การใช้เทคโนโลยี OLE 2.0สำหรับแอพพลิเคชันทางด้านการประมวลผลภาพ

โดย

นางสาวกนกพร ยิ่งยงวรปัญญา รหัสประจำตัว 35504101
นางสาวพัธรินทร์ กอทรัพย์ไพศาล รหัสประจำตัว 35504115
นางสาววิลาวัลย์ มาลัยนวล รหัสประจำตัว 35504124

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปีการศึกษา 2538

หัวข้อปัญหาพิเศษ	การใช้เทคโนโลยี OLE 2.0 สำหรับแอพพลิเคชันทางด้านการประมวลผลภาพ
โดย	นางสาวกนกพร ยิ่งยงวรปัญญา
	นางสาวพัธรินทร์ กอทรัพย์ไพศาล
	นางสาววิลาวัลย์ มาลัยนวล
ภาควิชา	คณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์สุนทร สุชาติเวชภูมิ
	อาจารย์จิตรลดา ผลนิมิตร

ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหารลาดกระบัง อนุมัติให้นับโครงงานพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต สาขาคณิตศาสตร์ประยุกต์ ประจำปีการศึกษา 2538

(รองศาสตราจารย์ภัคคินี ชิตสกุล) หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะกรรมการโครงงานพิเศษ

(อาจารย์ไพรบูลย์ พันธรักษ์พงษ์) ประธานกรรมการ

(อาจารย์กาญจนา คำนึงกิจ) กรรมการ

(ผู้ช่วยศาสตราจารย์ สุนทร สุชาติเวชภูมิ) กรรมการและอาจารย์ที่ปรึกษา

(อาจารย์จิตรลดา ผลนิมิตร) กรรมการและอาจารย์ที่ปรึกษาร่วม

หัวข้อโครงงานพิเศษ การใช้เทคโนโลยี OLE 2.0 สำหรับแอพพลิเคชันทางด้านการประมวลผลภาพ

นักศึกษา	นางสาวกนกพร ยิ่งยงวรปัญญา	**รหัสประจำตัว** 35504101
	นางสาวพัธรินทร์ กอทรัพย์ไพศาล	**รหัสประจำตัว** 35504115
	นางสาววิลาวัลย์ มาลัยนวล	**รหัสประจำตัว** 35504124

อาจารย์ที่ปรึกษา ผู้ช่วยศาสตราจารย์ สุนทร สุชาติเวชภูมิ
อาจารย์จิตรลดา ผลนิมิตร

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

ปีการศึกษา 2538

บทคัดย่อ

การพัฒนาแอพพลิเคชันการประมวลผลภาพนี้เพื่อให้สามารถใช้งานร่วมกับแอพพลิเคชันอื่นที่ทำงานบนระบบปฏิบัติการวินโดวส์ และเพื่อให้ง่ายต่อการปรับปรุงพัฒนาในอนาคต เทคนิคที่ใช้ในการพัฒนาคือ OLE 2.0 ซึ่งเป็นเทคนิคในการเชื่อมโยงออบเจกต์ และฝังออบเจกต์ โดยใช้การเขียนโปรแกรมแบบเชิงวัตถุ OLE ยังมีอินเตอร์เฟส มาตรฐานในการติดต่อเชื่อมโยงกันระหว่างออบเจกต์ ทำให้แอพพลิเคชันขยายขอบเขตการทำงานได้กว้างขึ้น และทำให้แอพพลิเคชันอื่นสามารถเรียกใช้งานได้

แอพพลิเคชันการประมวลผลภาพโดยใช้เทคนิค OLE 2.0 นี้ สามารถแก้ไขและดัดแปลงภาพได้โดยตัวโปรแกรมสามารถให้บริการแอพพลิเคชันอื่นได้ เช่น ทำการปรับสีและความสว่างของภาพ การสร้างภาพขาวดำ การขจัดสัญญาณรบกวนในภาพ การทำภาพมัว การหาขอบของภาพ การกลับภาพ และการหมุนภาพ โดยใช้กับแฟ้มข้อมูลภาพแบบบิตแมพ

Special Project Title	Applying OLE 2.0 Technology to Digital Image Processing Application	
Name	Miss Kanokporn Yingyongworapanya	35504101
	Miss Pattarin Korsubpaisarn	35504115
	Miss Wilawal Malainual	35504124
Special Project Advisor	Assistant Professor Sunthorn Suchatvejapoom	
	Miss Chitlada Pholnimit	
Department	Applied Mathematics and Computer Science	
Acadamic Year	1995	

Abstract

The development of image processing application is to create the Windows application that can use with other applications. Since it is going to develop easily in the future, We use the OLE technique. The OLE 2.0 is the technique in the object oriented programming for linking and embedding objects. The OLE has a standard interface for connection between object and another, so other applications can use it conveniently.

This OLE image processing application use for modify bitmap image (BMP) file, and it can be a image processing server. These are color and bright adjust, threshold and gray scale transformation, noise filtering, image bluring, edge detection, image flip and so on image rotation in this image algorithm.

กิตติกรรมประกาศ

ปัญหาพิเศษนี้สำเร็จได้เพราะความช่วยเหลือของบุคคลต่าง ๆ ดังนี้

1. อาจารย์สุนทร สุชาติเวชภูมิ อาจารย์ที่ปรึกษาปัญหาพิเศษ
   ช่วยแนะนำการจัดทำเอกสารคู่มือปัญหาพิเศษ และจัดหาเอกสารอ้างอิงต่าง ๆ ที่เกี่ยวข้องกับการทำปัญหาพิเศษ
2. อาจารย์จิตรลดา ผลนิมิตร อาจารย์ที่ปรึกษาปัญหาพิเศษ
   ช่วยแนะนำวิธีที่ใช้ในการตกแต่งภาพ, ช่วยให้แนวคิดในการเขียนโปรแกรม รวมทั้งให้คำปรึกษาต่าง ๆ
3. อาจารย์ปวีณา ทองใบ อาจารย์ที่ปรึกษาปัญหาพิเศษ
   ให้ข้อมูลและคำแนะนำในหลักการต่าง ๆ ที่เกี่ยวข้องกับปัญหาพิเศษ
4. อาจารย์ศรันย์ อินทโกสุม
   ช่วยให้คำแนะนำ และแนวคิด
5. คุณสืบสรวง แสงเพ็ชร
   คุณศักดิ์พงษ์ จันทร์ใบ
   ได้ให้คำแนะนำในการดำเนินการพัฒนาแอพพลิเคชั่นทางด้านเทคนิคโอแอลอีและอนุเคราะห์ทางด้านเอกสารต่าง ๆ ในการทำปัญหาพิเศษ เป็นอย่างดียิ่ง
6. เจ้าหน้าที่ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ที่ให้ความเอื้อเฟื้อในด้านอุปกรณ์และวัสดุที่ใช้ในการทำปัญหาพิเศษ

คณะผู้จัดทำ

สารบัญ

หน้า

บทคัดย่อปัญหาพิเศษภาษาไทย

บทคัดย่อปัญหาพิเศษภาษาอังกฤษ

กิติกรรมประกาศ

สารบัญ

สารบัญภาพประกอบ

บทที่ 1 บทนำ 1

1.1 ความสำคัญและที่มาของปัญหา 1

1.2 วัตถุประสงค์ของปัญหาพิเศษ 1

1.3 ขอบเขตของปัญหาพิเศษ 1

1.4 ขั้นตอนในการดำเนินงาน 2

1.5 ประโยชน์ที่คาดว่าจะได้รับ 2

บทที่ 2 ทฤษฎีที่ใช้ในการจัดการข้อมูลรูปภาพ 3

2.1 ความรู้เบื้องต้นเกี่ยวกับอิมเมจโปรเซสซิง 3

2.1.1 ความหมายของอิมเมจโปรเซสซิง 3

2.1.2 สัณฐานของอิมเมจ 4

2.2 สีของข้อมูลรูปภาพ (Image Colour Standard) 5

2.3 ข้อมูลรูปภาพแบบบิตแมพ 8

2.3.1 ประเภทของบิตแมพบนวินโดวส์ 8

2.3.2 ความเข้าใจในบิตแมพ 9

2.3.3 โครงสร้างของไฟล์บีเอ็มพี 10

บทที่ 3 การเขียนโปรแกรมเชิงวัตถุ (Object Oriented Programming) 12

3.1 ความหมายของโอโอพี (OOP) 12

3.2 แนวความคิดใหม่ของโอโอพี 12

3.2.1 วัตถุ (Object) 12

3.2.2 การสืบทอดคุณสมบัติ (Inheritance) 13

3.2.3 พหุลักษณ์ (Polymorphism) 14

หน้า

3.2.4 ตัวสร้าง (Constructors) 14

3.2.5 ตัวทำลาย (Destructors) 14

3.3 ข้อดีของปัญหาเชิงวัตถุ 14

บทที่ 4 การใช้เทคนิคโอแอลอี (OLE : Object Linking Embedding) 15

4.1 ความหมายของโอแอลอี 15

4.2 ลักษณะสำคัญของโอแอลอี (OLE feature) 21

4.3 ระดับของฟีเจอร์ในโอแอลอี 22

4.3.1 ฟีเจอร์ระดับล่าง 22

4.3.2 ฟีเจอร์ระดับกลาง 23

4.3.3 ฟีเจอร์ระดับสูง 26

บทที่ 5 การพัฒนาโปรแกรมการจัดการข้อมูลรูปภาพ 29
โดยใช้เทคนิคโอแอลอี

5.1 อัลกอริทึมที่ใช้ในการจัดการข้อมูลรูปภาพ 29

5.1.1 อัลกอริทึมแบบ Pixel Point Processing 29

ก. การปรับค่าความสว่าง 29

ข. การสร้างภาพเกรย์สเกล 31

ค. การสร้างภาพขาวดำ 33

5.1.2 อัลกอริทึมแบบ Area Processing 35

ก. การขจัดสัญญาณรบกวนของภาพ 35

- เทคนิค Low Pass 35

- เทคนิค High Pass 36

ข. การทำภาพมัว 38

ค. การหาขอบของภาพ 40

5.1.3 อัลกอริทึมแบบ Geometric Transformation
Processing 44

ก. การกลับภาพ 44

- กลับภาพตามแนวนอน 44

หน้า

- กลับภาพตามแนวตั้ง 45

ข. การหมุนภาพ 48

- หมุนภาพตามเข็มนาฬิกา 48

- หมุนภาพตามทวนเข็มนาฬิกา 48

5.2 การเขียนโปรแกรมโดยใช้โอแอลอี 50

5.2.1 In - Place Activation 50

5.2.2 โครงสร้างของแอพพลิเคชันเซิร์ฟเวอร์ 50

5.2.3 ความแตกต่างระหว่างมินิเซิร์ฟเวอร์และฟูลเซิร์ฟเวอร์ 51

5.2.4 ขั้นตอนในการสร้างเซิร์ฟเวอร์แบบฝังตัว 51

5.2.5 ตัวอย่างการทำงาน 53

5.3 การใช้งานโปรแกรมการจัดการข้อมูลรูปภาพ 56

บทที่ 6 สรุปผลและเสนอแนะ 66

บรรณานุกรม

ภาคผนวก

สารบัญรูปภาพ

รูปที่	หน้า
2.1 แสดงโมเดลรูปกรวยฐานหกเหลี่ยม	6
2.2 แสดงโมเดลรูปกรวยคู่	7
2.3 ไดอะแกรม 2 มิติที่เกิดจากกำหนดจุดที่แทนสีหลักและสีที่เกิดจากการผสมของสีหลัก	8
4.1 แสดงเอกสารเชิงซ้อนที่สร้างโดยการเชื่อมโยงออบเจกต์	15
4.2 แสดงเอกสารเชิงซ้อนที่สร้างโดยการฝังออบเจกต์	15
4.3 แสดงความสัมพันธ์ระหว่างอินเตอร์เฟสกับฟังก์ชัน	16
4.4 แสดงวิธีการเรียกใช้งานฟังก์ชันผ่านอินเตอร์เฟส	17
4.5 แสดงการเชื่อมโยงระหว่างอินเตอร์เฟสกับออบเจกต์	17
4.6 แสดงขอบเขตของการเข้าถึงฟังก์ชันของแต่ละพอยน์เตอร์ที่ชี้อินเตอร์เฟส	18
4.7 แสดงผลที่ได้จากการเรียกใช้ฟังก์ชัน QueryInterface	19
4.8 แสดงอินเตอร์เฟสที่น่าสนใจในเทคโนโลยีของโอแอลอี 2.0	20
4.9 แสดงฟีเจอร์ในระดับต่าง ๆ ของ OLE	21
4.10 แสดงตัวอย่างของไฟล์เชิงซ้อน	24
4.11 แสดงวิธีในการกำจัดแฟรกเมนท์ในไฟล์เชิงซ้อน	24
4.12 แสดงการตั้งชื่อของลิงค์ออบเจกต์	25
4.13 แสดงการดำเนินการลากและวาง	27
5.1 แสดงภาพที่ได้จากการปรับสว่าง	30
5.2 แสดงภาพที่ได้จากการสร้างภาพเป็นแบบเกรย์สเกล	32
5.3 แสดงภาพที่ได้จากการสร้างภาพขาวดำ	34
5.4 แสดงภาพที่ได้จากการขจัดสัญญาณรบกวนด้วยเทคนิค LOW PASS	37
5.5 แสดงภาพที่ได้จากการขจัดสัญญาณรบกวนด้วยเทคนิค HIGH PASS	37
5.6 แสดงภาพที่ได้จากการทำภาพมัว	39
5.7 แสดงภาพที่ได้จากการหาขอบเขตของภาพ	43
5.8 แสดงภาพที่ได้จากการกลับภาพตามแนวนอน	45
5.9 แสดงภาพที่ได้จากการกลับภาพตามแนวตั้ง	47
5.10 แสดงภาพที่ได้จากการหมุนภาพตามเข็มนาฬิกา	49

5.11 แสดงภาพที่ได้จากการหมุนภาพตามทวนเข็มนาฬิกา 49
5.12 แสดงโครงสร้างของแอพพลิเคชันเซิร์ฟเวอร์ 50
5.13 แสดงตัวอย่างการทำงานของแอพพลิเคชันคอนเทนเนอร์ 53
5.14 แสดงผลการทำงานของแอพพลิเคชันเซิร์ฟเวอร์คอนเทนเนอร์ 54
5.15 แสดงผลจากการเพิ่มออบเจกต์ลงในเอกสารเชิงซ้อน 54
5.16 แสดงแถบเมนูที่เกิดจากการรวมกันของเมนูคอนเทนเนอร์กับ กับเมนูเซิร์ฟเวอร์ 55
5.17 หน้าจอเซิร์ฟเวอร์ที่ได้จากการรันโปรแกรม 56
5.18 ผลที่ได้จากการเลือกเมนูคำสั่งไฟล์ 57
5.19 ผลที่ได้จากการเลือกเมนูคำสั่งอิมเมจ 57
5.20 ผลที่ได้จากการเลือกเมนูคำสั่งอิมเมจ 58
5.21 ผลที่ได้จากการเลือกเมนูคำสั่งอิมเมจ 59
5.22 ผลที่ได้จากการเลือกเมนูคำสั่งอิมเมจ 59
5.23 ผลที่ได้จากการเลือกเมนูคำสั่งวิว 60
5.24 ผลที่ได้จากการเลือกเมนูคำสั่งวินโดวส์ 60
5.25 ผลที่ได้จากการเลือกเมนูคำสั่งช่วยเหลือ 61
5.26 ผลที่ได้จากการเพิ่มออบเจกต์ 62
5.27 ผลที่ได้จากการเลือกออบเจกต์ 63
5.28 ผลที่ได้จากการแทรกออบเจกต์เข้าไป 64
5.29 ผลที่ได้จากการรันโปรแกรมฝังตัว 65

บทที่ 1

บทนำ

ความสำคัญและที่มาของปัญหา

ในปัจจุบันการนำโปรแกรมเก่าไปพัฒนาจนเกิดเป็นโปรแกรมใหม่ในแต่ละครั้ง โดยมีจุดประสงค์เพื่อเพิ่มประสิทธิภาพของซอฟท์แวร์ทำได้ยาก และสำหรับแอพพลิเคชันทางด้านการแก้ไขและดัดแปลงภาพที่มีอยู่ยังขาดคุณสมบัติที่ดีในการนำออบเจกต์ที่มีอยู่นั้นกลับมาใช้งานใหม่ ดังนั้นจึงมีนักคอมพิวเตอร์ได้คิดวิธีการที่ทำให้แอพพลิเคชันแต่ละตัวสามารถดึงโมดูลการทำงานของอีกแอพพลิเคชันหนึ่งมาร่วมทำงาน โดยใช้เทคโนโลยีของ โอแอลอี (OLE : Object Linking and Embedding) เพื่อเพิ่มประสิทธิภาพให้กับแอพพลิเคชันอื่น ๆ

การใช้เทคนิคโอแอลอีมาช่วยในการพัฒนาโปรแกรมด้านการแก้ไขดัดแปลงภาพ จะช่วยทำให้มีซอฟท์แวร์ทางด้านนี้ที่มีคุณภาพ และมีมาตรฐานยิ่งขึ้น อีกทั้งยังง่ายต่อการใช้งาน และการพัฒนาในอนาคต

วัตถุประสงค์ของปัญหา

1. เพื่อศึกษาถึงการพัฒนาแอพพลิเคชันโดยใช้โอแอลอี เพื่อให้มีมาตรฐาน และสามารถใช้งานร่วมกับแอพพลิเคชันอื่นได้ง่าย
2. เพื่อศึกษาถึงการเขียนโปรแกรมแบบโอโอพี (OOP : Object Oriented Programming)
3. เพื่อศึกษาทฤษฎีที่เกี่ยวข้องกับการแก้ไขและดัดแปลงภาพ
4. พัฒนาแอพพลิเคชันทางด้านการแก้ไขและดัดแปลงภาพโดยใช้เทคโนโลยีของโอแอลอี

ขอบเขตของปัญหา

1. ศึกษาการพัฒนาแอพพลิเคชั่นโดยใช้เทคโนโลยีของโอแอลอีบนไมโครซอฟท์
2. พัฒนาแอพพลิเคชันตัวอย่างสำหรับการแก้ไขและดัดแปลงภาพในเรื่องของ การปรับความสว่าง การปรับสี การสร้างภาพขาวดำ การขจัดสัญญาณรบกวนของภาพ การหาขอบของภาพ และการหมุนภาพ เป็นต้น
3. ศึกษาเฉพาะกรณีแฟ้มข้อมูลรูปภาพแบบบิตแมป เท่านั้น
4. แอพพลิเคชันที่พัฒนาขึ้นนี้ใช้งานบนระบบปฏิบัติการวินโดวส์ 95

ขั้นตอนในการดำเนินงาน

1. ศึกษารายละเอียดของการทำงานและการจัดการในส่วนของข้อมูลรูปภาพ และการจัดเก็บข้อมูลรูปภาพแบบบิตแมป
2. ศึกษารายละเอียดของการเขียนโปรแกรมแบบโอแอลอี
3. ศึกษาการเขียนโปรแกรมโดยใช้ MS Visual C++
4. ศึกษาการพัฒนาแอพพลิเคชันสำหรับใช้งานบนระบบปฏิบัติการ Windows 95
5. ออกแบบและพัฒนาโปรแกรม
6. ทดสอบการทำงานและปรับปรุงประสิทธิภาพให้ดียิ่งขึ้น
7. สรุปผลและเรียบเรียงปัญหาพิเศษ

ประโยชน์ที่ได้รับ

เพื่อเป็นแนวทางให้กับผู้พัฒนาแอพพลิเคชันที่ต้องการใช้เทคโนโลยี OLE 2.0 และเป็นพื้นฐานในการนำแอพพลิเคชันที่มีอยู่ในปัญหาพิเศษฉบับนี้ไปพัฒนาให้เป็นแอพพลิเคชันทางด้าน อิมเมจโปรเซสซิง (image processing) ให้มีประสิทธิภาพสูงสุดต่อไป

บทที่ 2
ความหมายและทฤษฎีของอิมเมจ

2.1 ความรู้เบื้องต้นเกี่ยวกับอิมเมจโปรเซสซิง

2.1.1 ความหมายของอิมเมจโปรเซสซิง

อิมเมจโปรเซสซิง (Image Processing) เป็นกระบวนการที่กระทำกับข้อมูลรูปภาพอิมเมจโดยคอมพิวเตอร์เพื่อที่จะสร้างอิมเมจอื่นขึ้นมา อาจกล่าวโดยละเอียดได้ว่าอิมเมจโปรเซสซิง คือ การแสดงข้อมูลรูปภาพที่เกิดจากการถ่ายรูป หรือ จากการสแกนภาพให้ปรากฏบนจอภาพคอมพิวเตอร์ วิธีการทางอิมเมจโปรเซสซิงจะต่างกับวิธีการทางคอมพิวเตอร์กราฟฟิก กล่าวคือในระบบคอมพิวเตอร์กราฟฟิกนั้น ตัวคอมพิวเตอร์เองจะเป็นตัวที่สร้างภาพแต่เทคนิคทางอิมเมจโปรเซสซิงนั้น ใช้คอมพิวเตอร์สำหรับจัดการรูปแบบของสี แสงเงา หรืออื่นๆ ที่มีอยู่แล้วในภาพให้เป็นข้อมูลทางดิจิตอล โดยอาจมีวิธีการทำให้ภาพที่รับเข้ามานั้นมีความชัดเจนมากขึ้นก่อน จากนั้นก็จัดการเก็บข้อมูลรูปภาพทางดิจิตอลนี้ แล้วส่งภาพออกไปที่จอภาพของคอมพิวเตอร์อีกครั้งหนึ่ง

การประยุกต์อัลกอลิทึมอิมเมจโปรเซสซิงของอิมเมจหนึ่ง ๆ อาจจะไม่ทำให้ได้ภาพที่ต้องการทุกครั้งเสมอ แต่ถ้ามีการแปลงภาพเกิดขึ้น มันจะถูกกำหนดโดยข้อมูลที่เพิ่มเติมขึ้นมาจากอิมเมจรูปเดิมและจะได้ผลของการแปลงภาพนั้นมา

