บทที่ 3

ทฤษฎีและแนวความคิดทางด้านคอมพิวเตอร์กราฟิก

ทฤษฎีเส้นโค้งเบเซีย

เส้นโค้งเบเซีย(Bezier curve) เป็นเส้นโค้งที่ถูกคิดค้นขึ้นโดยนักคณิตศาสตร์ชาวฝรั่งเศส Pierre Bezier ในปี 1972(Newman, 1989; Rankin, 1989; Rogers, 1990; Sproull and Sutherland, 1989) เพื่อนำมาใช้กับการออกแบบรถยนต์ของบริษัทเรโนลท์ โดยการกำหนดความโค้งของเส้นด้วยจุดควบคุมความโค้ง ซึ่งสามารถเขียนในรูปของฟังก์ชันพหุนามได้ดังนี้

$$P(u) = \sum_{i=0}^{n} p_i B_{i,n}(u)$$

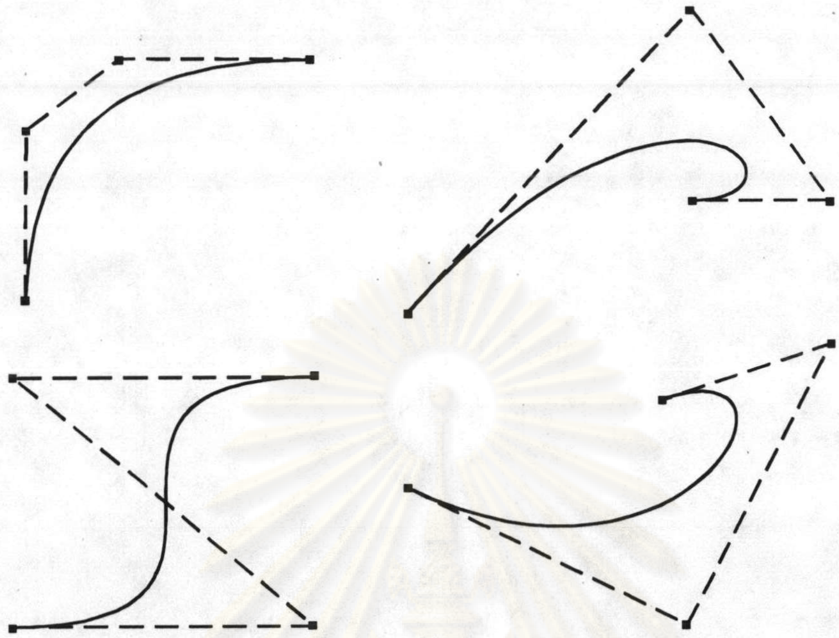
โดยที่ p_i คือจุดควบคุมความโค้งและ n คือจำนวนของจุดควบคุมความโค้ง และเรียก n - 1 ว่าดีกรีของเส้นโค้ง $B_{i,n}(u)$ คือฟังก์ชันเบลนดิง(Blending)หรือเรียกอีกชื่อหนึ่งว่าฟังก์ชันเบิร์นสไตน์(Bernstein) ที่ถูกกำหนดโดย $B_{i,n}(u) = C(n, i)\, u^{i}(1-u)^{n-i}$; $C(n, i) = n! / (i!(n-i)!)$ และ $0 \le u \le 1$

นั่นคือค่าของคู่ลำดับ x, y ของเส้นโค้งเบเซียจะสามารถคำนวณได้จาก

$$x(u) = \sum_{i=0}^{n} p_{xi} B_{i,n}(u)$$ และ

$$y(u) = \sum_{i=0}^{n} p_{yi} B_{i,n}(u)$$

สำหรับจำนวนของจุดควบคุมความโค้งที่จะใช้กำหนดเส้นโค้งเบเซียนั้น ขึ้นอยู่กับความต้องการใช้งาน แต่สำหรับการนำมาใช้กับการบรรยายเส้นโค้งของแบบอักษรโพสต์สคริปต์นั้นได้กำหนดจำนวนจุดควบคุมความโค้งไว้เท่ากับ 4 จุด(ดูรูปที่ 3.1) ซึ่งมีชื่อเรียกเส้นโค้งเบเซียที่มีจุดควบคุมความโค้ง 4 จุดว่า "คิวบิกเคอร์ฟ" (cubic curve) เนื่องจากเป็นเส้นโค้งดีกรี 3

