การเขียนโปรแกรมบนระบบ X WINDOW

APPLICATION PROGRAMMING ON X WINDOW SYSTEM

โดย

นางสาวนพวรรณ ศุภวิทยา

นายพันจันทร์ ธนวัฒนเสถียร

อาจารย์ที่ปรึกษา

ดร. บุญธีร์ เครือตราชู

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2537

ปริญญานิพนธ์ปีการศึกษา 2537

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การเขียนโปรแกรมบนระบบ X WINDOW

ผู้จัดทำ

1. นางสาวนพวรรณ ศุภวิทยา
2. นายพันจันทร์ ธนวัฒนเสถียร

.....อาจารย์ที่ปรึกษา

(ดร. บุญธีร์ เครือตราชู)

การเขียนโปรแกรมบนระบบ X WINDOW

นพวรรณ ศุภวิทยา
พันจันทร์ ธนวัฒนเสถียร
ดร. บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา
ปีการศึกษา 2537

บทคัดย่อ

ในการทำงานอย่างมีประสิทธิภาพ จำเป็นต้องมีการกำหนดระบบมาตรฐานเพื่อนำมาใช้กับเครื่องคอมพิวเตอร์ชนิดต่างๆ X Window อาจกล่าวได้ว่าเป็นระบบที่ได้รับการยอมรับว่าเป็นมาตรฐาน เนื่องจากมีสถาปัตยกรรมที่ถูกออกแบบมาเพื่อไม่ให้ขึ้นกับฮาร์ดแวร์ ปริญญานิพนธ์นี้ มุ่งเน้นที่จะศึกษาคุณสมบัติของระบบ X Window โดยใช้วิดเจตเซตที่มีชื่อว่า โมทีฟ เพื่อพัฒนาส่วนที่ติดต่อกับผู้ใช้ รวมทั้งมีการยกกรณีศึกษาในเรื่องของการหาเส้นทางการเดินทางโดยเครื่องบินที่ดีที่สุดมาแสดงผลด้วย หลักการในการเขียนโปรแกรมเป็นแบบเชิงวัตถุ อัลกอริธึมที่ใช้ในการคำนวณหาเส้นทางเป็นอัลกอริธึมของ Dijkstra

APPLICATION PROGRAMMING ON X WINDOW SYSTEM

Noppawan Suppawittaya

Punchan Thanawatsathein

Dr. Boontee Kruatrachue Advisor

1995

Abstract

A standard must be set forth to allow various computers to work efficiently together. X Window has been accepted as the de-facto for cross-platform compatability. It has been implemented with hardware-independent architecture. This thesis aims at investigating the properties of X Window using the Motif widget set to develop the user interface. In addition, a case study looking at airline navigation has been employed. The programming nature is object oriented and Dijkstra's algorithm has been adapted for the application.

สารบัญ

สารบัญรูปภาพ

บทที่ 1
บทนำ

ปัจจุบันได้มีการใช้ระบบปฏิบัติการยูนิกซ์อย่างแพร่หลายมากขึ้น โดยจะพบได้ทั้งในงานที่มีการประมวลผลข้อมูลปริมาณมหาศาล การติดต่อสื่อสารในเครือข่ายอินเตอร์เน็ท (Internet) และงานด้านอื่นๆ อีกทั้งเครื่องคอมพิวเตอร์ที่สามารถแสดงผลด้านกราฟิกได้ เปลี่ยนแปลงวิธีการเป็นแบบให้ผู้ใช้สามารถใช้เมาส์ ในการสั่งให้โปรแกรมทำงาน หรือทำการแก้ไข คัดลอกแฟ้มข้อมูลและงานอื่นๆ ได้โดยไม่ต้องป้อนคำสั่งจากแป้นพิมพ์ นอกจากนี้ยังมีการแบ่งการแสดงผลจอภาพออกเป็นส่วนย่อยๆ เรียกว่า วินโดว์ เพื่อแสดงผลลัพธ์ของโปรแกรมแต่ละตัว

ถึงแม้ว่า ส่วนติดต่อกับผู้ใช้นี้จะอำนวยความสะดวกมากมาย อย่างไรก็ตาม การพัฒนาโปรแกรมภายใต้สภาวะดังกล่าวไม่ใช่เรื่องง่าย ระบบวินโดว์แต่ละระบบมีความแตกต่างกัน และมักจะขึ้นอยู่กับฮาร์ดแวร์และระบบปฏิบัติการมาก

ในระดับของเครื่องเวิร์กสเตชัน ก็จะพบปัญหานี้ได้เช่นกัน เพราะหลายครั้งที่เราพบว่า เครื่องแต่ละยี่ห้อจะมีระบบวินโดว์ของมันเองซึ่งจะไม่เหมือนของตัวอื่น สถานการณ์เช่นนี้ทำให้ผู้พัฒนาโปรแกรมประสบปัญหาเมื่อต้องการเขียนโปรแกรมที่ใช้กับเครื่องเวิร์กสเตชันที่ต่างยี่ห้อกัน

ระบบ X Window เป็นส่วนติดต่อกับผู้ใช้ที่ถูกพัฒนาให้เป็นระบบมาตรฐานบนยูนิกซ์ การพัฒนาโปรแกรมบนระบบ X Window นั้น ทำได้โดย 2 วิธี วิธีแรกเป็นการกำหนดให้มีชั้นของซอฟต์แวร์ชั้นหนึ่ง ชั้นระหว่างโปรแกรมที่เราเขียนขึ้นกับระบบวินโดว์ ซึ่งวิธีนี้ทำได้ยาก เพราะโครงสร้างภายในและลักษณะการทำงานของระบบวินโดว์แต่ละระบบแตกต่างกัน ดังนั้น ประสิทธิภาพจะไม่ดีนัก

วิธีที่สองเป็นการออกแบบระบบวินโดว์ขึ้นมาใหม่ โดยกำหนดให้ใช้ไลบรารีมาตรฐานต่างๆ ซึ่งวิธีการนี้จะช่วยลดความซับซ้อนในการพัฒนาโปรแกรม และยังสามารถนำระบบวินโดว์ใหม่นี้ไปใช้กับเครื่องเวิร์กสเตชันยี่ห้อต่างๆ ได้

ระบบ X Window ถูกพัฒนาขึ้นโดยนายโรเบิร์ต ไชฟเลอร์, นายจิม เก็ตตียส์และทีมที่ MIT กลุ่มพัฒนานี้ได้กำหนดระบบพื้นฐานในการจัดการวินโดว์ การทำงานร่วมกับไลบรารีตัวอื่นๆ และมีการกำหนด X โปรโตคอลขึ้นมาใช้เพื่อรับข้อมูลที่ถูกส่งเข้ามาจากโปรแกรม เพราะลักษณะการทำงานของระบบ X Window เป็นแบบผู้ให้บริการกับลูกข่าย

ในโครงงานนี้ เป็นการศึกษาและใช้งานไลบรารี 3 ตัว ได้แก่ Xlib, XtIntrinsic และโมทีฟ โดยจะเป็นการเน้นถึงการศึกษาโมทีฟไลบรารี ซึ่งเป็นไลบรารีที่อยู่ในระดับบนสุดของไลบรารีทั้ง 3 ตัว โมทีฟเป็นภาษาสำหรับพัฒนาส่วนที่ติดต่อกับผู้ใช้โดยใช้งานร่วมกับภาษาซี เป็นภาษาที่มีประสิทธิภาพสูง เนื่องจากไม่ต้องทำการคอมไพล์โปรแกรมใหม่ทุกครั้งเมื่อมีการเปลี่ยนแปลงหลังจากมีการเรียกข้อมูลขึ้นมาแล้ว

สำหรับการสร้างแอพพลิเคชันโปรแกรม ซึ่งจะใช้เป็นกรณีศึกษานั้น จะเป็นแอพพลิเคชันโปรแกรมที่ช่วยในการวางแผนการเดินทางโดยเครื่องบิน เนื่องจากการเดินทางไปยังสถานที่ต่างๆ มีเส้นทางให้เลือกหลายเส้นทาง ทุกคนย่อมต้องการที่จะหาเส้นทางที่ตนจะเดินทางไปให้ถึงจุดหมายเร็วที่สุด โครงงานนี้จึงเป็นการช่วยผู้ใช้ในการตัดสินใจเลือกเส้นทางที่ดีที่สุดในการเดินทางผ่านเมืองต่างๆ โดยเครื่องบินไปยังจุดหมาย

ในการคำนวณหาเส้นทางการเดินทางที่ดีที่สุดนั้น ผู้ใช้สามารถระบุได้ด้วยว่าตนต้องการจะเดินทางไปยังจุดหมายโดยพิจารณาเลือกจากความต้องการหนึ่งในสาม ดังนี้

ต้องการเดินทางไปยังจุดหมายโดยใช้เส้นทางบินที่มีระยะทางสั้นที่สุดเป็นเกณฑ์

ต้องการเดินทางไปยังจุดหมายโดยใช้เส้นทางบินที่มีราคาถูกที่สุดเป็นเกณฑ์

ต้องการเดินทางไปยังจุดหมายโดยใช้เส้นทางบินที่ใช้เวลาน้อยที่สุดเป็นเกณฑ์

โดยเวลาน้อยที่สุดดังกล่าวนี้ รวมเวลาบินจริงและเวลารอเข้าด้วยกัน

1.1 วัตถุประสงค์

เพื่อศึกษาและทำความเข้าใจเกี่ยวกับระบบ X Window ซึ่งเป็นระบบมาตรฐานบนยูนิกซ์และมีสถาปัตยกรรมที่ไม่ขึ้นอยู่กับฮาร์ดแวร์

ให้สามารถเขียนโปรแกรมใช้งานบนระบบ X Window โดยใช้วิดเจตที่มีชื่อว่าโมทีฟได้

สามารถสร้างแอพพลิเคชันโปรแกรม ซึ่งใช้เป็นกรณีศึกษาที่จะช่วยในการวางแผนการเดินทางโดยเครื่องบิน ซึ่งช่วยผู้ใช้ในการตัดสินใจเลือกเส้นทางที่ดีที่สุดในการเดินทางผ่านเมืองต่างๆ ไปยังจุดหมาย

บทที่ 2
ทฤษฎีและหลักการ

2.1 พอยน์เตอร์และลิงค์ลิสต์

พอยน์เตอร์ คือ ตัวแปรที่ใช้เก็บตำแหน่งของตัวแปรอื่น จากคุณสมบัติของตัวแปรพอยน์เตอร์จึงมองดูเหมือนกับตัวชี้ซึ่งชี้ไปที่ตำแหน่งของตัวแปรอื่น โดยตัวของมันเองแล้วพอยน์เตอร์ไม่ใช่ข้อมูล แต่ถ้าเราตามไปที่ตำแหน่งที่พอยน์เตอร์นั้นชี้อยู่ เราก็สามารถเข้าถึงข้อมูลและทำการประมวลผลข้อมูลนั้นได้

พอยน์เตอร์สามารถเลื่อนตำแหน่งได้ กล่าวคือ เราสามารถให้พอยน์เตอร์ชี้ไปที่ข้อมูลตัวถัดไปหรือที่ข้อมูลตัวก่อนหน้าก็ได้

การใช้คำสั่งในการประกาศตัวแปรพอยน์เตอร์ ทำได้ดังนี้

```
char  *p ;
```

p เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรชนิดอักขระ

ในกรณีที่พอยน์เตอร์ไม่มีข้อมูลให้อ้างถึง จะแสดงได้ดังนี้

```
p = NULL ;
```

NULL เป็นสัญลักษณ์พิเศษ ที่มีค่าคล้ายกับค่าคงที่ศูนย์

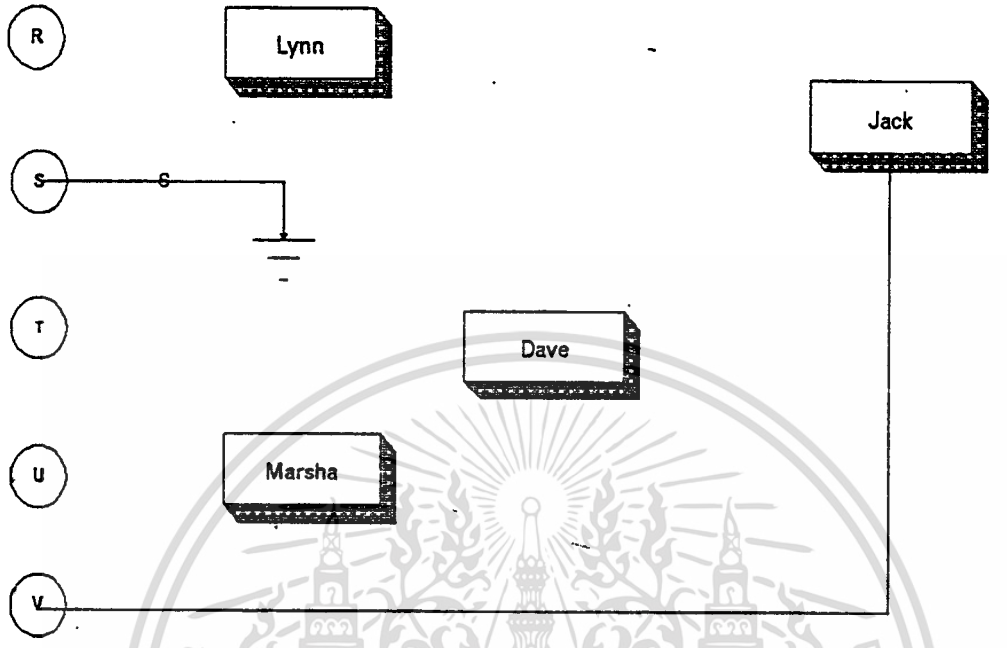
ในเรื่องของพอยน์เตอร์มีเครื่องหมาย 2 ชนิดที่ใช้คือ * และ & เครื่องหมาย * จะให้ค่าของข้อมูลซึ่งเก็บอยู่ในตำแหน่งที่เก็บอยู่ในตัวแปรพอยน์เตอร์ สำหรับเครื่องหมาย & จะให้ค่าตำแหน่งของตัวแปรซึ่งอยู่หลังเครื่องหมาย &

ในการนำพอยน์เตอร์มาประยุกต์ใช้งานในแอพพลิเคชันโปรแกรมนี้ จะเห็นได้จากการนำพอยน์เตอร์มาไว้รวมอยู่ในข้อมูลแบบโครงสร้าง เพื่อใช้เก็บตำแหน่งของข้อมูลโครงสร้างตัวถัดไป โดยหลักการนี้เป็นหลักการโครงสร้างข้อมูลแบบลิงค์ลิสต์นั่นเอง

โครงสร้างข้อมูลแบบลิงค์ลิสต์ เป็นโครงสร้างข้อมูลที่มีประโยชน์อย่างมากในการประมวลผลข้อมูล ซึ่งมีจำนวนไม่แน่นอนและสามารถจะเปลี่ยนแปลงได้ในขณะที่โปรแกรมทำงาน

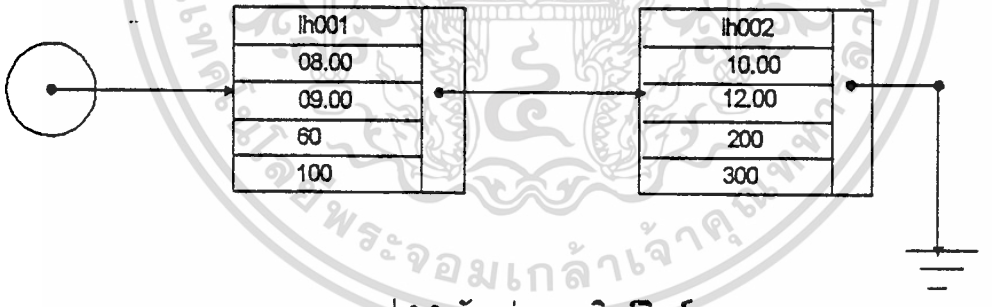
ในการอ่านข้อมูลที่เก็บไว้ในลิสต์ ให้เริ่มต้นที่จุดเริ่มต้นของลิสต์และไล่ตามลูกศรเรื่อยไปจนสิ้นสุดลิสต์ ในโครงงานนี้เราจะใช้ลิเนียร์ลิงค์ลิสต์ ซึ่งจะประกอบด้วยโหนด ที่เชื่อมต่อกันและลิสต์จะสิ้นสุดลงด้วยสัญลักษณ์พิเศษหรือ NULL

ตัวอย่างของพอยน์เตอร์ สามารถแสดงได้ดังรูป 2.1



รูป 2.1 ตัวอย่างของพอยน์เตอร์

ตัวอย่างของลิงค์ลิสต์ สามารถแสดงได้ดังรูป 2.2



รูป 2.2 ตัวอย่างของลิงค์ลิสต์

2.2 การจองเนื้อที่หน่วยความจำแบบไดนามิก

การแบ่งหน่วยความจำสำหรับเก็บตัวแปรแบบต่างๆ โดยปกติตัวแปรแบบโกลบอลจะเก็บไว้ในหน่วยความจำซึ่งมีขนาดแน่นอน ส่วนตัวแปรแบบโลคอลจะเก็บในหน่วยความจำที่เรียกว่า สแตก ทุกครั้งที่มีการเรียกฟังก์ชัน หน่วยความจำสแตกจะถูกใช้เสมอ หน่วยความจำที่อยู่ระหว่างสแตกและหน่วยความจำสำหรับตัวแปรแบบโกลบอลเรียกว่า หน่วยความจำแบบไดนามิก ซึ่งมีขนาดไม่แน่นอน

ในการจองเนื้อที่หน่วยความจำที่มีขนาดแน่นอนจะใช้ได้ดีในกรณีที่ จำนวนหน่วยความจำที่ถูกจองมีขนาดเล็ก หรือมีการใช้ค่าต่างๆ ในตารางข้อมูลนั้นๆ ตลอดเวลา

สำหรับการจองเนื้อที่หน่วยความจำแบบไดนามิก เป็นหลักการที่แอพพลิเคชันโปรแกรมนี้นำมาประยุกต์ใช้กับตารางเที่ยวบินต่างๆ เนื่องจากจำนวนเที่ยวบินของเมืองหนึ่งๆ มีค่าไม่แน่นอนและมีจำนวนไม่เท่ากันในแต่ละเมือง

หลักการจองเนื้อที่หน่วยความจำแบบไดนามิก เป็นการแก้ปัญหาหน่วยความจำไม่เพียงพอแก่การใช้งาน กล่าวคือ แทนที่จะใช้เป็นตารางอาร์เรย์ขนาดคงที่ในการเก็บข้อมูล เราจะจองเนื้อที่เท่าที่ตัวแปรของโปรแกรมต้องการใช้เท่านั้น และเมื่อเราต้องการใช้เนื้อที่เพิ่ม เราก็สามารถจองเพิ่มได้ และเมื่อใดที่เราไม่จำเป็นต้องใช้เนื้อที่ส่วนนั้นอีก เราก็สามารถคืนเนื้อที่ส่วนนั้นให้แก่ระบบ เพื่อนำไปใช้งานอื่นได้