ชนิดที่แตกต่างกันทั้ง 4 ของอัลกอริทึมในการประมวลผลภาพ คือ

1. พอยต์ โปรเซส (Point Processes) มีการประมวลผลโดยเลือกกระทำกับค่าของพิกเซลในอิมเมจเดิมเพื่อจะได้ค่าของพิกเซลในอิมเมจใหม่ขึ้นมา

2. แอเรีย โปรเซส (Area Processes) เป็นการประมวลผลภาพที่เลือกกระทำกับค่าของพิกเซลในภาพและกระทำกับพิกเซลที่อยู่รอบ ๆ พิกเซลนั้นด้วย

3. เฟรม โปรเซส (Frame Processes) เลือกค่าพิกเซลภายในภาพเดิมมาประมวลผลโดยจะนำเสนอในรูปแบบหนึ่งเฟรมหรือหลายๆเฟรม

4. จีโอเมตริกซ์ โปรเซส (Geometric Processes) การประมวลผลนี้เป็นการเปลี่ยนแปลงของตำแหน่งของพิกเซลในอิมเมจเดิมตามการเปลี่ยนแปลงทางเรขาคณิต

ในปัญหาพิเศษนี้จะไม่ขอกล่าวถึงในเรื่องของการสร้างภาพหรือกระบวนการในการได้ภาพอิมเมจขึ้นมา แต่จะกล่าวถึงเฉพาะในส่วนของการนำข้อมูลรูปภาพอิมเมจไปใช้ในแอพพลิเคชันที่พัฒนาขึ้นมานี้เท่านั้น

2.1.2 สัณฐานของอิมเมจ

อิมเมจโมเดลเชิงสี่เหลี่ยมเป็นโมเดลของหน่วยความจำที่นิยมที่สุด เกรย์เลเวลอิมเมจ (Gray level image) จะเกิดจากเซตของจุด หรือ พิกเซล (pixel) โดยแต่ละพิกเซลจะเก็บค่าเกรย์เลเวลอยู่ระหว่างค่า 0 ถึง 2^g-1 ซึ่ง g เป็นจำนวนเต็ม โครงสร้างของอิมเมจโมเดลจะเป็นอาร์เรย์ของพิกเซล ซึ่งมีขนาด M แถว และ N หลัก ดังนั้นแต่ละพิกเซลจะเป็นอิลีเมนต์ หรือสมาชิกของอาร์เรย์นั่นเอง

ในภาษา C และ C++ เราสามารถประกาศได้ดังนี้

```
unsigned char image[M][N] ;
```

โดยจะมีค่าอินเด็กซ์องสมาชิกของอาร์เรย์ ตั้งแต่ค่า 0 ถึง N-1 หลัก และ 0 ถึง M-1 แถว ในภาษา C นั้นการเพิ่มค่าอินเด็กซ์ 1 ค่า ก็คือการอ้างถึงค่าในหน่วยความจำเลื่อนขึ้นไปอีก 1ตำแหน่ง

ข้อมูลรูปภาพต้องการหน่วยความจำขนาด N x M x g บิต ในการเก็บค่าของภาพ โดยที่ค่า g จะมีค่าระหว่าง 1-10 แต่โดยทั่วไปแล้ว ค่า g จะเท่ากับ 8 ซึ่งให้ค่า 256 เกรย์เลเวล ส่วนค่า N และ M จะเป็นตัวแสดงความละเอียดชัดเจนของภาพ หรือเรียกว่าค่าเรโซลูชัน (resolution) สำหรับจอวีจีเอ (VGA : Video Graphics array) นั้น ค่า N x M ควรจะเป็น 640x480 ซึ่งจะให้ค่าสี 16 สีหรือ 16 เกรย์เลเวล

การกำหนดเรโซลูชันของข้อมูลรูปภาพจะขึ้นอยู่กับแอพพลิเคชันที่จะใช้ภาพนั้น ว่าต้องการให้ภาพมีความละเอียดชัดเจนอยู่ในระดับใด เช่นบางงานก็ต้องการเพียง 30 x 50 แต่บางงานอาจจะต้องการเรโซลูชันสูงถึง 1000 x 1000 ก็ได้

ค่าสีของแต่ละพิกเซลจะถูกเก็บแบบเรียงลำดับบิต ยกตัวอย่างเช่น ในแอพพลิเคชันทางด้านงานสิ่งพิมพ์แบบสีเดียว (mono desktop publishing application) สีของพิกเซลจะมีเพียงสีขาวกับสีดำ ดังนั้นจะใช้เพียง 1 บิตในการจัดเก็บข้อมูลสี อุปกรณ์การแสดงผลของคอมพิวเตอร์จะต้องมีการแมป อาร์เรย์ของพิกเซลไปยังอาร์เรย์ 1 มิติ ตัวอย่างเช่น อาร์เรย์ 2 มิติของบิตเดี่ยว ในทางปฏิบัติบิตเหล่านี้จะเป็นสมาชิกของกลุ่มบิต 8 บิต หรือไบท์ (byte) นั่นเอง ดังนั้นการโอเปอร์เรชันภายในโปรเซสเซอร์นี้จะไม่สามารถกระทำได้เฉพาะบิตแต่จะต้องกระทำกับไบต์ ตัวอย่างเช่นเราจะต้องคัดลอก (copy) ค่าไบต์ออกมาทำการแก้ไขค่าบิตใดๆ ในโปรเซสเซอร์หลักแล้วจึงทำการคัดลอกกลับลงไปที่เดิม

สำหรับระบบสีที่มากกว่า 2 สี (คือขาวและดำ) ต่อพิกเซล จำนวนของบิตต่อพิกเซลจะมีค่าเพิ่มขึ้นเช่น ระบบ 4 สี จะต้องการ 2 บิตต่อพิกเซล ดังนี้

00	blue
01	light blue
10	dark blue
11	mauve

2.2 สีของข้อมูลรูปภาพ (Image Colour Standard)

จำนวนสีมาตรฐานบนฮาร์ดแวร์จะขึ้นอยู่ กับโมเดลที่แตกต่างกันสำหรับการเก็บสี ในมาตรฐานทั่ว ๆ ไป พิกเซลของสีหนึ่งจะถูกแสดงโดยจุดในมิติ 3 มิติ มิติที่ว่าอาจจะมีแกนที่ระบุชื่อเป็นสีที่ไม่เกี่ยวกันเช่น สีแดง สีเขียว และสีน้ำเงิน หรืออาจจะใช้อินดิเคเตอร์อื่น ๆ เช่น สีอื่น ความสว่าง แสงอ่อนหรือแสงจ้า มาตรฐานในเรื่องสีที่นิยมใช้กันคือ อาร์จีบี (RGB), เอชเอสวี (HSV : Hue Saturation Value) และเอชแอลเอส (HLS : Hue Lightness Saturation) โดยที่ อาร์จีบี (RGB) เป็นการรวมกันของสีแดง เขียว และน้ำเงิน ที่แสดงผลบนหน้าจอโดยหลอดรังสีคาโธด, HSV เป็นการจัดการข้างต้นของสีแดง เขียว และน้ำเงิน ในขณะที่ HLS ค่อนข้างที่จะซับซ้อนกว่า

ใน HSV นั้น สี (hue) เป็นผลลัพธ์การวัดของความยาวคลื่นของค่าสีหลัก ในตัวอย่างจะมีค่าระหว่าง 0-255 กล่าวได้ว่า 0 จะแทนสีแดง เมื่อปรับค่าสเปกตรัมเป็น 256 สีที่ได้ก็ยังเป็นสีแดง ซึ่งสามารถแสดงได้ในรูปของมุม โดยที่ red = 0°, green = 120°, และ blue = 240°

โดยที่ แต่ละสีสามารถคำนวณได้จากค่า RGB ดังนี้

$$red_h = red - \min(red, green, blue)$$
$$green_h = green - \min(red, green, blue)$$
$$blue_h = blue - \min(red, green, blue)$$

ค่าต่ำสุดคือ 0 ถ้าเกิดค่า 0 สองตัวแล้วค่า hue ก็คือ มุมที่สัมพันธ์กับตัวที่ 3 ที่มีค่าสีไม่เท่ากับ 0 และถ้าทั้งสามค่ามีค่าเป็น 0 แล้วจะไม่เกิดสีใด ๆ ตัวอย่างคือจอภาพที่แสดงค่า grey level ระหว่างดำและขาว และถ้ามีเพียง 1 ตัวประกอบที่มีค่าเป็น 0 แล้ว มุมของสีจะอยู่ระหว่างมุมที่สัมพันธ์กับ 2 ตัวประกอบที่เหลือ ค่าที่ได้จะเป็นไปตามตัวประกอบที่ใหญ่กว่า อย่างเช่น ถ้า $red_h = 0$ มุมก็จะเป็น

$$\frac{(240 * blue_h) + (120 * green_h)}{blue_h + green_h}$$

แซทูเลชัน (saturation) เป็นจำนวนของความเข้มของสีขั้นสุดท้าย ถ้าความเข้มสี เป็น 0 ในขั้นสุดท้ายก็จะไม่มีสี ตัวอย่างเช่น สีที่มาจากแสงขาวเท่านั้น ถ้าความเข้มสีเป็น 255 ก็จะไม่มีแสงขาวเพิ่มมาในสีสุดท้าย ค่าของความเข้มสีหาได้โดย

$$Saturation = \frac{\max(red, green, blue) - \min(red, green, blue)}{\max(red, green, blue)}$$

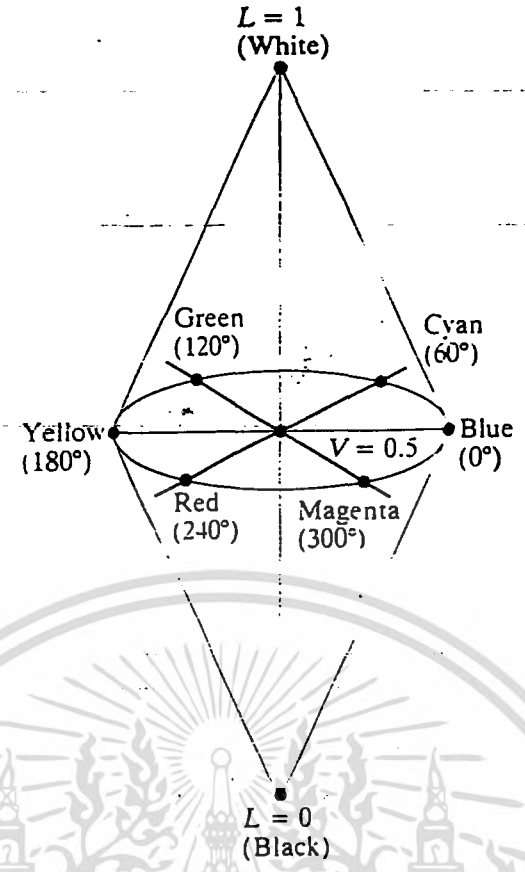
ค่าความสว่างเป็นเครื่องวัดของความเข้มของความสว่างของตัวประกอบสีกำหนดโดย

$$Value = \max(red, green, blue)$$

โมเดลดังกล่าวแสดงโดยใช้ กรวยฐานหกเหลี่ยม ดังรูปที่ 2.1

รูปที่ 2.1

โมเดล HLS พัฒนาโดยบริษัทเทคโทรนิค (Tektromix Incorporated) ใกล้เคียงกับโมเดลแบบ HSV ยกเว้นมุมของสีเริ่มที่ blue = 0° และโมเดลนี้จะเป็นแบบกรวยคู่ โดยมีความสว่างเป็นแกน จาก L = 0 (สีดำ) ถึง L = 1 (สีขาว) ดังรูปที่ 2.2



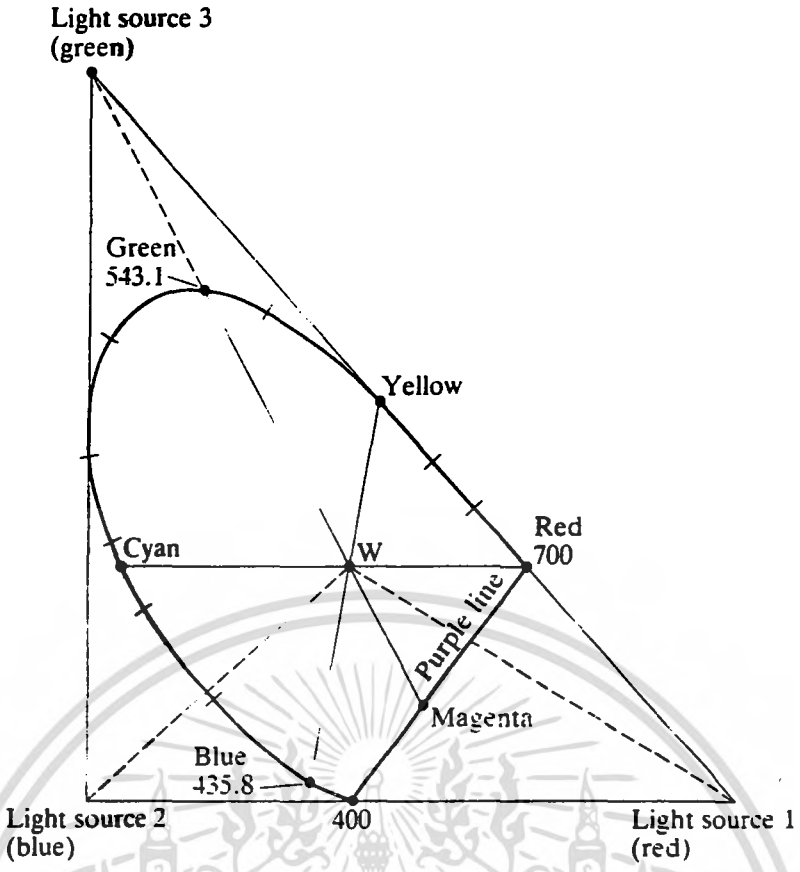
รูปที่ 2.2

สำหรับโมเดลแบบ HLS นั้น จะคำนวณสีได้แบบเดียวกับ HSV นอกจากให้ blue = 0° และหาความสว่าง (lightness) และ ความเข้มสี (saturation) ได้จากสมการดังนี้

$$lightness = \frac{\max(red, green, blue) - \min(red, green, blue)}{2}$$

$$Saturation = \frac{\max(red, green, blue) + \min(red, green, blue)}{\max(red, green, blue) - \min(red, green, blue)}; L \le 0.5$$

$$= \frac{\max(red, green, blue) - \min(red, green, blue)}{2 - \max(red, green, blue) - \min(red, green, blue)};$$ ถ้านอกจากนั้น



รูปที่ 2.3

รูปที่ 2.3 แสดงถึงไดอะแกรม 2 มิติที่กำหนดจุดที่แทนสีแดง เขียว และสีน้ำเงิน และจุดอื่น ๆ ที่เกิดจากการผสมสีของสีหลัก เพลตที่ 1 แสดงถึงลิมิตของหลอดรังสีคาโธดที่แสดงเซตย่อยของสีที่สามารถแสดงบนจอได้

2.3 ข้อมูลรูปภาพแบบบิตแมป

เนื่องจากในการจัดเก็บข้อมูลภาพสามารถเก็บข้อมูลได้หลายแบบ เช่น แต่ในปัญหาพิเศษนี้ ได้พัฒนาโปรแกรมสำหรับจัดการกับข้อมูลรูปภาพแบบบิตแมปเท่านั้น ดังนั้นในหัวข้อนี้จึงขอกล่าวเฉพาะรายละเอียดในส่วนของข้อมูลรูปภาพแบบบิตแมป ดังต่อไปนี้

2.3.1 ประเภทของบิตแมปบนวินโดวส์

บิตแมปบนวินโดวส์มีอยู่สองประเภทคือ

1. บิตแมปแบบ GDI (Graphic Device Interface)
2. บิตแมปแบบ DIBs (Device Independent Bitmaps)

ข้อมูลบิตแมปแบบ GDI จะขึ้นอยู่กับอุปกรณ์ (device-dependent) โปรแกรมต่างๆ สามารถได้รับการคัดลอกข้อมูลบิตแมป แต่การจัดเรียงของบิตขึ้นอยู่กับฮาร์ดแวร์แสดงผล ข้อมูลบิตแมปแบบ GDI สามารถถูกเคลื่อนย้ายอย่างอิสระระหว่างโปรแกรมบนเครื่องเดียวกัน แต่เนื่องจากการที่บิตแมปแบบนี้ขึ้นอยู่กับอุปกรณ์ การย้ายข้อมูลบิตแมปแบบ GDI โดยใช้ดิสก์หรือผ่านโมเด็มจึงไม่มีประโยชน์

24 สีบิตแมปด้วย และรูปแบบ DIB จะมีตัวเลือกสำหรับลดขนาดข้อมูลสำหรับบิตแมปแบบ 16 สี และ 256 สี การลดขนาดนี้จะช่วยให้ประหยัดเนื้อที่ดิสก์และหน่วยความจำอย่างมากเช่น บิตแมปสำหรับจอ VGA แบบ 16 สีที่ยังไม่ได้ลดขนาด จะใช้เนื้อที่ 154 KB แต่เมื่อลดขนาดแล้วจะใช้เนื้อที่เพียง 30 KB

ดังนั้นถ้าต้องการเก็บบิตแมปไว้ในไฟล์ข้อมูลที่เครื่องคอมพิวเตอร์เครื่องอื่นสามารถอ่านได้ จึงควรจะใช้รูปแบบที่เรียกว่าดีไวซ์อินดิเพนเดนต์ (device-independent) รูปแบบ DIB บนวินโดวส์เป็นสิ่งที่ดีอันหนึ่งเนื่องจากมี Windows API สนับสนุนอยู่ แอพพลิเคชันส่วนมากจะจัดการไฟล์ในรูปแบบ DIB (ไฟล์ BMP)

2.3.2 ความเข้าใจในข้อมูลรูปภาพแบบบิตแมป

ตัวอย่างบิตแมปอย่างง่ายคือบิตแมปที่เป็นเอกรงค์ คือมี 2 สีคือสีขาวกับสีดำ อิมเมจที่เห็นจะประกอบด้วยพิกเซล ในรูปที่เป็นขาวดำ พิกเซลจะเป็นแบบ on กับ off

ในความเป็นจริง บิตแมปจะเป็นเมตริกซ์ 2 มิติของพิกเซล ภาพบิตแมปส่วนใหญ่จะแสดงผลบนจอภาพหรือไม่ก็พิมพ์ออกมาเป็นภาพ บิตแมปจะประกอบด้วยสแต็กของสแกนไลน์ (stack of scanline) หนึ่งสแกนไลน์คือหนึ่งแถวตามแนวนอนของพิกเซล เมื่อต้องการแสดงบิตแมปบนจอภาพนั้น แต่ละสแกนไลน์ของบิตแมปจะถูกจัดการโดยสแกนไลน์ของจอภาพ ซึ่งจะแสดงผลลงหน้าจอทีละสแกนไลน์ ถ้าเป็นเครื่องพีซีคอมพิวเตอร์ ก็ไม่ใช่บิตแมปแต่จะเป็นไบต์แมป (byte map)

แต่ละพิกเซลในโมโนโครมบิตแมป จะสามารถแสดงได้ทางเดียวเรียกว่า "ซิงเกิลอิมเมจเพลน (single image plane)" สีของบิตแมปสามารถแสดงได้หลายอย่างที่แตกต่างกัน วิธีที่แสดงสีส่วนใหญ่คือแต่ละพิกเซลจะใช้ค่าสี RGB (ย่อมาจาก Red-Green-Blue) สามสีพื้นฐานนี้ใช้ในวีดีโอมอนิเตอร์ (Video Monitor) โดยกำหนดเปอร์เซนต์ที่เหมาะสมของสีแดง, เขียว, น้ำเงิน ซึ่งจะผสมกันเป็นสีที่มองเห็นได้ตามต้องการ

ในสี RGB แต่ละพิกเซลจะกำหนดไว้สีละ 8 บิตหรือ 1 ไบต์ เพราะฉะนั้นรวมแล้วจะเป็น 24 บิตที่จะบอกรายละเอียดของสีของแต่ละพิกเซล ดังนั้นจึงเรียก 24บิตสี (24-bit color) ในการที่จะวาดใหม่อีกครั้งของสีชนิดนี้ไฟล์จะใหญ่มากรูปขนาด 640x480 พิกเซลใช้ 24 บิตสี จะต้องการหน่วยความจำ 900 K ในการเก็บ แต่ในปัจจุบันนี้จะมีการ์ดซุปเปอร์วีจีเอ (card super VGA) ซึ่งเป็นฮาร์ดแวร์ที่ควบคุมอิมเมจของสี ซึ่งเรียกว่า higt-color หรือ 16 บิตสี และสามารถแสดงสีแบบ 24 บิต แม้ว่าจะมีบางสีหายไปบ้าง โดยการ์ดแสดงสีส่วนใหญ่ได้ออกแบบมาสำหรับเครื่องพีซี

สำหรับ 16 สีวีจีเอกราฟฟิกโหมด (16-color VGA graphicmode) แต่ละพิกเซลจะแทนด้วยตัวเลขซึ่งตัวเลขนี้จะอ้างอิงมาจากตารางค่าRGB ซึ่งเป็นจานสีของการ์ดแสดงผล (display

card) เลขแต่ละตัวจะมีค่าตั้งแต่ 0-15 คือมี 16 ค่าในจานสี ตัวอย่างเช่นถ้าจานสีแรกเป็นสีเขียว จะต้องใส่เลข 0 ในแต่ละตำแหน่งของหน้าจอที่ต้องการให้เป็นสีเขียว

256 สีบิตแมปแต่ละพิกเซลจะแทนได้ด้วย 1 ไบต์ ใน 1 ไบต์จะเก็บ 256 ค่าสีที่แตกต่างกันเหมือนกับมี 8 บิตสี เพราะว่าแต่ละพิกเซลจะแทนได้ด้วย 8 บิต ในบิตแมปจะสามารถมีสีได้ตั้งแต่ 4-128 สี สำหรับประโยชน์ที่ได้รับในกรณีของ 16 สีบิตแมป (4 บิตอิมเมจ)

