Suranaree University
of Technology

COMPUTER GRAPHICS
School of Computer Engineering

Lecture 8 3-Dimensional Object Representations (Part I)

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++

3D-Object Representations

การแสดงผลสำหรับองค์ประกอบเรขาคณิต (Geometry) สามารถแบ่งเป็นประเภทย่อยๆ ได้ดังนี้

Polygon และ Quadric Surfaces

เหมาะสำหรับ การแสดงรูปวัตถุอย่างง่าย อาทิเช่น รูปหลายเหลี่ยม วงรี (ทรงรี)

Spline Surfaces และ ขั้นตอนวิธีการขึ้นรูป (CSG)

เหมาะสำหรับ การแสดงรูปวัตถุที่มีความซับซ้อนเพิ่มขึ้น เช่น ชิ้นส่วนเครื่องบิน และ โครงสร้างทางวิศวกรรมต่างๆ

Fractal และ การอธิบายวัตถุด้วยลำดับขั้นของสมการคณิตศาสตร์

เหมาะสำหรับ การแสดงเนื้อหาที่ปรากฏตามธรรมชาติ เช่น ทุ่งหญ้า ใบไม้ ละออง ฝุ่น และ ควันไฟ

3D-Object Representations

การสร้างแบบจำลองทางกายภาพ (Physically Based Modeling)

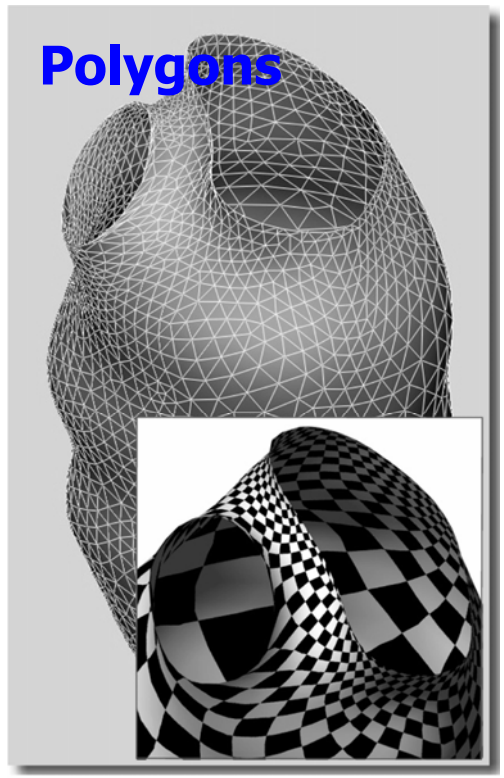
เหมาะสำหรับ การแสดงรูปวัตถุ ที่มีการเคลื่อนไหว ตามแรงกิริยาและปฏิกิริยา ซึ่งกำหนด การโต้ตอบกับผู้ใช้ (เช่น การเคลื่อนไหวของตัวละครในเกม) เสื้อผ้า และกล้ามเนื้อ

การเข้ารหัส Octree

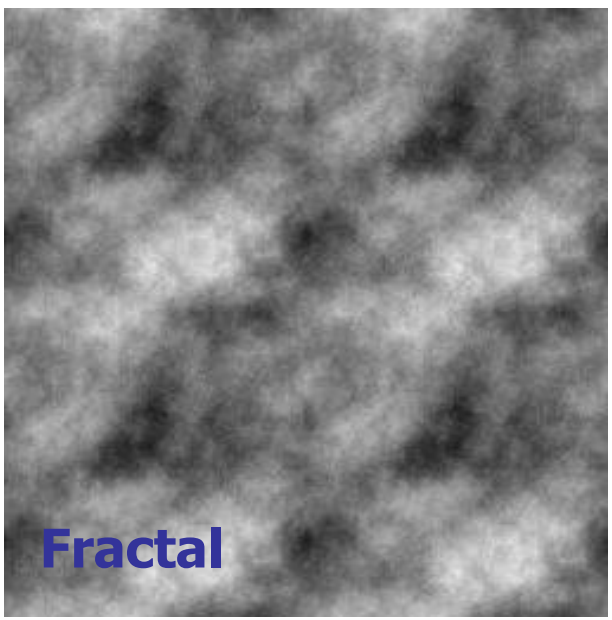
เหมาะสำหรับการแสดงวัตถุ ที่เน้นลักษณะโครงสร้างภายใน ได้แก่ ภาพถ่ายทางการแพทย์ จากเครื่อง Computed Tomography (CT) การแสดงผลข้อมูลทางอุตุนิยมวิทยา และ ธรณีวิทยา การแสดงผลข้อมูลการวิเคราะห์ทางวิทยาศาสตร์ในหลายมิติ ของ ข้อมูลชนิด เต็มหน่วย (Discrete) เช่น อุณหภูมิ คลื่นวิทยุ ฯลฯ

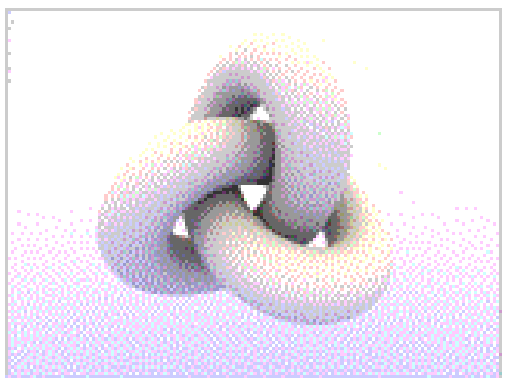
การแสดงผลดังกล่าว สามารถจำแนกได้เป็นแบบ Boundary Representation (B-Rep) และแบบ Space Partitioning ซึ่งจะครอบคลุมเนื้อหาหลัง midterm

