Representation of Curves and Surfaces

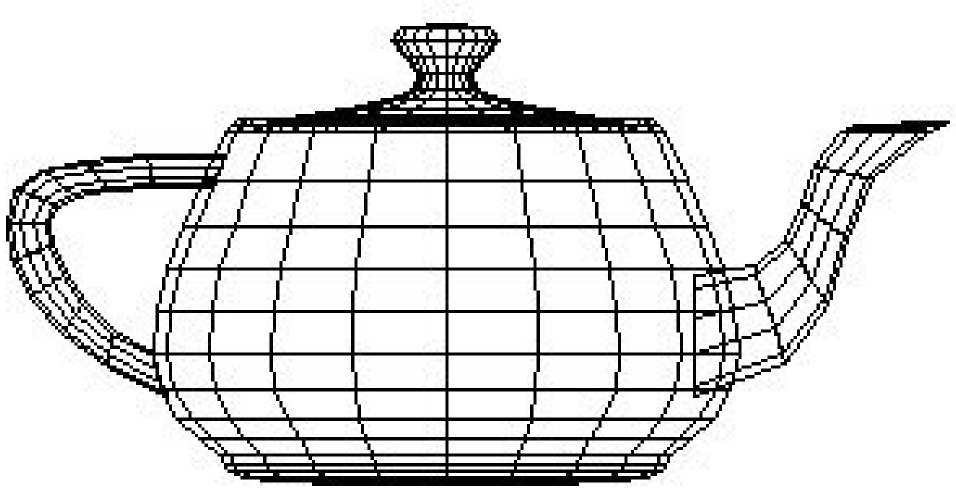
Graphics Systems /
Computer Graphics and Interfaces

Representation of Curves and Surfaces

Representation of surfaces: Allow describing objects through their surface. The three most common representations are:

- Polygonal mesh
- Bicubic parametric surfaces
- Quadratic surfaces

Parametric representation of curves: Important in 2D computer graphics because and because parametric surfaces are a generalization of these curves.



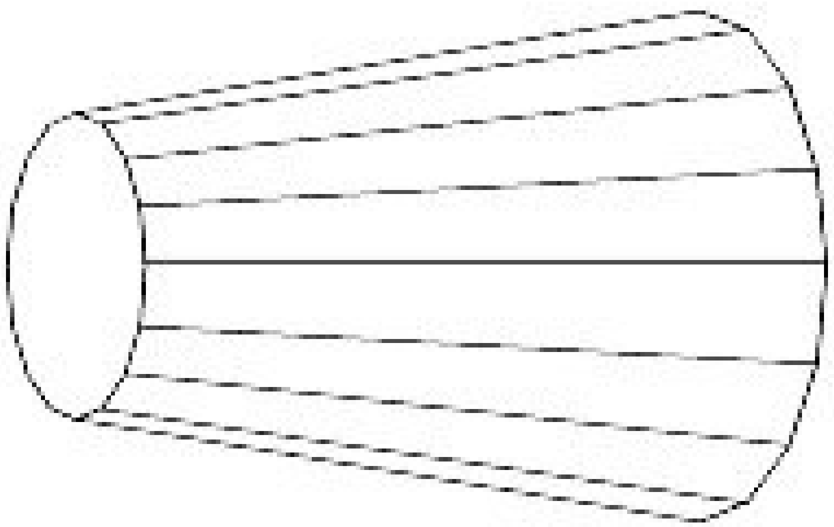
"Tea-pot" modeled by smooth curved surfaces (bicubic).

Reference model in computer graphics, especially for new techniques of realism texture and surface testing.

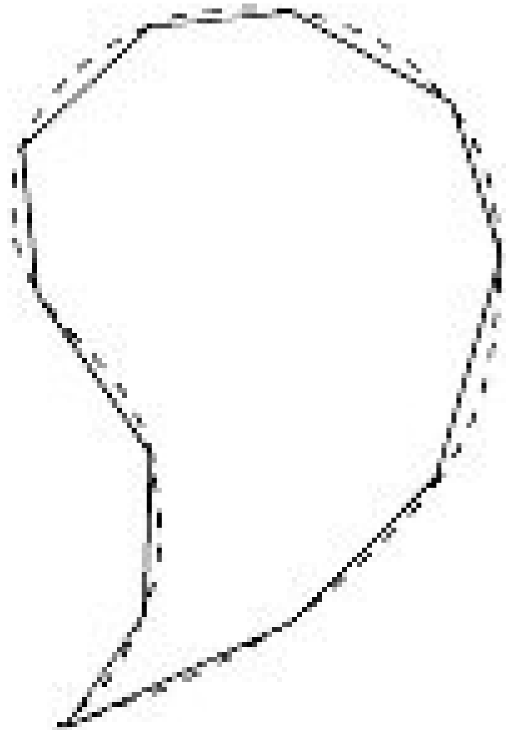
Created by Martin Newell (1975)

Polygonal Mesh

Polygonal Mesh: Is a collection of edges, vertices and polygons interconnected so that each edge is connected only by one or two polygons.



3D object represented by polygon mesh.



Curve ← → polyline

Section of a curved object.

The approximation error can be reduced by increasing the number of polygons, but...

Polygonal Mesh

Characteristics of polygonal mesh:

- An edge connects two vertices.
- A polygon is defined by a closed sequence of edges.
- An edge is connected to one or two (adjacent) polygons.
- A vertex is shared by at least two edges.
- All edges are part of a polygon.

The data structure to **represent the polygonal mesh** can have multiple configurations, which are evaluated by memory space and processing time needed to get a response, for example:

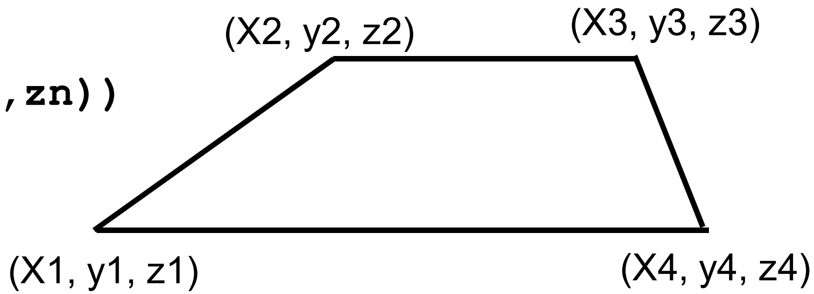
- Get all the edges that join a given vertex.
- Determine the polygons that share an edge or a vertex.
- Determine the vertices that are attached to an edge.
- Determine the edges of a polygon.
- Plot the mesh.
- Identify errors in the representation, as the lack of an edge, vertex or polygon.

Polygonal Mesh

1. Explicit Representation: each polygon is represented as a list of coordinates of its vertices.

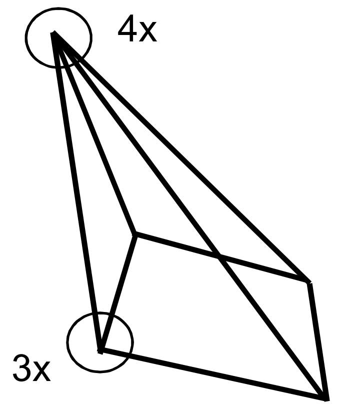
An edge is defined by two consecutive vertices, closing the polygon.

