การวาดภาพหน้าคนร้ายโดยระบบคอมพิวเตอร์

Computer Assisted Suspect Sketching Outfit

โดย

นาย ตรัสพงศ์ ไทยอุปถัมภ์

นาย ปรากฏ ลิมปกาญจน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2535

ปริญญานิพนธ์ปีการศึกษา 2535

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การวาดภาพหน้าคนร้ายโดยระบบคอมพิวเตอร์

ผู้จัดทำ

1. นาย ตรัสพงศ์ ไทยอุปถัมภ์ 321100 4D
2. นาย ปรากฏ ลิมปกาญจน์ 321177 4D

_____อาจารย์ที่ปรึกษา

(ดร. เอื้อน ปิ่นเงิน)

(____ /____ /____)

การวาดภาพหน้าคนร้ายโดยระบบคอมพิวเตอร์

นาย ตรัสพงศ์ ไทยอุปถัมภ์ 321100
นาย ปรากฏ ลิมปกาญจน์ 321177
อาจารย์ที่ปรึกษา :
ดร. เอื้อน ปิ่นเงิน

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์ในการที่จะประยุกต์ใช้คอมพิวเตอร์ให้เกิดประโยชน์ โดยทำการพัฒนาต้นแบบการวาดภาพคนร้ายโดยระบบคอมพิวเตอร์ หลักการง่ายๆของระบบนี้ก็คือ การเก็บรวบรวมชิ้นส่วนต่างๆที่มาประกอบเป็นใบหน้าคนให้อยู่ในรูปของฐานข้อมูลภาพ ในการนำมาใช้ก็จะดึงภาพจากฐานข้อมูลทีละชิ้นส่วนนำมาประกอบกันจนเป็นภาพหน้าที่สมบูรณ์ เนื่องจากแต่ละชิ้นส่วนของใบหน้านั้นที่พบเห็นจะมีไม่มากแบบเท่าใดนัก สิ่งที่ทำให้ใบหน้าแตกต่างกันจึงเป็นเรื่องของสัดส่วนและขนาด ระบบจึงจะต้องสามารถทำการปรับเปลี่ยนชิ้นส่วนได้ตามสมควร ในการพัฒนาโครงงานนี้จะทำบนสภาวะแวดล้อมของระบบวินโดวส์ 3.1 โดยผ่านการวิเคราะห์แล้วว่าการพัฒนาบนระบบนี้จะสามารถพัฒนาโครงงานได้สะดวกรวดเร็วและเสร็จทันกำหนดเวลา

COMPUTER ASSISTED SUSPECT SKETCHING OUTFIT

Trassapong Thaiupathump 321100
Prakot Limpakan 321177
Advisor :
Dr. Ouen Pin-ngern

ABSTRACT

The main objective of the project is aimed on the application of the computer in drawing a sketch of a suspect face by developing a prototype for this purpose.

The basic principles are to collect the components of faces in form of picture database and the application of these database in sketching face pictures.

In application these pictures will be selected from the database and pasted on the screen until the face of the suspect is completed or satisfied.

Since there are not so many different types in each component, the sigificant characteristics are the propotion and size of it. Therefore, the system is developed so that the modification on proportion and size of the component could be made properly.

This project is developed under the WINDOWS 3.1 environment since it has been carefully and analytically determined that the system couldbe used to finish the project conveniently in good time.

กิตติกรรมประกาศ

ในการทำโครงงานให้สำเร็จลุล่วงได้ในครั้งนี้ได้รับความช่วยเหลือเอื้อเฟื้อจากบุคคลหลายท่านจึงขอขอบคุณไว้ ณ ที่นี้

ศ.ดร. ศรีศักดิ์ จามรมาน
อ. วัชระ ฉัตรวิริยะ
อ. อภิเนตร อุนากูล
พ.ต.อ. ชาตรี สุนทรศร
ร.ต.อ. ธนารักษ์ เกิดผล
น.ส. ชนกรัฐ ยาคุ้มภัย
คงเดช กรกาญจนารักษ์
มานพ มานะศิลป์
ศวิยา ชโนวรรณะ
ศิริกุล วิบุลสันติ
พี่บุ้ง ที่ปู อ๊อฟ เจี๊ยบ ดี๋ เค
และคุณพ่อ - คุณแม่

สารบัญ

บทนำ

ทุกวันนี้คอมพิวเตอร์เริ่มเข้ามามีบทบาทที่สำคัญในชีวิตประจำวันของคนเรามากขึ้นมีการนำเอาคอมพิวเตอร์มาใช้ช่วยเหลือในการทำงานกันอย่างกว้างขวาง นอกจากนี้แล้วในปัจจุบันได้มีการนำเอาเทคโนโลยีทางด้านคอมพิวเตอร์กราฟิก มาประยุกต์ใช้ในงานทางด้านการประมวล ภาพต่าง ๆ เช่น ภาพหน้าคน หรือแม้กระทั่งอวัยวะภายในของคนได้โดยการถ่ายภาพ การฉายรังสี การใช้กล้องจุลทรรศน์อิเลคตรอน เข้าไปเก็บไว้ในคอมพิวเตอร์ เพื่อใช้งานทางด้านการแพทย์ ไม่ว่าจะเป็นการทำศัลยกรรมตกแต่ง หรือการผ่าตัดที่มีความเสี่ยงสูง เช่น การผ่าตัดสมอง หรือ จะเป็นโปรแกรมที่ทำงานที่เกี่ยวกับงานของตำรวจ เช่น โปรแกรมที่สามารถทำการคาดคะเนหน้าตาของเด็กที่หายไปหลายปีว่าเมื่อโตขึ้นจะมีหน้าตาเป็นอย่างไรและสามารถทำการค้นหาจนพบ โปรแกรมต่าง ๆ เหล่านี้ล้วนแล้วแต่ถูกพัฒนาโดยชาวต่างชาติโดยเกิดจากความร่วมมือหลายฝ่าย ใช้ทีมงานจำนวนมากและใช้เงินทุนในการวิจัยอย่างมหาศาล

ในเมืองไทยในปัจจุบันนี้ เรามีศักยภาพอย่างพอเพียงที่จะทำการเริ่มการพัฒนาผลงานขึ้นมาใช้กันเอง จึงควรมีความตื่นตัวมากขึ้น ในงานบางประเภทนั้น คนไทยสามารถสร้างผลงานที่มีประสิทธิภาพสูงและเหมาะสมกับการใช้งานในเมืองไทยมากกว่าผลิตภัณฑ์ของต่างประเทศอีกด้วย ในเมื่อเราสามารถทำงานพัฒนาผลงานขึ้นมาใช้เองเราก็ไม่จำเป็นต้องซื้อผลิตภัณฑ์จากต่างประเทศ ที่มีราคาแพงมากมาใช้ เพราะฉะนั้นจึงเป็นเหตุผลที่ดีที่ควรจะมีการเริ่มทำการพัฒนางานขึ้นมาใช้กัน ในการทำโครงงานการใช้คอมพิวเตอร์ช่วยในการวาดภาพคนร้ายนี้ เนื่องจากมีผู้ร่วมงานเพียงสองคนเท่านั้น และประกอบกับไม่มีหนังสือที่สามารถอ้างอิงได้โดยตรง ขอบเขตของงานก็จะเป็นในลักษณะการศึกษาค้นคว้าเกี่ยวกับข้อมูลทางด้านนี้ และทำการสร้างต้นแบบของระบบขึ้นมาให้เหมาะสมกับการใช้งานจริงมากที่สุด ผลงานอาจยังไม่สมบูรณ์พอสำหรับการใช้งานจริงทุกประการ แต่ทางผู้จัดทำก็พยายามที่จะทำผลงานให้ดีที่สุด และหวังเป็นอย่างยิ่งว่าจะเป็นความพยายามที่จะใช้เทคโนโลยีที่มีอยู่ในปัจจุบันก่อให้เกิดประโยชน์ต่อสังคมมากที่สุด

วัตถุประสงค์ของโครงงาน

- เพื่อศึกษาการนำระบบคอมพิวเตอร์มาประยุกต์ใช้ในการประกอบภาพใบหน้าคนร้าย
- เพื่อศึกษาทฤษฎีทางคอมพิวเตอร์กราฟิกต่าง ๆ เพื่อหาแนวทางในการพัฒนาการนำเอาระบบคอมพิวเตอร์มาใช้ในการประกอบภาพใบหน้าคนร้าย
- เพื่อนำทฤษฎีต่าง ๆ ทางคอมพิวเตอร์กราฟิกที่ได้ศึกษามาเพื่อทำการปรับปรุงและพัฒนาสร้างระบบต้นแบบในการนำเอาระบบคอมพิวเตอร์มาใช้ช่วยในการประกอบและวาดภาพใบหน้าคนร้าย

แผนการทำงาน

ในโครงงานขั้นนี้จุดประสงค์ในขั้นต้นคือการนำเอาคอมพิวเตอร์ไปประยุกต์ใช้ในงานการประกอบภาพโครงร่างของใบหน้าคนร้าย ในงานของการสืบสวนสอบสวนของกรมตำรวจ และการทำงานของโครงงาน สามารถแบ่งเป็นขั้นตอนต่าง ๆ ได้ดังนี้คือ

ศึกษาความเป็นไปได้และกำหนดขอบเขตของโครงงานเบื้องต้นจากแหล่งข้อมูลต่าง ๆ

การเริ่มต้นของการดำเนินงานของโครงงานนี้ เริ่มจากการศึกษาข้อมูลต่าง ๆ ที่จำเป็นในการกำหนดขอบข่าย เป้าหมายของงาน การได้มาซึ่งข้อมูล ซึ่งได้มาจากทั้งการไปทำการศึกษาจากผลงานที่กำลังทดลองใช้จริงที่กองทะเบียนประวัติอาชญากรรม กรมตำรวจ และผลงานซึ่งอยู่ในแนวทางที่ใกล้เคียงกันนั้นจากสถาบันการศึกษา และหน่วยงานราชการหลาย ๆ แห่ง ประกอบกับคำปรึกษาด้านแนวทางของผลงานที่จะออกมา จากบริษัท R&D อาจารย์ที่ปรึกษา และอาจารย์ท่านอื่นๆ นอกจากนั้นยังได้ศึกษาจากตำราทางด้านกายวิภาคศรีระรูปร่างลักษณะ และกล้ามเนื้อของส่วนประกอบต่าง ๆ ของส่วนหัว และใบหน้าทั้งในแนวทางของศิลปะและทางการแพทย์ได้ทำให้รูปแบบของโครงงานเริ่มที่จะชัดเจนยิ่งขึ้น

การกำหนดจุดประสงค์ของโครงงานนี้ หากไม่กำหนดลงไปในรายละเอียดแล้ว จะเห็นได้ในทันทีว่ามีมากมายหลายวิธีหลายแนวทาง และหลายแนวความคิด สุดแล้วแต่ความเหมาะสมทั้งทางด้าน การนำเทคโนโลยีมาใช้ให้เกิดประโยชน์ ความจำเป็น ความต้องการ งบประมาณ และบุคคลากร เช่น ในการใช้งานของ กรมตำรวจของประเทศญี่ปุ่น ที่มีความพร้อมในทุก ๆ ด้าน ทั้งทางด้าน เทคโนโลยี บุคคลากร และงบประมาณ ได้มีการนำมาใช้งานได้จริงอย่างมีประสิทธิภาพมานานนับสิบปี เช่นเดียวกันกับประเทศสหรัฐอเมริกา ผู้บุกเบิกยุค สารสนเทศ ได้มีการนำไปใช้งานจริงมานานนับสิบปีแล้วเช่นเดียวกัน อีกทั้งมีความสามารถในการเชื่อมโยงกับหน่วยงานต่าง ๆ เช่น สำนักงานสถิติ โรงพยาบาล เทศบาล ธนาคาร สถานีโทรทัศน์ฯลฯ เพราะทุกอย่างก็เป็นคอมพิวเตอร์หมดแล้ว ทำให้สามารถทำงานได้อย่างมีประสิทธิภาพมากยิ่งขึ้น และเนื่องจากในช่วงระยะ 2-3 ปีที่ผ่านมาสภาวะการณ์ของบ้านเมืองเรา เริ่มที่จะเอื้ออำนวยต่อการพัฒนาเทคโนโลยีในด้านต่าง ๆ ให้รุดหน้า คอมพิวเตอร์เริ่มจะมีบทบาทมากขึ้นในสังคมไทย กรมตำรวจไทยก็มีความพร้อม และศักยภาพที่จะนำคอมพิวเตอร์ มาช่วยในงานด้านต่าง ๆ โดยเฉพาะอย่างยิ่ง งานการสเก็ต ภาพหน้าคนร้าย ก็เป็นงานด้านหนึ่งที่ข้อมูลจากการศึกษาแสดงให้เห็นถึง การทำงานที่มีประสิทธิภาพมากขึ้นอย่างชัดเจนเมื่อมีการนำเอาคอมพิวเตอร์มาช่วย แต่เนื่องจากอยู่ในช่วง เริ่มบุกเบิก และ กรมตำรวจไม่มีงบประมาณในการซื้อระบบจากต่างประเทศ จึงต้องอาศัยผลงานจากภายในประเทศซึ่งความสามารถต่าง ๆ ในระยะแรกอาจจะไม่เพียบพร้อมเท่า ผลงานที่ใช้กันอยู่ในต่างประเทศ ประเทศก็ตาม แต่ก็ถือเป็นการเริ่มต้นความพยายามที่จะนำ เทคโนโลยีใหม่ๆ มาปรับปรุงการทำงานให้มีประสิทธิภาพยิ่งขึ้น พร้อมทั้งจะเป็นกระตุ้นและส่งเสริมให้คนในชาติ สร้างสรรค์ผลงานต่าง ๆ ขึ้นมาใช้กันให้มากขึ้น ในด้านความสามารถในช่วงเริ่มต้นนี้ให้มีความสามารถที่เพียงพอกับความต้องการ ขั้นต้นในปัจจุบันก่อน แล้วจึงทำการปรับปรุงเปลี่ยนแปลงให้ดีขึ้นในภายหน้าต่อไป เมื่อมีความพร้อมในด้านต่างๆ มาก กว่าที่เป็นอยู่ในปัจจุบัน

ผลงานในการนำเอาคอมพิวเตอร์มาช่วยในการวาดภาพคนร้ายที่มีการใช้จริงที่ กรม ตำรวจ เป็นการใช้ข้อมูลของส่วนประกอบของใบหน้าส่วนต่าง ๆ ที่ได้จากการ Scan ภาพที่วาดไว้ก่อนด้วยมือเก็บไว้ในหน่วยเก็บข้อมูลที่มีความจุค่อนข้างสูง ลักษณะจะคล้ายกับการนำแผ่นใสของส่วนประกอบต่าง ๆ มาวางซ้อน ๆ กัน ประกอบกันเป็นรูปหน้าคนที่สมบูรณ์ ที่เป็นวิธีดั้งเดิมของตำรวจ ก่อนนำคอมพิวเตอร์มาใช้นั่นเอง วิธีการนี้จะมีข้อดีตรงที่รวดเร็ว ไม่ซับซ้อน ส่วนเรื่องคุณภาพของภาพที่ได้มาในขั้นตอนสุดท้ายนั้นอยู่ในเกณฑ์ที่ดีมาก เพราะการทำงานบนเครื่องคอมพิวเตอร์และอุปกรณ์ต่อพ่วง(เช่น Laser Printer, Scanner) ที่ออกแบบมาเพื่อการทำงานทางด้านกราฟิกโดยเฉพาะอย่าง Apple Macintosh ที่มีศักยภาพการทำงานทางด้านกราฟิก ที่สูงมากเมื่อเทียบกับคอมพิวเตอร์ในตระกูล IBM PC โดยทั่วไป แต่ถ้าหากเครื่องในตระกูล PC นี้มีการใช้อุปกรณ์ที่ช่วย เพิ่มประสิทธิภาพทางด้านกราฟิก ให้มีความสามารถสูงกว่า PC ปกติโดยทั่วไป คุณภาพของงานที่ออกมาเมื่อเปรียบเทียบ ในแง่ของ Resolution แล้วก็อยู่ในขั้นที่พอจะยอมรับได้ และถ้าจะมองว่าความสามารถของอุปกรณ์ต่างๆ เป็นการพัฒนาทางด้านฮาร์ดแวร์แล้ว ประกอบกับการพัฒนาทางด้านฮาร์ดแวร์ของอุปกรณ์ต่อพ่วงของเครื่องคอมพิวเตอร์ในตระกูลต่างๆ ไม่ต่างกันมากนักจึงหันเหแนวทางการพัฒนาทางด้านซอฟท์แวร์ รูปแบบของแนวคิดใหม่ในการปรับปรุงจุดอื่นๆ ที่ทำให้ผลงานทั้งหมด โดยรวมพัฒนาขึ้น โดยที่แนวคิดนี้ควรสามารถนำไปใช้ในเครื่องมาตรฐานอื่นๆ ได้ ในการดำเนินการในโครงงานนี้จึงตัดสินใจ ที่จะทำการพัฒนาบนเครื่องคอมพิวเตอร์ใน ตระกูล IBM PC เมื่อทำการศึกษาข้อมูลที่จำเป็นเบื้องต้นแล้ว สามารถกำหนดขอบเขต และคุณลักษณะ ของผลงานขั้นสุดท้ายของโครงงานได้ดังนี้คือ

ในการดำเนินการในโครงงานนี้จะทำการพัฒนา บนเครื่องคอมพิวเตอร์ในมาตรฐาน ของ IBM PC Compatible ที่มีใช้กันอย่างแพร่หลายในปัจจุบัน และการพัฒนาผลงานจะทำการดำเนินการพัฒนาในสภาวะแวดล้อมของ Microsoft WINDOWS 3.1 ที่มีความสามารถจัดการและควบคุม resource ต่าง ๆ ของระบบได้อย่างมีประสิทธิภาพและนอกจากนั้น Windows ยังมีการติดต่อกับผู้ใช้แบบ Graphical User Interface(GUI) ที่ง่ายต่อการใช้งาน และมีแนวโน้มที่จะได้รับความนิยมมากขึ้นในอนาคตอันใกล้ เมื่อพัฒนาในระบบ Windows แล้ว ความต้องการทางด้านฮาร์ดแวร์จึงต้องเป็นไปตามความต้องการของระบบ Windows ไปโดยปริยาย เช่น จะต้องมีอุปกรณ์สำหรับเก็บข้อมูลที่มีความจุสูง เช่น ฮาร์ดดิสค์ และควรจะมีอุปกรณ์สำหรับการกำหนดตำแหน่ง เช่น เมาส์ เป็นต้น โดยจะมีการเพิ่มความละเอียดของการ ใช้อุปกรณ์ที่ช่วยเพิ่มประสิทธิภาพทางด้าน กราฟิกให้มีความสามารถสูงกว่า PC ปกติโดยทั่วไป คือ จะใช้การอุปกรณ์ในการแสดงผล แสดงผลภาพในแบบ SUPER VGA 800 x 600 จุด เป็นอย่างต่ำ

การกำหนดในขั้นตอนต่อไปหลังจากได้กำหนดชนิดของคอมพิวเตอร์ และกำหนดการพัฒนาในระบบ Windows ไปแล้วนั้นก็คือการกำหนดคุณลักษณะเบื้องต้นของผลงานและแนวทางในการพัฒนาซึ่งจะกำหนดได้ดังนี้คือ จะนำแนวคิดเดิมที่มีอยู่มาปรับปรุงให้ดีขึ้นหรือการนำแนวความคิดเดิมมาใช้ในรูปแบบใหม่ โดยจะมีการผสมผสานกัน ของแนวทางต่าง ๆ เพื่อให้ได้ผลงานอย่างสมบูรณ์ที่สุด โดยได้มีการศึกษาและทดลองในแนวทางต่างๆ เช่น แนวคิดทางด้านการกำหนดรหัสประจำใบหน้าที่สามารถจะทำการสร้างภาพใบหน้าขึ้นมาใหม่จากรหัสนั้นโดยเครื่องคอมพิวเตอร์ทั่วไป แนวคิดที่ให้คอมพิวเตอร์สามารถสร้างรูปแบบของ ส่วนประกอบของใบหน้าได้อย่างไม่จำกัด จากข้อมูลในฐานข้อมูลที่จำกัด แนวคิดในการทดลองเชื่อมต่อกับแฟ้มประวัติข้อมูลอย่างง่ายการกำหนดรหัส และรูปแบบการค้นหาประวัติจากฐานข้อมูล ฯลฯ นอกเหนือไปจากการปรับปรุงรูปแบบการใช้งานให้เหมาะสมยิ่งขึ้น อย่างเช่นที่ได้กล่าวมาแล้วตั้งแต่ตอนต้นแล้วว่าแนวคิดต่างๆ นั้นมีมากมาย ถ้าจะทำกันจริง ๆ ทั้งหมดคงต้องใช้ทรัพยากรต่าง ๆ จำนวนมากโดยเฉพาะทรัพยากรบุคคล ฉะนั้นเพื่อความเหมาะสมจึงเลือกเฉพาะบางแนวความคิดที่น่าจะเป็นไปได้เท่านั้น มาดำเนินการสร้างในโครงงานนี้

กำหนดขอบเขตของโครงงานโดยละเอียด

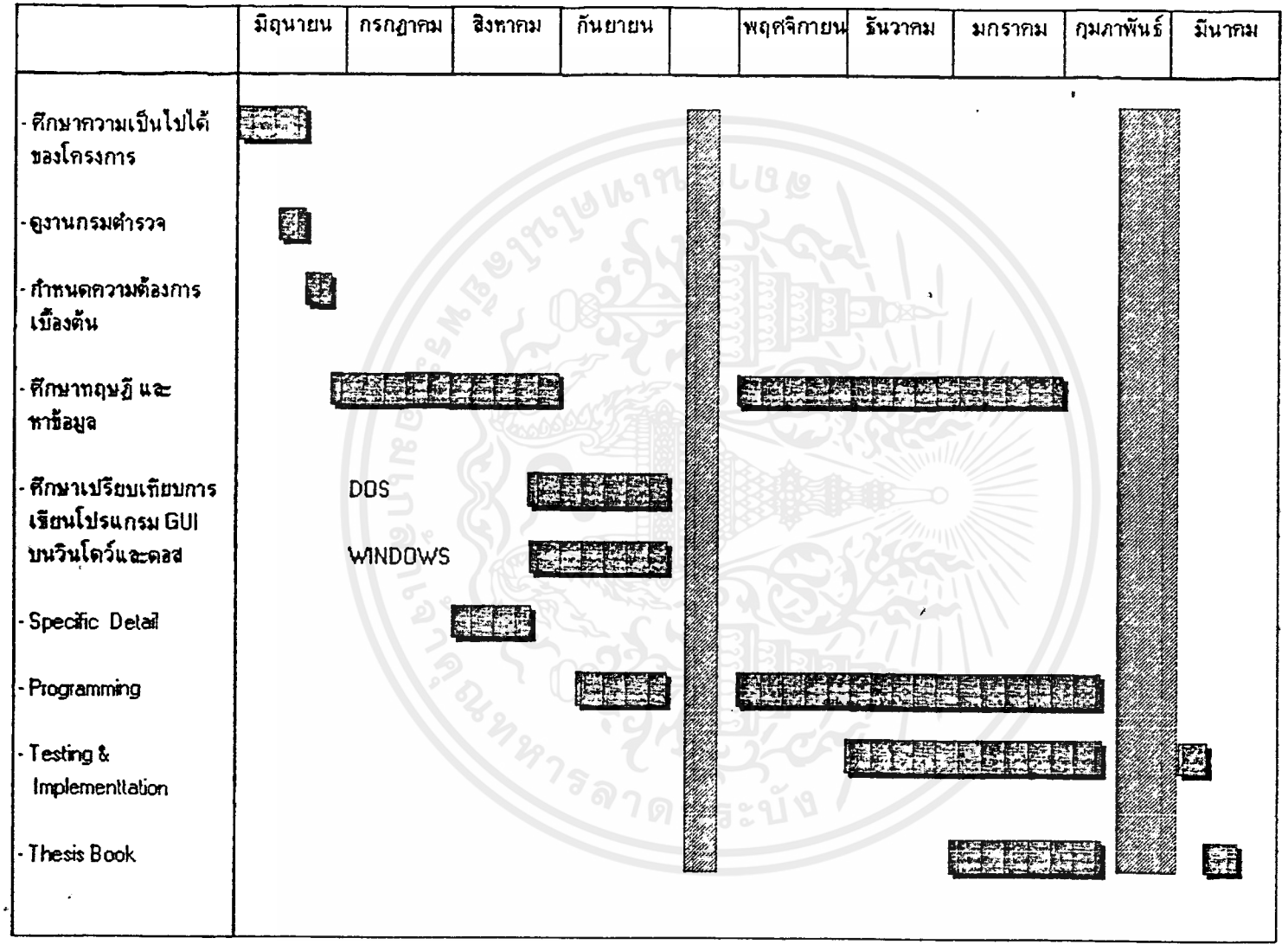
หลังจากที่ได้กำหนดคุณลักษณะภายนอก หรือสภาวะแวดล้อมของโครงงานแล้ว ต่อไปก็จะเป็นการกำหนดคุณลักษณะภายในของระบบ ในโครงงานนี้จะมีฐานข้อมูลภาพวาดส่วนประกอบของใบหน้าคน ซึ่งจะใช้วิธีวาดจากภาพถ่ายคนจริง แล้วใช้เครื่องอ่านภาพ (scanner) ป้อนเข้าเครื่องคอมพิวเตอร์เก็บไว้เป็นฐานข้อมูล ส่วนประกอบต่าง ๆ ของใบหน้าคนจะทำการเก็บแยกส่วนกันดังนี้คือ ทรงผม รูปหน้าและคาง ใบหู คิ้ว ตา จมูก ปาก และอื่นๆ เช่น หนวด และแว่นตา เป็นต้น ในระบบนี้จะช่วยให้พยานหรือเจ้าทุกข์และเจ้าหน้าที่ผู้วาดภาพสามารถสร้างภาพคนร้ายได้ด้วยเวลาอันรวดเร็ว ด้วยความสามารถของระบบดังนี้

1. ผู้ใช้สามารถเรียกชิ้นส่วนใบหน้ามาเลือกและประกอบบนจอภาพได้อย่างรวดเร็ว
2. แต่ละชิ้นส่วนสามารถปรับเปลี่ยนรูปลักษณะช่วยให้เห็นข้อแตกต่างกันได้
3. ขยับปรับตำแหน่งของแต่ละชิ้นส่วนได้ในระยะของใบหน้าคนปกติ
4. สามารถทำการปรับขนาดของแต่ละชิ้นส่วน โดยบีบหรือขยายภาพแต่ละชิ้นส่วนให้กว้าง แคบ หนา และบางได้ตามต้องการ
5. แต่งเติมรายละเอียด และปรับปรุงคุณภาพของภาพ โดยใช้ทฤษฎีการประมวลผลภาพ (Image Processing)
6. ภาพที่สร้างได้สามารถทำการบันทึกไว้ในแฟ้มข้อมูลประวัติอาชญากร ในลักษณะที่เป็นรหัสของส่วนประกอบต่างๆ หรือจะเป็นแบบ Bitmap ก็ได้พร้อมทั้งรายละเอียดเพิ่มเติมเพื่อความสะดวกในการค้นหาเรียกใช้ในภายหลัง

ในแต่ละขั้นตอน ได้มีการออกแบบการติดต่อกับผู้ใช้ที่ง่าย และเหมาะสมกับการใช้งานจริง โดยคำปรึกษาของทางกรมตำรวจโดยตรง เพื่อให้ผลงานที่ได้มีความสมบูรณ์มากที่สุดเท่าที่จะเป็นไปได้

การเขียนโปรแกรม

โครงงานชิ้นนี้เป็นโปรแกรมที่ทำงานในสภาวะแวดล้อม Windows จึงต้องใช้โปรแกรมสำหรับการสร้าง แอพลิเคชั่นบนวินโดวส์ ในโครงงานนี้ใช้ภาษา C เป็นภาษาในการพัฒนา โดยเลือกใช้โปรแกรม Borland C++ 3.0 & Application Framework และ Quick C for Windows ในการพัฒนา



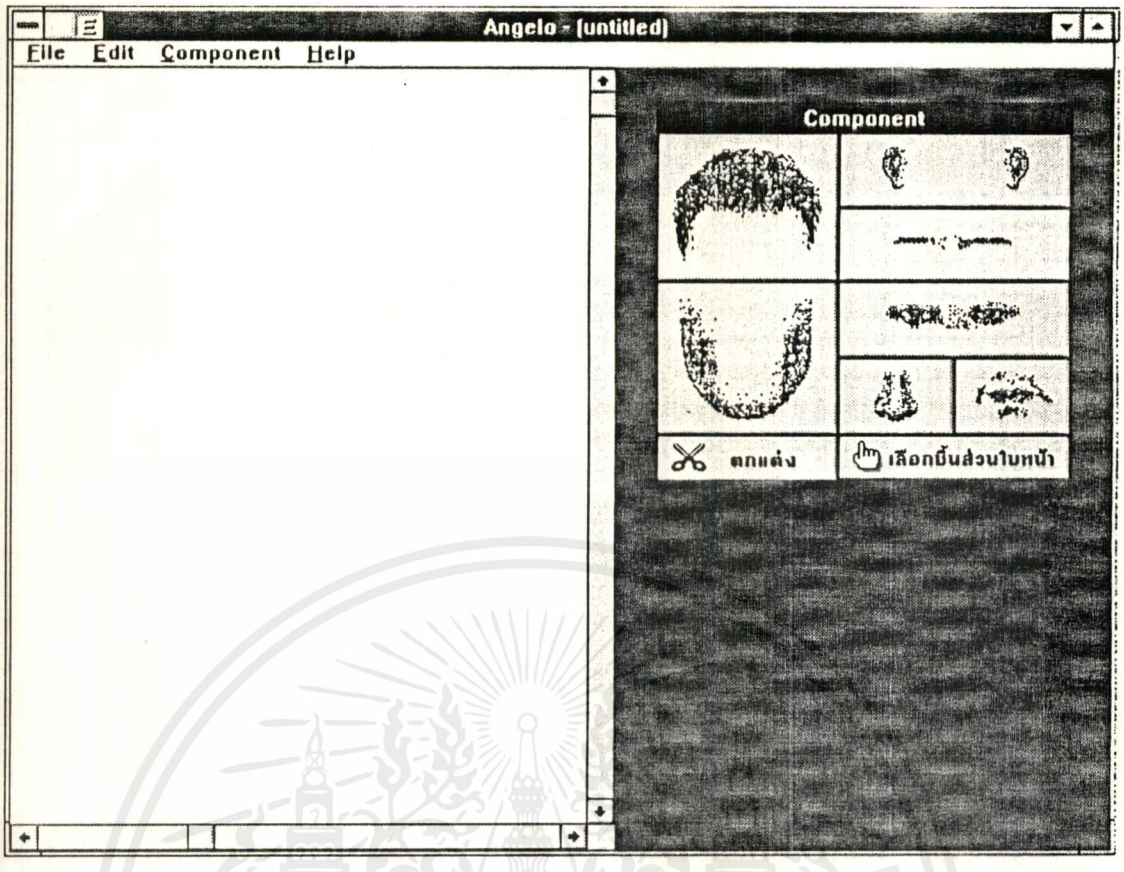
ตารางการทำงาน

ANGELO PROJECT

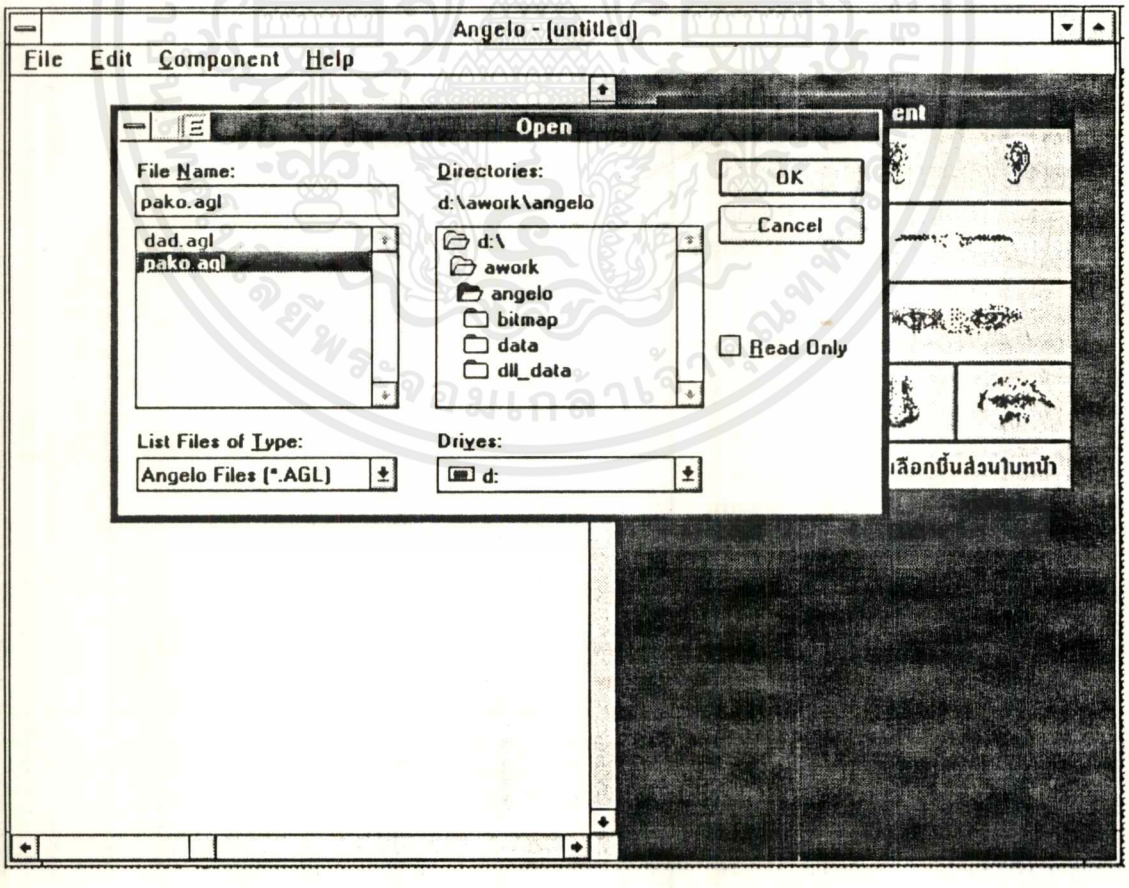
ผลงาน

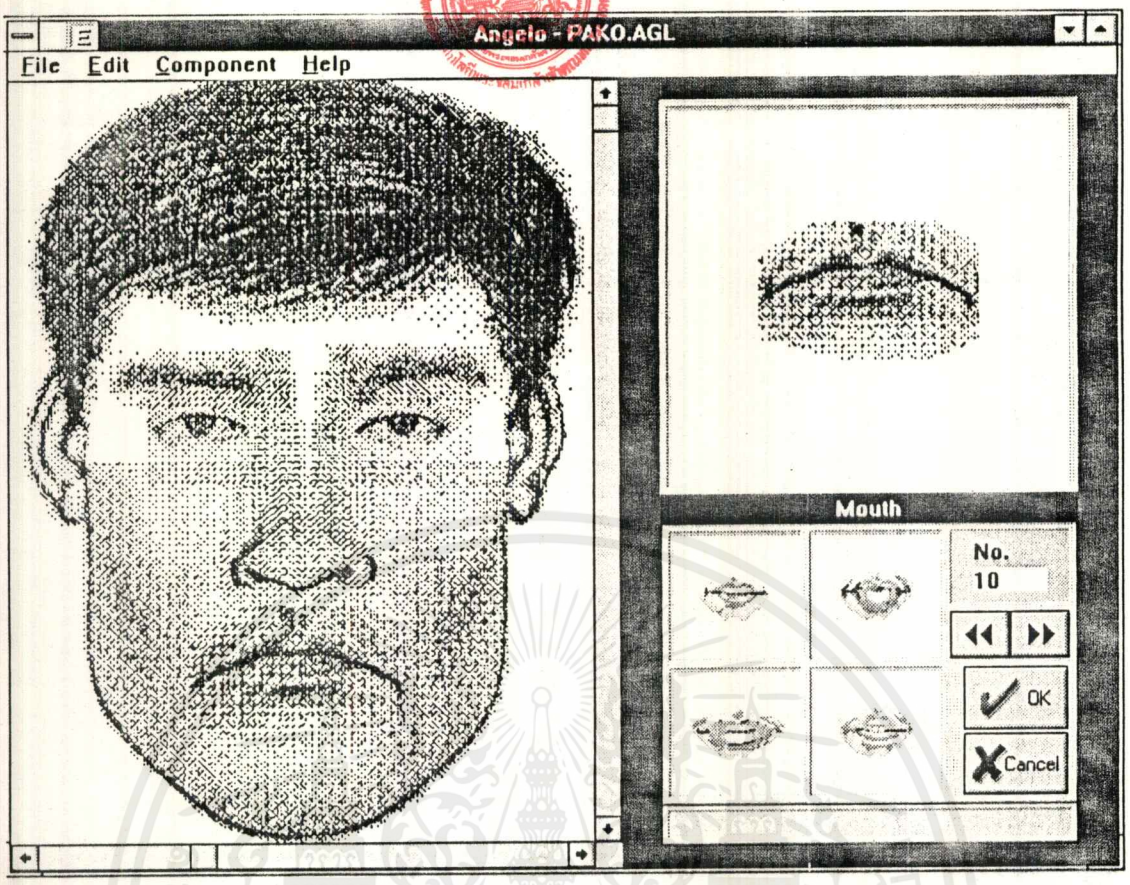
โครงงานมีคุณลักษณะความสามารถครบสมบูรณ์ตามตามความต้องการ ที่ได้กำหนดไว้ในขั้นตอนการวางแผนงาน และออกแบบระบบในช่วงแรกโดยสามารถให้ผู้ใช้สามารถเรียกชิ้นส่วนใบหน้ามาเลือก และประกอบบนจอภาพได้อย่างสะดวก รวดเร็วและง่าย แต่ละชิ้นส่วนสามารถทำการเลือก และปรับเปลี่ยนรูปลักษณะช่วยให้เห็นข้อแตกต่างกัน เพื่อให้ผู้ใช้สามารถ ทำการเลือกได้อย่างมีประสิทธิภาพ และขยับปรับตำแหน่งของแต่ละชิ้นส่วนได้ เช่นตำแหน่ง หรือระยะห่างของตา และคิ้วเป็นต้น สามารถทำการปรับขนาดของแต่ละชิ้นส่วน โดยบีบหรือ ขยายภาพแต่ละชิ้นส่วนให้ กว้าง แคบ หนา และบาง ได้ตามต้องการ และแต่งเติมรายละเอียด และปรับปรุงคุณภาพของภาพ โดยใช้ทฤษฎีการประมวลผลภาพ (Image Processing) เช่นการทำ Smoothing และ Scaling เป็นต้น นอกจากนั้นภาพที่สร้างได้สามารถทำการ บันทึกไว้ในแฟ้มข้อมูลประวัติอาชญากร ในลักษณะที่เป็นรหัสของส่วนประกอบต่าง ๆ พร้อมทั้งรายละเอียดเพิ่มเติม เพื่อความสะดวกในการค้นหาเรียกใช้ในภายหลัง โดยในแต่ละขั้นตอน ได้มีการออกแบบการติดต่อกับผู้ใช้ที่ง่าย และเหมาะสมกับการใช้งานจริงโดยได้รับคำปรึกษาจากทางกรมตำรวจโดยตรงเพื่อให้ผลงานที่ได้มีความสมบูรณ์มาก ที่สุดเท่าที่จะเป็นไปได้

