บทที่ 2

ทฤษฎีและแนวความคิดที่นำมาใช้ในการวิจัย

เจัยในอดีต

(7)

แก้สันจาก 2525 เป็น 2535

ายของฟอนต์

เมายของฟอนต์ในทางคอมพิวเตอร์นั้น คือการแทนรูปสัญลักษณ์ใด ๆ เตอร์ เช่น แอสกี (ASCII) หรือเอบซีดิก (EBCDIC) ซึ่งฟอนต์ที่ งตัวอักษรภาษาต่าง ๆ รวมทั้งสัญลักษณ์พิเศษที่มีใช้กันในภาษานั้น ๆ แต่จะต้องใช้สำหรับแทนภาษามนุษย์เท่านั้น เนื่องจากฟอนต์อาจใช้ ในบางรหัสโดยปะปนอยู่กับฟอนต์ตัวอักษร หรืออาจสร้างฟอนต์พิเศษสำ ๆ โดยเฉพาะก็ได้

ระเภทของฟอนต์

การสร้างฟอนต์ (font) ในทางคอมพิวเตอร์นั้น เราอาจแบ่งได้เป็น 2 วิธี 986) คือ

2.1.2.1 การสร้างในแบบราสเตอร์เซลล์ (raster cell)

จะเป็นการกำหนดตัวอักษรขึ้นจากตารางของจุด หรือที่เรียกกัน bitmap font) มีข้อดีคือ สามารถสร้างได้สะดวก และมีการแสดงผลที่รวด ยคือ การขยายฟอนต์ทำได้แต่การขยายให้ใหญ่ขึ้นและเป็นจำนวนเท่าของต้นแบบ กนี้ฟอนต์ที่ขยายขึ้นจะมีรายละเอียดที่หยาบ อีกทั้งการหมุนหรือการขยายฟอนต์ก็ทำ ประสิทธิภาพ

2.1.2.2 การสร้างในแบบเวกเตอร์ (vector)

จะเป็นการกำหนดตัวอักษรจากชุดของเส้น ซึ่งอยู่ในรูปแบบสม ศาสตร์ โดยบางครั้งเราจะเรียกฟอนต์แบบนี้ว่าสโตรกฟอนต์ (stroke font) ซึ่ง

มีข้อดีคือความยืดหยุ่นในการขยายขนาด หมุนหรือโย้ฟอนต์ได้อย่างอิสระ โดยที่ยังคงคุณภาพความสวยงามของฟอนต์ไว้ได้ แต่มีข้อเสียคือความยากลำบากในการสร้างและมีการแสดงผลที่ช้ากว่าแบบราสเตอร์เซลล์ อนึ่งเราสามารถแยกวิธีการสร้างฟอนต์ในแบบเวกเตอร์ได้เป็น 3 วิธี คือ

ก. การแทนที่ฟอนต์โดยประกอบส่วนต่าง ๆ ขึ้นจากสมการทางคณิตศาสตร์ เช่น ประกอบขึ้นจาก ฟังก์ชัน sin, cos, วงกลม เส้นตรง ตลอดจนสมการของรูปต่างๆ ซึ่งมีข้อเสียคือ ในการออกแบบฟอนต์แต่ละตัวมีความยุ่งยากเป็นอย่างมาก เพราะผู้ออกแบบต้องแยกฟอนต์แต่ละตัวเป็นส่วน ๆ และหาสมการที่เหมาะสมเพื่อนำมาแทนที่ฟอนต์นั้น ๆ

ข. การแทนที่ฟอนต์ด้วยเส้นตรงย่อย ๆ คือการนำเส้นตรงเล็ก ๆ จำนวนหนึ่ง ซึ่งมีขนาดและมุมต่างๆกันมาประกอบขึ้นเป็นฟอนต์แต่ละตัว เช่น การสร้างสโตรคฟอนต์ของเทอร์โบซี ซึ่งจะสร้างตัวอักษรแต่ละตัวด้วยการวาดชุดของเส้นตรง (line sequence) (BEN EZZELL 1989) ซึ่งวิธีนี้มีข้อเสียคือ การแทนที่ส่วนที่มีความโค้งมน จะปรากฏเป็นเหลี่ยมมุมได้เมื่อมีการขยายฟอนต์ให้ใหญ่ขึ้น

ค. การแทนที่ฟอนต์ด้วยเส้นโค้ง (curve) เป็นวิธีที่นิยมใช้กันอยู่ในปัจจุบัน เนื่องจากมีข้อดีคือสามารถแทนฟอนต์ในแบบต่าง ๆ ได้เหมือนจริง อีกทั้งผู้ออกแบบฟอนต์ไม่จำเป็นต้องมีความรู้ทางคณิตศาสตร์ในการสร้างฟอนต์ต่าง ๆ ขึ้นมา ส่วนข้อเสียคือการเสียเวลาในการจัดเส้นโค้ง (curve fitting) เพื่อทำการสร้างฟอนต์ให้ได้รูปแบบตามที่ต้องการ

