Computer Graphics

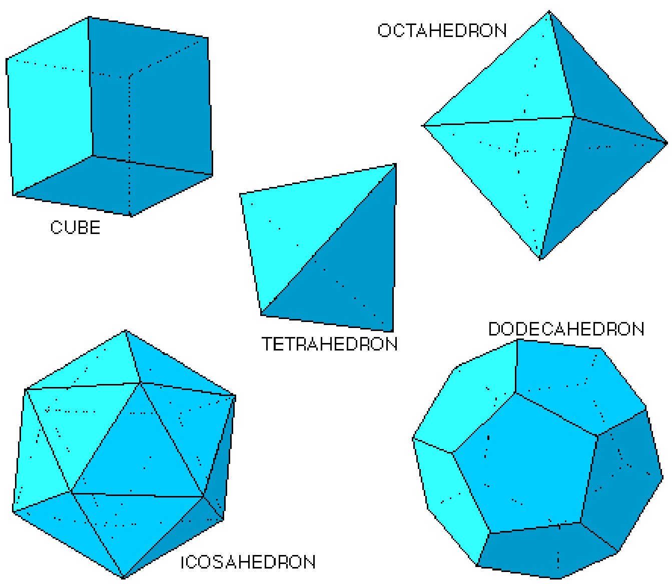
Prof. Jibum Kim

Department of Computer Science & Engineering

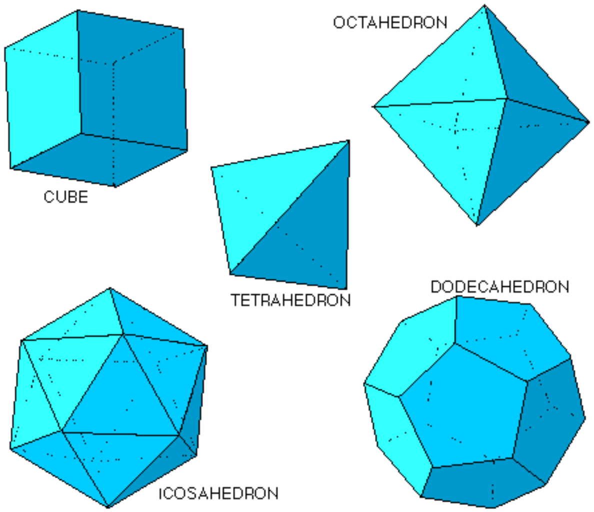
Incheon National University

- **Polyhedra (다면체)**

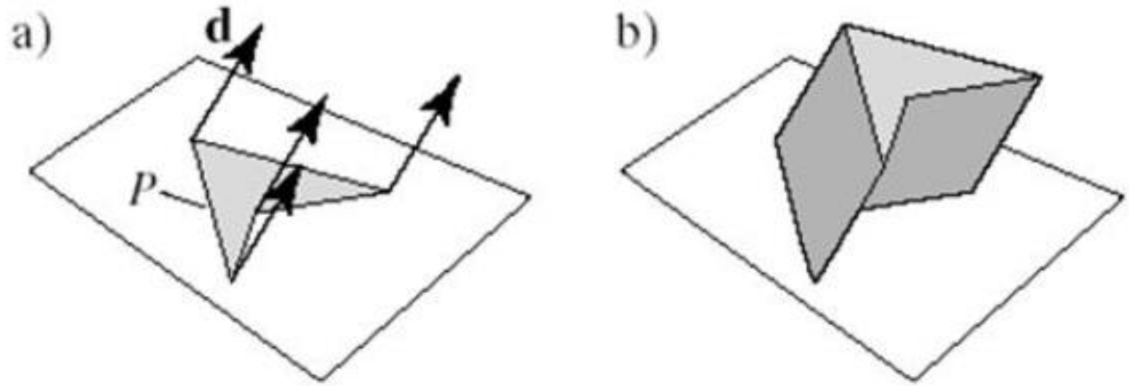
- 정의: a polyhedron is a connected mesh of simple planar polygons that encloses a finite amount of space. 아래 예들
- A very large number of solid objects of interest are polyhedra



- 오일러 공식 (Euler's formula)
- Simple polyhedron에 대해서 Number of Faces (F), edges (E), vertices (V)의 기본적인 관계 제공
- V+F-E=2
- 예: Cube, V=8, F=6, E=12

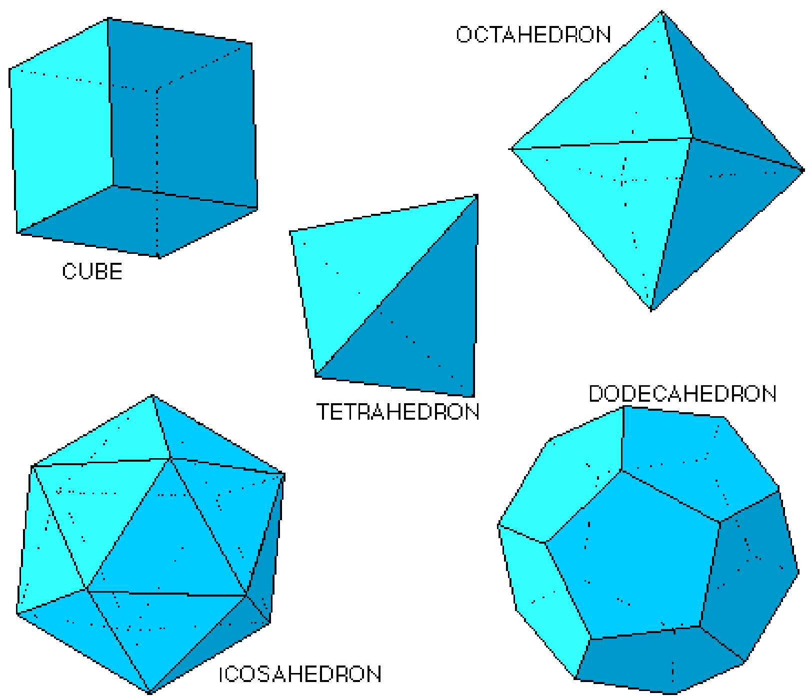


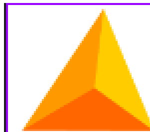
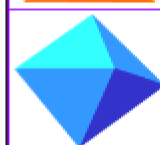
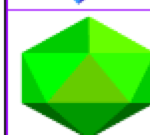
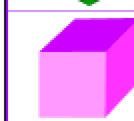
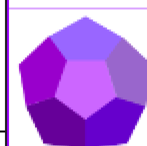
- A prism is a particular type of polyhedron that embodies symmetries (대칭)
- A prism is defined by sweeping a polygon along a straight line, turning a 2D polygon into a 3D polyhedron
- 예: a polygon P in part a) is swept along vector d to form the polyhedron as shown in part b)



