6. ตัวอย่างการจินตทัศน์อัลกอริทึมและการทดสอบประสิทธิภาพ

ในบทนี้จะแสดงถึงการทดลองนำระบบ AVis ไปใช้ในการพัฒนาการจินตทัศน์ของอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุด (minimum spanning tree) และอัลกอริทึมจัดเรียงข้อมูลในหน่วยความจำ (internal sorting) โดยจะกล่าวถึงโครงสร้างและองค์ประกอบการจินตทัศน์ที่สร้างขึ้น การเปรียบเทียบประสิทธิภาพการทำงานของการจินตทัศน์ที่สร้างขึ้นด้วยระบบ AVis กับการจินตทัศน์อัลกอริทึมที่พัฒนาขึ้นด้วยวิธีการอื่นๆ โดยองค์ประกอบการจินตทัศน์ทั้งหมดที่กล่าวถึงในบทนี้ รวมทั้งโปรแกรมที่พัฒนาขึ้นเพื่อเปรียบเทียบการทำงานกับการจินตทัศน์ที่พัฒนาด้วย AVis จะพัฒนาขึ้นโดยใช้วิชวลเบสิกรุ่น 3.0 ทั้งหมด เพื่อความสะดวกในการพัฒนาและไม่ให้ภาษาในการพัฒนาโปรแกรมเป็นสาเหตุของผลการทำงานของที่แตกต่างกัน

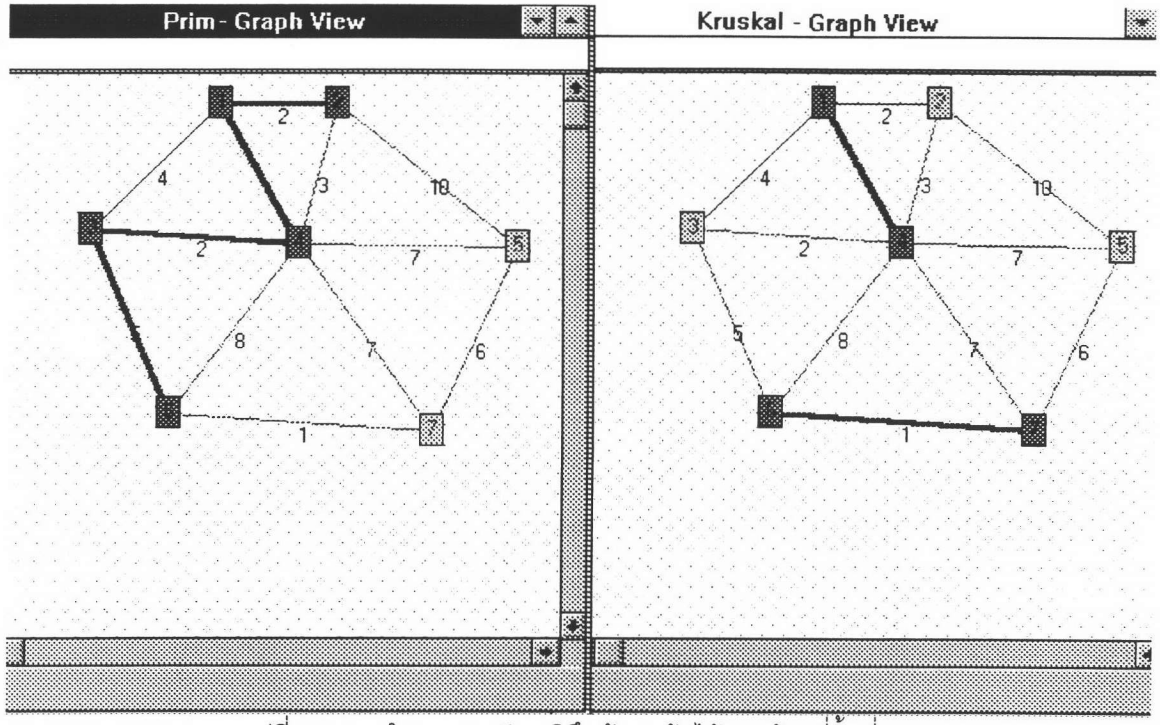
6.1 การจินตทัศน์อัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุด

6.1.1 โครงสร้างของการจินตทัศน์

ในการพัฒนาการจินตทัศน์ของอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุดได้ออกแบบให้แบ่งการจินตทัศน์ทั้งหมดออกเป็นองค์ประกอบการจินตทัศน์ 5 องค์ประกอบได้แก่

1. องค์ประกอบสร้างข้อมูลกราฟ (GraphEdit) เป็นองค์ประกอบที่ใช้ในการสร้างข้อมูลเพื่อนำข้อมูลของโครงสร้างข้อมูลรูปแบบกราฟที่ได้ไปใช้ในการทำงานของอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุด โดยโครงสร้างข้อมูลรูปแบบกราฟที่สร้างขึ้นจะอยู่ในรูปของเซตของจุด (vertex) ต่างๆของกราฟ และเซตของเส้นเชื่อม (edge) ระหว่างคู่จุดในกราฟ
2. องค์ประกอบอัลกอริทึมพริม (Prim) จะเป็นองค์ประกอบที่จัดเก็บอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุดของพริม
3. องค์ประกอบอัลกอริทึมครูสกัล (Kruskal) เป็นองค์ประกอบที่จัดเก็บอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุดของครูสกัล
4. องค์ประกอบแสดงข้อมูลแบบกราฟ (GraphView) ใช้แสดงภาพของโครงสร้างข้อมูลรูปแบบกราฟเพื่อให้ผู้ใช้การจินตทัศน์ทำการศึกษาทำความเข้าใจในขั้นตอนการทำงานของอัลกอริทึมต่างๆที่เกี่ยวกับโครงสร้างข้อมูลรูปแบบกราฟ
5. องค์ประกอบแปลงคำสั่งของอัลกอริทึมกราฟ (GraphConvertor) ทำหน้าที่แปลงข้อความคำสั่งแสดงผลขององค์ประกอบอัลกอริทึมให้อยู่ในรูปที่ GraphView เข้าใจ เพื่อจะได้ทำงานได้อย่างถูกต้อง

