Computer Graphics

Prof. Jibum Kim

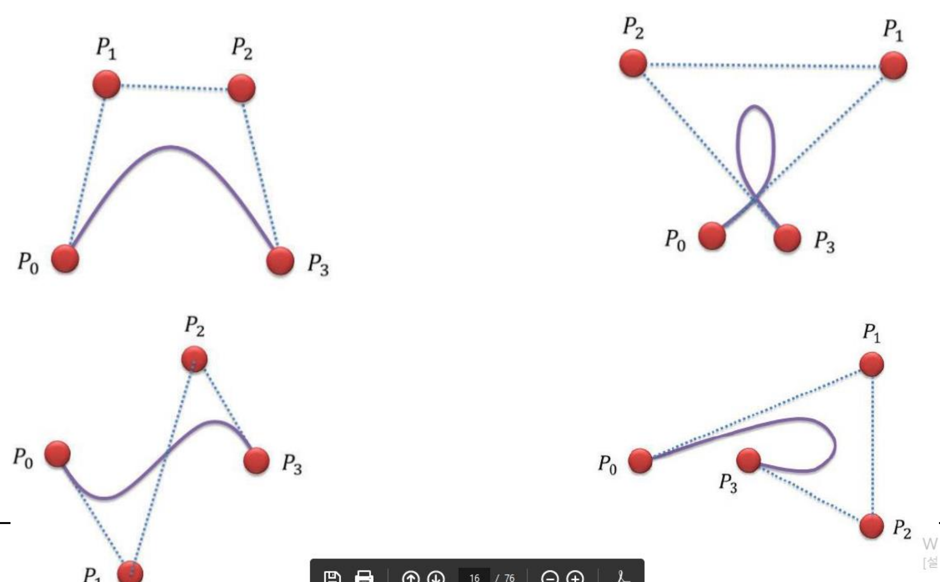
Department of Computer Science & Engineering

Incheon National University

- **Cubic Bezier curves**

- Cubic $\mathbf{B}\acute{e}zier$Curves **(3차 $\mathbf{B}\acute{e}zier$ curve)**
- 가장 흔하게 사용되는 $\mathbf{B}\acute{e}zier$Curves
- Quadratic $\mathbf{B}\acute{e}zier$Curve 보다 하나의 control point를
- 추가한4개의 control points를 사용한 $\mathbf{B}\acute{e}zier$Curve

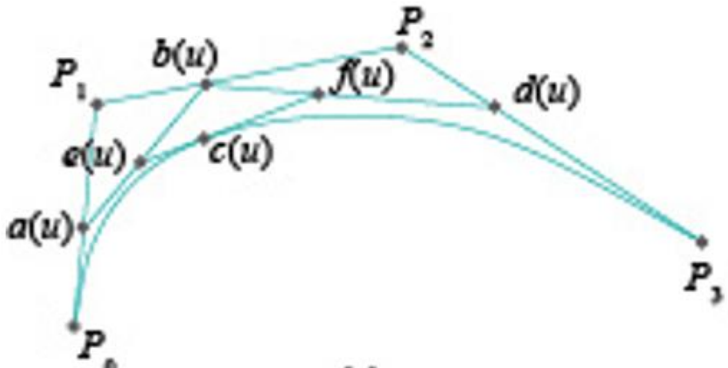
제어점들 P_0, P_1, P_2 및 P_3을 가지는 Bezier curve의 다양



- **Cubic $B\acute{e}zier$ Curves , control points, P0, P1, P2, P3, Given u, $0 \le u \le 1$**

1. $a(u)$ interpolates between P_0 and P_1, $a(u) = (1-u)P_0 + uP_1$
2. b(u) interpolates between P_1 and P_2, $b(u) = (1-u)P_1 + uP_2$
3. d(u) interpolates between P_2 and P_3, d$(u) = (1-u)P_2 + uP_3$
4. $e(u)$ between a(u) and b(u) , $e(u) = (1-u)a(u) + ub(u)$
5. $f(u)$ between b(u) and d(u) , f$(u) = (1-u)b(u) + ud(u)$
6. $c(u)$ between e(u) and f(u) , $c(u) = (1-)e(u) + uf(u)$

$$\mathrm{c}(u) = (1-u)^3P_0 + 3(1-u)^2uP_1 + 3(1-u)u^2P_2 + u^3P_3\ ,\ \ 0 \le u \le 1$$

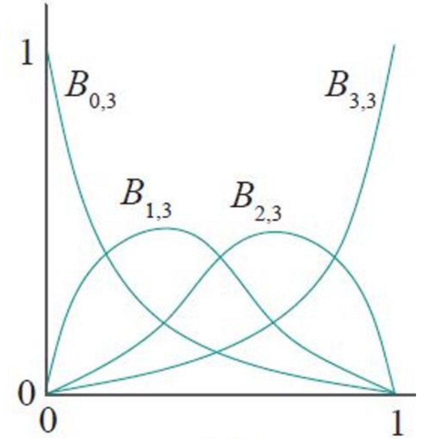


(a)