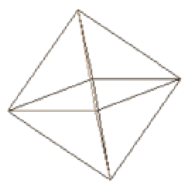
- **Regular polyhedron**

- If all of the faces of a polyhedron are identical and each is a regular polygon, the object is a regular polyhedron
- 이 중에 convex한 regular polyhedron을 platonic solids라고 별도로 부름. 아래의 5 platonic solids

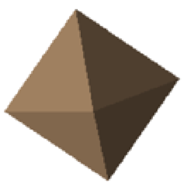


	Name	Faces	Edges	Vertices
	tetrahedron	4	6	4
	octahedron	8	12	6
	icosahedron	20	30	12
	cube	6	12	8
	dodecahedron	12	30	20

- **glutobject에 platonic solid가 있음**
- **예: glutWireOctahedron() 혹은 glutSolidOctahedron()**



(A) glutWireOctahedron();



(B) glutSolidOctahedron();

- **예: glutWireDodecahedron() 혹은 glutSolidDodcahedron()**

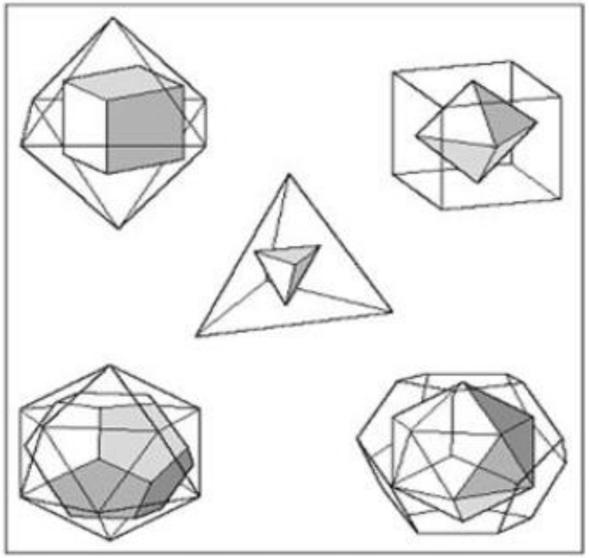
(A) glutWireDodecahedron();

(B) glutSolidDodecahedron();

- OpenGL에서의 glutobject 예들.

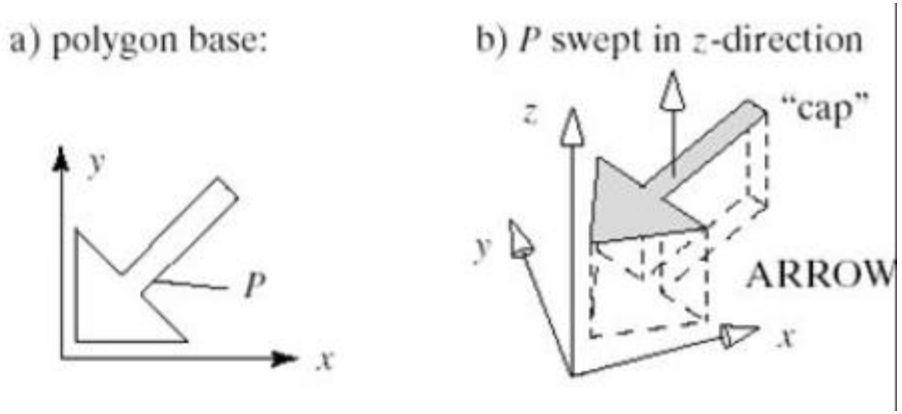
- Wireframe과 solid 모두 보여줌
- 키보드로 interaction 가능
- 키보드에서 ‘x’키 누름, ‘y’키 누름, ‘z’키 누름으로서 각 축에서 rotation
- 키보드에서 위아래 키 누름으로 다음 glutobject
- https://www.dropbox.com/s/61yeo5estnxo810/polyhedron.txt?dl=0

- Each of the Platonic solids P has a dual polyhedron D (쌍둥이 다변체)
- D의 vertices는 P의 faces의 중점
- D의 edges는 인접하는 P의 faces의 중점을 연결
- Duality
- Tetrahedron의 dual은 tetrahedron
- Cube와 octahedron은 서로 dual
- Icosahedron과 dodecahedron은 서로 dual

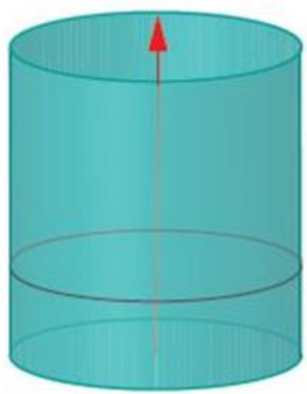


- **Extruded shapes (3D modeling)**

- A large class of shapes can be generated by **extruding or sweeping a 2D shape through shape**
- Prism을 xy 평면상에서 polygon으로 시작해서 z축으로 distance H 만큼 swept
- Prism의 모든 vertex 개수는 14개
- Prism은 flat faces이므로 각 face가 같은 normal vector를 가짐



