การจำลองรูปทรงแฟรคทอลโดยทฤษฎีระบบสมการที่มีการทำซ้ำ

SIMULATION OF FRACTAL IMAGES BY IFS

กัลยา โรจนวลี

ณัฐธร ทองระอา

ปรียาภรณ์ บุญพยนต์

ปัญหาพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2547

SIMULATION OF FRACTAL IMAGES BY IFS

KANLAYA ROJANAVALEE

CHUTTORN TONGRA-AR

PREEYAPORN BOONPAYONT

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2004

ปัญหาพิเศษเรื่อง	การจำลองรูปทรงแฟรคทอลโดยทฤษฎีระบบสมการที่มีการทำซ้ำ
	SIMULATION OF FRACTAL IMAGES BY IFS
ชื่อนักศึกษา	นางสาวกัลยา โรจนวลี 44050399
	นางสาวณัฐธร ทองระอา 44050404
	นางสาวปรียาภรณ์ บุญพยนต์ 44050431
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์
สาขาวิชา	วิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษา	ดร.กรกช ประชุมรักษ์

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ปีการศึกษา 2547

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ.ดร.นันทิกา เบญจเทพานันท์	
กรรมการ	อ.สายชล ใจเย็น	
กรรมการและอาจารย์ที่ปรึกษา	ดร.กรกช ประชุมรักษ์	

(รองศาสตราจารย์ ดร. วีระ บุญจริง)

หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ปัญหาพิเศษเรื่อง	การจำลองรูปทรงแฟรคทอลโดยทฤษฎีระบบสมการที่มีการทำซ้ำ		
ชื่อนักศึกษา	นางสาวกัลยา	โรจนวลี	44050399
	นางสาวณัฐธร	ทองระอา	44050404
	นางสาวปรียาภรณ์	บุญพยนต์	44050431
ปริญญา	วิทยาศาสตรบัณฑิต		
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์		
สาขาวิชา	วิทยาการคอมพิวเตอร์		
ปีการศึกษา	2547		
อาจารย์ที่ปรึกษา	ดร.กรกช ประชุมรักษ์		

บทคัดย่อ

ปัญหาพิเศษฉบับนี้ได้นำเสนอโปรแกรมที่ใช้จำลองรูปทรงทางเรขาคณิตที่มีลักษณะแปลกตาน่าสนใจหรือที่เรียกกันโดยทั่วไปว่า แฟรคทอล ซึ่ง แฟรคทอล นั้นคือรูปทรงที่มีความคล้ายคลึงกันในทุกๆรายละเอียดไม่ว่าจะมองในระยะใกล้หรือไกล โดยโครงการของเรานั้นได้นำทฤษฎีระบบสมการที่มีการทำซ้ำ (Iterated Function Systems) หรือ self-similarity มาใช้ในการสร้างภาพแฟรคทอล ซึ่งทฤษฎีระบบสมการที่มีการทำซ้ำ เป็นการใช้ระบบสมการเดิมซ้ำแล้วซ้ำอีกจนกระทั่งได้ผลลัพธ์ออกมาเป็นภาพแฟรคทอลที่ต้องการ โปรแกรมของเราจะสามารถให้ ผู้ใช้ สร้างภาพแฟรคทอล จากภาพเริ่มต้นที่เป็นรูปอะไรก็ได้โดยใช้ทฤษฎี ระบบสมการที่มีการทำซ้ำ โดยที่ผู้ใช้นั้นไม่จำเป็นต้องมีความรู้ทางด้านคณิตศาสตร์หรือคอมพิวเตอร์กราฟิกใดๆ เลย เพียงแค่ผู้ใช้จัดวางรูปแบบตามที่ต้องการ โดยรูปเริ่มต้นที่ใช้นั้นจะเป็นรูปอะไรก็ได้ และใส่จำนวนในการทำซ้ำของภาพ จากนั้นโปรแกรมจะแสดงภาพ แฟรคทอล ที่เสร็จสมบูรณ์ออกมา ซึ่งจะช่วยให้บุคคลทั่วไปที่มีความสนใจทางด้าน แฟรคทอล ผู้ที่กำลังศึกษาหรือผู้ที่เริ่มที่จะศึกษา สามารถทำความเข้าใจได้ง่ายขึ้นโดยไม่จำเป็นต้องมีความรู้ทางด้านทฤษฎีนี้มาก่อน และเป็นอีกวิธีหนึ่งที่จะทำให้บุคคลทั่วไปหันมาสนใจทางด้านคณิตศาสตร์มากขึ้น อีกทั้งโปรแกรมนี้ยังให้อิสระกับผู้ใช้ในการสร้างสรรค์รูปแบบลวดลายใหม่ๆของ แฟรคทอล ได้ด้วยตนเอง

Special Project Title	SIMULATION OF FRACTAL IMAGES BY IFS
Students	Ms. Kanlaya Rojanavalee 44050399
	Ms. Chuttorn Tongra-ar 44050404
	Ms. Preeyaporn Boonpayont 44050431
Degree	Bachelor of science
Department	Mathematics and Computer Sciences, Faculty of Science
Programme	Computer Science
Academic Year	2004
Special Project Advisor	Dr.Korakot Prachumrak

ABSTRACT

This special project develops the program that creates the mathematical shapes called fractals. Fractals are geometrical shapes that are very complex and infinitely detailed. When zoomed in on a section, it will have the same details as the whole fractal. They are recursively defined and the small sections are similar to the large ones. This project applies one of Fractal theory called IFS (Iterated Function System) or self-similarity to create fractals. This project can help anyone with or without computer graphics knowledge who is interested in studying fractals to understand them easily and clearly. The user can just draw the beginning image then map the iterated formula and enter number of iterated times then the program will show the complete fractal image. This program reduces the complication of the Theory. Therefore it can help anyone who is interested in fractal to study the theory easier and it can make the theory more interesting and easy for everyone.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่อง การจำลองรูปทรงแฟรคทอลโดยทฤษฎีระบบสมการที่มีการทำซ้ำสามารถสำเร็จลุล่วงไปด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์กรกช ประชุมรักษ์ อาจารย์ที่ปรึกษาปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษาในการแก้ไขปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณบิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนการทำปัญหาพิเศษนี้สำเร็จลุล่วงไปได้ด้วยดี รวมทั้งเพื่อนๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ เกี่ยวกับปัญหาพิเศษ สุดท้ายนี้ขอขอบพระคุณคณาจารย์ในภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ ที่ให้ความรู้และคอยแนะนำสิ่งดีๆ ทั้งเรื่องในตำราและนอกตำราไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2548

สารบัญ

สารบัญ (ต่อ)

หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหาพิเศษ

ในธรรมชาติรอบตัวเรานั้น หากเราลองพิจารณาดีๆแล้ว จะเห็นได้ว่าทุกๆส่วนในธรรมชาติมีรายละเอียดในตัวของมันเอง ไม่ว่าจะมองลึกลงไปแค่ไหนหรือใกล้เพียงไรก็ตาม ยกตัวอย่างเช่น ในใบเฟิร์น ถ้าเรานำกิ่งย่อยๆของมันมาขยายดู เราจะเห็นว่ากิ่งย่อยๆของมันนั้นเมื่อขยายออกมาแล้วก็จะมีลักษณะเหมือนกับใบใหญ่ที่เราเห็นได้โดยทั่วไป และในสาขาวิชาคอมพิวเตอร์กราฟิกได้มีทฤษฎีที่อธิบายลักษณะทางธรรมชาติเช่นนี้ไว้ ซึ่งก็คือแฟรคทอลโดยเป็นการนำรูปแบบสมการทางคณิตศาสตร์มาจำลองรูปทรงทางธรรมชาติโดยมีวิธีการสร้างได้หลากหลายวิธี

ในปัญหาพิเศษฉบับนี้จะเน้นการสร้างแฟรคทอลโดยใช้ทฤษฎีระบบสมการที่มีการทำซ้ำหรือ ไอเอฟเอส(Iterated Function Systems) เป็นหลัก ซึ่งโดยหลักของทฤษฎีไอเอฟเอสนั้นจะเน้นที่การทำซ้ำโดยสามารถจัดรูปแบบเริ่มต้นของการทำซ้ำได้ ตามทฤษฎีไอเอฟเอสไม่ว่าจะเริ่มต้นที่ภาพใดแต่หากมีการทำซ้ำในรูปแบบเดียวกันภาพสุดท้ายที่ได้จะมีลักษณะเหมือนกัน แต่ถ้าภาพเริ่มต้นเหมือนกันแต่มีรูปแบบการทำซ้ำที่ต่างกันก็จะได้ภาพที่มีลักษณะต่างกันอย่างสิ้นเชิง ซึ่งปัญหาพิเศษฉบับนี้จะนำทฤษฎีไอเอฟเอสและสมการทางคณิตศาสตร์เหล่านั้นมาประมวลผลและแสดงให้เห็นรูปทรงแฟรคทอลเพื่อให้เข้าใจมากยิ่งขึ้น

1.2 วัตถุประสงค์ของปัญหาพิเศษ

1.เพื่อให้เกิดความรู้ความเข้าใจในทฤษฎีการสร้างแฟรคทอลมากยิ่งขึ้น และสามารถนำมาใช้ในการจำลองรูปทรงต่างๆ ที่มีลักษณะของแฟรคทอลได้โดยใช้โปรแกรม Visual C#.NET ในการสร้างโปรแกรมสำเร็จรูป

2.สร้างภาพแฟรคทอล 2 มิติ โดยใช้ทฤษฎีระบบสมการที่มีการทำซ้ำหรือไอเอฟเอส (Iterated Function Systems)

3.เพื่อเป็นพื้นฐานของการพัฒนางานทางด้านคอมพิวเตอร์กราฟิกต่อไป

1.3 ขอบเขตของปัญหาพิเศษ

1.จำลองรูปทรงที่มีลักษณะของแฟรคทอลในแบบ 2 มิติ จากความรู้ทางด้านคอมพิวเตอร์กราฟิก โดยใช้ทฤษฎีแฟรคทอลไอเอฟเอสเป็นพื้นฐาน

2.ให้อิสระกับผู้ใช้ในการสร้างสรรค์ภาพแฟรคทอล 2 มิติ ในรูปแบบของตนเอง

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.เพื่อให้เกิดความรู้ความเข้าใจในด้านคอมพิวเตอร์กราฟิกมากยิ่งขึ้น

2.เป็นสื่อในการเรียนรู้สำหรับผู้ที่สนใจศึกษาทางด้านแฟรคทอลและคอมพิวเตอร์กราฟิกต่อไป

3.เป็นอีกทางเลือกหนึ่งที่จะแนะนำแฟรคทอลให้แก่ผู้ที่ยังไม่มีความรู้ เพื่อที่จะทำให้เกิดความสนใจและนำไปศึกษาต่อไป

1.5 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

เครื่องไมโครคอมพิวเตอร์โดยมีขนาดเครื่องดังนี้

-CPU ความเร็ว 600MHz ขึ้นไป

-RAM อย่างน้อย 256 MB

-HARDDISK 20 GB เป็นอย่างน้อย

-DRIVE CD-ROM ความเร็ว 12X ขึ้นไป

-ระบบปฏิบัติการ WINDOWS NT4.0,WINDOWS 2000 หรือ WINDOWS XP PROFESSIONAL

1.6 ขั้นตอนการดำเนินการ

1.ศึกษาเนื้อหาและทฤษฎีของแฟรคทอลโดยเน้นแนวทางศึกษาไปในเรื่องทฤษฎีไอเอฟเอส

2.ออกแบบโปรแกรมต้นแบบ กำหนดและแบ่งส่วนต่างๆของระบบที่ต้องดำเนินการสร้าง

3.ดำเนินการสร้างและพัฒนาโปรแกรมที่ได้ออกแบบ

4.ทำการทดสอบโปรแกรมที่ได้สร้างไว้และศึกษาถึงปัญหาที่เกิดขึ้นในการสร้างและดำเนินการแก้ไข

5.สรุปประสิทธิภาพและปัญหาที่เกิดขึ้น

6.จัดทำเอกสารประกอบโปรแกรม

1.7 แผนการทำงานของระบบ

No.	Task Name	June	July	August	September	October	November	December	January	February	March
1	ศึกษาปัญหาและที่มาของปัญหาพิเศษ										
2	ศึกษาเครื่องมือ software ขอบเขตและความเป็นไปได้										
3	จัดทำแบบขออนุมัติ ทำปัญหาพิเศษ										
4	ศึกษา software ที่ใช้										
5	ศึกษาทฤษฎีและหลักการที่เกี่ยวข้อง										
6	ออกแบบ interface, algorithm										
7	เขียนโปรแกรมโดยใช้ Visual C#.NET										
8	ทดสอบและแก้ไขโปรแกรม										
9	สรุปโครงงานปัญหาพิเศษ										
10	จัดทำเอกสารประกอบโครงงานปัญหาพิเศษ										

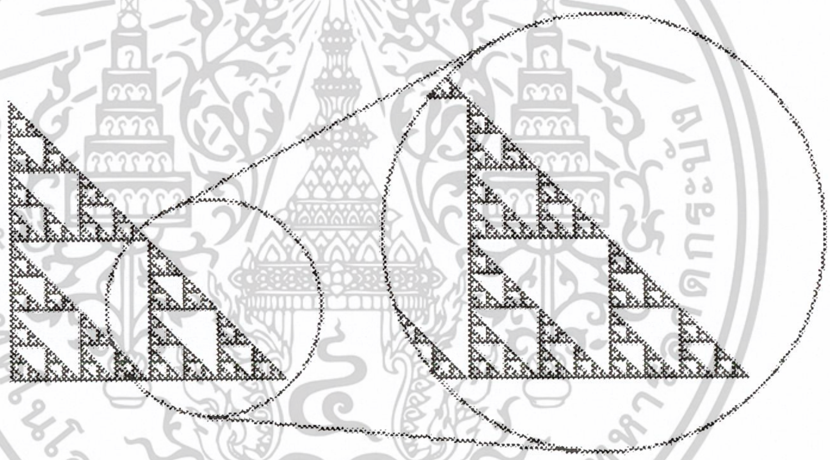
บทที่ 2
ความรู้พื้นฐานและทฤษฎีที่เกี่ยวข้อง

2.1 ทฤษฎีแฟรคทอล

แฟรคทอล คือ รูปทรงทางเรขาคณิตที่สามารถนำมาอธิบายรูปทรงต่างๆที่มีความซับซ้อนที่พบได้โดยทั่วไปตามธรรมชาติ เช่น รูปทรงของชายฝั่ง ภูเขา หรือกระทั่งก้อนเมฆ โดยแฟรคทอลเป็นรูปแบบที่มีการทำซ้ำไปเรื่อยๆ อย่างไม่มีที่สิ้นสุดซึ่งจะหลากหลายไปตามแต่กลุ่มของสูตรที่ใช้เป็นการผสมผสานกันของศิลปะและรูปทรงเรขาคณิต

คุณสมบัติที่สำคัญสองประการของแฟรคทอล คือ มีความเหมือนกันในตัวในทุกๆอัตราส่วน และ มิติของ แฟรคทอลจะต้องมากกว่ามิติทางเรขาคณิตทั่วๆไป

ความเหมือนกันในทุกๆอัตราส่วน หมายถึง เมื่อทำการมองแยกเป็นส่วนๆ ไม่ว่าจะเป็นส่วนที่เล็กลงไปมากเท่าไรก็ตามจะพบว่ามีรายละเอียดที่เหมือนกันในทุกๆ ส่วน เช่น



รูปที่ 2.1 ความเหมือนกันในทุกๆอัตราส่วน

มิติของแฟรคทอลจะต้องมากกว่ามิติทางเรขาคณิตทั่วๆไป คือ ในมิติของรูปทรงทางเรขาคณิตโดยทั่วไปจะเป็นเลขจำนวนเต็ม เช่น ศูนย์มิติเป็นจุด หนึ่งมิติเป็นเส้นตรงหรือเส้นโค้ง สองมิติเป็นลักษณะของวงกลม สี่เหลี่ยมหรือสามเหลี่ยมต่างๆ และสามมิติ เช่น ลักษณะของทรงกลมและลูกบาศก์ แต่ในมิติของรูปทรงแฟรคทอลจะมีความพิเศษออกไป คือ จะอยู่ระหว่างมิติที่เป็นจำนวนเต็มสองตัว ดังนั้นในขณะที่เส้นตรงมีมิติเป็นหนึ่งมิติ แต่โค้งของแฟรคทอลจะมีมิติอยู่ระหว่างหนึ่งมิติกับสองมิติ ขึ้นอยู่กับพื้นที่ที่ลดลงจากการบิดและโค้ง ยิ่งรูปทรงมีความแบนมากเท่าไหร่ก็ยิ่งเข้าใกล้สองมิติมากเท่านั้น ยกตัวอย่างเช่น ในการนำแฟรคทอลไปสร้างภาพภูเขา มิติ

ของภาพจะอยู่ระหว่างสองกับสามมิติ โดยพื้นผิวที่สร้างเป็นเนินเขาเตี้ยๆ และค่อนข้างกว้างจะมีลักษณะเข้าใกล้สองมิติ แต่หากเป็นภาพภูเขาที่มีขนาดใหญ่และขรุขระ จะมีลักษณะเข้าใกล้สามมิติ

แฟรคทอลนั้นมีหลายชนิดแตกต่างกัน โดยชนิดที่เป็นที่นิยมจะมีอยู่สองชนิด คือ คอมเพล็คนัมเบอร์แฟรคทอล และ อิตเทอเรตเต็ต ฟังก์ชั่น ซิสเต็ม(ไอเอฟเอส) แฟรคทอล โดยในปัญหาพิเศษฉบับนี้เราจะใช้แฟรคทอลไอเอฟเอส

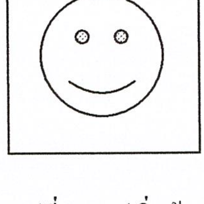
2.2 ทฤษฎีระบบสมการทำซ้ำ

ระบบสมการทำซ้ำ (Iterated Function System) เป็นระบบสมการซึ่งประกอบไปด้วยหลากหลายสมการโดยที่ทุกสมการใช้ในการแปลงภาพ(Transformation) ประกอบไปด้วยสมการต่างๆ ดังนี้

- การเปลี่ยนแปลงตำแหน่ง (Translation)
- การหมุนภาพ (Rotation)
- การย่อ-ขยายภาพ (Scaling)
- การสะท้อนภาพ (Reflection)
- การบิดภาพ (Shearing)

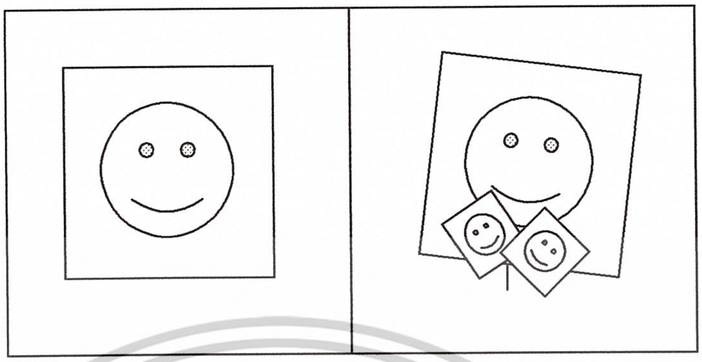
หลักการคร่าวๆ ของทฤษฎีนี้คือการทำซ้ำของตัวสมการไปเรื่อยๆ โดยที่เอาท์พุทที่ได้จากการคำนวณครั้งแรก จะเป็น อินพุท ให้กับการคำนวณที่ครั้งสอง และ เอาท์พุท ที่ได้จากการคำนวณครั้งที่สองนี้ก็จะเป็น อินพุท ให้กับการคำนวณครั้งที่สามต่อไป ทำเช่นนี้ไปเรื่อย ๆ เปรียบเสมือนกับการนำรูปไปถ่ายเอกสาร แล้วจัดเรียงรูปแบบของรูปใหม่ (โดยใช้สมการในการแปลงภาพที่ได้กล่าวไปแล้วข้างต้น) แล้วนำไปถ่ายเอกสารออกมาอีกครั้ง ทำเช่นนี้ไปเรื่อยๆ จนกระทั่งได้รูปที่ต้องการโดยความละเอียดหรือลักษณะความเหมือนจริงของภาพที่ออกมานั้นจะขึ้นกับจำนวนครั้งในการทำซ้ำ

เพื่อให้สามารถเห็นรายละเอียดและทำความเข้าใจได้ง่ายยิ่งขึ้นจะขอยกตัวอย่างการใช้ไอเอฟเอส ในการสร้างรูปใบเฟิร์นโดยใช้รูปเริ่มต้นดังภาพ



รูปที่ 2.2 รูปเริ่มต้น

เราจะนำภาพเริ่มต้นที่ได้มาทำการแปลงให้อยู่ในรูปแบบที่ต้องการ โดยในกรณีที่ต้องการสร้าง ใบเฟิร์น เราจะต้องทำการจัดวางภาพทั้งหมด 4 ภาพเพื่อนำมาเป็นรูปแบบในการทำซ้ำ โดยทำการย่อ หมุน เปลี่ยนตำแหน่ง ภาพเริ่มต้นให้อยู่ในลักษณะเหมือนภาพทางด้านขวามือ