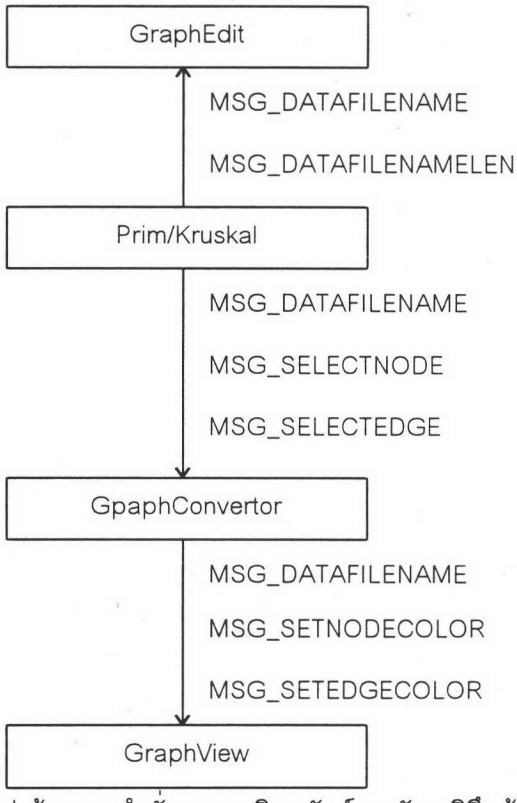
การทำงานขององค์ประกอบการจินตทัศน์เหล่านี้เมื่อนำมาประกอบเป็นบทการจินตทัศน์ดังแสดงไว้ในรูปที่ 6.1 จะพบว่าอัลกอริทึมของพริมจะทำงานโดยการเลือกจุดที่อยู่ใกล้กับจุดที่เลือกไว้แล้วที่สุดก่อน แต่อัลกอริทึมของครูสกัลจะใช้วิธีเลือกเส้นเชื่อมที่สั้นที่สุดก่อน



รูปที่ 6.1 การทำงานของอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุด

6.1.2 การติดต่อประสานงานระหว่างองค์ประกอบการจินตทัศน์

สิ่งสำคัญที่สุดสิ่งหนึ่งในการพัฒนาการจินตทัศน์ด้วยระบบ AVis ก็คือการออกแบบข้อความคำสั่งและการส่งข้อมูลระหว่างองค์ประกอบต่างๆในขณะทำการจินตทัศน์



รูปที่ 6.2 การส่งข้อความคำสั่งของการจินตทัศน์ของอัลกอริทึมค้นหาต้นไม้ทอดข้าม

ในรูปที่ 6.2 แสดงข้อความคำสั่งที่องค์ประกอบซึ่งใช้ในการจินตทัศน์อัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุดใช้ในการทำงาน เมื่อเริ่มการจินตทัศน์องค์ประกอบอัลกอริทึมทั้งสองจะส่งข้อความคำสั่งร้องขอข้อมูลไปยัง GraphEdit ซึ่ง GraphEdit จะตอบสนองด้วยการจัดเก็บข้อมูลของกราฟที่จะใช้ในการจินตทัศน์ลงเป็นแฟ้มข้อมูลส่งชื่อแฟ้มข้อมูลกลับมาให้องค์ประกอบอัลกอริทึม จากนั้นองค์ประกอบอัลกอริทึมก็จะส่งชื่อแฟ้มข้อมูลนี้ให้องค์ประกอบอื่นๆต่อไป

หลังจากได้รับข้อมูลองค์ประกอบอัลกอริทึมจะเริ่มทำการค้นหาต้นไม้ทอดข้ามที่สั้นที่สุด โดยในระหว่างทำงานจะส่งข้อความแสดงผลสองข้อความคือ MSG_SELECTNODE และ MSG_SELECTEDGE ข้อความคำสั่งทั้งสองจะถูกส่งออกไปให้ตัวแปลงคำสั่ง (GraphConvertor) ก็จะแปลงข้อความคำสั่งทั้งสองเป็นข้อความคำสั่ง MSG_SETNODECOLOR และ MSG_SETEDGECOLOR ตามลำดับ

