OLE มัลติมีเดียกับการเขียนโปรแกรมแบบ OOP บนวินโดว์

โดย

นายมานะชัย พรหมบัวดี รหัส 35504118
นางสาวสุดารัตน์ นรากร รหัส 35504126
นายอานนท์ กิจถาวรวงศ์ รหัส 35504131

ปัญหาพิเศษฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

สาขาคณิตศาสตร์ประยุกต์
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2538

OLE MULTIMEDIA WITH OOP PROGRAMMING FOR WINDOWS

BY

MR. MANACHAI	PROMBUADEE	CODE 35504118
MISS SUDARAT	NARAKORN	CODE 35504126
MR. ARNON	KITTAWORNWONG	CODE 35504131

A Special Project Submitted in Partial Fulfillment
of the Requirement of the Degree of Bachelor of Science

Department of Mathematics and Computer Science
Faculty of Science
King Mongkut's Institute of Technology Chaokhuntaharn
Lardkrabang

1995

ปัญหาพิเศษเรื่อง	OLE มัลติมีเดียกับการเขียนโปรแกรมแบบ OOP บนวินโดว์		
ชื่อนักศึกษา	1. นายมานะชัย	พรหมบัวดี	รหัส 35504118
	2. นางสาวสุดารัตน์	นรากร	รหัส 35504126
	3. นายอานนท์	กิจถาวรวงศ์	รหัส 35504131
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์		
อาจารย์ที่ปรึกษา	อาจารย์วีระชัย ตันยะสิทธิ์		

ปัญหาพิเศษฉบับนี้ กรรมการสอบปัญหาพิเศษได้ตรวจพิจารณาแล้วจึงอนุมัติให้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต สาขาคณิตศาสตร์ประยุกต์ ประจำปีการศึกษา 2537

(รองศาสตราจารย์ภัคคินี ชิตสกุล) หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะกรรมการโครงงานพิเศษ

(อาจารยพัชญา พิทักษ์ไพรวัน) ประธานกรรมการ

(อาจารย์สิริลักษณ์ เตียพิริยะกิจ) กรรมการ

(อาจารย์วีระชัย ตันยะสิทธิ์) กรรมการและอาจารย์ที่ปรึกษา

บทคัดย่อ

ปัญหาพิเศษฉบับนี้ได้จัดทำขึ้นโดยมีวัตถุประสงค์คือ นำหลักการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) และหลักการของ OLE (Object Linking and Embeding) มาใช้ในการพัฒนาสไลด์ประกอบเสียงซึ่งเป็นโปรแกรมแบบ OLE-Container โปรแกรมที่พัฒนาขึ้นมานี้มีระบบความช่วยเหลือแบบ Context-Sensitive help มีการใช้ Toolbar และ Status bar เพื่อความสะดวกในการใช้งาน โปรแกรมจะแบ่งออกเป็น 3 ส่วนหลักคือ การแสดงภาพแบบสไลด์ การเปิดไฟล์เสียง wave และ midi , ระบบความช่วยเหลือแบบ Context-Sensitive help

ผลจากปัญหาพิเศษจะได้แอพพลิเคชันที่สามารถสร้างสไลด์ประกอบเสียงที่มีระบบความช่วยเหลือแบบ Context-Sensitive help ซึ่งเป็น OLE แบบ Container

ABSTRACT

The purpose of this special project is to using Object Oriented Programming (OOP) structure and theory of Object Linking and Embeding (OLE) to develop program OLE-Container such that can build Slide with Sound on window. This program has Context-Sensitive help , Toolbar and Status bar for comfortable to use. This program is separated in three modules, show slide picture, play sound wave and midi, Context-Sensitive help.

The result of this project is an application that can build slide with sound and it has Context-Sensitive help

กิตติกรรมประกาศ

ปัญหาพิเศษฉบับนี้สำเร็จลงได้ด้วยดีก็เพราะหลายเหตุปัจจัย โดยเฉพาะอย่างยิ่ง

รองศาสตราจารย์ภัคคินี ชิตสกุล

อาจารย์วีระชัย ตันยะสิทธิ์

อาจารย์พัชญา พิทักษ์ไพรวัน

อาจารย์สิริลักษณ์ เตียพิริยะกิจ

ที่ได้ให้แนวทางในการวิจัย ตลอดจนคำปรึกษาอันก่อให้เกิดแนวความคิดที่สามารถแก้ปัญหาต่าง ๆ ที่เกิดขึ้นในระหว่างการทำการวิจัย นอกจากนี้ยังช่วยแนะนำแนวทางในการดำเนินงานและตรวจทางแก้ไขด้วยความเอาใจใส่เป็นอย่างดียิ่ง

ขอบคุณเจ้าหน้าที่ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ทุกท่านที่สนับสนุนในการใช้ห้องปฏิบัติการคอมพิวเตอร์ และให้ความสะดวกในการเบิกอุปกรณ์ต่าง ๆ ที่ใช้ในการวิจัย

คณะผู้จัดทำขอกราบขอบพระคุณอาจารย์ทุกท่านที่ได้ประสาทวิชาความรู้ทั้งในภาคทฤษฎีภาคปฏิบัติแก่ผู้จัดทำ จนกระทั่งงานวิจัยสัมฤทธิ์ผลได้ดีทุกประการ

ขอขอบคุณคุณพ่อ คุณแม่และเพื่อน ๆ ที่คอยเป็นกำลังใจจนงานลุล่วงไปได้ด้วยดี โดยเฉพาะอย่างยิ่งเพื่อนหนึ่ง เพื่อนเชษฐ์ และเพื่อนตั้ม ที่ให้คำปรึกษามากมาย

ขอขอบพระคุณ

คณะผู้จัดทำ

สารบัญรูปภาพ

สารบัญ

บทที่ 1
บทนำ

1.1 ความเป็นมาของปัญหาพิเศษ

ปัจจุบันพัฒนาการด้านซอฟท์แวร์มีความก้าวหน้ามาก จะเห็นว่าซอฟท์แวร์ใหม่ ๆ ที่ออกมาจะมีความสามารถมากขึ้นในการใช้งาน และอำนวยความสะดวกแก่ผู้ใช้มากขึ้นเช่นกัน ขณะเดียวกันทางด้านผู้พัฒนาโปรแกรมก็ได้หาวิธีการและเทคนิคใหม่ ๆ เพื่อพัฒนาซอฟท์แวร์ และเพื่อที่ให้การพัฒนานั้นเป็นไปได้รวดเร็วและง่ายขึ้น การเขียนโปรแกรมแบบใหม่จึงได้เกิดขึ้น ซึ่งรู้จักกันในชื่อ OOP (Object-Oriented Programming) นอกจากนั้นเพื่ออำนวยความสะดวกแก่ผู้ใช้และเพื่อเพิ่มความสามารถของซอฟท์แวร์ จึงได้เกิดมาตรฐาน ต่าง ๆ ที่จะทำให้แอพพลิเคชันที่สร้างขึ้นสามารถแลกเปลี่ยนข้อมูลกันได้ และ OLE (Object Linking and Embedded) ก็เป็นมาตรฐานหนึ่งที่มีการพัฒนาขึ้นมาและได้รับการยอมรับเป็นอย่างดี จะเห็นจากซอฟท์แวร์ใหม่ ๆ ที่ออกมาภายใต้ระบบปฏิบัติการวินโดว์จะสนับสนุนคุณสมบัตินี้แทบทั้งสิ้น

จากที่กล่าวมาวินโดว์เป็นระบบปฏิบัติการที่สนับสนุนการทำงานของซอฟท์แวร์ที่สนับสนุน OLE และ OOP เป็นการเขียนโปรแกรมที่สามารถพัฒนาโปรแกรมให้สนับสนุนมาตรฐาน OLE นี้ได้ง่ายขึ้น เพราะใช้หลักการของออบเจกต์เหมือนกัน และ ปัจจุบันนี้วินโดว์เป็นระบบปฏิบัติการที่ได้รับความนิยมมากอาจจะกล่าวได้ว่า ใครที่รู้จักคอมพิวเตอร์ก็ต้องรู้จักวินโดว์ เพราะมีแอพพลิเคชันที่รันภายใต้วินโดว์ให้ใช้มากและการใช้งานก็สะดวกนั่นเอง

ปัจจุบันการทำงานหลายสาขาจะมีคอมพิวเตอร์เข้าไปเกี่ยวข้องไม่ขั้นตอนใดก็ขั้นตอนหนึ่งของกระบวนงาน ดังนั้นการประยุกต์การใช้งานคอมพิวเตอร์ให้สนับสนุนการทำงานหลากหลายจึงเกิดเป็นความจำเป็น และเป็นที่มาของระบบคอมพิวเตอร์แบบหลายสื่อ(multimedia computers) ซึ่งพื้นฐานหลักของระบบคอมพิวเตอร์แบบนี้ก็คือ ภาพ และ เสียง และในปัจจุบันนี้ก็มีแอพพลิเคชันจำนวนมากที่สนับสนุนการทำงานแบบหลายสื่อนี้

การพัฒนาแอพพลิเคชันเพื่อให้มีความสามารถดังที่กล่าวมาต้องอาศัยเครื่องมือที่มีประสิทธิภาพ และอำนวยความสะดวกในการเขียนโปรแกรม เนื่องมาจากการพัฒนาแอพพลิเคชันบนวินโดว์เป็นสิ่งที่ยุ่งยากสำหรับโปรแกรมเมอร์ ดังนั้นเพื่อให้การพัฒนาแอพพลิเคชันมีประสิทธิภาพ และสามารถสนับสนุนมาตรฐาน OLE ได้ จึงมีการพัฒนาภาษาโปรแกรมขึ้นมาให้สามารถใช้เป็นเครื่องมือในการพัฒนาแอพพลิเคชันได้ง่ายขึ้น และ Visual C++ ก็เป็นภาษาคอมพิวเตอร์ที่เหมาะที่จะใช้เป็นเครื่องมือในการพัฒนาแอพพลิเคชันบนวินโดว์ เพราะเป็นภาษาที่ออกแบบมาเพื่อการเขียนโปรแกรมแบบ OOP อย่างแท้จริง และมีเครื่องมือที่มีประสิทธิภาพที่สนับสนุนการพัฒนาแอพพลิเคชันบน

วินโดว์เช่น AppWizard, ClassWizard, AppStudio, MFC (Microsoft Foundation Class) เป็นต้น นอกจากนั้น ยังสนับสนุน OLE, Context Sensitive Help อีกด้วย

จากที่กล่าวมาข้างต้นจึงเกิดมีแนวความคิดที่จะเรียนรู้หลักการ ทฤษฎี และ วิธีการพัฒนาโปรแกรมที่สนับสนุนการทำงานดังกล่าว จึงเป็นที่มาของปัญหาพิเศษ ภายใต้หัวข้อที่ว่า **" OLE มัลติมีเดียกับการเขียนโปรแกรมแบบ OOP บนวินโดว์ "**

1.2 วัตถุประสงค์ของปัญหาพิเศษ

1. เพื่อศึกษาทฤษฎีและการเขียนโปรแกรมที่ทำงานสนับสนุน OLE-Container
2. เพื่อศึกษาทฤษฎีและการเขียนโปรแกรมแบบ OOP (Object-Oriented Programming)
3. เพื่อพัฒนาโปรแกรมต้นแบบการทำสไลด์ประกอบเสียงบนวินโดว์
4. เพื่อให้เกิดทักษะ และประสบการณ์ด้านการพัฒนาโปรแกรม

1.3 ขอบเขตของปัญหาพิเศษ

1. โปรแกรมสามารถทำงานสนับสนุน OLE แบบ Container ได้
2. โปรแกรมมีความสามารถในการทำสไลด์ประกอบเสียงได้
3. โปรแกรมมีระบบความช่วยเหลือแบบ Context-Seniitive Help
4. โปรแกรมนี้จะพัฒนาภายใต้โปรแกรมไมโครซอฟท์วินโดว์ 95
5. เครื่องมือที่ใช้ในการพัฒนาโปรแกรมคือ โปรแกรมไมโครซอฟท์ Visual C++ 1.5& 2.0
6. รูปแบบของข้อมูลรูปภาพจะต้องมาจากโปรแกรมที่สนับสนุน OLE-Container
7. โปรแกรมมีความสามารถเล่นไฟล์เสียง 2 แบบ คือ .wav และ .mid

1.4 ขั้นตอนในการทำงาน

1. ศึกษาโปรแกรมไมโครซอฟท์ Visual C++ เวอร์ชัน 1.5และ2.0, การเขียนโปรแกรมแบบ OOP (Object-Oriented Programming) และ OLE (Object Linking and Embeded)
2. ออกแบบการทำงานของโปรแกรม และ ขอบเขตความสามารถของโปรแกรม
3. พัฒนาโปรแกรมตามที่ได้ออกแบบไว้
4. ทดสอบการทำงานของโปรแกรม
5. แก้ไขข้อบกพร่องของโปรแกรม และ เพิ่มเติมความสามารถของโปรแกรมให้สมบูรณ์
6. ทำเอกสารประกอบการออกแบบ และ การใช้งาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เรียนรู้หลักการ ทฤษฎี ของ OOP, OLE และ การพัฒนาแอพพลิเคชันบนวินโดว์

2. ได้โปรแกรมที่ทำงานสนับสนุน OLE-Container และ สามารถทำสไลด์ประกอบเสียงได้
3. เป็นแนวทางสำหรับผู้ที่ต้องการศึกษา และ พัฒนาโปรแกรมในด้านที่เกี่ยวข้องได้

บทที่ 2
ความหมายและทฤษฎี

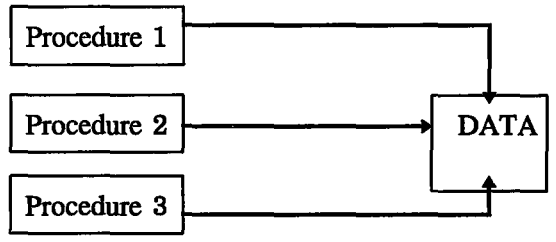
2.1 ทฤษฎีการโปรแกรมเชิงวัตถุ

2.1.1 ความหมายของการโปรแกรมเชิงวัตถุ

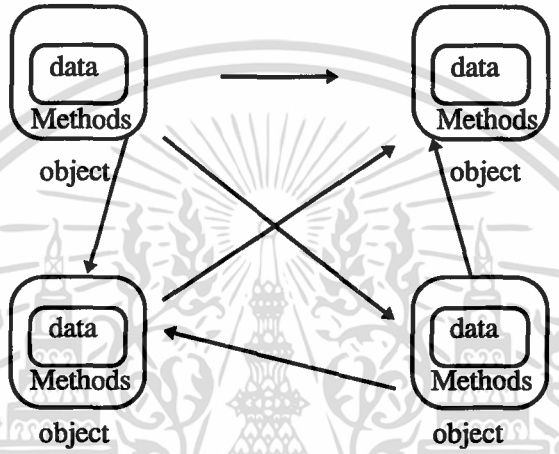
การโปรแกรมเชิงวัตถุ (Object-Oriented Programming หรือ OOP) เป็นวิธีการเขียนโปรแกรมแบบหนึ่งที่อ้างอิงแนวความคิดแบบเชิงวัตถุ (Object-oriented) โดยจะมองโปรแกรมเป็นระบบที่สนใจ และมีออบเจกต์เป็นสมาชิกในโปรแกรม ในออบเจกต์หนึ่งจะประกอบไปด้วยส่วนโค้ดหรือรูทีนจัดการต่าง ๆ และส่วนดาต้าได้แก่ ตัวแปรในโปรแกรม

ในลักษณะการเขียนโปรแกรมแบบเดิมนั้น จะพยายามแบ่งงานทั้งหมดเป็นงานเล็ก ๆ แบ่งโปรแกรมออกเป็นโพรซีเยอร์หรือรูทีนย่อย เพื่อจัดการส่วนต่าง ๆ ในโปรแกรมดังนั้นคำว่าโปรแกรมในนิยามแบบเดิมจึงหมายถึงเซตของโพรซีเยอร์ เมื่อมองโปรแกรมเป็นเซตของโพรซีเยอร์แล้ว ขั้นตอนการออกแบบโปรแกรม จึงเป็นขั้นตอนของการจัดโครงสร้างของโพรซีเยอร์ต่าง ๆ ของโปรแกรม ตามลำดับก่อนหลังจากใหญ่ไปหาเล็ก เรียกว่าการออกแบบจากบนลงล่าง (Top-down design) ซึ่งจะเห็นว่ามีข้อบกพร่องคือการเขียนโปรแกรมเดิม ไม่อาจแสดงการใช้งานข้อมูลสำคัญต่าง ๆ ที่มีในโปรแกรมได้อย่างเป็นระเบียบดี กล่าวคือ ทุก ๆ โพรซีเยอร์ในโปรแกรมมีสิทธิใช้งานข้อมูลทุกตัวเมื่อไรก็ได้ โดยที่ไม่ทราบอย่างแท้จริงว่า ข้อมูลตัวใดมีความสำคัญกับโพรซีเยอร์หรือรูทีนใดบ้าง นั้นก็คือ การเขียนโปรแกรมแบบเดิมไม่สามารถ ควบคุมการใช้งานของข้อมูลในโปรแกรมได้

แต่สำหรับในการเขียนโปรแกรมเชิงวัตถุนั้น โปรแกรมมีนิยามเป็นเซตของออบเจกต์โดยที่ออบเจกต์ถูกกำหนดให้เป็นหน่วยใหม่ที่สร้างขึ้นมา เพื่อรวมทั้งส่วนข้อมูลและโค้ดที่จัดการกับข้อมูลนั้นไว้ด้วยกัน เมื่อนิยามของการโปรแกรมเปลี่ยนไปการออกแบบโปรแกรมจึงกลายเป็นการออกแบบโครงสร้าง และความเกี่ยวข้องระหว่างออบเจกต์แต่ละตัวในโปรแกรมแทน



Object-Oriented
Paradigm



รูปที่ 2.1.1 แสดงให้เห็นโครงสร้างของโปรแกรมตามนิยามทั้งสองแบบ

จากรูป เป็นการเปรียบเทียบการเขียนโปรแกรมแบบเดิม ที่มีการแบ่งโปรแกรมออกเป็นโพรซีเยอร์หรือรูทีนย่อย เพื่อจัดการส่วนต่าง ๆ ในโปรแกรม โดยที่ทุก ๆ โพรซีเยอร์ในโปรแกรมมีสิทธิใช้งานข้อมูลทุกตัวเมื่อไหร่ก็ได้ เมื่อเปรียบเทียบกับการเขียนโปรแกรมเชิงวัตถุ ที่มองแต่ละส่วนของโปรแกรมเป็นออบเจกต์ โดยที่แต่ละออบเจกต์มีวิธีการและข้อมูลของออบเจกต์ตัวนั้น ๆ เอง นอกจากนี้ยังสามารถเรียกใช้วิธีการและข้อมูลของออบเจกต์ตัวอื่น ๆ ได้

เมื่อมีการนิยามโปรแกรมขึ้นใหม่จึงมีการสร้างกฎข้อบังคับต่าง ๆ ขึ้นมา สำหรับรองรับโครงสร้างของนิยามใหม่ กฎต่าง ๆ เหล่านี้จะอยู่ในรูปของหลักการสำคัญต่าง ๆ ของการโปรแกรมเชิงวัตถุที่มีหลายประการด้วยกัน อันประกอบไปด้วยคลาส (Class) เอนแคปซูเลชัน (Encapsulation) คุณสมบัติการสืบทอด (Inheritance) โพลีมอร์ฟิซึม (Polymorphism) และอื่น ๆ ซึ่งจะกล่าวต่อไป

ชนิดข้อมูลนามธรรม (Abstract data type) เป็นหลักของการโปรแกรมเชิงวัตถุ ที่มีการรวมชนิดข้อมูล (Type) และการปฏิบัติการ (Operation) เข้าไว้ด้วยกัน

การนิยามคลาส (Class Definition) คือการกำหนดการทำงานของชนิดข้อมูลนามธรรมโดยการนิยามวิธีการปฏิบัติการของชนิดข้อมูลนั้น นอกจากนี้การนิยามคลาสยังกำหนดโครงสร้างข้อมูลของชนิดข้อมูลด้วย โดยปกติแล้วโครงสร้างข้อมูลนี้จะเข้าถึงได้เฉพาะภายในคลาส เรียกว่าเป็น ชนิดข้อมูลแบบท้องถิ่น (private) แต่ถ้าข้อมูลนี้เข้าถึงได้จากภายนอกคลาส เรียกว่าเป็น ชนิดข้อมูลแบบทั่วไป (public)

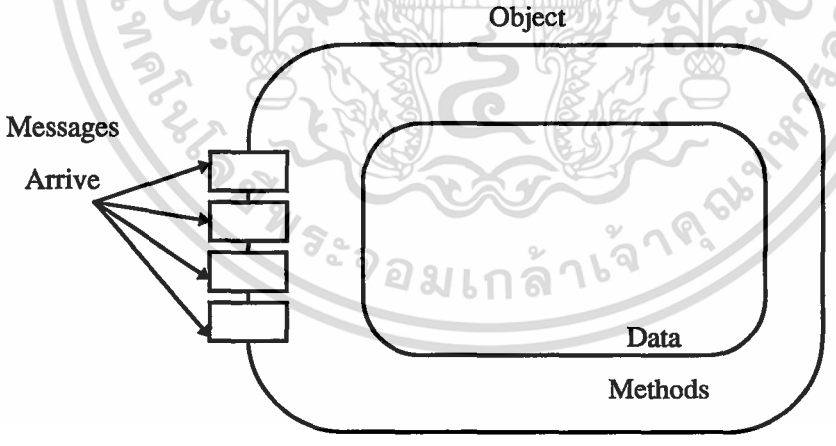
ตัวอย่างเช่น การนิยามคลาส จำนวนเต็ม ชนิดของข้อมูล คือ จำนวนเต็ม (integer) มีการปฏิบัติการคือ บวก ลบ คูณ หาร ดังนั้น ออบเจกต์ในคลาสคือ 1,2,3,4,... หรือการ นิยามคลาสจำนวนนับ มีการปฏิบัติการคือจำนวนนับที่เป็นเลขคู่ ดังนั้นออบเจกต์ในคลาส คือ 2,4,6,8,...

