บทที่ 3

เทคนิคการเขียนโปรแกรมบนไมโครซอฟต์วินโดวส์

ความเป็นมาของไมโครซอฟต์วินโดวส์

กิตศิลป์ แตระกุล (1992) ได้กล่าวถึงความเป็นมาของไมโครซอฟต์วินโดวส์คือ นับแต่ปี ค.ศ. 1981 บริษัทไอบีเอ็มได้ออกจำหน่ายคอมพิวเตอร์ส่วนบุคคลเป็นครั้งแรก ซึ่งบริษัทไมโครซอฟต์ก็ได้ออกจำหน่ายเอ็มเอส-ดอสด้วยเช่นกัน และเป็นการเริ่มต้นของการปฏิวัติวงการพีซี (Personal Computer: PC) ตั้งแต่ปี ค.ศ. 1980 เป็นต้นมา มีผู้รู้จักคำสั่งของเอ็มเอส-ดอสและใช้โปรแกรมประยุกต์ต่างๆ หลายล้านคนทีเดียว เมื่อสิ้นทศวรรษนี้ โดยทั่วๆ ไปแล้วผู้ใช้ส่วนใหญ่จะใช้โปรแกรมเกี่ยวกับการประมวลผลคำ (word processing), สเปรดชีท (spreadsheet) และฐานข้อมูล (database) ความจริงแล้วผู้ใช้ส่วนใหญ่ยังมีความต้องการอยากที่จะทำการแลกเปลี่ยนข้อมูลระหว่างโปรแกรมให้ได้ด้วยวิธีการที่ง่าย หรือหาวิธีการที่จะรวบรวมความต้องการทั้งหมดให้จบในโปรแกรมตัวเดียว ก่อนที่จะมองไปที่ข้อมูลที่เก็บโดยตัวอื่น

ต่อมาในปี ค.ศ. 1983 บริษัทไมโครซอฟต์ได้เปิดตัวซอฟต์แวร์ไมโครซอฟต์วินโดวส์เป็นครั้งแรก ซึ่งเป็นระบบการทำงานในสภาพแวดล้อมแบบกราฟิก ทำหน้าที่เป็นตัวเชื่อมต่อระหว่างผู้ใช้งานกับอุปกรณ์ต่างๆ ของคอมพิวเตอร์ ทำให้เกิดความสะดวกในการใช้งาน โดยเริ่มวางจำหน่ายวินโดวส์รุ่น 1.01 ในอีก 2 ปีถัดมา หลังจากนั้นไมโครซอฟต์ ได้ทำการพัฒนาขีดความสามารถเพิ่มขึ้นให้สามารถรองรับกับอุปกรณ์ต่างๆ ได้หลากหลายชนิด และออกรุ่น 2.0 ในปี ค.ศ. 1987 ซึ่งกล่าวได้ว่ารุ่นนี้ได้มีการเปลี่ยนแปลงไปจากรุ่น 1.01 ค่อนข้างมาก โดยวินโดวส์ที่เปิดขึ้นมาสามารถซ้อนทับกันได้หลายๆ ชั้น รวมทั้งยังสนับสนุนการใช้แผงแป้นอักขระร่วมกับเมาส์อีกด้วย

ภายหลังจากออกวินโดวส์รุ่น 2.0 ไม่นานนัก ไมโครซอฟต์ก็ได้ออกวินโดวส์รุ่น/386 เพื่อใช้กับเครื่องไมโครคอมพิวเตอร์ที่ใช้ซีพียู 80386 โดยตรง ซึ่งในรุ่นนี้ ไมโครซอฟต์ได้เพิ่มขีดความสามารถของวินโดวส์ขึ้นมาก กล่าวคือได้พัฒนาให้ใช้ความสามารถพิเศษของซีพียู 80386 คือการทำงานในภาวะเสมือน-86 (Virtual-86 mode) ทำให้สามารถนำโปรแกรมที่ทำงานภายใต้ระบบปฏิบัติการดอสโดยตรงมาทำงานบนไมโครซอฟต์วินโดวส์ได้พร้อมๆ กันหลายโปรแกรม และได้เปลี่ยนชื่อของไมโครซอฟต์วินโดวส์รุ่น 2.0 เป็นไมโครซอฟต์วินโดวส์/286 เพื่อให้เกิดความสอดคล้องกัน

ในปี ค.ศ. 1990 ไมโครซอฟต์ได้ออกวินโดวส์รุ่น 3.0 ซึ่งได้มีการเปลี่ยนแปลงพัฒนาไปจากรุ่นเดิมมาก โดยได้รวมไมโครซอฟต์วินโดวส์ทั้งสองรุ่นเข้าด้วยกัน ปรับปรุงการทำงานของเปลือก (shell) ของไมโครซอฟต์วินโดวส์ให้ดีขึ้น สนับสนุนการทำงานกับระบบเครือข่าย (network) และรูปแบบของแผนที่บิตชนิดใหม่ที่ไม่ขึ้นกับอุปกรณ์ต่างๆ (Device Independent Bitmap : DIB) นอกจากนี้ส่วนที่มีการเปลี่ยนแปลงที่สำคัญที่สุดคือการสนับสนุนการใช้งานหน่วยความจำแบบยืดขยาย (extended memory) โดยสามารถอ้างอิงหน่วยความจำได้ถึง 16 เมกะไบต์ รวมทั้งถ้าหากทำงานบนซีพียู 80386 หรือสูงกว่าในภาวะ 386-Enhanced แล้ว ไมโครซอฟต์วินโดวส์จะใช้การจัดการหน่วยความจำแบบเสมือนของซีพียู ซึ่งจะทำให้สามารถใช้หน่วยความจำได้มากกว่าที่มีอยู่จริงถึง 4 เท่า

พิชญะ จงตระกูล (2536) ได้กล่าวถึงพัฒนาการของไมโครซอฟต์วินโดวส์ไว้คือ วินโดวส์รุ่น 3.0 ก็ยังมีข้อจำกัดในเรื่องการแสดงผลของตัวอักษร เพราะตัวอักษรที่ใช้เป็นแบบแผนที่บิต ทำให้การแสดงผลตัวอักษรขนาดใหญ่ไม่สวยงามเนื่องจากนำแผนที่บิตมาขยาย จึงได้ร่วมมือกับบริษัทแอปเปิลพัฒนารูปแบบตัวอักษรใหม่ขึ้นในรุ่น 3.1 ในปี ค.ศ.1992 เรียกว่ารูปแบบอักษรทรูไทป์ (True Type Font) ซึ่งเป็นตัวอักษรแบบโครงร่าง ทำให้สามารถย่อ-ขยายตัวอักษรได้ตามความต้องการ นอกจากนี้ก็ยังได้พัฒนาขีดความสามารถแลกเปลี่ยนข้อมูลที่เรียกว่าโอแอลอี (Object Linked and Embedded : OLE) สนับสนุนระบบหลายสื่อ (Multimedia) รวมทั้งปรับปรุงโปรแกรมต่างๆ ให้มีประสิทธิภาพดียิ่งขึ้น และในรุ่นนี้ไมโครซอฟต์ก็ได้ตัดการสนับสนุนการใช้งานไมโครซอฟต์วินโดวส์บนเครื่องคอมพิวเตอร์ที่มีซีพียู 8086 หรือที่เรียกว่าภาวะจริง (real mode)

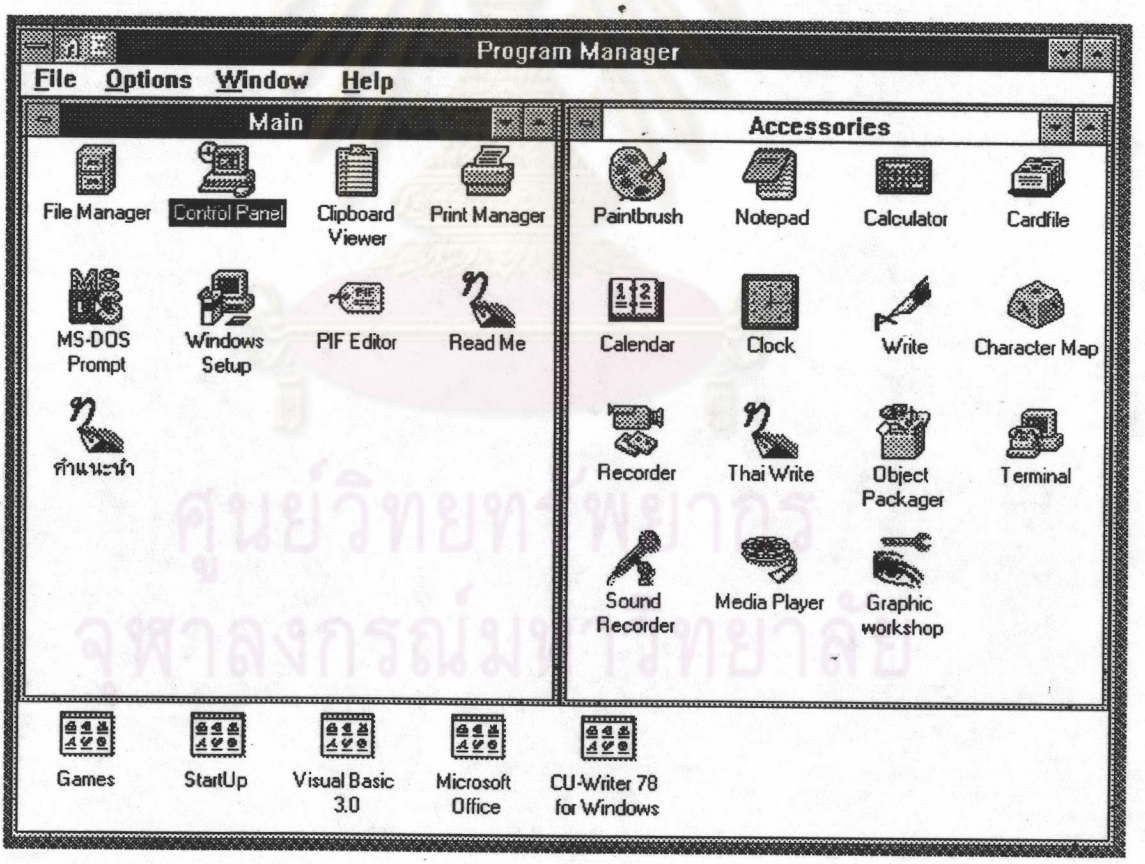
จนมาถึงปลายปี ค.ศ.1995 ไมโครซอฟต์ก็ได้ปฏิวัติครั้งใหญ่โดยได้เปลี่ยนโฉมวินโดวส์ใหม่อีกครั้ง และใช้ชื่อใหม่เป็นวินโดวส์ 95 การเปลี่ยนแปลงที่ต่างจากเดิมของวินโดวส์ 95 ก็คือไมโครซอฟต์ได้ทำให้วินโดวส์ของตนกลายเป็นระบบปฏิบัติการแบบ 32 บิตอย่างแท้จริง ซึ่งส่งผลให้วินโดวส์สามารถทำงานได้อย่างมีประสิทธิภาพสูงและเร็วขึ้น ทั้งยังถูกออกแบบให้ใช้งานได้ง่าย และสนับสนุนการทำงานกับอุปกรณ์คอมพิวเตอร์อื่นๆ อีกมาก ไม่ว่าจะเป็นอุปกรณ์สำหรับคอมพิวเตอร์ 486, Pentium ทั้งแบบ Desktop และ Notebook

