

โปรแกรมจำลองเสมือนจริง
VIRTUAL CENTER POINT



เลขหมู่.....
เลขทะเบียน..... 46123
วัน, เดือน, ปี 20 ส.ค. 2546

.b.....
.i.....

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาคามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมจำลองเสมือนจริง

VIRTUAL CENTER POINT

ผู้จัดทำ

- | | | |
|----------------|-----------------|-----------------------|
| 1. นายโมฆิต | ชุมทรัพย์ | รหัสประจำตัว 41014059 |
| 2. นายวิสิทธิ์ | เดชะเลิศถาวรกุล | รหัสประจำตัว 41014393 |
| 3. นายแสงฤกษ์ | งามไกวต์ | รหัสประจำตัว 41014495 |



อาจารย์ที่ปรึกษา

(อาจารย์เกียรติคุณ เจียรนัยระนงกิจ)



โปรแกรมจำลองเสมือนจริง

นายโมษิต ขุมทรัพย์ 41014059
 นายวิสิทธิ์ เศรษฐกิจวารกุล 41014393
 นายแสงฤทธิ์ งามไกวด์ 41014495
 เกียรติกุล เจียรนัยชนะกิจ อาจารย์ที่ปรึกษา
 ปีการศึกษา 2544

บทคัดย่อ

โครงการโปรแกรมจำลองเสมือนจริง เป็นโครงการที่กล่าวถึงการเขียนโปรแกรมเกม 3 มิติที่สามารถพูดคุยกันผ่านเครือข่ายได้ โดยใช้ไคลเอนต์และ มอร์ฟิเทนจิน เป็นเครื่องมือช่วยในการพัฒนาส่วนของโปรแกรม 3 มิติ และใช้โปรแกรมจาวาในการพัฒนาส่วนของเน็ตเวิร์ค ไคลเอนต์เป็นเทคโนโลยีที่ถูกสร้างขึ้นมาเพื่อช่วยให้ผู้พัฒนาซอฟต์แวร์มีความสามารถของอุปกรณ์ฮาร์ดแวร์ต่างๆ มาใช้อย่างมีประสิทธิภาพและเกิดประโยชน์สูงสุด ส่วน มอร์ฟิเทนจิน เป็น แอปพลิเคชันโปรแกรมอินเทอร์เน็ตที่ใช้ในการพัฒนาโปรแกรม 3 มิติ

ขอบเขตของโครงการนี้โครงการนี้เป็นการสร้างโปรแกรมพูดคุยสื่อสารผ่านเครือข่าย โดยนำเสนอในรูปแบบที่เป็นเกม 3 มิติ บนระบบปฏิบัติการวินโดวส์ โดยจะใช้มอร์ฟิเทนจิน เป็นหลักในการพัฒนากราฟฟิก 3 มิติ และใช้ไคลเอนต์ 8 เป็นเครื่องมือช่วยเขียนโปรแกรม เริ่มจากการสร้างโมเดลจำลองฉากในโปรแกรมโดยใช้ ทรีดี สตูดิโอ แมกซ์ สร้างตัวละครโดยใช้ มัลล์เซฟ ทรีดี แล้วใช้มอร์ฟิเทนจินในการควบคุมวัตถุ 3 มิติต่างๆรวมถึงการควบคุมลำดับเหตุการณ์ที่เกิดขึ้นภายในโปรแกรม และการตอบสนองผู้เล่นโดยใช้ไคลเอนต์และใช้ ภาษาจาวาในการทำแอปพลิเคชันต่างๆบนเครือข่าย

Virtual Center Point

Kosit Khumsap

Wisit Techalertavornkul

Sangreak Ngamgaiwan

Kietikul Jearanitanakij Advisor

Abstract

3D Game Development is the project that describes 3D game programming that uses DirectX SDK (Software Development Kit) and Morfit engine. And JAVA Development Kit 1.3 is produced to provide service for network application. DirectX technology is produced to provide service for software developer to get the hardware optimize performance. Morfit engine is application program interface for develop 3D application.

This thesis is concerned with 3D graphic and rest using DirectX 8 as a programming tool. First, build a scene using 3D Studio MAX. Then control 3D objects in the scene and game event through Morfit engine and response user input via through DirectInput. Finally using JAVA as a programming tool for network application.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้เกิดขึ้นได้เนื่องมาจากการทำงานร่วมกันในหลายๆ ส่วน หลายๆ คน ทั้งผู้ที่ตั้งใจช่วยและมิได้ตั้งใจหรือช่วยอย่างไม่รู้ตัว โดยเฉพาะอย่างยิ่งการอำนวยความสะดวกในด้านต่างๆ ของภาควิชาวิศวกรรมศาสตรคอมพิวเตอร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังแห่งนี้ ที่ทั้งสละเครื่องคอมพิวเตอร์ที่มีอยู่อย่างจำกัดในห้องภาคมาเป็นเครื่องทำโครงการ และเจ้าหน้าที่ประสานงานต่างๆ ตั้งแต่ เจ้าหน้าที่ห้องธุรการ, เจ้าหน้าที่ห้องปฏิบัติการ และพี่ๆ เพื่อนๆ ที่ได้ร่วมมือเป็นอย่างดี ทั้งอำนวยความสะดวกในการทำงานเลยเวลาราชการ, การตอบข้อซักถามต่างๆ ฯลฯ ที่ไม่สามารถบรรยายได้หมด ยังผลให้โครงการนี้ได้ทำงานอย่างต่อเนื่องมาตลอด และสำเร็จลุล่วงไปได้ด้วยดี

ที่จะขาดเสียมิได้เลยในการทำโครงการนี้ก็คือ อาจารย์เกียรติคุณ เจียรนัยธนกิจ อาจารย์ที่ปรึกษาที่ได้คิดหัวข้อโครงการนี้ขึ้นมา ซึ่งเป็นประโยชน์อย่างมากในการที่จะนำออกไปใช้เมื่อจบและทำงานแล้ว และยังเป็นผู้ซึ่งให้คำปรึกษาในเรื่องต่างๆ อีกด้วย รวมไปถึงอาจารย์ท่านอื่นๆ ที่มีได้กล่าวถึงที่ได้คอยแนะนำ และให้คำปรึกษากลายความข้องใจ ตลอดจนเพื่อนๆ และรุ่นพี่ที่ได้ให้คำแนะนำต่างๆ ในการทำโครงการนี้