6.2 การจินตทัศน์อัลกอริทึมจัดเรียงข้อมูล

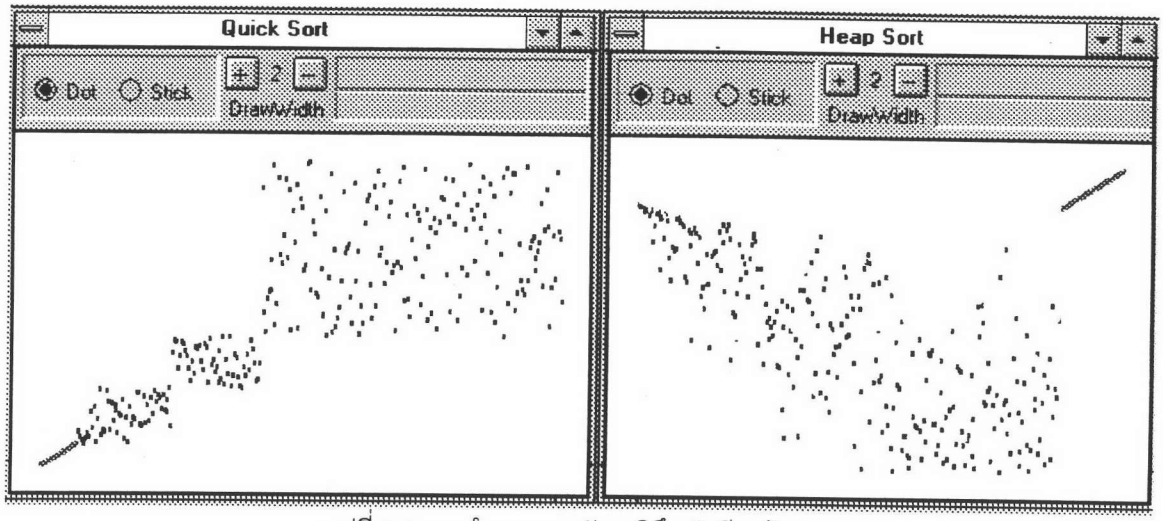
อัลกอริทึมจัดเรียงข้อมูลที่จะนำมากล่าวถึงในที่นี้ เป็นอัลกอริทึมจัดเรียงข้อมูลในกลุ่มที่เรียกว่า "การจัดเรียงข้อมูลในหน่วยความจำ" (internal sorting) ซึ่งเป็นการจัดเรียงข้อมูลที่ข้อมูลทั้งหมดถูกเก็บอยู่ในหน่วยความของระบบคอมพิวเตอร์ดังนั้นจึงสามารถอ้างถึงข้อมูลทั้งหมดได้อย่างรวดเร็วเท่าเทียมกันโดยไม่ขึ้นกับตำแหน่งของข้อมูล ตัวอย่างการจินตทัศน์ที่สร้างขึ้นจะจัดทำขึ้นสำหรับแสดงการจินตทัศน์ของอัลกอริทึมสองอัลกอริทึมได้แก่ อัลกอริทึมการจัดเรียงข้อมูลแบบฮีป และ อัลกอริทึมการจัดเรียงข้อมูลแบบเร็ว

6.2.1 โครงสร้างของการจินตทัศน์

ในการพัฒนาการจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลได้ออกแบบให้แบ่งการจินตทัศน์ทั้งหมดออกเป็นองค์ประกอบการจินตทัศน์ 5 องค์ประกอบได้แก่

1. องค์ประกอบสร้างข้อมูลเลข (Random) เป็นองค์ประกอบที่ใช้ในการสร้างข้อมูลตัวเลขเพื่อนำข้อมูลเลขเหล่านี้ไปให้อัลกอริทึมจัดเรียงข้อมูลนำข้อมูลไปใช้งานต่อไป
2. องค์ประกอบอัลกอริทึมการจัดเรียงข้อมูลแบบฮีป (HeapSort) จะเป็นองค์ประกอบที่จัดเก็บอัลกอริทึมจัดเรียงข้อมูลแบบฮีป
3. องค์ประกอบอัลกอริทึมการจัดเรียงข้อมูลแบบเร็ว (QuickSort) จะเป็นองค์ประกอบที่จัดเก็บอัลกอริทึมจัดเรียงข้อมูลแบบเร็ว
4. องค์ประกอบแสดงข้อมูลแบบจุดและแท่ง (StickView) ใช้แสดงภาพของข้อมูลตัวเลขในลักษณะของจุดหรือแท่งสีบนจอภาพเพื่อแสดงการเคลื่อนที่สลับตำแหน่งของข้อมูลในขณะที่อัลกอริทึมจัดเรียงข้อมูลกำลังทำงาน
5. องค์ประกอบแปลงคำสั่ง (DotConvertor) ทำหน้าที่แปลงข้อความคำสั่งแสดงผลขององค์ประกอบอัลกอริทึมให้อยู่ในรูปที่ StickView เข้าใจ เพื่อจะได้ทำงานได้อย่างถูกต้อง

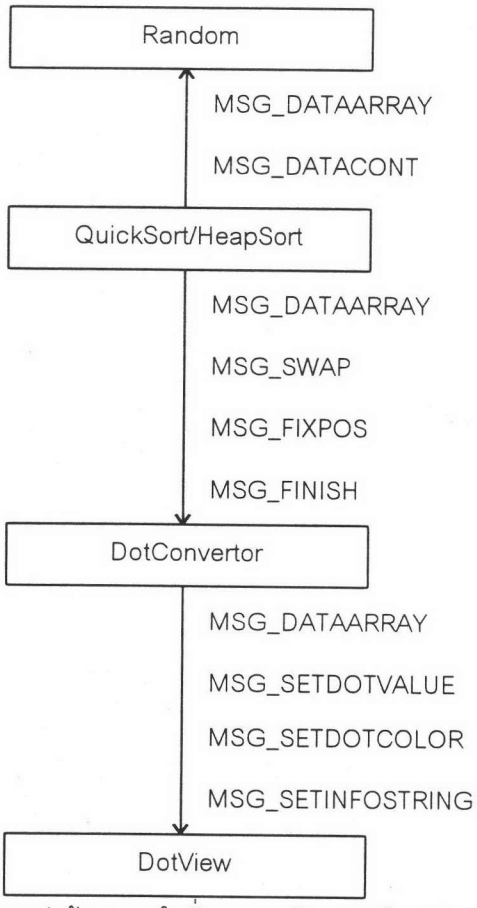
การงานขององค์ประกอบเหล่านี้จะแสดงไว้ในรูปที่ 6.3 โดยจะพบว่า HeapSort จะทำงานโดยการสร้างฮีปขึ้นก่อนแล้วค่อยๆดึงข้อมูลออกจากฮีป ในขณะที่ QuickSort จะใช้วิธีแบ่งข้อมูลออกเป็นกลุ่มย่อยๆ ตามค่าข้อมูล



รูปที่ 6.3 การทำงานของอัลกอริทึมจัดเรียงข้อมูล

6.2.2 การติดต่อประสานงานระหว่างองค์ประกอบการจินตทัศน์

ในรูปที่ 6.4 แสดงข้อความคำสั่งที่องค์ประกอบซึ่งใช้ในการจินตทัศน์อัลกอริทึมจัดเรียงข้อมูลใช้ในการทำงาน