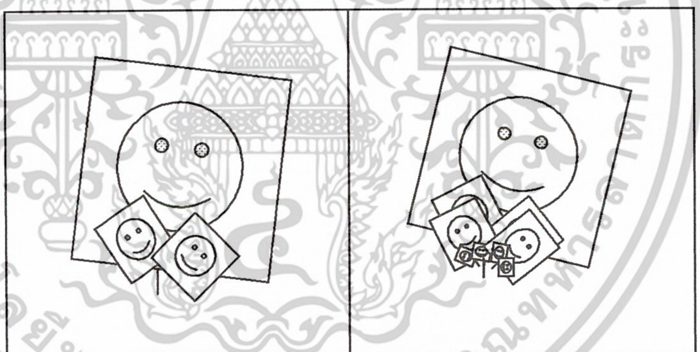


กรอบที่ 1 (รูปเริ่มต้น) — *กรอบที่ 2 (รูปแบบในการจัดวาง)*

รูปที่ 2. 3 ก — รูปที่ 2.3 ข

เริ่มการทำซ้ำไปเรื่อยๆ โดยจะมองรูปในกรอบที่ 2 ในรูปที่ 1 เสมือนเป็นรูปเริ่มต้นใหม่ในการทำซ้ำครั้งต่อไป โดยจะขอแสดงการทำซ้ำคร่าวๆ ดังนี้

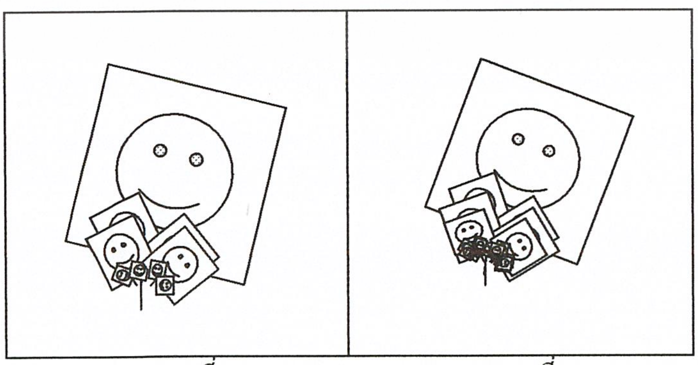
ทำซ้ำครั้งที่ 1



กรอบที่ 1 — *กรอบที่ 2*

รูปที่ 2.4 ก — รูปที่ 2.4 ข

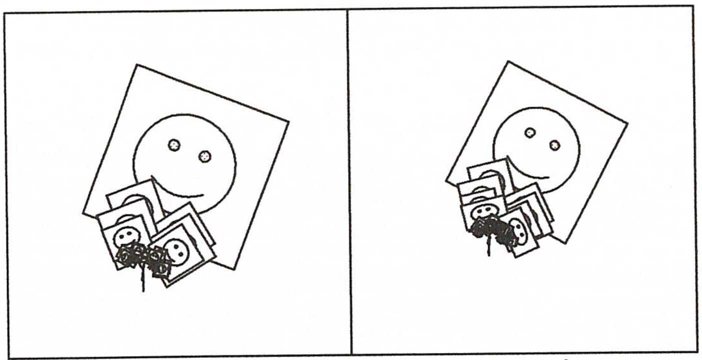
ทำซ้ำครั้งที่ 2



กรอบที่ 1 — *กรอบที่ 2*

รูปที่ 2. 5 ก — รูปที่ 2.5 ข

ทำซ้ำครั้งที่ 3



กรอบที่ 1 *กรอบที่ 2*

รูปที่ 2.6 ก รูปที่ 2.6 ข

และถ้าทำตามหลักการนี้ไปเรื่อยๆ สุดท้ายแล้วก็จะได้เป็นรูปใบเฟิร์นที่สมบูรณ์ดังนี้

รูปที่ 2.7

2.3 การแปลงภาพเรขาคณิตในระบบภาพ 2 มิติ

โดยพื้นฐานแล้วภาพของระบบคอมพิวเตอร์กราฟิก ถูกสร้างขึ้นด้วยส่วนของเส้นตรงหรือเวกเตอร์มาประกอบกัน ซึ่งเราสามารถกำหนดเส้นตรงหรือเวกเตอร์เหล่านี้ได้ด้วยจุดปลายทั้งสองของมัน การที่จะเปลี่ยนแปลงลักษณะภาพที่วาดจำเป็นต้องใช้การคำนวณทางคณิตศาสตร์กระทำกับจุดต่างๆเหล่านี้ วิธีหนึ่งที่จะช่วยให้เข้าใจได้ง่ายก็คือการใช้เมตริกซ์เข้าช่วย แต่สำหรับผู้ที่ไม่ถนัดการใช้เมตริกซ์ก็อาจจะจำสูตรไปใช้งานได้เช่นกัน

เราใช้เมตริกซ์ 2 มิติเพื่อช่วยทำการแปลง โดยแทนจุดต่างๆ ด้วยเมตริกซ์ขนาด 1 x 2 เช่น จุด (0.5,1) แทนด้วยเมตริกซ์ $[\,0.5 \quad 1\,]$ เมื่อต้องทำการแปลงใดๆ ก็เพียงแต่หาเมตริกซ์ที่เหมาะสมมาคูณกับเมตริกซ์ของจุด ผลลัพธ์ที่ได้คือเมตริกซ์ใหม่ ซึ่งก็คือจุดหรือตำแหน่งใหม่ของจุดปลายของเส้นตรงต่างๆ ทำให้เส้นเวกเตอร์เหล่านี้มีลักษณะที่เปลี่ยนไป เมตริกซ์ที่นำมาคูณเพื่อเปลี่ยนตำแหน่งของจุดนี้เราเรียกว่า เมตริกซ์การแปลง (Transformation matrix)

2.3.1 การแปลงแบบสเกล(Scaling)

ยกตัวอย่างว่าเรามีจุด $P_1 = [\, x_1 \quad y_1 \,]$ ซึ่งเป็นเมตริกซ์ขนาด 1 x 2 ถ้าเราคูณเมตริกซ์นี้ด้วยเมตริกซ์ T ขนาด

2 x 2 เราจะได้เมตริกซ์ขนาด 2 x 1 เมตริกซ์ใหม่ (P_2) กลับมาโดยที่

$$P_2 = [\, x_2 \quad y_2 \,] = P_1 T$$

การคูณเมตริกซ์นี้จะเป็นการคูณทางซ้าย และ T เป็นเมตริกซ์การแปลงที่จะแปลงจุด P_1 ไปเป็นจุด P_2 ถ้าเรานำเมตริกซ์การแปลง T นี้คูณเข้ากับจุดทุกๆ จุด เราจะได้ภาพที่มีลักษณะเปลี่ยนแปลงไป แต่จะเปลี่ยนแปลงไปอย่างไรนั้นขึ้นอยู่กับค่าต่างๆ ในเมตริกซ์ T เช่น

ถ้าเราคูณด้วย เมตริกซ์เอกลักษณ์ (Identity matrix)

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

$$P_2 = P_1 T = \begin{bmatrix} X_1 & Y_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} X_1 & Y_1 \end{bmatrix} = P_1$$

จุด P_1 กับ P_2 จะเป็นจุดเดียวกัน คือไม่มีการเปลี่ยนแปลงใดๆ แต่ถ้าเราเลือกเมตริกซ์ T_1 เป็น

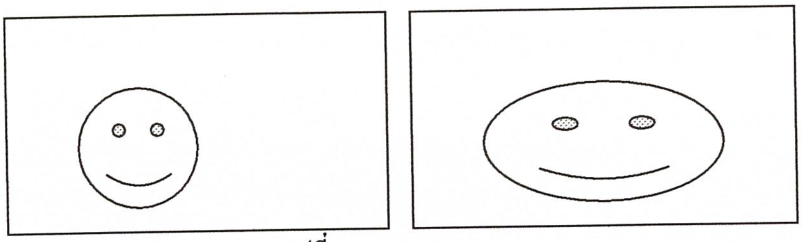
$$T_1 = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$P_2 = P_1 T_1 = \begin{bmatrix} X_1 & Y_1 \end{bmatrix} \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 2X_1 & Y_1 \end{bmatrix}$$

ผลลัพธ์ที่ได้คือโคออร์ดิเนต X ของทุกๆ จุด จะมีค่ามากขึ้นเป็น 2 เท่าของค่าเดิม เส้นตรงในแนวนอนก็จะมีความยามเป็น 2 เท่าของภาพเดิม ภาพใหม่ที่เปลี่ยนไปจะมีความสูงเท่าเดิม แต่ขยายออกทางแนวนอนออกจากจุดกำเนิด

(0 , 0) เป็น 2 เท่าของจากเดิม ดังตัวอย่างในรูป



รูปที่ 2.8 การขยายทางแนวนอน

ในทำนองเดียวกัน ถ้าเมตริกซ์การแปลง T_2 มีค่าเป็น

$$T_2 = \begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix}$$

โคออร์ดิเนต X ของทุกๆ จุด ก็จะลดลงเป็นครึ่งหนึ่งของภาพเดิม ภาพใหม่ที่ได้จะมีความสูงเท่าเดิม แต่จะถูกบีบให้แคบลงทางแนวนอนเป็นครึ่งหนึ่ง

หากเราลองขยายภาพออกทางแนวนอนเป็น 2 เท่า แล้วบีบภาพใหม่ให้แคบลงเป็นครึ่งหนึ่ง เราจะได้ภาพที่มีขนาดเท่าเดิมกลับมา

$$P_1 = (P_1T_1)T_2 \quad P_1(T_1T_2)$$

เราสามารถตรวจสอบคำตอบของเราได้โดย นำเมตริกซ์ T_1 และ T_2 มาคูณกัน

$$\begin{aligned} T_1T_2 &= \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} 0.5 & 0 \\ 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} (2*0.5)+(0*0) & (2*0)+(0*1) \\ (0*0.5)+(1*0) & (0*0)+(1*1) \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{aligned}$$

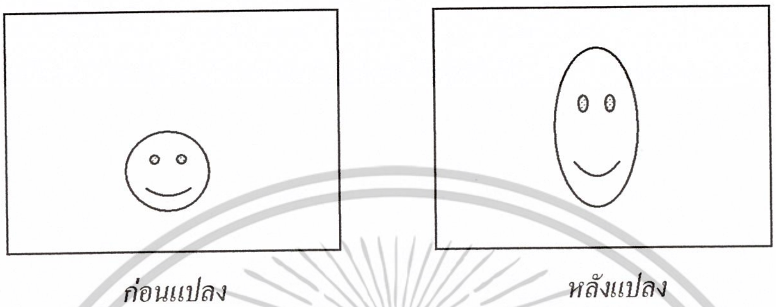
ผลลัพธ์ที่ได้คือเมตริกซ์เอกลักษณ์ นั่นคือ เราจะได้ภาพเดิมก่อนการแปลง เราสามารถที่จะขยายภาพให้สูงขึ้น หรือกดภาพให้เตี้ยลงในแนวตั้งได้เช่นกัน โดยการหาเมตริกซ์การแปลงที่ทำให้ค่าโคออร์ดิเนต Y เปลี่ยนโดยที่ค่า X ไม่เปลี่ยน ตัวอย่างเช่น เมตริกซ์การแปลง T_3

$$T_3 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$P_2 = P_1T_3 = \begin{bmatrix} X_1 & Y_1 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

$$= \begin{bmatrix} X_1 & 2Y_1 \end{bmatrix}$$

จะได้ภาพที่มีความสูงเพิ่มขึ้นจากเดิม 2 เท่า ดังในภาพ



รูปที่ 2.9 การขยายทางแนวตั้ง

ถ้าเรานำเมตริกซ์ T_1 และ T_3 มาคูณกับ P_1 เราก็จะได้ภาพที่มีการขยายออกทั้งแนวนอนและแนวตั้งเป็น 2 เท่าของภาพเดิม หรือกล่าวได้ว่า จะได้ภาพที่มีความโตเป็น 2 เท่าดังในรูป

$$P_2 = P_1T_1T_3 = P_1(T_1T_3)$$

รูปที่ 2.10 การขยายทางแนวนอนและแนวตั้ง

เรานำเมตริกซ์ T_1 และ T_3 มาคูณกันเพื่อให้ได้เมตริกซ์ใหม่ T_4 เมตริกซ์เดียว

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

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

เราอาจเขียนเมตริกซ์การแปลงแบบนี้ในรูปทั่วๆ ไปคือ

$$S = \begin{bmatrix} S_X & 0 \\ 0 & S_Y \end{bmatrix}$$

S_X เรียกว่า สเกลแฟคเตอร์ (Scale Factor) สำหรับโคออร์ดิเนต X ซึ่งจะมีผลต่อขนาดของภาพในแนวนอนหรือแนวแกน X

S_Y เรียกว่า สเกลแฟคเตอร์ สำหรับโคออร์ดิเนต Y ซึ่งจะมีผลต่อขนาดของภาพ ในแนวตั้งหรือแนวแกน Y

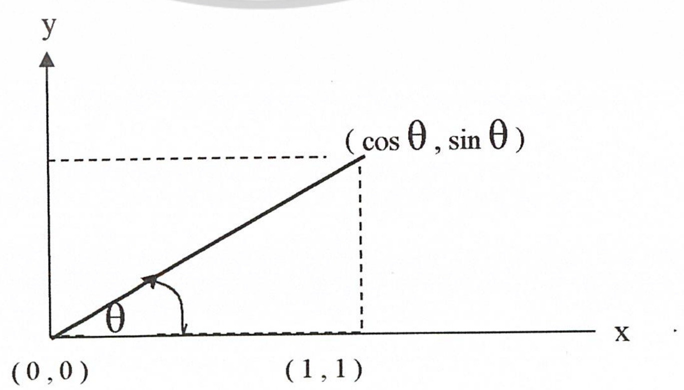
เมตริกซ์ S เป็นเมตริกซ์การแปลงซึ่งจะมีผลต่อขนาดและสัดส่วนของภาพ เราเรียกว่า เมตริกซ์การแปลงแบบสเกล (Scaling Transformation matrix) การแปลงโดยการคูณด้วยเมตริกซ์ S เราเรียกว่าเป็น การแปลงแบบสเกล (Scaling Transformation)

สังเกตว่าการสเกลภาพจะทำให้ทุกๆ จุดมีการเปลี่ยนแปลง ยกเว้นเพียงจุดเดียวคือจุดกำเนิด (0 , 0) ดังนั้นนอกจากขนาดของภาพที่จะเปลี่ยนไปแล้ว ตำแหน่งของจุดต่างๆ ก็เปลี่ยนไปด้วย ถ้า S_X มีค่ามากกว่า 1 ก็จะทำให้ภาพเลื่อนไปทางขวา (สำหรับภาพที่อยู่ทางขวาของแกน Y) และมีความกว้างมากขึ้นด้วย ถ้า S_X มีค่าน้อยกว่า 1 ภาพก็จะเลื่อนไปทางซ้ายและมีความแคบลง ในทำนองเดียวกัน ถ้า S_Y มีค่ามากกว่า 1 ก็จะทำให้ภาพขยายห่างออกจากแกน X และมีความสูงเพิ่มขึ้น ถ้า S_Y มีค่าน้อยกว่า 1 ก็จะทำให้ภาพหดเข้าหาแกน X และเตี้ยลง

2. 3.2 การแปลงแบบหมุน (Rotation Transformation)

เป็นวิธีการแปลงภาพโดยการหมุนจุด (หรือภาพ) ในทิศทางทวนเข็มหรือตามเข็มนาฬิกา โดยมีจุดศูนย์กลางของการหมุนอยู่ที่จุดกำเนิด สิ่งที่เราต้องการทราบก็คือเมตริกซ์ซึ่งมาคูณจุดที่เราต้องการหมุนเพื่อให้ได้ตำแหน่งใหม่ เราเรียกเมตริกซ์นี้ว่า เมตริกซ์การแปลงแบบหมุน (Rotation Transformation Matrix)

สมมติว่าเราทำการหมุนจุด (1 , 0) ไปในทิศทางบวก (ทวนเข็มนาฬิกา) เป็นมุม θ องศา ตำแหน่งใหม่จะเป็นจุด (cos(θ), sin (θ)) ดังภาพ



รูปที่ 2.11 การหมุนจุด (1 , 0) ไปในทิศทางทวนเข็มนาฬิกาเป็นมุม θ องศา

ถ้าเมตริกซ์การแปลงแบบหมุน คือ

$$S = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \qquad \ldots\ldots\ldots\ldots\ldots(1)$$

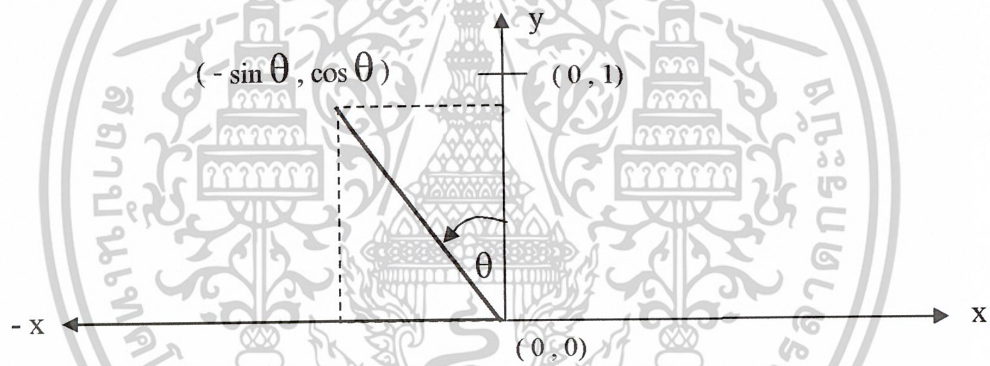
ดังนั้นจะได้ว่า

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

$$= \begin{bmatrix} a & b \end{bmatrix} \qquad \ldots\ldots\ldots\ldots\ldots(2)$$

ถ้าเราหมุนจุด (0 , 1) ไปในทิศทางบวกเป็นมุม θ เช่นกัน นั่นคือใช้เมตริกซ์การแปลงตัวเดิม ตำแหน่งใหม่จะเป็น

(- sin θ, cos θ) ดังรูป



รูปที่ 2.12 การหมุนจุด (0 , 1) ไปในทิศทางทวนเข็มนาฬิกาเป็นมุม θ องศา

ดังนั้น

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

$$= \begin{bmatrix} c & d \end{bmatrix}$$

จากสมการ (1) และ (2) จะได้ว่า

$$a = \cos(\theta)$$

$$b = \sin(\theta)$$

$$c = \cos(\theta)$$

$$d = \sin(\theta)$$

ดังนั้นเมตริกซ์การแปลงแบบหมุนทวนเข็มนาฬิกา คือ

$$R = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(3)$$

วิธีการหาเมตริกซ์การแปลงแบบหมุนนี้หาได้จากการพิสูจน์ทางตรีโกณมิติจนกระทั่งได้ผลลัพธ์เช่นเดียวกับสมการ (3)

ในสมการ (3) เป็นเมตริกซ์การแปลงที่ใช้กับการหมุนในทิศทางทวนเข็มนาฬิกา สำหรับการหมุนในทิศทางตามเข็มนาฬิกา คือ

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

ตัวอย่างเช่น ต้องการหมุนจุด P_1 (3 , 2) ในทิศทางทวนเข็มเป็นมุม 30 องศา เมตริกซ์การแปลงจะเป็น

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

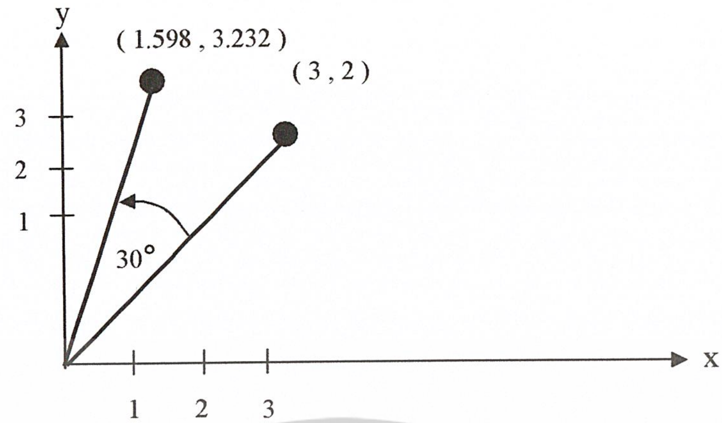
และจุดใหม่หลังการหมุน P_2 คือ

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

$$= \begin{bmatrix} 1.598 & 3.232 \end{bmatrix}$$

2. 3. 3 การแปลงแบบย้ายตำแหน่งและโฮโมจีเนียสโคออร์ดิเนต

การย้ายตำแหน่ง (Translation) เป็นการเลื่อนตำแหน่งของภาพทั้งภาพไปยังตำแหน่งอื่นๆ เป็นระยะทางเท่ากันทั้งหมด โดยขนาดของภาพไม่เปลี่ยนแปลงและไม่ทำให้ภาพเอียงไปจากแนวเดิม การย้ายตำแหน่งนี้ทำได้โดยการบวกจุดทุกๆจุดของภาพด้วยระยะทางที่ต้องการให้ภาพเลื่อนไป ดังนั้น