การใช้คำสั่งในการจองเนื้อที่หน่วยความจำแบบไดนามิก ทำได้ดังนี้

```
p = (Node_type *) malloc (sizeof (Node_type)) ;
```

p เป็นตัวแปรพอยน์เตอร์ที่ชี้ไปยังตัวแปรโครงสร้างที่ชื่อว่า Node_type

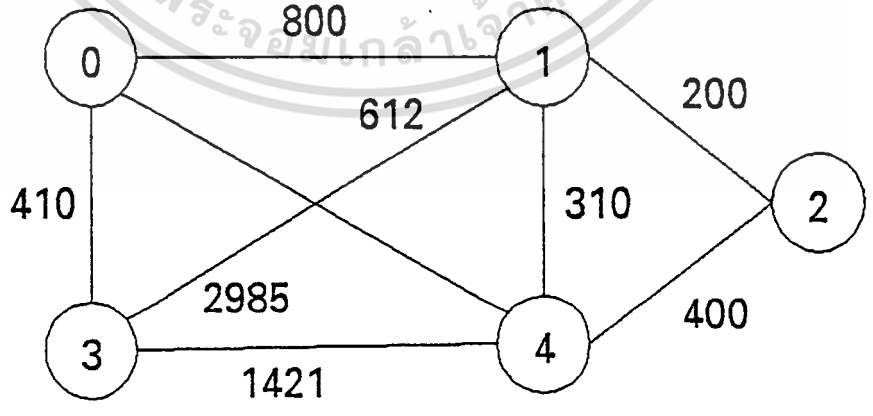
การใช้คำสั่งในการคืนเนื้อที่หน่วยความจำให้แก่ระบบ ทำได้ดังนี้

```
free (p) ;
```

2.3 อัลกอริธึมในการคำนวณหาเส้นทางที่ดีที่สุด

ในการคำนวณหาเส้นทางการเดินทางนั้น เราจะนำอัลกอริธึมของ Dijkstra มาประยุกต์ใช้งาน โดยมีการเปลี่ยนเกณฑ์ในการพิจารณาตามที่ผู้ใช้ต้องการ เช่น ใช้เกณฑ์เป็นระยะทาง, ราคาหรือเวลา เป็นต้น

เพื่อให้เกิดความเข้าใจ ลองดูตัวอย่างง่ายๆ ในการนำอัลกอริธึมนี้มาใช้งาน กำหนดให้ มีเส้นทางการเดินทางจากเมืองต่างๆ ถึงกันเป็นจำนวน 5 เมือง ดังรูป 2.3



รูป 2.3 รูปแสดงเส้นทางการเดินทางระหว่างเมืองตัวอย่าง 5 เมือง

โดยที่เมืองที่เครื่องบินหยุดรับส่งผู้โดยสารแทนด้วย รูปวงกลม และระยะทางจากเมืองหนึ่งไปยังอีกเมืองหนึ่งแทนด้วย ค่าตัวเลขที่อยู่บนเส้นเชื่อมเมืองต่างๆ นั้น

การประกาศตัวแปรสำคัญที่ใช้ในการคำนวณและเก็บข้อมูลในอัลกอริธึม มีดังนี้

VAR DISTANCE (N) : INTEGER เก็บระยะทางจากเมืองต้นทางถึงเมือง N

VAR PATH (N) : INTEGER เก็บเมืองที่เป็นทางผ่านก่อนถึงเมือง N

VAR INCLUDED (N) : BOOLEAN แบ่งได้เป็น 2 กรณี ดังนี้

ถ้าเป็นจริง คือหาเส้นทางสั้นที่สุดไปยังเมือง N ได้แล้ว

ถ้าเป็นเท็จ คือยังหาเส้นทางสั้นที่สุดไปยังเมือง N ไม่ได้

2.3.1 อัลกอริธึมของ Dijkstra

อัลกอริธึมสามารถแบ่งได้เป็น 2 ส่วนหลัก คือ

2.3.1.1. ส่วนของการให้ค่าเริ่มต้นตัวแปร

ในส่วนของการให้ค่าเริ่มต้นตัวแปรนี้ สามารถแบ่งได้เป็นการให้ค่าแก่ตัวแปรหลัก 3 ตัว ดังต่อไปนี้

การให้ค่าเริ่มต้นตัวแปร INCLUDED ทำได้โดย

INCLUDED (SOURCE) เป็นจริง

INCLUDED (J) เป็นเท็จ สำหรับเมือง J อื่นๆ

การให้ค่าเริ่มต้นตัวแปร DISTANCE ทำได้โดย

DISTANCE (SOURCE) = 0

DISTANCE (J) = ระยะทางในตารางที่คู่ลำดับ (SOURCE, J)

การให้ค่าเริ่มต้นตัวแปร PATH ทำได้โดย

PATH (SOURCE) = SOURCE

PATH (J) = SOURCE สำหรับเมือง J อื่นๆ

2.3.1.2. ส่วนของการวนลูปทำซ้ำ

ในส่วนของการวนลูปทำซ้ำนี้ จะเป็นการเปลี่ยนแปลงค่าตัวแปร DISTANCE (J), PATH (J) ไปเรื่อยๆ จนกระทั่งค่าใน INCLUDED (J) เปลี่ยนจากเท็จเป็นจริง ซึ่งหมายความว่า DISTANCE (J) ได้เก็บระยะทางสั้นที่สุดจากเมืองต้นทางไปยังเมือง J แล้ว

อัลกอริธึมในส่วนนี้ มีดังต่อไปนี้

REPEAT

FIND เมือง J ใดๆ ที่มีระยะทางห่างจากเมืองต้นทางน้อยที่สุด

โดยหาจากเมืองที่ยังไม่ถูกรวมอยู่ในอาร์เรย์ INCLUDED

MARK เมือง J เป็นเมืองที่ถูกรวมอยู่ในอาร์เรย์ INCLUDED

FOR แต่ละเมือง K อื่นๆ ที่ยังไม่ถูกรวมอยู่ในอาร์เรย์ INCLUDED

IE เมือง K มีเส้นทางเชื่อมต่อกับเมือง J THEN

IE ระยะห่างของเมือง J จากเมืองต้นทาง + M(J,K) < ระยะห่างของเมือง K จากเมืองต้นทาง

THEN

ระยะห่างของเมือง K จากเมืองต้นทาง := ระยะห่างของเมือง J จากเมืองต้นทาง + M(J,K)

PATH (K) := เมือง J

ENDIF

ENDIF

ENDFOR

UNTIL ทุกๆ เมืองถูกรวมอยู่ในอาร์เรย์ INCLUDED

เพื่อสร้างความเข้าใจในอัลกอริธึมให้มากยิ่งขึ้น เราสามารถติดตามผลการทำงานได้ดังนี้

ในตอนให้ค่าเริ่มต้นตัวแปรต่างๆ จะได้ค่าตัวแปรต่างๆ ดังนี้

DISTANCE(1) = 800 PATH(1) = 0

DISTANCE(2) = 2985 PATH(2) = 0

DISTANCE(3) = 310 PATH(3) = 0

DISTANCE(4) = 200 PATH(4) = 0

เมื่อเข้าสู่ส่วนของการวนลูปทำซ้ำ จะได้ค่าต่างๆ ดังนี้

การวนลูปครั้งที่ 1 INCLUDED โหนด 4 ค่าต่างๆ ใน DISTANCE, PATH ไม่เปลี่ยน

การวนลูปครั้งที่ 2 INCLUDED โหนด 3 เปลี่ยนแปลงค่าต่างๆ ใน DISTANCE, PATH ได้ดังนี้

DISTANCE(1) = 800 PATH(1) = 0

DISTANCE(2) = 1731 PATH(2) = 3

DISTANCE(3) = 310 PATH(3) = 0

DISTANCE(4) = 200 PATH(4) = 0

การวนลูปครั้งที่ 3 INCLUDED โหนด 1 เปลี่ยนแปลงค่าต่างๆ ใน DISTANCE, PATH ได้ดังนี้

DISTANCE(1) = 800 PATH(1) = 0

DISTANCE(2) = 1210 PATH(2) = 1

DISTANCE(3) = 310 PATH(3) = 0

DISTANCE(4) = 200 PATH(4) = 0

การวนลูปครั้งที่ 4 INCLUDED โหนด 2 ค่าต่างๆ ใน DISTANCE, PATH ไม่เปลี่ยน

จากตัวอย่างข้างต้น เป็นการคำนวณหาเส้นทางการเดินทางโดยใช้ ระยะทาง เป็นเกณฑ์ในการตัดสินใจ ถ้าต้องการคำนวณหาเส้นทางการเดินทางโดยใช้เกณฑ์อื่น เช่น ราคา หรือ เวลา เป็นต้น ก็สามารถประยุกต์ใช้อัลกอริธึมนี้ได้เช่นกัน โดยสร้างเป็นตารางเก็บราคา หรือ เวลา แทนการเก็บระยะทาง

สำหรับการใช้เกณฑ์ของเวลาในการคำนวณหาเส้นทางการเดินทาง มีข้อพิจารณาเพิ่มเติม ดังนี้

ต้องคำนึงถึงวันและเวลาในการเริ่มออกเดินทาง

เวลารวมที่ใช้ในการเดินทาง จะหมายความถึง เวลารวมที่ใช้ในการบินจริงกับเวลารวมที่ใช้ในการรอ ไม่ว่าจะรอเพื่อขึ้นเครื่องบินหรือรอเพื่อเปลี่ยนเที่ยวบินก็ตาม

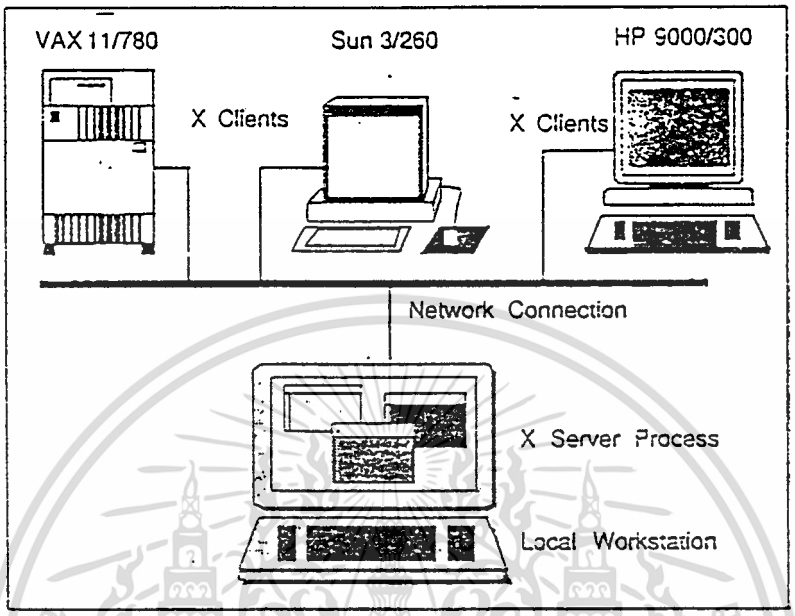
2.4 ระบบ X Window

X Window เป็นระบบวินโดว์ที่ทำให้โปรแกรมเมอร์สามารถพัฒนาส่วนที่ติดต่อกับผู้ใช้ (graphical user interface) ที่เคลื่อนย้าย (portable) ได้ คุณสมบัติที่สำคัญที่สุดของระบบ X Window (เรียกสั้นๆ ว่า X) คือ มีสถาปัตยกรรมที่ไม่ขึ้นกับฮาร์ดแวร์ X มีการแสดงผลเป็นแบบบิทแมปกราฟิก (bitmap graphic) ไม่ว่าจะเป็นรูปภาพหรือตัวอักษร X มีลักษณะการแสดงผลโดยแบ่งจอภาพออกเป็นส่วนๆ ที่เรียกว่า วินโดว์ (window) โดยแต่ละส่วนจะมีส่วนรับข้อมูลและส่วนแสดงผล (input และ output) เป็นของตนเอง

2.4.1 ลักษณะการทำงาน

X มีสถาปัตยกรรมเป็นระบบลูกข่ายกับผู้ให้บริการ (client-server) ดังรูป 2.4 โดยจะมีผู้ให้บริการ (server) รับผิดชอบการทำงานของอุปกรณ์อินพุต และเอาท์พุตทั้งหมด และมีลูกข่าย (client) เป็นผู้ใช้บริการ รายละเอียดการทำงานของผู้ให้บริการจะถูกซ่อนไว้จากลูกข่าย นอกจากนี้สถาปัตยกรรมแบบกระจายของ X ทำให้ผู้ให้บริการและลูกข่ายสามารถทำงานบนเครื่องใดก็ได้บนเน็ตเวิร์ค ข้อมูลที่ใช้ส่งไปมาระหว่างลูกข่ายกับผู้ให้บริการนั้นมีชื่อเรียกแตกต่างกันกล่าวคือ

ข้อมูลที่ลูกข่ายส่งไปยังผู้ให้บริการเรียกว่า request ซึ่งอาจไม่ถูกนำไปประมวลผลทันที แต่จะถูกนำไปเก็บในคิว ลูกข่ายจะไม่รอการตอบสนองจากผู้ให้บริการ ตัวอย่างเช่น การเปลี่ยนขนาดของวินโดว์ ลูกข่ายจะส่ง request ไปบอกให้ตัวผู้ให้บริการวาดหน้าจอใหม่ เป็นต้น



รูป 2.4 สถาปัตยกรรมระบบลูกข่ายกับผู้ให้บริการ

ข้อมูลที่ผู้ให้บริการส่งไปยังลูกข่ายเรียกว่า event เช่น เมื่อผู้ใช้ลากเมาส์ก็จะมีการส่ง event ไปยังลูกข่ายเพื่อให้รับรู้และทำตามคำสั่งที่กำหนดไว้

ในการจัดการข้อมูลในเครือข่าย ลูกข่ายจะใช้เพียงหมายเลขประจำตัว (ID) ของข้อมูลในการอ้างอิง ซึ่งจะช่วยลดภาระระบบเครือข่ายลง X โปรโตคอลเป็นข้อตกลงที่ใช้ในการส่งข้อมูลระหว่างลูกข่ายและผู้ให้บริการ ทำให้ลูกข่ายสามารถติดต่อกับผู้ให้บริการใดๆ ก็ได้ถ้าใช้โปรโตคอล X เหมือนกัน

2.4.2 สถาปัตยกรรมของระบบ X Window

เราสามารถเขียนโปรแกรมบนระบบ X Window ได้หลายแบบ ขึ้นกับการเลือกระดับขั้นของ X ไลบรารี ดังรูป 2.5

2.4.2.1 Xlib

2.4.2.2 Xt Intrinsic

2.4.2.3 Widget set

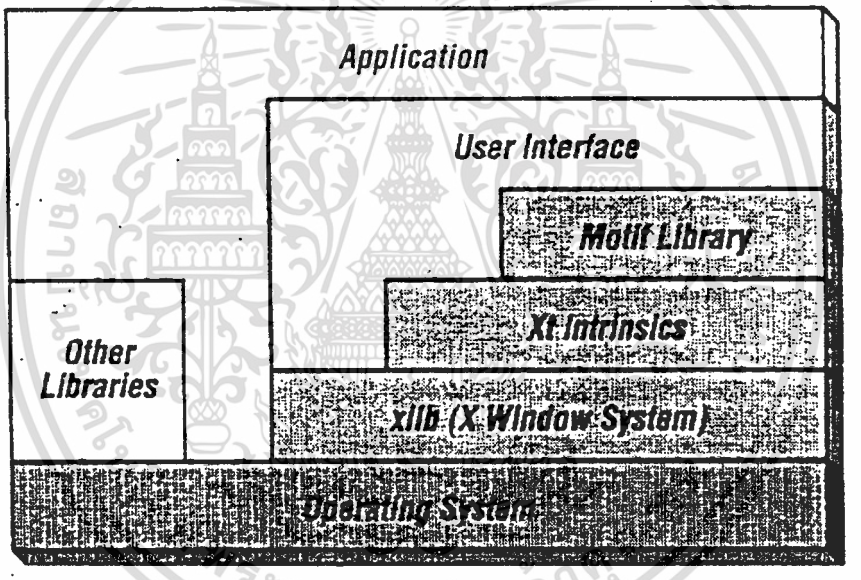
ทั้งสามส่วนนี้สามารถทำงานร่วมกันได้อย่างดี เราสามารถใช้ไลบรารีระดับสูงในการเขียนโปรแกรมและหันมาใช้ฟังก์ชันของ Xlib ได้เมื่อต้องการ

2.4.2.1 Xlib

Xlib เป็นไลบรารีภาษาซีที่เป็นชั้นล่างสุดของทั้ง 3 ชั้น มีฟังก์ชันสำหรับติดต่อกับ X โปรโตคอลควบคุมการแสดงผลวินโดว์ และอุปกรณ์อินพุต

2.4.2.2 XtIntrinsic

การใช้ Xlib สำหรับการเขียนโปรแกรมที่มีการติดต่อกับผู้ใช้ (graphical user interface : GUI) นั้นซับซ้อนและยุ่งยาก ดังนั้นจึงได้มีการออกแบบและสร้างฟังก์ชันต่างๆ ที่จำเป็นในการสร้าง GUI ขึ้นมา Xt มีฟังก์ชันและรูทีนต่างๆ สำหรับสร้างและเปลี่ยนแปลงส่วนที่จะต้องติดต่อกับผู้ใช้ที่เรียกว่า วิดเจต (widget) สำหรับ Xt นี้จะมีลักษณะการโปรแกรมเป็นแบบวัตถุ (object oriented) โดยมีพื้นฐานเป็นภาษาซี แต่เรียกใช้ไลบรารีที่เรียกว่า Xt หรือ X Toolkit Intrinsic



รูป 2.5 ไลบรารีสำหรับการพัฒนาส่วนที่ติดต่อกับผู้ใช้

Xt จะกำหนดคลาสพื้นฐาน (base class) ของวิดเจต โดยวิดเจตลูกสามารถสืบทอดหรือขยายลักษณะที่มีในคลาสพื้นฐานได้ วิดเจตถูกออกแบบมาโดยสามารถมีการเปลี่ยนค่าทรัพยากร (resource) โดยผู้ใช้ตอนรันไทม์ (runtime) ได้ เมื่อโปรแกรมหนึ่งทำงาน Xt จะทำการโหลดข้อมูลจากแฟ้มข้อมูลระบบและแฟ้มข้อมูลพิเศษ ที่เรียกรวมว่า ฐานข้อมูลทรัพยากร (resource database) เพื่อนำมากำหนดรายละเอียดของวิดเจต

2.4.2.3 Widget set

ชุดวิดเจตที่ทำงานร่วมกับ Xt มี 2 แบบใหญ่ๆ คือ Motif กับ Open Look แต่สำหรับโครงงานนี้เราจะกล่าวถึงและใช้เฉพาะวิดเจตของโมทีฟเท่านั้น โดยจะกล่าวถึงรายละเอียดในหัวข้อถัดไป

2.4.3 การแสดงผลและจอภาพ

ใน X นั้นการแสดงผลเป็นขบวนการหนึ่งของผู้ให้บริการ ในขณะที่จอภาพเป็นอุปกรณ์การแสดงผล การแสดงผลหนึ่งๆ สามารถรองรับได้หลายจอภาพ โดยปกติคำว่าจอภาพจะเป็นศัพท์ที่ใช้แทนได้ในความหมายเดียวกันกับผู้ให้บริการ

ก่อนที่ลูกข่ายจะสามารถติดต่อกับผู้ให้บริการจะต้องมีการเปิดการเชื่อมต่อกับผู้ให้บริการ เมื่อมีการติดต่อกันแล้ว ลูกข่ายสามารถใช้จอใดก็ได้ที่ผู้ให้บริการควบคุมอยู่ X มีระบบรักษาความปลอดภัย โดยผู้ให้บริการจะไม่ยอมให้ลูกข่ายที่ทำงานกับผู้ให้บริการอื่นทำการเชื่อมต่อได้