ระบบการประสานกับผู้ใช้แบบกราฟิกของวินโดวส์ 95 ก็ได้รับการปรับเปลี่ยนใหม่ทั้งหมด สัญรูป และวินโดวส์ต่างๆ ถูกลดรูปลงเป็นเมนูแบบดึงขึ้น จากปุ่ม Start ทางมุมซ้ายของแผงงาน (Task bar) ทางด้านล่างของจอภาพ และเรียกเมนูชนิดนี้ว่า Short Cut

วินโดวส์ 95 เป็นระบบปฏิบัติการที่สนับสนุนการใช้งานระบบหลายสื่อ ระบบ Plug & Play และระบบเครือข่ายแบบใหม่ล่าสุดช่วยให้การเชื่อมต่อเข้าสู่ระบบอินเตอร์เน็ตที่เป็นเครือข่าย

คอมพิวเตอร์ระดับโลกได้อย่างสะดวก นอกจากนี้วินโดวส์ 95 ยังสนับสนุนในเรื่องชื่อของแฟ้มที่ยาวได้ถึง 255 ตัวอักษร และยังมีโปรแกรมสำหรับเรียกดูแฟ้มได้อย่างรวดเร็ว พร้อมโปรแกรมเสริมอีกจำนวนมาก แต่จุดที่น่าสนใจมากที่สุดของวินโดวส์ 95 ก็คือการใช้และจัดการทรัพยากรของระบบที่ดีกว่าวินโดวส์รุ่นก่อนๆ เป็นอย่างมาก

นอกจากที่กล่าวแล้วไมโครซอฟต์ ยังได้ผลิตวินโดวส์ในรุ่นต่างๆ หลายรุ่นที่สนับสนุนการทำงานแบบกลุ่มคือ Windows for Workgroup และวินโดวส์รุ่นที่สามารถทำงานกลับเครือข่ายคอมพิวเตอร์โดยเฉพาะระบบเครือข่ายท้องถิ่น (Local Area Network: LAN) และระบบ Client/Server คือ Windows NT แม้ว่าวินโดวส์แต่ละรุ่นของไมโครซอฟต์ จะยังมีจุดบกพร่องที่เป็นปัญหาอยู่ไม่น้อย แต่ก็ได้รับการต้อนรับจากผู้ใช้จำนวนมาก จนส่งผลให้ยอดขายพุ่งสูง และเสริมสร้างให้ไมโครซอฟต์กลายเป็นบริษัทผลิตซอฟต์แวร์ที่ใหญ่เป็นที่หนึ่งของโลกในที่สุด รูปแบบจอภาพของไมโครซอฟต์วินโดวส์รุ่น 3.1 ซึ่งเป็นรุ่นที่ใช้ในการพัฒนาโปรแกรมวาดภาพนี้ ได้แสดงไว้ในรูปที่ 3.1



รูปที่ 3.1 แสดงจอภาพของไมโครซอฟต์วินโดวส์ 3.1

จุดเด่นของไมโครซอฟต์วินโดวส์

การที่ไมโครซอฟต์วินโดวส์ได้กลายเป็นซอฟต์แวร์ระบบปฏิบัติการที่ได้รับความนิยมแพร่หลาย ย่อมมีสาเหตุจากจุดเด่นหลายๆ ประการที่ไมโครซอฟต์วินโดวส์มีอยู่เหนือกว่าซอฟต์แวร์คู่แข่งอื่นๆ พิชญะ จงตระกูล (2536) ได้สรุปจุดเด่นของไมโครซอฟต์วินโดวส์ ไว้ดังนี้

1. ระบบตัวประสานกับผู้ใช้แบบกราฟิก (Graphic User Interface : GUI)

เป็นการติดต่อกับผู้ใช้ผ่านทางรูปภาพกราฟิกที่สื่อความหมายถึงการทำงานต่างๆ เช่น สัญรูป และ Scroll bar เป็นต้น ทำให้เกิดความสวยงาม มีความน่าใช้ สะดวกและง่ายต่อการเรียนรู้ นอกจากนี้ยังมีความเป็นมาตรฐาน โดยโปรแกรมต่างๆ จะมีระบบตัวประสานกับผู้ใช้เหมือนๆ กัน

2. การทำงานแบบหลายภารกิจ (Multitasking)

สภาพแวดล้อมของไมโครซอฟต์วินโดวส์ ยินยอมให้ผู้ใช้เรียกใช้โปรแกรมมาทำงานได้หลายๆ โปรแกรมพร้อมๆ กัน โดยไมโครซอฟต์วินโดวส์จะเป็นผู้จัดสรรหน่วยความจำให้แต่ละโปรแกรมอย่างพอเพียงทั้งคอยจัดการการใช้ทรัพยากรต่างๆ ร่วมกัน

3. ความเป็นอิสระต่ออุปกรณ์

เนื่องจากไมโครซอฟต์วินโดวส์เป็นสภาพแวดล้อมที่เป็นตัวกลางระหว่างโปรแกรมที่เขียนขึ้นกับอุปกรณ์ต่างๆ โปแกรมจึงไม่ต้องคอยจัดการกับอุปกรณ์ต่างๆ โดยตรง เพียงแต่เรียกใช้ฟังก์ชันต่างๆที่ไมโครซอฟต์วินโดวส์จัดเตรียมไว้ให้ถูกต้องเท่านั้น ทำให้ผู้ใช้สามารถเปลี่ยนอุปกรณ์ในระบบได้เลยโดยไม่จำเป็นต้องเปลี่ยนแปลงโปรแกรมที่มีอยู่

4. การแลกเปลี่ยนข้อมูลระหว่างโปรแกรม

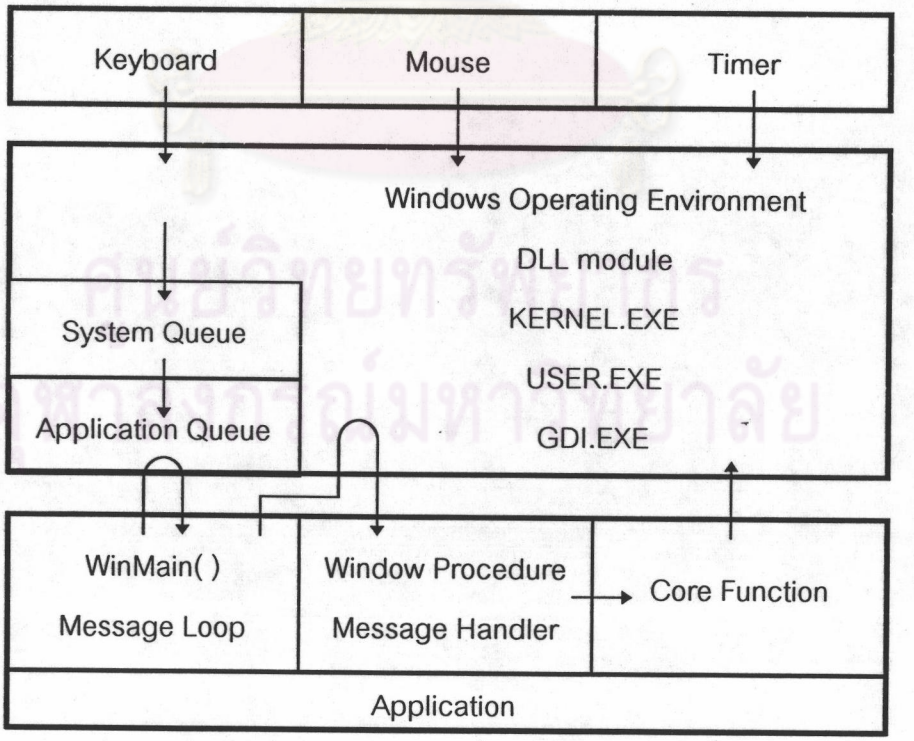
ไมโครซอฟต์วินโดวส์ได้สนับสนุนการโอนย้ายแลกเปลี่ยนข้อมูลระหว่างโปรแกรมได้ง่ายด้วยการคัดลอกข้อมูลจากโปรแกรมหนึ่งเข้าไปยังคลิปบอร์ด (clipboard) แล้วจึงคัดลอกจากคลิปบอร์ดไปยังอีกโปรแกรมหนึ่งได้ นอกจากนี้ยังมีขีดความสามารถในการเชื่อมโยงแฟ้มข้อมูลเข้าด้วยกันแบบดีดีอี (Dynamic Data Exchange: DDE) และโอแอลอี (Object Linked and Embedded: OLE)

