บทที่ 2

ทฤษฎีและแนวคิดที่นำมาใช้ในการวิจัย

ในการสร้างภาพเคลื่อนไหว (Animation) ไม่ว่าจะเป็นการ์ตูนหรือภาพยนตร์ก็ตาม มีหลักการเดียวกันคือการนำภาพที่บันทึกไว้มาแสดงแต่ละภาพอย่างต่อเนื่องกัน โดยมีความถี่ในการแสดงภาพหรือช่วงเวลาในการแสดงแต่ละภาพที่เหมาะสมซึ่งทำให้ประสาทสัมผัสทางตาของมนุษย์มองเห็นว่าเกิดการเคลื่อนไหว

เราสามารถทดสอบหลักการเบื้องต้นของการทำภาพเคลื่อนไหวได้โดย เตรียมกระดาษสัก 10 แผ่น วาดภาพลูกบอลลงในกระดาษแผ่นละลูกโดยกระดาษแผ่นแรกให้วาดลูกบอลอยู่ด้านซ้ายสุดของกระดาษ แผ่นที่สองให้วาดลูกบอลเยื้องมาทางขวาเล็กน้อย แผ่นต่อๆมาให้ทำเช่นเดียวกันคือวาดลูกบอลให้เยื้องไปทางขวาของแผ่นที่แล้ว ทำเช่นนี้จนครบทั้ง 10 แผ่น จากนั้นลองพลิกกระดาษทั้ง 10 แผ่นดูจะเห็นว่าลูกบอลในกระดาษนั้นดูคล้ายกับกำลังเคลื่อนไปทางขวาของกระดาษ ถ้าเราพลิกกระดาษเร็วขึ้นจะเห็นลูกบอลเคลื่อนที่ได้ราบเรียบ ถ้าพลิกช้าจะเห็นลูกบอลเคลื่อนที่เหมือนสะดุดหรือคล้ายเกิดการกระพริบ

จากวิธีการดังกล่าวนี้ เป็นหลักการเบื้องต้นของการทำภาพเคลื่อนไหว นั่นคือการบันทึกภาพและการนำมาแสดงใหม่ การทำการ์ตูนหรือภาพยนตร์ก็เช่นกัน จะทำการบันทึกภาพของตัวละครหรือวาดรูปของตัวการ์ตูนนั้นเก็บไว้ทีละเฟรม (Frame) เมื่อบันทึกเสร็จจึงนำมาเริ่มต้นแสดงใหม่ทีละเฟรมด้วยความถี่คงที่ที่เหมาะสมทำให้เกิดเป็นภาพยนตร์ ตัวละครหรือตัวการ์ตูนมีการเคลื่อนไหวได้เหมือนจริง อัตราความถี่ที่ใช้ในการแสดงภาพหรือการเล่นย้อนกลับ (Playback) นั้น ตามปกติสำหรับโทรทัศน์หรือวิดีโอเทป (VTR) ประมาณ 30 เฟรมต่อวินาที ในระบบ NTSC หรือ 25 เฟรมต่อวินาทีในระบบ PAL (Adam,Lee 1993) สำหรับการสร้างภาพเคลื่อนไหวด้วยคอมพิวเตอร์ส่วนบุคคลนั้นสามารถใช้สัญญานาฬิกาจากชิปบอกเวลาของระบบ (System Timer Chip) ได้

หลักการทำภาพเคลื่อนไหวบนคอมพิวเตอร์

จากหลักการเบื้องต้นในการทำภาพเคลื่อนไหวดังกล่าวสามารถนำมาประยุกต์ใช้บนคอมพิวเตอร์ได้ และสามารถแบ่งชนิดของการสร้างภาพเคลื่อนไหวบนคอมพิวเตอร์ได้เป็น 3 ชนิดหลักๆ (นเรศและธงชัย, Animation ศาสตร์แห่งจินตนาการ วารสารคอมพิวเตอร์รีวิว ฉบับที่ 98 ,2535) ได้แก่

1. BitBlt Animation
2. Frame Animation
3. Real-Time Animation

BitBlt Animation

หรือบางทีเรียกว่าวิธี Block Graphics Partial Screen Animation ซึ่งเป็นการทำภาพเคลื่อนไหวเฉพาะบางส่วนของจอภาพ หลักการของการทำภาพเคลื่อนไหวแบบนี้จะทำการย้ายกรอบข้อมูลบางส่วนของจอภาพในส่วนที่เราสนใจให้เคลื่อนไปตามจุดต่างๆที่กำหนด โดยย้ายข้อมูลเป็นกรอบสี่เหลี่ยมของภาพ การทำภาพเคลื่อนไหวแบบนี้จะมีความเร็วค่อนข้างสูงเนื่องจากส่วนที่คอมพิวเตอร์ต้องจัดการในการทำภาพเคลื่อนไหวนั้นเป็นการย้ายข้อมูลจำนวนไม่มากนัก แต่จะต้องทำการเก็บข้อมูลของภาพพื้นหลังที่ภาพเคลื่อนที่ผ่านไปด้วย

การทำ BitBlt Animation สามารถใช้กับจอภาพกราฟิกได้ทุกชนิดตั้งแต่จอภาพที่มีความละเอียดสูงสามารถสร้างสีได้หลายสี จนกระทั่งถึงจอภาพที่มีความละเอียดต่ำหรือจอภาพขาวดำ การใช้งาน BitBlt Animation นั้นส่วนใหญ่เป็นงานที่ต้องใช้ความเร็วสูง มีการเคลื่อนไหวเฉพาะวัตถุหรือสิ่งที่กำลังสนใจบางตัวเท่านั้น เช่น การจำลองการเคลื่อนไหวของวัตถุหรือ การจำลองการท่องไปตามปม (node) ในเรื่องของโครงสร้างข้อมูลแบบต้นไม้ เป็นต้น

Frame Animation

บางครั้งเรียกว่า การทำภาพเคลื่อนไหวแบบเต็มจอ หรือ Full Screen Animation หรือ Page Animation ต้องสร้างภาพที่ต้องการให้เคลื่อนไหวเก็บไว้ก่อนเป็นภาพๆหรือทีละเฟรม เช่นเดียวกับการถ่ายทำภาพยนตร์เมื่อต้องการทำให้ภาพเกิดการเคลื่อนไหวก็นำภาพนั้นขึ้นมาแสดงบนจอภาพทีละเฟรมโดยใช้ความเร็วสูงเหมือนกับการฉายภาพยนตร์ (ประมาณ 30 เฟรมต่อวินาที) Frame Animation นั้นเป็นทางเลือกที่ดีสำหรับการทำภาพเคลื่อนไหวคุณภาพสูง เนื่องจากเราสามารถสร้างภาพที่มีความ

สวยงาม ซับซ้อน โดยใช้เทคนิคการทำภาพแบบต่างๆ เช่นการสร้างภาพสามมิติให้มีแสงเงาเหมือนจริงบนคอมพิวเตอร์ทีละภาพ

Real-Time Animation

เป็นการสร้างภาพในขณะที่ทำการแสดงภาพเคลื่อนไหวไปพร้อมกัน ซึ่งจะต้องใช้เครื่องคอมพิวเตอร์ที่มีความเร็วค่อนข้างสูงมากจึงจะสามารถใช้งานได้อย่างดีบางครั้งเรียก Real-Time Animation ว่า Live Animation หรือ Dynamic Page-Flipping Animation การทำภาพเคลื่อนไหวแบบนี้จะต้องมีหน่วยความจำแสดงผลอย่างน้อย 2 ชุด โดยในขณะที่กำลังแสดงภาพในหน้าแรกก็จะทำการคำนวณและวาดภาพต่อไปในหน่วยความจำแสดงผลหน้าที่สอง เมื่อวาดเสร็จแล้วก็จะสลับหน้าที่สองขึ้นมาแสดงบนจอภาพแล้วทำการวาดภาพใหม่ลงในหน้าแรกสลับกันไปเรื่อยๆ ก็จะได้ภาพเคลื่อนไหวโดยไม่ต้องสร้างภาพเก็บไว้ล่วงหน้า