2.4.4 ทรัพยากร

ใน X นั้น ผู้ให้บริการจะควบคุมทรัพยากรที่ใช้งานโดยระบบวินโดว์ ทรัพยากร ได้แก่ วินโดว์ บิทแมป สี และโครงสร้างข้อมูลที่ใช้ในการเขียนโปรแกรม ผู้ให้บริการจะดูแลทรัพยากรโดยตัวมันเอง และยอมให้ลูกข่ายสามารถใช้ข้อมูลเหล่านี้ได้ โปรแกรมลูกข่ายจะอ้างอิงทรัพยากรเหล่านี้โดยใช้ รหัสทรัพยากร (resource identification) หรือเรียกสั้นๆ ว่า ID รหัสทรัพยากรจะถูกกำหนดโดยผู้ให้บริการ

ผู้ให้บริการจะสร้างและทำลายทรัพยากรตามคำขอของลูกข่าย ในกรณีที่ลูกข่ายสิ้นสุดการเชื่อมต่อกับผู้ให้บริการ ทรัพยากรต่างๆ จะถูกยกเลิกโดยอัตโนมัติ นอกจากนั้น ผู้ใช้สามารถกำหนดโหมดการยกเลิกของทรัพยากรได้ ซึ่งมีผลต่ออายุขัยของทรัพยากร

2.4.5 การทำงานพื้นฐานของวินโดว์

วินโดว์เป็นทรัพยากรที่พื้นฐานที่สุดของ X มีลักษณะเป็นพื้นที่สี่เหลี่ยมบนจอ ซึ่งจะต่างจากวินโดว์ในระบบอื่นตรงที่ วินโดว์ใน X นั้น ไม่มี title bar, scroll bar หรือส่วนที่เกี่ยวข้องกับการตกแต่งอื่นๆ วินโดว์จะปรากฏเป็นเพียงพื้นที่สี่เหลี่ยมที่มีขอบเท่านั้น โปรแกรมเมอร์สามารถนำวินโดว์หลายๆ ตัวมารวมกัน เพื่อสร้างส่วนที่เกี่ยวข้องกับการติดต่อกับผู้ใช้ เช่น title bar, scrool bar เป็นต้น

ผู้ให้บริการจะสร้างวินโดว์เมื่อมีการ request ขอจากลูกข่าย ผู้ให้บริการจะดูแลรักษาโครงสร้างข้อมูลที่กำหนดตัววินโดว์นั้น ในขณะที่ลูกข่ายจะใช้รหัสทรัพยากรเพื่อที่ติดต่อกับวินโดว์นั้น ลูกข่ายสามารถส่ง request ไปยังผู้ให้บริการ เพื่อขอเปลี่ยนขนาดของวินโดว์ ตำแหน่ง สีและคุณสมบัติอื่นๆ นอกจากนี้ ผู้ใช้ยังสามารถส่ง request เพื่อให้มีการแสดงข้อความหรือกราฟิกบนวินโดว์ได้ ถึงแม้วินโดว์หนึ่งจะถูกสร้างขึ้นมาเพื่อตอบสนอง request ที่มาจากลูกข่ายลูกข่ายหนึ่ง แต่ลูกข่ายอื่นก็สามารถทำงานกับวินโดว์นั้นได้เช่นเดียวกันหากมันมีรหัสทรัพยากรเพื่อใช้ในการอ้างอิง

2.4.6 การจัดโครงสร้างเป็นลำดับชั้นของวินโดว์

X จัดเรียงวินโดว์เป็นลำดับชั้นเรียกว่า ลำดับชั้นของวินโดว์ (Window tree) โดยวินโดว์ตัวที่อยู่บนสุดของลำดับชั้นมีชื่อว่า วินโดว์ราก (Root window) ผู้ให้บริการจะสร้างวินโดว์รากสำหรับแต่ละจอภาพที่มันควบคุม วินโดว์นี้จะกินพื้นที่ทั้งจอภาพและเคลื่อนย้ายหรือปรับขนาดไม่ได้ วินโดว์อื่นนอกเหนือจากวินโดว์ราก จะต้องมีวินโดว์แม่ (Parent window) และอาจมีวินโดว์ลูก (Child window) ด้วย

X จะจำกัดขนาดและตำแหน่งของวินโดว์ โดยผู้ให้บริการจะไม่ยอมให้มีการแสดงส่วนของวินโดว์ที่อยู่นอกขอบเขตของวินโดว์แม่ ลักษณะของการจัดวินโดว์บนจอนั้นจะมีลักษณะของการเรียงทับกัน

2.4.7 การจับคู่และการแสดงวินโดว์

วินโดว์อาจไม่ปรากฏบนจอภาพให้ผู้ใช้เห็นเสมอไป เมื่อผู้ให้บริการสร้างวินโดว์ขึ้นมา จะมีการกำหนดโครงสร้างข้อมูลที่เกี่ยวกับวินโดว์นั้นไว้ แต่ยังไม่เรียกใช้รูทีนที่นำวินโดว์นั้นไปแสดงบนจอภาพ ลูกข่ายสามารถขอให้วินโดว์นั้นปรากฏบนจอภาพได้โดยส่ง request เพื่อทำการขอจับคู่วินโดว์ แต่บางครั้งถึงแม้วินโดว์จะถูกจับคู่แล้ว แต่มันก็ยังไม่ปรากฏบนจอภาพ เนื่องจากเหตุผลต่อไปนี้

วินโดว์นั้นถูกบังโดยวินโดว์อื่น มันจะปรากฏบนจอภาพก็ต่อเมื่อวินโดว์ที่บังมันนั้น ถูกเคลื่อนย้ายออกไป

วินโดว์แม่ในลำดับชั้นของวินโดว์นั้นยังไม่ถูกจับคู่ ก่อนที่วินโดว์หนึ่งๆ จะปรากฏบนจอภาพได้นั้น วินโดว์ที่อยู่สูงขึ้นไปในลำดับชั้นของมันจะต้องถูกจับคู่

วินโดว์อยู่นอกขอบเขตการแสดงผลของวินโดว์แม่ มันจะปรากฏบนจอภาพก็ต่อเมื่อ ได้มีการปรับขนาดวินโดว์แม่หรือเปลี่ยนตำแหน่งของวินโดว์นั้นให้อยู่ภายในขอบเขตของวินโดว์แม่

2.4.8 การดูแลรักษาข้อมูลในวินโดว์

ข้อมูลในวินโดว์ จะต้องมีการถูกเก็บรักษาและนำมาแสดงผลใหม่ทุกครั้งที่มีการเปลี่ยนแปลงสถานะของวินโดว์บนจอภาพ ข้อมูลเหล่านี้จะถูกเก็บไว้ในรูปบิทแมป

ใน X นั้น หน้าที่การดูแลรักษาข้อมูลในวินโดว์ จะตกอยู่กับลูกข่ายที่ทำงานกับวินโดว์นั้น ส่วนผู้ให้บริการจะทำหน้าที่แจ้งให้ลูกข่ายรู้ว่าเมื่อใดต้องมีการแสดงวินโดว์บนจอภาพ เพราะฉะนั้นลูกข่ายจะต้องมีความพร้อมเสมอที่จะแสดงข้อมูลในวินโดว์

2.4.9 การบริหารวินโดว์

ตัวบริหารวินโดว์ (Window manager) จะยอมให้ผู้ใช้สามารถควบคุมขนาดและตำแหน่งของวินโดว์ที่ปรากฏบนจอภาพได้ ใน X ตัวบริหารวินโดว์นี้ แท้ที่จริงแล้ว ก็คือโปรแกรมลูกข่ายธรรมดาตัวหนึ่ง Inter-Client Communication Manual หรือ ICCM เป็นโปรโตคอลที่กำหนดการทำงานระหว่างตัวบริหารวินโดว์และโปรแกรมต่างๆ เพื่อให้สามารถทำงานร่วมกันได้อย่างมีประสิทธิภาพ

2.4.10 Request

เมื่อลูกข่ายต้องการบริการจากผู้ให้บริการ มันจะส่ง request ไปยังผู้ให้บริการ โดยทั่วไปนั้น request จะเกี่ยวกับการสร้าง ทำลาย ปรับวินโดว์ การแสดงข้อความหรือกราฟิกบนวินโดว์ ลูกข่ายสามารถขอข้อมูลเกี่ยวกับสถานะของวินโดว์และทรัพยากรต่างๆ ได้อีกด้วย

ผู้ให้บริการโดยปกติจะทำงานโดยไม่ขึ้นอยู่กับลูกข่าย และลูกข่ายก็ทำงานโดยไม่ขึ้นต่อกันด้วย ถึงแม้ผู้ให้บริการจะทำการตอบสนอง request ของโปรแกรมต่างๆ ตามลำดับที่เข้ามา แต่นั่นก็ไม่ได้หมายความว่า request จะถูกนำมาประมวลผลโดยทันที request ที่มาจากลูกข่ายนั้น จะถูกนำเข้าไปเก็บในคิวจนกว่าจะถูกนำไปประมวลผลโดยผู้ให้บริการ และลูกข่ายจะไม่รอการตอบสนองจากผู้ให้บริการด้วย

2.4.11 Event

ผู้ให้บริการจะติดต่อกับลูกข่ายโดยการส่ง event ผู้ให้บริการจะสร้าง event เพื่อเป็นการตอบสนองการกระทำของผู้ใช้ทั้งโดยทางตรงและทางอ้อม (เช่น การกดปุ่มแป้นพิมพ์ การลากเมาส์ เป็นต้น) นอกจากนี้ ผู้ให้บริการจะสร้าง event เพื่อแจ้งให้ลูกข่ายทราบถึงการเปลี่ยนแปลงทางสถานะของวินโดว์

เราจะพบว่ามี event ถึง 33 ชนิดใน X และมีกลไกที่ทำให้ลูกข่ายสามารถกำหนดชนิดของ Event ขึ้นมาใช้งานได้เองด้วย event ที่เกิดขึ้นนั้น จะถูกเก็บในคิวแบบ FIFO (First-In-First-Out) ลูกข่ายจะสามารถอ่านข้อมูลในคิวนี้ได้ ข้อมูลที่พบในคิวจะรวมถึง ชนิดของ event วินโดว์ที่เกิด event และข้อมูลที่เกี่ยวข้องสำหรับ event นั้น

2.5 โมทีฟ

โมทีฟ (Motif) เป็นข้อกำหนดลักษณะ look and feel ของส่วนที่ติดต่อกับผู้ใช้แบบกราฟิก โดยระบุรูปแบบของโปรแกรมที่ปรากฏบนจอและการติดต่อกับผู้ใช้โมทีฟถูกออกแบบโดย Open Software Foundation (OSF) ซึ่งเป็นองค์กรที่ไม่หวังผลกำไรที่เกิดจากการรวมตัวของบริษัทต่างๆ เช่น Hewlett-Packard, Digital, IBM เป็นต้น OSF มุ่งพัฒนาเทคโนโลยีที่ช่วยให้คอมพิวเตอร์จากผู้ผลิตรายต่างๆ สามารถทำงานร่วมกันได้

โมทีฟ ถูกกำหนดให้ทำงานภายใต้ระบบ X Window เพราะเราจะพบมันได้ในระบบปฏิบัติการต่างๆ เช่น ยูนิกซ์ อีเอ็มเอส ดอส และแมคอินทอช ทำให้มีความคล่องตัวและสามารถเคลื่อนย้ายไปใช้กับเครื่องอื่นได้ ในการเขียนโปรแกรมบนโมทีฟ นั้น เราจะมุ่งเน้นในการกำหนดรายละเอียดของวัตถุ (object) ที่ประกอบกันขึ้นเป็นส่วนที่ติดต่อกับผู้ใช้ (เช่น menu, button, dialog box เป็นต้น) โดยเราสามารถแบ่งการกำหนดรายละเอียดได้ดังนี้

แบบจำลองเอาท์พุต (output model) ที่ระบุรูปแบบการแสดงผลของวัตถุบนจอ เช่น รูปร่างของปุ่มกด ตำแหน่งของวินโดว์ เป็นต้น

แบบจำลองอินพุต (input model) ที่ระบุวิธีการที่ผู้ใช้สามารถติดต่อกับส่วนต่างๆ บนจอได้

ลักษณะการใช้งานโมทีฟ คล้ายกับการใช้ไมโครซอฟวินโดว์ เพราะโมทีฟถูกออกแบบตามข้อกำหนด CUA (Common User Access) ของไมโครซอฟท์

2.5.1 วิดเจต

โมทีฟไลบรารี จะกำหนดวิดเจตที่เป็นส่วนประกอบพื้นฐานของ GUI เช่น label, pushbutton, menu และมีวิดเจตควบคุม (manager widget) ที่มีหน้าที่ควบคุมการวัดเรียงของวิดเจตอื่นๆ โดยที่ผู้ใช้ไม่ต้องคำนึงถึงตำแหน่งของวิดเจตต่างๆ เมื่อวินโดว์ในโปรแกรมถูกย้ายตำแหน่งหรือเปลี่ยนขนาด

วิดเจตแต่ละตัวสามารถกำหนดได้โดยไม่ขึ้นกับโปรแกรม เช่น วิดเจต pushbutton รู้วิธีวาดตัวมันเองบนจอ การเน้น (highlight) ปุ่มเมื่อถูกเลือกโดยผู้ใช้ และการตอบสนองเมื่อมีการกดปุ่มเมาส์ ลักษณะการทำงานเหล่านี้ถูกกำหนดอยู่ในโมทีฟไลบรารี

2.5.2 วิดเจตที่ใช้ทั่วไปในโมทีฟ

วิดเจตที่ใช้ทั่วไป ได้แก่

2.5.2.1 Label

เป็นวิดเจตที่ถูกนำมาใช้มากที่สุดในการพัฒนาส่วนที่ติดต่อกับผู้ใช้ โดยมีหน้าที่สำหรับแสดงข้อความเป็นตัวอักษรหรือเป็นรูป และสามารถควบคุมสีและการจัดเรียงได้

2.5.2.2 Push Button

สามารถแสดงข้อความได้เช่นเดียวกับ label เพราะสืบทอดมา มีสภาวะ 3 สภาวะ ได้แก่

armed เมื่อตัวชี้เลื่อนเข้ามาบนปุ่ม - ปุ่มจะถูกเน้น (highlight)

activated เมื่อปุ่มที่มีสภาวะ armed นั้นถูกกด

unarmed เมื่อปุ่มไม่ได้ถูกเลือก

2.5.2.3 List

เป็นวิดเจตที่แสดงข้อมูลที่ผู้ใช้สามารถเลือกได้เป็นชุด list จะรับข้อมูลที่นำมาแสดงจากอาร์เรย์แบบ compound string ที่ถูกกำหนดขึ้นมาโดยโปรแกรม ข้อมูลนี้เมื่อถูกนำมาแสดง จะถูกนำมาจัดเรียงเป็นแถว 1 หลัก เราเลือกข้อมูลที่ต้องการได้โดยการใช้แป้นพิมพ์หรือเมาส์

วิดเจตควบคุม (manager widget) ได้แก่

2.5.2.4 Drawing Area ทำให้มีพื้นที่สำหรับแสดงกราฟิก

2.5.2.5 Scrolled Window มี scroll bar สำหรับดูข้อมูลที่มีขนาดใหญ่กว่าขนาดต่างๆ

2.5.2.6 Row Column จะจัดเรียงวิดเจตที่อยู่ภายใต้เป็นแถวและหลักตามที่กำหนด

2.5.2.7 Form จะจัดเรียงวิดเจตที่อยู่ภายใต้เป็นบล็อกๆ และให้วิดเจตเหล่านี้สามารถเชื่อมต่อกันได้

2.6 ความสัมพันธ์ระหว่าง Xlib Xt Intrinsic และโมทีฟ

ในการสร้างส่วนที่ติดต่อกับผู้ใช้นั้น เราจะต้องใช้ทั้งโมทีฟ (Xm) ไลบรารีและ Intrinsic Xt ไลบรารี โดยที่ Xt จะเกี่ยวกับการสร้างและกำหนดค่าทรัพยากร ส่วน Xm จะเป็นการสร้างวิดเจต โปรแกรมจะติดต่อกับ Xlib เมื่อต้องการทำงานด้านกราฟิกหรือประมวล event

ในระบบ จริงๆแล้วโปรแกรมจะติดต่อทำงานร่วมกับทุกชั้นในระบบ รวมทั้งระบบปฏิบัติการและไลบรารีอื่นๆ ด้วย แต่ส่วนที่ทำหน้าที่ติดต่อกับผู้ใช้ควรทำงานกับ Xlib, Xt และโมทีฟเท่านั้น

2.7 X กราฟิก

2.7.1 การแสดงจุดและสี

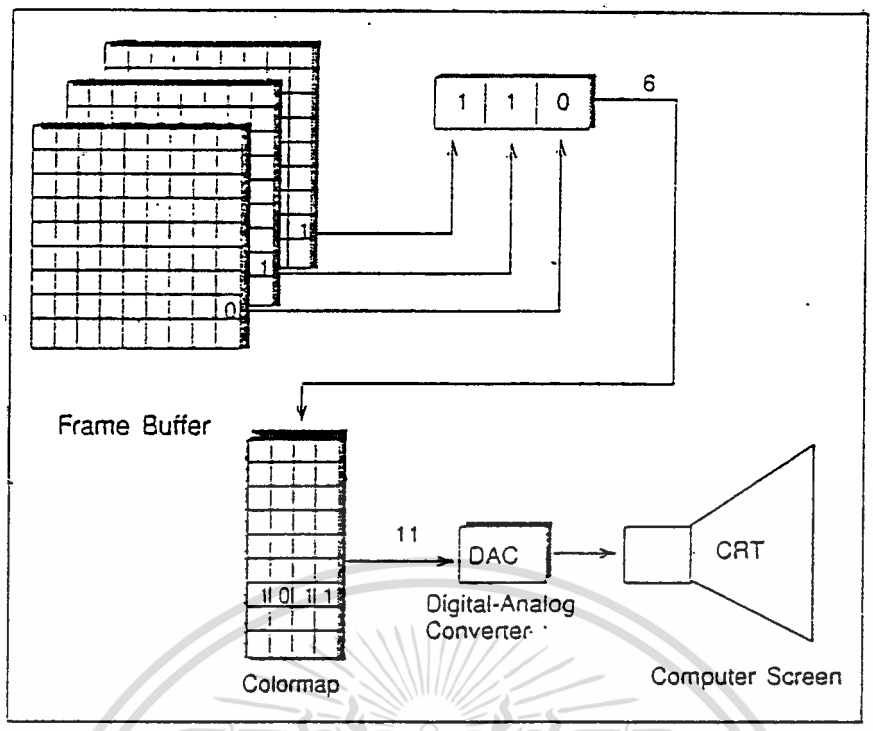
ระบบการแสดงผลของ X นั้น เป็นแบบบิทแมป สำหรับการแสดงผลแบบสีเดียวหรือจอขาวดำจะใช้จำนวน 1 บิท ต่อ 1 จุด (1บิทแทนได้ 2 ค่า คือ 0 กับ 1 หรือแทนขาวกับดำ) สำหรับจอสีที่สามารถแสดงผลเป็นระดับสีเทาหรือแสดงสีจะไม่ใช้การแทนค่าด้วยสีโดยตรง แต่จะใช้ดัชนี (index) ชี้ไปยังแผนผังสี (colormap) ซึ่งเป็นอาร์เรย์ของสี