การทำงานของไมโครซอฟต์วินโดวส์

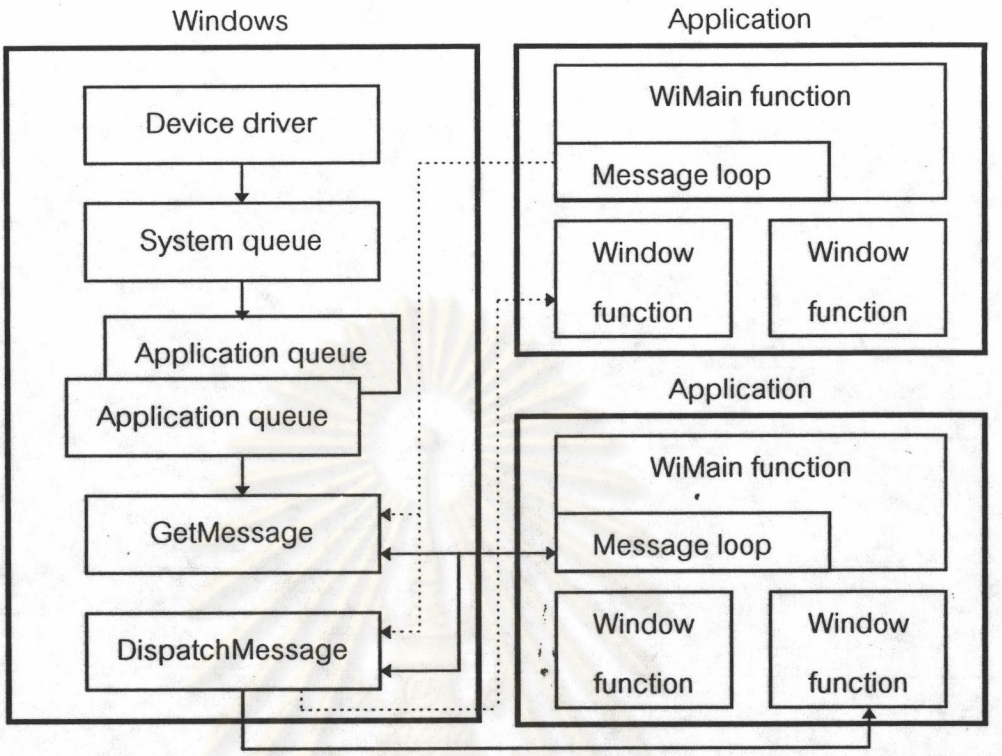
ไมโครซอฟต์วินโดวส์ในรุ่นที่ผ่านมา ได้แก่ วินโดวส์ 3.1 และ วินโดวส์ ฟอร์ เวิร์คกรุ๊พ ยังคงการทำงานเสมือนแบบหลายงาน กล่าวคือ แม้จะดูเหมือนวินโดวส์สามารถสั่งประมวลผลโปรแกรมได้

หลายๆ โปรแกรมพร้อมกัน แต่หากจะสังเกตให้ดีจะเห็นถึงช่วงจังหวะที่ต้องรอคอยโดยเฉพาะเมื่อวินโดวส์ต้องทำงานกับอุปกรณ์รับข้อมูล (I/O device) ได้แก่ เครื่องพิมพ์ และจานบันทึกแบบแข็ง (hard disk) เป็นต้น การทำงานของวินโดวส์จะหยุดชะงักไปอย่างเห็นได้ชัด ในขณะที่วินโดวส์ 95 ได้แก้ไขปัญหานี้แล้ว โดยไมโครซอฟต์ได้พัฒนาให้วินโดวส์ 95 เป็นระบบปฏิบัติการแบบ Preemptive multitasking อย่างแท้จริง แต่ในงานวิจัยนี้ ยังจำกัดอยู่ที่การพัฒนาโปรแกรมบนไมโครซอฟต์วินโดวส์ 3.1 และวินโดวส์ฟอร์ เวิร์คกรุ๊ฟ จึงจำต้องศึกษาการทำงานในแบบเดิม

พิชญะ จงตระกูล (2536) ได้อธิบายการทำงานของวินโดวส์รุ่น 3.1 ซึ่งเป็นรุ่นที่ใช้ทำงานวิจัยไว้ คือ ไมโครซอฟต์วินโดวส์จะเป็นตัวเชื่อมต่อระหว่างโปรแกรมประยุกต์กับอุปกรณ์ฮาร์ดแวร์ โดยจะมีฟังก์ชันการติดต่อให้โปรแกรมประยุกต์ต่างๆ เรียกใช้เอพีไอ (Application Programming Interface: API) (ดังรูปที่ 3.2) และไมโครซอฟต์จะเป็นตัวคอยจับสัญญาณจากอุปกรณ์รับข้อมูล (input device) อาทิเช่น แผงแป้นอักขระ และเมาส์ เป็นต้น เมื่ออุปกรณ์เหล่านี้ให้กำเนิดสัญญาณมา วินโดวส์ก็จะสร้างข้อความ (message) ที่เหมาะสมกับสัญญาณนั้นๆ เก็บไว้ในแถวคอยของระบบ (system queue) ตามหลักเข้าก่อนออกก่อน (First-In-First-Out: FIFO) จากนั้นวินโดวส์จะพิจารณาว่าข้อความต่างๆ นั้นเป็นของโปรแกรมประยุกต์ใด แล้วจึงส่งข้อความนั้นไปยังแถวคอยของโปรแกรมประยุกต์นั้นๆ โดยที่แต่ละโปรแกรมประยุกต์ก็จะมีแถวคอยของโปรแกรมประยุกต์ (application queue) ของตนเองคอยรับข้อความที่ส่งมา จากนั้นโปรแกรมประยุกต์จึงนำข้อความที่ได้รับไปประมวลผลอีกทีหนึ่ง



รูปที่ 3.2 แสดงการทำงานของไมโครซอฟต์วินโดวส์และโปรแกรมประยุต์



รูปที่ 3.3 แสดงการไหลของข้อความในการทำงานของไมโครซอฟต์วินโดวส์

จากรูปที่ 3.3 เมื่อเกิดข้อความเช่น ผู้ใช้กดแผงแป้นอักขระ วินโดวส์จะเก็บข้อความไว้ที่แถวคอยของระบบ และพิจารณาว่าเป็นข้อความของโปรแกรมประยุกต์ใด แล้วส่งไปเก็บไว้ที่แถวคอยของโปรแกรมประยุกต์นั้น จากนั้นโปรแกรมประยุกต์จะรับข้อความมาตรวจสอบว่าเป็นข้อความการหยุดโปรแกรมหรือไม่ ถ้าใช่ก็จะหยุดการทำงานของโปรแกรมนั้น แต่ถ้าไม่ใช่ก็จะจัดส่งกลับไปยังวินโดวส์เพื่อให้วินโดวส์จัดส่งไปยังวินโดวส์ฟังก์ชันของโปรแกรมประยุกต์ที่เหมาะสมเพื่อประมวลผลข้อความนั้นตามต้องการต่อไป

ลักษณะการทำงานของวินโดวส์โดยใช้ข้อความนี้เอง ทำให้วินโดวส์สามารถทำงานได้หลายงานพร้อมๆ กัน โดยการสลับการทำงานของแต่ละโปรแกรมประยุกต์จะขึ้นอยู่กับข้อความ กล่าวคือถ้าโปรแกรมประยุกต์ที่กำลังทำงานอยู่ไม่มีข้อความในแถวคอย แต่มีอีกโปรแกรมประยุกต์ที่มีข้อความรออยู่ในแถวคอย วินโดวส์ก็จะสลับการทำงานไปยังโปรแกรมประยุกต์หลังแทน ซึ่งการสลับการทำงานแบบนี้เรียกว่า Jumpy Multitasking หรือ Nonpreemptive Multitasking

ลักษณะของโปรแกรม

ไมโครซอฟต์วินโดวส์ เป็นระบบปฏิบัติการที่มีสภาพแวดล้อมแบบกราฟิก และยอมให้มีการพัฒนาโปรแกรมขึ้นทำงานภายใต้สภาพแวดล้อมของวินโดวส์ได้ โดยผ่านฟังก์ชันเอพีไอที่วินโดวส์จัดเตรียมไว้ให้ และใช้ภาษาคอมพิวเตอร์ที่มีตัวแปลภาษา (compiler) ให้สามารถทำงานร่วมกับวินโดวส์ได้ โดยตัวแปลภาษาเหล่านั้นจะทำการเชื่อมต่อกับฟังก์ชัน API ของวินโดวส์ให้เอง พิชญะ จงตระกูล (2536) ได้แบ่งลักษณะของโปรแกรมที่เขียนขึ้นสำหรับการทำงานบนไมโครซอฟต์วินโดวส์เป็น 2 ลักษณะ คือ

1. Executable Files (.EXE files)

เป็นโปรแกรมที่สามารถทำงานได้ด้วยตนเองเช่นเดียวกับโปรแกรมต่างๆ ที่เขียนขึ้นบนระบบปฏิบัติการดอสทั่วๆ ไป เพียงแต่ต้องสั่งประมวลผลวินโดวส์ขึ้นมาก่อน แล้วจึงเรียกโปรแกรมเหล่านี้มาทำงานได้

2. Dynamic Link Libraries Files (.DLL files)

เป็นโปรแกรมชนิดหนึ่งที่ไม่สามารถทำงานได้ด้วยตนเอง กล่าวคือโปรแกรมประเภทนี้จะเป็นการรวบรวมฟังก์ชันการทำงานต่างๆ ไว้ ให้โปรแกรมประเภทแรกเรียกใช้ ความแตกต่างที่สำคัญของโปรแกรมประเภทนี้ก็คือหน่วยความจำที่ใช้ เนื่องจากโปรแกรมที่ทำงานบนไมโครซอฟต์วินโดวส์สามารถทำงานได้พร้อมๆ กัน วินโดวส์จะจัดสรรหน่วยความจำให้ทุกครั้งที่มีการเรียกใช้โปรแกรมประเภทแรก นั่นคือ แต่ละโปรแกรมก็จะมีหน่วยความจำของตนเอง ในขณะที่โปรแกรมประเภท .DLL นี้ วินโดวส์จะจัดสรรหน่วยความจำให้เพียงชุดเดียวและถูกเรียกใช้จากโปรแกรมหลายๆ โปรแกรมได้ ทำให้สามารถประหยัดหน่วยความจำ และขนาดของโปรแกรมแบบ .EXE นั้นก็จะมีขนาดเล็กลง อีกทั้งการแก้ไขฟังก์ชันการทำงานต่างๆ ก็ทำกับโปรแกรมประเภท .DLL นี้เพียงอย่างเดียวไม่ต้องไปแก้ไขโปรแกรมแบบ .EXE อีก