2.1.3 <u>ทฤษฎีการแทนฟอนต์ด้วยวิธีเวกเตอร์</u>

ลักษณะการสร้างภาพโดยทั่วไป จะเน้นถึงวิธีการนำเส้นและสีต่างๆ มาประกอบกันเป็นรูป ซึ่งลักษณะของเส้นที่ใช้อยู่ อาจแบ่งได้เป็นสองประเภทใหญ่ๆ ทั้งนี้โดยไม่คำนึงถึงความเข้มหรือการต่อเนื่องและขาดตอนของเส้นที่ใช้ คือ เส้นตรง(straight lines) และเส้นโค้ง (curved lines) (สัญธาน ธีรนรวนิชย์ 2526) อนึ่ง ในการสร้างฟอนต์ เราอาจพิจารณาได้ว่าฟอนต์แต่ละตัวก็คือภาพนั่นเอง นั่นคือเราสามารถแทนฟอนต์แต่ละตัวได้โดยประกอบกันขึ้นจากเส้นตรงและเส้นโค้ง โดยเส้นแต่ละเส้นเราสามารถแทนอยู่ในรูปของสมการทางคณิตศาสตร์วิธีใดวิธีหนึ่ง ดังนั้นเราจึงสามารถสร้างฟอนต์สัญลักษณ์ใด ๆ ด้วยวิธีเวกเตอร์ได้เสมอ

อนึ่ง ในที่นี้เส้นที่ใช้ในการสร้างภาพจะต้องอยู่ในรูปแบบของเส้นบาง เพื่อความสะดวกในการวิเคราะห์ โดยในกรณีที่เป็นภาพที่มีความหนา เราจะทำการวิเคราะห์ให้อยู่ในแบบเส้นขอบ (outline) ก่อนที่จะนำมาแทนเป็นเวกเตอร์ฟอนต์ต่อไป และเพื่อความสวยงามของฟอนต์สัญลักษณ์ที่ได้ การแทนฟอนต์สัญลักษณ์จะใช้การแทนที่ด้วยเส้นโค้ง (curve) อันจะทำให้สามารถแทนภาพสัญลักษณ์ที่ประกอบด้วยส่วนโค้งมนต่าง ๆ ได้อย่างสวยงามและมีประสิทธิภาพ และยังสามารถแทนที่เส้นตรงได้ด้วย เนื่องจากเส้นตรงก็คือกรณีเฉพาะกรณีหนึ่งของเส้นโค้งนั่นเอง

2.1.4 <u>รูปแบบการแทนที่เส้นโค้ง (curve) ด้วยสมการทางคณิตศาสตร์</u>

การแทนที่เส้นต่าง ๆ ในทางคณิตศาสตร์ รวมทั้งเส้นโค้ง เราสามารถทำได้หลายรูปแบบด้วยกัน (EDWARD ANGEL 1990) คือ

2.1.4.1 <u>รูปแบบเอกพลิซิท (explicit form)</u> จะเป็นรูปแบบทางคณิตศาสตร์ที่เป็นที่คุ้นเคยมากที่สุด คือ

ก. $y = f(x)$ เช่น $y = mx + c$
ข. $x = f(y)$ เช่น $x = sqrt(r^2 - y^2)$

ซึ่งจาก ก. x จะเป็นตัวแปรอิสระ และ y จะเป็นตัวแปรไม่อิสระที่ขึ้นอยู่กับ x ในขณะที่จาก ข. y จะเป็นตัวแปรอิสระ และ x จะเป็นตัวแปรไม่อิสระที่ขึ้นอยู่กับ y อันจะทำให้รูปแบบนี้เกิดปัญหาความขึ้นกับแกน (axis dependent) และยังอาจเกิดกรณีหารด้วย 0 หรือการถอดรากค่าลบขึ้น ทำให้ต้องมีการแยกกรณีเฉพาะซึ่งไม่เหมาะกับการใช้งานโดยคอมพิวเตอร์

2.1.4.2 <u>รูปแบบอิมพลิซิท (implicit form)</u> จะอยู่ในรูปแบบ

$$g(x,y) = 0$$

เช่น สมการวงกลม คือ

$$x^2 + y^2 = r^2$$

การแทนที่ในรูปแบบนี้ จะสามารถหลีกเลี่ยงปัญหาที่ปรากฏในรูปแบบเอกพลิชิทได้ ซึ่งรูปแบบนี้จะเป็นประโยชน์ในการทดสอบ ว่าจุดที่ต้องการอยู่บนเส้นโค้งที่กำหนดหรือไม่ มากกว่าที่จะใช้ในการหาจุดซึ่งอยู่บนเส้นตรง เนื่องจากในกรณีทั่วไปเราไม่สามารถหาจุด (x, y) ที่อยู่บนเส้นโค้งได้ ยกเว้นแต่จะใช้วิธีการเชิงตัวเลข (numerical method) (Edward Angel 1990) ซึ่งช้าเกินกว่าจะนำมาใช้ในการแสดงผลแบบทันที (real-time) บนหน่วยแสดงผลกราฟฟิกในทางคอมพิวเตอร์ได้

2.1.4.3 <u>รูปแบบพาราเมตริก (parametric form)</u> จะเป็นการแทนที่ด้วยสมการ 2 สมการซึ่งเป็นแบบเอกพลิชิท โดยมีตัวแปรอิสระ คือ t ดังนี้

$$x = x(t)$$
$$y = y(t)$$

เช่น สมการวงกลมเราสามารถแทนได้ด้วย

$$x = r \cos t$$
$$y = r \sin t$$

ซึ่งเราสามารถพิสูจน์ความถูกต้องของสมการได้ โดยการกำจัดตัวแปร t ออกจากสมการ อันจะทำให้เราได้สมการวงกลมในรูปแบบอิมพลิชิทหรือเอกพลิชิท