3D-Object Examples



Conformal Map



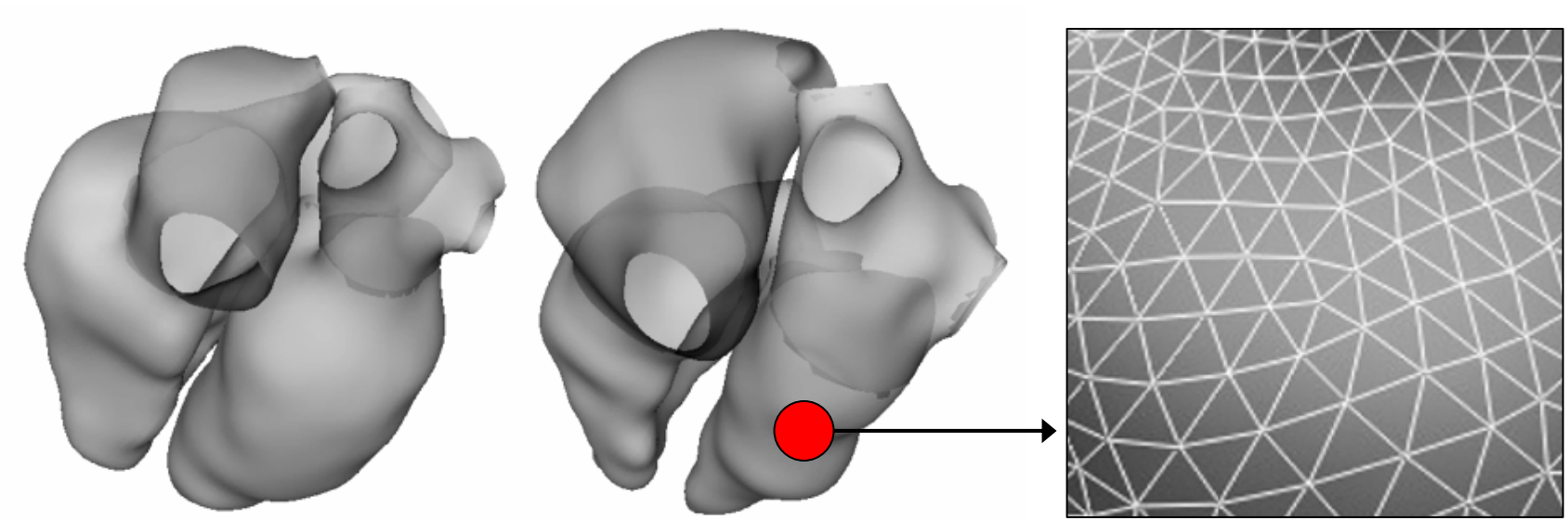


CSG

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

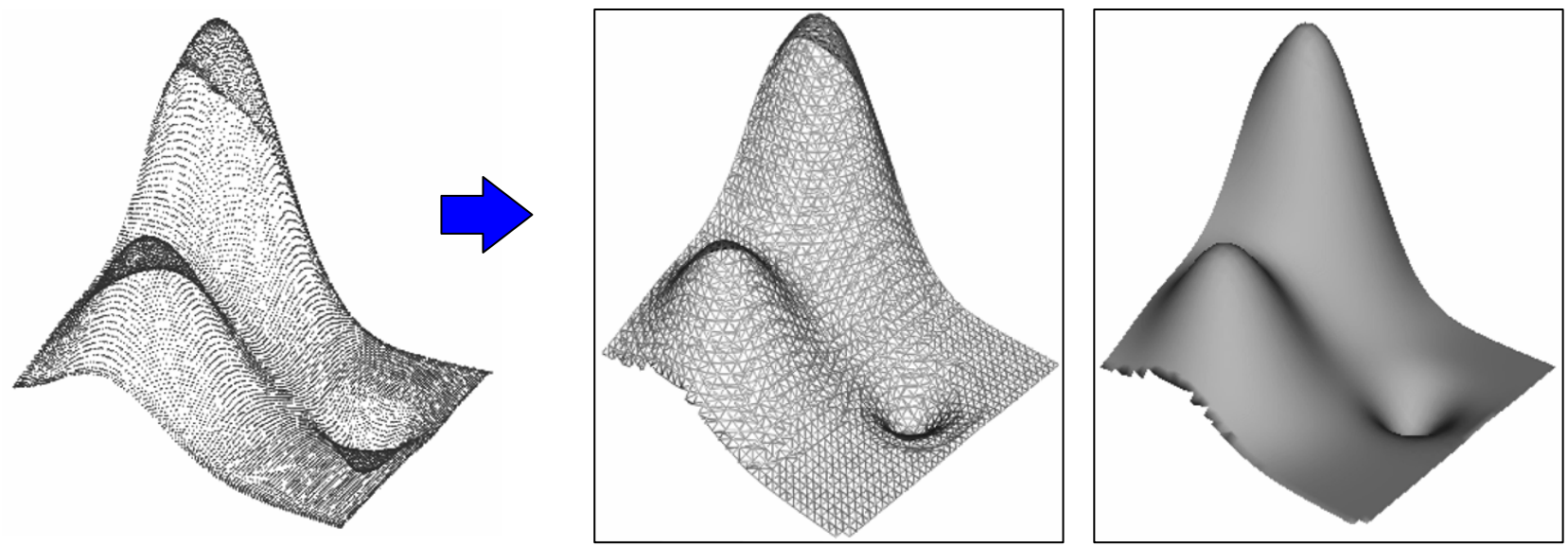
รูปนี้ แสดงตัวอย่างของการแสดง วัตถุด้วย Polygons ที่เป็นรูปสามเหลี่ยมต่อกัน (เรียกว่า Tile หรือ Tessellate เหมือนการปูกระเบื้อง หรือ Mosaic)



Polygonal Surface Display

ในทางปฏิบัติ เราสามารถแสดงผล Polygon ได้หลายวิธี กล่าวคือ

- แบบ Vertices คือวาดเฉพาะจุดยอดมุมของ Polygons ใช้เมื่อต้องการความเร็วสูง มากในการแสดงผล
- แบบ Edges วาดขอบของ Polygons ด้วย เมื่อต้องการแสดงรูปทรงของวัตถุ
- แบบ Shaded Render คือวาดพื้นผิว โดยพิจารณาคุณสมบัติด้านแสงเงาด้วย เพื่อ แสดง วัตถุ ให้ใกล้เคียงของจริงมากที่สุด



Polygon Tables

โดยทั่วไป เราจะจัดเก็บ Polygons ไว้ในโครงสร้างข้อมูลชนิดตาราง (Array) ประกอบด้วย

- **Geometric Tables**

  ประกอบด้วยข้อมูลทางเรขาคณิตของ Polygons ได้แก่ พิกัดจุดยอด (vertices) ของ Polygon แต่ละชิ้น

