การเขียนภาพเคลื่อนไหวในระบบ 3 มิติ

3D COMPUTER GRAPHICS ANIMATION

โดย

นาย คงเดช กรกาญจนารักษ์ รหัส 321041

นางสาว บุญล้วน วรรณวิมลศรี รหัส 321164

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปริญญานิพนธ์ปีการศึกษา 2535
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

เรื่อง การเขียนภาพเคลื่อนไหวในระบบ 3 มิติ

ผู้จัดทำ

นาย คงเดช กรกาญจนารักษ์ รหัส 321041
นางสาว บุญล้วน วรรณวิมลศรี รหัส 321164

.....อาจารย์ที่ปรึกษา
(ดร. เอื้อน ปิ่นเงิน)

บทคัดย่อ

ในปัจจุบันงานทางด้านโฆษณา มีการแข่งขันกันมาก ซึ่งได้ใช้เทคนิคต่าง ๆ เข้าช่วยมากมาย และเมื่อไม่นานมานี้ได้มีการใช้คอมพิวเตอร์กราฟฟิคมาช่วยในงานโฆษณา และได้รับความสนใจมาก จากนั้นก็มีโฆษณาที่ใช้คอมพิวเตอร์กราฟฟิคเพิ่มขึ้นเรื่อย ๆ โดยที่ software ที่ใช้นั้นมีราคาแพงมาก และต้องรันบนเครื่อง Workstation ซึ่งก็มีราคาแพงมากเช่นกัน เพราะฉะนั้นโครงงานนี้จึงเกิดขึ้น เพื่อเป็นแนวทางในการสร้างและพัฒนา software ที่รันทางด้านคอมพิวเตอร์กราฟฟิคที่ใช้ในการโฆษณา ซึ่งเป็น software ที่รันบนเครื่อง PC_ ซึ่งมีราคาถูก และใช้กันอย่างแพร่หลายในปัจจุบัน

โครงงานนี้เป็นโครงงานต่อเนื่อง 2 ภาคการศึกษาด้วยกัน โดยในภาคการศึกษาที่ 1 เป็นการศึกษาเกี่ยวกับการแสดงภาพ 3 มิติบนระบบคอมพิวเตอร์ ซึ่งประกอบไปด้วยหลักการพื้นฐานของคอมพิวเตอร์กราฟฟิค, ระบบโคออร์ดิเนท, เส้นตรง, รูปหลายเหลี่ยม, เมตริกซ์กับการแปลง(Transformation), วินโดว์และการคลิป, การทำโปรเจคชั่น, พารามิเตอร์ภาพ, การลบเส้นและพื้นผิวที่ถูกบัง, การวาดวัตถุบังกัน และในภาคการศึกษาที่ 2 ได้ศึกษาเกี่ยวกับการลงสีของวัตถุ(shading) และหลักการทำภาพเคลื่อนไหวในแบบต่าง ๆ

ส่วนผลงานที่ได้จากโครงงานนี้เป็น งานโฆษณาคณะวิศวกรรมศาสตร์ ของ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง ซึ่งเป็นการแนะนำให้ผู้ที่ประสงค์จะติดต่อกับทางคณะฯ สามารถที่จะทราบว่าจะต้องไปติดต่อที่ตึกใดและตั้งอยู่ที่ไหน โดยจัดทำเป็นภาพกราฟฟิคเคลื่อนไหวและมีเมนูให้เลือกในส่วนของตัวอาคารต่าง ๆ โดยผลงานนี้ใช้ Tools 2 ตัวในการสร้างคือ AUTODESK_3D Studio ใช้ในการสร้างตัวอาคารและออปเจคต่าง ๆ และ GRASP (Grasphics Animation System for Professional) เป็น interpreter ที่ใช้ในการสร้างภาพเคลื่อนตาม scrip ที่ได้กำหนดขึ้น และยังมี software ซึ่งใช้สร้างภาพ 3 มิติ ซึ่งเขียนจากทฤษฎีที่ได้ทำการศึกษามา

ABSTRACT

Nowadays, there's very competitive in advertising work that assisted by various techiques. In the recent days, Computer graphic has been taken part in advertising work and this brings about much being interested in it. Then advertisements that co-produced by computer graphic are incresing. However, a sofware for the affan is very expensive and must run on the workstation that also costly. That brings about this project, to be a guideline of the invention and development of the computer graphic software for advertisement. It's a software that running on the low-cost PC and becomes very popular today.

This project is continue in two terms. The first term learnt about 3D model on computer system that consist of basic of computer graphics, coordinate system, transformation matrix, windowing and cliping, projection, viewing parameters, hidden line and surface, visibility of sereral objects. The second term learnt about shading and animation in various techniques.

The product of this project is the "Advertising about factory of engineer in King Monkut's Institute of Technology Landkrabang" that advise peson who communicate with factory about which and where building by display computer graphics animation with menu to select. This product produced by two tools are AUTODESK 3D Studio (draw objects) and GRASP : Graphics Animation System for Professinals (the interpreter of scrip program)

บทนำ

ในปัจจุบันงานทางด้านโฆษณา มีการแข่งขันกันมาก ซึ่งได้ใช้เทคนิคต่าง ๆ เข้าช่วยมากมาย และเมื่อไม่นานมานี้ได้มีการใช้คอมพิวเตอร์กราฟฟิคมาช่วยในงานโฆษณา และได้รับความสนใจมาก งานที่อาศัยการโฆษณาโดยใช้คอมพิวเตอร์กราฟฟิค เช่น งานประเภทโฆษณาทางทีวี เพราะเป็นการลงทุนครั้งเดียวแล้วเกิดผลกระทบกับผู้ชมทั่วประเทศ เช่น AutoDome (ศูนย์ซ่อมรถยนต์ครบวงจร) สามารถแสดงถึงความไฮเทคของตัวสินค้า หรือเป็นการนำเสนอสินค้าที่ไม่มีอยู่จริง เช่น เมืองทองธานี ก็เป็นการนำเสนอตรงกับความต้องการของลูกค้าที่อยากจะชมโมเดลสิ่งของตัวบ้าน เพื่อเป็นข้อมูลในการตัดสินใจซื้อก่อนชมในพื้นที่จริง

การสร้างภาพ 3D Animation นั้นไม่แตกต่างจากการสร้างภาพยนต์เท่าใดนัก เริ่มต้นจะต้องสร้างวัตถุลงแบคกราวน์ และกำหนดการเคลื่อนไหวให้สอดคล้องกับวัตถุอื่นในระบบ ต้องมีการให้แสงเงาแก่วัตถุ (กำหนดดวงไฟ) และกำกับการเดินหรือการเคลื่อนที่ของวัตถุ และภาพที่ได้แต่ละเฟรมจะได้จากการกำหนดมุมกล้อง เนื่องจากภาพแต่ละเฟรมมีขนาดใหญ่และมีการแสดงผลอย่างต่ำ 30 ภาพต่อวินาที ดังนั้นจะต้องใช้หน่วยความจำมหาศาลในการเก็บข้อมูล เช่น ภาพจากจอขนาดความละเอียด 300x200 มีสีได้ 256 สีนั้น พบว่า จะต้องใช้หน่วยความจำขนาด 60,000 ไบต์/ภาพ ถ้าต้องการผลงานคุณภาพสูงแล้วละก็จะต้องใช้จำนวน 30 เฟรม/วินาที ดังนั้น 1 นาที จะต้องการข้อมูลถึง 1,800,000 ไบต์ซึ่งที่จริงแล้วความละเอียดขนาดนี้ยังไม่เพียงพอในการทำภาพยนตร์ และโฆษณาโดยความละเอียดไม่ควรต่ำกว่า 800 x 600 โดยมีสีอย่างต่ำ 65,536 สี ดังนั้นสำหรับมืออาชีพจะต้องใช้เครื่องที่มีความสามารถทางด้านกราฟฟิคสูงเช่น เครื่องเวิร์คสเตชั่น และมีหน่วยความจำขนาดใหญ่ โครงงานนี้ได้จัดทำงานโฆษณาซึ่งเป็นการแนะนำให้รู้จักคณะวิศวกรรมศาสตร์ของสถาบันโดยใช้เครื่องระดับ PC ซึ่งมีราคาถูกเพื่อเป็นแนวทางในการทำงานที่มีคุณภาพดีทัดเทียบเครื่องที่มีคุณภาพสูงและราคาสูงได้

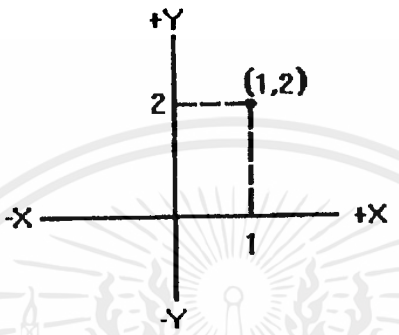
สารบัญ

บทที่ 1

Basic of Computer Graphics

1.1 ระบบโคออร์ดิเนท

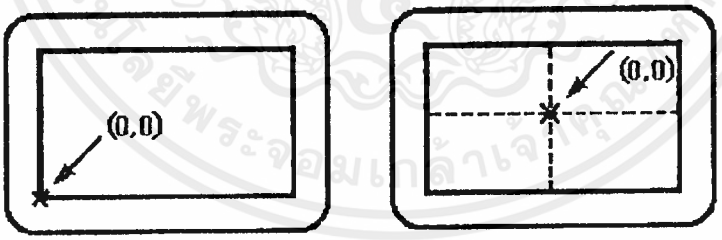
ระบบโคออร์ดิเนทที่รู้จักทั่ว ๆไปก็คือระบบโคออร์ดิเนท XY ซึ่งใช้คู่ลำดับในการกำหนดจุดต่าง ๆ ในระบบ เช่นจุด (1,2) หมายถึงจุดที่อยู่ห่างจากจุดเริ่มต้นไปทางแกน +X 1 หน่วยและแกน +Y 2 หน่วย ดังรูปแสดงในรูป 1.1



รูป 1.1 แสดงตำแหน่งของจุด (1,2)

ภาพในระบบกราฟฟิคจะถูกวาดขึ้นบนจอภาพดังนั้นการอ้างถึงจุดหรือตำแหน่งใด ๆ บนจอภาพเราจึงต้องใช้โคออร์ดิเนทด้วยเช่นกัน แต่เป็นโคออร์ดิเนทของจอภาพแทนที่จะเป็นโคออร์ดิเนท XY อย่างไรก็ตามเรามักจะพบปัญหาที่เกิดขึ้นจากความแตกต่างกันทางด้านฮาร์ดแวร์ของจอภาพ ซึ่งมักจะเกิดจาก

- จุดกำเนิด (0,0) ของโคออร์ดิเนทของจอภาพอยู่คนละตำแหน่ง จอบางชนิดอยู่มุมล่างซ้าย บางชนิดอยู่ตรงกลางจอภาพ ดังแสดงในรูป 1.2

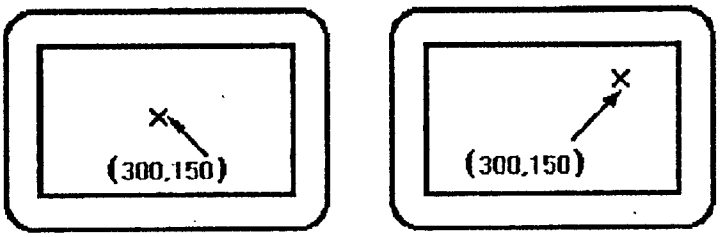


รูป 1.2 แสดงตำแหน่งของจุดเริ่มต้นบนจอภาพบางชนิด

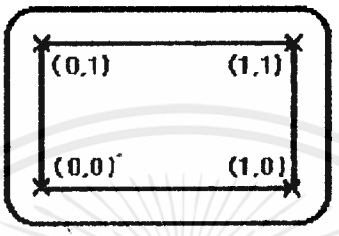
- สเกลที่ใช้ต่างกันอันเนื่องมาจากความละเอียดของจอภาพตัวอย่างเช่น จอภาพที่มีจุดเริ่มต้นตรงกลางเหมือนกันแบบแรกมีความละเอียด 600x300 แบบที่สองมีความละเอียด 1200x1000 ดังนั้นถ้าอ้างถึงจุด (300,150) จอภาพแบบแรกจะได้จุดอยู่บนตำแหน่งมุมบนขวาแต่สำหรับจอภาพแบบที่สองจะได้ตำแหน่งที่ใกล้จุดกึ่งกลางจอภาพเข้ามาอีก (ดูรูป 1.3)

เพื่อตัดปัญหาความยุ่งยากของความแตกต่างกันของจอภาพคนละชนิดในระบบคอมพิวเตอร์กราฟฟิคจึงได้กำหนดโคออร์ดิเนทบนจอภาพเอาไว้ให้ใช้ได้กับจอภาพทุกชนิดเป็นมาตรฐานเดียวกันหมดเรียกว่า *โคออร์ดิเนทของอุปกรณ์มาตราฐาน(Normal Device Coordinate)* การวางตำแหน่งของจุดต่าง ๆบนจอภาพไม่ว่าจะมีขนาดใหญ่หรือเล็ก

หรือมีความละเอียดเท่าใดก็จะมีลักษณะเหมือนกันหมด คือมีความกว้างและสูง 1 หน่วย จุดกำเนิดอยู่ที่มุมล่างซ้าย ดังในรูป 1.4



รูป 1.3 แสดงตำแหน่งของจุด (300,150) บนจอภาพที่มีความละเอียดต่างกัน



รูป 1.4 แสดงลักษณะของโคออร์ดิเนทของอุปกรณ์มาตรฐาน

ค่าพารามิเตอร์ของคำสั่งต่าง ๆ ที่เกี่ยวข้องกับโคออร์ดิเนท จะต้องใช้ค่าของโคออร์ดิเนทของอุปกรณ์มาตราฐานเท่านั้นในลักษณะนี้ทำให้ผู้ใช้ไม่ต้องพะวงว่ากำลังใช้งานจอภาพชนิดใด มีจุดกำเนิดของโคออร์ดิเนทอยู่ที่ใดและมีความละเอียดของจอภาพเท่าไรขอเพียงให้ทราบว่าโคออร์ดิเนทของอุปกรณ์มาตราฐานเป็นอย่างไรเท่านั้นก็พอ การทำงานของคำสั่งต่าง ๆ จะแปลงให้เป็นตำแหน่งโคออร์ดิเนทที่แท้จริงของจอภาพให้เอง

1.2 เส้นตรง

เส้นต่าง ๆเป็นส่วนประกอบที่สำคัญและพื้นฐานที่สุดในการวาดภาพ เส้นมีเพียง 2 ชนิดเท่านั้นคือ เส้นตรงและเส้นโค้ง ในที่นี้จะขอกล่าวถึงเส้นตรงเท่านั้นเพราะเข้าใจได้ง่ายในระบบโคออร์ดิเนท เส้นตรงทุกเส้นจะมีสมการเส้นตรงของมัน ดังนั้นการกล่าวอ้างถึงหรือชี้เฉพาะเจาะจงเส้นตรงหนึ่ง ๆ จะใช้สมการเส้นตรงระบุ สมการเส้นตรงมีหลายรูปแบบ เช่น

$$y = mx + b \qquad (1.1)$$

โดยที่ m คือ ความชันของเส้นตรง

b คือ จุดตัดแกน Y ของเส้นตรง

หรือมีรูปแบบเป็น

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1} \qquad (1.2)$$

โดยที่ $(x_1,y_1),(x_2,y_2)$ คือจุด 2 จุดบนเส้นตรง

จากสมการ (1.2)

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

$$y - y_1 = \left[\frac{y_2 - y_1}{x_2 - x_1}\right](x - x_1)$$

$$y = \left[\frac{y_2 - y_1}{x_2 - x_1}\right]x - \left[\frac{y_2 - y_1}{x_2 - x_1}\right]x_1 + y_1 \qquad (1.3)$$

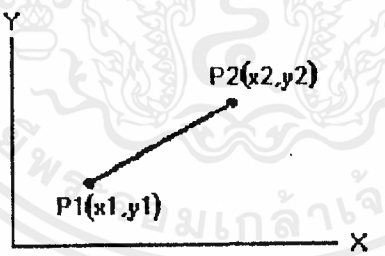
เปรียบเทียบสมการ (1.2) กับ (1.3) จะเห็นว่า ความชัน

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

และ

$$b = \left[\frac{y_2 - y_1}{x_2 - x_1}\right]x_1 + y_1 \qquad (1.4)$$

นั้นคือถ้าทราบจุด 2 จุดบนเส้นตรงเราสามารถหาความชันของเส้นตรงนั้นๆ ได้โดยใช้สมการ (1.4) สมการเส้นตรงนั้นใช้แทนเส้นตรงที่มีความยาวไม่จำกัด แต่การวาดเส้นตรงให้เกิดขึ้นบนจอเราต้องการเพียง "ส่วนของเส้นตรง" (Line Segment) เท่านั้น การกำหนดส่วนของเส้นตรงกำหนดด้วยตำแหน่งของจุดปลายทั้งสองของส่วนของเส้นตรง ดังตัวอย่างในรูป1.5 ซึ่งเราจะได้เส้นตรงระหว่างจุด P_1 และ P_2



รูป 1.5 ตัวอย่างการกำหนดส่วนของเส้นตรง

1.3 การสร้างเส้นตรงบนจอภาพ

บนจอภาพแบบราสเตอร์พิกเซลแต่ละพิกเซลมีตำแหน่งที่แน่นอนตายตัวซึ่งตำแหน่งของพิกเซลกำหนดได้ด้วยโคออร์ดิเนทของจอภาพตัวอย่างเช่นจอภาพโมโนโครมที่ใช้กับเครื่องไมโครคอมพิวเตอร์มีความละเอียด 720x348พิกเซลตำแหน่งที่ (0,0) คือจุดที่อยู่มุมบนซ้ายพิกเซลตำแหน่งที่ (0,347) คือจุดที่อยู่มุมล่างซ้าย พิกเซลตำแหน่งที่ (719,347) คือพิกเซลที่อยู่ตำแหน่งมุมล่างขวาจะเห็นว่าการที่จะกำหนดหรืออ้างถึงพิกเซลใดๆ เราต้องกำหนดเป็นโคออร์ดิเนทของพิกเซลนั้นๆให้ถูกต้องและที่สำคัญคือเป็นโคออร์ดิเนทของเลขจำนวนเต็มเท่านั้น

ในการสร้างเส้นตรงบนจอภาพแบบราสเตอร์เราจะอาศัยหลักการที่ว่า เส้นตรงเกิดจากจุดเล็ก ๆ เรียงติดต่อกัน ดังนั้นเราจะใช้พิกเซลของจอภาพแทนจุดทำให้เกิดเป็นเส้นตรงขึ้นมาบนจอภาพ การสร้างจุดให้เกิดขึ้นบนจอภาพทำได้

โดยเปลี่ยนสี (แอตตริบิ้ว) ของพิกเซล ณ ตำแหน่งที่เราต้องการให้เกิดจุดซึ่งระบุโดยใช้โคออร์ดิเนทของพิกเซลนั้น สีที่จะเปลี่ยนต้องต่างจากสีพื้นหรือสีแบคกราวนด์ เช่น สีแบคกราวนด์ของจอภาพเป็นสีขาว การสร้างจุดก็คือการเปลี่ยนสีของพิกเซลเป็นดำ หรือแดงหรือสีอื่น ๆ ที่ไม่ใช่สีขาว

ความจริงแล้วสีแบคกราวนด์ของจอภาพก็คือสีของพิกเซลทุกพิกเซลก่อนที่จะมีการทำคำสั่งใด ๆ เพื่อให้เกิดภาพขึ้นบนจอ ซึ่งเป็นสีเดียวกันหมดทั่วทั้งจอภาพ คำสั่งในการเปลี่ยนสีหรือแอตตริบิ้วของพิกเซลบนจอภาพ มักจะมีรูปแบบดังนี้

Set-Attribute (X,Y,C)

โดยที่ (X,Y) คือโคออร์ดิเนทหรือตำแหน่งของพิกเซลที่ต้องการเปลี่ยนสีของมันและ C คือสีที่ต้องการให้พิกเซลนั้นเป็น

ต่อไปจะเป็นการสร้างเส้นตรงบนจอภาพแบบราสเตอร์โดยอาศัยวิธีการสร้างจุดทีละจุดเพื่อให้เกิดเส้นตรงขึ้น จากสมการ (1.1)

$$y = mx + b \qquad (1.5)$$

ถ้าเรามีจุด $P1(x_1,y_1)$ และ $P2(x_2,y_2)$ อยู่บนเส้นตรงนี้ ดังนั้น

$$y_1 = mx_1 + b$$
$$y_2 = mx_2 + b$$
$$y_2-y_1 = m(x_2-x_1) \qquad (1.6)$$

ถ้า $x_2 = x_1 + 1 \qquad (1.7)$

แทนค่าสมการ (1.7) ลงในสมการ (1.6) จะได้

$$y_2-y_1 = m \qquad (1.8)$$

จากสมการ(1.6) (1.7) และ (1.8) อธิบายได้ว่า จุดต่าง ๆ ที่อยู่บนเส้นตรงเดียวกัน ถ้า X เพิ่มค่าขึ้น 1 ค่าของ Y จะต้องเพิ่มขึ้น m ในทำนองเดียวกันเราสามารถพิสูจน์ได้ว่าถ้าเราเพิ่มค่า Y ขึ้น 1 ค่าของ X ต้องเพิ่มขึ้น 1/m จากข้อสรุปที่กล่าวมานี้ทำให้เราสร้างอัลกอริทึมที่ใช้ในการวาดเส้นตรงบนจอภาพแบบราสเตอร์ได้ซึ่งต้องการเพียงตำแหน่งของจุดปลายทั้ง 2 ของเส้นตรงคือ (x_1,y_1) และ (x_2,y_2) เท่านั้น การทำงานของอัลกอริทึมเป็นดังนี้

1. คำนวณหาความชัน m
2. ให้ $x = x_1$ และ $y = y_1$
3. เปลี่ยนสีของพิกเซลที่มีโคออร์ดิเนทใกล้เคียงกับจุด (x,y) มากที่สุด
4. เพิ่มค่า x ขึ้น 1 และเพิ่มค่า y ขึ้น m

5. ทำซ้ำข้อ 2 3 4 จนกระทั่ง x มากกว่าหรือเท่ากับ x_2

<u>หมายเหตุ</u> x_1 ต้องน้อยกว่าหรือเท่ากับ x_2

อัลกอริทึมข้างบนนี้จะใช้ได้ดีกับเส้นตรงที่มีความชันน้อยกว่าหรือเท่ากับ 1 (มีมุมเอียงไม่เกิน 45 องศา)ถ้าเส้นตรงมีความชันมากกว่า 1 จะเกิดอลิแอสซิ่งและอาจทำให้เส้นที่ไม่ต่อเนื่อง ดังนั้นในกรณีที่ค่าความชันมากกว่า 1 เราจะแก้ไขวิธีการสร้างเส้นตรงเล็กน้อยดังนี้

1. คำนวณค่าส่วนกลับของ m : $m' = 1/m = (x_2-x_1)/(y_2-y_1)$
2. ให้ $x = x_1$ และ $y = y_1$
3. เปลี่ยนสีของพิกเซลที่มีโคออร์ดิเนทใกล้เคียงกับจุด (x,y) มากที่สุด
4. เพิ่มค่า y ขึ้น 1 และเพิ่มค่า x ขึ้น m'
5. ทำซ้ำข้อ 2 3 4 จนกระทั่ง y มากกว่าหรือเท่ากับ y_2

<u>หมายเหตุ</u> y_1 ต้องน้อยกว่าหรือเท่ากับ y_2

ดังนั้นการสร้างเส้นตรงจากจุดต้องแยกพิจารณากรณีของความชันmมากกว่า 1 และน้อยกว่าหรือเท่ากับ 1 เพื่อให้เส้นตรงมีความเรียบมากที่สุด เป็นการลดอลิแอสซิ่ง และทำให้เส้นตรงมีความต่อเนื่องกันตลอดอาศัยอัลกอริทึมที่กล่าวมานี้ เราสามารถสร้างคำสั่งในการลากเส้นซึ่งรับพารามิเตอร์เป็นโคออร์ดิเนทของจุดปลายทั้ง 2 ของเส้นตรง คำสั่งที่ทำหน้าที่ว่าเส้นตรงมักมีรูปแบบดังนี้

$$Line(X_1,Y_1,X_2,Y_2)$$

หมายความว่าเป็นการลากเส้นตรงจากจุด (X_1,Y_1) ไปยังจุด (X_2,Y_2) คำสั่งวาดเส้นตรงที่สร้างจากอัลกอริทึมที่กล่าวมาแล้วนั้น ต้องใช้คำสั่งเปลี่ยนแอตตริบิ้วของพิกเซลในการทำงานนั้นคือคำสั่ง Set-Attribute ก็เป็นคำสั่งพื้นฐานของการสร้างคำสั่ง Line

1.4 คำสั่งอื่น ๆ ที่เกี่ยวข้อง

ในระบบคอมพิวเตอร์กราฟฟิคบางระบบจะมีตัวแปรตัวหนึ่งชื่อ *CP (Current Pointer)* ตัวแปรนี้คล้าย ๆกับเคอร์เซอร์ในระบบคอมพิวเตอร์ทั่วไป ทุกครั้งที่มีการพิมพ์ข้อความออกสู่จอภาพตำแหน่งของเคอร์เซอร์จะไปอยู่ในตำแหน่งท้ายสุดของข้อความที่ถูกพิมพ์ออกมา CP ก็ เช่น เดียวกันแต่มีอยู่ในระบบการทำงานแบบกราฟฟิคและ CP นี้จะมองไม่เห็นบนจอภาพ ทุกครั้งที่มีการวาดรูปบนจอภาพ CP จะอยู่ที่ตำแหน่งสุดท้ายของการวาดครั้งนั้น ตัวอย่างเช่น เราสั่งให้วาดเส้นตรงจากจุด (X_1,Y_1) ไปยังจุด (X_2,Y_2) หลังจากทำคำสั่งนี้แล้ว CP จะไปอยู่ที่ตำแหน่ง (X2,Y2)

เราอาจจะเปรียบ CP หัวปากกาของพล็อตเตอร์ก็ได้ เราใช้ปากกานี้ในการเขียนเส้นบนกระดาษการลากเส้นจากจุด (X_1,Y_1) ไปยังจุด (X_2,Y_2) เราต้องยกปากกาไปไว้ที่จุด (X_1,Y_1) และลากเส้นตรงไปยังจุด (X_2,Y_2) เมื่อลากเส้นตรงเสร็จแล้ว ถ้าไม่มีคำสั่งอื่นต้องทำต่อหัวปากกาก็จะหยุดค้างอยู่ที่ตำแหน่ง (X_2,Y_2) นั้นคือ CP อยู่ที่ตำแหน่ง (X_2,Y_2) ถ้ามีการลากเส้นครั้งต่อไป จึงค่อยเลื่อนหัวปากกา (หรือ CP) ไปยังตำแหน่งที่ต้องการได้จะสังเกตุได้ว่าการเคลื่อนที่ของหัวปากกาหรือ CP นี้มี 2 แบบคือ การเคลื่อนที่ไปยังตำแหน่งใหม่แล้วเกิดเส้นตรงขึ้นมา (หัวปากกาลากเส้นขณะเคลื่อนที่) ลักษณะเช่นนี้เหมือนกับไปกระทำคำสั่งลากเส้นแต่ต่างกันตรงที่เป็นการลากเส้นจากจุดที่หัวปากกาหรือ CP อยู่ในขณะนั้น

ไปยังจุดอื่น ลักษณะเช่นนี้เรามักจะมีคำสั่งอีกคำสั่งหนึ่ง คือ

Line-to (x,y) หรือ Line-Abs (x,y)

คำสั่งนี้เป็นการลากเส้นตรงจากจุดที่ CP อยู่ในขณะนั้นไปยังจุด (x,y) ซึ่งหลังจากทำคำสั่งนี้แล้วตำแหน่งของ CP จะย้ายไปอยู่ที่ (x,y) ด้วย (คำสั่ง Line-Abs ย่อมาจาก Line-Absolute)

การเคลื่อนที่หัวปากกาอีกลักษณะหนึ่งคือ เคลื่อนที่ไปยังตำแหน่งอื่นโดยไม่มีเส้นเกิดขึ้น (ยกหัวปากกาไป) ลักษณะนี้เป็นการเปลี่ยนตำแหน่งของ CP เท่านั้น รูปแบบของคำสั่งมักจะเป็น

Move-to (x,y) หรือ Move-Abs (x,y)

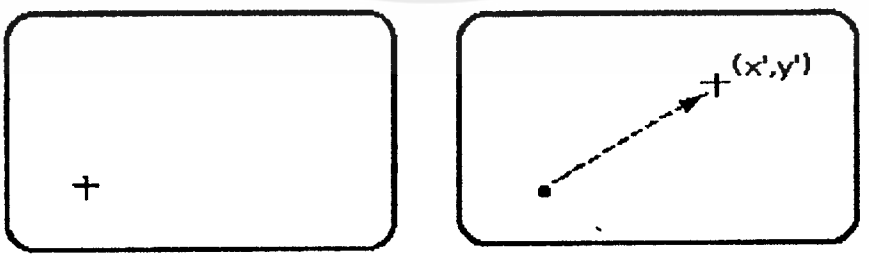
คำสั่งนี้เป็นการเปลี่ยนตำแหน่งของ CP ให้ไปอยู่ที่จุด (x,y) โดยไม่สนใจว่าก่อนที่ทำคำสั่งนี้ CP อยู่ที่ตำแหน่งใด (คำสั่ง Move-Abs ย่อมาจาก Move-Absolute) ในรูป 1.6 และ 1.7 เป็นตัวอย่างของผลลัพธ์ที่ได้จากการใช้คำสั่ง Line-to และ Move-to เครื่องหมายกากบาทเป็นเครื่องหมายที่แสดงให้เห็นถึงตำแหน่งของ CP เท่านั้น

ก) ก่อนทำคำสั่ง ข) หลังทำคำสั่ง

รูป 1.6 ผลจากการใช้คำสั่ง Line-to (x',y')

ยังมีคำสั่งเกี่ยวกับการเคลื่อนที่ของหัวปากกาหรือ CP อีก 2 คำสั่ง ซึ่งมีลักษณะคล้ายกับคำสั่ง Line-to และ Move-to คำสั่งแรกคือคำสั่ง

Line-Rel (dx,dy)

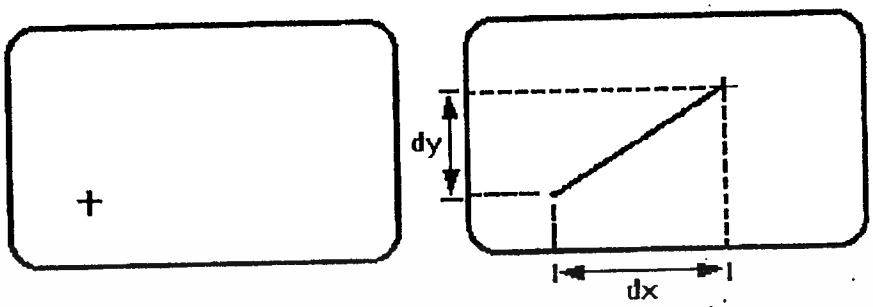


ก) ก่อนทำคำสั่ง ข) หลังทำคำสั่ง

รูป 1.7 ผลจากการใช้คำสั่ง Move-to (x',y')

(ย่อมาจาก Line-Relative) คือการสั่งให้ลากเส้นตรงจากจุดที่ CP อยู่ในขณะนั้นไปยังจุดที่ห่างไปทางแกน X

เป็นจำนวน dx และห่างไปทางแกน Y เป็นจำนวน dy ทั้ง dx และ dy นี้อาจเป็นได้ทั้งจำนวนบวกหรือจำนวนลบก็ได้สมมุติว่าเดิม CP อยู่ที่ตำแหน่ง (x_1,y_1) คำสั่งLine-Rel (dx,dy) จะลากเส้นตรงจากจุด (x_1,y_1) ไปยังจุด (x_1+dx,y_1+dy) จะเห็นว่าจุดสุดท้ายของเส้นตรงที่เกิดขึ้นขึ้นอยู่กับตำแหน่งของCPก่อนทำคำสั่ง Line-Rel ในรูป 1.8 เป็นตัวอย่างการใช้คำสั่ง Line-Rel



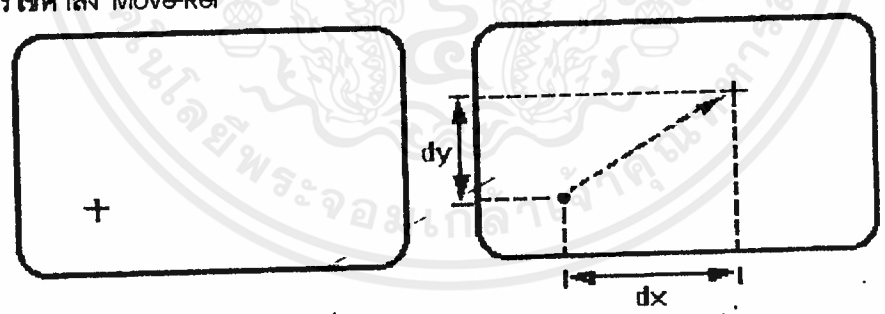
ก) ก่อนทำคำสั่ง ข) หลังทำคำสั่ง

รูป 1.8 ผลจากการใช้คำสั่ง Line-Rel (dx,dy)

ส่วนคำสั่งที่สองคือ คำสั่ง

Move-Rel (dx, dy)

(ย่อมาจาก Move-Relative) ซึ่งเป็นการสั่งให้ย้าย CP จากตำแหน่งที่อยู่ในขณะนั้นไปยังตำแหน่งใหม่ซึ่งห่างออกไปทางแกน X เป็นจำนวน dx และห่างออกไปทางแกน Y เป็นจำนวนdyและเช่นกันทั้ง dx และ dy เป็นได้ทั้งจำนวนบวกหรือจำนวนลบก็ได้ ตัวอย่างเช่นเดิม CP อยู่ที่ตำแหน่ง (x_1,y_1) คำสั่ง Move-Rel (dx,dy) จะทำให้ CP ย้ายไปยังตำแหน่ง (x_1+dy,y_1+dy) นั้นคือตำแหน่งใหม่ของ CP ขึ้นอยู่กับตำแหน่งเดิมของมันก่อนทำคำสั่ง Move-Rel รูป 1.9 เป็นตัวอย่างของการใช้คำสั่ง Move-Rel



ก) ก่อนทำคำสั่ง ข) หลังทำคำสั่ง

รูป 1.9 ผลจากการใช้คำสั่ง Move-Rel(dx,dy)

1.5 รูปหลายเหลี่ยม

รูปหลายเหลี่ยม (Polygon) เกิดจากการนำเอาส่วนของเส้นตรง 3 เส้นหรือมากกว่ามาต่อกันไปเรื่อยๆ จนกระทั่งมาบรรจบกับกันเป็นรูปปิด และไม่มีการตัดกันของส่วนของเส้นตรงเหล่านั้นดังเช่นแสดงไว้ในรูป 1.10 ก)และ ข) ส่วนรูป ค) และ ง) ไม่เป็นรูปหลายเหลี่ยมเพราะมีการตัดกันของเส้น และไม่เป็นรูปปิดตามลำดับ

ก) ข) ค) ง)

รูป 1.10 ตัวอย่างของรูปแบบต่าง ๆ

เราสามารถแบ่งประเภทของรูปหลายเหลี่ยมได้เป็น2ประเภทคือ *รูปหลายเหลี่ยมเว้าเข้า (ConcapePolygon)* และ *รูปหลายเหลี่ยมเว้าออก (Convex Polygon)* รูปหลายเหลี่ยมใด ๆ ที่สามารถลากเส้นตรงภายในรูปหลายเหลี่ยมเชื่อมจุด 2 จุดบนขอบของเส้นรอบรูปนั้นแล้วไม่มีส่วนหนึ่งส่วนใดของเส้นตรงอยู่ภายนอกขอบของรูปหลายเหลี่ยมเราเรียกรูปหลายเหลี่ยมประเภทนี้ว่า รูปหลายเหลี่ยมเว้าออกจากคำนิยามนี้จะเห็นว่า รูปสามเหลี่ยมเป็นรูปหลายเหลี่ยมเว้าออกเสมอ ในรูป 1.11 ก) เป็นตัวอย่างของรูปหลายเหลี่ยมเว้าออกและรูป 1.11 ข) เป็นตัวอย่างของรูปหลายเหลี่ยมเว้าเข้า

ก) รูปหลายเหลี่ยมเว้าเข้า

ข) รูปหลายเหลี่ยมเว้าออก

รูป 1.11 ตัวอย่างของรูปหลายเหลี่ยม

เนื่องจากรูปหลายเหลี่ยมเกิดจาก การนำเอาส่วนของเส้นตรงมาต่อกันจนเกิดเป็นรูปปิดขึ้นการสร้างคำสั่งสร้างรูปหลายเหลี่ยมจึงเป็นสิ่งที่ไม่ยากเลยเพียงแค่กำหนดจุดมุมต่าง ๆ ของรูปหลายเหลี่ยมที่เราต้องการสร้างจากนั้นลากเส้นเชื่อมโยงจากจุดสองจุดที่เป็นจุดมุมที่ติดกันไปเรื่อยๆ จนครบทุกจุด ก็จะได้รูปหลายเหลี่ยมขึ้นมา คำสั่งในการสร้างรูปหลายเหลี่ยมมักจะมีรูปแบบดังนี้

Polygon (AX,AY,N)

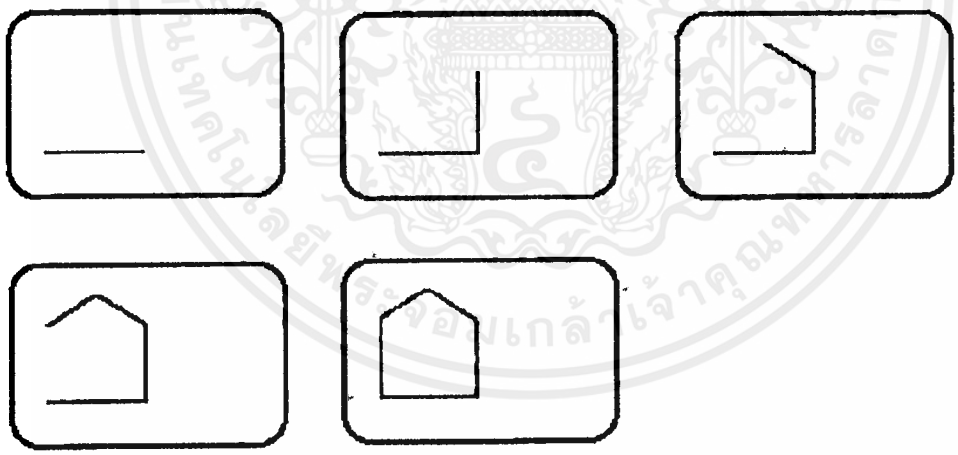
โดยที่ N คือจำนวนของจุดมุมของรูปหลายเหลี่ยม AX และ AY คือตัวแปรชนิดที่เป็นอาเรย์ (array)ขนาด 1 x N มิติ ซึ่งเก็บค่าโคออร์ดิเนท X และ Y ของจุดมุม N จุดของรูปหลายเหลี่ยมคำสั่ง Polygon จะเริ่มลากเส้นตรงจากจุดแรกไปยังจุดที่สอง และจากจุดที่สองไปยังจุดที่สาม และต่อ ๆ ไป จนกระทั่งถึงจุดที่ N จากนั้นลากเส้นตรงจากจุดที่ N กลับไปยังจุดแรกตัวอย่างเช่นเราต้องการสร้างรูป 5 เหลี่ยมรูปหนึ่ง ค่าที่เก็บไว้ในตัวแปร AX และ AY มีดังนี้

AX	AY	
0.1	0.1	จุดที่ 1 (0.1,0.1)
0.5	0.1	จุดที่ 2 (0.5,0.1)
0.5	0.5	จุดที่ 3 (0.5,0.5)
0.3	0.8	จุดที่ 4 (0.3,0.8)
0.1	0.5	จุดที่ 5 (0.1,0.5)

เราใช้คำสั่ง Polygon (Ax,Ay,5) เพื่อสร้างรูป 5 เหลี่ยมขึ้น การทำงานของคำสั่งPolygon จะเริ่มลากเส้นตรงจากจุดที่ 1 (0.1,0.1) ไปยังจุดที่ 2 (0.5,0.1) ลากต่อไปยังจุดที่3 (0.5,0.5) ลากต่อไปยังจุดที่ 4 (0.3,0.8) ลากต่อไปยังจุดที่ 5 (0.1,0.5) และลากเส้นตรงกลับไปหาจุดที่1 (0.1,0.1) อีกครั้ง ดังแสดงในรูป 1.12 เราสามารถแทนการทำงานของคำสั่ง Polygon (Ax,Ay,5) ด้วยคำสั่งที่กล่าวมาในหัวข้อที่แล้วได้ดังนี้

```
Move-to (0.1,0.1)     /* เลื่อน CP ไปยังจุดแรก */
Line-to  (0.5,0.1)
Line-to  (0.5,0.5)
Line-to  (0.3,0.8)
Line-to  (0.1,0.5)
Line-to  (0.1,0.1)
```

เราสร้างคำสั่ง Polygon นี้ได้จากคำสั่ง Move-to และ Line-to

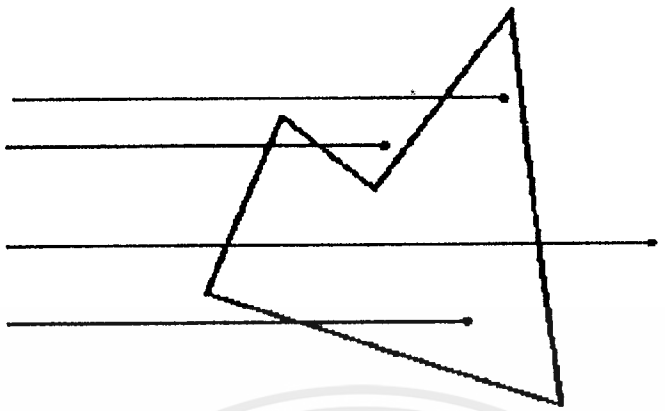


รูป 1.12 ตัวอย่างขั้นตอนการวาดรูป 5 เหลี่ยม

1.6 จุดภายในรูปหลายเหลี่ยม

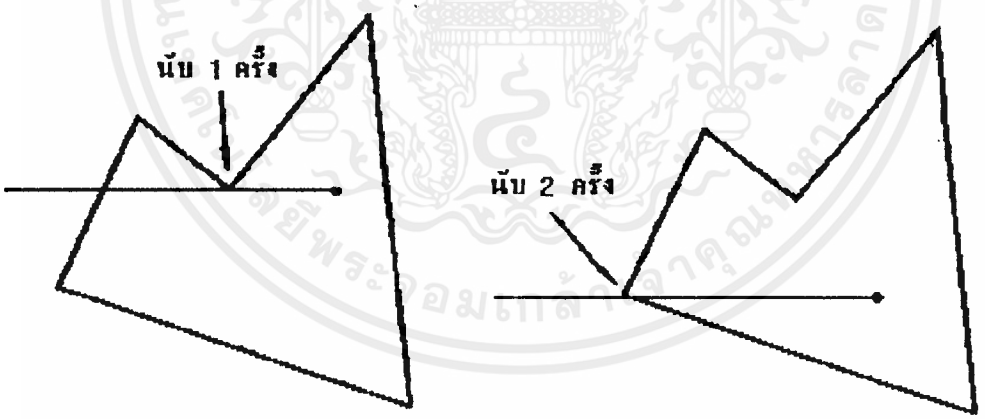
ในบางครั้งเราอาจต้องการตรวจสอบว่า ตำแหน่งของจุด ๆหนึ่งอยู่ภายในขอบเขตของรูปหลายเหลี่ยมหรือไม่ วิธีการที่จะตรวจสอบว่าจุด P'(x',y') อยู่ภายในขอบเขตของรูปหลายเหลี่ยมหรือไม่นั้น จะต้องสร้างเส้นตรงระหว่างจุด P' นี้กับจุด ๆ หนึ่งที่เราทราบแน่นอนว่าเป็นจุดที่อยู่ภายนอกรูปหลายเหลี่ยมจากนั้นให้นับจำนวนครั้งที่เส้นตรงนี้ตัดกับขอบของรูปหลายเหลี่ยมถ้าผลลัพธ์ที่ได้นับเป็นเลขจำนวนคี่จุด P' จะเป็นจุดที่อยู่ภายในรูปหลายเหลี่ยม แต่ถ้าผลลัพธ์เป็นจำนวนคู่จุด P' คือจุดที่อยู่นอกรูปหลายเหลี่ยม วิธีนี้เรียกว่า *วิธีตรวจสอบคู่-คี่ (Even-Odd method)*

วิธีที่ง่ายที่สุดในการหาจุดที่อยู่นอกรูปหลายเหลี่ยมคือ เลือกเอาจุดที่ค่าโคออร์ดิเนททางแกน X ของจุดนั้นน้อยกว่าโคออร์ดิเนททางแกน X ของจุดมุมทุกจุดของรูปหลายเหลี่ยม ดังเช่นแสดงในรูป 1.13



รูป 1.13 การใช้วิธีตรวจสอบคู่-คี่

การนับจำนวนที่เส้นตรงตัดกับขอบของรูปหลายเหลี่ยมอาจทำได้โดยตรวจสอบการตัดกันของส่วนของเส้นตรง 2 เส้น คือส่วนของเส้นตรงที่เราสร้างขึ้นนี้ กับส่วนของเส้นตรงที่เป็นขอบหรือด้านของรูปหลายเหลี่ยม ให้ตรวจสอบจนครบทุกด้านของรูปหลายเหลี่ยม เราก็จะทราบจำนวนครั้งที่เกิดการตัดกัน แต่ข้อผิดพลาดของวิธีนี้คือ ถ้าจุดที่ตัดกันเป็นจุดมุมของรูปหลายเหลี่ยม การนับจำนวนของการตัดกันก็จะผิดพลาดได้กล่าวคือ เราจะนับได้ว่าจุดตัดนี้เกิดขึ้นกับด้าน 2 ด้านของรูปหลายเหลี่ยมทำให้เรานับจำนวนที่ตัดกันเป็น 2 ครั้ง ซึ่งอาจทำให้เกิดผลที่ผิดพลาดได้ ดังตัวอย่างในรูป 1.14 ก) เป็นการนับที่ถูกต้องคือ ได้ผลลัพธ์เป็นเลขคี่ ส่วนในรูป 1.14 ข) จะได้ผลการนับที่ไม่ถูกต้อง คือได้ผลลัพธ์เป็นเลขคู่ ซึ่งที่จริงแล้วควรเป็นเลขคี่



ก) การนับให้ผลลัพธ์ถูกต้อง ข) การนับให้ผลลัพธ์ที่ผิด

รูป 1.14 ข้อผิดพลาดของวิธีตรวจสอบคู่-คี่

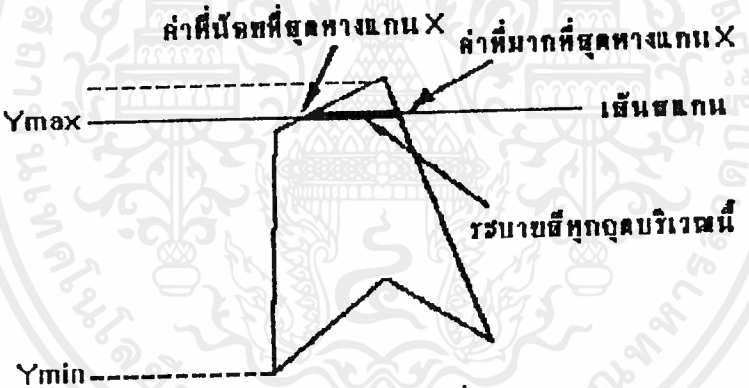
วิธีแก้ข้อผิดพลาดนี้คือถ้าจุดที่ตัดกันระหว่างด้านของรูปหลายเหลี่ยมกับเส้นตรงที่เราสร้างขึ้นคือจุดมุมของรูปหลายเหลี่ยม เราต้องตรวจสอบว่า ด้านทั้ง 2 ของรูปหลายเหลี่ยมที่เกิดจากจุดมุม (หรือจุดตัด) นี้ อยู่ด้านเดียวกันของเส้นตรงที่เราสร้างขึ้นหรือไม่ ถ้าอยู่ด้านเดียวกัน (ดังในกรณีของรูป1.14ก) ให้ถือว่าเกิดการตัด 2 ครั้ง แต่ถ้าไม่ได้อยู่ด้านเดียวกัน (ดังในกรณีของรูป 1.14 ข) ในถือว่าเกิดการตัดกันเพียง 1 ครั้งกับด้านทั้ง 2

1.7 การระบายรูปหลายเหลี่ยม

การระบายรูปหลายเหลี่ยมหมายถึงการเปลี่ยนสีหรือแอตตริบิ้วของบริเวณที่อยู่ภายในขอบเขตของรูปหลายเหลี่ยมให้เป็นมีสีและมีลวดลายตามที่ต้องการวิธีที่หนึ่งในการระบายรูปหลายเหลี่ยมคือ เริ่มต้นระบายจากจุด ๆ หนึ่ง ที่อยู่ภายในรูปหลายเหลี่ยม เราเรียกจุดนี้ว่า *จุดเริ่มต้น (seed)* เราจะเปลี่ยนแอตตริบิ้วหรือสีของจุดเริ่มต้นนี้ก่อนให้เป็นแอตตริบิ้วใหม่ที่ต้องการ จากนั้นค่อย ๆเปลี่ยนแอตตริบิ้ว ของจุดข้างเคียงรอบ ๆ และขยายออกไปเรื่อย ๆ จนถึงขอบของรูปหลายเหลี่ยมวิธีการระบายแบบนี้เรียกว่า *ฟลัดฟิวส์ (Flood Fill)* เราอาจใช้วิธีนี้ในการระบายพื้นที่อื่นๆ ได้เช่นกัน แต่มีข้อจำกัดคือต้องใช้กับบริเวณที่เป็นพื้นที่ปิดเท่านั้น ไม่เช่นนั้นการระบายจะระบายเลยออกไปทางช่องที่เปิดไว้ได้และนอกจากนั้นเราต้องระบุจุดเริ่มต้นให้กับมันด้วย

เราอาจระบายรูปหลายเหลี่ยมได้โดย ใช้วิธีตรวจสอบพิกเซลทุก ๆ พิกเซลบนจอภาพเพื่อดูว่าแต่ละพิกเซลอยู่ภายในรูปหลายเหลี่ยมที่เราต้องการระบายหรือไม่ถ้าอยู่ก็ให้เปลี่ยนสีของพิกเซลนั้นวิธีนี้เราไม่จำเป็นต้องใช้จุดเริ่มต้นอีกต่อไป เราใช้วิธีตรวจสอบจุดทุก ๆ จุดแทน แต่การทำงานจะช้ามากเพราะการตรวจสอบจุดภายในรูปหลายเหลี่ยมค่อนข้างเสียเวลา และยุ่งยากเราอาจปรับปรุงวิธีนี้ได้โดยการตรวจสอบเพียงแค่จุด หรือพิกเซลที่อยู่ภายในขอบเขตของรูปหลายเหลี่ยมได้ วิธีนี้เราเรียกว่า *สแกนไลน์อัลกอริทึม (Scan-line algorithms)*

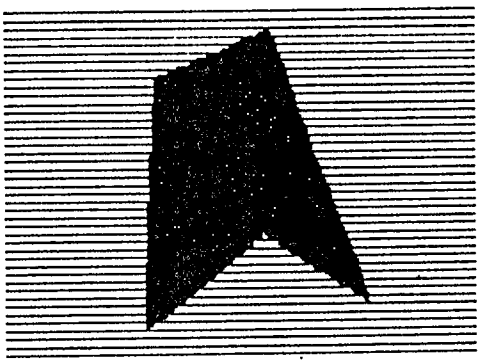
หลักการทำงานของสแกนไลน์อัลกอริทึมคือจะสร้างเส้นทดสอบหรือเส้นสแกน (Scan line)ซึ่งเป็นเส้นตรงขนานกับแกน X และหาค่า X ที่มากที่สุดและน้อยที่สุดบนเส้นสแกนนี้ จากนั้นเปลี่ยนสีของจุดทุก ๆ จุดบนเส้นสแกนที่อยู่ระหว่างค่า X ที่มากที่สุดและน้อยที่สุด ค่า X ที่มากและน้อยที่สุดนี้หาได้จากโคออร์ดิเนททางแกน X ของจุดตัดระหว่างเส้นสแกนกับเส้นขอบของรูปหลายเหลี่ยม ดังแสดงในรูป 1.15



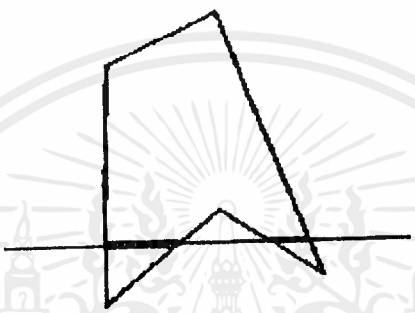
รูป 1.15 เส้นสแกนและส่วนที่จะถูกระบาย

เราจะสร้างเส้นสแกนนี้ตั้งแต่จุดสูงสุด จนไปถึงจุดที่ต่ำสุดของรูปหลายเหลี่ยม นั่นคือต้องหาค่ามากที่สุดและน้อยที่สุดทางแกนY ของรูปหลายเหลี่ยม ในรูป 1.15 ค่ามากที่สุดและน้อยที่สุดคือ Ymax และ Ymin เราต้องสร้างเส้นสแกนจาก Ymax ไปจนถึง Ymin หรือจาก Ymin ไปจนถึง Ymax ก็ได้ ในรูป 1.16 แสดงการระบายโดยใช้สแกนไลน์อัลกอริทึม

จุดที่ควรระวังของวิธีนี้คือกรณีที่มีการตัดกันของเส้นสแกนกับด้านของรูปหลายเหลี่ยมเกินกว่า2จุด (ดูรูป 1.17) ในกรณีนี้ต้องเปลี่ยนสีของจุดบนเส้นสแกนที่อยู่ภายในบริเวณของรูปหลายเหลี่ยมเท่านั้น



รูป 1.16 การระบายรูปหลายเหลี่ยมโดยใช้สแกนไลน์อัลกอริทึม



รูป 1.17 เส้นสแกนและส่วนที่ควรระบาย

บทที่ 2

การแปลง (Transformation)