เมื่อต้องการวาดจุดๆ หนึ่งบนจอ โปรแกรมจะนำค่าๆ หนึ่งไปเก็บในเฟรมบัฟเฟอร์ (frame buffer) ซึ่งเป็นพื้นที่ส่วนหนึ่งในหน่วยความจำ จะต้องมีการกำหนดในเฟรมบัฟเฟอร์ให้มีอย่างน้อย 1 บิทสำหรับแต่ละพิกเซลที่แสดงบนจอ จำนวนบิทสำหรับการแสดงพิกเซลหนึ่งๆ เรียกว่า ระนาบบิท (bit plane) ฉะนั้นจำนวนสีที่สามารถนำมาแสดงบนจอได้ในเวลาหนึ่งสามารถหาได้จากสมการ จำนวนสี = 2 ยกกำลังจำนวนระนาบบิท

จำนวนระนาบนี้เรียกอีกอย่างได้ว่า ความลึก (depth) เพราะฉะนั้นสำหรับจอสีที่มี 4 ระนาบจะสามารถแสดงได้ 16 สี ถ้ามี 8 ระนาบจะสามารถแสดงได้ 256 สี สำหรับ X นั้นได้ออกแบบให้รองรับได้ถึง 32 ระนาบเลยทีเดียว

ค่าในเฟรมบัฟเฟอร์จะถูกนำมาใช้เป็นดัชนีชี้ไปยังแผนผังสี การแสดงผลในวงจรสีใช้วิธีการแยกสีออกเป็น 3 สี ได้แก่ สีแดง สีเขียว และสีน้ำเงิน สีที่ปรากฏบนจอเกิดจากลำอีเลคตรอนทั้งสามสีที่ยิงออกมาบนจอเป็นจุดเดียวกัน ค่าที่พบในแผนผังสี ณ ตำแหน่งที่ดัชนีชี้จะกำหนดสีและความเข้มของจุดที่ปรากฏบนจอ รูป 2.6 แสดงขั้นตอนการนำค่าในเฟรมบัฟเฟอร์มาแปลงเป็นสีที่ปรากฏบนจอภาพโดยใช้แผนผังสี

การทำงานกับกราฟิกจะต้องใช้ค่าต่างๆ เช่น ขนาดของเส้น สี ฟอนต์ เป็นต้น X ได้มีการกำหนดค่าเหล่านี้ในโครงสร้างข้อมูลที่เรียกว่า สภาวะของกราฟิก (graphical content) เรียกสั้นๆ ว่า GC



รูป 2.6 สถาปัตยกรรมการกำหนดสีในระบบ X Window

2.7.2 พิกแมปและบิทแมป

พิกแมป เป็นหน่วยความจำส่วนหนึ่งคล้ายกับพื้นที่บนจอภาพ โดยเก็บค่าจุดต่างๆ ของภาพเป็นอาร์เรย์ แต่ยังไม่แสดงออกมาบนจอภาพจนกว่าจะมีการคัดลอกข้อมูลในพิกแมปไปยังวินโดว์ที่มองเห็น สำหรับพิกแมปที่มีความลึก 1 นั้นจะเรียกว่า บิทแมป

2.8 การโปรแกรมเชิงวัตถุ

การโปรแกรมเชิงวัตถุ (OOP) เป็นวิธีการเขียนโปรแกรมที่ใช้แนวความคิดแบบเชิงวัตถุ โดยมองวัตถุเป็นสมาชิกในโปรแกรม วัตถุหนึ่งจะประกอบไปด้วยส่วนโค้ด คือ รูทีนจัดการต่างๆ ใช้ในการตอบสนองต่อวัตถุตัวอื่นในระบบเดียวกัน เรียกว่า วิธีการ (method) และส่วนข้อมูล ข้อมูลมีทั้งแบบส่วนตัว (private) วัตถุอื่นจะมองไม่เห็นและไม่สามารถนำข้อมูลส่วนนี้ไปใช้งานได้ ข้อมูลอีกประเภทเป็นแบบสาธารณะ (public) โดยที่วัตถุอื่นสามารถเปลี่ยนค่าได้ วัตถุใดๆในระบบจะสื่อสารกับวัตถุอื่นเพื่อให้บรรลุความต้องการของตน การสื่อสารลักษณะนี้เป็นลักษณะ ร้องขอและตอบสนอง เมื่อวัตถุตัวหนึ่งขอความช่วยเหลือจากวัตถุหนึ่งเราเรียกว่า มันกำลังส่งข้อความ (massage) ไปยังวัตถุอื่น

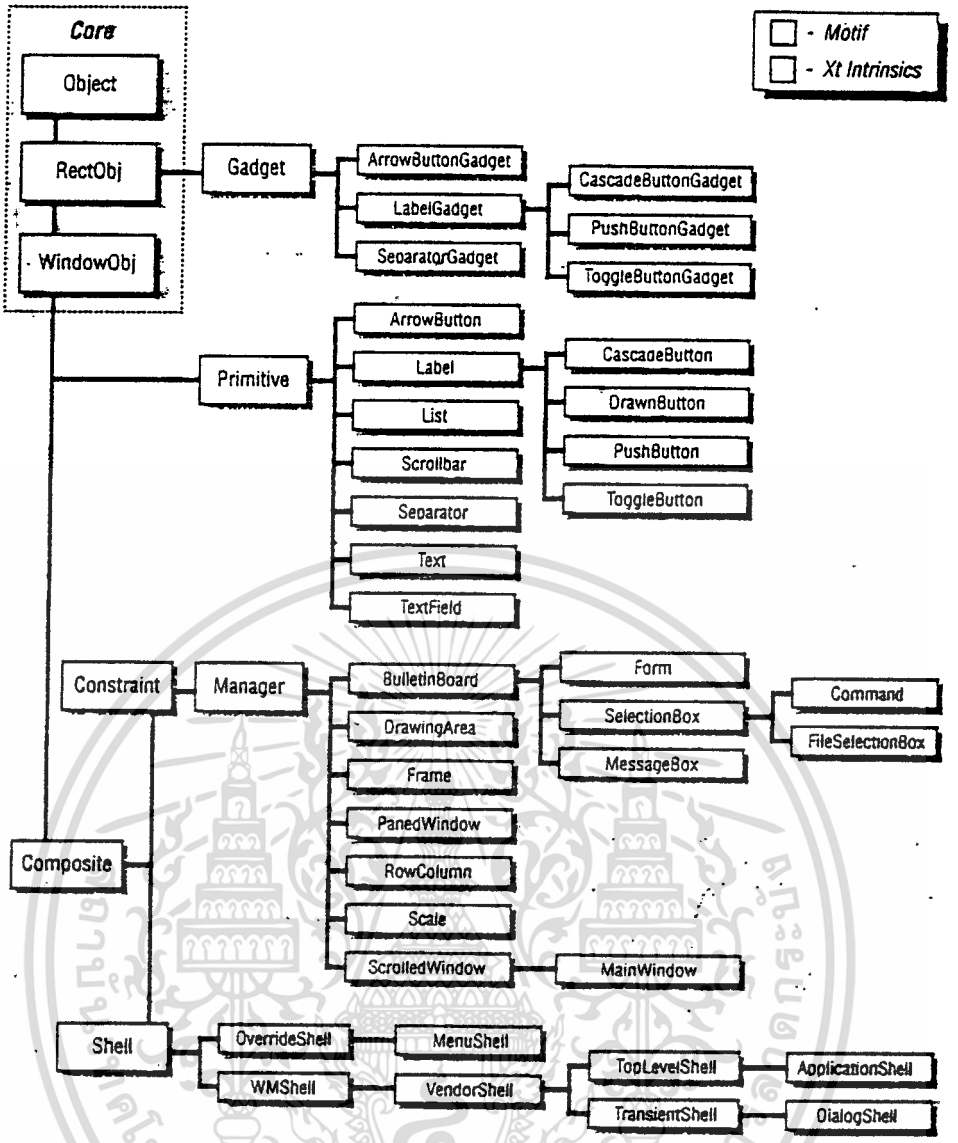
คลาส (class) กับวัตถุเป็นสิ่งคู่กัน คลาสจะให้ข้อมูลเกี่ยวกับลักษณะของวัตถุ เนื่องจากหลักการเขียนโปรแกรมเชิงวัตถุ เป็นหลักการอาศัยการถ่ายทอดคุณสมบัติเป็นหลัก คลาสแต่ละคลาสในระบบจึงสามารถให้กำเนิดลูกหลานได้ เรียกลูกหลานของคลาสว่า ซับคลาส (subclass)

ชุดหรือโครงสร้างความสัมพันธ์ของคลาสต่างๆ ในระบบ OOP เรียกว่า คลาสไลบรารี มันมีลักษณะคล้ายกับฟังก์ชันไลบรารีของภาษาซี แต่เป็นการเก็บนิยามทั้งหมดของคลาสทุกคลาส

การที่ Xt นั้นใช้การโปรแกรมเชิงวัตถุ ทำให้ผู้เขียนโปรแกรมใช้งานนั้นไม่ต้องยุ่งเกี่ยวกับโค้ดของวิดเจตแต่ละตัว โดยสามารถเรียกใช้ได้เฉพาะฟังก์ชันที่สร้าง ควบคุมและทำลายวิดเจต กับตัวแปรสาธารณะที่เป็นทรัพยากร ในระบบ X Window นั้นข่าวสารที่ส่งระหว่างวัตถุ (วิดเจต) อยู่ในรูปของ event และ request

การสร้างวิดเจตเป็นเพียงการกำหนด instance ของมันเท่านั้น โดยที่วิดเจตแต่ละตัวถูกกำหนดลักษณะไว้ก่อนแล้วในวิดเจตคลาส แต่ยังไม่มีการกำหนดค่าไว้ ดังนั้นวิดเจตชนิดเดียวกันอาจมีรูปร่างที่ไม่เหมือนกันก็ได้ ทั้งนี้ขึ้นอยู่กับการกำหนดค่ารายละเอียดต่างๆ เช่น ตำแหน่ง ความกว้าง ความสูง เป็นต้น

วิดเจตแต่ละชนิดมีการสืบทอด (inherit) ลักษณะการทำงานจากวิดเจตที่อยู่สูงขึ้นไปในลำดับชั้นของคลาส (class hierarchy) เช่น วิดเจต push button สามารถแสดง label ได้เพราะสืบทอดลักษณะนี้มาจาก label widget



รูป 2.7 โครงสร้างของวิดเจตคลาสในโมทีฟ

บทที่ 3
การคำนวณและการสร้าง

3.1 ความต้องการของระบบ

แอพพลิเคชันโปรแกรมที่สร้างขึ้นมา จะให้ผู้ใช้เลือกทำได้ ดังนี้

3.1.1 ให้ผู้ใช้ป้อนข้อมูลของตารางตามที่ผู้ใช้ต้องการ เก็บเป็นแฟ้มข้อมูล

3.1.2 อ่านแฟ้มข้อมูลที่ผู้ใช้ป้อนเก็บไว้มาแสดงบนหน้าจอ หรืออ่านแฟ้มข้อมูลที่เป็นฐานข้อมูลของระบบขึ้นมาแสดง

3.1.3 อ่านแฟ้มข้อมูลที่เป็นฐานข้อมูลของระบบขึ้นมาเพื่อใช้คำนวณตามเกณฑ์ที่ผู้ใช้สนใจ ดังนี้

3.1.3.1 เกณฑ์ระยะทางสั้นที่สุด

3.1.3.2 เกณฑ์ราคาถูกที่สุดเป็นเกณฑ์

3.1.3.3 เกณฑ์เวลาสั้นที่สุดเป็นเกณฑ์ โดยคิดเวลารวมเป็นเวลาที่ใช้บินจริงบวกเวลาที่ใช้ในการรอ

3.1.1 การป้อนข้อมูลของตาราง

แอพพลิเคชันโปรแกรมจะเก็บรายละเอียดต่างๆ ดังนี้

3.1.1.1 ชื่อแฟ้มข้อมูลที่ต้องการเก็บ

3.1.1.2 จำนวนเมืองที่ต้องการให้มีในตาราง โดยเก็บเป็นตัวแปรค่าคงที่

3.1.1.3 ระยะทางระหว่างเมืองต่างๆ ซึ่งถ้าเมืองใดไม่มีเส้นทางบินถึงกันโดยตรง จะต้องให้ผู้ใช้ป้อนค่าระยะทางเป็นศูนย์

3.1.1.4 เที่ยวบินที่บินระหว่างเมืองหนึ่งไปยังอีกเมืองหนึ่งอย่างน้อย 1 เที่ยวบินถ้ามีเส้นทางบินถึงกันโดยตรง โดยในแต่ละเที่ยวบินจะให้ผู้ใช้ป้อนข้อมูลเกี่ยวกับเที่ยวบินนั้นๆ ด้วย ดังนี้

หมายเลขประจำเครื่องบิน

เวลาออกของเครื่องบิน

เวลาถึง

เวลารวมที่ใช้ในการบินจริง

ราคา

มีเที่ยวบินเที่ยวต่อไปอีกหรือไม่

หลังจากนั้น จะเก็บข้อมูลที่รับเข้ามาไว้เป็นแฟ้มข้อมูลตามชื่อที่ผู้ใช้ระบุ

3.1.2 การอ่านแฟ้มข้อมูลขึ้นมา

แอพพลิเคชันโปรแกรมจะถามชื่อของแฟ้มข้อมูลที่ต้องการให้แสดง สำหรับรายละเอียดต่างๆ จะปรากฏในลักษณะเดียวกับตอนที่ป้อนข้อมูลเข้ามาเก็บไว้ในแฟ้มข้อมูล

3.1.3 การอ่านแฟ้มข้อมูลเพื่อคำนวณหาเส้นทางการเดินทางที่ดีที่สุด

ในส่วนนี้ ส่วนที่ใช้ในการติดต่อกับผู้ใช้จะผ่านทางหน้าจอของระบบ X Window ซึ่งเป็นการแสดงผลทางด้านกราฟิก ข้อมูลตารางของสายการบินที่ใช้ในการคำนวณจะถูกเก็บอยู่ในฐานข้อมูลของระบบ การถามชื่อของแฟ้มข้อมูลจึงถูกละไว้ ซึ่งผู้ใช้สามารถแก้ได้ในภายหลัง ผู้ใช้สามารถเลือกได้ว่าต้องการพิจารณาจากเกณฑ์ในข้อใด

3.1.3.1 เกณฑ์ในเรื่องของระยะทาง

แอพพลิเคชันโปรแกรมจะต้องหาเส้นทางการเดินทางจากเมืองต้นทางไปยังเมืองปลายทางที่มีระยะทางโดยรวมสั้นที่สุด แม้ว่าจะไม่มีเที่ยวบินตรงจากเมืองต้นทางไปยังเมืองปลายทางก็ตาม

3.1.3.2 เกณฑ์ในเรื่องของราคา

แอพพลิเคชันโปรแกรมจะต้องหาเส้นทางการเดินทางจากเมืองต้นทางไปยังเมืองปลายทางที่มีราคาโดยรวมถูกที่สุด แม้ว่าในการบินนั้นอาจจะต้องเปลี่ยนเที่ยวบินหลายเที่ยว แวะจอดหลายเมืองก็ตาม

3.1.3.3 เกณฑ์ในเรื่องของเวลา

แอพพลิเคชันโปรแกรมจะต้องหาเส้นทางการเดินทางจากเมืองต้นทางไปยังเมืองปลายทางที่มีเวลารวมน้อยที่สุด เวลารวมในที่นี้จะหมายความถึง เวลารวมที่ใช้ในการบินจริงและเวลารวมที่ใช้ในการรอ ไม่ว่าจะรอเพื่อขึ้นเที่ยวบินหรือรอเพื่อเปลี่ยนเที่ยวบินในการบินต่อ ดังนั้นสำหรับในส่วนผู้ใช้จะต้องมีการระบุข้อมูลเพิ่มเกี่ยวกับ วันและเวลาที่ต้องการออกเดินทางด้วย

3.2 การออกแบบโครงสร้างข้อมูล

3.2.1 พิจารณาว่า จำนวนเมืองในตารางควรจะเป็นค่าคงที่ และขนาดของตารางควรจะมีการจองเนื้อที่ไว้แน่นอนในหน่วยความจำ ดังนั้นจึงออกแบบให้มีการเก็บจำนวนเมืองเป็นตัวแปรค่าคงที่ และสร้างเป็นตารางที่มีโครงสร้างข้อมูลแบบอาร์เรย์สองมิติ

3.2.2 ระยะทางจากเมืองหนึ่งไปยังอีกเมืองหนึ่งควรจะรวมอยู่ในตารางด้วย เนื่องจากในการคำนวณหาเส้นทาง เราสามารถทราบได้ทันทีว่า จะมีเที่ยวบินจากเมืองนี้ไปยังเมืองนั้นหรือไม่ โดยพิจารณาจากการที่มีระยะทางเป็นสำคัญ เพราะถ้าเมืองใดมีระยะทางถึงกันเป็นศูนย์ย่อมแสดงว่า ไม่มีเที่ยวบินใดที่บินตรงจากเมืองนี้ไปยังเมืองนั้น

3.2.3 พิจารณาว่า ถ้ามีเที่ยวบินที่บินตรงจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง แสดงว่าอย่างน้อยต้องมีหนึ่งวันในสัปดาห์ที่มีข้อมูลของเที่ยวบินนั้นๆ เก็บไว้ ดังนั้น ควรจะต้องออกแบบให้มีการเก็บพอยน์เตอร์ที่ชี้ไปยังโครงสร้างข้อมูลที่เก็บข้อมูลของเที่ยวบินไว้ โดยค่าพอยน์เตอร์นี้ควรจะเก็บไว้ด้วยกันกับค่าระยะทางในตาราง เพราะเป็นการทดสอบความถูกต้องด้วยว่า ถ้าคู่ลำดับใดในตารางมีค่าระยะทาง ก็ต้องมีพอยน์เตอร์ด้วย และถ้าคู่ลำดับใดในตารางมีค่าระยะทางเป็นศูนย์ พอยน์เตอร์ก็ควรจะมีค่าเป็น NULL ด้วย

3.2.4 พิจารณาว่า ในการเก็บรายละเอียดต่างๆ เกี่ยวกับเที่ยวบินหนึ่งๆ ควรเก็บไว้รวมกัน เพื่อความง่ายแก่การเข้าใจและนำไปใช้งาน ดังนั้นจึงออกแบบโครงสร้างข้อมูลในการเก็บเป็นแบบข้อมูลโครงสร้างที่ประกอบด้วยส่วนต่างๆ ดังนี้

3.2.4.1 หมายเลขประจำเครื่องบิน เป็นข้อมูลชนิดอักขระ จำนวน 5 อักขระ

3.2.4.2 เวลาออกของเครื่องบิน เป็นข้อมูลชนิดอักขระ จำนวน 5 อักขระ

3.2.4.3 เวลาถึง เป็นข้อมูลชนิดอักขระ จำนวน 5 อักขระ

3.2.4.4 เวลารวมที่ใช้ในการบินจริง เป็นข้อมูลชนิดตัวเลข

3.2.4.5 ราคา เป็นข้อมูลชนิดตัวเลข

3.2.4.6 พอยน์เตอร์ที่ชี้ไปยังโครงสร้างข้อมูลแบบเดียวกันนี้ ในกรณีที่มีเที่ยวบินเที่ยวต่อไป

