



COMPUTER GRAPHICS
School of Computer Engineering

Suranaree University
of Technology

Lecture 9 3-Dimensional Object Representations (Part II)

Paramate Horkaew

School of Computer Engineering, Institute of Engineering
Suranaree University of Technology



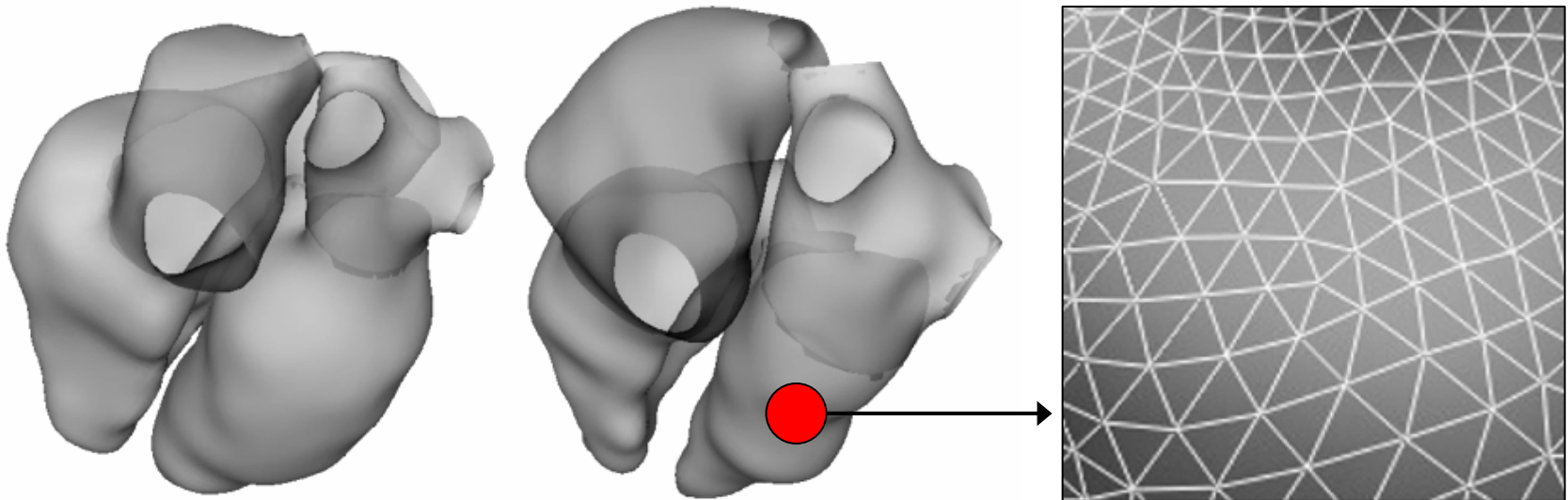
Lecture Outline

- 3-D Object Representations
 - Polygonal Surfaces
 - Planes in 3D Space
 - Quadric Surfaces and Blobs
- Spline Representations
 - Interpolation and Approximation Spline
 - Continuity Conditions
 - Cubic Spline Interpolation
 - Bezier Curves and Surfaces
 - B-Spline Curves and Surfaces
 - Other Splines and Their Conversions
- Introduction to OpenGL
 - OpenGL Programming using C/C++

Polygon Surfaces

การแสดงวัตถุด้วย Polygon ใช้กันมากที่สุดในโปรแกรมประยุกต์ด้านกราฟิก เนื่องจากสามารถคำนวณได้เร็ว จัดอยู่ในการแสดงผลประเภท B-Rep ซึ่งอธิบายวัตถุ ด้วยชุด ของ Polygons หลายๆ ชิ้นมาต่อกัน ล้อมรอบวัตถุไว้ โดยแบ่งบริเวณใน 3D ออกเป็น 2 บริเวณ ได้แก่ ด้านใน และ ด้านนอก วัตถุ

รูปนี้ แสดงตัวอย่างของการแสดง วัตถุด้วย Polygons ที่เป็นรูปสามเหลี่ยมต่อกัน (เรียกว่า Tile หรือ Tessellate เหมือนการปูกระเบื้อง หรือ Mosaic)





Polygon Tables

โดยทั่วไป เราจะจัดเก็บ Polygons ไว้ในโครงสร้างข้อมูลชนิดตาราง (Array) ประกอบด้วย

- **Geometric Tables**

ประกอบด้วยข้อมูลทางเรขาคณิตของ Polygons ได้แก่ พิกัดจุดยอด (vertices) ของ Polygon แต่ละชิ้น

- **Topological Tables**

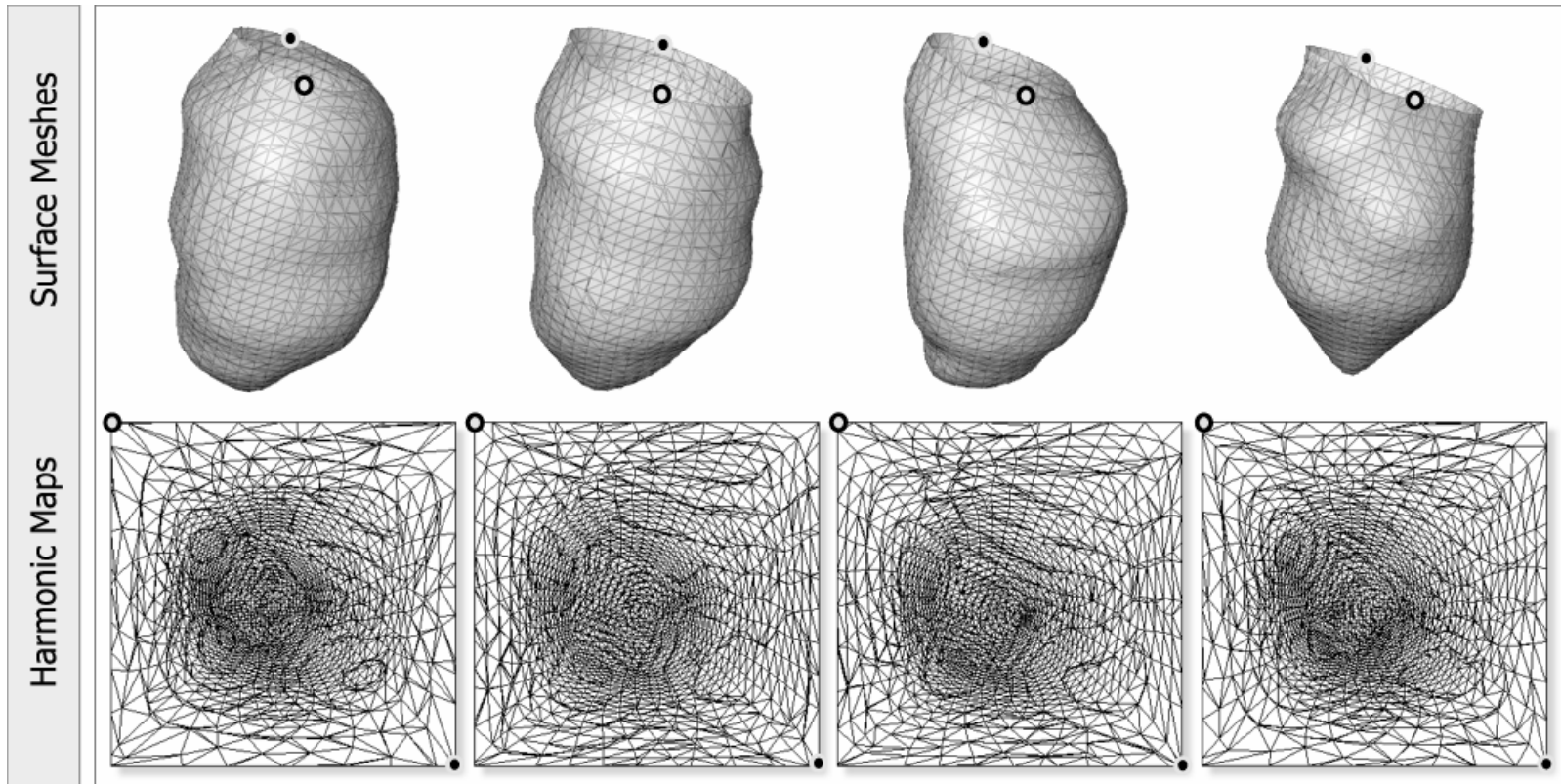
ประกอบด้วยข้อมูลการเรียง Polygon ได้แก่ ลำดับของขอบ Polygon ซึ่งกำหนด orientation (edge list) และ การเชื่อมต่อระหว่าง Polygons (face list)

- **Attribute Tables**

ประกอบด้วยข้อมูลทางกายภาพเพื่อแสดงผล เช่น ความโปร่งใส ดัชนีสะท้อน สี

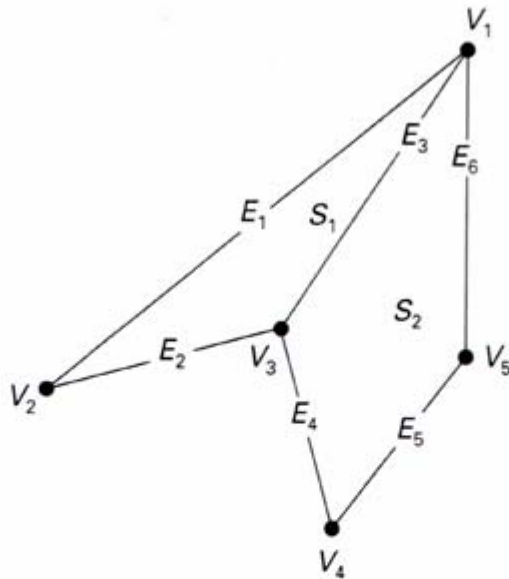
An Example

รูปด้านล่างแสดง ตัวอย่าง Geometry ของพื้นผิว หัวใจห้องล่างซ้าย (LV) ประกอบด้วย สามเหลี่ยมเล็กๆ เชื่อมต่อกัน ซึ่งนิยามด้วย Topology (หรือ graphs ด้านล่าง) คณิตศาสตร์เบื้องต้นของทฤษฎีกราฟ สามารถนำมาใช้ใน Graphics ได้