`P=((x1,y1,z1),(x2,y2,z2), ..., (xn,yn,zn))`



Evaluation of the data structure:

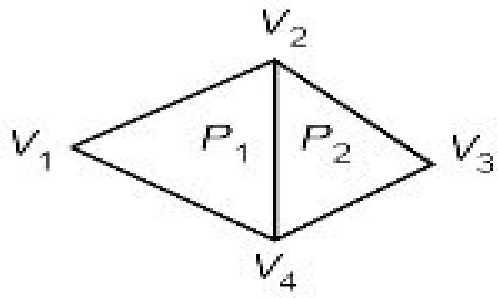
- Large Memory consumption (repeated vertices).
- There is no explicit representation of edges and shared vertices.
- In the graphical representation, the same edge is used (drawn) more than once.
- When you drag a vertex is necessary to know all the edges that share that vertex.



Polygonal Mesh

2. Representation by Pointers to a List of Vertices: each polygon is represented by a list of indices (or pointers) for a list of vertices.

List of Vertices `V=((x1,y1,z1),(x2,y2,z2), ..., (xn,yn,zn))`



`V=(V1,V2,V3,V4)=(x1,y1,z1),(x2,y2,z2), ..., (x4,y4,z4))`

`P1=(1,2,4)`

`P2=(4,2,3)`

Advantages:

– Each vertex of the polygonal mesh is stored only once in memory.
– A coordinate of a vertex is easily changed.

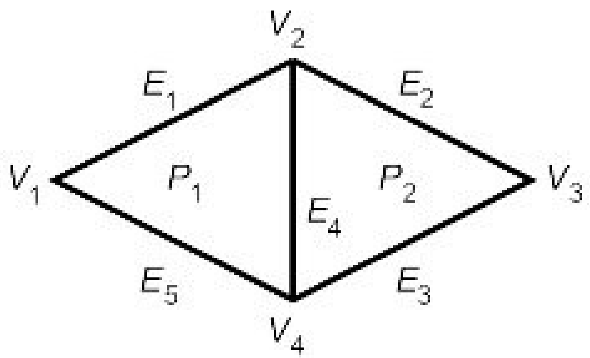
Disadvantages:

– Hard to get the polygons that share a given edge.
– The edges remain being used (drawn) more than once.

Polygonal Mesh

3. Representation by Pointers to a List of Edges: each polygon is represented by a list of pointers to a list of edges, wherein each edge appears only once. In turn, each edge points to two vertices that define it and also stores the polygons which it belongs.

A polygon is represented by **P = (E1, E2, ..., En)** and an edge is represented as **E = (V1, V2, P1, P2)**. If the edge belongs to only one polygon then **P2** is ***null***.



$$V = (V_1, V_2, V_3, V_4) = ((x_1, y_1, z_1), \dots, (x_4, y_4, z_4))$$

$$E_1 = (V_1, V_2, P_1,)$$

$$E_2 = (V_2, V_3, P_2,)$$

$$E_3 = (V_3, V_4, P_2,)$$

$$E_4 = (V_4, V_2, P_1, P_2)$$

$$E_5 = (V_4, V_1, P_1,)$$

$$P_1 = (E_1, E_4, E_5)$$

$$P_2 = (E_2, E_3, E_4)$$

Polygonal Mesh

Advantages:

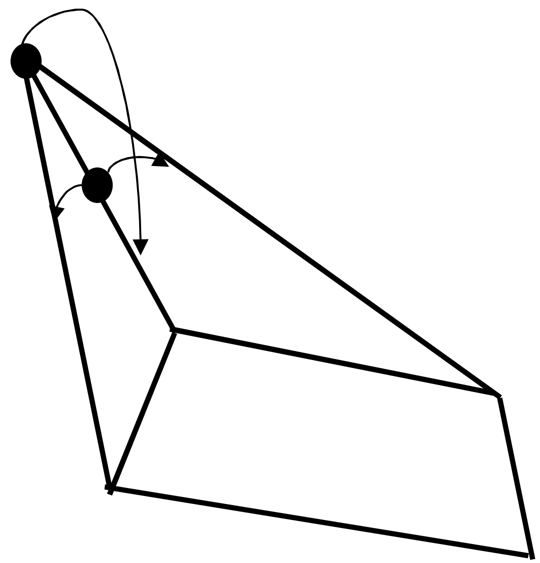
– The graphic design is easily obtained by scrolling through the list of edges. No repetition occurs in using *(*drawing) edges.
– The fill (color) of the polygons works is based on the list of polygons. It is easy to perform *clipping* on the polygons.

Disadvantages:

– Still not easy to determine the edges that share the same vertex.

Baumgart Solution

- Each vertex has a pointer to one of the edges (random) which share this vertex.
- Each edge has pointers to the "next" edge that share that vertex.



Cubic Curves

Motivation: Smooth curves to represent the real world.

- Representation by **polygonal mesh** is a first order approximation:
  - The curve is approximated by a sequence of linear segments.
  - Needs a large amount of data (vertices) to obtain a precise curve.
  - Difficult to change the shape of the curve, ie several points need to be repositioned accurately.

- **Usally: polynomials of degree 3 (Cubic Curves); the complete curve is formed by a set of smaller cubic curves.**
  - **degree <3** offer little flexibility in controlling the shape of the curves and do not permit the interpolation between two points using the definition of the derivative at the end points. A polynomial of degree 2 is specified by three points that define the plane where the curve takes place.
  - **degree > 3** may introduce unwanted oscillations and requires more computational calculation.

Cubic Curves

The representation of the curves is in the PARAMETRIC form :

