โปรแกรมจำลองการต่อสู้ของหุ่นยนต์ด้วยภาษาจาวา

The Simulation of Fighting Robots with Java Language (Jrobot)

โดย

นายมนานาท ลิขิตธนานันท์

นายวินัย จินดาดำรงเวช

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

โปรแกรมจำลองการต่อสู้ของหุ่นยนต์ด้วยภาษาจาวา

The Simulation of Fighting Robots with Java Language (Jrobot)

โดย

นายฆนานาท ลิขิตธนานันท์ 37014040

นายวินัย จินดาดำรงเวช 37014399

อาจารย์ที่ปรึกษา

อาจารย์ธนา หงษ์สุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

ปริญญานิพนธ์ปีการศึกษา 2540

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมจำลองการต่อสู้ของหุ่นยนต์ด้วยภาษาจาวา

ผู้จัดทำ

1. นายฆนานาท ลิขิตธนานันท์
2. นายวินัย จินดาดำรงเวช

..... อาจารย์ที่ปรึกษา

(.....)

โปรแกรมจำลองการต่อสู้ของหุ่นยนต์ด้วยภาษาจาวา

นายฆนานาท ลิขิธนานันท์
นายวินัย จินดาดำรงเวช

อาจารย์ธนา หงษ์สุวรรณ
ปีการศึกษา 2540

บทคัดย่อ

ปริญญานิพนธ์นี้นำเสนอการพัฒนาโปรแกรมประยุกต์บนระบบเครือข่าย ระบบเครือข่ายที่ทำการศึกษาคือ ระบบเครือข่ายอินเตอร์เน็ต (Internet) ซึ่งมีผู้ใช้บริการอย่างแพร่หลาย โดยมีโปรโตคอลที่ใช้ในการสื่อสารคือ โปรโตคอล TCP/IP ระบบปฏิบัติการที่เลือกใช้ในการพัฒนาโปรแกรมประยุกต์นี้คือ ระบบปฏิบัติการวินโดว์ เอ็น ที

โครงงานนี้ได้พัฒนาโปรแกรมจำลองการต่อสู้ของหุ่นยนต์ด้วยภาษาจาวา โดยเล่นผ่านทางระบบเครือข่ายอินเตอร์เน็ต เครื่องมือแปลภาษาที่ใช้ได้แก่ Java Developers Kit (Release 1.0.2) for Solaris 2.x ,Windows 95 ,Windows NT ,and Macintosh (JDK 1.0.2) เกมจำลองการต่อสู้ของหุ่นยนต์มิใช่เป็นเพียงแค่เกมเพื่อความสนุกสนานเพียงอย่างเดียว แต่ใช้ในการพัฒนาการเขียนโปรแกรมภาษาจาวาให้กับผู้เล่น และสามารถที่แสดงการต่อสู้ให้กับผู้ที่สนใจ ดูได้อย่างแบบ เรียลไทม์ (real time) โดยผู้สนใจสามารถดูจากที่ต่างๆ ได้โดนผ่านระบบเครือข่ายอินเตอร์เน็ต

The Simulation of Fighting Robots with Java Language (JRobots)

Kananart Likittananan

Winai Jindadamrongwet

Thana Hongsuwan

1997

Abstract

This thesis presents the development of an application on network systems . We'll focalize on the Internet system which is interesting . And it has owned protocol named TCP/IP . The select operation system is Windows NT .

This project developed the simulation of fighting robots with Java language to play in the Internet . We used Java Developers Kit (Release 1.0.2) for Solaris 2.x ,Windows 95 ,Windows NT ,and Macintosh (JDK 1.0.2) in order to design . The simulation of fighting robots is not game for only funny but this game is help developed of Java language for programmer . And the attendant can view the simulation in real-time mode , from everywhere through the Internet system by using web browser.

สารบัญ

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

จากยุคเริ่มต้นของคอมพิวเตอร์ เครื่องคอมพิวเตอร์เครื่องหนึ่งๆ มีราคาแพงมาก ในเวลาต่อมาการพัฒนาอุปกรณ์ทางอิเล็กทรอนิกส์ ได้ทำให้ราคาชิ้นส่วนของคอมพิวเตอร์ถูกลงเรื่อยๆ และประสิทธิภาพยิ่งดีขึ้น คอมพิวเตอร์จึงมีลักษณะการใช้งานแบบหนึ่งคน ต่อเครื่องคอมพิวเตอร์หนึ่งเครื่อง ซึ่งเรียกกันว่าระบบผู้ใช้คนเดียว (Single User)

เมื่อระบบโทรคมนาคมมีการพัฒนามากขึ้น มีการเชื่อมต่อการสื่อสารจากที่ต่างๆ ภายในประเทศและจากต่างประเทศ ได้มีการพัฒนา ได้มีการพัฒนาระบบเครือข่ายระยะไกล หรือ WAN (Wide Area Network) ทำให้การส่งข้อมูลในระยะไกลได้

การติดต่อสื่อสารของคอมพิวเตอร์จึงกลายมามีสำคัญอย่างมากในชีวิต ซึ่งมีการติดต่อสื่อสารระหว่างคอมพิวเตอร์ กันหลายแบบ และหนึ่งในนั้นคือ ระบบเครือข่ายอินเตอร์เน็ต และได้กลายมาเป็นระบบเครือข่ายที่ใหญ่ที่สุด มีเครื่องคอมพิวเตอร์มาเชื่อมต่อกันมากกว่าสิบล้านเครื่อง และระบบเครือข่ายอินเตอร์เน็ต มีการกำหนดมาตรฐานร่วมกัน เพื่อใช้ในการติดต่อสื่อสารระหว่างคอมพิวเตอร์ด้วยกัน โดยได้กำหนด โปรโตคอลที่ชื่อ TCP/IP (Transmission Control Protocol / Internet Protocol)

ระบบเครือข่ายอินเตอร์เน็ตได้ถูกนำมาใช้ในทางการศึกษา ทางการพาณิชย์ นอกจากนี้บริษัทซอฟต์แวร์ก็เพิ่มประสิทธิภาพการลงทะเบียน หรือการรีจิสเตอร์ผ่านทางอินเตอร์เน็ต และบริการส่งโปรแกรมหรืออัพเกรดโปรแกรมในทันที โดยไม่เสียเวลา และเงิน ในการสื่อสารผ่านทาง โทรศัพท์ระหว่างประเทศหรือจดหมาย ซึ่งมีราคาแพงกว่า ไม่สะดวกทันใจเท่ากับการใช้บริการทางอินเตอร์เน็ต

ในปัจจุบัน เครื่องคอมพิวเตอร์ประมาณกว่าครึ่งหนึ่งของจำนวนทั้งหมด เชื่อมต่อกับระบบเครือข่ายอินเตอร์เน็ต โดยใช้ระบบปฏิบัติการวินโดว์ และบริการพื้นฐานที่ผู้ใช้ทั่วไปส่วนใหญ่ใช้จากเครือข่ายอินเตอร์เน็ต ก็คือการรับส่งจดหมายอิเล็กทรอนิกส์ การคุยกันบนอินเตอร์เน็ต การใช้บริการเอกสารเครือข่ายในรูปของโฮมเพจ และโปรแกรมประยุกต์ ต่างๆ บนระบบอินเตอร์เน็ต

ประโยชน์ของ ระบบเครือข่ายอินเตอร์เน็ต

- เป็น แหล่งที่ทำให้เกิดการพัฒนาเทคโนโลยีใหม่ ๆ
- เกิดการติดต่อสื่อสารระหว่างกัน และให้บริการข่าวสาร
- ทำให้เกิดการแลกเปลี่ยนความรู้ แนวความคิด และ เกิดการศึกษา
- เป็นการติดต่อสื่อสารข้อมูลที่เร็วมาก
- ใช้สื่อสารทางด้านธุรกิจผ่านเครือข่ายคอมพิวเตอร์

เมื่อ ระบบเครือข่าย อินเตอร์เน็ต เป็นที่นิยมอย่างมากก็ทำให้มีผู้พัฒนา โปรแกรมประยุกต์ ต่างๆ เพื่อที่จะมาใช้งานร่วมกับ ระบบเครือข่าย อินเตอร์เน็ต และ ภาษาจาวา ก็เป็นภาษาใหม่ที่มีแนวโน้มจะเป็นที่นิยมในอนาคตที่ใช้ในการพัฒนา โปรแกรมประยุกต์ บน อินเตอร์เน็ต

ประโยชน์ของ JAVA

- เป็นภาษาแบบเชิงวัตถุ ที่เขียนง่าย
- มีการจัดเก็บกับข้อมูลอย่างมีประสิทธิภาพ และปลอดภัย
- สามารถดึง Class Library จากที่ต่างๆ มาใช้งานร่วมได้
- โปรแกรมผ่านการตรวจสอบอย่างดีขณะเวลาแปลหรือ คอมไพล์ และขณะปฏิบัติงาน เพื่อให้แน่ใจว่าไม่เกิดปัญหาขึ้นภายหลังเมื่อเวลาใช้จริง
- ไม่มีข้อมูลชนิด พอยน์เตอร์ เพื่อป้องกันการเข้าใช้ข้อมูลกับหน่วยความจำโดยตรง ซึ่งอาจก่อให้เกิดผลเสียกับระบบ
- architecture neutral ใช้ ไบต์โค้ด รูปแบบเดียวกันหมดสามารถถูกปฏิบัติงานได้ทุกที่ไม่ขึ้นกับระบบเครือข่าย
- Multithreaded ทำงานได้หลายงานในเวลาเดียวกัน (concurrency)
- ความสามารถด้านกราฟฟิก โดยเฉพาะ กราฟฟิกแสดงภาพเคลื่อนไหว และการเล่นไฟล์ เสียง ได้ในเวลาเดียวกัน
- ไม่ขึ้นกับ Platform ใดๆ

ซึ่งทำให้มีความคิดที่จะเขียน โปรแกรมประยุกต์ บน อินเตอร์เน็ต โดยใช้ภาษาจาวา ซึ่งเป็นภาษาที่เหมาะสมอย่างมากในการเขียน โปรแกรมประยุกต์ บน อินเตอร์เน็ต จึงได้ศึกษาว่าจะทำอะไร ก็ได้มองเห็นเกม CRobots ซึ่งเป็นเกมที่ดีมากๆ เกมหนึ่ง ไม่ใช้แค่เกมที่เล่นเพื่อความสนุกเพียงอย่างเดียว แต่ได้ช่วยในการพัฒนา การเขียนภาษา C ให้กับผู้เขียนอีกด้วย ทำให้มองเห็นประโยชน์ในการทำ โปรแกรมประยุกต์ ตัวนี้ ดังนั้นจึงได้มีความเห็นว่าจะเขียน โปรแกรม เกม **JRobots** เพื่อใช้ในการพัฒนาการเขียน ภาษาจาวา โดยมีลักษณะคล้ายกับ CRobots แต่สามารถปฏิบัติงานบน อินเตอร์เน็ต และสามารถ ดูการปฏิบัติงานแบบเรียลไทม์

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 ศึกษาการเรียนภาษาจาวา มีลักษณะการเขียนอย่างไร และศึกษาระบบเครือข่ายอินเตอร์เน็ต มีการเขียนโปรแกรมประยุกต์ บนระบบเครือข่ายอินเตอร์เน็ตอย่างไร

1.2.2 ศึกษาการทำงานระหว่าง เว็บเซิร์ฟเวอร์ กับ เว็บไคลเอนต์ มีการติดต่ออย่างไร มีลักษณะโครงสร้างอย่างไร และนำความรู้ไปพัฒนา Java Application

1.2.3 นำโปรแกรมที่พัฒนา มาช่วย โปรแกรมเมอร์ฝึกฝน ทดสอบ และพัฒนาการเขียน ภาษาจาวา และ โครงสร้างอัลกอริทึม

1.3 ลักษณะโครงงาน

JRobots เป็นเกมที่ช่วยพัฒนาด้านการเขียนโปรแกรม ภาษาจาวา ที่มีความแตกต่างจากเกมต่อสู้ทั่ว ๆ ไปซึ่งผู้เล่นต้องควบคุมหุ่นในขณะเล่นแต่ JRobots จะต้องกำหนดการเคลื่อนไหวต่างๆ ของหุ่น , แอคชั่นต่างๆ ของหุ่น อย่างเช่น รูปแบบการโจมตีศัตรู, รูปแบบการเคลื่อนที่, การตรวจสอบความเสียหายของหุ่น, เช็คตำแหน่งของศัตรู ฯลฯ แอคชั่นต่างๆ เหล่านี้จะถูกเขียนให้เสร็จก่อนแล้วนำมาคอมไพล์ เมื่อ คอมไพล์เสร็จจึงจะทำการจำลองการแข่งขัน เพื่อต่อสู้กันในสนามรบจำลอง (Simulate field) ระหว่างการแข่งขัน ผู้เล่นจะไม่สามารถควบคุมหุ่นของตนเองได้ ลักษณะของหุ่นจะเป็นโปรแกรมจาวา ซึ่งผู้เล่นสามารถพัฒนาหุ่นให้มีความสามารถได้ตามต้องการ แต่ต้องอยู่ภายใต้ข้อกำหนดที่ระบุไว้ในกติกา หุ่นยนต์ทุกตัวต้องมีความสามารถในการค้นหา, การติดตาม และ การทำลายหุ่นยนต์ของผู้เล่นอื่น

JRobots มีลักษณะคล้ายกับเกม Simulate of Fighting Robots อื่น ๆ อย่างเช่น CRobots (ซึ่งใช้ภาษาซี ในการเขียนหุ่น) , PRobots (ซึ่งใช้ภาษาปาสคาล ในการเขียนหุ่น) , TRobots (ซึ่งใช้ภาษาแอสเซมบลี ในการเขียนหุ่น) แต่ก็แตกต่างตรงที่ Simulate of Fighting Robots อื่น ๆ นั้นจะทำการรัน หรือการแข่งขันจำลองบนเครื่องคอมพิวเตอร์เพียงเครื่องเดียว และผู้เล่นจำเป็นต้องอยู่ที่นั้นด้วย แล้วทำการโหลดโปรแกรม หุ่นนั้นเพื่อ คอมไพล์ จนครบทุกตัว และ ทำการจำลองการแข่งขัน ที่นั้นเลย แต่ JRobots จะทำการจำลองการแข่งขันบน อินเตอร์เน็ต และจะทำการจำลองการแข่งขันการต่อสู้ของหุ่นยนต์แบบ เรียลไทม์ ซึ่งผู้เล่นสามารถใช้ระบบเครือข่ายอินเตอร์เน็ต ดูการแข่งขันต่อสู้กันจริงๆ ในขณะรัน อยู่ และดูสถานะ (Status) ต่าง ๆ ของหุ่นทุกตัวที่แข่งขันอยู่ ว่าเป็นอย่างไร, กำลังทำอะไร, มีความเร็วเท่าไร, เหลือพลังงานเท่าไร หรือสูญเสียพลังไปเท่าไรแล้ว สามารถดูผลการแข่งขันที่ผ่านมา และอันดับในการแข่งขันใน ตารางเวลาแข่งขันได้อีกด้วย

บทที่ 2
ทฤษฎี และความรู้เบื้องต้น

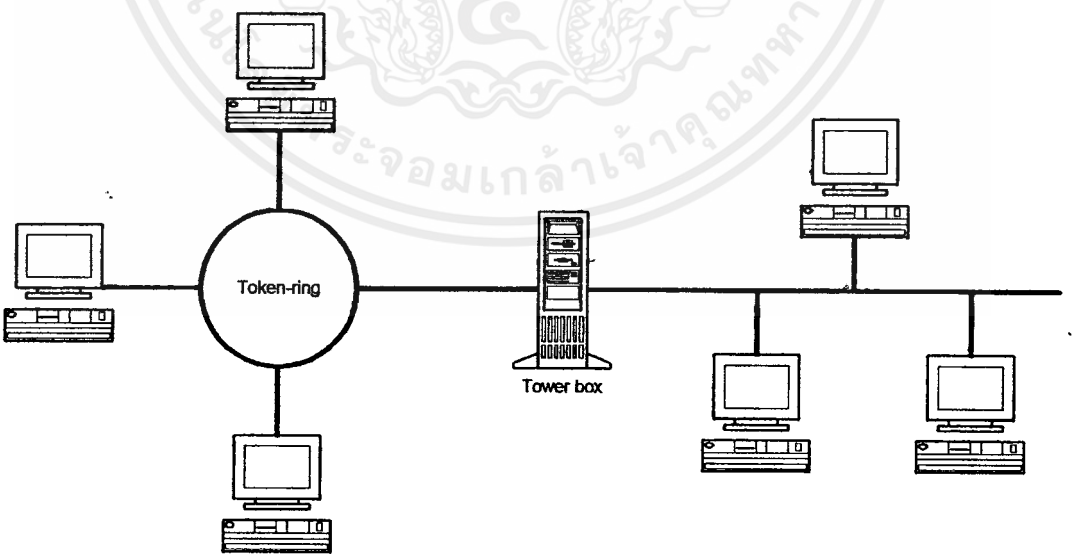
2.1 ระบบเครือข่าย และการโปรแกรมบนระบบเครือข่าย

การพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายจำเป็นต้องมีความรู้พื้นฐานทางด้านระบบเครือข่ายพอสมควร ในหัวข้อนี้จะอธิบายเกี่ยวกับคำศัพท์พื้นฐาน และหลักการของระบบเครือข่ายที่จำเป็นสำหรับการทำความเข้าใจในการพัฒนาโปรแกรมประยุกต์บนระบบเครือข่ายดังจะกล่าวต่อไป

2.1.1 ความหมายของระบบเครือข่ายคอมพิวเตอร์ และอินเตอร์เน็ตเวิร์คกิ้ง (Internet-working)

ระบบเครือข่ายคอมพิวเตอร์ (Computer Network) คือระบบการเชื่อมต่อระหว่างระบบปลายทาง (End-System) ซึ่งระบบปลายทางเป็นระบบที่เป็นอิสระจากกัน (Autonomous) ระบบปลายทางสามารถเป็นได้ตั้งแต่ไมโครคอมพิวเตอร์ ไปจนกระทั่ง ซุปเปอร์คอมพิวเตอร์ขนาดใหญ่ เพื่อจุดมุ่งหมายในการแลกเปลี่ยนข้อมูลและการแบ่งปันทรัพยากรของระบบ เช่น ไฟล์ (File) ข้อมูล , เครื่องพิมพ์ (Printer) , โมเด็ม (Modem) ตลอดจนการให้บริการฐานข้อมูลร่วม (Sharing database)

อินเตอร์เน็ตเวิร์คกิ้ง หรืออินเตอร์เน็ต คือการเชื่อมต่อของระบบเครือข่าย 2 เครือข่ายขึ้นไป ดังนั้นคอมพิวเตอร์บนระบบเครือข่ายหนึ่งก็สามารถติดต่อกับคอมพิวเตอร์บนระบบเครือข่ายอื่น ๆ ได้ เช่น เครือข่ายโทเคนริงค์ (Tokenring network) เชื่อมกับ เครือข่ายอีเทอร์เน็ต (Ethernet network) โดยมีเกตเวย์ (Gateway) เป็นตัวเชื่อมดังรูป



รูปที่2.1 ระบบเครือข่ายคอมพิวเตอร์เบื้องต้น

2.1.2 รูปแบบของโปรแกรมบนระบบเครือข่าย (Network Programming Models)

จากหัวข้อ2.1.1 ได้ให้ความหมายของระบบเครือข่าย แสดงถึงวิธีการที่ระบบคอมพิวเตอร์ใดๆ จะทำการเชื่อมต่อกับระบบเครือข่าย แต่ว่าคอมพิวเตอร์ที่เชื่อมต่ออยู่กับระบบเครือข่ายนั้น มีวิธีการอย่างไรในการแลกเปลี่ยนข้อมูล และแบ่งปันทรัพยากรเหล่านั้นได้ ทำให้ต้องมีโปรแกรมประยุกต์ ซึ่งสามารถที่จะจัดการในสิ่งที่กล่าวมาแล้วอย่างเหมาะสม ซึ่งในหัวข้อนี้จะกล่าวถึงรูปแบบของโปรแกรมบนระบบเครือข่าย 2 รูปแบบคือ ไคลเอนต์เซิร์ฟเวอร์ (Client / Server Computing) และการประมวลผลแบบกระจาย (Distributed Computing)

2.1.2.1 การประมวลผลแบบไคลเอนต์เซิร์ฟเวอร์ (Client/Server Computing)

การประมวลผลแบบไคลเอนต์เซิร์ฟเวอร์นี้ การประมวลผลของโปรแกรมประยุกต์จะแบ่งออกเป็น 2 ส่วน คือ ส่วนฟรอนต์เอนด์ (front-end) ที่ทำงานบน ไคลเอนต์ ส่วนนี้จะทำหน้าที่แสดงผลที่ได้จากการประมวลผล และรับข้อมูลจากผู้ใช้ อีกส่วนหนึ่งจากแบ็คเอนด์ (back-end) ทำงานบนเซิร์ฟเวอร์มีหน้าที่ในการเก็บรวบรวมและจัดการข้อมูลจากฟรอนต์เอนด์ในรูปแบบการประมวลผลไคลเอนต์เซิร์ฟเวอร์นี้ เครื่องเซิร์ฟเวอร์มักจะเป็นเครื่องที่มีความสามารถสูงกว่าเครื่องไคลเอนต์ โดยปกติเครื่องเซิร์ฟเวอร์มักจะเป็นเครื่องเมนเฟรมหรือมินิคอมพิวเตอร์และเครื่องไคลเอนต์มักจะเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล ซึ่งการติดต่อกันระหว่างส่วนฟรอนต์เอนด์ และแบ็คเอนด์ ทำโดยผ่านระบบเครือข่าย ซึ่งในทางปฏิบัติแล้ว ในส่วนแบ็คเอนด์ที่อยู่บนเซิร์ฟเวอร์ จะเป็นฝ่ายให้บริการแก่งานของไคลเอนต์หลายงานในเวลาเดียวกัน

2.1.2.2 การประมวลผลแบบกระจาย (Distributed Computing)

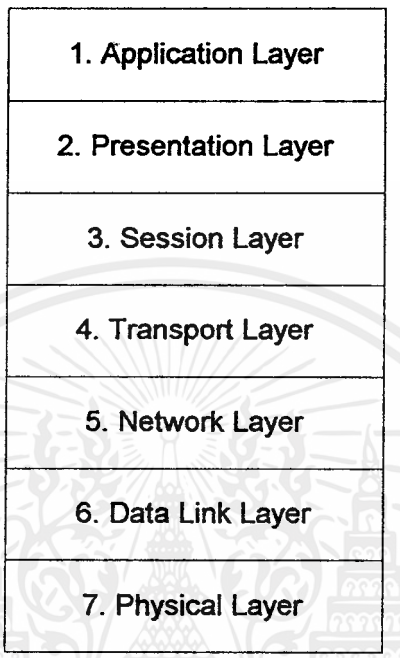
การประมวลผลของโปรแกรมประยุกต์มีรูปแบบการประมวล 2 แบบ คือ

1. **พรีคอลเล็คชั่น** เป็นลักษณะการทำงานที่ข้อมูลที่ต้องการในการจัดเก็บ และส่งต่อไปทั่วระบบเครือข่ายตลอดเวลาอย่างสม่ำเสมอ การทำงานในลักษณะนี้จะเหมาะสมกับงานบางงาน เช่น ต้องการเก็บสถานะของคอมพิวเตอร์ที่อยู่ในระบบเครือข่ายหนึ่งๆ ทุกเครื่อง
2. **การประมวลผลแบบขนาน** การประมวลผลในลักษณะนี้ งานใดๆ จะถูกประมวลผลด้วยคอมพิวเตอร์หลายๆ เครื่อง โดยเครื่องคอมพิวเตอร์เหล่านั้นสามารถจะติดต่อกันโดยระบบเครือข่าย เช่น การทำการพัฒนาโปรแกรมประยุกต์ขนาดใหญ่ โดยทีมพัฒนาที่มีผู้พัฒนาหลายคน สามารถลดเวลาของการแปล และการรวมโมดูล (module) ต่างๆ เข้าเป็นโปรแกรมเดียวกันโดยการแบ่งงานการแปลโมดูลเหล่านั้นแก่คอมพิวเตอร์ในระบบเครือข่ายทำการแปลในเวลาเดียวกัน

2.1.3 โมเดล OSI

OSI เป็นคำย่อที่มาจากคำว่า Open Systems Interconnection โดยที่เป็นมาตรฐานที่ถูกเสนอขึ้นโดย International Standards Organization ซึ่งเป็นองค์กรที่จัดตั้งขึ้นมาเพื่อดูแลและส่งเสริม ตลอดจนกำหนดมาตรฐานของการติดต่อสื่อสารของระบบเครือข่ายคอมพิวเตอร์ โดยโมเดล OSI นี้มีลักษณะเป็น

สถาปัตยกรรมแบบระบบเปิด เพราะมุ่งที่จะให้ระบบคอมพิวเตอร์ในหลายๆ รูปแบบที่แตกต่างกันสามารถเชื่อมต่อกันได้ OSI โมเดลได้แบ่งโปรโตคอล ในการสื่อสารออกเป็น 7 เลเยอร์ ซึ่งโปรโตคอลคือชุดของกฎหรือข้อตกลงในการติดต่อ ข้อสังเกตโมเดล OSI เป็นเพียงข้อเสนอแนะ มิใช่ข้อกำหนด และควรรู้ว่ายังไม่มีระบบการเชื่อมต่อใดที่สร้างเหมือนกับโมเดล OSI จริงๆ

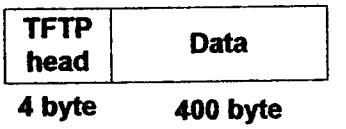


รูปที่ 2.2 OSI โมเดล ทั้ง 7 เลเยอร์

ในหนึ่งชั้นของเลเยอร์ไม่ได้กำหนดว่าจะต้องมีเพียงหนึ่งโปรโตคอลเท่านั้นที่อยู่ในระดับเลเยอร์เดียวกัน และในทางตรงข้าม ชุดของโปรโตคอลใดๆ อาจจะมีมากกว่าหนึ่งเลเยอร์ ประกอบกันเป็นข้อกำหนดของระบบเครือข่าย เรียกว่าชุดโปรโตคอล เช่น ชุดโปรโตคอล TCP/IP (Transmission Control Protocol/Internet Protocol) เป็นต้น ประโยชน์ในการแบ่งเป็นเลเยอร์ คือกำหนดการติดต่อระหว่างเลเยอร์ทำได้โดยไม่ต้องคำนึงถึงการเปลี่ยนในเลเยอร์ใดๆที่ติดกัน

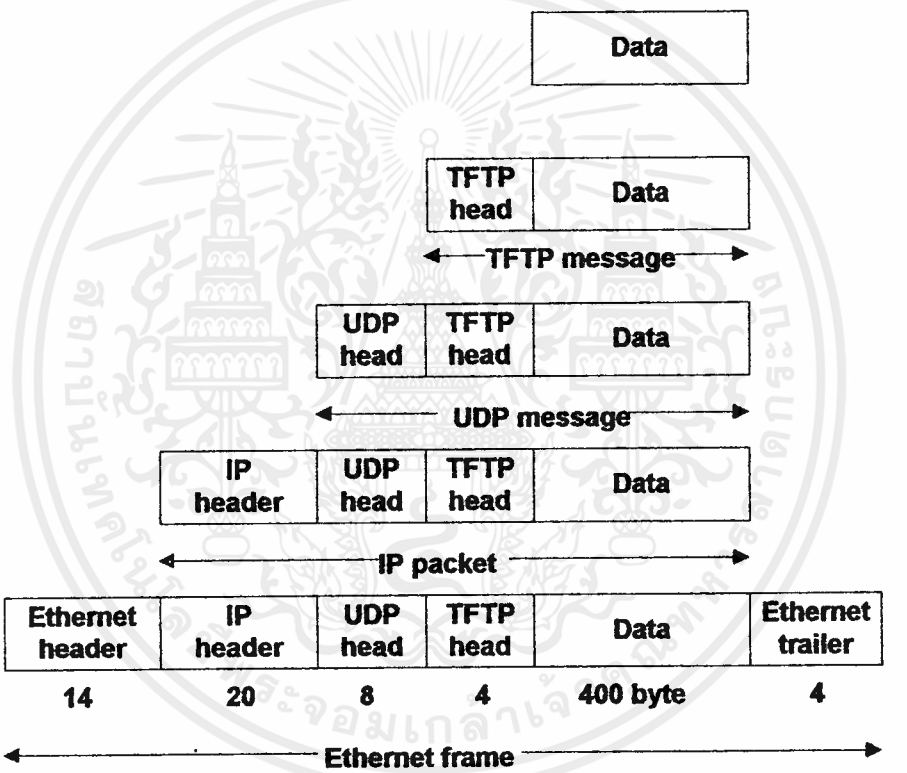
2.1.4 เอ็นแคปซูเลชั่น (Encapsulation)

พิจารณาโปรแกรมประยุกต์ TFTP (File Transfer Protocol) ซึ่งใช้ในโปรโตคอล UDP (User Datagram Protocol) ระหว่างสองระบบซึ่งเชื่อมต่อด้วยอีเทอร์เน็ต ถ้าโปรแกรมไคลเอนต์ TFTP มีข้อมูล 400 ไบต์ ต้องการส่งไปที่โปรแกรมเซิร์ฟเวอร์ โปรแกรมไคลเอนต์ TFTP จะเพิ่มข่าวสารควบคุม 4 ไบต์ เป็นส่วนหัวของข้อมูลก่อนที่จะผ่านข้อมูลไปสู่เลเยอร์ UDP การเพิ่มของข่าวสารควบคุมไปที่ข้อมูลเรียกว่า เอ็นแคปซูเลชั่น ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 การเอ็นแคปซูเลชั่น

เลเยอร์ UDP จะไม่มีการตีความส่วนหัว TFTP 4 ไบต์ งานของเลเยอร์ UDP คือส่งข้อมูล 404 ไบต์ไปสู่เลเยอร์ UDP ของโปรแกรมอีกด้านหนึ่ง จากนั้นเลเยอร์ UDP จะทำการเพิ่มส่วนหัว 8 ไบต์ แล้วส่งข้อมูล 432 ไบต์ไปยังเลเยอร์ดาต้าลิงค์ ที่เลเยอร์นี้จะมีการเพิ่มส่วนหัวอีก 14 ไบต์ และส่วนหางอีก 4 ไบต์ ดังรูปที่ 2.4



รูปที่ 2.4 ลำดับการเอ็นแคปซูเลชั่น

2.1.5 ลักษณะของการติดต่อ แบ่งออกเป็น 2 ชนิด คือ

2.1.5.1 Connection- oriented คือ การติดต่อที่ต้องมีการเชื่อมโปรเซสที่จะทำการติดต่อก่อนที่จะมีการส่งหรือรับข้อมูล ซึ่งสามารถใช้คำว่าวงจรเสมือน (Virtual Circuit) เพราะว่าจะทำงานเสมือนมีวงจรต่ออยู่ระหว่างโปรเซส ถึงแม้ว่าข้อมูลนี้อาจจะผ่าน Packet-Switching Network บริการชนิดนี้ส่วนมากจะใช้ในกรณีที่มีข่าวสารต้องการมากกว่าหนึ่งข่าวสาร ดังนั้นสามารถแบ่งขั้นการทำงานออกเป็น

- ขั้นการสร้างการติดต่อ (connection establishment)

- ขั้นการส่งผ่านข้อมูล (data transfer)
- ขั้นยกเลิกการติดต่อ (connection termination)

2.1.5.2 Connectionless หรือคาต้าแกรม (Datagram) คือจะไม่มีขั้นการสร้างการติดต่อ และขั้นยกเลิกการติดต่อ แต่จะมีขั้นการส่งผ่านข้อมูลอย่างเดียว โดยข้อมูลเรียกว่าคาต้าแกรม จะถูกส่งจากระบบหนึ่งไปสู่ระบบหนึ่งอย่างเป็นอิสระ โดยไม่ขึ้นอยู่กับคาต้าแกรมอื่น

2.1.6 แอดเดรส (Address)

การที่ระบบในระบบเครือข่ายสามารถติดต่อกันได้จำเป็นต้องมีแอดเดรสไว้คล้ายกับหมายเลขประจำตัว ซึ่งลำดับของแอดเดรสสามารถพิจารณาได้คือ

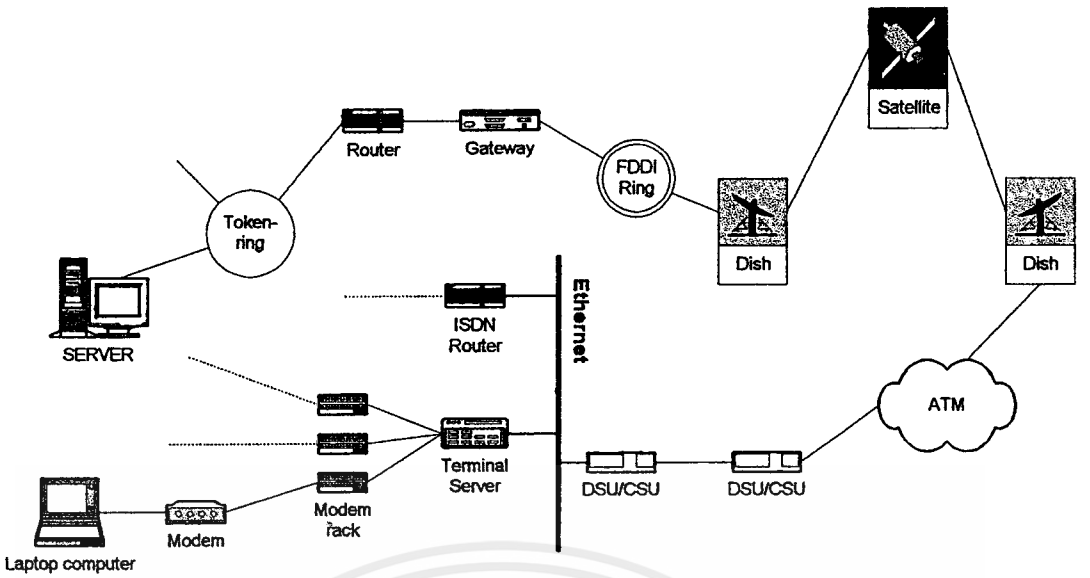
- แต่ละเครือข่ายจะต้องมีแอดเดรสสำหรับเครือข่าย
- คอมพิวเตอร์โฮสต์แต่ละเครื่องในเครือข่ายจะต้องมีแอดเดรส
- แต่ละโพรเซสในโฮสต์จะต้องมีหมายเลขประจำตัว

โดยทั่วไปแอดเดรสของโฮสต์จะประกอบด้วยหมายเลขเครือข่าย (Network ID) และหมายเลขของโฮสต์ (Host ID) ส่วนแอดเดรสของโพรเซสของผู้ใช้จะอยู่ในรูปจำนวนเต็มซึ่งกำหนดโดยโปรโตคอล เช่น โปรโตคอล TCP/IP จะใช้เลขจำนวนเต็มขนาด 32 บิต ในการกำหนดหมายเลขเครือข่าย และหมายเลขของโฮสต์ และทั้ง TCP และ UDP ใช้เลขจำนวนเต็มขนาด 16 บิต เป็นหมายเลขพอร์ต หรือหมายเลขของโพรเซส

ชุดของโปรโตคอลส่วนใหญ่ จะมีการกำหนดชุดของแอดเดรส สำหรับการบริการที่เป็นที่รู้กันโดยทั่วกัน เช่น คอมพิวเตอร์ที่โปรโตคอล TCP/IP ส่วนใหญ่จะมี FTP ซึ่งไคลเอนต์สามารถติดต่อได้โดยใช้หมายเลขพอร์ตคือ 21

2.2 อินเตอร์เน็ต

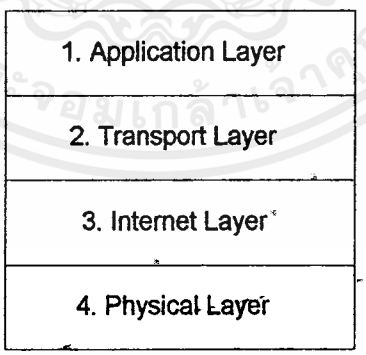
อินเตอร์เน็ตเป็นที่รวมของระบบเน็ตเวิร์ค ที่มากกว่า 1 เน็ตเวิร์ค โดยมีการเชื่อมต่อ พื้นฐานด้วยเน็ตเวิร์คโปรโตคอล อินเตอร์เน็ตมีโปรโตคอล TCP/IP เป็นลักษณะเฉพาะในการติดต่อของระบบเครือข่ายที่กว้าง โดยประกอบด้วย ไอพี แอดเดรส และ โปรโตคอล ที่เป็นพื้นฐานในการติดต่อระบบ อินเตอร์เน็ต



รูปที่ 2.5 An Internet

2.2.1 เปรียบเทียบระหว่างโปรโตคอล TCP/ IP และ OSI โมเดล

การออกแบบโปรโตคอล TCP/IP นั้นไม่ได้เป็นไปตามรูปแบบของ OSI โมเดล เนื่องด้วยถูกออกแบบโดยองค์กรขนาดใหญ่ ซึ่งใช้เวลานานในการออกแบบ ตลอดจนการรับรองมาตรฐานต่างกับโปรโตคอล TCP/IP ที่ถูกออกแบบด้วยความต้องการอันเร่งด่วนของรัฐบาลสหรัฐ จึงทำให้การพัฒนาโปรโตคอล TCP/IP มีเงื่อนไขในด้านความต้องการที่ต่างจาก OSI โมเดล ซึ่งหากเรามองโดยรวมแล้วจะเห็นว่าโปรโตคอล TCP/IP มีการแบ่งเป็นเลเยอร์ที่น้อยกว่า OSI โมเดล คือมี 4 ชั้นเท่านั้น ดังรูปที่ 2.6 โดยแบ่งเป็น



รูปที่ 2.6 เลเยอร์ของโปรโตคอล TCP/IP

2.2.1.1 แอปพลิเคชั่นเลเยอร์

ในเลเยอร์นี้ประกอบด้วยโปรแกรมประยุกต์ที่ใช้เครือข่าย เช่น โปรแกรมส่งถ่ายข้อมูล (file-transfer program) และอาจกล่าวได้ว่าเลเยอร์นี้โปรโตคอล TCP/IP ก็คือ เลเยอร์ในชั้นแอปพลิเคชั่นเล

เยอร์รวมกับชั้นพรีเซนเตชั่นเลเยอร์ ใน OSIโมเดลนั่นเอง และในเลเยอร์ชั้นนี้ของโปรโตคอล TCP/IP จะกลืนอยู่ในตัวโปรแกรมประยุกต์

2.2.1.2 ทรานสปอร์ตเลเยอร์

ในชั้นนี้เป็นชั้นที่ให้การส่งข้อมูลจากจุดปลายถึงจุดปลาย หากเปรียบเทียบกับ OSI โมเดล ก็สามารถเทียบได้กับชั้นเซสชั่นเลเยอร์ รวมกับทรานสปอร์ตเลเยอร์นั่นเอง โดยโปรโตคอล TCP/IP มีซอกเก็ต เป็นจุดปลาย (end-point) ในการสื่อสาร ซึ่งซอกเก็ตนี้ประกอบไปด้วยหมายเลขของคอมพิวเตอร์ และหมายเลขพอร์ต (port ID) ของเครื่องที่ต้องการส่งข้อมูลไปถึง ในชั้นนี้มีการรับรองการถึงที่หมาย และลำดับของข้อมูลที่ส่งโดยไม่ซ้ำ และความผิดพลาดของข้อมูล

2.2.1.3 อินเตอร์เน็ตเลเยอร์

เลเยอร์นี้มีการกำหนดดาต้าแกรม และทำการหาเส้นทางการส่ง หน้าที่ของเลเยอร์นี้เทียบเท่ากับเน็ตเวิร์คเลเยอร์ และดาต้าลิงค์เลเยอร์ ของ OSI โมเดล