เนื่องจากงานวิจัยนี้ ได้ทำการพัฒนาโปรแกรมด้วยไมโครซอฟต์วิชวลเบสิก จึงจำต้องกล่าวถึงลักษณะของโปรแกรมอีกประเภทหนึ่ง คือ โปรแกรมประเภท Visual Basic Extention (.VBX files) โดยโปรแกรมประเภทนี้ จะมีส่วนที่คล้ายกับโปรแกรมประเภท .DLL คือเป็นโปรแกรมที่ไม่สามารถทำงานได้ด้วยตัวเอง แต่จะถูกเรียกใช้จากโปรแกรม .EXE อื่นๆ ส่วนที่แตกต่างกันระหว่างโปรแกรม .VBX และ .DLL ก็คือ โปรแกรม .VBX จะสามารถเรียกใช้ได้ทั้งในเวลาประมวลผล (run time) และเวลาที่กำลังออกแบบโปรแกรม (design time) โดยโปรแกรม .VBX จะมีลักษณะคล้ายโปรแกรมเชิงวัตถุ ที่มีคุณสมบัติ (property) และวิธีใช้งาน (method) อยู่ในตัวเอง ผู้ใช้สามารถทำการเปลี่ยนแปลงค่าในคุณสมบัติ และเลือกวิธีใช้งานต่อวัตถุในโปรแกรม .VBX ได้ตามต้องการ โปรแกรม

.VBX นี้สามารถเรียกใช้ได้จากตัวแปลภาษาหลายตัว เช่น Visual C++ และ Visual Basic เป็นต้น ซึ่งต่อมาโปรแกรม .VBX นี้ก็ได้รับการพัฒนาให้สามารถรองรับคุณสมบัติของวินโดวส์ในเรื่อง OLE ดียิ่งขึ้น จึงได้เปลี่ยนชื่อเป็นโปรแกรม .OCX (OLE Custom Control)

ความเป็นมาของวิชวลเบสิก

ภาษาเบสิกจัดเป็นภาษาคอมพิวเตอร์ที่เก่าแก่ที่สุดภาษาหนึ่ง ในช่วงแรกเป็นภาษาที่มีตัวแปลภาษาแบบทีละบรรทัดเรียกว่า อินเตอร์พรีเตอร์ (interpreter) ซึ่งทำงานช้ามาก และมีข้อจำกัดในการพัฒนาโปรแกรม จึงมีการพัฒนาตัวแปลภาษาแบบคอมไพเลอร์ (compiler) เหมือนภาษาคอมพิวเตอร์อีกหลายภาษาในขณะนั้น เช่น FORTRAN, Pascal ซึ่งจะแปลงโปรแกรมให้เป็นแฟ้มประเภท EXE และสามารถดำเนินงาน (run) โปรแกรมเพื่อใช้งานได้ทันที

ตัวแปลภาษาเบสิกสำหรับไมโครคอมพิวเตอร์ตัวแรกเกิดขึ้นเมื่อปี ค.ศ. 1975 พัฒนาโดยบริษัทไมโครซอฟต์ ใช้กับเครื่องคอมพิวเตอร์ Atari และ Apple เป็นตัวแปลภาษาแบบอินเตอร์พรีเตอร์ มีหมายเลขบรรทัดกำหนดลำดับการทำงาน มีการใช้คำสั่ง Goto และ Gosub ทำให้ยากต่อการตรวจหาจุดผิดพลาด และทำความเข้าใจการทำงานของโปรแกรม ต่อมาบริษัทบอร์แลนด์อินเตอร์เนชันแนลได้ออกแบบตัวแปลภาษาเบสิกเมื่อต้นปี ค.ศ.1987 ชื่อว่า Turbo BASIC เป็นตัวแปลภาษาแบบคอมไพเลอร์ ไม่มีหมายเลขบรรทัด ลักษณะของภาษาโปรแกรมจะเป็นภาษาโครงสร้าง มีการแยกการทำงานเป็นมอดุล ทำให้สามารถไล่ลำดับการทำงานและพัฒนาโปรแกรมได้ง่ายขึ้น

หลังจากนั้นบริษัทผลิตซอฟต์แวร์ต่างๆ ก็พัฒนาภาษาเบสิกตามแนวทางของ Turbo BASIC และเพิ่มความสามารถในการจัดการด้านฮาร์ดแวร์ เช่น การจัดสรรหน่วยความจำ มีการเพิ่มฟังก์ชันต่างๆ เพื่อใช้งานร่วมกับซอฟต์แวร์ตัวอื่นๆ การพัฒนาเพื่อใช้งานร่วมกับวินโดวส์ การพัฒนาการตรวจหาและแก้ไขจุดผิดพลาดในโปรแกรม (debugging) การเชื่อมโยงข้อมูลกับซอฟต์แวร์อื่นๆ และการเรียกใช้โปรแกรมย่อยซึ่งเขียนด้วยภาษาแอสเซมบลี (Assembly) เป็นต้น

จนถึงปัจจุบันบริษัทผลิตซอฟต์แวร์ได้ออกตัวแปลภาษาเบสิกหลายตัว แต่ละตัวก็มีหลายเวอร์ชัน ซึ่งมีจุดเด่นแตกต่างกัน เหมาะกับประเภทของงานไม่เหมือนกัน และเหมาะกับผู้ใช้งานแต่ละระดับ กล่าวคือ

1. GW-BASIC ทำงานบนดอส เหมาะสำหรับผู้เริ่มต้นเรียนรู้ภาษาเบสิก

2. Turbo BASIC ทำงานได้ทั้งบนดอสและบนวินโดวส์ นิยมใช้กับงานโปรแกรมด้านกราฟิก

3. QBASIC ทำงานบนดอส มีมาพร้อมกับเอ็มเอส-ดอสรุ่น 5.0 เป็นต้นมา นิยมใช้กับงานโปรแกรมทั่วไป

4. Quick BASIC ทำงานบนดอส มีประสิทธิภาพสูง พัฒนามาจาก QBASIC นิยมมากในกลุ่มผู้ใช้คอมพิวเตอร์ทั่วไป

5. Visual BASIC for Windows ทำงานบนวินโดวส์ นิยมใช้สร้างโปรแกรมประยุกต์ต่างๆ บนวินโดวส์

6. BASIC PDS (BASIC Profession Development System) เป็นภาษาเบสิกสำหรับผู้เขียนโปรแกรมภาษาเบสิกมืออาชีพ

เนื่องจากงานวิจัยนี้จะใช้ไมโครซอฟต์วิชวลเบสิกเป็นหลัก จึงจะขอกล่าวในรายละเอียดเฉพาะไมโครซอฟต์วิชวลเบสิกนี้เท่านั้น โดยไมโครซอฟต์ได้ออกวิชวลเบสิกทั้งรุ่นที่ทำงานบนดอส และวินโดวส์ และงานวิจัยนี้ก็ได้พัฒนาขึ้นด้วยวิชวลเบสิกสำหรับวินโดวส์

จุดประสงค์หลักของการพัฒนาวิชวลเบสิกของไมโครซอฟต์ก็คือการสร้างและพัฒนาโปรแกรมประยุกต์บนวินโดวส์โดยเฉพาะ วิชวลเบสิกใช้ประโยชน์จากคุณสมบัติและสภาพแวดล้อมของวินโดวส์ในการออกแบบแอพพลิเคชัน ทำให้การเขียนโปรแกรมด้วยวิชวลเบสิกสั้นกระทัดรัดเมื่อเทียบกับการเขียนโปรแกรมด้วยภาษาอื่น เช่น Turbo Pascal, C++ หรือ Small Talk ซึ่งต้องเขียนโปรแกรมยาวกว่าเพื่อให้ได้รูปลักษณ์ของหน้าต่าง (window) แบบเดียวกัน

วิชวลเบสิกที่นิยมใช้ในปัจจุบันคือ วิชวลเบสิก 3.0 ซึ่งพัฒนามาจากวิชวลเบสิก 2.0 วิชวลเบสิก 3.0 มีคุณลักษณะเด่นเพิ่มเติมจากวิชวลเบสิก 2.0 หลายประการ ดังจะสรุปคุณลักษณะทั่วๆ ไปและที่น่าสนใจของวิชวลเบสิกทั้งรุ่น 2.0 และ 3.0 ดังนี้

1. ขนาดของโปรแกรมไม่ใหญ่จนเกินไป กล่าวคือ วิชวลเบสิก 2.0 ใช้เนื้อที่จานบันทึกแบบแข็งประมาณ 4.8 MB ซึ่งจะรวมตัวอย่างของแอพพลิเคชันและตัวอย่างไอคอนแบบต่างๆ ส่วนวิชวลเบสิก 3.0 ใช้เนื้อที่ฮาร์ดดิสก์ประมาณ 24 MB เพราะเพิ่มชุดคำสั่งเกี่ยวกับการจัดการฐานข้อมูล วินโดวส์

เอพีไอ และตัวอย่างแฟ้มรูปภาพแบบแผนที่บิต ซึ่งนำมาใช้ในฟอร์ม (form) ของวิชวลเบสิก

2. ใช้กับไมโครซอฟต์วินโดวส์ 3.0 ขึ้นไป

3. ออกแบบมาเพื่อใช้สร้างโปรแกรมประยุกต์บนวินโดวส์ และใช้ประโยชน์จากสภาพแวดล้อมในรูปแบบกราฟิกที่เรียกว่าจียูไอ (Graphic User Interface: GUI) โปรแกรมวิชวลเบสิกจึงสั้น เข้าใจง่าย และไม่ต้องใช้งานร่วมกับซอฟต์แวร์ทูล (software tool) ที่ใช้ในการพัฒนาโปรแกรมประยุกต์บนวินโดวส์ เช่น Software Development Kit (SDK) ของไมโครซอฟต์ C++ หรือ Turbo Vision