Polygon Tables

สำหรับตัวอย่าง โครงสร้างข้อมูลทาง Geometry และ Topology แสดงดังรูป



VERTEX TABLE

V_1 : x_1, y_1, z_1
 V_2 : x_2, y_2, z_2
 V_3 : x_3, y_3, z_3
 V_4 : x_4, y_4, z_4
 V_5 : x_5, y_5, z_5

EDGE TABLE

E_1 : V_1, V_2
 E_2 : V_2, V_3
 E_3 : V_3, V_1
 E_4 : V_3, V_4
 E_5 : V_4, V_5
 E_6 : V_5, V_1

POLYGON-SURFACE
TABLE

S_1 : E_1, E_2, E_3
 S_2 : E_3, E_4, E_5, E_6

ตาราง vertex เก็บพิกัดของจุดยอด ของ polygon นอกจากนี้ เราอาจจะ จัดเก็บข้อมูลอื่นๆ เช่น normal vector ที่ตั้งฉากกับพื้นผิวที่จุดนั้นซึ่ง นำมาใช้ในการคำนวณความสว่างของพื้นผิว (กล่าวถึงในหัวข้อถัดไป)

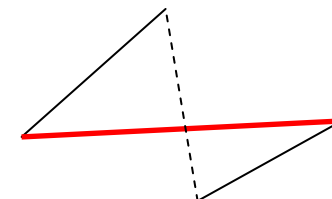
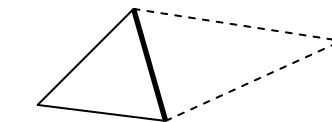
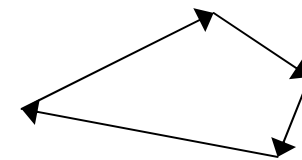
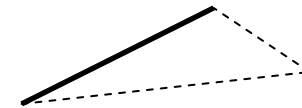
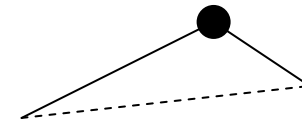
ตาราง edge จะเก็บ pointer ที่ชี้ไปยังตาราง vertices เพื่อกำหนดจุดปลายสองจุดของเส้นตรง (edge)

ตาราง face จะเก็บ pointer ที่ชี้ไปยัง ตาราง edge เพื่อกำหนดเส้นขอบที่นิยาม polygon

Surface Integrity

จากโครงสร้างข้อมูลที่กำหนด เราสามารถตรวจสอบ ได้ว่า Surface ที่สร้างขึ้น มีความถูกต้อง และ ครบถ้วนสมบูรณ์ เพียงใด โดยพิจารณาจากเงื่อนไขต่อไปนี้

1. แต่ละ Vertex จะต้องปรากฏเป็นส่วนประกอบของ Edge อย่างน้อย 2 เส้น
2. Edge แต่ละเส้น ต้องเป็นส่วนประกอบของ Polygon อย่างน้อย 1 ชิ้น
3. Polygon ทุกชิ้นต้องครบรอบเป็นวงปิด close
4. Polygon แต่ละชิ้น ต้องมี Edge ที่ใช้ ร่วมกับ Polygon อื่นๆ อย่างน้อย 1 เส้น
5. ไม่มี edge สองเส้นใดๆ ตัดกัน



Plane Equations

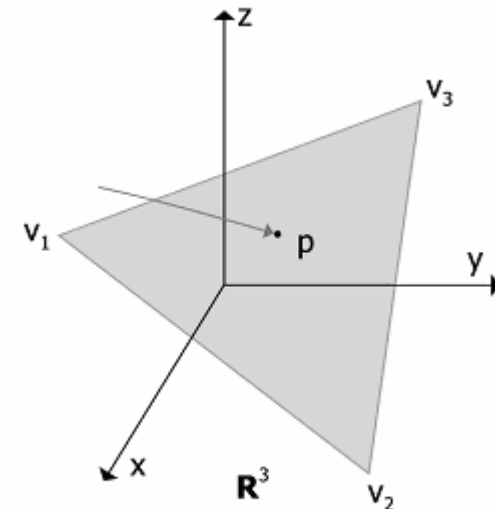
การแสดงผลแบบกราฟิก ประกอบด้วยการคำนวณหลายขั้นตอน ได้แก่

- 1) การ แปลงระหว่าง model, world, viewing และ device coordinates
- 2) การบังชี้ บริเวณที่มองเห็น/ไม่เห็น เช่น clipping, hidden surface removal
- 3) การ rendering กำหนด สีและความสว่างของจุดภาพ ขึ้นกับคุณสมบัติของวัตถุ

สำหรับ กระบวนการข้างต้น เราจำเป็นต้องทราบ ข้อมูลเกี่ยวกับ orientation ของ **องค์ประกอบพื้นฐานของพื้นผิว** ในที่นี้ ได้แก่ระนาบของชิ้นส่วน Polygon ดังรูป ซึ่งเขียนเป็นสมการ ได้ดังนี้

$$Ax + By + Cz + D = 0$$

โดยที่ (x, y, z) คือพิกัดบนระนาบ และ A, B, C, D คือค่าคงที่ของระนาบ



Geometrical Realisation $\pi_v(|K|)$



Solving a Plane Equation

ถึงแม้ว่าสมการระนาบจะมี 4 ตัวแปร แต่ว่า DOF ของสมการมีเพียง 3 ระดับ ดังนั้น การหาสัมประสิทธิ์ A, B, C, D ต้องทราบจุดยอดอย่างน้อย 3 จุด ที่ไม่อยู่ในแนวเส้นตรงเดียวกัน (**noncollinear**)

จัดรูปสมการใหม่ได้ $(A/D)x_k + (B/D)y_k + (C/D)z_k = -1 \quad k = 1, 2, 3$

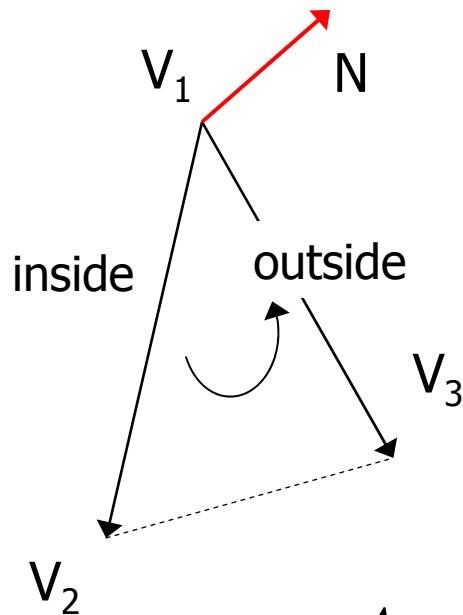
ซึ่งมีคำตอบอยู่ในรูปของ determinant ดังนี้ (**ใช้ Cramer's Rule พิสูจน์**)

$$A = \begin{vmatrix} 1 & y_1 & z_1 \\ 1 & y_2 & z_2 \\ 1 & y_3 & z_3 \end{vmatrix} \quad B = \begin{vmatrix} x_1 & 1 & z_1 \\ x_2 & 1 & z_2 \\ x_3 & 1 & z_3 \end{vmatrix} \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} \quad D = - \begin{vmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{vmatrix}$$

ดังนั้นระนาบจึงนิยามได้ด้วย **เวกเตอร์** ซึ่งตั้งฉากกับระนาบ (normal vector) กำหนดด้วยสัมประสิทธิ์ A, B, C โดยที่ $N = [A \ B \ C]^T$

Interior and Exterior Regions

เนื่องจากการแสดงผล Polygon เป็นแบบ B-Rep (Boundary Representation) ซึ่งแบ่งแยกบริเวณ **ภายใน** (ส่วนที่มองไม่เห็น) และ **ภายนอก** (ส่วนที่มองเห็น) วัตถุ ถ้าระบุดยอดของระนาบในทิศทาง ทวนเข็มนาฬิกา (CCW) เมื่อมองจากภายนอกวัตถุ แล้ว ทิศทางของ normal vector จะชี้จาก **ภายในออกสู่ภายนอก** วัตถุ ซึ่งคำนวณได้จากการหา cross product



ซึ่งมีสมการดังต่อไปนี้ $\mathbf{N} = (\mathbf{V}_2 - \mathbf{V}_1) \times (\mathbf{V}_3 - \mathbf{V}_1)$

ถ้าทราบว่าจุด **P** อยู่บนระนาบ เราสามารถหาค่าคงที่ D ได้

$$D = -\mathbf{N} \cdot \mathbf{P} \quad \equiv \text{ระยะทางจากจุด P ไปยังระนาบ}$$

เมื่อทราบ $[A, B, C] = \mathbf{N}$ และ D แล้วเราสามารถหาว่าจุด $P = (x, y, z)$ อยู่ภายในหรือภายนอกระนาบได้จาก