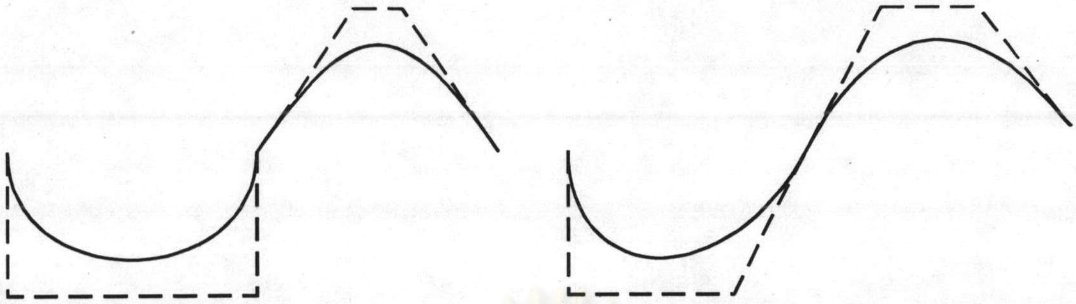


รูปที่ 3.1 แสดงลักษณะของเส้นโค้งเบเชียที่มีจุดควบคุมความโค้ง 4 จุด

โดยทั่วไปลักษณะสำคัญของเส้นโค้งเบเชียจะโค้งเข้าหาจุดควบคุมความโค้งแต่จะไม่สัมผัส(แต่กรณีที่จุดควบคุมความโค้งอยู่ในแนวเดียวกัน เส้นโค้งเบเชียจะเป็นเส้นตรงเดียวกับจุดควบคุมความโค้ง) ยกเว้นแต่จุดควบคุมความโค้งแรกและจุดควบคุมความโค้งสุดท้ายเท่านั้น นอกจากนี้เส้นโค้งจะมีความสมมาตร กล่าวคือลำดับของจุดควบคุมความโค้งของเส้นโค้งสามารถกำหนดแบบย้อนกลับได้โดยไม่มีการเปลี่ยนแปลงรูปร่างของเส้นโค้ง การเชื่อมต่อเส้นโค้งสองเส้นเข้าด้วยกันนั้นจะทำได้ 2 ลักษณะ (ดูรูปที่ 3.2) ดังนี้

1. Zero-order continuity เป็นการเชื่อมต่อเส้นโค้งสองเส้นเข้าด้วยกัน โดยจุดเชื่อมต่อจะไม่มีการปรับจุดควบคุมความโค้ง ดังนั้นการเชื่อมต่อลักษณะนี้เส้นโค้งทั้งสองจะยังคงลักษณะเดิมเสมอ

2. First-order continuity เป็นการเชื่อมต่อเส้นโค้งสองเส้นเข้าด้วยกันโดยมีการปรับจุดควบคุมความโค้งเพื่อทำให้เส้นโค้งทั้งสองต่อเนื่องกัน และช่วงเชื่อมต่อกลมกลืนเป็นแนวเดียวกัน โดยลักษณะของเส้นโค้งอาจเปลี่ยนแปลงไป



รูปที่ 3.2 แสดงลักษณะการเชื่อมต่อเส้นโค้งเบเซีย

<u>การแปลงรูป</u>

ภาพในระบบคอมพิวเตอร์กราฟิกส่วนใหญ่ จะถูกสร้างขึ้นจากส่วนของเส้นตรงหลายๆเส้นมาประกอบเข้าด้วยกัน ซึ่งเส้นเหล่านั้นถูกกำหนดโดยจุดปลายของเส้น ดังนั้นการที่จะเปลี่ยนแปลงลักษณะของภาพจึงสามารถทำได้โดยง่าย โดยการคำนวณทางคณิตศาสตร์กับจุดปลายของเส้นเหล่านั้น(Harrington, 1987; Rogers, 1990)

วิธีการคำนวณที่ใช้ในการแปลงรูปคือ การแทนจุดต่างๆด้วยเมทริกซ์ขนาด 1 x 2 และนำเมทริกซ์การแปลง (transformation matrix)ที่ต้องการมาคูณ ก็จะได้ผลลัพธ์เป็นเมทริกซ์ใหม่ ซึ่งคือจุดของปลายของเส้นตรงต่างๆที่เปลี่ยนแปลงไป

การแปลงรูปที่นำมาใช้ในการวิจัยนี้ ประกอบด้วย

1. การย่อหรือขยาย เป็นการเปลี่ยนแปลงขนาดของภาพ อาจจะทำให้ภาพมีขนาดใหญ่ขึ้นหรือเล็กลงได้ วิธีการย่อหรือขยายภาพทำได้โดยการคูณจุดต่างๆด้วยค่าการสเกลที่ต้องการ ก็จะได้จุดใหม่ที่ทำให้ภาพมีขนาดเปลี่ยนแปลงไป การเปลี่ยนแปลงขนาดของภาพสามารถเขียนแทนด้วยเมทริกซ์ดังนี้

สมมติให้จุด P1 = [x1 y1] เป็นเมทริกซ์ขนาด 1 x 2 และกำหนดให้ T เป็นเมทริกซ์ขนาด 2 x 2 ของค่าการเสกล

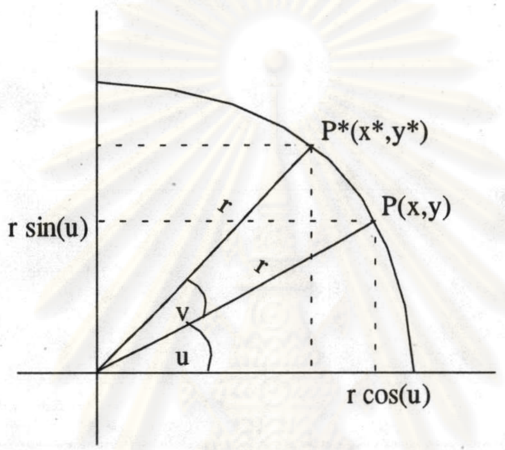
$$T = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$$

เมื่อ s_x คือค่าการสเกลในแนวแกน x
s_y คือค่าการสเกลในแนวแกน y

ถ้าคูณเมทริกซ์ P1 ด้วย T จะได้เมทริกซ์ใหม่(P2)ขนาด 1 x 2 ซึ่งเมทริกซ์นี้คือจุดใหม่หลังจากการย่อ-ขยายแล้วนั่นเอง