- 1. c is cubic in u

$$c(u) = (1-u)^3 P_0 + 3(1-u)^2 u P_1 + 3(1-u)u^2 P_2 + u^3 P_3\ ,\ \ 0 \le u \le 1$$

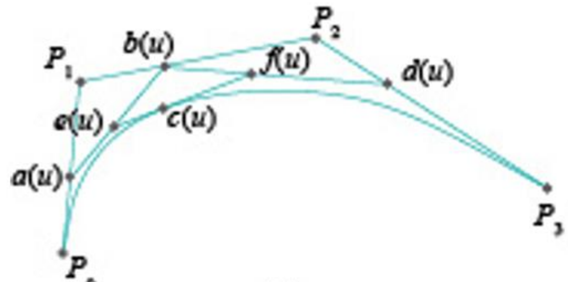
- 2. Blending function은 Berstein polynomial of degree 3
- $c(u) = B_{0,3}(u) P_0 + B_{1,3}(u)\ P_1 + B_{2,3}(u)\ P_2 + B_{3,3}(u)\ P_3$
- Berstein polynomial
- $B_{0,3}(u) = (1-u)^3, B_{1,3}(u) = 3(1-u)^2 u$
- $B_{2,3}(u) = 3(1-u)u^2, B_{3,3}(u) = u^3\ , 0 \le u \le 1$



- **행렬 형태 표현**

- $\mathbf{c}(u) = [\mathbf{P}_0 \quad \mathbf{P}_1 \quad \mathbf{P}_2 \quad \mathbf{P}_3] \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} u^3 \\ u^2 \\ u^1 \\ 1 \end{bmatrix}$

- Cubic Bezier curve의 특징
- 앞의 6 step으로 cubic Bezier curve를 만든것과 다음 3개의 step으로 cubic Bezier curve를 만드는 것은 동일하다
- 1. Draw the quadratic Bezier curve $c_0(u)$ approximating the three control points P_0, P_1, P_2
- 2. Draw the quadratic Bezier curve $c_1(u)$ approximating the three control points P_1, P_2, P_3
- 3. Interpolate along the straight line joining $c_0(u)$ and $c_1(u)$



(a)

- **From only the cases n=1, 2,3, it is clear how the variable part of the Berstein polynomial will change**

In fact, the variable part of $B_{0,n}(u)$ is $(1-u)^n u^0$. (Of course, $u^0 = 1$.) Next, for $B_{1,n}(u)$, the power of $1-u$ decreases by one and that of u increases by one, so its variable part is $(1-u)^{n-1}u^1$. And, so it continues, until the variable part of $B_{n,n}(u)$ is $(1-u)^0 u^n$.

How about the *constant coefficients* though? Let's see what they are.

For Bernstein polynomials of degree 1: 1 1

For Bernstein polynomials of degree 2: 1 2 1

For Bernstein polynomials of degree 3: 1 3 3 1

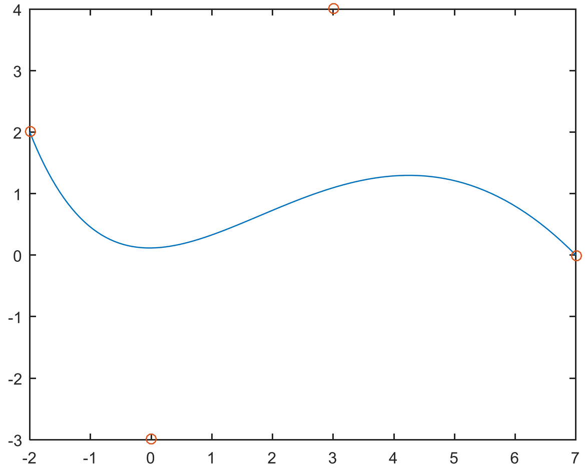
Do you see a pattern? (*Hints*: Pascal's triangle, binomial coefficients.) Can you write down now the parametric equation for a fifth-order Bézier curve, without going through a de Casteljau process?