รูปแบบพาราเมตริกจะเป็นรูปแบบที่มีการประยุกต์ใช้มากที่สุด ในการแทนที่เส้นโค้ง (curve) และพื้นผิว (surface) ทางคอมพิวเตอร์กราฟฟิก เนื่องจากตัวแปรแต่ละตัวจะขึ้นอยู่กับสมการของตัวเอง ทำให้รูปแบบนี้สามารถใช้ได้ทั่วไปและเป็นอิสระจากแกน (axis independent) และการที่แยกสมการยังทำให้ง่ายต่อขบวนการคำนวณจุดเพื่อแสดงผลในทางคอมพิวเตอร์กราฟฟิก (scan conversion algorithm) อีกด้วย

2.1.5 <u>เส้นโค้งแบบเบซิเยร์</u>

เส้นโค้งแบบเบซิเยร์ได้ถูกสร้างขึ้นโดยนักคณิตศาสตร์ชาวฝรั่งเศส คือ ปิแอร์ เบซิเยร์ (Pierre Bezier) เพื่อใช้ในงานแบบจำลองโดยคอมพิวเตอร์ภายในบริษัทรถยนต์ของฝรั่งเศส คือเรโนลส์ (Renault) โดยนำมาใช้ในระบบซอฟท์แวร์ชื่อ ยูนิเซิร์ฟ (UNISURF) ที่ใช้ในการออกแบบโครงสร้างของรถยนต์ (IBRAHIM ZEID 1991)

เส้นโค้งแบบเบซิเยร์ จะเป็นเส้นโค้งที่ถูกกำหนดรูปร่างโดยจุดควบคุม (control point) จำนวน n+1 จุด เมื่อ n เป็นดีกรีของเส้นโค้งเบซิเยร์ และมีสมการในการคำนวณเส้นโค้งจากจุดควบคุม คือ

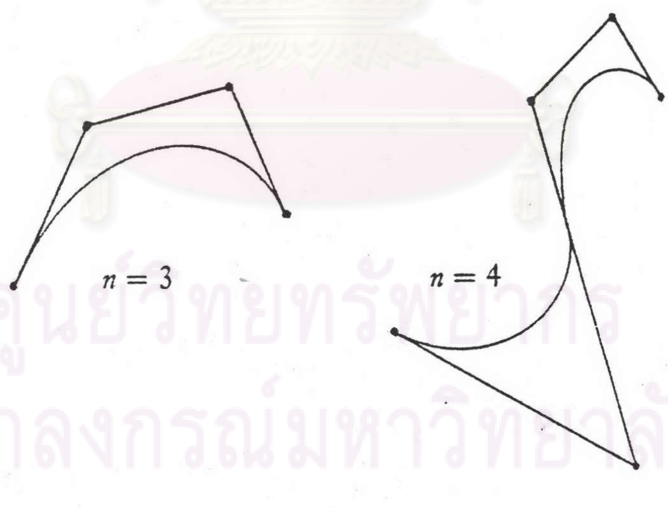
$$p(u) = \sum_{i=0}^{n} p_i B_{i,n}(u) \qquad u \in [0,1] \qquad (2.1)$$

เมื่อ p(u) คือจุดใด ๆ บนเส้นโค้ง p_i คือจุดควบคุม และมีฟังก์ชันประกอบ (blending function) คือ

$$B_{i,n}(u) = {}^{n}C_i u^i (1-u)^{n-i} \qquad (2.2)$$

โดยที่

$${}^{n}C_i = \frac{n!}{i!(n-i)!} \qquad (2.3)$$



รูปที่ 2.1 ตัวอย่างของเส้นโค้งแบบเบซิเยร์

2.1.5.1 นิยามที่เกี่ยวข้อง

นิยาม 2.1.5.1.1 เซตคอนเวกซ์ (convex set)

เซตคอนเวกซ์ของจุด ศือกลุ่มของจุดซึ่งเส้นใด ๆ ที่เชื่อมระหว่างจุดคู่หนึ่งภายในเซตของจุดนี้ยังคงอยู่ในเซต

นิยาม 2.1.5.1.2 คอนเวกซ์ฮัลของกลุ่มของจุด (convex hull of set of point)

เมื่อมีกลุ่มของจุดอยู่กลุ่มหนึ่ง คอนเวกซ์ฮัล คือ เซตคอนเวกซ์ที่เล็กที่สุด ที่สามารถครอบคลุมกลุ่มของจุดนั้น

2.1.5.2 คุณสมบัติของเส้นโค้งแบบเบซิเยร์

เส้นโค้งแบบเบซิเยร์ มีคุณสมบัติที่สำคัญสามารถสรุปได้ดังนี้

ก. เส้นโค้งจะต้องลากผ่านจุดควบคุมจุดแรก และจุดสุดท้ายเสมอ นั่นคือจะต้องผ่านจุด p_0 และ p_n ในสมการที่ 2.1 นั่นเอง

ข. เส้นโค้งแบบเบซิเยร์จะมีคุณสมบัติคอนเวกซ์ฮัลเสมอ นั่นคือ เส้นโค้งจะจำกัดอยู่ภายในพื้นที่ไม่เกินคอนเวกซ์ฮัลของจุดควบคุมของเส้นโค้งนั้น ซึ่งเราสามารถสร้างรูปเหลี่ยมควบคุม (control polygon) โดยลากเส้นตรงเชื่อมระหว่างจุดควบคุมต่าง ๆ เพื่อทราบขอบเขตที่เส้นโค้งจะจำกัดอยู่ภายใน (FRANCIS S. HILL, JR. 1990)