วินโดวส์เก็บ 16 สีบิตแมป โดยใช้นิบเบิลสแต็กกิ้ง (nibble stacking) โดย 1 นิปเบิล คือ 4 บิต หรือ ถ้าเป็น 2 บิต ก็จะเรียกว่า ครัมซ์ (crumb) ข้อมูลของบิตแมปเก็บพิกเซลแรกในเส้นเหมือนเป็น 4 บิตแรกในไบต์แรก พิกเซลที่ 3 เก็บใน 4 บิตบนไบต์ที่สอง และจะเป็นอย่างนี้เรื่อย ๆ ค่าแต่ละ 4 บิตจะแทนด้วยค่าอ้างอิงใน 16 สีวิโดวส์พาเลต (16-colar windows pallet) การเก็บอิมเมจด้วย 4 บิตมีข้อเสีย 2 อย่างคือ

1. ไม่ยืดหยุ่น คือ ถ้าเป็น 32 สีบิตแมปอินเมจ ก็ต้องเก็บให้เป็น แบบ 256 สี เพราะว่าไม่มีการใช้แบบ 5 บิตนิปเบิล

2. ไม่ได้อยู่ในรูปเดียวกับการ์ดการแสดงผลของเครื่องพีซี ใช้เก็บความจำตรงส่วนจอภาพและจะต้องผ่านการแปลงค่าก่อนการแสดงผล

ปัญหาดังกล่าวไม่ได้เป็นจริงทั้งหมด ในการ์ดวีจีเอรุ่นใหม่ ๆ จะช่วยจัดการปัญหาตรงนี้ได้

ในการจัดการบิตแมปอิมเมจจะทำเป็นระนาบการแสดงผล ทั้งอิมเมจจะเก็บเป็นหลาย ๆ โมโนโครมอิมเมจเพลน โดยที่ 1 เพลนสำหรับแต่ละบิตของสี, 16 สีอิมเมจมี 4 เพลน แต่ละพิกเซลใน 1 อิมเมจจะประกอบด้วยตัวเลข 4 บิต (4 บิตจะแทนเป็น 1 ไบต์/เพลน) ตัวอย่างเช่น ในการทำงานกับสีของพิกเซลแรก ๆ ในมุมซ้ายบนของอิมเมจจะต้องใช้บิตแรกของเพลนที่หนึ่งและให้ค่าเวท (weight) เป็น 2 บิตแรกในเพลนที่สาม ให้ค่าเวท (weight) เป็น 4 ไปเรื่อย ๆ

2.3.3 **โครงสร้างของไฟล์** BMP

โครงสร้างของไฟล์ BMP จะประกอบไปด้วยข้อมูลเฮดเดอร์ (header) ข้อมูลพาเลตต์และข้อมูลภาพตามลำดับ

ข้อมูลเฮดเดอร์ คือ ข้อมูลที่อยู่บริเวณส่วนหัวของไฟล์ ซึ่งประกอบด้วยข้อมูลที่บอกถึงรายละเอียดของภาพ เช่น ความกว้างยาวของภาพ จำนวนสี จำนวนบิต ความละเอียด ฯลฯ

ข้อมูลพาเลตต์ คือ ข้อมูลที่บอกถึงชุดของจานสี (palette) ที่เกิดจากการผสมแม่สีทั้งสาม คือ สีแดง (Red) สีเขียว (Green) สีน้ำเงิน (Blue) มาผสมกันได้เป็นสีต่าง ๆ ตามจำนวนสีของภาพ เช่นรูปขนาด 4 บิต ก็จะมี 16 สี รูป 8 บิต จะมี 256 สี รูป 24 บิต จะมี 16.7 ล้านสีเป็นต้น ซึ่งถ้ามีจำนวนสีน้อย ๆ ก็จะมีการเก็บค่าพาเลตต์นี้ลงไฟล์ไปด้วย แต่ถ้าเป็นรูปประเภท 24 บิตจะไม่มีค่าพาเลตต์แต่จะใช้วิธีเก็บค่าแม่สีทั้งสามลงไปเป็นข้อมูลแทนเพราะถ้าเก็บค่าพาเลตต์

ที่มีถึง 16.7 ล้านสีลงไปด้วยจะเปลืองพื้นที่มาก ไฟล์บีเอ็มพีนี้จะเก็บค่าพาเลตต์ ชุดละ 4 ไบต์ซึ่งในฟอร์แมตอื่นๆจะใช้กันแค่ 3 ไบต์โ ดย 4 ไบต์ที่ว่านี้ก็ใช้แค่ 3 ไบต์เช่นกัน คือ สีแดง (Red), สีเขียว (Green), สีน้ำเงิน (Blue) อย่างละไบต์ ส่วน ไบต์ที่ 4 นั้นไม่ได้ใช้

ข้อมูลภาพ คือ ข้อมูลสีของภาพแต่ละจุดบนจอที่มาประกอบกันเป็นรูปภาพ ซึ่งค่าที่เก็บนี้จะเป็นค่าที่ใช้ในการชี้ตารางพาเลตต์ว่าจุดนี้จะใช้สีหมายเลขอะไร เช่นจุดแรกมีค่าเป็น 10 ก็ไปเปิดตารางพาเลตต์ หมายเลข 10 สมมติว่าได้ความเข้มแม่สีเป็น R = 0 , G = 0, B = 100 ก็จะได้จุดนี้เป็นสีน้ำเงิน ซึ่งในกรณีของรูป 24 บิต จะเป็นการอ่านข้อมูลขึ้นมา 3 ค่าเป็นค่าของแม่สีอาร์จีบี แล้วนำไปผสมบนจอแทน

ข้อมูลภาพของบีเอ็มพี จะมีการจัดเก็บอยู่ 2 แบบ คือ แบบบีบข้อมูลกับแบบไม่ได้บีบข้อมูล แบบแรกนั้นจะมีวิธีการบีบอยู่ 2 แบบ คือ อาร์แอลอี 4 และ อาร์แอลอี 8 ซึ่งก็คือการบีบข้อมูลแบบรันเลงท์ (Run-Length Encoder) แบบ 4 บิตและ 8 บิตนั่นเอง ส่วนแบบที่สองนั้นเก็บข้อมูลจริงๆลงไฟล์ ซึ่งจะทำให้ขนาดไฟล์ค่อนข้างใหญ่ แต่จะทำการแสดงภาพได้เร็วกว่าเพราะไม่ต้องเสียเวลาคลายข้อมูล

บทที่ 3
การเขียนโปรแกรมเชิงวัตถุ

3.1 ความหมายของการโปรแกรมเชิงวัตถุ

การโปรแกรมเชิงวัตถุ (Object-Oriented Programming หรือ OOP) เป็นวิธีการเขียนโปรแกรมแบบหนึ่งที่อ้างอิงแนวความคิดแบบเชิงวัตถุ (Object Oriented) โดยจะมองโปรแกรมเป็นระบบที่น่าสนใจ และมีออบเจ็คเป็นสมาชิกในโปรแกรม ในออบเจ็คหนึ่งจะประกอบไปด้วยส่วนโค้ดหรือรูทีนจัดการต่าง ๆ และส่วนดาต้าได้แก่ ตัวแปรในโปรแกรม

การเขียนโปรแกรมเชิงวัตถุนั้น โปรแกรมมีนิยามเป็นเซตของวัตถุ (object) โดยที่วัตถุจะถูกกำหนดให้เป็นหน่วยใหม่ที่สร้างขึ้นมา เพื่อรวมทั้งส่วนข้อมูลและโค้ดที่จัดการกับข้อมูลนั้นไว้ด้วยกัน

3.2 แนวความคิดใหม่ของการโปรแกรมเชิงวัตถุ

เพื่อแก้ปัญหาและจุดอ่อนของการพัฒนาโปรแกรมแบบโครงสร้าง แนวความคิดใหม่ที่เพิ่มขึ้นที่น่าสนใจของการพัฒนาโปรแกรมเชิงวัตถุมีหลายอย่าง เช่น วัตถุ (Object) การสืบทอด (Inheritance) พหุลักษณ์ (Polymorphism) ตัวสร้าง (Constructor) ตัวทำลาย (Destructor) เป็นต้น

3.2.1 วัตถุ (Object)

เป็นลักษณะการวิเคราะห์งานของการพัฒนาโปรแกรมเชิงวัตถุที่พิจารณาสิ่งต่าง ๆ เป็นวัตถุ โดยแต่ละวัตถุจะมีอิสระในการทำงานและตัดสินใจภายในตัวเอง วัตถุประกอบไปด้วย 2 ส่วน คือ ข้อมูล (data) และรหัส (code) อยู่รวมกัน การทำงานภายในวัตถุ รหัสจะเป็นตัวควบคุมการประมวลผลของข้อมูล การใช้วัตถุจะทำได้โดยการส่งแมสเสจ (Message) ไปให้กับวัตถุ วัตถุจะนำแมสเสจนั้นไปตีความเพื่อปฏิบัติงาน

การสร้างวัตถุของการพัฒนาโปรแกรมเชิงวัตถุ ในภาษา C++ จะสร้างอยู่ในรูปแบบที่เรียกว่า คลาส (classes) โดยชั้นจะเป็นการกำหนดรูปแบบของฟังก์ชั่นหนึ่งที่เรียกว่า atd (abstract data type) ที่เหมือนกับการกำหนดรูปแบบข้อมูลชนิดโครงสร้างในภาษาซี (Structure) เพียงแต่ว่า ในชั้นสามารถมีข้อมูลและฟังก์ชั่นจะอยู่รวมกันได้ แต่ในรูปแบบข้อมูลชนิดโครงสร้างจะมีเฉพาะข้อมูลอย่างเดียวเท่านั้น

<u>รูปแบบ</u>

```
class  class-name{
        type-member:
                data and function
        }object-list;
```

class-name หมายถึง ชื่อของชั้น

type-member หมายถึง ประเภทของสมาชิกที่อยู่ในชั้นมี 3 ประเภท คือ

Private เป็นการบอกให้รู้ว่าข้อมูลและฟังก์ชั่นที่อยู่ในส่วนนี้ จะใช้ได้เฉพาะภายในชั้นเท่านั้น ยกเว้นชั้นที่เป็นเพื่อน (friend) กันที่สามารถเรียกใช้ได้ ถ้าไม่บอกประเภทสมาชิก จะถือว่าเป็นแบบ Private ทั้งหมด

Public เป็นการบอกให้รู้ว่าข้อมูลและฟังก์ชั่นที่อยู่ในส่วนนี้ สามารถถูกเรียกใช้จากฟังก์ชั่นอื่นได้

Protected เป็นการบอกให้รู้ว่าข้อมูลและฟังก์ชั่นที่อยู่ในส่วนนี้ สามารถที่จะถ่ายทอดคุณสมบัติให้กับชั้นอื่นได้

Object-list หมายถึง ตัวแปรชั้นที่มีโครงสร้างเหมือนกับชื่อของชั้นที่ประกาศไว้ จะมีหรือไม่มีก็ได้ ถ้าไม่มีก่อนการเรียกใช้ชั้น ต้องมีการประกาศตัวแปรชั้นก่อน โดยใช้รูปแบบ

```
class-name object-list1, object-list2,...;
```

3.2.2 การสืบทอดคุณสมบัติ (Inheritance)

เป็นคุณสมบัติที่สำคัญของวัตถุคุณสมบัติหนึ่งคือวัตถุหนึ่งจะสามารถสืบทอดคุณสมบัติไปยังวัตถุอื่น ๆ ได้ โดยวัตถุที่มีชั้นสูงกว่าจะถ่ายทอดคุณสมบัติให้กับวัตถุที่มีชั้นต่ำกว่าเสมอ การสืบทอดคุณสมบัติทำให้สามารถสร้างวัตถุใหม่ขึ้นได้ และวัตถุใหม่สามารถใช้คุณสมบัติของวัตถุเก่าได้อีกด้วย ในการพัฒนาโปรแกรม คุณสมบัตินี้ทำให้สามารถนำส่วนของโปรแกรมเก่ามาใช้ใหม่ได้โดยไม่ต้องทิ้งส่วนของโปรแกรมเก่า ทำให้เป็นการประหยัดเวลาและลดความซ้ำซ้อนของระบบโปรแกรม

จากลักษณะที่มีการถ่ายทอดคุณสมบัติจากชั้นหนึ่งไปยังชั้นอื่น ๆ ได้ ชั้นที่มีการถ่ายทอดคุณสมบัติ เรียกว่า เบสคลาส (base class) หรือพาเลนท์คลาส (parent class) ส่วนชั้นที่รับคุณสมบัติที่ถ่ายทอดมาเรียกซับคลาส (subclass) หรือ ดีไรฟ์ คลาส(derived class)

3.2.3 พหุลักษณ์ (Polymorphism)

เป็นคุณสมบัติที่สำคัญอีกประการหนึ่ง ของการพัฒนาโปรแกรมแบบโอโอพี ที่สามารถทำให้ชื่อเพียงชื่อเดียว สามารถทำงานได้หลายอย่าง ขึ้นกับชนิดของข้อมูลที่ส่งมา เช่น การตั้งชื่อฟังก์ชัน pow เพื่อให้ใช้ในการหาค่ายกกำลังของข้อมูล 3 ชนิด คือ ข้อมูลจำนวนเต็ม ข้อมูลทศนิยมขนาดใหญ่ และข้อมูลจำนวนเต็มขนาดใหญ่ จะทำได้โดยเขียนฟังก์ชั่นขึ้น 3 ฟังก์ชั่นให้มีชื่อเดียวกันแต่มีข้อมูลที่ส่งมาทำงานแตกต่างกัน ฟังก์ชันที่ชื่อซ้ำกันจะทำงานฟังก์ชันใดก็จะดูจากค่าที่ส่งมาให้ทำงาน ถ้าส่งมาเป็นจำนวนเต็ม ฟังก์ชัน pow ที่มีค่าข้อมูลเป็นจำนวนเต็มก็จะทำงานเป็นต้นจะทำให้สะดวกและประหยัดเวลาในการพัฒนาโปรแกรมขึ้นเป็นอย่างมาก

ในระบบแบบเชิงวัตถุ มีองค์ประกอบที่สำคัญที่สุดคือ คลาสไลบรารี่ (Class Library) ซึ่งประกอบไปด้วยคลาสต่างๆมากมายที่มีโอกาสจะกำหนดชื่อวิธีการของแต่ละคลาสซ้ำกันได้มาก สามารถแก้ไขได้ ด้วยการนำคุณสมบัติพหุลักษณ์เข้ามาช่วย จุดประสงค์ของพหุลักษณ์ คือ มีความต้องการเน้นความคิดที่ว่า " การโปรแกรมเชิงวัตถุที่ดีนั้น ควรที่สามารถตัดสินได้ว่า ควรใช้มาตรฐานของออบเจคต์ตัวไหนในการตอบสนองแมสเสจ ที่มีจุดประสงค์คล้าย ๆ กัน

3.2.4 ตัวสร้าง (Constructor)

เป็นวิธีการที่ทำให้ฟังก์ชันที่อยู่ในชั้นสามารถทำงานได้เองโดยอัติโนมัติก่อนมีการเรียกใช้ การกำหนดฟังก์ชันขึ้นให้มีลักษณะเป็นตัวสร้างทำได้โดย ต้องตั้งชื่อฟังก์ชัน ให้เป็นชื่อเดียวกับชื่อชั้น การทำงานของฟังก์ชันที่ประกาศเป็นตัวสร้างจะทำงานทุกครั้งที่มีการใช้คำสั่งประกาศตัวแปรชั้น

3.2.5 ตัวทำลาย (Destructor)

เป็นวิธีการทำให้ฟังก์ชันใดฟังก์ชันหนึ่งมีลักษณะการทำงานกลับกับฟังก์ชันที่เป็นตัวสร้าง คือฟังก์ชันที่เป็นชนิดตัวสร้างจะทำงานเมื่อเริ่มมีการประกาศตัวแปรชั้น แต่ฟังก์ชันชนิดตัวทำลายจะทำหลังจากจบการทำงานของโปรแกรมหลักเองโดยอัตโนมัติ การกำหนดฟังก์ชันให้เป็นชนิดตัวทำลายจะทำได้โดย ตั้งชื่อฟังก์ชันให้เป็นชื่อเดียวกันกับชั้น และใส่เครื่องหมาย ~ นำหน้าฟังก์ชั้น

3.3 ข้อดีของภาษาเชิงวัตถุ

เป้าหมายหลักของการเขียนโปรแกรมเชิงวัตถุคือ

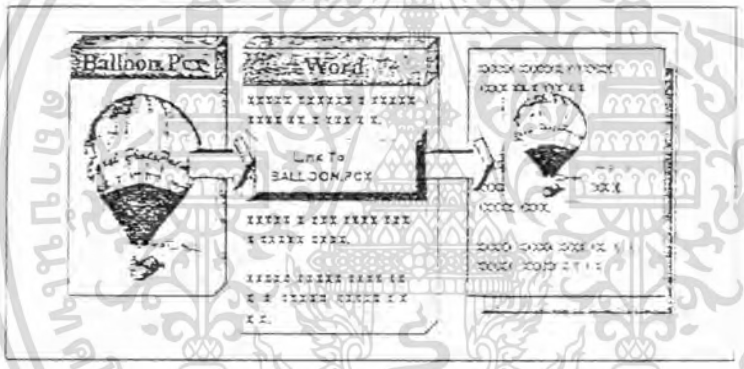
1. ทำให้การพัฒนาซอฟต์แวร์ใช้เวลาสั้น และต้นทุนต่ำลงโดยใช้คุณสมบัติคลาส ที่สามารถนำกลับมาใช้ได้อีก และสร้างซับคลาสขึ้นมาเพื่อนำมาใช้แก้ปัญหา
2. ทำให้ต้นทุนในการบำรุงรักษาซอฟต์แวร์ต่ำลงเพราะสามารถหาจุดที่ต้องการเปลี่ยนแปลงในซอฟต์แวร์ได้ และการเปลี่ยนแปลงไม่ทำให้เกิดผลกระทบไปยังภายนอกคลาสได้

บทที่ 4
การใช้เทคนิคโอแอลอี (OLE 2.0)

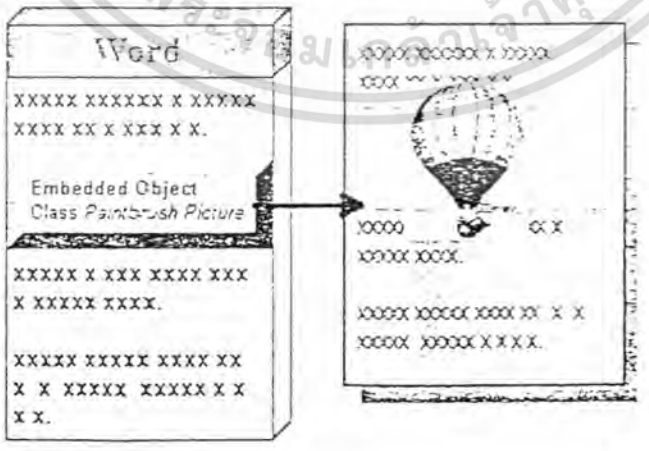
4.1 ความหมายของโอแอลอี

โอแอลอี (OLE : Object Linking and Embedding) เป็นเทคโนโลยีที่ใช้สร้างแอพพลิเคชันสมัยใหม่ โดยมีโอแอลอีเวอร์ชัน 2 (OLE 2.0) เป็นรุ่นล่าสุดที่มีอยู่ในขณะนี้ โอแอลอีนี้จะช่วยขยายความสามารถให้กับแอพพลิเคชันที่มีอยู่แล้ว ให้ใช้งานข้ามแพลตฟอร์มได้ อีกทั้งยังเป็นแนวทางในการพัฒนาระบบปฏิบัติการวินโดวส์ในรุ่นถัดไปอีกด้วย

โดยโอแอลอีจะจัดการกับสารสนเทศให้เป็นสารสนเทศที่มีเอกสารเป็นศูนย์กลาง (document centric) มากกว่าที่จะมีศูนย์กลางอยู่ที่ตัวแอพพลิเคชันอย่างในอดีต (application centric) ซึ่งเอกสารที่เป็นศูนย์กลางของสารสนเทศต่าง ๆ เรียกว่า "เอกสารเชิงซ้อน (compound document)" สร้างโดยการเชื่อมโยงออบเจกต์ (linking object) หรือฝังออบเจกต์ (embedding object) ลงในเอกสารนั้น ดังรูป ที่4.1 และ รูปที่ 4.2 ตามลำดับ



รูปที่ 4.1 แสดงเอกสารเชิงซ้อนที่สร้างโดยการเชื่อมโยงออบเจกต์

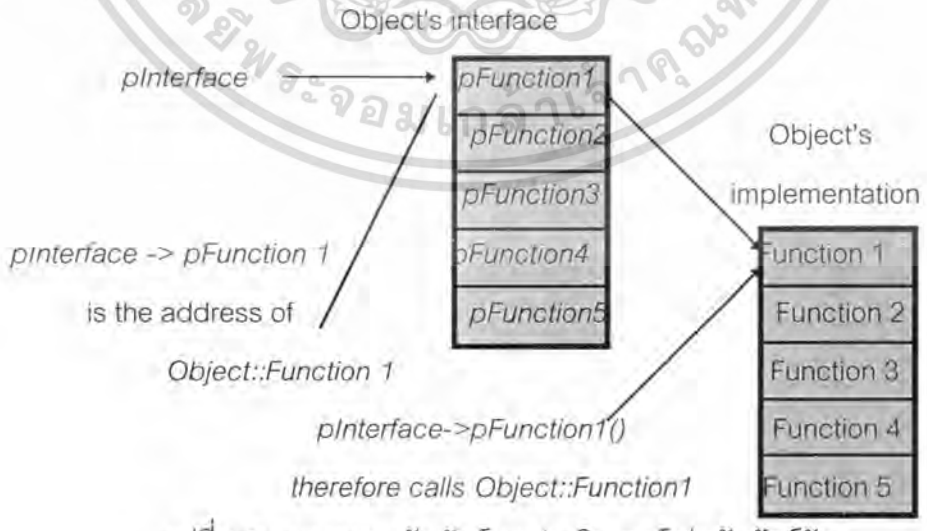


รูปที่ 4.2 แสดงเอกสารเชิงซ้อนที่สร้างโดยการฝังออบเจกต์

- การเชื่อมโยง เป็นกระบวนการที่สารสนเทศในไฟล์หนึ่งถูกนำไปใช้ในที่อื่น ๆ ซึ่งการเปลี่ยนแปลงสารสนเทศไม่ว่าจากที่ใด ๆ จะมีผลต่อสารสนเทศในที่อื่นเสมอ
- การฝัง เป็นกระบวนการที่สารสนเทศทั้งหมดถูกเก็บอยู่ในที่ที่ฝังตัวอยู่ ดังนั้นการเปลี่ยนแปลงสารสนเทศในที่หนึ่ง ๆ จะไม่มีผลต่อที่อื่น