การประยุกต์ใช้งานการทำภาพเคลื่อนไหวแบบนี้ นิยมใช้กับระบบจำลองการทำงานแบบ 3 มิติ หรือการทำภาพเคลื่อนไหวแบบสามารถโต้ตอบกับผู้ใช้ เช่น ระบบ Flight Simulation เป็นต้น การสร้างภาพเคลื่อนไหวบนคอมพิวเตอร์นั้นยังมีสิ่งที่ต้องคำนึงถึงอีกอย่างหนึ่งคือ วิธีการควบคุมการเคลื่อนไหวของภาพ

การควบคุมภาพเคลื่อนไหว

การควบคุมการแสดงภาพเคลื่อนไหวเป็นวิธีการทำให้ภาพที่นำเสนอออกมานั้นเป็นไปตามที่เราต้องการหรือให้เป็นไปตามกฎเกณฑ์ใดๆที่กำหนด วิธีการควบคุมภาพเคลื่อนไหวที่นิยมนำมาประยุกต์ใช้ได้แก่ (Adam,Lee 1993)

1. การควบคุมโดย ระบุขั้นตอนล่วงหน้า (Script Animation Control)
   เป็นการควบคุมการแสดงภาพหรือองค์ประกอบต่างๆ โดยกำหนดขั้นตอนเอาไว้
2. การควบคุมโดย กระบวนคำสั่งในโปรแกรม (Procedure Animation Control)
   เป็นการกำหนดพฤติกรรมเฉพาะขององค์ประกอบของภาพ มักใช้กับองค์ประกอบหลักของภาพเคลื่อนไหว เช่น สไปรต์ (Sprite) เป็นต้น
3. การควบคุมโดย ใช้กฎเกณฑ์ทางกายภาพ (Physically Animation Control)
   เป็นการควบคุมโดยใช้กฎเกณฑ์ต่างๆ เช่น กฎทางฟิสิกส์ เป็นต้น

สไปรต์กับการทำภาพเคลื่อนไหว

ภาพเคลื่อนไหวในเกมคอมพิวเตอร์หรือในวิดีโอเกมเช่น ยานอวกาศ จรวดขีปนาวุธ วัตถุต่างๆและอื่นๆอีกหลายอย่างที่เคลื่อนที่ไปมาในจอภาพนั้นเราเรียกว่า สไปรต์(Sprite) ซึ่ง Christopher Lampton ได้ให้ความหมายของสไปรต์ไว้ในหนังสือ Flight of Fantasy ว่า "สไปรต์เป็นภาพขนาดเล็กๆ บนจอคอมพิวเตอร์ซึ่งสามารถเคลื่อนที่ไปบนภาพพื้นหลังได้โดยไม่ทำให้ภาพพื้นหลังนั้นเสียหาย" (Lampton, Christopher 1993) ในคอมพิวเตอร์บางชนิดมีฮาร์ดแวร์พิเศษสนับสนุนการเคลื่อนที่ของสไปรต์บนจอภาพ ทำให้แบ่งเบางานของหน่วยประมวลผลกลางและงานของผู้เขียนโปรแกรมไปได้อย่างมาก แต่สำหรับคอมพิวเตอร์ส่วนบุคคลที่สามารถทำงานเข้ากันได้กับเครื่องคอมพิวเตอร์ส่วนบุคคลของบริษัทไอบีเอ็มนั้นไม่มีฮาร์ดแวร์พิเศษสนับสนุนสไปรต์ ดังนั้นจึงจะต้องเขียนโปรแกรมสนับสนุนสไปรต์ขึ้นมาเอง

ในการเขียนโปรแกรมเพื่อสนับสนุนสไปรต์นั้นยังคงใช้หลักการเบื้องต้นของการทำภาพเคลื่อนไหวตามที่ได้กล่าวไปแล้ว โดยการประยุกต์หลักการของการสร้างภาพเคลื่อนไหวแบบ BitBlt Animation หรือ Frame Animation หรือ Real-time Animation ก็ได้ขึ้นอยู่กับความเหมาะสมของงาน แต่วิธีการที่นิยมใช้คือใช้หลักการแบบ BitBlt Animation สำหรับในวิทยานิพนธ์นี้จะใช้หลักการแบบ BitBlt และ Frame Animation มาประยุกต์เข้าด้วยกัน

สภาพปฏิบัติการไมโครซอฟต์วินโดวส์

ไมโครซอฟต์วินโดวส์ (Microsoft Windows) (ต่อไปจะใช้คำว่า วินโดวส์แทนไมโครซอฟต์วินโดวส์ และใช้คำว่า วินโดว์ แทนหน้าต่างงานในโปรแกรมประยุกต์) เป็นสภาพแวดล้อมในการทำงาน (operating environment) ที่ทำงานภายใต้ระบบปฏิบัติการ DOS มีตัวประสานกับผู้ใช้แบบกราฟิกส์ (Graphical User Interface) ซึ่งสามารถทำงานแบบหลายภารกิจ (multitasking) ในปัจจุบันมีการติดตั้งวินโดวส์บนเครื่องคอมพิวเตอร์ส่วนบุคคลอย่างแพร่หลาย มีซอฟต์แวร์ที่ใช้งานกับวินโดวส์มากมาย เนื่องจากความง่ายในการใช้งานของทั้งตัววินโดวส์และซอฟต์แวร์ที่ใช้งานภายใต้วินโดวส์ โปรแกรมที่พัฒนาสำหรับวินโดวส์จะมีรูปแบบการใช้งานในลักษณะเดียวกันกับมาตรฐานของวินโดวส์ ผู้ใช้สามารถเรียกใช้โปรแกรมได้หลายๆโปรแกรมพร้อมๆกันและยังสามารถแลกเปลี่ยนข้อมูลระหว่างโปรแกรมเหล่านั้นได้ด้วย