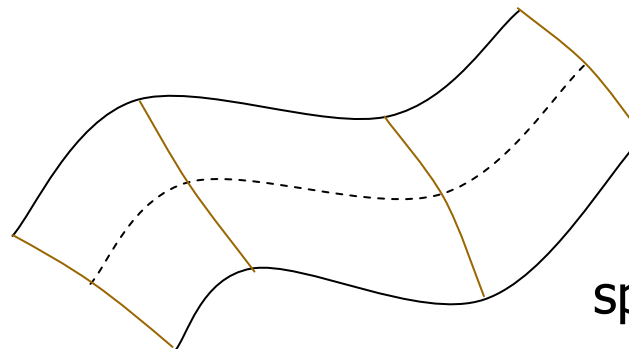
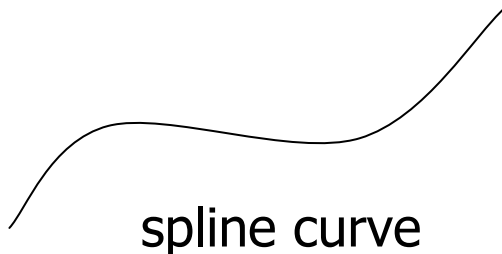
$$Ax + By + Cz + D \begin{cases} < 0 & (x, y, z) \text{ is inside} \\ > 0 & (x, y, z) \text{ is outside} \end{cases}$$

Spline Representations

Spline นิยามมาจากคำศัพท์ในวิชา เขียนแบบ (Engineering Drawing) ได้แก่ แถบโค้งยืดหยุ่น ซึ่งใช้สำหรับ วาดเส้นโค้งผ่านจุดที่กำหนด

สำหรับในทางคณิตศาสตร์ Spline นิยามด้วย ฟังก์ชันพหุนามกำลังสาม ที่แบ่งออกเป็นช่วง (piecewise cubic polynomial function) ดังนั้น

- spline curve คือ เส้นโค้งที่ได้จากการรวมเส้นโค้งพหุนามหลายๆ เส้นด้วยกัน โดยที่ต้องสอดคล้องกับเงื่อนไข ความต่อเนื่อง (continuity conditions) ระหว่างเส้น
- spline surface คือ พื้นผิว ที่ได้จากการรวม spline curve สองชุดที่ตั้งฉากกัน



Interpolation and Approximation

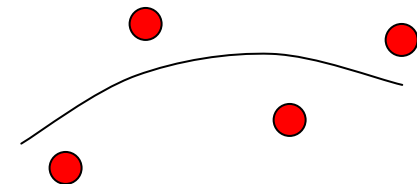
การนิยามเส้นโค้งแบบ spline ทำได้โดยระบุ กลุ่มของพิกัด เรียกว่า control points ซึ่งกำหนดรูปทรงของ เส้นโค้ง โดยที่การลากเส้นโค้งตาม control points สามารถทำได้ 2 วิธี

- ลากโดยให้เส้นโค้งผ่าน ตำแหน่ง ของจุดที่กำหนดทุกๆ จุด (Interpolation) เหมาะสำหรับ การลากเส้นตามแบบ และ การกำหนดเส้นทางของการเคลื่อนไหวของวัตถุ (animation)
- ลากโดยให้เส้นโค้งผ่าน เส้นทาง ของจุดที่กำหนด โดยไม่จำเป็นต้องผ่านจุดนั้นๆ ก็ได้ (Approximation) เหมาะสำหรับ การออกแบบโครงสร้างของ พื้นผิวของวัตถุ

Interpolation



Approximation





Parametric Continuity

เนื่องจากเส้นโค้งใดๆ เกิดจากการเชื่อมต่อกันของ ฟังก์ชันพหุนามของเส้นโค้ง
ดังนั้น เพื่อให้รอยต่อมีความต่อเนื่อง ต้องกำหนดเงื่อนไขความต่อเนื่อง ดังนี้

กำหนดให้เส้นโค้ง spline กำหนดด้วยพิกัด Cartesian ซึ่งแต่ละองค์ประกอบ
เป็นฟังก์ชัน 1 ตัวแปร (u)

$$\mathbf{P} = [x \quad y \quad z]^T \rightarrow x = x(u), \quad y = y(u), \quad z = z(u)$$

ดังนั้น เงื่อนไขความต่อเนื่องนิยามโดย พิจารณาให้ค่าอนุพันธ์ขององค์ประกอบ
 x, y, z เทียบกับตัวแปร u ที่**ขอบรอยต่อรวม** แบ่งได้เป็น 3 ชนิด

- Zero-order (C^0) อนุพันธ์อันดับ 0 **เท่ากัน** (ค่า x, y, z ที่จุดรอยต่อเท่ากัน)
- First-order (C^1) อนุพันธ์อันดับ 1 และ 0 **เท่ากัน** (เส้นความชัน tangent line หรือ x', y', z' ที่รอยต่อเท่ากัน) เส้นโค้งย่อยมีรูปร่างต่างกันได้
- Second-order (C^2) อนุพันธ์อันดับ 2 และ 1 **เท่ากัน** (x'', y'', z'') ที่รอยต่อ เส้นโค้งย่อย มีรูปร่างคล้ายกัน ใช้สำหรับการกำหนดเส้นทางการเคลื่อนไหว

Geometric Continuity

กำหนดให้ $\mathbf{P} = [x \quad y \quad z]^T \rightarrow x = x(u), \quad y = y(u), \quad z = z(u)$

ดังนั้นอนุพันธ์อันดับ 1 และ 2 $\mathbf{P}' = [x' \quad y' \quad z']^T \quad \mathbf{P}'' = [x'' \quad y'' \quad z'']^T$

Geometric Continuity นิยามให้ **ทิศทาง** ของอนุพันธ์มีค่าเท่ากัน ที่รอยต่อ โดยที่ขนาด**ไม่จำเป็น** ต้องเท่ากันก็ได้ (แตกต่างจาก Parametric Continuity)

- Zero-order (G^0) ทิศทางอนุพันธ์อันดับ 0 เท่ากัน (ค่า x, y, z ที่จุดรอยต่อ เท่ากัน) เหมือนกับ Parametric Continuity
- First-order (G^1) อนุพันธ์อันดับ 1 และ 0 มีทิศทางเดียวกัน (เส้นความชัน tangent line หรือ x', y', z' ที่รอยต่อขนานกัน)
- Second-order (G^2) อนุพันธ์อันดับ 2 และ 1 มีทิศทางเดียวกัน (x'', y'', z'')

เส้นโค้ง ที่กำหนดด้วยเงื่อนไขความต่อเนื่องแบบ Parametric และ Geometric จะมีรูปร่างต่างกันเพียงเล็กน้อยเท่านั้น

Spline Specifications

เราสามารถ สร้างเส้นโค้ง spline ได้ 3 วิธี

1. กำหนด เงื่อนไข ที่จุดปลายของเส้นโค้ง (Boundary Conditions)
2. กำหนด matrix ที่บ่งชี้คุณสมบัติของเส้นโค้ง
3. กำหนด ชุดของฟังก์ชันผสม (Blending Functions หรือ Basis Functions) ซึ่งระบุชุดเงื่อนไขทาง Geometry เพื่อใช้ในการคำนวณจุดพิกัด (x, y, z) สำหรับ (u) ใดๆ บนเส้นโค้ง

วิธีที่ 1 สำหรับเส้นโค้งพหุนาม $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 \leq u \leq 1$

ถ้ากำหนด $x(0)$, $x(1)$, $x'(0)$ และ $x'(1)$ เราสามารถหาค่าคงที่ a , b , c , d ได้

โดยการแก้ระบบสมการ 4 ตัวแปร 4 สมการ (สำหรับ y , z คิดเหมือนกัน)

$$x(0) = d_x$$

$$x(1) = a_x + b_x + c_x + d_x$$

$$x'(0) = c_x$$

$$x'(1) = 3a_x + 2b_x + c_x$$

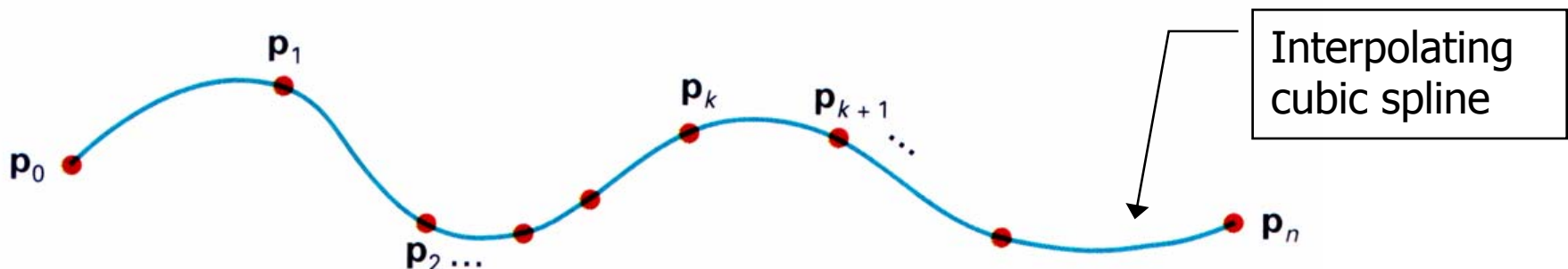
Cubic Spline Interpolation

Cubic Spline เหมาะสำหรับใช้ในการกำหนดเส้นทางการเคลื่อนไหวของวัตถุ หรือ กำหนดโครงร่างของวัตถุที่ได้จากการเก็บภาพภาพด้วยคอมพิวเตอร์ (เช่น laser digitization)