สิ่งที่ทำให้คอมพิวเตอร์กราฟฟิค มีประโยชน์อย่างมากในการสร้างภาพก็คือ ความง่ายดายสะดวกสบายในการเปลี่ยนแปลงภาพที่เราวาดขึ้นมาแล้ว เช่น เปลี่ยนสเกลของกราฟเปลี่ยนมุมมองของแบบตึกที่ออกแบบไว้หรือการเปลี่ยนขนาดภาพของแผนที่ เป็นต้น ทั้งหมดนี้สามารถทำได้ง่ายรวดเร็ว โดยใช้คอมพิวเตอร์กราฟฟิค เพราะข้อมูลของรูปภาพต่าง ๆ หรือข้อมูลทางกราฟฟิค ได้ถูกป้อนเก็บไว้ในคอมพิวเตอร์ เพียงแค่เปลี่ยนแปลงวิธีที่จะแปลงข้อมูลเหล่านี้ออกมาเป็นภาพเท่านั้น เราก็จะได้ภาพในลักษณะที่ต้องการ การเปลี่ยนแปลงนี้อาศัยการคำนวณทางคณิตศาสตร์โดยเปลี่ยนค่าพารามิเตอร์ต่าง ๆที่เกี่ยวข้องกับการสร้างภาพเท่านั้น เราเรียกวิธีการนี้ว่า *การแปลง (Transformation)*

2.1 เมตริกซ์

โดยพื้นฐานแล้วภาพของระบบคอมพิวเตอร์กราฟฟิค ถูกสร้างขึ้นด้วยส่วนของเส้นตรงหรือเวคเตอร์หลาย ๆ เวคเตอร์มาประกอบกัน ซึ่งเราสามารถกำหนดเส้นตรงหรือเวคเตอร์เหล่านี้ได้ด้วยจุดปลายทั้งสองของมัน การที่จะเปลี่ยนแปลงลักษณะภาพที่วาด จำเป็นจะต้องใช้การคำนวณทางคณิตศาสตร์กระทำกับจุดต่าง ๆ เหล่านี้ วิธีหนึ่งที่จะช่วยให้เข้าใจได้ง่ายก็คือ การใช้เมตริกซ์เข้าช่วย แต่สำหรับผู้ที่ไม่ถนัดการใช้เมตริกซ์ก็อาจจะจำสูตรไปใช้งานได้เช่นกัน เราใช้เมตริกซ์ 2 มิติเพื่อช่วยทำการแปลง โดยแทนจุดต่าง ๆ ด้วยเมตริกซ์ขนาด 1x2 เช่น จุด (0.5,1) แทนด้วย เมตริกซ์ (0.5 1) และจุด (-5,0.3) แทนด้วย (-5 0.3) เป็นต้น เมื่อต้องทำการแปลงใด ๆก็เพียงแต่หาเมตริกซ์ที่เหมาะสมมาคูณกับเมตริกซ์ของจุด ผลลัพธ์ที่ได้คือเมตริกซ์ใหม่ ซึ่งก็คือจุดหรือตำแหน่งใหม่ของจุดปลายของเส้นตรงต่าง ๆ ทำให้เส้น หรือเวคเตอร์เหล่านี้มีลักษณะที่เปลี่ยนไป เมตริกซ์ที่นำมาคูณเพื่อเปลี่ยนตำแหน่งของจุดนี้เราเรียกว่า *เมตริกซ์การแปลง (Transformation matrix)*

2.3 การแปลงแบบซเกล

สมมติว่าเรามีจุด $P_1 = (x_1 \quad y_1)$ ซึ่งเป็นเมตริกซ์ขนาด 1x2 ถ้าเราคูณเมตริกซ์นี้ด้วยเมตริกซ์ T ที่มีขนาด 2x2 เราจะได้เมตริกซ์ขนาด 1x2 เมตริกซ์ใหม่ (P2) กลับว่าโดยที่

$$P_2 = (x_2 \quad y_2) = P_1T$$

การคูณเมตริกซ์นี้จะเป็นการคูณทางซ้ายและ T เป็นเมตริกซ์การแปลงที่จะแปลงจุด P_1 ไปเป็นจุด P_2 ถ้าเรานำเมตริกซ์การแปลง T นี้คูณเข้ากับจุดทุก ๆ จุด แล้วอะไรจะเกิดขึ้น คำตอบก็คือเราจะได้ภาพที่มีลักษณะเปลี่ยนแปลงไปแต่จะเปลี่ยนอย่างไรนั้นขึ้นอยู่กับค่าต่าง ๆ ในเมตริกซ์ T เช่นถ้าเราคูณด้วย *เมตริกซ์เอกลักษณ์ (Identity matrix)*

$$T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (2.1)$$

$$P_2 = P_1T = [x_1 \ y_1]\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$$

$$= [x_1 \ y_1] = P_1 \qquad (2.2)$$

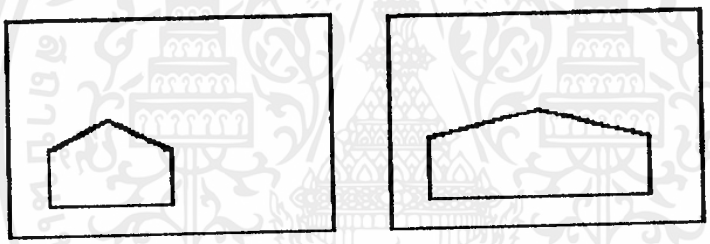
จุด P_2 กับ P_1 จะเป็นจุดเดียวกัน คือไม่มีการเปลี่ยนแปลงใด ๆ แต่ถ้าเราเลือกเมตริกซ์ T_1 เป็น

$$T_1 = \begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix} \qquad (2.3)$$

$$P_2 = P_1T_1 = [x_1 \ y_1]\begin{bmatrix}2 & 0\\ 0 & 2\end{bmatrix}$$

$$= [2x_1 \ y_1] \qquad (2.4)$$

ผลลัพธ์ที่ได้คือโคออร์ดิเนท X ของทุก ๆ จุด จะมีค่ามากขึ้นเป็น 2 เท่าของค่าเดิม เส้นตรงในแนวนอนก็จะมีความยาวเป็น 2 เท่าของภาพเดิม ภาพใหม่ที่เปลี่ยนไปจะมีความสูงเท่าเดิมแต่ขยายออกทางแนวนอนออกจากจุดกำเนิด (0,0) เป็น 2 เท่าจากของเดิม ดังตัวอย่างในรูป 2.1



ก่อนแปลง หลังแปลง

รูป 2.1 แสดงผลของการแปลงแบบสเกลโดยขยายออกทางแนวนอนเป็น 2 เท่า

ในทำนองเดียวกัน ถ้าเมตริกซ์การแปลง T_2 มีค่าเป็น

$$T_2 = \begin{bmatrix}0.5 & 0\\ 0 & 1\end{bmatrix} \qquad (2.5)$$

โคออร์ดิเนท X ของทุก ๆ จุด ก็จะลดลงเป็นครึ่งหนึ่งของค่าเดิม ภาพใหม่ที่ได้จะมีความสูงเท่าเดิม แต่จะถูกบีบให้แคบทางแนวนอนเป็นครึ่งหนึ่ง คราวนี้เราลองขยายภาพออกทางแนวนอนเป็น 2 เท่า แล้วบีบภาพใหม่นี้ให้แคบลงเป็นครึ่งหนึ่ง แน่นอนเราต้องได้ภาพที่มีขนาดเท่าเดิมกลับมา

$$P_1 = (P_1T_1)T_2 = P_1(T_1T_2) \qquad (2.6)$$

เราสามารถตรวจสอบคำตอบของเราได้โดย นำเมตริกซ์ T_1 และ T_2 มาคูณกัน

$$T_1T_2 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2*0.5 + 0*0 & 2*8 + 0*1 \\ 0*0.5 + 1*0 & 0*0 + 1*1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad (2.7)$$

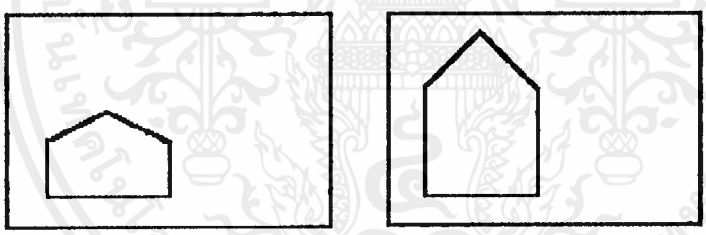
ผลลัพธ์ที่ได้คือเมตริกซ์เอกลักษณ์ นั่นคือเราจะได้ภาพเดิมก่อนเกิดการแปลงเราสามารถที่จะขยายภาพให้สูงขึ้นหรือกดภาพให้เตี้ยลงในแนวตั้งได้เช่นกัน โดยการหาเมตริกซ์การแปลงที่จะทำให้ค่าโคออร์ดิเนท Y เปลี่ยนไปโดยที่ค่า X ไม่เปลี่ยน ตัวอย่างเช่นเมตริกซ์การแปลง T_3

$$T_3 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad (2.8)$$

$$P_2 = P_1T_3 = [x_1 \; y_1]\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$= [x_1 \; 2y_1] \qquad (2.9)$$

จะได้ภาพที่มีความสูงเพิ่มขึ้นจากเดิม 2 เท่า ดังในภาพ 2.2

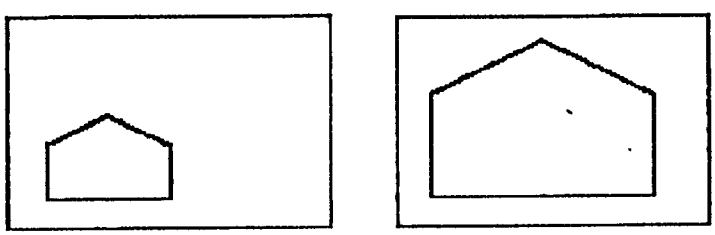


ก่อนแปลง หลังแปลง

รูป 2.2 แสดงผลของการแปลงแบบสเกลโดยเพิ่มความสูงขึ้นเป็น 2 เท่า

ถ้าเรานำเมตริกซ์ T_1 และ T_3 มาคูณกับ P_1 เราก็จะได้ภาพที่มีการขยายออกทั้งแนวนอนและแนวตั้งเป็น2 เท่าของภาพเดิมหรือกล่าวได้ว่าจะได้ภาพที่มีความโตเป็น 2 เท่า ดังในรูป 2.3

$$P_2 = P_1T_1T_2 = P_1(T_1T_3) \qquad (2.10)$$



ก่อนแปลง หลังแปลง

รูป 2.3 การแปลงภาพแบบสเกลเพื่อให้ได้ขนาดโตขึ้นเป็น 2 เท่าของภาพเดิม

เรานำเมตริกซ์ T_1 และ T_3 มาคูณกันเพื่อให้ได้เมตริกซ์ใหม่ T_4 เมตริกซ์เดียว

$$T_4 = T_1T_3 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \qquad (2.11)$$

เราอาจเขียนเมตริกซ์การแปลงแบบนี้ในรูปทั่ว ๆ ไป คือ

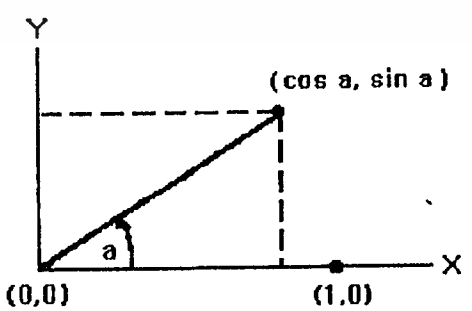
$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \qquad (2.12)$$

Sx เรียกว่า *สเกลแฟคเตอร์ (Scale Factor)* สำหรับโคออร์ดิเนท X ซึ่งจะมีผลต่อขนาดของภาพในแนวนอนหรือแนวแกน X Sy เรียกว่า *สเกลแฟคเตอร์สำหรับโคออร์ดิเนท Y* ซึ่งจะมีผลต่อขนาดของภาพในแนวตั้งหรือแนวแกน Y เมตริกซ์ S เป็นเมตริกซ์การแปลงซึ่งจะมีผลต่อขนาดและสัดส่วนของภาพเราเรียกว่า *เมตริกซ์การแปลงแบบสเกล (Scaling Transformation matrix)* การแปลงโดยการคูณด้วยเมตริกซ์ S เราเรียกว่าเป็น *การแปลง แบบสเกล (Scaling Transformation)* สังเกตว่าการสเกลภาพจะทำให้ทุก ๆ จุดมีการเปลี่ยนแปลง ยกเว้นเพียงจุดเดียวคือ จุดกำเนิด (0,0) ดังนั้นนอกจากขนาดของภาพจะเปลี่ยนไปแล้ว ตำแหน่งของจุดต่าง ๆ ก็เปลี่ยนไปด้วยถ้า Sx มีค่ามากกว่า 1 ก็จะทำให้ภาพเลื่อนไปทางขวา (สำหรับภาพที่อยู่ทางขวาของแกนY) และมีความกว้างมากขึ้นด้วย ถ้า Sx มีค่าน้อยกว่า 1 ภาพก็จะเลื่อนไปทางซ้ายและมีขนาดแคบลง ในทำนองเดียวกันถ้า Sy มีค่ามากกว่า 1 ก็จะทำให้ภาพขยายห่างออกจากแกน X และมีความสูงเพิ่มขึ้น และถ้า Sy มีค่าน้อยกว่า 1 ก็จะทำให้ภาพหดเข้าหาแกน X และเตี้ยลง

2.3 การแปลงแบบหมุน

การแปลงแบบหมุน (Rotation Transformation) เป็นวิธีการเปลี่ยนภาพโดยการหมุนจุด (หรือภาพ) ในทิศทางทวนเข็มหรือตามเข็มนาฬิกา โดยมีจุดศูนย์กลางของการหมุนอยู่ที่จุดกำเนิดสิ่งที่เราต้องการทราบก็คือเมตริกซ์ซึ่งมาคูณจุดที่เราต้องการหมุนเพื่อไปอยู่ที่ตำแหน่งใหม่ เราเรียกเมตริกซ์นี้ว่า *เมตริกซ์การแปลงแบบหมุน (Rotation Transformation Matrix)* สมมติว่าเราทำการหมุนจุด (1,0) ไปในทิศทางบวก (ทวนเข็มนาฬิกา) เป็นมุม a องศาตำแหน่งใหม่จะเป็นจุด (cos (a), sin (a)) (ดูรูป 2.4) ถ้าเมตริกซ์การแปลงแบบหมุนคือ

$$S = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad (2.13)$$



รูป 2.4 การหมุนทวนเข็ม

ดังนั้นจะได้ว่า

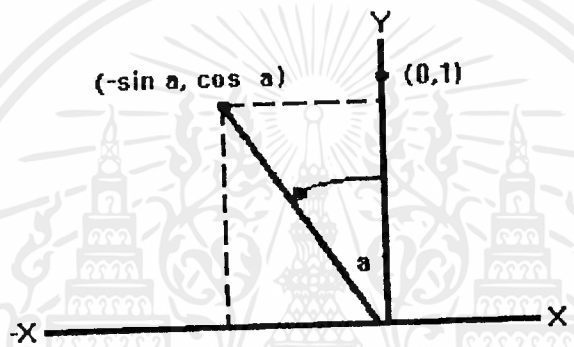
$$[\ \cos(a) \quad \sin(a)\ ] = [\ 1 \quad 0\ ]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$= [\ a \quad b\ ] \qquad (2.14)$$

ถ้าเราหมุนจุด (0,1) ในทิศทางบวกเป็นมุม a เช่นกัน นั้นคือใช้เมตริกซ์การแปลงตัวเดิมตำแหน่งใหม่จะเป็น (-sin (a), cos (a)) (ดูรูป 2.5) ดังนั้น

$$[\ -\sin(a) \quad \cos(a)\ ] = [\ 0 \quad 1\ ]\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$= [\ c \quad d\ ] \qquad (2.15)$$



รูป 2.5 การหมุนตามเข็ม

จากสมการ (2.13), (2.14) จะได้ว่า

$$\begin{aligned} a &= \cos(a) \\ b &= \sin(a) \\ c &= -\sin(a) \\ d &= \cos(a) \end{aligned} \qquad (2.16)$$

ดังนั้นเมตริกซ์การแปลงการหมุน (ทวนเข็ม) คือ

$$R = \begin{bmatrix} \cos(a) & \sin(a) \\ -\sin(a) & \cos(a) \end{bmatrix} \qquad (2.17)$$

ในสมการ (2.17) เป็นเมตริกซ์การแปลงที่ใช้กับการหมุนในทิศทางทวนเข็มนาฬิกาสำหรับการหมุนในทิศทางตามเข็มนาฬิกาคือ

$$R = \begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix} \qquad (2.18)$$

ตัวอย่างเช่นต้องการหมุนจุด $P_1(3,2)$ ในทิศทางทวนเข็มเป็นมุม 30 องศา เมตริกซ์การแปลงจะเป็น

$$\begin{bmatrix} \cos 30 & \sin 30 \\ -\sin 30 & \cos 30 \end{bmatrix} = \begin{bmatrix} 0.866 & 0.5 \\ -0.5 & 0.866 \end{bmatrix}$$

และจุดใหม่หลังการหมุน P_2 คือ

$$P_2 = [3 \quad 2] \begin{bmatrix} 0.866 & 0.5 \\ -0.5 & 0.866 \end{bmatrix}$$

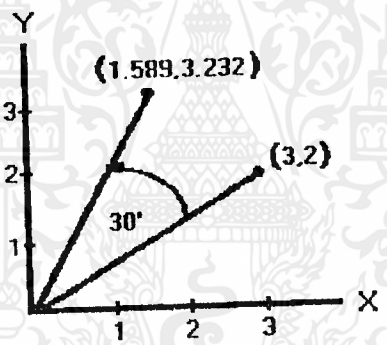
$$= [1.590 \quad 3.232]$$

2.4 การแปลงแบบย้ายและโฮโมจีเนียสโคออร์ดิเนท

การย้าย (Translation) เป็นการเลื่อนตำแหน่งของภาพทั้งภาพไปยังตำแหน่งอื่น ๆเป็นระยะทางเท่ากันทั้งหมดโดยขนาดของภาพไม่เปลี่ยนแปลงและไม่ทำให้ภาพเอียงไปจากแนวเดิมการย้ายนี้ทำได้โดยการบวกจุดทุก ๆจุดของภาพด้วยระยะทางที่ต้องการให้ภาพเลื่อนไปดังนั้น

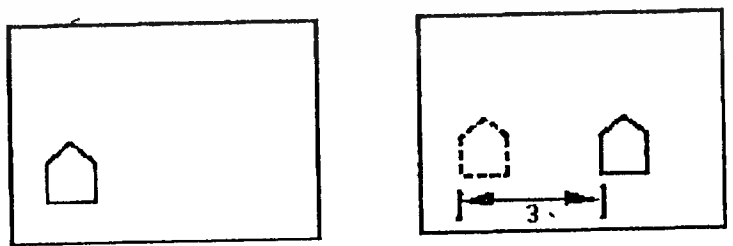
$$x2 = x1 + tx$$

และ $$y2 = y1 + ty \qquad (2.19)$$



รูป 2.6 การหมุนจุด (2,3) ในทิศทางทวนเข็มเป็นมุม 30 องศา

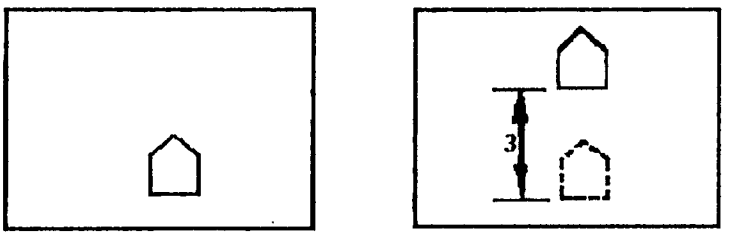
นั้นคือการย้ายตำแหน่งของภาพไปในแนวแกน X เป็นระยะ t_x และแนวแกน Y เป็นระยะทาง t_y หน่วย โดยที่ t_x และ t_y เป็นระยะทางที่ต้องการให้ภาพเลื่อนไปในแนวแกน X และแกน Y ตามลำดับ ตัวอย่างเช่น ต้องการย้ายภาพให้เลื่อนไปอยู่ทางขวา 3 หน่วย เราก็บวก 3 เข้ากับค่า X ของทุก ๆ จุดของภาพ ดังในภาพ 2.7



ก) ก่อนย้าย ข) หลังย้าย

รูป 2.7 การย้ายไปทางขวา

หรือถ้าต้องการย้ายภาพขึ้นข้างบน 3 หน่วย เราก็บวก 3 เข้ากับค่า Y ของทุก ๆ จุด ดังในรูป 2.8



ก) ก่อนย้าย ข) หลังย้าย

รูป 2.8 การย้ายภาพขึ้นข้างบน

ในทางตรงกันข้ามถ้าต้องการเลื่อนภาพไปทางซ้ายหรือลงข้างล่าง ก็นำจำนวนลบมาบวกเข้าไปเพื่อให้ค่า X หรือค่า Y ลงลด การย้ายภาพสามารถย้ายขึ้น-ลง, ไปซ้าย-ขวาได้พร้อมกัน โดยเปลี่ยนทั้งค่า X และ Y ของทุก ๆ จุดพร้อมกันจากที่กล่าวมาจะเห็นว่าการแปลงแบบย้ายนั้นไม่มีเมตริกซ์ที่มาใช้คูณเพื่อให้ได้ตำแหน่งใหม่ของจุดต่าง ๆ เราเพียงแค่นำค่าที่เหมาะสมมาบวกเข้ากับค่า X และ Y ของจุดต่าง ๆ เท่านั้น ลักษณะเช่นนี้ทำให้เรารวมการแปลงแบบย้ายเข้ากับการแปลงแบบสเกลหรือการแปลงแบบหมุนไม่ได้เราไม่สามารถสร้างเมตริกซ์เพียงเมตริกซ์เดียวมาคูณกับเมตริกซ์ของจุดต่าง ๆ เพื่อให้เกิดการแปลงหลาย ๆ แบบรวมกัน เกิดความไม่สะดวกในการทำงาน แต่ปัญหานี้เราสามารถแก้ไขได้โดยใช้ *โฮโมจีเนียสโคออร์ดิเนท (Homogeneous Coordinate)* เข้าช่วย โฮโมจีเนียสโคออร์ดิเนทจะใช้เมตริกซ์ขนาด 3x3 แทนเมตริกซ์ขนาด 2x2 ที่ได้กล่าวมาแล้ว จุดหรือตำแหน่งใหม่จะขึ้นอยู่กับจำนวน3จำนวนแทนที่จะเป็น 2 จำนวน (โคออร์ดิเนท X และ Y ของจุด) โดยเพิ่มโคออร์ดิเนทใหม่wขึ้นมา จำนวน 3 จำนวนนี้ได้แก่ ผลคูณของโคออร์ดิเนท X กับ W ผลคูณของโคออร์ดิเนทYกับ W และโคออร์ดิเนท W ดังนั้นโคออร์ดิเนทของจุดต่าง ๆ (x,y) จะถูกแทนด้วยจำนวน 3 จำนวนคือ (XW,YW,W) ถ้าเรามีโคออร์ดิเนทใหม่นี้ และต้องการทราบตำแหน่ง (x,y) เดิมของมัน ก็ทำได้โดยนำเอาโคออร์ดิเนทที่ 3 หารสองโคออร์ดิเนทแรกเท่านั้น โคออร์ดิเนท W นี้จะถูกใช้งานอย่างแท้จริงในการแปลงในระบบ 3 มิติ ส่วนกรณี 2 มิตินี้เราจะให้ค่า W เป็น 1 เสมอ

ในโฮโมจีเนียสโคออร์ดิเนท เมตริกซ์การแปลงแบบสเกลจะถูกเปลี่ยนจากเดิม

$$\begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$$

เป็น

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.20)$$

เราลองทดสอบโดยนำเมตริกซ์การแปลงของสมการ (2.20) มาคูณกับจุด $P_1(x,y)$ ซึ่งจะต้องแทนด้วยเมตริกซ์ (xw yw w)

$$P_2 = P_2S$$

$$= [xw \quad yw \quad w] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$P_2 = [S_xxw \quad S_yyw \quad w] \qquad (2.21)$$

เมื่อนำโคออร์ดิเนท W มาหารสองโคออร์ดิเนทแรก จะได้จุด P_2 เป็น $(S_x x, S_y y)$ ซึ่งกับการแปลงแบบสเกลที่กล่าวมาแล้ว

เมตริกซ์การแปลงแบบหมุน ในทิศทางทวนเข็มนาฬิกาจากเดิม

$$\begin{bmatrix} \cos(a) & \sin(a) \\ -\sin(a) & \cos(a) \end{bmatrix}$$

จะเปลี่ยนเป็น

$$R = \begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (2.22)$$

นำเมตริกซ์ในสมการ (2.22) คูณกับจุด $P_1(xw,yw,w)$ เราจะได้

$$P_2 = [\,xw \quad yw \quad w\,] \begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= [\,xcos\,a - ysin\,a \quad xcos\,a + ywcos\,a \quad w\,] \qquad (2.23)$$

ซึ่งจะได้จุด P_2 ที่ได้จากการหมุนจุด P_1 คือ (xcos (a) - ysin (a), xsin (a) + y cos(a)) ตรงกับการแปลงแบบหมุนที่กล่าวมาแล้ว สำหรับการแปลงแบบย้ายเมื่อต้องการเลื่อนภาพหรือจุดไปทางแนวนอน t_x และไปทางแนวตั้ง t_y จะได้เมตริกซ์การแปลงแบบย้ายคือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \qquad (2.24)$$

เราสามารถพิสูจน์ได้ว่าเมตริกซ์ในสมการ (2.24) ใช้งานได้โดยนำไปคูณกับจุด P_1 (xw,yw,w)

$$P_2 = P_1 T$$

$$= [\,xw \quad yw \quad w\,] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$$

$$P_2 = [\,xw+t_x w \quad yw+t_y w \quad w\,] \qquad (2.25)$$

เราจะได้จุด P_2ในโฮโมจีเนียสโคออร์ดิเนทเป็น P_2 $(xw+t_x w, yw+t_y w, w)$ ดังนั้นจุด P_2 ก็คือ $(x+t_x, y+t_y)$

2.5 การแปลงโคออร์ดิเนท

ที่กล่าวมาแล้วเราใช้การแปลงกับจุดต่าง ๆ แล้วได้ผลลัพธ์เป็นจุดใหม่ แต่เราอาจใช้การแปลงกับการแปลงระบบโคออร์ดิเนทได้ด้วยเช่นกันตัวอย่างเช่นระยะทางที่ใช้วัดเดิมมีหน่วยเป็นนิ้วก็สามารถแปลงเป็นหน่วยเซนติเมตร โดยมีระยะทางเท่าเดิมได้ด้วยการแปลงแบบสเกลการแปลงยังคงทำได้เหมือนเดิมเพียงแต่มีความหมายเปลี่ยนไปเท่านั้นนั่น

คือจุด ๆ หนึ่งในโคออร์ดิเนทหนึ่งเมื่อถูกคูณด้วยเมตริกซ์การแปลงก็ยังคงหมายถึงจุด ๆ เดิมที่อยู่ (หรือถูกวัด) ในโคออร์ดิเนทอื่น

การแปลงแบบย้ายก็สามารถใช้ในการแปลงโคออร์ดิเนทได้เช่นกันคือในกรณีที่จุดกำเนิดไม่ตรงกัน ตัวอย่างเช่น เราคิดว่ามุมล่างซ้ายของจอภาพมีโคออร์ดิเนท (0,0) แต่สำหรับจอภาพบางรุ่น ตำแหน่งนี้อาจหมายถึงพิกเซลที่มีโคออร์ดิเนท (100,319) ก็ได้ หรือในจอภาพบางแบบจะกำหนดจุด (0,0) ไว้ที่มุมบนซ้ายของจอภาพและค่าโคออร์ดิเนท Y ของแต่ละพิกเซลที่อยู่เหนือกว่า ที่เป็นเช่นนี้เพราะสาเหตุจากการสแกนของจอภาพที่เริ่มสแกนจากบนลงข้างล่าง และในการทำงานของเครื่องพิมพ์ปริ้นเตอร์ก็เช่นกันเริ่มพิมพ์จากส่วนบนของจอลงมาวิธีที่จะแปลงระบบโคออร์ดิเนทนี้ กับระบบโคออร์ดิเนทที่เราใช้กันธรรมดาทำได้โดยการสเกลค่า X ด้วย 1 และค่า Y ด้วย -1 เพื่อเปลี่ยนลำดับของเส้นสแกน และทำการแปลงแบบย้ายเฉพาะโคออร์ดิเนท Y ด้วยขนาดจอภาพในแนวตั้ง เพื่อให้จุดกำเนิดมาอยู่ในตำแหน่งที่ถูกต้อง

การแปลงแบบหมุนก็อาจใช้สำหรับการแปลงโคออร์ดิเนทได้เช่นกันแต่ส่วนมากแล้วจะเป็นการหมุนเป็นมุม 90 องศา ตัวอย่างของการแปลงแบบหมุน เพื่อเปลี่ยนโคออร์ดิเนท เช่น การใช้งานปริ้นเตอร์แบบดอตเมตริกซ์ซึ่งใช้กระดาษขนาด 8.5x11 นิ้ว จะมีแกน Y ไปตามแนวยาวของกระดาษและมีแกนXตามแนวขวางของกระดาษ กรณีที่ปริ้นเตอร์ทำงานในลักษณะนี้เราเรียกว่าเป็นการทำงานในโหมด *พรอตเทรต (Portait Mode)* ในบางครั้งเราอาจต้องการจัดให้แกน Y อยู่ในแนวขวางของกระดาษ และแกน X อยู่ตามแนวยาวของกระดาษ ในกรณีนี้เป็นการทำงานในโหมดที่เรียกว่าโหมด *แลนด์สเคป (Landscape Mode)* ซึ่งการใช้งานเหมาะกับภาพที่มีความยาวตามแนวนอนยาวกว่าแนวตั้งเราใช้การแปลงแบบหมุนด้วยมุม 90 และใช้การแปลงแบบย้ายเพื่อเปลี่ยนจุดกำเนิดให้เหมาะสม เพื่อให้ได้ภาพตามโหมดที่เราต้องการ (โหมดพรอตเทรตหรือแลนด์สเคป)

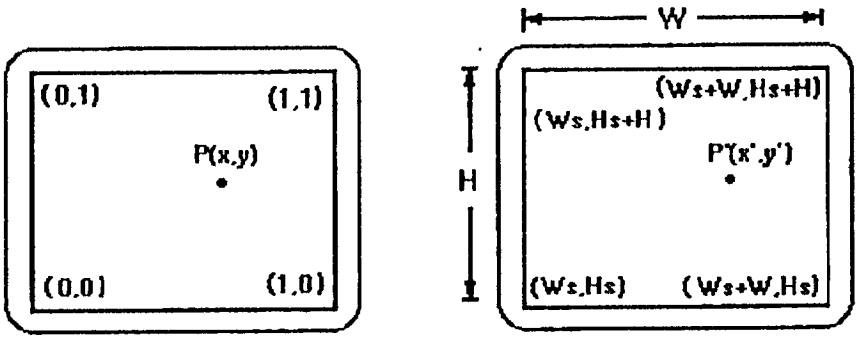
ตัวอย่างการแปลงโคออร์ดิเนทที่กล่าวมายังไม่ได้ยกตัวอย่างที่ใช้เมตริกซ์การแปลงตัวอย่างต่อไปนี้เป็นการแปลงจากโคออร์ดิเนทของอุปกรณ์มาตรฐาน ไปยังโคออร์ดิเนทของจอภาพจริง ๆโดยใช้เมตริกซ์การแปลงจุด P(x,y) บนโคออร์ดิเนทของอุปกรณ์มาตรฐาน จะถูกเปลี่ยนเป็นจุด P'(x',y) บนจอภาพซึ่งมีขนาดความกว้างตามแนวนอนWและสูง H โคออร์ดิเนทเริ่มต้นที่มุมล่างซ้ายเป็น (Ws,Hs) ดังนั้นจะได้ว่า

$$x' = xW + Ws \qquad (2.26)$$

$$y' = yH + Hs \qquad (2.27)$$

หมายความว่า เราทำการสเกลโคออร์ดิเนทXด้วยความกว้างของจอภาพ และสเกลโคออร์ดิเนท Y ด้วยความสูงของจอภาพ และทำการย้ายด้วยจุดเริ่มต้นของจอภาพ (ดูรูป 2.9) จากสมการ (2.26) และ สมการ (2.27) เราจะได้เมตริกซ์การแปลงโคออร์ดิเนทของจอภาพเป็น

$$D = \begin{bmatrix} W & 0 & 0 \\ 0 & H & 0 \\ W_s & H_s & 1 \end{bmatrix} \qquad (2.28)$$



โคออร์ดิเนทของอุปกรณ์มาตรฐาน โคออร์ดิเนทบนจอภาพ

รูป 2.9 การแปลงโคออร์ดิเนทของจอภาพ

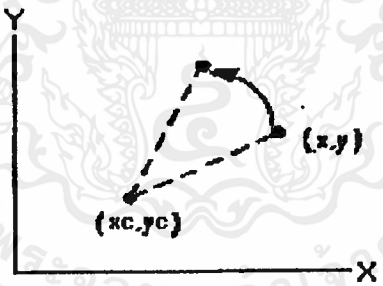
2.6 การหมุนรอบจุดใด ๆ

การแปลงแบบหมุนที่กล่าวมาเป็นการหมุนรอบจุดกำเนิด (0,0) เท่านั้น แต่ในหัวข้อนี้เราจะหาเมตริกซ์การแปลงแบบหมุนที่ใช้หมุนรอบจุดใด ๆ (x_c,y_c) (ดูรูป 2.10)

วิธีที่จะทำการหมุนแบบนี้มีขั้นตอนอยู่ 3 ขั้นตามลำดับ คือ

1. ทำการย้ายภาพเพื่อให้จุดศูนย์กลางของการหมุน (x_c,y_c) ไปอยู่ที่จุดกำเนิด
2. ทำการหมุนรอบจุดกำเนิด
3. ย้ายภาพเพื่อให้จุดศูนย์กลางของการหมุนกลับไปอยู่ในตำแหน่งเดิม

ในรูป 2.10 แสดงขั้นตอนการหมุนรอบจุด (x_c,y_c)

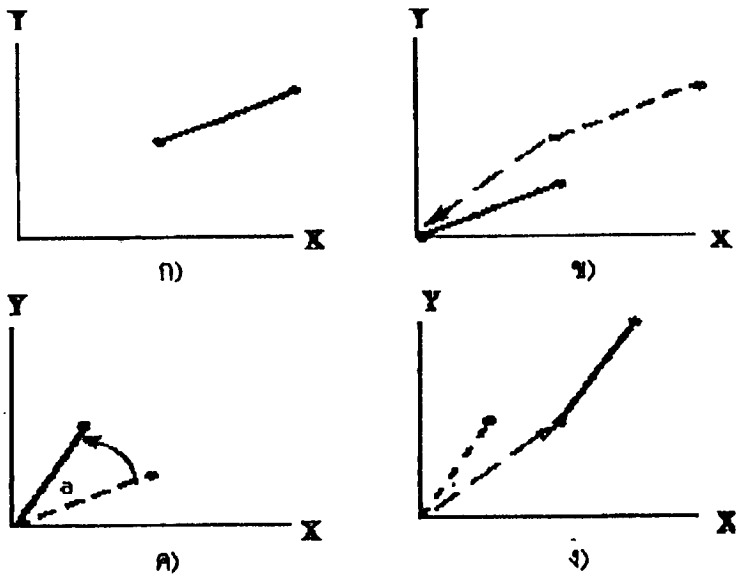


รูป 2.10 แสดงการหมุนรอบจุดใด ๆ

ขั้นตอนทั้ง 3 ขั้นนี้จะต้องทำทีละขั้นตอน เรียงลำดับให้ถูกต้อง เพราะในการทำแต่ละขั้นตอนต้องใช้เมตริกซ์การแปลงมาคูณเข้าไป และการคูณเมตริกซ์ไม่มีคุณสมบัติการสลับที่ ดังนั้นถ้าคูณเมตริกซ์การแปลงไม่เรียงลำดับ จะได้ผลลัพธ์ที่ไม่ถูกต้อง

เมตริกซ์ที่จะย้ายจุด (x_c,y_c) ไปยังจุดกำเนิดก็คือ

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



รูป 2.11 การหมุนรอบจุดใด ๆ

เมตริกซ์สำหรับการหมุนคือ

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

และเมตริกซ์ที่จะย้ายกลับไปยังจุดเดิมคือ

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_C & y_C & 1 \end{bmatrix}$$

นำเมตริกซ์ทั้ง 3 มาคูณเข้ากับจุดที่เราจะหมุน P(x,y) ได้จุดใหม่เป็น P1(x1,y1)

$$\begin{aligned} P_1 &= (\,(PT_1)R\,)\,T_2 \\ &= (\,P(T_1R)\,)\,T_2 \\ &= P\,(T_1RT_2) \end{aligned} \qquad (2.29)$$

ดังนั้นเมตริกซ์การแปลงแบบหมุนรอบจุด (x_c,y_c) คือ

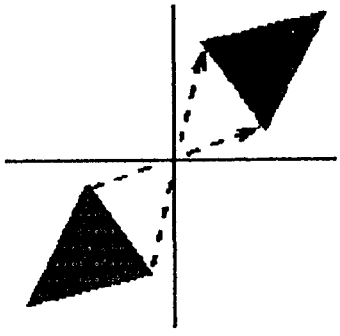
$$T_1RT_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_C & -y_C & 1 \end{bmatrix}\begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_C & y_C & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_C & -y_C & 1 \end{bmatrix}\begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ x_C & y_C & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos(a) & \sin(a) & 0 \\ -\sin(a) & \cos(a) & 0 \\ -x_C\cos(a)+y_C\sin(a)+x_C & -x_C\sin(a)-y_C\sin(a)+y_C & 1 \end{bmatrix} \qquad (2.30)$$

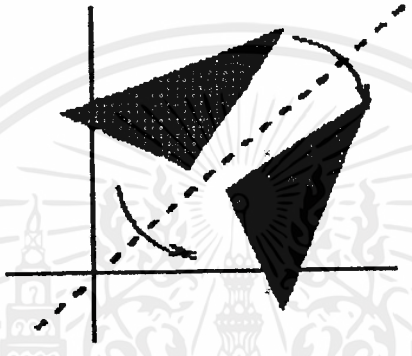
- *การสะท้อนกับจุดกำเนิด*

$$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$$



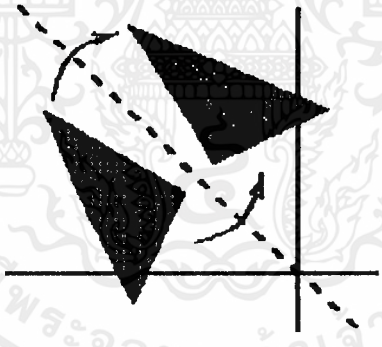
- *การสะท้อนกับเส้นตรง Y = X*

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$



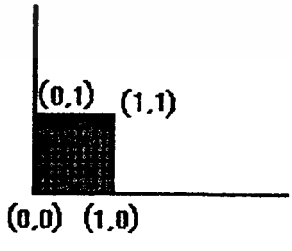
- *การสะท้อนกับเส้นตรง Y = -X*

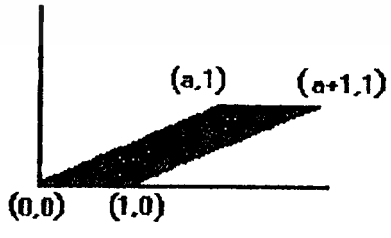
$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$



- *X shear*

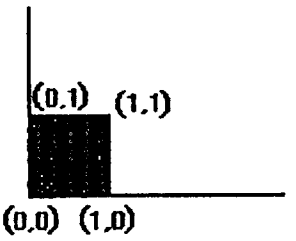
$$\begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}$$

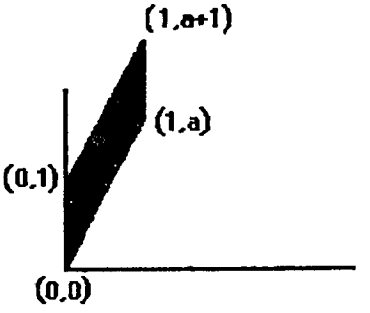




- *Y shear*

$$\begin{bmatrix} 1 & b \\ 0 & 1 \end{bmatrix}$$





2.8 การแปลงกลับ

เราได้เรียนรู้การแปลงแบบต่าง ๆที่ทำการแปลงจุด P(x,y) ไปเป็นจุด P'(x',y)มาแล้วในบางครั้งเราต้องการยกเลิกคำสั่งการแปลงนั้น ๆหรือต้องการทราบว่าจุด P'(x',y) นั้นถูกแปลงมาจากจุดใด ๆ ในกรณีเช่นนี้เราต้องทำการแปลงแบบเดียวกันกับการแปลงจุด P ไปเป็นจุด P' แต่ใช้เมตริกซ์การแปลงเมตริกซ์ใหม่ การแปลงจากจุด P' กลับไปเป็นจุด P เรียกว่า *การแปลงกลับ (Inverse Transformation)* เมตริกซ์ที่นำมาใช้ในการแปลงกลับก็คือ *อินเวอร์ส เมตริกซ์* ของเมตริกซ์การแปลงจากจุด P ไปเป็นจุด P'

สมมติว่า เราใช้เมตริกซ์การแปลง T แปลงจากจุด P ไปเป็นจุด P' ดังนั้น

$$P' = PT \qquad (2.33)$$

อินเวอร์สเมตริกซ์ของTเขียนแทนด้วย T^{-1} จากคุณสมบัติของอินเวอร์สเมตริกซ์ จะได้ว่า

$$TT^{-1} = T^{-1}T = I \qquad (2.34)$$

โดยที่ I คือ เมตริกซ์เอกลักษณ์

เราทำการแปลงจุด P' ด้วยอินเวอร์สเมตริกซ์ของเมตริกซ์การแปลง T ได้จุดใหม่เป็น P"

$$\begin{aligned} P'' &= P'T^{-1} \\ &= (P\ T)\ T^{-1} \\ &= PTT^{-1} \\ &= PI \\ &= P \end{aligned} \qquad (2.35)$$

จากสมการ (2.35) จะเห็นว่าถ้าเราใช้เมตริกซ์ T^{-1} แปลงจุด P' เราจะได้จุด P กลับคืนมา

อินเวอร์สเมตริกซ์ของเมตริกซ์

$$T = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad (2.36)$$

คือ

$$T^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$T^{-1} = \begin{bmatrix} \frac{d}{ad-bc} & \frac{-b}{ad-bc} \\ \frac{c}{ad-bc} & \frac{a}{ad-bc} \end{bmatrix} \quad (2.37)$$

ในกรณีของเมตริกซ์ขนาด 3x3

$$T = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ e & f & 1 \end{bmatrix} \quad (2.38)$$

จะได้อินเวอร์สเมตริกซ์ของ T คือ

$$T^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b & 0 \\ -c & a & 0 \\ cf-de & eb-af & ad-bc \end{bmatrix} \quad (2.39)$$

สำหรับการแปลงแบบสเกลที่มีเมตริกซ์เป็น

$$S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \quad (2.40)$$

จะมีอินเวอร์สเมตริกซ์เพื่อแปลงกลับ คือ

$$S^{-1} = \begin{bmatrix} 1/S_x & 0 \\ 0 & 1/S_y \end{bmatrix} \quad (2.41)$$

สำหรับการแปลงแบบหมุนที่มีเมตริกซ์เป็น

$$R = \begin{bmatrix} \cos(a) & \sin(a) \\ -\sin(a) & \cos(a) \end{bmatrix} \quad (2.42)$$

จะมีอินเวอร์สเมตริกซ์ คือ

$$R^{-1} = \begin{bmatrix} \cos(a) & -\sin(a) \\ \sin(a) & \cos(a) \end{bmatrix} \quad (2.43)$$

สำหรับการแปลงแบบย้ายที่มีเมตริกซ์เป็น

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \quad (2.44)$$

จะมีอินเวอร์สเมตริกซ์คือ

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -t_x & -t_y & 1 \end{bmatrix} \qquad (2.45)$$

บทที่ 3

การทำวินโดว์ (Windowing)

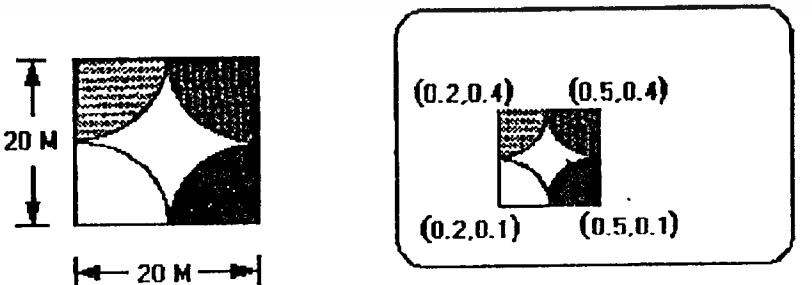
สำหรับสถาปนิกผู้ใช้คอมพิวเตอร์ออกแบบโครงสร้างของตึกทั้งอาคาร ถ้าเขาออกแบบแปลนต่าง ๆของตึกด้วยระบบคอมพิวเตอร์กราฟฟิค เขาสามารถให้คอมพิวเตอร์แสดงแบบแปลน ของอาคารบนจอภาพได้ตามความต้องการของเขาถ้าสถาปนิกผู้นี้ต้องการดูโครงสร้างในจุดเล็ก ๆ เช่น ห้องน้ำห้องหนึ่งภายในอาคารของตึกนี้เขาอาจจะมองเห็นห้องน้ำนี้เป็นเพียงจุดเล็ก ๆจากแบบแปลนของตึกทั้งอาคาร เพราะมันมีขนาดเล็กนิดเดียวเมื่อเทียบกับขนาดของตึก เขาอาจไม่สามารถเห็นรายละเอียดต่าง ๆของห้องน้ำได้ แต่โชคดีที่เขาใช้ระบบคอมพิวเตอร์ช่วยในการออกแบบเพราะเขาสามารถกำหนดขอบเขตของบริเวณที่สนใจแม้ว่าจะมีขนาดเล็กก็ตาม ให้อยู่ภายในกรอบสี่เหลี่ยม จากนั้นสั่งให้ระบบคอมพิวเตอร์ขยายภาพที่อยู่ภายในกรอบนั้น ให้ใหญ่มากขึ้นจนเห็นรายละเอียดต่าง ๆ ตามความพอใจได้ การกำหนดขอบเขตของบริเวณที่เราสนใจให้อยู่ภายในกรอบสี่เหลี่ยม และให้ระบบคอมพิวเตอร์แสดงภาพของวัตถุในบริเวณกรอบนั้น เรียกว่า *การทำวินโดว์ (Windowing)*

โดยมากเมื่อเรากำหนดกรอบที่ครอบคลุมบริเวณที่เราสนใจแล้ว เรามักจะให้คอมพิวเตอร์แสดงภาพของส่วนที่อยู่ภายในกรอบเท่านั้นและตัดภาพบริเวณที่อยู่นอกกรอบออกไป เพราะเป็นส่วนที่อยู่นอกเหนือความสนใจของเราการลบภาพบริเวณนอกกรอบที่กำหนดออกจากจอภาพนี้ เรียกว่า *การคลิป (Clipping)*

3.1 การแปลงภาพ

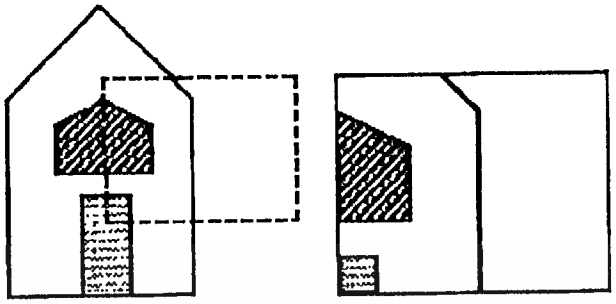
โดยทั่ว ๆ ไปแล้วในการวาดภาพของสิ่งใดก็ตามในระบบกราฟฟิค เราจะสร้างแบบจำลอง หรือโมเดลของสิ่งที่เราจะวาดขึ้นมา 2 ชนิดชนิดแรกคือเป็นแบบจำลองของวัตถุเรียกว่า *ออปเจ็กโมเดล (object model)* และชนิดที่สองเป็นแบบจำลองของภาพที่จะเกิดขึ้นจริงบนจอภาพ ออปเจ็กโมเดลนี้ไม่ได้มีอยู่จริงเป็นสิ่งที่เราสมมติขึ้น เราถือว่าวัตถุต่าง ๆ อยู่ในโลกของวัตถุ เรียกว่า *ออปเจ็กสเปซ (Object Space)* ในออปเจ็กสเปซขนาดของวัตถุต่าง ๆ จะมีหน่วยการวัด (ซึ่งส่วนมากหมายถึงความสูงความยาวต่าง ๆ) ที่เป็นหน่วยจริง ๆ ของวัตถุนั้นซึ่งอาจจะเป็นไมล์ นิ้ว เมตร ไมโครเมตร ปีแสง หรืออะไรก็ได้

แบบจำลองของภาพ หมายถึงภาพของวัตถุที่จะถูกวาดขึ้นบนจอภาพ ดังนั้นจึงมีหน่วยตามขนาดโคออร์ดิเนทของจอภาพชนิดนั้น ๆเช่น โคออร์ดิเนทของอุปกรณ์มาตรฐานก็จะมีความยาวอยู่ในช่วง 0 ถึง 1 เท่านั้น ภาพที่เกิดขึ้นของแบบจำลองบนจอภาพ เรียกว่าอยู่ใน *อิมเมจสเปซ(Image Space)* (ดูรูป 3.1) เราต้องทำการแปลงแบบสเกล เพื่อจะแปลงภาพหรือวัตถุจาก ออปเจ็กสเปซไปยังอิมเมจสเปซการสเกลจะลดขนาดของวัตถุที่มีขนาดใหญ่ ๆและขยายวัตถุขนาดเล็ก ๆ ให้มาอยู่ในช่วงขนาด 1 หน่วย (0 ถึง 1)



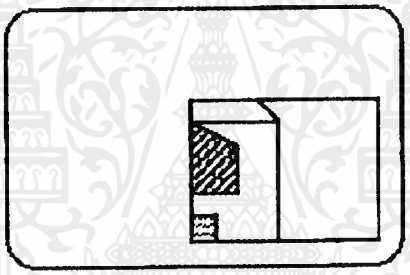
รูป 3.1 โคออร์ดิเนทในอิมเมจสเปซ

สำหรับภาพที่มีขนาดใหญ่หรือมีความซับซ้อนมาก แต่เราต้องการจะมองเพียงแค่บางส่วนของภาพ เราต้องวาดกรอบให้ครอบคลุมส่วนที่เรากำลังสนใจในออปเจ็กสเปซ แล้วแปลงภาพในกรอบไปยังอิมเมจสเปซ ดังตัวอย่างในรูป 3.2 กรอบที่เรากำหนดขึ้นนี้เรียกว่า *วินโดว์ (Window)*



รูป 3.2 การใช้กรอบเพื่อมองเฉพาะในส่วนที่เราต้องการ

ในบางโอกาส เราอาจไม่ต้องการให้ภาพภายในวินโดว์ปรากฏขึ้นเต็มทั้งจอภาพ แต่ให้ปรากฏอยู่บนบริเวณบางส่วนของจอภาพเท่านั้น ในกรณีนี้เราก็ต้องกำหนดขอบเขตที่จะแสดงภาพออกบนจอภาพ ขอบเขตของภาพบนจอนี้เรียกว่า*วิวพอร์ท (View port)* ดังแสดงในรูป 3.3

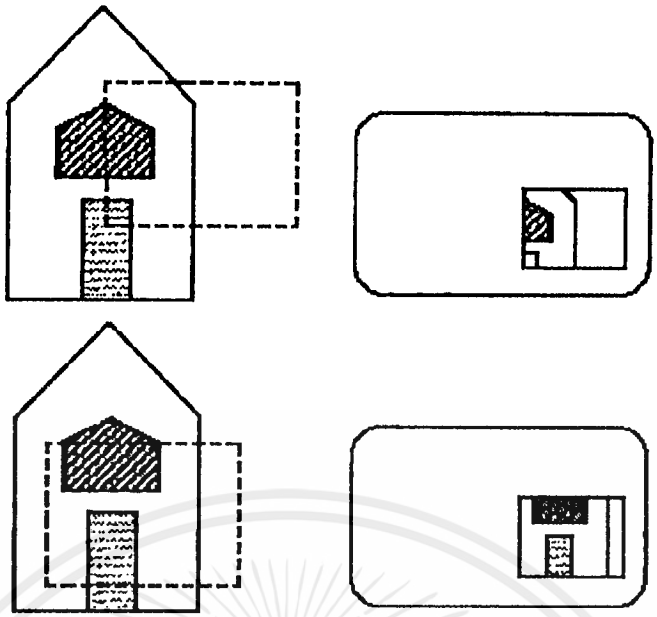


รูป 3.3 การกำหนดบริเวณที่จะแสดงภาพ (วิวพอร์ท)

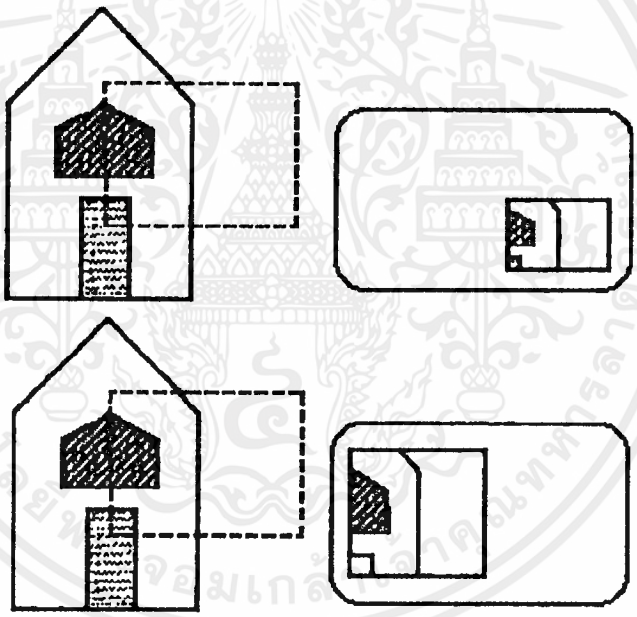
การกำหนดขอบเขตของวินโดว์และวิวพอร์ทเป็นอิสระต่อกัน การกำหนดขอบเขตของวินโดว์ต้องใช้โคออร์ดิเนทของออปเจ็กสเปซ ส่วนการกำหนดขอบเขตของวิวพอร์ทต้องใช้โคออร์ดิเนทของอิมเมจสเปซ การเปลี่ยนแปลงวินโดว์และวิวพอร์ทจะไม่มีผลซึ่งกันและกัน แต่จะมีผลต่อภาพที่จะเกิดขึ้นเท่านั้นเช่นถ้าเราเปลี่ยนตำแหน่งของวินโดว์ ภาพที่ปรากฏบนจอภาพก็จะเปลี่ยนไป แต่ยังคงอยู่ในบริเวณเดิม เพราะตำแหน่งของวิวพอร์ทยังคงเหมือนเดิม ดังแสดงในรูป 3.4 และเช่นเดียวกัน ถ้าเราเปลี่ยนแปลงวิวพอร์ท เราจะได้ภาพเดิมที่มีลักษณะต่างออกไปบนจอภาพ ดังแสดงในรูป 3.5