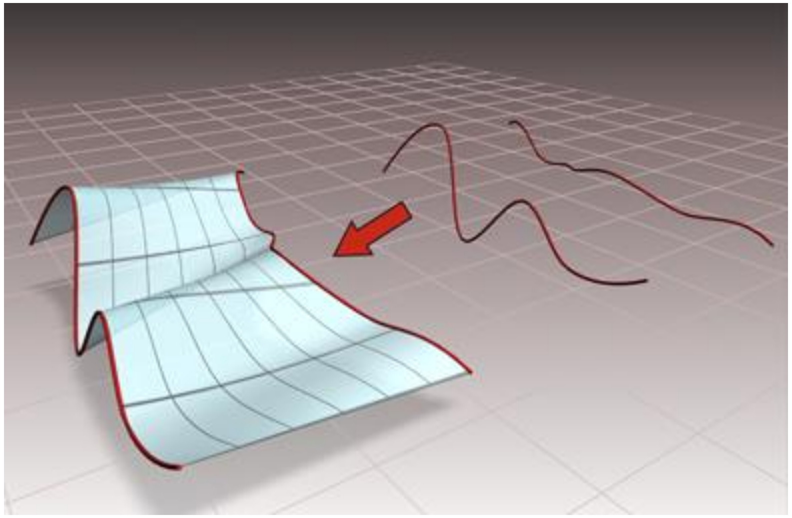
- 이는 curve에도 확장 가능하다
- Sweep Surfaces are surfaces (표면) that are generated from a section curve positioned along a path.
- 아래 cylinder를 보면 원의 중심이 어떤 궤적 (trajectory, 밑의 빨간색)를 따라서 위로 올라가면서 생기는 표면이라고 생각할 수 있다. 이와 같이 만들어진 표면을 swept surface라고 한다
- 아래 torus도 유사하게 원이 궤적을 따라서 돌면서 생기는 표면이다
- 앞의 cylinder와 같이 만일 trajectory가 직선 형태의 선분이라면 이를 extruded surface라고 한다



- **OpenGL을 이용한 torus sweeping surface 생성 예**
- **space 키 누르면 실행**
- **https://www.dropbox.com/s/mk7bng0gk594z88/sweeping_1.txt?dl=0**

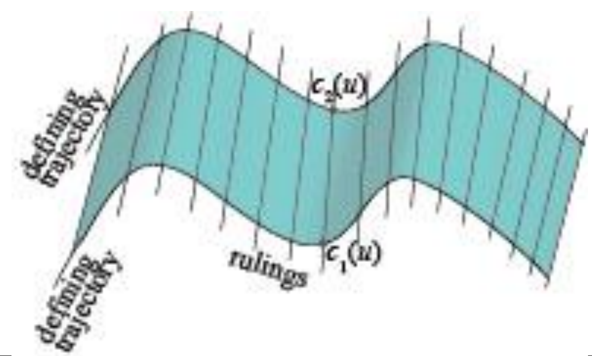
- **3D modeling: ruled surface**

- 3ds MAX에서 소개하는 2개의 curve를 기반으로 ruled surface로 만든 곡면
- Ruled surface 란? 먼저 양 끝의 곡선을 정의한 후 그 곡선들을 잇는 직선을 연결하여 평면 (곡면)을 만든 다

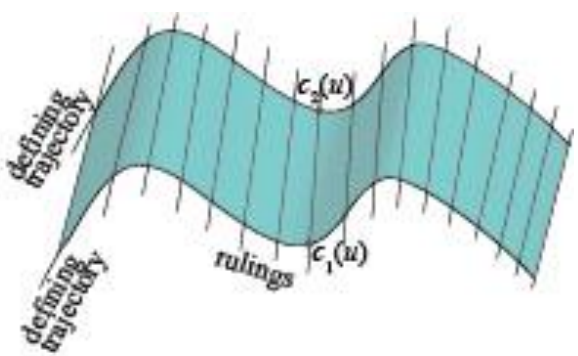


- **https://www.youtube.com/watch?v=BCHS8hasopw**

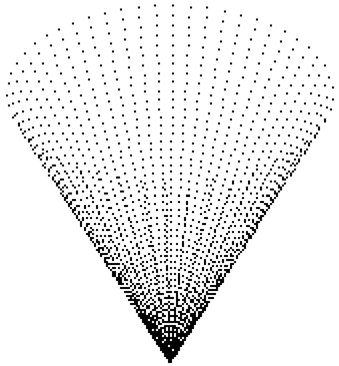
- Ruled surface: 아래 그림과 같이 생성된 두 개의 curve, $c_1(u), c_2(u)$를 직선으로 연결하여 만든 surface
- Ruled surface는 surface 위의 모든 점에 대하여, 그 점을 지나는 직선을 찾을 수 있음
- https://en.wikipedia.org/wiki/Ruled_surface
- Ruled surface는 다음과 같이 표현 가능 하다
- $s(u,v) = (1-v)c_1(u) + vc_2(u)$, 단, $0 \le v \le 1$, $u \in [a,b]$



- Ruled surface 개념을 이용하면 다양한 형태를 만들 수 있다
- 먼저 cone 모양을 생각해 보자
- Ruled surface 식
- $s(u,v) = (1-v)c_1(u) + vc_2(u)$, 단, $0 \le v \le 1$, $u \in [a,b]$
- 이 경우 한쪽 점이 고정, $c_1(u) = p$ 로 놓으면
- $s(u,v) = (1-v)p + vc_2(u)$, 단, $0 \le v \le 1$, $u \in [a,b]$



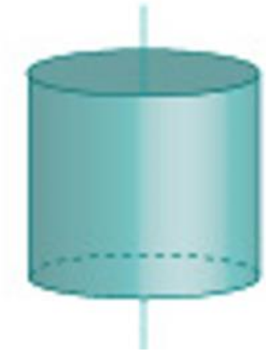
- $s(u,v) = (1-v)p + vc_2(u)$, 단, $0 \le v \le 1,\ u \in [a,b]$
- $c_2(u) = [cos(u), 3, sin(u)]$, y=3인 부분에 원 만듬
- $p = [0,0,0]$
- $s(u,v) = [v * cos(u), 3 * v, v * sin(u)]$
- 위쪽에 원 대신 다른 모양도 생성 가능

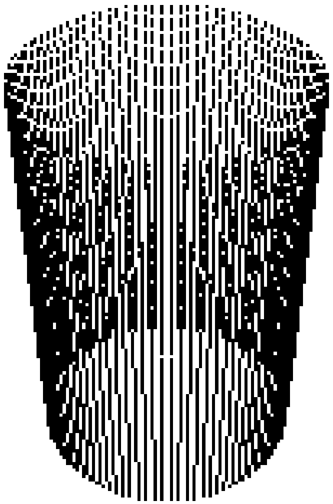


- https://www.dropbox.com/s/5uw2qblcr1uuqdl/ruled_0.txt?dl=0

- Ruled surface를 이용한 cylinder 모양 만들기
- $s(u, v) = (1 - v)c_1(u) + vc_2(u)$, 단, $0 \le v \le 1$, $u \in [a, b]$
- $c_2(u) = c_1(u) + d$