รูปที่ 6.4 การส่งข้อความคำสั่งของการจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูล

เมื่อเริ่มการจินตทัศน์องค์ประกอบอัลกอริทึมทั้งสองจะส่งข้อความคำสั่งร้องขอข้อมูลไปยังตัวสร้างข้อมูล (Random) ซึ่ง Random จะตอบสนองด้วยการส่งข้อมูลมาในรูปของแถวลำดับของเลขจำนวนเต็ม (array of integer) หลังจากได้รับข้อมูลแล้ว องค์ประกอบอัลกอริทึมจะเริ่มทำการจัดเรียงข้อมูล โดยในระหว่างทำงานจะ

ส่งข้อความแสดงผลสองข้อความคือ MSG_SWAP ซึ่งแสดงถึงเหตุการณ์การสลับตำแหน่งของข้อมูล และ MSG_FIXPOS เหตุการณ์ที่ข้อมูลได้ย้ายมาอยู่ในตำแหน่งที่ถูกต้องนั้นคือขึ้นข้อมูลนี้จะไม่มีการสลับค่ากับข้อมูลอื่นอีกแล้ว นอกจากข้อความคำสั่งทั้งสององค์ประกอบอัลกอริทึมยังส่งข้อความคำสั่งแสดงผลออกไปอีกสองข้อความคือ MSG_DATAARRAY ที่ใช้ส่งข้อมูลเริ่มต้นและ MSG_FINISH ที่จะส่งออกไปเมื่ออัลกอริทึมจัดเรียงข้อมูลเสร็จเรียบร้อยแล้ว

เมื่อตัวแปลงคำสั่ง (DotConvertor) ได้รับข้อความเหล่านี้ก็จะทำการแปลงค่าโดย MSG_SWAP จะถูกแปลงเป็น MSG_SETDOTVALUE ซึ่งใช้กำหนดค่า(ตำแหน่งในแกน y) ของจุดสองข้อความคำสั่งเนื่องจากการสลับค่าข้อมูลหนึ่งครั้งจะต้องเปลี่ยนค่าของจุดสองจุด ส่วนข้อความ MSG_FIXPOS จะแปลงเป็น MSG_SETDOTCOLOR เพื่อเปลี่ยนสีของจุดให้ต่างจากจุดที่ยังไม่อยู่ในตำแหน่งที่ถูกต้อง

6.3 การจินตทัศน์ที่พัฒนาขึ้นในลักษณะของโปรแกรมเดียว

6.3.1 โครงสร้างของการจินตทัศน์

การจินตทัศน์ที่ทำการสร้างขึ้นเพื่อใช้ในการทดลองเปรียบเทียบการทำงานกับการจินตทัศน์ที่สร้างขึ้นด้วยระบบ AVis นั้น จะใช้โปรแกรมต้นฉบับชุดเดียวกับโปรแกรมต้นฉบับขององค์ประกอบการจินตทัศน์ที่พัฒนาขึ้นแล้ว โดยจะทำการจะลบ AVISCOMP.VBX ออกจากองค์ประกอบทุกองค์ประกอบ แล้วดัดแปลงให้องค์ประกอบแต่ละองค์ประกอบเป็น Form ของวิชวลเบสิกหนึ่ง Form ซึ่งถูกเรียกมาทำงานจาก Form หลักที่จะสร้างขึ้นใหม่ โดย Form หลักที่สร้างขึ้นมาใหม่นี้ จะทำหน้าที่คล้ายกับโปรแกรมควบคุมการจินตทัศน์รวมกับ AVisExecutive ก็คือจะเป็นผู้สั่งให้เริ่มการจินตทัศน์ และทำการจำลองการส่งข้อความคำสั่งควบคุมการทำงานเพื่อควบคุมการทำงานของส่วนประกอบต่างๆ

6.3.2 การติดต่อประสานงานระหว่างส่วนประกอบ

ในการนำเอา Form ต่างๆ มาประกอบรวมกันเป็นการจินตทัศน์ด้วยวิชวลเบสิกรุ่น 3.0 นั้น จะมีปัญหาที่เกิดขึ้นอยู่ประการหนึ่งก็คือ Form ต่างๆไม่สามารถเรียกใช้ฟังก์ชันที่อยู่ในฟอร์มอื่นได้โดยตรง จึงจำเป็นต้องออกแบบวิธีการส่งข้อความคำสั่งขึ้นใหม่เพื่อจำลองการส่งข้อความคำสั่งของระบบ AVis

การส่งข้อความคำสั่งระหว่าง Form จะทำโดยใช้การส่งข้อความคำสั่ง WM_CHAR ของระบบวินโดว์ เมื่อ Form ใดได้รับข้อความคำสั่งนี้วิชวลเบสิกจะสร้างเหตุการณ์ KeyPress ขึ้น โดยค่า KeyAscii ของเหตุการณ์นี้จะเป็นค่าเดียวกับค่า wParam ของฟังก์ชัน SendMessage โดยในการจำลองการส่งข้อความคำสั่งขององค์ประกอบเราจะให้ค่า wParam เป็นค่า 0 เสมอ เพื่อให้ผู้รับสามารถแยกความแตกต่างระหว่างเหตุการณ์ที่เกิดขึ้นเนื่องจากการกดแป้นพิมพ์จริงๆ (ค่า KeyAscii > 0) หรือเหตุการณ์ที่เกิดจากการจำลองการส่งข้อความคำสั่งขององค์ประกอบ