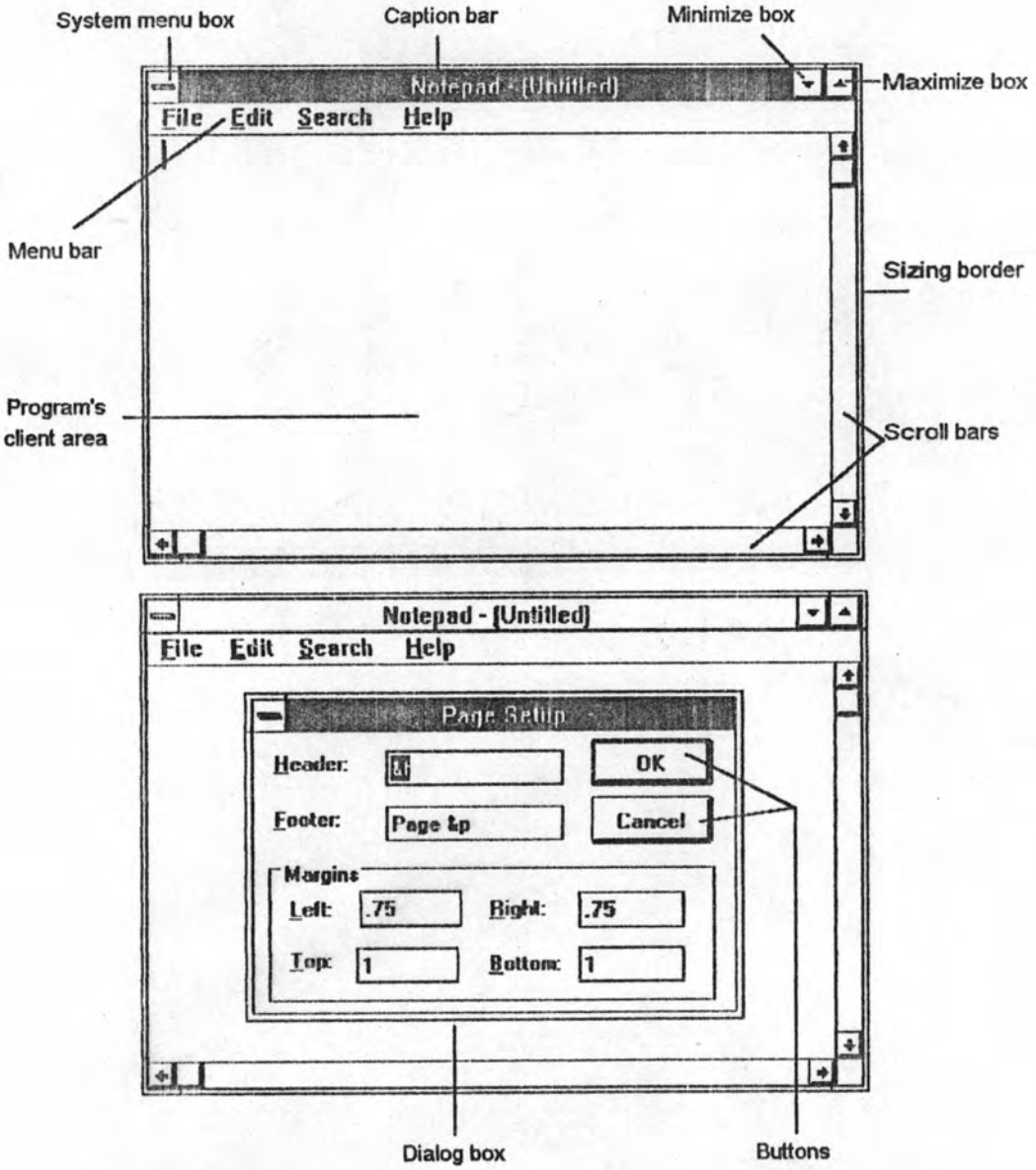
ความเป็นมาของไมโครซอฟต์วินโดวส์

บริษัทไมโครซอฟต์ทำการพัฒนาวินโดวส์และออกสู่ท้องตลาดในปี 2528 เป็นรุ่นแรก แต่ไม่ประสบความสำเร็จนัก เนื่องจากมีข้อจำกัดหลายอย่างและลักษณะการประสานกับผู้ใช้ยังไม่สวยงามเท่าที่ควร ต่อมาในวินโดวส์รุ่นที่ 2 จึงมีลักษณะการประสานกับผู้ใช้ที่ดีขึ้นจุดเด่นคือการปรากฏหน้าต่างหรือวินโดว์บนจอภาพในลักษณะซ้อนทับกันได้ (overlapping windows) นอกจากนี้ยังเพิ่มเติมคุณสมบัติของการใช้เมาส์และแป้นพิมพ์ การเลือกเมนู (menu) และการใช้กล่องโต้ตอบ (dialog box) ต่อมาวินโดวส์รุ่นที่ 2 ได้ถูกเปลี่ยนชื่อเป็น Windows/286 เนื่องจากวินโดวส์รุ่นใหม่คือ Windows/386 ที่ใช้คุณสมบัติของไมโครโปรเซสเซอร์ 80386 ต่อมาในปี 2533 Windows/286 และ Windows/386 ได้ถูกรวมเข้าด้วยกันเป็นวินโดวส์รุ่นที่ 3 โดยมีการปรับปรุงในวินโดวส์ทำให้สามารถอ้างอิงหน่วยความจำของระบบได้ถึง 16 เมกกะไบต์ วินโดว์บนจอภาพดูสวยงามขึ้น การใช้งานต่างๆมีผลตอบสนองที่ดีขึ้น และในปี พ.ศ. 2535 บริษัทไมโครซอฟต์ได้ทำการปรับปรุงวินโดวส์อีก เป็นรุ่นที่ 3.1 โดยเพิ่มการใช้เทคโนโลยีแบบอักษรที่เรียกว่า TrueType ซึ่งใช้แบบอักษรแบบย่อ/ขยายได้ (scalable font) ทำให้การแสดงผลหรือการพิมพ์ตัวอักษรมีลักษณะสวยงาม ไม่ขึ้นกับขนาดของตัวอักษร (ในรุ่นก่อนใช้แบบอักษรแบบบิตแมพ) นอกจากนี้วินโดว์ยังได้เพิ่มเติมส่วนการจัดการมัลติมีเดีย การเชื่อมโยงข้อมูลแบบ OLE (Object Linking and Embedding) และการใช้กล่องโต้ตอบร่วมส่วนกลาง (common dialog boxes)

การประสานกับผู้ใช้

การประสานกับผู้ใช้ในวินโดวส์เป็นแบบกราฟิกส์ (Graphical User Interface - GUI) ซึ่งเป็นวิธีการติดต่อกับผู้ใช้ที่มีต้นกำเนิดมาจากงานวิจัยที่ Xerox Palo Alto Research Center ในกลางทศวรรษ1970 โปรแกรมประยุกต์สามารถขอใช้จอภาพได้หลายโปรแกรมในเวลาเดียวกันโดยแบ่งพื้นที่บนจอภาพออกเป็นกรอบสี่เหลี่ยมเรียกว่า หน้าต่าง หรือ วินโดว์ (window) และมีการติดต่อกับผู้ใช้ผ่านทางส่วนประกอบต่างๆเช่น เมนู กรอบโต้ตอบ ปุ่มบังคับ หรือแถบเลื่อน เป็นต้น การแสดงผลแบบกราฟิกส์ทำให้การแสดงภาพหรือข้อความบนจอภาพมีลักษณะที่สวยงามสื่อความหมาย ซอฟต์แวร์ที่ทำงานแบบ GUI ซึ่งมีการแสดงสัญรูป (icon) แทนโปรแกรมหรือแฟ้มข้อมูล มีการแสดงปุ่มบังคับและเมนูที่แทนคำสั่งของผู้ใช้ มีการแสดงกล่องโต้ตอบเพื่อรับข้อมูลจากผู้ใช้ มีการแสดงแถบเลื่อนเพื่อเลื่อนภาพที่แสดงบนจอภาพ ประกอบกับการใช้เมาส์เพื่อควบคุมการเคลื่อนที่ของตัวชี้บนจอภาพไปยังตำแหน่งบนจอภาพ ที่มีสัญรูป ปุ่มทำงาน เมนู หรือแถบเลื่อนที่ต้องการ ทำให้ผู้ใช้งานทำงานได้ตรงกับที่เห็นบนจอภาพ เช่นถ้าผู้ใช้ต้องการย้ายแฟ้มข้อมูลจากสารบบ (directory) หนึ่งไปอีกสารบบหนึ่ง ก็เพียงแต่เลือกสัญรูปที่แทนแฟ้มข้อมูลนั้น แล้วก็ลากสัญรูปนั้นไปปล่อยไว้ในตำแหน่งของสารบบใหม่ (drag and

drop) ระหว่างการลากนั้น ระบบจะแสดงให้เห็นว่าสัญรูปนั้นถูกลากตามการเคลื่อนของเมาส์ที่ผู้ใช้ควบคุม เพื่อให้ผู้รู้สึกถึงการกระทำนั้นๆว่าเกิดขึ้นจริง เป็นต้น

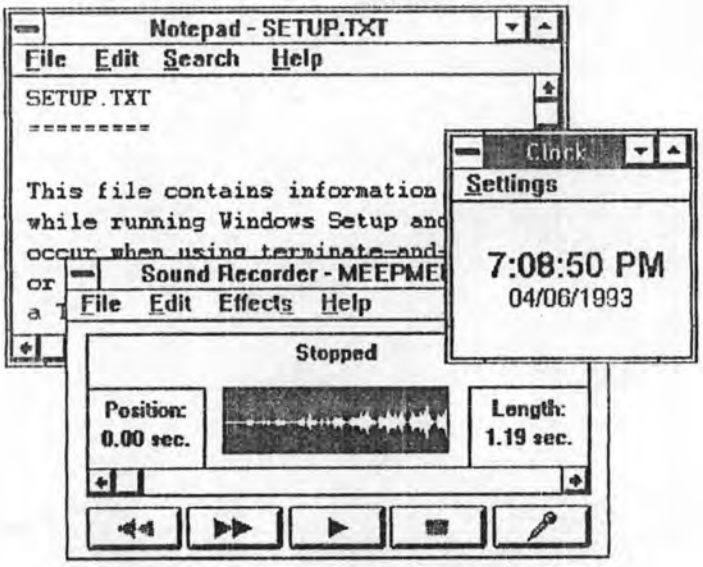


รูปที่ 2.1 องค์ประกอบต่างๆของการประสานกับผู้ใช้แบบกราฟิกส์ของวินโดวส์

การทำงานแบบหลายภารกิจ

วินโดวส์อนุญาตให้โปรแกรมหลายโปรแกรมถูกแสดงเป็นวินโดว์ที่อาจซ้อนทับกันบนจอภาพ ดังตัวอย่างในรูปที่ 2.2 แสดงให้เห็นถึงการทำงานของโปรแกรมสามโปรแกรม แต่ละโปรแกรมปรากฏอยู่ในวินโดว์ของตัวเอง วินโดว์เหล่านี้อาจซ้อนทับกันดังแสดงในรูป โดยผู้ใช้สามารถย้ายตำแหน่ง หรือย่อขยายขนาดของวินโดว์ได้ตามต้องการ วินโดวส์ที่มี caption bar ที่เด่นกว่าวินโดว์อื่นๆและพร้อมที่จะรับข้อมูลจากผู้ใช้ (ในตัวอย่างนี้คือ วินโดว์ Clock) เราเรียกวินโดว์นี้ว่ามี input focus นอกจากวินโดวส์จะสามารถแสดงวินโดว์ของโปรแกรมต่างๆบนจอภาพเดียวกันได้แล้ว วินโดวส์ยังอนุญาตให้โปรแกรมเหล่านี้ทำงานพร้อมๆกันไปได้ด้วย เรียกว่าเป็นการทำงานแบบหลายภารกิจ