4. มีระบบ Online help และตัวอย่างโปรแกรมแบบ Card file เพื่อสอนวิธีการใช้โปรแกรม ซึ่งจะเริ่มจากตัวอย่างโปรแกรมประยุกต์แบบง่ายๆ จนถึงตัวอย่างโปรแกรมประยุกต์ที่ซับซ้อน

5. สามารถเรียกใช้วินโดวส์เอพีไอซึ่งเป็นหัวใจการทำงานของวินโดวส์ และทรัพยากรต่างๆ ของวินโดวส์ เช่น สัญรูป (icon) ภาพแบบแผนที่บิต ภาพแบบวินโดวส์เมตาไฟล์ (Windows Meta file) และการรับส่งข้อมูลจากคลิปบอร์ดของวินโดวส์

6. ตัวแปลภาษาเป็นแบบคอมไพเลอร์ ทำให้สามารถเรียกใช้โปรแกรมย่อยร่วมกับโปรแกรมประยุกต์อื่นๆ ขณะทำงานได้ และในการแปลชุดคำสั่งจะไม่รวมฟังก์ชันของวิชวลเบสิกเข้าไปในโปรแกรมหลัก แต่แยกเก็บไว้ใน Runtime Libraly ที่เป็นแฟ้ม .DLL ทำให้ไม่ใช้เนื้อที่ในหน่วยความจำมากนัก

7. คำสั่งและฟังก์ชันการใช้งานคล้ายคลึงกับ Quick BASIC แต่ออกแบบให้ใช้งานได้ง่ายกว่า และเพิ่มคำสั่งเพื่อใช้งานกับวินโดวส์

8. โปรแกรมที่เขียนด้วยวิชวลเบสิก สามารถใช้ระบบแลกเปลี่ยนหรือเชื่อมโยงข้อมูลกับโปรแกรมประยุกต์ต่างๆ ของวินโดวส์ ในลักษณะที่เรียกว่าดีดีอี (Dynamic Data Exchange : DDE) และโอแอลอี (Object Linking and Embedding : OLE)

9. สามารถกำหนดให้โปรแกรมทำงานทีละส่วน หรือกำหนดจุดหยุดในโปรแกรม เพื่อตรวจหาและแก้ไขจุดผิดพลาดในโปรแกรม รวมทั้งมีเครื่องมือแก้จุดบกพร่อง (debuging tools) ช่วยอำนวยความสะดวกในการทำงานดังกล่าว

10. มีซอฟต์แวร์เพิ่มขีดความสามารถของวิชวลเบสิกเพิ่มหลายตัว ได้แก่ Control Development Kit, Button Tool, Custom Control Factory และ Quick Pak Professional เป็นต้น

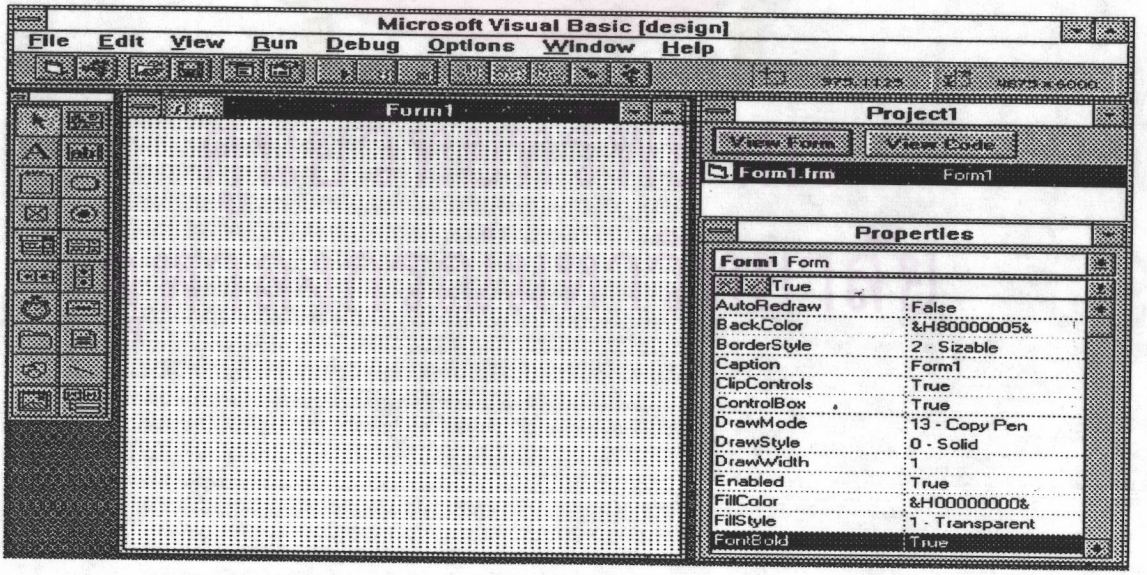
คุณสมบัติและความสามารถของวิชวลเบสิกที่กล่าวมา เป็นคุณสมบัติร่วมของทั้งสองรุ่น โดยที่วิชวลเบสิก 3.0 ได้มีการเพิ่มเติมศักยภาพในด้านอื่นๆ อีกหลายด้าน กล่าวคือ

1. มีชุดคำสั่งเกี่ยวกับการจัดการฐานข้อมูลของ Access, FoxPro, dBase, Paradox, Btrive, ODBC (Open Database Connectivity) และเข้าถึงฐานข้อมูลด้วยคอนโทรล "Data" ไม่ใช่การเขียนรหัส

2. พัฒนากระบวนการโอแอลอี ให้เป็นมาตรฐาน และใช้งานคล่องตัวขึ้น

3. เพิ่มความสามารถเกี่ยวกับการสร้างเมนูคำสั่ง การสร้างคอนโทรล ได้แก่ การสร้างป็อปอัพเมนู การเพิ่มเหตุการณ์ของคอนโทรล และการสร้าง Hierarchical List Box

รูปแบบจอภาพของวิชวลเบสิก รุ่น 3.0 ได้แสดงดังรูปที่ 3.4



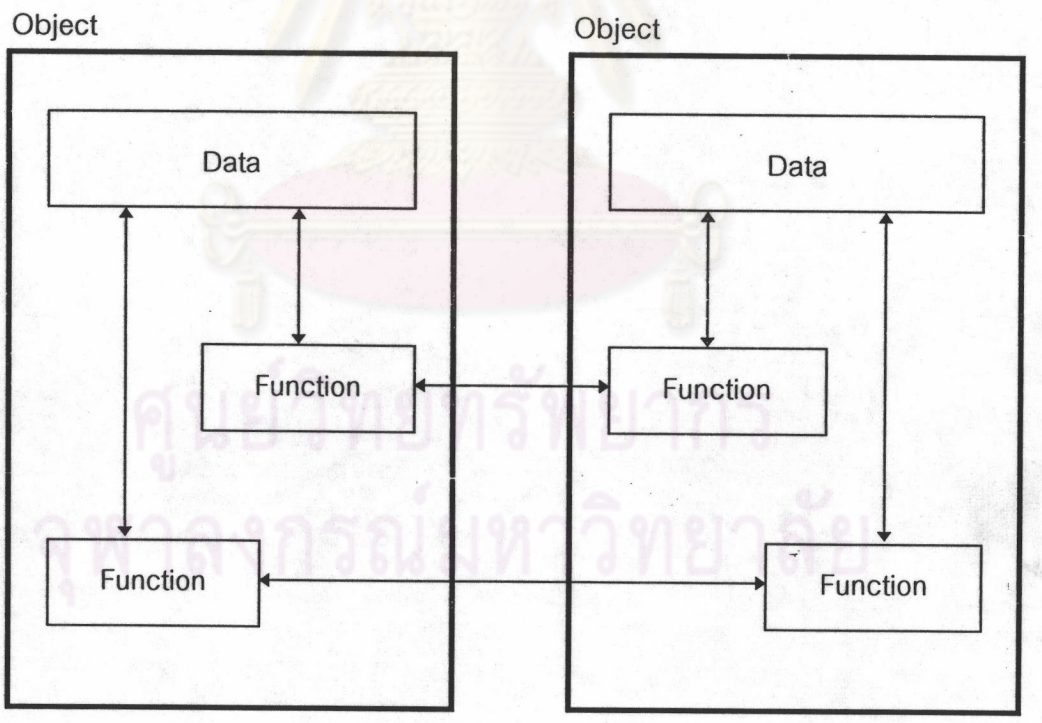
รูปที่ 3.4 แสดงจอภาพของวิชวลเบสิก 3.0

โครงสร้างโปรแกรมของวิชวลเบสิก

วิชวลเบสิกถูกออกแบบให้ง่ายและสะดวกต่อการสร้างโปรแกรมประยุกต์บนวินโดวส์ ในเวลาอันสั้น โครงสร้างของวิชวลเบสิกจึงเกิดจากการผสมผสานของแนวคิดในการพัฒนาโปรแกรมหลายๆ รูปแบบเข้าด้วยกัน อาจกล่าวได้ว่าเป็นการประยุกต์เอาจุดเด่นของแต่ละแนวคิดมาใช้ประโยชน์ก็ได้ กล่าวคือ

1. แนวคิดการโปรแกรมเชิงวัตถุ (Object Oriented Programming : OOP)