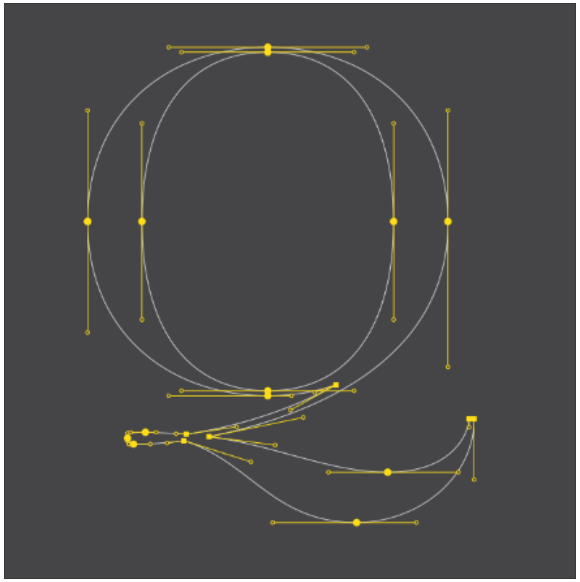
- 예)
- $c_1(u) = [cos(u), 0, sin(u)]$, y=0인 부분에 원 만듬
- $c_2(u) = [cos(u), 3, sin(u)]$, y=3인 부분에 원 만듬





- **Interactive한 곡선 및 곡면 설계**

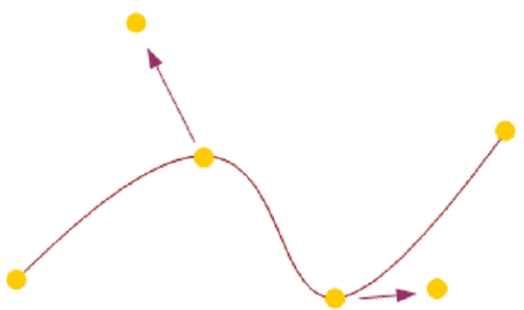
- **자동차, 비행기의 외형 , 로고 등을 설계할 때에도 곡선, 곡면이 필요하다**

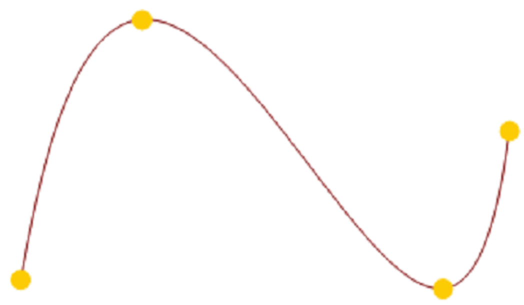


curve 를 이용한 Logo 디자인

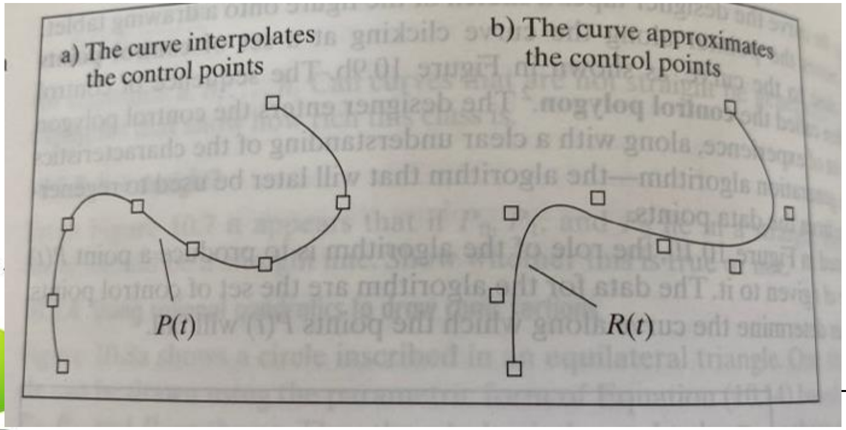
curve 를 이용한 차 디자인

- 곡선 (curve)
- 곡선의 모양을 제어 및 결정하는 특징적인 점들을 control points (제어점) 이라고 한다
- 사용자는 이러한 제어점을 추가, 삭제, 위치 변경들을 통해 곡선의 모습을 제어할 수 있다
- 예; https://www.youtube.com/watch?v=GC0OK8j7-B8


(a)

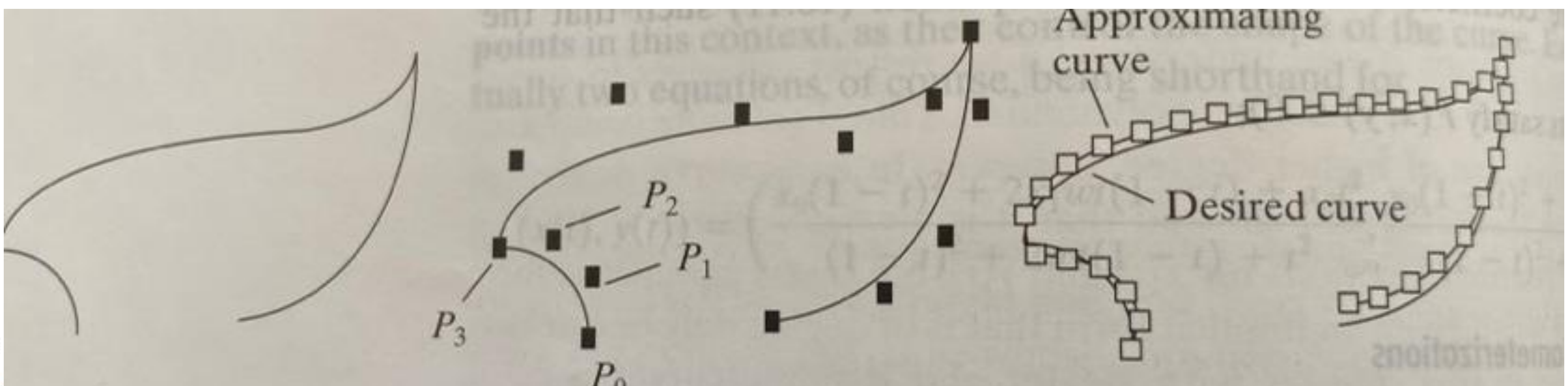

(b)