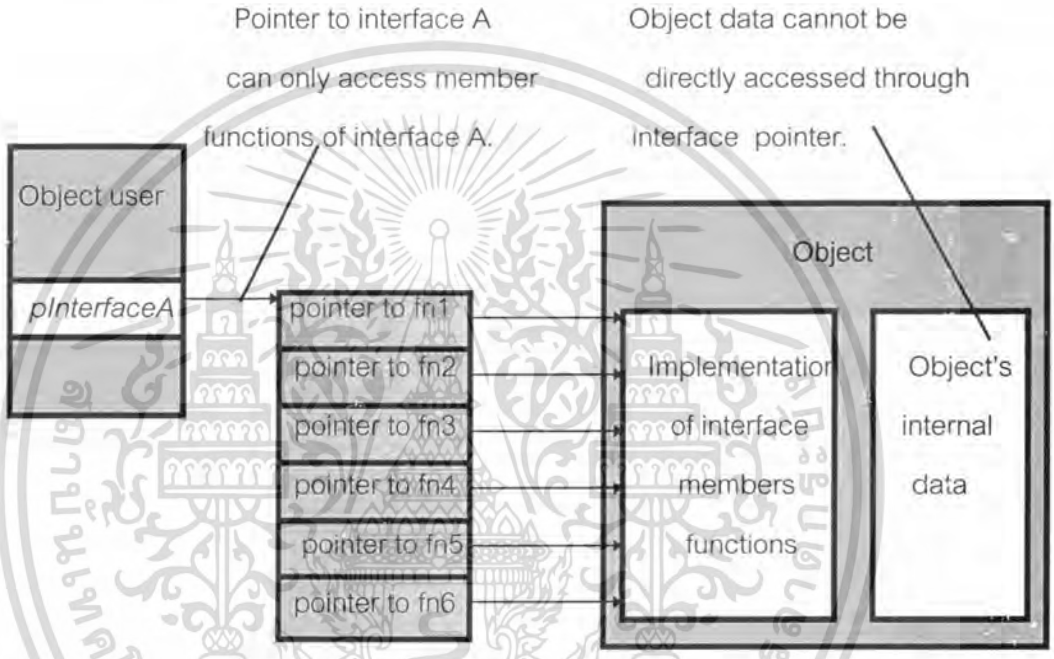
จากนิยามของการเชื่อมโยงออบเจกต์ และการฝังออบเจกต์ จะเห็นได้ว่าเอกสารที่เกิดจากการฝังออบเจกต์จะมีขนาดที่ใหญ่กว่าเอกสารที่เกิดจากการเชื่อมโยงออบเจกต์ และการเชื่อมโยง ออบเจกต์ทำให้รายการย่อยในออบเจกต์หนึ่งที่นำมาเชื่อมโยงนั้น สามารถนำไปใช้ในแอพพลิเคชันอื่นได้ แต่จะทำไม่ได้ในกรณีของรายการย่อยในออบเจกต์ที่เป็นการฝังตัวในเอกสาร เพราะจะติดต่อรายการเหล่านั้นได้ภายในเอกสารเท่านั้น เนื่องจากว่าการฝังออบเจกต์ลงในเอกสารต้องเก็บสารสนเทศทั้งหมดที่มีอยู่ลงในเอกสารดังนั้นจึงทำให้เอกสารที่เกิดจากการฝังออบเจกต์มีขนาดใหญ่กว่าเอกสารที่เกิดจากการเชื่อมโยงออบเจกต์ แต่จะมีจำนวนไฟล์ที่ใช้น้อยกว่า และจากความแตกต่างดังที่ได้กล่าวไปนี้ จะเป็นเครื่องช่วยในการตัดสินใจวิธีที่ใช้ในการจัดการกับออบเจกต์ ซึ่งไม่ว่าจะจัดการด้วยวิธีใดก็ตามแต่ภาพโดยรวมแล้วผลที่ได้ก็จะออกมาในทำนองเดียวกัน จะต่างกันก็เพียงประโยชน์ที่จะได้ในส่วนรายละเอียดเท่านั้น

ปัจจัยสำคัญ ที่ทำให้แอพพลิเคชันขยายขอบเขตการทำงานได้กว้างขึ้นก็คือ อินเตอร์เฟส (interface) เป็นมาตรฐานสำหรับการปฏิสัมพันธ์ (interaction) กันระหว่างออบเจกต์ โดยจะแทนได้ด้วยกลุ่มพอยน์เตอร์ที่ชี้ไปยังฟังก์ชันซึ่งเป็นการทำงานของแอพพลิเคชันนั้น ๆ ที่จะให้แอพพลิเคชันอื่นเรียกใช้งานได้ หรือเรียกอีกอย่างว่า "VTBL (virtual table)" ดังแสดงไว้ใน รูปที่ 4.3

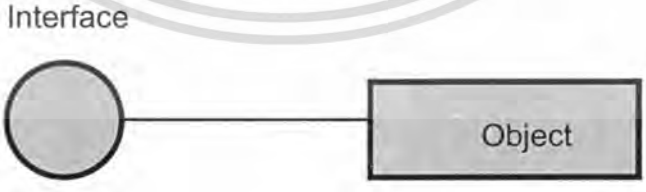


รูปที่ 4.3 แสดงความสัมพันธ์ระหว่างอินเตอร์เฟสกับฟังก์ชัน

หากแอพพลิเคชันใดที่ต้องการเรียกใช้ฟังก์ชันของแอพพลิเคชันอื่น จะต้องทำการอิมพลเมนท์อินเตอร์เฟสของแอพพลิเคชันที่ต้องการใช้งานจึงจะเข้าไปเรียกฟังก์ชันของแอพพลิเคชันนั้นได้ กล่าวคือผู้ใช้บริการ (Client) จะต้องมีพอยน์เตอร์ชี้ไปยังอินเตอร์เฟส และเรียกฟังก์ชันที่สมาชิกของ VTBL ชี้ไปนั้นว่า "วิธีการ (method) หรือ สมาชิกที่เป็นฟังก์ชันของอินเตอร์เฟส (member function)" ซึ่งก็คือฟังก์ชันที่นำไปใช้งานนั่นเอง ในหนังสือเกี่ยวกับโอแอลอีทั่ว ๆ ไปนิยมแทนตารางของฟังก์ชันด้วยวงกลมที่เชื่อมต่อกับออบเจกต์ หรือเรียกวงกลมดังกล่าวว่า "แจคก์ (jack)"



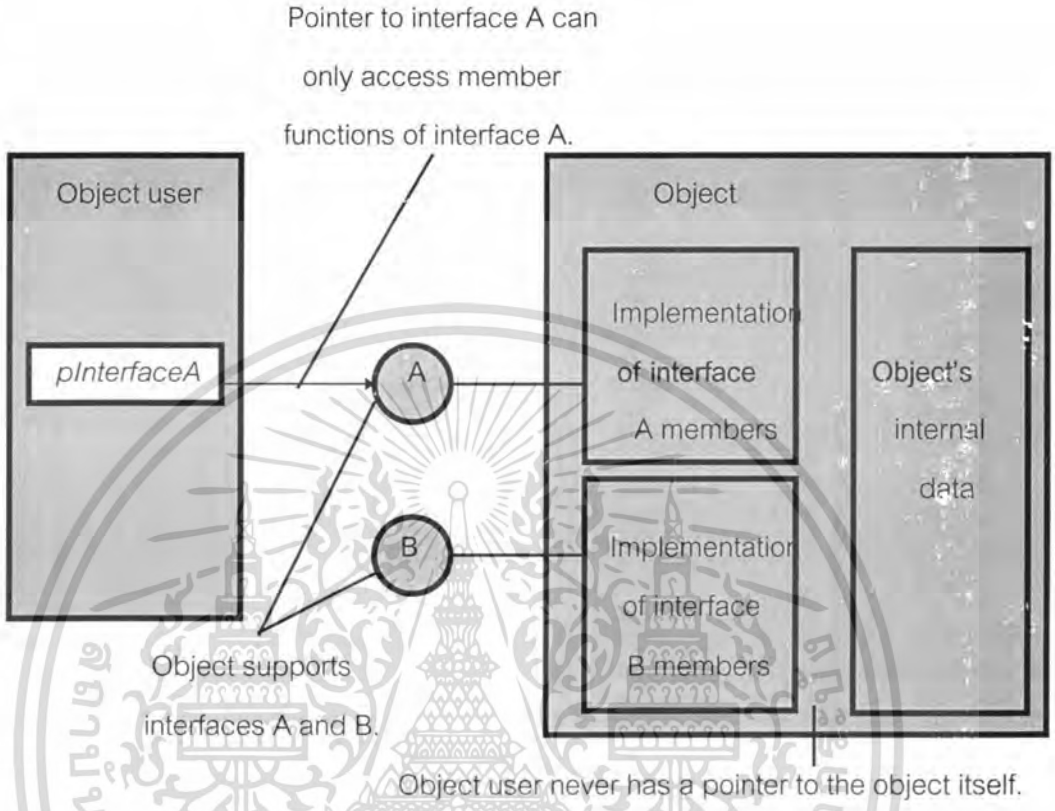
รูปที่ 4.4 แสดงวิธีการเรียกใช้งานฟังก์ชันผ่านอินเตอร์เฟส



A circle (or jack) is used to represent the entire function table for an interface.

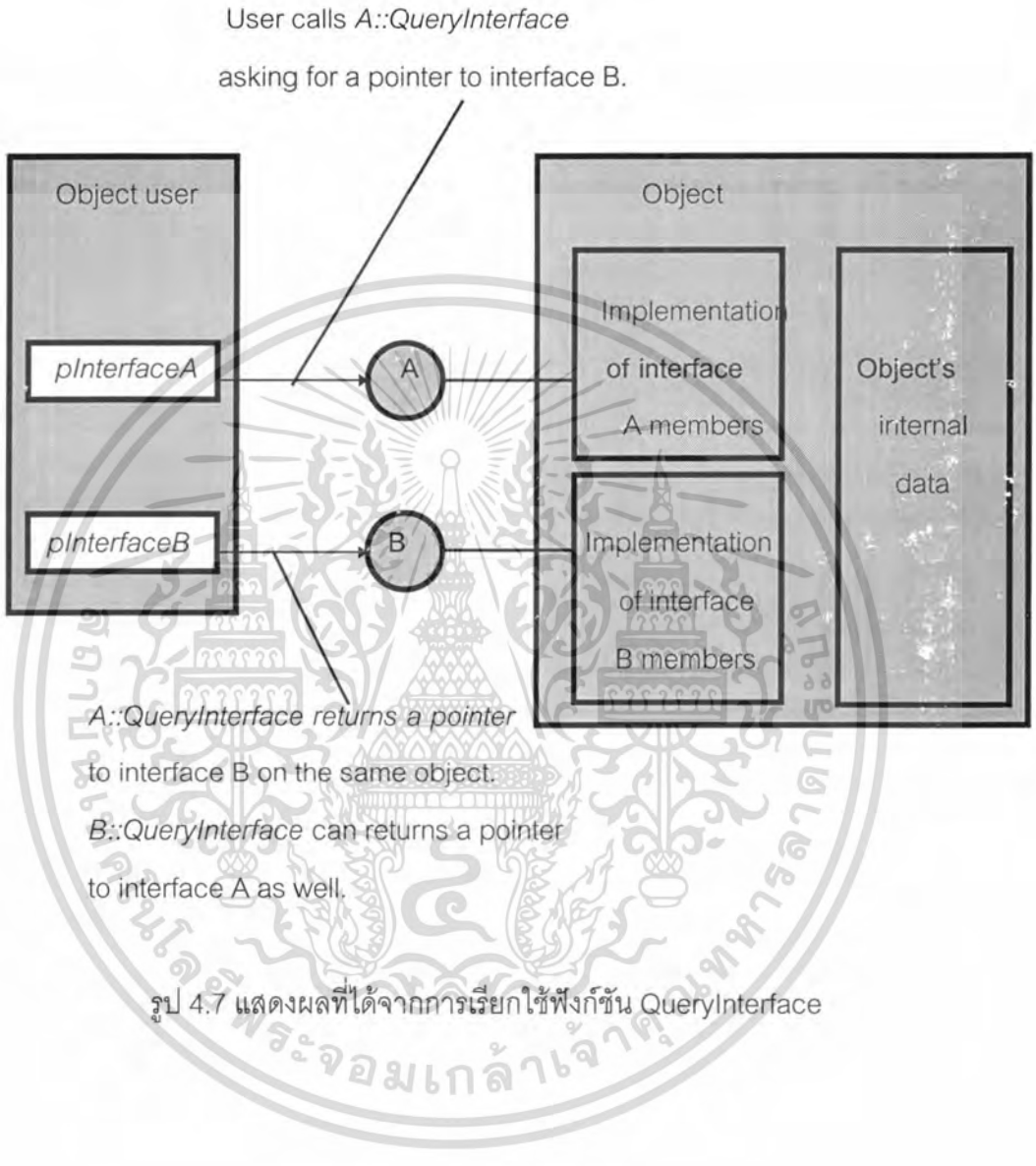
รูปที่4.5

และเนื่องจากที่พอยน์เตอร์ที่ชี้ไปยังอินเตอร์เฟสแต่ละอัน จะเรียกใช้ได้แต่ฟังก์ชันที่เป็นสมาชิกของอินเตอร์เฟสที่ชื้อยู่เท่านั้น ดังรูปที่ 4.6

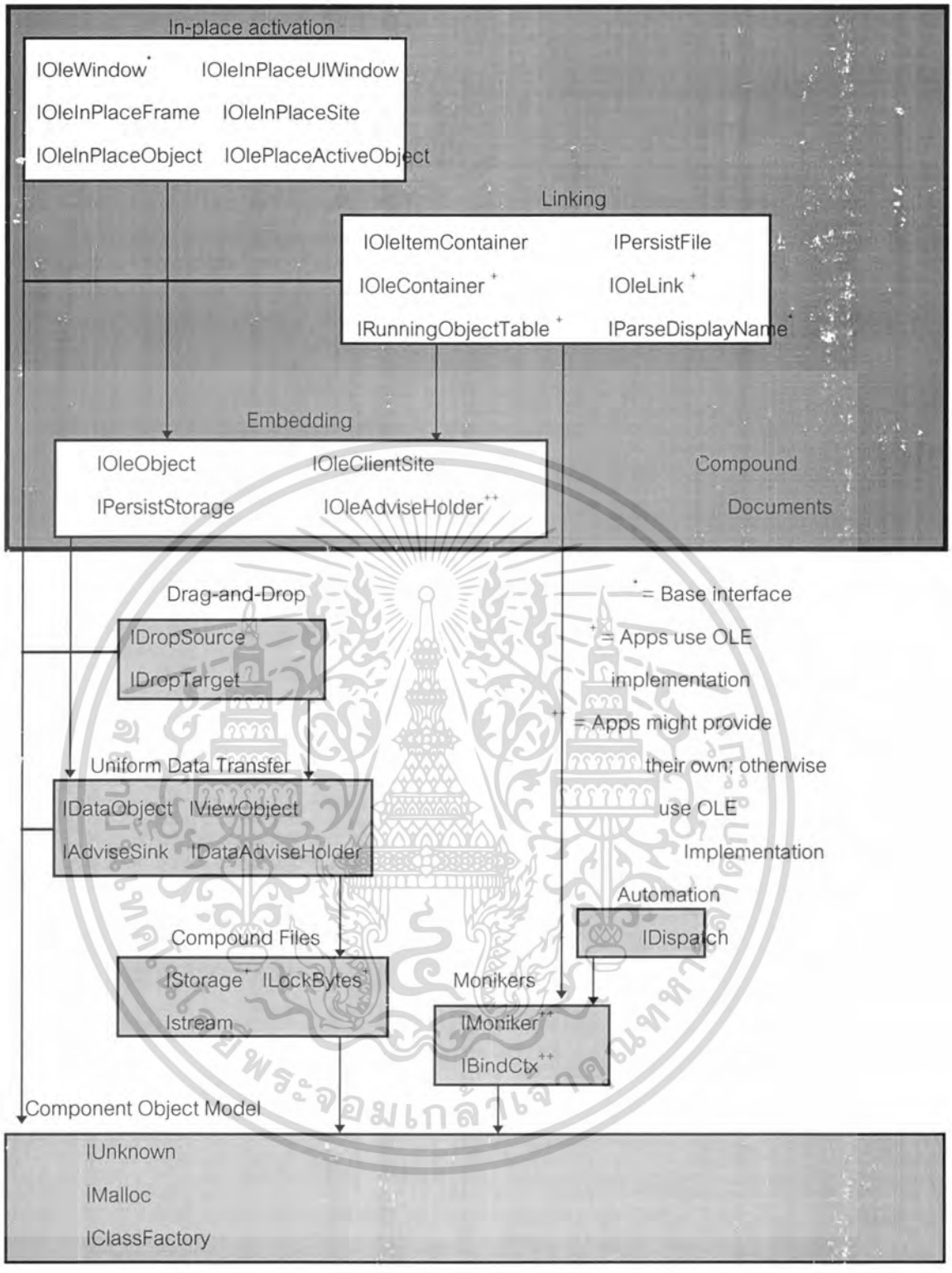


รูปที่ 4.6 แสดงการขอบเขตของการเข้าถึงฟังก์ชันของแต่ละพอยน์เตอร์ที่ชี้อินเตอร์เฟส

ในโอแอลอี มีฟังก์ชันมาตรฐานเพื่อให้ผู้ใช้อินเตอร์เฟสหนึ่งสามารถใช้งานอีกอินเตอร์เฟสในออบเจกต์เดียวกันได้ โดยการเรียกใช้ฟังก์ชันมาตรฐาน คือ QueryInterface ซึ่งจะคืนค่าเป็นพอยน์เตอร์ที่ชี้ไปยังอินเตอร์เฟส หากว่าออบเจกต์ที่กำลังทำงานสนับสนุนอินเตอร์เฟสนั้น (รูปที่ 4.7)



รูป 4.7 แสดงผลที่ได้จากการเรียกใช้ฟังก์ชัน QueryInterface

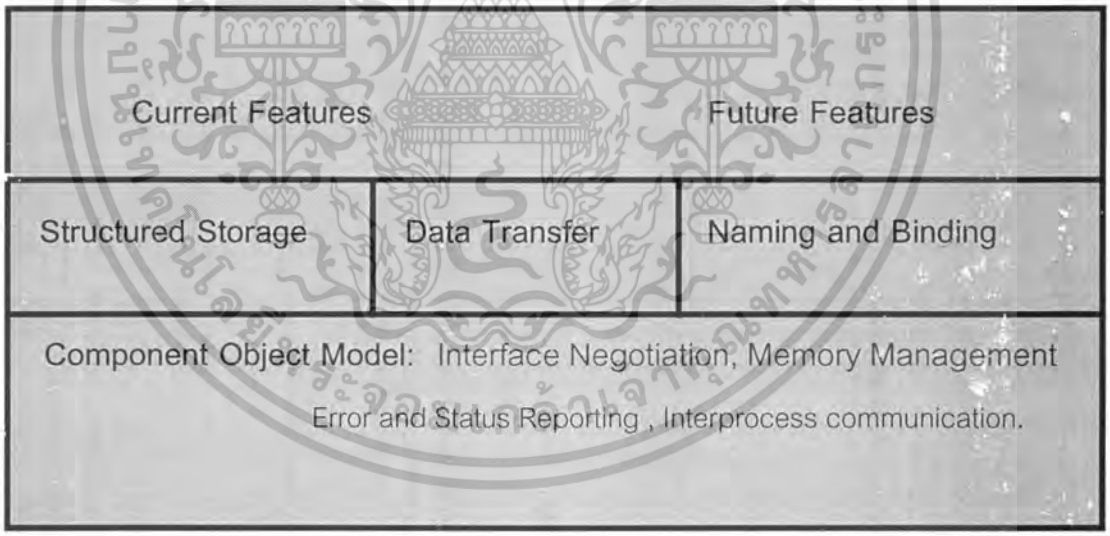


รูปที่4.8 แสดงอินเตอร์เฟสที่น่าสนใจในเทคโนโลยีของโอแอลอี 2.0

4.2 ลักษณะสำคัญของโอแอลอี (OLE features)

ในหัวข้อต่อไปนี้จะได้กล่าวถึงลักษณะสำคัญของโอแอลอี หรือ ฟีเจอร์ (feature) โดยต่อไปนี้จะใช้คำว่าฟีเจอร์ในการกล่าวถึงในส่วนของรายละเอียดต่อไป โดยจะแบ่งฟีเจอร์ที่มีอยู่ในโอแอลอี เวอร์ชัน 2.0 ออกเป็น 3 ฟีเจอร์ใหญ่ ๆ ได้ดังที่แสดงในรูปที่ 4.9

