การติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

WEB-BASED NETWORK MANAGEMENT INTERFACE using VRML

แดง ชลีพรพิมลรัตน์
DANG CHOLPRIPIMONRAT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาเทคโนโลยีสารสนเทศ
บัณฑิตวิทยาลัย
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
พ.ศ. 2543

ISBN 974-622-854-4

การติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

WEB-BASED NETWORK MANAGEMENT INTERFACE using VRML

แดง ชลไพรพิมลรัตน์

DANG CHOLPRIPIMONRAT

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต

สาขาวิชาเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2543

ISBN 974-622-854-4

WEB-BASED NETWORK MANAGEMENT INTERFACE using VRML

DANG CHOLPRIPIMONRAT

A THESIS SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY

SCHOOL OF GRADUATE STUDIES

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2000

ISBN 974-622-854-4

บัณฑิตวิทยาลัย

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองวิทยานิพนธ์

หัวข้อวิทยานิพนธ์ การติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

WEB-BASED NETWORK MANAGEMENT INTERFACE USING VRML

ชื่อนักศึกษา นางสาวแดง ชลไพรพิมลรัตน์

รหัสประจำตัว 40067006

ปริญญา วิทยาศาสตรมหาบัณฑิต

สาขาวิชา เทคโนโลยีสารสนเทศ

อาจารย์ผู้ควบคุมวิทยานิพนธ์ อาจารย์อัครินทร์ คุณกิตติ

คณะกรรมการสอบวิทยานิพนธ์		ลายมือชื่อ
อาจารย์อัครินทร์	คุณกิตติ	
รศ.ดร.รัตติกร	วิรากูลศิริพันธุ์	
ดร.นพพร	โชติกุกำธร	
ดร.รัฐการ	อภิวัฒน์วาจ่า	
ดร.จันทร์บูรณ์	สถิตวิริยวงศ์	

วัน/เดือน/ปี ที่สอบ 1 มิถุนายน 2543 **เวลา** 10.00 น. เป็นต้นไป

สถานที่สอบ ณ ห้องบรรยายชั้น 2 คณะเทคโนโลยีสารสนเทศ

บัณฑิตวิทยาลัยรับรองแล้ว

(รศ.ดร.มานะ สงวรศิลป์)

คณบดีบัณฑิตวิทยาลัย

วันที่ 5 **เดือน** มิถุนายน **พ.ศ.** 2543

หัวข้อวิทยานิพนธ์	การติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML
นักศึกษา	นางสาวแดง ชลไพรพิมลรัตน์
รหัสประจำตัว	40067006
ปริญญา	วิทยาศาสตรมหาบัณฑิต
สาขาวิชา	เทคโนโลยีสารสนเทศ
พ.ศ.	2543
อาจารย์ผู้ควบคุมวิทยานิพนธ์	อาจารย์อัครินทร์ คุณกิตติ

บทคัดย่อ

ระบบการติดต่อผู้ใช้ในการจัดการระบบเครือข่ายผ่านเว็บด้วย VRML อยู่บนพื้นฐานการติดต่อผ่านเครือข่ายอินเตอร์เน็ตที่ผสมผสานความสามารถทางด้าน CGI กับ VRML เข้าด้วยกัน โดยจะเป็นการนำเสนอการจัดการเครือข่ายผ่านเว็บด้วยส่วนติดต่อผู้ใช้ที่เป็น 3 มิติ โดยการนำเสนอข้อมูลในหน้าจอการทำงานของ VRML Browser ในแต่ละ view มีโครงสร้างเป็นลำดับชั้น โดยในแต่ละออบเจ็กต์จะมี viewpoint เป็นตัวกำหนดมุมมองในแต่ละ view ว่าต้องการดูข้อมูลลึกลงไปกี่ระดับ พร้อมทั้งการวิเคราะห์และผลการทดลองประกอบการวิจัยที่แสดงให้เห็นว่าจำนวนออบเจ็กต์และคุณสมบัติเพิ่มเติมที่ใส่ให้กับออบเจ็กต์ในแต่ละ view มีผลกับเวลาในการประมวลผลทั้งเวลาในการโหลดข้อมูลจากเว็บเซิร์ฟเวอร์มายังเว็บเบราเซอร์และเวลาในการท่องโลกสามมิติของผู้ใช้ ทั้งนี้ย่อมขึ้นกับประสิทธิภาพของระบบที่ใช้ด้วย เช่น เว็บเซิร์ฟเวอร์, เว็บเบราเซอร์, ปลั๊กอิน VRML Browser และระบบจัดการฐานข้อมูล และเพื่อเป็นการลดเวลาที่ต้องใช้ในการประมวลผลจึงได้มีการแบ่งกลุ่มของออบเจ็กต์ออกเป็น 2 กลุ่ม คือ ออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายและออบเจ็กต์ที่เป็นสภาพแวดล้อมในการจัดการเครือข่าย เพื่อเป็นตัวกำหนดอัตราส่วนในการนำเสนอว่ามีความเหมือนจริงมากน้อยเพียงใดเพื่อแลกกับเวลาในการประมวลผลที่ดีขึ้น ซึ่งปัจจัยเหล่านี้จะเป็นสิ่งที่ผู้ใช้สามารถนำไปพิจารณาเพื่อให้เกิดประสิทธิภาพการทำงานของระบบที่ดีขึ้น

Thesis Title	Web-Based Network Management Interface using VRML
Student	Miss Dang Cholpripimonrat
Student ID.	40067006
Degree	Master of Science
Programme	Information Technology
Year	2000
Thesis Advisor	Mr. Akharin Khunkitti

ABSTRACT

Web-Based Network Management Interface using VRML is based on the Internet which blends capability of CGI and VRML together. This research presents network management with three-dimension interface. The presentation of network management information focuses on the operation of VRML Browser. Each view has hierarchical structure and each object has viewpoint which can display own subsystem. The analysis and experimental result in the research show that the number of object and its appended attribute in each view affect time performance; data-loading time from Web Server to Web Browser and user's usage time in 3-D Internet world navigation, also depending on system performance such as Web Browser, Web Server, VRML Browser plug-in and database management system. To reduce computational time, object has to be divided into 2 groups, networked devices and network environment. Both groups can specify the proportion of realistic performance for trade-off between time and realistic one. These factors are useful to user's consideration for better performance system.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้อย่างดี เพราะได้รับความเมตตากรุณา คำแนะนำและคำปรึกษาจากอาจารย์อัครินทร์ คุณกิตติ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ผู้วิจัยรู้สึกซาบซึ้งในความอนุเคราะห์จากท่านและขอกราบขอบพระคุณเป็นอย่างสูง

นอกจากนั้นผู้วิจัยขอขอบคุณและเอ่ยนามถึงบุคคลต่างๆ ที่ได้มีส่วนร่วมช่วยในการสร้างงานวิจัยชิ้นนี้ขึ้นมาจนสำเร็จ

ขอกราบขอบพระคุณอาจารย์ประจำคณะเทคโนโลยีสารสนเทศ เป็นอย่างสูงที่เสียสละเวลาอันมีค่าให้ข้อคิดแนวทางในบางจุดที่ผู้วิจัยติดปัญหาบางอย่าง ซึ่งมีส่วนช่วยทำให้ผู้วิจัยเข้าใจในปัญหานั้น

เจ้าหน้าที่คณะเทคโนโลยีสารสนเทศ ซึ่งให้ความอนุเคราะห์ในการอำนวยความสะดวกต่างๆ ในการทำงานตั้งแต่ต้น รวมถึง พี่ๆ เพื่อนๆ นักศึกษาทุกคนที่ช่วยเหลือให้คำแนะนำต่างๆ พร้อมทั้งช่วยตรวจเทียบและแก้ไขทฤษฎีและอื่นๆ ที่ผิดพลาด จนสำเร็จสมบูรณ์ยิ่งขึ้นและยังให้กำลังใจต่อผู้วิจัยอย่างใกล้ชิดตลอดมา

Microsoft Co.,Ltd. ที่ได้จัดสร้างและพัฒนาระบบปฏิบัติการไมโครซอฟท์วินโดวส์และระบบฐานข้อมูลขึ้นมาให้ผู้ใช้งานคอมพิวเตอร์ทั้งหลายได้รับความสะดวกสบายจากการใช้งานคอมพิวเตอร์

Borland Co.,Ltd. ที่ได้สร้างโปรแกรม Delphi ซึ่งเป็นคอมไพเลอร์ภาษาปาสคาลสำหรับสร้างโปรแกรมที่ทำงานภายใต้สภาวะแวดล้อมของไมโครซอฟท์วินโดวส์ ให้ผู้พัฒนาโปรแกรมได้ใช้เขียนโปรแกรมเพื่อพัฒนาวงการคอมพิวเตอร์ให้ดีขึ้น

PLATINUM Technology, Inc. ที่ได้สร้างโปรแกรมกราฟฟิกทูล 3 มิติ ได้แก่ Cosmo World และปลั๊กอิน 3 มิติ ได้แก่ Cosmo Player เพื่อใช้ในการพัฒนาโปรแกรมภาษา VRML

บิดา, มารดา และครอบครัวของผู้วิจัย ที่ได้ให้การสนับสนุนทั้งทางด้านกำลังใจ และทางการเงิน

สุดท้ายขอขอบคุณบัณฑิตวิทยาลัย สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง และมูลนิธิเพื่อการศึกษาคอมพิวเตอร์และการสื่อสาร (C&C EDUCATION FOUNDATION) ที่ได้ให้ทุนสนับสนุนการทำวิทยานิพนธ์ครั้งนี้

คุณค่าและประโยชน์อันพึงมีจากวิทยานิพนธ์ฉบับนี้ ผู้วิจัยขอมอบแด่ผู้มีพระคุณทุกท่าน

แดง ชลไพรพิมลรัตน์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การจัดการระบบเครือข่ายนับได้ว่าเป็นงานที่สำคัญอย่างยิ่ง ไม่ว่าจะเป็นการควบคุม วางแผนหรือการเฝ้าติดตามกิจกรรมที่เกิดขึ้นกับเครือข่าย โดยที่สามารถเข้าไปจัดการเครือข่ายได้ไม่ว่าผู้บริหารเครือข่ายจะอยู่ที่ใดๆ ก็ตาม ซึ่งการจัดการระบบเครือข่ายทางอินเตอร์เน็ตนับได้ว่าเป็นส่วนหนึ่งในการใช้งานที่สำคัญในปัจจุบัน เวิลด์ไวด์เว็บจึงกำเนิดขึ้นในฐานะรูปแบบใหม่ในการเข้าถึงและแสดงข้อมูลผ่านอินเตอร์เน็ต เนื่องจากความง่ายของการสร้างข้อมูลสารสนเทศและการเชื่อมโยงเข้ากับแหล่งข้อมูลที่สัมพันธ์กันผ่านเว็บเบราเซอร์ อันเป็นเครื่องมือในการติดต่อสื่อสารที่มีประสิทธิภาพ เพื่อความสะดวกและรวดเร็วในการควบคุมเครือข่าย [4] แต่เนื่องด้วยแอปพลิเคชั่นที่ใช้ในการจัดการเครือข่ายมีอินเตอร์เฟซที่เป็นลักษณะ 2 มิติ ทำให้ผู้บริหารเครือข่ายไม่มีความสะดวกในการที่จะเข้าไปจัดการเครือข่ายเพราะไม่อาจรู้ตำแหน่งที่ตั้งของอุปกรณ์เครือข่ายที่เกิดปัญหานั้น ทำให้ต้องเสียเวลาในการหาตำแหน่งที่ตั้งของอุปกรณ์ จากข้อมูลที่กล่าวมาทั้งหมด คือ ปัญหาที่ต้องหาทางแก้ไขเพื่อทำให้ระบบจัดการเครือข่ายมีประสิทธิภาพมากขึ้น

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

ความมุ่งหมายและวัตถุประสงค์ของการศึกษางานวิทยานิพนธ์นี้ เพื่อลดข้อจำกัดทางการจัดการเครือข่าย ซึ่งแต่เดิมการจัดการเครือข่ายนำเสนอในรูปแบบ 2 มิติ จึงเปลี่ยนรูปแบบการนำเสนอให้เหมือนกับโลกของความเป็นจริงมากขึ้นในรูปแบบที่เป็น 3 มิติ นอกจากนั้นยังเป็นการศึกษาเพื่อพัฒนารูปแบบการนำเสนอข้อมูล 3 มิติ โดยอาศัยข้อมูลที่ได้จัดเก็บจากความเป็นจริง เช่น ตำแหน่งที่ตั้งหรือขนาดของอุปกรณ์เครือข่ายและอื่นๆ เพื่อนำมาใช้ในการจัดวางที่อิงกับตำแหน่งจริงทางกายภาพด้วย และนำผลลัพธ์ที่ได้จากการทดสอบมาวิเคราะห์หาผลกระทบ ข้อดีข้อเสียและขนาดของระบบเครือข่ายที่เหมาะสมต่อการใช้ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

1.3 สมมติฐานของการศึกษา

ในงานวิจัยนี้จะนำเสนอผลการพัฒนาและทดสอบโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML (Virtual Reality Modeling Language) อันเป็นระบบจัดการเครือข่ายที่ผสมผสานความสามารถทางด้าน CGI (Common Gateway Interface) กับ VRML เข้าด้วยกัน โดยจะเป็นการนำเสนอการจัดการเครือข่ายผ่านเว็บด้วยส่วนติดต่อผู้ใช้ที่เป็น 3 มิติ ซึ่ง

สามารถที่จะนำเสนอรายละเอียดของข้อมูลที่เป็นประโยชน์ในการจัดการเครือข่าย ในเรื่องของตำแหน่งที่ตั้งจริงทางกายภาพของอุปกรณ์ได้เป็นอย่างดี รวมถึงสามารถตรวจดูสถานะปัจจุบันของระบบเครือข่ายและสภาพแวดล้อมทางกายภาพได้ อันจะเป็นหนทางไปสู่การจัดการระบบเครือข่ายที่มีประสิทธิภาพ

1.4 ทฤษฎีหรือแนวคิดที่ใช้ในการวิจัย

แนวความคิด คือ การปรับปรุงรูปแบบในการนำเสนอการจัดการเครือข่ายเป็นหลัก เพื่อพัฒนารูปแบบในการนำเสนอ โดยอาศัยทฤษฎีต่างๆ ซึ่งการจัดการโครงสร้างในการนำเสนอกราฟฟิกแบบ 3 มิติผ่านเว็บได้มีการนำภาษา VRML มาใช้ในการสร้างภาพ 3 มิติ และอีกทั้ง VRML ยังสามารถเรียกดูผ่านเว็บเบราเซอร์ได้ด้วย แต่เนื่องจาก VRML เป็นการนำเสนอข้อมูลแบบสแตติก จึงได้มีการนำเทคโนโลยี CGI มาใช้เพื่อจัดการกับ VRML ให้มีคุณสมบัติในการนำเสนอข้อมูลแบบไดนามิก นอกจากนั้นยังนำความสามารถของภาษาจาวามาช่วยในการโต้ตอบระหว่างผู้ใช้กับระบบด้วย

1.5 ขอบเขตของการศึกษา

ขอบเขตของวิทยานิพนธ์นี้ คือ การพัฒนาและทดสอบโปรแกรมการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML โดยมีการพัฒนาอินเตอร์เฟซในการจัดการในรูปแบบ 3 มิติ ที่อิงกับตำแหน่งจริงทางกายภาพ สุดท้ายนำผลลัพธ์ที่ได้จากการทดสอบมาวิเคราะห์หาผลกระทบ ข้อดีข้อเสียและขนาดของระบบเครือข่ายที่เหมาะสมต่อการใช้ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

1.6 ขั้นตอนของการศึกษา

ขั้นตอนของการศึกษาเริ่มด้วยการรวบรวมข้อมูลทางฟิสิกคอลของวัตถุและสภาพแวดล้อมอื่นๆ ที่เกี่ยวข้องให้อยู่ในรูปแบบของวัตถุที่เหมือนจริงหรือใกล้เคียงความจริงทั้งด้านรูปร่างลักษณะและตำแหน่งการจัดวางรวมทั้งสถานะต่างๆ ของวัตถุ ซึ่งแปรผันไปตามข้อมูลลอจิคอล เพื่อให้ผู้ที่มีหน้าที่ในการดูแลระบบเครือข่ายสามารถทราบถึงตำแหน่งที่อยู่และสถานะของอุปกรณ์ได้ง่ายและสามารถเข้าไปดำเนินการกับอุปกรณ์ต่างๆ ได้รวดเร็วและถูกต้องมากขึ้น โดยจัดเก็บลงในฐานข้อมูล จากนั้นทำการพัฒนา CGI Script เพื่อดึงข้อมูลจากฐานข้อมูลทางฟิสิกคอลและลอจิคอลในการนำมาสร้างเป็นออบเจ็กต์ VRML แต่ละออบเจ็กต์ เมื่อทำการพัฒนาสคริปต์ต่างๆ เรียบร้อยแล้วจึงทำการทดลองใช้งาน โดยนำสคริปต์ที่พัฒนาขึ้นมาได้เก็บในเว็บเซิร์ฟเวอร์ เพื่อให้ผู้ใช้ทำการร้องขอการทำงานที่ต้องการผ่านเว็บเบราเซอร์ โดยการระบุ URL ไปยังเว็บเซิร์ฟเวอร์ เมื่อ

สคริปต์ดังกล่าวถูกประมวลผลจะส่งผลลัพธ์ที่ได้กลับไปให้เว็บเซิร์ฟเวอร์แล้ว เว็บเซิร์ฟเวอร์จะส่งผลลัพธ์ดังกล่าวกลับไปให้กับเว็บเบราเซอร์ที่ทำการร้องขอมาในรูปแบบของ VRML

1.7 ข้อตกลงเบื้องต้น

ความเหมือนจริงในการนำเสนอออบเจ็กต์ต่างๆ นั้น ขึ้นอยู่กับโปรแกรมว่าสามารถกำหนดเงื่อนไขในการนำเสนอได้มากน้อยเพียงใด ดังนั้นถ้ามีการเก็บข้อมูลทางฟิสิกคอลและลอจิคอลลงในฐานข้อมูลละเอียดมากเท่าไร ความเหมือนจริงก็จะมีมากขึ้นแต่เวลาที่ใช้ในการนำเสนอก็จะช้าลงไปด้วย [15]

1.8 ข้อจำกัดของการศึกษา

เนื่องจากการพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML อาจจะมีข้อจำกัดในด้านของเวลาที่เครื่องไคลเอนต์จะต้องใช้ในการประมวลผลกราฟฟิก 3 มิติ ที่ได้รับ VRML Script จากการประมวลผลของฝั่งเว็บเซิร์ฟเวอร์ เนื่องด้วย ถ้า VRML Script ที่ได้รับประกอบด้วยออบเจ็กต์จำนวนมากแล้ว เวลาที่ VRML Browser ต้องใช้ในการประมวลผลก็จะมากขึ้นเช่นกัน [15] แต่เมื่อประสิทธิภาพของฮาร์ดแวร์และการประมวลผลทางกราฟฟิก 3 มิติ มีประสิทธิภาพมากขึ้น จะทำให้สามารถเพิ่มเติมองค์ประกอบต่างๆ เข้าไปในระบบจัดการเครือข่ายผ่านเว็บในรูปแบบ 3 มิติได้อย่างเต็มที่และยังทำให้มีความเสมือนจริงมากขึ้น และในการสร้างวัตถุต่างๆ โดยใช้โปรแกรมกราฟฟิกที่สามารถบันทึกไฟล์เป็นไฟล์ VRML ได้ ยังขาดเครื่องมือที่มีประสิทธิภาพ ไฟล์ของวัตถุที่สร้างได้มีขนาดจำนวนไบต์ที่ใหญ่เกินไป เนื่องจากมีการใช้ตัวเลขทศนิยมหลายตำแหน่งเกินความจำเป็น ปัญหาเหล่านี้สามารถแก้ไขได้โดยนำไฟล์ที่ได้มาทำการแก้ไขปรับปรุงอีกที ซึ่งจะใช้เวลาพอสมควร แต่ในอนาคตโปรแกรมกราฟฟิกเหล่านี้ อาจมีเวอร์ชันใหม่ที่มีประสิทธิภาพออกมาใช้ก็ได้

1.9 รายละเอียดของเครื่องคอมพิวเตอร์และเครื่องมือที่ใช้ในการพัฒนาระบบ

การพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ต้องการเครื่องคอมพิวเตอร์และเครื่องมือที่ใช้ในการพัฒนาดังต่อไปนี้

- เซิร์ฟเวอร์

  คุณสมบัติด้านฮาร์ดแวร์

  CPU : Pentium 300

  RAM : 64 MB

  HardDisk : 6 GB

ระบบปฏิบัติการ (Operating System)

Windows NT 4.0

- ระบบเว็บเซิร์ฟเวอร์

  Microsoft Internet Information Server 4.0 (IIS 4.0)

- ระบบจัดการฐานข้อมูล (DBMS : Database Management System)

  SQL Server 7.0

- ไคลเอนต์

  คุณสมบัติด้านฮาร์ดแวร์

  CPU : Pentium 133

  RAM : 32 MB

  HardDisk : 1.2 GB

  ระบบปฏิบัติการ (Operating System)

  Windows 95

- เว็บเบราเซอร์

  Netscape Communicator 4.0 หรือ Internet Explorer 4.0 ที่มี VRML plug-in เช่น Cosmo Player 2.1

- เครื่องมือที่ใช้พัฒนา

  Borland Delphi 4.0

  โปรแกรมกราฟฟิกทูล 3 มิติ ได้แก่ Cosmo World

  เท็กซ์เอดิเตอร์ สำหรับเขียน VRML

1.10 รายละเอียดของแต่ละบท

สำหรับรายละเอียดในวิทยานิพนธ์ ซึ่งเป็นหลักการในการพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML มีรายละเอียดดังต่อไปนี้

บทที่ 2 เครือข่ายอินเตอร์เน็ตและภาษา VRML อธิบายทฤษฎีและหลักการทำงานการจัดการเครือข่ายผ่านเว็บ, เวิลด์ไวด์เว็บ, โปรโตคอล HTTP, TCP/IP, เว็บแอพพลิเคชัน, แนวคิดพื้นฐานของฐานข้อมูลเวิลด์ไวด์เว็บ, ส่วนประกอบของระบบฐานข้อมูลเวิลด์ไวด์เว็บ, ข้อดีของระบบฐานข้อมูลเวิลด์ไวด์เว็บ, การรักษาความปลอดภัย, CGI, API, VRML และภาษาจาวา

บทที่ 3 หลักการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML โดยจะกล่าวถึงแนวความคิดของระบบ, การทำงานของระบบ, ขั้นตอนการทำงานในการรับข่าวสารของไคลเอนต์และการลบ IP ของเซิร์ฟเวอร์และการออกแบบฐานข้อมูลของระบบ

บทที่ 4 การวิเคราะห์การทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML โดยจะกล่าวถึงการวิเคราะห์ขั้นตอนการทำงานของระบบ และบทสรุปการวิเคราะห์ขั้นตอนการทำงานหลักของระบบ

บทที่ 5 การทดลองและผลการทดลองของระบบ โดยจะกล่าวถึงผลการทดลองการใช้ระบบ, การทดลองและผลการทดลองปัจจัยที่มีผลกับการทำงานของระบบ, การทดลองและผลการทดลองเพื่อพิสูจน์บทวิเคราะห์การทำงานของระบบ และเปรียบเทียบผลการวิเคราะห์เชิงทฤษฎีกับผลการทดลอง

บทที่ 6 สรุปผลการวิจัยและข้อเสนอแนะของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML เพื่อเป็นประโยชน์ในการปรับปรุงและพัฒนางานวิจัยให้ดียิ่งขึ้น

ภาคผนวก ก. บทความที่ตีพิมพ์ในวารสาร

ภาคผนวก ข. วิธีการใช้งานโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

บทที่ 2
เครือข่ายอินเตอร์เน็ตและภาษา VRML

การจัดการข้อมูลและการแจกจ่ายข้อมูลมีความสำคัญอย่างสูงในปัจจุบัน ระบบฐานข้อมูลมีบทบาทอย่างมากในการรวบรวม, วิเคราะห์, จัดการและประมวลผล ในขณะที่ระบบอินเตอร์เน็ตเข้ามามีบทบาทในฐานะสื่อในการนำข้อมูลเผยแพร่ออกไปและระบบอินเตอร์เน็ตมีแนวโน้มว่าจะเป็นที่นิยมอย่างแพร่หลายทั้งในปัจจุบันและจะทวีความสำคัญขึ้นเป็นลำดับในอนาคตอันใกล้นี้ แอปพลิเคชั่นอย่างหนึ่งในระบบอินเตอร์เน็ตซึ่งกำลังเป็นที่นิยมใช้กันอย่างมากคือเวิลด์ไวด์เว็บ

เวิลด์ไวด์เว็บกำเนิดขึ้นในฐานะรูปแบบใหม่ในการเข้าถึงและแสดงข้อมูลผ่านอินเตอร์เน็ต เนื่องจากความง่ายของการสร้างข้อมูลสารสนเทศและการเชื่อมโยงเข้ากับแหล่งข้อมูลที่สัมพันธ์กันผ่านเว็บเบราเซอร์ จึงทำให้การจัดการเครือข่ายผ่านเว็บเป็นกลยุทธ์ที่จะทำให้ผู้บริหารเครือข่ายสามารถที่จะติดตามและเฝ้าดูเครือข่ายโดยใช้เว็บเบราเซอร์ ซึ่งเป็นเครื่องมือในการติดต่อสื่อสารที่มีประสิทธิภาพ เพื่อความสะดวกและรวดเร็วในการควบคุมเครือข่าย การจัดการเครือข่ายผ่านเว็บจึงเป็นวิวัฒนาการในการจัดการเครือข่าย [2] แต่เนื่องด้วยอินเตอร์เฟซของระบบจัดการเครือข่ายที่มีอยู่ในปัจจุบันนั้นยังคงเป็นลักษณะ 2 มิติ และไม่อิงกับตำแหน่งจริงทางกายภาพ ทำให้ยังไม่สามารถแก้ปัญหาของผู้บริหารเครือข่ายในเรื่องของการหาตำแหน่งที่ตั้งจริงๆ ของอุปกรณ์ ซึ่งปัญหาที่ตามมาก็คือ เมื่ออุปกรณ์เครือข่ายเกิดปัญหาขึ้นและมีความจำเป็นที่จะต้องเข้าไปทำการแก้ไข ณ จุดที่เกิดปัญหาสามารถทำได้ยาก เนื่องจากไม่ทราบตำแหน่งที่ตั้งจริงๆ ของอุปกรณ์ที่เกิดปัญหา ทำให้การแก้ปัญหาทำได้ล่าช้า อันเป็นวัตถุประสงค์ที่ต้องการนำเสนอในวิทยานิพนธ์ฉบับนี้

ดังนั้นในบทนี้จะได้กล่าวถึงทฤษฎีและหลักการทำงานที่เกี่ยวข้องที่ได้นำมาใช้พัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML เช่น หลักการทำงานของเวิลด์ไวด์เว็บ, โปรโตคอล TCP/IP, HTTP, เว็บแอปพลิเคชั่น, แนวคิดพื้นฐานของฐานข้อมูลเวิลด์ไวด์เว็บ, ส่วนประกอบของระบบฐานข้อมูลเวิลด์ไวด์เว็บ, ข้อดีของระบบฐานข้อมูลเวิลด์ไวด์เว็บและการรักษาความปลอดภัยของเครือข่ายอินเตอร์เน็ตเนื่องจากพื้นฐานของวิทยานิพนธ์นี้เป็นการทำงานบนเครือข่ายอินเตอร์เน็ต และจะได้กล่าวถึงการทำงานของ CGI และภาษา VRML อันเป็นเครื่องมือในการพัฒนาวิทยานิพนธ์นี้ขึ้นมาเพื่อให้สามารถจัดการเครือข่ายผ่านเว็บด้วยการทำงานแบบไดนามิกและเพื่อให้นำเสนอในรูปแบบกราฟฟิก 3 มิติผ่านเว็บได้และการนำภาษาจาวามาใช้เพื่อรองรับการโต้ตอบกันระหว่างผู้ใช้กับระบบ

2.1 การจัดการเครือข่ายผ่านเว็บ

คือ ระบบจัดการเครือข่ายผ่านอินเตอร์เน็ต ซึ่งประกอบไปด้วยการควบคุมระบบเครือข่าย และ/หรือ การรวบรวมข้อมูลและวิเคราะห์ข้อมูลผ่านเว็บ โดยระบบจัดการเครือข่ายผ่านเว็บนี้ยังคงมีหน้าที่เหมือนกับการจัดการระบบเครือข่ายแบบเดิม แต่เปลี่ยนมาใช้เทคโนโลยีเว็บแทน [4] โดยผู้ใช้ระบบจัดการเครือข่ายสามารถเข้าถึงระบบจัดการได้ทั้งแบบโลคอลและรีโมต โดยใช้เว็บเบราเซอร์ เช่น Netscape Navigator, Microsoft Internet Explorer ที่ใช้ได้กับหลากหลายเพล็ตฟอร์ม โดยระบบจัดการเครือข่ายผ่านเว็บยังสามารถที่จะรวบรวมหรือกระจายข้อมูลของเครือข่ายออกไปผ่านทางอินเตอร์เน็ตได้ด้วย

ดังนั้นระบบจัดการเครือข่ายผ่านเว็บจึงเกิดขึ้นเนื่องจากการเติบโตของเครือข่ายอินเตอร์เน็ตที่ใช้ TCP/IP และใช้โปรโตคอลที่สัมพันธ์กับ Hypertext Markup Language (HTML) ทำให้การติดต่อกับเซิร์ฟเวอร์ด้วยเว็บเบราเซอร์จากเพล็ตฟอร์มใดๆ สะดวก, ง่ายและราคาถูก และด้วยเครือข่ายอินเตอร์เน็ตที่เกิดขึ้นทำให้ผู้บริหารเครือข่ายค้นพบถึงประโยชน์อื่นๆ ของเว็บ ตัวอย่างเช่น เมื่อมองย้อนกลับไปเปรียบเทียบกับรูปแบบไคลเอนต์/เซิร์ฟเวอร์ จะเห็นได้ชัดว่าเทคโนโลยีด้านเว็บจะส่งเสริมการใช้เครือข่ายให้มีประสิทธิภาพมากที่สุดและลดเวลาในการพัฒนา, ลดเครื่องมือที่ใช้และลดค่าใช้จ่ายกว่ามาก เนื่องจากเว็บเบราเซอร์ต้องการเครื่องที่มีกำลังปานกลางกับเนื้อที่ของดิสก์ไม่มากนัก ซึ่งผู้บริหารเครือข่ายสามารถจะโยกย้ายการคำนวณและงานในการจัดเก็บไปยังเว็บเซิร์ฟเวอร์และยังทำให้การติดตั้งไคลเอนต์ง่ายขึ้นด้วย รวมถึงราคาของเพล็ตฟอร์มที่จะเข้าถึงเว็บเซิร์ฟเวอร์มีราคาถูกและด้วยรูปแบบ "Thin client/fat server" นี้เองจะช่วยลดค่าใช้จ่ายทางด้านฮาร์ดแวร์และยังทำให้ผู้ใช้มีความยืดหยุ่นมากขึ้นอีกด้วย [4]

2.1.1 ประโยชน์ของการจัดการเครือข่ายผ่านเว็บ

การจัดการเครือข่ายผ่านเว็บได้รวมเอาฟังก์ชันการทำงานต่างๆ ของเว็บและการจัดการเครือข่ายเข้าไว้ด้วยกัน เพื่อความสามารถที่สูงขึ้นจากเครื่องมือที่มีอยู่เดิม ทำให้ผู้บริหารเครือข่ายที่ใช้ระบบจัดการเครือข่ายผ่านเว็บสามารถที่จะเฝ้าดูและควบคุมเครือข่ายด้วยเว็บเบราเซอร์ ณ ที่ใดๆ ได้ ทำให้ไม่ผูกติดกับเพล็ตฟอร์มและยังสามารถขจัดปัญหาในเรื่องของ Interoperability ซึ่งมักจะเกิดขึ้นกับโครงสร้างที่เป็นมัลติเพล็ตฟอร์ม [10] นอกจากนั้นระบบจัดการเครือข่ายยังมีกราฟฟิกอินเตอร์เฟซทำให้นำเสนอข้อมูลสารสนเทศในรูปแบบของการมองเห็นได้มากขึ้นและการใช้งานที่ง่ายขึ้น เนื่องจากการทำงานของเว็บเบราเซอร์และอินเตอร์เฟซของเว็บเพจเป็นรูปแบบที่ใช้กันทั่วไปในเวิลด์ไวด์เว็บและด้วยเหตุผลนี้เองทำให้ลดค่าใช้จ่ายในการฝึกบุคลากรและเป็นการเพิ่มจำนวนผู้ใช้ในการใช้เครือข่ายอีกด้วย [4]

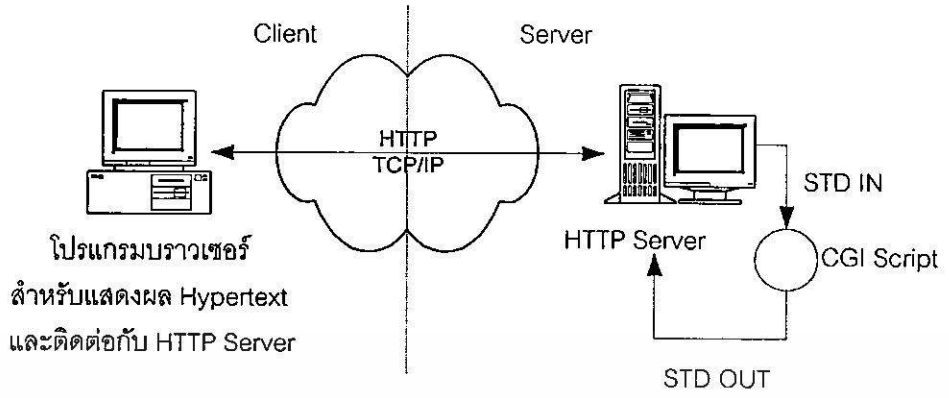
นอกจากนั้นระบบจัดการเครือข่ายผ่านเว็บยังเป็นความมุ่งหมายในอุดมคติอย่างหนึ่งของการกระจายข้อมูลสารสนเทศที่เกี่ยวข้องกับการปฏิบัติการบนเครือข่าย ดังตัวอย่างเช่น การติดต่อ

โดยตรงจากเว็บเบราเซอร์ไปยังอินเตอร์เน็ตเว็บไซต์ ซึ่งผู้ใช้สามารถเข้าถึงเครือข่ายและทำการอัพเดตข้อมูลได้เลย ด้วยเหตุนี้จึงเป็นการขจัดขั้นตอนในการที่จะต้องโทรไปขอความช่วยเหลือยัง Help desks ขององค์กรหรือเจ้าหน้าที่ MIS ยิ่งไปกว่านั้นระบบจัดการเครือข่ายผ่านเว็บต้องการเพียงการติดตั้งของเว็บเซิร์ฟเวอร์เท่านั้น ทำให้การรวมระบบจัดการเครือข่ายผ่านเว็บเข้าสู่อินเตอร์เน็ตเป็นงานที่รวดเร็วและไม่ยุ่งยากมากนัก [4]

2.2 เวิลด์ไวด์เว็บ

เครือข่ายอินเตอร์เน็ตทำงานอยู่บนสถาปัตยกรรมไคลเอนต์/เซิร์ฟเวอร์ ด้านเซิร์ฟเวอร์ประกอบด้วยโปรแกรมเว็บเซิร์ฟเวอร์เพื่อเป็นผู้ให้บริการข้อมูลต่างๆ แก่ผู้ใช้ ทางด้านไคลเอนต์ประกอบด้วยโปรแกรมเว็บเบราเซอร์เพื่อติดต่อกับโปรแกรมเว็บเซิร์ฟเวอร์ ทั้งเว็บเซิร์ฟเวอร์และโปรแกรมเว็บเบราเซอร์ติดต่อสื่อสารกันภายใต้โปรโตคอล HTTP [3]

สำหรับข้อจำกัดของลักษณะไคลเอนต์/เซิร์ฟเวอร์ จะอยู่ที่คุณภาพของระบบเครือข่ายว่าสามารถส่งผ่านข้อมูลได้มากน้อยเพียงใดและขนาดของการใช้งานเครือข่ายว่าคับคั่ง (Traffic) มากหรือไม่ การทำงานของเวิลด์ไวด์เว็บจะทำโดยมีการส่งรายการขอ (HTTP Request) ไปที่เครื่องที่ให้บริการหรือเครื่องที่มีรายการที่ต้องการ ซึ่งรายละเอียดที่ต้องการสามารถลงรายการได้โดยผ่านฟอร์มที่ใช้งานภายในโปรแกรมเว็บเบราเซอร์ โดยแบบฟอร์มที่ลงรายการจะอยู่ในรูปแบบภาษา HTML ซึ่งเป็นภาษาที่ใช้สำหรับบริการแบบเวิล์ดไวด์เว็บที่มีลักษณะการใช้งานเป็นไฮเปอร์เท็กซ์ โดยอาศัยรูปแบบตัวกำหนด (TAG) ในการกำหนดหน้าที่และส่วนต่างๆ ในหน้าเอกสาร [3] สำหรับการบริการแบบเวิลด์ไวด์เว็บ ส่วนที่ทำหน้าที่ติดต่อกับฐานข้อมูลภายนอกหรือโปรแกรมภายนอกที่ทำงานเฉพาะอย่าง เช่น การนำรายการที่กรอกในแบบฟอร์มไปค้นหารายละเอียดในฐานข้อมูลแล้วนำมาแสดงผล ส่วนที่ทำหน้าที่นี้คือ CGI ซึ่งทำหน้าที่เป็นส่วนที่กำหนดรูปแบบการติดต่อระหว่างบริการเวิลด์ไวด์เว็บกับโปรแกรมภายนอก โดยรับรายการจากแฟ้มข้อมูลของเว็บเพจ ดังแสดงในรูปที่ 2.1 [13]



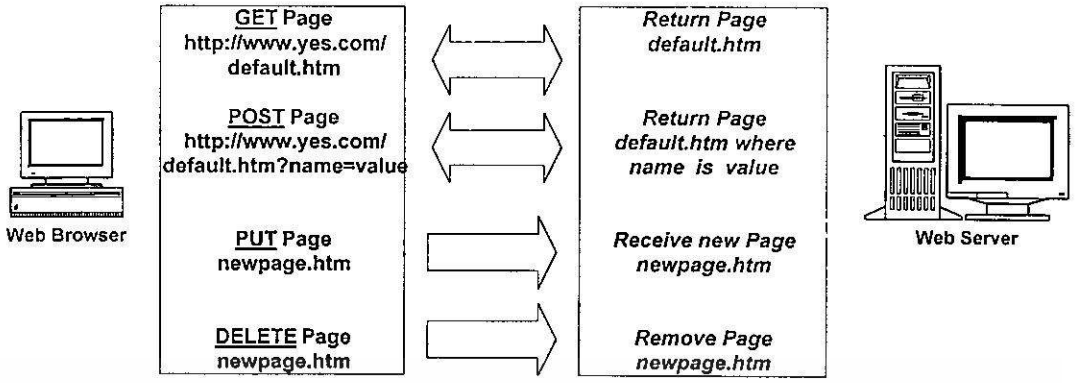
รูปที่ 2.1 แสดงโครงสร้างสถาปัตยกรรมของเครือข่ายอินเตอร์เน็ตที่ทำงานร่วมกับ CGI

2.3 โปรโตคอล HTTP

เครือข่ายอินเตอร์เน็ตใช้โปรโตคอลในการสื่อสารคือ TCP/IP [19] แต่ในระบบเวิลด์ไวด์เว็บนั้นโปรโตคอลที่ใช้เป็นมาตรฐานคือ HTTP (Hypertext Transfer Protocol) อันเป็นโปรโตคอลที่ทำงานอยู่เหนือ TCP/IP โดยโปรโตคอลจะทำการแปลงคำร้องขอจากเว็บเพจไปสู่รูปแบบคำร้องขอสำหรับส่งข้ามเครือข่าย โดยจะเป็นการนำเอาคำร้องขอจากเว็บเบราเซอร์ในรูปแบบของ Method ซึ่งจะมีดังนี้คือ GET, PUT, POST และ DELETE

- GET Method ทำหน้าที่ในการร้องขอไฟล์จากเว็บเซิร์ฟเวอร์
- POST Method ถูกใช้ในรูปแบบการส่งค่าพารามิเตอร์สู่เว็บเซิร์ฟเวอร์
- PUT Method อนุญาติให้มีการสร้างไฟล์ใหม่หรือเพิ่มเติมได้ในกรณีที่มีไฟล์เดิมอยู่แล้ว
- DELETE Method ใช้ในการลบไฟล์จากเว็บเซิร์ฟเวอร์

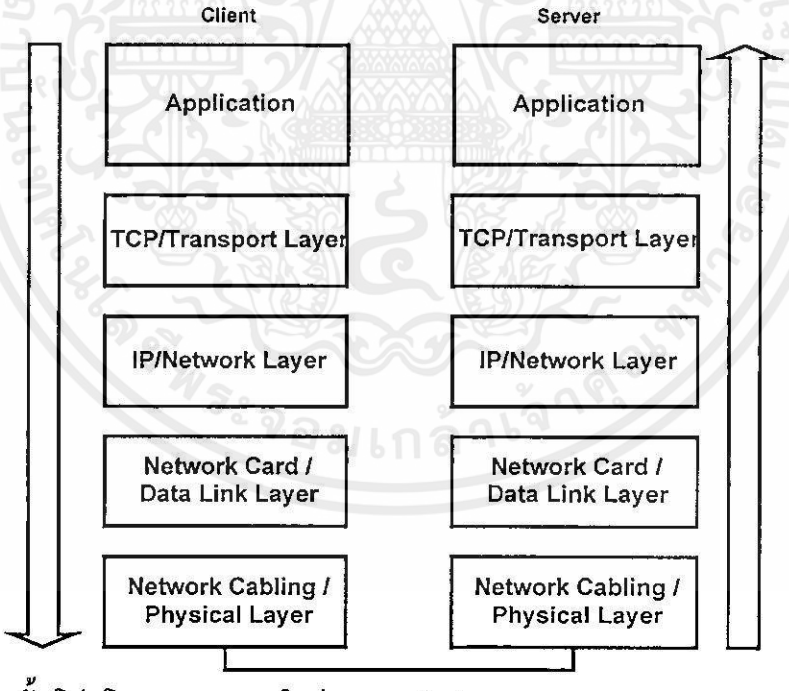
HTTP เป็นวิธีการส่งข้อมูลแบบไฮเปอร์เท็กซ์โดยพอร์ทมาตรฐานสำหรับโปรโตคอล HTTP คือ พอร์ทหมายเลข 80 โปรโตคอล HTTP ทำงานโดยใช้หลักการรีเควส์/เรสพ็อนท์ กล่าวคือจะเริ่มการสื่อสารเมื่อมีการร้องขอจากไคลเอนต์ไปยังเซิร์ฟเวอร์ เมื่อเซิร์ฟเวอร์ตอบรับการร้องขอนั้นจึงเริ่มการสื่อสาร และการสื่อสารจะยุติเมื่อฝ่ายใดฝ่ายหนึ่งทำการปิดการติดต่อไป ซึ่งรูปแบบวิธีการทำงานของ HTTP จะแสดงในรูปที่ 2.2



รูปที่ 2.2 แสดงรูปแบบวิธีการทำงานของ HTTP

2.4 โปรโตคอล TCP/IP

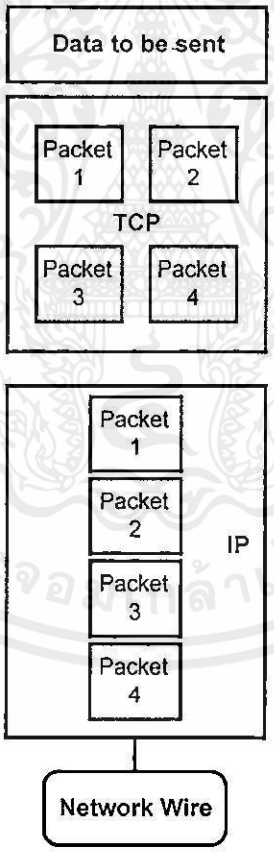
การสื่อสารบนเครือข่ายคอมพิวเตอร์ต้องพึ่งพาโปรโตคอล ซึ่งเป็นกฎระเบียบและกระบวนการที่ทำให้การแลกเปลี่ยนข้อมูลและสารสนเทศทำงานได้ อินเตอร์เน็ตทำงานอยู่บนโปรโตคอลมาตรฐานที่เรียกว่า TCP/IP (Transmission Control Protocol/Internet Protocol) การทำงานของ TCP/IP ได้กำหนดการทำงานออกเป็นระดับชั้น (Layer) [19] ดังรูปที่ 2.3



รูปที่ 2.3 แสดงชั้นโปรโตคอลและการติดต่อของ TCP/IP และ HTTP

การทำงานในระดับล่างสุดของโปรโตคอล TCP/IP คือ ฟิสิกคอล อันเป็นส่วนที่เกี่ยวข้องกับอุปกรณ์ต่างๆ ที่ใช้ในการติดต่อสื่อสารตลอดจนถึงสายเคเบิลแบบต่างๆ ที่ใช้ในการติดต่อ ในชั้นดาต้าลิงค์เป็นส่วนของตัวจัดการให้อยู่ในรูปแบบการส่งข้อมูล, ตรวจสอบและแก้ไขความผิด

พลาดก่อนส่งเข้าสู่ชั้นฟิสิกคอล ในชั้น IP หรือชั้นเน็ตเวิร์กเลเยอร์นี้จะคอยทำหน้าที่ในการส่งถ่ายข้อมูลไปยังปลายทางที่ต้องการ ในชั้นที่อยู่สูงกว่า IP คือชั้นทรานสปอร์ตเลเยอร์นั้นเป็นชั้นที่จะคอยตรวจสอบความถูกต้องในการส่งข้อมูล หากมีข้อผิดพลาดในการส่งข้อมูล TCP จะเป็นตัวคอยจัดการให้ส่งข้อมูลส่วนที่ขาดหายไปใหม่ ทำให้ในบางครั้งการส่งถ่ายข้อมูลอาจจะต้องทำมากกว่า 1 ครั้งหากเกิดข้อผิดพลาดในการส่งข้อมูล และหน้าที่อีกประการหนึ่งของ TCP คือ การจัดเรียงข้อมูลที่ได้รับ เนื่องจากในการส่งข้อมูลจะทำการแบ่งข้อมูลออกเป็นส่วนๆ ที่เรียกว่าแพ็กเกตแล้วจึงทำการส่ง ดังนั้นอาจเป็นไปได้ว่าข้อมูลที่ได้รับมานั้นไม่เรียงลำดับอย่างถูกต้องหรือมีการซ้ำซ้อนของข้อมูล หน้าที่ของ TCP ตรงส่วนนี้ คือ จะคอยเรียงลำดับข้อมูลที่ได้มาให้ถูกต้องตรงตามที่ส่งมาและตัดข้อมูลที่ซ้ำซ้อนออกไป ซึ่งหนึ่งในรูปแบบของ TCP/IP แอปพลิเคชั่นก็คือ FTP (File Transfer Protocol) ส่วนชั้นบนสุดคือชั้นแอปพลิเคชั่น ซึ่งเป็นชั้นของโปรแกรมที่ทำการติดต่อระหว่างกัน จากรูปที่ 2.4 ได้แสดงแผนภาพความสัมพันธ์ระหว่าง TCP และ IP [19]



รูปที่ 2.4 แสดงรูปแบบความสัมพันธ์ระหว่าง TCP และ IP

2.5 เว็บแอปพลิเคชั่น

เว็บแอปพลิเคชั่นเป็นการพัฒนาระบบงานบนเวิลด์ไวด์เว็บภายใต้เครือข่ายอินเตอร์เน็ต โดยลักษณะการทำงานจะแบ่งออกเป็น 2 ส่วนใหญ่ๆ คือ ส่วนของผู้ใช้บริการ (Client) และผู้ให้

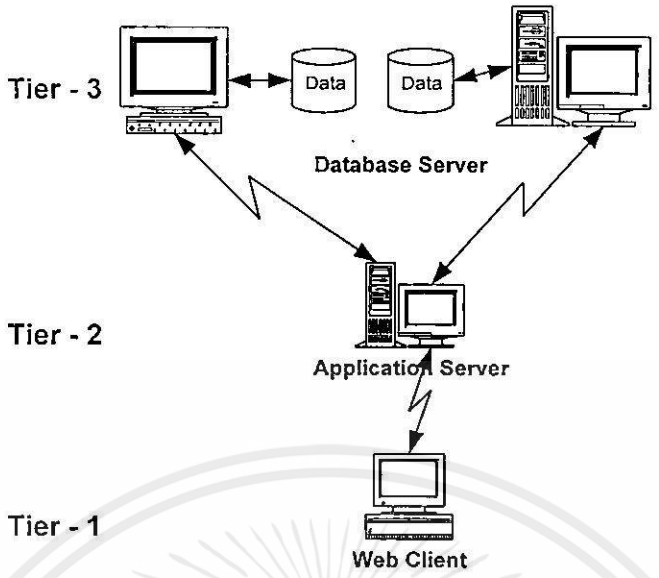
บริการ (Server) เรียกว่า ไคลเอนต์/เซิร์ฟเวอร์ โดยส่วนใหญ่จะทำงานอยู่บนเครื่องคอมพิวเตอร์คนละเครื่องเชื่อมต่ออยู่ภายใต้เครือข่ายสื่อสารข้อมูล ซึ่งอาจจะเป็นเครือข่ายอินเตอร์เน็ตหรืออินทราเน็ต วิธีการทำงาน คือ ไคลเอนต์จะทำการส่งคำร้องขอไปยังเซิร์ฟเวอร์ โดยคำร้องขอจะถูกส่งผ่านเครือข่ายไปยังเซิร์ฟเวอร์ที่ให้บริการ และเมื่อเซิร์ฟเวอร์ได้รับคำร้องขอจะทำการประมวลผลและส่งผลลัพธ์เข้าสู่เครือข่ายเพื่อส่งต่อไปให้ไคลเอนต์

ระบบเว็บแอปพลิเคชั่นมีลักษณะเป็นระบบแบบกระจายโดยมีรูปแบบการทำงาน 4 ส่วนหลักดังนี้คือ [3]

- **ระบบการรับและแสดงผล (Presentation Processing Logic)**
  เป็นส่วนที่ติดต่อกับผู้ใช้ เช่น การเขียน, การอ่าน, การป้อนข้อมูล, การแสดงผล
- **ระบบการจัดการและประมวลผลการทำงาน (Business Processing Logic)**
  เป็นส่วนที่เกี่ยวข้องกับการเขียนโปรแกรมในการรับข้อมูลจากผู้ใช้ในการประมวลผล
- **ระบบการประมวลผลและการเข้าถึงฐานข้อมูล (Database Processing Logic)**
  เป็นส่วนที่ทำหน้าที่ในการประมวลผลและจัดการฐานข้อมูลที่ต้องเกี่ยวข้องกับแอปพลิเคชั่น ซึ่งใช้ระบบการจัดการฐานข้อมูล (DBMS) เพื่อเข้าถึงและจัดการฐานข้อมูลโดยอาจจะเป็นแบบรีเลชันและใช้ภาษา SQL (Structured Query Language) ในการเข้าถึงฐานข้อมูล
- **ระบบการจัดการฐานข้อมูล (Database Management System :DBMS)**
  เป็นส่วนที่ดูแลและจัดการฐานข้อมูลโดยตรง

เว็บแอปพลิเคชั่นส่วนใหญ่จะให้การทำงานในส่วนการรับและแสดงผลข้อมูลในส่วนของไคลเอนต์ ส่วนที่เหลือจะทำงานอยู่ในส่วนของเซิร์ฟเวอร์ทั้งหมด ซึ่งสามารถแยกกระจายการทำงานให้อยู่ใน 3 ส่วน คือ ส่วนที่ 1 ทำหน้าที่ในการรับและแสดงผลข้อมูล, ส่วนที่ 2 ทำหน้าที่ระบบการจัดการและประมวลผลการทำงาน และส่วนที่ 3 ทำหน้าที่เก็บข้อมูลและมีระบบการจัดการฐานข้อมูลคอยควบคุมดูแล ซึ่งในส่วนของเว็บแอปพลิเคชั่นสามารถแบ่งโครงสร้างการทำงานและหน้าที่ความรับผิดชอบออกเป็น 3 ระดับ (3-Tier) ดังรูปที่ 2.5 และรายละเอียดดังต่อไปนี้

- ระดับที่ 1 เป็นส่วนของเว็บไคลเอนต์ที่ทำหน้าที่ส่งคำร้องขอข้อมูลไปยังเว็บเซิร์ฟเวอร์และคอยรับข้อมูลเพื่อแสดงผลบนหน้าจอ
- ระดับที่ 2 เป็นส่วนของแอปพลิเคชั่นเซิร์ฟเวอร์ที่ภายในประกอบด้วยเว็บเซิร์ฟเวอร์ที่ทำหน้าที่ติดต่อรับส่งข้อมูลกับเว็บไคลเอนต์และส่วนของเซิร์ฟเวอร์แอปพลิเคชั่นที่ทำหน้าที่ในการประมวลผลและติดต่อกับฐานข้อมูล
- ระดับที่ 3 เป็นระดับบนสุดที่ทำหน้าที่เป็นระบบจัดเก็บและจัดการฐานข้อมูล



รูปที่ 2.5 แสดงสถาปัตยกรรมการทำงาน 3 ระดับของเว็บแอปพลิเคชั่น

จากการทำงานของเว็บแอปพลิเคชั่นในการทำงานแบบ 3 ระดับก่อให้เกิดประโยชน์ทั้งในแง่การทำงานและประสิทธิภาพของระบบ กล่าวคือ ระบบจัดเก็บและจัดการฐานข้อมูลสามารถให้บริการฐานข้อมูลเดียวกันหรือต่างกันให้แก่เซิร์ฟเวอร์แอปพลิเคชั่นหรือแอปพลิเคชั่นอื่นๆ ได้พร้อมๆ กันหลายๆ ตัวได้ (ความสามารถในการบริการจัดการขึ้นอยู่กับระบบจัดการฐานข้อมูล) โดยในทางเดียวกันแอปพลิเคชั่นเซิร์ฟเวอร์ก็สามารถบริการแก่ไคลเอนต์พร้อมๆ กันได้หลายๆ ตัวเช่นกัน

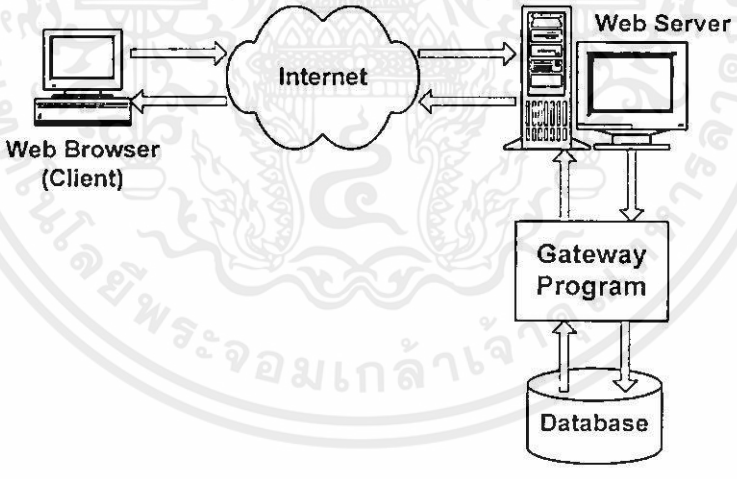
2.6 ฐานข้อมูลเวิลด์ไวด์เว็บ

ฐานข้อมูลเวิลด์ไวด์เว็บ คือ ระบบฐานข้อมูล ซึ่งเชื่อมต่อกับระบบเว็บแอปพลิเคชั่น, เว็บเบราเซอร์ โดยใช้เว็บเพจที่สร้างเป็นแบบฟอร์มจากภาษา HTML [3]

ระบบฐานข้อมูลเวิลด์ไวด์เว็บเป็นระบบไคลเอนต์/เซิร์ฟเวอร์แบบหนึ่ง กล่าวโดยรวมแล้วมีรูปแบบการทำงานที่ไม่ต่างจากเว็บแอปพลิเคชั่นทั่วๆ ไปมากนัก คือ ไคลเอนต์ (ในกรณีนี้คือเว็บเบราเซอร์) จะทำหน้าที่ร้องขอข้อมูลและบริการจากเซิร์ฟเวอร์ (ในที่นี้คือเว็บเซิร์ฟเวอร์) แล้วทำการแสดงผลให้ผู้ใช้ ในทางกลับกันทางฝั่งเซิร์ฟเวอร์จะคอยรับการร้องขอข้อมูลและบริการต่างๆ จากไคลเอนต์และคอยให้บริการสนองต่อการร้องขอเหล่านั้น

จุดที่น่าสนใจในการพัฒนาระบบฐานข้อมูลเวิลด์ไวด์เว็บ คือ ขั้นตอนในการเชื่อมระบบฐานข้อมูลเข้ากับระบบเวิลด์ไวด์เว็บ ซึ่งในการเชื่อมต่อระหว่างเว็บเซิร์ฟเวอร์กับเซิร์ฟเวอร์ฐานข้อมูลนี้จะอาศัยการทำงานของโปรแกรมเพื่อทำหน้าที่เป็นเกต์เวย์เชื่อมการทำงานระหว่างเว็บเซิร์ฟเวอร์และเซิร์ฟเวอร์ฐานข้อมูล ซึ่งมีขั้นตอนการทำงานดังรูปที่ 2.6 และรายละเอียดดังต่อไปนี้

- เมื่อเว็บเบราเซอร์รับการป้อนข้อมูลจากผู้ใช้และส่งการร้องขอตามที่ผู้ใช้ต้องการพร้อมกับข้อมูลที่จำเป็นสำหรับการสืบค้นข้อมูลตามที่ผู้ใช้ป้อนไปยังเว็บเซิร์ฟเวอร์
- เมื่อเว็บเซิร์ฟเวอร์ได้รับการร้องขอจากไคลเอนต์แล้วเว็บเซิร์ฟเวอร์จะกระตุ้นการทำงานของเกต์เวย์โปรแกรมและส่งผ่านข้อมูลที่จำเป็นสำหรับการสืบค้นข้อมูลตามที่ผู้ใช้ระบุส่งไปยังเกต์เวย์โปรแกรม
- เกต์เวย์โปรแกรมจะทำการประมวลผลตามที่ถูกกำหนดไว้กับเซิร์ฟเวอร์ฐานข้อมูล
- เมื่อระบบเซิร์ฟเวอร์ฐานข้อมูลได้รับข้อมูลที่จำเป็นสำหรับการประมวลผลจากเกต์เวย์โปรแกรมแล้ว ระบบเซิร์ฟเวอร์ฐานข้อมูลจะมองข้อมูลที่มาจากเกต์เวย์โปรแกรมเป็นทรานเซคชั่น เมื่อเซิร์ฟเวอร์ฐานข้อมูลทำการประมวลผลและได้ข้อมูลตามที่ต้องการแล้วจะส่งข้อมูลที่ได้ออกไปยังเกต์เวย์โปรแกรม
- เมื่อเกต์เวย์โปรแกรมได้รับผลที่ได้จากเซิร์ฟเวอร์ฐานข้อมูลแล้ว เกต์เวย์โปรแกรมจะส่งผ่านข้อมูลเหล่านั้นไปยังเว็บเซิร์ฟเวอร์
- เมื่อเว็บเซิร์ฟเวอร์ได้รับผลลัพธ์ที่ส่งมาจากเกต์เวย์โปรแกรมแล้ว เว็บเซิร์ฟเวอร์ก็จะส่งผ่านข้อมูลเหล่านั้นไปยังเว็บเบราเซอร์ในรูปของ HTML Page



รูปที่ 2.6 แสดงภาพโดยรวมของระบบฐานข้อมูลเวิลด์ไวด์เว็บ

2.7 ส่วนประกอบของระบบฐานข้อมูลเวิลด์ไวด์เว็บ

ระบบฐานข้อมูลเวิลด์ไวด์เว็บมีองค์ประกอบหลักๆ คือ ไคลเอนต์, เว็บเซิร์ฟเวอร์, เกต์เวย์โปรแกรมและเซิร์ฟเวอร์ฐานข้อมูล ดังรายละเอียดต่อไปนี้ [3]

2.7.1 ไคลเอนต์

ไคลเอนต์ในระบบฐานข้อมูลเวิลด์ไวด์เว็บ ส่วนใหญ่จะหมายถึงเว็บเบราเซอร์ การใช้เว็บเบราเซอร์เป็นไคลเอนต์ช่วยให้เกิดข้อดีในแง่ของการไม่ขึ้นกับเพล็ตฟอร์มใด เนื่องจากเว็บเบราเซอร์ทำงานได้ในแทบทุกเพล็ตฟอร์มคอมพิวเตอร์

ในการสร้างเว็บเพจ เพื่อใช้เป็นฟอร์มสำหรับติดต่อกับผู้ใช้นั้น ควรใช้ภาษา HTML อันเป็นมาตรฐานไม่ควรใช้ชุดคำสั่งหรือภาษาที่ยังไม่เป็นมาตรฐานเนื่องจากจะเกิดปัญหากับการทำงานร่วมกับเว็บเบราเซอร์บางชนิด ซึ่งจะส่งผลให้เกิดข้อจำกัดในการเข้าถึงข้อมูลในระบบฐานข้อมูลเวิลด์ไวด์เว็บ

2.7.2 เว็บเซิร์ฟเวอร์

เว็บเซิร์ฟเวอร์ทำหน้าที่ทุกอย่างคล้ายกับเซิร์ฟเวอร์ในระบบไคลเอนต์/เซิร์ฟเวอร์ เช่น การจัดการไฟล์, การสนองตอบการร้องขอบริการของไคลเอนต์, การส่งข้อมูลไปยังไคลเอนต์ตามที่ไคลเอนต์ต้องการ

เว็บเซิร์ฟเวอร์สามารถเชื่อมต่อกับแอปพลิเคชั่นใดๆ ที่ไม่ใช่ HTTP Application อาทิเช่น เซิร์ฟเวอร์ฐานข้อมูลได้โดยการทำงานผ่านเกต์เวย์ เช่น CGI (Common Gateway Interface) หรือ API (Application Programming Interface) เป็นต้น เว็บเซิร์ฟเวอร์เป็นองค์ประกอบที่สำคัญมากในระบบเวิลด์ไวด์เว็บ ประสิทธิภาพของเว็บเซิร์ฟเวอร์หรือข้อกำหนดใดๆ ในระบบเว็บเซิร์ฟเวอร์ย่อมส่งผลโดยตรงต่อการทำงานของระบบเว็บแอปพลิเคชั่น ซึ่งรวมถึงระบบฐานข้อมูลเวิลด์ไวด์เว็บด้วย ในปัจจุบันมีระบบเว็บเซิร์ฟเวอร์อยู่มากมายหลายโปรแกรม ซึ่งเว็บเซิร์ฟเวอร์เหล่านี้ล้วนสนับสนุนมาตรฐาน HTTP แต่จะมีข้อแตกต่างกันในรายละเอียดปลีกย่อยต่างๆ เช่น ความสามารถในการจัดการกับหลายความต้องการ, การจัดการทรัพยากรของระบบหรือระบบรักษาความปลอดภัย เป็นต้น

ข้อที่ควรพิจารณาในกรณีที่ต้องการเลือกระบบเว็บเซิร์ฟเวอร์

- จำนวนของไคลเอนต์ที่ติดต่อกับเซิร์ฟเวอร์
- เพล็ตฟอร์มและระบบปฏิบัติการ (Operating System)
- ระดับความจำเป็นในระบบรักษาความปลอดภัย
- ฟังก์ชันพิเศษต่างๆ
- งบประมาณ
- การสนับสนุนและประสิทธิภาพในการทำงานร่วมกับ CGI
- กลไกในการ Logging และการตรวจสอบ
- การมี API ที่ทำหน้าที่แทน CGI หรือไม่
- การมีระบบเชื่อมโยงกับฐานข้อมูลหรือไม่

ตัวอย่างเว็บเซิร์ฟเวอร์ที่นิยมใช้กันอย่างแพร่หลายในปัจจุบัน เช่น Microsoft Internet Information Server, Netscape Enterprise Server, Apache Server เป็นต้น

2.7.3 Gateway Program

ในการเชื่อมต่อระหว่าง HTTP Server กับแอปพลิเคชั่นใดที่ไม่ใช่ HTTP Application จำเป็นต้องอาศัยการทำงานของโปรแกรม ซึ่งแบ่งออกได้เป็น 2 ประเภทใหญ่ๆ คือ

1. CGI (Common Gateway Interface) ดังรายละเอียดในหัวข้อ 2.10
2. API (Application Programming Interface) ดังรายละเอียดในหัวข้อ 2.11

2.7.4 เซิร์ฟเวอร์ฐานข้อมูล

เซิร์ฟเวอร์ฐานข้อมูลโดยทั่วไปจะไม่สามารถเชื่อมต่อกับเว็บเซิร์ฟเวอร์ได้โดยตรงแต่จะอาศัยการเชื่อมต่อโดยผ่านทาง Native driver หรือ ODBC เซิร์ฟเวอร์ฐานข้อมูลจะมองคำสั่งจากการคิวรีจากเวิลด์ไวด์เว็บเป็นทรานเซกชั่นธรรมดาๆ เท่านั้น ดังนั้นหัวใจในการสร้างการเชื่อมต่อระบบเวิลด์ไวด์เว็บเข้ากับเซิร์ฟเวอร์ฐานข้อมูลจึงอยู่ที่ API หรือ CGI ปัจจุบันผู้ผลิตระบบฐานข้อมูลต่างเห็นความสำคัญของการเชื่อมต่อระบบฐานข้อมูลของตนเข้ากับระบบเวิลด์ไวด์เว็บ ดังนั้นจะเห็นได้ว่าระบบฐานข้อมูลที่ออกมาในรุ่นใหม่ๆ นี้จะมีเครื่องมือหรือคุณลักษณะที่ช่วยในการเชื่อมต่อระบบฐานข้อมูลเข้ากับระบบเวิลด์ไวด์เว็บให้สามารถทำได้ง่ายขึ้นและมีประสิทธิภาพที่ดีขึ้น

2.8 ข้อดีของระบบฐานข้อมูลเวิลด์ไวด์เว็บ

- **การมีระบบติดต่อกับผู้ใช้แบบกราฟฟิก**

  การใช้ระบบเว็บเบราเซอร์เป็นเครื่องมือในการติดต่อกับผู้ใช้ ทำให้ช่วยเพิ่มประสิทธิภาพในการใช้งานของผู้ใช้ เนื่องจากเว็บเบราเซอร์มีระบบการติดต่อกับผู้ใช้แบบกราฟฟิกจึงง่ายต่อการใช้งานและอยู่ในรูปแบบติดต่อกับผู้ใช้ที่เป็นมาตรฐาน

- **ความเป็นมาตรฐาน**

  ด้วยการทำงานระบบเวิลด์ไวด์เว็บจึงมีภาษา HTML เป็นภาษามาตรฐาน ผู้ใช้หรือนักพัฒนาระบบสามารถทำงานร่วมกับระบบฐานข้อมูลเวิลด์ไวด์เว็บได้โดยง่าย ด้วยภาษาเพียงภาษาเดียวและมาตรฐานกราฟฟิกแบบเดียวกัน

- **การไม่ขึ้นกับเพล็ตฟอร์ม**

  อาศัยการทำงานของระบบเวิลด์ไวด์เว็บจึงทำให้ระบบฐานข้อมูลเวิลด์ไวด์เว็บได้รับการถ่ายทอดคุณสมบัติในการไม่ขึ้นกับเพล็ตฟอร์มของระบบคอมพิวเตอร์ใดๆ ช่วยให้ผู้ใช้ไม่ว่าจะใช้คอมพิวเตอร์ในระบบใดก็สามารถที่จะเข้าถึงข้อมูลได้

- เพิ่มความสามารถในการเข้าถึงข้อมูล

  การใช้ระบบอินเตอร์เน็ตช่วยให้สามารถเข้าถึงและใช้งานระบบเครือข่ายคอมพิวเตอร์ได้ โดยผู้ใช้หรือโปรแกรมเมอร์ไม่จำเป็นต้องมีความรู้เกี่ยวกับระบบเครือข่ายคอมพิวเตอร์มากนัก ขณะเดียวกันจะช่วยแก้ปัญหาเกี่ยวกับเพล็ตฟอร์มของฮาร์ดแวร์และค่าใช้จ่ายในการซื้อซอฟท์แวร์เพิ่มเติม

- ง่ายต่อการขยายระบบ

  สามารถขยายระบบฐานข้อมูลได้โดยเชื่อมระบบฐานข้อมูลที่ต่างเพล็ตฟอร์มกันเข้าด้วยกัน

2.9 การรักษาความปลอดภัย

ข้อที่ควรดำนึงถึงเป็นอย่างมากประการหนึ่งในการพัฒนาระบบฐานข้อมูลเวิลด์ไวด์เว็บก็คือ การรักษาความปลอดภัย เนื่องจากระบบฐานข้อมูลที่อยู่ในระบบฐานข้อมูลเวิลด์ไวด์เว็บมีโอกาสที่จะถูกเข้าถึงได้จากผู้ใช้ต่างๆ มากมายจากทั่วทุกมุมโลกและทุกเวลา ดังนั้นการรักษาความปลอดภัยของข้อมูลที่อยู่ในระบบฐานข้อมูลเวิลด์ไวด์เว็บจึงเป็นสิ่งที่ต้องตระหนักให้มากเมื่อทำการพัฒนาระบบฐานข้อมูลเวิลด์ไวด์เว็บ [3]

ตามที่ได้กล่าวแล้วว่าระบบฐานข้อมูลเวิลด์ไวด์เว็บเป็นระบบที่เกิดจากการรวมเอาเทคโนโลยีระบบเวิลด์ไวด์เว็บเข้ากับเทคโนโลยีระบบฐานข้อมูล ดังนั้นฐานข้อมูลเวิลด์ไวด์เว็บจึงเป็นที่รวมข้อดีของทั้งสองเทคโนโลยีเข้าไว้ด้วยกัน ในขณะเดียวกันก็ได้รับจุดอ่อนในด้านความปลอดภัยของทั้งสองด้วยเช่นกัน ดังนั้นประเด็นการรักษาความปลอดภัยของระบบฐานข้อมูลเวิลด์ไวด์เว็บจึงแบ่งออกได้เป็น 2 ส่วนใหญ่ๆ คือ

1. การรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์
2. การรักษาความปลอดภัยของระบบฐานข้อมูล

2.9.1 การรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์

เป้าหมายหลักในการรักษาความปลอดภัยของระบบเครือข่ายคอมพิวเตอร์มี 5 ประการคือ

- ป้องกันการแก้ไขข้อมูลโดยผู้ไม่ได้รับอนุญาต
- ป้องกันการขาดหายหรือซ้ำซ้อนของข้อมูล
- ข้อมูลต้องเป็นความลับ ผู้ไม่ได้รับสิทธ์จะไม่สามารถเข้าถึงข้อมูลนั้นๆ ได้
- ต้องสามารถยืนยันถึงผู้ส่งข้อมูลนั้นๆ ได้
- ต้องรับประกันได้ว่าข้อมูลไปถึงผู้รับที่ถูกต้อง

2.9.2 การรักษาความปลอดภัยในระบบฐานข้อมูล

ในการรักษาความปลอดภัยของระบบฐานข้อมูลเวิลด์ไวด์เว็บนั้น ระบบฐานข้อมูลนี้ย่อมมีโอกาสเสี่ยงสูงต่อการถูกโจมตี เนื่องจากระบบฐานข้อมูลถูกเชื่อมต่อออกไปยังระบบอินเตอร์เน็ต จึงมีโอกาสเสี่ยงสูงต่อการถูกเข้าถึงได้ง่ายขึ้น ดังนั้นการรักษาความปลอดภัยในระบบฐานข้อมูลเวิลด์ไวด์เว็บจึงจำเป็นต้องกำหนดมาตรการในการรักษาความปลอดภัยทั้งกับตัวฐานข้อมูลเองด้วย นอกเหนือจากการรักษาความปลอดภัยในระบบเครือข่ายคอมพิวเตอร์ซึ่งจะต้องมีระบบรักษาความปลอดภัยที่ดีพอควร การรักษาความปลอดภัยในตัวระบบฐานข้อมูลดังเช่น ควบคุมระดับการเข้าถึงข้อมูลของผู้ใช้ด้วยการให้สิทธิ์ที่จำกัดแก่ผู้ใช้งานจากระบบฐานข้อมูลเวิลด์ไวด์เว็บ เช่น ให้สิทธิ์เพียงการอ่านอย่างเดียวไม่สามารถแก้ไขข้อมูลได้ เป็นต้น

2.10 Common Gateway Interface

Common Gateway Interface (CGI) มีการพัฒนาการแสดงผลข่าวสารแบบไดนามิกบนเวิลด์ไวด์เว็บเป็นรูปแบบแรก [13] โดย CGI ที่ทำงานบนระบบคอมพิวเตอร์นั้นจะทำการสร้างเว็บเพจโดยทันทีหลังจากที่ได้รับการร้องขอจากผู้ใช้ CGI สามารถทำความเข้าใจวิธีการต่างๆ ในการสร้างเพจได้และสามารถเรียกใช้ฟังก์ชันต่างๆ ของเครื่องเซิร์ฟเวอร์ได้ เช่น ฟังก์ชันที่ใช้ในการเรียกดูวันที่หรือเวลาของเครื่องเซิร์ฟเวอร์ ในการเขียนโปรแกรมในรูปแบบ CGI นั้นสามารถใช้ภาษาคอมพิวเตอร์ได้หลายภาษามาทำการเขียนได้ ซึ่งการประยุกต์ใช้ที่โดดเด่นของ CGI คือ การสร้างเว็บเพจที่สามารถรับข้อมูลจากผู้ใช้ที่เรียกกันว่าการสร้างฟอร์ม โดยหน้าตาของฟอร์มนั้นสามารถใช้ภาษา HTML ในการสร้างได้ แต่หลังจากที่ผู้ใช้ทำการกรอกข้อมูลเข้าไปในฟอร์มแล้วจะเป็นหน้าที่ของ CGI ที่จะนำข้อมูลนี้ไปทำงานอื่นๆ นอกจากนั้นยังสามารถนำ CGI ไปประยุกต์ใช้งานอื่นๆ ได้อีก เช่น อาจจะมีคำถามทิ้งไว้ให้ผู้เข้ามาเยี่ยมชมเว็บไซท์ได้ตอบคำถาม หรือการให้ข้อมูลข่าวสารที่ดึงจากฐานข้อมูลขององค์กร ซึ่งถ้าฐานข้อมูลเปลี่ยน เว็บเพจก็จะเปลี่ยนตามโดยอัตโนมัติ นั่นคือ CGI เป็นส่วนขยายความสามารถในการทำงานของเว็บเซิร์ฟเวอร์ โดย CGI มีรูปแบบกลไกการทำงานที่เกิดขึ้นระหว่างการใช้เว็บเซิร์ฟเวอร์กับโปรแกรมภายนอก (โปรแกรมที่สามารถประมวลผลได้เองภายใต้ระบบปฏิบัติการ) ที่พัฒนาขึ้นเพื่อประมวลผลและเข้าถึงฐานข้อมูล ซึ่งโปรแกรมภายนอกดังกล่าวจะมีการทำงานในลักษณะต่างๆ เช่น การประมวลผลการรับข้อมูลเข้าจากผู้ใช้, การคำนวณค่าต่างๆ, การสืบค้นข้อมูล, การเพิ่มเติมแก้ไขเปลี่ยนแปลงฐานข้อมูลและอื่นๆ โดยหลักการทำงานอย่างกว้างๆ ก็คือ รับข้อมูลจากผู้ใช้โดยผ่านทางเว็บเบราเซอร์ซึ่งเป็นไคลเอนต์และส่งข้อมูลนั้นไปยังเว็บเซิร์ฟเวอร์ และเมื่อเว็บเซิร์ฟเวอร์ตรวจคำสั่งแล้วพบว่าต้องมีการประมวลผลจากโปรแกรม CGI ก็จะส่งเงื่อนไขคำสั่งที่ได้รับไปให้โปรแกรม CGI ที่ระบุไว้ใน URL (Uniform Resource Locator) แล้วทำการประมวลผลและเมื่อโปรแกรมได้ผลลัพธ์ตามที่ต้องการแล้วจะส่งผลลัพธ์ดังกล่าวคืนสู่เว็บเซิร์ฟเวอร์และเว็บเซิร์ฟเวอร์ก็จะส่งกลับไปยังเว็บไคลเอนต์ที่ต้องการ

ข้อมูลในส่วนที่ค้นหาและแสดงผลต่อไป ถ้าไม่มี CGI แล้ว จะทำให้เว็บเซิร์ฟเวอร์นำเสนอได้เพียงแค่เอกสารแบบสแตติกและการเชื่อมโยงไปยังเพจหรือเซิร์ฟเวอร์อื่นเท่านั้น ดังนั้น CGI จึงเป็นตัวที่ทำให้เว็บสามารถโต้ตอบกันได้

2.10.1 GET และ POST Method

ขบวนการติดต่อกันระหว่างเซิร์ฟเวอร์กับ CGI เป็นเรื่องที่ผู้ที่ต้องการเขียนโปรแกรมติดต่อจะต้องเรียนรู้เพราะว่าจะมีขั้นตอนที่แตกต่างกันไปขึ้นกับเซิร์ฟเวอร์แต่ละตัวเป็นตัวกำหนด โดยทั่วไปแล้วการส่งข้อมูลระหว่างเว็บเบราเซอร์กับเว็บเซิร์ฟเวอร์จะใช้ method ที่เรียกว่า get และ post การใช้ method post จะเป็นการส่งข้อมูลไปพร้อมกับการเรียกชื่อเว็บไซท์ เช่น http://www.abcd.xyz?data1=one&data2=two ตัวอย่างนี้หมายความว่าได้ส่งข้อมูลมาพร้อมกับการเรียกเว็บไซท์ abcd.xyz โดยข้อมูลจะอยู่ถัดจากเครื่องหมาย ? แต่ละข้อมูลแยกกันด้วยเครื่องหมาย & ตัวอย่างนี้มี 2 ข้อมูล คือ data1 มีค่าเป็น one และ data2 มีค่าเป็น two ส่วนวิธีการ get จะมองไม่เห็นโดยตรงจากหน้าจอเพราะจะอาศัยการส่งจากโปรแกรมไปเลย ข้อจำกัดของการส่งแบบ post คือ ความยาวโดยรวมของข้อมูลที่ส่งต้องไม่เกิน 255 ตัว ส่วนวิธีการ get ไม่มีข้อจำกัดนี้ หลังจากเว็บเซิร์ฟเวอร์ได้รับข้อมูลแล้ว เว็บเซิร์ฟเวอร์ก็จะส่งต่อไปให้ CGI โดยการเรียกโปรแกรม CGI ซึ่งจะเรียกไปยังโปรแกรมที่อ้างมาพร้อมกับข้อมูลในหัวข้อ action ดังตัวอย่างต่อไปนี้

```
<form action="/cgi-bin/xyz.exe" method="post">
   ข้อมูลที่ 1 : <input type=text name=one>
   ข้อมูลที่ 2: <input type=text name=two>
   <input type="submit" value="Send">
</form>
```