2.2.1.4 ฟิสิคอลเลเยอร์

โปรโตคอล TCP/IP ไม่ได้กำหนดรูปแบบของการเชื่อมต่อในระดับนี้ไว้ใหม่ แต่ได้ใช้มาตรฐานที่มีอยู่เดิมที่กำหนดไว้ก่อน เช่น RS232, อีเทอร์เน็ต เป็นต้น

2.2.2 รูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP/IP

ในหัวข้อนี้จะได้อธิบายรูปแบบการกำหนดแอดเดรสของโปรโตคอล TCP/IP ซึ่งลักษณะแอดเดรสของโปรโตคอลนี้ ค่าของแอดเดรสของเครื่องคอมพิวเตอร์ในระบบเครือข่ายจะไม่ซ้ำกันเลย โดยเลขนี้เรียกว่า IP แอดเดรส เป็นเลข 32 บิต IP แอดเดรสจะอยู่ในรูปของ x.x.x.x. ซึ่งแต่ละ x จะเป็น 1 ไบต์ เช่น 152.2.254.81 ทุกๆ IP แอดเดรส จะสามารถแบ่งออกเป็นเครือข่ายได้ โดยแบ่งเป็นบล็อกของกลุ่ม IP แอดเดรสเพื่อจุดประสงค์ของการดูแลและปรับปรุง ซึ่งแบ่งเป็นคลาส ตามหลักในการพิจารณาที่จะได้กล่าวต่อไปนี้

2.2.2.1 การแบ่งเน็ตเวิร์คคลาส

เนื่องจากหมายเลขแอดเดรสของคอมพิวเตอร์เครื่องใดๆ นั้นจะต้องสามารถบอกถึง ความแตกต่างระหว่างตัวเครื่องเอง ตลอดจนถึงเครือข่ายที่คอมพิวเตอร์นั้นเชื่อมต่ออยู่ด้วย หมายเลข IP แอดเดรสจึงแยกออกเป็น 2 ส่วน ได้แก่ ส่วนที่แสดงหมายเลขของคอมพิวเตอร์โฮสต์ และส่วนที่เป็นหมายเลขของเครือข่าย

การแบ่งคลาสของแอดเดรสทำได้โดยพิจารณาจำนวนบิตของ 2 ส่วนประกอบข้างต้น ซึ่งมีการแบ่งออกเป็น 5 คลาส แต่มีการใช้เพียง 3 คลาสแรก คือ คลาส A , คลาส B และคลาส C ส่วนคลาส D และคลาส E ถูกสงวนไว้สำหรับจุดประสงค์พิเศษ

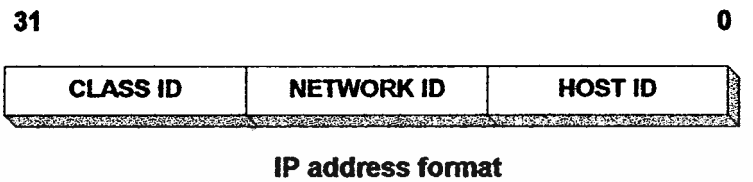
คลาส A : 1-126 (16 M โฮสต์ในแต่ละอัน)

คลาส B: 128-191 (65536 โฮสต์ในแต่ละอัน)

คลาส C: 192-223 (256 โฮสต์ในแต่ละอัน)

คลาส D: 224-239 (โหมด multicast)

คลาส E: 240-255 (สำรองไว้สำหรับใช้ในอนาคต)

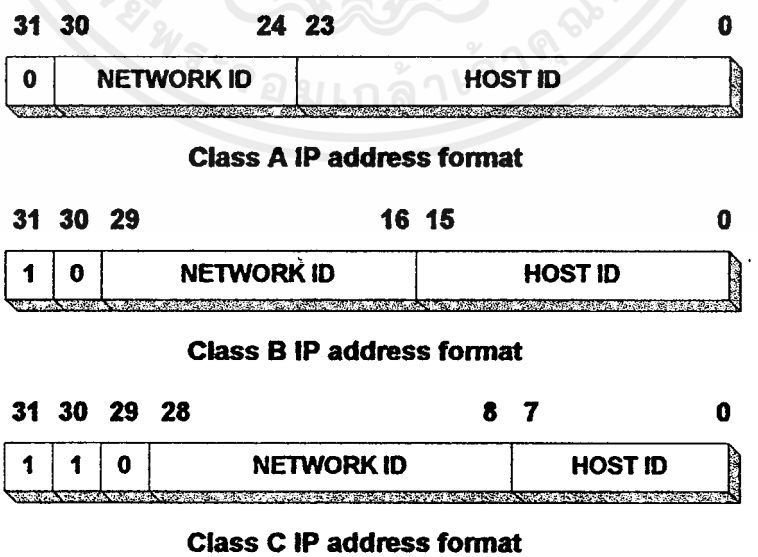


รูปที่2.7 หมายเลข IP แอดเดรส

Network Class	Networks	Hosts per Network
A	126	16,777,214
B	16,328	65,534
C	2,097,150	254

รูปที่ 2.8 แสดงคลาส,จำนวนเครือข่าย และจำนวนโฮสต์ของแต่ละคลาส

โดยปกติแล้วผู้พัฒนาโปรแกรมไม่ต้องสนใจความแตกต่างระหว่างคลาสของ IP แอดเดรส



รูปที่ 2.9 แสดงรูปแบบของ IP แอดเดรสในคลาสต่างๆ

2.2.2.2 การแทนด้วยเลขฐานเลขสิบ และจุด (Dotted Decimal Notation)

เนื่องจากการแทนหมายเลข IP แอดเดรสเป็นเลขฐาน 2 ซึ่งค่อนข้างอ่านไม่สะดวกจึงมีการแทนเลขฐานสองเหล่านั้นในรูปเลขฐานสิบ และจุด โดยเลขฐานสิบแต่ละตัวจะแทนเลขฐานสองจำนวน 8 บิต โดยระหว่างเลขฐานสิบแต่ละตัวจะแทรกด้วยจุด ดังนั้นจะต้องใช้เลขฐานสิบ 4 ตัว ในการแทนเลข 32 บิต ที่เป็น IP แอดเดรส ดังตัวอย่างตามรูปที่ 2.10 จะสังเกตว่าหมายเลข IP แอดเดรสนี้จัดอยู่ในคลาส B โดยหมายเลขของเครือข่าย 166.78 และหมายเลขประจำตัวเครื่อง 4.139

Dotted Decimal Notation
166.78.4.139
Binary Representation
10100110 01001110 00000100 10001011

รูปที่ 2.10 ตัวอย่างหมายเลข IP แอดเดรสทั้งสองแบบ

2.2.2.3 การทำซับเน็ตติ้ง (Subnetting)

IP จะใช้ประโยชน์ในซับเน็ต ซึ่งจะเป็นการแบ่งทางลอจิก ของเครือข่ายให้เป็นเครือข่ายขนาดเล็กๆ ซึ่งแต่ละเครือข่ายจะประกอบด้วยขอบเขตของแอดเดรสจาก IP ต้นแบบ ซับเน็ตเป็นสิ่งจำเป็นเพื่อที่จะออกแบบเครือข่าย IP ซึ่งมันจะทำมาจากมากกว่า 1 broadcast base network เช่น อีเทอร์เน็ต แต่ละซับเน็ตจะบรรจุ broadcast แอดเดรสของมัน

การทำซับเน็ตติ้งเป็นการเปลี่ยนแปลงการใช้หมายเลขของเครื่องโฮสต์ และหมายเลขของเครือข่ายในระดับท้องถิ่น โดยในทางตรรกคือการเลื่อนเส้นแบ่งที่แยกหมายเลขเครื่อง และหมายเลขของเน็ตเวิร์กที่อยู่ในหมายเลข IP แอดเดรส โดยที่ปริมาณของหมายเลขเครื่องโฮสต์ และหมายเลขเครือข่ายจะแปรผกผันกัน เช่น หากมีปริมาณของเน็ตเวิร์กมาก ก็จะทำให้เครื่องใดๆที่จะต่อกับระบบเครือข่ายหนึ่งๆ จะน้อยลงเป็นต้น ในทางปฏิบัติการทำซับเน็ตติ้งทำโดยการนำซับเน็ตติ้งมาร์ค (Subnet mark) คือตัวเลขจำนวน 32 บิต มาทำการกระทำตรรกและ (AND) กับหมายเลข IP แอดเดรส ดังตัวอย่างโดยกำหนดหมายเลข IP แอดเดรส คือ 166.78.4.139 และซับเน็ตมาร์ค คือ 255.255.255.0 ทำการกระทำตรรกและ ดังรูป

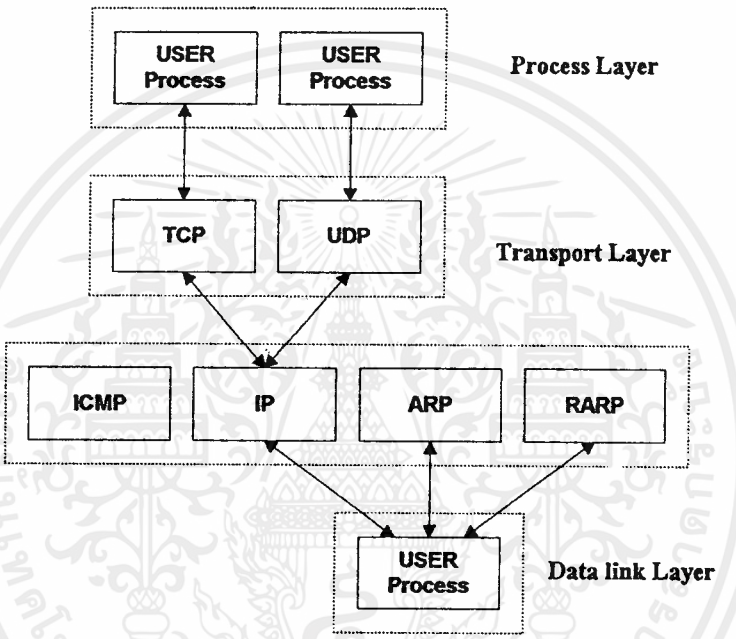
166.78.4.139
AND
255.255.255.0
166.78.4.0

รูปที่ 2.11 การทำซับเน็ตติ้ง

จะเห็นว่าหากพิจารณาโดยไม่มีการทำซับเน็ตดิ้งแล้วจะได้หมายเลขของเน็ตเวิร์กคือ 166.78 และหมายเลขประจำเครื่องคือ 4.139 แต่ผลที่ได้จากการกระทำตรรกและ ในรูปข้างต้น ผลลัพธ์ นี้คือหมายเลขเน็ตเวิร์กคือ 166.78.4.0 และหมายเลขเครื่องคือ 139 หรืออาจพูดได้ว่าคอมพิวเตอร์เครื่องนี้มีหมายเลขเครื่องเท่ากับ 139 และอยู่บนเครือข่ายย่อยหมายเลข 166.78.4

2.2.3 ชุดโปรโตคอลTCP/IP

ชุดโปรโตคอล TCP/IP นอกจากมีโปรโตคอล TCP และ IP แล้ว ยังมีโปรโตคอลอย่างอื่นอีก ดังรูปที่ 2.12 แสดง ความสัมพันธ์ของชุดโปรโตคอลโดยแบ่งตามเลเยอร์



รูปที่ 2.12 ความสัมพันธ์ระหว่างชุดโปรโตคอล

2.2.3.1 โปรโตคอล IP (Internet Protocol)

โปรโตคอล IP เป็นโปรโตคอลแบบคอนเนคชั่นเลส (Connectionless protocol) ซึ่งได้กล่าวถึงลักษณะของโปรโตคอลชนิดนี้ไปแล้วในหัวข้อ 2.1.5 โดยที่โปรโตคอล IP ไม่รับประกันว่าข้อมูลที่ส่งจะไปถึงปลายทางซึ่งแพคเก็ต ของข้อมูลอาจไปถึงในลักษณะที่ผิดลำดับ,ซ้ำกัน หรือไม่ไปถึงเลย โดยความน่าเชื่อถือของการส่งจะถูกควบคุมในโปรโตคอลในเลเยอร์ต่างๆ ไป การหาเส้นทางของข้อมูลจะทำในระดับของโปรโตคอล IP นี้ โดยพิจารณาแต่ละแพคเก็ตแยกออกจากกัน และยังมีหน้าที่ในการจัดเรียงข้อมูลใหม่ที่ปลายทางอีกด้วย

2.2.3.2 โปรโตคอล ARP (Address Resolution Protocol)

โปรโตคอลนี้ทำหน้าที่จับคู่ ระหว่างหมายเลข IP แอดเดรสเข้ากับ หมายเลขแอดเดรสทางฮาร์ดแวร์ โดยโปรโตคอลนี้ทำการส่งข้อความไปทั่วเครือข่ายท้องถิ่น ซึ่งข้อความนี้เป็นลักษณะข้อความที่ตรวจสอบว่ามีคอมพิวเตอร์ที่มีหมายเลข IP แอดเดรสตรงกับที่ต้องการหาหรือไม่ หากคอมพิวเตอร์ที่มีหมายเลข IP แอดเดรสตรงกันนั้นได้รับข้อความนี้ ก็จะตอบกลับ และเป็นที่น่าสังเกตว่าโปรโตคอลนี้ทำงานได้กับระบบเครือข่ายท้องถิ่นเท่านั้นเพราะว่าโครงสร้างหมายเลขทางฮาร์ดแวร์ของเครื่องคอมพิวเตอร์จะขึ้นอยู่กับชนิดของระบบเครือข่ายด้วย

2.2.3.4 โปรโตคอล RARP (Reverse Address Resolution Protocol)

เป็นโปรโตคอลที่ทำหน้าที่จับคู่ระหว่างหมายเลขของฮาร์ดแวร์กับหมายเลข IP แอดเดรส หรือทำงานกลับกันเป็นโปรโตคอล ARP

2.2.3.5 IP routing

ตามแนวความคิด เครือข่าย IP จะอยู่บนพื้นฐานของโฮสต์และ routers อุปกรณ์โฮสต์ IP จะรวบรวมและทำการส่งควบคุมการไหล การตรวจสอบความผิดพลาด (ถ้ามี) และกระบวนการข้อมูลอื่นๆ router เป็นตัวหาเส้นทางที่ดีที่สุด เพื่อใช้ในการติดต่อของระบบเครือข่าย

กุญแจที่นำไปสู่ IP คือความสามารถในการหาเส้นทางของมัน ไอพี IP router มีหลายๆเครือข่ายเชื่อมต่อ , และตรวจสอบแต่ละแพคเก็ต มันจะคำนวณเส้นทางที่มีราคาน้อยที่สุดที่จะไปยังปลายทางโดยวิธีโปรโตคอล routing ของมัน และส่งแพคเก็ตเส้นทางที่เหมาะสมไปสู่ปลายทาง router อย่างง่ายไม่ต้องการความรู้ทั้งหมดของเครือข่าย แต่ต้องรู้การเชื่อมต่อท้องถิ่นของมัน แพคเก็ตควรจะถูกส่งไปข้างหน้าบนพื้นฐานแอดเดรสปลายทาง

2.2.3.6 โปรโตคอล UDP

โปรโตคอลนี้ เป็นโปรโตคอลที่อยู่ในระดับ ทรานสปอร์ตเลเยอร์ และ มีความสำคัญ เพราะว่าเป็นโปรโตคอล ที่ผู้พัฒนาโปรแกรมสามารถใช้ได้โดยตรงโปรโตคอลนี้เป็นโปรโตคอลแบบคอนเนคชั่นเลสมีความน่าเชื่อต่ำ ไม่รับรองว่าข้อมูลที่ส่งไปจะไปถึงปลายทางหรือไม่ และอาจซ้ำซ้อน หรือผิดลำดับได้ แต่ข้อดีของโปรโตคอลนี้ คือ ค่าความสิ้นเปลือง (overhead) ที่ต่ำ

2.2.3.7 โปรโตคอล TCP

โปรโตคอลนี้อยู่ในระดับทรานสปอร์ตเลเยอร์ เหมือนกับโปรโตคอล UDP แต่มีลักษณะที่ตรงข้ามกันคือ เป็นโปรโตคอลแบบคอนเนคชั่นออเรียนเต็ด โดยที่มีความน่าเชื่อถือในการรับส่งข้อมูล และลำดับของข้อมูลจะมีลำดับเหมือนกับต้นทางและเนื้อข้อมูลไม่ผิดพลาด จึงทำให้เกิดความสิ้นเปลืองในการเชื่อมต่อของการส่งข้อมูลมากกว่าโปรโตคอล UDP

2.2.4 หมายเลขพอร์ต

เนื่องจากในเวลาใดๆ สามารถมีโพรเซสของผู้ใช้สามารถใช้ UDP หรือ TCP ได้พร้อมกันดังนั้นจึงต้องมีวิธีแยกแยะว่าข้อมูลเป็นของผู้ใช้คนใด ซึ่งวิธีที่ TCP และ UDP ใช้คือการใช้หมายเลขพอร์ต

เมื่อโพรเซสไคลเอนต์ ต้องการที่จะติดต่อกับเซิร์ฟเวอร์ ไคลเอนต์จะต้องเจาะจงเซิร์ฟเวอร์ ที่ต้องการติดต่อแต่ลำพังรู้แอดเดรสอินเตอร์เน็ต 32 บิตเพียงอย่างเดียวนั้นไม่เพียงพอ เพราะว่าสามารถติดต่อกับโฮสต์ได้เพียงอย่างเดียวแต่ไม่สามารถเจาะจงโพรเซสที่จะทำการติดต่อได้ ดังนั้นเพื่อแก้ปัญหานี้ทั้ง TCP และ UDP ได้มีการกำหนดหมายเลขพอร์ตมาตรฐาน (well-known ports) ซึ่งไคลเอนต์เลิกใช้หมายเลขพอร์ตนี้แล้ว สามารถกำหนดหมายเลขพอร์ตนี้ให้ไคลเอนต์อื่นได้โพรเซสที่ได้รับหมายเลขพอร์ตชั่วคราวนี้จะไม่สนใจว่ามีค่าเท่าไร แต่เป็นหน้าที่ของอีกโพรเซสหนึ่งที่ต่อกันที่ต้องสนใจ เพราะต้องส่งข้อมูลกลับมาที่พอร์ตนี้ใน TCPและ UDP นั้นหมายเลขพอร์ตตั้งแต่ 1-1023 เป็นพอร์ตที่สงวนไว้สำหรับหมายเลขพอร์ตมาตรฐาน

บทที่ 3
การเขียนโปรแกรมบนเครือข่าย

ตัวกลางเชื่อมต่อใช้งานโปรแกรมร่วม (COMMON GATEWAY INTERFACE, CGI)

เป็นส่วนที่กำหนดรูปแบบการติดต่อระหว่างผู้ให้บริการเว็บ (เว็บเซิร์ฟเวอร์) กับโปรแกรมภายนอก (แอปพลิเคชั่น) โดยรับรายการจากแฟ้มข้อมูลของเว็บเพจ โดยที่มีเกตเวย์ ทำหน้าที่เป็นตัวกลางในการเชื่อมต่อ

เกตเวย์ สามารถที่ จะออกแบบ และใช้งานได้หลายประโยชน์ โดยทั่วไป จะใช้ป้าย (TAG)<ISINDEX> และ ป้าย <FORM> ใน HTTP เป็นรูปแบบในการทำ คิวรี่ รับ-ส่งข้อมูลจากผู้ใช้บริการ กับ ผู้ให้บริการ โดยมี ซีจีไอ เป็นตัวกลางทำหน้าที่ในการรับ-ส่งข้อมูลให้ ซึ่งจะใช้ ซีจีไอ เมื่อ

1. ใช้ในการป้อนข้อมูลที่ผู้ใช้บริการตอบคำถามกลับมายังผู้ให้บริการ โดยใช้ฟอร์มของเฮชทีเอ็มแอล
2. ใช้เปลี่ยนเอกสารที่มีในระบบที่เปลี่ยนแปลง (Dynamic) ได้ ให้เป็น เฮชทีเอ็มแอล เพื่อให้ง่ายต่อการค้นหาในเว็บเบส
3. ใช้เป็นการตั้งคำถามที่ต้องการใช้ฐานข้อมูล และให้ผลเป็นเอกสารของเฮชทีเอ็มแอล

เนื่องจาก การบริการพื้นฐานของ เวิลดไซด์เว็บมีการเรียกขอรายการ (Request) จากเครื่องที่ให้บริการ ซึ่งมีการจัดเก็บโดยแยกกันเป็นส่วนๆ ภายใต้ไดเร็กทอรี่ ในเครื่องผู้ให้บริการ โดยความสัมพันธ์ระหว่างแฟ้มข้อมูลจะอาศัยตัวเชื่อม (Link) ระหว่างเอกสารเท่านั้น บางครั้งต้องการเป็นงานในลักษณะสรุปเรื่องใดเรื่องหนึ่ง ซึ่งจะได้จากฐานข้อมูล หรือต้องการใช้เฮชทีเอ็มแอล เป็นส่วนที่ติดต่อกับผู้ใช้โดยตรง โดยการใช้ซีจีไอนี้ ซึ่งจะรับการทำงานของผู้ให้บริการเว็บ การพัฒนางานเพื่อให้มีความสามารถดีขึ้นได้ แก้ไขรายการโดยไม่มีผลกระทบต่อฐานข้อมูล, ผู้ใช้ไม่ต้องเรียนรู้ถึงการใช้งานใหม่กับเครื่องต่างระบบกันได้ การเขียนโปรแกรมใช้งานซีจีไอ (ซีจีไอ แอพพลิเคชัน) จะใช้ภาษาที่มีลักษณะเป็นภาษาสคริปต์ที่มีโครงสร้างไม่ซับซ้อน เช่นในระบบยูนิกซ์ อาจใช้ภาษาเพิร์ล (Perl) ทีซีแอลทีเค (TCL/TK) , แพนทอม (Pythom) และยูนิกซ์สคริปท์เชลล์ ก็ได้ ในระบบดอส อาจใช้ภาษาซี , ซีพลัสพลัส , ปาสคาล , วิชวลเบสิค ก็ได้ หรืออาจกล่าวว่า การเขียนโปรแกรมใช้งานซีจีไอสามารถที่จะเขียนด้วยภาษาใดก็ได้ ที่มีส่วนติดต่อการทำงาน์กับภายในภายนอกที่เป็นมาตรฐานชนิดเดียวกัน (Standard input/output) เป็นการทำงานในการส่งค่าข้อมูลได้ และขึ้นกับการทำงานของผู้ให้บริการที่ใช้งานในขณะนั้นๆ และภาษาที่นำมาใช้ควรเป็นภาษาที่สามารถใช้ประมวลผลข้อความ และจัดการกับแฟ้มข้อมูลได้สะดวก อีกทั้งมีมาตรฐานความปลอดภัยของข้อมูลพอสมควร ในระบบที่ทำงานด้วยผู้ให้บริการเฮชทีทีพี (HTTP Server) มีโปรแกรมทำงานที่ใช้ซีจีไอในการทำงานนั้น ด้านผู้ใช้บริการจะต้องมีการขอใช้งานมายังผู้ให้บริการ และเฮชทีทีพีจะได้รับเป็นข้อมูลประกอบด้วย

1. ที่อยู่ของเอกสาร (Uniform Resource Identifier , URI)
2. วิธีการรับข้อมูล

3. ข้อมูลที่สำคัญต่างๆ ที่ต้องการโดยใช้ซีจีไอช่วย

3.1 หน้าที่ของซีจีไอสคริปต์ (CGI Application Script)

ตามปกติแล้ว ผู้ให้บริการเว็บได้ค่าที่อยู่ของเอกสารจากการกดเลือกรายการที่มี หรือใส่ที่อยู่ในรายการ จากนั้นผู้ให้บริการเว็บจะทำการติดต่อไปยังผู้ให้บริการเฮชทีทีพี ตามที่อยู่ของเอกสารที่ระบุไว้นั้น เมื่อเซิร์ฟเวอร์ได้รับการขอแฟ้มข้อมูลเข้ามา ก็จะทำการส่งแฟ้มข้อมูลนั้นกลับไปให้ บางครั้งในการขอแฟ้มข้อมูลอาจจะต้องมีการตรวจสอบผู้ใช้ก่อน โปรแกรมนี้จะใช้ซีจีไอทำหน้าที่นี้ โดยในผู้ให้บริการเว็บจะสามารถดูได้จากไดเรกทอรี่ ที่ชื่อ /cgi-bin/ นอกจากนี้ยังสามารถใช้ซีจีไอนี้ในการนับจำนวนผู้ที่เข้ามาดูเว็บเพจ ของผู้ให้บริการเว็บที่ให้บริการอยู่ ส่วนที่ติดต่อกับผู้ใช้งานเว็บจะทำหน้าที่เพียงรับรายการจากผู้ใช้เท่านั้น แต่จะมีการประมวลผลก็ต่อเมื่อ ส่งมาถึงเครื่องผู้ให้บริการเว็บแล้วเท่านั้น ซึ่งการทำงานแบบผู้ใช้บริการและผู้ให้บริการนี้ ส่วนที่ติดต่อกับผู้ใช้ คือ โปรแกรมที่ใช้ดูเว็บ (เว็บบราวเซอร์) จะทำหน้าที่รับรายการจากผู้ใช้ แล้วส่งต่อให้ผู้ให้บริการเว็บ เมื่อผู้ให้บริการเว็บได้รับข้อมูลเป็นอินพุท ในรูปการร้องขอ เฮชทีทีพี (HTTP Request) ก็จะไปทำการเรียกโปรแกรมซีจีไอ โดยจะส่งค่าตัวแปร (Parameter) ผ่านต่อให้กับซีจีไอนี้ ซึ่งจะทำการแปลความหมาย (Parse) ข้อมูลที่รับเข้ามา เช่น ถ้าเป็นการขออนุญาตเข้าไปใช้งานใน ไดเรกทอรี่ที่ระบุผู้ใช้งานไว้ จะมีการส่งค่าที่เป็น ชื่อผู้ใช้ (User name) และรหัสผ่าน (Password) ให้ผู้ขอใช้ป้อนกลับมา แล้วทำการตรวจสอบว่าถูกต้องหรือไม่ ถ้าใช่ ก็จะส่งข้อมูลที่ต้องการให้หรือ ถ้าเป็นการสอบถาม รายการ ค่าตัวแปร ก็จะเป็นตัวบอกของรายการที่สอบถาม ซีจีไอก็จะนำค่านี้มาค้นหารายการในฐานข้อมูล จากนั้นฐานข้อมูลจะส่งข้อมูลที่ต้องการกลับมายังซีจีไอ โปรแกรมซีจีไอจะทำการสร้างแฟ้มข้อมูลในแบบเฮชทีเอ็มแอล ใส่ผลลัพธ์ที่ได้จากการสอบถามจากฐานข้อมูลไว้ในแฟ้มข้อมูลนี้ แล้วส่งกลับมาให้ผู้ให้บริการเว็บ เพื่อให้ผู้ให้บริการส่งกลับมายัง โปรแกรมที่ใช้ดูเอกสารให้ผู้ใช้ทราบต่อไป แต่รายการที่ส่งกลับมาจะมีการระบุชนิดของแฟ้มข้อมูลนั้นไว้ด้วย โดยผู้ให้บริการ จะใส่ค่าตัวแปรที่ใช้งานหลัก (Environment Variable) ชนิดการบอกการใช้งานเอกสาร(Content_type) เพื่อบอกชนิดของแฟ้มข้อมูล ไว้ที่ส่วนหน้าสุดของแฟ้มข้อมูล (Header Field) ด้วย

3.2 ตัวแปรรอบด้าน (Environment Variable)

คือ สิ่งที่เราสนใจ (Entity) ที่อยู่ในส่วนใช้งานกับสิ่งรอบด้านของเครื่องคอมพิวเตอร์ ของผู้ใช้งานที่ใช้งาน ซึ่งซีจีไอเป็นตัวผ่านข้อมูลของเฮชทีทีพี ไปยังเครื่องที่ให้บริการโดยมีโปรแกรมการทำงานของซีจีไอทำหน้าที่นี้ โดยที่ค่าตัวแปรจะถูกค้นหาได้ทั้งจาก เครื่องที่ให้บริการ และ โปรแกรมการทำงานของซีจีไอเอง

ผู้ให้บริการเว็บ ใช้ค่าตัวแปรที่ทำงานทีละบรรทัด (Command Line Argument) ที่เรียกว่า ค่าตัวแปรรอบด้าน เป็นตัวผ่านข้อมูลที่ต้องการจากผู้ใช้บริการ โดยมีการกำหนดหรือควบคุมค่าตัวแปรของโปรแกรม หรือที่จุดเริ่มของโปรแกรมได้ ซึ่งวิธีการนี้ทำให้ผู้ให้บริการผ่านการรับข้อมูล ที่เป็นการต่อเนื่องสั้นๆ ไปยังโปรแกรมการทำงานของซีจีไอที่เกี่ยวข้องกัน ในทางเดียวกัน ตัวแปรเหล่านี้จะถูกจัด

หรือกำหนดให้เป็นค่าใดๆ เมื่อเครื่องที่ให้บริการเริ่มทำงานกับโปรแกรมการทำงานของ ซีจีไอ โดยที่ค่าตัวแปรรอบด้าน แบ่ง ออกเป็น 3 ประเภท มีดังนี้

ตัวแปรที่เกี่ยวกับข้อมูลของผู้ให้บริการ (Server Information Environment)

3.2.1 SERVER_NAME : คือ ชื่อของเครื่อง ที่เป็นเครื่องที่ให้บริการ อาจจะเป็น ดีเอ็นเอส (DNS) หรือตำแหน่งในอินเตอร์เน็ตก็ได้ ที่จะปรากฏในตำแหน่งที่เก็บเอกสาร

3.2.2 SERVER_PORT : ใช้ในการบอกถึงส่วนที่ใช้ติดต่อ (พอร์ต) ที่เครื่องที่ใช้บริการสามารถที่จะติดต่อกันได้กับเครื่องที่ให้บริการ ปกติจะกำหนดไว้เป็นค่าเท่ากับ 80

3.2.3 SERVER_PROTOCOL : ใช้บอกถึงมาตรฐาน หรือข้อตกลงที่ใช้ในการติดต่อ โดยจะแสดงถึงมาตรฐานของเครื่องที่ให้บริการที่ใช้ เช่น SERVER_PROTOCOL = HTTP/1.0

3.2.4 SERVER_SOFTWARE :ใช้แสดงชื่อและรุ่น ของข้อมูล ซอฟต์แวร์ที่มีในเครื่องที่ให้บริการ เพื่อให้เครื่องที่ใช้บริการทราบ เช่น SERVER_SOFTWARE = NCSA/1.4

ตัวแปรที่เกี่ยวกับข้อมูลของผู้ใช้บริการ (Client Information Environment)

3.2.5 REMOTE_ADDR :ใช้แสดงตำแหน่งในอินเตอร์เน็ต (ไอพี แอดเดรส) ของเครื่องที่ให้บริการ ที่ใช้งาน เพื่อให้เครื่องที่ใช้บริการสามารถที่จะติดต่อได้ เช่น REMOTE_ADDR = 199.1.78.25

3.2.6 REMOTE_HOST :ใช้บอกชื่อเครื่อง หรือระบบชื่อหลัก (Domain Name System , DNS) ของเครื่อง เครื่องที่ให้บริการที่ใช้งาน เพื่อให้เครื่องที่ใช้บริการทราบในการติดต่อด้วย ถ้าไม่มีการกำหนดจะให้เป็น นัล เช่น REMOTE_HOST = http://www.kmitl.ac.th

3.2.7 REMOTE_IDENT : ใช้บอกชื่อของผู้ใช้งาน ที่ทำการใช้งานทางไกล (Remote) มายังเครื่องที่ให้บริการ และต้องการเรียกคืน (Retrieved) ซึ่งจะมีการทำการเก็บรายละเอียดที่ทำงาน (Logging) ไว้ด้วย

3.2.8 REMOTE_USER : คือ การกำหนดชื่อของเครื่องผู้ใช้บริการ ใช้ใน 2 กรณี คือ

3.2.8.1 เครื่องที่ให้บริการ มีการติดต่อกับเครื่องที่ใช้บริการนั้นอย่างแน่นอน และบ่อยครั้ง

3.2.8.2 การติดต่อนั้น จะต้องมีการทำเกี่ยวกับระบบความปลอดภัยด้วย โปรแกรมที่ทำงานด้วยซีจีไอ สำหรับผู้ที่ไม่ได้กำหนดให้ไว้ ซึ่งถ้าค่า ออทไทพ์ (AUTH_TYPE) มีค่าเป็น " Basic " ค่าของรีโมทยูสเซอร์ นี้ จะเป็นค่าที่กำหนดเฉพาะผู้ใช้งาน ที่ส่งการร้องขอมายังเครื่องให้บริการได้ เช่น REMOTE_USER = gail_snokes

ตัวแปรที่เกี่ยวกับข้อมูลของสคริปต์ (Script Information Environment)

3.2.9 AUTH_TYPE :ใช้ในการทำระบบป้องกัน (Security) ของผู้ให้บริการ ซึ่งจะบ่งบอกถึงข้อกำหนดที่ผู้ใช้บริการ สามารถที่จะทำงานกับเครื่องที่ให้บริการได้ โดยปกติ HTTP จะมีการกำหนดค่านี้โดยตรงอยู่แล้ว ซึ่งจะกำหนดเป็น นัล ไว้ ในกรณีที่ไม่มีการกำหนด การค้นหาหรือเข้าถึงในเครื่องที่ให้บริการ เช่น

ถ้าเครื่องที่ให้บริการ กำหนดให้ผู้ใช้บริการ มีเลขประจำตัว และรหัสผ่าน ในการใช้งานเว็บของเครื่องที่ให้บริการเอง ทางเครื่องที่ให้บริการจะต้องกำหนดค่า ออทไทพ์ ให้มีค่าตรงกับ

เครื่องที่ให้บริการด้วย เช่น AUTH_TYPE = Basic หมายถึง เครื่องที่ให้บริการกำหนดค่าให้ตรงกับที่เครื่องที่ให้บริการ คือ Basic

3.2.10 CONTENT_LENGTH : เป็นค่าตัวเลขฐานสิบ 2 ค่า เพื่อกำหนดให้ สิ่งที่สนใจอยู่ขณะนั้น สามารถที่จะทำงานกับเครื่องที่ให้บริการ ตามค่านี้เท่านั้น ถ้าไม่กำหนดเอาไว้จะให้เป็นนัล ไว้ เช่นCONTENT_LENGTH = 9

3.2.11 CONTENT_TYPE : เป็นตัวบอกประเภทของสื่อตัวกลางที่ใช้ในเครื่องที่ให้บริการนั้น ถ้าไม่มีการกำหนดไว้จะกำหนดให้เป็นค่า นัล โดยที่เซชทีทีพี 1.0 (HTTP 1.0) กำหนดไว้เป็น MIME (Multipurpose Internet Mail Extention) ซึ่งสามารถขยาย หรือเพิ่มรายละเอียดของข้อมูลได้ดีกว่าแบบอื่นๆ

ในเซชทีทีพี สามารถใช้สื่อที่ต่างกันได้ เช่น ข้อความ , รูปภาพนิ่ง , เสียง และภาพเคลื่อนไหว เป็นต้น โดยที่ในสื่อนั้นๆ ก็ยังมีความแตกต่างกันได้ เช่น รูปภาพนิ่ง อาจเป็นประเภทที่บีบอัดข้อมูลแบบต่างๆ (gif , jpeg , ief , tiff) ซึ่งค่าในคอนเทนไทพ์ (Content_type) ถ้าต้องการกำหนดให้มีรูปภาพเป็นแบบ เจเพ็ก (JPEG) ก็จะกำหนดดังนี้

CONTENT_TYPE = Image/ jpeg โดยที่รูปภาพเป็นชนิดและ เจเพ็กเป็นรูปแบบ นอกจากนี้ยังมีชนิดและรูปแบบอื่นอีก เช่น

Application/Octet-stream เป็นการบอกถึง การส่งข้อมูล โปรแกรมที่ทำงานโดยที่ไม่มีการแปลเป็นข้อมูลไบนารี่ ซึ่งมีประโยชน์ในกรณีที่ผู้ใช้งานทางผู้ใช้บริการ ต้องการที่จะเก็บโปรแกรมที่ทำงานอยู่ลงในฮาร์ดดิสก์ ของเครื่องที่ให้บริการเองได้

Text/Plain ใช้บอกข้อมูลที่เป็นข้อความ โดยไม่มีการจำกัดรูปแบบข้อความ

Multiport/Mix,Alternative ใช้บอกส่วนประกอบต่างๆที่รวมกันของข้อมูลที่เป็นอิสระ

Digest,Parallel มี 4 แบบ คือ

- มิกซ์ (Mix) เป็นแบบทั่วไป
- อัลเทอร์เนทีฟ (Alternative) ใช้แสดงข้อมูลเดียวกัน แต่มีรูปแบบต่างกัน
- ไดเจส (Digest) ใช้บอกส่วนที่เป็นข้อความของข้อมูล
- พาราแรล (Parallel) ใช้บอกส่วนที่ต้องการให้เห็นในเวลาเดียวกัน (พร้อมกัน)

Message/rfc 822,Partial เป็นการแสดงการห่อรวมกันไว้ของข้อมูล โดยที่ข้อความ

External_body ถึงส่วนของการรูปแบบ ข้อมูลที่มีการห่อหุ้มไว้แล้ว และมีวิธีการเป็นชนิดรองอีก คือ

- อาร์เอฟซี 822 (rfc 822) และเป็นวิธีการมาตรฐาน
- พาร์เชี่ยล (Partial) เป็นวิธีการที่แยกส่วนแบบธรรมดา
- เอ็กเทอร์นอล บอดี้ (External_body) ใช้ในการอ้างอิงข้อมูลภายนอก

Image/gif , jpeg ใช้ในการบอกประเภทของรูปภาพนิ่งที่ใช้ในกราฟฟิก เป็นต้น ส่วนของประเภทนิครอง เป็นการบอกประเภทองการบีบอัดข้อมูลรูปภาพนั้น เช่น จีไอเอฟ (Graphic Image Format, gif) เป็นต้น **Audio/Basic,MIDI** เป็นข้อมูลประเภทเสียง ซึ่งชนิครอง จะบอกเป็นอุปกรณ์ทาง แสดงผลของระบบ เช่น Basic หมายถึง ลำโพง หรือ MIDI , X-MIDI โดยที่ เอ็กซ์ (X คือ External Data type) เป็นต้น **Video/mpeg** เป็นข้อมูลที่ใช้กับภาพเคลื่อนไหว ซึ่งขึ้นกับความสามารถของตัวเครื่อง และโปรแกรมที่ใช้

3.2.12 HTTP_* : ใช้ในการขอกการเริ่มต้น ที่เอชทีทีพีและ (HTTP_) ใช้ในการบอกส่วนของเริ่มต้นของข้อมูล เพื่อให้ผู้ใช้บริการสามารถที่จะติดต่อได้ ข้อความที่ต่อจาก เอชทีทีพีและ จะเป็นค่าของชนิดเอกสารด้วย คอนเทนไทท์ ที่กำหนดไว้แล้ว โดยใช้ คอมม่า (Comma [,]) ในการคั่นแต่ละ ค่าตัวแปรที่ต้องการกำหนด เช่น

HTTP_ACCEPT = Aplication/zip, Audio/basic, Image/jpeg, Text/plain เป็นต้น

3.2.13 PATH_INFO : ใช้บอกส่วนที่ต้องการเพิ่มเติมข้อมูล ตามที่เครื่องที่ให้บริการต้องการ หมายถึง คือแหล่งที่จะให้ส่งข้อมูลกลับไป โดยที่ โปรแกรมทำงานด้วย ซีจีไอ ได้ โดยใช้ ชื่อของส่วนที่เก็บไว้ใช้งาน (Pathname) เสมือน (Virtual Pathname) ตามด้วยข้อมูลเพิ่มเติมที่ต้องการ เนื่องจากตำแหน่งที่เก็บเอกสาร (URLs) ใช้วิธีการ โดยการเข้ารหัสพิเศษสำหรับข้อมูลตัวอักษร ในส่วนนี้จะทำการถอดรหัสโดย เครื่องที่ให้บริการ ก่อนที่จะส่งให้ ซีจีไอ ทำงานต่อไป ซึ่งค่าของ พาทอินโฟ (PATH_INFO) นี้ ทำได้ดังนี้