* รูปที่ 1 แสดงหน้าจอเมื่อเริ่มการทำงาน
* รูปที่ 2 แสดงหน้าจอเมื่อจะเปิดไฟล์
* รูปที่ 3 แสดงหน้าจอหลังเปิดไฟล์สำเร็จ
* รูปที่ 4 แสดงหน้าจอเมื่อจะทำการเคลื่อนย้ายหรือเปลี่ยนแปลงขนาดของส่วนประกอบใบหน้า
* รูปที่ 5 แสดงหน้าจอขณะทำการเลือกปากให้มีลักษณะแบบใด
* รูปที่ 6 แสดงหน้าจอภายหลังเปลี่ยนแบบปากเป็นแบบใหม่

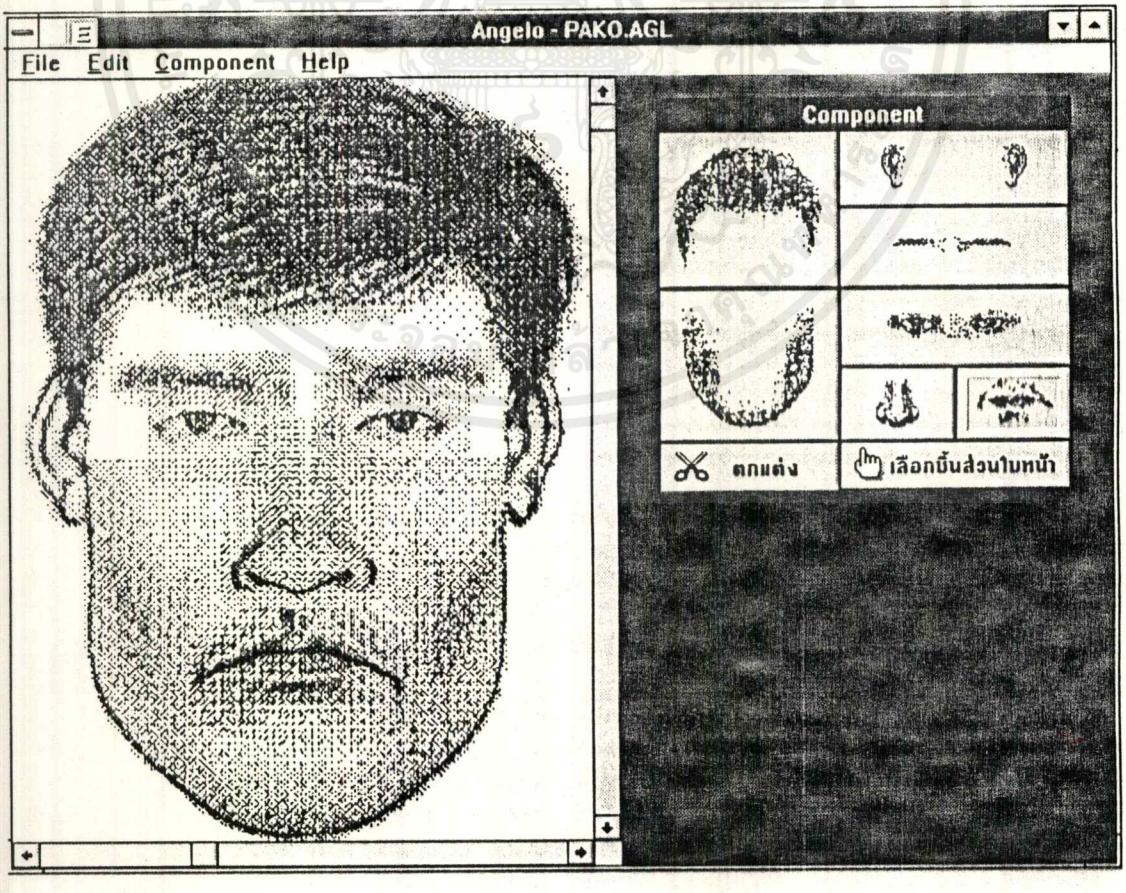


(บน) รูปที่ 1 (ล่าง) รูปที่ 2

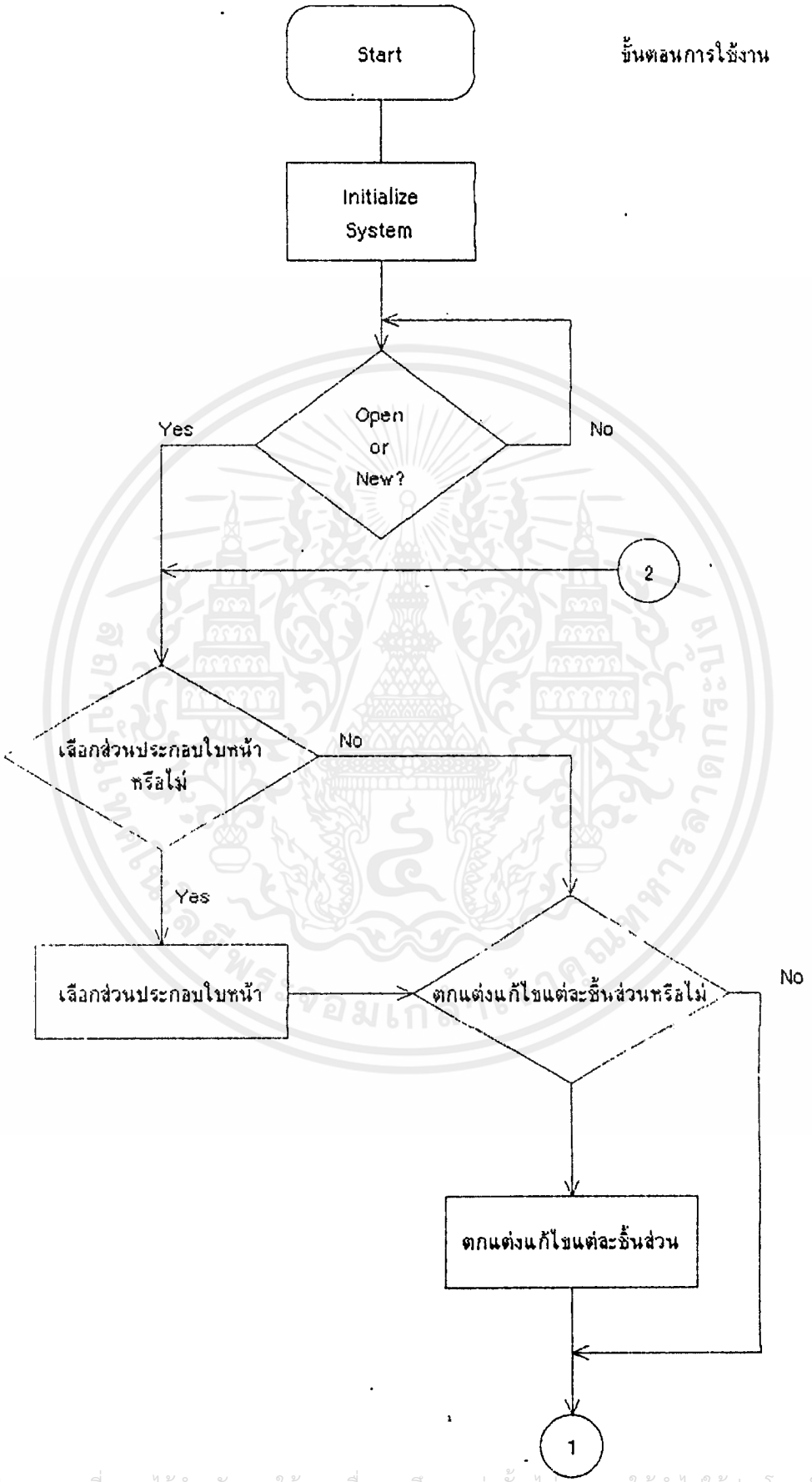


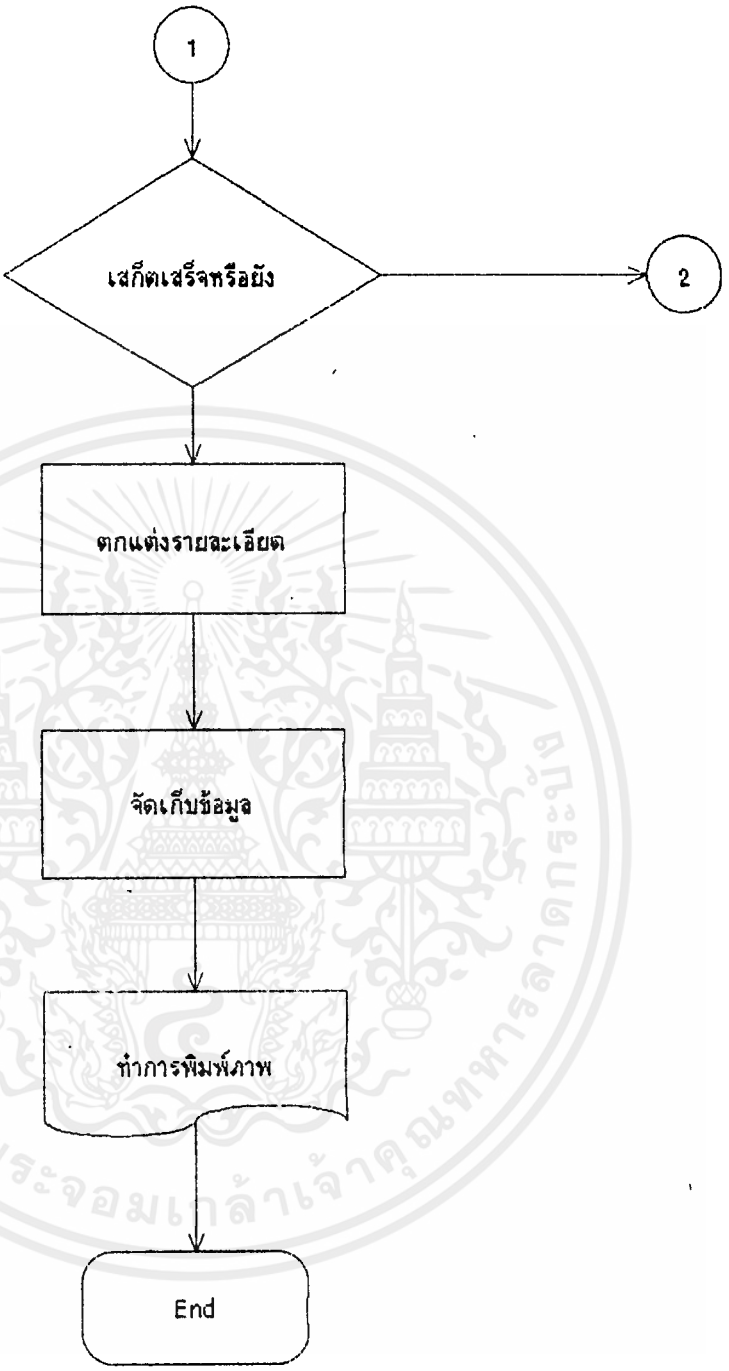


(บน) รูปที่ 5 (ล่าง) รูปที่ 6



รูปแสดงผลงานที่ได้จากการพิมพ์

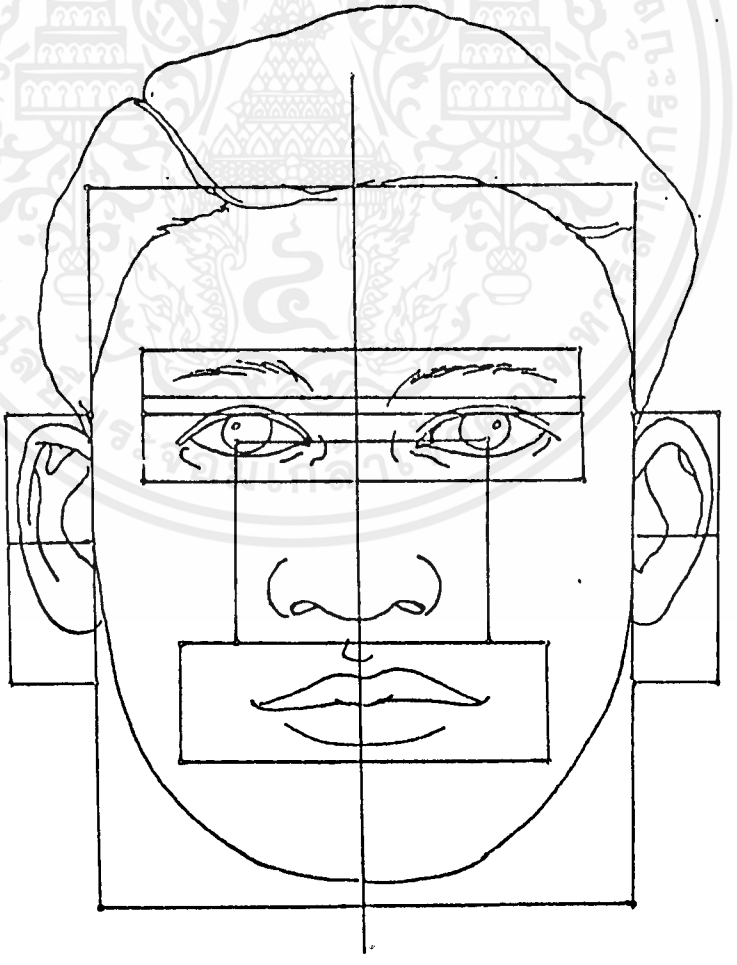




ส่วนประกอบของใบหน้า

ส่วนประกอบต่างๆของใบหน้าที่จะทำการเก็บในฐานข้อมูลจะมีการแบ่งแยกเป็นส่วนต่างๆ ดังนี้คือ

1. ทรงผม (Hair)
2. เค้าโครงรูปหน้าและคาง (Face outline)
3. หูซ้าย
4. หูขวา
5. คิ้วซ้าย
6. คิ้วขวา
7. ตาซ้าย
8. ตาขวา
9. จมูก
10. ปาก
11. ส่วนอื่นๆ เช่น หนวด, แว่นตา, รอยย่น ฯลฯ



โครงสร้างของฐานข้อมูล

มีการแบ่งหมวดหมู่ของแฟ้มข้อมูลเป็น 12 ชนิดคือ

1. ผม	HAIR*.BMP
2. รูปหน้า	FACE*.BMP
3. หูซ้าย	LEAR*.BMP
4. หูขวา	REAR*.BMP
5. คิ้วซ้าย	LBOW*.BMP
6. คิ้วขวา	RBOW*.BMP
7. ตาซ้าย	LEYE*.BMP
8. ตาขวา	REYE*.BMP
9. จมูก	NOSE*.BMP
10. ปาก	MOUT*.BMP
11. อื่นๆ	*.ANG

โครงสร้างของไฟล์ข้อมูล

โครงสร้างของแฟ้มหน้าคนร้ายประกอบด้วยการเก็บชื่อของภาพต้นแบบและฟอร์แมตภาพบิตแมปที่ได้จากการตกแต่งแก้ไข มีรูปแบบทางไบนารี่ดังนี้

ให้ x แทนเลขฐานสิบหก

HAIR No.	OriginX	OriginY	WIDTH	HEIGHT	รูปแบบผม
xxxx	xxxx	xxxx	xxxx	xxxx	
FACE	OriginX	OriginY	WIDTH	HEIGHT	No.โครงหน้า
xxxx	xxxx	xxxx	xxxx	xxxx	
LEAR	OriginX	OriginY	WIDTH	HEIGHT	No.หูซ้าย
xxxx	xxxx	xxxx	xxxx	xxxx	
REAR	OriginX	OriginY	WIDTH	HEIGHT	No.หูขวา
xxxx	xxxx	xxxx	xxxx	xxxx	
LBOW	OriginX	OriginY	WIDTH	HEIGHT	No.คิ้วซ้าย
xxxx	xxxx	xxxx	xxxx	xxxx	
RBOW	OriginX	OriginY	WIDTH	HEIGHT	No.คิ้วขวา
xxxx	xxxx	xxxx	xxxx	xxxx	

LEYE	OriginX	OriginY	WIDTH	HEIGHT	No.ตาซ้าย
xxxx	xxxx	xxxx	xxxx	xxxx	
REYE	OriginX	OriginY	WIDTH	HEIGHT	No.ตาขวา
xxxx	xxxx	xxxx	xxxx	xxxx	
NOSE	OriginX	OriginY	WIDTH	HEIGHT	No.จมูก
xxxx	xxxx	xxxx	xxxx	xxxx	
MOUT	OriginX	OriginY	WIDTH	HEIGHT	No.ปาก
xxxx	xxxx	xxxx	xxxx	xxxx	

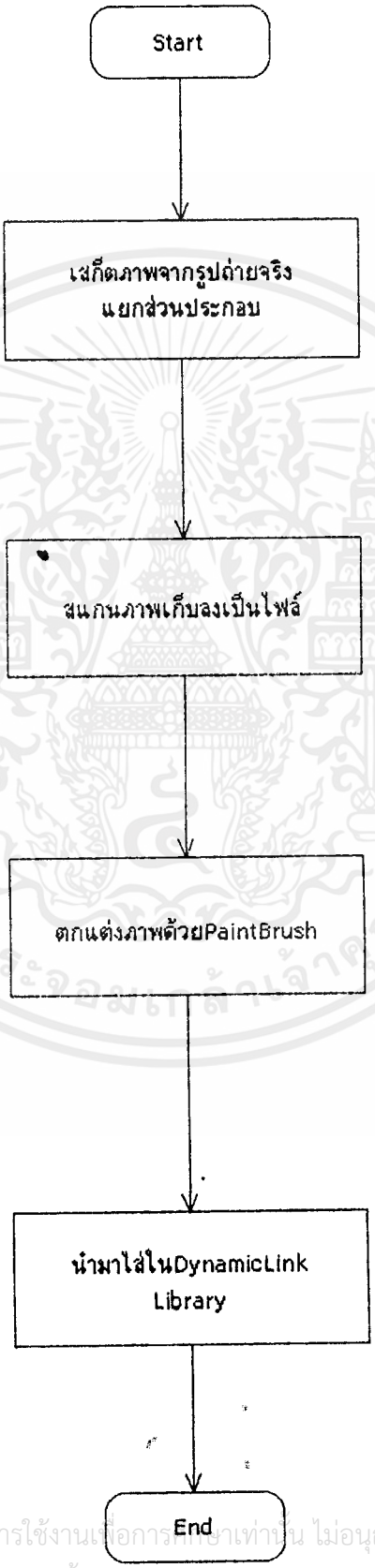
BITMAP TEMPLATE NAME ชื่อไฟล์บิตแมปที่จะนำมาทำการ AND กัน

xxxxxxxxxx

จะเห็นว่าขนาดของแฟ้มข้อมูลที่เก็บภาพคนร้ายใดๆ จะมีขนาดคงที่เสมอจากการเพราะใช้การอ้างถึงข้อมูลในฐานข้อมูลกลางที่เก็บไว้ ส่วนองค์ประกอบอื่นๆนั้น ในโครงงานนี้จะยังไม่มีการใช้ BITMAP TEMPLATE NAME ข้อมูลจถูกแบ่งเป็นกลุ่มๆ ตามจำนวนกลุ่มของฐานข้อมูลหลัก แต่ละกลุ่มจะเก็บ Integer ไว้ 5 ตัวคือ

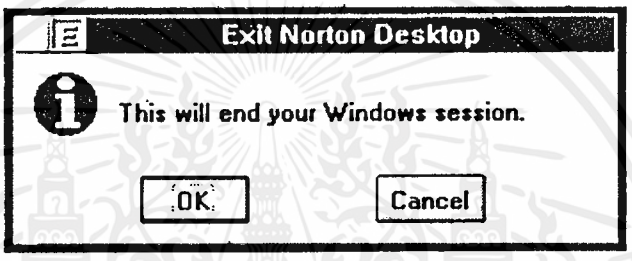
xxxx	ตัวแรกในแต่ละบรรทัด หมายถึง Style ขององค์ประกอบนั้นว่าตรงกับหมายเลขที่เท่าไรในฐานข้อมูล
OriginX	หมายถึง จุดอ้างอิงในแกนนอนที่จะนำภาพไปแสดงบนจอภาพ
OriginY	หมายถึง จุดอ้างอิงในแกนตั้งที่จะนำภาพไปแสดงบนจอภาพ
WIDTH	หมายถึง ความกว้างเป็นจำนวน pixel ขององค์ประกอบนั้น
HEIGHT	หมายถึง ความ^'เป็นจำนวน pixel ขององค์ประกอบนั้น

ขั้นตอนการนำข้อมูลเข้า



การเขียนโปรแกรมบน Microsoft Windows

Microsoft Windows เป็นโปรแกรมประเภท GUI ซึ่งย่อมาจาก Graphical User Interface ซึ่งเป็นการติดต่อกับผู้ใช้คอมพิวเตอร์โดยใช้รูปภาพ รูปกราฟฟิกหรือรูปสัญลักษณ์ต่างๆ ที่แสดงออกทางจอภาพแทนข้อความต่าง ๆ หรือ การใช้คำสั่ง หลายขั้นตอน ผู้ใช้สามารถที่จะสั่งงานหรือทำงานกับคอมพิวเตอร์โดยผ่านทางรูปภาพนั้น ซึ่งวิธีการนี้ได้พิสูจน์ให้เห็นแล้วว่าทำให้ผู้ใช้สามารถติดต่อกับคอมพิวเตอร์ได้ง่ายกว่ารูปแบบที่ปรากฏออกมา และการใช้งานนั้นเป็นธรรมชาติมากกว่าการติดต่อกับคอมพิวเตอร์ในรูปแบบเดิมที่ต้องมีการพิมพ์ตัวอักษรหรือข้อความเพื่อออกคำสั่งให้กับคอมพิวเตอร์ อุปกรณ์ที่เป็นอินพุตที่ใช้ในการติดต่อกับคอมพิวเตอร์ในแบบ GUI โดยปกติจะใช้เมาส์และคีย์บอร์ด แต่ถ้าใช้เมาส์จะเกิดความคล่องตัว และใช้ง่ายกว่าการใช้คีย์บอร์ดมาก ตัวอย่างที่เห็นได้ทั่วไป เช่นเมื่อจบโปรแกรมโดยทั่วไป จะมีข้อความแสดงออกมาทางจอภาพ "EXIT (Y/N)?" เพื่อถามความแน่ใจ ผู้ใช้ก็กดปุ่ม Y หรือ N เป็นการตอบรับ แต่ในรูปแบบของ GUI จะเกิดบล็อกสี่เหลี่ยมหน้าตาคล้าย ๆ รูปที่ 1 ปรากฏขึ้นมา



รูปที่ 1 แมสเสจ box ของ Microsoft Windows

การตอบรับด้วยวิธีนี้ ผู้ใช้ก็เพียงแต่ทำการเลื่อนเคอร์เซอร์ของเมาส์ไปที่ปุ่ม Ok หรือ Cancel แล้วคลิกปุ่มของเมาส์เพื่อตอบรับ จะเห็นได้ว่าวิธีนี้จะสามารถเข้าใจได้ง่ายกว่า เหมือนกับการกดปุ่มเครื่องใช้ไฟฟ้าทั่ว ๆ ไปและถ้าเป็นระบบ GUI ที่ดี ๆ เมื่อมีการคลิกที่ปุ่มที่ปรากฏบนจอภาพ รูปของปุ่มจะยุบลงไปเหมือนกับถูกกดจริง ๆ ด้วย

Microsoft Windows กับ GUI

จากที่กล่าวมาแล้วว่า Microsoft Windows เป็นโปรแกรม GUI ประเภทหนึ่งซึ่งเป็นโปรแกรม GUI ที่ดังมากในเครื่องระดับไมโครคอมพิวเตอร์ ลักษณะของ Microsoft Windows เป็นโปรแกรมที่มีลักษณะของโปรแกรมประเภท GUI อย่างเต็มที่ มีการติดต่อกับผู้ใช้ผ่านทางรูปกราฟิก เช่น ไอคอน อินพุตสามารถใช้ได้ทั้งเมาส์ และ คีย์บอร์ด (แต่ถ้าใช้เมาส์จะสะดวกกว่ามาก) โปรแกรมนี้จะจัดการกับสภาวะแวดล้อมต่าง ๆ ให้กับโปรแกรมประยุกต์ที่ทำงานอยู่ภายใต้โปรแกรมนี้ได้เกือบทั้งหมด (ที่ว่าเกือบทั้งหมดก็เพราะแอพพลิเคชันโปรแกรมต้องจัดการกับไฟล์เอง Microsoft Windows จะไม่ยุ่งเกี่ยวด้วย) ซึ่งรวมถึงการจัดการหน่วยความจำ และนอกจากนี้ยังมีลักษณะที่สามารถโหลดเอาโปรแกรมหลาย ๆ ตัวมาอยู่ในหน่วยความจำ และปรากฏบนหน้าจอได้พร้อมกัน โดยสามารถเลือกสลับโปรแกรมที่ต้องการทำงานขึ้นมาทำงานได้ การใช้ลักษณะของ GUI ที่เห็นได้ง่ายและชัดเจนบน Microsoft Windows ก็คือ การที่ผู้ใช้ต้องการเรียกใช้งานแอพพลิเคชันโปรแกรม ผู้ใช้ก็เพียงแต่ลากเมาส์ให้เคอร์เซอร์ของเมาส์ชี้ไปที่ไอคอน ซึ่งเป็นรูปสัญลักษณ์แทนตัวแอพพลิเคชันโปรแกรม แล้วทำดับเบิลคลิกเมาส์ ซึ่งผู้ใช้ที่เคยลองใช้ดูก็จะรู้สึกว่าเหมือนกับการชี้นิ้วสั่งอยากให้แอพพลิเคชันตัวไหนขึ้นมารันก็เพียงแต่ชี้นิ้วสั่งซึ่งสะดวกและง่ายกว่าการพิมพ์ชื่อโปรแกรมนอกจากนี้ก็เช่นในไดอล็อกบ็อกซ์ที่มี control ต่างๆ เช่น radio button ถ้า radio button อยู่ในสภาวะ ON ผู้ใช้ลากเมาส์ไปคลิกที่ radio button สถานะจะถูกเปลี่ยนเป็น OFF

และถ้าคลิกอีกทีก็จะกลายเป็น ON เหมือนกับการใช้ปุ่มของเครื่องใช้ไฟฟ้า

รูปที่ 2 radio botton ขณะ ON และ OFF

การสร้าง Windows Application

การเขียนโปรแกรมที่ทำงานบน Microsoft Windows หรือที่เรียกกันว่าโปรแกรมแอพพลิเคชันนั้นจะมีลักษณะแตกต่างกับการเขียนโปรแกรมที่ทำงานทั่ว ๆ ไป เนื่องจากความแตกต่าง ของลักษณะโครงสร้างและการทำงานขั้นตอนโดยย่อของการสร้างแอพพลิเคชันก็มีดังนี้

1. ทำการสร้าง source ไฟล์ด้วยโปรแกรมอิดิเตอร์ทั่วไป ในโปรแกรมจะต้องมีฟังก์ชัน Winmain และ window procedure และโปรแกรมที่ต้องการให้ทำงาน (ฟังก์ชันที่เขียนขึ้นเพื่อรับการทำงานมาจาก window function อีกทีหนึ่ง)
2. สร้าง Icon, Cursor, Bitmap ตามที่โปรแกรมต้องการใช้
3. สร้าง Resource script ไฟล์หรือไฟล์ที่นิยาม resource ทั้งหมดที่จะใช้ในโปรแกรมเช่น Menu, Icon, Dialog boxes ตามที่สร้างไว้ในขั้นที่2
4. สร้าง Module definition
5. ทำการคอมไพล์และลิงค์โปรแกรมที่เขียนไว้
6. ใช้ Resource compiler คอมไพล์ผลลัพธ์ของขั้นตอนที่ 5 ร่วมกับไฟล์ resource script เพื่อแปลงให้เป็น .EXE

ขั้นตอนที่หนึ่ง ซึ่งก็คือการเขียนแอพพลิเคชันที่ทำงานบน Microsoft Windows ส่วนใหญ่มักเขียนกันโดยใช้ภาษา C แต่จะใช้ภาษาอื่นเขียนก็ได้ แต่มีข้อแม้ว่าคอมไพเลอร์ที่ใช้ต้องเป็น คอมไพเลอร์ที่สามารถสร้างแอพพลิเคชันบน Windows เท่านั้น โครงสร้างของตัวโปรแกรมอย่างคร่าวๆนั้น เริ่มที่ต้องมีการ include ไฟล์ windows.h ซึ่งเป็นที่ที่เก็บการประกาศฟังก์ชันต่าง ๆ data type ค่าคงที่ และตัวแปร structure ต่างๆ ที่ต้องใช้ในการเขียนโปรแกรมบน Windows ส่วนนี้เป็นส่วนที่ขาดไม่ได้เลย ส่วนต่อไปจะเป็นจุดที่โปรแกรมเริ่มต้นทำงาน ถ้าเป็นภาษา C มาตรฐานก็จะเป็นฟังก์ชันที่ชื่อว่า main แต่สำหรับแอพพลิเคชันบน Windows จะเป็นฟังก์ชันที่ชื่อ WinMain (ต้องชื่อนี้เท่านั้น ห้ามเปลี่ยนชื่อ) ภายในฟังก์ชันนี้ก็จะประกอบไปด้วยรูทีนต่าง ๆ แบ่งได้ 4 ส่วนดังต่อไปนี้

ส่วนแรกเป็นส่วนที่ทำการลงทะเบียนคลาสของวินโดว์ (register window class) ส่วนนี้จะเป็นการกำหนดค่าให้กับแต่ละฟิลด์ของตัวแปร structure ชนิด WNDCLASS การทำเช่นนี้ก็เพื่อกำหนดรายละเอียดลักษณะพื้นฐานของวินโดว์หลัก ของแอพพลิเคชันที่จะถูกแสดงบนจอภาพซึ่ง มีอยู่ทั้งหมด 10 ฟิลด์ เมื่อทำการกำหนดค่าเรียบร้อยแล้วก็คล้ายกับการเขียนใบสมัครงาน เมื่อกรอกรายละเอียดเสร็จ ต้องนำใบสมัครนั้นไปส่ง ในกรณีนี้ก็เช่นกันต้องส่งข้อมูลเหล่านั้นให้กับ โปรแกรม Microsoft Windows โดยการเรียกใช้ฟังก์ชัน RegisterClass พารามิเตอร์ที่ให้ กับฟังก์ชันนี้จะเป็นพอยน์เตอร์ที่ชี้ไปยังตัวแปรชนิด WNDCLASS ที่ถูกใส่ค่าไว้นั่นเอง ส่วนแรกทั้งหมดนี้จะอยู่ภายใน if statement ซึ่งจะทดสอบว่ามีแอพพลิเคชันที่เหมือนกันกับแอพพลิเคชันทำงานอยู่ก่อนหน้านี้หรือไม่ ถ้ามีอยู่แล้วแอพพลิเคชันจะไม่ทำการลงทะเบียนคลาสของวินโดว์ซ้ำ (โดยทั่วไป แอพพลิเคชันโปรแกรมตัวเดียวกันสามารถถูกเรียกขึ้นมาใช้งานซ้ำๆกันได้ เช่น การเรียกแอพพลิเคชัน Clock ของ Windows ขึ้นมาทำงานพร้อมกันหลาย ๆ ตัว) จะใช้ข้อมูลที่มีอยู่แล้ว

ซึ่งการทำเช่นนี้เป็นผลดีเพราะทำให้ไม่เปลืองเนื้อที่ในหน่วยความจำ ตัวอย่างของส่วนนี้จะเห็นได้ดังที่แสดงข้างล่าง

```
if(!nPrevInstance)
{
        wndclass.lpszClassName  =  "Sample:MAIN";//Class Name
        wndclass.lpfnWndProc    =  WndProc;//Diary Window Procedure
        wndclass.cbClsExtra     =  0;
        wndclass.cbWndExtra     =  0;
        wndclass.hInstance      =  hInstance;
        wndclass.hIcon          =  LoadIcon (hInstance,"ICON_RES");
        wndclass.hCursor        =  LoadCursor (NULL,IDC_ARROW); //Normal arrow
        wndclass.hbrBackground  =  COLOR_WINDOW + 1;
        wndclass.lpszMenuName   =  NULL; //Menu Name
        wndclass.style          =  CS_HREDRAW|CS_VREDRAW|CS_DBLCLKS;
        RegisterClass(&wndclass);
}
```

ส่วนที่สอง เป็นส่วนที่ทำหน้าที่กำหนดรายละเอียดปลีกย่อยของวินโดว์นอกเหนือจากข้อมูลที่ถูกลงทะเบียนในส่วนแรก ส่วนนี้จะมีการเรียกใช้ฟังก์ชัน CreateWindow ข้อมูลส่วนที่เหลือที่ต้องการส่งไปจะถูกส่งทางพารามิเตอร์ของฟังก์ชัน CreateWindow ซึ่งทำให้ลักษณะบางอย่างของวินโดว์ที่ถูกสร้างขึ้นของแอพพลิเคชันเดียวกันแตกต่างกันได้ เช่นพิกัดบนหน้าจอตอนที่เริ่มต้นทำงาน

ส่วนที่สาม เมื่อโปรแกรมทำงานถึงส่วนนี้จะยังไม่มีวินโดว์ของแอพพลิเคชันที่ร้าเรียกใช้ถูกแสดงขึ้นมาบนจอภาพ ต้องมีการเรียกใช้ฟังก์ชันสองฟังก์ชันก่อน ฟังก์ชันแรกคือฟังก์ชัน ShowWindow และฟังก์ชันที่สองคือฟังก์ชัน UpdateWindow ฟังก์ชัน ShowWindow ทำหน้าที่สร้างวินโดว์ตามลักษณะที่ต้องการบนหน้าจอ เมื่อเรียกใช้ฟังก์ชันนี้แล้วที่สร้างวินโดว์ของ แอพพลิเคชันจะปรากฏขึ้นมาบนจอภาพ แต่เนื้อที่ภายในวินโดว์จะยังว่างเปล่าอยู่ยกตัวอย่างเช่น ถ้าเป็นโปรแกรมแอพพลิเคชัน Clock ก็จะมีวินโดว์ของ Clock ปรากฏขึ้นมา แต่รูปนาฬิกาที่มีเข็มซึ่งอยู่ภายในวินโดว์จะยังไม่ปรากฏ ส่วนฟังก์ชัน UpdateWindow จะทำหน้าที่ส่งแมสเสจ WM_PAINT ไปให้กับ window procedure ซึ่งจะทำให้พื้นที่ภายในวินโดว์ของแอพพลิเคชันถูกวาดเป็นรูปร่างหรือเขียนข้อความตามที่โปรแกรมกำหนด

ส่วนสุดท้าย เป็นส่วนของแมสเสจ loop ซึ่งเป็นส่วนที่ตัวแอพพลิเคชันจะเชื่อมต่อกับ Microsoft Windows ส่วนนี้จะปรากฏให้เห็นในโปรแกรมที่ทำงานในลักษณะ GUI ซึ่งจะมีโครง สร้างพื้นฐานลักษณะเดียวกัน คือจะเป็นลูปที่วนไปเรื่อย ๆ จนกว่าต้องการจะจบการทำงานของ แอพพลิเคชันซึ่งภายในลูปจะมีการทำงานพื้นฐานอยู่สองอย่างคือ การตรวจสอบเหตุการณ์ที่เกิดขึ้นและส่วนที่ทำการประมวลผลเหตุการณ์ที่เกิดขึ้น สำหรับส่วนนี้ของแอพพลิเคชันโปรแกรมที่ทำงานบน Microsoft Windows จะเป็น while loop ซึ่งในส่วนที่ตรวจสอบว่าจะหลุดจาก while loop นี้เมื่อไร จะเป็นการเรียกใช้ฟังก์ชัน GetMessage ซึ่งทำหน้าที่ดึงเอาแมสเสจมาจาก message queue ฟังก์ชันนี้ปกติจะส่ง ค่ากลับเป็น 0 แต่ถ้าได้รับแมสเสจ WM_QUIT จากแมสเสจ queue เมื่อไรก็ตาม ค่าก็ถูกส่งกลับจะไม่เป็น 0 ซึ่งทำให้โปรแกรมหลุดออกจาก while loop ซึ่งเป็นการจบการทำงานของแอพพลิเคชัน ตัวอย่างของแมสเสจ loop มีดังนี้

```
while (GetMessage(&msg,NULL,0,0))
{
        TranslateMessage(&msg);
        DispatchMessage(&msg);
}
```