แสดงว่า CGI คือโปรแกรม xyz.exe อยู่ที่เครื่องเซิร์ฟเวอร์ตำแหน่งไดเรคทอรีที่ชื่อ cgi-bin ข้อมูลที่ส่งมาพร้อมกัน คือ one และ two และจะมีค่าตามค่าที่เว็บเบราเซอร์กรอกลงไปตรงตำแหน่งข้อมูลที่ 1 และข้อมูลที่ 2 ตามลำดับ โปรแกรม CGI ในที่นี้ คือ xyz.exe จะรับข้อมูล one และ two ไปทำงานตามที่โปรแกรม xyz.exe ถูกเขียนขึ้นมา เช่น เอาข้อมูลไปค้นหาฐานข้อมูล เป็นต้น เมื่อได้ผลลัพท์ที่ตรงตามข้อกำหนดแล้ว CGI ก็จะเริ่มสร้างหรือส่งข้อมูลที่เป็น HTML script language เช่น

```
<html>
Result of ....
</html>
```

เมื่อเว็บเซิร์ฟเวอร์ได้รับก็จะส่งต่อกลับไปให้เว็บเบราเซอร์ แล้วเว็บเบราเซอร์ก็จะเอาข้อมูลดังกล่าวขึ้นสู่หน้าจอทันที จะเห็นว่า CGI ต้องสามารถรับค่าที่ส่งผ่านจากเว็บเซิร์ฟเวอร์ได้และส่งค่ากลับไปหาเว็บเซิร์ฟเวอร์ได้ด้วย

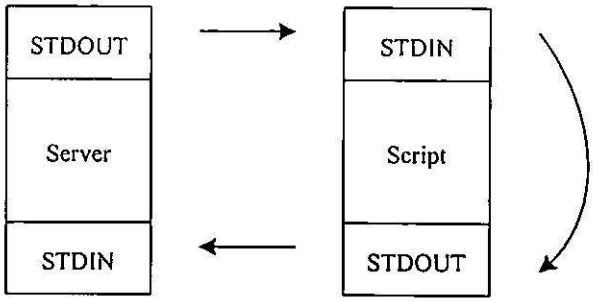
2.10.2 MIME Content Types

MIME (Multipurpose Internet Mail Extensions) เป็นมาตรฐานซอฟท์แวร์ที่กำหนดโดย Internet community เพื่อบอกถึงชนิดของเนื้อหาในไฟล์ที่ส่งผ่านทางอินเตอร์เน็ต โดยเว็บเบราเซอร์จะเข้าใจ MIME content type และใช้ในการแสดงข้อมูลในหน้าจอของเว็บเบราเซอร์ [1] เช่น ถ้า MIME content type ระบุว่าเป็นไฟล์ชนิด HTML text ดังนั้นเว็บเบราเซอร์จะจัดรูปแบบ HTML text และแสดงออกมาทางหน้าจอของเว็บเบราเซอร์ ในลักษณะเดียวกัน ถ้า MIME content type ระบุว่าเป็นไฟล์ชนิดที่เป็น audio ดังนั้นเว็บเบราเซอร์จะเล่นเสียงออกมาผ่านทางลำโพงของคอมพิวเตอร์

2.10.3 STDIN และ STDOUT

จากมาตรฐานการอินเตอร์เฟซ STDIN (Standard Input) และ STDOUT (Standard Output) ที่มีทำให้ผู้พัฒนาสามารถใช้เครื่องมือในการเขียนโปรแกรมได้อย่างมากมายที่สามารถอ่านค่าจาก STDIN ส่งค่าไปที่ STDOUT และสามารถอ่านค่า environment variable ต่างๆ ได้ ซึ่ง CGI สามารถรับค่า environment variable และอ่านค่าจาก STDIN ถ้าจำเป็น และทำงานตามที่ได้โปรแกรมไว้และเขียนผลลัพธ์ไปยัง STDOUT

ทุกๆ โปรแกรมจะมี STDIN และ STDOUT โดยเว็บเซิร์ฟเวอร์ส่งค่าทาง STDOUT ให้แก่ STDIN ของ CGI และ STDOUT ของ CGI ป้อนค่าให้กับ STDIN ของเว็บเซิร์ฟเวอร์ นั่นคือโปรแกรม CGI จะประมวลผลข้อมูลและส่งผลลัพธ์ที่ได้ไปยัง STDOUT แล้วเว็บเซิร์ฟเวอร์จะนำผลลัพธ์ที่ได้ผ่านทาง STDIN ของเว็บเซิร์ฟเวอร์และส่งผลลัพธ์ที่ได้นี้ไปยังไคลเอนต์ผ่านทาง STDOUT ของเว็บเซิร์ฟเวอร์ ในมุมมองของ CGI นั้น STDIN คืออะไรก็ตามที่รับจากเว็บเซิร์ฟเวอร์ และ STDOUT คืออะไรก็ตามที่ส่งออกไปให้เว็บเซิร์ฟเวอร์เมื่อ CGI ใช้วิธี post ส่วนวิธี get จะไม่ใช้ STDOUT แต่อย่างไรก็ตามทั้งสองกรณีเว็บเซิร์ฟเวอร์คาดว่า CGI จะส่งข้อมูลมาทาง STDOUT ของ CGI ดังแสดงในรูปที่ 2.7

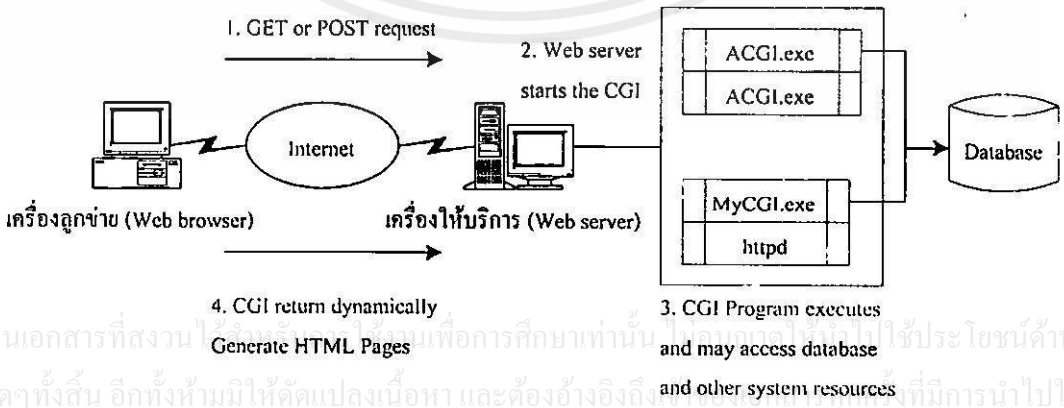


รูปที่ 2.7 แสดงการส่งผ่านข้อมูลระหว่างสคริปต์และเซิร์ฟเวอร์

โดย CGI จะส่งโคดที่เรียกว่า MIME (Multi-purpose Internet Mail Extensions) [14] ซึ่งเป็นมาตรฐานในการเปิดไฟล์ข้อมูลชนิดต่างๆ ไปยังเว็บเบราเซอร์ ทำให้เว็บเบราเซอร์รู้ว่าข้อมูลที่กำลังมาเป็นชนิดใด เนื่องจากข้อมูลนี้จะต้องส่งไปก่อนเสมอจึงเรียกว่า "header" ซึ่งเซิร์ฟเวอร์ไม่สามารถส่ง header ได้เพราะไม่อาจรู้ได้ว่า CGI จะส่งข้อมูลชนิดใด จึงเป็นหน้าที่ของ CGI เองที่จะต้องส่ง header และ MIME นี้ โดย CGI อาจส่งข้อมูลที่เป็นเสียง, กราฟฟิก, เท็กซ์, HTML หรืออื่นๆ เช่น การส่งข้อมูล "Content-type: text/html" ในส่วน header เป็นการบอกให้เว็บเบราเซอร์รู้ว่าข้อมูลที่กำลังมาเป็น HTML และถ้าหากส่ง "Content-type : image/gif" ก็จะเป็นการบอกว่าข้อมูลที่จะมานั้นเป็นภาพชนิด Gif ดังนั้นอาจเกิดความผิดพลาดขึ้นมาได้หาก CGI ไม่ส่ง header เพราะว่าเว็บเบราเซอร์จะไม่เข้าใจข้อมูลที่ส่งมา

2.10.4 โครงสร้างของ CGI

CGI เป็นการเตรียมรูปแบบกลไกการทำงานที่ใช้สำหรับติดต่อกับเว็บเซิร์ฟเวอร์ เพื่อที่จะทำการประมวลผลโปรแกรมภายในเว็บเซิร์ฟเวอร์ โดยโปรแกรม CGI สามารถเข้าถึงทรัพยากรของระบบได้อย่างอิสระและจะทำการแสดงผลออกมาในรูปแบบของ HTML กลับมายังเครื่องลูกข่าย (เว็บเบราเซอร์) ที่ทำการติดต่อ ในรูปที่ 2.8 เป็นการแสดงโครงสร้างการทำงานของ CGI



รูปที่ 2.8 แสดงโครงสร้างของ CGI

จากรูปที่ 2.8 สามารถอธิบายการทำงานเป็นลำดับขั้นตอน โดยแต่ละขั้นตอนจะมีหมายเลขกำกับไว้ดังรายละเอียดต่อไปนี้

1. เครื่องลูกข่ายจะทำการติดต่อไปยังเว็บเซิร์ฟเวอร์ เพื่อทำการประมวลผลโปรแกรม CGI โดยในการติดต่อระหว่างเครื่องลูกข่ายและเว็บเซิร์ฟเวอร์จะมีรูปแบบในการติดต่ออยู่สองวิธีคือ POST และ GET ซึ่งได้กล่าวรายละเอียดของวิธีทั้งสองไว้ในหัวข้อที่ 2.10.1 ที่ผ่านมา
2. เว็บเซิร์ฟเวอร์จะทำการเรียกแอปพลิเคชั่น CGI แล้วเริ่มต้นการประมวลผลโปรแกรม CGI โดยทำการติดต่อกับระบบปฏิบัติการ เพื่อทำการสร้างโปรเซสขึ้นมาใหม่ ซึ่งทำหน้าที่รันโปรแกรม CGI ซึ่งถ้าใช้ยูนิกซ์ กระบวนการนี้เรียกว่า fork แต่ถ้าใช้ภายใต้ Win32 API (เช่น Windows NT) จะเรียกกระบวนการนี้ว่า CreateProcess เมื่อระบบปฏิบัติการสร้างโปรเซสนี้แล้ว ก็จะต้องทำการจองหน่วยความจำและสภาพแวดล้อม (environment) ต่างๆ ซึ่งสภาพแวดล้อมเหล่านี้จะเป็นข้อมูลที่เกี่ยวกับการร้องขอ (Request) จากเครื่องลูกข่าย
3. ต่อจากนั้นแอปพลิเคชั่น CGI จะทำการประมวลผลและทำการแบ่งแยกการทำงานเป็นคนละโปรเซส
4. ผลที่ได้จากการประมวลผลโปรแกรม CGI จะทำการส่งออกมาทาง STDOUT ของโปรเซสที่ทำการรันโปรแกรม CGI นั้นๆ ทุกๆ ผลลัพธ์ที่ได้จากการรันจะถูกส่งออกมาจาก STDOUT แล้วส่งผ่านไปยังเครื่องลูกข่ายโดยจะส่งเป็นรูปแบบของ HTTP Header และผลลัพธ์ที่จะอ่านโดยผู้ใช้ ซึ่งจะต้องเป็นรูปแบบ HTML

2.10.5 การทำงานภายในของ CGI

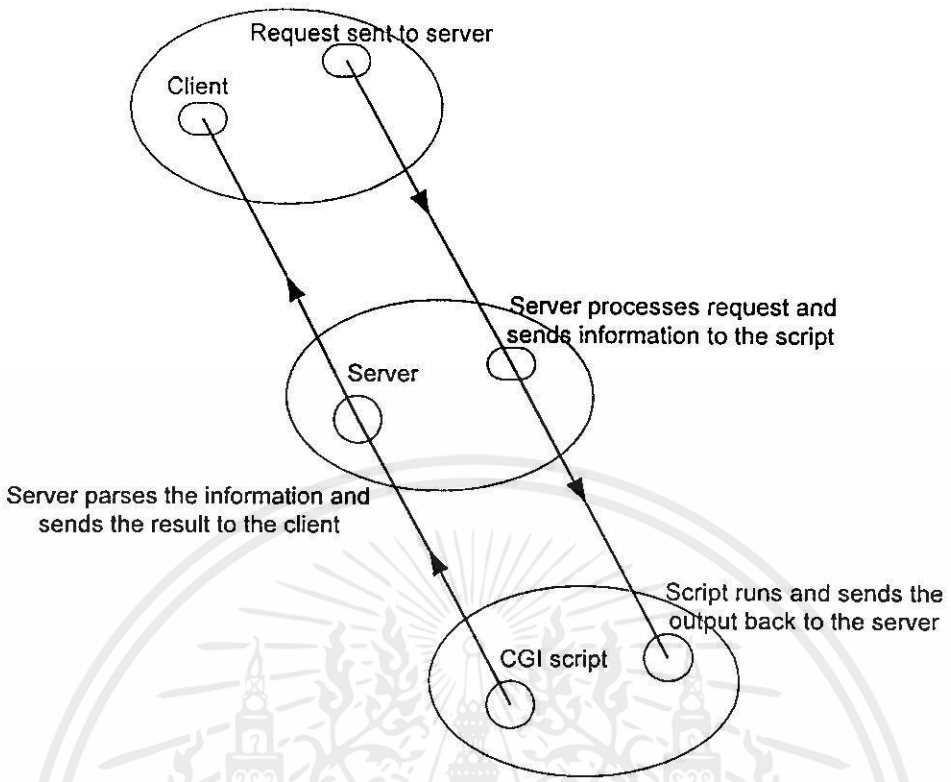
โดยส่วนใหญ่แล้ว CGI โปรแกรมและสคริปต์ต่างๆ จะถูกเก็บอยู่ในไดเรคทอรีพิเศษ โดยทั่วไปจะทำการเก็บอยู่ที่ไดเรคทอรี cgi-bin (การกำหนดไดเรคทอรีขึ้นอยู่กับการกำหนดเว็บเซิร์ฟเวอร์ของระบบ) เมื่อผู้ใช้ทำการส่ง URL แล้วเกี่ยวเนื่องกับโปรแกรม CGI เครื่องลูกข่ายจะทำการส่งการร้องขอเข้ามายังเครื่องเซิร์ฟเวอร์แล้วทำการติดต่อกับไฟล์ที่เป็นโปรแกรม CGI ที่อยู่ในไดเรคทอรีดังกล่าว [7]

โดยส่วนใหญ่แล้ว การร้องขอโปรแกรม CGI จะมีวิธีการหรือหลักการคล้ายกับการร้องขอเอกสารธรรมดาทั่วๆ ไป แต่จะแตกต่างกันตรงที่เมื่อมีการร้องขอมา เว็บเซิร์ฟเวอร์จะทำการแยกที่อยู่ (Address) ของเครื่องลูกข่ายและจะไม่ทำการส่งข้อมูลในไฟล์ที่ระบุมาเหมือนกับการร้องขอเอกสารธรรมดาแต่จะทำการประมวลผลโปรแกรมในไฟล์นี้ที่เป็นโคดของ CGI [7]

2.10.6 หลักการทำงานของ CGI

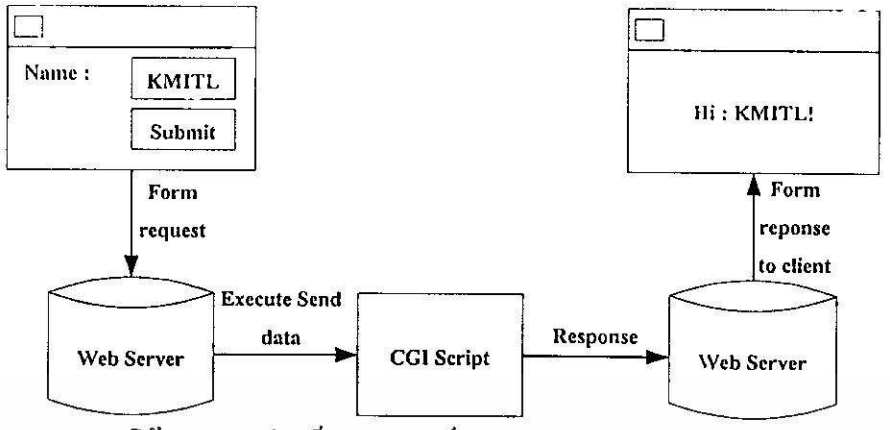
การทำงานของ CGI ขึ้นอยู่กับ method ที่ไคลเอนต์ร้องขอมา อาจจะใช้ method GET หรือ POST ก็ได้ โดยเว็บเซิร์ฟเวอร์จะรวบรวมข้อมูลที่ร้องขอมาจากไคลเอนต์ส่งต่อไปยัง CGI script ซึ่ง CGI script จะประมวลผลข้อมูลและส่งผลลัพธ์ที่ได้ไปยังเว็บเซิร์ฟเวอร์ ต่อจากนั้นเว็บเซิร์ฟเวอร์จะส่งผลลัพธ์กลับไปยังไคลเอนต์อีกทีหนึ่ง ดังแสดงในรูปที่ 2.9 การทำงานจะเริ่มจากใช้เว็บเบราเซอร์เรียกไปที่เว็บเซิร์ฟเวอร์ เช่น เรียกไปที่ http://161.246.10.21 โดยเว็บเซิร์ฟเวอร์แต่ละเครื่องจะมีหมายเลขประจำตัวซึ่งจะเอามาจากหมายเลข IP address แต่การจดจำหมายเลขอาจจะจำได้ยากจึงต้องมีการตั้งชื่อขึ้นมาใหม่ เช่นเป็น www.chaokhun.kmitl.ac.th การจะใช้ชื่ออะไรก็ต้องไปขอจดทะเบียนก่อนจึงจะได้ชื่อที่ต้องการ หลังจากนั้นเว็บเซิร์ฟเวอร์ก็จะส่งเอกสารกลับไปให้เว็บเบราเซอร์ เอกสารที่ส่งมาจะอยู่ในรูปของ HTML ซึ่งเป็นเท็กซ์ไฟล์ แล้วเว็บเบราเซอร์ก็จะตีความตามสคริปต์ HTML สร้างผลลัพท์เป็นเอกสารตามที่เว็บเซิร์ฟเวอร์ต้องการขึ้นมาที่หน้าจอของเว็บเบราเซอร์

จะเห็นว่าเอกสารต่างๆ จะต้องถูกส่งมาจากเว็บเซิร์ฟเวอร์ หมายความว่า เว็บเซิร์ฟเวอร์จะต้องมีเอกสารอยู่และจะต้องตรงกับเว็บเบราเซอร์ต้องการหรือเป็นไปตามที่เว็บเซิร์ฟเวอร์ต้องการเผยแพร่ หากสมมติว่าเว็บเซิร์ฟเวอร์นั้นให้บริการข้อมูลหนังสือในห้องสมุด โดยเอกสารจะแสดงหนังสือทีละเรื่องต่อเอกสาร 1 ฉบับ นั้นหมายความว่าหากมีหนังสือ 10,000 เล่ม เว็บเซิร์ฟเวอร์จะต้องเตรียมเอกสารเอาไว้ประมาณ 10,000 เอกสารตามชื่อหนังสือหรือตามเงื่อนไขที่อยากจะแสดง ซึ่งเป็นไปไม่ได้ที่จะทำเช่นนั้น เพราะว่าจะเกิดปัญหาตามมามากเช่น ระบบปฏิบัติการของวินโดวส์ 95 แต่ละไดเรคทอรีจะเก็บไฟล์ได้จำนวนจำกัด และถ้าหากต้องการอัพเดตข้อมูลจะทำอย่างไรจึงจะอัพเดตหมดทั้ง 10,000 ไฟล์ได้



รูปที่ 2.9 แสดงการโต้ตอบระหว่างไคลเอนต์, เซิร์ฟเวอร์และ CGI Script

ดังนั้นจึงมีการคิดวิธีการแก้ไขปัญหาที่กล่าวมา โดยการแทนที่จะเตรียมเอกสารเอาไว้ก่อน จะใช้วิธีการสร้างเอกสารขึ้นมาใหม่ทุกครั้ง เอกสารประเภทนี้จะถือว่าเป็นเอกสารที่ค่อนข้างทันสมัยที่สุดเพราะว่าเอกสารจะถูกสร้างขึ้นมาใหม่เสมอเมื่อเว็บเบราเซอร์ต้องการ แต่จะรู้ได้อย่างไรว่าเว็บเบราเซอร์ต้องการอะไร อย่างไรหรือเว็บเซิร์ฟเวอร์ให้บริการอะไรได้บ้าง เว็บเซิร์ฟเวอร์จึงจำเป็นต้องขอข้อมูลจากเว็บเบราเซอร์ โดยการให้เว็บเบราเซอร์ส่งหรือกรอกข้อมูลในแบบฟอร์มที่กำหนด แล้วส่งกลับไปให้เว็บเซิร์ฟเวอร์ จากนั้นเว็บเซิร์ฟเวอร์ก็สร้างเอกสารขึ้นมาแล้วส่งเอกสารกลับไปให้ แต่เนื่องจากเว็บเซิร์ฟเวอร์ต้องคอยบริการเว็บเบราเซอร์ซึ่งมีเข้ามาพร้อมๆ กันหลายราย ดังนั้นเว็บเซิร์ฟเวอร์จำเป็นต้องมอบงานการสร้างเอกสารใหม่นี้ไปให้ผู้ช่วยช่วยทำแทนในรูปแบบที่เรียกว่า Common Gateway Interface หรือ CGI ซึ่งจะเรียกโปรแกรมผู้ช่วยดังกล่าวว่าเป็น CGI การติดต่อดังแสดงได้ในรูปที่ 2.10



รูปที่ 2.10 แสดงการโต้ตอบระหว่างเว็บเบราเซอร์และ CGI script

การจะใช้อะไรเป็นเว็บเซิร์ฟเวอร์ อะไรเป็น CGI มีปัจจัยหลายประการ เช่นขึ้นกับโปรแกรมเมอร์ถนัดอะไร เช่น หากใช้โปรแกรม Web site ของ O'Reilly เป็นเว็บเซิร์ฟเวอร์ ก็จะใช้ Visual Basic เป็น CGI หรือใช้ Microsoft Personal Web Server (PWS) เป็นเว็บเซิร์ฟเวอร์ก็จะใช้ Active Server Page ผ่าน Visual InterDev เป็น CGI เป็นต้น

2.10.7 รหัส URL

การเข้ารหัส URL เป็นวิธีการที่การร้องขอของไคลเอนต์ถูกเปลี่ยนโดยการเข้ารหัสตัวอักษร ก่อนที่เว็บเซิร์ฟเวอร์จะส่งผ่านข้อมูลไปยัง CGI ซึ่งการเข้ารหัส URL ทำให้เกิดสิ่งที่ตามมา คือ ตัวอักษรว่างถูกเข้ารหัสเป็นตัวอักษร '+' ตัวอักษร ASCII ที่มีรหัสสูงกว่าตัวที่ 127 และต่ำกว่า 32 จะถูกเข้ารหัสโดยใช้เครื่องหมาย '%' และตามด้วยตัวเลขสองหลัก ซึ่งจะเป็นค่าของตัวอักษรนั้นในแบบเลขฐานสิบหก (hex) ฟิลด์ข้อมูลที่มาจากฟอร์มของ HTML ถูกแยกด้วยตัวอักษร '&' สำหรับในแต่ละฟิลด์จากฟอร์มจะใช้ตัวอักษร '=' เป็นตัวแยกระหว่างชื่อฟิลด์กับค่าที่ผู้ใช้ป้อนให้แก่ฟิลด์นั้น

ตัวอย่างเช่น สมมติว่าสร้างฟอร์มโดยใช้วิธี post ฟอร์มนี้มีฟิลด์ให้ผู้ใช้ป้อนข้อมูลอยู่ 3 อย่างคือ ชื่อผู้ใช้, ที่อยู่ e-mail และฟิลด์ที่ให้เติมอะไรก็ได้ ซึ่งอาจเป็นข้อเสนอแนะ หรือข้อคิดเห็นต่างๆ โดยผู้ใช้ได้ป้อนข้อมูลดังต่อไปนี้ คือ

- ฟิลด์ชื่อผู้ใช้ (uname) ให้ค่าเป็น "Dang Cholpripimonrat"
- ฟิลด์ที่อยู่ e-mail (email) ให้ค่าเป็น "s0067006@kmitl.ac.th"
- ฟิลด์ที่ให้เติมอะไรก็ได้ (description) ให้ค่าเป็น ~!@#$%^&*()-++|\/~

เมื่อฟอร์มได้ถูกส่งไป ข้อมูลของไคลเอนต์จะถูกเข้ารหัสบน URL ให้เว็บเซิร์ฟเวอร์และเว็บเซิร์ฟเวอร์จะส่งผ่านข้อมูลนี้ไปให้ CGI โดยข้อมูลข้างบนจะถูกเข้ารหัสเป็น

uname=Dang+Cholpripimonrat&email=s0067006@kmitl.ac.th

2.10.8 ภาษาที่สามารถเขียน CGI ได้

ภาษาที่สามารถเขียน CGI ได้มีอยู่ด้วยกันหลายภาษามากมาย ตัวอย่างเช่น

- AppleScript (Macintosh only)
- C/C++ (Unix, Windows, Macintosh)
- C shell (Unix only)
- PERL (Unix, Windows, Macintosh)
- Tcl (Unix only)
- Visaul Basic (Windows only)
- Borland Delphi (Windows only)

2.10.9 ข้อได้เปรียบของ CGI

วัตถุประสงค์เบื้องต้นของ CGI คือ เพื่อตอบสนองต่อการร้องขอข้อมูลจากไคลเอนต์ ซึ่งข้อมูลนั้นไม่ได้ถูกเก็บอยู่ในรูปแบบของเอกสาร HTML แต่อยู่ในรูปของแอปพลิเคชั่น ซึ่งสามารถปฏิบัติงานต่างๆ ได้ CGI Application จะถูกเรียกให้ปฏิบัติงานในรูปของโปรเซสบนระบบปฏิบัติการ โดยได้รับข้อมูลผ่านทาง Standard Input ในรูปแบบของ Environment Variables จากเว็บเซิร์ฟเวอร์ หลังจากประมวลผล CGI Application แล้ว ข้อมูลผลลัพธ์จะถูกส่งไปยังเว็บเซิร์ฟเวอร์ผ่านทาง Standard Output เพื่อส่งกลับไปยังไคลเอนต์ต่อไป โดยหนึ่งโปรเซสต่อหนึ่งการร้องขอ เมื่อมีการร้องขอหลายๆ อันเกิดขึ้นในเวลาพร้อมๆ กัน เว็บเซิร์ฟเวอร์ก็จะต้องสร้างโปรเซสใหม่ๆ ซึ่งทำงานแยกกันเพื่อจัดการกับการร้องขอนั้นๆ จะเห็นได้ว่า หลักการทำงานพื้นฐานเหล่านี้เป็นหลักการทำงานของระบบปฏิบัติการยูนิกซ์ตั้งแต่ดั้งเดิมซึ่งเป็นระบบปฏิบัติการที่สามารถรองรับการทำงานแบบหลายงาน (Preemptive Multitasking) และหลายผู้ใช้ (Multi-user) ได้ในเวลาเดียวกันอย่างไม่มีปัญหา รวมไปถึงหลักการของการรับส่งข้อมูลเข้าออกผ่านทาง Standard Input และ Standard Output ข้อมูลที่ได้รับทาง Standard Input จะอยู่ในรูปแบบของ Environment variable ฉะนั้นการพัฒนาและการแก้ไข CGI Application จึงทำได้ง่ายโดยเฉพาะผู้ที่มีพื้นฐานในการเขียนโปรแกรมยูนิกซ์อยู่แล้วหรือเป็นผู้ใช้ทั่วไป

2.10.9.1 การแยกโปรเซสทำงาน

ขณะที่ CGI Application หลายๆ อันกำลังปฏิบัติงานอยู่นั้น การใช้ทรัพยากรของแต่ละโปรเซส เช่น พื้นที่ของหน่วยความจำจะถูกแยกออกจากกันและแยกจากโปรเซสของเว็บเซิร์ฟเวอร์อย่างเด็ดขาด แม้ว่ามีโปรเซสใดๆ เกิดไม่ยอมทำงานขึ้นมาก็ไม่ส่งผลอย่างไรต่อการทำงานของเว็บเซิร์ฟเวอร์หรือโปรเซสอื่นๆ แต่อย่างใด เมื่อถึงเวลาที่กำหนดเว็บเซิร์ฟเวอร์ก็จะส่งข้อมูลแจ้งข้อผิดพลาดในการสั่งงานกลับไปสู่เครื่องไคลเอนต์ ส่งผลให้การทำงานของเว็บเซิร์ฟเวอร์มีความมั่นคง

2.10.9.2 ง่ายต่อการแก้ไข

การแก้ไข CGI Application ทำได้ค่อนข้างง่าย แต่ทั้งนี้ต้องขึ้นอยู่กับความซับซ้อนของเนื้อหาภายในซอสโคดหรือตัวโปรแกรมต้นแบบด้วย เพราะว่า CGI Application ทำงานแบบแยกโปรเซสจากเว็บเซิร์ฟเวอร์จึงสามารถแก้ไขแอปพลิเคชั่นโดยที่ไม่มีการแทรกแซงการทำงานของเว็บเซิร์ฟเวอร์หรือโปรเซสอื่นๆ เลย และโดยทั่วไปแล้ว CGI Application ที่ใช้บนยูนิกซ์จะใช้ภาษาสคริปต์ (Script Language) เช่น Perl, Tcl/Tk, Shell Script ในการสร้างแอปพลิเคชั่น ภาษาเหล่านี้จะใช้ตัวแปรภาษาแบบอินเตอร์พรีทเตอร์ (Interpreter) อ่านสคริปต์ไฟล์ ซึ่งเป็นไฟล์ตัวอักษรธรรมดาและสั่งให้ปฏิบัติงานทีละคำสั่งในเวลาปฏิบัติงานจริง การแก้ไขจึงทำได้ง่าย

2.10.10 ข้อจำกัดของ CGI

จุดเด่นของ CGI Application ที่ปฏิบัติงานแบบแยกโปรเซสที่เครื่องเว็บเซิร์ฟเวอร์ถือเป็นจุดบกพร่องอย่างยิ่ง เมื่อแต่ละการร้องขอในเวลาเดียวกันมีจำนวนมากๆ ทำให้เว็บเซิร์ฟเวอร์ต้องรับภาระในการสร้างและดูแลโปรเซสขึ้นมาใหม่เป็นจำนวนมาก อันเป็นสาเหตุทำให้เกิดผลกระทบดังนี้

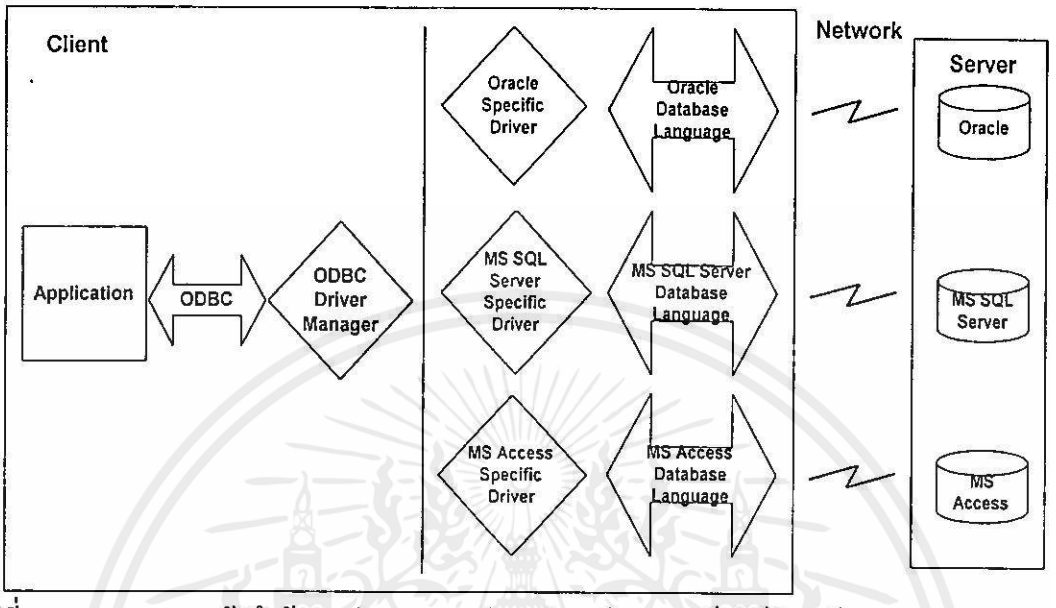
- ทรัพยากรของระบบถูกใช้เพิ่มมากขึ้นในแต่ละโปรเซสของ CGI Application โดยเฉพาะหน่วยความจำและอุปกรณ์อินพุท, เอาท์พุท เช่น ฮาร์ดดิสก์
- เวลาที่ใช้ในการตอบสนองมากขึ้นเพราะต้องเสียไปกับการสร้าง, การดูแลและการทำลายแต่ละโปรเซส

2.11 API (Application Program Interface)

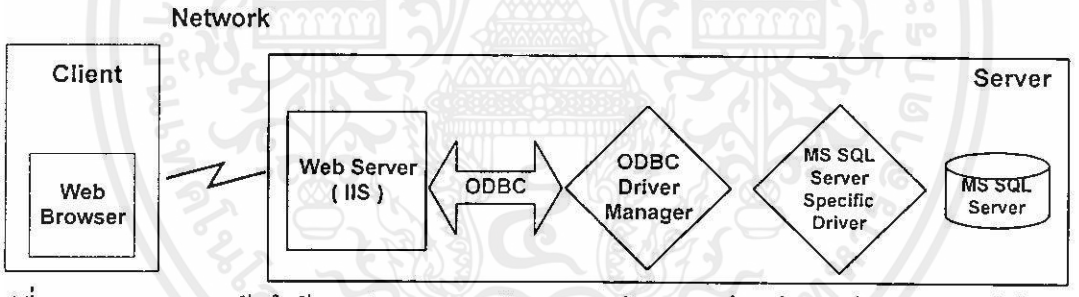
สืบเนื่องจากข้อจำกัดของ CGI ผู้ผลิตเว็บเซิร์ฟเวอร์หลายๆ แห่งจึงพัฒนา API ของตนเองขึ้นมาควบคู่กับเว็บเซิร์ฟเวอร์เพื่อให้ทำงานกับเว็บเซิร์ฟเวอร์ของตนเองอย่างมีประสิทธิภาพสูงที่สุดแทนการใช้ CGI [12] ซึ่ง API เป็น Native Code ที่ทำงานร่วมกับเว็บเซิร์ฟเวอร์เพื่อขยายขีดความสามารถของเว็บเซิร์ฟเวอร์ โดย API เหล่านี้สามารถทำหน้าที่อย่างที่ CGI ทำได้ทุกอย่างและยังมีความสามารถพิเศษ ซึ่งจะใช้ CGI ทำได้ไม่ง่ายนัก แต่ API มีคุณสมบัติแบบ Proprietary application (ระบบที่ยึดติดกับผู้ผลิตเฉพาะราย ไม่เป็นมาตรฐานกลาง) เนื่องจากถูกออกแบบมาเพื่อให้ทำงานได้อย่างมีประสิทธิภาพสูงสุดกับ Native Web Server ของผู้ผลิตรายนั้นๆ

ข้อดีของ API นอกจากจะมีประสิทธิภาพสูงกว่า CGI แล้ว ข้อดีอีกประการหนึ่งของ API คือ การใช้ทรัพยากรของระบบที่น้อยกว่า CGI อย่างไรก็ตามการพัฒนา API ด้วยการเขียนโปรแกรมเองแบบ Manual นั้นจะมีความยุ่งยากกว่าการพัฒนา CGI เป็นอย่างมาก เนื่องจากการเขียน API จำเป็นที่จะต้องใช้เทคนิคในการเขียนโปรแกรมขั้นสูง เช่น Multithreading, Process

Synchronization, Direct Protocol Programming และ Error Handing เป็นต้น จึงมักจะใช้ API ที่มากับผู้ผลิตเว็บเซิร์ฟเวอร์



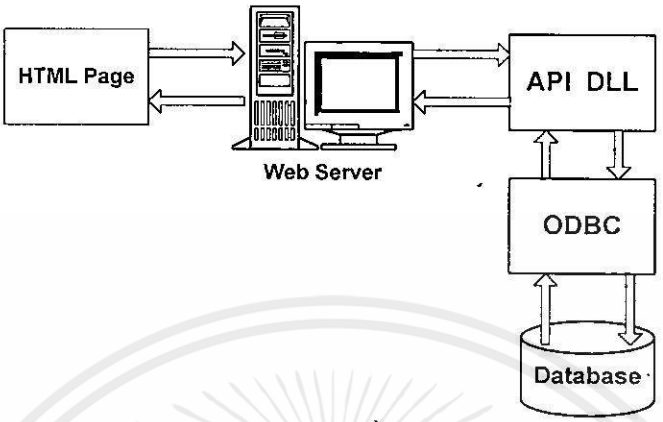
รูปที่ 2.11 แสดงการเข้าถึงข้อมูลผ่าน ODBC ในรูปแบบไคลเอนต์/เซิร์ฟเวอร์



รูปที่ 2.12 แสดงการเข้าถึงข้อมูลผ่าน ODBC ในรูปแบบไคลเอนต์/เซิร์ฟเวอร์ผ่านอินเตอร์เน็ต

จากรูปที่ 2.11 ในรูปแบบของไคลเอนต์/เซิร์ฟเวอร์นั้น การเชื่อมต่อฐานข้อมูลที่ต่างกันจะอิงกับ ODBC (Open Database Connectivity) ซึ่งเป็น Middleware อันเป็น API ตัวหนึ่งที่ต่อเชื่อม Proprietary Language เข้ากับฐานข้อมูล โดยการร้องขอจาก Client Application จะถูกส่งไปให้ ODBC Driver Manager บนตัวไคลเอนต์และถูกส่งต่อไปให้ Database Driver ของแต่ละ Proprietary Database Driver ต่อจากนั้นก็จะทำการส่งต่อไปสู่ฐานข้อมูล ทำให้โปรแกรมสามารถที่จะเรียกใช้หรือกระทำใดๆ กับตัวข้อมูลได้ถึงแม้จะต่างเพล็ตฟอร์ม ดังนั้นในรูปแบบของไคลเอนต์/เซิร์ฟเวอร์บนอินเตอร์เน็ตนั้น ดังรูปที่ 2.12 ก็จะมีลักษณะที่คล้ายๆ กันเพียงแต่การทำงานจะเริ่มการทำงานจากตัวเว็บเบราเซอร์ส่งการร้องขอข้อมูลสู่ตัวเว็บเซิร์ฟเวอร์ จากนั้นในส่วนเว็บเซิร์ฟเวอร์จะมี API DLL (Dynamic Link Library) ดังรูปที่ 2.13 ซึ่งจะทำหน้าที่ในการอินเตอร์เฟซเพื่อทำให้ตัว

เว็บเซิร์ฟเวอร์สามารถที่จะคุยกับ Database Server ได้โดยผ่าน ODBC อีกต่อหนึ่ง ซึ่ง ODBC เป็นตัวเชื่อมกับฐานข้อมูลของแต่ละบริษัท



รูปที่ 2.13 แสดงลักษณะการทำงานของ API : การเชื่อมต่อกับฐานข้อมูล

2.12 ภาษาแบบจำลองเสมือนจริง (VRML)

ปัจจุบันหลากหลายเทคโนโลยีที่เกี่ยวข้องกับเครือข่ายอินเตอร์เน็ตได้มีการพัฒนาอย่างต่อเนื่อง อินเตอร์เน็ตจึงเป็นเทคโนโลยีที่ได้รับการกล่าวขานมากที่สุดเทคโนโลยีหนึ่งจากคนทั่วโลก ทำให้ผู้คนจำนวนมากต้องการที่จะสร้างเว็บไซท์ เพื่อให้ข้อมูลข่าวสารของตนกับผู้ที่เข้ามาใช้บริการอินเตอร์เน็ต เพื่อประโยชน์ทางด้านธุรกิจและการศึกษา ภาษาที่ใช้ในการพัฒนาเว็บไซท์ได้มีการพัฒนาอย่างต่อเนื่องเช่นกัน ไม่ว่าจะเป็น SGML, HTML, CGI/C++, CGI/Perl หรือ Java ซึ่ง VRML เป็นภาษาหนึ่งที่สามารถนำไปใช้ในการพัฒนาเว็บไซท์ได้ แต่มีความสามารถที่จะนำเสนอภาพแบบ 3 มิติได้ VRML มีลักษณะการทำงานใกล้เคียงกับ HTML มาก คือ มีการเก็บข้อมูลในรูปแบบของแอสกีไฟล์ เพื่อให้โปรแกรมที่ทำหน้าที่แสดงผล (โปรแกรมเว็บเบราเซอร์) ทำหน้าที่แปลความผ่านโปรโตคอลเดียวกัน (HTTP) ดังนั้น VRML จึงสามารถทำการเชื่อมโยงไปใช้ทรัพยากรอื่นๆ บนเครือข่ายอินเตอร์เน็ตได้

2.12.1 บทนำ

VRML (Virtual Reality Modeling Language) เป็นภาษาที่ใช้อธิบายลักษณะของวัตถุ 3 มิติ (3D) เพื่อใช้ในการสร้างภาพ 3 มิติให้สามารถโต้ตอบกับผู้ใช้ได้ โดยปกตินามสกุลของไฟล์ VRML จะมีทั้ง .wrl และ .wrz โดยต้องใช้ปลั๊กอินที่จะเสริมให้เบราเซอร์สามารถดูโมเดล 3 มิติได้ ซึ่ง VRML คือ Virtual Reality รูปแบบหนึ่งของอินเตอร์เน็ตในอนาคต แต่เนื่องจากรูปแบบของภาษาที่เข้าใจยาก ทำให้เขียนยาก (ถ้าเทียบกับ HTML แล้ว HTML เขียนง่ายกว่า) เวลาเขียนผู้ใช้ต้องกำหนดพิกัดต่างๆ บนแกน X,Y,Z ต้องกำหนดรูปแบบของโพลีกอน, ภาพประกอบพื้นผิวอื่นๆ แต่ปัจจุบันมีทูลหลายๆ ตัวที่ช่วยให้การเขียนง่ายขึ้น แต่อีกเหตุผลหนึ่งที่ VRML ไม่เป็นที่นิยม คือ

เวลาในการโหลดค่อนข้างช้า, ยังใหม่เกินไป และผู้ใช้ส่วนใหญ่ยังชอบลักษณะอินเตอร์เฟซแบบ 2 มิติอยู่ แต่การใช้ VRML จะประหยัดเวลาในการโหลดข้อมูลมาเพราะเป็นเท็กซ์ไฟล์ แต่ก็จะมารับภาระหนักที่เครื่องไคลเอนต์ที่จะต้องแปลข้อความนั้นให้อยู่ในรูป 3 มิติที่สามารถหมุนหรือเข้าไปดูข้างในๆ ได้ จึงต้องทำงานบนเครื่องที่มีประสิทธิภาพสูงๆ [1,15]

ภาษา VRML2.0 ถูกพัฒนาต่อจากภาษา VRML1.0 ซึ่งมีการปรับปรุงเปลี่ยนแปลง, มีการเพิ่มเติมและลดลงของส่วนต่างๆ ของภาษา แต่ความสามารถของภาษา VRML2.0 ที่เพิ่มขึ้นมาเป็นสิ่งที่สำคัญมาก โดยคุณสมบัติที่มีเพิ่มเติมขึ้นมาในภาษา VRML2.0 คือ การสร้างการเคลื่อนไหวให้กับวัตถุ, การสร้างการโต้ตอบระหว่างผู้ใช้กับวัตถุ, การให้เสียงแก่โลกเสมือน, การนำวัตถุกลับมาใช้ใหม่ และความสามารถในการสร้างโปรแกรมสคริปต์จากภาษาอื่น เช่น JAVA เป็นต้น ด้วยคุณสมบัติเหล่านี้ทำให้สามารถสร้างโลกเสมือนให้มีความเหมือนจริงได้มากยิ่งขึ้น โลกเสมือนที่ได้จะมีชีวิตชีวา, มีการเคลื่อนไหว, มีเสียง, มีการโต้ตอบกันของสิ่งต่างๆ ซึ่งสิ่งต่างๆ เหล่านี้เป็นที่คุ้นเคยกันดีอยู่แล้วในโลกแห่งความเป็นจริง

2.12.2 ประวัติความเป็นมาของ VRML

VRML เกิดมาจากความสนใจในเทคโนโลยีที่เกี่ยวกับ Virtual Reality และภาพ 3 มิติ ในปี 1994 จากงานประชุม ณ กรุงเจนีวา ประเทศสวิสเซอร์แลนด์และหลังจากการประชุมครั้งนั้นได้มีการจัดสร้างจดหมายเวียน เพื่อเสาะหาสมาชิกเพื่อช่วยพัฒนา VRML ซึ่งมีสมาชิกกว่าพันคนเข้าร่วมแลกเปลี่ยนความคิดเห็นกันเพื่อสร้าง VRML หลังจากนั้นประมาณ 6 เดือน Mark Pesce จาก Labyrinth Group ได้เปิดเผยฉบับร่างของ VRML เป็นครั้งแรกและเริ่มทำการค้นหาเทคโนโลยีที่จะสามารถนำมาประยุกต์ทำงานร่วมกันกับ VRML ซึ่งก็ได้ผลเป็นที่น่าพอใจ จากการค้นพบรูปแบบแอสกีไฟล์ โดยบริษัท Silicon Graphics ที่มีความสามารถในการจัดการงานเกี่ยวกับภาพ 3 มิติที่ดี จากหลักการพื้นฐานของ VRML นั้น Gavin Bell บริษัท Silicon Graphics ได้นำไฟล์ที่ค้นพบมาดัดแปลงให้เข้ากับ VRML เพื่อเริ่มนำเอา VRML เข้าสู่ตลาด

2.12.3 โปรแกรมที่ใช้แสดงผล (VRML Browsers)

ถ้า VRML คือ ภาษา แล้ว VRML Browser คือ ตัวแปลภาษา เมื่อระบุชื่อ URL ของ VRML เข้าไปยังเว็บเบราเซอร์ เมื่อเว็บเบราเซอร์พบแฟ้มข้อมูลที่มีส่วนขยายที่เป็น .wrl เว็บเบราเซอร์จะค้นหาข้อมูลในไฟล์ระบบของเว็บเบราเซอร์ เพื่อตรวจสอบว่าได้มีการกำหนดแอปพลิเคชั่นที่จะใช้สำหรับแฟ้มข้อมูลที่มีส่วนขยายที่เป็น .wrl หรือไม่ ถ้ามีการกำหนดอย่างถูกต้องแล้ว เว็บเบราเซอร์จะเรียกใช้โปรแกรม VRML Browser เมื่อ VRML Browser ถูกเรียกใช้ โปรแกรมจะเริ่มทำการตีความ แล้วทำการแปลความ VRML ไปเป็นภาพแบบกราฟฟิก ส่วนการกำหนดให้โปรแกรมเว็บเบราเซอร์เรียกใช้ VRML Browser จะทำได้โดยการกำหนด MIME type ในโปรแกรมเว็บเบราเซอร์ให้รู้จักส่วนขยาย (File Extension) ของ VRML โดยใส่ส่วนขยาย .wrl .wrz เข้าไปในช่องของ File

Extension และกำหนดให้เว็บเบราเซอร์ทราบว่าจะไปเรียกใช้โปรแกรม VRML Browser ใด โดยการกำหนดชื่อของแฟ้มข้อมูลและไดเรคทอรีของโปรแกรม VRML Browser ที่ต้องการ

2.12.4 MIME Content Types

MIME (Multipurpose Internet Mail Extensions) เป็นมาตรฐานซอฟท์แวร์ที่กำหนดโดย Internet community เพื่อบอกถึงชนิดของเนื้อหาในไฟล์ที่ส่งผ่านทางอินเตอร์เน็ต โดยเว็บเบราเซอร์จะเข้าใจ MIME content type และใช้ในการแสดงข้อมูลในหน้าจอของเว็บเบราเซอร์ [1] เช่น ถ้า MIME content type ระบุว่าเป็นไฟล์ชนิด HTML text ดังนั้นเว็บเบราเซอร์จะจัดรูปแบบ HTML text และแสดงออกมาทางหน้าจอของเว็บเบราเซอร์ ในลักษณะเดียวกัน ถ้า MIME content type ระบุว่าเป็นไฟล์ชนิดที่เป็น audio ดังนั้นเว็บเบราเซอร์จะเล่นเสียงออกมาผ่านทางลำโพงของคอมพิวเตอร์

MIME content type แบ่งออกเป็น 2 ส่วน โดยใช้ slash (/) คั่น ในส่วนแรกบอกถึง general type ของเนื้อหา เช่น text, audio หรือ video ในส่วนที่สองจะบอกถึง subtype ของเนื้อหา ตารางที่ 2.1 แสดงชนิดของ MIME content type

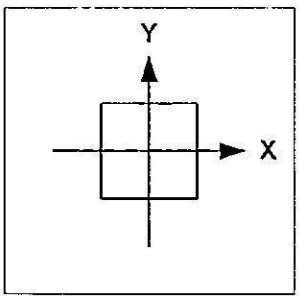
ตารางที่ 2.1 แสดง MIME Content Types

[illegible]	[illegible]
text/plain	Plain, unformatted text, such as in e-mail message
text/html	HTML text, such as a Web page
image/gif	GIF image, such as a picture used on a Web page
video/mpeg	MPEG-encoded video

โดย MIME content type ของ VRML คือ model/vrml

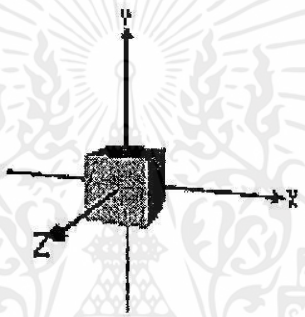
2.12.5 พื้นที่บนโลกความจริงเสมือน

บนโลกของ VRML จะใช้พื้นที่ที่เป็นลักษณะแบบ 3 มิติที่มีแกน X จากซ้ายไปขวา แกน Y จากล่างขึ้นบน และแกน Z จากไกลมาใกล้ ซึ่งแตกต่างจากภาพแบบ 2 มิติที่มีเพียงแกน X และแกน Y ดังรูปที่ 2.14 ที่แสดงถึงการวาดภาพสี่เหลี่ยมแบบ 2 มิติ ที่มีพิกัดอยู่ทั้งหมด 4 ตำแหน่ง แต่ละตำแหน่งจะถูกกำหนดค่าเป็น (x,y) จุดตัดของแกน X และแกน Y ที่พิกัด 0,0 จะถูกเรียกว่า จุดกำเนิด (Origin)



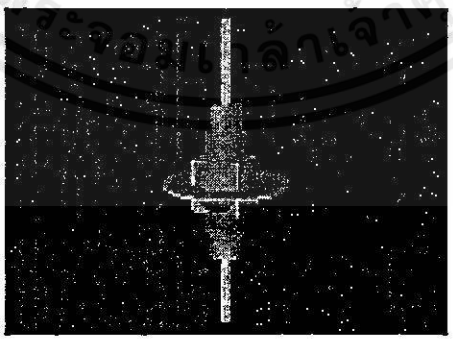
รูปที่ 2.14 แสดงภาพสี่เหลี่ยมแบบ 2 มิติ

ซึ่งในรูปภาพแบบ 3 มิติ จุดกำเนิดจะอยู่ที่พิกัด 0,0,0 ภาพวัตถุใดๆ ที่ VRML สร้างขึ้นจะถูกสร้างให้อยู่ที่จุดกึ่งกลางของจุดกำเนิด ดังภาพวัตถุสี่เหลี่ยมที่เป็นภาพแบบ 3 มิติ ดังรูปที่ 2.15



รูปที่ 2.15 แสดงภาพความสัมพันธ์ของวัตถุกับจุดกำเนิด

ในกรณีที่สร้างกลุ่มของวัตถุที่มีวัตถุมากกว่าหนึ่งวัตถุ วัตถุทั้งหมดจะถูกสร้างจากจุดกำเนิดเดียวกันทั้งสิ้น ดังรูปที่ 2.16 ที่แสดงถึงการสร้างกลุ่มของวัตถุที่ประกอบไปด้วยวัตถุทรงกลม วัตถุทรงกระบอก และวัตถุทรงลูกบาศก์



รูปที่ 2.16 แสดงวัตถุที่สร้างจากหลายรูปทรง

2.12.6 รูปแบบของไฟล์ VRML2.0

ไฟล์ VRML มีลักษณะเป็นเท็กซ์ไฟล์ ดังนั้นในการสร้างวัตถุหรือโลกเสมือนโดยใช้ VRML สามารถทำได้โดยใช้เอดิเตอร์ธรรมดาที่สามารถสร้างเท็กซ์ไฟล์ได้ เช่น Windows Notepad

หรือ UNIX vi เป็นเครื่องมือในการสร้างไฟล์ VRML ซึ่งลักษณะของไฟล์ VRML เป็นดังตัวอย่างที่ 2.1

```
#VRML V2.0 utf8
#This is an object
Shape {
    appearance Appearance {
        material Material {
            diffuseColor 1 0 0
        }
    }
    geometry Cone { }
}
```

ตัวอย่างที่ 2.1 รูปแบบของไฟล์ VRML

จากตัวอย่างที่ 2.1 เป็นไฟล์ VRML ที่สร้างวัตถุทรงกรวย 3 มิติ มีสีแดง ส่วนประกอบต่างๆ มีดังนี้

"VRML V2.0 utf8" หมายถึง ส่วนที่เรียกว่า ส่วนหัวของไฟล์ (Header file) ทุกไฟล์ VRML ต้องขึ้นต้นด้วยบรรทัดนี้

"V2.0" คือ เวอร์ชันของ VRML

"utf8" คือ วิธีเข้ารหัสไฟล์ VRML ในที่นี้หมายถึง เข้ารหัสตัวอักษรแบบ UTF8 เป็นเท็กซ์ไฟล์ VRML

"#" เป็นส่วนอธิบายความ (Comment) ที่เติมเข้าไป โดยไม่มีผลในการแปลความของไฟล์ VRML

ส่วนที่อยู่ใต้ส่วนหัวของไฟล์ลงมา คือ ส่วนที่แสดงโหนดต่างๆ และฟิลด์รวมถึงค่าที่ใช้ในการกำหนดคุณสมบัติของวัตถุนั้นๆ

2.12.7 โหนดและฟิลด์

VRML จะมีส่วนที่คล้ายคลึงกับภาษาซี คือมีการใช้เครื่องหมาย { และ } เพื่อกำหนดบล็อคของข้อมูลที่มีความเกี่ยวข้องกับ VRML โหนดในภาษา VRML จะมีคุณสมบัติและหน้าที่อย่างใดอย่างหนึ่ง ซึ่งสามารถสังเกตดูได้จากชื่อของโหนด อันจะเป็นตัวบอกคุณสมบัติและหน้าที่ของโหนดนั้นๆ ภายในโหนดแต่ละโหนดจะประกอบด้วยฟิลด์ต่างๆ ซึ่งฟิลด์นี้จะเป็นตัวกำหนดค่าพารามิเตอร์ของโหนดนั้นๆ บางโหนดที่เป็นกลุ่มของโหนดหรือ Group Node ค่าที่อยู่ภายในโหนดนั้นก็คือ โหนดลูกหรือ Children Node นั่นเอง ตัวอย่างเช่น โหนด Cone สามารถกำหนดได้ดังนี้

```
Cone {
    field SFFloat bottomRadius      1
    field SFFloat height            2
    field SFFloat side              true
    field SFFloat bottom            true
    }
```

ตัวอย่างที่ 2.2 รูปแบบของโหนด Cone

จากตัวอย่างที่ 2.2 เป็นโหนดชื่อ Cone คือ โหนดที่ทำหน้าที่สร้างรูปกรวย

"Cone" คือ ชื่อโหนด

"field" คือ ตัวกำหนดคลาส ซึ่งใน VRML จะมีอยู่ 4 คลาส คือ evenIn, eventOut, field และ exposedField

"SFFloat,SFBool" คือ ชนิดของข้อมูล ได้แก่ ชนิดตัวเลขทศนิยมและชนิดบูลีน ตามลำดับ

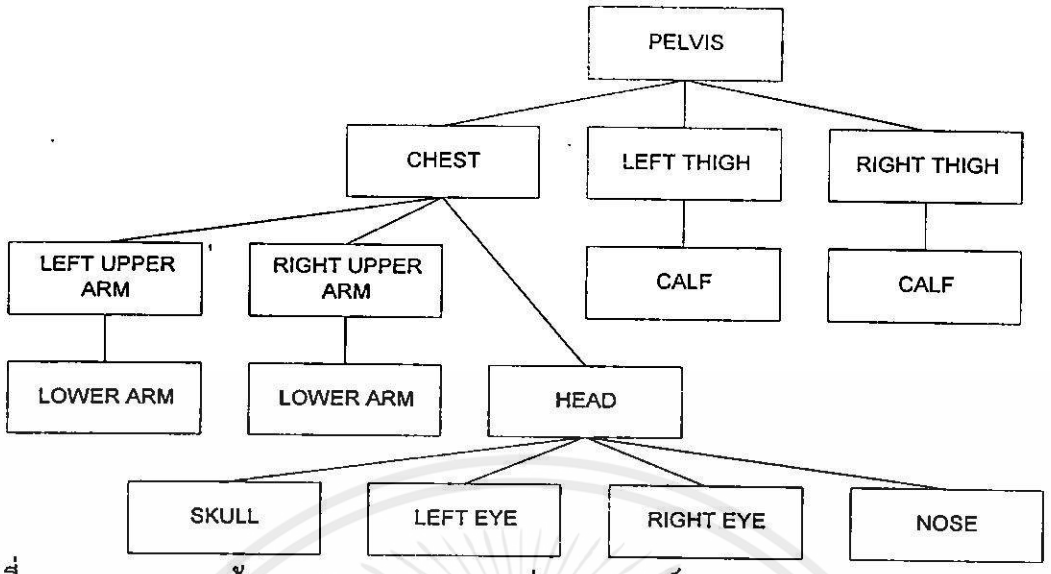
"bottomRadius,height,side,bottom" คือ ชื่อของฟิลด์ข้อมูล

ข้อมูลทุกๆ ฟิลด์ จะมีค่าเริ่มต้นอยู่แล้ว ถ้าไม่มีการกำหนดค่าให้ก็สามารถทำงานได้

ในแต่ละโหนด ผู้ใช้สามารถกำหนดชื่อของโหนดได้ เพื่อใช้ในการอ้างอิง โดยใช้คำสั่ง DEF (Define) นำหน้าโหนดที่ต้องการจะกำหนดชื่อ เช่นDEF MyFirst_Cone Cone{..}

2.12.8 Object, Hierarchy และ Separator Node

ในโลกของ VRML ข้อมูลที่เกี่ยวข้องกับวัตถุนั้นจะถูกเก็บไว้ใน Separator หรืออีกนัยหนึ่ง Separator จะหมายถึงที่บรรจุของวัตถุ แต่บางครั้ง Separator อาจจะถูกใช้เพื่อการจัดกลุ่มของวัตถุเท่านั้น ส่วนการกำหนดออบเจ็กต์ให้เป็น Object Hierarchy ใน VRML คือการกำหนดวัตถุหนึ่งให้ไปอยู่ในอีกวัตถุหนึ่งหรือการกำหนดให้ Separator Node มี Children Node และแต่ละ Children Node สามารถมี Children ต่อไปได้อีก ถ้ายกตัวอย่างการทำ Object Hierarchy จากร่างกายมนุษย์ โดยใช้โครงสร้างแบบต้นไม้ (Tree diagram) จะได้โครงสร้างดังรูปที่ 2.17



รูปที่ 2.17 แสดงการสร้าง Object Hierarchy จากร่างกายมนุษย์

และถ้านำไปสร้างเป็น VRML จะได้ซอสโคดดังต่อไปนี้

```
#VRML V2.0 utf8
DEF Body Separator {
    DEF Pelvis Separator {
        DEF Chest Separator {
            DEF LeftUpperArm Separator {
                DEF LowerArm Separator { }
            }
            DEF RightUpperArm Separator {
                DEF LowerArm Separator { }
            }
            DEF Head Separator {
                DEF Skull Separator { }
                DEF LeftEye Separator { }
                DEF RightEye Separator { }
                DEF Nose Separator { }
            }
        DEF LeftThigh Separator {
            DEF Calf Separator { }
        }
        DEF RightThigh Separator {
            DEF Calf Separator { }
        }
    }
}
```


ตัวอย่างที่ 2.3 รูปแบบ Object Hierarchy จากร่างกายมนุษย์

Separator นอกจากจะใช้ในการจัดกลุ่มของวัตถุเพื่อใช้ในการจัดทำ Object Hierarchy แล้ว Separator ยังสามารถจัดเก็บข้อมูลเกี่ยวกับรูปร่างของวัตถุ (Shape), คุณสมบัติของพื้นผิว (Surface Property), ตำแหน่ง (Location) และขนาด (Size) ของวัตถุที่มีความสัมพันธ์กับวัตถุที่อยู่เหนือขึ้นไป (Parent Object) ดังรูปแบบคำสั่งต่อไปนี้

```
Separator {
    Transform
    surface properties
    shapes
    children
}
```

ถ้าคุณสมบัติของ Surface หรือคุณสมบัติใดๆ ไม่ได้ถูกกำหนดไว้ คุณสมบัติดังกล่าวจะสืบทอด (Inherit) คุณสมบัติของวัตถุที่อยู่เหนือขึ้นไป (Parent Object) ซึ่งเป็นคุณสมบัติของ VRML

Transforms เป็นคำสั่งที่ใช้สำหรับเปลี่ยนตำแหน่งวัตถุ (Scaling), การหมุนของวัตถุ (Rotation) และการเคลื่อนย้ายวัตถุ (Transformation) ดังตัวอย่างต่อไปนี้

```
Transform {
    center x y z
    scalingFactor x y z
    scaleOrientation x y z a
    rotation x y z angle
    translation x y z
}
```

2.12.9 ชนิดของฟิลด์ข้อมูล (Field Data Types)

ชนิดของฟิลด์ข้อมูล แบ่งเป็นชนิดของฟิลด์ที่มีค่าเดียว (Single Value Field) ซึ่งชื่อจะขึ้นต้นด้วย "SF" และชนิดของฟิลด์ที่มีหลายค่าหรือเป็นชุดของฟิลด์ (Multiple Value Field) ซึ่งชื่อจะขึ้นต้นด้วย "MF" ตัวอย่างของชนิดของฟิลด์ของข้อมูล เช่น

SFBool เป็น Single Value Field มีค่าเป็น TRUE หรือ FALSE

SFFLOAT เป็น Single Value Field แทนตัวเลขทศนิยม

MFFLOAT เป็น Multiple Value Field เป็นชุดของฟิลด์ข้อมูลที่เป็นเลขทศนิยมหรือจะบอกว่าเป็นชุดของชนิดฟิลด์ SFFloat ก็ได้

2.12.10 องค์ประกอบพื้นฐานของวัตถุ

ตามปกติโลกเสมือนเกิดจากวัตถุต่างๆ มาวางประกอบกันเป็นฉากของโลกเสมือน วัตถุต่างๆ นั้นจะมีองค์ประกอบพื้นฐานหลักๆ ดังนี้

- File header
- Shape node
- Geometry node
- Appearance node
- Grouping node

2.12.10.1 File header (ส่วนหัวของไฟล์)

ดังที่กล่าวไปแล้วขั้นต้นว่า บรรทัดแรกของไฟล์ VRML2.0 ทุกไฟล์ต้องขึ้นต้นด้วยส่วนหัวดังต่อไปนี้เสมอ #VRML V2.0 utf8

2.12.10.2 โหนด Shape

โหนด Shape คือ โหนดพื้นฐานซึ่งบรรจุเอาโหนดที่แสดงวัตถุเรขาคณิต (Geometry) และโหนดที่กำหนดรายละเอียดการแสดงผลต่างๆ ของวัตถุ เช่น สี, คุณสมบัติพื้นผิว, การปะผิว เป็นต้น ตัวอย่างที่ 2.4

```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
        material Material { }
    }
    geometry NULL
}
```

ตัวอย่างที่ 2.4 โหนด Shape

2.12.10.3 โหนด Geometry

โหนด Geometry คือ โหนดที่มีไว้สำหรับกำหนดรูปทรงของวัตถุ ในภาษา VRML มีโหนดรูปทรงพื้นฐานไว้ให้ใช้งานดังนี้ คือ ทรงกลม (Sphere), ทรงกระบอก (Cylinder), ทรงลูกบาศก์ (Cube) และทรงกรวยกลม (Cone) ดังรายละเอียดการกำหนดรูปร่างวัตถุต่อไปนี้

- Sphere (รูปทรงกลม) มีรูปแบบการใช้ดังนี้

```
Sphere {
  radius รัศมี
}
```

- Cylinder (รูปทรงกระบอก) มีรูปแบบการใช้ดังนี้

```
Cylinder {
  radius รัศมี
  height ความสูง
  side แสดงผลด้านข้างหรือไม่
  top แสดงผลด้านบนหรือไม่
  bottom แสดงผลด้านล่างหรือไม่
}
```

- Cube (รูปทรงลูกบาศก์) มีรูปแบบการใช้ดังนี้

```
Cube {
  width ความกว้าง
  height ความสูง
  depth ความลึก
}
```

- Cone (รูปทรงกรวย) มีรูปแบบการใช้ดังนี้

```
Cone {
  bottomRadius รัศมีของพื้นกรวย
  height ความสูง
}
```

นอกจากรูปร่างของวัตถุพื้นฐานทั้ง 4 แบบ (Sphere, Cylinder, Cube, Cone) VRML ยังใช้รูปร่างวัตถุในรูปแบบอื่นด้วย

- IndexedFaceSet เป็นวิธีการอธิบายวัตถุโดยการใช้มุมต่างๆ (Vertices) มารวมกันได้ ยกตัวอย่างเช่น ถ้าต้องการสร้างวัตถุที่มีรูปร่างเหมือนกับปิรามิด ดังนั้นรูปร่างของวัตถุดังกล่าวจะประกอบด้วยมุมทั้งหมด 5 มุมด้วยกัน นั่นก็คือเป็นโหนดสำหรับปิดพื้นผิวตามแนวจุดพิกัดต่างๆ นั่นเอง
- IndexedLineSet เป็นวิธีการอธิบายวัตถุโดยใช้เส้น เนื่องจากวัตถุบางประเภทสามารถใช้เส้นในการอธิบายได้ง่ายกว่าการอธิบายด้านทั้ง 4 ด้านของวัตถุ

- PointSet ใช้สำหรับวัตถุที่ต้องการให้เป็นแค่จุดเล็กๆ ยกตัวอย่างเช่น ถ้าต้องการสร้างวัตถุที่เป็นดาว ผู้ใช้ไม่จำเป็นที่จะต้องใช้วัตถุทรงกลม (Sphere) ที่มีขนาดเล็กๆ

ตัวอย่างของการกำหนดโหนด Cylinder ดังนี้

```
#VRML V2.0 utf8
Shape {
    appearance Appearance {
        material Material { }
    }
    geometry Cylinder { }
}
```

ตัวอย่างที่ 2.5 โหนด Cylinder ที่ไม่มีการกำหนดค่าเริ่มต้น

จากตัวอย่างที่ 2.5 เป็นโหนด Cylinder จะทำหน้าที่สร้างวัตถุรูปทรงกระบอก ซึ่งจะมีคุณสมบัติของรูปทรงกระบอกตามค่าเริ่มต้น เพราะไม่มีการกำหนดค่าคุณสมบัติให้ สังเกตได้จาก Cylinder{ } ภายในเครื่องหมาย { } จะเป็นที่กำหนดค่าคุณสมบัติต่างๆ ถ้าไม่ใส่จะใช้ค่าเริ่มต้นที่ทางภาษา VRML กำหนดไว้ ดังดูตัวอย่างของโหนด Cylinder ที่มีการกำหนดค่าคุณสมบัติของโหนดด้วย

```
geometry Cylinder {
    radius      3
    height      6
    side        TRUE
    top         TRUE
    bottom      TRUE
}
```

ตัวอย่างที่ 2.6 โหนด Cylinder ที่มีการกำหนดค่าเริ่มต้น

2.12.10.4 โหนด appearance

โหนด appearance เป็นโหนดที่รวบรวมโหนดที่ใช้ในการกำหนดคุณลักษณะพื้นผิวและแสดงผลของวัตถุ เช่น สีของพื้นผิว, ความเรียบหรือความขุรขระของพื้นผิว, ความสว่างของพื้นผิว เป็นต้น โหนดที่กำหนดคุณลักษณะเหล่านี้ที่ใช้กันบ่อยๆ ได้แก่ โหนด Material และ Texture ตัวอย่างการกำหนดวัตถุให้มีสีม่วงและมีความสว่างปานกลาง สามารถทำได้ดังนี้

```
Material {
    diffuseColor        .5 0 .5
    shininess           .5
}
```

ตัวอย่างที่ 2.7 โหนด Material

ฟิลด์ diffuseColor มีชนิดของข้อมูลเป็น SFColor คือ เป็นฟิลด์ที่กำหนดค่าสี โดยค่าที่ใช้กำหนดมี 3 ค่า เรียงกัน คือ สีแดง สีเขียว และสีน้ำเงินตามลำดับ โดยค่าที่เป็นไปได้จะเริ่มจาก 0-1.0 จากตัวอย่างที่ 2.7 ได้กำหนดไว้เป็น .5 0 .5 หมายถึง ค่าสีแดงเป็น 50% สีเขียว 0% สีน้ำเงิน 50% ดังนั้นสีที่ผสมออกมาจึงปรากฏเป็นสีม่วง

ฟิลด์ shininess จะมีค่าได้ตั้งแต่ 0-1.0 ถ้าค่าน้อยจะทำให้วัตถุดูนุ่มนวล ถ้าค่ามากๆ จะทำให้ดูคมชัดมากขึ้น

2.12.10.5 โหนด Grouping

โหนด Grouping เป็นโหนดที่บรรจุเอาโหนดอื่นๆ เป็นโหนดลูก โดยที่โหนด Grouping เป็นโหนดพ่อแม่ จุดประสงค์หลักของโหนดประเภทนี้ คือ การรวมเอาโหนดต่างๆ หรือวัตถุต่างๆ ไว้เป็นกลุ่ม เพื่อจุดประสงค์ใดอย่างหนึ่งแล้วแต่ว่าเรียกใช้โหนด Grouping ใด ในภาษา VRML2.0 มีโหนด Grouping อยู่หลายโหนด เช่น โหนด Group, โหนด Transform, โหนด Anchor และโหนด Inline เป็นต้น ตัวอย่างรูปแบบการกำหนดของโหนด Group ซึ่งโหนดนี้มีหน้าที่รวมเอาวัตถุต่างๆ ที่อยู่ในขอบเขตที่กำหนดให้อยู่ในกลุ่มเดียวกัน

```
Group {
    eventIn         MFNode      addChildren
    eventIn         MFNode      removeChildren
    exposedField    MFNode      children        [ ]
    field           SFVec3f     bboxCenter      0 0 0
    field           SFVec3f     bboxSize        -1 -1 -1
}
```

ตัวอย่างที่ 2.8 การกำหนดโหนด Group

จากตัวอย่างที่ 2.8

ฟิลด์ children เป็นฟิลด์ที่รวบรวมโหนดลูก

ฟิลด์ bboxCenter และ bboxSize เป็นฟิลด์ที่กำหนดจุดกึ่งกลางและขนาดของขอบเขตของโหนด Group ตามลำดับ

ฟิลด์ addChildren และ removeChildren เป็นฟิลด์ที่แสดงเหตุการณ์ การเพิ่มโหนดลูก และการกำจัดโหนดลูกออก ตามลำดับ

2.12.11 การกำหนดตำแหน่งและทิศทางของวัตถุ

โลกเสมือนเกิดจากวัตถุต่างๆ ถูกจัดวางในตำแหน่งต่างๆ และในทิศทางที่เหมาะสม โหนดที่มีหน้าที่ในการกำหนดตำแหน่งและลักษณะทิศทางของวัตถุ คือ โหนด Transform โหนดนี้ทำให้สามารถกำหนดหรือเปลี่ยนแปลงตำแหน่งที่อยู่ของวัตถุโดยใช้ระบบจุดพิกัด ซึ่งสามารถเปลี่ยนแปลงทิศทางของวัตถุให้มีมุมมองที่ต่างออกไปได้ โหนด Transform เป็นโหนด Grouping ซึ่งโหนดลูกที่อยู่ภายในมักจะเป็นโหนด Shape ซึ่งเป็นโหนดที่แสดงวัตถุ ดังนั้นโหนดลูกทั้งหมดที่อยู่ภายในโหนด Transform จะถูกกำหนดคุณสมบัติตามโหนด Transform นั้น ตัวอย่างที่ 2.9 แสดงรูปแบบการกำหนดโหนด Transform และตัวอย่างที่ 2.10 แสดงการใช้โหนด Transform เพื่อกำหนดตำแหน่งของรูปทรงกรวยสีแดงให้อยู่ที่ตำแหน่ง x=4, y=5, z=0

```
Transform {
    eventIn       MFNode      addChildren
    eventIn       MFNode      removeChildren
    exposedField  SFVec3f     children        0 0 0
    exposedField  MFNode      children        [ ]
    exposedField  SFRotation  children        0 0 1 0
    exposedField  SFVec3f     scale           1 1 1
    exposedField  SFRotation  scaleOrientation 0 0 1 0
    exposedField  SFVec3f     translation     0 0 0
    field         SFVec3f     bboxCenter      0 0 0
    field         SFVec3f     bboxSize        -1 -1 -1
}
```

ตัวอย่างที่ 2.9 การกำหนดโหนด Transform

```
Transform {
    tanslation 4 5 0
    children Shape {
        appearance Appearance {
            material Material {
                diffuseColor 1 0 0
            }
        }
        geometry Cone { }
    }
}
```

ตัวอย่างที่ 2.10 การใช้โหนด Transform

2.12.12 การทำภาพเคลื่อนไหว

พื้นฐานของการเคลื่อนไหวของวัตถุ คือ การเปลี่ยนค่าคุณสมบัติของวัตถุ เช่น เปลี่ยนตำแหน่ง, เปลี่ยนมุมมอง, เปลี่ยนสี เป็นต้น ซึ่งจะควบคุมการเปลี่ยนแปลงหรือการเคลื่อนไหวของวัตถุในโลกเสมือนได้อย่างไรนั้นสามารถทำได้โดยใช้เหตุการณ์ (Event) ซึ่งในโลกเสมือน เหตุการณ์จะเป็นตัวบอกว่าเกิดอะไรขึ้น และเหตุการณ์ที่เกิดขึ้นส่งผลกระทบต่อวัตถุใดหรือเหตุการณ์ใดบ้าง

โหนดต่างๆ บางโหนดสามารถสร้างเหตุการณ์ได้ บางโหนดสามารถรับเหตุการณ์ได้และบางโหนดสามารถทั้งสร้างและรับเหตุการณ์ได้ จากตัวอย่างที่ 2.9 แสดงรูปแบบการกำหนดโหนด Transform ที่กล่าวมาแล้วนั้น จะเห็นว่าในคอลัมน์แรกที่เป็นตัวกำหนดคลาสของฟิลด์มีการกำหนดเป็น eventIn, exposedField, และ field ภาษา VRML มีคลาสของชนิดของข้อมูล อยู่ 4 คลาส คือ

1. eventIn เหตุการณ์รับเข้ามา หรือโหนดใดๆ สามารถส่งค่าผ่านเข้ามาทางฟิลด์นี้ได้
2. eventOut เหตุการณ์ส่งออกไป หรือโหนดใดๆ สามารถรับค่าที่ถูกส่งออกไปจากฟิลด์นี้
3. field เป็นฟิลด์ส่วนตัว โหนดอื่นไม่สามารถเข้าถึงได้
4. exposedField โหนดใดๆ สามารถส่งค่าผ่านหรือรับค่าออกไปจากฟิลด์นี้

การทำให้วัตถุสามารถเคลื่อนที่ได้ก็โดยการเปลี่ยนค่าฟิลด์ translation ในโหนด Transform ซึ่งเป็นคลาส exposeField แสดงว่าสามารถรับและส่งเหตุการณ์ได้ ซึ่งต้องทำการส่งเหตุการณ์เข้าไปเพื่อเปลี่ยนค่า translation ใหม่ วัตถุนั้นก็จะย้ายตำแหน่งไปตามค่าที่เปลี่ยน และถ้าใช้เวลาเข้ามาควบคุมให้วัตถุย้ายตำแหน่งไปมาก็จะเป็นการทำให้วัตถุเคลื่อนที่ได้ เหตุการณ์ที่ทำให้เกิดผลกระทบต่อวัตถุต่างๆ ส่วนใหญ่จะเริ่มจากเหตุการณ์ภายนอก คือ เหตุการณ์ที่ผู้ใช้ก่อขึ้น เช่น การคลิกที่วัตถุ, การวางลูกศรของเมาส์ไว้บนวัตถุ เป็นต้น เหตุการณ์เหล่านี้จะเป็นตัวกระตุ้นให้เกิดการเปลี่ยนแปลงตามที่ได้กำหนด หรือไปกระตุ้นให้เกิดเหตุการณ์อื่นๆ ต่อไป เหตุการณ์ต่างๆ ที่เกิดขึ้นจะส่งผลต่อวัตถุต่างๆ ตามที่กำหนดไว้เป็นลูกโซ่ต่อกันไป โหนดที่สามารถสร้างเหตุการณ์ได้มี

หลายโหนด เช่น โหนด TimeSensor, โหนด TouchSensor, โหนด ProximitySensor เป็นต้น ซึ่งแต่ละโหนดสามารถสร้างเหตุการณ์และตรวจจับเหตุการณ์ได้ในสถานะการณ์ต่างๆ กัน เมื่อทราบถึงโหนดที่สร้างเหตุการณ์และโหนดที่รับเหตุการณ์แล้ว การส่งผ่านเหตุการณ์จากโหนดหนึ่งไปยังอีกโหนดหนึ่งจะกระทำผ่านเส้นทางที่เรียกว่า ROUTE โดยที่ ROUTE จะเป็นตัวกำหนดเส้นทางของเหตุการณ์ว่าจากโหนดใดไปยังโหนดใด เหตุการณ์ที่ไม่ได้กำหนด ROUTE จะไม่ถูกพิจารณา

รูปแบบการกำหนด ROUTE คือ

ROUTE node1.eventOutName_changed TO node2.set_eventInName

node1, node2 เป็นชื่อของโหนดที่กำหนดโดยใช้คำสั่ง DEF ซึ่งสามารถเชื่อมเส้นทางจากฟิลด์ที่สามารถเป็น eventOut ไปยังฟิลด์ที่สามารถเป็น eventIn เท่านั้นและฟิลด์ที่เชื่อมต่อกันทาง ROUTE ต้องเป็นชนิดเดียวกันด้วย ซึ่งสามารถใช้ ROUTE เป็นตัวเชื่อมต่อเหตุการณ์ต่างๆ ให้เกิดขึ้นอย่างต่อเนื่องเป็นลูกโซ่ได้