โมฆิต ชุมทรัพย์

วิศิษฐ์ เศรษฐ์เลิศวารกุล

แสงฤกษ์ งามไฉน

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญตาราง	VIII
สารบัญภาพ	IX
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของงานโครงการ	1
1.3 ขอบเขตในการดำเนินงาน	2
1.4 วิธีการดำเนินงาน	2
1.5 สาเหตุที่เลือกกราฟฟิคเอนจินในการพัฒนา	2
1.6 สาเหตุที่เลือกเกมเอนจินมอร์ฟิท (Morfit) ในการพัฒนา	3
1.7 ปัญหาที่คาดว่าจะพบ	4
บทที่ 2 ไคลเอนต์/เซิร์ฟเวอร์	5
2.1 รูปแบบของไคลเอนต์/เซิร์ฟเวอร์	5
2.2 โครงสร้างพื้นฐานของ Client/Server	6
2.2.1 Client	6
2.2.2 Middleware	6
2.2.3 Server	9
บทที่ 3 ภาษาจาวา	11
3.1 ลักษณะของภาษาจาวา	11
3.2 ลักษณะการเขียนโปรแกรมด้วยภาษาจาวา	12
3.2.1 คลาส (Classes)	12
3.2.2 เมธอด (Method)	13
3.2.3 การซ่อนรายละเอียดข้อมูล (Data Encapsulation)	13
3.2.4 การสืบทอดคุณสมบัติ (Inheritance)	13
3.3 ประเภทของโปรแกรมจาวา	13
3.3.1 จาวาแอปพลิเคชัน (Java Application)	13
3.3.2 จาวาแอปเพล็ต (Java Applets)	13

สารบัญ
(ต่อ)

	หน้าที่
บทที่ 4 จาวา เนทีฟ อินเทอร์เฟซ	15
4.1 การเขียนเนทีฟเมธอด	17
บทที่ 5 พื้นฐานมอร์ฟิท	19
5.1 โคออดิเนต 2 มิติ	19
5.1.1 World Space	19
5.1.2 Object Space	19
5.2 ระบบโคออดิเนต 3 มิติ	20
5.2.1 การบวกเวกเตอร์	21
5.2.2 การลบเวกเตอร์	22
5.2.3 ระนาบ	22
5.3 แฮนเดิล (HANDLES)	22
5.4 ภาพรวมของเกมเอนจินมอร์ฟิท	23
บทที่ 6 Object	25
6.1 Viewer Mode และ Dynamic Objects	25
6.2 การดึงชื่อและการเข้าใช้วัตถุ	25
6.3 การเคลื่อนย้ายวัตถุ	25
6.3.1 การควบคุมโดยตรง	25
6.3.2 การกำหนดให้วัตถุเคลื่อนที่ไปตามเส้นทาง (Track)	26
6.3.3 การกำหนดให้เคลื่อนที่ไปตามวัตถุ	26
6.3.4 การกำหนดให้วัตถุเคลื่อนที่ตามกฎฟิสิกส์	27
6.4 ตัวอย่างการสร้างวัตถุ	28
6.5 การตรวจสอบการชน (Collision Detection)	28
6.6 กล้อง (Camera)	29
6.6.1 การดึงชื่อ Camera และ handles	29
6.6.2 ตำแหน่งของกล้อง	29
6.6.3 ทิศทางของกล้อง	30
6.6.4 การซูม (Zoom)	30
6.6.5 การควบคุมคุณภาพ	30
บทที่ 7 โปรแกรมสร้างโมเดล 3 มิติ Milkshape 3D	31
7.1 Milkshape 3D	31
7.2 การสร้างรูปโมเดล 3 มิติ	32

สารบัญ
(ต่อ)

	หน้าที่
7.3 การกำหนดพื้นผิว(Texture Mapping)	32
7.4 การกำหนดกระดูกและข้อต่อให้โมเดล 3 มิติ	33
7.5 การจัดการแอนิเมชันของโมเดล 3 มิติ	34
7.6 การเอ็กซ์พอร์ตเพื่อนำไปใช้กับมอร์ฟิฟเอนจิน	35
บทที่ 8 การสร้างโมเดล 3 มิติ	37
8.1 หลักการสร้างกราฟิก 3 มิติด้วยโปรแกรม 3D Studio Max	37
8.2 โครงสร้างและหน้าจอการทำงานภายใน โปรแกรม 3D Studio Max	37
8.2.1 Menu bar	37
8.2.2 Tab Panel	37
8.2.3 Command Panel	38
8.2.4 Status Windows	38
8.2.5 หน้าจอย่อย (Viewport)	38
8.3 ส่วนควบคุมหน้าจอย่อย	38
8.4 การ Merge ไฟล์	39
8.5 การ Import / Export ไฟล์	39
8.6 การ Convert ไฟล์เป็น WLD	40
8.7 การสร้างวัตถุ 3 มิติรูปทรงพื้นฐาน	40
8.7.1 สร้างวัตถุโดยใช้เมาส์ในจอภาพย่อย	40
8.7.2 สร้างกล่องโดยการกำหนดค่าจากคีย์บอร์ด	41
8.8 การกำหนดพื้นผิววัตถุ	41
บทที่ 9 การออกแบบโปรแกรมจำลองเสมือนจริง	43
9.1 ภาพรวมของโครงการ	43
9.2 การพัฒนาโปรแกรมประยุกต์ โคลเอนต์ / เซิร์ฟเวอร์	43
9.2.1 ส่วนของโคลเอนต์	43
9.2.2 ส่วนของเซิร์ฟเวอร์	44
9.3 การพัฒนาโปรแกรมกราฟิก 3 มิติ	44
9.3.1 โปรแกรม 3D Studio Max	44
9.3.2 โปรแกรม Milkshape 3D	44
9.4 การนำโปรแกรมในส่วน 3 มิติและส่วนเน็ตเวิร์กมารวมกัน	45
9.5 แผนภาพต่างๆของการออกแบบระบบ (Diagram)	46