การทำงานแบบหลายภารกิจของวินโดวส์เป็นไปได้ก็ด้วยความสามารถในการจัดการเกี่ยวกับเนื้อที่ในหน่วยจำ การนำโปรแกรมใหม่เข้าในระบบ หรือการลบโปรแกรมเก่าที่ทำงานเสร็จแล้วออกจากระบบ ล้วนยุ่งเกี่ยวกับหน่วยความจำทั้งสิ้น โดยเฉพาะอย่างยิ่งการจัดการเรื่องตำแหน่งของชุดคำสั่งและตำแหน่งของข้อมูลของแต่ละโปรแกรม วินโดวส์จะทำการเคลื่อนย้ายโปรแกรมที่เก็บในหน่วยความจำหลัก เพื่อให้มีเนื้อที่ว่างพอที่นำโปรแกรมใหม่เข้ามา และบางกรณีอาจจะต้องนำส่วนของบางโปรแกรมออกจากหน่วยความจำไปเก็บไว้ในหน่วยความสำรองเป็นการชั่วคราว โดยจะนำส่วนของโปรแกรมเหล่านี้คืนสู่หน่วยความจำเมื่อโปรแกรมเหล่านี้ต้องการทำงานต่อ นอกจากนี้ในกรณีที่วินโดวส์ทำงานบนเครื่องที่ใช้ไมโครโปรเซสเซอร์ 80386 หรือสูงกว่า และมีหน่วยความจำเพียงพอ วินโดวส์จะสามารถทำงานโดยใช้เนื้อที่บางส่วนในฮาร์ดดิสก์ เป็นหน่วยความจำเสมือน (virtual memory) เพื่อให้เสมือนกับว่าระบบมีหน่วยความจำมากขึ้นกว่าที่มี ในแง่ของผู้ใช้แล้วก็เปรียบเสมือนกับมีหน่วยความจำหลักที่มากขึ้นจึงทำให้โปรแกรมที่มีขนาดใหญ่ หรือใช้เนื้อที่ข้อมูลในหน่วยความจำมาก ทำงานได้



รูปที่ 2.2 การทำงานแบบหลายภารกิจของวินโดวส์

<u>การพัฒนาโปรแกรมภายใต้วินโดวส์</u>

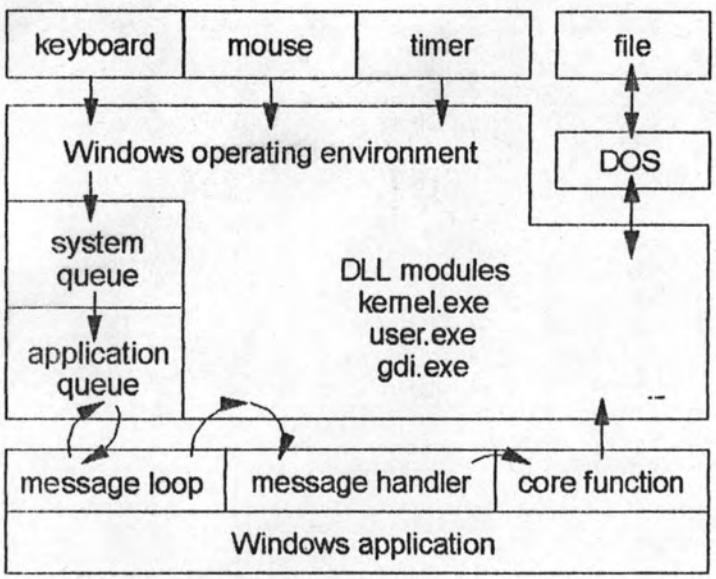
เนื่องจากความง่ายในการใช้งานจึงทำให้วินโดวส์ได้รับความนิยมเป็นอย่างสูง แต่การพัฒนาโปรแกรมประยุกต์หรือซอฟต์แวร์ที่ทำงานภายใต้วินโดวส์นั้นเป็นเรื่องยากและซับซ้อน โดยเฉพาะซอฟต์แวร์ที่พัฒนาด้วยภาษาซี ผู้เขียนโปรแกรมที่ไม่เคยมีประสบการณ์กับการพัฒนาในสภาพปฏิบัติการวินโดวส์จะพบกับแนวคิดและลักษณะขั้นตอนการพัฒนาที่ต่างจากการพัฒนาซอฟต์แวร์สำหรับระบบปฏิบัติการทั่วๆไปการทำงานของโปรแกรมภายใต้วินโดวส์ เมื่อโปรแกรมประยุกต์ถูกเลือกให้ทำงาน วินโดวส์จะเริ่มจัดเตรียมเนื้อที่ในหน่วยจำให้เพียงพอกับความต้องการโปรแกรมนั้น จากนั้นจึงนำโปรแกรม ข้อมูล และทรัพยากรต่างๆที่โปรแกรมนี้ใช้ (เช่นสัญรูป เมนู กล่องโต้ตอบ และอื่นๆ) เข้าสู่หน่วยความจำ พร้อมทั้งจัดเตรียมสแตกเพื่อการทำงานของโปรแกรมนั้น จากนั้นวินโดวส์จึงผ่านการทำงานไปให้โปรแกรม ในกรณีที่โปรแกรมนั้นถูกพัฒนาด้วยภาษาซี ฟังก์ชัน WinMain ของโปรแกรมนั้นจะเป็นฟังก์ชันที่วินโดวส์เรียก ภายใน WinMain จะมีวงวน (loop) พิเศษอยู่หนึ่งวง ซึ่งเป็นวงวนการทำงานเพื่ออ่านข้อความขาเข้าจากแถวคอยของโปรแกรมประยุกต์ (application queue) นั้น เราเรียกวงวนนี้ว่า วงวนข้อความ (message loop) ในรูปที่ 2.3 แสดงวงวนข้อความในโปรแกรมที่พัฒนาด้วยภาษาซี เมื่อใดที่มีข้อความขาเข้าอยู่ในแถวคอย ตัวโปรแกรมจะอ่านข้อความนั้น แล้วแปลงข้อความเสียก่อน (เฉพาะในกรณีที่เป็นข้อความจากการกดแป้นพิมพ์) จากนั้นจึงส่งข้อความนั้นคืนให้วินโดวส์เพื่อให้วินโดวส์จะส่งข้อความ (DispatchMessage) ไปยังโปรแกรมย่อยที่เรียกว่า วินโดว์ฟังก์ชัน (window function) หรือฟังก์ชันประจำวินโดว์ซึ่งทำหน้าที่ตามความหมายของข้อความต่างๆที่เหมาะสม (message

handler) ของโปรแกรมนั้นต่อไป วงวนข้อความจะทำงานในลักษณะเช่นนี้ไปจนกว่าจะได้รับข้อความ WM_QUIT ที่แสดงถึงการเสร็จสิ้นการทำงานของโปรแกรมนี้

```
while ( GetMessage( &msg, NULL, NULL, NULL ) ) {
    TranslateMessage( &msg );
    DispatchMessage( &msg );
};
```

รูปที่ 2.3 วงวนข้อความ (message loop)