ค. เส้นโค้งจะสัมผัสกับรูปเหลี่ยมควบคุม ณ จุดควบคุมเริ่มต้นและจุดควบคุมสิ้นสุดเสมอ และจากสมการที่ 2.1 และ 2.2 เราจะได้ว่า อนุพันธ์ที่ 1 ณ จุดเริ่มต้นและจุดสิ้นสุด คือ

$$p'(0) = n(p_1 - p_0)$$
$$p'(1) = n(p_n - p_{n-1})$$

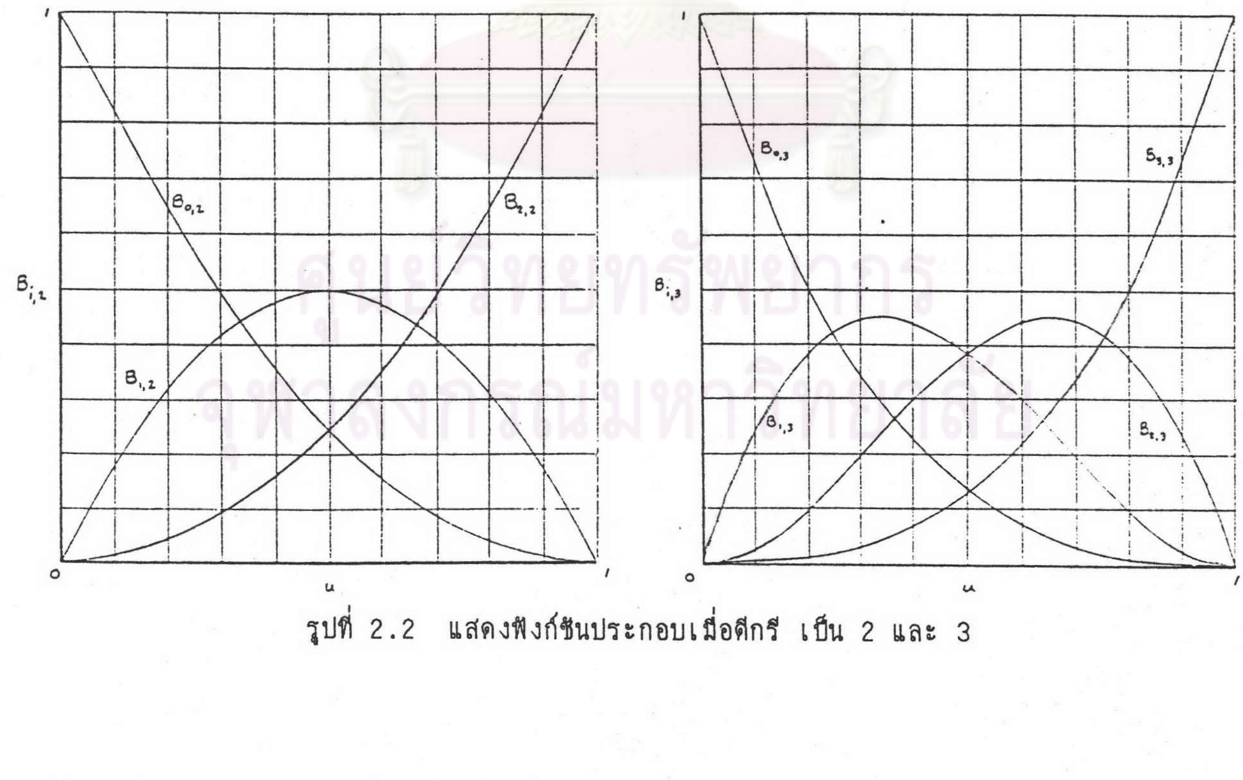
ซึ่งก็คือ เราสามารถหาความชันของเส้นโค้ง ณ จุดเริ่มต้นและจุดสิ้นสุดได้จากจุดควบคุม

ง. เส้นโค้งจะมีความสมมาตรระหว่าง u และ (u-1) นั่นคือ ลำดับของจุดควบคุมของเส้นโค้งสามารถย้อนกลับได้ โดยไม่มีการเปลี่ยนแปลงรูปร่างของเส้นโค้ง (IBRAHIM ZEID 1991)

จ. รูปร่างของเส้นโค้งสามารถเปลี่ยนแปลงได้ โดยการเปลี่ยนตำแหน่งของจุดควบคุมใด ๆ

ฉ. เส้นโค้งแบบเบซิเยร์มีคุณสมบัติในการแปลง (transform) รูป คือ เปลี่ยนขนาด เปลี่ยนองศา และเปลี่ยนตำแหน่งได้ตามลำดับ และมีคุณสมบัติที่สำคัญคือ เราสามารถทำการคำนวณเพื่อทำการแปลงรูป เฉพาะกับจุดควบคุม แล้วจึงทำการสร้างเส้นโค้งขึ้นใหม่จากสมการคำนวณเส้นโค้ง (FRANCIS S. HILL, JR. 1990) ทำให้การแปลงรูปสามารถทำได้อย่างสะดวกรวดเร็ว