สารบัญ
(ต่อ)

	หน้าที่
บทที่ 10 ปัญหาและการแก้ไข	54
10.1 การใช้งานร่วมกันของภาษาจาวาและซี	54
10.2 ปัญหาเนื่องจากพอลีกอนและผู้เล่นเยอะเกินไป	54
10.3 ขีดจำกัดของจำนวนสีของบิตแมปที่ใช้เป็นฉากหลัง	54
10.4 การเข้าถึงแอนิเมชันของโมเดลเอ็มดีทู (MD2)	55
10.5 ข้อบกพร่องของการตรวจสอบการชนโดยใช้ฟังก์ชันของมอร์ฟิท	55
บทที่ 11 ผลลัพธ์ของโครงการงาน	56
11.1 รูปแบบของโครงการงาน	56
11.2 วิธีการใช้งาน	56
บทที่ 12 บทวิจารณ์และสรุป	62
12.1 ประเมินผล	62
12.2 แนวทางการพัฒนาต่อ	62
12.3 สรุป	62
บรรณานุกรม	63

สารบัญตาราง

หน้าที่

ตารางที่ 7.1 แสดงคำสั่งในไฟล์ md2.qc

36



สารบัญรูปภาพ

	หน้าที่
รูปที่ 2.1 แสดงโครงสร้างระดับล่างของ Transport stack	8
รูปที่ 2.2 แสดงการใช้ Protocol TCP/IP ในระบบไคลเอนต์/เซิร์ฟเวอร์	9
รูปที่ 3.1 การทำงานของคอมไพเลอร์และตัวแปลในภาษาจาวา	12
รูปที่ 4.1 แสดงให้เห็นถึงการที่โปรแกรมที่เขียนด้วยภาษาซี ใช้เจเอ็นไอในการเชื่อมต่อกับจาวาไลบรารี การเรียกใช้จาวาเมธอด การใช้จาวาคลาส	16
รูปที่ 4.2 อธิบายถึงการเรียกใช้ฟังก์ชันภาษาซีที่พาสจาวาแอปพลิเคชัน	16
รูปที่ 4.3 แสดงให้เห็นถึงความสัมพันธ์ของแอปพลิเคชันภาษาซีและจาวา	16
รูปที่ 4.4 แสดงให้เห็นถึงขั้นตอนของ โปรแกรม HelloWorld	18
รูปที่ 5.1 แกน x-y ของระบบ โคออดิเนต 2 มิติ	19
รูปที่ 5.2 ระบบ โคออดิเนตของวัตถุเทียบกับ โลก (World Space)	19
รูปที่ 5.3 Object Space	20
รูปที่ 5.4 ระบบ โคออดิเนต 3 มิติตามกฎมือขวา	21
รูปที่ 5.5 ระบบ โคออดิเนต 3 มิติตามกฎมือขวาเทียบกับวัตถุ	21
รูปที่ 7.1 แสดงหน้าจอการทำงานของ โปรแกรม Milkshape 3D	31
รูปที่ 7.2 แสดงการทำโมเดล 3 มิติ โดยการนำรูปสี่เหลี่ยมมาต่อเป็นรูปร่าง	32
รูปที่ 7.3 แสดงการปรับแต่งขนาดให้ได้รูปตามต้องการ	32
รูปที่ 7.4 แสดงส่วนที่ใช้ในการกำหนด material ใน Milkshape 3D	33
รูปที่ 7.5 แสดงเครื่องมือ Texture Coordinate Editor และการกำหนด โคออดิเนตให้พื้นผิว	33
รูปที่ 7.6 แสดงการกำหนดข้อต่อและกระดูกใน Milkshape 3D	34
รูปที่ 7.7 ส่วนที่ใช้ในการทำแอนิเมชันใน Milkshape 3D	34
รูปที่ 7.8 ตัวอย่าง ไฟล์ md2.qc	35
รูปที่ 8.1 แสดงหน้าจอการทำงานภายในโปรแกรม 3D Studio Max	37
รูปที่ 8.2 แสดงส่วนควบคุมหน้าจอย่อย	38
รูปที่ 8.3 แสดงสร้างวัตถุโดยการ ใช้เมาส์ในจอภาพย่อย	40
รูปที่ 8.4 แสดงแถบคำสั่ง Rollout หัวข้อ Parameters	41
รูปที่ 8.5 แสดงคุณสมบัติเบื้องต้นของวัตถุ	41
รูปที่ 8.6 แสดงการกำหนดคุณสมบัติ	42
รูปที่ 9.1 แสดงยูสเครสไดอะแกรมของระบบ	46
รูปที่ 9.2 แสดงซีเควนไดอะแกรมของการสมัครเป็นสมาชิก	47
รูปที่ 9.3 แสดงซีเควนไดอะแกรมของการล็อกอินของผู้ใช้	47

สารบัญรูปภาพ

(ต่อ)