เนื่องจากวินโดวส์เป็นระบบที่ทำงานแบบหลายภารกิจโดยไม่ยื้อแย่งการทำงานของโปรแกรมใด ที่เรียกกันว่า Non-preemptive multitasking system หมายความว่าหากโปรแกรมหนึ่งทำงานนานมาก โปรแกรมอื่นๆในระบบก็ไม่สามารถทำงานได้ ดังนั้นโปรแกรมในระบบเช่นนี้จะต้องยอมปล่อยการทำงาน เพื่อให้วินโดวส์สั่งการให้โปรแกรมอื่นทำงานบ้าง



รูปที่ 2.4 การโต้ตอบระหว่างโปรแกรมกับสภาพปฏิบัติการวินโดวส์

ผังในรูปที่ 2.4 แสดงการส่งผ่านข้อมูลต่างๆระหว่างฮาร์ดแวร์ วินโดวส์ และโปรแกรมประยุกต์ ข้อมูลขาเข้าที่ถูกจัดการโดยวินโดวส์มีสามประเภทคือ ข้อมูลจากแป้นพิมพ์ จากเมาส์ และจากตัวจับเวลา (timer) เมื่อใดที่มีข้อมูลเข้า เช่นผู้ใช้กดแป้นพิมพ์ หรือเคลื่อนเมาส์ เป็นต้น เหตุการณ์เหล่านี้จะถูกบันทึกเป็นข้อความลงในแถวคอยของระบบ จากนั้นข้อความเหล่านี้จะถูกย้ายไปเก็บไว้ในแถวคอยของโปรแกรมประยุกต์ที่เกี่ยวข้อง (แต่ละโปรแกรมประยุกต์จะมีแถวคอยของตัวเอง) ด้วยการใช้วงวนข้อความในโปรแกรม ข้อความเหล่านี้จะถูกอ่านออกมา แล้ววินโดวส์ฟังก์ชันที่เกี่ยวข้องก็จะถูกสั่งงาน

ในกรณีที่โปรแกรมประยุกต์มีการติดต่อกับระบบเช่น การเปิดวินโดว์ใหม่ การวาดวงกลมบนวินโดว์ การจองเนื้อที่ข้อมูลในหน่วยความจำ การพิมพ์แฟ้มข้อมูลออกทางเครื่องพิมพ์ เป็นต้น โปรแกรมเหล่านี้กระทำได้โดยการเรียกผ่านฟังก์ชันสำหรับตัวประสานวินโดวส์กับโปรแกรมประยุกต์ (Windows Application Programming Interface - API) วินโดวส์มีคลังโปรแกรมมากมายกว่า 600 ฟังก์ชันที่ทำงานติดต่อควบคุมระบบ ซึ่งถูกจัดเก็บไว้ในแฟ้มข้อมูลตามประเภท ดังนี้

1. USER ประกอบด้วยคลังโปรแกรมที่จัดการเกี่ยวกับวินโดว์บนจอภาพ
2. KERNEL ประกอบด้วยคลังโปรแกรมที่เกี่ยวกับการทำงานแบบหลายภารกิจ การจัดการหน่วยความจำและการจัดการทรัพยากร
3. GDI ประกอบด้วยคลังโปรแกรมเพื่อจัดการการแสดงผลแบบกราฟิกส์

เนื่องจากโปรแกรมประยุกต์ต้องเรียกใช้ Windows API ในกรณีที่ต้องการติดต่อกับระบบ ดังนั้นวินโดวส์สามารถแยกการควบคุมฮาร์ดแวร์ออกจากโปรแกรมประยุกต์ต่างๆได้ ทำให้ผู้พัฒนาซอฟต์แวร์สำหรับวินโดวส์ไม่ต้องกังวลถึงฮาร์ดแวร์ที่ใช้อยู่จริงในระบบ เพียงแต่เลือกใช้ฟังก์ชันที่เหมาะสมที่วินโดวส์มีให้เลือกใช้มากมาย วินโดวส์ควบคุมอุปกรณ์ต่างๆภายในระบบผ่านทางตัวขับอุปกรณ์ (device driver) ซึ่งถูกติดตั้งอยู่สำหรับแต่ละอุปกรณ์ในระบบ การเปลี่ยนอุปกรณ์เช่นการเปลี่ยนเครื่องพิมพ์จากยี่ห้อหนึ่งไปใช้อีกยี่ห้อหนึ่งซึ่งมีคำสั่งควบคุมเครื่องพิมพ์ที่ต่างกันนั้น จะไม่มีผลอันใดต่อโปรแกรมประยุกต์ เพียงแต่ผู้ใช้เปลี่ยนตัวขับอุปกรณ์ให้ถูกต้องเท่านั้น โปรแกรมประยุกต์ต่างๆที่มีอยู่ก็จะใช้ได้กับอุปกรณ์ตัวใหม่ทันที

การทำงานเชิงข้อความ

โครงสร้างของโปรแกรมที่ทำงานภายใต้วินโดว์จะถูกจัดให้อยู่ในลักษณะการทำงานที่เรียกว่า การทำงานเชิงข้อความ (message-driven) โปรแกรมที่ทำงานในลักษณะนี้จะประกอบไปด้วยโปรแกรมย่อยจำนวนหนึ่งซึ่งแต่ละโปรแกรมย่อยจะทำงานเมื่อมีเหตุการณ์บางอย่างเกิดขึ้น เช่นโปรแกรมย่อย A จะทำงานเมื่อมีการกดปุ่มของเมาส์ โปรแกรมย่อย B จะทำงานเมื่อได้รับสัญญาณจากตัวจับเวลา โปรแกรมย่อย C จะทำงานเมื่อขนาดของวินโดว์มีการเปลี่ยนแปลงเหล่านี้เป็นต้น เมื่อเหตุการณ์เหล่านี้เกิดขึ้นกับโปรแกรมประยุกต์หนึ่งๆ วินโดวส์จะเพิ่มข้อความที่มีความหมายแทนเหตุการณ์ที่เกิดขึ้นไว้ในแถวคอยข้อความของโปรแกรมประยุกต์นั้น ข้อความเหล่านี้จะถูกนำออกมาตีความในวงวนข้อความที่กล่าวในหัวข้อที่แล้ว ประเด็นสำคัญอยู่ที่ว่า หลังจากอ่านข้อความนี้ออกจากแถวคอยแล้ว จะส่งข้อความนี้คืนสู่วินโดวส์เพื่อให้วินโดวส์เป็นตัวส่งข้อความนี้ไปยังวินโดว์ฟังก์ชันที่เกี่ยวข้องกับข้อความนั้น วินโดว์ฟังก์ชันในที่นี้ก็คือโปรแกรมย่อยที่เตรียมไว้รองรับเหตุการณ์ต่างๆที่จะเกิดขึ้น จึงเปรียบเสมือนกับการที่วินโดวส์เรียกใช้วินโดว์ฟังก์ชัน ซึ่งเป็นแนวคิดที่ต่างกับการทำงานดั้งเดิม ซึ่งมีแต่โปรแกรมประยุกต์จะเรียกใช้บริการที่ระบบปฏิบัติการมีให้ใช้ แต่ในที่นี้ระบบปฏิบัติการกลับเรียกใช้โปรแกรมย่อยที่โปรแกรมประยุกต์สร้างไว้ ฟังก์ชันลักษณะนี้เรียกว่า callback function