การกำหนดวินโดว์ และวิวพอร์ท จะทำให้เรามีข้อมูลเพียงพอในการแปลงภาพ จากออปเจ็กสเปซไปยังอิมเมจสเปซ ซึ่งทำได้โดยอาศัยการแปลงแบบย้ายและการแปลงแบบสเกล ดังต่อไปนี้

1. ทำการแปลงแบบย้ายวัตถุและวินโดว์ให้มุมล่างซ้ายของวินโดว์ไปอยู่ที่จุดกำเนิด
2. ทำการสเกลวัตถุและวินโดว์ให้มีขนาดกว้าง-ยาวเท่ากับวิวพอร์ทนั่นคือ การแปลงจากออปเจ็กสเปซ ไปยังอิมเมจสเปซ
3. ทำการแปลงแบบย้าย ให้วิวพอร์ทไปอยู่ในตำแหน่งที่ถูกต้องบนจอภาพ



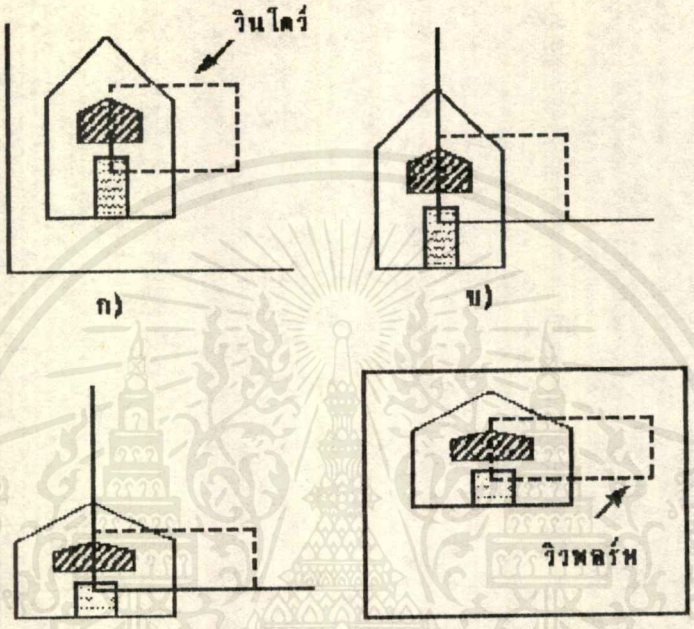
รูป 3.4 แสดงผลของการเปลี่ยนวินโดว์



รูป 3.5 แสดงผลของการเปลี่ยนวิวพอร์ท

การแปลงทั้งหมดที่กล่าวมาแสดงไว้ในรูป 3.6 และ เราเรียกการแปลงในลักษณะนี้ว่า *การแปลงภาพ (Viewing Transformation)* การแปลงภาพเป็นการกำหนดมุมของวัตถุ เพื่อให้ปรากฏบนจอภาพในตำแหน่งและขนาดที่เราต้องการ ในรูป 3.7 เป็นตัวอย่างแสดงขั้นตอนการแปลงภาพ

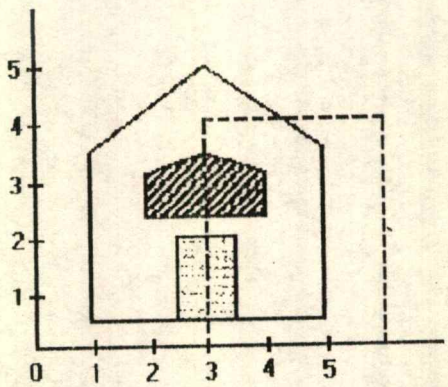
รูป 3.6 การแปลงภาพ



รูป 3.7 ตัวอย่างแสดงขั้นตอนการแปลงภาพ

ลองทำตัวอย่างการคำนวณการแปลงภาพดังนี้สมมติว่า เรากำหนดขอบเขตของวินโดว์ที่จุดโคออร์ดิเนท(3,0) (6,0)(6,4) และ (3,4) ในออปเจ็กสเปซ (ดังแสดงในรูป 3.8) เราจะได้เมตริกซ์การแปลงแบบย้าย คือ

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix}$$



รูป 3.8 แสดงตำแหน่งของวินโดว์

สมมติว่าวิวพอร์ทมีขอบเขตอยู่ระหว่าง 0.5 กับ 1.0 ทั้งแนวนอนและแนวตั้ง (กึ่งบนขวาของจอภาพ)ความกว้างของวินโดว์คือ 6-3 = 3 และความกว้างของวิวพอร์ทคือ 1.0-0.5 = 0.5 ดังนั้น สเกลแฟคเตอร์ในแนวแกน X เท่ากับ 0.5/3 = 0.167 ในทำนองเดียวกัน เราจะได้สเกลแฟคเตอร์ในแนวแกนYเท่ากับ 0.5/(4-0) = 0.125 ดังนั้นเมตริกซ์การแปลงแบบสเกลจะเป็น

$$\begin{bmatrix} 0.167 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

และเนื่องจากจุดเริ่มต้นของวิวพอร์ทอยู่ที่โคออร์ดิเนท (0.5,0.5) ดังนั้นเมตริกซ์การแปลงแบบย้ายคือ

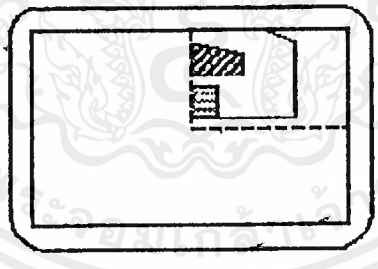
$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

จากรูป 3.6 เราจะได้เมตริกซ์สำหรับการแปลงภาพคือ

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.167 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.167 & 0 & 0 \\ 0 & 0.125 & 0 \\ 0 & 0.5 & 1 \end{bmatrix}$$

หลังจากใช้เมตริกซ์การแปลงนี้ก็จะได้ภาพดังรูป 3.9 เส้นประแสดงตำแหน่งและขอบเขตของวิวพอร์ท



รูป 3.9 ขอบเขตของวิวพอร์ทและผลของการแปลงภาพ

เมตริกซ์การแปลงภาพอาจเขียนในรูปทั่ว ๆ ไปได้ดังนี้

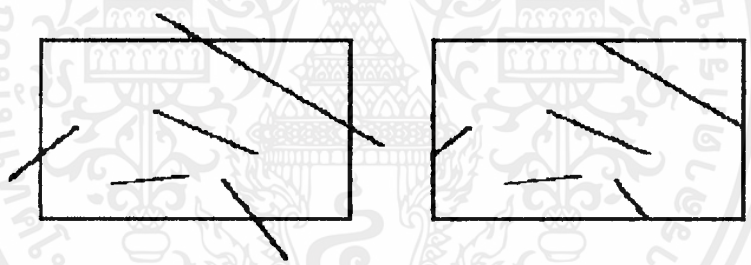
$$\begin{bmatrix} \frac{V_{xh}-V_{xz}}{W_{xh}-W_{xz}} & 0 & 0 \\ 0 & \frac{V_{yh}-V_{yz}}{W_{yh}-W_{yz}} & 0 \\ V_{x1}-\frac{W_{x1}(V_{xh}-V_{xz})}{(W_{xh}-W_{xz})} & V_{yz}-\frac{W_{yz}(V_{yh}-V_{yz})}{(W_{yh}-W_{yz})} & 1 \end{bmatrix}$$

โดยที่ V_{xh} คือ ค่าโคออร์ดิเนท X ของขอบทางด้านขวาของวิวพอร์ท
V_{xl} คือ ค่าโคออร์ดิเนท X ของขอบทางด้านซ้ายของวิวพอร์ท
V_{yh} คือ ค่าโคออร์ดิเนท Y ของขอบทางด้านบนของวิวพอร์ท
V_{yl} คือ ค่าโคออร์ดิเนท Y ของขอบทางด้านล่างของวิวพอร์ท
V_{xh} คือ ค่าโคออร์ดิเนท X ของขอบทางด้านขวาของวินโดว์
V_{xl} คือ ค่าโคออร์ดิเนท X ของขอบทางด้านซ้ายของวินโดว์
V_{yh} คือ ค่าโคออร์ดิเนท Y ของขอบทางด้านบนของวินโดว์
V_{yl} คือ ค่าโคออร์ดิเนท Y ของขอบทางด้านล่างของวินโดว์

การแปลงภาพนี้เราจะสังเกตเห็นว่า ถ้าสัดส่วนของวินโดว์กับวิวพอร์ทต่างกัน ภาพที่ได้ก็จะบิดเบือนไปตามสัดส่วนนั้น

3.2 การคลิป

การคลิปเส้นตรงจะต้องตรวจสอบดูว่า เส้นตรงเส้นนั้นอยู่นอกวินโดว์หรือไม่ หรืออยู่ภายในวินโดว์หรือพาดผ่านวินโดว์ถ้าเส้นตรงอยู่ภายในวินโดว์มันก็จะปรากฏอยู่บนจอภาพ ถ้าอยู่นอกวินโดว์ก็จะไม่ปรากฏถ้าเส้นตรงพาดผ่านวินโดว์เราต้องหาจุดตัดของเส้นตรงกับขอบวินโดว์ และตัดส่วนที่อยู่นอกวินโดว์ทิ้งไปเหลือไว้แต่ส่วนที่อยู่ภายในวินโดว์เท่านั้น รูป 3.10 แสดงตัวอย่างการคลิปเส้นตรง



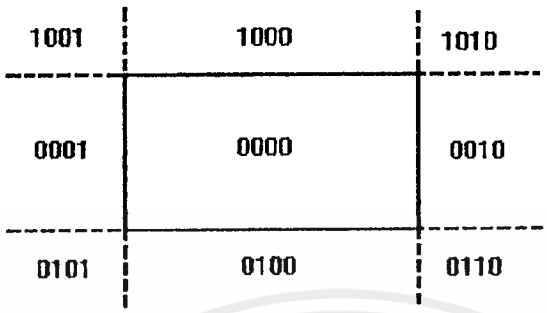
รูป 3.10 ผลของการคลิป

3.3 โคเฮน-ซัตเตอร์แลนด์ เอาต์โค้ด อัลกอริทึม

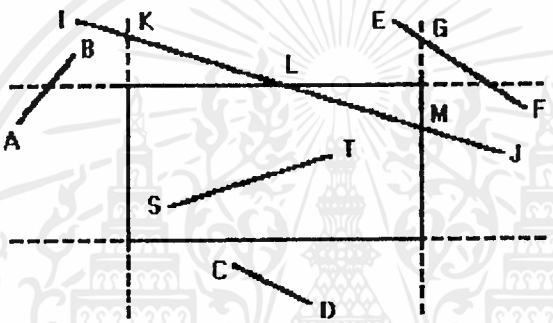
วิธีการที่ได้รับความนิยมมากที่สุดวิธีหนึ่ง ในการคลิปเส้นตรงคือ *โคเฮน-ซัตเตอร์แลนด์เอาต์โค้ด อัลกอริทึม (Cohen-Sutherlandoutcode algorithm)* อัลกอริทึมนี้สามารถกำจัดเส้นตรงที่อยู่ด้านใดด้านหนึ่งของขอบเขตการคลิปได้ดี คือปลายทั้งสองของเส้นตรงทั้ง 2 จุด จะอยู่ด้านบนด้านล่าง ด้านซ้าย หรือด้านขวาของวินโดว์เหมือนกัน วิธีการนี้ใช้บิตโอเปอร์เรชั่น (bitoperation) กระทำกับ *เอาต์โค้ด (outcode)* ของจุดในการตรวจเช็คอย่างมีประสิทธิภาพกล่าวคือที่จุดปลายของส่วนของเส้นตรงแต่ละจุดจะมีไบนารีโค้ด 4 บิต บิตแรก (ทางซ้ายสุด)จะถูกเซ็ตให้เป็น 1 ถ้าจุดนั้นอยู่เหนือขอบบนของวินโดว์ บิตที่สองถัดมาจะถูกเซ็ตเป็น 1 ถ้าจุดนั้นอยู่ทางขวาของวินโดว์ และบิตสุดท้ายจะถูกเซ็ตให้เป็น1 ถ้าจุดนั้นอยู่ทางซ้ายของวินโดว์ เส้นตรงที่ประกอบกันเป็นวินโดว์จะแบ่งพื้นที่ของจอภาพออกเป็น 9 ส่วน ซึ่งในแต่ละส่วนจะมีไบนารีโค้ดของส่วนนั้นอยู่ตามลักษณะที่กล่าวไว้ ดังแสดงไว้ในรูป 3.11

ดังนั้นถ้าเส้นตรงอยู่ภายในขอบเขตของวินโดว์ จุดปลายทั้งสองของเส้นตรง ก็จะมีเอาต์โค้ดเป็น 0000 (เส้นตรง ST ในรูป 3.12) เส้นตรงที่มีคุณสมบัติแบบนี้จะผ่านอัลกอริทึมนี้ไป (ไม่โดนคลิปออก) ถ้าเส้นตรงทั้งเส้นอยู่

ด้านหนึ่งของวินโดว์เช่นอยู่ด้านซ้ายของวินโดว์ จุดปลายทั้งสองก็จะมี1อยู่ในบิตเดียวกันของเอาต์โค้ด ดังนั้นเราสามารถตรวจสอบเส้นตรงที่อยู่ด้านใดด้านหนึ่งของวินโดว์ ได้โดยการใช้โอเปอร์เรชั่นลอจิคอล 'AND' กับเอาต์โค้ดของจุดปลายทั้งสองของเส้นตรงถ้าผลการ AND ไม่เป็นศูนย์ เส้นตรงเส้นนั้นก็จะถูกตัดทิ้งไป ตัวอย่างของเส้นตรงในกรณีนี้ได้แก่ เส้นตรง AB และเส้นตรง CD ในรูป 3.12



รูป 3.11 เอาต์โค้ดของแต่ละพื้นที่



รูป 3.12 การทดสอบและการแบ่งส่วนของเส้นตรง

ในกรณีของเส้นตรง EF และเส้นตรง IJ จะเป็นกรณีที่ยุ่งยากของอัลกอริทึมนี้ คือเป็นเส้นที่พาดผ่านเส้นตรงซึ่งประกอบกันเป็นขอบของการคลิปวิธีคลิปเส้นตรงประเภทนี้คือ หาจุดตัดของเส้นตรงนั้นกับเส้นตรงซึ่งประกอบกันเป็นขอบของการคลิป จากนั้นก็แบ่งเส้นตรงนี้เป็นเส้นตรงใหม่ 2 เส้น และก็ทดสอบเส้นตรงใหม่ทั้ง 2 นี้ ในลักษณะเดิมที่กล่าวมา เช่น เส้นตรง EF อาจถูกแบ่งเป็นเส้นตรงEGและ GF เส้นตรง EG จะถูกจัดว่าเป็นเส้นที่อยู่เหนือกว่าขอบของวินโดว์ทั้งเส้น และเส้นตรง GF ก็จะเป็นเส้นตรงที่อยู่ทางด้านขวาของวินโดว์เช่นกัน ดังนั้นเส้นตรง EG และ GF จะถูกตัดทิ้งไป ส่วนในกรณีของเส้นตรง IJ สมมติว่าถูกแบ่งเป็นเส้นตรง JK และเส้นตรง KJ เส้นตรง JK จะถูกตัดทิ้งไปเพราะอยู่ทางด้านซ้ายของวินโดว์ แต่เส้นตรง KJ ก็จะถูกแบ่งต่อไปสมมติว่าแบ่ง KJ ออกเป็นเส้นตรง KL และ LJ ก็จะถูกแบ่งต่อไปเป็นเส้นตรง LM และ MJ เส้นตรง MJ จะถูกตัดทิ้งไปเพราะอยู่ทางขอบของวินโดว์ เหลือเพียงเส้นตรง LM เท่านั้นที่ผ่านไปได้ จะเห็นว่าถ้าเส้นตรงลากผ่านเส้นที่ประกอบกันเป็นขอบของวินโดว์ การคลิปก็จะช้าเพราะจะต้องทำการคำนวณตัดแบ่งและทดสอบหลายครั้ง

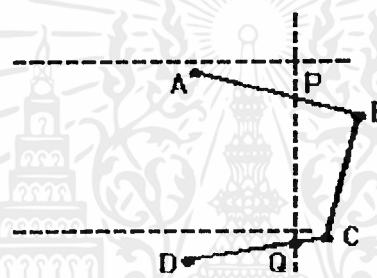
3.4 ซัตเตอร์แลนด์-ฮอด์แมน อัลกอริทึม

โคเฮน-ซัตเตอร์แลนด์ เอาต์โค้ด อัลกอริทึม ใช้ได้ดีกับการคลิปเส้นตรง แต่อัลกอริทึมของซัตเตอร์แลนด์และฮอดก์แมน (Suterland-Hodgman Algorithm) สามารถจะใช้กับการคลิปรูปหลายเหลี่ยมได้ด้วย หลักการของวิธีการนี้จะใช้การคลิปเส้นตรงกับขอบของวินโดว์ขอบใดก็ได้ การคลิปภาพจึงกระทำกับขอบของวินโดว์ทีละด้าน จนครบทั้ง 4 ด้าน

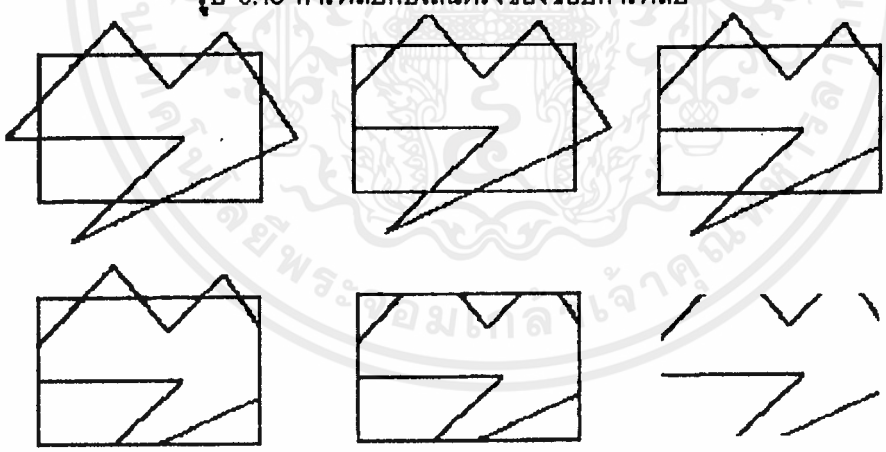
วิธีคลิปของอัลกอริทึมนี้จะทำไปตามคำสั่งในการวาดรูปหลายเหลี่ยม กล่าวคือ คำสั่งวาดรูปหลายเหลี่ยมจะ

ต้องเก็บตำแหน่งโคออร์ดิเนทของจุดมุมต่าง ๆของรูปหลายเหลี่ยมไว้ทั้งหมด โดยเก็บเรียงกันไปจากโคออร์ดิเนทของจุดหนึ่ง ลากเส้นตรงไปยังจุดถัดไป เพื่อวาดกรอบของรูปหลายเหลี่ยมเมื่อทำการคลิปเราจะต้องตรวจสอบดูว่าจุดถัดไปจุดใหม่นี้ จะทำให้เกิดเส้นตรงที่พาดผ่านขอบของวินโดว์หรือไม่ถ้าใช่จะต้องหาจุดตัดของเส้นตรงกับขอบของวินโดว์ (ขอบใดขอบหนึ่งเพียงขอบเดียวเท่านั้นเพราะอัลกอริทึมจะทำการคลิปทีละขอบ) จากนั้นก็จะทดสอบจุดต่าง ๆ ที่เหลือ ว่าจุดใดอยู่ด้านเดียวกับวินโดว์หรือไม่ เมื่อเทียบกับเส้นขอบของการคลิป (ไม่จำเป็นว่าจุดจะต้องอยู่ในวินโดว์เพียงแต่ตรวจสอบว่าอยู่ด้านเดียวกัน) ถ้าใช่ จุดทุกจุดที่อยู่กับด้านเดียวกับวินโดว์ และจุดตัดที่อยู่บนขอบของการคลิปจะถูกส่งผ่านไป นอกนั้นจะถูกตัดทิ้งหมด

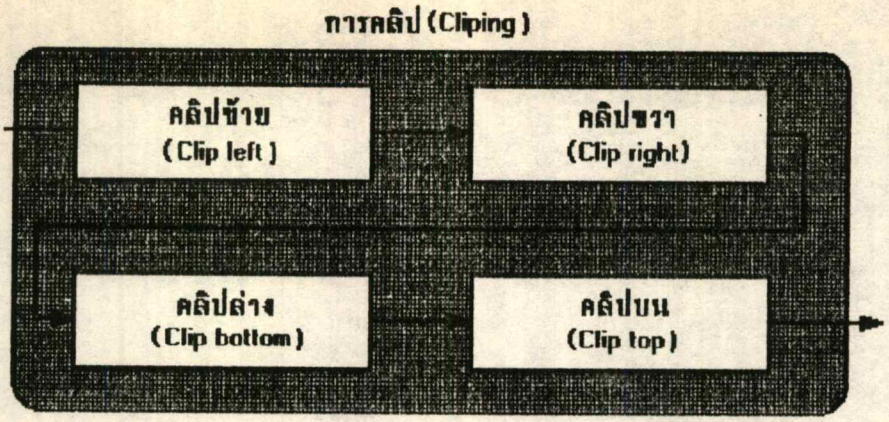
เช่นในรูป 3.13 ขอบของรูปหลายเหลี่ยมคือเส้นตรง AB BC และ CD แต่เมื่อผ่านการคลิปกับขอบด้านขวาของวินโดว์จุด BC จะถูกตัดทิ้งไป การวาดขอบ AB จะแทนจุด B ด้วย P กลายเป็นเส้นตรงAPส่วนเส้นตรง CD จะเหลือเพียงเส้นตรง DQ จะเห็นว่าแม้จุด D จะอยู่นอกวินโดว์ก็ยังคงไม่ถูกตัดทิ้งในขั้นนี้แต่เมื่อทำการคลิปกับขอบล่างแล้วเส้นตรง DQ จะถูกตัดออกไป ในรูป 3.13 นี้ช่วง PQ จะไม่ใช้คำสั่งลากเส้น แต่จะใช้คำสั่งย้ายจุดจากจุด P มาจุด Q เท่านั้นดังนั้นถ้ารูปหลายเหลี่ยมถูกคลิปออกบางส่วนผลที่ได้จะเป็นรูปหลายเหลี่ยมแบบเปิด ตัวอย่างการคลิปรูปหลายเหลี่ยมแสดงในรูป 3.14 และมีไดอะแกรมแสดงการคลิปในรูป 3.15



รูป 3.13 การคลิปกับเส้นตรงของขอบการคลิป

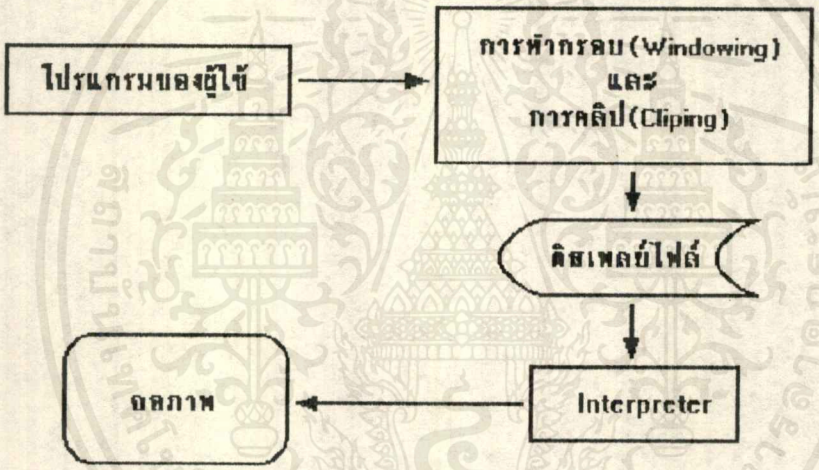


รูป 3.14 แสดงการคลิปโดยใช้ซัตเตอร์แลนด์-ฮอดก์แมน อัลกอริทึม



รูป 3.15 ขั้นตอนการทำการคลิปของซัตเตอร์แลนด์-ฮอดก์แมน อัลกอริทึม

ในระบบคอมพิวเตอร์กราฟฟิค ถ้าจะเพิ่มการคลิปเข้าไปในระบบจะต้องให้คำสั่งต่าง ๆ ของผู้ใช้ผ่านการทำกรอบและการคลิปเสียก่อนเพื่อที่จะแก้ไขคำสั่งต่าง ๆ ให้ได้ผลลัพธ์ถูกต้องตามความต้องการ จากนั้นจึงค่อยส่งผ่านไปเก็บในดิสเพลย์ไฟล์ต่อไป ขั้นตอนการทำงานของระบบจะเป็นไปตามรูป 3.16



รูป 3.16 การเพิ่มเติมการทำกรอบและการคลิปให้กับระบบ

บทที่ 4

คณิตศาสตร์สำหรับคอมพิวเตอร์กราฟิกในระนาบ 2 มิติ

4.1 TRANSFORMATION OF POINTS

เมื่อเราแทนจุด P ในระนาบ 2 มิติบนพิกัด (X,Y) แล้วแปลงจุด P เป็นจุด P' พิกัด (X',Y') ซึ่งเกิดจากการคูณจุด P กับเมตริกซ์ T ขนาด 2x2

$$T = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

$$P' = PT = (x,y)\begin{bmatrix} A & B \\ C & D \end{bmatrix} \qquad (4.1)$$

$$= [(Ax + Cy),(Bx + Dy)]$$

4.1.1 IDENTITY

ถ้า B = C = 0 และ A = D = 1 แล้วเมตริซ์ T จะเป็นเมตริกซ์เอกลักษณ์

$$(x',y') = (x,y)\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = (x,y) \qquad (4.2)$$

จะเห็นได้ว่าถ้าเป็น Identity transformation แล้วพิกัด P จะไม่เปลี่ยนแปลง

4.1.2 SCALING

กรณี D = 1 และ B = C = 0 จะเป็นการเปลี่ยน scale ในทิศทางของแกน X

$$(x',y') = (x,y)\begin{bmatrix} A & 0 \\ 0 & 1 \end{bmatrix} = (Ax,y) \qquad (4.3)$$

แต่ถ้า A = 1 และ B = C = 0 จะทำให้เปลี่ยน scale ตามแนวแกน y

$$(x',y') = (x,y)\begin{bmatrix} 1 & 0 \\ 0 & D \end{bmatrix} = (x,Dy) \qquad (4.4)$$

ถ้าเพียงแต่ B = C = 0 จะเกิดการเปลี่ยน scale ทั้งในทิศทางแกน X และ Y

$$(x',y') = (x,y)\begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix} = (Ax,Dy) \qquad (4.5)$$

4.1.3 REFLECTION

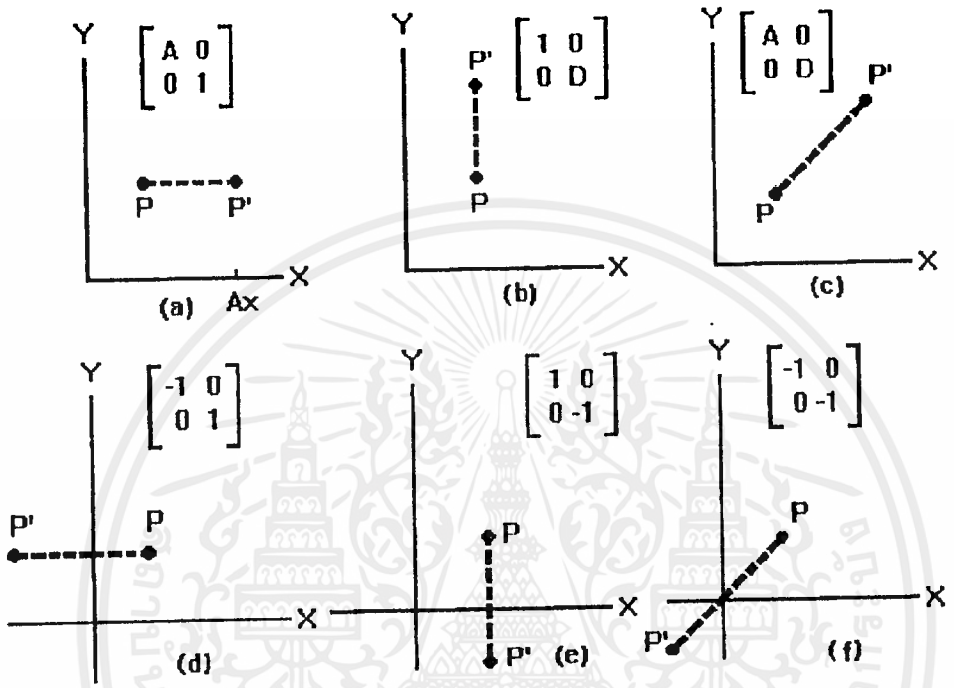
จะเกิดการ Reflection ก็ต่อเมื่อAหรือ D หรือทั้งคู่ มีเครื่องหมายตรงข้าม ในเมตริกซ์ T ถ้าเกิด Reflection ของจุดกับแกน X หรือแกน Y จะเกิดภาพสะท้อนของจุดบนฝั่งตรงกันข้ามของแกน

$$(x',y') = (x,y)\begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = (-x,y) \qquad (4.6a)$$

กรณีนี้จะเกิดการสะท้อนในแนวแกน X

$$(x',y') = (x,y)\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix} = (x,-y) \qquad (4.6b)$$

จะเกิดการสะท้อนในแนวแกน Y แต่ถ้า A = D = -1 และ B = C = 0 แล้วจะเกิดการสะท้อนกับจุด origin (0,0)



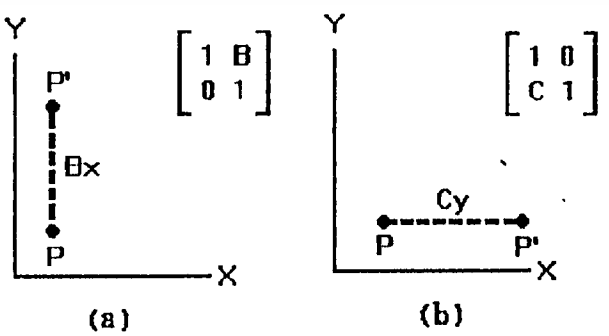
รูป 4.1 Transformation of points

4.1.4 SHEAR

เมตริกซ์ T ที่มี A = D = 1 และ C = 0

$$(x',y') = (x,y)\begin{bmatrix}1 & B\\ 0 & 1\end{bmatrix} = (x,Bx+y) \qquad (4.7)$$

เป็นการแปลงโดยที่พิกัดทางแกนXไม่เปลี่ยนแปลง ในขณะที่พิกัดทางแกน Y มีความสัมพันธ์เชิงเส้นกับXและ Y เรียกว่า Shear ในทิศทางแกน Y ถ้า Shear ในทิศทางแกน X แสดงดังนี้



รูป 4.2 ผลที่เกิดจาก shearing

$$(x',y') = (x,y)\begin{bmatrix} 1 & 0 \\ C & 1 \end{bmatrix} = (x+Cy,y) \qquad (4.8)$$

4.1.5 ROTATION

จุดสามารถหมุนเป็นมุม theta กับจุด origin สามารถแสดงในรูปเมตริกซ์ ดังนี้

$$(x',y') = (x,y)\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \qquad (4.9)$$

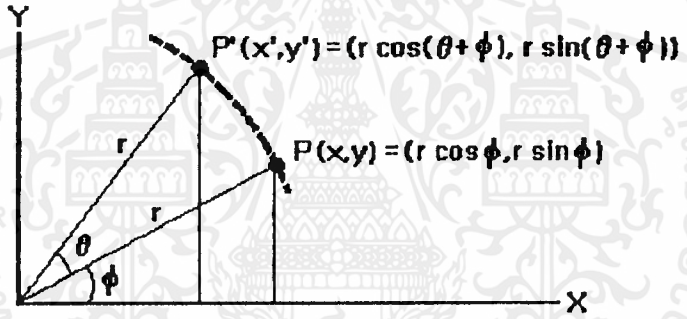
$$= (x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta)$$

ถ้ามุม θ เป็นบวกจะหมุนทวนเข็มนาฬิกา (counterclockwise) จากแกน X ไปยังแกน Y ถ้าต้องการหมุนตามเข็มนาฬิกา (clockwise) ใส่ -theta ลงไป

$$(x',y') = (x,y)\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \qquad (4.10)$$

$$= (x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta)$$

ได้จาก $\cos(-\theta) = \cos(\theta)$ $\sin(-\theta) = -\sin(\theta)$



รูป 4.3 การหมุนของจุดในระนาบ 2 มิติ

จากรูปจุด P หมุนเป็นมุมบวก เกิดจุดใหม่คือ P' โดยระยะจากจุด origin (0,0) คงที่คือระยะ r เพราะเป็นการหมุนเทียบกับจุด origin ที่จุด P แสดงได้ดังนี้

$$x = r\cos\phi$$

$$y = r\sin\phi$$

และ $x' = r\cos(\theta+\phi) = r\cos\phi\cos\theta - r\sin\phi\sin\theta$

$$= x\cos\theta - y\sin\theta$$

$$y' = r\sin(\theta+\phi) = r\cos\phi\sin\theta + r\sin\phi\cos\theta$$

$$= x\sin\theta + y\cos\theta$$

ซึ่งจะเหมือนกับสมการที่แสดงไว้ข้างบน

4.2 TRANSFORMATION OF LINES AND OBJECT

เส้นตรงเกิดจากการกำหนดจุดตำแหน่งของ vector ที่ปลายทั้งสองการทำ line transformationสามารถกระทำที่จุดปลายทั้งสองของ vector และลากเส้นตรงขึ้นใหม่ ระหว่างจุดที่เกิดขึ้นใหม่

object ก็ถูกสร้างขึ้นจากเส้นตรงหลายเส้นเราสามารถทำ transformation object โดยการ transform vertices แต่ละจุดบน object ถ้า object ประกอบขึ้นจาก n vertices รูปแบบการแปลงเป็นดังนี้

Transformated object coordinate **Original object coordinate** **Transformation matrix**

$$\begin{bmatrix} P'_1 \\ P'_2 \\ \cdot \\ \cdot \\ \cdot \\ P'_n \end{bmatrix} = \begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x'_n & y'_n \end{bmatrix} = \begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x'_n & y'_n \end{bmatrix} \qquad \begin{bmatrix} A & B \\ C & D \end{bmatrix}$$

4.2.1 SCALING

$$\begin{bmatrix} P'_1 \\ P'_2 \\ P'_3 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ x_3 & y_3 \end{bmatrix} \begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix} = \begin{bmatrix} x'_1 & y'_1 \\ x'_2 & y'_2 \\ x'_3 & y'_3 \end{bmatrix}$$

4.2.2 ROTATION

$$\begin{bmatrix} P'_1 \\ P'_2 \\ \cdot \\ \cdot \\ \cdot \\ P'_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_n & y_n \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \qquad (4.11)$$

4.2.3 TRANSLATION

วัตถุ (Object) สามารถtranslatedคือการ shift บน plane โดยการบวก magnitude กับแต่ละ vertex ของ Object ซึ่งมีรูปแบบ

Translated coordinate **Original coordinate** **Translation magnitude**

$$\begin{bmatrix} P'_1 \\ P'_2 \\ \cdot \\ \cdot \\ \cdot \\ P'_n \end{bmatrix} = \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ x_n & y_n \end{bmatrix} + \begin{bmatrix} T_x & T_y \\ T_x & T_y \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ T_x & T_y \end{bmatrix} \qquad (4.12)$$

4.3 HOMOGENEOUS COORDINATE SYSTEM

ใน homogeneous coordinate system จุด (x,y) แทนโดย (x,y,H) เมื่อ H เป็น scale ที่ไม่เท่ากับ0 normalized homogeneous coordinate แทนจุด (x,y) ด้วย (x/H,y/H,1) เมื่อ H เป็นค่าคงที่ไม่เท่ากับศูนย์ ปกติเรามักเลือก H = 1 จึงเขียนในรูป (x,y,1)

4.3.1 REVISED TRANSFORMATION MATRIX

ในระบบ homogeneous coordinate transformation matrix 2x2 จะต้องขยายเป็นเมตริกซ์ 3x3 ดังนี้

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \longrightarrow \begin{bmatrix} A & B & 0 \\ C & D & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.13)$$

ต้องการเพิ่มคุณสมบัติ translation เข้าไปในเมตริกซ์ 3x3 โดยการเปลี่ยนแถวที่ 3 จาก (0 0 1) เป็น (L M 1) ซึ่งจะมีผลกับการเคลื่อนที่เชิงเส้นของจุดทางแนวนอน L หน่วย และแนวตั้ง M หน่วย เมตริกซ์ที่เปลี่ยนแปลงจะอยู่ในรูป

$$T = \begin{bmatrix} A & B & 0 \\ C & D & 0 \\ L & M & 1 \end{bmatrix} \qquad (4.14)$$

ผลของการแปลงจุด (x,y) แสดงโดย

$$(x',y',1) = (x,y,1) \cdot T$$

หรือ

$$x' = Ax + Cy + L$$
$$y' = Bx + Dy + M$$

4.3.2 TRANSLATION

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ L_1 & M_1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ L_2 & M_2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ L_1+L_2 & M_1+M_2 & 1 \end{bmatrix} \qquad (4.15)$$

4.3.3 SCALING

$$\begin{bmatrix} A_1 & 0 & 0 \\ 0 & D_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} A_2 & 0 & 0 \\ 0 & D_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} A_1A & 0 & 0 \\ 0 & D_1D & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.16)$$

4.3.4 ROTATION

$$\begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 \\ -\sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_2 & \sin\theta_2 & 0 \\ -\sin\theta_2 & \cos\theta_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.17)$$

$$= \begin{bmatrix} \cos(\theta_1+\theta_2) & \sin(\theta_1+\theta_2) & 0 \\ -\sin(\theta_1+\theta_2) & \cos(\theta_1+\theta_2) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

4.3.5 INVERSES OF TRANSFORMATION

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -L & -M & 1 \end{bmatrix}, \quad T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ L & M & 1 \end{bmatrix}$$

เมตริกซ์ T1 และ T2 จะให้ผลที่ตรงกันข้าม T1 ทำให้เกิดการ translation(-L,-M)ขณะที่ T2 ทำให้เกิด translation (L,M) ถ้าคูณ T1 กับ T2 จะเกิด Identity transformation ซึ่งตำแหน่งของจุดไม่มีการเปลี่ยนแปลง

$$T_1T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -L & -M & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ L & M & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.18)$$

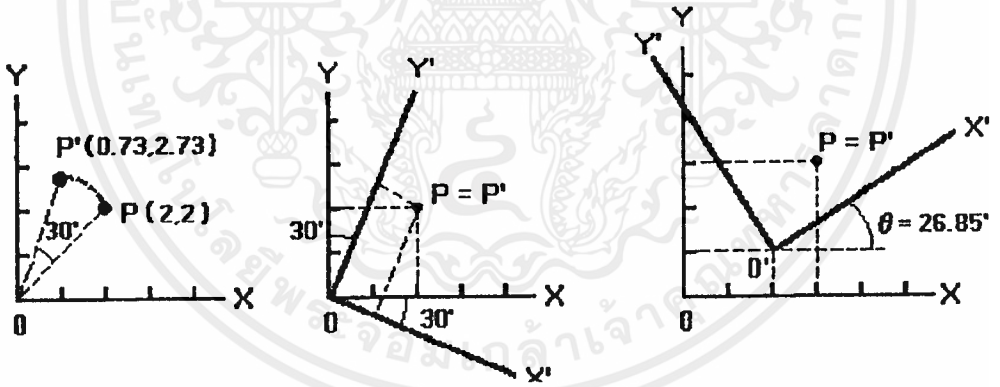
เช่นเดียวกับการหมุนที่มุมเท่ากัน แต่ตรงกันข้ามก็จะได้ Identity Matrix

เช่น $R(30) \cdot R(-30) = I$

4.3.6 TRANSFORMAING A COORDINATE SYSTEM

จากที่กล่าวมาเป็นการแปลง set of point ไปเป็น set of point ซึ่งอยู่ในระบบ coordinateเดียวกันซึ่งคล้ายกับการแปลง set of point โดยการเปลี่ยนระบบ coordinate

ถ้าจุด P หมุนไปทวนเข็มนาฬิกาเป็นมุม theta เป็นจุด P' ซึ่งพิกัดกับ P' บน coordinate ที่หมุนตามเข็มนาฬิกาเป็นมุม theta จะเป็นจุดเดียวกับ P พอดี ดังรูป



รูป 4.4 ความสัมพันธ์ระหว่าง (a) transforming a point,
(b) transforming a coordinate system
(c) จุด P กับ coordinate ทั้งสองระบบ

จากรูปที่ 4.4 (c) จุด P (3,3) บน coordinate XY จะมีตำแหน่งเดียวกัน แต่พิกัด (3.5,2.5)บน coordinate X'Y' การ translation coordinate จาก XY ไปยัง X'Y'เริ่มจากtranslateorigin ไป O' เปลี่ยน scale ทั้ง 2 แกน และสุดท้ายหมุน coordinate X'Y' ตามเข็มนาฬิกาเป็นมุม theta

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -2 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2\cos\theta & -2\sin\theta & 0 \\ 2\sin\theta & 2\cos\theta & 0 \\ -4\cos\theta - 2\sin\theta & 4\sin\theta - 2\cos\theta & 1 \end{bmatrix}$$

$$P' = (3,3,1).T = (2\cos\theta + 4\sin\theta, -2\sin\theta + 4\cos\theta, 1)$$

ถ้าเราทราบมุม θ ก็สามารถทราบตำแหน่ง P' จากตัวอย่าง P' (3.5,2.5)

$$2\cos\theta + 4\sin\theta = 3.5$$
$$-2\sin\theta + 4\cos\theta = 2.5$$

ปกติแล้ว ถ้าเรากำหนด T คือผลคูณของ n เมตริกซ์

$$T = T1.\ T2\ .....\ Tn\text{-}1\ .\ Tn$$
$$T\text{-}1 = Tn\text{-}1\ .\ Tn\text{-}1\text{-}1\ .....\ T2\text{-}1\ .\ T1\text{-}1$$

จากตัวอย่างจะได้

$$T^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 2 & 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1/2\cos\theta & 1/2\sin\theta & 0 \\ -1/2\sin\theta & 1/2\cos\theta & 0 \\ 2 & 1 & 1 \end{bmatrix}$$

$$P = P'.T^{-1} = (1.75\cos\theta - 1.25\sin\theta + 2, 1.75\sin\theta + 1.25\cos\theta + 1, 1)$$
$$= (3,3,1)$$

4.4 SEQUENTIAL 2-D TRANSFORMATION

4.4.1 หมุนรอบจุด

ขั้นที่ 1 ย้ายจุด center ของการหมุนไปที่ origin

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -q_1 & -q_2 & 1 \end{bmatrix}$$

ขั้นที่ 2 หมุนรอบ origin

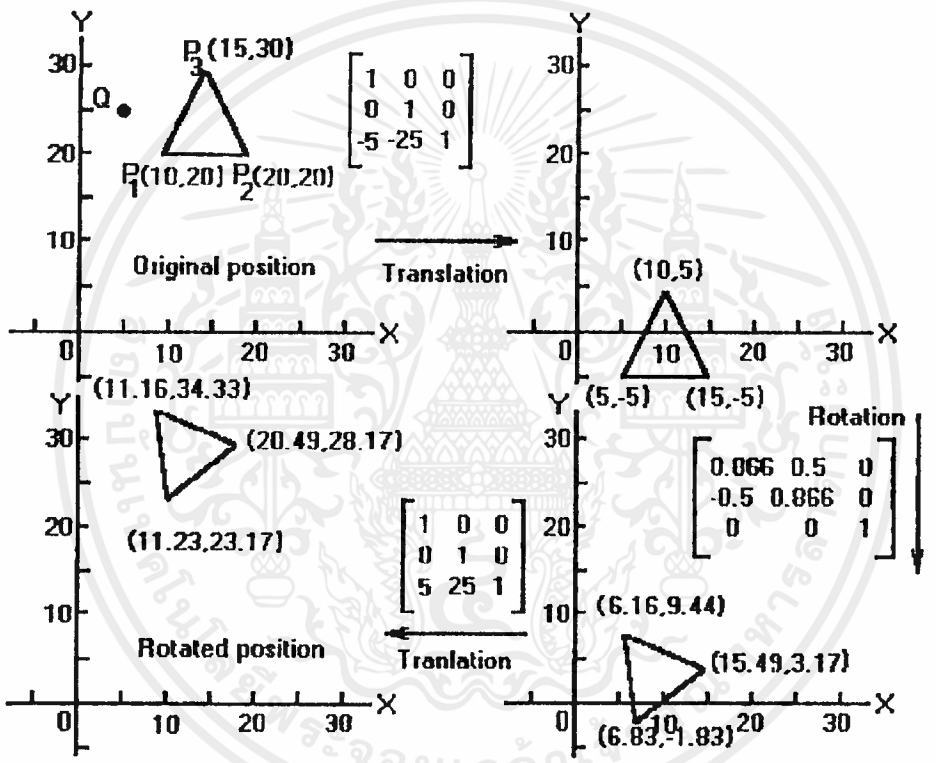
$$R(\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ขั้นที่ 3 ย้ายจุด Q ไปยังที่เดิม

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ q_1 & q_2 & 1 \end{bmatrix}$$

จะได้ $$T = T_1 . R(\theta) . T_2$$

$$= \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ q_1(1-\cos\theta) + q_2\sin\theta & q_2(1-\cos\theta) + q_1\sin\theta & 1 \end{bmatrix}$$



รูป 4.5 การหมุนรอบจุดในระนาบ 2 มิติ

4.4.2 REFLECTION กับแกนใด ๆ

แกนซึ่งมีสมการ y = a + bx

ขั้นที่ 1 ย้าย origin ไปยังพิกัด (0,a).

