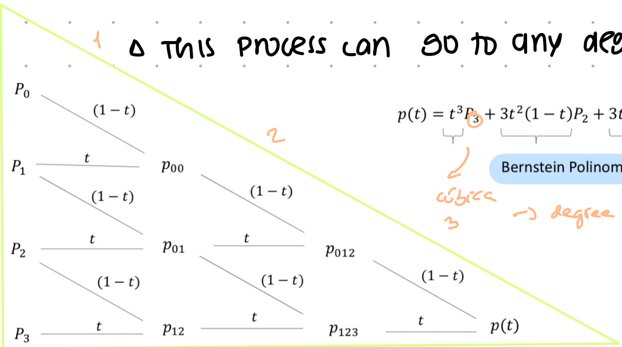
$p_{23}(t) = (1-t)p_2 + tp_3$ $\quad p(t) = (1-t)p_{012} + tp_{123}$

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

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




⚠ This process can go to any degree. With degree n we need n+1 control points.




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

cúbica 3 → degree 3

Bernstein Polinomials ⟶ Polynomial that is a linear combination of Bernstein basis polynomials

$b_{v,n}(x) = \binom{n}{v} x^v (1-x)^{n-v}$

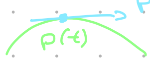
↳ binomial coefficient

**Matrix Form** → Matriz Bezier

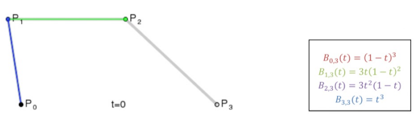
$$p(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

derivada — derivar este vetor

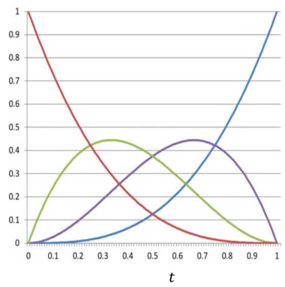
$$p'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$




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



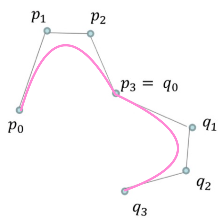

↳ fazer isto para (x, y, z)




→ Continuity in Joining Curves: the last control point of the first curve must be the same as the first control point of the next curve.

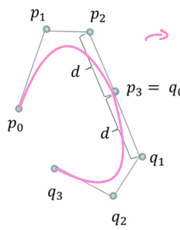
curva contida na área convexa definida pelos pontos de controlo

Continuity of position ($C_0$)



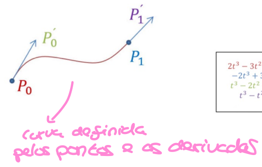

Continuity of first derivative ($C_1$) $\Rightarrow p_3 - p_2 = q_1 - q_0$

→ sem mudança brusca

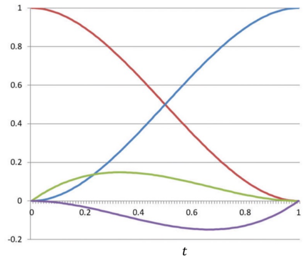


**Hermite curves:** spline where each piece is a 3rd degree polynomial specified in Hermite form, that is, its values and first derivatives at the end points of the corresponding domain interval.

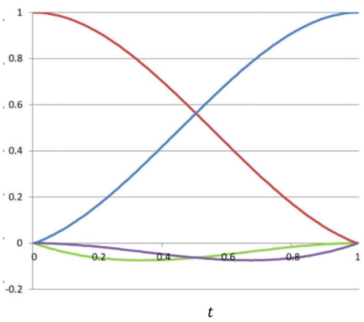
$$p(t) = [t^3 \quad t^2 \quad t \quad 1]\begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} P_0 \\ P_1 \\ P_0' \\ P_1' \end{bmatrix}$$




$2t^3 - 3t^2 + 1$
$-2t^3 + 3t^2$
$t^3 - 2t^2 + t$
$t^3 - t^2$

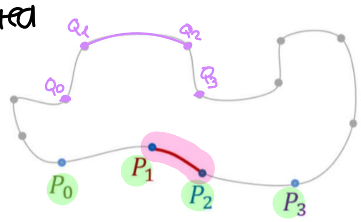



**Catmull-Rom curves:** the specified curve will pass through all of the control points, which is not true for all types of curves. To calculate a point on the curve, two points on either side of the desired point are required.