คลังโปรแกรมเชื่อมโยงแบบพลวัต

คลังโปรแกรม (library) จะถูกเก็บอยู่ในรูปแบบที่เรียกว่า คลังโปรแกรมเชื่อมโยงแบบพลวัต (Dynamic Linking Library - DLL) แฟ้มข้อมูลที่เป็นโปรแกรมที่ทำงานภายใต้วินโดวส์นั้น (ในรูปแบบ EXE) จะมีตารางซึ่งเก็บชื่อหรือหมายเลขซึ่งแทนฟังก์ชันในคลังโปรแกรมที่ถูกอ้างอิงถึงในโปรแกรมนี้ เมื่อโปรแกรมนี้ถูกนำเข้าสู่หน่วยความจำ ตำแหน่งต่างๆในโปรแกรมที่อ้างอิงถึงฟังก์ชันแบบ DLL จะได้รับการเชื่อมโยงไปสู่ตำแหน่งจริงของฟังก์ชันเหล่านั้น (ซึ่งได้นำเข้าเก็บในหน่วยความจำไว้ก่อนแล้ว) การเชื่อมโยงตำแหน่งขณะนำโปรแกรมเข้านี่เองที่แตกต่างกับการเชื่อมโยงตำแหน่งแบบที่กระทำขณะสร้าง object code (ซึ่งเรียกว่าการเชื่องโยงแบบสถิต - Static linking) ถ้ามีการเปลี่ยนแปลงการทำงานของฟังก์ชันในคลังโปรแกรมผู้เขียนโปรแกรมไม่จำเป็นต้องสร้าง object code ใหม่ ถ้าใช้ DLL ข้อดีที่สำคัญอีกประการหนึ่งของการใช้ DLL คือการที่มีฟังก์ชันใน DLL อยู่ในหน่วยความจำเพียงชุดเดียว ถึงแม้ว่าจะมีมากกว่าหนึ่งโปรแกรมที่ใช้ฟังก์ชันเดียวกันใน DLL นั้น ในขณะที่ถ้าโปรแกรมสองโปรแกรมที่ใช้การเชื่อมโยงแบบสถิตกับฟังก์ชันหนึ่งขณะที่ทั้งสองโปรแกรมนี้ทำงาน ภายในหน่วยความจำจะมีฟังก์ชันที่ทั้งคู่เรียกใช้สองชุด นอกจาก DLL จะเก็บฟังก์ชันต่างๆแล้วเรายังสามารถใช้ DLL ในการเก็บข้อมูล

ทรัพยากร (เช่น แบบอักษร สัญรูป เมนู กล่องโต้ตอบ รูปแบบตัวชี้ เป็นต้น) และตัวขับอุปกรณ์ ได้ด้วย ทั้งนี้เพื่อจุดประสงค์ในการใช้สิ่งเหล่านี้ร่วมกันระหว่างโปรแกรมต่างๆที่ทำงานภายใต้วินโดวส์

การสร้าง Dynamic Link Library

ในการสร้าง DLL จะมีรูปแบบคล้ายโปรแกรมหลักทั่วๆไป ดังโครงร่างต่อไปนี้

```
int FAR PASCAL LibMain (HANDLE hInstance,   WORD   wDataSeg,
                        WORD   wHeapSize,  LPSTR   lpszCmdLine)
{
    Initial Routine;
    return  1;
}
VOID  FAR PASCAL WEP (int   nParameter)
{
   Shutdown Routine;
}
VOID  FAR PASCAL   FunctionName(....)
{
   : : :
   : : :
}
```

จากโครงสร้างดังกล่าวนี้ มีฟังก์ชันที่จะต้องอยู่ใน DLL File เสมอคือ ฟังก์ชัน LibMain() ซึ่งทำหน้าที่คล้าย WinMain() ในโปรแกรมทั่วๆไป โดยที่ LibMain() จะเป็นฟังก์ชันแรกที่ทำงาน เมื่อ DLL นี้ถูกโหลดเข้าสู่หน่วยความจำหลัก ฟังก์ชัน WEP() หรือ Windows Exit Procedure เป็นฟังก์ชันที่ระบบไมโครซอฟต์วินโดวส์ จะเรียกใช้ก่อนที่จะลบ DLL นี้ออกจากหน่วยความจำหลัก

เทคนิคการเขียนโปรแกรมสร้างภาพเคลื่อนไหวภายใต้วินโดวส์

การพัฒนาโปรแกรมประยุกต์ที่ทำงานภายใต้วินโดวส์นั้นจำเป็นอย่างยิ่งที่จะต้องคำนึงถึงการทำงานของตัววินโดวส์เองด้วย เนื่องจากความเป็น Non-preemptive ของวินโดวส์ดังนั้นโปรแกรมประยุกต์จะต้องมีการคืนทรัพยากรที่วินโดวส์ให้ใช้ในเวลาอันสมควรและเหมาะสมจึงจะทำให้งานทั้งระบบดำเนินไปอย่างราบรื่น ในการพัฒนาโปรแกรมทำภาพเคลื่อนไหวก็เช่นกัน

การเขียนโปรแกรมทำให้เกิดภาพเคลื่อนไหว

ภายใต้วินโดวส์มีวิธีการเขียนโปรแกรมที่ต่างจากการเขียนโปรแกรมบนระบบ MS-DOS อยู่เล็กน้อย สิ่งที่ต้องคำนึงถึงเสมอคือ บนวินโดวส์นั้นทรัพยากรทุกอย่างจะต้องแบ่งกันใช้ เมื่อใช้เสร็จแล้วต้องคืนให้กับระบบ เพื่อให้โปรแกรมอื่นมีโอกาสได้ใช้ทรัพยากรนั้นบ้าง วิธีสร้างโปรแกรมทำให้เกิดภาพเคลื่อนไหวบน MS-DOS มักใช้วงวนแบบปกติทั่วไป แต่ภายใต้วินโดวส์สามารถใช้วิธีการต่อไปนี้ (Adam,Lee 1993)

1. Fixed-loop
2. Idle-loop
3. Timer-based message-handler
4. Timer-based direct-call

วิธีการแบบวงวน

Fixed-loop เป็นวงวนทั่วๆไปในภาษาคอมพิวเตอร์ โดยเป็นวงวนที่กำหนดจำนวนรอบในการทำงานของกลุ่มคำสั่งภายใน Block ตัวอย่างเช่น วงวนแบบ for การใช้ fixed-loop นี้หน่วยประมวลผลกลางไม่สามารถแบ่งเวลาให้กับโปรแกรมอื่นได้มีโอกาสใช้เลย เนื่องจากวินโดวส์เป็นระบบการทำงานหลายภารกิจแบบ Non-Preemptive ดังนั้นการใช้วงวนแบบนี้มักใช้เมื่อต้องการวนอย่างสั้นๆ เพื่อให้โปรแกรมอื่นที่กำลังทำงานอยู่นั้น ได้มีโอกาสใช้หน่วยประมวลผลกลางและยังสามารถดำเนินต่อไปได้โดยไม่มีผลกระทบมากนัก

เราสามารถปรับแต่ง Fixed-loop มาเป็นวงวนแบบ Idle-loop ได้โดยใช้ฟังก์ชัน GetMessage() และ DispatchMessage() เพื่อดึงข้อความจากแถวคอย (Application Queue) ในระหว่างวงวน ซึ่งฟังก์ชันเหล่านี้มีไว้ให้แล้วใน API ของวินโดวส์ ทำให้ระบบสามารถแบ่งเวลาของการใช้บริการหน่วยประมวลผลกลางให้กับโปรแกรมอื่นๆได้อย่างทั่วถึงกัน

วิธีการแบบใช้ตัวจับเวลา

ตัวจับเวลาในวินโดวส์คืออุปกรณ์ผลิตข้อมูลขาเข้าที่ส่งข้อความแจ้งให้กับโปรแกรมประยุกต์เป็นระยะๆ เมื่อช่วงเวลาที่กำหนดให้ตัวจับเวลานั้นหมดลง ตัวอย่างเช่นโปรแกรมหนึ่งบอกให้วินโดวส์ติดตั้งตัวจับเวลาที่จะส่งสัญญาณให้กับโปรแกรมนี้ทุกๆ 10 วินาที จากนั้นวินโดวส์จะส่งข้อความ WM_TIMER มาให้โปรแกรมนี้ทุกๆ 10 วินาที จนกว่าจะยกเลิกการใช้ตัวจับเวลา ตัวอย่างของงานที่ใช้ตัวจับเวลา

- การแสดงเวลา เช่นนาฬิกา ตัวจับเวลาจะเป็นตัวส่งสัญญาณให้กับนาฬิกาเพื่อแสดงเวลาใหม่บนวินโดว์ เช่นแสดงเวลาใหม่ทุกๆ 1 วินาที หรือทุกๆ 1 นาที เป็นต้น

- การแสดงสภาวะ เช่นโปรแกรมที่แสดงเนื้อที่หน่วยความจำที่ยังไม่ได้ถูกใช้ในระบบ ซึ่งอาจจะแสดงสภาวะนี้ทุกๆ 1 วินาที เป็นต้น