- ระดับล่าง เป็นรากฐานของฟีเจอร์อื่นๆ ของโอแอลอี คือตัวแบบของคอมโพเนนท์ออบเจกต์ (Component Object Model)
- ระดับกลาง ประกอบไปด้วย โครงสร้างของหน่วยสะสม (Structured Storage) , การส่งผ่านข้อมูล (Data Transfer) , การตั้งชื่อและการผูกออบเจกต์เข้าด้วยกัน (Nameing and Binding)
- ระดับสูง เป็นส่วนที่ช่วยเพิ่มประสิทธิภาพให้กับแอพพลิเคชันต่างๆ ประกอบไปด้วยฟีเจอร์ที่มีอยู่แล้วในปัจจุบัน อย่างเช่น การทำงานในสถานที่ (In-Place Activation) , การลากและวาง (Drag and Drop) และการทำงานอัตโนมัติ (Automation) กับฟีเจอร์ที่จะเกิดขึ้นต่อไปในอนาคต



รูปที่ 4.9 แสดงฟีเจอร์ในระดับต่าง ๆ ของ OLE

4.3 ระดับของฟีเจอร์ในโอแอลอี

4.3.1 ฟีเจอร์ระดับล่าง

จากที่ได้กล่าวไปข้างต้น จะเห็นได้ว่าตัวแบบของคอมโพเนนท์ออบเจกต์ (Component Object Model) เป็นฟีเจอร์ที่อยู่ล่างสุดของทุก ๆ ฟีเจอร์ที่มีอยู่ในเทคโนโลยีของโอแอลอี นั่นก็หมายความว่า ทุก ๆ ฟีเจอร์จะต้องเกิดจากฟีเจอร์ของตัวแบบของคอมโพเนนท์ออบเจกต์ ซึ่งเป็นสิ่งที่กำหนดวิธีในการติดต่อกันของออบเจกต์ในหนึ่งแอพพลิเคชัน หรือต่างแอพพลิเคชัน

ในตัวแบบนี้จะมีออบเจกต์ที่เรียกว่า "คอมโพเนนท์ออบเจกต์ (component object)" เป็นสิ่งที่ใช้ในการอิมพลีเมนท์อินเตอร์เฟส และทำการเรียกใช้อินเตอร์เฟสที่สนับสนุนการปฏิสัมพันธ์ของออบเจกต์

ด้วยเทคโนโลยีของโอแอลอี 2.0 จะใช้ตัวแบบของคอมโพเนนท์ออบเจกต์เป็นสิ่งที่ใช้กำหนด ฟีเจอร์ย่อย ๆ ดังต่อไปนี้

- ตัวแบบของคอมโพเนนท์ออบเจกต์จะกำหนดมโนภาพของอินเตอร์เฟสโดยใช้ผู้รับบริการของการติดต่อแบบมีการบริการ (service communication)
- ตัวแบบของคอมโพเนนท์ออบเจกต์จะทำให้ออบเจกต์ของผู้ใช้บริการ (Client) สามารถสนับสนุนอินเตอร์เฟสพิเศษ ด้วยออบเจกต์ที่มีอินเตอร์เฟสหลาย ๆ อินเตอร์เฟส
- ตัวแบบของคอมโพเนนท์ออบเจกต์จะกำหนดตัวแบบของตัวนับการอ้างอิง (reference counting) ซึ่งจะนำไปใช้ในเรื่องการจัดการกับออบเจกต์ ที่อาจมีการเรียกใช้จากผู้ใช้บริการหลาย ๆ ตัว พร้อมกัน
- ตัวแบบของคอมโพเนนท์ออบเจกต์ จะกำหนดเทคนิคที่ใช้ในการจองเนื้อที่ในหน่วยความจำ และการปล่อยหน่อยความจำเมื่อเลิกใช้ (Memory Management)
- ตัวแบบของคอมโพเนนท์ออบเจกต์จะกำหนดตัวแบบที่ใช้ในสถานะและความผิดพลาด
- ตัวแบบของคอมโพเนนท์ออบเจกต์จะกำหนดเทคนิคที่ใช้ ในการสื่อสารผ่านขอบเขตของ โปรเซส (process boundaries)
- ตัวแบบของคอมโพเนนท์ออบเจกต์จะกำหนดวิธีที่จะชี้ออบเจกต์พิเศษที่ได้อิมพลีเมนท์ไว้ และโหลด (load) แบบไดนามิก (dynamically) เข้าไปยังระบบที่ทำงานอยู่ โดยใช้อิมพลีเมนท์ออบเจกต์พิเศษที่อยู่ใน DLL (Dynamic Link Library) หรือ EXE

4.3.2 ฟีเจอร์ระดับกลาง

ประกอบไปด้วยฟีเจอร์ที่เป็นเรื่องของ การถ่ายโอนข้อมูล, ตัวแบบของโครงสร้างของหน่วยสะสม, การตั้งชื่อและการผูกออบเจกต์เข้าด้วยกัน

- การถ่ายโอนข้อมูล (Data Transfer)

ในโอแอลอี 2.0 มีเทคนิคที่ใช้ในการถ่ายโอนข้อมูลดังต่อไปนี้ คือคลิปบอร์ด (Clipboard), DDE, ลากและวาง (Drag and Drop), และ โอแอลอี 1.0 นอกจากนี้ยังมีออบเจกต์ที่ใช้จัดการเกี่ยวกับการถ่ายข้อมูลคือ ดาต้าออบเจกต์ (data object) เป็นออบเจกต์ที่เก็บข้อมูล ที่ไม่เพียงแต่จะอยู่ในรูปแบบของคลิปบอร์ดเท่านั้น แต่ยังสามารถอยู่ในรูปแบบของอุปกรณ์เป้าหมาย (target device) อีกทั้งยังอาจเป็นสิ่งที่กำลังแสดงอยู่ หรืออาจเป็นอุปกรณ์ที่ใช้เก็บข้อมูลนั้น ๆ

ตัวกลางที่ใช้ในการถ่ายโอนข้อมูล ก็ไม่จำเป็นต้องใช้แต่หน่วยความจำส่วนรวมเท่านั้น (global memory) ยังสามารถถ่ายโอนข้อมูลผ่านหน่วยสะสม (storage) บนดิสก์ รวมทั้งยังสามารถถ่ายโอนข้อมูลในเทรดดิชันไฟล์ (tradition file) หรือในสตอเรจ / สตรีม ออบเจกต์ ได้อีกด้วย

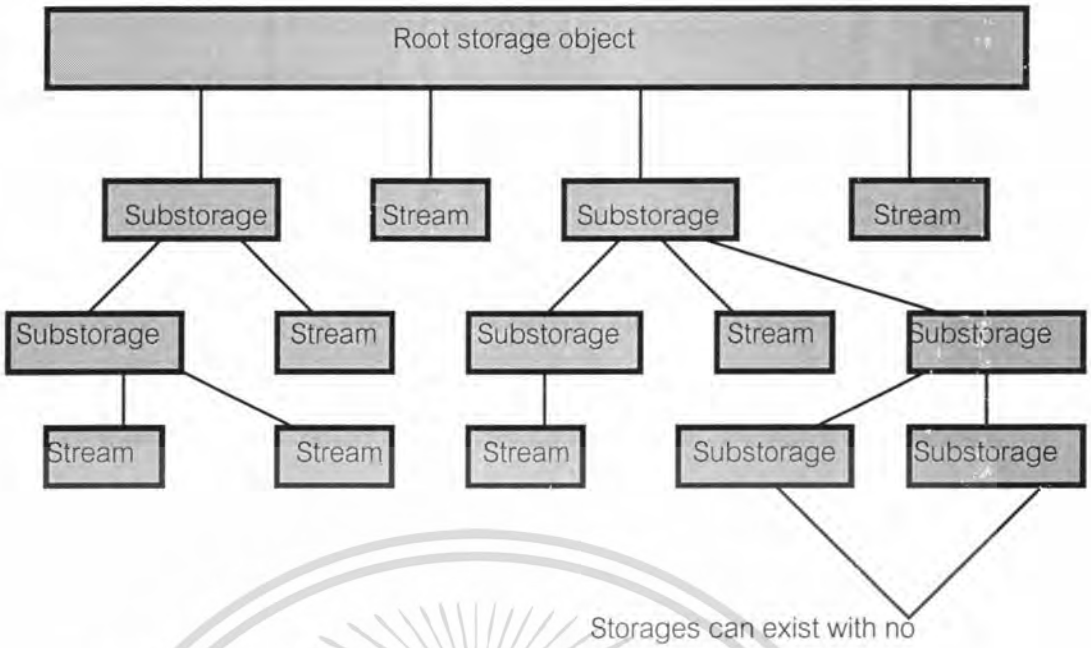
ด้วยวิธีการถ่ายโอนข้อมูลโดยใช้ดาต้าออบเจกต์นี้เอง ทำให้สามารถแก้ปัญหาที่เกิดขึ้นในอดีต (ก่อนการใช้เทคโนโลยีโอแอลอี 2.0) ที่เป็นเรื่องของข้อจำกัดของขนาดของหน่วยความจำส่วนรวมที่ใช้เป็นตัวกลาง, ข้อจำกัดในการใช้โปรโตคอล เป็นต้น

- โครงสร้างของหน่วยสะสม (Sturcture Storage Model)

เป็นตัวแบบที่กล่าวถึงการบันทึกข้อมูล และการเรียกใช้ข้อมูลจากหน่วยสะสม (retrieve) โดยจะอิมพลีเมนท์ตัวแบบนี้ด้วยไฟล์เชิงซ้อน (Compound file) ดังแสดงไว้ในรูปที่ 10. ซึ่งเป็นตัวจัดการเนื้อที่ว่างที่มีอยู่ โดยอยู่ในรูปของออบเจกต์ 2 ชนิดคือ สตรีมออบเจกต์ (stream object) และสตอ เรจออบเจกต์ (storage object)

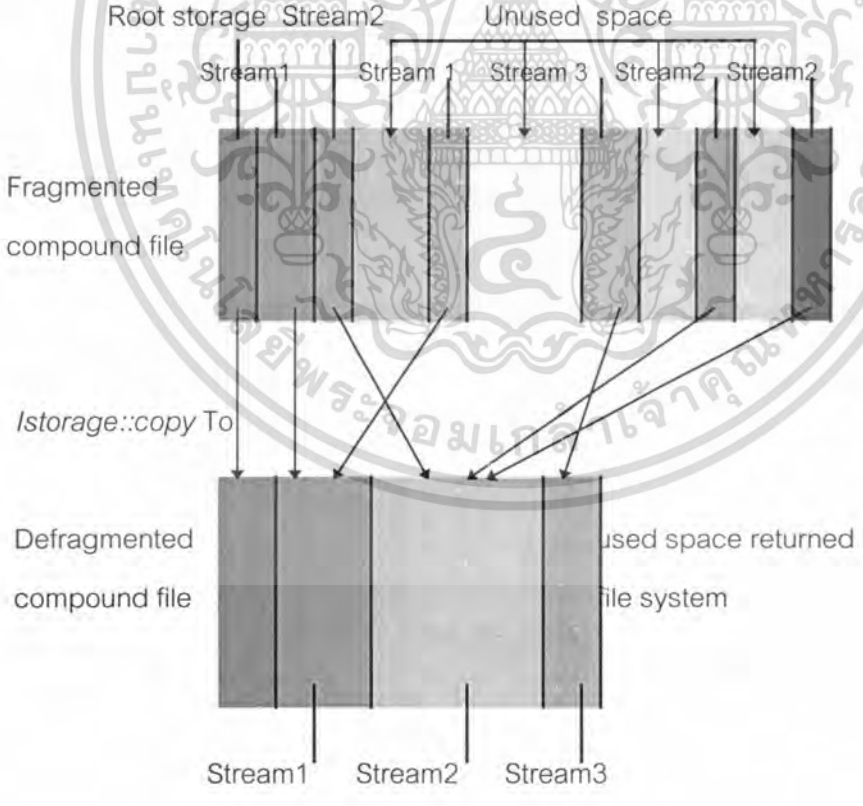
- สตรีมออบเจกต์ มีความหมายเหมือนกับไฟล์ในระบบปฏิบัติการดอส เป็นส่วนประกอบพื้นฐานซึ่งเป็นที่อยู่ของข้อมูล

- สตอเรจออบเจกต์ มีความหมายเหมือนกับไดเรกทอรีในดอส โดยแต่ละสตอเรจจะมีสตอเรจย่อย และสตรีม ได้มากเท่าที่ดิสก์จะมีเนื้อที่ว่างให้เก็บ แต่ในสตอเรจจะไม่มีการเก็บข้อมูลที่จะนำไปใช้งาน เก็บแต่เพียงข้อมูลของโครงสร้างของสตอเรจในระดับถัดไป



รูปที่ 4.10 แสดงตัวอย่างของไฟล์เชิงซ้อน

ไฟล์เชิงซ้อนมีความสามารถที่โดดเด่นในการกำจัดแฟรกเมนท์ (defragmentation) ด้วยวิธีการง่าย ๆ ดังรูปที่ 4.11 อีกทั้งยังนำไฟล์เชิงซ้อนไปใช้ในการถ่ายโอนข้อมูลได้อีกด้วย

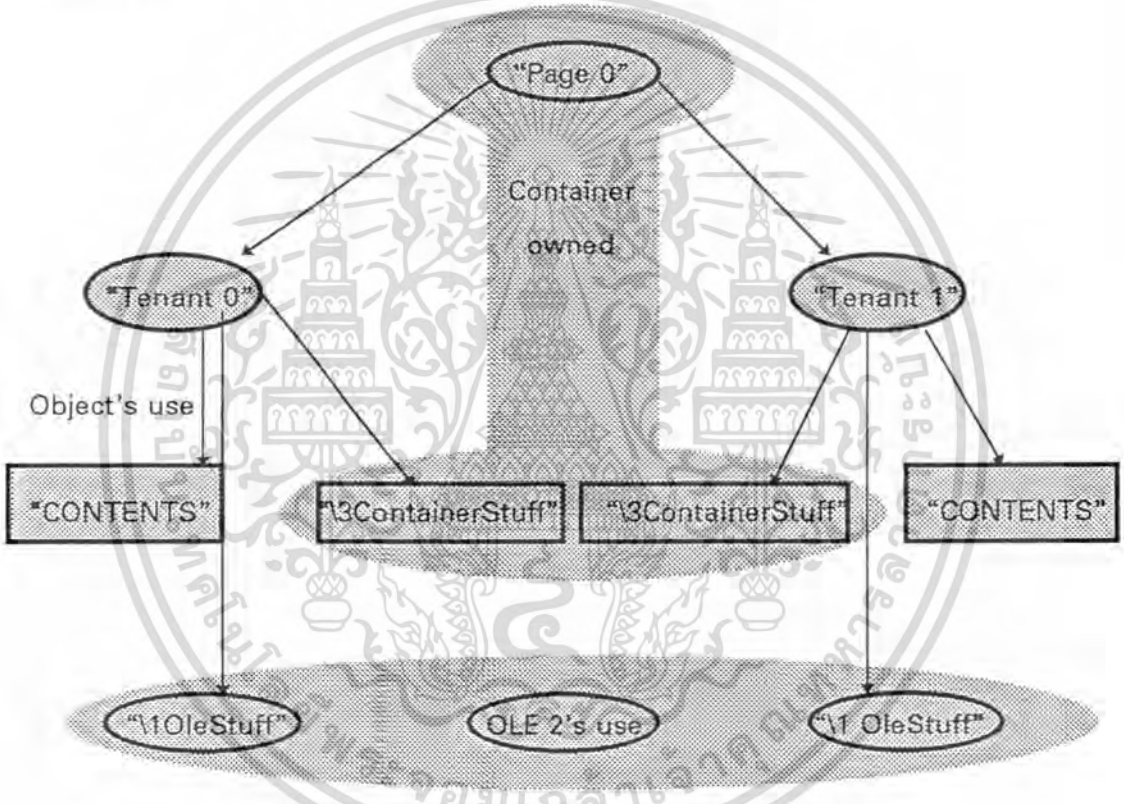


รูปที่ 4.11 แสดงวิธีในการกำจัดแฟรกเมนท์ในไฟล์เชิงซ้อน

การตั้งชื่อและการผูกออบเจกต์เข้าด้วยกัน (Nameing and Binding)

- การตั้งชื่อ จะเป็นสิ่งที่ใช้ในการระบุสตอเรจ กับ สตรีมออบเจกต์ โดยสามารถตั้งชื่อได้ยาวถึง 31 ตัวอักษร และชื่อของสตอเรจราก จะต้องเป็นไปตามข้อจำกัดของระบบไฟล์ แต่ก็ยังสามารถตั้งชื่อโดยใช้อักขระที่มีรหัสแอสกีมากกว่า 32 ได้ และห้ามมี ., \, /, : และ ! ผสมอยู่ในชื่อที่ตั้ง

ชื่อใดที่มีเลข 3 นำหน้า จะเป็นอีลิเมนท์ที่ไม่มีการควบคุมจากเจ้าของของสตอเรจที่อยู่เหนือสตอเรจที่มีอีลิเมนท์นั้นอยู่ ซึ่งนำไปใช้ในการเก็บสารสนเทศในสตอเรจนั้น ซึ่งสตอเรจอื่นต้องการใช้



รูปที่ 4.12

- การผูกออบเจกต์เข้าด้วยกัน เป็นกระบวนการเชื่อมต่ออีกครั้ง (reconnecting) ของออบเจกต์ที่เกิดจากการเชื่อมโยงไปยังแหล่งข้อมูลที่ระบุไว้โดยมอนิเกอร์ (moniker) โดยอาจเกิดขึ้นตอนปล่อยแอพพลิเคชัน (launching application), การเรียกใช้ไฟล์ (load file) และการขอพอยน์เตอร์ที่ชี้ไปยังอินเตอร์เฟส

ซึ่งมอนิเกอร์ เป็นออบเจกต์อีกชนิดหนึ่งที่สนับสนุนการผูกออบเจกต์ ประกอบไปด้วยการอ้างอิงไปยังข้อมูลที่จะทำการเชื่อมโยง และรหัส (code) ที่รู้วิธีการผูกการเชื่อมโยงออบเจกต์เข้าด้วยกัน โดยทั่วไปมักจะใช้มอนิเกอร์ในการบอกรายละเอียดของแหล่งข้อมูลสำหรับออบเจกต์ที่

จะทำการเชื่อมโยงในเอกสารเชิงซ้อน จะเป็นผลให้มีการร้องขอการอ้างอิงไปยังไฟล์ และคุณลักษณะของบางส่วนของไฟล์ที่เป็นแหล่งที่แท้จริงของการเชื่อมโยง

4.3.3 ฟีเจอร์ระดับสูง

ฟีเจอร์ระดับนี้จะเป็นการเพิ่มสิ่งอำนวยความสะดวกให้กับผู้ใช้ โดยจะแบ่งออกเป็นฟีเจอร์ที่มีอยู่ในปัจจุบัน ซึ่งประกอบไปด้วย การลากและวาง (Drag and Drop), การทำงานในสถานที่ (In-Place Activation) และการทำงานอัตโนมัติ (Automation) และฟีเจอร์ที่จะเกิดขึ้นในอนาคต (เป็นเทคโนโลยีของ OLE รุ่นถัดไป)

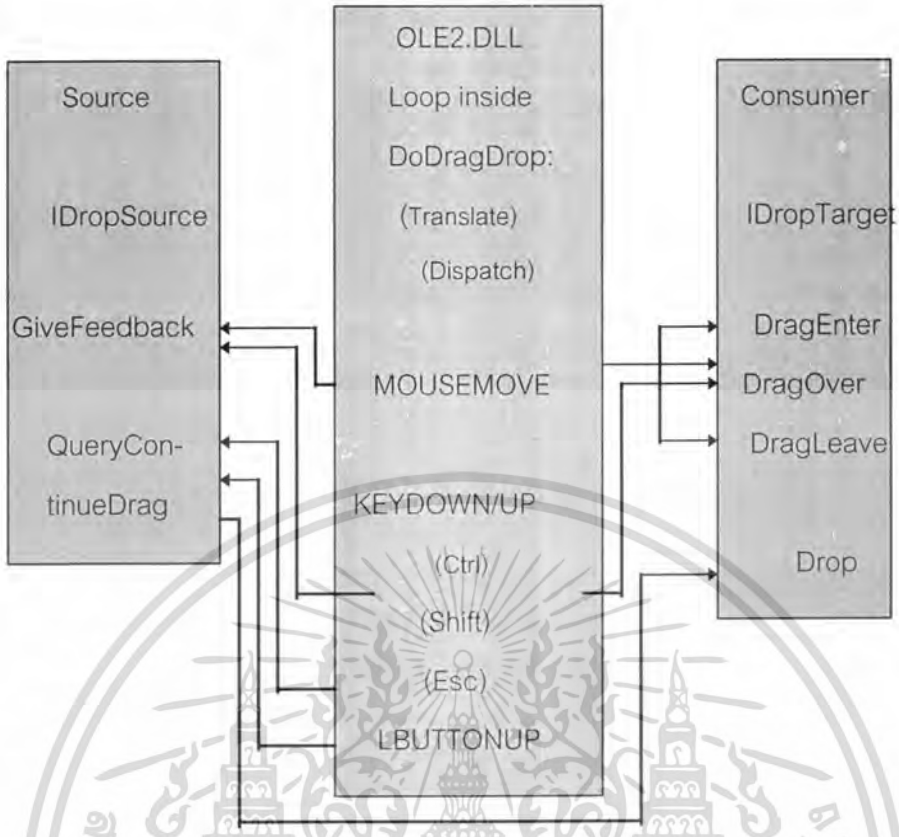
- **ฟีเจอร์ที่มีอยู่ในปัจจุบัน**

  - **การลากและวาง** (Drag and Drop)

ทำงานโดยทำการเลือกข้อมูลจากแอพพลิเคชันที่เป็นแหล่งกำเนิด แล้วหยิบข้อมูลนั้นขึ้นมาโดยการคลิกเมาส์ หลังจากนั้นก็ลากข้อมูลดังกล่าวจากหน้าต่างของแอพพลิเคชันที่เป็นแหล่งกำเนิด (source application) ไปยังหน้าต่างของแอพพลิเคชันที่เป็นเป้าหมาย (consumer application)