ฟังก์ชันพหุนามกำลัง 3 ใช้งานประเภทเหล่านี้อย่างแพร่หลายเนื่องจาก มีความยืดหยุ่น เทียบกับ ความเร็วในการคำนวณ ที่เหมาะสม ซึ่งมีขั้นตอนดังต่อไปนี้

ขั้นที่ 1 กำหนดพิกัดจุด control points จำนวน $(n + 1)$ จุด

$$p_k = (x_k, y_k, z_k), \quad k = 0, 1, 2, \dots, n$$



Solving Spline Coefficient

ขั้นที่ 2 สำหรับเส้นโค้งแต่ละช่วง (p_k, p_{k+1}) เราจำแก้สมการเพื่อหาสัมประสิทธิ์ a, b, c, d ที่อธิบายเส้นโค้งในช่วงนั้นๆ

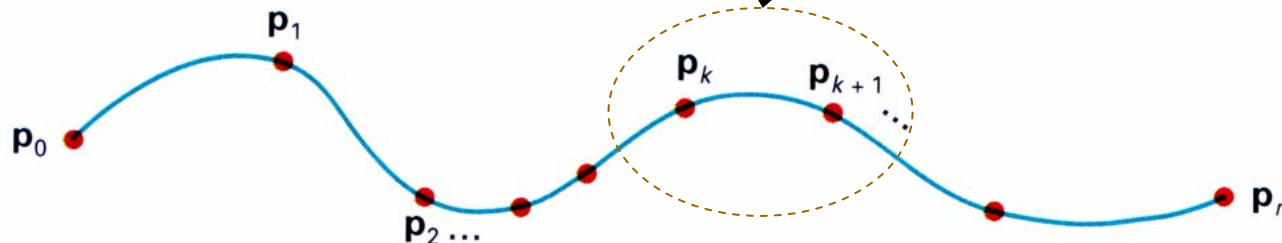
$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$

$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \quad 0 \leq u \leq 1$$

$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

สำหรับแต่ละสมการเราต้องการเงื่อนไขที่
รอยต่อ 4 เงื่อนไข ซึ่งจะได้อธิบายต่อไป

Solving for each curve
segment ($k, k+1$)



Natural Cubic Spline

พิจารณาเฉพาะพิกัด x ในระบบ Cartesian ซึ่งเป็นฟังก์ชันของ u

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 \leq u \leq 1$$

เส้นโค้งที่มี $n + 1$ control points ประกอบด้วยจุดปลาย 2 จุด และ จุดภายใน $n - 1$ จุด โดยที่จุดภายในแต่ละจุด จะระบุเงื่อนไข parametric ด้วยอนุพันธ์อันดับที่ 0, 1 และ 2 ได้ 4 สมการ และ เงื่อนไขที่จุดปลายได้ 4 สมการ

อนุพันธ์อันดับ 1 $x'(0) = c_x$

อนุพันธ์อันดับ 2 $x''(0) = 2b_x$

$$x'(1) = 3a_x + 2b_x + c_x$$

$$x''(1) = 6a_x + 2b_x$$

อนุพันธ์อันดับ 0 $x(0) = d_x$

$$x(1) = a_x + b_x + c_x + d_x$$

Solving for the $k - 1, k^{\text{th}}$ Segments

สำหรับการหาสัมประสิทธิ์ $(a, b, c, d)^{k-1, k}$ ของเส้นโค้ง 2 เส้น ที่เชื่อมต่อที่ control point ที่ p_k สามารถสรุปได้ดังนี้

อนุพันธ์อันดับที่ 1 และ 2 ของเส้นโค้งที่ $k - 1$ และ k มีค่าเท่ากันที่จุด p_k

$$x'_{k-1}(1) = x'_k(0) \quad x''_{k-1}(1) = x''_k(0)$$

อนุพันธ์อันดับที่ 0 ของเส้นโค้งที่ $k - 1$ และ k มีค่าเท่ากันที่จุด p_k เท่ากับ p_k

$$x_{k-1}(1) = p_k \quad x_k(0) = p_k$$

ถ้าเรามีเส้นโค้ง n เส้น จุด p_k ที่เป็นจุดเชื่อมต่อของเส้นโค้งสองเส้น มีเพียง $n - 1$ จุด (จุดภายใน) ดังนั้นเราจะมีสมการเพียง $4(n - 1) = 4n - 4$ สมการ

แต่เส้นโค้ง n เส้นต้องการ $4n$ สมการ ดังนั้นเราต้องหาสมการเพิ่มอีก 4 สมการ



Specifying Boundary Conditions

สมการอีก 4 สมการที่เหลือ สามารถสร้างได้จาก จุดปลายของเส้นโค้ง คือจุด p_0 และ จุด p_n

อนุพันธ์อันดับที่ 2 ที่จุดปลายของ เส้นโค้งที่ 0 และเส้นโค้งที่ $n - 1$ มีค่าเท่ากับ 0

$$x_0''(0) = 0 \qquad x_{n-1}''(1) = 0$$

อนุพันธ์อันดับที่ 0 ที่จุดปลายของ เส้นโค้งที่ 0 และ $n - 1$ มีค่าเท่ากับพิกัดจุดนั้น

$$x_0(0) = p_0 \qquad x_{n-1}(1) = p_n$$

ดังนั้นเราจะมีสมการที่ได้จาก จุดภายใน p_k จำนวน $4n - 4$ สมการ และ สมการที่ได้จากเงื่อนไขขอบเขต ที่จุดปลายอีก 4 สมการ รวมทั้งสิ้น $4n$ สมการ ซึ่งเพียงพอสำหรับคำนวณหาค่าสัมประสิทธิ์ของเส้นโค้ง n เส้น

$(a, b, c, d)^k$ โดยที่ $k = 0$ ถึง $n - 1$

An Example

ถ้ากำหนด control points 3 จุด จะมีทั้งหมด $(3 - 1) \times 4 = 8$ สมการ

$$\begin{array}{l}
 \text{จุดภายใน} \\
 \left\{ \begin{array}{l} x'_0(1) = x'_1(0) \\ x''_0(1) = x''_1(0) \\ x_0(1) = p_1 \\ x_1(0) = p_1 \end{array} \right. \\
 \text{เงื่อนไขขอบเขต} \\
 \left\{ \begin{array}{l} x_0(0) = p_0 \\ x_1(1) = p_2 \\ x''_0(0) = 0 \\ x''_1(1) = 0 \end{array} \right.
 \end{array}
 \Rightarrow
 \begin{bmatrix}
 3 & 2 & 1 & 0 & 0 & 0 & -1 & 0 \\
 6 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\
 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 6 & 2 & 0 & 0
 \end{bmatrix}
 \begin{bmatrix}
 a_x^0 \\
 b_x^0 \\
 c_x^0 \\
 d_x^0 \\
 a_x^1 \\
 b_x^1 \\
 c_x^1 \\
 d_x^1
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 p_1 \\
 p_1 \\
 p_0 \\
 p_2 \\
 0 \\
 0
 \end{bmatrix}$$

ดังนั้นในการหา สัมประสิทธิ์ a, b, c, d ของตัวแปร x สำหรับ spline ที่มี control points 3 จุดสามารถทำได้โดยแก้สมการ 8 สมการ สำหรับ 8 ตัวแปร (หา **matrix** สำหรับ 2 จุด)

Hermite Interpolation

ข้อเสียของ Natural Cubic Spline คือถ้าเราเปลี่ยนตำแหน่งของ control point เพียงตำแหน่งเดียว จะทำให้รูปร่างของเส้นโค้งเปลี่ยนไปโดยสิ้นเชิง (ไม่สามารถควบคุมเส้นโค้งเฉพาะบริเวณที่กำหนดได้)

Hermite Spline แก้ไขข้อจำกัดนี้ โดยที่สำหรับเส้นโค้งแต่ละช่วงจะขึ้นอยู่กับจุดปลาย 2 ของช่วงนั้นถ้า $\mathbf{P}(u)$ แทนฟังก์ชันพหุนามกำลัง 3 สำหรับเส้นโค้งช่วงที่อยู่ระหว่าง control point p_k กับ p_{k+1}

$$\mathbf{P}(u) = \mathbf{a}u^3 + \mathbf{b}u^2 + \mathbf{c}u + \mathbf{d} \quad 0 \leq u \leq 1$$

แล้วนิยามเงื่อนไขขอบเขต ดังนี้

$$\mathbf{P}(0) = \mathbf{p}_k$$

$$\mathbf{P}'(0) = \mathbf{Dp}_k$$

$$\mathbf{P}(1) = \mathbf{p}_{k+1}$$

$$\mathbf{P}'(1) = \mathbf{Dp}_{k+1}$$

โดยที่ \mathbf{D} แทน ตัวดำเนินการอนุพันธ์ อันดับที่ 1

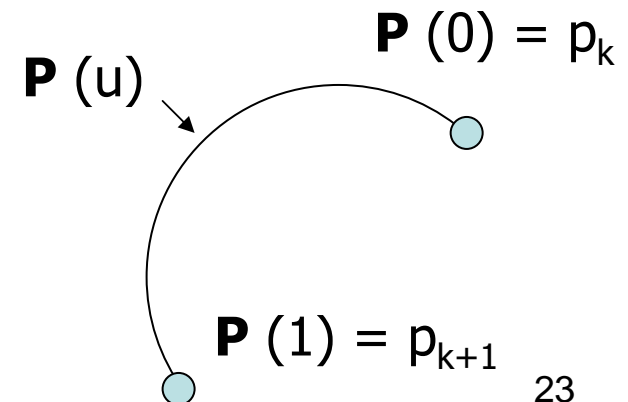
Hermite Spline Matrix

จากเงื่อนไขของ Hermite เราสามารถเขียนในรูปของ Matrix ได้ดังนี้

$$\begin{array}{l} \text{0th order} \\ \mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \end{array} \quad \begin{array}{l} \text{1st order} \\ \mathbf{P}'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} \end{array}$$