$P1 \times T = [s_x * x1 \quad s_y * y1]$

2. การหมุนรอบจุดกำเนิด เป็นการเปลี่ยนแปลงภาพโดยการหมุนจุดในทิศทางตามหรือทวนเข็มนาฬิกา โดยมีจุดศูนย์กลางของการหมุนอยู่ที่จุดกำเนิด ซึ่งมีวิธีดังนี้



รูปที่ 3.3 แสดงการหมุนจุด P ทวนเข็มนาฬิกา

จากรูปที่ 3.3 สมมติว่าทำการหมุนจุด P = [x y] ทวนเข็มนาฬิกาเป็นมุม b องศา

จุด p = [x y] = [rcos(u) rsin(u)]

และจุด P* = [x* y*] = [rcos(u+v) rsin(u+v)]

ดังนั้น P* = [x* y*] = [r(cos(u)cos(v)-sin(u)sin(v)) r(cos(u)sin(v)+sin(u)cos(v))]

แทนค่า rcos(u) ด้วย x และ rsin(u) ด้วย y จะได้

P* = [x* y*] = [xcos(v)-ysin(v) xsin(v)+ycos(v)]

ดังนั้นจุด P = [x y] หมุนไปเป็นจุด P* = [x* y*] โดย

x* = xcos(v) - ysin(v)

y* = xsin(v) + ycos(v)

หรือเขียนในรูปเมทริกซ์ได้ดังนี้

$$P^* = [x^* \; y^*] = [x \; y]\begin{vmatrix} \cos(v) & \sin(v) \\ -\sin(v) & \cos(v) \end{vmatrix}$$

3. การย้าย เป็นการเลื่อนภาพจากตำแหน่งหนึ่งไปอีกตำแหน่ง โดยขนาดและรูปร่างของภาพไม่มีการเปลี่ยนแปลง การย้ายจึงทำได้โดยง่ายโดยการบวกหรือลบจุดทุกจุดด้วยระยะทางที่ต้องการย้าย ดังนี้

$x^* = x + t_x$ และ $y^* = y + t_y$ เมื่อ t_x และ t_y คือระยะทางที่ต้องการเคลื่อนย้ายในแนวแกน x และ y

4. การโย้ (shear) เป็นการเปลี่ยนรูปทรงของภาพให้เกิดความลาดเอียง ซึ่งสามารถเปลี่ยนรูปทรงได้ทั้งในแนวแกน x และแกน y

การโย้ในแนวแกน x คือการเปลี่ยนค่า x ไปตามค่าของ y ในขณะที่คงค่า y ไว้ตามเดิม เป็นผลให้ภาพเกิดการเอียงซ้ายหรือขวา

การโย้ในแนวแกน y คือการเปลี่ยนค่า y ไปตามค่าของ x ในขณะที่คงค่า x ไว้ตามเดิม เป็นผลให้ภาพเกิดการเอียงขึ้นหรือลง

$$T = \begin{bmatrix} 1 & h_y \\ 0 & 1 \end{bmatrix}$$ เป็นเมทริกซ์การโย้ในแนวแกน y เมื่อ h_y เป็นค่าการโย้

$$T = \begin{bmatrix} 1 & 0 \\ h_x & 1 \end{bmatrix}$$ เป็นเมทริกซ์การโย้ในแนวแกน x เมื่อ h_x เป็นค่าการโย้

5. พิกัดเอกพันธ์(homogeneous coordinate) ความแตกต่างของวิธีการย้ายกับการย่อหรือขยาย หมุนและโย้ก็คือการย้ายจะไม่มีเมทริกซ์การแปลงมาคูณเพื่อให้ได้ตำแหน่งของจุดต่างๆใหม่ทำให้ไม่สามารถสร้างเมทริกซ์การแปลงเพียงเมทริกซ์เดียวเพื่อใช้กับการแปลงหลายๆอย่างได้ การแก้ไขปัญหานี้ก็คือการนำพิกัดเอกพันธ์เข้ามาช่วย โดยวิธีการนี้จะใช้เมทริกซ์ขนาด 3 x 3 แทนเมทริกซ์การแปลงเดิม ดังนี้

$$T = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

โดย a, b, c, d คือค่าของเมทริกซ์การแปลง(T)ขนาด 2 x 2 เดิม

t_x, t_y คือระยะทางที่ต้องการเคลื่อนย้ายในทิศทางแกน x และ y ตามลำดับ

และเมทริกซ์ของจุดที่จะทำการแปลงจะเพิ่มพิกัดอีกหนึ่งตัวคือ 1 ดังนี้ $P = [x \quad y \quad 1]$

5.1 เมทริกซ์การแปลงของการย่อ-ขยาย คือ

$$T = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

เมื่อ s_x และ s_y คือค่าการสเกล

5.2 เมทริกซ์การแปลงของการหมุนรอบจุดกำเนิด คือ

$$T = \begin{bmatrix} \cos(v) & \sin(v) & 0 \\ -\sin(v) & \cos(v) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

เมื่อ v คือมุมที่หมุนทวนเข็มนาฬิกา

5.3 เมทริกซ์การแปลงของการย้าย คือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

เมื่อ t_x และ t_y คือระยะทางที่ต้องการย้าย

5.4 เมทริกซ์การแปลงของการโย้ คือ

$$T = \begin{bmatrix} 1 & h_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

เมื่อ h_y คือค่าการโย้ในแนวแกน y

$$T = \begin{bmatrix} 1 & 0 & 0 \\ h_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