โหนดที่ทำหน้าที่กำหนดค่าที่จะทำให้คุณสมบัติของโหนดที่ถูกเหตุการณ์มากระทำมีค่าคุณสมบัติเปลี่ยนไป คือ โหนด Interpolator โหนดนี้มีหลายชนิดแล้วแต่ค่าที่จะส่งไปเปลี่ยน เช่น โหนด PositionInterpolator ทำหน้าที่ส่งค่าตำแหน่งของวัตถุ, โหนด OrientationInterpolator ทำหน้าที่ส่งค่าที่ใช้ในการหมุนวัตถุ, โหนด ColorInterpolator ทำหน้าที่ส่งค่าที่ใช้ในการเปลี่ยนสีของวัตถุ เป็นต้น

ตัวอย่าง VRML ที่ใช้ TimeSensornode เพื่อใช้ในการควบคุมการทำภาพเคลื่อนไหว (Animation) โดยการกำหนดเวลาเริ่ม (StartTime) และเวลาสิ้นสุด (StopTime) เมื่อเวลากำลังเปลี่ยนไปจะเกิดเหตุการณ์ (Event) ที่เรียกว่า fraction_changed ที่มีผลต่อ PositionInterpolator node หรือ OrientationInterpolator node ซึ่งเป็น node ที่เก็บค่าเศษส่วนของเวลาไว้ใน Key field และค่าที่เกี่ยวข้องไว้ใน KeyValue เพื่อสร้างค่าที่เป็นผลลัพธ์ในแต่ละเศษส่วนของเวลา (Value_changed EventOut) แล้ว จะส่งค่าดังกล่าวไปยัง Transform node เพื่อเปลี่ยนแปลงค่าตำแหน่งพิกัดหรือค่าใดๆ ของ Transform node เพื่อให้วัตถุมีการเคลื่อนไหวตามค่าดังกล่าว ดังแสดงในตัวอย่างที่ 2.11

```
#VRML V2.0
Group {
    children [
    #Pulsing ball
        DEF Ball Transform {
            children Shape {
                appearance Appearance {
                    material Material { }
                }
                geometry Sphere { }
            }
        },
    #Animation clock
        DEF Clock TimeSensor {
            cycleInterval 2.0
            loop TRUE
        },
    #Animation path
        DEF BallPath PositionInterpolator {
            key [ 0.0, 0.20, 0.65, 1.0]
            keyValue [
                1.0 1.0 1.0,
                1.5 1.5 1.5,
                1.1 1.1 1.1,
                1.0 1.0 1.0,]
        }
    ]
}
ROUTE Clock.fraction_changed TO BallPath.set_fraction
ROUTE BallPath.value_changed TO Ball.set_scale
```

ตัวอย่างที่ 2.11 การทำให้วัตถุเคลื่อนที่อย่างง่าย

จากตัวอย่างที่ 2.11 เป็นการสร้างภาพเคลื่อนไหวของวัตถุทรงกลมที่มีการย่อและขยายขนาดของวัตถุได้ ซึ่งมีขั้นตอนการทำงานดังนี้

- สร้าง Transform node ที่มีรูปร่างทรงกลมโดยให้ชื่อว่า Ball
- สร้าง TransformsSensor node ชื่อ Clock และกำหนดให้มีช่วงของเวลาคือ 2.0 วินาที โดยไม่มีการสิ้นสุด (loops forever)
- สร้าง PositionInterpolator node ชื่อ BallPath โดยมีการกำหนดค่าเศษส่วนของเวลา 4 ค่าและค่าของการเปลี่ยนแปลงที่ต้องการ 4 ค่าเช่นเดียวกัน

- สร้างเส้นทางของเหตุการณ์จาก Position Interpolator node ไปหา Transform Node

ระหว่างการทำงานจะมีการส่งค่าเศษส่วนของเวลาจาก TimeSensor node ไปหา PositionInterpolator node เพื่อหาค่าของการย่อและขยายรูป โดยจะถูกกำหนดว่าที่เวลา 0.2 วินาที ค่าของรูปจะมีขนาดใหญ่ขึ้น 1.5 เท่า และที่เวลา 0.65 วินาที รูปจะมีขนาดใหญ่ขึ้น 1.1 เท่า สุดท้ายที่เวลา 1.0 วินาที รูปจะมีขนาดเท่าเดิม แต่ละค่าจะถูกส่งไปยัง Transform node เพื่อเปลี่ยนแปลงค่าในทุกเศษส่วนวินาที

2.12.13 การโต้ตอบกับผู้ใช้

การสร้างให้โลกเสมือนจริงมีการโต้ตอบกับผู้ใช้ได้นั้น VRML จะใช้ Sensor Node ที่ประกอบไปด้วย TouchSensor, PlaneSensor, SphereSensor และ CylinderSensor เป็นตัวตรวจรับการทำงานของเมาส์ เช่น การเคลื่อนหรือการคลิก เป็นต้น โดยการกำหนดเส้นทางการทำงานว่าผลของการเคลื่อนหรือคลิกเมาส์ดังกล่าวจะมีผลต่อวัตถุอย่างไรต่อไป TouchSensor ถูกออกแบบไว้สำหรับการเคลื่อนเมาส์ผ่านวัตถุ (isOver eventOut) และการคลิกเมาส์ที่วัตถุ (isActive eventOut) ยกตัวอย่างเช่น การสร้างวัตถุทรงลูกบาศก์ขึ้นมา 1 วัตถุ ถ้าผู้ใช้เคลื่อนเมาส์ผ่านบนวัตถุดังกล่าวให้วัตถุนั้นหมุนรอบตนเอง และถ้าเคลื่อนเมาส์ออกจากวัตถุให้วัตถุนั้นหยุดหมุน ซึ่งสามารถเขียนเป็นซอสโคดได้ดังนี้

```
#VRML V2.0
Group {
    children [
    #Rotating Cube
        DEF Cube Transform {
            children Shape {
                appearance Appearance {
                    material Material { }
                }
                geometry Box { }
            }
        },
    #Sensor
        DEF Touch TouchSensor { },
    #Animation clock
        DEF Clock TimeSensor {
            enabled FALSE
            cycleInterval 4.0
            loop TRUE
        },
    #Animation path
        DEF CubePath OrientationInterpolator {
            key [0.0, 0.50, 1.0]
            keyValue [
                0.0 1.0 0.0 0.0,
                0.0 1.0 0.0 3.14,
                0.0 1.0 0.0 6.28,
            ]
        }
    ]
}
ROUTE Touch.isOver TO Clock.set_enabled
ROUTE Clock.fraction_changed TO CubePath.set_fraction
ROUTE CubePath.value_changed TO Cube.set_rotation
```

ตัวอย่างที่ 2.12 การโต้ตอบกับผู้ใช้อย่างง่าย

การทำงานในตัวอย่างที่ 2.12 จะคล้ายๆ กับตัวอย่างการย่อและขยายวัตถุแต่การกำหนดเส้นทางการทำงานให้เริ่มจาก TouchSensor node ไปมีผลต่อ TimeSensor node นั่นคือ ถ้าผู้ใช้เคลื่อนเมาส์เข้าไปบนวัตถุ isOver จะมีค่าเป็นจริง (True) จะทำให้ TimeSensor node เริ่มทำงาน แต่

ในทางตรงข้ามถ้าผู้ใช้เคลื่อนเมาส์ออกจากวัตถุ isOver จะมีค่าเป็นเท็จ (False) จะทำให้ TimeSensor node หยุดการทำงาน (จาก Enabled Field) จากนั้น TimeSensor node จะมีผลต่อ OrientationInterpolator node เพื่อส่งค่าการหมุนไปยังวัตถุดังกล่าว ส่วน PlaneSensor, PhereSensor และ Cylinder node จะมีการทำงานที่คล้ายคลึงกันแต่เปลี่ยนจากการเคลื่อนหรือคลิกเมาส์ที่วัตถุเป็นการทำงานที่เป็นลักษณะเหมือนการลากวัตถุ โดย PlaneSensor จะสามารถลากวัตถุไปในแนวของแกน xy หรือ xz แต่ PhereSensor จะหมุนวัตถุไปได้รอบทิศทางเหมือนกับการหมุนลูกบอล ส่วน Cylinder จะลากวัตถุเหมือนกับการหมุนวัตถุไปด้านซ้ายหรือขวาเท่านั้น

2.12.14 การเพิ่มรายละเอียดให้เหมือนจริง

วัตถุบางอย่างจะมีรายละเอียดของส่วนประกอบเป็นจำนวนมาก ยกตัวอย่างเช่น ต้นไม้จะประกอบด้วยใบไม้เป็นจำนวนมากหรือชั้นหนังสือจะมีหนังสือหลายเล่มเช่นกัน ด้วย ImageTexture node จะทำให้การสร้างต้นไม้นั้นไม่จำเป็นต้องสร้างใบไม้ทุกๆ ใบ โดย ImageTexture node จะระบุชื่อของแฟ้มข้อมูลรูปภาพที่เป็นไฟล์แบบ JPEG, PNG หรือ GIF ดังตัวอย่างซอสโคดต่อไปนี้ที่ใช้รูปต้นไม้ชื่อ Tree1.PNG เพื่อสร้างต้นไม้ในโลกความจริงเสมือน

```
#VRML V2.0
Group {
    children [
    #Ground
        Shape {
            appearance Appearance {
                material Material { }
            }
        geometry IndexedFaceSet {
            coord Coordinate {
                point [
                    -5.0 0.0 5.0, 5.0 0.0 5.0,
                    5.0 0.0 –5.0, -5.0 0.0 –5.0,
                ]
            }
            coordIndex [0, 1, 2, 3]
            solid FALSE
        }
    },
    #Tree face
        Shape {
            appearance Appearance {
                #No material, use emissive texturing
                texture ImageTexture {
                    url "tree1.png"
                }
            }
            geometry IndexedFaceSet {
                coord Coordinate {
                    point [
                        -1.51 0.0 0.0, 1.51 0.0 0.0,
                        1.51 3.0 0.0, -1.51 3.0 0.0,
                    ]
                }
                coordIndex [0, 1, 2, 3]
                solid FALSE
            }
        }
    ]
}
```

ตัวอย่างที่ 2.13 การสร้างต้นไม้จากไฟล์รูปภาพ

2.12.15 การกำหนดตำแหน่งของแสง

โดยปกติ VRML browser จะสร้างแสงไฟให้โดยอัตโนมัติ 1 หน่วยที่มุมมองของสายตาผู้ดูเปรียบเสมือนว่าผู้ดูสวมหมวกที่ใช้ในธุรกิจเหมืองแร่ที่มีไฟฉายอยู่บนหมวก โดย VRML สามารถเพิ่มแสงไฟเข้าไปได้อีกเพื่อให้โลกความจริงเสมือนดูเหมือนจริงมากยิ่งขึ้น โดยใช้ PointLight, DirectionalLight และ SpotLight node สนับสนุน

- PointLight จะเป็นไฟที่มีรัศมีรอบทิศทางเช่นเดียวกับแสงเทียน ผู้เขียน VRML จะต้องกำหนดตำแหน่งที่ตั้งและรัศมีของไฟ ดังตัวอย่างต่อไปนี้

```
PointLight {
    location 0.0 0.0 0.0
    radius 12.0
}
```

คือการกำหนดตำแหน่งของไฟอยู่ที่จุดกำเนิดโดยมีรัศมี 12 หน่วย

- DirectionalLight จะเป็นแสงไฟที่ส่งไปในทิศทางเดียว เช่นเดียวกับแสงของพระอาทิตย์ โดยผู้ใช้ต้องกำหนดทิศทางของแสงว่าจะให้แสงไปอยู่ในทิศทางใดในแนวแกน x, y, z ยกตัวอย่างถ้าต้องการให้ทิศทางของแสงอยู่ด้านขวาจะต้องเขียนซอสโคดดังนี้

```
DirectionalLight {
    direction 1.0 0.0 0.0
}
```

- SpotLight จะเป็นแสงที่ส่องไปในทิศทางที่ได้กำหนดไว้มีลักษณะเป็นทรงกรวย โดยผู้ใช้ต้องกำหนดทิศทางของแสงเหมือนกับ DirectionalLight แต่ต้องกำหนดขนาดของรัศมีเพิ่มด้วย

2.12.16 การกำหนดสีพื้น (Background Color)

จากตัวอย่างที่ผ่านๆ มานั้น สังเกตได้ว่าสีพื้นในทุกๆ ตัวอย่างจะเป็นสีดำทั้งสิ้น แต่ในกรณีที่ผู้เขียนต้องการจะเพิ่มสีสันของโลกความจริงเสมือนโดยการเปลี่ยนสีพื้น VRML จะใช้ Background node ในการสร้างสีพื้น โดยสีพื้นจะประกอบด้วยส่วนของฟ้า (sky) และส่วนของพื้น (ground) ตัวอย่างที่ 2.14 แสดงการสร้างสีพื้นโดยใช้เพียงส่วนของฟ้า (skycolor) มีซอสโคดดังต่อไปนี้

```
#VRML
Moreland
Background {
    skyColor [
        0.0 0.2 0.7,
        0.0 0.5 1.0,
        1.0 1.0 1.0
    ]
    skyAngle [1.309, 1.571]
}
```

ตัวอย่างที่ 2.14 การกำหนดสีพื้นเฉพาะส่วนของฟ้า

และถ้าเพิ่ม GroundColor จะได้ผลลัพธ์ดังตัวอย่างที่ 2.15

```
groundColor [
    0.1 0.10 0.0,
    0.4 0.25 0.2,
    0.6 0.60 0.6,
]
groundAngle [1.309, 1.571]
```

ตัวอย่างที่ 2.15 การกำหนดสีพื้นในส่วนของพื้น

2.12.17 การใส่เสียง

ผู้สร้างโลกความจริงเสมือนสามารถเพิ่มความสมจริงได้โดยการเพิ่มเสียงเข้าไปให้เข้ากับสภาวะแวดล้อม ยกตัวอย่างเช่น การเดินเข้าไปในสวนสนุกจะได้ยินเสียงของเครื่องเล่นต่างๆ, เสียงของผู้คนรอบข้าง เป็นต้น VRML จะใช้ Sound node เป็นตัวกำหนดแหล่งที่มาของเสียงโดยใช้ AudioClip node หรือ MovieTexture node ในการกำหนดชื่อของแฟ้มข้อมูลเสียงที่มีนามสกุลเป็น .mid, .wav หรือ .mpg และยังสามารถกำหนดความดังของเสียงได้ ยกตัวอย่างเช่น การเปิดโทรทัศน์ เมื่อผู้ดูอยู่ไกลจากโทรทัศน์จะได้ยินเสียงจากลำโพงค่อนข้างเบา แต่เมื่อผู้ดูเดินเข้าไปใกล้โทรทัศน์มากขึ้นก็จะได้ยินเสียงของโทรทัศน์ดังมากขึ้นเช่นกัน ซึ่งคุณสมบัติดังกล่าวจะช่วยให้โลกความเป็นจริงเสมือนมีความสมจริงมากขึ้น โดยใช้ minFront, minBack และ maxFront, maxBack Field ในการกำหนดความดังของเสียง ดังตัวอย่างซอสโคดที่ 2.16 ต่อไปนี้

```
#VRML V2.0
Moreland
Group {
    Children [
    #Sound emitter
        Sound {
            Source AudioClip {
                url "willow1.wav"
                loop TRUE
            }
            minFront 5.0
            minBack 5.0
            maxFront 10.0
            maxBack 10.0
        },
    #Sound emitter markers
        Inline { url "sndmark.wrl"}
    ]
}
```

ตัวอย่างที่ 2.16 วัตถุที่มีความสัมพันธ์กับเสียง

เป็นการสร้างวัตถุทรงกลมจาก "sndmark.wrl" ถ้าผู้ใช้เข้าใกล้วัตถุก็จะมีเสียงดัง ถ้าอยู่ไกลออกไปก็จะมีเสียงเบาลงและจะไม่ได้ยินเสียงถ้าอยู่ไกลจนเกินไป

2.12.18 การควบคุมรายละเอียด

ในการสร้างโลกความจริงเสมือนที่มีสภาพแวดล้อมที่มีรายละเอียดต่างๆ เป็นจำนวนมาก นั้นจะมีผลทำให้ VRML browser จะต้องใช้เวลาในการแปลความนาน การทำให้โลกความจริงเสมือนมีการโต้ตอบที่เร็วขึ้นจะทำได้โดยการจัดระดับของความละเอียดในการแสดงผลออกเป็นหลายๆ ระดับ เมื่อเริ่มเข้าสู่โลกความจริงเสมือนถ้าผู้ดูอยู่ห่างจากวัตถุ ก็ไม่จำเป็นต้องสร้างรายละเอียดของวัตถุดังกล่าวให้ครบทุกส่วนเพื่อให้การแปลความของ VRML browser ทำได้รวดเร็วขึ้น ต่อเมื่อผู้ดูเข้าใกล้วัตถุมากขึ้นจึงทำให้วัตถุดังกล่าวมีรายละเอียดมากขึ้น การควบคุมรายละเอียดนั้นจะทำได้โดย LOD (Level of Detail) node เพื่อแบ่งระดับของรายละเอียดออกเป็นส่วนๆ ดังตัวอย่างซอสโคดที่ 2.17 ต่อไปนี้

```
#VRML V2.0 utf8
Moreland
LOD {
    center 0.0 0.0 0.0
    range [7.5, 12.0]
    level [
    #High-detail
        Inline { url "torch1.wrl"},
    #Medium-detail
        Inline { url "torch2.wrl"},
    #Low-detail
        Inline { url "torch3.wrl"}
    ]
}
```

ตัวอย่างที่ 2.17 การสร้างรายละเอียดของวัตถุที่สัมพันธ์กับระยะของผู้ดู

เป็นการสร้างโคมไฟ เมื่อผู้ดูอยู่ห่างจากวัตถุจะแสดงรายละเอียดของวัตถุต่ำ แต่เมื่อผู้ดูเข้าใกล้วัตถุจะแสดงรายละเอียดของวัตถุมากขึ้น

2.12.19 การควบคุมมุมมองของสายตา

จากตัวอย่างที่ผ่านมา VRML browser แสดงภาพจากหลากหลายมุมมองแล้วแต่ผู้ดูจะเคลื่อนที่ไปในทิศทางที่ต้องการ โดย VRML สามารถจะกำหนดมุมมองไว้ล่วงหน้าได้โดยใช้ Viewpoint node เป็นตัวกำหนด แล้วให้ผู้ดูเป็นผู้เลือกมุมมองต่างๆ ที่ได้กำหนดไว้จากเมนูของ VRML browser โดยเมนูดังกล่าวจะเป็นคำอธิบายสั้นๆ ที่ผู้เขียนจัดเตรียมไว้โดยใช้ description field ดังตัวอย่างซอสโคดที่ 2.18 ต่อไปนี้

```
#VRML V2.0 utf8
Moreland
Group {
    children [
    #Viewpoints
        Viewpoint {
            description "Forward view"
            position 0.0 1.6 5.0
        },
        Viewpoint {
            description "Corner view"
            position 3.0 1.6 3.0
            orientation 0.0 1.0 0.0 0.611
        },
        Viewpoint {
            description "60.0 FOV degree corner view"
            position 3.0 1.6 3.0
            orientation 0.0 1.0 0.0 0.611
            fieldOfView 1.047
        },
        Viewpoint {
            description "90.0 FOV degree corner view"
            position 3.0 1.6 3.0
            orientation 0.0 1.0 0.0 0.611
            fieldOfView 1.57
        },
    #Navigation
        NavigationInfo {
            type "WALK"
            speed 1.0
            headlight FALSE
            avatarSize [0.5, 1.6, 0.5]
        },
    #World
        Inline {url "dungeon.wrl"}
    }
]
```

ตัวอย่างที่ 2.18 การใช้ ViewPoint node

การควบคุมมุมมองโดยใช้ ViewPoint node จากตัวอย่างนี้จะมีทั้งหมด 4 มุมมอง โดยจะแสดงภาพห้องจากมุมตรงชื่อ Forward view, ภาพห้องจากมุมมองชื่อ Corner view, ภาพห้องจากมุมมองชื่อ 60.0 FOV degree, และภาพห้องจากมุมมองชื่อ 90.0 FOV degree

2.12.20 การเชื่อมต่อกับ URL อื่น

เนื่องจาก VRML เป็นหนึ่งในเว็บเทคโนโลยีจึงสามารถเชื่อมต่อไปเว็บไซท์อื่นได้โดยใช้ Anchor Grouping node วัตถุใดๆ ที่เป็นลูก (Children) ของ anchor ถ้าผู้ดูคลิกเมาส์ที่วัตถุดังกล่าวข้อมูลที่ anchor node อ้างไปถึงจะถูกอ่านเข้ามาแปลความโดย VRML browser ดังตัวอย่างของซอสโคดที่ 2.19 ต่อไปนี้

```
#VRML
Group {
    #Anchor doors
        Anchor {
            url "dngnwrld.wrl"
            description "The Dungeon"
            children [
                shape {
                    cylinder
                }
            ]
        }
}
```

ตัวอย่างที่ 2.19 การเชื่อมโยงไปยัง URL อื่น

จากตัวอย่างที่ 2.19 เป็นการสร้างวัตถุทรงกระบอกซึ่งถ้าผู้ใช้คลิกเมาส์ที่วัตถุ VRML browser จะทำการอ่านข้อมูลจาก dngnwrld.wrl มาแปลความ ซึ่งการอ้างถึง url ดังกล่าวอาจจะเปลี่ยนเป็นการอ้างอิงถึงเว็บไซท์อื่นก็ได้

2.12.21 การใช้สคริปต์ (Scripting)

การกำหนดพฤติกรรมให้กับวัตถุโดยผ่านทางโหนด Script ซึ่งภายในโหนด Script จะมีภาษาสคริปต์บรรจุอยู่ภายใน ภาษาสคริปต์ที่สามารถใช้ได้ คือ VRMLScript, JAVA และ JavaScript

รูปแบบการกำหนดโหนด Script

```
Script {
    exposeField         MFString        url             [ ]
    field               SFBool          directOutput    FALSE
    field               SFBool          mustEvaluate    FALSE
    #and any number of :
    eventIn             eventType       eventName
    field               fieldType       fieldName       initialValue
    eventOut            eventType       eventName
}
```

ตัวอย่างที่ 2.20 การกำหนดโหนด Script

ส่วนที่เป็นภาษาสคริปต์จะอยู่ในฟิลด์ url ชนิดของ eventIn และ eventOut สามารถกำหนดได้เอง การทำงานของโหนด Script คือ โหนด Script จะทำงานเมื่อได้รับเหตุการณ์ โดยฟิลด์ mustEvaluate ต้องเป็น TRUE เบราเซอร์จะทำการส่งข้อมูลอินพุทให้กับโหนด Script เพื่อปฏิบัติตามโปรแกรมที่อยู่ในฟิลด์ url เมื่อทำงานจบครบตามคำสั่งฟิลด์ directOutput จะเป็น TRUE เพื่อให้โหนด Script ทำการส่งเหตุการณ์ไปยังโหนดอื่นที่ต้องการใช้ผลจากการทำงานของโหนด Script นี้

ตัวอย่างการใช้งานโหนด Script

```
Script {
    field SFNode nodeIn
    url "vrmlscript :
        function nodeIn(value) {
        // assume node being passed in is a Transform
        a = value.translation_changed;
        a[0] += 1;
        value.set_translation = new SFVec3f(1,0,0);
    }"
}
```

ตัวอย่างที่ 2.21 การใช้งานโหนด Script

จากตัวอย่างเป็นโหนด Script ที่มีหน้าที่ไปอ่านเอาค่าฟิลด์ translation ของโหนดอื่น แล้วทำการบวกค่า x ด้วย 1 จากนั้นส่งค่าผลลัพธ์กลับคืนไป

2.12.22 การนำกลับมาใช้ใหม่ (Reusability)

เมื่อต้องการใช้งานโหนดหรือวัตถุใดซ้ำกันบ่อยๆ หรือต้องการใช้งานหลายๆ ครั้งในโลกเสมือน VRML2.0 มีคุณสมบัติสนับสนุนการนำกลับม่าใช้ใหม่ โดยแบ่งลักษณะการใช้งานเป็น 2 ประเภท คือ

- การใช้โหนดเดิมหรือวัตถุเดิมหลายๆ ครั้ง โดยไม่มีการเปลี่ยนแปลงคุณสมบัติภายในสามารถใช้คำสั่ง DEF ในการกำหนดชื่อให้กับโหนดหรือวัตถุที่ต้องการใช้หลายครั้ง และใช้คำสั่ง USE ตามด้วยชื่อที่กำหนดไว้ เมื่อเวลาต้องการใช้งาน

```
Appearance {
    texture DEF BOWS ImageTexture { url "bows.bmp" }
}
```

ตัวอย่างที่ 2.22 แสดงการใช้ DEF

จากตัวอย่างที่ 2.22 กำหนดให้ BOWS แทนประโยค ImageTexture { url "bows.bmp" } เวลาเรียกใช้งานจะเรียกใช้ผ่านทาง USE ดังตัวอย่าง

```
Appearance {
    texture USE BOWS
}
```

ตัวอย่างที่ 2.23 แสดงการใช้ USE

การใช้ DEF และ USE นั้น จะใช้เวลาที่ไม่ต้องการเปลี่ยนแปลงอะไรกับสิ่งที่นำกลับมาใช้ใหม่

- การใช้โหนดเดิมหรือวัตถุเดิมหลายๆ ครั้ง โดยสามารถเปลี่ยนแปลงคุณสมบัติภายในได้ ซึ่งต้องทำการสร้างตัวต้นแบบ (prototyping) การทำตัวต้นแบบจะอำนวยความสะดวกในการนำกลับมาใช้ใหม่ โดยสามารถกำหนดค่าให้กับสิ่งที่จะนำกลับมาใช้ใหม่ได้ง่าย คำสั่งที่ใช้สร้างตัวต้นแบบ คือ PROTO ซึ่งมีรูปแบบการกำหนดดังตัวอย่าง

```
PROTO name [ parameters ]
{
    implementation
}
```

ตัวอย่างที่ 2.24 การกำหนด PROTO

สมมติว่าจะสร้างตัวต้นแบบสำหรับสร้างวัตถุทรงกระบอกและสามารถกำหนดสีได้ โดยให้ชื่อว่า โหนด Column โดยสามารถใช้ PROTO ได้ดังตัวอย่าง

```
PROTO Column [
    field SFColor columnColor .5 .5 .5
]
{
    Shape {
        appearance Appearance {
            material Material {
                diffuseColor IS columnColor
            }
        }
        geometry Cylinder { }
    }
}
```

ตัวอย่างที่ 2.25 การสร้างตัวต้นแบบ

จากตัวอย่างที่ 2.25 จะเห็นว่าฟิลด์ columnColor เป็นฟิลด์ที่มีหน้าที่กำหนดค่าสีให้กับตัวต้นแบบ การใช้งานตัวต้นแบบต้องทำการกำหนดค่าสีทางฟิลด์นี้ ถ้าไม่มีการกำหนดค่าสีจะเป็นค่าเริ่มต้น เช่น ถ้าจะสร้างวัตถุทรงกระบอกสีแดง สามารถเรียกใช้ตัวต้นแบบโดยกำหนดดังนี้

```
Column { columnColor 1 0 0 }
```

2.12.23 สรุปภาษาแบบจำลองเสมือน

VRML ไม่ใช่เพียงแค่รูปแบบภาษาใหม่ในการพัฒนาเว็บไซท์เท่านั้น แต่เป็นการรวบรวมเอาความคิด ความฝันและจินตนาการต่างๆ เอาไว้ เพื่อใช้ในการเปลี่ยนแปลงรูปแบบของการนำเสนอให้ใกล้เคียงกับความสมจริงมากที่สุด ภาษา VRML2.0 มีความสามารถในการสร้างวัตถุรูปทรงต่างๆ และประกอบวัตถุต่างๆ จัดวางเป็นโลกเสมือน ซึ่งผู้ใช้สามารถมองเห็นโลกเสมือนเป็น 3 มิติ ผู้ใช้สามารถท่องไปในโลกเสมือนนั้นๆ ภาษา VRML2.0 สามารถสร้างโลกเสมือนที่มีการเคลื่อนไหว, มีการโต้ตอบ ทำให้ผู้ใช้สามารถท่องไปในโลกเสมือนได้อย่างสมจริงมากขึ้น แต่ภาษา

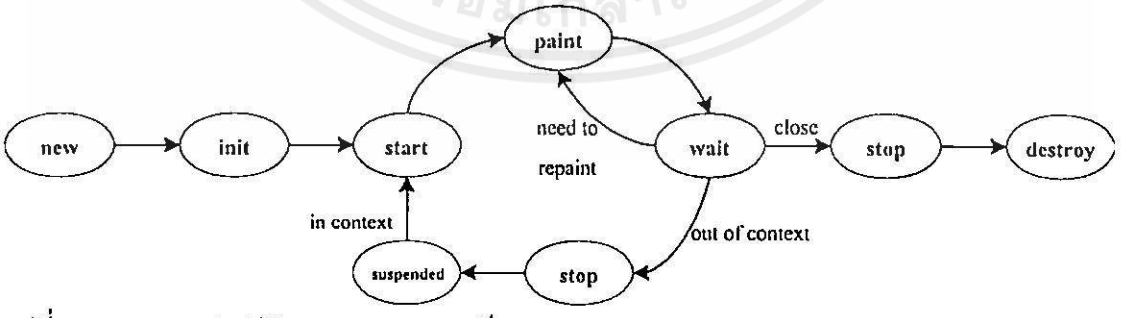
VRML2.0 ก็ยังมีจุดด้อยอยู่บางประการ อย่างเช่น ความสามารถในการสื่อความหมายเพื่อใช้ในการติดต่อกับผู้ใช้ยังมีข้อจำกัดอยู่ ตัวอย่างเช่น การติดต่อกับผู้ใช้ที่ต้องอาศัยคุณสมบัติของความเป็น 3 มิติ เช่น การจับต้องวัตถุรูปทรงต่างๆ , การเคลื่อนย้ายเปลี่ยนตำแหน่งวัตถุโดยผู้ใช้ยังทำได้ไม่ดีพอ เนื่องจากอุปกรณ์ที่ผู้ใช้ใช้ติดต่อกับโลกเสมือน คือ เมาส์ ซึ่งเป็นอุปกรณ์แบบ 2 มิติ ซึ่งภาษา VRML2.0 ไม่มีความสามารถในการติดต่อกับผู้ใช้ทางแป้นพิมพ์ แต่ในบางกรณีภาษา VRML2.0 ก็สามารถสื่อความหมายและติดต่อกับผู้ใช้ได้ดี เช่น เมื่อใช้เดินอยู่ในห้องต่างๆ แล้วมองเห็นสวิทช์ไฟที่ผนังห้อง ผู้ใช้จะทราบทันทีว่าเป็นสวิทช์ไฟ และสามารถเปิดปิดได้โดยใช้เมาส์คลิก ดังนั้นในการใช้ภาษา VRML2.0 ในขั้นตอนการออกแบบต้องคำนึงถึงเรื่องการติดต่อและการโต้ตอบกับผู้ใช้ให้เหมาะสมกับรูปแบบและสถานการณ์หรือเหตุการณ์ต่างๆ ด้วย

2.13 ภาษาจาวา

หลักการภาษาจาวาที่นำมาใช้ในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML ประกอบด้วย 2 เรื่องคือ จาวาแอพเพล็ตและจาวาเน็ตเวิร์ก ดังหลักการทำงานต่อไปนี้

2.13.1 จาวาแอพเพล็ต

แอพเพล็ต คือ โปรแกรมขนาดเล็กที่สร้างขึ้นด้วยภาษาจาวาสามารถถูกเรียกจากใน HTML Page ให้ทำงานเป็นส่วนหนึ่งของเว็บเพจได้ การเรียกให้แอพเพล็ตทำงานนั้นต้องทำจากภายใน HTML Page โดยใช้ APPLET tag และแอพเพล็ตนั้นจะทำงานในสภาวะแวดล้อมของเว็บเบราเซอร์ ซึ่งแอพเพล็ตจะมีช่วงชีวิตในการทำงานแบ่งออกเป็นสถานะต่างๆ ได้ดังนี้ init, start, paint, wait, suspended, stop และ destroy โดยทุกๆ แอพเพล็ตจะดำเนินการผ่านสถานะเหล่านี้ โดยบางส่วนจะเป็นไปอย่างอัตโนมัติและบางส่วนจะอยู่ในการควบคุมของผู้ใช้ที่เรียกดูแอพเพล็ตนั้น ช่วงชีวิตของแอ็พเพลตแสดงดังรูปการเปลี่ยนสถานะรูปที่ 2.18 [21]



รูปที่ 2.18 แสดงช่วงชีวิตของจาวาแอพเพล็ต

จากรูปเมื่อเริ่มต้น แอพเพล็ตถูกสร้างขึ้นโดยการ new() แล้วจะเริ่มการทำงานทันที [21] คือ

1. เข้าสู่สถานะ init แล้วผ่านเข้าสู่สถานะ start จากนั้นจะเข้าสู่สถานะ paint และเข้าสู่สถานะ wait ซึ่งจะหยุดรอเหตุการณ์หรือสัญญาณจากผู้ใช้ (คือ Input ที่รับเข้าทางเมาส์, คีย์บอร์ด หรือ GUI) ที่จะเข้ามาให้เกิดการทำงานของ method ที่กำหนดไว้สำหรับจัดการกับเหตุการณ์นั้น เมื่อเสร็จแล้วจะกลับเข้าสู่สถานะ wait เพื่อรอเหตุการณ์อื่นต่อไป
2. ระหว่างที่แอพเพล็ตอยู่ในสถานะ wait อยู่ หากผู้ใช้เปลี่ยนแปลงเว็บเพจจนทำให้
   2.1 พื้นที่แสดงผลของแอพเพล็ตนั้นต้องมีการวาดใหม่ เช่น เมื่อแอพเพล็ตถูกย้ายที่, เปลี่ยนขนาด (resize), ขยายขนาด (maximize) หรือมีหน้าต่างอื่นมาทับแอพเพล็ต เว็บเบราเซอร์จะออกคำสั่งให้แอพเพล็ตทำการวาดอีกครั้ง โดยเรียก repaint() ซึ่งจะส่งผลให้แอพเพล็ตเข้าสู่สถานะ paint และทำการวาดพื้นที่แสดงผลของแอพเพล็ตนั้นอีกครั้ง เมื่อเสร็จแล้วจะกลับเข้าสู่สถานะ wait เพื่อรอเหตุการณ์อื่นต่อไป
   2.2 พื้นที่แสดงผลของแอพเพล็ตนั้นไม่ปรากฏในเว็บเพจปัจจุบัน เช่น เมื่อผู้ใช้ทำการเลื่อนเพจไปดูหน้าอื่นหรือกดปุ่ม minimize จะทำให้เว็บเบราเซอร์ออกคำสั่งให้แอพเพล็ตผ่านเข้าสู่สถานะ stop และไปสู่สถานะ suspended ซึ่งแอพเพล็ตจะหยุดรออยู่ในสถานะนี้โดยไม่ยอมรับ Input จากผู้ใช้และไม่โต้ตอบใดๆ ไปจนกว่าพื้นที่แสดงผลของแอพเพล็ตนั้นจะถูกทำให้กลับมาปรากฏอีกครั้ง เว็บเบราเซอร์ก็จะออกคำสั่งให้แอพเพล็ตนั้นเข้าสู่สถานะ start, paint และ wait เพื่อเริ่มทำงานอีก
3. ระหว่างที่แอพเพล็ตอยู่ในสถานะ wait หากผู้ใช้ปิดเว็บเบราเซอร์ เพื่อหยุดการทำงาน จะทำให้แอพเพล็ตเข้าสู่สถานะ stop และไปสู่สถานะ destroy และสิ้นสุดการทำงาน

การสร้างคลาสของแอพเพล็ตควรทำการ override methods: init(), start(), paint(), stop(), และ destroy() แต่ไม่ควรเขียนโปรแกรมที่เรียก methods เหล่านี้โดยตรง แม้ว่าสามารถทำได้แต่โดยปกติจะปล่อยให้เว็บเบราเซอร์เป็นผู้เรียก methods เหล่านี้ เงื่อนไขที่แต่ละ method นี้จะถูกเรียกมีดังนี้ [21]

- init() จะถูกเรียกเมื่อแอพเพล็ตเริ่มต้นทำงานเป็นครั้งแรกและจะทำงานครั้งเดียวเท่านั้น จึงเหมาะสำหรับใช้ในการกำหนดค่าเริ่มต้นให้แก่ตัวแปรในแอพเพล็ตนั้น
- start() จะถูกเรียกหลังการทำงานของ init() เป็นการเริ่มทำงานตามที่กำหนดไว้และจะถูกเรียกทุกครั้งเมื่อพื้นที่แสดงผลของแอพเพล็ตถูกทำให้ปรากฏในเว็บเพจ
- paint() โดยปกติ paint() จะถูกเรียกหลังจากทำงานของ start() และจะถูกเรียกเมื่อเว็บเบราเซอร์ต้องการวาดในพื้นที่แสดงผลของแอพเพล็ตนั้นใหม่ ซึ่งอาจเกิดขึ้นได้หลาย

กรณีเช่น พื้นที่แสดงผลของแอพเพล็ตนั้นถูกหน้าต่างอื่นวาดทับและเมื่อหน้าต่างนั้นถูกนำออกไปแล้วหรือในกรณีที่เว็บเพจนั้นถูก minimized (หรือ maximized) และเมื่อถูกนำกลับมาอีก paint() ก็จะถูกเรียกเช่นกัน

- stop() จะถูกเรียกเมื่อเว็บเพจถูกเลื่อนไปหน้าอื่นที่ไม่มีพื้นที่แสดงผลของแอพเพล็ตนั้นปรากฏหรือเว็บเพจนั้นถูก minimized เมื่อใดที่แอพเพล็ตถูก stop() จะทำให้ไม่มีการแสดงผลและไม่สามารถโต้ตอบกับผู้ใช้ แต่ถ้าแอพเพล็ตนั้นมี threads อยู่ด้วยก็จะยังมีการทำงานต่อไป หากต้องการให้ threads เหล่านั้นหยุดทำงาน ก็ต้องเรียก stop() ของ threads เหล่านั้นด้วย
- destroy() จะถูกเรียกเมื่อแอพเพล็ตนั้นสิ้นสุดการทำงานแล้ว ซึ่งจะเกิดขึ้นเมื่อเว็บเพจนั้นถูกยกเลิกการทำงาน เช่น ปิดเว็บเบราเซอร์ก็จะทำให้เว็บเบราเซอร์เรียก stop() ก่อนจะเรียก destroy() เมื่อแอพเพล็ตทำการ destroy() พื้นที่หน่วยความจำและทรัพยากรของมันจะถูกเก็บกลับคืนไปไม่สามารถถูก start() ได้อีก

2.13.2 จาวาเน็ตเวิร์ก

ภาษาจาวาได้ให้ความสามารถและอำนวยความสะดวกในการสร้างหรือเปิดช่องทางการติดต่อสื่อสาร (Socket) ระหว่างกันเอาไว้แล้ว ทั้งลักษณะที่เปิดเป็น Server Socket เพื่อคอยฟังการติดต่อสื่อสารจากไคลเอนต์และเป็น Client Socket เพื่อทำการติดต่อสื่อสารไปยังเซิร์ฟเวอร์ โดยมีหลักการใช้งานคร่าวๆ ดังนี้

การสร้าง Client Socket เพื่อทำการติดต่อไปยังเซิร์ฟเวอร์ในภาษาจาวานั้นสามารถทำได้โดยการสร้างออบเจ็กต์จากคลาส Socket โดยมีพารามิเตอร์เป็น Remote Address และหมายเลขพอร์ทที่จะติดต่อไป ดังตัวอย่าง

```
Socket t = new Socket(remote_address,port_number);
```

เมื่อฝั่งเซิร์ฟเวอร์รับการติดต่อสื่อสารนี้หรือเมื่อ Socket ที่สร้างขึ้นถูกเปิดแล้ว เซิร์ฟเวอร์ก็สามารถที่จะส่งแมสเสจออกไปบนช่องทางหรือ Socket (t) ที่ถูกสร้างขึ้นมานี้ได้ ด้วยขั้นตอนดังนี้

1. ทำการเรียก method getOutputStream() ของ Socket (t) ที่ถูกสร้างขึ้นมา เพื่อให้ส่งออบเจ็กต์ที่เป็น OutputStream ออกมา แล้วนำ OutputStream ที่ได้ไปใช้สร้างเป็น PrintStream (out) อีกทีหนึ่ง ดังตัวอย่าง

```
PrintStream out = new PrintStream(t.getOutputStream());
```

<u>หมายเหตุ</u> นอกจาก PrintStream แล้ว สามารถนำ OutputStream ที่ได้จาก method getOutputStream() ของออบเจ็กต์ Socket มาแปลงหรือมาสร้างเป็น OutputStream ชนิดอื่นๆ ได้ เช่น BufferedOutputStram, DataOutputStram เป็นต้น ซึ่งขึ้นอยู่กับผู้ใช้

ว่าต้องการที่จะส่งผลลัพธ์ออกไปเป็นรูปแบบใดและด้วยวิธีการแบบใด ดังเช่นในตัวอย่างต้องการส่งผลลัพธ์ออกเป็นข้อความจึงใช้ PrintStream

2. จากนั้นก็จะสามารถที่จะส่งผลลัพธ์ออกไปทาง PrintStream ดังกล่าวได้โดยใช้ method println() ของ PrintStream (out) นั้น ดังแสดงในตัวอย่าง

```
out.println(message);
```

สำหรับการสร้าง Server Socket เพื่อรับการติดต่อสื่อสารจากไคลเอนต์นั้น ในภาษาจาวาสามารถทำได้โดยการสร้างออบเจ็กต์จากคลาส ServerSocket โดยมีพารามิเตอร์เป็นหมายเลขพอร์ทที่จะใช้เปิดรับการติดต่อสื่อสารจากไคลเอนต์ดังตัวอย่าง

```
ServerSocket s = new ServerSocket(port_number);
```

และสามารถให้ ServerSocket ที่สร้างขึ้นมานี้คอยรับฟังการติดต่อสื่อสารจากไคลเอนต์ได้โดยเรียก method accept() ของออบเจ็กต์ ServerSocket (s) ที่ถูกสร้างขึ้นมา ซึ่งหากว่ามีไคลเอนต์ทำการติดต่อสื่อสารมายังพอร์ทที่ถูกกำหนดไว้แล้ว method นี้จะส่งค่าเป็นออบเจ็กต์ Socket ออกมา ดังตัวอย่าง

```
Socket incomming = s.accept();
```

ซึ่งสามารถที่จะทำการอ่านแมสเสจที่ไคลเอนต์ส่งมาผ่านทาง Socket (incomming) นี้ได้ดังขั้นตอนต่อไปนี้

1. ทำการเรียก method getInputStream() ของ Socket(incomming) ดังกล่าว ซึ่งจะได้ผลลัพธ์เป็นออบเจ็กต์ InputStream ออกมา แล้วนำ InputStream ที่ได้ไปใช้สร้างเป็น DataInputStream อีกทีหนึ่ง ดังตัวอย่าง

```
DataInputStream in = new DataInputStream(incoming.getInputStream());
```

<u>หมายเหตุ</u> นอกจาก DataInputStream แล้ว ยังสามารถนำ InputStream ที่ได้จาก method getInputStream() ของออบเจ็กต์ Socket มาแปลงหรือมาสร้างเป็น InputSream ชนิดอื่นๆ ได้ เช่น BufferedInputStram, PushbackInputStram เป็นต้น ขึ้นอยู่กับผู้ใช้ว่าต้องการที่จะรับ Input เข้ามาเป็นรูปแบบใดและด้วยวิธีการแบบใด ดังในตัวอย่างต้องการรับ Input มาเป็นข้อความจึงใช้ DataInputStream

2. จากนั้นก็จะสามารถที่จะรับ Input จากทาง DataInputStream ดังกล่าวได้โดยใช้ method readLine() ของ DataInputStream (in) นั้น ดังแสดงในตัวอย่าง

```
String str = in.readLine();
```

2.14 สรุป

หลังจากที่มีการใช้บริการเวิลด์ไวด์เว็บในระบบอินเตอร์เน็ต อันเป็นบริการที่สามารถใช้งานกับเครื่องต่างรุ่นต่างระบบแล้ว ข้อดีของการให้บริการเวิลด์ไวด์เว็บคือ การที่ผู้ใช้สามารถเรียนรู้วิธีการใช้งานและสามารถพัฒนางานได้ง่ายและสะดวก สำหรับการบริการแบบเวิลด์ไวด์เว็บส่วนที่ทำหน้าที่ติดต่อกับฐานข้อมูลภายนอกหรือโปรแกรมภายนอกที่ทำงานเฉพาะอย่าง เช่น การนำรายการที่กรอกในแบบฟอร์มไปค้นหารายละเอียดในฐานข้อมูลแล้วนำมาแสดงผล ส่วนที่ทำหน้าที่นี้คือ CGI ซึ่งทำหน้าที่เป็นส่วนที่กำหนดรูปแบบการติดต่อระหว่างบริการเวิลด์ไวด์เว็บกับโปรแกรมภายนอก โดยรับรายการจากแฟ้มข้อมูลของเว็บเพจ ดังนั้นในการออกแบบระบบหรือรูปแบบการทำงานของส่วนต่างๆ ภายในระบบ ผู้ออกแบบจะต้องคำนึงถึงสิ่งที่จะต้องนำมาใช้ประกอบในการทำงานของส่วนต่างๆ ที่เกี่ยวข้อง โดยอาจดูจากความเหมาะสม, หน้าที่การทำงานและความเข้ากันได้ของส่วนต่างๆ โดยบริการเวิลด์ไวด์เว็บเป็นบริการที่ได้จากการรวมลักษณะการทำงานแบบ Hypertext และกติกาการส่งข้อมูลในระบบเครือข่าย แต่ส่วนประกอบของแฟ้มข้อมูลแบบ Hypertext หรือ HTML เอง ไม่สามารถทำหน้าที่บางอย่างที่จำเป็นได้ จึงต้องอาศัยส่วนประกอบอื่นช่วยที่เรียกว่า CGI ซึ่งเป็นโปรแกรมที่ทำหน้าที่เฉพาะอย่าง เช่น ใช้เป็นส่วนที่ติดต่อกับฐานข้อมูล, ใช้ในการนับจำนวนผู้ที่เข้ามาอ่านเว็บเพจหรือใช้ติดต่อกับบริการแบบอื่นๆ เป็นต้น

บทที่ 3
หลักการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่าย ผ่านเว็บด้วย VRML

เนื่องจากลักษณะโปรแกรมการติดต่อกับผู้ใช้ในการจัดการเครือข่ายในรูปแบบเดิมเป็นอินเตอร์เฟซแบบ 2 มิติ และไม่อิงกับตำแหน่งจริงทางกายภาพ ดังนั้นในบทนี้จะได้กล่าวถึงแนวคิด, สถาปัตยกรรม, หลักการทำงานของระบบ และการออกแบบฐานข้อมูล เพื่อนำเสนอลักษณะการติดต่อกับผู้ใช้ในรูปแบบ 3 มิติ ผ่านเครือข่ายอินเตอร์เน็ตและอิงกับตำแหน่งจริงทางกายภาพ

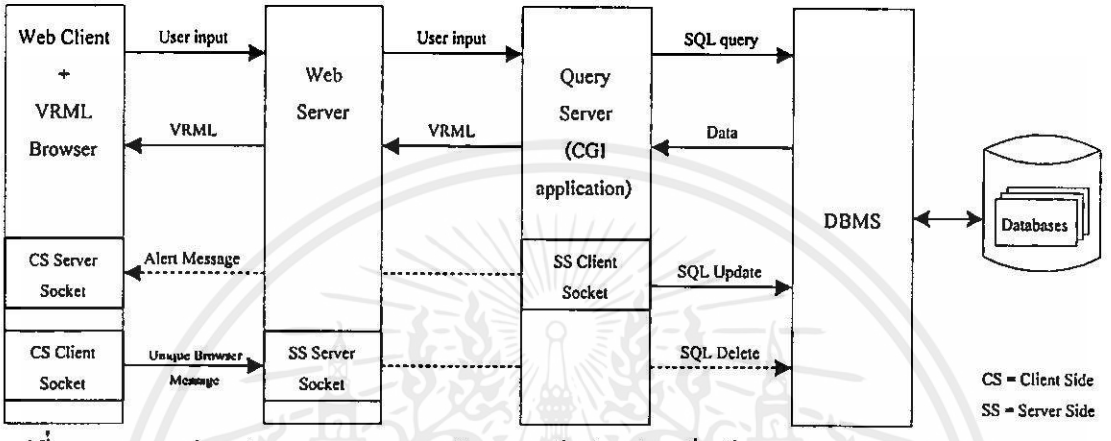
3.1 แนวความคิดของระบบ

แนวความคิด คือ การปรับปรุงรูปแบบในการนำเสนอการจัดการเครือข่ายเป็นหลัก ด้วยการพัฒนาส่วนติดต่อผู้ใช้ผ่านระบบเครือข่ายอินเตอร์เน็ต โดยผู้ใช้ทำการระบุคำร้องขอผ่านเว็บเบราเซอร์ไปยังเว็บเซิร์ฟเวอร์ เพื่อส่งต่อให้กับ CGI ในการประมวลผล แล้วส่งผลลัพธ์กลับคืนให้กับเว็บเซิร์ฟเวอร์ แล้วเว็บเซิร์ฟเวอร์จะส่งผลลัพธ์นั้นให้กับเว็บเบราเซอร์ที่ส่งการร้องขอมา เพื่อพัฒนารูปแบบในการนำเสนอเป็นแบบ 3 มิติ โดยอาศัยทฤษฎีต่างๆ ในการจัดการโครงสร้างในการนำเสนอกราฟฟิกแบบ 3 มิติผ่านเว็บ ด้วยการนำภาษา VRML มาใช้ในการสร้างภาพ 3 มิติ และอีกทั้ง VRML ยังสามารถเรียกดูผ่านเว็บเบราเซอร์ได้ด้วย แต่เนื่องจาก VRML เป็นการนำเสนอข้อมูลแบบสแตติก จึงได้มีการนำเทคโนโลยี CGI มาใช้เพื่อจัดการกับ VRML ให้มีคุณสมบัติในการนำเสนอข้อมูลแบบไดนามิกและด้วยข้อดีของ CGI ในการทำงานแบบแยกพื้นที่หน่วยความจำและโปรเซสของเว็บเซิร์ฟเวอร์อย่างเด็ดขาด แม้ว่าโปรเซสใดๆ เกิดไม่ยอมทำงานขึ้นมาก็ไม่ส่งผลอย่างไรต่อการทำงานของเว็บเซิร์ฟเวอร์หรือโปรเซสอื่นๆ แต่อย่างใด เมื่อถึงเวลาที่กำหนดเว็บเซิร์ฟเวอร์ก็จะส่งข้อมูลแจ้งข้อผิดพลาดในการสั่งงานกลับไปสู่เครื่องไคลเอนต์ ส่งผลให้การทำงานของเว็บเซิร์ฟเวอร์มีความมั่นคงและเนื่องจากลักษณะการทำงาน เป็นการดึงข้อมูลจากฐานข้อมูลมาแสดงผ่านเว็บเบราเซอร์ ซึ่งข้อมูลที่นำมาแสดงเป็นข้อมูลเชิงไดนามิก เพราะเป็นข้อมูลของเครือข่ายที่มีการเปลี่ยนแปลงตลอดเวลา โดยที่ CGI จะให้เว็บเซิร์ฟเวอร์ประมวลผลโปรแกรมและรวบรวมผลลัพธ์ที่ได้จากโปรแกรมเหล่านั้นให้อยู่ในรูปของเท็กซ์ เพื่อส่งไปยังเว็บเบราเซอร์ เพื่อให้ HTML page ทำงานแบบไดนามิก และที่สำคัญ CGI สามารถรับค่าที่ส่งผ่านจากเซิร์ฟเวอร์ได้และยังสามารถส่งค่ากลับไปหาเซิร์ฟเวอร์ได้อีกด้วย

3.2 การทำงานของระบบ

3.2.1 ส่วนประกอบของระบบ

การพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ประกอบด้วยส่วนต่างๆ ดังรูปที่ 3.1 โดยมีรายละเอียดดังต่อไปนี้



รูปที่ 3.1 แสดงส่วนประกอบของระบบจัดการเครือข่ายผ่านเว็บด้วย VRML

เว็บไคลเอนต์ทำหน้าที่ในการร้องขอบริการจากเว็บเซิร์ฟเวอร์และนำเสนอข้อมูลที่ได้รับกลับมา โดยในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML นี้ เว็บไคลเอนต์จะต้องสนับสนุนการเรียกดูข้อมูล VRML โดยเว็บเบราเซอร์ต้องมีการติดตั้งปลั๊กอินเพื่อให้สามารถเรียกดูข้อมูล VRML ได้ นอกจากนี้เว็บไคลเอนต์จะต้องสนับสนุนการทำงานของจาวาแอ็พเพล็ตด้วย เพราะจะต้องมีการโหลดแอ็พเพลตที่ทำหน้าที่เป็น Server Socket ไปไว้ที่เว็บไคลเอนต์ เพื่อทำหน้าที่ในการติดต่อสื่อสารกับ Client Socket ที่อยู่ฝั่งเซิร์ฟเวอร์ เมื่อมีการเปลี่ยนแปลงข้อมูลในฐานข้อมูลฝั่งเซิร์ฟเวอร์ ซึ่งในส่วนของกระบวนการแจ้งการเปลี่ยนแปลงข้อมูลบนฝั่งเซิร์ฟเวอร์ไปให้ฝั่งไคลเอนต์ทราบนั้นจะกล่าวถึงรายละเอียดของการทำงานของกระบวนการนี้ในหัวข้อที่ 3.3 ต่อไป และสำหรับกรณีที่ผู้ใช้ออกจากระบบหรือการปิดเว็บเบราเซอร์โดยไม่ได้ Logout ในระบบการพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ได้ทำการพัฒนาแอพเพล็ต ซึ่งจะทำหน้าที่ในการแจ้งข้อความไปให้เว็บเซิร์ฟเวอร์รู้ว่าเว็บเบราเซอร์ที่มันฝังตัวอยู่นั้นถูกปิดลง โดยรายละเอียดของกระบวนการทำงานในหัวข้อนี้จะกล่าวในหัวข้อ 3.3 แต่กระบวนการทำงานหลักๆ โดยทั่วไประหว่างส่วนประกอบอื่นๆ จะมีลักษณะดังนี้คือ เว็บไคลเอนต์จะส่งการร้องขอด้วยการระบุ URL (Uniform Resource Locator) ไปยังเว็บเซิร์ฟเวอร์ ต่อจากนั้นเว็บเซิร์ฟเวอร์จะทำการเรียกใช้งานแอปพลิเคชั่นที่อยู่ฝั่งเว็บเซิร์ฟเวอร์ที่เรียกว่า CGI ให้ทำการประมวลผลและส่งผลลัพธ์กลับมาในรูปแบบของ VRML ต่อจากนั้นเว็บเซิร์ฟเวอร์จะทำการส่งข้อมูล VRML ที่ได้ดังกล่าวกลับไปให้เว็บไคลเอนต์ที่ทำการร้องขอมา นั่นคือ ในระบบนี้จะอาศัยหลักการทำงานของ CGI ในการเขียนโปรแกรมสคริปต์

สำหรับใช้ในการประมวลผลเพื่อให้ได้ผลลัพธ์ออกมาเป็น VRMLโดยที่ CGI จะติดต่อกับดาต้าเบสเซิร์ฟเวอร์ที่มีหน้าที่จัดเก็บและควบคุมดูแลการเรียกใช้ข้อมูลให้เป็นไปอย่างถูกต้อง ซึ่งจะถูกเข้าถึงจากโปรแกรมสคริปต์ (CGI) ที่พัฒนาขึ้นมา

3.2.2 หลักการทำงานของระบบ

ในงานวิจัยนี้มีโครงสร้างพื้นฐานเป็นเวิล์ดไวด์เว็บ โดยผู้ใช้งานจะทำการร้องขอการทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยการระบุ URL ไปยังเว็บเซิร์ฟเวอร์ ต่อจากนั้นจะเข้าสู่การทำงานของ CGI นั่นคือ เมื่อเว็บเซิร์ฟเวอร์รับการร้องขอดังกล่าวไปจะทำการประมวลผลสคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซิร์ฟเวอร์แอปพลิเคชั่น (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผลและจะทำการดึงข้อมูลจากฐานข้อมูลเพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกประมวลผลจะส่งผลลัพธ์กลับไปให้เว็บเซิร์ฟเวอร์ แล้วเว็บเซิร์ฟเวอร์จะส่งผลลัพธ์นั้นกลับไปให้กับเว็บเบราเซอร์ที่ส่งการร้องขอมา ซึ่งในงานวิจัยนี้สามารถแบ่งสคริปต์ในระบบออกได้เป็น 2 ประเภท คือ

1. สคริปต์ที่ให้เรสพ็อนท์ในรูปแบบของ HTML ซึ่งมี MIME Type เป็น text/html
2. สคริปต์ที่ให้เรสพ็อนท์ในรูปแบบของ VRML ซึ่งมี MIME Type เป็น model/vrml

โดยหลักการทำงานของ VRML จะต้องมี VRML Browser ซึ่งเป็นตัวแปลภาษา เมื่อระบุชื่อ URL ของ VRML เข้าไปยังเว็บเบราเซอร์แล้วพบว่าแฟ้มข้อมูลมี MIME-Type เป็น "model/vrml" ดังนั้นเว็บเบราเซอร์จะเรียกใช้โปรแกรม VRML Browser เพื่อทำการตีความ VRML ไปเป็นภาพแบบกราฟฟิก [1]

หน้าที่หลักอย่างหนึ่งของระบบจัดการเครือข่ายก็คือ การมอนิเตอร์ดูสถานะของอุปกรณ์เครือข่าย ซึ่งระบบจัดการเครือข่ายที่เป็นลักษณะของ Web-Based โดยทั่วไปจะอาศัยการตั้งค่าช่วงเวลาที่จะให้มีการรีเฟชเอาไว้ เพื่อให้เว็บเพจแสดงสถานะล่าสุดของอุปกรณ์ ซึ่งลักษณะการทำงานที่กล่าวมาจะเป็นลักษณะที่ไคลเอนต์ทำการร้องขอไปที่เว็บเซิร์ฟเวอร์ทุกๆ ช่วงเวลาที่กำหนด ไม่ว่าข้อมูลที่เว็บเซิร์ฟเวอร์จะมีการเปลี่ยนแปลงหรือไม่ก็ตาม ซึ่งหากนำหลักการนี้มาประยุกต์ใช้กับระบบจัดการเครือข่ายผ่านเว็บด้วย VRML แล้ว อาจจะทำให้ประสิทธิภาพในการทำงานลดลง หากมีการรีเฟชบ่อยๆ เนื่องจากต้องเสียเวลาในการประมวลผลเพื่อแสดงภาพกราฟฟิก 3 มิติ ดังนั้นจึงได้เปลี่ยนมาอาศัยหลักการที่ว่าเว็บเซิร์ฟเวอร์จะเป็นฝ่ายส่งแมสเสจไปบอกไคลเอนต์เองเมื่อเวลาที่ข้อมูลฝั่งเว็บเซิร์ฟเวอร์มีการเปลี่ยนแปลงโดยที่ไคลเอนต์ไม่ต้องทำการร้องขอมาทุกๆ ช่วงเวลา ซึ่งหลักการที่กล่าวมานี้ ด้วยคุณสมบัติของโปรโตคอล HTTP เองไม่สามารถทำได้ เนื่องจากการทำงานของ HTTP จะมีลักษณะที่เป็น รีเควสท์/เรสพ็อนท์ ทำให้เว็บเซิร์ฟเวอร์ไม่สามารถติดต่อไปยังไคลเอนต์เองได้ ถ้าไม่มีการร้องขอจากไคลเอนต์ จึงได้มีการนำเทคโนโลยีของจาวาแอพเพล็ต

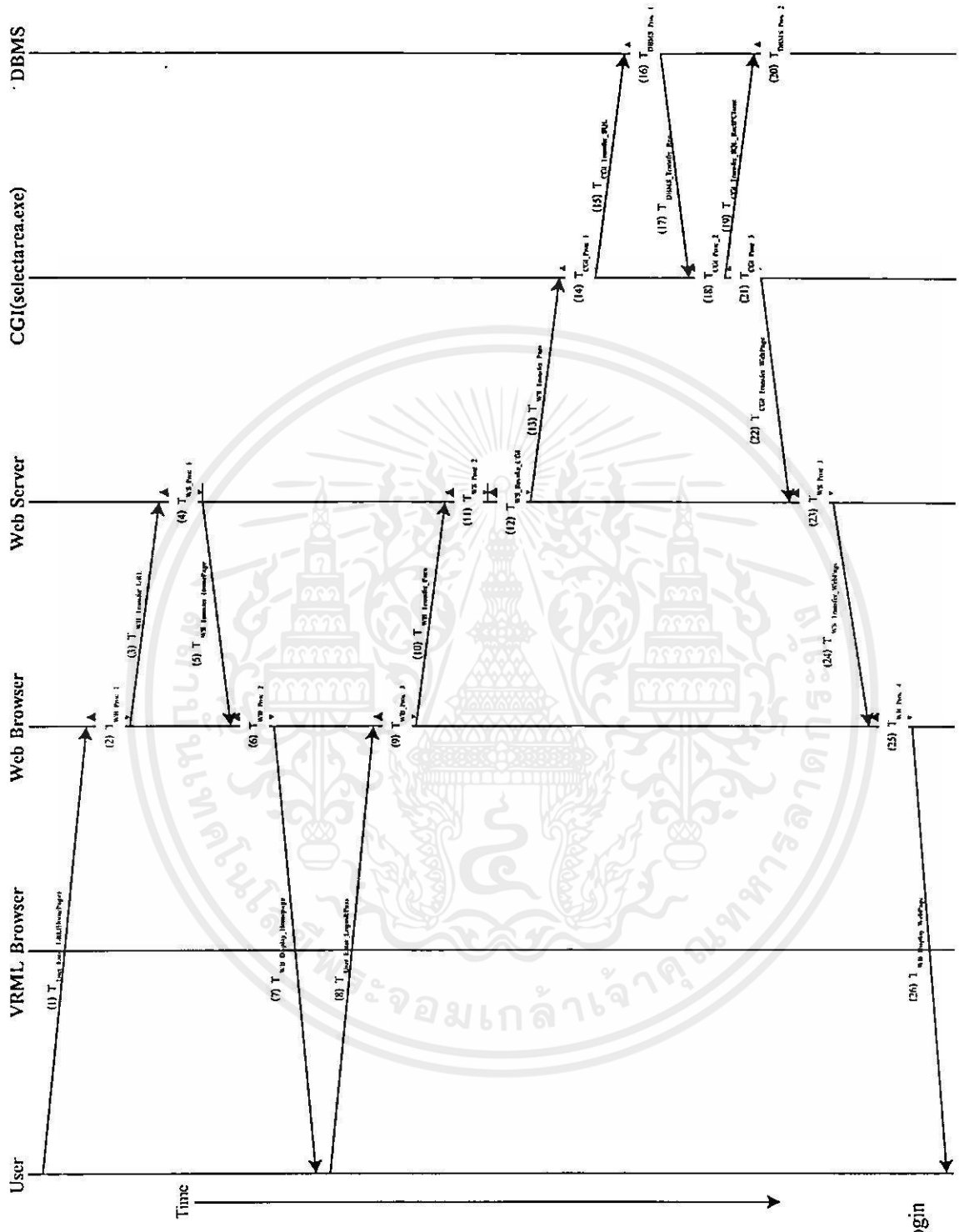
และจาวาเน็ตเวิร์ก [5,21] ในเรื่อง Server Socket และ Client Socket เข้ามาช่วยในการทำงาน เพื่อให้สามารถรองรับการทำงานในลักษณะที่กล่าวมาข้างต้นได้ โดยจะได้กล่าวถึงในหัวข้อที่ 3.3 ต่อไป

สำหรับหลักการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML มีรายละเอียดขั้นตอนการทำงานดังนี้

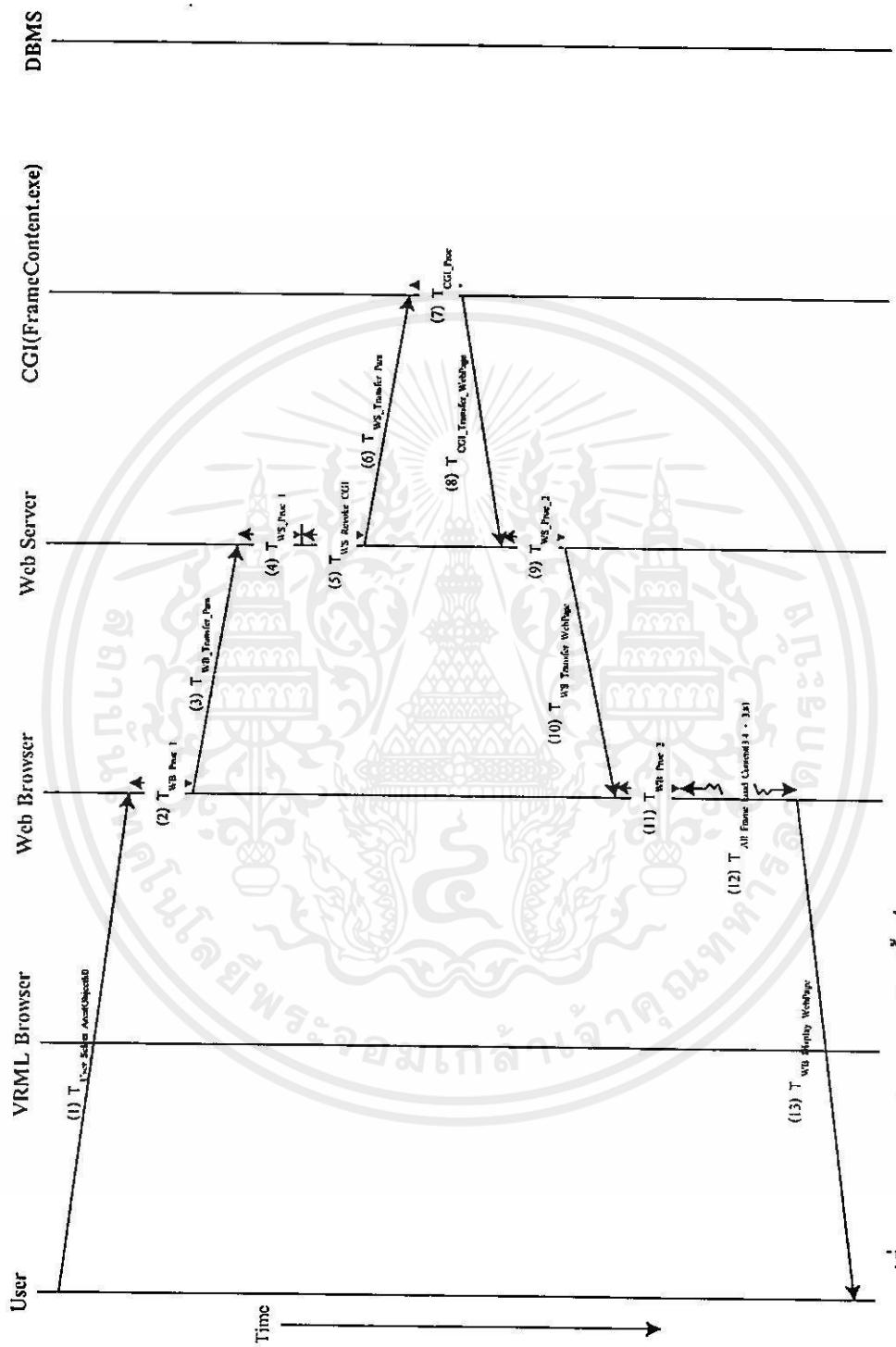
1. ผู้ใช้ทำการร้องขอโฮมเพจด้วยการระบุ URL ผ่านเว็บเบราเซอร์
2. เมื่อเว็บเบราเซอร์รับการป้อนข้อมูลจากผู้ใช้แล้วจะส่งการร้องขอตามที่ผู้ใช้ต้องการพร้อมกับข้อมูลที่จำเป็นสำหรับการสืบค้นข้อมูลตามที่ผู้ใช้ป้อนไปยังเว็บเซิร์ฟเวอร์
3. เว็บเซิร์ฟเวอร์ส่งโฮมเพจดังกล่าว ซึ่งเป็นหน้าจอให้ Login เข้าสู่ระบบมาให้แล้ว จากนั้นผู้ใช้ที่มีหน้าที่ในการบริหารเครือข่ายใส่ Login Name และ Password ซึ่งจะเป็นการไปเรียก CGI Script ที่ทำหน้าที่ในการ Login ทำงาน (ดังรูปที่ 3.2)
4. CGI Script ที่ทำหน้าที่ในการ Login ดังกล่าว ทำการตรวจสอบ Login Name และ Password ถ้าถูกต้องจะทำการบันทึก IP Address ของเครื่องไคลเอนต์และ ID ของเว็บเบราเซอร์ที่ทำการ Login ลงในฐานข้อมูลและทำการส่งผลลัพธ์หน้าจอเริ่มการทำงานกลับไปด้วย
5. ในส่วนของหน้าจอเริ่มการทำงานนี้ จะเป็นหน้าจอให้ผู้ใช้ทำการเลือกพื้นที่ที่จะเข้าไปสำรวจ ซึ่งเมื่อผู้ใช้ทำการเลือกพื้นที่ที่ต้องการ อันเป็นการไปเรียก CGI Script ที่ทำหน้าที่แสดงเว็บเพจแสดงรายละเอียดของพื้นที่นั้น (ดังรูปที่ 3.3)
6. CGI Script ดังกล่าวจะทำการตรวจสอบว่าผู้ใช้เลือกพื้นที่ใดและทำการประมวลผลหลังจากนั้นจะทำการส่งหน้าจอ ซึ่งแสดงรายละเอียดของพื้นที่นั้น โดยในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML นี้ได้แบ่งออกเป็น 4 เฟรม (ผู้ใช้สามารถแก้ไขโปรแกรมเพื่อเพิ่มหรือลดขนาดเฟรมได้) คือ Header (ดังรูปที่ 3.4), Footer (ดังรูปที่ 3.5), Menu (ดังรูปที่ 3.6) และ Content (ดังรูปที่ 3.7-3.8) ซึ่งในส่วนของ Content นี้เองที่จะเป็นการนำเสนอข้อมูลในส่วนการจัดการเครือข่ายในรูปแบบกราฟฟิก 3 มิติ นอกจากนี้ในส่วนของ Footer จะมีการฝัง HTML Tag ที่จะใช้ในการโหลดแอพเพล็ต ซึ่งจะมีหน้าที่เป็น Server Socket เพื่อคอยฟังการติดต่อสื่อสารจากไคลเอนต์ และ Client Socket เพื่อทำการติดต่อสื่อสารไปยังเซิร์ฟเวอร์
7. เมื่อไคลเอนต์ได้รับเว็บเพจดังกล่าวอย่างครบถ้วนแล้ว แอพเพล็ต Server Socket ที่รันอยู่ฝั่งไคลเอนต์ในส่วนหน้าจอที่เป็น Footer จะมีหน้าที่รอรับข่าวสารที่จะถูกส่งมาจาก Client Socket ที่รันอยู่ฝั่งเซิร์ฟเวอร์ โดยจะคอยฟังข่าวสารอยู่ตลอดเวลาเมื่อมีการเปลี่ยนแปลงฐานข้อมูลเกิดขึ้น (ดังรูปที่ 3.5)
8. สำหรับการตรวจสอบการออกจากระบบหรือการปิดเว็บเบราเซอร์ โดยไม่ได้ Logout แอพเพล็ต Client Socket ที่รันอยู่ฝั่งไคลเอนต์มีหน้าที่เปิด Socket ไปหาเซิร์ฟเวอร์ที่มี

การเปิด Server Socket ที่คอยรับฟังอยู่ โดย Client Socket จะต้องส่งแมสเสจซึ่งเป็นข้อมูลเกี่ยวกับการ Unique ID ของเว็บเบราเซอร์ โดยแอพเพล็ตนี้จะทำงานเองอัตโนมัติ เมื่อเว็บเพจนั้นถูกยกเลิกการทำงาน เพื่อให้เว็บเซิร์ฟเวอร์ทราบหมายเลข IP ของไคลเอนต์ที่ติดต่อเข้ามา เพื่อทำการลบ IP ที่ได้บันทึกไว้ในฐานข้อมูลในข้อ 4 และลดปริมาณการส่งข้อมูลข่าวสารในกรณีที่ฐานข้อมูลมีการเปลี่ยนแปลง (ดูรายละเอียดในหัวข้อ 3.3)

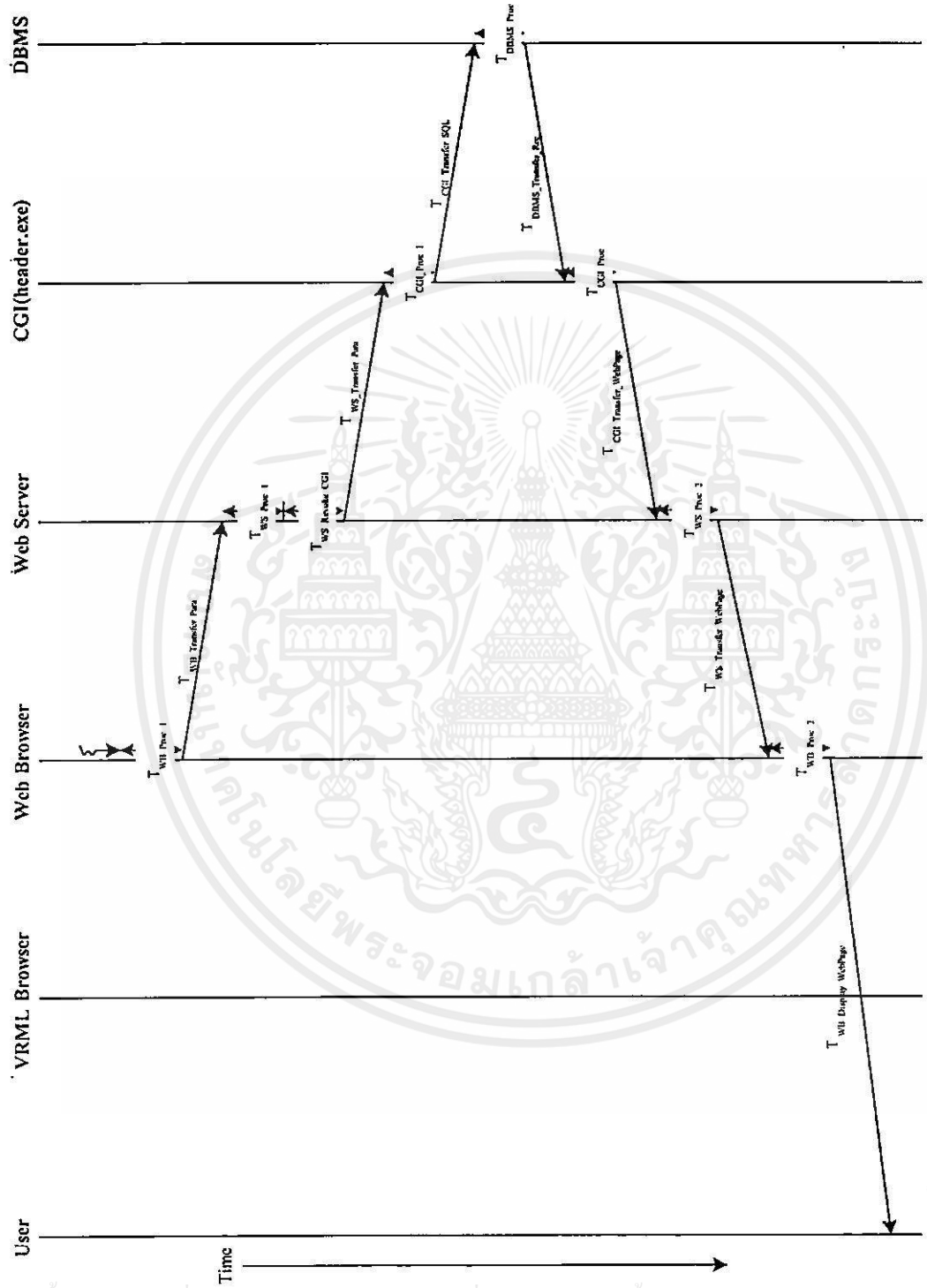
9. เมื่อผู้ใช้งานจะทำการร้องขอการทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ (ดังรูปที่ 3.9-3.10) โดยเว็บเซิร์ฟเวอร์จะคอยฟังการร้องขอที่ส่งมาและทำการประมวลผลสคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซิร์ฟเวอร์แอปพลิเคชั่น (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผลและทำการดึงข้อมูลจากฐานข้อมูล เพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกประมวลผลเรียบร้อยแล้วจะส่งผลลัพธ์กลับไปให้เว็บเซิร์ฟเวอร์ ต่อจากนั้นเว็บเซิร์ฟเวอร์จึงส่งผลลัพธ์ดังกล่าวกลับไปให้เว็บเบราเซอร์ที่ส่งการร้องขอมา
10. เนื่องจากหลักการนำเสนอข้อมูลของโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML ในแต่ละหน้าจอการทำงานของ VRML Browser (View) จะประกอบด้วยออบเจ็กต์ต่างๆ ที่เป็นโครงสร้างลำดับชั้น (Parent-Child Relationship) ดังรูปที่ 3.11 โดยในแต่ละออบเจ็กต์จะมี Viewpoint อันเป็นตัวกำหนดมุมมองในแต่ละ View และในแต่ละออบเจ็กต์นี้เองที่ผู้ใช้สามารถระบุตามความต้องการลงในฐานข้อมูลได้ว่าต้องการดูข้อมูลของออบเจ็กต์ในแต่ละ View ลึกลงไปกี่ระดับ ซึ่งการนำเสนอข้อมูลตามระดับต่างๆ ในวิทยานิพนธ์นี้จะใช้ URL เดียวกันคือ http://hostname/scripts/vnms/mainvrml.exe?objectid=xx ถ้าผู้ใช้กำหนดให้โปรแกรมแสดงหลายระดับในคราวเดียว เวลาที่ใช้ในการประมวลผลจะมากขึ้น ซึ่งมากกว่าการแสดงทีละระดับเนื่องจากโปรแกรมจะสามารถทยอยประมวลผลเพื่อส่งให้กับผู้ใช้ได้ ดังนั้นจึงขึ้นกับวิธีการเขียน CGI Script และการเก็บข้อมูลในฐานข้อมูลเพื่อนำมาสร้างเป็น VRML Script ด้วย (ดังรายละเอียดในหัวข้อ 3.6) ทั้งนี้ VRML Browser จะต้องสนับสนุนการกระทำกับคำสั่ง inline ที่มีมาในภาษา VRML ด้วยว่า VRML Browser นั้นๆ ใช้วิธีการประมวลผลแบบ one-time loading หรือทยอยแสดงผล [1]