- การทำงานแบบหลายภารกิจ เนื่องจากวินโดวส์จะไม่ยื้อแย่งการทำงานของโปรแกรมใดๆ (non-preemptive) จึงจำเป็นอย่างยิ่งที่จะต้องคืนการควบคุมการใช้หน่วยประมวลผลให้เร็วที่สุด เพื่อให้โปรแกรมอื่นได้มีโอกาสใช้ ดังนั้นหากโปรแกรมหนึ่งมีการคำนวณซึ่งใช้เวลานานมาก อาจทำให้โปรแกรมอื่นๆในระบบไม่สามารถทำงานได้ จึงจำต้องแบ่งการทำงานที่ใช้เวลามากๆออกเป็นงานย่อย และแต่ละงานย่อยนี้จะทำเมื่อได้รับสัญญาณจากตัวจับเวลา

- การกำหนดความเร็วภาพเคลื่อนไหว การแสดงภาพเคลื่อนไหวสามารถถูกกำหนดความเร็วในการแสดงได้โดยใช้ตัวจับเวลา เพื่อส่งสัญญาณมาให้หน่วยแสดงผลเปลี่ยนการแสดงภาพเป็นภาพถัดไป ดังนั้นความเร็วในการเคลื่อนไหวของภาพจึงขึ้นกับความถี่ของสัญญาณจากตัวจับเวลาที่ส่งมาให้โปรแกรม

วินโดวส์มีตัวจับเวลาทั้งหมด 32 ตัว ให้โปรแกรมต่างๆในระบบใช้ วินโดวส์ใช้ชุดคำสั่งควบคุมตัวจับเวลาที่อยู่ใน BIOS ของเครื่อง (Interrupt 08H) ที่กำหนดให้สร้างสัญญาณออกมาได้ทุกๆ 54.925 มิลลิวินาที (ประมาณ 18.2 ครั้งในหนึ่งวินาที) ซึ่งคือช่วงเวลาที่สั้นที่สุดที่สามารถกำหนดให้กับตัวจับเวลาตัวหนึ่ง ข้อพึงระวังในการใช้ตัวจับเวลาคือ เมื่อถึงเวลาที่กำหนดของตัวจับเวลาตัวหนึ่ง วินโดวส์จะส่งข้อความ WM_TIMER ไปให้โปรแกรมที่ใช้ตัวจับเวลาตัวนั้น ข้อความนี้จะถูกเก็บอยู่ในแถวคอยข้อความของโปรแกรมนั้น ดังนั้นกว่าที่ข้อความนี้จะถูกอ่านโดยโปรแกรม (ในวงวนข้อความ) อาจใช้เวลาห่างจากช่วงที่เหตุการณ์ที่ตัวจับเวลาส่งสัญญาณก็ได้ ข้อพึงระวังอีกประการหนึ่งคือวินโดวส์จะใส่ข้อความ WM_TIMER ไว้ในแถวคอยข้อความก็ต่อเมื่อไม่มีข้อความใดในแถวคอย (ยกเว้น WM_PAINT) ทั้งนี้เพื่อป้องกันกรณีที่แถวคอยมีข้อความเต็มไปด้วย WM_TIMER ในกรณีที่ได้ตั้งช่วงเวลาไว้สั้นเกินไป

เนื่องจากมีการสร้างสัญญาณทุกๆ 55 มิลลิวินาที (โดยประมาณ) ดังนั้นการกำหนดช่วงเวลาในการส่งข้อความ WM_TIMER สามารถปรับได้ตามตารางต่อไปนี้

สัญญาณครั้งที่	ช่วงเวลา (มิลลิวินาที)	จำนวนครั้งต่อวินาที
1	55	18.2
2	110	9.1
3	165	6.1
4	220	4.5
5	275	3.6
6	330	3.03
7	385	2.6

ตารางที่ 2.1 แสดงระยะเวลาที่มีการสร้างสัญญาณเวลาจากตัวจับเวลา

ตัวจับเวลาที่กล่าวมานั้นทำงานในแบบส่งข้อความเข้าแถวคอย แต่ในวินโดวส์รุ่น 3.1 มีตัวจับเวลาอีกชนิดหนึ่งซึ่งรวมอยู่ในกลุ่มของฟังก์ชันควบคุมการทำงานอุปกรณ์มัลติมีเดีย (Multimedia Timer) ตัวจับเวลานี้ไม่ส่งข้อความ WM_TIMER แต่จะทำงานตามสัญญานขัดจังหวะที่ CPU ได้รับจากชิบ Timer ของคอมพิวเตอร์ โดยจะไปทำงานที่ฟังก์ชันชนิด callback ตามที่ระบุตอนเรียกใช้ตัวจับเวลา เนื่องจากไม่มีการส่งข้อความเข้าแถวคอยทำให้ไม่ต้องเสียเวลารอข้อความจากแถวคอย (เนื่องจาก WM_TIMER เป็น ข้อความที่มีลำดับความสำคัญน้อยกว่าข้อความชนิดอื่น) นอกจากนี้ตัวจับเวลาชนิดที่ส่งข้อความนั้นจะกำหนดช่วงเวลาสั้นที่สุดได้ที่ประมาณ 55 มิลลิวินาทีและความแม่นยำของเวลาขึ้นอยู่กับความหนาแน่นของข้อความอื่นในแถวคอย ส่วนตัวจับเวลาชนิด Multimedia Timer นั้นสามารถกำหนดช่วงเวลาได้ถึง 1 มิลลิวินาทีและมีความแม่นยำกว่า แต่ถ้ากำหนดช่วงเวลาให้สั้นมากๆ (น้อยกว่า 10 มิลลิวินาที) จะทำให้โปรแกรมประยุกต์ตัวอื่นมีโอกาสทำงานได้ช้าลงและประสิทธิภาพของระบบจะลดลงไป

<u>การเรียกใช้บริการตัวจับเวลา</u>

การเรียกใช้ตัวจับเวลาชนิดส่งข้อความ WM_TIMER ทำได้โดยเรียกใช้ฟังก์ชัน SetTimer() ซึ่งมีวิธีเรียกใช้ได้ 2 แบบ ได้แก่

1. กำหนดให้ส่ง WM_TIMER ไปยังแถวคอยของโปรแกรมประยุกต์ เพื่อรอการดึงข้อความไปใช้จากฟังก์ชัน GetMessage()

2. กำหนดให้มีการเรียกฟังก์ชัน (ชนิด callback) โดยตรงตามช่วงเวลาที่ตั้งเอาไว้

```
UINT  nTimerID;
nTimerID = SetTimer(hWnd, nTimerID, Time-Interval, lpProc);
```

hWnd = แฮนเดิลของวินโดว์ที่จะส่งข้อความ WM_TIMER ไปให้
nTimerID = หมายเลขของตัวจับเวลา
Time-Interval = ช่วงเวลาในการรอเพื่อจะส่งข้อความ
lpProc = พอยเตอร์ชี้ตำแหน่งของฟังก์ชันที่จะให้วินโดวส์เรียก (callback function)
ถ้ากำหนดเป็น NULL จะส่ง WM_TIMER ไปยังฟังก์ชันประจำวินโดว์แทน

เมื่อต้องการยกเลิกตัวจับเวลาจะใช้ฟังก์ชัน KillTimer() ดังนี้

```
KillTimer ( hWnd,  nTimerID);
```

สำหรับ Mutimedia Timer สามารถเรียกใช้ได้ดังนี้ (Microsoft Corp., Multimedia Programmer's workbook, 1992)

```
FARPROC    lpProc;
UINT       nTimerID;
WORD       wTimerRes;

lpProc = MakeProcInstance ( TimerFunction, hInstance );
timeBeginPeriod (wTimerRes);
nTimerID = timeSetEvent (wDelay, wResolution,
                         (LPTIMECALLBACK) lpProc, dwUser, wFlag );
```

โดยที่ TimerFunction จะต้องเป็นฟังก์ชันชนิด callback มีต้นแบบของฟังก์ชันดังนี้

```
void CALLBACK __export  TimerFunction ( UINT  nTimerID,      WORD wMsg,
                                        DWORD dwUser,    DWORD dw1,
                                        DWORD dw2);
```

ถ้า TimerFunction อยู่ใน DLL การเรียกใช้ไม่จำเป็นต้องหาตำแหน่งของ TimerFunction ด้วย MakeProcInstance() สามารถเรียกใช้ได้โดยตรง