ภายในแมสเสจ loop ซึ่งเป็น while loop นี้จะมีการเรียกใช้ฟังก์ชันสองฟังก์ชันฟังก์ชันแรก คือ ฟังก์ชัน TranslateMessage ฟังก์ชันนี้จะทำหน้าที่แปลงแมสเสจที่ถูกส่งมาในกรณีที่เป็นแมสเสจที่ถูกส่งมาในกรณีที่เป็นแมสเสจที่เกี่ยวกับคีย์บอร์ดให้มีค่าที่สามารถนำไปใช้งานได้ ส่วนอีกฟังก์ชันหนึ่งคือฟังก์ชัน DispatchMessage ฟังก์ชันนี้จะทำหน้าส่งแมสเสจที่ได้รับมานั้นกลับไปให้กับ Windows ตัวโปรแกรม Windows ก็จะส่งแมสเสจนั้นไปให้กับ window procedure ที่เหมาะสมจะนำแมสเสจนั้นไปประมวลผล ส่วนต่อไปเป็นส่วนของ window procedure ส่วนนี้เป็นส่วนที่ทำหน้าที่ประมวลผลแมสเสจต่าง ๆ ที่ถูกส่งมา ส่วนนี้จะเป็นฟังก์ชันซึ่งมีรูปแบบและพารามิเตอร์ที่แน่นอน ส่วนชื่อของฟังก์ชันนี้ถูกกำหนดไว้ตอนที่ลงทะเบียนคลาสของวินโดว์ ที่ต้องทำเช่นนี้ก็เพื่อทำให้ Windows รู้ว่าต้องส่งแมสเสจไปประมวลผลที่ฟังก์ชันใดในโปรแกรม รูปแบบภายใน window procedure อยู่ในรูปของ switch และ case statement โดยจะใช้ statement switch เลือกเอาแมสเสจที่ต้องการไปทำการประมวลผลซึ่ง case statement จะเป็นส่วนที่ทำการระบุว่า แต่ละ case รับผิดชอบแมสเสจไหน ตัวแอพพลิเคชันไม่จำเป็นต้องจัดการกับทุกแมสเสจที่ได้รับจะรับแมสเสจใดขึ้นอยู่กับว่าต้องการแมสเสจไหน ส่วนแมสเสจที่ไม่ได้ถูกใช้งานต้องถูกส่งกลับคืนให้กับตัวโปรแกรม Windows โดยใช้ฟังก์ชัน DefWindowProc ซึ่งโดยปกติฟังก์ชันนี้จะอยู่ภายใน default statement นอกจากนี้ยังมีข้อกำหนดในการเขียนโปรแกรมอีกว่าทุก ๆ case statement ต้องจบด้วยคำสั่ง return 0 ทุกครั้ง ตัวอย่างของ window procedure มีหน้าตาแบบนี้

```
long FAR PASCAL WndProc(HWND hWnd,WORD Message, WORD wParam,LONG lParam)
{
        switch(Message)
        {
                case WM_PAINT:          [คำสั่งที่ทำงานเมื่อมี WM_PAINT message เข้ามา]
                    return 0;
                case WM_DESTROY:        [คำสั่งที่ทำงานเมื่อมี WM_DESTROY message เข้ามา]
                    return 0;
        }
        return DefWindowProc(hWnd,Message,wParam,lParam);
}
```

จะสรุปได้ว่าหลักใหญ่ๆ ของการเขียนแอพพลิเคชันโปรแกรมของ Windows แท้ที่จริงแล้วก็คือ การจัดการกับแมสเสจนั่นเอง และนอกจากนี้โปรแกรมก็สามารถจะส่งแมสเสจไปให้กับวินโดว์เพื่อให้วินโดว์มีลักษณะหรือเปลี่ยนไปตามที่กำหนดได้อีกด้วย โดยการใช้ฟังก์ชันที่ทำหน้าที่ส่งแมสเสจ ส่วนแมสเสจของ Windows จะขอกล่าวถึงแมสเสจที่สำคัญไว้เพียง 2 แมสเสจดังนี้ ตัวแมสเสจแรกก็คือ WM_PAINT แมสเสจโปรแกรม Windows จะส่งแมสเสจนี้มาให้กับแอพพลิเคชันเมื่อต้องมีการวาดเนื้อที่บางส่วนภายในวินโดว์ใหม่ เช่นเนื่องจากถูกส่วนของเมนูกับหรือถูกวินโดว์อื่น ๆ ทับไว้หรือในกรณีตอนเริ่มเรียกแอพพลิเคชันนี้มาใช้งานในตอนแรก เนื้อที่ภายในวินโดว์ก็ต้องถูกวาดใหม่หมด จากที่กล่าวมาแล้วว่าฟังก์ชัน UpdateWindow จะส่ง WM_PAINT แมสเสจไปให้กับ Windows ซึ่งจะส่งมาให้กับ window procedure อีกต่อหนึ่ง ซึ่งการทำเช่นนี้จะทำให้เนื้อที่ภายในวินโดว์ถูกวาดเป็นครั้งแรก คำสั่งหรือฟังก์ชันที่ใช้ในการวาดรูปภาพต่าง ๆ ข้อความหรือตัวอักษรจะอยู่ภายในคำสั่ง case WM_PAINT นี้ซึ่งคำสั่งเหล่านี้ต้องถูกประกบอยู่ระหว่างฟังก์ชันสองฟังก์ชันคือ BeginPaint และ EndPaint ดังนี้

```
case   WM_PAINT:
       BeginPaint(hwnd,&ps);
       . . .
       EndPaint(hwnd,&ps);
       return 0;
```

และนอกจากนี้ยังมีข้อห้ามเกี่ยวกับ WM_PAINT แมสเสจนี้อยู่ข้อหนึ่งก็คือ ห้ามใช้คำสั่งดังตัวอย่างนี้

```
case WM_PAINT :
    return 0;
```

ส่วนแมสเสจที่สำคัญอีกแมสเสจหนึ่งก็คือ WM_DESTROY แมสเสจซึ่งแมสเสจนี้จะถูกส่งมาเมื่อผู้ใช้เลือก Close ที่อยู่ภายใน system menu หรือกด Alt-F4 เมื่อมีแมสเสจนี้เกิดขึ้นสิ่งที่แอพพลิเคชันต้องทำถ้าต้องการจบการทำงานของแอพพลิเคชันก็คือ เรียกใช้ฟังก์ชัน PostQuitMessage ซึ่งฟังก์ชันนี้จะส่ง WM_QUIT แมสเสจลงในแมสเสจ queue ของโปรแกรมซึ่งทำให้การทำงานหลุดจากแมสเสจ loop

resource script file (ไฟล์ .RC)

resource script file เป็นไฟล์ที่เก็บรูปแบบของ resource ต่าง ๆ ที่ใช้ในการสร้างแอพพลิเคชัน เช่น เมนูและไดอล็อกบ็อกซ์ และกำหนดว่าบิตแมปหรือไอคอนที่ต้องการใช้ในแอพพลิเคชันถูกเก็บไว้ในไฟล์ที่ไหน และจะกำหนดชื่อบิตแมปหรือไอคอนนั้นไว้ด้วยซึ่ง resource เหล่านี้ Microsoft Windows จะจัดการสิ่งต่าง ๆ ให้จนทำให้ส่วนโค้ดที่เกี่ยวข้องกับ resource เหล่านี้น้อยลงอย่างเห็นได้ชัด เมื่อเทียบกับโปรแกรมปกติที่ให้ผลที่คล้ายๆกันไฟล์นี้ดูปร่างหน้าตาเป็นไฟล์ข้อความธรรมดา ส่วนที่สำคัญในไฟล์ชนิดนี้ก็คือส่วนของเมนู และส่วนของไดอล็อกบ็อกซ์ ส่วนของเมนูซึ่งเมนูของแอพพลิเคชันมีอยู่สองประเภท คือเมนูของระบบ (เมนูที่ใช้ในการ เลื่อนวินโดว์ ลดหรือขยายขนาดของวินโดว์นั่นเอง) ซึ่งเป็นเมนูที่ทุกแอพพลิเคชันมีและเมนูอีกประเภทหนึ่งคือเมนูที่แอพพลิเคชันกำหนดขึ้นมาเอง ซึ่งก็คือเมนูที่กำลังจะกล่าวถึง รายละเอียดของเมนูในไฟล์นี้จะบอกว่าเมนูนี้ชื่ออะไร มีการเรียงลำดับกันอย่างไร มีเมนูย่อยอะไรบ้างแต่ละเมนูมีตัวเลือกอะไร และค่าที่ถูกส่งไปที่แอพพลิเคชันเมื่อผู้ใช้เลือกตัวเลือกในเมนู ซึ่งจะถูกส่งไปพร้อมกับ WM_COMMAND แมสเสจต้องการให้เป็นค่าใด ซึ่งจะทำให้แอพพลิเคชันรู้ว่าตอนนี้ผู้ใช้เลือกตัวเลือกใดได้ ที่จริงแล้วการกำหนดรายละเอียดของเมนูไม่จำเป็นที่จะต้องอยู่ใน resource script file ก็ได้ นอกจากวิธีนี้แล้วยังมีหลายวิธี แต่วิธีนี้เป็นวิธีที่นิยมที่สุด ตัวอย่างที่กำหนดรายละเอียดของเมนูมีหน้าตาดังนี้เมนูนี้มีชื่อว่า MenuName ต่อจากชื่อของเมนูจะเป็นส่วน load option ซึ่งจะมีไว้เพื่อบอกว่าต้องการให้ Windows โหลดข้อมูลของเมนูนี้เข้าไปเก็บไว้ในหน่วยความจำอย่างไรซึ่งมีได้ดังต่อไปนี้

```
Menuname [load option][memory option]
    {
        MENUITEM "&One", 1
        POPUP "&Two"
            {
                MENUITEM "&Ten",    4
                MENUITEM "&Eleven", 5
                MENUITEM "Ei&ghteen", 6
            }
        MENUITEM "Th&ree", 3
    }
```

PRELOAD หมายความว่า Windows จะโหลด resource นี้(เมนู) เข้าไปในหน่วยความจำทันทีที่โปรแกรมถูกเรียกใช้งาน

LOADONCALL หมายความว่า Windows จะโหลด resource นี้(เมนู) เข้าไปในหน่วย ความจำเฉพาะในกรณีที่ต้องการใช้งานเมนูเท่านั้น ถ้าไม่มีการกำหนด load option นี้ไว้ค่านี้จะถูกกำหนดให้เป็นLOADONCALL ส่วน memory option จะมีไว้เพื่อกำหนดว่าข้อมูลของเมนูนี้จะถูกเคลื่อนย้ายหรือถูกจัดการในหน่วยความจำอย่างไรซึ่งเป็นได้ดังต่อไปนี้

FIXED ข้อมูลเกี่ยวกับเมนูที่ถูกเก็บไว้ในหน่วยความจำ ตำแหน่งที่เก็บจะคงที่

MOVEABLE ข้อมูลเกี่ยวกับเมนูสามารถเคลื่อนย้ายไปที่ตำแหน่งอื่นในหน่วยความจำได้

DISCARDABLE ข้อมูลเกี่ยวกับเมนูสามารถที่จะถูกลบทิ้งได้ถ้าไม่มีการใช้ต่อไป

ถ้าไม่มีการกำหนด load option นี้ไว้ก็จะถูกกำหนดให้เป็น MOVEABLE เมนูระดับบนสุดจะต้องอยู่ภายในเครื่องหมายวงเล็บปีกกา (ซึ่งสามารถใช้ข้อความ BEGIN และ END แทนได้) รูปแบบของข้อความที่ปรากฏอยู่ภายในเครื่องหมายปีกกาเป็นไปได้สองแบบดังต่อไปนี้

MENUITEM "text", wID, options

และ

POPUP "text",options

ถ้าเป็น MENUITEM จะหมายความว่าเป็นตัวเลือกตัวหนึ่งในเมนู wID เป็นตัวเลขที่ต้องการให้ Windows ส่งไปให้กับแอพพลิเคชันเมื่อเลือกตัวเลือกนี้ (พร้อมกับ WM_COMMAND เมสเสจ) แต่ถ้าเป็น POPUP จะหมายความว่าถ้าเลือกตัวเลือกนี้จะเกิดเมนูย่อยของตัวเลือกนี้ขึ้นมาซึ่งก็จะมีเครื่องหมายปีกกา ซึ่งภายในเครื่องหมายปีกกาจะเป็นรายละเอียดของเมนูย่อยของตัว เลือกนั้น ข้อความซึ่งต้องการให้ปรากฏเป็นตัวเลือกในเมนูต้องอยู่ภายในเครื่องหมาย "" สำหรับ เครื่องหมาย ampersand (&) ที่ปรากฏอยู่ภายใน

ข้อความที่เป็นตัวเลือกนั้นจะทำให้ตัวอักษรตัวที่ อยู่ตามหลังนั้นมีเส้นใต้ปรากฏอยู่ (เช่น T&ext ในเมนูจะเป็น Text) และตัวอักษรนั้นจะกลาย เป็น Hot key ของตัวเลือกนั้นด้วย

ส่วนของไดอล็อกบ็อกซ์เป็นส่วนที่กำหนดชื่อของไดอล็อกบ็อกซ์ขนาด ตำแหน่ง และรายละเอียดของ control ต่างๆ ในไดอล็อกบ็อกซ์ รายละเอียดเหล่านี้โปรแกรม Windows จะนำไปสร้างไดอล็อกบ็อกซ์ให้ตามที่กำหนดไว้

ไดอล็อกบ็อกซ์มีอยู่2 ประเภทด้วยกันคือ modal ไดอล็อกบ็อกซ์และ modeless dialog box ซึ่งทั้งสองชนิดมีข้อแตกต่างกันคือแบบแรกเมื่อไดอล็อกบ็อกซ์ถูกกำหนดให้ขึ้นมาแสดงบนจอภาพ ผู้ใช้ไม่สามารถสลับวินโดวอื่นๆ ให้ขึ้นมาทำงานแทนได้ต้องจบการทำงานและปิดไดอล็อก บ็อกซ์นั้นเสียก่อน ซึ่งโดยปกติจะใช้การกดปุ่ม Ok หรือ Cancel ส่วนอีกชนิดหนึ่งนั้นผู้ใช้สามารถสลับเปลี่ยนวินโดวอื่นขึ้นมาทำงานได้โดยไดอล็อกบ็อกซ์นั้นยังไม่จบการทำงานโดยสมบูรณ์ได้ด้วย

ดังที่กล่าวมาแล้วข้างต้นว่า การกำหนด รูปแบบของไดอล็อกบ็อกซ์ที่ทำการสร้างนั้นจะกำหนด ไว้ใน resource script file (ไฟล์.rc) หรืออาจอยู่แยกกันก็ได้ซึ่งถ้าอยู่แยกกันโดยปกติไฟล์ที่เก็บรูปแบบของไดอล็อกบ็อกซ์จะมีนามสกุล .DLG และต้องเพิ่มข้อความในไฟล์ .rc เพื่อให้โปรแกรม rc คอมไพเลอร์ดึงไฟล์ที่เก็บข้อมูลของไดอล็อกบ็อกซ์มาคอมไพล์ด้วย ข้อความที่ว่ามีรูปแบบดังนี้

```
#include filename.dlg
```

รูปแบบการกำหนดจะคล้ายกับการกำหนดรูปแบบของเมนู ซึ่งจะอยู่ในรูปของเท็กซ์ไฟล์ (โดยปกติจะมีโปรแกรมยูทิลิตี้ที่ช่วยในการสร้างไดอล็อกบ็อกซ์โดยสร้างเป็นรูปเหมือนจริงแล้วโปรแกรมจะแปลงรูปที่ได้เป็นเท็กซ์ไฟล์ โปรแกรมเหล่านี้เช่น Dialog Editor และ Whitewater Resource Toolkit) ตัวอย่างของการกำหนดรูปแบบของไดอล็อกบ็อกซ์จะเป็นดังนี้

จากตัวอย่างบรรทัดแรกจะเป็นชื่อของไดอล็อกบ็อกซ์และเช่นเดียวกับทรัพยากรชนิดอื่น ชื่อสามารถจะกำหนดให้เป็นตัวเลขได้ ต่อจากชื่อก็เป็นโคออดิเนต x,y ของตำแหน่งมุมบนซ้ายสุด ของไดอล็อกบ็อกซ์ โดยจะเทียบกับ Client area ของ Parent window

ที่ทำการเรียกไดอล็อกบ็อกซ์ ส่วนค่าที่เหลืออีกสองค่าเป็นค่าความกว้างและความสูงของไดอล็อกบ็อกซ์ข้อความไดอล็อกบ็อกซ์สามารถจะเพิ่ม load option (PRELOAD และ LOADONCALL) และ memory option (FIXED, MOVEABLE และ DISCARDABLE) เข้าไปด้วยส่วนลักษณะของ ไดอล็อกบ็อกซ์ที่กำหนดไว้หลังข้อความ STYLE ในบรรทัดที่สองของตัวอย่างจะคล้ายกับการกำหนดในฟิลด์ style ของการใช้ฟังก์ชัน CreateWindow

ข้อความที่เหลือเป็นข้อความที่อยู่ภายในเครื่องหมายปีกกา (ซึ่งสามารถใช้คำว่า BEGIN และ END แทนข้อความเหล่านี้ได้) จะเป็นข้อกำหนดเกี่ยวกับ child window control ซึ่งจากตัวอย่างนี้มีอยู่ 3 ชนิดคือ CTEXT (centered text), ICON, DEFPUSHBUTTON (Default push button) รูปแบบทั่วไปของการกำหนด control เป็นดังนี้

```
MyDialog DIALOG 20,20,160,80
STYLE WS_POPUP|WS_DLGFRAME
{
        CTEXT "Text No. 1"
        ICON
        CTEXT "Text No. 2"
        CTEXT "This is test dialog box"
        DEFPUSHBUTTON "ok"
}
```

```
control-type "text" nID, xPos, yPos, xWidth, yHeight, DwStyle
```

control-type	มีไว้เพื่อบอกชนิดของ control ที่ต้องการแสดง text เป็นข้อความที่ต้องการแสดงของ control บริเวณที่ข้อความนี้จะปรากฏขึ้นอยู่กับชนิดของ control
nID	เป็นค่า ID ของ control ซึ่งเป็นค่าประจำของแต่ละ control
xPos และ yPos	บ็นตำแหน่งของโคออดิเนตที่ control นั้นปรากฏ ซึ่งเป็นตำแหน่งมุมบนขวาสุดของ control
xWidth และ yHeight	มีไว้เพื่อบอกความกว้างของ control ในแนวแกน x และความสูงในแนวแกน y ตามลำดับ
DwStyle	เป็นรูปแบบของ control ค่านี้จะเป็นอย่างไรขึ้นอยู่กับชนิดของ control

ส่วนการสร้างไฟล์ที่เก็บไอคอน, บิตแมปนั้น ต้องใช้เครื่องมือที่ใช้ในการสร้างไฟล์พวกนี้โดยเฉพาะที่พบกันบ่อยๆ ก็มี SDK tools และ WhiteWater Resource Toolkit ซึ่ง เครื่องมือเหล่านี้มีส่วนที่ทำการสร้างไดอล็อกบ็อกซ์อยู่ด้วยซึ่งทำให้การสร้างสะดวกขึ้นมากเพราะเครื่องมือในรูปแบบของรูปกราฟฟิก

โปรแกรมส่วนที่จัดการกับเมนูทำได้โดยรอรับ WM_COMMAND แมสเสจว่ามีแมสเสจนี้ข้ามาหรือไม่ ถ้ามีแมสเสจนี้ข้ามาก็ทำการตรวจสอบพารามิเตอร์ที่ได้ Windows ส่งมาให้กับ window procedure อีกทีหนึ่ง พารามิเตอร์นี้ปกติจะกำหนดให้ชื่อ wParam (เป็นพารามิเตอร์ ตัวที่ 3 ของ window procedure) ว่าเท่ากับค่าที่กำหนดไว้ให้ ส่งเมื่อมีการเลือกเมนูหรือไม่ จากค่านี้จะทำให้ทราบว่าผู้ใช้เลือกตัวเลือกใดของเมนู

โปรแกรมส่วนที่จัดการกับไดอล็อกบ็อกซ์ สามารถแบ่งย่อยออกได้อีกสองส่วนใหญ่ คือส่วนที่สั่งให้ Window สร้างและจัดการกับไดล็อกบ็อกซ์ และอีกส่วนเป็นฟังก์ชันที่จัดการเกี่ยวกับการประมวลผลแมสเสจต่าง ๆ ขณะที่ไดอล็อกบ็อกซ์ทำงานอยู่ เรียกฟังก์ชันนี้ว่าไดอล็อกบ็อกซ์ procedure

การเรียกไดอล็อกบ็อกซ์ให้แสดงขึ้นมาบนจอภาพต้องใช้ฟังก์ชันดังต่อไปนี้เรียงตามลำดับ เริ่มจากฟังก์ชัน MakeProcInstance ซึ่งทำหน้าที่สร้างส่วนของโปรแกรมซึ่งเรียกว่า instance trunk ซึ่งจะเป็นคำสั่งของภาษาแอสเซมบลีซึ่งมีลักษณะดังนี้

```
MOV AX,XXXX
JMP SSSS:0000
```

instance trunk นี้จะอยู่ในหน่วยความจำที่ถูกจองไว้ที่ตำแหน่งหนึ่ง ค่า XXXX เป็นค่า ของ data segment ของแอพพลิเคชันที่เรียกใช้ฟังก์ชันนี้ SSSS:0000 เป็นค่าเซ็กเมนต์และออฟเซ็ตของฟังก์ชันที่จะกระโดดไปทำต่อไป (ซึ่งในที่นี้คือ ไดอล็อกบ็อกซ์ procedure) การย้าย ค่า data segment ไปไว้ที่รีจิสเตอร์ AX ก็เพื่อจะได้ทำการเซ็ตค่า data segment ของ ฟังก์ชันที่จะกระโดดไปทำนั้น ให้เป็นค่าเดียวกันกับ data segment ของโมดูลที่เรียกใช้ที่ต้อง ทำเช่นนี้เนื่องจากว่าส่วนที่เรียกใช้คำสั่งนี้กับไดอล็อกบ็อกซ์ procedure จะอยู่คนละโมดูลกันซึ่ง จะใช้ data segment คนละส่วนกันแต่ต้องมาทำงานและใช้ data segment ร่วมกันส่วนค่าที่ได้จากฟังก์ชัน MakeProcInstance เป็นแอดเดรสของ instance trunk ซึ่งจะต้องใช้เป็นพารามิเตอร์ให้กับฟังก์ชันต่อไป

ฟังก์ชันที่ต้องเรียกใช้ต่อไปคือฟังก์ชัน DialogBox (สำหรับสร้าง Model ไดอล็อกบ็อกซ์) ซึ่งเมื่อเรียกใช้ฟังก์ชันนี้ Windows จะทำการสร้างไดอล็อกบ็อกซ์ให้พร้อมกับการทำงานและการจัดการกับแมสเสจจะย้ายไปทำที่ไดอล็อกบ็อกซ์ procedure ของไดอล็อกบ็อกซ์นั้น

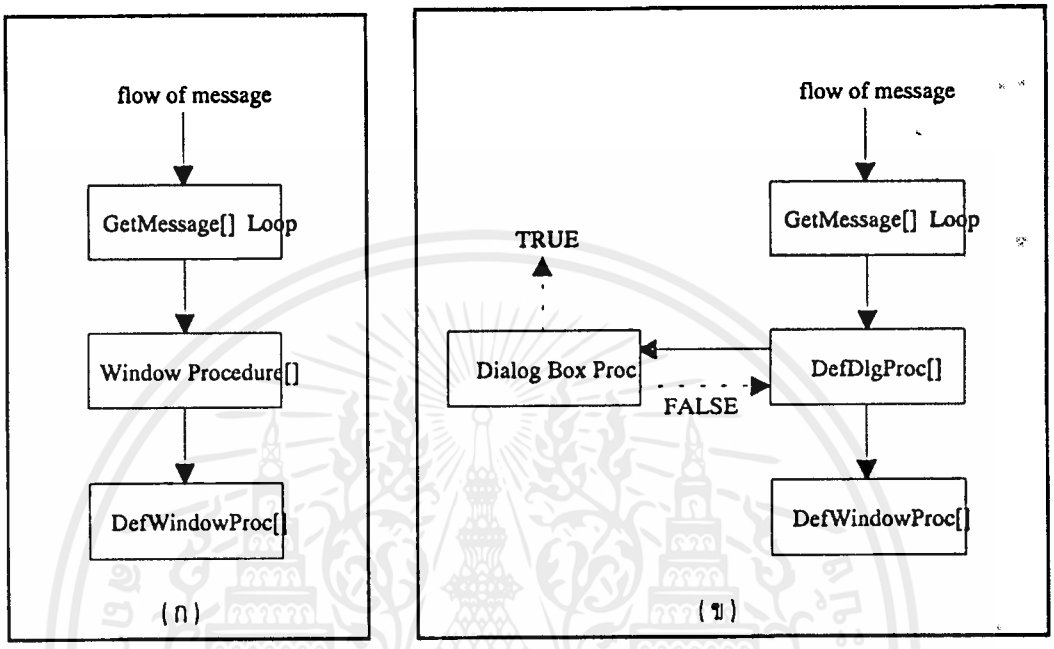
เมื่อเลิกใช้ไดอล็อกบ็อกซ์การทำงานก็จะถูกส่งคืนมาทำคำสั่งต่อจากฟังก์ชัน DialogBox ซึ่งต่อจากฟังก์ชันนี้ต้องเรียกใช้ฟังก์ชัน FreeProcInstance ฟังก์ชันนี้จะทำหน้าที่คืนหน่วยความจำจองไว้ซึ่งเป็นที่อยู่ของ instance trunk

ส่วนต่อไปก็เป็นส่วนของไดอล็อกบ็อกซ์ procedure โครงสร้างทั่วไปจะคล้ายกับ window procedure หรือเรียกอีกชื่อหนึ่งว่า callback function ตัวอย่างจะเป็นดังนี้

```
/*************************************************
            Dialog box Procedure
*************************************************/
BOOL FAR PASCAL SamPleDlgProc(HWND hDlg,unsigned wMsg, WORD wParam,LONG lParam)
{
    switch(wMsg)
        {
            case  WM_INITDIALOG:
                  return TRUE;
            case  WM_COMMAND:
                  switch( wParam )
                   {
                      case IDA_OK:
                      EndDialog(hDlg,TURE);
                      return TRUE;
                   }
            default : return FALSE;
        }
}
```

จากตัวอย่างนี้จะเห็นได้ว่าคำสั่งภายใน case statement จะจบด้วยคำสั่ง return TURE ส่วน default statement จะจบด้วยคำสั่ง return FALSE ที่ต้องทำเช่นนี้เพื่อบอกว่ามีการจัดการกับแมสเสจที่ถูกส่งมาหรือไม่ซึ่งถ้าไม่มีการจัดการกับแมสเสจ แมสเสจจะถูกส่งไปให้

กับ DefDlgProc (default dialogbox procedure) ซึ่งไดอล็อกบ็อกซ์ procedure นี้ไม่จำเป็นจะต้องเขียนขึ้นมา จะมีให้อยู่แล้ว และจะทำการเชื่อมต่อการทำงานให้เองโดยอัตโนมัติ และถ้าใช้ความสังเกต จะพบว่าไดอล็อกบ็อกซ์ procedure จะแตกต่างกับ window procedure อยู่อีกข้อหนึ่งก็คือ ไม่ต้องเรียกใช้ฟังก์ชัน DefWindowProc ในส่วนการจบการทำงานของไดอล็อกบ็อกซ์ และคืนการทำงานให้กับจุดที่เรียกใช้ไดอล็อกบ็อกซ์ต้องใช้ฟังก์ชัน EndDialog



รูปที่3 เปรียบเทียบการไหลของแมสเสจ (ก) ไดอล็อกบ็อกซ์ procedure (ข) window procedure

Module Definition File

ไฟล์นี้จะมีนามสกุลเป็น .DEF เป็นเท็กซ์ไฟล์ธรรมดา หน้าที่ของไฟล์นี้คือช่วยโปรแกรม linker สร้างไฟล์ .EXE โดยไฟล์นี้จะบอกสมบัติต่าง ๆ ของ code statement และ data statement ของโปรแกรมขนาดของ heap และ stack และข้อมูลอื่น ๆ อีกหลายอย่าง module definition file มีหน้าตาดังตัวอย่างต่อไปนี้

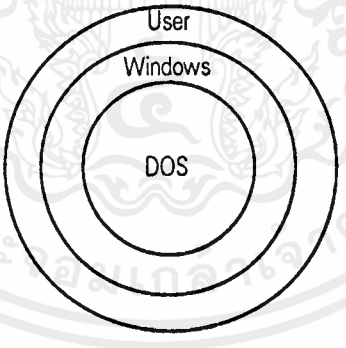
```
NAME          WinApp
DESCRIPTION 'Sample Microsoft Windows Application'
EXETYPE       WINDOWS
STUB          'winstub.exe'
CODE          PRELOAD DISCARDABLE
DATA          PRELOAD MOVEABLE
HEAPSIZE      512
STACKSIZE     5000
EXPORT        MinWndProc     @1
              AboutDlgProc   @2
```

รายละเอียดของส่วนต่าง ๆ มีดังนี้ บรรทัดแรก จะเป็นการบอก linker ว่าโมดูลนี้เป็น executable program หรือเป็น dynamic linking library ถ้าเป็น executable program บรรทัดนี้จะขึ้นต้นด้วยคำว่า NAME แต่ถ้าเป็น dynamic linking library จะขึ้นต้นด้วยคำว่า LIBRARY และทั้งสองกรณีต้องตามด้วยชื่อของโมดูล บรรทัดถัดมาเป็นข้อความที่จะถูกเก็บไว้ใน executable file ซึ่ง โดยทั่วไปเป็นข้อความเกี่ยวกับลิขสิทธิ์ และเวอร์ชันของโปรแกรม บรรทัดนี้จะขึ้นต้นด้วยคำว่า DESCRIPTION และตามด้วยข้อความที่อยู่ภายในเครื่องหมาย '' ข้อความนี้สามารถดูได้โดยใช้โปรแกรมยูทิลิตี EXEDIR

บรรทัดที่สาม มีไว้เพื่อบอกถึงชนิดของโปรแกรมที่จะถูกสร้างขึ้น ซึ่งต้องเป็นคำว่า EXETYPE WINDOWS เสมอเพราะต้องการที่สร้างแอพพลิเคชันบน Windows บรรทัดที่มีไว้เพื่อกำหนดชื่อโปรแกรมที่จะใส่ไว้ในไฟล์ .exe ที่ถูกสร้างขึ้น โปรแกรมนี้จะทำงานเมื่อมีการเรียกใช้แอพพลิเคชันนี้ที่ DOS ปกติจะเป็นไฟล์ winstub.exe ซึ่งจะพิมพ์ข้อความ "This program require Microsoft Windows" เสร็จแล้วก็จบการทำงาน บรรทัดที่ 4 และ 5 มี ไว้เพื่อบอกลักษณะของ code segment และ data segment คำที่อยู่ต่อคำว่า CODE และ DATA จะเป็น MOVEABLE FIXED และ DISCARDABLE (ส่วนความหมายเหมือนกับ memory option ของเมนูและไดอล็อกบ็อกซ์) STACKSIZE เป็นการกำหนดขนาด ของสแต็กของโปรแกรม ซึ่งค่าน้อยที่สุดที่เป็นไปได้คือ 5KB HEAPSIZE เป็นการกำหนดขนาดเริ่มต้นของ local heap ซึ่งมีขนาดได้ไม่เกิน 64KB ส่วนสุดท้าย ขึ้นต้นด้วยคำว่า EXPORTS ตามด้วยชื่อของฟังก์ชันเป็นการกำหนดชื่อของ ฟังก์ชันต่าง ๆ ซึ่งจะเป็นชื่อของ window procedure และไดอล็อกบ็อกซ์ procedure โดยจะวางชื่อของฟังก์ชันเหล่านี้ลงไป แล้วตามด้วยตัวอักษร @ และหมายเลขเรียงกันไป (ในกรณีที่มีหลายฟังก์ชัน) ดังตัวอย่างข้างต้น

สถาปัตยกรรมและการทำงานของ Windows

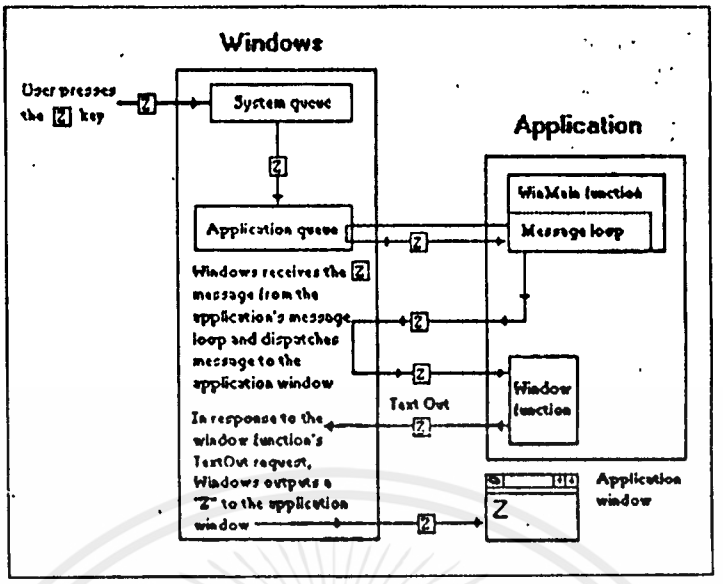
ลักษณะทั่วไปโปรแกรม Microsoft Windows เป็นโปรแกรมที่ช่วยให้ใช้งานเครื่องคอมพิวเตอร์เป็นไปได้อย่างมีประสิทธิภาพ Windows จะอยู่ในตำแหน่งที่ห่อหุ้ม DOS และขั้นอยู่ระหว่าง DOS กับ ผู้ใช้ดังรูปที่ 4



รูปที่ 4 แสดงลำดับขั้นของ Dos Windows และผู้ใช้

MESSAGE

การทำงานของ Windows นั้นส่วนหนึ่ง จะมีความเกี่ยวข้องกับการเขียนแอพพลิเคชันด้วย ภาษา C เป็นอย่างมาก และเนื่องจากระบบของ Windows มีการทำงานโดยการใช้แมสเสจอย่าง ที่กล่าวมาแล้ว การเขียนแอพพลิเคชั่นก็เป็นลักษณะการจัดการกับแมสเสจ ซึ่งเรียกว่า message-driven programming ลักษณะการจัดการของแมสเสจของ Windows ดังแสดงในรูปที่ 5



รูปที่5 การจัดการกับแมสเสจ

จากรูปที่5 เป็นตัวอย่างของการจัดการกับแมสเสจ เช่นเมื่อผู้ใช้กดคีย์บอร์ด จะเกิดแมสเสจขึ้น Windows จะเก็บแมสเสจไว้ที่ system queue และจัดการแบ่งแมสเสจเหล่านั้นว่าเป็นแมสเสจของวินโดว์ไหนจะนำไปวางไว้ที่ application queue ของแต่ละแอพพลิเคชั่น ในส่วน ของแอพพลิเคชั่น ส่วน message loop ซึ่งอยู่ในฟังก์ชั่น WinMain จะทำการรับแมสเสจมาจาก application queue เพื่อจัดการบางอย่างกับแมสเสจแล้วส่งคืนไปให้กับ Windows โปรแกรม Windows จะจัดการส่งแมสเสจไปยัง window function (หรือ window procedure) ที่เหมาะสมจะจัดการกับแมสเสจนั้นซึ่งภายใน window function จะมีคำสั่งที่ให้ Windows ทำงานเมื่อมีแมสเสจที่ต้องการเข้ามา

คำศัพท์เกี่ยวกับเรื่องWindows

MESSAGE เป็นโครงสร้างของข้อมูลชนิดประเภทหนึ่ง ประกอบด้วยฟิลด์หลายฟิลด์ ข้อมูลในฟิลด์จะบอกว่าเกิดเหตุการณ์อะไรขึ้น เกิดกับวินโดว์ไหนที่เวลาเท่าไร และขณะที่เกิดนั้นเคอร์เซอร์ของเมาส์อยู่ที่ตำแหน่งใดซึ่งเมื่อเกิดเหตุการณ์ต่าง ๆ ขึ้น Windows จะทำการเปลี่ยนเหตุ การณ์ต่าง ๆ นั้นเป็นแมสเสจ

RESOURCE เป็นทรัพยากรต่าง ๆ ที่Windows ต้องทำการจัดสรรหรือแบ่งปันให้กับแต่ละ แอพพลิเคชันใช้ และเป็นสิ่งที่วินโดว์ต้องทำการดูแล เช่นคีย์บอร์ด เมาส์ พอร์ตต่าง ๆ จอภาพ หน่วยความจำ และนอกจากนี้ยังรวมถึง resource ที่เป็นของแต่ละแอพพลิเคชัน แต่Windows ต้องเข้าไปจัดการให้ เช่นพวกเมนู, ไดอล็อกบ็อกซ์, เคอร์เซอร์, ไอคอน และบิตแมปเป็นต้น

DIALOG BOX เป็นวินโดว์ชนิดหนึ่งบน Windows ซึ่งหน้าที่ทั่วไปก็คือทำหน้าที่บอกข่าวสารและรับข้อมูลจากผู้ใช้ภายใน ไดอล็อกบ็อกซ์จะประกอบไปด้วย control ต่าง ๆ

CONTROL เป็นวัตถุต่าง ๆ ซึ่งปรากฏในไดอล็อกบ็อกซ์ซึ่งทำหน้าที่เป็นตัวรับข่าวสารจากผู้ใช้ หรือทำการแสดงข้อมูลหรือข่าวสารต่าง ๆ ให้กับผู้ใช้ ตัวอย่างของ control เช่น pushbotton, listbox, edit control, static text

ICON เป็นสัญลักษณ์ที่เป็นรูปภาพซึ่งจะใช้แทนตัวโปรแกรมต่าง ๆ ทุกโปรแกรมที่ทำงานบนWindows สามารถที่จะมีไอคอนเฉพาะตัวได้ ซึ่งการเรียกให้แอพพลิเคชันทำงานก็ใช้วิธีสั่งจากไอคอนนั่นเอง

BITMAP เป็นอาเรย์ของบิตซึ่งถูกเก็บรวมกันใช้ในการเก็บข้อมูลซึ่งเป็นรูปภาพ ซึ่งแต่ละจุดของรูปภาพก็คือข้อมูลแต่ละบิตในบิตแมปนั่นเอง โดยปกติจะถูกเก็บไว้เป็นไฟล์

Graphics Device Interface (GDI)