$x = f_x(t)$, $y = f_y(t)$ ex: $x=3t^3 + t^2$ $y=2t^3+t$

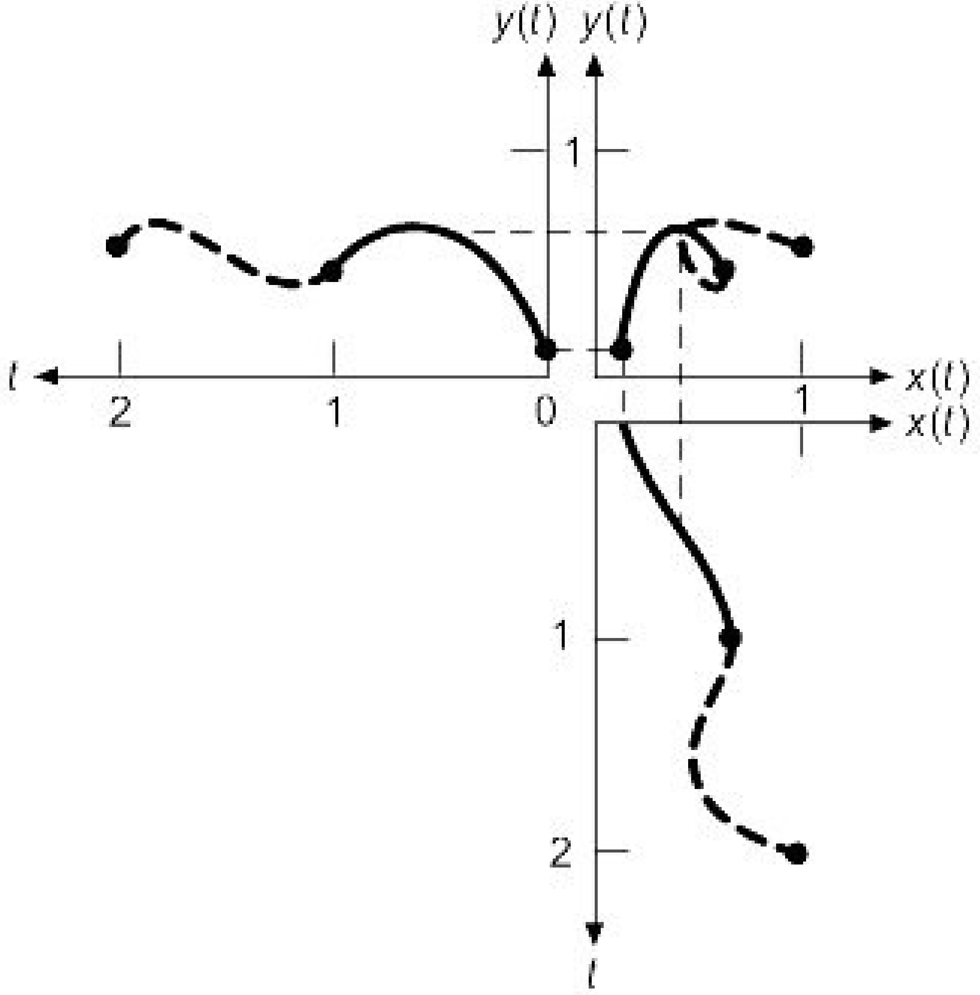
The explicit form : $y=f(x)$ ex: $y=x^3+2x^2$

1. We cannot have multiple values **y** for the same **x**
2. Not possible to describe curves with vertical tangents

The implicit form: $f(x,y)=0$ ex: $x^2+y^2-r^2 = 0$

1. Restrictions are need to be able to model only one part of the curve
2. Difficult to smoothly join two curves

Parametric Cubic Curves



The figure shows a curve formed by two parametric cubic curves in 2D.

Parametric Cubic Curves

General representation of the curve:

$x(t) = a_x t^3 + B_x t^2 + C_x t + d_x$

$y(t) = a_y t^3 + B_y t^2 + C_y t + d_y$

$z(t) = a_z t^3 + B_z t^2 + C_z t + d_z \quad 0 \leq t \leq 1$

Being: $T = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix}$$

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.C$$

Parametric Cubic Curves

The above representation is used to represent a single curve. **How to bring together the various segments of the curve?**

We intend joining a point → geometric continuity and

That have the same slope at the junction → smoothness (continuity of the derivative).

Ensuring continuity and smoothness at the junction is ensured by matching the derivatives (tangent) curves at the junction point. To this end we calculate:

$$\frac{\partial Q(t)}{\partial t} = \left(\frac{\partial x(t)}{\partial t} \quad \frac{\partial y(t)}{\partial t} \quad \frac{\partial z(t)}{\partial t} \right) = \frac{\partial (CT)}{\partial t} = C\frac{\partial T}{\partial t}$$

With: $\frac{\partial T}{\partial t} = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}$

Parametric Cubic Curves

Types of Continuity:

G^0 - geometric continuity, degree Zero → curves just join at a point.

G^1 - Geometric continuity, degree One→ the direction of the tangent vectors is equal.

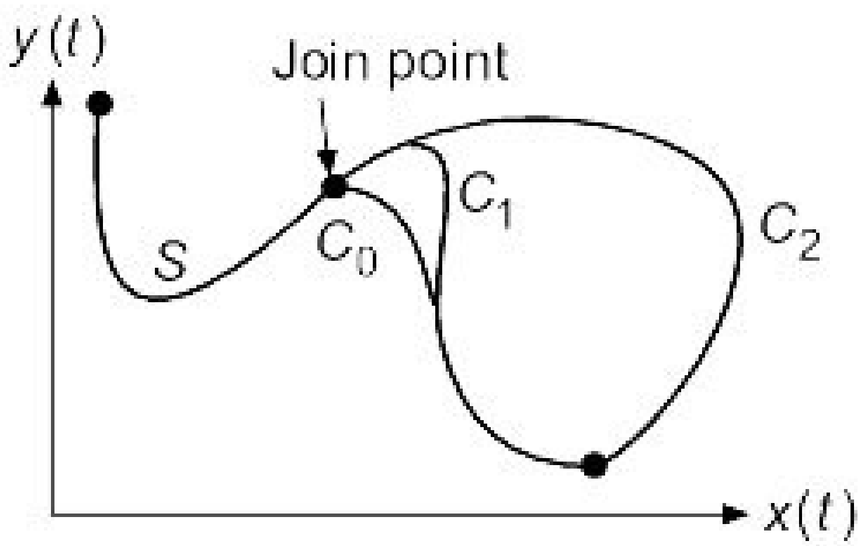
C^1 - Parametric continuity, degree One→ the tangent at the point of junction have the same direction and amplitude (the first derivative equal).

C^n - Parametric continuity, degree N → curves have, at the junction point, all the same derivatives up to order n.

Parametric Cubic Curves

If we consider ***t*** as **time,** the continuity C^1 means that the speed of an object moving along the curve remains continuous

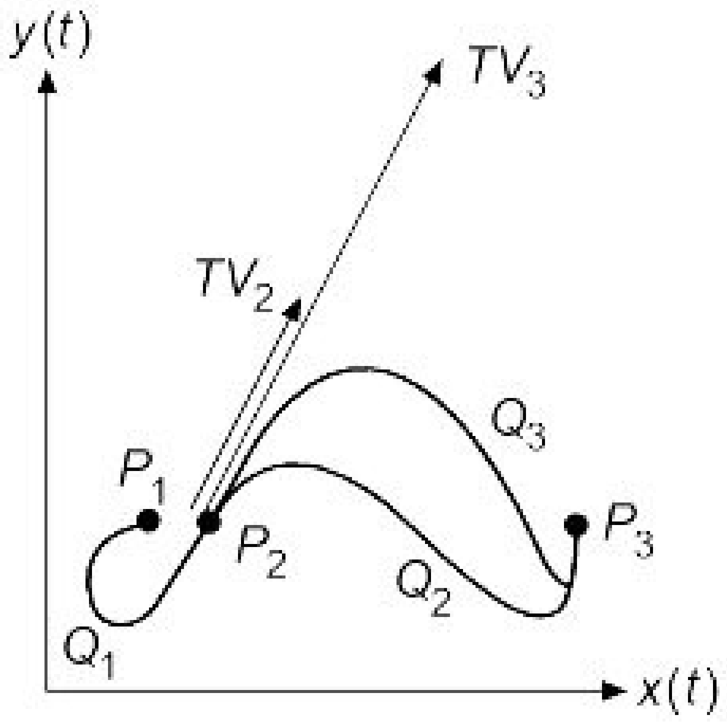
The continuity C^2 imply that the acceleration would also be continuous.