อย่างไรก็ตามการส่งข้อความคำสั่งด้วยวิธีนี้ก็มีข้อเสียคือเราไม่สามารถส่งค่าพารามิเตอร์ไปพร้อมกับข้อความคำสั่งได้ จึงต้องกำหนดตัวแปรแบบ global ขึ้นมากลุ่มหนึ่งเพื่อใช้เก็บค่าพารามิเตอร์และเพื่อให้ไม่เกิดความสับสนในการตั้งชื่อตัวแปรจึงกำหนดให้ตัวแปรทุกตัวที่ใช้เก็บค่าพารามิเตอร์จะต้องมีตัวอักษร P_ (Parameter) นำหน้าเสมอ ในรูปที่ 6.5 แสดงตัวอย่างการจำลองการส่งข้อความคำสั่งแสดงผลให้องค์ประกอบโดยก่อนที่จะทำการเรียกใช้ฟังก์ชัน SendMessage ก็จะต้องนำค่าพารามิเตอร์ต่างๆที่ต้องการส่งจัดเก็บลงในตัวแปร global ต่างๆที่กำหนดไว้แล้วจึงเรียกใช้ฟังก์ชัน SendMessage เพื่อส่งข้อความ

```
rem Sender
Sub OutputNotify(Receiver as Form,msg%,lp1&,lp2&,s$)
    P_EVent = AVIS_WM_OUTPUTNOTIFY
    P_Msg = msg
  P_LParam1 = lp1 :  P_LParam2 = lp2 :P_StrParam = s
    I = SendMessage(Receiver.hWnd, WM_CHAR, 0, 0)
End Function

rem Reeceiver
Sub Form_KeyPress(KeyAscii As Integer)
    If KeyAscii <> 0 Then Exit Sub
    Select Case P_Event
        Case AVIS_WM_OUTPUTNOTIFY
            Call Component1_OutputNotify(0,P_Msg,lp1,lp2,s)
        Case AVIS_WM_INPUTREQUEST
            Call Component1_InputRequest(0,P_Msg,lp1,lp2,s,P_Retval)
        Case AVIS_WM_BEGINSESSION
            Call Component1_BeginSession
    End Select
End Sub
```

รูปที่ 6.5 การติดต่อประสานงานของส่วนประกอบ

6.4 การทดลอง

การทดลองที่นำเสนอในหัวข้อนี้เปรียบเทียบการทำงานระหว่างการจินตทัศน์อัลกอริทึมที่สร้างขึ้นด้วยระบบ AVis กับการทำงานของการจินตทัศน์อัลกอริทึมที่ถูกพัฒนาขึ้นในลักษณะของโปรแกรมเดี่ยว โปรแกรมเดี่ยวทั้งหมดจะสร้างขึ้นโดยการดัดแปลงจากโปรแกรมต้นฉบับที่ใช้ในการพัฒนาองค์ประกอบทั้งนี้เพื่อทำให้สามารถนำการจินตทัศน์ทั้งหมดมาเปรียบเทียบการทำงานได้เที่ยงตรงยิ่งขึ้น การจินตทัศน์ที่จะนำมาใช้ในการทดลองจะมีอยู่สองกลุ่มได้แก่การจินตทัศน์ของอัลกอริทึมต้นไม้ทอดข้ามที่สั้นที่สุดและการจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูล

การจินตทัศน์ทั้งสองกลุ่มจะมีข้อแตกต่างอยู่อีกประการหนึ่งได้แก่การควบคุมการจินตทัศน์ ทั้งนี้เพราะโปรแกรมควบคุมการจินตทัศน์ที่สร้างขึ้นเพื่อใช้กับระบบ AVis มีความซับซ้อนและต้องทำงานรวมกับระบบอย่างใกล้ชิด ทำให้การปรับเปลี่ยนเพื่อนำมาสร้างการจินตทัศน์แบบโปรแกรมเดี่ยวทำได้ยุ่งยาก ทำให้ต้องตัดความสามารถในส่วนนี้ออกไปจากการสร้างการจินตทัศน์แบบโปรแกรมเดียว ดังนั้นในการทดลองจึงให้การจินตทัศน์ทั้งหมดทำงานเพียงรอบเดียวในลักษณะของการทำงานตามปรกติ

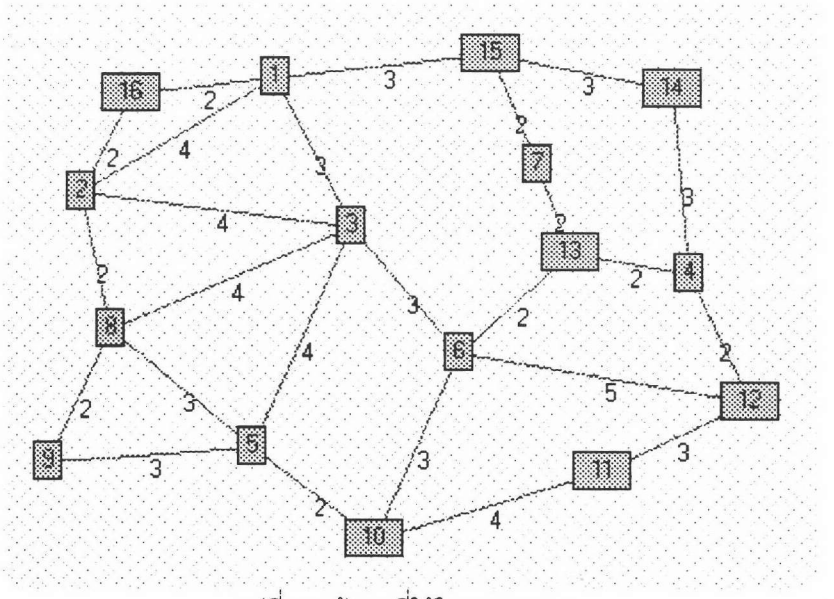
ในการทดลองจะมีการวัดค่าเพื่อเปรียบเทียบการทำงานในแง่ของเวลาที่ใช้ในการทำงาน ซึ่งการวัดค่านี้จะทำโดยการใช้ฟังก์ชัน Timer ของวิชวลเบสิก ซึ่งค่าที่ได้จะเป็นเลขจำนวนจริงที่แสดงเวลาของระบบตั้งแต่ระบบวินโดว์เริ่มทำงานมีหน่วยเป็นวินาที ส่วนการวัดค่าทรัยากรของระบบที่ถูกใช้ในขณะทำการจินตทัศน์จะทำโดยการใช้โปรแกรมซึ่งมีชื่อว่า STRESS.EXE ซึ่งให้มาพร้อมกับตัวแปลภาษาวิชวลซีพลัสพลัส ค่าทรัพยากรของระบบที่โปรแกรมนี้แสดงออกมาจะมีสามค่าคือค่าเปอร์เซ็นต์ของ GDI Heap ที่เหลืออยู่ ค่าเปอร์เซ็นต์ของ User Heap ทีเหลืออยู่ และหน่วยความจำของระบบที่เหลืออยู่มีหน่วยเป็นกิโลไบต์ (KB)