1. ใช้ที่ส่วนปลายของ URL ที่เครื่องที่ใช้บริการติดต่ออยู่
2. ที่ข้อความที่กำหนดจากเครื่องที่ใช้บริการ เพื่อให้เครื่องที่ให้บริการ ใช้ได้
3. ป้อนที่ตำแหน่งของเอกสารโดยตรง จากเครื่องที่ให้บริการ ผู้เขียนโปรแกรมทำงานด้วย ซีจีไอ ต้องกำหนดข้อความที่จะต้องใช้ตัวแปรในโปรแกรมทำงานด้วย ซีจีไอ ด้วย เช่น

PATH_INFO = /raising/bangkok/

3.2.14 PATH_TRANALATED : ใช้บอกส่วนที่เก็บ (Path) ของระบบที่ทำงานกับแฟ้มข้อมูลของเครื่องที่ให้บริการที่จะยอมให้เครื่องที่ใช้บริการ ทำงานกับข้อมูลที่กำหนดจากตำแหน่งที่เก็บเอกสารที่เก็บอยู่ในพาทอินโฟ ซึ่งถ้าเป็นการทำงานด้านระบบความปลอดภัยจะกำหนดให้เป็น นัล ไว้ เช่น

PATH_TRANSLATED = /u/web/raising/bangkok/

3.2.15 QUERY_STRING : ใช้ในการค้นหาข้อความที่เข้ารหัสของตำแหน่งที่เก็บเอกสาร โดยจะมีเครื่องหมายคำถาม (?) ต่อท้ายที่ตำแหน่งที่เก็บเอกสารนั้น ซึ่งซีจีไอจะรู้ว่าเป็นการค้นหา และที่เครื่องที่ให้บริการ จะไม่ทำการถอดรหัสข้อความที่อยู่ต่อจากเครื่องหมายคำถามนี้ แต่จะส่งให้โปรแกรมทำงานด้วยซีจีไอโดยตรง สำหรับข้อมูลหลังเครื่องหมาย " ? " นี้จะสามารถเพิ่ม หรือต่อท้ายได้ ในกรณีต่อไปนี้

1. เป็นข้อมูล เฮชทีเอ็มแอล แบบ <ISINDEX>
2. เป็นโปรแกรมทำงานด้วย ซีจีไอ ที่ใช้เฮชทีเอ็มแอลแบบ <FORM> ชนิดวิธีการแบบ เกต (GET)
3. ในรูปแบบเอกสารที่ผู้เขียน เฮชทีเอ็มแอล อ้างอิงไว้ เช่น การเชื่อมโยงโดยใช้ตัวบอกในการเชื่อมโยง คือ <A> (Anchor Statement)

การเข้ารหัสของคิวรี่สตริง (Query_string) ในตำแหน่งที่เก็บเอกสารมาตรฐานทั่วไป จะทำการเปลี่ยนช่องว่าง (Space) ให้เป็นเครื่องหมาย " + " และจะแปลงหลังตัวอักษรท้ายสุดของ ตำแหน่งที่เก็บเอกสาร (ช่องว่างท้ายข้อความทั้งหมด) ให้เป็น " %dd " เมื่อ d คือ ค่าเลขฐาน 16 ของรหัสแอสกี้ โดยปกติจะกำหนดไว้เมื่อมีการใช้ข้อมูลแบบ คิวรี่จากผู้ใช้บริการ เช่น

http://www.flagstaff.az.us/cgi-bin/place.pl?sunset_crater หมายถึง เครื่องที่ใช้บริการ ต้องการทำคิวรี่เพื่อที่จะให้ ซีจีไอ ทำการค้นหาที่ place.pl ในชื่อ sunset_crater โดยที่ต้องกำหนด QUERY_STRING = Sunset_crater ด้วย

3.2.16 REQUEST_METHOD : ใช้แสดงถึง วิธีการที่เครื่องที่ใช้บริการส่งมาให้เครื่องที่ให้บริการ ทราบเพื่อใช้ในการทำงาน ประกอบด้วย GET, POST, HEAD, PUT, DELETE, LINK, UNLINK เป็นต้น โดยที่จะมีผลต่อการทำงานเป็นอย่างมากในการกำหนดค่าเหล่านี้ เช่น

REQUEST_METHOD = Post หมายถึง ผู้ใช้บริการต้องการข้อมูลหลัง เฮชทีเอ็มแอล ที่รับจากเครื่องให้บริการ เป็นต้น

3.2.17 SCRIPT_NAME :คือ ตำแหน่งที่เก็บเอกสาร ที่กำหนดไว้ในโปรแกรมที่ใช้งานด้วย ซีจีไอ เป็นส่วนเสมือนของโปรแกรมที่ใช้งานด้วย ซีจีไอ ที่เริ่มทำงานจากเครื่องที่ให้บริการ

3.3 การทำงานของซีจีไอกับการรับและแสดงผล (Input/Output CGI)

คาต้าอินพุท เมื่อเครื่องที่ให้บริการได้รับข้อมูลจากเครื่องที่ใช้บริการ ในการทำงานเครื่องที่ให้บริการ จะส่งจำนวนไบต์ ของข้อมูลไปยังโปรแกรมที่ทำงานด้วย ซีจีไอ อย่างน้อยเท่ากับจำนวนใน คอนเทนเลนก์ (Content_length) โดยใช้ มาตรฐานการรับข้อมูล (Standard Input, STDIN) เป็นตัวผ่านข้อมูลให้กับ ซีจีไอ ซึ่งก็อ่านค่าที่เท่ากับ จำนวนในคอนเทนเลนก์ เช่นกัน

จากนั้นเครื่องที่ให้บริการ จะนำค่าข้อมูลที่กำหนดโดย คอนเทนไทพ์ ให้ผ่านไปยัง โปรแกรมที่ทำงานด้วย ซีจีไอ เมื่อข้อมูลถูกต้อง (ซีจีไอ ยอมรับทำงาน) จะใช้ข้อมูลนี้ในการแปล ความหมายในส่วนท้ายของข้อมูล โดยไม่มีการส่ง ค่าที่บอกการจบแฟ้ม (End Of File, EOF) ไปให้ด้วย เพราะผู้ให้บริการ และซีจีไอ จะรู้ว่าหมดข้อมูล จากค่าใน คอนเทนเลนก์ แล้ว ตัวอย่างเช่น

ถ้าเราใช้ฟอร์ม (FORM) ของ เอชทีเอ็มแอล ในการทำคิวรี่ ด้วยวิธีการแบบ พอร์ต โดยให้มีการเข้ารหัสทุกๆ 15 ไบต์ จะต้องทำการกำหนดค่าใน คอนเทนเลนก์ เท่ากับ 15 และถ้าให้มี คอนเทนไทพ์ เท่ากับ Application/x-www-form-urlencoded โดยที่ไบต์แรกของ สคริปต์ มาตรฐานการรับข้อมูล

(STDIN Script) จะเป็นตัวบอกค่าเริ่มต้น จนถึงอักขระก่อนตัวสุดท้าย คือตัวที่ 14 จะเป็นการบอกถึงการสิ้นสุดข้อมูลที่ส่งมาขอข้อความนี้ เช่น

ถ้าส่งค่า one=bob&two=cat โปรแกรมจะรับค่าเริ่มต้นไปเรื่อยๆ จนตัวอักษร " a " โปรแกรมจะรู้ว่าสิ้นสุดข้อความแล้ว เพราะเป็นตัวอักษรที่ 14 (เริ่มจาก " 0 ") และตัวต่อไป ต้องเป็นตัวสุดท้ายแน่นอน

คาต้า เอาพุท เมื่อโปรแกรมที่ทำงานด้วย ซีจีไอ ส่งข้อมูลกลับมาให้เครื่องที่ให้บริการ จะส่งเป็นมาตรฐานการแสดงผล (Standard Output , STDOUT) และที่เครื่องที่ใช้บริการจะสร้างเป็นเอกสารเอชทีเอ็มแอล จากโปรแกรมที่ทำงานของ ซีจีไอ นั้น หรืออาจเป็นข้อมูลที่ เรียกย้อนคืน คำสั่งของเครื่องที่ให้บริการก็ได้

เอาพุท (Output) จะสามารถรับค่ากลับมาได้ 2 แบบ คือ

1. ไม่ทำการแยกหัวของเอาพุท (Head Output) โดยการใช้ <FORM> ซึ่งโปรแกรมที่ทำงานด้วยจะทำ การส่งข้อมูลกลับไปอย่างสมบูรณ์ให้กับเอชทีทีพี
2. ทำการแยกหัวของเอาพุท ออกจากกัน ซึ่งเครื่องที่ให้บริการ จะทำการใช้ < FORM > นี้ ซึ่งข้อมูลที่ส่งกลับโดย ซีจีไอ จะถูกแยกเป็น 2 ส่วน คือ
   - ส่วนหัว (Header)
   - ส่วนตัวข้อมูล (Body)

โดยใช้การเว้นบรรทัดเป็นตัวแยก และที่หัวจะถูกทำการแปลความหมายโดยเครื่องที่ให้บริการ ส่วนตัวข้อมูลจะเป็น การเลือกได้ว่า ที่ใช้บอก หรือกำหนด เย็นไอเอ็มอี ของ คอนเทนไทพ์ที่ ส่วนหัวด้วย

3.4 การส่งข้อมูลด้วยวิธีการ POST และ GET

การที่ผู้ใช้บริการ ได้ส่งข้อมูลมายังเครื่องที่ให้บริการ โดยใช้โปรแกรมที่ใช้ดูข้อมูลจะต้องมีการกำหนดวิธีการส่งข้อมูลมาด้วย เพื่อให้เครื่องที่ให้บริการ ทราบและดำเนินการต่อไป อย่างถูกต้องโดยมีวิธีการอยู่ 2 แบบ คือ

3.4.1 <u>วิธีการแบบโพลท์</u> (POST Method) จะเป็นการบอกเครื่องที่ให้บริการ ว่าจะส่งข้อมูลมาเป็นส่วนๆ ซึ่งใน ดิวรี่สตริง จะใช้เครื่องหมาย "&" คั่นระหว่างตัวแปร และใช้เครื่องหมาย "+" แทนการเว้นวรรค (Space) เช่น

QUERY_STRING = "star = Eastwood&cert = 15&movie = Pale+Rider"

เมื่อทำการแยกเป็นคำจะได้ว่า star = Eastwood และ cert = 15 และ movie = Pale Rider โดยที่เครื่องหมายแต่ละตัวจะถูกแปลงให้อยู่ในรูปของเลขฐาน 16 และ ถูกนำมาแยกตัวแปรเก็บใน อาร์เรย์ โดยซีจีไอสคริปท์ อีกครั้งหนึ่ง จะเป็นวิธีการที่ได้รับความนิยม เนื่องจากข้อมูลที่ส่งจะมี เป็นส่วนๆ ทำให้สะดวกกว่า การใช้วิธีการอื่น

3.4.2 <u>วิธีการแบบเกต (GET Method)</u> เป็นการรวบรวมข้อมูลที่ตำแหน่งที่ใช้เก็บเอกสาร แล้วเครื่องที่ให้บริการ จะส่งให้โปรแกรมที่ทำงานด้วย ซีจีไอ ในรูปของตัวแปรดิวรี่สตริง โดยมีเครื่อง

หมายคำถาม " ? " ดั่นไว้ให้รู้ ว่าหลังเครื่องหมาย? นี้เป็นคิวรี่ทั้งหมด โดยที่ผู้ให้บริการ จะเรียก ข้อมูลที่อ้างอิงจากตำแหน่งที่ใช้เก็บเอกสาร ที่ใช้วิธี เกต จาก คิวรี่แบบ <ISINDEX> ซึ่งเป็นการเรียกค้นหาตำแหน่งของฐานข้อมูล จาก การบอกด้วย <ISINDEX> เพื่อใช้อ้างอิงตำแหน่งที่ใช้เก็บเอกสารนั้น ส่งค่าไปให้โปรแกรมที่ทำงานด้วย ซีจีไอ ทำการค้นหาข้อมูลที่ต้องการ เช่น

<HEAD>

<ISINDEX href= http://www.hal.com/hal-bin/query>

</HEAD>

จากตัวอย่าง ซีจีไอ จะรู้ว่าตำแหน่งที่ใช้เก็บเอกสารที่กำหนดโดย < ISINDEX > โดยมีส่วนที่เป็นเอชอาร์อีเอฟ (HREF) เป็นตัวบอกที่ที่จะติดต่อไป ส่วน

http://www.hal.com./hal-bin/query?blues+brothers

ซีจีไอ จะได้รับค่าเป็น Blues กับ brothers โดยที่เครื่องหมาย " + " จะถูกแปลงเป็นเลขฐาน 16 = %2B ส่วนช่องว่างข้างหลัง จะถูกแปลงเป็นเลขฐาน 16 = %20 และที่โปรแกรมที่ใช้ดูเอกสาร จะรับค่าต่างๆ เหล่านี้ และทำการแยกเป็นคำโดยดูจากเครื่องหมายดังกล่าว ดังตัวอย่าง

http://www.hal.com/hal-bin/query?blues%2Bbrothers%20

โดยปกติถ้ามีการใช้เครื่องหมาย ? หลังตำแหน่งที่ใช้เก็บเอกสารใน เอชทีทีพี จะกำหนดค่า วิธีการเป็น เกต โดยอัตโนมัติ และค่าใน คิวรี่สตริง จะถูกกำหนดให้เป็นค่าตามหลังเครื่องหมาย ? เสมอ และมีเครื่องหมาย + เป็นตัวบอกการแยกช่องว่างไว้เป็นคำๆ เช่น

<A href= http://www.hal.com/hal-bin/test.cgi?One+Two+Three+Four> และ

<A href= http://www.hal.com/hal-bin/test.cgi?One Two Three Four>

จะได้ผลลัพธ์ที่คิวรี่สตริงเป็น <QUERY_STRING = One+Two+Three+Four> เช่นกันทั้งสองกรณีข้างบน จากนั้นจะใช้ภาษาสคริปต์ หรือภาษาที่มีมาตรฐานการรับแสดงผล เขียนเป็นโปรแกรมที่ทำงานด้วยซีจีไอ เพื่อนำค่าที่ได้รับจากโปรแกรมที่ใช้ดู เอกสาร มาทำการรวบรวมเป็นความหมายต่อไป

3.4.3 ตัวอย่างการใช้งาน วิธีการแบบเกต

<FORM Method = GET Action = http://www.hal.com/hal-bin/test-cgi>

Enter Text : <INPUT Name = "Widget">

<INPUT TYPE = "Submit" VALUE = "Submit Query">

</FORM>

ซึ่งที่ /hal-bin/test-cgi จะทำให้ที่คิวรี่สตริง มีค่าต่างๆดังนี้

QUERY_STRING = Widget = I%27d%20rather%20be%20Sailing.&

Widget1 = Gone%20fishing%27

โดยที่ %27 = Apostrophe (*)

%20 = Space (ช่องว่าง) เมื่อแทนค่าโดยใช้โปรแกรมที่ทำงานด้วย ซีจีไอ จะได้ผลลัพธ์ คือ

Widget1 = Gone fishing'

Widget2 = I'd rather be sailing.

ซึ่งฟูลสตอป (Full Stop) ที่หลังคำว่า แซลลิ่ง (Sailing) จะไม่ถูกแปลงค่าไปด้วย เนื่องจากไม่มีรหัสเป็นตัวอักษร ที่สามารแปลค่าได้

3.5 การเลือกรูปแบบในการที่จะเขียนสคริปต์

การเขียนสคริปต์ ขึ้นมาด้วยภาษาใดนั้น จำเป็นที่จะต้องศึกษาถึง ระบบปฏิบัติการที่ใช้งาน อยู่ โดยที่โปรแกรมที่ทำงานด้วย ซีจีไอ มีงานที่ทำงานได้ง่ายในเครื่องที่มีระบบปฏิบัติการหนึ่ง แต่จะทำได้ยากกับเครื่องที่ปฏิบัติการอีกอย่างหนึ่ง ดังนั้นการเขียน การทำงานของซีจีไอ ควรที่จะเลือกรูปแบบที่เหมาะสม ให้เข้ากับชนิดของงานที่เราจะทำ ดังต่อไปนี้

3.5.1 งานเกี่ยวกับข้อความและการค้นหา (TEXT and SEARCHING)

ควรเขียนโปรแกรมสคริปต์ บนยูนิกซ์ เนื่องจากยูนิกซ์ เป็นระบบที่มีคำสั่ง ในภาษาขั้นสูง ที่จะทำงานเกี่ยวกับการค้นหา การเรียงข้อมูล และอื่นๆที่เกี่ยวกับข้อมูลที่เป็น ตัวอักษรทำให้เกิดความสะดวกในการเขียนเป็นสคริปต์ขึ้นมา ส่วนในระบบ ดอส ก็สามารถทำได้เช่นกัน แต่มีข้อจำกัดอยู่มาก ส่วน ระบบวินโดว์ จะต้องใช้ ดอส เป็นตัวช่วย (BATCH) ไว้ เพราะไม่สามารถทำงานในแบบ แปลการทำงานเป็นคำสั่ง (Command Line) ได้ ทำให้ต้องเสียเวลาในการทำงานที่ดอสก่อน จึงไม่เหมาะที่จะใช้ทำงานประเภทนี้

3.5.2 งานที่ไม่เกี่ยวกับข้อความ (NON TEXT DATA)

ได้แก่งานข้อมูลที่ได้จากการใช้โปรแกรมประมวลผลคำ (Word Processing) เช่น ตารางต่างๆ หรือเอกสารการเปรียบเทียบ ควรเขียนโปรแกรมสคริปต์ ด้วย วินโดว์ เนื่องจากระบบวินโดว์สามารถที่จะ แลกเปลี่ยนข้อมูลระหว่างโปรแกรมประยุกต์ด้วยกันได้ดี ทำให้ง่ายและสะดวกกว่า ดอส และยูนิกซ์

3.5.3 งานที่เกี่ยวกับการประมวลผลฐานข้อมูล (DATABASE ACCESS)

จะขึ้นกับชนิดของโปรแกรมที่ใช้เป็นฐานข้อมูลนั้นๆ ว่าทำงานอยู่ภายใต้รูปแบบการทำงานเช่นไร ถ้าใช้บนยูนิกซ์ เช่น โปรแกรม ออราเคิล (Oracle) หรือ ซายส์เบส (Sybase) ก็จะต้องเขียนโปรแกรม ซีจีไอ บนยูนิกส์ ถ้าใช้เป็น ระบบวินโดว์ เช่น ใช้ ไมโครซอฟท์แอกเซส ก็เขียนโปรแกรมการทำงานของซีจีไอ ภายใต้ระบบวินโดว์

3.6 หลักทั่วไปในการเขียน ซีจีไอ

การทำงานของซีจีไอคือการรับข้อมูลจากผู้ใช้บริการส่งมาให้เครื่องที่ให้บริการจากนั้นโปรแกรมการทำงานซีจีไอ ที่อยู่ในเครื่องที่ให้บริการ จะทำงานโดยติดต่อกับแหล่งข้อมูล ที่ผู้ใช้บริการต้องการ เมื่อได้เป็นผลลัพธ์ หรือข้อมูลที่ต้องการ ก็จะส่งกลับมาให้เครื่องที่ให้บริการ ทำการแสดงผลให้ผู้ใช้บริการ ทราบได้ ซึ่งมีหลักในการเขียนซีจีไอ เป็นสคริปต์ ดังนี้

3.6.1 การรับข้อมูล (Data Receiving)

ทำได้หลายแบบขึ้นกับรูปแบบที่ต้องการใช้ และวิธีการที่ผู้ให้บริการเว็บ สามารถที่จะยอมรับให้ทำงานได้ ดังนี้

CGI	GET	POST
UNIX	Environment Variable	STDIN
DOS	Environment Variable	Content File
WINDOWS	CGI Data File	Content File

ตารางที่ 3.1 ตารางเปรียบเทียบประเภท ซีจีไอ กับการใช้งานต่างๆ

จะเห็นว่า วินโดว์ มีการใช้แฟ้มข้อมูลใหม่ทั้งแบบเกต และโพสท์ ส่งผ่านข้อมูลและตัวแปรมายัง โปรแกรมของ ซีจีไอ เพราะวินโดว์ไม่สามารถใช้ตัวแปรแวดล้อม แบบยูนิกซ์ได้

3.6.2 การแยกข้อมูล (Data Parsing)

เมื่อผู้ใช้บริการ ป้อนข้อมูลมายังเครื่องที่ให้บริการ เครื่องที่ให้บริการก็จะทำการแยกข้อความของผู้ใช้บริการออก เป็นส่วนๆ เพื่อใช้ในการทำงานของโปรแกรมซีจีไอ ต่อไป ในส่วนของช่องว่างจะเปลี่ยนให้เป็นเครื่องหมาย " + " และเปลี่ยนเครื่องหมายคณิตศาสตร์ รวมทั้ง อักขระพิเศษ ให้เป็นเครื่องหมายตามรหัสแอสกี เมื่อโปรแกรมของซีจีไอ ทำงานก็จะแปลงค่ากลับ ให้เป็นข้อมูลเดิม เพื่อทำงานต่อไป

3.6.3 ส่งผลลัพธ์กลับไปยังผู้ใช้บริการ (Result Returning)

โปรแกรมของซีจีไอ สามารถที่จะส่งข้อมูลกลับ มาให้เครื่องที่ให้บริการ ได้ 3 แบบ

3.6.3.1 ส่งกลับในรูปเอกสารเอชทีเอ็มแอล (HTML Document return) โปรแกรมซีจีไอ จะต้องสร้างเอกสารเอชทีเอ็มแอลขึ้นมาแล้วส่งกลับไปซึ่งจะต้องมีส่วนหัวของเอกสารเพื่อกำหนดการทำงานไว้ด้วย เช่น

```
Content_Type : Text/html
<TITLE> HEADER </TITLE>
<HI> RETURNING </HI>
This is Feedback Result. It returning by HTML
```

3.6.3.2 ส่งข้อมูลกลับในลักษณะอื่นๆ อาจเป็นในรูปของ เสียง ภาพ โปรแกรมต่างๆ โดยจะต้องแจ้งไว้ในส่วนของหัวเอกสารที่ส่งไปให้ผู้ใช้บริการรู้ถึง ชนิดของข้อมูลที่ส่งมาด้วย

```
Content_Type : Image/gif
Content_Type : Sound/MIDI
```

3.6.3.3 ส่งกลับในลักษณะของตำแหน่งที่เก็บเอกสาร โดยจะบอกเป็นตำแหน่งที่เก็บเอกสาร ให้ผู้ที่ใช้งานทราบ เพื่อที่จะได้ใช้โปรแกรมในการดูเอกสาร ดูข้อมูลที่ต้องการนั้น เช่น

```
Location : protocol ://hostname/path_info/
```

3.7 การใช้ ฟอร์ม เพื่อรับข้อมูลจากผู้บริการ (FORM TAG)

วิธีการที่จะรับข้อมูลจากผู้ใช้บริการที่ง่ายที่สุด คือการใช้รูปแบบของ แบบฟอร์ม ในการให้ผู้ใช้งานกรอก หรือเติมข้อความที่ต้องการ โดยอาจจะมี ข้อความ, ปุ่ม, เมนู, และใช้การเลือกข้อความแบบเชคบอกซ์ (Checkbox) เพื่อรับข้อมูลแล้วส่งไปให้ เครื่องที่ให้บริการ ทำงานส่งต่อไปให้ โปรแกรมทีซีจีไอ ทำงานที่ต้องการต่อไป

ส่วนประกอบของฟอร์ม มี 3 ส่วน คือ

- ส่วนเริ่มต้น (Header)
- ส่วนแสดงชื่อ และตำแหน่งที่ใช้งาน (Names Input Files)
- ส่วนที่เป็นการทำงานในการเลือกข้อมูล

3.7.1 ส่วนเริ่มต้น

FORM เป็นตัวบอกเริ่มต้นว่าในเอกสารนั้นจะมีการใช้เป็นข้อมูลที่ส่งกลับไปให้เครื่องที่ให้บริการต่อไป ต้องมีส่วนปิดท้ายบอกการหยุดรับข้อมูลนี้ด้วย </FORM>

3.7.2 ส่วนแสดงชื่อ และตำแหน่งที่ใช้งาน

ACTION เป็นตัวบอกตำแหน่งที่จะมีการทำงานด้วย ซีจีไอในการที่นำข้อมูลนี้ไปปฏิบัติ ต้องเขียนร่วมในส่วน ฟอร์ม เช่น

<FORM ACTION = “ตำแหน่งที่เก็บโปรแกรม ซีจีไอ”>

METHOD เป็นวิธีการที่ใช้ในการติดต่อส่งข้อมูลกัน ระหว่างเครื่องที่ให้บริการ กับโปรแกรมซีจีไอ ควรเลือกวิธีการนี้ให้ถูกต้องกับเครื่องที่ให้บริการ โดยทั่วไปจะมี 2 แบบ คือ โพสท์ , เกต

- โพสท์ เป็นการส่งข้อมูลจากแหล่งข้อมูลมายังโปรแกรมที่ใช้ดูข้อมูล
- เกต เป็นการรับข้อมูลจากผู้ใช้บริการมายังให้บริการ และโปรแกรมซีจีไอ

3.7.3 ส่วนที่เป็นการทำงานในการเลือกข้อมูล

INPUT เป็นการสั่งให้มีการรับข้อมูลจากผู้ใช้งานโดยมีรูปแบบย่อยๆให้เลือกใช้งานเพื่อให้น่าสนใจ และสะดวกในการป้อนข้อมูล

TEXT จะเป็นการกำหนดให้ข้อมูลที่ป้อนเป็นข้อความขนาดสั้นๆ และมีการกำหนดชื่อให้กับข้อความนั้นด้วย

<INPUT TYPE= “TEXT” NAME= “NAME”>

ยังสามารถที่จะกำหนดขนาดความกว้างของช่องที่ป้อนข้อมูลได้ด้วยคำสั่ง SIZE เช่น

<INPUT TYPE= “TEXT” SIZE=25 NAME= “Name”>

CHECKBOX เป็นการเลือกหัวข้อที่กำหนดไว้ให้โดยผู้ใช้บริการ สามารถ ที่จะเลือก โดยกำหนดค่าในช่องนั้นๆได้ ทำเป็นรูปสี่เหลี่ยม สามารถที่จะเลือกแล้วเกิดเป็นเครื่องหมาย (Mark) แจ้งให้ทราบ

RADIO จะมีการทำงาน และใช้งานที่เหมือนกับ Checkbox แต่รูปร่างจะเป็นวงกลม เมื่อเลือกหัวข้อแล้ว แสดงเป็นจุดดำกลางวงไว้ให้ทราบ

SUBMIT เป็นการสั่งให้ทำงานโดยจะส่งข้อมูลที่ป้อนไว้แล้วไปยังเครื่องที่ให้บริการแล้วให้โปรแกรมซีจีไอ ใช้งานต่อไป จะเกิดเป็นกล่องสี่เหลี่ยม ให้เลือก และมีข้อความตามที่กำหนดได้ด้วย

RESET เป็นการยกเลิกการทำงาน หรือยกเลิกข้อมูลที่ป้อนไว้ ก่อนหน้านี้ทั้งหมด โดยจะกลับไปเริ่มที่จุดเริ่มต้นใหม่ ข้อมูลจะยังไม่ถูกส่งไป จะเกิดเป็นกล่องสี่เหลี่ยม ให้เลือกและมีข้อความตามที่กำหนดได้ด้วยเป็นการทำงานที่ตรงข้ามกับ SUBMIT

SELECT เป็นการกำหนดให้ผู้ใช้งาน เลือกข้อมูลในย่านของหัวข้อที่กำหนดเป็นการเฉพาะ มีการใช้งานร่วมกับคำสั่ง OPTION เพื่อใช้เป็นตัวกำหนดหัวข้อให้เลือก เช่น

```
<SELECT NAME = "Do">
<OPTION> YES
<OPTION> NO
<OPTION> MAYBE
</SELECT>
```

TEXTAREA เป็นการกำหนดพื้นที่เพื่อใช้ผู้ใช้งานสามารถที่ป้อนข้อมูลขนาดมากๆได้ เพื่อใช้ในการแสดงความคิดเห็น หรือข้อเสนอแนะ สามารถที่จะกำหนดขนาดที่ให้ป้อนข้อมูลได้

```
<TEXTAREA NAME = "Comment" COLS=40 ROWS = 10>
```

บทที่ 4

ภาษาจาวา (JAVA LANGUAGE)

จาวา เป็นภาษาคอมพิวเตอร์แบบอ็อบเจ็กโอเรียนเต็ด (Object Oriented) มีความคล้ายกับภาษาซี หรือซีพลัสพลัส (C/C++) มาก แต่ก็ไม่เหมือนเลยทีเดียว โดยตัดข้อเสียบางอย่างของภาษาซี หรือซีพลัสพลัส ทิ้งไปแล้วเพิ่มเติมข้อดีหลาย ๆ อย่างเข้าไป โดยเฉพาะด้านการใช้งาน เกี่ยวกับอินเตอร์เน็ต แต่ยังคงความง่ายในการเขียนโปรแกรมแบบภาษาซี หรือซีพลัสพลัส

4.1.1 จาวาสามารถเขียนโปรแกรมได้ 4 แบบ คือ

4.1.1.1 จาวาแอปพลิเคชั่น (Java Applications) เป็นแอปพลิเคชั่นที่สามารถทำงานได้ด้วยตัวเอง เหมือนกับ โปรแกรมที่เขียนด้วย ภาษาซีหรือซีพลัสพลัส แต่แอปพลิเคชั่นของจาวายังต้องอาศัยจาวา อินเตอร์เพรสเตอร์ (Java Interpreter) ช่วยด้วย

4.1.1.2 จาวาแอพเพลท (Java Applets) เป็นแอปพลิเคชั่นขนาดเล็กที่ฝังตัวอยู่ในเว็บเพจ เป็นโปรแกรมที่มีระบบรักษาความปลอดภัยมากกว่าแอปพลิเคชั่นปกติ เนื่องจากมีข้อจำกัดหลายอย่าง ใน แอพเพลท เช่นแอพเพลทไม่สามารถทำไดนามิกลิงค์กับภาษาซีหรือซีพลัสพลัส ได้แอพเพลทจะปฏิบัติงานร่วมกับไฟล์เอชทีเอ็มแอล (HTML) อาศัยแอพเพลทวิวเวอร์ (Applets Viewer) หรือ เว็บบราวเซอร์ เช่น เน็ทสแคปเนวิเกเตอร์ (Netscape Navigator) ไมโครซอฟท์อินเตอร์เน็ทเอ็กโพร์เลอร์ (Microsoft Internet Explorer) ในการดูผลการปฏิบัติงาน

4.1.1.3 คอนเท็นท์แฮนด์เลอร์ (Content Handlers) เป็นโปรแกรมของจาวาที่เขียนด้วยมีจุดประสงค์พิเศษ คือเขียนเพื่อทำให้จาวาเข้าใจข้อมูลชนิดใหม่ เช่น ภาพยนต์ควิ๊กทาม (Quick Time) , เสียง (Voice) , โฟโต้ซีดี (PhotoCD)

4.1.1.4 โปรโตคอลแฮนด์เลอร์ (Protocol Handlers) เป็นโปรแกรมที่มีคุณสมบัติคล้ายกับคอนเท็นท์แฮนด์เลอร์ ต่างกันที่เขียนเพื่อทำให้เว็บบราวเซอร์ เข้าใจโปรโตคอลใหม่ ๆ เช่น มิดี้ (MIDI)

4.1.2 ข้อดีของภาษาวาจา

4.1.2.1 ความง่าย (Simple) : จาวาถูกออกแบบมาให้ใช้งานง่าย ไม่ต้องการเรียนรู้มากนัก ถึงแม้ว่าการเรียนรู้ภาษาซีหรือซีพลัสพลัส นั้นยุ่งยาก แต่จาวาได้ตัดข้อยุ่งยากหลายอย่างในภาษาซีหรือซีพลัสพลัส ซึ่งไม่ค่อยได้ใช้ ทิ้งไป เช่น โอเวอร์โหลดดิ้ง (OverLoading) , มัลติเปิ้ลอินเฮอริแทน (Multiple Inheritance) แล้วยังเพิ่มความสามารถบางอย่างเข้าไป เช่น ออโต้การ์แบกคอลเลกชั่น (Auto Garbage Collection)

4.1.2.2 อ๊อบเจ็กโอเรียนเต็ด : ซึ่งการเขียนโปรแกรมแบบนี้กำลังเป็นที่นิยมในปัจจุบัน เนื่องจากสามารถนำโค้ดเก่ามาใช้ได้ และปรับปรุงแก้ไข เพิ่มเติมได้ โดยไม่มีผลกับการทำงานของโปรแกรมเดิม และจาวาก็เป็นภาษาแบบอ๊อบเจ็กโอเรียนเต็ด สมบูรณ์ยิ่งกว่าภาษาซีพลัสพลัส

4.1.2.3 การกระจาย (Distributed) : จาวาถูกออกแบบมาให้ใช้ในเน็ตเวิร์ค ดังนั้นจึงมีชุดคำสั่งหรือเมทธอดต่าง ๆ ที่ใช้จัดการในเรื่อง โปรโตคอล เช่น เอชทีทีพี , เอฟทีพี ดังนั้นแอปพลิเคชั่นที่เขียนด้วยภาษาจาวา จึงสามารถเปิดและเรียกใช้ออบเจ็กต์ต่าง ๆ ที่อยู่บนอินเตอร์เน็ตได้โดยผ่านทาง ยูอาร์แอล (URLs)

4.1.2.4 ความแข็งแกร่ง (Robust) : จาวานั้นผ่านการตรวจสอบอย่างดี ในขณะแปล และขณะปฏิบัติงาน เพื่อให้แน่ใจว่าจะไม่สร้างปัญหาภายหลังเมื่อเวลาใช้งานจริง ที่สำคัญจาวาไม่มีข้อมูลชนิดพอยน์เตอร์ เพื่อป้องกันการเข้าถึงหน่วยความจำโดยตรง ซึ่งอาจก่อผลเสียให้กับระบบได้

4.1.2.5 ความปลอดภัย (Secure) : จาวาเป็นเทคโนโลยีแบบไคลเอนต์ เมื่อตัวโปรแกรมแปลให้เป็นไบต์โค้ด แล้วตัวโปรแกรมจะถูกดาวน์โหลดจากโฮสต์ด้วยเทคโนโลยีแบบเว็บเซอร์ฟเวอร์ และทำงานต่อบนไคลเอนต์ เนื่องจากขณะเรากำลังดาวน์โหลดจากเครื่องที่อยู่ระยะไกล ให้เข้ามายังเครื่องเรา ดังนั้นจึงจำเป็นจะต้องระมัดระวังเรื่องไวรัสที่มีกระจายอยู่ทั่ว ๆ ไป เรื่องนี้เราอาจจะไม่เชื่อ แต่บริษัทซัน ก็ได้ยืนยันว่าปลอดภัย เพราะรันไทม์ซิสเต็มที่ทำงานอยู่บนเครื่องเราจะสามารถทำการตรวจสอบ ไบต์โค้ดที่ได้รับมาว่าปลอดภัยหรือไม่ ถ้าไม่ปลอดภัยก็จะไม่ยอมให้โปรแกรมทำงานต่อ นอกจากนี้การเปลี่ยนแปลงโครงสร้างของพอยน์เตอร์ใหม่นั้น ก็มีส่วนทำให้ปลอดภัยมากขึ้น

4.1.2.6 ความเป็นกลางทางโครงสร้าง (Architecture Neutral) : คอมพายเลอร์ของจาวาจะสร้างโค้ดคำสั่งที่มีขนาดหนึ่งไบต์ (Byte Code) ซึ่งไม่ขึ้นกับโครงสร้างฮาร์ดแวร์ โค้ดคำสั่งที่สร้างขึ้นนี้จะถูกแปลบนเครื่องใด ๆ ให้กลายเป็นภาษาเครื่องระหว่างการทำงานอย่างรวดเร็ว ดังนั้นจาวาจึงทำงานได้หลายเพลตฟอร์ม

4.1.2.7 พอร์ตเทเบิ้ล (Portable) : จาวาได้กำหนดชนิดข้อมูลให้เป็นมาตรฐาน เช่น ข้อมูลชนิด Int จะเป็นแบบมีเครื่องหมายชนิด 32 บิต Float ก็เป็นเลขจำนวนจริงขนาด 32 บิตแบบ IEEE 754 Character ก็ใช้แบบ Unicode ขนาด 16 บิต ซึ่งมาตรฐานเหล่านี้ใช้กันใน CPU หลาย ๆ แบบ

4.1.2.8 อินเตอร์เพลทเต็ด (Interpreted) : โค้ดคำสั่งขนาด 1 ไบต์ที่ได้จากคอมพายเลอร์ของจาวานั้น ไม่ขึ้นกับฟอร์แมตของเครื่องใด ๆ จะต้องผ่านจาวาอินเตอร์เพลทเตอร์ โดยตรงบนเครื่อง

4.1.2.9 ประสิทธิภาพการทำงานสูง (High Performance) :ประสิทธิภาพของการแปลไบต์โค้ดที่ได้ค่อนข้างสูง ไบต์โค้ดจะถูกแปลเป็นแมชชีนโค้ดขณะโปรแกรมทำงานตามชนิดของซีพียู ฟอร์แมตของไบต์โค้ดได้รับการออกแบบให้มีประสิทธิภาพมาก สามารถจองรีจิสเตอร์มาใช้งานได้

4.1.2.10 มัลติเธด (multithreaded) : เป็นวิธีสร้างแอปพลิเคชั่นที่สามารถทำงานในหลาย ๆ งานได้ในขณะเดียวกัน ซึ่งจะทำให้โปรแกรมตอบสนองในลักษณะอินเตอร์แอคทีฟได้ดี

4.1.2.11 พลวัตร (Dynamic) : ตัวอย่างปัญหาที่พบได้เมื่อใช้ ซีพลัสพลัส เขียนโปรแกรม เช่น บริษัท ก . ไก่ ซื้อคลาสไลบรารีของบริษัท ข. ไข่ ไปใช้งานในการสร้างซอฟแวร์ ต่อมาหากบริษัท ข.ไข่ ปรับปรุงไลบรารี่ของตัวเอง และออกเวอร์ชันใหม่มา บริษัท ก.ไก่ ก็ต้องคอมพายล์ซอฟต์แวร์ของตัวเอง

ใหม่ และส่งให้ผู้ซื้อทุกราย จาวาแก้ปัญหานี้ด้วยการโปรแกรมแบบออบเจ็กต์โอเรียนเต็ด อย่างแท้จริง ทำให้การเชื่อมโยงระหว่างโมดูลทำได้อย่างแท้จริง

4.1.3 ชนิดข้อมูล (Data Type)

2.5.3.1 เลขจำนวนเต็ม (Integer) เป็นแบบมีเครื่องหมายทั้งหมด ประกอบด้วย

8 - bit **byte** 16 - bit **short** 32 - bit **int** 64 - bit **long**	ตัวอย่าง	**byte** frame; **short** count=0; **int** a , b , c = 10 ; **long** result = 0 ;

2.5.3.2 Real ตามมาตรฐาน IEEE-754 ประกอบด้วย

32 - bit **float** 64 - bit **double**	ตัวอย่าง	**float** total = 0.0 ; **double** fact ;

4.1.3.3 Character เป็นข้อมูลชนิดตัวอักษร ใช้ขนาด 16-Bit แบบ Unicode Character

4.1.3.4 Boolean ใช้ขนาด 1-Bit กับข้อมูล true หรือ false เขียนด้วยตัวเล็กเท่านั้น

4.1.3.5 Array เป็นแบบออบเจ็กต์ สร้างขึ้นในขณะปฏิบัติงานโปรแกรม และมีการป้องกันการใช้ Array เกินขนาด

การประกาศตัวแปร Array เช่น
```
int     grade[ ] ;
char    name[ ] ;
```
จากนั้นก็กำหนดหน่วยความจำให้
```
grade = new int [10];
name  = new char[10][10];
```
ขั้นสุดท้ายก็สามารถให้ค่าได้
```
grade[2] = 60 ;
name[0][2] = "A" ;
```