ศรัณย์ อินทโกสุม (2536) ได้สรุปแนวคิดการโปรแกรมเชิงวัตถุไว้คือ การเขียนโปรแกรมโดยใช้หลักการของการโปรแกรมเชิงวัตถุจะช่วยแก้ปัญหาของการเขียนโปรแกรมแบบโครงสร้าง โดยหลักการที่สำคัญของการโปรแกรมเชิงวัตถุก็คือการผนึกข้อมูลและฟังก์ชันเข้าไว้ด้วยกันให้กลายเป็นวัตถุ ซึ่งการผนึกข้อมูล และฟังก์ชันเข้าด้วยกันนี้เรียกว่า Encapsulation โดยข้อมูลที่อยู่ในวัตถุจะถูกเรียกใช้จากฟังก์ชันที่อยู่ในวัตถุเท่านั้น ถ้าฟังก์ชันที่อยู่ในวัตถุหนึ่งต้องการจะใช้ข้อมูลที่อยู่ในอีกวัตถุหนึ่ง จะต้องเรียกผ่านฟังก์ชันที่อยู่ในวัตถุนั้นเท่านั้น ดังรูปที่ 3.5



รูปที่ 3.5 แสดงการเรียกใช้ข้อมูลระหว่างวัตถุ 2 วัตถุ

จะเห็นได้ว่า หลักการของการโปรแกรมเชิงวัตถุจะช่วยแก้ปัญหาในการเขียนโปรแกรมแบบเป็นโครงสร้างเนื่องจากข้อมูลจะถูกเปลี่ยนแปลงโดยฟังก์ชันที่อยู่ในวัตถุเท่านั้น และเมื่อมีการแก้ไขโครงสร้างข้อมูลใหม่ ผู้พัฒนาโปรแกรมก็เพียงแต่แก้ไขเฉพาะฟังก์ชันที่เกี่ยวข้องที่อยู่ในวัตถุนั้นเท่านั้น

วิชวลเบสิกได้นำแนวคิดบางส่วนของการโปรแกรมเชิงวัตถุมาใช้คือ วิชวลเบสิกได้เสนอให้ผู้ใช้มองการพัฒนาโปรแกรมในรูปของการนำวัตถุชนิดต่างๆ มาประกอบเข้าด้วยกัน ได้แก่ ฟอร์ม (form) และคอนโทรล (control) ต่างๆ เช่น ปุ่มคำสั่ง (command button) กล่องรับข้อความ (text box) กล่องรูปภาพ (picture box) เป็นต้น โดยแต่ละคอนโทรลจะมีคุณสมบัติ (property) และวิธีการใช้งาน (method) ประจำของแต่ละคอนโทรล เมื่อเปรียบกับแนวคิดของโปรแกรมเชิงวัตถุจะได้ว่า คุณสมบัติก็คือข้อมูลที่ฝังตัวอยู่ในวัตถุแต่ละชิ้น และวิธีการใช้งานก็คือฟังก์ชันของแต่ละวัตถุ

2. แนวคิดการโปรแกรมเชิงเหตุการณ์ (Event-Driven Programming)

เป็นแนวคิดที่มองการพัฒนาโปรแกรมโดยการจัดการกับเหตุการณ์ต่างๆ ที่อาจเกิดกับวัตถุใดๆ ในโปรแกรม แนวคิดนี้เป็นแนวคิดหนึ่งที่ประยุกต์มาจากการโปรแกรมเชิงวัตถุ โดยจะมองเหตุการณ์ต่างๆ ที่จะเกิดกับวัตถุเป็นหลัก เมื่อมีเหตุการณ์ใดเกิดขึ้น ก็จะส่งการทำงานไปที่โปรแกรมย่อยที่จัดการกับเหตุการณ์นั้นๆ โดยเฉพาะ เหตุการณ์ที่จะเกิดกับวัตถุ ส่วนหนึ่งจะเป็นการดักจับสัญญาณที่เกิดจากอุปกรณ์รับข้อมูล เช่น แผงแป้นอักขระ หรือเมาส์ เป็นต้น เหตุการณ์ส่วนอื่นๆ ก็อาจจะเกิดก่อนหรือหลังจากการทำงานใดๆ ที่มีผลต่อวัตถุนั้นๆ เช่น การเรียกใช้ฟอร์ม (load form) หรือเลิกการใช้ฟอร์ม (unload form) เป็นต้น

วิชวลเบสิกได้เตรียมโปรแกรมย่อยที่คอยดักจับเหตุการณ์ต่างๆ ของแต่ละคอนโทรลเป็นจำนวนมาก ผู้ใช้ต้องการดักจับเหตุการณ์ใดหรือต้องการให้เกิดการทำงานเมื่อเหตุการณ์ใดเกิดขึ้นก็สามารถเลือกระบุการทำงานให้กับเหตุการณ์ของวัตถุนั้นๆ ได้ เหตุการณ์ที่วิชวลเบสิกเตรียมไว้ให้ เช่น เหตุการณ์ที่ดักจับการกดปุ่มที่แผงแป้นอักขระ (Keypress or Keydown event) เหตุการณ์ที่เกิดเมื่อมีการกดปุ่มที่เมาส์ (Mouse click or Mouse down event) เป็นต้น

3. แนวคิดการโปรแกรมเชิงรูปภาพ (Visual Programming)

เป็นแนวคิดอีกแนวหนึ่งที่สนับสนุนการโปรแกรมเชิงวัตถุ โดยเน้นให้เห็นภาพในขณะออกแบบโดยตรงทางจอภาพ คือยินยอมให้ผู้ใช้สามารถนำวัตถุต่างๆ มาประกอบกัน โดยเลือกวัตถุและนำมาไว้ในตำแหน่งที่ต้องการ ทั้งสามารถกำหนดคุณสมบัติต่างๆ ของวัตถุได้ เช่น ขนาดของตัวอักษรที่แสดง สีและขนาดของวัตถุเป็นต้น

วิชวลเบสิกได้เตรียมสภาพแวดล้อมสำหรับการโปรแกรมในแนวคิดนี้ไว้โดยตรง และอาจถือเป็นแนวทางพัฒนาโปรแกรมหลักของวิชวลเบสิกในการออกแบบจอภาพ ผู้ใช้สามารถเลือกวัตถุที่ต้องการจาก Toolbox แล้วนำมาวางบนฟอร์ม เพื่อทำการออกแบบจนได้จอภาพที่ต้องการ และสามารถปรับเปลี่ยนแก้ไขได้โดยสะดวก

4. แนวคิดการโปรแกรมเชิงโครงสร้าง (Structure Programming)

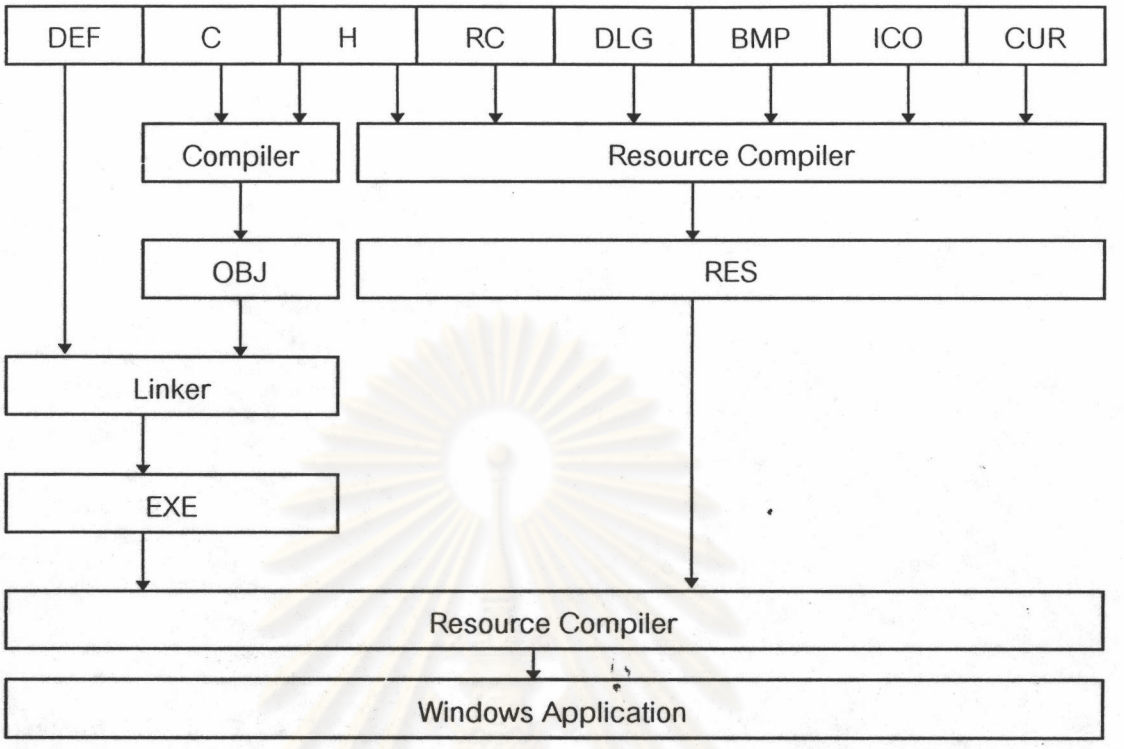
เป็นแนวคิดของการโปรแกรมแบบเดิม ซึ่งมีข้อดีในเรื่องของการควบคุมทิศทางทำงานของโปรแกรม และเหมาะกับการพัฒนาที่แบ่งโปรแกรมออกเป็นมอดุลย่อยๆ เพื่อสะดวกต่อการควบคุมและแก้ไขความผิดพลาด ที่อาจเกิดจากการประมวลผลของโปรแกรม

ดังที่ได้กล่าวแล้ว วิชวลเบสิกได้รับการพัฒนาต่อมาจาก Quick BASIC ของไมโครซอฟต์ จึงเป็นธรรมดาที่วิชวลเบสิกจะยังคงความสามารถในการโปรแกรมแบบโครงสร้างของ Quick BASIC มาด้วย และได้กลายเป็นส่วนสำคัญที่ทำให้วิชวลเบสิก สามารถทำงานได้อย่างมีประสิทธิภาพ กล่าวคือ อาจเรียกได้ว่าเป็นกลไกที่สำคัญในการประมวลผลของโปรแกรมที่พัฒนาโดยวิชวลเบสิก ผู้ใช้สามารถเขียนรหัสของโปรแกรมโครงสร้างเหล่านี้ไว้ในเหตุการณ์ต่างๆ ที่ต้องการตรวจจับในวัตถุ และเขียนเป็นฟังก์ชันหรือโปรแกรมย่อยไว้ในส่วนของหน้าต่างมอดุล (Module Window) ของวิชวลเบสิก เหมือนการโปรแกรมใน Quick BASIC