วิธีการ (method) คือ การนำชนิดข้อมูลมาใช้งาน ตามวิธีการการตอบสนองต่อ message การเรียกใช้วิธีการทำได้โดยส่ง messege ไปยัง ออบเจกต์

การใช้งานของชนิดข้อมูล (type) สามารถแบ่งได้เป็น 2 แบบ คือ แบบทั่วไป (public) และแบบท้องถิ่น (private)

- การทำงานแบบทั่วไป สามารถเรียกใช้ได้จากภายนอกคลาส
- การทำงานแบบท้องถิ่น สามารถเรียกใช้ได้เฉพาะภายในคลาส

ออบเจกต์ ในความหมายเชิงวิชาการนิยามว่า คือ ปริมาณหนึ่งในระบบที่ประกอบขึ้นด้วยองค์ประกอบ 2 ส่วนคือ ข้อมูลและโค้ดโปรแกรม ส่วนข้อมูลใช้เก็บสถานะของตัวเองเรียกว่า ข้อมูล (data) และส่วนโค้ดโปรแกรมใช้ในการตอบสนองต่อออบเจกต์ตัวอื่นในระบบเดียวกัน เรียกว่า วิธีการ (method)



รูปที่ 2.1.2 โครงสร้างความสัมพันธ์ระหว่างออบเจกต์ ข้อมูล message และวิธีการ

ตัวอย่างออบเจกต์ เช่น โปรแกรมไมโครซอฟท์เวิร์ดสำหรับวินโดว์ก็เป็นออบเจกต์หนึ่งของวินโดว์

2.1.2 การทำงานของออบเจกต์

ออบเจกต์ใด ๆ ในระบบ จะสื่อสารกับออบเจกต์อื่นเพื่อให้บรรลุความต้องการของตนการสื่อสารนี้เป็นลักษณะ " ร้องขอและตอบสนอง " โดยที่สามารถเรียกใช้วิธีการซึ่งนิยามไว้ในการ นิยามคลาส เพื่อให้เกิดการทำงานกับออบเจกต์ การเรียกใช้ทำได้โดยการส่ง message ไปยัง ออบเจกต์ที่ต้องการ แล้วตัวออบเจกต์จะตีความหมายของ message ออกมาว่าควรจะตอบสนองต่อ message นั้นด้วยวิธีใด นั่นคือ เมื่อออบเจกต์หนึ่งขอความช่วยเหลือจากอีกออบเจกต์หนึ่งเรียกว่ากำลังส่ง message ไปยังออบเจกต์อื่น

ตัวอย่างเช่น ในขณะที่กำลังใช้ไมโครซอฟท์วินโดว์อยู่นั้น ผู้ใช้คือออบเจกต์หนึ่งในระบบในขณะที่วินโดว์ก็เป็นอีกออบเจกต์หนึ่ง เมื่อผู้ใช้มีการคลิกเมาส์ นั่นคือ กำลังส่ง message ให้กับวินโดว์เพื่อรอรับผลตอบสนอง เมื่อมีการรับ message ออบเจกต์วินโดว์ต้องพิจารณาว่าจะใช้วิธีการใดในการตอบสนอง และเมื่อวินโดว์ตอบสนองก็จะหมายความว่า เป็นการตอบสนองต่อ message ที่ผู้ใช้ร้องขอไปด้วย วิธีการที่สร้างขึ้นเพื่อรองรับ message นั้น และเมื่อมีการตอบสนอง message ครั้งหนึ่ง ก็จะมีผลทำให้สถานภาพของวินโดว์เปลี่ยนแปลงไป

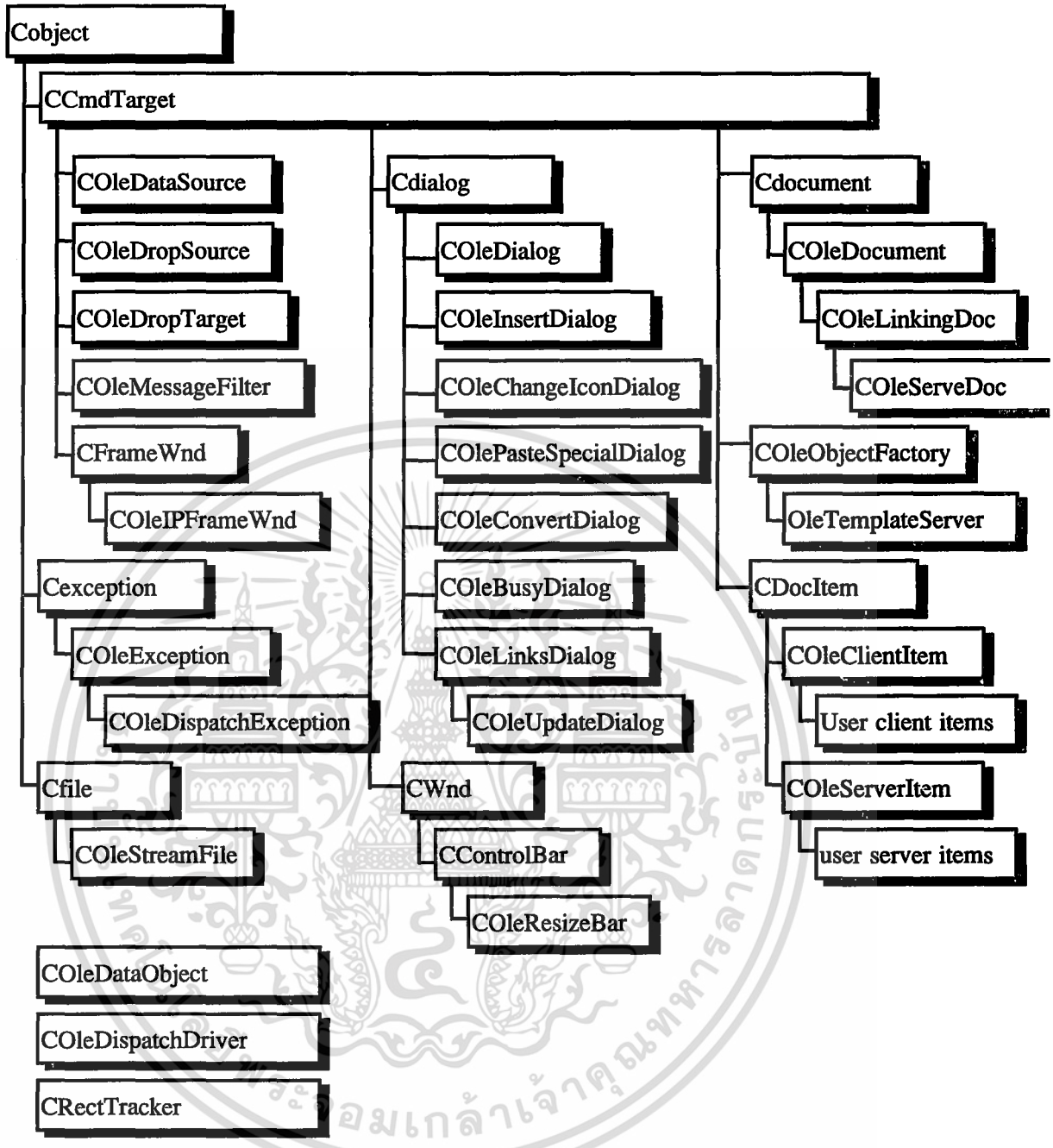
2.1.3 หลักการสำคัญ

ดังที่กล่าวมาข้างต้นว่า เมื่อมีการนิยามโปรแกรมขึ้นใหม่สำหรับการโปรแกรมเชิงวัตถุ ให้มีนิยามเป็นเซตของออบเจกต์นั้น เมื่อนิยามของการโปรแกรมเปลี่ยนไปโครงสร้างของการโปรแกรมเชิงวัตถุต้องเปลี่ยนไปด้วย จึงได้มีการสร้างกฎข้อบังคับต่าง ๆ ขึ้นมาใหม่สำหรับรองรับโครงสร้างของโปรแกรมเชิงวัตถุ โดยกฎต่าง ๆ เหล่านี้จะอยู่ในรูปของหลักการสำคัญต่าง ๆ ของการโปรแกรมเชิงวัตถุที่มีหลายประการด้วยกันดังจะกล่าวถึงคร่าว ๆ ดังต่อไปนี้

1. คลาสและสับคลาส

คลาสเป็นเซตของออบเจกต์ที่มีคุณสมบัติและพฤติกรรมร่วมกัน และอธิบายได้ว่า ออบเจกต์เหล่านี้มีโครงสร้างภายในอย่างไร เช่น คลาสของบุคคล คลาสของพนักงาน ออบเจกต์ทุกออบเจกต์ จะต้องอยู่ภายใต้คลาสอย่างน้อยหนึ่งคลาส ในแต่ละคลาสอาจประกอบด้วยออบเจกต์มากกว่าหนึ่งออบเจกต์ ซึ่งออบเจกต์เหล่านี้อาจจะมีลักษณะที่เหมือนกันที่เป็นลักษณะโดยรวมของคลาสนี้ เราเรียกลักษณะโดยรวมของคลาสนี้ว่าค่าตัวแปรคลาส (Class variable) และสำหรับลักษณะเฉพาะตัวของออบเจกต์ เรียกว่าตัวแปรตัวอย่างของแต่ละออบเจกต์ในคลาส (instance variable)

เพื่อให้เห็นภาพความสัมพันธ์ระหว่างคลาสและสับคลาสให้ชัดเจนยิ่งขึ้น จึงขอยกตัวอย่างลำดับขั้นของ OLE 2 ซึ่งแสดงได้ดังรูป



รูปที่ 2.1.3 OLE 2 Classes

จากการกำหนดคลาสโดยอาศัยการสืบทอด ทำให้เกิดคลาสขึ้นมามากมายในระบบที่เรียกว่า คลาสไลบรารี (class libraly) ขึ้น หรือบางครั้งเรียกว่าคลาสลำดับชั้น (class hierarchy) ซึ่งหมายถึงชุดหรือโครงสร้างความสัมพันธ์ของคลาสต่าง ๆ ในระบบแบบเชิงวัตถุ ซึ่งเป็นเสมือนกับไลบรารีที่ใช้เก็บโปรแกรมที่จะนำมาลิงค์ต่อของโปรแกรมที่เขียนขึ้นแบบเก่า แต่แทนที่จะเป็นชุด

ฟังก์ชัน ก็เป็นโครงข่ายของคลาสในระบบแทน มีข้อดีคือสามารถลดเวลาในการพัฒนาโปรแกรมลงได้;เพราะสามารถทำการเปลี่ยนแปลงแก้ไขในคลาสไลบรารีได้

2.เอนแคปซูเลชัน

เอนแคปซูเลชัน คือ การเปลี่ยนแปลงแก้ไข ค่าตัวแปรภายในออบเจกต์จะต้องกระทำผ่านทาง วิธีการ (Method) ของออบเจกต์ดังกล่าวเท่านั้น เป็นลักษณะของการป้องกันการแก้ไขค่าตัวแปรสมาชิกภายในออบเจกต์โดยตรง ตัวแปรใด ๆ ภายในออบเจกต์จะมีสถานะค่าปริยาย (default) เป็นข้อมูลท้องถิ่น (Private) ข้อมูลแบบนี้จะไม่สามารถจ้าถึงได้โดยตรง การจะของดูหรือเปลี่ยนแปลงค่า ต้องกระทำผ่านโพรซิเยอร์ที่เป็นวิธีการใน ออบเจกต์นั้นเท่านั้น แต่ก็มีข้อเสียคือ ทำให้ความเร็วในการทำงานของโปรแกรมลดลง

การกำหนดเอนแคปซูเลชันซึ่งเป็นคุณสมบัติของออบเจกต์ ควรมีลักษณะดังนี้

1. กำหนดขอบเขตที่ชัดเจนให้กับออบเจกต์
2. กำหนดส่วนเชื่อมโยง (interface) หมายความว่า ออบเจกต์นั้นจะติดต่อกับ ออบเจกต์อื่นอย่างไร
3. ส่วนการนำไปใช้งาน ไม่สามารถเข้าถึงได้นอกเหนือขอบเขตของคลาส ที่ผลิตออบเจกต์นั้น

3.โพลิมอร์ฟิซึม

โพลิมอร์ฟิซึม หมายถึง คุณสมบัติที่ระบบยอมให้การส่ง message แบบเดียวกันสามารถตอบสนองได้หลายแบบ และไม่จำเป็นต้องได้รับการตอบสนองเหมือนกัน ขึ้นอยู่กับออบเจกต์ที่รับ message เป็นสำคัญ โดยออบเจกต์จะตอบสนองได้โดยใช้วิธีการที่มีอยู่ในตัวมันเอง ตัวอยางเช่น เมื่อออบเจกต์ A ส่ง message Show ไปให้กับออบเจกต์ B ออบเจกต์ B จะตอบสนองโดยใช้วิธีการ Show () ที่มีในตัวเอง ถ้าออบเจกต์ B เป็น ออบเจกต์ในคลาส Rectangle ก็จะวาดรูปสี่เหลี่ยม ถ้าเป็นออบเจกต์ในคลาส Circle ก็จะวาดรูปวงกลม โดยที่ทั้งหมดนี้ใช้ message เดียวกันคือ Show..

การยินยอมให้มีการตั้งชื่อโพรซีเยอร์ หรือวิธีการซ้ำกันได้ในโปรแกรมนั้น เป็นคุณสมบัติขั้นมูลฐานของภาษาแบบเชิงวัตถุโดยทั่ว ๆ ไป แต่ในบางภาษาไม่เรียกส่วนนี้ว่าเป็นโพลิมอร์ฟิซึมแต่จะรวมเข้ากับคุณสมบัติการสืบทอด เพราะว่าในคลาสลูกที่รับเอาวิธีการมาจากคลาสพ่อสามารถทำการแก้ไขวิธีการดังกล่าวได้ทุกกรณี

ในระบบแบบเชิงวัตถุ มีองค์ประกอบที่สำคัญที่สุด คือ คลาสไลบรารี (class libraly) ซึ่งประกอบไปด้วยคลาสต่าง ๆ มากมาย ที่มีโอกาสจะกำหนดชื่อวิธีการของแต่ละคลาสซ้ำกันได้มาก สามารถแก้ไขได้ ด้วยการนำคุณสมบัติโพลิมอร์ฟิซึมเข้ามาช่วย ตัวอย่างเช่น การ Show ของแต่ละคลาสจะมีวิธีการตอบสนองที่ต่างกัน ผู้พัฒนาระบบไม่จำเป็นต้องตรวจดูชื่อของวิธีการในทุก ๆ

คลาสเพื่อป้องกันการตั้งชื่อซ้ำก่อนประกาศชื่อวิธีการ ในด้านการใช้งานระบบ ผู้ใช้ก็ไม่จำเป็นต้องจดจำวิธีการส่ง message หลายวิธี อย่างเช่นวินโดว์ใช้การคลิกเมาส์เพื่อออกคำสั่ง วินโดว์ต่าง ๆ ซึ่งถือเป็นออบเจกต์แต่ละตัวในระบบนั้น ใช้การ คลิกเมาส์สองครั้งซ้อนเหมือนกันทั้งหมด ไม่ว่าผู้ใช้ต้องการจะเปิดวินโดว์ File manager หรือเปิดวินโดว์เกมส์ก็ใช้วิธีการเดียวกัน แต่การตอบสนองจากระบบขึ้นอยู่กับว่ากำลังต้องการเปิดวินโดว์อะไรอยู่

จุดประสงค์ของโพลิมอร์ฟิซึมมีความต้องการเน้นความคิดที่ว่า

" ภาษา object-oriented ที่ดีนั้น ควรที่จะสามารถตัดสินได้ว่าควรใช้มาตรฐาน ของออบเจกต์ตัวไหนในการตอบสนอง message ที่มีจุดประสงค์คล้าย ๆ กัน "

4.คุณสมบัติการสืบทอด

การสืบทอดคุณสมบัติของคลาส เป็นวิธีการสร้างคลาสใหม่ โดยอาศัยรูปร่างของคลาสเดิม เป็นคุณสมบัติที่ขาดไม่ได้ของภาษาเชิงวัตถุ เพราะโปรแกรมในนิยามของการโปรแกรมเชิงวัตถุ หมายถึงความสัมพันธ์ของออบเจกต์ ถ้าหากขาดคุณสมบัติการสืบทอดการสร้างชนิดข้อมูลหรือคลาสก็จะทำได้ยากมากขึ้น การที่โปรแกรมเมอร์สามารถกำหนดชื่อ วิธีการในคลาสลูก ให้มีชื่อเดียวกันกับวิธีการในคลาสพ่อได้ก็เพื่อลดความยุ่งยากในการกำหนดชื่อ และเพื่อแสดงกิริยาตอบสนองของออบเจกต์ได้ชัดเจนยิ่งขึ้น

ในภาษาแบบเชิงวัตถุเวอร์ชันใหม่นั้น จะมีการนำคุณสมบัติทางการสืบทอดแบบหลายพ่อลายแม่มาใช้ด้วย เรียกว่า คุณสมบัติการสืบทอดแบบหลายทาง (Multiple inheritance) หมายถึง การสืบทอดโดยรับเอาลักษณะของคลาสมากกว่าหนึ่งคลาสมาสร้างคลาสใหม่ ตัวอย่างเช่น โปรแกรมเวิร์ดโปรเซสเซอร์ในปัจจุบัน มักจะรวมความสามารถในการเก็บภาพกราฟฟิกไว้รวมกับเอกสารได้ หากสร้างโปรแกรมที่มีคลาสชื่อ WordProcessor ขึ้น ก็จะเป็นคลาสที่สืบทอดมาจากคลาสสองคลาสคือ TextEditor และ ImageProcessor เพราะมีลักษณะของคลาสทั้งสองรวมไว้ในตัวคือสามารถเขียนข้อความได้เหมือนกับ Editor ทั่วไป และยังสามารถแสดงและเก็บภาพกราฟฟิกเช่นเดียวกับโปรแกรมจัดการภาพกราฟฟิกได้ด้วย

ข้อดีที่ได้จากคุณสมบัติการสืบทอด

1. การช่วยลดเวลาในการพัฒนาระบบ
2. ลดค่าใช้จ่ายผู้พัฒนา
3. ได้ระบบที่มีโครงสร้างเป็นระเบียบและปรับปรุงเปลี่ยนแปลงได้ง่าย

2.1.4 ข้อดีของภาษาเชิงวัตถุ

เป้าหมายหลักของการพัฒนาซอฟต์แวร์แบบเชิงวัตถุ คือ

1. ทำให้การพัฒนาซอฟต์แวร์ใช้เวลาสั้นลง และต้นทุนต่ำลง โดยการใช้คุณสมบัติคลาส ที่สามารถนำกลับมาใช้ได้อีก และสร้างสับคลาสขึ้นมา เพื่อนำมาใช้แก้ปัญหา

2. ทำให้ต้นทุนในการบำรุงรักษาซอฟต์แวร์ต่ำลง เพราะสามารถหาจุดที่ต้องการเปลี่ยนแปลงในซอฟต์แวร์ได้ และการเปลี่ยนแปลงไม่ทำให้เกิดผลกระทบไปยังภายนอก คลาสได้

2.2 Object Linking and Embedding (OLE)

2.2.1 วิวัฒนาการของ OLE 2.0

ในตอนแรกวินโดว์ สามารถแลกเปลี่ยนข้อมูลระหว่างแอพพลิเคชันได้เพียงวิธีเดียว ก็คือ การใช้ Clipboard คุณต้องเข้าไปในโปรแกรมหนึ่ง จากนั้นสั่งให้ Clipboard เก็บหน้าจอ เสร็จแล้วเข้าสู่โปรแกรมหนึ่งเพื่อนำข้อมูลหน้าจอที่ Clipboard เก็บไว้มา paste หรือแปะลงในโปรแกรมนั้น