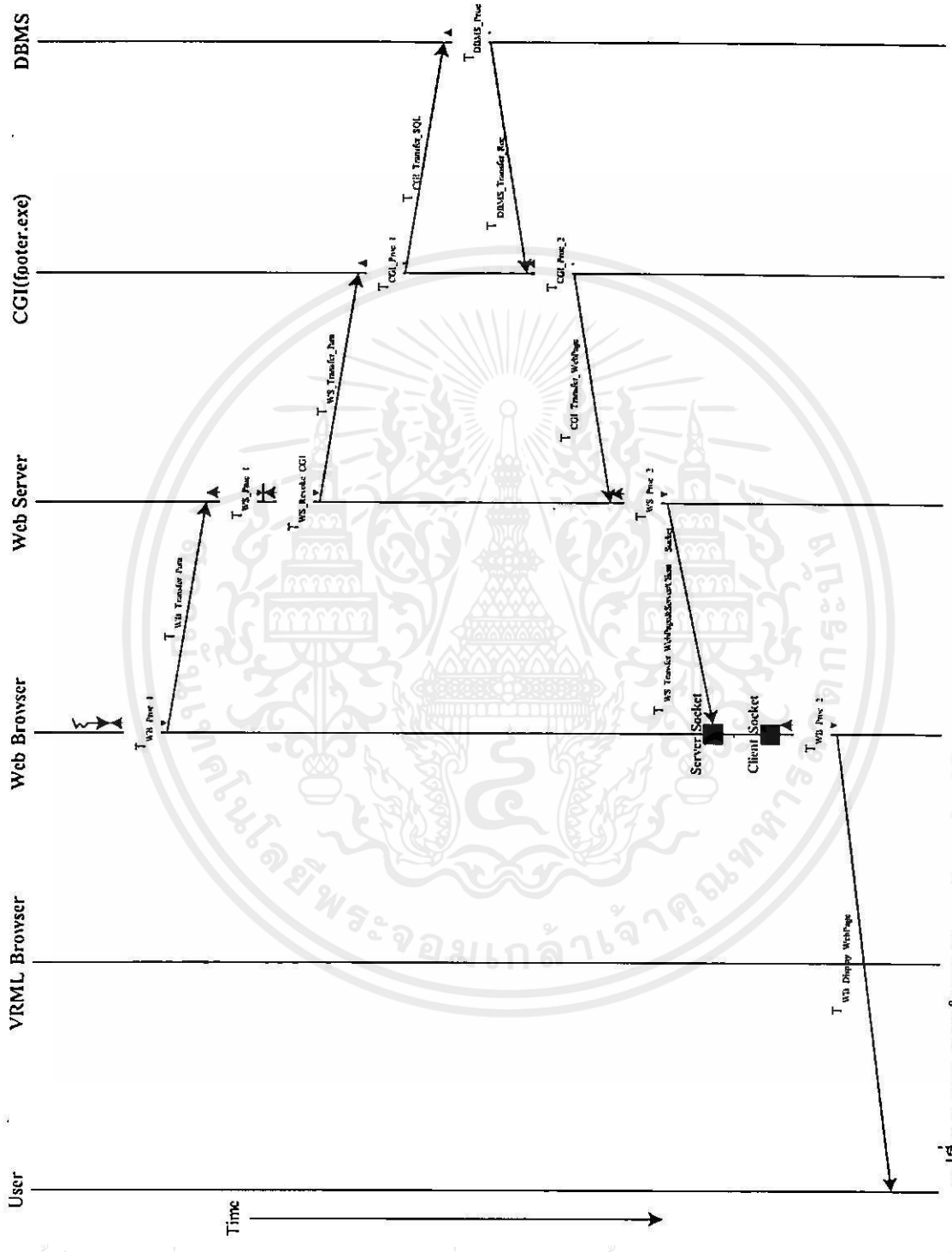
รูปที่ 3.2 แสดงการทำงาน Request Homepage & Login



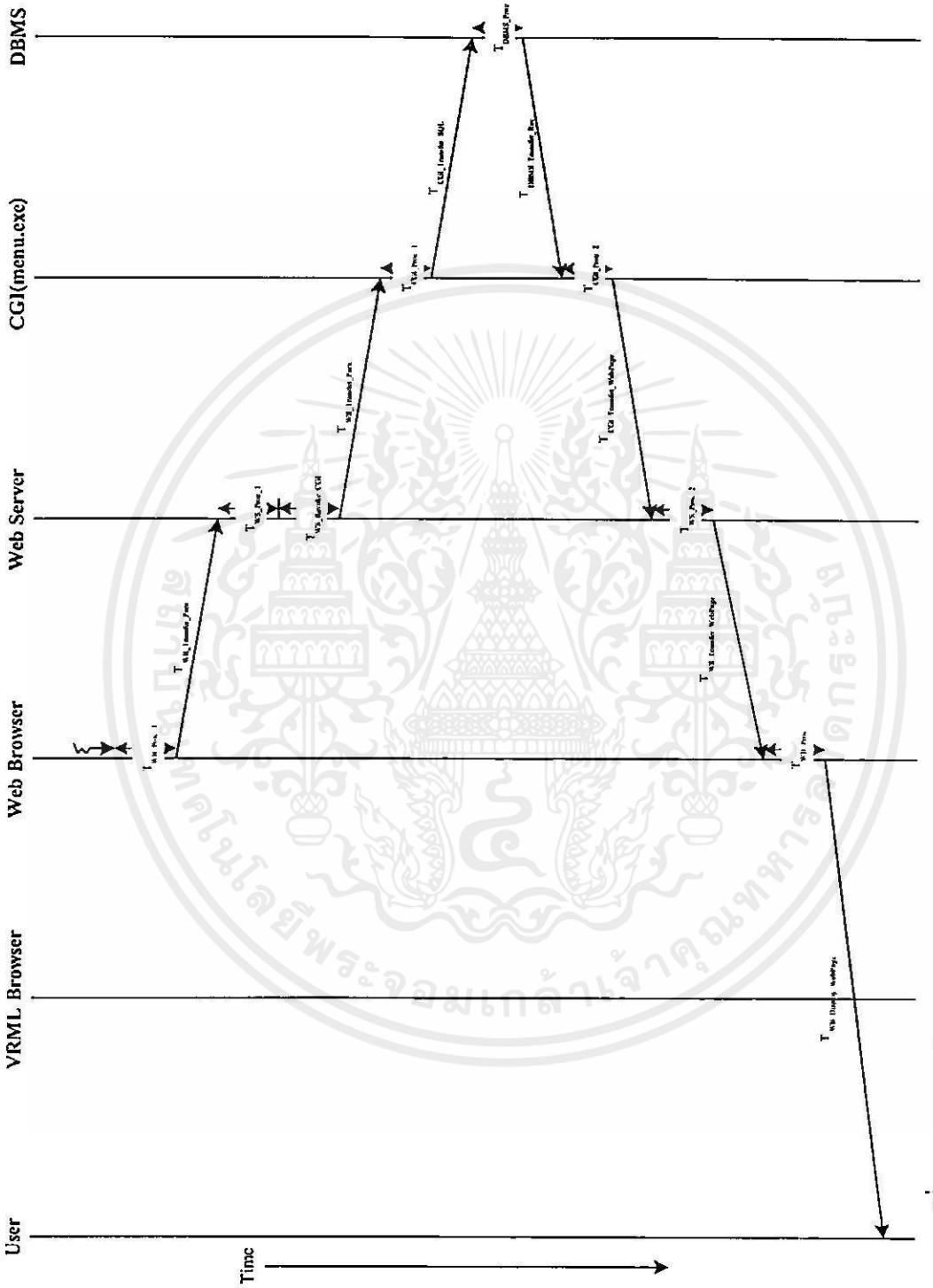
รูปที่ 3.3 แสดงการทำงานการเลือกพื้นที่ (Select Area)



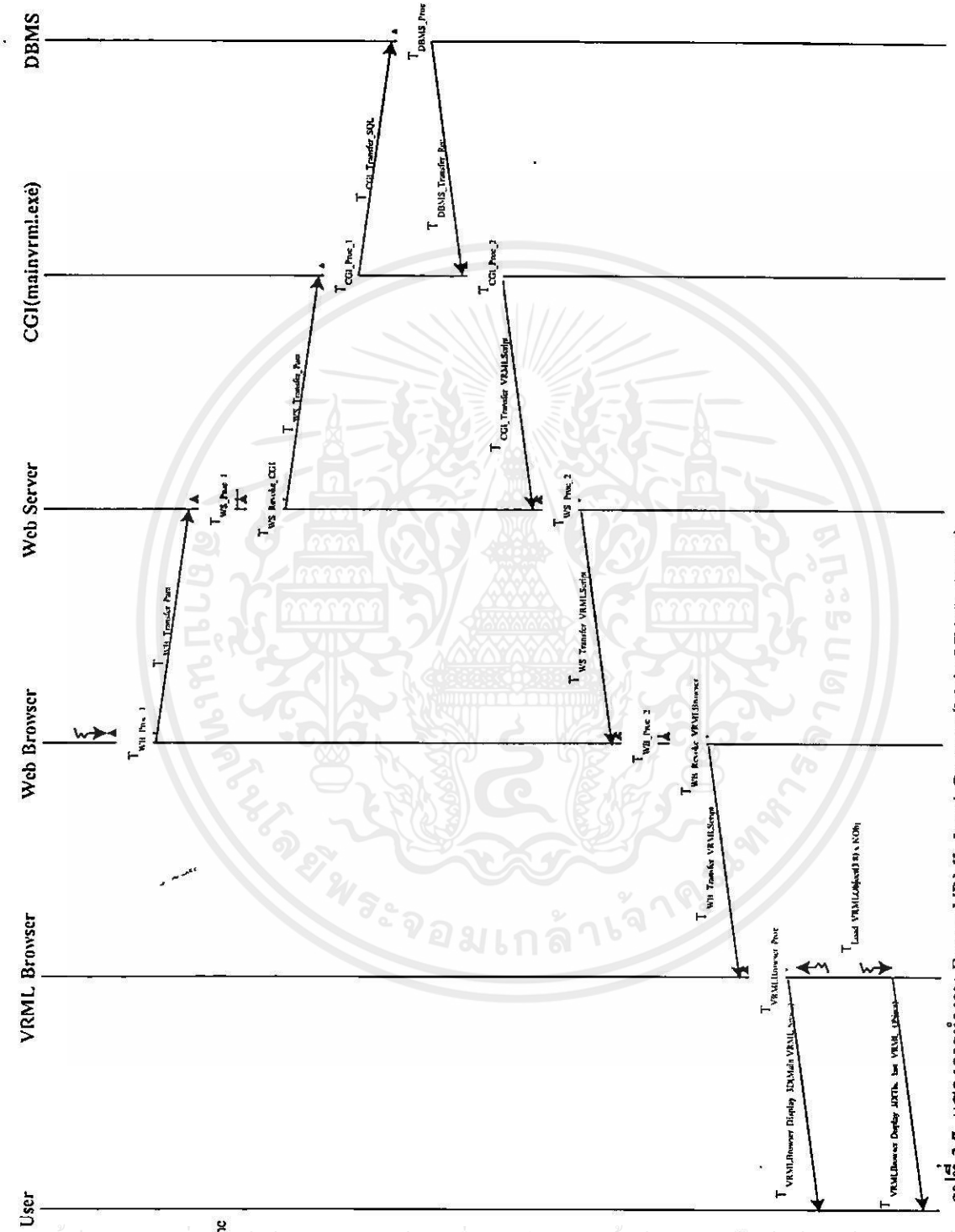
รูปที่ 3.4 แสดงการทำงาน Frame Header Load Content



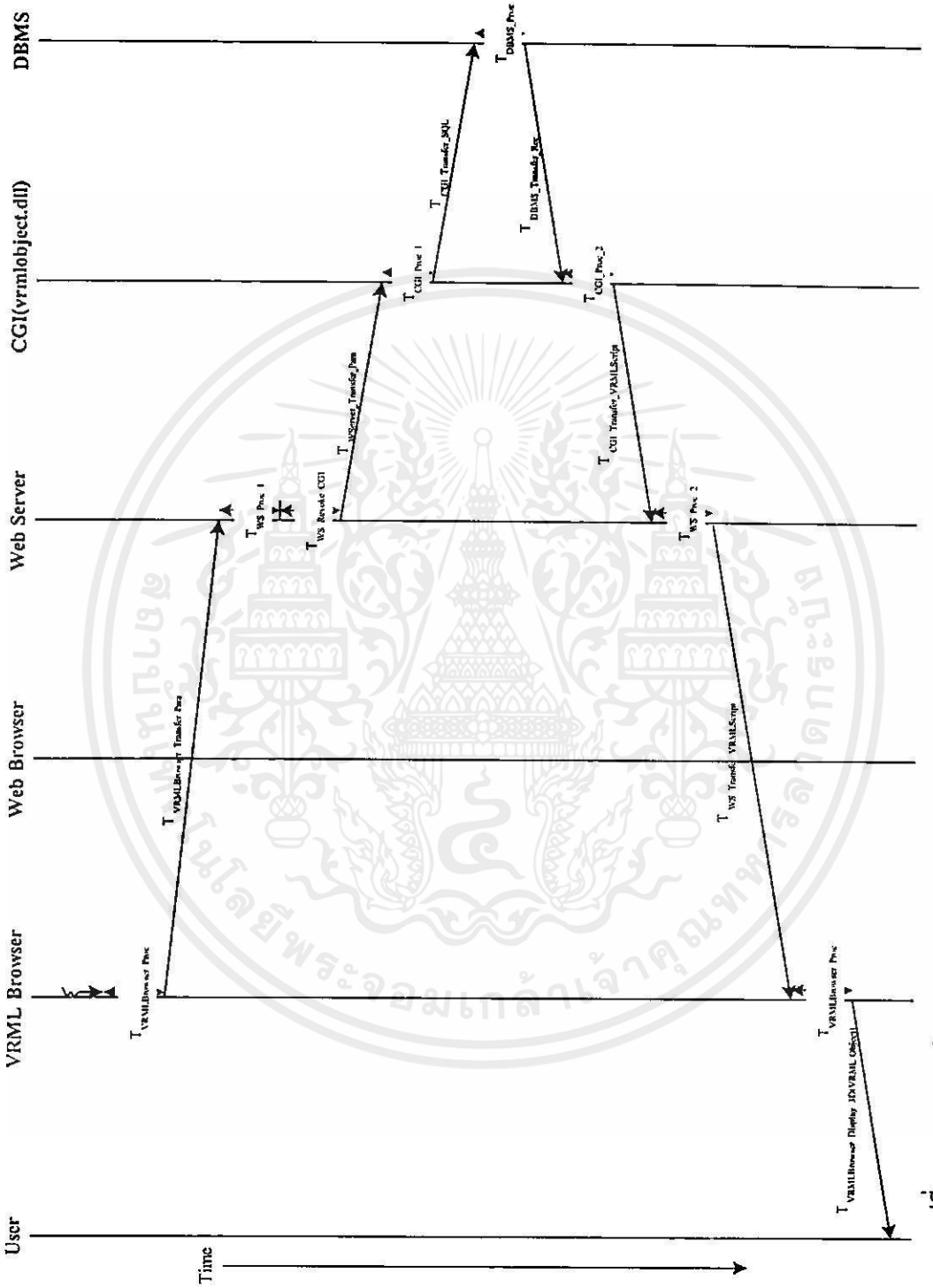
รูปที่ 3.5 แสดงการทำงาน Frame Footer Load Content



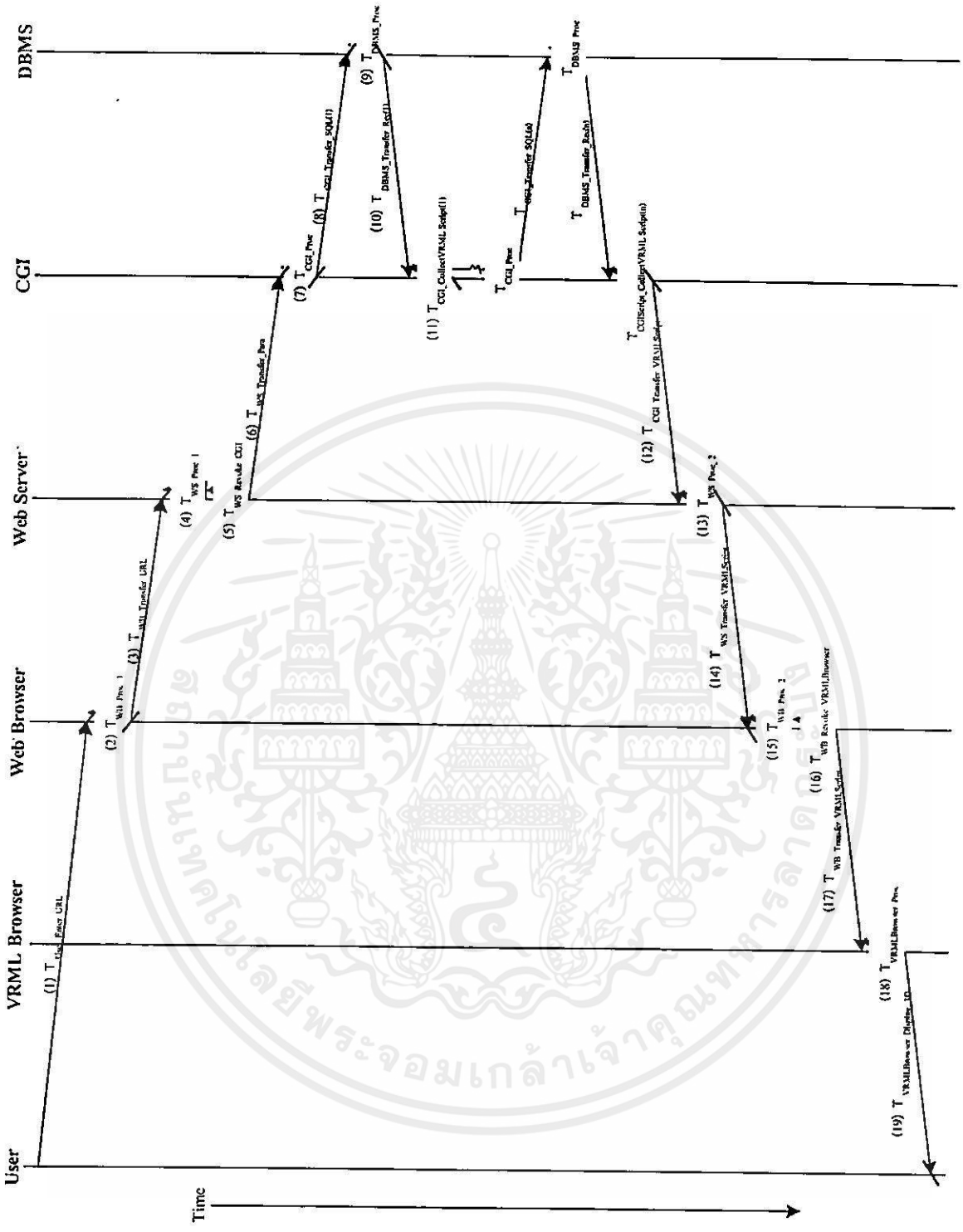
รูปที่ 3.6 แสดงการทำงาน Frame Menu Load Content



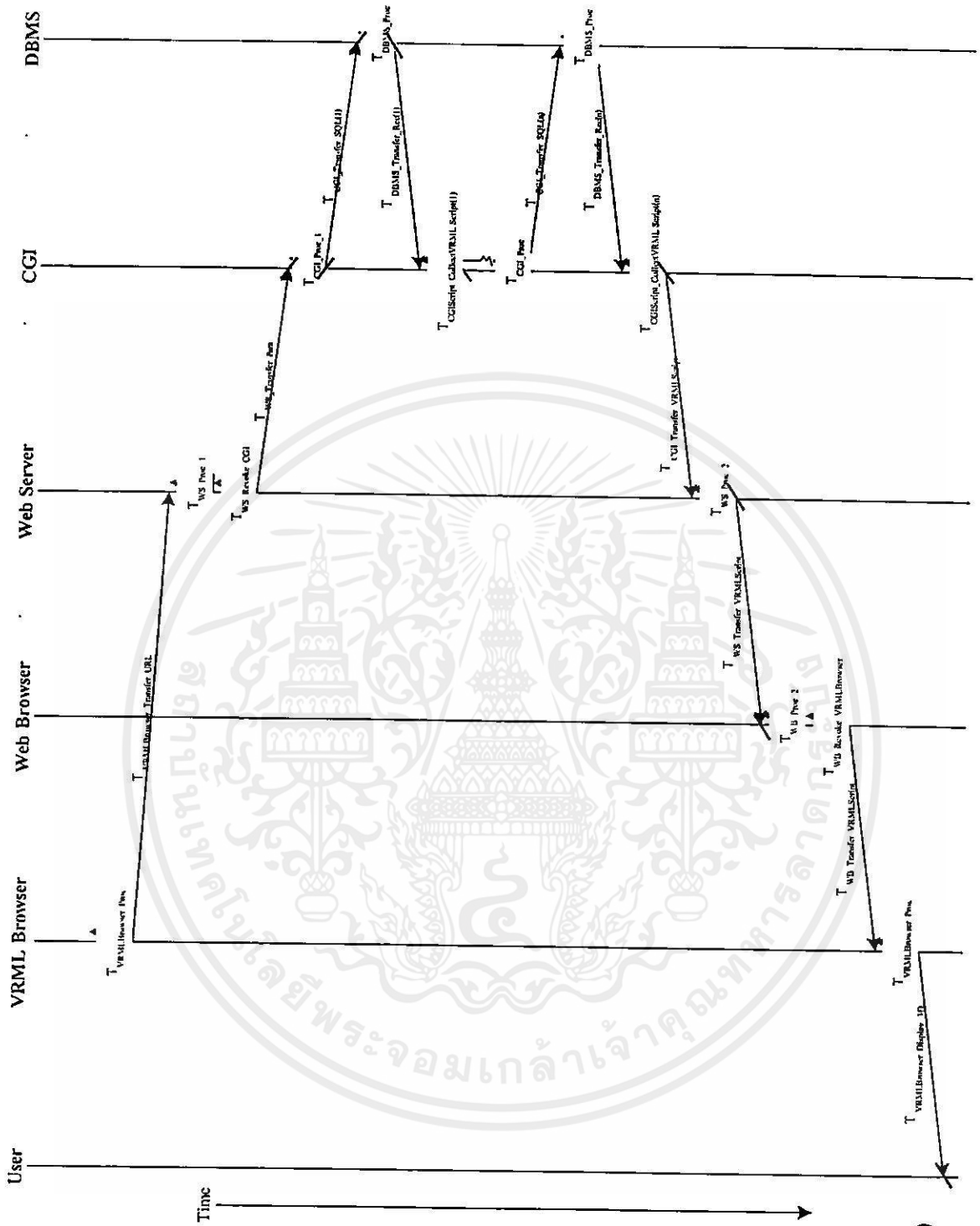
รูปที่ 3.7 แสดงการทำงาน Frame VRML Load Content (Main VRML Space)



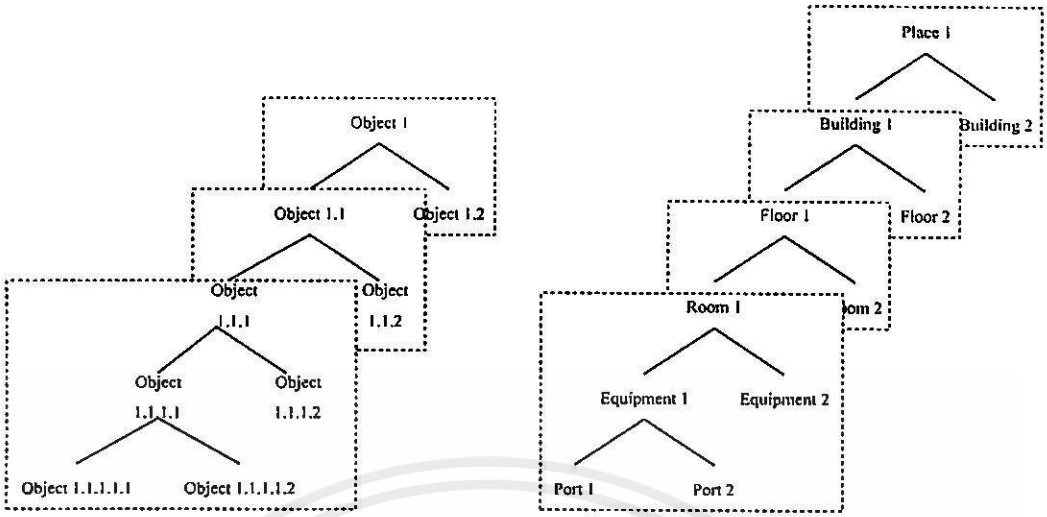
รูปที่ 3.8 แสดงการทำงาน Frame VRML Load Content (VRMLObject)



รูปที่ 3.9 แสดงการทำงาน Request Page (1)



รูปที่ 3.10 แสดงการทำงาน Request Page (2)



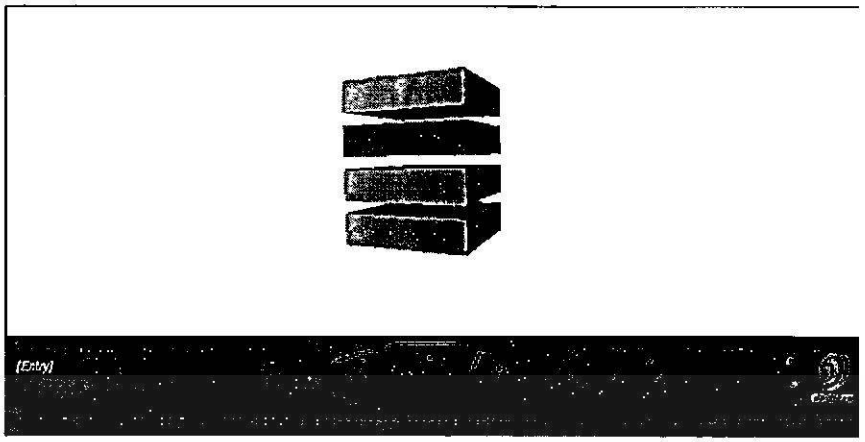
รูปที่ 3.11 แสดงตัวอย่างแผนภาพลำดับต้นไม้ในการนำเสนอข้อมูลตาม View ต่างๆ

จากรูปที่ 3.11 ถ้าผู้ใช้ระบุ Viewpoint ที่ ObjectId=1 และระบุในฐานข้อมูลว่าต้องการดูออบเจ็กต์ต่างๆ ภายใต้ Object1 ลงไป 1 ระดับ (ระบุในฟิลด์ Always_show_children = 'Yes' ในตาราง Object ที่ OjbectId=1) โปรแกรมจะแสดงผลของ Object1, Object1.1 และ Object1.2 เป็นต้น อย่างเช่น

- ถ้าค่าของพารามิเตอร์ objectid เป็น id ของออบเจ็กต์ที่เป็นสถานที่ (Place) ผลลัพธ์ที่ได้จะเป็นออบเจ็กต์ของอาคาร (Building) ทั้งหมดภายในสถานที่นั้น โดยใช้ออบเจ็กต์ Box 1 อันแทนอาคาร 1 อาคาร ดังตัวอย่างรูปที่ 3.12

รูปที่ 3.12 แสดงภาพตัวอย่างการเลือกระดับสถานที่

- ถ้าค่าของพารามิเตอร์ objectid เป็น id ของออบเจ็กต์ที่เป็นอาคาร (Building) ผลลัพธ์ที่ได้จะเป็นออบเจ็กต์ของชั้น (Floor) ทั้งหมดภายในอาคารนั้น โดยใช้ออบเจ็กต์ Box 1 อันแทนชั้น 1 ชั้น ดังรูปตัวอย่างที่ 3.13



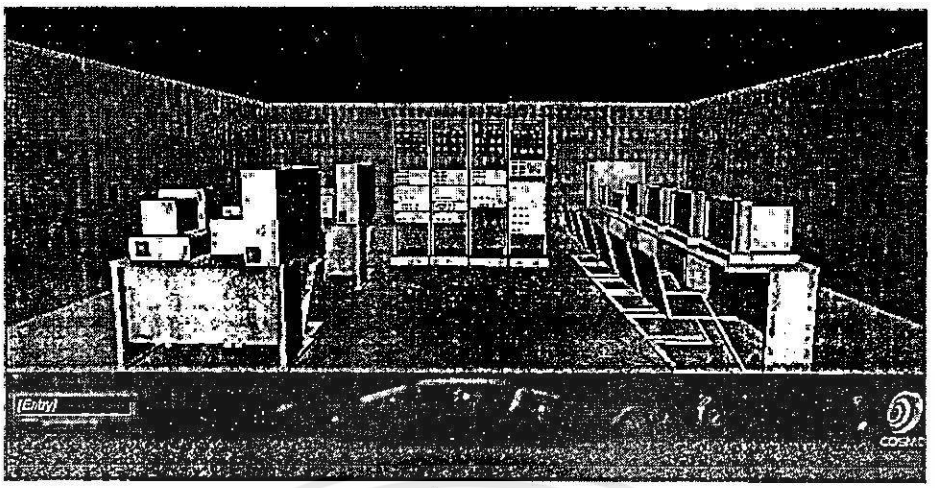
รูปที่ 3.13 แสดงภาพตัวอย่างการเลือกระดับอาคาร

- ถ้าค่าของพารามิเตอร์ objectid เป็น id ของออบเจ็กต์ที่เป็นชั้น (Floor) ผลลัพธ์ที่ได้จะเป็นออบเจ็กต์ของห้อง (Room) ทั้งหมดภายในชั้นนั้น โดยใช้ออบเจ็กต์ Box 1 อันแทนห้อง 1 ห้อง ดังรูปตัวอย่างที่ 3.14

รูปที่ 3.14 แสดงภาพตัวอย่างการเลือกระดับชั้น

- ถ้าค่าของพารามิเตอร์ objectid เป็น id ของออบเจ็กต์ที่เป็นห้อง (Room) ผลลัพธ์ที่ได้จะเป็นออบเจ็กต์ของอุปกรณ์ทั้งหมดภายในชั้นนั้นรวมถึงพอร์ทของอุปกรณ์ด้วย โดยนำเสนอในรูปแบบที่เหมือนจริง ดังรูปตัวอย่างที่ 3.15

<u>หมายเหตุ</u> ผนังห้องถือเป็น Child object หนึ่งภายในออบเจ็กต์ห้อง (Parent object)



รูปที่ 3.15 แสดงภาพตัวอย่างการเลือกระดับห้อง

11. เมื่อเว็บเซิร์ฟเวอร์ได้รับการร้องขอจากไคลเอนต์แล้วเว็บเซิร์ฟเวอร์จะกระตุ้นการทำงานของเกต์เวย์โปรแกรม (CGI) และส่งผ่านข้อมูลที่จำเป็นสำหรับการสืบค้นข้อมูลตามที่ผู้ใช้ระบุส่งไปยังเกต์เวย์โปรแกรม
12. เกต์เวย์โปรแกรมจะทำการประมวลผลตามที่ถูกกำหนดไว้กับเว็บเซิร์ฟเวอร์ฐานข้อมูล
13. เมื่อระบบเว็บเซิร์ฟเวอร์ฐานข้อมูลได้รับข้อมูลที่จำเป็นสำหรับการประมวลผลจากเกต์เวย์โปรแกรม โดยระบบเว็บเซิร์ฟเวอร์ฐานข้อมูลจะมองข้อมูลที่มาจากเกต์เวย์โปรแกรมเป็นทรานเซคชั่น เมื่อเว็บเซิร์ฟเวอร์ฐานข้อมูลทำการประมวลผลและได้ข้อมูลตามที่ต้องการแล้วจะส่งข้อมูลที่ได้ออกไปยังเกต์เวย์โปรแกรม โดยการติดต่อกับเว็บเซิร์ฟเวอร์ฐานข้อมูลอาจฝังภาษาที่เรียกใช้ข้อมูลที่สามารถระบุความต้องการได้ เช่น ภาษา SQL (Structured Query Language) ไว้ใน CGI Script โดยการทำงานจะเริ่มตั้งแต่ผู้ใช้ส่งข้อความร้องขอข้อมูลจากเครื่องลูกหรือเครื่องที่เป็นไคลเอนต์ไปยังเครื่องเว็บเซิร์ฟเวอร์ ต่อจากนั้นโปรแกรม CGI บนเว็บเซิร์ฟเวอร์ก็จะแยกส่วนที่เป็นภาษา SQL ออกมา และส่งส่วนที่เป็นภาษา SQL ไปให้กับระบบการจัดการฐานข้อมูล และผลลัพธ์ที่ได้จะถูกส่งกลับมาในรูปของภาษา HTML คืนให้กับเครื่องไคลเอนต์
14. เมื่อเกต์เวย์โปรแกรมได้รับผลที่ได้จากเว็บเซิร์ฟเวอร์ฐานข้อมูลแล้ว เกต์เวย์โปรแกรมจะส่งผ่านข้อมูลเหล่านั้นไปยังเว็บเซิร์ฟเวอร์
15. เมื่อเว็บเซิร์ฟเวอร์ได้รับผลลัพธ์ที่ส่งมาจากเกต์เวย์โปรแกรมแล้ว เว็บเซิร์ฟเวอร์ก็จะส่งผ่านข้อมูลเหล่านั้นไปยังเว็บเบราเซอร์ในรูปของ HTML Page
16. ในระบบนี้สามารถแบ่งสคริปต์ที่ให้ผลลัพธ์เป็น VRML ออกได้เป็น 2 ประเภท คือ
    - สคริปต์หลักที่ใช้ในการสร้าง VRML Space หรือโครงของแต่ละพื้นที่ (ดังรูปที่ 3.7)

- สคริปต์ย่อยที่ใช้ในการสร้าง VRML ออบเจ็กต์ต่างๆ ที่เกี่ยวข้องกับ VRML Space นั้นๆ หรือโครงของแต่ละพื้นที่ (ดังรูปที่ 3.8)

โดยทั้ง 2 ประเภทจะมีลักษณะการทำงานร่วมกัน คือ เมื่อสคริปต์หลักถูกเรียกใช้ สคริปต์จะทำการประมวลผลโดยการดึงข้อมูลของออบเจ็กต์ต่างๆ ที่เกี่ยวข้องกับพื้นที่นั้นๆ จากฐานข้อมูลแล้วนำมาประกอบขึ้นเป็น URL เพื่อใช้เรียกสคริปต์ย่อยในการสร้างออบเจ็กต์ต่างๆ และเพื่อให้สามารถดูผลลัพธ์ที่เป็น VRML ได้จะต้องมี VRML Browser เพื่อทำหน้าที่ตีความ VRML ไปเป็นภาพแบบกราฟฟิก

17. เมื่อไคลเอนต์กระทำการใดๆ บนเว็บเพจ อันเป็นการไปกระตุ้นให้ CGI scripts ที่ทำหน้าที่อัพเดตข้อมูลในฐานข้อมูลทำงานแล้ว (ดังรูปที่ 3.16) โมดูลที่เป็น Client Socket ซึ่งฝังติดอยู่ใน CGI Scripts นั้น จะขอเปิดการติดต่อและส่งข่าวสารไปยังแอพเพล็ตที่เป็น Server Socket ที่รันอยู่ทางฝั่งไคลเอนต์ตาม IP Address และ ID ของเว็บเบราเซอร์ที่อ่านมาได้จากฐานข้อมูลซึ่งมีการบันทึกไว้ในขั้นตอนที่ 4 โดยที่เครื่องไคลเอนต์จะต้องมีแอดแดรสที่ได้ลงทะเบียนถูกต้อง (Registered) ซึ่งเครื่องไคลเอนต์อาจจะติดต่อตรงไปหาเว็บเซิร์ฟเวอร์หรือติดต่อผ่านพร็อกซีก็ได้
18. เมื่อแอพเพล็ตที่เป็น Server Socket ทางฝั่งไคลเอนต์ได้รับข่าวสารแจ้งว่าฐานข้อมูลมีการเปลี่ยนแปลง ผู้ใช้ก็จะทำการสั่งให้เว็บเบราเซอร์ทำการโหลดใหม่อีกหนึ่งครั้ง
19. และเมื่อผู้ใช้ได้รับเว็บเพจครบถ้วนแล้ว ผู้ใช้อาจต้องการตรวจสอบระบบหรือสำรวจพื้นที่รอบๆ บริเวณการจัดการเครือข่ายในโลก 3 มิติ เพียงแค่ผู้ใช้ระบุความต้องการผ่าน VRML Browser เพื่อส่งคำสั่งความต้องการให้กับ VRML Browser ประมวลผลออกมาในรูปแบบกราฟฟิก 3 มิติ ดังรูปการทำงานที่ 3.17
20. สุดท้ายเมื่อผู้ใช้ทำการ Logout (ดังรูปที่ 3.18) ออกจากระบบก็จะเป็นการไปเรียก CGI Scripts ที่ทำหน้าที่ในการ Logout ให้ทำงาน ซึ่ง CGI Scripts ดังกล่าวมีหน้าที่ในการตรวจสอบว่าไคลเอนต์เครื่องใดได้มีการ Logout ออกจากระบบและจะทำการลบ IP Address และ ID ของเว็บเบราเซอร์ของเครื่องนั้นออกจากฐานข้อมูลเพื่อลดปริมาณการส่งข่าวสารไปหาไคลเอนต์เมื่อมีการเปลี่ยนแปลงฐานข้อมูล

3.2.3 หลักการแม็พข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ

เนื่องจากคุณสมบัติที่สำคัญอย่างหนึ่งของการจัดการเครือข่ายผ่านเว็บด้วย VRML นี้คือการอ้างอิงวัตถุกับตำแหน่งที่ตั้งจริงทางกายภาพ ดังนั้นจุดประสงค์ของการแม็พข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ ก็คือ การนำเสนอข้อมูลด้านฟิสิกคอลของวัตถุและสภาพแวดล้อมอื่นๆ ที่เกี่ยวข้องให้อยู่ในรูปแบบของวัตถุที่เหมือนจริงหรือใกล้เคียงความจริงทั้งด้านรูปร่างลักษณะและตำแหน่งการจัดวางรวมทั้งสถานะต่างๆ ของวัตถุ ซึ่งแปรผันไปตามข้อมูลด้านลอจิคอล

เพื่อให้ผู้ที่มีหน้าที่ในการดูแลระบบเครือข่ายสามารถทราบถึงตำแหน่งที่อยู่และสถานะของอุปกรณ์ได้ง่ายและสามารถเข้าไปดำเนินการกับอุปกรณ์ต่างๆ ได้รวดเร็วและถูกต้องมากขึ้น โดยนำความสามารถของระบบจัดการฐานข้อมูล, VRML และ CGI มาใช้เป็นเครื่องมือในการพัฒนางานในส่วนนี้ ดังแสดงหลักการแม็พข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติในรูปที่ 3.19 ซึ่งความหมายของข้อมูลฟิสิกคอลและลอจิคอลและกระบวนการในการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ คือ

3.2.3.1 ข้อมูลฟิสิกคอล

ข้อมูลฟิสิกคอล คือ วัตถุต่างๆ ที่เห็นในสภาพแวดล้อมที่เราสนใจและต้องการนำเสนอในรูปแบบ 3 มิติ เช่น สายและอุปกรณ์ที่ใช้ในการเชื่อมต่อคอมพิวเตอร์ในระบบเครือข่าย โดยพิจารณาวัตถุต่างๆ ในสภาพแวดล้อมให้สัมพันธ์กัน โดยพยายามกรองเฉพาะวัตถุที่มีความเกี่ยวข้องกับตำแหน่งการจัดวางของวัตถุ รวมทั้งตัววัตถุเอง โดยพิจารณาวัตถุต่างๆ ที่มีความสัมพันธ์กันในลักษณะของ Top-down model คือ แตกวัตถุออกไปเป็นลักษณะความสัมพันธ์แบบ one-to-many ซึ่งวัตถุที่ถูกแตกออกมาตามความสัมพันธ์นั้นจะต้องมีความหมายได้ในตัวเอง เช่น เริ่มพิจารณาจากสถานที่ คือ 1 สถานที่มีได้หลายอาคาร, 1 อาคารมีได้หลายชั้น, 1 ชั้นมีได้หลายห้อง, 1 ห้องมีอุปกรณ์ได้หลายชิ้น เป็นต้น เมื่อสามารถแยกวัตถุต่างๆ ออกได้ตามความสัมพันธ์ดังกล่าวแล้วก็สามารถนำวัตถุและความสัมพันธ์ที่ได้ไปทำการหารายละเอียดทางด้านฟิสิกคอลของวัตถุเพื่อทำการออกแบบฐานข้อมูลในลักษณะฐานข้อมูลรีเลชัน

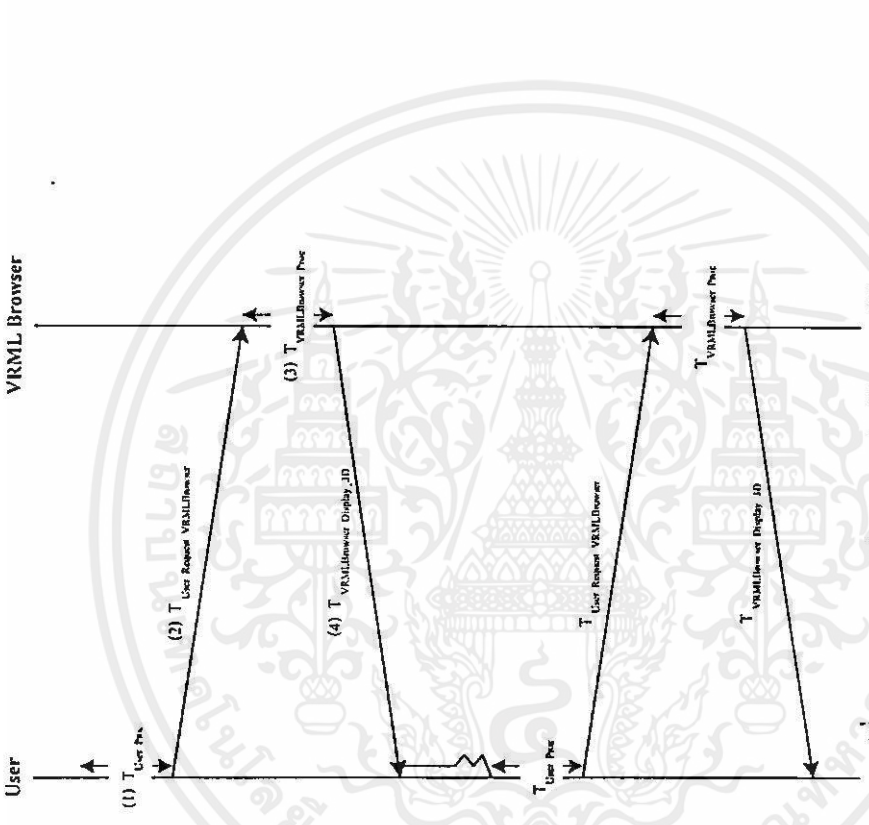
3.2.3.2 ข้อมูลลอจิคอล

หลังจากทำการรวบรวมข้อมูลทางฟิสิกคอลของวัตถุและสภาพแวดล้อมต่างๆ แล้ว ต่อจากนั้นจึงทำการรวบรวมข้อมูลทางลอจิคอลของวัตถุ โดยข้อมูลลอจิคอล อย่างเช่นในกรณีของการจัดการเครือข่ายนั้น ฐานข้อมูลลอจิคอล คือฐานข้อมูลทั้งหมดในเครือข่ายที่ดึงจาก Management Information Base (MIB), รูปแบบการเชื่อมต่อเครือข่าย, โปรโตคอลที่ใช้ รวมถึงคุณสมบัติและรายละเอียดต่างๆ ที่จำเป็นในการจัดเก็บลงในฐานข้อมูลเพื่อนำมาเป็นข้อมูลพื้นฐานในการสร้างรูป 3 มิติ

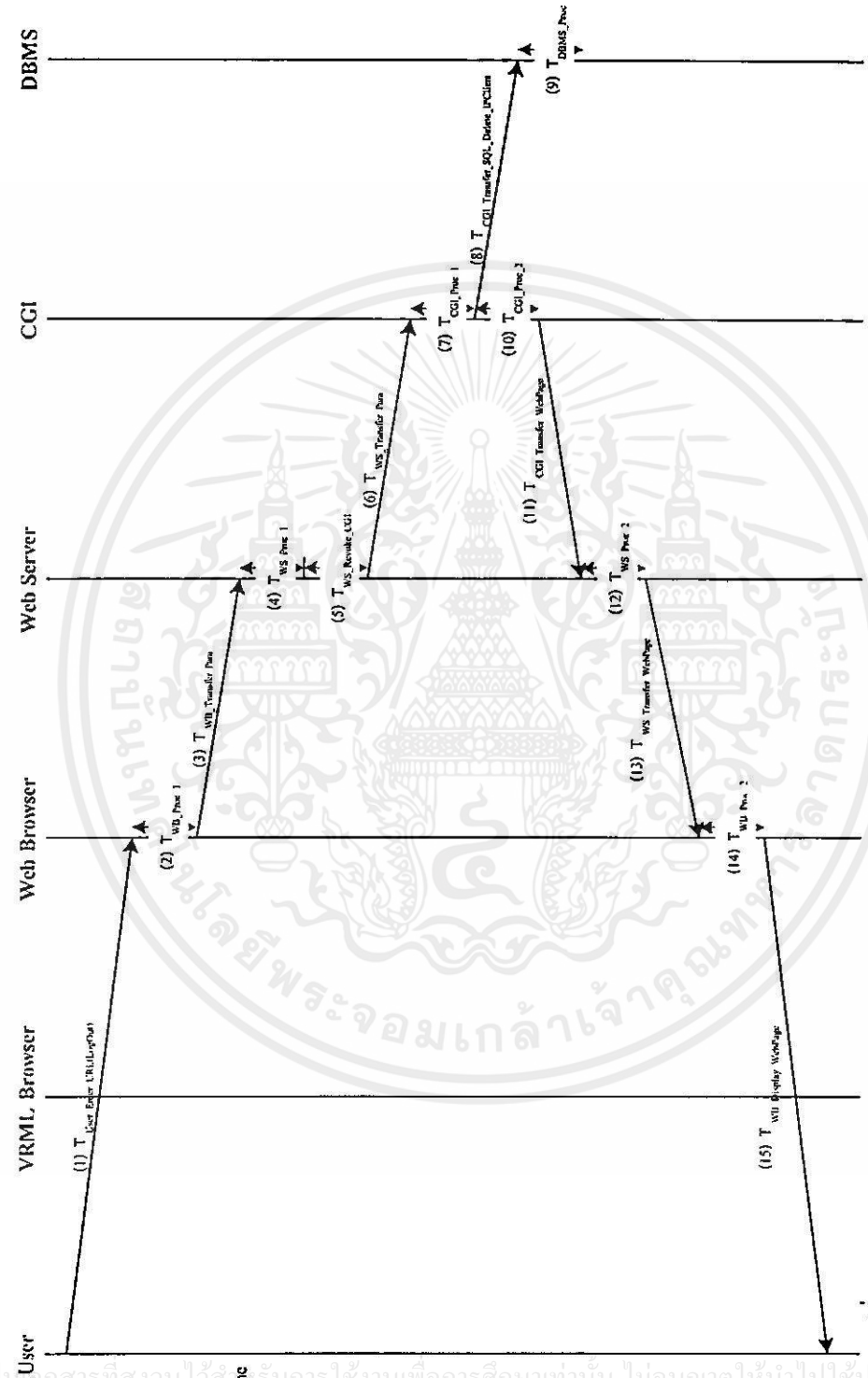
3.2.3.3 กระบวนการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ

กระบวนการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ แสดงดังรูปที่ 3.20 และมีขั้นตอนดังต่อไปนี้

1. จัดคลาสหรือกลุ่มให้กับวัตถุที่จะทำการแม็พ เช่น Building, Room, Hub, Server, Workstation เป็นต้น
2. พิจารณาหาคุณสมบัติทั่วไปที่สนใจของแต่ละคลาส เช่น ชื่อ, ชนิด, ความกว้าง, ความยาว, ความสูง เป็นต้น เพื่อเป็นข้อมูลในการออกแบบฐานข้อมูล

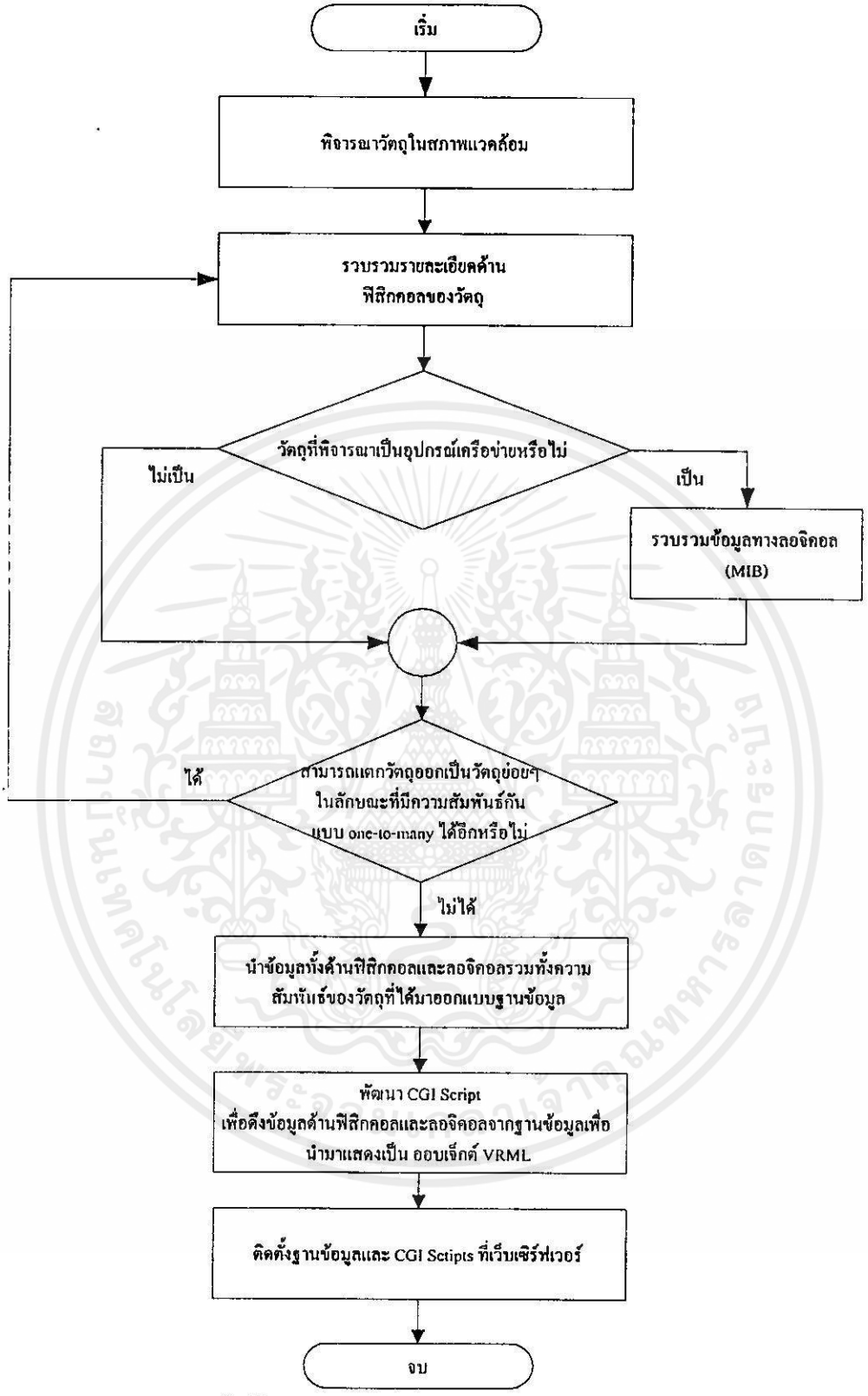


รูปที่ 3.17 แสดงการทำงานการท่องโลกสามมิติ (Navigate)

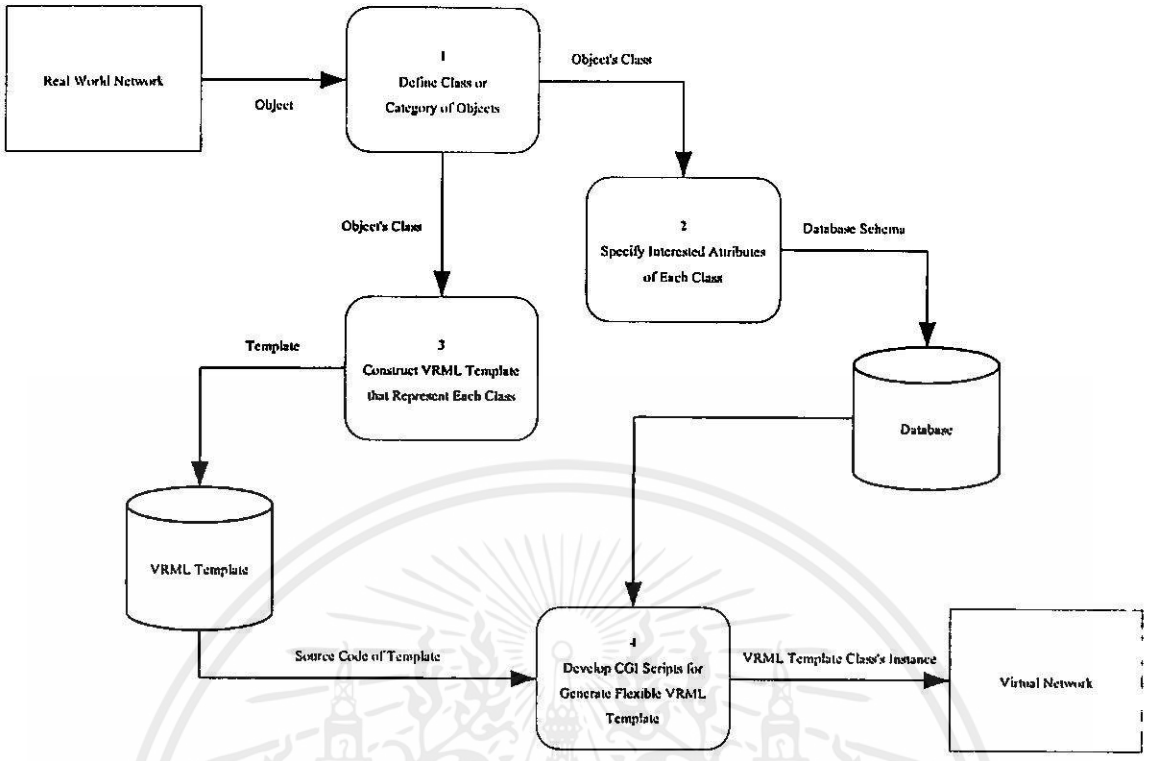


รูปที่ 3.18 แสดงการทำงาน User Logout

3. นำข้อมูลต่างๆ ที่ได้พิจารณามาจัดทำเทมเพล็ตของ VRML ที่จะใช้นำเสนอแทนคลาสเหล่านั้น
4. สร้างฐานข้อมูลสำหรับทำการจัดเก็บค่าคุณสมบัติต่างๆ ของคลาสเหล่านั้น โดยจะได้กล่าวถึงการออกแบบฐานข้อมูลในหัวข้อที่ 3.4
5. พัฒนา CGI Script สำหรับสร้างเทมเพล็ต VRML ที่ได้จากข้อ 3 ให้สามารถยืดหยุ่นได้ตามคุณสมบัติเฉพาะของแต่ละคลาส เช่น ในด้านของความกว้าง, ความยาว, ความสูง, ตำแหน่งการจัดวาง เป็นต้น
6. เมื่อทำการพัฒนาสคริปต์ต่างๆ เรียบร้อยแล้วจึงทำการทดลองใช้งาน โดยนำสคริปต์ที่พัฒนาขึ้นมาได้เก็บไว้ในเว็บเซิร์ฟเวอร์ แล้วให้ผู้ใช้ทำการร้องขอการทำงานที่ต้องการผ่านเว็บเบราเซอร์



รูปที่ 3.19 แสดงหลักการแม็พข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ



รูปที่ 3.20 แสดงกระบวนการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ

3.3 ขั้นตอนการทำงานในการรับข่าวสารของไคลเอนต์และการลบ IP ของเซิร์ฟเวอร์

จากหัวข้อที่ 3.2.2 ได้กล่าวถึงการพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ว่า แอพเพล็ต Server Socket ที่รันอยู่ฝั่งไคลเอนต์ในส่วนหน้าจอที่เป็น Footer จะมีหน้าที่รอรับข่าวสารที่จะถูกส่งมาจาก Client Socket ที่รันอยู่ฝั่งเซิร์ฟเวอร์ โดยจะคอยฟังข่าวสารอยู่ตลอดเวลา เนื่องจากข้อมูลที่นำมาแสดงเป็นข้อมูลเชิงไดนามิก เพราะเป็นข้อมูลของเครือข่ายที่มีการเปลี่ยนแปลงตลอดเวลา ดังนั้นจึงต้องมีการพัฒนาในส่วนที่จะทำให้ผู้ใช้ทราบถึงความเปลี่ยนแปลงของฐานข้อมูลเครือข่าย โดยระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML จะต้องสามารถรับข่าวสารจากเว็บเซิร์ฟเวอร์ได้ และได้กล่าวถึง CGI Script ที่ทำหน้าที่ในการ Login ว่าจะทำการตรวจสอบ Login Name และ Password ถ้าถูกต้องจะทำการบันทึก IP Address และ ID ของเว็บเบราเซอร์ของเครื่องไคลเอนต์ที่ทำการ Login ลงในฐานข้อมูลและเมื่อผู้ใช้ทำการ Logout ออกจากระบบก็จะเป็นการไปเรียก CGI Scripts ที่ทำหน้าที่ในการ Logout ให้ทำงาน ซึ่ง CGI Scripts ดังกล่าวมีหน้าที่ในการตรวจสอบว่าไคลเอนต์เครื่องใดได้มีการ Logout ออกจากระบบและจะทำการลบ IP Address และ ID ของเว็บเบราเซอร์ของเครื่องนั้นออกจากฐานข้อมูลเพื่อลดปริมาณการส่งข่าวสารไปหาไคลเอนต์เมื่อมีการเปลี่ยนแปลงฐานข้อมูล แต่ถ้าผู้ใช้ออกจากระบบหรือทำการปิดเว็บเบราเซอร์โดยไม่ได้ Logout ดังนั้นระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML จำเป็นที่จะต้องมีการพัฒนาส่วนประกอบบางอย่างที่มีความสามารถในการฝัง

ตัวอยู่บนเว็บเพจและสามารถที่จะเปิดช่องทางการติดต่อสื่อสารกับเว็บเซิร์ฟเวอร์ได้ด้วยตัวเอง ซึ่งเมื่อพิจารณาดูถึงความสามารถที่ต้องการนี้แล้วก็พบว่าเครื่องมือหรือภาษาที่เหมาะสมและมีความสามารถเพียงพอที่จะพัฒนาส่วนประกอบดังกล่าวนี้ ได้แก่ ภาษาจาวา โดยสามารถที่จะนำความสามารถของภาษาจาวา ในเรื่องของจาวาแอพเพล็ตและจาวาเน็ตเวิร์กมาประยุกต์ใช้กับความต้องการดังกล่าวได้เป็นอย่างดี

3.3.1 การประยุกต์ใช้จาวาแอพเพล็ตและจาวาเน็ตเวิร์กในระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

จากที่ได้กล่าวถึงหลักการทำงานของจาวาแอพเพล็ตและจาวาเน็ตเวิร์กในบทที่ 2 ในหัวข้อ 2.13 มาแล้วนั้น ต่อไปนี้เป็นหลักการที่นำมาประยุกต์ใช้เพื่อทำการพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML มีดังต่อไปนี้

1. การเพิ่มความสามารถให้ไคลเอนต์สามารถรับข่าวสารจากเว็บเซิร์ฟเวอร์ได้ เมื่อมีการเปลี่ยนแปลงข้อมูลทางฝั่งเว็บเซิร์ฟเวอร์ โดยการพัฒนาแอพเพล็ตเพื่อฝังตัวกับเว็บเบราเซอร์ของไคลเอนต์ไว้ ซึ่งแอพเพล็ตนี้จะทำหน้าที่ในการสร้าง Server Socket เพื่อคอยรับการติดต่อสื่อสารจาก Client Socket ทางฝั่งเว็บเซิร์ฟเวอร์ โดยให้แอพเพล็ตทำการสร้าง Server Socket เมื่อเริ่มทำงานเป็นครั้งแรก ซึ่งสามารถทำได้โดยการนำส่วนของโปรแกรมที่ทำการสร้าง Server Socket ไปไว้ใน method init() ของแอพเพล็ต ซึ่ง method นี้จะถูกเรียกให้ทำงานเองอัตโนมัติ เมื่อแอพเพล็ตเริ่มต้นการทำงานเป็นครั้งแรกหรืออยู่ในสถานะ init นั่นเอง ดังตัวอย่าง

```
:
public void init() {
    try {
    ServerSocket s = new ServerSocket(port_number);
    Socket incomming = s.accept();
    :
    }
catch (Exception e) {}
:
```

ในขณะที่ทางฝั่งเว็บเซิร์ฟเวอร์ เมื่อมีการเปลี่ยนแปลงข้อมูลแล้วต้องการแจ้งข่าวสารออกไปให้ไคลเอนต์ทราบก็จะมีการเปิด Socket (Client Socket) ไปยัง Server Socket ตามพอร์ทที่ถูกกำหนดไว้ ตามแต่ละไคลเอนต์ ดังตัวอย่าง

```
:
Socket connection = new Socket(remote_address,port_number)
:
```

2. สำหรับการตรวจสอบการออกจากระบบหรือการปิดเว็บเบราเซอร์ โดยไม่ได้ Logout (ดังรูปที่ 3.21) ในระบบการพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ได้ทำการพัฒนาแอพเพล็ต ซึ่งจะทำหน้าที่ในการแจ้งข้อความไปให้เว็บเซิร์ฟเวอร์รู้ว่าเว็บเบราเซอร์ที่มันฝังตัวอยู่นั้นถูกปิดลง ซึ่งสามารถทำได้โดยการนำส่วนของโปรแกรมที่ทำหน้าที่ในการเปิด Socket (Client Socket) ไปหาเซิร์ฟเวอร์ที่มีการเปิด Server Socket ที่คอยรับฟังอยู่และในส่วนของโปรแกรมที่ทำการส่งแมสเสจซึ่งเป็นข้อมูลเกี่ยวกับการ Unique ID ของเว็บเบราเซอร์ (ในกรณีที่หนึ่งไคลเอนต์เปิดเว็บเบราเซอร์หลายหน้าต่าง) ไปไว้ใน method destroy() ของแอพเพล็ต ซึ่ง method นี้จะถูกเรียกให้ทำงานเองอัตโนมัติ เมื่อแอพเพล็ตนั้นสิ้นสุดการทำงานแล้ว ซึ่งจะเกิดขึ้นเมื่อเว็บเพจนั้นถูกยกเลิกการทำงาน เช่น เมื่อผู้ใช้ทำการปิดเว็บเบราเซอร์หรือแอพเพล็ตอยู่ในสถานะ destroy นั่นเอง ดังแสดงในตัวอย่าง

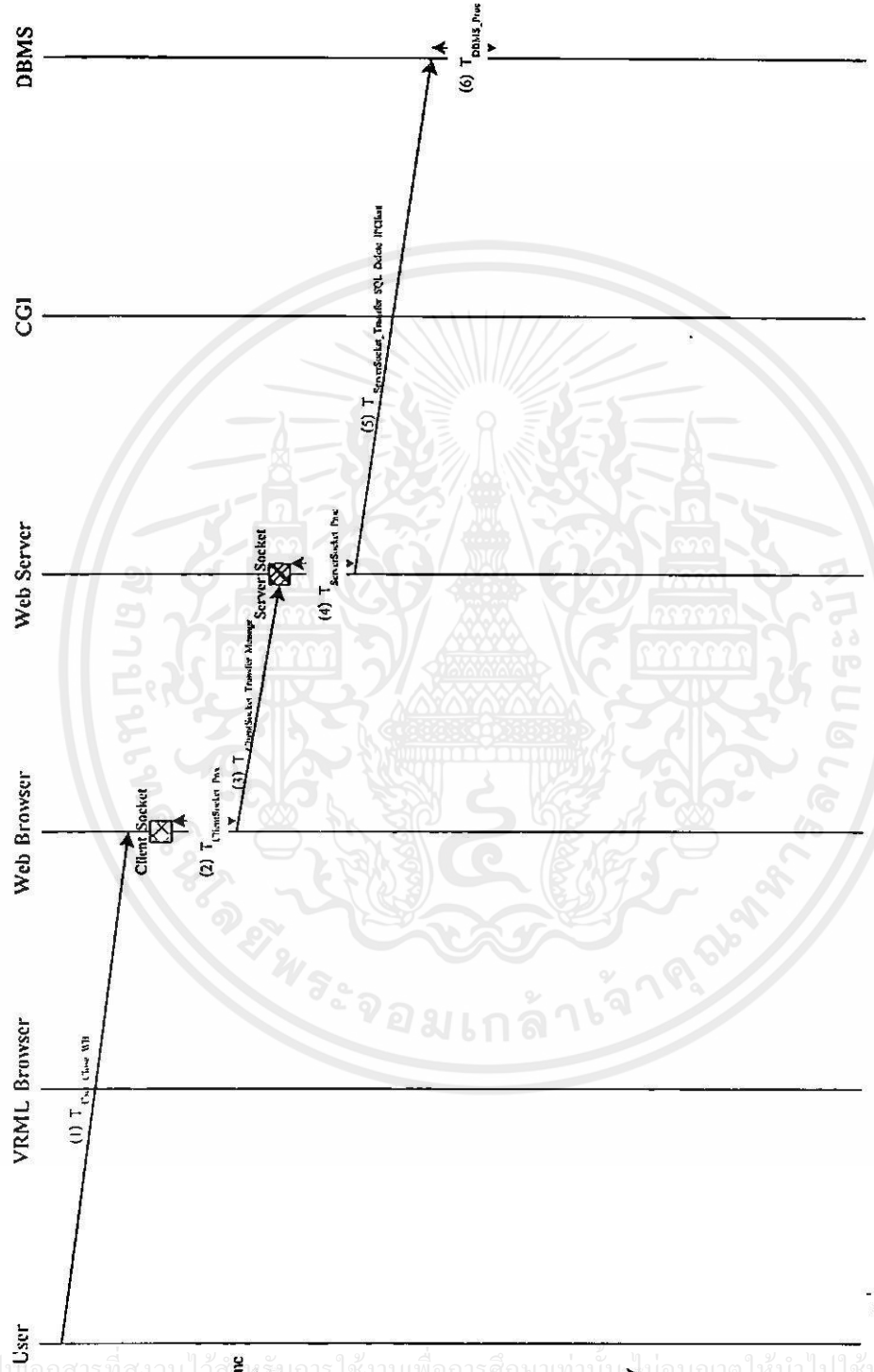
```
:
public void destroy()
    {
        try {
        Socket t = new Socket(remote_address,port_number);
        PrintStream out = new PrintStream(t.getOutputStream());
        out.println(message); //Information in message is Client's unique ID of
                              browser
        ;
    }
catch(IOException e) { }
:
```

เมื่อโปรแกรมฝั่งเว็บเซิร์ฟเวอร์ได้รับการติดต่อสื่อสารจากไคลเอนต์ที่เปิดไปแล้ว เว็บเซิร์ฟเวอร์ก็จะทราบหมายเลข IP ของไคลเอนต์ที่ติดต่อเข้ามา และเมื่อเว็บเซิร์ฟเวอร์ได้รับแมสเสจที่ถูกส่งมาจากไคลเอนต์แล้ว เว็บเซิร์ฟเวอร์ก็จะรู้ว่าเว็บเบราเซอร์ไหนของไคลเอนต์ถูกปิดไป แล้วเว็บเซิร์ฟเวอร์ก็จะทำการแก้ไขข้อมูลหมายเลข IP ของไคลเอนต์และ ID ของเว็บเบราเซอร์ที่ได้เคยติดต่อเข้ามาก่อนหน้านี้ในฐานข้อมูลต่อไป ดังแสดงในตัวอย่าง

```
:
public static void main(String[] args) {
    try {
     ServerSocket s = new ServerSocket(1999);
     Socket incomming = s.accept(); // Accept Socket from closed browser.
     :
      DataInputStream in = new DataInputStream(incoming.getInputStream());
      String clientip = incoming.getInetAddress().getHostAddress();//get client's IP
       String uport = in.readLine();//get unique port of client
          :
      String deletestmt = "delete from dbo.ConnectingClient where ip = '"+clientip+"'
and uport = '"+uport+"'"; // create sql statement for delete ip and uport of client
      int r = stmt.executeUpdate(deletestmt); // execute sql for delete
          :
      }
      catch (SQLException e)  {}
      catch (Exception e) {}
```

3.4 การออกแบบฐานข้อมูลของระบบ

จากกระบวนการในการสร้างข้อมูลฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ ในหัวข้อที่ 3.2.3.3 จะต้องมีการดึงข้อมูลจากฐานข้อมูลเพื่อนำเสนอข้อมูลฟิสิกคอลและลอจิคอลในรูปแบบ 3 มิติ ที่อิงกับตำแหน่งจริงทางกายภาพ ซึ่งในวิทยานิพนธ์นี้สามารถแบ่งประเภทของฐานข้อมูลในมุมมองของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ออกได้เป็น 3 ประเภท ได้แก่ ฐานข้อมูลลอจิคอล, ฐานข้อมูลฟิสิกคอลและฐานข้อมูลการแม็พระหว่างข้อมูลในฐานข้อมูลลอจิคอลและฟิสิกคอล โดยชนิดของข้อมูลลอจิคอลที่เก็บเช่น ชื่อวัตถุ, โปรโตคอลที่เชื่อมต่อภายใน

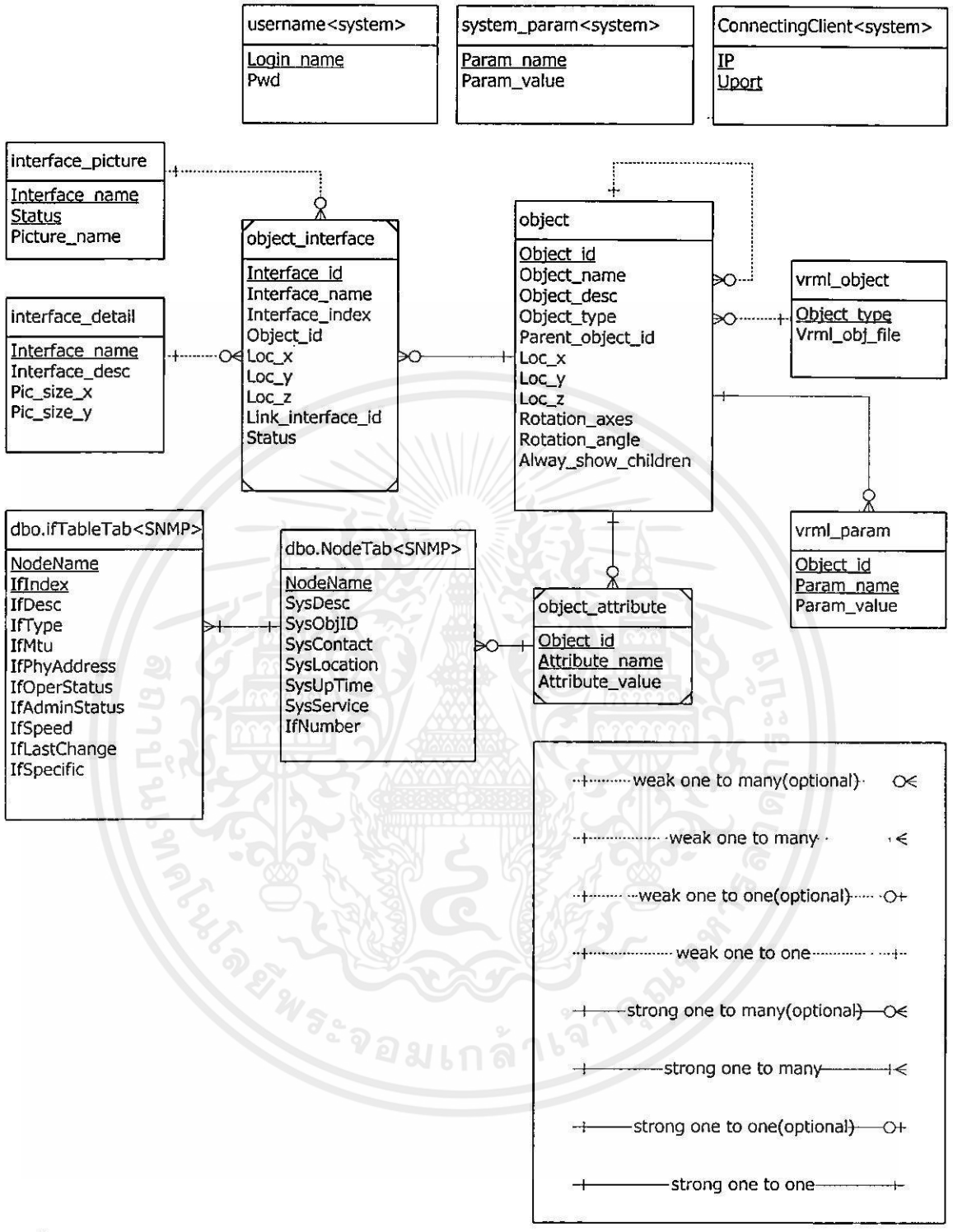


รูปที่ 3.21 แสดงการทำงาน User Close Browser without Logout

เครือข่าย, IP Address, สถานะของอุปกรณ์เครือข่าย, ชื่อโหนด เป็นต้น และชนิดของข้อมูลที่เก็บในฐานข้อมูลฟิสิกคอล ได้แก่ ข้อมูลคุณสมบัติทางกายภาพของวัตถุ เช่น ความกว้าง ความยาว ความสูงของวัตถุ, ตำแหน่งและมุมการวางของวัตถุในแนวแกน X, Y, Z ส่วนชนิดของข้อมูลในฐานข้อมูลการแม็พระหว่างข้อมูลในฐานข้อมูลลอจิคอลและฟิสิกคอล ได้แก่ ความสัมพันธ์ระหว่างฐานข้อมูลลอจิคอลและฟิสิกคอล, สคริปต์ CGI ที่ใช้ในการสร้างโมเดล 3 มิติ ซึ่งฐานข้อมูลทั้ง 3 ประเภทนี้จะถูกนำมาออกแบบให้อยู่ในรูปของฐานข้อมูลเชิงความสัมพันธ์ โดยมีโครงสร้างความสัมพันธ์ดังรูปที่ 3.22 และคำอธิบายของแต่ละเอ็นติตี้จากโครงสร้างความสัมพันธ์ดังตารางที่ 3.1

ตารางที่ 3.1 แสดงคำอธิบายของแต่ละเอ็นติตี้จากโครงสร้างความสัมพันธ์

Interface_Detail	เก็บรายละเอียดของอินเตอร์เฟซแต่ละประเภท
Interface_Picture	เก็บชื่อไฟล์รูปภาพของอินเตอร์เฟซแต่ละชนิด
Object	เก็บรายละเอียดของออบเจ็กต์
Object_Attribute	เก็บคุณสมบัติของออบเจ็กต์
Object_Interface	เก็บรายละเอียดอินเตอร์เฟซของออบเจ็กต์
System_Param	เก็บพารามิเตอร์ของระบบ
Username	เก็บพารามิเตอร์ของผู้ใช้
Vrml_Object	เก็บชนิดของออบเจ็กต์ VRML กับชื่อไฟล์เทมเพล็ต VRML ของออบเจ็กต์
Vrml_Param	เก็บพารามิเตอร์ของออบเจ็กต์ VRML
ConnectingClient	เก็บรายละเอียดของเครื่องไคลเอนต์ที่กำลังทำงานอยู่ในระบบ
NodeTab	เก็บข้อมูลรายละเอียดเชิงลอจิคอลของโหนดต่างๆ
IfTableTab	เก็บข้อมูลรายละเอียดเชิงลอจิคอลอินเตอร์เฟซของโหนดต่างๆ



รูปที่ 3.22 แสดงโครงสร้างความสัมพันธ์ของฐานข้อมูลเชิงสัมพันธ์ของระบบ

3.5 รายละเอียดตารางฐานข้อมูล

ในระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML มีการเก็บรายชื่อของทรัพยากรต่างๆ รวมถึงรายละเอียดของทรัพยากรที่ต้องการจัดการดังตัวอย่างต่อไปนี้

ตารางที่ 3.2 แสดงตารางข้อมูลรายละเอียดของอินเตอร์เฟซแต่ละประเภท : Interface_Detail

1	Interface_name	ชื่ออินเตอร์เฟซ	VARCHAR(30)	P.K.
2	Interface_desc	คำอธิบายของอินเตอร์เฟซ	VARCHAR(100)	
3	Pic_size_x	ขนาดของรูปอินเตอร์เฟซในแนวแกน X	VARCHAR(30)	
4	Pic_size_y	ขนาดของรูปอินเตอร์เฟซในแนวแกน Y	VARCHAR(30)	

ตารางที่ 3.3 แสดงตารางข้อมูลชื่อไฟล์รูปภาพของอินเตอร์เฟซแต่ละชนิด : Interface_Picture

1	Interface_name	ชื่ออินเตอร์เฟซ	VARCHAR(30)	P.K
2	Status	สถานะของอินเตอร์เฟซ	VARCHAR(10)	P.K.
3	Picture_name	ชื่อไฟล์รูปภาพของอินเตอร์เฟซ	VARCHAR(50)	

ตารางที่ 3.4 แสดงตารางข้อมูลรายละเอียดของออบเจ็กต์ : Object

1	Object_id	ID ของออบเจ็กต์ที่โปรแกรมสร้างขึ้นมา	INT	P.K.
2	Object_name	ชื่อออบเจ็กต์	VARCHAR(30)	
3	Object_desc	คำอธิบายของออบเจ็กต์	VARCHAR(50)	
4	Object_type	ชนิดของออบเจ็กต์	VARCHAR(30)	
5	Parent_object_id	ออบเจ็กต์นี้มี Parent Object ID คืออะไร	INT	
6	Link_object_id	ออบเจ็กต์นี้ลิงค์กับ Object ID ไหน	INT	
7	Loc_x	ตำแหน่งการวางออบเจ็กต์ในแนวแกน X	FLOAT	
8	Loc_y	ตำแหน่งการวางออบเจ็กต์ในแนวแกน Y	FLOAT	
9	Loc_z	ตำแหน่งการวางออบเจ็กต์ในแนวแกน Z	FLOAT	
10	Rotation_axes	แกนการหมุนของออบเจ็กต์	VARCHAR(1)	
11	Rotation_angle	มุมการหมุนของออบเจ็กต์	FLOAT	
12	Always_show_childen	ต้องการให้ออบเจ็กต์นี้แสดงออบเจ็กต์ลูกอื่นๆ พร้อมกันทันทีหรือไม่	BIT(1)	

ตารางที่ 3.5 แสดงตารางข้อมูลคุณสมบัติของออบเจ็กต์ : Object_Attribute

1	Object_id	ID ของออบเจ็กต์ที่โปรแกรมสร้างขึ้นมา	INT	P.K.
2	Attribute_name	ชื่อคุณสมบัติของออบเจ็กต์นี้	VARCHAR(30)	P.K.
3	Attribute_value	ค่าคุณสมบัติของออบเจ็กต์นี้	VARCHAR(30)	

ตารางที่ 3.6 แสดงตารางข้อมูลรายละเอียดอินเตอร์เฟซของออบเจ็กต์ : Object_Interface

ลำดับ	ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	คีย์
1	Interface_id	ID ของอินเตอร์เฟซที่โปรแกรมสร้างขึ้นมา	INT	P.K.
2	Interface_name	ชื่อของอินเตอร์เฟซ	VARCHAR(30)	
3	Interface_index	อินเด็กซ์ของอินเตอร์เฟซ	INT	
4	Object_id	ID ของออบเจ็กต์	INT	
5	Loc_x	ตำแหน่งการวางอินเตอร์เฟซในแนวแกน X	FLOAT	
6	Loc_y	ตำแหน่งการวางอินเตอร์เฟซในแนวแกน Y	FLOAT	
7	Loc_z	ตำแหน่งการวางอินเตอร์เฟซในแนวแกน Z	FLOAT	
8	Link_interface_id	อินเตอร์เฟซนี้ลิงค์กับอินเตอร์เฟซ ID ไหน	INT	
9	Status	สถานะของอินเตอร์เฟซ	VARCHAR(10)	

ตารางที่ 3.7 แสดงตารางข้อมูลพารามิเตอร์ของระบบ : System_Param

ลำดับ	ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	คีย์
1	Param_name	ชื่อของพารามิเตอร์	VARCHAR(30)	P.K.
2	Param_value	ค่าของพารามิเตอร์	VARCHAR(30)	

ตารางที่ 3.8 แสดงตารางข้อมูลพารามิเตอร์ของผู้ใช้ : Username

ลำดับ	ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	คีย์
1	Login_name	ชื่อการล็อกอิน	VARCHAR(15)	P.K.
2	Pwd	Password การเข้าใช้ระบบ	VARCHAR(15)	

ตารางที่ 3.9 แสดงตารางข้อมูลออบเจ็กต์ VRML : Vrml_Object

ลำดับ	ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	คีย์
1	Object_type	ชนิดของออบเจ็กต์ VRML	VARCHAR(30)	P.K.
2	Vrml_obj_file	ชื่อไฟล์เทมเพลิตออบเจ็กต์ VRML ที่ใช้ในการสร้างออบเจ็กต์ชนิดนี้	VARCHAR(30)	

ตารางที่ 3.10 แสดงตารางข้อมูลพารามิเตอร์ของออบเจ็กต์ VRML : Vrml_Param

ลำดับ	ชื่อฟิลด์	คำอธิบาย	ชนิดข้อมูล	คีย์
1	Object_id	ID ของออบเจ็กต์	INT	P.K.
2	Param_name	ชื่อพารามิเตอร์ของออบเจ็กต์	VARCHAR(30)	P.K.
3	Param_value	ค่าพารามิเตอร์ของออบเจ็กต์	VARCHAR(30)	

ตารางที่ 3.11 แสดงตารางข้อมูล IP Address ของเครื่องไคลเอนต์ : ConnectingClient

ลำดับ	ชื่อแอตทริบิวต์	คำอธิบาย	ชนิดข้อมูล	คีย์
1	IP	Internet Protocol Address ของเครื่องไคลเอนต์	VARCHAR(30)	P.K.
2	Uport	Unique port ที่โปรแกรมจะทำการสร้างและกำหนดให้กับไคลเอนต์แต่ละหน้าจอ	VARCHAR(10)	P.K.