ช. ฟังก์ชันประกอบ (blending function) ตามสมการ 2.2 จะมีค่า ${}^{n}C_{i}u^{i}(1-u)^{n-i}$ สูงสุดที่ $u = i/n$ (สามารถหาได้จาก $d(B_{i,n})/du = 0$) นั่นคือ จุดควบคุมแต่ละจุดจะมีผลกระทบต่อรูปร่างของเส้นโค้งมากที่สุดที่จุด $u = i/n$ โดยเราสามารถแสดงฟังก์ชันประกอบเมื่อดีกรีต่าง ๆ ได้ตามรูป



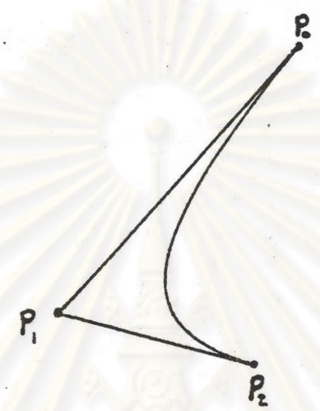
รูปที่ 2.2 แสดงฟังก์ชันประกอบเมื่อดีกรี เป็น 2 และ 3

2.1.5.3 <u>เส้นโค้งแบบเบซิเยร์ดีกรี 2</u>

เส้นโค้งแบบเบซิเยร์เมื่อดีกรีเป็น 2 จะประกอบด้วยจุดควบคุม 3 จุดและมีสมการในการคำนวณเส้นโค้ง คือ

$$p(u) = (1-u)^2 p_0 + 2u(1-u)p_1 + u^2 p_2$$

เมื่อ $u \quad [0,1]$



รูปที่ 2.3 รูปตัวอย่างของเส้นโค้งแบบเบซิเยร์เมื่อดีกรีเป็น 2

2.2 <u>เทคนิคการแปลงฟอนต์แบบจุดภาพเป็นฟอนต์แบบเวกเตอร์</u>

ในงานวิจัยนี้ เราจะทำการแปลงฟอนต์แบบจุดภาพไปเป็นฟอนต์แบบเวกเตอร์โดยอัตโนมัติ โดยฟอนต์แบบเวกเตอร์ที่ได้ จะเป็นฟอนต์ซึ่งเกิดจากเส้นโค้ง เพื่อให้มีความสวยงามและเหมาะสมในการพัฒนาเพื่อประยุกต์ใช้กับภาษาไทยต่อไป

2.2.1 <u>เทคนิคการแทนฟอนต์สัญลักษณ์ด้วยเส้นโค้ง</u>

เราสามารถแทนฟอนต์สัญลักษณ์ใด ๆ ด้วยเส้นโค้ง โดยการตัดแบ่งฟอนต์สัญลักษณ์นั้นเป็นส่วนย่อย ๆ ที่สามารถแทนด้วยเส้นโค้งพื้นฐานได้ (เส้นตรงก็เป็นเส้นโค้งชนิดหนึ่ง) โดยเราจะต้องคำนึงถึงความต่อเนื่องของเส้นโค้งย่อยในแต่ละจุดเชื่อมต่อด้วย

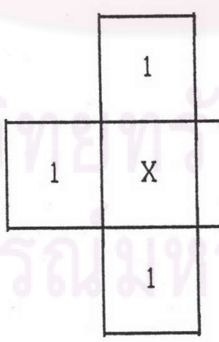
เส้นโค้งที่เลือกใช้ในงานวิจัยนี้ คือเส้นโค้งแบบเบซิเยร์ดีกรี 2 เนื่องจากเส้นโค้งแบบเบซิเยร์ในดีกรีนี้ มีข้อดีคือ เป็นเส้นโค้งแบบเบซิเยร์ ซึ่งสามารถสร้างเส้นโค้งได้โดยมีการคำนวณเลขยกกำลังที่น้อยที่สุด ทำให้มีความเหมาะสมด้านความเร็วในการใช้งานทางคอมพิวเตอร์กราฟฟิก แต่มีข้อเสียคือ จะมีความจำกัดของรูปร่างเส้นโค้งที่สร้างไม่สามารถซับซ้อนมาก ซึ่งเราสามารถแก้ปัญหาได้โดยการตัดแบ่งเส้นโค้งเป็นเส้นโค้งย่อย ๆ ที่สามารถสร้างได้โดยเส้นโค้งเบซิเยร์ดีกรี 2

ในการตัดแบ่งฟอนต์สัญลักษณ์ออกเป็นส่วนโค้งย่อย ๆ เราสามารถเชื่อมต่อส่วนโค้งย่อย ๆ ซึ่งสร้างจากเส้นโค้งเบซิเยร์เหล่านั้น โดยให้จุดควบคุมแรกของเส้นโค้งถัดไปเป็นจุดเดียวกับจุดควบคุมสุดท้ายของเส้นโค้งปัจจุบัน และจากคุณสมบัติของเส้นโค้งแบบเบซิเยร์ข้อ 2.1.5.2.ค ทำให้เราทราบว่าเราสามารถหาความชันของจุดเริ่มต้นและจุดสิ้นสุดของเส้นโค้งได้จากจุดควบคุม ซึ่งก็คือเราสามารถหาความชัน ณ จุดเชื่อมต่อของเส้นโค้งย่อยได้นั่นเอง ทำให้เราสามารถทำให้เส้นโค้งย่อยที่เชื่อมต่อกันมีความราบเรียบ (smooth) ได้ โดยให้ความชัน ณ จุดสิ้นสุดของเส้นโค้งหน้าเท่ากับความชัน ณ จุดเริ่มต้นของเส้นโค้งหลัง