- Interpolating curve (보간 커브)
- Control point를 정확히 통과하는 곡선 (a)
- Approximating curve (근사 커브)
- Control point를 일반적으로 통과하지 않는다. 통과하더라도 처음과 마지막 제어점만 통과한다 (b)
- 제어점을 통과하지 않는 이유?
- (b) 처럼 조금 더 부드러운 (smooth)한 곡선 만듬



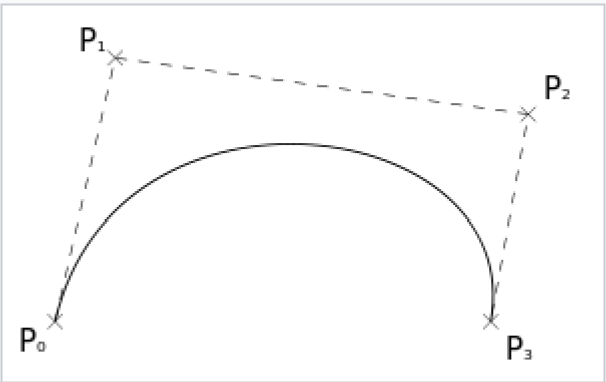
- Interactive design process
- 1. 최초의 control points 설정
- 2. 알고리즘을 이용하여 curve 생성
- 3. curve가 만족스러우면 stop
- 4. 그렇지 않으면 control points 조정
- 5. go to step 2

- **A curve design scenario**
- **왼쪽: desired curve**
- **가운데: control points**
- **오른쪽: 알고리즘과 approximating curve로 만들어진 curve 결과**

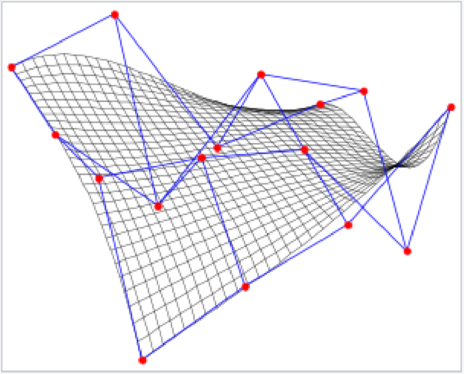


- $\mathrm{B}\acute{e}zier$ **curve (베지에 곡선)**

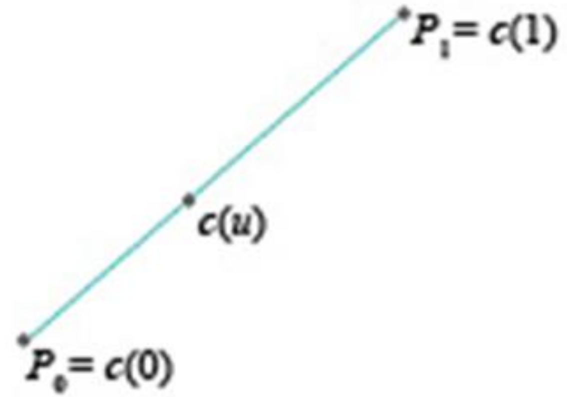
- $\textit{Bézier}$ curve
- 1960년대 르노 자동차 공장의 엔지니어였던 $\textit{Bézier}$ 가 제안한 곡선
- 원래 자동차 디자인에 사용하기 위한 부드러운 곡선을 만들기 위해 개발
- 이 $\textit{Bézier}$ curve 를 확장하여 $\textit{Bézier}$ surface 도 생성
- $\textit{Bézier}$ curve를 만드는 알고리즘을 de Casteljau 알고리즘이라고도 함



Cubic Bézier curve with four control points


Sample Bézier surface; red - control points, blue - control grid, black - surface approximation

- **1. Linear *Bézier* curve (1차 *Bézier* curve, 선형 보간)**
- 두 개의 control point, P0, P1만 있다고 하자
- P0와 P1을 연결하는 선분 parametric form으로 표현하면
- $c(u) = (1-u)P_0 + uP_1$, 단, $0 \le u \le 1$
- Note: 가중치 합=1



- Linear $\mathrm{B}\acute{e}zier$ curve
- https://en.wikipedia.org/wiki/B%C3%A9zier_curve

2. Quadratic $Bézier$Curves (2차 $Bézier$ curve)

세 개의 control points를 사용한다, P0, P1, P2

1. P0과 P1 선분 사이의 임의의 한 점, $a(u) = (1-u)P_0 + uP_1$, $0 \le u \le 1$

2. P1과 P2 선분 사이의 임의의 한 점, $b(u) = (1-u)P_1 + uP_2$, , $0 \le u \le 1$

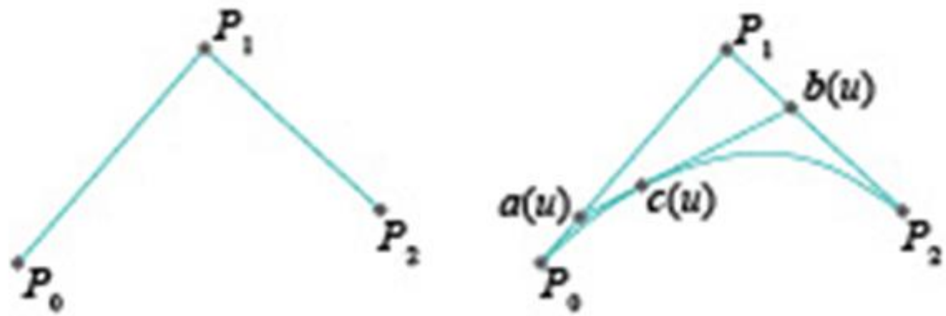
3. a(u)와 b(u)를 선형 보간, 그 선분 위의 한 점, $c(u) = (1-u)a(u) + ub(u)$, , $0 \le u \le 1$

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

앞의 식과 같이 계수들의 합은 1이다

Observation: u 값을 0에서 부터 증가시키면서 1까지 그리면 곡선이 나옴

u=0.3이면 어느 위치?