- **Topological Tables**

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

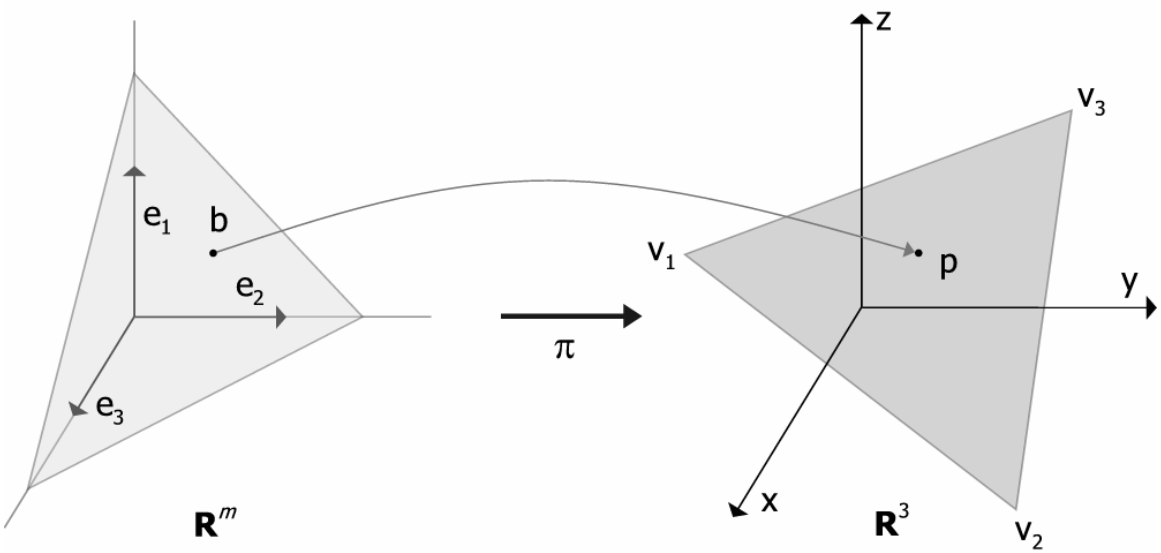
- **Attribute Tables**

  ประกอบด้วยข้อมูลทางกายภาพเพื่อแสดงผล เช่น ความโปร่งใส ดัชนีสะท้อน สี

Topology vs. Geometry

นิยามทางคณิตศาสตร์ ของ Topology และ Geometry สามารถอธิบายได้ด้วย ทฤษฎี Manifold (ไม่กล่าวถึงในที่นี้) มีที่ใช้แพร่หลายในวิศวกรรมหลายแขนง

สำหรับ Computer Graphics องค์ประกอบทางคณิตศาสตร์ ดังกล่าวจำเป็น สำหรับการ แสดงผล พื้นผิวในสามมิติ (3D surface)



Topological Realisation | K |

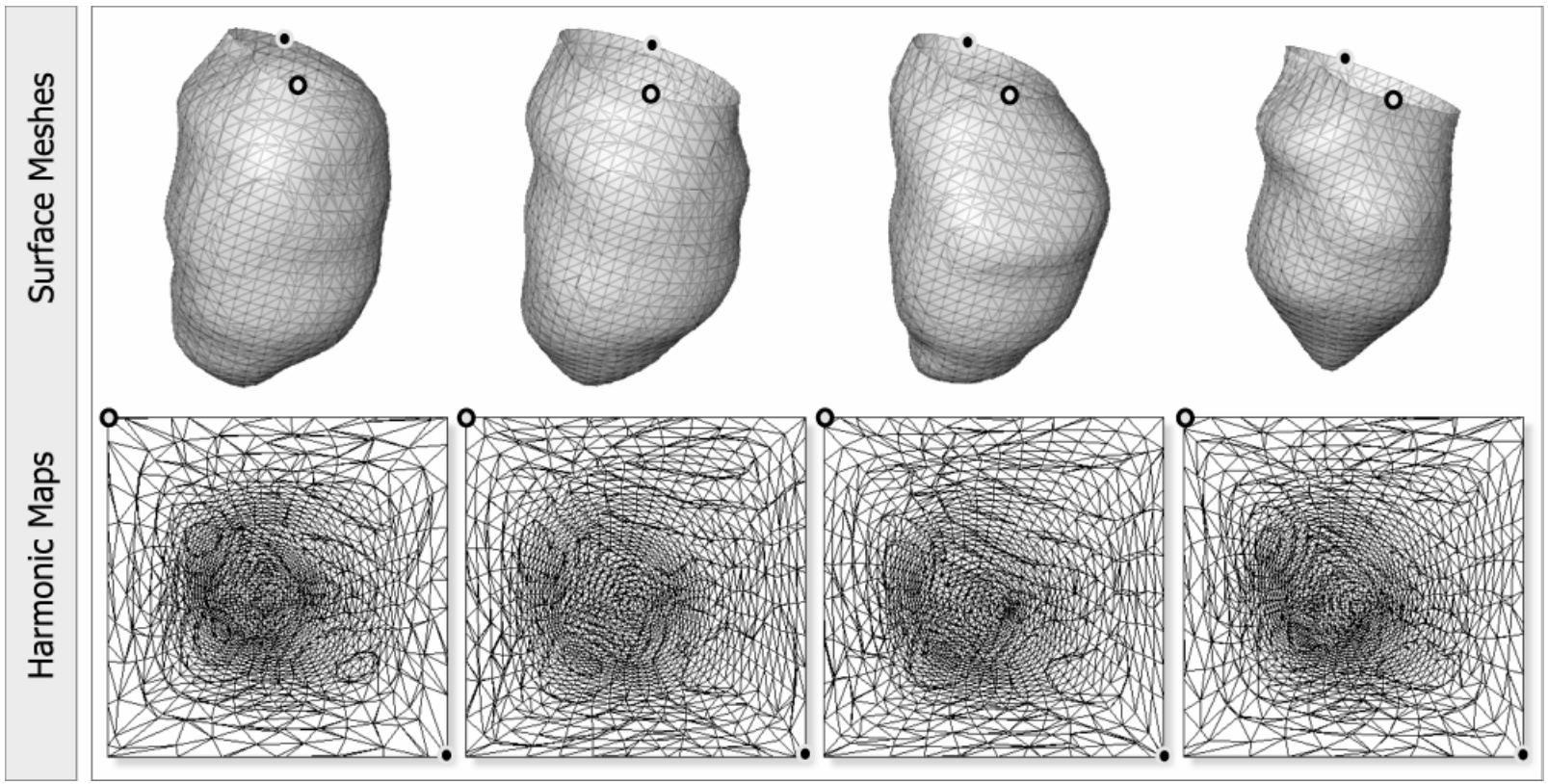
Geometrical Realisation $\pi_V(|K|)$

Topology นิยามการเชื่อมต่อระหว่าง จุดยอด (vertex) ของ polygon เพื่อนำมาประกอบเป็นพื้นผิว (ซ้าย)

Geometry นิยาม พิกัดทางกายภาพของจุดยอดนั้นในปริภูมิที่สนใจ (ขวา)

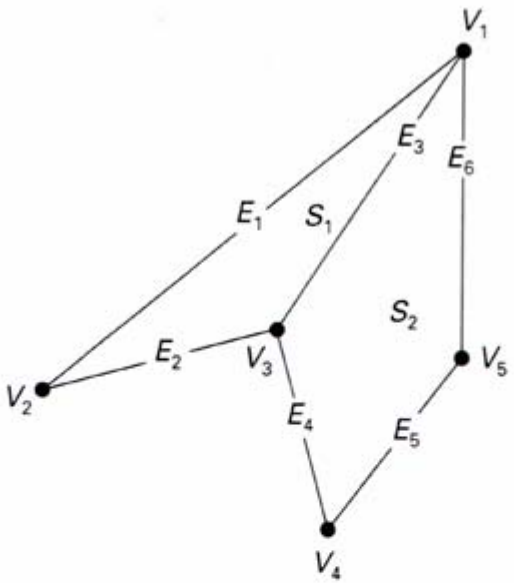
An Example

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



Implementations

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)

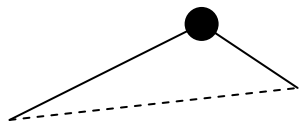
ตาราง edge จะเก็บ pointer ที่ชี้ไปยังตาราง vertices เพื่อกำหนดจุดปลายสองจุดของเส้นตรง (edge)