รูปแบบกราฟิกบนวินโดวส์ถูกจัดการโดยฟังก์ชั่นจากโมดูล GDI.EXE (ถึงแม้ว่าจะมีบางฟังก์ชั่นในการวาดจะถูกเรียกจากไฟล์ USER.EXE) โมดูล GDI.EXE จะทำหน้าที่เรียกรูทีนจากไฟล์ไดรเวอร์ต่างๆ ที่มี .DRV สำหรับจอภาพหนึ่งไฟล์ และอาจมีไฟล์ไดรเวอร์อื่นๆ อีกสำหรับควบคุม เครื่องพิมพ์หรือเครื่องพลอตเตอร์ ไฟล์ไดรเวอร์สำหรับจอภาพจะทำการติดต่อกับฮาร์ดแวร์ของจอภาพ สำหรับจอภาพหรือเครื่องพิมพ์ที่แตกต่างกันก็ต้องใช้ไดรเวอร์ที่ต่างกันด้วย

ระบบ GDI นี้ถูกสร้างขึ้นเพื่อวินโดวส์จะสามารถพิจารณาจากตัวไดรเวอร์ว่ามีอะไรบ้างที่ไดรเวอร์นั้นสามารถจัดการได้และจัดการไม่ได้ ตัวอย่าง เช่น ถ้าฮาร์ดแวร์ของการแสดงผลประกอบด้วยโคโปรเซสเซอร์ทางด้านกราฟิกที่สามารถวาดภาพวงรี ดังนั้น GDI สามารถใช้ประโยชน์จากมันได้ หรืออีกทางหนึ่ง ตัวโมดูล GDI เองนั้นจะต้องคำนวณจุดทั้งหมดของวงรีนั้นแล้วส่งผ่านจุดเหล่านั้นแก่ไดรเวอร์ต่อไป

เนื่องจากอุปกรณ์สำหรับการแสดงผลมีจำนวนมากมายที่สามารถ ต่อเข้ากับเครื่องพีซีทั่วไปได้ จุดประสงค์หลักอย่างหนึ่งของ GDI ก็คือการที่ทำให้ระบบกราฟิกเป็นอิสระจากอุปกรณ์ ต่างๆ เช่น จอแสดงผล, เครื่องพิมพ์, และเครื่องพลอตเตอร์เป็นต้น โปรแกรมในระบบวินโดวส์นั้นควรที่จะสามารถ ทำงานไปได้โดย ปราศจากปัญหาใดๆเกี่ยวกับอุปกรณ์ที่แสดงผลทางกราฟิกที่วินโดว์รู้จัก ระบบ GDI ทำให้บรรลุเป้าหมายนี้โดยการจัดให้มีความสามารถที่จะแยกโปรแกรมต่างๆ จากคุณสมบัติเฉพาะที่แตกต่างกันของอุปกรณ์แต่ละชนิด ในแนวทางนี้เป็นแนวทางเดียวกับภาษาที่มีไว้สำหรับการเขียนโปรแกรมทางด้านกราฟิกแบบ device-independent โดยทั่วไปนั่นเอง แต่ระบบ GDI ของวินโดวส์นั้นจะแตกต่างไปตรงที่ความสามารถที่จะจัดการในระดับ Pixelได้ดีกว่า นั้นเอง

ในโลกของอุปกรณ์สำหรับการแสดงผลทางด้านกราฟิกนั้นสามารถแบ่งได้อย่างกว้างๆ ได้ 2 ประเภทคือ raster device และ vector device อุปกรณ์ในการแสดงผลส่วนใหญ่ของพีซีนั้น จะเป็นประเภท raster device ซึ่งหมายความว่าอุปกรณ์ประเภทนี้จะแสดงผลภาพด้วยแพทเทินของจุด อุปกรณ์ในจำพวกนี้ก็ได้แก่จอแสดงผล เครื่องพิมพ์ชนิด dot matrix และชนิด laser ส่วน vector device นั้นจะสร้างภาพโดยการใช้เส้น ตัวอย่างเช่น เครื่องพล็อตเตอร์ เป็นต้น

ถึงแม้ว่าจอแสดงผล และเครื่องพิมพ์ส่วนใหญ่จะเป็น raster device ภาษาทางด้านกราฟิกส่วนใหญ่ก็ยังคงใช้ vector ในการแทนภาพนี่ก็หมายความว่าโปรแกรมที่ใช้ภาษาทางกราฟิกเหล่านี้อยู่ในชั้นที่แยกออกมาจากฮาร์ดแวร์อย่างชัดเจน อุปกรณ์แสดงผลใช้ pixel ในการแสดงภาพ แต่โปรแกรมไม่ได้ใช้ในรูปแบบของ pixel ในขณะที่เมื่อใช้ Windows GDI ฟังก์ชั่นในการ วาดภาพแบบ high-level vector หรือสามารถจะวาดในแบบ low-level pixel ก็ได้ ในแง่มุมนี้ Windows GDI จึงเปรียบเสมือนเป็นภาษาติดต่อทางด้านกราฟิก และภาษาซีก็เป็นภาษาทางการเขียนโปรแกรม เป็นที่รู้กันเป็นอย่างดีมีความ portability สูงในระบบปฏิบัติ การ และสภาพแวดล้อมที่ต่าง ๆ กัน นอกจากนั้น ภาษาซียังสามารถให้โปรแกรมเมอร์เรียกใช้ system function ในระดับ low-level ได้ซึ่งคุณลักษณะนี้ไม่มีปรากฏในภาษาระดับสูงอื่น ๆ ภาษาซีจึงมีลักษณะคล้ายเป็น "high-level assembly language" และ GDI มีลักษณะเป็น การติดต่อในระดับสูงกับฮาร์ดแวร์ที่แสดงผลกราฟิก โดยปกติแล้วการอ้างถึงพิกัดตำแหน่งในแบบจุด pixel ภาษาทางด้านกราฟิกอื่นๆ ส่วนใหญ่ จะใช้ระบบพิกัดเสมือน (virtual coordinate syetem) ที่มีแกน horizontal และแกน vertical

THE DEVICE CONTEXT

เมื่อต้องการวาดภาพสู่อุปกรณ์แสดงผลกราฟิก (เช่นจอภาพ หรือเครื่องพิมพ์) เริ่มแรกจะต้องได้รับค่าแฮนเดิล (handle) สำหรับ device context (หรือ DC) ในการได้ค่าแฮนเดิล นี้มาวินโดวส์จะให้สิทธิในการใช้ดีไวซ์(device)นั้น แล้วจะต้องใช้ค่าแฮนเดิลนี้เป็นพารามิเตอร์หนึ่ง ในการเรียกใช้ฟังก์ชั่น GDI เพื่อที่จะบอกให้วินโดวส์รู้ว่าดีไวซ์ใดที่ต้องการจะวาดภาพลงไป ดีไวซ์คอนเทกซ์ประกอบด้วยค่าแอทริบิว(attribute) ต่างๆ มากมายที่จะใช้พิจารณาการ ทำงานของฟังก์ชั่น GDIว่าทำงาน

อย่างไรบนดีไวซ์ตัวนี้ ค่าแอทริบิวเหล่านี้จะทำให้ค่าพารามิเตอร์ ของฟังก์ชั่น GDI ที่ต้องใส่ในการเรียกใช้เหลือเพียงแค่ค่าพิกัดเริ่มต้นหรือขนาดเท่านั้นที่วินโดวส์ ต้องการในการแสดงผลใดๆ บนดีไวซ์นั้น ตัวอย่างเช่น ในการเรียกใช้ฟังก์ชั่น TextOut ค่า พารามิเตอร์ที่ต้องการมีเพียงค่าดีไวซ์คอนเทกซ์แฮนเดิล จุดพิกัดเริ่มต้น ข้อความ และความยาวของข้อความนั้นเท่านั้น โดยไม่ต้องกำหนดแบบของตัวอักษร สีของตัวอักษร สีของพื้นหลังตัวอักษร และการเว้นช่องว่างระหว่างตัวอักษร เพราะว่าค่าต่างๆ เหล่านี้เป็นส่วนหนึ่งในดีไวซ์คอนเทกซ์ เมื่อต้องการที่จะเปลี่ยนแปลงค่าเหล่านี้เราจะเรียกใช้ฟังก์ชั่นที่ทำการเปลี่ยนแปลงค่าในดีไวซ์ คอนเทกซ์ การเรียกใช้ฟังก์ชั่น TextOut ในครั้งต่อ ๆ ไปก็จะใช้ค่าที่ถูกเปลี่ยนใหม่นั้น

การได้มาซึ่งค่าแฮนเดิลของดีไวซ์คอนเทกซ์

วินโดวส์มีวิธีการให้ได้มาซึ่งค่าดีไวซ์คอนเทกซ์แฮนเดิลอยู่หลายรูปแบบ ถ้าต้องการได้ค่าดีไวซ์ คอนเทกซ์แฮนเดิล ขณะทำการประมวลผลแมสเซสอยู่ก็ควรจะทำการ release(หรือdelete) ก่อนการออกจากวินโดว์ฟังก์ชั่นนั้น หลังจากที่ release ไปแล้ว ก็จะใช้ไม่ได้อีกต่อไป วิธีการโดยทั่ว ๆไปที่จะได้ค่าดีไวซ์คอนเทกซ์แฮนเดิลและการปล่อยคืนนั้นเกี่ยวกับการเรียกใช้ BeginPaint และ EndPaint ขณะทำการประมวลผลแมสเซส WM_PAINT

```
hdc = BeginPaint (hwnd, &ps) ;
        [other program lines]
EndPaint (hwnd, &ps) ;
```

ตัวแปร ps เป็นตัวแปรโครงสร้างชนิด PAINTSTRUCT ในฟิลด์ hdc ของโครงสร้างนี้จะเป็นค่าแฮนเดิลสำหรับดีไวซ์คอนเทกซ์ที่ BeginPaint ส่งคืนค่ามาให้ ในโครงสร้าง PAINTSTRUCT นี้จะมีตัวแปรโครงสร้าง RECT (สี่เหลี่ยม) ที่มีชื่อว่า rcPaint ที่จะกำหนดพื้นที่ invalid ของ Client areaของวินโดว์นั้น ด้วยดีไวซ์คอนเทกซ์แฮนเดิลที่ได้รับมาจาก BeginPaint นั้นจะสามารถใช้วาดได้เฉพาะในพื้นที่ที่ว่านี้เท่านั้น การเรียกใช้ BeginPaint จะทำให้พื้นที่เหล่านี้ valid วินโดวส์โปรแกรมสามารถได้ค่าดีไวซ์คอนเทกซ์แฮนเดิล ขณะที่ประมวลผลกับแมสเซสอื่นที่นอกเหนือไปจาก WM_PAINT:

```
hdc = GetDC (hwnd) ;
        [other program lines]
ReleaseDC (hwnd, hdc) ;
```

ดีไวซ์คอนเทกซ์นี้จะใช้กับไคลแอนท์เอเรียของวินโดว์ที่มีค่าแฮนเดิลเป็น hwnd ความ แตกต่างที่สำคัญระหว่างการเรียกใช้ในรูปแบบนี้ กับการใช้ BeginPaint และ EndPaint ก็คือค่าแฮนเดิลที่ได้จาก GetDC จะสามารถวาดได้ทั้งไคลแอนท์เอเรีย อย่างไรก็ตาม GetDC และ Re-leaseDC จะไม่ทำการ validate พื้นที่ที่ invalid ของไคลแอนท์เอเรีย วินโดวส์โปรแกรม สามารถได้ค่าดีไวซ์คอนเทกซ์แฮนเดิล ที่ใช้ได้กับทั้งวินโดว์ ไม่เพียงแต่เฉพาะพื้นที่ไคลแอนท์เอเรียของวินโดว์เท่านั้น

```
hdc = GetWindowDC (hwnd) ;
        [other program lines]
ReleaseDC (hwnd, hdc) ;
```

ดีไวซ์คอนเทกซ์นี้ประกอบด้วยทั้ง caption bar, menu, scroll bars, และขอบนอกเหนือไปจากพื้นที่ไคลแอนท์

เอเรีย ฟังก์ชั่น GetWindowDC นี้ไม่ค่อยใช้บ่อยนัก ถ้าต้องการทดลอง ใช้ก็ควรทดลองด้วยการดักแมสเซส WM_NCPAINT ("nonclient paint") ซึ่งจะป้องกันวินโดวส์ จากการวาดในพื้นที่ nonclient area ของวินโดว์ การเรียกใช้ฟังก์ชั่น BeginPaint, GetDC และ GetWindowDC นั้นจะได้รับดีไวซ์คอน เทกซ์ที่สัมพันธ์กับวินโดว์ใดวินโดว์หนึ่ง เราสามารถที่จะได้ดีไวซ์คอนเทกซ์สำหรับทั้งจอภาพโดย การเรียกใช้ฟังก์ชั่น CreateDC

```
hdc = CreateDC (lpszDriver, lpszDevice, lpszOutput, lpData) ;
        [other program lines]
DeleteDC (hdc) ;
```

Image Smoothing

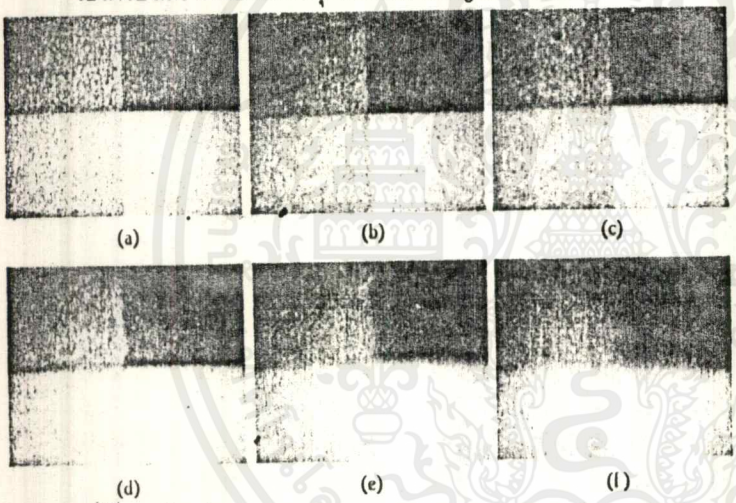
การ smooth ภาพนี้ใช้สำหรับการขจัดผลข้างเคียงที่เกิดขึ้นในภาพดิจิตอลที่เป็นผลมาจากระบบ การ sampling ข้อมูล หรือแชแนลการสื่อสารที่ไม่ดีพอ ในโครงงานชิ้นนี้ได้เลือกวิธีที่ทำงานได้เร็วที่สุด และให้ผลเป็นที่ดีในระดับที่เหมาะสมกับงานนี้ ก็คือวิธี Neighborhood Averaging

Neighborhood Averaging

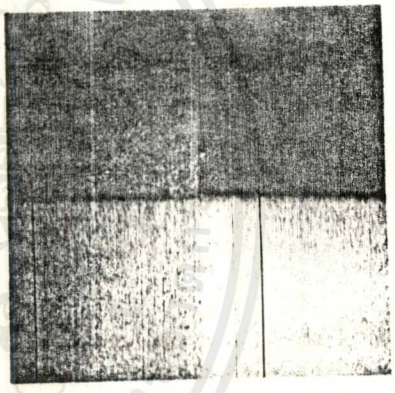
Neighborhood Averaging เป็นเทคนิคแบบ spatial-domain ที่ตรงไปตรงมาสำหรับ การ smooth ภาพ โดยที่ภาพ N x N จุดมีฟังก์ชั่นเป็น f(x,y) วิธีการนี้จะสร้างภาพที่ถูก smooth เป็นภาพที่แสดงด้วยฟังก์ชั่น g(x,y) โดยที่ระดับของ gray-level ที่ทุก ๆ จุด (x,y) ได้มาจาก การเฉลี่ยของค่า gray-level ของจุดที่ถูกกำหนดเป็น neighborhood ของจุด (x,y) นั้น อีกนัยหนึ่ง ภาพที่ถูก smooth จะได้มาจากความสัมพันธ์ดังนี้คือ

$$g(x,y) = \frac{1}{M} \sum_{(n,m)\in S} f(n,m) \qquad (1)$$

สำหรับ x,y = 0, 1, ..., N-1 และ S เป็นเซ็ตของพิกัดของจุดที่ถูกกำหนดเป็น neighborhood ของจุด (x,y) รวมทั้งจุด (x,y) เองด้วย และ M คือจำนวนจุดทั้งหมดใน neighborhood นั้น



รูปที่ 1 ตัวอย่างของ Neighborhood Averaging (a) Noisy image (b) - (f) ใช้สมการที่ 1 ด้วยจุดรอบข้าง n x n โดย n = 3, 5, 9, 15, 31 ตามลำดับ



รูปที่ 2 แสดงการ Smooth โดยสมการที่ 2 ด้วยจุดรอบข้าง 9 x 9 และ T = 10

สำหรับ neighborhood ที่ใช้นั้นอาจทำให้เกิด blurring effect ขึ้นที่เป็นผลมาจากการ ทำ neighborhood averaging นั้นคือบางส่วนของภาพที่เราต้องการให้มีความคมชัด เช่นขอบของวัตถุจะพลอยถูก smooth ไปด้วย ผลนี้สามารถลดลงได้โดยการใช้ thresholding procedure ดังนี้คือ แทนที่เราจะใช้สมการที่ (1) เราจะใช้ g(x,y) ใหม่ดังนี้คือ

$$g(x,y) = \begin{cases} \frac{1}{M} \sum_{(n,m)\in S} f(n,m) & \text{if } \left| f(x,y) - \frac{1}{M} \sum_{(n,m)\in S} f(n,m) \right| < T \\ f(x,y) & \text{Otherwise} \end{cases} \qquad (2)$$

ซึ่ง T จะเป็นตัวกำหนดค่า threshold ที่ไม่เป็นลบ ซึ่งวิธีการนี้จะสามารถลด blurring ได้โดยการไม่ทำการเปลี่ยนแปลงค่าในบริเวณที่มีความแตกต่างของค่า gray-level สูง โดยทั่วไปแล้ว เราคาดว่าความแตกต่างมาก ๆ นี้ก็คือบริเวณที่เป็นขอบต่างๆนั่นเอง ดังนั้นสมการที่ 2 จึงสามารถลด blurring ที่เกิดกับขอบได้ ส่วนบริเวณอื่นของภาพก็จะถูกคำนวณในรูปแบบปกติ