	หน้าที่
รูปที่ 9.4 แสดงซีเควณโคอะแกรมของการควบคุมตัวละคร	48
รูปที่ 9.5 แสดงซีเควณโคอะแกรมของการสนทนาผ่านทางเครือข่ายของผู้ใช้	48
รูปที่ 9.6 แสดงซีเควณโคอะแกรมของการออกจากระบบของผู้ใช้	49
รูปที่ 9.7 แสดงซีเควณโคอะแกรมของการกระจายข่าวของผู้ควบคุมระบบ	49
รูปที่ 9.8 แสดงซีเควณโคอะแกรมของการตรวจสอบประสิทธิภาพระบบของผู้ควบคุมระบบ	50
รูปที่ 9.9 แสดงซีเควณโคอะแกรมของการดูแลรักษาระบบของผู้ควบคุมระบบ	50
รูปที่ 9.10 แสดงซีเควณโคอะแกรมของการสำรองข้อมูลผู้ใช้ของผู้ควบคุมระบบ	51
รูปที่ 9.11 แสดงซีเควณโคอะแกรมของการเปลี่ยนแปลงข้อมูลตัวละครของผู้ควบคุมระบบ	51
รูปที่ 9.12 แสดงซีเควณโคอะแกรมของการเปลี่ยนแปลงสิ่งแวดล้อมของผู้ควบคุมระบบ	52
รูปที่ 9.13 แสดงคลาสโคอะแกรมของโปรแกรมจำลองเสมือนจริง (Virtual Center Point)	53
รูปที่ 11.1 แสดงหน้าจอแรกเมื่อเริ่มเปิดโปรแกรม	56
รูปที่ 11.2 หน้าจอสำหรับลงทะเบียนขอใช้บริการ	57
รูปที่ 11.3 หน้าจอสำหรับเลือกตัวละครของผู้ใช้	58
รูปที่ 11.4 แสดงหน้าจอเลือกสีเสื้อผ้าตัวละคร	58
รูปที่ 11.5 แสดงหน้าต่างการพูดคุย	59
รูปที่ 11.6 แสดงการแจ้งข้อความเมื่อมีผู้เล่นคนอื่นเข้ามา	59
รูปที่ 11.7 หน้าต่างแสดงตัวละครบนโลกจำลอง	59
รูปที่ 11.8 หน้าต่างการพูดคุยสำหรับผู้ควบคุมระบบ	60
รูปที่ 11.9 หน้าต่างการเลือกป้ายโฆษณา	61
รูปที่ 11.10 ผลเมื่อเลือกเปลี่ยนป้ายโฆษณา	61

บทที่ 1

บทนำ

1.1 ความสำคัญและที่มา

การพัฒนาโปรแกรมในปัจจุบันนี้ โปรแกรมประยุกต์แบบโคลเอนต์/เซิร์ฟเวอร์ เป็นอีกทางเลือกหนึ่งที่ได้รับความนิยมเป็นอย่างมากเพราะเป็นสถาปัตยกรรมที่มีความสามารถสูง และใช้งานได้อย่างง่าย มีการแบ่งหน้าที่การทำงานอย่างเป็นระบบ และเนื่องจากเป็นโปรแกรมที่ใช้งานบนเครือข่าย ซึ่งในปัจจุบันเป็นเรื่องง่ายที่จะใช้ เนื่องจากมีเครือข่ายขนาดใหญ่ เช่น เครือข่ายอินเทอร์เน็ตเข้ามารองรับอยู่แล้ว จึงทำให้การพัฒนาโดยใช้สถาปัตยกรรมโคลเอนต์/เซิร์ฟเวอร์นี้ น่าจะเป็นทางเลือกที่ดีที่สุดในทางหนึ่งในการพัฒนาโปรแกรมประยุกต์ขึ้นมาสักโปรแกรมหนึ่งทีเดียว

ความเป็นจริงเสมือน (Virtual reality) เป็นเทคโนโลยีที่ทำให้ผู้ใช้มีความรู้สึกเสมือนกับได้ทำงานอย่างใดอย่างหนึ่ง โดยจะทำการตอบสนองผู้ใช้โดยอาศัยหลักการพื้นฐานของประสาทสัมผัสทั้ง 5 การนำแนวคิดของความเป็นจริงเสมือน ที่มีการแสดงผลเป็นรูปภาพฟิก 3 มิติที่สามารถตอบสนองประสาทสัมผัสทางตาได้ดีในระดับหนึ่งมาผสมผสานกับสถาปัตยกรรมโคลเอนต์/เซิร์ฟเวอร์ทำให้โปรแกรมที่ใช้งานผ่านทางเครือข่ายเน็ตเวิร์คมีความน่าสนใจยิ่งขึ้น

สำหรับโครงการนี้เป็นการประยุกต์โปรแกรมสนทนาผ่านทางเครือข่ายเน็ตเวิร์คในรูปแบบของกราฟฟิก 3 มิติ โดยจะเปรียบเสมือนให้ผู้สนทนาแต่ละคนเป็นตัวละครตัวหนึ่งที่สามารถเคลื่อนไหวอยู่ในสิ่งแวดล้อมที่เราได้สร้างขึ้นมาในรูปแบบของกราฟฟิก 3 มิติ และเมื่อมีการเคลื่อนไหวไปพบตัวละครตัวอื่นๆ ก็สามารถที่จะทำการสนทนากันได้ และผู้ใช้อย่างยังสามารถดูแผนที่เพื่อทำการค้นหาผู้ใช้คนอื่นๆ ได้

สำหรับโครงการนี้จะทำการแยกพัฒนาออกเป็น 2 ส่วน คือ ส่วนของสถาปัตยกรรมโคลเอนต์/เซิร์ฟเวอร์ซึ่งจะใช้ภาษา ภาษาจาวา ในการติดต่อผ่านทางเครือข่ายเน็ตเวิร์ค และในส่วนของกราฟฟิก 3 มิติซึ่งในโครงการนี้ได้เลือกใช้โปรแกรมสามดี สตูดิโอ แมกซ์ เวอร์ชัน 4.0 (3D Studio Max 4.0) การสร้างโมเดลต่างๆ ให้มีความเสมือนจริง รวมทั้งใช้โปรแกรมมิลล์เชฟ (MilkShape) ในการสร้างแอนิเมชันในรูปแบบต่างๆ ให้มีความสวยงามอีกด้วย และยังได้นำโปรแกรมมอร์ฟิท (Morfit) ซึ่งเป็น เอนจิน มาใช้ในการควบคุมเกี่ยวกับกราฟฟิกรวมทั้งติดต่อกับ ไดเรกซ์เอกซ์ (DirectX) ทำให้ภาพที่ได้ออกมาอย่างมีประสิทธิภาพ