การทดลองที่กล่าวถึงทั้งหมดทำงานบนเครื่องคอมพิวเตอร์ intel 486DX2 - 66 MH_Z ระบบปฏิบัติการที่ใช้คือระบบปฏิบัติการไมโครซอฟต์วินโดว์ฟอร์เวิร์คกรุปรุ่น 3.11 การทดลองที่จะกล่าวถึงในบทนี้จะมีอยู่ทั้งหมดสี่การทดลองได้แก่ (1) การจินตทัศน์ของอัลกอริทึมต้นไม้ทอดข้ามด้วยอัลกอริทึมของพริม (2) การจินตทัศน์ของอัลกอริทึมต้นไม้ทอดข้ามด้วยอัลกอริทึมของครูสกัล (3) การจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลแบบเร็ว และ (4) การจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลแบบฮีป

6.4.1 การจินตทัศน์ของอัลกอริทึมต้นไม้ทอดข้ามด้วยอัลกอริทึมของพริม

การทดลองนี้จะทำการเปรียบเทียบการทำงานของการจินตทัศน์อัลกอริทึมของอัลกอริทึมต้นไม้ทอดข้าม โดยจะแบ่งออกเป็นการจินตทัศน์สี่การจินตทัศน์ การจินตทัศน์แรกจะเป็นการจินตทัศน์ซึ่งพัฒนาขึ้นด้วยระบบ AVis มีได้แก่ AVisPrim การจินตทัศน์ที่สองจะเป็นการจินตทัศน์ที่พัฒนาขึ้นในลักษณะของโปรแกรมเดี่ยว ซึ่งในการทดลองจะเรียกว่า Prim การจินตทัศน์ที่สามและสี่จะคล้ายกับการจินตทัศน์ที่หนึ่งและสองตามลำดับแต่จะตัดส่วนของโปรแกรมที่ใช้ในการแสดงผลแบบกราฟฟิกของส่วนแสดงผลออกไป เพื่อศึกษาถึงผลกระทบของการแสดงผลที่มีต่อความเร็วในการทำงานของการจินตทัศน์

ในการทดลองของการจินตทัศน์สองการจินตทัศน์แรกจะทำการวัดค่าเวลาและทรัพยากรของระบบที่ถูกใช้ ส่วนในการจินตทัศน์ที่สามและสี่นั้นจะทำการวัดเฉพาะเวลาที่ใช้เท่านั้น การวัดค่าจะวัดจากการทำงานของการจินตทัศน์ทั้งสี่การจินตทัศน์เมื่อทำงานกับข้อมูลในรูปที่ 6.6 ซึ่งเป็นกราฟที่มีจุด(node)ทั้งสิ้น 36 จุด และมีเส้นเชื่อม (edge) ทั้งสิ้น 24 เส้น โดยจะทำการทดลองทั้งหมด 10 ครั้ง แล้วนำค่าผลการทำงานที่ได้มาหาค่าเฉลี่ยเพื่อลดผลกระทบของระบบการทำงานร่วมกันแบบไม่ขัดจังหวะของวิชวลเบสิกที่จะมีต่อความเร็วในการทำงานของการจินตทัศน์



รูปที่ 6.6 ข้อมูลที่ใช้ในการทดลอง

6.4.2 การจินตทัศน์ของอัลกอริทึมต้นไม้ทอดข้ามด้วยอัลกอริทึมของครูสกัล

การทดลองนี้จะทำเช่นเดียวกับการทดลองแรกเพียงแต่เปลี่ยนอัลกอริทึมที่ใช้ในการทดลองจากอัลกอริทึมของพริมเป็นอัลกอริทึมของครูสกัลเท่านั้น

6.4.3 การจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลแบบเร็ว

การทดลองนี้จะมีลักษณะคล้ายกับการทดลองที่แล้วเพียงเปลี่ยนจากอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุดมาเป็นอัลกอริทึมจัดเรียงข้อมูลแบบเร็ว (Quick Sort)

ข้อมูลที่จะนำมาใช้ในการทดลองนี้จะเป็นข้อมูลตัวเลขแบบสุ่มจำนวน 300 จำนวน มีค่าอยู่ระหว่าง 1 ถึง 300 ไม่ซ้ำกัน

6.4.4 การจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลแบบฮีป

การทดลองนี้จะมีลักษณะคล้ายกับการทดลองที่แล้วเพียงเปลี่ยนจากอัลกอริทึมอัลกอริทึมจัดเรียงข้อมูลแบบเร็ว มาเป็นอัลกอริทึมจัดเรียงข้อมูลแบบฮีป (Heap Sort)

6.5 ผลการทดลอง

หัวข้อนี้จะกล่าวถึงผลการทดลองของการทดลองจากหัวข้อที่แล้ว ซึ่งแบ่งการทดลองออกเป็นสี่การทดลอง ผลการทดลองจะนำเสนอในลักษณะของตารางแสดงเวลา(มีหน่วยเป็นวินาที) และทรัพยากรของระบบทั้งสามชนิดได้แก่ GDI Heap , USER Heap ซึ่งมีหน่วยเป็นเปอร์เซ็นต์ที่ถูกใช้ และหน่วยความจำของระบบมีหน่วยเป็นกิโลไบต์ ที่ใช้ในการจินตทัศน์ โดยผลการทดลองของการทดลองทั้งสี่จะเป็นดังตาราง