ภาคผนวก

```
/*****************************************************************
   PROGRAM: Angelo.c
   Author: Prakot Limpakarn            321177
           Trasapong Thaiupathump      321100
   PURPOSE: Computer Assisted Suspect Sketching Outfit
           version 2.00 for Windows
           KMITL 1992-1993
   FUNCTIONS:
           WinMain() - calls initialization function, processes message loop
           InitApplication() - initializes window data and registers window
           InitInstance() - saves instance handle and creates main window
           MainWndProc() - processes messages
           About() - processes messages for "About" dialog box
   COMMENTS: Windows can have several copies of your application running at the
same time. The variable hInst keeps track of which instance this application is so that
processing will be to the correct window.
*****************************************************************/

#include <windows.h>
#include <commdlg.h>
#include <stdlib.h>
#include <string.h>
#include "bwcc.h"
#include "define.h"   // header file that contain all definition that use while execution
#include "function.h" // header file for function definition
#include "global.h"   // header file for global variables


/*****************************************************************
   FUNCTION: WinMain(HANDLE, HANDLE, LPSTR, int)
   PURPOSE: calls initialization function, processes message loop
   COMMENTS: Windows recognizes this function by name as the initial entry point for
the program. This function calls the application initialization routine, if no other instance
of the program is running, and always calls the instance initialization routine. It then
executes a message retrieval and dispatch loop that is the top-level control structure
for the remainder of execution. The loop is terminated when a WM_QUIT message is
received, at which time this function exits the application instance by returning the value
passed by PostQuitMessage(). If this function must abort before entering the message
loop, it returns the conventional value NULL.
*****************************************************************/

int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                   LPSTR lpCmdLine, int nCmdShow)
{
   MSG   msg;      // message

   if (!hPrevInstance)     // Other instances of app running?
    if (!InitApplication(hInstance))          // Initialize shared things
      return (FALSE);          // Exits if unable to initialize
   // Perform initializations that apply to a specific instance
   if (!InitInstance(hInstance, nCmdShow))
         return (FALSE);

   // Acquire and dispatch messages until a WM_QUIT message is received.
   while (GetMessage(&msg,    // message structure
                  NULL,   // handle of window receiving the message
                  NULL,   // lowest message to examine
                  NULL))  // highest message to examine
   {
         TranslateMessage(&msg); // Translates virtual key codes
         DispatchMessage(&msg); // Dispatches message to window
   }
   App_UnRegisterClasses();
   return (msg.wParam);        // Returns the value from PostQuitMessage
}

/*****************************************************************
   FUNCTION: InitApplication(HANDLE)
   PURPOSE: Initializes window data and registers window class
   COMMENTS: This function is called at initialization time only if no other instances of
the application are running. This function performs initialization tasks that can be done
once for any number of running instances. In this case, we initialize a window class by
filling out a data structure of type WNDCLASS and calling the Windows RegisterClass()
function.ce alna of this application use the same window class, we only need to do this
when the first instance is initialized.
*****************************************************************/

BOOL InitApplication(HANDLE hInstance)
{
   WNDCLASS wc;
   HBITMAP        hPkBackground;

   hMyCursor       = LoadCursor( NULL, IDC_ARROW );
   hPkBackground   = LoadBitmap(hInstance,MAKEINTRESOURCE(5000));
   wc.style        = CS_HREDRAW | CS_VREDRAW; // Class style(s)
   wc.lpfnWndProc  = MainWndProc;     // Function to retrieve messages
   wc.cbClsExtra   = 0;       // No per-class extra data
   wc.cbWndExtra   = 0;       // No per-window extra data
   wc.hInstance    = hInstance;          // Application that owns the class
   wc.hIcon        = LoadIcon(hInstance, "MyIcon");
   wc.hCursor      = hMyCursor;
   wc.hbrBackground= GetStockObject(DKGRAY_BRUSH);
   wc.lpszMenuName= szAppName;       // Name of menu resource in .RC file.
   wc.lpszClassName= szAppName;      // Name used in call to CreateWindow.
   /* Register the window class and return success/failure code. */
         if(!RegisterClass(&wc)) return (FALSE);
   wc.lpfnWndProc  = PkOwnerDrawRadioProc; // Function to retrieve messages
   wc.cbWndExtra   = sizeof(WORD);    // flag on/off
   wc.hCursor      = LoadCursor(hInstance,"HAND");
   wc.lpszClassName= szPkOwnerDrawRadioButtonClass;
         if(!RegisterClass(&wc)) return (FALSE);
   wc.lpfnWndProc  = PkRadioProc;     // Function to retrieve messages
   wc.cbWndExtra   = sizeof(WORD);    // flag on/off
   wc.hCursor      = hMyCursor;
   wc.hbrBackground = GetStockObject(LTGRAY_BRUSH);
  wc.lpszClassName = szPkRadioButtonClass;
         if(!RegisterClass(&wc)) return (FALSE);
   wc.lpfnWndProc  = PictureProc;     // Function to retrieve messages
   wc.hIcon        = NULL;
   wc.hbrBackground = GetStockObject(WHITE_BRUSH);
   wc.lpszClassName= szPicClass;
         if(!RegisterClass(&wc)) return (FALSE);
   wc.lpfnWndProc  = PictureSelectFrameProc;     // Function to retrieve messages
   wc.hCursor      = hMyCursor;
   wc.hbrBackground= CreatePatternBrush(hPkBackground);
   wc.lpszClassName= szPicSelectFrameClass;
         if(!RegisterClass(&wc)) return (FALSE);
   wc.lpfnWndProc  = PictureSelectProc;  // Function to retrieve messages
   wc.hbrBackground= GetStockObject(WHITE_BRUSH);
   wc.lpszClassName= szPicSelectClass;
```

```
    if(!RegisterClass(&wc)) return (FALSE);

    wc.lpfnWndProc   = ComponentDlgProc; // Function to retrieve messages
    wc.lpszClassName= szToolDlgClass;
        if(!RegisterClass(&wc)) return (FALSE);
    wc.lpfnWndProc   = ToolOperateDlgProc; // Function to retrieve messages
    wc.hbrBackground= CreatePatternBrush(hPkBackground);
    wc.hCursor   = hMyCursor;
    wc.lpszClassName= szToolOpDlgClass;
        if(!RegisterClass(&wc)) return (FALSE);
    wc.lpfnWndProc   = SelectComponentProc; // Function to retrieve messages
    wc.cbWndExtra    = sizeof(WORD);        // flag on/off
    wc.hCursor   = LoadCursor(hInstance,"HAND");
    wc.lpszClassName = szSelectComponent;
        if(!RegisterClass(&wc)) return (FALSE);
    return(TRUE);
}

/*****************************************************************
    FUNCTION: InitInstance(HANDLE, int)
    PURPOSE: Saves instance handle and creates main window
    COMMENTS: This function is called at initialization time for every instance of this
application. This function performs initialization tasks that cannot be shared by multiple
instances. In this case, we save the instance handle in a static variable and
create and display the main program window.
*****************************************************************/
BOOL InitInstance(HANDLE hInstance, int nCmdShow)
{
    RECT  rc;
    int   y_start,y1_start;
    int   caption_height;
    int   i;

    /* Save the instance handle in static variable, which will be used in
      many subsequent calls from this application to Windows.   */
    hInst = hInstance;
    /* Create a main window for this application instance.        */
  hArrowCursor = LoadCursor(NULL, IDC_ARROW);
  hMyCursor = hArrowCursor;
    MainhWnd = CreateWindow(
        szAppName,          /* See RegisterClass() call */
        "Angelo",           /* Text for window title bar */
        WS_CLIPCHILDREN|
        WS_OVERLAPPEDWINDOW,    /* Window style */
        CW_USEDEFAULT,          /* Default horizontal position */
        CW_USEDEFAULT,          /* Default vertical position */
        CW_USEDEFAULT,          /* Default width */
        CW_USEDEFAULT,          /* Default height */
        NULL,               /* Overlapped windows have no parent */
        NULL,               /* Use the window class menu */
        hInstance,          /* This instance owns this window */
        NULL                /* Pointer not needed */
    );

    /* If window could not be created, return "failure" */

    if (!MainhWnd)
        return (FALSE);
    /* Make the window visible; update its client area; and return "success" */
    ShowWindow(MainhWnd, SW_SHOWMAXIMIZED); /* Show the window */
    UpdateWindow(MainhWnd);  /* Sends WM_PAINT message */
    GetClientRect(MainhWnd,&rc);
    InitialData();
    /* create window to handle picture operation */

    hPicWnd = CreateWindow(szPicClass,"",
            WS_CHILD|WS_VISIBLE|WS_CLIPSIBLINGS,
            0,0,rc.right-300,rc.bottom,
            MainhWnd,NULL,hInst,NULL);

    rc.left = rc.right-276;
    caption_height = GetSystemMetrics(SM_CYCAPTION);
    y_start = ( rc.bottom - 432 - (2*caption_height) )/2;
    if(y_start < 0) y_start = 0;
    y1_start = y_start + caption_height + 231;

    hPicSelectFrame= CreateWindow(szPicSelectFrameClass,"",
            WS_CHILD|WS_CLIPSIBLINGS,
            rc.left,0+y_start,252,231+caption_height,
            MainhWnd,NULL,hInst,NULL);

    hToolDlg1 = CreateWindow(szToolDlgClass,"Component",
            WS_CHILD|WS_CAPTION|WS_VISIBLE|
            WS_CLIPSIBLINGS,rc.left,0+y_start,
            252,231+GetSystemMetrics(SM_CYCAPTION),
            MainhWnd,NULL,hInst,NULL);

    hToolDlg2 = CreateWindow(szToolOpDlgClass,"Edit Manager",
            WS_CHILD|WS_CAPTION|WS_CLIPSIBLINGS,
            rc.left,231+y_start+GetSystemMetrics(SM_CYCAPTION),
            252,221,MainhWnd,NULL,hInst,NULL);

    lpfnEditStyleProc = MakeProcInstance ((FARPROC) EditStyleProc, hInst);

    for( i = 0; i <= 9; i++)
     {
      if( (i == 3) ||  // Right Ear
        (i == 5) ||  // Right Eyebrow
        (i == 7)     // Right Eye
        ) i++;

      COMPONENT = i;
      hSelectComponent[i] = CreateWindow(szSelectComponent,ComponentCaption[i],
            WS_CHILD|WS_CAPTION|WS_CLIPSIBLINGS,
            rc.left,y1_start,252,201+caption_height,
            MainhWnd,NULL,hInst,NULL);
     }

    return(TRUE);    /* Returns the value from PostQuitMessage */
}

/*****************************************************************
    FUNCTION: MainWndProc(HWND, unsigned, WORD, LONG)
    PURPOSE: Processes messages
    MESSAGES:
        WM_COMMAND      - application menu (About dialog box)
        WM_DESTROY      - destroy window
    COMMENT: To process the IDM_ABOUT message, call MakeProcInstance() to get
```

```
the current instance address of the About() function. Then call Dialog box which will
create the box according to the information in your generic.rc file and turn control over
to the About() function. When it returns, free the intance address.
*************************************************************************/
LONG FAR PASCAL _export MainWndProc(HWND hWnd, UINT message,
                        UINT wParam, LONG lParam)
{
    FARPROC     lpfnAboutDlgProc;       /* pointer to the "About" function */
    FARPROC     lpfnPersonDlgProc;
    FARPROC     lpfnEditDlgProc;
    RECT        rc;
    int         y_start,y1_start,i;
    static int  caption_height;
    static BOOL bNeedSave = FALSE;
    static char szFileName [_MAX_PATH];
    static char szTitleName [_MAX_FNAME + _MAX_EXT];
    static int  iOffset;
    LONG        lSelect;
    WORD        wEnable;

    switch (message)
    {
    case WM_COMMAND: /* message: command from application menu */
        switch(wParam)
        {
        case IDM_ABOUT: /* show about box */
            SetCapture(hWnd);
            SetCursor(LoadCursor(NULL,IDC_WAIT));
            lpfnAboutDlgProc = MakeProcInstance(AboutDlgProc, hInst);
            DialogBox(hInst,            // current instance
            "AboutBox",                 // resource to use
            hWnd,                       // parent handle
            lpfnAboutDlgProc);                  // AboutDlgProc()
            FreeProcInstance(lpfnAboutDlgProc);
            break;
        case IDM_NEW: /* initial the variables properly */
            int i;
            if (bNeedSave && IDCANCEL ==
             AskAboutSave (hWnd, szTitleName))
            return 0;
            szFileName [0] = '\0';
            szTitleName [0] = '\0';
            DoCaption (hWnd, szTitleName);
            bNeedSave = FALSE;
            for (i=0;i<=9;i++)
                nCurrentStyle[i] = -1;
            bNoCompSelect = TRUE;
            COMPONENT = -1;
            CheckRadioButton(hToolDlg1,IDC_HAIR,IDC_MOUTH,NULL);
            CheckRadioButton(hToolDlg1,IDC_SELECT,IDC_EDIT,NULL);
            UpdateWindow(hToolDlg1);
            InvalidateRect(hPicWnd,NULL,TRUE);
            if(IsWindowVisible(hToolDlg2))
                ShowWindow(hToolDlg2,SW_HIDE);
            break;

        case IDM_OPEN: /* open angelo data file to display on the screen */
            if (bNeedSave && IDCANCEL ==
             AskAboutSave (hWnd, szTitleName))
            return 0;
            if (PopFileOpenDlg (hWnd, szFileName, szTitleName))
            {
                if (!FileRead (szFileName))
                {
                    OkMessage (hWnd, "Could not read file %s!",
                                szTitleName) ;
                    szFileName [0] = '\0';
                    szTitleName [0] = '\0';
                }
            }
            DoCaption (hWnd, szTitleName);
            bNeedSave = FALSE;
            return 0;
            break;

    case IDM_SAVE: /* save the component code of the current picture to file */
        if (szFileName [0])
        {
            if (FileWrite (szFileName))
            {
                bNeedSave = FALSE;
                return 1;
            }
            else
            OkMessage (hWnd, "Could not write file %s",
                        szTitleName) ;
            return 0;
        }
                    // fall through
    case IDM_SAVEAS:
        if (PopFileSaveDlg (hWnd, szFileName, szTitleName))
        {
            DoCaption (hWnd, szTitleName);
            if (FileWrite (szFileName))
            {
                bNeedSave = FALSE;
                return 1;
            }
            else
                OkMessage (hWnd, "Could not write file %s",
                        szTitleName) ;
        }
        return 0;

    case IDM_PRINT:
        /* print current picture that display on the screen out */
        if (PrintMyPage(hWnd))
            MessageBox(hWnd,
                    "Could not print page",
                    "Angelo Application",
                    MB_OK|MB_ICONEXCLAMATION);
        return 0;
    case IDM_EXIT:      /* process exit message */
        DestroyWindow(hWnd);
        break;

    case IDM_PERSON: /* edit menu commands */
        lpfnPersonDlgProc = MakeProcInstance(Person, hInst);
```

```
            DialogBox(hInst,            /* current instance          */
            "PersonFile",               /* resource to use           */
            hWnd,                       /* parent handle             */
            lpfnPersonDlgProc);         /* Person() instance address */
            FreeProcInstance(lpfnPersonDlgProc);
            break;

        case IDM_HAIR:
        case IDM_CHIN:
        case IDM_EAR:
        case IDM_EBROW:
        case IDM_EYE:
        case IDM_NOSE:
        case IDM_MOUTH:
            /* choose component to edit */
            if(IsWindowVisible(hToolDlg1))
                SendMessage(hToolDlg1,WM_COMMAND,wParam-190,NULL);
            break;

        case IDM_MHAIR:
        case IDM_MCHIN:
        case IDM_MLEAR:
        case IDM_MREAR:
        case IDM_MLEYE:
        case IDM_MREYE:
        case IDM_MLEBROW:
        case IDM_MREBROW:
        case IDM_MNOSE:
        case IDM_MMOUTH:
            /* choose component to move */
            MoveOperation(hWnd,message,wParam,lParam);
            return 0;

        case IDM_BLUR:
            /* blur the region around the position that point by the mouse */
            BlurringOperation(hWnd,message,wParam,lParam);
            return 0;

        case IDM_SCALE:
            /* scaling by mouse */
            ScalingOperation(hWnd,message,wParam,lParam);
            return 0;

        case IDM_SMOOTH:
            /* smooth the select region */
            SmoothingOperation(hWnd,message,wParam,lParam);
            return 0;
        default:    /* Lets Windows process it        */
            return (DefWindowProc(hWnd, message, wParam, lParam));
        }

    case WM_CREATE:/* initial some variable when create window */
        {
        HDC hDC;
        caption_height = GetSystemMetrics(SM_CYCAPTION);
        if ((hLibrary[0] = LoadLibrary("HAIR.DLL")) < 32)
            DestroyWindow(hWnd);
        if ((hLibrary[1] = LoadLibrary("CHIN.DLL")) < 32)
            DestroyWindow(hWnd);
        if ((hLibrary[2] = hLibrary[3] = LoadLibrary("EAR.DLL")) < 32)
            DestroyWindow(hWnd);
        if ((hLibrary[4] = hLibrary[5] = LoadLibrary("EYEBROW.DLL")) < 32)
            DestroyWindow(hWnd);
        if ((hLibrary[6] = hLibrary[7] = LoadLibrary("EYE.DLL")) < 32)
            DestroyWindow(hWnd);
        if ((hLibrary[8] = LoadLibrary("NOSE.DLL")) < 32)
            DestroyWindow(hWnd);
        if ((hLibrary[9] = LoadLibrary("MOUTH.DLL")) < 32)
            DestroyWindow(hWnd);
        PopFileInitialize(hWnd);
      nXDisplaySize = GetSystemMetrics(SM_CXSCREEN);
        nYDisplaySize = GetSystemMetrics(SM_CYSCREEN);
        hDC = GetDC(hPicWnd);
        hMainBitmap = CreateCompatibleBitmap(hDC, PictureSize.x, PictureSize.y);
        ReleaseDC(hPicWnd, hDC);
        }
        break;

    case WM_SIZE:   /* set some variable when the window size has changed */
        GetClientRect(hWnd,&rc);
        if(IsWindow(hPicWnd))
         MoveWindow(hPicWnd,0,0,LOWORD(lParam)-300,HIWORD(lParam),TRUE);
        rc.left = rc.right-276;
        y_start = (rc.bottom - 432 - (2*caption_height))/2;
        if(y_start < 0) y_start = 0;
        y1_start = y_start + caption_height + 231;
        if(IsWindow(hToolDlg1)) MoveWindow(hToolDlg1,rc.left,0+y_start,
                                    252,231+caption_height,TRUE);
        if(IsWindow(hToolDlg2)) MoveWindow(hToolDlg2,rc.left,y1_start,
                                    252,201+caption_height,TRUE);
        if(IsWindow(hPicSelectFrame))
            MoveWindow(hPicSelectFrame,rc.left,0+y_start,
                    252,231+caption_height,TRUE);
        for( i = 0; i <= 9; i++)
        {
            if( (i == 3) ||  // Right Ear
             (i == 5) ||  // Right Eyebrow
             (i == 7)    // Right Eye
            ) i++;
            if(IsWindow(hSelectComponent[i]))
            MoveWindow(hSelectComponent[i],
                    rc.left,y1_start,252,201+caption_height,TRUE);
        }
        return 0;

    case WM_DESTROY:        /* message: window being destroyed */
        DeleteObject(hMainBitmap);
            for(i=1;i<10;i++)
                if (hLibrary[i] >= 32)
                    FreeLibrary(hLibrary[i]);
        PostQuitMessage(0);
        break;

    default:            /* Passes it on if unproccessed            */
        return (DefWindowProc(hWnd, message, wParam, lParam));
    }
    return DefWindowProc(hWnd, message, wParam, lParam);
}
```

```
/*****************************************************************
FUNCTION: App_UnRegisterClasses(void)
PURPOSE: Release Application class
*****************************************************************/
void App_UnRegisterClasses(void)
{
    WNDCLASS  wc;  /* struct to define a window class */
    memset(&wc, 0x00, sizeof(WNDCLASS));
    UnregisterClass(szAppName, hInst);
    UnregisterClass(szPicClass, hInst);
    UnregisterClass(szPicSelectFrameClass, hInst);
    UnregisterClass(szPicSelectClass, hInst);
    UnregisterClass(szToolDlgClass, hInst);
    UnregisterClass(szToolOpDlgClass, hInst);
    UnregisterClass(szPkRadioButtonClass, hInst);
    UnregisterClass(szPkOwnerDrawRadioButtonClass, hInst);
    UnregisterClass(szSelectComponent, hInst);
}


/*****************************************************************
    FUNCTION: AboutDlgProc(HWND, UINT, UINT, LONG)
    PURPOSE: Processes messages for "About" dialog box
    MESSAGES:
        WM_INITDIALOG - initialize dialog box
        WM_COMMAND  - Input received
    COMMENTS: No initialization is needed for this particular dialog box, but TRUE
must be returned to Windows. Wait for user to click on "Ok" button, then close the
dialog box.
*****************************************************************/
BOOL FAR PASCAL _export AboutDlgProc(HWND hDlg, UINT message,
                                     UINT wParam, LONG lParam)
{
    switch (message)
    {
    case WM_INITDIALOG:       /* message: initialize dialog box */
        return (TRUE);

    case WM_COMMAND:          /* message: received a command */
        if (wParam == IDOK || wParam == IDCANCEL)
        {
            EndDialog(hDlg, TRUE);     /* Exits the dialog box */
            return (TRUE);
        }
        break;
    }
    return (FALSE);           /* Didn't process a message */
}
/*****************************************************************
    FUNCTION: InitialData(void)
    PURPOSE: Initialize the data required while program execute.
*****************************************************************/
void InitialData(void)
{
  int i;
  for (i=0;i<10;i++)
    {
     nMaxStyle[i] = GetMaxStyle(i);
     ptPosition[i] = GetComponentPos(i,1);
     nCurrentStyle[i] = 1;
    }
}

/*****************************************************************
    FUNCTION: DoCaption (HWND, char)
    PURPOSE: Change the string in caption bar when change working file.
*****************************************************************/
void DoCaption (HWND hwnd, char *szTitleName)
{
    char szCaption[64 + _MAX_FNAME + _MAX_EXT];
    wsprintf (szCaption, "%s - %s", (LPSTR) szAppName,
        (LPSTR) (szTitleName[0] ? szTitleName : UNTITLED));
    SetWindowText (hwnd, szCaption);
}

/*****************************************************************
    FUNCTION: OkMessage(HWND, char, char)
    PURPOSE: Create message box to wait for answer from user.
*****************************************************************/
void OkMessage (HWND hwnd, char *szMessage, char *szTitleName)
{
    char szBuffer[64 + _MAX_FNAME + _MAX_EXT];
    wsprintf (szBuffer, szMessage,
        (LPSTR) (szTitleName[0] ? szTitleName : UNTITLED));
    MessageBox(hwnd, szBuffer, szAppName, MB_OK|MB_ICONEXCLAMATION);
}

/*****************************************************************
    FUNCTION: AskAboutSave (HWND, char)
    PURPOSE: Ask user to save current data file if it not be saved when it being replaced
by the new one.
*****************************************************************/
short AskAboutSave (HWND hwnd, char *szTitleName)
{
    char    szBuffer[64 + _MAX_FNAME + _MAX_EXT];
    short   nReturn;
   wsprintf (szBuffer, "Save current changes in %s?",
        (LPSTR) (szTitleName[0] ? szTitleName : UNTITLED));
    nReturn = MessageBox(hwnd, szBuffer, szAppName,
                MB_YESNOCANCEL|MB_ICONQUESTION);
   if (nReturn == IDYES)
        if (!SendMessage (hwnd, WM_COMMAND, IDM_SAVE, 0L))
            nReturn = IDCANCEL;
    return nReturn;
}
```

```
/*****************************************************************

    PROGRAM: PicProc.c (Angelo subprogram)
    PURPOSE: this file contain the routine that handle picture window.

*****************************************************************/

#include <windows.h>
#include <stdio.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************

FUNCTION: PictureProc(HWND, UINT, UINT, LONG)
PURPOSE:  Processes messages for Drawing Picture.
MESSAGES:
    WM_CREATE   - initialize child dialog box
COMMENTS:
*****************************************************************/
LONG FAR PASCAL _export PictureProc(HWND hWnd,
                UINT message, UINT wParam, LONG lParam)
{
    RECT        rc;
    static int  nVscrollMax = 100;
    static int  nHscrollMax = 100;
    int         nVscrollInc, nHscrollInc;
    switch (message)
    {
    case WM_PAINT:
        switch(nOperationMode)
        {
        /* window will paint according to current mode */
            case PIC_SELECT:
                PictureSelect(hWnd, message, wParam, lParam);
                break;
            case PIC_CANCEL:
                PictureCancel(hWnd, message, wParam, lParam);
                break;
            case PIC_MOVE:
                PictureMove(hWnd);
                break;
            case PIC_SIZE:
                PictureSizeOp(hWnd);
                break;
            case PIC_DELETE:
                PictureDelete(hWnd, message, wParam, lParam);
                break;
            case PIC_REDRAW:
                PictureRedraw(hWnd,message,wParam,lParam);
                break;
            case PIC_XOR:
                PictureXnor(hWnd,message,wParam,lParam);
                break;
            case PIC_GAP:
                PictureGap(hWnd);
                break;

            default:
                PictureRedraw(hWnd,message,wParam,lParam);
                break;
        }
        return 0;

    case WM_LBUTTONDOWN:
        if (bGetArea)
            GetAreaOperation(hPicWnd, message, wParam, lParam);
        return 0;

    case WM_MOUSEMOVE:
        SetCursor(hMyCursor);
        if (bGetArea)
            GetAreaOperation(hPicWnd, message, wParam, lParam);
        return 0;

    case WM_LBUTTONUP:
        if (bGetArea)
            GetAreaOperation(hPicWnd, message, wParam, lParam);
        return 0;
    case WM_VSCROLL:
    {
        switch(wParam)
        {
            case SB_LINEUP:
                nVscrollInc = -10;
                break;
            case SB_LINEDOWN:
                nVscrollInc = 10;
                break;
            case SB_PAGEUP:
                nVscrollInc = -30;
                break;
            case SB_PAGEDOWN:
                nVscrollInc = 30;
                break;
            case SB_THUMBPOSITION:
```

```
                nVscrollInc= LOWORD (lParam) - nVscrollPos;
                break;
            default:
                nVscrollInc = 0;
        }
        nVscrollInc = max(-nVscrollPos, min(nVscrollInc, nVscrollMax - nVscrollPos));
        if (nVscrollInc != 0)
            {
            nVscrollPos += nVscrollInc;
            ScrollWindow(hWnd, 0, -nVscrollInc, NULL, NULL);
            SetScrollPos (hWnd, SB_VERT, nVscrollPos, TRUE);
            UpdateWindow(hWnd);
            }
        }
    return 0;
  case WM_HSCROLL:
    {
        switch(wParam)      {
            case SB_LINEUP:
                nHscrollInc = -10;
                break;
            case SB_LINEDOWN:
                nHscrollInc = 10;
                break;
            case SB_PAGEUP:
                nHscrollInc = -30;
                break;
            case SB_PAGEDOWN:
                nHscrollInc = 30;
                break;
            case SB_THUMBPOSITION:
                nHscrollInc = LOWORD (lParam) - nHscrollPos;
                break;
            default:
                nHscrollInc = 0;
        }
        nHscrollInc = max(-nHscrollPos, min(nHscrollInc, nHscrollMax - nHscrollPos));
        if (nHscrollInc != 0)
            {
            nHscrollPos += nHscrollInc;
            ScrollWindow(hWnd, -nHscrollInc,0, NULL, NULL);
            SetScrollPos (hWnd, SB_HORZ, nHscrollPos, TRUE);
            UpdateWindow(hWnd);
            }
        }
    return 0;
    case WM_SIZE:
        StartPoint.x = (((int) LOWORD(lParam)) - 340) / 2;
        StartPoint.y = (((int) HIWORD(lParam)) - 380) / 2;
        SetScrollRange (hWnd, SB_VERT, 0, nVscrollMax, FALSE);
        SetScrollPos  (hWnd, SB_VERT, nVscrollPos, TRUE);
        SetScrollRange (hWnd, SB_HORZ, 0, nHscrollMax, FALSE);
        SetScrollPos  (hWnd, SB_HORZ, nHscrollPos, TRUE);
      break;
    }
  return DefWindowProc(hWnd, message, wParam, lParam);
}

/*****************************************************************************
FUNCTION: PictureSelectProc(HWND, UINT, UINT, LONG)
PURPOSE: Processes messages for Drawing Picture.
MESSAGES:
    WM_CREATE   - initialize child dialog box
COMMENTS:
*****************************************************************************/

LONG FAR PASCAL _export PictureSelectProc(HWND hWnd, UINT message,
                                          UINT wParam, LONG lParam)
{
    RECT        rc;

    switch (message)
    {

    case WM_CREATE:
        break;

    case WM_PAINT:
        DrawSelectPicture(hWnd);
        break;

    case WM_SIZE:
        GetClientRect(hWnd,&rc);
        break;
    }
  return DefWindowProc(hWnd, message, wParam, lParam);
}


void DrawButton (HDC hDC, int x, int y, int x_size, int y_size, int status)
{
    int         count;
    RECT        rc;
    HBRUSH      hBrush_GRAY,hBrush_LTGRAY;

    rc.left  = x;
    rc.right  = x+x_size;
    rc.top  = y;
    rc.bottom    = y+y_size;
```

```
    hBrush_GRAY = CreateSolidBrush(RGB(128, 128, 128));
    hBrush_LTGRAY = CreateSolidBrush(RGB(192, 192, 192));

    if(status == BUTTON_RELEASE)
    {
        FillRect(hDC,&rc,hBrush_LTGRAY);
        for(count = 1; count <= 2; count++)
        {
            rc.left  += 1;
            rc.right -= 1;
            rc.top   += 1;
            rc.bottom-= 1;
            FrameRect(hDC,&rc,hBrush_GRAY);
        }
        rc.left  = x;
        rc.right = x+x_size;
        rc.top   = y;
        rc.bottom = y+y_size;
        SelectObject(hDC,GetStockObject(WHITE_PEN));
        MoveTo(hDC,rc.left+1,rc.top+1);
        LineTo(hDC,rc.right-2,rc.top+1);
        MoveTo(hDC,rc.left+2,rc.top+2);
        LineTo(hDC,rc.right-3,rc.top+2);

        MoveTo(hDC,rc.left+1,rc.top+1);
        LineTo(hDC,rc.left+1,rc.bottom-2);
        MoveTo(hDC,rc.left+2,rc.top+2);
        LineTo(hDC,rc.left+2,rc.bottom-3);
    }
    if(status == BUTTON_PRESS)
    {
        for(count = 1; count <= 2; count++)
        {
            rc.left  += 1;
            rc.right -= 1;
            rc.top   += 1;
            rc.bottom-= 1;
            FrameRect(hDC,&rc,hBrush_LTGRAY);
        }
    }
    FrameRect(hDC,&rc,GetStockObject(BLACK_BRUSH));
    DeleteObject(hBrush_GRAY);
    DeleteObject(hBrush_LTGRAY);
}

void DrawSelectPicture(HWND hWnd)
{
    PAINTSTRUCT    ps;
    HBITMAP        hBmp;
    HANDLE         hOldBmp;
    HDC            hDC;
    HDC            hMemDC;
    RECT           Rect;
    BITMAP         Bitmap;     /* bitmap structure */
    int            nResource;
    int            i,n;

    BeginPaint(hWnd, &ps);
    hDC = ps.hdc;
    n = COMPONENT;
    if((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE))
        {
            nCurrentStyle[n+1] = nCurrentStyle[n];
            n++;
        }
    /* if component is left ear, left eyebrow or left eye, set to draw both left and right
component */

    for(i=COMPONENT;i<=n;i++)          {
    if(nCurrentStyle[i]>-1)
    {
        nResource = nCurrentStyle[i];
     if ((i==REAR)||(i==REBROW)||(i==REYE))
        nResource += 5000;
        hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC = CreateCompatibleDC(hDC);
        hOldBmp = SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap);
        GetClientRect(hWnd, &Rect);
        switch(COMPONENT)
        {
        case NOSE:
        case MOUTH:
            BitBlt(hDC, (Rect.right-Bitmap.bmWidth)/2,
                (Rect.bottom-Bitmap.bmHeight)/2,
                Bitmap.bmWidth, Bitmap.bmHeight,
                hMemDC, 0, 0, SRCAND);
            break;

        case HAIR:
            StretchBlt(hDC, Rect.left+20, Rect.top+35,
                Rect.right-40, Rect.bottom-70,
                hMemDC, 0, 0,
                Bitmap.bmWidth, Bitmap.bmHeight,
                SRCAND);
            break;

        case CHIN:
```

```
                StretchBlt(hDC, Rect.left+20, Rect.top+20,
                    Rect.right-40, Rect.bottom-40,
                    hMemDC, 0, 0,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    SRCAND);
                break;

            case LEYE:
                if(i!=n)
                BitBlt(hDC, Rect.left+20,
                    (Rect.bottom-Bitmap.bmHeight)/2,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCAND);
                else
                BitBlt(hDC, Rect.right-Bitmap.bmWidth-20,
                    (Rect.bottom-Bitmap.bmHeight)/2,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCAND);
                break;

            case LEBROW:
            case LEAR:
                if(i!=n)
                BitBlt(hDC, Rect.left+5,
                    (Rect.bottom-Bitmap.bmHeight)/2,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCAND);
                else
                BitBlt(hDC, Rect.right-Bitmap.bmWidth-5,
                    (Rect.bottom-Bitmap.bmHeight)/2,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCAND);
                break;

        }
        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }

    EndPaint(hWnd, &ps);
}

/*****************************************************************
FUNCTION: PictureSelectFrameProc(HWND, UINT, UINT, LONG)
PURPOSE: Processes messages for Drawing Picture.
MESSAGES:
    WM_CREATE - initialize child dialog box
COMMENTS:
*****************************************************************/
LONG FAR PASCAL _export PictureSelectFrameProc(HWND hWnd, UINT message,
                                    UINT wParam, LONG lParam)
{
    PAINTSTRUCT     ps;
    HDC             hDC;
    static RECT     rc;

    switch (message)
    {
    case WM_CREATE:
        GetClientRect(hWnd,&rc);
        hPicSelectWnd = CreateWindow("PictureSelectClass","",
                    WS_CHILD|WS_CLIPSIBLINGS|WS_VISIBLE,
                    rc.left+7, rc.top+7, rc.right-12, rc.bottom-12,
                    hWnd,NULL,hInst,NULL);
        break;

    case WM_PAINT:
        hDC = BeginPaint(hWnd, &ps);
        DrawFrame(hDC,rc.left+4, rc.top+4,
                rc.right-8, rc.bottom-8,DRAWBOTH);
        EndPaint(hWnd, &ps);
        break;

    case WM_SIZE:
        GetClientRect(hWnd,&rc);
        break;
    }
    return DefWindowProc(hWnd, message, wParam, lParam);
}
```

```
/*****************************************************************
   PROGRAM: Picture.c (Angelo subprogram)
   PURPOSE: this file contain the routine that process WM_PAINT message of picture
window in each mode.
*****************************************************************/
#include <windows.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************
   FUNCTION: PictureRedraw (HWND, UINT, UINT, LONG)
   PURPOSE: Repaint the picture window with all current style, position and size.
*****************************************************************/
LONG PictureRedraw(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
   PAINTSTRUCT     ps;
   HBITMAP         hBmp;
   HANDLE          hOldBmp;
   HDC             hDC;
   HDC             hMemDC;
   char            szTemp[10];
   BITMAP          Bitmap;     /* bitmap structure */
   int             nResource;
   int             i;
   BeginPaint(hWnd, &ps);
   hDC = ps.hdc;
   for (i=0;i<10;i++) {
   if (nCurrentStyle[i]>-1) {
         nResource = nCurrentStyle[i];
         if (i==REAR)||(i==REBROW)||(i==REYE))
            nResource += 5000;
         hBmp = LoadBitmap(hLibrary,(i, MAKEINTRESOURCE(nResource));
         hMemDC = CreateCompatibleDC(hDC);
         hOldBmp = SelectObject(hMemDC, hBmp);
         GetObject(hBmp, 16, (LPSTR) &Bitmap);
         if (ptSize[i].x == 0) {
            ptSize[i].x = Bitmap.bmWidth;
            ptSize[i].y = Bitmap.bmHeight;
            bStretchComponent[i] = TRUE;
            }
         if (!bStretchComponent[i])
            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
               ptPosition[i].y+StartPoint.y-nVscrollPos,
               Bitmap.bmWidth, Bitmap.bmHeight,
               hMemDC, 0, 0, SRCAND);
         else
            StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
               ptPosition[i].y+StartPoint.y-nVscrollPos,
               ptSize[i].x,ptSize[i].y,
               hMemDC, 0, 0,
               Bitmap.bmWidth, Bitmap.bmHeight,
               SRCAND);

         SelectObject(hMemDC, hOldBmp);
         DeleteDC(hMemDC);
         DeleteObject(hBmp);
   }
   }
   EndPaint(hWnd, &ps);
   return (0L);
}

/*****************************************************************
   FUNCTION: PictureDelete (HWND, UINT, UINT, LONG)
   PURPOSE: Repaint the picture window with all current style, position and size, except
one component that being changed.
*****************************************************************/
LONG PictureDelete(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
   PAINTSTRUCT     ps;
   HBITMAP         hBmp;
   HANDLE          hOldBmp;
   HDC             hDC;
   HDC             hMemDC;
   char            szTemp[10];
   BITMAP          Bitmap;     /* bitmap structure */
   int             nResource;
   int             i,n;

   BeginPaint(hWnd, &ps);
   hDC = ps.hdc;
   n=COMPONENT;
   if ((COMPONENT == LEAR)||(COMPONENT == LEBROW)||(COMPONENT == LEYE)) {
         nCurrentStyle[n+1] = nCurrentStyle[n],
         n++;
   }
   for (i=0,i<10,i++) {
   if ((nCurrentStyle[i]>-1)&&(i!=COMPONENT)&&(i!=n)) {
         nResource = nCurrentStyle[i];
         if ((i==REAR)||(i==REBROW)||(i==REYE))
            nResource += 5000;
         hBmp = LoadBitmap(hLibrary,(i, MAKEINTRESOURCE(nResource));
         hMemDC = CreateCompatibleDC(hDC);
         hOldBmp = SelectObject(hMemDC, hBmp);
         GetObject(hBmp, 16, (LPSTR) &Bitmap);
         if (!bStretchComponent[i])
            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
```

```
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                Bitmap.bmWidth, Bitmap.bmHeight,
                hMemDC, 0, 0, SRCAND);
        else
            StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                ptSize[i].x, ptSize[i].y,
                hMemDC, 0, 0,
                Bitmap.bmWidth, Bitmap.bmHeight,
                SRCAND);

        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }
    for(i=COMPONENT;i<=n;i++){
    if(nCurrentStyle[i]>1){
        nResource=nCurrentStyle[i];
        if((i==REAR)||(i==REBROW)||(i==REYE))
            nResource+=5000;
        hBmp=LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC=CreateCompatibleDC(hDC);
        hOldBmp=SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap);
        //get new component's position
        ptPosition[i]=GetComponentPos(i,nCurrentStyle[i]);
        BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
            ptPosition[i].y+StartPoint.y-nVscrollPos,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hMemDC, 0, 0, SRCXNOR);
        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }
    EndPaint(hWnd, &ps);
    return(0L);
}

/*****************************************************************************
    FUNCTION: PictureSelect(HWND, UINT, UINT, LONG)
    PURPOSE: Paint the picture window with only one component.
*****************************************************************************/
LONG PictureSelect(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
    PAINTSTRUCT     ps;
    HBITMAP         hBmp;
    HANDLE          hOldBmp;
    HDC             hDC;
    HDC             hMemDC;
    char            szTemp[10];
    BITMAP          Bitmap;     /* bitmap structure */
    int             nResource;
    int             i,n;
    BeginPaint(hWnd, &ps);
    hDC = ps.hdc;
    n=COMPONENT;
    if((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE)){
        nCurrentStyle[n+1]=nCurrentStyle[n];
        n++;
    }
    /* if component is left ear, left eyebrow or left eye, set to draw both left and right
     component */
    for(i=COMPONENT;i<=n;i++)  {
    if(nCurrentStyle[i]>1){
        nResource=nCurrentStyle[i];
        if((i==REAR)||(i==REBROW)||(i==REYE))
            nResource+=5000;
        hBmp=LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC=CreateCompatibleDC(hDC);
        hOldBmp=SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap),

        BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
            ptPosition[i].y+StartPoint.y-nVscrollPos,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hMemDC, 0, 0, SRCAND);

        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }
    nOperationMode=PIC_REDRAW;
    EndPaint(hWnd, &ps);
    return(0L);
}

/*****************************************************************************
    FUNCTION: PictureXnor(HWND, UINT, UINT, LONG)
    PURPOSE: Repaint the picture window with only one component. By paint the old style
in XNOR manner to disappear it, and paint the new style on the screen in XNOR manner
too.
*****************************************************************************/
LONG PictureXnor(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
    PAINTSTRUCT ps;
```

```
    HBITMAP         hBmp;
    HANDLE          hOldBmp;
    HDC             hDC;
    HDC             hMemDC;
    char            szTemp[10];
    BITMAP          Bitmap;     /* bitmap structure */
    int             nResource;
    int             i,n;

    BeginPaint(hWnd, &ps);
    hDC = ps.hdc;
    n = COMPONENT;
    if((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE)) {
            nCurrentStyle[n+1] = nCurrentStyle[n];
            n++;
    }
    /* if component is left ear, left eyebrow or left eye, set to draw both left and right
      component */

    for(i=COMPONENT;i<=n;i++)  {
    if (nOldStyle>-1) {
            nResource = nOldStyle;
            if((i==REAR)||(i==REBROW)||(i==REYE))
                nResource += 5000;
            hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
            hMemDC = CreateCompatibleDC(hDC);
            hOldBmp = SelectObject(hMemDC, hBmp);
            GetObject(hBmp, 16, (LPSTR) &Bitmap);

            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                    ptPosition[i].y+StartPoint.y-nVscrollPos,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCXNOR);

            SelectObject(hMemDC, hOldBmp);
            DeleteDC(hMemDC);
            DeleteObject(hBmp);
    }
    }
    for(i=COMPONENT;i<=n;i++)  {
     if (nCurrentStyle[i]>-1) {
            nResource = nCurrentStyle[i];
            if((i==REAR)||(i==REBROW)||(i==REYE))
                nResource += 5000;
            hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
            hMemDC = CreateCompatibleDC(hDC);
            hOldBmp = SelectObject(hMemDC, hBmp);
            GetObject(hBmp, 16, (LPSTR) &Bitmap);
            //get new component's position
            ptPosition[i] = GetComponentPos(i,nCurrentStyle[i]);

            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                Bitmap.bmWidth, Bitmap.bmHeight,
                hMemDC, 0, 0, SRCXNOR);

            SelectObject(hMemDC, hOldBmp);
            DeleteDC(hMemDC);
            DeleteObject(hBmp);
    }
    }
    EndPaint(hWnd, &ps);
    return (0L);
}

/*****************************************************************************

FUNCTION :PictureCancel(HWND, UINT, UINT, LONG)
PURPOSE: Repaint the picture window with only one component.
    By delete the style on screen and restore the old
    style.

*****************************************************************************/
LONG PictureCancel(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
    PAINTSTRUCT     ps;
    HBITMAP         hBmp;
    HANDLE          hOldBmp;
    HDC             hDC;
    HDC             hMemDC;
    POINT           ptPos;
    BITMAP          Bitmap;     /* bitmap structure */
    int             nResource;
    int             i,n;

    BeginPaint(hWnd, &ps);
    hDC = ps.hdc;
    n = COMPONENT;
    if((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE)) {
            nCurrentStyle[n+1] = nCurrentStyle[n];
            n++;
    }
    /* if component is left ear, left eyebrow or left eye, set to draw both left and right
      component */

    for(i=COMPONENT;i<=n;i++)  {
    if (nOldStyle>-1) {
            nResource = nOldStyle;
```

```
        if ((i==REAR)||(i==REBROW)||(i==REYE))
            nResource += 5000;
        hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC = CreateCompatibleDC(hDC);
        hOldBmp = SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap);
        ptPos = GetComponentPos(i, nOldStyle);

        BitBlt(hDC, ptPos.x+StartPoint.x-nHscrollPos,
            ptPos.y+StartPoint.y-nVscrollPos,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hMemDC, 0, 0, SRCXNOR);

        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }
    for (i=COMPONENT;i<=n;i++)  {
    if (nCurrentStyle[i]>-1) {
        nResource = nCurrentStyle[i];
        if ((i==REAR)||(i==REBROW)||(i==REYE))
            nResource += 5000;
        hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC = CreateCompatibleDC(hDC);
        hOldBmp = SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap);

        if (!bStretchComponent[i])
            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                Bitmap.bmWidth, Bitmap.bmHeight,
                hMemDC, 0, 0, SRCAND);
        else
            StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                ptSize[i].x, ptSize[i].y,
                hMemDC, 0, 0,
                Bitmap.bmWidth, Bitmap.bmHeight,
                SRCAND);

        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }
    EndPaint(hWnd, &ps);
    return (0L);
}
```

```
/*****************************************************************
   PROGRAM: Picture2.c  (Angelo subprogram)
   PURPOSE: this file contain the routine that process WM_PAINT message of picture
,window in each mode.
*****************************************************************/

#include <windows.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************
   FUNCTION: PictureSizeOp(HWND)
   PURPOSE: Repaint the picture window and handle picture window while size
operation.
*****************************************************************/
LONG PictureSizeOp(HWND hWnd)
{
   PAINTSTRUCT    ps;
   HDC            hDC;
   HBITMAP        hBmp;
   HANDLE         hOldBmp;
   HDC            hMemDC;
   char           szTemp[10];
   BITMAP         Bitmap;     /* bitmap structure */
   int            nResource;
   int            i,n;

   BeginPaint(hWnd, &ps);
   hDC = ps.hdc;
   n = COMPONENT;
   if ((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE)) {
         nCurrentStyle[n+1] = nCurrentStyle[n];
         n++;
   }
   /* if component is left ear, left eyebrow or left eye, set to draw both left and right
    component */

   for(i=COMPONENT;i<=n;i++)  {
   if (nCurrentStyle[i]>-1) {
         nResource = nCurrentStyle[i];
         if ((i==REAR)||(i==REBROW)||(i==REYE))
               nResource += 5000;
         hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
         hMemDC = CreateCompatibleDC(hDC);
         hOldBmp = SelectObject(hMemDC, hBmp);
         GetObject(hBmp, 16, (LPSTR) &Bitmap);
         if (!bStretchComponent[i])
               BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                     ptPosition[i].y+StartPoint.y-nVscrollPos,
                     Bitmap.bmWidth, Bitmap.bmHeight,
                     hMemDC, 0, 0, SRCXNOR);
         else
               StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                     ptPosition[i].y+StartPoint.y-nVscrollPos,
                     ptSize[i].x, ptSize[i].y,
                     hMemDC, 0, 0,
                     Bitmap.bmWidth, Bitmap.bmHeight,
                     SRCXNOR);

         SelectObject(hMemDC, hOldBmp);
         DeleteDC(hMemDC);
         DeleteObject(hBmp);
   }
   }
   switch(nDirection) {
   case LEFT:
      switch(nPointResize) {
         case 0:
         case 5:
            ptPosition[COMPONENT].x-=(2*nMoveSizeStep);
            if(n==COMPONENT)
                  ptSize[COMPONENT].x+=(2*nMoveSizeStep);
            else {
                  ptPosition[COMPONENT+1].x-=nMoveSizeStep;
                  ptSize[COMPONENT].x+=nMoveSizeStep;
                  ptSize[COMPONENT+1].x+=nMoveSizeStep;
            }
            break;
         case 2:
         case 7:
            if(n==COMPONENT)
                  ptSize[COMPONENT].x-=(2*nMoveSizeStep);
            else {
                  ptPosition[COMPONENT+1].x-=nMoveSizeStep;
                  ptSize[COMPONENT].x-=nMoveSizeStep;
                  ptSize[COMPONENT+1].x-=nMoveSizeStep;
            }
            break;
      }
      break;
   case RIGHT:
      switch(nPointResize) {
         case 0:
         case 5:
            ptPosition[COMPONENT].x+=(2*nMoveSizeStep);
            if(n==COMPONENT)
```

```
                ptSize[COMPONENT].x-=(2*nMoveSizeStep);
            else{
                ptPosition[COMPONENT+1].x+=nMoveSizeStep;
                ptSize[COMPONENT].x-=nMoveSizeStep;
                ptSize[COMPONENT+1].x-=nMoveSizeStep;
            }
            break;
        case 2:
        case 7:
            if(n==COMPONENT)
                ptSize[COMPONENT].x+=(2*nMoveSizeStep);
            else{
                ptPosition[COMPONENT+1].x+=nMoveSizeStep;
                ptSize[COMPONENT].x+=nMoveSizeStep;
                ptSize[COMPONENT+1].x+=nMoveSizeStep;
            }
            break;
    }
    break;
case UP:
    switch(nPointResize){
        case 0:
        case 2:
            ptPosition[COMPONENT].y-=(2*nMoveSizeStep);
            ptSize[COMPONENT].y+=(2*nMoveSizeStep);
            if(n!=COMPONENT){
                ptPosition[COMPONENT+1].y-=(2*nMoveSizeStep);
                ptSize[COMPONENT+1].y+=(2*nMoveSizeStep);
            }
            break;
        case 5:
        case 7:
            ptSize[COMPONENT].y-=(2*nMoveSizeStep);
            if(n!=COMPONENT)
                ptSize[COMPONENT+1].y-=(2*nMoveSizeStep);
            break;
    }
    break;
case DOWN:
    switch(nPointResize){
        case 0:
        case 2:
            ptPosition[COMPONENT].y+=(2*nMoveSizeStep);
            ptSize[COMPONENT].y-=(2*nMoveSizeStep);
            if(n!=COMPONENT){
                ptPosition[COMPONENT+1].y+=(2*nMoveSizeStep);
                ptSize[COMPONENT+1].y-=(2*nMoveSizeStep);
            }
            break;
        case 5:
        case 7:
            ptSize[COMPONENT].y+=(2*nMoveSizeStep);
            if(n!=COMPONENT)
                ptSize[COMPONENT+1].y+=(2*nMoveSizeStep);
            break;
    }
    break;
}
for(j=COMPONENT;j<=n;j++) {
if(nCurrentStyle[j]>-1){
    nResource=nCurrentStyle[j];
    if((j==REAR)||(j==REBROW)||(j==REYE))
        nResource+=5000;
    hBmp=LoadBitmap(hLibrary,j,MAKEINTRESOURCE(nResource));
    hMemDC=CreateCompatibleDC(hDC);
    hOldBmp=SelectObject(hMemDC,hBmp);
    GetObject(hBmp, 16, (LPSTR)&Bitmap);

    if(!bStretchComponent[j])
        BitBlt(hDC,ptPosition[j].x+StartPoint.x-nHscrollPos,
            ptPosition[j].y+StartPoint.y-nVscrollPos,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hMemDC,0,0,SRCXNOR);
    else
        StretchBlt(hDC,ptPosition[j].x+StartPoint.x-nHscrollPos,
            ptPosition[j].y+StartPoint.y-nVscrollPos,
            ptSize[j].x,ptSize[j].y,
            hMemDC,0,0,
            Bitmap.bmWidth, Bitmap.bmHeight,
            SRCXNOR);

    SelectObject(hMemDC,hOldBmp);
    DeleteDC(hMemDC);
    DeleteObject(hBmp);
}
}
EndPaint(hWnd,&ps);
return (0L);
}

/*****************************************************************************

FUNCTION: PictureGap(HWND)
PURPOSE: Repaint the picture window and handle picture window while
    being changed the gap between eye, ear or eyebrow.

*****************************************************************************/
```

```
LONG PictureGap(HWND hWnd)
{
    PAINTSTRUCT     ps;
    HDC             hDC;
    HBITMAP         hBmp;
    HANDLE          hOldBmp;
    HDC             hMemDC;
    char            szTemp[10];
    BITMAP          Bitmap;     /* bitmap structure */
    int             nResource,n,i;

    BeginPaint(hWnd, &ps);
    hDC = ps.hdc;
    n=COMPONENT;
    if((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE)) {
        nCurrentStyle[n+1] = nCurrentStyle[n];
        n++;
    }
    /* if component is left ear, left eyebrow or left eye, set to draw both left and right
     component */

    for(i=COMPONENT;i<=n;i++)  {
    if (nCurrentStyle[i]>-1) {
        nResource = nCurrentStyle[i];
        if((i==REAR)||(i==REBROW)||(i==REYE))
            nResource += 5000;
        hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC = CreateCompatibleDC(hDC);
        hOldBmp = SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap);

        if(!bStretchComponent[i])
            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                Bitmap.bmWidth, Bitmap.bmHeight,
                hMemDC, 0, 0, SRCXNOR);
        else
            StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                ptSize[i].x,ptSize[i].y,
                hMemDC, 0, 0,
                Bitmap.bmWidth, Bitmap.bmHeight,
                SRCXNOR);

        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }

switch(nDirection) {
        case LEFT:
            ptPosition[COMPONENT].x-=nMoveSizeStep;
            ptPosition[COMPONENT+1].x+=nMoveSizeStep;
            break;
        case RIGHT:
            ptPosition[COMPONENT].x+=nMoveSizeStep;
            ptPosition[COMPONENT+1].x-=nMoveSizeStep;
            break;

}
 for(i=COMPONENT;i<=n;i++)  {
 if (nCurrentStyle[i]>-1) {
        nResource = nCurrentStyle[i];
        if((i==REAR)||(i==REBROW)||(i==REYE))
            nResource += 5000;
        hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
        hMemDC = CreateCompatibleDC(hDC);
        hOldBmp = SelectObject(hMemDC, hBmp);
        GetObject(hBmp, 16, (LPSTR) &Bitmap);

        if(!bStretchComponent[i])
            BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                Bitmap.bmWidth, Bitmap.bmHeight,
                hMemDC, 0, 0, SRCXNOR);
        else
            StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                ptPosition[i].y+StartPoint.y-nVscrollPos,
                ptSize[i].x,ptSize[i].y,
                hMemDC, 0, 0,
                Bitmap.bmWidth, Bitmap.bmHeight,
                SRCXNOR);

        SelectObject(hMemDC, hOldBmp);
        DeleteDC(hMemDC);
        DeleteObject(hBmp);
    }
    }
 EndPaint( hWnd, &ps);
  return (0L);
}

/*****************************************************************************
   FUNCTION: PictureMove(HWND)
   PURPOSE: Repaint the picture window and handle picture window while move
operation.
*****************************************************************************/

LONG PictureMove(HWND hWnd)
```

```
{
    PAINTSTRUCT     ps;
    HDC             hDC;
    HBITMAP         hBmp;
    HANDLE          hOldBmp;
    HDC             hMemDC;
    char            szTemp[10];
    BITMAP          Bitmap;     /* bitmap structure */
    int             nResource,n,i;

    BeginPaint(hWnd, &ps);
    hDC = ps.hdc;
    n = COMPONENT;
    if ((COMPONENT==LEAR)||(COMPONENT==LEBROW)||(COMPONENT==LEYE)) {
            nCurrentStyle[n+1] = nCurrentStyle[n];
            n++;
    }
    /* if component is left ear, left eyebrow or left eye, set to draw both left and right
      component */

    for (i=COMPONENT;i<=n;i++)  {
      if (nCurrentStyle[i]>-1) {
            nResource = nCurrentStyle[i];
            if ((i==REAR)||(i==REBROW)||(i==REYE))
                nResource += 5000;
            hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
            hMemDC = CreateCompatibleDC(hDC);
            hOldBmp = SelectObject(hMemDC, hBmp);
            GetObject(hBmp, 16, (LPSTR) &Bitmap);

            if (!bStretchComponent[i])
                BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                    ptPosition[i].y+StartPoint.y-nVscrollPos,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCXNOR);
            else
                StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                    ptPosition[i].y+StartPoint.y-nVscrollPos,
                    ptSize[i].x, ptSize[i].y,
                    hMemDC, 0, 0,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    SRCXNOR);

            SelectObject(hMemDC, hOldBmp);
            DeleteDC(hMemDC);
            DeleteObject(hBmp);

      }
    }

    switch(nDirection) {
        case LEFT:
            ptPosition[COMPONENT].x-=nMoveSizeStep;
            if(n!=COMPONENT) ptPosition[COMPONENT+1].x-=nMoveSizeStep;
            break;
        case RIGHT:
            ptPosition[COMPONENT].x+=nMoveSizeStep;
            if(n!=COMPONENT) ptPosition[COMPONENT+1].x+=nMoveSizeStep;
            break;
        case UP:
            ptPosition[COMPONENT].y-=nMoveSizeStep;
            if(n!=COMPONENT) ptPosition[COMPONENT+1].y-=nMoveSizeStep;
            break;
        case DOWN:
            ptPosition[COMPONENT].y+=nMoveSizeStep;
            if(n!=COMPONENT) ptPosition[COMPONENT+1].y+=nMoveSizeStep;
            break;
    }
    for (i=COMPONENT;i<=n;i++)  {
      if (nCurrentStyle[i]>-1) {
            nResource = nCurrentStyle[i];
            if ((i==REAR)||(i==REBROW)||(i==REYE))
                nResource += 5000;
            hBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nResource));
            hMemDC = CreateCompatibleDC(hDC);
            hOldBmp = SelectObject(hMemDC, hBmp);
            GetObject(hBmp, 16, (LPSTR) &Bitmap);

            if (!bStretchComponent[i])
                BitBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                    ptPosition[i].y+StartPoint.y-nVscrollPos,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hMemDC, 0, 0, SRCXNOR);
            else
                StretchBlt(hDC, ptPosition[i].x+StartPoint.x-nHscrollPos,
                    ptPosition[i].y+StartPoint.y-nVscrollPos,
                    ptSize[i].x, ptSize[i].y,
                    hMemDC, 0, 0,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    SRCXNOR);

            SelectObject(hMemDC, hOldBmp);
            DeleteDC(hMemDC);
            DeleteObject(hBmp);

      }
    }
    EndPaint(hWnd, &ps);
    return (0L);
}
```

```
/*****************************************************************
    PROGRAM: Select.c    (Angelo subprogram)
    PURPOSE: this file contain the routine that create select style dialog box and process
their messages.
*****************************************************************/

#include <windows.h>
#include <stdlib.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************
FUNCTION: SelectComponentProc(HWND, UINT, UINT, LONG)
PURPOSE: Select Chin Style then pass global parameter to current selection
         Processes messages for "Chin Style" dialog box
MESSAGES:
    WM_CREATE   - initialize child dialog box
*****************************************************************/
LONG FAR PASCAL _export SelectComponentProc(HWND hDlg, UINT message,
                                    UINT wParam, LONG lParam)
{
    static RECT         button_rc,rc;
    PAINTSTRUCT         ps;
    HDC                 hDC;
    LPDRAWITEMSTRUCT    lpds;
    char                szBuffer[10];
    switch(message)
    {
        case WM_CREATE:                 /* message: initialize dialog box */
            SendMessage(hDlg,WM_NCACTIVATE,1,NULL);
    /* Set InstanceProc to Operate "TAB" & "ENTER" keycode before "windows
     EDIT Proc */
            hStyleNo[COMPONENT] =
            CreateWindow("edit",NULL,WS_CHILD|WS_VISIBLE|
                        ES_AUTOHSCROLL|ES_AUTOVSCROLL|
                        ES_MULTILINE,188,28,45,15,hDlg,
                        IDC_STYLE,hInst,NULL);
            SetWindowText(hStyleNo[COMPONENT],
                        itoa(nCurrentStyle[COMPONENT],szBuffer,10));

            hOkSid[COMPONENT]       =
            CreateWindow("button","",WS_CHILD|WS_VISIBLE|
                        BS_OWNERDRAW,181,83,65,41,hDlg,
                        IDC_OK,hInst,NULL);
            hCancelSid[COMPONENT]   =
            CreateWindow("button","",WS_CHILD|WS_VISIBLE|
                        BS_OWNERDRAW,181,129,65,41,hDlg,
                        IDC_CANCEL,hInst,NULL);

            hPrevSid[COMPONENT]     =
            CreateWindow("PkOwnerDrawRadioBtn","",WS_CHILD|
                        WS_VISIBLE,174,55,36,29,hDlg,
                        IDC_PREV,hInst,NULL);

            hNextSid[COMPONENT]     =
            CreateWindow("PkOwnerDrawRadioBtn","",WS_CHILD|
                        WS_VISIBLE,210,55,36,29,hDlg,
                        IDC_NEXT,hInst,NULL);

            hSample1[COMPONENT]     =
            CreateWindow("PkOwnerDrawRadioBtn","",WS_CHILD|
                        WS_VISIBLE,7,7,76,76,hDlg,
                        IDC_SAM1+(4*COMPONENT),hInst,NULL);

            hSample2[COMPONENT]     =
            CreateWindow("PkOwnerDrawRadioBtn","",WS_CHILD|
                        WS_VISIBLE,92,7,76,76,hDlg,
                        IDC_SAM2+(4*COMPONENT),hInst,NULL);

            hSample3[COMPONENT]     =
            CreateWindow("PkOwnerDrawRadioBtn","",WS_CHILD|
                        WS_VISIBLE,7,92,76,76,hDlg,
                        IDC_SAM3+(4*COMPONENT),hInst,NULL);

            hSample4[COMPONENT]     =
            CreateWindow("PkOwnerDrawRadioBtn","",WS_CHILD|
                        WS_VISIBLE,92,92,76,76,hDlg,
                        IDC_SAM4+(4*COMPONENT),hInst,NULL);

            button_rc.left = 0;
            button_rc.top = 0;
            button_rc.right = 65;
            button_rc.bottom = 41;

            GetClientRect(hDlg,&rc);
            break;

        case WM_PAINT:
            hDC = BeginPaint(hDlg, &ps);
            DrawFrame(hDC,174,4,72,46,DRAWBOTH); //Style No. frame
            SetBkColor(hDC,RGB(192,192,192));
            TextOut(hDC,188,10,"No.",3);
            DrawFrame (hDC,4,4,80,80,DRAWBOTH); //sample1 frame
            DrawFrame (hDC,89,4,80,80,DRAWBOTH);//sample2 frame
            DrawFrame (hDC,4,89,80,80,DRAWBOTH);//sample3 frame
            DrawFrame (hDC,89,89,80,80,DRAWBOTH);//sample4 frame
            DrawHelpBox(hDC,rc.left+1,rc.bottom-26,rc.right+2,26);
```

```
        EndPaint(hDlg, &ps);
        break;

    case WM_COMMAND:
        if(wParam==(UINT)(IDC_SAM1+(4*COMPONENT)))
            SendMessage(hSample1[COMPONENT],BM_SETCHECK,0,0L);
        if(wParam==(UINT)(IDC_SAM2+(4*COMPONENT)))
            SendMessage(hSample1[COMPONENT],BM_SETCHECK,0,0L);
        if(wParam==(UINT)(IDC_SAM3+(4*COMPONENT)))
            SendMessage(hSample1[COMPONENT],BM_SETCHECK,0,0L);
        if(wParam==(UINT)(IDC_SAM4+(4*COMPONENT)))
            SendMessage(hSample1[COMPONENT],BM_SETCHECK,0,0L);
        switch(wParam)
        {
        case IDC_NEXT:
            SendMessage(hNextStc[COMPONENT],BM_SETCHECK,0,0L);
            UpdateWindow(hNextStc[COMPONENT]);
            nOldStyle=nCurrentStyle[COMPONENT];
            nCurrentStyle[COMPONENT]++;
            if(nCurrentStyle[COMPONENT]==0)
                nCurrentStyle[COMPONENT]=1;
            if(nCurrentStyle[COMPONENT]>nMaxStyle[COMPONENT])
            {
                nCurrentStyle[COMPONENT]=nMaxStyle[COMPONENT];
                return0L;
            }
            SetWindowText(hStyleNo[COMPONENT],
                itoa(nCurrentStyle[COMPONENT],szBuffer,10));
            UpdateWindow(hStyleNo[COMPONENT]);
            SetCapture(hDlg);
            SetCursor(LoadCursor(NULL,IDC_WAIT));

            InvalidateRect(hPicSelectWnd,NULL,TRUE);
            UpdateWindow(hPicSelectWnd);
            InvalidateRect(hPicWnd,NULL,FALSE);
            UpdateWindow(hPicWnd);
            ReleaseCapture();
            break;

        case IDC_PREV:
            SendMessage(hPrevStc[COMPONENT],BM_SETCHECK,0,0L);
            UpdateWindow(hPrevStc[COMPONENT]);
            nOldStyle=nCurrentStyle[COMPONENT];
            nCurrentStyle[COMPONENT]--;
            if(nCurrentStyle[COMPONENT]<1){
                nCurrentStyle[COMPONENT]=1;
                return0L;
            }
            SetWindowText(hStyleNo[COMPONENT],
            itoa(nCurrentStyle[COMPONENT],szBuffer,10));
            UpdateWindow(hStyleNo[COMPONENT]);
            SetCapture(hDlg);
            SetCursor(LoadCursor(NULL,IDC_WAIT));
            InvalidateRect(hPicSelectWnd,NULL,TRUE);
            UpdateWindow(hPicSelectWnd);
            InvalidateRect(hPicWnd,NULL,FALSE);
            UpdateWindow(hPicWnd);
            ReleaseCapture();
            break;

        case IDC_CANCEL:
            nOldStyle=nCurrentStyle[COMPONENT];
            nCurrentStyle[COMPONENT]=nStyleNo;
            ptPosition[COMPONENT]=ptOldPos[0];
            ptSize[COMPONENT]=ptOldSize[0];
            if(  (COMPONENT==LEAR) ||
                 (COMPONENT==LEBROW)||
                 (COMPONENT==LEYE) )
            {
                ptPosition[COMPONENT+1]=ptOldPos[1];
                ptSize[COMPONENT+1]=ptOldSize[1];
            }
            SetWindowText(hStyleNo[COMPONENT],
                    itoa(nCurrentStyle[COMPONENT],szBuffer,10));
            UpdateWindow(hStyleNo[COMPONENT]);
            ShowWindow(hDlg,SW_HIDE);
            ShowWindow(hPicSelectFrame,SW_HIDE);
            ShowWindow(hToolDlg1,SW_SHOW);
            nOperationMode=PIC_CANCEL;
            InvalidateRect(hPicWnd,NULL,FALSE);
            UpdateWindow(hPicWnd);
            break;

        case IDC_OK:
            nOldStyle=nCurrentStyle[COMPONENT];
            ShowWindow(hDlg,SW_HIDE);
            ShowWindow(hPicSelectFrame,SW_HIDE);
            ShowWindow(hToolDlg1,SW_SHOW);
            nOperationMode=PIC_SELECT;
            bStretchComponent[COMPONENT]=FALSE;
            InvalidateRect(hPicWnd,NULL,FALSE);
            UpdateWindow(hPicWnd);
            break;

        case IDC_STYLE:
            switch(HIWORD(lParam))
```

```
                {
                    case EN_SETFOCUS:
                        SendMessage(LOWORD(lParam),EM_SETSEL,
                                NULL,MAKELONG(0,-1));
                        break;
                    case EN_KILLFOCUS:
                        SendMessage(LOWORD(lParam),EM_GETSEL,
                                NULL,MAKELONG(0,-1));
                        break;

                }
                break;
            }
            break;  // End case WM_COMMAND

        case WM_DRAWITEM:
            lpdis=(LPDRAWITEMSTRUCT) lParam;
            if(lpdis->itemState&ODS_SELECTED)
            {
            DrawBitmap(lpdis->hDC,1,1,lpdis->CtlID+8000,SRCCOPY);
            return 0L;
            }
            if(lpdis->itemState&ODS_FOCUS)
            {
            FrameRect (lpdis->hDC,&button_rc,GetStockObject(BLACK_BRUSH));
            DrawBitmap(lpdis->hDC,1,1,lpdis->CtlID+9000,SRCCOPY);
            }
            else
            {
            FrameRect (lpdis->hDC,&button_rc,GetStockObject(WHITE_BRUSH));
            DrawBitmap(lpdis->hDC,1,1,lpdis->CtlID+7000,SRCCOPY);
            }
            break;

        default:
            return(DefWindowProc(hDlg, message, wParam, lParam));
    }
    return(FALSE);          /* Didn't process a message */
}

/*****************************************************************************
FUNCTION: EditStyleProc(HWND, UINT, UINT, LONG)
PURPOSE: To control ENTER & TAB keycode for windows EDIT class
MESSAGES:
    WM_KEYDOWN  - Pass keyboard input of windows EDIT class to this procedure
                  first then continue.
*****************************************************************************/
LONG FAR PASCAL _export EditStyleProc(HWND hEdit, UINT message,
                        UINT wParam, LONG lParam)
{
    switch (message)
    {
        case WM_KEYDOWN:
            switch (wParam)
            {
            case VK_TAB:
                SendMessage(GetParent(hEdit),PK_TAB,NULL,NULL);
                break;


            case VK_RETURN:
                SendMessage(GetParent(hEdit),PK_ENTER,NULL,NULL);
                break;
            }
            break;

    }

    return CallWindowProc(lpfnOldEdit[COMPONENT], hEdit, message, wParam, lParam);
}
```

```
/*****************************************************************************
    PROGRAM: Tool.c  (Angelo subprogram)
    PURPOSE: this file contain the routine that create Tool dialog box and process their
messages.
*****************************************************************************/

#include <windows.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************************
FUNCTION: PkRadioProc(HWND, UINT, UINT, LONG)
PURPOSE: Processes messages for Pako radio button child window.
MESSAGES:
    WM_CREATE   - initialize
    WM_SETCHECK - Button is in focus now.
COMMENTS:
*****************************************************************************/
LONG FAR PASCAL _export PkRadioProc(HWND hButton, UINT message,
                                    UINT wParam, LONG lParam)
{

    PAINTSTRUCT     ps;
    HDC             hDC;
    static RECT     rc;
    static char     text[50];
    static int      textlen;
    static int      x,y;
    static int      press;
    static int      status;
    switch (message)
    {
        case WM_CREATE:
            SetWindowWord(hButton,0,0);
            textlen=GetWindowText(hButton,text,50);
            rc.left  = 0;
            rc.top   = 0;
            rc.right = 13;
            rc.bottom=13;
            press=0;
            status=0;
            break;

        case WM_PAINT:
            hDC=BeginPaint(hButton,&ps);
            if(status)
                DrawBitmap(hDC,0,0,RADIOSELECT,SRCAND);
            else
                switch(GetWindowWord(hButton,0))
                {
                    case 0 : DrawBitmap(hDC,0,0,RADIOUP,SRCCOPY);
                             break;
                    case 1 : DrawBitmap(hDC,0,0,RADIODOWN,SRCCOPY);
                             break;
                    default: DrawBitmap(hDC,0,0,RADIOUP,SRCCOPY);
                             break;
                }
            SetBkColor(hDC,RGB(192,192,192));
            TextOut(hDC,20,0,text,textlen);
            EndPaint(hButton,&ps);
            break;

        case WM_LBUTTONDOWN:
            x=LOWORD(lParam);
            y=HIWORD(lParam);
            if(x>=0 && y>=0 && x<=13 && y<=13)
            {
                press=1;
                status=1;
                InvalidateRect(hButton,&rc,FALSE);
                SetCapture(hButton);
            }
            break;

        case WM_LBUTTONUP:
            if(!press) return(DefWindowProc(hButton, message, wParam, lParam));
            x=LOWORD(lParam);
            y=HIWORD(lParam);
            status=0;
            press=0;
            ReleaseCapture();
            if(x>=0 && y>=0 && x<=13 && y<=13)
                SendMessage(GetParent(hButton),WM_COMMAND,
                            GetMenu(hButton),NULL);
            InvalidateRect(hButton,&rc,FALSE);
            break;

        case WM_MOUSEMOVE:
            status=0;
            if(!press) return(DefWindowProc(hButton, message, wParam, lParam));
            x=LOWORD(lParam);
            y=HIWORD(lParam);
            if(x>=0 && y>=0 && x<=13 && y<=13)
                status=1;
            InvalidateRect(hButton,&rc,FALSE);
            break;
```

```
        case BM_SETCHECK:
            status = 0;
            SetWindowWord(hButton,0,wParam);
            InvalidateRect(hButton,&rc,FALSE);
            break;
    }
    return (DefWindowProc(hButton, message, wParam, lParam));
}

/*********************************************************************
FUNCTION: PkOwnerDrawRadioProc(HWND, UINT, UINT, LONG)
PURPOSE: Processes messages for Pako owner draw button and Pako owner draw radio
        button child window.
MESSAGES:
  WM_CREATE  - initialize
  WM_SETCHECK - Button is in focus now.
*********************************************************************/
LONG FAR PASCAL _export PkOwnerDrawRadioProc(HWND hButton, UINT message,
                                    UINT wParam, LONG lParam)
{
    PAINTSTRUCT    ps;
    HDC            hDC;
    static RECT    rc;
    static int     x,y;
    static int     press;
    static int     status,inside_state,outside_state;
    static HANDLE  hWorkBtn;
    static HCURSOR hHandCur;

    switch (message)
    {
    case WM_CREATE:
        hWorkBtn=hButton;
        hHandCur=LoadCursor(NULL,"HAND");
        SetWindowWord(hButton,0,0);
        press=0;
        status=0;
        inside_state=0;
        outside_state=0;
        break;
    case WM_PAINT:
        hDC=BeginPaint(hButton,&ps);
        if(status && (hButton==hWorkBtn)) // Draw button down while mouse click
        {
            if(!DrawBitmap(hDC,0,0,2000+GetMenu(hButton),SRCCOPY))
                DrawBitmap(hDC,0,0,1300,SRCINVERT);// Inverse button color
        }
        else
            switch(GetWindowWord(hButton,0))
            {
            case 0: // button up
                DrawBitmap(hDC,0,0,1000+GetMenu(hButton),SRCCOPY);
                break;
            case 1 : // button up & selected
                DrawBitmap(hDC,0,0,3000+GetMenu(hButton),SRCCOPY);
                break;
            default: // button up
                DrawBitmap(hDC,0,0,1000+GetMenu(hButton),SRCCOPY);
                break;
            }
        EndPaint(hButton,&ps);
        break;
    case WM_LBUTTONDOWN:
        x=LOWORD(lParam);
        y=HIWORD(lParam);
        press=1;
        status=1;
        inside_state=1;
        hWorkBtn=hButton;
        SetFocus(hButton);
        GetClientRect(hButton,&rc);
        InvalidateRect(hButton,&rc,FALSE);
        SetCapture(hButton);
        break;

    case WM_LBUTTONUP:
        if(!press) return(DefWindowProc(hButton, message, wParam, lParam));
        x=LOWORD(lParam);
        y=HIWORD(lParam);
        ReleaseCapture();
        status=0;
        press=0;
        GetClientRect(hButton,&rc);
        if(x>=rc.left && y>=rc.top &&    // mouse inside button
         x<=rc.right && y<=rc.bottom)
        {
        SendMessage(GetParent(hButton),WM_COMMAND,GetMenu(hButton),NULL);
        InvalidateRect(hButton,&rc,FALSE);
        }
        break;
    case WM_MOUSEMOVE:
        status=0;
        if(!press) return(DefWindowProc(hButton, message, wParam, lParam));
        x=LOWORD(lParam);
        y=HIWORD(lParam);
        if(x>=rc.left && y>=rc.top &&  // mouse inside button
         x<=rc.right && y<=rc.bottom)
        {
```

```
            status = 1;             // radio button down now!
            switch(inside_state)
            {
                case 0 : inside_state = 1;   // draw button down
                    InvalidateRect(hButton,&rc,FALSE);
                    break;
                case 1 : outside_state = 0;
                    inside_state = 2;
                    break;
                case 2 : break;         // remain status after draw down button
            }
        }
        else                        // mouse outside button
        {
            switch(outside_state)
            {
                case 0 : outside_state = 1;   // draw button up
                    InvalidateRect(hButton,&rc,FALSE);
                    break;
                case 1 : inside_state = 0;  // button is already refresh
                    outside_state = 2;
                    break;
                case 2 : break;         // remain status after draw down button
            }

        }
        break;

    case BM_SETCHECK:
        status = 0;
        SetWindowWord(hButton,0,wParam);
        GetClientRect(hButton,&rc);
        InvalidateRect(hButton,&rc,FALSE);
        break;
    }
    return (DefWindowProc(hButton, message, wParam, lParam));
}

/*****************************************************************************
FUNCTION: ComponentDlgProc(HWND, UINT, UINT, LONG)
PURPOSE: Processes messages for "COMPONENT" dialog box
MESSAGES:
    WM_CREATE   - initialize child dialog box
    WM_DRAWITEM - Input received
*****************************************************************************/
LONG FAR PASCAL _export ComponentDlgProc(HWND hDlg, UINT message,
                                         UINT wParam, LONG lParam)
{
HDC             hDC;

static HWND     hHairBtn,
                hChinBtn,
                hEarBtn,
                hEbrowBtn,
                hEyeBtn,
                hNoseBtn,
                hMouthBtn;

static HWND     hCurrentComponent;
static int      OldComponent;
LPDRAWITEMSTRUCT lpdis;

switch(message)
{
    case WM_CREATE:                 /* message: initialize dialog box */
        hCurrentComponent=NULL;
        OldComponent=-1;
        bNoCompSelect=TRUE;
        COMPONENT=-1; //None of component selected
        SendMessage(hDlg,WM_NCACTIVATE,1,NULL);
        hHairBtn = CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    0,0,110,100,hDlg,IDC_HAIR,hInst,NULL);
        hChinBtn = CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    0,100,110,100,hDlg,IDC_CHIN,hInst,NULL);
        hEarBtn  = CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    110,0,140,50,hDlg,IDC_EAR,hInst,NULL);
        hEbrowBtn= CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    110,50,140,50,hDlg,IDC_EBROW,hInst,NULL);
        hEyeBtn  = CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    110,100,140,50,hDlg,IDC_EYE,hInst,NULL);
        hNoseBtn = CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    110,150,70,50,hDlg,IDC_NOSE,hInst,NULL);
        hMouthBtn= CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    180,150,70,50,hDlg,IDC_MOUTH,hInst,NULL);
        hEditBtn = CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    0,200,110,30,hDlg,IDC_EDIT,hInst,NULL);
        hSelectBtn= CreateWindow("PkOwnerDrawRadioBtn","",
                    WS_CHILD|WS_VISIBLE,
                    110,200,140,30,hDlg,IDC_SELECT,hInst,NULL);

        return (TRUE);
```

```
caseWM_COMMAND:
    switch(wParam)
    {
        caseIDC_HAIR:
        caseIDC_CHIN:
        caseIDC_EAR:
        caseIDC_EBROW:
        caseIDC_EYE:
        caseIDC_NOSE:
        caseIDC_MOUTH:
                OldComponent=COMPONENT;
        switch(wParam)
        {
            caseIDC_HAIR: COMPONENT=HAIR;break;
            caseIDC_CHIN: COMPONENT=CHIN;break;
            caseIDC_EAR: COMPONENT=LEAR;break;
            caseIDC_EBROW:COMPONENT=LEBROW;break;
            caseIDC_EYE: COMPONENT=LEYE;break;
            aseIDC_NOSE: COMPONENT=NOSE;break;
            caseIDC_MOUTH:COMPONENT=MOUTH;break;
        }
        CheckRadioButton(hDlg,IDC_HAIR,IDC_MOUTH,wParam);
        if  (IsWindowVisible(hToolDlg2) &&
        (OldComponent!=COMPONENT)
        )
        {
            if(!bValidComponent)
            {
                //Update last component
                MoveSizeRedraw(hDlg);
                //Show Scaling frame point
                SetScalingPoint(COMPONENT);
            }
            else
            {
                //Check exist previous Component check?
                if(!bNoCompSelect)
                    *//Delete last Scaling frame point
                     SetScalingPoint(OldComponent);
                //Show selected Scaling frame point
                SetScalingPoint(COMPONENT);
            }
            ptOldPos[0]=ptPosition[COMPONENT];
            ptOldSize[0]=ptSize[COMPONENT];
            //REAR, REBROW, REYE
            if(    (COMPONENT==LEAR) ||
                (COMPONENT==LEBROW)||
                (COMPONENT==LEYE))
            {ptOldPos[1]=ptPosition[COMPONENT+1];
             ptOldSize[1]=ptSize[COMPONENT+1];}

        }

        if(!IsWindowVisible(hToolDlg2))
        {
        if(nEditMode==PIC_MOVE)
         SendMessage(hToolDlg2,WM_COMMAND,IDC_PICMOVE,NULL);
        if(nEditMode==PIC_SIZE)
         SendMessage(hToolDlg2,WM_COMMAND,IDC_PICSIZE,NULL);
        }
        bValidComponent=TRUE;
        hCurrentComponent=hSelectComponent[COMPONENT];
        bNoCompSelect=FALSE;
        break;

        caseIDC_SELECT:
            SendMessage(hEditBtn,BM_SETCHECK,0,0L);
            SendMessage(hSelectBtn,BM_SETCHECK,0,0L);
            if(IsWindowVisible(hToolDlg2))
            {
                ShowWindow(hToolDlg2,SW_HIDE);
                if((!bValidComponent) &&
                 (hCurrentComponent!=NULL))
                        MoveSizeRedraw(hDlg);
                if((bValidComponent) &&
                 (hCurrentComponent!=NULL))
                        //Delete scaling frame point
                        SetScalingPoint(COMPONENT);
            }
            if(hCurrentComponent!=NULL)
            {
                SetCapture(hDlg);
                SetCursor(LoadCursor(NULL,IDC_WAIT));
                ShowWindow(hDlg,SW_HIDE);
                ShowWindow(hCurrentComponent,SW_SHOW);

            //Save Old Style, position and size of current COMPONENT
                nStyleNo=nCurrentStyle[COMPONENT];
            //ALL COMPONENT
                ptOldPos[0]=ptPosition[COMPONENT];
                ptOldSize[0]=ptSize[COMPONENT];
            //REAR, REBROW, REYE
                if((COMPONENT==LEAR) ||
                 (COMPONENT==LEBROW)||
                 (COMPONENT==LEYE))
                {ptOldPos[1]=ptPosition[COMPONENT+1];
                 ptOldSize[1]=ptSize[COMPONENT+1];}
                nOperationMode=PIC_DELETE;
```

```
                InvalidateRect(hPicSelectWnd,NULL,TRUE);
                UpdateWindow(hPicSelectWnd);
                ShowWindow(hPicSelectFrame,SW_SHOW);
                InvalidateRect(hPicWnd,NULL,TRUE);
                UpdateWindow(hPicWnd);
                nOperationMode=PIC_XOR;
                ReleaseCapture();
            }
            break;

        case IDC_EDIT:
            if(!IsWindowVisible(hToolDlg2))
            {
                SendMessage(hEditBtn,BM_SETCHECK,1,0L);
                bValidComponent=TRUE;
                ShowWindow(hToolDlg2,SW_SHOW);
            //Save Old position & size current COMPONENT
                ptOldPos[0]=ptPosition[COMPONENT];
                ptOldSize[0]=ptSize[COMPONENT];
            //REAR, REBROW, REYE
                if((COMPONENT==LEAR) ||
                 (COMPONENT==LEBROW)||
                 (COMPONENT==LEYE))
                 {ptOldPos[1]=ptPosition[COMPONENT+1];
                 ptOldSize[1]=ptSize[COMPONENT+1];}

                SendMessage(hToolDlg2,WM_COMMAND,
                        IDC_PICMOVE,NULL);
            //Exist Selelected COMPONENT?
                if(hCurrentComponent!=NULL)
                //show scaling frame point
                  SetScalingPoint(COMPONENT);

            }
            break;
        default:
            break;
    }
    break;
case WM_DRAWITEM:
    lpdis=(LPDRAWITEMSTRUCT)lParam;
    if(lpdis->itemState&ODS_SELECTED)
        DrawBitmap(lpdis->hDC,0,0,lpdis->CtlID+2000,SRCCOPY);
    else
        DrawBitmap(lpdis->hDC,0,0,lpdis->CtlID+1000,SRCCOPY);
    break;
case WM_CLOSE:
    DestroyWindow(hDlg);        /* Exits the dialog box */
    return(TRUE);
        default:
            return (DefWindowProc(hDlg, message, wParam, lParam));
    }
    return(FALSE);          /* Didn't process a message */
}

/****************************************************************************
FUNCTION: ToolOperateDlgProc(HWND, UINT, UINT, LONG)
PURPOSE: Processes messages for "COMPONENT" dialog box
MESSAGES:
    WM_CREATE  - initialize child dialog box
****************************************************************************/
LONG FAR PASCAL _export ToolOperateDlgProc(HWND hDlg, UINT message,
                                    UINT wParam, LONG lParam)
{
    PAINTSTRUCT     ps;
    HDC             hDC;
    LPDRAWITEMSTRUCT lpdis;
    static HWND     hFocus,hOkBtn,hCancelBtn,hRadioMove,hRadioSize,
                    hRadioLeft,hRadioRight,hRadioTop,hRadioBottom,hRadioCentre;
    static RECT     rc[5],rcpic,button_rc;
    static short    SelectArrow;
    int             x,y; /* mouse position */
    int             count;
    static  int     x_start=171, y_start=20;
    static int      Direction;
    static int      LeftRight,TopBottom;    /* save origin position point to resize */
                    //LeftRight-LEFT or RIGHT
                    //TopBottom-TOP or BOTTOM
    static          Press;
    switch(message)
    {
        case WM_CREATE:             /* message: initialize dialog box */
            SendMessage(hDlg,WM_NCACTIVATE,1,NULL);
            hOkBtn    =CreateWindow("button","",WS_CHILD|WS_VISIBLE|
                                BS_OWNERDRAW,x_start-2,91,65,41,hDlg,
                                IDC_OK,hInst,NULL);

            hCancelBtn =CreateWindow("button","",WS_CHILD|WS_VISIBLE|
                                BS_OWNERDRAW,x_start-2,144,65,41,hDlg,
                                IDC_CANCEL,hInst,NULL);

            button_rc.left  = 0;
            button_rc.top   = 0;
            button_rc.right = 65;
            button_rc.bottom= 41;
            SetFocus(hCancelBtn);

            hRadioMove =CreateWindow("PkRadioBtn","",WS_CHILD|
```

```
                    WS_VISIBLE,30,30,20,20,hDlg,
                    IDC_PICMOVE,hInst,NULL);

hRadioSize = CreateWindow("PkRadioBtn","",WS_CHILD|
                    WS_VISIBLE,30,60,20,20,hDlg,
                    IDC_PICSIZE,hInst,NULL);

hRadioLeft = CreateWindow("PkRadioBtn","",WS_CHILD|
                    WS_VISIBLE,30,102,20,20,hDlg,
                    IDC_LEFT,hInst,NULL);

hRadioRight = CreateWindow("PkRadioBtn","",WS_CHILD|
                    WS_VISIBLE,30,132,20,20,hDlg,
                    IDC_RIGHT,hInst,NULL);

hRadioTop = CreateWindow("PkRadioBtn","",WS_CHILD|
                    WS_VISIBLE,105,102,20,20,hDlg,
                    IDC_TOP,hInst,NULL);

hRadioBottom = CreateWindow("PkRadioBtn","",WS_CHILD|
                    WS_VISIBLE,105,132,20,20,hDlg,
                    IDC_BOTTOM,hInst,NULL);

hRadioCentre = CreateWindow("PkRadioBtn","",WS_CHILD|
                    WS_VISIBLE,45,162,20,20,hDlg,
                    IDC_CENTRE,hInst,NULL);
LeftRight = IDC_LEFT;
TopBottom = IDC_TOP;
SendMessage(hRadioMove,BM_SETCHECK,1,0L);
nEditMode = PIC_MOVE;

SelectArrow = IDC_DIRECT;
Direction = 0;
Press = 0;
rcpic.left = x_start;
rcpic.top = y_start;
rcpic.right = x_start+61;
rcpic.bottom = y_start+61;

/* LEFTARROW */
rc[1].left = x_start+0;
rc[1].top = y_start+23;
rc[1].right = x_start+0+22;
rc[1].bottom = y_start+23+15;

/* RIGHTARROW */
rc[2].left = x_start+39;
rc[2].top = y_start+23;
rc[2].right = x_start+39+22;
rc[2].bottom = y_start+23+15;

/* UPARROW */
rc[3].left = x_start+23;
rc[3].top = y_start+0;
rc[3].right = x_start+23+15;
rc[3].bottom = y_start+0+22;

/* DOWNARROW */
rc[4].left = x_start+23;
rc[4].top = y_start+39;
rc[4].right = x_start+23+15;
rc[4].bottom = y_start+39+22;

return(TRUE);

case WM_PAINT:
    hDC = BeginPaint(hDlg, &ps);
    DrawFrame (hDC,19,20,x_start-19,61,DRAWBK);
    DrawFrame (hDC,19,19,x_start-19+62,63,DRAWFRAME);
    DrawFrame (hDC,19,91,x_start-29,92,DRAWBOTH);
    DrawBitmap(hDC,x_start,y_start,1000+SelectArrow,SRCCOPY);
    DrawBitmap(hDC,50,25,1216,SRCAND);
    DrawBitmap(hDC,50,55,1215,SRCAND);
    DrawBitmap(hDC,50,97,1217,SRCAND); //left  text
    DrawBitmap(hDC,50,127,1218,SRCAND);//right text
    DrawBitmap(hDC,125,97,1219,SRCAND);//top  text
    DrawBitmap(hDC,125,127,1220,SRCAND);//bottom text
    DrawBitmap(hDC,65,157,1221,SRCAND);//centre text
    EndPaint(hDlg, &ps);
    break;

case WM_LBUTTONDOWN:
    SetFocus(hFocus);
    Press = 1;
    x=LOWORD(lParam);
    y=HIWORD(lParam);
    SetCapture(hDlg);
    for(count = 1; count <= 4; count++)
    {
  if((x >= rc[count].left) && (y >= rc[count].top) &&
    (x <= rc[count].right) && (y <= rc[count].bottom))
    {
    switch(count)
      {
     case LEFT : SelectArrow = IDC_ARRLEFT;
            Direction = LEFT;
            break;
     case RIGHT : SelectArrow = IDC_ARRRIGHT;
```

```
                Direction = RIGHT;
                break;
            case UP  : SelectArrow = IDC_ARRUP;
                Direction = UP;
                break;
            case DOWN : SelectArrow = IDC_ARRDOWN;
                Direction = DOWN;
                break;
             }
           InvalidateRect(hDlg,&rcpic,FALSE);
           count = 4;
           }
         }
         break;

       caseWM_LBUTTONUP:
         x = LOWORD(lParam);
         y = HIWORD(lParam);
          Press = 0;
         //Move fast when press control key while button UP
         if(wParam == MK_CONTROL) nMoveSizeStep = 3;
         else nMoveSizeStep = 1;

         ReleaseCapture();
         SelectArrow = IDC_DIRECT;
         InvalidateRect(hDlg,&rcpic,FALSE);
         UpdateWindow(hDlg);
         if(bNoCompSelect) return 0L;
         for(count = 1; count <= 4; count++)
           {
           if((x >= rc[count].left) && (y >= rc[count].top) &&
              (x <= rc[count].right) && (y <= rc[count].bottom)&&
               (Direction == count))
               {

               if(nOperationMode != PIC_GAP)
                  {
                  bValidComponent = FALSE;
                  Send2SizeMove(LeftRight, TopBottom, Direction);
                  }
               else
                  if( (count == LEFT) || (count == RIGHT) )  //UP & DOWN take
noeffect
                  {
                  bValidComponent = FALSE;
                  Send2SizeMove(LeftRight, TopBottom, Direction);
                  }
               count = 4;
               }
           }
         Direction = 0; /* Reset Direction */
         break;

       caseWM_MOUSEMOVE:
         SetCursor(hMyCursor);
         if(!Press) return(DefWindowProc(hDlg, message, wParam, lParam));
         x = LOWORD(lParam);
         y = HIWORD(lParam);
         SelectArrow = IDC_DIRECT;
         for(count = 1; count <= 4; count++)
           {
           if((x >= rc[count].left) && (y >= rc[count].top) &&
              (x <= rc[count].right) && (y <= rc[count].bottom)&&
              (Direction == count))
              {
              switch(count)
                {
                case 1 : SelectArrow = IDC_ARRLEFT;
                     break;
                case 2 : SelectArrow = IDC_ARRRIGHT;
                     break;
                case 3 : SelectArrow = IDC_ARRUP;
                     break;
                case 4 : SelectArrow = IDC_ARRDOWN;
                     break;
                }
              count = 4;
              }
           }
         InvalidateRect(hDlg,&rcpic,FALSE);
         break;

       caseWM_COMMAND:
         switch(wParam)
           {
           case IDC_PICMOVE:
             CheckRadioButton(hDlg,IDC_PICMOVE,IDC_PICSIZE,wParam);
             CheckRadioButton(hDlg,IDC_LEFT,IDC_BOTTOM,NULL);
             UpdateWindow(hDlg);
             nOperationMode = PIC_MOVE;
             nEditMode = nOperationMode;
              break;

           case IDC_PICSIZE:
             CheckRadioButton(hDlg,IDC_PICMOVE,IDC_PICSIZE,wParam);
             CheckRadioButton(hDlg,IDC_LEFT,IDC_CENTRE,LeftRight);
             CheckRadioButton(hDlg,IDC_TOP,IDC_BOTTOM,TopBottom);
             UpdateWindow(hDlg);
```

```
nOperationMode = PIC_SIZE;
nEditMode = nOperationMode;
break;

case IDC_LEFT:
if(!GetWindowWord(hRadioMove,0))
  {
    LeftRight = wParam;
    CheckRadioButton(hDlg,IDC_LEFT,IDC_CENTRE,wParam);
    CheckRadioButton(hDlg,IDC_TOP,IDC_BOTTOM,TopBottom);
    nOperationMode = PIC_SIZE;
  }
break;

case IDC_RIGHT:
if(!GetWindowWord(hRadioMove,0))
  {
    LeftRight = wParam;
    CheckRadioButton(hDlg,IDC_LEFT,IDC_CENTRE,wParam);
    CheckRadioButton(hDlg,IDC_TOP,IDC_BOTTOM,TopBottom);
    nOperationMode = PIC_SIZE;
  }
break;

case IDC_TOP:
if(!GetWindowWord(hRadioMove,0))
  {
    TopBottom = wParam;
    CheckRadioButton(hDlg,IDC_CENTRE,IDC_BOTTOM,wParam);
    CheckRadioButton(hDlg,IDC_LEFT,IDC_RIGHT,LeftRight);
    nOperationMode = PIC_SIZE;
  }
break;

case IDC_BOTTOM:
if(!GetWindowWord(hRadioMove,0))
  {
    TopBottom = wParam;
    CheckRadioButton(hDlg,IDC_CENTRE,IDC_BOTTOM,wParam);
    CheckRadioButton(hDlg,IDC_LEFT,IDC_RIGHT,LeftRight);
    nOperationMode = PIC_SIZE;
  }
break;

case IDC_CENTRE:
if( ((COMPONENT == LEAR) ||
    (COMPONENT == LEBROW)||
    (COMPONENT == LEYE))
      &
    (!GetWindowWord(hRadioMove,0))
   )
    {
    CheckRadioButton(hDlg,IDC_LEFT,IDC_BOTTOM,wParam);
    UpdateWindow(hDlg);
    nOperationMode = PIC_GAP;
    }
  break;

case IDC_OK:
SendMessage(hEditBtn,BM_SETCHECK,0,0L);
ShowWindow(hDlg,SW_HIDE);
if(!bValidComponent)
    MoveSizeRedraw(hDlg);
   else
    if( COMPONENT != -1)
     SetScalingPoint(COMPONENT);
  break;

case IDC_CANCEL:
SendMessage(hEditBtn,BM_SETCHECK,0,0L);
ShowWindow(hDlg,SW_HIDE);
if(!bValidComponent)
    {
    // restore old position & size
    ptPosition[COMPONENT] = ptOldPos[0];
    ptSize[COMPONENT] = ptOldSize[0];
    // REAR, REBROW, REYE
     if((COMPONENT == LEAR) ||
      (COMPONENT == LEBROW)||
      (COMPONENT == LEYE)
      )
      {ptPosition[COMPONENT+1] = ptOldPos[1];
       ptSize[COMPONENT+1] = ptOldSize[1];}

    SetCapture(hDlg);
    SetCursor(LoadCursor(NULL,IDC_WAIT));
    nOperationMode = PIC_REDRAW;
    InvalidateRect(hPicWnd,NULL,TRUE);
    UpdateWindow(hPicWnd);
    ReleaseCapture();
    }
    else
     if( COMPONENT != -1)
      SetScalingPoint(COMPONENT);
  break;
}

SetFocus(hFocus);
```

```
            break;

        caseWM_DRAWITEM:
          lpdis=(LPDRAWITEMSTRUCT)lParam;
          if(lpdis->itemState&ODS_SELECTED)
            {
            DrawBitmap(lpdis->hDC,1,1,lpdis->CtlID+8000,SRCCOPY);
            return0;
            }
          if(lpdis->itemState&ODS_FOCUS)
            {
            FrameRect(lpdis->hDC,&button_rc,GetStockObject(BLACK_BRUSH));
            DrawBitmap(lpdis->hDC,1,1,lpdis->CtlID+9000,SRCCOPY);
            if(lpdis->CtlID==IDC_OK)hFocus=hOkBtn;
            if(lpdis->CtlID==IDC_CANCEL)hFocus=hCancelBtn;
            }
           else
            {
            FrameRect(lpdis->hDC,&button_rc,GetStockObject(WHITE_BRUSH));
            DrawBitmap(lpdis->hDC,1,1,lpdis->CtlID+7000,SRCCOPY);
            }
          break;

        caseWM_CLOSE:
          DestroyWindow(hDlg);          /* Exits the dialog box */
          return(TRUE);

        default:
          return(DefWindowProc(hDlg,message,wParam,lParam));
    }

  return(FALSE);          /* Didn't process a message */
}

/*****************************************************************************
FUNCTION:Send2SizeMove(int,int,int)
PURPOSE: Find direction & point to move or resize. Then Invalidate hPicWnd.
*****************************************************************************/
voidSend2SizeMove(intLeftRight,intTopBottom,intDirection)
{
 if(LeftRight==IDC_LEFT)
   {
  if(TopBottom==IDC_TOP)nPointResize=0;
  if(TopBottom==IDC_BOTTOM)nPointResize=5;
   }
 if(LeftRight==IDC_RIGHT)
   {
  if(TopBottom==IDC_TOP)nPointResize=2;
  if(TopBottom==IDC_BOTTOM)nPointResize=7;
   }
 nDirection=Direction;
 InvalidateRect(hPicWnd,NULL,FALSE);
}

/*****************************************************************************
FUNCTION:MoveSizeRedraw(HWND)
PURPOSE: Find size,position and direction then Invalidate hPicWnd
*****************************************************************************/
voidMoveSizeRedraw(HWNDhWnd)
{
 SetCapture(hWnd);
 SetCursor(LoadCursor(NULL,IDC_WAIT));
 nOperationMode=PIC_REDRAW;
 InvalidateRect(hPicWnd,NULL,TRUE);
 UpdateWindow(hPicWnd);
 bValidComponent=TRUE;
 // Save Old Style, position and size of current COMPONENT
 nStyleNo=nCurrentStyle[COMPONENT];
  // ALL COMPONENT
  ptOldPos[0]=ptPosition[COMPONENT];
  ptOldSize[0]=ptSize[COMPONENT];
  // REAR , REBROW, REYE
  if((COMPONENT==LEAR)||
    (COMPONENT==LEBROW)||
    (COMPONENT==LEYE)
    )
    {ptOldPos[1]=ptPosition[COMPONENT+1];
     ptOldSize[1]=ptSize[COMPONENT+1];}

 ReleaseCapture();
}

HBITMAPDrawBitmap(HDChdc,intX,intY,WORDpic,DWORDdwRop)
{
  BITMAP    bm;
  HDC   hMemDC;
  POINT pt;
  HANDLE    hBitmap;
 hBitmap=LoadBitmap(hInst,MAKEINTRESOURCE(pic));
 if(!hBitmap)
   return(NULL);

 hMemDC=CreateCompatibleDC(hdc);
 SelectObject(hMemDC,hBitmap);
 GetObject(hBitmap,sizeof(BITMAP),(LPSTR)&bm);
```

```
    pt.x = bm.bmWidth;
    pt.y = bm.bmHeight;

    BitBlt(hdc, X, Y, pt.x, pt.y, hMemDC, 0, 0, dwRop);

    DeleteDC(hMemDC);
    DeleteObject(hBitmap);
    return(1);
}

void DrawFrame(HDC hDC, int x, int y, int x_size, int y_size, int mode)
{
    RECT    rc;
    HPEN    hPenDKGRAY;
    rc.left  = x;
    rc.right  = x+x_size;
    rc.top  = y;
    rc.bottom    = y+y_size;
    hPenDKGRAY=CreatePen(PS_SOLID,1,RGB(128,128,128));

    if(mode==DRAWBK||mode==DRAWBOTH)
      FillRect(hDC,&rc,GetStockObject(LTGRAY_BRUSH));

    if(mode==DRAWFRAME||mode==DRAWBOTH)
      {
      SelectObject(hDC,hPenDKGRAY);
      MoveTo(hDC,rc.left,rc.top);
      LineTo(hDC,rc.left,rc.bottom);

      MoveTo(hDC,rc.left,rc.top);
      LineTo(hDC,rc.right,rc.top);

      SelectObject(hDC,GetStockObject(WHITE_PEN));
      MoveTo(hDC,rc.right,rc.top+1);
      LineTo(hDC,rc.right,rc.bottom);

      MoveTo(hDC,rc.left+1,rc.bottom);
      LineTo(hDC,rc.right,rc.bottom);
      }

    DeleteObject(hPenDKGRAY);
}


void DrawHelpBox(HDC hDC, int x, int y, int x_size, int y_size)
{
    RECT    rc;
    HPEN    hPenDKGRAY;
    if ( (x_size <= 9) || (y_size <= 9) )
    return;

    hPenDKGRAY=CreatePen(PS_SOLID,1,RGB(156,156,156));

    rc.left  = x;
    rc.right  = x+x_size;
    rc.top  = y;
    rc.bottom    = y+y_size;
    FillRect(hDC,&rc,GetStockObject(LTGRAY_BRUSH));

    rc.left  = x+1;
    rc.right  = x+x_size;
    rc.top  = y+1;
    rc.bottom    = y+y_size;
    FrameRect(hDC,&rc,GetStockObject(WHITE_BRUSH));

    rc.left  = x;
    rc.right  = x+x_size;
    rc.top  = y;
    rc.bottom    = y+y_size;
    FrameRect(hDC,&rc,GetStockObject(BLACK_BRUSH));

    DrawFrame(hDC,x+4,y+4,x_size-8,y_size-8,DRAWFRAME);
    DeleteObject(hPenDKGRAY);
}
```

```
/*****************************************************************************
    PROGRAM: GetPos.c (Angelo subprogram)
    PURPOSE: this file contain all routine that involve reading necessary value from
STRINGTABLE in resource script file.
*****************************************************************************/

#include <windows.h>
#include <stdlib.h>
#include "function.h"

extern HANDLE hInst;

/*****************************************************************************
    FUNCTION: GetComponentPos (int, int)
    PURPOSE: Get position of each component style from STRINGTABLE in resource
script file.
*****************************************************************************/
POINT GetComponentPos (int nComponent, int nStyle)
{
    POINT ptSetPoint;
    int nID;

    nID = (nComponent * 1000) + nStyle;
    ptSetPoint.x = atoi(String(nID));
    nID += 10000;
    ptSetPoint.y = atoi(String(nID));
    return (ptSetPoint);
}

/*****************************************************************************
    FUNCTION: GetMaxStyle (int)
    PURPOSE: Get maximum style of each component from STRINGTABLE in resource
script file.
*****************************************************************************/
int GetMaxStyle (int nComponent)
{
    int nMax;
    int nID;

    nID = nComponent + 30000;
    nMax = atoi(String(nID));
    return (nMax);
}

/*****************************************************************************
    FUNCTION: *String (WORD)
    PURPOSE: Load string from STRINGTABLE
*****************************************************************************/
char *String (WORD wID)
{
    char szBuffer[256];

    LoadString (hInst, wID, szBuffer, 256);
    return szBuffer;
}
```

```
/*-----------------------------------------------------------
  POPFILE.C -- Popup File Functions
-----------------------------------------------------------*/

#include <windows.h>
#include <commdlg.h>
#include <stdlib.h>
#include "define.h"
#include "globalex.h"

static OPENFILENAME ofn;

void PopFileInitialize (HWND hwnd)
     {
     static char * szFilter[] = { "Angelo Files (*.AGL)", "*.agl",
                                  "All Files (*.*)",      "*.*",
                                  "" } ;

     ofn.lStructSize       = sizeof (OPENFILENAME) ;
     ofn.hwndOwner         = hwnd ;
     ofn.hInstance         = NULL ;
     ofn.lpstrFilter       = szFilter [0] ;
     ofn.lpstrCustomFilter = NULL ;
     ofn.nMaxCustFilter    = 0 ;