- Cubic Bezier curves are the ones most commonly used in design applications as three is a sort of “Goldilocks” degree, high enough to allow the curve good flexibility, yet not too high as to be computationally cumbersome

- 네 개의 control point, (-2, 2), (0, -3), (3, 4), (7,0)을 이용한 cubic $Bézier$Curve

$\Rightarrow$ $$\mathbf{c}(u) = (1-u)^3 P_0 + 3(1-u)^2 u P_1 + 3(1-u)u^2 P_2 + u^3 P_3$$

$\Rightarrow$ $$\mathbf{c}(u) = \begin{bmatrix} -2(1-u)^3 + 9(1-u)u^2 + 7u^3 \\ 2(1-u)^3 - 9(1-u)^2 u + 12(1-u)u^2 \end{bmatrix},\ 0 \le u \le 1\ ,$$



■ $Bézier$Curve의 일반화

- 일반적인 $B\acute{e}zier$Curve 표현식
- n+1개의 control points, $P_0, P_1, \ldots, P_n$ 에대하여
- $c(u) = \sum_{i=0}^{n} \binom{n}{i} (1-u)^{n-i} u^i P_i$, 단, $0 \le u \le 1$
- 여기서 $\binom{n}{i} = \frac{n!}{i!(n-i)!}$ 을 binomial coefficient (이항 계수)라 함
- Pascal 삼각형

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1

인천대학교

- **Bernstein polynomial**
- https://en.wikipedia.org/wiki/Bernstein_polynomial
- **정의: ith Bernstein polynomial of degree n**
- $B_{i,n}(u) = \binom{n}{i}(1-u)^{n-i}u^{i}$
- $c(u) = \sum_{i=0}^{n} B_{i,n}(u)P_i$, 단, $0 \leq u \leq 1$
- $\sum_{i=0}^{n} B_{i,n}(u) = 1$

- Bezier curve의 특징들
- If c is the Bezier curve approximating the sequence of n+1 control points $P_0, P_1, \ldots, P_n$, then the following hold
- 1. c is polynomial of degree n in the parameter u
- 2. c is a weighted sum of the control points, where the weight of P_i is its blending function $B_{i,n}(u)$
- 3. (convex hull property) c lies inside the convex hull of $P_0, P_1, \ldots, P_n$

- Convex hull of a set of points $P_0, P_1, \dots, P_n$ is the set of all convex combinations of the points – that is the set of all points given by
- $\sum_{i=0}^{n} \alpha_i P_i$, where $\sum_{i=0}^{n} \alpha_i = 1$ (가중치합), $0 \le \alpha_i \le 1$
- 이유: Bernstein polynomial은 항상 음수가 아니고 합이 1
- Convex hull: smallest convex set that contains it

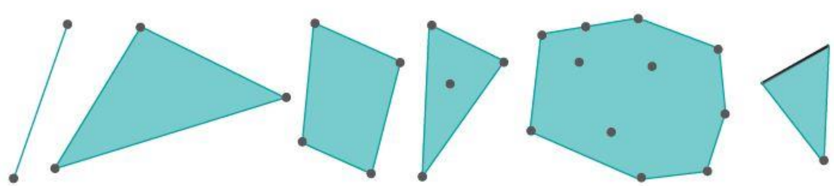
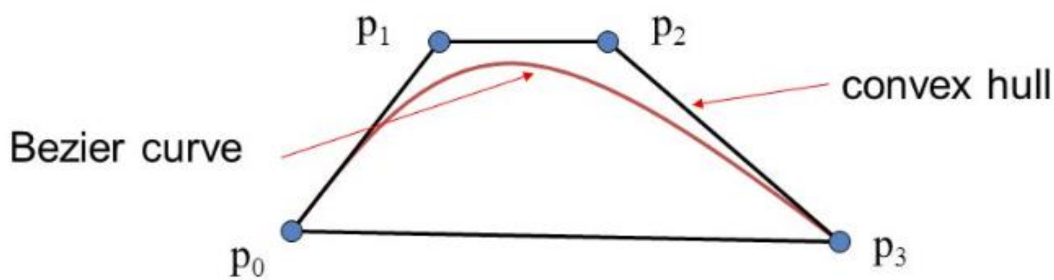


Figure 7.13: Convex hulls.



- 4. c interpolates the first and last control points, but not necessarily intermediate ones
- 5. (Affine invariance) Affine invariance means that the transformed curve is identical to the curve based on the transformed control points
- 의미: 어떤 control points들로 이루어진 Bezier curve를 affine 변환시킨다고 할때 먼저 최초 control points로 Bezier curve를 그리고 이것을 affine 변환하지 않고 control points 자체를 affine 변환시키면된다

- 예: The transformations

```
glScalef(1.0, 2.0, 2.0);
glTranslatef(2.0, 3.0, 0.0);
```

applied to the cubic Bezier curve with control points

$\begin{bmatrix} 2 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 3 \\ 2 \end{bmatrix}, \begin{bmatrix} -2 \\ 7 \\ -1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 4 \end{bmatrix}$. Descirbe the resulting curve