At the junction point of the curve ***S*** the curves $\boldsymbol{C_0}$,$\boldsymbol{C_1}$ and $\boldsymbol{C_2}$ present different continuities

Parametric Cubic Curves

Parametric continuity is more restrictive than the geometric continuity:



For example: C^1 implies G^1

At the junction point P_2 we have:

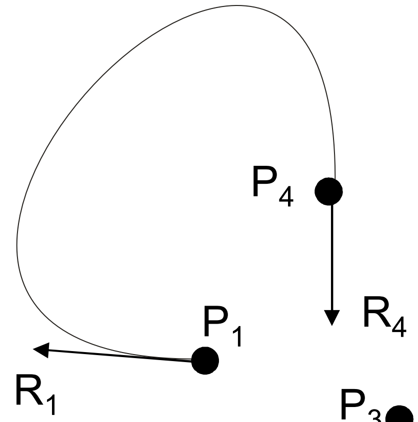
Q_2 and Q_3 are G^1 with Q_1

Only Q_2 is C^1 with Q_1 ($TV_1 = TV_2$)

Parametric Cubic Curves- Types of Curves

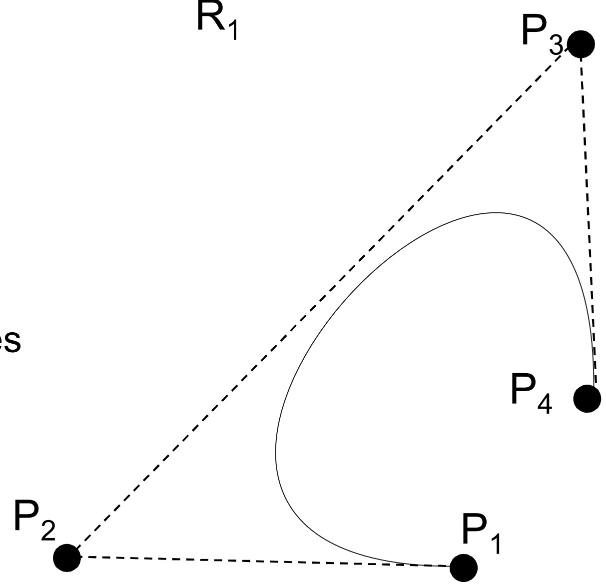
1. Hermite curves

- Continuity **G^1** at junction points
- Geometric vectors:
  - 2 endpoints and
  - The tangent vectors at those points



2. Bezier curves

- Continuity **G^1** at junction points
- Geometric vectors:
  - 2 endpoints and
  - 2 points that control the tangent vectors such extremes



3. Curves Splines

- Very extended family of curves
- Greater control continuity at junction points (C Continuity^1 and C^2)

Common notation

$$Q(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix} = T.C$$

$$Q(t) = T.M.G$$

$$\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \qquad \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \qquad \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

Matrix *T* **Base Matrix** **Geometric Vector**

Base Matrix: Characterizes the type of curve (Hermite, Bezier, etc)

Geometric Vector: Characterizes the geometry of a particular curve.

Common notation

$$Q(t) = T.M.G$$

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} . \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \\ G_4 \end{bmatrix}$$

$$Q(t) = (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}).G_1 + (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{2}).G_2 +$$
$$(t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}).G_3 + (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}).G_4$$

Conclusion 1: Q (t) is a weighted sum of the elements of the geometric vector

Conclusion 2: Weights are cubic polynomials in t → BLENDING FUNCTIONS

(Blending functions) $Q(t) = T.C = T.M.G = B.G$

Hermite curves

$$Q(t) = T.M_H.G_H = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}.M_H.G_H = B_H.G_H$$

$$Q'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}.M_H.G_H$$

Geometric vectors:

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$$

$$\begin{cases} Q(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}.M_H.G_H = P_1 \\ Q(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}.M_H.G_H = P_4 \\ Q'(0) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}.M_H.G_H = R_1 \\ Q'(1) = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix}.M_H.G_H = R_4 \end{cases}$$

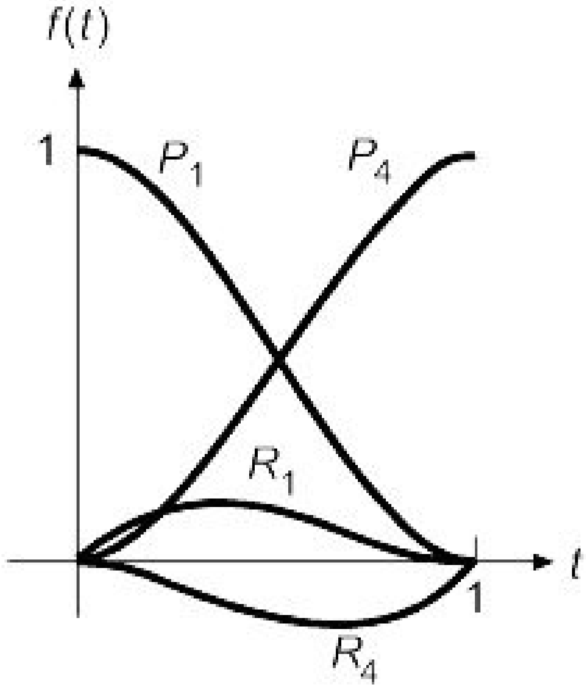
$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}.M_H.G_H = \boxed{\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = G_H} \Rightarrow M_H = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Hermite curves
Blending functions

$$Q(t) = T.M_H.G_H = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} M_H.G_H = B_H.G_H$$