     ofn.nFilterIndex      = 0 ;
     ofn.lpstrFile         = NULL ;     // Set in Open and Close functions
     ofn.nMaxFile          = _MAX_PATH ;
     ofn.lpstrFileTitle    = NULL ;     // Set in Open and Close functions
     ofn.nMaxFileTitle     = _MAX_FNAME + _MAX_EXT ;
     ofn.lpstrInitialDir   = NULL ;
     ofn.lpstrTitle        = NULL ;
     ofn.Flags             = 0 ;        // Set in Open and Close functions
     ofn.nFileOffset       = 0 ;
     ofn.nFileExtension    = 0 ;
     ofn.lpstrDefExt       = "agl" ;
     ofn.lCustData         = 0L ;
     ofn.lpfnHook          = NULL ;
     ofn.lpTemplateName    = NULL ;
     }

BOOL PopFileOpenDlg (HWND hwnd, LPSTR lpstrFileName, LPSTR lpstrTitleName)
     {
     ofn.hwndOwner     = hwnd ;
     ofn.lpstrFile     = lpstrFileName ;
     ofn.lpstrFileTitle = lpstrTitleName ;
     ofn.Flags         = OFN_CREATEPROMPT ;

     return GetOpenFileName (&ofn) ;
     }

BOOL PopFileSaveDlg (HWND hwnd, LPSTR lpstrFileName, LPSTR lpstrTitleName)
     {
     ofn.hwndOwner     = hwnd ;
     ofn.lpstrFile     = lpstrFileName ;
     ofn.lpstrFileTitle = lpstrTitleName ;
     ofn.Flags         = OFN_OVERWRITEPROMPT ;

     return GetSaveFileName (&ofn) ;
     }

static long PopFileLength (int hFile)
     {
     long lCurrentPos = _llseek (hFile, 0L, 1) ;
     long lFileLength = _llseek (hFile, 0L, 2) ;

     _llseek (hFile, lCurrentPos, 0) ;

     return lFileLength ;
     }

BOOL FileRead (LPSTR lpstrFileName)
     {
     long  lLength ;
     int   hFile, i ;
     char  Buffer[10];

     if (-1 == (hFile = _lopen (lpstrFileName, OF_READ | OF_SHARE_DENY_WRITE)))
          return FALSE ;
     if ((lLength = PopFileLength (hFile)) >= 32000)
          {
          _lclose (hFile) ;
          return FALSE ;
          }
     for (i=0;i<=9;i++) {
      _lread (hFile, (LPSTR) Buffer, 4) ;
      nCurrentStyle[i] = atoi(Buffer);
      _lread (hFile, (LPSTR) Buffer, 4) ;
      ptPosition[i].x = atoi(Buffer);
      _lread (hFile, (LPSTR) Buffer, 4) ;
      ptPosition[i].y = atoi(Buffer);
      _lread (hFile, (LPSTR) Buffer, 4) ;
      ptSize[i].x = atoi(Buffer);
      _lread (hFile, (LPSTR) Buffer, 4) ;
      ptSize[i].y = atoi(Buffer);
      }
     _lclose (hFile) ;
     nOperationMode = PIC_REDRAW;
     InvalidateRect(hPicWnd, NULL, TRUE);
```

```
    return TRUE;
     }

BOOL FileWrite(LPSTR lpstrFileName)
     {
   int   hFile ,i;
   LPSTR lpstrBuffer;
   char Buffer[10];

   if (-1 == (hFile = _lopen (lpstrFileName, OF_WRITE | OF_SHARE_EXCLUSIVE)))
     if (-1 == (hFile = _lcreat (lpstrFileName, 0)))
          return FALSE;

     for (i=0;i<=9;i++) {

            if (4 != _lwrite (hFile, itoa( nCurrentStyle[i], Buffer, 10), 4))
              {
       _lclose (hFile);
       return FALSE;
         }
     if (4 != _lwrite (hFile, itoa( ptPosition[i].x, Buffer, 10), 4))
              {
       _lclose (hFile);
       return FALSE;
         }
     if (4 != _lwrite (hFile, itoa( ptPosition[i].y, Buffer, 10), 4))
              {
       _lclose (hFile);
       return FALSE;
         }
     if (4 != _lwrite (hFile, itoa( ptSize[i].x, Buffer, 10), 4))
              {
       _lclose (hFile);
       return FALSE;
         }
     if (4 != _lwrite (hFile, itoa( ptSize[i].y, Buffer, 10), 4))
              {
       _lclose (hFile);
       return FALSE;
         }

      }

     _lclose (hFile);

     return TRUE;
      }
```

```
/*****************************************************************************
   PROGRAM: Print.c(Angelo subprogram)
   PURPOSE: this file contain all routine about print function
*****************************************************************************/

#include<windows.h>
#include<string.h>
#include "define.h"
#include "globalex.h"

extern HBITMAP hBitmapTemp;

HDC GetPrinterDC(void)
   {
  static char szPrinter[80];
  char    *szDevice, *szDriver, *szOutput;

  GetProfileString("windows", "device", ",,,", szPrinter, 80);

  if (NULL != (szDevice = strtok(szPrinter, ",")) &&
    NULL != (szDriver = strtok(NULL,   ", ")) &&
    NULL != (szOutput = strtok(NULL,   ", ")))

      return CreateDC(szDriver, szDevice, szOutput, NULL);

  return 0;
   }

BOOL PrintMyPage(HWND hwnd)
   {
  static char szMessage[] = "Print1: Printing";
  BOOL    bError = FALSE;
  HDC     hdcPrn;
  HDC       hDC;
  HDC       hMemoryDC;
  short    xPage, yPage;
  HBITMAP  hOldBitmap, hBitmapTemp;
  BITMAP     Bitmap;

  if (NULL == (hdcPrn = GetPrinterDC()))
    return TRUE;

  hDC = GetDC(hwnd);
  hMemoryDC = CreateCompatibleDC(hDC);
   hBitmapTemp = CreateCompatibleBitmap(hDC, 400, 540);
   hOldBitmap = SelectObject(hMemoryDC, hBitmapTemp);
   BitBlt(hMemoryDC, 0, 0, 400, 540, hDC,
        StartPoint.x+ptPosition[0].x-nHscrollPos*20,
        StartPoint.y+ptPosition[0].y-nVscrollPos*20, SRCCOPY);

GetObject(hBitmapTemp, sizeof(BITMAP), (LPSTR) &Bitmap);

xPage = GetDeviceCaps(hdcPrn, HORZRES);
yPage = GetDeviceCaps(hdcPrn, VERTRES);

if (Escape(hdcPrn, STARTDOC, sizeof szMessage - 1, szMessage, NULL) > 0)
    {
  StretchBlt(hdcPrn, 250, 230,
      (xPage-10)/2,
     ((yPage-30) * 10)/22,
      hMemoryDC,0,0,
      Bitmap.bmWidth,
      Bitmap.bmHeight,
      SRCCOPY);

   TextOut(hdcPrn, 345, 30, "CODE: A0148167-89",17);
   TextOut(hdcPrn, 345, 58, "NAME: James Wattana",19);
   TextOut(hdcPrn, 345, 86, "AGE: 24 SEX: Male",18);

   if (Escape(hdcPrn, NEWFRAME, 0, NULL, NULL) > 0)
    Escape(hdcPrn, ENDDOC, 0, NULL, NULL);
    else
     bError = TRUE;
    }
  else
    bError = TRUE;

   DeleteDC(hdcPrn);
   SelectObject(hMemoryDC,hOldBitmap);
   DeleteDC(hMemoryDC);
   ReleaseDC(hwnd,hDC);
   return bError;
   }
```

```
/*****************************************************************************
   PROGRAM: AngSub1.c (Angelo subprogram)
*****************************************************************************/
#include <windows.h>
#include <stdio.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************************
   FUNCTION: GetAreaOperation (HWND, UINT, UINT, LONG)
   PURPOSE: Handle some mouse operation in picture window, such as, scaling,
smoothing, blurring etc.
*****************************************************************************/
LONG GetAreaOperation (HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
   static int nResult = 0;

   switch(message) {
      case WM_LBUTTONDOWN:
         switch(GetAreaMode) {
            case SMOOTH_MODE:
               SelectRect.left = SelectRect.right = LOWORD(lParam);
               SelectRect.top = SelectRect.bottom = HIWORD(lParam);
               DisplaySelectFrame(hPicWnd);
               SetCapture(hPicWnd);
               return 0;
            case SCALE_MODE:
               if ((nResult = CheckScalingButtonDown(lParam))>0) {
                  switch(nResult) {
                     case 1:
                        SelectRect.left = LOWORD(lParam);
                        SelectRect.top = HIWORD(lParam);
                        SelectRect.right = ScalingPoint[7].x;
                        SelectRect.bottom = ScalingPoint[7].y;
                     break;
                     case 2:
                        SelectRect.left = ScalingPoint[0].x;
                        SelectRect.top = HIWORD(lParam);
                        SelectRect.right = ScalingPoint[7].x;
                        SelectRect.bottom = ScalingPoint[7].y;
                     break;
                     case 3:
                        SelectRect.left = ScalingPoint[0].x;
                        SelectRect.top = HIWORD(lParam);
                        SelectRect.right = LOWORD(lParam);
                        SelectRect.bottom = ScalingPoint[7].y;
                     break;
                     case 4:
                        SelectRect.left = LOWORD(lParam);
                        SelectRect.top = ScalingPoint[0].y;
                        SelectRect.right = ScalingPoint[7].x;
                        SelectRect.bottom = ScalingPoint[7].y;
                     break;
                     case 5:
                        SelectRect.left = ScalingPoint[0].x;
                        SelectRect.top = ScalingPoint[0].y;
                        SelectRect.right = LOWORD(lParam);
                        SelectRect.bottom = ScalingPoint[7].y;
                     break;
                     case 6:
                        SelectRect.left = LOWORD(lParam);
                        SelectRect.top = ScalingPoint[0].y;
                        SelectRect.right = ScalingPoint[7].x;
                        SelectRect.bottom = HIWORD(lParam);
                     break;
                     case 7:
                        SelectRect.left = ScalingPoint[0].x;
                        SelectRect.top = ScalingPoint[0].y;
                        SelectRect.right = ScalingPoint[7].x;
                        SelectRect.bottom = HIWORD(lParam);
                     break;
                     case 8:
                        SelectRect.left = ScalingPoint[0].x;
                        SelectRect.top = ScalingPoint[0].y;
                        SelectRect.right = LOWORD(lParam);
                        SelectRect.bottom = HIWORD(lParam);
                     break;
                  }
                  DisplaySelectFrame(hPicWnd);
                  SetCapture(hPicWnd);
               }
               return 0;
            case BLUR_MODE:
               hMyCursor = LoadCursor (NULL, IDC_SIZE);
               SetCapture(hPicWnd);
               bBlurring = TRUE;
               BlurringArea(hPicWnd, lParam);
               return 0;
            case MOVE_MODE:
               MoveOperation(hWnd,message,wParam,lParam);
               return 0;
         }
         return 0;
      case WM_MOUSEMOVE:
         switch(wParam) {
            case MK_LBUTTON:
               switch (GetAreaMode) {
```

```
case SMOOTH_MODE:
  DisplaySelectFrame(hPicWnd);
  SelectRect.right = LOWORD(lParam);
    SelectRect.bottom = HIWORD(lParam);
  DisplaySelectFrame(hPicWnd);
    return 0;
case SCALE_MODE:
  if (nResult!=0) {
    DisplaySelectFrame(hPicWnd);
    switch(nResult) {
     case 1:
              SelectRect.left = LOWORD(lParam);
              SelectRect.top = HIWORD(lParam);
              break;
     case 2:
              SelectRect.top = HIWORD(lParam);
              break;
     case 3:
              SelectRect.top = HIWORD(lParam);
              SelectRect.right = LOWORD(lParam);
              break;
     case 4:
              SelectRect.left = LOWORD(lParam);
              break;
     case 5:
              SelectRect.right = LOWORD(lParam);
              break;
     case 6:
              SelectRect.left = LOWORD(lParam);
              SelectRect.bottom = HIWORD(lParam);
              break;
     case 7:
              SelectRect.bottom = HIWORD(lParam);
              break;
    case 8:
              SelectRect.right = LOWORD(lParam);
              SelectRect.bottom = HIWORD(lParam);
              break;
     }
     DisplaySelectFrame(hPicWnd);
    }
    return 0;
case BLUR_MODE:
    if (bBlurring) {
     RECT rcTemp;
     rcTemp.left = LOWORD(lParam)-3;
     rcTemp.top = HIWORD(lParam)-3;
     rcTemp.right = LOWORD(lParam)+3;
     rcTemp.bottom = HIWORD(lParam)+3;
        SmoothingArea(hPicWnd,rcTemp);
      }
      return 0;
   }
   return 0;

   default:
    switch(GetAreaMode) {
      case SCALE_MODE:
       CheckScaling(lParam);
       break;
      case MOVE_MODE:
       MoveOperation(hWnd,message,wParam,lParam);
       return 0;
    }
          }
          return 0;
case WM_LBUTTONUP:
    /* bGetArea must be set to be FALSE to end Scaling and Smoothing Operation */
    switch (GetAreaMode) {
      case SMOOTH_MODE:
        DisplaySelectFrame(hPicWnd);
        ReleaseCapture();
        SmoothingArea(hPicWnd, SelectRect);
        GetMainBitmap(hPicWnd);
          return 0;
      case SCALE_MODE:
       if (nResult!=0) {
          DisplaySelectFrame(hPicWnd);
          ReleaseCapture();
          hMyCursor = hArrowCursor;
          nResult = 0;
        }
        return 0;
        case BLUR_MODE:
          if (bBlurring) {
          bBlurring = FALSE;
           ReleaseCapture();
           hMyCursor = hArrowCursor;
          }
          return 0;
    }
    return 0;
  }
 return 0L;
}
```

```
/*************************************************************************
    FUNCTION: SmoothingOperation(HWND, UINT, UINT, LONG)
    PURPOSE: this function is called when mode is changed into smooth mode.
 *************************************************************************/
LONG SmoothingOperation(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
  bGetArea = TRUE;
    GetAreaMode = SMOOTH_MODE;
    hMyCursor = LoadCursor(NULL, IDC_CROSS);
    return 0L;
}

/*************************************************************************
    FUNCTION: SmoothingArea(HWND, RECT)
    PURPOSE: Smooth the image in area indicated by RECT.
 *************************************************************************/
LONG SmoothingArea(HWND hWnd, RECT SmoothArea)
{
  HDC hDC, hMemoryDC;
  int x,y,i,j,level, nWidth, nHeight;
  UINT Count;
  char buffer[20];
  COLORREF Color[26];
  COLORREF NewColor;
  HCURSOR hOldCursor;
  HBITMAP hBitmap, hOldBitmap;

  /* initial Gray Scale color reference */
  for (i=0;i<26;i++) {
    Color[i] = RGB((i*10),
                   (i*10),
                   (i*10));
  }
  hOldCursor = SetCursor(LoadCursor(NULL, IDC_WAIT));
  SetCapture (hWnd);
  hDC = GetDC(hWnd);

  /* find Width and Height of selected region */

  nWidth = SmoothArea.right - SmoothArea.left + 5;
  nHeight = SmoothArea.bottom - SmoothArea.top + 5;

  /* create compatible DC to paste selected region into memory for Smoothing Operation
  */

    hMemoryDC = CreateCompatibleDC(hDC);
    hBitmap = CreateCompatibleBitmap(hDC, nWidth, nHeight);
    hOldBitmap = SelectObject(hMemoryDC, hBitmap);
    BitBlt(hMemoryDC, 0, 0, nWidth, nHeight,
           hDC, SmoothArea.left-2, SmoothArea.top-2, SRCCOPY);

  /* start smoothing operation */

  for(y=2;y<=nHeight-2;y++)
    for(x=2;x<=nWidth-2;x++) {
        Count = 0;
        for (j=-2;j<=2;j++) {
            for (i=-2;i<=2;i++) {
                Count+=(LOBYTE(LOWORD(GetPixel(hMemoryDC,x+i,y+j))));
            }
        }
        level = Count / 255;
        NewColor = Color[level];
        SetPixel(hDC, SmoothArea.left + x - 2, SmoothArea.top + y - 2, NewColor);
    }

  SetCursor (hOldCursor);
  hMyCursor = hArrowCursor;
  ReleaseCapture();
  SelectObject (hMemoryDC, hOldBitmap);
  DeleteObject (hBitmap);
  DeleteDC (hMemoryDC);
  ReleaseDC (hWnd, hDC);
  return 0L;
}

/*************************************************************************
    FUNCTION: BlurringOperation(HWND, UINT, UINT, LONG)
    PURPOSE: this function is called when mode is changed into blur mode.
 *************************************************************************/
LONG BlurringOperation(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
  bGetArea = TRUE;
    GetAreaMode = BLUR_MODE;
    hMyCursor = LoadCursor(NULL, IDC_CROSS);
    return 0L;
}

/*************************************************************************
    FUNCTION: BlurringArea(HWND, LONG)
    PURPOSE: Blur the region around the position that pointed by mouse.
 *************************************************************************/
LONG BlurringArea(HWND hWnd, LONG lParam)
{
    RECT BlurringArea;
    HDC hDC, hMemoryDC;
    int x,y,i,j,level, nWidth, nHeight;
```

```
UINT Count;
char buffer[20];
COLORREF Color[26];
COLORREF NewColor;
HCURSOR hOldCursor;
HBITMAP hBitmap, hOldBitmap;

/* initial Gray Scale color reference */
for (i=0;i<26;i++) {
  Color[i] = RGB((i*10),
                 (i*10),
                 (i*10));
}
hDC = GetDC(hWnd);

/* initial Blurring Area */

BlurringArea.left = LOWORD(lParam) - 6;
BlurringArea.top = HIWORD(lParam) - 6;
BlurringArea.right = BlurringArea.left + 12;
BlurringArea.bottom = BlurringArea.top + 12;

/* find Width and Height of region */

nWidth = nHeight = 13;

// create compatible DC to paste selected region into memory for Blurringing Operation

  hMemoryDC = CreateCompatibleDC(hDC);
  hBitmap = CreateCompatibleBitmap(hDC, nWidth, nHeight);
  hOldBitmap = SelectObject(hMemoryDC, hBitmap);
  BitBlt(hMemoryDC, 0, 0, nWidth, nHeight,
         hDC, BlurringArea.left, BlurringArea.top, SRCCOPY);

/* start blurring operation */

for (y = 2; y<=10 ;y+=4) {
  for (x = 2; x<=10 ; x+=4) {
      Count=0;
      for(j=-2;j<=2;j++){
        for(i=-2;i<=2;i++){
          Count+=(LOBYTE(LOWORD(DWORD)
                 GetPixel(hMemoryDC,x+i,y+j)));
        }
      }
      level = Count/255;
      NewColor = Color[level];
      for(j=-2;j<=2;j++){
        for(i=-2;i<=2;i++){
          SetPixel(hDC, BlurringArea.left+x+i, BlurringArea.top+y+j, NewColor);
        }
      }
  }
}

SelectObject(hMemoryDC, hOldBitmap);
DeleteObject(hBitmap);
DeleteDC(hMemoryDC);
ReleaseDC(hWnd,hDC);
  return 0L;
}

/*****************************************************************************
  FUNCTION: ScalingOperation(HWND, UINT, UINT, LONG)
  PURPOSE: this function is called when the mode is changed into scaling mode to initial
some necessary value for operation.
*****************************************************************************/
LONG ScalingOperation(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
 HDC hDC, hTempMemDC;
 RECT rc;
 BITMAP Bitmap;
 char szTemp[20];
 HBITMAP hOldBmp, hTempBmp;
  int i;

 bGetArea = TRUE;
 GetAreaMode = SCALE_MODE;
  nMoveItem = HAIR;
  i = 0;
  hTempBmp = LoadBitmap(hLibrary[i], MAKEINTRESOURCE(nCurrentStyle));
  GetObject(hTempBmp, 16, (LPSTR) &Bitmap);
  ScaleRect.left = ptPosition[i].x + StartPoint.x;
  ScaleRect.top = ptPosition[i].y + StartPoint.y;
  ScaleRect.right = ScaleRect.left + Bitmap.bmWidth;
  ScaleRect.bottom = ScaleRect.top + Bitmap.bmHeight;
  Display6Square(hPicWnd, ScaleRect);
  DeleteObject(hTempBmp);
  return 0L;
}
/*****************************************************************************
  FUNCTION: CheckScaling(LONG)
  PURPOSE: Check the current mouse position whether it is on working region or not.
*****************************************************************************/
void CheckScaling(LONG lParam)
{
  POINT pt;
  int i;
```

```
    int nResult=0;

    pt=MAKEPOINT(lParam);
    for(i=0;i<8;i++){
        if((pt.x>=ScalingPoint[i].x-4)&&(pt.x<=ScalingPoint[i].x+4)&&
         (pt.y>=ScalingPoint[i].y-4)&&(pt.y<=ScalingPoint[i].y+4)){
            nResult=i+1;
            break;
        }
    }
    switch(nResult){
            case 1:
            case 8:
                    hMyCursor=LoadCursor(NULL,IDC_SIZENWSE);
                    break;
            case 3:
            case 6:
                    hMyCursor=LoadCursor(NULL,IDC_SIZENESW);
                    break;
            case 2:
            case 7:
                    hMyCursor=LoadCursor(NULL,IDC_SIZENS);
                    break;
            case 4:
            case 5:
                    hMyCursor=LoadCursor(NULL,IDC_SIZEWE);
                    break;
            default:
                    hMyCursor=hArrowCursor;
    }
}

/*************************************************************************
    FUNCTION: CheckScalingButtonDown(LONG)
    PURPOSE: Check mouse position and left button that necessary to control the scaling
operation.
*************************************************************************/
int CheckScalingButtonDown(LONG lParam)
{
POINT pt;
int i;
int nResult=0;

    pt=MAKEPOINT(lParam);
    for(i=0;i<8;i++){
        if((pt.x>=ScalingPoint[i].x-4)&&(pt.x<=ScalingPoint[i].x+4)&&
         (pt.y>=ScalingPoint[i].y-4)&&(pt.y<=ScalingPoint[i].y+4)){
            nResult=i+1;
            break;
        }
    }
    return nResult;
}

/*************************************************************************
    FUNCTION: DisplaySelectFrame(HWND)
    PURPOSE: draw rectangle in XOR manner.
*************************************************************************/
void DisplaySelectFrame(HWND hWnd)
{
    HDC hDC;
    int OldROP;

    hDC=GetDC(hWnd);
    OldROP=SetROP2(hDC, R2_NOT);
    MoveTo(hDC, SelectRect.left, SelectRect.top);
    LineTo(hDC, SelectRect.right, SelectRect.top);
    LineTo(hDC, SelectRect.right, SelectRect.bottom);
    LineTo(hDC, SelectRect.left, SelectRect.bottom);
    LineTo(hDC, SelectRect.left, SelectRect.top);
    SetROP2(hDC, OldROP);
    ReleaseDC(hWnd, hDC);
}
```

```
/*****************************************************************************
   PROGRAM: AngSub2.c
        (Angelo subprogram)
   PURPOSE: this file contain the routine that used while testing some operation.
*****************************************************************************/
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************************
   FUNCTION: Person(HWND, UINT, UINT, LONG)
   PURPOSE: Processes messages for "Person" dialog box
   MESSAGES:

        WM_INITDIALOG    - initialize dialog box
        WM_COMMAND       - Input received
   COMMENTS:
        No initialization is needed for this particular dialog box, but TRUE must be
returned to Windows. Wait for user to click on "Ok" button, then close the dialog box.
*****************************************************************************/
BOOL FAR PASCAL _export Person(HWND hDlg, UINT message,
                               UINT wParam, LONG lParam)
{
   switch (message)
   {
        case WM_INITDIALOG:                    /* message: initialize dialog
box */
             return (TRUE);

        case WM_COMMAND:              /* message: received a command */
             if (wParam == IDOK || wParam == IDCANCEL)
             {
                  EndDialog(hDlg, TRUE); /* Exits the dialog box */
                  return (TRUE);
             }
             break;
   }
   return (FALSE);          /* Didn't process a message */
}

/*****************************************************************************
   FUNCTION: RedrawFace(short)
   PURPOSE: repaint the picture window by emit some component.
*****************************************************************************/
short RedrawFace(short nItem)
{
   short   nTemp;
   nTemp=nCurrentStyle[nItem];
   nCurrentStyle[nItem]=-1;
   bRedrawFace=TRUE;
   InvalidateRect(hPicWnd, NULL, TRUE);
   UpdateWindow(hPicWnd);
   return(nTemp);
}

/*****************************************************************************
   FUNCTION: DisplayMainBitmap(HWND)
   PURPOSE: Restore image saved in memory by calling GetMainBitmap function to
display on the screen.
*****************************************************************************/
void DisplayMainBitmap(HWND hWnd)
{
   PAINTSTRUCT ps;
   HDC hDC;
   HDC hMemoryDC;
   BITMAP Bitmap;
   HBITMAP hOldBitmap;

   BeginPaint(hWnd, &ps);
   hDC = ps.hdc;
   hMemoryDC = CreateCompatibleDC(hDC);
   hOldBitmap = SelectObject(hMemoryDC, hMainBitmap);
   GetObject(hMainBitmap, sizeof(BITMAP), (LPSTR) &Bitmap);
    BitBlt(hDC, StartPoint.x - nHscrollPos + 50, StartPoint.y - nVscrollPos + 50,
          Bitmap.bmWidth, Bitmap.bmHeight,
          hMemoryDC, 0, 0, SRCCOPY);
   SelectObject(hMemoryDC, hOldBitmap);
   DeleteDC(hMemoryDC);
    EndPaint(hWnd, &ps);
}

/*****************************************************************************
   FUNCTION: GetMainBitmap(HWND)
   PURPOSE: Save current face image temporary in memory.
*****************************************************************************/
void GetMainBitmap(HWND hWnd)
{
   HDC      hDC;
   HDC      hMemoryDC;
   HBITMAP  hBitmap;
   HBITMAP  hOldBitmap;
   POINT    GetPoint, SavePoint, SaveSize;
   RECT     ClientRect;
```

```
    hDC=GetDC(hWnd);
    hMemoryDC=CreateCompatibleDC(hDC);
    hOldBitmap=SelectObject(hMemoryDC,hMainBitmap);
    GetClientRect(hWnd,&ClientRect);
    GetPoint.x=max(0,StartPoint.x-nHscrollPos+50);
    GetPoint.y=max(0,StartPoint.y-nVscrollPos+50);
    SavePoint.x=max(0,-(StartPoint.x-nHscrollPos+50));
    SavePoint.y=max(0,-(StartPoint.y-nVscrollPos+50));
    SaveSize.x=min(ClientRect.right, PictureSize.x);
    SaveSize.y=min(ClientRect.bottom, PictureSize.y);
    BitBlt(hMemoryDC, SavePoint.x, SavePoint.y, SaveSize.x, SaveSize.y,
                hDC, GetPoint.x, GetPoint.y, SRCCOPY);
    SelectObject(hMemoryDC,hOldBitmap);
    DeleteDC(hMemoryDC);
    ReleaseDC(hWnd,hDC);
}
```

```
/*****************************************************************************
   PROGRAM: AngSub3.c (Angelo subprogram)
   PURPOSE: this file contain the routine that used while testing some operation.

*****************************************************************************/
#include <windows.h>
#include <stdio.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************************
   FUNCTION: Display8Square (HDC, RECT)
   PURPOSE: Display the points to indicate select region.
*****************************************************************************/
void Display8Square(HDC hDC, RECT Rect)
{
 HDC hTempMemDC;
 HBITMAP hTempBmp, hOldBmp;
 BITMAP Bitmap;

 hTempBmp = LoadBitmap(hInst, "square");
 hTempMemDC = CreateCompatibleDC(hDC);
 hOldBmp = SelectObject(hTempMemDC, hTempBmp);
 GetObject(hTempBmp, 16, (LPSTR) &Bitmap);
 BitBlt(hDC, Rect.left-6, Rect.top-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
 BitBlt(hDC, Rect.left-6, ((Rect.top+Rect.bottom)/2)-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
 BitBlt(hDC, Rect.left-6, Rect.bottom-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
  BitBlt(hDC, ((Rect.left+Rect.right)/2)-6, Rect.top-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
 BitBlt(hDC, Rect.right-6, Rect.top-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
 BitBlt(hDC, Rect.right-6, ((Rect.top+Rect.bottom)/2)-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
  BitBlt(hDC, Rect.right-6, Rect.bottom-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
  BitBlt(hDC, ((Rect.left+Rect.right)/2)-6, Rect.bottom-6,
                    Bitmap.bmWidth, Bitmap.bmHeight,
                    hTempMemDC, 0, 0, SRCXNOR);
 SelectObject(hTempMemDC, hOldBmp);
 DeleteDC(hTempMemDC);
 DeleteObject(hTempBmp);
 ScalingPoint[0].x = ScalingPoint[4].x = ScalingPoint[6].x = Rect.left;
 ScalingPoint[2].x = ScalingPoint[4].x = ScalingPoint[7].x = Rect.right;
 ScalingPoint[0].y = ScalingPoint[1].y = ScalingPoint[2].y = Rect.top;
 ScalingPoint[5].y = ScalingPoint[6].y = ScalingPoint[7].y = Rect.bottom;
 ScalingPoint[1].x = ScalingPoint[6].x = (Rect.left+Rect.right)/2;
 ScalingPoint[3].y = ScalingPoint[4].y = (Rect.top+Rect.bottom)/2;
}

/*****************************************************************************
   FUNCTION: MoveOperation (HWND, UINT, UINT, LONG)
   PURPOSE: this routine handle using the mouse to control position of each component.
*****************************************************************************/
LONG MoveOperation(HWND hWnd, UINT message, UINT wParam, LONG lParam)
{
   RECT Rect;
   HDC hDC, hTempMemDC;
   HBITMAP hBitmap, hOldBmp;
   BITMAP Bitmap;
   POINT pt;
   int nXDif, nYDif;
   int nResource;
   char szTemp[20];
   hDC = GetDC(hPicWnd);
   switch(message) {
        case WM_COMMAND:
            GetAreaMode = MOVE_MODE;
            bGetArea = TRUE;
            hMyCursor = LoadCursor(NULL, IDC_SIZE);
            switch(wParam) {
                case IDM_MHAIR:
                    nMoveItem = HAIR;
                    break;
                case IDM_MCHIN:
                    nMoveItem = CHIN;
                    break;
                case IDM_MLEAR:
                    nMoveItem = LEAR;
                    break;
                case IDM_MREAR:
                    nMoveItem = REAR;
                    break;
                case IDM_MLEBROW:
                    nMoveItem = LEBROW;
                    break;
                case IDM_MREBROW:
                    nMoveItem = REBROW;
```

```
                break;
            case IDM_MLEYE:
                nMoveItem=LEYE;
                break;
            case IDM_MREYE:
                nMoveItem=REYE;
                break;
            case IDM_MNOSE:
                nMoveItem=NOSE;
                break;
            case IDM_MMOUTH:
                nMoveItem=MOUTH;
                break;
        }
        nResource=nCurrentStyle[nMoveItem];
        if((nMoveItem==3)||(nMoveItem==5)||(nMoveItem==7))
            nResource+=5000;
        hBitmap=LoadBitmap(hLibrary[nMoveItem],
                    MAKEINTRESOURCE(nResource));
        hTempMemDC=CreateCompatibleDC(hDC);
    hOldBmp=SelectObject(hTempMemDC,hBitmap);
    GetObject(hBitmap,16,(LPSTR)&Bitmap);
    nCurrentStyle[nMoveItem]=RedrawFace[nMoveItem];
    MovingPoint.x=ptPosition[nMoveItem].x+StartPoint.x+nHscrollPos;
    MovingPoint.y=ptPosition[nMoveItem].y+StartPoint.y+nVscrollPos;
    BitBlt(hDC,MovingPoint.x,MovingPoint.y,
        Bitmap.bmWidth,Bitmap.bmHeight,
        hTempMemDC,0,0,SRCXNOR);
    SelectObject(hTempMemDC,hOldBmp);
    DeleteDC(hTempMemDC);
    DeleteObject(hBitmap);
    pt=MovingPoint;
    ClientToScreen(hPicWnd,&pt);
    SetCursorPos(pt.x+(Bitmap.bmWidth/2),pt.y+(Bitmap.bmHeight/2));
    break;

case WM_MOUSEMOVE:
    pt=MAKEPOINT(lParam);
    nResource=nCurrentStyle[nMoveItem];
    if((nMoveItem==3)||(nMoveItem==5)||(nMoveItem==7))
        nResource+=5000;
    hBitmap=LoadBitmap(hLibrary[nMoveItem],
                MAKEINTRESOURCE(nResource));
    hTempMemDC=CreateCompatibleDC(hDC);
    hOldBmp=SelectObject(hTempMemDC,hBitmap);
    GetObject(hBitmap,16,(LPSTR)&Bitmap);
    pt.x-=(Bitmap.bmWidth/2);
    pt.y-=(Bitmap.bmHeight/2);
    BitBlt(hDC,MovingPoint.x,MovingPoint.y,
        Bitmap.bmWidth,Bitmap.bmHeight,
        hTempMemDC,0,0,SRCXNOR);
    BitBlt(hDC,pt.x,pt.y,
        Bitmap.bmWidth,Bitmap.bmHeight,
        hTempMemDC,0,0,SRCXNOR);
    MovingPoint=pt;
    sprintf(szTemp,"(%i,%i)",pt.x,pt.y);
    TextOut(hDC,10,10,szTemp,sizeof(szTemp));
    SelectObject(hTempMemDC,hOldBmp);
    DeleteDC(hTempMemDC);
    DeleteObject(hBitmap);
    break;
case WM_LBUTTONDOWN:
    hMyCursor=hArrowCursor;
    bGetArea=FALSE;
    nResource=nCurrentStyle[nMoveItem];
    if((nMoveItem==3)||(nMoveItem==5)||(nMoveItem==7))nResource+=5000;
    hBitmap=LoadBitmap(hLibrary[nMoveItem],MAKEINTRESOURCE(nResource));
    hTempMemDC=CreateCompatibleDC(hDC);
    hOldBmp=SelectObject(hTempMemDC,hBitmap);
    GetObject(hBitmap,16,(LPSTR)&Bitmap);
    BitBlt(hDC,MovingPoint.x,MovingPoint.y,
        Bitmap.bmWidth,Bitmap.bmHeight,
        hTempMemDC,0,0,SRCAND);
    MovingPoint.x-=nHscrollPos+StartPoint.x;
    MovingPoint.y-=nVscrollPos+StartPoint.y;
    ptPosition[nMoveItem]=MovingPoint;
    sprintf(szTemp,"(%i,%i)",pt.x,pt.y);
    TextOut(hDC,10,10,szTemp,sizeof(szTemp));
    SelectObject(hTempMemDC,hOldBmp);
    DeleteDC(hTempMemDC);
    DeleteObject(hBitmap);

        break;
    }
//  DisplaySquare(hDC,Rect);
    ReleaseDC(hPicWnd,hDC);
    return0L;
}
```

```
/*****************************************************************************
    PROGRAM: SetScale.c (Angelo subprogram)
    PURPOSE: this file contain the routine that used to set cordinate of point that used in
edit operation.
*****************************************************************************/
#include <windows.h>
#include "define.h"
#include "globalex.h"
#include "function.h"

/*****************************************************************************
    FUNCTION: SetScalingPoint (int)
    PURPOSE: Set cordinate of point that is boudary of working item and display boudary
point.
*****************************************************************************/
int SetScalingPoint(int nComponent)
{
    HDC         hDC;
    RECT        Rect;
    int         nResource;
    BITMAP      Bitmap;
    HBITMAP     hBitmap,hTempBmp,hOldBmp;
    HDC     hTempMemDC;

    if((nComponent<0)||(nComponent>9))
            return -1;
    hDC = GetDC(hPicWnd);
    nResource = nCurrentStyle[nComponent];
    if((nComponent==3)||(nComponent==5)||(nComponent==7))  nResource += 5000;
    hBitmap = LoadBitmap(hLibrary[nComponent],MAKEINTRESOURCE(nResource));
    GetObject( hBitmap, 16, (LPSTR) &Bitmap);
    Rect.left = ptPosition[nComponent].x + StartPoint.x - nHscrollPos;
    Rect.top = ptPosition[nComponent].y + StartPoint.y - nVscrollPos;
    Rect.right = Rect.left + Bitmap.bmWidth;
    Rect.bottom = Rect.top + Bitmap.bmHeight;
    if(!bStretchComponent[nComponent]) {
            ptSize[nComponent].x = Bitmap.bmWidth;
            ptSize[nComponent].y = Bitmap.bmHeight;
            bStretchComponent[nComponent] = TRUE;
    }
    if((nComponent==LEAR)||(nComponent==LEBROW)||(nComponent==LEYE)) {
            nResource = nCurrentStyle[nComponent+1] + 5000;
            hBitmap = LoadBitmap(hLibrary[nComponent],
                                    MAKEINTRESOURCE(nResource));
            GetObject( hBitmap, 16, (LPSTR) &Bitmap);
            Rect.right = ptPosition[nComponent+1].x + StartPoint.x -
                    nHscrollPos + Bitmap.bmWidth;
            Rect.bottom = ptPosition[nComponent+1].y + StartPoint.y -
                    nVscrollPos + Bitmap.bmHeight;
            if(!bStretchComponent[nComponent+1]) {
                ptSize[nComponent+1].x = Bitmap.bmWidth;
                ptSize[nComponent+1].y = Bitmap.bmHeight;
                bStretchComponent[nComponent+1] = TRUE;
            }
    }
    DeleteObject(hBitmap);

    hTempBmp = LoadBitmap(hInst, "square");
    hTempMemDC = CreateCompatibleDC(hDC);
    hOldBmp = SelectObject(hTempMemDC, hTempBmp);
    GetObject(hTempBmp, 16, (LPSTR) &Bitmap);
    BitBlt( hDC, Rect.left-6, Rect.top-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt( hDC, Rect.left-6, ((Rect.top+Rect.bottom)/2)-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt( hDC, Rect.left-6, Rect.bottom-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt( hDC, ((Rect.left+Rect.right)/2)-6, Rect.top-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt(hDC, Rect.right-6, Rect.top-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt( hDC, Rect.right-6, ((Rect.top+Rect.bottom)/2)-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt( hDC, Rect.right-6, Rect.bottom-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    BitBlt( hDC, ((Rect.left+Rect.right)/2)-6, Rect.bottom-6,
            Bitmap.bmWidth, Bitmap.bmHeight,
            hTempMemDC, 0, 0, SRCXNOR);
    SelectObject(hTempMemDC, hOldBmp);
    DeleteDC(hTempMemDC);
    DeleteObject(hTempBmp);
    ScalingPoint[0].x = ScalingPoint[4].x = ScalingPoint[6].x = Rect.left;
    ScalingPoint[2].x = ScalingPoint[4].x = ScalingPoint[7].x = Rect.right;
    ScalingPoint[0].y = ScalingPoint[1].y = ScalingPoint[2].y = Rect.top;
    ScalingPoint[5].y = ScalingPoint[6].y = ScalingPoint[7].y = Rect.bottom;
    ScalingPoint[1].x = ScalingPoint[6].x = (Rect.left+Rect.right)/2;
    ScalingPoint[3].y = ScalingPoint[4].y = (Rect.top+Rect.bottom)/2;
    ReleaseDC(hPicWnd,hDC);
    return 0;
}
```

```
/*****************************************************************************
   PROGRAM:Define.h
*****************************************************************************/

#include   "menu.h"
#define    SRCNOR      0x00990066L
#define    BUTTON_RELEASE 0
#define    BUTTON_PRESS 1

#define    SMOOTH_MODE 1000
#define    SCALE_MODE 1001
#define    BLUR_MODE 1002
#define    MOVE_MODE 1003

#define    DRAWBK      0
#define    DRAWFRAME   1
#define    DRAWBOTH    2

/* file menu items */

#define    HAIR     0
#define    CHIN     1
#define    LEAR     2
#define    REAR     3
#define    LEBROW   4
#define    REBROW   5
#define    LEYE     6
#define    REYE     7
#define    NOSE     8
#define    MOUTH    9

#define    LEFT     1
#define    RIGHT    2
#define    UP       3
#define    DOWN     4

#define    PIC_SELECT   0
#define    PIC_CANCEL   1
#define    PIC_MOVE     2
#define    PIC_SIZE     3
#define    PIC_DELETE   4
#define    PIC_REDRAW   5
#define    PIC_XOR      6
#define    PIC_GAP      7

#define    PK_TAB       10000
#define    PK_ENTER     10001
#define    PK_ESCAPE    10002

#define    UNTITLED     "(untitled)"
```

```
/*****************************************************************************
   PROGRAM:function.h
*****************************************************************************/

int PASCAL WinMain(HANDLE, HANDLE, LPSTR, int);
BOOL InitApplication(HANDLE);
BOOL InitInstance(HANDLE, int);
long FAR PASCAL _export MainWndProc(HWND, UINT, UINT, LONG);
BOOL FAR PASCAL _export AboutDlgProc(HWND, UINT, UINT, LONG);
BOOL FAR PASCAL _export Person(HWND, UINT, UINT, LONG);
void App_UnRegisterClasses(void);
LONG FAR PASCAL _export PkRadioProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export PkOwnerDrawRadioProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export PictureProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export PictureSelectFrameProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export ComponentDlgProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export ToolOperateDlgProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export SelectComponentProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export PictureSelectProc(HWND, UINT, UINT, LONG);
LONG FAR PASCAL _export EditStyleProc(HWND, UINT, UINT, LONG);
void CalculateNewSize(void);
void MoveSizeRedraw(HWND);
void Send2SizeMove(int, int, int);
void CheckScaling(LONG);
void  DrawHelpBox(HDC, int, int, int, int);
void  InitialData(void);
int CheckScalingButtonDown(LONG);
void DisplayScalingPoint(HDC, int, int, int);
void DrawScreen(HWND);
void DrawPicture(HWND);
HBITMAP DrawBitmap(HDC, int, int, WORD, DWORD);
void DrawButton(HDC, int, int, int, int, int);
LONG PictureRedraw(HWND, UINT, UINT, LONG);
LONG PictureDelete(HWND, UINT, UINT, LONG);
LONG PictureSelect(HWND, UINT, UINT, LONG);
LONG PictureXnor(HWND, UINT, UINT, LONG);
LONG PictureCancel(HWND, UINT, UINT, LONG);
LONG PictureMove(HWND);
LONG PictureSizeOp(HWND);
LONG PictureGap(HWND);
LONG WM_PAINTMsg(HWND, UINT, UINT, LONG);
LONG WM_LBUTTONDOWNMsg(HWND, UINT, UINT, LONG);
LONG MoveOperation(HWND, UINT, UINT, LONG);
LONG ScalingOperation(HWND, UINT, UINT, LONG);
LONG SmoothingOperation(HWND, UINT, UINT, LONG);
LONG SmoothingArea(HWND, RECT);
LONG BlurringOperation(HWND, UINT, UINT, LONG);
LONG BlurringArea(HWND, LONG);
void DrawSelectPicture(HWND);
void GetMainBitmap(HWND);
```

```
void Display8Square(HDC, RECT);
LONG GetAreaOperation(HWND, UINT, UINT, LONG);
short RedrawFace(short);
void DisplayBitmap(HWND);
void DisplayMainBitmap(HWND);
HDC GetPrinterDC(void);
BOOL PrintMyPage(HWND hwnd);
void DisplaySelectFrame(HWND);
int SetScalingPoint(int);
void DrawFrame(HDC, int, int, int, int, int);
POINT GetComponentPos(int, int);
char *String(WORD);
    //Functions in POPFILE.C

void PopFileInitialize(HWND);
BOOL PopFileOpenDlg  (HWND, LPSTR, LPSTR);
BOOL PopFileSaveDlg  (HWND, LPSTR, LPSTR);
BOOL FileRead   (LPSTR);
BOOL FileWrite    (LPSTR);
void DoCaption(HWND, char *);
void OkMessage(HWND, char *, char *);
short AskAboutSave(HWND, char *);
```

```
/*****************************************************************************
    PROGRAM: Global.h
*****************************************************************************/
/* handle for component data link library */
HANDLE   hLibrary[10];
BOOL     bGetArea=FALSE;
POINT    ScalingPoint[8];
POINT    MovingPoint;
HCURSOR  hArrowCursor;
RECT     ScaleRect;
RECT     SelectRect;
FARPROC  lpfnEditStyleProc;
FARPROC  lpfnOldEdit[10];
POINT    StartPoint;
POINT    PictureSize={320,420};
POINT    ptPosition[10];
POINT    ptSize[10]={{0,0},{0,0},{0,0},{0,0},{0,0},{0,0},{0,0},{0,0},{0,0},{0,0}};
int      nMoveItem;
BOOL     bMove=FALSE;
BOOL     bMoveWindow=FALSE;
BOOL     bBlurring=FALSE;
BOOL     bRedrawFace=TRUE;
int      nScalingMode=0;
HBITMAP hMainBitmap;
int      nXPos, nYPos;
int      nOldXPos;
int      nOldYPos;
int      nXDisplaySize;
int      nYDisplaySize;
int      GetAreaMode;
int      nVscrollPos=50;
int      nHscrollPos=50;
int      COMPONENT;
/* operation mode is used to indicate WM_PAINT operation of PictureProc */
int      nOperationMode=5;
BOOL bStretchComponent[10]={FALSE,FALSE,FALSE,FALSE,FALSE,
                            FALSE,FALSE,FALSE,FALSE,FALSE};
static char szAppName[] = "Angelo";
static char szPicClass[] = "PictureClass";
static char szToolDlgClass[] = "ToolDlgClass";
static char szToolOpDlgClass[] = "ToolOpDlgClass";
static char szPkRadioButtonClass[] = "PkRadioBtn";
static char szPkOwnerDrawRadioButtonClass[] = "PkOwnerDrawRadioBtn";
static char szPicSelectFrameClass[] = "PictureSelectFrameClass";
static char szPicSelectClass[] = "PictureSelectClass";
static char szSelectComponent[] = "SelectComponentClass";
HANDLE hInst;            /* current instance */
HWND     MainHWnd;
HWND     hToolDlg1,hToolDlg2,hSelectBtn,hEditBtn;
```

```
HWND    hPicWnd,hPicSelectFrame,hPicSelectWnd;
HWND    hSelectHair,
        hSelectChin,
        hSelectEar,
        hSelectEye,
        hSelectEbrow,
        hSelectNose,
        hSelectMouth;
HCURSOR hMyCursor;
static char  *ComponentCaption[10] = { "Hair Style\0",
                    "Face Outline\0",
                    "Ears\0",
                    "Ears\0",
                    "Eyebrows\0",
                    "Eyebrows\0",
                    "Eyes\0",
                    "Eyes\0",
                    "Nose\0",
                    "Mouth\0" };

HWND    hSelectComponent[10];
HWND    hOkSlc[10],
        hCancelSlc[10],
        hPrevSlc[10],
        hNextSlc[10],
        hStyleNo[10],
        hSample1[10],
        hSample2[10],
        hSample3[10],
        hSample4[10];
int     nStyleNo;               // Save Style before PIC_SELECT mode
int     nOldStyle;              // Save Old style while PIC_SELECT mode
POINT   ptOldSize[2];
POINT   ptOldPos[2];
int     nCurrentStyle[10];      // Style No. of each component
int     nMaxStyle[10];          // Maximum Style of each component
int     nEditMode;
BOOL    bValidComponent;
BOOL    bNoCompSelect;
int     nDirection;
int     nPointResize;
int     nMoveSizeStep = 1;
struct  Angelo  {
            int     nStyle;
            POINT   ptPos;
            POINT   ptSiz;
            };

struct  Angelo *ptrAngelo;
```

```
/*****************************************************************************
   PROGRAM: Globelex.h
*****************************************************************************/

extern HANDLE   hLibrary[10];
extern BOOL     bGetArea;
extern POINT    ScalingPoint[8];
extern POINT    MovingPoint;
extern HCURSOR  hArrowCursor;
extern FARPROC  lpfnEditStyleProc;
extern FARPROC  lpfnOldEdit[10];
extern RECT     ScaleRect;
extern RECT     SelectRect;
extern int      nCurrentStyle[10];
extern int      nMaxStyle[10];
extern int      nOldStyle;
extern int      nStyleNo;
extern POINT    ptPosition[10];
extern POINT    ptSize[10];
extern POINT    ptOldPos[2];
extern POINT    ptOldSize[2];
extern char     *ComponentCaption[];
extern HWND     hOkSlc[10],
                hCancelSlc[10],
                hPrevSlc[10],
                hNextSlc[10],
                hStyleNo[10],
                hSample1[10],
                hSample2[10],
                hSample3[10],
                hSample4[10];
extern int      nMoveItem;
extern BOOL     bMove;
extern BOOL     bBlurring;
extern BOOL     bRedrawFace;
extern int      nScalingMode;
extern HBITMAP  hMainBitmap;
extern int      nXDisplaySize;
extern int      nYDisplaySize;
extern int      GetAreaMode;
extern int      COMPONENT;
extern int      nOperationMode;
extern BOOL     bStretchComponent[10];
extern HANDLE   hInst;      /* current instance */
extern HWND     hToolDlg1,hToolDlg2,hSelectBtn,hEditBtn;
extern HWND     hPicWnd,hPicSelectFrame,hPicSelectWnd;
extern HCURSOR hSizeArrow;
extern HCURSOR hSaveCursor;
extern HCURSOR hMyCursor;
extern HWND hSelectComponent[10];
```

```
extern HWND   MainHWnd;
extern int    nVscrollPos;
extern int    nHscrollPos;
extern POINT  StartPoint;
extern POINT  PictureSize;
extern HWND   hSelectHair,
              hSelectChin,
              hSelectEar,
              hSelectEye,
              hSelectEbrow,
              hSelectNose,
              hSelectMouth;
extern int    nDirection;
extern int    nPointResize;
extern int    nEditMode;
extern BOOL   bValidComponent;
extern int    nMoveSizeStep;
extern BOOL   bNoCompSelect;
extern struct Angelo  {
                      int nStyle;
                      POINT ptPos;
                      POINT ptSiz;
                      };
extern struct Angelo *ptrAngelo;
```

```
/*****************************************************************************
   PROGRAM:Menu.h
*****************************************************************************/

/* File menu items */
#define   IDM_NEW      101
#define   IDM_OPEN     102
#define   IDM_SAVE     103
#define   IDM_SAVEAS   104
#define   IDM_PRINT    105
#define   IDM_PRINTSET 106
#define   IDM_EXIT     107

/* Edit menu items */
#define   IDM_UNDO       201
#define   IDM_MOVE       202
#define   IDM_MHAIR      2111
#define   IDM_MCHIN      2112
#define   IDM_MLEAR      2113
#define   IDM_MREAR      2114
#define   IDM_MLEBROW    2115
#define   IDM_MREBROW    2116
#define   IDM_MLEYE      2117
#define   IDM_MREYE      2118
#define   IDM_MNOSE      2119
#define   IDM_MMOUTH     2120
#define   IDM_PARTEDIT   203
#define   IDM_CLEAR      204
#define   IDM_REFRESH    205
#define   IDM_BLUR       206
#define   IDM_SCALE      207
#define   IDM_PERSON     208
#define   IDM_SMOOTH     209

/* Component menu item */
#define   IDM_HAIR       301
#define   IDM_CHIN       302
#define   IDM_EAR        303
#define   IDM_EBROW      304
#define   IDM_EYE        305
#define   IDM_NOSE       306
#define   IDM_MOUTH      307

/* Help menu item */
#define   IDM_INDEX      401
#define   IDM_USEHELP    402
#define   IDM_PROC       403
#define   IDM_ABOUT      404
```

```
/* operation mode */
#define IDC_NEXT 105
#define IDC_PREV 106
#define IDC_STYLE 107
#define IDC_HAIR 111
#define IDC_CHIN 112
#define IDC_EAR 113
#define IDC_EBROW 114
#define IDC_EYE 115
#define IDC_NOSE 116
#define IDC_MOUTH 117
#define IDC_SELECT 118
#define SELECT 1
#define IDC_EDIT 119
#define EDIT 2

/* direction mode */
#define IDC_DIRECT 210
#define IDC_ARRLEFT 211
#define IDC_ARRRIGHT 212
#define IDC_ARRUP 213
#define IDC_ARRDOWN 214
#define IDC_PICMOVE 215
#define IDC_PICSIZE 216
#define IDC_LEFT 217
#define IDC_RIGHT 218
#define IDC_CENTRE 219
#define IDC_TOP 220
#define IDC_BOTTOM 221

/* button */
#define IDC_OK 10
#define IDC_CANCEL 20
#define RADIOUP 1001
#define RADIODOWN 1002
#define RADIOSELECT 1003

/* sample */
#define IDC_SAM1 301
#define IDC_SAM2 302
#define IDC_SAM3 303
#define IDC_SAM4 304
```

```
/*****************************************************************************
    PROGRAM: Angelo.rc
*****************************************************************************/

#include <windows.h>
#include "person.h"
#include "person.dlg"
#include "menu.h"

MyIconICON bitmap\angelo.ico

//ANGELO LOGO
1100 BITMAP bitmap\angelo.bmp
1101 BITMAP bitmap\sparkle.bmp

1001 BITMAP bitmap\radioup.bmp
1002 BITMAP bitmap\radiodn.bmp
1003 BITMAP bitmap\radiosle.bmp

//COMPONENT Dialog BITMAP
//UP button
1105 BITMAP bitmap\nextup.bmp
1106 BITMAP bitmap\prevup.bmp

1111 BITMAP bitmap\hairup.bmp
1112 BITMAP bitmap\chinup.bmp
1113 BITMAP bitmap\earup.bmp
1114 BITMAP bitmap\ebrowup.bmp
1115 BITMAP bitmap\eyeup.bmp
1116 BITMAP bitmap\noseup.bmp
1117 BITMAP bitmap\mouthup.bmp
1118 BITMAP bitmap\slctup.bmp
1119 BITMAP bitmap\editup.bmp

//DOWN button
2105 BITMAP bitmap\nextdown.bmp
2106 BITMAP bitmap\prevdown.bmp

2111 BITMAP bitmap\hairdn.bmp
2112 BITMAP bitmap\chindn.bmp
2113 BITMAP bitmap\eardn.bmp
2114 BITMAP bitmap\ebrowdn.bmp
2115 BITMAP bitmap\eyedn.bmp
2116 BITMAP bitmap\nosedn.bmp
2117 BITMAP bitmap\mouthdn.bmp
2118 BITMAP bitmap\slctdn.bmp
2119 BITMAP bitmap\editdn.bmp

//UP & SELECTED button
3111 BITMAP bitmap\hairdn1.bmp
```

```
3112 BITMAP bitmap\chindn1.bmp
3113 BITMAP bitmap\eardn1.bmp
3114 BITMAP bitmap\ebrowdn1.bmp
3115 BITMAP bitmap\eyedn1.bmp
3116 BITMAP bitmap\nosedn1.bmp
3117 BITMAP bitmap\mouthdn1.bmp
3118 BITMAP bitmap\sktdn1.bmp
3119 BITMAP bitmap\edftdn1.bmp

//EDIT Manager Dialog BITMAP
1210 BITMAP bitmap\direct.bmp
1211 BITMAP bitmap\left.bmp
1212 BITMAP bitmap\right.bmp
1213 BITMAP bitmap\up.bmp
1214 BITMAP bitmap\down.bmp
1215 BITMAP bitmap\tooltxt1.bmp
1216 BITMAP bitmap\tooltxt2.bmp
1217 BITMAP bitmap\tooltxt3.bmp
1218 BITMAP bitmap\tooltxt4.bmp
1219 BITMAP bitmap\tooltxt5.bmp
1220 BITMAP bitmap\tooltxt6.bmp
1221 BITMAP bitmap\tooltxt7.bmp

//Sample BITMAP of each component
1300 BITMAP bitmap\samxor.bmp
1301 BITMAP bitmap\samhair1.bmp
1302 BITMAP bitmap\samhair2.bmp
1303 BITMAP bitmap\samhair3.bmp
1304 BITMAP bitmap\samhair4.bmp

1305 BITMAP bitmap\samchin1.bmp
1306 BITMAP bitmap\samchin2.bmp
1307 BITMAP bitmap\samchin3.bmp
1308 BITMAP bitmap\samchin4.bmp

1309 BITMAP bitmap\samear1.bmp
1310 BITMAP bitmap\samear2.bmp
1311 BITMAP bitmap\samear3.bmp
1312 BITMAP bitmap\samear4.bmp

1317 BITMAP bitmap\samerow1.bmp
1318 BITMAP bitmap\samerow2.bmp
1319 BITMAP bitmap\samerow3.bmp
1320 BITMAP bitmap\samerow4.bmp

1325 BITMAP bitmap\sameye1.bmp
1326 BITMAP bitmap\sameye2.bmp
1327 BITMAP bitmap\sameye3.bmp
1328 BITMAP bitmap\sameye4.bmp

1333 BITMAP bitmap\samnose1.bmp
1334 BITMAP bitmap\samnose2.bmp
1335 BITMAP bitmap\samnose3.bmp
1336 BITMAP bitmap\samnose4.bmp

1337 BITMAP bitmap\sammout1.bmp
1338 BITMAP bitmap\sammout2.bmp
1339 BITMAP bitmap\sammout3.bmp
1340 BITMAP bitmap\sammout4.bmp

//Brush BITMAP
5000 BITMAP bitmap\brush1.bmp

//SYSTEM button-state UP
7010 BITMAP bitmap\ok1.bmp
7020 BITMAP bitmap\cancel1.bmp

//SYSTEM button-state DOWN
8010 BITMAP bitmap\ok2.bmp
8020 BITMAP bitmap\cancel2.bmp

//SYSTEM button-state UP & SELECTED
9010 BITMAP bitmap\ok3.bmp
9020 BITMAP bitmap\cancel3.bmp

//Angelo CURSOR
HAND CURSOR bitmap\hand.cur
101 CURSOR bitmap\wait1.cur
102 CURSOR bitmap\wait2.cur
103 CURSOR bitmap\wait3.cur
104 CURSOR bitmap\wait4.cur
105 CURSOR bitmap\wait5.cur
106 CURSOR bitmap\wait6.cur
107 CURSOR bitmap\wait7.cur
108 CURSOR bitmap\wait8.cur

square BITMAP bitmap\square.bmp

STRINGTABLE
  {
    /* xposition */
        1,      "27"
        2,      "27"
        3,      "27"
        4,      "27"
        5,      "27"
        6,      "27"
        7,      "27"
        8,      "27"
```