ตารางที่ 3.12 แสดงตารางข้อมูลฐานข้อมูลเครือข่าย : NodeTab

ลำดับ	ชื่อแอตทริบิวต์	คำอธิบายข้อมูล	ชนิดข้อมูล	คีย์
1	NodeName	ชื่อโหนด	VARCHAR(50)	P.K.
2	SysDesc	คำอธิบายของระบบ	VARCHAR(50)	
3	SysObjId	หมายเลขวัตถุของระบบ	VARCHAR(50)	
4	SysContact	ชื่อผู้ติดต่อของระบบ	VARCHAR(50)	
5	SysLocation	สถานที่ของระบบ	VARCHAR(50)	
6	SysUpTime	เวลาของระบบ	FLOAT(8)	
7	SysService	ชื่อบริการของระบบ	INT(2)	
8	IfNumber	หมายเลขอินเตอร์เฟซ	INT(2)	

ตารางที่ 3.13 แสดงตารางข้อมูลฐานข้อมูลเครือข่าย : IfTable

ลำดับ	ชื่อแอตทริบิวต์	คำอธิบายข้อมูล	ชนิดข้อมูล	คีย์
1	NodeName	ชื่อโหนด	VARCHAR(50)	P.K.
2	IfIndex	หมายเลขอินเตอร์เฟซ	INT(2)	P.K.
3	IfDesc	ความหมายของอินเตอร์เฟซ	VARCHAR(50)	
4	IfType	ชนิดของอินเตอร์เฟซ	INT(2)	
5	IfMtu	ขนาดที่ใหญ่ที่สุดของ Protocol Data Unit	INT(2)	
6	IfPhyAddress	ที่อยู่ของอินเตอร์เฟซระดับ Protocol Layer	VARCHAR(50)	
7	IfOperStatus	สถานะของอินเตอร์เฟซ	INT(2)	
8	IfAdminStatus	สถานะของอินเตอร์เฟซที่ต้องการ	INT(2)	
9	IfSpeed	Data rate capacity ของอินเตอร์เฟซ	FLOAT(8)	
10	IfLastChange	การเปลี่ยนแปลงครั้งสุดท้ายของอินเตอร์เฟซ	FLOAT(8)	
11	IfSpecific	สื่อที่ใช้ของอินเตอร์เฟซ	VARCHAR(50)	

3.6 อัลกอริทึมการทำงานของระบบ

จากหลักการทำงานของระบบในหัวข้อที่ 3.2.2 ในระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML มีอัลกอริทึมการทำงานของโปรแกรมที่เป็นหลักๆ บางส่วนดังนี้

- ผู้ใช้ทำการล็อกอินเข้าระบบจัดการเครือข่ายผ่านเว็บด้วย VRML

  ผู้ใช้ป้อนรหัสผ่านเข้าสู่ระบบจัดการเครือข่ายผ่านเว็บด้วย VRML

- ผู้ใช้ทำการเลือกหน้าจอการทำงานของพื้นที่ต่างๆ

  ผู้ใช้ทำการเลือกพื้นที่ที่จะเข้าไปสำรวจในระบบจัดการเครือข่าย ด้วยการระบุ URL ผ่านเว็บเบราเซอร์ ผ่านโปรแกรม mainvrml.exe โดยที่ mainvrml.exe จะทำการเรียกโพรซีเดอร์ GenerateContent อันเป็นสคริปต์หลักที่ใช้ในการสร้าง VRML Space หรือโครงของแต่ละพื้นที่ และโพรซีเดอร์ GenerateContent จะทำการเรียกโพรซีเดอร์ GenerateContentDetail อันเป็นสคริปต์ย่อยที่ใช้ในการสร้าง VRML ออบเจ็กต์ต่างๆ ที่เกี่ยวข้องกับ VRML Space นั้นๆ หรือโครงสร้างของแต่ละพื้นที่

- ผู้ใช้แก้ไขเพิ่มเติมออบเจ็กต์ในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML
- ผู้ใช้ทำการ Logout เมื่อต้องการออกจากระบบ แบ่งออกเป็น 2 กรณี
  - กรณี Logout แบบปกติ โดยการกดปุ่ม Logout
  - กรณี Logout แบบไม่ปกติ เป็นการออกจากระบบโดยไม่กดปุ่ม Logout

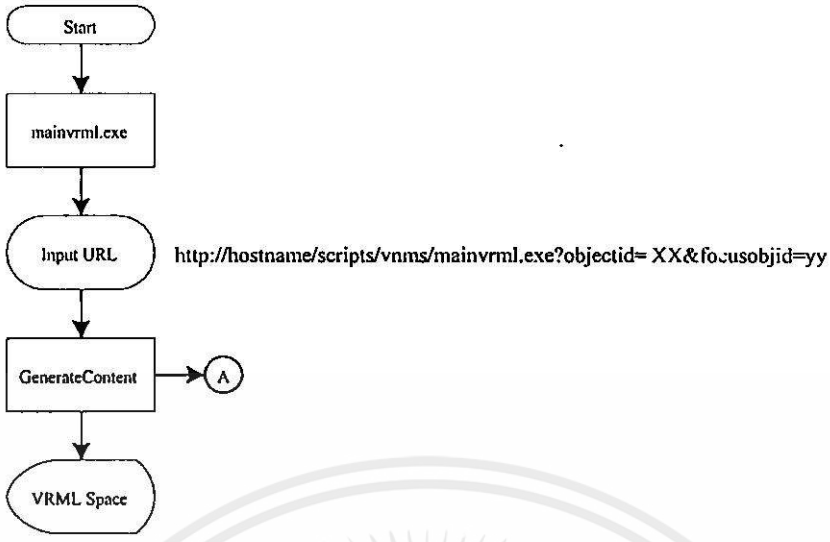
ดังมีรายละเอียดอัลกอริทึมดังนี้

- **Program mainvrml.exe**
  1. Input URL

     http://hostname/scripts/vnms/mainvrml.exe?objectid=xx&focusobjid=yy
  2. Get two input parameter

     FMainObjectId := objectid

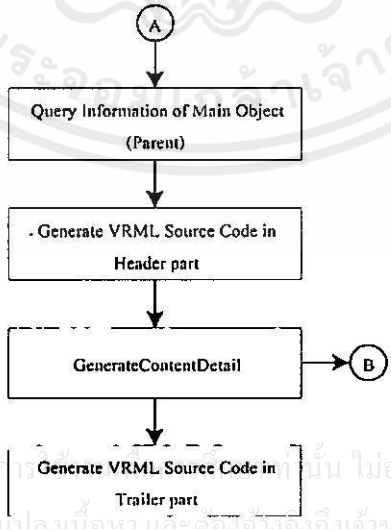
     FFocusedObjectId := focusobjId
  3. Revoke procedure : **GenerateContent**
  4. Response ContentType := model/vrml
  5. Response Content := vrmlspace



รูปที่ 3.23 แสดงอัลกอริทึมของโปรแกรม mainvrml.exe

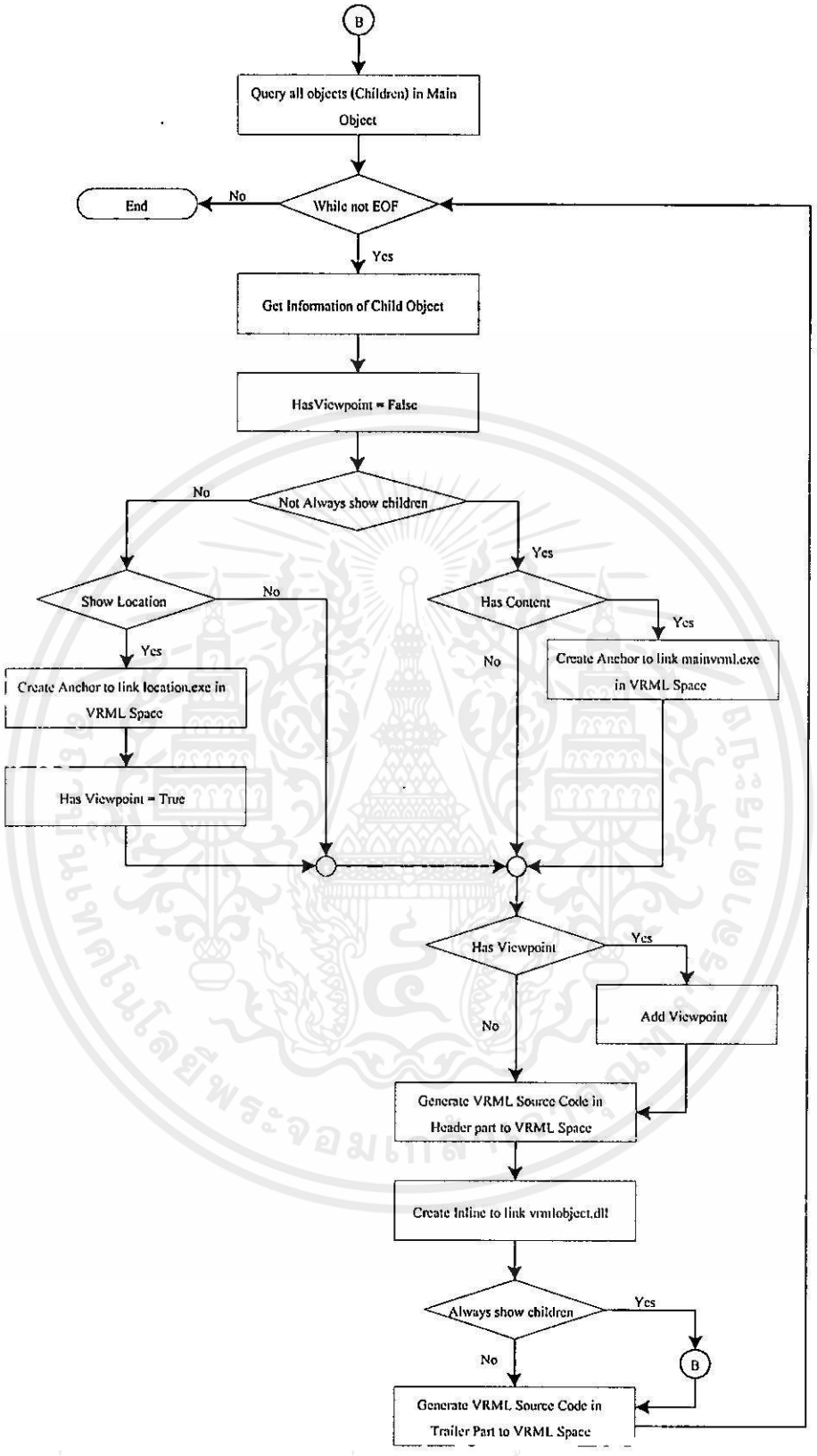
Procedure GenerateContent

1. Query Information of Main Object by FMainObjectId
   1.1 Assign retreived value to FMainLocX , FMainLocY , FMainLocZ ,FMainAxes , FMainAngle
2. Generate VRML source code in Header part
3. Revoke procedure : **GenerateContentDetail(FMainObjectId)**
4. Generate VRML source code in Trailer part



รูปที่ 3.24 แสดงอัลกอริทึมของโพรซีเดอร์ GenerateContent

```
Procedure GenerateContentDetail(FmainObjectId)
1. Query all objects in MainObject by FMainObjectId
      1.1 while not EOF do begin
            get information of object and assign value to FObjectId ,FObjectType , FLocX,
              FLocY , FLocZ , FAxes , FAngle , FAlwayShowChildren
            if FObjectId = FfocusedObjectId   then Focused := 'true'
                                              else Focused := 'false'
            HasViewpoint := False
            if Not FAlwaysShowChildren then
                if HasContent(FObjectId) then Create Anchor to link mainvrml.exe
                  (FObjectId) in vrmlspace
            else if ShowLocation(FObjectType) then
                Create Anchor to link location.exe(FObjectId) in vrmlspace
                HasViewPoint := True
            if HasViewPoint then AddViewPoint(FObjectId,vrmlspace)
            Generate VRML source code in Header part to vrmlspace
            Create Inline to link vrmlobject.dll(FObjectId,Focused)
            if FAlwayShowChildren then GenerateContentDetail(FObjectId)
            Generate VRML source code in Trailer part to vrmlspace
          Next
```



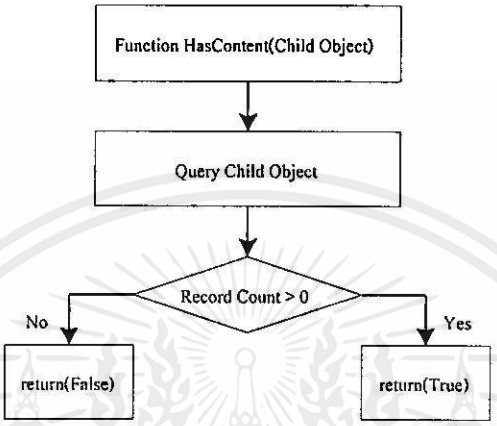
รูปที่ 3.25 แสดงอัลกอริทึมของโพรซีเดอร์ GenerateContentDetail

- **Function HasContent(FobjectId) : boolean**

1. Query ObjectId

if RecordCount > 0 then return(True)

else return(False)

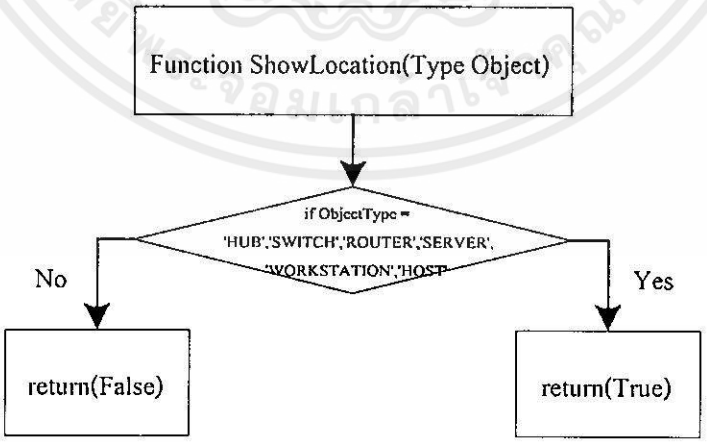


รูปที่ 3.26 แสดงอัลกอริทึมของฟังก์ชัน HasContent

- **Function ShowLocation(FObjectType) : boolean**

if ObjectType = 'HUB' , 'SWITCH' , 'ROUTER' , 'SERVER' , 'WORKSTATION' ,
'HOST' then return(True)

else return(False)



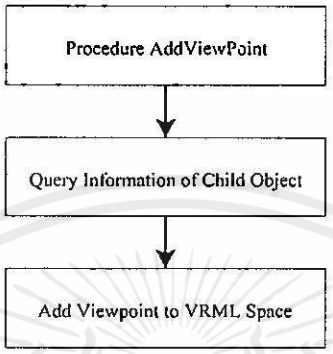
รูปที่ 3.27 แสดงอัลกอริทึมของฟังก์ชัน ShowLocation

- **Procedure AddViewPoint(FObjectId,vrmlspace)**

1. Query Information of Object by FObjectId

    1.1 Assign retreived value to FLocX , FLocY , FLocZ , FObjectName

2. Add Viewpoint by using FLocX , FLocY , FLocZ , FObjectName to vrmlspace



รูปที่ 3.28 แสดงอัลกอริทึมของโพรซีเดอร์ AddViewPoint

- **vrmlobject.dll**

1. Get two input parameter

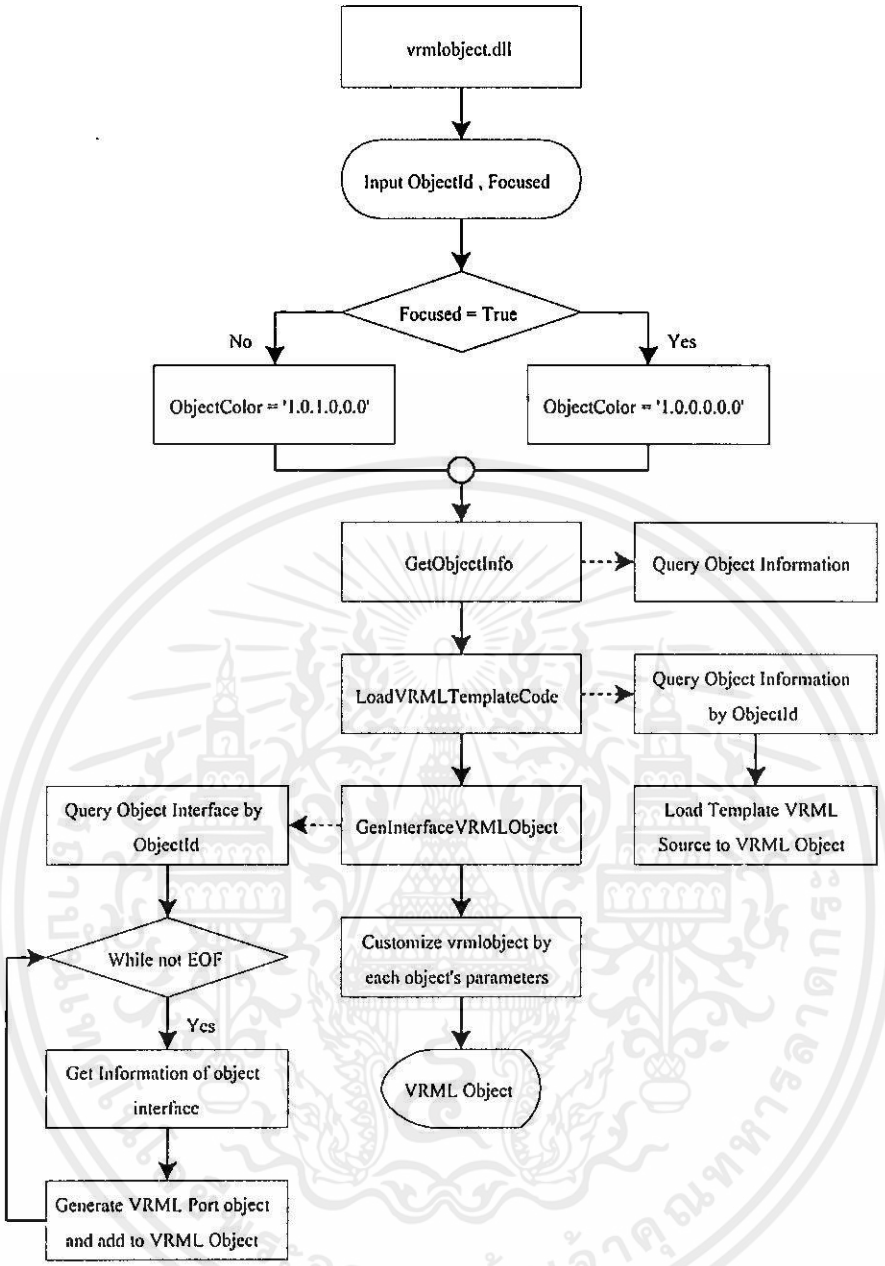
    ObjectId := objectid

    Focusecd := focused

    if Focused = 'true' then FObjectColor := '1.0.0.0.0.0'

    else FObjectColor := '1.0.1.0.0.0'

2. Revoke procedure : **GetObjectInfo**
3. Revoke procedure : **LoadVRMLTemplateCode**
4. Revoke procedure : **GenInterfaceVRMLObject(vrmlobject)**
5. Customize vrmlobject by each object's parameters (see detail in **vrmlObjectHTMLTag** Method)
6. Response ContentType := model/vrml
7. Response Content := vrmlobject



รูปที่ 3.29 แสดงอัลกอริทึมของ vrmlobject.dll

- **Procedure GetObjectInfo**

1. Query Object Information

    1.1 Assign retreived value to FObjectName

- **Procedure LoadVRMLTemplateCode**

1. Query Object Information by ObjectId
    1.1 Assign retreived value to TplFilename
    1.2 Load Template VRML Source (TplFilename) to vrmlobject

- **Procedure GenInterfaceVRMLObject(vrmlobject)**

1. Query object interface by ObjectId
    1.1 while not EOF do begin
        get information of object interface and assign value to FInterfacePicture , FPicSizeX , FPicSizeY , FInterfaceLocX , FInterfaceLocY, FInterfaceLocZ
        Generate VRML Port object and add to vrmlobject
    Next

Procedure vrmlObjectHTMLTag

(Use this procedure(Method) for customize webpage(vrmlpage/vrmlobject) before response)

1. Query object's parameter by ObjectId
    1.1 while not EOF do begin
        get parameter value and replace to placeholder of that parameter in vrmlobject
    Next

3.7 สรุป

จากหลักการทำงานของระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ทำให้สามารถเข้าไปจัดการ, ตรวจสอบและดูแลระบบเครือข่ายคอมพิวเตอร์ผ่านเครือข่ายอินเตอร์เน็ตโดยใช้เว็บเบราเซอร์ในรูปแบบที่เสมือนจริงและอิงกับลักษณะทางกายภาพ แต่เนื่องจากข้อมูลที่จัดเก็บที่นำมาใช้ในการทดลองครั้งนี้ยังมีความซับซ้อนและรายละเอียดไม่มากนัก ซึ่งผู้ใช้สามารถที่จะจัดเก็บข้อมูลดังกล่าวได้มากกว่านี้ โดยสามารถที่จะเพิ่มระดับความสัมพันธ์ของออบเจ็กต์ต่างๆ ได้มากเท่าที่ต้องการ เพื่อให้เห็นภาพที่ละเอียดและมีความเหมือนจริงมากยิ่งขึ้น แต่ทั้งนี้ยิ่งข้อมูลที่จัดเก็บมีความซับซ้อนและมีรายละเอียดมากเท่าไร เวลาที่จะถูกใช้ในการประมวลผลและการแสดงผลของระบบก็ยิ่งจะต้องมากขึ้นเช่นกัน

บทที่ 4

การวิเคราะห์การทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

ในงานวิจัยนี้ได้ทำการพัฒนาเครื่องมือในการจัดการเครือข่ายโดยมีระบบติดต่อผู้ใช้แบบ 3 มิติที่สามารถตรวจดูเครือข่ายและจัดการเครือข่าย เช่น การตรวจดูสถานะของเครือข่าย, การเพิ่มหรือนำอุปกรณ์เครือข่ายออกจากระบบ เป็นต้น โดยการนำเสนอในรูปแบบ 3 มิติ ด้วยการนำภาษา VRML เข้ามาใช้ ซึ่งอาจจะทำให้มีผลกับการใช้งานการจัดการเครือข่ายในแง่ของเวลาที่ใช้ในการประมวลผลและประสิทธิภาพของเครื่องที่ใช้ ดังรายละเอียดที่จะกล่าวในบทนี้

จากบทที่ 3 ได้กล่าวถึงหลักการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML จนสำเร็จตามขอบเขตและวัตถุประสงค์แล้ว ในบทนี้จะได้ทำการวิเคราะห์การทำงานของระบบในแต่ละส่วนว่าประกอบด้วยเวลาที่เกิดจากขั้นตอนการทำงานใดบ้าง เพื่อศึกษาถึงเวลาตอบสนองของการทำงานในแต่ละส่วน

4.1 การวิเคราะห์ขั้นตอนการทำงานหลักของระบบ

จากบทที่ 3 หัวข้อที่ 3.2.2 ได้กล่าวถึงหลักการทำงานของระบบไปแล้ว ในหัวข้อนี้จะได้นำหลักการในแต่ละขั้นตอนการทำงานนำมาวิเคราะห์ว่าแต่ละช่วงเวลาเกิดจากกระบวนการทำงานใดบ้าง โดยจะทำการวิเคราะห์แยกออกเป็น 2 ส่วน คือ

- ขั้นตอนการ View Loading ดังในรายละเอียดในหัวข้อ 4.1.1
- ขั้นตอนการ View Navigate ดังในรายละเอียดในหัวข้อ 4.1.2

4.1.1 การวิเคราะห์ขั้นตอนการ View Loading

คือ ส่วนที่เริ่มตั้งแต่ผู้ใช้งานทำการร้องขอการทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยเว็บเซิร์ฟเวอร์จะคอยฟังการร้องขอที่ส่งมาและทำการประมวลผลสคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซิร์ฟเวอร์แอปพลิเคชั่น (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผลและทำการดึงข้อมูลจากฐานข้อมูล เพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกประมวลผลก็จะส่งผลลัพธ์กลับไปให้เว็บเซิร์ฟเวอร์ ต่อจากนั้นเว็บเซิร์ฟเวอร์จึงส่งผลลัพธ์ดังกล่าวกลับไปให้เว็บเบราเซอร์ที่ส่งการร้องขอมา ดังรูปที่ 3.9

แต่เนื่องด้วยเวลาที่เกิดขึ้นจากขั้นตอนการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ประกอบด้วยเวลาในหลายๆ ช่วง ดังรูปที่ 3.9 เริ่มจากผู้ใช้งานจะทำการร้องขอการทำงานที่ต้องการผ่านเว็บเบราเซอร์ โดยเวลาที่เริ่มตั้งแต่ผู้ใช้ป้อน URL ผ่านเว็บเบรา

เซอร์ จนกระทั่งกดปุ่ม Enter เพื่อส่งให้กับเว็บเซิร์ฟเวอร์นั้น ในที่นี้จะไม่ทำการพิจารณาเนื่องจากเวลาที่ผู้ใช้ป้อนไม่สามารถควบคุมได้ในแง่ของทักษะการพิมพ์ ดังนั้นจะเริ่มพิจารณาเวลาโดยแบ่งเป็นช่วงกว้างๆ ตามจุดที่รับส่งข้อมูลได้ดังนี้ คือ (ตามหมายเลขกำกับรูปที่ 3.9)

- $T_{WB\_to\_WS}$ = ช่วงเวลาที่เว็บเบราเซอร์ติดต่อสื่อสารกับเว็บเซิร์ฟเวอร์ (การทำงานหมายเลข 2-3)
- $T_{WS\_to\_CGI}$ = ช่วงเวลาที่เว็บเซิร์ฟเวอร์ติดต่อสื่อสารกับ CGI (การทำงานหมายเลข 4-6)
- $T_{CGI\_to\_DBMS}$ = ช่วงเวลาที่ CGI ติดต่อสื่อสารกับระบบจัดการฐานข้อมูล (การทำงานหมายเลข 7-11)
- $T_{CGI\_to\_WS}$ = ช่วงเวลาที่ CGI ติดต่อสื่อสารกับเว็บเซิร์ฟเวอร์ (การทำงานหมายเลข 12)
- $T_{WS\_to\_WB}$ = ช่วงเวลาที่เว็บเซิร์ฟเวอร์ติดต่อสื่อสารกับเว็บเบราเซอร์ (การทำงานหมายเลข 13-14)
- $T_{WB\_to\_VRMLBrowser}$ = ช่วงเวลาที่เว็บเบราเซอร์ติดต่อสื่อสารกับผู้ใช้ (การทำงานหมายเลข 15-17)

ดังรายละเอียดต่อไปนี้

1. $T_{WB\_to\_WS}$ เป็นช่วงเวลาที่เว็บเบราเซอร์ติดต่อสื่อสารกับเว็บเซิร์ฟเวอร์ โดยเมื่อเว็บเบราเซอร์รับการป้อนข้อมูลจากผู้ใช้แล้วจะส่งการร้องขอตามที่ผู้ใช้ต้องการพร้อมกับข้อมูลที่จำเป็นสำหรับการสืบค้นข้อมูลตามที่ผู้ใช้ป้อนไปยังเว็บเซิร์ฟเวอร์ ซึ่งประกอบด้วยเวลาในแต่ละช่วง ดังสมการ

$$T_{WB\_to\_WS} = T_{WB\_Proc\_1} + T_{WB\_Transfer\_URL} \quad \text{.....(4.1)}$$

โดยที่ $T_{WB\_Proc\_1}$ = เวลาหลังจากเว็บเบราเซอร์รับการป้อน URL จากผู้ใช้แล้ว เว็บเบราเซอร์จะทำการประมวลผลเพื่อตรวจสอบว่าต้องส่ง URL ไปยังเว็บเซิร์ฟเวอร์ใดและสร้างการติดต่อสื่อสารเพื่อเตรียมส่งพารามิเตอร์ให้กับเว็บเซิร์ฟเวอร์

$T_{WB\_Transfer\_URL}$ = เวลาที่เริ่มตั้งแต่เว็บเบราเซอร์ส่ง URL ให้กับเว็บเซิร์ฟเวอร์ผ่านโปรโตคอล HTTP จนกระทั่งเว็บเซิร์ฟเวอร์ได้รับข้อมูลครบถ้วน โดยเวลาของ $T_{WB\_Transfer\_URL}$ คำนวณได้จากสมการ

$$T_{WB\_Transfer\_URL} = \frac{S_{URL}}{R_{HTTPRate}} \quad \text{.....(4.2)}$$

โดยที่ S_{URL} = ขนาดของ URL ที่เว็บเบราเซอร์จะทำการส่งให้กับเว็บเซิร์ฟเวอร์ผ่านโปรโตคอล HTTP จนกระทั่งเว็บเซิร์ฟเวอร์ได้รับข้อมูลครบถ้วน

$R_{HTTPRate}$ = ขนาดความเร็วของ HTTP Transmission Rate ณ เวลาที่เว็บเบราเซอร์ทำการส่ง URL ให้กับเว็บเซิร์ฟเวอร์

แทนสมการ (4.2) ใน (4.1) จะได้เวลาที่เกิดขึ้นทั้งหมดในการติดต่อสื่อสารกันระหว่างเว็บเบราเซอร์และเว็บเซิร์ฟเวอร์

$$T_{WB\_to\_WS} = T_{WB\_Proc\_1} + \frac{S_{URL}}{R_{HTTPRate}} \quad \text{.....(4.3)}$$

จากสมการที่ 4.3 เวลาของ $T_{WB\_to\_WS}$ ขึ้นอยู่กับประสิทธิภาพของเครื่องไคลเอนต์และเว็บเบราเซอร์ เนื่องจากเว็บเบราเซอร์ต้องใช้ประสิทธิภาพของเครื่องไคลเอนต์ในการประมวลผล รวมถึงขนาดของ URL ที่ส่งข้ามเครือข่ายกันระหว่างเว็บเบราเซอร์และเว็บเซิร์ฟเวอร์ และสภาพของเครือข่ายที่ทำการรับส่งขณะนั้นด้วย เช่น ความเร็วของเครือข่ายและความหนาแน่นของเครือข่ายที่ทำการรับส่ง ณ เวลานั้น

2. $T_{WS\_to\_CGI}$ เป็นช่วงเวลาที่เว็บเซิร์ฟเวอร์ติดต่อสื่อสารกับ CGI โดยเมื่อเว็บเซิร์ฟเวอร์ได้รับการร้องขอจากไคลเอนต์แล้วเว็บเซิร์ฟเวอร์จะกระตุ้นการทำงานของเกต์เวย์โปรแกรมและส่งผ่านข้อมูลที่จำเป็นสำหรับการสืบค้นข้อมูลตามที่ผู้ใช้ระบุส่งไปยังเกต์เวย์โปรแกรม ซึ่งประกอบด้วยเวลาในแต่ละช่วง ดังสมการ

$$T_{WS\_to\_CGI} = T_{WS\_Proc\_1} + T_{WS\_Revoke\_CGI} + T_{WS\_Transfer\_Para} \quad \text{.....(4.4)}$$

โดยที่ $T_{WS\_Proc\_1}$ = เวลาที่เว็บเซิร์ฟเวอร์ได้รับข้อมูลการร้องขอจากเว็บเบราเซอร์แล้วจึงทำการประมวลผลตามที่เว็บเบราเซอร์ร้องขอมา

$T_{WS\_Revoke\_CGI}$ = เวลาที่เว็บเซิร์ฟเวอร์เรียก CGI Script ขึ้นมาทำงานตามที่ระบุใน URL

เนื่องจากเวลาของ $T_{WS\_Proc\_1}$ และ $T_{WS\_Revoke\_CGI}$ ขึ้นอยู่กับประสิทธิภาพของเว็บเซิร์ฟเวอร์ (E_{WS}) ได้แก่ Response Latency, Connection-handling Capacity, Server Throughput, Number of Connections per Time Interval และ Number of Fixed-size HTTP

Requests Processed เป็นต้น โดยการทดลองในบทที่ 5 จะได้ทำการวัดประสิทธิภาพโดยรวมในรูปของฟังก์ชันประสิทธิภาพของเว็บเซิร์ฟเวอร์ $f_{WS}\left(E_{WS}\right)$

$$T_{WS\_Proc\_I} + T_{WS\_Revoke\_CGI} \approx f_{WS}\left(E_{WS}\right) \quad \text{.....(4.5)}$$

โดยที่ E_{WS} = ประสิทธิภาพของเว็บเซิร์ฟเวอร์และการทำงานร่วมกันระหว่างเว็บเซิร์ฟเวอร์และ CGI ในการประมวลผล ด้วยการวัดเวลาตอบสนองที่เว็บเซิร์ฟเวอร์ส่งความต้องการ (จำนวนการร้องขอ) ให้กับ CGI ในการประมวลผลจนกระทั่ง CGI ส่งผลลัพธ์กลับคืนให้กับเว็บเซิร์ฟเวอร์ครบถ้วน มีหน่วยเป็นจำนวนจำนวนครั้งต่อวินาที

$f_{WS}\left(E_{WS}\right)$ = เป็นเวลาที่ใช้ในการประมวลผลของเว็บเซิร์ฟเวอร์ต่อหนึ่งการร้องขอ โดยอยู่ในรูปของฟังก์ชันที่สัมพันธ์กับประสิทธิภาพของเว็บเซิร์ฟเวอร์ มีหน่วยเป็นวินาที

$T_{WS\_Transfer\_Para}$ = เวลาที่เว็บเซิร์ฟเวอร์ทำการส่งพารามิเตอร์ที่ได้รับจากเว็บเบราเซอร์ให้กับ CGI Script เพื่อใช้ในการประมวลผล โดยเวลาของ $T_{WS\_Transfer\_Para}$ คำนวณได้จากสมการ

$$T_{WS\_Transfer\_Para} = \frac{S_{Para}}{R_{WS\_CGI\_Rate}} \quad \text{.....(4.6)}$$

โดยที่ S_{Para} = ขนาดของพารามิเตอร์ที่เว็บเซิร์ฟเวอร์ทำการส่งให้กับ CGI

$R_{WS\_CGI\_Para}$ = ขนาดความเร็ว ณ เวลาที่เว็บเซิร์ฟเวอร์ทำการส่งพารามิเตอร์ให้กับ CGI

แทนสมการ (4.5) และ (4.6) ใน (4.4) จะได้เวลาที่เกิดขึ้นทั้งหมดในการติดต่อสื่อสารกันระหว่างเว็บเซิร์ฟเวอร์และ CGI

$$T_{WS\_to\_CGI} \approx f_{WS}\left(E_{WS}\right) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} \quad \text{.....(4.7)}$$

3. $T_{CGI\_to\_DBMS}$ เป็นช่วงเวลาที่ CGI ติดต่อสื่อสารกับระบบจัดการฐานข้อมูล โดยเมื่อระบบเว็บเซิร์ฟเวอร์ฐานข้อมูลได้รับข้อมูลที่จำเป็นสำหรับการประมวลผลจากเกต์เวย์โปรแกรมแล้ว จะทำการประมวลผลและเมื่อได้ข้อมูลตามที่ต้องการแล้วจะส่งข้อมูลที่ได้ออกไปยังเกต์เวย์โปรแกรม ซึ่งการติดต่อกับเว็บเซิร์ฟเวอร์ฐาน

ข้อมูลอาจฝังภาษาที่เรียกใช้ข้อมูลที่สามารถระบุความต้องการได้ เช่น ภาษา SQL (Structured Query Language) ไว้ใน CGI Script โดยการทำงานจะเริ่มตั้งแต่ผู้ใช้ส่งข้อความร้องขอข้อมูลจากเครื่องลูกหรือเครื่องที่เป็นไคลเอนต์ไปยังเครื่องเว็บเซิร์ฟเวอร์ ต่อจากนั้นโปรแกรม CGI บนเว็บเซิร์ฟเวอร์ก็จะแยกส่วนที่เป็นภาษา SQL ออกมา และส่งส่วนที่เป็นภาษา SQL ไปให้กับระบบการจัดการฐานข้อมูล เมื่อเกต์เวย์โปรแกรมได้รับผลที่ได้จากเว็บเซิร์ฟเวอร์ฐานข้อมูลแล้ว เกต์เวย์โปรแกรมจะส่งผ่านข้อมูลเหล่านั้นไปยังเว็บเซิร์ฟเวอร์ ซึ่งประกอบด้วยผลรวมของเวลาในการประมวลผลของแต่ละช่วง ดังสมการ

$$T_{CGI\_to\_DBMS} = \sum_{k=1}^{N_{Obj}} \left(T_{CGI\_Proc,k} + T_{CGI\_Transfer\_SQL,k} + T_{DBMS\_Proc,k} \right)$$

$$T_{CGI\_to\_DBMS} = \sum_{k=1}^{N_{Obj}} \left(T_{CGI\_Proc,k} \right) + \sum_{k=1}^{N_{Obj}} \left(T_{CGI\_Transfer\_SQL,k} \right) + \sum_{k=1}^{N_{Obj}} \left(T_{DBMS\_Proc,k} \right) \quad \text{.....(4.8)}$$

โดยที่ N_{Obj} = จำนวนออบเจ็กต์ที่ CGI Script ติดต่อกับระบบจัดการฐานข้อมูลต่อหนึ่ง view

เนื่องจากเวลาของ $\sum_{k=1}^{N_{Obj}} \left(T_{CGI\_Proc\_k} \right)$ ขึ้นอยู่กับประสิทธิภาพของเว็บเซิร์ฟเวอร์ที่ใช้รัน CGI และประสิทธิภาพการทำงานของ CGI เอง $\left(E_{WS\_CGI} \right)$ รวมถึงจำนวนออบเจ็กต์ต่อหนึ่ง view $\left(N_{Obj} \right)$

$$\sum_{k=1}^{N_{Obj}} \left(T_{CGI\_Proc,k} \right) = f_{WS\_CGI} \left(N_{Obj}, E_{WS\_CGI} \right) \quad \text{..... (4.9)}$$

โดยที่ E_{WS_CGI} = ประสิทธิภาพของเว็บเซิร์ฟเวอร์และการทำงานร่วมกันระหว่างเว็บเซิร์ฟเวอร์และ CGI ในการประมวลผล ด้วยการวัดเวลาตอบสนองที่เว็บเซิร์ฟเวอร์ส่งความต้องการ (จำนวนออบเจ็กต์) ให้กับ CGI ในการประมวลผลจนกระทั่ง CGI ส่งผลลัพธ์กลับคืนให้กับเว็บเซิร์ฟเวอร์ครบถ้วน มีหน่วยเป็นจำนวนออบเจ็กต์ต่อวินาที

$f_{WS\_CGI} \left(N_{Obj}, E_{WS\_CGI} \right)$ = เป็นเวลาที่ใช้ในการประมวลผลของ CGI ต่อหนึ่ง View โดยอยู่ในรูปของฟังก์ชันที่สัมพันธ์กับจำนวนออบเจ็กต์ในหนึ่ง view และประสิทธิภาพของเว็บเซิร์ฟเวอร์และการทำงานร่วมกันระหว่างเว็บเซิร์ฟเวอร์และ CGI ด้วยการวัดเวลาในการประมวลผลของออบเจ็กต์ลำดับที่ k ที่ CGI Script ทำการประมวลผลตามพารามิเตอร์ใน URL ที่

ผู้ใช้ร้องขอมา และเวลาในการประมวลผลของออบเจ็กต์ลำดับที่ k หลังจากที่ CGI Script ได้รับข้อมูลจากระบบจัดการฐานข้อมูลแล้วทำการรวบรวมข้อมูลที่ได้ใน VRML Script เพื่อเป็นข้อมูลในการส่งต่อให้กับเว็บเซิร์ฟเวอร์ รวมถึงเวลาในการประมวลผลของออบเจ็กต์ลำดับที่ k ที่ CGI Script รวบรวม VRML Script เรียบร้อยแล้ว ทำการส่ง VRML Script ให้กับเว็บเซิร์ฟเวอร์ มีหน่วยเป็นวินาที

$\sum_{k=1}^{N_{obj}}\left(T_{CGI\_Transfer\_SQL,k}\right)$ = เวลาในการประมวลผลของออบเจ็กต์ในหนึ่ง view ที่ CGI Script ส่งพารามิเตอร์ต่างๆ ที่ต้องใช้สำหรับการประมวลผลตามคำร้องขอที่ระบุมาใน URL ให้กับระบบจัดการฐานข้อมูล เพื่อนำไปประมวลผลและทำการดึงข้อมูลจากฐานข้อมูล เพื่อเป็นข้อมูลร่วมในการประมวลผลด้วยในรูปของภาษา SQL ให้กับระบบจัดการฐานข้อมูล และเวลาในการประมวลผลของออบเจ็กต์ลำดับที่ k หลังจากระบบจัดการฐานข้อมูลประมวลผลเรียบร้อยแล้วทำการส่งผลที่ได้กลับให้กับ CGI Script

เนื่องจากเวลาของ $\sum_{k=1}^{N_{obj}}\left(T_{CGI\_Transfer\_SQL,k}\right)$ ขึ้นอยู่กับอัตราการรับ-ส่งข้อมูลระหว่าง CGI กับระบบจัดการฐานข้อมูล $\left(R_{CGI\_DBMS\_Rate}\right)$ และขนาดของ SQL $\left(S_{SQL}\right)$

$$\sum_{k=1}^{N_{obj}}\left(T_{CGI\_Transfer\_SQL,k}\right)=f_{CGI\_DBMS}\left(S_{SQL},R_{CGI\_DBMS\_Rate}\right) \quad \text{.....(4.10)}$$

โดยที่ $f_{CGI\_DBMS}\left(S_{SQL},R_{CGI\_DBMS\_Rate}\right)$ = เวลาในการประมวลผลของออบเจ็กต์ลำดับที่ k ที่ระบบจัดการฐานข้อมูลใช้ในการประมวลผลตามคำสั่ง SQL ที่ได้รับจาก CGI Script มีหน่วยเป็นวินาที

เนื่องจากเวลาของ $\sum_{k=1}^{N_{obj}}\left(T_{DBMS\_Proc,k}\right)$ ขึ้นอยู่กับประสิทธิภาพของระบบจัดการฐานข้อมูล $\left(E_{DBMS}\right)$ อันได้แก่ จำนวนข้อมูล, จำนวนเรคคอร์ด, ความยาวเรคคอร์ด, อินเด็กซ์และสถาปัตยกรรมของฐานข้อมูล [10] รวมถึงจำนวนออบเจ็กต์ต่อหนึ่ง view $\left(N_{Obj}\right)$ โดยการทดลองในบทที่ 5 จะได้ทำการประสิทธิภาพโดยรวมในรูปของฟังก์ชันประสิทธิภาพของ $f_{DBMS}\left(N_{Obj},E_{DBMS}\right)$

$$\sum_{k=1}^{N_{obj}}\left(T_{DBMS\_Proc,k}\right)=f_{DBMS}\left(N_{Obj},E_{DBMS}\right) \quad \text{..... (4.11)}$$

โดยที่ E_{DBMS} = ประสิทธิภาพของระบบจัดการฐานข้อมูล ด้วยการวัดเวลาตอบสนองที่ CGI ส่งคำร้องขอ (SQL Statement) ให้กับระบบจัดการฐานข้อมูลในการประมวลผลจนกระทั่งระบบจัดการฐานข้อมูลส่งผลลัพธ์กลับคืนให้กับ CGI ครบถ้วน มีหน่วยเป็นจำนวนคำสั่งต่อวินาที

$f_{DBMS}\left(N_{Obj}, E_{DBMS}\right)$ = เป็นเวลาที่ใช้ในการประมวลผลของระบบจัดการฐานข้อมูลหนึ่งการร้องขอ โดยอยู่ในรูปของฟังก์ชันที่สัมพันธ์กับจำนวนออบเจ็กต์ในหนึ่ง view และประสิทธิภาพของระบบจัดการฐานข้อมูล ด้วยการวัดเวลาในการประมวลผลของระบบจัดการฐานข้อมูลต่อหนึ่งออบเจ็กต์ มีหน่วยเป็นวินาที

แทนสมการ (4.9), (4.10) และ (4.11) ใน (4.8) จะได้เวลาที่เกิดขึ้นทั้งหมดในการติดต่อสื่อสารกันระหว่าง CGI และระบบจัดการฐานข้อมูล

$$T_{CGI\_to\_DBMS} \approx f_{WS\_CGI}\left(N_{Obj}, E_{WS\_CGI}\right) + f_{CGI\_DBMS}\left(S_{SQL}, R_{CGI\_DBMS\_Rate}\right) + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) \quad (4.12)$$

จากสมการ (4.12) เนื่องจากเวลาโดยประมาณที่ $T_{CGI\_to\_DBMS}$ ใช้ในการประมวลผลจะขึ้นอยู่กับฟังก์ชันของเวลาที่ใช้ในการประมวลผลตามจำนวนออบเจ็กต์ และประสิทธิภาพของฐานข้อมูล ดังเขียนเป็นสมการได้ดังนี้

$$T_{CGI\_to\_DBMS} \approx f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) \quad (4.13)$$

4. $T_{CGI\_to\_WS}$ เป็นช่วงเวลาที่ CGI ติดต่อสื่อสารกับเว็บเซิร์ฟเวอร์ โดยเมื่อ CGI ทำการประมวลผลตามคำร้องขอที่ส่งมาจากเว็บเซิร์ฟเวอร์และทำการรวบรวม VRML Script เรียบร้อยแล้ว ถัดมา CGI จะต้องส่งผลลัพธ์ VRML Script ที่ได้ให้กับเว็บเซิร์ฟเวอร์ ดังสมการ

$$T_{CGI\_to\_WS} = T_{CGI\_Transfer\_VRMLScript} \quad (4.14)$$

โดยที่ $T_{CGI\_Transfer\_VRMLScript}$ = เวลาที่ CGI ใช้ติดต่อและส่ง VRML Script ให้กับเว็บเซิร์ฟเวอร์ โดยเวลาของ $T_{CGI\_Transfer\_VRMLScript}$ คำนวณได้จากสมการ

$$T_{CGI\_Transfer\_VRMLScript} = \frac{S_{VRMLScript}}{R_{CGI\_WS\_Rate}} \quad \text{.....(4.15)}$$

โดยที่ $S_{VRMLScript}$ = ขนาดของ VRML ที่ CGI ทำการส่งให้กับเว็บเซิร์ฟเวอร์

$R_{CGI\_WS\_Rate}$ = ขนาดความเร็ว ณ เวลาที่ CGI ทำการส่ง VRML Script ให้กับเว็บเซิร์ฟเวอร์

แทนสมการ (4.15) ใน (4.14) จะได้เวลาที่เกิดขึ้นทั้งหมดในการติดต่อสื่อสารกันระหว่าง CGI และเว็บเซิร์ฟเวอร์

$$T_{CGI\_to\_WS} = \frac{S_{VRMLScript}}{R_{CGI\_WS\_Rate}} \quad \text{.....(4.16)}$$

ซึ่งเวลาของ $T_{CGI\_to\_WS}$ ขึ้นอยู่กับประสิทธิภาพของเครื่องเว็บเซิร์ฟเวอร์ในการประมวลผลและการทำงานร่วมกันระหว่างเว็บเซิร์ฟเวอร์และ CGI ด้วย รวมถึงขนาดของ VRML Script ที่ส่งให้กับเว็บเซิร์ฟเวอร์ ซึ่ง $S_{VRMLScript}$ มีขนาดโดยประมาณเท่ากับ

$$S_{VRMLScript} = S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right) \quad \text{.....(4.17)}$$

โดยที่ S_{Min} = ขนาดของ VRML Script ที่เล็กที่สุด นั่นคือจำนวนออบเจ็กต์เป็นศูนย์

S_{AVG_Obj} = ขนาดของ VRML Script โดยเฉลี่ยต่อหนึ่งออบเจ็กต์ ด้วยชนิดออบเจ็กต์แบบต่างๆ และคุณสมบัติเพิ่มเติมให้กับออบเจ็กต์ VRML

N_{AVG_Obj} = จำนวนออบเจ็กต์โดยเฉลี่ยที่ CGI ติดต่อกับเว็บเซิร์ฟเวอร์ต่อหนึ่ง view

แทนสมการ (4.17) ใน (4.16)

$$T_{CGI\_to\_WS} = \frac{S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)}{R_{CGI\_WS\_Rate}} \quad \text{.....(4.18)}$$

5. $T_{WS\_to\_WB}$ เป็นช่วงเวลาที่เว็บเซิร์ฟเวอร์ติดต่อสื่อสารกับเว็บเบราเซอร์ โดยเมื่อเว็บเซิร์ฟเวอร์ได้รับผลลัพธ์ที่ส่งมาจากเกต์เวย์โปรแกรมแล้ว เว็บเซิร์ฟเวอร์ก็จะส่งผ่านข้อมูลเหล่านั้นไปยังเว็บเบราเซอร์ ซึ่งประกอบด้วยเวลาในแต่ละช่วง ดังสมการ

$$T_{WS\_to\_WB} = T_{WB\_Proc\_2} + T_{WS\_Transfer\_VRMLScript} \quad \text{.....(4.19)}$$

โดยที่ $T_{WS\_Proc\_2}$ = เวลาหลังจากเว็บเซิร์ฟเวอร์ได้รับ VRML Script แล้วทำการเปิดการติดต่อกลับไปหาเว็บเบราเซอร์ที่ร้องขอ URL มา

$T_{WS\_Transfer\_VRMLScript}$ = เวลาที่เว็บเซิร์ฟเวอร์ใช้ในการส่ง VRML Script ให้กับเว็บเบราเซอร์ โดยเวลาของ $T_{WS\_Transfer\_VRMLScript}$ คำนวณได้จากสมการ

$$T_{WS\_Transfer\_VRMLScript} = \frac{S_{VRMLScript}}{R_{HTTPRate}} \quad \text{..... (4.20)}$$

โดยที่ $S_{VRMLScript}$ = ขนาดของ VRML Script ที่เว็บเซิร์ฟเวอร์ทำการส่งให้กับเว็บเบราเซอร์ผ่านโปรโตคอล HTTP จนกระทั่งเว็บเบราเซอร์ได้รับข้อมูลครบถ้วน โดยขนาดของ $S_{VRMLScript}$ ขึ้นอยู่กับจำนวนออบเจ็กต์, ชนิดของออบเจ็กต์และคุณสมบัติต่างๆ ที่ใส่เพิ่มเติมให้กับออบเจ็กต์ ซึ่ง $S_{VRMLScript}$ เท่ากับสมการ (4.17) ดังนั้นแทนสมการ (4.17) และ (4.20) ใน (4.19) จะได้

$$T_{WS\_to\_WB} = T_{WB\_Proc\_2} + \frac{S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)}{R_{HTTPRate}} \quad \text{..... (4.21)}$$

จากสมการที่ 4.21 เวลาของ $T_{WS\_to\_WB}$ ขึ้นอยู่กับประสิทธิภาพของเครื่องเว็บเซิร์ฟเวอร์และเครื่องไคลเอนต์ในการประมวลผล รวมถึงขนาดของ VRML Script ที่ส่งข้ามเครือข่ายกันระหว่างเว็บเซิร์ฟเวอร์และเว็บเบราเซอร์ และสภาพของเครือข่ายที่ทำการรับส่งขณะนั้นด้วย เช่น ความเร็วของเครือข่ายและความหนาแน่นของเครือข่ายที่ทำการรับส่ง ณ เวลานั้นด้วย

6. $T_{WB\_to\_VRMLBrowser}$ เป็นช่วงเวลาที่เว็บเบราเซอร์ติดต่อสื่อสารกับปลั๊กอิน VRML Browser โดยเมื่อเว็บเบราเซอร์ได้รับ VRML Script จากเว็บเซิร์ฟเวอร์แล้วทำการ

ตรวจสอบชนิดของข้อมูลที่ได้รับเพื่อนำเสนอให้ถูกต้องตามรูปแบบที่ได้รับมา ถ้าเป็น VRML เว็บเบราเซอร์จะเรียกปลั๊กอิน VRML Browser ขึ้นมาประมวลผล โดยหลังจากเรียก VRML Browser ขึ้นมาทำงานแล้ว เว็บเบราเซอร์จะทำการส่ง VRML Script ให้กับ VRML Browser เพื่อทำการประมวลผลต่อไป ซึ่งประกอบด้วยเวลาในแต่ละช่วง ดังสมการ

$$T_{WB\_to\_VRMLBrowser} = T_{WB\_Proc\_2} + T_{WB\_Revoke\_VRMLBrowser} + T_{IVB\_Transfer\_VRMLScript} \quad \text{....(4.22)}$$

โดยที่ $T_{WB\_Proc\_2}$ = เวลาที่เว็บเบราเซอร์ทำการตรวจสอบชนิดของข้อมูล (MIME type) ที่ได้รับเพื่อนำเสนอให้ถูกต้องตามรูปแบบที่ได้รับมา ซึ่งถ้าเป็นข้อมูลที่เว็บเบราเซอร์ไม่สามารถนำเสนอได้ก็อาจจะต้องมีการเรียกปลั๊กอินขึ้นมานำเสนอแทน

$T_{WB\_Revoke\_VRMLBrowser}$ = เวลาที่เว็บเบราเซอร์เรียก VRML Browser ขึ้นมาประมวลผล ถ้าชนิดของข้อมูลเป็น VRML นั่นคือ เมื่อเว็บเบราเซอร์ตรวจสอบพบว่าชนิดของข้อมูลที่เว็บเซิร์ฟเวอร์ส่งมาให้นั้นเป็น VRML เว็บเบราเซอร์จะต้องมีการเรียก VRML Browser เพื่อใช้เป็นปลั๊กอินในการดูกราฟฟิก 3 มิติที่เป็นภาษา VRML เพื่อให้ VRML Browser เป็นตัวประมวลผลและนำเสนอในรูปแบบกราฟฟิก 3 มิติ

$T_{IVB\_Transfer\_VRMLScript}$ = เวลาหลังจากเรียก VRML Browser ขึ้นมาทำงานแล้ว เว็บเบราเซอร์จะทำการส่ง VRML Script ให้กับ VRML Browser เพื่อทำการประมวลผลออกมาในรูปแบบกราฟฟิก 3 มิติ

ซึ่งเวลาของ $T_{WB\_to\_VRMLBrowser}$ ขึ้นอยู่กับประสิทธิภาพของเครื่องไคลเอนต์ $\left(E_{WB}\right)$ ในการประมวลผลและการทำงานร่วมกันระหว่างเว็บเบราเซอร์และ VRML Browser ด้วย รวมถึงจำนวนออบเจ็กต์โดยเฉลี่ยต่อหนึ่ง view $\left(N_{AVG\_Obj}\right)$ โดยการทดลองในบทที่ 5 จะได้ทำการวัดประสิทธิภาพโดยรวมในรูปของฟังก์ชันประสิทธิภาพของ $f_{WB}\left(N_{AVG\_Obj}, E_{WB}\right)$ ด้วยการวัดจำนวนออบเจ็กต์ต่อการประมวลผลของเว็บเบราเซอร์หนึ่งครั้ง นั่นคือ

$$T_{WB\_to\_VRMLBrowser} \approx f_{WB}\left(N_{AVG\_Obj}, E_{WB}\right) \quad \text{.....(4.23)}$$

จากบทวิเคราะห์ทั้งหมดที่กล่าวมา จะได้เวลาทั้งหมดของขั้นตอนการ View Loading $\left(T_{Load}\right)$ โดยเริ่มตั้งแต่ผู้ใช้งานทำการร้องขอการทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยเว็บเซิร์ฟเวอร์จะคอยฟังการร้องขอที่ส่งมาและทำการประมวลผลสคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซิร์ฟเวอร์แอปพลิเคชั่น (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย

เพื่อนำไปประมวลผลและอาจจะมีการดึงข้อมูลจากฐานข้อมูล เพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกประมวลผลก็จะส่งผลลัพธ์กลับไปให้เว็บเซิร์ฟเวอร์ ต่อจากนั้นเว็บเซิร์ฟเวอร์จึงส่งผลลัพธ์ดังกล่าวกลับไปให้เว็บเบราเซอร์ที่ส่งการร้องขอมาและถ้าข้อมูลที่ได้รับกลับมาเป็น VRML เว็บเบราเซอร์ก็จะมีการเรียกปลั๊กอิน VRML Browser ขึ้นมาประมวลผลไฟล์ VRML ให้ออกมาเป็นกราฟฟิก 3 มิติ นั่นคือ

$$T_{Load} = T_{WB\_to\_WS} + T_{WS\_to\_CGI} + T_{CGI\_to\_DBMS} + T_{CGI\_to\_WS} + T_{WS\_to\_WB} + T_{WB\_to\_VRMLBrowser} \quad (4.24)$$

โดยที่

$$T_{WB\_to\_WS} = T_{WB\_Proc\_1} + \frac{S_{URL}}{R_{HTTPRate}} \quad (4.3)$$

$$T_{WS\_to\_CGI} \approx f_{WS}(E_{WS}) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} \quad (4.7)$$

$$T_{CGI\_to\_DBMS} \approx f_{DBMS}(N_{Obj}, E_{DBMS}) \quad (4.13)$$

$$T_{CGI\_to\_WS} = \frac{S_{Min} + (S_{AVG\_Obj} \times N_{AVG\_Obj})}{R_{CGI\_WS\_Rate}} \quad (4.18)$$

$$T_{WS\_to\_WB} = T_{WB\_Proc\_2} + \frac{S_{Min} + (S_{AVG\_Obj} \times N_{AVG\_Obj})}{R_{HTTPRate}} \quad (4.21)$$

$$T_{WB\_to\_VRMLBrowser} \approx f_{WB}(N_{AVG\_Obj}, E_{WB}) \quad (4.23)$$

จะได้

$$T_{Load} \approx T_{WB\_Proc\_1} + \frac{S_{URL}}{R_{HTTPRate}} + f_{WS}\left(E_{WS}\right) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) + \frac{S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)}{R_{CGI\_WS\_Rate}} + T_{WB\_Proc\_2} + \frac{S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)}{R_{HTTPRate}} + f_{WB}\left(N_{AVG\_Obj}, E_{WB}\right)$$

ให้ $T_{WB\_Proc\_1} + T_{WB\_Proc\_2} = T_{WB\_Proc}$

$$T_{Load} \approx T_{WB\_Proc} + \frac{S_{URL}}{R_{HTTPRate}} + f_{WS}\left(E_{WS}\right) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) + f_{WB}\left(N_{AVG\_Obj}, E_{WB}\right) + \left[S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)\right] \times \left[\frac{1}{R_{CGI\_WS\_Rate}} + \frac{1}{R_{HTTPRate}}\right]$$

ให้ $T_{WB\_Proc} + \frac{S_{URL}}{R_{HTTPRate}} + f_{WS}\left(E_{WS}\right) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} + f_{WB}\left(N_{AVG\_Obj}, E_{WB}\right) = T_{Min}$

เนื่องจากเป็นคุณลักษณะและประสิทธิภาพของระบบที่ปัจจุบันมีความเร็วในการประมวลผลสูง

ดังนั้นสามารถสรุปสมการของ T_{Load} ได้ดังนี้

$$T_{Load} \approx T_{Min} + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) + \left[S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)\right] \times \left[\frac{1}{R_{CGI\_WS\_Rate}} + \frac{1}{R_{HTTPRate}}\right] \quad \text{.....(4.25)}$$

4.1.2 การวิเคราะห์ขั้นตอนการ View Navigate

คือ ส่วนที่เริ่มตั้งแต่ผู้ใช้ได้รับเว็บเพจครบถ้วนแล้ว ผู้ใช้อาจต้องการตรวจสอบระบบหรือสำรวจพื้นที่รอบๆ บริเวณการจัดการเครือข่ายในโลก 3 มิติ โดยผู้ใช้เพียงแค่ระบุความต้องการผ่าน VRML Browser เพื่อส่งคำสั่งความต้องการให้กับปลั๊กอิน VRML Browser ประมวลผลออกมาในรูปแบบกราฟฟิก 3 มิติ ดังรูปที่ 3.17

$$T_{Navigate} = T_{VRMLBrowser\_Proc} + T_{VRMLBrowser\_Display\_3D} \quad \text{.....(4.26)}$$

โดยที่ $T_{VRMLBrowser\_Proc}$ = เวลาที่ VRML Browser ใช้ในการประมวลผล VRML Script

$T_{VRMLBrowser\_Display\_3D}$ = เวลาที่ VRML Browser ใช้ในการนำเสนอเป็นกราฟฟิก 3 มิติ

ซึ่งเวลาของ $T_{Navigate}$ ขึ้นอยู่กับประสิทธิภาพของปลั๊กอิน VRML Browser ในการประมวลผลและขนาดของ VRML Script ที่ VRML Browser ต้องใช้ในการประมวลผลต่อหนึ่ง view ดังสมการ

$$T_{Navigate} \approx f_{VRMLBrowser}\left(\left(C_{Nav\_AVG\_Obj} \times N_{AVG\_Obj}\right), E_{VRMLBrowser}\right) \quad \text{.....(4.27)}$$

โดยที่ $C_{Nav\_AVG\_Obj}$ = ชนิดออบเจ็กต์แบบต่างๆ และคุณสมบัติเพิ่มเติมที่ใส่ให้กับออบเจ็กต์ VRML ในช่วง Navigate

N_{AVG_Obj} = จำนวนออบเจ็กต์โดยเฉลี่ยที่ VRML Browser ต้องประมวลผลต่อหนึ่ง view

$E_{VRMLBrowser}$ = ประสิทธิภาพของปลั๊กอิน VRML Browser ในการประมวลผล ด้วยการวัดเวลาตอบสนองที่ผู้ใช้ส่งความต้องการให้กับปลั๊กอิน VRML Browser ในการประมวลผลจนกระทั่งปลั๊กอิน VRML Browser ประมวลผลครบถ้วน มีหน่วยเป็นจำนวนออบเจ็กต์ต่อวินาที

$f_{VRMLBrowser}\left(\left(C_{Nav\_AVG\_Obj} \times N_{AVG\_Obj}\right), E_{VRMLBrowser}\right)$ = เป็นเวลาที่ใช้ในการประมวลผลของปลั๊กอิน VRML Browser โดยขึ้นอยู่กับจำนวนออบเจ็กต์และชนิดของออบเจ็กต์แบบต่างๆ รวมถึงคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML รวมทั้งประสิทธิภาพของปลั๊กอิน VRML Browser ด้วยการวัดเวลาในการประมวลผลของปลั๊กอิน VRML Browser ต่อหนึ่งออบเจ็กต์ มีหน่วยเป็นวินาที

4.2 สรุปการวิเคราะห์ขั้นตอนการทำงานหลักของระบบ

จากการวิเคราะห์ขั้นตอนการทำงานหลักของระบบในหัวข้อ 4.1 ซึ่งแบ่งออกเป็น 2 ขั้นตอนคือ T_{Load} และ $T_{Navigate}$ โดยที่จะทำการพิสูจน์สมการต่างๆ ด้วยการทดลองในบทที่ 5

$$T_{Load} \approx T_{Min} + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) + \left[S_{Min} + \left(S_{AVG\_Obj} \times N_{AVG\_Obj}\right)\right] \times \left[\frac{1}{R_{CGI\_WS\_Rate}} + \frac{1}{R_{HTTPRate}}\right] \quad \text{.....(4.25)}$$

$$T_{Nuvigate} \approx f_{VRMLBrowser}\left(\left(C_{Nav\_AVG\_Obj} \times N_{AVG\_Obj}\right), E_{VRMLBrowser}\right) \quad \text{.....(4.27)}$$

บทที่ 5

การทดลองและผลการทดลองของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

5.1 ผลการทดลองการใช้ระบบ

จากหลักการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ที่ได้กล่าวไปแล้วในบทที่ 3 จึงได้นำมาพัฒนาระบบขึ้น ซึ่งในส่วนนี้จะเป็นการแสดงผลการทดสอบการใช้งานระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML โดยสมมติสถานการณ์ว่า ผู้บริหารเครือข่ายได้รับแจ้งปัญหาเกี่ยวกับเครือข่ายจากผู้ใช้งาน ผู้บริหารเครือข่ายจึงเข้าไปตรวจสอบดูสถานะของอุปกรณ์ที่สงสัยว่าจะมีปัญหาพร้อมทั้งแจ้งตำแหน่งที่ตั้งของอุปกรณ์ให้เจ้าหน้าที่สนามเข้าไปดำเนินการแก้ไขต่อไป โดยผู้บริหารเครือข่ายจะใช้ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML นี้เป็นเครื่องมือในการช่วยปฏิบัติภารกิจดังกล่าว โดยการทดลองทั้งหมดที่เกิดขึ้นในวิทยานิพนธ์นี้ได้ควบคุมสภาพแวดล้อม อันได้แก่ ประสิทธิภาพของเครื่องไคลเอนต์และเซิร์ฟเวอร์, ระบบจัดการฐานข้อมูลและปลั๊กอิน VRML Browser เพื่อศึกษาเวลาในการประมวลผลว่าขึ้นอยู่กับจำนวนออบเจ็กต์, ชนิดของออบเจ็กต์ VRML และคุณสมบัติต่างๆ ที่เพิ่มเติมมาในออบเจ็กต์ ดังรายละเอียดเครื่องมือที่ใช้ในการทดลองดังต่อไปนี้

เครื่องมือที่ใช้ในการทดลอง

- Web Browser = Netscape Navigator 4.5
- Plug-in VRML Browser = Cosmo Player 2.1
- ระบบเว็บเซิร์ฟเวอร์ = IIS (Microsoft Internet Information Server) 4.0
- ระบบจัดการฐานข้อมูล (DBMS : Database Management System) = SQL Server 7.0
- Server = OS WinNT, CPU Pentium 300, RAM 64 MB, HardDisk 6 GB
- Client = OS Win95, CPU Pentium 133, RAM 32 MB, HardDisk 1.2 GB

ในการทดลองต่อไปนี้ จะไม่ทำการพิจารณาถึงเวลาของ HardDisk Seek ในการเข้าถึงข้อมูล, ขนาดของหน่วยความจำ, ความหนาแน่นของเครือข่ายและทดลองโดยมีผู้ใช้เพียงคนเดียว

ผลการทดลองการใช้ระบบเบื้องต้นมีรายละเอียดดังนี้

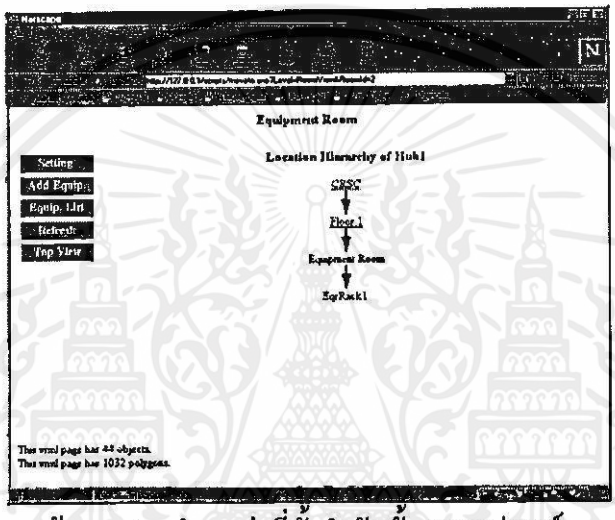
1. ผู้บริหารเครือข่ายเข้าสู่ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วยเว็บเบราเซอร์
2. ต่อจากนั้นผู้บริหารเครือข่ายก็จะทำการคลิกเลือกชื่อของอุปกรณ์ที่ต้องการตรวจสอบจากรายชื่อของ List box Viewpoint ดังรูปที่ 5.1

รูปที่ 5.1 แสดงตัวอย่างหน้าจอการคลิกเลือกชื่ออุปกรณ์ที่ต้องการตรวจสอบจาก List box Viewpoint

3. เมื่อได้ทำการเลือกอุปกรณ์แล้ว VRML Browser จะ Navigate ไปยังตำแหน่งที่อุปกรณ์ดังกล่าวถูกติดตั้งอยู่ ซึ่งจะเห็นสถานะของอุปกรณ์นั้น จากรูปที่ 5.2 จะพบว่ามีพอร์ทหนึ่ง fail อยู่

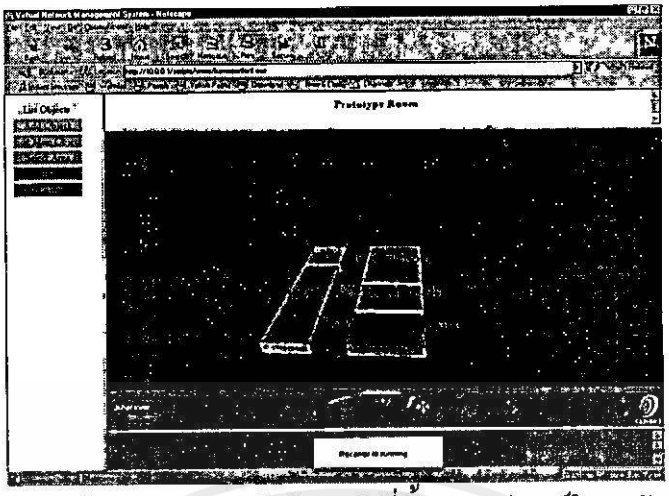
รูปที่ 5.2 แสดงตัวอย่างหน้าจอแสดงสถานะของอุปกรณ์

4. จากตำแหน่งของอุปกรณ์ที่ผู้บริหารเครือข่ายได้เห็นใน VRML นั้น ผู้บริหารเครือข่ายก็จะทราบว่าอุปกรณ์ถูกติดตั้งอยู่ตำแหน่งบนสุดของ Rack แต่ยังไม่ทราบว่าตำแหน่งที่เห็นนั้น อยู่ ณ ที่ใดในสภาพแวดล้อมจริง ผู้บริหารเครือข่ายจึงต้องการข้อมูลมากกว่านั้นเพื่อสามารถแจ้งให้เจ้าหน้าที่สนามทราบถึงตำแหน่งดังกล่าวได้ชัดเจนขึ้น ซึ่งสามารถทำได้โดยคลิกเลือกที่ตัวอุปกรณ์ โดยจะแสดงเป็นลักษณะของลำดับชั้นของตำแหน่งที่ตั้งอุปกรณ์ดังกล่าว นั่นคืออุปกรณ์ที่มีพอร์ทหนึ่ง fail อยู่นั้นคืออุปกรณ์ชื่อ Hub 1 ตั้งอยู่ที่ตึก CRSC ชั้น 1 ห้อง Equipment Room บน EqrRack1 ดังรูปที่ 5.3

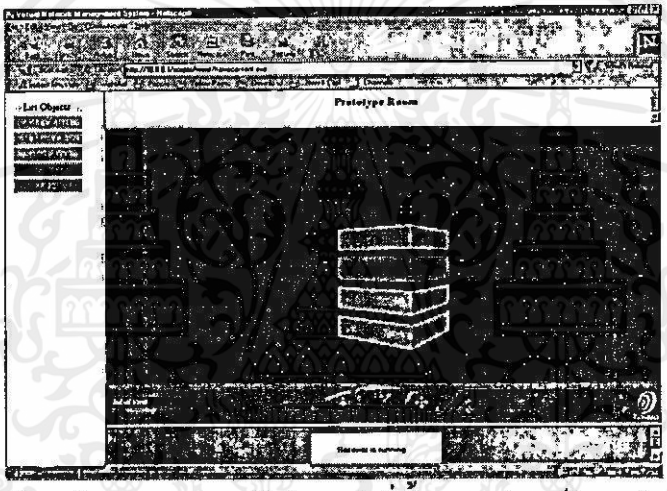


รูปที่ 5.3 แสดงตัวอย่างหน้าจอแสดงตำแหน่งที่ตั้งลำดับชั้นของอุปกรณ์

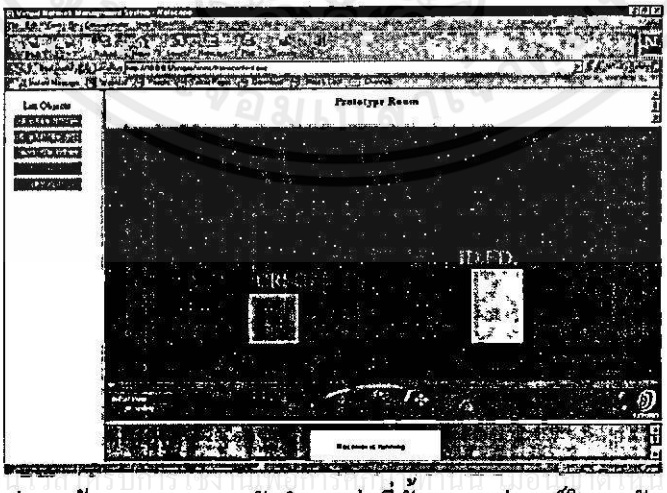
5. โดยในแต่ละระดับของข้อมูลตำแหน่งที่ตั้งจะมีแผนผังแสดงตำแหน่งที่ตั้งของอุปกรณ์ในแต่ละระดับอยู่ด้วย อย่างเช่น ถ้าคลิกที่ชื่อห้องหรือชั้นหรืออาคาร ก็จะแสดงสัญลักษณ์ของแผนผังรวมของห้อง, ชั้น, อาคารต่างๆ โดยห้อง, ชั้น, อาคารที่ติดตั้งอุปกรณ์ที่เราสนใจจะแสดงเป็นสีแดง ได้ผลลัพธ์ดังรูปที่ 5.4-5.6 ตามลำดับ



รูปที่ 5.4 แสดงตัวอย่างหน้าจอแสดงแผนผังตำแหน่งที่ตั้งของอุปกรณ์ในระดับห้อง



รูปที่ 5.5 แสดงตัวอย่างหน้าจอแสดงแผนผังตำแหน่งที่ตั้งของอุปกรณ์ในระดับชั้น



รูปที่ 5.6 แสดงตัวอย่างหน้าจอแสดงแผนผังตำแหน่งที่ตั้งของอุปกรณ์ในระดับอาคาร

6. และจากข้อมูลที่ได้จากแผนผังในทุกระดับ ผู้บริหารเครือข่ายก็สามารถที่จะนำมาหาเส้นทางในการไปยังอุปกรณ์และแผนที่ภาพรวมที่ผู้บริหารเครือข่ายอยู่เพื่อไปยังสถานที่ที่อุปกรณ์เครือข่ายติดตั้งอยู่ได้ ดังนั้นผู้บริหารเครือข่ายสามารถนำข้อมูลตำแหน่งที่ตั้งของอุปกรณ์ที่ได้ทั้งหมดทำการแจ้งให้กับเจ้าหน้าที่สนามเข้าไปทำการแก้ไขปัญหาที่ตัวอุปกรณ์ได้อย่างชัดเจนและรวดเร็วขึ้น

จากผลการทดลองการทำงานเบื้องต้น จะเห็นว่าผู้บริหารเครือข่ายสามารถใช้ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ในการตรวจสอบหาตำแหน่งที่ตั้งของอุปกรณ์ได้อย่างชัดเจนมากขึ้น ทำให้สามารถสื่อสารกับเจ้าหน้าที่ผู้รับผิดชอบได้ถูกต้อง ซึ่งจะส่งผลทำให้การแก้ปัญหาเครือข่ายที่เกิดขึ้นดังกล่าว สามารถทำได้อย่างถูกต้องและรวดเร็วมากขึ้นไปด้วย