รูปที่ 2. 13 แสดงการหมุนจุด (3 , 2) ในทิศทางทวนเข็มเป็นมุม 30 องศา

$$x' = x + t_x$$

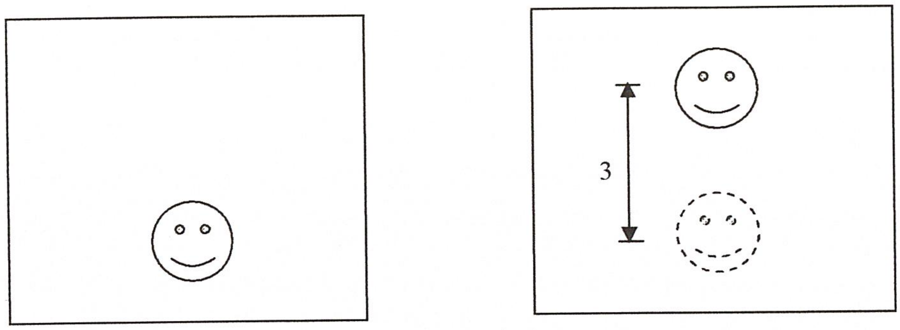
และ

$$y' = y + t_y$$

นั่นคือการย้ายตำแหน่งของภาพไปในแนวแกน x เป็นระยะ t_x หน่วย และแนวแกน y เป็นระยะทาง t_y หน่วย โดยที่ t_x และ t_y เป็นระยะทางที่ต้องการให้ภาพเลื่อนไปในแนวแกน x และแกน y ตามลำดับ ตัวอย่างเช่น ต้องการย้ายภาพให้เลื่อนไปอยู่ทางขวา 3 หน่วย เราก็บวก 3 กับค่า x ของทุกๆจุดของภาพ ดังภาพ

รูปที่ 2.14 การย้ายภาพให้เลื่อนไปอยู่ทางขวา 3 หน่วย

หรือถ้าต้องการย้ายภาพขึ้นข้างบน 3 หน่วย เราก็บวก 3 เข้ากับค่า Y ของทุกๆ จุด ดังภาพ



ก่อนย้าย *หลังย้าย*

รูปที่ 2.15 การย้ายภาพขึ้นข้างบน 3 หน่วย

ในทางตรงข้าม ถ้าต้องการเลื่อนภาพไปทางซ้ายหรือลงข้างล่าง ก็นำจำนวนลบมาบวกเข้าไป เพื่อให้ค่า X หรือค่า Y ลดลง การย้ายภาพสามารถย้ายขึ้น – ลง และ ซ้าย – ขวา ได้พร้อมๆ กัน โดยเปลี่ยนทั้งค่า X และ Y ของทุกๆจุดพร้อมๆ กัน

จากที่กล่าวมาจะเห็นว่าการแปลงแบบย้ายนั้นไม่มีเมตริกซ์ที่มาคูณเพื่อให้ได้ตำแหน่งใหม่ของจุดต่างๆ เราเพียงแค่นำค่าที่เหมาะสมมาบวกเข้ากับค่า X และ Y ของจุดต่างๆ เท่านั้น ลักษณะเช่นนี้ทำให้เรารวมการแปลงแบบย้ายเข้ากับการแปลงแบบสเกลหรือการแปลงแบบหมุนไม่ได้ เราไม่สามารถสร้างเมตริกซ์เพียงเมตริกซ์เดียว มาคูณกับเมตริกซ์ของจุดต่างๆ เพื่อให้เกิดการแปลงหลายๆ แบบรวมกัน เกิดความไม่สะดวกในการทำงาน แต่ปัญหานี้เราสามารถแก้ไขได้โดยใช้โฮโมจีเนียสโคออร์ดิเนต (Homogeneous Coordinate) เข้าช่วย โฮโมจีเนียสโคออร์ดิเนตจะใช้เมตริกซ์ขนาด 3x3 แทนเมตริกซ์ขนาด 2x2 ที่ได้กล่าวมาแล้ว จุดหรือตำแหน่งใหม่จะขึ้นอยู่กับจำนวน 3 จำนวนแทนที่จะเป็น 2 จำนวน (โคออร์ดิเนต X และ Y ของจุด) โดยเพิ่มโคออร์ดิเนตใหม่ W ขึ้นมา จำนวน 3 จำนวนนี้ได้แก่ ผลคูณของโคออร์ดิเนต X กับ W ผลคูณของโคออร์ดิเนต Y กับ W และโคออร์ดิเนต W ดังนั้นโคออร์ดิเนตของจุดต่างๆ (x , y) จะถูกแทนด้วยจำนวน 3 จำนวนคือ (XW , YW , W) ถ้าเรามีโคออร์ดิเนตใหม่นี้ และต้องการทราบตำแหน่ง (x , y) เดิมของมัน ก็ทำได้โดยนำเอาโคออร์ดิเนตที่ 3 หารสองโคออร์ดิเนตแรกเท่านั้น โคออร์ดิเนต W นี้จะถูกใช้งานอย่างแท้จริงในการแปลงในระบบ 3 มิติ ส่วนกรณี 2 มิติเราจะให้ค่า W เป็น 1 เสมอ

ในโฮโมจีเนียสโคออร์ดิเนต เมตริกซ์การแปลงแบบสเกลจะถูกเปลี่ยนจากเดิม

$$\begin{bmatrix} S_X & 0 \\ 0 & S_Y \end{bmatrix}$$

เป็น

$$S = \begin{bmatrix} S_X & 0 & 0 \\ 0 & S_Y & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots(4)$$

เราทดสอบโดยนำเมตริกซ์การแปลงสมการ (4) มาคูณกับจุด $P_1(x, y)$ ซึ่งต้องแทนด้วยเมตริกซ์ [xw yw w]

$$P_2 = P_1S$$

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

$$= [\,S_Xxw \quad S_Yyw \quad w\,]$$

เมื่อนำโคออร์ดิเนต W มาหารสองโคออร์ดิเนตแรก จะได้จุด P_2 เป็น (S_Xxw , S_Yyw) ซึ่งตรงกับการแปลงสเกลที่กล่าวมาแล้ว

เมตริกซ์การแปลงแบบหมุน ในทิศทางทวนเข็มนาฬิกาจากเดิม

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

จะเปลี่ยนเป็น

$$R = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{.....................(5)}$$

นำเมตริกซ์ในสมการที่ 5 คูณกับจุด $P_1(xw, yw, w)$ เราจะได้

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

$$= [xw*\cos(\theta)-yw*\sin(\theta) \quad xw*\sin(\theta) + yw*\cos(\theta) \quad w]$$

ซึ่งจะได้จุด P_2 ที่ได้จากการหมุนจุด P_1 คือ ($xw*\cos(\theta)-yw*\sin(\theta)$, $xw*\sin(\theta) + yw*\cos(\theta)$,w) ตรงกับการแปลงแบบหมุนที่กล่าวมาแล้ว

สำหรับการแปลงแบบย้าย เมื่อต้องการเลื่อนภาพหรือจุดไปทางแนวนอน t_x และไปทางแนวตั้ง t_y จะได้เมตริกซ์การแปลงแบบย้ายคือ

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix} \qquad \text{.....................(6)}$$

เราสามารถพิสูจน์ได้ว่าเมตริกซ์ในสมการที่ 6 ใช้งานได้โดยนำไปคูณกับจุด $P_1(xw,yw,w)$

$$P_2 = P_1T$$

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

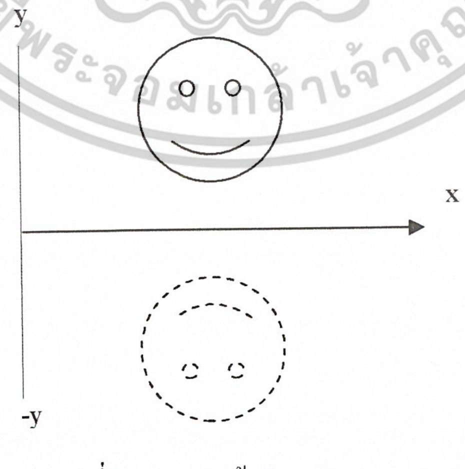
$$P_2 = [\, xw + t_xw \quad yw + t_yw \quad w\,]$$

เราจะได้จุด $P_2(xw + t_xw, yw + t_yw, \quad w)$ ดังนั้นจุด P_2 ก็คือ $(x + t_x , y + t_y)$

2. 3. 4 การสะท้อนภาพ (Reflection)

การสะท้อนภาพจะทำให้ภาพที่ได้ออกมานั้นมีลักษณะเหมือนกับภาพเริ่มต้น เพียงแต่ภาพจะมีลักษณะเหมือนกับการสะท้อนภาพของกระจก คือ จากตำแหน่งซ้ายของภาพเริ่มต้น เมื่อทำการสะท้อนภาพจะไปอยู่ที่ตำแหน่งขวาเป็นต้น โดยจะพิจารณาตามแกนที่สะท้อน ได้แก่ การสะท้อนภาพตามแนวแกน x และการสะท้อนภาพตามแนวแกน y

การสะท้อนภาพตามแนวแกน x

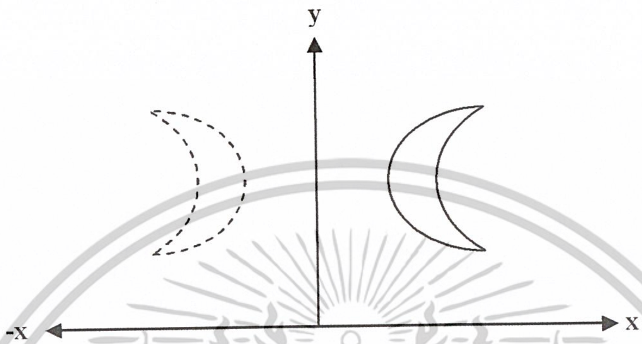


รูปที่ 2.16 การสะท้อนภาพตามแนวแกน x

$$x' = x$$

$$y' = -y$$

การสะท้อนภาพตามแนวแกน y



รูปที่ 2.17 การสะท้อนภาพตามแนวแกน y

$$x' = -x$$

$$y' = y$$

2. 3. 5 การบิดภาพ (Shearing)

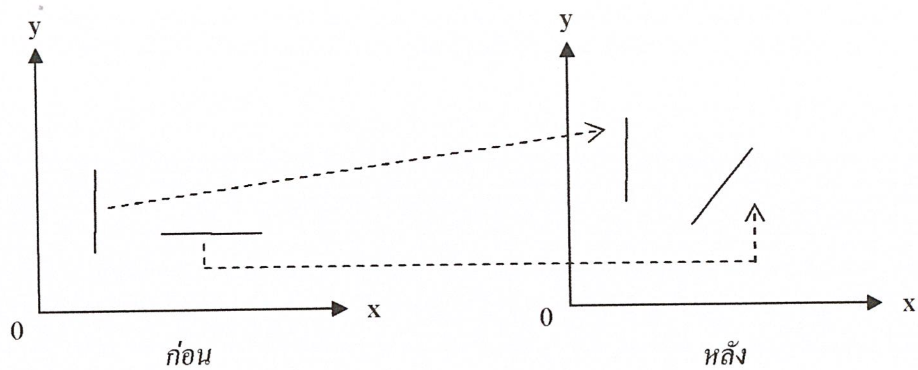
การบิดภาพจะทำให้บางส่วนของภาพหรือภาพทั้งหมดเกิดการบิดเบือนขึ้น เราจะพิจารณาเพียง 2 แบบคือ การบิดภาพทางแกน x และการบิดภาพทางแกน y

การบิดภาพทางแกน y จะทำให้เกิดการย้ายจุด (x,y) ไปยังจุด (x', y') โดยที่

$$x' = x$$

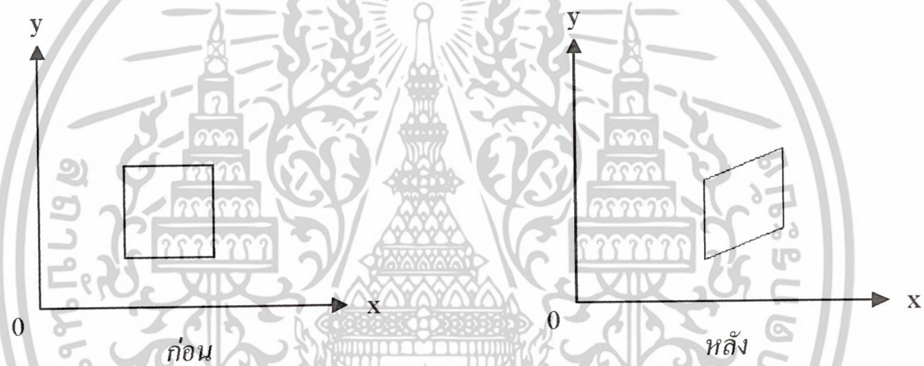
$$y' = Shy * x + y \qquad , Shy \neq 0$$

การบิดภาพทางแกน y จะทำให้จุดต่างๆ ในแกน y เลื่อนขึ้นหรือเลื่อนลงขึ้นอยู่กับเครื่องหมายของแฟคเตอร์ Shy เส้นตรงในแนวนอนจะถูกเปลี่ยนให้เป็นเส้นตรงในแนวเฉียงด้วยความลาดชันเท่ากับ Shy พิจารณาจากรูป



รูปที่ 2.18 การบิดภาพทางแกน y ทำให้เส้นแนวนอนเปลี่ยนไปเป็นเส้นแนวเฉียง

ในรูปต่อไปนี้จะเป็นการแสดงการบิดภาพสี่เหลี่ยมทางแกน y ซึ่งทำให้เกิดภาพเดิมซึ่งเป็นภาพสี่เหลี่ยมผืนผ้าเปลี่ยนแปลงไปเป็นภาพสี่เหลี่ยมด้านขนาน



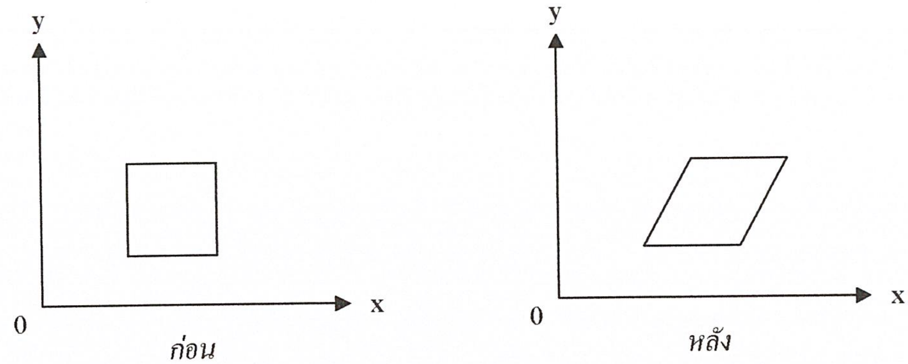
รูปที่ 2.19 การบิดภาพสี่เหลี่ยมทางแกน y

สำหรับการบิดภาพทางแกน x จะให้ผลตรงข้ามกับการบิดภาพทางแกน y กล่าวคือ จุด (x , y) ของภาพจะถูกแปลงเป็นจุด (x' , y') โดยที่

$$x' = x + Shx * y \quad , Shx \neq 0$$

$$y' = y$$

ในกรณีที่เป็นเส้นในแนวนอนก็จะถูกย้ายไปทางซ้ายหรือทางขวา ขึ้นอยู่กับเครื่องหมายของแฟกเตอร์ Shx ส่วนเส้นตรงในแนวดิ่งก็จะถูกบิดไปเป็นเส้นตรงในแนวเฉียงด้วยความลาดชัน Shx

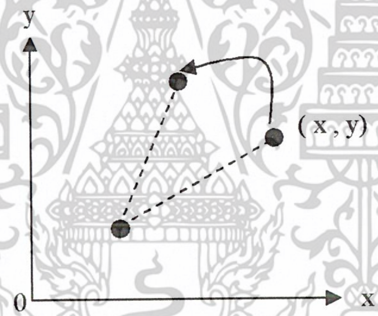


รูปที่ 2.20การบิดภาพสี่เหลี่ยมทางแกน

จากภาพจะเป็นการบิดภาพสี่เหลี่ยมทางแกน x ภาพจะเปลี่ยนจากสี่เหลี่ยมผืนผ้าให้กลายเป็นสี่เหลี่ยมด้านขนาน

2. 3. 6 การหมุนรอบจุดใด ๆ

การแปลงแบบหมุนที่กล่าวมาเป็นการหมุนรอบจุดกำเนิน (0 , 0) เท่านั้น แต่ในหัวข้อนี้เราจะหาเมตริกซ์การแปลงแบบหมุนที่ใช้หมุนรอบจุดใดๆ (x_c , y_c) ได้ดังรูป

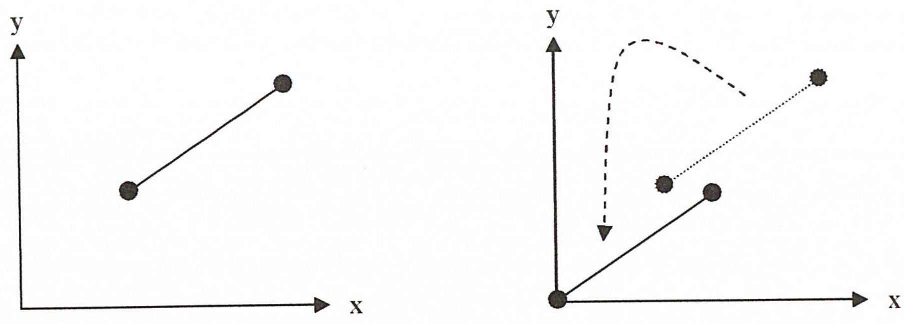


รูปที่ 2.21 การหมุนรอบจุด (x_c , y_c)

ขั้นตอนทั้ง 3 นี้จะต้องทำทีละขั้นตอน โดยต้องเรียงลำดับให้ถูกต้อง เพราะในการทำแต่ละขั้นตอน ต้องใช้เมตริกซ์การแปลงมาคูณเข้าไป และการคูณเมตริกซ์ไม่มีคุณสมบัติการสลับที่ดังนั้นถ้าคูณเมตริกซ์การแปลงไม่เรียงลำดับ จะได้ผลลัพธ์ที่ไม่ถูกต้อง

เมตริกซ์ที่จะย้ายจุด (x_c , y_c) ไปยังจุดกำเนิดก็คือ

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix}$$



ก)　　　　　　　　ข)

รูปที่ 2.22ขั้นตอนการหมุนจุดใดๆ (x_c , y_c)

เมตริกซ์สำหรับการหมุน คือ

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

และเมตริกซ์ที่เราจะย้ายกลับไปยังจุดเดิม คือ

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

นำเมตริกซ์ทั้ง 3 มาคูณเข้ากับจุดที่เราจะหมุน P(x , y) ได้จุดกำเนิดใหม่เป็น $P_1(x_1 , y_1)$

$$P_1 = [(PT_1) R] T_2$$

$$= [P(T_1 R)] T_2$$

$$= P(T_1 R T_2)$$

ดังนั้นเมตริกซ์การแปลงแบบหมุนรอบจุด *(x_c, y_c)* คือ

$$T_1 R T_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_c & -y_c & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_c & y_c & 1 \end{bmatrix}$$

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

สมการ คือ เมตริกซ์การแปลงที่หมุนภาพในทิศทางทวนเข็มนาฬิกาเป็นมุม a องศารอบจุด $P(x_c, y_c)$ เพื่อความสะดวกเราอาจเขียนเป็นสูตรสำหรับการหมุนจุด P(x , y) รอบจุดเป็นมุม a องศาได้เป็นจุดใหม่ P'(x' , y') ดังนี้