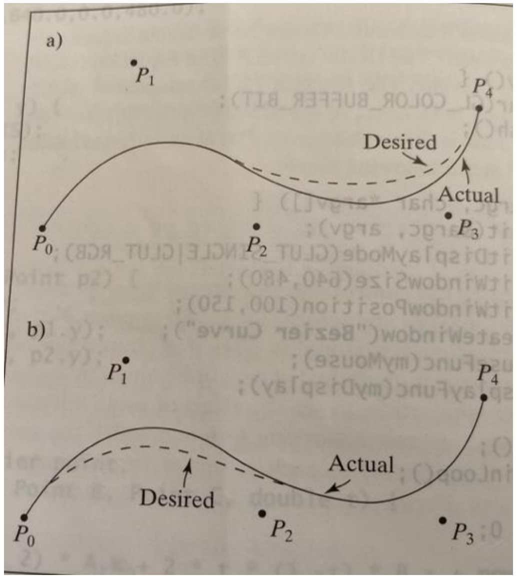
- **Bezier curve 구현 예**
- **Bezier curve can be drawn by an approximating polyline (**glBegin(GL_LINES);)
- **It uses Point class for taking the points**
- **bezier-curve/bezeir-curves.cpp at master · detel/bezier-curve · GitHub**

- 교재 Bezier curve 예
- 위/아래 키를 누르면 Bezier curve의 차수를 정할 수 있고, enter키를 누른 후에
- Space 키를 눌러서 제어점을 선택한 후에 제어점의 위치 조절이 가능하다
- https://www.dropbox.com/s/9yhl9v0xtpvvh7c/bezier_3.txt?dl=0

- **Finding better blending functions**

- It might appear that Bezier curve provide the ultimate tool for designing curves
- One problem is that the degree of the Berstein polynomials used is coupled to the number of control points: a Bezier curve based on L+1 control points is a combination of L-degree polynomials
- High-degree polynomials are expensive to compute and are vulnerable to numerical round-off errors
- We want the designer to be free as many control points as desired, even 40 or more

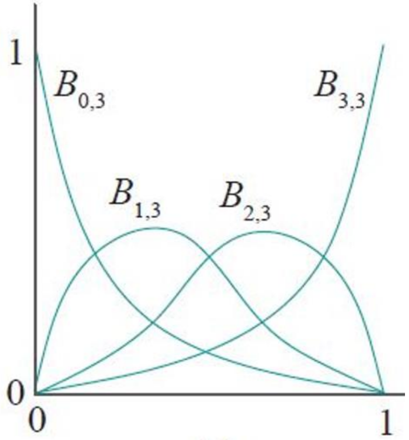
- The problem of local control
- An even more significant problem is that Bezier curve do not offer enough local control of the curve shape
- Below figure shows a situation where five control points are used
- But it deviates (벗어나다) somewhat from the desired cuve near t=1
- To correct this, the user would move P_2 and P_3 up to force the Bezier curve closer to the desired curve
- But this also affects the shape of the first half of the curve



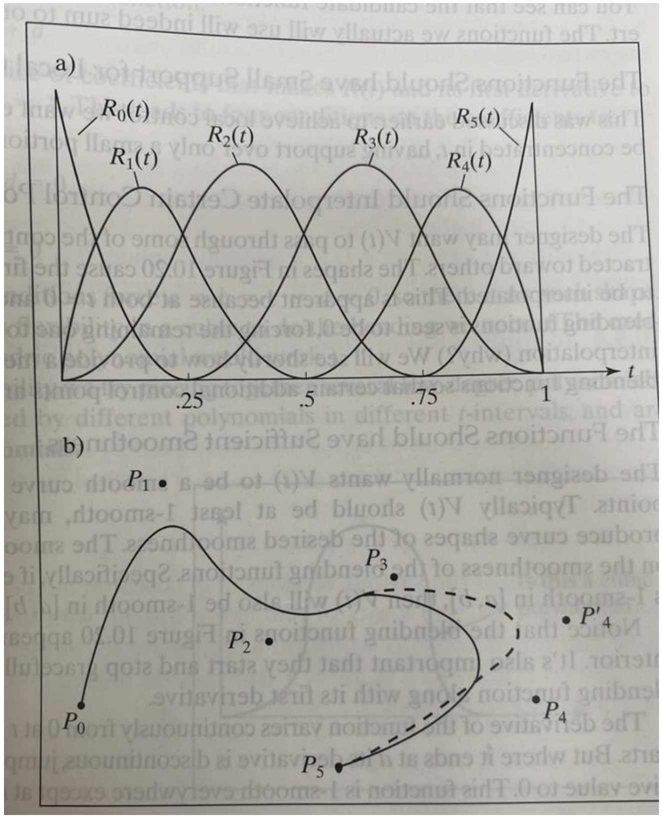
- The problem is that a change to any control points alters the entire curve
- This arises from the nature of Berstein polynomials: each one is "active" (meaning nonzero) over the entire interval [0, 1]
- The interval over which a function is nonzero is often called its support
- Because every Bernstein polynomial has support over the entire interval [0, 1] and the curve is a blend of these functions
- Therefore, adjusting any control points affect