แทนค่าตัวแปร $u = 0$ และ 1 แล้วหาสมการตามเงื่อนไขของ Hermite Spline ได้

$$\begin{bmatrix} \mathbf{P}(0) \\ \mathbf{P}(1) \\ \mathbf{P}'(0) \\ \mathbf{P}'(1) \end{bmatrix} = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$



Solving for Coefficients

จากสมการ Matrix ของ Hermite Spline เราสามารถแก้หาเวกเตอร์สัมประสิทธิ์ **a, b, c, d** ได้โดยการหา inverse ของ Matrix

$$\begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix} = \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}$$

โดยที่ \mathbf{M}_H คือ Hermite Matrix ซึ่งมีค่าเฉพาะสำหรับ Hermite Spline

Hermite Closed Form

เมื่อแก้สมการหา ค่าเวกเตอร์ สัมประสิทธิ์ได้แล้ว เราสามารถเขียน Hermite Spline สำหรับจุด **P** (u) ใดๆ ระหว่าง control point ทั้งสอง ได้

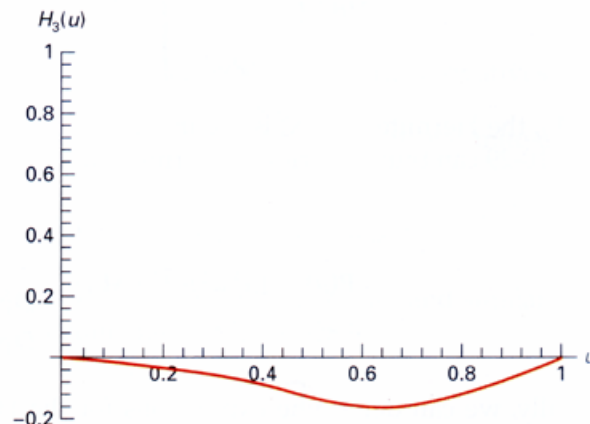
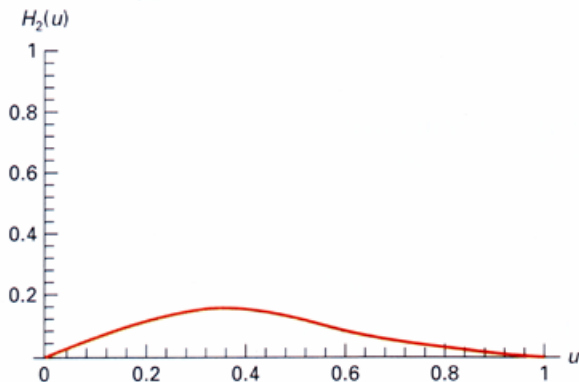
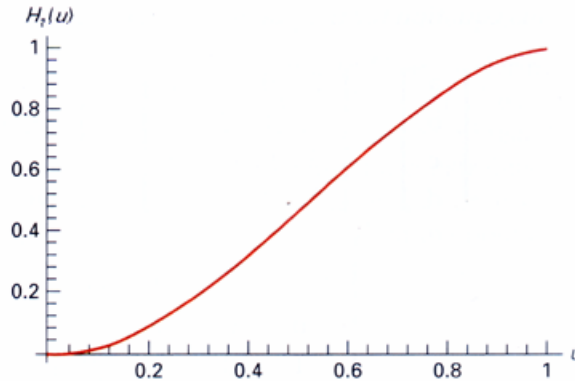
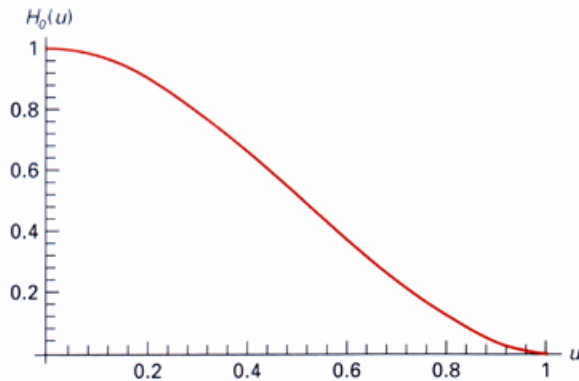
$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\ \mathbf{Dp}_{k+1} \end{bmatrix}$$

จากสมการพบว่าผู้ใช้ต้องกำหนด ค่าความชัน **Dp** ที่ control points ด้วย

Hermite Spline as a Blending Function

$$\begin{aligned} \mathbf{P}(u) &= \mathbf{p}_k (2u^3 - 3u^2 + 1) + \mathbf{p}_{k+1} (-2u^3 + 3u^2) + \mathbf{Dp}_k (u^3 - 2u^2 + u) + \mathbf{Dp}_{k+1} (u^3 - u^2) \\ &= \mathbf{p}_k H_0(u) + \mathbf{p}_{k+1} H_1(u) + \mathbf{Dp}_k H_2(u) + \mathbf{Dp}_{k+1} H_3(u) \end{aligned}$$

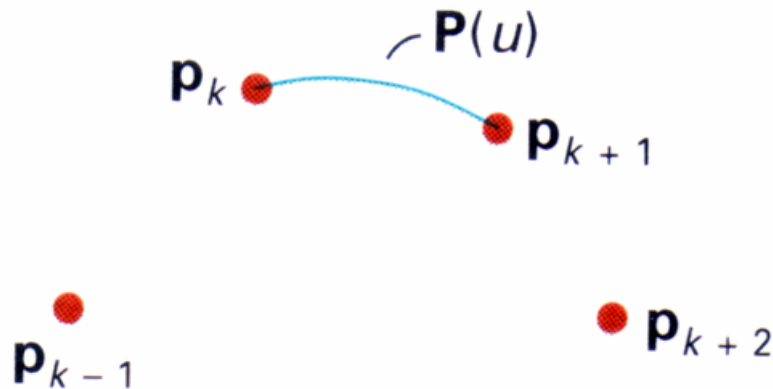


เมื่อกำหนดจุดปลาย 2 จุด และค่าอนุพันธ์ที่จุดดังกล่าวแล้ว เราสามารถหา พิกัดของจุดใดๆ ภายในช่วงได้ โดยการหาผลบวกถ่วงน้ำหนักของ vector โดยที่ค่าผลคูณขึ้นอยู่กับระยะ u ระหว่าง \mathbf{p}_k และ \mathbf{p}_{k+1}

Cardinal Spline

ข้อเสียของ Hermite Spline คือผู้ออกแบบต้องกำหนด ทั้งพิกัดของ control points (\mathbf{p}) และ ความชันของเส้นโค้ง ณ จุด control points (\mathbf{Dp}) ซึ่งไม่สะดวก สำหรับการใช้งาน บางประเภท

Cardinal Spline แก้ปัญหานี้โดยทำการคำนวณค่าความชันโดยอัตโนมัติ ด้วยวิธีการ Finite Difference (FD) ทั้งนี้ สำหรับเส้นโค้ง 1 ช่วงต้องพิจารณา control points จำนวน 4 จุด พร้อมๆ กัน ดังรูป และ สมการต่อไปนี้



อนุพันธ์ลำดับที่ 0 ที่จุดปลายมีค่าเท่ากับพิกัด

$$\mathbf{P}(0) = \mathbf{p}_k \quad \mathbf{P}(1) = \mathbf{p}_{k+1}$$

อนุพันธ์ลำดับที่ 1 ที่จุดปลายหาได้จาก FD

$$\mathbf{P}'(0) = 0.5(1-t)(\mathbf{p}_{k+1} - \mathbf{p}_{k-1})$$

$$\mathbf{P}'(1) = 0.5(1-t)(\mathbf{p}_{k+2} - \mathbf{p}_k)$$

ค่าตัวแปร t เป็นตัวควบคุม ความตึง หรือ tension

Cardinal Spline Formula

หมายเหตุ สำหรับจุดปลายสุด p_0 จะไม่สามารถหา p_{0-1} ได้ และ สำหรับจุด p_{n-1} จะไม่สามารถหา $p_{(n-1)+2}$ ได้ ซึ่งจะทำให้สมการหายไป 2 สมการ คือ $P_0'(0)$ และ $P_{n-1}'(1)$ วิธีการแก้ไขคือ อาจแทนสมการที่หายไป ด้วยเงื่อนไขอนุพันธ์อันดับสอง เหมือนกรณี natural cubic spline ได้ โดย กำหนดให้ $P_0''(0) = 0$ และ $P_{n-1}''(1) = 0$ สำหรับจุดภายในที่ k สัมประสิทธิ์ ของเส้นโค้งที่ k หาได้จากชุดสมการ

$$P_k(u) = a_k u^3 + b_k u^2 + c_k u + d_k$$

$$P_k(0) = p_k \longrightarrow P_k(0) = d_k = p_k$$

$$P_k(1) = p_{k+1} \longrightarrow P_k(1) = a_k + b_k + c_k + d_k = p_{k+1}$$

$$P'_k(u) = 3a_k u^2 + 2b_k u + c_k \quad s = 0.5(1-t)$$

$$P'(0) = 0.5(1-t)(p_{k+1} - p_{k-1}) \longrightarrow c_k = s(p_{k+1} - p_{k-1})$$

$$P'(1) = 0.5(1-t)(p_{k+2} - p_k) \longrightarrow 3a_k + 2b_k + c_k = s(p_{k+2} - p_k)$$