인천대학교

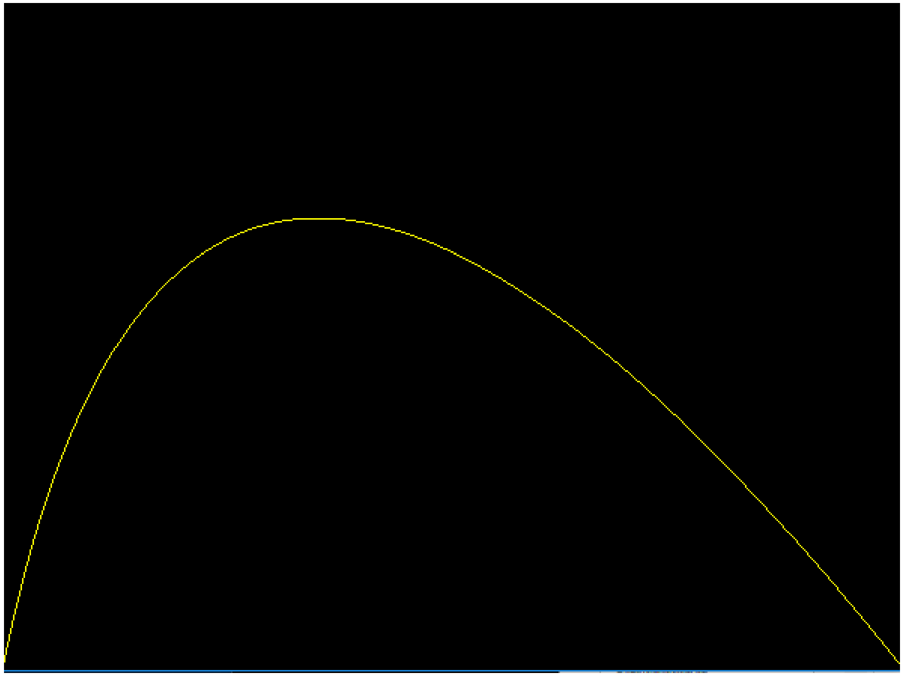
- **Quadratic $B\acute{e}zier$ Curves (2차 $B\acute{e}zier$ curve)**
- **https://en.wikipedia.org/wiki/B%C3%A9zier_curve**

- Quadratic $B\acute{e}zier$Curves를 OpenGL에서 u값을 조금씩 증가시키면서 곡선을 그려보았다
- $\mathbf{c}(u) = (1-u)^2 P_0 + 2u(1-u)P_1 + u^2 P_2,\ 0 \le u \le 1$
- 왼쪽 오른쪽 키를 이용하여 u값을 증가시키거나 감소시키면서 2차 베지에 곡선을 그려보자

- **https://www.dropbox.com/s/xjebn6lr40lozt5/bezier_2.txt?dl=0**

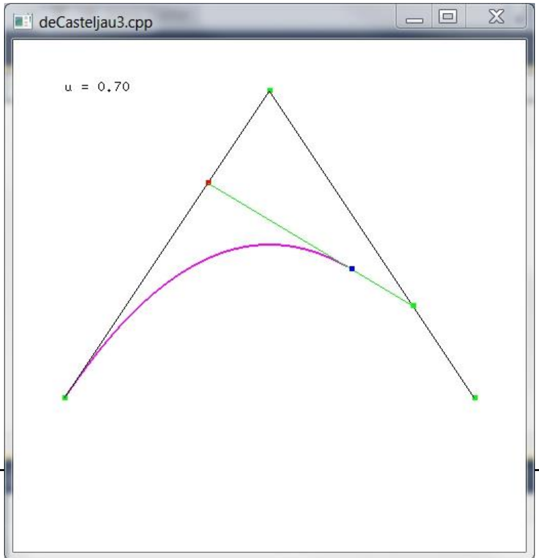
- 예) 세 개의 control point, (0, -1), (1, 2), (5, -1)이 주어져 있을 때 Quadratic $\mathrm{B}\acute{e}zier$Curve (2차)를 앞의 수식을 이용하여 구해 보자

- GL_LINE_STRIP을 이용하여 세 개의 control point, (0, -1), (1, 2), (5, -1)이 주어져 있을 때 Quadratic $\mathbf{B}\acute{e}zier$Curve (2차)를 그려보았다



- https://www.dropbox.com/s/8o1k7u7h036da3p/bezier_1.txt?dl=0

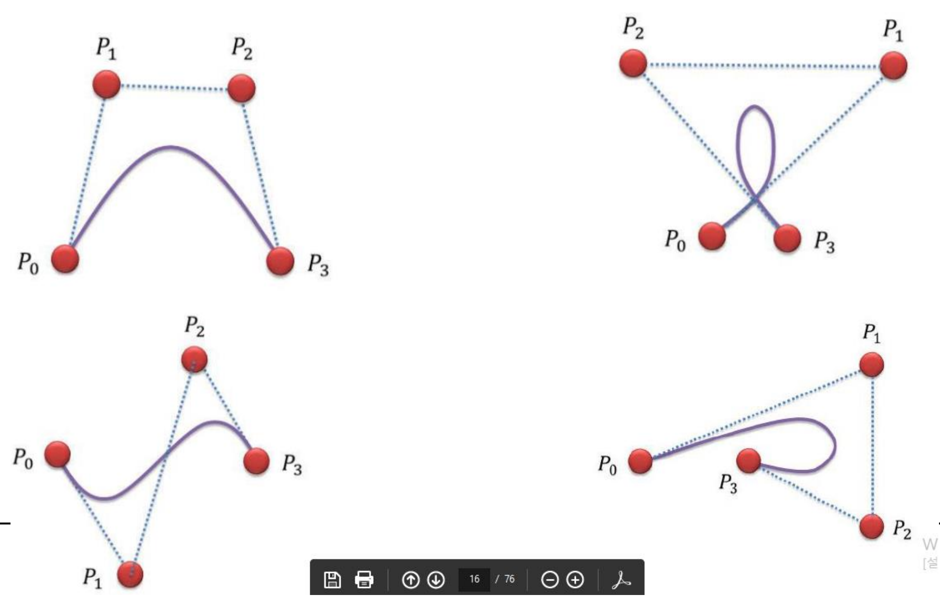
- 앞의 예제에서 보았듯이 세 개의 control point를 이용한 Quadratic $\mathit{Bézier}$Curve는 smooth한 곡선이 된다
- Observation
- 1. 단, 세 개의 control point중 처음과 마지막 control point만 지난다 => Quadratic $\mathit{Bézier}$Curve 는 approximating curve
- 2. 세 개의 control point에 의해 형성되는 convex hull 내부에만 $\mathit{Bézier}$Curve 가 존재 한다