3.2.5 พิจารณาว่า ในการอ้างถึงแต่ละเที่ยวบินควรจะอ้างตามวันในหนึ่งสัปดาห์ ดังนั้นจึงออกแบบให้มีโครงสร้างข้อมูลอีกหนึ่งตัว เพื่อนำมาเก็บเที่ยวบินแยกเป็นวันต่างๆ ของการบินจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง

3.3 โครงสร้างข้อมูล

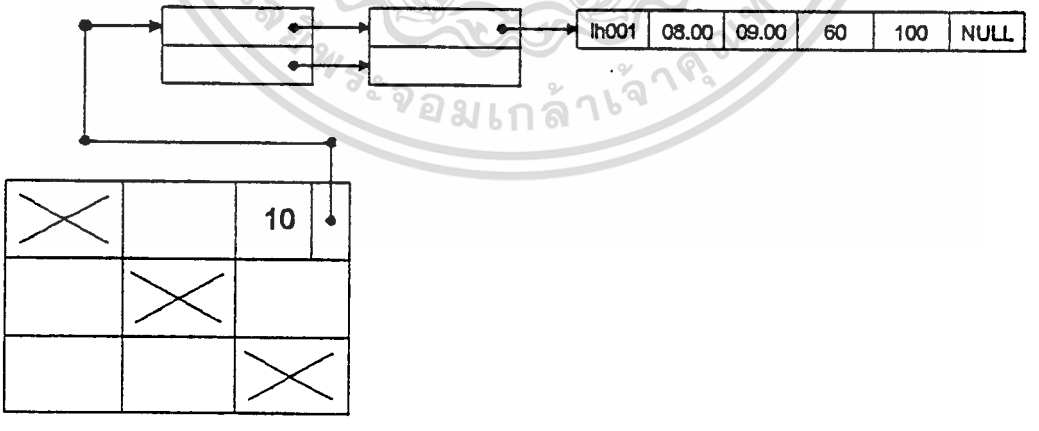
หลังจากออกแบบโครงสร้างข้อมูลเพื่อใช้เก็บข้อมูลต่างๆ ที่เกี่ยวข้องกับการเดินทางโดยเครื่องบิน เราสามารถแบ่งโครงสร้างข้อมูลหลักได้เป็น 3 โครงสร้างด้วยกัน ดังนี้

3.3.1 ตารางโครงสร้างข้อมูลแบบอาร์เรย์สองมิติ ใช้เก็บค่าระยะทางระหว่างเมืองหนึ่งไปยังอีกเมืองหนึ่ง และเก็บพอยน์เตอร์ที่ชี้ไปยังโครงสร้างข้อมูลที่มีชื่อว่า struct schedule_struct ซึ่งถ้าคู่ลำดับใดในตารางมีค่าระยะทางเป็น 0 หมายความว่า ไม่มีเที่ยวบินจากเมืองนี้ไปยังอีกเมืองหนึ่งและพอยน์เตอร์ที่ชี้ไปยัง struct schedule_struct จะมีค่าเป็น NULL

3.3.2 โครงสร้างข้อมูลที่มีชื่อว่า struct schedule_struct ประกอบด้วยโครงสร้างข้อมูลที่มีชื่อว่า struct flight_struct จำนวนเท่ากับจำนวนวันในหนึ่งสัปดาห์ที่มีเที่ยวบินบินจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง

3.3.3 โครงสร้างข้อมูลที่มีชื่อว่า struct flight_struct ใช้เก็บรายละเอียดต่างๆ ของเที่ยวบินหนึ่งๆ ที่บินจากเมืองหนึ่งไปยังอีกเมืองหนึ่ง เช่น หมายเลขประจำเครื่องบิน, เวลาออกของเครื่องบิน, เวลาถึง, เวลารวมที่ใช้ในการบินจริง, ราคาของเที่ยวบินนั้นและพอยน์เตอร์ที่ชี้ไปยังเที่ยวบินเที่ยวต่อไป ซึ่งถ้าไม่มีเที่ยวบินเที่ยวต่อไปแล้ว พอยน์เตอร์นี้จะมีค่าเป็น NULL

โครงสร้างข้อมูลทั้งสามโครงสร้าง สามารถแสดงให้เห็นถึงการใช้งานร่วมกันได้ดังรูป 3.1



รูป 3.1 รูปแสดงการใช้งานร่วมกันของโครงสร้างข้อมูล 3 โครงสร้าง

3.4 การออกแบบส่วนที่ติดต่อกับผู้ใช้ (user interface)

ในการออกแบบส่วนที่ติดต่อกับผู้ใช้นั้น เราควรพิจารณาสิ่งต่างๆ ดังนี้

ให้ส่วนนั้นดูง่าย

ให้มีการจับคู่ (map) ส่วนต่างๆ กับวัตถุโดยตรง

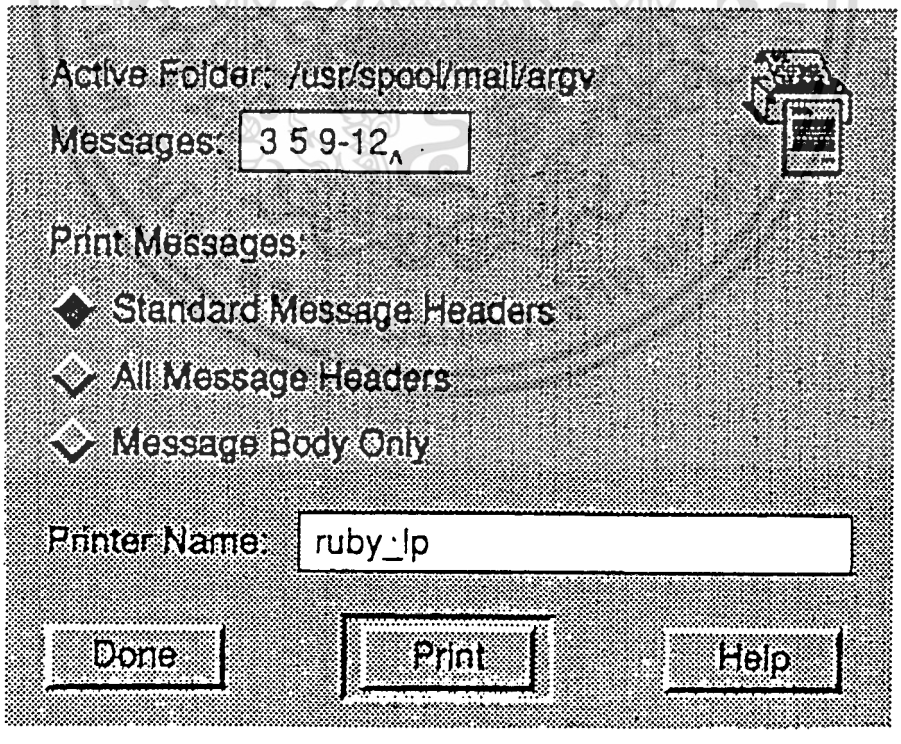
หมายความว่า เราควรใช้วิดเจต ฟอนต์ สี เมนู อย่างเหมาะสม มิฉะนั้นจะทำให้ผู้ใช้มีปัญหาในการแยกแยะข้อมูลที่สำคัญในโปรแกรม

3.5 การเขียนส่วนที่ติดต่อกับผู้ใช้

การเขียนโปรแกรมแท้จริงแล้ว เป็นการใช้ภาษาซีโดยเรียกฟังก์ชันไลบรารีของ Xlib Xt และโมทีฟนั่นเอง การสร้างวิดเจตใหม่ก็ทำได้โดยใช้วิดเจตที่มีอยู่แล้วเป็นพื้นฐาน โดยนำวิดเจตเหล่านี้มารวมกัน การเขียนส่วนที่ติดต่อกับผู้ใช้นั้นจะต้องสร้างวิดเจตย่อยต่างๆ โดยนำมาเรียงลำดับให้ถูกต้อง เริ่มจากวิดเจตที่เป็นซุปเปอร์คลาส (super class) หรือ วิดเจตแม่ (parent widget) ก่อน แล้วจึงกำหนดวิดเจตที่อยู่ในคลาสถัดๆ ลงมาเป็นวิดเจตลูก หรือวิดเจตสร้างวินโดว์ที่มีลักษณะดังรูป 3.2 เราต้องทำตามขั้นตอนดังนี้:

3.5.1 แยกส่วนประกอบในวินโดว์เป็นวิดเจตย่อย

3.5.2 ศึกษาลักษณะการจัดเรียงของวิดเจตว่า ควรจัดโครงสร้างของวิดเจตอย่างไรให้ได้ตามที่ต้องการ แต่ละวิดเจตต้องมีวิดเจตแม่อะไรบ้าง และสภาวะของวิดเจตต้องเป็นอย่างไร



รูป 3.2 ตัวอย่างหน้าจอของส่วนที่ติดต่อผู้ใช้

3.5.3 วางโครงสร้างของวิดเจตและกำหนดค่าตัวแปรต่างๆ เช่น ชื่อวิดเจต ให้ครบ

3.5.4 สร้างวิดเจตที่ต้องการตามลำดับขั้น โดยสร้างวิดเจตที่อยู่ในคลาสที่ใหญ่ที่สุดก่อน เช่น shell widget เป็นต้น

3.5.5 สำหรับวิดเจตที่สร้างแต่ละตัวจะต้องมีการกำหนดให้อยู่ในความควบคุมของวิดเจตแม่โดยใช้คำสั่ง Xt Manage Child

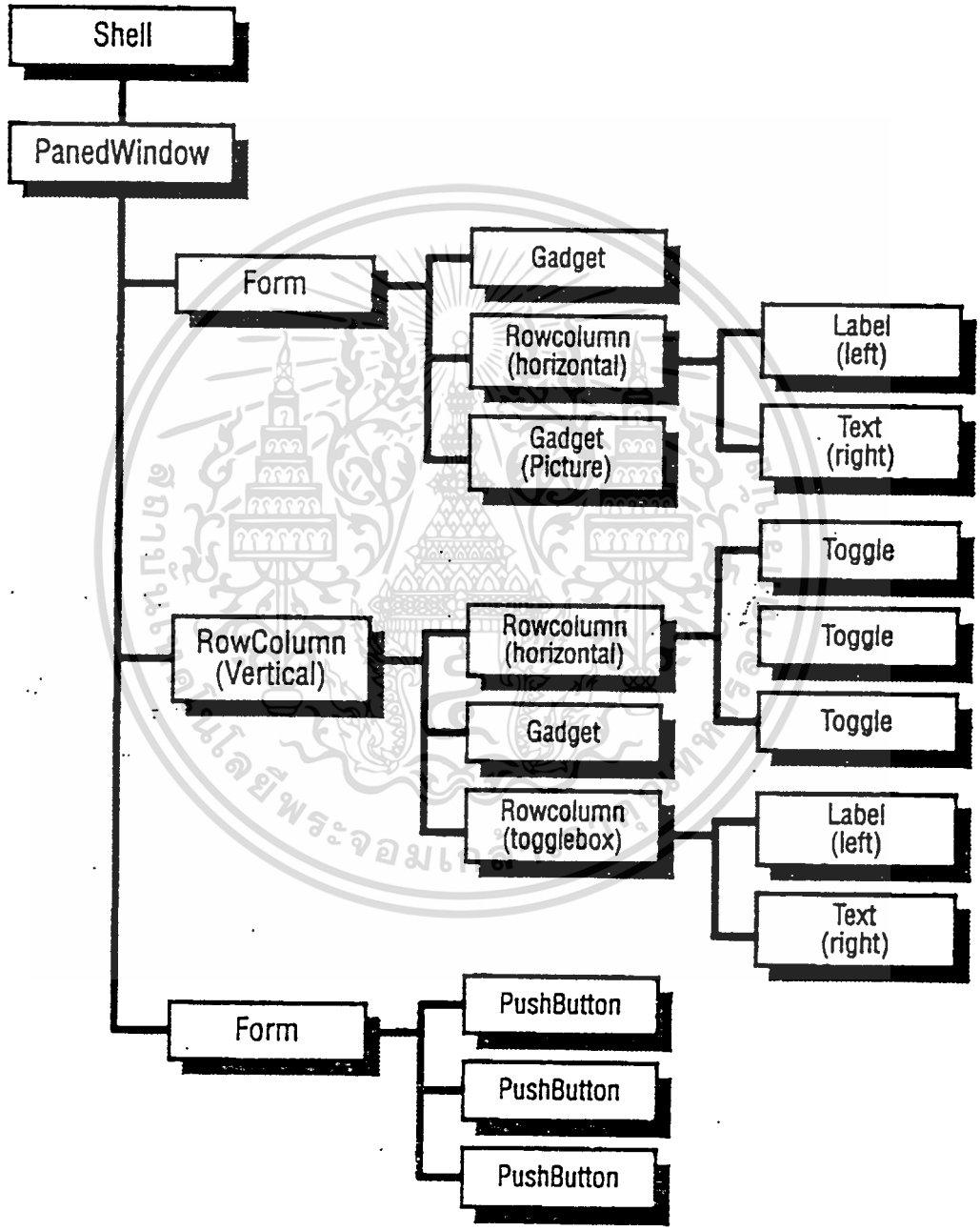
3.5.6 เรียกฟังก์ชัน Mainloop เพื่อให้โปรแกรมทำการรอ event ต่างๆ และจัดการทำงาน

3.5.7 กำหนด Callback สำหรับวิดเจตแต่ละตัว โดย Callback เป็นฟังก์ชันที่จะทำเวลาวิดเจตนั้นได้รับ event ที่กำหนดไว้ว่าจะรับ เมื่อทำ Callback เรียบร้อยแล้วการควบคุมก็จะกลับไปที่ Mainloop เพื่อรอ event ต่อไป

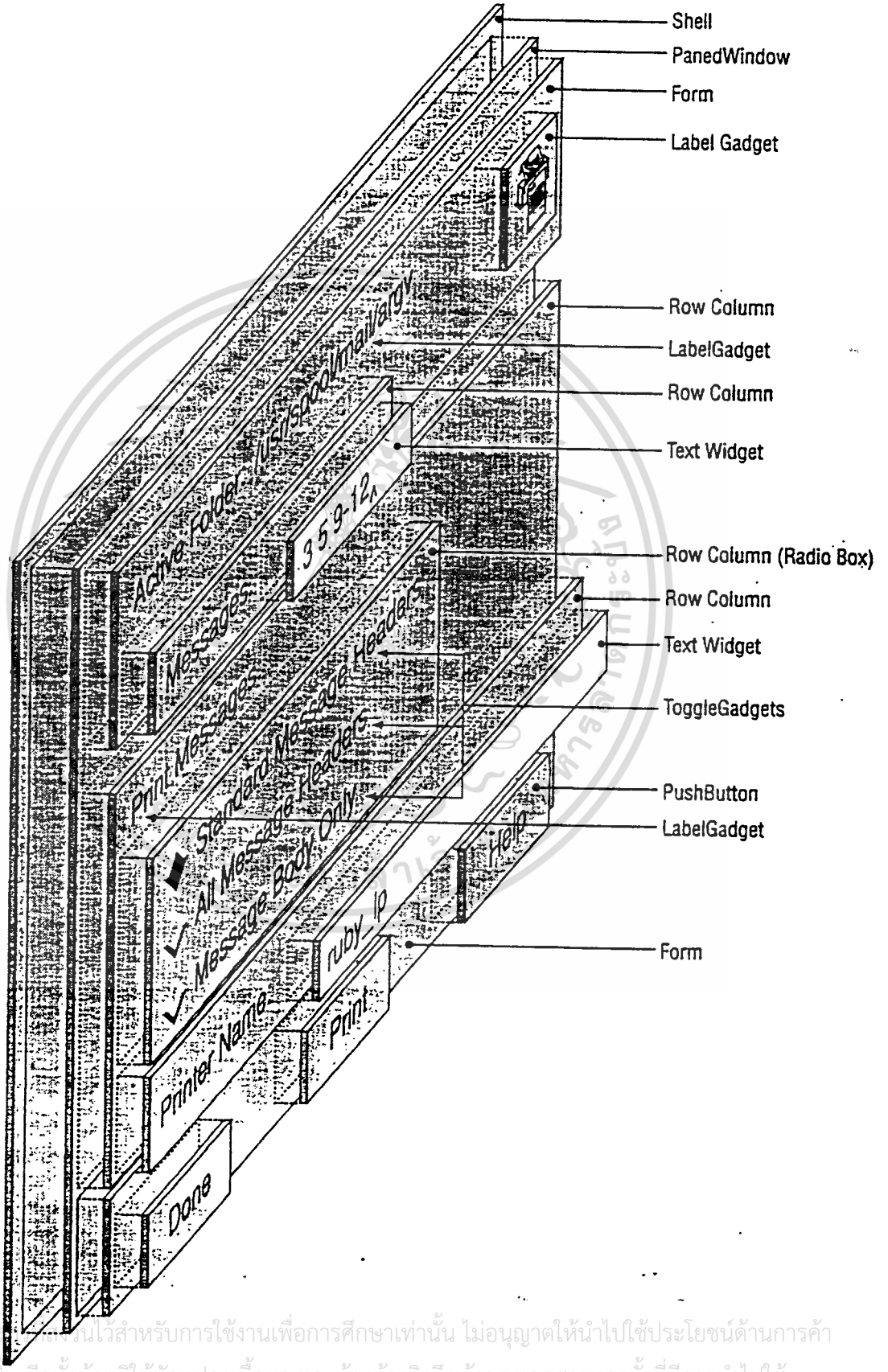
3.5.8 กำหนดค่าทรัพยากรต่างๆ ของวิดเจตแต่ละตัวในแฟ้มข้อมูลโปรแกรมมาตรฐาน (Application Default File) ซึ่งแฟ้มข้อมูลนี้จะถูกนำมากำหนดค่าทรัพยากรต่างๆ เช่น ตำแหน่งของวินโดว์ สี ขนาดความกว้าง ความสูง การเปลี่ยนค่าในแฟ้มข้อมูลนี้จะส่งผลให้รูปร่างหน้าตาของโปรแกรมเปลี่ยนไปด้วย

จากขั้นตอนที่กล่าวมา เราจะได้โครงสร้างวิดเจตดังรูป 3.3 และรูป 3.4 แสดงการจัดเรียงของวิดเจตต่างๆ จะเห็นได้ว่า เราสามารถสร้างโปรแกรมที่ซับซ้อนได้โดยใช้วิดเจตพื้นฐาน