5.2 การทดลองและผลการทดลองเพื่อพิสูจน์บทวิเคราะห์การทำงานของระบบ

เพื่อพิสูจน์และประกอบการวิเคราะห์การทำงานของระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ในบทที่ 4 ในบทนี้จึงได้ทำการทดลองกับสภาพแวดล้อมจริงในระบบจัดการเครือข่ายเพื่อทำการวิเคราะห์ว่ามีปัจจัยใดบ้างที่มีผลต่อระบบดังที่ทำการวิเคราะห์ไว้แล้ว อย่างเช่น

- จำนวนออบเจ็กต์ของแต่ละหน้าจอ (View) หรือ
- คุณสมบัติของภาษา VRML ที่นำมาใช้ในการพัฒนาระบบ อย่างเช่น ชนิดของออบเจ็กต์ที่นำมาใช้ (Box, Sphere, Cylinder, Cone) หรือ
- คุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ใน VRML เพื่อความเหมือนจริง

ต่อไปนี้จะแสดงถึงการทดลอง, วิธีการทดลองและผลการทดลองเพื่อหาค่าของเวลาแต่ละตัวจากสมการในบทที่ 4 ในหัวข้อที่ 4.1.1 และ 4.1.2

1. จากสมการ (4.3) $T_{WB\_to\_WS} = T_{WB\_Proc\_1} + \frac{S_{URL}}{R_{HTTPRate}}$

<u>การทดลอง</u>

$T_{WB\_to\_WS}$: วัดเวลาเริ่มตั้งแต่เมื่อเว็บเบราเซอร์ทำการส่ง URL ให้กับเว็บเซิร์ฟเวอร์ โดยส่ง URL รูปแบบต่างๆ ที่ใช้กับระบบจัดการเครือข่ายผ่านเว็บด้วย VRML ด้วยการใช้ Protocol Analyzer เป็นตัวจับเวลาที่ติดตั้งอยู่บนฝั่งเซิร์ฟเวอร์และไคลเอนต์ หลังจากนั้นติดตั้ง NTP (Network Time Protocol) อันเป็นโปรโตคอลเพื่อให้เวลาที่ฝั่งเว็บเซิร์ฟเวอร์และไคลเอนต์มีเวลาใกล้เคียงกันมากที่สุด [19] แล้วจับเวลาจนกระทั่งเว็บเซิร์ฟเวอร์ได้รับข้อมูลครบถ้วน (หน่วยเป็นวินาที)

โดยในการทดลองจะทำการควบคุมสภาพแวดล้อมในการทดลองเพื่อคุมประสิทธิภาพของเครื่องเว็บไคลเอนต์และเว็บเซิร์ฟเวอร์ เพื่อมุ่งประเด็นในการทดลองว่าขนาดของ URL มีผลกับเวลาในการทำงานของ $T_{WB\_to\_WS}$

S_{URL} : วัดขนาดของ URL ด้วยการใช้ Protocol Analyzer เป็นตัววัดขนาด (หน่วยเป็นไบต์)

$R_{HTTPRate}$: วัดขนาดความเร็วของ HTTP Transmission Rate โดยเฉลี่ย ณ เวลาที่เว็บเบราเซอร์ทำการส่ง URL ให้กับเว็บเซิร์ฟเวอร์ (หน่วยเป็น Kbps)

$T_{WB\_Proc\_1}$: เกิดจากการคำนวณ $T_{WB\_Proc\_1} = T_{WB\_to\_WS} - \frac{S_{URL}}{R_{HTTPRate}}$

ผลการทดลอง ดังแสดงในตารางที่ 5.1 การทดลองการวัดเวลาของ $T_{WB\_to\_WS}$

วิเคราะห์ผลการทดลอง

จากผลการวัดเวลา $T_{WB\_to\_WS}$ ดังตารางที่ 5.1 จะพบว่าขึ้นอยู่กับขนาดของ URL ที่ส่งข้ามเครือข่ายกันระหว่างเว็บเบราเซอร์และเว็บเซิร์ฟเวอร์รวมถึงความเร็วของเครือข่ายที่ทำการรับส่ง ณ เวลานั้นด้วย และจากผลการทดลองสามารถประมาณค่าเฉลี่ยโดยประมาณของ $T_{WB\_to\_WS} \approx 0.04389$ วินาที

2. จากสมการ (4.7) $T_{WS\_to\_CGI} \approx f_{WS}(E_{WS}) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}}$

การทดลอง

$T_{WS\_to\_CGI}$: เนื่องจากเป็นการทำงานภายในของเว็บเซิร์ฟเวอร์ซึ่งไม่สามารถจับเวลาในการติดต่อระหว่างเว็บเซิร์ฟเวอร์และ CGI ได้ โดยเว็บเซิร์ฟเวอร์จะทำการเรียกแอปพลิเคชั่น CGI แล้วเริ่มต้นการประมวลผลโปรแกรม CGI ทำการติดต่อกับระบบปฏิบัติการ เพื่อทำการสร้างโปรเซสขึ้นมาใหม่ ซึ่งทำหน้าที่รันโปรแกรม CGI ซึ่งถ้าใช้ระบบปฏิบัติการยูนิกซ์ กระบวนการนี้เรียกว่า fork แต่ถ้าใช้ Win32 API (เช่น Windows NT) จะเรียกกระบวนการนี้ว่า CreateProcess เมื่อระบบปฏิบัติการสร้างโปรเซสนี้แล้วจะต้องทำการจองหน่วยความจำและสภาพแวดล้อมต่างๆ ซึ่งสภาพแวดล้อมเหล่านี้จะเป็นข้อมูลที่เกี่ยวกับการร้องขอ (Request) จากเครื่องลูกข่าย ดังนั้นจะทำการทดลองเวลาทั้งหมด แล้วผลการทดลองจะได้จากการคำนวณจากสมการ (5.1)

ตารางที่ 5.1 แสดงผลการทดลองการวัดเวลาของ $T_{WB\_to\_WS}$ (sec)

ครั้งที่	URL	S_{URL} (byte)	R_{HTTP_Rate} (Kbps)	$\frac{S_{URL}}{R_{HTTP\_Rate}}$ (sec	$T_{WB\_Proc\_1}$ (sec)	$T_{WB\_to\_WS}$ (sec)
1	http://10.0.0.01/scripts/vnms/test.exe?shape=box&caseno=x&no=xxx	62	16.4	0.00369	0.04	0.04369
2	http://10.0.0.01/scripts/vnms/test.exe?shape=box&caseno=x&no=xxxx	63	16.4	0.00375	0.04	0.04375
3	http://10.0.0.01/scripts/vnms/test.exe?shape=box&caseno=x&no=xxxxx	64	16.4	0.00381	0.04	0.04381
4	http://10.0.0.01/scripts/vnms/test.exe?shape=sphere&caseno=x&no=xxx	65	16.4	0.00387	0.04	0.04387
5	http://10.0.0.01/scripts/vnms/test.exe?shape=sphere&caseno=x&no=xxxx	66	16.4	0.00393	0.04	0.04393
6	http://10.0.0.01/scripts/vnms/test.exe?shape=sphere&caseno=x&no=xxxxx	67	16.4	0.00399	0.04	0.04399
7	http://10.0.0.01/scripts/vnms/test.exe?shape=cylinder&caseno=x&no=xxx	67	16.4	0.00399	0.04	0.04399
8	http://10.0.0.01/scripts/vnms/test.exe?shape=cylinder&caseno=x&no=xxxx	68	16.4	0.00405	0.04	0.04405
9	http://10.0.0.01/scripts/vnms/test.exe?shape=cylinder&caseno=x&no=xxxxx	69	16.4	0.00411	0.04	0.04411
10	http://10.0.0.01/scripts/vnms/test.exe?shape=cone&caseno=x&no=xxx	63	16.4	0.00375	0.04	0.04375
11	http://10.0.0.01/scripts/vnms/test.exe?shape=cone&caseno=x&no=xxxx	64	16.4	0.00381	0.04	0.04381
12	http://10.0.0.01/scripts/vnms/test.exe?shape=cone&caseno=x&no=xxxxx	65	16.4	0.00387	0.04	0.04387
					$sum(T_{WB\_to\_WS})$ (sec)	0.52662
					$avg(T_{WB\_to\_WS})$ (sec)	0.04389

หมายเหตุ : $avg\ (R_{HTTP\_Rate})$ = 16.4 Kbps

$avg(T_{WB\_Proc\_1})$ = 0.04 second

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx T_{Load} - \left(T_{WB\_to\_WS} + T_{CGI\_to\_DBMS} + T_{WS\_to\_WB}\right) \quad \ldots(5.1)$$

ผลการทดลอง ดังแสดงในตารางที่ 5.5 การทดลองการวัดเวลาของ

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$$

3. จากสมการ (4.13) $T_{CGI\_to\_DBMS} \approx f_{DBMS}\left(N_{Obj}, E_{DBMS}\right)$

การทดลอง

$T_{CGI\_to\_DBMS}$:วัดเวลาเริ่มตั้งแต่ CGI ทำการประมวลผล หลังจากรับพารามิเตอร์จากเว็บเซิร์ฟเวอร์แล้ว โดยจะส่งคำสั่งในรูปแบบ SQL ให้กับระบบจัดการฐานข้อมูลเพื่อประมวลผล แล้วส่งผลลัพธ์คืนกลับให้กับ CGI เพื่อรวบรวม VRML Script รอส่งต่อให้กับเว็บเซิร์ฟเวอร์ ซึ่งพารามิเตอร์ที่ CGI ได้รับเป็นคำร้องขอจากผู้ใช้ที่ระบุผ่าน URL ว่าต้องการดูข้อมูลในการจัดการเครือข่ายลักษณะใด เช่น มีจำนวนออบเจ็กต์ใน view นั้นจำนวนเท่าไร, และชนิดของออบเจ็กต์ VRML เป็นแบบใด รวมถึงมีการใส่ Texture, Shading ให้กับออบเจ็กต์ VRML เพื่อให้ดูเหมือนจริงหรือไม่

ดังนั้นในการทดลองจึงควบคุมสภาพแวดล้อมในการทดลองเพื่อคุมประสิทธิภาพของเครื่องเว็บเซิร์ฟเวอร์และระบบจัดการฐานข้อมูล เพื่อมุ่งประเด็นในการทดลองว่าจำนวนออบเจ็กต์ ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML มีผลกับการเวลาในการทำงานของ $T_{CGI\_to\_DBMS}$

ผลการทดลอง ดังแสดงในตารางที่ 5.2 การทดลองการวัดเวลาของ $T_{CGI\_to\_DBMS}$

ตารางที่ 5.2 แสดงผลการทดลองการวัดเวลาของ $T_{CGI\_to\_DBMS}$ (sec)

(ก) $T_{CGI\_to\_DBMS}$ by Object Types

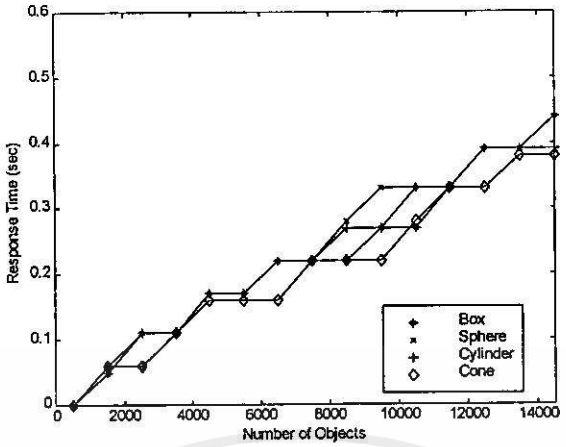
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500
Box	0.00	0.05	0.11	0.11	0.17	0.17	0.22	0.22	0.22	0.27	0.27	0.33	0.39	0.39
Sphere	0.00	0.06	0.11	0.11	0.17	0.17	0.22	0.22	0.28	0.33	0.33	0.33	0.39	0.39
Cylinder	0.00	0.06	0.06	0.11	0.17	0.17	0.22	0.22	0.27	0.27	0.33	0.33	0.39	0.39
Cone	0.00	0.06	0.06	0.11	0.16	0.16	0.16	0.22	0.22	0.22	0.28	0.33	0.33	0.38

(ข) $T_{CGI\_to\_DBMS}$ by Object Types & Add Shading

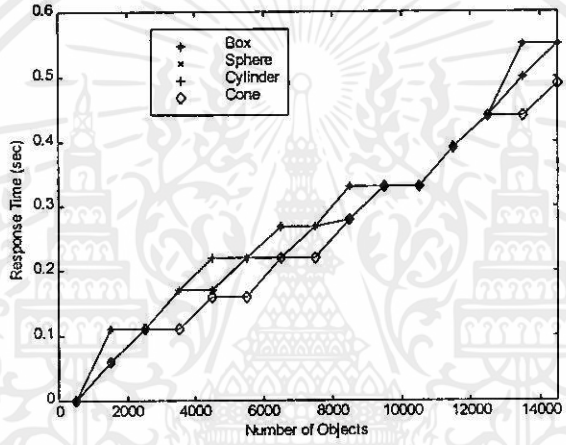
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500
Box	0.00	0.06	0.11	0.17	0.17	0.22	0.27	0.27	0.28	0.33	0.33	0.39	0.44	0.5
Sphere	0.00	0.11	0.11	0.17	0.17	0.22	0.27	0.27	0.33	0.33	0.33	0.39	0.44	0.55
Cylinder	0.00	0.11	0.11	0.17	0.22	0.22	0.22	0.27	0.33	0.33	0.33	0.39	0.44	0.55
Cone	0.00	0.06	0.11	0.11	0.16	0.16	0.22	0.22	0.28	0.33	0.33	0.39	0.44	0.44

(ค) $T_{CGI\_to\_DBMS}$ by Object Types & Add Texture

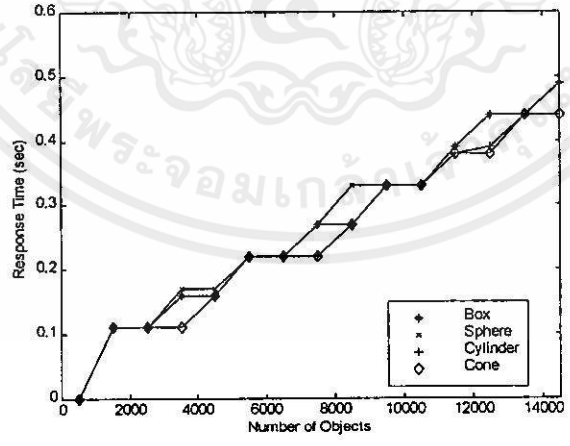
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500
Box	0.00	0.11	0.11	0.16	0.16	0.22	0.22	0.27	0.27	0.33	0.33	0.39	0.44	0.44
Sphere	0.00	0.11	0.11	0.17	0.17	0.22	0.22	0.27	0.33	0.33	0.33	0.39	0.44	0.44
Cylinder	0.00	0.11	0.11	0.16	0.16	0.22	0.22	0.22	0.27	0.33	0.33	0.38	0.39	0.44
Cone	0.00	0.11	0.11	0.11	0.16	0.22	0.22	0.22	0.27	0.33	0.33	0.38	0.38	0.44



(ก) แสดงผลการทดลองการวัดเวลา $T_{CGI\_to\_DBMS}$ by Object Types



(ข) แสดงผลการทดลองการวัดเวลา $T_{CGI\_to\_DBMS}$ by Object Types & Add Shading



(ค) แสดงผลการทดลองการวัดเวลา $T_{CGI\_to\_DBMS}$ by Object Types & Add Texture

รูปที่ 5.7 (ก,ข,ค) แสดงผลการทดลองการวัดเวลา $T_{CGI\_to\_DBMS}$

วิเคราะห์ผลการทดลอง

จากผลการวัดเวลา $T_{CGI\_to\_DBMS}$ ดังรูปที่ 5.7 จะพบว่าขึ้นอยู่กับจำนวนออบเจ็กต์ ยิ่งมีจำนวนออบเจ็กต์จำนวนมาก จะส่งผลให้ $T_{CGI\_to\_DBMS}$ ใช้เวลาในการประมวลผลมากขึ้นด้วย รวมถึงชนิดของออบเจ็กต์ VRML, คุณสมบัติต่างๆ เพิ่มเติมมาในออบเจ็กต์ อย่างเช่น การใส่ Texture หรือ Shading เพื่อให้โลกสามมิติมีความเหมือนจริงมากขึ้น สิ่งต่างๆ เหล่านี้ล้วนมีผลกับเวลาที่ต้องใช้ในการประมวลผลทั้งสิ้น

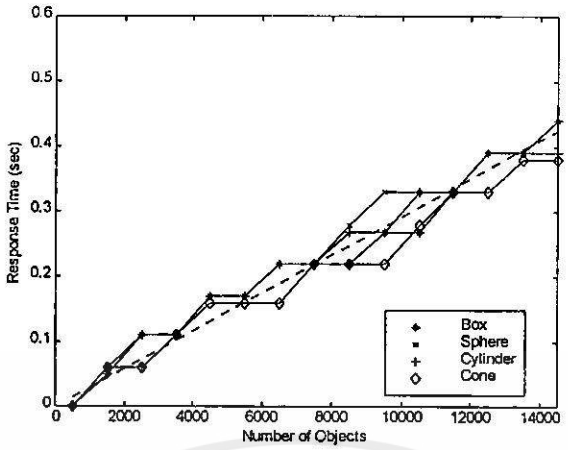
จากผลการทดลองที่เกิดขึ้นจากผลการทดลองตารางที่ 5.2 สามารถหาความสัมพันธ์ของสมการที่มีค่าความผิดพลาดเมื่อเปรียบเทียบผลจากการทดลองได้ดังนี้

$T_{CGI\_to\_DBMS}$ by Object Types	$T_{CGI\_to\_DBMS} = 0.000029281 \times N_{Obj}$ (ดังรูปที่ 5.8ก)	0.0122	1.8772e-004
$T_{CGI\_to\_DBMS}$ by Object Types & Add Shading	$T_{CGI\_to\_DBMS} = 0.000035796 \times N_{Obj\_Shading}$ (ดังรูปที่ 5.8ข)	0.0158	2.8865e-004
$T_{CGI\_to\_DBMS}$ by Object Types & Add Texture	$T_{CGI\_to\_DBMS} = 0.000033498 \times N_{Obj\_Texture}$ (ดังรูปที่ 5.8ค)	0.0140	2.5665e-004

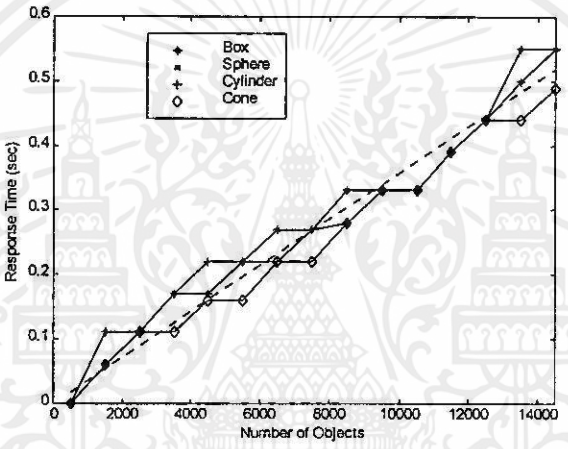
โดยที่ N_{Obj} = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone)

$N_{Obj\_Texture}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture

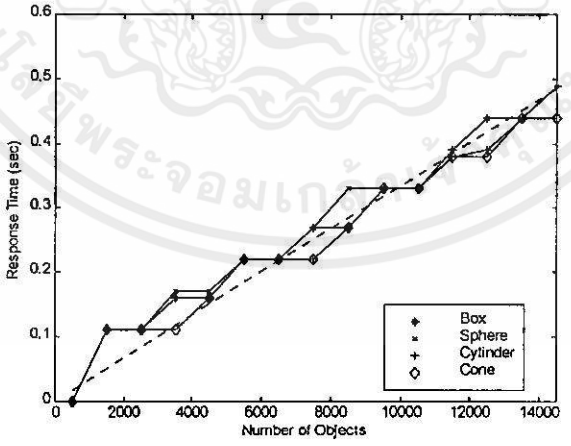
$N_{Obj\_Shading}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Shading



(ก) แสดงผลการหาสมการจากการวัดเวลา $T_{CGI\_to\_DBMS}$ by Object Types



(ข) แสดงผลการหาสมการจากการวัดเวลา $T_{CGI\_to\_DBMS}$ by Object Types & Add Shading



(ค) แสดงผลการหาสมการจากการวัดเวลา $T_{CGI\_to\_DBMS}$ by Object Types & Add Texture

รูปที่ 5.8 (ก,ข,ค) แสดงผลการหาสมการจากการวัดเวลา $T_{CGI\_to\_DBMS}$

ดังนั้นจากสมการ (4.13) สามารถประมาณค่าของเวลา $T_{CGI\_to\_DBMS}$ ได้ว่า

$$T_{CGI\_to\_DBMS} \approx \left[\left(0.000029281 + 0.0000035796 + 0.000033498\right)/3\right] \times N_{AVG\_Obj}$$

และค่าเฉลี่ยโดยประมาณของ $T_{CGI\_to\_DBMS}$ คือ

$$T_{CGI\_to\_DBMS} = 0.0000328583 \times N_{AVG\_Obj} \quad \ldots\ldots(5.2)$$

จากสมการ (5.2) ค่า

N_{AVG_Obj} หมายถึง จำนวนเฉลี่ยของออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) ที่ได้เพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture และ Shading

4. จากสมการ (4.18) $T_{CGI\_to\_WS} = \dfrac{S_{Min} + S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{CGI\_WS\_Rate}}$

<u>การทดลอง</u>

$T_{CGI\_to\_WS}$: เนื่องจากเป็นการทำงานภายในของเว็บเซิร์ฟเวอร์ซึ่งไม่สามารถจับเวลาในการติดต่อระหว่าง CGI และเว็บเซิร์ฟเวอร์ได้ โดยเมื่อ CGI ทำการประมวลผลตามคำร้องขอที่ส่งมาจากเว็บเซิร์ฟเวอร์และทำการรวบรวม VRML Script เรียบร้อยแล้ว ถัดมา CGI จะต้องส่งผลลัพธ์ VRML Script ที่ได้ให้กับเว็บเซิร์ฟเวอร์ ดังนั้นจะทำการทดลองเวลาทั้งหมด แล้วทำการคำนวณจากสมการ (5.1)

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx T_{Load} - \left(T_{WB\_to\_WS} + T_{CGI\_to\_DBMS} + T_{WS\_to\_WB}\right) \quad \ldots(5.1)$$

<u>ผลการทดลอง</u> ดังแสดงในตารางที่ 5.5 การทดลองการวัดเวลาของ

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$$

5. จากสมการ (4.21) $T_{WS\_to\_WB} = T_{WB\_Proc\_2} + \frac{S_{Min} + S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{HTTPRate}}$

<u>การทดลอง</u>

$T_{WS\_to\_WB}$: วัดเวลาเริ่มตั้งแต่เมื่อเว็บเซิร์ฟเวอร์รับ VRML Script จาก CGI แล้วทำการส่ง VRML Script ให้กับเว็บเบราเซอร์ โดยทำการทดลองส่งรูปแบบของ VRML Script ที่มีออบเจ็กต์ VRML แบบต่างๆ, คุณสมบัติต่างๆ กัน ด้วยจำนวนต่างๆ กัน หลังจากนั้นติดตั้ง NTP (Network Time Protocol) อันเป็นโปรโตคอลเพื่อให้เวลาที่ฝั่งเว็บเซิร์ฟเวอร์และเว็บเบราเซอร์มีเวลาใกล้เคียงกันมากที่สุด [19] แล้วจับเวลาจนกระทั่งเว็บเบราเซอร์ได้รับข้อมูลครบถ้วนด้วย Protocol Analyzer (หน่วยเป็นวินาที)

ดังนั้นในการทดลองจึงควบคุมสภาพแวดล้อมในการทดลองเพื่อคุมประสิทธิภาพของเครื่องเว็บเซิร์ฟเวอร์และเครื่องไคลเอนต์ เพื่อมุ่งประเด็นในการทดลองว่าจำนวนออบเจ็กต์, ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML มีผลกับการเวลาในการทำงานของ $T_{WS\_to\_WB}$ โดยจะทำการวัดขนาดของ VRML Script ตามจำนวนออบเจ็กต์, ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML ด้วยการใช้ Protocol Analyzer เป็นตัววัดขนาด (หน่วยเป็นไบต์)

$R_{HTTPRate}$: วัดขนาดความเร็วของ HTTP Transmission Rate โดยเฉลี่ย ณ เวลาที่เว็บเซิร์ฟเวอร์ทำการส่ง VRML Script ให้กับเว็บเบราเซอร์ (หน่วยเป็น Kbps)

$T_{WB\_Proc\_2}$:เกิดจากการคำนวณ

$$T_{WB\_Proc\_2} = T_{WS\_to\_WB} - \frac{S_{Min} + S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{HTTPRate}} \text{ (หน่วยเป็นวินาที)}$$

<u>ผลการทดลอง</u> ดังแสดงในตารางที่ 5.3 การทดลองการวัดเวลาของ $T_{WS\_to\_WB}$

ตารางที่ 5.3 แสดงผลการทดลองการวัดเวลาของ $T_{WS\_to\_WB}$

(ก) $T_{WS\_to\_WB}$ by Object Types

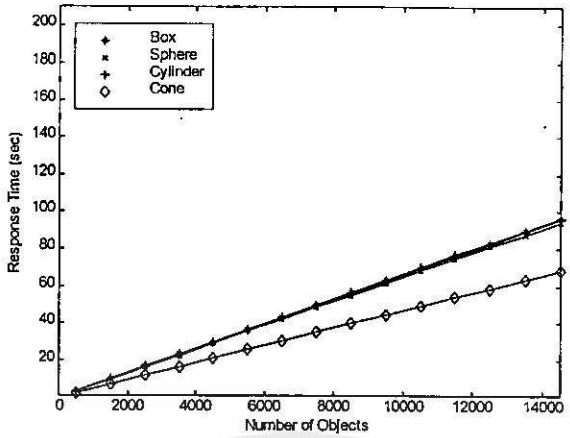
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Size of Box (byte)	56,015	168,015	280,015	392,015	504,015	616,015	728,015	840,015	952,015	1,064,015	1,176,015	1,288,015	1,400,015	1,512,015	1,624,015
Size of Sphere (byte)	54,515	163,515	272,515	381,515	490,515	599,515	708,515	817,515	926,515	1,035,515	1,144,515	1,253,515	1,362,515	1,471,515	1,580,515
Size of Cylinder (byte)	55,515	166,515	277,515	388,515	499,515	610,515	721,515	832,515	943,515	1,054,515	1,165,515	1,276,515	1,387,515	1,498,515	1,609,515
Size of Cone (byte)	39,515	118,515	197,515	276,515	355,515	434,515	513,515	592,515	671,515	750,515	829,515	908,515	987,515	1,066,515	1,145,515
Box Size/Obj (byte)	112.03000	112.01000	112.00600	112.00429	112.00333	112.00273	112.00231	112.00200	112.00176	112.00158	112.00143	112.00130	112.00120	112.00111	112.00103
Sphere Size/Obj (byte)	109.03000	109.01000	109.00600	109.00429	109.00333	109.00273	109.00231	109.00200	109.00176	109.00158	109.00143	109.00130	109.00120	109.00111	109.00103
Cylinder Size/Obj (byte)	111.03000	111.01000	111.00600	111.00429	111.00333	111.00273	111.00231	111.00200	111.00176	111.00158	111.00143	111.00130	111.00120	111.00111	111.00103
Cone Size/Obj (byte)	79.03000	79.01000	79.00600	79.00429	79.00333	79.00273	79.00231	79.00200	79.00176	79.00158	79.00143	79.00130	79.00120	79.00111	79.00103
S_{Avg_Obj} (byte)	102.75467														
S_{min} (byte)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R_{HTTP_Rate} (Kbps)	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4
$T_{WS\_Proc\_2}$ (sec)	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
$T_{WS\_Transfer\_VRMLScript}$ (Box) (sec)	3.33550	10.00470	16.67391	23.34312	30.01233	36.68153	43.35074	50.01995	56.68916	63.35836	70.02757	76.69678	83.36598	90.03519	96.70440
$T_{WS\_Transfer\_VRMLScript}$ (Sphere) (sec)	3.24618	9.73674	16.22731	22.71788	29.20845	35.69902	42.18958	48.68015	55.17072	61.66129	68.15186	74.64242	81.13299	87.62356	94.11413
$T_{WS\_Transfer\_VRMLScript}$ (Cylinder) (sec)	3.30572	9.91538	16.52505	23.13471	29.74437	36.35403	42.96369	49.57335	56.18301	62.79267	69.40233	76.01199	82.62165	89.23131	95.84098
$T_{WS\_Transfer\_VRMLScript}$ (Cone) (sec)	2.35298	7.05715	11.76133	16.46550	21.16967	25.87384	30.57802	35.28219	39.98636	44.69054	49.39471	54.09888	58.80306	63.50723	68.21140
$T_{WS\_to\_WB}$ (Box) (sec)	3.37550	10.04470	16.71391	23.38312	30.05233	36.72153	43.39074	50.05995	56.72916	63.39836	70.06757	76.73678	83.40598	90.07519	96.74440
$T_{WS\_to\_WB}$ (Sphere) (sec)	3.28618	9.77674	16.26731	22.75788	29.24845	35.73902	42.22958	48.72015	55.21072	61.70129	68.19186	74.68242	81.17299	87.66356	94.15413
$T_{WS\_to\_WB}$ (Cylinder) (sec)	3.34572	9.95538	16.56505	23.17471	29.78437	36.39403	43.00369	49.61335	56.22301	62.83267	69.44233	76.05199	82.66165	89.27131	95.88098
$T_{WS\_to\_WB}$ (Cone) (sec)	2.39298	7.09715	11.80133	16.50550	21.20967	25.91384	30.61802	35.32219	40.02636	44.73054	49.43471	54.13888	58.84306	63.54723	68.25140

(ข) $T_{WS\_to\_WB}$ by Object Types & Add Shading

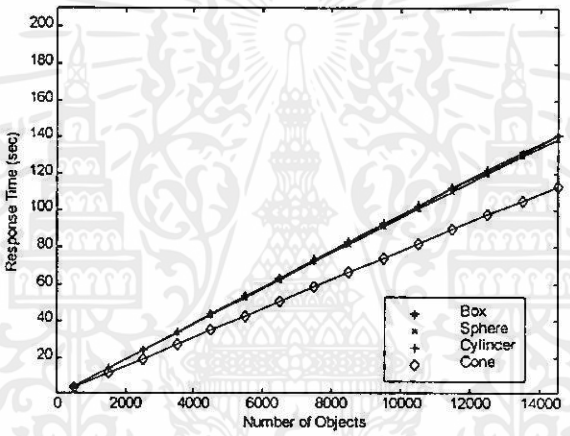
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Size of Box (byte)	82,015	246,015	410,015	574,015	738,015	902,015	1,066,015	1,230,015	1,394,015	1,558,015	1,722,015	1,886,015	2,050,015	2,214,015	2,378,015
Size of Sphere (byte)	80,515	241,515	402,515	563,515	724,515	885,515	1,046,515	1,207,515	1,368,515	1,529,515	1,690,515	1,851,515	2,012,515	2,173,515	2,334,515
Size of Cylinder (byte)	81,515	244,515	407,515	570,515	733,515	896,515	1,059,515	1,222,515	1,385,515	1,548,515	1,711,515	1,874,515	2,037,515	2,200,515	2,363,515
Size of Cone (byte)	65,515	196,515	327,515	458,515	589,515	720,515	851,515	982,515	1,113,515	1,244,515	1,375,515	1,506,515	1,637,515	1,768,515	1,899,515
Box Size/Obj (byte)	164.03000	164.01000	164.00600	164.00429	164.00333	164.00273	164.00231	164.00200	164.00176	164.00158	164.00143	164.00130	164.00120	164.00111	164.00103
Sphere Size/Obj (byte)	161.03000	161.01000	161.00600	161.00429	161.00333	161.00273	161.00231	161.00200	161.00176	161.00158	161.00143	161.00130	161.00120	161.00111	161.00103
Cylinder Size/Obj (byte)	163.03000	163.01000	163.00600	163.00429	163.00333	163.00273	163.00231	163.00200	163.00176	163.00158	163.00143	163.00130	163.00120	163.00111	163.00103
Cone Size/Obj (byte)	131.03000	131.01000	131.00600	131.00429	131.00333	131.00273	131.00231	131.00200	131.00176	131.00158	131.00143	131.00130	131.00120	131.00111	131.00103
S_{Avg_Obj} (byte)	154.75467														
S_{min} (byte)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R_{HTTP_Rate} (Kbps)	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4
$T_{WS\_Proc\_2}$ (sec)	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
$T_{WS\_Transfer\_VRMLScript}$ (Box) (sec)	4.88371	14.64933	24.41496	34.18058	43.94621	53.71183	63.47746	73.24308	83.00871	92.77433	102.53996	112.30558	122.07121	131.83683	141.60246
$T_{WS\_Transfer\_VRMLScript}$ (Sphere) (sec)	4.79439	14.38137	23.96836	33.55534	43.14233	52.72931	62.31630	71.90328	81.49027	91.07726	100.66424	110.25123	119.83821	129.42520	139.01218
$T_{WS\_Transfer\_VRMLScript}$ (Cylinder) (sec)	4.85393	14.56001	24.26609	33.97217	43.67825	53.38432	63.09040	72.79648	82.50256	92.20864	101.91472	111.62080	121.32687	131.03295	140.73903
$T_{WS\_Transfer\_VRMLScript}$ (Cone) (sec)	3.90119	11.70178	19.50237	27.30296	35.10355	42.90414	50.70473	58.50532	66.30591	74.10650	81.90710	89.70769	97.50828	105.30887	113.10946
$T_{WS\_to\_WB}$ (Box) (sec)	4.92371	14.68933	24.45496	34.22058	43.98621	53.75183	63.51746	73.28308	83.04871	92.81433	102.57996	112.34558	122.11121	131.87683	141.64246
$T_{WS\_to\_WB}$ (Sphere) (sec)	4.83439	14.42137	24.00836	33.59534	43.18233	52.76931	62.35630	71.94328	81.53027	91.11726	100.70424	110.29123	119.87821	129.46520	139.05218
$T_{WS\_to\_WB}$ (Cylinder) (sec)	4.89393	14.60001	24.30609	34.01217	43.71825	53.42432	63.13040	72.83648	82.54256	92.24864	101.95472	111.66080	121.36687	131.07295	140.77903
$T_{WS\_to\_WB}$ (Cone) (sec)	3.94119	11.74178	19.54237	27.34296	35.14355	42.94414	50.74473	58.54532	66.34591	74.14650	81.94710	89.74769	97.54828	105.34887	113.14946

(ค) $T_{WS\_to\_WB}$ by Object Types & Add Texture

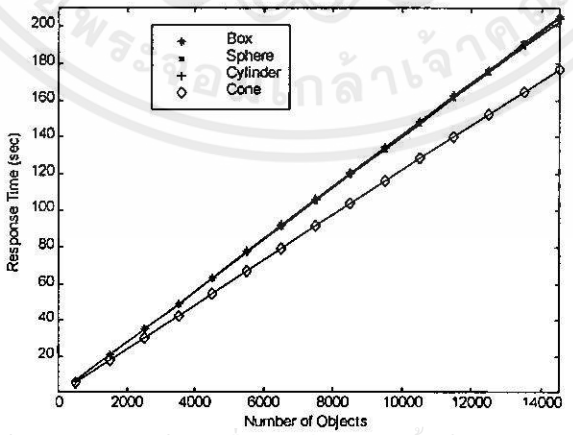
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Size of Box (byte)	119,015	357,015	595,015	833,015	1,071,015	1,309,015	1,547,015	1,785,015	2,023,015	2,261,015	2,499,015	2,737,015	2,975,015	3,213,015	3,451,015
Size of Sphere (byte)	117,515	352,515	587,515	822,515	1,057,515	1,292,515	1,527,515	1,762,515	1,997,515	2,232,515	2,467,515	2,702,515	2,937,515	3,172,515	3,407,515
Size of Cylinder (byte)	118,515	355,515	592,515	829,515	1,066,515	1,303,515	1,540,515	1,777,515	2,014,515	2,251,515	2,488,515	2,725,515	2,962,515	3,199,515	3,436,515
Size of Cone (byte)	102,515	307,515	512,515	717,515	922,515	1,127,515	1,332,515	1,537,515	1,742,515	1,947,515	2,152,515	2,357,515	2,562,515	2,767,515	2,972,515
Box Size/Obj (byte)	238.03000	238.01000	238.00600	238.00429	238.00333	238.00273	238.00231	238.00200	238.00176	238.00158	238.00143	238.00130	238.00120	238.00111	238.00103
Sphere Size/Obj (byte)	235.03000	235.01000	235.00600	235.00429	235.00333	235.00273	235.00231	235.00200	235.00176	235.00158	235.00143	235.00130	235.00120	235.00111	235.00103
Cylinder Size/Obj (byte)	237.03000	237.01000	237.00600	237.00429	237.00333	237.00273	237.00231	237.00200	237.00176	237.00158	237.00143	237.00130	237.00120	237.00111	237.00103
Cone Size/Obj (byte)	205.03000	205.01000	205.00600	205.00429	205.00333	205.00273	205.00231	205.00200	205.00176	205.00158	205.00143	205.00130	205.00120	205.00111	205.00103
S_{Avg_Obj} (byte)	228.75467														
S_{Obj} (byte)	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
R_{HTTP_Rate} (Kbps)	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4	16.4
$T_{WS\_Proc\_2}$ (sec)	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
$T_{WS\_Transfer\_VRMLScript}$ (Box) (sec)	7.08693	21.25899	35.43106	49.60312	63.77519	77.94725	92.11932	106.29138	120.46345	134.63552	148.80758	162.97965	177.15171	191.32378	205.49584
$T_{WS\_Transfer\_VRMLScript}$ (Sphere) (sec)	6.99761	20.99103	34.98446	48.97788	62.97131	76.96474	90.95816	104.95159	118.94501	132.93844	146.93187	160.92529	174.91872	188.91215	202.90557
$T_{WS\_Transfer\_VRMLScript}$ (Cylinder) (sec)	7.05715	21.16967	35.28219	49.39471	63.50723	77.61975	91.73227	105.84479	119.95731	134.06982	148.18234	162.29486	176.40738	190.51990	204.63242
$T_{WS\_Transfer\_VRMLScript}$ (Cone) (sec)	6.10441	18.31144	30.51847	42.72550	54.93253	67.13957	79.34660	91.55363	103.76066	115.96769	128.17472	140.38175	152.58878	164.79582	177.00285
$T_{WS\_to\_WB}$ (Box) (sec)	7.12693	21.29899	35.47106	49.64312	63.81519	77.98725	92.15932	106.33138	120.50345	134.67552	148.84758	163.01965	177.19171	191.36378	205.53584
$T_{WS\_to\_WB}$ (Sphere) (sec)	7.03761	21.03103	35.02446	49.01788	63.01131	77.00474	90.99816	104.99159	118.98501	132.97844	146.97187	160.96529	174.95872	188.95215	202.94557
$T_{WS\_to\_WB}$ (Cylinder) (sec)	7.09715	21.20967	35.32219	49.43471	63.54723	77.65975	91.77227	105.88479	119.99731	134.10982	148.22234	162.33486	176.44738	190.55990	204.67242
$T_{WS\_to\_WB}$ (Cone) (sec)	6.14441	18.35144	30.55847	42.76550	54.97253	67.17957	79.38660	91.59363	103.80066	116.00769	128.21472	140.42175	152.62878	164.83582	177.04285



(ก) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_WB}$ by Object Types



(ข) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_WB}$ by Object Types & Add Shading



(ค) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_WB}$ by Object Types & Add Texture

รูปที่ 5.9 (ก,ข,ค) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_WB}$

วิเคราะห์ผลการทดลอง

จากผลการวัดเวลา $T_{WS\_to\_WB}$ ดังรูปที่ 5.9 และผลการทดลองตารางที่ 5.3 จะพบว่าขึ้นอยู่กับขนาดของ VRML Script ที่ส่งข้ามเครือข่ายกันระหว่างเว็บเซิร์ฟเวอร์และเว็บเบราเซอร์ ซึ่งขึ้นอยู่กับจำนวนออบเจ็กต์ VRML ชนิดต่างๆ, คุณสมบัติต่างๆ รวมถึงความเร็วของเครือข่ายที่ทำการรับส่ง ณ เวลานั้นด้วย

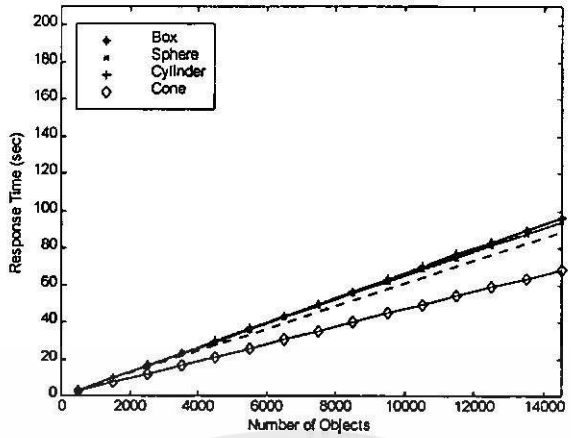
จากผลการทดลองที่เกิดขึ้นจากผลการทดลองตารางที่ 5.3 สามารถหาความสัมพันธ์ของสมการที่มีค่าความผิดพลาดเมื่อเปรียบเทียบผลจากการทดลองได้ดังนี้

$T_{WS\_to\_WB}$ by Object Types	$T_{WS\_to\_WB} \approx 0.0061 \times N_{Obj} - 0.0139$ (ดังรูปที่ 5.10ก)	3.6624	21.1415
$T_{WS\_to\_WB}$ by Object Types & Add Shading	$T_{WS\_to\_WB} \approx 0.0092 \times N_{Obj\_Shading} - 0.0139$ (ดังรูปที่ 5.10ข)	3.6620	3.6620
$T_{WS\_to\_WB}$ by Object Types & Add Texture	$T_{WS\_to\_WB} \approx 0.0136 \times N_{Obj\_Texture} - 0.0139$ (ดังรูปที่ 5.10ค)	3.6628	21.0365

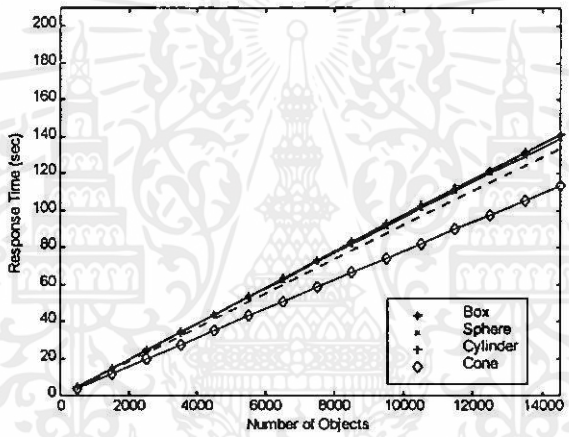
โดยที่ N_{Obj} = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone)

$N_{Obj\_Texture}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture

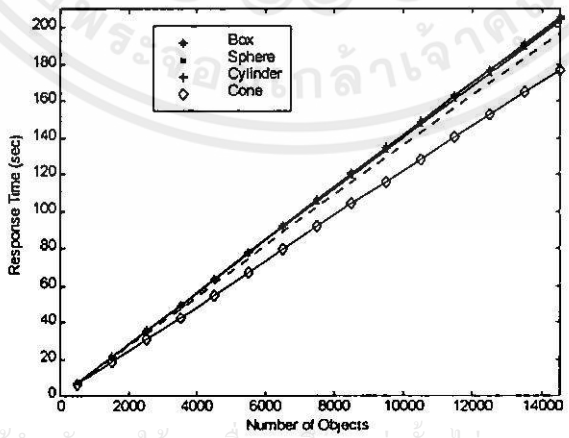
$N_{Obj\_Shading}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Shading



(ก) แสดงผลการหาสมการจากการวัดเวลา $T_{WS\_to\_WB}$ by Object Types



(ข) แสดงผลการหาสมการจากการวัดเวลา $T_{WS\_to\_WB}$ by Object Types & Add Shading



(ค) แสดงผลการหาสมการจากการวัดเวลา $T_{WS\_to\_WB}$ by Object Types & Add Texture

รูปที่ 5.10 (ก,ข,ค) แสดงผลการหาสมการจากการทดลองการวัดเวลา $T_{WS\_to\_WB}$

ดังนั้นจากสมการ (4.21) สามารถประมาณค่าของเวลา $T_{WS\_to\_WB}$ ได้ว่า

$$T_{WS\_to\_WB} \approx \left[(0.0061+0.0092+0.0136)/3\right] \times N_{AVG\_Obj} - (0.0139 \times 3)/3$$

และค่าเฉลี่ยโดยประมาณของ $T_{WS\_to\_WB}$ คือ

$$T_{WS\_to\_WB} \approx 0.0096 \times N_{AVG\_Obj} - 0.0139 \quad \text{.....(5.3)}$$

จากสมการ (5.3) ค่า

N_{AVG_Obj} หมายถึง จำนวนเฉลี่ยของออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) ที่ได้เพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture และ Shading

6. จากสมการ (4.23) $T_{WB\_to\_VRMLBrowser} \approx f_{WB}\left(N_{AVG\_Obj}, E_{WB}\right)$

<u>การทดลอง</u>

$T_{WB\_to\_VRMLBrowser}$: วัด $T_{WB\_to\_VRMLBrowser}$ ด้วยนาฬิกาจับเวลา วัดเวลาเริ่มตั้งแต่เมื่อเว็บเบราเซอร์ได้รับ VRML Script จากเว็บเซิร์ฟเวอร์แล้วทำการตรวจสอบชนิดของข้อมูลที่ได้รับเพื่อนำเสนอให้ถูกต้องตามรูปแบบที่ได้รับมา ถ้าเป็น VRML เว็บเบราเซอร์จะเรียกปลั๊กอิน VRML Browser ขึ้นมาประมวลผล โดยหลังจากเรียก VRML Browser ขึ้นมาทำงานแล้ว เว็บเบราเซอร์จะทำการส่ง VRML Script ให้กับ VRML Browser เพื่อทำการประมวลผลต่อไป

เนื่องจากเป็นการทำงานภายในของเว็บเบราเซอร์และปลั๊กอิน VRML Browser ซึ่งไม่สามารถจับเวลาในการติดต่อระหว่างเว็บเบราเซอร์และปลั๊กอิน VRML Browser ได้ ดังนั้นจะทำการทดลองเวลาทั้งหมด แล้วทำการคำนวณจากสมการ (5.1)

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx T_{Load} - \left(T_{WB\_to\_WS} + T_{CGI\_to\_DBMS} + T_{WS\_to\_WB}\right) \quad \text{...(5.1)}$$

ในการทดลองได้ทำการควบคุมสภาพแวดล้อมในการทดลองเพื่อคุมประสิทธิภาพของเครื่องไคลเอนต์และปลั๊กอิน VRML Browser เพื่อมุ่งประเด็นในการทดลองว่าจำนวนออบเจ็กต์ ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML มีผลกับการเวลาในการทำงานของ $T_{WB\_to\_VRMLBrowser}$

ผลการทดลอง ดังแสดงในตารางที่ 5.5 การทดลองการวัดเวลาของ

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$$

7. จากสมการ (4.23)

$$T_{Load} = T_{WB\_to\_WS} + T_{WS\_to\_CGI} + T_{CGI\_to\_DBMS} + T_{CGI\_to\_WS} + T_{WS\_to\_WB} + T_{WB\_to\_VRMLBrowser}$$

การทดลอง

T_{Load} : วัด T_{Load} ด้วยนาฬิกาจับเวลา วัดเวลาเริ่มตั้งแต่ส่วนที่เริ่มตั้งแต่ผู้ใช้งานทำการร้องขอการทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยเว็บเซิร์ฟเวอร์จะคอยฟังการร้องขอที่ส่งมาและทำการประมวลผลสคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซิร์ฟเวอร์แอปพลิเคชั่น (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผลและอาจจะมีการดึงข้อมูลจากฐานข้อมูล เพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกประมวลผลก็จะส่งผลลัพธ์กลับไปให้เว็บเซิร์ฟเวอร์ ต่อจากนั้นเว็บเซิร์ฟเวอร์จึงส่งผลลัพธ์ดังกล่าวกลับไปให้เว็บเบราเซอร์ที่ส่งการร้องขอมา

ผลการทดลอง ดังแสดงในตารางที่ 5.4 การทดลองการวัดเวลาของ T_{Load}

ตารางที่ 5.4 แสดงผลการทดลองการวัดเวลาของ T_{Load}

(ก) T_{Load} by Object Types

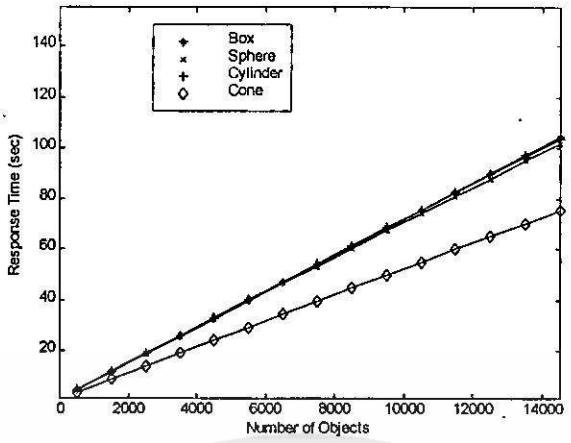
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500
Box	0.94	1.69	2.15	2.50	3.06	3.38	3.78	4.25	4.72	5.22	5.59	6.13	6.53	7.04
Sphere	1.13	1.59	2.15	2.69	3.10	3.44	4.18	4.31	4.82	5.53	5.88	6.19	6.68	7.13
Cylinder	1.07	1.63	2.28	2.72	3.03	3.59	4.06	4.41	4.85	5.47	5.91	6.38	6.84	7.16
Cone	1.00	1.53	1.97	2.34	2.84	3.22	3.72	4.10	4.56	5.09	5.31	5.90	6.28	6.72

(ข) T_{Load} by Object Types & Add Shading

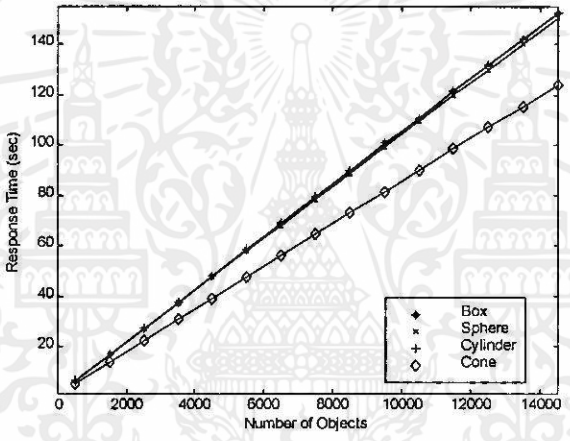
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500
Box	1.22	2	2.63	3.41	4.19	4.91	5.5	6.31	6.97	7.84	8.35	9.22	9.75	10.44
Sphere	1.31	2	2.78	3.53	4.21	4.93	5.67	6.38	6.97	7.85	8.5	9.26	9.75	10.63
Cylinder	1.31	2	2.75	3.5	4.18	4.93	5.75	6.38	7.12	7.85	8.53	9.31	9.97	10.75
Cone	1.16	1.94	2.62	3.28	3.91	4.79	5.22	6.1	6.79	7.22	7.97	8.67	9.4	9.97

(ค) T_{Load} by Object Types & Add Texture

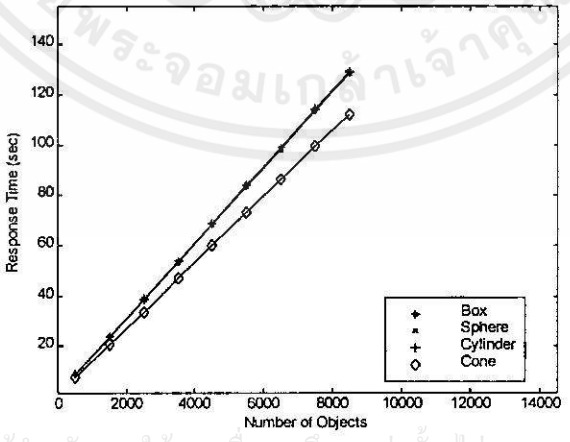
จำนวนออปเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500
Box	1.41	2.31	3.28	4.16	5.09	6.06	6.97	7.91	8.69
Sphere	1.43	2.31	3.28	4.16	5.25	6.06	7.13	8.26	9.17
Cylinder	1.34	2.28	3.25	4.29	5.28	6.12	7.12	8.21	9.03



(ก) แสดงผลการทดลองการวัดเวลา T_{Load} by Object Types



(ข) แสดงผลการทดลองการวัดเวลา T_{Load} by Object Types & Add Shading



(ค) แสดงผลการทดลองการวัดเวลา T_{Load} by Object Types & Add Texture

รูปที่ 5.11 (ก,ข,ค) แสดงผลการทดลองการวัดเวลา T_{Load}

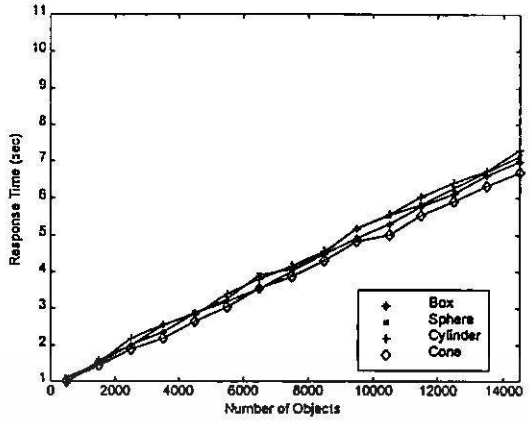
8. จากสมการ (5.1)

สำหรับการวัดเวลาของ $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ ในข้อ 2, 4 และ 6 ซึ่งเกิดจากการคำนวณในสมการ (5.1) ดังนี้

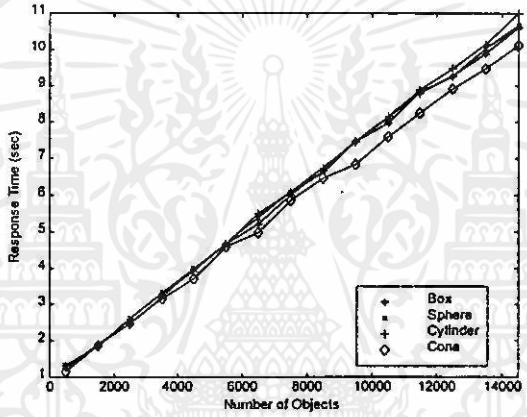
$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx T_{Load} - \left(T_{WB\_to\_WS} + T_{CGI\_to\_DBMS} + T_{WS\_to\_WB}\right) \quad \ldots(5.1)$$

<u>ผลการทดลอง</u> ดังแสดงในตารางที่ 5.5 การทดลองการวัดเวลาของ

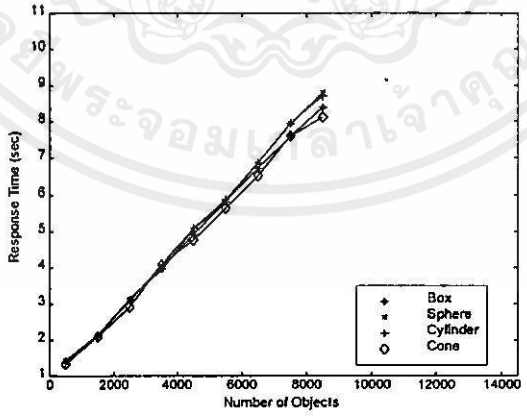
$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$$



(ก) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types



(ข) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_CGI} + T_{CGI\_to\_IVS} + T_{IVB\_to\_VRMLBrowser}$ by Object Types & Add Texture



(ค) แสดงผลการทดลองการวัดเวลา $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Texture

รูปที่ 5.12 (ก,ข,ค) แสดงผลการทดลองการวัดเวลา $T_{IVS\_to\_CGI} + T_{CGI\_to\_IVS} + T_{WB\_to\_VRMLBrowser}$

ตารางที่ 5.5 แสดงผลการทดลองการวัดเวลาของ $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$

(ก) $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types

จำนวนออบเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Box	0.04201	0.04283	0.04312	0.04324	0.04329	0.04334	0.04338	0.04338	0.04338	0.0434	0.04341	0.04342	0.04343	0.04343	0.04343
Sphere	0.04163	0.04291	0.04312	0.04318	0.04328	0.04332	0.04332	0.04338	0.04339	0.04338	0.04339	0.04341	0.04342	0.04342	0.04343
Cylinder	0.04175	0.04288	0.04303	0.04317	0.0433	0.0433	0.04334	0.04336	0.04338	0.04337	0.04339	0.0434	0.0434	0.04342	0.04341
Cone	0.04189	0.04287	0.0431	0.04322	0.04326	0.0433	0.04332	0.04334	0.04335	0.04335	0.04338	0.04338	0.04339	0.04339	0.0434

(ข) $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Shading

จำนวนออบเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Box	0.04145	0.04271	0.04297	0.04307	0.04307	0.04312	0.04316	0.04316	0.04317	0.04316	0.04318	0.04319	0.04322	0.04323	0.04323
Sphere	0.04127	0.04275	0.04291	0.04303	0.04307	0.04311	0.04314	0.04315	0.04318	0.04316	0.04317	0.04318	0.04322	0.04322	0.04322
Cylinder	0.04127	0.04271	0.04292	0.04302	0.04308	0.0431	0.04312	0.04314	0.04315	0.04316	0.04317	0.04318	0.0432	0.0432	0.0432
Cone	0.04157	0.04268	0.04293	0.04303	0.04309	0.04309	0.04317	0.04315	0.04316	0.0432	0.04319	0.04321	0.04321	0.04322	0.04323

(ค) $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Texture

จำนวนออบเจ็กต์	500	1,500	2,500	3,500	4,500	5,500	6,500	7,500	8,500
Box	0.04107	0.04257	0.04271	0.04284	0.04287	0.04291	0.04292	0.04294	0.0426
Sphere	0.04103	0.04257	0.04271	0.04284	0.04283	0.04291	0.04289	0.04289	0.04258
Cylinder	0.04121	0.04259	0.04272	0.04279	0.04282	0.0429	0.0429	0.04289	0.0426
Cone	0.04125	0.04254	0.04275	0.04276	0.04286	0.0429	0.04291	0.04291	0.04261

วิเคราะห์ผลการทดลอง

จากผลการวัดเวลา $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ ดังรูปที่ 5.12 จะพบว่าขึ้นอยู่กับจำนวนออบเจ็กต์ ยิ่งมีจำนวนออบเจ็กต์จำนวนมาก จะส่งผลให้ $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ ใช้เวลาในการประมวลผลมากขึ้นด้วย รวมถึงชนิดของออบเจ็กต์ VRML, คุณสมบัติต่างๆ เพิ่มเติมมาในออบเจ็กต์ อย่างเช่น การใส่ Texture หรือ Shading เพื่อให้โลกสามมิติมีความเหมือนจริงมากขึ้น สิ่งต่างๆ เหล่านี้ล้วนมีผลกับเวลาที่ต้องใช้ในการประมวลผลทั้งสิ้น

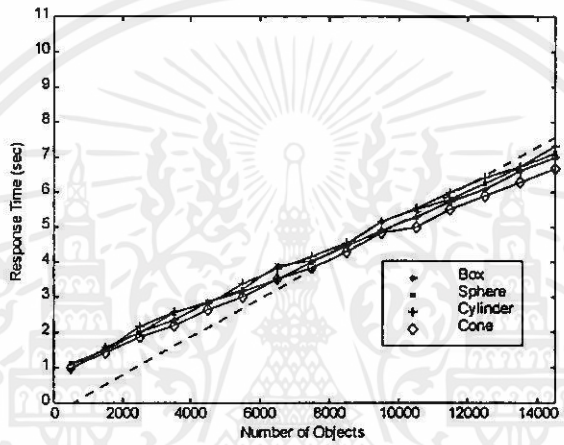
จากผลการทดลองที่เกิดขึ้นจากผลการทดลองตารางที่ 5.5 สามารถหาความสัมพันธ์ของสมการเชิงเส้นตรงและสมการโพลิโนเมียลโดยเลือกสมการที่มีค่าความผิดพลาดน้อยที่สุด เมื่อเปรียบเทียบผลจากการทดลองได้ดังนี้

$T_{WS\_to\_CGI} +$ $T_{CGI\_to\_WS} +$ $T_{WB\_to\_VRMLBrowser}$	$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx$ $0.00054263 \times N_{Obj} - 0.3015$ (ดังรูปที่ 5.13ก1)	0.3172	0.1139
by Object Types	$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx$ $0.00044415 \times N_{Obj}^{1.0077}$ (ดังรูปที่ 5.13ก2)	0.2937	0.0654
$T_{WS\_to\_CGI} +$ $T_{CGI\_to\_WS} +$ $T_{WB\_to\_VRMLBrowser}$	$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx$ $0.00079146 \times N_{Obj\_Shading} - 0.3048$ (ดังรูปที่ 5.13ข1)	0.3481	0.1101
by Object Types & Add Shading	$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx$ $0.00065953 \times N_{Obj\_Shading}^{1.0077}$ (ดังรูปที่ 5.13ข2)	0.3243	0.0527
$T_{WS\_to\_CGI} +$ $T_{CGI\_to\_WS} +$ $T_{WB\_to\_VRMLBrowser}$	$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx$ $0.0011 \times N_{Obj\_Texture} - 0.2853$ (ดังรูปที่ 5.13ค1)	0.3845	0.1189
by Object Types & Add Texture	$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx$ $0.00091437 \times N_{Obj\_Texture}^{1.0078}$ (ดังรูปที่ 5.13ค2)	0.3841	0.0455

โดยที่ N_{Obj} = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone)

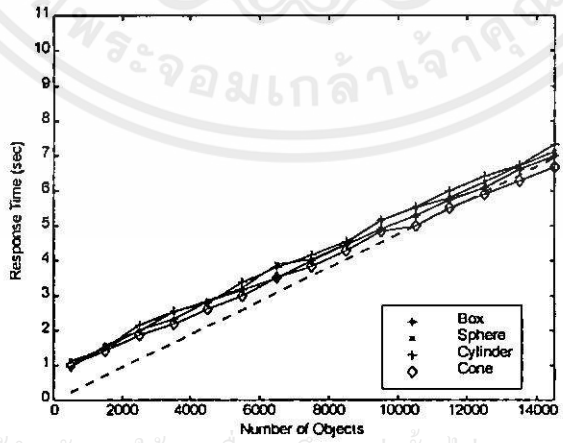
$N_{Obj\_Texture}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture

$N_{Obj\_Shading}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Shading



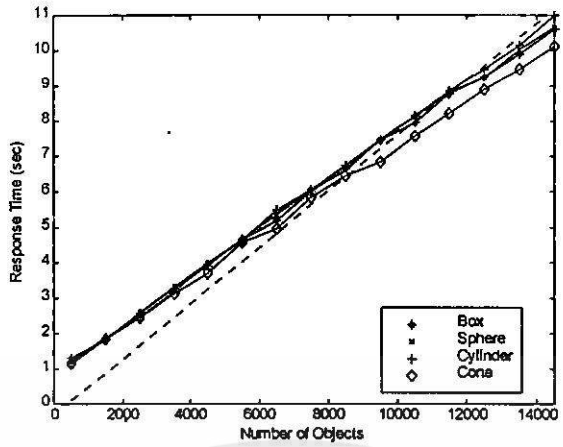
(ก1) แสดงผลการหาสมการเชิงเส้นตรงจากการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types



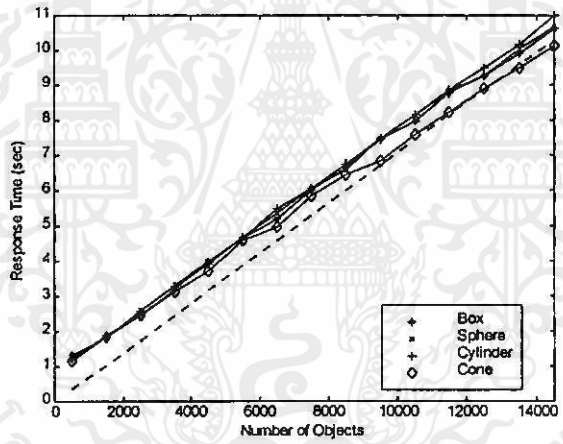
(ก2) แสดงผลการหาสมการเชิงโพลิโนเมียลจากการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types



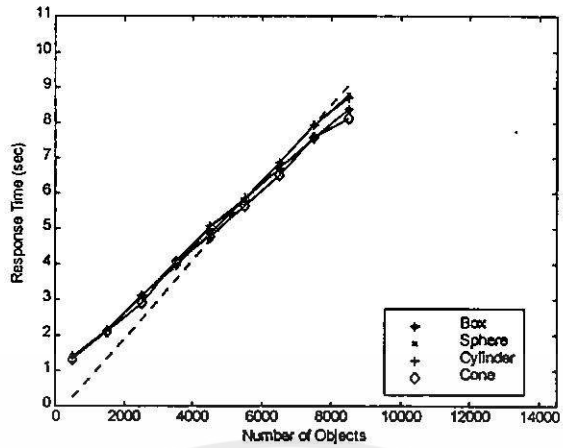
(ข1) แสดงผลการหาสมการเชิงเส้นตรงจากการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Shading



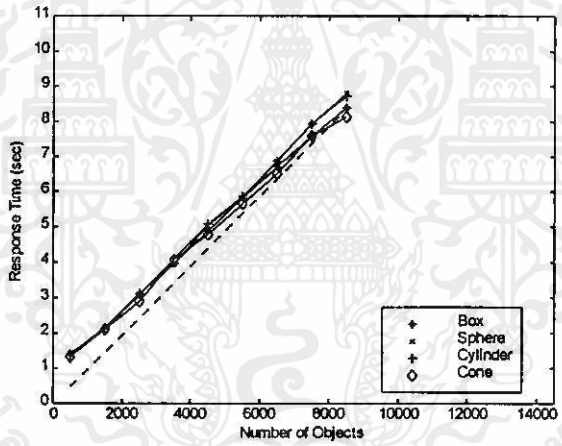
(ข2) แสดงผลการหาสมการเชิงโพลิโนเมียลจากการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Shading



(ค1) แสดงผลการหาสมการเชิงเส้นตรงจากการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Texture



(ค2) แสดงผลการหาสมการชิงโพลิโนเมียลจากการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ by Object Types & Add Texture

รูปที่ 5.13 แสดงผลการหาสมการจากการทดลองการวัดเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$

ดังนั้นจากสมการ (5.1) สามารถประมาณค่าของเวลา

$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ (เลือกสมการที่มีค่าความผิดพลาดน้อยที่สุด เพื่อเป็นสมการตัวแทน) ได้ว่า

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx \left[(0.00044415 + 0.00065953 + 0.00091437)\right] \times N_{AVG\_Obj}^{1.0077}$$

และค่าเฉลี่ยโดยประมาณของ $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser}$ คือ

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx 0.000672683 \times N_{AVG\_Obj}^{1.0077} \quad \text{.....(5.4)}$$

จากสมการ (5.4) ค่า

N_{AVG_Obj} หมายถึง จำนวนเฉลี่ยของออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) ที่ได้เพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture และ Shading

9. จากสมการ (4.27) $T_{Navigate} \approx f_{VRMLBrowser}\left(\left(C_{Nav\_AVG\_Obj} \times N_{AVG\_Obj}\right), E_{VRMLBrowser}\right)$

<u>การทดลอง</u>

$T_{Navigate}$: วัด $T_{Navigate}$ ด้วยนาฬิกาจับเวลา หลังจากเว็บเบราเซอร์ทำการส่ง VRML Script ให้กับปลั๊กอิน VRML Browser แล้ว VRML Browser จะทำการประมวลผลและนำเสนอในรูปแบบกราฟฟิก 3 มิติ

ในการทดลองได้ทำการควบคุมสภาพแวดล้อมในการทดลองเพื่อคุมประสิทธิภาพของเครื่องไคลเอนต์และปลั๊กอิน VRML Browser เพื่อมุ่งประเด็นในการทดลองว่าจำนวนออบเจ็กต์ ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML มีผลกับการเวลาในการทำงานของ $T_{Navigate}$

<u>ผลการทดลอง</u> ดังแสดงในตารางที่ 5.6 การทดลองการวัดเวลาของ $T_{Navigate}$

ตารางที่ 5.6 แสดงผลการทดลองการวัดเวลาของ $T_{Navigate}$

(ก) $T_{Navigate}$ by Object Types

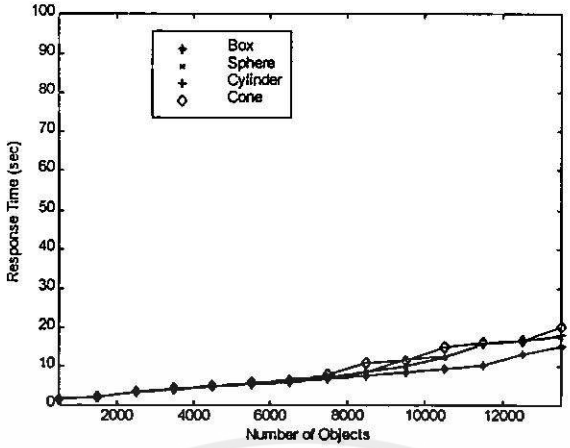
จำนวนออปเจ็กต์	500	1,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Box	1.58	1.92	3.27	3.85	4.56	5.18	5.99	6.86	7.69	8.51	9.38	10.39	13.34	15.20
Sphere	1.86	2.08	3.40	4.05	4.81	5.63	6.39	7.24	8.39	9.88	12.22	15.82	16.57	17.71
Cylinder	1.45	2.19	3.66	4.29	5.01	5.68	6.68	7.41	8.42	11.84	12.49	15.91	16.29	18.22
Cone	1.42	2.19	3.49	4.31	4.90	6.01	6.50	7.92	10.78	11.30	15.01	16.04	16.85	20.24
S_{min} (byte)	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S_{Avg_Obj} (byte)	0.00316													

(ข) $T_{Navigate}$ by Object Types & Add Shading

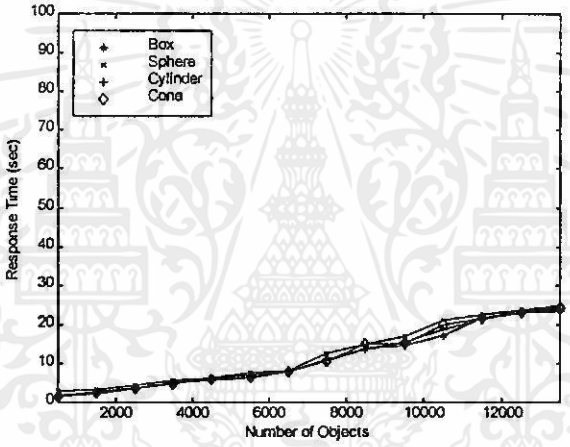
จำนวนออปเจ็กต์	500	1,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500	10,500	11,500	12,500	13,500	14,500
Box	1.59	2.15	3.5	4.85	5.66	6.25	7.87	11.28	13.72	14.62	17.18	21.6	22.85	23.5
Sphere	2.91	3.21	4.34	5.53	6.32	7.5	8.31	12.62	14.87	17.02	21.03	22.56	23.79	24.8
Cylinder	1.75	2.69	3.59	4.87	6.1	7.02	8.26	10.65	13.89	15.65	18.72	21.4	23.05	24.1
Cone	1.56	2.45	3.53	4.82	5.97	6.48	7.98	10.66	14.87	15.22	20.01	21.8	23.22	24.2
S_{min} (byte)	1	1	1	1	1	1	1	1	1	1	1	1	1	1
S_{Avg_Obj} (byte)	0.00391													

(ค) $T_{Navigate}$ by Object Types & Add Texture

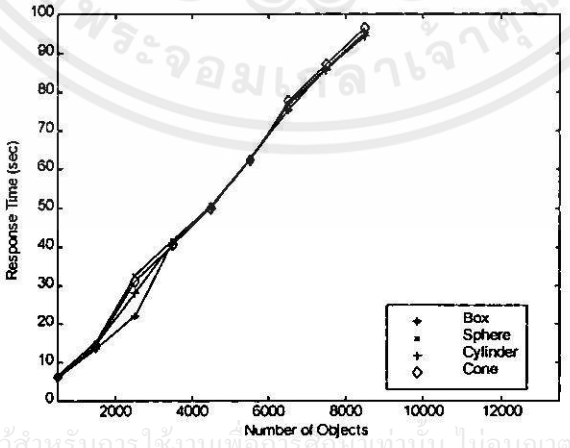
จำนวนออปเจ็กต์	500	1,500	3,500	4,500	5,500	6,500	7,500	8,500	9,500
Box	5.97	13.38	21.87	41.05	50.09	62.58	75.34	86.02	95.23
Sphere	6.78	15.15	32.56	41.53	50.78	63.12	77.1	86.3	95.44
Cylinder	6.25	14.29	28.02	40.66	50.38	62.88	76.42	85.66	94.31
Cone	6.05	14.47	31.06	40.53	49.81	62.32	77.8	87.03	96.42
S_{min} (byte)	1	1	1	1	1	1	1	1	1
S_{Avg_Obj} (byte)	0.25954								



(ก) แสดงผลการทดลองการวัดเวลา $T_{Navigate}$ by Object Types



(ข) แสดงผลการทดลองการวัดเวลา $T_{Navigate}$ by Object Types & Add Shading



(ค) แสดงผลการทดลองการวัดเวลา $T_{Navigate}$ by Object Types & Add Texture

รูปที่ 5.14 (ก,ข,ค) แสดงผลการทดลองการวัดเวลา $T_{Navigate}$

วิเคราะห์ผลการทดลอง

จากผลการวัดเวลา $T_{Navigate}$ จะพบว่าขึ้นอยู่กับเวลาในการประมวลผลของปลั๊กอิน VRML Browser ว่า VRML Script ประกอบด้วย VRML ออบเจ็กต์แบบใด, มีคุณสมบัติต่างๆ เพิ่มเติมมาในสคริปต์หรือไม่ อย่างเช่น การใส่ Texture หรือ Shading เพื่อให้โลก 3 มิติมีความเหมือนจริงมากขึ้น สิ่งต่างๆ เหล่านี้ล้วนมีผลกับเวลาที่ VRML Browser ต้องใช้ในการประมวลผลทั้งสิ้น

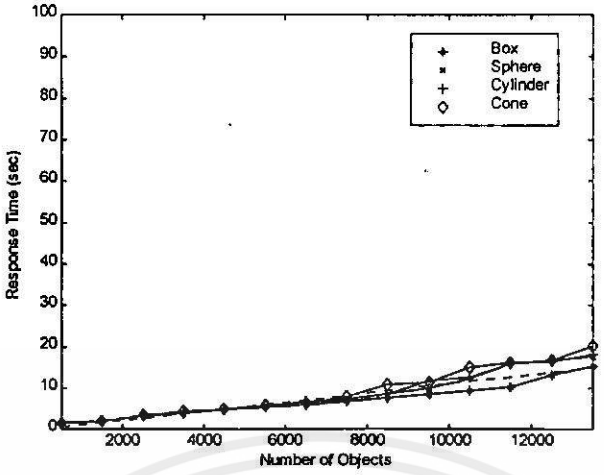
จากผลการทดลองที่เกิดขึ้นจากผลการทดลองตารางที่ 5.6 สามารถหาความสัมพันธ์ของสมการเชิงเส้นตรงและสมการโพลิโนเมียลโดยเลือกสมการที่มีค่าความผิดพลาดน้อยที่สุด เมื่อเปรียบเทียบผลจากการทดลองได้ดังนี้

$T_{Navigate}$ by Object Types	$T_{Navigate} \approx 0.001 \times N_{Obj}^{1.0077}$ (ดังรูปที่ 5.15ก1)	0.9442	1.5446
	$T_{Navigate} \approx (0.0012 \times N_{Obj}) + 0.1188$ (ดังรูปที่ 5.15ก2)	0.8245	0.8586
$T_{Navigate}$ by Object Types & Add Shading	$T_{Navigate} \approx 0.0015 \times N_{Obj\_Shading}^{1.0077}$ (ดังรูปที่ 5.15ข1)	1.0136	1.2382
	$T_{Navigate} \approx (0.0017 \times N_{Obj\_Shading}) + 0.4709$ (ดังรูปที่ 5.15ข2)	0.9681	0.9710
$T_{Navigate}$ by Object Types & Add Texture	$T_{Navigate} \approx 0.01 \times N_{Obj\_Texture}^{1.0078}$ (ดังรูปที่ 5.15ค1)	2.8153	3.9573
	$T_{Navigate} \approx (0.0114 \times N_{Obj\_Texture}) + 0.1870$ (ดังรูปที่ 5.15ค2)	1.3979	2.0377

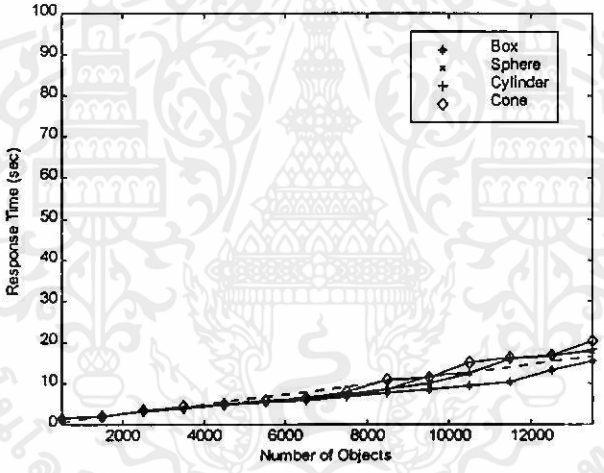
โดยที่ N_{Obj} = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone)

$N_{Obj\_Texture}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture

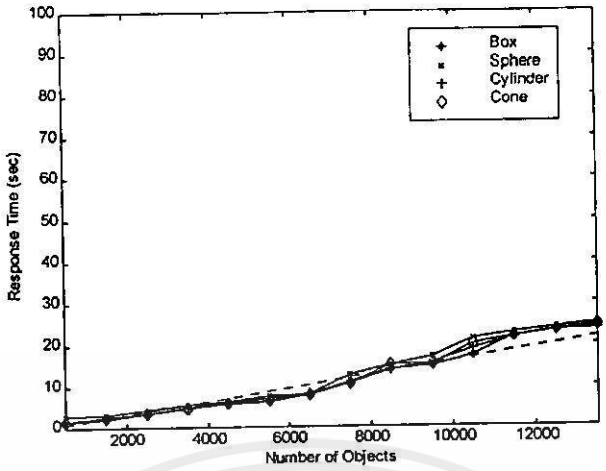
$N_{Obj\_Shading}$ = จำนวนออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) โดยเพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Shading