4.1.3.6 String เป็นชนิดออบเจ็กต์ มี 2 แบบคือ

String สำหรับอ่านได้อย่างเดียว

StringBuffer แก้ไขได้ เปลี่ยนแปลงขนาดได้

ตัวอย่าง

```
String name = " ChaiYa  ChaiYaDum";
StringBuffer  strbff ;
        Strbff = new String ("ChaiYa YenKhae") ;
```

สามารถเขียนโดยใช้เครื่องหมายบวกในการต่อกันของสตริงได้ ดังนี้

```
strbff = new String ("ChaiYa "+"YenKhae") ;
strbff = new String("ChaiYa "+"YenKhae"+"Chaiya"+"Chaiyadum") ;
```

4.1.4 การเขียนหมายเหตุ (Comments) สามารถเขียนได้ทั้ง 2 แบบ คือ

4.1.4.1 แบบหมายเหตุของภาษาซี /*.... ส่วนของหมายเหตุ ....*/ จะเขียนยาวกี่บรรทัดก็ได้ ขอให้อยู่ภายในเครื่องหมายนี้ เช่น

```
/*-------------------------------------------------------
-------------------------------------------------------*/
```

หรือ

```
/*-------------------------------------------------------*/
/*-------------------------------------------------------*/
/*-------------------------------------------------------*/
```

4.1.4.2 แบบหมายเหตุของภาษาซีพลัสพลัสคือเขียนหลังเครื่องหมายแบ็กสแลช // (Back Slash) หนึ่งบรรทัดต่อหนึ่ง หมายเหตุ เช่น

```
// ---------- หมายเหตุ 1 ----------
// ---------- หมายเหตุ 2 ----------
```

4.1.5 คำสั่งควบคุม (Control Statements)

จาวาใช้คำสั่งควบคุมเหมือนซีหรือซีพลัสพลัสทุกประการ พร้อมทั้งรูปแบบการเขียนก็เหมือนกัน แต่ภาษาจาวาไม่มีคำสั่ง Goto

4.1.5.1 if เป็นคำสั่งให้ไปทำงานยังคำสั่งใดคำสั่งหนึ่ง โดยมีเงื่อนไขให้ตัดสินใจอย่างใดอย่างหนึ่งก่อน แล้วจึงไปทำงาน โดยผลของการตัดสินใจจะมีโอกาสเป็นไปได้ 2 ทาง คือ

4.1.5.1.1 จริง หมายความว่ามีค่ามากกว่าศูนย์

4.1.5.1.2 ไม่จริง หมายความว่ามีค่าน้อยกว่า หรือเท่ากับศูนย์

4.1.5.2 switch เป็นคำสั่งให้ไปทำคำสั่งใดคำสั่งหนึ่งตามที่ต้องการ โดยมีทางเลือกให้ไปทำคำสั่งได้หลาย ๆ ทางเลือก และจะไปทำคำสั่งใดนั้น จะขึ้นอยู่กับค่าของตัวแปรที่ทำหน้าที่ควบคุมคำสั่ง switch นั้น

4.1.5.3 while เป็นคำสั่งเมื่อต้องการให้คอมพิวเตอร์ทำงานเป็นลูป (LOOP)

4.1.5.4 do while เป็นคำสั่งเมื่อต้องการให้คอมพิวเตอร์ทำงานเป็นลูปเช่นเดียวกับคำสั่ง while แต่คำสั่ง do while จะทำคำสั่งก่อน 1. ครั้งแล้วจึงตรวจสอบเงื่อนไข

4.1.5.5 for เป็นคำสั่งให้มีการทำงานซ้ำ โดยทราบจำนวนครั้งการทำงานซ้ำแน่นอน

4.1.5.6 break เป็นคำสั่งให้หลุดจากการทำงานซ้ำ เพื่อให้ไปทำคำสั่งที่อยู่ถัดจากคำสั่งการทำงานซ้ำ นั่นคือคำสั่ง switch , for , while , do while

4.1.6 คำหลักสงวน (Reserved Keyword)

abstract	Else	int	static
Boolean	Extends	interface	super
Break	False	long	switch
Byte	Final	native	synchronized
Byvalue*	Finally	new	this
Case	Float	null	threadsafe
cast*	for	operator*	throw
Catch	future*	outer*	transient
Char	generic*	package	true
Class	goto*	private	try
conts*	if	protected	var*
Continue	inner*	public	void
Default	implement	rest*	while
Do	import	return	
Double	instanceof	short	

* เป็นคำหลักสงวน แต่จาวา เลิกใช้แล้ว

4.1.7 เครื่องที่สามารถทำงานโดยภาษาจาวาได้

จาวาใช้ได้กับเครื่องที่ใช้ระบบปฏิบัติการดังต่อไปนี้ Windows95, Windows-NT, Solaris2.3 ถึง 2.5 บน Sparc , MacOS 7.5 บน PowerMacs , Mac68020, 68030, 68040

กำลังพัฒนาให้ใช้ได้กับแพลทฟอร์มอื่น ๆ เช่น Linux , OS/2, MIPS, Alpha หรือ PowerPC บน NT, Windows 3.1, Solaris บน X86, Amiga เป็นต้น

4.1.8 อักขระที่ใช้ในภาษาจาวา

ตัวอักษรตัวใหญ่และตัวเล็ก เช่น "A" กับ "a" ภาษา Java จะมองเป็นคนละตัวกันการตั้งชื่อสำหรับตัวแปร คลาส และ เม็ทธอด จะต้องขึ้นต้นด้วยตัวอักษร "a" –"z", "A" – "Z" , หรือเครื่องหมายขีดล่าง " _ " (UnderScore) หรือเครื่องหมายดอลล่าร์ " $ " (Dollar Sign) เท่านั้น ตัวอักษรต่อ ๆ ไปสามารถเป็น "0" – "9" หรือตัวอักษรพิเศษอื่น ๆ ก็ได้ ที่สำคัญการตั้งชื่อคลาส ต้องขึ้นต้นด้วยตัวอักษรตัวใหญ่เท่านั้น การตั้งชื่อตัวแปร และเม็ทธอด ควรขึ้นต้นด้วยตัวอักษรตัวเล็ก

4.1.9 เครื่องหมายดำเนินการ (Operator)

4.1.9.1 เครื่องหมายดำเนินการทางคณิตศาสตร์ (Arithmetic Operator)

เครื่องหมาย	คำอธิบาย	ตัวอย่าง
-	เปลี่ยนจำนวนบวกเป็นลบ หรือลบเป็นบวก	-i;

	เปลี่ยน 0 เป็น 1 และ 1 เป็น 0	done ;
++	เพิ่มขึ้นทีละหนึ่ง	++y ; เพิ่มค่าให้กับตัวแปร y b++ ; เพิ่มค่าให้กับตัวแปร b
--	ลดลงทีละหนึ่ง	--y ; ลดค่าตัวแปร y b--; ลดค่าตัวแปร b
+	บวก	y+4 ;
-	ลบ	a – b ;
*	คูณ	d * 8 ;
/	หาร	h / i ;
%	หารเอาเฉพาะเศษ (Modulus)	n % 2 ;

ตารางที่ 4.1 Arithmetic Operator

4.1.9.2 เครื่องหมายดำเนินการเชิงสัมพันธ์ (Relational And Logical Operators)

เครื่องหมาย	คำอธิบาย	ตัวอย่าง
<	น้อยกว่า	if (a<b) statement_1;
>	มากกว่า	if (a>b) statement_1;
<=	น้อยกว่าหรือเท่ากับ	if (a<=b) statement_1;
>=	มากกว่าหรือเท่ากับ	if (a>=b) statement_1;
==	เท่ากับ	if (a==b) statement_1;
!=	ไม่เท่ากับ	if (a!=b) statement_1;
!	เปลี่ยนจากเท็จเป็นจริงหรือจากจริงเป็นเท็จ	if (!b) statement_1;
&&	AND	if((a>b)&&(b<c)) statement_1;
||	OR	if((a>0) || (b>0)) statement_1;
^^	XOR	

ตารางที่ 4.2 Relational And Logical Operators

4.1.9.3 เครื่องหมายดำเนินการทางลอจิก (Logic Operators)

เครื่องหมาย	คำอธิบาย	ตัวอย่าง
&	AND	d & 0x0ff ;
|	OR	d | 127 ;
^	XOR	d ^ 00 ;
<<	Rotate Left	<<f; หรือ f<<;
>>	Rotate Right	>>t; หรือ t>>;

>>>	Shift Right	>>>t; หรือ t>>;

ตารางที่ 4.3 Logic Operators

4.1.9.4 ลำดับความสำคัญของตัวดำเนินการ (Priority of Operator)

	[]	()		
++	--	!		instanceof
*	/	%		
+	-			
<	>	<=	>=	
==	!=			
&				
^				
&&				
||				
?:				
=				
,	op=			

ตารางที่ 4.4 Priority of Operator

4.1.10 การแอสซายน์เม็นท์ (Assignment) และ นิพจน์ (Expression)

4.1.10.1 การแอสซายน์เม็นท์ คือการนำค่าตัวแปร หรือค่าคงที่ หรือนิพจน์ หรือค่าของฟังก์ชัน ทางด้านขวาของเครื่องหมายเท่ากับ มาให้กับตัวแปรทางด้านซ้าย เช่น

c = 5 ;

a = b+c ;

ในกรณีเพิ่มค่า เครื่องหมาย ++ หรือ - - อยู่ก่อนตัว หรืออยู่หลังตัวแปรมีความหมายต่างกัน เช่น

a = ++b : หมายความว่า เพิ่มค่าให้กับตัวแปร b ก่อน แล้วจึงนำมาให้กับตัวแปร a

a = b++ : หมายความว่า นำค่ามาให้กับตัวแปร a ก่อนแล้วจึงเพิ่มค่าให้กับตัวแปร b

ถ้าการแอสซายน์เม็นท์ เป็นแบบอ้างถึงตัวแปรเดิม สามารถเขียนแบบย่อได้ เช่น

a = a-10 : สามารถเขียนแบบย่อได้เป็น a+= 10 ;

d = d&b : สามารถเขียนแบบย่อได้เป็น d &= b ;

a = a*10 : สามารถเขียนแบบย่อได้เป็น a*= 10 ;

คำสั่ง การแอสซายน์เม็นท์ พิเศษ เช่น

a (a > 0) ? (a —) : 10 ; มีค่าเท่ากับ if (a > 0) a- - ; else a = 10 ;

จาวาสามารถประกาศตัวแปรเฉพาะบล๊อกได้ เช่น

```
for ( int m = 0 ; m < 10 ; m++ ) {
        ;ใช้ตัวแปร m ในโปรแกรม ได้เฉพาะบล็อกนี้เท่านั้น
                }
```

4.1.10.2 นิพจน์ (Expression)

4.1.10.2.1 นิพจน์ทางคณิตศาสตร์ ผลที่ได้เป็นตัวเลข เช่น

```
( a * b )+(d / n )+( m*m*m ) ;
```

4.1.10.2.2 นิพจน์ทางการเปรียบเทียบ ผลที่ได้เป็นค่าจริง (true) หรือ เท็จ (false) เท่านั้น

```
[((a >= b)&&(a <=c)) || ((n! = m )&&(!k))] ;
```

4.1.10.2.3 นิพจน์ทางลอจิก ผลที่ได้เป็นตัวเลข ซึ่งมีการเปลี่ยนแปลงระดับบิท

```
c & b ;
a || n ;
```

4.1.11 การเขียนโปรแกรมภาษาจาวาแบบแอปพลิเคชั่น (Writing Java Application)

การเขียนภาษาจาวาให้ปฏิบัติงานแบบเดี่ยว ๆ (Standard Alone) นั้น จะต้องสร้างคลาส (Class) ให้มี เมทธอดที่ชื่อ main เสมอ เพื่อเป็นจุดเริ่มต้นในการปฏิบัติงานซึ่งก็จะเหมือนกับภาษาซีหรือซีพลัสพลัส ตัวอย่างเช่น

```
class example {
    public static void main ( String args[ ] ){
        // ---- ส่วนของคำสั่งในโปรแกรม
```

ซึ่งจะแตกต่างจากภาษาจาวาแบบแอพเพลท คือ ชื่อคลาส (class name) ไม่จำเป็นต้องมีเอ็กเท็นต์ (extends) นั่นคือไม่จำเป็นต้องทำตัวเป็นคลาสย่อย (Sub Class) ของคลาสอื่น ๆ เหมือนกับภาษาจาวาแบบแอพเพลท สรุปว่าจะเขียนภาษาจาวาแบบแอปพลิเคชั่น จะต้องประกอบด้วย 4. ข้อ ต่อไปนี้

- จะต้องมี Public static
- ต้องมีเมทธอดชื่อ main หนึ่งเมทธอดเสมอ
- ต้องไม่มีการส่งค่ากลับในเมทธอด main
- จะต้องมี Command Line Argument

4.1.11.1 เหตุผลที่ต้องมี public static

เพราะว่าภาษาจาวานี้ไม่ได้ออกแบบมาให้ ปฏิบัติงานแบบโปรแกรมทั่ว ๆ ไ ป นั่นคือไม่ใช่ให้ระบบปฏิบัติการทำการ โหลดโปรแกรมเข้าสู่หน่วยความจำหลัก แล้วปฏิบัติงาน (Execute) แต่เมื่อจาวาจะเริ่มปฏิบัติงาน ก็จะทำการจัดการค่าเริ่มต้นต่าง ๆ (initialized) จัดการค่าเริ่มต้นตัวแปรแบบสถิต (static) และจัดการค่าเริ่มต้นเกี่ยวกับบล็อกเพื่อใช้ปฏิบัติงานโปรแกรม ตัวแปรแบบธรรมดาจะไม่นำมาจัดการค่าเริ่มต้นด้วย และจะไม่โหลดโปรแกรมเข้าสู่หน่วยความจำหลัก ดังนั้นเมทธอด main จะไม่สามารถเข้าถึง (Access) ฟิลด์ (Fields) ต่าง ๆ ของคลาสที่ไม่เป็นแบบสแตติกได้ (Non-static)

4.1.11.2 การจบโปรแกรมในเมทธอด main

ภาษาซีหรือซีพลัสพลัส นั้น ตัวฟังก์ชันหลักจะเขียนเป็น int main () ดังนั้นการจบโปรแกรม (Terminate) จึงทำได้ 2 วิธี คือ เรียกใช้ฟังก์ชัน exit() ; หรือ ส่งค่ากลับเป็นจำนวนเต็ม (return int) จาวาจบโปรแกรมได้กรณีเดียวคือ เรียกใช้เมทธอด System.exit(); เพราะเมทธอด main ของจาวาไม่มีการส่งค่ากลับ

4.1.11.3 คอมมานด์ลาย และอาร์กิวเม็นท์ (Command-Line Argument)

ภาษาซีหรือซีพลัสพลัสจะเขียนเป็น main (int argc, char argv[ ]) ดังนั้น argc จะเก็บจำนวน Argument รวมทั้งชื่อไฟล์ของโปรแกรมด้วย นั่นคือ argv[0] = ชื่อไฟล์ของโปรแกรม

agrv[1] = Argument ตัวที่ 1.

agrv[2] = Argument ตัวที่ 2.

และถ้ามีอาร์กิวเม็นท์ ตัวต่อ ๆ ไป ก็จะเป็นตัวที่ 3,4,5,...

จาวาเขียนเป็น main (String args[ ]) จำนวนอาร์กิวเมนท์เก็บที่อ๊อบเจ็ก args , length และเพราะจาวาใช้คำสั่ง ปฏิบัติงานเป็น

Java ชื่อไฟล์ Argument_1 Argument_2 Argument_3

ชื่อไฟล์ที่ว่านี้มีนามสกุลเป็น class

argv[0] = Argument ตัวที่ 1.

argv[1] = Argument ตัวที่ 2.

argv[2] = Argument ตัวที่ 3.

และถ้ามี Argument ตัวต่อ ๆ ไปก็จะเป็นตัวที่ 3,4,5,...

4.1.11.4 การใช้งาน JDK (Java Development Kit)

ให้พิมพ์โปรแกรมดังต่อไปนี้ที่เอดิเตอร์ (Editor) ของระบบปฏิบัติการ Windows-NT หรือ Windows 95 แล้วจัดเก็บ (Save) ชื่อไฟล์เป็น CountChar.java

```
class CountCharTest{      //--- การตั้งชื่อคลาส
        public static void main ( String args[ ] )      //---- ชื่อเมทธอด main
        throws java.io.IOException{      //---- การละเว้นการตรวจสอบความผิดพลาด
        int n=0 ;
        while ( System.in.read ( ) !='\n' ) n++;  //--- จะหลุดจากหลูปเมื่อกดคีย์ ENTER
        System.out.println( "Char Count = " +n );   //--- แสดงผล
                                } //--- end of main function
} //---- end of class
```

เมื่อพิมพ์เสร็จก็จะเป็นการคอมไพล์ โดยใช้คำสั่ง

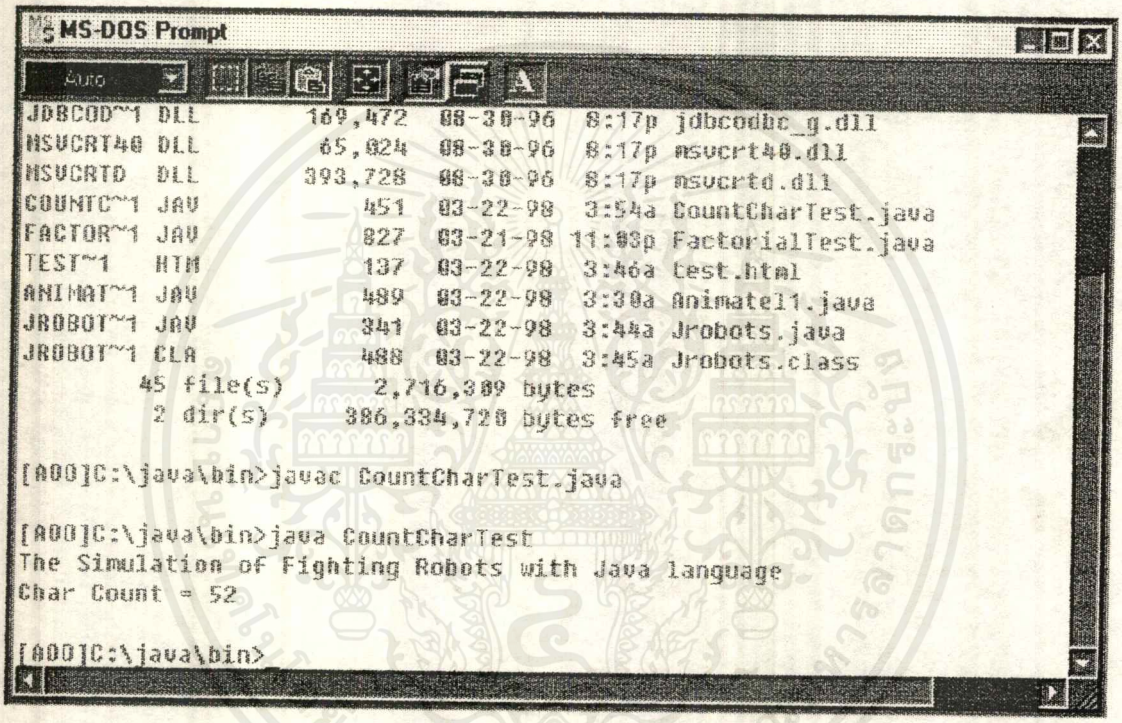
Javac CountChar.java <ENTER>

ถ้ามีข้อผิดพลาดจากการคอมไพล์ เครื่องก็จะแสดงข้อผิดพลาดนั้น พร้อมทั้งแสดงหมายเลขบรรทัดที่ผิด และเราก็ต้องกลับแก้ไข ไฟล์ต้นฉบับให้ถูกต้อง

ถ้าไม่มีข้อผิดพลาด ก็จะได้ไฟล์ตามชื่อของคลาส นั่นคือ CharCountTest.class การปฏิบัติงานให้ใช้คำสั่ง

Java CountCharTest <ENTER>

ข้อสังเกตคือ การคอมไพล์ใช้โปรแกรม Javac.exe ตามด้วยชื่อไฟล์ใส่นามสกุล . Java การปฏิบัติงานใช้โปรแกรม Java.exe ตามด้วยชื่อไฟล์ไม่ต้องใส่นามสกุล .class เท็กเอดิทเตอร์ตัวนั้น รวมทั้งระบบปฏิบัติการ จะต้องตั้งชื่อไฟล์ได้ยาวมากกว่า 8 ตัวอักษร อักษรตัวใหญ่ตัวเล็กต่างกัน นามสกุลยาวได้เกิน 3 ตัวอักษร



รูปที่ 4.1 แสดงการปฏิบัติงานของโปรแกรม CountChar.Java

4.1.11.5 การรับข้อมูลเข้า

จะใช้ตัวแปรคลาส System.in และใช้เมทธอด read(); ในการรับข้อมูลเข้า แต่ข้อมูลที่รับเข้ามาจะอยู่ใน buffer ก่อนจนกว่าเราจะป้อนข้อมูลจบนั่นคือกดคีย์ <ENTER> แล้วจึงจะนำค่าจากบัฟเฟอร์ (Buffer) มาใส่ในตัวแปรที่จะรับข้อมูลทีละตัวจนครบทุกตัว

4.1.11.6 การแสดงผลข้อมูล

จะใช้ตัวแปรคลาส System.out และใช้เมทธอด println( ); ในการแสดงผลข้อมูลจาวา สามารถเขียนโปรแกรม เกี่ยวกับโครงสร้างข้อมูลได้เช่นเดียวกับภาษาซีหรือซีพลัสพลัส ดังตัวอย่างต่อไป แสดงโปรแกรมหาค่าแฟกโทเรียล (Factorial) จาก 0 ถึงค่าที่รับเข้ามาจาก คีย์บอร์ด และเช่นเดียวกับตัวอย่างแรก เราจะคอมไพล์โดยใช้คำสั่ง

Javac Factorial Java <ENTER>

ปฏิบัติงาน โดยใช้คำสั่ง

Java FactorialTest <ENTER>

ชื่อไฟล์ Factorial . Java

```
class FactorialTest{
public static void main( String args[ ] )
throw java.io.IOException }
int k 0.input. 0;
long result;
  System.out.println( "Enter Number " ).
do;
  k   System.in.read( );
  if( k'   a ) input ( input*10 )*( k
  if(input*0) input = -input;
  if(k  'n') for(int j 0: j < input
          result fact( j );
          System.out.println("Factorial"+j+ " = " + result);
                                  }
  }while( k!= '\n' );
                  } //- - - สิ้นสุดบล็อกฟังก์ชันหลัก ( main function )

static long fact( int i ) {  //- - ฟังก์ชันเรียกตัวเอง
  if( i= = 0 ) return 1;
    else return i*fact( i-1 );
                        } //- - - จบบล็อกฟังก์ชันหาค่า factorial
 } //- - - - end of class
```

4.1.11.7 พรอพเพอร์ตี้ (Properties)

ใช้งานพรอพเพอร์ตี้ของระบบ(System Properties) เพื่อเรียนรู้สิ่งแวดล้อมของจาวาจำพวก เครื่องระบบปฏิบัติงาน โปรแกรมจาวาพรอพเพอร์ตี้ช่วยให้ข้อมูลข่าวสารเกี่ยวกับผู้ใช้ (User) ในระบบขณะนั้นผู้ใช้

ท่านใดกำลังใช้งานจาวาบ้าง รู้ถึงชื่อล็อกอิน (Login Name) , ไดเรกทอรีหลัก (Home Directory) , ไดเรกทอรีที่กำลังทำงานอยู่ (Current Directory)

ตัวอย่าง พรอพเพอร์ตี้ต่างๆที่มีใน java.util.Properties;(Subclass ของ Hashtable)

Property Name	ความหมาย
java . version	เวอร์ชันของจาวาที่ใช้
java . vender	รายละเอียดเกี่ยวกับผู้ขาย
java . vender . url	URL ของผู้ขาย
java . class . version	เวอร์ชันของคลาส
java . class . path	PATH ของคลาส
os . name	ระบบปฏิบัติการที่ใช้
os . arch	สถาปัตยกรรมของระบบปฏิบัติการ
os . version	เวอร์ชันของระบบปฏิบัติการ
file . separator	"\" ภายใต้ระบบปฏิบัติการ Windows
path . separator	";" ภายใต้ระบบปฏิบัติการ Windows
line . separator	"\r\n" ภายใต้ระบบปฏิบัติการ Windows
user . name	User Account Name
user . home	Home Directory
user . dir	User 's Current working directory

ตารางที่ 4.5 พรอพเพอร์ตี้ต่างๆที่มีใน java.util.Properties

สามเมทธอดที่จะเข้าถึง ข้อมูลของ พรอพเพอร์ตี้ ได้คือ

getProperty(String key) ส่งค่ากลับเป็นชื่อของพรอพเพอร์ตี้

getProperty(String key ,String def) ส่งค่ากลับเป็นชื่อของพรอพเพอร์ตี้ และ ดีฟอลท์(default)

getProperty(); ส่งค่ากลับ พรอพเพอร์ตี้ของระบบ ทั้งหมดใน java.util.Properties

การแก้ไขค่า Properties ใน Hashtable โดยใช้เมทธอด

setProperties (Properties prop)

แต่ค่าใหม่จะถูกแทนที่ค่าเดิมในลิสต์ทั้งหมดเลย

4.1.12 การเขียนโปรแกรมภาษาจาวาแบบแอพเพลท (Java Applet)

การเขียนจาวาแอพเพลทนี้ จะเป็นการเขียนโปรแกรมแบบเหตุการณ์กำหนด (Event Driven Programming) การทำงานของโปรแกรมไม่ได้ไล่จากบนลงล่าง เหมือนกับภาษาทั่วๆไปแต่การทำงานจะเป็น

ดังนี้คือ ระบบปฏิบัติการจะคอยตรวจสอบ เหตุการณ์ที่เกิดขึ้น แล้วเก็บเรียงเป็นลำดับไว้ เพื่อส่งให้แก่โปรแกรมที่สมควรจะได้รับเหตุการณ์ที่ได้รับมา เช่น เหตุการณ์ที่เกิดจากเม้าส์ เหตุการณ์ที่เกิดจากคีย์บอร์ด เหตุการณ์ที่เกิดจากเครื่องพิมพ์ หรือแม้กระทั่งข้อมูลที่ได้รับจากระบบเน็ตเวิร์คของการ์ดอีเธอร์เน็ต

ทุกๆโปรแกรมที่ทำงานแบบนี้ จะต้องมีระบบตรวจสอบเหตุการณ์ ซึ่งจะใช้ไวล์ลูป (While Loop) โดยไวล์ลูป นี้จะวนรอบอยู่ตลอดเวลา เพื่อคอยตรวจสอบเหตุการณ์ต่างๆที่ถูกเรียงลำดับ ซึ่งได้รับมาจากระบบปฏิบัติการ จะถูกส่งเข้าสู่วงรอบเหตุการณ์นี้ทุกครั้ง เพื่อให้โปรแกรมทำงาน

จาวาแอพเพลทก็มีการทำงานอย่างนี้เช่นกัน เพียงแต่เวลาเขียน เราไม่ต้องกำหนดส่วนไวล์ลูป ที่ว่านี้เพื่อทำหน้าที่เป็นวงรอบเหตุการณ์แต่อย่างใด เพราะระบบของจาวาจะจัดการให้เองเพียงแต่กำหนดเมทธอดซึ่งทำหน้าที่ตอบสนองต่อเหตุการณ์นั้นๆก็พอแล้ว

ไฟล์ต้นฉบับ (Source Code)ภาษาจาวาแบบแอพเพลทยังคงมีรูปแบบคำสั่งใช้ได้กับจาวาแบบ แอพพลิเคชั่นททุกประการ รวมทั้งใช้โปรแกรมคอมไพล์เลอร์ Javac.exe ตัวเดียวกัน จะกันต่างตรงที่ตอนปฏิบัติงาน ดังนั้นจาวาแบบแอพเพลทจึงต้องเพิ่มเติมโครงสร้าง ดังนี้

4.1.12.1 การเขียน Source Code

ประกอบไปด้วย 4 ส่วนหลัก แต่จะต้องมีส่วนที่เป็น Class Declaration เสมออย่างน้อย 1 คลาส ส่วน Import Statement ควรจะมี Package Statement และ Interface Declaration นั้นจะมีหรือไม่ก็ได้

4.1.12.1.1 Package Statement คือส่วนที่รวบรวมคลาสไลบรารีชุดใหญ่ ตัวอย่างเช่น

package COM.MCP.Samsnet.tig ;

4.1.12.1.2 Import Statement คือส่วนย่อยๆของ Package อีกที ตัวอย่างเช่น

import java.awt.Graphics;

4.1.12.1.3 Class Declaration จะต้องมีอย่างน้อย 1 คลาส การตั้งชื่อคลาส ต้องมี public และ extends เสมอเพราะจาวาแบบแอพเพลท นี้ไม่สามารถ ปฏิบัติงาน แบบเดี่ยวๆได้จำเป็นต้องเรียกใช้คลาสอื่น หรือให้คลาสอื่นมาเรียกใช้งาน ตามแบบฉบับของโปรแกรมเหตุการณ์กำหนดคำว่า extends นั้นหมายความว่าจะเป็นคลาสลูก (Sub Class)

4.1.12.1.4 Interface Declaration ใช้เมื่อต้องการอิมพลิเม้นท์ (Implement) ร่วมกับส่วนอินเตอร์เฟส (Interface) เช่น การใช้เธด

<u>ตัวอย่าง</u>

```
//--- ไม่มีส่วน package statement
import java.applet. *;//-- ส่วน import statement
import java.awt. *;  //-- ส่วน import statement
public class ChaiyaTest extends Applet { //-- ส่วน class declaration
public void paint (Graphics g) { //-- เมทธอดตั้งชื่อตามเหตุการณ์
        g.drawString ("ChaiYa ChaiYaDum");
                                }
```

จาวาแบบแอพเพลทไม่จำเป็นต้องมีเมทธอดที่ชื่อ main() เหมือนกับจาวาแบบแอปพลิเคชั่น แต่ขณะที่ทำการคอมไพล์ จะต้องมีไฟล์โปรแกรมภาษาเอชทีเอ็มแอลรวมอยู่ด้วยเสมอ และที่สำคัญชื่อเมทธอดจะต้องตั้งชื่อตามเหตุการณ์ เท่านั้น

4.1.12.2 เหตุการณ์ตัวอย่าง

เหตุการณ์ตัวอย่างที่จำเป็นเพื่อ ที่ควรจะได้รับการตอบสนองด้วยเมทธอดที่เหมาะสม

init() เมทธอดนี้จะถูกเรียกเมื่อแอพเพลทเริ่มการทำงานครั้งแรก ครั้งเดียวเท่านั้น ควรใช้เพื่อเตรียมสถานะเริ่มต้นของโปรแกรม เช่น การโหลดภาพ หรือกิจกรรมใดๆก็ตามที่เราคิดว่า จะต้องทำเพียงครั้งเดียว หรือควรจัดการให้เรียบร้อยก่อนเข้าสู่การทำงานจริงๆ ควรนำมาไว้ยังเมทธอดนี้

start() จะถูกเรียกหลังจากเมทธอด init() และจะทำงานทุกครั้งเมื่อแอพเพลทถูกเรียกให้กลับมาทำงาน สามารถนำไปสั่งให้เธดทำงานได้ เช่น mythread start().

Stop() ทำงานตรงกันข้ามกับเมทธอด start() คือจะถูกเรียกเมื่อออกจากโฮมเพจหน้านั้นๆ สามารถสั่งให้เธด หยุดทำงานได้ (ฆ่าเธด) เช่น mythread stop();

Destroy() ทำงานต่อจากเมทธอด stop() เพื่อเคลียร์ ทรัพยากรต่างๆ ที่ถูกเรียกใช้ก่อนหน้านี้ เช่น ยกเลิกการติดต่อกับระบบเน็ตเวิร์คใดๆ ที่เราติดต่อยู่ หรือหากมีกิจกรรมใดๆ ก็ตามที่เราควรตะทำก่อนจบการทำงานของแอพเพลท ควรนำมาไว้ยังเมทธอดนี้

Paint() จะทำงานเมื่อต้องการ วาดภาพภายในพื้นที่แสดงผลกของแอพเพลทใหม่ และเมทธอด paint() คือเมทธอดเดียวที่เราใช้แสดงผลทางกราฟิก

Update() ทำหน้าที่คล้ายกับเมทธอด paint() แต่เมทธอด update() จะไม่ลบพื้นที่แสดงผลก่อนการแสดงกราฟิกใหม่ ดังนั้นหากเราจะปรับปรุงภาพเดิมควรทำ ที่เมทธอดนี้ แล้วใช้เมทธอด update() ไปเรียกใช้เมทธอด paint() อีกที

MouseDown() จะถูกเรียกเมื่อมีการกดปุ่มของเม้าส์ ภายในพื้นที่แสดงผลของแอพเพลทและภายในเมทธอด mouseDown() ยังสามารถรับค่าตำแหน่งของเม้าส์ได้อีกด้วย ดังนี้

```
public boolean mouseDown(Event evt, int x, int y)
```

MouseUp() จะถูกเรียกเมื่อปุ่มของเม้าส์ถูกปล่อย หลังจากการกด และก็จะได้รับตำแหน่งของเม้าส์ได้เช่นกันคือ

```
public bookean mouseUp(Event evt, int x, int y)
```

MouseMove() จะถูกเรียกเมื่อมีการเลื่อนเม้าส์ภายในพื้นที่แสดงผลของแอพเพลท โดยไม่มีการกดปุ่มใดๆ และสามารถรับค่าตำแหน่งของเม้าส์ได้

```
public boolean mouseMove(Event evt, int x, int y)
```

MouseDrag จะถูเรียกเมื่อมีการเลื่อนเม้าส์ภายในพื้นที่แสดงผลของแอพเพลท พร้อมทั้งกดปุ่ม ของเม้าส์ในขณะเคลื่อนที่ด้วย และสามารถรับค่าตำแหน่งของเม้าส์ได้

public boolean mouseDrag (Event evt, int x, int y)

MouseEnter() จะทำงานเมื่อเราเลื่อนเม้าส์ จากภายนอกพื้นที่เข้ามายังพื้นที่แสดงผลของแอพเพลท และเช่นกัน สามารถรับค่าตำแหน่งของเม้าส์ที่เริ่มเข้ามาได้

MouseExit() จะทำงานเมื่อเราเลื่อนเม้าส์ จากออกนอกพื้นที่แสดงผลของแอพเพลท และเช่นกัน สามารถรับค่าตำแหน่งของเม้าส์ล่าสุดก่อนที่จะออกไปได้

public boolean mouseExit(Event evt, int x, int y)

KeyDown() จะถูกเรียกเมื่อมีการกดปุ่มใดๆของคีย์บอร์ด ในขณะที่แอพเพลทนั้นทำงานอยู่ และจะได้รับอินพุท เป็นค่ารหัสสแกนโค้ด ของคีย์ที่ถูกกด มาในรูปรหัสฐานสิบ

public boolean keyDown(Event evt, int key)

KeyUp() จะถูกเรียกเมื่อปล่อยคีย์ หลังจากมีการกดปุ่มใดๆ ของคีย์บอร์ด ในขณะที่แอพเพลทนั้นทำงานอยู่ และจะได้รับอินพุท เป็นค่ารหัสสแกนโค้ด ของคีย์ที่ถูกกด มาในรูปรหัสฐานสิบ

public boolean keyUp(Event evt, int key)

ตัวอย่างโปรแกรม CheckEvent.java

```
import java.applet.Applet ;
import java . awt . * ;
public class CheckEvent extends Applet {
        public void init ( ) {
                System.out.println(" int Event ");
                        }
        public void start( ) {
                System.out.println(" start Event ");
                        }
        public void stop( ) {
                System.out.println( " stop Event.");
                        }
        public void paint( ) {
                System.out.println(" paint Event ");
                        }
        public void update( ) {
                System.out.println(" update Event ");
                        }
        public boolean mouseDown ( Event ext , int x , int y ) {
```

```
System.out.println ("motseDown Event [ "+x+" , "+y+" ]"); return false;
                                    }
                            } //--- end of class
```

4.1.12.3 การคอมไพล์

ใช้คอมไพล์เลอร์ Javac,exe เช่นเดียวกับแบบแอปพลิเคชั่น และขณะที่ทำการคอมไพล์ จะต้องมีไฟล์โปรแกรมภาษาเอชที่เอ็มแอลรวมอยู่ด้วย เพราะว่าตอนปฏิบัติงานจาวา แบบแอพเพลทจะถูกเรียกใช้งาน โดยโปรแกรม ภาษาเอชทีเอ็มแอล โดยเรียกจากแทกแอพเพลท (TAG APPLET) ดังนี้

<APPLET CODE ="ชื่อไฟล์.class""WIDTH=200 HEIGHT = 100>

เมื่อการคอมไพล์ไม่มีข้อผิดพลาด ก็จะได้ไฟล์ตามชื่อของคลาสที่ตั้งชื่อไว้ในโปรแกรมไฟล์ต้นฉบับของภาษาจาวา มีนามสกุลจุดคลาส เช่น JRobots.class เมื่อชื่อคลาสในโปรแกรมเป็น class JRobots

4.1.12.4 การปฏิบัติงาน (execute)

จะปฏิบัติงานเป็นส่วนหนึ่งของภาษาเอชทีเอ็มแอล และเราสามารถดูผลการ ปฏิบัติงานได้จากโปรแกรม AppletViewer.exe โดยใช้คำสั่งเป็น

AppletViewer ชื่อไฟล์ .HTML<ENTER>

หรือจะดูผลการปฏิบัติงาน ได้จากโปรแกรมเว็บบราวเซอร์ต่างๆ เช่น เน็ตสแคปเนวิเกเตอร์ หรือไมโครซอฟท์อินเตอร์เน็ตเอ็กโปรเรอร์

ตัวอย่างโปรแกรมเอชทีเอ็มแอลเพื่อร่วมปฏิบัติงาน กับจาวาแอพเพลท (JRobots . HTML)

```
<HTML>
<HEAD>
<TITLE> Java  Applet Test <TITLE>
<HEAD>
<BODY>
<CENTER>
<H2> This is Java Applet  H2
<APPLET  CODE = "JRobots.class" WIDTH = 460 HEIGHT  200
</APPLET>
</CENTER>
</BODY>
</HTML>
```

ตัวอย่างการเขียนโปรแกรมจาวาแอพเพลท (JRobots . Java)

```
import java.awt.Graphics;


public class JRobots extends  java.applet.Applet
{
        public void paint( Graphics g){
                g.drawString("JRobots", 100, 60 );
                g.drawString("WELCOME TO JRobots" , 100 , 100 );
                g.drawString("Kananart & Wimai", 100 , 140 );
        } //---- end of paint function
} //---- end of class
```

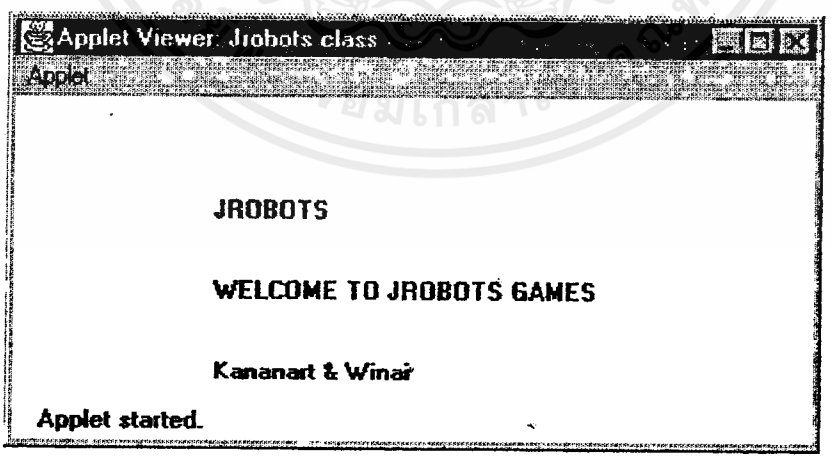
จากตัวอย่างให้ใช้โปรแกรมเท็กอิดิทเตอร์ เพื่อทำการสร้างไฟล์ทั้ง 2 ไฟล์ดังกล่าวแล้ว เก็บชื่อไฟล์เป็น JRobots.html และ JRobots.Java ตามลำดับ แล้วใช้คำสั่งในการคอมไพล์เป็น