the shape of the curve everywhere,

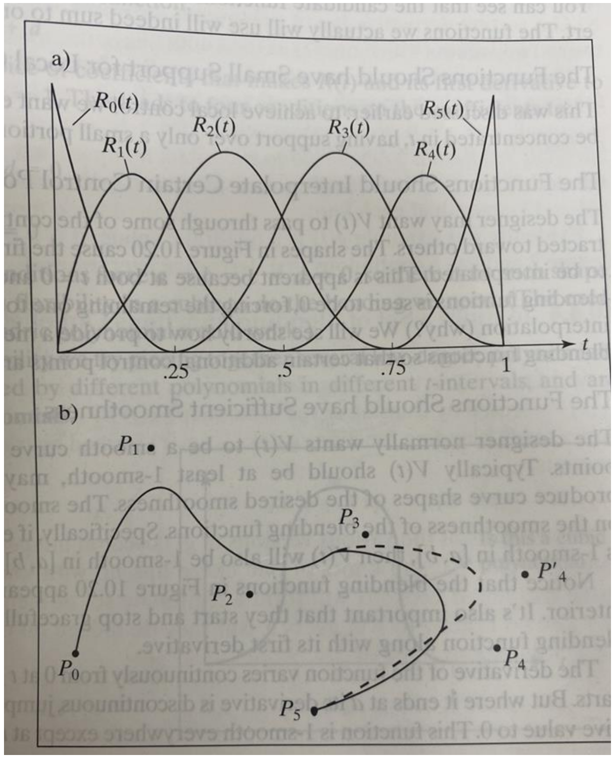
with no local control



- **More favorable set of hypothetical blending functions are drawn**
- **The six blending functions $R_0(t), R_1(t), \ldots, R_5(t)$ has support that is only a part of the interval [0, 1]**
- **In fact, at any value of t, no more than three of the blending functions are active**



- We use the same kind of parametric form as for Bezier curves
- $V(t) = \sum_{k=0}^{5} P_k R_k(t) = P_0 R_0(t) + P_1 R_1(t) + P_2 R_2(t) + P_3 R_3(t) + P_4 R_4(t) + P_5 R_5(t)$
- For example, for all t in [0.75, 1.0], only the points P_3, P_4, and P_5 control the shape of the curve
- If the single control point P_4 is moved to P_4', only a portion of the curve will change (why?)
- Thus, this set of blending functions give some local control



- **Better blending functions and piecewise polynomials**

- Curve의 smoothness 정의
- A curve is 0-smooth in an interval if it is continuous (연속)
- A curve is 1-smooth in an interval if its first derivative (일차 도함수) exists and is continuous throughout the given interval
- A curve is 2-smooth if its first and second derivatives exist and are continuous throughout the given interval

- Blending function의 조건들
- 1. be easy to compute and numerically stable
- simple, minimal rounding error, smallest degree possible
- 2. sum to one at every t in [a, b]
- weighted sum of points at each t
- 3. have small support to offer local control
- 4. be smooth enough
- at least 1-smooth
- 5. interpolate certain control points, chosen by the designer

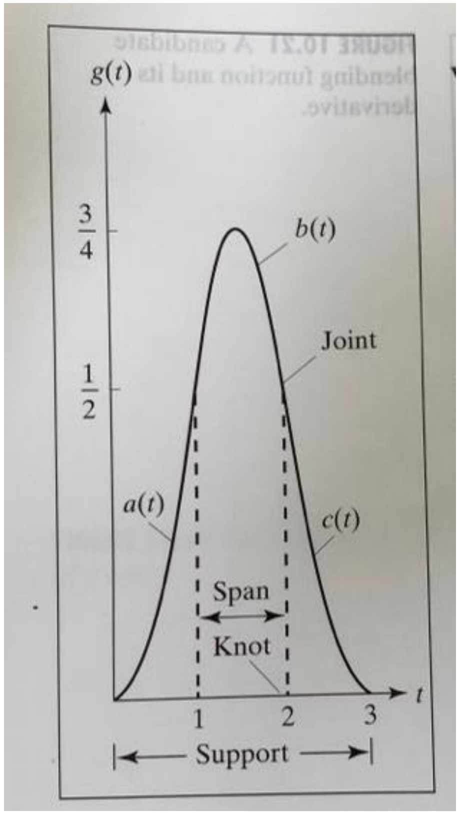
- Piecewise polynomial (조각 다항)
- 앞에서 본 Bernstein polynomial의 문제점들을 인지하고 이를 해결하고자 한다. 어떤 문제점들?
- To attain more flexibility, we try piecing together several low-degree polynomials
- Such curves are defined by different polynomials in different t-intervals and are called piecewise polynomials