เมื่อ h_x คือค่าการโย้ในแนวแกน x

5.5 เมทริกซ์การแปลงของการหมุนรอบจุดใดใด

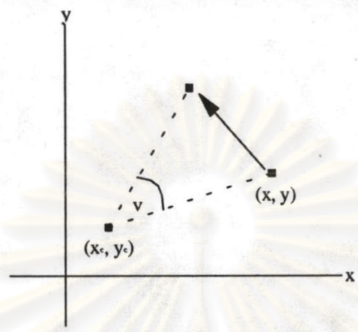
วิธีการแปลงแบบการหมุนรอบจุดใดใด(ดูรูปที่3.4)จะมีขั้นตอนเพิ่มขึ้นจากการหมุนรอบจุดกำเนิด ดังนี้

(1) ย้ายภาพเพื่อให้จุดศูนย์กลางของการหมุน (x_c,y_c) ไปอยู่ที่จุดกำเนิด

(2) หมุนรอบจุดกำเนิด

(3) ย้ายภาพเพื่อให้จุดศูนย์กลางของการหมุนกลับไปอยู่ที่ตำแหน่งเดิม

การแปลงทั้ง 3 ขั้นตอนนี้ จะต้องทำทีละขั้นตอนและตามลำดับ เพราะในการทำแต่ละขั้นตอนต้องใช้เมทริกซ์การแปลงมาคูณเข้าไป ซึ่งการคูณเมทริกซ์ไม่มีคุณสมบัติการสลับที่



รูปที่ 3.4 แสดงการหมุนรอบจุดใดใด

เมทริกซ์การแปลงที่จะย้ายจุด (x_c, y_c) ไปยังจุดกำเนิดคือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix}$$

เมทริกซ์การแปลงที่จะหมุนเป็นมุม v คือ

$$T = \begin{bmatrix} \cos(v) & \sin(v) & 0 \\ -\sin(v) & \cos(v) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

และเมทริกซ์การแปลงที่จะย้ายจุดไปยังจุดเดิมคือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

เมื่อนำเมทริกซ์ทั้ง 3 มาคูณเข้ากับจุดที่เราจะหมุน P(x,y) ได้จุดใหม่เป็น P1(x1,y1) ดังนั้นเมทริกซ์การแปลงแบบหมุนรอบจุด (x_c, y_c) คือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix} \begin{bmatrix} \cos(v) & \sin(v) & 0 \\ -\sin(v) & \cos(v) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

$$T = \begin{bmatrix} \cos(v) & \sin(v) & 0 \\ -\sin(v) & \cos(v) & 0 \\ -x_c\cos(v)+y_c\sin(v)+x_c & -x_c\sin(v)-y_c\sin(v)+y_c & 1 \end{bmatrix}$$

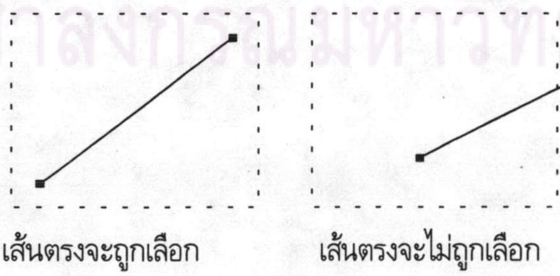
<u>การเลือกวัตถุ</u>

การเลือกวัตถุเป็นเครื่องมือสำคัญสิ่งหนึ่งในการทำงานของระบบคอมพิวเตอร์กราฟิกเพื่อให้ผู้ใช้สามารถเลือกส่วนใดส่วนหนึ่งหรือทั้งหมดของวัตถุ แล้วทำการเปลี่ยนแปลง เช่น ลบ ย้าย ย่อ-ขยาย หมุน หรือคัดลอกได้โดยง่าย แนวทางในการเลือกวัตถุที่ใช้กันในกราฟิกโปรแกรมต่างๆโดยทั่วไป มี 2 รูปแบบ คือ

- การเลือกโดยการลากกรอบสี่เหลี่ยมล้อมรอบวัตถุที่ต้องการ(Bounding Box selection)

- การเลือกโดยการใช้ตัวชี้ตำแหน่งชี้ไปยังวัตถุที่ต้องการ(Point selection)

1. การเลือกโดยการลากกรอบสี่เหลี่ยมล้อมรอบวัตถุที่ต้องการ(Newman, 1989) วิธีการเลือกแบบนี้ทำได้โดยง่าย กล่าวคือเป็นการหาว่าวัตถุใดที่อยู่ภายในกรอบสี่เหลี่ยมที่ลาก ก็จะเป็นวัตถุที่ถูกเลือก เช่น การเลือกเส้นตรงก็จะทดสอบจุดปลายทั้ง 2 จุดว่าอยู่ภายในกรอบสี่เหลี่ยมหรือไม่(ดูรูปที่ 3.5) ส่วนกรณีเส้นโค้งจะต้องทำการคำนวณหาชุดของเส้นตรงที่ใช้แทนเส้นโค้งก่อน แล้วจึงทำการทดสอบว่าทุกเส้นอยู่ภายในกรอบสี่เหลี่ยมหรือไม่