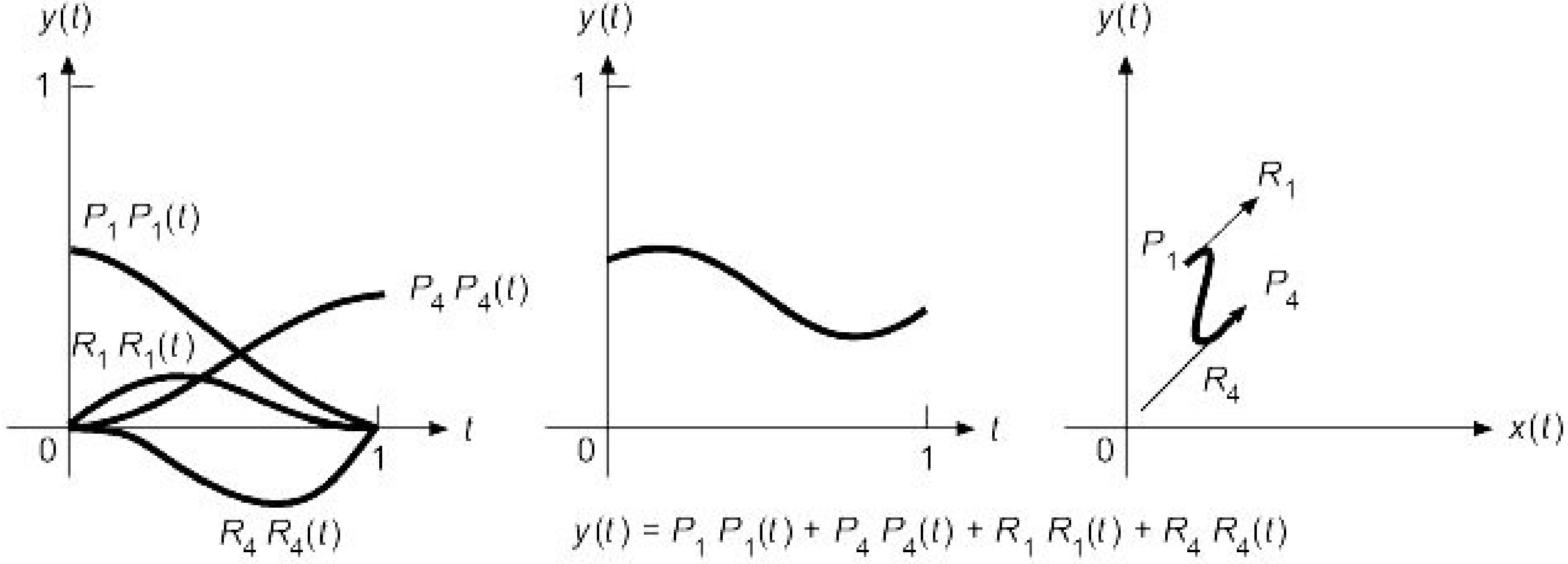
$$M_H = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix}$$

$$\begin{aligned} Q(t) = {} & (2t^3-3t^2+1)P_1 + \\ & (-2t^3+3t^2)P_4 + \\ & (t^3-2t^2+t)R_1 + \\ & (t^3-t^2)R_4 \end{aligned}$$



Blending functions of Hermite curves, referenced by the element of the geometric vector that multiplies, respectively.

Hermite curves - Example



$y(t) = P_1\,P_1(t) + P_4\,P_4(t) + R_1\,R_1(t) + R_4\,R_4(t)$

Left: Blending Functions

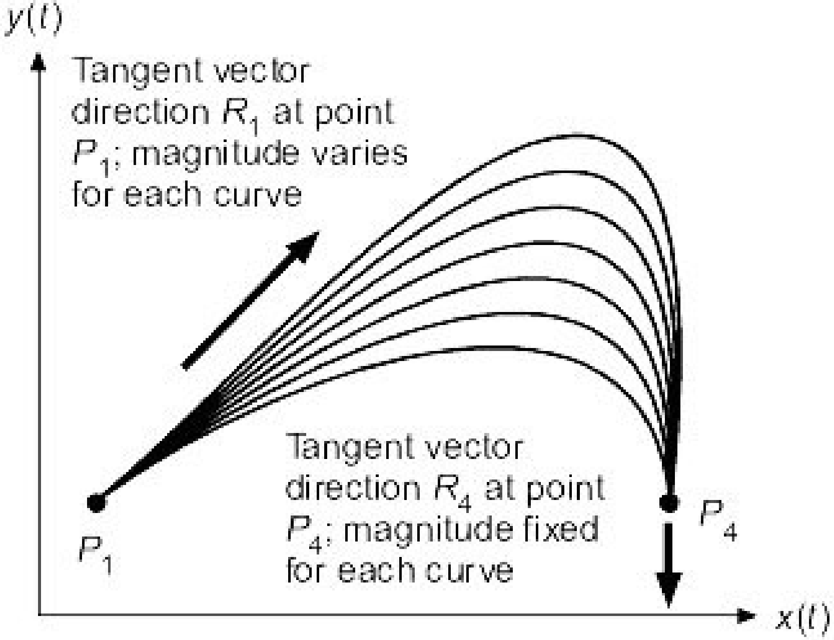
Center: Y (t) = sum of the four functions of the left

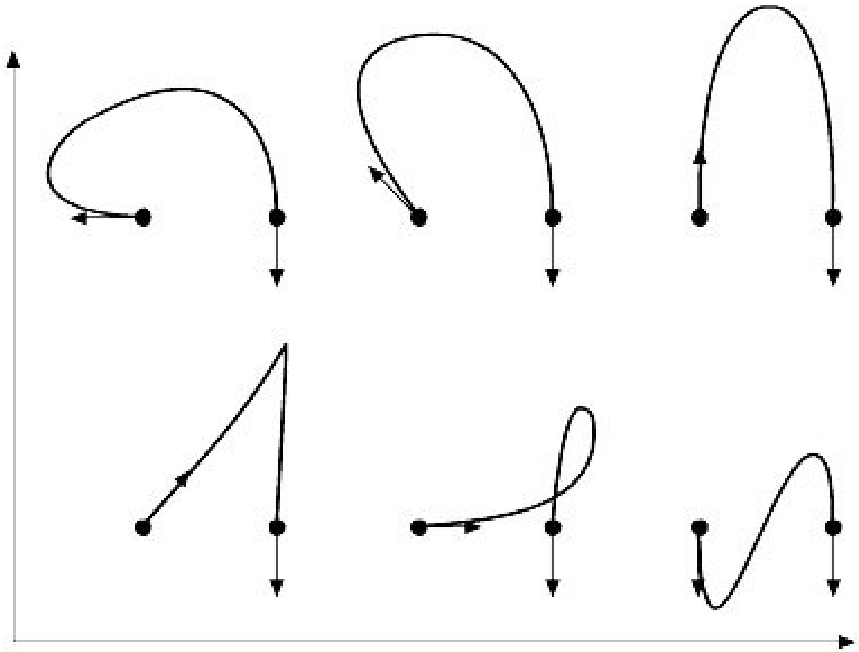
Right: Hermite curve

Hermite curves - Examples

- P_1 and P_4 fixed
- R_4 fixed
- R_1 changing in amplitude

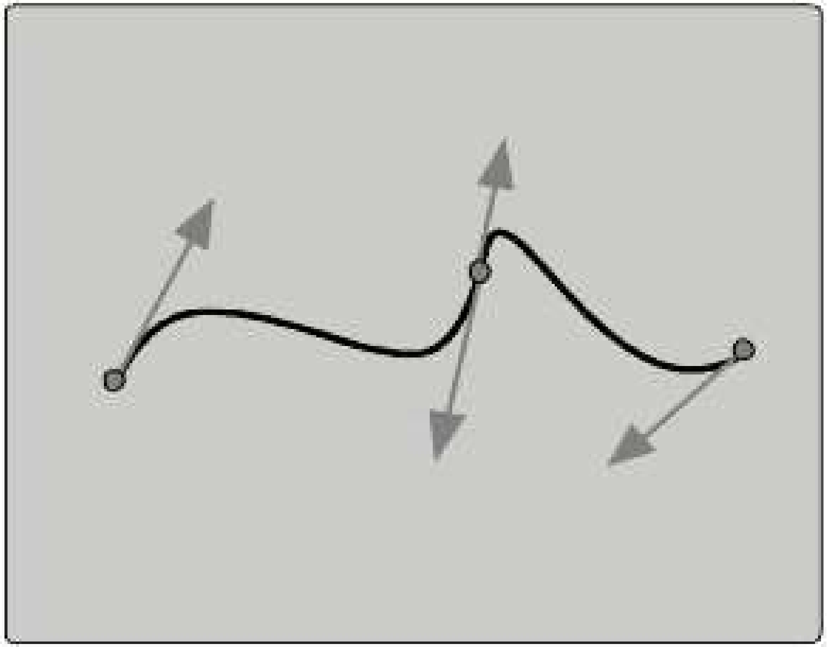
- P_1 and P_4 fixed
- R_4 fixed
- R_1 changing in direction