1.2 วัตถุประสงค์ของงานโครงการ

1. เพื่อศึกษาการพัฒนาโปรแกรมประยุกต์แบบโคลเอนต์/เซิร์ฟเวอร์
2. เพื่อศึกษาเทคโนโลยีต่างๆ ของการพัฒนาโปรแกรมโดยใช้จาวา
3. เพื่อศึกษาเทคโนโลยีในการสร้างภาพ 3 มิติ
4. เพื่อนำความรู้ที่ได้ศึกษามาประยุกต์ใช้ในการพัฒนาโปรแกรม 3 มิติ

5. เพื่อศึกษาขั้นตอนต่างๆของการเขียนและพัฒนาซอฟต์แวร์ ตั้งแต่การวิเคราะห์ความต้องการ การออกแบบ การเขียน และการทดสอบระบบ

1.3 ขอบเขตในการดำเนินงาน

โครงการนี้จะเป็นการสร้างโปรแกรมสนทนาผ่านทางเครือข่ายเน็ตเวิร์ก ซึ่งจะนำเสนอในรูปแบบกราฟฟิก 3 มิติ โดยจะสร้าง โลก (World) ในลักษณะ 3 มิติเพื่อเป็นสถานที่ในการดำเนินการสนทนา ซึ่งจะกำหนดให้ผู้ใช้แต่ละท่านทำการเลือกตัวละครที่กำหนดให้ และยังสามารถเลือกชุดแต่งกายของแต่ละตัวละครได้ จากนั้นจะทำการจัดให้ตัวละครที่ได้เลือกไว้เข้าไปอยู่ภายใน โลก ทางฝ่ายผู้ใช้งานสามารถที่จะ ควบคุม ตัวละครให้กระทำการกิจกรรมต่างๆ ภายใน โลก และเมื่อมีการพบตัวละครอื่นๆ ก็สามารถเลือกที่จะสนทนากับบุคคลอื่นๆ ได้ นอกจากนี้ยังมีแผนที่เพื่อช่วยในการทราบตำแหน่งของตนเองและค้นหาตำแหน่งของบุคคลอื่นๆ ได้ และภายใน โลก นี้จะมีการคิดแผนป้ายโฆษณาไว้ตามจุดต่างๆ ซึ่งแผนป้ายโฆษณานี้สามารถปรับเปลี่ยนรูปแบบของโฆษณาได้ ซึ่งจุดนี้ต่อไปจะสามารถหารายได้ให้แก่ผู้พัฒนาโปรแกรมได้ นอกจากนี้ในการเข้ามาใช้งานครั้งแรกของผู้ใช้แต่ละท่าน จะมีการสอบถามข้อมูลต่างๆของผู้ใช้เพื่อนำไปเก็บไว้ยังเครื่อง เซิร์ฟเวอร์

1.4 วิธีการดำเนินงาน

1. ศึกษาการทำงานของระบบ โคลเอนต์/เซิร์ฟเวอร์
2. ศึกษาเทคโนโลยีต่างๆ ของจาวา
3. ศึกษาโปรแกรมประยุกต์ 3 มิติ เช่น โปรแกรมสามดี สตูดิโอ แมกซ์ เวอร์ชัน 4.0 , มัลติเซฟ, มอร์ฟิท
4. วิเคราะห์และออกแบบระบบ รวมถึงกำหนดขอบเขตในการทำงานของโปรแกรมประยุกต์
5. กำหนดและเลือกเครื่องมือที่จะนำพัฒนาโปรแกรมประยุกต์
6. เริ่มเขียนโปรแกรมโดยจะแบ่งโครงการออกเป็น 2 ส่วน คือ ส่วนของการติดต่อผ่านทางเครือข่ายเน็ตเวิร์ก ซึ่งจะใช้ภาษา ภาษาจาวา และส่วนของการพัฒนากราฟฟิก 3 มิติซึ่งจะใช้โปรแกรมประยุกต์ 3 มิติ
7. ทำการทดสอบโปรแกรมทั้ง 2 ส่วน
8. ทำการรวมโปรแกรมทั้งสองส่วนให้สามารถทำงานร่วมกันได้
9. ทำการทดสอบโปรแกรมที่เขียนทั้งหมด
10. แก้ไขส่วนที่ผิดพลาดให้มีการใช้งานที่ถูกต้อง

1.5 สาเหตุที่เลือกกราฟฟิกเอนจินในการพัฒนา

ในปัจจุบันการผลิตแอปพลิเคชันขนาดกลางถึงใหญ่ หรือแม้กระทั่งขนาดเล็กก็ตามที่ นิยมใช้เฟรมเวิร์ค หรือ เอพีไอ เข้ามาช่วยในการผลิต เพื่อลดความซับซ้อนของโค้ด และ ทำให้สามารถผลิตเกมออกมาได้มากขึ้นโดยใช้ทรัพยากรน้อยลง และ เนื่องจากเกมเป็นแอปพลิเคชันรูปแบบหนึ่ง จึงมีหลาย

บริษัทที่ผลิตเฟรมเวิร์คสำหรับการพัฒนาเกมออกมา โดยเฟรมเวิร์คสำหรับพัฒนาเกมนี้ถูกเรียกกันในหมู่นักพัฒนาเกมว่า เกมเอนจิน