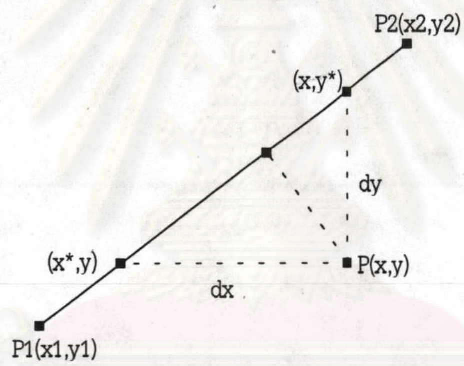


เส้นตรงจะถูกเลือก เส้นตรงจะไม่ถูกเลือก

รูปที่ 3.5 แสดงการเลือกวัตถุโดยการลากกรอบสี่เหลี่ยมล้อมรอบ

2. การเลือกโดยการใช้ตัวชี้ตำแหน่งชี้ไปยังวัตถุที่ต้องการ(Harrington, 1987; Newman, 1989) วิธีการเลือกแบบนี้จะมีความยุ่งยากกว่าวิธีแรก เพราะว่าจะต้องหาว่าวัตถุใดที่ตัวชี้ตำแหน่งชี้อยู่ ซึ่งสามารถทำได้หลายวิธี เช่น การหาระยะทางที่สั้นที่สุดจากตำแหน่งที่ตัวชี้ตำแหน่งชี้อยู่ไปยังวัตถุต่างๆ แต่การพิจารณาเลือกใช้จะต้องคำนึงถึงความยุ่งยากและเวลาที่ต้องใช้ในการคำนวณซึ่งจะต้องไม่นานมากเพื่อตอบสนองผู้ใช้ได้อย่างรวดเร็ว ซึ่งในการวิจัยนี้ได้เลือกใช้วิธีการเปรียบเทียบระยะทางตามแนวแกน x หรือ y กับค่าที่กำหนด ถ้าระยะทางในแนวแกนใดแกนหนึ่งมีค่าน้อยกว่าที่กำหนดก็จะเลือกวัตถุนั้น ทั้งนี้เนื่องจากการหาระยะทางตามแนวแกน x หรือ y จะทำได้ง่ายและรวดเร็วกว่าการหาระยะทางสั้นที่สุด(ซึ่งจะต้องหาระยะทางของเส้นตั้งฉากจากจุดไปยังเส้นที่ต้องการทดสอบซึ่งจะต้องทำการคำนวณที่สลับซับซ้อน) ดังนี้

จากรูปที่ 3.6 สมมติจุด P(x, y) เป็นจุดที่ตัวชี้ตำแหน่งชี้อยู่ โดย P1P2 เป็นเส้นตรงที่ลากจากจุด P1(x1, y1) ไป P2(x2, y2) การหาระยะทางจากจุด P ไปตามแกน x และ y ทำได้โดย



รูปที่ 3.6 แสดงการหาระยะทางจากจุด P ตามแนวแกน x และ y

กำหนดให้ dx และ dy คือระยะทางจากจุด P ไปยังเส้น P1P2 ตามแกน x และแกน y ตามลำดับ และให้ m คือความชันของเส้น P1P2

$$dx = x^* - x$$

เนื่องจาก $(y - y1) / (x^* - x1) = m$

$$x^* = (y - y1) / m + x1$$

ดังนั้น $dx = (y - y1) / m + (x1 - x)$

$$dy = y^* - y$$

เนื่องจาก $(y^* - y1) / (x - x1) = m$

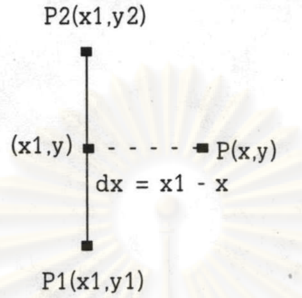
$$y^* = m(x - x1) + y1$$

ดังนั้น $dy = m(x - x1) + (y1 - y)$

เส้นตรง P1P2 จะถูกเลือกก็ต่อเมื่อ

$|dx| \le A$ หรือ $|dy| \le A$ โดย A คือค่าที่กำหนดขึ้น(ในการวิจัยนี้ กำหนดให้ a = 5)

ข้อพึงระวังในการใช้วิธีการเลือกแบบนี้ก็คือ กรณีที่เส้น P1P2 เป็นเส้นตรงที่ขนานกับแกน y เนื่องจากจะไม่สามารถหาค่าของ m ได้ ดังนั้นการพิจารณาจะเป็นการหาค่าของ dx จาก x1 - x โดยตรงแทน(ดูรูปที่ 3.7)

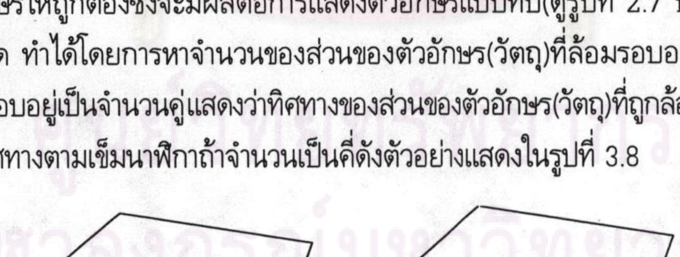


รูปที่ 3.7 แสดงการหาระยะทางจากจุด P กับเส้นตรง P1P2 ที่ขนานกับแกน y

สำหรับการนำไปประยุกต์ใช้กรณีการเลือกเส้นโค้งนั้นทำได้โดยง่าย เพียงแต่ทำการคำนวณหาชุดของเส้นตรงที่ใช้แทนเส้นโค้งก่อน แล้วทำการทดสอบกับเส้นตรงเหล่านั้น ถ้าพบว่าเส้นตรงเส้นใดเส้นหนึ่งในชุดของเส้นตรงถูกเลือกแสดงว่าผู้ใช้เลือกเส้นโค้งนั้น