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

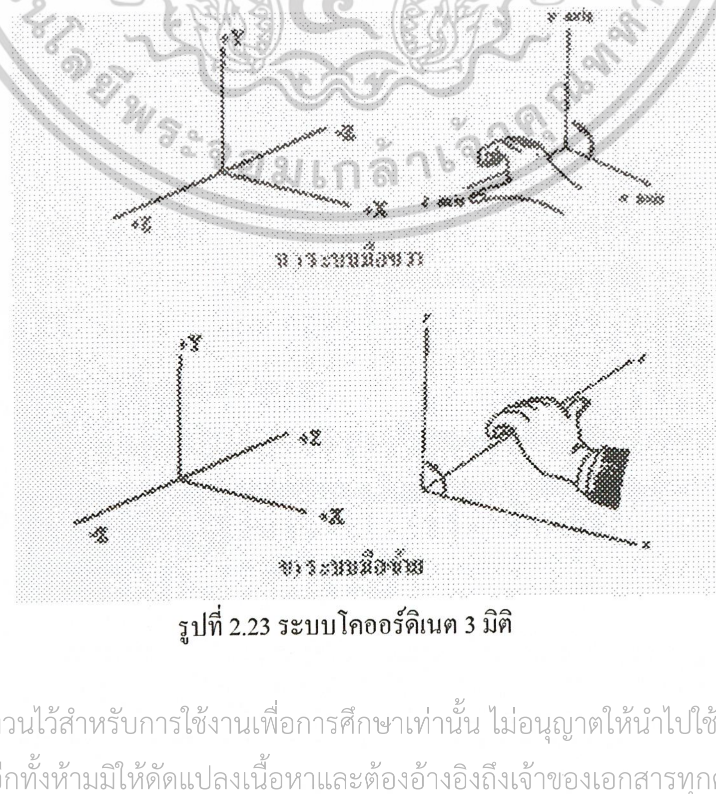
$$
x' = (x - x_c)\cos(\theta) - (y - y_c)\sin(\theta) + x_c
$$

$$
y' = (x - x_c)\sin(\theta) + (y - y_c)\cos(\theta) + y_c
$$

2.4. ระบบภาพ 3 มิติ

2.4.1 ระบบโคออร์ดิเนต

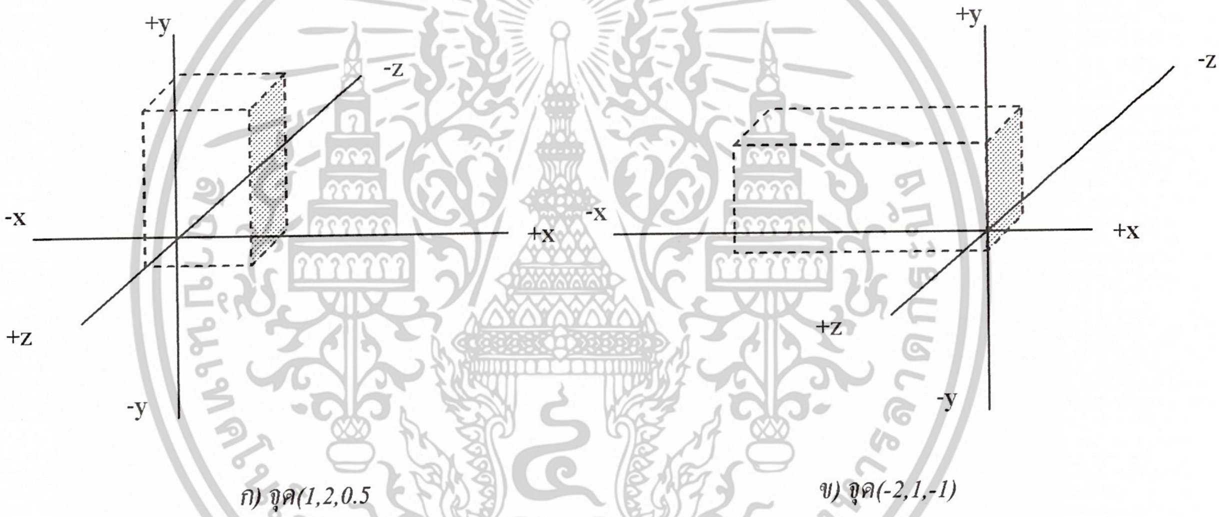
ในระบบ 2 มิติ มีแกนเพียง 2 แกนเท่านั้นคือ แกน X และแกน Y แต่ในระบบ 3 มิติต้องเพิ่มแกน Z เข้าไปอีกหนึ่งแกน การกำหนดทิศทางของแกน Z มา 2 แบบ คือ แบบระบบมือขวา และ แบบระบบมือซ้าย ดังแสดงดังรูป



รูปที่ 2.23 ระบบโคออร์ดิเนต 3 มิติ

ในระบบมือขวาเราใช้นิ้วโป้งของมือขวาชี้ไปในแกน +z งอนิ้วที่เหลือทั้ง 4 นิ้ว ทิศทางการหมุนของนิ้วทั้ง 4 จะหมุนจากแกน +x เข้าหาแกน +y ส่วนในระบบมือซ้าย เราใช้นิ้วโป้งของมือซ้ายชี้ไปในแนวแกน +z นิ้วทั้ง 4 จะหมุนจากแกน +x เข้าหาแกน +y โดยทั่วไปแล้วเรามักจะใช้โคออร์ดิเนตแบบระบบมือขวา เช่นในงานทางคณิตศาสตร์หรือการใช้งานในระบบภูมิศาสตร์ แต่สำหรับระบบคอมพิวเตอร์กราฟิก มักนิยมใช้ระบบมือซ้ายทั้งนี้เพราะเราจะถือว่าระนาบของจอภาพเป็นระนาบ xy (เป็นแกน x และ y) และให้ระยะความลึกเข้าไปหลังจอภาพมีค่าเป็นบวก(ค่าโคออร์ดิเนต z เป็นบวก)

การกำหนดจุดในระบบ 3 มิติ ต้องใช้จำนวน 3 จำนวน เพื่อเป็นการระบุว่าจุดนั้นห่างจากจุดกำเนิด (0,0,0) ไปตามแนวแกน xy และ z เป็นค่าเท่าใด เช่นจุด (1,2,0.5) คือจุดที่ห่างจากจุดกำเนิดไปในแนวแกน +x 1 หน่วย แกน +y 2 หน่วย และแกน +z 0.5 หน่วย จุด(-2,1,-1) คือจุดที่ห่างจากจุดกำเนิดไปในแนวแกน –x 2 หน่วย แกน +y 1 หน่วย และแกน –z 1 หน่วย ดังแสดงในรูป



รูปที่ 2.24 การกำหนดจุดในระบบ 3 มิติ

2.4.2 เส้นตรงและเวคเตอร์

เส้นตรงในระบบ 3 มิติ ยังคงกำหนดด้วยจุดปลายทาง 2 จุด เช่นเดียวกับในระบบ 2 มิติ เราใช้จุด 2 จุดนี้กำหนดความยาว และทิศของเส้นตรง สมการของเส้นตรงต้องใช้ 2 สมการประกอบกันซึ่งมีรูปแบบของสมการดังนี้

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

$$\frac{z - z_1}{x - x_1} = \frac{z_2 - z_1}{x_2 - x_1} \qquad \text{.....(1)}$$

โดยที่จุด (x_1,y_1,z_1) และ (x_2,y_2,z_2) คือจุดที่อยู่บนเส้นตรงเดียวกัน ตัวอย่างเช่น มีจุด(1,2,1) และจุด (0,1,2) อยู่บนเส้นตรงเดียวกัน สมการของเส้นตรงนี้หาได้โดยแทนค่า x_1 , y_1 , z_1 , x_2 , y_2 และ z_2 ลงในสมการ (1)

$$\frac{y-2}{x-1} = \frac{1-2}{0-1}$$

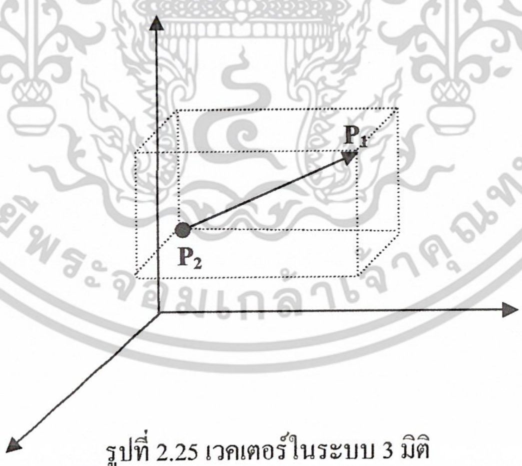
$$\frac{z-2}{x-1} = \frac{2-1}{0-1}$$

ดังนั้นสมการของเส้นตรงนี้คือ

$$y = x + 1$$

$$z = -x + 3$$

สำหรับเวคเตอร์ในระบบ 3 มิติ ก็คล้ายกับส่วนของเส้นตรงคือ สามารถกำหนดได้ด้วยจุด 2 จุดในระบบ แต่เวคเตอร์ต่างกับส่วนของเส้นตรงคือ เวคเตอร์มีทิศทางโดยจุดหนึ่งเป็นจุดเริ่มต้น และอีกหนึ่งจุดคือจุดปลาย ตัวอย่างเช่น ถ้าเรามีเวคเตอร์ที่มีจุดเริ่มต้นคือ P_1 (x_1, y_1, z_1) และมีจุดปลายคือ $P_2(x_2, y_2, z_2)$ เวคเตอร์นี้จะมีทิศทางแสดงดังภาพ



รูปที่ 2.25 เวคเตอร์ในระบบ 3 มิติ

ขนาดของเวคเตอร์นี้หาได้จากสมการ

$$\text{ขนาดของเวคเตอร์} = [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{1/2}$$

โดยทั่วไปเรามักสนใจเฉพาะทิศทางของเวคเตอร์เท่านั้นซึ่งเราสามารถแทนแนวทิศทางของเวคเตอร์ได้ด้วยเมตริกซ์ [x y z] ค่าของ x y และ z เป็นอัตราส่วนซึ่งแสดงว่าเวคเตอร์นั้นหันเหไปในแนวแกน x y และ z มากน้อยเท่าใด ตามลำดับสำหรับในระบบคอมพิวเตอร์กราฟิก เรามักใช้เมตริกซ์ [x y z] แทนเวคเตอร์นั้นไปเลย ถ้าจุดตั้งต้นของเวคเตอร์คือ $P_1(x_1, y_1, z_1)$ และจุดปลายของเวคเตอร์คือ $P_2(x_2, y_2, z_2)$ เราสามารถหาเวคเตอร์นี้ได้โดยที่ ค่า x y z คือ

$$x = x_2 - x_1$$

$$y = y_2 - y_1$$

$$z = z_2 - z_1$$

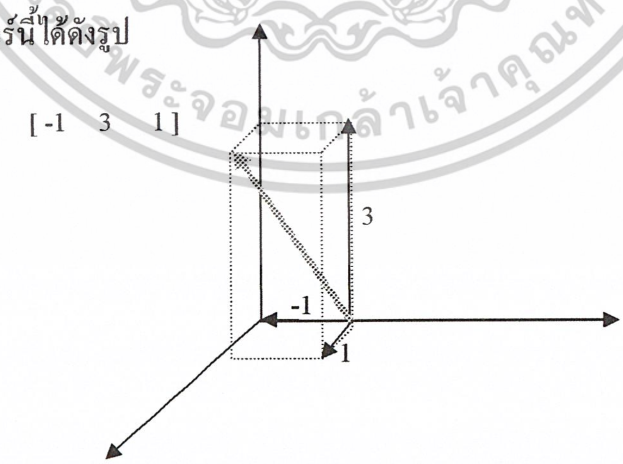
ยกตัวอย่างจุด P1 คือ (5, 2, 0) และ P2 คือ (4, 3, 3) ดังนั้น

$$x = 4-5 = -1$$

$$y = 3-0 = 3$$

$$z = 3-2 = 1$$

เวคเตอร์นี้จะหันเหไปในแนวแกน XY และ Z ด้วยอัตราส่วน -1 : 3 : 1 ตามลำดับซึ่งสามารถแสดงทิศทางของเวคเตอร์นี้ได้ดังรูป



รูปที่ 2.26 แสดงทิศทางของเวคเตอร์ที่มีอัตราส่วน -1 : 3 : 1

จะสังเกตได้ว่าเวคเตอร์ในรูปเป็นการแสดงเฉพาะแนวทิศทางของเวคเตอร์เท่านั้น ซึ่งเราไม่สนใจขนาดและจุดตั้งต้นของเวคเตอร์ ดังนั้นเราอาจใช้เวคเตอร์ที่มีขนาดเล็กกว่า หรือยาวกว่าก็ได้ ขอเพียงให้เป็นเวคเตอร์ที่มีทิศทางเดียวกันก็พอ นั้นคือค่าของ x y และ z ในเมตริกซ์ [x y z] สามารถเปลี่ยนแปลงได้ด้วยอัตราส่วนที่เท่ากัน เช่นเวคเตอร์ [-1 3 1] จะมีทิศทางเดียวกับเวคเตอร์ [-2 6 2] หรือ [-1/3 1 1/3] แค่เพื่อให้สอดคล้องกัน เรามักจะใช้เวคเตอร์ที่มีขนาดหนึ่งหน่วย (เรียกว่าเวคเตอร์หนึ่งหน่วย) เพื่อกำหนดทิศทาง เวคเตอร์หนึ่งหน่วย ที่มีทิศทางตรงกับเวคเตอร์ที่เกิดจากจุด $P_1(x_1, y_1, z_1)$ และ $P_2(x_2, y_2, z_2)$ คือ [X Y Z] โดยที่

$$X = (x_2 - x_1) / S$$

$$Y = (y_2 - y_1) / S$$

$$Z = (z_2 - z_1) / S$$

โดยที่ S คือขนาดเวคเตอร์ $= [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]^{1/2}$

ตัวอย่างเช่นถ้าต้องการหาเวคเตอร์หนึ่งหน่วย ที่มีทิศทางเดียวกันกับเวคเตอร์ [-1 3 1] ต้องทราบขนาดของเวคเตอร์ก่อน

$$S = [(-1)^2 + (3)^2 + (1)^2]^{1/2}$$

$$= (11)^{1/2}$$

$$= 3.31$$

ดังนั้นเวคเตอร์หนึ่งหน่วยคือ [-1 / 3.32 3 - 3.32 1 / 3.32]

= [-0.30 0.90 0.30]

2.4.3 ระนาบ

ในระบบภาพ 3 มิติ นอกเหนือจากจุดและเส้นตรงแล้ว เรายังต้องยุ่งเกี่ยวกับระนาบอีกด้วย ระนาบเปรียบได้กับแผ่นกระดาษราบเรียบที่ไม่มีความหนา แต่มีขนาดใหญ่ไม่จำกัดตั้งอยู่ในระบบโคออร์ดิเนตเช่นเดียวกันกับเส้นตรง ระนาบมีสมการในการกำหนดหรือบ่งชี้ระนาบนั้นๆ สมการระนาบมีรูปแบบดังนี้

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

เมื่อ A , B และ C คือค่าคงที่ และ (x, y ,z) คือจุดใดๆ ที่อยู่บนระนาบ

ถ้าเราทราบ 3 จุดที่อยู่บนระนาบเดียวกันคือ $P_1(x_1, y_1, z_1)$ และ $P_2(x_2, y_2, z_2)$ เราสามารถหาสมการของระนาบได้โดยแทนค่าจุดต่างๆ ลงบนสมการข้างต้นได้ดังนี้

$$Ax_1 + By_1 + Cz_1 + D_1 = 0$$

$$Ax_2 + By_2 + Cz_2 + D_2 = 0$$

$$Ax_3 + By_3 + Cz_3 + D_3 = 0$$

จากสมการทั้ง 3 ข้างต้นนี้ เราสามารถแก้สมการเพื่อหาค่าของ A , B และ C ได้คือ

$$A = y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2)$$

$$B = z_1(x_2 - x_1) + z_2(x_3 - x_1) + z_3(x_1 - x_2)$$

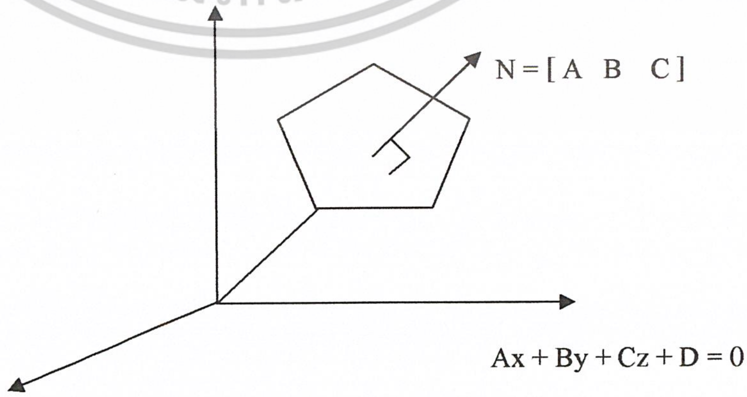
$$C = x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2)$$

$$D = -x_1(y_2z_3 - y_3z_2) - x_2(y_3z_1 - y_1z_3) - x_3(y_1z_2 - y_2z_1)$$

การจัดเรียงตัวของระนาบในระบบโคออร์ดิเนต สามารถกำหนดได้ด้วยเวคเตอร์หนึ่งเวคเตอร์ซึ่งเป็นเวคเตอร์ซึ่งมีทิศทางตั้งฉากกับพื้นผิวของระนาบเราเรียกเวคเตอร์นี้ว่าเวคเตอร์ตั้งฉาก (Normal Vector) ถ้าสมการของระนาบคือ

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

เวคเตอร์ตั้งฉากของระนาบนี้คือ [A B C] แสดงได้ดังรูป



รูปที่ 2.27 เวคเตอร์ [A B C] ซึ่งตั้งฉากกับระนาบ

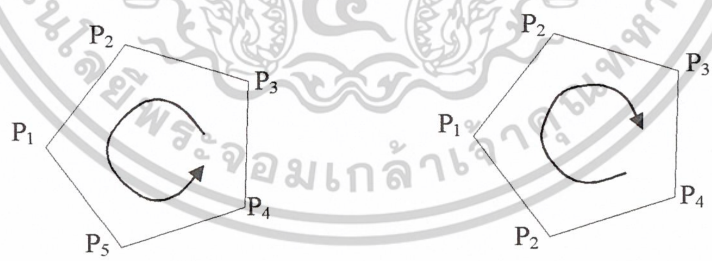
เวคเตอร์ตั้งฉากของระนาบเป็นสิ่งที่บ่งบอกทิศทางของระนาบได้ด้วย พื้นผิวของระนาบทุกระนาบมีอยู่ 2 ด้าน คือ ผิวด้านนอกและผิวด้านใน เวคเตอร์ตั้งฉากของระนาบจะพุ่งเข้าสู่ระนาบที่ผิวด้านในและพุ่งออกจากระนาบที่ผิวด้านนอก พื้นผิวของระนาบที่เห็นในรูปต่างๆ ในระบบเป็นจุดที่อยู่ฝั่ง “ด้านใน” หรือเป็นจุดที่อยู่ฝั่ง “ด้านนอก” ของระนาบ จุด (x, y ,z) ใดๆ ที่อยู่ด้านนอกของระนาบ จะทำให้สมการ

$$Ax + By + Cz + D > 0 \quad \text{เป็นจริง}$$

และในทำนองเดียวกัน จุด (x, y, z) ใดๆ ที่อยู่ด้านในของระนาบ จะทำให้สมการ

$$Ax + By + Cz + D < 0 \quad \text{เป็นจริง}$$

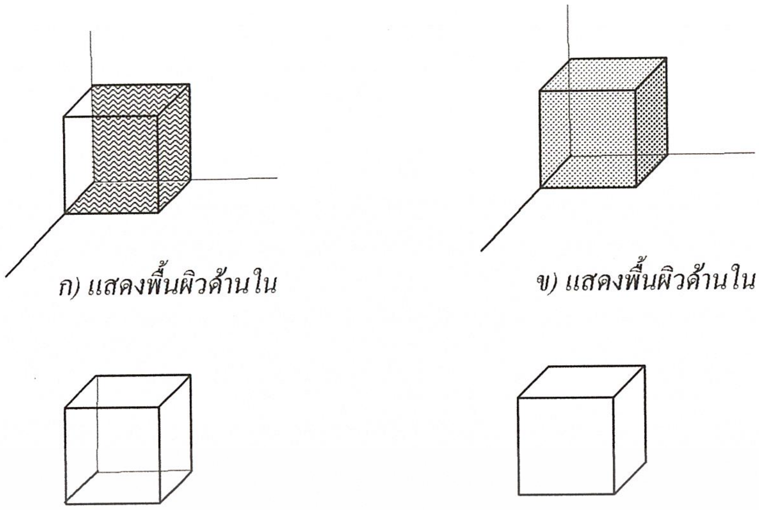
สำหรับรูปหลายเหลี่ยมในระบบ 3 มิติ ต้องวางตัวอยู่บนระนาบใดระนาบหนึ่ง รูปหลายเหลี่ยมจึงมีผิวด้านนอกและผิวด้านในด้วยเช่นกัน การกำหนดรูปหลายเหลี่ยมทำได้โดยกำหนดจุดยอดทุกจุดของรูปหลายเหลี่ยมเรียงตามลำดับและลากเส้นจากจุดแรกไปจนถึงจุดสุดท้าย ถ้าเรามองจากจุดนอกระนาบของรูปหลายเหลี่ยมแล้วเห็นจุดยอดต่างๆ ของรูปหลายเหลี่ยม เรียงกันไปในทิศทางทวนเข็มนาฬิกา เราจะถือว่า ผิวของรูปหลายเหลี่ยมที่เห็นนั้นคือผิวด้านนอก และในทางตรงกันข้าม ถ้าจุดยอดต่างๆ เรียงกันไปในทิศทางตามเข็มนาฬิกา เราจะมองเห็นผิวด้านในของรูปหลายเหลี่ยม ดังตัวอย่างในรูป