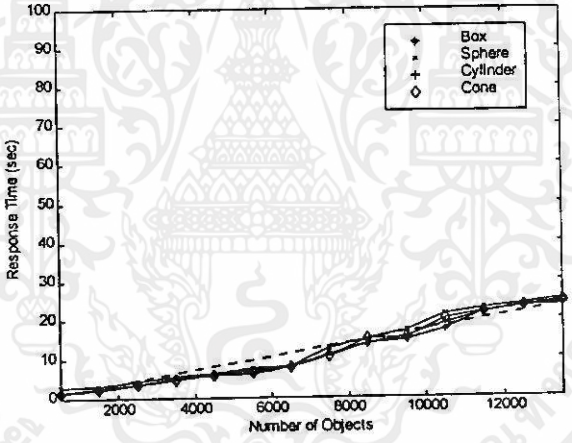
(ก1) แสดงผลการหาสมการเชิงโพลิโนเมียลจากการวัดเวลา $T_{Navigate}$ by Object Types



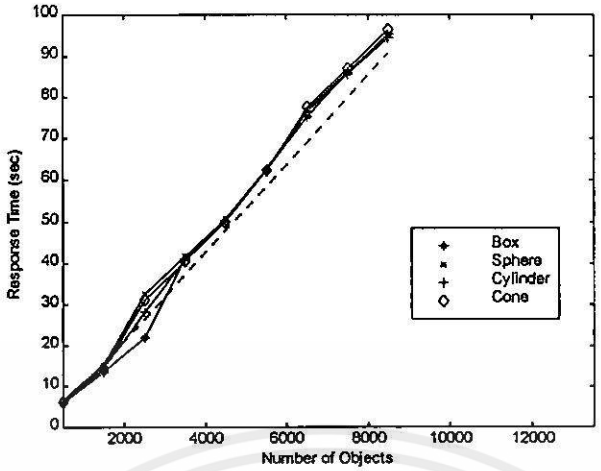
(ก2) แสดงผลการหาสมการเชิงเส้นตรงจากการวัดเวลา $T_{Navigate}$ by Object Types



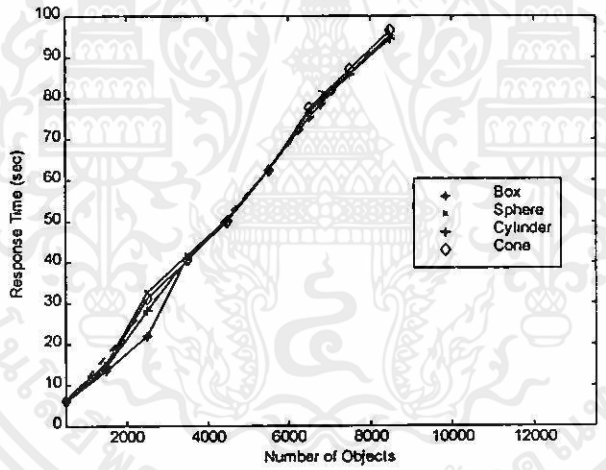
(ข1) แสดงผลการหาสมการเชิงโพลิโนเมียลจากการวัดเวลา $T_{Navigate}$ by Object Types & Add Shading



(ข2) แสดงผลการหาสมการเชิงเส้นตรงจากการวัดเวลา $T_{Navigate}$ by Object Types & Add Shading



(ค1) แสดงผลการหาสมการเชิงโพลีโนเมียลจากการวัดเวลา $T_{Navigate}$ by Object Types & Add Texture



(ค2) แสดงผลการหาสมการเชิงเส้นตรงจากการวัดเวลา $T_{Navigate}$ by Object Types & Add Texture

รูปที่ 5.15 (ก1-2,ข1-2,ค1-2) แสดงผลการหาสมการจากการวัดเวลา $T_{Navigate}$

ดังนั้นจากสมการ (4.27) สามารถประมาณค่าของเวลา $T_{Navigate}$ ได้ว่า

$$T_{Navigate} \approx \left[(0.0012+0.0017+0.0114)/3\right] \times N_{AVG\_Obj} + (0.1188+0.4709+0.1870)/3$$

นั่นคือ $T_{Navigate}$ เท่ากับ

$$T_{Navigate} \approx \left(0.00476 \times N_{AVG\_Obj}\right) + 0.2589 \quad \text{.....(5.5)}$$

สามารถเขียนในรูปทั่วไปได้ว่า

$$T_{Navigate} \approx \left(C_{Nav\_AVG\_Obj} \times N_{AVG\_Obj}\right) + T_{Nav\_Proc} \quad \text{.....(5.6)}$$

จากสมการ (5.5) ค่า

$C_{Nav\_AVG\_Obj}$ หมายถึง เวลาในการประมวลผลโดยเฉลี่ยของออบเจ็กต์ VRML ชนิดต่างๆ และคุณสมบัติเพิ่มเติมที่ใส่ให้กับออบเจ็กต์ VRML ในช่วง Navigate

N_{AVG_Obj} หมายถึง จำนวนเฉลี่ยของออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) ที่ได้เพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture และ Shading

T_{Nav_Proc} หมายถึง เวลาในการประมวลผลและส่งถ่ายข้อมูล

5.3 เปรียบเทียบผลการวิเคราะห์เชิงทฤษฎีกับผลการทดลอง

จากความสัมพันธ์ของสมการจากการวิเคราะห์หลังการทดลองดังกล่าวข้างต้น สามารถสรุปได้ดังนี้

เวลาทั้งหมดที่ใช้ในการประมวลผล คือ

$$T_{Load} = T_{WB\_to\_WS} + T_{WS\_to\_CGI} + T_{CGI\_to\_DBMS} + T_{CGI\_to\_WS} + T_{WB\_to\_WB} + T_{WB\_to\_VRMLBrowser} \quad \text{.....(4.24)}$$

โดยที่

$$T_{WB\_to\_WS} \approx 0.04389 \text{ second} \quad \text{.....(4.3)}$$

$$T_{CGI\_to\_DBMS} = 0.0000328583 \times N_{AVG\_Obj} \quad \text{.....(5.2)}$$

$$T_{WS\_to\_WB} = 0.0096 \times N_{AVG\_Obj} - 0.0139 \quad \text{.....(5.3)}$$

$$T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx 0.000672683 \times N_{AVG\_Obj}^{1.0077} \quad \text{.....(5.4)}$$

แทนค่าสมการ (4.24) ด้วยสมการ (4.3) , (5.2) , (5.3) และ (5.4) จะได้

$$T_{Load} \approx \left(0.0103055413 \times N_{AVG\_Obj}\right) + 0.02999 \quad \text{.....(5.7)}$$

สามารถเขียนในรูปทั่วไปได้ว่า

$$T_{Load} \approx \left(T_{Load\_AVG\_Obj} \times N_{AVG\_Obj}\right) + T_{Load\_Proc} \quad \text{.....(5.8)}$$

โดย T_{Load_Proc} = เวลาในการประมวลผลและส่งถ่ายข้อมูลภายในระบบ ประมาณเท่ากับ

$$\approx T_{WB\_Proc\_1} + \frac{S_{URL}}{R_{HTTPRate}} + f_{WS}\left(E_{WS}\right) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) + T_{WS\_Proc\_2}$$

$T_{Load\_AVG\_Obj}$ = เวลาในการประมวลผลโดยเฉลี่ยของ VRML หนึ่งออบเจ็กต์ ประมาณเท่ากับ

$$\approx \frac{S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{CGI\_WS\_Rate}} + \frac{S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{HTTPRate}}$$

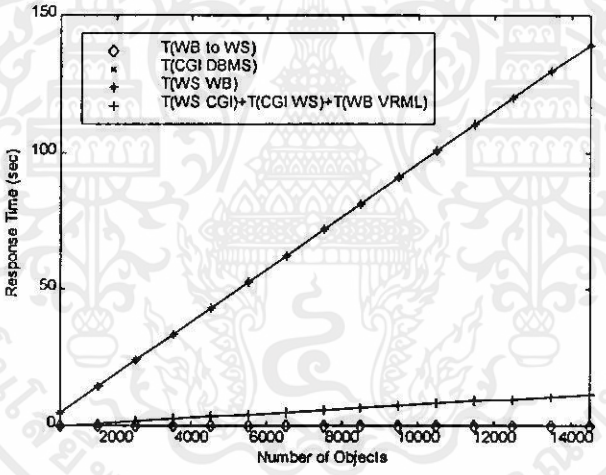
N_{AVG_Obj} = จำนวนเฉลี่ยของออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) ที่ได้เพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture และ Shading

จากสมการ (5.7) เวลาในแต่ละช่วงของ T_{Load} จะพบว่าเวลาที่ต้องใช้มากที่สุดเรียงลำดับได้ดังนี้ (ดังแสดงในรูปที่ 5.16)

- $T_{WS\_to\_WB} \approx 92.66\%$
- $T_{WS\_to\_CGI} + T_{CGI\_to\_WS} + T_{WB\_to\_VRMLBrowser} \approx 6.99\%$
- $T_{CGI\_to\_DBMS} \approx 0.32\%$
- $T_{WB\_to\_WS} \approx 0.03\%$

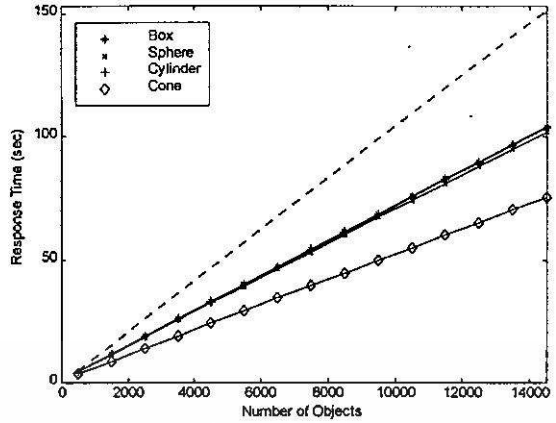
อันเนื่องมาจาก

- อัตราเร็วของเครือข่าย, ความหนาแน่นของเครือข่าย ณ เวลานั้น
- ประสิทธิภาพของเว็บเซิร์ฟเวอร์และการทำงานร่วมกับ CGI
- ประสิทธิภาพของฐานข้อมูล

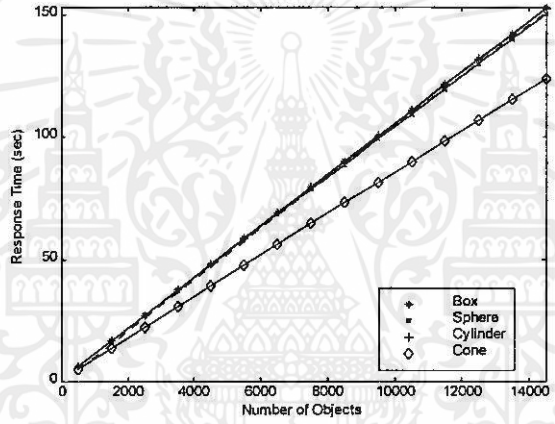


รูปที่ 5.16 แสดงช่วงเวลาต่างๆ ของ T_{Load}

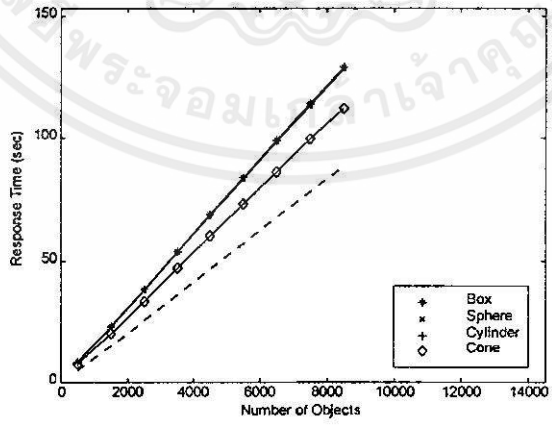
และจากสมการ (5.7) เวลา T_{Load} ที่ได้จากการวิเคราะห์นำมาเปรียบเทียบกับผลการทดลองดังแสดงในรูปที่ 5.17 จะพบว่าเวลาโดยเฉลี่ยแล้วใกล้เคียงกับเวลาที่ออบเจ็กต์แต่ละแบบมีการเพิ่ม Shading อันเป็นค่าระหว่างออบเจ็กต์แต่ละแบบที่ไม่มีการเพิ่มคุณลักษณะใดๆ กับการเพิ่ม Texture



(ก) แสดงสมการที่ได้จากการวิเคราะห์เปรียบเทียบกับผลการทดลองจากการวัดเวลา T_{Load} by Object Types



(ข) แสดงสมการที่ได้จากการวิเคราะห์เปรียบเทียบกับผลการทดลองจากการวัดเวลา T_{Load} by Object Types & Add Shading



(ค) แสดงสมการที่ได้จากการวิเคราะห์เปรียบเทียบกับผลการทดลองจากการวัดเวลา T_{Load} by Object Types & Add Texture

รูปที่ 5.17 แสดงสมการที่ได้จากการวิเคราะห์เปรียบเทียบกับผลการทดลองจากการวัดเวลา T_{Load}

จากสมการ (5.7) เวลา T_{Load} ที่เป็นเวลาที่ผู้ใช้สามารถยอมรับได้ในการรอผลลัพธ์แรกกลับมาจากเว็บเซิร์ฟเวอร์คืออยู่ระหว่าง 2-4 วินาที [6]

ดังนั้นจำนวนของ N_{AVG_Obj} สำหรับการรอผลลัพธ์แรกจากเว็บเซิร์ฟเวอร์ มีค่าอย่างต่ำประมาณ 385 ออบเจ็กต์

$$T_{Load} \approx 0.0103055413 \times N_{AVG\_Obj} + 0.02999$$
$$0.0103055413 \times N_{AVG\_Obj} + 0.02999 \leq 4$$
$$N_{AVG\_Obj} \leq 385$$

5.4 การวิเคราะห์ชนิดของออบเจ็กต์ในระบบ

เนื่องจากหลักการนำเสนอข้อมูลของโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML ในแต่ละหน้าจอการทำงานของ VRML Browser (View) จะประกอบด้วยออบเจ็กต์ต่างๆ ที่เป็นโครงสร้างลำดับชั้น (Parent-Child Relationship) ดังรูปที่ 3.11 ในบทที่ 3 โดยในแต่ละออบเจ็กต์จะมี Viewpoint อันเป็นตัวกำหนดมุมมองในแต่ละ View และในแต่ละออบเจ็กต์นี้เองที่ผู้ใช้สามารถระบุตามความต้องการลงในฐานข้อมูลได้ว่าต้องการดูข้อมูลของออบเจ็กต์ในแต่ละ View ลึกลงไปกี่ระดับ ถ้าผู้ใช้กำหนดให้โปรแกรมแสดงหลายระดับในคราวเดียว เวลาที่ใช้ในการประมวลผลจะมากขึ้น ซึ่งมากกว่าการแสดงทีละระดับเนื่องจากโปรแกรมจะสามารถทยอยประมวลผลเพื่อส่งให้กับผู้ใช้ได้ ดังนั้นจึงขึ้นกับวิธีการเขียน CGI Script และการเก็บข้อมูลในฐานข้อมูลเพื่อนำมาสร้างเป็น VRML Script ด้วย (ดังรายละเอียดในหัวข้อ 3.6) และจากรูปที่ 3.11 ในสภาพแวดล้อมการจัดการระบบเครือข่ายในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถแบ่งแยกชนิดของออบเจ็กต์เป็นประเภทใหญ่ๆ ได้ 2 ประเภท คือ ออบเจ็กต์ที่เป็นออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายที่ใช้ในการควบคุมเครือข่าย เช่น เครื่องคอมพิวเตอร์, เราท์เตอร์, สวิทช์, เครื่องพรินเตอร์ เป็นต้น และสำหรับออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่าย เช่น ตึก, ห้อง, โต๊ะ, เก้าอี้ เป็นต้น

ดังนั้นจะให้ N_{Equip} แทน จำนวนออบเจ็กต์ที่เป็นอุปกรณ์เครือข่าย และ

N_{Env} แทน จำนวนออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่าย

โดยความสัมพันธ์ระหว่าง N_{Equip} และ N_{Env} ขึ้นอยู่กับอัตราส่วนจำนวนออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายและออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายในแต่ละหน้าจอ (View) ของเว็บเบราเซอร์ เนื่องจากการนำเสนอข้อมูลของโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถเลือกที่จะแสดงว่าต้องการแสดงละเอียดไปกี่ระดับขึ้นอยู่กับผู้ใช้ร้องขอมาแต่ถ้าผู้ใช้กำหนดให้โปรแกรมแสดงหลายระดับในคราวเดียว เวลาที่ใช้ในการประมวลผลจะมากขึ้น ซึ่งมากกว่าการแสดงทีละระดับเนื่องจากโปรแกรมจะสามารถทยอยประมวลผลเพื่อส่งให้กับผู้ใช้ได้ ดังนั้นจึงขึ้นกับวิธีการเขียน CGI Script เพื่อสร้าง VRML Script ด้วย โดยอาจจะเป็น 1 หรือ 2 ระดับ ตามความเหมาะสมในการนำเสนอของผู้เขียน VRML Script จากบทพิสูจน์เวลาที่ใช้ในแต่ละช่วงจะ

เห็นว่าจำนวนออบเจ็กต์, ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML เพื่อให้ดูเหมือนจริง สิ่งเหล่านี้ล้วนมีผลกับเวลาที่ต้องใช้ในการประมวลผลทั้งสิ้น ดังนั้นขนาดของเครือข่ายที่เราต้องการเข้าไปจัดการเครือข่ายจะต้องคำนึงถึงสัดส่วนของออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายและออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายในแต่ละหน้าจอด้วย นั่นคือ ถ้าหากขนาดของเครือข่ายที่เราต้องการเข้าไปจัดการในแต่ละหน้าจอ (View) มีขนาดใหญ่อาจจะต้องยอมแลกกับเวลาที่ต้องใช้ในการประมวลผลเพื่อให้ได้มาซึ่งเวลาตอบสนองที่ผู้ใช้สามารถยอมรับได้กับความเหมือนจริง แต่ถ้าผู้ใช้ต้องการเน้นเพียงแค่อุปกรณ์ในเครือข่ายเท่านั้น อาจจะตัดออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายออกก็ได้เพื่อแลกกับความเร็วที่ผู้ใช้ไม่ต้องรอเวลาในการประมวลผลนานกับความเหมือนจริงที่ลดลง เป็นต้น และจากผลการทดลองในตารางที่ 5.7 เป็นการสร้างแบบจำลองการทำงานที่ค่อนข้างใกล้เคียงความเป็นจริงอันเป็นผลการทดลองเพื่อยืนยันว่าเมื่อขนาดของเครือข่ายมีขนาดใหญ่หรือมีจำนวนออบเจ็กต์ใน 1 view มากย่อมใช้เวลาในการประมวลผลมากกว่าเครือข่ายขนาดเล็กหรือประกอบด้วยจำนวนออบเจ็กต์ที่น้อยกว่า ดังนั้นผู้ใช้อาจต้องลดจำนวนออบเจ็กต์ลงเพื่อแลกกับเวลาที่ต้องใช้ในการประมวลผลให้น้อยลงและความเหมือนจริงที่ลดลงด้วย โดยรายละเอียดส่วนประกอบของอุปกรณ์เครือข่ายที่ใช้ในการทดลองแสดงในตารางที่ 5.10

ตารางที่ 5.7 แสดงผลการทดลองเวลาที่ใช้ในการประมวลผลระหว่างขนาดของ N_{Equip} และ N_{Env}

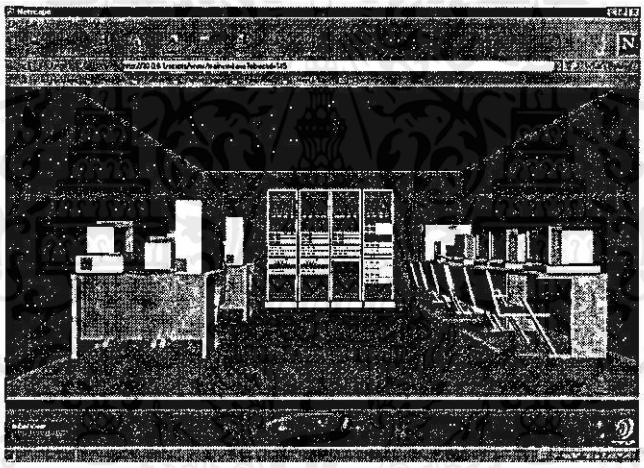
	$N_{Equip} + N_{Env}$ (รูปที่ 5.18)	N_{Env} (ตารางที่ 5.8)	N_{Equip} (รูปที่ 5.19 , ตารางที่ 5.9)
Number of Boxes	1,422	140	1,282
Number of Textures	1,099	5	1,094
Number of Shadings	203	45	158

ตารางที่ 5.8 แสดงรายชื่อของออบเจ็กต์ใน N_{Env}

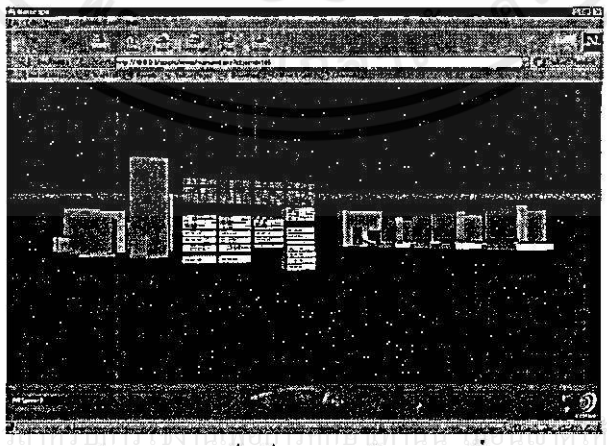
DESK	5
RACK	4
ROOM_PARTITION	1

ตารางที่ 5.9 แสดงรายชื่อของออบเจ็กต์ใน N_{Equip}

ออบเจ็กต์	จำนวน
HOST	1
HUB	16
PATCH	32
PRINTER	1
ROUTER	6
SERVER	3
SWITCH	14
WORKSTATION	4



รูปที่ 5.18 แสดงภาพการทดลองจำนวนออบเจ็กต์ $N_{Equip} + N_{Env}$ ในตารางที่ 5.7



รูปที่ 5.19 แสดงภาพการทดลองจำนวนออบเจ็กต์ N_{Equip} ในตารางที่ 5.7

ตารางที่ 5.10 แสดงรายละเอียดส่วนประกอบของออบเจ็กต์อุปกรณ์เครือข่ายที่ใช้ในระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML

		1		1
		12	12	
		1		1
		5	5	
		1		1
		24	24	
		1		1
		16	16	
		8		2
		24	1	5
		4		4
		4	4	
		18		15
		11		8
		1		
		14		11

จากผลการทดลองสามารถนำมาหาอัตราส่วน N_{Equip} และ N_{Env} ได้โดยอิงกับแบบจำลองที่ใช้ในการทดลองครั้งนี้ โดย

$$\text{Equipment Ratio} \approx \frac{N_{Equip}}{N_{Equip} + N_{Env}}$$

$$\approx \frac{1}{1.07} \approx 91.30\%$$

$$\text{Environment Ratio} \approx \frac{N_{Env}}{N_{Equip} + N_{Env}}$$

$$\approx \frac{1}{14.34} \approx 8.69\%$$

และเนื่องจากในสภาพแวดล้อมการจัดการระบบเครือข่ายผ่านเว็บด้วย VRML จะมีข้อจำกัดในด้านของเวลาที่ต้องใช้ในการประมวลผลกราฟฟิก 3 มิติ ดังนั้นถ้าผู้ใช้ต้องการจัดการกับระบบเครือข่ายที่มีจำนวนออบเจ็กต์ในเครือข่ายจำนวนมากแล้ว อาจต้องใช้เวลาในการประมวลผลสูง นอกจากนั้นคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML ไม่ว่าจะเป็นการเพิ่ม Shading หรือ Texture จะพบว่าเวลาในการประมวลผลของอยบเจ็กต์ที่เพิ่ม Texture จะใช้เวลาในการประมวลสูงกว่า เนื่องจากปลั๊กอิน VRML Browser จะต้องมีการไปเรียก url ของไฟล์รูปภาพเพื่อใส่ Texture แต่การประมวลผลของ Shading ไม่ต้องมีการเรียกไฟล์จึงไม่เสียเวลาที่ใช้ไปของ Input/Output Time ในการติดต่อกับอุปกรณ์เพราะฉะนั้นผู้ใช้อาจจะต้องลดความเหมือนจริงของสภาพแวดล้อมลง เพื่อแลกกับเวลาในการประมวลผลที่น้อยลง

5.5 สรุป

เนื่องจากการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML เป็นการนำเสนอข้อมูลในรูปแบบ 3 มิติ ซึ่งส่วนนี้เป็นปัจจัยหนึ่งที่จะมีผลต่อประสิทธิภาพการทำงานของระบบจัดการเครือข่าย จากปัจจัยดังกล่าว ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML อาจจะมีข้อจำกัดอยู่ที่การใช้งานกับระบบเครือข่ายขนาดใหญ่ และเพื่อให้โปรแกรมการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถจัดการกับระบบเครือข่ายขนาดใหญ่ได้ ผู้ใช้อาจจะลดจำนวนออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายลง อย่างเช่น เก้าอี้, ชั้นวางหนังสือ เป็นต้น

นอกจากนั้นผลการวิเคราะห์และผลการทดลองที่ได้ในบทนี้อิงกับระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML เท่านั้น ดังนั้นการที่จะนำไปประยุกต์ใช้กับระบบงานอื่นๆ อาจจะได้ผลที่ไม่ตรงกัน เพราะฉะนั้นจะต้องทำการตรวจสอบปัจจัยอื่นๆ ที่อาจจะมีผลกับการนำไปประยุกต์ใช้กับระบบงานอื่นๆ ด้วย

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

6.1 สรุปผลการวิจัยของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

ในงานวิจัยนี้มีโครงสร้างพื้นฐานเป็นเวิล์ดไวด์เว็บ โดยผู้ใช้งานจะทำการร้องขอการทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยการระบุ URL ไปยังเว็บเซิร์ฟเวอร์ ต่อจากนั้นจะเข้าสู่การทำงานของ CGI นั่นคือ เมื่อเว็บเซิร์ฟเวอร์รับการร้องขอดังกล่าวไปจะทำการประมวลผลสคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซิร์ฟเวอร์แอปพลิเคชั่น (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผลและจะทำการดึงข้อมูลจากฐานข้อมูลเพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกประมวลผลจะส่งผลลัพธ์กลับไปให้เว็บเซิร์ฟเวอร์ แล้วเว็บเซิร์ฟเวอร์จะส่งผลลัพธ์นั้นกลับไปให้กับเว็บเบราเซอร์ที่ส่งการร้องขอมาเพื่อทำการแสดงผลให้กับผู้ใช้ แต่เนื่องจากหลักการนำเสนอข้อมูลของโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML ในแต่ละหน้าจอการทำงานของ VRML Browser (View) จะประกอบด้วยออบเจ็กต์ต่างๆ ที่เป็นโครงสร้างลำดับชั้น (Parent-Child Relationship) โดยในแต่ละออบเจ็กต์จะมี Viewpoint อันเป็นตัวกำหนดมุมมองในแต่ละ View และในแต่ละออบเจ็กต์นี้เองที่ผู้ใช้สามารถระบุตามความต้องการลงในฐานข้อมูลได้ว่าต้องการดูข้อมูลของออบเจ็กต์ในแต่ละ View ลึกลงไปกี่ระดับ ถ้าผู้ใช้กำหนดให้โปรแกรมแสดงหลายระดับในคราวเดียว เวลาที่ใช้ในการประมวลผลจะมากขึ้น ซึ่งมากกว่าการแสดงทีละระดับเนื่องจากโปรแกรมจะสามารถทยอยประมวลผลเพื่อส่งให้กับผู้ใช้ได้ ดังนั้นจึงขึ้นกับวิธีการเขียน CGI Script และการเก็บข้อมูลในฐานข้อมูลเพื่อนำมาสร้างเป็น VRML Script ด้วย โดยปัจจัยของเวลาในการประมวลผลในระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถสรุปออกเป็น 2 ส่วน (ดูรายละเอียดได้จากบทที่ 5) ได้ดังนี้

$$T_{Load} \approx \left(T_{Load\_AVG\_Obj} \times N_{AVG\_Obj}\right) + T_{Load\_Proc}$$

$$T_{Navigate} \approx \left(C_{Nav\_AVG\_Obj} \times N_{AVG\_Obj}\right) + T_{Nav\_Proc}$$

โดย T_{Load_Proc} หมายถึง เวลาในการประมวลผลและส่งถ่ายข้อมูลภายในระบบ ประมาณเท่ากับ

$$\approx T_{WB\_Proc\_1} + \frac{S_{URL}}{R_{HTTPRate}} + f_{WS}\left(E_{WS}\right) + \frac{S_{Para}}{R_{WS\_CGI\_Rate}} + f_{DBMS}\left(N_{Obj}, E_{DBMS}\right) + T_{WS\_Proc\_2}$$

$T_{Load\_AVG\_Obj}$ หมายถึง เวลาในการประมวลผลโดยเฉลี่ยของ VRML หนึ่งออบเจ็กต์ ประมาณเท่ากับ

$$\approx \frac{S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{CGI\_WS\_Rate}} + \frac{S_{AVG\_Obj} \times N_{AVG\_Obj}}{R_{HTTPRate}}$$

$C_{Nav\_AVG\_Obj}$ หมายถึง เวลาในการประมวลผลโดยเฉลี่ยของออบเจ็กต์ VRML ชนิดต่างๆ และคุณสมบัติเพิ่มเติมที่ใส่ให้กับออบเจ็กต์ VRML ในช่วง Navigate

N_{AVG_Obj} หมายถึง จำนวนเฉลี่ยของออบเจ็กต์ชนิด Box (Sphere, Cylinder, Cone) ที่ได้เพิ่มเติมคุณลักษณะให้กับ VRML ออบเจ็กต์เพื่อให้ดูเหมือนจริงมากขึ้น โดยการใส่ Texture และ Shading

T_{Nav_Proc} หมายถึง เวลาในการประมวลผลและส่งถ่ายข้อมูล

ดังนั้นถ้าต้องการทราบถึงเวลาการประมวลผลโดยประมาณของ T_{Load} และ $T_{Navigate}$ สามารถแทนค่าตัวแปรตามสมการข้างต้นได้ แต่ทั้งนี้ต้องขึ้นอยู่กับสภาพแวดล้อมของระบบ, ประสิทธิภาพของเครื่องไคลเอนต์, เว็บเซิร์ฟเวอร์, ระบบจัดการฐานข้อมูลและปลั๊กอิน VRML Browser รวมถึงลักษณะของเครือข่ายที่ทำการรับส่งข้อมูล ณ เวลานั้นระหว่างไคลเอนต์และเซิร์ฟเวอร์ด้วย และเนื่องด้วยในสภาพแวดล้อมการจัดการระบบเครือข่ายในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถแบ่งแยกชนิดของออบเจ็กต์เป็นประเภทใหญ่ๆ ได้ 2 ประเภท คือ ออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายที่ใช้ในการควบคุมเครือข่าย เช่น เครื่องคอมพิวเตอร์, เราท์เตอร์, สวิทช์, เครื่องพรินเตอร์ เป็นต้น และสำหรับออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่าย เช่น ตึก, ห้อง, โต๊ะ, เก้าอี้ เป็นต้น

ดังนั้นจะให้ N_{Equip} แทน จำนวนออบเจ็กต์ที่เป็นอุปกรณ์เครือข่าย และ

N_{Env} แทน จำนวนออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่าย

โดยความสัมพันธ์ระหว่าง N_{Equip} และ N_{Env} ขึ้นอยู่กับอัตราส่วนจำนวนออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายและออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายในแต่ละหน้าจอ (View) ของเว็บเบราเซอร์ เนื่องจากการนำเสนอข้อมูลของโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML

สามารถเลือกที่จะแสดงว่าต้องการแสดงละเอียดไปกี่ระดับขึ้นอยู่กับผู้ใช้ร้องขอมาแต่ถ้าผู้ใช้กำหนดให้โปรแกรมแสดงหลายระดับในคราวเดียว เวลาที่ใช้ในการประมวลผลจะมากขึ้น ซึ่งมากกว่าการแสดงทีละระดับเนื่องจากโปรแกรมจะสามารถทยอยประมวลผลเพื่อส่งให้กับผู้ใช้ได้ ดังนั้นจึงขึ้นกับวิธีการเขียน CGI Script เพื่อสร้าง VRML Script ด้วย โดยอาจจะเป็น 1 หรือ 2 ระดับ ตามความเหมาะสม จากบทพิสูจน์เวลาที่ใช้ในแต่ละช่วงจะเห็นว่าจำนวนออบเจ็กต์, ชนิดของออบเจ็กต์และคุณสมบัติที่ใส่เพิ่มเติมให้กับออบเจ็กต์ VRML เพื่อให้ดูเหมือนจริง สิ่งเหล่านี้ล้วนมีผลกับเวลาที่ต้องใช้ในการประมวลผลทั้งสิ้น ดังนั้นขนาดของเครือข่ายที่เราต้องการเข้าไปจัดการเครือข่ายจะต้องคำนึงถึงสัดส่วนของออบเจ็กต์ที่เป็นอุปกรณ์เครือข่ายและออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายในแต่ละหน้าจอด้วย นั่นคือ ถ้าหากขนาดของเครือข่ายที่เราต้องการเข้าไปจัดการในแต่ละหน้าจอ (View) มีขนาดใหญ่อาจจะต้องยอมแลกกับเวลาที่ต้องใช้ในการประมวลผลเพื่อให้ได้มาซึ่งเวลาตอบสนองที่ผู้ใช้สามารถยอมรับได้กับความเหมือนจริง แต่ถ้าผู้ใช้ต้องการเน้นเพียงแค่อุปกรณ์ในเครือข่ายเท่านั้น อาจจะตัดออบเจ็กต์ที่เป็นสภาพแวดล้อมของเครือข่ายออกก็ได้เพื่อแลกกับความเร็วที่ผู้ใช้ไม่ต้องรอเวลาในการประมวลผลนานกับความเหมือนจริงที่ลดลง เป็นต้น

และจากการพัฒนาระบบทั้งหมดที่ได้กล่าวมาทำให้การจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถเข้าไปจัดการ, ตรวจสอบและดูแลระบบเครือข่ายคอมพิวเตอร์ผ่านทางอินเตอร์เน็ตโดยใช้เว็บเบราเซอร์ในรูปแบบที่เสมือนจริงและอิงกับลักษณะทางกายภาพและจะเป็นตัวต้นแบบที่จะสามารถถูกนำไปพัฒนาให้มีความสามารถและความสมบูรณ์เพิ่มมากขึ้น เพื่อรองรับการจัดการระบบเครือข่ายที่มีความซับซ้อนมากขึ้นต่อไปได้

แต่เนื่องจากข้อมูลที่จัดเก็บที่นำมาใช้ในการทดลองครั้งนี้ยังมีความซับซ้อนและรายละเอียดไม่มากนัก ซึ่งผู้ใช้สามารถที่จะจัดเก็บข้อมูลดังกล่าวได้มากกว่านี้ โดยสามารถที่จะเพิ่มระดับความสัมพันธ์ของวัตถุต่างๆ ได้มากเท่าที่ต้องการ เพื่อให้เห็นภาพที่ละเอียดและมีความเหมือนจริงมากยิ่งขึ้น แต่ทั้งนี้ยิ่งข้อมูลที่จัดเก็บมีความซับซ้อนและมีรายละเอียดมากเท่าไร เวลาที่จะถูกใช้ในการประมวลผลและการแสดงผลของระบบก็ยิ่งจะต้องมากขึ้นเช่นกัน แต่อย่างไรก็ตามยังมีอีกหลายๆ ปัจจัยที่มีผลต่อการทำงานของระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML อย่างเช่น [1]

- ความเหมือนจริงในการนำเสนอออบเจ็กต์ต่างๆ นั้น ขึ้นอยู่กับโปรแกรมว่าสามารถกำหนดเงื่อนไขในการนำเสนอได้มากน้อยเพียงใด ดังนั้นถ้ามีการเก็บข้อมูลทางฟิสิกคอลและลอจิกคอลลงในฐานข้อมูลละเอียดมากเท่าไร ความเหมือนจริงก็จะมีมากขึ้นแต่เวลาที่ใช้ในการนำเสนอก็จะช้าลงไปด้วย
- ความเร็วและความต่อเนื่องในการท่องไปในแต่ละฉาก จะพบว่า การที่มีวัตถุมากเกินไปหรือมีวัตถุที่มีความซับซ้อนมากๆ จะทำให้ความเร็วในการท่องลดลง แต่เมื่อทำการลดจำนวนวัตถุที่ไม่จำเป็นออกหรือดัดแปลงวัตถุให้มีความซับซ้อนน้อยลงจะเป็นการแก้ไขให้ดีขึ้นได้

- เครื่องไคลเอนต์ที่จะต้องแปลข้อความ VRML เท็กไฟล์นั้นให้อยู่ในรูป 3 มิติ จึงต้องทำงานบนเครื่องมีประสิทธิภาพสูงๆ
- เวลาที่ใช้ในการนำเสนอออบเจ็กต์ต่างๆ ในรูปแบบ 3 มิตินั้น ขึ้นอยู่กับอัตราส่วนที่ใช้ในการแม็พข้อมูลฟิสิกคอลและข้อมูลลอจิกคอลไปเป็นการนำเสนอแบบกราฟฟิก 3 มิติ ด้วยว่าใช้อัตราส่วนเท่าไร ยิ่งใช้อัตราส่วนในการนำเสนอมากเท่าไร เวลาที่ใช้ในการแสดงรูป 3 มิติก็จะมากขึ้นตามไปด้วย
- เมื่อประสิทธิภาพของฮาร์ดแวร์และการประมวลผลทางกราฟฟิก 3 มิติ มีประสิทธิภาพมากขึ้น จะทำให้สามารถเพิ่มเติมองค์ประกอบต่างๆ เข้าไปในระบบได้อย่างเต็มที่และยังทำให้มีความเสมือนจริงมากขึ้น

6.2 ข้อเสนอแนะของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

เนื่องด้วยโปรแกรมระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ได้ถูกออกแบบมาให้มีความยืดหยุ่นพอสมควร ดังนั้นในกรณีที่ต้องการนำโปรแกรมระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML นำไปพัฒนาต่อก็สามารถทำได้ไม่ยากนัก ตัวอย่างเช่น

- ในอนาคตถ้าต้องการเพิ่มเติมอุปกรณ์ในเครือข่ายในรูปแบบ 3 มิติที่ไม่มีอยู่ในฐานข้อมูล ก็สามารถทำได้โดยเพิ่มเติมคุณลักษณะทั้งทางด้านฟิสิกคอลและลอจิคอลลงในฐานข้อมูล Object ตามความสัมพันธ์ของโครงสร้างเชิงสัมพันธ์ของฐานข้อมูลที่ได้กล่าวในบทที่ 3 ถัดจากนั้นจึงทำการพัฒนา VRML เทมเพล็ตของออบเจ็กต์นั้นเพื่อทำเป็นเทมเพล็ต เก็บไว้ใน Repository ที่เซิร์ฟเวอร์และทำการปรับปรุง CGI Script ที่มีอยู่ให้สนับสนุนการทำงานร่วมกับออบเจ็กต์ใหม่ที่ถูกสร้างขึ้นมาต่อไป
- ในอนาคตหากมีฟังก์ชันที่ต้องการเพิ่มเติมลงในระบบโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML ตัวอย่างเช่น ถ้าต้องการรับเมสแสจจากอุปกรณ์เครือข่ายเข้าสู่ระบบโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML ก็สามารถทำได้เนื่องจากในระบบมีโปรแกรมที่ใช้ในการ Broadcast ข้อความออกไป ซึ่งในระบบมีฐานข้อมูลเก็บ IP Address ของไคลเอนต์อยู่แล้ว ดังนั้นหากต้องการแจ้งข่าวสารให้ไคลเอนต์ทราบ เมื่อเกิดเหตุการณ์อื่นๆ ก็ต้องพิจารณาว่า เมื่อเกิดเหตุการณ์ขึ้นแล้วจะสามารถเรียกโปรแกรมที่ทำหน้าที่ Broadcast ดังกล่าวให้ทำงานได้อย่างไร เช่น หากมีสัญญาณ trap เข้ามา ก็ต้องพิจารณาว่าโปรแกรมอะไรที่เป็นตัวรับสัญญาณตัวนั้น ซึ่งก็ต้องพิจารณาต่อว่าโปรแกรมตัวนั้นมีคุณลักษณะในการไปเรียกโปรแกรมตัวอื่นให้ทำงานได้หรือไม่ ก็จะสามารถเรียกโปรแกรม Broadcast ให้ทำงานได้แล้ว

เอกสารอ้างอิง

[1] Andrea L. Ames , David R. Nadeau , and John L. Moreland. **VRML 2.0 Sourcebook Second Edition.** Canada : John Wiley & Sons,Inc.. 1997.

[2] Bay Networks. "Web-Based Network Management: Linking to the Future of Network Management." [Online] Available : http://www.baynetworks.com/products/Papers/webbased.html. 1998.

[3] Daniel Donnelly. **WWW design : web pages from around the world.** MA : Rockport Publishing. 1997.

[4] Douglas Hyde. "Web-Based Management." [Online] Available : http://www.3com.com/nsc/500627.html. 1996.

[5] Gary Cornell and Cay S. Horstmann. **Core Java Second Edition.** California : Sun Microsystems, Inc.. 1997.

[6] Jeffrey C. Mogul. 1995. "The Case for Persistent-Connection HTTP." **ACM.** 1995 : 299-313.

[7] Jesffry Dwight and Michacel Erwin. **Special Edition Using CGI.** Indianapolis : Que Corporation. 1996.

[8] Laura Lemay and Michael G. Moncur. **JavaScript.** Indianapolis : Sun Microsystems,Inc.. 1996.

[9] Mark Brown , John Jung , and Tom Savola. **Special Edition Using HTML , Second Edition.** Indianapolis : Que Corporation. 1996.

[10] Michael Gering, John Potok, and Terry Smetanka. "IBM's Java Web-Based Network Management:Easy Manageability From Anywhere." [Online] Available : http://www.ibm.eom/Java/education/network-management.html. 1998.

[11] Ramez Elmasri and Shamkant B. Navathe. **Fundamentals of Database Systems.** California : The Benjamin/Cummings Publishing Company, Inc. 1989.

[12] Sharon Hanson. "Web-Based Enterprise Management." [Online] Available : http://www.us.dell.com/r&d/vectors/vect_2-2/v2-2wbem.htm. 1998.

[13] Shishir Gundavaram. **CGI Programming on the World Wide Web.** O'Reilly & Associates,Inc. 1996.

[14] Stephen Asbury , Jason Mathews , Selena Sol , and Kevin Greer. **CGI HOW-TO.** Corte Madera : The Waite Group,Inc.. 1996.

[15] Stephen N. Matsuba and Bernie Roehl. **Special edition Using VRML.** Indianapolis : Que Corporation. 1998.

[16] Tom Sheldon. **The Windows NT Web Server Handbook.** Osborne : McGraw-Hill. 1996.

[17] Victor Wolters. **Special Edition Using Microsoft Internet Information Server2.** Indianapolis : Que Corporation. 1996.

[18] William Stallings. **SNMP SNMPv2 and RMON.** Massachusetts : Addison-Wesley Publishing Company,Inc.. 1996.

[19] William Stallings. **Data and Computer Communications Fifth Edition.** New Jersey : Prentice-Hall, Inc. 1997.

[20] Xavier Pacheco and Steve Teixeria. **Delphi 4 Developer's Guide.** Indianapolis : Sams Publishing. 1998.

[21] ดร.วีระศักดิ์ ซึงถาวร. **Fundamental of Java Programming Volume 2.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น. 2543.

ภาคผนวก ก.
บทความที่ตีพิมพ์ในวารสาร

NECTEC Technical Journal

วารสารวิชาการเนคเทค ปีที่ 1 ฉบับที่ 4 เดือน กันยายน-ตุลาคม 2542 Vol. 1, No. 4, September-October 1999 ISSN 1513-2145

Full Paper

Short Paper

Letters

NECTEC Technical Journal
132 Vol 1, No. 4, September - October 1999

การออกแบบส่วนติดต่อผู้ใช้แบบ 3 มิติในระบบจัดการเครือข่ายโดยใช้ VRML

3-D Web-Based Network Management Interface Using VRML

แดง ชลไพรพิมลรัตน์, อัครินทร์ คุณกิดดิ
นักศึกษา, อาจารย์ คณะเทคโนโลยีสารสนเทศ
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง, กรุงเทพมหานคร 10520

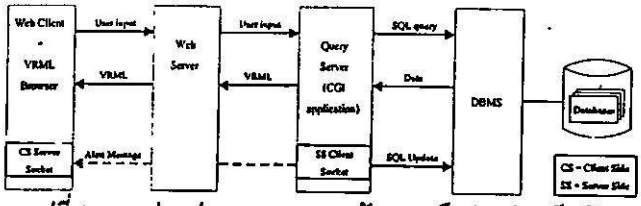
บทคัดย่อ --ในบทความนี้จะกล่าวถึงการพัฒนาและทดสอบโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML อันเป็นระบบจัดการเครือข่ายที่ผสมผสานความสามารถทางด้าน CGI กับ VRML เข้าด้วยกัน โดยจะเป็นการนำเสนอการจัดการเครือข่ายผ่านเว็บด้วยส่วนติดต่อผู้ใช้ที่เป็น 3 มิติ ซึ่งสามารถที่จะนำเสนอรายละเอียดของข้อมูลที่เป็นประโยชน์ในการจัดการเครือข่าย ในเรื่องของตำแหน่งที่ตั้งจริงทางกายภาพของอุปกรณ์ได้เป็นอย่างดี รวมถึงสามารถตรวจดูสถานะปัจจุบันของระบบเครือข่ายและสภาพแวดล้อมทางกายภาพได้ ซึ่งในบทความนี้จะกล่าวถึงที่มาของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML รายละเอียดของส่วนประกอบของระบบ หลักการทำงานของระบบ รวมถึงรายละเอียดของหลักการสร้างข้อมูลฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ และขั้นตอนในการพัฒนาระบบในส่วนต่างๆ พร้อมทั้งตัวอย่างผลการทดลองเบื้องต้นประกอบการวิจัย

ABSTRACT --This paper presents development and testing of the VRML in Web-Based Network Management application using CGI technology and VRML capabilities for developing tools, which has three-dimension interface. Moreover, it gives right information for network management in the term of locations of devices, network's current status and its physical environment. These will bring about an efficient network management. Parts of the paper mention source of The VRML in Web-Based Network Management, system components, working rules, system developing steps and an example experiment.

1. บทนำ

การจัดการเครือข่ายนับได้ว่าเป็นงานที่สำคัญอย่างยิ่ง ไม่ว่าจะเป็นการควบคุม วางแผนหรือการเฝ้าติดตามกิจกรรมที่เกิดขึ้นกับเครือข่าย แต่เนื่องด้วยอินเตอร์เฟซของระบบจัดการเครือข่ายยังคงเป็นลักษณะ 2 มิติ และไม่อิงกับตำแหน่งจริงทางกายภาพ [5],[6] ทำให้ยังไม่สามารถแก้ปัญหาของผู้บริหารเครือข่ายในเรื่องของการหาตำแหน่งที่ตั้งจริงๆ ของอุปกรณ์ ซึ่งปัญหาที่ตามมาก็คือ เมื่ออุปกรณ์เครือข่ายเกิดปัญหาขึ้นและมีความจำเป็นที่จะต้องเข้าไปทำการแก้ไข ณ จุดที่อุปกรณ์ถูกติดตั้งอยู่ การเข้าไปยังจุดที่เกิดปัญหาเองหรือการแจ้งให้ผู้รับผิดชอบที่อยู่ในพื้นที่เข้าไปทำงาน ณ จุดที่เกิดปัญหาสามารถทำได้ยาก เนื่องจากไม่ทราบตำแหน่งที่ตั้งจริงๆ ของอุปกรณ์ที่เกิดปัญหา ทำให้การแก้ปัญหาทำได้ล่าช้า ส่วนปัจจัยที่สำคัญอีกอย่างหนึ่งที่จะทำให้การจัดการเครือข่ายประสบความสำเร็จมากยิ่งขึ้นก็คือ ระบบจัดการเครือข่ายจะต้องสามารถจัดการเครือข่ายผ่านเว็บได้ ซึ่งสามารถช่วยแก้ปัญหาของผู้บริหารเครือข่ายในเรื่องของการเข้าถึงข้อมูลและจัดการเครือข่ายได้จากหลายๆ แห่ง

ในบทความนี้จะนำเสนอผลการพัฒนาและทดสอบโปรแกรมการจัดการเครือข่ายผ่านเว็บด้วย VRML (Virtual Reality Modeling Language) โดยมุ่งเน้นไปในส่วนของการติดต่อผู้ใช้เพื่อแก้ไขปัญหาที่เกิดขึ้นดังกล่าว ทำให้ผู้บริหารระบบเครือข่ายสามารถที่จะจัดการและตรวจสอบการทำงานของระบบได้ด้วยอินเตอร์เฟซที่เสมือนจริงและอิงกับตำแหน่งทางกายภาพรวมถึงการระบุ



รูปที่ 1. แสดงส่วนประกอบของระบบจัดการเครือข่ายผ่านเว็บด้วย VRML

และตรวจดูสถานะปัจจุบันของระบบเครือข่าย รวมถึงสภาพแวดล้อมทางกายภาพ เพิ่มความสะดวกให้กับผู้บริหารเครือข่ายมากขึ้นทำให้สามารถแก้ปัญหาของระบบเครือข่ายได้อย่างทันท่วงที

โดยจะกล่าวถึงการนำ VRML เข้ามาประยุกต์ใช้ในการจัดการเครือข่าย เนื่องด้วยลักษณะการทำงานเป็นการจัดการเครือข่ายผ่านเว็บในรูปแบบ 3 มิติ ซึ่ง VRML เป็นภาษาที่ใช้ในการสร้างภาพ 3 มิติได้ดี นอกจากนี้ยังเป็นภาษาที่ง่ายต่อการเข้าใจและการอิมพลีเมนต์อีกด้วย เพียงแค่มีเว็บเบราเซอร์ก็สามารถดูโมเดล 3 มิติผ่านเว็บได้ และที่สำคัญสามารถนำ VRML มาพัฒนาร่วมกับ CGI (Common Gateway Interface) ได้ง่าย ต่อจากนั้นจะกล่าวถึงหลักการการทำงาน, ผลการทดลองและบทสรุปของระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML

2. การจัดการเครือข่ายผ่านเว็บด้วย VRML

2.1 ส่วนประกอบของระบบ

การพัฒนาระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วยVRML ประกอบด้วยส่วนต่างๆ ดังรูปที่ 1 โดยมีรายละเอียดดังต่อไปนี้

เว็บไคลเอนต์ทำหน้าที่ในการร้องขอบริการจากเว็บเซอร์ฟเวอร์และนำเสนอข้อมูลที่ได้รับกลับมา โดยในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML นี้ เว็บไคลเอนต์จะต้องสนับสนุนการเรียกดูข้อมูล VRML นอกจากนี้เว็บไคลเอนต์จะต้องสนับสนุนการทำงานของจาวาแอ็พเพลต (Java Applet) ด้วย เพราะจะต้องมีการโหลดแอ็พเพลตที่ทำหน้าที่เป็น Server Socket ไปไว้ที่เว็บไคลเอนต์ เพื่อทำหน้าที่ในการติดต่อสื่อสารกับ Client Socket ที่อยู่ฝั่งเซอร์ฟเวอร์เมื่อมีการเปลี่ยนแปลงข้อมูลในฐานข้อมูลฝั่งเซอร์ฟเวอร์ ซึ่งในส่วนของกระบวนการแจ้งการเปลี่ยนแปลงข้อมูลบนฝั่งเซอร์ฟเวอร์ไปให้ฝั่งไคลเอนต์ทราบนั้นจะกล่าวถึงรายละเอียดของการทำงานของกระบวนการนี้ในหัวข้อที่ 2.2 ต่อไป แต่กระบวนการทำงานหลักๆ โดยทั่วไประหว่างส่วนประกอบอื่นๆ จะมีลักษณะดังนี้คือ เว็บไคลเอนต์จะส่งการร้องขอด้วยการระบุ URL (Uniform Resource Locator) ไปยังเว็บเซอร์ฟเวอร์ ต่อจากนั้นเว็บเซอร์ฟเวอร์จะทำการเรียกใช้งานแอพพลิเคชันที่อยู่ฝั่งเว็บเซอร์ฟเวอร์ที่เรียกว่า CGI ให้ทำการประมวลผลและส่งผลลัพธ์กลับมาในรูปแบบของ VRML ต่อจากนั้นเว็บเซอร์ฟเวอร์จะทำการส่งข้อมูล VRML ที่ได้ดังกล่าวกลับไปให้เว็บไคลเอนต์ที่ทำการร้องขอมา นั่นคือ ในระบบนี้จะอาศัยหลักการทำงานของ CGI ในการเขียนโปรแกรมสคริปต์ สำหรับใช้ในการประมวลผลเพื่อให้ได้ผลลัพธ์ออกมาเป็น VRML โดยที่ CGI จะติดต่อกับดาต้าเบสเซอร์ฟเวอร์ที่มีหน้าที่จัดเก็บและควบคุมดูแลการเรียกใช้ข้อมูลให้เป็นไปอย่างถูกต้อง ซึ่งจะถูกเข้าถึงจากโปรแกรมสคริปต์ (CGI) ที่พัฒนาขึ้นมา

2.2 หลักการทำงานของระบบ

ในบทความนี้มีโครงสร้างพื้นฐานเป็นเวิร์ดไวด์เว็บ โดยผู้ใช้งานจะทำการรีเควสท์การทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยการระบุ URL ไปยังเว็บเซอร์ฟเวอร์ และหลักการทำงานของ VRML [1] จะต้องมี VRML Browser ซึ่งเป็นตัวแปลภาษา เมื่อระบุชื่อ URL ของ VRML เข้าไปยังเว็บเบราเซอร์แล้วพบว่าแฟ้มข้อมูลมี MIME-Type เป็น "model/vrml" ดังนั้นเว็บเบราเซอร์จะเรียกใช้โปรแกรม VRML Browser เพื่อทำการตีความ VRML ไปเป็นภาพแบบกราฟฟิก

ต่อจากนั้นจะเข้าสู่การทำงานของ CGI [2] นั่นคือ เมื่อเว็บเซอร์ฟเวอร์รับรีเควสท์ดังกล่าวไปจะทำการเอ็กซีคิวท์สคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซอร์ฟเวอร์แอพพลิเคชัน (CGI) พร้อมทั้งส่งผ่านพารา

Technical Journal
134 Vol 1, No. 4, September - October 1999

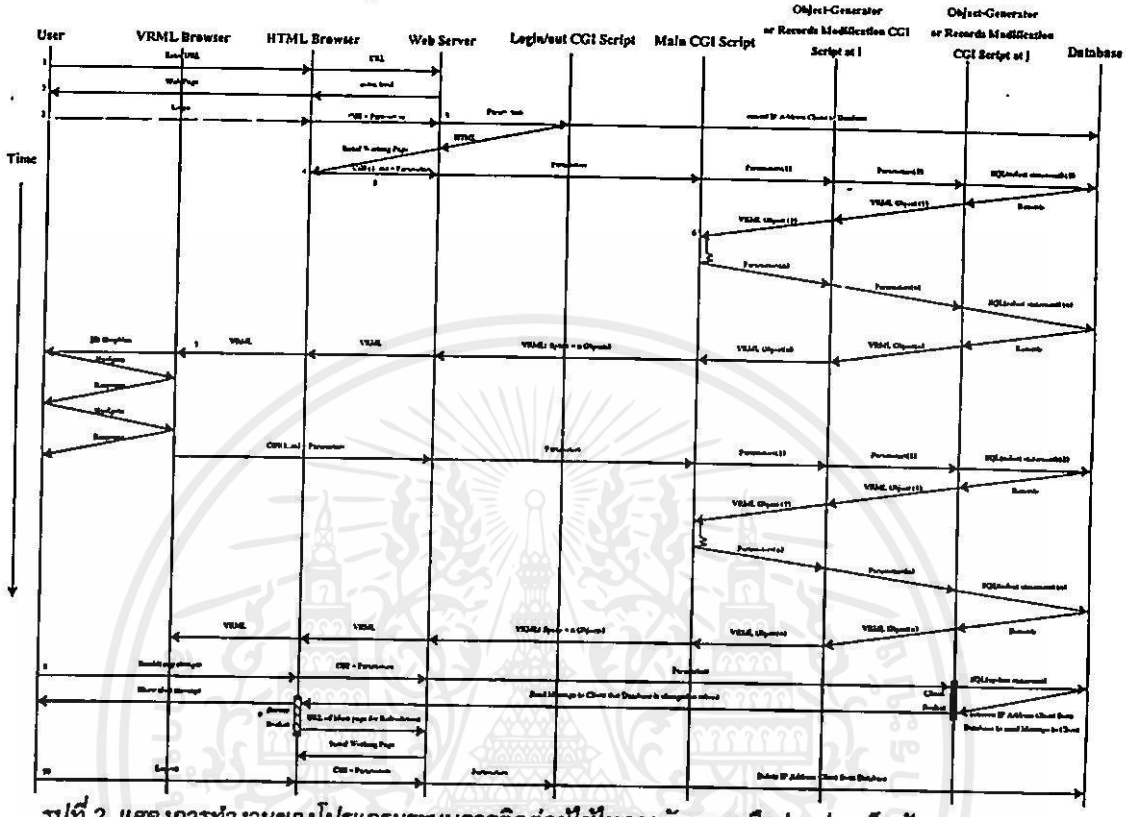
มิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผล และอาจจะมีการดึงข้อมูลจากฐานข้อมูล เพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกเอ็กซีคิวท์จะส่งเรสพอนท์กลับไปให้เว็บเซอร์ฟเวอร์ แล้วเว็บเซอร์ฟเวอร์จะส่งเรสพอนท์ดังกล่าวกลับไปให้กับเว็บเบราเซอร์ที่ส่งรีเควสท์มา ซึ่งในงานวิจัยนี้สามารถแบ่งสคริปต์ในระบบออกได้เป็น 2 ประเภท คือ

- สคริปต์ที่ให้เรสพอนท์ในรูปแบบของ HTML ซึ่งมี MIME Type เป็น text/html
- สคริปต์ที่ให้เรสพอนท์ในรูปแบบของ VRML ซึ่งมี MIME Type เป็น model/vrml

แต่หน้าที่หลักอย่างหนึ่งของระบบจัดการเครือข่ายก็คือ การมอนิเตอร์ดูสถานะของอุปกรณ์เครือข่าย ซึ่งระบบจัดการเครือข่ายที่เป็นลักษณะของ Web-Based โดยทั่วไปจะอาศัยการตั้งค่าช่วงเวลาที่จะให้มีการรีเฟชเอาไว้ เพื่อให้เว็บเพจแสดงสถานะล่าสุดของอุปกรณ์ ซึ่งลักษณะการทำงานที่กล่าวมาจะเป็น ลักษณะที่ไคลเอนต์ทำการรีเควสท์ไปที่เซอร์ฟเวอร์ทุกๆ ช่วงเวลาที่กำหนด ไม่ว่าข้อมูลที่เซอร์ฟเวอร์ จะมีการเปลี่ยนแปลงหรือไม่ก็ตาม ซึ่งหากนำหลักการนี้มาประยุกต์ใช้กับระบบจัดการเครือข่ายผ่านเว็บด้วย VRML แล้ว อาจจะทำให้ประสิทธิภาพในการทำงานลดลง หากมีการรีเฟชบ่อยๆ เนื่องจากการแสดงผล 3 มิติ ดังนั้นจึงได้เปลี่ยนมาอาศัยหลักการที่ว่าเซอร์ฟเวอร์จะเป็นฝ่ายส่งแมสเสจไปบอกไคลเอนต์เองเมื่อเวลาที่ข้อมูลฝั่งเซอร์ฟเวอร์มีการเปลี่ยนแปลง โดยที่ไคลเอนต์ไม่ต้องทำการรีเควสท์มาทุกๆ ช่วงเวลา ซึ่งหลักการที่กล่าวมานี้ ด้วยคุณสมบัติของโปรโตคอล HTTP เองไม่สามารถทำได้ [3] เนื่องจากการทำงานของ HTTP จะมีลักษณะที่เป็น request/response ทำให้เซอร์ฟเวอร์ไม่สามารถติดต่อไปยังไคลเอนต์เองได้ ถ้าไม่มีรีเควสท์มาจากไคลเอนต์ จึงได้มีการนำเทคโนโลยีของ JAVA Networking [4] ในเรื่อง Server Socket และ Client Socket เข้ามาช่วยในการทำงาน เพื่อให้สามารถรองรับการทำงานในลักษณะที่กล่าวมาข้างต้นได้

ซึ่งหลักการทำงานของระบบจัดการเครือข่ายผ่านเว็บด้วย VRML ทั้งหมดมีแผนภาพดังรูปที่ 2 (ตามหมายเลขอ้างอิงของแต่ละขั้นตอน) โดยรายละเอียดขั้นตอนการทำงานมีดังนี้ [7]

1. ผู้ใช้ทำการรีเควสท์โฮมเพจ
2. เว็บเซอร์ฟเวอร์จะเรสพอนท์โฮมเพจดังกล่าว ซึ่งเป็นหน้าจอให้ Login เข้าสู่ระบบ. จากนั้นผู้ใช้ที่มีหน้าที่ในการบริหารเครือข่ายใส่ Login name และ Password แล้วทำการคลิก OK ซึ่งจะเป็นการไปเรียก CGI Scripts ที่ทำหน้าที่ในการ Login ทำงาน
3. CGI Scripts ที่ทำหน้าที่ในการ Login ดังกล่าว จะทำการตรวจสอบ Login name และ Password ถ้าถูกต้องจะทำการบันทึก IP Address ของเครื่องไคลเอนต์ที่ทำการ Login ลงในฐานข้อมูลและทำการเรสพอนท์หน้าจอเริ่มการทำงาน โดยจะรวมเอาแอพเพล็ตที่ทำหน้าที่เป็น Server Socket เข้าไว้ด้วยกันกลับไปให้ไคลเอนต์
4. เมื่อไคลเอนต์ได้รับเว็บเพจดังกล่าวอย่างครบถ้วนแล้ว แอพเพล็ต Server Socket ที่รันอยู่ฝั่งไคลเอนต์จะมีหน้าที่รอรับแมสเสจที่จะถูกส่งมาจาก Client Socket ที่รันอยู่ฝั่งเซอร์ฟเวอร์ โดยจะคอยฟังแมสเสจอยู่ตลอดเวลา
5. ต่อจากนั้นผู้ใช้งานจะทำการรีเควสท์การทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยเซอร์ฟเวอร์จะคอยฟังรีเควสท์ที่ส่งมาและทำการเอ็กซีคิวท์สคริปต์ที่ระบุมาใน URL ซึ่งจะเป็นส่วนของเว็บเซอร์ฟเวอร์แอพพลิเคชัน (CGI) พร้อมทั้งส่งผ่านพารามิเตอร์ต่างๆ ที่ระบุมาในคิวรีด้วย เพื่อนำไปประมวลผลและอาจจะมีการดึงข้อมูลจากฐานข้อมูลเพื่อเป็นข้อมูลร่วมในการประมวลผลด้วย เมื่อสคริปต์ดังกล่าวถูกเอ็กซีคิวท์ก็จะส่งเรสพอนท์กลับไปให้เว็บเซอร์ฟเวอร์ ต่อจากนั้นเว็บเซอร์ฟเวอร์จึงส่งเรสพอนท์ดังกล่าวกลับไปให้เว็บเบราเซอร์ที่ส่งรีเควสท์มา
6. ในระบบนี้สามารถแบ่งสคริปต์ที่ให้เรสพอนท์เป็น VRML ออกได้เป็น 2 ประเภท คือ
   - สคริปต์หลักที่ใช้ในการสร้าง VRML Space ของแต่ละพื้นที่



รูปที่ 2. แสดงการทำงานของโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

- สคริปต์ย่อยที่ใช้ในการสร้าง VRML ออปเจ็กต์ต่างๆ ที่เกี่ยวข้องกับ VRML Space นั้นๆ

โดยทั้ง 2 ประเภทจะมีลักษณะการทำงานร่วมกัน คือ เมื่อสคริปต์หลักถูกเรียกใช้ สคริปต์จะทำการประมวลผลโดยการดึงข้อมูลของออปเจ็กต์ต่างๆ ที่เกี่ยวข้องกับพื้นที่นั้นๆ จากฐานข้อมูลแล้วนำมาประกอบขึ้นเป็น URL เพื่อใช้เรียกสคริปต์ย่อยในการสร้างออปเจ็กต์ต่างๆ

7. หลักการทำงานของ VRML จะต้องมี VRML Browser เพื่อทำหน้าที่ตีความ VRML ไปเป็นภาพแบบกราฟฟิกและจัดการในเรื่องของการโต้ตอบและการแอนิเมชัน
8. เมื่อไคลเอนต์กระทำการใดๆ บนเว็บเพจ อันเป็นการไปกระตุ้นให้ CGI scripts ที่ทำหน้าที่อัพเดตข้อมูลในฐานข้อมูลทำงานแล้ว โมดูลที่เป็น Client Socket ซึ่งฝังติดอยู่ใน CGI scripts นั้น จะขอเปิด connection และส่งแมสเสจไปยังแอพเพล็ตที่เป็น Server Socket ที่รันอยู่ทางฝั่งไคลเอนต์ตาม IP Address ที่อ่านมาได้จากฐานข้อมูลซึ่งมีการบันทึกไว้ในขั้นตอนที่ 3 โดยที่เครื่องไคลเอนต์จะต้องมี Address ที่ได้ลงทะเบียนถูกต้อง (Registered) ซึ่งเครื่องไคลเอนต์อาจจะติดต่อตรงไปหาเว็บเซอร์ฟเวอร์หรือติดต่อผ่านพร็อกซีก็ได้ [3]
9. เมื่อแอพเพล็ตที่เป็น Server Socket ทางฝั่งไคลเอนต์ได้รับแมสเสจแจ้งว่าฐานข้อมูลมีการอัพเดต ผู้ใช้ก็จะทำการสั่งให้เบราเซอร์ที่ active อยู่ทำการรีเฟช
10. และเมื่อผู้ใช้ทำการ Logout ออกจากระบบ ก็จะเป็นการไปเรียก CGI Scripts ที่ทำหน้าที่ในการ Logout ให้ทำงาน ซึ่ง CGI Scripts ดังกล่าวมีหน้าที่ในการตรวจสอบว่าไคลเอนต์เครื่องใดได้มีการ Logout ออกจากระบบและจะทำการลบ IP Address ของเครื่องนั้นออกจากฐานข้อมูลเพื่อลด

Technical Journal
136 Vol 1, No. 4, September - October 1999

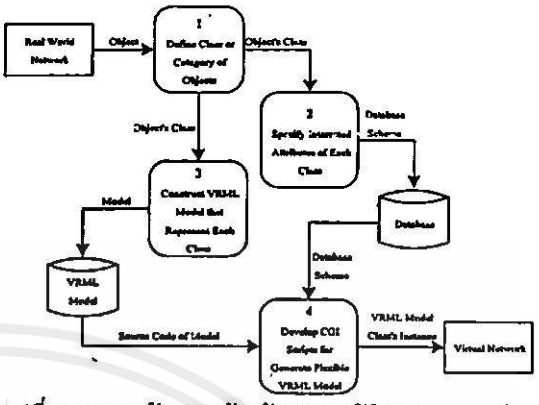
ปริมาณการส่งแมสเสจไปหาไคลเอนต์เมื่อมีการอัพเดตฐานข้อมูล

2.3 หลักการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ

เนื่องจากคุณสมบัติที่สำคัญอย่างหนึ่งของการจัดการเครือข่ายผ่านเว็บด้วย VRML นี้คือ การอ้างอิงวัตถุกับตำแหน่งที่ตั้งจริงทางกายภาพ ดังนั้นจุดประสงค์ของการแม็พข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ ก็คือ การนำเสนอข้อมูลด้านฟิสิกคอลของวัตถุและสภาพแวดล้อมอื่นๆ ที่เกี่ยวข้องให้อยู่ในรูปแบบของวัตถุที่เหมือนจริงหรือใกล้เคียงความจริงทั้งด้านรูปร่างลักษณะและตำแหน่งการจัดวางรวมทั้งสถานะต่างๆ ของวัตถุ ซึ่งแปรผันไปตามข้อมูลด้านลอจิคอล เพื่อให้ผู้ที่มีหน้าที่ในการดูแลระบบเครือข่ายสามารถทราบถึงตำแหน่งที่อยู่และสถานะของอุปกรณ์ได้ง่ายและสามารถเข้าไปดำเนินการกับอุปกรณ์ต่างๆ ได้รวดเร็วและถูกต้องมากขึ้น โดยนำความสามารถของระบบจัดการฐานข้อมูล, VRML และ CGI มาใช้เป็นเครื่องมือในการพัฒนางานในส่วนนี้ โดยกระบวนการในการแม็พ คือ

2.3.1 กระบวนการสร้างข้อมูลทางฟิสิกคอลและและลอจิคอลไปสู่รูปแบบ 3 มิติ

1. จัดคลาสหรือกลุ่มให้กับวัตถุที่จะทำการแม็พ เช่น Building, Room, Hub, Server, Workstation เป็นต้น
2. พิจารณาหาคุณสมบัติทั่วไปที่สนใจของแต่ละคลาส เช่น ชื่อ, ชนิด, ความกว้าง, ความยาว, ความสูง เป็นต้น เพื่อเป็นข้อมูลในการออกแบบฐานข้อมูล
3. นำข้อมูลต่างๆ ที่ได้พิจารณามาจัดทำโมเดลของ VRML ที่จะใช้นำเสนอแทนคลาสเหล่านั้น
4. สร้างฐานข้อมูลสำหรับทำการจัดเก็บค่าคุณสมบัติต่างๆ ของคลาสเหล่านั้น โดยจะได้กล่าวถึงการออกแบบฐานข้อมูลในหัวข้อที่ 2.4
5. พัฒนา CGI Script สำหรับสร้างโมเดล VRML ที่ได้จากข้อ 3 ให้สามารถยืดหยุ่นได้ตามคุณสมบัติเฉพาะของแต่ละคลาส เช่น ในด้านของความกว้าง, ความยาว, ความสูง, ตำแหน่งการจัดวาง เป็นต้น

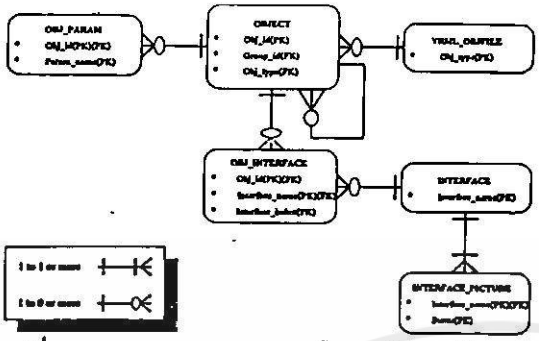


รูปที่ 3. แสดงหลักการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ

6. เมื่อทำการพัฒนาสคริปต์ต่างๆ เรียบร้อยแล้วจึงทำการทดลองใช้งาน โดยนำสคริปต์ที่พัฒนาขึ้นมาได้เก็บไว้ในเว็บเซอร์ฟเวอร์ แล้วให้ผู้ใช้ทำการรีเควสท์การทำงานที่ต้องการผ่านเว็บเบราเซอร์ ดังรูปที่ 3

2.4 การออกแบบฐานข้อมูล

จากกระบวนการในการสร้างข้อมูลทางฟิสิกคอลและลอจิคอลไปสู่รูปแบบ 3 มิติ ในหัวข้อที่ 2.3.1 จะต้องมีการดึงข้อมูลจากฐานข้อมูลเพื่อนำเสนอข้อมูลฟิสิกคอลและลอจิคอลในรูปแบบ 3 มิติที่อิงกับตำแหน่งจริงทางกายภาพ ซึ่งในบทความนี้สามารถแบ่งประเภทของฐานข้อมูลในมุมมองของระบบจัดการเครือข่ายผ่านเว็บด้วย VRML ออกได้เป็น 3 ประเภท ได้แก่ ฐานข้อมูลลอจิคอล, ฐานข้อมูลฟิสิกคอลและฐานข้อมูลการแม็พระหว่างข้อมูลในฐานข้อมูลลอจิคอลและฟิสิกคอล โดยชนิดของข้อมูลที่เก็บเช่น ชื่อวัตถุ, โปรโตคอลที่เชื่อมต่อภายในเครือข่าย, IP Address, สถานะของอุปกรณ์เครือข่าย, NodeName เป็นต้น และชนิดของข้อมูลที่เก็บในฐานข้อมูลฟิสิกคอล ได้แก่ ข้อมูลคุณสมบัติทางกายภาพของวัตถุ เช่น ความกว้าง ความยาว ความสูงของวัตถุ, ตำแหน่งและมุมการวางของวัตถุในแนวแกน X, Y, Z ส่วนชนิดของข้อมูลในฐานข้อมูลการแม็พระหว่างข้อมูลในฐานข้อมูลลอจิคอลและฟิสิกคอล ได้แก่ ความสัมพันธ์ระหว่างฐานข้อมูลลอจิคอลและฟิสิกคอล, สคริปต์ CGI ที่ใช้ในการสร้างโมเดล 3 มิติ ซึ่งฐานข้อมูลทั้ง 3 ประเภทนี้



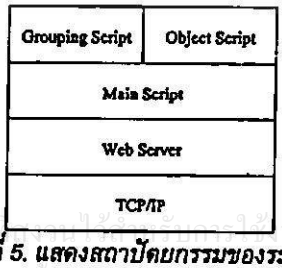
รูปที่ 4. แสดงER-Diagram แสดงโครงสร้างฐานข้อมูลเชิงสัมพันธ์ของระบบ

จะถูกนำมาออกแบบให้อยู่ในรูปของฐานข้อมูลเชิงความสัมพันธ์ โดยมีโครงสร้างดังรูปที่ 4 และคำอธิบายของแต่ละเอ็นติตี้จาก ER-Diagram ดังตารางที่ 1

2.5 สถาปัตยกรรมของระบบ

จากหลักการที่ได้กล่าวมาแล้วข้างต้น สามารถเขียนสถาปัตยกรรมของระบบการจัดการเครือข่ายผ่านเว็บในรูปแบบ 3 มิติ ได้ดังรูปที่ 5

โดยเว็บเบราเซอร์และเว็บเซอร์ฟเวอร์ติดต่อสื่อสารกันผ่านเครือข่ายอินเตอร์เน็ตที่ใช้ TCP/IP โดยผู้ใช้งานจะทำการรีเควสท์การทำงานที่ต้องการผ่านทางเว็บเบราเซอร์ โดยการระบุ URL ต่อจากนั้นเว็บเซอร์ฟเวอร์จะรวบรวมข้อมูลที่ร้องขอมาส่งต่อไปยัง CGI script หลัก ซึ่งก็คือ Main Script แล้วทำการประมวลผลข้อมูล โดยที่สคริปต์หลักอาจจะมีการเรียกสคริปต์ย่อยๆ เข้ามาประมวลผลร่วมด้วย สุดท้ายเมื่อได้ผลลัพธ์แล้ว เว็บเซอร์ฟเวอร์จะส่งผลลัพธ์กลับไปยังเว็บเบราเซอร์ที่ส่งรีเควสท์มา โดยรายละเอียดแต่ละส่วนแสดงในตารางที่ 2



รูปที่ 5. แสดงสถาปัตยกรรมของระบบ

โดยหน้าที่ของ Grouping Script และ Object Script นอกจากจะสร้าง VRML code ในส่วนที่เป็นวัตถุต่างๆ แล้ว ยังทำการสร้าง VRML code ที่ทำหน้าที่เป็น Viewpoint เพื่อหาตำแหน่งที่ตั้งของวัตถุฝังไว้กับทุกๆ วัตถุอีกด้วยและในส่วนของ Viewpoint นี้ เราสามารถที่จะใส่รายละเอียดข้อมูลเพิ่มเติมให้กับ Viewpoint ได้ด้วย ซึ่งในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML สิ่งที่ถูกนำไปใส่ไว้เป็นรายละเอียดเพิ่มเติมของแต่ละ Viewpoint ก็คือ ข้อมูลเกี่ยวกับตำแหน่งและสถานที่ตั้งจริงของ Viewpoint หรืออุปกรณ์ที่เราเห็นใน VRML ซึ่งจะมีลักษณะเป็นลำดับชั้น (Hierarchy) ของตำแหน่งที่ตั้งอุปกรณ์ที่เห็นอยู่ดังกล่าว เช่น อาคารสำนักวิจัย->ชั้น3->ห้องปฏิบัติการเครือข่าย->เครื่องคอมพิวเตอร์ เป็นต้น และเมื่อ VRML code เหล่านี้ถูกแปลโดย VRML Browser ชื่อของ Viewpoint ดังกล่าวจะปรากฏอยู่ในส่วนที่เป็นยูสเซอร์อินเตอร์เฟซของ VRML Browser และเมื่อผู้ใช้ทำการเลือกชื่อวัตถุที่ต้องการค้นหาแล้ว VRML Browser ก็จะทำการ Navigate ไปยังตำแหน่งของวัตถุนั้นโดยอัตโนมัติพร้อมทั้งแสดงข้อมูลเกี่ยวกับตำแหน่งและสถานที่ตั้งจริงของ Viewpoint หรืออุปกรณ์ที่เห็นอยู่นั้นด้วย ซึ่งลักษณะการทำงานนี้ จะทำให้ผู้บริหารเครือข่ายสามารถเห็นตำแหน่งที่ตั้งจริงๆ ของอุปกรณ์พร้อมกับทราบว่าตำแหน่งที่เห็นอยู่นั้นอยู่ ณ ที่ใดในสภาพแวดล้อมจริงของเครือข่าย

3. ผลการทดลองและบทวิเคราะห์ในการใช้ระบบ

จากทฤษฎีและหลักการที่ได้กล่าวมาแล้วข้างต้น ได้นำมาพัฒนาระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ขึ้น ซึ่งในส่วนนี้จะเป็นการแสดงผลการทดสอบการใช้งานระบบจัดการเครือข่ายผ่านเว็บด้วย VRML โดยสมมุติสถานการณ์ว่า ผู้บริหารเครือข่ายได้รับแจ้งปัญหาเกี่ยวกับเครือข่ายจากผู้ใช้งาน ผู้บริหารเครือข่ายก็จะเข้าไปตรวจสอบดูสถานะของอุปกรณ์ที่สงสัยว่าจะมีปัญหาพร้อมทั้งแจ้งตำแหน่งที่ตั้งของอุปกรณ์ให้เจ้าหน้าที่สนามเข้าไปดำเนินการแก้ไขต่อไป โดยผู้บริหารเครือข่ายจะใช้ระบบจัดการเครือข่ายผ่านเว็บด้วย VRML นี้