- Piecewise polynomial의 예
- $g(t)$ 는 3개의 polynomial segment로 구성됨
- $g(t)$ 의 support는 [0, 3], 각 구간을 span 이라 부름
- Knots are values of t where segments meet
- Joints are values of the function of adjacent segments
- 각 polynomial segment는 저차 다항식임

- $a(t) = \frac{1}{2}t^2, t \in [0,1]$

- $b(t) = \frac{3}{4} - \left(t - \frac{3}{2}\right)^2, t \in [1,2]$

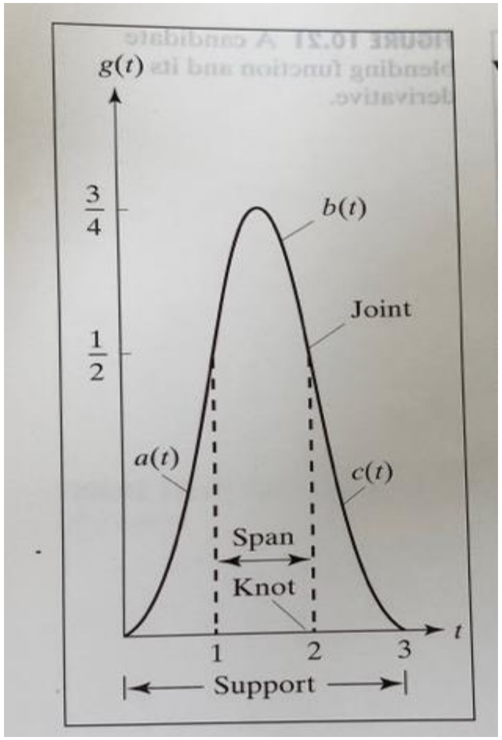
- $c(t) = \frac{1}{2}(3-t)^2, t \in [2,3]$



- The shape of $g(t)$ here is an example of a spline function, a piecewise polynomial function that enjoys enough smoothness

- 정의: An Mth-degree spline function is a piecewise polynomial of degree M that is (M-1)-smooth at each knot

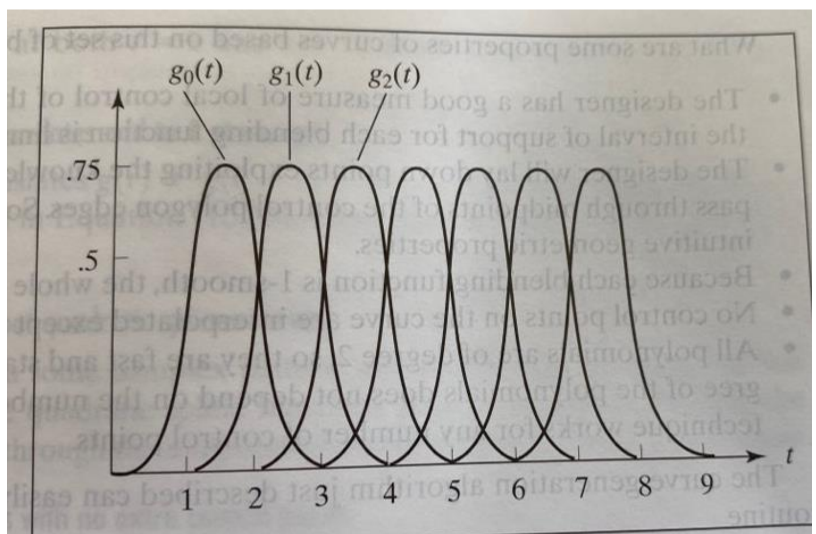
- **Q: 앞의 piecewise polynomial, $g(t)$, 은 continuous every where?**
- **Q: 앞의 piecewise polynomial, $g(t)$,은 1-smooth at each knot?**
- **확인해보자**
- $a(t) = \frac{1}{2}t^2, a'(t) = t, \quad t \in [0, 1]$
- $b(t) = \frac{3}{4} - \left(t - \frac{3}{2}\right)^2, b'(t) = -2t + 3, t \in [1, 2]$
- $c(t) = \frac{1}{2}(3 - t)^2, \mathrm{c}'(\mathrm{t}) = \mathrm{t} - 3, t \in [2, 3]$