จากการทำงานข้างต้น จะเห็นได้ว่าผลที่ได้เหมือนกับการดำเนินการแบบตัด/ทำสำเนาทับข้อมูล (Cut / Copy-Paste operation) แต่อันที่จริงแล้วทั้งสองวิธีนี้มีความแตกต่างกัน กล่าวคือ การลากและวางต่างกับการดำเนินการแบบตัด/ทำสำเนา-แปะข้อมูล ตรงที่กระบวนการถ่ายโอนทั้งหมดเกิดขึ้นในครั้งเดียว และแหล่งกำเนิดกับเป้าหมายถ่ายโอนข้อมูลกันโดยไม่ต้องใช้คลิปบอร์ด

การลากและวางในเทคโนโลยีของโอแอลอี 2.0 เป็นการเคลื่อนย้ายพอยน์เตอร์ของดาต้าออบเจกต์ (IDataObject) จากแอพพลิเคชันที่เป็นแหล่งกำเนิดไปยังแอพพลิเคชันที่เป็นเป้าหมาย โดยแอพพลิเคชันดังกล่าว อาจเป็นแอพพลิเคชันเดียวกัน หรือต่างกันก็ได้ เพราะโอแอลอี 2.0 จะเป็นตัวกลางระหว่างแหล่งกำเนิดกับเป้าหมาย ดังแสดงไว้ในรูปที่ 4.13



รูปที่ 4.13 แสดงการดำเนินการลากและวาง

- **การทำงานในสถานที่** (In-Place Activation)

ด้วยมุมมองของการโปรแกรมแบบมีเอกสารเป็นศูนย์กลางนั้น การฝังหรือเชื่อมโยงออบเจกต์เข้ากับเอกสารของคอนเทนเนอร์ (container document) ซึ่งคอนเทนเนอร์จะมีฟังก์ชันบางฟังก์ชันที่ใช้ในการกระตุ้นออบเจกต์ในเอกสารให้มีการจัดการออบเจกต์นั้น

ซึ่งเทคโนโลยีของโอแอลอี 2.0 สามารถกระตุ้นออบเจกต์ในสถานที่ (In-Place) คือมีการทำงานกับออบเจกต์ภายในหน้าต่างของแอพพลิเคชันของคอนเทนเนอร์ แทนที่จะให้ออบเจกต์เปิดหน้าต่างใหม่เพื่อปฏิบัติงาน

ทั้งนี้ผู้ใช้จะได้ประโยชน์จากการที่ไม่ต้องออกจากเอกสารที่ใช้งานอยู่ ทั้งยังไม่ต้องสับสนกับหน้าต่างอื่น ๆ อีกด้วย

- **การทำงานโดยอัตโนมัติ** (Automation)

เป็นเทคโนโลยีที่ออบเจกต์หนึ่งแสดงกลุ่มของรหัส และฟังก์ชัน เพื่อสามารถนำไปใช้ได้ในแอพพลิเคชันอื่น ๆ ได้

ด้วยเทคโนโลยีนี้เองทำให้แอพพลิเคชันที่เป็นผู้ใช้บริการ (Client) สามารถนำการทำงานของแอพพลิเคชันที่เป็นผู้ให้บริการ (Server) มาใช้งาน โดยการเรียกใช้มาโคร หรือภาษาในการโปรแกรมของผู้ให้บริการ เหมือนกับเป็นส่วนขยายของมาโคร หรือเป็นภาษาในการโปรแกรมของผู้ใช้บริการเอง

บทที่ 5

การพัฒนาโปรแกรมการจัดการข้อมูลรูปภาพโดยใช้เทคนิคโอแอลอี

5.1 อัลกอริทึมที่ใช้ในการจัดการข้อมูลรูปภาพ

5.1.1 อัลกอริทึมแบบ Pixel Point Processing

ก. การปรับค่าความสว่าง (Brightness)

คำอธิบาย เป็นการปรับค่าความสว่างของสีในแต่ละพิกเซล เพื่อให้ภาพมีความสว่างมากขึ้นหรือมืดลง

วิธีการ จะทำการบวกหรือลบด้วยค่าคงที่เข้ากับค่าสีในแต่ละพิกเซล

ส่วนโปรแกรม

```
for(i=0; I<(WORD)NumberofColors;i++) {
    if (mPaletteBuffer[i*3]+R > 255)
        mPaletteBuffer[i*3] = 255;
    else mPaletteBuffer[i*3] = mPaletteBuffer[i*3]+R;
    if (mPaletteBuffer[i*3+1]+G > 255)
        mPaletteBuffer[i*3+1] = 255;
    else mPaletteBuffer[i*3+1] = mPaletteBuffer[i*3+1]+R;
    if (mPaletteBuffer[i*3+2]+B > 255)
        mPaletteBuffer[i*3+2] = 255;
    else mPaletteBuffer[i*3+2] = mPaletteBuffer[i*3+2]+B;
}
for( i=0;i<(WORD)NumberOfColors;i++) {
    npPal->palPalEntry[i].peRed = mPaletteBuffer[i*3];
    npPal->palPalEntry[i].peGreen = mPaletteBuffer[i*3+1];
    npPal->palPalEntry[i].peBlue = mPaletteBuffer[i*3+2];
```

รูปที่ 5.1 ภาพที่ได้จากการปรับความสว่าง

ข. การสร้างภาพเกรย์สเกล (Gray Scale)

คำอธิบาย เป็นการแปลงภาพสีไปเป็นภาพเกรย์สเกลคือจะใช้จำนวนของเกรย์เลเวลที่สามารถมีได้ แทนความสว่างของภาพ

วิธีการ จะนำค่าอาร์จีบีของแต่ละพิกเซลมาคำนวนตามสูตร ดังนี้

Gray = (BYTE) ((0.3*R)+(0.59*G)+(0.11*B)) ;

ส่วนโปรแกรม

```
for(i=0; I<(WORD)NumberofColors;i++) {
 Red =  0.3 * mPaletteBuffer[i*3] ;
 Green = 0.59 * mPaletteBuffer[i*3+1];
 Blue = 0.11 * mPaletteBuffer[i*3+2];
 Gray = Red + Green + Blue ;
 npPal->palPalEntry[i].peRed   = Gray;
 npPal->palPalEntry[i].peGreen = Gray;
 npPal->palPalEntry[i].peBlue  = Gray;
}
```

รูปที่ 5.2 ภาพที่ได้จากการสร้างภาพเป็นแบบเกรย์สเกล

ค. การสร้างภาพขาวดำ (Threshold)

คำอธิบาย เป็นการแปลงภาพเกรย์สเกลไปเป็นภาพขาวดำ ซึ่งแต่ละพิกเซลจะสามารถมีได้เพียงสองสีคือสีขาวหรือสีดำเท่านั้น

วิธีการ จะตั้งค่าคงที่ค่าหนึ่ง ซึ่งถ้า ค่าพิกเซลมีค่ามากกว่าค่าคงที่นั้นๆ ค่าพิกเซลจะถูกกำหนดให้มีค่าพิกเซลใหม่เท่ากับค่ามากสุดของความสว่างคือสีขาว (255) และถ้าค่าพิกเซลมีค่าน้อยกว่าค่าคงที่ ค่าพิกเซลใหม่ใหม่ที่ได้จะมีค่าเท่ากับค่าน้อยสุดของความสว่างดำ (0)

ส่วนโปรแกรม

```
for(i=0; I<(WORD)NumberofColors;i++) {
  if ( (mPaletteBuffer[i*3] > threshold) &&
       (mPaletteBuffer[i*3+1] > threshold) &&
       (mPaletteBuffer[i*3+2] > threshold) )
          {  mPaletteBuffer[i*3] = 255;
             mPaletteBuffer[i*3+1] = 255;
             mPaletteBuffer[i*3+2] = 255;
          }
    else  {  mPaletteBuffer[i*3] = 0;
             mPaletteBuffer[i*3+1] = 0;
             mPaletteBuffer[i*3+2] = 0;
          }
}
```

รูปที่ 5.3 ภาพที่ได้จากการสร้างภาพขาวดำ

5.1.2 อัลกอริทึมแบบ Area Process

ก. การขจัดสัญญาณรบกวน (noise) ของภาพ

- เทคนิค Low Pass

<u>คำอธิบาย</u> เป็นฟิวเตอร์ที่ใช้ในการแปลงภาพ เพื่อทำให้ภาพดูนุ่มขึ้น เป็นการขจัดนอยส์ชนิดหนึ่งโดยการกระทำกับเมตริกซ์มีค่า $\begin{Bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{Bmatrix}$

<u>วิธีการ</u> นำค่าอาร์จีบีของแต่ละพิกเซลมากระทำกับเมตริกซ์ที่กำหนดไว้ แล้วนำค่าอาร์จีบีที่ได้ไปเทียบในตารางพาเลตต์เพื่อแสดงผล

<u>ส่วนโปรแกรม</u>

```
for (short y=0; y<mHeight; y++) {
  for (short x=0; x<mWidth; x++){
     if( (x<1 || y<1) || (x>mWidth-1 || y>mHeight-1)
       data = Getdata(x, y, DestinationBuffer);
     else {
       GetRGB(x, y, tmp, mDestinationBuffer);
       r = 1*(temp[0].peRed) + 2*(temp[1].peRed) + 1*(temp[2].peRed)
           + 2*(temp[3].peRed) + 4*(temp[4].peRed) + 2*(temp[5].peRed)
           + 1*(temp[6].peRed) + 2*(temp[7].peRed) + 1*(temp[8].peRed)
       g = 1*(temp[0].peGreen) + 2*(temp[1].peGreen) + 1*(temp[2].peGreen)
           + 2*(temp[3].peGreen) + 4*(temp[4].peGreen) +
           2*(temp[5].peGreen)
           + 1*(temp[6].peGreen) + 2*(temp[7].peGreen) +
           1*(temp[8].peGreen)
       b = 1*(temp[0].peBlue) + 2*(temp[1].peBlue) + 1*(temp[2].peBlue)
           + 2*(temp[3].peBlue) + 4*(temp[4].peBlue) + 2*(temp[5].peBlue)
           + 1*(temp[6].peBlue) + 2*(temp[7].peBlue) + 1*(temp[8].peBlue)
       data = (unsigned char)Matching(r, g, b);
       }
       Putdata(x, y, Buffer, data);  }  }
```

- เทคนิค High Pass

<u>คำอธิบาย</u> เป็นฟิวเตอร์ที่ใช้ในการแปลงภาพชนิดหนึ่งสำหรับการหาลักษณะที่เด่นชัดของภาพซึ่งจะได้ผลคล้ายกับการหาขอบภาพ โดยการกระทำกับเมตริกซ์ ซึ่งมีค่าเท่ากับ

$$\begin{Bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{Bmatrix} \text{ หรือ } \begin{Bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{Bmatrix} \text{ หรือ } \begin{Bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{Bmatrix}$$

<u>วิธีการ</u> นำค่าอาร์จีบีของแต่ละพิกเซลมากระทำกับเมตริกซ์ที่กำหนดไว้ แล้วนำค่าอาร์จีบีที่ได้ไปเทียบในตารางพาเลตต์เพื่อแสดงผล

<u>ส่วนโปรแกรม</u>

```
for (short y=0; y<mHeight; y++) {
   for (short x=0; x<mWidth; x++){
     if( (x<1 || y<1) || (x>mWidth-1 || y>mHeight-1) )
        data = Getdata(x, y, DestinationBuffer);
     else {
        GetRGB(x, y, tmp, mDestinationBuffer);
      r = 0*(temp[0].peRed) + (-1)*(temp[1].peRed) + 0*(temp[2].peRed)
          + (-1)*(temp[3].peRed) + 5*(temp[4].peRed) + (-)*(temp[5].peRed)
          + 0*(temp[6].peRed) + (-1)*(temp[7].peRed) + 0*(temp[8].peRed);
      g = 0*(temp[0].peGreen)+(-1)*(temp[1].peGreen)+0*(temp[2].peGreen)
      +(-1)*(temp[3].peGreen)+5*(temp[4].peGreen)+(-1)*(temp[5].peGreen)
      +0*(temp[6].peGreen)+(-1)*(temp[7].peGreen)+ 0*(temp[8].peGreen);
      b = 0*(temp[0].peBlue) + (-1)*(temp[1].peBlue) + 0*(temp[2].peBlue)
         + (-1)*(temp[3].peBlue) + 5*(temp[4].peBlue)
         + (-1)*(temp[5].peBlue) + 0*(temp[6].peBlue)
         + (-1)*(temp[7].peBlue) + 0*(temp[8].peBlue);
     data = (unsigned char)Matching(r, g, b);
      }
      Putdata(x, y, Buffer, data);
        }          }
```

ภาพเดิม

รูปที่ 5.4 High Pass

รูปที่ 5.5 Low Pass

ข. การทำภาพมัว (Blur)

คำอธิบาย เป็นฟิวเตอร์ที่ใช้ในการแปลงภาพชนิดหนึ่งสำหรับทำให้ภาพดูนุ่มขึ้นผลที่ได้คล้ายกับการใช้ฟิวเตอร์ Low Pass โดยกระทำกับเมตริกซ์ซึ่งมี

ค่าเท่ากับ $\begin{Bmatrix} 1/8 & 1/8 & 1/8 \\ 1/8 & 0 & 1/8 \\ 1/8 & 1/8 & 1/8 \end{Bmatrix}$

วิธีการ นำค่าอาร์จีบีของแต่ละพิกเซลมากระทำกับเมตริกซ์ที่กำหนดไว้ แล้วนำค่าอาร์จีบีที่ได้ไปเทียบในตารางพาเลตต์เพื่อแสดงผล

ส่วนโปรแกรม

```
for (short y=0; y<mHeight; y++) {
   for (short x=0; x<mWidth; x++){
    if( (x<1 || y<1) || (x>mWidth-1 || y>mHeight-1) )
        data = Getdata(x, y, DestinationBuffer);
    else {
      GetRGB(x, y, tmp, mDestinationBuffer);
      r = ((temp[0].peRed) + (temp[1].peRed) + (temp[2].peRed)
          + (temp[3].peRed) + 0*(temp[4].peRed) + (temp[5].peRed)
          + (temp[6].peRed) + (temp[7].peRed) + (temp[8].peRed))/8 ;
      g = ((temp[0].peGreen) + (temp[1].peGreen) + (temp[2].peGreen)
          + (temp[3].peGreen) + 0*(temp[4].peGreen) + (temp[5].peGreen)
          + (temp[6].peGreen) + (temp[7].peGreen) + (temp[8].peGreen))/8 ;
      b = ((temp[0].peBlue) + (temp[1].peBlue) + (temp[2].peBlue)
          + (temp[3].peBlue) + 0*(temp[4].peBlue) + (temp[5].peBlue)
          + (temp[6].peBlue) + (temp[7].peBlue) + (temp[8].peBlue)) /8 ;
      data = (unsigned char)Matching(r, g, b);
      }
      Putdata(x, y, Buffer, data);
    }  }
```

รูปที่ 5.6 ภาพที่ได้จากการทำภาพมัว

ค. การหาขอบของภาพ (Edge Detection)

<u>คำอธิบาย</u> การหาขอบของภาพเป็นองค์ประกอบที่สำคัญของอิมเมจโปรเซสซิง เป็นการเพิ่มความเด่นของขอบของภาพให้เด่นชัดขึ้น มีหลายวิธี ในที่นี้จะเป็นแบบวิธีของโซเบล(Sobel Algorithm)โดยการกระทำกับเมตริกซ์ ซึ่งมีค่าเท่ากับ

$$\begin{Bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{Bmatrix} \text{ และ } \begin{Bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{Bmatrix} \text{ และ } \begin{Bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{Bmatrix} \text{ และ } \begin{Bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{Bmatrix}$$

<u>วิธีการ</u> นำค่าอาร์จีบีของแต่ละพิกเซลมากระทำกับเมตริกซ์ที่กำหนดไว้ทั้ง 4 เมตริกซ์ แล้วนำค่าอาร์จีบีที่ได้ของแต่ละการคำนวณมาหาค่าสัมบูรณ์ แล้วนำค่าทั้ง 4 ของ อาร์จีบีมาเปรียบเทียบกันแล้วเอาค่าที่มากสุด แล้วนำไป เทียบในตารางพาเลตต์เพื่อแสดงผล

<u>ส่วนโปรแกรม</u>

```
for ( short y=0; y<mHeight; y++)
{   for ( short   x=0; x < mWidth; x++) {
        if ( (x < 1 || y < 1) || (x > mWidth-1 || y > mHeight-1) )
      data = Getdata(x, y, mDestinationBuffer);
      else {   GetRGB(x, y, temp, mDestinationBuffer);
               r = (-1)*temp[0].peRed + (-2)*temp[1].peRed +
                   (-1)*temp[2].peRed + 1*temp[6].peRed +
                    2*temp[7].peRed + 1*temp[8].peRed;
               rTemp [0] = abs(r);
               g = (-1)*temp[0].peGreen + (-2)*temp[1].peGreen +
                   (-1)*temp[2].peGreen + 1*temp[6].peGreen +
                    2*temp[7].peGreen + 1*temp[8].peGreen;
               gTemp [0] = abs(g);
               b = (-1)*temp[0].peBlue + (-2)*temp[1].peBlue +
                   (-1)*temp[2].peBlue + 1*temp[6].peBlue +
                    2*temp[7].peBlue + 1*temp[8].peBlue;
```

```
bTemp [0] = abs(b);
r = (-1)*temp[1].peRed + (-2)*temp[2].peRed +
     1*temp[3].peRed   + (-1)*temp[5].peRed +
    2*temp[6].peRed + 1*temp[7].peRed;
rTemp [1] = abs(r);
g = (-1)*temp[1].peGreen + (-2)*temp[2].peGreen +
    1*temp[3].peGreen + (-1)*temp[5].peGreen +
    2*temp[6].peGreen + 1*temp[7].peGreen;
gTemp [1] = abs(g);
b = (-1)*temp[1].peBlue + (-2)*temp[2].peBlue +
     1*temp[3].peBlue + (-1)*temp[5].peBlue +
     2*temp[6].peBlue + 1*temp[7].peBlue;
bTemp [1] = abs(b);
r = (-1)*temp[0].peRed + 1*temp[2].peRed +
     (-2)*temp[3].peRed        + 2*temp[5].peRed +
     (-1)*temp[6].peRed + 1*temp[8].peRed;
rTemp [2] = abs(r);
g = (-1)*temp[0].peGreen + 1*temp[2].peGreen +
     (-2)*temp[3].peGreen + 2*temp[5].peGreen +
    (-1)*temp[6].peGreen + 1*temp[8].peGreen;
gTemp [2] = abs(g);
b = (-1)*temp[0].peBlue + 1*temp[2].peBlue +
     (-2)*temp[3].peBlue + 2*temp[5].peBlue +
     (-1)*temp[6].peBlue + 1*temp[8].peBlue;
bTemp [2] = abs(b);
r = (-2)*temp[0].peRed + (-1)*temp[1].peRed +
     (-1)*temp[3].peRed + 1*temp[5].peRed +
     1*temp[7].peRed + 2*temp[8].peRed;
rTemp [3] = abs(r);
g = (-2)*temp[0].peGreen + (-1)*temp[1].peGreen +
     (-1)*temp[3].peGreen + 1*temp[5].peGreen +
```

```
                1*temp[7].peGreen + 2*temp[8].peGreen;
            gTemp [3] = abs(g);
            b = (-2)*temp[0].peBlue + (-1)*temp[1].peBlue +
                (-1)*temp[3].peBlue + 1*temp[5].peBlue +
                1*temp[7].peBlue + 2*temp[8].peBlue;
            bTemp [3] = abs(b);
            r = 0; g = 0; b = 0;
            for(int j=0; j<4; j++) {
                    if( rTemp[ j ] > r )  r = rTemp[ j ];
                    if( gTemp[ j ] > g )  g = gTemp[ j ];
                    if( bTemp[ j ] > b )  b = bTemp[ j ];
                    }
            data = (unsigned char)Matching(r, g, b);
            }
        Putdata( x, y, Buffer, data);
    }
}
```

รูปที่ 5.7 ภาพที่ได้จากการหาขอบเขตของภาพ

5.1.3 Geometric Transformation Processing

ก. การกลับภาพ (Flip)

- กลับภาพตามแนวนอน (Flip Horizontal)

คำอธิบาย เป็นการกลับภาพตามแนวนอนซึ่งภาพที่ได้จะเหมือนกับการส่องกระจก คือภาพที่ได้จะกลับจากซ้ายเป็นขวา

วิธีการ ทำการย้ายตำแหน่งของแต่ละพิกเซลไปยังตำแหน่งใหม่ เช่น พิกเซลแรกจะถูกย้ายไปที่ตำแหน่งที่ N พิกเซลที่สองจะถูกย้ายไปที่ตำแหน่งที่ N - 1 เป็นต้น

ส่วนของโปรแกรม

```
for ( short I= 0;I <mHeight; I++) {
    k = mWidth-1;
    for( short j= 0; j<mWidth; j++) {
      Buffer[i*mWidth+k] =
      mDestinationBuffer[i*mWidth+j];
      k--;
    }
}
```

รูปที่ 5.8 ภาพที่ได้จากการกลับภาพตามแนวนอน

- กลับภาพตามแนวตั้ง (Flip Vertical)

คำอธิบาย เป็นการกลับภาพตามแนวตั้งซึ่งภาพที่ได้จะเหมือนกับการส่องกระจกคือภาพที่ได้จะกลับจากบนลงล่าง

วิธีการ ทำการย้ายตำแหน่งของแต่ละพิกเซลไปยังตำแหน่งใหม่ เช่นพิกเซลแรกจะถูกย้ายไปยังตำแหน่งแรกของแถวสุดท้าย, พิกเซลที่สองจะถูกย้ายไปยังตำแหน่งที่สองของแถวสุดท้ายเป็นต้น