<u>การหาทิศทางที่ถูกต้องของเส้น</u>

เนื่องจากการบรรยายการสร้างตัวอักษรแบบโพสต์สคริปต์ประเภทที่ 1 นั้น จำเป็นต้องกำหนดทิศทางของส่วนต่างๆของตัวอักษรให้ถูกต้องซึ่งจะมีผลต่อการแสดงตัวอักษรแบบทึบ(ดูรูปที่ 2.7 ประกอบ) การพิจารณาว่าส่วนใดควรจะมีทิศทางใด ทำได้โดยการหาจำนวนของส่วนของตัวอักษร(วัตถุ)ที่ล้อมรอบอยู่ กล่าวคือถ้าจำนวนของส่วนของตัวอักษรที่ล้อมรอบอยู่เป็นจำนวนคู่แสดงว่าทิศทางของส่วนของตัวอักษร(วัตถุ)ที่ถูกล้อมรอบนั้นจะต้องมีทิศทางทวนเข็มนาฬิกา และมีทิศทางตามเข็มนาฬิกาถ้าจำนวนเป็นคี่ดังตัวอย่างแสดงในรูปที่ 3.8



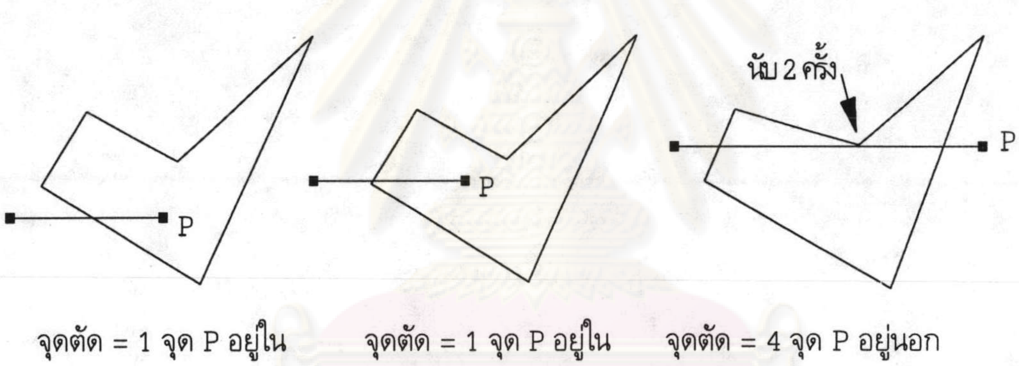
จำนวนวัตถุที่ล้อมรอบวัตถุ A มีค่า 1 ซึ่งเป็นจำนวนคี่ วัตถุ A จะต้องมีทิศทางตามเข็มนาฬิกา

จำนวนวัตถุที่ล้อมรอบวัตถุ A มีค่า 2 ซึ่งเป็นจำนวนคู่ วัตถุ A จะต้องมีทิศทางทวนเข็มนาฬิกา

รูปที่ 3.8 แสดงความสัมพันธ์ระหว่างทิศทางกับจำนวนของวัตถุที่ล้อมรอบ

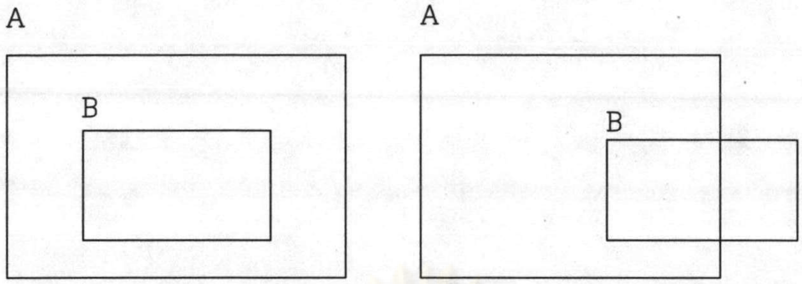
แนวทางในการหาจำนวนวัตถุที่ล้อมรอบ ได้ปรับมาจากการทดสอบตำแหน่งของจุดว่าอยู่ภายในหรือภายนอกรูปหลายเหลี่ยม (inside-outside test) โดยวิธีการคู่-คี่ (even-odd method) กล่าวคือสมมติจุด P(x, y) เป็นจุดที่ต้องการทดสอบ ให้ลากเส้นตรงระหว่างจุด P ขนานกับแกน x ไปทางซ้ายหรือขวาจนถึงจุดที่ทราบแน่นอนว่าเป็นจุดที่อยู่ภายนอกรูปหลายเหลี่ยม ถ้าจำนวนจุดตัดของเส้นตรงนี้กับรูปหลายเหลี่ยมเป็นเลขจำนวนคี่ จุด P จะเป็นจุดที่อยู่ภายในรูปหลายเหลี่ยม แต่ถ้าจำนวนจุดตัดเป็นเลขจำนวนคู่ จุด P จะเป็นจุดที่อยู่ภายนอกรูปหลายเหลี่ยม(Harrington, 1987; Mortenson, 1990) ดูรูปที่ 3.9 ประกอบ