ชื่อการจินตทัศน์	เวลา (วินาที)	GDI Heap ที่ใช้(%)	USER Heap ที่ใช้ (%)	หน่วยความจำ ที่ใช้ (KB)
AVisPrim	2.19	64	22	810.78
Prim	2.14	51	6	296.41
AVisPrim (no graphics)	0.07	-	-	-
Prim (no graphics)	0.05	-	-	-

ตารางที่ 6.1 ผลการจินตทัศน์ของอัลกอริทึมค้นหาต้นไม้ทอดข้ามด้วยอัลกอริทึมของพริม

ชื่อการจินตทัศน์	เวลา (วินาที)	GDI Heap ที่ใช้ (%)	USER Heap ที่ใช้ (%)	หน่วยความจำ ที่ใช้ (KB)
AVisKruskal	3.15	64	22	814.51
Kruskal	3.05	51	6	303.65
AVisKruskal (no graphics)	0.10	-	-	-
Kruskal (no graphics)	0.06	-	-	-

ตารางที่ 6.2 ผลการจินตทัศน์ของอัลกอริทึมค้นหาต้นไม้ทอดข้ามด้วยอัลกอริทึมของครูสกัล

ชื่อ	เวลา (วินาที)	GDI Heap ที่ใช้(%)	USER Heap ที่ใช้ (%)	หน่วยความจำ ที่ใช้ (KB)
AVisQuickSort	6.50	19	20	755.53
QuickSort	5.84	6	3	107.41
AVisQuickSort (no graphics)	2.03	-	-	-
QuickSort (noupdate)	1.57	-	-	-

ตารางที่ 6.3 ผลการจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลแบบเร็ว

ชื่อ	เวลา (วินาที)	GDI Heap ที่ใช้(%)	USER Heap ที่ใช้ (%)	หน่วยความจำ ที่ใช้ (KB)
AVisHeapSort	19.90	19	20	760.03
HeapSort	18.54	6	3	107.90
AVisHeapSort (no graphics)	5.63	-	-	-
HeapSort (no graphics)	4.56	-	-	-

ตารางที่ 6.4 ผลการจินตทัศน์ของอัลกอริทึมจัดเรียงข้อมูลแบบฮีป

6.6 สรุปผลการทดลอง

เมื่อนำผลการทดลองที่ได้ทั้งหมดมาเปรียบเทียบกันในรูปตาราง จะได้ผลดังตารางที่ 6.5

ชื่อการจินตทัศน์	ผลต่างของ เวลา (%)	ผลต่างของ GDI Heap ที่ใช้(%)	ผลต่างของ USER Heap ที่ ใช้ (%)	ผลต่างของ หน่วยความจำ ที่ใช้ (%)
Prim	+2.34	+13	+16	+173.53
Kruskal	+3.28	+13	+16	+175.91
QuickSort	+11.30	+13	+17	+ 603.41
HeapSort	+7.34	+13	+17	+ 604.38

ตารางที่ 6.5 เปรียบเทียบผลการทำงาน

จากตารางจะพบว่าการทำงานของการจินตทัศน์ที่พัฒนาขึ้นด้วยระบบ AVis จะใช้เวลาในการทำงานนานกว่าระบบที่พัฒนาโดยใช้โปรแกรมเดียวประมาณ 2%-11% ซึ่งในงานซึ่งเป็นกลุ่มเป้าหมายหลักในการนำเอาระบบ AVis ไปประยุกต์ใช้งานนั้นคือในงานทางด้านการเรียนการสอนนั้นความเร็วที่ลดลงไปนี้ไม่ถือเป็นข้อเสียที่สำคัญมากนัก

ในแง่ของทรัพยากรของระบบ (GDI Heap , User Heap และ หน่วยความจำ) การจินตทัศน์ที่พัฒนาขึ้นด้วยระบบ AVis ก็ใช้มากกว่าการจินตทัศน์ที่พัฒนาขึ้นโดยในลักษณะของโปรแกรมเดี่ยว ทั้งนี้เนื่องมาจากการที่ระบบ AVis ออกแบบให้แยกองค์ประกอบการจินตทัศน์ และ โปรแกรมควบคุมการจินตทัศน์ออกเป็นโปรแกรมแยกต่างหากจากกัน ทำให้ทรัพยากรโดยรวมของระบบที่ต้องใช้ในการการทำงานขององค์ประกอบเพิ่มขึ้น ซึ่งจากการสร้างโปรแกรมซึ่งประกอบด้วยหน้าต่างว่างๆเพียงหน้าต่างเดียวขึ้นด้วยวิชวลเบสิกแล้วทดลองเรียกใช้หลายครั้งพร้อมๆกัน พบว่าการใช้ทรัพยากรของระบบเป็นดังตารางที่ 6.6

	ครั้งที่หนึ่ง (First Instance)	ครั้งที่สอง (Second Instance)	ครั้งที่สาม (Third Instance)	ครั้งที่สี่ (Forth Instance)
หน่วยความจำ (KB)	98.5	50.75	50.17	52.33
User Heap (%)	0	1	0	0
GDI Heap (%)	3	1	2	2

ตารางที่ 6.6 การใช้ทรัพยากรของระบบของโปรแกรมที่พัฒนาด้วยวิชวลเบสิก

จากตารางพบว่าการเรียกใช้โปรแกรมใหม่หนึ่งโปรแกรมระบบจะสูญเสียหน่วยความจำไประหว่าง 50KB-100KB ขึ้นกับว่าโปรแกมนั้นเคยถูกเรียกใช้มาแล้วหรือไม่ โดยหากถูกเรียกใช้มาแล้วระบบจะใช้หน่วย