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

ขั้นที่ 2 หมุนรอบแกนตามเข็มนาฬิกาเป็นมุม θ จากสมการ $y = a + bx$ มุม theta สามารถคำนวณจาก slop จากแกนคือ b โดย cos = 1/r,sin = b/r โดยที่ $r = (b^2 + 1)^{1/2}$

$$R(-\theta) = \begin{bmatrix} 1/r & -b/r & 0 \\ b/r & 1/r & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ขั้นที่ 3 Reflect กับแกน X

$$T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ขั้นที่ 4 ทำตรงข้ามกับขั้นที่ 2

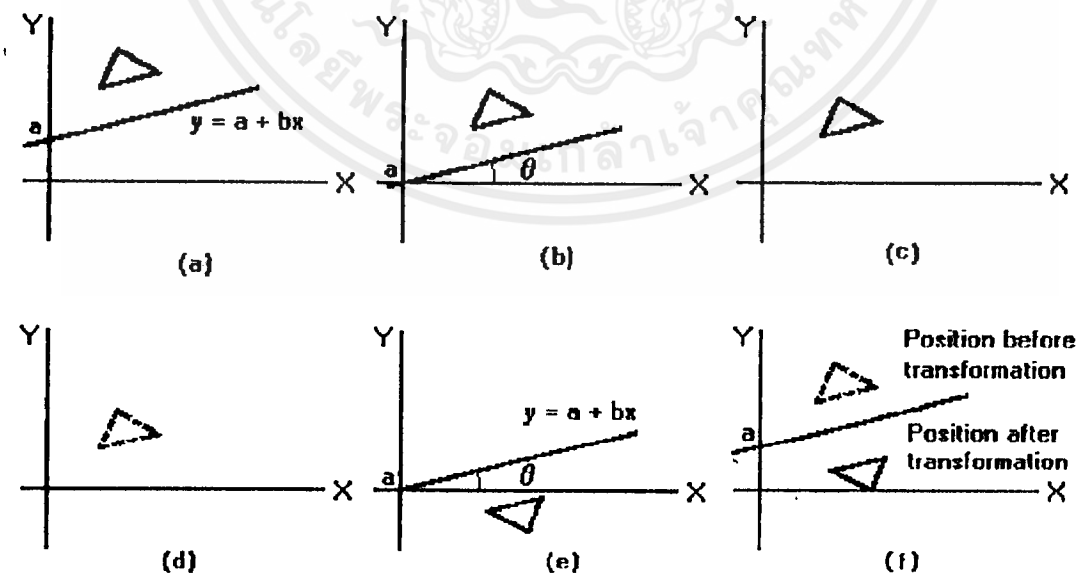
$$R(\theta) = \begin{bmatrix} 1/r & b/r & 0 \\ -b/r & 1/r & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

ขั้นที่ 5 ทำตรงข้ามกับขั้นที่ 1

$$T_1^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & a & 1 \end{bmatrix}$$

จะได้ transformation matrix ที่สมบูรณ์

$$T = T_1 R(-\theta) \cdot T_2 R(\theta) \cdot T_1^{-1}$$



รูป 4.6 ภาพละท้อนกับแกน (a) orginal figure, (b) translation, (c) rotation, (d) reflection, (e) rotation, (f) translation

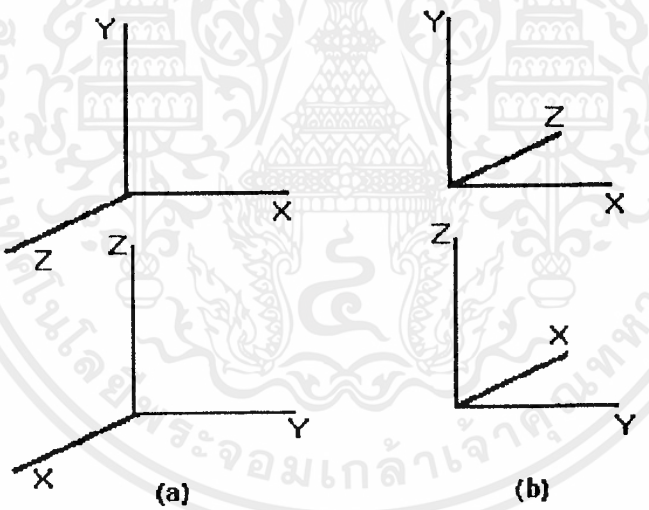
บทที่ 5

คณิตศาสตร์สำหรับคอมพิวเตอร์กราฟิกในระนาบ 3 มิติ

ในบทนี้จะกล่าวถึงเทคนิคที่ใช้ในกราฟิก 3 มิติ เช่น การวาด wire frame หรือ รูปเงาการกระทำกับภาพ 3 มิตินั้นจะต้องใช้เทคนิคที่ยุ่งยาก โชคร้ายที่จอภาพของคอมพิวเตอร์แสดงได้เพียง 2 มิติ ดังนั้นจะต้องทำการแปลงภาพวัตถุ 3 มิติให้อยู่ในรูป 2 มิติ ซึ่งจะทำได้โดยการแนะนำเรื่อง perspective projection และการแปลงวัตถุ 3 มิติให้อยู่ในระนาบ projection 2 มิติดังนั้นในบทนี้กล่าวถึงตั้งแต่เริ่มต้นถึง projection และการแปลงรูป 3 มิติ และจะอธิบายว่าจะต้องใช้คณิตศาสตร์ช่วยอย่างไร

5.1 ระบบโคออร์ดิเนท

ในโลกของ 3 มิติ เราจะต้องมีระบบโคออร์ดิเนท 3 มิติ โดยที่ระบบโคออร์ดิเนท 3 มิต สามารถแสดงให้อยู่ในระบบโคออร์ดิเนท 2 มิติได้ (ระนาบ XY) ซึ่งจะเพิ่มแกน Z ขึ้นมา ทำให้มีระนาบเพิ่มขึ้นอีก 2 ระนาบ คือ ระนาบ XZ และระนาบ YZ คำแนะนำสองข้อที่เป็นไปได้เมื่อเราเลือกที่จะเพิ่มแกน Z ขึ้นมา คือ ถ้าจุดในแกน Z พุ่งเข้าหาตา ระบบโคออร์ดิเนทที่เกี่ยวข้องที่ถูกอ้างถึงคือ กฎมือขวา (rigth-hand system) แต่ถ้าระบบโคออร์ดิเนทที่แกน Z พุ่งออกไปจะใช้กฎมือซ้าย (left-hand system) ซึ่ง 2 กฎนี้จะแสดงให้เห็นได้ดังรูปที่ 5.1 ซึ่งโดยทั่วไปแล้วจะใช้กฎมือขวามากกว่า



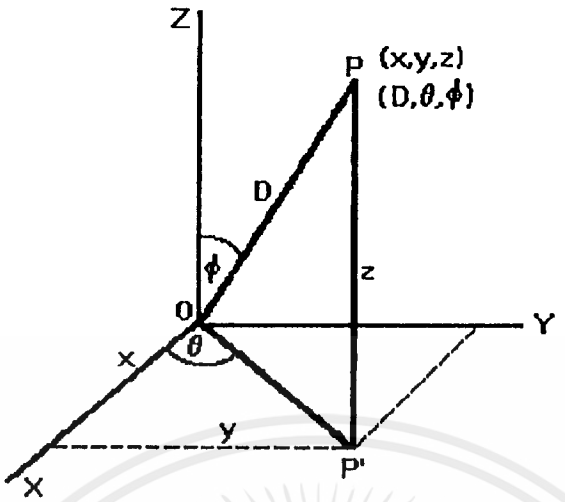
รูป 5.1 ระบบ coordinate (a) right-hand system (b) left-hand system

ระบบโคออร์ดิเนทที่แทนจุดในระบบ3มิตินั้นมีอยู่ 2 ระบบคือ ระบบโคออร์ดิเนทแบบ ตั้งฉาก(rectangular coordinate system) และระบบโคออร์ดิเนทแบบ ทรงกลม (spherical coordinate system)

Rectangular coordinate system จุดจะถูกกำหนดโดยตัวเลข 3 ตัวซึ่งเป็นตัวเลขในแกน X, Y และ Z โดยที่จะเป็นการบอกระยะทางของจุดในแต่ละแกน (ดูรูปที่ 5.2)

Spherical coordinate system ในระบบนี้แต่ละจุดจะถูกแทนด้วยตัวเลข3 ตัวคือ (D, θ, ϕ) ระยะทางจากจุด P ถึง origin คือ D มุมระหว่างเส้น OP และแกน Z ที่มีค่าเป็นบวก คือ theta มุมระหว่างแกน X ที่มีค่าเป็นบวกและเส้น OP' (projection ของ OP บนระนาบ XY) คือ theta มุมนี้จะถูกวัดในทิศทางทวนเข็มนาฬิกาจากแกน X

5.1.2 ความสัมพันธ์ระหว่างระบบโคออร์ดิเนตแบบ Rectangular และ Spherical



รูป 5.2 ความสัมพันธ์ระหว่างระบบโคออร์ดิเนตแบบ Rectangular และ Spherical

เพราะระบบโคออร์ดิเนททั้ง 2 ถูกใช้ในระบบคอมพิวเตอร์กราฟิก ดังนั้นจำเป็นที่จะต้องแปลงไปมาระหว่าง 2 ระบบ ซึ่งเราสามารถสร้างตามความสัมพันธ์ระหว่างระบบโคออร์ดิเนท 2 ระบบดังนี้

$$\begin{aligned} x &= D \cdot \sin\phi \cos\theta \ , \\ y &= D \cdot \sin\phi \sin\theta \ , \\ z &= D \cdot \cos\phi \end{aligned} \qquad (5.1)$$

และ

$$\begin{aligned} D &= (x^2 + y^2 + z^2)^{1/2} , \\ \theta &= \tan^{-1}(y/x) \ , \\ \phi &= \cos^{-1}(z/D) \end{aligned}$$

สำหรับตัวอย่าง , จุด (4,2,3) ใน rectangular coordinate system จะเท่ากับจุด (5.4,27,56) ใน spherical coordinate system

5.2 TRANSFORMATION MATRIX

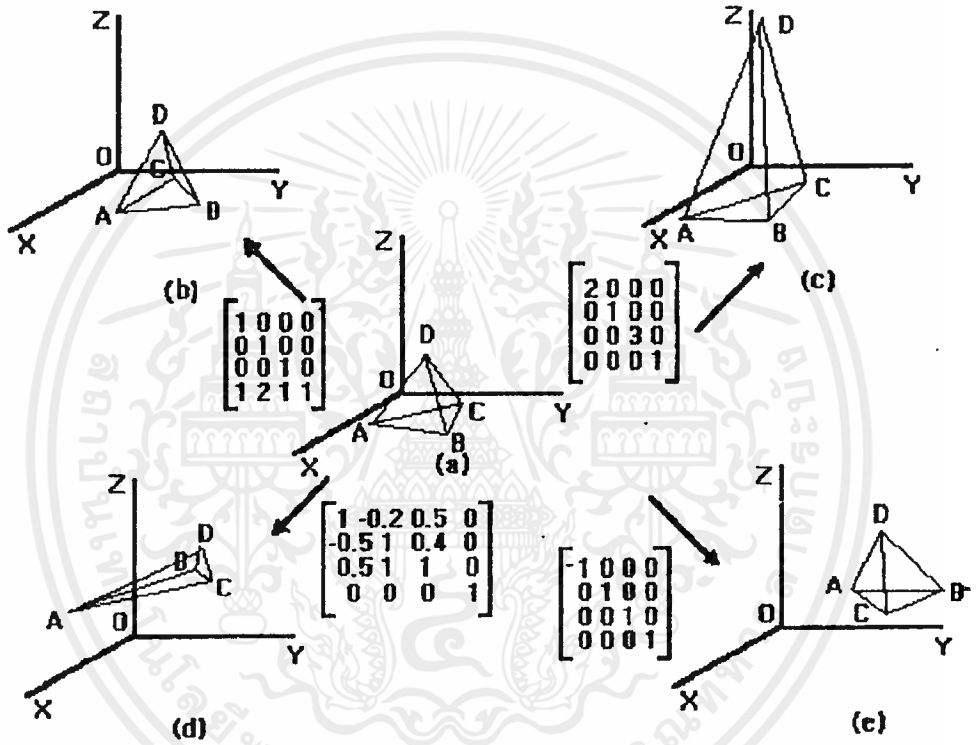
เราจะนำระบบ homogeneous coordinate มาใช้สำหรับการแปลงรูป 3 มิติ เราจะใช้ rectangular coordinate system แทนตำแหน่งเวกเตอร์ (x,y,z) transformation matrix โดยทั่วไปจะใช้ 4x4 เมตริกซ์ สำหรับจุดใน 3 มิติ homogeneous coordinate system คือ

$$\left[\begin{array}{ccc|c} A & B & C & 0 \\ D & E & F & 0 \\ G & H & I & 0 \\ \hline L & M & N & 1 \end{array}\right] \qquad (5.2)$$

และสามารถแบ่งเป็น 3 ส่วนของเมตริกซ์ย่อย

$$\left[\begin{array}{c|c} \begin{matrix} \text{I} \\ (3\times 3) \end{matrix} & \begin{matrix} \text{III} \\ 4 \\ \times \end{matrix} \\ \hline \text{II}(1\times 3) & 1 \end{array}\right] \qquad (5.3)$$

เมตริกซ์ย่อย I จะใช้กระทำ Scale , Shearing , Reflection และการหมุนของวัตถุ 3 มิติ เมตริกซ์ย่อย II จะทำการเคลื่อนย้ายเส้นตรง และเมตริกซ์ย่อย III อนุญาตให้ทำการเคลื่อนย้าย และการแปลงในแบบการคูณ เราจะพิจารณาคุณสมบัติของเมตริกซ์ย่อยเหล่านี้ในส่วนนี้ และแสดงให้เห็นในรูปประมิด ซึ่งจะเห็นดังรูป 5.3



รูป 5.3 transformation ในระบบ 3 มิติ

5.2.1 3-D TRANSLATION

การแปลงการเคลื่อนย้ายจุด(x,y,z)ไปเป็นจุดใหม่ (x',y',z) จะใช้ (L,M,N) ซึ่งถูกแสดงโดย

$$[x',y',z',1] = [x,y,z,1]\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline L & M & N & 1 \end{array}\right] \qquad (5.4)$$

โดยที่ L,M และ N เป็นระยะทางที่เลื่อนจาก x,y และ z ตามลำดับ รูป 5.3 (b) จะแสดงผลการเคลื่อนย้ายรูปประมิด 3 มิติ ถึง 1,2 และ 1 หน่วยในแกน x,y และ z ตามลำดับ

5.2.2 3-D SCALING

เทอมของเส้นทแยงมุมของ transformation matrix 4x4 โดยทั่วไปจะจัดอยู่ในรูป scale การแปลงดังนี้

$$[x',y',z',1] = [x,y,z,1]\left[\begin{array}{ccc|c} A & 0 & 0 & 0 \\ 0 & E & 0 & 0 \\ 0 & 0 & I & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad (5.5)$$

จากสมการ (5.5) ทำให้เราเห็นว่า เราสามารถควบคุม scale ของโคออร์ดิเนทแต่ละแกนได้ การตั้ง scale แบบง่าย ๆ ทำได้โดยเซตค่า A = E = I ถ้าต้องการให้วัตถุใหญ่กว่าเดิมเป็นเฟกเตอร์ของ2จะทำได้โดยเซตค่าA= E = I = 2 ซึ่งผลของการตั้ง scale แสดงดังรูปที่ 5.3(c)

5.2.3 3-D SHEARING

3-D Shearing จะทำได้โดยการปิดเทอมของเส้นทแยงมุมในเมตริกซ์ 3x3 บนซ้ายของสมการ (5.2) ผลลัพธ์ของการ shearing เป็นดังนี้

$$[x',y',z',1] = [x,y,z,1]\left[\begin{array}{ccc|c} 1 & B & C & 0 \\ D & 1 & F & 0 \\ G & H & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \qquad (5.6)$$

3-D Shearing บนปิระมิดจะแสดงได้ดังรูป 5.3(d)

5.2.4 3-D REFLECTION

วัตถุจะสะท้อนไปยังระนาบ ๆ หนึ่งทำได้โดย การย้ายตัวเลขในเส้นทแยงมุมของ 3-D transformation matrix พิจารณาจากบทที่ 1 วัตถุจะสะท้อนถึงระนาบ ๆ หนึ่งโดยจะสร้างภาพสะท้อน(ภาพเงา)ของจุดทั้งหมดของวัตถุบนระนาบตรงกันข้าม สำหรับตัวอย่าง,ในการสะท้อนถึงระนาบXYเพียงค่าในโคออร์ดิเนทZของตำแหน่งของวัตถุเท่านั้น ที่จะมีผลที่ถูกเปลี่ยนแปลงซึ่งจะต้องทำการเปลี่ยนเครื่องหมายค่าในโคออร์ดิเนทZ นั้น ดังนั้น transformation matrix สำหรับการสะท้อนนี้ คือ

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.7)$$

สำหรับการสะท้อนถึงระนาบ YZ จะได้เมตริกซ์ดังนี้

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.8)$$

และสำหรับการสะท้อนถึงระนาบ XZ จะได้เมตริกซ์ ดังนี้

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.9)$$

การสะท้อนของปิระมิดถึงระนาบ YZ จะแสดงได้ดังรูป 5.3(e)

5.2.5 การหมุนรูป 3 มิติ

ก่อนที่จะพิจารณากรณีทั่วไปของการหมุนรูป 3 มิติรอบแกนจำลอง เราจะกล่าวถึงการหมุนรอบแกนโคออร์ดิเนท X,Y,Z ก่อน อันดับแรกจะพิจารณาการหมุนรอบแกน Z ในกรณีนี้มิติในแกน Z จะไม่เปลี่ยน ซึ่งเมตริกซ์การแปลงจะมีค่าเป็น 0 ในแถวที่ 3 ทั้งแถวและคอลัมน์ที่ 3 ทั้งคอลัมน์ ยกเว้นในเส้นทแยงมุมหลักจะเป็น 1 สำหรับเทอมอื่น ๆ จะถูกกำหนดโดยการพิจารณาได้จากการหมุนของวัตถุ 2 มิติ แสดงในสมการ (4.9) ตัวอย่างเช่นการหมุนรอบแกน Z ใน 3 มิติ transformation matrix สำหรับการหมุนของมุม θ รอบแกน Z คือ

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.10)$$

เราจะวัดมุมของการหมุน ซึ่งจะวัดทวนเข็มนาฬิการอบจุด origin เมื่อดูที่จุด origin จากจุดบนแกน X ที่เป็นบวก

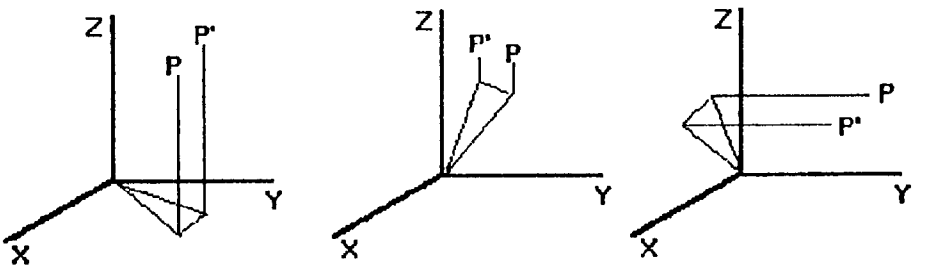
transformation matrix ที่มีผลต่อการหมุนเป็นมุม θ รอบแกน X และแกน Y มีดังนี้

$$R(\theta)_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.11)$$

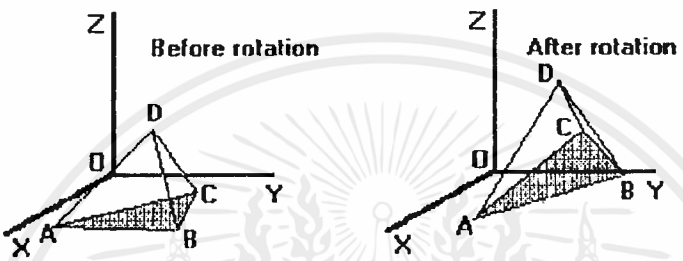
และ

$$R(\theta)_y = \begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (5.12)$$

การหมุนรูป 3 มิติรอบแต่ละแกน จะแสดงได้ดังรูป 5.4



(a)



(b)

รูป 5.4 การหมุนในระบบ 3 มิติ (a) การหมุนของจุด, (b) การหมุนของวัตถุ

ตัวอย่างของการหมุนรูป3มิติ , พิจารณาปิระมิดที่ให้ในรูป 5.4 (a) ถ้าปิระมิดถูกหมุนไป 30 องศา (ทวนเข็มนาฬิกา) รอบแกน Z ใช้สมการ (5.10) เราจะได้โคออร์ดิเนทใหม่ดังนี้

$$\begin{bmatrix} A' \\ B' \\ C' \\ D' \end{bmatrix} = \begin{bmatrix} 3 & 1 & 0 & 1 \\ 3 & 5 & 0 & 1 \\ 1 & 4 & 0 & 1 \\ 2 & 3 & 3 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2.1 & 2.37 & 0 & 1 \\ 0.1 & 5.83 & 0 & 1 \\ 1.13 & 3.96 & 0 & 1 \\ 0.232 & 3.60 & 3 & 1 \end{bmatrix}$$

ผลของการแปลงจะแสดงในรูป 5.4(b)

สมมติปิระมิดเดิมแต่หมุนไป -30 องศา (ตามเข็มนาฬิกา) รอบแกน Z ด้วย theta = -30 และใช้สมการ (5.10) เราจะได้

$$R(-30) = \begin{bmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

นี้คือ อินเวอร์สเมตริกซ์ของ I :

$$R(30).R(-30) = \begin{bmatrix} 0.866 & 0.5 & 0 & 0 \\ -0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0.866 & -0.5 & 0 & 0 \\ 0.5 & 0.866 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = R(30).R(30)^{-1} = I$$

ที่กล่าวมาข้างต้นนี้เมตริกซ์การหมุนมุมตามเข็มนาฬิการอบแกน จะเหมือนกับเมื่ออินเวอสร์ (R(θ)-1)ของเมตริกซ์การหมุนตามมุมทวนเข็มนาฬิการอบแกนเดียวกัน (R(θ)) โดยที่จะแทนเป็นกฎทั่วไป ซึ่งเราจะพบว่าเป็นประโยชน์ในคอมพิวเตอร์กราฟิกมาก โดยที่กฎนี้มีดังนี้

$$R(\theta)_z^{-1} = R(-\theta)_z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.13)$$

$$R(\theta)_x^{-1} = R(-\theta)_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.14)$$

และ

$$R(\theta)_y^{-1} = R(-\theta)_y = \begin{bmatrix} \cos\theta & 0 & \cos\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \sin\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.15)$$

หมายเหตุที่สำคัญ,โดยทั่วไป order ของการคูณจะมีผลซึ่งเป็นผลลัพธ์สุดท้าย ทั้งนี้เพราะการหมุนรูป 3 มิติบางทีเป็นแบบ noncommutative เช่นอาจจะหมุนรอบแกน Z เป็นมุม θ_1 ตามด้วยหมุนรอบแกน X เป็นมุม θ_2 :

$$R(\theta_1)_z . R(\theta_2)_x = \begin{bmatrix} \cos\theta_1 & \sin\theta_1\cos\theta_2 & \sin\theta_1\sin\theta_2 & 0 \\ -\sin\theta_1 & \cos\theta_1\cos\theta_2 & \cos\theta_1\sin\theta_2 & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.16)$$

ในทางกลับกันถ้าหมุนรอบแกน X ด้วยมุม ตามด้วยการหมุนรอบแกน Z ด้วยมุม จะได้

$$R(\theta_2)_z . R(\theta_1)_x = \begin{bmatrix} \cos\theta_1 & \sin\theta_1 & 0 & 0 \\ -\cos\theta_2\sin\theta_1 & \cos\theta_2\cos\theta_1 & \sin\theta_2 & 0 \\ \sin\theta_2\sin\theta_1 & -\sin\theta_2\cos\theta_1 & \cos\theta_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.17)$$

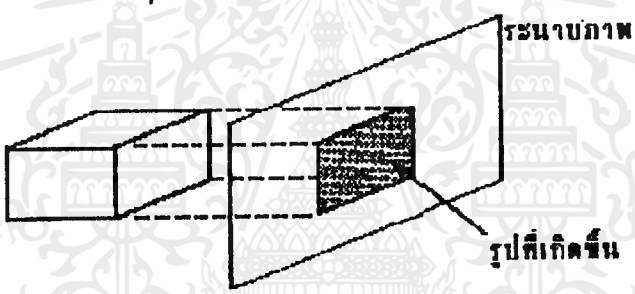
บทที่ 6

การสร้างภาพ 3 มิติ

ในบทที่แล้วเราได้ศึกษาเกี่ยวกับระบบภาพ 3 มิติ ซึ่งเป็นการกล่าวถึงโคออร์ดิเนทในระบบ 3 มิติทั้งหมดสำหรับการแปลงเป็นการแปลงวัตถุในระบบ 3 มิติเท่านั้น อย่างไรก็ตามถึงแม้ว่าเราจะมีวัตถุ 3 มิติ ซึ่งต้องใช้โคออร์ดิเนทถึง 3 แกน ภาพของวัตถุต่าง ๆ เหล่านี้ จะต้องถูกวาดลงบนระนาบราบเรียบ 2 มิติเพื่อให้ผู้ใช้เห็นภาพของวัตถุนั้น เช่น แสดงภาพวัตถุออกทางจอภาพ หรือเครื่องพิมพ์ภาพ ดังนั้นจึงต้องมีวิธีการสร้างภาพ 2 มิติของวัตถุที่ถูกกำหนดไว้ในโคออร์ดิเนท 3 มิติ

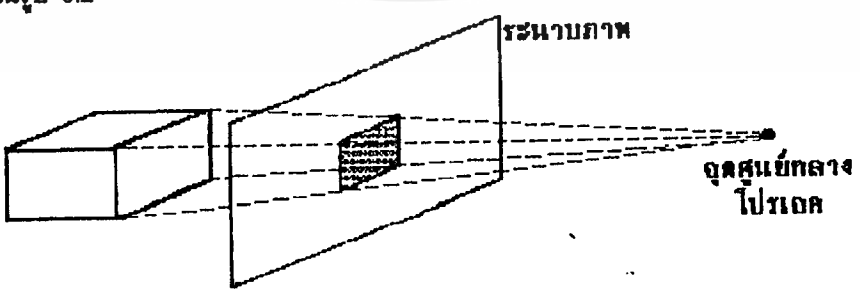
6.1 PROJECTION (โปรเจ็กชั่น)

โปรเจ็กชั่น(Projection) เป็นวิธีการหนึ่งที่สามารถสร้างภาพ 2 มิติได้ โดยการฉายเงา(Project) ของวัตถุให้ไปตกลงบนระนาบหนึ่ง ที่ระนาบก็จะเกิดเป็นเงา หรือภาพ 2 มิติของวัตถุนั้นระนาบนี้เราเรียกว่า *ระนาบภาพ (View Plane)* ดังแสดงในรูป 6.1 จะสังเกตุได้ว่าภาพที่เกิดขึ้นบนระนาบนั้นขึ้นอยู่กับการวางตัวของระนาบภาพ ทิศทางของแสงที่ฉายเงาไปยังระนาบ และองค์ประกอบอื่น ๆ อีก



รูป 6.1 โปรเจ็กชั่น

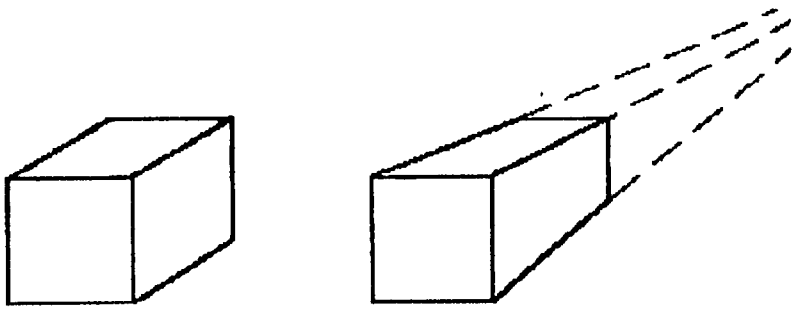
การทำโปรเจ็กชั่นนั้นมีหลายประเภท แต่ที่นิยมใช้กันมีเพียง 2 ประเภทคือ *โปรเจ็กชั่นแบบขนาน (ParallelProjection)* และโปรเจ็กชั่นแบบ *เปอร์สเป็คตีฟ (Perspective Projection)* สำหรับโปรเจ็กชั่นแบบขนานนั้น มีลักษณะเหมือนกับในรูป 6.1 กล่าวคือ แสงของเงาที่ฉายจะพุ่งขนานกับไปหมดตรงสู่จอภาพส่วนโปรเจ็กชั่นแบบเปอร์สเป็คตีฟนั้นแสงของเงาทั้งหมดจะพุ่งสู่จุด ๆ หนึ่ง ไม่ขนานกัน จุดที่แสงของเงาพุ่งไปนี้เรียกว่า *จุดศูนย์กลางโปรเจ็กชั่น(Center of Projection)* ดังแสดงในรูป 6.2



รูป 6.2 โปรเจ็กชั่นแบบเปอร์สเป็คตีฟ

ลักษณะของภาพ ที่เกิดขึ้นจากการทำโปรเจ็กชั่นต่างประเภทกัน จะมีลักษณะต่างกันดังแสดงในรูป6.3เป็น

การเปรียบเทียบภาพที่ได้จากการทำโปรเจ็กชั่นแบบขนาน และโปรเจ็กชั่นแบบเปอร์สเป็คตีฟ



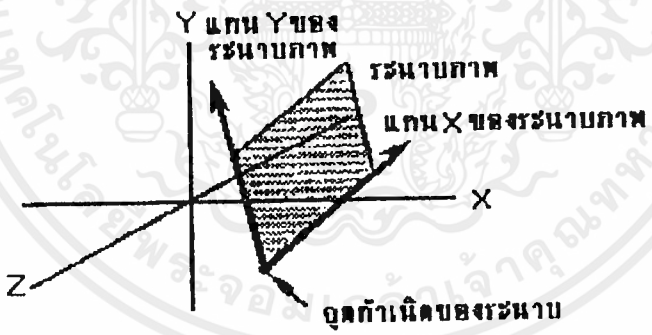
ก) แบบขนาน ข) แบบเปอร์สเปคตีฟ

รูป 6.3 ภาพที่ได้จากการทำโปรเจ็กชั่น

6.2 พารามิเตอร์ภาพ

ดังที่กล่าวไว้แล้วว่าภาพที่เกิดขึ้นบนระนาบภาพของวัตถุเดียวกัน อาจได้ภาพที่ต่างกันก็ได้ขึ้นอยู่กับองค์ประกอบต่าง ๆ ซึ่งมีผลต่อการสร้างภาพ เราเรียกว่าองค์ประกอบต่าง ๆ นี้ *พารามิเตอร์ภาพ (Viewing Parameters)*

ก่อนที่จะกล่าวถึงพารามิเตอร์ต่าง ๆ เราควรรู้จักระนาบภาพให้ละเอียดยิ่งขึ้นเสียก่อน ระนาบภาพ คือระนาบที่เราฉายเงาของวัตถุลงไปภาพของเงาที่เกิดขึ้นบนระนาบจะเป็นภาพ 2 มิติของวัตถุและเป็นภาพที่จะถูกวาดออกทางจอภาพหรืออุปกรณ์พิมพ์ภาพ ระนาบภาพสามารถเป็นระนาบใด ๆก็ได้ที่ตั้งอยู่ในโคออร์ดิเนท3มิติ ระบบโคออร์ดิเนท 3 มิติที่วัตถุอยู่นี้เรียกว่า *ออปเจ็กโคออร์ดิเนท (Object Coordinates)* เราจะมองระนาบภาพนี้เสมือนกับเป็นระนาบ XY ธรรมดาที่วางตัวอยู่ในระบบ มีแกน X แกน Y และจุดกำเนิดของระนาบ ดังแสดงในรูป 6.4 ระบบโคออร์ดิเนทXYของระนาบภาพนี้เรียกว่า *โคออร์ดิเนทของระนาบภาพ (View Plane Coordinates)*

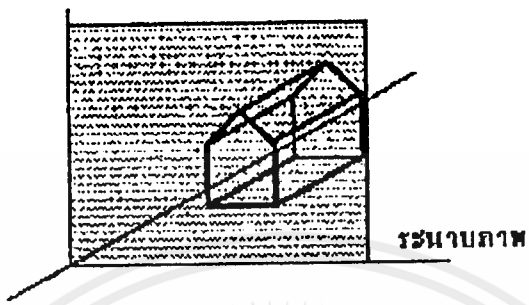


รูป 6.4 ระนาบภาพในระบบโคออร์ดิเนท 3 มิติ

เราอาจเปรียบเทียบระนาบภาพได้กับฟิล์มในกล้องที่เราใช้ถ่ายภาพทั่วไป วัตถุที่เราต้องการถ่ายคือวัตถุในระบบโคออร์ดิเนทซึ่งเงาของวัตถุนั้นจะตกไปอยู่บนแผ่นฟิล์มในกล้องเกิดเป็นภาพบนฟิล์ม(ภาพบนระนาบภาพ)ขณะเดียวกันตำแหน่งของฟิล์มก็คือตำแหน่งที่เรายืนมองวัตถุด้วยเราสามารถย้ายตำแหน่งหรือเปลี่ยนมุมของกล้องถ่ายภาพเพื่อให้ได้ภาพถ่ายที่ออกมามีลักษณะต่างออกไป ระนาบภาพก็เช่นกัน เราสามารถโยกย้ายและวางระนาบภาพไว้ในลักษณะใดก็ได้ที่เราต้องการ ตำแหน่งและการวางตัวของระนาบภาพต้องใช้พารามิเตอร์ 4 ตัวในการกำหนด

พารามิเตอร์ตัวแรกคือ *จุดอ้างอิงภาพ (View Referance Point)* $P_r(x_r,y_r,z_r)$ จุดนี้เป็นจุดศูนย์กลางของความสนใจในการมองภาพของเราพารามิเตอร์ตัวที่สองคือ *เวคเตอร์ตั้งฉากของระนาบภาพ (View Plane Normal Vector)* $N = (x_n, y_n, z_n)$ เป็นเวคเตอร์ที่ตั้งฉากกับระนาบภาพถ้าเราลากเส้นตรงจากจุดอ้างอิงภาพ ให้ขนานกับเวคเตอร์

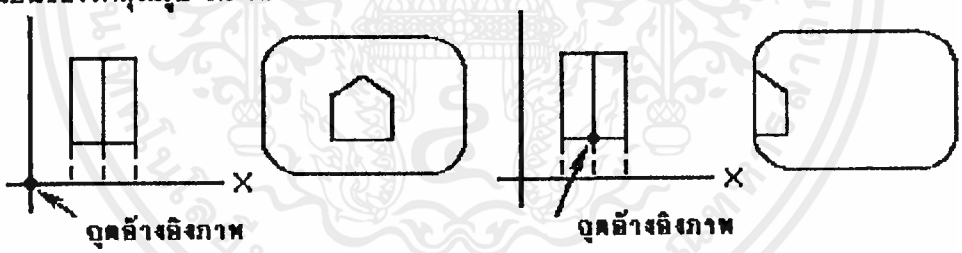
N ไปตั้งฉากกับระนาบภาพจุดที่เส้นตรงนี้ตัดกับระนาบภาพคือจุดกำเนิดของโคออร์ดิเนทของระนาบภาพนั้นเองพารามิเตอร์ตัวต่อไปคือ *ระยะภาพ (ViewDistance)* หมายถึง ระยะห่างระหว่างจุดอ้างอิงภาพและระนาบภาพ โดยปกติ เรามักกำหนดระยะภาพให้เป็นศูนย์ ทำให้จุดอ้างอิงภาพและจุดกำเนิดของระนาบภาพเป็นจุดเดียวกันและพารามิเตอร์ตัวสุดท้ายคือ *วิวอัพเวคเตอร์ (View-upVector)* $V = (x_v\ y_v\ z_v)$ เป็นเวคเตอร์ที่กำหนดแนวทิศทางตั้งขึ้นของโคออร์ดิเนทของระนาบภาพนั่นคือ เวคเตอร์ V เป็นเวคเตอร์ที่ขนานและมีทิศทางเดียวกับแกน +Y ของโคออร์ดิเนทของระนาบภาพ



รูป 6.5 ตัวอย่างวัตถุในระบบออปเจ็กโคออร์ดิเนท

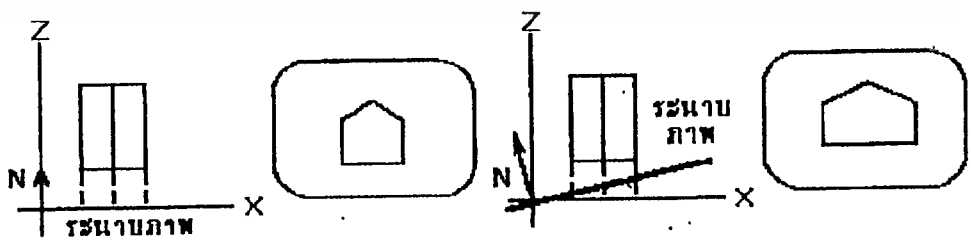
ในระบบคอมพิวเตอร์กราฟฟิคผู้ใช้ต้องกำหนดพารามิเตอร์เหล่านี้เพื่อให้ได้ลักษณะภาพที่ต้องการการกำหนดค่าของพารามิเตอร์ที่ต่างกันย่อมได้ภาพที่ต่างกันออกไป สมมติว่ามีวัตถุชิ้นหนึ่งในออปเจ็กโคออร์ดิเนท เราต้องการให้เกิดภาพบนระนาบ XY นั่นคือระนาบ XY เป็นระนาบภาพ ในรูป 6.5 แสดงวัตถุและระนาบภาพดังกล่าว

การเปลี่ยนจุดอ้างอิงภาพจะทำให้ได้ภาพที่เกิดขึ้น มีการย้ายตำแหน่ง ส่วนของวัตถุที่ปรากฏที่จุดกำเนิดบนโคออร์ดิเนทของระนาบภาพจะเปลี่ยนไปในรูป 6.6 แสดงตัวอย่างผลของการเปลี่ยนตำแหน่งของจุดอ้างอิงภาพ ภาพทางซ้ายเป็นมุมมองด้านบนของวัตถุในรูป 6.5 คือมองตรงลงมาจากแกน Y และภาพทางขวาคือ ภาพที่เกิดขึ้นบนระนาบภาพ



รูป 6.6 การเปลี่ยนจุดมองภาพ

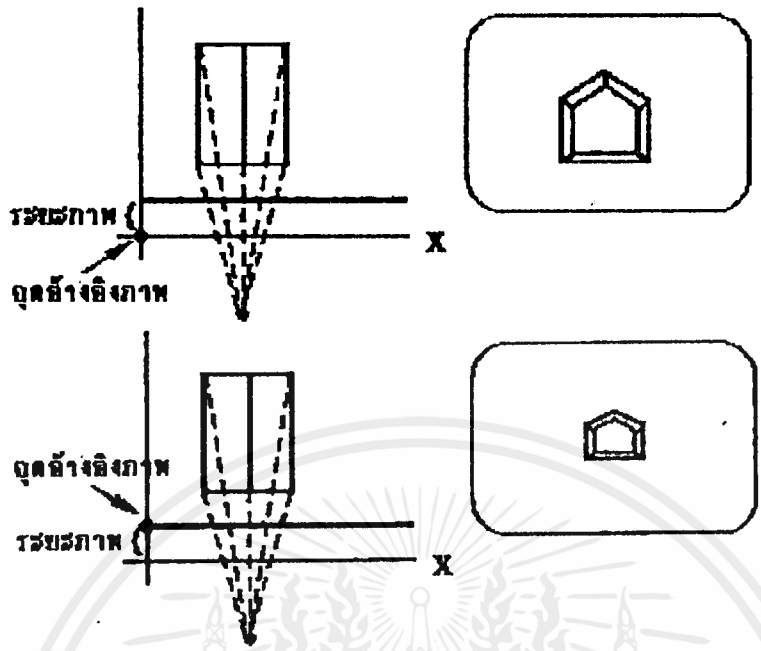
ถ้าเปลี่ยนเวคเตอร์ตั้งฉากของระนาบภาพ การจัดวางระนาบในระบบจะเปลี่ยนไปรูป 6.7 แสดงตัวอย่างผลของการเปลี่ยนทิศของเวคเตอร์ตั้งฉากของระนาบภาพ



รูป 6.7 การเปลี่ยนเวคเตอร์ตั้งฉากของระนาบภาพ

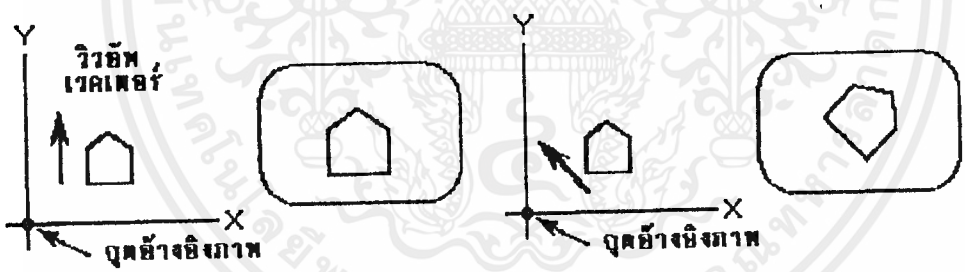
ถ้าเปลี่ยนระยะภาพจะทำให้ได้ภาพที่มีขนาดใหญ่เล็กต่างกันดังแสดงในรูป 6.8 จะสังเกตได้ว่ารูป6.8นั้นเป็น

ภาพจากการทำโปรเจ็กชั่นแบบเปอร์สเป็คตีฟถ้าเป็นการทำโปรเจ็กชั่นแบบขนาน ขนาดของภาพจะไม่ขึ้นอยู่กับระยะภาพ



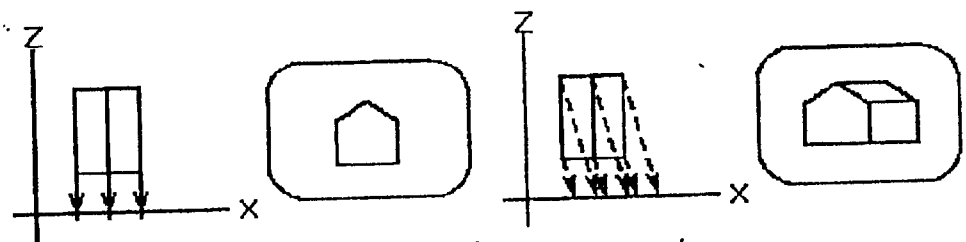
รูป 6.8 การเปลี่ยนระยะภาพสำหรับโปรเจ็กชั่นแบบเปอร์สเป็คตีฟ

การเปลี่ยนทิศทางของวิวอัฟเวคเตอร์เปรียบได้กับการที่เราเอียงกล้องถ่ายภาพไปมานั่นเอง ทำให้ภาพที่ได้มีการเอนเอียงต่างกัน ดังแสดงในรูป 6.9



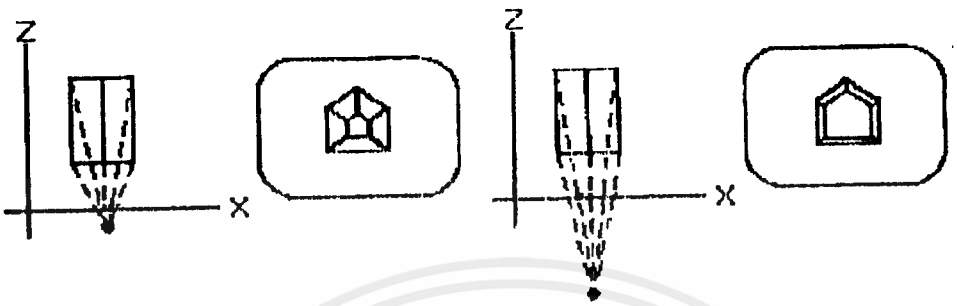
รูป 6.9 การเปลี่ยนวิวอัฟเวคเตอร์

พารามิเตอร์ที่กล่าวมาเป็นพารามิเตอร์ที่เกี่ยวข้องกับการมองวัตถุเสมือนว่าเราจะมองวัตถุในแง่มุมไหนลักษณะใด นอกจากพารามิเตอร์ทั้ง 4 ที่กล่าวมา แล้วยังมีพารามิเตอร์ที่เกี่ยวกับโปรเจ็กชั่นของระบบอีกด้วยสำหรับโปรเจ็กชั่นแบบขนานต้องกำหนดทิศทางของแสงที่ฉายหรือ *ทิศทางโปรเจ็กชั่น (Projection Direction)* $D_p = (x_p\ y_p\ z_p)$ ผลของการเปลี่ยนทิศทางโปรเจ็กชั่น แสดงไว้ในรูป 6.10



รูป 6.10 การเปลี่ยนทิศทางโปรเจ็กชั่น

สำหรับโปรเจ็กชั่นแบบเปอร์สเป็คตีฟต้องกำหนดจุดศูนย์กลางโปรเจ็กชั่นแทนทิศทางของแสงรูป 6.11 แสดงผลของการเปลี่ยนตำแหน่งของจุดศูนย์กลางโปรเจ็กชั่น นอกจากพารามิเตอร์ทั้ง 2 แล้ว ระบบคอมพิวเตอร์กราฟฟิคควรมีพารามิเตอร์หรือตัวแปรอีกตัวหนึ่ง ซึ่งบ่งบอกประเภทของการทำโปรเจ็กชั่นไว้ด้วยว่า เป็นแบบขนานหรือ แบบเปอร์สเป็คตีฟ



รูป 6.11 การเปลี่ยนจุดศูนย์กลางโปรเจ็กชั่น

6.3 ตัวอย่างการทำโปรเจ็กชั่น

โดยทั่วไปแล้วสำหรับการทำโปรเจ็กชั่นแบบขนานเรามักใช้ระนาบ XY เป็นระนาบภาพและทิศทางโปรเจ็กชั่นมีทิศทางขนานกับแนวแกน Z เงาของจุดต่าง ๆ ที่เกิดขึ้นบนระนาบภาพจะมีค่าโคออร์ดิเนท X และ Y ที่ไม่เปลี่ยนแปลง เพราะทิศทางโปรเจ็กชั่นลากตรงจากจุดต่าง ๆ ขนานแกนZลงมาตัดกับระนาบXYทำให้การทำโปรเจ็กชั่นนั้นง่ายมากคือเพียงแค่ตัดค่าโคออร์ดิเนท Z ของจุดเหล่านั้นออก เราก็จะได้ตำแหน่งเงาของจุดนั้นบนระนาบภาพ เช่น จุด P(x,y,z) จะทำให้เกิดจุด P'(x,y) บนระนาบภาพ ในบางครั้งถ้าทิศทางโปรเจ็กชั่นเปลี่ยนไปในทิศทางอื่นวิธีดังกล่าวก็ใช้งานไม่ได้ ถ้าทิศทางของโปรเจ็กชั่นเป็นไปตามทิศทางของเวคเตอร์ $(x_p\ y_p\ z_p)$ แต่ยังคงใช้ระนาบ XY เป็นระนาบภาพ จะได้ว่าจุด P(x,y,z) จะมีเงาหรือทำให้เกิดจุด P'(x',y) บนระนาบภาพ ซึ่ง

$$x' = x - z(x_p/z_p) \qquad (6.1)$$

$$y' = y - z(y_p/z_p) \qquad (6.2)$$

หรืออาจเขียนให้อยู่ในรูปของเมตริกซ์ได้คือ

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ x_p/z_p & y_p/z_p & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.3)$$

โดยที่ z' คือโคออร์ดิเนท Z ของจุด P' ซึ่งเราตัดทิ้ง และ z' นี้จะมีค่าเป็นศูนย์เสมอ เพราะเงาโปรเจ็กชั่นไปบนระนาบ XY (ระนาบ Z = 0) สำหรับตัวอย่างของโปรเจ็กชั่นแบบเปอร์สเป็คตีฟ ถ้าเรายังคงใช้ระนาบ XY เป็นระนาบภาพเช่นเดิม และให้จุด (x_c, y_c, z_c) เป็นจุดศูนย์กลางเปอร์สเป็คตีฟ จุด P(x,y,z) จะมีเงาหรือทำให้เกิดจุด P'(x',y) บนระนาบภาพ ซึ่ง

$$x' = \frac{x_c z - x z_c}{z - z_c} \qquad (6.4)$$

$$y' = \frac{y_c z - y z_c}{z - z_c} \qquad (6.5)$$

หรืออาจเขียนให้อยู่ในรูปเมตริกซ์คือ

$$[x'\ y'\ z'\ 1] = [x\ y\ z\ 1]\begin{bmatrix} z_c & 0 & 0 & 0 \\ 0 & -z_c & 0 & 0 \\ x_c & y_c & 0 & 1 \\ 0 & 0 & 0 & -z_c \end{bmatrix} \qquad (6.6)$$

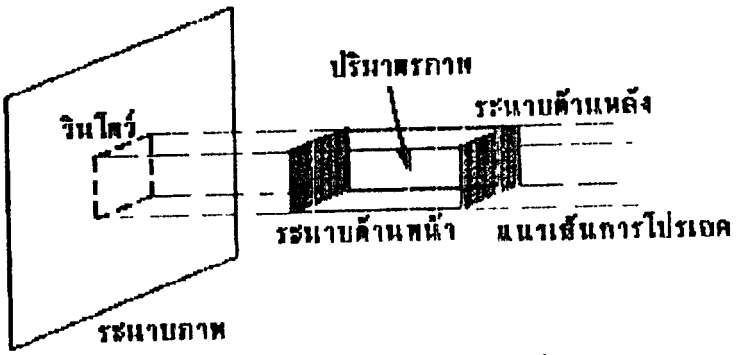
ในบางครั้งเราจะกำหนดจุดศูนย์กลางของโปรเจ็กชั่นอยู่ที่ จุดกำเนิด(0,0,0) และให้ระนาบภาพอยู่ที่ระนาบ Z = d ในกรณีนี้จะต้องใช้เมตริกซ์ต่อไปนี้ แทนเมตริกซ์ในสมการ (6.6)

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/d \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (6.7)$$

ถ้าเรานำเมตริกซ์ในสมการ(6.7)ไปแทนที่เมตริกซ์ในสมการ(6.8) จะได้ค่า z' เท่ากับ d เพราะเมตริกซ์ในสมการ (6.7) เป็นการทำโปรเจ็กชั่นลงบนระนาบ Z = d

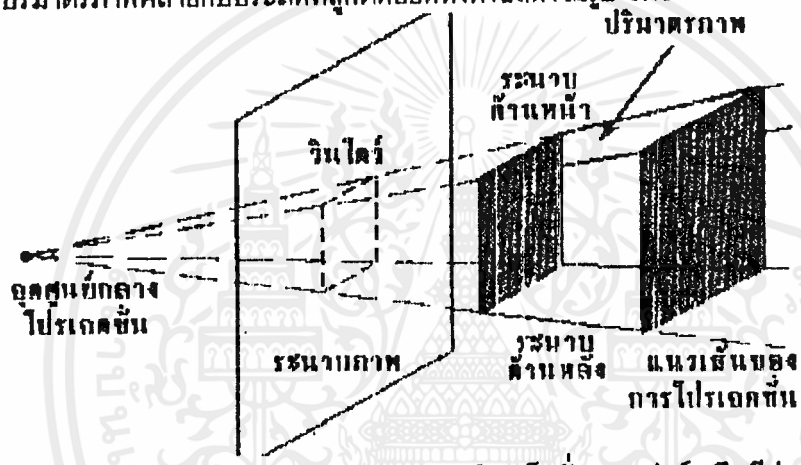
6.4 การคลิป

ในระบบภาพ 2 มิติ เรามีการกำหนดวินโดว์ เพื่อกำหนดขอบเขตของภาพที่เราต้องการเห็นจากนั้นจะคลิปภาพ หรือตัดส่วนของภาพที่อยู่นอกเหนือวินโดว์ออก ทำนองเดียวกัน ในระบบ 3 มิติเรามีการกำหนดขอบเขตของภาพที่จะมองแต่แทนที่จะเป็นกรอบสี่เหลี่ยมดัง เช่นในระบบ 2 มิติ ขอบเขตของภาพในระบบ 3 มิติจะถูกกำหนดด้วยปริมาตรหรือกล่องใบหนึ่งใน ออปเจ็กสเปซกล่องหรือปริมาตรนี้เรียกว่า *ปริมาตรภาพ (View Volum)* วัตถุใด ๆ ก็ตามที่อยู่ภายในบริเวณของปริมาตรภาพจะถูกมองเห็น แต่ส่วนที่อยู่ภายนอกจะถูกคลิปออกไป ลักษณะเช่นนี้คล้าย ๆกับเวลาที่เรามองบ้านหลังหนึ่งจากนอกบ้าน เราจะเห็นประตูบ้าน เมื่อค่อย ๆ เดินเข้าไปภายในบ้าน ประตูบ้านจะค่อย ๆ เลื่อนเข้าใกล้เราจนกระทั่งเมื่อเข้าไปอยู่ภายในตัวบ้านเราจะไม่เห็นประตูหน้าบ้าน เพราะประตูอยู่ด้านหลังของเรานั่นคือ ประตูอยู่นอกปริมาตรภาพ ลักษณะรูปทรงของปริมาตรภาพที่นิยมใช้กันมี 2 แบบ ตามลักษณะของโปรเจ็กชั่น สำหรับโปรเจ็กชั่นแบบขนานปริมาตรภาพมีรูปทรงเป็นกล่องสี่เหลี่ยม ดังแสดงในรูป 6.12 เรากำหนดขอบเขตของวินโดว์บนระนาบภาพ บนขอบแต่ละด้านของวินโดว์ลากเส้นขอบออกมา ในทิศทางของการโปรเจ็กชั่นทำให้เราได้ท่อที่มีรูปทรงสี่เหลี่ยมขนาดเท่ากับวินโดว์ท่อนี้เกิดจากระนาบทั้งสี่ระนาบตัดกันเราต้องกำหนดระนาบอีก 2 ระนาบ คือ ระนาบด้านหน้า (หรือระนาบใกล้) และระนาบด้านหลัง(หรือระนาบไกล) เพื่อปิดหัวท้ายของท่อทรงสี่เหลี่ยมเกิดเป็นกล่องรูปทรงเหลี่ยม บริเวณที่อยู่ภายในกล่องนี้ก็คือ บริเวณภายในของปริมาตรภาพนั่นเอง



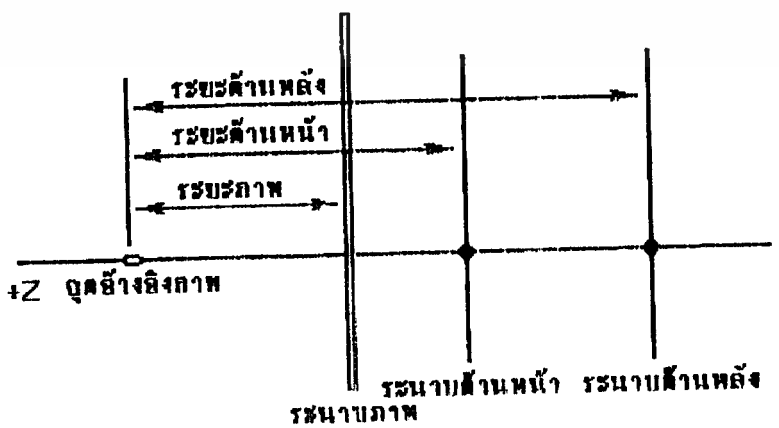
รูป 6.12 ลักษณะของปริมาตรภาพของโปรเจ็กชั่นแบบขนาน

สำหรับปริมาตรภาพของโปรเจ็กชั่นแบบเปอร์สเปคตีฟจะมีวิธีการกำหนดเหมือนกับในกรณีของโปรเจ็กชั่นแบบขนานรูปทรงของปริมาตรภาพคล้ายกับปิระมิดที่ถูกตัดยอดทิ้งดังแสดงในรูป 6.13



รูป 6.13 ลักษณะของปริมาตรภาพของโปรเจ็กชั่นแบบเปอร์สเปคตีฟ

การกำหนดปริมาตรภาพต้องกำหนดขอบเขตหรือกรอบของวินโดว์ บนระนาบภาพเสียก่อนขั้นตอนนี้คล้ายกับการกำหนดวินโดว์ในระบบ2 มิติ ขอบเขตหรือวินโดว์บนระนาบภาพ จะเป็นตัวกำหนดระนาบด้านข้างทั้งสี่(บน ล่าง ซ้าย ขวา) ของปริมาตรภาพ ส่วนระนาบด้านหน้าและระนาบด้านหลังกำหนดได้ด้วยระยะห่างระหว่างระนาบทั้ง 2 กับระนาบภาพ ทั้งระนาบด้านหน้า และระนาบด้านหลังจะขนานกับระนาบภาพ (ดูรูป 6.12 และ 6.13) ดังนั้นเราเพียงกำหนดแค่ระยะห่างระหว่างระนาบทั้งสอง กับระนาบภาพก็เพียงพอแล้ว ดังแสดงในรูป 6.14



รูป 6.14 การกำหนดระนาบด้านหน้าและด้านหลังของปริมาตรภาพ

ในระบบภาพ 2 มิติเราตรวจสอบวัตถุต่าง ๆ กับเส้นขอบของวินโดว์ เพื่อทำการคลิป แต่สำหรับระบบ 3 มิติ เราต้องตรวจสอบตำแหน่งของจุดหรือวัตถุกับระนาบ แทนที่จะตรวจสอบกับเส้นระนาบนี้ได้แก่ระนาบของด้านทั้ง 6 ด้าน ของปริมาตรภาพนั่นเอง เราอาจใช้วิธีเดียวกันกับซัทเตอร์แลนด์-ฮอดก์แมนอัลกอริทึมคือ คลิปออกทีละด้าน โดยเพิ่มการคลิปด้านหน้าและหลังเข้าไป และต้องเปลี่ยนจากการเปรียบเทียบกับเส้นขอบของวินโดว์เป็นการเปรียบเทียบกับระนาบของปริมาตรภาพ ดังแสดงในรูป 6.15

รูป 6.15 ขั้นตอนการคลิปในระบบภาพ 3 มิติ

การตรวจสอบว่าจุด P(x1,y1,z1) อยู่ด้านใดของระนาบ ต้องอาศัยสมการระนาบช่วยถ้าระนาบที่เราต้องการตรวจสอบกับจุดมี สมการเป็น

$$Ax + By + Cz + D = 0 \qquad (6.8)$$

จุด $P(x_1,y_1,z_1)$ จะอยู่บนระนาบนี้ถ้า

$$Ax_1 + By_1 + Cz_1 + D = 0 \qquad (6.9)$$

แต่ถ้าจุด P ไม่ได้อยู่บนระนาบ ผลลัพธ์ของสมการ (6.9) จะไม่เท่ากับศูนย์ ถ้าผลลัพธ์ที่ได้เป็นบวก

$$Ax_1 + By_1 + Cz_1 + D > 0 \qquad (6.10)$$

จุด P จะอยู่ที่ด้านหนึ่งของระนาบ และถ้าผลลัพธ์เป็นลบ

$$Ax_1 + By_1 + Cz_1 + D < 0 \qquad (6.11)$$

จุด P จะอยู่อีกด้านหนึ่งของระนาบ ดังนั้นเราสามารถตรวจสอบได้ว่าจุดต่าง ๆ อยู่ภายในปริมาตรภาพหรือไม่นั้น ดูได้จากเครื่องหมายของผลลัพธ์จากการแทนค่าจุดนั้นลงบนสมการระนาบ ถ้าเราทำโปรเจ็กชั่นแบบขนาน โดยใช้ระนาบ XY เป็นระนาบภาพ และทิศทางโปรเจ็กชั่นขนานกับแกน Z ภาพต่าง ๆ ก็จะเกิดขึ้นบนระนาบ XY การทำโปรเจ็กชั่นจะง่ายมาก คือตัดเอาค่าโคออร์ดิเนททางแกนZทิ้งไป เหลือแต่โคออร์ดิเนท X และ Y เราอาจใช้วิธีการคลิปภาพ(เพื่อทำการคลิปภาพที่อยู่นอกวินโดว์) เป็นการคลิปซ้าย ขวา บน และล่าง ของปริมาตรภาพได้เลยส่วนการคลิปหน้าและหลังก็ทำ

ได้ไม่ยากเช่นกันถ้าระนาบด้านหน้าของปริมาตรภาพคือ ระนาบ

$$z = z_f \qquad (6.12)$$

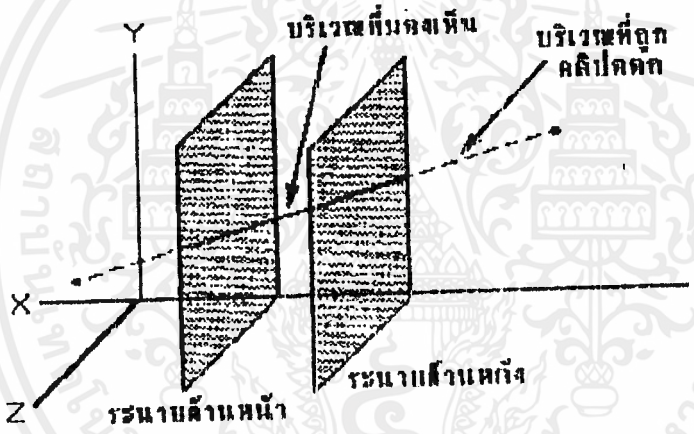
และระนาบด้านหลัง คือ

$$z = z_b \qquad (6.13)$$

จุด $P(x_1,y_1,z_1)$ จะอยู่ภายในปริมาตรภาพ ก็ต่อเมื่อ

$$z_f < z_1 < z_b \qquad (6.14)$$

นั่นคือจะวาดภาพของจุดหรือวัตถุต่าง ๆที่โคออร์ดิเนท z ของจุดหรือวัตถุนั้นอยู่ระหว่างค่า z_f และ z_b ดังแสดงในรูป 6.16