Javac JRobots.Java <ENTER>

จากนั้นก็จะได้ไฟล์ชื่อ JRobots.class ตามชื่อคลาสในโปรแกรม จากนั้นให้ดูผลการปฏิบัติงาน ได้จากโปรแกรม AppletViewer.exe โดยใช้คำสั่ง

AppletViewer JRobot.html

จะได้ผลการปฏิบัติงานของโปรแกรม JRobots.class ดังรูปที่ 4.2



รูปที่ 4.2 แสดงผลการปฏิบัติงานของโปรแกรม JRobts.class

4.1.13 การใช้งานกราฟิก (Using Graphics)

การเขียนโปรแกรมกราฟิกในภาษาจาวามีความคล้ายคลึงภาษาซีหรือซีพลัสพลัส มากเพียงแต่เราต้องอิมพอร์ต (import) เอาคลาสที่ทำหน้าที่แสดงกราฟิกมาใช้งาน โดยมีรูปแบบ

import java.awr.Graphics;

จากนั้นภายในโปรแกรม เราก็ประกาศตัวแปรคลาสแบบกราฟิก โดยมีรูปแบบ

Graphics ชื่อตัวแปรคลาส;

ตัวอย่าง

```
import java . awt . Graphics ;   //- - - การอิมพอร์ตคลาสกราฟิกมาใช้งาน
public class GraphicsTest extends Applet {
        Graphics gph;      //- - - การประกาศตัวแปรคลาสแบบกราฟิก
        //- - - - คำสั่งต่างๆ
                                }
```

ตัวแปร Gph ในตัวอย่างสามารถนำไปใช้งานได้โดยใช้เป็นตัวนำหน้า จุด และเมทธอดที่จะเรียกใช้งาน ตัวอย่างเช่น Gph.drawImage(img.startx.starty.component);

4.1.13.1 การประกาศตัวแปรออบเจ็กแบบ Image

ตัวแปรออบเจ็กแบบ Image นี้มีประโยชน์คือจะใช้เป็นตัวแปรในการเก็บไฟล์แบบรูปภาพ มีรูปแบบเป็น Image ตัวแปร []; หรือ Image ตัวแปร_1 , ตัวแปร_2,ตัวแปร_3;

ตัวอย่าง Image img[];

หลังจากประกาศตัวแปรอ็อบเจกแบบ Image ที่เป็นอาร์เรย์ แล้วจะต้องจองพื้นที่ในหน่วยความจำให้กับตัวแปรด้วยโดยใช้เมทธอด new ในการจองพื้นที่หน่วยความจำ (Allocation) ดังนี้

img[]=new Image[จำนวนภาพสูงสุด]

จากนั้นก็จะใช้ ตัวแปร img ในการเก็บภาพได้ เช่น

img[10]=getImage(getDocumentBase(0,ชื่อไฟล์ .gif);

4.1.13.2 ตัวอย่างเมทธอดอื่นของการใช้กราฟิก

setColor(Color.สี) เมทธอดที่ทำหน้าที่เปลี่ยนสีของเมทธอดที่เกี่ยวกับการใช้สี สีแทนด้วยตัวเลข 0 ถึง 15 หรือ สามารถเขียนเป็นตัวแปรค่าคงที่ได้ เช่น red , green , yellow ,blue , gray

fillRect (startx , starty , endx ,endy) เมทธอดที่ใช้วาดรูปสี่เหลี่ยมพร้อมทั้งระบายสีภาพด้วย ซึ่งสีที่วาดนั้นมีผล มาจากเมทธอด setColor(); startx ,starty คือตำแหน่งมุมบนซ้ายสุด ของรูปที่จะวาด endx , endy คือตำแหน่งมุมล่างขวาของรูปที่จะวาด

getBackground() เมทธอดที่ส่งค่ากลับเป็นค่าของสีพื้น

drawImage(ตัวแปรแบบอิมเมจ , startx ,starty ,component) เมทธอดนี้ทำหน้าที่วาดภาพ ตัวแปรแบบอิมเมจนั้นจะเก็บรูปภาพไว้ startx ,starty คือตำแหน่งมุมบนซ้ายสุดส่วนตำแหน่งมุมล่างขวานั้นกำหนดจากเมทธอด creteImage(); component คือชื่ออุปกรณ์ปกติใช้ component เป็น this

creatImage(ความกว้าง , ความยาว) ใช้เตรียมอุปกรณ์สำหรับที่จะวาดภาพ

4.1.14 อินเตอร์แลกซิ่ง (Interlacing)

อินเตอร์แลกซิ่ง คือ เทคนิคการแสดงภาพโดย ข้อมูลของภาพ จะถูกถ่ายโอน (Transmission) เข้าไปเก็บในรูปแบบของอินเตอร์แลกซิ่ง ข้อมูลจะถูกโอนเข้ามาเรื่อยๆ จนหมด การแสดงภาพก็จะค่อยๆ แสดงได้ตั้งแต่เริ่มมีข้อมูล เริ่มจากภาพพร่ามัว จนกระทั่งเป็นภาพที่สมบูรณ์ เมื่อการถ่ายโอนข้อมูลภาพจบสิ้นสุดลง

การใช้เทคนิคอินเตอร์แลกซิ่ง จะช่วยให้ผู้ใช้สามารถเห็นภาพได้เร็วขึ้น โดยไม่ต้องรอให้การถ่ายโอนข้อมูลภาพจบสิ้นเสียก่อน แต่ก็ใช้ได้กับภาพแบบสถิต (Static Image) เท่านั้น

4.1.14.1 การแสดงภาพกราฟิกโดยใช้เทคนิคอินเตอร์แลกซิ่ง

ตัวอย่างโปรแกรม GraphicsTest.Java มีชื่อคลาสคือ GraphicsTest.class ซึ่งจะมีตัวแปรออบเจ็ก Image[10] ไว้เก็บภาพ เมทธอด init( ) ทำการอ่านไฟล์ภาพมาเก็บในตัวแปร Image เมทธอด update( ) ทำการวาดภาพซ้ำเพื่อให้ได้ข้อมูลภาพที่ทันสมัยที่สุด เมทธอด paint( ) ถูกเรียกโดยเมทธอด update( ) อีกทีทำหน้าที่วาดภาพจำนวน 10 ภาพ ตัวอย่างไฟล์ต้นฉบับดังนี้

โปรแกรมตัวอย่างการแสดงภาพกราฟฟิค GraphicsTest . Java

```
import java.applet.*;   //- - - นำส่วนของ Java Applet มาใช้
import java.awt.*;      //- - - ดึงเอาคลาส awt ทั้งหมดซึ่งจะรวมทั้ง java.awt.Graphics ด้วย
public class GraphicsTest extends Applet{
Image img[ ]= new Image[10];  //- - - ประกาศตัวแปรอิมเมจพร้อมจองพื้นที่เป็น Array 10. ค่า
Public void init( ){
  for( int i=0; i<10; I++)   //- - - นำภาพทั้ง 10. มาใส่ใน Array ของตัวแปร Image
  img[i]=getImage(getDocumentBase( ) , "image/" +i+ ".gif ");
                    }
public void paint ( Graphics g ){
          paint(g);       //- - -วาดภาพซ้ำ

public void paint( Graphics g ){
```

```
for( int i=0; i<10; i++ )          //- - -วาดภาพจำนวน 10 ภาพ
    g.drawImage( img[i], i*42 ,0, this );  //- - - วาดที่ตำแหน่ง Y คงที่ ส่วนตำแหน่ง X เปลี่ยนไป
                                    } //- - - end of paint function


} //- - - end of class
```

4.1.15 การใช้งานมิเดียแทรกเกอร์ (Using the Media Tracker)

ภาษาจาวาถูกออกแบบมาให้ใช้งานกับระบบมัลติมีเดีย อาจกล่าวได้ว่าเป็นเทคโนโลยีแห่งอนาคตของระบบมัลติมีเดียในเวิลด์ไวด์เว็บ (World Wide Web) แต่การจะรวมเอามิเดียต่างๆที่กระจายอยู่ทั่วไปตามไซท์ (Site)ต่างๆ จะเกิดปัญหาที่ตามมาคือ ความล่าช้าขณะถ่ายโอน (Transmission Delay) เวลามีการส่งผ่านข้อมูลจำพวกมัลติมีเดีย ข้ามระหว่างเน็ตเวิร์ค ภาษาจาวาจะต้องคอยข้อมูลจำพวกนี้ ภาพ , เสียง , ดนตรี และมิเดียออบเจ็ก (Media Object) อื่นๆ

การใช้งานเทคนิคอินเตอร์แลกซิ่ง แม้จะช่วยได้บ้างแต่ก็มีข้อเสีย คือจะแสดงภาพด้วยความเร็วติดต่อกันไม่ได้ ดังนั้นข้อมูลมัลติมิเดียจำพวกเสียงและดนตรี จึงเล่นไม่ได้

จาวามิเดียแทรกเกอร์ (Java Media Tracker) เป็นออบเจ็ก ของวาจาตัวหนึ่งที่จะอำนวยความสะดวกในการแสดงภาพด้วยความเร็วสูง แม้ว่าจะเสียเวลาคอยในช่วงแรกนานก็ตาม แต่หลังจากนั้นภาพจะถูกแสดงด้วยความเร็วสูง จาวามิเดียแทรกเกอร์จึงเหมาะที่จะนำไปประยุกต์ใช้งานกับมัลติมิเดียจำพวกเสียงและดนตรี รวมทั้งการแสดงภาพเคลื่อนไหวด้วย

มิเดียแทรกเกอร์ (Media Tracker) จะทำการโหลดภาพทั้งหมดเข้าไปเก็บในลิสต์ (List) ของอิมเมจ จนหมดทุกภาพก่อน จากนั้นก็จะนำภาพออกมาแสดงได้ด้วยความเร็วสูง

4.1.15.1 จาวามิเดียแทรกเกอร์คลาส (Java Media Tracker class) เป็นส่วนหนึ่งของ AWT Package มีรายการเมทธอดดังนี้

4.1.15.1.1 แฟลก (Flag)

final static int LOADING แสดงสถานะว่า Media Object กำลังถูกโหลด

final static int ABORTED เมื่อการโหลดภาพทำไม่สำเร็จ

final static int ERRORED เมื่อมีข้อผิดพลาดบางอย่างเกิดขึ้น

final static int COMPLETE เมื่อทำการโหลดภาพเสร็จสมบูรณ์

4.1.15.1.2 เมทธอด (Method)

Media Tracker (Component comp); เมทธอดที่บอกให้เริ่มต้นใช้งานมิเดียแทรกเกอร์ได้ โดยต้องประกาศตัวแปรออบเจ็กแบบมิเดียแทรกเกอร์ ก่อนโดยมีรูปแบบดังนี้

Media Tracker ชื่อตัวแปร;

จากนั้นต้องทำการจองพื้นที่ในหน่วยความจำให้กับตัวแปร ดังนี้

ชื่อตัวแปร = new Media Tracker (ชื่ออุปกรณ์);

ตัวอย่างการเริ่มใช้งานมิเดียแทรกเกอร์

Media Tracker tck = new Media Tracker (this);

void addImage (Image img , int id); เป็นเมทธอดที่ทำหน้าที่ในการนำข้อมูลของภาพ ที่ได้จากการอ่านไฟล์ (ใช้เมทธอด getImage ( ) ในการอ่าน) มาเก็บในลิสต์ข้อมูล (Data List) ของมิเดียแทรกเกอร์ โดย Image img คือตัวแปรอ็อบเจ็กแบบอิมเมจ int id คือหมายเลขของลิสต์ข้อมูลในมิเดียแทรกเกอร์

ตัวอย่าง

```
Image img = new Image [ 10 ];
Media Tracker tacker = new Media Tracker ( this );
Tacker . addImage ( img[ 10 ] , 0 );
```

Synchronized void addImage (Image img , int id , int w , int h); หน้าที่เหมือนกับ void addImage (Imgae img , int id); เพียงแต่ส่งค่าความกว้าง และความสูงของภาพไปด้วย ซึ่งค่าดังกล่าวสามารถทำได้โดยเมทธอด createImage (int w , int h);

Boolean checkID (int id); เมื่อ รูปภาพถูกรวมเข้าไปในลิสต์โดยเมทธอด ddImage ( ) เราสามารถใช้เมทธอดนี้ ทำการตรวจสอบว่าข้อมูลเข้าไปเรียบร้อยหรือไม่ ถ้าข้อมูลภาพเข้าไปเก็บเรียบร้อยจะส่งค่ากลับเป็น true ถ้าไม่เรียบร้อยส่งค่ากลับเป็น false

Synchronized boolean checkID (int id , boolean load); เหมือนกับ boolean checkID (int id); เพียงแต่ ถ้าการรวมภาพเข้าในลิสต์ไม่สำเร็จเนื่องจากยังไม่มี การสั่งโหลดภาพเมทธอดนี้จะทำการสั่งให้โหลดภาพด้วย

Boolean chekAll ( ); คล้ายๆ boolean cheakID (int id); เพียงแต่เมทธอดนี้ตรวจสอบทุกๆ อิมเมจที่จะรวมเข้าไปในลิสต์ เพียงภาพใดภาพหนึ่งทำไม่สำเร็จก็ส่งค่า false กลับมา และต้องทำสำเร็จทุกภาพ จึงส่งค่า true กลับมา

Synchronized boolean checkAll (boolean load); คล้ายๆ boolean checkAll ( ); ตรวจสอบทุกๆ อิมเมจ ถ้าไม่สำเร็จเนื่องจากยังไม่มีการสั่งโหลดภาพ เมทธอดนี้จะทำการสั่งให้โหลดภาพด้วย

Void waitForID (int id); ใช้เมื่อเริ่มสั่งงานให้โหลดภาพ และจะรอจนกว่าการโหลดภาพจะสำเร็จ

Synchronized boolean void waitForID (int id , long ms); เหมือน void waitForID (int id); แต่สามารถกำหนดเวลาในการรอสูงสุดได้ด้วย

Void waitForall ( ); เหมือน void waitForID (int id); แต่เมทธอดนี้จะคอยจนกว่าภาพทั้งหมดจะทำสำเร็จ

Synchronized boolean void waitForall (long ms); เหมือน void waitForall ( ) ; แต่สามารถกำหนดเวลาในการรอสูงสุดได้ด้วย

Int statusID (int id , boolean load) ; ใช้ร่วมกับแฟลกในการตัดสินใจสั่งงานโปรแกรมอย่างหนึ่งอย่างใด ส่วนค่าตรรกที่ส่งไปก็เหมือนกับตัวอย่างที่ผ่านๆมา คือถ้าภาพยังไม่โหลดค่า true คือสั่งให้โหลดด้วย ตัวอย่างเช่น

```
If ( traker statusID ( 0 .true ) & Mrdia Tracker . ERRORED ) }
        //- - - กรณีเกิดข้อผิดพลาดขึ้น
                                                    }
```

int statusAll (boolean load); เหมือน int statusID (int id., boolean load); แต่ใช้กับทุกๆภาพ

synchronized boolean isErrorID (int id); ส่งค่ากลับเป็น true ถ้าเกิดข้อผิดพลาด

synchronized boolean isErrorAny ( ) ; เหมือน synchronized boolean isErrorID (int id); แต่ใช้กับทุกๆภาพ

synchronized object [ ] getErrorsID (int id); เมื่อเกิดข้อผิดพลาดขึ้น เมทธอดนี้จะส่งข่าวสารเกี่ยวกับข้อผิดพลาดนั้น

synchronized object [ ] getErrorsAny ( ); เหมือน synchronized object [ ] getErrorsID (int id); แต่ใช้กับทุกๆภาพ

ตัวอย่างโปรแกรมการใช้มิเดียแทรกเกอร์แสดงภาพ โปรแกรม MediaTekTest . Java

```
import java.applet.*; //- - -นำส่วนของ Java Applet มาใช้
import java.awt.*;    //- - - ดึงเอาคลาส awt ทั้งหมดซึ่งจะรวมทั้ง java.awt.Graphics ด้วย
public class MediaTekTest extends Applet implements Runnable {
Imag img[ ] = new Image[10]; //- - -ประกาศตัวแปรอิมเมจพร้อมจองพื้นที่เป็น Array 10. ค่า
Thread thread;               //- - -ประกาศตัวแปรแบบ Thread
Media Tracker tracker        //- - - ประกาศตัวแปรแบบ Media Tracker
public void init( ) {
   tracker = new MediaTracher(this);
       for ( int I = 0 ; I < 10 ; I++ ) {
          img[i] = getImage(getDocumentBase( ) , "image/" + i + ".gif" );
          tracker . addImage ( img[i] , 0 );  //- - - นำภาพเข้าไปเก็บในลิสต์ของ Media Tracker
                    } //- - - end of for loop
                }
public void start( ){
```

```
    thread = new Thread( this );  //- - - จองพื้นที่ในหน่วยความจำให้กับตัวแปร Thread
    thread . start( );                  //- - - สั่งให้เริ่มทำงาน
                    }
public void stop( ){
    thread. stop( );                    //- - - สั่งให้ Thread หยุดทำงาน
    thread = null;
                    }
public void run( ){
    try{
      tracker.waitForID(0);             //- - - คอยจนกว่าจะโหลดภาพเสร็จ
      } catch ( InterruptedException e ) {    //- - - เรียกใช้ Interrupt ของ Thread
                        return;
                                        }
    repaint( );
                    }
public void update ( Graphics g ){
          paint(g);
                                        }
public void paint ( Graphics g ){
    if (( tracker.statusID(0,true) & MediaTracker.ERRORED ) !=0 ) {
          g.setColor ( Color.red );
          g.fillRect(0,0,size( ).width,size( ).height );
          return;
                                                    }
if(( tracker.statusID(0,true) & Media Tracker.COMPLETE ) !=0 ){
              for( int  i=0;i<10;i++ )
                    g.drawImage( img[i],i*42,0,this );
                                                              }
          else  {
                Font  font = new  Font ( " Helvetica ", Font.PLAIN,18 );
                FontMetrics  fm = g.getFontMetrics( font );
                String  str = new  String( "Loading  Images....." );
                g.setFont( font );
```

4.1.16 เทคนิคการแสดงภาพเคลื่อนไหว (Animation Techniques)

4.1.16.1 ภาพเคลื่อนไหวคืออะไร (What is Animation ?)

ภาพเคลื่อนไหว คือภาพนิ่งหลายๆ ภาพ แสดงติดต่อกันเป็นการลวงตาผู้ชมให้เห็นเป็นเสมือนภาพของวัตถุกำลังเคลื่อนไหว

โดยปกติแล้ว สมองของมนุษย์สามารถจดจำภาพที่เห็นได้ทางสายตาในช่วงระยะเวลาหนึ่ง แม้ว่าภาพนั้นจะหายไปแล้วก็ตาม และช่วงเวลาที่ว่านี้นั้นสั้นมาก

จากหลักการดังกล่าว ถ้าหากมีการแสดงภาพให้ชม แล้วปิดภาพนั้นเสีย จากนั้นก็เปลี่ยนเป็นแสดง ภาพใหม่ที่มีลักษณะท่าทางต่อเนื่องกัน ที่ตำแหน่งเดียวกัน ผู้ชมก็จะมองเห็นเป็นภาพ ภาพเดียวแต่มีการเคลื่อนไหว จากการทดลองพบว่า ถ้าแสดงภาพติดต่อกันเกินกว่า 12 ภาพ ต่อวินาที แล้วสายตาของคนปกติจะจับการกระพริบของภาพ ขณะเปลี่ยนภาพไม่ได้

ในชีวิตประจำวันเราเห็นตัวอย่างการแสดงภาพเคลื่อนไหวได้จาก การฉายภาพยนต์ โทรทัศน์ วีดีโอเกมส์ ในการฉายภาพยนต์นั้น ฟิล์มสีของภาพยนต์จะเป็นภาพต่อเนื่องกัน หลายๆภาพ 1 ภาพ เรียกว่า 1 เฟรม

ในการฉายภาพนั้นจะฉายแสงผ่านฟิล์มสีโปร่งแสงไปกระทบจอรับภาพสีขาว ก็จะเกิดเป็นภาพขึ้นที่จอ จากนั้นก็จะปิดแสงไฟ ภาพก็จะไม่ปรากฏที่จอ จากนั้นก็จะเปลี่ยนภาพใหม่พร้อมกับเปิดแสงไฟอีกครั้ง การกระทำนั้นทำด้วยความเร็วสูง คนชมภาพยนต์จึงจับการกระพริบไม่ได้ ในปัจจุบันภาพยนต์ใช้ ความเร็วในการฉายภาพเป็น 24 ภาพ ต่อวินาที (24 Frame Per Second หรือ 24 fps) และใน 1 ภาพนั้นยังทำการปิดและเปิดไฟ 2 ครั้งจึงเสมือนว่าฉายทั้งหมด 48 fps

ในโทรทัศน์ ระบบพาล (PAL) ใช้แสดง 25 ภาพต่อวินาที แต่ 1 ภาพของโทรทัศน์ นั้นจะประกอบขึ้นจากลายเส้นหลายๆเส้นดังนั้นระบบพาล ใช้ 25 ภาพต่อวินาที 625 เส้น ส่วนโทรทัศน์ระบบเอ็นทีเอสซี (NTSC) ใช้แสดง 30 ภาพต่อวินาที 525 เส้น

อาศัยหลักการข้างต้น ในคอมพิวเตอร์ก็สามารถแสดงภาพเคลื่อนไหวได้ แต่การแสดงภาพให้เร็ว นั้นขึ้นอยู่กับปัจจัยหลายอย่าง เช่น ความเร็วของซีพียู (Speed of CPU) ขนาดของหน่วยความจำเป็นต้น

4.1.16.2 ชนิดของภาพเคลื่อนไหว (Types of Animation)

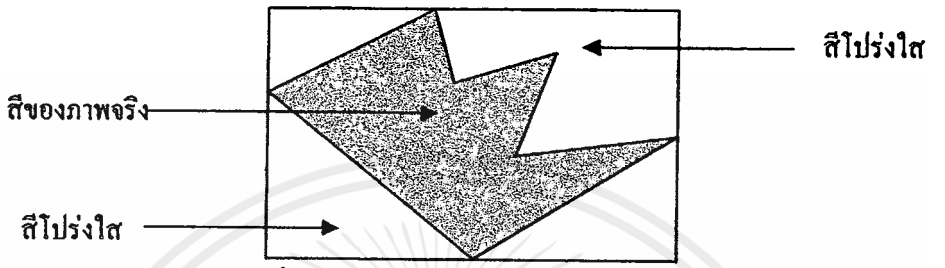
4.1.16.2.1 เฟรมเบสแอนิเมชั่น (Frame-Based Animation) เป็นเทคนิคที่ง่าย โดยอาศัยหลักการแสดงภาพต่อเนื่องกันเพียงอย่างเดียว ไม่มีการแบ่งระหว่างภาพวัตถุกับภาพ ฉากหลัง (BackGround) ภาพทั้งหมดรวมเป็น 1 เฟรม (Frame) นิยมใช้ในการฉายภาพยนต์ หรือโทรทัศน์

4.1.16.2.2 เคสท์เบสแนิเมชั่น (Cast-Based Animation) หรือ สปริทแอนิเมชั่น (Sprite Animation) เทคนิคนี้จะแบ่งภาพออกเป็น 2 ส่วน คือวัตถุ และ ฉากหลัง โดยวัตถุ นั้นจะสามารถเปลี่ยนตำแหน่งได้ แต่ฉากหลัง นั้นอยู่กับที่ จากหลังกการนี้จะพบว่าเมื่อแสดงวัตถุที่จุดใด จุดหนึ่งบนฉากหลัง แล้วเปลี่ยนที่แสดงไปตามาจุดต่างๆ ของฉากหลังก็จะดูเสมือนหนึ่งวัตถุสามารถเคลื่อนที่ได้ เทคนิคนี้นิยมใช้ในเกมส์คอมพิวเตอร์

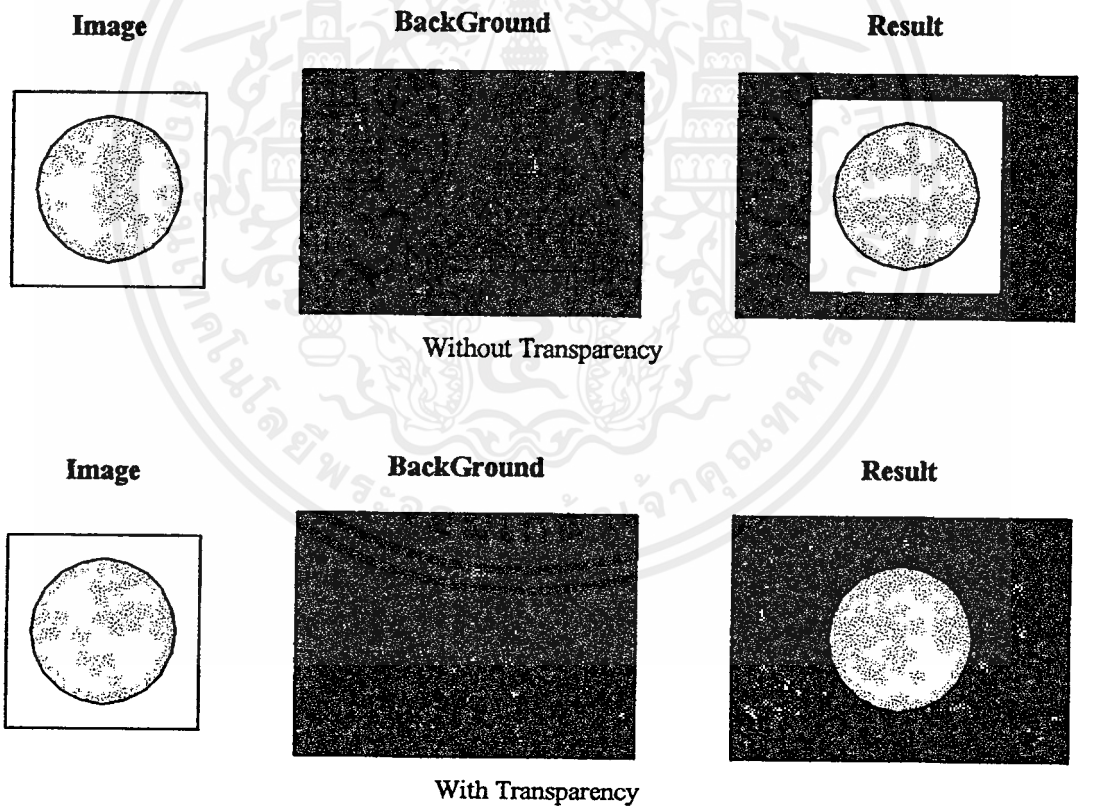
4.1.16.2.3 ฟิล์มสีโปร่งแสง (Tramsparency) ในลักษณะการเก็บภาพจะเป็นบิทแมป (Bitmap) จึงมีลักษณะเป็นภาพรูปสี่เหลี่ยม ปัญหาอยู่ที่ว่าเมื่อแสดงบิทแมป ที่วัตถุไม่ใช่รูปสี่เหลี่ยม

นั้นลงไปบนฉากหลังจะทำได้อย่างไรเพราะรอบๆ รูปวัตถุนั้นจะไปซ้อนทับกับภาพฉากหลัง ทางแก้ไขคือต้องทำให้ภาพดังกล่าวมีคุณสมบัติแบบสีโปร่งใส

ภาษาจาวานั้นสนับสนุนรูปภาพดังกล่าวอยู่แล้ว ในแบบของ GIF 89a เมื่อภาพถูกวาดลงไปทับยัง ฉากหลังจุดสี (Pixels) รอบๆ รูป วัตถุ (แทนด้วยโปร่งใส หรือ ไม่มีสี) จะไปแมทช์ (Match) กับสีของภาพ ฉากหลังแล้วก็จะถูกตัดทิ้งไป ทำให้จุดสีของฉากหลังรอบๆ วัตถุไม่เปลี่ยนแปลง



รูปที่ 4.3 แสดงการเก็บภาพแบบบิทแมป



รูปที่ 4.4 แสดงการใช้สีแบบโปร่งแสงวาดภาพ

4.1.16.3 แซดออร์เดอร์ (Z- Order)

แซดออร์เดอร์ หมายถึงความลึกของภาพในการแสดงภาพแบบสปริท เมื่อการมองภาพเหมือนกับการมองพิกัดฉากในแกน X , Y , Z ดังนั้น แซดออร์เดอร์ ก็คือภาพทางแกน Z นั่นเอง เมื่อวัตถุเคลื่อนที่ไปรอบๆ จอภาพ นั่นคือการเปลี่ยนตำแหน่งทางแกน X , Y แต่ถ้าวัตถุเคลื่อนที่ออกมานอกจอหรือหายเข้าไปในจอนั่นคือ วัตถุเปลี่ยนตำแหน่งในทางแกน Z ในจอภาพทำได้เพียงแสดงภาพซ้อนทับภาพเดิม (OverLap) จึงไม่สามารถมองเห็นได้ และ แซดออร์เดอร์ ดังกล่าวก็ไม่ใช่เทคนิคของภาพ 3 มิติ

4.1.16.4 การตรวจสอบการชน (Collision Detection)

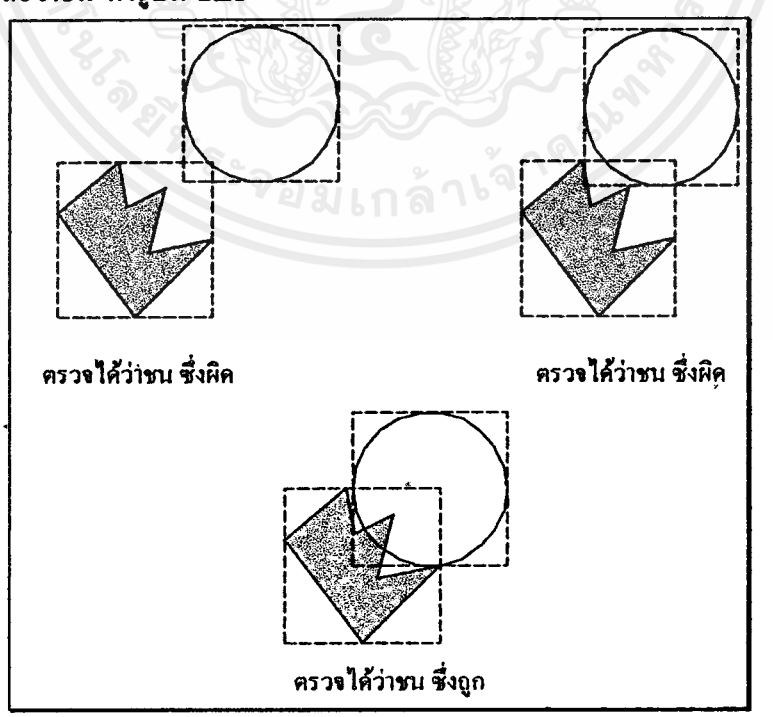
การชน คือการตรวจสอบอย่างง่ายๆ ว่าภาพ 2 ภาพ มีการชนกันเกิดขึ้น เมื่อมีการเคลื่อนที่ของรูป 2 รูป ใดๆ ตัวอย่างในเกมส์คอมพิวเตอร์ เช่นเกมส์แอสเตอร์รอยด์ (Asteroids) เมื่อมีการชนกันระหว่างยานอวกาศ กับลูกอุกกาบาศ ยานจะต้องพัง เป็นต้น

มีหลายวิธีที่ใช้ในการตรวจสอบการชน แต่จะขอเสนอ 3 วิธีดังนี้

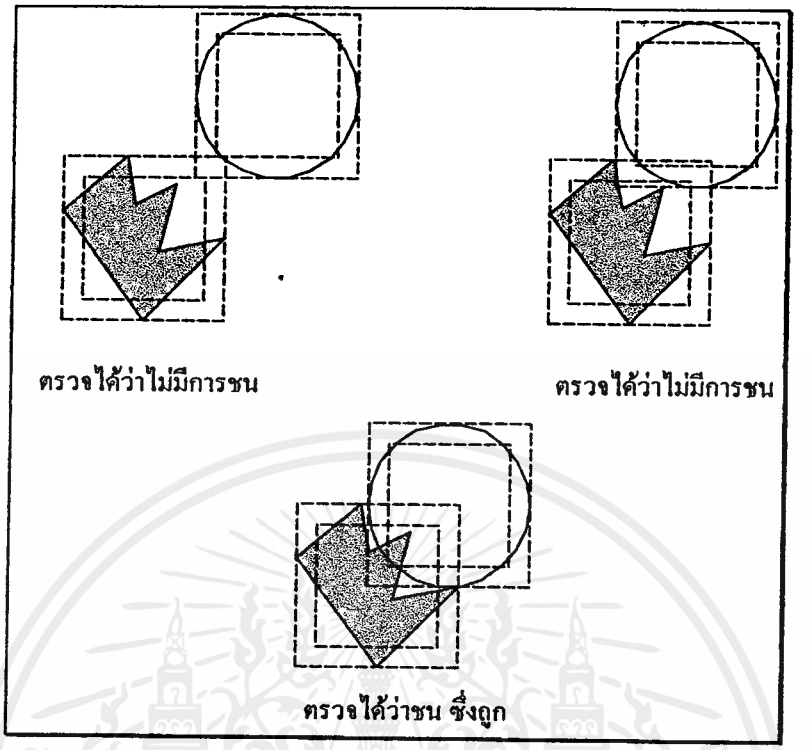
4.1.16.5.1 ซิมเปิ้ลแร็กแทงเกิ้ลคอลลิสชั่น (Simple Retangle Collision) ใช้รูปสี่เหลี่ยมของบิทแมป ในการเช็คเลย ซึ่งเป็นวิธีที่ง่าย แต่มีข้อเสียคือ กรณีที่ภาพไม่เป็นรูป สี่เหลี่ยมจะมีข้อผิดพลาดเกิดขึ้นได้ ดังรูปที่ 2.19

4.1.16.5.2 ชรั้งแร็กแทงเกิ้ลคอลลิสชั่น (Shrunk Rectangle Collision) การตรวจสอบการชนอีกวิธี ชรั้งแร็กแทงเกิ้ลคอลลิสชั่น โดยการลดขนาดของสี่เหลี่ยมใหญ่ให้เล็กลง แม้จะดีกว่าแบบแรก แต่ก็ยังไม่สมบูรณ์แบบอยู่ดี ดังรูปที่ 2.20

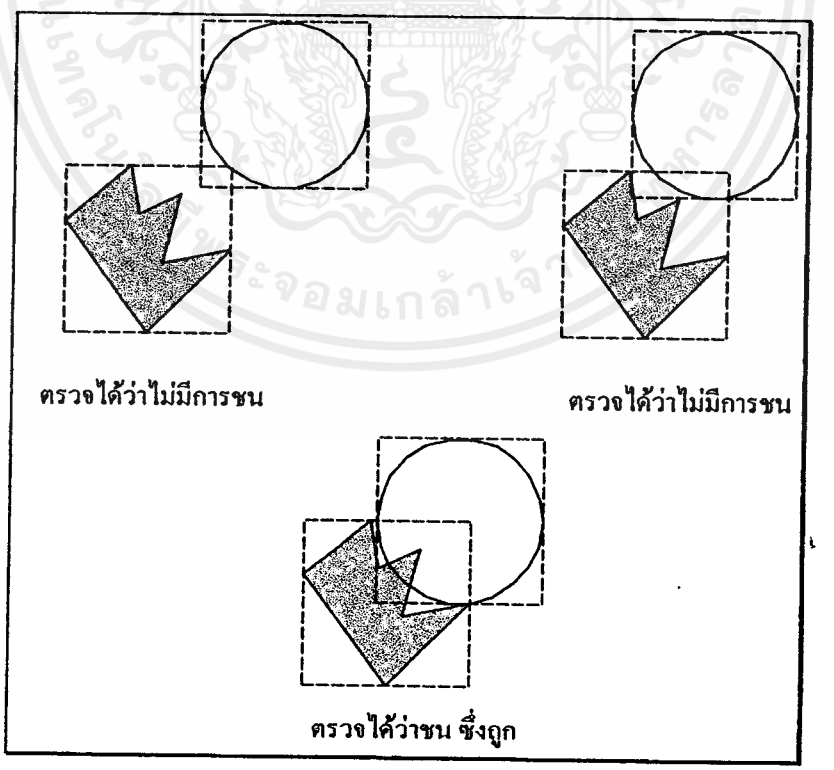
4.1.16.5.3 สปริทอิมเมจดาต้า (Sprite Image Data) การตรวจสอบการชนวิธีสามคือ สปริทอิมเมจดาต้า โดยใช้หลักการของสีโปร่งใน (Transparent Color) เมื่อจุดสี (Pixels) ของภาพจริงชนเท่านั้น จึงจะถือว่าชน ดังรูปที่ 2.21



รูปที่ 4.5 แสดงการตรวจสอบการชนแบบ ซิมเปิ้ลแร็กแทงเกิ้ลคอลลิสชั่น



รูปที่ 4.6 แสดงการตรวจสอบการชนแบบ ชรั้งแร็กแทงเกิ้ลคอลลิสชั่น



รูปที่ 4.7 แสดงการตรวจสอบการชนแบบ สปริทอิมแมจคาต้า

4.1.17 การแสดงภาพเคลื่อนไหวโดยภาษาจาวา (Implementing Frame Animation)

ภาพเคลื่อนไหวในภาษาจาวา มาจากหลักการของการแสดงภาพต่อเนื่องกันด้วยความเร็วภาพต่อวินาที ดังโปรแกรมตัวอย่าง

// - โปรแกรม Animatel1.java ➔ Animatel.class

```
import java.applet.* ;        ----- import state,emt
import java.awt.*;            ----- import statement
public class Animatel extends Applet implements Runnable {
Image img[] = new Image[10];
Thead thead;
Media Tracker tracker;
int frame = 0;
public void init() {                    // --- ทำหน้าที่โหลดภาพ และติดตามการแสดงภาพ
        tracker = new MediaTracker(this) ;
        for(int I = 0 ; I< 10; I++ )
          {     img[I]= getImage(getDocumentBase(), 'image/' + I + '.gif');
                tracker.addImage(img[I],0);
          } // --- จบหลูปฟอร์
// --- จบฟังก์ชั่น  init ()
public void start() {
        if (thread == null ) { //--- ถ้ายังไม่มีเธดชื่อ thread ให้สร้างใหม่
                thread = new Thread (this);          // --- สร้างเธดใหม่
                thread.start();                      // ---- สั่งให้เธดทำงาน
        }                                            // ---- จบหลูปอีฟ
}                                                    // --- จบฟังก์ชั่น start()
public void stop() {
        if (thread == null ) {
        thread.stop();                               // ----
        thread = null;
        }                                            // ---- จบหลูปอีฟ
}                                                    // --- จบฟังก์ชั่น stop()
public void run() {
  try {
        tracker.waitForID(0) ;                       // ---รอจนกว่าการโหลดภาพจะเรียบร้อย
```

```
        } catch ( InterruptedException e ) {          // ---เรียกอินเตอร์รัพท์ของเธด
                return;                               // ---กลับจากการอินเตอร์รัพท์
        }
  for (;;) {                                          // ---ทำงานตลอด
        if ( -- frame > 9 ) frame = 0 ;
        repaint () ;
        }
}                                                     // --- จบฟังก์ชั่น run ()
public void paint (Graphic g) {
        if ((tracker.statusID(0,true)&MediaTracker.COMPLETE) != 0 )
        { g.drawImage(img[frame],0,0,this);
        }
        else {
                Font font = new Font("Helvetica" , Font.PLAIN,18);
                FontMetrics fm = g.getFontMetrics(font);
                String str = new String ("Loading Images.....");
                g.setFont(font);
                g.drawString(str,(size().width-fm.stringWidth(str))/2,((size().height-fm.getHeight()) /
                2 )+ fm.getAscent());
        }
}                                                     // --- end of Paint function
}                                                     // --- end of class
```