DDE เข้ามาพัฒนาขั้นตอนเหล่านั้นให้ง่ายขึ้น โดย DDE ช่วยให้แอพพลิเคชันเชื่อมต่อกันได้โดยตรงไม่ต้องผ่านคนกลางหรือ " ที่พักข้อมูล " อย่าง Clipboard (ในความเป็นจริงยังคงผ่าน วินโดว์ อยู่แต่เรามองไม่ออก) อย่างไรก็ดี DDE เป็นการสถาปนาการติดต่อระหว่างโปรแกรม 2 ตัวเท่านั้น และที่สำคัญถ้าโปรแกรมส่วนที่เป็น client ไม่มีความสามารถครบถ้วนเหมือน server ก็จะไม่สามารถแสดงหรือใช้ข้อมูลทุกอย่างได้ เช่น โปรแกรมเวิร์ด ก็จะไม่สามารถรับข้อมูลที่เป็นกราฟฟิคจากโปรแกรมฟรีเซนเตชันได้

OLE จึงเป็นส่วนขยายความสามารถของ DDE ให้ซับซ้อนไปอีกโดย OLE ช่วยให้แอพพลิเคชันที่เป็น client สั่ง server บริหารและจัดการกับออบเจกต์ที่ลิงก์เข้ามาได้โดยตรงกลายเป็นเอกสารแบบเชิงซ้อน (compound documents) ไปในที่สุด ด้วยวิธีการดังกล่าวทำให้แอพพลิเคชันหนึ่ง ๆ สามารถจัดการข้อมูลได้ทุกรูปแบบถึงแม้ว่ามันจะไม่มีฟังก์ชันหรือความสามารถชนิดนั้นก็ตาม เช่นไม่มีความสามารถในการอ่านไฟล์กราฟฟิค ก็จะมีความสามารถดังกล่าวได้ โดยการไปหยิบยืมความสามารถดังกล่าวจากโปรแกรมอื่นเข้ามาอยู่เนี่ยน
แทน เป็นต้น อนึ่ง OLE ในตอนแรกก็ยังคงเป็นเวอร์ชัน 1.0 อยู่ ต่อมาเมื่อกลางปี 1993 OLE 2.0 จึงได้กลายเป็นมาตรฐานใหม่ซึ่งคาดว่าโอเอสรุ่นใหม่ ๆ ของไมโครซอฟท์จะใช้เป็นแนวทาง **linking หรือ Embeding อย่างใดอย่างหนึ่งเท่านั้น**

แม้ว่า OLE จะย่อมาจากคำว่า Object linking and Embedding แต่ในความเป็นจริงแล้ว ออบเจกต์จะสามารถ ลิงก์ หรือ embed ได้อย่างเดียวในขณะหนึ่ง

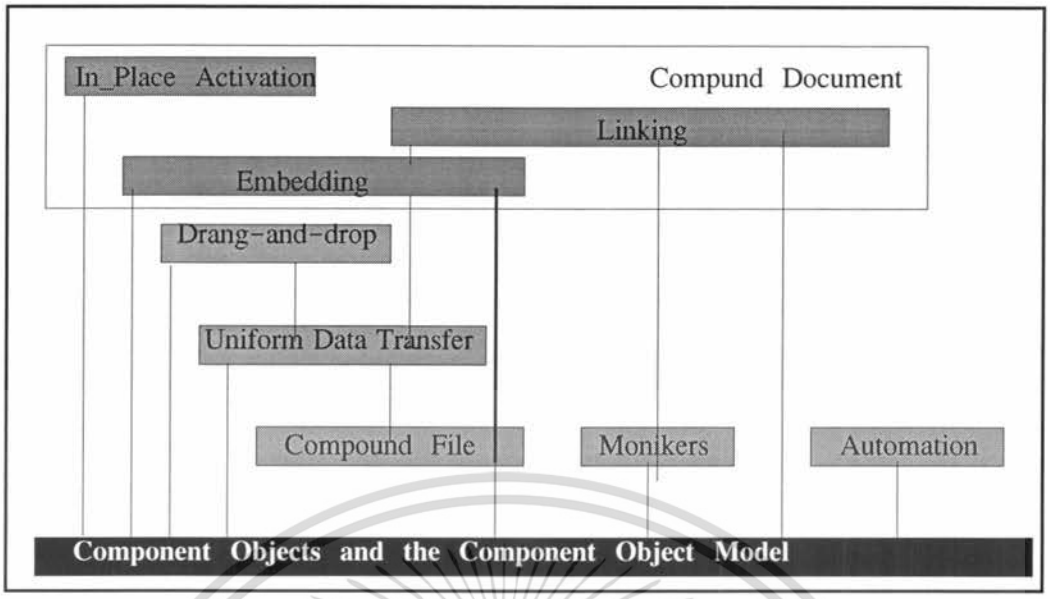
ถ้าเป็นการลิงก์ ทางกายภาคข้อมูลที่ถูกนำมาเชื่อมโยง (linked data) จะถูกเก็บและรักษาด้วยแอพพลิเคชันที่เป็น server ขณะที่แอพพลิเคชันชนิด client จะเป็นคนสถาปนาการลิงก์ข้อมูล แต่ถ้าออบเจกต์หรือข้อมูลนั้นถูก embed ทางกายภาคข้อมูลจะไม่ได้ปรากฏเป็น "ส่วนหนึ่ง" ของแอพพลิเคชันชนิด client เพียงโปรแกรมเดียวแต่ข้อมูลดังกล่าวจะถูกเก็บจริงไว้ทั้งแอพพลิเคชันที่เป็น server หรือ client

ยกตัวอย่าง ในกรณีที่คุณสร้างเอกสารไฟล์หนึ่ง ๆ ที่เกิดจาก MS-WORD และแอพพลิเคชันอื่น ๆ ของ วินโดว์ เอกสารแบบนี้เรียกว่า เอกสารเชิงซ้อน (compound documents) ซึ่งไม่มีข้อมูลจริงอยู่ แต่จะเป็นข้อมูลการลิงก์โยงออบเจกต์ของ OLE ส่วนข้อมูลจริงที่คุณไปหยิบมาจากแอพพลิเคชันอื่น ๆ จะถูกเก็บโดยแอพพลิเคชันเหล่านั้นเอง โดยไฟล์แยกย่อยดังกล่าวจะถูกมองเป็น OLE object แทน

2.2.2 Component Object Model

OLE 2.0 มีฟีเจอร์หลายอย่างที่คล้ายกับระบบปฏิบัติการ เช่น การจองพื้นที่หน่วยความจำ การจัดการไฟล์ การโอนถ่ายข้อมูล แต่ข้อแตกต่างอย่างสิ้นเชิงเมื่อเปรียบเทียบกับระบบปฏิบัติการ ก็คือ การให้บริการต่าง ๆ ดังเช่นที่กล่าวมา จะอยู่ในรูปของซิสเต็มออบเจกต์ในความเป็นจริงแล้วฟีเจอร์ต่าง ๆ ใน OLE 2.0 จะมีราวหนึ่งในสามของวินโดว์

ฟีเจอร์ขั้นสูงหลาย ๆ ฟิวเจอร์ใน OLE 2.0 จะอยู่บนเลเยอร์ขั้นต่ำอีกที (ดูรูปที่2.2.1) เลเยอร์ที่อยู่ล่างสุดจะเป็นเลเยอร์ที่เรียกว่า component object model ซึ่งอิมพลีเมนต์ใน COMPOBJ.DLL ใน OLE จะมีการกำหนดมาตรฐานบางอย่างขึ้นมาออบเจกต์ใดที่ยึดแนวของมาตรฐานที่ว่านี้ก็จะถูกเรียกว่าวินโดว์ออบเจกต์ มาตรฐานดังกล่าวที่จะทำให้แอพพลิเคชัน 2 แอพพลิเคชันสามารถติดต่อกันได้ทางออบเจกต์ โดยแต่ละแอพพลิเคชันไม่จำเป็นต้องรู้ว่าอีกแอพพลิเคชันมีโครงสร้างอย่างไร ตัวอย่างเช่น คุณอาจจะเลือกที่จะอิมพลีเมนต์วินโดว์ออบเจกต์โดยใช้ภาษา C++ เนื่องจาก C++ ใช้งานได้สะดวก ส่วนผู้ใช้ที่ใช้ออบเจกต์นี้อาจจะใช้ Visual Basic หรือแอพพลิเคชั่นอื่น ๆ ที่เขียนด้วยภาษาซี เป็นต้น



รูปที่ 2.2.1 แสดงเลเยอร์ที่สำคัญใน OLE 2.0

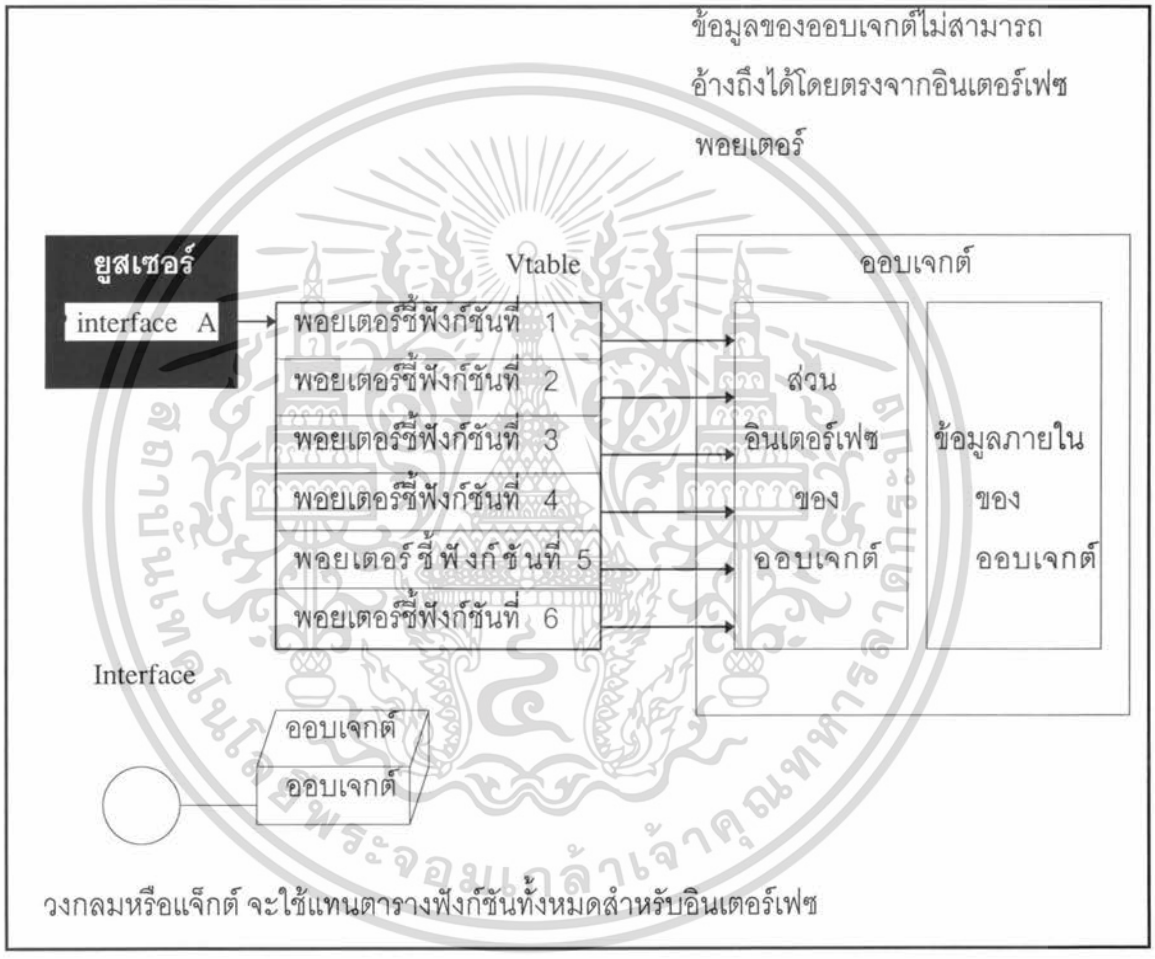
Component Object Model จะมี API ที่จะให้คุณเรียกใช้เพื่อสร้างวินโดว์ออบเจกต์ของคุณ และยังมีพอยเตอร์ที่ชี้ไปยังออบเจกต์นั้น เพื่อให้เรียกใช้ฟังก์ชันในออบเจกต์ ซึ่งก็สามารถทำให้ออบเจกต์นี้ไม่ถูกจำกัดอยู่แค่เพียงอยู่ใน DLL แต่ยังสามารถอยู่ในไฟล์ใด ๆ ก็ได้ หรือแม้แต่อยู่คนละเครื่อง คนละแพลตฟอร์ม OLE 2.0 ยังไม่มีความสามารถด้านนี้ในตอนนี้ แต่ว่า OLE 2.0 ได้เตรียม API ที่จะทำฟีเจอร์ด้านนี้ไว้ครบ แล้ว นอกจากนี้ OLE 2.0 ยังมีฟีเจอร์อีกอย่างหนึ่งที่เรียกว่า มาร์แชลลิ่ง (mar shaling) มาร์แชลลิ่งเป็นการผ่านฟังก์ชันและพารามิเตอร์ข้ามโปรเซส เนื่องจากโค้ดของออบเจกต์จะอยู่คนละส่วนของผู้ใช้หรืออาจจะอยู่คนละแพลตฟอร์ม มาร์แชลลิ่งจะทำหน้าที่แปลงพารามิเตอร์จาก 16 บิต เป็น 32 บิต เมื่อออบเจกต์และผู้ใช้รันอยู่คนละแพลตฟอร์ม ตัวอย่างเช่น ออบเจกต์ที่รันอยู่ในโปรเซส 32 บิต เช่น วินโดว์NT เก็บข้อมูลในรูปของยูนิโค้ด ในขณะผู้ใช้รันอยู่ในสภาวะแวดล้อมแบบ 16 บิตเก็บข้อมูลในรูปแอสกี มาร์แชลลิ่งจะทำหน้าที่เป็นตัวประสานระหว่างแอสกีกับยูนิโค้ด

เทคนิคมาร์แชลลิ่งที่ว่านี้ไม่ใช่เทคนิคใหม่ใน OLE 2.0 แต่มีมาตั้งแต่ OLE 1.0 แล้ว ใน OLE 1.0 จะใช้ DDE หรือ Dynamic Data Exchange ในการย้ายฟังก์ชันและพารามิเตอร์ข้ามโปรเซส แต่มาร์แชลลิ่งใน OLE 1.0 จะมีปัญหาตรงที่ขึ้นกับ DDE ซึ่งเป็นกระบวนการแบบอะซิงโครนัส ซึ่งการเรียกฟังก์ชันในออบเจกต์ จะต้องมีการรอในเมสเสจลูป จนกว่าฟังก์ชันนั้นจะทำงานจบลงซึ่งอาจจะเกิดปัญหาเรื่อง time outs หรือ resource blocking ได้ มาร์-แชลลิ่งใน OLE 2.0 จะใช้วิธีที่เรียกว่า

Lightreight Remote Procedure Calls หรือ LRPC ซึ่งเป็นการเรียกฟังก์ชันซิงโครนัส เมื่อมีการเรียก API ในออบเจกต์ ถ้าออบเจกต์ยังโปรเซส API นั้นไม่เสร็จสมบูรณ์ก็จะไม่มีการรีเทิร์นกลับมา

2.2.3 ออบเจกต์และการอินเทอร์เฟซ

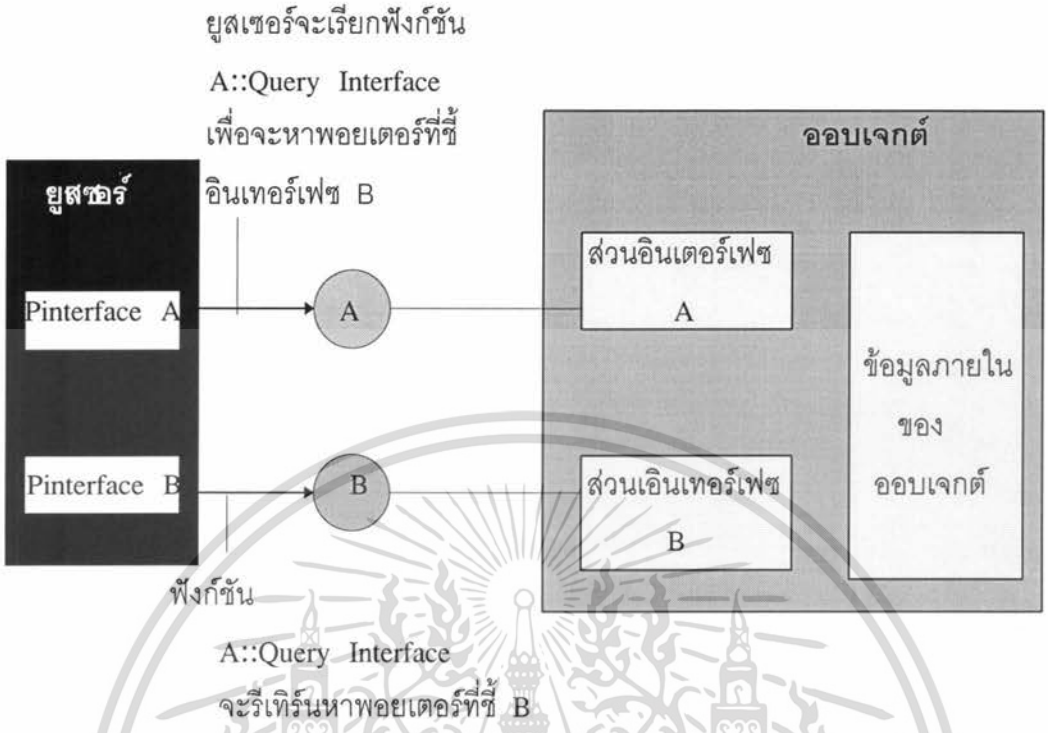
เมื่อมีออบเจกต์ก็ต้องมีการอินเทอร์เฟซระหว่างยูเซอร์กับออบเจกต์ การอินเทอร์เฟซกับออบเจกต์จะทำได้ง่ายโดยอินเทอร์เฟซผ่านทางอาร์เรย์ของพอยเตอร์ที่เรียกว่า vtable (ดูรูปที่ 2.2.2) ในเอกสารของ OLE 2.0 จะใช้วงกลมหรือเจ็กแทน vtable ดังรูปที่ 2.2.3



รูปที่ 2.2.2 แสดงส่วนอินเตอร์เฟซกับออบเจกต์ผ่านทาง vtable

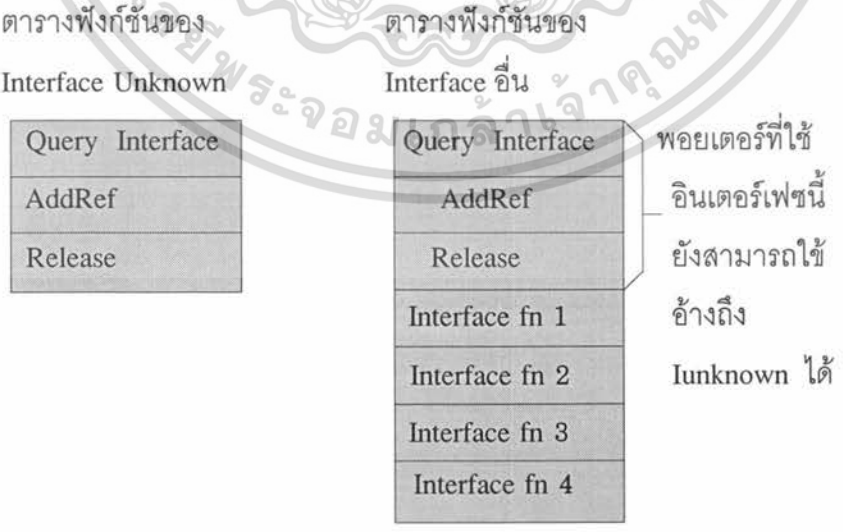
รูปที่ 2.2.3 แสดงอินเตอร์เฟซพอยเตอร์

ในวินโดว์ออบเจกต์หนึ่ง ๆ อาจจะมีอินเทอร์เฟซได้หลายอินเทอร์เฟซ ออบเจกต์ที่มีโครงสร้างไม่ซับซ้อนอาจจะมีอินเทอร์เดียว ส่วนออบเจกต์ที่มีโครงสร้างซับซ้อน เช่น คอมเพาวด์ด็อกคิวเมนต์อาจจะมีมากกว่า 3 อินเทอร์เฟซ ซึ่งจะมีมากแค่ไหนนั้นก็ขึ้นอยู่กับฟีเจอร์ของออบเจกต์นั้น การเข้าถึงออบเจกต์ด้วยพอยเตอร์ดังได้กล่าวมาแล้วนั้นจะเป็นเพียงแค่แต่ละอินเทอร์เฟซเท่านั้น ผู้ใช้จะไม่สามารถติดต่อหรือเข้าถึงข้อมูลทั้งออบเจกต์ได้ด้วยพอยเตอร์นี้ (ดูรูปที่ 2.2.3) OLE 2.0 ไม่ได้กำหนดมาตรฐานของการอินเทอร์เฟซเอาไว้ แต่ OLE 2.0 ได้กำหนดฟังก์ชันมาตรฐานสำหรับหาอินเทอร์เฟซพอยเตอร์ของอินเทอร์เฟซอื่นในออบเจกต์นั้นเอาไว้ ฟังก์ชันที่ว่านี้คือ QueryInterface ฟังก์ชันนี้เป็นส่วนหนึ่งของอินเทอร์เฟซที่เรียกว่า IUnknown (I คือ Interface) ซึ่งแสดงตัวอย่างในรูปที่ 4 IUnknown เป็นกลุ่มของฟังก์ชันที่วินโดว์ออบเจกต์ทุก ๆ ออบเจกต์จะต้องมี