1. มาตรฐานของซอฟต์แวร์ และ ฮาร์ดแวร์มีการเปลี่ยนแปลงอย่างไม่หยุดยั้ง ทั้งการเร่งการแสดงผล 3 มิติ และ เอฟีไอ ต่างก็พัฒนากันอย่างรวดเร็ว ในทุกๆไตรมาส ถ้าไม่มีการ์ดรุ่นใหม่ออกมา ก็มีการปล่อยเอฟีไอใหม่ๆ หรือเวอร์ชันใหม่ๆออกมา การที่จะทำเกมให้สนับสนุน เอฟีไอ และการแสดงผล 3 มิติ ให้มากที่สุด นั้นต้องใช้เวลามากเกินไป ทำให้ผู้พัฒนามีเวลาที่จะไปพัฒนาตัวเนื้อเกมน้อยลง
2. เวลาที่ใช้ในการศึกษา เทคโนโลยีที่คืนนอกจากจะต้องตอบสนองความต้องการของเราได้แล้ว ควรจะง่ายต่อการทำความเข้าใจด้วย เพราะการที่จะเข้าใจและเป็นผู้เชี่ยวชาญเอฟีไอ ได้นั้นจำเป็นต้องใช้เวลา โดยเฉพาะอย่างยิ่งถ้าจะศึกษา เอฟีไอ ทั้งหมดที่มีอยู่ในท้องตลาดจนเชี่ยวชาญย่อมเป็นไปได้ยาก
3. เวลาที่ใช้ในการผลิต ถึงแม้คุณจะเป็นผู้เชี่ยวชาญ เอฟีไอนั้นก็ติดตาม แต่ถ้ามการสร้างซอฟต์แวร์โดยใช้เอฟีไอนั้นใช้เวลามากและต้องเขียนทุกๆส่วนขึ้นมาใหม่ทุกครั้ง ทั้งๆที่ควรจะนำกลับมาใช้ได้ ทำให้เสียเวลาในส่วนนั้นโดยเปล่าประโยชน์ ในการสร้างเกมก็เช่นกัน การตรวจสอบการชนกันของวัตถุ, การบริการหน่วยความจำสำหรับเท็กซ์เจอร์ (texture) และ ซอฟต์แวร์เรนเดอร์ (Software Render) ล้วนเป็นสิ่งที่ผู้พัฒนาทำอยู่แล้ว การพัฒนาขึ้นจากศูนย์ใหม่นอกจากจะเสียเวลาแล้ว ยังอาจทำไม่ได้เท่าของที่มีอยู่เดิมด้วย
4. ความสามารถ เอฟีไอบางชนิดที่ทำหน้าที่ติดต่อกับฮาร์ดแวร์เป็นหลัก เช่น ไดรেকซ์เอกซ์ อาร์ เอ็ม (Retain Mode) มีประสิทธิภาพต่ำ และ ไม่สามารถจัดการกับโลก 3 มิติ ที่มีขนาดใหญ่มาๆได้ และ ไดรেকซ์เอกซ์ ไอเอ็ม ที่มีประสิทธิภาพสูงกว่าแต่ไม่สนับสนุนการทำแอนิเมชัน โมเดล 3 มิติ
5. เครื่องมือ เอนจินที่ดีมักจะมีเครื่องมือช่วยพัฒนาเป็น (GUI หรือ Graphic ผู้ใช้งาน Interface) มาด้วยทำให้ลดเวลาในการพัฒนาลงไปได้มาก เช่น การ โมเดลลิ่งโดยใช้ออโต้แคด (AutoCAD) ซึ่งต้องวาดทุกอย่างขึ้นมาทั้งหมด

1.6 สาเหตุที่เลือกเกมเอนจินมอร์ฟิท (MorfIt) ในการพัฒนา

1. เกมเอนจินมอร์ฟิทมีทีมงาน และ เว็บไซต์สนับสนุนที่จะให้ความช่วยเหลือ เมื่อเกิดปัญหาในการใช้เกมเอนจิน เราสามารถปรึกษาหาวิธีแก้ไขจากทางมอร์ฟิท หรือ ขอความช่วยเหลือจากผู้ใช้รายอื่นที่เคยพบปัญหาเดียวกัน
2. มีผู้ใช้แพร่หลายอยู่ในระดับหนึ่ง นอกจากจะทำให้วางใจในระดับหนึ่งเนื่องจากถ้ามอร์ฟิทไม่ใช้เอนจินที่ดีในระดับที่น่าพอใจ ก็ไม่น่าจะมีผู้ใช้มาก เมื่อเอนจินเกิดข้อผิดพลาดผู้ใช้ก็สามารถรายงานไปยังผู้ผลิต ฉะนั้นจึงหมายความว่าผู้ตรวจสอบ และ รายงานข้อผิดพลาดไปยังผู้ผลิต เพื่อให้ผู้ผลิตได้แก้ไขข้อผิดพลาดนั้น ทำให้เอนจินสมบูรณ์ขึ้น

3. สนับสนุนและสามารถเข้าถึงข้อมูลแอนิเมชันของโมเดล 3 มิติ เอ็มดี2 (.md2) ซึ่งเป็นโมเดล 3 มิติ ที่นิยมใช้ในเกม 3 มิติบนพีซี โดยโมเดล 3 มิติชนิดนี้สามารถสร้างได้โดย 3 มิติ โมเดลลิงทูต ชื่อ มิลค์เชฟ 3 มิติ (Milkshape 3 มิติ) ซึ่งมีราคาถูก ใช้งานง่าย

1.7 ปัญหาที่คาดว่าจะพบ

1. ไม่สามารถแก้ไขปัญหบางอย่างได้ หากปัญหาที่เกิดขึ้นเกิดมีสาเหตุมาจากข้อจำกัด หรือ ข้อผิดพลาดของเกมเอนจิน เนื่องจากมอร์ฟิทเปิดเผยแต่เพียงอินเตอร์เฟซของเอพีไอเท่านั้น
2. เกมที่พัฒนามีข้อจำกัดหลายด้าน เนื่องจากข้อจำกัดบางประการของมอร์ฟิท เช่น สนับสนุนสีของเท็กซ์เจอร์ 8 บิตเท่านั้น, ไม่สนับสนุนโมเดลเอ็มดีทรี (.md3) ซึ่งเป็นโมเดลที่ใช้กันอยู่ในปัจจุบัน (เช่น เกม Quake 3), ไม่สามารถเรียกใช้คุณสมบัติเฉพาะของการ์ด 3 มิติได้
3. ประสิทธิภาพของเกมต่ำ เนื่องจากสาเหตุหลายอย่าง ไม่ว่าจะเกิดจากการขาดประสบการณ์ของผู้พัฒนา หรือ เนื่องจากเกมเอนจินมอร์ฟิทมีประสิทธิภาพต่ำ
4. วัตถุ 3 มิติในเกมดูไม่เหมือนจริง, ไม่สวยงาม หรือ การเคลื่อนไหวของวัตถุไม่สมจริง เนื่องจากผู้พัฒนาไม่ได้เป็นผู้เชี่ยวชาญในการสร้างโมเดล 3 มิติ