2.2.2 <u>เทคนิคการแปลงฟอนต์แบบจุดภาพเป็นฟอนต์แบบจุดภาพเส้นขอบ</u>

การแปลงฟอนต์แบบจุดภาพทึบเป็นฟอนต์แบบจุดภาพเส้นขอบ เราสามารถทำได้โดยการกำจัดจุดภาพที่อยู่ในพื้นที่ส่วนทึบของฟอนต์สัญลักษณ์นั้น ให้เหลือแต่จุดภาพที่เป็นเส้นขอบ โดยเราจะสามารถตัดสินว่าจุดภาพจุดใดเป็นจุดภาพที่เป็นเส้นขอบได้ ดังนี้

กำหนดให้จุดภาพที่เราจะตรวจสอบคือ จุด X



รูปที่ 2.4 ช่องตารางแสดงตำแหน่งที่ใช้ระบุว่าจุด X เป็นเส้นขอบหรือไม่

ถ้ามีจุดภาพอื่น ๆ ปรากฏอยู่รอบจุด X ทั้ง 4 จุดตามตำแหน่งที่เป็นเลข 1 ในรูป เราจะตัดสินว่าจุด X เป็นจุดภาพภายในของฟอนต์สัญลักษณ์ นั่นก็คือไม่ใช่จุดภาพที่เป็นเส้นขอบ ส่วนกรณีอื่น ๆ จะถือว่าเป็นจุดภาพเส้นขอบทั้งหมด

2.2.3 <u>เทคนิคการตามรอย (trace) ฟอนต์แบบจุดภาพเส้นขอบ</u>

ในการตามรอย (trace) ฟอนต์แบบจุดภาพเส้นขอบ เพื่อหาว่าจุดภาพถัดไปของเส้นขอบคือจุดภาพใด เราสามารถทำได้โดย

ก. ในการหาจุดภาพเริ่มต้นของฟอนต์แบบจุดภาพเส้นขอบ เราจะทำการหาได้โดยการตรวจหา (scan) จุดภาพจุดแรก ซึ่งจะทำการตรวจจากส่วนซ้ายไปขวาและจากส่วนบนลงส่วนล่างของฟอนต์สัญลักษณ์นั้น

ข. เมื่อได้จุดภาพเริ่มต้นของเส้นขอบแล้ว ก็จะใช้จุดภาพนี้สำหรับเป็นจุดเริ่มต้นในการหาจุดภาพต่อไป ที่จะมาประกอบกันเป็นเส้นขอบของฟอนต์สัญลักษณ์ ดังนี้

ให้ P_i เป็นจุดภาพ ณ ตำแหน่งปัจจุบัน
P_{i-1} เป็นจุดภาพก่อนหน้าจุดปัจจุบัน
C_n เป็นจุดรอบ ๆ P_i ที่จะตรวจสอบหาจุดภาพถัดไป
เราจะตรวจสอบจุด C_n ในลักษณะตามเข็มนาฬิกา ดังรูป

C_0	C_1	C_2
C_7	P_i	C_3
C_6	C_5	C_4

รูปที่ 2.5 ช่องตารางแสดงตำแหน่งของจุดรอบจุดปัจจุบันที่จะตรวจสอบตามเข็มนาฬิกา

จากรูปที่ 2.5 หากจุดภาพ P_{i-1} ปรากฏอยู่ ณ ตำแหน่ง C_1 เราจะทำการตรวจจากจุด C_2, C_3, .., C_7, C_0 วนตามเข็มนาฬิกาจนมาบรรจบกับจุด

P_{i-1} อีกครั้งหากพบจุดภาพ ณ ตำแหน่งใดก็แสดงว่าจุดภาพนั้นเป็นจุดภาพถัดไปของเส้นขอบ แต่หากไม่พบจุดภาพใด ๆ แสดงว่าจุดภาพ P_i คือจุดปลายของเส้นขอบแล้ว

ในกรณีที่จุด P_{i-1} ปรากฏอยู่ ณ ตำแหน่ง C_n อื่น ๆ ก็จะตรวจสอบจนครบวงรอบเช่นเดียวกัน ดังรูปที่ 2.6 ซึ่งแสดงกรณีที่ P_{i-1} ปรากฏอยู่ ณ ตำแหน่ง C_0 ดังนั้นเราจะทำการตรวจจากจุด C_1, C_2, .., C_7