รูปที่ 2.2.4 จาก introducing OLE 2.0

ส่วนอินเทอร์เฟซอื่น ๆ ก็จะดีไรว์มาจาก IUnknown อีกทอดหนึ่ง ดังนั้นทุก ๆ อินเทอร์เฟซจะต้องประกอบด้วย QueryInterface เสมอและมีข้อสังเกตอีกอย่างหนึ่งคือ เมื่อมีการสร้างอินเทอร์เฟซในวินโดว์ออบเจกต์ขึ้นมาก็จะมี IUnknown เกิดขึ้นด้วยเสมอโดยพอยเตอร์ที่ชี้ไปยังอินเทอร์เฟซนั้นก็จะใช้สำหรับการอ้างถึง IUnknow ได้ (ดูรูปที่ 2.2.5)



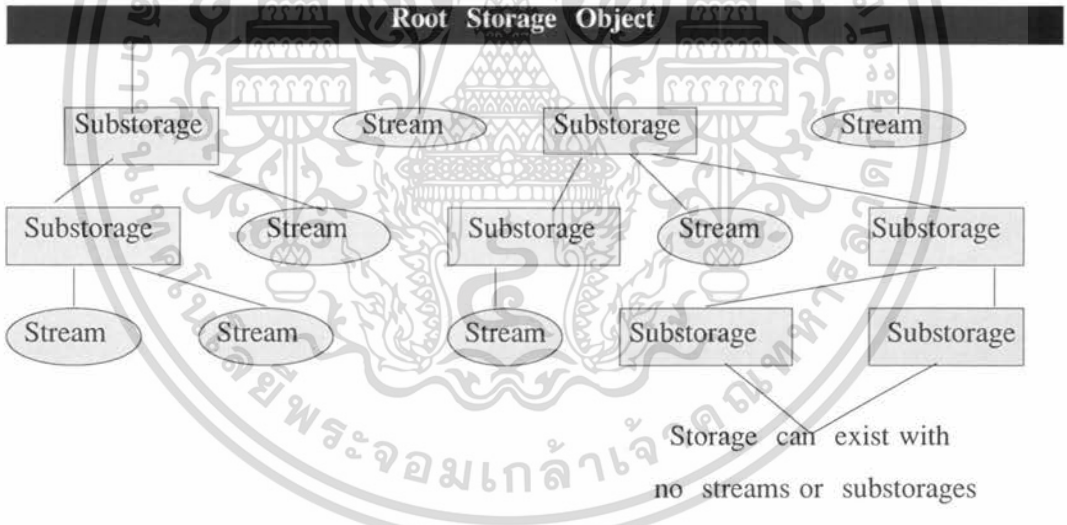
รูปที่ 2.2.5 แสดงอินเตอร์เฟซ IUnknown

QueryInterface ยังสามารถที่จะทำให้ผู้ใช้ตรวจสอบความสามารถของออบเจกต์ที่ยูสเซอร์ไม่รู้จักได้ในขณะที่กำลังรันอยู่โดยการหาอินเทอร์เฟซพอยเตอร์จาก QueryInterface ถ้าพอยเตอร์ที่รีเทิร์นกลับมาไม่ใช่ **NULL** ก็แสดงว่าออบเจกต์ดังกล่าวสนับสนุนอินเทอร์เฟซนั้น ๆ

2.2.4 โครงสร้างของข้อมูลและคอมเพาวด์ไฟล์

OLE 2.0 ได้มีการกำหนดจำนวนของการเก็บข้อมูลหรือโครงสร้างขึ้นมาเรียกว่า structured storage โดยที่ structured storage จะประกอบด้วยออบเจกต์ 2 ชนิด คือ สตอเรจออบเจกต์ และสตรีมออบเจกต์

สตรีมออบเจกต์ เปรียบเทียบได้กับระบบไฟล์แบบ FAT แต่ละสตรีมจะมีแอตทริบิวต์ที่เรียกว่า access right และมีพอยเตอร์ที่ใช้อ้างอิง ส่วนสตอเรจออบเจกต์เปรียบเทียบได้กับไดเร็กทอรี แต่ละสตอเรจสามารถประกอบด้วยหลาย ๆ สตอเรจและสตรีม ดังในรูปที่ 2.2.6 ในสตอเรจออบเจกต์จะไม่มีข้อมูลของผู้ใช้อยู่ ดังเช่นในสตรีมออบเจกต์ สตอเรจจะมีหน้าที่เพียงรู้ว่ามีสตรีมและสตอเรจใดบ้างในตัวมัน ดังนั้นคุณจึงสามารถที่จะลบสร้างสำเนาสตอเรจนี้ได้โดยไม่จำเป็นต้องรู้ถึงโครงสร้างในสตรีม ซึ่งตรงกับคอนเซปต์ของ "ออบเจกต์" นั่นเอง



รูปที่ 2.2.6 แสดงออบเจกต์ชนิด "สตรีม" และ " สตอเรจ "

OLE 2.0 ยังใช้ความสามารถของ structured storage นี้กับฟีเจอร์อย่างหนึ่งที่เรียกว่าคอมเพาวด์ไฟล์ (compound files) คุณสามารถที่จะใช้คอมเพาวด์ไฟล์นี้แทนระบบไฟล์แบบ แฮนเดิลของดอสได้เลยคอมเพาวด์ไฟล์จะเป็นส่วนที่อินเทอร์เฟซกับแอพพลิเคชันของคุณกับที่เก็บข้อมูลจริงใน

ไฟล์ ดังเช่นที่ดอสอินเทอร์เฟซระหว่าง เซกเตอร์ในดิสกับไฟล์ของคุณ โดยที่อ้างอิงถึงไฟล์แฮนเดิลเท่านั้น

กล่าวอีกนัยหนึ่งก็คือ strutured storage และ compound files ถูกออกแบบมาเพื่อกำหนดมาตรฐานของการเลย์เอาต์ข้อมูลในไฟล์ โดยมาตรฐานนี้จะทำให้แอพพลิเคชันใด ๆ ก็ได้สามารถใช้หรือเข้าถึงข้อมูลนี้ได้โดยไม่มี ความจำเป็นต้องรู้ถึงฟอร์แมตจริง ๆ ของข้อมูลนั้น

2.2.5 ยูนิฟอร์มดาต้าทรานสเฟอร์

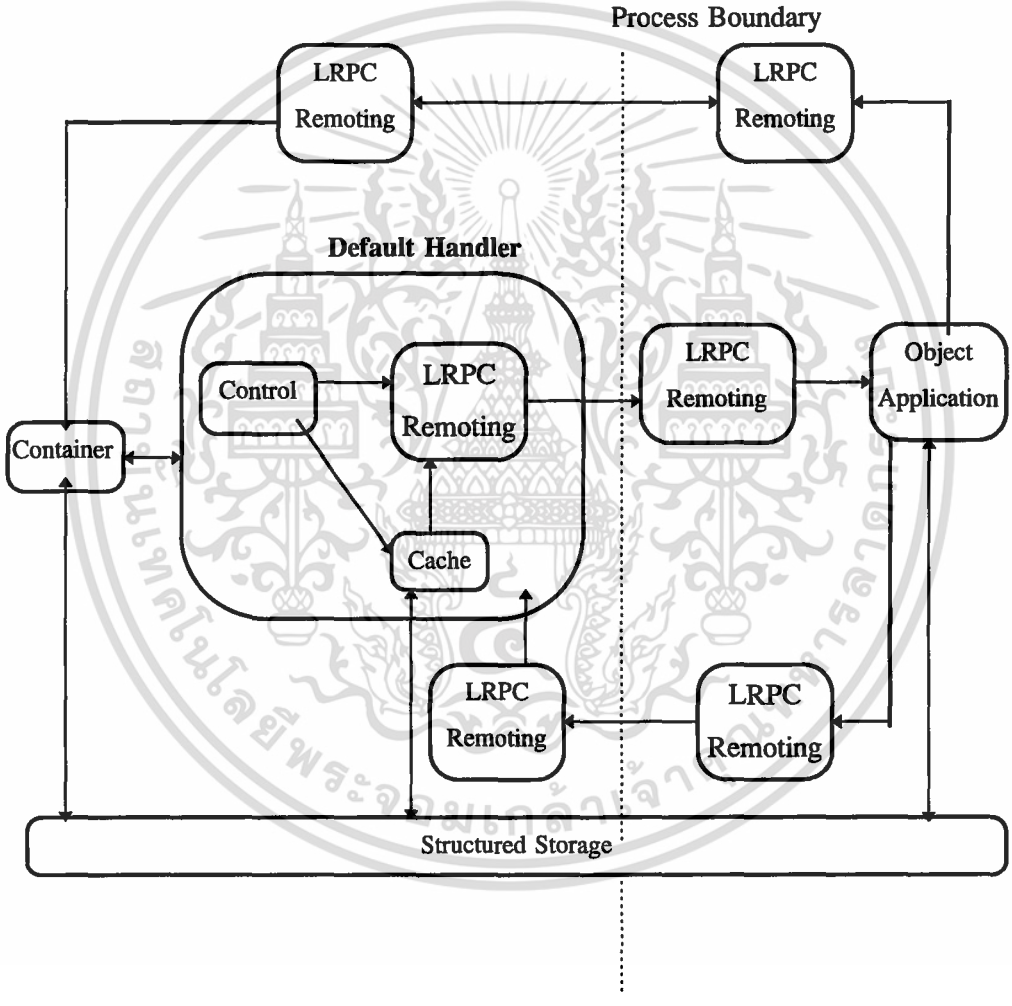
เลเยอร์ที่อยู่บน Component Object Model และ คอมเพาวด์ไฟล์ใน OLE 2.0 คือ ยูนิฟอร์มดาต้าทรานสเฟอร์หรือดาต้าออบเจกต์ซึ่งใช้ในการโอนถ่ายข้อมูลระหว่างแอพพลิเคชันโดยสามารถระบุสื่อในการโอนถ่าย ระบุอุปกรณ์ ที่ออบเจกต์นั้นต้องการ เช่น " มี DIB ในสตรีมออบเจกต์ ที่จะพิมพ์ผ่านโพสสคริปต์พรินเตอร์ ความละเอียด 300 dpi" นอกจากนี้การโอนถ่ายข้อมูลใน OLE 2.0 ยังสามารถโอนผ่านได้ หลายสื่อ เช่น คอมเพาวด์ไฟล์ ดิสก์ไฟล์ หน่วยความจำ ฯลฯ ในการโอนถ่ายข้อมูลแอพพลิเคชันจะใช้ API ใน OLE 2.0 ในการโอนถ่ายพอยเตอร์ที่ชี้ออบเจกต์จากแหล่งเก็บไปยังที่ที่ต้องการผ่านสื่อ

2.2.6 ดาต้าออบเจกต์และ Drag-and-Drop

การใช้ดาต้าออบเจกต์กับ Drag-and-Drop ไม่เพียงแต่จะใช้ ได้เฉพาะกับไฟล์อย่างในไฟล์แมเนเจอร์เท่านั้น แต่ OLE 2.0 จะทำให้ คุณสามารถที่ใช้ drag หรือ drop ออบเจกต์ใด ๆ ก็ได้ โดยซอร์ส ของออบเจกต์จะคอยตรวจสอบ และ ควบคุมขั้นตอนการลากออบเจกต์เมื่อออบเจกต์ถูก drop ลงในวินโดว์แอพพลิเคชันหรือวินโดว์ปลายทางจะตรวจสอบฟอร์แมตของออบเจกต์นั้น โดยใช้ฟีเจอร์ของ OLE 2.0 ลักษณะที่เป็นไปได้ทั้งหมดของการจัดการในการ ลากและวางจะแสดงได้ดังสัญลักษณ์ดังตารางต่อไปนี้

2.2.7 Object Embedding

object embedding เป็นการรวมเอาเทคโนโลยีของ OLE 2.0 อันได้แก่ คอมโพเนนต์ออบเจกต์ คอมเพาวด์ไฟล์ และดาต้าทรานสเฟอร์เข้าด้วยกัน คอมเพาวด์ด็อกคิวเมนต์จะเป็นส่วนที่สำคัญในการรวมออบเจกต์หลาย ๆ ออบเจกต์เข้าด้วยกัน คอมเพาวด์ด็อกคิวเมนต์สามารถที่จะอิมพลีเมนต์ได้ทั้งใน EXE และ DLL ออบเจกต์ที่ฝังสามารถที่จะเก็บข้อมูลใด ๆ ไว้ในสตอเรจออบเจกต์ และหลาย ๆ ออบเจกต์ที่รวมกันเป็นคอมเพาวด์ด็อกคิวเมนต์นั้นไม่มีความจำเป็นต้องรู้ถึงโครงสร้างของแต่ละออบเจกต์ เนื่องจากการอิจเทอร์เฟซผ่านทาง OLE แผนภาพแสดง Data flow สำหรับ embeded ออบเจกต์ แสดงได้ดังรูป



รูปที่ 2.2.7 แสดง Data Flow ของ Embeded ออบเจกต์

2.2.8 Object linking และ Monikers

ในการที่จะทำให้containner (แอพพลิเคชันที่รวมหลาย ๆ ออบเจกต์ไว้ในด็อกคิวเมนต์) และออบเจกต์แอพพลิเคชันการลิงก์กันได้คุณจำเป็นต้องเข้าไปยุ่งกับโครงสร้างข้อมูลอันหนึ่งของ OLE 2.0 ซึ่งทำหน้าที่บอกต้นทางของการลิงก์ และคุณต้องเพิ่มส่วนอินเทอร์เฟซให้กับออบเจกต์ การลิงก์ออบเจกต์นั้นจะมีการเปลี่ยนโค้ดในส่วนคอมเพาวด์ด็อกคิวเมนต์ที่คุณใช้จัดการการฝังออบเจกต์

การลิงก์ออบเจกต์ใน OLE 2.0 ส่งผลกับcontainnerมากกว่าใน OLE 1.0 containnerใน OLE 2.0 ไม่เพียงแต่เป็นการใช้การลิงก์ออบเจกต์เท่านั้นแต่ตัว containnerยังสามารถกลายเป็นต้นทางการลิงก์ได้ด้วยกลไกของ OLE 2.0 ทำให้containnerสามารถที่จะทำตัวเป็นต้นทางของการลิงก์ โดยเก็บข้อมูลของออบเจกต์ที่ฝังในด็อกคิวเมนต์ ดังนั้น ในcontainnerเดียวกัน คุณสามารถที่จะฝังหรือ embed ออบเจกต์โดยที่มีออบเจกต์อื่นลิงก์อยู่ได้

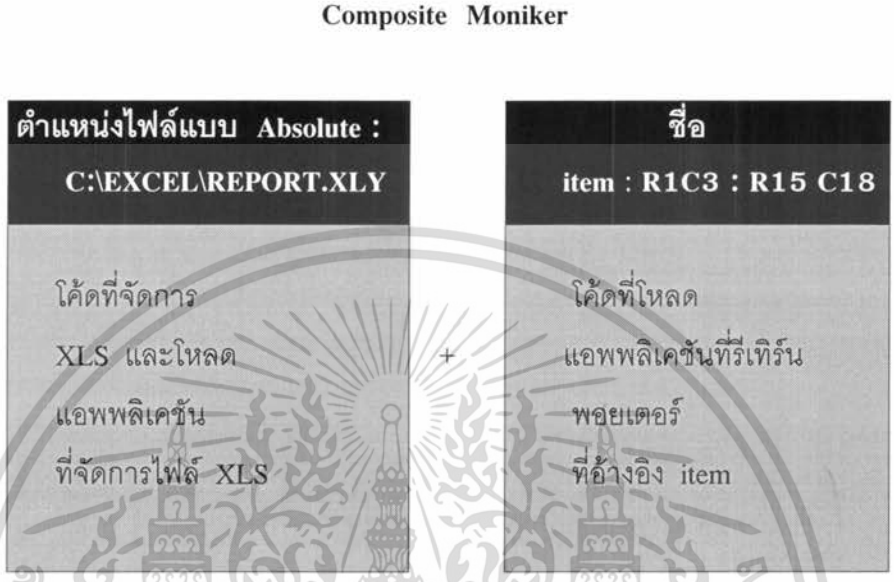
ในเริ่มแรกของการพัฒนา OLE การลิงก์ออบเจกต์เป็นส่วนที่เป็นปัญหามากที่สุด เนื่องจากข้อมูลที่อ้างอิงโดยออบเจกต์ที่ลิงก์นั้นถูกเก็บอยู่คนละไฟล์ การลิงก์อาจจะถูกทำลายได้ถ้ามีการเปลี่ยนแปลงตำแหน่งของไฟล์ดังกล่าว ใน OLE 1.0 จะเก็บตำแหน่งของลิงก์ไฟล์แบบ absulute ดังนั้นการเปลี่ยนแปลงตำแหน่งของไฟล์จำทำให้การลิงก์ถูกทำลาย แม้ว่าตำแหน่งของไฟล์แบบสัมพันธ์จะยังคงไม่เปลี่ยนแปลง ตัวอย่างเช่น c:\reports\may\report.doc และ c:\reports \april\status.xls เป็นตำแหน่งแบบ absolute ส่วน ...\april\ status.xls เป็นตำแหน่งแบบสัมพันธ์ และ นอกจากนี้ OLE 1.0 ยังไม่สามารถจัดการกับออบเจกต์ที่ซ้อนกันด้วยการอ้างถึงเพียงออบเจกต์เดียว ใน OLE 2.0 ปัญหานี้ได้ถูกแก้ไขโดยเพิ่มชนิดออบเจกต์ใหม่ที่เรียกว่า moniker ขึ้นมา

moniker ประกอบด้วย 2 ส่วน ส่วนแรกคือ ส่วนที่อ้างอิงถึงการลิงก์ออบเจกต์ ส่วนที่สองคือโค้ดที่ทำหน้าที่ลิงก์ออบเจกต์เข้าด้วยกันกระบวนการลิงก์นี้เรียกว่า binding ซึ่งเป็นการเรียกแอพพลิเคชันที่จัดการคลาสของการลิงก์ออบเจกต์นั้น ๆ สั่งแอพพลิเคชันนั้นโหลดไฟล์ที่เก็บออบเจกต์และสั่งให้แอพพลิเคชันนั้นอ้างอิงชื่อออบเจกต์ด้วยการใช้พอยเตอร์ moniker ใน OLE 2.0 มี 2 ชนิดคือ file moniker และ item moniker

file moniker ใช้เก็บลิงก์ไฟล์ทั้งแบบ absolute และ แบบสัมพันธ์ ถ้า moniker หาลิงก์แบบ absolute ไม่พบ ก็จะเปลี่ยนไปหาแบบสัมพันธ์แต่ในกรณีที่ลิงก์ไฟล์ถูกย้ายตำแหน่งไปอย่างสมบูรณ์ คือ ย้ายทั้ง absolute และ แบบสัมพันธ์ ในกรณีนี้ moniker จะไม่สามารถจัดการได้