รูป 3.3 โครงสร้างการจัดเรียงวิดเจตในโปรแกรมตัวอย่าง



รูป 3.4 รูปจำลองการจัดเรียงวิดเจตในโปรแกรมตัวอย่าง



3.6 การพัฒนาส่วนติดต่อกับผู้ใช้

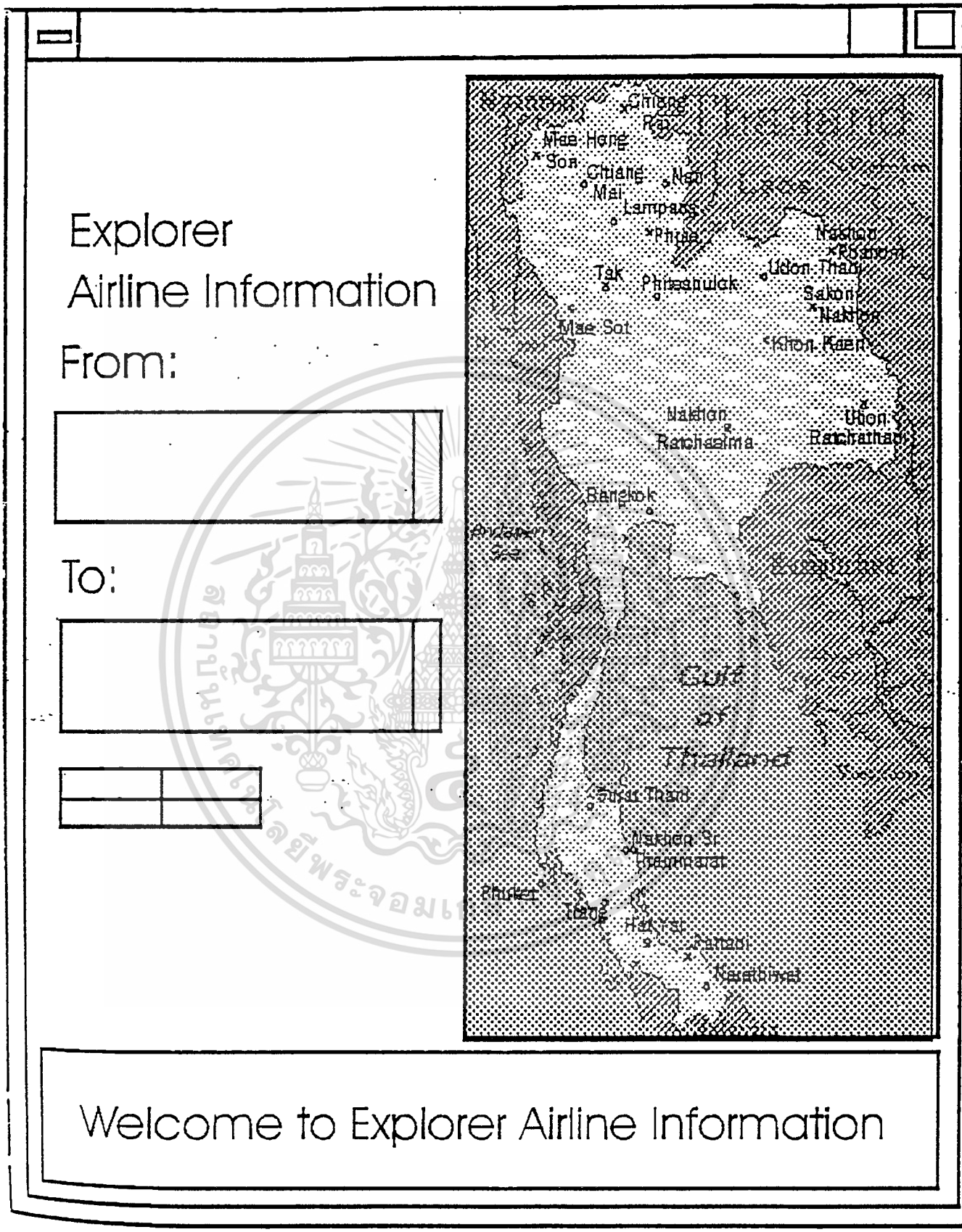
ส่วนติดต่อกับผู้ใช้ในโครงงานนี้ เราจะเรียกในชื่อว่า Explorer สำหรับขั้นตอนในการเขียนส่วนติดต่อกับผู้ใช้บนโมทีฟที่กล่าวไปข้างต้นนั้น สามารถนำมาใช้เพื่อออกแบบหน้าจอของ Explorer ดังรูปที่ 3.5 ดังนี้

3.6.1 ทำการออกแบบลักษณะหน้าจอและการติดต่อสื่อสารกับผู้ใช้ โดยกำหนดให้ผู้ใช้ไม่มีโอกาสในการป้อนข้อมูลจากแป้นพิมพ์ แต่ใช้การป้อนข้อมูลเข้ามาจากวิดเจตต่างๆ เช่น scrolled list, slidebar และ pushbutton เป็นต้น ซึ่งการทำเช่นนี้จะช่วยลดข้อผิดพลาดที่เกิดขึ้นจากการป้อนข้อมูลเข้าของผู้ใช้ได้มาก และไม่ต้องพัฒนาโปรแกรมในส่วนที่เกี่ยวข้องกับการตรวจสอบข้อผิดพลาด สำหรับการแสดงผลลัพธ์จะใช้แผนที่ประเทศไทยแสดงผลในส่วนของ drawing area label และ text area เพื่อสื่อสารกับผู้ใช้ เมื่อผู้ใช้เลือกเมืองต้นทางและเมืองปลายทาง เส้นทางการเดินทางจะปรากฏบนแผนที่และรายละเอียดต่างๆ เช่น เที่ยวบิน ระยะทางและเวลาจะปรากฏบน message area

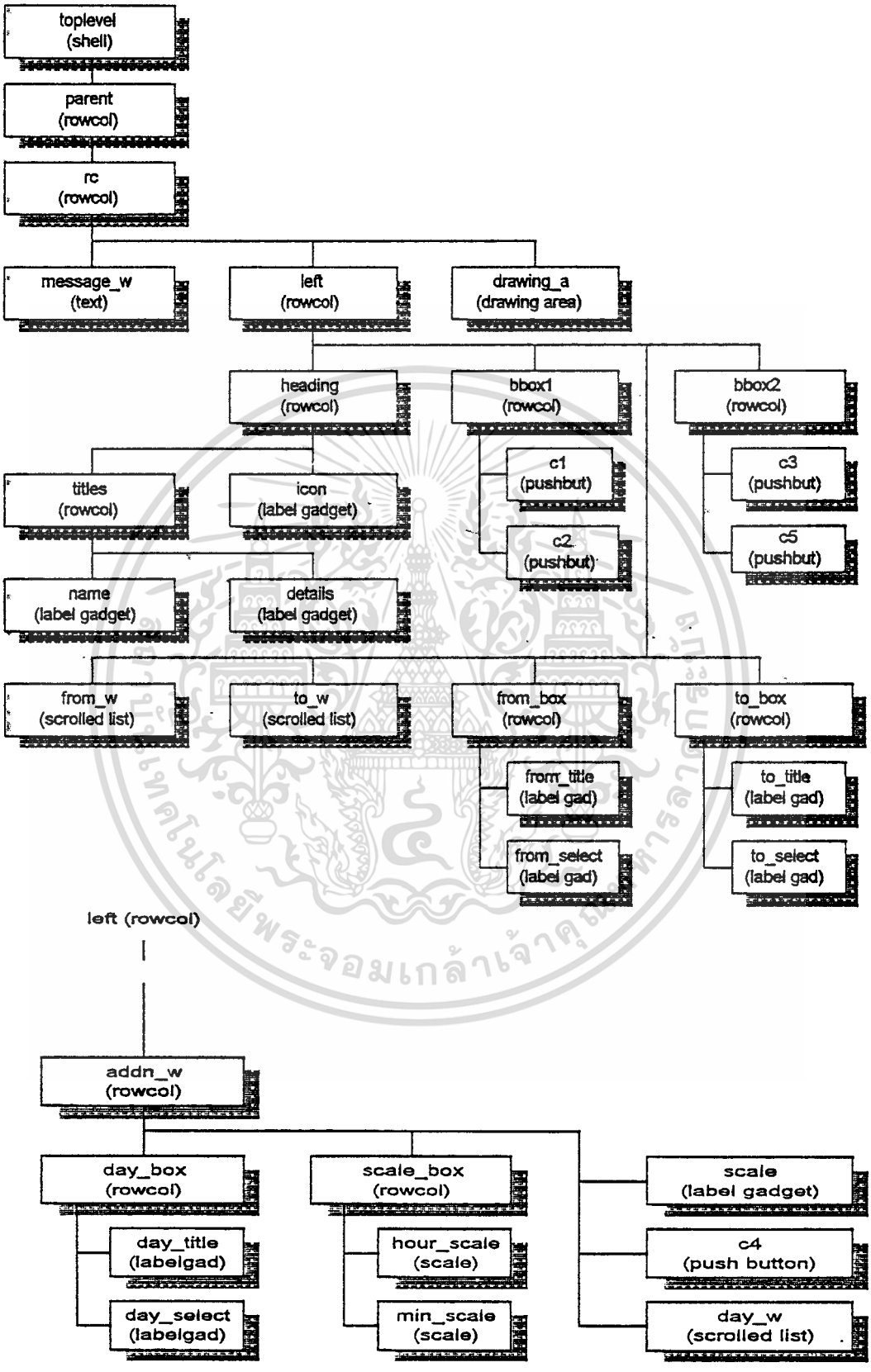
3.6.2 หลังจากรู้ว่ามีวิดเจตพื้นฐานอะไรบ้างที่ต้องนำมาเป็นส่วนประกอบพื้นฐานของโปรแกรม สิ่งต่อไปที่ต้องทำก็คือ การจัดเรียงวิดเจตเหล่านี้บนหน้าจอ การจัดเรียงนั้นจะต้องคำนึงถึงการติดต่อสื่อสารกับผู้ใช้ที่ชัดเจน ลักษณะการจัดเรียงวิดเจตใน Explorer นั้นเป็นดังรูปที่ 3.6 สำหรับวิดเจตบางตัวที่ไม่จำเป็นต้องใช้งานตลอดเวลา เราสามารถจัดให้มีการปรากฏบนหน้าจอก็ต่อเมื่อมีการใช้งานที่เกี่ยวข้องกับมันเท่านั้น เช่น เมื่อมีการกดเลือก time ก็จะมีวิดเจตปรากฏขึ้นมาอีกชุดหนึ่งเพื่อให้ผู้ใช้ป้อนวันและเวลาที่ต้องการออกเดินทาง และเมื่อป้อนเสร็จก็ให้กดเลือก done แล้ววิดเจตชุดนี้ก็จะหายไป

3.6.3 หลังจากที่เราได้จัดเรียงวิดเจตดังรูปที่ 3.6 แล้ว เราต้องกำหนดโครงสร้างของวิดเจตโดยเริ่มต้นจากวิดเจตระดับบนสุด ได้แก่ top level shell แล้วไล่ลงมา วิดเจตควบคุมมีบทบาทสำคัญในการเรียงวิดเจตเหล่านี้อย่างเหมาะสม และจากรูปที่ 3.6 แสดงโครงสร้างของวิดเจตสำหรับโปรแกรม จะเห็นได้ว่าการจัดเรียงวิดเจตส่วนใหญ่ใน Explorer จะอยู่ในแนวนอนและแนวตั้ง

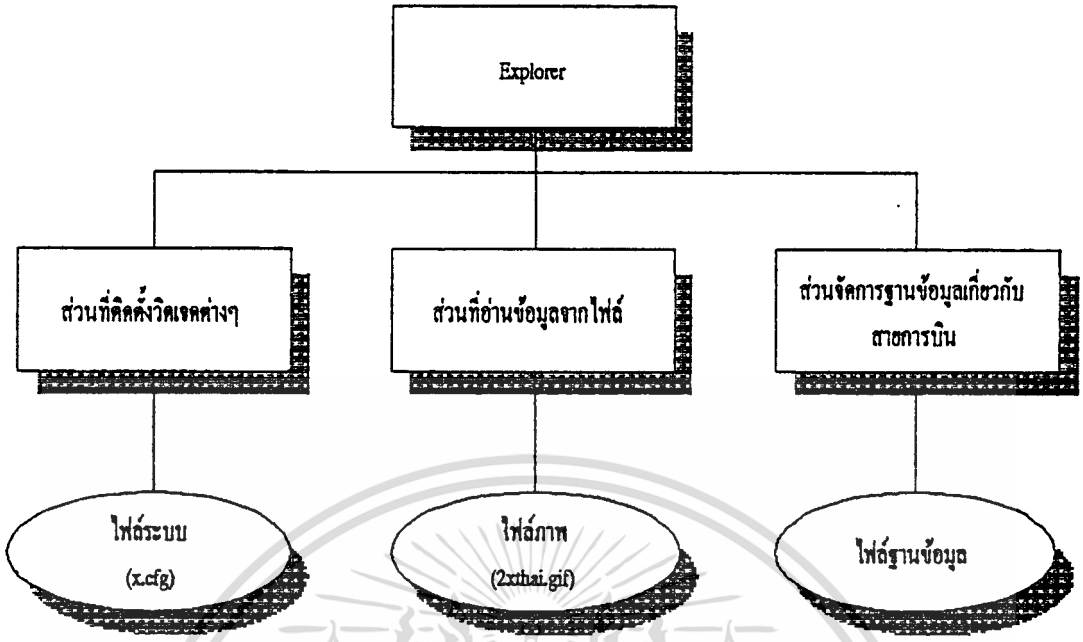
3.6.4 เมื่อมีการกำหนดโครงสร้างวิดเจตต่างๆ แล้ว ก็ทำการกำหนด callback สำหรับวิดเจตต่างๆ เช่น callback สำหรับ pushbutton เมื่อมีการกดปุ่ม โดยการกำหนด callback นี้จะต้องสอดคล้องกับวัตถุประสงค์ของโปรแกรม เนื่องจากเป็นช่องทางที่เปิดให้เราสามารถกำหนดลักษณะการตอบสนองของโปรแกรมได้ สำหรับ Explorer จะมี callback อยู่ 5 ตัวสำหรับวิดเจตต่างๆ



รูปที่ 3.5 หน้าจอแสดงโปรแกรม Explorer



รูปที่ 3.6 แสดงโครงสร้างวิดเจตในโปรแกรม Explorer



รูปที่ 3.7 ส่วนประกอบใน Explorer

3.6.5 ลงมือเขียนโปรแกรมจริงๆ ซึ่งการกำหนดวิดเจตนั้นเราจะทำที่ละขั้นตอนแต่เป็นไปตามโครงสร้างที่วางไว้ โดยเริ่มจากวิดเจตระดับบนก่อนแล้วไล่ลงมา เราจะกำหนดฟังก์ชันการ callback ควบคู่กับการกำหนดวิดเจตนั้นด้วย เมื่อทำการรันโปรแกรมจะต้องมีการเปลี่ยนค่าต่างๆ บ้างเพื่อให้วิดเจตต่างๆ เรียงกันอย่างเหมาะสม

3.6.6 กำหนดฟังก์ชันอื่นที่เกี่ยวข้องกับโปรแกรม ฟังก์ชันเหล่านี้ส่วนใหญ่จะถูกเรียกโดยฟังก์ชัน callback อีกทีหนึ่ง ฟังก์ชันเหล่านี้จะเกี่ยวข้องกับการติดตั้งระบบ การจบการทำงาน การอ่านข้อมูลจากแฟ้มข้อมูล เป็นต้น

3.6.7 รวบรวมส่วนต่างๆ ในโปรแกรมเข้าด้วยกันเพื่อตรวจสอบการทำงานและเพื่อหาข้อผิดพลาด

3.7 การรวบรวมส่วนต่าง ๆ ในส่วนติดต่อกับผู้ใช้

Explorer ประกอบด้วยส่วนสำคัญต่างๆ ดังรูปที่ 3.7 ซึ่งมีหลักการสำคัญดังนี้

3.7.1 ส่วนที่ติดตั้งวิดเจต จะเป็นส่วนที่เรียกให้ส่วนอื่นใน Explorer ทำงาน โดยผ่านการ activate ฟังก์ชัน callback

3.7.2 การส่งค่าระหว่างส่วนต่างๆ นั้น ทำได้โดยการผ่านค่าตัวแปรเป็นพารามิเตอร์หรือการใช้ตัวแปรโกลบอล

3.7.3 ข้อมูลในส่วนที่ผู้ใช้สามารถเปลี่ยนแปลงได้ จะถูกกำหนดค่าในแฟ้มข้อมูลที่ชื่อว่า x.cfg ซึ่งถือเป็นแฟ้มข้อมูล configuration ของระบบ

3.7.4 เราใช้ดัชนีในการระบุตำแหน่งของเมืองที่ถูกเลือกขึ้นมา โดยนับจาก 0, 1, 2,...

3.7.5 ส่วนอื่นๆ ที่อยู่แยกเป็นแฟ้มข้อมูล เราจะใช้คำสั่ง #include เพื่อรวมแฟ้มข้อมูลเหล่านี้เข้าด้วยกัน

บทที่ 4

การทดลองและผลการทดลอง

แอพพลิเคชันโปรแกรมนี้ สามารถแบ่งส่วนประกอบหลักออกเป็น 3 ส่วนด้วยกัน ดังนี้

4.1 ส่วนของการรับข้อมูล และเขียนเก็บลงในแฟ้มข้อมูล

4.2 ส่วนของการอ่านแฟ้มข้อมูลขึ้นมาแสดง

4.3 ส่วนของการอ่านแฟ้มข้อมูลขึ้นมาเพื่อใช้ในการคำนวณหาเส้นทาง

สำหรับรายละเอียดต่างๆ ในแต่ละส่วน มีดังต่อไปนี้

4.1 การรับข้อมูลและเก็บลงแฟ้มข้อมูล

ในส่วนนี้ จะมีฟังก์ชันหลักเพียงฟังก์ชันเดียว ได้แก่ int main( ) ทำหน้าที่รับข้อมูลเข้ามาจากการป้อนทางแป้นพิมพ์ จองเนื้อที่ในหน่วยความจำขนาดเท่าที่จำเป็นต้องใช้และเขียนข้อมูลทั้งหมดเก็บไว้ในแฟ้มข้อมูล ซึ่งมีรายละเอียดต่างๆ ในโปรแกรม ดังนี้

ถามชื่อของแฟ้มข้อมูลที่ต้องการจะเก็บ
เปิดแฟ้มข้อมูลตามชื่อที่ป้อนเข้ามาเป็นแฟ้มข้อมูลแบบไบนารี่
FOR แต่ละพิกัด m ในแนวดิ่ง
  FOR แต่ละพิกัด n ในแนวนอน
    IF ค่าของพิกัดแนวดิ่งไม่เท่ากับค่าของพิกัดแนวนอน คือ m != n
      ถามระยะทางจากเมือง m ไปยังเมือง n
      IF ระยะทางที่ป้อนเข้ามาไม่เป็นศูนย์
        เก็บค่าระยะทางลงในตาราง table
        ใช้พอยน์เตอร์ table_ptr จองเนื้อที่ขนาดเท่ากับ schedule_struct
        เก็บค่าระยะทางและพอยน์เตอร์ลงในแฟ้มข้อมูล
        FOR แต่ละ day ในหนึ่งสัปดาห์
          ให้พอยน์เตอร์ tour_ptr ชี้ไปยังแอดเดรสของตาราง schedule [day]
          ถามว่าในวันนี้มีเที่ยวบินหรือไม่
          IF คำตอบ คือ ใช่ วันนี้มีเที่ยวบิน
            ถามข้อมูลต่างๆ เกี่ยวกับเที่ยวบิน
            ถามอีกครั้งว่าในวันนี้ยังมีเที่ยวบินอีกหรือไม่
            WHILE คำตอบ คือ ใช่ วันนี้ยังมีเที่ยวบินอีก