Technical Journal
138 Vol 1, No. 4, September - October 1999

ตารางที่ 1. แสดงคำอธิบายของแต่ละ Entity จาก ER-Diagram

ชื่อ Entity	คำอธิบาย
OBJECT	เก็บรายละเอียดของวัตถุที่เราสนใจ
OBJECT_PARAM	เก็บพารามิเตอร์ เฉพาะของแต่ละออปเจ็กต์
VRML_OBJFILE	เก็บความสัมพันธ์ระหว่างชนิดของวัตถุกับชื่อของสคริปต์ที่ใช้สร้างโมเดล VRML ของวัตถุนั้นๆ
OBJ_INTERFACE	เก็บรายละเอียดของวัตถุในส่วนที่เป็น Network Interface กรณีที่วัตถุนั้นเป็นอุปกรณ์เครือข่าย
INTERFACE	เก็บรายละเอียดของ Interface แต่ละชนิด ซึ่งจะใช้เป็นข้อมูล Lookup สำหรับการทำงานของระบบ
INTERFACE_PICTURE	เก็บรายละเอียดของรูปภาพที่จะใช้นำเสนอแทน Interface แต่ละชนิดในสถานะต่างๆ ซึ่งจะใช้เป็นข้อมูลส่วนหนึ่งในการสร้างโมเดล VRML

ตารางที่ 2. แสดงคำอธิบายรายละเอียดแต่ละส่วนของสถาปัตยกรรมของระบบ

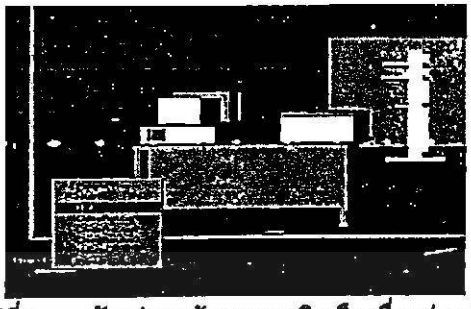
Grouping Script	ทำหน้าที่ในการสร้าง VRML Code ของส่วนที่เป็น Parent Coordinate System ซึ่งใช้สำหรับการจัดวางตำแหน่งต่างๆ ของวัตถุที่เป็น Child Coordinate System ซึ่งได้จาก Object Script ให้อยู่ในกลุ่มของ coordinate ที่สัมพันธ์กัน เช่น อาคาร, ห้องหรือพื้นที่ที่มีการกำหนดขอบเขต เป็นต้น
Object Script	ทำหน้าที่ในการสร้าง VRML Code ในส่วนที่เป็นวัตถุต่างๆ เช่น อุปกรณ์เครือข่าย, โต๊ะ, ชั้นวางอุปกรณ์ เป็นต้น ซึ่งบาง Object Script สามารถทำหน้าที่เป็น Grouping Script ได้ด้วย
Main Script	ทำหน้าที่เป็นสคริปต์หลักในการเริ่มต้นการทำงานของระบบ โดยจะเป็นสคริปต์แรกที่ถูกเว็บเซอร์ฟเวอร์เอ็กซีคิวท์ หลังจากนั้น ภายใน Main Script จะมีกลไกและกระบวนการในการเรียกใช้ Grouping Script และ Object Script อีกทีหนึ่ง ซึ่ง Main Script ถือว่าเป็น Root Script ของระบบ
Web Server	ทำหน้าที่รับรีเควสท์จากผู้ใช้ แล้วทำการเอ็กซีคิวท์สคริปต์ตามที่ถูกรีเควสท์และทำหน้าที่ในการส่งผลลัพธ์ที่ได้จากสคริปต์กลับไปยังเว็บเบราเซอร์
TCP/IP	เป็นโปรโตคอลที่ใช้ในการติดต่อสื่อสารระหว่างเว็บเซอร์ฟเวอร์และเว็บเบราเซอร์

เป็นเครื่องมือในการช่วยปฏิบัติภารกิจดังกล่าว โดยเครื่องมือที่ใช้ในการทดลองมีดังนี้

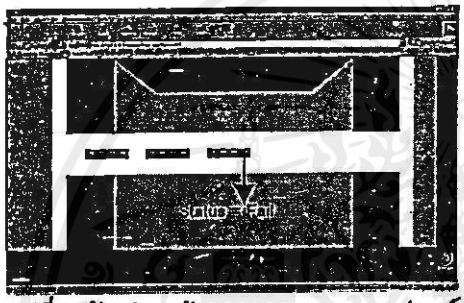
- Web Browser = Netscape Navigator 4.5
- Plug-in VRML Browser = Cosmo Player 2.1
- ระบบเว็บเซอร์ฟเวอร์ = IIS (Internet Information Server) 4.0
- ระบบจัดการฐานข้อมูล (DBMS : Database Management System) = Paradox 7.0
- Server = OS WinNT, CPU Pentium 166, Ram 96 MB, HardDisk 2 GB
- Client = OS Win95, CPU Pentium 133, Ram 32 MB, HardDisk 1 GB

โดยมีรายละเอียดการทำงานเป็นดังนี้

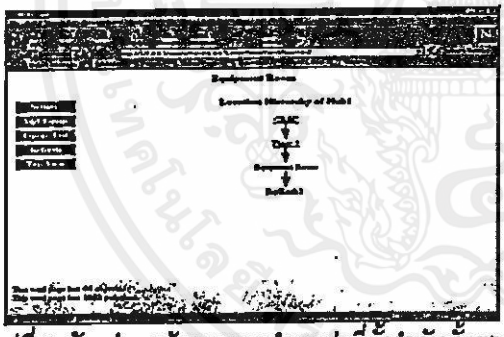
1. ผู้บริหารเครือข่ายเข้าสู่ระบบจัดการเครือข่ายผ่านเว็บด้วยเว็บเบราเซอร์ [7]
2. จากนั้นผู้บริหารเครือข่ายทำการคลิกเลือกชื่ออุปกรณ์ที่ต้องการตรวจสอบจาก List Box View point ดังรูปที่ 6
3. เมื่อได้ทำการเลือกอุปกรณ์แล้ว VRML Browser จะ Navigate ไปยังตำแหน่งที่อุปกรณ์ดังกล่าวถูกติดตั้งอยู่ ซึ่งจะปรากฏเห็นสถานะของอุปกรณ์นั้น จากรูปที่ 7 จะพบว่ามีพอร์ทหนึ่ง fail อยู่
4. จากตำแหน่งของอุปกรณ์ที่ผู้บริหารเครือข่ายได้เห็นใน VRML นั้น ผู้บริหารเครือข่ายก็จะทราบว่าอุปกรณ์ถูกติดตั้งอยู่ตำแหน่งบนสุดของ Rack แต่ยังไม่ทราบว่าตำแหน่งที่เห็นนั้น อยู่ ณ ที่ใดในสภาพแวดล้อมจริง ผู้บริหารเครือข่ายจึงต้องการข้อมูลมากกว่านั้นเพื่อสามารถแจ้งให้เจ้าหน้าที่สนามทราบถึงตำแหน่งดังกล่าวได้ชัดเจนขึ้น ซึ่งสามารถทำได้โดยคลิกเลือกที่ตัวอุปกรณ์ โดยจะแสดงเป็นลักษณะของ



รูปที่ 6. ตัวอย่างหน้าจอการคลิกเลือกชื่ออุปกรณ์ที่ต้องการตรวจสอบจาก List box Viewpoint



รูปที่ 7. ตัวอย่างหน้าจอแสดงสถานะของอุปกรณ์



รูปที่ 8. ตัวอย่างหน้าจอแสดงตำแหน่งที่ตั้งลำดับชั้นของอุปกรณ์

ลำดับชั้นของตำแหน่งที่ตั้งอุปกรณ์ดังกล่าว นั่นคือ อุปกรณ์ที่มีพอร์ทหนึ่ง fail อยู่นั้นคืออุปกรณ์ชื่อ Hub 1 ตั้งอยู่ที่ตึก CRSC ชั้น 1 ห้อง Equipment Room บน EqrRack1 ดังรูปที่ 8

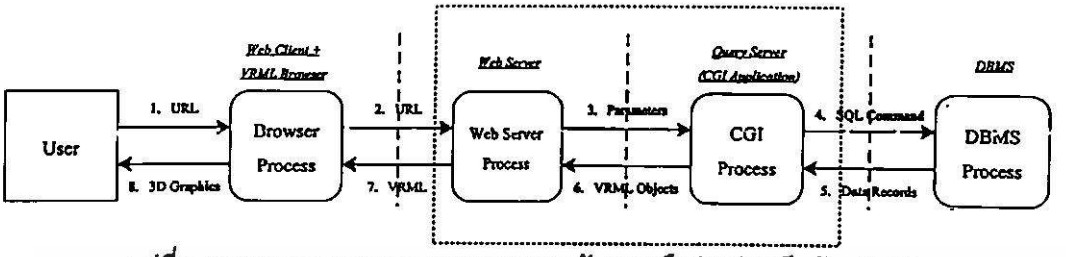
5. โดยในแต่ละระดับของข้อมูลตำแหน่งที่ตั้งจะมีแผนผังแสดงตำแหน่งที่ตั้งของอุปกรณ์ในแต่ละระดับอยู่ด้วย อย่างเช่นถ้าคลิกที่ชื่อห้องหรือชั้นหรืออาคาร ก็จะแสดงสัญลักษณ์ของแผนผังรวมของห้อง, ชั้น, อาคารต่างๆ
6. และจากข้อมูลที่ได้จากแผนผังในทุกระดับ ผู้บริหารเครือข่ายก็สามารถที่จะนำมาหาเส้นทางในการไปยังอุปกรณ์และแผนที่ภาพรวมที่ผู้บริหารเครือข่ายอยู่เพื่อไปยังสถานที่ที่อุปกรณ์เครือข่ายติดตั้งอยู่ได้ ดังนั้นผู้บริหารเครือข่ายสามารถนำข้อมูลตำแหน่งที่ตั้งของอุปกรณ์ที่ได้ทั้งหมด ทำการแจ้งให้กับเจ้าหน้าที่สนามเข้าไปทำการแก้ไขปัญหาที่ตัวอุปกรณ์ได้อย่างชัดเจนและรวดเร็วขึ้น

จากผลการทดลองเบื้องต้น จะเห็นว่าผู้บริหารเครือข่ายสามารถใช้ระบบจัดการเครือข่ายผ่านเว็บด้วย VRML ในการตรวจสอบหาตำแหน่งที่ตั้งของอุปกรณ์ได้อย่างชัดเจนมากขึ้น ทำให้สามารถสื่อสารกับเจ้าหน้าที่ผู้รับผิดชอบได้ถูกต้อง ซึ่งจะส่งผลทำให้การแก้ปัญหาเครือข่ายที่เกิดขึ้นดังกล่าว สามารถทำได้อย่างถูกต้องและรวดเร็วมากขึ้นไปด้วย แต่เนื่องด้วยการทำงานของระบบจัดการเครือข่ายผ่านเว็บด้วย VRML จะเป็นการนำเสนอข้อมูลในรูปแบบ 3 มิติ ซึ่งส่วนนี้เป็นปัจจัยหนึ่งที่จะมีผลต่อประสิทธิภาพการทำงานของระบบจัดการเครือข่าย จากปัจจัยดังกล่าว ระบบจัดการเครือข่ายผ่านเว็บด้วย VRML อาจจะมีข้อจำกัดอยู่ที่การใช้งานกับระบบเครือข่ายขนาดใหญ่ จากรูปที่ 9 (การทำงานหมายเลข 8) หลังจากที่เครื่องไคลเอนต์ได้รับเท็กไฟล์จากเว็บเซอร์ฟเวอร์เรียบร้อยแล้ว เมื่อผู้ใช้ทำการ navigate ไปในโลก 3 มิติ ประสิทธิภาพในการ navigate นี้ขึ้นอยู่กับประสิทธิภาพของเครื่องไคลเอนต์ [1] เนื่องจากเมื่อเครื่องไคลเอนต์ได้รับเท็กซ์ไฟล์ VRML มาแล้ว ถัดจากนี้เป็นหน้าที่ของเครื่องไคลเอนต์ในการประมวลผลเพื่อทำหน้าที่ตีความ VRML ไปเป็นภาพแบบกราฟฟิกและจัดการในเรื่องของการโต้ตอบและการแอนิเมชัน

โดยการวิเคราะห์ต่อไปนี้จะดำเนินตามวิธีและขั้นตอนการทำงานของระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML โดยเริ่มที่ผู้ใช้ทำการร้องขอข้อมูลจากเว็บเซอร์ฟเวอร์ดังรูปที่ 9

สุดท้ายจึงได้ทำการศึกษาถึงปัจจัยที่มีผลต่อขนาดของ Response Time หลังจากเครื่องไคลเอนต์ได้รับเท็กซ์ไฟล์จากเว็บเซอร์ฟเวอร์เรียบร้อยแล้ว โดยจะเริ่มจับเวลาตั้งแต่ผู้ใช้ทำการ navigate ไปในโลก 3 มิติ

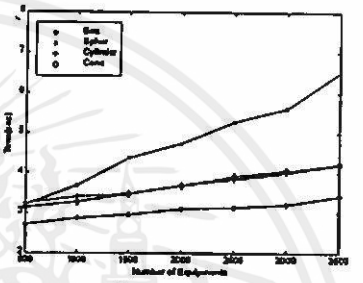
Technical Journal
140 Vol 1, No. 4, September - October 1999



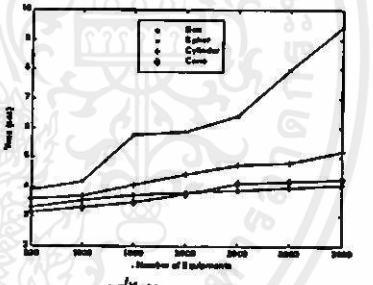
รูปที่ 9. แสดง Diagram Level-0 ของระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML

ในการเคลื่อนที่จากตำแหน่งของ Viewpoint หนึ่งไปยังอีก Viewpoint หนึ่ง (Viewpoint คือ การกำหนดตำแหน่งการมองของโลก 3 มิติ) นั่นคือจะทำการศึกษาเวลาที่ผู้ใช้ทำการ Travel ไปใน VRML World หรือเปลี่ยน Viewpoint ซึ่งกิจกรรมที่กล่าวมานี้เกิดจากการประมวลผลที่เครื่องไคลเอนต์เพียงอย่างเดียว และเนื่องจากถ้ามีจำนวนออปเจ็กต์ในเท็กไฟล์ VRML มากขึ้น เวลาที่ใช้ในการตีความ VRML ไปเป็นภาพแบบกราฟฟิกและการ navigate ก็ต้องใช้เวลามากขึ้นตามไปด้วย โดยปัจจัยที่จะทำการศึกษา คือ ชนิดของ VRML Object ต่างๆ หรือโหนด Geometry [1] ที่มีไว้สำหรับกำหนดรูปทรงของวัตถุ ในภาษา VRML มีโหนดรูปทรงพื้นฐานให้ใช้งานดังนี้ คือ ทรงกลม (Sphere), ทรงกระบอก (Cylinder), ทรงลูกบาศก์ (Box), และทรงกรวยกลม (Cone) ดังผลการทดลองตารางที่ 3 และกราฟรูปที่ 10 จะพบว่าไม่ว่า VRML Object ชนิดใดๆ เมื่อมีจำนวนออปเจ็กต์มากขึ้น เวลาที่ใช้ในการ navigate จาก viewpoint หนึ่งไปยังอีก viewpoint หนึ่ง ก็ต้องใช้เวลามากขึ้นด้วย

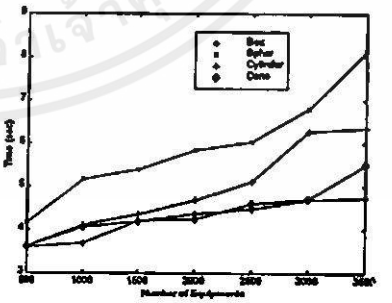
นอกจากรูปร่างของ VRML Object ชนิดต่างๆ ที่มีผลต่อขนาดของ Response Time ในการ navigate แล้ว คุณสมบัติของ VRML Object ก็เป็นอีกส่วนหนึ่งที่มีผลต่อ Response time ดังนั้นจึงได้ทำการทดลองเพื่อหา Response time ในการ navigate โดยการใส่ Shading และ Texture ให้กับ VRML Object ชนิดต่างๆ ดังผลการทดลองตารางที่ 4,5 และกราฟรูปที่ 11,12 จากผลการทดลองจะพบว่าการใส่คุณสมบัติให้กับ VRML Object ไม่ว่าจะเป็นการใส่ shading หรือ texture เพื่อให้ออปเจ็กต์ในโลก 3 มิติ มีความเหมือนจริงมากยิ่งขึ้นนั้น ผลที่ตามมาคือเวลาที่ใช้ในการ navigate จาก viewpoint หนึ่งไปยังอีก viewpoint หนึ่ง ก็ต้องใช้เวลามากขึ้นเช่นกัน



รูปที่ 10. แสดงกราฟที่ได้จากการทดลองของตารางที่ 3



รูปที่ 11. แสดงกราฟที่ได้จากการทดลองของตารางที่ 4



รูปที่ 12. แสดงกราฟที่ได้จากการทดลองของตารางที่ 5

จากผลการทดลองข้างต้นจะพบว่าปัจจัยที่มีผลต่อ Response Time ในการ navigate ไปในโลก 3 มิติของเครื่องไคลเอนต์ ได้แก่ ชนิดของ VRML Object

ตารางที่ 3. แสดง Response Time (Seconds) โดยเฉลี่ยของผลการทดลอง 3 ครั้งในการเปลี่ยน viewpoint จาก (0,0,500) ไปที่ (0,0,0) ของ VRML Object ชนิดต่างๆ

จำนวนออปเจกต์	500	1000	1500	2000	2500	3000	3500
Box	2.73	2.85	2.95	3.04	3.09	3.25	3.40
Sphere	3.21	3.64	4.30	4.74	5.22	5.56	6.42
Cylinder	3.22	3.41	3.48	3.65	3.82	3.98	4.21
Cone	3.07	3.17	3.47	3.59	3.78	3.96	4.22

หมายเหตุ : ในโลก 3 มิติ ประกอบด้วย Coordinate x y และ z โดยที่ x จะเป็นตำแหน่งจาก ซ้ายไปขวา, y เป็นตำแหน่งจากล่างขึ้นบน และ z เป็นตำแหน่งจากไกลมาใกล้

ตารางที่ 4. แสดง Response Time (Seconds) โดยเฉลี่ยของผลการทดลอง 3 ครั้งในการเปลี่ยน viewpoint จาก (0,0,500) ไปที่ (0,0,0) โดยการการใส่ Shading ให้กับ VRML Object ชนิดต่างๆ

จำนวนออปเจกต์	500	1000	1500	2000	2500	3000	3500
Box	3.27	3.55	3.72	3.78	3.84	4.00	4.05
Sphere	3.98	4.23	5.70	5.88	6.37	7.92	9.43
Cylinder	3.55	3.69	4.02	4.32	4.73	4.80	5.09
Cone	3.17	3.35	3.46	3.73	3.98	4.18	4.24

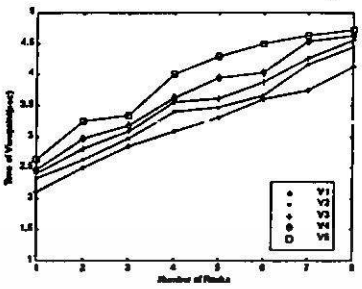
ตารางที่ 5. แสดง Response Time (Seconds) โดยเฉลี่ยของผลการทดลอง 3 ครั้งในการเปลี่ยน viewpoint จาก (0,0,500) ไปที่ (0,0,0) โดยการใส่ Texture ให้กับ VRML Object ชนิดต่างๆ

จำนวนออปเจกต์	500	1000	1500	2000	2500	3000	4000
Box	3.52	3.71	4.13	4.38	4.50	4.68	4.76
Sphere	4.19	5.02	5.38	5.84	6.04	6.75	8.20
Cylinder	3.59	4.04	4.31	4.67	5.11	6.18	6.37
Cone	3.63	4.02	4.20	4.27	4.59	4.78	5.50

ตารางที่ 6. แสดง Response Time (Seconds) โดยเฉลี่ยของผลการทดลอง 3 ครั้งในการเปลี่ยนตำแหน่ง viewpoint ต่างๆ กับจำนวนอุปกรณ์ที่แตกต่างกันในสภาพแวดล้อมจริงในการจัดการเครือข่าย

จำนวน Viewpoint	1 Rack (16 Switch)	2 Rack (16 Switch)	4 Rack (24 Switch)	5 Rack (32 Switch)	5 Rack (40 Switch)	6 Rack (48 Switch)	7 Rack (56 Switch)	8 Rack (64 Switch)
V_1	2.10	2.50	2.84	3.09	3.31	3.60	3.75	4.12
V_2	2.32	2.63	2.97	3.41	3.47	3.66	4.15	4.44
V_3	2.41	2.81	3.09	3.56	3.62	3.87	4.25	4.56
V_4	2.47	2.97	3.18	3.63	3.94	4.03	4.53	4.63
V_5	2.63	3.25	3.35	4.00	4.28	4.50	4.62	4.72

Technical Journal
142 Vol I, No. 4, September - October 1999



รูปที่ 13. แสดงกราฟที่ได้จากการทดลองของตารางที่ 6

ต่างๆ และคุณสมบัติที่ใส่ให้กับ VRML Object ได้แก่ การใส่ Texture และ Shading

สุดท้ายจึงได้ทำการทดลองกับสภาพแวดล้อมจริงในระบบจัดการเครือข่าย เพื่อทำการวิเคราะห์ว่าจำนวนอุปกรณ์ในเครือข่ายมีผลกับการทำงานของระบบจัดการเครือข่ายผ่านเว็บด้วย VRML นี้หรือไม่ โดยตัวแปรที่ใช้แทนจำนวนอุปกรณ์ในเครือข่าย คือ จำนวนออปเจ็กต์ของ VRML นั่นเอง โดยจะทำการทดลองเพื่อหา Response Time ในการใช้งานกับจำนวนอุปกรณ์แตกต่างกัน โดยในการทดลองแต่ละครั้งจะเพิ่มจำนวนอุปกรณ์เข้าไปทีละ 1 Rack ที่ประกอบขึ้นจาก VRML Object ชนิด Box จำนวน 4 ออปเจ็กต์ โดยทุกๆ 1 Rack ที่เพิ่มเข้าไปจะประกอบด้วยอุปกรณ์เครือข่าย 8 ชิ้น ซึ่งอุปกรณ์เครือข่ายที่ใช้ในการทดลองครั้งนี้จะเป็น Switch ที่ประกอบขึ้นจาก VRML Object ชนิด Box จำนวน 13 ออปเจ็กต์และ Image รูปพอร์ทจำนวน 12 ออปเจ็กต์ นั่นคือ 1 Rack ประกอบด้วย 204 ออปเจ็กต์ โดยได้จำลองมาจาก Switch ยี่ห้อ 3 Com รุ่น Super Stack II Switch 2000 ดังผลการทดลองตารางที่ 6 และกราฟรูปที่ 13

ให้ V_1 = การเปลี่ยนตำแหน่ง Viewpoint จากตำแหน่ง Coordinate(x,y,z) ที่ (0,0,10) เป็น (0,0,0) หน่วยเป็นเมตร

ให้ V_2 = การเปลี่ยนตำแหน่ง Viewpoint จากตำแหน่ง Coordinate(x,y,z) ที่ (0,0,50) เป็น (0,0,0)

ให้ V_3 = การเปลี่ยนตำแหน่ง Viewpoint จากตำแหน่ง Coordinate(x,y,z) ที่ (0,0,100) เป็น (0,0,0)

ให้ V_4 = การเปลี่ยนตำแหน่ง Viewpoint จากตำแหน่ง Coordinate(x,y,z) ที่ (0,0,150) เป็น (0,0,0)

ให้ V_5 = การเปลี่ยนตำแหน่ง Viewpoint จากตำแหน่ง Coordinate(x,y,z) ที่ (0,0,200) เป็น (0,0,0)

จากตารางที่ 6 และกราฟรูปที่ 13 แสดงผลการทดลองที่ได้ จะพบว่า ผลลัพธ์ไปในทิศทางเดียวกัน นั่นคือ ตัวแปรที่มีผลกระทบที่ชัดเจนที่สุดต่อขนาดของเวลาที่เครื่องไคลเอนต์ของผู้ใช้ใช้ในการประมวลผลในการเปลี่ยนตำแหน่ง Viewpoint คือ จำนวนออปเจ็กต์ ดังนั้นในสภาวะที่มีจำนวนออปเจ็กต์จำนวนมาก จะทำให้เวลาที่ใช้ในการประมวลผลมากขึ้นด้วย ซึ่งก็คือจำนวนอุปกรณ์ในเครือข่าย นั่นหมายถึงขนาดของเครือข่ายมีผลกับการใช้งานระบบจัดการเครือข่ายผ่านเว็บด้วย VRML

ดังนั้น เมื่อผู้ใช้ทำการ Travel ไปใน VRML World หรือเปลี่ยน Viewpoint กิจกรรมที่กล่าวมานี้เกิดจากการประมวลผลที่เครื่องไคลเอนต์เพียงอย่างเดียว ดังนั้นประสิทธิภาพของเครื่องไคลเอนต์เป็นอีกปัจจัยหนึ่งที่มีผลกระทบต่อประสิทธิภาพในการทำกิจกรรมดังกล่าว จากปัจจัยนี้ ในบางครั้งอาจจำเป็นที่จะต้องมีการ Optimize การนำเสนอ VRML ให้อยู่ในรูปแบบที่ใช้เวลาในการประมวลผลน้อยลง เช่น การเลือกใช้ชนิดของ VRML Object ที่มีการประมวลผลน้อย, การลดคุณสมบัติความเหมือนจริงของ VRML Object, การลดความซับซ้อน, รายละเอียดและขนาดของออปเจ็กต์ต่างๆ ลง [1]

4. บทสรุป

การจัดการเครือข่ายผ่านเว็บด้วย VRML ทำให้สามารถเข้าไปจัดการ, ตรวจสอบและดูแลระบบเครือข่ายคอมพิวเตอร์ผ่านทางอินเตอร์เน็ตโดยใช้เว็บเบราเซอร์ในรูปแบบที่เสมือนจริงและอิงกับลักษณะทางกายภาพและจะเป็นตัวต้นแบบที่จะสามารถถูกนำไปพัฒนาให้มีความสามารถและความสมบูรณ์เพิ่มมากขึ้น เพื่อรองรับการจัดการระบบเครือข่ายที่มีความซับซ้อนมากขึ้นต่อไปได้

แต่เนื่องจากข้อมูลที่จัดเก็บที่นำมาใช้ในการทดลองครั้งนี้ยังมีความซับซ้อนและรายละเอียดไม่มาก

นัก ซึ่งผู้ใช้สามารถที่จะจัดเก็บข้อมูลดังกล่าวได้มากกว่านี้ โดยสามารถที่จะเพิ่มระดับความสัมพันธ์ของวัตถุต่างๆ ได้มากเท่าที่ต้องการ เพื่อให้เห็นภาพที่ละเอียดและมีความเหมือนจริงมากยิ่งขึ้น แต่ทั้งนี้ยิ่งข้อมูลที่จัดเก็บมีความซับซ้อนและมีรายละเอียดมากเท่าไร เวลาที่จะถูกใช้ในการประมวลผลและการแสดงผลของระบบก็ยิ่งจะต้องมากขึ้นเช่นกัน แต่อย่างไรก็ตามก็ยังมีอีกหลายๆ ปัจจัยที่มีผลต่อการทำงานของระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML อย่างเช่น

- ความเหมือนจริงในการนำเสนอออปเจ็กต์ต่างๆ นั้นขึ้นอยู่กับโปรแกรมว่าสามารถกำหนดเงื่อนไขในการนำเสนอได้มากน้อยเพียงใด ดังนั้นถ้ามีการเก็บข้อมูลทางฟิสิกคอลและลอจิกคอลลงในฐานข้อมูลละเอียดมากเท่าไร ความเหมือนจริงก็จะมีมากขึ้นแต่เวลาที่ใช้ในการนำเสนอก็จะช้าลงไปด้วย
- ความเร็วและความต่อเนื่องในการท่องไปในแต่ละฉาก จะพบว่า การที่มีวัตถุมากเกินไปหรือมีวัตถุที่มีความซับซ้อนมากๆ จะทำให้ความเร็วในการท่องลดลง แต่เมื่อทำการลดจำนวนวัตถุที่ไม่จำเป็นออกหรือดัดแปลงวัตถุให้มีความซับซ้อนน้อยลงจะเป็นการแก้ไขให้ดีขึ้นได้
- เวลาในการโหลดข้อมูลจากเว็บเซอร์ฟเวอร์มายังเว็บเบราเซอร์จะประหยัดเวลาเนื่องจากเป็นการโหลดข้อมูล VRML ที่เป็นเท็กไฟล์ แต่จะมารับภาระหนักที่เครื่องไคลเอนต์ที่จะต้องแปลข้อความนั้นให้อยู่ในรูป 3 มิติ จึงต้องทำงานบนเครื่องที่มีประสิทธิภาพสูงๆ
- เวลาที่ใช้ในการนำเสนอออปเจ็กต์ต่างๆ ในรูปแบบ 3 มิตินั้น ขึ้นอยู่กับอัตราส่วนที่ใช้ในการแม็พข้อมูลฟิสิกคอลและข้อมูลลอจิกคอลไปเป็นการนำเสนอแบบกราฟฟิก 3 มิติ ด้วยว่าใช้อัตราส่วนเท่าไร ยิ่งใช้อัตราส่วนในการนำเสนอมากเท่าไร เวลาที่ใช้ในการแสดงรูป 3 มิติก็จะมากขึ้นตามไปด้วย
- เมื่อประสิทธิภาพของฮาร์ดแวร์และการประมวลผลทางกราฟฟิก 3 มิติ มีประสิทธิภาพมากขึ้น จะทำให้สามารถเพิ่มเติมองค์ประกอบต่างๆ เข้าไปในระบบได้อย่างเต็มที่และยังทำให้มีความเสมือนจริงมากขึ้น

สิ่งที่จะทำการศึกษาต่อไปนั้น จะทำการศึกษาและวิเคราะห์ถึง Response Time ในส่วนต่างๆ จากการทำงานในรูปที่ 9 ว่าการทำงานส่วนใดที่มีผลต่อการจัดการเครือข่ายมากที่สุด ไม่ว่าจะเป็นเวลาที่ไคลเอนต์ติดต่อกับเว็บเซอร์ฟเวอร์, เว็บเซอร์ฟเวอร์ติดต่อกับ CGI Script หรือ CGI Script ติดต่อกับฐานข้อมูล เพื่อหาว่าเครือข่ายขนาดใดที่เหมาะสมต่อการจัดการเครือข่ายผ่านเว็บด้วย VRML

เอกสารอ้างอิง

[1] Andrea L. Ames, David R. Nadeau and John L. Moreland, *VRML 2.0 Sourcebook*, Second Edition. New York : John Wiley & Sons,Inc. 1997.

[2] Jeffry Dwight and Michael Erwin, *Special Edition Using CGI*, Indianapolis : Que Corporation. 1996.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk and T. Berners-Lee, *Network Working Group Request for Comments 2068*, [Online]. Available : http://www.w3.org/Protocols/rfc2068/rfc2068. 1997.

[4] Gary Cornell and Cay S. Horstmann, *Core Java*, Second Edition. California : SunSoft Press, A Prentice Hall Tittle. 1997.

[5] Hewlett Packard, *HP OpenView*, [Online], Available : http://www.openview.hp.com. 1999.

[6] Sun Microsystems, *Solstice SunNet Manager*, [Online]. Available : http://www.sun.com/software/solstice/em-products/network/sunnetmgr.html. 1999.

[7] Dang Cholpripimonrat, *3-D Web-Based Network Management Interface Using VRML*, [Online]. Available : http://161.246.38.18/dang/wwwroot/vnms/default.htm. 1999.

KMITL Technical Journal
144 Vol 1, No. 4, September - October 1999

แดง ชลไพรพิมลรัตน์ ปัจจุบันเป็นนักศึกษาระดับปริญญาโท หลักสูตรวิทยาศาสตร์มหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง จบการศึกษาวิทยาศาสตร์บัณฑิต มหาวิทยาลัยหอการค้าไทย พ.ศ. 2539

อัครินทร์ คุณกิตติ จบการศึกษาวิศวกรรมศาสตร์มหาบัณฑิต (วิศวกรรมไฟฟ้า คอมพิวเตอร์) และวิศวกรรมศาสตร์บัณฑิต (อิเล็กทรอนิกส์) เกียรตินิยมอันดับหนึ่ง จากสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ปัจจุบันเป็นอาจารย์ประจำคณะเทคโนโลยีสารสนเทศ และผู้ช่วยผู้อำนวยการสำนักวิจัยและบริการคอมพิวเตอร์ นอกจากนี้ยังเป็นนักวิจัยสำนักวิจัยการสื่อสารและเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทำงานวิจัยด้านระบบเครือข่ายการสื่อสารคอมพิวเตอร์ การประมวลผลแบบกระจาย และการจัดการระบบเครือข่ายและคอมพิวเตอร์

ภาคผนวก ข.

วิธีการใช้งานโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

โปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML เป็นโปรแกรมคอมพิวเตอร์ ซึ่งถูกพัฒนาขึ้นจากภาษาปาสคาลและใช้โปรแกรมเดลไฟล์ในการพัฒนา โดยมีความสามารถในการตรวจดูสถานะปัจจุบันของระบบเครือข่ายและสภาพแวดล้อมทางกายภาพได้ จุดประสงค์ของการใช้งานโปรแกรมนี้ เพื่อใช้เป็นส่วนติดต่อกับผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วยอินเตอร์เฟซที่เสมือนจริงและอิงกับตำแหน่งทางกายภาพ เพื่อช่วยให้ผู้บริหารเครือข่ายสามารถเข้าไปตรวจสอบและแก้ปัญหาของระบบเครือข่ายได้อย่างทันท่วงที

โปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถทำการติดต่อโดยการระบุ URL จากเว็บเบราเซอร์ผ่านโปรโตคอล HTTP เมื่อโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML สามารถทำการติดต่อกับเว็บเซิร์ฟเวอร์ได้แล้ว โปรแกรมจะแสดงรายชื่อข้อมูลทางฟิสิกคอลและลอจิคอล โดยผู้ใช้สามารถเลือกที่จะตรวจสอบข้อมูลใดๆ ได้ตามต้องการ นอกจากนั้นผู้ใช้ยังสามารถเพิ่มหรือแก้ไขข้อมูลเครือข่ายได้เฉพาะผู้ใช้ที่มีสิทธิ์เท่านั้น โดยจะมีการตรวจสอบ Username และ Password เมื่อทำการ login เข้าสู่ระบบ แล้วโปรแกรมจะทำการบันทึก IP Address และ ID ของเว็บเบราเซอร์ของผู้ใช้ในฐานข้อมูล เพื่อใช้เป็นข้อมูลในการส่งข่าวสารกลับไปยังเครื่องคอมพิวเตอร์นั้นได้โดยตรงในภายหลัง และเพื่อเป็นตรวจสอบดูจำนวนผู้ใช้งานโปรแกรมได้อีกวิธีหนึ่งด้วย ซึ่ง IP Address และ ID ของเว็บเบราเซอร์ดังกล่าวโปรแกรมจะทำการลบออกจากฐานข้อมูล เมื่อผู้ใช้งานทำการ logout ออกจากระบบหรือทำการปิดเว็บเบราเซอร์ หลังจากนั้นผู้ใช้ก็สามารถที่จะทำการตรวจหาข้อมูลสถานที่ตั้งของอุปกรณ์เครือข่าย ในเชิงลำดับชั้นได้ เช่น อุปกรณ์เครือข่ายอยู่ที่อาคารไหน ชั้นไหน และห้องไหน นอกจากการแสดงข้อมูลรูปแบบของข้อความแล้ว โปรแกรมยังสามารถแสดงข้อมูลสถานที่ตั้งของอุปกรณ์เครือข่ายที่เราสนใจ ในรูปแบบของกราฟฟิก 3 มิติได้ด้วย โดยจะแสดงออบเจ็กต์ของสถานที่ตั้งอุปกรณ์เครือข่ายที่เราตรวจสอบอยู่นั้นให้อยู่ในรูปแบบที่แตกต่างไปจากออบเจ็กต์อื่นๆ เช่น ปกติออบเจ็กต์ของอาคารทั่วๆ ไป จะแสดงด้วยรูปทรงสี่เหลี่ยม 3 มิติสีเหลือง แต่ถ้าเป็นอาคารสถานที่ตั้งของอุปกรณ์เครือข่ายที่เราสนใจจะถูกแสดงด้วยรูปทรงสี่เหลี่ยม 3 มิติสีแดง เป็นต้น

ข.1 ระบบที่ต้องการ

เนื่องจากโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ถูกพัฒนาขึ้นภายใต้สภาวะแวดล้อมของวินโดวส์ โดยต้องการเครื่องคอมพิวเตอร์และเครื่องมือที่ใช้ในการพัฒนาดังต่อไปนี้

- เซิร์ฟเวอร์

  คุณสมบัติด้านฮาร์ดแวร์

  CPU : Pentium 300

  RAM : 64 MB

  Hard-Disk : 6 GB

  ระบบปฏิบัติการ (Operating System)

  Windows NT 4.0

- ระบบเว็บเซิร์ฟเวอร์

  Microsoft Internet Information Server 4.0 (IIS 4.0)

- ระบบจัดการฐานข้อมูล (DBMS : Database Management System)

  SQL Server 7.0

- ไคลเอนต์

  คุณสมบัติด้านฮาร์ดแวร์

  CPU : Pentium 133

  RAM : 32 MB

  Hard-Disk : 1.2 GB

  ระบบปฏิบัติการ (Operating System)

  Windows 95

- เว็บเบราเซอร์

  Netscape Communicator 4.0 หรือ Internet Explorer 4.0 ที่มี VRML plug-in เช่น Cosmo Player 2.1

- เครื่องมือที่ใช้พัฒนา

  Borland Delphi 4.0

  โปรแกรมกราฟฟิกทูล 3 มิติ ได้แก่ Cosmo World

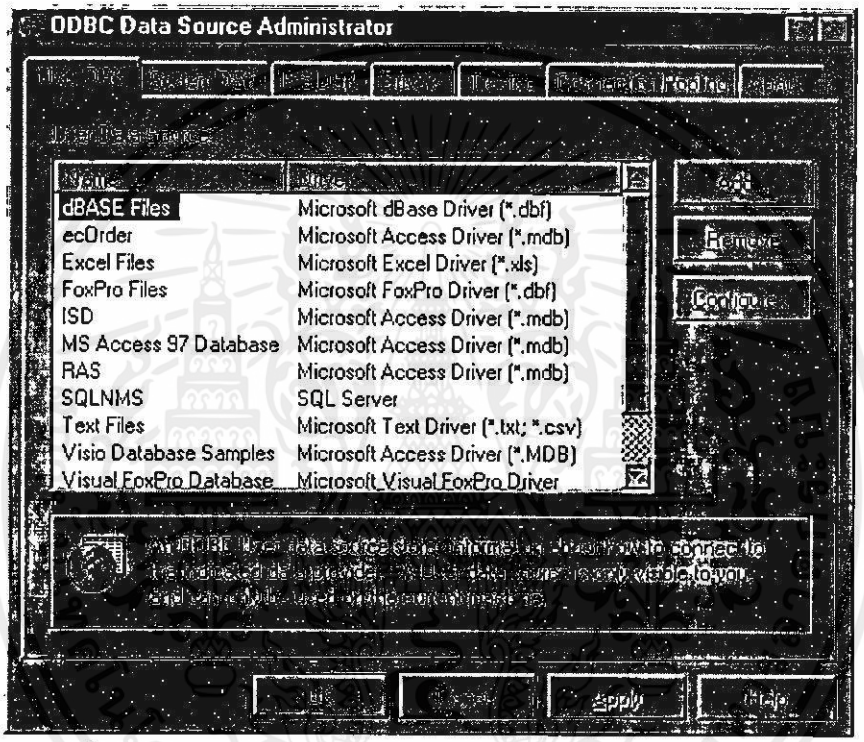
  เท็กซ์เอดิเตอร์ สำหรับเขียน VRML

ข.2 การติดตั้ง

ข.2.1 การติดตั้งค่า ODBC Data Sources

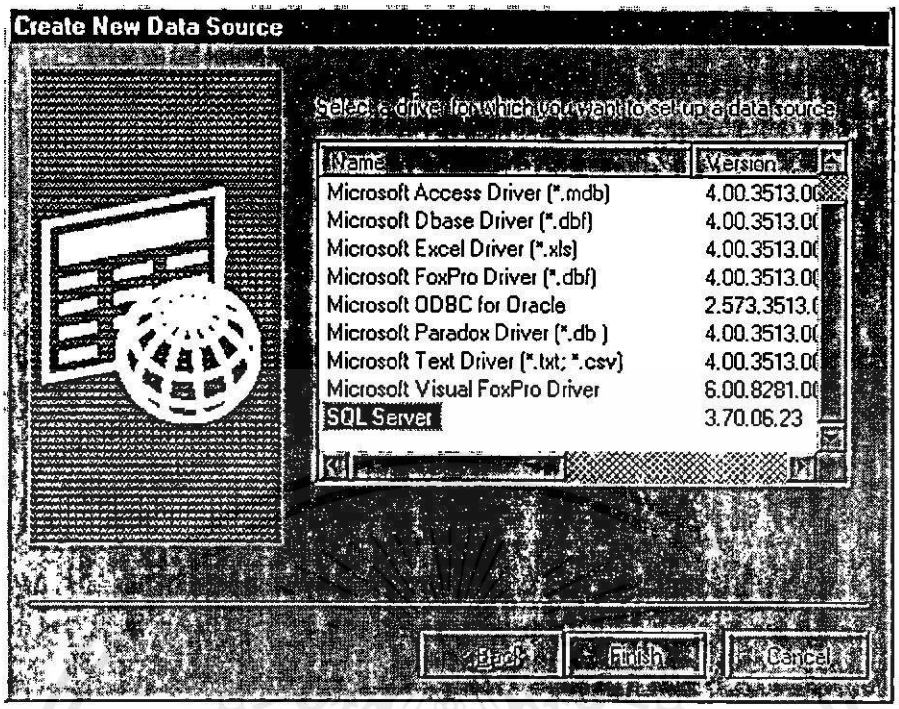
ขั้นตอนในการติดตั้งค่า ODBC Data Sources มีดังนี้

1. เรียกโปรแกรม ODBC Data Source Administrator จากเมนู Start->Settings->Control Panel->ODBC Data Sources (32 bit) จะปรากฏหน้าจอ ODBC Data Source Administrator ขึ้นมาดังรูปที่ ข.1



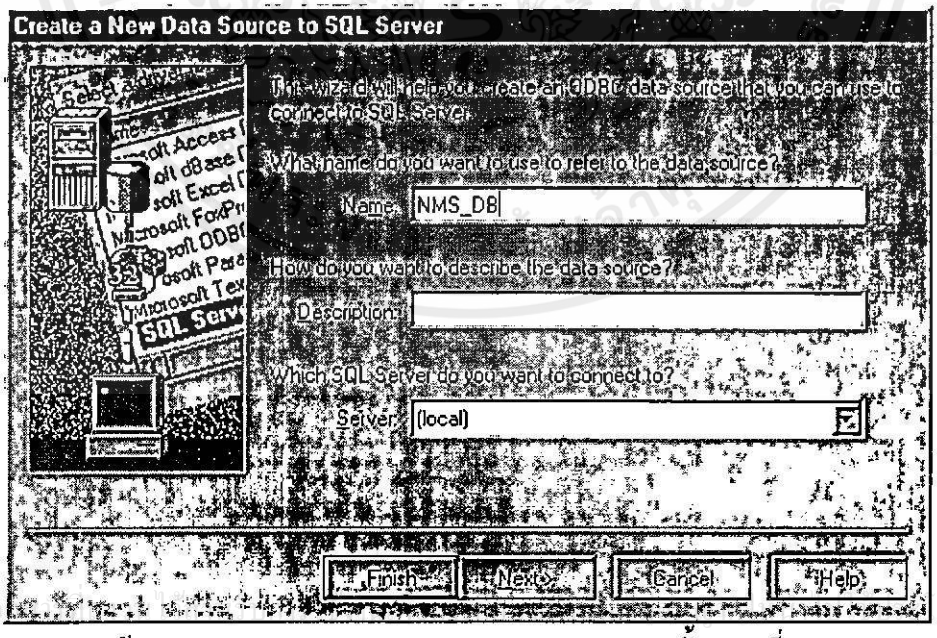
รูปที่ ข.1 แสดงหน้าจอ ODBC Data Source Administrator

2. กดปุ่ม Add จะปรากฏหน้าจอ Create New Data Source ดังรูปที่ ข.2 ให้คลิกเลือก Driver "SQL Server" แล้วกดปุ่ม Finish



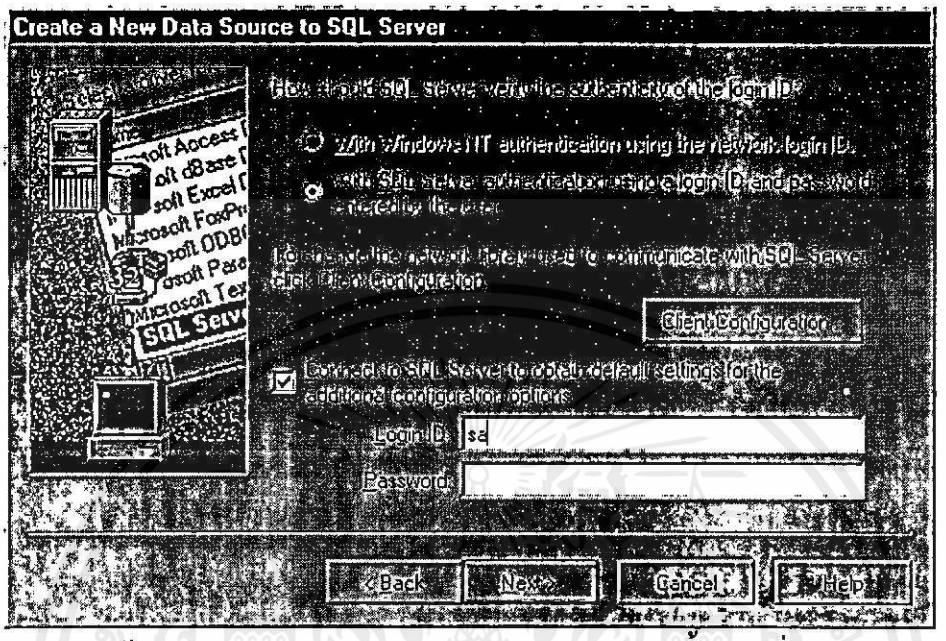
รูปที่ ข.2 แสดงหน้าจอ Create New Data Source

3. หลังจากนั้นจะปรากฏหน้าจอ "Create a New Data Source to SQL Server" ขึ้นมาดังรูปที่ ข.3 ให้ใส่ค่าในช่อง data source Name: เป็น "NMS_DB" และเลือก connect to Server : เป็น "(local)" จากนั้นให้คลิกปุ่ม Next > เพื่อไปสู่ขั้นตอนต่อไป



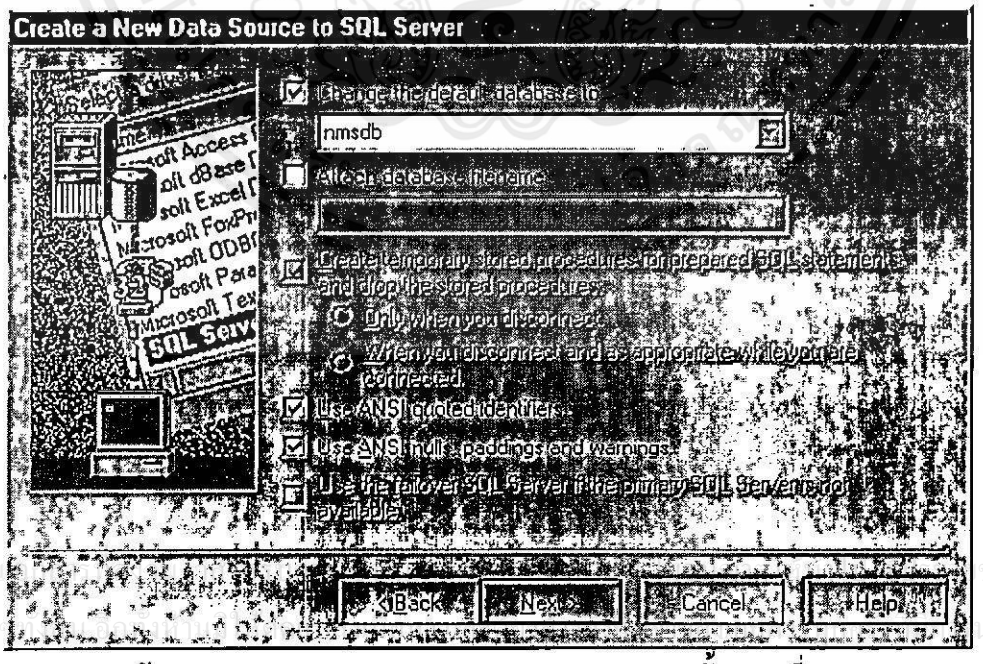
รูปที่ ข.3 แสดงหน้าจอ Create a New Data Source to SQL Server : ขั้นตอนที่ 1

4. เมื่อเข้าสู่ขั้นตอนถัดมาให้ใส่ค่าในช่อง Login ID: เป็น "sa" และในช่อง Password ปล่อยให้เป็นค่าว่างไว้ ดังรูปที่ ข.4 จากคลิกปุ่ม Next > เพื่อเข้าสู่ขั้นตอนต่อไป



รูปที่ ข.4 แสดงหน้าจอ Create a New Data Source to SQL Server : ขั้นตอนที่ 2

5. ถัดมาให้ทำการคลิกเลือก option "Change the default database to:" และทำการเลือก "nmsdb" ดังรูปที่ ข.5 เสร็จแล้วคลิกปุ่ม Next > เพื่อไปยังขั้นตอนต่อไป



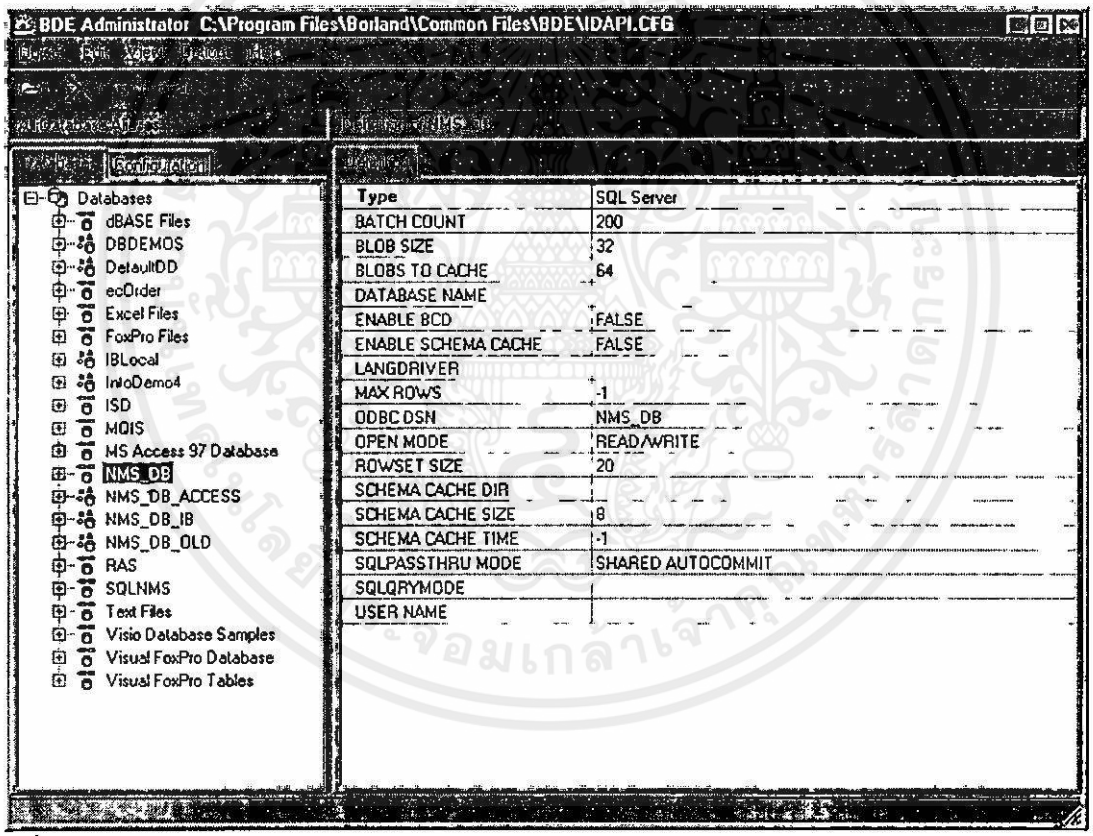
รูปที่ ข.5 แสดงหน้าจอ Create a New Data Source to SQL Server : ขั้นตอนที่ 3

6. ในขั้นตอนสุดท้ายให้ปล่อยค่าเป็น default เอาไว้ และคลิกปุ่ม Finish เป็นอันจบขั้นตอนการสร้าง Data Source

ข.2.2 การตรวจสอบการติดตั้งค่าใน BDE (Borland Database Engine)

ขั้นตอนในการตรวจสอบการติดตั้งค่าใน BDE มีดังต่อไปนี้

1. เรียกโปรแกรม BDE Administrator จาก เมนู Start->Programs->Borland Delphi 4-> BDE Administrator จะปรากฏหน้าจอ BDE Administrator ขึ้นมาดังรูปที่ ข.6 ให้สังเกตว่าจะปรากฏรายชื่อ ODBC Data Source Name ชื่อ "NMS_DB" ที่ได้ทำการติดตั้งไปแล้วขึ้นมาด้วย ซึ่งหมายความว่า BDE นั้นรู้จัก ODBC Data Source ที่เราได้เพิ่มเข้าไปแล้ว



รูปที่ ข.6 แสดงหน้าจอ BDE Administrator

2. เมื่อตรวจสอบว่า BDE รู้จัก ODBC Data Source ถูกต้องเรียบร้อยแล้ว ก็ให้ออกจากการตรวจสอบการติดตั้งค่าใน BDE โดยคลิกที่เมนู Object->Exit

ข.2.3 ระบบไฟล์และตำแหน่งที่ตั้งของระบบไฟล์

สำหรับโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ประกอบไปด้วยไฟล์สำคัญต่างๆ ดังนี้

ตารางที่ ข.1 แสดงไฟล์สำคัญต่างๆ ที่ใช้ในระบบจัดการเครือข่ายผ่านเว็บด้วย VRML

[illegible]	[illegible]	[illegible]
.exe,.dll	โปรแกรมระบบหลัก	<scripts_path>
*.htm	ไฟล์เว็บเพจ	<www_path>
*.class	ไฟล์คลาสของแอพเพล็ต	<www_path>/class
*.jpg	ไฟล์รูปภาพ	<www_path>/images
*.ini	ไฟล์ตั้งค่าใช้งาน	<windows_path>
*.tpl	ไฟล์เทมเพล็ตของ VRML object	<vrmlcode_path>

ข.2.4 รายละเอียดของไฟล์ INI

สำหรับไฟล์ ini ในโปรแกรมระบบการติดต่อผู้ใช้ในระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ชื่อ vnms.ini มีรายละเอียดดังนี้

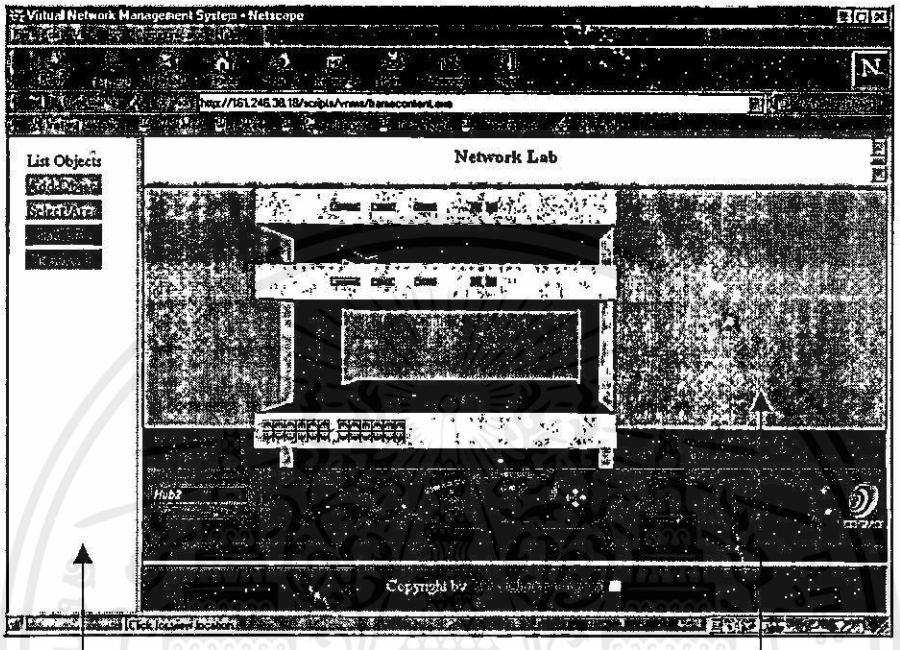
```
[Server]
ServerName=161.246.38.18          //ระบุ IP Address หรือ Domain name ของเซิร์ฟเวอร์

[Path]
ScriptPath=/scripts/vnms          //ระบุ Scripts Path ซึ่งเป็นที่ตั้งของไฟล์ *.exe
WWWPath=/wwwroot/vnms             //ระบุ WWW Path ซึ่งเป็นที่ตั้งของไฟล์ *.htm, class path
                                  และ images path
VRMLCodePath=c:\webshare\scripts\vnms\vrml_template    //ระบุ Path ซึ่งจะเป็นที่จัดเก็บไฟล์
                                                       *.tpl

[System_Param]
Ratio=5          //ระบุอัตราส่วนต่อหนึ่งหน่วยใน VRML ที่ระบบใช้ในการแสดงรูปทรง 3 มิติ
```

ข.3 ลักษณะโดยรวมของระบบ

โปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML แยกการทำงานออกเป็น 2 ส่วน คือ ส่วนเมนูและส่วน Navigate โดยหน้าจอแสดงดังรูปที่ ข.7



รูปที่ ข.7 แสดงหน้าจอการทำงานหลักของโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML

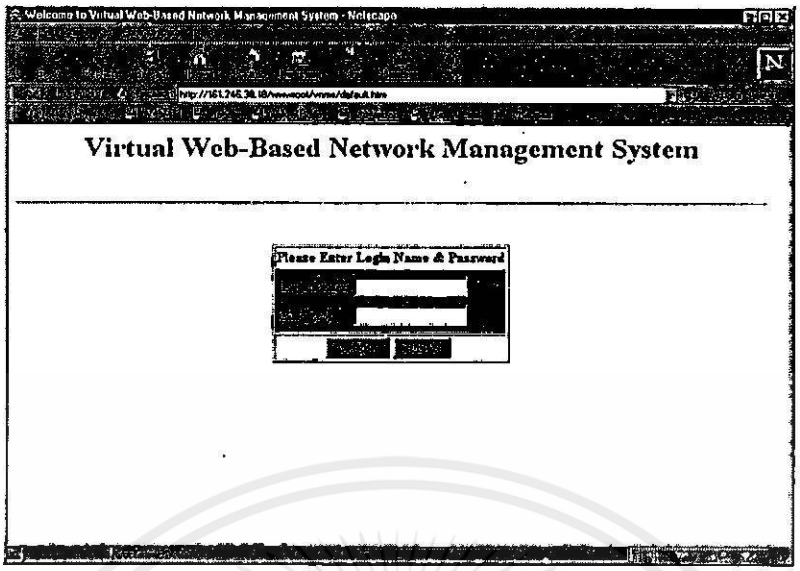
ข.4 การใช้งานระบบ

1. ผู้ใช้เรียกใช้งานระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ผ่านเว็บเบราเซอร์โดยการระบุ URL ไปที่

   http://161.246.38.18/dang/wwwroot/vnms/Default.html

2. การเข้าสู่ระบบ

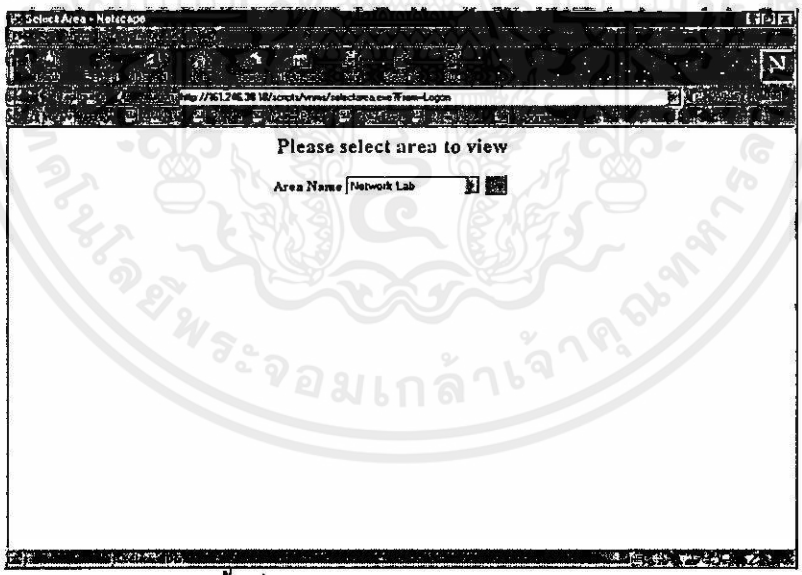
   เมื่อผู้ใช้ทำการระบุ URL ดังข้อที่ 1 แล้ว จะปรากฏหน้าจอ การเข้าสู่ระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML ดังรูปที่ ข.8 โดยผู้ใช้ต้องใส่ Login Name และ Password ที่ถูกต้อง



รูปที่ ข.8 แสดงหน้าจอการ Login

3. การเลือกพื้นที่

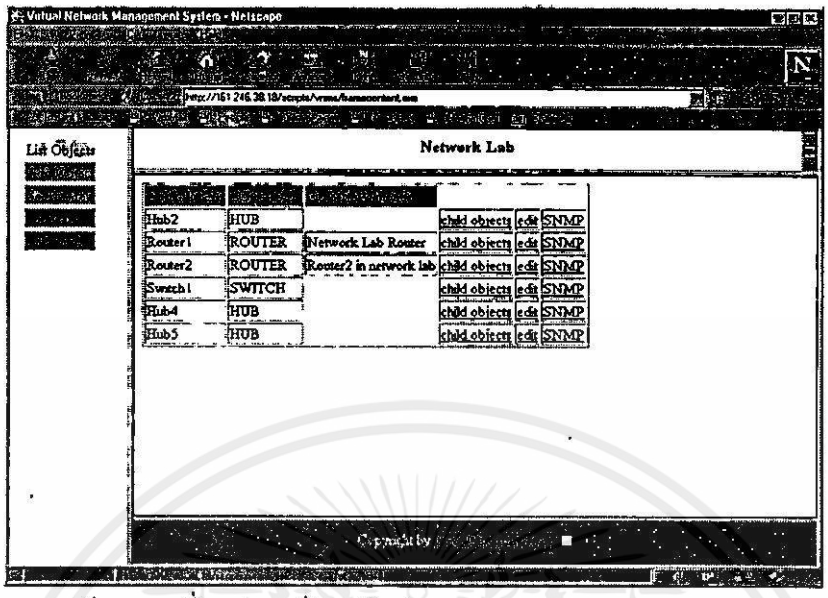
ถ้าผู้ใช้ใส่ Login Name และ Password ถูกต้อง ผู้ใช้จะพบกับหน้าจอถัดไป ซึ่งจะเป็นหน้าจอให้ผู้ใช้ทำการเลือกพื้นที่ในการเข้าไปทำงาน ดังรูปที่ ข.9



รูปที่ ข.9 แสดงหน้าจอการเลือกพื้นที่

4. การแสดงรายการอุปกรณ์ในเครือข่าย

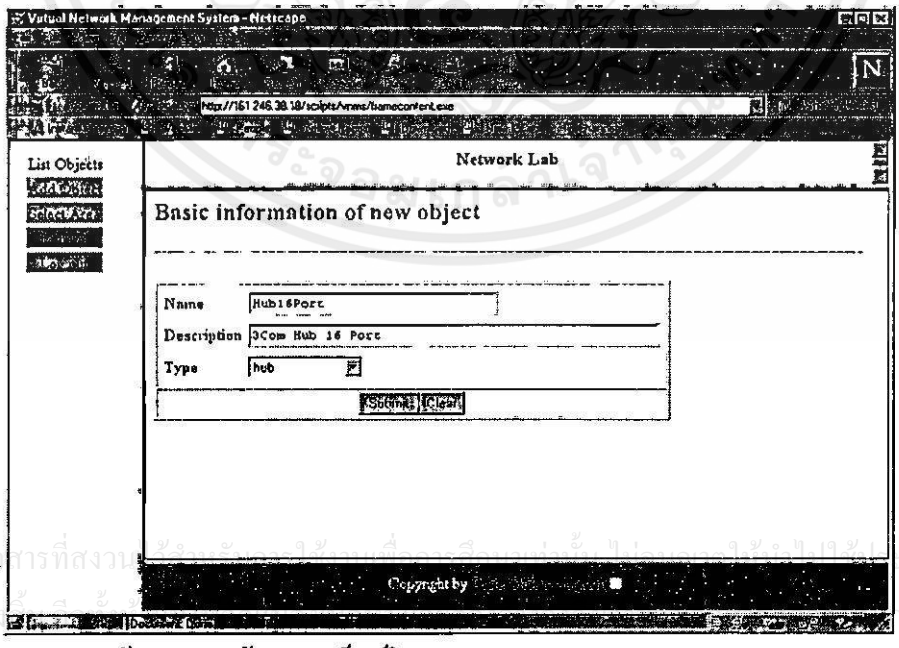
ผู้ใช้สามารถที่จะเรียกดูรายการอุปกรณ์ในเครือข่าย ซึ่งอยู่ภายในพื้นที่ที่ผู้ใช้กำลังเข้าทำงานอยู่ได้ โดยการคลิกเลือกเมนู List Objects ด้านซ้ายมือดังรูปที่ ข.10 โปรแกรมก็จะทำการแสดงรายการอุปกรณ์ทั้งหมดออกมา



รูปที่ ข.10 แสดงหน้าจอรายชื่ออุปกรณ์ในเครือข่าย

5. การเพิ่มอุปกรณ์ในระบบเครือข่าย

ผู้ใช้สามารถเพิ่มข้อมูลอุปกรณ์ในระบบเครือข่ายได้ โดยทำการคลิกเมนู Add Objects ด้านซ้ายมือดังรูปรูปที่ ข.11 โปรแกรมจะแสดงแบบฟอร์มสำหรับการกรอกข้อมูลรายละเอียดอุปกรณ์ที่ต้องการเพิ่ม จากนั้นผู้ใช้ก็สามารถกรอกข้อมูลรายละเอียดของอุปกรณ์เข้าไปและทำตามขั้นตอนที่โปรแกรมแนะนำ จนกระทั่งเสร็จสิ้นขั้นตอน

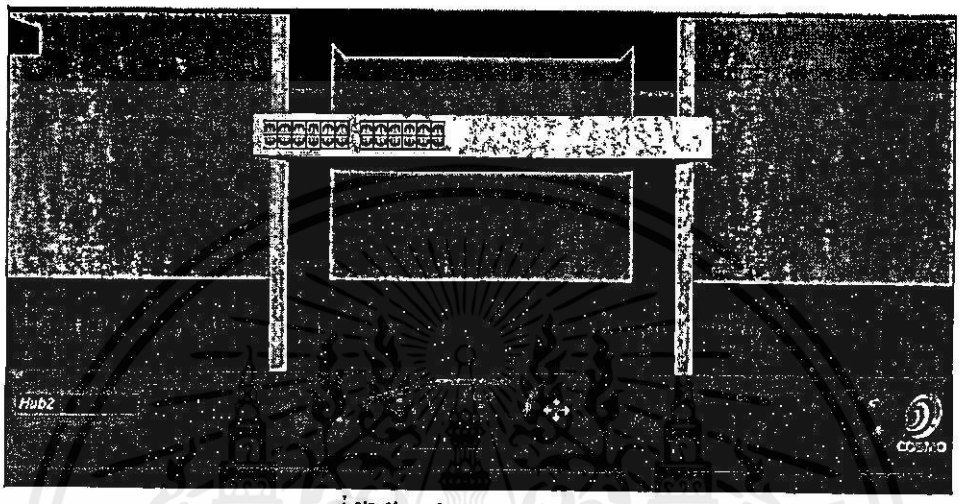


รูปที่ ข.11 แสดงหน้าจอการสร้างออบเจ็กต์ในระบบ

6. การตรวจสอบตำแหน่งที่ตั้งทางกายภาพของอุปกรณ์เครือข่าย

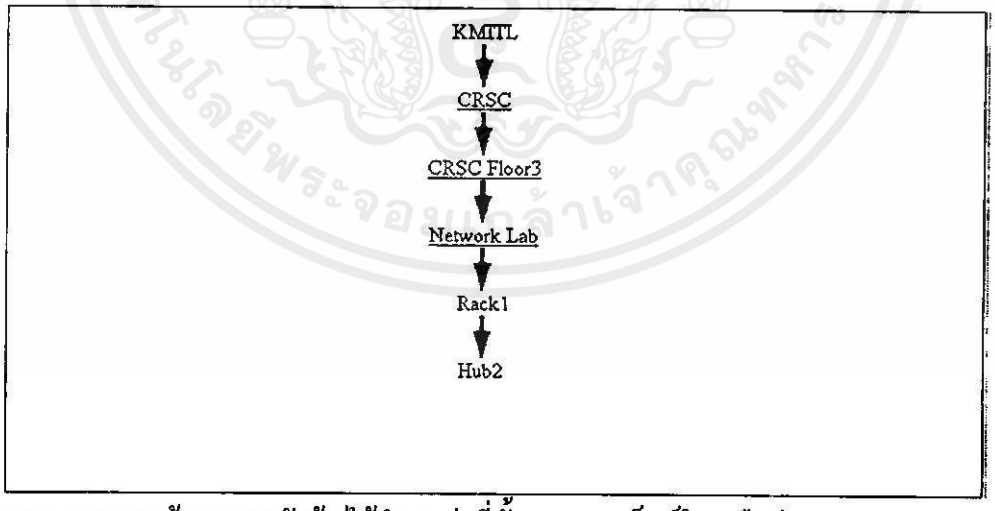
ผู้ใช้สามารถตรวจสอบตำแหน่งที่ตั้งทางกายภาพของอุปกรณ์เครือข่ายได้ ดังขั้นตอนต่อไปนี้

- ผู้ใช้ทำการ Navigateไปหน้าอุปกรณ์ที่สนใจดังรูปที่ ข.12



รูปที่ ข.12 แสดงภาพการ Navigate ที่ผู้ใช้สนใจ

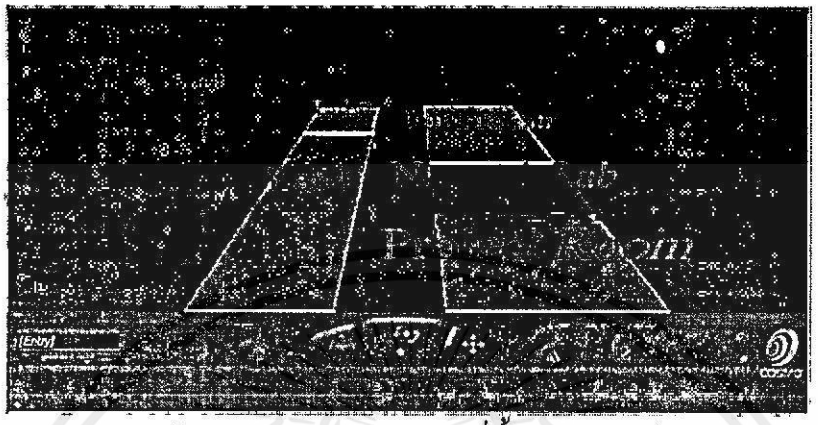
- ผู้ใช้ทำการคลิกไปที่ตัวอุปกรณ์นั้น โปรแกรมจะแสดงข้อมูลตำแหน่งที่ตั้งทางกายภาพของอุปกรณ์นั้น ในลักษณะของแผนผังโครงสร้างต้นไม้ ดังรูปที่ ข.13



รูปที่ ข.13 แสดงหน้าจอแผนผังต้นไม้ตำแหน่งที่ตั้งของออบเจ็กต์ในเครือข่าย

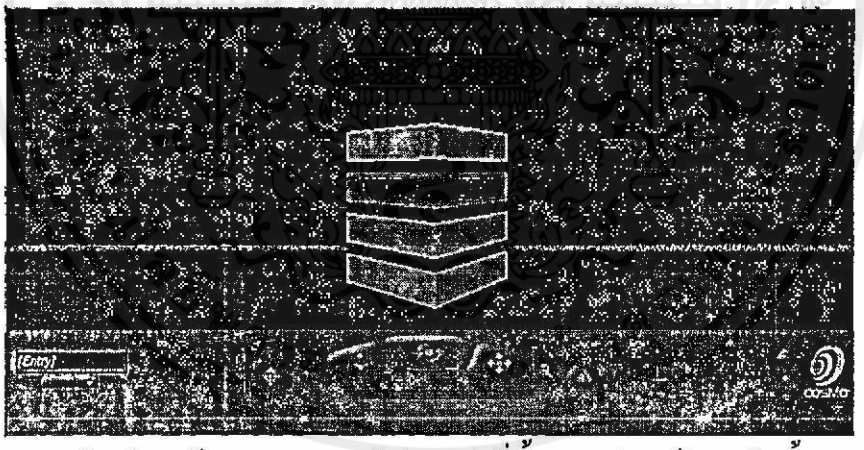
- ผู้ใช้สามารถดูตำแหน่งที่ตั้งของสถานที่ต่างๆ ในแผนผังต้นไม้นี้ได้ โดยการคลิกเลือกชื่อสถานที่ ต่อจากนั้นโปรแกรมจะแสดงตำแหน่งที่ตั้งของสถานที่ดังกล่าวในรูปแบบของ VRML โดยสถานที่ดังกล่าวจะถูกแสดงด้วยสีแดง เพื่อ

เป็นเครื่องหมายแสดงให้ผู้ใช้ทราบถึงบริเวณซึ่งเป็นสถานที่ตั้งทางกายภาพของอุปกรณ์ที่สนใจ ตัวอย่างเช่น เมื่อคลิกที่ชื่อ Network Lab โปรแกรมจะแสดงภาพดังรูปที่ ข.14



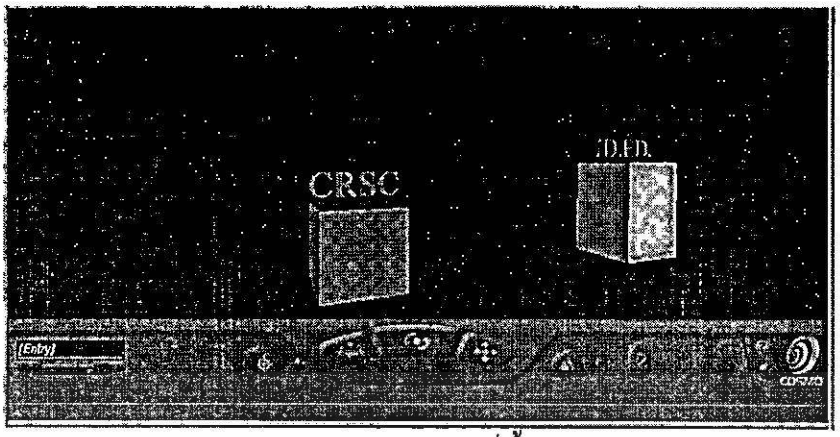
รูปที่ ข.14 แสดงตัวอย่างหน้าจอแสดงแผนผังตำแหน่งที่ตั้งของอุปกรณ์ในระดับห้อง

- ถ้าผู้ใช้คลิกที่ชื่อ CRSC Floor3 ในแผนผังต้นไม้ โปรแกรมจะแสดงดังรูปที่ ข.15



รูปที่ ข.15 แสดงตัวอย่างหน้าจอแสดงแผนผังตำแหน่งที่ตั้งของอุปกรณ์ในระดับชั้น

- ถ้าผู้ใช้คลิกที่ชื่อ CRSC ในแผนผังต้นไม้ โปรแกรมจะแสดงดังรูปที่ ข.16



รูปที่ ข.16 แสดงตัวอย่างหน้าจอแสดงแผนผังตำแหน่งที่ตั้งของอุปกรณ์ในระดับอาคาร

7. การเปลี่ยนพื้นที่การทำงาน

ในระหว่างการทำงานกับระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ผู้ใช้สามารถเปลี่ยนหรือย้ายพื้นที่การทำงานจากที่อยู่ปัจจุบันไปยังพื้นที่อื่นได้ โดยการคลิกเลือกเมนู Select Area โปรแกรมจะแสดงหน้าจอให้ทำการเลือกพื้นที่การทำงานใหม่อีกครั้ง

8. การออกจากระบบ

เมื่อผู้ใช้ต้องการออกจากระบบการจัดการเครือข่ายผ่านเว็บด้วย VRML ผู้ใช้สามารถทำได้ โดยการคลิกเลือกเมนู Logout โปรแกรมจะทำการออกจากระบบและกลับไปยังหน้าจอการเข้าสู่ระบบใหม่อีกครั้ง

จากรายละเอียดที่กล่าวมาข้างต้น คือ วิธีการใช้งานโปรแกรมระบบการติดต่อผู้ใช้ในการจัดการเครือข่ายผ่านเว็บด้วย VRML เพื่อใช้ในการจัดการเครือข่าย ซึ่งสามารถที่จะนำเสนอรายละเอียดของข้อมูลที่เป็นประโยชน์ในการจัดการเครือข่าย ในเรื่องของตำแหน่งที่ตั้งจริงทางกายภาพของอุปกรณ์ได้เป็นอย่างดี รวมถึงสามารถตรวจดูสถานะปัจจุบันของระบบเครือข่ายและสภาพแวดล้อมทางกายภาพได้

ประวัติผู้เขียน

นางสาวแดง ชล่ไพรพิมลรัตน์ เกิดวันที่ 11 กันยายน 2518 ที่จังหวัดกรุงเทพมหานคร

ประวัติการศึกษา

- ศึกษามัธยมปลาย โรงเรียนศึกษานารี กรุงเทพมหานคร ปีการศึกษา 2535
- ศึกษาปริญญาตรี สาขาวิทยาการคอมพิวเตอร์ มหาวิทยาลัยหอการค้าไทย ปีการศึกษา 2539

ประวัติการทำงาน

- ธันวาคม/2539-มิถุนายน/2540 ตำแหน่งนักพัฒนาโปรแกรม ฝ่ายคอมพิวเตอร์ บริษัทเงินทุนหลักทรัพย์นวธนกิจ จำกัด (มหาชน)
- ธันวาคม/2542-มิถุนายน/2543 ตำแหน่งนักพัฒนาโปรแกรมอาวุโส ฝ่ายพัฒนาระบบงาน บริษัท อาร์ ไอ เอส จำกัด
- กรกฎาคม/2543-ปัจจุบัน ตำแหน่งที่ปรึกษา บริษัท ออราเคิล ซิสเต็ม (ประเทศไทย) จำกัด