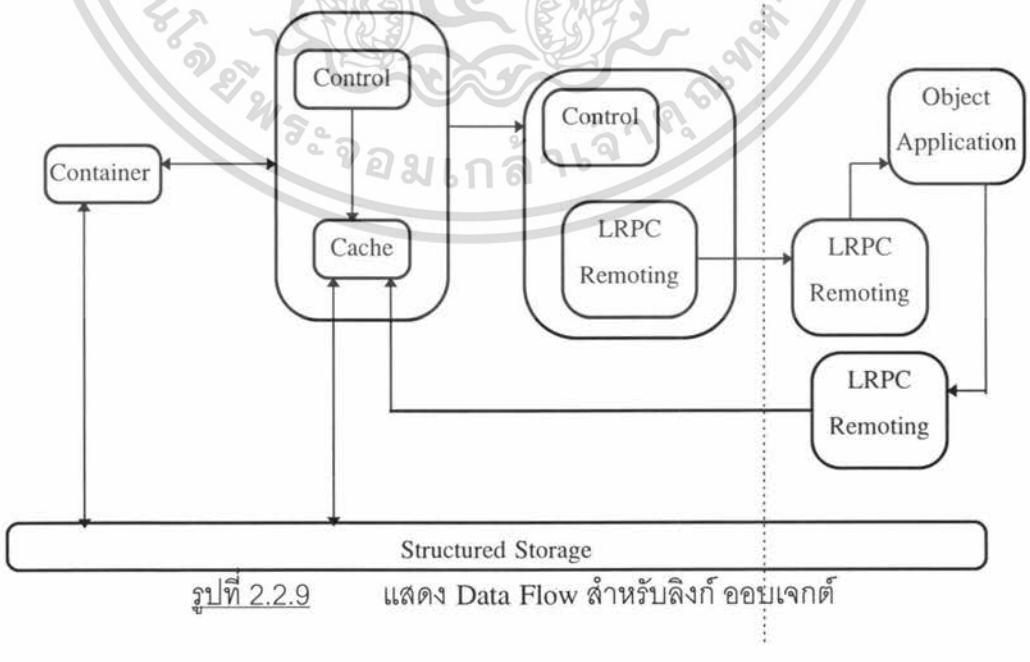
การอ้างอิงถึงออบเจกต์ที่ซับซ้อนลงไปอีก จาก moniker จะใช้ composit moniker โดย composit moniker อาจจะประกอบไปด้วย file moniker หรือ\และ item moniker หรือแม้แต่ composit moniker เองก็ตาม การลิงก์ส่วนใหญ่จะเป็น composit moniker ที่ประกอบด้วยหนึ่ง file moniker และหนึ่ง item moniker หรือลิงก์ไปยังออบเจกต์ที่ embed อยู่ในcontainner item moniker จะเก็บสตริงที่ใช้อ้างถึงชื่อของ item และชื่อของ item นี้จะมีความหมายเฉพาะแอพพลิเคชันที่สร้าง

ออบเจกต์ขึ้นมาเท่านั้น ซึ่งจะใช้เมื่อต้องการหาพอยเตอร์ที่ชี้ออบเจกต์โดยระบุชื่อ item ไป ตัวอย่างของ moniker ที่ประกอบด้วยหนึ่ง File moniker และหนึ่ง item moniker ก็ได้แก่ moniker ในไมโครซอฟท์เอกเซลซึ่งแสดงในรูปที่ 7 นอกจากนี้แล้ว OLE 2.0 ยังมี moniker ชนิดอื่น ๆ อีก แม้กระทั่งสามารถทำหนดขึ้นมาได้โดยผู้ใช้เอง



รูปที่ 2.2.8 แสดง composite moniker

แผนภาพแสดง Data flow สำหรับ ลิงก์ออบเจกต์ แสดงได้ดังรูป

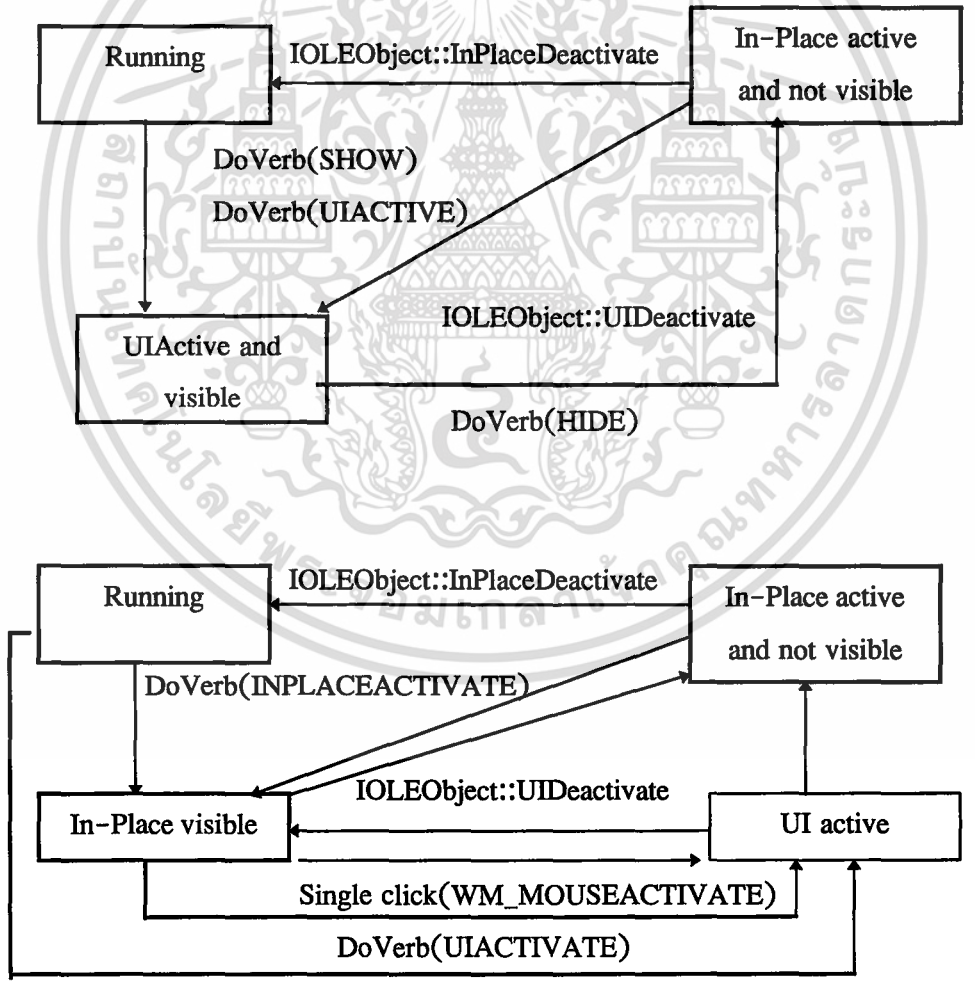


รูปที่ 2.2.9 แสดง Data Flow สำหรับลิงก์ ออบเจกต์

2.2.9 In-Place Activation

ใน OLE คอมเพาวด์ด็อกคิวเมนต์จะเป็นตัวบ่งบอกลักษณะของการ embed หรือลิงก์ออบเจกต์เข้าไปยังcontainner containnerก็จะคอยจัดการแต่ละออบเจกต์ เมื่อมีการแก้ไข ออบเจกต์ไม่ว่าจะเป็นออบเจกต์ที่ embed หรือลิงก์ ก็จะถูกเปิดขึ้นแก้ไขในอีกวินโดว์หนึ่งแยกต่างหาก ซึ่งเหมือนกับใน OLE 1.0 แต่ใน OLE 2.0 จะมีความสามารถอยู่อย่างหนึ่งคือ การ activate ออบเจกต์แทนที่ขึ้นมาในวินโดว์ของcontainnerเลยแทนที่จะมีการสร้างวินโดว์ใหม่ขึ้นมา ทำให้ผู้ใช้ไม่ต้องออกจากวินโดว์ที่กำลังทำงานอยู่ เพื่อไปแก้ไขออบเจกต์ในอีกวินโดว์หนึ่ง

ในการที่จะทำให้ออบเจกต์และcontainnerสนับสนุนฟีเจอร์ inplace activate ออบเจกต์และcontainnerจะต้องมีการเพิ่มส่วนอินเทอร์เฟซกับ OLE 2.0 ซึ่งส่วนที่เพิ่มมานี้จะครอบอยู่บนชั้นของคอมเพาวด์ด็อกคิวเมนต์อีกที และจะไม่ไปรบกวนหรือมีผลกระทบกับ อินเทอร์เฟซพื้นฐานอื่น ๆ ในคอมเพาวด์ด็อกคิวเมนต์ ภาวะการ In-Place เป็นไปได้ 2 ภาวะ คือ Outside-in และ Inside-out ซึ่งแผนภาพการทำงานของทั้งสองแบบแสดงตามลำดับดังนี้



รูปที่ 2.2.10 แสดงภาวะ outside-in และ inside-out การ In-Place

2.2.10 Automation

ฟีเจอร์อีกฟีเจอร์หนึ่งใน OLE 2.0 ที่แตกต่างจากฟีเจอร์อื่น ๆ อย่างสิ้นเชิง คือ automation ซึ่งฟีเจอร์นี้จะทำให้เราทราบถึงคำสั่งหรือฟังก์ชันที่ออบเจกต์จัดเตรียมไว้ให้เราใช้ประโยชน์ของ automation คือ การพัฒนาเครื่องมือเขียนโปรแกรมในลักษณะมาโคร โดยเมื่อคุณเลือกออบเจกต์ขึ้นมาออบเจกต์หนึ่ง คุณสามารถที่จะสั่งให้ออบเจกต์นั้น (ออบเจกต์นั้นต้องสนับสนุนฟีเจอร์นี้) แสดงฟังก์ชันตลอดจนพารามิเตอร์ที่ต้องการออกมาได้ ทำให้คุณสามารถเลือกฟังก์ชันที่จะเรียกใช้ได้

Client & Container

คำว่า Client และ Container เป็นคำที่แสดงถึงคุณสมบัติของแอพพลิเคชันที่ทำงานสนับสนุน OLE เช่น โปรแกรม Write ในวินโดว์ เป็น Container แอพพลิเคชัน

Server

Server เป็นแอพพลิเคชันที่สร้างส่วนของ OLE เพื่อใช้สำหรับ Container แอพพลิเคชัน

MiniServer

MiniServer ไม่สามารถที่จะรันได้ด้วยตัวของมันเอง มันสามารถรันได้จากการ Insert Object dialog จุดมุ่งหมายก็คือ เพื่อสร้างส่วนของ OLE สำหรับใช้กับ Container แอพพลิเคชัน ยกตัวอย่าง miniserver เช่น ไมโครซอฟท์ Equation 2.0 ไมโครซอฟท์ WordArt เป็นต้น

Full-Server

Full-Server สามารถจุะรันได้ด้วยตัวของมันเอง และสามารถรันได้ในลักษณะของ OLE-Server แอพพลิเคชันลักษณะนี้สามารถที่จะเก็บและโหลดไฟล์ของมันเพื่อที่จะแทรกลงไปใน OLE Client ตัวอย่าง Full-Server เช่น Paintbrush

Container-Server

Container-Server สามารถที่จะเป็นได้ทั้ง client และ full-server ในเวลาเดียวกัน แอพพลิเคชันประเภทนี้สามารถที่จะมี OLE item จาก server อื่น ๆ ได้ ยกตัวอย่างแอพพลิเคชันที่เป็น full-server เช่น ไมโครซอฟท์เวิร์ด และ เอกเซล เป็นต้น

Verbs

Verbs เป็นคำสั่งที่สนับสนุนโดย OLE item ใน client แอพพลิเคชัน OLE item สามารถเป็นเหตุให้ server ทำงานได้หลาย verb ต่าง ๆ กัน เช่น OLE item เสียง อาจจะสนับสนุน verbs Edit และ Play

2.8 OLE 2 Classes

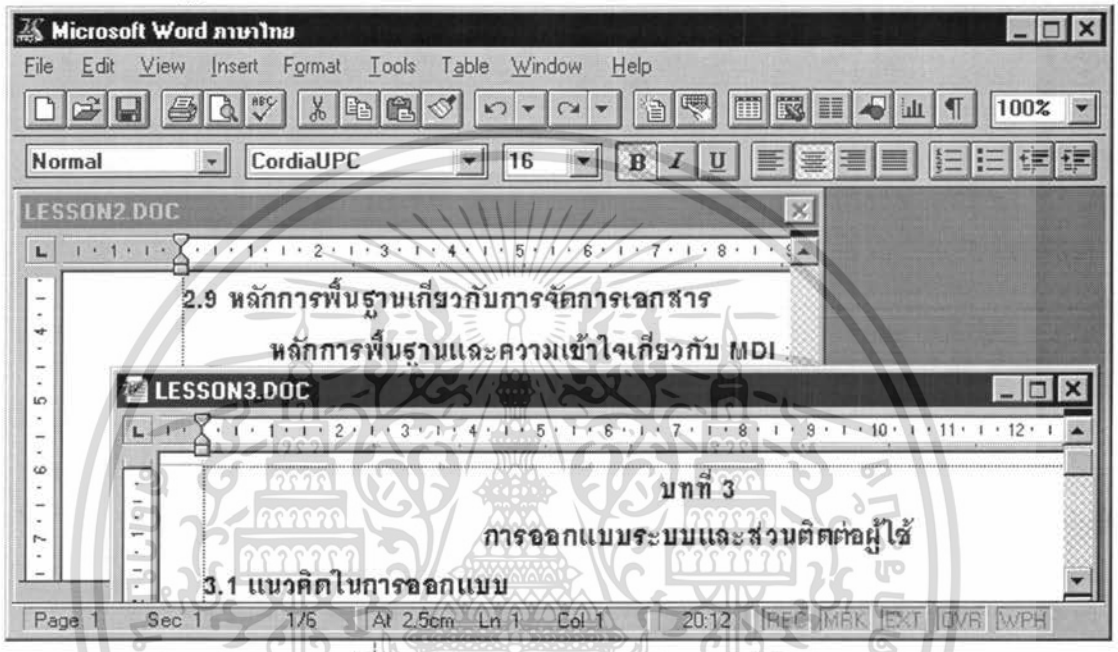
Class	Description
OLE Base Class	
COLEDialog	ใช้โดยเฟรมเวิร์กเพื่อบรรจุการสร้าง OLE ไดอะล็อกบ็อกทั่ว ๆ ไป ทุก ๆ คลาสของไดอะล็อกบ็อกทั้งหมดใน user-interface ถูกถ่ายทอดมาจากคลาสหลักนี้ ซึ่งไม่สามารถใช้ได้โดยตรง
CDocItem	เป็นคลาสหลักของ COLEClientItem และ OLEServerItem ออบเจกต์ของคลาสนี้ถ่ายทอดมาจาก CDocItem ซึ่งเป็นส่วนหนึ่งของด็อกคิวเมนต์
COLEDispatchDriver	ใช้เพื่อเรียก ออโตเมชันเซิฟเวอร์จาก ออโตเมชัน client ของคุณ ClassWizard ใช้คลาสนี้เพื่อสร้าง type-sefe คลาสสำหรับออโตเมชันเซิฟเวอร์ ซึ่งกำหนดประเภทของไลบรารี
COLEDocument	ใช้สำหรับการอิมพลีเมนต์เอกสารเชิงซ้อนซึ่งสนับสนุน Container พื้นฐาน ถ่ายทอดมาจาก CDocItem คลาสนี้สามารถใช้เป็นคลาสหลักสำหรับเอกสารของ Container และเป็นคลาสพื้นฐานสำหรับ COLEServerDoc
OLE Container Classes	
COLElinkgDoc	เป็นคลาสที่ถ่ายทอดมาจาก COLEDoccument ซึ่งกำหนดสำหรับพื้นฐานโครงสร้างของการลิงก์ container แอพพลิเคชันถ้าต้องการรองรับการลิงก์ไปยังออบเจกต์ที่ embed มา ควรรับการถ่ายทอดคลาสเอกสารของตัวเองจากคลาสนี้แทนที่จะเป็น COLEDocument
COLEClientItem	แสดงถึงขนาดของ OLE item ที่ embed หรือ ลิงก์ เข้ามา Client item ของคุณต้องถ่ายทอดมาจากคลาสนี้
OLE Data Transfer Classes	
COLEDropSource	ควบคุมการสั่ง การลากและวาง (drag-and-drop) จากเริ่มต้นจนจบคลาสนี้กำหนดเมื่อ drag operation เริ่มและเมื่อจบลงมันแสดงคอเซอร์ feedback ในขณะที่ปฏิบัติการ ลากและวางด้วย
COLEDropTarget	แสดงเป้าหมายของโอเปอร์เรชัน ลากและวาง ออบเจกต์

	COLEDropTarget สมนับกันกับวินโดว์ บนสกรีน มันกำหนดเพื่อตอบสนองข้อมูลที่วางบนมันและส่งเสริมปฏิบัติการการวางที่แท้จริง
COLEDataSource	ใช้เมื่อแอพพลิเคชันกำหนดข้อมูลสำหรับข้อมูลที่จะทรานสเฟอร์ COLEDataSource สามารถที่จะวิวเช่นเดียวกับ โอเรียนด์-ออบเจกต์ คลิปบอร์ด
COLEDataObject	ใช้เหมือนเป็นด้านหนึ่งของ COLEDataSource ออบเจกต์ COLEDataObject กำหนดการเข้าถึงขอ้มูลซึ่งออบเจกต์ COLEDataSource มีอยู่
OLE Server Classes	
COLEObjectFactory	ใช้สร้าง OLE item เมื่อ OLE Container ร้องขอ คลาสนี้บริการเหมือน เช่นคลาสหลักสำหรับระบุประเภทของ factories เพิ่มขึ้นซึ่งรวมไปถึง COLETemplateServer
COLETemplateServer	ใช้สร้างเอกสารโดยใช้สถาปัตยกรรม document/view ของเฟรมเวิร์ก ส่วนมากออบเจกต์ COLETemplateServer จะสั่งงานของมันไปยังออบเจกต์ CDocTemplate
COLEServerDoc	ใช้เหมือนคลาสหลักสำหรับคลาสเอกสารของเซิฟเวอร์แอพพลิเคชัน ออบเจกต์ COLEServerDoc กำหนดขนาดของเซิฟเวอร์ซึ่งสนับสนุนการทำงานด้วยกันกับ COLEServerItem ความสามารถของ Visual Editting จัดหามาโดยการใช้สถาปัตยกรรม document/view ของ MFC ไลบรารี
COLEServerItem	ใช้แสดงถึง OLE อินเทอร์เฟซกับออบเจกต์ COLEServerDoc ปกติมีหนึ่ง COLEServerItem ออบเจกต์ ซึ่งแสดงถึงส่วนที่ embed เข้ามาในเอกสาร และ มีหลาย COLEServerItem ออบเจกต์ที่แสดงถึงส่วนต่าง ๆ ในการลิงก์ของเอกสาร
COLEIPFramWnd	กำหนด เฟรมวินโดว์สำหรับวิว เมื่อเอกสารของเซิฟเวอร์กำลังถูก Edit in place
COLEResizeBar	กำหนดมาตรฐานของยูสเซอร์อินเทอร์เฟซสำหรับการเปลี่ยนขนาดภายใน ออบเจกต์ของคลาสนี้ถูกใช้ร่วมกับออบเจกต์ COLEIPFrameWnd เสมอ
OLE Dialog Box Classes	

COLEInsertDialog	แสดง Insert Object dialog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐาน สำหรับการเพิ่ม item ใหม่ ๆ ของ OLE
COLEConvertDialog	แสดง Convert dialog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับการแปลง OLE item จากชนิดหนึ่งไปเป็นอีกชนิดหนึ่ง
COLEChangeIconDialog	แสดง Change Icon dealog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับการเปลี่ยน item ของ OLE ให้เป็น icon ที่สัมพันธ์กัน
COLEPasteSpecialDialog	แสดง Paste Special dialog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับอิมพลีเมนต์คำสั่ง Edit Paste Special
COLElinksDialog	แสดง Edit links dialog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับการแก้ไขเปลี่ยนแปลงข่าวสารที่เกี่ยวกับ item ที่ลิงก์อยู่
COLEUpdateDialog	แสดง Update dialog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับการปรับปรุงทุก ๆ ลิงก์ในเอกสารให้ทันสมัย dialog box นี้บรรจุตัวทดสอบความคืบหน้าเพื่อทดสอบว่าจะจบขนวนการ update อย่างไรถึงจะสมบูรณ์
COLEBusyDialog	แสดง dialog box เมื่อ server ไม่ว่างและ server ไม่ตอบสนองซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับการเรียกแอพพลิเคชันที่ไม่ว่างปกติจะแสดงอัตโนมัติโดยการสนับสนุนของ COLEMessageFilter
OLE Miscellaneous Classes	
COLEException	การยกเว้นผลลัพธ์จากการผิดพลาดในการทำงานของ OLE คลาสนี้ใช้ทั้ง container และ server
CRectTracker	ใช้เพื่ออนุญาตให้เคลื่อนย้าย เปลี่ยนแปลงขนาด และ เปลี่ยนแปลงใหม่ของ in-place item
COLEStreamFile	ใช้ อินเทอร์เฟซของ OLE 2 Istream กำหนด Cfile เพื่อเข้าถึงไฟล์เชิงซ้อน คลาสนี้ (กำเนิดมาจาก Cfile) อนุญาตให้ MFC ใช้โครงสร้าง storage แบบต่อเนื่อง
COLEDispatchException	ยกเว้นผลลัพธ์จากข้อผิดพลาดในระหว่างการปฏิบัติของ OLE automation การยกเว้น OLE automation ถูกมอบโดย

	automation servers และรับโดย automation client
COLEMessageFilter	แสดง Edit links dialog box ซึ่งเป็นยูสเซอร์อินเทอร์เฟซมาตรฐานสำหรับการเปลี่ยนแปลงปรับปรุงข่าวสารเกี่ยวกับ items ที่ถูกลิงก์