ก) ผิวด้านนอก *ข) ผิวด้านใน*

รูปที่ 2.28 การกำหนดผิวด้านนอก-ในของระนาบหรือรูปหลายเหลี่ยม

การที่เรากำหนดผิวด้านนอก-ในของระนาบหรือรูปหลายเหลี่ยม ก็เพื่อประโยชน์ในการสร้างภาพ โดยทั่วไปเรามักจะตัดภาพของส่วนที่เป็นพื้นผิวด้านใน (Back Face Removal) ทั้งนี้เพื่อไม่ให้ภาพปรากฏขึ้นมาสับสนเกินไป และได้ภาพที่มีลักษณะเหมือนกับความเป็นจริง ดังเช่นตัวอย่างในรูปเรามีรูปหลายเหลี่ยมหลายรูป ประกอบกันเป็นกล่องสี่เหลี่ยมกล่องหนึ่ง



ก) แสดงพื้นผิวด้านใน *ข) แสดงพื้นผิวด้านใน*

ก) แสดงพื้นทั้งผิวด้านในและด้านนอก *ง) แสดงเฉพาะผิวด้านนอก*

รูปที่ 2.29 แสดงผิวด้านนอก-ในของกล่องสี่เหลี่ยม

ในรูป ก) พื้นผิวที่ถูกแรเงาคือพื้นผิวด้านใน

ในรูป ข) พื้นผิวที่ถูกแรเงาคือพื้นผิวด้านนอก

ในรูป ค) เป็นการวาดพื้นผิวทั้งด้านนอกและด้านใน

ถ้าเราตัดภาพของรูปหลายเหลี่ยมที่หันผิวด้านในออก ภาพที่ได้จะเป็นภาพในรูป ง) ซึ่งจะเป็นภาพที่เราเห็นในความเป็นจริง

2.4.4 การแปลงในระบบ 3 มิติ

การแปลงในระบบ 3 มิติ มีความคล้ายคลึงกับการแปลงในระบบ 2 มิติ เรายังคงใช้เมตริกซ์ช่วยในการแปลง แต่ต้องเพิ่มขนาดของเมตริกซ์ไว้สำหรับโคออร์ดิเนต z ดังนั้นจุดในระบบ 3 มิติ ต้องแทนด้วยเมตริกซ์ขนาด 1 x 3 จุด (x, y, z) จะถูกแทนด้วย [x y z]

2.4.4.1 การแปลงแบบสเกล (Scaling)

ในระบบภาพ 2 มิติ เรามีเมตริกซ์การแปลงแบบสเกล ขนาด 2 x 2 มีรูปแบบเป็น

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

และถ้าเป็นโฮโมจีเนียสโคออร์ดิเนตเมตริกซ์ มีรูปแบบเป็น

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

โดยที่ S_x และ S_y คือ สเกลแฟคเตอร์สำหรับโคออร์ดิเนต X และ Y ตามลำดับ ในระบบภาพ 3 มิติ เมตริกซ์การแปลงแบบสเกลจะกลายเป็นเมตริกซ์ขนาด 3 x 3 โดยมีรูปแบบเป็น

$$S = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix}$$

และถ้าเป็นโฮโมจีเนียสโคออร์ดิเนตเมตริกซ์ จะมีรูปแบบเป็นเมตริกซ์ขนาด 4 x 4 คือ

$$S = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

โดยที่ S_x S_y และ S_z คือ สเกลแฟคเตอร์สำหรับโคออร์ดิเนต X Y และ Z ตามลำดับ

การแปลงจุดต่างๆทำได้โดยนำเมตริกซ์การแปลงไปคูณกับจุดนั้นเหมือนกับการแปลงในระบบ 2 มิติ

$$P_2 = \begin{bmatrix} X_2 & Y_2 & Z_2 & W \end{bmatrix}$$

$$= P_1 T$$

$$= \begin{bmatrix} X_2 & Y_2 & Z_2 & W \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_xX_1 & S_yY_1 & S_zZ_1 & W \end{bmatrix}$$

ดังนั้น

$$X_2 = S_xX_1$$

$$Y_2 = S_yY_1$$

$$Z_2 = S_zZ_1$$

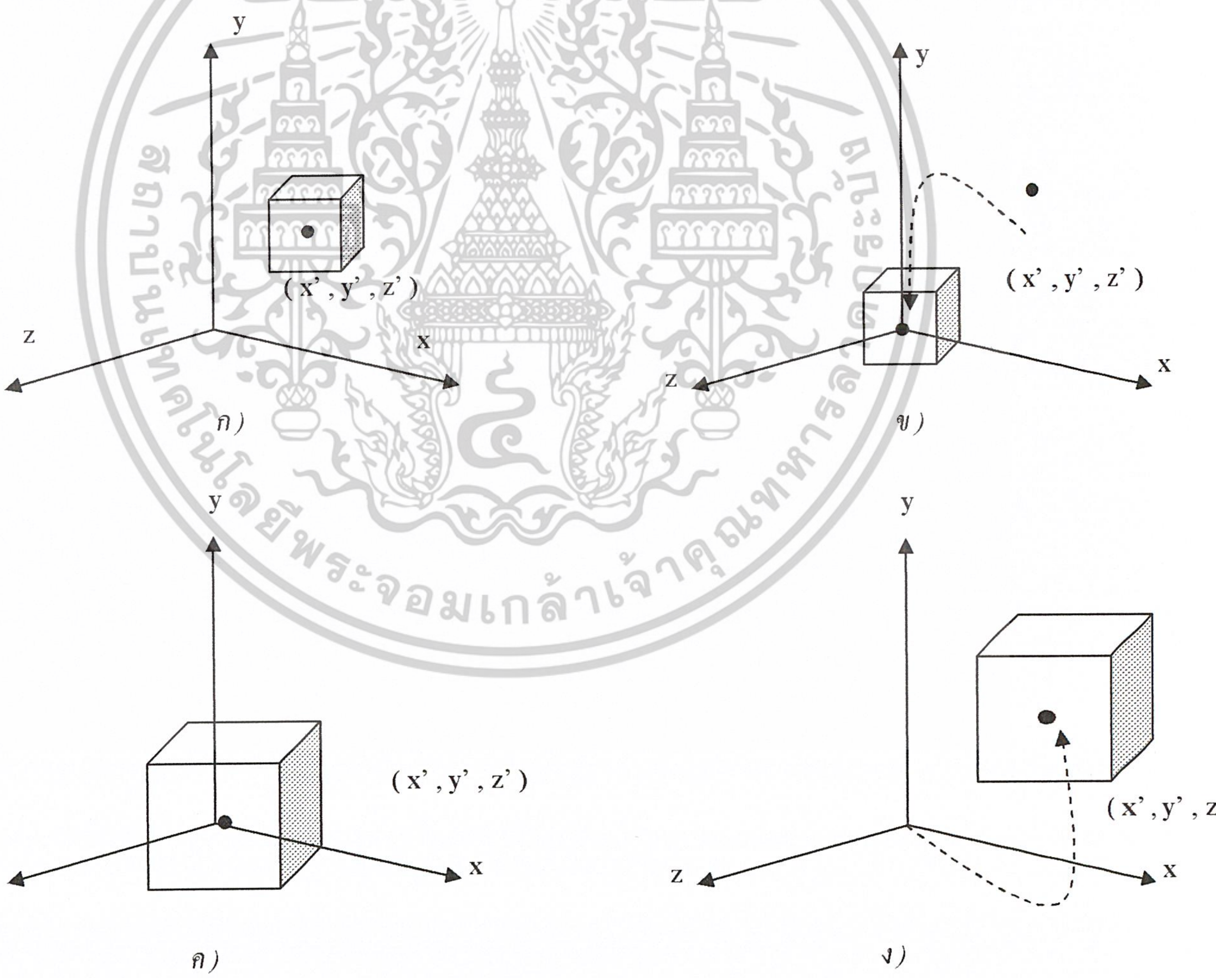
2.4.4.2 การแปลงแบบย้าย(Translation)

เมตริกซ์การแปลง คือ

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_Y & T_Z & 1 \end{bmatrix}$$

เมื่อ T_x T_Y T_Z คือ ระยะที่ต้องเลื่อนไปตามโคออร์ดิเนต X Y และ Z ตามลำดับ

ในทำนองเดียวกับระบบ 2 มิติ การสเกลวัตถุใดๆ ที่ไม่ได้อยู่ที่จุดกำเนิดต้องย้ายในตำแหน่งเดิม ดังตัวอย่างในรูป



รูปที่ 2.30 ขั้นตอนการย้ายวัตถุที่ไม่ได้อยู่ที่จุดกำเนิด

ดังนั้นเมตริกซ์การแปลง คือ

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -X & -Y & -Z & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ X & Y & Z & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ (1 - S_x)x & (1 - S_y)y & (1 - S_z)z & 1 \end{bmatrix}$$

เมื่อ S_x S_y S_z คือ สเกลแฟคเตอร์สำหรับโคออร์ดิเนต X Y และ Z ตามลำดับ และ (x , y , z) คือจุดเริ่มต้นของวัตถุ

2.4.4.3 การแปลงแบบหมุน (Rotation)

ในระบบ 2 มิติ การหมุนรอบจุดกำเนิด (0 , 0) มีเมตริกซ์การแปลง คือ

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

แต่ในระบบ 3 มิติ การแปลงแบบหมุนไม่ได้เป็นการหมุนรอบจุดๆหนึ่ง แต่เป็นการหมุนรอบแกนๆหนึ่ง ถ้าเราเปลี่ยนเมตริกซ์การแปลงแบบหมุนในสมการข้างต้นให้ใช้ได้ในระบบ 3 มิติ เมตริกซ์ที่ได้จะเป็นเมตริกซ์การแปลงแบบหมุนตามแนวแกน Z เพราะการหมุนนี้ค่าของโคออร์ดิเนต Z คงที่ไม่เปลี่ยนแปลง เราเขียนเมตริกซ์นี้ให้อยู่ในรูปของโฮโมจีเนียสโคออร์ดิเนตเมตริกซ์ คือ

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

การหมุนเช่นนี้ เราอาจคิดว่าเป็นการหมุนวัตถุไปรอบๆแกนที่หยุดนิ่ง หรืออาจคิดว่าแกนหมุนเคลื่อนที่ไปรอบๆวัตถุที่อยู่นิ่งก็ได้ ความแตกต่างของการแปลความหมายของการหมุนก็คือทิศทางของการหมุนเท่านั้น การหมุนวัตถุไปรอบๆแกนที่อยู่กับที่ในทิศทางทวนเข็มนาฬิกา ก็คือการหมุนแกนไปรอบๆวัตถุที่อยู่กับที่ในทิศตามเข็มนาฬิกานั่นเอง

ถ้าหมุนจุด $P_1(X_1, Y_1, Z_1)$ รอบแกน Z เป็นมุม θ จะได้จุด $P_2(X_2, Y_2, Z_2)$ คือ

$$X_2 = X_1 \cos(\theta) + Y_1 \sin(\theta)$$

$$Y_2 = -X_1 \sin(\theta) + Y_1 \cos(\theta)$$

$$Z_2 = Z_1$$

สำหรับการหมุนรอบแกน X และ Y ก็มีสมการที่คล้ายกับสมการการหมุนของแนวแกน Z โดยเมตริกซ์การแปลงแบบหมุนรอบแกน X คือ

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

ดังนั้นจุด $P_2(X_2, Y_2, Z_2)$ คือ

$$X_2 = X_1$$

$$Y_2 = Y_1 \cos(\theta) + Z_1 \sin(\theta)$$

$$Z_2 = -Y_1 \sin(\theta) + Z_1 \cos(\theta)$$

เมตริกซ์การแปลงแบบหมุนรอบแกน Y คือ

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

ดังนั้นจุด $P_2(X_2, Y_2, Z_2)$ คือ

$$X_2 = Z_1 \sin(\theta) - X_1 \cos(\theta)$$

$$Y_2 = Y_1$$

$$Z_2 = Z_1 \cos(\theta) + X_1 \sin(\theta)$$

2.4.4.3.1 การหมุนรอบแกนใดๆ

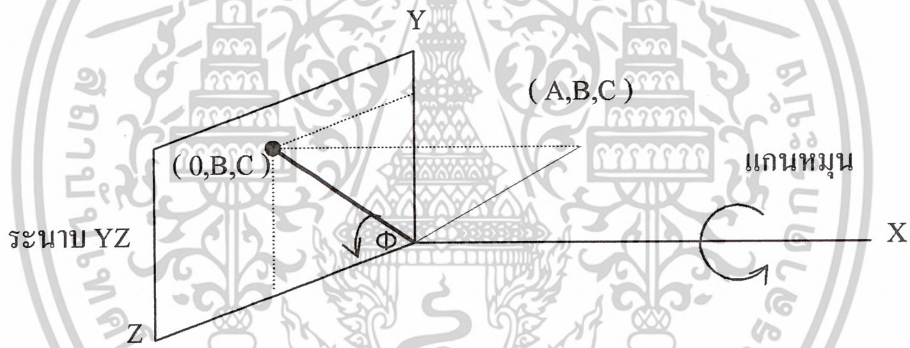
การแปลงแบบหมุนที่กล่าวไว้ในหัวข้อที่แล้ว เป็นการหมุนรอบแกนโคออร์ดิเนต ทั้ง 3 เท่านั้น อย่างไรก็ตามเราสามารถหมุนวัตถุไปรอบแกนอื่นๆในระบบนอกเหนือจากแกนทั้ง 3 ได้ด้วยเช่นกัน โดยปกติแล้วเส้นตรงในระบบสามารถทำหน้าที่เป็นแกนหมุนได้ ซึ่งต่อไปเราจะหาเมตริกซ์การแปลงแบบหมุนรอบแกน(เส้นตรง)ใดๆ ในระบบ โดยเริ่มจากการย้ายจุดกำเนิดไปอยู่บนเส้นตรงหรือแกนหมุนนั้น จากนั้นเราจะหมุนไปรอบแกน x และแกน y เพื่อจัดให้แกนหมุนนั้นมีทิศทางเดียวกับแกน z ดังนั้นการหมุนรอบแกน z จึงเป็นการหมุนรอบแกนหมุนนี้ด้วย และท้ายที่สุด ทำการแปลงกลับ เพื่อให้แกนหมุนกลับไปอยู่ในแนวเดิม และย้ายจุดกำเนิดกลับไปอยู่ในตำแหน่งเดิม การกำหนดเส้นตรงในระบบโคออร์ดิเนต สามารถกำหนดได้ด้วยจุดหนึ่งจุดบนเส้นตรงนั้นกับเวคเตอร์ที่มี

$$T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ x_1 & y_1 & z_1 & 1 \end{bmatrix}$$

หลังจากการแปลงด้วยเมตริกซ์ T แล้ว จุด P จะไปอยู่ที่จุดกำเนิดหลังจากการแปลงแบบหมุนเสร็จสิ้น เราต้องอินเวอร์สเมตริกซ์ T เพื่อแปลงจุด P กลับไปยังตำแหน่งเดิมก่อนการย้าย เมตริกซ์นี้คือ

$$T^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -x_1 & -y_1 & -z_1 & 1 \end{bmatrix}$$

ลำดับต่อไปคือการหมุนเส้นตรงรอบแกน X จนกระทั่งเส้นตรงอยู่บนระนาบ XY เพื่อให้เข้าใจการหามุมของการหมุน เราจะฉายเงาของเส้นตรงให้เกิดขึ้นบนระนาบ YZ สมมุติว่าเรามีส่วนของเส้นตรงนี้ซึ่งอยู่ระหว่างจุด (0,0,0) และจุด (A,B,C) เราจะได้เงาของส่วนของเส้นตรงนี้ อยู่บนระนาบ YZ เป็นเส้นตรงที่อยู่ระหว่างจุด (0,0,0) และ (0,B,C)



รูปที่ 2.31 การหมุนเส้นตรงรอบแกน X ให้มาอยู่บนระนาบ XY

การหมุนเส้นตรงนี้ให้มาอยู่บนระนาบ XZ เงาของเส้นตรงจะเคลื่อนมาทับแกน Z พอดี นั่นคือมุมที่ใช้ในการหมุน คือ Φ ดังแสดงในรูปข้างต้น ถ้า V คือความยาวของเงาบนระนาบ YZ ดังนั้น

$$V = (B^2 + C^2)^{1/2} \qquad \text{......(a)}$$

และ $$\sin(\theta) = B/V \qquad \text{......(b)}$$

$$\cos(\theta) = C/V \qquad \text{......(c)}$$

เมตริกซ์การแปลงแบบหมุนรอบแกน X คือ

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

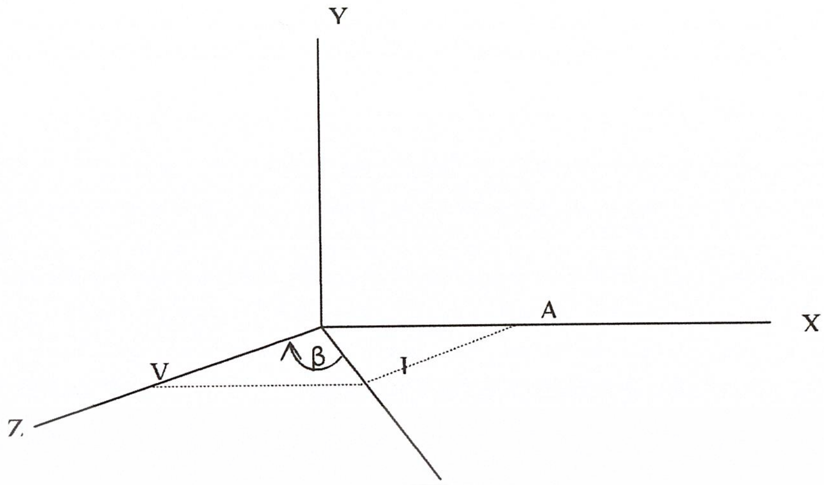
เมื่อแทนค่าในสมการ (b) และ (c) ลงในสมการของ R_x จะได้

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & B/V & 0 \\ 0 & -B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

สำหรับการแปลงกลับของเมตริกซ์ในสมการนี้คือการหมุนด้วยมุมเท่าเดิมแต่ในทิศทางตรงกันข้าม ทำให้ได้ อินเวอร์สเมตริกซ์ คือ

$$R_x^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & C/V & -B/V & 0 \\ 0 & B/V & C/V & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

หลังจากที่หมุนเส้นตรงไปบนระนาบ XZ แล้ว เราจะหมุนเส้นตรงนี้รอบแกน Y จนกระทั่งเส้นตรงนี้ทับ กับแนวเส้นพอดีดังแสดงในรูป



รูปที่ 2.32 การหมุนเส้นตรงรอบแกน Y

ซึ่งมุมที่ใช้ในการหมุนคือ β ถ้าความยาวของส่วนของเส้นตรงนี้ คือ L ดังนั้น

$$L = (A^2 + V^2)^{1/2}$$

$$= (A^2 + B^2 + C^2)^{1/2} \quad \text{...............(d)}$$

และ $$\sin(\beta) = A / L \quad \text{...............(e)}$$

$$\cos(\beta) = A / L \quad \text{...............(f)}$$

เมตริกซ์การแปลงแบบหมุนรอบแกน Y คือ

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

$$= \begin{bmatrix} V/L & 0 & -A/L & 0 \\ 0 & 1 & 0 & 0 \\ A/L & 0 & V/L & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ในทำนองเดียวกันการแปลงกลับต้องหมุนรอบแกน Y ในทิศทางตรงกันข้ามเป็นมุม β ดังนั้นอินเวอร์สเมตริกซ์ คือ

$$R_y^{-1} = \begin{bmatrix} V/L & 0 & A/L & 0 \\ 0 & 1 & 0 & 0 \\ -A/L & 0 & V/L & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

ต่อไปเราจะหมุนวัตถุไปรอบแกน Z เป็นมุม θ โดยที่มุม θ นี้คือมุมที่เราต้องการหมุนวัตถุรอบๆ แกนหรือเส้นตรงที่เรากำลังกล่าวถึงอยู่ดังนั้นเมตริกซ์การแปลงแบบหมุนรอบแกน Z คือ

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

ดังนั้นเมตริกซ์การแปลงแบบหมุนรอบแกนใดๆ R_0 คือ

$$\boxed{R_0 = T\, R_x\, R_y\, R_z\, R_y^{-1}\, R_x^{-1}\, T^{-1}}$$

บทที่ 3
การออกแบบระบบและขั้นตอนการดำเนินงาน

3.1 กระบวนการและการพัฒนาระบบ

3.1.1 การศึกษาค้นคว้า

1. ศึกษาหลักการทางคณิตศาสตร์และหลักการทางคอมพิวเตอร์กราฟิกดังต่อไปนี้
   - ทฤษฎีแฟรคทอล โดยเน้นที่ ระบบสมการที่มีการทำซ้ำ หรือ IFS (Iterated Function System)
   - การแปลงภาพเรขาคณิต 2 มิติ และ 3 มิติ
2. ศึกษาโปรแกรม Visual C# .NET เพื่อนำมาใช้ในการเขียนโปรแกรม