ข้อพึงระวังในการใช้วิธีการคู่-คี่นี้ก็คือ ถ้าจุดตัดเกิดที่มุมของรูปหลายเหลี่ยมอาจจะให้ผลลัพธ์ผิดพลาดได้ การแก้ปัญหานี้จะต้องตรวจสอบว่าจุดตัดนั้นมาจากเส้นตรงที่อยู่ด้านเดียวกันกับเส้นตรงที่ลากขึ้นหรือไม่ ถ้าอยู่ด้านเดียวกันแสดงว่า จุดนั้นเป็นจุดยอดหรือจุดต่ำสุด ซึ่งจะต้องนับเป็นจำนวนคู่ (0 หรือ 2) เนื่องจากจุดที่ต้องการทดสอบจะอยู่ภายนอก แต่ถ้าอยู่คนละด้านจะต้องนับตามปรกติ



รูปที่ 3.9 แสดงวิธีการนับคู่-คี่

การนำเอาวิธีการนับคู่-คี่มาประยุกต์เพื่อใช้ทดสอบการอยู่ในหรือนอกของ 2 วัตถุ สามารถทำได้โดยพิจารณาวัตถุหนึ่งเป็นเสมือนรูปหลายเหลี่ยม และอีกวัตถุหนึ่งจะนำเอาจุดยอดของมุมใดมุมหนึ่งมาพิจารณา โดยถือว่าจุดยอดของมุมนั้นเป็นจุดที่ต้องการทดสอบ ถ้าจุดนั้นอยู่ในรูปหลายเหลี่ยมก็แสดงว่าวัตถุนี้อยู่ภายในวัตถุแรก แต่อย่างไรก็ดีสำหรับวัตถุที่มีความซับซ้อน เช่น มีการตัดกันของบางส่วน การพิจารณาจุดยอดของมุมเพียงจุดเดียวอาจจะให้ผลที่ไม่ถูกต้องได้ (ดูรูปที่ 3.10) ก็จำเป็นที่จะต้องนำเอาจุดยอดของมุมทุกจุดมาทำการทดสอบ(ในการวิจัยครั้งนี้หากวัตถุมีการตัดกัน จะถือว่าวัตถุ 2 วัตถุนั้นไม่มีความเกี่ยวข้องกัน)



การทดสอบจุดเดียวจะให้ผลเหมือนกับการทดสอบทุกจุด

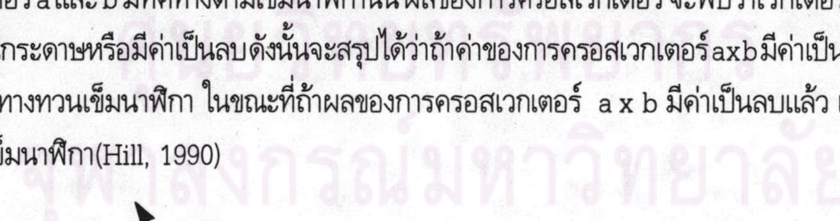
การทดสอบจุดเดียวจะให้ผลต่างจากการทดสอบทุกจุด

รูปที่ 3.10 แสดงการทดสอบการอยู่ใน-นอกของ 2 วัตถุ

สำหรับวัตถุที่ประกอบด้วยเส้นตรงและเส้นโค้งนั้นก็สามารถใช้หลักการดังกล่าวข้างต้นโดยการทำการคำนวณหาชุดของเส้นตรงที่ใช้แทนเส้นโค้งก่อน

<u>การหาทิศทางปัจจุบันของเส้น</u>

เมื่อสามารถหาทิศทางที่ควรจะเป็นของเส้นที่ลากเชื่อมต่อกันเป็นส่วนของตัวอักษรได้แล้ว จำเป็นที่จะต้องทำการเปรียบเทียบกับทิศทางปัจจุบันของเส้นเพื่อที่จะปรับทิศทางให้ถูกต้อง ซึ่งแนวทางการหาทิศทางของเส้นว่าวิ่งทวนเข็มนาฬิกาหรือตามเข็มนาฬิกานั้น สามารถทำได้โดยการใช้เวกเตอร์เข้ามาช่วย ดังรูปที่ 3.11 จะเห็นว่ากรณีที่เวกเตอร์ a และ b มีทิศทางทวนเข็มนาฬิกานั้น หากใช้กฎมือขวาหาทิศทางของเวกเตอร์ c ซึ่งเป็นผลของการครอสเวกเตอร์(cross vector)ระหว่าง a x b จะพบว่าเวกเตอร์ c นั้น จะมีทิศทางที่พุ่งออกจากกระดาษหรือมีค่าเป็นบวก ในขณะที่ถ้าเวกเตอร์ a และ b มีทิศทางตามเข็มนาฬิกานั้น ผลของการครอสเวกเตอร์ จะพบว่าเวกเตอร์ c นั้น จะมีทิศทางที่พุ่งกลับเข้าไปในกระดาษหรือมีค่าเป็นลบ ดังนั้นจะสรุปได้ว่าถ้าค่าของการครอสเวกเตอร์ axb มีค่าเป็นบวกแล้วเวกเตอร์ a และ b จะมีทิศทางทวนเข็มนาฬิกา ในขณะที่ถ้าผลของการครอสเวกเตอร์ a x b มีค่าเป็นลบแล้ว เวกเตอร์ a และ b จะมีทิศทางตามเข็มนาฬิกา(Hill, 1990)



การครอสเวกเตอร์จะให้ค่าบวก ทิศทางจะทวนเข็มนาฬิกา

การครอสเวกเตอร์จะให้ค่าลบ ทิศทางจะตามเข็มนาฬิกา

รูปที่ 3.11 แสดงการหาทิศทางของเส้นโดยใช้การครอสเวกเตอร์

ในการทำการครอสเวกเตอร์ a และ b เนื่องจากเป็นการพิจารณาระนาบของเวกเตอร์ที่เป็น 2 มิติ(x และ y) ค่าพิกัด z ของเวกเตอร์ a และ b จะมีค่าเป็น 0 เสมอ ดังนั้นเมื่อทำการครอสเวกเตอร์ a x b แล้ว ผลของการครอสเวกเตอร์จึงได้ค่าพิกัดของเวกเตอร์ c ในแนวแกน z ดังนี้