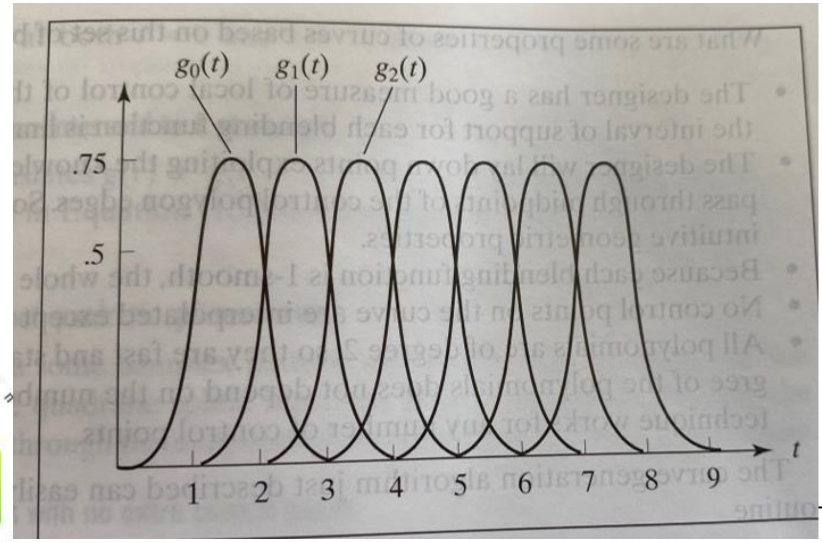
- 이 $g(t)$ 는 **quadratic (2차)-spline** 함수이다
- **Why?**
- **it is a picewise polynomial of degree 2 and has a continuous first derivative everywhere (1-smooth at each knot)**

- **B-spline function**

- 그렇다면 원래 문제로 돌아와서 이러한 $g(t)$ 와 같은 spline 함수를 사용하여 어떻게 blending 함수를 만들 수 있을까?
- 흔히 사용되는 방법: use translated version of $g(t)$
- 즉, $g_k(t) = g(t-k),$ for k=0, 1, …
- It is crucial that we translate each function by an integer, to make the shapes line up properly so they sum to 1



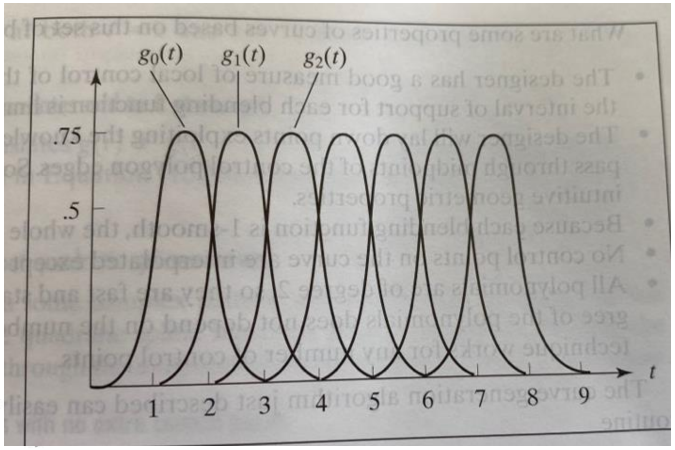
- $g_k(t) = g(t-k)$, for k=0, 1, …6
- Blending function: $\sum_{k=0}^{6} g(t-k) = 1$ for t in [2, 7]
- 1. only values of t between 2 and 7 can be used
- 2. 각 구간 (t가 2에서 7사이)에서 3개의 함수만 active
- 3. t=2, 3, ..7에서 only two of the functions are active and they both have value of 0.5 (다음 페이지 그림 참고)



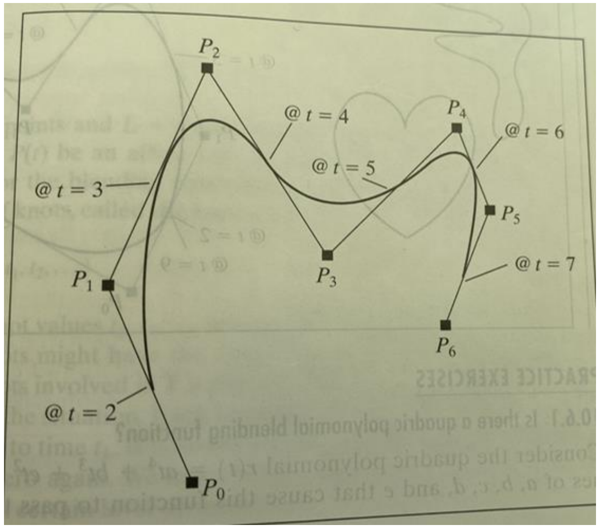
Blending functions representing

a spline curve

- **Finally, the designer chooses seven control points and generates the cruve using the algorithm:**
- $V(t) = \sum_{k=0}^{6} P_k g(t-k)$
- **단,** $g_k(t) = g(t-k)$, **for k=0, 1, …6**
- **Only values of t between 2 and 7 can be used**