บทที่ 2

ไคลเอนต์/เซิร์ฟเวอร์

Client/Server

จากอดีตการเราใช้ระบบแบ่งปันเวลา (Time Sharing) ซึ่งมีเครื่องเมนเฟรมเป็นโฮสต์ในการจัดการงานทุกอย่าง ทั้งเก็บข้อมูล ประมวลผล การคำนวณต่างๆ โดยที่คัมพิวเตอร์มีหน้าที่เพียงแค่แสดงผลลัพธ์ของข้อมูลเท่านั้น ต่อมาเมื่อมีการออกแบบระบบเครือข่ายท้องถิ่น (LAN) มีเซิร์ฟเวอร์อยู่ 1 ตัว ทำหน้าที่เป็นไฟล์เซิร์ฟเวอร์คอยเก็บข้อมูลและแอปพลิเคชันไว้ จึงต้องมีฮาร์ดดิสก์ความจุหลายจีเกไบต์อยู่บนไฟล์เซิร์ฟเวอร์เครื่องนี้ เพราะเครื่องไคลเอนต์ไม่มีฮาร์ดดิสก์ จะมีเพียงแค่อุปกรณ์สำหรับใส่แผ่นบูตเข้าสู่ระบบเน็ตเวิร์กหรือก๊อปปี้ข้อมูลเท่านั้น การทำงานส่วนใหญ่จะยังอยู่ที่เซิร์ฟเวอร์ เมื่อมีผู้เข้ามาขอใช้ไฟล์และแอปพลิเคชันมากขึ้นและไฟล์มีขนาดใหญ่ขึ้นก็จะทำให้ประสิทธิภาพการทำงานของไฟล์และแอปพลิเคชันลดลงทันที

ระบบไคลเอนต์/เซิร์ฟเวอร์ได้ถูกพัฒนาขึ้นมาเพื่อตอบสนองแนวความคิดการดาวน์โหลด (Downsizing) ให้มีประสิทธิภาพและค่าใช้จ่ายที่ต่ำกว่าระบบแบ่งปันเวลา (Time Sharing) ของเครื่องเมนเฟรม ไคลเอนต์/เซิร์ฟเวอร์ เป็นระบบการประมวลผลแบบกระจาย (Distributed Processing) โดยจะแบ่งการประมวลผลระหว่างเซิร์ฟเวอร์และไฟล์ แทนที่โปรแกรมแอปพลิเคชันจะรันอยู่เฉพาะบนเครื่องเซิร์ฟเวอร์ ก็จะมีการแบ่งการทำงาน การคำนวณของ โปรแกรมแอปพลิเคชันให้มาทำงานบนเครื่องไคลเอนต์ด้วยและเมื่อใดที่เครื่องไคลเอนต์ต้องการผลลัพธ์บางส่วนข้อมูล จะมีการร้องขอไปยังเครื่องเซิร์ฟเวอร์ เพื่อให้ส่งเฉพาะข้อมูลบางส่วนเท่านั้นกลับมาให้เครื่องไคลเอนต์เพื่อคำนวณข้อมูลนั้นอีกทีหนึ่ง อาจกล่าวได้ว่าทศวรรษที่ 90 เป็นระบบการทำงานของไคลเอนต์/เซิร์ฟเวอร์ ซึ่งจะช่วยให้องค์กรต่างๆ ลดค่าใช้จ่ายบำรุงรักษา สำหรับระบบเมนเฟรมและมินิลงได้มาก

2.1 รูปแบบของไคลเอนต์/เซิร์ฟเวอร์ ที่ใช้งานมีอยู่ 4 ชนิดด้วยกันคือ

1. แสตนด์ ออล (Stand alone) ไคลเอนต์/เซิร์ฟเวอร์ การทำงานแบบนี้ผู้ให้บริการหรือเซิร์ฟเวอร์จะอยู่บนเครื่องเดียวกับผู้ขอใช้บริการหรือไคลเอนต์ ทำให้มีความเร็วในการติดต่อสื่อสารระหว่างผู้ให้บริการและผู้ขอใช้บริการสูงมาก แต่ประสิทธิภาพในการประมวลผลระบบฐานข้อมูลจะลดลงบ้าง ระบบนี้เรียกอีกอย่างว่า ไทน์ (Tiny) ไคลเอนต์/เซิร์ฟเวอร์
2. ดีพาร์ตเมนต์ (Department) ไคลเอนต์/เซิร์ฟเวอร์ หรือ ระบบเครือข่ายแบบฐานเดียว (LAN based single) เซิร์ฟเวอร์ การทำงานแบบนี้จะมีผู้ให้บริการเกี่ยวกับฐานข้อมูล แอปพลิเคชัน ฯลฯ อยู่บนเครื่องเซิร์ฟเวอร์ และผู้ขอใช้บริการทั้งหลายจะอยู่บนเครื่องไคลเอนต์ โดยจะเชื่อมต่อกันด้วยระบบเครือข่ายท้องถิ่น (LAN) และมีมิดเดิลแวร์ (Middleware) เป็นตัวกลางที่ทำงานอยู่ระหว่างไคลเอนต์และเซิร์ฟเวอร์ การติดต่อระหว่างผู้ให้บริการและผู้ขอใช้บริการจะช้ากว่าแบบ แสตนด์ ออล เพราะจะต้องติดต่อผ่านระบบเครือข่าย ยิ่งถ้ามีผู้ขอใช้บริการเข้ามาดึงข้อมูลกันครั้ง

ละมากๆ หลายๆ เครื่อง ประสิทธิภาพจะลดลงอย่างเห็นได้ชัด วิธีเพิ่มประสิทธิภาพก็คือการเพิ่มเครื่องเซิร์ฟเวอร์ขึ้นในระบบ