รูป 6.16 ตัวอย่างการคลิปหน้าและหลัง

6.5 การแปลงภาพ

วัตถุต่าง ๆ ที่ถูกสร้างขึ้นในระบบ 3 มิติ เป็นการสร้างในออปเจ็กสเปซ หมายความว่า การกำหนดตำแหน่งและขนาดต่าง ๆเป็นไปตามขนาดของวัตถุนั้นจริง ๆ นอกจากนี้ระนาบภาพและปริมาตรภาพยังถูกกำหนดไว้ในออปเจ็กสเปซด้วยเช่นกัน นั่นคือทั้งระนาบภาพและปริมาตรภาพใช้โคออร์ดิเนทเดียวกันกับวัตถุแต่ภาพที่เราต้องการจะต้องถูกสร้างขึ้นบนจอภาพ 2 มิติ ซึ่งเป็นระบบโคออร์ดิเนทของจอภาพเอง ดังนั้นเจึงต้อง แปลงภาพ (Viewing Transformation) จากโคออร์ดิเนทในออปเจ็กสเปซ ไปยังโคออร์ดิเนทของจอภาพ ซึ่งมีขั้นตอนการแปลงภาพดังนี้

1.สร้างหรือกำหนดตำแหน่งของส่วนต่าง ๆ ของวัตถุ ลงบนโคออร์ดิเนทของออปเจ็กสเปซซึ่งเป็นระบบ 3 มิติ

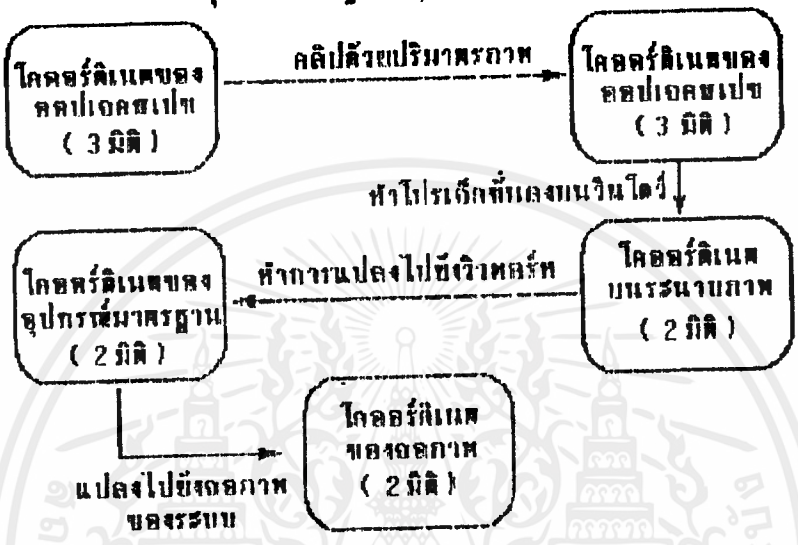
2.เมื่อสร้างวัตถุไว้ในโคออร์ดิเนทของออปเจ็กสเปซเรียบร้อยแล้วคลิปวัตถุในออปเจ็กสเปซด้วยปริมาตรภาพ ในขั้นนี้โคออร์ดิเนทต่าง ๆ ของภาพยังคงไม่เปลี่ยนแปลงเพียงแต่เป็นการกำหนดขอบเขตที่เราต้องการสร้างภาพ หรือขอบเขตของความสนใจเท่านั้น

3.ทำโปรเจ็กชั่นวัตถุภายในปริมาตรภาพ ลงบนระนาบภาพซึ่งจะได้ภาพของวัตถุอยู่ภายในขอบเขต ของ

วินโดว์ด้วยในขั้นนี้โคออร์ดิเนทของจุดต่าง ๆ จะถูกเปลี่ยนไปเป็น โคออร์ดิเนทบนระนาบภาพซึ่งเป็นระบบโคออร์ดิเนท 2 มิติเท่านั้น เช่นจุด $P_1(x_1,y_1,z_1)$ ในออปเจ็กสเปซถูกแปลงเป็นจุด P'(x',y) หมายความว่าจุด P_1 นี้มีเงาอยู่ในตำแหน่ง (x',y) บนโคออร์ดิเนทของระนาบภาพ(ระนาบภาพมีโคออร์ดิเนท XY อยู่บนระนาบ) และแน่นอนจุด P'(x',y) ต้องอยู่ภายในขอบเขตของวินโดว์บนระนาบภาพด้วยเช่นกัน

4.แปลงภาพภายในวินโดว์ลงบนวิวพอร์ทซึ่งเป็นการแปลงจากโคออร์ดิเนท 2 มิติบนระนาบภาพ ไปยังโคออร์ดิเนทของ อุปกรณ์มาตรฐาน (Normal Device Coordinate) ซึ่งเป็น 2 มิติเช่นกัน

5.แปลงจากโคออร์ดิเนทของอุปกรณ์มาตรฐานไปยังโคออร์ดิเนทของจอภาพ



รูป 6.16 ลำดับการแปลงภาพในระบบ 3 มิติจากออปเจ็กสเปซไปยังจอภาพ

บทที่ 7

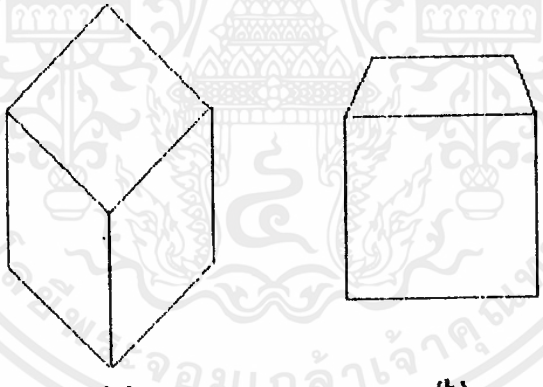
การลบเส้นและพื้นผิวที่ถูกบัง

การสร้างรูป 3 มิติที่สมจริงจะต้องไม่ลากเส้นที่ถูกบังหรือพื้นผิวที่มองไม่เห็น การกระทำเช่นนี้จะต้องพิจารณาว่าเส้นจะมองเห็นหรือมองไม่เห็น แล้วแสดงเฉพาะส่วนที่มองเห็น ส่วนที่ถูกบังหรือมองไม่เห็นไม่ต้องแสดง อัลกอริทึมในการทำจะต้องใช้เวลาในการคำนวณสูงจึงเสียเวลามากในการวาด

ก่อนที่จะพูดถึงขบวนการในการทำมี 2 สมมติฐานคือ หนึ่งวัตถุมีพื้นผิวภายนอกปิด เราใช้เส้นประกอบเป็นรูปทรงหลายเหลี่ยม (polygon) สองคือทุกพื้นผิวมีลักษณะแบนราบ หมายความว่า ทุกจุดที่ล้อมรอบ polygon นั้นอยู่บนระนาบเดียวกัน

7.1 การมองวัตถุทรงเหลี่ยม 1 ก้อน

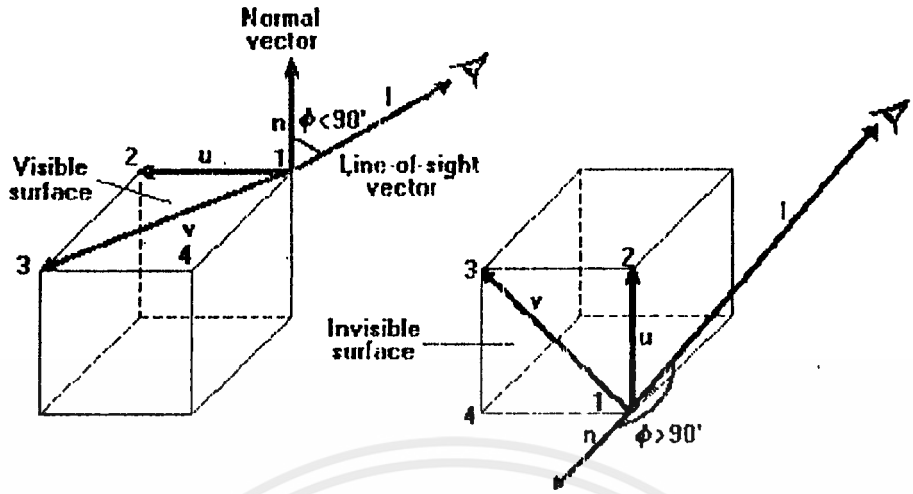
ถ้าเรามีวัตถุ1 ก้อนลักษณะ convex (คือ วัตถุที่มีมุมแหลม วัตถุที่ เส้นต่อเชื่อมระหว่างจุดในวัตถุโดยทั้งหมดบรรจุในวัตถุ เช่น ลูกบาศก์) เราสามารถแก้ปัญหาการมองโดยคิดจากของ normals ของพื้นผิววัตถุ ซึ่งก็คือ face normal vector เป็นเวกเตอร์ที่ทำมุมฉากกับพื้นผิววัตถุทรงตันเช่นลูกลูกบาศก์ (ดังรูป 7.1) มี 8 vertex ประกอบด้วยเส้น 12 เส้นและ 6 พื้นผิว แต่ละพื้นผิวเรียบเป็นรูปสี่เหลี่ยม ถ้าเรามองลูกบาศก์จากจุด P เราจะมองเห็นด้านของลูกบาศก์เพียง 2 ถึง 3 ด้านเท่านั้น เพื่อเป็นการหลีกเลี่ยงการวาดพื้นผิวถูกบังจะต้องทำ visible test ในการทดสอบว่าพื้นผิวนั้นจะมองเห็นหรือไม่



รูป 7.1 จำนวนพื้นผิวของลูกบาศก์ที่สามารถมองเห็น
(a) สามด้านที่มองเห็น (b) สองด้านที่มองเห็น

การทำ visible test จะใช้เวกเตอร์ 2 ตัว คือ surface-normal vector และ line-of-sight vector เราแทน surface-normal vector ด้วย n เป็นเวกเตอร์ที่ชี้ออกตั้งฉากกับพื้นผิว ส่วน line-of-sight vector แทนด้วย l เป็นเวกเตอร์ที่ลากจาก *จุดสายตา (viewpoint)* ไปยังจุดเริ่มต้นของ surface-normal vector(ดังแสดงในรูป 7.2)

เราจะคำนวณ มุม phi ระหว่าง n กับ l ถ้ามุมนั้นอยู่ระหว่าง 0 - 90 องศา แสดงว่าพื้นผิวนั้นมองเห็นและต้องวาดแสดงออกมา แต่ถ้ามุมมากกว่า 90 องศา พื้นผิวนั้นจะมองไม่เห็น และไม่ต้องแสดงออกมา



รูป 7.2 แสดง surface-normal และ line-og-sight vector

surface-normal vector การคำนวณ เวกเตอร์ n เราจะต้องทราบหมายเลขของแต่ละ vextor ของแต่ละพื้นผิวในทิศทางทวนเข็มนาฬิกา หลังจากนั้นกำหนดเวกเตอร์ u ซึ่งลากจากจุดที่ 1 ไปยังจุดที่ 2 และเวกเตอร์ v ลากจากจุดที่ 1 ไปยังจุดที่ 3 ผลจากการทำ cross produce u X v จะได้เวกเตอร์ n ซึ่งทิศทางชี้ออกจากพื้นผิว

$$n = u \times v = (bf-ce,\ cd-af,\ ae-bd) \qquad (7.1)$$

ซึ่ง $u = (a,b,c)$ และ $v = (d,e,f)$

line-of-sight vector เราคำนวณจากเส้นที่ลากจากจุดสายตา $P(D,\theta,\phi)$ ไปยังจุดใดๆ บนพื้นผิววัตถุ เพื่อความสะดวก เราเลือกจุดที่ตั้งต้นของ surface-normal vector (vertex 1)

(7.2)

เมื่อ (x_1,y_1,z_1) คือพิกัด vertex 1

visible test กำหนด dot produce ของเวกเตอร์ n และ l ดังนี้

$$n.l = (n_1,n_2,n_3).(l_1,l_2,l_3) = (n_1l_1 + n_2l_2 + n_3l_3) \qquad (7.3)$$

$$n.l = |n|.|l|.\cos\phi \qquad (7.4)$$

$$\phi = \arccos\left[\frac{n.l}{|n||l|}\right]$$

โดยที่ $|n|$ = ความยาวของเวกเตอร์ n
$|l|$ = ความยาวของเวกเตอร์ l
= มุมระหว่าง n กับ l

สำหรับพื้นผิวที่มองเห็นมุม phi จะต้องอยู่ระหว่าง 0-90 องศา นั่นคือ $n.l > 0$ แต่พื้นผิวที่ถูกบัง มุม phi จะอยู่ระหว่าง 90 องศา และ 180 องศา นั่นคือ $n.l < 0$

7.2 การมองเห็นของวัตถุหลายก้อน

ในหัวข้อก่อนเราก่าวถึงการวาดวัตถุ 1 ก้อน แต่ในหัวข้อนี้จะวาดวัตถุหลายก้อน ซึ่งวัตถุ อาจจะถูกบังโดยวัตถุที่อยู่ใกล้กว่า ซึ่งเพิ่มจากการคำนวณการมองเห็นของวัตถุก้อนเดียว อัลกอริทึมสำหรับการเอาส่วนที่ถูกบังจะกระทำใน *object space* หรือ *image space* อย่างใดอย่างหนึ่ง

สำหรับ *object space algorithm* ใช้ความสัมพันธ์ทางเรขาคณิตของวัตถุในฉาก สำหรับการตัดสินส่วนที่ถูกบัง *image space algorithm* จะใช้ระนาบ 2 มิติ พิจารณาสิ่งที่มอง เห็นแต่ละ pixel วิธีนี้ออกแบบใช้บน raster-scan graphical device โดยทั้งสองวิธีจะให้คุณลักษณะและความต้องการด้าน hardware ที่แตกต่างกัน object space method ใช้พื้นฐานของ hidden line algorithm ส่วน image space method ใช้ hidden surface algorithm อัลกอริทึมที่จะใช้ต่อไปนี้บางส่วนกระทำใน object space และบางส่วนกระทำใน image space โดยการใช้ object space สำหรับการตัดสินส่วนเฉพาะวัตถุเดี่ยวๆ และใช้ image space กำหนดความสัมพันธ์ทางเรขาคณิตระหว่างวัตถุหลายก้อน

วิธีทางเรขาคณิตสำหรับ hidden line ประกอบด้วยขั้นตอนใหญ่ๆ 3 ขั้นตอน

ขั้นที่ 1 ใช้ visible test สำหรับวัตถุทรงตันในการกำหนด hidden surface ของวัตถุ

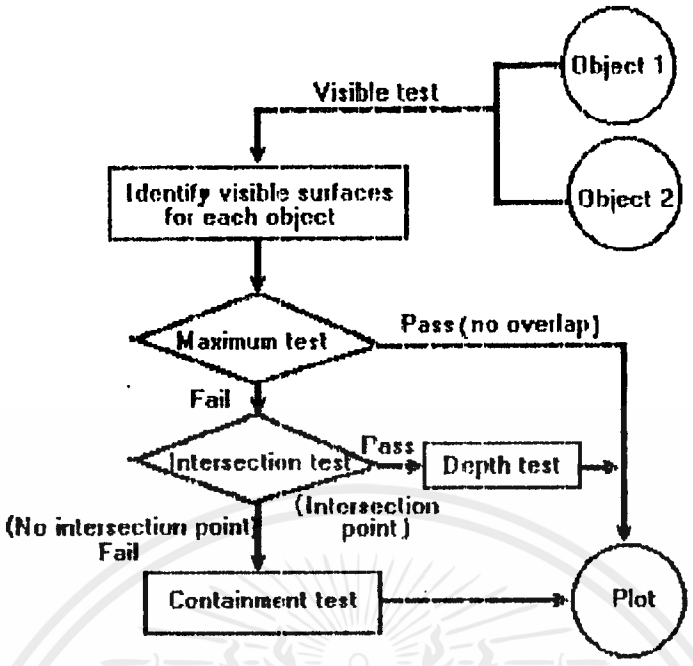
ขั้นที่ 2 ดูความสัมพันธ์ทางเรขาคณิตของวัตถุที่ปรากฏบนฉาก ซึ่งมี 3 กรณีที่เป็นไปได้สำหรับ polygon แต่ละคู่

1) ทั้งสองเหลื่อมล้ำกัน (intersect)
2) polygon หนึ่งถูกล้อมรอบด้วยอีก polygon หนึ่ง (surround)
3) ทั้งสองไม่เกี่ยวข้องกันเลย (not overlap)

การทดสอบนี้เราใช้ minimax test สำหรับการตรวจสอบ

ขั้นที่ 3 ถ้า minimax test ผลออกมาว่าเป็นไปได้ที่จะเกิดการทับกัน เราจะทดสอบ polygon ทั้งสอง intersact หรือ polygon หนึ่งถูกล้อมรอบโดยอีกอันหนึ่ง โดยการกระทำ intersect test ก่อน การ chect กับเส้นที่เป็นขอบรูปของ polygon หนึ่งจะไปทับกับขอบรูปของอีก polygon หนึ่งหรือไม่ การตรวจดูว่าที่บังกันจะต้องใช้ depth test (priority test) ในการตัดสินว่า polygonใด จะตัวบัง polygon ใดจะถูกบัง แต่ถ้าไม่มีการตัดกันของเส้นขอบรูป มี 2 ทาง ที่เป็นไปได้คือ polygon หนึ่งถูกบังมิดเลย กับอีกกรณีคือทั้งสองไม่บังกันเลย โดยเราจะใช้ contenment test ในการตรวจสอบ

ซึ่งขั้นตอนต่างๆได้แสดงเป็น flowchart ดังในรูป 7.3



รูป 7.3 ขั้นตอนในการตรวจสอบซึ่งใช้ทำ hidden line elimination

MINIMAX TEST

รูปทรงปิด A กับ B บน image plane ซึ่ง projection จาก 2 พื้นผิวของวัตถุในระนาบ 3 มิติ ดังแสดงในรูป 7.4 แต่ละ polygon เราจะหาจุดสูงสุดและจุดต่ำสุดของพิกัด X และจุดสูงสุดและจุดต่ำสุดของพิกัด Y และนำเอาจุดสูงสุดและต่ำสุดมาเปรียบเทียบกัน ถ้าอย่างน้อยสมการต่อไปนี้เป็นจริงแม้เพียงสมการเดียว แสดงว่า polygon A กับ B ไม่มีทางทับกัน

$$\begin{aligned} \max x_A &< \min x_B \\ \max x_B &< \min x_A \\ \max y_A &< \min y_B \\ \max y_B &< \min y_A \end{aligned} \qquad (7.5)$$

โดย x_j = คือพิกัด X ของ polygon j (j = A,B)

y_j = คือพิกัด Y ของ polygon j (j = A,B)

ผลจากการทดสอบอาจจะไม่แน่เสมอไปที่วัตถุจะตัดกันจริง ดังตัวอย่างในรูป 7.4(d) และ 7.4(e) ซึ่งจะต้องทำการทดสอบต่อไปโดยใช้ intersection test

INTERSECTION TEST

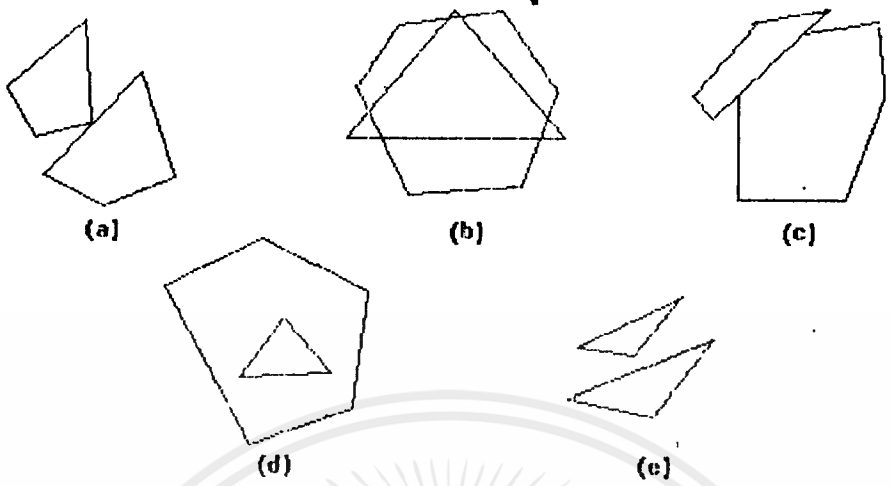
การตัดกันของเส้นตรงสองเส้น ให้สมการเส้นตรง 2 เส้นคือ L กับ L'

$$L = A_1X + B_1Y + C_1 = 0 \qquad (7.6)$$

(A,B ต้องไม่เท่ากับศูนย์)

$$L' = A_2X + B_2Y + C_2 = 0$$

(A,B ต้องไม่เท่ากับศูนย์)



รูป 7.4 ภาพที่เป็นไปได้ซึ่งทำให้เห็นความแตกต่างของการตัดกันของ 2 polygon (a) 1 จุด (b) หลายจุด (c) จำนวนจุดอนันต์ (d) ไม่มีจุดตัด (e) ไม่ทับกัน

มี 3 กรณีที่เป็นไปได้

1. เส้นตรง L และ L' อาจจะขนานกัน ไม่มีการตัดกัน

$$\begin{vmatrix} A_1 & B_1 \\ A_2 & B_2 \end{vmatrix} = A_1B_2 - B_1A_2 = 0$$

2. เส้นตรง L และ L' อาจจะทับกัน จะมีจุดตัดกันมากมาย

$$A_1/A_2 = B_1/B_2 = C_1/C_2$$

3. เส้นตรงตัดกันที่พิกัด (x1,y1) โดยหาได้จากสมการ

$$x = \frac{B_1C_2 - B_2C_1}{A_1B_2 - B_1A_2}, \quad y = \frac{C_1A_2 - C_2A_1}{A_1B_2 - B_1A_2} \qquad (7.7)$$

การตัดกันกันของส่วนของเส้นตรงสองเส้น เรากำหนดของส่วนของเส้นตรง G และ G' บน L และ L' โดย G มีจุดปลายคือ (x_1,y_1) และ (x_2,y_2) และส่วนของเส้นตรง G' มี (x_1',y_1') และ (x_2',y_2') ให้ L และ L' ตัดกันที่จุด I = (x_I,y_I) และ G กับ G' จะไม่ตัดกันที่จุด I เมื่อสมการต่อไปนี้เป็นจริง

$$xI < A1,\ xI > B1,\ xI < C1,\ xI > D1, \qquad (7.8)$$
$$yI < A2,\ yI > B2,\ yI < C2,\ yI > D2$$

เมื่อ

A1 = min (x1,x2); B1 = max (x1,x2);
A2 = min (y1,y2); B2 = max (y1,y2);
C1 = min (x1',x2'); D1 = max (x1',x2');
C2 = min (y1',y2'); D2 = max (y1',y2');

การทดสอบการตัดกันของขอบรูป งานที่ต้องทำไม่เพียงแต่หาว่าเส้นขอบตัดกันหรือไม่ แต่ต้องหาจุดตัดระหว่างเส้นตัดระหว่างเส้นตัด ซึ่งเส้นขอบจะถูกกำหนดโดยจุดปลาย 2 จุดให้จุดตัด (x_1,y_1) และ (x_2,y_2) เป็นจุดปลายของ polygon A จะได้สมการของเส้นขอบดังนี้

$$y = y_1 + m(x - x_1), \qquad (7.9a)$$

เมื่อ

$$m = (y_2 - y_1)/(x_2 - x_1)$$

หรือเขียนได้อีกอย่างคือ

$$mx - y + (y_1 - mx_1) = 0$$

ให้จุดตัด (x_1',y_1') และ (x_2',y_2') เป็นจุดปลายของ polygon B จะได้สมการ

$$m'x - y + (y_1' - m'x_1') = 0 \qquad (7.9b)$$

เมื่อ

$$m' = (y_2' - y_1')/(x_2' - x1')$$

การค้นหาจุดตัด

ขั้นที่ 1:

$$\begin{vmatrix} m & -1 \\ m' & -1 \end{vmatrix} = m' - m$$

ถ้า $m' - m = 0$ ข้ามไป ขั้นที่ 5

ขั้นที่ 2: ถ้า $m/m' = 1 = (y_1 - mx_1)/(y_1' - m'x_1')$ ข้ามไปขั้นที่ 5

ขั้นที่ 3: คำนวณหาจุดตัด (x_i,y_i) จากสมการ

$$x = \frac{(y_1 - y_1') - (mx_1 - m'x_1')}{m' - m}$$

$$y = \frac{(y_1 - mx_1)m' - (y_1' - m'x_1')m}{m' - m} \qquad (7.9c)$$

ขั้นที่ 4: ดูว่าส่วนของเส้นตรงนั้นตัดกันหรือไม่ โดยทดสอบเหมือนกับการตัดกันของส่วนของเส้นตรงดังที่กล่าวมาแล้ว

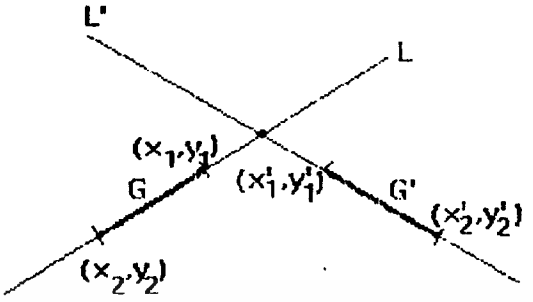
ขั้นที่ 5: เลือกขอบรูปจาก polygon B เส้นอื่น และกระทำซ้ำตั้งแต่ขั้นที่1 ถ้าไม่มีเส้นขอบเส้นอื่นอีก หยุดทำงาน

CONTAINMENT TEST

ถ้าไม่มีการตัดกันของเส้น 2 polygon เป็นไปได้ที่ polygon หนึ่งอาจถูกล้อมรอบโดยอีก polygon หนึ่ง หรือไม่ก็ทับกันเลย เราจะทดสอบจุด (vertex) ของ polygon ซึ่งอาจถูกบังโดยอีก polygon หนึ่งโดยใช้ 2 ขบวนการทางเรขาคณิต

การทดสอบโดยคำนวณ a sum of angle เราคำนวณผลรวมของมุมจากจุดของ polygon A ซึ่งลากไปยัง

ทุกๆจุดบน polygon B (ดังรูป 7.6) ให้ $P_j = (x_j, y_j)$ โดย j คือมุมที่ j ของ polygon B โดย j = 1...n และ P_n คือ P_1 โดยมุมเป็นบวกเมื่อทำมุมทวนเข็มนาฬิกา



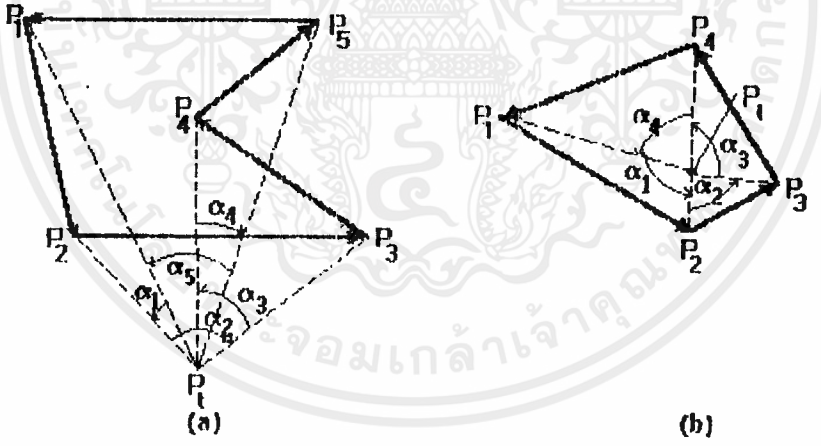
รูป 7.5 การตัดกันของเส้นขอบ

P_iP_j เป็นเส้นตรงที่เชื่อม P_i กับ P_j ให้ α_j เป็นมุมระหว่าง $P_{i-1}P_j$ กับ P_iP_{j+1} ถ้าการวัดมุมจาก P_iP_j ไปยัง P_iP_{j+1} ทวนเข็มนาฬิกาก็ให้ α_j เป็นมุมบวก แต่ถ้าทำมุมตามเข็มนาฬิกา α_j ก็เป็นมุมลบ เราคำนวณผลรวมของ α_j สำหรับทุก j

ถ้า $\alpha_j = 0$ แสดงว่าจุด P_i อยู่นอก polygon B

ถ้า $\alpha_j = 360$ แสดงว่าจุด P_i อยู่ใน polygon B

จากรูป 7.6 (a) เราสามารถคำนวณมุม α_1 เมื่อ $P_1(30,80)$, $P_2(40,50)$ และ $P_i(60,40)$ ดังนี้



รูป 7.6 Containmen test โดยใช้วิธีรวมผลบวกของมุม

$$P_iP_1 = (30,80) - (60,40) = (-30,40)$$

$$P_iP_2 = (40,50) - (60,40) = (-20,10)$$

$$\left|P_iP_1\right| = \sqrt{(-30)^2 + 40^2} = 50$$

$$\left|P_iP_2\right| = \sqrt{(-20)^2 + 10^2} = 22.36$$

$$P_tP_1 \cdot P_tP_2 = (-30,40).(-20,10) = 1000$$

$$\theta_t = \cos^{-1}\frac{1000}{(50).(22.36)} = 26.56°$$

การทดสอบโดยใช้ half-plan method การตัดสินว่า half-plane ใดของ polygon B ที่ test point(จุดของ polygon A) อยู่ เมื่อเราลากเส้นเชื่อมระหว่างจุด D และ Q บนระนาบ XY จะแบ่งพื้นที่ออกเป็น half-plane 2 ระนาบ ทางซ้ายของเวกเตอร์ PQ เรียกว่า left half-plane ส่วนทางขวาเรียก right half-plane

เราบวกแต่ละเส้นขอบของ polygon เป็นเวกเตอร์ที่ตัดระนาบ XY และกำหนดจุด P_t เป็นจุดทดสอบ ก่อนอื่นกำหนด 2 เวกเตอร์ u กับ v โดยเวกเตอร์ v ลากจากจุด P_j ไปยัง P_tเวกเตอร์ u ลากจาก Pj ไปยัง P_j+1 เราสามารถกล่าวได้ว่า

- ถ้า P_t อยู่ทางซีก right half-plane ของ $P_jP_j + 1$ แล้ว u X v จะเป็นสัมประสิทธิ์ลบของ (0,0,1)
- ถ้า P_t อยู่ทางซีก left half-plane ของ PjPj+1 แล้ว u X v จะเป็นสัมประสิทธิ์บวกของ (0,0,1)

ให้ P_j = (a,b) Pj+1 = (e,f) และ P_t = (c,d)

u = (e-a,f-b)

v = (c-a,d-b)

แต่เราคิดในระบบ 3 มิติ เนื่องจากเวกเตอร์ 2 มิติบนระนาบ XY คือเวกเตอร์ 3 มิติที่มีพิกัด Z = 0 ดังนั้นเราได้

u = (e-a,f-b,0)

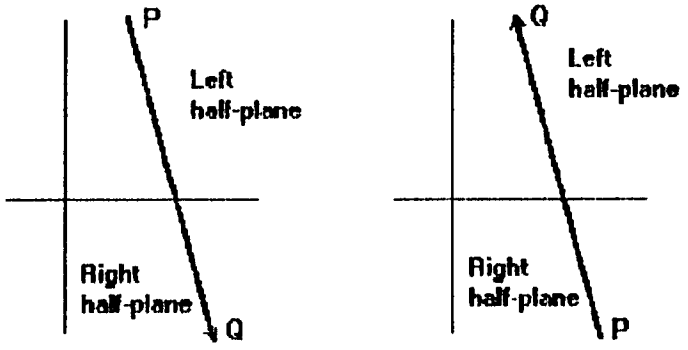
v = (c-a,d-b,0)

u X v = (0,0,(e-a)(d-b) - (f-b)(c-a)) (7.10)

ถ้า (e-a)(d-b)>(f-b)(c-a) แสดงว่า P_t อยู่ left half-plane ของ P_jP_j+1

ถ้า (e-a)(d-b)<(f-b)(c-a) แสดงว่า P_t อยู่ right half-plane ของ P_jP_j+1

การคำนวณ containment test เราจะต้องทำทุกๆเส้นขอบของ polygon B โดยหมุนทวนเข็มนาฬิกา ถ้า Pt อยู่ใน polygon B แล้ว Pt จะต้องอยู่ซีก left half-plane ของแต่ละเส้นขอบของทุกเส้นของ polygon B ถ้ามีเพียงเส้นใดที่ Ptไม่อยู่ซีก left half-plane แล้วละก็ Pt จะอยู่นอก polygon B แต่วิธีนี้จะใช้ได้กับวัตถุทรง convex เท่านั้น ถ้าเป็นรูป 7.6 (a) จะใช้ไม่ได้



รูป 7.7 containment test โดยใช้ half-plane method

DEPTH TEST

สมการระนาบ ให้ $P_j = (x_j,y_j,z_j)$ เมื่อ j = 1,2,3 เป็นจุดของพื้นผิว polygon เรากำหนดเวกเตอร์ u และ v อยู่บนระนาบพื้นผิว

$$u = P_1P_2 = (x_2,y_2,z_2) - (x_1,y_1,z_1) = (x_2-x_1,y_2-y_1,z_2-z_1)$$
$$v = P_1P_3 = (x_3,y_3,z_3) - (x_1,y_1,z_1) = (x_3-x_1,y_3-y_1,z_3-z_1)$$

สมการทั่วไปของระนาบ

$$Ax + By + Cz + D = 0 \qquad (7.11)$$

โดยสัมประสิทธิ์ (A,B,C) จะได้จาก

$$(A,B,C) = u \times v \qquad (7.12)$$

$$A = y_1(z_2-z_3) + y_2(z_3-z_1) + y_3(z_1-z_2)$$
$$B = z_1(x_2-x_3) + z_2(x_3-x_1) + z_3(x_1-x_2)$$
$$C = x_1(y_2-y_3) + x_2(y_3-y_1) + x_3(y_1-y_2)$$
$$D = -Ax - BY - Cz \qquad (7.13)$$

ลำดับความสำคัญ คือ การคำนวณว่าพื้นผิวที่ทับกันพื้นผิวใดอยู่ใกล้จุดมุมมองมากกว่า ก่อนอื่นจะต้องคำนวณจุดตัดกันของ polygon (อาจตัดกันหลายจุด แต่ต้องการเพียงจุดเดียว) ขั้นที่สอง คือ คำนวณ Zs ของแต่ละ polygon ที่จุดตัดนั้น

1) ให้ (x_i,y_i) คือพิกัดบนจอของจุดตัด

2) สมการระนาบของ polygon บนระนาบพิกัดจอ

$a_1x_s + b_1y_s + c_1z_s + d_1 = 0$ สำหรับ polygon 1

$a_2x_s + b_2y_s + c_2z_s + d_2 = 0$ สำหรับ polygon 2

3) แทน $x_s = x_i$ และ $y_s = y_i$ แก้สมการหา z_s

4) เปรียบเทียบ z_s กับ z_s ของ polygon ใดน้อยกว่าแสดงว่าอยู่ใกล้จุดมอง polygonนั้นจะมองเห็น ส่วนอีก polygon หนึ่งจะถูกบัง

บทที่ 8

Shading Techniques

การสร้างโมเดล 3 มิติ ซึ่งจะต้องแสดงบนอุปกรณ์แสดงผล 2 มิติ ขบวนการนี้เรียกว่า *rendering* ซึ่งประกอบไปด้วย 3 ขั้นตอนดังต่อไปนี้

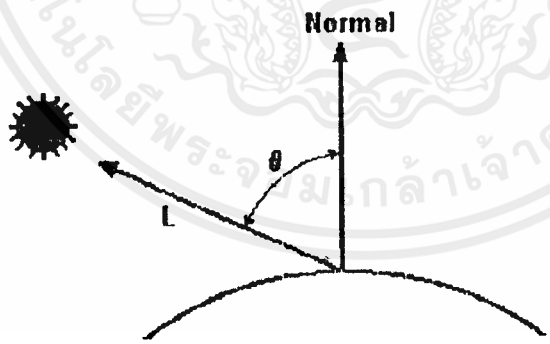
1) โปรเจคภาพ 3 มิติลงบนระนาบ 2 มิติ
2) คำนวณหาว่าพื้นผิวไหนเห็นทั้งหมดหรือเห็นเป็นบางส่วน
3) คำนวณความเข้ม(หรือสีสัน) ของแต่ละจุดในภาพ

ในทางปฏิบัติแล้วขั้นตอนต่าง ๆ จะไม่แยกจากกัน เวลาส่วนใหญ่จะได้ใช้ไปกับการทำ shading ซึ่งวัตถุต่าง ๆ มักจะถูกสร้างขึ้นจากเส้นแสดงขอบเขตของพื้นผิว วัตถุส่วนใหญ่ของจริงมักจะมีผิวเรียบและโค้งมน ดังนั้นในการ render จะต้องจัดรูปทรงของวัตถุผิวโค้งให้ประกอบไปด้วยพื้นผิวเรียบเล็ก ๆ จำนวนมาก แต่ถ้าทำเช่นนั้นจะทำให้โมเดลมีขนาดใหญ่และเพิ่มเนื้อที่ในการจัดเก็บและเวลาในการคำนวณมากขึ้น แทนที่จะทำเช่นนี้ เราได้สร้าง smooth shading model ในการทำ render ซึ่งมีด้วยกันหลายโมเดลคือ

1. Basic Shading Model

เป็นวิธีที่ง่ายที่สุด โดยการสมมติว่าแสงตกกระทบวัตถุ และสะท้อนกลับออกมาในทุกทิศทุกทาง ความเข้มของแสงที่สะท้อนในทิศทางต่าง ๆ แสดงได้โดยกฎของ Lambert

$$I_\theta = I_L K_D \cos\theta$$



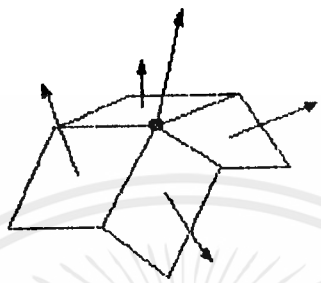
รูป 8.1 Basic shading model

ซึ่ง I_θ คือความเข้มของแสงที่สะท้อนออกมาเป็นมุม θ จาก normal vector ของพื้นผิว I_L คือความเข้มของแสงที่ตกกระทบ และ K_D คือสัมประสิทธิ์ของการสะท้อนของพื้นผิว (diffusivity) ซึ่งขึ้นกับคุณสมบัติของพื้นผิวว่าเรียบ, มัน หรือด้าน

วิธีนี้มีข้อเสียคือ object ที่ไม่ได้รับแสงโดยตรงจะมองไม่เห็นเลยซึ่งในความเป็นจริงแล้วยังมีแสงสะท้อนจากวัตถุรอบ ๆ ข้าง (ambient illumination) มาตกกระทบ ซึ่งจะต้องรวมแสงสะท้อนจากวัตถุรอบ ๆ นี้เป็นเทอมเดียวแล้วทำการคำนวณ

2. Gouraud Shading

พื้นผิวที่อยู่ติดกัน แต่ละพื้นผิวจะมี normal vector ซึ่งบอกทิศทางของพื้นผิว ที่จุดซึ่งพื้นผิวมาบรรจบกัน ถ้าเราเอา normal vector ของพื้นผิวแต่ละอันมาหาค่าเฉลี่ย จะได้ค่า normal vector ถาวรสำหรับกลุ่มพื้นผิวที่อยู่ติดกัน เสร็จแล้วกระจายความเข้มที่มี เดิมเราจะคำนวณความเข้มทีละพื้นผิว เช่น 4 พื้นผิวก็จะได้ 4 ค่า ซึ่งแตกต่างกันชัดเจน เมื่อแสดงออกมาก็จะเห็นเป็นสัน ถ้าต้องการให้ดูเรียบก็จะต้องกระจายความเข้มออกไปทั้ง 4 พื้นผิว แต่วิธีนี้มีข้อเสียคือ ถ้าภาพมีสันจริง ๆ จะทำให้สันที่ควรจะเห็นมองไม่เห็นไปเรียกว่า Mach Band

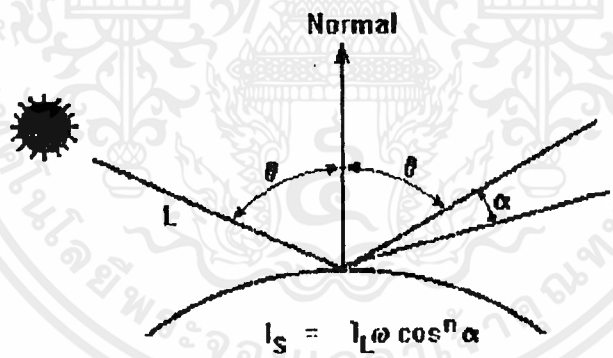


รูป 8.2 แสดงพื้ผิวที่อยู่ติดกันและ normal vectors ของแต่ละพื้นผิว

3. Phong Shading

เป็นวิธีที่ใช้แก้ไข Gouraud Shading โดยใช้ different interpolation technique ซึ่งอธิบายได้ด้วยสมการ

$$I_S = I_L \omega \cos^n \alpha$$



$$I_S = I_L \omega \cos^n \alpha$$

รูป 8.3 Phong shading model

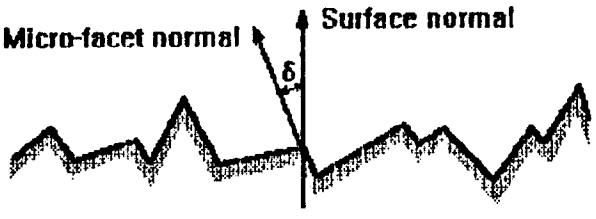
I_S คือความเข้มของแสงที่สะท้อนเข้าตา I_L คือความเข้มของแสงที่มาตกกระทบ ω คือ reflactance คล้ายกับค่า K_D แสดงคุณลักษณะการสะท้อนของวัตถุ ส่วน n คือ เป็นความกว้างของคลื่นรังสี ถ้า n น้อย ๆ จะทำให้ภาพสว่าง ซึ่งโดยปรกติมักจะให้ค่า n ประมาณ 10 α คือมุมที่แสงสะท้อนทำมุมกับแสงที่เข้าตา

4. Torrance-Cook Shading

เป็นการพิจารณาถึงผิวละเอียดของผิวของวัตถุ โดยที่จริง ๆ แล้วผิวจะไม่เรียบจริง และในการขรุขระของมันก็ขรุขระอย่างไม่มีแบบแผน จะต้องสมมติว่าขรุขระในลักษณะไหน อันที่ขรุขระจริง ๆ แล้วมันก็มี micro facetnormal ทำมุม δ กับ normal vector surfave ซึ่งถ้ามองผิวเผินจะเรียบแต่ทาง model บอกว่าไม่เรียบจริง model ที่กำหนดได้ของพื้นผิวนั้นใช้สมการการกระจายของ Beckmann สำหรับการคำนวณหาพื้นผิวก็คือทิศทางของพื้นผิว เมื่อได้ทิศทางแล้วก็

สามารถไปคำนวณ shading ต่อไปได้ ซึ่งภาพที่ได้จะสมจริงยิ่งขึ้น

$$\frac{1}{m^2\cos^4\delta}\, e^{-\tan(\delta/m)}$$



รูป 8.4 The Torrance-Cook surface model

ซึ่ง δ เป็นมุมระหว่าง normal ของพื้นผิวกับ micro-facet normal และ m คือ r.m.s ความขรุขระของพื้นผิวเล็กๆ สำหรับค่า m มีข้อแตกต่างระหว่างโมเดลนี้กับ Phong Shading แต่เราสามารถควบคุมการมองพื้นผิว โดยแสดงพื้นผิวเป็นพลาสติกหรือเป็นโลหะโดยการเปลี่ยนแปลงค่าพารามิเตอร์

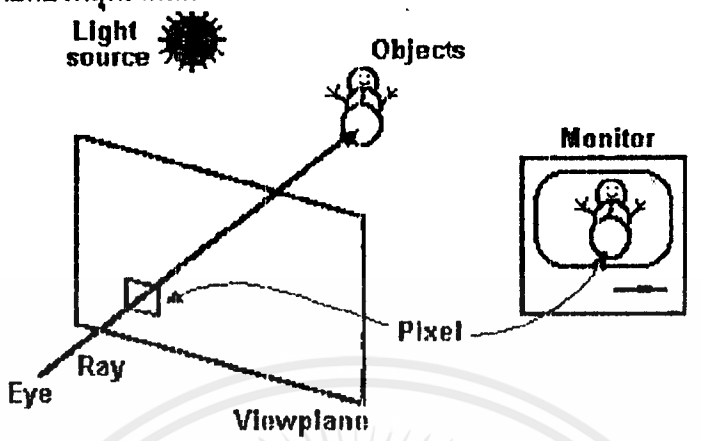
5. Ray-traced rendering

เป็นการทดสอบว่ามองรูปอยู่ตรงนี้ มีแสงจากที่ใดตกมากระทบและหักเหที่พื้นผิวแต่ละพื้นผิว (มองแต่ละจุดแล้ว trace ดูว่าจุดนั้นมีแสงมาจากที่ใดบ้าง) แล้วรวมแสงทั้งหมดคำนวณออกมาเป็นสีที่จุดนั้น วิธีที่ละเอียดที่สุดโดยการติดตามรังสีจากตาไปยังภาพ ซึ่งแสงที่สะท้อนกลับออกมา การหักเหของแสง และเงาจะปรากฏออกมาอัตโนมัติ แต่วิธีนี้ใช้เวลาสูงเพราะต้องคำนวณทุกจุด

วิธีนี้ไม่ได้ฉายภาพลงบนเพลน 2 มิติก่อน แต่กระทำบนโคออร์ดิเนท 3 มิติ ด้วยเหตุนี้จึงเรียกวิธีนี้ว่า object space techniques ส่วนวิธีอื่นที่ได้กล่าวมาแล้ว เรียกว่า Image space techniques ซึ่งจะต้อง map 3D มาเป็น 2D ก่อน Ray Tracing หรือจะเรียกเป็นไทย ๆ ว่าการติดตามรังสี ก็คือความพยายามที่จะจำลองรังสีของแสงในระนาบ 3 มิติ เพื่อจะได้พรรณาถึงรูปลักษณ์ของวัตถุ เช่น ทรงกลม รูปทรงหลายเหลี่ยมหรือกล่อง เป็นต้น ด้วยแหล่งกำเนิดแสงที่ถูกต้อง โดยแหล่งกำเนิดแสงก็คือจุดที่มีการแผ่รังสีของแสงออกมาทั่วทุกทิศทาง

ดังนั้นเราจะถือว่าวัตถุจะไม่มีการแปล่งแสงออกจากตัวเอง และแหล่งกำเนิดแสงก็ไม่จำเป็นต้องอยู่ในทิศของการมองเห็น(และต้องคำนึงถึงแสงที่แผ่ออกจากผิวของวัตถุอื่นที่อาจได้จากการคำนวณแต่ในการเขียนโปรแกรมเรามักตัดทิ้ง)เราก็จะเห็นภาพจากจุดในมิติที่ว่าง นั่นคือตาของเราโดยมองผ่านกรอบหน้าต่างในมิติที่ว่างซึ่งก็คือ *"ระนาบการมองเห็น (View plane)"* ซึ่งภาพที่ปรากฏบนระนาบการมองเห็นในระนาบ 3 มิติก็คือ ภาพที่จะแสดงบนจอภาพโดยอยู่ระหว่างจุดบนจอภาพกับจุดบนระนาบการมองเห็น ดังรูปที่ 8.5 วิธีที่ตรงที่สุดในการสร้างภาพก็คือการติดตามองค์ประกอบของแสง (Photo) จากแหล่งกำเนิดแสงไปยังวัตถุเรียกว่า *การติดตามรังสีแบบไปข้างหน้า (Forward ray tracing)* แต่การทำ Forward ray tracing ก็ไม่อาจใช้ได้จริงเสมอไป เช่น ในกรณีที่เราติดตามการเดินทางของรังสีของแสงที่ตกกระทบวัตถุเงาจากนั้นติดตามแสงที่สะท้อนจากวัตถุนั้นไปจนแสงนั้นเข้าสู่ตาเรา จะเห็นว่า หากใช้วิธีนี้แล้ว เราจะสามารถสร้างภาพได้ช้ามาก เพราะจะต้องพิจารณาทุก ๆ รังสีที่ออกจากแหล่งกำเนิดดังนั้นจึงมีการคิดค้นกระบวนการใหม่ที่มีประสิทธิภาพและง่ายกว่าเดิม โดยมีขั้นตอนที่ย้อนกลับจากกระบวนการการติดตามรังสีแบบไปข้างหน้ากล่าว คือ เราจะเลือกจุดบนระนาบการมองเห็นซึ่งอยู่บนเส้นรังสีที่พุ่งออกจากตาไปนังระนาบการมองเห็นหรือที่เรียกว่า *รังสีตา (Eye Ray)* จุดแรกบนรังสีนี้จะเป็นจุดที่เห็นบนระนาบการมองเห็น เมื่อรวมจุดเข้าด้วยกันแล้วก็จะได้ภาพขึ้นวิธีนี้เรียกว่า *การติดตามรังสีแบบย้อนกลับ(ReverseRay Tracing)* เพราะเป็นการติดตามรังสีจากตากลับไปยังระนาบและเช่นเดียวกับระนาบการมอง

เห็นจอภาพจะสร้างภาพจากการรวมกริดสี่เหลี่ยมที่เรียกว่า พิกเซล (pixel) เราจะต้องหาว่าแต่ละพิกเซลบนจอภาพนั้นควรจะแสดงสีอะไรจึงจะเป็นสีที่ถูกต้องของวัตถุจริงที่ถูกฟุ้งขนโดยรังสีของแสง ดังเช่นในรูปที่ 8.5 จะเห็นรังสีฟุ้งออกจากตาผ่านพิกเซลบนจอภาพ แล้วกระทบกับวัตถุทรงกลม



รูปที่ 8.5 แสดงลักษณะพิกเซลบนระนาบการมองเห็นและภาพที่ปรากฏบนจอภาพ

แสงตกกระทบ

เมื่อรู้จักกับการเกิดจุดบนระนาบการมองเห็นแล้วเราก็จะมาดูวิธีหาสีที่ถูกต้องของวัตถุ สมมติว่ารังสีจากตาตกกระทบวัตถุทรงกลมสีขาวลูกหนึ่งที่จุด P เราต้องการหาสีที่แท้จริงของแสงที่ออกจาก P และฟุ้งกลับไปยังดวงตา ซึ่งสามารถทำได้โดยวิธีการติดตามรังสีแบบย้อนกลับ เนื่องจากวัตถุที่แสงตกกระทบเป็นสีขาว ดังนั้น สีของแสงที่สะท้อนออกมาจาก P จะเป็นสีเดียวกับแสงที่ส่องจากแหล่งกำเนิดแสงมายังจุด P เราเรียกว่า *แสงตกกระทบ (Incident Light)* ในกรณีนี้เราจะถือว่าวัตถุจะไม่เปล่งแสงออกจากตัวเองได้เราสามารถแยกแสงที่ตกกระทบวัตถุนี้ออกเป็น 2 ประเภท คือ แสงมากที่สุดที่สามารถสะท้อนออกจากผิว และแสงที่ส่องทะลุผ่านผิวในกรณีที่วัตถุทึบแสง นอกจากนี้ การสะท้อนหรือการทะลุผ่านพื้นผิวของแสงนี้ยังสามารถแบ่งย่อยออกเป็นอีก 2 ชนิดคือ *Specular Propagation* ซึ่งเป็นการสะท้อนส่องผ่านอย่างเป็นระเบียบเช่นเดียวกับการกระเด้งกลับของลูกบาสบนพื้นเรียบโดยจะมีมุมสะท้อนเท่ากับมุมตกกระทบกับอีกชนิดคือ *Diffuse Propagation* จะเป็นการสะท้อนแบบกระจัดกระจายในลักษณะของลูกบาสที่กระเด้งจากพื้นขรุขระที่เราไม่สามารถรู้ทิศทางที่แน่นอนของลูกบาสและถ้าเราส่งลูกบาสให้กระเด้งออกจากพื้นที่นี้หลายๆ ลูกจะเห็นว่าลูกบาสกระจายออกในทุกๆ ทิศทาง ในกรณีที่เป็นแสงการสะท้อนดังกล่าวจะมีความเข้มของแสงเท่าๆ กันในทุกทิศทุกทาง แต่จะน้อยกว่าความเข้มของแสงตกกระทบในบางครั้งเราจะเรียก Diffuse Propagation ว่า Diffuse Scattering สรุปแล้วแสงจะแผ่ออกจากวัตถุได้ 4 วิธีคือ การสะท้อนแบบเป็นระเบียบและไม่เป็นระเบียบ กับ การส่องผ่านทั้งแบบเป็นระเบียบและไม่เป็นระเบียบ อีกสิ่งหนึ่งที่เราไม่สามารถละทิ้งได้ในเมื่อพูดถึงการสะท้อนของแสงนั่นคือต้นกำเนิดของแสงที่ตกกระทบวัตถุนั้นมาจากที่ใด แหล่งกำเนิดแสงจะมีได้ 2 ชนิด คือ แสงจากแหล่งกำเนิดและแสงที่เกิดจากการสะท้อนจะพื้นผิวอื่น ซึ่งทั้ง 2 แหล่งสามารถเกิดขึ้นพร้อมกันได้ การพิจารณาการสะท้อนและแหล่งกำเนิดแสงนี้ จะนำไปสู่การระบายสี หรือ Shading ต่อไป

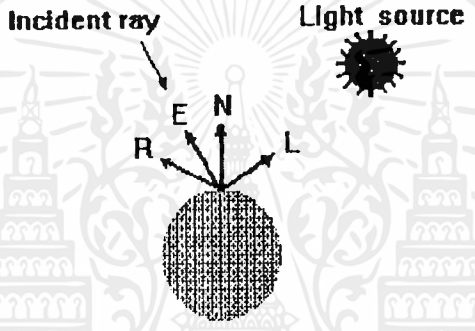
แสงที่ส่องตรงจากแหล่งกำเนิด

ในการพิจารณาแสงที่ส่องตรงจากแหล่งกำเนิดสิ่งแรกที่จะต้องทำก็คือพิจารณาว่ามีแหล่งกำเนิดแสงแหล่งใดบ้างที่ส่องแสงไปยังวัตถุอยู่ถ้าจุด P ซึ่งเป็นจุดหนึ่งบนวัตถุที่เราพิจารณาถูกส่องโดยแหล่งกำเนิดแสง S แล้วเราจะสามารถบอกได้ว่า ที่แหล่งกำเนิดแสง S เราจะมองเห็นจุด P และ ในทางกลับกันที่จุด P เราจะมองเห็นแหล่งกำเนิดแสง S เช่น