	P_{i-1}	C_1	C_2	
	C_7	P_i	C_3	
	C_6	C_5	C_4	

รูปที่ 2.6 รูปแสดงจุดรอบ ๆ จุดภาพ P_i ซึ่งจะถูกตรวจสอบตามลำดับถัดจากจุด P_{i-1}

2.2.4 <u>เทคนิคการหาทิศทางที่แท้จริงของจุดภาพเส้นขอบ</u>

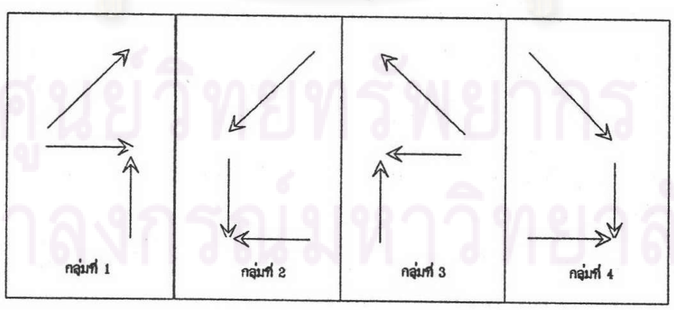
เนื่องจากในทางคอมพิวเตอร์กราฟฟิกแบบจุดภาพ เส้นขอบที่ได้จะประกอบขึ้นจากจุดที่ไม่ต่อเนื่องกัน ดังนั้นทิศทางจากจุดหนึ่งไปยังอีกจุดหนึ่งอาจไม่ใช่ทิศทางจริงของเส้นขอบก็ได้ ดังรูปที่ 2.7 ซึ่งเราจะเห็นได้ว่าทิศทางรวมจาก 1 ไปยัง 5 คือ จากซ้าย-บน ไปขวา-ล่าง แต่ในระหว่างจุด 1 และจุด 5 จะมีทิศทาง ณ จุดที่ 4 เป็น ทิศจากซ้าย ดังนั้น ในขั้นตอนนี้เราจะต้องทำการตีความให้จุดที่ 4 มีทิศทางเป็นจากซ้าย-บนเช่นเดียวกับจุดอื่น ๆ เพื่อให้การตัดแบ่งเส้นโค้งไม่เกิดการผิดพลาด ทำการตัดยังจุดที่ไม่ใช่จุดเปลี่ยนทิศทางจริง ๆ

1				
	2			
		3	4	
				5

รูปที่ 2.7 รูปแสดงข้อจำกัดของการวาดจุดที่ไม่ต่อเนื่องเป็นเส้นตรง

เนื่องจากในทิศทางตรงทั้ง 4 ทิศ คือ ซ้าย ขวา บน ล่าง ไม่มีข้อจำกัดในการวาดจุดแบบไม่ต่อเนื่องอยู่แล้ว ในการตรวจสอบการเปลี่ยนทิศทางที่แท้จริงของจุดภาพเส้นขอบ เราจึงจะกระทำกับเฉพาะจุดซึ่งมีทิศทางในแนวเฉียงทั้ง 4 ทิศ คือจุดในทิศจากซ้าย-บน จากขวา-ล่าง จากขวา-บน และจากซ้าย-ล่าง เท่านั้น โดยเราจะทำการเปลี่ยนจุดซึ่งอยู่ถัดจากจุดเริ่มต้นที่เป็นทิศทางเฉียง และมีทิศทางอยู่ในกลุ่มเดียวกับทิศทางเฉียงนั้นให้เป็นทิศทางเดียวกัน จนกว่าจะพบจุดซึ่งไม่ใช่จุดในกลุ่มเดียวกัน

ในรูปที่ 2.8 จะแสดงทิศทางเฉียงและทิศทางซึ่งจัดอยู่ในกลุ่มเดียวกัน

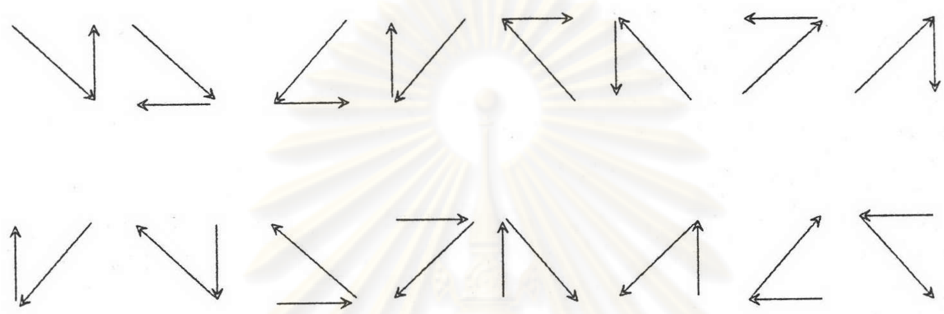


รูปที่ 2.8 แสดงทิศทางซึ่งจัดอยู่ในกลุ่มเดียวกันทั้ง 4 กลุ่ม

2.2.5 <u>เทคนิคการตัดแบ่งเส้นขอบเป็นเส้นโค้งย่อย</u>

เราสามารถตัดแบ่งเส้นขอบ (outline) ออกเป็นเส้นโค้งย่อย (curve segment) ได้ โดยตัด ณ จุดเปลี่ยนแปลงทิศทางของเส้นขอบ ภายใต้เงื่อนไขดังนี้

ก. จะทำการตัดแบ่งเป็นเส้นโค้งย่อย เมื่อจุดที่มีการเปลี่ยนแปลงทิศทางนั้นมีรูปแบบการเปลี่ยนทิศทางที่ตรงกับรูปแบบใดรูปแบบหนึ่งใน 16 รูปแบบ ดังรูปที่ 2.9