การยกเลิกตัวจับเวลาชนิดนี้ทำได้โดยใช้ฟังก์ชัน timeKillEvent() ดังนี้

```
timeKillEvent ( nTimerID );
timeEndPeriod (wTimerRes);
FreeProcInstance ( lpProc );   // ไม่ต้องใช้ถ้าฟังก์ชันอยู่ใน DLL
```

ค่าของ wTimerRes เป็นค่า minimum resolution ซึ่งหาได้จากฟังก์ชัน timeGetDevCaps() ดังนี้

```
TIMECAPS   tc;
WORD       wTimerRes;

timeGetDevCaps (&tc,  sizeof(TIMECAPS));
```

โครงสร้าง TIMECAPS ประกอบไปด้วยสมาชิก 2 ตัวได้แก่

```
UINT   wPeriodMin;      // ช่วงเวลาน้อยที่สุดที่ระบบจะให้บริการได้ (msc.)
UINT   wPeriodMax;      // ช่วงเวลามากที่สุดที่ระบบจะให้บริการได้ (msc.)
```

ในการเขียนโปรแกรมใช้ตัวจับเวลาชนิด Multimedia Timer จะต้องมี header file ชื่อ mmsystem.h และยังมีข้อบังคับอีกหลายอย่างเช่น Code segment และ data segment ของฟังก์ชันชนิด callback ที่จะเรียกใช้นั้นจะต้องกำหนดให้เป็นหน่วยความจำแบบคงที่ (FIXED) ถ้ากำหนดให้เป็นแบบเคลื่อนย้ายได้ (MOVEABLE) อาจทำให้ระบบทั้งหมดหยุดทำงานได้ เป็นต้น

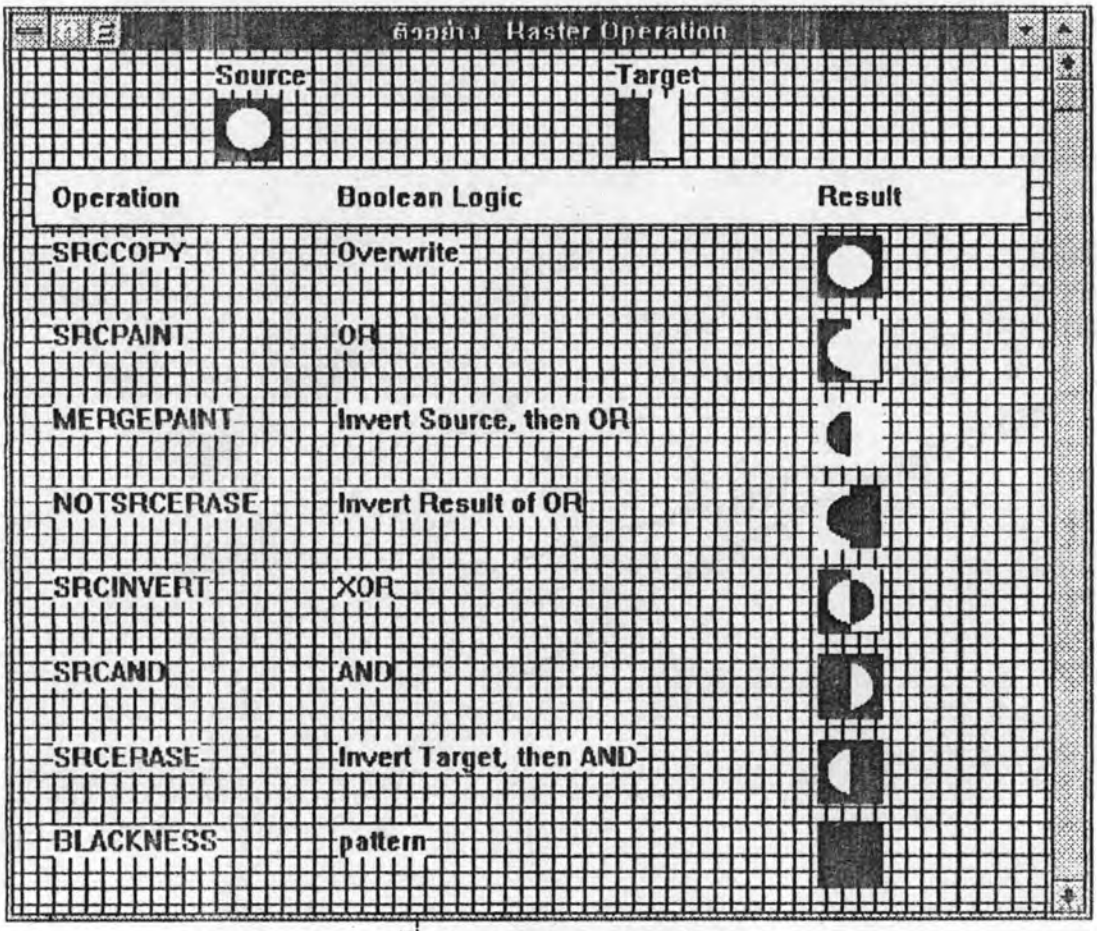
<u>เครื่องมือระดับ Low-Level</u>

ความสามารถของ Microsoft Windows Graphics Device Interface (เรียกสั้นๆว่า Windows GDI) ที่ใช้ในการจัดการตกแต่งและจัดเก็บ Graphic Arrays นั้นได้เตรียมเครื่องมือระดับพื้นฐาน (Low-

Level) ที่มีประโยชน์สำหรับการทำภาพเคลื่อนไหวไว้ให้แล้ว ซึ่งเครื่องมือเหล่านี้สามารถเรียกใช้งานได้โดยตรงและมีประสิทธิภาพมากสำหรับงานที่เกี่ยวกับ Graphic Arrays หรือ Block ของหน่วยความจำที่เก็บข้อสนเทศของภาพ โดยภาพเหล่านี้เรียกว่า ภาพบิตแมพ (Bitmap)

Blitting เป็นวิธีการสำเนาบิตแมพขนาดเล็กๆ ไปบนจอภาพ (ทำสำเนา block ของหน่วยความจำขนาดเล็กๆที่เก็บข้อสนเทศของภาพไปแสดงบนจอภาพ) ในทางปฏิบัติแล้ว ฟังก์ชันของ GDI ชื่อ BitBlt ทำหน้าที่นี้ได้ เนื่องจากฟังก์ชัน BitBlt() จะทำสำเนา block ของหน่วยความจำไปที่ใดๆก็ได้ ดังนั้นเราจึงใช้ฟังก์ชันนี้ทำสำเนาข้อมูลจากหน่วยความจำไปที่จอภาพ หรือทำสำเนาข้อมูลจากจอภาพไปเก็บในหน่วยความจำก็ได้

นอกจากวิธีการทำสำเนาข้อมูลดังกล่าวมาแล้ว GDI ยังมีการดำเนินการ Raster เพื่อใช้สร้างผลงานที่แตกต่างออกไป เมื่อมีการทำสำเนาบิตแมพไปที่ตำแหน่งใหม่ การดำเนินการ Raster เหล่านี้ใช้วิธีการตรรกแบบบูล (Boolean logic) เช่น ใช้ตัวดำเนินการแบบ XOR AND OR ทำให้เกิด Transparent ซึ่งกรรมวิธี Transparent put นี้เป็นการทำสำเนาบิตแมพลงไปยังภาพพื้นหลังแต่สามารถเห็นรายละเอียดของภาพพื้นหลังตามช่องว่างของรูปบิตแมพที่ทำสำเนาลงไปได้ (ถ้ารูปบิตแมพนั้นมีช่องว่าง) เหมือนกับว่าเรามองทะลุหรือมองลอดตามช่องได้ กรรมวิธี Transparent put นี้ทำให้เราสามารถปรับแต่งรูปบิตแมพที่เป็นสีเคลื่อนที่ไปบนภาพพื้นหลังที่เป็นสีได้อย่างสมจริงโดยไม่ทำให้ภาพพื้นหลังเสียหายหรือถูกลบไป ดังนั้นหลักการนี้จึงสามารถนำมาใช้ในการสร้างสไปรต์ได้เป็นอย่างดี รูปต่อไปนี้แสดงตัวอย่างของการดำเนินการ Raster ที่ใช้กับฟังก์ชัน BitBlt()



รูปที่ 2.5 แสดง Raster Operation