Hermite curves
Example of Interactive Design

- The extreme points can be repositioned
- The tangent vectors can be changed by pulling the arrows
- The tangent vectors are forced to be collinear (continuity G^1) and R_4 is displayed in the opposite direction (higher visibility)
- It is common to have commands to force continuity G^0, G^1 or C^1



Continuity at the junction:

$$\begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} \Rightarrow \begin{bmatrix} P_4 \\ P_7 \\ K.R_4 \\ R_7 \end{bmatrix}$$

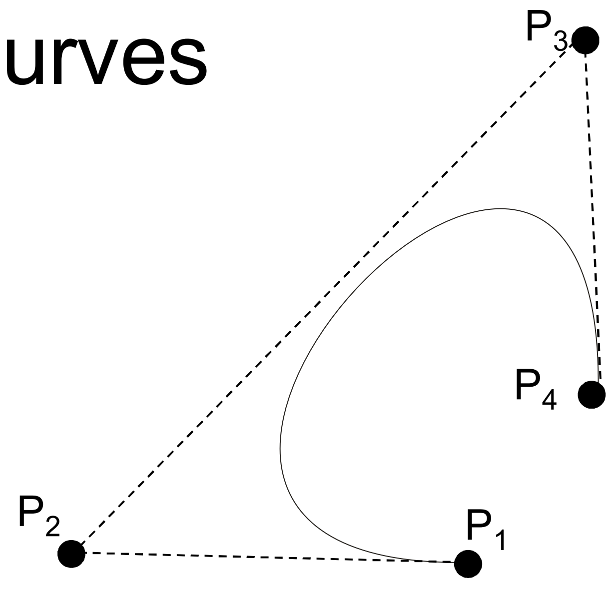
- $K > 0 \rightarrow G^1$
- $K = 1 \rightarrow C^1$

Hermite curves

3. Seja a sucessão $\boldsymbol{C1, C2, C3, C4}$ de curvas de Hermite representadas pelos vectores geométricos juntos. Complete estes com os valores em falta, de forma a obter continuidade do tipo $\boldsymbol{C^1}$ em todos os pontos de junção e justifique os casos em que isso não seja possível, de acordo com os dados fornecidos.

$$C1 = \begin{bmatrix} 0,0 \\ 3,3 \\ 0,2 \\ ?,? \end{bmatrix}; \quad C2 = \begin{bmatrix} ?,? \\ ?,? \\ 2,0 \\ 0,2 \end{bmatrix}; \quad C3 = \begin{bmatrix} 6,6 \\ 3,6 \\ 0,1 \\ 0,-1 \end{bmatrix}; \quad C4 = \begin{bmatrix} 3,3 \\ 6,3 \\ ?,? \\ 2,0 \end{bmatrix}$$

Bezier curves



Geometric Vector:

$$G_B = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

For a given curve we can demonstrate that, compared with G_H:

R_1 = Q '(0) = 3. (P_2- P_1)

R_4 = Q '(1) = 3. (P_4- P_3)

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \cdot \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

$$\boxed{G_H = M_{HB} \,.\, G_B}$$

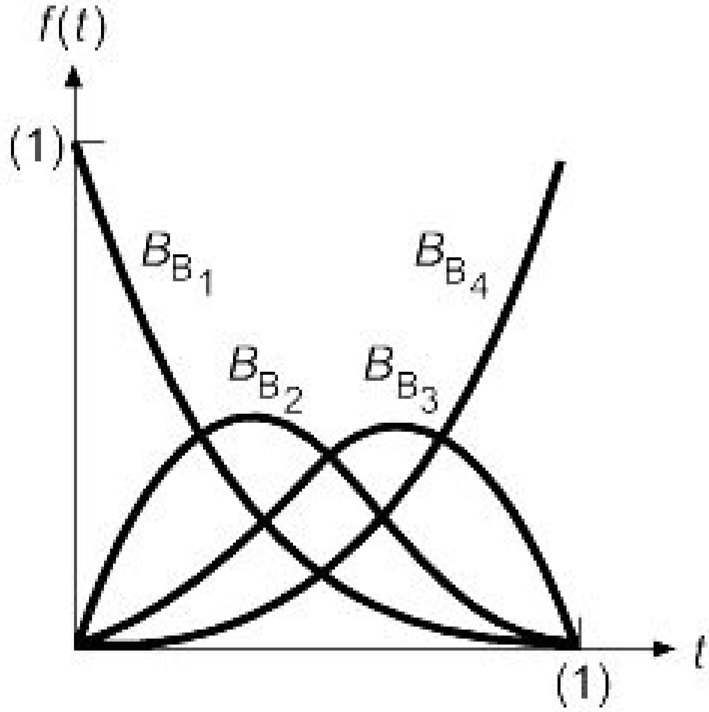
Bezier curves

$$Q(t) = T.M_H.G_H = T.M_H.(M_{HB}.G_B) = T.\underbrace{(M_H.M_{HB})}.G_B$$

- The same Bezier curve representation: Q (t) = T . M_B . G_B

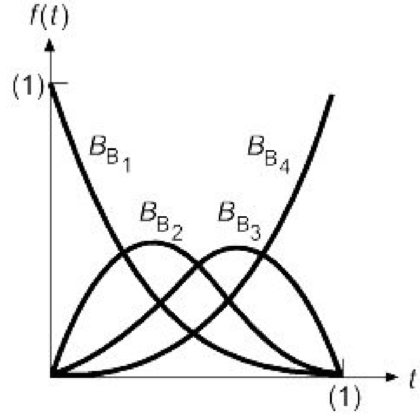
$$M_B = M_H.M_{HB} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{aligned} Q(t) = \; & (1-t)^3 P_1 + \\ & 3t(1-t)^2 P_2 + \\ & 3t^2(1-t) P_3 + \\ & t^3 P_4 \end{aligned}$$



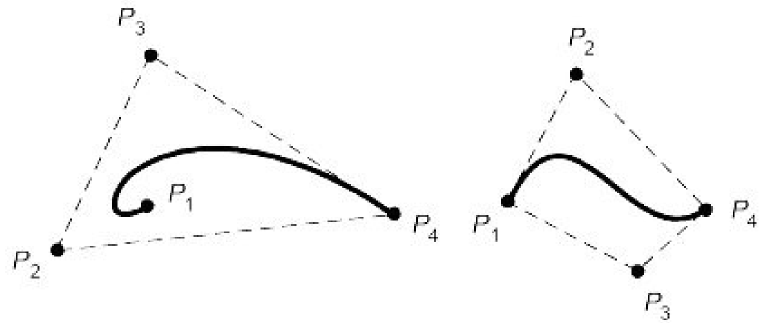
Bezier curves

$$Q(t) = (1-t)^3\ P_1 + 3t(1-t)^2\ P_2 + 3t^2(1-t)\ P_3 + t^3\ P_4$$



Additional information about the blending functions :

- For $t=0$ $Q(t)=P_1$; For $t=1$ $Q(t)=P_4$ → The curve goes through P_1 and P_4
- The sum at any point is 1.
- We can see that Q(t) is a **weighted average of the four control points**; then the **curve is contained within the convex polygon** defined by these points, called the "convex hull".