<u>องค์ประกอบของแฟ้มข้อมูลที่เกี่ยวข้อง</u>

การทำงานของวิชวลเบสิก ที่ประสานกับสภาพแวดล้อมของไมโครซอฟต์วินโดวส์ ก็ไม่ต่างอะไรกับโปรแกรมที่พัฒนาด้วยภาษาอื่นๆ เช่น ภาษา C++ ส่วนประกอบและขั้นตอนการสร้างโปรแกรมด้วยภาษา C++ ได้แสดงดังรูปที่ 3.6

แต่วิชวลเบสิกได้ลดปัญหาและความซับซ้อนของการพัฒนาโปรแกรมลงได้มาก โดยวิชวลเบสิกจะทำการจัดการในส่วนประกอบและขั้นตอนเหล่านั้นเอง ผู้ใช้เพียงแต่ทำการออกแบบและเขียนโปรแกรมในสภาพแวดล้อมและใช้เครื่องมือที่วิชวลเบสิกเตรียมไว้ให้เท่านั้น ส่วนรายละเอียดอื่นๆ วิชวลเบสิกจะทำการบันทึกลงในแฟ้มประเภทต่างๆ จำนวนหนึ่ง ซึ่งมีส่วนประกอบคล้ายกับที่สร้างด้วยภาษา C++ ข้างต้น โดยแฟ้มดังกล่าวสามารถจำแนกได้ดังนี้



รูปที่ 3.6 แสดงส่วนประกอบและขั้นตอนการสร้างโปรแกรมด้วยภาษา C++

1. Project Make File (.MAK)

เป็นแฟ้มที่เก็บรายละเอียดของแฟ้ม .FRM, .BAS, .VBX และรายละเอียดในการแปลชุดคำสั่งเพื่อสร้างแฟ้ม .EXE ตัวอย่างของแฟ้ม .MAK ดูได้จากรูปที่ 3.7

```
EXAMPLE.BAS
C:\WINDOWS\EXAMPLE1.VBX
C:\WINDOWS\EXAMPLE2.VBX
EXAMPLE1.FRM
EXAMPLE2.FRM
EXAMPLE3.FRM
EXAMPLE4.FRM
ProjWinSize=71,417,211,359
ProjWinShow=2
IconForm="example1"
Title="EXAMPLE"
ExeName="EXAMPLE.EXE"
```

รูปที่ 3.7 แสดงตัวอย่างแฟ้มข้อมูลประเภท .MAK

2. Basic Source Code File (.BAS)

เป็นแฟ้มที่เก็บรายละเอียดของฟังก์ชัน โปรแกรมย่อย ค่าคงที่ ตัวแปรส่วนกลาง (global variable) และการกำหนดชื่อฟังก์ชันหรือโปรแกรมย่อยที่จะเรียกใช้ผ่านเอพีไอของวินโดวส์ ตัวอย่างของแฟ้ม .BAS ได้แสดงในรูปที่ 3.8

```
'****** EXAMPLE.BAS *********
Global variable1 As Integer          'Decalar Integer global variable
Global variable2 As Single           'Decalar Single global variable
Global variable3 As Double           'Decalar Double global variable

Dim array1(10) As integer            'Decalar Integer global array ,
Dim array2(10) As String*10          'Decalar String global array

Sub prog1(parameter1,parameter2)

      ..... (Sub program body)

End Sub
```

รูปที่ 3.8 แสดงตัวอย่างแฟ้มข้อมูลประเภท .BAS

3. Form Description File (.FRM)

เป็นแฟ้มที่เก็บรายละเอียดของวัตถุหรือคอนโทรลต่างๆ ที่ใช้ในฟอร์ม รายละเอียดของเมนู โค้ดของเหตุการณ์ต่างๆ ฟังก์ชัน และโปรแกรมย่อยที่ผู้ใช้กำหนด ตัวอย่างของแฟ้ม .FRM ได้แสดงในรูปที่ 3.9

4. Resource Data file (.FRX)

เป็นแฟ้มที่เก็บรายละเอียดของทรัพยากรต่างๆ ของวินโดวส์ ที่ถูกเรียกใช้จากโปแกรมของวิชวลเบสิก เช่น สัญรูป และภาพแผนที่บิต เป็นต้น รายละเอียดเหล่านี้ปกติจะเก็บอยู่ในแฟ้ม .FRM แต่ในกรณีที่แฟ้ม .FRM ถูกระบุให้บันทึกเป็นข้อมูลแบบตัวอักษร (text file) ข้อมูลเหล่านี้จะถูกเก็บลงในแฟ้ม .FRX โดยใช้ชื่อเดียวกับแฟ้ม .FRM เดิม

นอกจากแฟ้มเหล่านี้แล้ว วิชวลเบสิกยังยินยอมให้ผู้ใช้ตั้งชื่อประเภทแฟ้มตามที่ต้องการได้ โดยชื่อเหล่านั้นจะต้องถูกบันทึกไว้ในแฟ้ม .MAK ด้วย วิชวลเบสิกจึงจะนำมาใช้งานได้ แฟ้มประเภทอื่นๆ

อื่นๆ เช่น แฟ้ม .TXT หรือ .GBL ส่วนใหญ่จะไว้เก็บพวกค่าคงที่ ตัวแปรส่วนกลาง หรือการกำหนดชื่อฟังก์ชันหรือโปรแกรมย่อย ที่จะเรียกจากโปรแกรม .DLL อื่นๆ

```
VERSION 2.00
Begin MDIForm example
   Caption         =   "EXAMPLE"
   ClientHeight    =   4185
   ClientLeft      =   1140
   ClientTop       =   1770
   ClientWidth     =   7590
   Height          =   4875
   Icon            =   EXAMPLE.FRX:0000
   Left            =   1080
   LinkTopic       =   "Form1"
   Top             =   1140
   Width           =   7710
   WindowState     =   2 'Maximized
   Begin Menu Menu
      Caption =   "&Menu1"
      Index   =   0
      Begin Menu Menu1
         Caption =   "&SubMenu1"
         Enabled =   0 'False
         Index   =   0
      End
      Begin Menu Menu1
         Caption =   "&SubMenu2"
         Enabled =   0 'False
         Index   =   1
      End
End
```

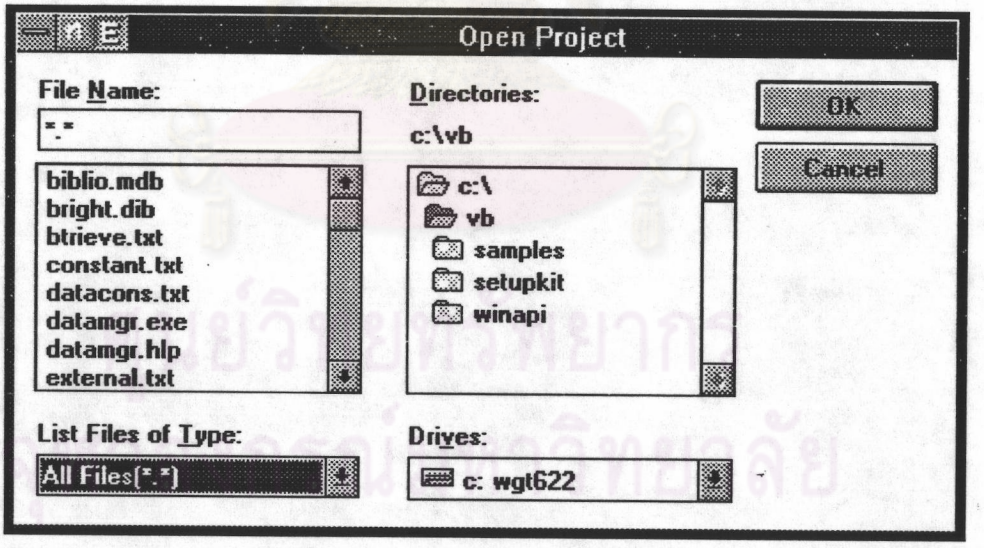
รูปที่ 3.9 แสดงตัวอย่างแฟ้มข้อมูลประเภท .FRM

แนวทางสร้างโปรแกรมในวิชวลเบสิก

ดังที่ได้กล่าวไว้แล้ว วิชวลเบสิก ตามความหมายคือภาษาเบสิกที่มองเห็นได้ ในที่นี้จะหมายถึงภาษาเบสิกที่สร้างโปรแกรมประยุกต์โดยผู้ใช้สามารถเห็นหน้าตาของโปรแกรมประยุกต์ในขณะทำการสร้างได้ การสร้างโปรแกรมประยุกต์จะใช้วิธีวาดวัตถุ ซึ่งโดยทั่วไปจะเรียกว่าคอนโทรล ลงบนหน้าต่างที่เรียกว่าฟอร์ม คอนโทรลเหล่านี้จะมีหลายแบบ จะปรากฏบน Toolbox เช่น Label, Text box, Command button, Timer หรือ Image เป็นต้น คอนโทรลจะมีคุณสมบัติเฉพาะตัวที่สามารถกำหนดได้ และสามารถเขียนรหัสโปรแกรมย่อยของเหตุการณ์ (event procedure) หรือโปรแกรมย่อยทั่วไป เพื่อทำคำสั่งต่างๆ บนคอนโทรลนั้นได้ด้วย

ขั้นตอนการสร้างโปรแกรมประยุกต์บนวินโดวส์ด้วยวิชวลเบสิก สามารถอธิบายโดยย่อๆ เพื่อให้เห็นหลักการของการเขียนและหน้าตาของโปรแกรมวิชวลเบสิกได้ดังนี้