3.1.2 ความรู้เบื้องต้นเกี่ยวกับ Visual C#.NET

C# เป็นภาษาที่สืบทอดสายตระกูลมาจาก C และ C++ เช่นเดียวกับ JAVA เป็นภาษาเชิงวัตถุอย่างแท้จริง ทุกสิ่งใน C# เป็น ออปเจ็กต์ ภาษาC# มีความง่ายและทันสมัยกว่า C++ สิ่งที่เป็นความซับซ้อนใน C++ ถูกกำจัดออกไปหลายอย่างและมีการเพิ่มฟีเจอร์และคุณสมบัติอื่นลงไปเช่น เพิ่มคลาส ไลบรารีพื้นฐาน ทำให้จำนวนโค้ดใน C# มีจำนวนลดลง ขนาดของโปรแกรมจึงเล็กลงแต่มีประสิทธิภาพสูงขึ้น และเนื่องจากมีโครงสร้างคล้ายกับ C และ C++ ทำให้ไม่ต้องเริ่มต้นเรียนรู้ใหม่ทั้งหมด

3.1.3 ขั้นตอนการดำเนินงาน

1. ศึกษาหลักการทางคณิตศาสตร์ และความรู้ทางคอมพิวเตอร์กราฟิกที่ต้องนำมาใช้ในการวางแผน ออกแบบ และพัฒนาระบบ
2. ศึกษาระบบ และวิธีการใช้งานแอพพลิเคชั่นดิวิลล็อปเม็นท์ทูลเพื่อให้เหมาะสมและมีความสามารถที่ใช้พัฒนาระบบได้ โดยเลือก Visual C# .NET
3. ศึกษาซอฟแวร์ที่เกี่ยวกับคอมพิวเตอร์กราฟิกที่มีอยู่ในปัจจุบัน เพื่อศึกษา และนำมาประยุกต์ใช้ในการพัฒนาโปรแกรม
4. ทำการพัฒนาระบบงานจริง
   - ออกแบบแผนภูมิการไหลของระบบงาน
   - กำหนดอินพุท , เอาท์พุทของระบบ
   - ออกแบบส่วนติดต่อกับผู้ใช้

- ดำเนินงานพัฒนาโปรแกรม
- ให้ผู้ใช้งานทดลองใช้เพื่อหาข้อผิดพลาดและข้อเสนอแนะแล้วนำมาปรับปรุงโปรแกรม ให้ได้ตรงตามความต้องการ ใช้งานง่ายมากที่สุด
- จัดทำเอกสารประกอบการใช้งาน

3.2 การออกแบบระบบ

ประกอบด้วย 3 ส่วนต่อไปนี้

1.ส่วนการนำข้อมูลเข้า

เป็นข้อมูลที่มาจากผู้ใช้โดยที่ผู้ใช้ใส่ข้อมูลต่างๆที่ต้องใช้ในการสร้างรูปแฟรคทอล ตามที่ต้องการ เช่น รูปเริ่มต้นที่ต้องการ สีของรูป จำนวนรูปที่ต้องใช้ในการจัดรูปแบบ จำนวนครั้งในการทำซ้ำ เป็นต้น

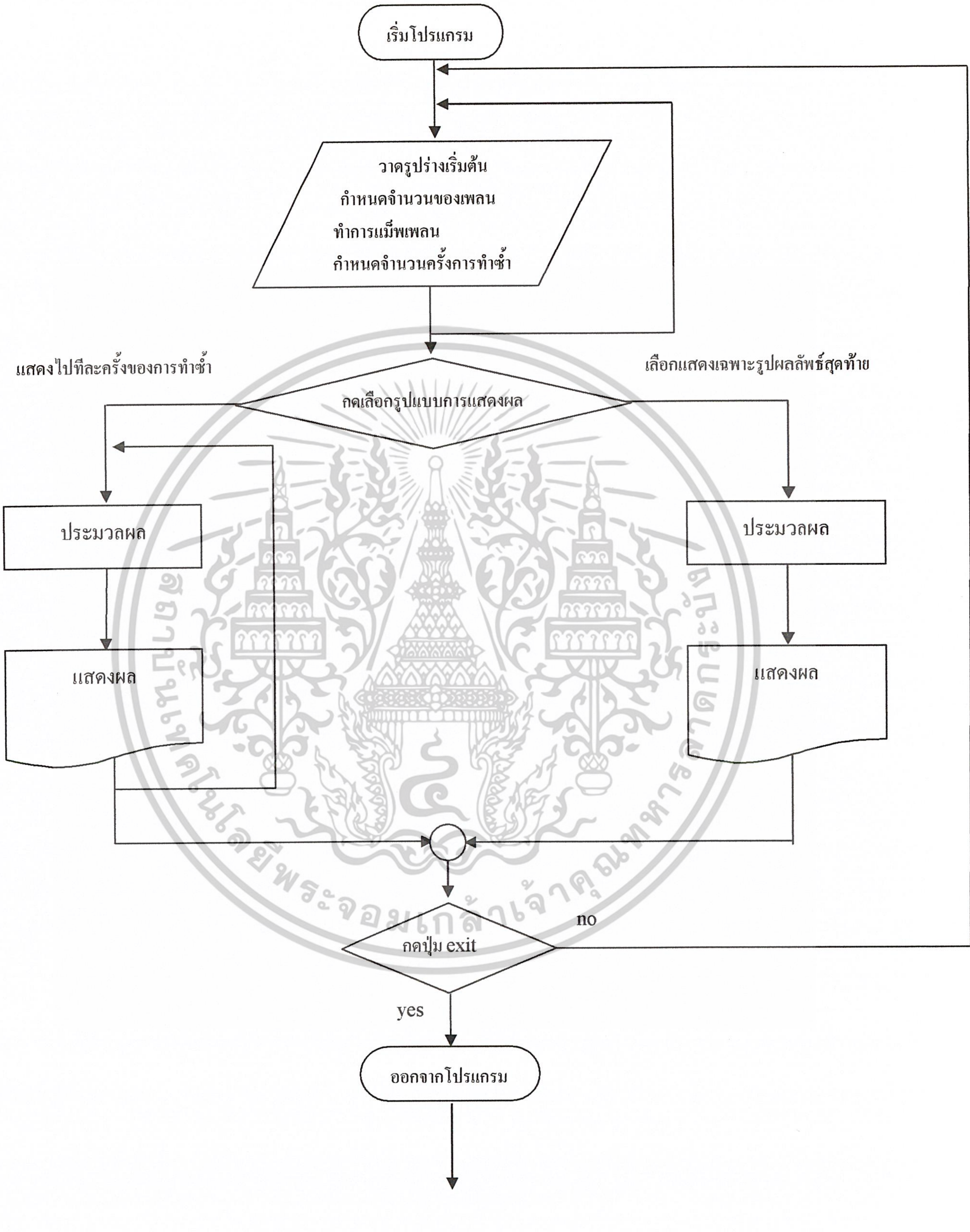
2. ส่วนการวิเคราะห์

นำข้อมูลที่รับเข้ามาทำการประมวลผล คำนวณข้อมูลภาพ

3. ส่วนการแสดงผล

นำค่าจากข้อ 2 มาแสดงออกทางจอภาพ

3.3 ขั้นตอนการทำงานของโปรแกรม



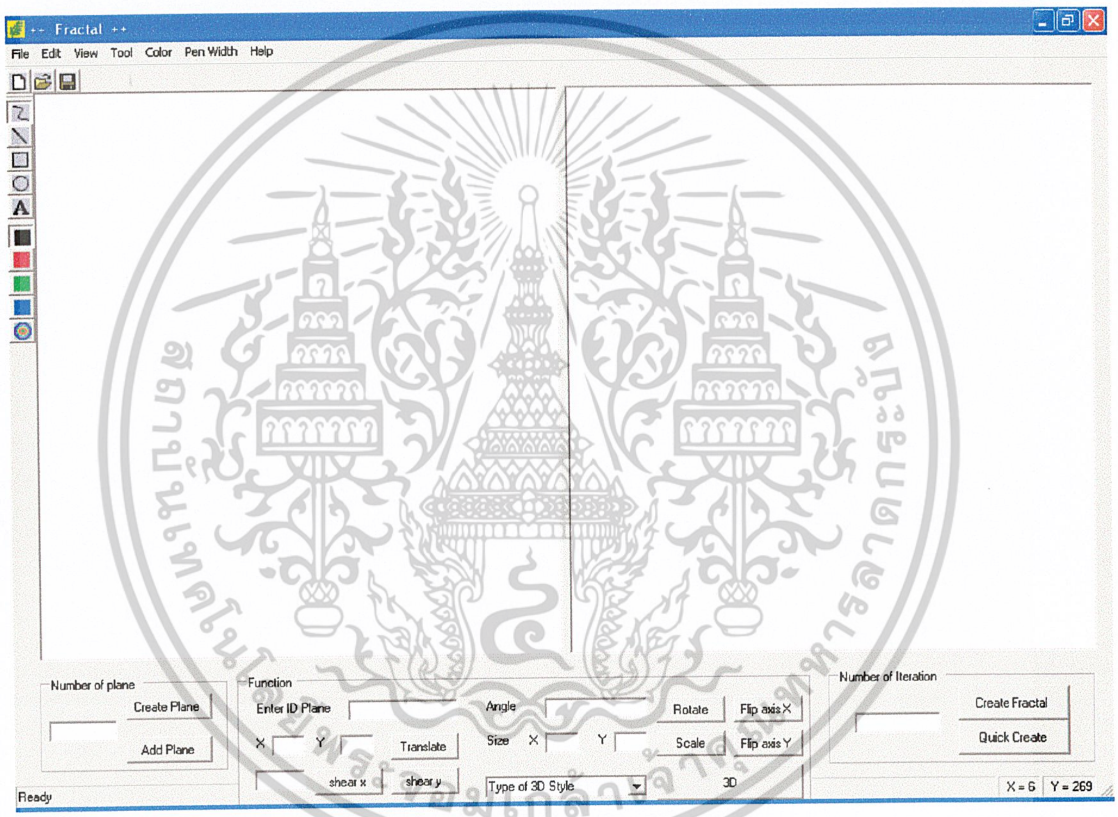
บทที่ 4

การประเมินผล

ผลลัพธ์ที่ได้จากการพัฒนาโปรแกรมการจำลองรูปทรงแฟรคทอลโดยทฤษฎีระบบสมการที่มีการทำซ้ำสามารถประเมินผลได้ดังนี้

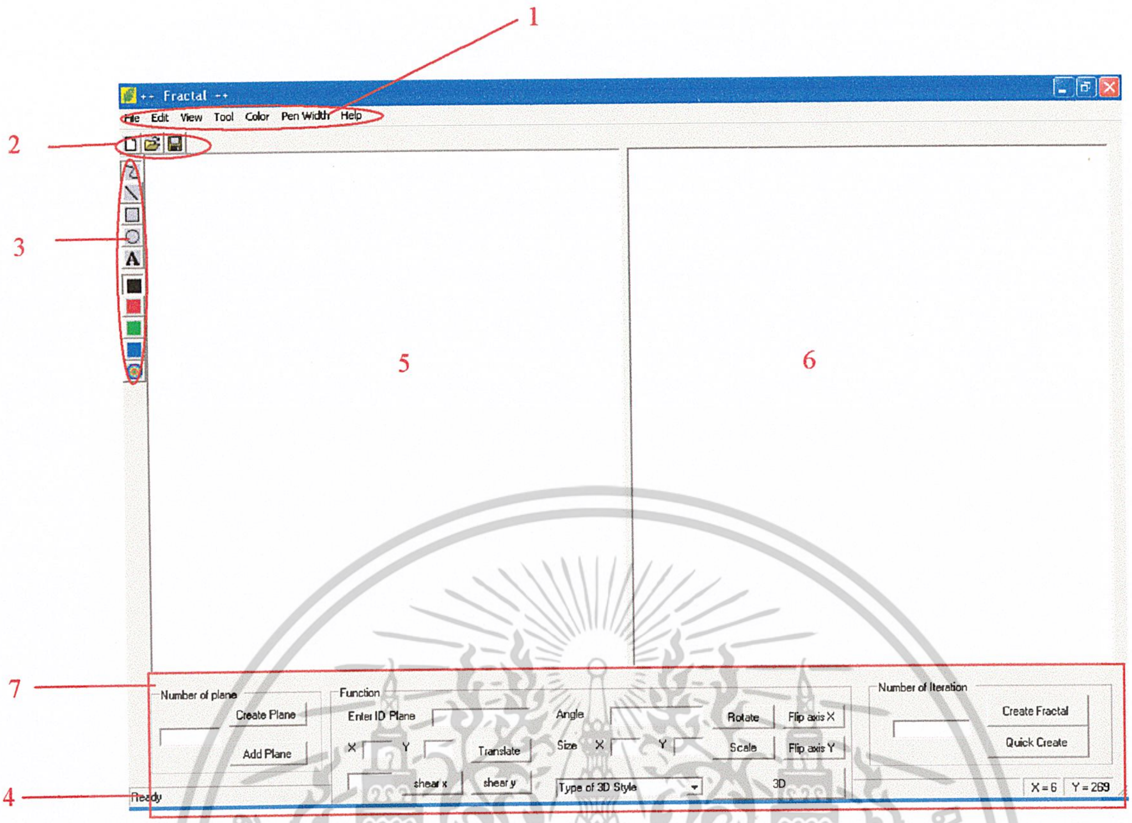
4.1 ลักษณะของโปรแกรม

เมื่อเริ่มต้นรันโปรแกรม จะปรากฏหน้าต่างใช้งานดังภาพ



รูปที่ 4.1 ลักษณะหน้าจอของโปรแกรม

ส่วนต่างๆ ของโปรแกรมแบ่งออกได้เป็น 6 ส่วนหลัก ๆ ได้แก่

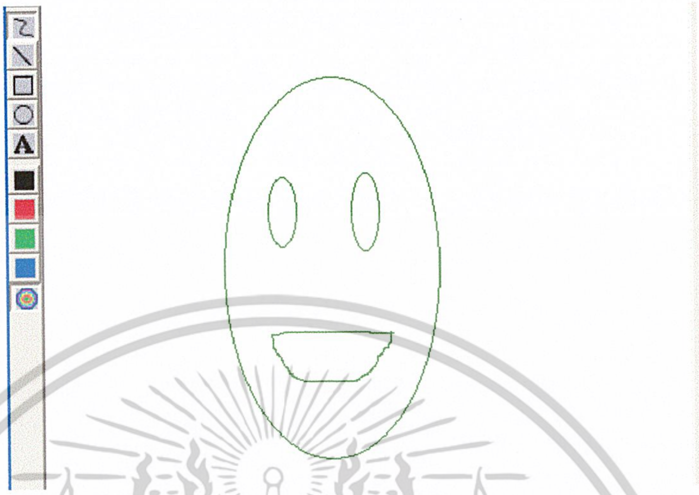


รูปที่ 4.2 แสดงองค์ประกอบของโปรแกรม

1. เมนูบาร์ (Menu Bar)
2. ทูลบาร์ (Tool Bar)
3. เพนท์บาร์ (Paint Bar)
4. สเตตัสบาร์ (Status Bar)
5. หน้าต่างที่ใช้ในการสร้างภาพเริ่มต้นในการทำแฟรคทอล
6. หน้าต่างที่ใช้ในการจัดรูปแบบในการทำซ้ำของแฟรคทอล
7. ฟังก์ชั่นที่ใช้ในการสร้างแฟรคทอล

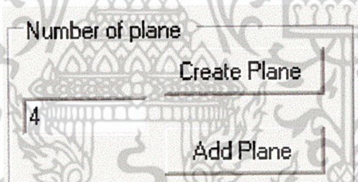
4.2 การทำงานของโปรแกรม

4.2.1 ผู้ใช้จะต้องทำการสร้างรูปภาพเริ่มต้น ที่จะใช้ในการสร้างภาพแฟรคทอลขึ้นมาใน ส่วนของหน้าต่างฝั่งซ้าย (5) โดยสร้างจากเครื่องมือที่เพนท์บาร์



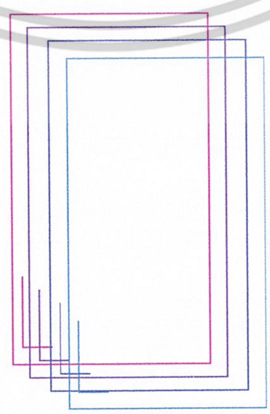
รูปที่ 4.3 แสดงการสร้างภาพเริ่มต้นในหน้าต่างฝั่งซ้าย

4.2.2 ใส่จำนวนเพลน (Plane) ที่จะใช้ในการจัดรูปแบบในการทำซ้ำของแฟรคทอล



รูปที่ 4.4 จำนวนเพลนที่ต้องการสร้าง

จากนั้นเมื่อกด Create Plane จะได้เพลนขึ้นมาตามจำนวนที่กำหนดที่หน้าต่างทางด้านขวา ของโปรแกรม(6) โดยเพลนที่สร้างขึ้นมานั้นจะมีขนาดเท่ากับขอบเขตของภาพเริ่มต้น



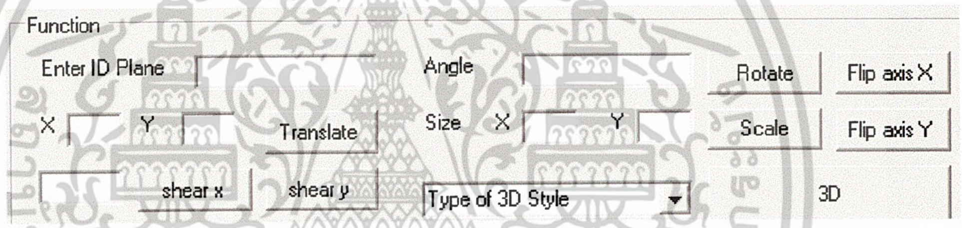
รูปที่ 4.5 แสดงเพลนที่ถูกสร้างขึ้นที่หน้าต่างทางด้านขวาของโปรแกรม

4.2.3 ผู้ใช้จะทำการเปลี่ยนแปลงหรือจัดการกับเพลนโดยการกำหนดค่าลงไปในส่วนของฟังก์ชั่น ซึ่งประกอบด้วย

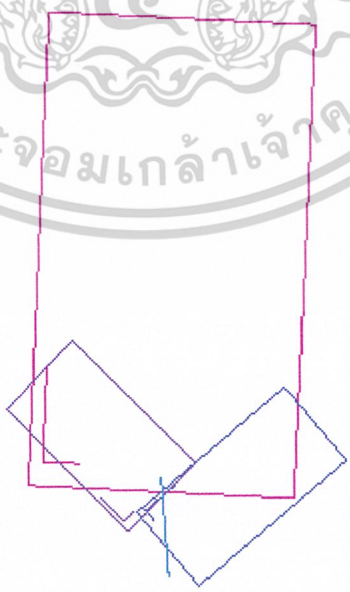
- การเปลี่ยนตำแหน่ง (Translation)
- การหมุน (Rotation)
- การย่อ-ขยาย (Scaling)
- การสะท้อนภาพ (Reflection)
- การบิดภาพ (Shearing)

โดยผู้ใช้จะต้องกำหนดหมายเลขของเพลนที่ต้องการทำการเปลี่ยนแปลง ซึ่งหมายเลขของเพลนนั้นจะมีค่าเรียงตามจำนวนของเพลนที่สร้าง ในตัวอย่างได้สร้าง เพลนขึ้นทั้งหมดเป็นจำนวน 4 เพลน ดังนั้นหมายเลขของเพลนจะเป็น 1, 2, 3, และ 4 ตามลำดับ

ในกรณีของการเปลี่ยนตำแหน่งผู้ใช้จะสามารถทำได้ 2 วิธี คือ การใส่ค่าลงไปในส่วน Function ของการเปลี่ยนตำแหน่ง หรือใช้เมาท์คลิ้กที่เพลนแล้วลากวางตามตำแหน่งที่ต้องการ