ตาราง face จะเก็บ pointer ที่ชี้ไปยัง ตาราง edge เพื่อกำหนดเส้นขอบที่นิยาม polygon

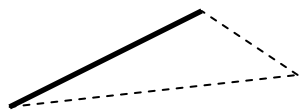
Surface Integrity

จากโครงสร้างข้อมูลที่กำหนด เราสามารถตรวจสอบ ได้ว่า Surface ที่สร้างขึ้น มีความถูกต้อง และ ครบถ้วนสมบูรณ์ เพียงใด โดยพิจารณาจากเงื่อนไขต่อไปนี้

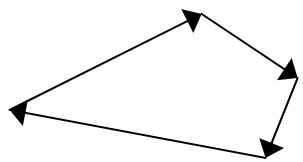
1. แต่ละ Vertex จะต้องปรากฏเป็นส่วนประกอบของ Edge อย่างน้อย 2 เส้น



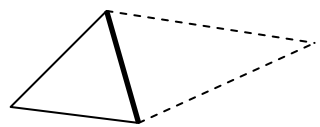
2. Edge แต่ละเส้น ต้องเป็นส่วนประกอบของ Polygon อย่างน้อย 1 ชิ้น



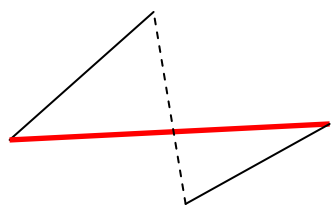
3. Polygon ทุกชิ้นต้องครบรอบเป็นวงปิด close



4. Polygon แต่ละชิ้น ต้องมี Edge ที่ใช้ ร่วมกับ Polygon อื่นๆ อย่างน้อย 1 เส้น



5. ไม่มี edge สองเส้นใดๆ ตัดกัน



Plane Equations

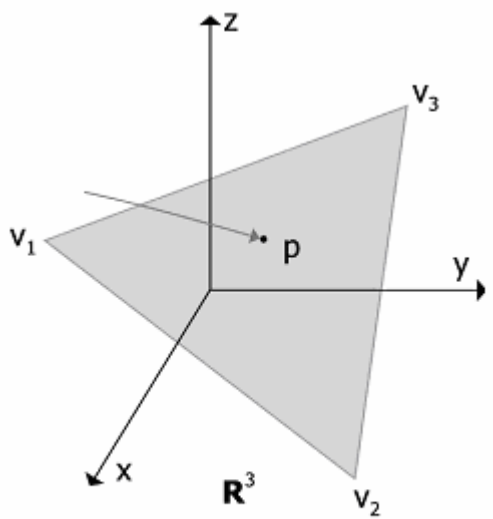
การแสดงผลแบบกราฟิก ประกอบด้วยการคำนวณหลายขั้นตอน ได้แก่

1) การ แปลงระหว่าง model, world, viewing และ device coordinates
2) การบ่งชี้ 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)

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

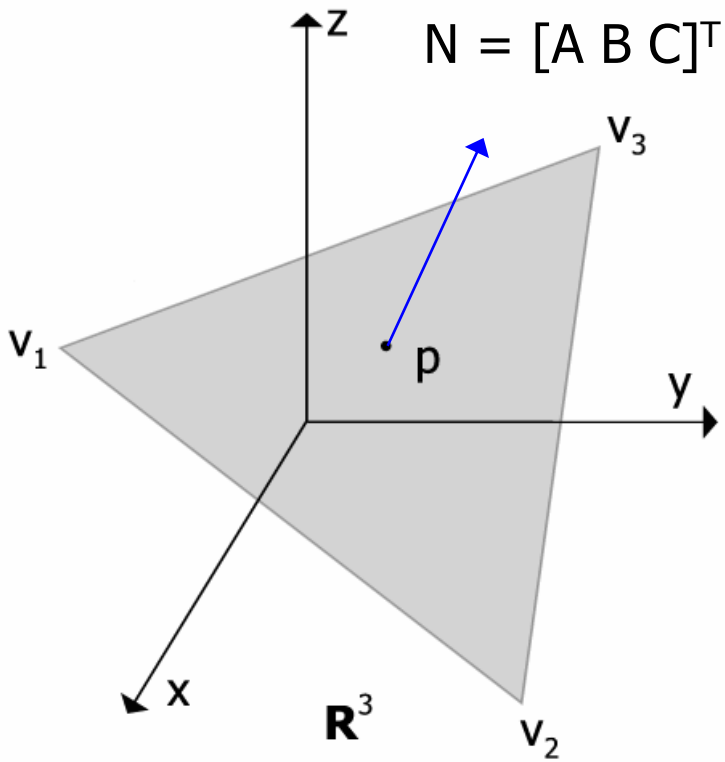
งานกลุ่ม (2-4 คน) ค้นคว้าทำรายงาน **การประมาณ** สมการระนาบเมื่อกำหนดจุดยอดมากกว่าหรือเท่ากับ 4 (DOF < จำนวนสมการ) เช่น การใช้เทคนิค Least Mean Squares, Covariance Approximation, ฯลฯ

Exact Plane Solution

สำหรับกรณีที่ DOF เท่ากับ จำนวนจุดที่กำหนดให้ เราสามารถหาสมการแม่นตรงสำหรับคำนวณสัมประสิทธิ์ค่าคงที่ของระนาบได้โดยกระจาย determinant ดังนี้

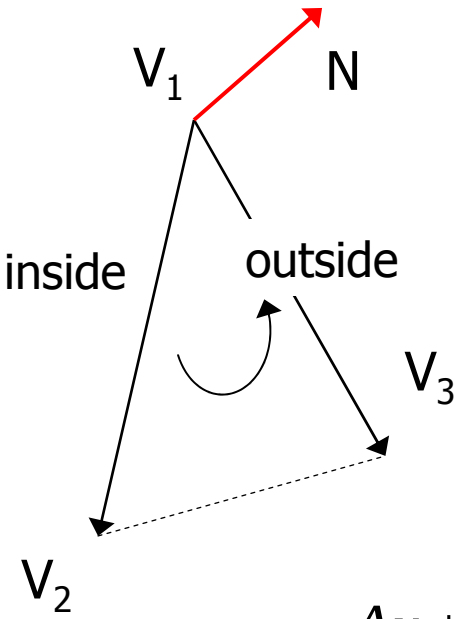
$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$
$$B = z_1(x_2 - x_3) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$
$$D = -x_1(y_2 z_3 - y_3 z_2) - x_2(y_3 z_1 - y_1 z_3) - x_3(y_1 z_2 - y_2 z_1)$$

ดังนั้นการวางตัวของระนาบในปริภูมิ จึงนิยามได้ด้วย **เวคเตอร์** ซึ่งตั้งฉากกับระนาบ (normal vector) กำหนดด้วยสัมประสิทธิ์ A, B, C ที่ปรากฏในสมการระนาบนั่นเอง



