# Bezier Curves and Surfaces

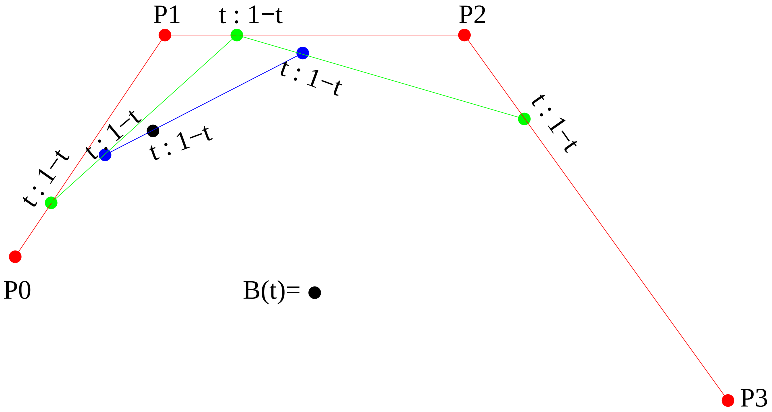
Andrzej Szymczak

November 18, 2007

# Curves

- We will think of curves as trajectories of a moving point over an interval of time
- Curves can be defied by specifying the location $c(t)$ of the moving point at time $t$, i.e. a function from an interval to $\mathbf{R}^n$
- Large number of curve types used in Computer Aided Design
- Curves can be classified based on the form of $c(t)$
  - Polynomial
  - Rational (invariant under perspective projection)
  - Trigonometric
  - ....
- Usually, curve segments with a relatively simple form are connected into longer curves (splines)
- Curves are defined by a sequence of control points and are edited by moving the points

# De Casteljau Algorithm



# De Casteljau Algorithm

- Let $P_0, P_1, \ldots, P_n$ be the control points
- The polygonal curve consisting of intervals $P_i P_{i+1}$ is called the *control polygon* (even though it is not a closed polygon)
- De Casteljau algorithm splits the intervals of the control polygon in ratio t:(1-t), where $t \in [0, 1]$ and connects the consecutive control points into a polygonal curve; intervals of that curve are split and split points connected .... until we are left with just one split point
- Polygonal curve with $n$ segments has $n-1$ intervals (and therefore split points too): number of vertices decreases by 1 with each step

# Properties of Bezier Curves

- Starting point: $P_0$
- Endpoint: $P_n$
- Polynomial formula
  - Recursive formula:

$$B_{P_0P_1\dots P_N}(t) = (1-t)B_{P_0P_1\dots P_{N-1}}(t) + tB_{P_1P_2\dots P_N}(t).$$

  - 2,3 and 4 control points:

$$B_{P_0P_1} = (1-t)B_{P_0} + tB_{P_1} = (1-t)P_0 + tP_1$$

---

$$\begin{aligned} B_{P_0P_1P_2} = (1-t)B_{P_0P_1} + tB_{P_1P_2} = \\ = (1-t)((1-t)P_0 + tP_1) + t((1-t)P_1 + tP_2) = \\ = (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2. \end{aligned}$$

---

$$\begin{aligned} B_{P_0P_1P_2P_3} = (1-t)B_{P_0P_1P_2} + tB_{P_1P_2P_3} = \\ \dots\dots\dots = \\ (1-t)^3P_0 + 3(1-t)^2tP_1 + 3(1-t)t^2P_2 + t^3P_3 \end{aligned}$$

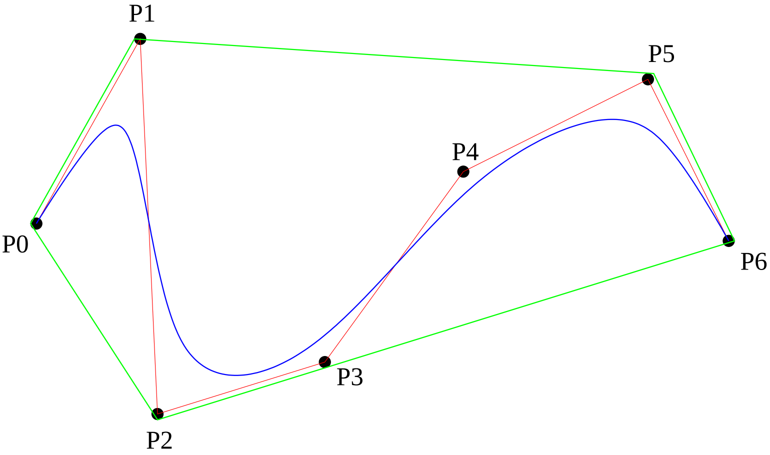
# Properties of Bezier Curves

- General formula for the curve:

$$B_{P_0P_1\ldots P_n}(t) = \sum_{i=0}^{n} \binom{n}{i}(1-t)^{n-i}t^iP_i$$

- Bezier curve with $n+1$ control points is a polynomial curve of degree $n$
- $B_{\ldots}(t)$ is a convex combination of the control points (coefficients, $\binom{n}{i}(1-t)^{n-i}t^i$, sum to $((1-t)+t)^n = 1$)
- Bezier curves have *convex hull property*: the curve is contained in the convex hull of the control points