2.9 หลักการพื้นฐานเกี่ยวกับการจัดการเอกสาร



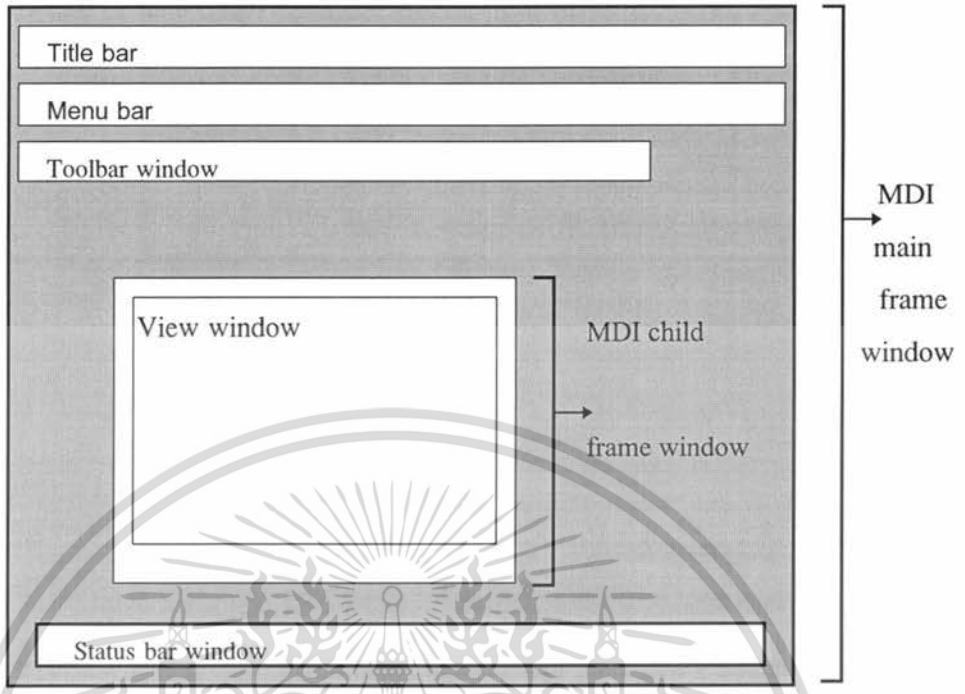
รูปที่ 2.2.11 แสดงเอกสารแบบ MDI

หลักการพื้นฐานและความเข้าใจเกี่ยวกับ MDI : จากรูปจะเห็นว่ามี Document 2 ไฟล์ ที่แยกจากกันเปิดอยู่ซึ่งแต่ละไฟล์อยู่ใน Child วินโดว์ที่แยกจากกันแต่มี Child วินโดว์ตัวเดียวที่ทำงานอยู่ ซึ่ง Child วินโดว์ที่ทำงานอยู่จะอยู่ข้างบนและเมนูของแอพพลิเคชันจะมีผลกับ Child วินโดว์ที่ทำงานอยู่เท่านั้น ซึ่ง Child จะมีปุ่มปิด (รูปกากบาท) ปุ่ม Minimize และปุ่ม Maximize (รูปสี่เหลี่ยม) ซึ่งอยู่ทางด้านมุมขวาบน และจากเมนู Window จะมีคำสั่งที่จัดการ Child วินโดว์ที่ทำอยู่โดยตรง เมื่อเราปิด Child วินโดว์ทั้งสอง ปุ่มทูลบาร์ส่วนมากจะใช้การไม่ได้ (Disable) พร้อมทั้งเมนูจะเปลี่ยนไปเป็นของวินโดว์หลัก

Document Template Class สำหรับ MDI การสร้าง Template สำหรับ MDI จะถูกเรียกในฟังก์ชัน InitInstance ซึ่งโค้ดการสร้างอยู่ในไฟล์ slide.cpp

เฟรมวินโดว์และ Child วินโดว์ใน SDI(Single Document Interface) มีคลาสเฟรมวินโดว์และออบเจกต์เฟรมวินโดว์เพียงแค่หนึ่งตัวโดย AppWizard จะสร้างคลาสชื่อ CMainFrame ซึ่งดีไรฟ์มา

จากคลาส CFrameWnd สำหรับ SDI แต่แอพพลิเคชัน MDI มีเบสคลาสเฟรมวินโดว์ 2 ตัวและออบเจกต์เฟรมได้หลายตัว ซึ่งความสัมพันธ์ระหว่างเฟรมและวิววินโดว์แสดงดังรูป



รูปที่ 2.2.12 MDI main frame window และ MDI child frame window

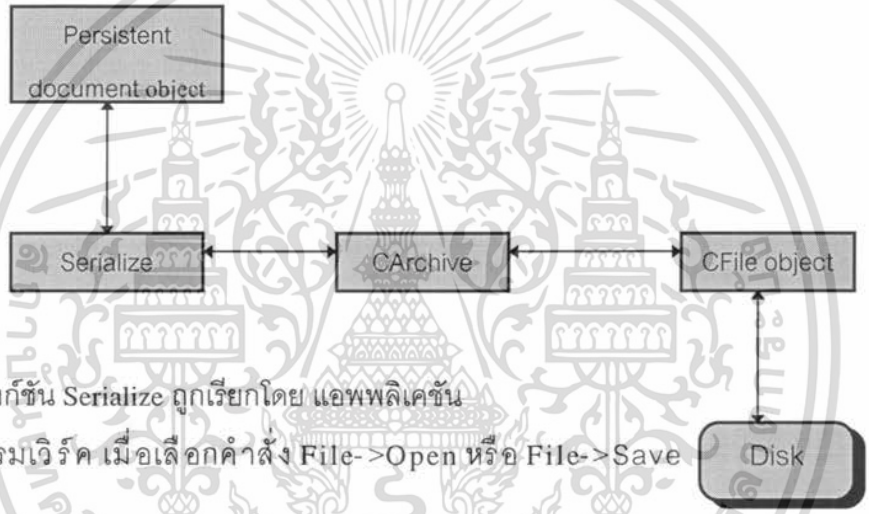
การโหลดและเก็บดอกคิวเมนต์ การโหลดและเก็บดอกคิวเมนต์ของ MDI ในที่นี้ใช้หลักการ Serialize ซึ่งเหมือนกันกับที่ใช้ใน SDI เพียงแต่ต่างกันสองข้อที่สำคัญคือ ออบเจกต์ดอกคิวเมนต์ใหม่ถูกสร้างขึ้นมาทุกครั้งที่ไฟล์ดอกคิวเมนต์ถูกโหลดจากดิสก์ และดอกคิวเมนต์ถูกทำลายเมื่อ Child วินโดว์ถูกปิด

ความเข้าใจพื้นฐานเกี่ยวกับการเก็บและโหลดข้อมูลจากดิสก์ : เราจำเป็นต้องรู้จักกับคำว่า Serialization ซึ่งคำว่า Serialization อาจจะเป็นคำใหม่ แต่ในการเขียนโปรแกรมแบบ object-oriented ซึ่งมีหลักคือ ออบเจกต์นั้นสามารถเก็บไว้ได้ตลอด ซึ่งหมายความว่า ออบเจกต์สามารถเก็บไว้บนดิสก์ เมื่อโปรแกรมเลิกการทำงานและเรียกขึ้นมาใหม่ได้เมื่อโปรแกรมทำงานใหม่ และขั้นตอนของการบันทึก และเรียกคืนออบเจกต์จากดิสก์เรียกว่า Serialization ซึ่งใน Microsoft Foundation Class Library คลาสที่ได้ออกแบบมามีเมมเบอร์ฟังก์ชันชื่อ Serialize และแอพพลิเคชัน เฟรมเวิร์ค จะติดต่อกับฟังก์ชัน Serialize สำหรับออบเจกต์ใด ๆ เช่นออบเจกต์ของคลาส CSlideCntrItem ซึ่งเป็นรูปภาพหรือข้อความจะถูกบันทึกหรืออ่านจากดิสก์

ดิสก์ไฟล์และ Archives

Serialize จะอ่านและเขียนข้อมูลเมื่อใด, Serialize ติดต่อกับดิสก์ไฟล์ได้อย่างไร ซึ่งสำหรับ MFC ดิสก์ไฟล์ถูกแสดงด้วยออบเจกต์ของคลาส CFile ซึ่งออบเจกต์ CFile ได้รวมไฟล์ Handle ซึ่งเราได้จากฟังก์ชัน runtime ในภาษาซี คือ _Open แต่ไม่ได้เป็นพอยเตอร์ FILE ที่มีบัฟเฟอร์ที่ได้จากการเรียก fOpen แต่เป็น handle ไปยัง BinaryFile ไฟล์ handle นี้ถูกใช้โดยแอพพลิเคชัน เฟรมเวิร์ค สำหรับเรียกฟังก์ชัน _read, _write และ _lseek

ถ้าแอพพลิเคชันไม่ได้ติดต่อกับดิสก์ I/O โดยตรง แต่ขึ้นอยู่กับขั้นตอน Serialization เราสามารถหลีกเลี่ยงการใช้ออบเจกต์ CFile โดยตรงได้ โดยที่ระหว่างฟังก์ชัน Serialize และออบเจกต์ CFile จะเป็นออบเจกต์ archive (คลาส Carchive) ดังรูป



รูปที่ 2.2.13 ขั้นตอนการ erialize

โดยที่ออบเจกต์ CArchive เตรียมบัฟเฟอร์สำหรับเก็บข้อมูล ให้กับออบเจกต์ CFile และจัดการกับแฟล็กภายในซึ่งบอก archive ว่าเมื่อไหร่จะเก็บข้อมูลหรือโหลดข้อมูลจากดิสก์ซึ่งจะมี archive ที่ทำงานตัวเดียวในเวลาหนึ่ง ๆ และแอพพลิเคชัน เฟรมเวิร์ค จะดูแลเกี่ยวกับโครงสร้างของออบเจกต์ CFile และ ออบเจกต์ Carchive, การเปิดดิสก์ไฟล์สำหรับออบเจกต์ CFile, และสร้างความสัมพันธ์ระหว่างออบเจกต์ archive กับไฟล์ เพราะฉะนั้นสิ่งที่เราต้องทำทั้งหมดในฟังก์ชัน Serialize คือ โหลดข้อมูลหรือบันทึกข้อมูลลงในออบเจกต์ archive โดยที่แอพพลิเคชัน เฟรมเวิร์ค จะเรียกฟังก์ชัน Serialize ของดอกคิวเมนต์ผ่านคำสั่งการเปิดไฟล์ (File->Open) และบันทึกไฟล์ (File->Save หรือ Save As)

การเชื่อมต่อ File->Open กับโค้ด Serialization

เมื่อแอพพลิเคชัน เฟรมเวิร์ค สร้างโครงร่างของโค้ดโปรแกรมมันจะทำการแมปเมนู Item File->Open กับเมมเบอร์ฟังก์ชัน CWinApp::OnFileOpen ซึ่งจะเรียกไปยังฟังก์ชัน CWinApp::OpenDocumentFile ซึ่งมีหลักการทำงานดังนี้

1. เรียกเวอร์ชวลเมมเบอร์ฟังก์ชัน OnOpenDocument สำหรับออบเจกต์ดอกคิวเมนต์ที่มีอยู่แล้ว ฟังก์ชันนี้รอให้ผู้ใช้เลือกไฟล์และเปิดไฟล์, สร้างออบเจกต์ CArchive เพื่อโหลดและเรียก DeleteContents (SDI)

2.เรียกฟังก์ชัน Serailize ของดอกคิวเมนต์ซึ่งโหลดข้อมูลจาก archive

3.เรียกฟังก์ชัน OnInitialUpdate ของวิว

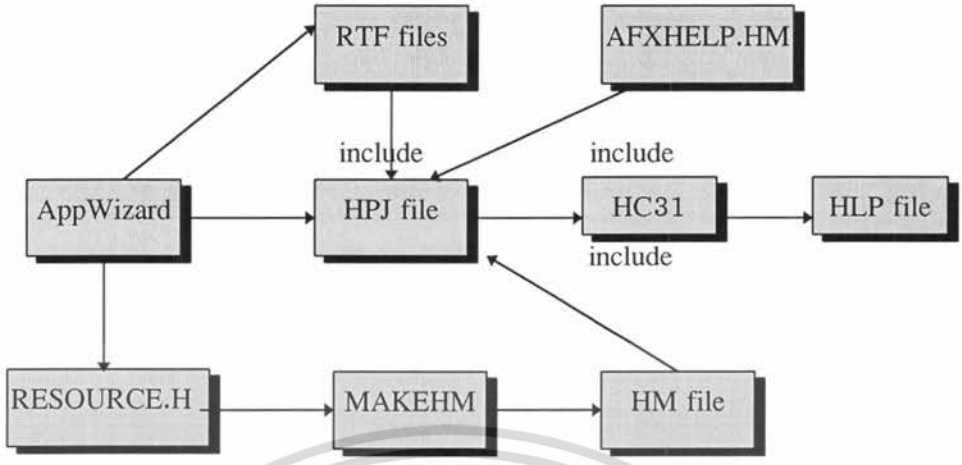
*ฟังก์ชันบางตัวไม่ได้ถูกเรียกโดยตรงจาก OpenDocumentFile แต่เรียกทางอ้อมโดยแอพพลิเคชัน เฟรมเวิร์ค

การเชื่อมต่อ File->Save และ File->Save As กับโค้ดการ Serialization

เมื่อ AppWizard สร้างโครงสร้างของโค้ดตอนเริ่มแรก มันจะแมปเมนู Item File->Save เข้ากับเมมเบอร์ฟังก์ชัน OnFileSave ของคลาส CDocument ฟังก์ชัน OnFileSave เรียกฟังก์ชัน CDocument::OnSaveDocument ซึ่งจะเรียกฟังก์ชัน Serialize ของดอกคิวเมนต์ของเรากับออบเจกต์ Archive และเซตเป็นการเก็บข้อมูล เมนู Item File->Save As ก็ถูกจัดการในลักษณะเดียวกันโดยที่มันจะถูกแมปกับฟังก์ชัน CDocument::OnFileSaveAs ซึ่งจะเรียก OnSaveDocument อีกทีหนึ่งและแอพพลิเคชัน เฟรมเวิร์ค จะจัดการไฟล์ทั้งหมดที่จำเป็นเพื่อบันทึกดอกคิวเมนต์ลงในดิสก์

2.3 Context-Sensitive Help

Context-Sensitive Help คือ ระบบความช่วยเหลือที่มีลักษณะไฮเปอร์เท็กซ์คือจะมีข้อความที่เป็นคีย์เวิร์ด (ข้อความที่มีสีเขียวและขีดเส้นใต้) เมื่อเราใช้เมาส์คลิกที่ข้อความที่เป็นคีย์เวิร์ดนี้ก็จะเป็นการเข้าถึงข้อความอื่นที่ลิงก์กับคีย์เวิร์ดนั้น ซึ่งระบบความช่วยเหลือแบบนี้เป็นที่นิยมใช้กันโดยทั่วไปเพราะมีประสิทธิภาพในการให้ความช่วยเหลือผู้ใช้ได้เป็นอย่างดี อีกทั้ง Visual C++ ก็มีออบชันที่ช่วยในการสร้างโดยตรงซึ่งทำให้การสร้างนั้นเป็นไปได้ง่ายขึ้นมาก ซึ่งขบวนการสร้างไฟล์ช่วยเหลือแสดงได้ดังรูปต่อไปนี้

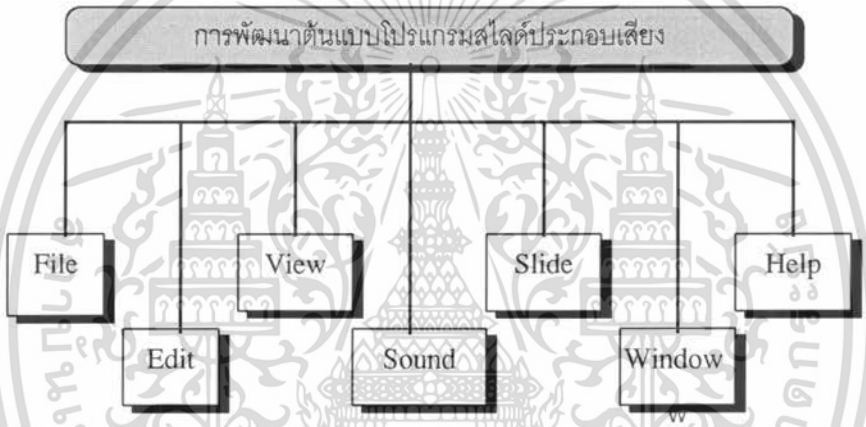


รูปที่ 2.3.14 แสดงขั้นตอนการสร้างไฟล์ช่วยเหลือ

บทที่ 3
การออกแบบระบบและส่วนติดต่อผู้ใช้

3.1 แนวคิดในการออกแบบ

เนื่องจากโปรแกรมที่จะทำการพัฒนานี้สามารถแบ่งออกได้ 3 ส่วนหลัก คือ การแสดงภาพแบบสไลด์, เสียง, และระบบความช่วยเหลือแบบ Context Sensitive Help ซึ่งจากการแบ่งโปรแกรมออกเป็น 3 ส่วนทำให้เราสามารถพัฒนาโปรแกรมแยกส่วนกันได้แล้วจึงนำมารวมกันภายหลัง ทำให้การพัฒนาโปรแกรมมีความเป็นอิสระ และทำงานได้สะดวกมากขึ้น ซึ่งเป็นผลดีข้อหนึ่งของการเขียนโปรแกรมแบบ OOP (Object Oriented Programming) ซึ่งออบเจกต์แต่ละออบเจกต์มีความเป็นอิสระต่อกัน จากการออกแบบทำให้เราได้ความสัมพันธ์ดังรูป



รูปที่3.1.1 แสดงโครงสร้างการพัฒนาโปรแกรมสไลด์ประกอบเสียง

3.2 โครงร่างของโปรแกรม

เมื่อได้แนวความคิดหลักแล้วจากนั้นก็จะเป็นการสร้างโครงร่างของโปรแกรมจาก AppWizard ซึ่งก่อนที่จะสร้างโค้ดส่วนนี้จำเป็นต้องมีข้อตกลงในการที่จะเลือกออปชั่นของการสร้างดังนี้

- เลือกออปชัน OLE เป็นแบบคอนเทนเนอร์ (Container)
- เปลี่ยนคลาสจาก CView เป็น CScrollView
- จากออปชันเลือก Multiple Document Interface, Initeal Tollbar, Printing and Print Preview, Context Sensitive Help

เมื่อได้ข้อสรุปเกี่ยวกับคุณสมบัติพื้นฐานของโปรแกรมที่ต้องมีแล้ว จากนั้นก็ใช้ AppWizard จากเมนู Project เพื่อสร้างโครงร่างของโค้ดตามออปชั่นที่ได้วางไว้ หลังจากที่ AppWizard ได้สร้างโค้ดโครงร่างแล้ว จะได้ไฟล์โปรแกรมหลักดังนี้

Cntritem.cpp, Cntritem.h

Mainfrm.cpp, Mainfrm.h

Slide.cpp, Slide.h

Slidedoc.cpp, Slidedoc.h

Slidevw.cpp, Slidevw.h

Stdafx.cpp, Stdafx.h

Slide.rc, Slide.rc2

ซึ่งโปรแกรมที่ได้สามารถคอมไพล์และรันได้ทันที และจะมีคุณสมบัติพื้นฐานตามที่ได้เลือกเอาไว้ แต่ยังขาดคุณสมบัติหลักคือการแสดงภาพแบบสไลด์, เสียง, Context-Sensitive Help จากที่ได้วิเคราะห์และแบ่งโปรแกรมออกเป็น 3 ส่วน ทำให้สามารถที่จะพัฒนาโปรแกรมแยกส่วนกันได้ (ตามคุณสมบัติของ ออบเจกต์) จากนั้นจึงค่อยมารวมทั้ง 3 ส่วนเข้าด้วยกันซึ่งทำให้การพัฒนาโปรแกรมเป็นไปได้เร็วขึ้น

3.3 การออกแบบการแสดงภาพแบบสไลด์

แนวความคิดในการจัดการออบเจกต์ที่นำเข้า (Embeded) จากเมนู Edit->Insert New Object คือ ออบเจกต์ที่นำเข้ามาต้องมีคุณสมบัติหลัก ๆ ดังนี้

1. สามารถย่อหรือขยายขนาดได้
2. สามารถเรียกใช้เมนูเพื่อจัดการกับออบเจกต์ได้เมื่อคลิกเมาส์ปุ่มขวา
3. สามารถเคลื่อนย้ายได้
4. มีขนาดเท่าของจริง
5. มีคุณสมบัติ In-Place Activate
6. มีคุณสมบัติ ลากและวาง (Drag and Drop)

หลังจากที่ใช้ AppWizard สร้างโครงร่างของโปรแกรมแล้ว เราสามารถที่จะ embed ออบเจกต์เข้ามาได้จากเมนู Edit->Insert New Object แต่ออบเจกต์ที่นำเข้ามาไม่มีคุณสมบัติดังที่ออกแบบไว้ ดังนั้นจึงจำเป็นต้องเพิ่มฟังก์ชันเพื่อจัดการกับคุณสมบัติดังกล่าวดังกล่าว และ ฟังก์ชันที่จัดการกับคุณสมบัติดังกล่าวโดยตรงมีดังต่อไปนี้