รูปที่ 2.9 แสดงจุดเปลี่ยนทิศทางที่ตรงตามเงื่อนไขการตัดเส้นขอบเป็นเส้นโค้งย่อย

ข. ในกรณีที่จุดเปลี่ยนทิศทางไม่ตรงกับ 16 รูปแบบข้างต้น เราจะถือว่าจุดนั้นเป็นเพียงความโค้งของเส้นโค้งเท่านั้น ยังไม่ใช่จุดตัดของเส้นโค้งย่อย แต่จะทำการนับจุดนี้ไว้ และเมื่อเกิดจุดเปลี่ยนแปลงทิศทางขึ้นอีกครั้งจะถือว่าจุดนั้นเป็นจุดตัดเป็นเส้นโค้งย่อยทันที ไม่ว่า ณ จุดเปลี่ยนแปลงทิศทางนั้นจะตรงกับ 16 รูปแบบข้างต้นหรือไม่ก็ตาม

อนึ่ง จากเงื่อนไขข้างต้น จะสังเกตเห็นได้ว่า กรณีที่เส้นขอบมีเส้นตรงปรากฏอยู่ เทคนิคที่ใช้ในการตัดเส้นขอบจะมองว่าเส้นตรงนั้นเป็นส่วนหนึ่งของเส้นโค้งเท่านั้น โดยจะไม่มีแยกแยะเส้นตรงภายในเส้นขอบเลย ในกรณีที่ต้องการแยกเส้นตรงออกจากเส้นโค้ง เราอาจใช้วิธีการนับจุดซึ่งมีทิศทางเดียวกันซ้ำ ๆ เป็นจำนวน n ครั้ง เพื่อตีความว่าเป็นเส้นตรงที่ต้องทำการตัด โดยค่า n เราจะต้องทำการประมาณค่าเองว่าควรจะมีค่าเท่าใด จึงจะตีความว่าเป็นเส้นตรง

2.2.6 <u>เทคนิคการแปลงเส้นโค้งย่อยเป็นเส้นโค้งแบบเบซิเยร์</u>

ในการแปลงเส้นโค้งย่อยซึ่งอยู่ในรูปแบบจุดภาพ ให้อยู่ในรูปแบบเวกเตอร์ ซึ่งในที่นี้ก็คือเส้นโค้งแบบเบซิเยร์ดีกรี 2 นั้น เราสามารถทำได้โดยการวิเคราะห์จุดปรากฏซึ่งประกอบกันขึ้นเป็นเส้นโค้งนั้น เพื่อแปลงย้อนกลับเป็นจุดควบคุมทั้ง 3 จุดของเส้นโค้งแบบเบซิเยร์ดีกรี 2 ดังนี้

ก. เนื่องจากเส้นโค้งแบบเบซิเยร์มีคุณสมบัติตามข้อ 2.1.5.2.ก คือเส้นโค้งจะผ่านจุดควบคุมจุดแรกและจุดควบคุมจุดสุดท้ายเสมอ ประกอบกับมีคุณสมบัติคอนเวกซ์ฮัลตามข้อ 2.1.5.2.ข ซึ่งทำให้เส้นโค้งจะจำกัดอยู่ภายในขอบเขตของรูปเหลี่ยมควบคุม ดังนั้นเราจะได้ว่าจุดควบคุมที่ 1 และ 3 ของเส้นโค้งย่อยนี้ คือจุดเริ่มต้นและจุดสิ้นสุดของเส้นโค้งย่อย

ข. สำหรับจุดควบคุมที่เหลือคือ จุดที่ 2 เราสามารถหาได้จากการแปลงสมการเบซิเยร์ดีกรี 2 ย้อนกลับ ดังนี้

จากสมการเบซิเยร์ดีกรี 2

$$p(u) = (1-u)^2 p_0 + 2u(1-u)p_1 + u^2 p_2$$

เมื่อ u $[0,1]$

p_0, p_1, p_2 คือจุดควบคุมที่ 1 2 และ 3 ตามลำดับ

$p(u)$ คือจุดบนเส้นโค้งเบซิเยร์ ณ ตำแหน่ง u

ดังนั้น เราจะได้ว่า

$$p_1 = (\, p(u) - (1-u)^2 p_0 - u^2 p_2 \,) \,/\, 2u(1-u)$$

โดยที่เราจะต้องทำการกำหนดค่า u เป็นค่าคงที่ค่าหนึ่ง และประมาณว่าจุดใดบนเส้นโค้งย่อยที่ตรงกับค่า u ค่านี้ จากนั้นจึงทำการคำนวณค่า p_1 ออกมา แต่เนื่องจาก $p(u)$ เกิดจากการประมาณค่า ทำให้อาจเกิดความผิดพลาดได้ เราจึงต้องทำการตรวจสอบว่าจุดควบคุมที่ได้จะสร้างเส้นโค้งที่ตรงกับเส้นโค้งย่อยเดิมหรือไม่ หากมีความผิดพลาด ความผิดพลาดนั้นอยู่ในช่วงที่ยอมรับได้หรือไม่ หากไม่อาจยอมรับได้ ก็จะต้องทำการประมาณค่าใหม่เพื่อหาค่าที่เหมาะสมที่สุดต่อไป และหากทำการประมาณค่าจนจุดในเส้นโค้งย่อยจนหมดก็ยังไม่สามารถหาจุด P_1 ซึ่งเหมาะสมได้ เราจะต้องทำการตัดแบ่งเส้นโค้งย่อยให้เล็กลง และทำการประมาณค่าใหม่ โดยการตัดแบ่งอาจทำได้หลายวิธี ในที่นี้จะใช้วิธีที่ง่ายที่สุดคือการแบ่งครึ่ง