Bezier curves

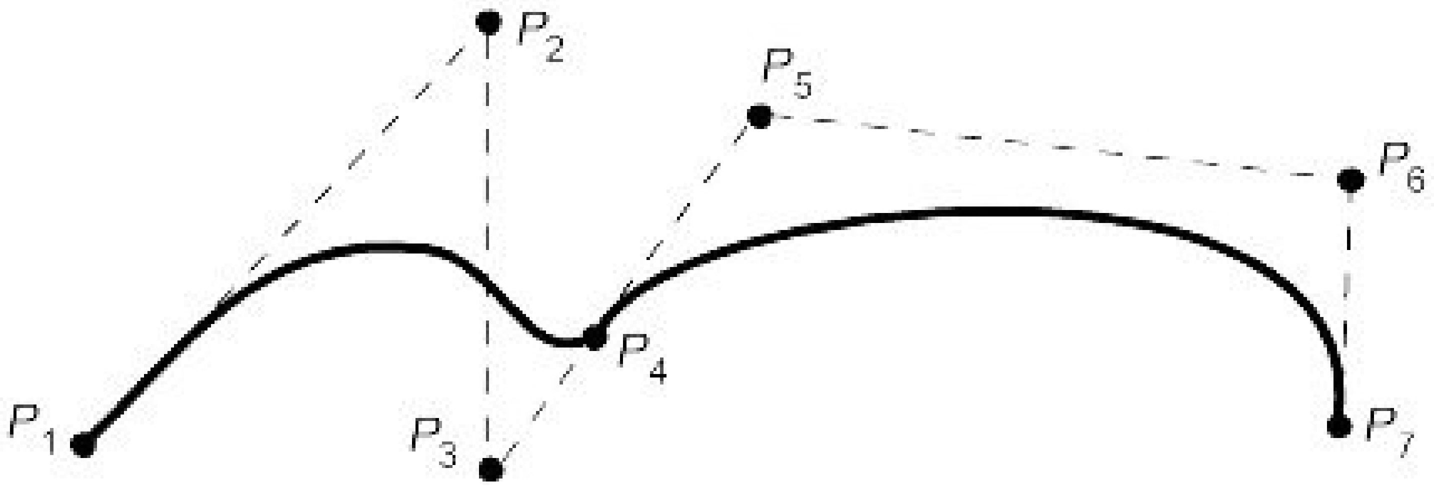
Continuity of Bezier curves

Continuity G^1:

$P_4 – P_3 = K.(P_5 – P_4)$ with $K > 0$ i.e. P_3P_4 and P_5 must be collinear

Continuity C^1:

$P_4 - P_3 = K.(P_5 - P_4)$ making $K = 1$



Drawing Cubic curves

Two algorithms:

1. Evaluation *x (t), y (t)* and *z (t)* incremental values of *t* between 0 and 1.
2. Subdivision of the curve: Casteljau Algorithm

1. Evaluation of *x(t), y(t)* and *z(t)*

   It is possible to decrease the number of operations, from 11 multiplications and 10 additions to 9 and 10, respectively.

   $f(t) = at^3 + bt^2 + ct + d = ((at + b).t + c).t + d$

2. Casteljau algorithm

   Perform the recursive subdivision of the curve, stopping only when the curve in question is sufficiently "flat" and “small” to be able to be approximated by a line segment.

   Efficient Algorithm: it requires only 6 *shifts* and 6 additions in each division.

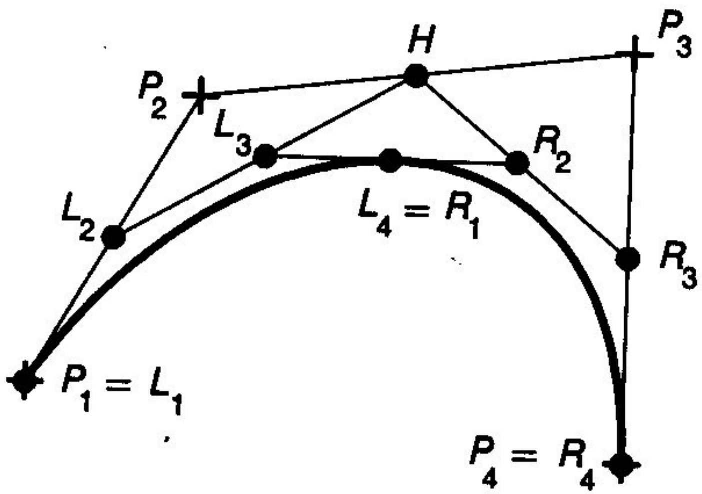
Drawing of Cubic curves - Casteljau Algorithm

Possible stop criteria:

- The curve in question is sufficiently "flat" to be able to be approximated by its convex hull.
- The four control points are in the same pixel.

$L_2 = (P_1+P_2)/2$, $H=(P_2+P_3)/2$, $L_3 = (L_2+H)/2$, $R_3 = (P_3+P_4)/2$

$R_2 = (H+R_3)/2$, $L_4 = R_1 = (L_3 + R_2)/2$



Drawing of Cubic curves

Calteljau Algorithm

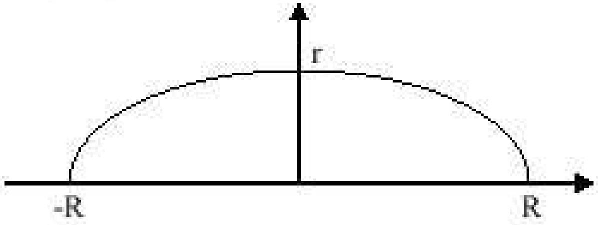
```
void DrawCurveRecSub(curve,ε)
{    if (Straight(curve,ε))
         DrawLine(curve);
     else {
         SubdivideCurve(curve, leftCurve, rightCurve);
         DrawCurveRecSub(leftCurve, ε);
         DrawCurveRecSub(rightCurve, ε);
     }
}
```

Exercise

6. Determine as posições dos quatro pontos de controlo de uma curva de Bézier equivalente à elipse da figura junta:

a)- Analiticamente.

b)- Usando métodos baseados no algoritmo de Casteljou.



Cubic Surfaces

Cubic surfaces are a generalization of cubic curves. The equation of the surface is obtained from the equation of the curve:

Q (t) = T. M. G, being G constant.