Interior and Exterior Regions

เนื่องจากการแสดงผล Polygon เป็นแบบ B-Rep (Boundary Representation) ซึ่งแบ่งแยกบริเวณ ***ภายใน*** (ส่วนที่มองไม่เห็น) และ ***ภายนอก*** (ส่วนที่มองเห็น) วัตถุ ถ้าระบุจุดยอดของระนาบในทิศทาง 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

$D = -\mathbf{N} \cdot \mathbf{P}$ ≡ **ระยะทางจากจุด P ไปยังระนาบ**

เมื่อทราบ [A, B, C] = **N** และ D แล้วเราสามารถหาว่าจุด P = (x, y, z) อยู่ภายในหรือภายนอกระนาบได้จาก

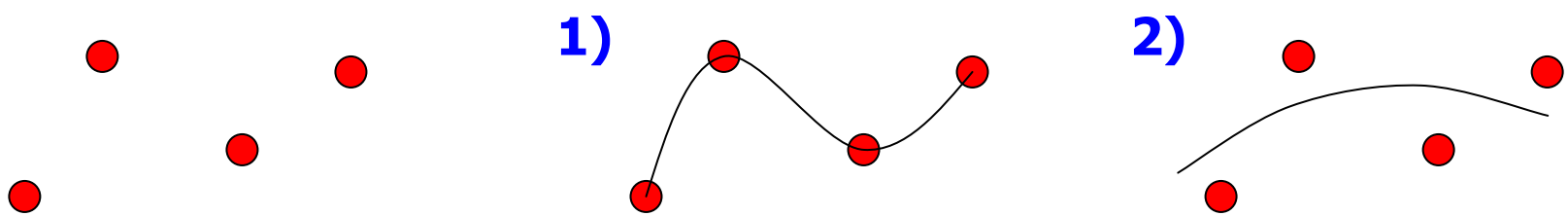
$$Ax + By + Cz + D \begin{cases} < 0 & (x, y, z) \text{ is inside} \\ > 0 & (x, y, z) \text{ is outside} \end{cases}$$

Curved Line and Surfaces

เราทราบแล้วว่า การวาดเส้นตรง บนระนาบสามารถทำได้โดยกำหนดจุดปลาย 2 จุด แล้ว โปรแกรมคอมพิวเตอร์ จะทำหน้าที่วาดจุดภาพระหว่าง จุดปลายดังกล่าว เพื่อสร้างภาพ (ประมาณ) ของเส้นตรง

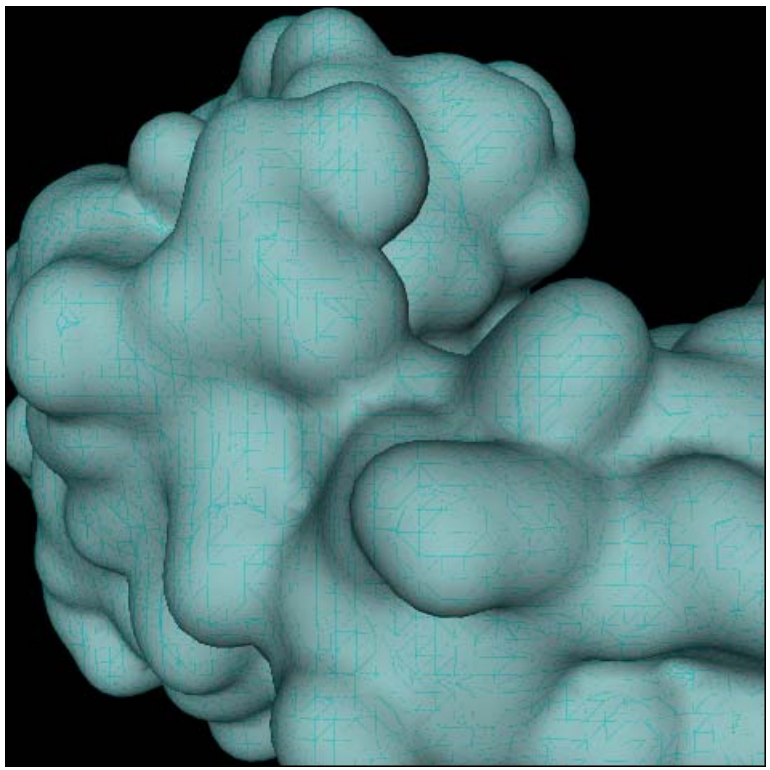
สำหรับเส้นโค้ง สามารถสร้างได้โดยกำหนดจุดเริ่มต้น จุดปลาย และ สมการของเส้นโค้งนั้นๆ อย่างไรก็ดี เส้นโค้งในธรรมชาติ มีหลายรูปแบบ การหาสมการเฉพาะสำหรับเส้นโค้งแต่ละแบบนั้นไม่สะดวก จึงไม่เป็นที่นิยม

ในทางปฏิบัติ โปรแกรม computer graphics จะให้ผู้ใช้กำหนดจุดข้อมูล (Data Points) ซึ่ง **1)** อยู่บน หรือ **2)** อยู่ใกล้ เส้นโค้ง ที่ต้องการ แล้ว ทำการลากเส้นโค้งให้ผ่านจุดที่กำหนด (กรณีแรก) หรือ ให้มีความแตกต่างเฉลี่ยระหว่างเส้นโค้งกับจุดที่กำหนด น้อยที่สุด (กรณีที่สอง)



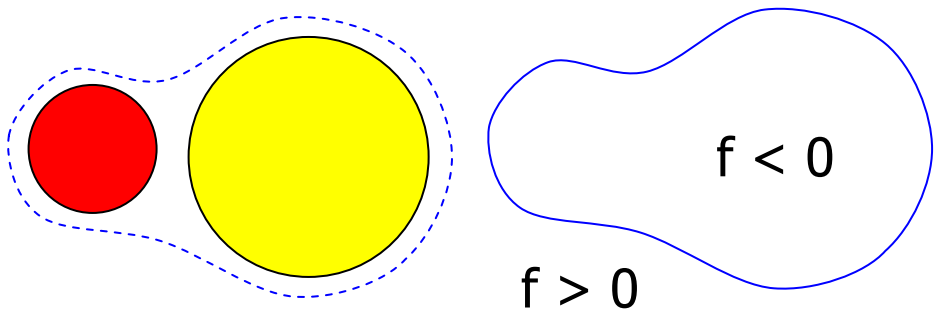
Blobby Objects

การแสดงภาพของพื้นผิวโค้ง รูปแบบหนึ่ง เหมาะสำหรับการวาด โครงสร้างของหยดของเหลว โครงสร้างโมเลกุล และ กล้ามเนื้อมนุษย์ โดยนิยามโดยสมการ implicit