Cardinal Spline Matrix

นำระบบสมการ ไปจัดรูป (จาก slide projector) แล้วเขียนในรูป Matrix ได้ดังนี้

$$\begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \frac{s-1}{s} & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ \frac{3}{s} & \frac{2}{s} & \frac{1}{s} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a}_k \\ \mathbf{b}_k \\ \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix}$$

หา Inverse ของ Matrix เพื่อแก้สมการหาสัมประสิทธิ์ \mathbf{a} , \mathbf{b} , \mathbf{c} , \mathbf{d}

$$\begin{bmatrix} \mathbf{a}_k \\ \mathbf{b}_k \\ \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix} = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

ให้ นักศึกษา ไปทดลองพิสูจน์ที่มาของสมการนี้ด้วยตนเอง



Cardinal Closed Form

เราสามารถเขียน Cardinal Spline สำหรับเส้นโค้งที่ k ในรูป matrix ได้ดังนี้

$$\mathbf{P}_k(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a}_k \\ \mathbf{b}_k \\ \mathbf{c}_k \\ \mathbf{d}_k \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \mathbf{M}_{CAR} \cdot \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

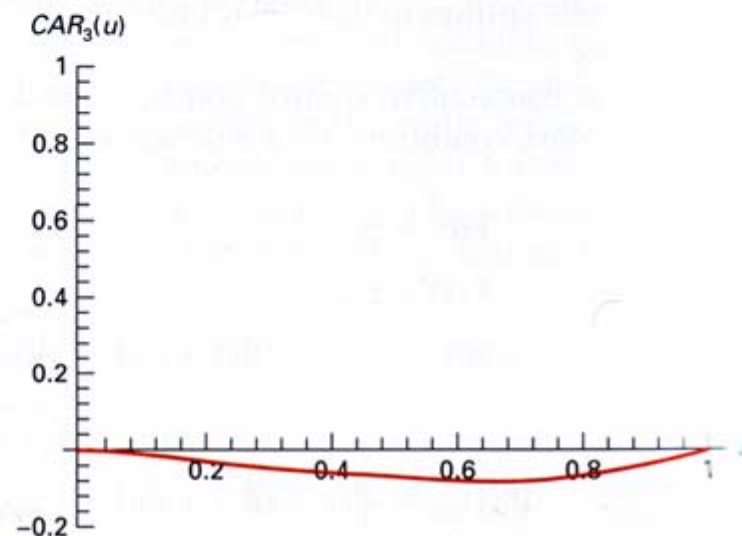
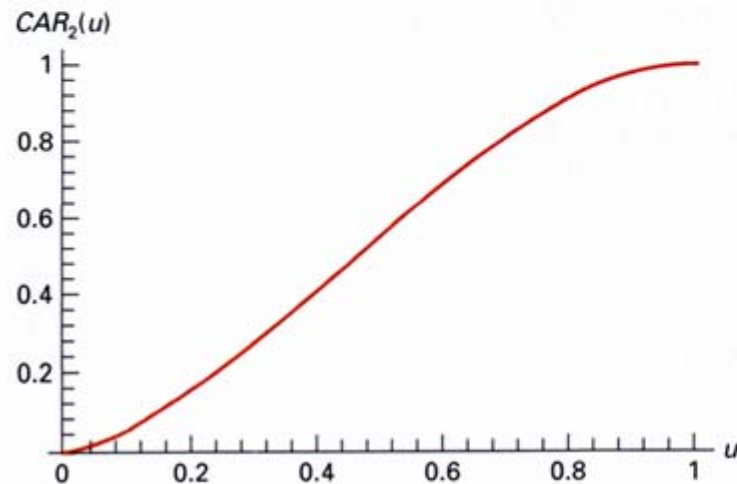
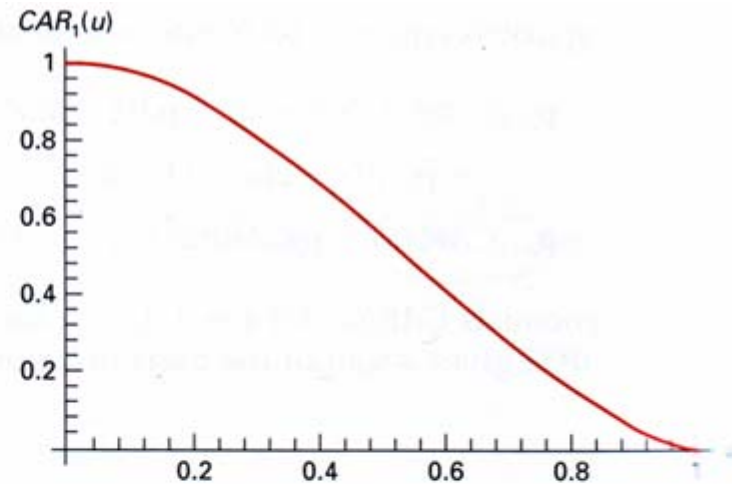
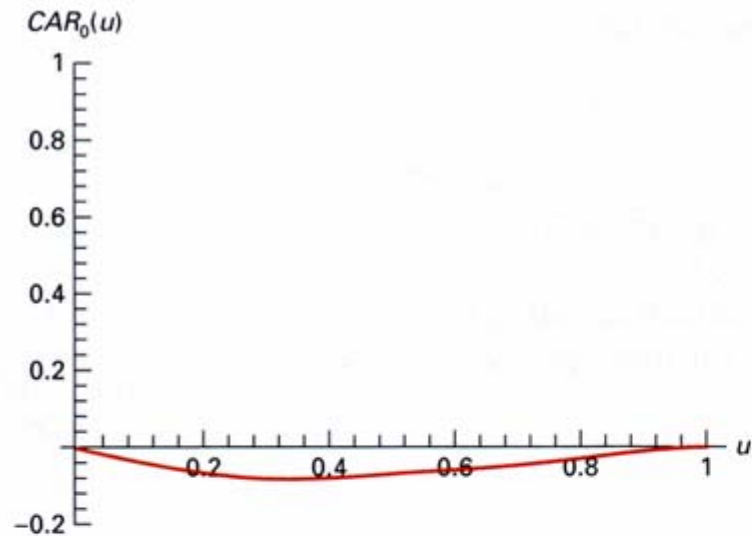
$$= \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

เมื่อกระจายแล้วจะอยู่ในรูป

$$\mathbf{P}_k(u) = \mathbf{p}_{k-1} \text{CAR}_0(u) + \mathbf{p}_k \text{CAR}_1(u) + \mathbf{p}_{k+1} \text{CAR}_2(u) + \mathbf{p}_{k+2} \text{CAR}_3(u)$$



Cardinal Basis Function



Bezier Curves and Surfaces

Bezier Spline เป็น Approximating Spline (เส้นโค้งไม่จำเป็นต้องผ่าน control point ทุกจุด) ซึ่งมีคุณสมบัติคือ ผู้ใช้งานสามารถ ออกแบบเส้นโค้งได้สะดวก ทำให้มีใช้แพร่หลายในซอฟต์แวร์ Graphics เกือบทุกชนิด

โดยทฤษฎีแล้วเราสามารถสร้าง Bezier Curve ได้ 3 วิธีเหมือนกับ Interpolating Spline แต่วิธีการกำหนดพหุนามของ Basis/Blending Function จะสะดวกที่สุด

กำหนด control point จำนวน $(n+1)$ จุด $\mathbf{p}_k = (x_k, y_k, z_k)$ โดยที่ $k = 0 \dots n$ แล้ว Blending Function ของเส้นโค้ง ทั้งเส้น $\mathbf{P}(u)$ นิยามโดย

$$\mathbf{P}(u) = \sum_{k=0}^n \mathbf{p}_k BEZ_{k,n}(u)$$

โดยที่ $BEZ_{k,n}$ คือ Bernstein Polynomials อันดับที่ n ของจุดที่ k

$$BEZ_{k,n}(u) = C(n, k) u^k (1-u)^{n-k} \quad \text{โดยที่} \quad C(n, k) = \frac{n!}{k!(n-k)!}$$