4.1.17.1 การทำงานของเมทธอดต่าง ๆ อธิบายได้ดังนี้

เมทธอด init ( ) โหลดภาพ 10 ภาพแล้วนำไปเก็บยังลิสต์ของมิเดียแทรกเกอร์

เมทธอด start ( ) สร้างเธด ชื่อ thread พร้อมสั่งให้เริ่มทำงาน

เมทธอด stop ( ) ยกเลิกการใช้เธดชื่อ thread

เมทธอด run ( ) ทำงานเมื่อเธดถูกเรียกใช้ จะรอจนกว่าการโหลดภาพจะพร้อม จากนั้นจะทำงานที่ลูป for ไปเรื่อยๆ ตั้งแต่ frame = 0 จนถึง frame = 9 แล้วก็จะวนกลับมาเป็น 0 ใหม่

เมทธอด paint ( ) มี 3 ทางเลือก

- ถ้าเกิดข้อผิดพลาด (Error) จะวาดรูปสี่เหลี่ยมสีแดง
- ถ้าเสร็จสมบูรณ์ (Complete) จะวาดรูปตามตัวแปร img [frame]

- ถ้าไม่ใช่ 2 กรณีบน ก็จะถือว่าโปรแกรมพึ่งจะเริ่มทำงาน คำสั่งอยู่ระหว่างการโหลดภาพ

การทำงานของโปรแกรมเราจะพบว่า แสดงผลรวดเร็วมากจนดูแทบไม่ทัน ยิ่งถ้าขนาดของภาพใหญ่ขึ้น มาอีกเราก็จะยิ่งดูไม่รู้เรื่องเลย

4.1.17.2 การปรับแต่งความเร็วในการแสดงภาพ (Establishing a Frame Rate)

จากผลการปฏิบัติงานของโปรแกรม 5.1 Animatel. Java จะพบว่าการแสดงภาพนั้นเร็วเกินไป จนสายตามองไม่ทัน ดังนั้นควรจะมีการปรับความเร็วในการแสดงภาพ (Frame Rate) โดยใช้ตัวแปรหนึ่งตัวเป็นตัวนับการทดเวลา โดยค่าเวลาเราจะถือเป็น 1000 มิลลิเซกอันด์ (MilliSecond) และถ้าเราใส่ค่าทดเวลาในตัวแปร เราจะคำนวณอัตราการเขียนภาพได้คือ 1000 หารด้วย ค่าตัวแปร

ตัวอย่าง int delay = 200;

อัตรา Rate = 1000/200 = 5 fps

เราจะนำค่าทดเวลานี้บวกเข้ากับค่าเวลาเดิมของระบบ เพื่อหน่วงเวลา แล้วรวมโปรแกรม

เป็นโปรแกรม Animate2.java ➔ Animate2.class

```
Public void run () {
    try {
        tracker.waitForID(0);                          // -- รอจนกว่าการโหลดภาพจะเรียบร้อย
        } catch ( InterruptedExeption e) {             // -- เรียกอินเตอร์รัพท์ของเธด
                return;                                // -- กลับจากการอินเตอร์รัพท์
          }
        long t = System.currentTimeMillis();          // -- อ่านค่าเวลาเดิมของระบบ

//-- ถ้าหากว่าเธดปัจจุบันที่กำลังทำงานอยู่นี้เป็น thread ให้ทำ
        while (Thread.currentThread() == thread) {
                if ( ++ frame > 9 ) frame = 0;
                repaint();
                try {
                  t += delay;                          // -- ใส่ค่าเวลาใหม่รวมกับค่าเวลาเดิม
                  Thread.sleep(Math.max(0,t-System.currentTimeMillis()));
                } catch (InterruptedExeption e)  { break ; }
        }                                              // -- จบหลูปไวล์
}                                                      // --จบฟังก์ชั่น run()
```

โปรแกรมนี้ดูดีขึ้นมาหน่อย แต่ก็ยังเหมือนว่าภาพยังไหวๆ ยังไงชอบกล เหมือนภาพซ้อนกัน

4.1.17.3 การปรับแต่งฟริกเกอร์ (Eliminating Flicker)

จากโปรแกรม 4.1.2 ถึงแม้ว่าจะมีการหน่วงเวลาเพื่อปรับอัตราการแสดงภาพให้พอดีกับระดับความเร็วของสายตามนุษย์แล้ว ก็จะพบว่าการแสดงภาพยังไม่สมบูรณ์ คือพอเราแสดงภาพที่ 1 .แล้วในชั่วระยะเวลาหนึ่ง และจะแสดงภาพต่อไปเราก็แสดงทับภาพเดิม จากคุณสมบัติของสายตามนุษย์ที่จะจดจำภาพได้ในชั่วระยะเวลาหนึ่งนั้น

จะกลายเป็นว่า เราเห็น 2 ภาพในขณะเดียวกัน การแก้ไขมีวิธีดังนี้

4.1.17.3.1 วิธีโอเวอร์ไรด์ดิ้ง (Overriding the update ( ) Method)

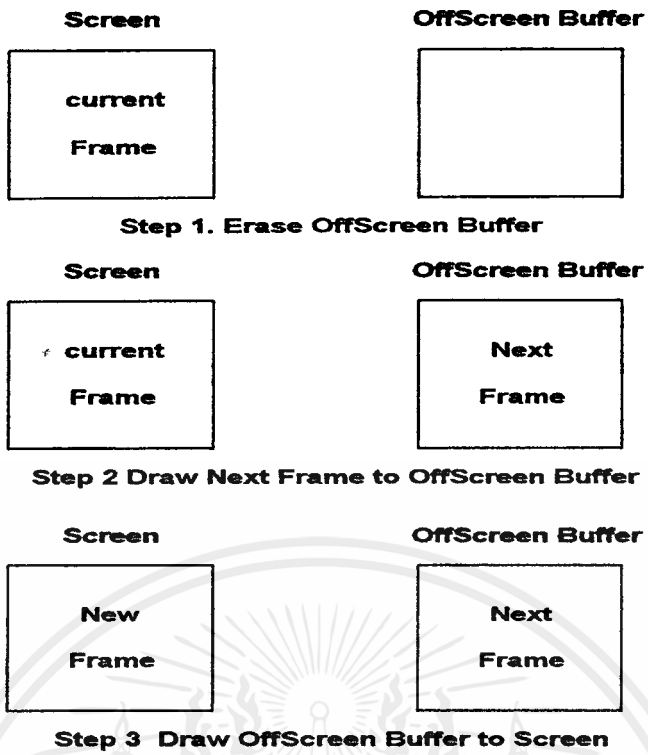
โดยการสร้างเมทธอดชื่อ update ( ) ขึ้นมาเพื่อทำการลบภาพเดิม ทิ้งไปเสียก่อนที่จะแสดงภาพใหม่ ดังนี้

โปรแกรมเป็น Animate3.java ==> Animate3.class

```
public void update( Graphics g ){
        g.setColor( getBackground( ) );          //- - กำหนดให้เป็นสีของ BackGround
                                                 //- - - วาดรูปสี่เหลี่ยมพร้อมระบายสี
        g.fillRect ( ตำแหน่งเริ่มต้นแกน X, ตำแหน่งเริ่มต้นแกน Y, ความกว้าง, ความสูง )ว
        g.getColor( getForeground( ) );
        paint(g);
                        }
```

4.1.17.3.2 วิธีดับเบิ้ลบัฟเฟอร์ริง (Double Buffering)

โดยการจองเนื้อที่หน่วยความจำเพิ่มขึ้นมาจากจำนวนภาพเดิม สองเท่า เพื่อเก็บภาพว่างๆ 1 ภาพต่อจากภาพจริง ดังรูป 2.22



รูปที่ 4.8 วิธีดับเบิ้ลบัฟเฟอร์ริง

การโปรแกรมโดยเพิ่มตัวแปรอีก 2 ตัวคือ

Image offImg;

Graphics offGph;

จากนั้นแก้ไขเมทธอด update ( ) ใหม่เป็นโปรแกรม Animate4.java

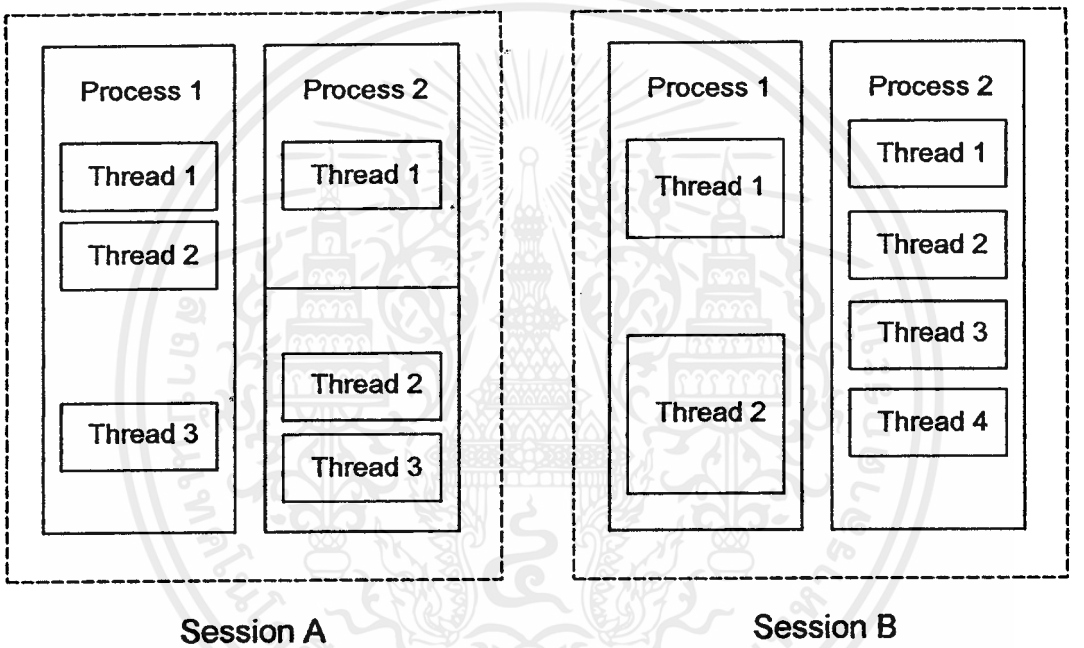
โปรแกรม Animate4.java

```
public void update (Graphics g) {
// ---- สร้างส่วนที่ใช้ในการปิดหน้าจอ
        Dimension dim = size();
        If (offGph == null ) {
        OffImge =  createImage( dim.width.dim.height );
        OffGph = offImg.getGraphics( );
                }
--- ลบภาพเดิมทั้ง
offGph.setColor( getBackground( ) );
offGph.fillRect(0,0,dim.width,dim.height );
offGph.setColor( Color.black );
offGph.drawImage( img[ frame ],0,0,this );  //- - วาดภาพจริง
```

```
g.drawImage( offImg,0,0,null ); //-- วาดภาพว่างๆ
                }
```

4.1.18 การใช้งานเธด (Using Thread)

เธดเป็นหน่วยย่อยที่สุด ในระบบที่ทำงานจริงๆ โปรเซสใดๆ จะประกอบไปด้วยเธด มากว่าหรือเท่ากับ หนึ่งเธดเสมอ เพื่อช่วยกันทำงาน เธดเป็นส่วนที่โค้ดของโปรเซสถูกเอ็กซีคิ้ว ดังนั้นขณะใดขณะหนึ่ง จึงมีเธดเพียงเธดเดียวเท่านั้นที่ทำงานอยู่ ดังรูป เป็นการแสดงถึงความสัมพันธ์ระหว่าง เซสชั่นของโปรเซสธรรมดา กับเซสชั่นของโปรเซสที่มีเธด



รูปที่ 4.9 แสดงการทำงานของเธดในระบบ OS/2

4.1.18.1 การใช้เธดในภาษา จาวา

จาวาได้แบ่งกลุ่มของเธดออกเป็น 7 กลุ่มดังนี้

```
public class Thread implements Runnable {
   //--- 4.1.18.1.1 Thread Class Constant
        public static final int  MIN_PRIORITY = 1;
        public static final int  NORM_PRIORITY = 5;
        public static final int  MAX_PRIORITY = 10;

   //--- 4.1.18.1.2 Static Method
```

```
        public native static Thread currentThread();
        public native static void  yied();
        public native static void sleep(long);
        public static void sleep(long, int);
        public static int activeCount();
        public static int enumerate(Thread[]);
    //--- 4.1.18.1.3 Constructors
        public Thread();
        public Thread( Runnable );
        public Thread( ThreadGroup, Runnable );
        public Thread( String );
        public Thread( ThreadGroup, String );
        public Thread( Runnable, String );
        public Thread( ThreadGroup, Runnable, String );
    //--- 4.1.18.1.4 Thread Control Method
        public void run();
        public native sychronized void start();
        public final void join();
        public final sychronized void join(long);
        public final sychronized void join(long, int);
        public final void suspend();
        public final void resume();
        public final void stop();
        public final synchronized void stop( Throwable );
        public void interrupted();
        public static boolean interrupted();
        public boolean isInterrupted();
        public void destroy();
    //--- 4.1.18.1.5 Thread Attributes
        public final native boolean isAlive();
        public final void setPriority(int);
        public final int getPriority();
        public final void setName( String );
        public final String getName();
```

```
        public final ThreadGroup getThreadGroup();
        public final void setDaemon(boolean);
        public final boolean isDaemon();
    //--- 4.1.18.1.6 Security Related Method
        public void checkAccess();
    //--- 4.1.18.1.7 Debugging Help
        public native int countStackFrames();
        public static void dumpStack();
        public String toString();
                                    }
```

4.1.18.2 เธด (Creating Thread)

ใช้คำหลัก implements Runnable ในการตั้งชื่อคลาส ดังตัวอย่าง

```
public class ชื่อคลาสเราตั้งเอง extends ชื่อคลาสพ่อ implements Runnable {
                                                                    }
```

เมทธอดที่เธดจะเข้าไปทำงาน ชื่อ run() ดังนั้นงานใดๆ ที่จะให้เธดทำจะต้องนำไปเขียนไว้ในเมทธอดนี้

```
คำสั่ง      try {
              //--- คำสั่งใดๆ
              } catch ( InterruptedException e ) {
                                    return หรือ break;
                                  }
```

try เป็นกลุ่มคำสั่งที่เธดจะสามารถกลับมาทำงานต่อได้ หลังจากที่เกิดคอนเท็กสวิทช์ (Context Switch) เมื่อหมดช่วงเวลาของเธด ดังนั้นคำสั่งใดๆ ที่จะให้เธดรอ หรือไม่ยอมให้เธดทำงาน หรือหน่วงเวลา ควรจะใส่ไว้ในคำสั่ง try

return เพื่อคืนการทำงานให้กับระบบปฏิบัติการ

break; เพื่อหลุดจาก loop try แล้วทำคำสั่งหลังจากคำสั่ง try { } catch

4.1.18.3การควบคุมเธด (Controlling Threads)

yield(); เมทธอดที่ใช้ควบคุมการสเก็ดดูเลอร์ (Scheduler) โดยการ สเก็ดดูเลอร์ จะใส่เธดอีอบ (Thread off) ในรีดดี้คิว (Ready Queue) และ เธดอีอบ ดังกล่าวนี้จะมีค่าลำดับความสำคัญ (Priority) สูงสุด

sleep(); เป็นการทำให้เธดชื่อนี้ เข้าไปรอการเยกอยู่ในรีดดี้คิว จนกว่าจะถึงค่าเวลาที่กำหนดไว้ในเมทธอด sleep(long, int);

join() เพื่อรวมค่าเวลาเป็นมิลลิเซกอันด์ ในการบอกหมดเวลาของเธด

suspend ( ) เพื่อนำเธด ออกไปจากรีดดี้คิว แต่ไม่มอบซีพียูให้ นั่นหมายความว่า เธดนี้จะไม่ได้รับซีพียู อีกเป็นเวลานาน

resume ( ); ใส่เธดไว้หลังสุดของรีดดี้คิว

4.1.18.3.1 เธดจะเริ่มทำงานเมื่อใด ?

เธดจะเริ่มทำงานที่คำสั่ง ดังตัวอย่าง

```
public void start ( ) {
        ชื่อเธด = new Thread ( this ) ;
        ชื่อเธด .start ( ) ;
                }
```

4.1.18.3.2 จะหยุดทำงานเธดได้อย่างไร (How Threads End?)

เธดจะหยุดทำงานที่คำสั่ง

```
public void stop ( ) {
        ชื่อเธด . stoop ( ) ;
        ชื่อเธด = null ;
                }
```

เมื่อเรานำการเริ่มทำงานของเธดมาไว้ที่เมทเธด start ( ) และ นำการหยุดทำงานของเธดมาไว้ที่ stop ( ) ดังนั้นเธดจะเริ่มเข้าไปอยู่ใน รีดดี้คิว ทุกครั้ง ที่หน้าโฮมเพจ (Home Page) ที่โปรแกรมอยู่ และก็จะถูกคัดชื่อออกจาก รีดดี้คิว ทุกครั้งที่ย้ายหน้าโฮมเพจ หนีไปที่อื่น

4.1.8.19 การเขียนโปรแกรมให้มีหลายๆคลาส

การเขียนโปรแกรมให้มีหลายๆคลาส สามารถทำได้โดยการเขียนโปรแกรมต่อกันเป็นไฟล์เดียว หรือจะเขียนแยกกันเป็น 1 คลาสต่อ 1 ไฟล์ก็ได้

การเขียนเป็นไฟล์เดียวกัน (แนะนำว่าไม่ควรใช้วิธีนี้) เวลาคอมไพล์ ตัวคอมไพเลอร์จะฟ้องเตือนว่า ควรจะตั้งชื่อไฟล์ของคลาสต่อๆ มา แยกต่างหาก แต่เราสามารถละเว้นการตรวจสอบ ข้อผิดพลาดตรงส่วนนี้ได้โดยใช้คำสั่งในการคอมไพล์ ดังนี้

Javac -NOWARN File_ Name.Java <ENTER>

และเมื่อคอมไพล์เสร็จจะได้ไฟล์ตามชื่อของคลาสบนสุด

การเขียนแยกกันเป็น 1 คลาสต่อ 1 ไฟล์ ให้ตั้งชื่อไฟล์ตามชื่อของแต่ละคลาส เมื่อจะทำการคอมไพล์ ทุกไฟล์จะต้องอยู่ที่ไดรว์เดียวกัน ให้ใช้คำสั่งคอมไพล์ที่ไฟล์ที่ 1 เมื่อในไฟล์ที่ 1 มีการประกาศตัวแปรคลาส และ Initialized โดยคำสั่ง New คอมไพเลอร์ ก็จะไปคอมไพล์ ไฟล์ ที่อ้างถึงนั้นให้โดยอัตโนมัติ และถ้ามีหลายๆ ไฟล์ ก็จะถูกคอมไพล์ทุกไฟล์ จนสุดท้ายจะได้ไฟล์จุดคลาสเท่ากับจำนวนไฟล์ที่เขียน

ตัวอย่างโปรแกรม // -- โปรแกรม PomApplet.ava +++> PomApplet.class

```
import java.applet.*;                    // -- import statement
import java.awt.*;                       // -- import statement
import java.io.*;                        // -- import statement

public class PomApplet extends Applet {
        public void init() {             // -- ทำหน้าที่โหลดภาพ และติดตามการแสดงภาพ
                Image img;
                SetLaout (new BorderLaout());
                Add ("North",new Label (" Pon Logo "));
                Img = getImage (getCodeBase(),"T10.gif");
                Add ("Center", new PomCanvas(img));      // -- มีการอ้างถึงไฟล์ PomCanvas
                        }                                // -- จบฟังก์ชั่น
}                                // -- end of class
```

//- - -โปรแกรม PomCanvas . java ===> PomCanvas . class

```
import java.awt.*;      //- - - import statement
public class PonmCanvas extends Canvas {
        Image  img;
        public PomCanvas( Image img ){
            localImg  =  img ;
                                        } //- - - จบฟังก์ชัน init( )
        public void paint ( Graphics g ) {
            g.drawImage( localImg,0,0,this );
                                        } //- - - จบฟังก์ชัน paint( )
                                        } //- - - end of class
```

//- - - โปรแกรม PomCanvas . html

```
<HTML>
<HEAD>
```

```
<TITLE> Java Applet Test </TITLE>
</HEAD>
<BODY>
<CENTER>
<H2> This is Java Applet </H2>
<APPLET CODE = "PomApplet.class" WIDTH=460 HEIGHT = 200>
</APPLET>
</CENTER>
</BODY>
</HTML>
```

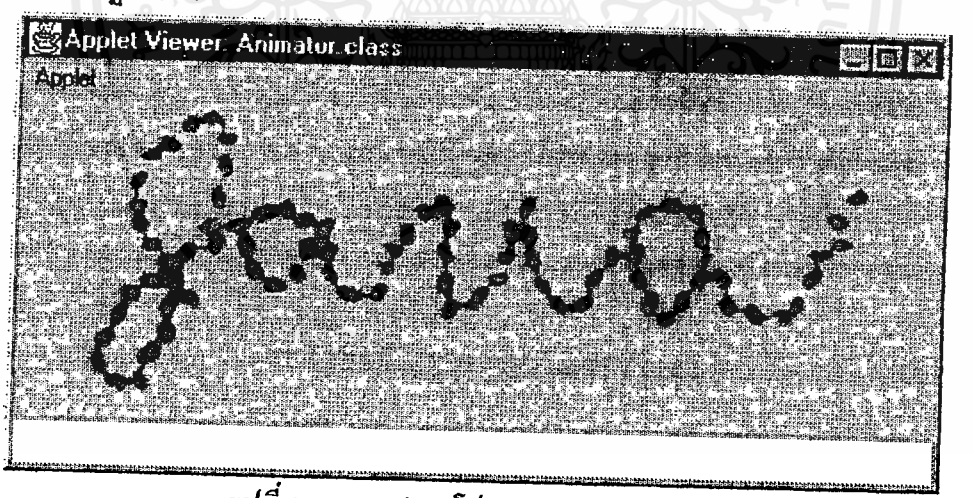
เมื่อใช้เท็กอิดิทเตอร์ สร้างไฟล์ได้ 3 ไฟล์แล้วใช้คำสั่ง

Javac PomApplett.Java <ENTER>

จะได้ไฟล์ 2 ไฟล์คือ

1. PomApplett . class
2. PomCanvas . class

ตัวอย่างผลการปฏิบัติงาน



รูปที่ 4.10 แสดงผล โปรแกรม Aminator.java

4.2 Multithreading

Thread มีลักษณะเหมือนกับโปรแกรมทั่วไป thread 1 ตัวจะมีจุดเริ่มต้น จุดสิ้นสุด โปรแกรม (เป็นเพียงตอนหรือช่วงหนึ่ง) และเวลาใดๆขณะที่ thread กำลังทำงานจะมีจุดประสงค์ของการ execute เพียงอย่างเดียว อย่างไรก็ตาม thread โดยตัวมันเองแล้วไม่ใช่โปรแกรม เพราะไม่สามารถทำงานด้วยตัวเองได้ แต่ต้องทำงานกับโปรแกรม

นิยาม thread เป็นการทำงานอย่างใดอย่างหนึ่งในโปรแกรม

การทำงานของ thread ที่จะพูดถึงไม่ใช่การทำงานของ thread ตัวเดียว แต่เป็นการทำงานแบบ multiple threads ภายในโปรแกรมเดียวกัน โดย thread จะทำงานพร้อมๆกันแต่จะทำหน้าที่คนละอย่าง

ในหนังสือบางเล่มจะใช้คำว่า *lightweight process* แทนคำว่า thread (เนื่องจาก thread มีลักษณะคล้าย process และที่เรียกว่า lightweight เนื่องจาก thread จะทำงานภายใต้สิ่งแวดล้อมเดียวกับโปรแกรมและใช้ resource ที่ถูกจองให้กับโปรแกรม) แต่อย่างไรก็ตาม thread จะต้องมี resource บางอย่างที่เป็นของตัวเอง เช่น stack , program counter เป็นต้น โค้ดที่อยู่ใน thread จะต้องทำงานภายใต้สิ่งแวดล้อมที่กล่าวมาเท่านั้น ฉนั้นเราอาจเรียก thread อีกอย่างว่า *execution context*

4.2.1 Thread Attributes

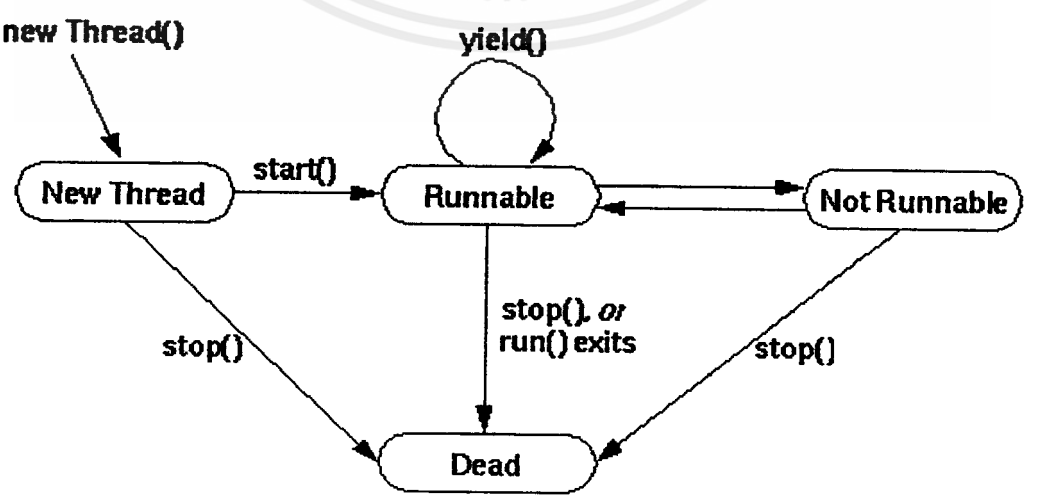
Thread ของจาวาถูก implement โดย Thread class ซึ่งอยู่ใน package java.lang thread class จะ implement ข้อกำหนดที่เป็นอิสระจากระบบของ thread

Thread Body การทำงานทุกอย่างเกิดขึ้นใน thread body (ใน run method) หลังจากที่ thread ถูสร้างและ initial แล้ว runtime system จะเรียก run method ของ thread นั้น โค้ดใน run method จะ implement การทำงานของ thread ที่สร้างขึ้น

บ่อยครั้งที่การทำงานของ thread จะเป็นการทำงานแบบ loop ทำให้บางครั้งการทำงานใช้เวลานาน คุณสามารถเลือกหนึ่งในสองวิธีต่อไปนี้ เพื่อลดขนาดของ run method สำหรับ thread ของจาวา

1. สร้าง subclass ของ Thread class ใน java.lang และ override run method
2. สร้างซึ่ง implement Runable interface ซึ่งอยู่ใน package java.lang เช่นกัน ดังนั้นถ้าคุณสร้าง instance ของ thread (ทั้งทางตรงจาก Thread class และทางอ้อมจาก subclass ของ Thread) ให้สร้าง handle ของ instance ของ Runable class ของคุณให้กับ thread ที่สร้างใหม่ Runable object นี้จะเตรียม run method ให้กับ thread

4.2.2 Thread State รูปข้างล่างนี้แสดงสถานะของ thread ในจาวา



รูปที่ 4.11 สถานะของ Thread ในจาวา

- New Thread : statement ต่อไปนี้ใช้ในการสร้าง thread ใหม่ แต่ยังไม่มีการทำงานเกิดขึ้น
  - Thread myThread = new myThreadClass ( );

  - เมื่อเป็น thread ใหม่ มันจะเป็น thread object ที่ว่างเปล่า ยังไม่มีการจอง resource ของระบบให้ เมื่อ thread อยู่ในสถานะ new thread มันสามารถจะเริ่มหรือหยุด thread ได้เท่านั้น
- Runnable : พิจารณาโค้ดต่อไปนี้
  - Thread myThread = new myThreadClass ( );
  - myThread . start ( );
  - start method จะทำการจอง resource ที่จำเป็นในการทำงานของ thread ให้ จัดตารางการทำงานของ thread และเรียก run method ของ thread เมื่อถึงจุดนี้ thread จะอยู่ในสถานะ "Runnable" ที่ไม่เรียก running เนื่องจากยังไม่มีการทำงานเกิดขึ้น เนื่องจากคอมพิวเตอร์ส่วนใหญ่มี processor ตัวเดียว ทำให้ไม่สามารถ runnable thread ทุกตัวทำงานพร้อมกันได้ runtime system ของจาวาจะต้อง implement หลักการจัดตาราง ซึ่งจะแบ่งการใช้งาน processor ให้กับ runnable thread ทุกตัว
- Not Runnable : thread จะมาอยู่ที่สถานะ not runnable เมื่อเหตุการณ์ใดเหตุการณ์หนึ่งใน 4 เหตุการณ์นี้เกิดขึ้น
  1. เมื่อมีการอ้าง sleep method ของมัน
  2. เมื่อมีการอ้าง suspend method ของมัน
  3. thread ใช้ wait method ของมันเอง เพื่อรอสถานะของ variable
  4. thread ติดอยู่กับ I/O

  การออกจากสถานะ not runnable (ไม่ว่าจะเข้ามาทางไหน) เกิดขึ้นเมื่อ
  1. ถ้า thread เข้าสู่สถานะ sleep จะออกได้เมื่อเวลาที่กำหนดไว้หมดลง
  2. ถ้า thread เข้าสู่สถานะ suspend จะออกได้เมื่อมีการเรียก resume method
  3. ถ้า thread เข้าสู่สถานะ wait เพื่อรอสถานะ variable จะออกได้เมื่อ object ที่เป็นเจ้าของ variable ปล่อยมันโดยการเรียก notify หรือ notifyAll
  4. ถ้า thread ติดอยู่กับ I/O จะออกได้เมื่อ I/O ทำงานเสร็จ
- Dead : thread สามารถสิ้นสุดได้ 2 วิธี คือ สิ้นสุดด้วยตัวเอง (ออกจาก run method ตามปรกติ) หรือ ถูกทำให้สิ้นสุด(หยุด) ซึ่ง thread สามารถถูกทำให้หยุดได้ทุกเวลาโดยการเรียก stop method stop method จะส่ง ThreadDeath object thread จะสิ้นสุดเมื่อมันได้รับ ThreadDeath exception

4.2.3 Thread Priority

การจะให้ multithread ทำงานบน CPU ตัวเดียวนั้น จะต้องมีการทำ *scheduling* โดย runtime system ของจาวาจะมีขั้นตอนในการทำ scheduling เรียกว่า *fixed priority scheduling* ซึ่งขั้นตอนเหล่านี้จะจัด thread ตาม priority (ความสำคัญ) ซึ่งจะสัมพันธ์กับ thread อื่น

thread แต่ละตัวในจาวาจะถูกกำหนดค่าความสำคัญให้ซึ่งจะอยู่ระหว่าง MIN_PRIORITY และ MAX_PRIORITY (เป็นค่าคงที่ซึ่งกำหนดใน Thread class) ณ.เวลาใดๆเมื่อ multithread พร้อมที่จะทำงาน thread ที่มีความสำคัญมากที่สุดจะถูก execute ก่อน เมื่อ thread ที่ทำงานอยู่สิ้นสุดหรือ พักการทำงานเท่านั้น thread ที่มีความสำคัญน้อยกว่าถึงจะสามารถทำงานได้

scheduling ของ CPU จะมีสิทธิในการเลือก thread เต็มที่ ถ้า thread ที่มีความสำคัญมากกว่า thread ที่ทำงานอยู่ในขณะนั้น thread ที่มีความสำคัญมากกว่าจะถูกทำ schedule ในทันที

runtime system จะไม่ใช้สิทธิ์ในการเรียกการทำงานคือเพื่อให้ thread อื่นที่มีความสำคัญเท่ากันทำงาน

ในขณะใดๆ thread อาจยกเลิกสิทธิ์ในการทำงานโดยการเรียก yield method thread สามารถมอบ CPU ให้กับ thread อื่นที่มีความสำคัญเท่ากัน (ไม่สามารถให้กับ thread ที่มีความสำคัญต่ำกว่าได้)

เมื่อ runnable thread ในระบบทุกๆตัวมีความสำคัญเท่ากัน runtime system จะทำ scheduling โดยการเลือก thread ที่อยู่ถัดไป

thread ที่มี while loop อยู่ภายใน run method เมื่อทำงานมันจะไม่ปล่อยการควบคุม CPU ให้กับ thread อื่นจนกว่ามันจะสิ้นสุดด้วยตัวมันเอง (while loop สิ้นสุด) หรือจนกระทั่งถูกอ้างสิทธิ์จาก thread ที่มีความสำคัญมากกว่า thread เช่นนี้เรียกว่า *selfish thread* ในบางระบบหาวิธีแก้ selfish thread ด้วยการทำ *time-slicing* time-slicing จะมีบทบาทเมื่อมี runnable thread หลายตัวที่มีความสำคัญเท่ากันและ thread เหล่านั้นซึ่งเป็น thread ที่มีความสำคัญสูงที่สุดแข่งกันกันเพื่อแย่ง CPU โดยระบบ time-sliced จะแบ่ง CPU ออกเป็นเวลาออกเป็นช่วงๆและจะแบ่งให้กับ thread ทุกๆตัวที่มีความสำคัญเท่ากันและมีความสำคัญสูงสุดเป็นรอบๆ ในแต่ละรอบจะให้ช่วงเวลา thread ละ 1 ช่วงเพื่อให้ทำงานของมัน จะสังเกตุได้ว่า time-sliced จะไม่รับประกันความถี่หรือลำดับการทำ schedule

Daemon Thread thread ในจาวาสามารถเป็น *daemon thread* ได้ daemon thread เป็น thread ที่ให้บริการ thread อื่นที่กำลังทำงานอยู่ใน process เดียวกัน run method ของ daemon thread ทำซ้ำไม่มีที่สิ้นสุดเพื่อรอการขอบริการจาก thread อื่น เมื่อ thread เดียวที่เหลือใน process เป็น daemon thread interpreter จะออกจากการทำงาน (เนื่องจากไม่มี thread อื่นที่ต้องบริการ)

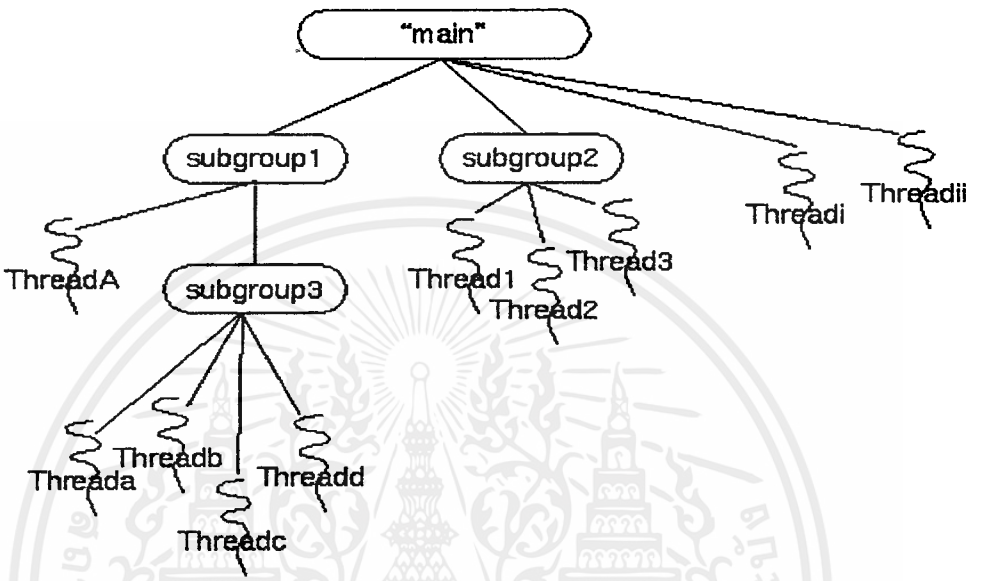
ในการกำหนดว่าให้ thread เป็น daemon thread ทำได้โดยการเรียก setDaemon method พร้อม argument "true" ส่วนการตรวจสอบว่า thread ใดเป็น daemon thread ทำได้โดยใช้ isDaemon method

Thread Group ทุก thread ในจาวาเป็นสมาชิกของ *thread group* thread group เป็นเครื่องมือในการรวม multithread เข้าด้วยกันเป็น object เดียวและใช้ thread เหล่านั้นทั้งหมดพร้อมกัน thread group ถูก implement ใน ThreadGroup class ใน package java.lang

runtime system จะทำรวม thread เข้า thread group ระหว่างที่thread ถูกสร้าง เมื่อคุณสร้าง thread ใหม่ คุณสามารถให้ runtime system รวม thread ของคุณ เข้ากับ thread group ใดๆที่มีอยู่

หรือจะตั้ง thread group ใหม่ก็ได้ เมื่อ thread เป็นสมาชิกของ thread group ใดแล้วไม่สามารถจะย้าย thread group ได้

เมื่อโปรแกรมจาวาเริ่มทำงาน runtime system จะสร้าง thread group ชื่อว่า main ถ้าไม่ได้เจาะจงไว้ thread จะถูกรวมเข้าไว้ใน main thread group (แต่ถ้าคุณสร้าง thread ใน Applet thread group ใหม่อาจใช้ชื่ออื่นที่ไม่ใช่ main)



รูปที่ 4.12 ThreadGroup class

4.2.4 Multithreaded Programs

Synchronizing Threads บ่อยครั้งที่ thread ต้องการใช้ข้อมูลร่วมกันและต้องพิจารณาถึงสถานะและกิจกรรมที่ thread อื่นทำอยู่ โดยกลุ่มของโปรแกรมที่อยู่ในสถานะการณ์นี้เรียกว่า producer / consumer โดย producer จะสร้างข้อมูล ซึ่งจะถูก consumer นำไปใช้ ซึ่งทำให้การทำงานของทั้ง producer และ consumer เข้าจังหวะกัน (synchronize) ปัญหาของการทำงานไม่เข้าจักหวะกันเรียกว่า *race conditions* เกิดขึ้นเมื่อ multithread ที่ทำงานไม่เข้าจังหวะกันเข้าถึง object ตัวเดียวกันในเวลาเดียวกัน ทำให้ได้ผลลัพธ์ที่ผิดพลาด เพื่อป้องกัน race conditions ไม่ให้เกิดขึ้นสำหรับการทำงานแบบ producer / consumer เรามีวิธีแก้ 2 วิธีคือ monitors และ ใช้ notifyAll กับ wait method

1. **Monitors :** object ที่ถูกใช้ร่วมกันระหว่าง thread สองตัวซึ่งการเข้าถึงจะต้องเข้าจังหวะกัน เรียกว่า *condition variable* monitors จะเกี่ยวกับการกำหนดข้อมูล (condition variable) และการทำงานเช่น lock ในข้อมูลเหล่านั้น เมื่อ thread ทำ monitor บนข้อมูลตัวใด แล้วมี thread อื่นมาทำการ lock บนข้อมูลตัว จะทำให้ thread ที่ทำ monitor บนข้อมูลนั้น (thread ที่ไม่ได้เป็นผู้ lock) ไม่สามารถใช้ข้อมูลตัวนั้นได้ ส่วนของโค้ดที่อยู่ในโปรแกรมที่เข้าถึง

ข้อมูลตัวเดียวกันใน thread เรียกว่า *critical sections* ในจาวาคุณสามารถกำหนด critical sections ในโปรแกรมของคุณได้โดยใช้ synchronized keyword

2. **norifyAll และ wait methods :** method ทั้ง 2 ตัวนี้เป็นสมาชิกของ java.lang.Object class