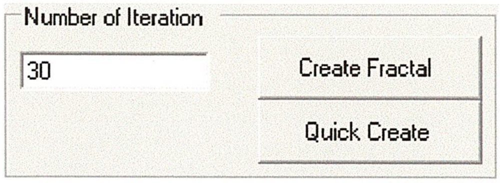


รูปที่ 4.6 แสดงการใส่ค่าลงไปใน Function ต่างๆ



รูปที่ 4.7 แสดงรูปแบบของเพลนเมื่อได้ทำการจัดรูปแบบตามต้องการ

4.2.4 หลังจากจัดรูปแบบของเพลนแล้ว ผู้ใช้จะต้องทำการกำหนดจำนวนครั้งในการทำซ้ำ



รูปที่ 4.8 การใส่จำนวนครั้งในการทำซ้ำ

ในการจะประมวลผลสร้างรูปแฟรคทอลผู้ใช้มีทางเลือกสองทางคือ

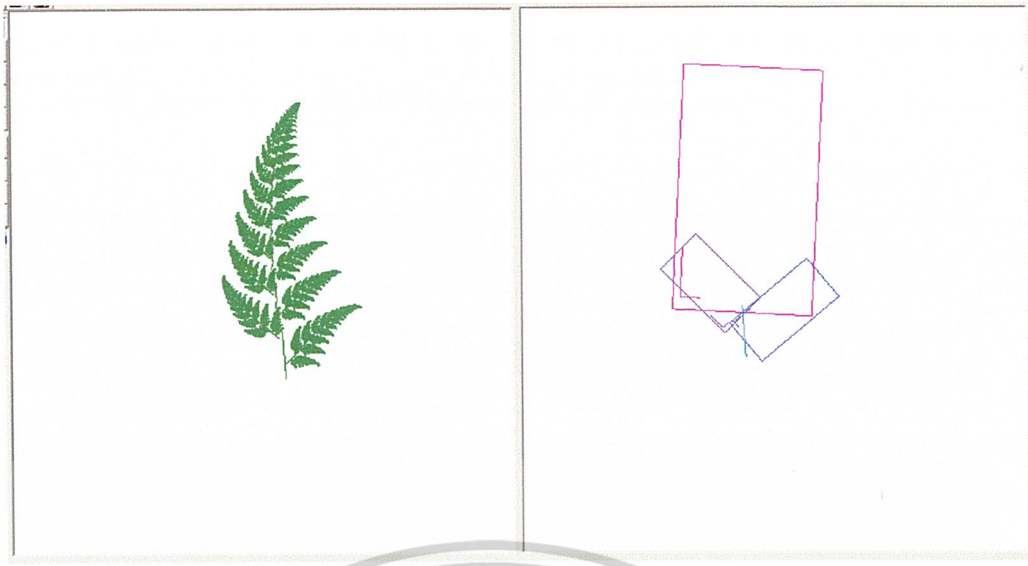
1. กดปุ่ม "Create Fractal" หน้าต่างทางด้านซ้ายมือจะค่อยๆ ปรากฏรูปแฟรคทอลในแต่ละรอบของการทำซ้ำจนครบตามจำนวนที่ได้ระบุไป

ครั้งที่ 1 *ครั้งที่ 2* *ครั้งที่ 3* *ครั้งที่ 10* *ครั้งที่ 30*

รูปที่ 4.9 แสดงรูปการทำซ้ำในแต่ละครั้ง

2. กดปุ่ม "Quick Create" หน้าต่างทางด้านซ้ายมือจะปรากฏรูปแฟรคทอลของการทำซ้ำในครั้งสุดท้ายออกมาให้เลย

รูปที่ 4.10 แสดงรูปการทำซ้ำในครั้งสุดท้าย

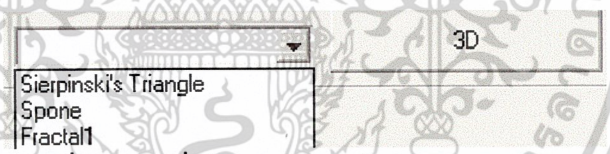


รูป ก) ภาพแฟรคทอลจะปรากฏที่หน้าต่างด้านซ้ายรูป ข) การจัดการกับเพลนจะกระทำที่หน้าต่างด้านขวา

รูปที่ 4.11แสดงภาพของหน้าต่างทั้งสองฝั่ง

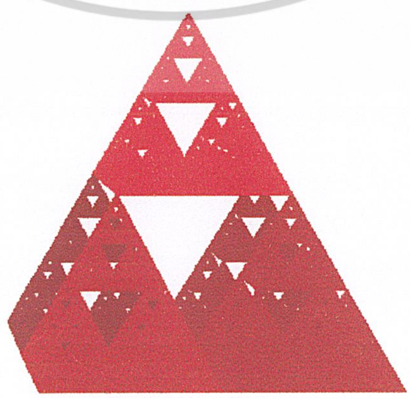
4.2.5 ผู้ใช้สามารถบันทึกภาพเริ่มต้นและบันทึกรูปแบบการทำซ้ำ (บันทึกรูปแบบการจัดการของเพลนที่ได้เปลี่ยนแปลงไว้) เพื่อสามารถนำรูปแบบการทำซ้ำมาเปลี่ยนแปลงได้ในภายหลัง

4.2.6 ในส่วนของรูปทรงแฟรคทอลแบบ 3 มิติ ผู้ใช้สามารถเลือกดูได้จากตัวเลือกด้านล่าง



รูปที่ 4.12 ฟังค์ชั่นในการเลือกดูแฟรคทอลแบบ 3 มิติ

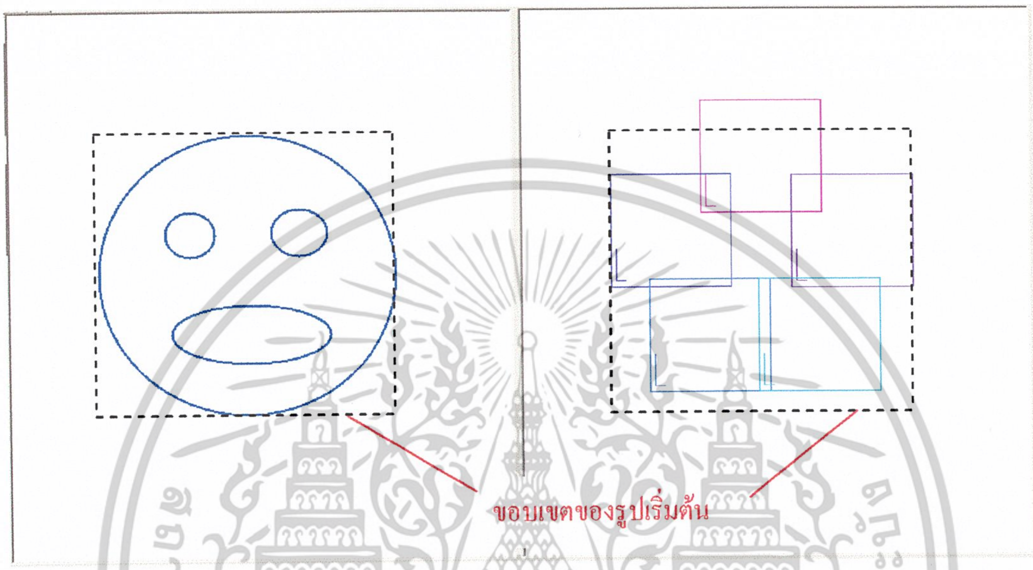
เมื่อกดปุ่ม 3D โปรแกรมจะแสดงรูปทรงแฟรคทอลแบบ 3 มิติ ตามที่เลือกขึ้นมาให้ดูที่หน้าต่างด้านซ้ายดังรูปที่ 4.13



รูปที่ 4.13 รูปสามเหลี่ยม Sierpinski แบบ 3 มิติ

4.3 ปัญหาที่พบในการทำปัญหาพิเศษ

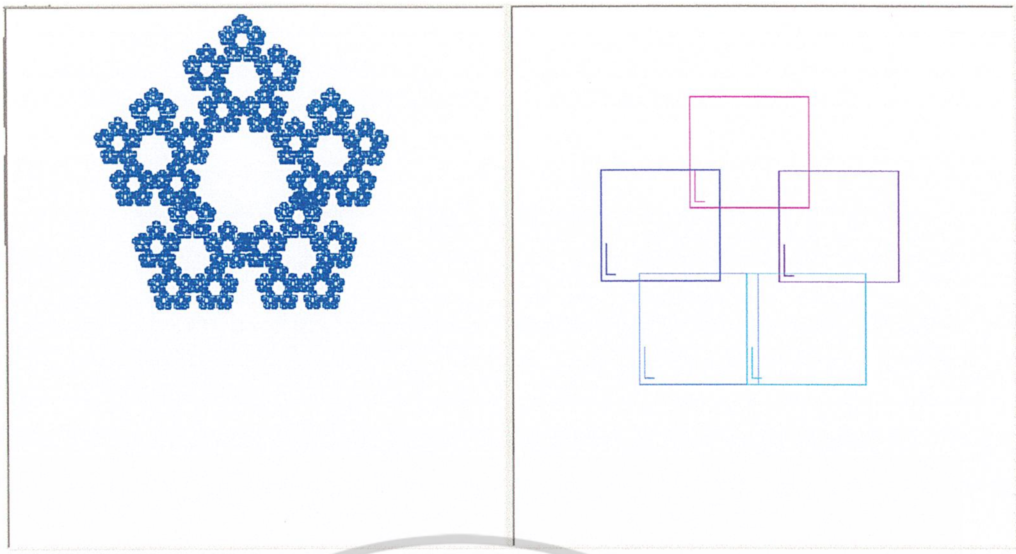
ในส่วนของการจัดรูปแบบเพลนเพื่อนำไปเป็นรูปแบบของการทำซ้ำ (หน้าต่างทางฝั่งขวามือ) หากมีการเคลื่อนย้ายเพลนออกนอกขอบเขตของรูปเริ่มต้น เมื่อมีการทำซ้ำจะทำให้ภาพที่ได้ออกมาเคลื่อนไปตามระยะที่เพลนเคลื่อน ในกรณีที่มีจำนวนของการทำซ้ำมากจะทำให้ภาพสุดท้ายที่ได้ออกมาอาจจะเลยขอบเขตของหน้าต่างที่ใช้ในการแสดงภาพแฟรคทอล ภาพที่ได้จึงออกมาไม่สมบูรณ์



รูปที่ 4.14 รูปแบบเพลนที่ถูกเคลื่อนย้ายเลยขอบเขตของรูปเริ่มต้น

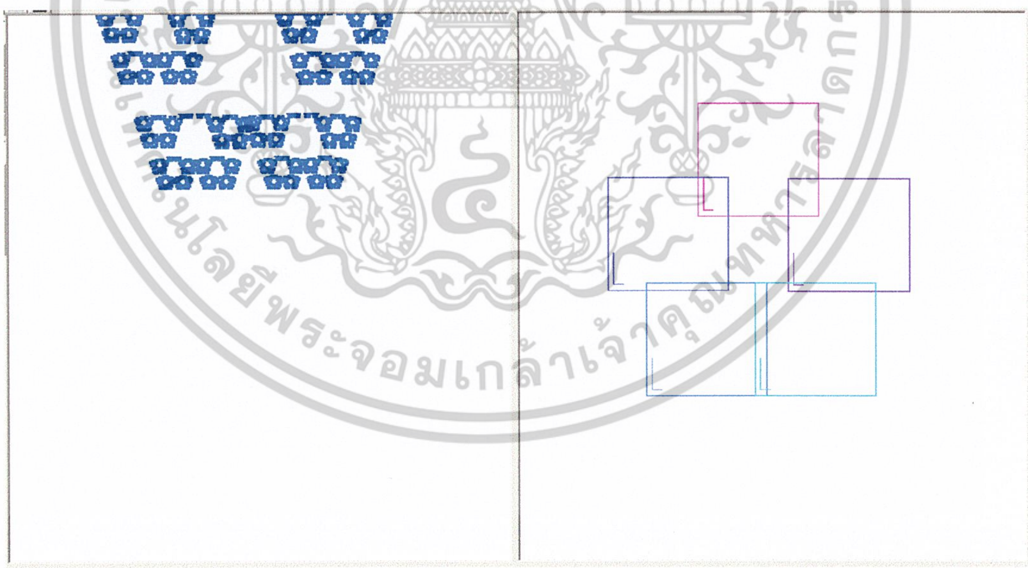
จากรูปที่ 4.12 จะเห็นได้ว่าเพลนได้ถูกจัดให้อยู่เลยขอบเขตของรูปเริ่มต้นไปทางด้านบน ดังนั้นเมื่อมีการทำซ้ำภาพแฟรคทอลที่ได้จะเลื่อนขึ้นตามไปด้วย เนื่องจากกระบวนการในการสร้างภาพแฟรคทอลจะต้องมีการอ้างอิงค่าจากตำแหน่งของเพลนจากหน้าต่างทางด้านขวามือ ดังนั้นภาพแฟรคทอลที่ได้ก็จะมีการเคลื่อนไปตามตำแหน่งเพลนหรือตำแหน่งรูปแบบในการทำซ้ำ

ในกรณีที่มีจำนวนการทำซ้ำน้อยปัญหาการตกขอบของหน้าต่างแสดงผลจะไม่เกิดขึ้น ตัวอย่างเช่นการทำซ้ำที่มีจำนวน 4 ครั้ง ภาพแฟรคทอลที่ได้จะออกมาสมบูรณ์ ดังที่แสดงในรูปที่ 4.15



รูปที่ 4.15 แสดงภาพแฟรคทอลที่มีจำนวนการทำซ้ำ 4 ครั้ง

แต่ในกรณีที่มีจำนวนการทำซ้ำค่อนข้างมากจะเกิดปัญหาการตกขอบของหน้าต่างแสดงผล ตัวอย่างเช่นการทำซ้ำที่มีจำนวนการทำซ้ำ 10 ครั้ง ภาพแฟรคทอลที่ได้ออกมาจะไม่สมบูรณ์ ดังที่แสดงในรูปที่ 4.16



รูปที่ 4.16 แสดงภาพแฟรคทอลที่มีจำนวนการทำซ้ำ 10 ครั้ง

บทที่ 5
สรุปวิจารณ์และแนวทางในการพัฒนาระบบ

5.1 ความสามารถของโปรแกรม

1. ผู้ใช้สามารถสร้างรูปแฟรคทอลในแบบของตนได้อย่างง่ายๆ โดยวาดรูปเริ่มต้นแล้วนำมาจัดรูปแบบด้วยฟังก์ชั่น การหมุน ย้ายที่ ย่อ-ขยาย สะท้อนและบิดภาพ

2. ผู้ใช้สามารถเซฟรูปเริ่มต้นหรือรูปแบบที่ได้ทำการจัดไว้แล้วได้เพื่อเก็บไว้ใช้ต่อ

3. ในส่วน 3 มิติ จะมีตัวอย่างของรูปทรงแฟรคทอล 3 มิติแบบต่างๆที่สร้างจากระบบสมการทีมีการทำซ้ำให้เลือกชม

5.2 ข้อจำกัดของโปรแกรม

1. โปรแกรมสามารถสร้างรูปแฟรคทอลด้วยวิธีระบบสมการที่มีการทำซ้ำเท่านั้น
2. ผู้ใช้จะต้องสร้างรูปเริ่มต้นขึ้นมาเองด้วยฟังก์ชั่นที่มีให้
3. จำนวนกรอบรูปสูงสุดที่สามารถนำไปจัดรูปแบบได้คือ 50 กรอบ
4. รูปทรงแฟรคทอลที่เป็น 3 มิติ ผู้ใช้ไม่สามารถสร้างขึ้นเองได้

5.3 แนวทางในการพัฒนาโปรแกรมต่อ

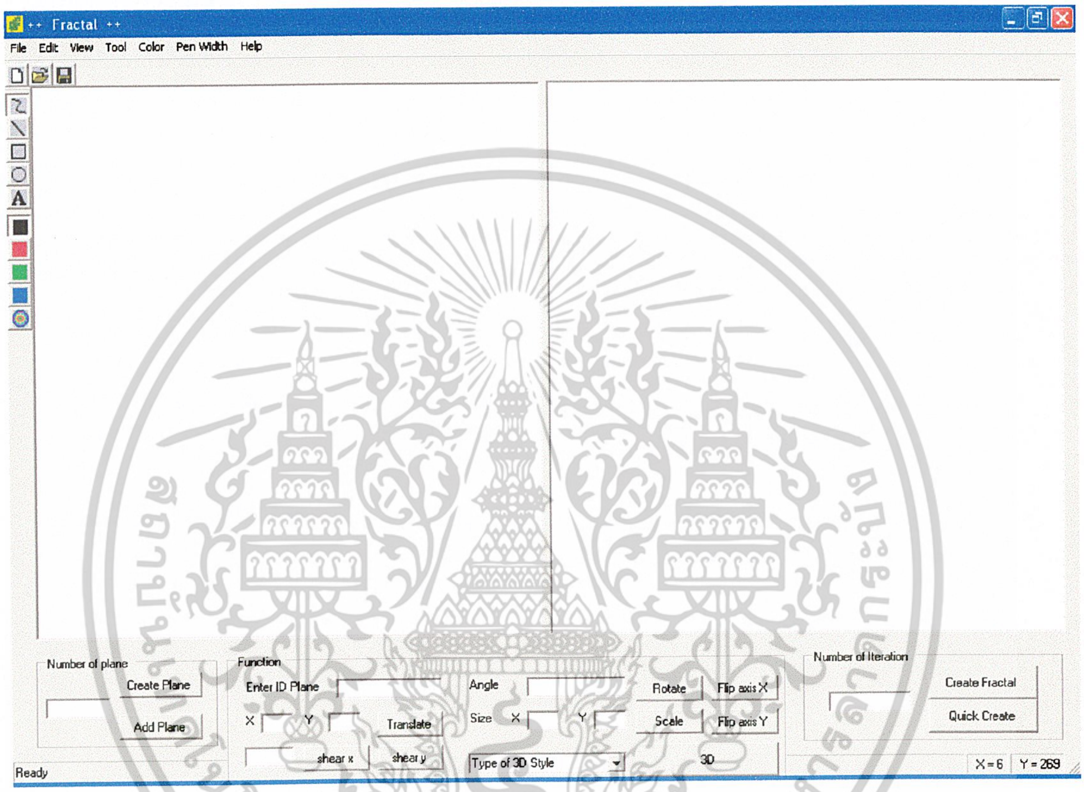
1. ควรพัฒนาให้ผู้ใช้สามารถสร้างรูปทรงแฟรคทอล 3 มิติ ได้เองโดยอิสระเช่นเดียวกับแบบ 2 มิติ

2. อาจเพิ่มการสร้างรูปแฟรคทอลด้วยวิธีการอื่นด้วยเช่น แมนเดอร์บรอท์เซตและจูเลียเซต

ภาคผนวก ก

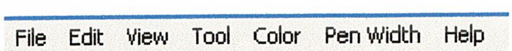
ขั้นตอนการติดตั้งและการใช้งานโปรแกรม

วิธีการใช้งานโปรแกรม



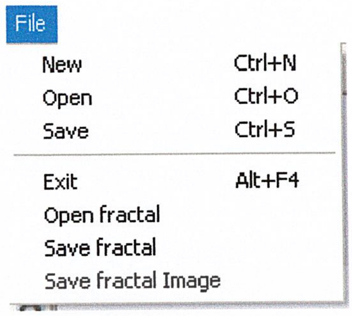
รูปที่ 6.1 หน้าจอโดยรวมของโปรแกรม

1.เมนูบาร์



รูปที่ 6.2 เมนูบาร์

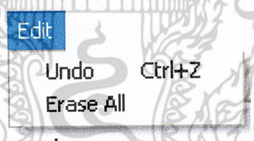
การใช้งานเมนู File



รูปที่ 6.3ภายในปุ่ม File

- New เป็นการเคลียร์หน้าต่างด้านซ้ายมือ สำหรับการสร้างรูปเริ่มต้นใหม่
- Open เป็นการเปิดไฟล์รูปเริ่มต้นที่เป็น.spd ที่ได้บันทึกไว้ออกมาใช้
- Save เป็นการบันทึกรูปเริ่มต้นที่สร้างไว้ในหน้าต่างด้านซ้ายมือเก็บเป็นไฟล์ .spd
- Exit ออกจากโปรแกรม
- Open fractal เป็นการเปิดไฟล์รูปเริ่มต้นและรูปแบบของการทำซ้ำเพื่อนำมาใช้งาน
- Save fractal เป็นการบันทึกรูปเริ่มต้นและรูปแบบการทำซ้ำที่ทำการจัดไว้เก็บเป็นไฟล์ .spd

การใช้งานเมนูEdit



รูปที่ 6.4 ภายในปุ่ม Edit

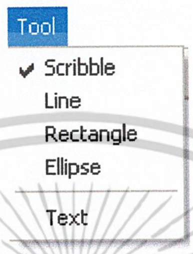
- Undo เป็นการย้อนกลับไปยังการกระทำก่อนหน้าของการสร้างรูปเริ่มต้น
- Erase All เป็นการลบรูปที่อยู่ในหน้าต่างด้านซ้ายออกทั้งหมด

การใช้งานเมนู View

รูปที่ 6.5 ภายในปุ่ม Edit

- Tool Bar เป็นการแสดงหรือซ่อน Tool Bar
- Status Bar เป็นการแสดงหรือซ่อน Status Bar
- Paint Bar เป็นการแสดงหรือซ่อน Paint Bar

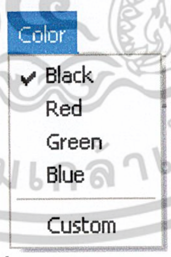
การใช้งานเมนู Tool



รูปที่ 6.6 ภายในปุ่ม Tool

- Scribble เป็นการเลือกเครื่องมือการวาดรูปแบบเส้นอิสระ
- Line เป็นการเลือกเครื่องมือการวาดรูปแบบเส้นตรง
- Rectangle เป็นการเลือกเครื่องมือการวาดรูปแบบรูปสี่เหลี่ยม
- Ellipse เป็นการเลือกเครื่องมือการวาดรูปแบบรูปวงรี
- Text เป็นการเลือกเครื่องมือการวาดรูปแบบตัวอักษร