ความจำน้อยกว่าเพราะไม่ต้องจองหน่วยความจำสำหรับรหัสโปรแกม (code segment) ใหม่ จากตารางจะพบว่าค่าทรัพยากรส่วนอื่นๆทั้งหมดของระบบที่ถูกใช้ไปมีค่าเป็นศูนย์ การที่ค่าเป็นศูนย์ไม่ใช่เป็นการแสดงว่าไม่ถูกใช้เลยเพียงแต่แสดงว่าค่าที่ใช้ไปมีค่าน้อยมากจนเมื่อนำมาคำนวนและแสดงผลเป็นเปอร์เซ็นต์โดยโปรแกรม STRESS.EXE จะได้ค่าเป็นศูนย์

ในระบบ AVis นอกจากจะสิ้นเปลืองหน่วยความจำเนื่องจากการแยกองค์ประกอบออกเป็นส่วนประกอบย่อยๆแล้ว ระบบยังต้องใช้ทรัพยากรส่วนหนึ่งสำหรับการทำงานของโปรแกรมควบคุมการจินตทัศน์ด้วย ตารางที่ 6.7 แสดงการใช้ทรัพยากรของโปรแกรมควบคุมการจินตทัศน์ซึ่งพัฒนาขึ้นด้วยวิชวลเบสิก ที่พัฒนาขึ้นใช้กับการวิจัยนี้

หน่วยความจำ (KB)	User Heap (%)	GDI Heap (%)
300	4	8

ตารางที่ 6.7 การใช้ทรัพยากรของ AVisController

ดังนั้นจะพบว่าการจินตทัศน์ที่ใช้ในการทดลองซึ่งมีสี่องค์ประกอบจะใช้หน่วยตวามจำมากกว่าการจินตทัศน์แบบโปรแกรมเดียวอย่างน้อย 450 KB (300 + 3* 50)

จากการทดลองยังพบอีกว่าขั้นตอนการทำงานของการจินตทัศน์อัลกอริทึมที่มีผลกระทบต่อการทำงานโดยรวมของระบบมากที่สุดก็คือขั้นตอนของการแสดงผลของส่วนแสดงผล ดังแสดงไว้ในตารางที่ 6.8

ชื่อ	อัตราส่วนของเวลา(%)
AVisPrim	96.80%
Prim	97.66%
AVisKruskal	96.82%
Kruskal	98.03%
AVisQuickSort	68.77%
QuickSort	73.11%
AVisHeap	71.71%
HeapSort	75.40%

ตารางที่ 6.8 อัตราส่วนของเวลาที่ใช้ในการทำงานทั้งหมดและเวลาที่ใช้ในการแสดงผล

การแสดงผลการทำงานจะใช้เวลามากกว่า 60% ของเวลาในการทำงานทั้งหมดของการจินตทัศน์ขึ้นกับความซับซ้อนของการแสดงผลการทำงาน ดังนั้นหากต้องการเพิ่มความเร็วในการทำงานของการจินตทัศน์ จุดแรกที่ควรจะปรับปรุงคือการเพิ่มความเร็วของการแสดงผล ซึ่งการปรับแก้นี้จะให้ผลที่ดีกว่าการปรับแก้การทำงานของส่วนอื่นๆ

6.7 สรุป

ในบทนี้ได้กล่าวถึงการนำระบบ AVis ไปใช้ในการสร้างการจินตทัศน์ของอัลกอริทึมค้นหาต้นไม้ทอดข้ามที่สั้นที่สุด และอัลกอริทึมจัดเรียงข้อมูลในหน่วยความจำ เพื่อทดสอบประสิทธิภาพของการจินตทัศน์ที่พัฒนาด้วยระบบ AVis และการจินตทัศน์ที่พัฒนาขึ้นในลักษณะของโปรแกรมเดียว ซึ่งจากการทดลองนำการจินตทัศน์ทั้งสองแบบมาทำงานเปรียบเทียบกันพบว่าการจินตทัศน์ที่พัฒนาขึ้นโดยใช้ระบบ AVis ทำงานช้ากว่าประมาณ 2%-10% และใช้ทรัพยากร GDI และ User Heap มากกว่า 13%-17% ส่วนในกรณีของหน่วยความจำที่ใช้มาก

กว่าถึง 600% นั้นก็เนื่องมาจากในการจินตทัศน์หนึ่งๆต้องใช้หน่วยความจำในจำนวนที่คงที่ค่าหนึ่ง (300 + n*50 เมื่อ n คือจำนวนองค์ประกอบ) ซึ่งเมื่อนำมาคำนวนเป็นเปอร์เซ็นก็อาจทำให้ค่าที่ได้มีค่ามากได้

อย่างไรก็ตามเมื่อนำข้อเสียนี้มาเปรียบเทียบกับความสะดวกในการพัฒนาและความสามารถของการจินตทัศน์ที่เพิ่มขึ้นแล้ว จะพบว่าการพัฒนาการจินตทัศน์อัลกอริทึมด้วยระบบ AVis มีประสิทธิภาพดีกว่าการพัฒนาการจินตทัศน์ด้วยตนเองทั้งหมดเมื่อเทียบกันในแง่ของความสะดวกและเวลาที่ใช้ในการพัฒนา

นอกจากนี้จากการทดลองทำให้เราได้ข้อมูลเพิ่มเติมว่าเวลาที่ใช้ในการทำงานของการจินตทัศน์อัลกอริทึมส่วนใหญ่ (68%-98%)จะเป็นเวลาที่ใช้ในการแสดงผล หากการแสดงผลมีความซับซ้อนอัตราส่วนของเวลาที่ใช้ก็จะเพิ่มขึ้นด้วย ดังนั้นการทำงานของระบบ AVis จึงไม่ใช่จุดหลักที่มีผลต่อความเร็วในการทำงานของการจินตทัศน์