→ Given the control points $P_0, P_1, P_2$ and $P_3$ and the value $t$ the location of the point can be calculated by $p(t)$

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



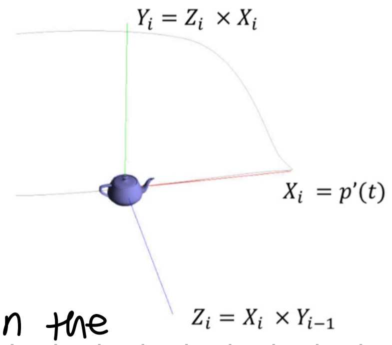

$-0.5t^3 + t^2 - 0.5t$
$1.5t^3 - 2.5t^2 + 1$
$-1.5t^3 + 2t^2 + 0.5t$
$0.5t^3 - 0.5t^2$

→ The formula gives the Catmull-Rom curve the following characteristics:

- the curve passes through all the control points
- the curve is $C^1$ continuous, meaning that there are no discontinuities in the tangent direction and magnitude
- The curve is not $C^2$ continuous since the 2nd derivative is linearly interpolated within each segment, causing the curvature to vary linearly over the length of the segment
- Points on a segment may lie outside of the domain of the 2 points

→ Axis for Rotation Matrix

- Available data at instance $t$
  - $p(t)$ - 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
  - $Y_0 = (0, 1, 0)$




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

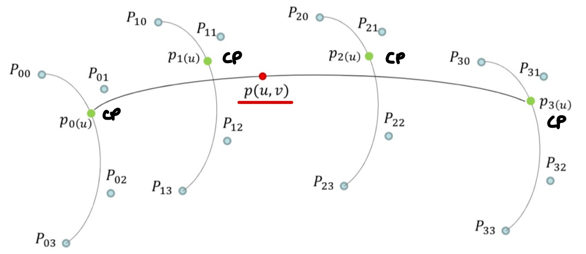
$$\vec{X_i} = p'(t)$$
$$\vec{Z_i} = \vec{X_i} \times \vec{Y_{i-1}}$$
$$\vec{Y_i} = \vec{Z_i} \times \vec{X_i}$$

$$M = \begin{bmatrix} X_x & Y_x & Z_x & 0 \\ X_y & Y_y & Z_y & 0 \\ X_z & Y_z & Z_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

⚠ All vectors need to be normalized!

$\boxed{\text{glMultMatrix(float *m)}}$ current OpenGL MODE_VIEW matrix gets multiplied by m

**Bezier Patches:** set of control points, however, does not, in general, pass through the central points; rather, it is stretched toward them as though each were an attractive force.




Bezier curve of degree 3

→ consider 4 distinct Bezier curves, select a value for parameter u, equal for all curves and compute a point in each curve.

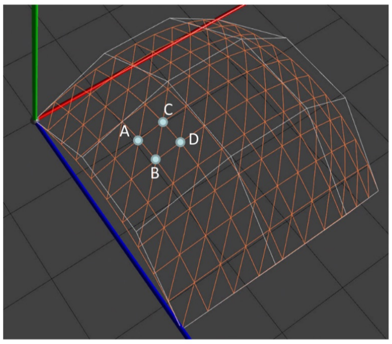
→ consider the resulting 4 points as the control points of a new Bezier curve. Now select a curve for parameter v and the result is a point in the patch p(u,v).

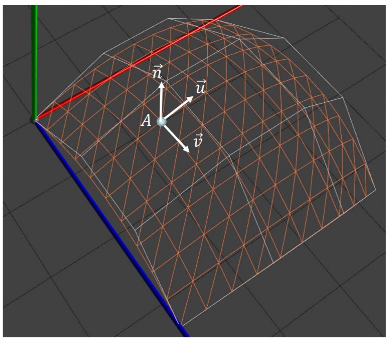
$$p(u,v) = [u^3 \quad u^2 \quad u \quad 1] M \begin{bmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{bmatrix} M^T \begin{bmatrix} v^3 \\ v^2 \\ v \\ 1 \end{bmatrix}$$

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

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

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




→ Building a triangulation for the patch
- white lines are control points
- level of tessellation (divisions) = 10

$A = p(0.2, 0.4)$
$B = p(0.2, 0.5)$
$C = p(0.3, 0.4)$
$D = p(0.3, 0.5)$

$A = (0.3, 0.4)$

$\vec{u} = \dfrac{\partial p(0.3, 0.4)}{\partial u}$ $\quad \vec{v} = \dfrac{\partial p(0.3, 0.4)}{\partial v}$

$\vec{n} = \vec{v} \times \vec{u}$