Note : notifyAll และ wait method ถูกอ้างโดย thread ที่ถือ lock เท่านั้น

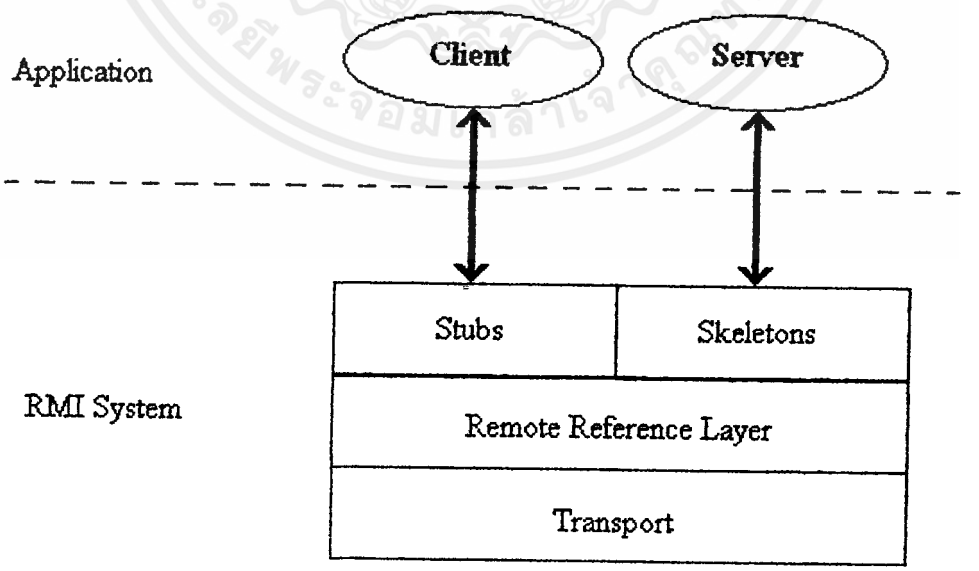
notifyAll method : ใช้ในการบอกให้กับ thread ตัวอื่นๆที่รอ บน monitor โดย thread ปัจจุบันให้ตื่นขึ้น หนึ่งใน thread ที่รออยู่ก็จะทำ monitor และทำงานต่อ

wait method : จะทำให้ thread ปัจจุบันหยุดการทำงานจนกว่า thread อื่นจะบอกมันถึงสถานะที่เปลี่ยนไป

4.3 Remote Method Invocation (RMI)

RMI เป็นกลุ่มของคลาสของจาวา ซึ่งจัดหาการคำนวณให้กับ object แบบ n-tired distributed เครื่องรับบริการสามารถติดต่อกับเครื่องที่ให้บริการ object ของจาวานั้นได้ โดยการส่งผ่านข้อมูลไปมา RMI ยังยืนยันการติดต่อไปยังเครื่องให้บริการและจัดการการแลกเปลี่ยนข้อมูลระหว่างเครื่องให้บริการและเครื่องรับบริการ

RMI ทำให้ผู้เขียนโปรแกรมสามารถสร้างโปรแกรมสำเร็จรูปที่กระจายจากจาวาไปจาวา ซึ่ง method ของ object ของจาวาที่อยู่ห่างไกลสามารถอ้างจาก Java Virtual Machine อื่นและจาก host อื่นได้ โปรแกรมจาวาสามารถเรียก object ที่อยู่ห่างไกลซึ่งมีตัวอ้างอิงไปยัง object ที่อยู่ห่างไกลนั้นได้ โดยทำการหา object ที่อยู่ห่างไกลจากตารางรายชื่อซึ่ง RMI จัดไว้ให้ หรือโดยการรับตัวอ้างอิงในฐานะ argument หรือโดยค่าที่ส่งกลับมา เครื่องรับบริการสามารถเรียก object ที่อยู่ห่างไกลในเครื่องให้บริการได้ และเครื่องให้บริการนั้นสามารถเป็นเครื่องให้บริการสำหรับ object ที่อยู่ห่างไกลอื่นได้เช่นกัน



รูปที่ 4.13 โครงสร้างของ RMI

Applet ของจาวา (client object) ติดต่อกับ object ที่อยู่ห่างไกล (server object) โดยการเข้าถึงตัวอ้างอิงไปยัง object ที่อยู่ห่างไกลผ่านอุปกรณ์ที่เรียกว่า *RMI Registry*

RMI Registry เป็นโปรแกรมสำเร็จรูปซึ่งจะทำงานอยู่เบื้องหลังขบวนการบนเครื่องให้บริการ ซึ่งจะเก็บตารางที่ให้ชื่อบริการ object ที่อยู่ห่างไกล และให้บริการติดต่อระหว่าง Applet และ object ที่อยู่ห่างไกลผ่านทางการติดต่อแบบ gateway

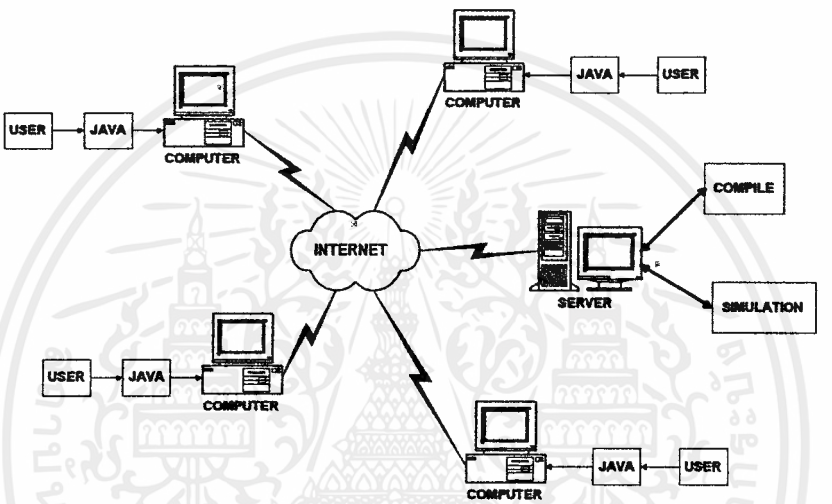
Object ที่อยู่ห่าง ไกลจะถูก Execute เป็นขบวนการที่อยู่เบื้อหลังบนเครื่องที่ให้บริการ และ จะต่อตัวเองเข้ากับ RMI Registry โดยใช้ชื่อบริการ registry จะเก็บชื่อบริการของ object ที่อยู่ห่างไกลและทำการอ้างถึง object ที่อยู่ห่างไกลนั้น

ท้ายสุด Applet จะทำการโหลดตัวที่ผูกติด object จาก RMI Registry ซึ่ง object จะใช้ตัวนี้เพื่อให้เข้าใจถึงโครงสร้างของ parameter ของ method ของ object ที่อยู่ห่างไกล จากนี้ไป Apple จะสามารถเรียกหรืออ้างถึง method ของ object ที่อยู่ห่างไกลเสมือนว่าอยู่ที่เครื่องนับบริการ argument ทั้งหมดที่ส่งผ่านกันระหว่าง Apple กับ method ของ object ที่อยู่ห่างไกลจะใช้เทคนิคที่เรียกว่า Object Serialization

บทที่ 5
การคำนวณ การสร้าง และการออกแบบ

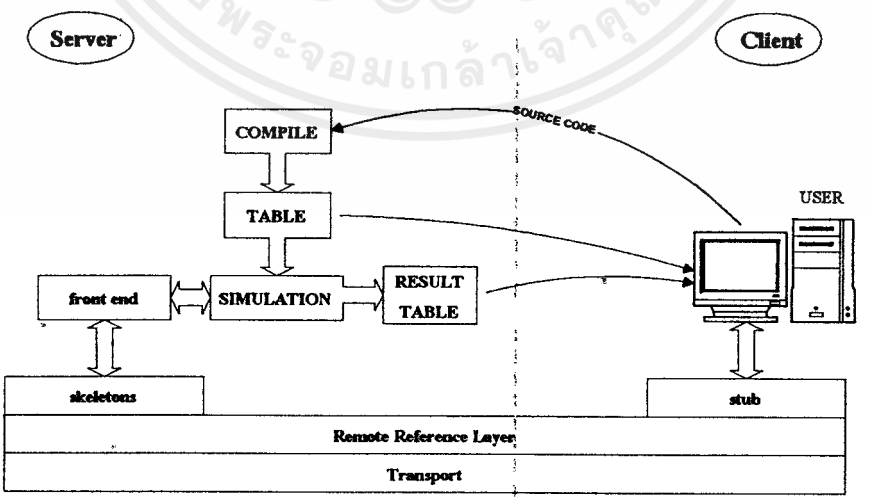
5.1 หลักในการออกแบบ

จุดมุ่งหมายของโปรแกรมเกมส์การจำลองการต่อสู้ของหุ่นยนต์ด้วยภาษาจาวานี้ ต้องการให้ผู้เล่นเขียนโปรแกรมควบคุมหุ่นยนต์ด้วยภาษาจาวา แล้วส่งโปรแกรมควบคุมมาที่เซิร์ฟเวอร์ และสามารถดูการแข่งขันโดยผ่านระบบเครือข่ายได้อย่างเรียลไทม์ (real-time)



รูปที่ 5.1 ลักษณะการเล่นของ JRobots

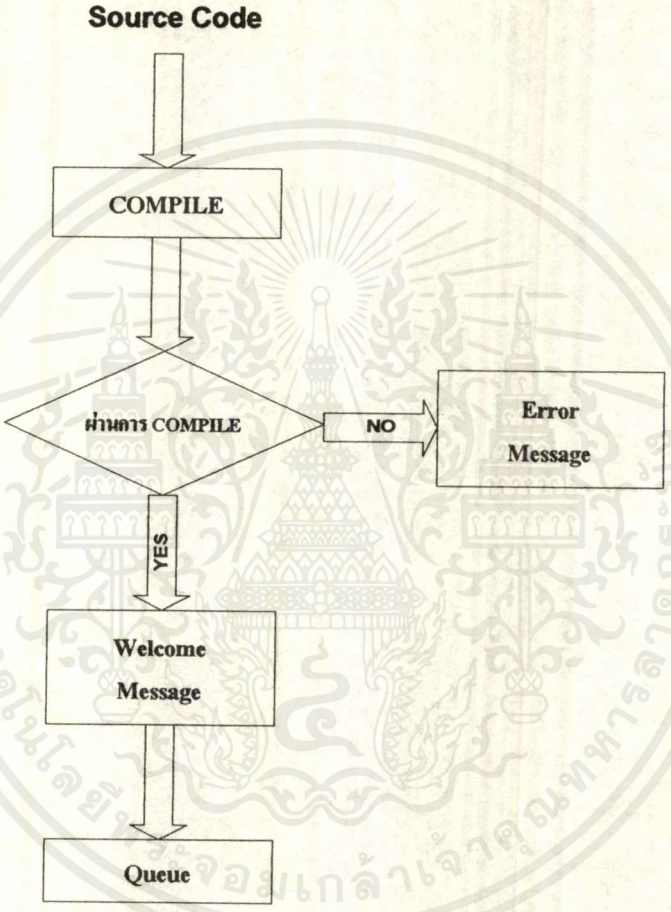
เมื่อพิจารณาลักษณะที่ต้องการดังกล่าว จึงแบ่งโครงสร้างของโปรแกรมออก 4 ส่วนโดยมีโครงสร้างดังรูปที่ 5.2



รูปที่ 5.2 โครงสร้างโปรแกรม

5.1.1 ส่วน Compile

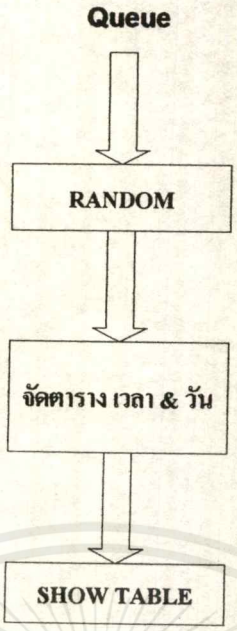
เป็นส่วนรับ Source Code เข้ามาเพื่อมาทำการ Compile ให้ได้ Byte Code เมื่อทำการ Compile เสร็จจะตรวจสอบว่า ถูกต้องหรือเปล่า ถ้าไม่ผ่าน จะแสดง error message แต่ถ้าหากผ่านการ Compile ก็จะแสดง Welcome message เพื่อให้ผู้ส่ง Source Code ได้ทราบว่า Source Code ของตนผ่านหรือไม่ หลังจากนั้นจะส่ง Byte Code ไปเก็บไว้ใน Queue เพื่อรอการจัดตารางการแข่งขัน โดยมีลักษณะดัง Flow Chart ต่อไปนี้



รูปที่ 5.3 รูปแสดง Flowchart ของส่วน Compile

5.1.2 ส่วนจัดตารางการแข่งขัน

เมื่อถึงเวลา 24.00 น. (00.00 am. Local Time) ส่วนของการสร้างตารางจะไปดึง ข้อมูลใน Queue มาทำการ Random เพื่อจัดรอบการแข่งขัน ซึ่งแต่ละรอบการแข่งขันจะมีจำนวนผู้แข่งได้ 4 คน โดยจะแสดงรอบการแข่งขันด้วยตาราง โดยมีลักษณะดัง Flow Chart ต่อไปนี้

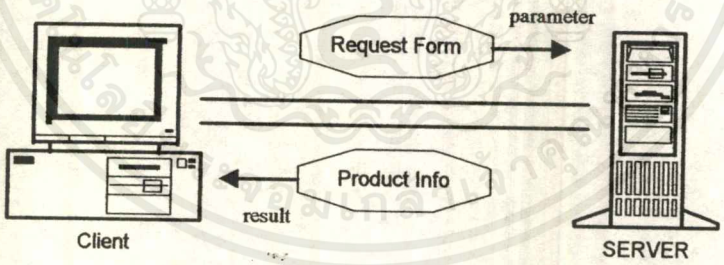


รูปที่ 5.4 แสดง Flow Chart ของส่วน สร้างตารางการแข่งขัน

5.1.3 ส่วนจำลองการแข่งขัน (Simulation)

ส่วนนี้จะทำงานแบ่งออกเป็น 2 ส่วน คือ

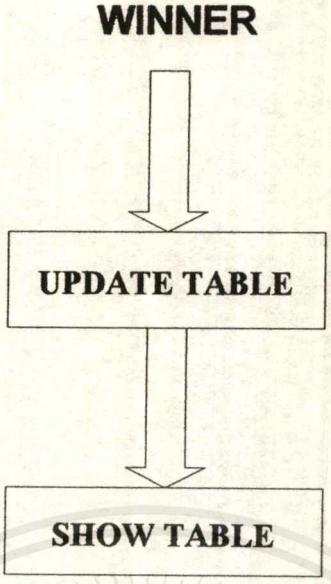
1. ส่วนเซิร์ฟเวอร์ ที่ทำการ Simulation และส่งข้อมูลการแสดง ไปให้ยังฝั่งไคลเอนต์
2. ส่วนไคลเอนต์ จะทำการติดต่อกับ เซิร์ฟเวอร์ และทำการร้องขอ ข้อมูลที่ต้องการไปยังเซิร์ฟเวอร์ และรับข้อมูลมาแปลให้เป็นภาพแสดงบนน้ำจอของฝั่งไคลเอนต์



รูปที่ 5.5 การร้องขอข้อมูลของไคลเอนต์

5.1.4 ส่วนแสดงผลการแข่งขัน (Show Result)

ส่วนนี้จะแสดงรายชื่อของผู้ชนะในแต่ละรอบ โดยนำชื่อผู้ที่ชนะในการแข่งขันของรอบนั้นๆ มาทำการ Update Table เพื่อแสดงผลการแข่งขันของรอบที่แข่งขันไป โดยมีลักษณะดัง Flow Chart ต่อไปนี้



รูปที่ 5.6 แสดง Flow Chart ของส่วน สร้างตารางผลการแข่ง

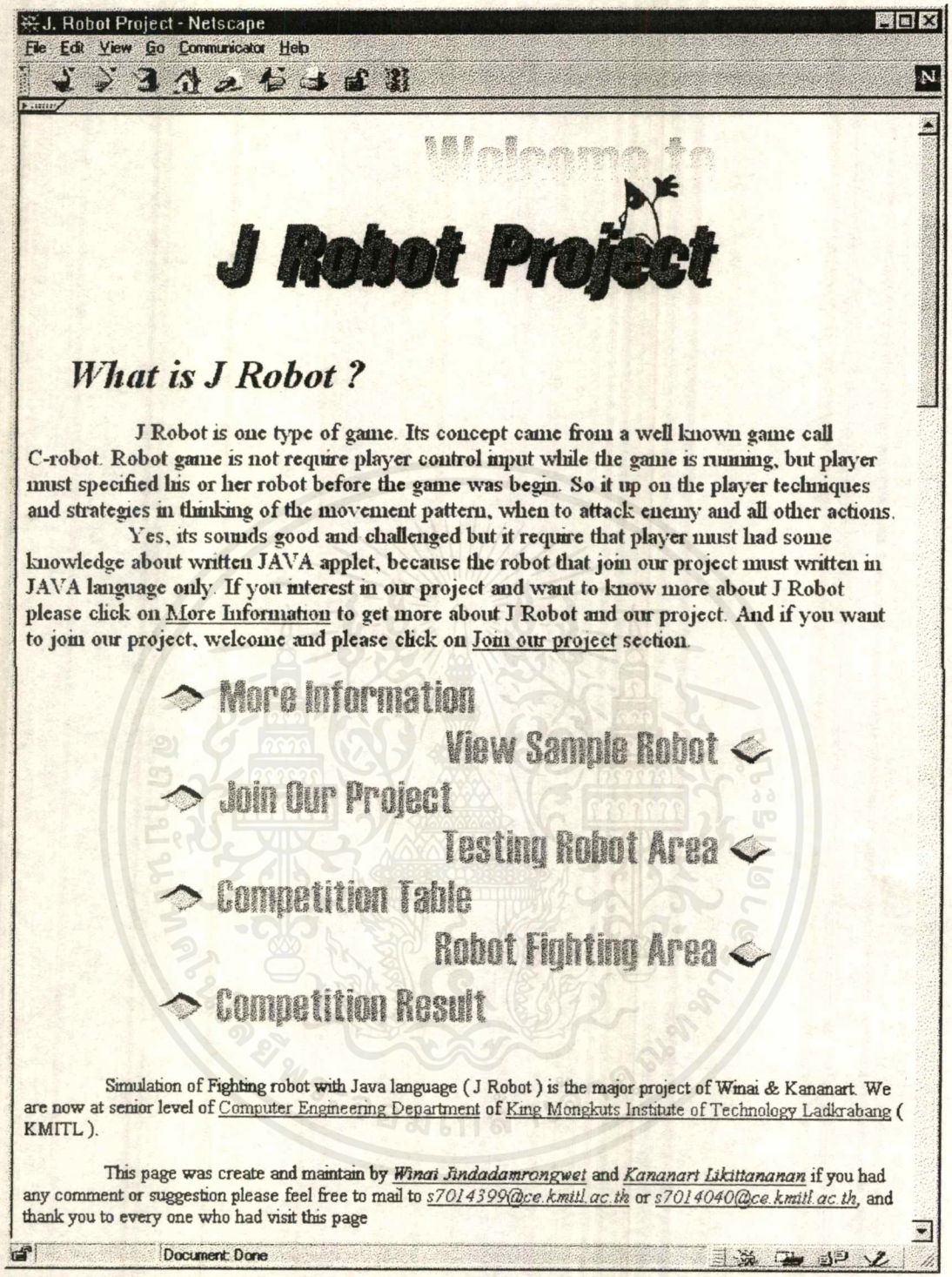
5.2 การออกแบบส่วนติดต่อกับผู้ใช้ (User Interface)

ส่วนการติดต่อกับผู้ใช้แบ่งออกเป็น 5 ส่วน ดังต่อไปนี้

5.2.1 ส่วน Main Web

เมื่อต้องเล่นเกมส์ JRobots จะต้องเข้า Web Browser และ เข้าโฮมเพจแรก จะพบกับหน้าจอหลัก ดังภาพที่ 3.5 ภายในหน้าจอจะประกอบด้วย 2 ส่วนดังนี้

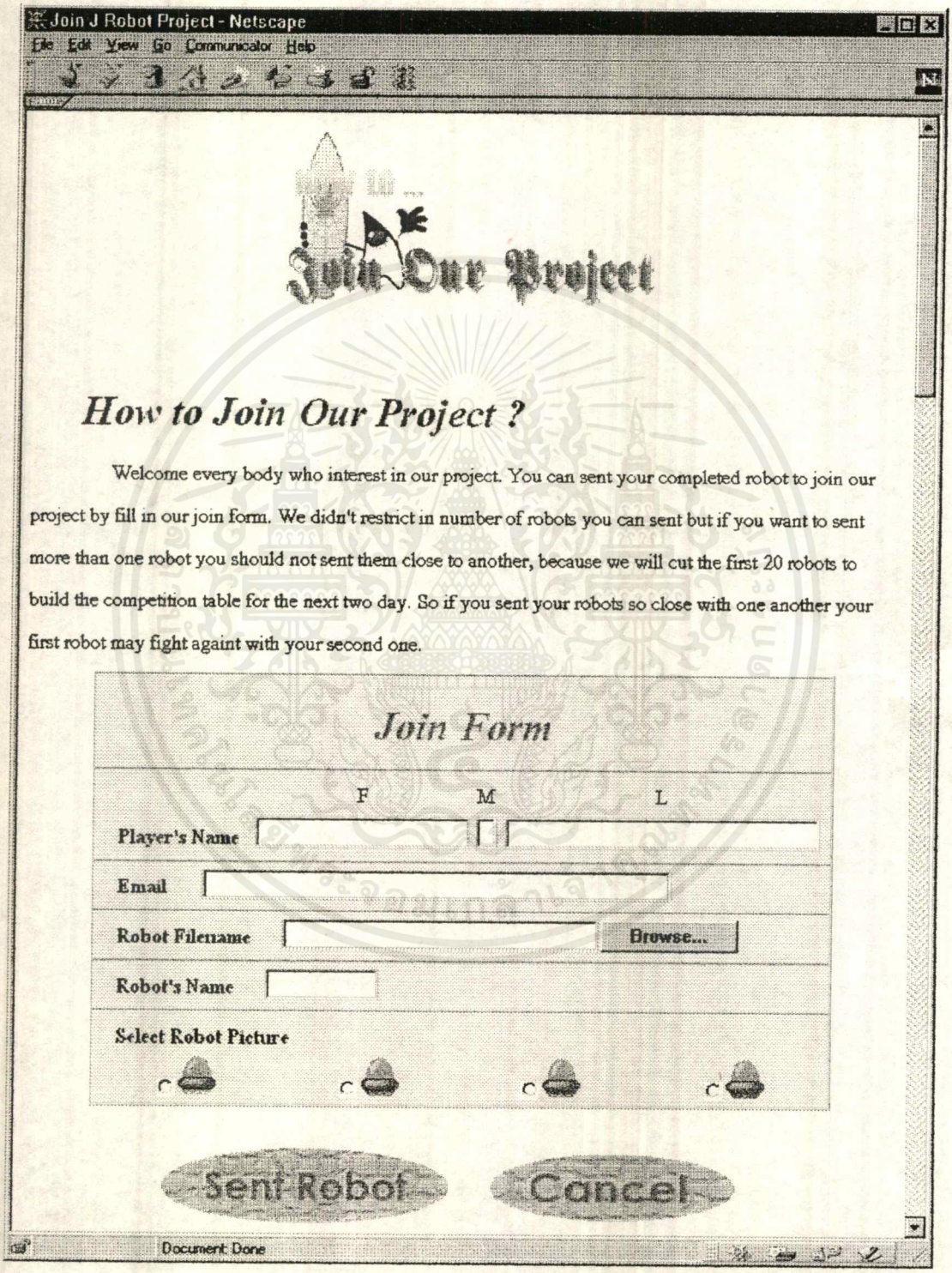
1. ส่วนแนะนำ
2. ส่วนติดต่อกับ web หน้าอื่นๆ เช่น การแข่งขัน , รายละเอียด เป็นต้น



รูปที่ 5.7 แสดงหน้าจอ Main Web

5.2.2 ส่วนการเข้าร่วมการแข่งขัน

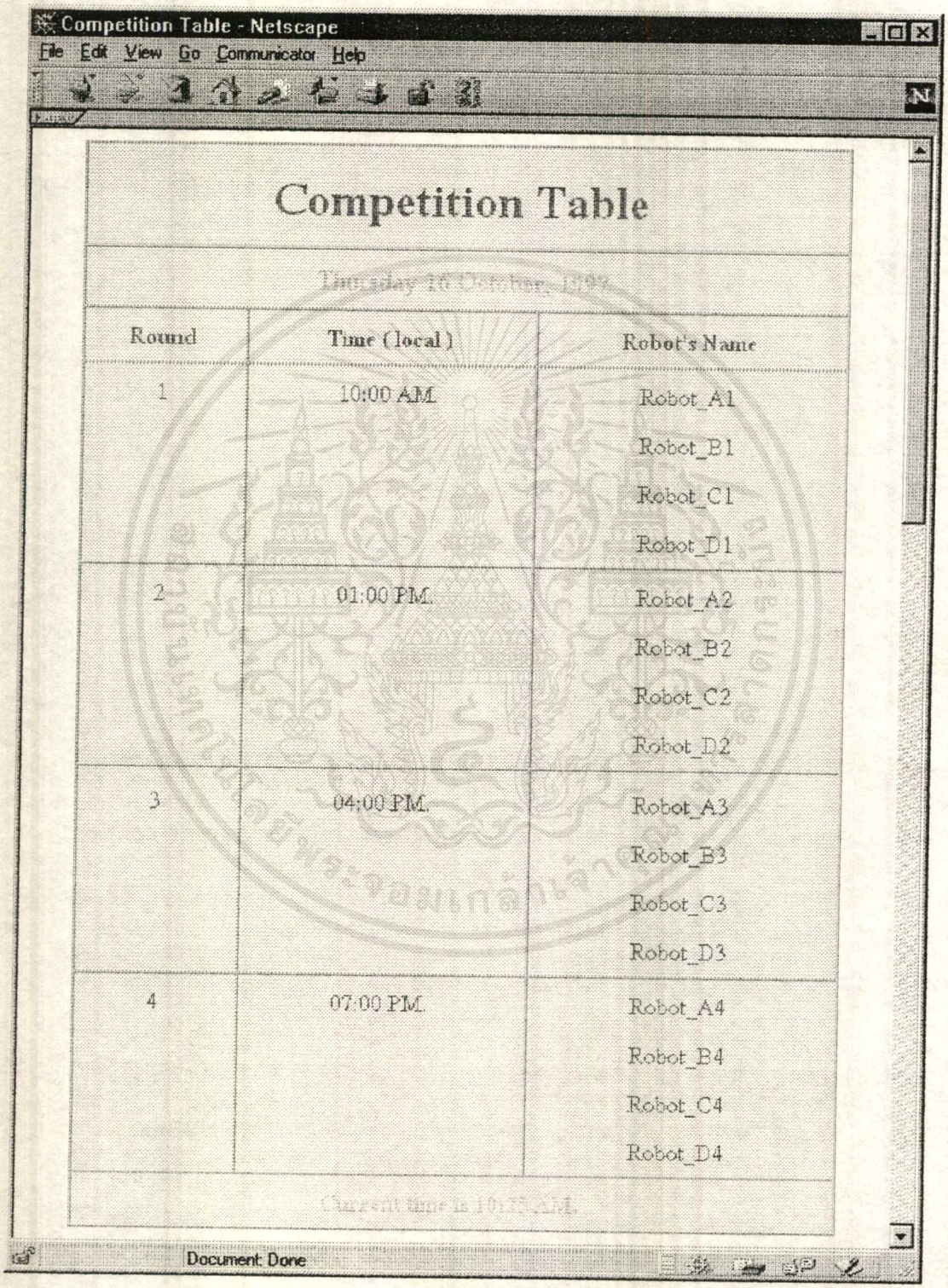
เป็นส่วนที่ให้ผู้สนใจเล่นเกมส์ จะส่งหุ่นเข้าร่วมแข่งขัน โดยหุ่นทุกตัวจะที่รับเข้ามาจะถูก Compile ใหม่ และหากเกิดข้อผิดพลาดขึ้น ก็จะแสดงข้อความแจ้งให้เจ้าของทราบว่า หุ่นมีข้อผิดพลาดเกิดขึ้น หรือ ผ่านการ Compile เรียบร้อยแล้ว



รูปที่ 5.8 หน้าจอแสดงส่วนการเข้าร่วมแข่งขัน.

5.2.3 ส่วนการแสดงตารางการแข่งขัน

ส่วนนี้เป็นในส่วนแสดงว่า หุ่นตัวไหนแข่งขันเมื่อไร และ มีหุ่นตัวไหนบ้างที่ร่วมแข่งขันด้วย โดยลักษณะของตารางแสดงได้ดังรูปที่ 3.9



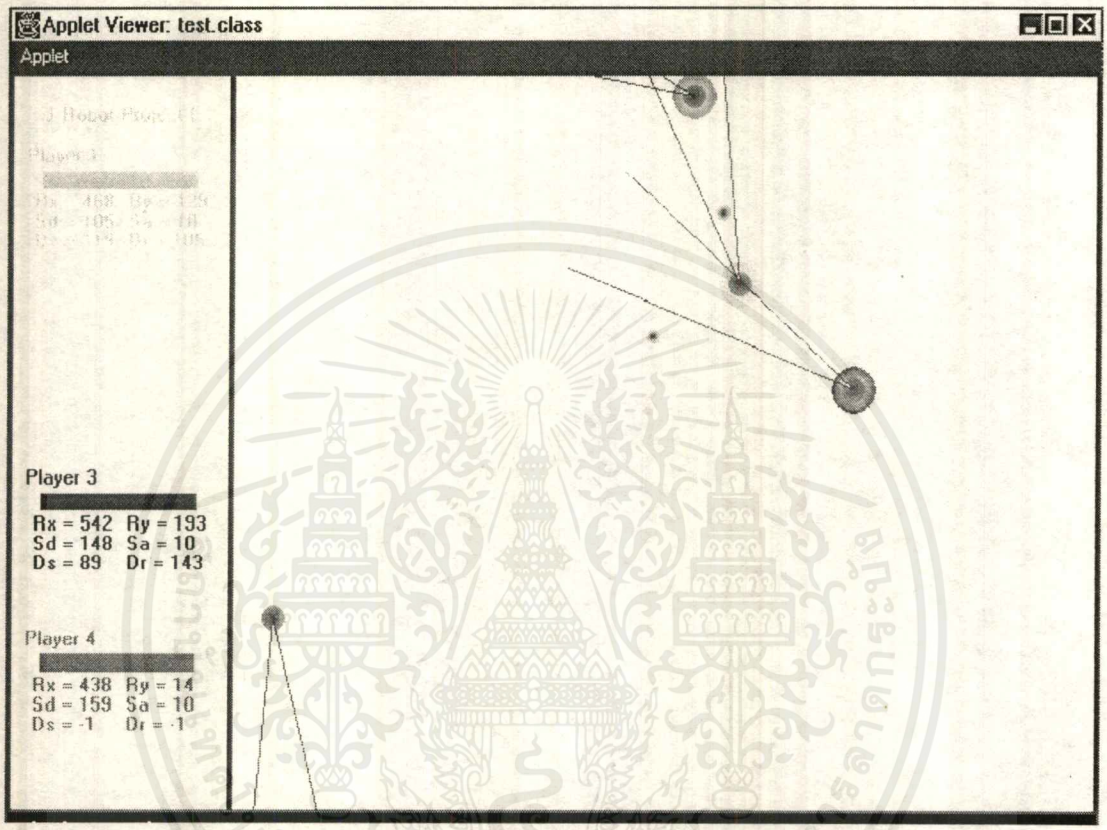
รูปที่ 5.9 หน้าจอแสดงตารางการแข่งขัน

5.2.3 ส่วนแสดงการจำลองการต่อสู้

ส่วนนี้เป็นส่วนแสดงการแข่งขัน ที่ได้กำหนดไว้ในตารางการแข่งขัน โดยภายในหน้าจอจะประกอบด้วย 2 ส่วนดังนี้

1. ส่วนแสดงการแข่งขัน
2. ส่วนแสดงสถานะของหุ่นแต่ละตัว

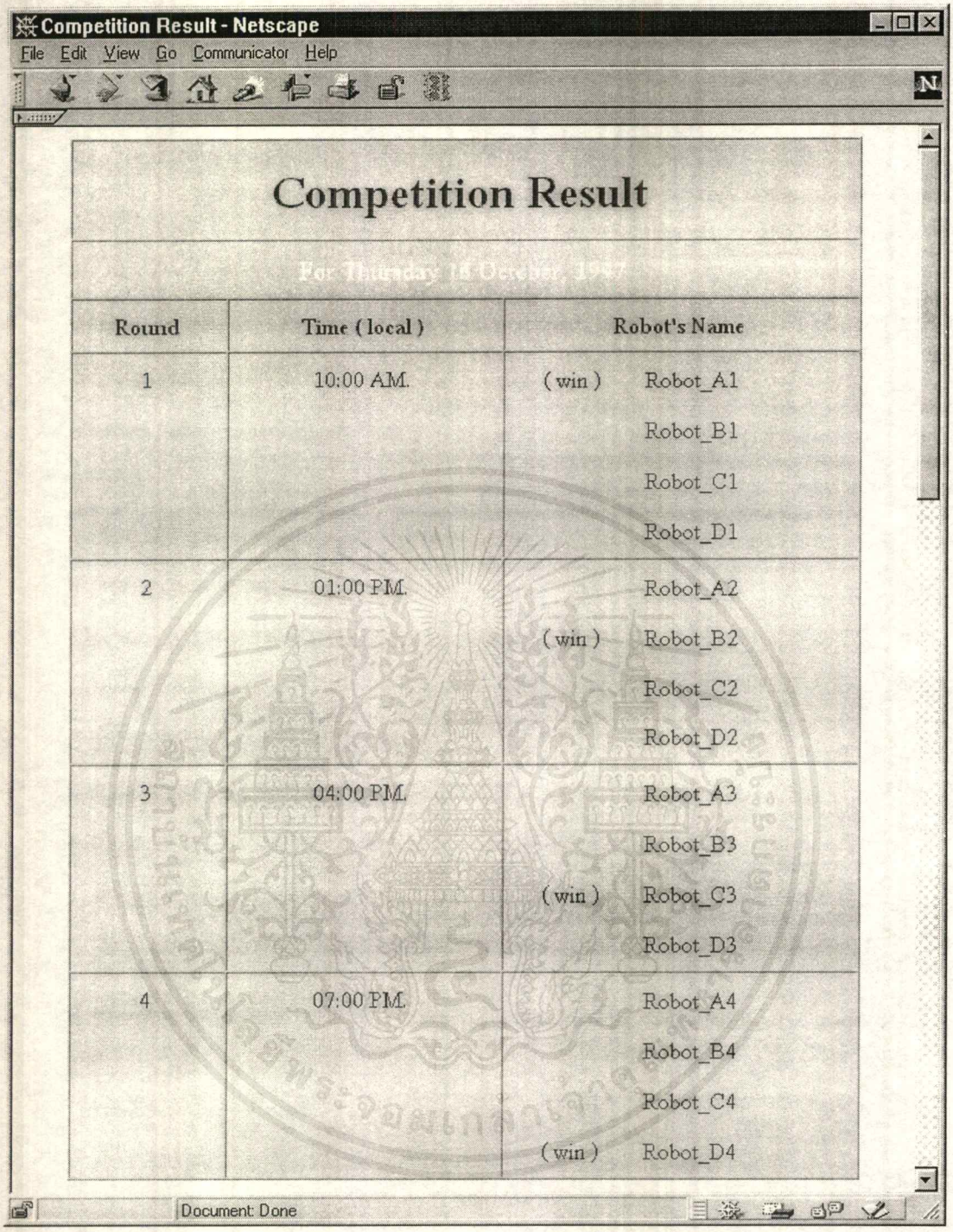
มีลักษณะของหน้าจอดังนี้



รูปที่ 5.10 หน้าจอแสดงการจำลองการแข่งขัน

5.2.4 ส่วนแสดงผลการแข่งขัน

ส่วนนี้เป็นส่วนที่แสดงผลการแข่งขันที่ได้แข่งขันผ่านไปแล้ว โดยมีลักษณะดังรูปที่ 5.9



รูป 5.11 หน้าจอแสดงตารางผลการแข่งขัน

5.3 ฟังก์ชัน

แบ่งออกเป็น 2 ส่วน ดังนี้

5.3.1Math functions

ฟังก์ชัน	ความหมาย
Int **getDistance**(int X1, int Y1, int X2, int Y2)	แจ้งระยะทางระหว่าง (X1, Y1) และ (X1, Y1)
Int **getDistance**(Point P1, Point P2)	แจ้งระยะทางระหว่าง จุก P1 กับ จุด P2
Int **abs**(int X)	แจ้งค่าสมบูรณ์ของ X
Double **sqrt(double** X)	แจ้งค่ารากที่สองของ X
Double **cosine(int** degree)	แจ้งค่า Cos ของมุม ในหน่วยองศา
Double **sine(int** degree)	แจ้งค่า Sine ของมุม ในหน่วยองศา
Int **arctan**(int dy , int dx) • dy: ระยะในแกน y • dx: ระยะในแกน x	แจ้งค่า arc-tangent ของ slope ในรูปขององศา (degree) ค่าที่ให้ ตั้งแต่ 0 - 359

ตารางที่ 5.1 ฟังก์ชันทางคณิต

5.3.2 Robot functions

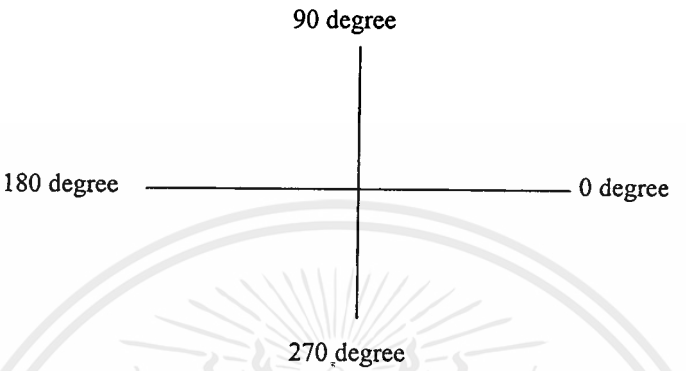
ฟังก์ชัน	ความหมาย
Void **move (int** direction, int speed)	ให้หุ่นเคลื่อนที่ไปในทิศทาง 'direction' ด้วยความเร็ว 'speed' หน่วย/วินาที speed >= 0
Boolean **scan (int** direction, int arc)	วิเคราะห์ทิศทางเพื่อหาเป้าหมายในทิศทาง direction - arc จนถึง direction + arc ถ้าเจอเป้าหมายจะแจ้งระยะทางที่ใกล้ที่สุด ถ้าเจอ จะให้ค่า true และไม่เจอ จะให้ค่า false
Void **fire(int** direction)	การยิงกระสุนในทิศทาง direction
Point **getPosition**( )	แจ้งจุดตำแหน่งของหุ่น (จะให้ค่า X , Y)
Int **getX**( )	แจ้งตำแหน่ง X ปัจจุบันของหุ่นยนต์
Int **getY**( )	แจ้งตำแหน่ง Y ปัจจุบันของหุ่นยนต์
Int **getHP**( )	แจ้งจุดที่ถูกโจมตีในปัจจุบันของหุ่นยนต์
Int **getSpeed**( )	แจ้งความเร็วปัจจุบันของหุ่นยนต์
Int **getDirection**( )	แจ้งทิศทางปัจจุบันของหุ่นยนต์

ตารางที่ 5.2 ฟังก์ชันทางการคอนโทรลหุ่นยนต์

5.4 กติกาการแข่งขัน

5.4.1 ทั่วไป (General)

หน่วยของมุมเป็น องศา (degree) และมีทิศทางดังรูปที่ 14 มุมทั้งหมดที่เกิน 0องศา หรือมากกว่า 360 องศา จะทำการปรับมุมโดยอัตโนมัติ ให้อยู่ระหว่าง 0 ถึง 359 องศา แต่สนามที่แข่งขันบนหน้าจอ ทิศทางของแกน Y จะมีทิศทางตรงข้ามกันที่ 90 องศา ของรูปที่ 12 จะมีลักษณะสลับที่กัน