ส่วนของโปรแกรม

```
for ( short i= 0;i <mHeight; I++) {
    k = mHeight-(i+1);
    for( short j= 0; j<mWidth; j++) {
      Buffer[k*mWidth+j] =
      mDestinationBuffer[i*mWidth+j];
    }
}
```

รูปที่ 5.9 ภาพที่ได้จากการกลับภาพตามแนวตั้ง

ข. การหมุนภาพ (Rotate)

- หมุนภาพตามเข็มนาฬิกา

คำอธิบาย เป็นการหมุนภาพตามเข็มนาฬิกาไปเป็นมุมทีละ 90 องศา

วิธีการ ทำการย้ายตำแหน่งของแต่ละพิกเซล เช่นพิกเซลแรกจะถูกย้ายไปอยู่ที่แถวแรกของคอลัมภ์สุดท้ายของภาพ พิกเซลที่สองถูกย้ายไปอยู่ที่แถวที่สองของคอลัมภ์เดียวกัน เป็นต้น

ส่วนของโปรแกรม

```
for( short i=0; i<mHeight; i++) {
k = mHeight - 1;
for( short j=0; j<mWidth; j++) {
        Buffer[ j*ByteHeight+(k-i)] =
            mDestinationBuffer[i*mByteWidth+j];
}
}
```

- หมุนภาพทวนเข็มนาฬิกา

คำอธิบาย เป็นการหมุนภาพทวนเข็มนาฬิกาเป็นมุมทีละ 90 องศา

วิธีการ ทำการย้ายตำแหน่งของแต่ละพิกเซล เช่นพิกเซลแรกจะถูกย้ายไปอยู่คอลัมภ์สุดท้ายของแถวแรกของภาพ พิกเซลที่สองจะถูกย้ายไปอยู่ที่คอลัมภ์ถัดมาของแถวเดียวกัน เป็นต้น

ส่วนของโปรแกรม

```
for( short i=0; i<mHeight; i++) {
        k = mWidth - 1;
        for( short j=0; j<mWidth; j++) {
            Buffer[ (k*ByteHeight)+i] =
            mDestinationBuffer[i*mByteWidth+j];
            k--;
        }
}
```

รูปที่ 5.8 ภาพที่ได้จากการหมุนภาพตามเข็มนาฬิกา

รูปที่ 5.11 ภาพที่ได้จากการหมุนภาพทวนเข็มนาฬิกา

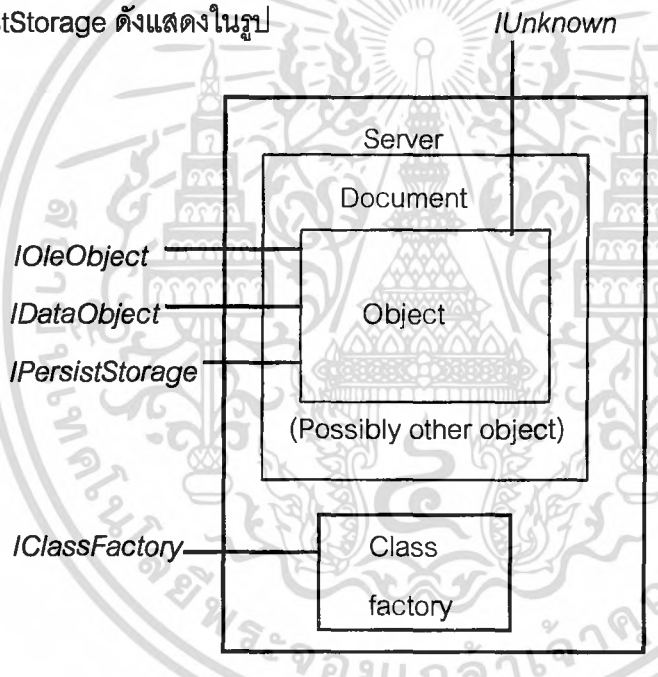
5.2 การเขียนโปรแกรมโดยใช้โอแอลอี

5.2.1 IN-PLACE ACTIVATION

เป็นหนึ่งในฟีเจอร์ที่เพิ่มขึ้นมาจาก OLE 1.0 เป็นความสามารถในการแก้ไขออบเจกต์ในขณะที่ยังทำงานอยู่ในคอนเท็กซ์(context) ของคอนเทนเนอร์แอพพลิเคชัน (container application) ภายใต้แนวความคิดของการโปรแกรมแบบใช้เอกสารเป็นศูนย์กลาง (document-centric)โดยที่ผู้ ใช้ไม่จำเป็นต้องรู้ที่มาของแอพพลิเคชันที่กำลังใช้งานอยู่ว่ามาจากไหน ใครเป็นผู้แทนจำหน่าย สนใจแต่เพียงวิธีใช้งานเท่านั้น ซึ่งเป็นหัวใจสำคัญในการพัฒนาแอพพลิเคชันบนฟีเจอร์นี้ และไม่สนใจที่จะออกแบบหน้าตาให้สวยงามมากนัก

5.2.2 โครงสร้างของแอพพลิเคชันเซิร์ฟเวอร์

จะประกอบไปด้วย 3 อินเตอร์เฟสคือ IOleObject, IDataObject และ IPersistStorage ดังแสดงในรูป



รูปที่ 5.12

โดยแต่ละอินเตอร์เฟสจะมีจุดประสงค์ในการใช้ที่แตกต่างกันออกไปดังนี้

IPersistStorage : เป็นอินเตอร์เฟสผ่านออบเจกต์ที่เกี่ยวกับ IStorage ในเอกสารเชิงซ้อนของคอนเทนเนอร์ ออบเจกต์จะเป็นตัวเรียกใช้ และเก็บข้อมูลเดิม ซึ่งจะใช้สำหรับออบเจกต์แบบฝังตัวเท่านั้น

IDataObject : เป็นอินเตอร์เฟสผ่าน handler ตามปกติแล้ว OLE2.DLL จะใช้สำหรับการนำเสนอเพื่อเก็บเมตาไฟล์ หรือบิตแมปไฟล์ handler มักจะเรียกออบเจกต์สำหรับคำแนะนำในการติดต่อ ซึ่งจะบอกได้ว่าเมื่อใดที่ข้อมูลมีการเปลี่ยนแปลงในเซิร์ฟเวอร์ และในกรณีที่แหล่ง

เก็บข้อมูลต้องการปรับปรุงให้ทันสมัยขึ้น คอนเทนเนอร์จะเป็นตัวเรียกออบเจกต์ข้อมูลสำหรับคำแนะนำในการติดต่อ หรือต้องการคำร้องขอเพื่อทำสำเนาของข้อมูลพิเศษอื่น ๆ นอกจากข้อมูลที่อยู่ในรูปแบบของกราฟฟิก

IOleObject : เป็นอินเตอร์เฟสที่ใช้ควบคุมในส่วนอื่น ๆ นอกเหนือจากที่ได้กล่าวมาแล้ว และจะพบว่ามีฟังก์ชันสมาชิกมากมายให้ใช้งานใน IOleObject อินเตอร์เฟสดังต่อไปนี้

5.2.3 ความแตกต่างระหว่างมินิเซิร์ฟเวอร์ (Mini-Server) และฟูลเซิร์ฟเวอร์ (Full-Server)

มินิเซิร์ฟเวอร์ : เป็นเซิร์ฟเวอร์ที่จะทำงานได้ภายในเอกสารที่ได้ฝังตัวลงไปเท่านั้น และจะทำงานได้ดีสำหรับข้อมูลที่มีขนาดไม่ใหญ่มาก ยกตัวอย่างเช่น ตัวอักษร หรือ ตัวอย่างในการวาดภาพเป็นต้น ทั้งนี้เพราะเป็นเซิร์ฟเวอร์ที่สนับสนุนแต่การฝังตัว ดังนั้นหากข้อมูลมีขนาดใหญ่มากจะทำให้เอกสารมีขนาดใหญ่ตามไปด้วย และเนื่องจากที่มินิเซิร์ฟเวอร์ไม่สามารถทำงานได้ด้วยตนเอง จึงมีข้อจำกัดในการเรียก, ปรับแต่ง และเก็บข้อมูล

ฟูลเซิร์ฟเวอร์ : เป็นแอพพลิเคชันที่สนับสนุนทั้งการฝังตัว และการเชื่อมโยง ทั้งยังสามารถทำงานได้ด้วยตนเอง

- การติดตั้ง

การเข้าถึงเอกสารเชิงซ้อนของเซิร์ฟเวอร์ ต้องทำการปรับแต่งโปรแกรมติดตั้งบางส่วนคือจะต้องลงทะเบียนลงในฐานข้อมูลการลงทะเบียน ที่จะให้ข้อมูลเกี่ยวกับออบเจกต์พิเศษเชิงซ้อนแก่ OLE และแอพพลิเคชันอื่น ๆ สำหรับวิธีที่ลงทะเบียนในฐานข้อมูลการลงทะเบียนนั้น ทำได้ใช้เอดิเตอร์ในการลงทะเบียนฐานข้อมูล (RegEdit) เพื่อสร้างไฟล์ .REG

อันดับแรกจะทำสำเนาของไฟล์ .REG ที่ได้จากการสร้างด้วยเอดิเตอร์ดังที่กล่าวไปแล้วนั้น ลงในไดเรกทอรี่ที่ติดตั้ง ในลำดับต่อไปจะเรียกใช้ฟังก์ชัน API ของวินโดว์ที่ชื่อ WinExec โดยใช้คำสั่ง regedit-s <path to .REG file> ซึ่งคำสั่งนี้จะรวมสารบัญของไฟล์ .REG กับฐานข้อมูลการลงทะเบียน อันดับสุดท้ายจะใช้ฟังก์ชันของ SHELL.DLL อย่าง RegOpenKey และ RegSetValue เพื่อปรับปรุงทางเข้า LocalServer, InProcServer และ InProcHandler สำหรับแอพพลิเคชันที่ทำการติดตั้ง และสิ่งที่จะขาดไม่ได้ก็คือเส้นทางที่เป็นทางเข้าสู่แอพพลิเคชันโดยนับจากระบบ

5.2.4 ขั้นตอนในการสร้างเซิร์ฟเวอร์แบบฝังตัว

แนวคิดในการสร้างเซิร์ฟเวอร์แบบฝังตัว

- ร่างการทำงานของแอพพลิเคชันเซิร์ฟเวอร์ที่ต้องการ
- เขียนแอพพลิเคชันโดยยังไม่ต้องคำนึงถึงการสนับสนุนการฝังตัว

●วางแผนถึงวิธีที่เซิร์ฟเวอร์จะใช้ในการสร้างออบเจกต์ และการติดตั้งเซิร์ฟเวอร์ ในขั้นตอนนี้จะสนใจ 3 อินเตอร์เฟสที่ได้กล่าวไปแล้วข้างต้น และบางส่วนของการติดต่อกับผู้ใช้

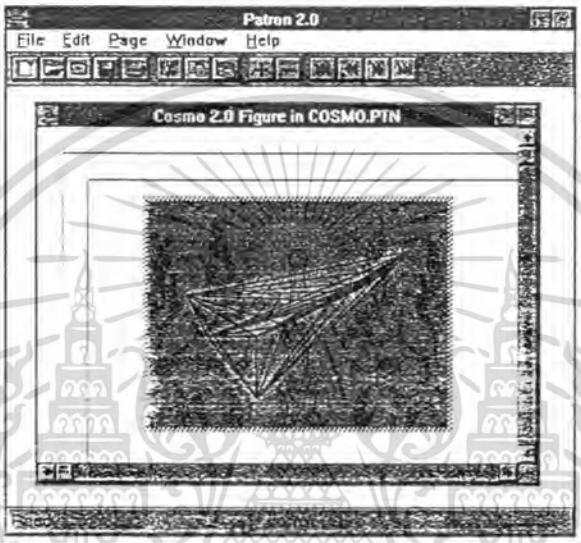
●สำหรับขั้นตอนนี้ จะเป็นส่วนที่เพิ่มขึ้นมาในฟูลเซิร์ฟเวอร์ โดยเฉพาะอย่างยิ่งความแตกต่างในเรื่องของ MDI และ SDI

จากแนวคิดข้างต้นจะได้ขั้นตอนในการสร้างเซิร์ฟเวอร์แบบฝังตัวดังนี้คือ

1. เรียกฟังก์ชัน SetMessageQueue(96) (เฉพาะวินโดวส์เวอร์ชัน 3.1 เท่านั้น), OleBuildVersion, OleInitial และ OleUninitialize
2. สร้างทางเข้าเอกสารเชิงซ้อนในฐานข้อมูลการลงทะเบียน
3. จัดการคลาสแฟคทอรีและการลงทะเบียนบางแอพพลิเคชันจะต้องลงทะเบียนคลาสเมื่อพบ-Embedding ที่คำสั่งในบรรทัดเมื่อเริ่มการทำงาน รวมทั้งเทคนิคในการปิดแอพพลิเคชัน
4. จัดการกับออบเจกต์ด้วย Iunknown อินเตอร์เฟส และขยายคำสั่งต่าง ๆ ในแอพพลิเคชันเพื่อสร้างและจัดการออบเจกต์นี้
5. จัดการ IPersistStorage อินเตอร์เฟสสำหรับออบเจกต์ ซึ่งคำสั่งส่วนใหญ่จะเกี่ยวกับการอ่านและเขียนข้อมูลสู่ IStorage ออบเจกต์
6. จัดการ IDataObject อินเตอร์เฟสสำหรับออบเจกต์ ในกรณีที่ต้องการคำสั่งในการจัดการกับคลิปบอร์ดด้วยออบเจกต์ข้อมูล
7. จัดการ IOleObject อินเตอร์เฟสสำหรับออบเจกต์ ในการทำงานในส่วนของกริยาของออบเจกต์
8. เพิ่มส่วนของการติดต่อกับผู้ใช้ที่เกี่ยวกับการเขียนออบเจกต์ที่ฝังตัวเพื่อแยก หรือทำให้บางส่วนของเมนูไม่ทำงาน เมื่อเข้าไปอยู่ในสถานะของการฝังตัว
9. แจ้งการเปลี่ยนแปลงของข้อมูล, การปิด, การเก็บข้อมูล รวมทั้งการเปลี่ยนชื่อตลอดอายุ ของออบเจกต์
10. (สำหรับฟูลเซิร์ฟเวอร์)ขยายคำสั่งที่เกี่ยวกับคลิปบอร์ดเพื่อสร้างCF_EMBED SOURCE และ CF_OBJECTDESCRIPTOR เพื่อการทำสำเนาจากเซิร์ฟเวอร์ และแปะออบเจกต์ฝังตัวลงในคอนเทนเนอร์
11. (ทางเลือกของฟูลเซิร์ฟเวอร์) ให้ทางเลือกแก่การติดต่อกับผู้ใช้สำหรับแอพพลิเคชันแบบหลายหน้าต่าง ในการเพิ่มเงื่อนไขในการเลิกทำงาน

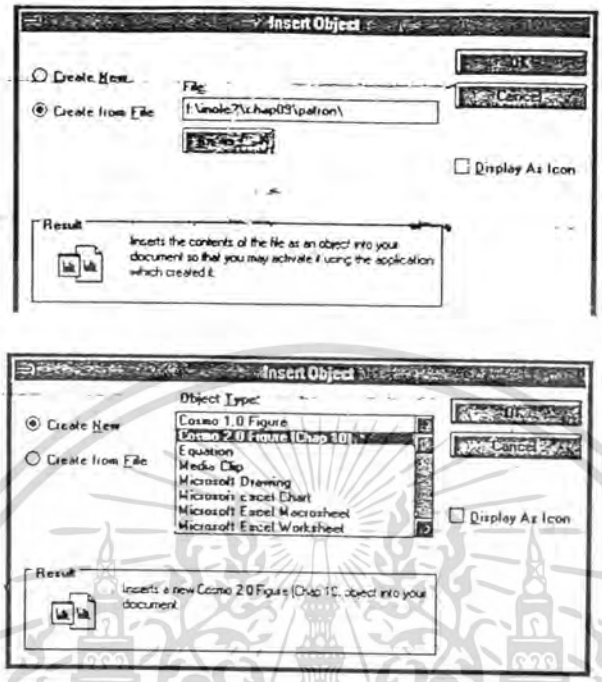
5.2.5 ตัวอย่างการทำงาน

สมมติให้แอพพลิเคชันคอนเทนเนอร์ชื่อ patron ซึ่งเป็นแอพพลิเคชันที่สามารถสร้างเอกสารเชิงซ้อนได้ และมีแถบเมนู (menu bar) ในรูป 5.13

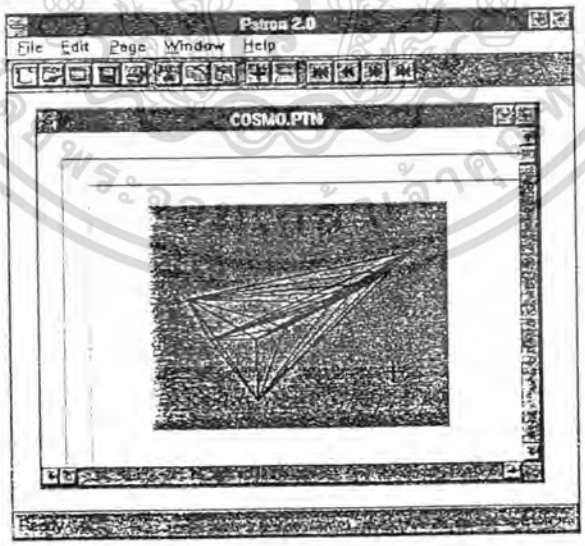


รูปที่ 5.13

และให้แอพพลิเคชันที่ชื่อ cosmo เป็นแอพพลิเคชันเซิร์ฟเวอร์ ซึ่งมีแถบเมนู (menu bar) เมื่อ patron ทำการ insert object (cosmo) รูป 5.14 ลงในเอกสารเชิงซ้อน ผลที่ได้จะแสดงไว้ที่ รูป 5.15

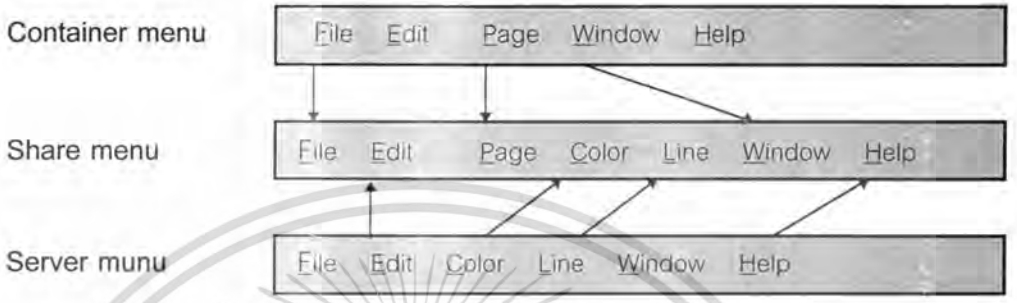


รูปที่ 5.14



รูปที่ 5.15

หากต้องการทำการเปลี่ยนแปลงรูปภาพ (เรียกแอพพลิเคชันเซิร์ฟเวอร์ขึ้นมาทำงาน) ก็เพียงแต่ทำการคลิ้ก 2 ครั้งติดกัน (double click)จะพบว่าแถบเมนูมีความเปลี่ยนแปลงเกิดขึ้น และจะพบว่าแถบเมนูใหม่นี้เกิดมาจากการรวมกันของเมนูคอนเทนเนอร์กับเมนูเซิร์ฟเวอร์ ดังรูป5.16



รูปที่ 5.16

สิ่งสำคัญ 2 สิ่งที่ต้องคำนึงในการสร้างออบเจกต์ in-place ก็คือ

- เมนูของแอพพลิเคชันเจ้าของออบเจกต์จะกลายไปเป็นเมนูร่วม ระหว่างเจ้าของออบเจกต์กับผู้สร้างออบเจกต์

- หน้าต่างของแอพพลิเคชันที่ถูกเรียกใช้งานจะมี parent ที่ไม่สามารถทราบล่วงหน้าว่าจะเป็นหน้าต่างของแอพพลิเคชันใด และยังไม่สามารถควบคุมได้อีกด้วย

ขั้นตอนในการทำงาน

ถ้าคอนเทนเนอร์ สนับสนุนการทำงานแบบ IN-PLACE ACTIVATION ออบเจกต์เริ่มกระบวนการ 3 ขั้นตอนดังต่อไปนี้

1. ย้ายหน้าต่างที่ออบเจกต์เขียนอยู่เข้าไปอยู่ในคอนเทนเนอร์

2. รวม (merge) เมนูของคอนเทนเนอร์ และเซิร์ฟเวอร์ไปเป็นเมนูที่ใช้ร่วมกันเพียงเมนูเดียวเรียกว่า "shared menu"

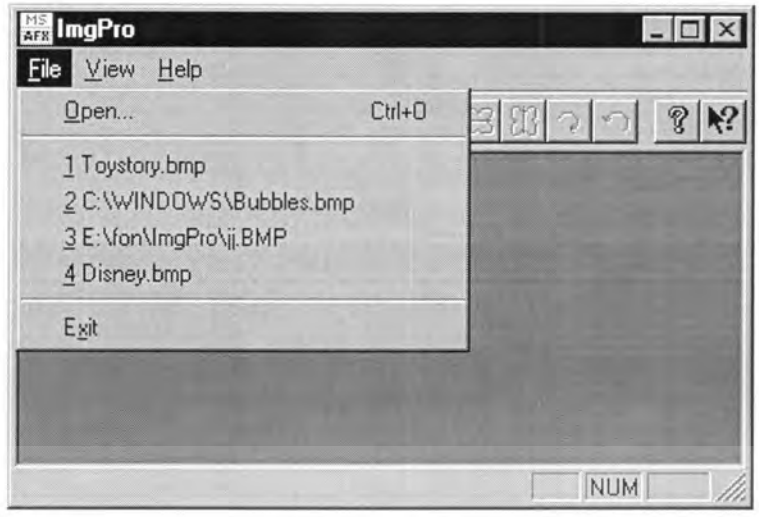
3. สร้าง editing tools สำหรับออบเจกต์ในหน้าต่างคอนเทนเนอร์