1001, "55"
1002, "55"
1003, "55"
1004, "55"
1005, "55"
1006, "55"
1007, "55"
1008, "55"
1009, "55"
1010, "55"
1011, "55"
1012, "55"
1013, "55"
1014, "55"
1015, "55"

2001, "17"
2002, "17"
2003, "17"
2004, "17"
2005, "17"
2006, "17"
2007, "17"
2008, "17"
2009, "17"
2010, "17"
2011, "17"
2012, "17"
2013, "17"
2014, "17"
2015, "17"

3001, "323"
3002, "323"
3003, "323"
3004, "323"
3005, "323"
3006, "323"
3007, "323"
3008, "323"
3009, "323"
3010, "323"
3011, "323"
3012, "323"
3013, "323"
3014, "323"
3015, "323"

4001, "79"
4002, "79"
4003, "79"
4004, "79"
4005, "79"
4006, "79"
4007, "79"
4008, "79"
4009, "79"
4010, "79"
4011, "79"
4012, "79"

5001, "213"
5002, "213"
5003, "213"
5004, "213"
5005, "213"
5006, "213"
5007, "213"
5008, "213"
5009, "213"
5010, "213"
5011, "213"
5012, "213"

6001, "101"
6002, "101"
6003, "101"
6004, "101"
6005, "101"
6006, "101"
6007, "101"
6008, "101"
6009, "101"
6010, "101"
6011, "101"
6012, "101"
6013, "101"

7001, "207"
7002, "207"
7003, "207"
7004, "207"
7005, "207"
7006, "207"
7007, "207"
7008, "207"
7009, "207"
7010, "207"

```
7011, "207"
7012, "207"
7013, "207"

8001, "151"
8002, "151"
8003, "151"
8004, "151"
8005, "151"
8006, "151"
8007, "151"

9001, "131"
9002, "131"
9003, "131"
9004, "131"
9005, "131"
9006, "131"
9007, "131"
9008, "131"
9009, "131"
9010, "131"

/* yposition */
10001,    "-68"
10002,    "-68"
10003,    "-68"
10004,    "-68"
10005,    "-68"
10006,    "-68"
10007,    "-68"
10008,    "-68"

11001,    "148"
11002,    "148"
11003,    "148"
11004,    "148"
11005,    "148"
11006,    "148"
11007,    "148"
11008,    "148"

12001,    "128"
12002,    "128"
12003,    "128"
12004,    "128"
12005,    "128"
12006,    "128"
12007,    "128"
12008, -  "128"
12009,    "128"
12010,    "128"
12011,    "128"
12012,    "128"
12013,    "128"
12014,    "128"
12015,    "128"

13001,    "130"
13002,    "130"
13003,    "130"
13004,    "130"
13005,    "130"
13006,    "130"
13007,    "130"
13008,    "130"
13009,    "130"
13010,    "130"
13011,    "130"
13012,    "130"
13013,    "130"
13014,    "130"
13015,    "130"

14001,    "120"
14002,    "120"
14003,    "120"
14004,    "120"
14005,    "120"
14006,    "120"
14007,    "120"
14008,    "120"
14009,    "120"
14010,    "120"
14011,    "120"
14012,    "120"

15001,    "118"
15002,    "118"
15003,    "118"
15004,    "118"
15005,    "118"
15006,    "118"
15007,    "118"
15008,    "118"
15009,    "118"
15010,    "118"
15011,    "118"
```

```
15012,  "118"

16001,  "132"
16002,  "132"
16003,  "132"
16004,  "132"
16005,  "132"
16006,  "132"
16007,  "132"
16008,  "132"
16009,  "132"
16010,  "132"
16011,  "132"
16012,  "132"
16013,  "132"

17001,  "132"
17002,  "132"
17003,  "132"
17004,  "132"
17005,  "132"
17006,  "132"
17007,  "132"
17008,  "132"
17009,  "132"
17010,  "132"
17011,  "132"
17012,  "132"
17013,  "132"

18001,  "156"
18002,  "156"
18003,  "156"
18004,  "156"
18005,  "156"
18006,  "156"
18007,  "156"

19001,  "282"
19002,  "282"
19003,  "282"
19004,  "282"
19005,  "282"
19006,  "282"
19007,  "282"
19008,  "282"
19009,  "282"
19010,  "282"
```

```
        /* maximum component style */
        30000, "8"
        30001, "8"
        30002, "15"
        30003,  "15"
        30004,  "12"
        30005,  "12"
        30006,  "13"
        30007,  "13"
        30008,  "7"
        30009,  "10"
}


ANGELOMENU
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New\tF5",        IDM_NEW
        MENUITEM "&Open...",        IDM_OPEN
        MENUITEM "&Save",           IDM_SAVE
        MENUITEM "Save &As...",     IDM_SAVEAS
        MENUITEM "&Print",          IDM_PRINT
        MENUITEM SEPARATOR
        MENUITEM "E&xit",           IDM_EXIT
    END

    POPUP   "&Edit"
    BEGIN
        MENUITEM "&Undo\tAlt+BkSp", IDM_UNDO ,GRAYED
        MENUITEM SEPARATOR
        POPUP    "&Move"
        BEGIN
        MENUITEM "&Hair Style",     IDM_MHAIR
        MENUITEM "&Face Outline",   IDM_MCHIN
        MENUITEM "L Ear",           IDM_MLEAR
        MENUITEM "R Ear",           IDM_MREAR
        MENUITEM "L Eyebrow",       IDM_MLEBROW
        MENUITEM "R Eyebrow",       IDM_MREBROW
        MENUITEM "L Eye",           IDM_MLEYE
        MENUITEM "R Eye",           IDM_MREYE
        MENUITEM "&Nose",           IDM_MNOSE
        MENUITEM "&Mouth",          IDM_MMOUTH
        END
    MENUITEM "&Partial Edit\tF6",   IDM_PARTEDIT,GRAYED
    MENUITEM "&Delete\tDel",        IDM_CLEAR ,GRAYED
    MENUITEM SEPARATOR
    MENUITEM "&Blurring",           IDM_BLUR
    MENUITEM "&Scaling",            IDM_SCALE
```

```
    MENUITEM "Smoothing",                   IDM_SMOOTH
    MENUITEM "P&ersonal File",              IDM_PERSON
    END

    POPUP   "&Component"
    BEGIN
        MENUITEM "&Hair Style",             IDM_HAIR
        MENUITEM "&Face Outline",           IDM_CHIN
        MENUITEM "E&ars",                   IDM_EAR
        MENUITEM "E&yebrows",               IDM_EBROW
        MENUITEM "&Eyes",                   IDM_EYE
        MENUITEM "&Nose",                   IDM_NOSE
        MENUITEM "&Mouth",                  IDM_MOUTH
    END

    POPUP "&Help"
    BEGIN
        MENUITEM "&Index\tF1",              IDM_INDEX
        MENUITEM "&Using help",             IDM_USEHELP
        MENUITEM "&Procedures",             IDM_PROC
        MENUITEM SEPARATOR
        MENUITEM "&About Angelo...",        IDM_ABOUT
    END
END

ABOUTBOXDIALOG LOADONCALL MOVEABLE DISCARDABLE 50, 16, 254, 244
CAPTION "About Angelo"
FONT 8, "Helv"
CLASS "BorDlg"
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
BEGIN
    CONTROL "", 102, "BorShade", 1 | WS_CHILD | WS_VISIBLE, 5, 5, 244, 176
    CONTROL "3", 100, "BorBtn", 32768 | WS_CHILD | WS_VISIBLE, 14, 11, 226, 166
    CONTROL "", -1, "BorShade", 2 | WS_CHILD | WS_VISIBLE, 0, 208, 255, 2
    CONTROL "", 1, "BorBtn", 1 | WS_CHILD | WS_VISIBLE | WS_GROUP |
                WS_TABSTOP, 109, 215, 36, 25
    CONTROL "", -1, "BorShade", 1 | WS_CHILD | WS_VISIBLE, 5, 187, 244, 15
    CTEXT "Computer Assisted Suspect Sketching Outfit", -1, 11, 190, 233, 8
END
```

บรรณานุกรม

Charles Petzold . 1992 . PROGRAMMING WINDOWS 3.1 . (third edition) . Washington : Microsoft Press.

Rafael c. Gonzales and Paul Wintz . 1987 . DIGITAL IMAGE PROCESSING .(second edition) . California, U.S.A. : Addison Wesley Publishing Company.

Microsoft Corporation . 1990 . Microsoft Windows Programmer's Reference . Washington, U.S.A. : Microsoft Press.

Microsoft Corporation . 1990 . Microsoft Windows Software Development Kit Guide to Programming version 3.0 . Wasington, U.S.A. : Microsoft Press.

Peter Norton and Paul L. Yao . 1990 . Peter Norton's Windows 3.0 Power Programming Techniques . New York, U.S.A : Bantan Book Inc.

J.Sheppard . 1976 . Drawing the Male Figure . (1st Printing) . New York, U.S.A. : Watsan-Guptill Pub.

J.Sheppard . 1979 . ANATOMY a Complete Guide for Artists . New York, U.S.A. : Watsan-Guptill Pub.

สุทธิชัย โฆษิตวรรณรัตน์ . ๒๕๓๕ . "คอมพิวเตอร์วาดภาพคนร้าย" . วารสารอีคอนนิวส์ . ๘-๑๔ มิถุนายน ๒๕๓๕ : น. ๖๑-๖๘.

ศิวัฒน์ศิวะบวร . ๒๕๓๕ . "ปฐมบทการเขียนโปรแกรมบน Microsoft Windows" . นิตยสารคอมพิวเตอร์รีวิว . ๙๔ (มิถุนายน ๒๕๓๕) : น. ๑๔๗-๑๕๘