พื้นผิวด้านบนสร้างได้โดยหารากของสมการ implicit **f (x, y, z) = 0**

$$f(x,y,z)=\sum_k b_k e^{-a_k r_k^2} - T = 0$$



โดยที่ $r = (x^2 + y^2 + z^2)^{1/2}$ กำหนดรัศมีของทรงกลมย่อย a กำหนดระยะทางยังผล และ b กำหนด ค่าถ่วงน้ำหนัก ของแต่ทรงกลม และ T กำหนดขนาดของวัตถุโดยรวม

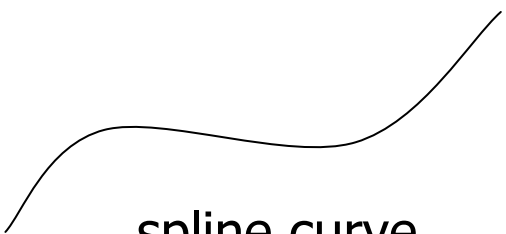
Spline Representations

Spline นิยามมาจากคำศัพท์ในวิชา เขียนแบบ (Engineering Drawing) ได้แก่ แถบโค้งยืดหยุ่น ซึ่งใช้สำหรับ **วาดเส้นโค้งผ่านจุดที่กำหนด**

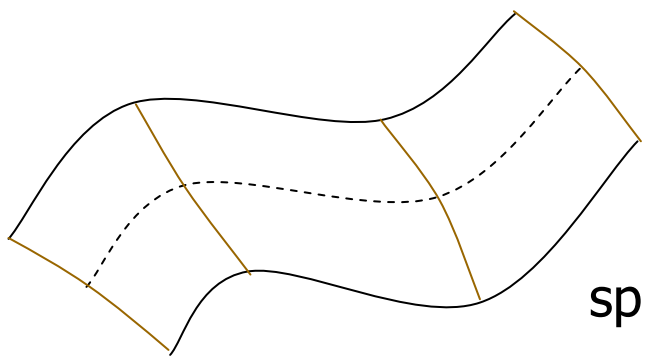
สำหรับในทางคณิตศาสตร์ Spline นิยามด้วย ฟังก์ชันพหุนามกำลังสาม ที่แบ่งออกเป็นช่วง (piecewise cubic polynomial function) ดังนั้น

- spline curve คือ เส้นโค้งที่ได้จากการรวมเส้นโค้งพหุนามหลายๆ เส้นด้วยกัน โดยที่ต้องสอดคล้องกับเงื่อนไข ความต่อเนื่อง (continuity conditions) ระหว่างเส้น

- spline surface คือ พื้นผิว ที่ได้จากการรวม spline curve สองชุดที่ตั้งฉากกัน



spline curve



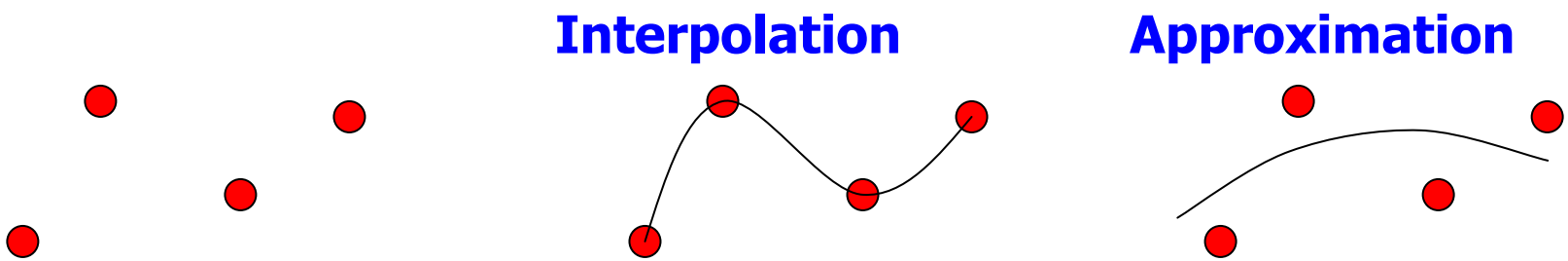
spline surface

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



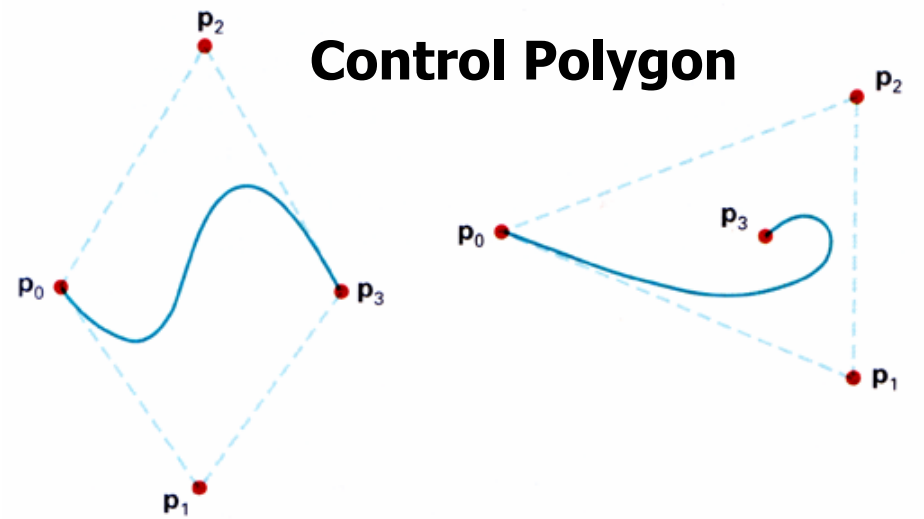
Convex Hull

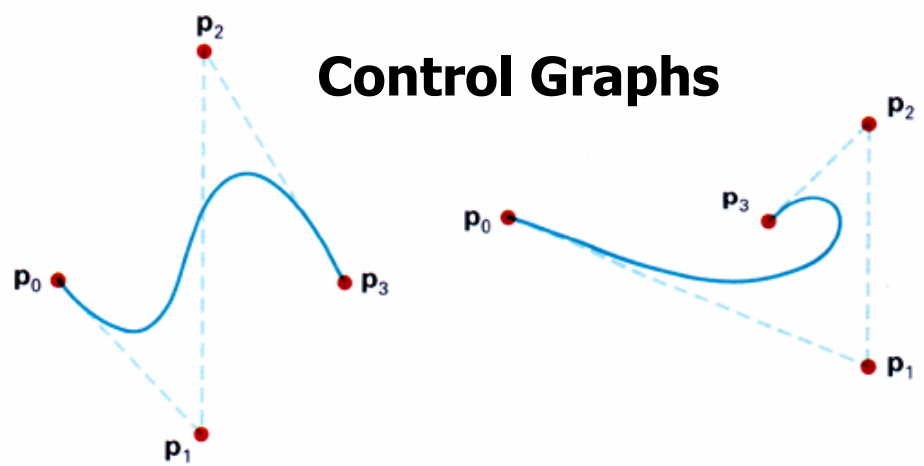
Convex Hull คือ ขอบเขตของ polygon ที่ล้อมรอบกลุ่มของ control points โดยที่สำหรับ control point ใดๆ เป็นไปได้เพียง 2 กรณี คือ อยู่บนเส้นขอบของ polygon หรือ อยู่ภายใน polygon