การใช้งานเมนู Color

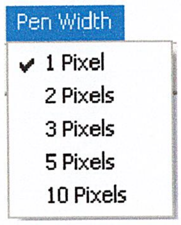


รูปที่ 6.7 ภายในปุ่ม Color

- Black เป็นการเลือกดินสอสีดำ
- Red เป็นการเลือกดินสอสีแดง
- Green เป็นการเลือกดินสอสีเขียว
- Blue เป็นการเลือกดินสอสีน้ำเงิน

- Custom เป็นการเลือกสีของดินสอแบบผสมสีเอง

การใช้งานเมนู Pen Width



รูปที่ 6.8ภายในปุ่ม Pen Width

- 1 Pixel เป็นการเลือกขนาดดินสอ 1 Pixel
- 2 Pixels เป็นการเลือกขนาดดินสอ 2 Pixel
- 3 Pixels เป็นการเลือกขนาดดินสอ 3 Pixel
- 5 Pixels เป็นการเลือกขนาดดินสอ 5 Pixel
- 10 Pixels เป็นการเลือกขนาดดินสอ 10 Pixel

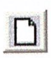
การใช้งานเมนู Pen Width

รูปที่ 6.9 ภายในปุ่ม Help

- About แสดงรายชื่อคณะผู้จัดทำ

2.ทูลบาร์

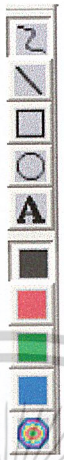
รูปที่ 6.10 ทูลบาร์

 เป็นการเคลียร์หน้าต่างด้านซ้ายมือ สำหรับการสร้างรูปเริ่มต้นใหม่

 เป็นการเปิดไฟล์รูปเริ่มต้นที่เป็น.spd ที่ได้บันทึกไว้ออกมาใช้

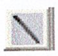
 เป็นการบันทึกรูปเริ่มต้นที่สร้างไว้ในหน้าต่างด้านซ้ายมือเก็บเป็นไฟล์ .spd

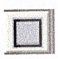
3. เพนท์บาร์

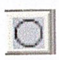


รูปที่ 6.11 เพนท์บาร์

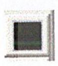
 เป็นการเลือกเครื่องมือการวาดรูปแบบเส้นอิสระ

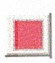
 เป็นการเลือกเครื่องมือการวาดรูปแบบเส้นตรง

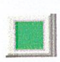
 เป็นการเลือกเครื่องมือการวาดรูปแบบรูปสี่เหลี่ยม

 เป็นการเลือกเครื่องมือการวาดรูปแบบรูปวงรี

 เป็นการเลือกเครื่องมือการวาดรูปแบบตัวอักษร

 เป็นการเลือกดินสอสีดำ

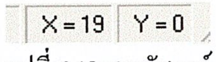
 เป็นการเลือกดินสอสีแดง

 เป็นการเลือกดินสอสีเขียว

 เป็นการเลือกดินสอสีน้ำเงิน

 เป็นการเลือกสีของดินสอแบบผสมสีเอง

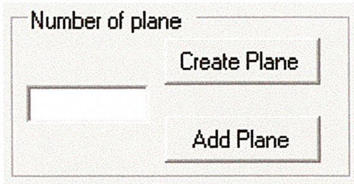
4. สเตตัสบาร์



รูปที่ 6.12 สเตตัสบาร์

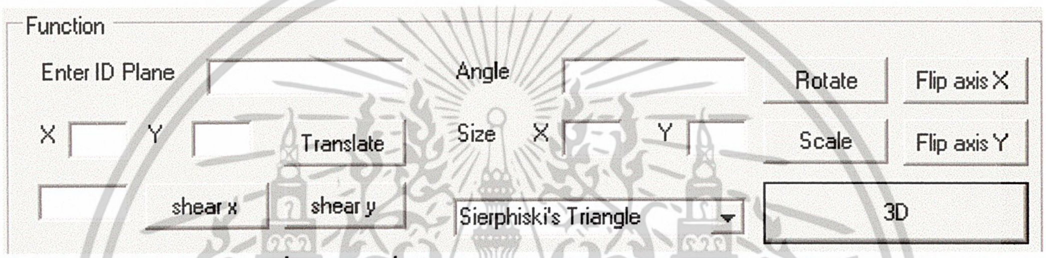
จะแสดงตำแหน่ง X และ Y ของหน้าต่างทั้งสอง

5. ฟังก์ชั่นที่ใช้ในการจัดรูปแบบในการทำซ้ำ



รูปที่ 6.13 ฟังก์ชั่นในการสร้างเพลนและเพิ่มเพลน

เอาไว้ระบุจำนวนของเพลนที่ต้องใช้ในการจัดรูปแบบในการทำซ้ำ หรือจำนวนของเพลนที่ต้องการเพิ่ม



รูปที่ 6.14 ฟังก์ชั่นในการจัดการกับเพลนและรูปแบบสามมิติ

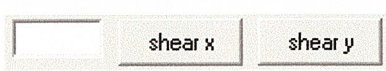
เป็นฟังก์ชั่นในการจัดรูปแบบในการทำซ้ำประกอบด้วย

รูปที่ 6.15 การใส่หมายเลขเพลน

เป็นส่วนที่เอาไว้ใส่เลขที่ของเพลนที่ต้องการจัด

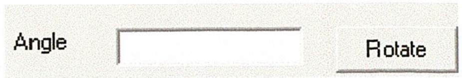
รูปที่ 6.16 ฟังก์ชั่นในการเปลี่ยนตำแหน่งเพลน

ส่วนของการย้ายภาพซึ่งต้องกำหนดระยะ(เป็น pixel)ที่ต้องการย้ายทั้งในแนวแกน X และแกน Y



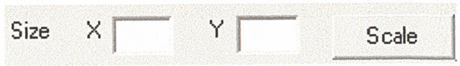
รูปที่ 6.17 ฟังก์ชั่นในการบิดภาพของเพลน

ส่วนของการบิดภาพทั้งในแนวแกน X และแกน Y ซึ่งต้องกำหนดค่าแฟคเตอร์ในการบิด



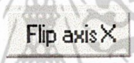
รูปที่ 6.18 ฟังก์ชั่นในการหมุนเพลน

ส่วนของการหมุนภาพซึ่งต้องกำหนดมุมที่ต้องการจะหมุน หากค่ามุมมีค่าเป็นบวกจะเป็นการหมุนแบบตามเข็มนาฬิกา หากค่ามุมเป็นลบจะเป็นการหมุนแบบทวนเข็มนาฬิกา



รูปที่ 6.19 ฟังก์ชั่นในการย่อ-ขยายเพลน

ส่วนของการย่อ - ขยายภาพซึ่งต้องกำหนดแฟคเตอร์ในการย่อ-ขยาย หากกำหนดค่าเป็นค่าบวกจะเป็นการขยายภาพ หากกำหนดค่าเป็นค่าลบจะเป็นการย่อภาพ ตัวอย่างเช่น หากเราต้องการให้ภาพย่อลงให้เป็นจำนวนห้าสิบเปอร์เซ็นต์ของภาพเดิมจะต้องใส่ค่า X เป็น -.5 และY เป็น-.5 ตามลำดับ



รูปที่ 6.20 ฟังก์ชั่นในการสะท้อนภาพทางแกน X

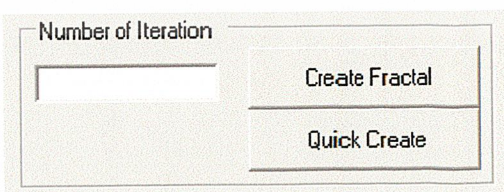
เป็นส่วนของการกลับด้านภาพในแนวแกน X โดยกลับจากซ้ายไปขวาและจากขวาไปซ้าย

รูปที่ 6.21 ฟังก์ชั่นในการสะท้อนภาพทางแกน Y

เป็นส่วนของการกลับด้านภาพในแนวแกน Y โดยกลับจากบนลงล่างและจากล่างขึ้นบน

รูปที่ 6.22 ฟังก์ชั่นในการแสดงรูปแบบแฟรคทอลสามมิติ

เป็นส่วนที่ใช้เลือกดูตัวอย่างของรูปทรงแฟรคทอลแบบ 3 มิติชนิดต่างๆ



รูปที่ 6.23 ฟังก์ชั่นในการสร้างแฟรคทอล

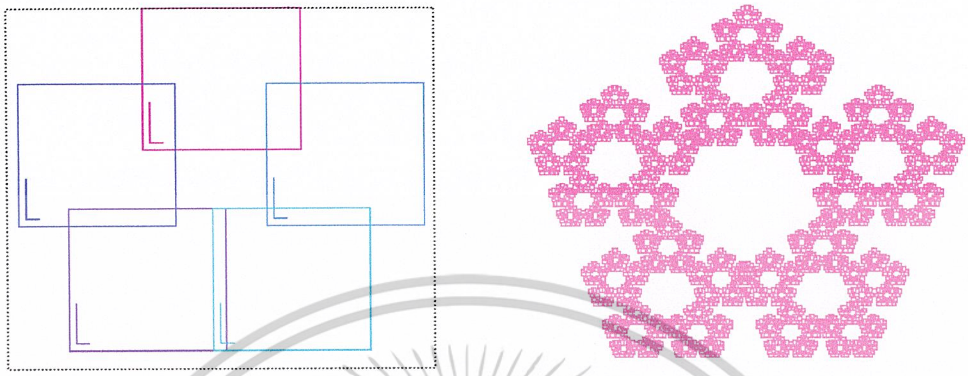
เป็นส่วนที่ใช้กำหนดค่าในการทำซ้ำ ซึ่งต้องกำหนดจำนวนครั้งของการทำซ้ำ และเลือกว่าจะประมวลผลแบบใด ระหว่าง

- Create Fractal โปรแกรมจะแสดงผลของการทำซ้ำทุกครั้งออกมาทางหน้าต่างด้านซ้าย
- Quick Create โปรแกรมจะแสดงเฉพาะผลของการทำซ้ำครั้งสุดท้ายออกมา ซึ่งเป็นการประหยัดเวลาในการประมวลผล

ภาคผนวก ข

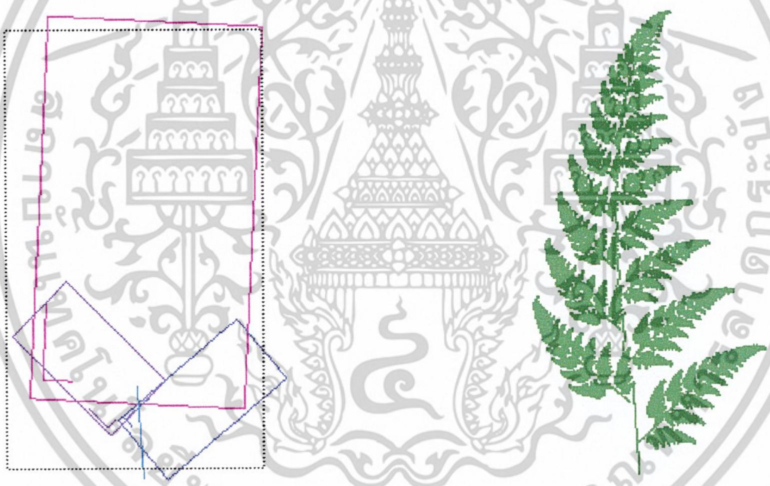
ตัวอย่างภาพแฟรคทอลและรูปแบบการจัด

Crystal with Five Transformations



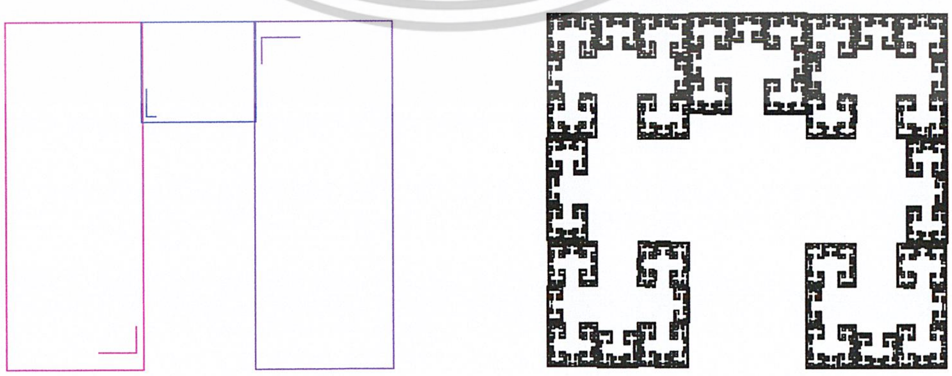
รูปที่ 7.1

Barnsley's Fern



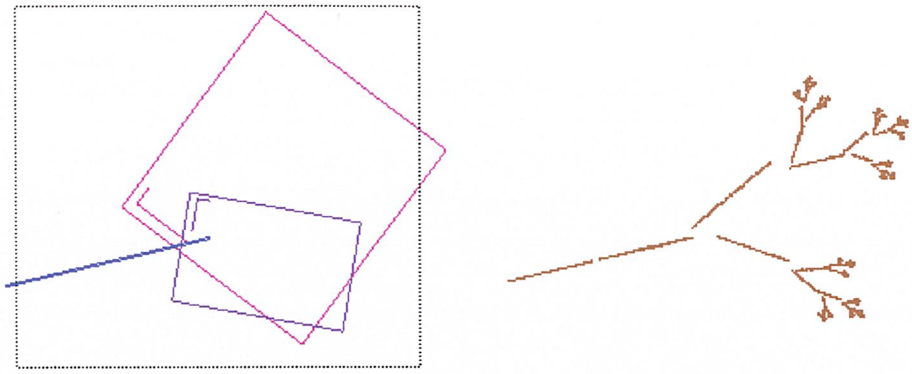
รูปที่ 7.2

The Cantor Maze



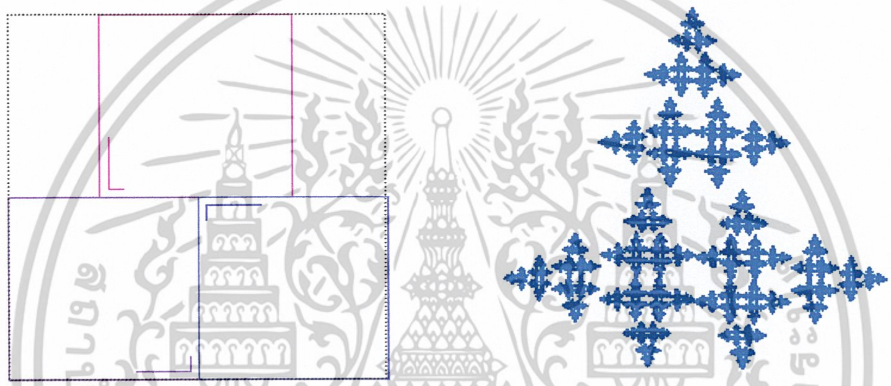
รูปที่ 7.3

A Twig



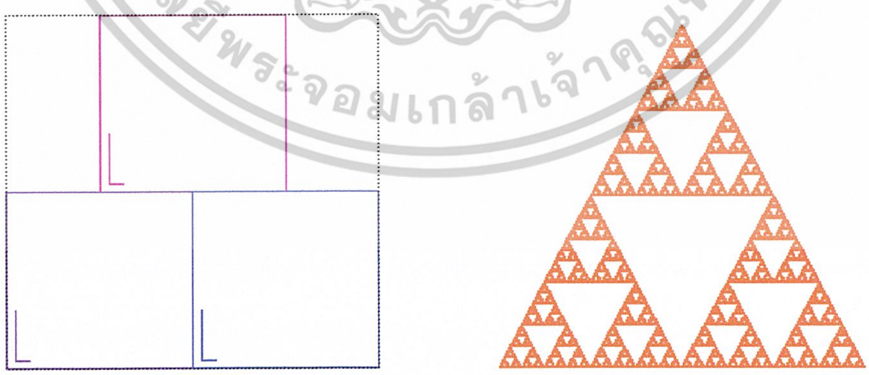
รูปที่ 7.4

The Twin Christmas Tree



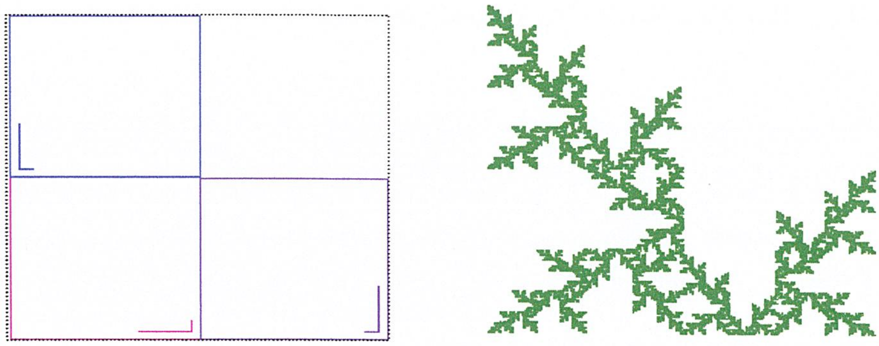
รูปที่ 7.5

Sierpinski Triangle

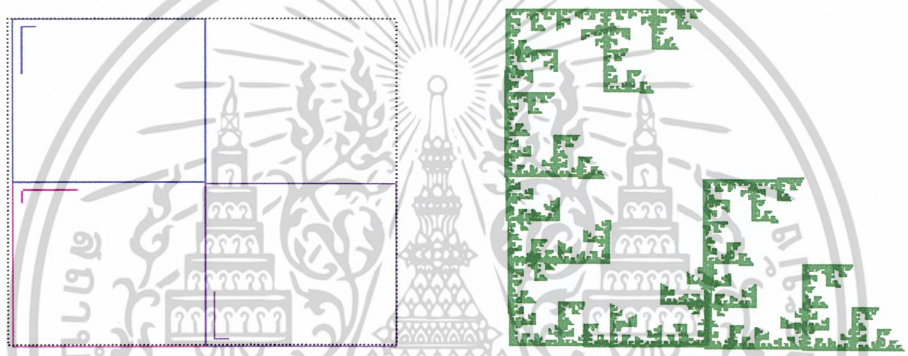


รูปที่ 7.6

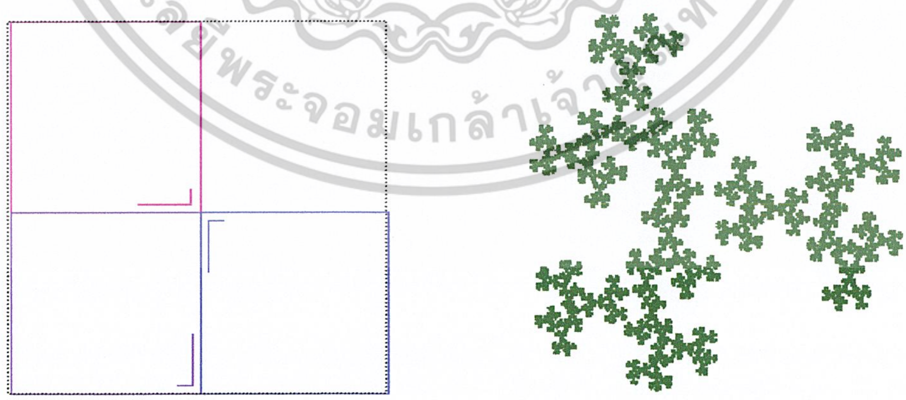
การจัดวางตำแหน่งเดียวกัน แต่ต่างกันที่การพลิกรูปทำให้รูปแฟรคทอลที่ได้มีรูปแบบออกมาแตกต่างกัน ตัวอย่างเช่น



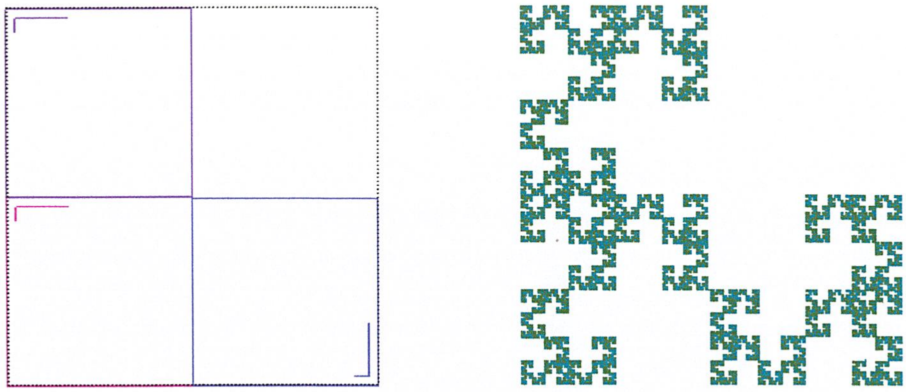
รูปที่ 7.7



รูปที่ 7.8



รูปที่ 7.9



รูปที่ 7.10

บรรณานุกรม

Michael Barnsley. 1988 . **Fractal Everywhere** . Academic Press .

Peitgen , Jurgens , Saupe. 1992 . **Chaos and Fractals New Frontiers of Science** . Springer-Verlag .

Armin Bunde , Shlomo Havlin (Eds). 1991,1996 . **Fractals and Disordered Systems** . Springer

Tim Wegner , Bert Tyler. 1993 . **Fractal Creations Second Edition** . Waite Group Press