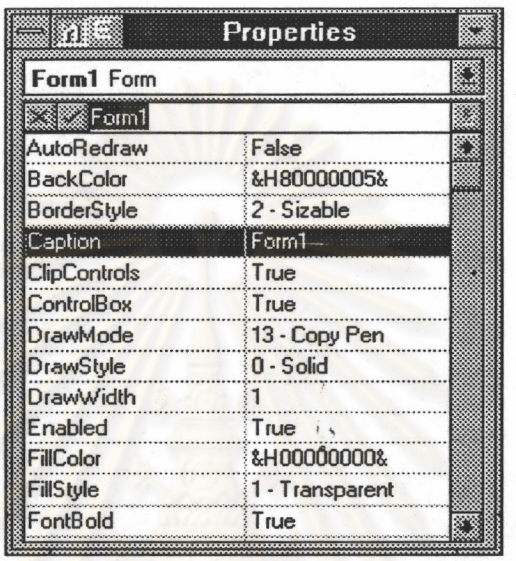
1. สร้างหน้าต่างติดต่อกับผู้ใช้งาน (user interface window) โดยใช้หน้าต่างฟอร์ม การสร้างหน้าต่างติดต่อผู้ใช้งาน เริ่มต้นด้วยการเปิดหน้าต่างฟอร์ม จากนั้นเลือกคอนโทรลที่ต้องการจาก Toolbox มาวางตำแหน่งลงบนหน้าต่างฟอร์ม ตัวอย่างของหน้าต่างติดต่อผู้ใช้งาน แสดงดังรูปที่ 3.10



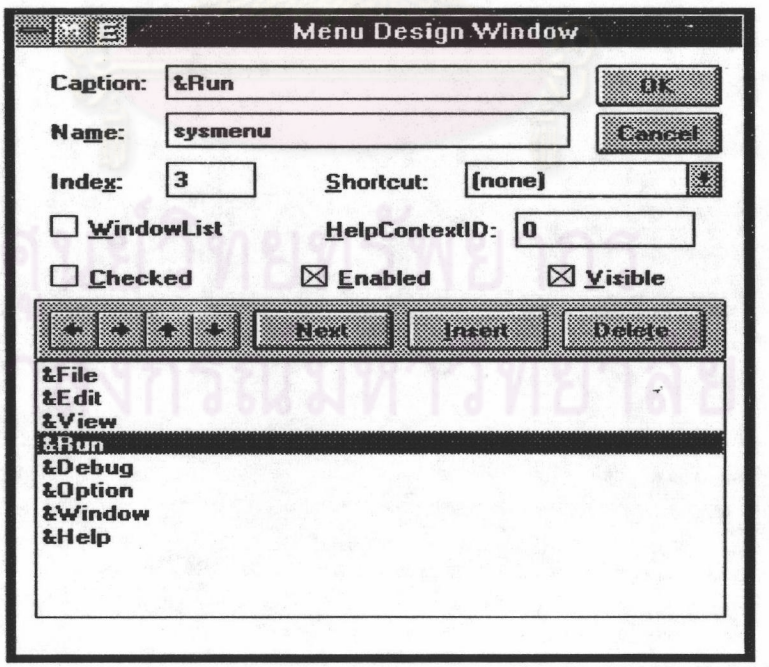
รูปที่ 3.10 แสดงตัวอย่างหน้าต่างติดต่อกับผู้ใช้

2. กำหนดคุณสมบัติของฟอร์ม และคอนโทรลต่างๆ รวมทั้งสร้างเมนูคำสั่งของโปรแกรมประยุกต์ โดยใช้หน้าต่างคุณสมบัติ (properties window) และหน้าต่างออกแบบเมนู (menu design

window) การกำหนดคุณสมบัติของฟอร์มและคอนโทรลต่างๆ ทำได้โดยการกำหนดคุณสมบัติต่างๆ ที่ปรากฏในหน้าต่างคุณสมบัติ ซึ่งมีอยู่หลายประการ นอกจากนี้ยังสามารถสร้างเมนูของฟอร์มและคอนโทรล โดยใช้หน้าต่างออกแบบเมนู ตัวอย่างของหน้าต่างคุณสมบัติ และหน้าต่างออกแบบเมนู ได้แสดงไว้ในรูปที่ 3.11 และ 3.12 ตามลำดับ

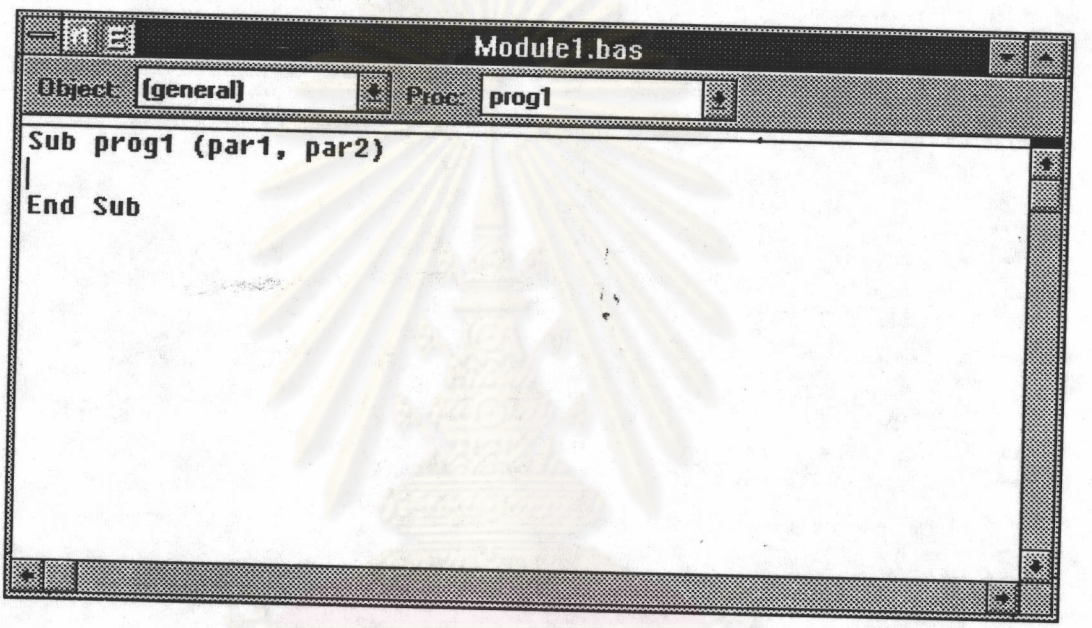


รูปที่ 3.11 แสดงตัวอย่างหน้าต่างคุณสมบัติ



รูปที่ 3.12 แสดงตัวอย่างหน้าต่างออกแบบเมนู

3. เขียนรหัสคำสั่งหรือเขียนโปรแกรมย่อยของฟอร์ม คอนโทรล และเมนูคำสั่งโดยใช้หน้าต่างรหัส (code window) การเขียนรหัสคำสั่งหรือโปรแกรมย่อยของฟอร์ม คอนโทรลแต่ละตัวและเมนูคำสั่งจะเขียนด้วยภาษาเบสิกที่มีลักษณะเป็นโปรแกรมแบบโครงสร้าง คือจะประอบด้วยโปรแกรมย่อยแสดงเหตุการณ์หลังการกระทำ (action) ต่างๆ โดยผู้ใช้ เช่น การกดปุ่มเมาส์ การกดปุ่มที่แผงแป้นอักขระ ชุดของเหตุการณ์เหล่านี้เรียกว่า Event Procedure สำหรับคอนโทรลแต่ละตัวจะมีเหตุการณ์ที่สามารถใช้ได้ไม่เหมือนกัน ผู้ใช้สามารถเลือกกำหนดหรือปรับเปลี่ยนตามความต้องการได้ ตัวอย่างหน้าต่างรหัส ได้แสดงไว้ในรูปที่ 3.13



รูปที่ 3.13 แสดงตัวอย่างหน้าต่างรหัส

4. การเรียกใช้ฟังก์ชันเอพีไอของวินโดวส์ ผู้ใช้จะสามารถเลือกดูรายการฟังก์ชันเอพีไอที่ต้องการได้จาก Online help เรื่อง "Win SDK Help" และ "Win API 3.1 Help"โดย Win SDK Help จะบอกรายละเอียดของการใช้งานฟังก์ชันเอพีไอทั้งหมดของวินโดวส์ ส่วน Win API 3.1 Help จะบอกถึงการประกาศ (declaration) ชื่อของฟังก์ชันเอพีไอ ในรูปแบบของรหัสภาษาเบสิก พร้อมกับค่าคงที่ต่างๆ ที่ต้องใช้ประกอบกับฟังก์ชันเอพีไอเหล่านั้น การเรียกใช้ฟังก์ชันเอพีไอ ผู้ใช้จึงต้องทำการประกาศชื่อฟังก์ชันดังกล่าวในส่วนที่เป็น declaration ของฟอร์ม หรือมอดุลของวิชวลเบสิก จากนั้นก็สามารถเรียกใช้ได้เหมือนฟังก์ชันปกติของเบสิก กล่าวคือต้องส่งพารามิเตอร์ (parameter) ให้ถูกต้องตามที่แต่ละฟังก์ชันต้องการด้วย ตัวอย่างการประกาศชื่อฟังก์ชันเอพีไอ และการใช้งาน แสดงดังรูปที่ 3.14

```
'******* Example for drwing polygon by useing Polygon function in API of Windows

Type POINTAPI
        x As Integer
        y As Integer
End Type

Declare Function Polygon Lib "GDI" (ByVal hDC As Integer, lpPoints As POINTAPI, ByVal nCount As
    Integer) As Integer

Sub drawpolygon (picture As Control,pointdata$)
        ReDim pxy(10) As POINTAPI
        For I% = 0 To 9
                pxy(i%).x = val(Mid$(pointdata$, i% * 4 + 1, 2))
                pxy(i%).y = val(Mid$(pointdata$, i% * 4 + 3, 2))
        Next
        hm% = Polygon(ct.hDC, pxy(0), 10)
End Sub
```

รูปที่ 3.14 แสดงตัวอย่างการประกาศชื่อฟังก์ชัน API และการใช้งาน