ด้วยเหตุนี้ เราเรียก

ลำดับของ Control Points ว่า **Control Graphs** และ

และเรียก Convex Hull ว่า **Control Polygon**





Parametric Continuity

เนื่องจากเส้นโค้งใดๆ เกิดจากการเชื่อมต่อกันของ ฟังก์ชันพหุนามของเส้นโค้ง ดังนั้น เพื่อให้รอยต่อมีความต่อเนื่อง ต้องกำหนดเงื่อนไขความต่อเนื่อง ดังนี้

กำหนดให้เส้นโค้ง spline กำหนดด้วยพิกัด Cartesian ซึ่งแต่ละองค์ประกอบเป็นฟังก์ชัน 1 ตัวแปร (u)

$$\mathbf{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T \rightarrow x = x(u), \quad y = y(u), \quad z = z(u)$$

ดังนั้น เงื่อนไขความต่อเนื่องนิยามโดย พิจารณาให้ค่าอนุพันธ์ขององค์ประกอบ x, y, z เทียบกับตัวแปร u **ที่ขอบรอยต่อร่วม** 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″) ที่รอยต่อเส้นโค้งย่อย มีรูปร่างคล้ายกัน ใช้สำหรับการกำหนดเส้นทางการเคลื่อนไหว

Geometric Continuity

กำหนดให้ $\mathbf{P} = \begin{bmatrix} x & y & z \end{bmatrix}^T \rightarrow x = x(u), \quad y = y(u), \quad z = z(u)$

ดังนั้นอนุพันธ์อันดับ 1 และ 2 $\mathbf{P}' = \begin{bmatrix} x' & y' & z' \end{bmatrix}^T \quad \mathbf{P}'' = \begin{bmatrix} x'' & y'' & z'' \end{bmatrix}^T$

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

- Zero-order (C^0) ทิศทางอนุพันธ์อันดับ 0 เท่ากัน (ค่า x, y, z ที่จุดรอยต่อเท่ากัน) เหมือนกับ Parametric Continuity
- First-order (C^1) อนุพันธ์อันดับ 1 **และ** 0 *มีทิศทางเดียวกัน* (เส้นความชัน tangent line หรือ x′, y′, z′ ที่รอยต่อขนานกัน)
- Second-order (C^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 $x(u) = a_x u^3 + b_x u^2 + c_x u + d_x \quad 0 \le u \le 1$

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

$$x(0) = d_x$$
$$x(1) = a_x + b_x + c_x + d_x$$
$$x'(0) = c_x$$
$$x'(1) = 3a_x + 2b_x + c_x$$

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

Matrix Specifications

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

ถ้ากำหนด ตัวแปร a, b, c, d ให้ (อาจหาได้จากวิธีที่ 1) เราสามารถเขียนสมการของเส้นโค้งในรูป matrix ได้ดังนี้

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} \quad \rightarrow \quad \mathbf{U} \cdot \mathbf{C}$$

โดยที่ U คือ matrix กำลังของตัวแปร u และ C คือ matrix ค่าคงที่ (a, b, c, d)

วิธีที่ 3 กำหนดเงื่อนไขของเส้นโค้งผ่านทาง matrix **M** ดังรายละเอียดกล่าวถึงในหัวข้อถัดไป ผลลัพธ์ที่ได้คือ matrix **C**

$$\mathbf{C} = \mathbf{M}_{spline(4\times4)} \cdot \mathbf{M}_{geom(4\times1)}$$

$$x(u) = \mathbf{U} \cdot \mathbf{M}_{spline} \cdot \mathbf{M}_{geom}$$

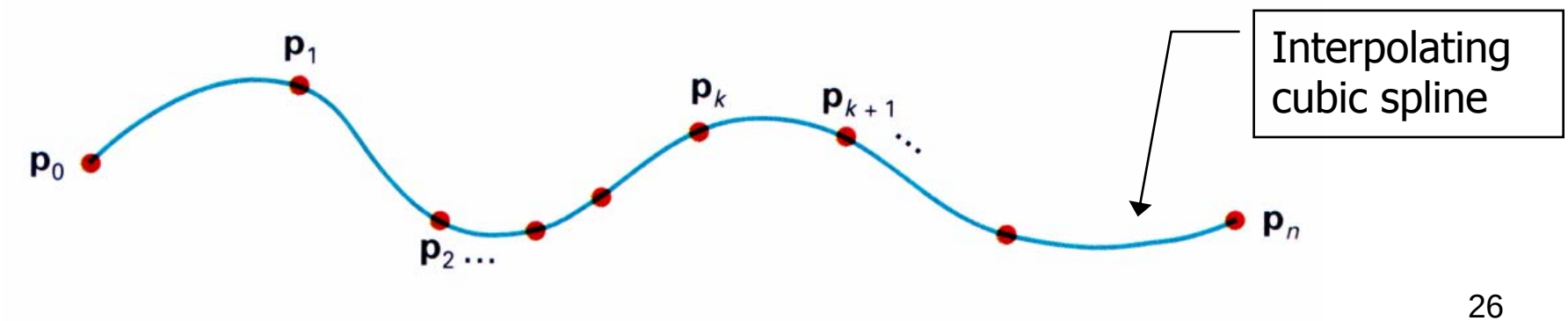
Cubic Spline Interpolation

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

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



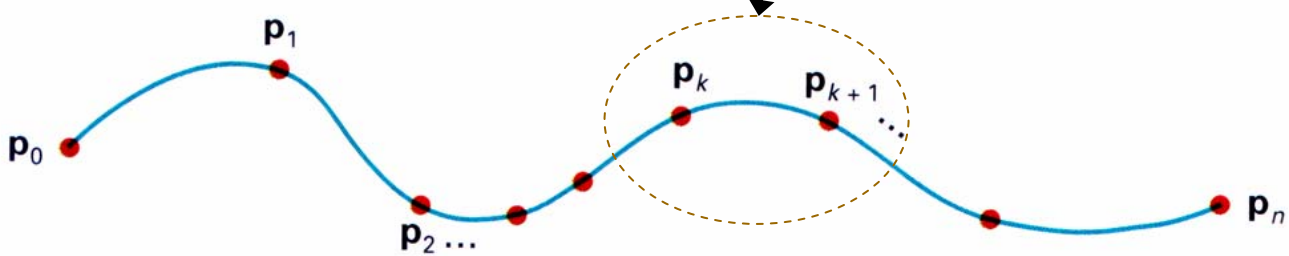
Solving Spline Coefficient

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

$$x(u) = a_x u^3 + b_x u^2 + c_x u + d_x$$
$$y(u) = a_y u^3 + b_y u^2 + c_y u + d_y \quad 0 \le u \le 1$$
$$z(u) = a_z u^3 + b_z u^2 + c_z u + d_z$$