กัน รังสีที่มีทิศทางจากจุด P พุ่งไปยังแหล่งกำเนิด S นี้จะเรียกว่ารังสีเงา หรือ Shadow Ray ในการพิจารณาแสงตรงนี้เราจะใช้รังสีเงาแทนที่จะเป็นรังสีที่ออกจากแหล่งกำเนิดตามวิธี Reverse Ray Tracing ถ้ารังสีนี้กลับสู่รโดยไม่ชนกับวัตถุใดก่อนแล้วเราสามารถหาได้ว่ามีแสงจาก S ตกกระทบจุด P แล้วสะท้อนเข้าสู่ตาของเราในปริมาณเท่าใด แต่ถ้ารังสีจาก P ถึง S ถูกขัดขวาง เช่น รังสีตัดผ่านวัตถุอื่นเสียก่อนแล้ว P จะอยู่ในเงาที่เกิดจากแหล่งกำเนิด S ดังนั้นเราไม่จำเป็นที่จะต้องพิจารณาผลที่เกิดขึ้นกับจุด P เนื่องจากแหล่งกำเนิด S อีกต่อไป

การระบายสี

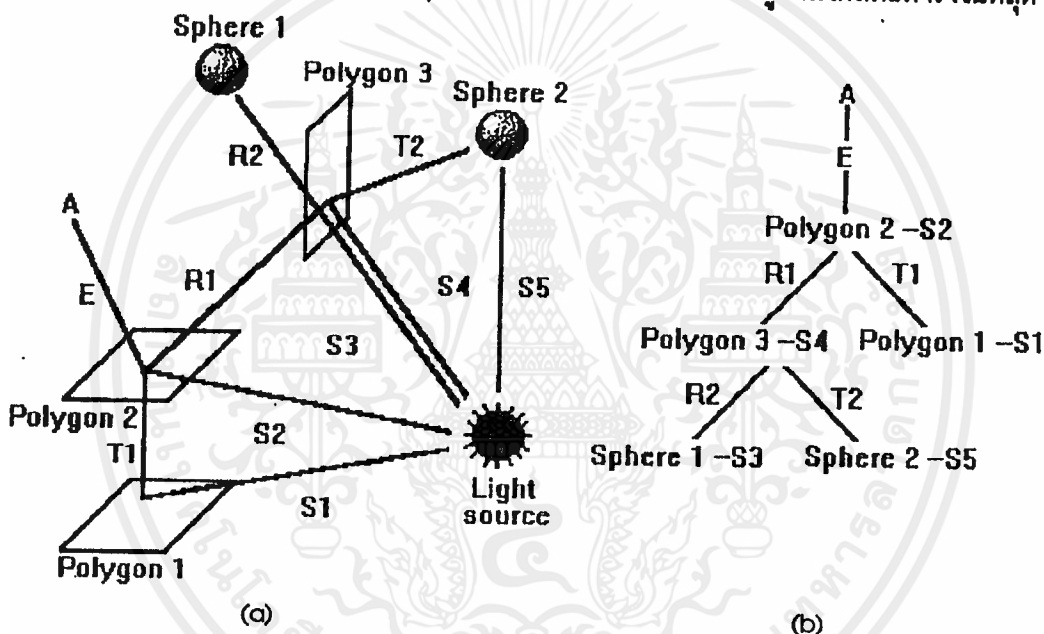
สมมุติให้จุด P ถูกส่องโดยแสงจากแหล่งกำเนิด S โดยทั่วไปแล้ว จุดบนผิวของวัตถุทั่วไปนั้นจะมีคุณสมบัติที่สำคัญประการหนึ่งคือเวกเตอร์ของพื้นผิววัตถุที่จุดใด ๆ จะพุ่งออกจากผิววัตถุและจะตั้งฉากกับระนาบสัมผัสที่จุดนั้นเสมอ อย่างเช่น พื้นผิวทรงกลมจะมีเวกเตอร์ของพื้นผิวในทิศพุ่งออกจากจุดศูนย์กลางเสมอดังรูปที่ 8.6 เราเรียกเวกเตอร์นี้ว่า Normal Vector ในรูปคือ เวกเตอร์ N ส่วนเวกเตอร์ L เป็นรังสีเงาจากจุด P ไปยังแหล่งกำเนิด S ซึ่งเราจะพิจารณารังสีสะท้อนของ L เทียบกับ Normal Vector N เสมอ



รูปที่ 8.6 การคำนวณเพื่อทำ Shading บนทรงกลมเนื่องจากรังสี L ที่พุ่งสู่แหล่งกำเนิด ซึ่งจะสะท้อนไปตามทิศทางของเวกเตอร์ R ซึ่งเวกเตอร์ทั้งสองจะทำมุมกับ N ที่เป็นเวกเตอร์ตั้งฉากพื้นผิวที่จุด P เท่า ๆ กัน

ในการหาตำแหน่งของแหล่งกำเนิดแสง S เทียบกับจุด P ที่เราพิจารณานั้น สามารถกระทำได้โดยการหา Dot Product ของเวกเตอร์ N และเวกเตอร์ L นั่นคือ ถ้า N.L < 0 แล้ว S จะอยู่ด้านหลังของผิว ถ้า N.L > 0 แล้ว S จะอยู่ด้านหน้าของผิว และถ้า N.L = 0 แล้ว S จะอยู่บนระนาบสัมผัสกับผิวและแสงที่ตกกระทบที่จุดPจะสะท้อนกลับไปยังแหล่งกำเนิด S หมด เราจะสมมุติให้ N.L > 0 ดังนั้นซึ่งจะทำให้เกิดการสะท้อนกลับของแสงจากพื้นผิว เราสามารถหาความเข้มของแสงที่สะท้อนอย่างไม่เป็นระเบียบจากพื้นผิวได้โดยจะกำหนดให้ความเข้มของแหล่งกำเนิดแสง S มีค่าเป็น I และกำหนดให้ DR เป็นค่าของความเข้มแสงที่สะท้อนกลับจากพื้นผิว เราจะได้ความสัมพันธ์ระหว่าง DR และ I เป็น DR = I(N.L) สมการนี้ได้เรารู้จักกันดีในนามของ Lambert' & Law ในการหาจุดที่เป็นจุดที่สว่างที่สุดบนวัตถุนั้นเราจะถือหลักการที่ว่าจุดที่เกิดการสะท้อนกลับมาสู่ตาของผู้สังเกตโดยตรงจะเป็นจุดที่มีความสว่างมากที่สุดซึ่งจะขึ้นกับตำแหน่งที่อยู่ของตาด้วย ส่วนวิธีการหานั้นเราจะสร้างเวกเตอร์ E จากจุด P ไปยังดวงตา ดังรูปที่ 2 โดย L จะพุ่งเข้าสู่แหล่งกำเนิดแสงและ R เป็นทิศของแสงที่สะท้อนจากพื้นผิว จากกฎของการสะท้อนอย่างเป็นระเบียบเราจะได้ความสัมพันธ์ของ L,R และ N ว่า R = 2Nd-L เมื่อ d = N.L ทำให้สามารถหาจำนวนรังสีที่พุ่งเข้าสู่ตาได้โดยการหาว่ามี R และ E เกิดขึ้นเท่าไรด้วยสมการ $SR = I (E.R)^k$ เมื่อ SR คือค่าความเข้มของการสะท้อนกลับอย่างเป็นระเบียบ และ k คือค่าความหยาบของพื้นผิวขั้นต่อไปคือการหาแสงตกกระทบที่มาจากวัตถุอื่นโดยใช้รังสีตาพิจารณาในทิศทางย้อนรังสีแสงด้วยวิธี Reverse Ray Tracing ว่ามีแสงใดที่จะสะท้อนกลับสู่ตาบ้างจากสมการ R' = 2N(E.N) - E เหตุที่ใช้ R' แทน R ก็เพราะใน

การคำนวณนี้ เราใช้วิธีการย้อนกลับของรังสี ดังนั้นแสงที่สะท้อนกลับจาก E ซึ่งเป็นรังสีตาจะเขียนในรูป R' ต่อไปเราก็จะพิจารณาว่ามีรังสีใดที่พุ่งมาตามทิศทาง R' บ้างโดยการย้อนทิศของรังสีตกกระทบที่พุ่งจากวัตถุมายังตา แล้วพิจารณาว่าที่จุดนั้นได้รับแสงสะท้อนจากวัตถุใดเป็นสีใดบ้าง สีที่เรามองเห็นจะเป็นสีที่เกิดจากการผสมของสีเหล่านี้ในการคำนวณจริง จะมีแสงสะท้อนจากวัตถุหลายชิ้น และแต่ละชิ้นก็จะมีการสะท้อนเกิดขึ้นเช่นกัน ซึ่งจะทำให้การคำนวณเป็นไปได้ยาก เราอาจจะใช้วิธีเขียนแผนภูมิต้นไม้ของรังสีดังรูปที่ 8.7 (b) ขึ้นเพื่อให้สามารถหาสีที่แท้จริงได้ง่ายขึ้น โดยพิจารณาจากรังสีสะท้อนโดยตรงไล่ไป ตัวอย่างเช่น เราอยู่ที่จุด A และกำลังพิจารณาวัตถุ Polygon 2 ซึ่งมีรังสี R1 และ T1 ตกกระทบอยู่ และจะเห็นว่ารังสี R1 นั้นเกิดจากรังสีสะท้อน R2 กับ T2 และจากแหล่งกำเนิดแสงตรง S4 ในขณะเดียวกัน R2 จากทรงกลม Sphere 1 ก็เกิดจากรังสีจากแหล่งกำเนิด S3 และรังสี T2 ก็เกิดจากทรงกลม Sphere2 ซึ่งมีรังสี S5 จากแหล่งกำเนิดตกกระทบอยู่ เราสามารถเขียนแผนภูมิต้นไม้ของรังสีได้ดังรูป 8.7(b) ข้างๆ จากนั้นเราจะค่อย ๆ คำนวณหาสีที่แท้จริงจากรังสีต่าง ๆ ไล่ขึ้นมาจากส่วนล่างของแผนภูมิจนได้สีที่แท้จริงในที่สุด

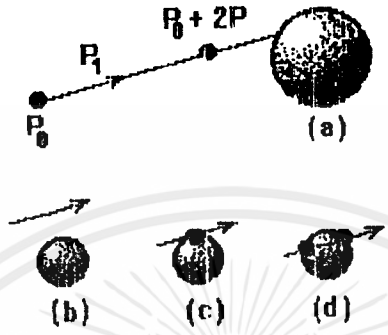


รูปที่ 8.7 การหาสีที่แท้จริงด้วยวิธีแผนภูมิต้นไม้ของรังสี

<u>การหาจุดตัดของรังสีกับวัตถุ</u>

การหาจุดตัดของรังสีกับวัตถุเป็นหนึ่งในวิธีการที่สำคัญในการติดตามรังสี การหาวัตถุที่ถูกตัดโดยรังสีเป็นวัตถุแรกจะเป็นขั้นตอนที่เสียเวลามากวิธีที่จะหาได้โดยตรงคือการหาจุดตัดของรังสีกับวัตถุแต่ละชิ้นแล้วทำการหาวัตถุที่อยู่ใกล้กับแหล่งกำเนิดในทิศทางของรังสีนั้น ขบวนการนี้ใช้เวลามากและน่าเบื่อหน่ายแต่ก็ทำงานได้ดี สมมุติว่าเราต้องการหาการตัดของรังสีกับทรงกลมเราจะสมมุติให้รังสีเริ่มจากจุด P_0 มีทิศไปตาม P_1 จุด P ใด ๆ ที่อยู่บนรังสีจะหาได้จากสมการ $P = P_0 + P_1 t$ รูป 8.8(a) แสดงตัวอย่างรังสีที่ตกกระทบทรงกลมที่มีจุดศูนย์กลางที่ C รัศมี r กำหนดให้จุดใด ๆ บนทรงกลมนี้เป็นจุด Q ซึ่งสามารถหาได้จากสมการ $(Q-C).(Q-C) - r^2 = 0$ ถ้ารังสีตัดกับทรงกลม เราจะสามารถหาจุด Q บนรังสีนั้นและที่ผิวทรงกลมพร้อม ๆ กัน ดังนั้นถ้าจุด Q อยู่บนเส้นรังสีแล้วเราสามารถเขียนได้ว่า $Q = P_0 + P_1 t$ จัดรูปสมการจากสมการของจุด Q บนทรงกลม เราจะได้สมการ $(P_0+P_1 t).(P_0+P_1 t)-r^2 = 0$ ทำการแก้สมการและจัดรูปสมการให้เหมาะสมโดยเขียนสมการในเทอมของ t เพราะเราจะใช้ค่า t ในการหาจุดตัดที่เกิดขึ้นบนรังสี โดยพิจารณาจากสมการ Quadratic $at_2 + bt + c = 0$ จะได้คำตอบจากสมการ 2 คำตอบคือ $t_1 = (-b+d)/2a$ และ $t_2 = (-b-d)/2a$ เมื่อ d = b2-4ac ค่าของ d สำคัญมาก เราทราบมาว่าจำนวนลบจะไม่สามารถหารากที่ 2

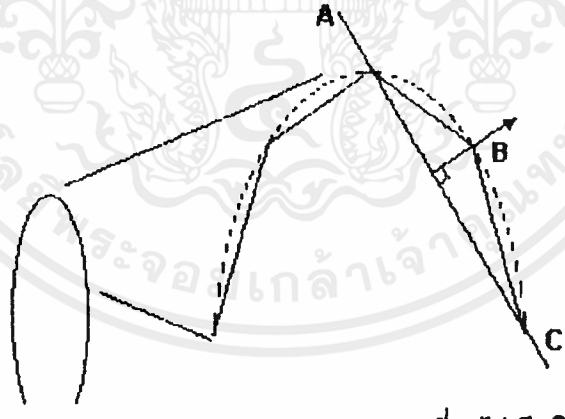
ที่เป็นจำนวนจริงได้ดังนั้นหากค่า d มีค่าเป็นลบ เราจะไม่สามารถหาค่าของ t ที่เป็นจำนวนจริงได้ หรือในทางฟิสิกส์ก็คือ รังสีจะไม่ชนวัตถุนั่นเอง ดังรูป 8.8(b) ถ้า d = 0 รังสีจะวิ่งเฉียดผิวทรงกลมไป 1 จุด ดังรูป 8.8(c) แต่ถ้า d มากกว่า 0 แล้วรังสีจะตัดผ่านทรงกลมทั้งกลมทั้งสองด้าน ดังรูป 8.8(d) และจะหาจุดตัดได้โดยการแทนค่า t ลงในสมการ $P=P_0+P_1t$ ถ้ามีจุดตัด 2 จุด ก็เลือกเอาจุดที่ค่า t น้อยที่สุด (ค่า t จะต้องเป็นบวก ถ้าเป็นลบทั้งคู่แล้วจุดตัดจะอยู่หลังจุดกำเนิดของรังสี)



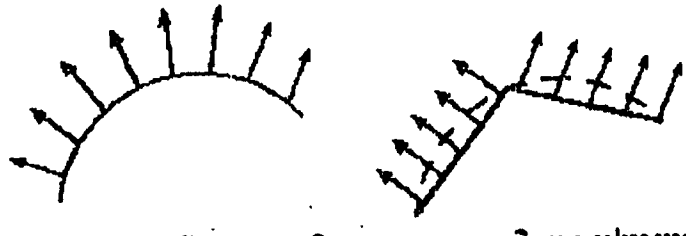
รูปที่ 8.8 แสดงการตัดกันของรังสี

<u>ความยุ่งยากของการแสดงผล</u>

วิธีง่ายๆ สำหรับการติดตามรังสีคือ การสร้างสัญญาณรังสีของแต่ละพิกเซลบนจอ แล้วใส่สัญญาณสีลงที่พิกเซลนั้น อย่างไรก็ตามจอสีจะมีข้อจำกัดในด้านจำนวนสีที่สามารถแสดงได้ในขณะที่บางครั้งสีที่ได้จากการคำนวณมีปริมาณมากกว่าแล้วเราจะทำอย่างไรให้ได้วิธีที่ดีสุด คำตอบก็คือเราต้องรวมเอาสีที่อยู่ในบริเวณใกล้เคียงนำมาเฉลี่ยเข้าด้วยกันจะได้ค่าของสีที่ควรจะใส่ลงไปในแต่ละพิกเซล



รูปที่ 8.9 แสดงการปรับผิวของวัตถุจากผิวของรูปทรงหลายเลี่ยมไปเป็นผิวแบบ quadatic



รูปที่ 8.10 เปรียบเทียบการสะท้อนแสงจากผิว quadatic และผิวแบบรูปทรงหลายเหลี่ยม

สำหรับการเริ่มต้นของการทำการติดตามรังสีสิ่งแรกอาจเริ่มจากการสร้างรูปทรงหลายเหลี่ยมจำนวนมากมายก่อนเมื่อทำการคำนวนแล้วจึงค่อยปรับเป็นผิวแบบ Quadratic (ผิวที่มีความโค้ง) เพราะการตัดของรังสีในรูปทรงหลายเหลี่ยมใช้คำนวณระนาบของรังสีได้ง่ายกว่า การตัดผ่านทรงกลมชั้นที่สองถ้าจุดตัดมีจริงภายในบริเวณปิดของระนาบของรูปทรงหลายเหลี่ยม มีหลายวิธีที่ใช้ในการแก้ปัญหา การทดสอบการตัดของรังสีผ่านทรงกลมนั้นยากต่อการทำความเข้าใจจึงจะไม่ขอกล่าวในที่นี้เพราะมีกรณีเกิดขึ้นมากมายในการคำนวณส่วนปัญหาต่อไปที่จะเจอก็ คือประสิทธิภาพในการสร้างรูป

บทที่ 9

หลักการของการทำ Computer Animation

พื้นฐานของ Animation

การสร้างภาพเอนิเมชันทั่ว ๆ ไป การสร้างภาพเอนิเมชันในคอมพิวเตอร์ และภาพในวิดีโอหรือแม้แต่ภาพยนต์มีทฤษฎีในการสร้างที่เหมือนกันเพื่อให้ได้ภาพที่เคลื่อนไหวได้ เมื่อภาพนิ่งจำนวนหลาย ๆภาพ ที่มีลักษณะต่อเนื่องกันนำมาแสดงต่อเนื่องกันโดยที่มีความเร็วเพียงพอ สายตาของมนุษย์จะมองไม่ออกว่าเป็นภาพนิ่งหลาย ๆภาพ แต่จะมองเป็นภาพที่เคลื่อนไหว โดยทั่วไปแล้วภาพที่แสดงบนจอคอมพิวเตอร์จะใช้ความเร็วในการแสดงภาพจำนวน 10 ถึง 15 ภาพต่อวินาที

ชนิดของ Animation

การทำภาพเคลื่อนไหวโดยใช้เครื่องคอมพิวเตอร์นั้นแบ่งออกได้เป็นประเภทหลัก ๆ 3 ประการคือ

1.BITBLT Animation หรือบางครั้งเราจะเรียกเป็น Block Graphics, Partial screen Animation (การทำภาพเคลื่อนไหวบางส่วนของจอ) ฯลฯ หลักการของการทำภาพเคลื่อนไหวแบบนี้คือ จะทำการย้ายกรอบข้อมูลบางส่วนของจอไปตามจุดต่าง ๆ ที่กำหนด โดยย้ายข้อมูลเป็นบล็อคสี่เหลี่ยมเล็ก ๆการทำ Animation แบบนี้จะมีความเร็วค่อนข้างสูง เนื่องจากส่วนที่คอมพิวเตอร์ต้องจัดการในการทำภาพเคลื่อนไหวนั้นเป็นการย้ายข้อมูลจำนวนไม่มากนัก แต่จะต้องทำการเก็บฉากหลังที่บล็อกของภาพเคลื่อนที่ผ่านไปด้วย การทำ Animation แบบนี้สามารถใช้กับจอภาพกราฟิกได้ทุกชนิดตั้งแต่จอภาพความละเอียดสูงที่สร้างสีได้หลายสีจนกระทั่งถึงจอภาพแบบรายละเอียดต่ำหรือจอภาพที่มีสีเดียว ส่วนใหญ่ของการใช้งาน BITBLT Animation นั้นคือ งานที่ต้องใช้ความเร็วสูง มีการเคลื่อนไหวเฉพาะตัวละครบางตัวเท่านั้น เช่น เกมส์, การจำลองการเคลื่อนไหวของวัตถุ ฯลฯ

2.Frame Animation บางทีอาจเรียกว่าการทำภาพเคลื่อนไหวแบบเต็มจอ (Full Screen Animation) หรือ Page Animation หลักการทำ Animation แบบนี้คือสร้างภาพที่ต้องการให้เคลื่อนไหวเก็บไว้ก่อนเป็นภาพ ๆ ทีละเฟรม แบบการถ่ายทำภาพยนต์ เมื่อต้องการให้ภาพเคลื่อนไหวก็นำภาพนั้นขึ้นมาแสดงบนจอทีละภาพ โดยใช้ความเร็วสูงเหมือนกับการฉายภาพยนตร์(ประมาณ30ภาพต่อวินาที) Frame Animation นั้น เป็นทางเลือกที่ดีที่สุด สำหรับการทำAnimation คุณภาพสูง เนื่องจากเราสามารถสร้างภาพที่มีความสวยงามซับซ้อนโดยใช้เทคนิคการทำภาพแบบต่าง ๆ เช่นการทำ 3D Modeling ร่วมกับการทำ Rendering และ Ray Tracing (เป็นการสร้างภาพ 3 มิติที่มีแสงเงาเหมือนจริงบนคอมพิวเตอร์ทีละเฟรมการทำ Cel Animation , การทำ Morphing ในภาพยนตร์เรื่อง T2 ตอนที่คนเหล็กถูกหลอมในไฟเป็นของเหลว โดยสามารถใช้เทคนิคร่วมกันแล้วนำภาพถ่ายจากกล้องมาผสมและตกแต่งภาพได้ทีละเฟรมเหมือนกับการทำภาพยนตร์ในเครื่องดังนั้นจะพบเห็นการทำ Frame Animation ในงานคุณภาพสูง เช่น การทำโฆษณา การทำภาพยนตร์ ฯลฯ

3.Realtime Animation สิ่งที่ไม่เหมือนการทำงานของ BITBLT Animation และ Frame Animation คือ การทำ Animation ทั้งสองแบบที่กล่าวมานั้น เป็นการวาดภาพเก็บไว้ก่อนในหน่วยความจำบางชนิดแล้วค่อยนำมาแสดงผลในภายหลังแต่การทำ Realtime Animation นั้นเป็นการวาดหรือสร้างภาพขณะที่ทำ Animation พร้อมกันไปซึ่งจะต้องใช้คอมพิวเตอร์ที่มีความเร็วค่อนข้างสูงจึงจะสามารถใช้งานได้จริงบางครั้งการทำ Realtime Animation อาจเรียกอีกชื่อว่า Live Animation, Ping-Pong Animation หรือ Dynamic Page-Flipping Animation การทำ Animation แบบนี้

ต้องมีหน่วยความจำแสดงผลอย่างน้อย 2 หน้า (Page) ในการทำภาพเคลื่อนไหว โดยในขณะที่กำลังแสดงภาพในหน้า กก็ทำการคำนวณและวาดภาพต่อไปในหน้าที่สองเมื่อวาดเสร็จก็จะสลับหน้าที่สองขึ้นมาแสดงแล้ววาดภาพที่หน้าที่ งต่อไปเมื่อทำอย่างนี้ซ้ำไปเรื่อย ๆ ก็จะได้การทำภาพเคลื่อนไหวโดยไม่จำเป็นที่จะต้องเก็บภาพที่ต้องการสร้าง ล่วงหน้า การประยุกต์ใช้ Animation แบบนี้นิยมใช้กับระบบจำลองการทำงานใน 3 มิติหรือการทำภาพเคลื่อนไหวโดย ทอบกับผู้ใช้(Interactive Animation)เช่น ระบบจำลองการบิน (FlightSimulation) แต่จะต้องใช้เครื่องที่ค่อนข้างเร็วจึง สามารถทำงานทันเวลา และใช้งานได้จริง

ดีและข้อเสียของการทำ Animation แต่ละแบบ

การทำ Animation แต่ละวิธีนั้นมีข้อเด่นและข้อด้อยต่างกันไป ดังนั้นการเลือกใช้ควรจะระมัดระวัง โดยเลือก รงตามความสามารถและจุดเด่นของมัน เพื่อประสิทธิภาพและผลลัพธ์ที่ดีโดยหัวข้อต่อไปจะเป็นการสรุปข้อดีและข้อ ของการทำ Animation แต่ละแบบ

BITBLT Animation

การทำ BITBLT Animation ถึงแม้จะไม่เร็วกว่า Frame Animation แต่ค่อนข้างประหยัดหน่วยความจำ เนื่อง เครื่องจะเก็บเพียงแค่สิ่งที่ต้องการเคลื่อนไหวบนจอและฉากหลังที่รูปผ่านไปเท่านั้นเช่นภาพรถแข่ง ภาพกระสุน ข้อเสีย สามารถใช้กับงานบางชนิดเท่านั้น คืองานประเภทการย้ายภาพตัวละครในกรอบสี่เหลี่ยมไปมาเท่านั้น

FRAME Animation

การสร้างผลงานด้วย Frame Animation จะมีความเร็วในการแสดงผลเร็วที่สุด แต่เนื่องจากภาพ ะเฟรมมีขนาดใหญ่และมีการแสดงผลอย่างต่ำ 30 ภาพต่อวินาที ดังนั้นจะต้องใช้หน่วยความจำมหาศาลในการเก็บข้อ เช่น ภาพจากจอขนาดความละเอียด 300x200 มีสีได้ 256 สีนั้น (โหมดแสดงผลมาตรฐานของ 3D Studio บน VGA) า จะต้องใช้หน่วยความจำขนาด 60,000ไบต์/ภาพ ถ้าต้องการผลงานคุณภาพสูงแล้วละก็จะต้องใช้จำนวน 30 เฟรม/วิ ดังนั้น 1 วินาที จะต้องการข้อมูลถึง 1,800,000 ไบต์ (หนึ่งล้านแปดแสนไบต์) ซึ่งที่จริงแล้วความละเอียดขนาดนี้ยังไม่ พอในการทำภาพยนตร์ และโฆษณาโดยความละเอียดไม่ควรต่ำกว่า 800 x 600 โดยมีสีอย่างต่ำ 65,536 สี ดังนั้น หน่วยความจำที่ใช้ในการทำ Frame Animation มืออาชีพนั้นจะต้องใช้เครื่องที่มีหน่วยความจำค่อนข้างใหญ่ มีฮาร์ด มหาศาลและต้องพึ่งการบีบข้อมูล (Data Compression)

REALTIME Animation

Real-Time Animation นั้นมีความสามารถหลายประการ แต่มีความเร็วต่ำ ดังนั้นการประยุกต์ใช้งานนั้นจำ ี่จะต้องใช้เครื่องที่ค่อนข้างมีประสิทธิภาพสูงจึงสามารถใช้งานได้ดีตัวอย่างการใช้งานเช่นการทำโปรแกรมจำลองการ ight Simulation) ที่จำเป็นที่จะต้องสร้างภาพที่จำลองเหตุการณ์ตามสถานการณ์ที่เปลี่ยนไปตามการบังคับของนักบิน ป็นการทำงานแบบโต้ตอบ (Interactive)

แนะนำการใช้งานโปรแกรม

1. การ Install โปรแกรม

ใส่แผ่นโปรแกรมหมายเลข 1 ใน disk drive เรียกโปรแกรม install ซึ่งจะทำการ restore program ที่ back up ไว้ ใส่แผ่นโปรแกรมตามหมายเลขที่เขียนไว้จนครบ 52 แผ่น จะได้ subdirectory PROJ3D

copy program GRASPC.EXE ซึ่งอยู่ในแผ่นที่ 52 ลงใน subdirectory PROJ3D

หมายเหตุ แผ่น install โปรแกรมมีจำนวนทั้งสิ้น 52 แผ่น เพราะฉะนั้นจะต้องมีเนื้อที่ว่างใน hard disk อย่างน้อย 63 Mbytes และโปรแกรมมีการแสดงภาพในโหมด 640X480 256 color จึงจะต้องมีการ์ด VGA ที่มีหน่วยความจำอย่างน้อย 307.2 Kbytes

2. การเรียกใช้งานโปรแกรม

- เข้าไปใน subdirectory PROJ3D
- พิมพ์ GRASPC TEXT1

3. เริ่ม RUN โปรแกรม

- ปรากฏไตเติ้ล และแนะนำคณะผู้จัดทำ
- ปรากฏข้อความ

ขอต้อนรับสู่

ค ณ ะ วิ ศ ว ก ร ร ม ศ า ส ต ร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง

- โปรแกรมจะแสดงภาพโดยรวมของคณะฯ โดยเป็นภาพเคลื่อนไหวต่อเนื่องวนซ้ำไปมา

4. เข้าสู่เมนูหลักของโปรแกรม

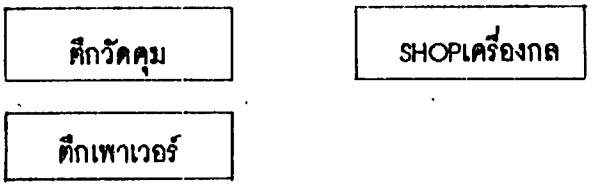
- ขณะแสดงภาพเคลื่อนไหวอยู่กดคีย์ 'M' จะเข้ามาเมนู
- หน้าจอแรกจะแสดงภาพของตึก 4 ตึกดังต่อไปนี้

ตึก 6 ชั้น	ตึกโทรคมนาคม
ห้องสมุด	หอประชุม

- กด 'N' เพื่อเข้าสู่เมนูถัดไป เมนูที่ 2 จะแสดงภาพของตึก 4 ตึกดังต่อไปนี้

ตึกคอมพิวเตอร์	โรงอาหาร
ตึกกิจกรรม	ตึก ท.ส.

- กด 'N' เพื่อเข้าสู่เมนูถัดไป เมนูที่ 3 จะแสดงภาพของตึก 3 ตึกดังต่อไปนี้



- กด 'N' เพื่อเข้าสู่เมนูหลักคือเมนูที่ 1

- กด 'P' เพื่อเข้าสู่เมนูของตึกพระเทพฯ ซึ่งจะแบ่งออกเป็น 6 ZONE (B-F) ซึ่งสามารถเรียกดูแต่ละส่วนได้โดยกดคีย์ 'B' - 'F' ถ้ากดคีย์ 'N' จะกลับสู่เมนูหลัก

- กดคีย์ 'Q' ออกจากโปรแกรม

- การเลือกอาคารในคณะ กดอักษรที่ปรากฏใต้ตึกที่ต้องการแสดง ถ้ายังไม่พบตึกที่ต้องการ กดคีย์ 'N' เพื่อเข้าสู่เมนูถัดไป

5. การแสดงรายละเอียด

หลังจากกดคีย์เลือกตึกที่ต้องการแล้ว จะแสดงการเดินทางไปยังตึกนั้นจากจุดเริ่มต้นคือตึก A (ตึก6 ชั้น) แสดงการหมุนตึกให้เห็นลักษณะภายนอกให้ครบทุกด้าน และมีตัวหนังสือแสดงรายละเอียดเกี่ยวกับ ชื่อเรียก, จำนวนชั้นและแต่ละชั้นประกอบด้วยห้องอะไรบ้าง หลังจากนั้นจะกลับเข้าสู่เมนู

6. การออกจากโปรแกรม

เมื่อต้องการออกจากโปรแกรมกดคีย์ 'Q' ขณะอยู่ในเมนูละไรก็ได้ยกเว้นเมนูของตึกพระเทพฯ

SOURCE PROGRAM

โปรแกรมแบ่งออกเป็น 3 file คือ TEXT1, TEXT2 และ TEXT3 โดย TEXT1 เป็นส่วนของไตเติ้ล TEXT2 เป็นส่วนของการแสดงตึกในคณะและ TEXT3 เป็นส่วนของการแสดงอาคารเรียนรวมตึกพระเทพฯ

```
; ------ SHOW TITLE CAR FLY ------

video l                    ; เลือกกราฟิคโหมด 320x200 256 สี
pload black256.plc 3       ; โหลดรูปที่มี Palette เป็นสีดำทั้ง 256 สี
pload kmitlcar.plc 4       ; โหลดรูปรถวิ่งในคณะ Frame ที่ 1
dload kmitlcar.fll 1 1     ; โหลดภาพเคลื่อนไหวของรถวิ่งในคณะแล้วบินไป
palette 3                  ; เลือกใช้ Palette สีดำทั้ง 256 สี
pfade 0 4                  ; วาดรูปรถวิ่งในคณะ Frame ที่ 1 บนจอภาพ โดยที่จะยังมองไม่เห็นภาพเพราะ Palette เป็นสีดำ
spread 3 4                 ; เปลี่ยน Palette เป็นสีตามรูปรถวิ่งในคณะ Frame 1 โดยที่จะค่อย ๆ เห็นภาพปรากฏขึ้นมา
pfree 3 4                  ; เคลียร์ Buffer หมายเลข 3 และ 4
putdff 1 5 0 999           ; ทำภาพเคลื่อนไหวเริ่มจาก Frame แรกถึงสุดท้าย โดยที่เราจะเห็นรถวิ่ง แล้วบินไป
pload black256.plc 1       ; โหลดรูปที่มี Palette เป็นสีดำทั้ง 256 สี
pload c3dcga.plc 2         ; โหลดรูปที่มีตัวหนังสือว่า 3D Computer Graphics Animation
spread 0 1                 ; ให้จอภาพค่อย ๆ มืดไป
dfree 1                    ; เคลียร์ Buffer หมายเลข 1

; ------ PRINT 3D Computer Graphics Animation & Name DEJ, SINE ------

pfade 0 2                  ; วาดรูปที่มีตัวหนังสือว่า 3D Computer Graphics Animation บนจอภาพ โดยที่จะยังมองไม่
                           ; เห็นภาพเพราะ Palette เป็นสีดำ
spread 1 2                 ; เปลี่ยน Palette เป็นสีตามรูปตัวหนังสือที่อยู่บนจอ โดยที่จะค่อย ๆ เห็นภาพปรากฏขึ้นมา
pfree 2                    ; เคลียร์ Buffer หมายเลข 2
dload c3dcga.fll 1 1       ; โหลดภาพเคลื่อนไหวของตัวหนังสือ
putdff 1 8 0 10            ; ทำภาพเคลื่อนไหวเริ่มจาก Frame แรกถึง Frame ที่ตัวหนังสือมาหยุดตรงกลางจอภาพ โดยที่เรา
                           ; จะเห็นตัวหนังสือ 3D Computer Graphics Animation เคลื่อนไหวแล้วมาหยุดตรงกลางจอภาพ
waitkey 200                ; หน่วงเวลาไว้ 2 วินาที
putdff 1 8 10 20           ; ทำภาพเคลื่อนไหวจาก Frame ที่หยุดจนจบ โดยที่เราจะเห็นตัวหนังสือวิ่งเข้ามาหา แล้วหายไป
dfree 1                    ; เคลียร์ Buffer หมายเลข 2
pload pibik3.plc 3         ; โหลดภาพที่มี Palette ดำหมดทั้ง 16 สี
pload char2.clp 2          ; โหลดภาพตัวหนังสือ "คงเดช กรกาญจนารักษ์"
pfree 1                    ; เคลียร์ Buffer หมายเลข 1

; ------ PRINT NAME

video m                    ; เปลี่ยนโหมดของจอภาพให้เป็น VGA 16 สี
palette 3                  ; ใช้ Palette ดำหมดทั้ง 16 สี
window 172 193 468 287     ; กำหนดกรอบที่ให้เขียนภาพ
pfade 0 2                  ; เขียนภาพตัวหนังสือ "คงเดช กรกาญจนารักษ์" บนจอ โดยที่เราจะยังมองไม่เห็นเพราะใช้
                           ; Palette ดำหมด
spread 3 2 128             ; เปลี่ยน Palette ภาพจะค่อย ๆ ปรากฏออกมา
pload char2.plc 2          ; โหลดภาพตัวหนังสือ บุญล้วน วรรณวิมลศรี"
waitkey 300                ; หน่วงเวลา 3 วินาที
spread 0 3 128             ; ตัวหนังสือบนจอค่อย ๆ หายไป
palette 3                  ; ใช้ Palette ดำหมดทั้ง 16 สี
window 172 194 461 290     ; กำหนดกรอบที่ให้เขียนภาพ
pfade 0 2                  ; เขียนภาพตัวหนังสือ บุญล้วน วรรณวิมลศรี" บนจอโดยที่เราจะยังมองไม่เห็นเพราะใช้
                           ; Palette ดำหมด
spread 3 2 128             ; เปลี่ยน Palette ภาพจะค่อย ๆ ปรากฏออกมา
pload char3.plc 2          ; โหลดภาพตัวหนังสือ "ดร.เอื้อน ปิ่นเงิน"
waitkey 300                ; หน่วงเวลา 3 วินาที
spread 0 3 128             ; ตัวหนังสือบนจอค่อย ๆ หายไป
palette3                   ; ใช้ Palette ดำหมดทั้ง 16 สี
window 168 198 461 289     ; กำหนดกรอบที่ให้เขียนภาพ
pfade 0 2                  ; เขียนภาพตัวหนังสือ "ดร.เอื้อน ปิ่นเงิน" บนจอ โดยที่เราจะยังมองไม่เห็นเพราะใช้ Palette ดำหมด
```

```
spread 3 2 128          ; เปลี่ยน Palette ภาพจะค่อย ๆ ปรากฏออกมา
waitkey 300             ; หน่วงเวลา 3 วินาที
spread 0 3 128          ; ตัวหนังสือบนจอค่อย ๆ หายไป
window                  ; กำหนดกรอบที่ใช้งานให้เต็มจอเหมือนเดิม
pfree 2                 ; เคลียร์ Buffer หมายเลข 2
```

: ------ PRINT WELCOME Version 2 ------

```
video m                 ; เลือกโหมดจอภาพเป็น VGA 16 สี
cload s_text1.pic 1     ; โหลดภาพตัวหนังสือ "มากับเราชิดรับ"
cfade 19 90 300 1 100   ; ให้ตัวหนังสือ"มากับเราชิดรับ" ค่อย ๆ ไหลออกมา
edge on 4               ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload s_text2.pic 1     ; โหลดภาพตัวหนังสือ "แล้วเราจะพาท่านขบคณะของเรา"
cfade 1 90 230 1 200    ; ให้ตัวหนังสือ "แล้วเราจะพาท่านขบคณะของเรา" ค่อย ๆ เขียนขึ้นมาจากซ้ายไปขวา
edge off                ; ยกเลิกไม่ให้มีเส้นสีแดงตามที่เขียน
cload helpmenu.pic 1    ; โหลดรูปคำอธิบายถึงปุ่มที่ใช้กดในการทำงาน
cfade 4 64 30 1 150     ; แสดงรูปคำอธิบายนั้น
cfree 1                 ; เคลียร์ Buffer หมายเลข 1
pload black256.pic 1    ; โหลดรูปที่มี Palette ดำหมดทั้ง 256 สี
pload cohpkmitl.pic 2   ; โหลดรูปคอปเตอร์
dload cohpkmitl.fll 1 1 ; โหลดภาพเคลื่อนไหวของคอปเตอร์ ฉากที่ 1
dload cokmitl.fll 2 1   ; โหลดภาพเคลื่อนไหวของคอปเตอร์ ฉากที่ 2
spread 0 3 128          ; หน้าจอจะค่อย ๆ มืดลงจนดำสนิท
pfree 3                 ; เคลียร์ Buffer หมายเลข 3
```

: ------ SHOW COPPER FLY ------

```
video l                 ; เลือกโหมดของจอภาพเป็น VGA 320x200 256 สี
palette 1               ; ใช้ Palette ดำหมดทั้ง 256 สี
pfade 0 2               ; วาดรูปคอปเตอร์บนจอ โดยที่เรายังมองไม่เห็นเพราะ เราใช้ Palette ดำหมดทั้ง 256 สี
spread 1 2              ; รูปคอปเตอร์จะค่อย ๆ ปรากฏออกมา
pfree 2                 ; เคลียร์ P Buffer หมายเลข 2
putdff 1 8 0 100        ; ทำภาพเคลื่อนไหวจาก Frame แรกถึง Frame 100 โดยเราจะเห็นคอปเตอร์บินออกไป
dfree 1                 ; เคลียร์ D Buffer หมายเลข 1
putdff 2 8 0 170        ; ทำภาพเคลื่อนไหวจาก Frame แรกถึง Frame 170 โดยเราจะเห็นคอปเตอร์บินไปในเมืองแล้ว
                        ; ผ่าน KMITL
dfree 2                 ; เคลียร์ D Buffer หมายเลข 2
spread 0 1              ; จอค่อย ๆ มืดลง
pfree 1 2 3 4           ; เคลียร์ P Buffer
cfree 1 2 3 4           ; เคลียร์ C Buffer
dfree 1 2 3 4           ; เคลียร์ D Buffer
: ---------------

link proj2.txt          ; ไปทำยังโปรแกรม PROJ2.TXT ต่อ
```

: ------ PRINT WELCOME TO KMITL ------

```
video m                 ; เลือกโหมดของจอภาพเป็น VGA 16 สี
pload pblk3.pic 3       ; โหลดภาพที่มี Palette ดำหมดทั้ง 16 สี
palette 3               ; ใช้ Palette ดำหมดทั้ง 16 สี
pload paldeful.pic 2    ; โหลดภาพที่มี Palette ปกติของ VGA 16 สี
cload f_text1.pic 1     ; โหลดภาพตัวหนังสือ "ขอต้อนรับสู่"
cfade 13 220 340 1      ; แสดงภาพตัวหนังสือ "ขอต้อนรับสู่" บนจอ โดยที่เรายังมองไม่เห็นเพราะใช้ Palette ดำหมด
spread 3 2 128          ; เปลี่ยน Palette ภาพตัวหนังสือจะค่อย ๆ ปรากฏ
```

```
cload f_text2.pic 1          ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
waitkey 50                   ; หน่วงเวลา 0.5 วินาที
edge on 4                    ; กำหนดให้มีเส้นสีแดงตามที่เขียนด้วย
cfade 12 180 250 1 100       ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload f_text3.pic 1          ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 1 60 150 1 300         ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง" บนจอ จากซ้ายมาขวา
edge off                     ; ยกเลิกเส้นสีแดง
pfree 2                      ; เคลียร์ P Buffer หมายเลข 2
cfree 1                      ; เคลียร์ C Buffer หมายเลข 1
pload black256.pic 1         ; โหลดภาพที่มี Palette ตำหนดทั้ง 256 สี
pload move.pic 4             ; โหลดภาพของคณะวิศวะ Frame แรก
dload move.fli 3 1           ; โหลดภาพเคลื่อนไหวของคณะวิศวะ
spread 0 3 128               ; ตัวหนังสือบนจอค่อย ๆ มืดลงไป
pfree 3                      ; เคลียร์ P Buffer หมายเลข 3

; ------ SHOW AROUND KMITL -------

video 1                      ; เลือกโหมดของจอภาพเป็น 320x200 256 สี
palette 1                    ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 4                    ; เขียนภาพของคณะวิศวะ Frame แรกลงบนจอภาพ โดยที่เรายังมองไม่เห็นเพราะใช้ Palette ตำหนด
spread 1 4                   ; ภาพของคณะวิศวะ Frame แรกค่อย ๆ ปรากฏ
pfree 4                      ; เคลียร์ P Buffer หมายเลข 4

mark 9999                    ; วนลูป 9999 รอบ
set Y 0                      ; ให้ตัวแปร Y มีค่าเท่ากับ 0
set X 5                      ; ให้ตัวแปร X มีค่าเท่ากับ 5
mark 76                      ; วนลูป 76 รอบ
puldff 3 10 @Y @X            ; แสดงภาพเคลื่อนไหวจาก Frame Y ถึง X
set X @X+6                   ; ให้ X = X+6
set Y @Y+6                   ; ให้ Y = Y+6
ifkey m MENU                 ; ถ้ามีการกดคีย์ M ให้ไปทำต่อที่ Label MENU
loop                         ; จบลูป 76 รอบ
loop                         ; จบลูป 9999 รอบ
MENU:                        ; Label MENU
dfree 3                      ; เคลียร์ D Buffer หมายเลข 3
pfree 1 2 3 4                ; เคลียร์ P Buffer

; ------ MENU -------

MAINMENU1:                   ; Label MAINMENU1
pload black256.pic 4         ; โหลดภาพที่มี Palette ตำหนดทั้ง 256 สี
MAINMENU:                    ; Label MAINMENU
pload menu1.pic 1            ; โหลดภาพ MENU แรกซึ่งมี ตึก6, ตึกโทร, ห้องสมุด, หอประชุม
spread 0 4                   ; ภาพบนจอค่อย ๆ มืดลงไป
STARTMENU:                   ; Label STARTMENU
video 8                      ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                    ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 1                    ; นำภาพ MENU แรกลงจอ โดยที่ยังมองไม่เห็น
spread 4 1                   ; เปลี่ยน Palette ภาพ MENU ค่อย ๆ ปรากฏ
pfree 1                      ; เคลียร์ P Buffer หมายเลข 1
cload press1.pic 3           ; โหลดภาพปุ่มที่ถูกกด

Top:                         ; Label Top
Ifkey a WALK1 b WALK4 c WALK3 d WALK2 n NEXT q END
```

```
                        ; ถ้ามีการกดคีย์ A ให้ไปทำต่อที่ Label WALK1
                        ; ถ้ามีการกดคีย์ B ให้ไปทำต่อที่ Label WALK4
                        ; ถ้ามีการกดคีย์ C ให้ไปทำต่อที่ Label WALK3
                        ; ถ้ามีการกดคีย์ D ให้ไปทำต่อที่ Label WALK2
                        ; ถ้ามีการกดคีย์ N ให้ไปทำต่อที่ Label NEXT
                        ; ถ้ามีการกดคีย์ Q ให้ไปทำต่อที่ Label END
goto Top                ; ถ้ากดคีย์ให้กับไปรอรับคีย์ใหม่

NEXT:                   ; Label NEXT
pload menu2.plc 1       ; โหลดภาพ MENU ที่สองซึ่งมี ตึกคอม, โรงอาหาร, ตึกกิจ, ตึก ท.ส.
spread 0 4              ; ภาพบนจอจะค่อย ๆ มืดลงไป
STARTMENU2:             ; Label STARTMENU2
video s                 ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4               ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 1               ; นำภาพ MENU ที่สองลงบนจอภาพ โดยที่ยังมองไม่เห็น
spread 4 1              ; เปลี่ยน Palette ภาพ MENU จะค่อย ๆ ปรากฏ
pfree 1                 ; เคลียร์ P Buffer หมายเลข 1

Top2:                   ; Label Top2
lfkey e WALK6 f WALK5 g WALK7 h WALK9 n NEXT2 q END
                        ; ถ้ามีการกดคีย์ E ให้ไปทำต่อที่ Label WALK6
                        ; ถ้ามีการกดคีย์ F ให้ไปทำต่อที่ Label WALK5
                        ; ถ้ามีการกดคีย์ G ให้ไปทำต่อที่ Label WALK7
                        ; ถ้ามีการกดคีย์ H ให้ไปทำต่อที่ Label WALK9
                        ; ถ้ามีการกดคีย์ N ให้ไปทำต่อที่ Label NEXT
                        ; ถ้ามีการกดคีย์ Q ให้ไปทำต่อที่ Label END
goto Top2               ; ถ้ากดคีย์ให้กับไปรอรับคีย์ใหม่

NEXT2:                  ; Label NEXT2
pload menu3s.plc 1      ; โหลดภาพ MENU ที่สามซึ่งมี ตึก I, Shop, ตึกเพาเวอร์
cload press1_3.plc 3    ; โหลดรูปปุ่มที่ถูกกด
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลง
STARTMENU3:             ; Label STARTMENU3
video s                 ; เลือกโหมดจอภาพเป็น 640x480 256 สี
palette 4               ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 1               ; นำภาพ MENU ที่สามลงบนจอ โดยที่ยังมองไม่เห็น
spread 4 1              ; ภาพ MENU ที่สามค่อย ๆ ปรากฏ
pfree 1                 ; เคลียร์ P Buffer หมายเลข 1

Top3:                   ; Label Top3
lfkey I WALK8 J WALK10 k WALK11 p MENU_PATAP n F_MAINMENU q F_END
                        ; ถ้ามีการกดคีย์ I ให้ไปทำต่อที่ Label WALK8
                        ; ถ้ามีการกดคีย์ J ให้ไปทำต่อที่ Label WALK10
                        ; ถ้ามีการกดคีย์ K ให้ไปทำต่อที่ Label WALK11
                        ; ถ้ามีการกดคีย์ P ให้ไปทำต่อที่ Label MENU_PATAP
                        ; ถ้ามีการกดคีย์ N ให้ไปทำต่อที่ Label F_MAINMENU
                        ; ถ้ามีการกดคีย์ Q ให้ไปทำต่อที่ Label F_END
goto Top3               ; ถ้ากดคีย์ให้กับไปรอรับคีย์ใหม่

MENU_PATAP:             ; เข้าสู่ MENU ตึกพระเทพฯ
cload press1_3.plc 1    ; โหลดภาพปุ่มที่ถูกกด
cfade 13 411 110 1      ; เขียนภาพปุ่มที่ถูกกดบนตำแหน่งของปุ่ม P
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
color 0                 ; เลือกให้สีดำในการเขียน
```

```
text 418 115 "P"          ; เขียนตัว "P" ลงบนปุ่มที่ถูกกด
spread 0 4                ; ภาพบนจอค่อย ๆ มืดลง
link proj3.txt            ; ให้ไปทำต่อในโปรแกรม PROJ3.TXT

F_MAINMENU:               ; ไปเริ่ม MENU แรกใหม่
cload press1_3.pic 1      ; โหลดภาพปุ่มที่ถูกกด
cfade 13 411 150 1        ; เขียนภาพปุ่มที่ถูกกดบนตำแหน่งของปุ่ม N
color 0                   ; เลือกใช้สีดำในการเขียน
text 418 155 "N"          ; เขียนตัว "N" ลงบนปุ่มที่ถูกกด
cfree 1                   ; เคลียร์ C Buffer หมายเลข 1
goto MAINMENU             ; กลับไปยัง MENU แรก
```

; ------ ตึก 6 ชั้น ------

```
WALK1:                    ; เดินไปยังตึก 6 ชั้น
cfade 13 80 248 3         ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม A
color 0                   ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 89 255 "A"           ; เขียนตัวหนังสือ "A" ลงบนปุ่ม
pload walk1.pic 1         ; โหลดรูปแสดงการเดินไปตึก 6 Frame แรก
dload walk1.fll 1 1       ; โหลดภาพเคลื่อนไหวการเดินไปตึก 6
spread 0 4                ; ภาพบนจอค่อย ๆ มืดลง
video 1                   ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                 ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1                 ; นำรูปแสดงการเดินไปตึก 6 Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
spread 4 1                ; ภาพการเดินไปตึก 6 Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 99          ; ทำภาพเคลื่อนไหวเดินไปยังตึก 6
pfree 1 2                 ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1 2                 ; เคลียร์ D Buffer หมายเลข 1 2
pload walk1_s.pic 2       ; โหลดรูปการหมุนโชว์ตึก 6 Frame แรก
cload ch_eng1.pic 1       ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
waitkey 200               ; หน่วงเวลา 2 วินาที
spread 0 4                ; ภาพบนจอค่อย ๆ มืดลงไป
video 8                   ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                 ; ใช้ Palette ดำหมดทั้ง 256 สี
cfade 13 60 430 1         ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_eng2.pic 1       ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1         ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                   ; เคลียร์ C Buffer หมายเลข 1
color 234                 ; เลือกใช้สีขาวในการวาด
box 0 175 321 376         ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479      ; กำหนดกรอบที่จะให้วาด
pfade 0 2                 ; นำรูปการหมุนโชว์ตึก 6 Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
window                    ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_a1.pic 1         ; โหลดภาพตัวหนังสือ "ตึก 6 ชั้น"
cfade 13 400 315 1        ; เขียนภาพตัวหนังสือ "ตึก 6 ชั้น" แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
color 254                 ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 390 317 540 358       ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึก 6 ชั้น"
spread 4 2                ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 254               ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_a2.pic 1         ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก A"
cfade 1 390 275 1 100     ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก A" บนจอ จากซ้ายไปขวา
waitkey 50                ; หน่วงเวลา 0.5 วินาที
cload ch_a3.pic 1         ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น"
cfade 1 393 245 1 100     ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น" บนจอ จากซ้ายไปขวา
```

```
edge off                    ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dload walk1_s.fll 1 1       ; โหลดภาพเคลื่อนไหวหมุนโชว์ตึก 6
window 1 176 320 479        ; กำหนดกรอบที่ใช้วาด
putdff 1 8 1 999            ; ทำภาพเคลื่อนไหวหมุนโชว์ตึก 6
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
window                      ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_a41.plc 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
cfade 19 0 50 1 150         ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_a42.plc 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 464 174        ; กำหนดกรอบที่จะให้ภาพหายไป
pfade 20 4 20               ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
window 0 0 320 174
pfade 20 4 150
cfade 19 0 0 1 150          ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
cload ch_a435.plc 1 1       ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 3-5 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 464 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 352 174
pfade 20 4 150
cfade 19 64 55 1 150        ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 3-5 มีอะไรบ้าง
cload ch_a46.plc 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 6 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 464 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
cfade 19 0 60 1 150         ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 6 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 464 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1                     ; เคลียร์ P Buffer หมายเลข 1
pload menu1.plc 1           ; โหลดภาพ MENU แรก
spread 0 4                  ; ภาพบนจอจะค่อย ๆ มืดลง
goto STARTMENU              ; กลับไปยัง MENU แรก
```