ให้พอยน์เตอร์ new_flight จองเนื้อที่ขนาดเท่ากับ flight_struct
ให้พอยน์เตอร์ tour_ptr->next ชี้ไปยังตำแหน่งเดียวกับ new_flight
เก็บข้อมูลต่างๆ เกี่ยวกับเที่ยวบินลงในแฟ้มข้อมูล
ให้พอยน์เตอร์ tour_ptr ชี้ไปยังตำแหน่งเดียวกับ tour_ptr->next
ถามข้อมูลต่างๆ เกี่ยวกับเที่ยวบิน
ถามอีกครั้งว่าในวันนี้ยังมีเที่ยวบินอีกหรือไม่
ENDWHILE
ให้ค่าพอยน์เตอร์เป็น NULL เพราะเป็นเที่ยวบินสุดท้าย
เก็บข้อมูลต่างๆ เกี่ยวกับเที่ยวบินสุดท้ายลงในแฟ้มข้อมูล
ENDIF
ELSE
คำตอบ คือ ไม่ใช่ วันนี้ไม่มีเที่ยวบิน ตัวแปรต่างๆ จะถูกให้ค่าพิเศษ
เก็บค่าพิเศษต่างๆ ที่ระบุว่าวันนี้ไม่มีเที่ยวบินลงในแฟ้มข้อมูล
ENDELSE
ENDFOR
ENDIF
ELSE
ระยะทางเป็นศูนย์ คือ ไม่มีเส้นทางการเดินทางจากเมือง m ไปยังเมือง n
เก็บค่าที่ระบุว่าไม่มีเส้นทางการเดินทางลงในแฟ้มข้อมูล
ENDELSE
ELSE
พิกัดแนวดิ่งเท่ากับพิกัดแนวนอน คือ m = n
เก็บค่าที่ระบุว่าไม่นำคู่ลำดับนี้มาใช้ในการคำนวณลงในแฟ้มข้อมูล
ENDELSE
ENDFOR
ENDFOR
ปิดแฟ้มข้อมูล

4.2 การอ่านแฟ้มข้อมูลขึ้นมาแสดง

ในส่วนนี้ จะมีฟังก์ชันหลักเพียงฟังก์ชันเดียว ได้แก่ int main( ) ทำหน้าที่ในการอ่านแฟ้มข้อมูลขึ้นมาเก็บในเนื้อที่หน่วยความจำที่จองไว้ขนาดเท่าที่จำเป็นต้องใช้ และแสดงผล ซึ่งมีรายละเอียดต่างๆ ในโปรแกรม ดังนี้

ถามชื่อของแฟ้มข้อมูลที่ต้องการจะอ่าน
เปิดแฟ้มข้อมูลตามชื่อที่ป้อนเข้ามาเป็นแฟ้มข้อมูลแบบไบนารี่
FOR แต่ละพิกัด m ในแนวดิ่ง
 FOR แต่ละพิกัด n ในแนวนอน
  อ่านคู่ลำดับแรกขึ้นมาจากตาราง table[m][n]
  IF ค่าของพิกัดแนวดิ่งไม่เท่ากับค่าของพิกัดแนวนอน คือ m != n
   และพอยน์เตอร์ table_ptr ไม่เท่ากับ NULL
   ให้พอยน์เตอร์ table_ptr จองเนื้อที่ขนาดเท่ากับ schedule_struct
   FOR แต่ละ day ในหนึ่งสัปดาห์
    อ่านข้อมูลเกี่ยวกับเที่ยวบินขึ้นมาเก็บไว้ที่พอยน์เตอร์ new_flight
    คัดลอกข้อมูลเกี่ยวกับเที่ยวบินมาเก็บไว้ในเที่ยวบินแรกของตาราง table[m][n]
    IF พอยน์เตอร์ new_flight->next ไม่เท่ากับ NULL
     ให้พอยน์เตอร์ schedule[day].next จองเนื้อที่ขนาดเท่ากับ flight_struct
     ให้พอยน์เตอร์ tour_ptr ชี้ไปที่เนื้อที่ใหม่ที่จองขึ้นมา
     อ่านข้อมูลเกี่ยวกับเที่ยวบินใหม่ขึ้นมาเก็บไว้ที่พอยน์เตอร์ new_flight
     คัดลอกข้อมูลเกี่ยวกับเที่ยวบินใหม่มาเก็บไว้ในเที่ยวบินต่อไป
     IF พอยน์เตอร์ new_flight->next ไม่เท่ากับ NULL
      DO
       ให้พอยน์เตอร์ tour_ptr->next จองเนื้อที่ขนาดเท่ากับ flight_struct
       ให้พอยน์เตอร์ tour_ptr ชี้ไปยังเนื้อที่ใหม่ที่จองขึ้นมา
       อ่านข้อมูลเกี่ยวกับเที่ยวบินใหม่ขึ้นมาเก็บไว้ที่พอยน์เตอร์ new_flight
       คัดลอกข้อมูลเกี่ยวกับเที่ยวบินใหม่มาเก็บไว้ที่เที่ยวบินต่อไป
      WHILE tour_ptr->next เท่ากับ NULL คือเป็นเที่ยวบินสุดท้าย
     ENDIE
     ให้พอยน์เตอร์ tour_ptr->next มีค่าเป็น NULL เพราะเป็นเที่ยวบินสุดท้าย
    ENDIE

ELSE

มีเที่ยวบินเดียวในวันนี้จึงให้พอยน์เตอร์ schedule[day].next เป็น NULL

ENDFOR

ENDIF

ENDFOR

ENDFOR

ปิดแฟ้มข้อมูล

หลังจากนั้น นำข้อมูลในตารางมาแสดงผล ซึ่งการแสดงผลจะเริ่มต้นไล่ตามตารางในแนวนอนก่อน และในแต่ละคู่ลำดับจะไล่เข้าไปในแต่ละวันในหนึ่งสัปดาห์ว่ามีเที่ยวบินอะไรบ้าง แต่ละเที่ยวบินมีข้อมูลอะไรบ้าง เช่น หมายเลขประจำเครื่องบิน เวลาออกของเครื่องบิน เวลาถึง เวลารวมที่ใช้ในการบินจริงและราคา เป็นต้น ถ้าวันใดไม่มีเที่ยวบิน ก็จะระบุไว้ว่า วันนี้ไม่มีเที่ยวบิน เป็นต้น

4.3 การอ่านแฟ้มข้อมูลขึ้นมาคำนวณ

ในส่วนนี้ จะเป็นส่วนที่เชื่อมต่อกับระบบ X Window ฟังก์ชันการทำงานหลัก int main( ) จะถูกละไว้ เพราะส่วนนี้จะถูกเรียกในระบบ X Window อีกที อย่างไรก็ตาม ในส่วนนี้ประกอบด้วย ฟังก์ชันเพื่อทำการอ่านแฟ้มข้อมูลขึ้นมาเก็บไว้ในเนื้อที่หน่วยความจำขนาดเท่าที่จำเป็นต้องใช้ และเรียกฟังก์ชันอื่นๆ เพื่อทำการคำนวณหาเส้นทางการเดินทางตามเกณฑ์ที่ผู้ใช้เลือก ดังนั้นในส่วนนี้ประกอบด้วย

4.3.1 ฟังก์ชัน void Retrieve_data (void)

4.3.2 ฟังก์ชัน void Cal_distance (void)

4.3.3 ฟังก์ชัน void Cal_cost (void)

4.3.4 ฟังก์ชัน void Cal_time (void)

สำหรับรายละเอียดในแต่ละฟังก์ชัน มีดังต่อไปนี้

4.3.1 **ฟังก์ชัน** void Retrieve_data (void)

ทำหน้าที่ในการอ่านแฟ้มข้อมูลขึ้นมาเก็บไว้ในเนื้อที่หน่วยความจำขนาดเท่าที่จำเป็นต้องใช้ รายละเอียดต่างๆ ในฟังก์ชันนี้จึงเหมือนกับการอ่านข้อมูลขึ้นมาแสดงแต่ตัดส่วนของการแสดงผลออกไปเท่านั้น

4.3.2 **ฟังก์ชัน** void Cal_distance (void)

ทำหน้าที่ในการคำนวณหาเส้นทางการเดินทางโดยพิจารณาจากระยะทางสั้นที่สุดเป็นเกณฑ์ การทำงานของฟังก์ชันนี้ เริ่มต้นจากการรับค่าว่าเมืองใดเป็นเมืองต้น

ทางและเมืองใดเป็นเมืองปลายทางจากระบบ X Window (source, distination เป็นตัวแปรซึ่งนำมาจากแฟ้มข้อมูลของระบบ X Window) หลังจากนั้นก็เริ่มทำการคำนวณโดยใช้อัลกอริธึมของ Dijkstra ดังที่ได้อธิบายไว้ข้างต้น และในตอนท้ายของฟังก์ชัน จะมีการนำค่าในตัวแปร p_found[ ] มาทำการกลับค่าจากหลังมาหน้าเพื่อใช้แสดงชื่อเมืองที่จะต้องผ่านในเส้นทางไปยังเมืองปลายทาง โดยการแสดงผลจะเรียกฟังก์ชันในระบบ X Window ซึ่งได้แก่ draw_line(เมืองต้นทาง, เมืองที่จะผ่านหรือเมืองปลายทาง) เป็นการลากเส้นเชื่อมเมืองต่างๆ ที่ต้องผ่านและ disp_message(buf) เป็นการแสดงข้อความ

4.3.3 ฟังก์ชัน void Cal_cost (void)

ทำหน้าที่ในการคำนวณหาเส้นทางการเดินทางโดยพิจารณาจากราคาถูกที่สุดเป็นเกณฑ์ การทำงานของฟังก์ชันก็คล้ายๆ กับฟังก์ชัน void Cal_distance (void) แต่จะมีฟังก์ชันย่อย srch_mincost (int i, int j, struct flight_struct *find_flight) เพิ่มเข้ามาด้วย ฟังก์ชันvoid Cal_cost (void) นี้เริ่มต้นจากการรับค่าว่าเมืองใดเป็นเมืองต้นทางและเมืองใดเป็นเมืองปลายทาง หลังจากนั้นก็เรียกใช้ฟังก์ชัน srch_mincost (int i, int j, struct flight_struct *find_flight) เพื่อทำการหาเที่ยวบินของแต่ละคู่ลำดับในตารางที่มีราคาถูกที่สุด แล้วคัดลอกมาเก็บไว้ในตาราง cost_table สำหรับการคำนวณหาเส้นทางการเดินทางก็ประยุกต์ใช้อัลกอริธึม Dijkstra เช่นเดียวกับการหาระยะทาง เพียงแต่เกณฑ์ในการพิจารณาจะใช้ราคาแทนระยะทาง และการค้นหาก็จะทำกับตาราง cost_table แทนตาราง table ที่เก็บระยะทาง

ในส่วนของฟังก์ชันย่อย srch_mincost (int i, int j, struct flight_struct *find_flight) นั้น มีรายละเอียดดังต่อไปนี้

ให้ค่าของพอยน์เตอร์ find_flight->cost เป็นตัวเลขค่ามากที่สุด
FOR แต่ละ day ในหนึ่งสัปดาห์
ให้พอยน์เตอร์ tour_ptr ชี้ไปที่ตำแหน่งของเที่ยวบินแรกของ day นั้น
IF ค่าของพอยน์เตอร์ tour_ptr->cost ไม่เป็นค่าพิเศษที่ระบุว่าวันนี้ไม่มีเที่ยวบิน
DO
IF ค่าของพอยน์เตอร์ tour_ptr->cost < ค่าของพอยน์เตอร์ find_flight->cost
เก็บค่าข้อมูลเกี่ยวกับเที่ยวบินที่พอยน์เตอร์ tour_ptr ชี้อยู่ไว้ใน find_flight
ENDIF
IF ค่าของพอยน์เตอร์ tour_ptr->next ไม่เท่ากับ NULL
ให้พอยน์เตอร์ tour_ptr ชี้ไปยังตำแหน่งของพอยน์เตอร์ tour_ptr->next
ELSE

ออกจากการวนลูป

WHILE ค่าของพอยน์เตอร์ tour_ptr->next ไม่เท่ากับ NULL

ENDIF

ENDFOR

4.3.4 ฟังก์ชัน void Cal_time (void)

ทำหน้าที่ในการคำนวณหาเส้นทางการเดินทาง โดยพิจารณาจากเวลาสั้นที่สุดเป็นเกณฑ์ การทำงานของฟังก์ชันเริ่มต้นจากการรับค่าว่าเมืองใดเป็นเมืองต้นทางและเมืองใดเป็นเมืองปลายทางรวมทั้งวันและเวลาที่ต้องการออกเดินทางด้วย ซึ่งข้อมูลเหล่านี้จะถูกส่งมาจากระบบ X Window หลังจากนั้นก็เรียกใช้ฟังก์ชัน srch_mintime (int index, int col, struct flight_struct *find_flight, int *day_flight) เพื่อทำการหาเที่ยวบินของแต่ละคู่ลำดับในตารางที่มีเวลาสั้นที่สุด แล้วคัดลอกมาเก็บไว้ในตาราง time_table

ในส่วนของตาราง time_table นี้ นอกจากจะเก็บข้อมูลต่างๆ เกี่ยวกับเที่ยวบินที่มีเวลาสั้นที่สุดแล้ว ยังเก็บวันที่ออกเดินทางของเที่ยวบินนั้นด้วย สำหรับส่วนของการคำนวณหาเส้นทางการเดินทางยังคงเหมือนเดิม คือ ประยุกต์ใช้อัลกอริธึม Dijkstra เช่นเดียวกับการหาระยะทาง เพียงแต่เกณฑ์ในการพิจารณาจะใช้เวลาแทนระยะทาง และการค้นหาก็จะทำกับตาราง time_table แทนตาราง table ที่เก็บระยะทาง และหลังจากที่สามารถหาเส้นทางจากเมืองต้นทางไปยังเมือง J เมืองหนึ่งๆ ที่ใช้เวลาในการเดินทางสั้นที่สุดแล้ว จะต้องทำการเปลี่ยนแปลงค่าในตาราง time_table ทุกๆ เมือง J ด้วย ซึ่งทำได้ดังนี้

คัดลอกเวลาถึงเมือง J ของเที่ยวบินมาเก็บไว้ในตัวแปร hour, mins แทนเวลาที่ผู้ใช้ต้องการออกเดินทาง และให้ตัวแปร present_day แทนด้วยวันที่เที่ยวบินเดียวกันนี้ถึงเมือง J หลังจากนั้นก็เรียกใช้ฟังก์ชัน srch_mintime (int index, int col, struct flight_struct *find_flight, int *day_flight) เพื่อทำการหาเที่ยวบินของแต่ละคู่ลำดับในตารางที่มีเวลาสั้นที่สุด แล้วคัดลอกมาเก็บไว้ในตาราง time_table อีกครั้งหนึ่ง

ในส่วนของฟังก์ชันย่อย srch_mintime (int index, int col, struct flight_struct *find_flight, int *day_flight) นั้น มีรายละเอียดดังต่อไปนี้

กำหนดค่าเริ่มต้นให้กับตัวแปร ดังนี้

day = 0 ใช้ตรวจสอบไม่ให้มีการวนลูปเกินจำนวนวันในหนึ่งสัปดาห์

first_day = 1 ใช้ระบุว่าสามารถหาเที่ยวบินได้ในวันที่ต้องการจะเดินทางเลย

แต่ถ้ามีค่าเป็นศูนย์ คือ จะต้องรอจนกว่าจะมีเที่ยวบิน

status_found = FALSE ใช้ระบุว่ายังหาเที่ยวบินที่ใช้เวลาน้อยไม่ได้

find_flight->total = INFINITY ให้เป็นตัวเลขค่ามากที่สุดเพื่อใช้เปรียบเทียบ

WHILE status_found มีค่าเป็นเท็จและ day มีค่าน้อยกว่าจำนวนวันในหนึ่งสัปดาห์

ให้พอยน์เตอร์ tour_ptr ชี้ไปที่ตำแหน่งของเที่ยวบินแรกในวันที่มาถึง

IF ค่าของพอยน์เตอร์ tour_ptr->total ไม่เท่ากับ NULL

IF ค่าของตัวแปร first_day = 1

ให้ค่าของตัวแปร first_day เป็นศูนย์เพื่อระบุว่าวันนี้มีเที่ยวบินจะออกเดินทาง

DO

IF มีเที่ยวบินที่จะออกหลังจากเวลาที่มาถึงเมืองนี้

หาค่าเวลารวมที่ใช้ในการบินจริงกับเวลาที่ใช้ในการรอเพื่อขึ้นเครื่องบิน

IF เวลารวมที่หาได้นี้น้อยกว่าเวลาใน find_flight->total

ให้ status_found มีค่าเป็นจริง คือ หาเที่ยวบินที่ใช้เวลารวมน้อยได้แล้ว

ให้ find_flight->total มีค่าเท่ากับเวลารวมที่หาได้

เก็บข้อมูลจากเที่ยวบินที่หาได้ไว้ใน find_flight

เก็บค่าตัวแปร present_day ไว้ในตัวแปร day_flight

ENDIF

ENDIF

ให้พอยน์เตอร์ oldtour_ptr เก็บตำแหน่งของพอยน์เตอร์ tour_ptr->next ไว้

IF ค่าของพอยน์เตอร์ tour_ptr->next ไม่เท่ากับ NULL คือ ยังมีเที่ยวบินอีก

ให้พอยน์เตอร์ tour_ptr ชี้ไปยังเที่ยวบินเที่ยวต่อไปเพื่อคำนวณค่าเวลารวม

อีก

ELSE

ออกจากการวนลูป

WHILE ค่าของพอยน์เตอร์ oldtour_ptr ไม่เท่ากับ NULL

ENDIF

ELSE

DO

หาค่าเวลารวมที่ใช้ในการบินจริงและเวลารอในการขึ้นเครื่องบินรวมทั้งจำนวน

วันที่รอเพื่อขึ้นเที่ยวบินนี้

IF เวลารวมที่หาได้นี้น้อยกว่าเวลาใน find_flight->total

ให้ status_found มีค่าเป็นจริง คือ หาเที่ยวบินที่ใช้เวลารวมน้อยได้แล้ว
ให้ find_flight->total มีค่าเท่ากับเวลารวมที่หาได้
เก็บข้อมูลจากเที่ยวบินที่หาได้ไว้ใน find_flight
เก็บค่าตัวแปร present_day ไว้ในตัวแปร day_flight
ENDIF
ให้พอยน์เตอร์ oldtour_ptr เก็บตำแหน่งของพอยน์เตอร์ tour_ptr->next ไว้
IF ค่าของพอยน์เตอร์ tour_ptr->next ไม่เท่ากับ NULL คือ ยังมีเที่ยวบินอีก
ให้พอยน์เตอร์ tour_ptr ชี้ไปยังเที่ยวบินเที่ยวต่อไปเพื่อคำนวณค่าเวลารวมอีก
ELSE
ออกจากการวนลูป
WHILE ค่าของพอยน์เตอร์ oldtour_ptr ไม่เท่ากับ NULL
ENDELSE
ENDIF
IF ค่าของ status_found เป็นเท็จ
ให้ค่าของ present_day เพิ่มไปอีกหนึ่งวันแต่ต้องไม่เกินจำนวนวันในหนึ่งสัปดาห์
ให้ตัวแปร first_day มีค่าเป็นศูนย์
เพิ่มค่า day ขึ้นเป็นวันถัดไป
ENDIF
ENDWHILE

สำหรับการหาค่าเวลารวมที่ใช้ในการบินจริงกับเวลาที่ใช้ในการรอเพื่อขึ้นเครื่องบินนั้น จะเรียกใช้ฟังก์ชัน int waiting_time (char *depart) ซึ่งจะทำการเปลี่ยนค่าในตัวแปร depart และค่าตัวแปร hour, mins เป็นค่าตัวเลขหน่วยเป็นนาที แล้วทำหน้าที่ลบค่าของตัวแปร depart ด้วยค่าของตัวแปร hour และ min เพื่อหาเวลาในการรอเพื่อที่จะออกเดินทาง แล้วส่งค่ากลับไปเพื่อที่จะนำไปรวมกับเวลาที่ใช้ในการบินจริง ได้เป็นเวลารวมที่ใช้ในการคำนวณ