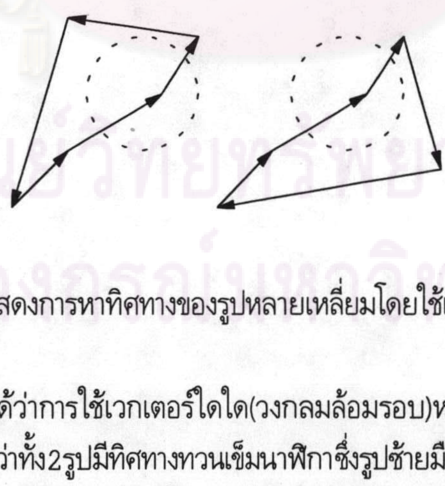
กำหนดให้ $a = [a_x\ a_y\ a_z]$ และ $b = [b_x\ b_y\ b_z]$

ดังนั้น $a \times b = c = [a_y{*}b_z - a_z{*}b_y \quad a_z{*}b_x - a_x{*}b_z \quad a_x{*}b_y - a_y{*}b_x]$

เนื่องจาก a_z และ $b_z = 0$ ดังนั้น $c = [0 \quad 0 \quad a_x{*}b_y - a_y{*}b_x]$

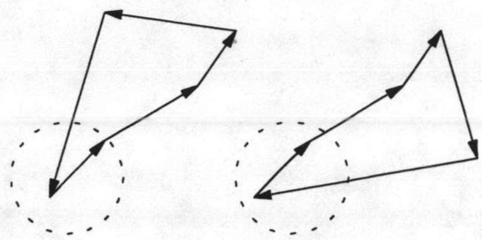
นั่นคือ ค่าของ $a_x{*}b_y - a_y{*}b_x$ จะเป็นค่าเวกเตอร์ c ในแนวแกน z ซึ่งทิศทางที่เป็นบวกจะแสดงถึงเวกเตอร์ a และ b มีทิศทางทวนเข็มนาฬิกา ในขณะที่ทิศทางที่เป็นลบจะแสดงถึงเวกเตอร์ a และ b มีทิศทางตามเข็มนาฬิกา

อย่างไรก็ดีการหาทิศทางของเส้นโดยใช้เวกเตอร์นี้จะเป็นจริงเฉพาะในกรณีของ 2 เวกเตอร์เท่านั้น แต่ในการวิจัยนี้จำเป็นที่จะต้องหาทิศทางของรูปหลายเหลี่ยม ซึ่งประกอบกันขึ้นจากหลายๆเวกเตอร์ แนวทางในการแก้ปัญหานี้ก็คือ จะต้องพยายามหาเวกเตอร์ 2 เวกเตอร์ที่เป็นตัวแทนเพื่อใช้ในการหาทิศทาง โดยใช้เวกเตอร์ที่เชื่อมต่อกันตรงจุดคู่ลำดับที่มีค่าพิกัดxอยู่ทางซ้ายสุดของรูปหลายเหลี่ยมนั้น(สาเหตุที่ต้องใช้เวกเตอร์ซ้ายสุดของรูปหลายเหลี่ยมนั้นเพราะว่าทิศทางของเวกเตอร์ที่หาได้นั้นจะบอกถึงทิศทางที่แท้จริง ในขณะที่ถ้าใช้เวกเตอร์ในส่วนอื่นๆนั้น ทิศทางที่หาได้จะเป็นจริงเฉพาะเวกเตอร์นั้นๆเท่านั้น (ดังรูปที่ 3.12) เนื่องจากเวกเตอร์ที่ต่อกับเวกเตอร์ที่ใช้หาทิศทางนั้นมีโอกาสที่จะเปลี่ยนทิศทางได้)



รูปที่ 3.12 แสดงการหาทิศทางของรูปหลายเหลี่ยมโดยใช้เวกเตอร์ใดใด

จากรูปที่ 3.12 จะเห็นได้ว่าการใช้เวกเตอร์ใดใด(วงกลมล้อมรอบ)หาทิศทางของรูปหลายเหลี่ยมมีโอกาสผิดได้โดยในรูปนี้การหาทิศทางจะได้ว่าทั้ง2รูปมีทิศทางทวนเข็มนาฬิกาซึ่งรูปซ้ายมือทิศทางจะถูกต้อง ในขณะที่รูปขวามือจะต้องมีทิศทางตามเข็มนาฬิกาซึ่งแสดงให้เห็นว่าการใช้เวกเตอร์ใดใดจะให้ผลผิดพลาด



รูปที่ 3.13 แสดงการหาทิศทางของรูปหลายเหลี่ยมโดยใช้เวกเตอร์ซ้ายสุด

จากรูปที่ 3.13 จะเห็นได้ว่าการใช้เวกเตอร์ซ้ายสุด(วงกลมล้อมรอบ)จะให้ผลลัพธ์ที่ถูกต้องทั้ง 2 กรณี โดยรูปซ้ายจะให้ทิศทางทวนเข็มนาฬิกา ในขณะที่รูปขวาจะให้ทิศทางตามเข็มนาฬิกา