Bezier Basis Function

เราสามารถเขียน Bezier Basis Function ในรูปความสัมพันธ์เวียนบังเกิดได้ดังนี้

$$BEZ_{k,n}(u) = (1-u)BEZ_{k,n-1}(u) + uBEZ_{k-1,n-1}(u) \quad n > k \geq 1$$

โดยที่ $BEZ_{k,k}(u) = u^k \quad BEZ_{0,k}(u) = (1-u)^k$

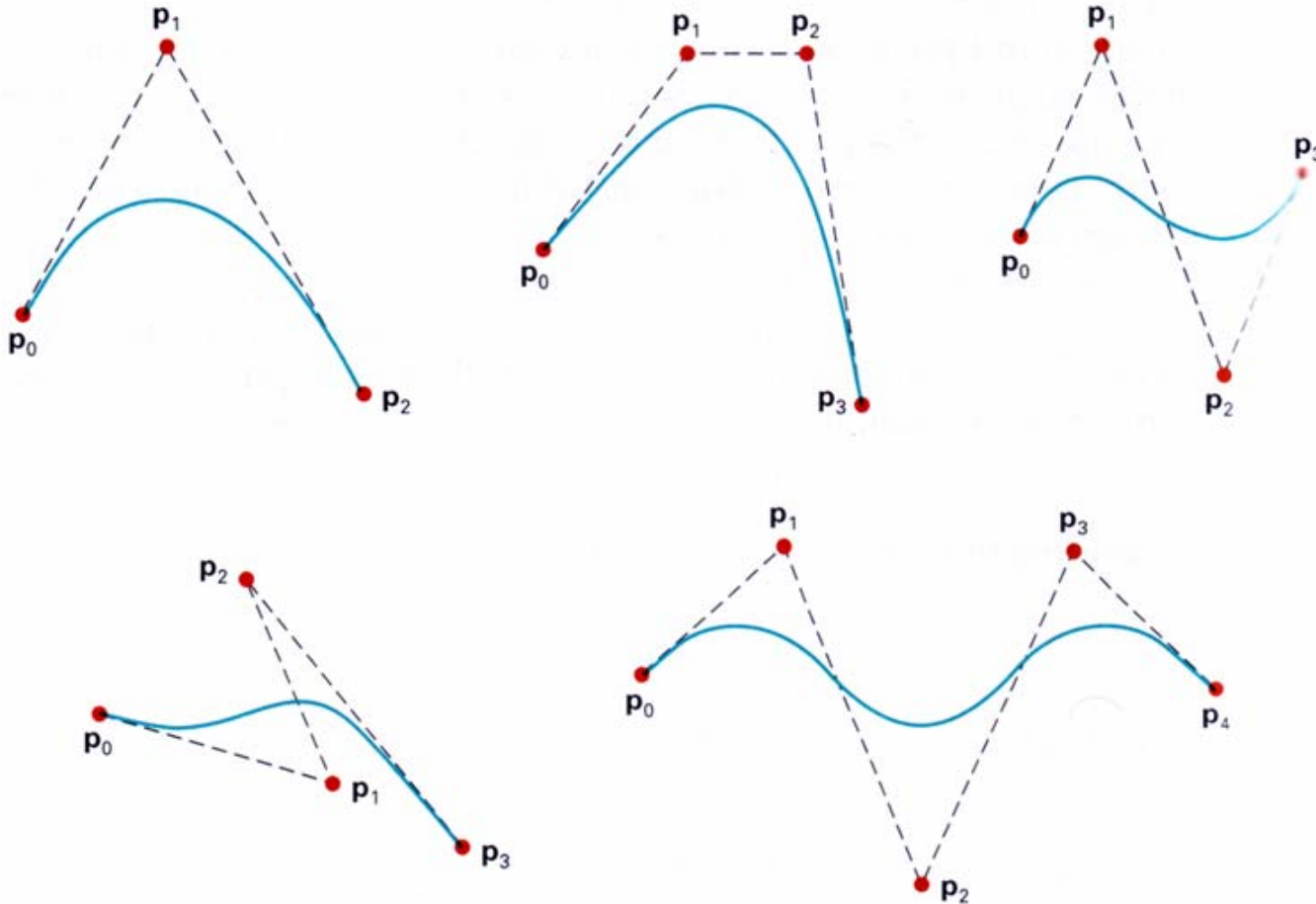
หรืออาจใช้การกระจาย Binomial มาช่วยได้ $C(n,k) = \frac{n-k+1}{k} C(n,k-1)$

$$\begin{aligned} BEZ_{0,n}(u) &= C(n,0) \cdot u^0 \cdot (1-u)^{n-0} \\ &= \frac{n!}{0!(n-0)!} \cdot (1) \cdot (1-u)^n = (1-u)^n \end{aligned}$$

$$\begin{aligned} BEZ_{1,n}(u) &= C(n,1) \cdot u^1 \cdot (1-u)^{n-1} \\ &= \frac{n-1+1}{1} \cdot C(n,0) \cdot u^1 (1-u)^{n-1} = n \cdot u^1 (1-u)^{n-1} \end{aligned}$$

Bezier Curve Appearance

อันดับ (degree) ของเส้นโค้ง Bezier จะมีค่าน้อยกว่าจำนวน control point อยู่ 1



Properties of Bezier Curves

- เส้นโค้งจะผ่าน control point จุดปลายสองจุดเสมอ

$$\mathbf{P}(0) = \mathbf{p}_0 \quad \mathbf{P}(1) = \mathbf{p}_n$$

- ค่าอนุพันธ์อันดับที่ 1 ของจุดปลายเส้นโค้ง = ความชัน ของเส้นตรง ส่วนปลาย

$$\mathbf{P}'(0) = -n\mathbf{p}_0 + n\mathbf{p}_1 \quad \mathbf{P}'(1) = -n\mathbf{p}_{n-1} + n\mathbf{p}_n$$

- ค่าอนุพันธ์อันดับที่ 2 ของจุดปลายเส้นโค้ง = ความโค้ง ของเส้นโค้ง ส่วนปลาย

$$\mathbf{P}''(0) = n(n-1)[(\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)]$$
$$\mathbf{P}''(1) = n(n-1)[(\mathbf{p}_{n-2} - \mathbf{p}_{n-1}) - (\mathbf{p}_{n-1} - \mathbf{p}_n)]$$

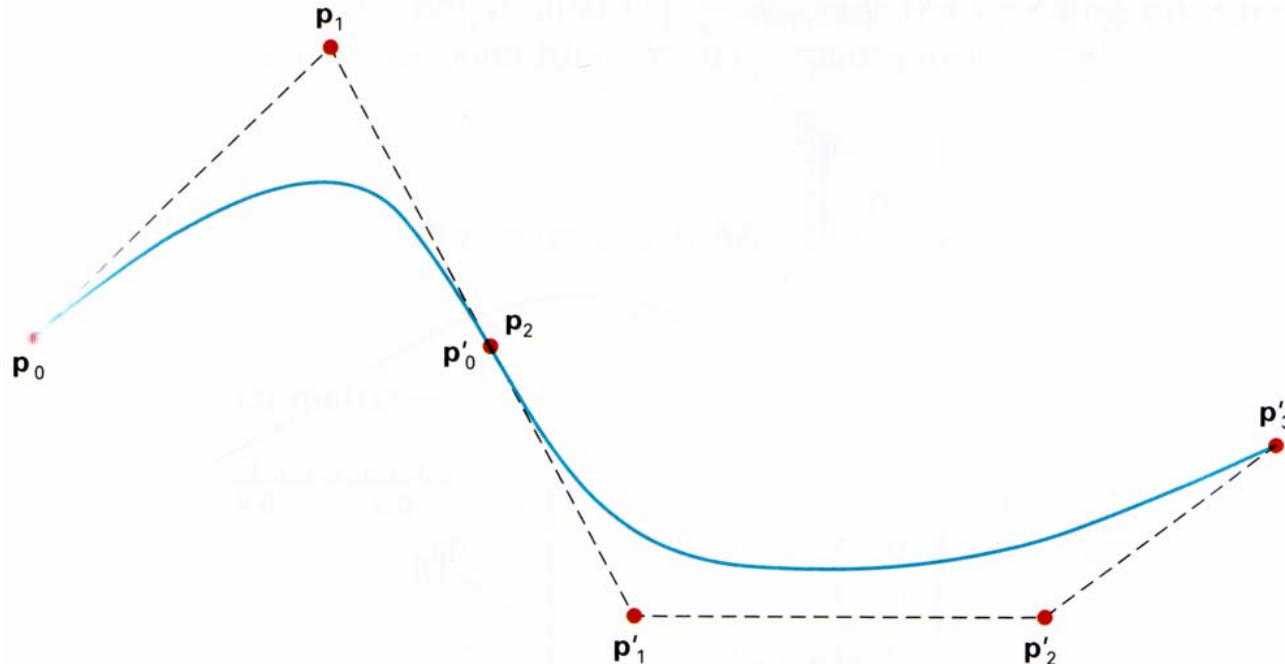
โดยที่ขนาดของอนุพันธ์ในข้อ 1 และ 2 แปรผันตรงกับ n และ n^2 ตามลำดับ

- เส้นโค้ง Bezier จะอยู่ใน Convex Hull เสมอ

$$\sum_{k=0}^n BEZ_{k,n}(u) = 1$$

Design Techniques

จุดใดๆ บนเส้นโค้ง Bezier คำนวณมาจากฟังก์ชันพหุนาม อันดับที่ n ซึ่งถึงแม้ว่าจะเพิ่มความเร็วโดยใช้ recursion แต่ก็ยังไม่เหมาะสมสำหรับการวาดเส้นโค้งที่ซับซ้อน นักออกแบบจึงนิยม นำเส้นโค้ง Bezier อันดับต่ำมาต่อกันดังรูป



- จุดแรกของเส้นโค้งที่ 2 (p'_0) อยู่ที่พิกัดเดียวกับจุดสุดท้ายของเส้นโค้งแรก (p_2)
- เส้นตรง $p'_0 p'_1$ อยู่ในแนวเส้นตรงเดียวกันกับ $p_1 p_2$ และมีความยาวเท่ากัน