- **Cubic $\text{B}\acute{e}zier$ Curves**

- Cubic Bézier Curves **(3차 Bézier curve)**
- 가장 흔하게 사용되는 Bézier Curves
- Quadratic Bézier Curve 보다 하나의 control point를
- 추가한4개의 control points를 사용한 Bézier Curve

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



인천대학교

University of Incheon

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

1. 선분 P0과 P1사이의 임의의 한 점, $a(u) = (1-u)P_0 + uP_1$

2. 선분 P1과 P2사이의 임의의 한 점, $b(u) = (1-u)P_1 + uP_2$

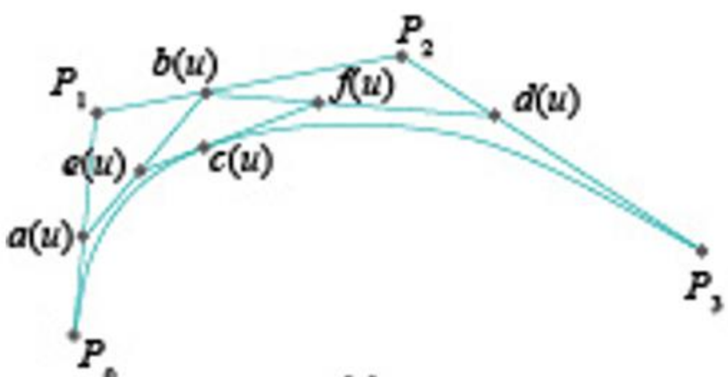
3. 선분 P2와 P3사이의 임의의 한 점, $\mathrm{d}(u) = (1-u)P_2 + uP_3$

4. a(u)와 b(u)사이의 임의의 한 점, $e(u) = (1-u)a(u) + ub(u)$

5. b(u)와 d(u)사이의 임의의 한 점, $\mathrm{f}(u) = (1-u)b(u) + ud(u)$

6. e(u)와 f(u)사이의 임의의 한 점, $c(u) = (1-)e(u) + uf(u)$

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



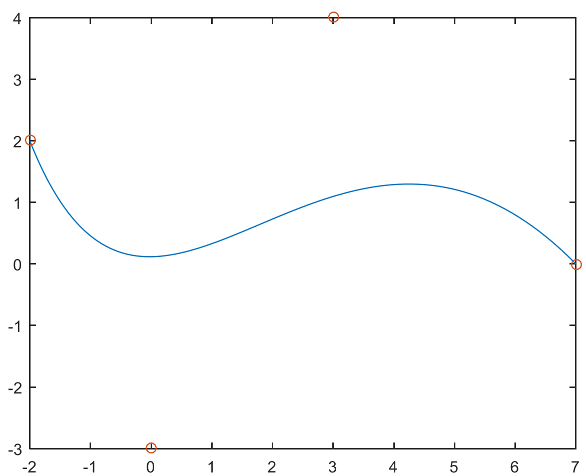
(a)

- Cubic $\mathbf{B}\acute{e}zier$ Curves
- https://en.wikipedia.org/wiki/B%C3%A9zier_curve

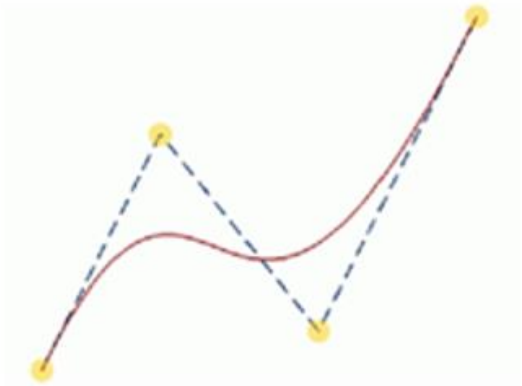
- 네 개의 control point, (-2, 2), (0, -3), (3, 4), (7,0)을 이용한 cubic $Bézier$Curve

$\Rightarrow$ $$\mathbf{c}(u) = (1-u)^3P_0 + 3(1-u)^2uP_1 + 3(1-u)u^2P_2 + u^3P_3$$