ฟังก์ชัน	ความหมาย
1.void CSlideView::SetupTracker(CRectTracker* pTracker, CSlideCntrItem* pItem, Crect* pTrueRect)	เมื่อออบเจกต์ถูกเลือกโดยการคลิกเมาส์ซ้าย หนึ่งครั้งกรอบของออบเจกต์นั้นจะเปลี่ยนไปให้ สามารถเปลี่ยนขนาดได้โดยทำงานสัมพันธ์กับ ฟังก์ชัน BOOL OnSetCursor(CWnd* pWnd, UINT nHitTest, UINT message)
2.void CSlideView::OnRButtonDown(UINT nFlags, Cpoint point)	เมื่อผู้ใช้คลิกเมาส์ขวาหนึ่งครั้งที่ออบเจกต์จะ เรียก popup เมนูขึ้นมาซึ่งจะมีคำสั่งที่ใช้บ่อย ๆ กับออบเจกต์ปรากฏอยู่และคำสั่งนั้น ๆ จะเรียก ฟังก์ชันที่สัมพันธ์กันอีกครั้งหนึ่ง
4. BOOL CSlideCntrItem::UpdateFromServerExent()	ฟังก์ชันนี้จะจัดการเกี่ยวกับขนาดให้เท่ากับของ จริงที่ได้นำเข้ามา
5. void CSlideView::OnLButtonDblClk(UINT mFlags, Cpoint point)	เมื่อผู้ใช้ดับเบิ้ลคลิกเมาส์ซ้ายจะเป็นการเรียก ฟังก์ชัน DoVerb เพื่อ open ออบเจกต์ตาม คุณสมบัติ In-place Activate
6.DROPEFFECT CSlideView::OnDragEnter(COleDataObject * pDataObject, DWORD grfKeyState, Cpoint point)	ทั้ง 3 ฟังก์ชันจะจัดการเกี่ยวกับการลากและ วาง (Dara and Drop) ออบเจกต์โดยสามารถ ทำได้ 4 กรณี คือ
DROPEFFECT CSlideView::OnDragOver(COleDataObject* pDataObject, DWORD grfKeyState, Cpoint point)	1. กดเมาส์ซ้ายค้างแล้วลากออบเจกต์ไปยัง ตำแหน่งที่ต้องการ (Defalt=Move)
BOOL CSlideView::OnDrop(COleDataObject* pDataObject, DROPEFFECT dropEffect, Cpoint point)	2. กดเมาส์ซ้าย + Crtl ค้างไว้ (Force Copy)
void CSlideView::OnDragLeave()	3. กดเมาส์ซ้าย + Alt ค้างไว้ (Force Move)
	4. กดเมาส์ซ้าย + Crtl+ Shift (Link)

การแสดงภาพแบบสไลด์นั้น ปัญหาหลักมีอยู่ว่า จะเก็บภาพหลาย ๆ ภาพไว้ใน ไฟล์เอกสารเดียวกันได้อย่างไร ใช้หลักการเก็บแบบไหน และ จะแสดงผลอย่างไร ? จากหลักการซีเรียลไลท์ที่ได้กล่าวในบทที่ 2 เราสามารถนำมาประยุกต์เพื่อจัดเก็บภาพหลาย ๆ ภาพไว้ในเอกสารเดียวกัน โดยใช้

หลักที่ว่าเอกสารหนึ่งเอกสารมีได้หลายหน้าและจากหลักการนี้ เราจึงนำมาประยุกต์กับการซีเรียลไลท์ โดยการเพิ่มตัวแปรเก็บเลขแต่ละหน้าในการซีเรียลไลท์ ดังนี้

```
void CSlideCntrItem::Serialize(CArchive& ar)
{
        ASSERT_VALID(this);
        if (ar.IsStoring())                    ขนาดจริงของออบเจกต์
        {
                ar<<m_rect<<m_byPageID<<m_exent;
        }
        else {   ขอบรอบออบเจกต์   ตัวแปรเก็บเลขหน้า
                ar>>m_rect>>m_byPageID>>m_exent;
        }
}
```

จากนั้นจึงเพิ่มฟังก์ชันในการแสดงออบเจกต์หรือภาพ ทีละภาพ ซึ่งมี 2 ฟังก์ชันคือ ฟังก์ชัน OnBackwrd() และ OnForward() ดังรายละเอียด ดังนี้

```
void CSlideView::OnBackward()
{
        if (m_byActivePage == 0)      /* เช็คว่า ถ้า M_byActivePage = 0
                m_Backward = FALSE;  จะไม่สามารถเลื่อนภาพถอยหลังได้ */
        else {
                m_byActivePage--;     // แสดงภาพก่อนหน้านี้
                Invalidate();         // update วินโดว์
        }
}
void CSlideView::OnForward()
{
        m_byActivePage++;     // แสดงภาพต่อไป
        Invalidate();
}
```

3.4 การออกแบบการจัดการเสียง

ในการออกแบบการทำงานของไฟล์เสียงนั้น เราจะวิเคราะห์ถึงความเหมาะสมของระบบที่เอื้ออำนวยซึ่งระบบในที่นี้หมายถึงฟังก์ชันมาตรฐานที่วินโดว์หรือ Visual C++ ซึ่งวินโดว์จะมี API (Application Program Interface) ที่สนันสนุนการทำงานกับไฟล์เสียงแบบ Wave และ Midi และฟังก์ชันที่เลือกใช้ในการจัดการไฟล์เสียงทั้ง 2 แบบคือ mciSendCommand midiOutSetVolume waveOutSetVolume จากตรงนี้ทำให้เราได้ออกแบบการจัดการไฟล์เสียงสองรูปแบบคือ ไฟล์เสียง wave (.wav) และไฟล์เสียง midi (.mid) เมื่อได้รูปแบบของไฟล์ที่ต้องการแล้วสามารถแบ่งออบเจกต์ได้เป็นสอง ออบเจกต์คือ ออบเจกต์ wave และ ออบเจกต์ midi ทำให้ออกแบบเป็นสองคลาสคือ คลาส wave และ คลาส midi จากแนวความคิดดังกล่าวทำให้ออกแบบคลาสทั้ง 2 คลาสดังรายละเอียดดังต่อไปนี้

```
class CMidi
{
        CMidi (CString);
        ~CMidi ();
        UINT mDeviceID;
private:
        CString mFilename;
        int mErrorCode;
        DWORD mMCIErrorCode;
public:
        BOOL Play (void);                   //เปิดไฟล์เสียง midi
        BOOL IsPlaying (void);              //จัดการเสียงขณะที่กำลังเล่นเพลง
        void Stop (void);                   //ปิดไฟล์เสียง midi
        CString GetErrorString (void);  //เช็คข้อผิดพลาด
        static int GetNumMidiDevices (void);  //รับค่าดีไวซ์
private:
        BOOL Open (CString);//เปิดไฟล์
};
class CWave
{
public:
        CWave (CString, int Type = WT_DISK);
        ~CWave ();
```

```
        UINT mDeviceID;
// Attributes
private:
        CString mFilename;
        int mErrorCode;
        int mType;
        HANDLE mResHndl;
        void *mpRes;
        DWORD mMCIErrorCode;
public:
        BOOL PlayAsync (void);          //เปิดไฟล์เสียง wave
        void Stop (void);               //ปิดไฟล์เสียง wave
        CString GetErrorString (void); //เช็คข้อผิดพลาด
        static int GetNumWaveDevices (void); //รับค่าดีไวซ์
private:
        BOOL Open (CString, int);       //เปิดไฟล์เสียง
};
```

3.5 การออกแบบระบบความช่วยเหลือแบบ Context-Sensitive Help

การออกแบบระบบความช่วยเหลือแบบ Context-Sensitive Help นี้จะอ้างอิงถึงเมนู และ ทูลบาร์เป็นหลักซึ่งเมื่อได้พิจารณาถึงเมนูของโปรแกรมสมารถออกแบบระบบความช่วยเหลือได้ดังนี้

ระดับที่ 1	ระดับที่ 2	ระดับที่ 3
File	New	สร้างไชด์วินโดว์ใหม่เพื่อสร้างสไลด์ใหม่
	Open	เปิดสไลด์ที่บันทึกไว้ในดิสก์
	Close	ปิดสไลด์ปัจจุบัน
	Save	บันทึกลงดิสก์ในชื่อเก่า
	Save As	บันทึกลงดิสก์เพื่อเปลี่ยนชื่อ
	Print	พิมพ์เอาสารที่แสดงบนหน้าจอ
	Print Preview	แสดงเอกสารก่อนพิมพ์
	Print Setup...	จัดรูปแบบของเอกสารหรือเลือกชนิดครื่องพิมพ์ก่อนพิมพ์จริง
	Exit	จบการทำงานและออกจากโปรแกรม

Edit	Cut	ตัดออบเจกต์ที่เลือกลงคลิปบอร์ด
	Copy	สำเนาออบเจกต์ที่เลือกลงคลิปบอร์ด
	Paste	แปะออบเจกต์จากคลิปบอร์ดลงเอกสาร
	Paste Link	เชื่อมออบเจกต์จากคลิปบอร์ดลงเอกสาร
	Delete	ลบออบเจกต์ที่เลือก
	Insert New Object	นำเข้าออบเจกต์ใหม่จากเซิฟเวอร์
	Object	คำสั่งที่เกี่ยวข้องกับออบเจกต์
View	Toolbar	แสดง / ไม่แสดง ทูลบาร์
	Status bar	แสดง / ไม่แสดง สเตตัสบาร์
	PlayWaveFile...	เปิดไฟล์เสียง wave (.wav)
Sound	PlayMidiFile...	เปิดไฟล์เสียง midi (.mid)
	StopWaveFile	ปิดเสียงไฟล์ wave
	StopMidiFile	ปิดเสียงไฟล์ midi
	Volume Control...	แสดง Dialog เพื่อปรับระดับเสียง
Slide	Forward	แสดงภาพถัดไป
	Backward	แสดงภาพก่อนหน้านี้
	New Window	เปิดวินโดว์ใหม่ที่มีเอกสารเดียวกับวินโดว์ปัจจุบัน
Window	Cascade	จัดเรียงวินโดว์ซ้อนเหลื่อมกัน
	TileHorizontally	จัดเรียงวินโดว์ตามแนวนิน
	Tile Verizontally	จัดเรียงวินโดว์ตามแนวตั้ง
	Arrange Icon	จัดเรียงไอคอนให้เป็นระเบียบ
Help	Slide Help	แสดงข้อความช่วยเหลือการใช้งานโปรแกรม
	Using Help	แสดงข้อความช่วยเหลือการใช้ Help
	About Slide	แสดง Dialog ข้อความการสร้างโปรแกรม
Toolbar	(รูปทูลบาร์ และบอกความหมายของแต่ละภาพ)	(ระดับ 3 ไม่มี)
	Double Click	In-Place Activate ออบเจกต์
	Left Arrow	เลื่อน Scroll Bar ไปทางซ้าย

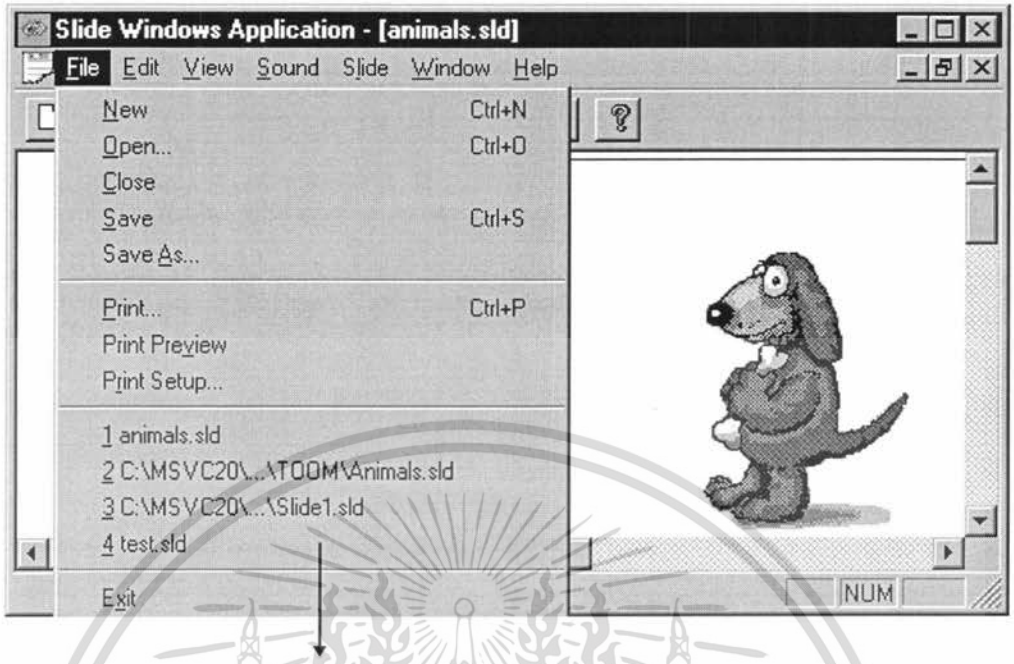
Other Command	Right Arrow	เลื่อน Scroll Bar ไปทางขวา
	Up Arrow	เลื่อน Scroll Bar ขึ้นบน
	Down Arrow	เลื่อน Scroll Bar ลงล่าง
	Page Down	แสดงเอกสารหน้าต่อไป
	Home	แสดงจุดเริ่มต้นเอกสาร
	End	แสดงจุดสิ้นสุดเอกสาร
	Page Up	แสดงเอกสารก่อนหน้านี้
	Right Button Click	เรียกใช้ popup เมนู
Project's Team	(รายชื่อผู้พัฒนาโปรแกรม)	

3.6 การติดต่อกับผู้ใช้

เมื่อรันโปรแกรมที่สมบูรณ์แล้วจะปรากฏหน้าจอดังรูป

รูปที่3.6.2 หน้าจอเมื่อรันโปรแกรมครั้งแรก

เมื่อเลือกเมนู File จะปรากฏคำสั่งต่าง ๆ ที่อยู่ภายใต้เมนูซึ่งแต่ละคำสั่งมีความหมายในการใช้งานดังต่อไปนี้

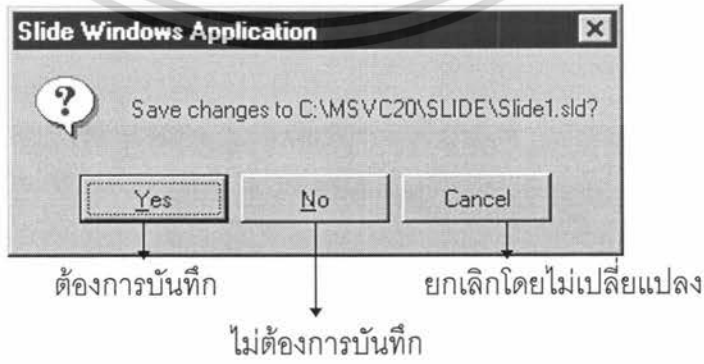


รูปที่3.6.3 แสดงคำสั่งในเมนู File

New - คำสั่งนี้จะสร้าง ไชด์วินโดว์ตัวใหม่เพื่อการสร้างสไลด์ใหม่

Open - คำสั่งนี้ใช้เมื่อต้องการเปิดไฟล์ที่เก็บอยู่ในดิสก์ โดยฟอร์แมตของไฟล์จะต้องมีนามสกุลเป็น .sld เท่านั้น

Close - จะปิดไชด์วินโดว์ที่กำลังทำงานอยู่ ถ้ามีการแก้ไขเอกสารในไชด์วินโดว์นั้นแต่ยังไม่ได้บันทึกจะมีข้อความเตือนดังนี้



รูปที่3.6.4 Dialog แสดงข้อความเตือนให้ Save

Save - บันทึกข้อมูลลงในชื่อปัจจุบัน ถ้าเป็นการบันทึกครั้งแรกจะแสดง Dialog box ให้ใส่ชื่อที่ต้องการบันทึก

Save As - บันทึกข้อมูลเป็นครั้งแรก หรือ ต้องการบันทึกโดยต้องการเปลี่ยนชื่อใหม่ซึ่งจะแสดง Dialog box ให้ใส่ชื่อที่ต้องการบันทึก

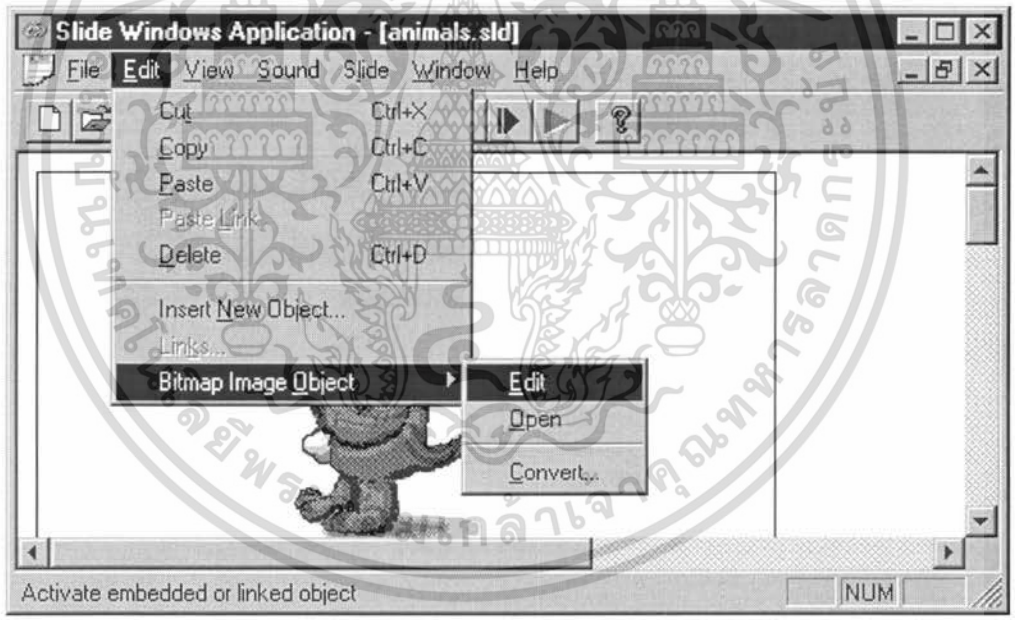
Print... - แสดง Dialog box มาตรฐานเพื่อการพิมพ์ข้อมูล

Print Preview - แสดงรูปแบบการวางหน้ากระดาษก่อนการพิมพ์จริง

Print Setup... - แสดง Dialog box มาตรฐานเพื่อตั้งค่าต่าง ๆ เกี่ยวกับการพิมพ์ เช่น ขนาดกระดาษ, ชนิดเครื่องพิมพ์ เป็นต้น

Exit - จบการทำงานและออกจากโปรแกรม

เมนู Edit จะมีคำสั่งที่เกี่ยวข้องกับออบเจกต์โดยตรง เช่น Cut, Copy, Delete เป็นต้น เมื่อเลือกเมนู Edit และเลือกคำสั่ง Microsoft ClipArt Gallery Object (คำสั่งนี้ ขึ้นกับชนิดของออบเจกต์ที่ถูกเลือก) จะปรากฏหน้าจอดังรูป และ รายละเอียดของคำสั่งมีดังนี้



รูปที่3.6.5 แสดงคำสั่งในเมนู Edit

Cut - ตัดออบเจกต์ที่เลือกลงคลิปบอร์ด

Copy - คัดลอกออบเจกต์ที่เลือกลงคลิปบอร์ด

Paste - วางออบเจกต์ในเอกสารจากคลิปบอร์ด

Paste Link - เชื่อมออบเจกต์จากคลิปบอร์ดลงเอกสาร

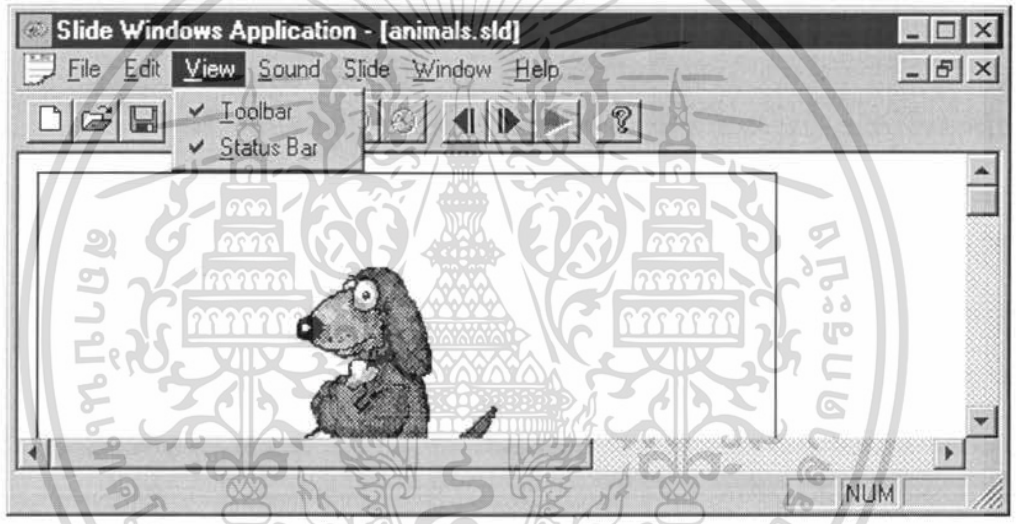
Delete - ลบออบเจกต์ที่เลือก

Insert New Object... - แสดง dialog box มาตรฐานในการ Embed หรือ ลิงก์ ออบเจกต์ลงเอกสาร