สำหรับแต่ละสมการเราต้องการเงื่อนไขที่รอยต่อ 4 เงื่อนไข ซึ่งจะได้อธิบายต่อไป

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



Natural Cubic Spline

พิจารณาเฉพาะพิกัด x ในระบบ Cartesian ซึ่งเป็นฟังก์ชันของ u

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

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 $x'(0) = c_x$

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

อนุพันธ์อันดับ 2 $x''(0) = 2b_x$

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

อนุพันธ์อันดับ 0 $x(0) = d_x$

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

กำหนดให้ $x''(p_0) = x''(p_n) = 0$ เพื่อให้มี 4 สมการเพื่อแก้ 4 ตัวแปร

An Example

ถ้ากำหนด control points 3 จุด จะมีทั้งหมด (3-1) x 4 = 8 สมการ

$$\begin{bmatrix} x_0'(1) = x_1'(0) \\ x_0''(1) = x_1''(0) \\ x_0''(0) = 0 \\ x_1''(1) = 0 \\ x_0(0) = p_0 \\ x_1(1) = p_2 \\ x_0(1) = p_1 \\ x_1(0) = p_1 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 & 2 & 1 & 0 & 0 & 0 & -1 & 0 \\ 6 & 2 & 0 & 0 & 0 & -2 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6 & 2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_x^0 \\ b_x^0 \\ c_x^0 \\ d_x^0 \\ a_x^1 \\ b_x^1 \\ c_x^1 \\ d_x^1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ p_0 \\ p_2 \\ p_1 \\ p_1 \end{bmatrix}$$

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 **P** (u) แทนฟังก์ชันพหุนามกำลัง 3 สำหรับเส้นโค้งช่วงที่อยู่ระหว่าง control point p_k กับ p_{k+1}

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

แล้วนิยามเงื่อนไขขอบเขต ดังนี้

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

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

โดยที่ D แทน ตัวดำเนินการอนุพันธ์ อันดับที่ 1

Hermite Spline Matrix

จากเงื่อนไขของ Hermite เราสามารถเขียนในรูปของ Matrix ได้ดังนี้

0th order

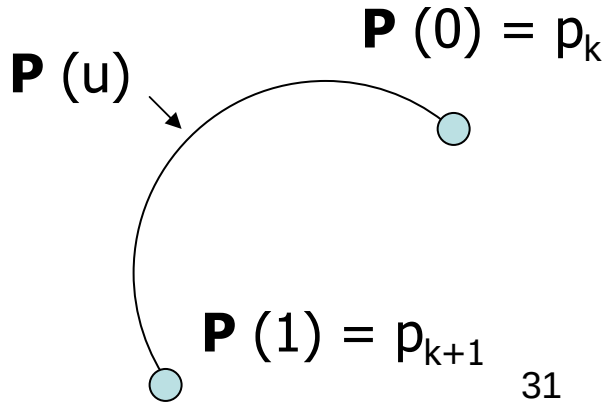
$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

1st order

$$\mathbf{P}'(u) = \begin{bmatrix} 3u^2 & 2u & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

แทนค่าตัวแปร u = 0 และ 1 แล้วหาสมการตามเงื่อนไขของ Hermite Spline ได้

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



Solving for Coefficients

จากสมการ Matrix ของ Hermite Spline เราสามารถแก้หาเวคเตอร์ สัมประสิทธิ์ **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 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

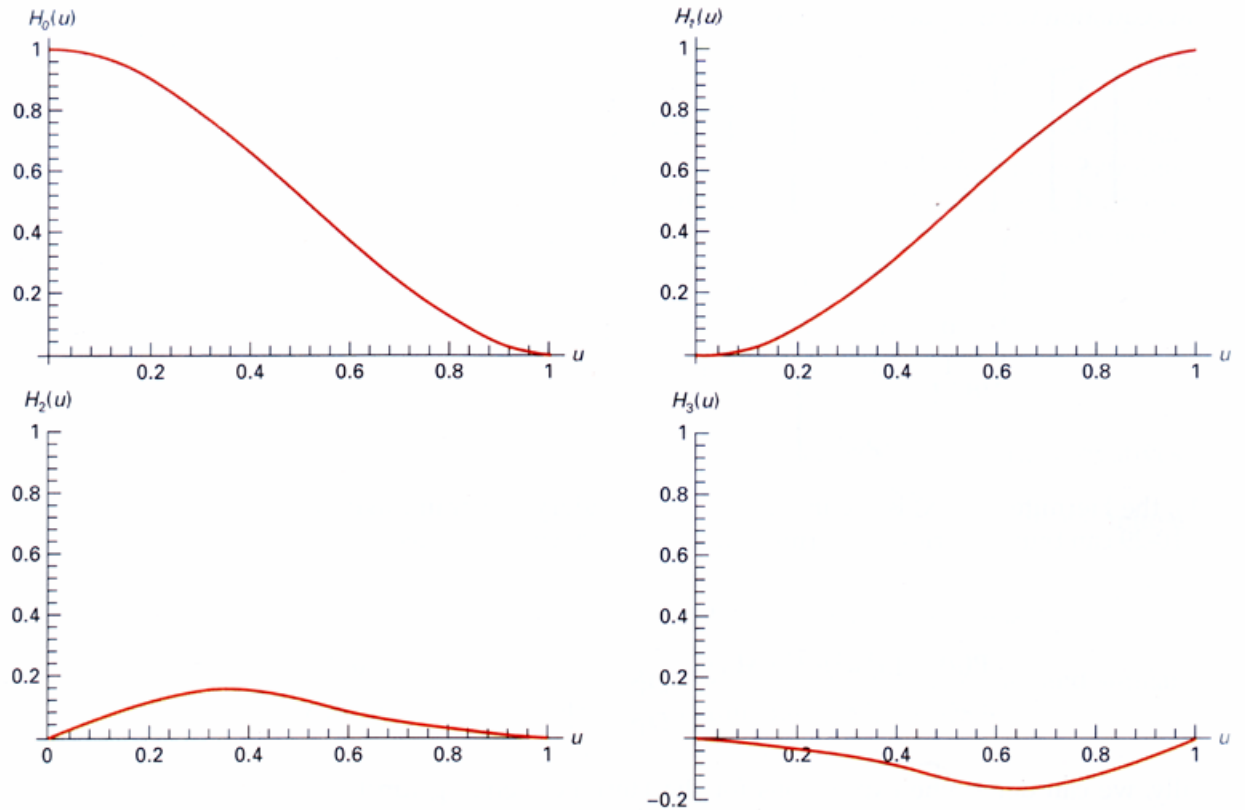
$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} u^3 & u^2 & u^1 & 1 \end{bmatrix} \cdot \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{Dp}_k \\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

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

จากสมการพบว่าผู้ใช้ต้องกำหนด ค่าความชัน **Dp** ที่ control points ด้วย

Hermite Spline as a Blending Function

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



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

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++