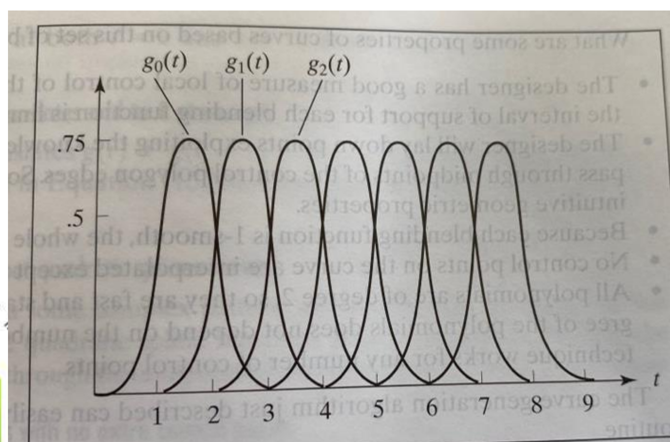
Blending functions

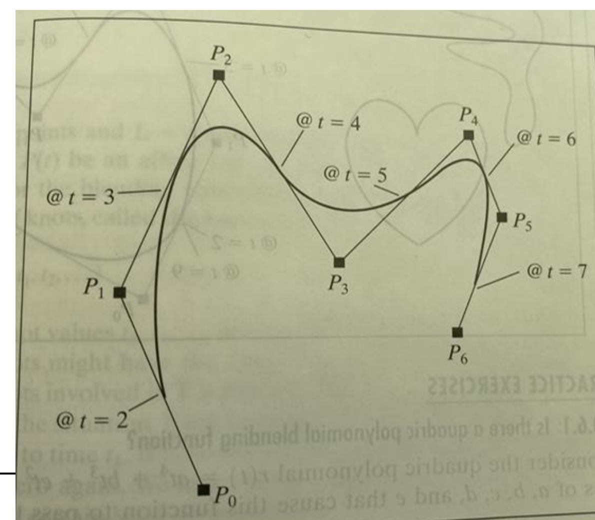


Curve design using translates of g(t)

- **앞의 예와 같은 spline함수를 사용했을 때의 특징**
- **1. Local control 가능: the interval of support for each blending function is limited to length 3 (앞의 그림 확인)**
- **2. Each blending function is 1-smooth, the whole curve is 1-smooth**
- **3. No control points on the curve are interpolated except P0 and P6**
- **4. All polynomials are of degree 2 (low-degree)**

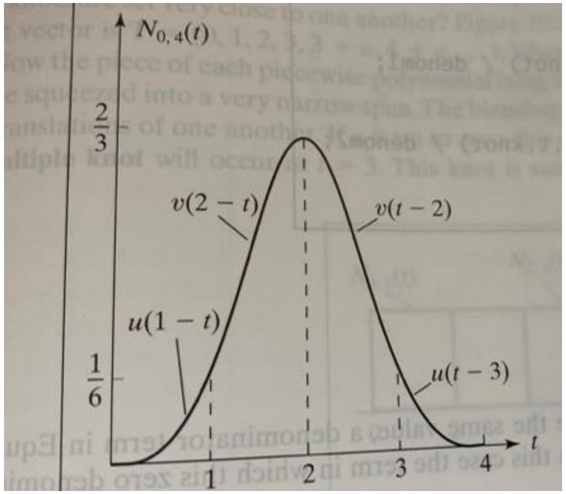
- 5. **The curve must pass through midpoints of the control polygon edges**. So, the algorithm has some intuitive geometric propertis
- 예: t=2인 위치는 어디?
- $V(t) = \sum_{k=0}^{6} P_k g(t-k) = P_O g(t) + P_1 g(t-1) + \cdots$
- $= P_O g_0(t) + P_1 g_1(t) = \frac{1}{2}(P_O + P_1)$, 즉 P_O 와 P_1의 중점에서 시작





- 앞에서 본 spline 함수 ($g(t)$)는 B-spline 함수의 한 예이다. B (basis), 이를 quadratic (2차) B-spline 함수라 한다
- B-spline 함수는 blending function이 smallest support를 갖으면서 greatest local control을 가능하게 한다
- B-spline 함수 중에 가장 널리 쓰이는 형태는 cubic B-spline 함수이다

- Cubic B-spline 함수의 예



- $$g(t) = \begin{cases} u(1-t), 0 \le t \le 1 \\ v(2-t), 1 \le t \le 2 \\ v(t-2), 2 \le t \le 3 \\ u(t-3), 3 \le t \le 4 \\ 0, otherwise \end{cases}$$
- 여기서 $u(t) = \frac{1}{6}(1-t)^3, v(t) = \frac{1}{6}(3t^3 - 6t^2 + 4)$
- t=2를 기준으로 대칭함수

- **이를 앞에서와 같이 t 축에서 정수만큼 translate 시켜서 사용함**