Links... - แสดง dialog box มาตรฐานในการเชื่อมออบเจกต์มายังเอกสาร

Object - จะเปลี่ยนไปตามชื่อของออบเจกต์ที่เลือกขณะนั้น เช่น ขณะนี้เลือกออบเจกต์จาก Microsoft ClipArt Gallery ก็จะปรากฏดังรูป และ ภายใต้คำสั่งนี้จะมีคำสั่งอีกซึ่งขึ้นอยู่กับชนิดของ ออบเจกต์เช่นกัน คำสั่งเหล่านี้จะเรียกว่า Verb ของออบเจกต์ที่สนับสนุน In-Place Activate

เมื่อเลือกเมนู View จะปรากฏหน้าจอดังรูป ซึ่งมีคำสั่ง 2 คำสั่งคือ ทูลบาร์ และ สเตตัสบาร์



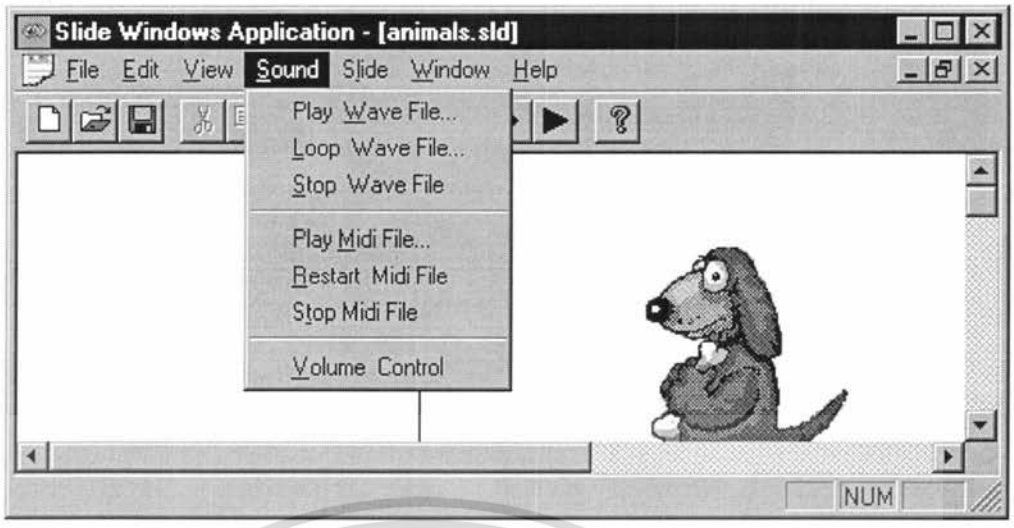
รูปที่3.6.6 แสดงคำสั่งในเมนู View

มีความหมายดังนี้

ทูลบาร์ - แสดงหรือไม่แสดงทูลบาร์ ถ้ามีเครื่องหมายขีดถูกแสดงว่าแสดงทูลบาร์

สเตตัสบาร์ - แสดงหรือไม่แสดงสเตตัสบาร์ถ้ามีเครื่องหมายขีดถูกแสดงว่าแสดงสเตตัสบาร์

เมนู Sound จะมีคำสั่งที่จัดารเกี่ยวกับเสียง ซึ่งแต่ละคำสั่งมีความหมายดังนี้



รูปที่3.6.7 แสดงคำสั่งในเมนู Sound

Play Wave File... - แสดง dialog box มาตรฐานในการเปิดไฟล์เพื่อจะเล่นเพลง โดยไฟล์ที่จะเลือกได้ต้องมีนามสกุลเป็น .wav เท่านั้น

Loop Wave File... - แสดง dialog box มาตรฐานในการเปิดไฟล์ โดยไฟล์ที่จะเลือกได้ต้องมีนามสกุล .wav เท่านั้น เล่นเพลงแบบวนเพลงนั้นจนกว่าจะกดปุ่มปิด

Stop Wave File - ปิดเสียงไฟล์เวฟที่กำลังเปิดอยู่

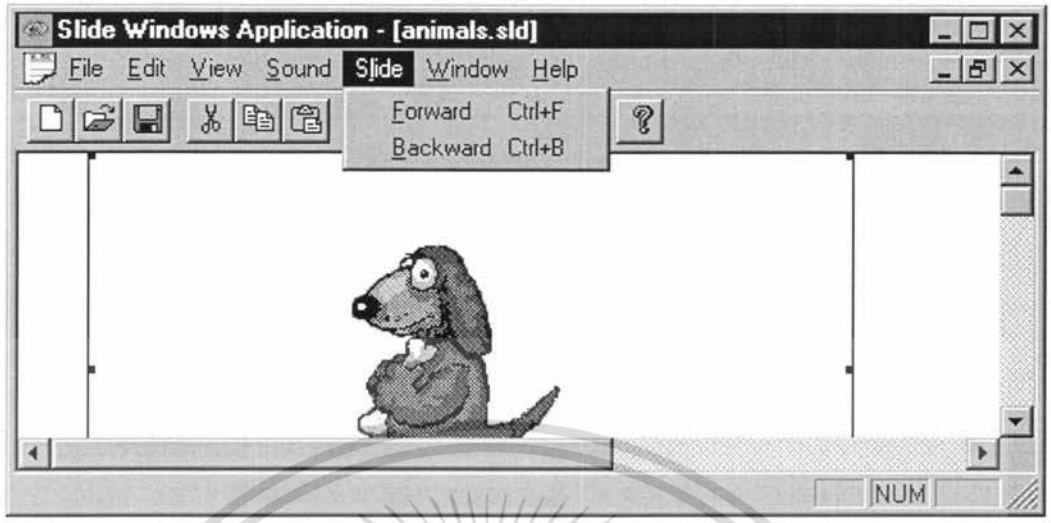
Play Midi File... - แสดง มาตรฐานในการเปิดไฟล์โดยฟอร์แมตของไฟล์เพื่อที่จะเล่นเพลงที่จะเปิดได้ต้องเป็นไฟล์เสียงนามสกุล .mid เท่านั้น

Restart Midi File - เล่นเพลงที่เลือกไว้แล้วอีกครั้ง

Stop Midi File - ปิดเสียงเพลงที่กำลังเปิดอยู่

Volume Control... - แสดง dialog box เพื่อปรับระดับความดังหรือค่อยของเสียง

เมนู Slide มีคำสั่ง Backward และ Forward ใช้ในการเลื่อนภาพย้อนหลัง หรือ ไปข้างหน้าทีละหนึ่งภาพดังรายละเอียดดังนี้

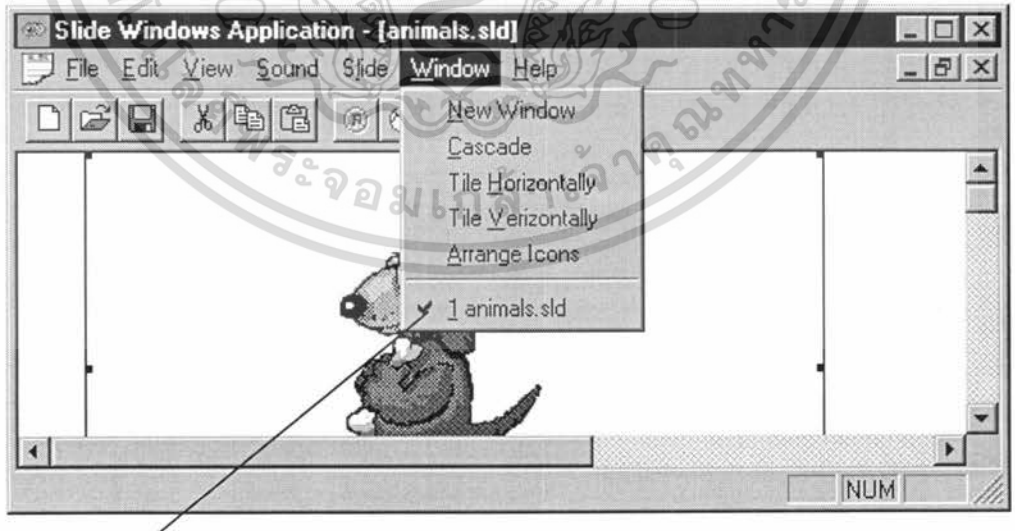


รูปที่3.6.8 แสดงคำสั่งในเมนู Slide

Backward - เลื่อนภาพย้อนหลังไปหนึ่งภาพ

Forward - เลื่อนภาพไปข้างหน้าหนึ่งภาพ

เมนู Window จะมีคำสั่งที่จัดการเกี่ยวกับไชด์วินโดว์ และ ไอคอนของเอกสาร พร้อมทั้งแสดงชื่อไฟล์ที่กำลังทำงานอยู่ รายละเอียดของคำสั่งในเมนู Window มีดังนี้



รูปที่3.6.9 แสดงคำสั่งในเมนูวิโดว์

New Window - สร้างไซด์วินโดว์ใหม่ที่มีข้อมูลเดียวกันกับไซด์วินโดว์ปัจจุบัน

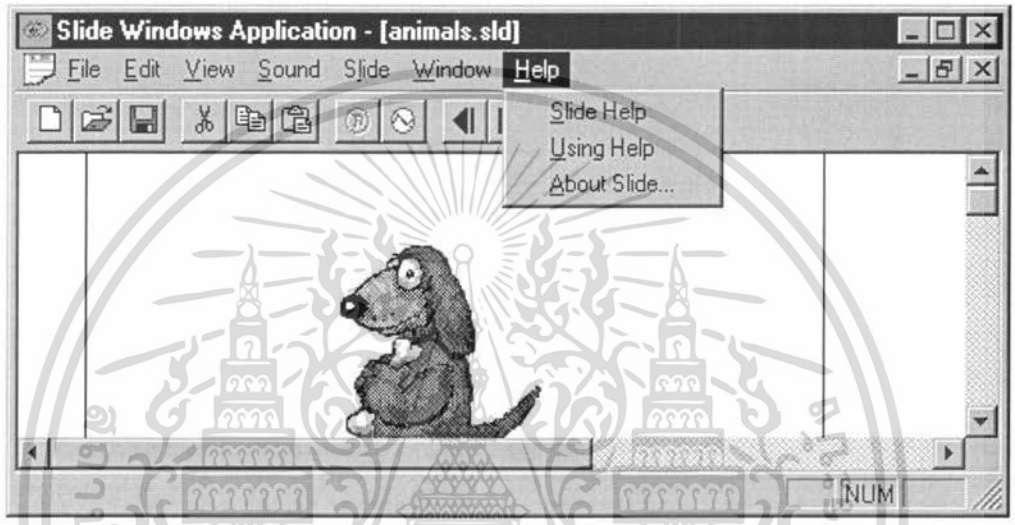
Cascade - จัดเรียงไซด์วินโดว์ซ้อนเหลื่อมกัน

Tile Horizontally - จัดเรียงไซด์วินโดว์ตามแนวนอน

Tile Verizontally - จัดเรียงไซด์วินโดว์ตามแนวตั้ง

Arrange Icons - จัดเรียงไอคอนที่เกิดจากการ minimize ไซด์วินโดว์ให้เป็นระเบียบ

เมนู Help มี 3 คำสั่งดังแสดงในรูป ซึ่งมีความหมายดังนี้



รูปที่3.6.10 แสดงคำสั่งในเมนู Help

Slide Help - แสดงข้อความช่วยเหลือที่เกี่ยวกับรายละเอียด และคำสั่งต่าง ๆ ในโปรแกรมทั้งหมด

Using Help - แสดงข้อความช่วยเหลือเกี่ยวกับการใช้คำสั่งใน Help

About Slide... - แสดง dialog box ที่มีข้อมูลเกี่ยวกับการสร้างโปรแกรม

บทที่ 4
การประเมินผล

ผลที่ได้จากการศึกษาและพัฒนาโปรแกรมที่ทำงานสนับสนุน OLE (Objct Linking and Embedding) และ เสียง โดยใช้หลักการเขียนโปรแกรมแบบ OOP (Object-Oriented programming) ซึ่งพัฒนาโปรแกรมบนวินโดว์ และ ใช้ Visual C++ เป็นเครื่องมือในการพัฒนา ภายใต้หัวข้อที่ว่า " OLE มัลติมีเดีย กับการเขียนโปรแกรมแบบ OOP บนวินโดว์ " ซึ่งโปรแกรมที่พัฒนาเป็นโปรแกรมการทำสไลด์ประกอบเสียงซึ่งเป็น OLE แบบ คอนเทนเนอร์ (Container) จากการศึกษาและพัฒนาโปรแกรมดังกล่าว สามารถประเมินผลได้ดังนี้

คุณสมบัติและความสามารถของโปรแกรม

1. ออบเจกต์ที่ Embeded หรือ Links เข้ามาจะมีขนาดเท่าของจริงไม่มีการย่อหรือขยาย แต่เราสามารถ ย่อหรือขยายเองได้โดยการเปลี่ยนแปลงกรอบที่ล้อมรอบออบเจกต์นั้น
2. สามารถเคลื่อนย้าย (Move) ออบเจกต์ไปที่ตำแหน่งต่าง ๆ ของวินโดว์ได้
3. สามารถ Embeded หรือ Links ออบเจกต์ได้หลายออบเจกต์ในแต่ละเฟรมของสไลด์
4. สามารถลบออบเจกต์ที่ไม่ต้องการออกจากระบบได้
5. โปรแกรมสนับสนุนการทำงานของคลิปบอร์ดได้ เช่น Cut, Copy, Paste เป็นต้น
6. โปรแกรมมีคุณสมบัติที่สนับสนุน OLE โดยสามารถ Embeded, Link, In-Place Activate, และ Drag&Drop ได้
7. โปรแกรมสนับสนุนการทำงานแบบ MDI (Multiple Document Interface) ซึ่งสามารถสร้างและแสดงเอกสาร หรือสไลด์ได้หลายสไลด์ในเวลาเดียวกัน
8. โปรแกรมที่พัฒนาจะมีทูลบาร์ และ สเตตัสบาร์ เพื่อความสะดวกและง่ายแก่การใช้งานและเข้าใจ
9. สามารถเรียกใช้ popup เมนู เพื่อจัดการกับออบเจกต์ได้ด้วยการคลิกเมาส์ขวา
10. โปรแกรมที่พัฒนาใช้วิว (View) แบบ ScrollView จึงสามารถเลื่อน scroll เพื่อดูภาพที่มีขนาดใหญ่กว่าหน้าจอได้
11. การอินเทอร์เฟซของโปรแกรมใช้ระบบพูลดาวเมนู และ ทูลบาร์แบบเดียวกับวินโดว์ซึ่งคำสั่งในเมนูจะมีคีย์ด่วนให้ใช้เพื่อความสะดวกในการใช้งาน
12. โปรแกรมสนับสนุนการเปิดไฟล์เสียง 2 แบบ คือ Wave และ Midi ซึ่งสามารถปรับระดับความดังของเสียงที่กำลังเปิดอยู่ได้
13. โปรแกรมมีระบบให้ความช่วยเหลือแบบ Context-Sensitive Help ซึ่งเป็นระบบความช่วยเหลือที่เป็นที่นิยมและเป็นมาตรฐานบนวินโดว์

14. โปรแกรมพัฒนาบน Visual C++ 1.5 และ 2.0 และสามารถรันได้บนวินโดว์ 3.1(Slide.exe) และวินโดว์95 (Slide32.exe)

อย่างไรก็ตามโปรแกรมที่พัฒนาขึ้นมานี้ยังมีข้อจำกัดหลายประการดังต่อไปนี้

1. ภาพหรือออบเจกต์ที่ใช้จะต้องนำเข้าหรือลิงก์จากเซิฟเวอร์ที่มีในระบบ เท่านั้นไม่สามารถเปิดขึ้นมาเองได้
2. ภาพที่มีความละเอียดสูงและมีจำนวนสีมากเมื่อนำเข้าบางครั้งสีจะเพี้ยนดูไม่สมจริง
3. การพิมพ์ข้อความโปรแกรมไม่มีความสามารถที่จะพิมพ์ได้เองแต่ต้องพิมพ์ผ่านทางไมโครซอฟเวิร์ด หรือโปรแกรมอื่น ๆ ที่มีสามารถพิมพ์ข้อความได้และเป็นเซิฟเวอร์
4. การปรับระดับดัง-ค่อยของเสียงยังขาดความสมบูรณ์ คือเมื่อปรับเสร็จแล้วต้องออกจากไดอะล็อก Volume Control ก่อน การเปลี่ยนแปลงจึงจะเกิดขึ้น
5. ไม่สามารถลากและวางออบเจกต์ไปยังโปรแกรมอื่นได้ จะสามารถลากและวางได้ภายในโปรแกรมเท่านั้น

บทที่ 5
สรุปผลและข้อเสนอแนะ

สรุปผล

ผลที่ได้จากการศึกษาและพัฒนาโปรแกรมสร้างสไลด์ประกอบเสียง โดยใช้การเขียนโปรแกรมแบบ OOP (Object Oriented Programming) และ โปรแกรมที่พัฒนาขึ้นสนับสนุน OLE (Object Linking and Embeding) โดยโปรแกรมมีความสามารถในการสร้างสไลด์ประกอบเสียง และมีระบบความช่วยเหลือแบบ Context-Sensitive Help ซึ่งโปรแกรมที่ได้มีความยืดหยุ่นในการใช้งานมาก โดยผู้ใช้สามารถ Embeded หรือ Link ออบเจกต์ที่ต้องการจากเซิฟเวอร์ ที่มีในระบบได้ทุกชนิด และสามารถจัดการออบเจกต์เหล่านั้นได้ เช่น การย่อหรือขยายออบเจกต์, การลากและวางออบเจกต์ เป็นต้น การอินเทอร์เฟซจะใช้ระบบเมนูและทูลบาร์เช่นเดียวกับวินโดว์ซึ่งง่ายและสะดวกแก่การใช้งาน และยังมีสเตตัสบาร์เพื่อบอกถึงสิ่งที่กำลังทำอยู่ นอกจากนั้นทางด้านเสียงโปรแกรมสามารถเปิดเสียงแบบ Wave และ Midi และปรับระดับเสียงได้

ข้อเสนอแนะ

1. ควรเพิ่มเติมให้โปรแกรมสนับสนุนฟอร์แมตรูปภาพแบบ .gif ได้ และควรจะสามารถนำเข้ารูปภาพวิธีอื่นได้
2. ควรเพิ่มเติมในส่วนการจัดการข้อความให้สามารถพิมพ์ข้อความด้วยตัวของโปรแกรมเองได้
3. ควรมีเครื่องมือในการวาดรูปในโปรแกรม
4. การปรับระดับเสียงควรปรับปรุงให้เห็นผลทันทีขณะปรับระดับเสียง
5. ควรเพิ่มความสามารถในการลากและวางออบเจกต์ให้สามารถลากและวางข้ามโปรแกรมได้

ภาคผนวก

การติดตั้งระบบ

ฮาร์ดแวร์

- เครื่องคอมพิวเตอร์ที่มี CPU ตั้งแต่รุ่น 486 ขึ้นไป
- มีเนื้อที่ว่างในฮาร์ดดิสก์อย่างน้อย 3 เมกกะไบต์
- ควรมีหน่วยความจำ 8 เมกกะไบต์ขึ้นไป
- ไมโครซอฟท์เมาส์ หรือที่เข้ากันได้
- จอแสดงผลสี

ซอฟท์แวร์

- โปรแกรมระบบปฏิบัติการไมโครซอฟท์วินโดว์95
- โปรแกรมให้บริการที่เป็นเซิฟเวอร์ เช่น ไมโครซอฟท์เอกเซล

ไฟล์ที่ใช้ทำงาน

- Slide.exe
- Slide32.exe
- Slide32.hlp

บรรณานุกรม

โรเบิร์ต ลาฟอเร่, ผู้เรียบเรียง ราบินเดอร์ ศรีกิจจาภรณ์ , **การเขียนโปรแกรมแบบโอโอพีด้วยเทอร์โบและบอร์แลนด์ C++**

COMPUER REVIEW ฉบับที่ 116 ประจำเดือน เมษายน 2537 , หน้า 132 ถึง 133 , หน้า 137 ถึง 142

DAVID A. HOLZGANG , **teach yourself Visaul C++,**1995

DAVID J. KRUGLINSKI, ผู้เรียบเรียง โชคชัย เตชพรรุ่ง, **การเขียนโปรแกรม Visual C++ เวอร์ชัน 1.5 บนวินโดวส์**

ANTHONY PORTER, **C++ Programming for Windows,** 1993

Richard C. Leinecker & Jamie Nye, **Visual C++ Power Toolkit**,1995