5.3 การใช้งานโปรแกรมการจัดการข้อมูลรูปภาพ

เนื่องจากแอพพลิเคชันนี้เป็นฟูลเซิร์ฟเวอร์แบบฝังตัวที่เลือกใช้ฟีเจอร์ การทำงานในสถานที่ เมื่อทำการรันโปรแกรมด้วยตัวเองจะปรากฏหน้าจอเซิร์ฟเวอร์ ดังต่อไปนี้

รูปที่ 5.17 หน้าจอเซิร์ฟเวอร์ที่ได้จากการรันโปรแกรม

จากหน้าจอโปรแกรมดังที่แสดงในรูป 5.17 นั้น ประกอบด้วยเมนู ซึ่งแต่ละส่วนมีหน้าที่แตกต่างกัน ดังจะแสดงต่อไปนี้



รูปที่ 5.18 ผลที่ได้จากการเลือกเมนูคำสั่ง File หรือ Alt+F

File มีหน้าที่จัดการเกี่ยวกับแฟ้มข้อมูล ดังนี้

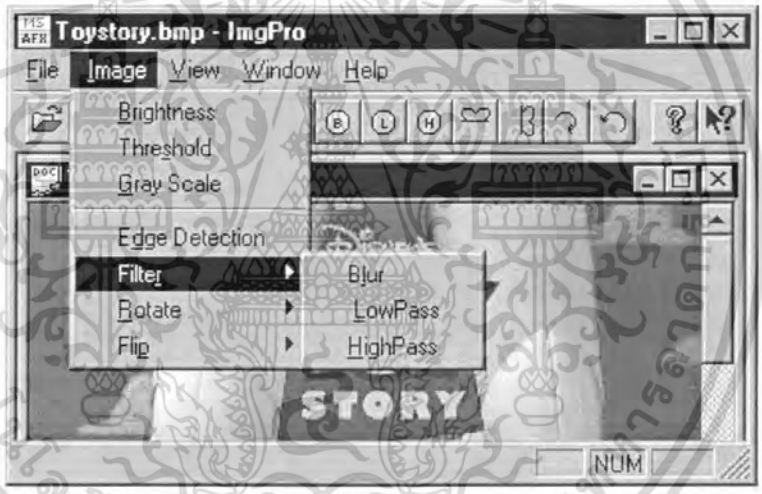
Open เปิดเอกสารหรือแบบเอกสารที่มีอยู่แล้ว และเมื่อทำการเปิดเอกสารแล้ว เมนูก็จะเปลี่ยนไป ดังรูปที่ 5.19

รูปที่ 5.19

Image มีหน้าที่จัดการเกี่ยวกับการปรับแต่งภาพ ดังนี้

Brightness	จะปรับความสว่างของรูปภาพ
Threshold	เปลี่ยนภาพสีให้เป็นภาพขาวดำ
Gray Scale	ปรับภาพขาวดำโดยไล่ระดับของสีเกรย์สเกล
Edge Detection	หาขอบเขตของภาพ
Filter	ขจัดสัญญาณรบกวนของภาพ โดยวิธีการ 3 วิธีดังนี้

- Blur
- Low Pass
- High Pass

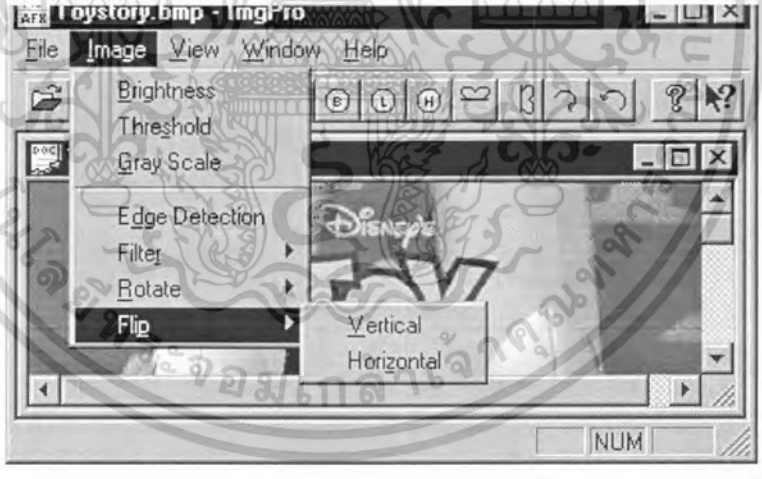


รูปที่ 5.20

Rotate จะทำการหมุนภาพ 90 ° ตามเข็มนาฬิกาหรือทวนเข็มนาฬิกา

รูปที่ 5.21

Flip การกลับภาพสามารถกลับภาพได้ตามแนวตั้งและแนวนอน



รูปที่ 5.22

View จะจัดการหน้าจอให้ปรากฏทูลบาร์หรือสเตตัส

ทูลบาร์ จะแสดงสถานะเปิดหรือปิดทูลบาร์

สเตตัส แสดงสถานะเปิดหรือปิดสเตตัสบาร์

ดังรูปที่5.23

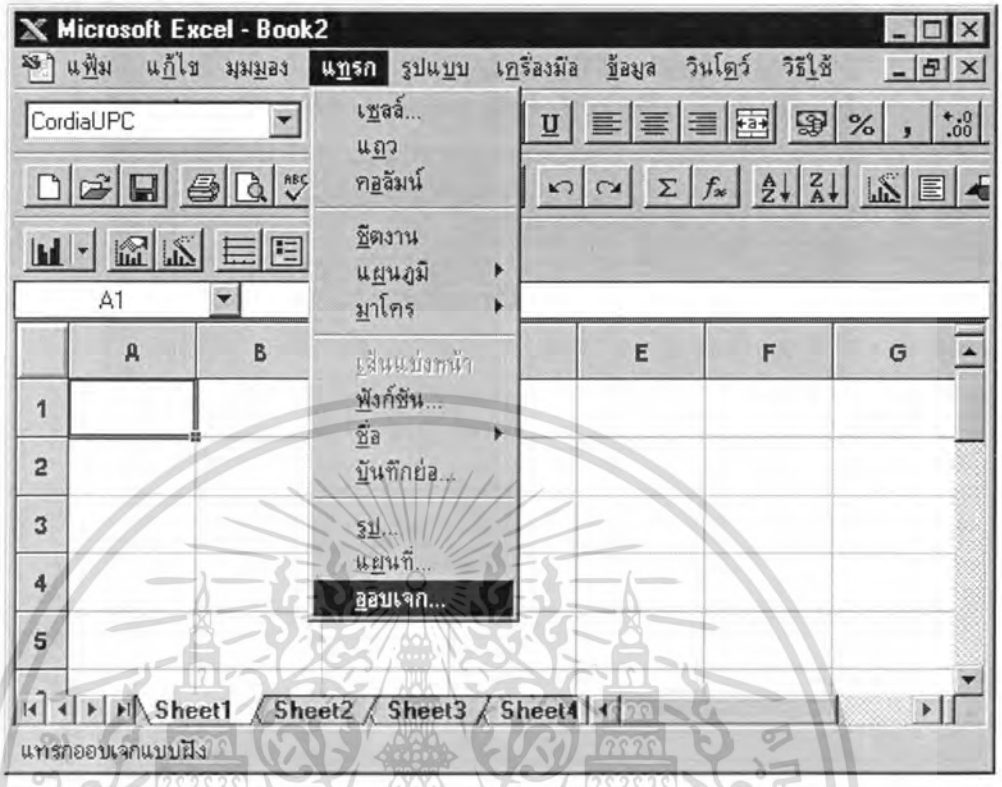
รูปที่ 5.23

Window เป็นการจัดการหน้าจอที่มีการเปิดออกมาหลายๆหน้าต่าง มีการจัดเรียงไอคอน

รูปที่ 5.24

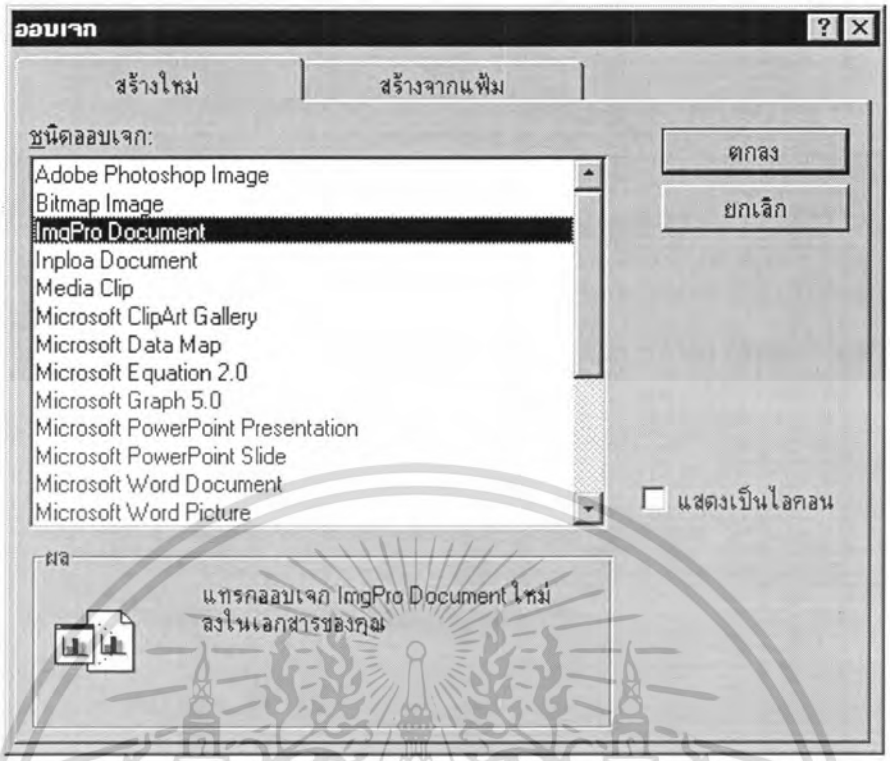
Help แสดงเมนู Help เป็นคำแนะนำเกี่ยวกับตัวโปรแกรม

รูปที่ 5.25

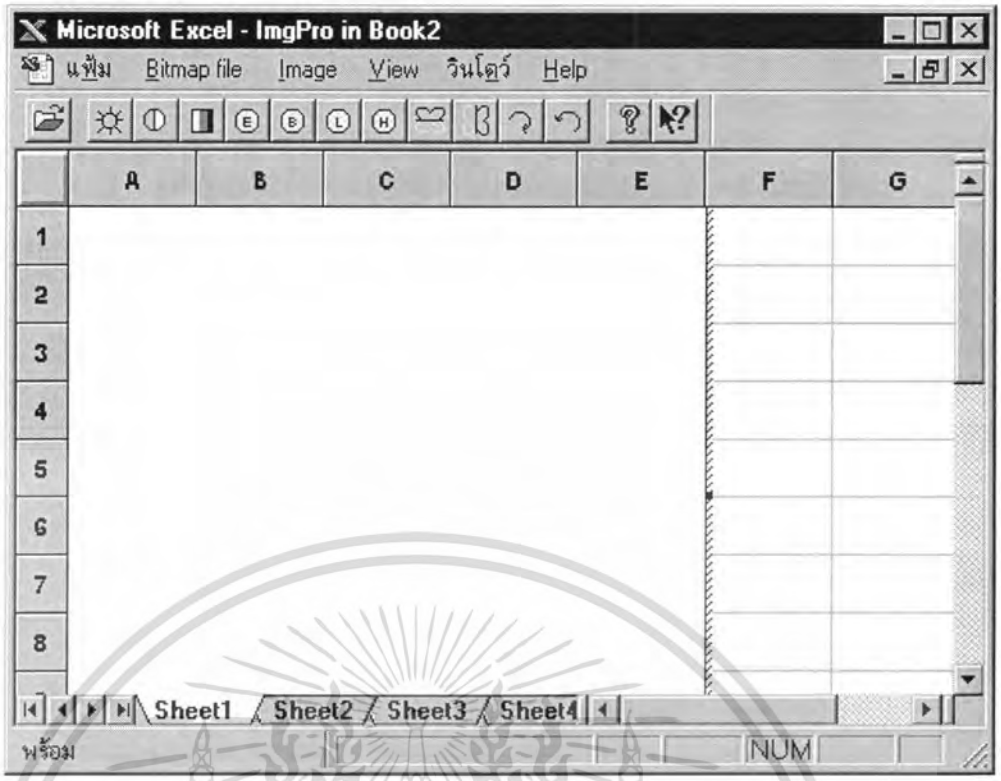


รูปที่ 5.26

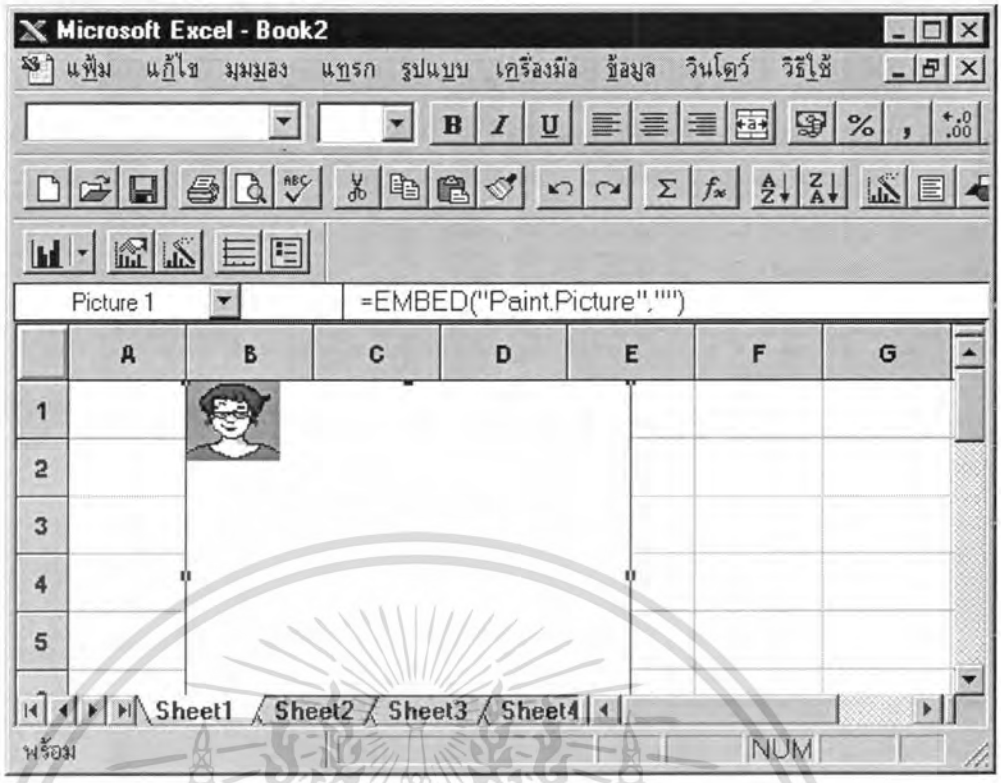
และเมื่อโปรแกรมอื่นได้แทรกออบเจกต์นี้เข้าไปแล้ว จะปรากฏหน้าต่างให้เลือกให้เลือกว่าจะแทรกออบเจกต์ใด ดังต่อไปนี้



รูปที่ 5.27



รูปที่ 5. 28 ภาพแสดงเมื่อเราได้ทำการแทรกออบเจกต์เข้าไป



รูปที่ 5. 29 ผลที่ได้หลังจากการรันโปรแกรมฝังตัว

บทที่ 6
สรุปผลและเสนอแนะ

สรุปผล

ผลที่ได้จากการพัฒนาแอพพลิเคชันทางด้านการประมวลผลภาพโดยใช้เทคนิค OLE 2.0 จะได้แอพพลิเคชันเซิร์ฟเวอร์ทางด้านการประมวลผลภาพที่ทำงานบนระบบปฏิบัติการวินโดวส์ โดยแอพพลิเคชันอื่นที่สามารถเรียกใช้แอพพลิเคชันนี้คือ แอพพลิเคชันที่สนับสนุนเทคโนโลยีของ OLE และเป็นคอนเทนเนอร์ ยกตัวอย่างเช่น ไมโครซอฟท์เวิร์ด และไมโครซอฟท์เอกเซล โดยตัวแอพพลิเคชันสามารถทำงานในการแก้ไขและดัดแปลงภาพได้ดังต่อไปนี้คือ การปรับสีและความสว่างของภาพ การสร้างภาพขาวดำ การทำภาพมัว การขจัดสัญญาณรบกวนของภาพ การหาขอบของภาพ การกลับภาพ และการหมุนภาพ โดยสามารถใช้ได้กับไฟล์ข้อมูลรูปภาพแบบบิตแมพ

ข้อจำกัดของระบบคือ แอพพลิเคชันต้นแบบนี้สามารถทำงานบนระบบปฏิบัติการวินโดวส์ 95 ได้ตามที่ได้ทดสอบ แต่ในระบบปฏิบัติการวินโดวส์ NT เท่าที่ทดสอบการทำงานของแอพพลิเคชันต้นแบบนี้จะพบว่าการจัดการในเรื่องการใช้งานหน่วยความจำนั้นตัววินโดวส์ NT ทำได้ดีมากทำให้ไม่ค่อยพบปัญหาในโปรแกรม และเนื่องจากเป็นแอพพลิเคชันที่ค่อนข้างใช้หน่วยความจำมาก จึงจะมีปัญหาเรื่องการทำงานช้าถ้าหากหน่วยความจำในเครื่องน้อย

นอกจากนี้ ประเภทของไฟล์ข้อมูลรูปภาพที่ใช้อยู่ในปัจจุบันมีหลายประเภทไม่ใช่มีเพียงบิตแมพเท่านั้น แต่ในแอพพลิเคชันนี้ต้องการศึกษาเน้นในเรื่องของเทคนิคการพัฒนาแอพพลิเคชันโดยใช้ OLE 2.0 จึงได้กำหนดเพียงว่าแอพพลิเคชันจะสามารถทำได้กับข้อมูลรูปภาพประเภทบิตแมพเท่านั้น ดังนั้นจึงไม่สามารถใช้กับไฟล์อื่นได้ ถ้าหากเป็นแอพพลิเคชันทางด้านการประมวลผลรูปภาพที่สมบูรณ์ควรจะกำหนดให้ใช้ได้กับไฟล์หลายๆประเภทเพื่อให้ผู้ใช้เลือกใช้ได้สะดวกยิ่งขึ้น

ข้อเสนอแนะ

1. จากที่ได้กล่าวไปแล้วในเรื่องของข้อจำกัด ถ้าหากจะทำการพัฒนาแอพพลิเคชันนี้ต่อควรจะเพิ่มประเภทของไฟล์ข้อมูลรูปภาพหลายๆประเภท เช่น รูปแบบกิฟ (GIF) รูปแบบทีจีเอ เป็นต้น

2. เนื่องจากแอพพลิเคชันต้นแบบนี้ได้ออกแบบโดยเขียนโปรแกรมเชิงวัตถุ (OOP) ดังนั้นในการพัฒนาต่อ จึงสามารถนำคลาสที่ได้ออกแบบไว้แล้วนี้ ไปประยุกต์ใช้กับระบบที่จะพัฒนาต่อไปได้

3. การพัฒนาแอพพลิเคชันทางด้านการประมวลผลภาพ ถ้าต้องการจะให้สมบูรณ์ยิ่งขึ้น ในการพัฒนาต่อในอนาคต น่าจะให้มีอัลกอริทึมในการดัดแปลงแก้ไขภาพแต่ละวิธีหลายๆ อัลกอริทึมเพื่อให้เปรียบเทียบกัน เนื่องจากลักษณะของภาพบางอย่างอาจจะใช้ได้ดีกับอัลกอริทึมบางอย่าง จึงควรจะให้ผู้ใช้ได้เลือกทดสอบจากหลายๆ อัลกอริทึมเพื่อเลือกใช้อัลกอริทึมที่ดีที่สุดที่เหมาะสมกับข้อมูลรูปภาพของผู้ใช้ได้

4.สามารถนำแอพพลิเคชันต้นแบบนี้ไปเป็นเอกสารอ้างอิงของเทคโนโลยี OLE เนื่องจากเทคโนโลยีนี้ยังไม่แพร่หลายนักในประเทศไทยโดยเฉพาะในส่วนของเซิฟเวอร์

ภาคผนวก

การติดตั้งระบบ

ฮาร์ดแวร์

- เครื่องคอมพิวเตอร์ที่มี CPU ตั้งแต่รุ่น DX2-66 ขึ้นไป
- มีหน่วยความจำหลักตั้งแต่ 8 MB ขึ้นไป
- จอภาพแบบซุปเปอร์วีจีเอ ที่สามารถทำงานที่ความละเอียด 800*600 ที่ 256 สี ได้

ซอร์ฟแวร์

- ระบบปฏิบัติการแบบ 32 บิต เช่น Windows 95
- โปรแกรมไมโครซอฟต์ออฟฟิศ

ไฟล์ที่ใช้

- ImgPro.exe
- Microsoftword
- MicrosoftExcel
- ไฟล์ที่เป็นรูปภาพ ฟอร์แมต bmp 256 สี

บรรณานุกรม

1. Kraig Brockschmidt, Inside OLE 2, Microsoft Press , United Status of America ,1993
2. Rob Krumm, Mading Windows applications work togerher , M&T Books, NewYork, 1994
3. Richard C. Leinecker & Jamie Nye, Visual C++ Power Toolkit , Ventana Press , United States of America, 1995
4. Bryan Waters, Writing Windows Applications With Microsoft Foundation Classes , M&T Books, NewYork, 1994
5. Steve Holzer, Heavy Metal Visual C++ Programming , Programmer Press, United States of America, 1994
6. Adrian Low, Introductory Computer Vision and Image Processing , United States of America, 1993
7. Steve Rimmer, Windows Bitmapped Graphics , Windcrest/McGraw-Hill, United States of America, 1993
8. Ori Gurewich & Nathan Gurewich, Master Visual C++ 2, Sams Publishing United States of America, 1994