สำหรับการหาค่าเวลารวมที่ใช้ในการบินจริงและเวลารอในการขึ้นเครื่องบินรวมทั้งจำนวนวันที่รอเพื่อขึ้นเที่ยวบินนี้ จะเรียกใช้ฟังก์ชัน int cross_waiting_time (char *depart, int day) ซึ่งจะหน้าที่คล้ายๆ กับฟังก์ชัน int waiting_time (char *depart) ต่างกันเพียงตอนส่งค่า

กลับไปเพื่อนำไปรวมกับเวลาที่ใช้ในการบินจริง จะต้องมีการบวกจำนวนวันที่ต้องรอเข้าไปด้วย

4.4 ฟังก์ชันที่ใช้ในส่วนติดต่อกับผู้ใช้

4.4.1 ฟังก์ชันสำหรับติดตั้งระบบ

initialize (void) เป็นฟังก์ชันที่อ่านค่าจากแฟ้มข้อมูล x.cfg เพื่อนำมากำหนดค่าให้แก่ตัวแปรแบบโกลบอล ดังนี้

placenum เก็บจำนวนเมืองที่เรากำหนดไว้ในโปรแกรม

width, high ระบุขนาดของรูปที่เก็บในแฟ้มข้อมูล 2xthai.gif ที่จะถูกนำมาแสดงบน drawing area

dest[ ] เป็นอาร์เรย์สำหรับเก็บชื่อเมืองต่างๆ ที่จะปรากฏใน scrolled list

x[ ], y[ ] ระบุคู่ลำดับที่เป็นตำแหน่งของเมืองในแผนที่ที่เราจะนำมาแสดงบน drawing area

place[ ] เป็นอาร์เรย์ที่เก็บฐานข้อมูลสำหรับแต่ละเมือง ข้อมูลเหล่านี้จะถูกนำมาแสดงเมื่อผู้ใช้กดเลือกเมืองนั้นบนแผนที่

4.4.2 ฟังก์ชัน callback

4.4.2.1 disp_select(w,w_no,cbs) เป็นฟังก์ชัน callback ของ scrolled list ทั้ง 3 ตัวใน Explorer มันจะกำหนดค่าตัวแปรโกลบอลสำหรับติดต่อกับโปรแกรมส่วนอื่น เมื่อผู้ใช้ทำการเลือกค่าๆ หนึ่งใน scrolled list

พารามิเตอร์ที่ใช้มีดังนี้

w เป็นชื่อวิดเจตที่เรียก callback

w_no เป็นลำดับของวิดเจตที่เรียก callback ในที่นี้มีค่าตั้งแต่ 1 ถึง 3 สำหรับ scrolled list แต่ละตัว w_no ช่วยให้ callback สามารถตอบสนองการทำงานสำหรับวิดเจตแต่ละตัวได้อย่างถูกต้อง

cbs เป็น callback struct ของ scrolled list ที่จะให้ข้อมูลเกี่ยวกับวิดเจตตัวนี้ เช่น ตำแหน่งของข้อมูลใน list ที่ผู้ใช้เลือก

4.4.2.2 act(w,w_no,cbs) เป็นฟังก์ชัน callback ของ pushbutton ทั้ง 4 ตัว ได้แก่ 'distance' 'time' 'cost' และ 'done' มันจะเรียกใช้ฟังก์ชันที่เกี่ยวข้องกับปุ่มเหล่านี้ สำหรับค่าพารามิเตอร์นั้นจะเป็นเหมือนในฟังก์ชัน disp_select

4.4.2.3 redraw(w,unused,cbs) เป็นฟังก์ชัน callback ที่จะทำงานเมื่อมีการเคลื่อนย้าย drawing area บนจอภาพ, เมื่อมีการ expose หรือเมื่อมีการเปลี่ยนขนาดภาพ

ฟังก์ชันนี้จะคัดลอกรูปบิทแมปในหน่วยความจำไปบน drawing area สำหรับพารามิเตอร์ unused ในที่นี้ไม่ได้ถูกนำมาใช้งานแต่อย่างไร

4.4.2.4 click(w,data,cbs) เป็นฟังก์ชัน callback ที่จะถูกเรียกโดย drawing area เมื่อมีการกดปุ่มเมาส์ มันจะทำหน้าที่เทียบคู่ลำดับตำแหน่งที่เมาส์กดเลือกกับค่าคู่ลำดับต่างๆ ที่เราได้กำหนดไว้ล่วงหน้าในแฟ้มข้อมูลของระบบ เพื่อดูว่าผู้ใช้ได้กดปุ่มใกล้จุดที่เป็นเมืองบนแผนที่หรือไม่

4.4.2.5 quit_notice(w) เป็นฟังก์ชัน callback ที่จะถูกเรียกใช้เมื่อมีการกดปุ่ม 'quit' โดยจะมีการสร้างเป็น dialog box ขึ้นมาเพื่อขอการยืนยันจากผู้ใช้ว่าต้องการจะออกจาก Explorer จริงหรือไม่ ถ้าต้องการจะออกจริง ก็จะมีการเรียกฟังก์ชัน leave( ) เพื่อจบการทำงาน

4.4.3 ฟังก์ชันทั่วไป

4.4.3.1 disp_message(message) และ dispw_message(message) เป็นฟังก์ชันสำหรับการแสดงข้อความออกทางวิดเจต โดยผู้ใช้จะส่งพารามิเตอร์เป็นค่าสตริงที่ต้องการให้แสดง โดยฟังก์ชันทั้งสองจะทำงานคล้ายกันเพียงแต่ dispw_message(message) จะสามารถแสดงข้อความต่อเนื่องในแนวนอนของวิดเจต

พารามิเตอร์ message เก็บค่าสตริงที่ต้องการนำมาแสดงออกทางวิดเจต

4.4.3.2 vicinity(cx,cy,x,y) เป็นฟังก์ชันที่จะส่งค่ากลับหนึ่งค่า ถ้าคู่ลำดับ (cx,cy) นั้นอยู่ใกล้กับจุด (x,y) โดยห่างกันไม่เกิน 3 จุดทั้งในแนวนอนและแนวตั้ง ฟังก์ชันนี้ก็จะส่งค่ากลับเป็น 1 ฟังก์ชันนี้จะช่วยให้การกดเมาส์เพื่อเลือกเมืองทำได้ง่ายขึ้น เพราะไม่ต้องกดที่จุดที่เรากำหนดไว้จริงๆ

4.4.3.3 leave( ) เป็นฟังก์ชันที่ถูกเรียกเมื่อต้องการจบการทำงานของ Explorer

4.4.4 ฟังก์ชันด้านกราฟิก

4.4.4.1 loadgif(filename,vp) เป็นฟังก์ชันที่เปิดแฟ้มข้อมูลภาพนามสกุล .gif ทำการอ่านและนำมาเก็บอยู่ในรูปของโครงสร้างภาพ (image struct) แฟ้มข้อมูลภาพที่ใช้จะต้องเป็นแบบ GIF87a ค่าต่างๆ ที่เกี่ยวข้องกับกราฟิกเช่น แผนผังสี บน X จะถูกกำหนดจากแฟ้มข้อมูลกราฟิกตัวนี้

พารามิเตอร์

filename เป็นชื่อของแฟ้มข้อมูลนามสกุล .gif

vp เป็นโครงสร้างข้อมูลที่ได้มีการกำหนดไว้แล้วในแฟ้มข้อมูลที่มีชื่อว่า vardef.h โดยภายในมีตัวแปรที่เป็นพิกแมป, โครงสร้างภาพและชื่อแฟ้มข้อมูล

4.4.4.2 disp_view( ) เป็นฟังก์ชันที่ให้ข้อมูลของเมืองต่างๆ เมื่อผู้ใช้กดเลือกเมืองนั้นบนจอภาพ ฟังก์ชันนี้จะถูกเรียกผ่านฟังก์ชัน click( ) อีกที จะมีการใช้ค่า vindex เป็นดัชนี (เป็นตัวแปรแบบโกลบอล) เพื่อระบุเมืองที่ต้องการข้อมูล

4.4.4.3 line(srcx,srcy,destx,desty) เป็นฟังก์ชันที่ทำการวาดเส้นบน drawing area ระหว่างจุด (srcx,srcy) และจุด (destx,desty)

พารามิเตอร์

srcx, srcy เป็นจุดคู่ลำดับเริ่มต้น

destx, desty เป็นจุดคู่ลำดับสุดท้าย

4.4.4.4 loadpicture( ) เป็นฟังก์ชันที่ทำการเรียกภาพจากแฟ้มข้อมูลที่ชื่อว่า 2xthai.gif เพื่อแปลงมาเก็บไว้ในฟังก์ชันตัวนี้ก่อนที่จะเรียกใช้ฟังก์ชัน loadgif( ) อีกที

4.4.4.5 drawpixmap( ) เป็นฟังก์ชันที่ทำการสร้างพิกแมป จากข้อมูลภาพที่ถูกอ่านขึ้นมาโดยฟังก์ชัน loadpicture( ) ที่เก็บอยู่ในรูปของโครงสร้างภาพ เพื่อนำไปใช้แสดงที่ drawing area ต่อไป

4.4.4.6 draw_line(srci,desti) เป็นฟังก์ชันที่ทำการลากเส้นระหว่างจุด 2 จุด โดยเรียกใช้ line( ) อีกที

พารามิเตอร์

srci เป็นดัชนีของเมืองที่เป็นจุดเริ่มต้น

desti เป็นดัชนีของเมืองที่เป็นจุดสิ้นสุดของการเดินทาง

4.4.4.7 clear_map( ) เป็นฟังก์ชันที่ทำการวาดแผนที่บน drawing area ใหม่ ฟังก์ชันนี้จะถูกนำมาใช้เมื่อเราต้องการนำเส้นทางที่วาดบนแผนที่โดยคำสั่ง draw_line ออก มันจะถูกเรียกเมื่อมีการกดปุ่ม pushbutton ซึ่งก็จะมีการคัดลอกภาพพิกแมปใหม่ให้กับพิกแมปเดิมที่ถูกนำไปแสดงบน drawing area

4.5 ค่าของทรัพยากร fallback_resource

ใน Explorer จะมีการกำหนดทรัพยากรแบบที่เรียกว่า fallback_resource โดยเมื่อมีการสร้างวิดเจต fallback_resource จะทำการกำหนดค่าต่างๆ เช่น ชื่อของ pushbutton, font ที่ใช้ ผู้ใช้สามารถเปลี่ยนค่าใน fallback_resource ได้ แต่การเปลี่ยนค่าเหล่านี้จะมีผลต่อการ

จัดเรียงวิดเจตอย่างมาก ดังนั้นจึงไม่เหมาะที่จะกำหนดค่าเอง fallback_resource จะปรากฏอยู่ใน vardef.h ดังนี้

```
fallback_resource[ ] = {
        "*.....",
        "*.....",
        NULL };
```

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทวิจารณ์

โครงงานนี้ เป็นโครงงานหนึ่งที่น่าสนใจในการนำไปพัฒนาต่อ เนื่องจากเป็นโครงงานที่มีการเริ่มต้นที่ดีแต่การแสดงผลข้อมูลยังไม่ครบถ้วนพอ หน้าจอในส่วนที่ติดต่อกับผู้ใช้ก็มีเพียงหน้าจอเดียว ความสามารถต่างๆ จะถูกจำกัดจากปัจจัยในด้านของเวลา

การทำงานของ Explorer นั้นกล่าวได้ว่าเป็นการสื่อสารกับผู้ใช้ที่ช่วยลดความซับซ้อนในการติดต่อเป็นอย่างมาก เพราะมีการใช้วิดเจตต่างๆ สำหรับป้อนข้อมูลและแสดงผลลัพธ์ที่ได้อย่างชัดเจน การทำงานของโปรแกรมก็เป็นไปอย่างรวดเร็ว การอนุญาตให้ผู้ใช้สามารถกำหนดเมืองรวมทั้งรูปภาพที่นำมาแสดงบนจอได้นั้น เปิดโอกาสให้สามารถนำ Explorer ไปปรับให้เข้ากับงานต่างๆ โดยที่ผู้ใช้ไม่จำเป็นต้องเข้าใจโครงสร้างและการทำงานของโปรแกรมมากนัก ทั้งยังสามารถเปลี่ยนแปลงลักษณะการทำงานของโปรแกรมได้สะดวกเพราะไม่ต้องนำโปรแกรมมาคอมไพล์ใหม่

อย่างไรก็ตามการใช้ Explorer นั้นต้องการพื้นที่และทรัพยากรจำนวนมาก จึงทำให้การใช้งานโปรแกรมแบบตัวมันเองเดี่ยวๆ อาจไม่คุ้มค่า การพัฒนาโครงงานนี้ต่อไปจึงควรปรับโปรแกรมนี้ให้ทำงานเป็นแบบมีผู้ใช้หลายคนร่วมกันได้ โดยให้มีการใช้ทรัพยากรร่วมกันให้มากที่สุด

ส่วนที่เป็นรูปภาพใน Explorer มีช่องทางมากมายสำหรับการนำมาขยายเพิ่มเติม เช่น เราอาจแทนฐานข้อมูลแบบข้อความของแต่ละเมืองที่ถูกนำมาแสดงเมื่อผู้ใช้กดเลือกเมืองนั้น ด้วยฐานข้อมูลแบบภาพของเมืองนั้นเพื่อดึงดูดความสนใจของผู้ใช้

นอกจากนี้การพัฒนาต่ออาจทำเพิ่มในส่วนที่เกี่ยวกับการเพิ่มหน้าจอของการรับข้อมูลเพื่อใช้แก้ไขตารางหรือ หน้าจอของการแสดงข้อมูลต่างๆ ในตาราง การแสดงรายละเอียดต่างๆ ของเที่ยวบิน เป็นต้น

5.2 ปัญหาที่พบ

โครงงานที่เกี่ยวกับการออกแบบส่วนติดต่อกับผู้ใช้บนระบบ X Window หารายละเอียดได้ยาก ส่วนตัวอย่างนั้นก็มีน้อยและไม่หลากหลาย ทำให้การแก้ปัญหาขาดประสิทธิภาพและล่าช้า นอกจากนี้ยังขาดแคลนบุคลากรผู้เชี่ยวชาญ ทำให้ในบางครั้งวิธีการแก้ปัญหา ต้องใช้การตั้งสมมติฐานและการลองผิดลองถูก

ในการเขียนโปรแกรมบนระบบ X Window นั้น ถึงแม้เราจะมุ่งเน้นไปที่การเขียนโปรแกรมบนโมทีฟ แต่เนื่องจากโมทีฟทำงานร่วมกับไลบรารีตัวอื่นๆ เช่น Xlib และ XtIntrinsic ทำให้ต้องมีการศึกษาการทำงานและการนำไลบรารีเหล่านั้นมาใช้งานด้วย ส่งผลให้ขอบเขตการศึกษากว้างมากและใช้เวลานานกว่าที่จะสรุปหาวิธีการที่เหมาะสมสำหรับพัฒนาโครงงาน

เนื่องจากระบบ X Window เป็นระบบที่ทำงานบนระบบปฏิบัติการยูนิกซ์ ที่ขาดเครื่องมือที่ช่วยในการพัฒนาโปรแกรม อีกทั้งตัวดีบักเกอร์บนยูนิกซ์ก็ขาดความคล่องตัว

5.3 สรุป

การพัฒนาส่วนติดต่อกับผู้ใช้โดยใช้โมทีฟที่เป็นวิดเจตเซตนั้น พบว่า มีความซับซ้อนพอสมควรเมื่อเทียบกับการพัฒนาโปรแกรมทั่วๆ ไป ทั้งนี้เพราะ ลักษณะการเขียนโปรแกรมแบบเชิงวัตถุ ที่มีการสืบทอดคุณสมบัติของคลาสเป็นลำดับชั้นลงมา และลักษณะการทำงานของโปรแกรมที่เป็นแบบ event driven ซึ่งแตกต่างจากการเขียนโปรแกรมแบบเป็นลำดับขั้นที่พบทั่วไป

อย่างไรก็ตาม หากมีความเข้าใจในรูปแบบพื้นฐานของการเขียนโปรแกรมโดยใช้โมทีฟแล้ว การพัฒนาโปรแกรมต่อไปนั้น สามารถทำได้อย่างมีระบบและรวดเร็ว

กิตติกรรมประกาศ

ผู้จัดทำขอขอบพระคุณบุคคลต่างๆ ที่ให้ความช่วยเหลือทั้งในด้านการให้ข้อมูล ให้คำปรึกษา รวมทั้งเป็นกำลังใจในการทำโครงงานนี้ให้สำเร็จลุล่วงไปด้วยดี ดังต่อไปนี้

1. ดร. บุญธีร์ เครือตราชู อาจารย์ที่ปรึกษา ผู้ให้คำปรึกษา คำแนะนำและข้อมูลที่เป็นประโยชน์ในการทำโครงงานนี้มาโดยตลอด

2. อาจารย์ วัชระ ฉัตรวิริยะ ผู้ให้คำแนะนำและแนวความคิดในการออกแบบการทำงานของโปรแกรม รวมทั้งชี้ช่องทางในการหาข้อมูล

3. คุณณรงค์ศักดิ์ (พี่เจี๊ยบ) ผู้ดูแลระบบเครือข่ายของศูนย์ CAD/CAM ที่ให้ความช่วยเหลือและอำนวยความสะดวกในการใช้เครื่อง

4. คุณพ่อคุณแม่ พี่น้อง ที่ช่วยเป็นกำลังใจ และให้การสนับสนุนในการทำโครงงานมาโดยตลอด

5. คุณเกียรติกูล เจียรนัยธนะกิจ (ลี) 34101036 เพื่อนผู้ให้ความช่วยเหลือในการทำโครงงานส่วนของกราฟิก และสละเวลาในการหาข้อผิดพลาดของโปรแกรม และคุณน้องโก้ น้องสาวผู้น่ารักที่มีส่วนช่วยในการจัดพิมพ์ปริญญานิพนธ์

หนังสืออ้างอิง

1. ดวงแก้ว สวามิภักดิ์, "ระบบดำเนินงานยูนิกซ์", บริษัท ซีเอ็ดยูเคชัน จำกัด, 2532

2. บุญเลิศ เอี่ยมทัศนา, ยืน ภู่วรวรรณและสมนึก คีรีโต, "โปรแกรมคอมพิวเตอร์ภาษาซี", บริษัท ซีเอ็ดยูเคชัน จำกัด, 2531

3. มนตรี พจนารถลาวัณย์, "การเขียนโปรแกรมคอมพิวเตอร์ด้วยเทอร์โบซี", บริษัท ซีเอ็ดยูเคชัน จำกัด, 2535

4. สุขกรี เสรีวัลย์สถิตย์, "เรียนรู้ภาษาซี", บริษัท ซีเอ็ดยูเคชัน จำกัด, 2533

5. Bhagat Singh and Thomas L. Naps, "Introduction to Data Structure"

6. Dan Heller, "Motif Programming Manual", O'Reilly & Associater,Inc., 998 p., 1992.

7. Douglas A. Young , "The X Window System : programming and applications with Xt", Prentice Hall, 533p, 1990

8. Robert L. Kruse, Bruce P. Leung and Clovis L. Tondo, "Data Structures and Program Design in C", Prentice-Hall, 1991