Switch to the variable *s*:*Q (s) = S. M. G*

By varying the points of the vector G3D eométrico along a path parameterized by *t* are obtained:

$$Q(s,t) = S.M.G(t) = S.M.\begin{bmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{bmatrix}$$

The geometrical matrix is composed of 16 points.

Surface Hermite

For the x coordinate:

$$x(s,t) = S.M_H.G_{Hx}(t) = S.M_H.\begin{bmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{bmatrix}_x$$

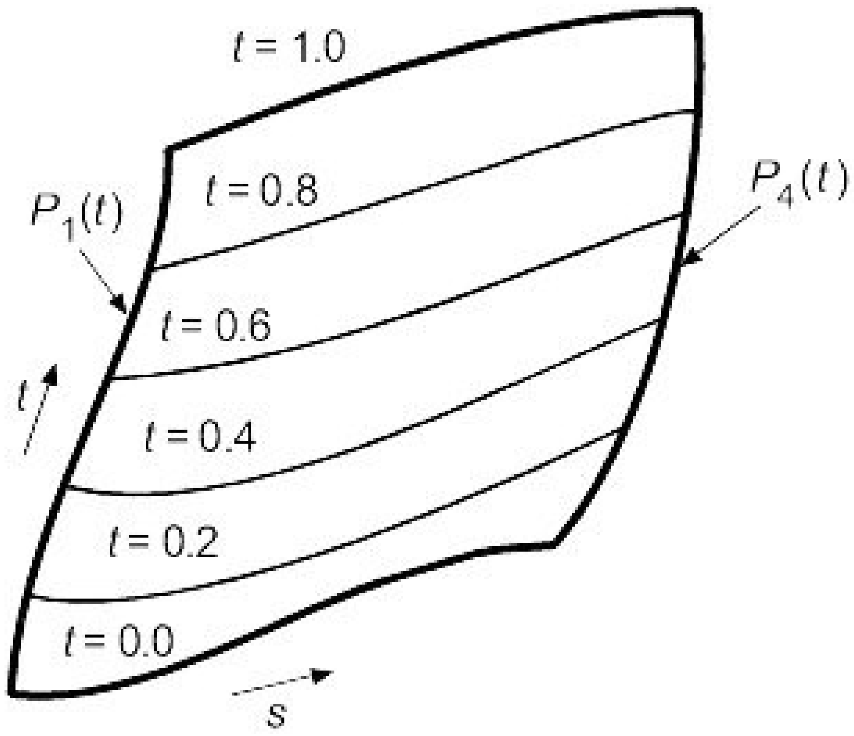
$$P_{1x}(t) = T.M_H.\begin{bmatrix} g_{11} \\ g_{12} \\ g_{13} \\ g_{14} \end{bmatrix}_x \quad P_{4x}(t) = T.M_H.\begin{bmatrix} g_{21} \\ g_{22} \\ g_{23} \\ g_{24} \end{bmatrix}_x \quad R_{1x}(t) = T.M_H.\begin{bmatrix} g_{31} \\ g_{32} \\ g_{33} \\ g_{34} \end{bmatrix}_x \quad R_{4x}(t) = T.M_H.\begin{bmatrix} g_{41} \\ g_{42} \\ g_{43} \\ g_{44} \end{bmatrix}_x$$

$$\begin{bmatrix} P_1(t) \\ P_4(t) \\ R_1(t) \\ R_4(t) \end{bmatrix}_x = \underbrace{\begin{bmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{24} \\ g_{41} & g_{42} & g_{43} & g_{24} \end{bmatrix}}_{G_{Hx}} .M_H^T.T^T = G_{Hx}.M_H^T.T^T$$

It is concluded that:

$$x(s,t) = S.M_H.G_{Hx}.M_H^T.T^T$$

Surface Hermite



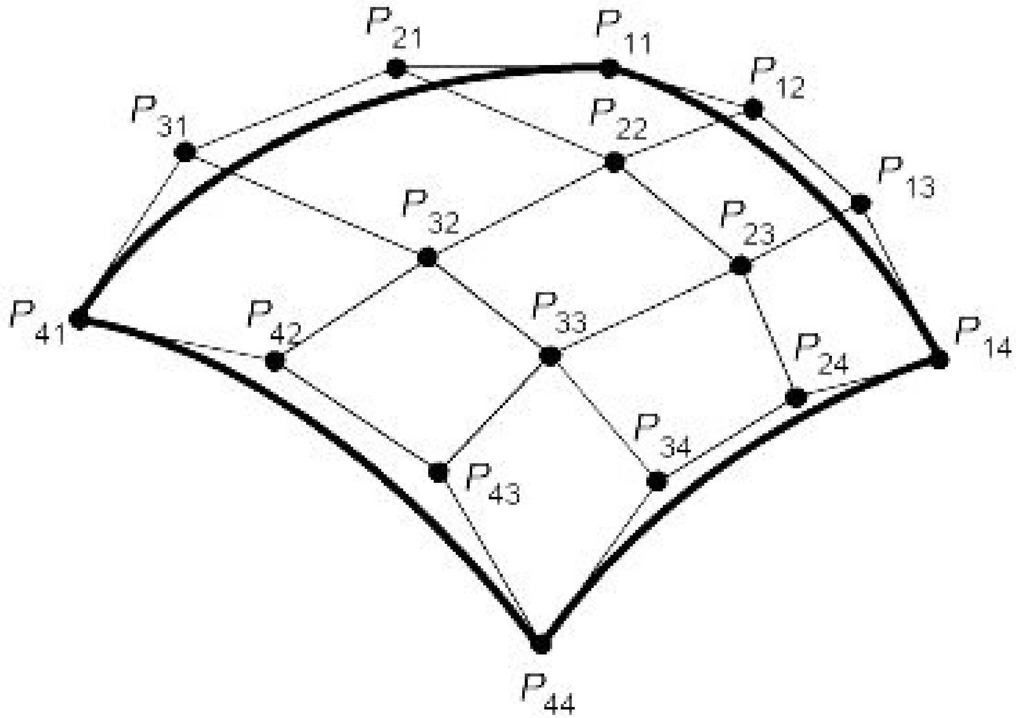
Bezier Surface

The equations for the Bezier surface can be obtained in the same way that the Hermite, resulting in:

$$x(s,t) = S.M_B.G_{Bx}.M_B^T.T^T$$

$$y(s,t) = S.M_B.G_{By}.M_B^T.T^T$$

$$z(s,t) = S.M_B.G_{Bz}.M_B^T.T^T$$



The geometric matrix has 16 control points.

Bezier Surface

Continuity *C^0 and G^0* is obtained by matching the four points of border control: $P_{14}P_{24}P_{34}P_{44}$

For *G^1* should be collinear:

$P_{13}P_{14}$ and P_{15}

$P_{23}P_{24}$ and P_{25}

$P_{33}P_{34}$ and P_{35}

$P_{43}P_{44}$ and P_{45}

and

$(P_{14}-P_{13}) / (P_{15} - P_{14}) = K$

$(P_{24}-P_{23}) / (P_{25} - P_{24}) = K$

$(P_{34}-P_{33}) / (P_{35} - P_{34}) = K$

$(P_{44}-P_{43}) / (P_{45} - P_{44}) = K$