รูปที่ 5.12 ทิศทางในสนามการแข่งขัน

5.4.2 หุ่นยนต์

- ขนาดของหุ่นยนต์มีขนาด 29×29 หน่วย (unit)
- ความเร็วมากที่สุดในการเคลื่อนที่ของหุ่นยนต์ เท่ากับ 10 หน่วย/วินาที
- ความเร่งของการเคลื่อนที่หุ่นยนต์ เท่ากับ $\pm$ 3 หน่วย^2/วินาที
- ความเร็วในการหมุน เท่ากับ 5 หน่วย/วินาที ถ้าความเร็วเกิน 5 หน่วย/วินาที มันจะทำให้ ลดลงจนถึง 5 หน่วย/วินาที โดยอัตโนมัติ
- มุมมากที่สุดที่หุ่นยนต์จะหมุนได้ใน 1 วินาที คือ 15 องศา (degree)
- ระยะที่แสกน (scan) ได้เท่ากับ 200 หน่วย

5.4.3 การระเบิด

หุ่นยนต์สามารถยิงกระสุนได้ 2 ลูก ในเวลาเดียว หมายความว่า สามารถยิงกระสุนให้ได้ 2 ลูกในอากาศพร้อมกัน แต่ถ้าคุณพยายามยิงมากกว่า 2 คุณจะได้รับค่า 'false' กลับมา จากฟังก์ชัน fire( )
ความเร็วของกระสุน มีความเร็วเท่ากับ 40 หน่วย/วินาที

5.4.4 ขอบเขตสนามแข่ง

- ขนาดของสนามการแข่งขันมีขนาด 558×450 ตารางหน่วย
- ตำแหน่ง (0,0) จะเป็นตำแหน่งของมุมล่างซ้ายของสนามรบ โดยที่เคยกล่าวแล้ว ดังนั้นตำแหน่ง (0,0) ที่มุมบนซ้ายบนจอ

- ในการแข่งแต่ละครั้ง ตำแหน่งของหุ่นแต่ละตัวจะถูกตัวการสุ่มตำแหน่งให้อยู่ในที่ต่างๆ
- ในการแข่งขันจะใช้เวลา 300 วินาที

5.5 อัลกอริทึมของโปรแกรม

แบ่งออกเป็นส่วนๆ ได้ 4 ส่วน ดังนี้

5.5.1 ส่วนรับ Source Code

หน้าที่หลักของส่วนรับ Source Code คือการ Upload file จากเครื่องของผู้เล่น (ไคลเอนต์) มาไว้ที่เครื่อง Server และต้องสามารถตรวจสอบดูว่า Source Code ที่รับเข้ามานั้นสามารถนำไปรับได้อย่างไม่มีข้อผิดพลาด และทำการบันทึกชื่อของหุ่นและรายละเอียดต่างๆไว้ใน Queue เพื่อให้ส่วนสร้างตารางสามารถนำรายชื่อหุ่นเหล่านั้นไปทำการสร้างตารางการแข่งขันได้

วิธีการที่ง่ายและสะดวกที่สุดในการตรวจสอบว่า Source Code ที่เรารับเข้ามานั้นมี error หรือไม่ก็โดยการลองทำการ คอมไพล์โค้ด นั้นดูถ้าทำการ คอมไพล์ แล้วมี error message รายงานออกมา ก็แสดงว่า Source Code นั้นยังมีข้อผิดพลาดอยู่และนำมาใช้ไม่ได้ แต่ถ้าหาก คอมไพล์ แล้วไม่พบ error message ใดๆ เกิดขึ้น เราก็มั่นใจได้ในระดับหนึ่งว่า Source Code นั้นไม่น่าจะมีปัญหา จะมีข้อผิดพลาดได้ก็น่าจะเกิดจาก อัลกอริทึม ที่ผิดพลาดซึ่งไม่น่าจะก่อให้เกิดความเสียหายมากนัก

จากที่กล่าวมาแล้วทั้งหมดนั้นเป็นเพียงความสามารถหลักๆ ของส่วนรับ Source Code เท่านั้น นอกจากความสามารถหลักๆ เหล่านั้นแล้ว ส่วนรับ Source Code อาจจะมีความสามารถอื่นๆ เพิ่มเติมขึ้นมาได้ เช่น อาจมี text editor เพื่อใช้สำหรับแก้ไข Source Code ในกรณีที่ คอมไพล์ แล้วพบว่ามีความผิดพลาดเกิดขึ้น เป็นต้น

เนื่องจากหน้าที่การทำงานของส่วนนี้ที่ตรงไปตรงมาไม่ค่อยซับซ้อน และมีหน้าที่ไม่มากนัก ดังนั้นเราจะทำการเขียน โปรแกรม ซีจีไอ ขึ้นมาตัวหนึ่งเพื่อทำหน้าที่ในส่วนของการรับ Source Code นี้ โดยอัลกอริทึมในการทำงานของโปรแกรมนี้จะเป็นดังนี้

1. ทำการ Upload Source Code ของผู้เล่น จากเครื่องของผู้เล่น (ไคลแอนต์) ไปยัง เซิร์ฟเวอร์
2. ทำการคอมไพล์ Source Code รับเข้ามาและตรวจสอบดูว่ามี error เกิดขึ้นในระหว่างการ คอมไพล์ หรือไม่
3. ถ้ามี error เกิดขึ้น ต้องแสดง error message เหล่านั้นให้ผู้เล่นทราบด้วยว่าเกิด error ที่ตรงไหนบ้าง เพื่อที่ผู้เล่นจะได้ แก้ไขข้อผิดพลาดได้
4. ถ้าหากไม่มีข้อผิดพลาดใดๆเกิดขึ้น ก็จะทำการ Save Source Code นั้นและเพิ่มชื่อของหุ่นตัวนี้เข้าไปใน queue ของหุ่นที่ส่งเข้ามา
5. รายงานผลให้ผู้เล่นทราบว่าหุ่นของเขาได้ถูกนำไปเก็บไว้ใน queue เพื่อรอการจัดรอบการแข่งขันในกรณีที่หุ่นนั้นผ่านการคอมไพล์ โดยไม่มีข้อผิดพลาด และรายงานข้อผิดพลาดใน Source Code ให้ผู้เล่นทราบ ในกรณีที่คอมไพล์ แล้วเกิดข้อผิดพลาดขึ้น

5.3.2 ส่วนสร้างตารางการแข่งขัน

ส่วนนี้มีหน้าที่ในการสร้างตาราง การแข่งขันขึ้นมา โดยนำรายชื่อหุ่นใน queue ซึ่งสร้าง โดยส่วนรับ Source Code มาทำการสร้างรอบการแข่งขันขึ้น เนื่องจากจำนวนหุ่นที่ส่งเข้ามามีปริมาณที่ไม่ แน่นอน ดังนั่นในการสร้างตารางการแข่งขันนี้ จึงต้องทำการตรวจสอบหลายๆ ครั้ง อย่างเช่น ในการแข่งขันแต่ละรอบจะมีหุ่นที่เข้าทำการแข่งขันกัน 4 ตัว ซึ่งถ้าหาก Source Code ที่รับเข้ามามีไม่ถึง 4 ตัว รอบการแข่งขันนั้นก็ควรจะว่างลง คือ ไม่มีการแข่งขันกัน เนื่องจากเราไม่ทราบปริมาณที่หุ่นถูกส่งเข้ามา ดังนั้นเราจึงกำหนดลงไปเลยว่าในแต่ละวันจะมีการแข่งขัน 3 รอบ โดยที่แต่ละรอบจะมีหุ่นเข้าแข่งขันกัน 4 ตัว ดังนั้นเมื่อถึงเวลาสร้างตารางการแข่งขันส่วนสร้างตารางการแข่งขันก็จะอ่านลงชื่อหุ่นที่อยู่ใน queue ออกมา (อาจเป็นแบบ Random หรือ แบบ Sequential ก็ได้) ครั้งละ 4 ตัว เพื่อจัดเป็นรอบการแข่งขัน 1 รอบ และจัด 3 รอบ สำหรับแต่ละวัน แต่ถ้าหากจำนวนหุ่นที่อยู่ใน queue มีไม่ถึง 4 ตัว รอบการแข่งขันรอบนั้นก็ต้องว่างไปเพราะมีจำนวนหุ่นไม่ครบ และเมื่อจัดรอบการแข่งขันเสร็จแล้วก็ทำการ Save ตารางการแข่งขันนั้นไว้ สำหรับให้ผู้เล่นเข้ามาดูได้

เนื่องจากส่วนนี้ เป็นส่วนที่ไม่ต้องมีการติดต่อกับผู้เล่นเลย และเป็นโปรแกรมที่ต้องทำการรันทุกวัน วันละหนึ่งครั้งเพื่อสร้างตาราง ดังนั้นเราจึงสามารถเขียน โปรแกรม ด้วยภาษาใดๆ ก็ได้เพื่อมาทำหน้าที่นี้

อัลกอริทึมในการทำงาน

1. อ่านข้อมูลรายชื่อของหุ่นทั้งหมด จาก queue
2. ตรวจสอบจำนวนหุ่นที่อยู่ใน queue ว่ามีจำนวนมากกว่า หรือเท่ากับ 4 ตัว หรือไม่
3. ถ้ามีจำนวนหุ่นใน queue น้อยกว่า 4 ตัว สร้างตารางการแข่งขันโดยรอบการแข่งขันรอบนั้นไม่มีการแข่งขัน
4. ถ้ามีจำนวนหุ่นถึง 4 ตัว นำรายชื่อหุ่นทั้ง 4 ตัว ออกจาก queue (ลบรายชื่อออกจาก queue) แล้วนำไปสร้างรอบการแข่งขันในตารางการแข่งขันโดยใช้รายชื่อหุ่นทั้ง 4 ตัวนั้น
5. วนทำ ข้อ 2 - 4 จนได้รอบการแข่งขันครบ 3 รอบ
6. Save ตารางการแข่งขันที่สร้างขึ้นมา

5.5.3 ส่วน Simulation (จำลองการต่อสู้)

ส่วนนี้เป็นส่วนที่สำคัญที่สุดของระบบ และก็เป็นส่วนที่ซับซ้อนที่สุดอีกด้วย เนื่องจากในการต่อสู้กันแต่ละรอบนั้นจะมีหุ่นยนต์เข้าต่อสู้กันทั้งหมด 4 ตัว ซึ่งขณะแข่งขันหุ่นแต่ละตัวก็จะเคลื่อนไหวและต่อสู้ไปตามโปรแกรมของมันเองที่ผู้เล่นได้ทำการโปรแกรมไว้ก่อนแล้ว เพื่อให้การเคลื่อนไหวของหุ่นเป็นธรรมชาติและเท่าเทียมกันมากที่สุด เราจะใช้ความสามารถของภาษา Java ซึ่งสามารถสร้าง Thread ขึ้นมาทำงานและควบคุมการทำงานของ thread เหล่านั้นได้ด้วย โดยเราจะให้หุ่นแต่ละตัวเป็น thread 1 thread ซึ่งมีการทำงานเป็นอิสระต่อกัน และเนื่องจากแต่ละ thread ทำงานเป็นอิสระต่อกันนี้เอง ทำให้เราต้องมี program ส่วนกลางที่คอยติดต่อกับ thread แต่ละ thread และต้องมีเนื้อที่ส่วนกลางไว้คอยเก็บข้อมูล

ถานะการทำงานของ thread ต่างๆ เพื่อให้เราสามารถติดตามได้ว่า ขณะนี้หุ่นเคลื่อนที่ไปทางใด ขณะนี้อยู่ที่ตำแหน่งใดบ้าง เป็นต้น นอกจากนี้ยังต้องคอยส่งข้อมูลออกไปแสดงผลที่จอของผู้ชมอีกด้วย

หน้าที่ต่างๆ ของส่วน Simulation คือ

1. สร้างพื้นที่เก็บข้อมูลสำหรับให้แต่ละ thread ใช้เพื่อให้ thread แต่ละ thread สามารถติดต่อกับโปรแกรมที่ทำการ Simulation ได้
2. สร้าง thread สำหรับหุ่นทุกตัวและเริ่มการทำงานของ thread ทุกๆตัว โดยจัดให้ทุกๆ thread มี priority เท่ากัน เพื่อให้แต่ละ thread ได้รับ CPU time เท่าๆกัน
3. ทำการส่งข้อมูลสถานะของหุ่นแต่ละตัวไปแสดงผลให้ผู้ชมได้ชม
4. คอยตรวจสอบดูสถานะของหุ่นแต่ละตัวว่าเป็นอย่างไรบ้าง เช่น ตำแหน่งปัจจุบันของหุ่นแต่ละตัว , มีหุ่นตัวไหนโดนยิงบ้างหรือไม่ และมีหุ่นตัวไหนตายบ้างหรือไม่ เป็นต้น แล้วทำการติดต่อกับ thread แต่ละตัวตามสถานะของมันเช่น เมื่อมีหุ่นถูกยิง ส่วนนี้จะต้องส่ง message ไปบอกหุ่นตัวที่ถูกยิงว่า มันถูกยิง หรือกรณีที่มีหุ่นตาย ส่วนนี้ก็ต้องหยุดการทำงานของมันเป็นต้น
5. คอยตรวจสอบสถานะของเกม ว่าขณะนี้เกมการแข่งขันดำเนินไปเป็นอย่างไรบ้าง เช่นเกมการแข่งขันสิ้นสุดลงแล้ว และได้ผู้ชนะ หรือ เกมการแข่งขันจบลงแล้วแต่ไม่ได้ผู้ชนะเนื่องจากหุ่นทุกตัวถูกฆ่าตายหมด เป็นต้น
6. เมื่อการแข่งขันสิ้นสุด ส่วนนี้ต้องส่งข้อมูลผลการแข่งขันไปให้กับส่วนสร้างตารางผลการแข่งขันเพื่อทำการสร้างตารางผลการแข่งขัน เพื่อรายงานผลการแข่งขันให้กับผู้เล่นและผู้สนใจได้ทราบ

3.5.4 ส่วนสร้างตารางผลการแข่งขัน

เมื่อการแข่งขันแต่ละรอบเสร็จสิ้นลงไปแล้ว ก่อนที่โปรแกรม Simulation จะจบการทำงาน มันจะต้องไปเรียกโปรแกรมของส่วนสร้างตารางผลการแข่งขันขึ้นมาทำงานเสียก่อน และต้องส่งข้อมูลต่างๆ ไปให้ส่วนสร้างตารางผลการแข่งขันด้วย เช่น รายชื่อหุ่นทั้ง 4 ตัว ที่ทำการแข่งขันในรอบนั้น , ชื่อหุ่นที่ชนะในรอบนั้น เป็นต้น เมื่อส่วนสร้างตารางผลการแข่งขันถูกเรียกให้ขึ้นมาทำงานแล้วมันจะใช้ข้อมูลต่างๆที่ได้รับเข้ามาเพื่อนำไปสร้างตารางผลการแข่งขันของรอบนั้นๆ

และเหมือนกับส่วนสร้างตารางการแข่งขัน ส่วนสร้างตารางผลการแข่งขันไม่มีส่วนที่ต้องติดต่อกับผู้เล่น ดังนั้นเราจึงสามารถใช้ภาษาใดๆ ก็ได้ในการเขียนโปรแกรมเพื่อมาทำงานในส่วนนี้ได้

อัลกอริทึม ในการทำงาน

1. อ่านข้อมูลของรอบการแข่งขันนั้น , รายชื่อหุ่นที่เข้าแข่งขัน , ผู้ชนะ ฯลฯ
2. นำข้อมูลเหล่านั้นมาจัดสร้างตารางผลการแข่งขัน
3. จัดเก็บ (Save) ไฟล์ตารางผลการแข่งขัน เพื่อให้ผู้เล่นและผู้สนใจอื่นๆ เข้ามาดูได้

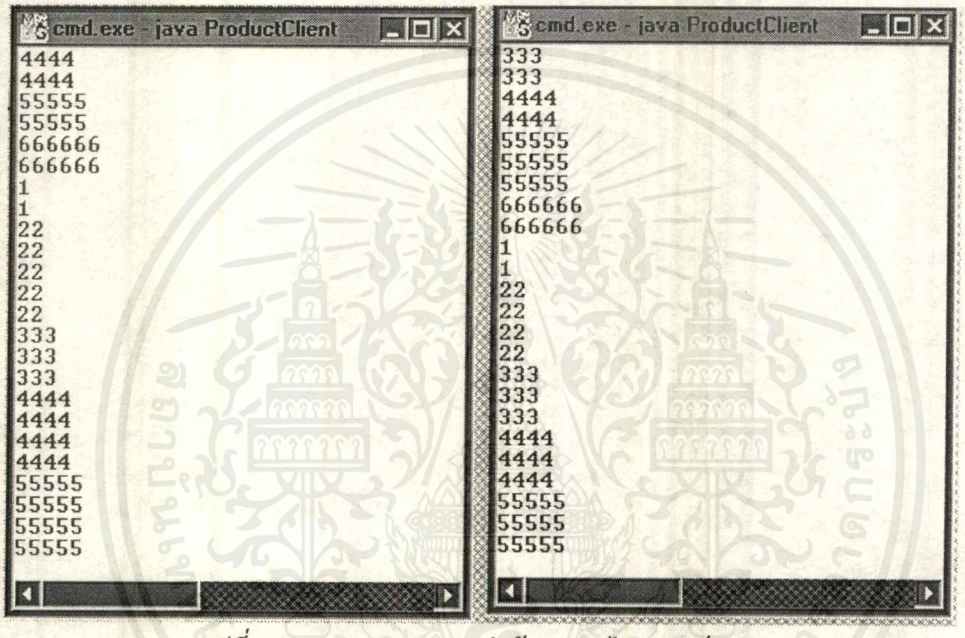
บทที่ 6

การทดลองและผลการทดลอง

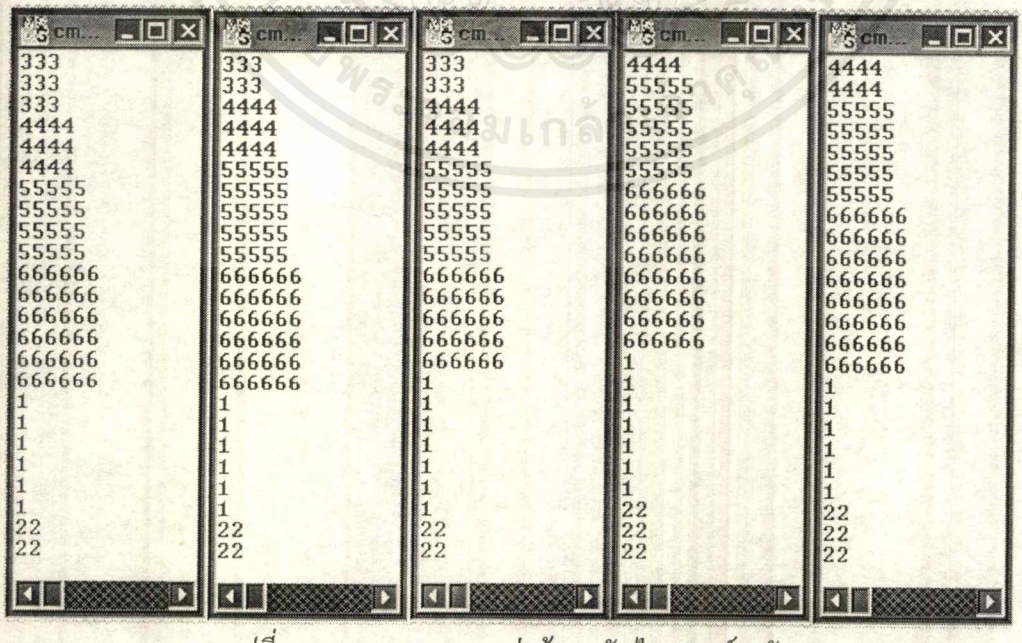
6.1 การทดลองการส่งข้อมูลระหว่างเซิร์ฟเวอร์กับไคลเอนต์

โปรแกรมทดสอบส่วนนี้ จะทดสอบเพื่อดูข้อมูลจากฝั่งไคลเอนต์ที่มีการร้องขอมากกว่า 1 ตัว โดยดูจากผลที่แสดงว่ามีการแสดงของข้อมูลเหมือนกันหรือไม่ หรือมีผลต่างของความเร็วขนาดไหน

การทดลองจะเห็นว่า ยังมีการส่งข้อมูลที่ยังไม่สมบูรณ์พอ ที่เป็นเช่นนี้อาจเป็นเพราะว่ามีไคลเอนต์มาก ถ้าเปรียบเทียบกับการทดสอบระหว่างไคลเอนต์ 2 ตัว กับ ไคลเอนต์ 5 ตัว



รูปที่ 6.1 การทดสอบการส่งข้อมูลกับไคลเอนต์ 2 ตัว



รูปที่ 6.2 การทดสอบการส่งข้อมูลกับไคลเอนต์ 5 ตัว

6.2 การทดลองส่วนของโปรแกรมหุ่น

6.2.1 การทดสอบโปรแกรมการเคลื่อนที่ของ หุ่นยนต์

เป็นการทดสอบหุ่นยนต์ให้เคลื่อนที่ตามโปรแกรมคอนโทรลที่ได้โปรแกรมไว้ว่าสามารถเคลื่อนที่ได้ตามต้องการหรือไม่

การทดสอบสามารถที่ จะทำได้ ตามที่ต้องการ มีลักษณะดังรูป

รูปที่ 6.3 แสดงหน้าจอการเคลื่อนที่ของหุ่นยนต์

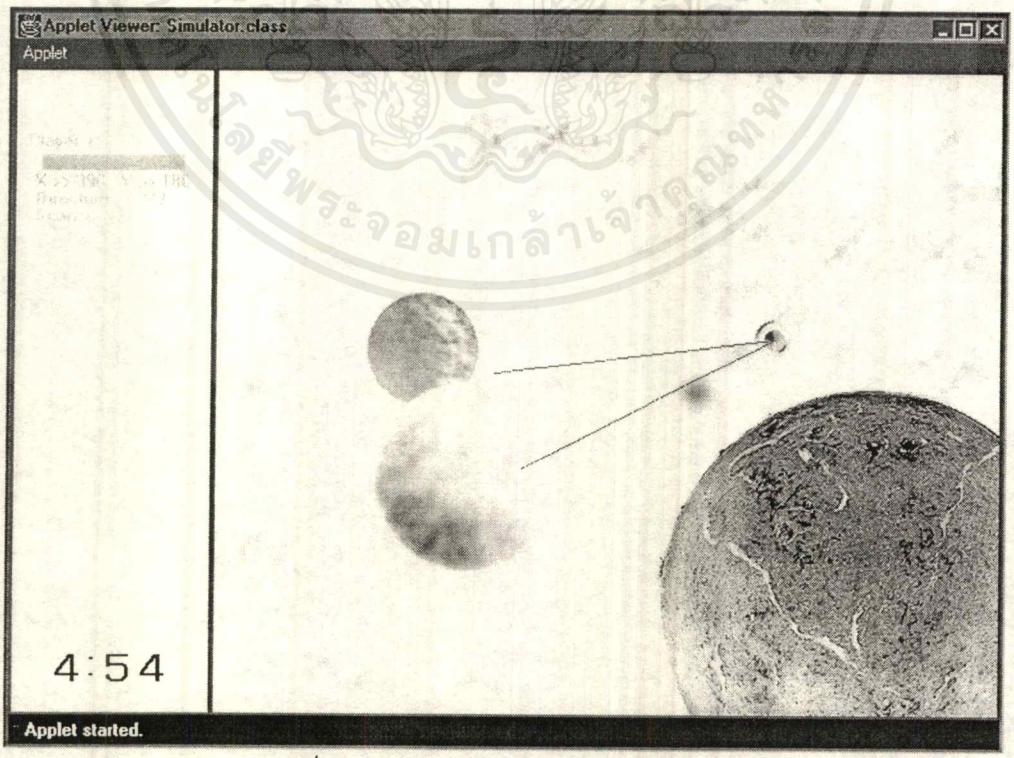
6.2.2 การทดสอบโปรแกรมการยิงกระสุนของหุ่นยนต์

เป็นการทดสอบต่อจากการเคลื่อนที่ของหุ่นยนต์เพื่อทกสอบการยิง ในทิศทางต่างๆ ที่โปรแกรมคอนโทรลได้ โปรแกรมเอาไว้ มีผลการแสดงดังรูป

รูปที่ 6.4 แสดงหน้าจอการยิงของหุ่นยนต์

6.2.3 การทดสอบโปรแกรมการ Scan หาตำแหน่งของคู่ต่อสู้

เป็นการทดสอบในส่วนการหาตำแหน่งของคู่ต่อสู้ ว่าได้รับค่าตำแหน่งถูกต้องหรือไม่ เพื่อใช้ในการหาตำแหน่งและทำลายคู่ต่อสู้



รูปที่ 6.5 แสดงหน้าจอการ scan ของหุ่นยนต์

บทที่ 7
วิจารณ์ และสรุป

จุดประสงค์ของโครงงานนี้คือ การเขียนโปรแกรม Application ที่ปฏิบัติการบนระบบเครือข่ายอินเตอร์เน็ต โดยจุดประสงค์ต้องการให้ผู้เล่นสามารถที่จะเล่นจากที่ไหนก็ได้ และจะสามารถดูการแข่งขันได้แบบเรียลไทม์ ซึ่งในการทำโครงงานนี้ ได้ศึกษาในส่วนของ การเขียนโปรแกรมแอพพลิเคชั่น บนระบบเครือข่ายอินเตอร์เน็ต และส่วนของการส่งขอมูลระหว่าง เซิร์ฟเวอร์ กับ ไคลเอนต์ โดยใช้ RMI ซึ่งเป็น class หนึ่งในภาษาจาวาโดยการศึกษาได้ลองส่ง ข้อความต่างๆ และเปรียบเทียบการแสดงผลจากหน้าจอว่ามีลักษณะเหมือนกันหรือไม่ มีความช้าเร็วขนาดไหนในการส่งข้อมูล ซึ่งผลที่ได้ออกมามีลักษณะที่ว่า เมื่อ ไคลเอนต์ ทำการร้องขอข้อมูลจาก เซิร์ฟเวอร์ และเมื่อเซิร์ฟเวอร์ได้รับการร้องขอก็จะส่งข้อมูลขณะนั้นไปให้ที่ ไคลเอนต์ และเมื่อลองใช้ ไคลเอนต์หลายตัวร้องขอข้อมูล ซึ่งมีผลทำให้การส่งข้อมูลมีความเร็วลดลง และถ้าเพิ่ม ไคลเอนต์ให้มีจำนวนมากขึ้น มีผลทำให้การส่งข้อมูลหยุดชั่วขณะหนึ่ง

ส่วนของการศึกษาการทำงานของโปรแกรมทำงานพัฒนาจากเขียนโปรแกรมให้ทำงานได้ในระดับ local โดยแบ่งออกเป็นส่วนย่อยๆ โดยมีลักษณะเป็นเธด และสะดวกในการทดสอบ เริ่มพัฒนาโปรแกรมโดยเขียนส่วนการเคลื่อนที่ของหุ่น และทำการทดสอบหุ่นว่าสามารถเคลื่อนที่ได้ตามที่โปรแกรมเขียนไว้หรือไม่ และจากนั้นก็พัฒนาต่อให้สามารถที่จะยิง และเช็คความถูกต้องของสถานะปัจจุบันของหุ่นยนต์ เมื่อถูกต้องแล้วก็พัฒนาให้หุ่นยนต์สามารถที่จะค้นหาตำแหน่งของคู่ต่อสู้โดยมีรัศมีความกว้าง และระยะในการค้นหา ซึ่งปัญหาในส่วนนี้คือการส่งค่าตำแหน่งของคู่ต่อสู้ยังไม่ถูกต้อง ซึ่งก็ได้แก้ไขเรียบร้อยแล้ว สุดท้ายจึงทำการแบ่งโปรแกรมหลักให้เป็น 2 ส่วน โดยโปรแกรมส่วนแรก ไว้ที่ฝั่งเซิร์ฟเวอร์ เพื่อใช้ทำการ จำลองการต่อสู้ และโปรแกรมส่วนที่สองไว้ที่ฝั่งไคลเอนต์โดยใช้ RMI ในการส่งค่าร้องขอ และรับข้อมูลไปแสดงการแข่งขันที่ หน้าจอโดยลักษณะแบบ เรียลไทม์ นอกจากโปรแกรมหลัก ในส่วนอื่นๆ ได้นำเอา หลักการทำงานของ CGI มาใช้ เพื่อช่วยจัดการ และดึงข้อมูลมาประมวลผลและแสดงผลในรูปแบบตาราง

การพัฒนาโปรแกรมได้พัฒนามาถึงในระดับ Local ซึ่งสามารถจะเล่นได้จริง และสามารถดูผ่านระบบเครือข่าย อินเตอร์เน็ต แต่การดูได้แบบเรียลไทม์ ยังไม่สามารถพัฒนาโปรแกรมให้ทำได้ เนื่องจากโปรแกรมที่พัฒนามีข้อมูลเป็นจำนวนมากที่ต้องใช้ส่งระหว่าง เซิร์ฟเวอร์ และ ไคลเอนต์ เพราะจากการทดสอบการส่งข้อมูล ใช้ข้อมูลที่มีขนาดไม่มากนักยังไม่มีความสามารถพอในการส่ง เมื่อมี ไคลเอนต์จำนวนมากๆ ดังนั้นในการพัฒนาโปรแกรมจึงยังไม่สามารถใช้งานเรียลไทม์ได้

และหวังว่าจะนำเอาโปรแกรมไปใช้ในการพัฒนาการเขียนโปรแกรมภาษาจาวา ให้กับผู้สนใจที่อยากลองเขียนภาษาจาวา และคาดว่าน่าจะทำการจัดการแข่งขันขึ้น และทำให้เป็นการแข่งขันประจำ เพื่อส่งเสริมให้คนหันมาสนใจในการพัฒนาโปรแกรม และชื่อเสียงของคณะ และสถาบันฯ

แนวทางในการพัฒนาโปรแกรม ภาษาจาวายังมีวิธีการอีกหลายอย่าง ที่ใช้ในการส่งข้อมูล อย่างเช่น Socket ซึ่งอาจจะนำไปสู่การทำงานแบบเรียลไทม์ได้ และภาษาจาวายังมีการพัฒนาเพิ่มขึ้นเรื่อยๆ ซึ่ง

อาจจะมี การพัฒนาในส่วนการส่งข้อมูลที่มีประสิทธิภาพมากกว่านี้ แต่สำหรับในปัจจุบันจากการสังเกตโปรแกรมต่างๆ ของจาวา ยังเป็นโปรแกรมที่ใช้พัฒนาในระดับ Local อยู่ ไม่ว่าจะเป็นเกมส์ หรือโปรแกรมแอพพลิเคชั่นต่างๆ

ผลจากการพัฒนาโปรแกรมนี้ พอจะเห็นแนวทางในการพัฒนาโปรแกรมที่ใช้ ภาษาจาวา ได้ว่าเหมาะสมกับงานประเภทฐานข้อมูล หรือ โปรแกรมแอพพลิเคชั่นที่ไม่มีการเปลี่ยนแปลงข้อมูลตลอดเวลา ซึ่งถ้านำโปรแกรมจาวามาใช้กับงานที่มีการเปลี่ยนแปลงข้อมูลตลอดเวลาจะมีลักษณะความเร็วที่ช้ามาก ในการพัฒนาโปรแกรมต่างจำควรจะพิจารณาจากจุดประสงค์ และความสามารถของโปรแกรมที่จะใช้ว่าเหมาะสมพอหรือไม่ที่จะพัฒนา

ภาคผนวก A

CRobots

CRobots เป็นเกมพื้นฐานในการเขียนโปรแกรมบนคอมพิวเตอร์ ซึ่งแตกต่างจาก เกมอาร์เขต (arcade) ที่ต้องใช้คนมาควบคุมการกระทำของมัน แผนการทั้งหมดใน CRobots ต้องสมบูรณ์ก่อนการแข่งขันในที่แท้จริง เกมการต่อสู้นี้ถูกออกแบบ และเขียนด้วยภาษาซี ผู้เขียนจะโปรแกรมหุ่นยนต์ของแต่ละคน เพื่อทำการค้นหา , ต่อสู้ และทำลายหุ่นยนต์ตัวอื่นๆ แต่ละโปรแกรมที่ทำการรัน จะแตกต่างกัน หุ่นยนต์แต่ละตัวจะมีอุปกรณ์เท่ากันและแข่งขันครั้งละ 4 ตัว CRobots เป็นการเล่นที่ดีที่สุดระหว่างบุคคลหลายๆ บุคคล แต่ละคนจะสร้างหุ่นยนต์ของแต่ละคนให้ดีเยี่ยมและสมบูรณ์

CRobots ประกอบด้วย ซี คอมไพล์เลอร์ , Virtual Computer และ จอแสดงการต่อสู้ (สนามรบ) (monochrome หรือ color) CROBOTS มี function ช่วยในการเขียน มีการแสวงหาตำแหน่งของคู่ต่อสู้ , การเคลื่อนตำแหน่งและหยุด การยิงกระสุนและ function อื่นๆ หลังจากโปรแกรมทำการ compile และ load แยก robot แต่ละตัวแล้ว เฝ้าดูการต่อสู้ Robot เคลื่อนที่ , ยิงกระสุน และระเบิดจะมีการแสดงสถานะของแต่ละตัวด้วย

อุปกรณ์และโปรแกรมที่ต้องใช้

- ไอบีเอ็ม - พีซี
- แรม 192 k
- DOS 2.0 หรือ สูงกว่า
- disk drive
- Monochrome or Color/Graphics display
- Text editor

ลักษณะการ running CRobots

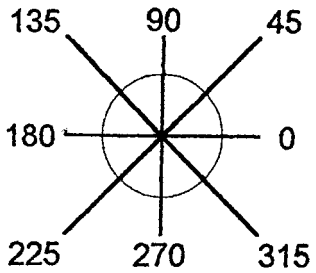
เริ่มจาก DOS PROMPT CRobots มีลักษณะคำสั่งดังนี้

A:>= CRobots [options] robot-program-1 [robot-program-n] [>file]

ขอบเขตของ CROBOTS

สนามรบมีขนาด กว้าง 1000 เมตร ยาว 1000 เมตร มีกำแพงล้อมรอบเพื่อว่า robot จะเคลื่อนที่อยู่ภายในกำแพง โดยกำหนดมุมล่างซ้าย มีจุดพิกัด (Coordinates) x = 0 , y = 0 และมุมบนขวามีค่าจุดพิกัด x = 999 , y = 999 กำหนดทิศ โดย

ทิศตะวันออก (ขวา)	เป็น	0	องศา
ทิศเหนือ	เป็น	90	องศา
ทิศตะวันตก	เป็น	180	องศา
ทิศใต้	เป็น	270	องศา



รูปที่ A1 ทิศทางในการเคลื่อนที่หุ่นยนต์

การโจมตีของหุ่นยนต์

อาวุธหลักในการโจมตี คือ ปืน และสแกนเนอร์ ปืนนั้นมีรัศมีในการยิง 700 เมตร สำหรับกระสุนของปืนไม่มีจำนวนจำกัด แต่การยิงแต่ละครั้ง จะยิงได้เพียงครั้งละ 1 ลูกเท่านั้น

สแกนเนอร์ เป็นอุปกรณ์แสงที่ใช้ในการหาเป้าหมายโดยหัวหมุนได้ 360 องศา ความแน่นอนจะใช้ได้ +10 , -10 องศา ทำให้สามารหาตำแหน่ง ของคู่ต่อสู้ไอย่างรวดเร็ว

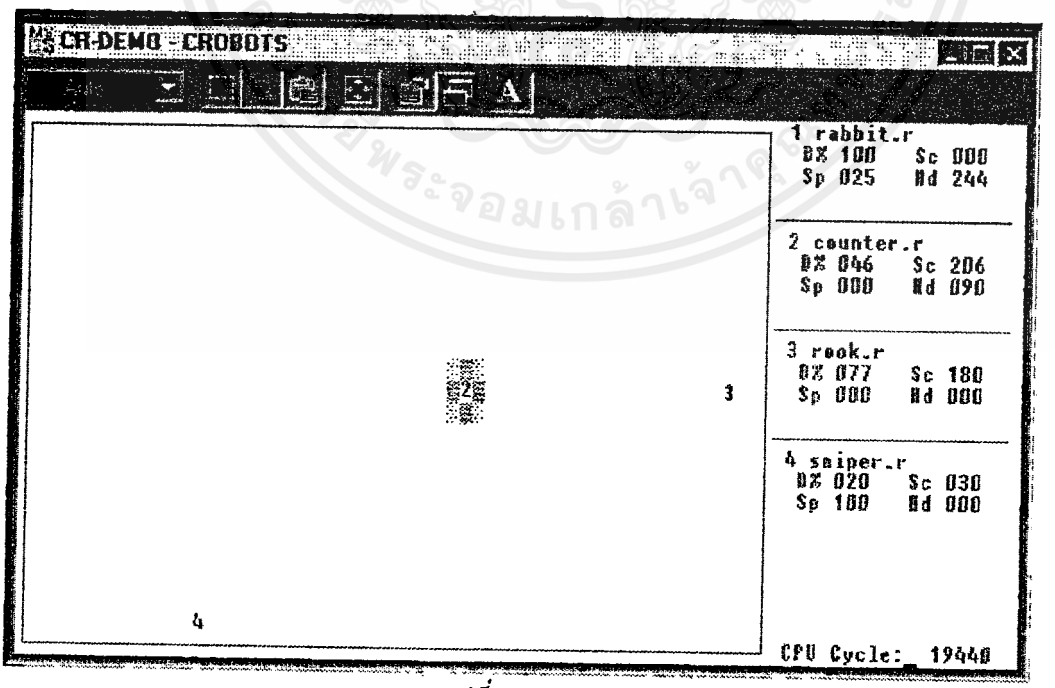
การป้องกัน

การป้องกันของหุ่นยนต์ อาศัยการทำงาน 2 ส่วน คือ

ส่วนขับเคลื่อน (Motor Drive) จะทำให้สามารถหมุนทิศทางเพื่อหาเป้าหมายได้ 0-360 องศา มี speed ในการหมุน 0-100% ของ เพาว์เวอร์

ส่วนแสดงสถานะ (Status Register) เป็นตัวส่งค่ากลับมายังหุ่นยนต์ บอกตำแหน่งของหุ่นยนต์

หน้าจอ แสดง การแข่งขัน



รูปที่ A2 User Interface

เอกสารอ้างอิง

Karanjit S.Siyan, PH. D. & James L. Weaver : " **Inside Java** " , New Riders Publishing, Indianapolis, Indiana

Michael Morrison : " **Teach Yourself INERNET GAME PROGRAMMING WITH JAVA in 21 Days** ", Samsnet , Indianapolis , Indiana

Merlin and Conrad Hughes , Michael Shoffner , Maria Winslow : " **JAVA Network Programing** " , Manning , Greenwich , CT,Printed in the United States of America

Joel Fan , Eric Ries , Calin Tenitchi : " **Black Art of JAVA Game Programming** " , 1996 , The waite Group , Inc . , Cort Madera

Neil Bartlett , Alex leslie , and Steve Simkin : " **JAVA Programming Explorer** " , 1996 , The Coriolis Group , Inc .

Daniel Groner , K.C. Hopson , Harish Prabandham , Todd Sundsted : " **JAVA Language API Superbible** " , 1996 , Waite Group , Inc .

Ken Arnold , James Gosling : " **The JAVA™ Programming Language** " , Addison- Wesley Publishing Company , Inc.

Gary Cornell , Cay S. Horstmann : " **CORE JAVA** " second edition , 1997 , Sun Microsystem , Inc. , Sunsoft Press A Prentice Hall Title

Clayton Walnum : " **JAVA by Example** " , 1996 , Que® Corporation

กิตติกรรมประกาศ

ขอขอบพระคุณ

บิดามารดา ที่ส่งเสียเล่าเรียน และเป็นกำลังใจให้

อาจารย์ธนา หงษ์สุวรรณ ที่ค่อยให้คำปรึกษา และคำแนะนำ

อาจารย์ภาควิชาวิศวคอมพิวเตอร์ทุกท่าน ที่ประสิทธิ์ประสาทวิชาความรู้ให้

พี่ธานิณ ที่ช่วยแนะนำ และให้คำปรึกษา ในส่วนภาษาจาวา

เพื่อนๆ น้องๆ ที่ช่วยทำรายงาน