# Convex hull property



# Tangent lines at the endpoints

- Bezier curve is tangent to the first interval of its control polygon at the starting point
- Bezier curve is tangent to the last interval of its control polygon at the endpoint
- Proof for cubic curves:

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

so:

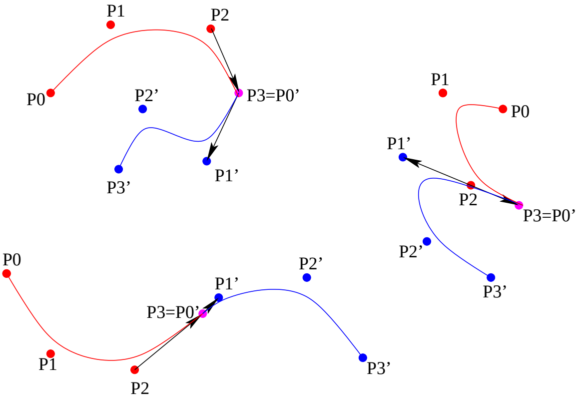
$$B'(0) = -3P_0 + 3P_1 = 3\vec{P_0P_1}$$

and

$$B'(1) = 3P_3 - 3P_2 = 3\vec{P_2P_3}.$$

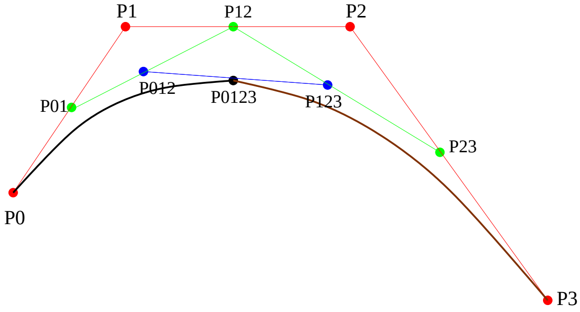
# Joining Bezier curves

- Bezier curves of low degree can be combined to form a complex smooth curve
- At the points where curves meet, velocity vector directions have to match



# Subdivision property

- The Bezier curve with control points $P_0, P_1, P_2, P_3$ is the union of two Bezier curves with control points $P_0, P_{01}, P_{012}, P_{0123}$ and $P_{0123}, P_{123}, P_{23}, P_3$, where all points are split points obtained by applying the de Casteljau algorithm for $t = 0.5$



# Subdivision property

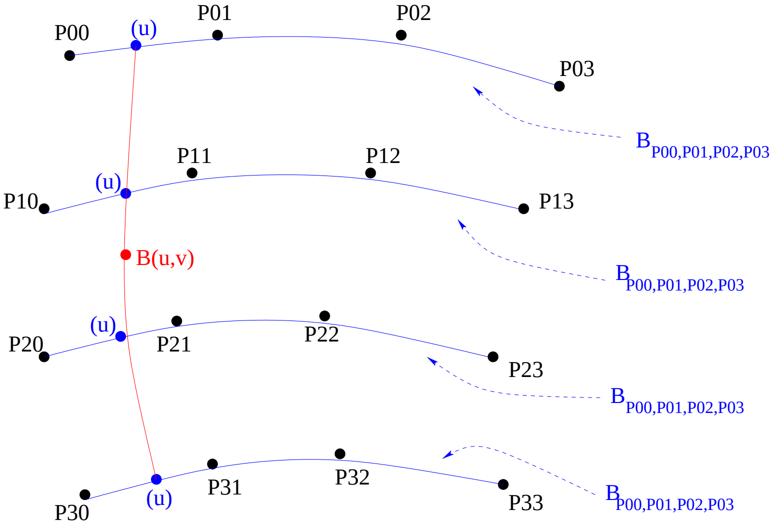
- $P$'s are easy to compute, with only cheap floating point operations (addition, division by 2 - can be done by decrementing the exponent):

$$
\begin{aligned}
P_{01} &= (P_0 + P_1)/2 \\
P_{12} &= (P_1 + P_2)/2 \\
P_{23} &= (P_2 + P_3)/2 \\
P_{012} &= (P_{01} + P_{12})/2 \\
P_{123} &= (P_{12} + P_{23})/2 \\
P_{0123} &= (P_{012} + P_{123})/2
\end{aligned}
$$

# Subdivision algorithm for Bezier curves

```
procedure draw_Bezier ( P0, P1, P2, P3 ):
    if enough subdivisions have been done then
        draw lines P0--P1, P1--P2, P2--P3
        return
    compute P01, P12, P23, P012, P123, P0123;
    draw_Bezier(P0,P01,P012,P0123);
    draw_Bezier(P0123,P123,P23,P3);
```

# Bezier Patches: swept by Bezier curves in 3D



# Bezier Patches

- defined by a $4 \times 4$ array of control points

$$\left[\begin{array}{cccc} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{array}\right].$$

- 

$$B_{[P_{ij}]}(u,v) = B_{B_{P_{00}P_{01}P_{02}P_{03}}(u), B_{P_{10}P_{11}P_{12}P_{13}}(u), B_{P_{20}P_{21}P_{22}P_{23}}(u), B_{P_{30}P_{31}P_{32}P_{33}}(u)}(v).$$

- The Bezier patch consists of all points $B_{[P_{ij}]}(u,v)$ where $u, v \in [0,1]$.