; ------ หอประชุม ------

```
WALK2:                      ; เดินไปหอประชุม
cfade 13 410 8 3            ; นำภาพปุ่มถูกกดมาทับปุ่ม D
color 0                     ; เลือกสีดำในการวาด
text 419 15 "D"             ; เขียนตัวหนังสือ "D" บนปุ่มที่ถูกกด
pload walk2.plc 1           ; โหลดรูปแสดงการเดินไปหอประชุม Frame แรก
dload walk2.fll 1 1         ; โหลดภาพเคลื่อนไหวการเดินไปหอประชุม
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
video 1                     ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                   ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1                   ; นำรูปแสดงการเดินไปหอประชุม Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
```

```
spread4 1                   ; ภาพการเดินไปหอประชุม Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 99            ; ทำภาพเคลื่อนไหวเดินไปยังหอประชุม
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1 2                   ; เคลียร์ C Buffer หมายเลข 1 2
waitkey 200                 ; หน่วงเวลา 2 วินาที
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลงไป
video s                     ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                   ; ใช้ Palette ตำหนดทั้ง 256 สี
pload walk2_s.pic 2         ; โหลดรูปการหมุนโชว์หอประชุม Frame แรก
cload ch_engd1.pic 1        ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1           ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engd2.pic 1        ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1           ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
color 203                   ; เลือกให้สีขาวในการวาด
box 0 175 321 376           ; วาดกรอบสีขาวรอบ ๆ ที่จะเอารูปมาใส่
window 1 176 320 479        ; กำหนดกรอบที่จะให้วาด
pfade 0 2                   ; นำรูปการหมุนโชว์หอประชุม Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหนด
window                      ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_d1.pic 1           ; โหลดภาพตัวหนังสือ "หอประชุม"
cfade 13 400 315 1          ; เขียนภาพตัวหนังสือ "หอประชุม" แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหนด
color 129                   ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 390 317 540 355         ; วาดกรอบสีแดงรอบตัวหนังสือ "หอประชุม"
spread 4 2                  ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 129                 ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_d2.pic 1           ; โหลดภาพตัวหนังสือ "ชื่อเรียก : หอประชุม"
cfade 1 390 275 1 100       ; เขียนภาพตัวหนังสือ "ชื่อเรียก : หอประชุม" บนจอ จากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_d3.pic 1           ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น
cfade 1 389 245 1 100       ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น บนจอ จากซ้ายไปขวา
cload ch_d4.pic 1           ; โหลดภาพตัวหนังสือ "ระบบเสียง : ..."
cfade 1 495 215 1 100       ; เขียนภาพตัวหนังสือ "ระบบเสียง : ..." บนจอ จากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_d5.pic 1           ; โหลดภาพตัวหนังสือ "ระบบไฟ : ..."
cfade 1 389 185 1 100       ; เขียนภาพตัวหนังสือ "ระบบไฟ : ..."
edge off                    ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dload walk2_s.fll 1 1       ; โหลดภาพเคลื่อนไหวหมุนโชว์หอประชุม
window 1 176 320 479        ; กำหนดกรอบที่ให้วาด
putdff 1 8 1 999            ; ทำภาพเคลื่อนไหวหมุนโชว์หอประชุม
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
window                      ; กำหนดกรอบที่ให้วาดให้เต็มจอ
cload ch_d6.pic 1 1         ; โหลดภาพตัวหนังสือ "เหมาะสำหรับ..."
cfade 19 64 80 1 150        ; เขียนภาพตัวหนังสือ "เหมาะสำหรับ..." โดยไหลลงมา
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 464 174        ; กำหนดกรอบที่จะให้ภาพหายไป
pfade 20 4 20               ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ จางหายไป
window 0 0 504 174
pfade 20 4 100
window
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
pload menu1.pic 1           ; โหลดภาพ MENU แรก
```

```
spread 0 4                  ; ภาพบนจอจะค่อย ๆ มืดลง
goto STARTMENU              ; กลับไปยัง MENU แรก
```

; ------ ห้องสมุด ------

```
WALK3:                      ; เดินไปห้องสมุด
cfade 13 80 8 3             ; นำภาพปุ่มถูกกดมาทับปุ่ม C
color 0                     ; เลือกสีดำในการวาด
text 89 15 "C"              ; เขียนตัวหนังสือ "C" บนปุ่มที่ถูกกด
pload walk3.plc 1           ; โหลดรูปแสดงการเดินไปห้องสมุด Frame แรก
dload walk3.fli 1 1         ; โหลดภาพเคลื่อนไหวการเดินไปห้องสมุด
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
video 1                     ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                   ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1                   ; นำรูปแสดงการเดินไปห้องสมุด Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
spread 4 1                  ; ภาพการเดินไปห้องสมุด Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 99            ; ทำภาพเคลื่อนไหวเดินไปห้องสมุด
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1 2                   ; เคลียร์ D Buffer หมายเลข 1 2
waitkey 200                 ; หน่วงเวลา 2 วินาที
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลงไป
video s                     ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                   ; ใช้ Palette ดำหมดทั้ง 256 สี
pload walk3_s.plc 2         ; โหลดรูปการหมุนโชว์ห้องสมุด Frame แรก
cload ch_engc1.plc 1        ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1           ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engc2.plc 1        ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ถาดกระบัง"
cfade 13 60 390 1           ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ถาดกระบัง"
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
color 234                   ; เลือกให้สีขาวในการวาด
box 0 175 321 376           ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479        ; กำหนดกรอบที่จะให้วาด
pfade 0 2                   ; นำรูปการหมุนโชว์ห้องสมุด Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
window                      ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_c1.plc 1           ; โหลดภาพตัวหนังสือ "ห้องสมุด"
cfade 13 400 315 1          ; เขียนภาพตัวหนังสือ "ห้องสมุด" บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
color 254                   ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 390 317 540 358         ; วาดกรอบสีแดงรอบตัวหนังสือ "ห้องสมุด"
spread 4 2                  ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 254                 ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_c2.plc 1           ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ห้องสมุด"
cfade 1 390 275 1 100       ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ห้องสมุด" บนจอ จากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_c3.plc 1           ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น"
cfade 1 390 245 1 100       ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น" บนจอ จากซ้ายไปขวา
edge off                    ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dload walk3_s.fli 1 1       ; โหลดภาพเคลื่อนไหวหมุนโชว์ห้องสมุด
window 1 176 320 479        ; กำหนดกรอบที่ใช้วาด
putdff 1 1 1 999            ; ทำภาพเคลื่อนไหวหมุนโชว์ห้องสมุด
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
window                      ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_c411.plc 1 1       ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง (ภาคที่ 1)
```

```
cfade 19 0 55 1 150          ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_c412.plc 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 1 มีอะไรบ้าง (ภาคที่ 2)
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 464 174         ; กำหนดกรอบที่จะให้ภาพหายไป
pfade 20 4 20                ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
window 0 0 320 174
pfade 20 4 150
window
cfade 19 0 60 1 150          ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_c421.plc 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง (ภาคที่ 1)
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 464 174         ; กำหนดกรอบที่จะให้ภาพหายไป
pfade 20 4 20                ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
window 0 0 352 174
pfade 20 4 150
window
cfade 19 0 45 1 150          ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_c422.plc 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง (ภาคที่ 2)
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 464 174         ; กำหนดกรอบที่จะให้ภาพหายไป
pfade 20 4 20                ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
window 0 0 320 174
pfade 20 4 150
window
cfade 19 0 55 1 150          ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 464 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                      ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                    ; เคลียร์ P Buffer หมายเลข 1 2
pload menu1.plc 1            ; โหลดภาพ MENU แรก
spread 0 4                   ; ภาพบนจอจะค่อย ๆ มืดลง
goto STARTMENU               ; กลับไปยัง MENU แรก
```

; ------ ตึกโทรคมนาคม ------

```
WALK4:                       ; เดินไปยังตึกโทร
cfade 13 410 248 3           ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม B
color 0                      ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 419 255 "B"             ; เขียนตัวหนังสือ "B" ลงบนปุ่ม
pload walk4.plc 1            ; โหลดรูปแสดงการเดินไปตึกโทร Frame แรก
dload walk4.fil 1 1          ; โหลดภาพเคลื่อนไหวการเดินไปตึกโทร
spread 0 4                   ; ภาพบนจอค่อย ๆ มืดลง
video 1                      ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                    ; ใช้ Palette ตำทั้ง 256 สี
pfade 0 1                    ; นำรูปแสดงการเดินไปตึกโทร Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
spread 4 1                   ; ภาพการเดินไปตึกโทร Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 99             ; ทำภาพเคลื่อนไหวเดินไปยังตึกโทร
pfree 1 2                    ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1 2                    ; เคลียร์ D Buffer หมายเลข 1 2
pload walk4_s.plc 2          ; โหลดรูปการหมุนโกว์ตึกโทร Frame แรก
```

```
waitkey 200              ; หน่วงเวลา 2 วินาที
spread 0 4               ; ภาพบนจอค่อย ๆ มืดลงไป
video s                  ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                ; ใช้ Palette ดำหมดทั้ง 256 สี
cload ch_engb1.pic 1     ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1        ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engb2.pic 1     ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1        ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                  ; เคลียร์ C Buffer หมายเลข 1
color 242                ; เลือกให้สีขาวในการวาด
box 0 175 321 376        ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479     ; กำหนดกรอบที่จะให้วาด
pfade 0 2                ; นำรูปการหมุนโชว์ตึกโทร Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
window                   ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_b1.pic 1        ; โหลดภาพตัวหนังสือ "ตึกโทรคมนาคม"
cfade 13 400 315 1       ; เขียนภาพตัวหนังสือ "ตึกโทรคมนาคม"
color 161                ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 390 317 540 355      ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึกโทรคมนาคม"
spread 4 2               ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 161              ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_b2.pic 1        ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก I"
cfade 1 390 275 1 100    ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก I" บนจอ จากซ้ายไปขวา
waitkey 50               ; หน่วงเวลา 0.5 วินาที
cload ch_b3.pic 1        ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น"
cfade 1 389 245 1 100    ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น" บนจอ จากซ้ายไปขวา
edge off                 ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                  ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                ; เคลียร์ P Buffer หมายเลข 1 2
dload walk4_s.fli 1 1    ; โหลดภาพเคลื่อนไหวหมุนโชว์ตึกโทร
window 1 176 320 479     ; กำหนดกรอบที่ให้วาด
putdff 1 8 1 999         ; ทำภาพเคลื่อนไหวหมุนโชว์ตึกโทร
dfree 1                  ; เคลียร์ D Buffer หมายเลข 1
window                   ; กำหนดกรอบที่ให้วาดให้เต็มจอ
cload ch_b41.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
cfade 19 0 55 1 150      ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_b42.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
waitkey 500              ; หน่วงเวลา 5 วินาที
window 320 0 464 174     ; กำหนดกรอบที่จะให้ภาพหายไป
pfade 20 4 20            ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
window 0 0 320 174
pfade 20 4 150
window
cfade 19 0 67 1 150      ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
cload ch_b43.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 3 มีอะไรบ้าง
waitkey 500              ; หน่วงเวลา 5 วินาที
window 320 0 464 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 352 174
pfade 20 4 150
window
cfade 19 0 45 1 150      ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 3 มีอะไรบ้าง
waitkey 500              ; หน่วงเวลา 5 วินาที
window 320 0 464 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
```

```
window 0 0 504 174
pfade 20 4 100
window
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2               ; เคลียร์ P Buffer หมายเลข 1 2
pload menu1.pic 1       ; โหลดภาพ MENU แรก
spread 0 4              ; ภาพบนจอจะค่อย ๆ มืดลง
goto STARTMENU          ; กลับไปยัง MENU แรก
```

; ------- ตึกคอมพิวเตอร์ -------

```
WALK6:                  ; เดินไปตึกคอมพิวเตอร์
cfade 13 80 248 3       ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม E
color 0                 ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 89 255 "E"         ; เขียนตัวหนังสือ "E" ลงบนปุ่ม
pload walk6.plc 1       ; โหลดรูปแสดงการเดินไปตึกคอมพิวเตอร์ Frame แรก
dload walk6.fli 1 1     ; โหลดภาพเคลื่อนไหวการเดินไปตึกคอมพิวเตอร์
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลง
video 1                 ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4               ; ใช้ Palette ตำทั้ง 256 สี
pfade 0 1               ; นำรูปแสดงการเดินไปตึกคอมพิวเตอร์ Frame แรก ขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้
                        ; Palette ดำหมด
spread 4 1              ; ภาพการเดินไปตึกคอมพิวเตอร์ Frame แรกค่อย ๆ ปรากฏ
putdf 1 15 0 999        ; ทำภาพเคลื่อนไหวเดินไปยังตึกคอมพิวเตอร์
pfree 1 2               ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1 2               ; เคลียร์ D Buffer หมายเลข 1 2
waitkey 200             ; หน่วงเวลา 2 วินาที
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลงไป
video s                 ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4               ; ใช้ Palette ดำหมดทั้ง 256 สี
pload walk6_s.plc 2     ; โหลดรูปการหมุนโชว์ตึกคอมพิวเตอร์ Frame แรก
cload ch_enge1.plc 1    ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1       ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_enge2.plc 1    ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1       ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
color 228               ; เลือกใช้สีขาวในการวาด
box 0 175 321 376       ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479    ; กำหนดกรอบที่จะให้วาด
pfade 0 2               ; นำรูปการหมุนโชว์ตึกคอม Frame แรกวางไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
window                  ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_e1.plc 1       ; โหลดภาพตัวหนังสือ "ตึกคอมพิวเตอร์"
cfade 13 400 315 1      ; เขียนภาพตัวหนังสือ "ตึกคอมพิวเตอร์" บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
color 199               ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 392 312 597 350     ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึกคอมพิวเตอร์"
spread 4 2              ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 199             ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_e2.plc 1       ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก B"
cfade 1 420 275 1 100   ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก B" บนจอ จากซ้ายไปขวา
waitkey 50              ; หน่วงเวลา 0.5 วินาที
cload ch_e3.plc 1       ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น"
cfade 1 421 245 1 100   ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น" บนจอ จากซ้ายไปขวา
edge off                ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
```

```
pfree 1 2                ; เคลียร์ P Buffer หมายเลข 1 2
dload walk6_s.fli 1 1    ; โหลดภาพเคลื่อนไหวหมุนโชว์ตึกคอม
window 1 176 320 479     ; กำหนดกรอบที่ใช้วาด
putdff 1 8 1 999         ; ทำภาพเคลื่อนไหวหมุนโชว์ตึกคอม
dfree 1                  ; เคลียร์ D Buffer หมายเลข 1
window                   ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_e40.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้นใต้ฉุน มีอะไรบ้าง
cfade 19 0 20 1 150      ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_e41.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
waitkey 500              ; หน่วงเวลา 5 วินาที
window 320 0 639 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 0 60 1 150      ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง โดยไหลลงมา
cload ch_e42.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
waitkey 500              ; หน่วงเวลา 5 วินาที
window 320 0 639 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 0 67 1 150      ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง โดยไหลลงมา
cload ch_e43.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 3 มีอะไรบ้าง
waitkey 500              ; หน่วงเวลา 5 วินาที
window 320 0 639 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                  ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                ; เคลียร์ P Buffer หมายเลข 1 2
pload menu2.pic 1        ; โหลดภาพ MENU ที่สอง
spread 0 4               ; ภาพบนจอจะค่อย ๆ มืดลง
goto STARTMENU2          ; กลับไปยัง MENU ที่สอง
```

; ------ โรงอาหาร -------

```
WALK5:                   ; เดินไปโรงอาหาร
cfade 13 410 248 3       ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม F
color 0                  ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 419 255 "F"         ; เขียนตัวหนังสือ "F" ลงบนปุ่ม
pload walk5.pic 1        ; โหลดรูปแสดงการเดินไปโรงอาหาร
dload walk5.fli 1 1      ; โหลดภาพเคลื่อนไหวการเดินไปโรงอาหาร
spread 0 4               ; ภาพบนจอค่อย ๆ มืดลง
video 1                  ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1                ; นำรูปแสดงการเดินไปโรงอาหาร Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
spread 4 1               ; ภาพการเดินไปโรงอาหาร Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 999        ; ทำภาพเคลื่อนไหวเดินไปยังโรงอาหาร
pfree 1 2                ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1                  ; เคลียร์ D Buffer หมายเลข 1
waitkey 200              ; หน่วงเวลา 2 วินาที
```

```
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลงไป
video s                     ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                   ; ใช้ Palette ตำหมดทั้ง 256 สี
pload walk5_s.plc 2         ; โหลดรูปการหมุนโชว์โรงอาหาร Frame แรก
cload ch_engj1.plc 1        ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1           ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engj2.plc 1        ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1           ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
color 255                   ; เลือกใช้สีขาวในการวาด
box 0 175 321 376           ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479        ; กำหนดกรอบที่จะให้วาด
pfade 0 2                   ; นำรูปการหมุนโชว์โรงอาหาร Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหมด
window                      ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_f1.plc 1           ; โหลดภาพตัวหนังสือ "โรงอาหาร"
cfade 13 390 330 1          ; เขียนภาพตัวหนังสือ "โรงอาหาร" แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหมด
color 254                   ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 370 328 543 365         ; วาดกรอบสีแดงรอบตัวหนังสือ "โรงอาหาร"
spread 4 2                  ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 254                 ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_f2.plc 1           ; โหลดภาพตัวหนังสือ "ชื่อเรียก : โรงอาหาร"
cfade 1 385 291 1 100       ; เขียนภาพตัวหนังสือ "ชื่อเรียก : โรงอาหาร" บนจอ จากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_f3.plc 1           ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 1 ชั้น"
cfade 1 385 261 1 100       ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 1 ชั้น" บนจอ จากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_f4.plc 1           ; โหลดภาพตัวหนังสือ "จำนวนร้านอาหาร"
cfade 1 385 232 1 100       ; เขียนภาพตัวหนังสือ "จำนวนร้านอาหาร" บนจอจากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_f5.plc 1           ; โหลดภาพตัวหนังสือ "จำนวนร้านเครื่องดื่ม"
cfade 1 386 204 1 100       ; เขียนภาพตัวหนังสือ "จำนวนร้านเครื่องดื่ม" บนจอจากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_f6.plc 1           ; โหลดภาพตัวหนังสือ "จำนวนเมนูอาหาร"
cfade 1 387 170 1 100       ; เขียนภาพตัวหนังสือ "จำนวนเมนูอาหาร" บนจอจากซ้ายไปขวา
edge off                    ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dload walk5_s.fli 1 1       ; โหลดภาพเคลื่อนไหวหมุนโชว์โรงอาหาร
window 1 176 320 479        ; กำหนดกรอบที่ใช้วาด
putdfl 1 8 1 999            ; ทำภาพเคลื่อนไหวหมุนโชว์โรงอาหาร
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
window                      ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_f7.plc 1 1         ; โหลดภาพตัวหนังสือ "อาหารอร่อย ..."
cfade 19 64 70 1 80         ; เขียนภาพตัวหนังสือ "อาหารอร่อย ..." โดยไหลลงมา
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
pload menu2.plc 1           ; โหลดภาพ MENU ที่สอง
spread 0 4                  ; ภาพบนจอจะค่อย ๆ มืดลง
```

```
goto STARTMENU2          ; กลับไปยัง MENU ที่สอง

; ------- ตึกกิจกรรม -------

WALK7:                   ; เดินไปตึกกิจกรรม
cfade 13 80 8 3          ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม G
color 0                  ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 89 15 "G"           ; เขียนตัวหนังสือ "G" ลงบนปุ่ม
pload walk7.pic 1        ; โหลดรูปแสดงการเดินไปตึกกิจ Frame แรก
dload walk7.fll 1 1      ; โหลดภาพเคลื่อนไหวการเดินไปตึกกิจ
spread 0 4               ; ภาพบนจอค่อย ๆ มืดลง
video 1                  ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1                ; นำรูปแสดงการเดินไปตึกกิจ Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
spread 4 1               ; ภาพการเดินไปตึกกิจ Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 999        ; ทำภาพเคลื่อนไหวเดินไปยังตึกกิจ
pfree 1 2                ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1                  ; เคลียร์ D Buffer หมายเลข 1
waitkey 200              ; หน่วงเวลา 2 วินาที
spread 0 4               ; ภาพบนจอค่อย ๆ มืดลง
video 3                  ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                ; ใช้ Palette ดำหมดทั้ง 256 สี
pload walk7_st.pic 2     ; โหลดรูปการหมุนโชว์ตึกกิจ Frame แรก
cload ch_engg1.pic 1     ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1        ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engg2.pic 1     ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1        ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                  ; เคลียร์ C Buffer หมายเลข 1
color 255                ; เลือกใช้สีขาวในการวาด
box 0 175 321 376        ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปวางใส่
window 1 176 320 479     ; กำหนดกรอบที่จะให้วาด
pfade 0 2                ; นำรูปการหมุนโชว์ตึกกิจ Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
window                   ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_g1.pic 1        ; โหลดภาพตัวหนังสือ "ตึกกิจกรรม"
cfade 13 400 315 1       ; เขียนภาพตัวหนังสือ "ตึกกิจกรรม" บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
color 254                ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 392 312 550 350      ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึกกิจกรรม"
spread 4 2               ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 254              ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_g2.pic 1        ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก S"
cfade 1 405 275 1 100    ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก S" บนจอจากซ้ายไปขวา
waitkey 50               ; หน่วงเวลา 0.5 วินาที
cload ch_g3.pic 1        ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น"
cfade 1 406 245 1 100    ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น" บนจอจากซ้ายไปขวา
edge off                 ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                  ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                ; เคลียร์ P Buffer หมายเลข 1 2
dload walk7_s.fll 1 1    ; โหลดภาพเคลื่อนไหวหมุนโชว์ตึกกิจ
window 1 176 320 479     ; กำหนดกรอบที่ให้วาด
putdff 1 8 8 999         ; ทำภาพเคลื่อนไหวหมุนโชว์ตึกกิจ
dfree 1                  ; เคลียร์ D Buffer หมายเลข 1
window                   ; กำหนดกรอบที่ให้วาดให้เต็มจอ
cload ch_g41.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
cfade 19 64 50 1 150     ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
```

```
cload ch_g42.pic 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 20 1 150        ; เขียนภาพตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
pload menu2.pic 1           ; โหลดภาพ MENU ที่สอง
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
goto STARTMENU2             ; กลับไปยัง MENU ที่สอง
```

; ------ ตึกเทคโนโลยีก่อสร้าง ------

```
WALK9:                      ; เดินไปตึก ท.ส.
cfade 13 410 8 3            ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม H
color 0                     ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 419 15 "H"             ; เขียนตัวหนังสือ "H" ลงบนปุ่ม
pload walk9.pic 1           ; โหลดรูปแสดงการเดินไปตึก ท.ส. Frame แรก
dload walk9.fli 1 1         ; โหลดภาพเคลื่อนไหวการเดินไปตึก ท.ส.
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
video l                     ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                   ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 1                   ; นำรูปแสดงการเดินไปตึก ท.ส. Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหนด
spread 4 1                  ; ภาพการเดินไปตึก ท.ส. Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 999           ; ทำภาพเคลื่อนไหวเดินไปยังตึก ท.ส.
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
waitkey 200                 ; หน่วงเวลา 2 วินาที
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลงไป
video s                     ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                   ; ใช้ Palette ตำหนดทั้ง 256 สี
pload walk9_s.pic 2         ; โหลดรูปการหมุนโชว์ตึก ท.ส. Frame แรก
cload ch_engh1.pic 1        ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1           ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engh2.pic 1        ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1           ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง" บนจอ
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
color 184                   ; เลือกใช้สีขาวในการวาด
box 0 175 321 376           ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479        ; กำหนดกรอบที่จะให้วาด
pfade 0 2                   ; นำรูปการหมุนโชว์ตึก ท.ส. Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหนด
window                      ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_h1.pic 1           ; โหลดภาพตัวหนังสือ "ตึก ท.ส."
cfade 13 345 315 1          ; เขียนภาพตัวหนังสือ "ตึก ท.ส." แต่ยังมองไม่เห็น
color 182                   ; กำหนดสีที่ใช้วาดเป็นสีแดง
```

```
box 330 312 620 350          ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึก ท.ส."
spread 4 2                   ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 182                  ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_h2.pic 1            ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก ท.ส."
cfade 1 390 275 1 100        ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก ท.ส." บนจอจากซ้ายไปขวา
waitkey 50                   ; หน่วงเวลา 0.5 วินาที
cload ch_h3.pic 1            ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 4 ชั้น"
cfade 1 381 245 1 100        ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 4 ชั้น" บนจอจากซ้ายไปขวา
edge off                     ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                      ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                    ; เคลียร์ P Buffer หมายเลข 1 2
dload walk9_s.fil 1 1        ; โหลดภาพเคลื่อนไหวหมุนโชว์ตึก ท.ส.
window 1 176 320 479         ; กำหนดกรอบที่ใช้วาด
putdff 1 8 1 999             ; ทำภาพเคลื่อนไหวหมุนโชว์ตึก ท.ส.
dfree 1                      ; เคลียร์ D Buffer หมายเลข 1
window                       ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_h41.pic 1 1         ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
cfade 19 64 50 1 150         ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_h42.pic 1 1         ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 639 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 65 1 50          ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 2 มีอะไรบ้าง
cload ch_h43.pic 1 1         ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 3 มีอะไรบ้าง
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 639 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 352 174
pfade 20 4 150
window
cfade 19 64 68 1 50          ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 3 มีอะไรบ้าง
cload ch_h44.pic 1 1         ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 4 มีอะไรบ้าง
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 639 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 70 1 50          ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 4 มีอะไรบ้าง
waitkey 500                  ; หน่วงเวลา 5 วินาที
window 320 0 639 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                      ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                    ; เคลียร์ P Buffer หมายเลข 1 2
pload menu2.pic 1            ; โหลดภาพ MENU ที่สอง
spread 0 4                   ; ภาพบนจอค่อย ๆ มืดลงไป
goto STARTMENU2              ; กลับไปยัง MENU ที่สอง
```

```
; ------ ตึกวัดคุม ------

WALK8:                  ; เดินไปตึกวัดคุม
cfade 13 81 249 3       ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม 1
color 0                 ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 90 254 "1"         ; เขียนตัวหนังสือ "1" ลงบนปุ่ม
ploud walk8.plc 1       ; โหลดรูปแสดงการเดินไปตึก 1 Frame แรก
dload walk8.fil 1 1     ; โหลดภาพเคลื่อนไหวการเดินไปตึก 1
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลง
video 1                 ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4               ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 1               ; นำรูปแสดงการเดินไปตึก 1 Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ตำหนด
spread 4 1              ; ภาพการเดินไปตึก 1 Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 200       ; ทำภาพเคลื่อนไหวเดินไปยังตึก 1
pfree 1                 ; เคลียร์ P Buffer หมายเลข 1
dfree 1                 ; เคลียร์ D Buffer หมายเลข 1
waitkey 200             ; หน่วงเวลา 2 วินาที
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลงไป
video s                 ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4               ; ใช้ Palette ตำหนดทั้ง 256 สี
ploud walk8_s.plc 2     ; โหลดรูปการหมุนโชว์ตึก 1 Frame แรก
cload ch_engi1.plc 1    ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1       ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engi2.plc 1    ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1       ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
color 164               ; เลือกใช้สีขาวในการวาด
box 0 175 321 376       ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479    ; กำหนดกรอบที่จะให้วาด
pfade 0 2               ; นำรูปการหมุนโชว์ตึก 1 Frame แรกไว้บนจอ แต่ยังมองไม่เห็น เพราะใช้ Palette ตำหนด
window                  ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_j1.plc 1       ; โหลดภาพตัวหนังสือ "ตึกวัดคุม"
cfade 13 385 315 1      ; เขียนภาพตัวหนังสือ "ตึกวัดคุม" แต่ยังมองไม่เห็น
color 255               ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 365 312 510 360     ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึกวัดคุม"
spread 4 2              ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 255             ; กำหนดให้มีสีแดงตามที่เขียน
cload ch_j2.plc 1       ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก 1"
cfade 1 380 275 1 100   ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก 1" บนจอจากซ้ายไปขวา
waitkey 50              ; หน่วงเวลา 0.5 วินาที
cload ch_j3.plc 1       ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น"
cfade 1 381 245 1 100   ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 3 ชั้น" บนจอจากซ้ายไปขวา
edge off                ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2               ; เคลียร์ P Buffer หมายเลข 1 2
dload walk8_s.fil 1 1   ; โหลดภาพเคลื่อนไหวหมุนโชว์ตึก 1
window 1 176 320 479    ; กำหนดกรอบที่ใช้วาด
putdff 1 8 1 999        ; ทำภาพเคลื่อนไหวหมุนโชว์ตึก 1
dfree 1                 ; เคลียร์ D Buffer หมายเลข 1
window                  ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_j40.plc 1 1    ; โหลดภาพตัวหนังสือที่บอกว่า ชั้นใต้ถุน มีอะไรบ้าง
cfade 19 64 70 1 50     ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_j41.plc 1 1    ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
waitkey 500             ; หน่วงเวลา 5 วินาที
```

```
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 50 1 100        ; เขียนภาพตัวหนังสือที่บอกว่า ขั้น 1 มีอะไรบ้าง
cload ch_j42.pic 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 45 1 100        ; เขียนภาพตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง
cload ch_j431.pic 1 1       ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 3 มีอะไรบ้าง (ภาคที่ 1)
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 352 174
pfade 20 4 150
window
cfade 19 64 60 1 150        ; เขียนภาพตัวหนังสือที่บอกว่า ขั้น 3 มีอะไรบ้าง (ภาคที่ 1)
cload ch_j432.pic 1 1       ; โหลดภาพตัวหนังสือที่บอกว่า ขั้น 3 มีอะไรบ้าง (ภาคที่ 2)
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 464 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 60 1 150        ; เขียนภาพตัวหนังสือที่บอกว่า ขั้น 3 มีอะไรบ้าง (ภาคที่ 2)
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
pload menu3s.pic 1          ; โหลดภาพ MENU ที่สาม
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลงไป
goto STARTMENU3             ; กลับไปยัง MENU ที่สาม
```

; ------ SHOP ------

```
WALK10:                     ; เดินไป Shop
cfade 13 411 249 3          ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม J
color 0                     ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 418 254 "J"            ; เขียนตัวหนังสือ "J" ลงบนปุ่ม
pload walk10.pic 1          ; โหลดรูปแสดงการเดินไป Shop Frame แรก
dload walk10.fli 1 1        ; โหลดภาพเคลื่อนไหวการเดินไป Shop
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
video 1                     ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4                   ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1                   ; นำรูปแสดงการเดินไป Shop Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็นเพราะใช้ Palette ดำหมด
```

```
spread 4 1                  ; ภาพการเดินไป Shop Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 200           ; ทำภาพเคลื่อนไหวเดินไปยัง Shop
pfree 1                     ; เคลียร์ P Buffer หมายเลข 1
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
waitkey 200                 ; หน่วงเวลา 2 วินาที
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
video s                     ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                   ; ใช้ Palette ตำหนดทั้ง 256 สี
pload walk10_s.pic 2        ; โหลดรูปการหมุนโชว์ Shop Frame แรก
cload ch_engj1.pic 1        ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1           ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engj2.pic 1        ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1           ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง" บนจอ
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
color 255                   ; เลือกให้สีขาวในการวาด
box 0 175 321 376           ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479        ; กำหนดกรอบที่จะให้วาด
pfade 0 2                   ; นำรูปการหมุนโชว์ Shop Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็น เพราะใช้ Palette ตำหนด
window                      ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_j1.pic 1           ; โหลดภาพตัวหนังสือ "SHOP"
cfade 13 380 315 1          ; เขียนภาพตัวหนังสือ "SHOP" แต่ยังมองไม่เห็น
color 254                   ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 365 312 580 360         ; วาดกรอบสีแดงรอบตัวหนังสือ "SHOP"
spread 4 2                  ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 254                 ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_j2.pic 1           ; โหลดภาพตัวหนังสือ "ชื่อเรียก : SHOP"
cfade 1 400 265 1 100       ; เขียนภาพตัวหนังสือ "ชื่อเรียก : SHOP" บนจอจากซ้ายไปขวา
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_j3.pic 1           ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 1 ชั้น"
cfade 1 401 235 1 100       ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 1 ชั้น" บนจอจากซ้ายไปขวา
edge off                    ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dload walk10_s.fli 1 1      ; โหลดภาพเคลื่อนไหวโชว์ SHOP
window 1 176 320 479        ; กำหนดกรอบที่ใช้วาด
putdff 1 8 1 999            ; ทำภาพเคลื่อนไหวหมุนโชว์ SHOP
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
window                      ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_j40.pic 1 1        ; โหลดภาพตัวหนังสือที่บอก ชื่อหัวหน้าภาค
cfade 19 64 70 1 100        ; เขียนภาพตัวหนังสือนั้น โดยไหลลงมา
cload ch_j41.pic 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 320 174
pfade 20 4 150
window
cfade 19 64 50 1 100        ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
```

```
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2               ; เคลียร์ P Buffer หมายเลข 1 2
pload menu3s.pic 1      ; โหลดภาพ MENU ที่สาม
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลงไป
goto STARTMENU3         ; กลับไปยัง MENU ที่สาม

: ------ ตึก POWER ------

WALK11:                 ; เดินไปตึกเพาเวอร์
cfade 13 81 9 3         ; นำภาพปุ่มที่ถูกกดมาทับกับตำแหน่งของปุ่ม K
color 0                 ; กำหนดสีที่ใช้วาดเป็นสีดำ
text 88 14 "K"          ; เขียนตัวหนังสือ "K" ลงบนปุ่ม
pload walk11.pic 1      ; โหลดรูปแสดงการเดินไปตึกเพาเวอร์ Frame แรก
dload walk11.fli 1 1    ; โหลดภาพเคลื่อนไหวการเดินไปตึกเพาเวอร์
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลง
video 1                 ; เลือกโหมดจอภาพเป็น 320x200 256 สี
palette 4               ; ใช้ Palette ดำทั้ง 256 สี
pfade 0 1               ; นำรูปแสดงการเดินไปตึกเพาเวอร์ Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็น เพราะใช้ Palette ดำหมด
spread 4 1              ; ภาพการเดินไปตึกเพาเวอร์ Frame แรกค่อย ๆ ปรากฏ
putdff 1 15 0 200       ; ทำภาพเคลื่อนไหวเดินไปยังตึกเพาเวอร์
pfree 1                 ; เคลียร์ P Buffer หมายเลข 1
dfree 1                 ; เคลียร์ D Buffer หมายเลข 1
waitkey 200             ; หน่วงเวลา 2 วินาที
spread 0 4              ; ภาพบนจอค่อย ๆ มืดลงไป
video 3                 ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4               ; ใช้ Palette ดำหมดทั้ง 256 สี
pload walk11_s.pic 2    ; โหลดรูปการหมุนโชว์ตึกเพาเวอร์ Frame แรก
cload ch_engj1.pic 1    ; โหลดภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์"
cfade 13 60 430 1       ; เขียนภาพตัวหนังสือ "คณะวิศวกรรมศาสตร์" บนจอ
cload ch_engj2.pic 1    ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfade 13 60 390 1       ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
cfree 1                 ; เคลียร์ C Buffer หมายเลข 1
color 255               ; เลือกใช้สีขาวในการวาด
box 0 175 321 376       ; วาดกรอบสีขาวรอบ ๆ ที่จะนำรูปมาใส่
window 1 176 320 479    ; กำหนดกรอบที่จะให้วาด
pfade 0 2               ; นำรูปการหมุนโชว์ตึกเพาเวอร์ Frame แรกมาไว้บนจอ แต่ยังมองไม่เห็น เพราะใช้ Palette ดำหมด
window                  ; กำหนดกรอบที่ให้วาดเต็มจอ
cload ch_p1.pic 1       ; โหลดภาพตัวหนังสือ "ตึกเพาเวอร์"
cfade 13 380 330 1      ; เขียนภาพตัวหนังสือ "ตึกเพาเวอร์" แต่ยังมองไม่เห็น
color 254               ; กำหนดสีที่ใช้วาดเป็นสีแดง
box 370 328 553 365     ; วาดกรอบสีแดงรอบตัวหนังสือ "ตึกเพาเวอร์"
spread 4 2              ; ภาพที่วาดทั้งหมดจะค่อย ๆ ปรากฏขึ้นมาบนจอ
edge on 254             ; กำหนดให้มีเส้นสีแดงตามที่เขียน
cload ch_p2.pic 1       ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึกเพาเวอร์"
cfade 1 375291 1 100    ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึกเพาเวอร์" บนจอจากซ้ายไปขวา
waitkey 50              ; หน่วงเวลา 0.5 วินาที
cload ch_p3.pic 1       ; โหลดภาพตัวหนังสือ "แบ่งออกเป็น 3 ตึกย่อย"
cfade 1 375 261 1 100   ; เขียนภาพตัวหนังสือ "แบ่งออกเป็น 3 ตึกย่อย" บนจอจากซ้ายไปขวา
waitkey 50              ; หน่วงเวลา 0.5 วินาที
cload ch_p4.pic 1       ; โหลดภาพตัวหนังสือ "1..."
cfade 1 387 232 1 100   ; เขียนภาพตัวหนังสือ "1..." บนจอจากซ้ายไปขวา
waitkey 50              ; หน่วงเวลา 0.5 วินาที
cload ch_p5.pic 1       ; โหลดภาพตัวหนังสือ "2..."
cfade 1 387 199 1 100   ; เขียนภาพตัวหนังสือ "2..." บนจอจากซ้ายไปขวา
```

```
waitkey 50                  ; หน่วงเวลา 0.5 วินาที
cload ch_p6.pic 1           ; โหลดภาพตัวหนังสือ "3..."
cfade 1 387 170 1 100       ; เขียนภาพตัวหนังสือ "3..." บนจอจากซ้ายไปขวา
edge off                    ; ยกเลิกเส้นสีแดงที่เขียนตาม
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
dload walk11_s.fli 1 1      ; โหลดภาพเคลื่อนไหวหมุนโชว์ดีกเพาเวอร์
window 1 176 320 479        ; กำหนดกรอบที่ใช้วาด
putdff 1 8 1 999            ; ทำภาพเคลื่อนไหวหมุนโชว์ดีกเพาเวอร์
dfree 1                     ; เคลียร์ D Buffer หมายเลข 1
window                      ; กำหนดกรอบที่ใช้วาดให้เต็มจอ
cload ch_p70.pic 1 1        ; โหลดตัวหนังสือ "ดีกปฏิบัติการ..."
cfade 19 32 130 1 60        ; เขียนตัวหนังสือ "ดีกปฏิบัติการ..." ไหลลงมา
cload ch_p71.pic 1 1        ; โหลดตัวหนังสือที่บอกว่า ขั้น 1 มีอะไรบ้าง
waitkey 200                 ; หน่วงเวลา 2 วินาที
cfade 19 96 15 1 100        ; เขียนตัวหนังสือที่บอกว่า ขั้น 1 มีอะไรบ้าง
cload ch_p72.pic 1 1        ; โหลดตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 129        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 352 129
pfade 20 4 150
window
cfade 19 96 25 1 150        ; เขียนตัวหนังสือที่บอกว่า ขั้น 2 มีอะไรบ้าง
waitkey 500                 ; หน่วงเวลา 5 วินาที
window 320 0 639 174        ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1                     ; เคลียร์ C Buffer หมายเลข 1
pfree 1 2                   ; เคลียร์ P Buffer หมายเลข 1 2
pload menu3s.pic 1          ; โหลดภาพ MENU ที่สาม
spread 0 4                  ; ภาพบนจอค่อย ๆ มืดลง
goto STARTMENU3             ; กลับไปยัง MENU ที่สาม

F_END: ;ถ้ากดปุ่ม Q
cload press1_3.pic 1        ; โหลดภาพปุ่มที่ถูกกด
cfade 13 411 70 1           ; เขียนภาพปุ่มที่ถูกกดบนตำแหน่งของปุ่ม Q
color 0                     ; เลือกใช้สีดำในการเขียน
text 418 75 "Q"             ; เขียนตัว "Q" ลงบนปุ่มที่ถูกกด

END:
pfree 1 2 3                 ; เคลียร์ Buffer ที่ใช้มาทั้งหมด
cfree 1 2 3 4
dfree 1 2 3 4
pload manwalk.pic 2         ; โหลดภาพคนเดิน Frame แรก
dload manwalk.fli 1 1       ; โหลดภาพเคลื่อนไหวของคนเดิน
spread 0 4                  ; หน้าจอจะค่อย ๆ มืด
```

```
: ------- Man Walk ------

video 1                    ; เลือกโหมดของจอภาพเป็น 320x200 256 สี
palette 4                  ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 2                  ; นำรูปคนเดิน Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็น
spread 4 2                 ; รูปคนเดิน Frame แรกค่อย ๆ ปรากฏ
pfree 2                    ; เคลียร์ P Buffer หมายเลข 2
putdff 1 8 0 205           ; ทำภาพเคลื่อนไหวของคนเดิน
dfree 1                    ; เคลียร์ D Buffer หมายเลข 1
spread 0 4                 ; หน้าจอค่อย ๆ มืดลง
pfree 4                    ; เคลียร์ P Buffer หมายเลข 4
exit                       ; ออกจากโปรแกรม

: ------- END of MAIN PROGRAM -------

: ------ ศูนย์เรียนรวม ตึกพระเทพฯ ------

pload black256.pic 4       ; โหลดภาพที่มี Palette ตำหนดทั้ง 256 สี
pload menu_bp.pic 2        ; โหลดภาพ MENU ตึกพระเทพฯ
MENU_BP:
video s                    ; เลือกโหมดของจอภาพเป็น 640x480 256 สี
palette 4                  ; ใช้ Palette เป็นสีตำหนดทั้ง 256 สี
pfade 0 2                  ; นำภาพ MENU ตึกพระเทพฯ ขึ้นบนจอแต่ยังมองไม่เห็น
spread 4 2                 ; ภาพ MENU ตึกพระเทพฯ ค่อย ๆ ปรากฏ
pfree 2                    ; เคลียร์ P Buffer หมายเลข 2

TOP_BP:
ifkey b WALK_BPB c WALK_BPC d WALK_BPD e WALK_BPE f WALK_BPF n NEXT4
                           ; ถ้ามีการกดคีย์ B ให้ไปทำต่อที่ Label WALK_BPB
                           ; ถ้ามีการกดคีย์ C ให้ไปทำต่อที่ Label WALK_BPC
                           ; ถ้ามีการกดคีย์ D ให้ไปทำต่อที่ Label WALK_BPD
                           ; ถ้ามีการกดคีย์ E ให้ไปทำต่อที่ Label WALK_BPE
                           ; ถ้ามีการกดคีย์ F ให้ไปทำต่อที่ Label WALK_BPF
                           ; ถ้ามีการกดคีย์ N ให้ไปทำต่อที่ Label NEXT4
goto TOP_BP                ; ถ้ากดคีย์อื่น ให้กลับไปรับคีย์ใหม่

: ------ ศูนย์สนเทศ ------

WALK_BPB:                  ; เดินไปศูนย์สนเทศ
cload press1_p.pic 1       ; โหลดภาพปุ่มที่ถูกกด
cfade 13 300 213 1         ; เขียนภาพปุ่มที่ถูกกดบนตำแหน่งของปุ่ม B
color 0
text 307 218 "B"
cfree 1
pload bp_bctw.pic 2        ; โหลดภาพแสดงการเดินไปศูนย์สนเทศ Frame แรก
dload bp_temp.fli 1 1      ; โหลดภาพเคลื่อนไหวการเดินไปศูนย์สนเทศ
spread 0 4                 ; ภาพบนจอค่อย ๆ มืดลง
video 1                    ; เลือกโหมดของจอภาพเป็น 320x200 256 สี
palette 4                  ; ใช้ Palette ตำหนดทั้ง 256 สี
pfade 0 2                  ; นำภาพแสดงการเดินไปศูนย์สนเทศ Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็น เพราะใช้
                           ; Palette ตำหนดทั้ง 256 สี
spread 4 2                 ; ภาพแสดงการเดินไปศูนย์สนเทศ Frame แรกค่อย ๆ ปรากฏ
pfree 2
putdff 1 8 0 30            ; ทำภาพเคลื่อนไหวเดินไปศูนย์สนเทศ
```

```
putdff 1 8 161 200
dfree 1
pload bp_b.pic 2          ; โหลดรูปการหมุนโชว์ศูนย์สนเทศ Frame แรก
pload ch_bpf1b.pic 1      ; โหลดภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
waitkey 200
spread 0 4
video s
palette 4
window 35 430 639 479
pfade 0 1                 ; เขียนภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
pload ch_bpf2b.pic 1      ; โหลดภาพตัวหนังสือ "พระเทพฯ...(2)"
window 351 429 639 479
pfade 0 1                 ; เขียนภาพตัวหนังสือ "พระเทพฯ...(2)"
pload ch_bpsb.pic 1       ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window 60 390 639 479
pfade 0 1                 ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window
pfree 1
color 161
box 0 175 321 376
window 1 176 320 479
pfade 0 2                 ; เขียนรูปการหมุนโชว์ศูนย์สนเทศ Frame แรก
window
cload ch_bpb1.pic 1       ; โหลดภาพตัวหนังสือ "ศูนย์สนเทศ"
cfade 13 380 330 1        ; เขียนภาพตัวหนังสือ "ศูนย์สนเทศ"
color 127
box 370 328 533 375
spread 4 2
edge on 127
cload ch_bpb2.pic 1       ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก B"
cfade 1 375 291 1 100     ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก B" บนจอจากซ้ายไปขวา
waitkey 50
cload ch_bpb3.pic 1       ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น"
cfade 1 375 261 1 100     ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น" บนจอจากซ้ายไปขวา
waitkey 50
edge off
cfree 1
pfree 1 2
dload bp_b.fli 1 1        ; โหลดภาพเคลื่อนไหวการหมุนโชว์ศูนย์สนเทศ
window 1 176 320 479
putdff 1 8 1 999          ; ทำภาพเคลื่อนไหวการหมุนโชว์ศูนย์สนเทศ
dfree 1
window
cload ch_bpb4.pic 1 1     ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
cfade 19 32 40 1 80       ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง โดยไหลลงมา
waitkey 500
window 320 0 639 174      ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1
pfree 1 2
pload menu_bp.pic 2       ; โหลดภาพเมนูตึกพระเทพฯ
```

```
spread 0 4
goto MENU_BP                ; กลับไปยัง MENU ตึกพระเทพฯ

; ------ อาคารบรรยายรวม -------

WALK_BPC:                   ; เดินไปอาคารบรรยายรวม
cload press1_p.pic 1        ; โหลดภาพปุ่มที่ถูกกด
cfade 13 300 173 1          ; เขียนภาพปุ่มที่ถูกกด
color 0
text 307 178 "C"
cfree 1
pload bp_bcfw.pic 2         ; โหลดภาพแสดงการเดินไปอาคารบรรยายรวม Frame แรก
dload bp_temp.fll 1 1       ; โหลดภาพเคลื่อนไหวการเดินไปอาคารบรรยายรวม
spread 0 4
video 1
palette 4
pfade 0 2                   ; นำภาพแสดงการเดินไปอาคารบรรยายรวม Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็น เพราะใช้
                            ; Palette ดำหมด
spread 4 2
pfree 2
putdff 1 8 0 60             ; ทำภาพเคลื่อนไหวเดินไปอาคารบรรยายรวม
dfree 1
pload bp_cs.pic 2           ; โหลดรูปการหมุนโชว์อาคารบรรยายรวม Frame แรก
pload ch_bpf1c.pic 1        ; โหลดภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
waitkey 200
spread 0 4
video s
palette 4
window 35 430 639 479
pfade 0 1                   ; เขียนภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
pload ch_bpf2c.pic 1        ; โหลดภาพตัวหนังสือ "พระเทพฯ...(2)"
window 351 429 639 479
pfade 0 1                   ; เขียนภาพตัวหนังสือ "พระเทพฯ...(2)"
pload ch_bpsc.pic 1         ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window 60 390 639 479
pfade 0 1                   ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window
pfree 1
color 166
box 0 175 321 376
window 1 176 320 479
pfade 0 2                   ; เขียนรูปการหมุนโชว์อาคารบรรยายรวม
window
cload ch_bpc1.pic 1         ; โหลดภาพตัวหนังสือ "อาคารบรรยายรวม"
cfade 13 375 330 1          ; เขียนภาพตัวหนังสือ "อาคารบรรยายรวม"
color 252
box 370 328 570 365
spread 4 2
edge on 252
cload ch_bpc2.pic 1         ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก C"
cfade 1 390 291 1 100       ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก C" บนจอจากซ้ายไปขวา
waitkey 50
cload ch_bpc3.pic 1         ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น"
cfade 1 390 261 1 100       ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น"
```

```
waitkey 50
edge off
ctree 1
ptree 1 2
dload bp_cs.fil 1 1          ; โหลดภาพเคลื่อนไหวการหมุนโชว์อาคารบรรยายรวม
window 1 176 320 479
putdff 1 8 1 999             ; ทำภาพเคลื่อนไหวการหมุนโชว์อาคารบรรยายรวม
dfree 1
window
cload ch_bpc4.pic 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง.
cfade 19 32 60 1 80          ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 1 มีอะไรบ้าง
waitkey 500
window 320 0 639 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
ctree 1
ptree 1 2
pload menu_bp.pic 2          ; โหลดภาพ MENU ตึกพระเทพฯ
spread 0 4
goto MENU_BP                 ; กลับไปเมนูตึกพระเทพฯ

; ------- อาคารบรรยาย -------

WALK_BPD:                    ; เดินไปอาคารบรรยาย
cload press1_p.pic 1         ; โหลดภาพปุ่มที่ถูกกด
cfade 13 300 133 1           ; เขียนภาพปุ่มที่ถูกกด
color 0
text 307 136 "D"
ctree 1
pload walk_bpd.pic 2         ; โหลดภาพแสดงการเดินไปอาคารบรรยาย Frame แรก
dload walk_bpd.fil 1 1       ; โหลดภาพเคลื่อนไหวการเดินไปอาคารบรรยาย
spread 0 4
video 1
palette 4
pfade 0 2                    ; นำภาพแสดงการเดินไปอาคารบรรยาย Frame ขึ้นจอภาพแต่ยังมองไม่เห็น เพราะใช้ Palette ดำหมด
spread 4 2                   ; ภาพแสดงการเดินไปอาคารบรรยาย Frame ค่อย ๆ ปรากฏ
pfree 2
putdff 1 8 1 999             ; ทำภาพเคลื่อนไหวเดินไปอาคารบรรยาย
dfree 1
pload bp_ds.pic 2            ; โหลดรูปการหมุนโชว์อาคารบรรยาย Frame แรก
pload ch_bpf1d.pic 1         ; โหลดภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
waitkey 200
spread 0 4
video s
palette 4
window 35 430 639 479
pfade 0 1                    ; เขียนภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
pload ch_bpf2d.pic 1         ; โหลดภาพตัวหนังสือ "พระเทพฯ...(2)"
window 351 429 639 479
pfade 0 1                    ; เขียนภาพตัวหนังสือ "พระเทพฯ...(2)"
pload ch_bpsd.pic 1          ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window 60 390 639 479
```

```
pfade 0 1                ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window
pfree 1
color 178
box 0 175 321 376
window 1 176 320 479
pfade 0 2                ; เขียนรูปการหมุนโชว์อาคารบรรยาย Frame แรก
window
cload ch_bpd1.pic 1      ; โหลดภาพตัวหนังสือ "อาคารบรรยาย"
cfade 13 380 330 1       ; เขียนภาพตัวหนังสือ "อาคารบรรยาย"
color 19
box 375 328 560 361
spread 4 2
edge on 20
cload ch_bpd2.pic 1      ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก D"
cfade 1 395 291 1 100    ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก D" บนจอจากซ้ายไปขวา
waitkey 50
cload ch_bpd3.pic 1      ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 5 ชั้น"
cfade 1 397 261 1 100    ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 5 ชั้น" บนจอจากซ้ายไปขวา
waitkey 50
edge off
cfree 1
pfree 1 2
dload bp_ds.fli 1 1      ; โหลดภาพเคลื่อนการหมุนโชว์อาคารบรรยาย
window 1 176 320 479
putdff 1 8 1 999         ; ทำภาพเคลื่อนไหวการหมุนโชว์อาคารบรรยาย
dfree 1
window
cload ch_bpd41.pic 1 1   ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1-2 มีอะไรบ้าง
cfade 19 32 35 1 80      ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 1-2 มีอะไรบ้าง
cload ch_bpd42.pic 1 1   ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 3-5 มีอะไรบ้าง
waitkey 500
window 320 0 639 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfade 19 32 40 1 80      ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 3-5 มีอะไรบ้าง
waitkey 500
window 320 0 639 174     ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1
pfree 1 2
pload menu_bp.pic 2      ; โหลดภาพ MENU ตึกพระเทพฯ
spread 0 4
goto MENU_BP             ; กลับไปเมนูตึกพระเทพ

; ------ อาคารปฏิบัติการ ------

WALK_BPE:                ; เดินไปอาคารปฏิบัติการ
cload press1_p.pic 1     ; โหลดภาพปุ่มที่ถูกกด
```

```
cfade 13 300 93 1          ; เขียนภาพปุ่มที่ถูกกด
color 0
text 307 98 "E"
cfree 1
pload walk_bpe.pic 2       ; โหลดภาพแสดงการเดินไปอาคารปฏิบัติการ Frame แรก
dload walk_bpe.fil 1 1     ; โหลดภาพเคลื่อนไหวการเดินไปอาคารปฏิบัติการ
spread 0 4
video 1
palette 4
pfade 0 2                  ; นำภาพการเดินไปอาคารปฏิบัติการ Frame แรกขึ้นจอ โดยที่ยังมองไม่เห็น
spread 4 2                 ; ภาพค่อย ๆ ปรากฏ
pfree 2
putdff 1 8 1 999           ; ทำภาพเคลื่อนไหวเดินไปอาคารปฏิบัติการ
dfree 1
pload bp_es.pic 2          ; โหลดรูปการหมุนโชว์อาคารปฏิบัติการ Frame แรก
pload ch_bpf1e.pic 1       ; โหลดภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
waitkey 200
spread 0 4
video s
palette 4
window 35 430 639 479
pfade 0 1                  ; เขียนภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
pload ch_bpf2e.pic 1       ; โหลดภาพตัวหนังสือ "พระเทพ...(2)"
window 351 429 639 479
pfade 0 1                  ; เขียนภาพตัวหนังสือ "พระเทพ...(2)"
pload ch_bpse.pic 1        ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window 60 390 639 479
pfade 0 1                  ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window
pfree 1
color 152
box 0 175 321 376
window 1 176 320 479
pfade 0 2                  ; เขียนรูปการหมุนโชว์อาคารปฏิบัติการ Frame แรก
window
cload ch_bpe1.pic 1        ; โหลดภาพตัวหนังสือ "อาคารปฏิบัติการ"
cfade 13 380 330 1         ; เขียนภาพตัวหนังสือ "อาคารปฏิบัติการ"
color 145
box 375 328 585 372
spread 4 2
edge on 145
cload ch_bpe2.pic 1        ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก E"
cfade 1 400 291 1 100      ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก E" บนจอจากซ้ายไปขวา
waitkey 50
cload ch_bpe3.pic 1        ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 5 ชั้น"
cfade 1 400 261 1 100      ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 5 ชั้น" บนจอจากซ้ายไปขวา
waitkey 50
edge off
cfree 1
pfree 1 2
dload bp_es.fil 1 1        ; โหลดภาพเคลื่อนไหวการหมุนโชว์อาคารปฏิบัติการ
window 1 176 320 479
putdff 1 8 1 999 ;ทำภาพเคลื่อนไหวการหมุนโชว์อาคารปฏิบัติการ
dfree 1
```

```
window
cload ch_bpe41.pic 1 1          ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 1-2 มีอะไรบ้าง
cfade 19 64 50 1 80             ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 1-2 มีอะไรบ้าง
cload ch_bpe42.pic 1 1          ; โหลดภาพตัวหนังสือที่บอกว่า ชั้น 3-5 มีอะไรบ้าง
waitkey 500
window 320 0 639 174            ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfade 19 64 40 1 80             ; เขียนภาพตัวหนังสือที่บอกว่า ชั้น 3-5 มีอะไรบ้าง
waitkey 500
window 320 0 639 174            ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1
pfree 1 2
pload menu_bp.pic 2             ; โหลดภาพ MENU ตึกพระเทพฯ
spread 0 4
goto MENU_BP                    ; กลับไปยังเมนูตึกพระเทพฯ

; ------ โรงอาหาร ตึกพระเทพฯ -------

WALK_BPF:                       ; เดินไปโรงอาหาร
cload press1_p.pic 1            ; โหลดภาพปุ่มที่ถูกกด
cfade 13 300 53 1               ; เขียนภาพปุ่มที่ถูกกด
color 0
text 307 58 "F"
cfree 1
pload bp_bcfw.pic 2             ; โหลดภาพการเดินไปโรงอาหาร Frame แรก
dload bp_temp.fil 1 1           ; โหลดภาพเคลื่อนไหวการเดินไปโรงอาหาร
spread 0 4
video 1
palette 4
pfade 0 2                       ; นำภาพการเดินไปโรงอาหาร Frame แรกขึ้นบนจอ แต่ยังมองไม่เห็น
spread 4 2
pfree 2
putdff 1 8 1 160                ; ทำภาพเคลื่อนไหวเดินไปโรงอาหาร
dfree 1
pload bp_f.pic 2                ; โหลดรูปการหมุนโชว์โรงอาหาร Frame แรก
pload ch_bpf1f.pic 1            ; โหลดภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
waitkey 200
spread 0 4
video s
palette 4
window 35 430 639 479
pfade 0 1                       ; เขียนภาพตัวหนังสือ "อาคารเรียนรวม...(1)"
pload ch_bpf2f.pic 1            ; โหลดภาพตัวหนังสือ "พระเทพฯ...(2)"
window 351 429 639 479
pfade 0 1                       ; เขียนภาพตัวหนังสือ "พระเทพฯ...(2)"
pload ch_engf2.pic 1            ; โหลดภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window 60 390 639 479
```

```
pfade 0 1                    ; เขียนภาพตัวหนังสือ "สถาบันเทคโนฯ ลาดกระบัง"
window
pfree 1
color 255
box 0 175 321 376
window 1 176 320 479
pfade 0 2                    ; เขียนรูปการหมุนโชว์โรงอาหาร Frame แรก
window
cload ch_bpf1.pic 1          ; โหลดภาพตัวหนังสือ "โรงอาหาร"
cfade 13 386 330 1           ; เขียนภาพตัวหนังสือ "โรงอาหาร"
color 254
box 375 328 530 367
spread 4 2
edge on 254
cload ch_bpf2.pic 1          ; โหลดภาพตัวหนังสือ "ชื่อเรียก : ตึก F"
cfade 1 390 291 1 100        ; เขียนภาพตัวหนังสือ "ชื่อเรียก : ตึก F" บนจอจากซ้ายไปขวา
waitkey 50
cload ch_bpf3.pic 1          ; โหลดภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น"
cfade 1 390 261 1 100        ; เขียนภาพตัวหนังสือ "จำนวนชั้น : 2 ชั้น" บนจอจากซ้ายไปขวา
waitkey 50
edge off
cfree 1
pfree 1 2
dload bp_f.fil 1 1           ; โหลดภาพเคลื่อนไหวการหมุนโชว์โรงอาหาร
window 1 176 320 479
putdff 1 8 1 999             ; ทำภาพเคลื่อนไหวการหมุนโชว์โรงอาหาร
dfree 1
window
cload ch_bpf4.pic 1 1        ; โหลดภาพตัวหนังสือที่บอกว่า ตึกนี้มีอะไรบ้าง
cfade 19 96 60 1 80          ; เขียนภาพตัวหนังสือที่บอกว่า ตึกนี้มีอะไรบ้าง
waitkey 500
window 320 0 639 174         ; ทำภาพตัวหนังสือเหล่านั้นค่อย ๆ สลายไป
pfade 20 4 20
window 0 0 504 174
pfade 20 4 100
window
cfree 1
pfree 1 2
pload menu_bp.pic 2          ; โหลดภาพ MENU ตึกพระเทพฯ
spread 0 4
goto MENU_BP                 ; กลับไปยังเมนูตึกพระเทพฯ

NEXT4:                       ; ถ้ากดคีย์ N กลับไปสู่ MENU ตึกวิศวะ
cload press1_p.pic 1         ; โหลดภาพปุ่มที่ถูกกด
cfade 13 300 13 1            ; เขียนภาพปุ่มที่ถูกกด
color 0
text 307 18 "N"
cfree 1
spread 0 4
pfree 1 2 3 4
dfree 1
link prof2.txt MAINMENU1     ; กลับไปยังเมนูวิศวะ MENU แรก

; ---------- THE END ----------
```