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

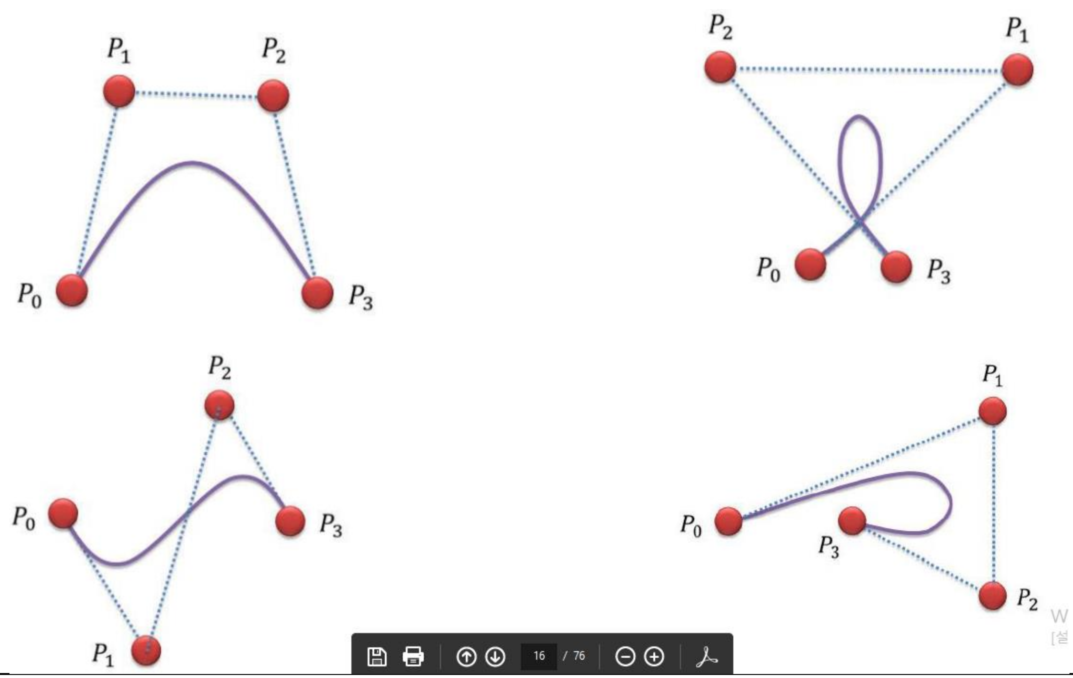


- Observation
- 1. 네 개의 control point중 처음과 마지막 control point만 지난다
- Cubic $Bézier$Curve 도 approximating curve
- 2. 네 개의 control point에 의해 형성되는 convex hull 내부에만 $Bézier$Curve 가 존재 한다
- 3. 더 높은 차수의 $Bézier$Curve 는 계산 상의 복잡함으로 인해서 복잡한 $Bézier$Curve 도 3차로 나누어서 그림



- 처음 제어점과 마지막 제어점은 지나지만 중간 제어점들은 가까이 다가가기만 한다

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



■ $B\acute{e}zier$Curve의 일반화

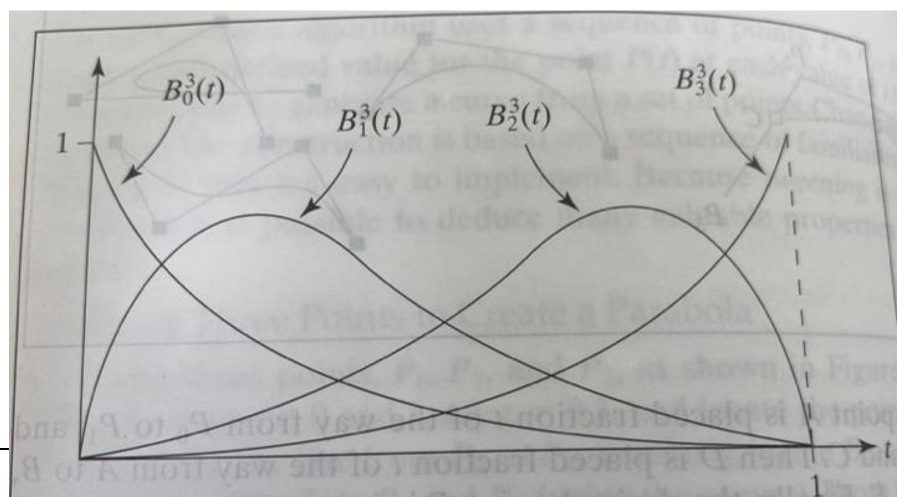
- 일반적인 $\boldsymbol{B\acute{e}zier}$Curve 표현식
- n+1개의 control points, $\boldsymbol{P_0, P_1, \ldots, P_n}$ 에대하여
- $\boldsymbol{c(u) = \sum_{i=0}^{n} \binom{n}{i} (1-u)^{n-i} u^i \, P_i}$, 단, $\boldsymbol{0 \le u \le 1}$
- 여기서 $\boldsymbol{\binom{n}{i} = \frac{n!}{i!(n-i)!}}$ 을 binomial coefficient 함수라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

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, $0 \le u \le 1$
- $\sum_{i=0}^{n} B_i^n(u) = 1$

인천대학교

- $B_i^n(t) = \binom{n}{i}(1-t)^{n-i}t^i$, , 단, $0 \le t \le 1$
- **n=3인 경우, 아래 그림**
- $B_i^3(t) = \binom{3}{i}(1-t)^{3-i}t^i$, , 단, $0 \le t \le 1$
- $B_0^3(t) = (1-t)^3, B_1^3(t) = 3(1-t)^2t$

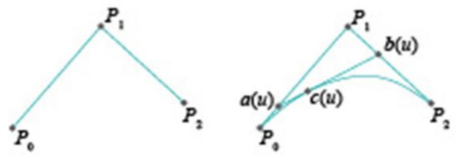


- **$\boldsymbol{Bézier}$Curve의 특징**
- **If c(u) is the $\boldsymbol{Bézier}$Curve approximating the sequence of n+1 control points, P0, P1, …, Pn, then the following hold**

(a) $\boldsymbol{Bézier}$Curve, c(u) 함수 다항식의 차수는 n차 이다

(예: quadratic => c(u)는 2차 다항식)

$$\mathbf{c}(\boldsymbol{u}) = (\mathbf{1}-\boldsymbol{u})^{2}\boldsymbol{P}_{0} + \mathbf{2}\boldsymbol{u}(\mathbf{1}-\boldsymbol{u})\boldsymbol{P}_{1} + \boldsymbol{u}^{2}\boldsymbol{P}_{2},\ \mathbf{0} \le \boldsymbol{u} \le \mathbf{1}$$



(b) c lies inside the convex hull of P0, P1, … Pn

(c) c는 처음과 마지막 control point를 지난다. 하지만 중간 control point는 지날 수도 있고 안 지날 수도 있다

인천대학교

- $Bézier$Curve 의 단점은 무엇일까?

1. control point가 늘어나면 차수도 늘어난다

식도 더 복잡해지고 그로 인해 수치 오차가 생기기 쉽다. 또한 계산량도 늘어난다. L+1개의 control points를 사용하면 L차 다항식이된다.

2. 모든 control point의 정보를 사용한다

만일 하나의 control point가 움직이면 curve 모양 전체가 바뀐다

3. Local control의 문제

$Bézier$Curve은 curve 모양의 충분한 local control을 제공하지 못한다

University of Incheon
인천대학교

- **Problem of local control**
- **아래 예: 5개의 control points를 사용한 $\boldsymbol{B\acute{e}zier}$Curve**

- 다음은 Bezier curve를 OpenGL로 구현한 것이다
- 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