Cubic Bezier Curves

ด้วยหลักการเดียวกันนี้เราสามารถสร้างเส้นโค้งใดๆ จากเส้นโค้ง Bezier อันดับที่ 3 (Cubic Bezier) ได้ ซึ่งนิยามในรูปของ Blending Function ดังนี้

$$BEZ_{0,3} = (1-u)^3$$

$$BEZ_{1,3} = 3u(1-u)^2$$

$$BEZ_{2,3} = 3u^2(1-u)$$

$$BEZ_{3,3} = u^3$$

โดยมีเงื่อนไขขอบนพจน์อันดับที่ 1 ดังนี้ (แก้ใน Text ด้วย)

$$\mathbf{P}'(0) = 3(\mathbf{p}_1 - \mathbf{p}_0)$$

$$\mathbf{P}'(1) = 3(\mathbf{p}_3 - \mathbf{p}_2)$$

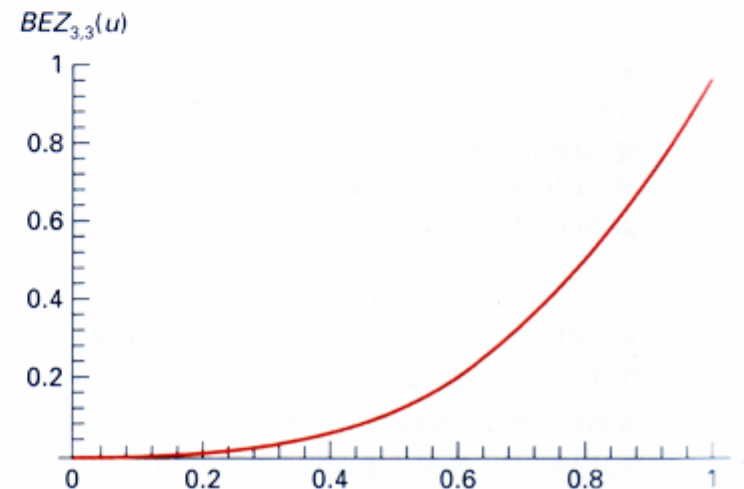
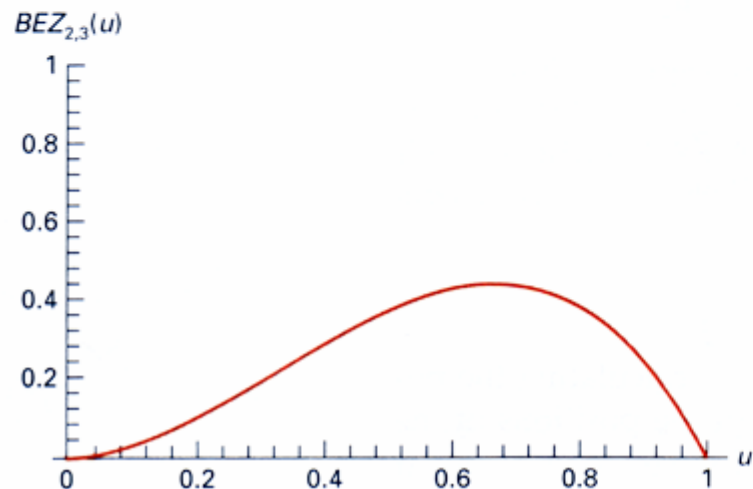
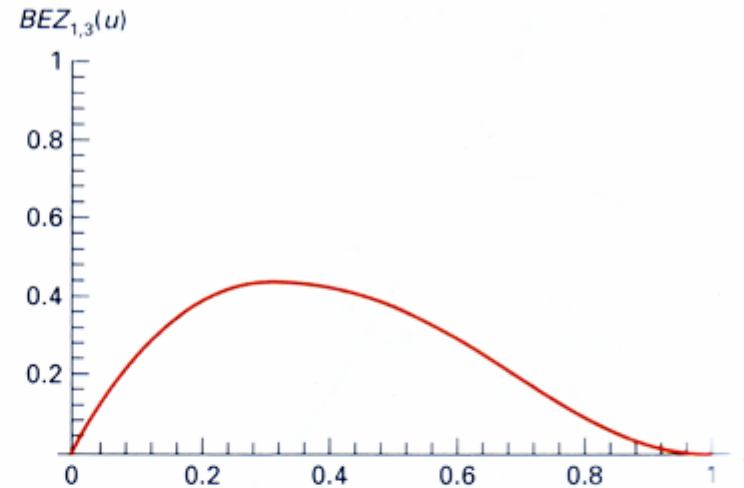
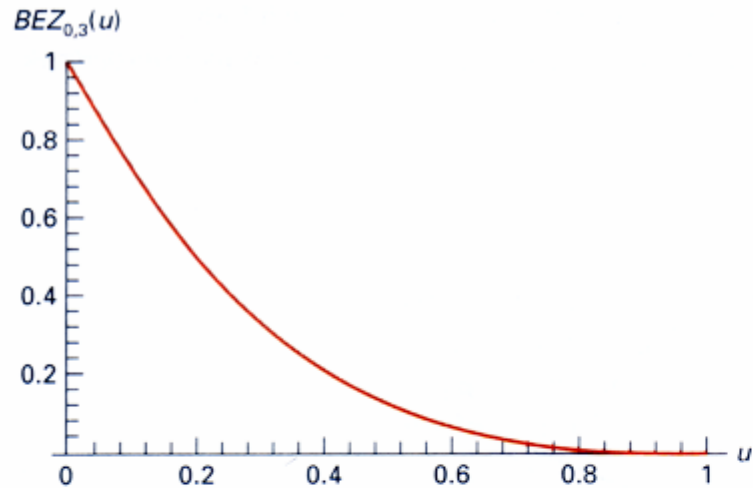
หรือในรูปของ Matrix

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \mathbf{M}_{BEZ} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{M}_{BEZ} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$



Cubic Bezier Basis Functions

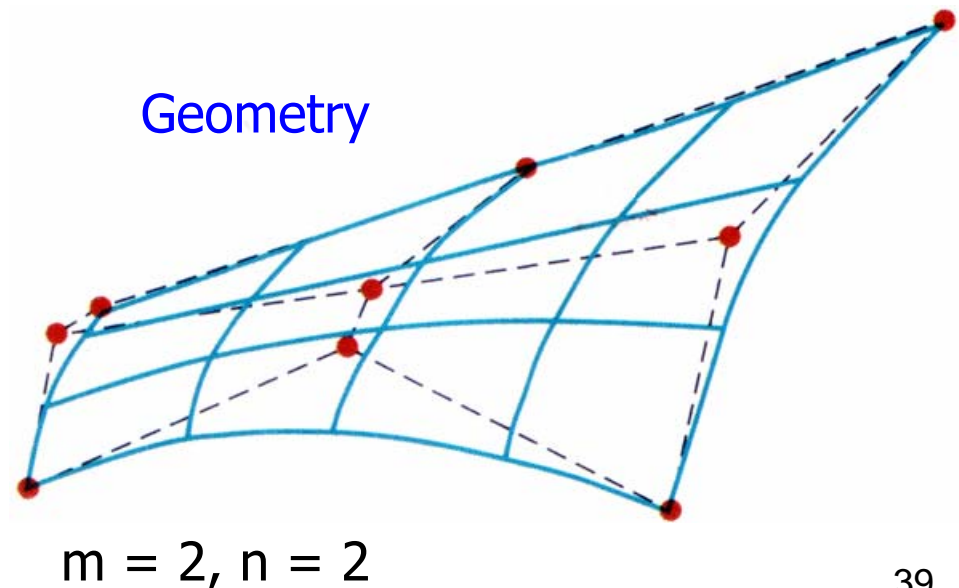
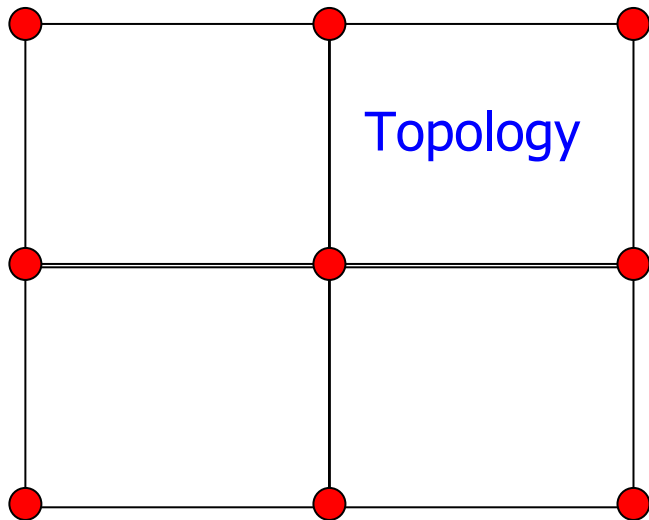


Bezier Surfaces

พื้นผิว Bezier (ใช้ในการออกแบบครั้งแรกสำหรับ ตัวถังรถ Renault) สร้างได้จากผลคูณ Tensor (Tensor Product) ของเส้นโค้ง Bezier สองชุด ดังนี้

$$\mathbf{P}(u, v) = \sum_{j=0}^m \sum_{k=0}^n \mathbf{p}_{j,k} BEZ_{j,m}(u) BEZ_{k,n}(v)$$

โดยที่ $\mathbf{p}_{j,k}$ คือ control point หนึ่งจาก $(m + 1) \times (n + 1)$ control points





Conclusions

- 3-D Object Representations
 - Polygonal Surfaces
 - Planes in 3D Space
 - Quadric Surfaces and Blobs
- Spline Representations
 - Interpolation and Approximation Spline
 - Continuity Conditions
 - Cubic Spline Interpolation
 - Bezier Curves and Surfaces
 - B-Spline Curves and Surfaces
 - Other Splines and Their Conversions
- Introduction to OpenGL
 - OpenGL Programming using C/C++