ภาคผนวก

แนะนำ TOOLS ที่ใช้ในโครงงาน

1 AUTODESK 3D Studio

สำหรับ software ที่ใช้ในการสร้างภาพ graphic แต่ละเฟรมนั้นเราได้เลือกโปรแกรม AUTODESK 3D Studio เพราะสามารถสร้างภาพที่มีความละเอียดสูง มีการลงสีแสงเงาที่สมจริง โปรแกรมนี้ต้องการอุปกรณ์อย่างน้อยดังต่อไปนี้

- คอมพิวเตอร์ที่มี microprocessor เบอร์ 80386 หรือ 80486
- หน่วยความจำอย่างน้อย 4 megabytes
- floppy disk drive 1.2 megabyte หรือ 1.44 megabyte
- Microsoft mouse หรือเทียบเท่า
- VGA compatible display board และ monitor
- hard disk drive เนื้อที่ว่างอย่างน้อย 20 megabytes
- ถ้าเป็น CPU 80386 จะต้องมี math coprocessor Intel หรือ Weitek 3167

อุปกรณ์เพิ่มเติมซึ่งโปรแกรมนี้ support และทำให้ประสิทธิภาพดีขึ้น

- Weitek 4167 Math Coprocessor สำหรับ CPU 80486 จะทำให้ reder เร็วขึ้น 30 percent
- Digitizing Tablet ใช้แทน mouse เป็น pointing device
- Frame Buffer and Monitor สามารถเพิ่ม frame buffer 16, 24 หรือ 32 bits/pixel ซึ่ง compatible กับ Targa 16, 24, 32, Targa+, Visa, Everex Vision 16 frame buffers, หรือ frame buffers อื่นที่ support กับ RDPADI 4.1 interface
- Super VGA Display Card และ Monitor ซึ่งใช้ VESA driver
- Vedeotape Recorder (VTR) Controller สำหรับเก็บ image ที่สร้างขึ้น ซึ่ง compatible กับ VTPADI 4.1/4.2 interface หรือ Lyon Lamb และ Videomedia
- Color Printer ซึ่งทำให้งานมีสีสันขึ้น

AUTODESK ประกอบด้วยโมดูลย่อย 5 โมดูล คือ

2D Shaper
3D Lofter
3D Editor
Keyframer
Material Editor

2D Shaper

เป็นเอดิเตอร์สำหรับการสร้างรูปทรงในระนาบ 2 มิติ ทำให้สามารถสร้างรูปทรงที่มีความซับซ้อนในระนาบ 3 มิติได้ และสามารถสร้างฟอนต์ของตัวอักษรได้

3D Lofter

เมื่อสร้างรูปทรงใน 2D Shaper จะได้รูปในระนาบ 2 มิติ ประกอบด้วย กว้างxยาว ตามแนวแกน X และแนวแกน Y เมื่อโหลดวัตถุมายัง 3D Lofter ก็จะสามารถกำหนดความลึกของวัตถุตามแนวแกนZได้ ซึ่งจะได้รูปทรงในระนาบ 3 มิติ เช่นเดียวกับ 2D shaper 3D Lofterใช้ระบบแกนคาร์ทีเซียน ที่จุด origin ค่าบนทุกแกนจะมีค่าเป็น 0 จากจุด origin ไปทางขวาของแกน X จะมีค่าเป็นบวกเพิ่มขึ้น ตรงกันข้ามทางซ้ายของแกน X ค่าจะเป็นลบมากขึ้น ถ้าเลยจุด origin ไปค่าจะติดลบในการสร้างวัตถุ สามารถเลือกมุมมองโดยการโปรเจค2 มิติ และ 3 มิติ เป็น Front(X/Y), Back(X/Y), Top(X/Z), Bottom(X/Z), Left(Z/Y) และ Right(Z/Y) การแสดงภาพในลักษณะนี้เรียกว่า orthographic view หรืออาจแสดงในลักษณะ 3 มิติ ซึ่งเรียกว่า user view หรือ isometric view ซึ่งการแสดงในมุมมองต่าง ๆนี้จะเหมือนกับใน 3D Editor แต่ใน 3D Editor ยังสามารถเลือกดูจากกล้องตัวใด ๆก็ได้(ที่ได้กำหนดไว้ก่อน) ซึ่งภาพที่จะเห็นเป็นแบบ perspective view การสร้างวัตถุใน3DLofter ไม่เพียงแต่กำหนดความลึกของวัตถุได้เท่านั้น เราอาจสร้างวัตถุที่มีโครงร่างซับซ้อนได้ เช่น ปลายด้านหนึ่งเป็นวงกลมอีกด้านเป็นสี่เหลี่ยม หรือการสร้างรูปทรงแบบแจกัน หลังจากกำหนดรูปทรงเสร็จแล้ว วัตถุจะถูกส่งไปยัง 3D Editor หรือ Keyframer ต่อไป

3D Editor

เป็นเอดิเตอร์ที่มีความสามารถ

1) สร้างวัตถุรูปทรงพื้นฐาน เช่น กล่อง,ทรงกระบอก,ทรงกลม,กรวย,ท่อ,โดนัท ฯลฯ
2) สามารถโหลดวัตถุที่สร้างจาก AUTOCAD ได้(.dxf)
3) สามารถสร้างวัตถุที่ได้จากการเชื่อม, ตัด โดยการใช้ Boolean Operation เช่น union, subtraction และ intersection
4) กำหนดวัสดุ (Matirial) สำหรับพื้นผิวของวัตถุ
5) สร้างและกำหนดลักษณะและตำแหน่งของกล้องและดวงไฟ
6) Render(ลงแสงเงา) ภาพที่ได้สร้างขึ้น

Keyframer

การสร้าง Animation วัตถุ ซึ่ง Animation มีพื้นฐานบนจิตวิทยาการมองถ้าเห็นภาพแสดงต่อเนื่องด้วยความเร็วพอควร สมองจะรับว่าเป็นการเคลื่อนไหวต่อเนื่อง ซึ่งภาพที่เห็นเรียกว่า frame ความเร็วต่ำสุดที่ภาพจะปรากฏเดี่ยว ๆ ประมาณ 10 frames/second ถ้าความเร็วยิ่งสูง การเคลื่อนไหวจะดูนิ่มนวลยิ่งขึ้น

ตัวอย่างอัตราการแสดงภาพในปัจจุบัน

การ์ตูน 12-24 frames/second
ภาพเคลื่อนไหว 24 frames/second
NTSC Television 30 frames/second
PAL Television 25 frames/second

ซึ่ง Keyframer จะสร้างภาพเป็น frame ที่เราสามารถกำหนดความเร็วในการแสดงได้แต่ก่อนงานที่หนักก็คือ การสร้างภาพแต่ละภาพซึ่งเพียง 1 นาที ต้องการ 720-1800 ภาพที่แตกต่างกัน Keyframe จะช่วยได้เพียงแต่

เรากำหนดจุดเริ่มและจุดสิ้นสุดของการเคลื่อนที่ของวัตถุในภาพ Keyframe จะจัดการคำนวณframeเชื่อมต่อที่เรียกว่า tweens เช่น เราเคลื่อนที่วัตถุในเฟรมที่ 10 ซึ่งเคลื่อนที่แตกต่างจากเฟรมที่ 0 ไป 20 unitโปรแกรมจะจัดการให้วัตถุเคลื่อนที่ไปเฟรมละ 2 unit โดยอัตโนมัติ หรือเราสามารถปรับให้เคลื่อนที่จากเร็วไปช้าหรือช้าไปเร็วก็ได้เพราะมีกราฟสำหรับปรับอัตราเร็วของการเคลื่อนที่ได้

Material Editor

ทำให้สามารถสร้างและ edit พื้นผิวของวัสดุ โดยสามารถรวมคุณลักษณะ เช่น สี, ความสว่าง และโปร่งแสง รวมถึง special effect ต่าง ๆ

การปรับแต่งสี

ถ้าเรามองวัตถุรอบ ๆตัวสีที่เราเห็นเรียกว่าสีวัตถุธาตุ(pigmentcolor) เมื่อแสงตกกระทบดอกกุหลาบสีแดง ดอกกุหลาบจะรับเอาสีทั้งหมดของสเปคตรัมยกเว้นสีแดง ซึ่งจะสะท้อนกลับมายังตาของเรา เราจึงเห็นดอกกุหลาบมีสีแดง ถ้าวัตถุใดสะท้อนสเปคตรัมทั้งหมดเราจะเห็นวัตถุเป็นสีขาว แต่ถ้าวัตถุไม่สะท้อนสีใดออกมาเลยเราจะเห็นวัตถุเป็นสีดำ แม่สีวัตถุธาตุมีสีแดง, สีเหลือง และสีน้ำเงิน ถ้าเป็นขั้นที่สอง จะเป็นผลมาจากการผสมแม่ 2 สีเข้าด้วยกัน ถ้าวัตถุเป็นพวกแสงจะเป็นการส่งสีจากตัววัตถุเองแทนที่จะสะท้อนออกมา เมื่อเรามองภาพจากโทรทัศน์หรือ monitor ของเครื่องคอมพิวเตอร์ ที่เราเห็นไม่ใช่สีวัตถุธาตุ แต่เป็นสีแสงอาทิตย์(lightcolor)แม่สีแสงอาทิตย์จะมีสีแดง,สีเขียว และสีน้ำเงิน (ที่เราเรียกว่า RGB system ในระบบคอมพิวเตอร์ทั่วไป) ถ้าเราผสมแม่สี 2 สีเข้าด้วยกันเราก็จะได้สีขั้นที่สองถ้าทุกสีผสมเข้าด้วยกันในความเข้มที่เท่ากันแล้วจะได้สีขาว ถ้าไม่ใส่สีใดเลยจะเป็นสีดำ ยังมีอีกระบบหนึ่งที่ใช้การผสมสีแสงอาทิตย์ คือ HLS (Hue, Luminance, Saturation) คล้ายการผสมสี แต่แทนที่จะผสมความเข้มของแม่สีแต่ละสี ก็เปลี่ยนมาเปลี่ยนระดับของ Hue, Luminance และ Saturation แทนถ้าเราต้องการให้วัตถุของเรามีลวดลาย เราก็สามารถ mapping ภาพที่เป็น bipmap ไปยังพื้นผิวของวัตถุ ไม่ว่าพันผิวของวัตถุจะเป็นระนาบเรียบ ทรงกลม หรือทรงกระบอก โดยภาพที่ map จะต้องมีฟอร์แมตเป็น .CEL .GIF .TGA หรือ .FLC

สรุปขั้นตอนในการสร้างภาพจะต้องประกอบไปด้วย

1.สร้างหรือโหลดวัตถุที่ต้องการ

2.วางตำแหน่งของวัตถุลงบนฉากหรือเคลื่อนย้าย,ปรับสเกล โดยอาจเลือกแสดงภาพในโหมด construction planes ซึ่งจะแสดงวัตถุเป็นกล่อง ๆโดยไม่มีลายละเอียดเพื่อความเร็วในการแสดงภาพ

3.สร้างและกำหนดตำแหน่งของดวงไฟ(จัดแสง) และกล้อง(เลือกมุมกล้อง)

4.กำหนดวัสดุพื้นผิวของวัตถุ และลงแสงเงาบนเฟรม

2 GRASP : Graphics Animation System for Professionals

GRASP เป็นโปรแกรมสร้างภาพ Animation บน PC ซึ่งจะประกอบด้วย 3 ส่วนใหญ่ ๆ

GRASP : เป็น Editor และ interperter ซึ่งจะทำการสร้างโปรแกรม, แก้ไขโปรแกรม, Run โปรแกรม และควบคุมระบบ

PICTOR : เป็น Tool ที่ใช้สำหรับสร้างภาพ หรือ แก้ไขภาพ (คล้าย ๆ Paint Brush ทั่ว ๆ ไป)

ARTOOLS : กลุ่มของโปรแกรมที่ใช้สำหรับสร้าง Animation effect

รูปแบบของโปรแกรม

ในแต่ละบรรทัดของโปรแกรมจะสามารถเป็นได้ 1 ใน 3 อย่างดังนี้

- คำสั่งของโปรแกรม ตามด้วย Parameter
- label ซึ่งจะเป็นชื่อ ตามด้วย colon
- comment ซึ่งจะนำหน้าด้วย semicolon

ขั้นตอน 4 ขั้นตอนที่จำเป็นที่จะแสดงรูปออกหน้าจอ

- เซตโหมดของจอภาพ
- โหลดรูปจาก Disk
- แสดงรูปบนหน้าจอ
- หยุดภาพ

เซตโหมดของจอภาพ

ก่อนที่โปรแกรมจะแสดงภาพบนหน้าจอได้นั้น จะต้องทำการเซตโหมดของจอภาพก่อน โดยใช้คำสั่ง Video ตามด้วยชื่อของโหมดจอภาพ เช่น

```
video a          ;จะเป็นการเซตโหมดของจอภาพให้เป็นโหมด CGA 4 สี
```

โหลดรูปจาก Disk

รูปจะต้องถูกโหลดลงหน่วยความจำก่อนที่จะแสดงรูปนั้นบนจอภาพ เราสามารถจะโหลดรูปหลาย ๆ รูป ในเวลาเดียวกัน ซึ่งการโหลดแต่ละครั้งจะโหลดลงสู่ BUFFER และเราสามารถอ้างถึงรูปต่าง ๆ ได้โดยบอกตัวเลขของ BUFFER การโหลดรูปของโปรแกรม GRASP จะใช้คำสั่ง Pload ตามด้วยชื่อของรูป และเลขของ BUFFER เช่น

```
Pload grasp 1    ;จะโหลดรูปชื่อ grasp ลงสู่ buffer หมายเลข 1
```

แสดงรูปบนหน้าจอ

การแสดงรูปบนหน้าจอจะใช้คำสั่ง Pfade ตามด้วยหมายเลขของลักษณะการปรากฏภาพ และหมายเลขของ BUFFER ที่จะให้ปรากฏบนจอ เช่น

```
pfade 0 1        ;จะนำรูปที่อยู่ใน buffer หมายเลข 1 ออกแสดงบนหน้าจอ
```

การหยุดภาพ

ถ้าคำสั่ง Pfade เป็นคำสั่งสุดท้ายของโปรแกรมโปรแกรมGRASPจะกลับเข้าสู่ editor หลังจากแสดงรูปบนจอภาพแล้วจะทำให้ไม่เห็นรูปที่ปรากฏบนจอภาพดังนั้นหลังจากแสดงรูปบนจอภาพแล้ว จะต้องหยุดภาพนั้นไว้ โดยใช้คำสั่ง Waitkey เช่น

```
waitkey 500      ;จะหยุดรอจนกว่ามีการกดคีย์ หรือรอ 5 วินาทีคำสั่ง Waitkey จะ
```

หยุดรอจนกระทั่งมีการกดคีย์ หรือรอจนกระทั่งถึงเวลาที่กำหนดเมื่อเราคีย์โปรแกรมเสร็จแล้วจะต้องทำการ Save ก่อนทำการ RUN หรือกดคีย์ F10 เครื่องจะทำการSave และ RUN ให้เลย เมื่อทำการ RUN โปรแกรมเสร็จแล้ว GRASP จะกลับเข้าสู่ editor

ข้อความแสดงความผิดพลาด

ถ้าคุณพบข้อความแสดงความผิดพลาด "invalid mode" เมื่อทำการ RUN โปรแกรม ให้เช็คว่าระบบของคุณนั้นสนับสนุนจอภาพ CGA หรือไม่ และถ้าได้ข้อความ "error loading picture" ให้กด Alt-F2 เพื่อที่จะ list รายชื่อของรูปใน directory นั้น ออกมาดูว่ามีรูปที่ต้องการหรือไม่

การวาดและการเก็บภาพ

GRASP จะประกอบไปด้วยคำสั่งวาดรูปลงบนจอภาพ, คำสั่งการสร้างสี, คำสั่งเขียนเส้นตรง, คำสั่งเขียน BOX และคำสั่งวาดสี่เหลี่ยมผืนผ้าที่มีการระบายสีภายใน เราจะขอกล่าวถึงคำสั่งBox และคำสั่ง Line ก่อน ซึ่งจะกล่าวถึงเรื่องการเปลี่ยนสี และการเก็บรูปบนหน้าจอลง disk

การกำหนดสี

คำสั่ง Color จะใช้ในการเปลี่ยนสีของการวาดปัจจุบัน ซึ่งสีนี้จะสามารถใช้วาด box, เขียนตัวหนังสือ และอื่น ๆ เช่น

```
color 1                 ;เลือกสีจาก palette ช่องที่ 1 สำหรับการวาดรูป
```

ในโหมด A default palette จะมีช่อง 4 ช่อง คือสีดำ, สีเขียวน้ำเงิน, สีแดง และสีเทา ซึ่งคำสั่งนี้จะเซตสีของการวาดเป็นสีเขียวน้ำเงิน

การลบหน้าจอ

คำสั่ง Clearscr จะระบายหน้าจอทั้งหน้าจอ ให้เป็นสีที่ใช้วาดปัจจุบัน เช่น ถ้าต้องการระบายสีหน้าจอให้เป็นสีแดงในโหมด A (CGA 4 สี) จะใช้คำสั่ง

```
color 2                 ;เซตสีที่ใช้วาดปัจจุบันให้เป็นสีแดง
clearscr                ;ระบายสีหน้าจอ
```

การวาด

คำสั่ง Box จะวาดรูปสี่เหลี่ยม ซึ่งต้องการ parameter 4 ตัว โดยที่เป็นตำแหน่ง x,y ของมุมตรงข้าม 2 มุม เช่น

```
box 30 10 100 120       ;วาด box
```

คำสั่ง Line จะวาดเส้นตรง ซึ่งต้องการ parameter 4 ตัวเช่นกัน เช่น

```
ine 10 20 90 100        ;วาดเส้น
```

เส้นตรงนี้จะถูกวาดจากตำแหน่ง 10,20 ไปยัง 90,100

การเก็บรูปหน้าจอ

คำสั่งที่ต้องใช้ในการเก็บรูปหน้าจอลง disk นั้นจำเป็นต้องใช้คำสั่ง Pgetbuf และ Psave โดยที่ Pgetbuf จะสร้าง buffer และ copy หน้าจอลงบน buffer นั้น โดยที่จะมีหมายเลขของ buffer คล้าย ๆ กับคำสั่ง Pload เช่น

```
pgetbuf 2               ;จะ copy รูปภาพบนหน้าจอลงบน buffer หมายเลข 2
```

ต่อไปงานของเราคือการเก็บรูปภภาพจาก buffer ลงบน disk โดยใช้คำสั่ง Psave ซึ่งจะต้องใส่ชื่อไฟล์ลงไปด้วย เช่น

psave OURPIC.PIC 2 ;จะ copy รูปจาก buffer หมายเลข 2 ลงบน disk

ในการสร้างไฟล์ OURPIC.PIC บน disk นั้น คุณสามารถจะทำการ display รูปใหม่นี้ลงได้ด้วยคำสั่ง ShowpicหรือโหลดรูปบนPictor หรือคุณจะสามารถโหลดมันในโปรแกรม GRASP อื่น ๆ ได้

Clipping

ชิ้นส่วนของรูป(clipping) จะคล้าย ๆ กับรูป (picture) ต่างกันตรงที่ clipping ไม่มีรายละเอียดเกี่ยวกับ palette โดยทั่วไปแล้ว clipping จะเล็กกว่า picture และมีได้หลาย ๆ ขนาด เหมือน ๆ กับ picture, clipping จำเป็นจะต้องถูกโหลดลงสู่ buffer ก่อนที่จะถูก display ไม่เหมือนกับ picture ตรงที่มันสามารถ display ได้มากกว่า 1 ทาง

การโหลด clipping clipping จะถูกโหลดลงสู่หน่วยความจำด้วยคำสั่งCload เหมือนกับ picture, ที่ clipping หลาย ๆ ชิ้นสามารถถูกโหลดเข้าสู่หน่วยความจำได้ในเวลาเดียวกัน โดยถูกกำหนดด้วยหมายเลข buffer ซึ่งจะใช้คำสั่ง Cload ตามด้วยชื่อ clipping และหมายเลข buffer เช่น

cload man1 ;โหลด clipping ลงสู่ buffer หมายเลข 1

หมายเหตุ ถ้าเราไม่ใส่ชื่อจุดหลังชื่อไฟล์ GRASP จะสมมติให้เป็นจุด CLP แต่ถ้าเป็นจุดที่นอกเหนือกว่านี้จะต้องใส่จุดให้ด้วย

Display Clipping ใช้คำสั่ง Cfade สำหรับ display clipping โดยที่จะต้องตามด้วยหมายเลข fade, ตำแหน่งของจอภาพที่จะให้ clipping ปรากฏ , หมายเลข buffer และ speed ของ fade เช่น

cfade 9 90 27 1 200 ;display the clipping

คำสั่ง Cfade ไม่สามารถใช้ fade หมายเลข 0 ได้ แต่ถ้าเป็นรูปเล็ก ๆ และไม่มี delay time ใช้ fade อื่น ๆ ก็จะไม่เห็นความแตกต่าง คำสั่ง Cfade จะไม่สนับสนุนเรื่อง Transparency และมันจะใช้ได้เพียงบน byte boundary

Transparency ในตัวอย่าง clipping ที่มี background เป็นสีดำ ซึ่ง GRASP จะสามารถให้มันโปร่งใส่ได้ โดยใช้คำสั่ง Tran ซึ่งมันจะยอมให้คุณเลือกสีที่จะไม่ให้ display ได้ 1 สี เช่น

tran on 0 ;จะทำให้ไม่ display สีดำ

background ใน clipping ที่เป็นสีดำ จะไม่ถูก display และคำสั่งนี้จะสำเร็จลงได้เราจำเป็นต้องใช้คำสั่ง Putup แทนคำสั่ง Cfade ซึ่ง Cfade จะไม่สนับสนุนเรื่อง Transparency เช่น

putup 90 27 1 ;จะ display clipping ใน buffer หมายเลข 1

Byte boundary ข้อจำกัดสุดท้ายของคำสั่ง Cfade มันจะไม่สามารถ display clipping ทุกๆ ตำแหน่งในแนวแกน x ได้ แต่จะต้องอยู่บน byte boundary ซึ่ง byte boundary จะอ้างถึง video memory ที่ถูก organize และการติดต่อ โดยที่จะขึ้นอยู่กับ video mode ตัวอย่างเช่นการแทนใน mode A, byte boundary จะตกอยู่ที่ทุก ๆ 4 จุด จะทำให้ Cfade สามารถ display รูปที่ตำแหน่งในแกนxเป็น 0 , 4 , 8 ตามลำดับ ถ้าคุณกำหนดให้ display ที่ตำแหน่ง 5 มันจะทำการปัดลงมาให้เหลือ 4

Load-time Tran การจะใช้คำสั่ง Tran จะต้องใช้ก่อนการโหลด clipping

Animation

ในตอนแรกเราจะกล่าวถึง concept พื้นฐานที่จำเป็นสำหรับ animation เช่น video mode , การโหลด และการ display ซึ่งในส่วนนี้เราจะพิจารณา 3 เทคนิคดังนี้

- auto animation โดยใช้คำสั่ง Float และคำสั่ง Fly
- manual animation โดยใช้คำสั่ง Putup
- canned animation โดยใช้คำสั่ง Putdff

คำสั่ง Float และคำสั่ง Fly

คำสั่ง Float และ Fly จะเป็น 2 คำสั่ง สำหรับทำ automatic animation คุณจะมีรายละเอียดของตำแหน่งเริ่มต้น และตำแหน่งสุดท้าย, ระยะทางของการเคลื่อนที่แต่ละครั้ง, แฟกเตอร์ของ delay, และหมายเลข buffer ของ clipping ที่ใช้ ซึ่งอาจจะมีหลายตัวก็ได้โดยใช้ commas คั่นกลาง เช่น

```
fly 0,0 320,100 2 10 1,2,3,4,5
```

คำสั่ง Float จะทำงานคล้าย ๆ กับคำสั่ง Fly ต่างกันตรงที่คำสั่ง Float จะรักษา background เดิมไว้ โดยที่การเลื่อน clipping แต่ละครั้งจะลบ clipping ครั้งก่อนออกไปด้วย แต่คำสั่ง Fly จะไม่ลบแต่เป็นการเขียนทับ

คำสั่ง Putdff

งานที่เป็น differential animation ที่ถูกสร้างขึ้นด้วย Gdff ซึ่งได้ใช้ Gdff ทำการอัดไฟล์ให้เล็กลงได้ DFF buffer ก็เหมือนกับ picture หรือ clipping buffer ซึ่งมันก็คือพื้นที่ในหน่วยความจำที่ถูกจองโดย GRASP

คำสั่ง Putup

เราจะใช้คำสั่ง Putup เพื่อทำการ dispiay clipping ออกหน้าจอ และจะใช้คำสั่ง Putup ซ้ำ ๆ กัน เพื่อให้ภาพเลื่อนตำแหน่งไปมาได้ ซึ่งเราจะอธิบายเทคนิคนี้ในส่วนต่อ ๆ ไป เมื่อเราพิจารณาถึงการใช้หน้าจอ2 page แล้ว จากกฎนี้จะทำให้คุณสามารถควบคุมวัตถุหลาย ๆ ชิ้นให้เคลื่อนที่ได้ในเวลาเดียวกัน และเป็นอิสระต่อกัน

Looping และ Offset

Looping

คำสั่ง Mark และ Loop ถูกออกแบบมาให้กระทำกลุ่มของคำสั่งใด ๆ มากกว่า 1 ครั้งคำสั่ง Mark จะกำหนดจุดเริ่มต้นของกลุ่มคำสั่งที่จะให้กระทำซ้ำ ตามด้วยหมายเลขของจำนวนครั้งที่จะให้กระทำ คำสั่ง Loop จะกำหนดจุดสิ้นสุดของกลุ่มคำสั่ง สำหรับตัวอย่าง เราจะให้รูปคนเดินข้ามหน้าจอหลาย ๆ รอบ ซึ่งหมายถึงว่าจะต้องกระทำคำสั่ง Putdff หลาย ๆ ครั้ง โดยเราจะใช้คำสั่ง Mark และ Loop ไว้ข้างบนและล่างตามลำดับ

```
mark 7          ;เซตให้กระทำซ้ำ 7 รอบ
putdff          ;Run a dff animation
loop            ;repeat the loop
```

กลุ่มคำสั่งนี้จะถูกเรียกว่า ลูป (loop)

Coordinate

คำสั่งการเขียนสิ่งต่าง ๆ ลงบนจอภาพจะต้องกำหนดโคออร์ดิเนท ที่จะให้เขียนลงไปด้วย เช่น คำสั่ง Box , Line , Cfade และ Putup

Offset

คำสั่ง Offset นี้จะยอมให้คุณเซตระยะห่างในการเคลื่อนที่ทั้งแนวแกน x และแนวแกน y และจะมีผลกับคำสั่งการวาดรูปบนจอภาพสำหรับตัวอย่าง , ถ้าคุณมีโปรแกรมวาดรูป box และต้องการวาดรูปboxนั้นให้เลื่อนไปทางขวา 4 จุด และเลื่อนขึ้นไป 9 จุด คุณสามารถทำให้โดยเพิ่มคำสั่งนี้ลงไปที่จุดเริ่มของโปรแกรม

offset 4 9 ;สร้างการเคลื่อนที่ของแกน x,y

คำสั่ง Offset จะสามารถใช้ในรูปแบบของความสัมพันธ์ได้โดยเพิ่มตัว R ลงไปหลังคำสั่ง Offset นั้น ๆ เช่น

offset 4 9 r

ในลักษณะนี้เมื่อมีการวนลูปจะทำให้เกิดการเปลี่ยนตำแหน่งการวาดใหม่ซึ่งจะสัมพันธ์กับตำแหน่งเดิมดังนี้คือเลื่อนไปทางขวา 4 จุดและเลื่อนขึ้นไป 9 จุด

<u>window และ position</u>

window

คุณสามารถจะกำหนดส่วนของหน้าจอที่จะให้เกิดการเปลี่ยนแปลงได้โดยการกำหนด window ของหน้าจอ เมื่อคำสั่งอื่น ๆ เขียนบนหน้าจอ จะทำให้พื้นที่เพียงใน window ที่กำหนดเท่านั้นที่เกิดการเปลี่ยนแปลงคำสั่งนี้ต้องการ2โคออร์ดิเนทคู่ที่เป็นมุมตรงข้ามของพื้นที่ active เช่น

window 0 0 40 40 ;กำหนด window บนหน้าจอที่จะให้ active

คำสั่ง Pfade ถ้าตามด้วยคำสั่งนี้จะdisplay เพียงมุมซ้ายล่างของรูป จากตำแหน่ง 0,0 ถึงตำแหน่ง 40, 40 คำสั่ง window นั้นมี Relative option เหมือนกับคำสั่ง Offset โดยจะทำการเพิ่มโคออร์ดิเนทเทียบกับโคออร์ดิเนทของ window อันล่าสุดและจะสร้าง window โดยใช้โคออร์ดิเนทใหม่นั้น เช่น

window 4 4 4 4 r

จากตัวอย่างข้างบนจะสร้าง window ใหม่ที่มีมุมทั้งหมดเลื่อนไปทางขวาบน 4 จุด

<u>หมายเหตุ</u> window จะสามารถสร้างได้เพียงตำแหน่งที่อยู่บน byte boundary เท่านั้น

position

เมื่อคุณสร้างwindowขึ้นมาคำสั่ง Pfade จะเปลี่ยนมุมซ้ายล่างของรูปให้อยู่บนมุมซ้ายล่างของwindow โดยอัตโนมัติ ซึ่งหมายความว่า GRASP จะไม่สนใจว่า window จะอยู่ส่วนไหนของจอภาพ , คำสั่ง Pfade จะ display มุมซ้ายล่างของรูปที่จุดนั้นคำสั่ง position จะสามารถทำให้เปลี่ยนตำแหน่งของรูปไปตามตำแหน่งของ window ได้ไม่จำเป็นต้องเริ่มจากมุมซ้ายล่างของรูปเสมอไป โดยที่คำสั่ง position จะมีรายละเอียดของหมายเลข buffer ของรูปที่จะมีผล และโคออร์ดิเนทของจุดในรูปที่จะให้ตรงกับมุมซ้ายล่างของ window เช่น

position 1 0 0

คำสั่งนี้ จะทำให้ตำแหน่งที่ 0,0 ของรูปตรงกับตำแหน่ง 0,0 ใน window ปัจจุบันเหมือนกับคำสั่ง Offset และ window , คำสั่ง Position จะมี Relative switch

<u>Data</u>

ในคำสั่งทั้งหมดที่ผ่านมานั้นเราใส่รายละเอียดของคำสั่งในบรรทัดของคำสั่งนั้นๆ เลย ในงานที่ยากๆนั้นจะต้องสามารถเปลี่ยนรายละเอียดของคำสั่งระหว่างการ execute ได้ โดยเฉพาะในลูป การแก้ปัญหานี้ใน GRASP คือ

การใช้ data structure data structure ในโปรแกรม GRASP คือหนึ่งหรือมากกว่าหนึ่งบรรทัดของบรรทัดที่ใช้ในการกำหนดค่าหรือ character string ที่จะถูกใช้ในคำสั่งอื่น ๆ ในโปรแกรม

data structure ของโปรแกรม GRASP นี้จะมีทั้งหมด 3 ชนิดดังนี้

- data line
- inline data block
- labeled data block

ซึ่งคุณจะใช้ data structure แบบใดนั้นขึ้นอยู่กับจำนวนของข้อมูล

Data line

Dataline เป็น data structure ที่ง่ายที่สุด ซึ่งประกอบด้วยคำสั่ง Data ตามด้วยลำดับของค่าหรือ character string ซึ่งค่าทั้งหมดจะต้องอยู่บนบรรทัดเดียว ค่าบนบรรทัดข้อมูลจะถูกเรียกว่า"Data element" โดยที่แต่ละ element สามารถถูกแยกได้โดยช่องว่าง หรือ commas

โดยที่สัญลักษณ์@ ในคำสั่ง Color จะเป็นตัวแปรของข้อมูลที่ได้จากบรรทัดของข้อมูล ในแต่ละครั้งที่โปรแกรมGRASP ทำการอ่านคำสั่งนี้ มันจะแทนสัญลักษณ์ @ ด้วยค่าจากบรรทัดข้อมูล(dataline) จากตัวอย่างสีจะเปลี่ยนไปจากสี 1,2,3,... ตามลำดับ บรรทัดของข้อมูล(data line) จะต้องถูกกระทำก่อนที่จะพบตัวแปรข้อมูลเมื่อโปรแกรม GRASP กระทำคำสั่ง Data มันจะทำการเก็บ list ของ element นี้และเริ่มใช้มันแทนตัวแปรของข้อมูล โดยที่จะมี pointer เป็นตัวชี้ data element ตามลำดับ

Data skip

คำสั่ง Dataskip จะยอมให้คุณเปลี่ยนตำแหน่งของ pointer การใช้คำสั่งนี้ค่าที่กำหนดจะแสดงให้เห็นว่าจะข้าม data element ไปทีละกี่ element

Inline data block

ชนิดที่ 2 ของ data structure จะไม่มีข้อจำกัดว่าจะต้องมีข้อมูลเพียงบรรทัดเดียวคำสั่ง Databegin และ Dataend จะเป็นคำสั่งเริ่มต้นก่อนที่จะเป็นกลุ่มของ data element(จะมีกี่บรรทัดก็ได้) และคำสั่งปิดท้ายของกลุ่ม data element ตามลำดับ ส่วนตัวแปรของข้อมูลจะเหมือนกับ data line

Labeled data block

ชนิดที่ 3 ของ data structure แตกต่างไปจาก 2 ชนิดแรก โดยที่ data elementจะอยู่ที่จุดสุดท้ายของโปรแกรม หลังจากคำสั่ง Exit มันจะแตกต่างจาก Inline data block ซึ่งมันจะกำหนดเป็นชื่อของ label แทนกลุ่มของ data element ในคำสั่ง Databegin

Multimedia

GRASP ยอมให้คุณใช้อุปกรณ์ภายนอกและภายใน(เครื่องที่ติดตั้งภายในหรือภายนอกคอมพิวเตอร์และถูกควบคุมโดยคอมพิวเตอร์) ซึ่งในที่นี้ยังไม่รวม PRINTER หรือ PLOTTER แต่จะเป็น slide projectors , cd players

และระบบ lighting effect คำสั่งที่ใช้ควบคุมอุปกรณ์ต่าง ๆ ประกอบด้วย Int , Out และ Send โปรแกรม GRASP จะมีคำสั่งสำหรับผลิตเสียง โดยออกทางลำโพงของคอมพิวเตอร์

Out

คำสั่ง Out จะยอมให้คุณส่งค่าไปยัง port โดยที่ port คือทางของคอมพิวเตอร์ที่ใช้สำหรับติดต่อกับวงจรหน่วยความจำอื่น ๆ มันจะถูกกำหนดโดยแอดเดรส ทั้ง Disk drive, คีย์บอร์ด และจอภาพ จะถูกติดต่อโดย port ของคอมพิวเตอร์

Send

คำสั่ง Send จะส่ง character string ไปยัง DOS device เช่น parallel หรือ serial port ตัวอย่างนี้จะพิมพ์ข้อความบนเครื่องพิมพ์ที่ต้องอยู่กับ parallel port ตัวที่หนึ่ง

```
send lpt1 "This is a test"
```

คำสั่ง Send สามารถถูกใช้กับอุปกรณ์อื่น ๆ ได้เช่น modems ฯลฯ

Int

อุปกรณ์บางตัวถูกควบคุมด้วยโปรแกรม resident ซึ่งจะถูกโหลดก่อนตอนเริ่มต้น GRASPจะสามารถส่งคำสั่งถึง driver เหล่านี้ด้วยคำสั่ง Int

Note

คำสั่ง Note จะเล่นโน๊ตเพลง มันจะมีรายละเอียดของระดับเสียง, เสียงสูงเสียงต่ำ และระยะเวลาที่จะให้เสียงนั้น ๆ เกิด ซึ่งมีหน่วยเป็นเศษหนึ่งส่วนร้อยวินาที

```
note 286 25 30
```

ตัวอย่างนี้จะเล่นโน๊ตเป็นเวลา 30/100 วินาที โดยที่เลข 286 จะหมายถึงโน๊ต C

Noise

คำสั่ง Noise จะสร้างเสียง effect โดยที่จะใช้ระดับเสียง 2 ค่า และระยะเวลาที่จะให้เสียงนั้น ๆ เกิด เช่น

```
noise 400 300 50
```

text and timing

มากกว่าการวาด box หรือ เส้นตรง และ display รูปภาพ , GRASP สามารถพิมพ์ตัวหนังสือบนหน้าจอได้ คำสั่งที่ใช้ในการพิมพ์ตัวอักษร คือ Text เช่น

```
text 10 10 "Hello"        ;พิมพ์ "Hello" ที่ตำแหน่ง 10,10
```

Fonts

โปรแกรม GRASP จะมี font ของตัวอักษรต่าง ๆ สำหรับใช้ในงานกราฟิก ซึ่งจะเป็นไฟล์ที่มีนามสกุล SET หรือ FNT แต่ละไฟล์ของ font จะบรรจุเซตของตัวอักษรในสไตล์ที่แตกต่างกัน เหมือนกับรูปและ clipping

โดยที่คุณจะต้องโหลด font ก่อนที่คุณจะใช้มัน ด้วยคำสั่ง Fload

```
fload russell 1
```

คำสั่งนี้จะโหลด font Russell เข้าสู่ font buffer หมายเลขหนึ่ง เหมือนกับรูปและ clipping ที่คุณสามารถโหลด font หลาย ๆ font เข้าไปในเวลาเดียวกันได้ font เหล่านี้จะสามารถโหลดในPictorได้ และถ้านานสกุลของไฟล์ font ไม่ใช่ SET คุณจำเป็นจะต้องเพิ่มนานสกุลเข้าไปให้กับมันด้วย

หมายเหตุ font เหล่านี้จะไม่ display ใน text mode

ถ้าคุณโหลดหลาย font จะต้องใช้คำสั่ง Font สำหรับการเลือก font หนึ่ง font ที่จะใช้ในการ display โดยที่จะต้องตามด้วยหมายเลข buffer ของ font ที่คุณต้องการจะใช้ เช่น

```
font 3                    ;เริ่มต้นใช้ font หมายเลข 3
```

ในคำสั่งนี้ไม่จำเป็นถ้าคุณโหลดfontเพียง fontเดียว แต่ถ้าโหลดหลาย font และไม่มีการเลือก font GRASP จะเลือกใช้ font ที่โหลดมาเป็นตัวสุดท้าย

window text

เราได้กล่าวถึงคำสั่งWindowไปแล้วว่าสามารถจำกัดพื้นที่ของหน้าจอที่จะให้มีการ display ได้ ซึ่งจะมีผลต่อคำสั่ง Text เช่นกัน

Delay

เราจะขอกล่าวถึง factor delay ของคำสั่ง Pfade, Cfade, Fly, Waitkey และคำสั่งอื่น ๆ เวลาจะมีหน่วยเป็นเศษหนึ่งส่วนร้อยวินาที คำสั่ง Putup จะยอมให้มี factor ของ delay ซึ่งจะทำให้ GRASP หยุดภาพหลังจาก Putup ไปแล้ว เช่น

```
cload man1 1
putup 10 10 1 50
putup 20 20 1 50
putup 30 30 1 50
putup 40 40 1 50
```

ตัวอย่างจะหยุดภาพเป็นเวลาครึ่งวินาทีมันจะเหมือนกับการใช้คำสั่ง Putup แล้วตามด้วย Waitkey 50 คำสั่ง Text จะยอมให้มี factor ของ delay ซึ่งมันจะหยุดหลังจากเขียนตัวอักษรแต่ละตัวบนจอภาพ เช่น

```
text 10 40 "Lazy Sunday Afternoon" 20
```

ตัวแปรและนิพจน์

เหมือนกับโปรแกรมอื่นๆ โปรแกรม GRASP จะมีตัวแปรเป็นได้ทั้ง string, ตัวแปรทางคณิตศาสตร์และฟังก์ชันการเปรียบเทียบของตัวแปร ในส่วนนี้จะขออธิบายการใช้ตัวแปรในโปรแกรม GRASP

ตัวแปร คุณสามารถสร้างตัวแปรและกำหนดค่าโดยใช้คำสั่ง Set เช่น

```
set myname "Toby"    ;กำหนดค่าตัวแปรชื่อ myname
```

ตัวอย่างนี้จะสร้างตัวแปรชื่อ"myname"และกำหนดค่าให้เป็น string "Toby" ถ้าตัวแปร myname ถูกกำหนดค่าไว้ก่อนแล้ว คำสั่งนี้จะทำการเปลี่ยนค่ามันเมื่อตัวแปรถูกกำหนดค่าแล้วคุณสามารถใช้ค่าของตัวแปรนั้นๆ

ได้ โดยที่จะต้องขึ้นต้นด้วยตัว @ แล้วตามด้วยชื่อของตัวแปร เช่น

```
set myname "Toby"   ;define the variable myname
text 0 0 @myname    ;print the variable
waitkey             ;wait for a keystroke
```

ในตัวอย่างนี้จะพิมพ์คำว่า 'Toby' บนหน้าจอ

หมายเหตุ คำสั่ง Set ใช้กำหนดค่าลให้กับตัวแปร เวลาที่ต้องการใช้ค่าของตัวแปรนั้น จะต้องใช้สัญญลักษณ์ @ นำหน้าแล้วถึงตามด้วยชื่อตัวแปรนั้น ๆ

นิพจน์ ตัวแปรสามารถถูกใช้โดยCommandlineอื่นๆ มันจะถูกเก็บในรูป character string แต่โปรแกรม GRASP จะแปลงมันเป็นค่าตัวเลขเมื่อจำเป็น และจะมีเครื่องหมายทางคณิตศาสตร์ เช่น บวก , ลบ , คูณ , และ หาร

```
set x 7             ;define the variable x
set y 6             ;define the variable y
set z @x+@y         ;define the variable z
text 0 0 @z         ;print z's value
waitkey             ;wait for a keystroke
```

คำสั่ง Text จะทำการ display ผลลัพธ์ของการบวก ซึ่งในที่นี่คือ 13 ที่จริงแล้วเราไม่จำเป็นต้องใช้คำสั่ง Set บรรทัดที่ 3 เราสามารถใช้นิพจน์การบวก โดยตรงกับคำสั่ง Text ได้เลยเช่น

```
set x 7             ;define the variable x
set y 6             ;define the variable y
text 0 0 @x+@y      ;print their total
waitkey             ;wait for a keystroke
```

โปรแกรม GRASP จะกระทำการบวก และจะทำการพิมพ์ผลลัพธ์ที่ได้นั้นออกมาเลย

หมายเหตุ character string "Toby" จะถูกแปลงให้มีค่าเป็น 0 ถ้ามันถูกนำมาใช้กับนิพจน์ทางคณิตศาสตร์

Two - page animation

ถ้าจอภาพของคุณเป็นจอ EGA คุณสามารถจะใช้เทคนิคต่าง ๆ ต่อไปนี้ได้ เช่น Two-page animation, panning, และ split-screen display

การกระพริบของภาพ

การทำ Animation กับรูปภาพใหญ่ ๆ นั้นจะทำให้เกิดการกระพริบของภาพ ทั้งนี้เพราะการเปลี่ยนภาพใน video memory และก่อนที่จะทำการวาดภาพในตำแหน่งใหม่ จะต้องทำการลบภาพในตำแหน่งเก่าออกก่อนถ้าภาพนั้นมีขนาดใหญ่จะทำให้เกิดการกระพริบของภาพขึ้น Two-page animation จะเป็นวิธีแก้ปัญหาการกระพริบของภาพโดยทำการสลับ buffer ก่อนที่มันจะถูกก๊อกปี้ไปบนหน้าจอ

Video page

EGA display card จะมี video memory เพียงพอที่จะบรรจุหน้าจอ 2 หน้าจออย่างสมบูรณ์ ซึ่งหน้าจอทั้ง 2 นั้นจะมีรายละเอียดของภาพเท่ากับ 640x350 จุด 16 สี เราจะอ้างถึงภาพใน page 2 page นี้ได้ด้วยหลายเลข 0 และ 1 โดยปกติแล้ว GRASP จะ display บน page 0 และจะวาดบน page นี้ด้วยเทคนิค two-page animation นี้จะทำให้เครื่องแยกว่า page หนึ่งจะทำการวาด และอีก page หนึ่งจะทำการแสดงออกหน้าจอ ซึ่งก่อนที่จะมีการแสดงรูปออกทางหน้าจอจะต้องทำการวาดรูปนั้น ๆ ไว้ก่อนแล้วเพื่อเป็นการลดอาการกระพริบของภาพ

หมายเหตุ คุณจะต้องใช้โหมด EGA (E,F,หรือ G) สำหรับการใช้งานในเทคนิคนี้

setpage

คำสั่ง Setpage จะใช้สำหรับเซตว่าจะให้ page ไหนเป็น page สำหรับแสดงออกหน้าจอ และ page ไหนเป็น page สำหรับใช้ในการวาด เช่น

```
setpage 0 1
```

ในตัวอย่างนี้จะกำหนดให้ page 0 เป็น page ที่ใช้สำหรับ display ออกทางหน้าจอ แต่ page 1 จะใช้สำหรับการวาดต่าง ๆ ซึ่งการวาดนั้น ๆ จะไม่ปรากฏบนหน้าจอ จนกว่าจะมีการสลับ (swap) page โดยใช้คำสั่ง

```
setpage 1 0
```

คำสั่งนี้จะ display page 1 , ซึ่ง page 0 จะกลายเป็น page ที่ใช้สำหรับการวาดเทคนิค two-page animation จะยอมให้คุณวาง clipping หลาย ๆ รูปบนหน้าจอก่อนที่จะทำการ display มัน คำสั่ง Float และ Fly จะมีผลกับ two-page animation เช่นกัน โดยที่มันจะทำการ swappage แบบอัตโนมัติ เราเพียงแต่สั่ง setpage 0 1 มันจะทำการ swap page ไปมาให้เอง

Revpage

คำสั่ง Revpage จะทำการswap page 2 page มันจะมีความหมายเหมือนกับคำสั่ง Setpage 0 1 หรือ Setpage 1 0 ขึ้นอยู่กับการกำหนดปัจจุบัน

หมายเหตุ เมื่อต้องการจะใช้ page เดียวเหมือนเดิมก็ให้ใช้คำสั่ง setpage 0 0 จะมีความหมายว่า page การวาด และ page การแสดงเป็น page เดียวกัน

บทวิจารณ์และสรุปผล

ในการโครงงานนี้ได้เลือกการทำ Animation แบบ Frame Animation ซึ่งเป็นการเคลื่อนไหวแบบเต็มจอ (Full Screen Animation) จะต้องสร้างภาพที่ต้องการให้เคลื่อนไหวเก็บไว้ก่อนเป็นภาพ ๆ ทีละเฟรมแล้วนำมาแสดงทีละภาพแบบการฉายภาพยนตร์ ซึ่งวิธีนี้จะได้งานที่มีคุณภาพดีแต่ต้องใช้เวลาในการทำนาน และหน่วยความจำสูงมากในการเก็บมูล และงานที่ใช้เป็นโหมด 256 สีซึ่งเป็นโหมดมาตรฐานของ VGA ไม่จำเป็นต้องใช้การ์ดพิเศษในการควบคุม แต่ทำให้สีมีน้อยเกินไปทำให้ภาพไม่สมจริงเท่าที่ควรเพราะต้องมีการ share palette กัน แต่ถ้ามีการเลือกให้แสดงสีได้มากกว่านี้หน่วยความจำในการเก็บก็จะมากไปด้วย

กิตติกรรมประกาศ

การทำโครงงานนี้ได้สำเร็จไปได้ด้วยดีเพราะ การควบคุมดูแลของ อาจารย์ เอื้อน ปิ่นเงิน ซึ่งท่านเป็นอาจารย์ประจำภาควิชาคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง โดยท่านได้เสียสละเวลาให้คำแนะนำและคำปรึกษาตลอดในการทำโครงงานนี้

และต้องขอขอบคุณเจ้าหน้าที่ของกองอาคารสถานที่ของสถาบันฯ ที่ให้ความกรุณาให้ยืมแบบแปลนของตัวอาคาร และขอขอบคุณหัวหน้าบรรณารักษ์ห้องสมุดของคณะฯ ที่กรุณาให้ข้อมูลข่าวสารประกอบโครงงานนี้

หนังสืออ้างอิง

Autodesk 3D Studio Release 2 Reference Manual, Audesk, Inc Publication, February 26, 1992.

Blinn, J.F., and M.E Newell, *Computer Graphics,* August 1987

Brodlie, K.W., *Mathematical Methode in Computer Graphics and Design,* Academic Press, London(edited), 1980

ChanS. Park, *Interactive Microcomputer Graphics,* Addison Wesley, September 1985

Don Pearson, *Image Processing,* McGraw-Hill International Editions 1991

Edward Angel, *Computer Graphics,* Addison Wesley, June 1990

Graphics Animation System for Professionals version 4.0 Reference Manual, Paul Mace GRAPHIC SOFTWARE

Harrington,S., *Computer graphics : A Programming Approach,* McGraw-Hill, New York, 1993