3. เวิร์กกรุ๊ป (Workgroups) โคลเอนต์/เซิร์ฟเวอร์ การทำงานแบบเวิร์กกรุ๊ปนี้จะเป็นกลุ่มของเซิร์ฟเวอร์ที่หลากหลายแพลตฟอร์ม หลายผู้ผลิต มีความแตกต่างกันของเซิร์ฟเวอร์ แต่ทั้งหมดนี้จะเชื่อมต่อกันทางระบบเครือข่าย ระบบเครือข่ายท้องถิ่น และ ระบบเครือข่ายแบบกว้าง (WAN) และใช้มิดเดิลแวร์มาตรฐานในการทำงาน

4. เอ็นเทอร์ไพรส์ (Enterprise) โคลเอนต์/เซิร์ฟเวอร์ การทำงานแบบเอ็นเทอร์ไพรส์หรือระดับองค์กรจะทำให้มีการเชื่อมโยงเครื่องเซิร์ฟเวอร์หรือโฮสต์ต่างแพลตฟอร์มเข้าด้วยกัน ทำให้มีการใช้ทรัพยากรบนระบบได้อย่างมีประสิทธิภาพสูงสุด โดยที่โคลเอนต์สามารถที่จะเลือกใช้ทรัพยากรบนระบบได้อย่างมีประสิทธิภาพสูงสุดโดยที่โคลเอนต์สามารถจะเลือกใช้ทรัพยากร ฐานข้อมูลจากเซิร์ฟเวอร์เครื่องใดก็ได้ผ่านทางมิดเดิลแวร์

2.2 โครงสร้างพื้นฐานของ โคลเอนต์/เซิร์ฟเวอร์

จากการพัฒนาของระบบปฏิบัติการเน็ตเวิร์คทั้งทางค่ายไมโครซอฟท์ คือ วินโดวส์ เอ็นทีเซิร์ฟเวอร์ และ โนเวล คือ เน็ตแวร์ (Netware) ทำให้การทำงานบนระบบเน็ตเวิร์คเป็นไปอย่างสะดวกรวดเร็ว และกว้างขวางขึ้น โดยเฉพาะโปรโตคอลแบบอีเทอร์เน็ต (Ethernet) ที่นิยมใช้ในปัจจุบันซึ่งเหมาะกับระบบโคลเอนต์/เซิร์ฟเวอร์ อีกทั้งระบบปฏิบัติการที่รันอยู่บนเครื่องโคลเอนต์ก็มีความสามารถทางเน็ตเวิร์ค และการทำงานแบบมัลติเทรด วินโดวส์ เอ็นที เซิร์ฟเวอร์ เป็นระบบปฏิบัติการ 32 บิต ที่ทำงานแบบ Preemptive Multithreading ซึ่งจะแย่งงานของ แอปพลิเคชันแต่ละตัวออกจากกันทำให้ไม่มีการรบกวนการทำงานระหว่างกัน เพราะโมเดลในการออกแบบ วินโดวส์ เอ็นที เซิร์ฟเวอร์ ก็เป็นสถาปัตยกรรมโคลเอนต์/เซิร์ฟเวอร์อยู่แล้ว

2.2.1 โคลเอนต์ เป็นส่วนที่จะรันแอปพลิเคชันบนโคลเอนต์โดยใช้ระบบจียูไอ (GUI (Graphical User Interface)) หรือ โอโอยูไอ (OOUI (Object Oriented User Interface)) หรือ ดีเอสเอ็ม (DSM (Distributed System Management)) เป็นการติดต่อกับ ผู้ใช้งาน ผ่านระบบกราฟฟิกซึ่งทำงานแบบเชิงวัตถุ

2.2.2 มิดเดิลแวร์ (Middleware) เป็นส่วนที่ทำงานอยู่ระหว่างโคลเอนต์และเซิร์ฟเวอร์เป็นเสมือนสะพานเชื่อมการทำงานสามารถเล่นออกเป็น 4 แบบ คือ การบริการโดยเฉพาะ (Service Specific), DSM, NOS และ ทรานสปอร์ตสแต็ก (Transport stack)

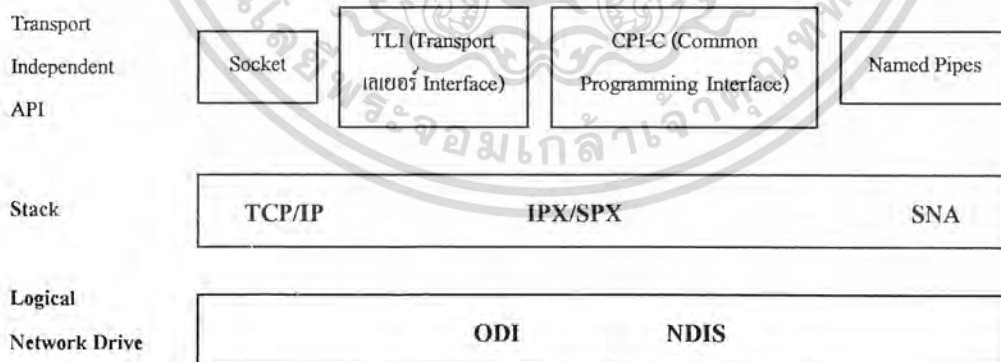
- การบริการ โดยเฉพาะ (Service Specific) จะขึ้นอยู่กับการใช้แอปพลิเคชันในการทำงาน เช่น แอปพลิเคชันของ object แบบกระจายจะใช้ มิดเดิลแวร์ ORB (Object Request Broker) แอปพลิเคชันกรุ๊ปแวร์ จะใช้มิดเดิลแวร์ เมสส์ และ ทรานส์แอคชั่น (TP monitor) จะใช้มิดเดิลแวร์ TxRPC (Transactional Remote Procedure Call) ส่วนระบบฐานข้อมูลซีคิวล (SQL) จะใช้ ODBC (Open Database Connection) DRDA (Distribute Relational Database

Architecture) ของ ไอพีเอ็ม, RDA (Remote Database Access), Oracle Glue, CLI (Call-Level Interface)

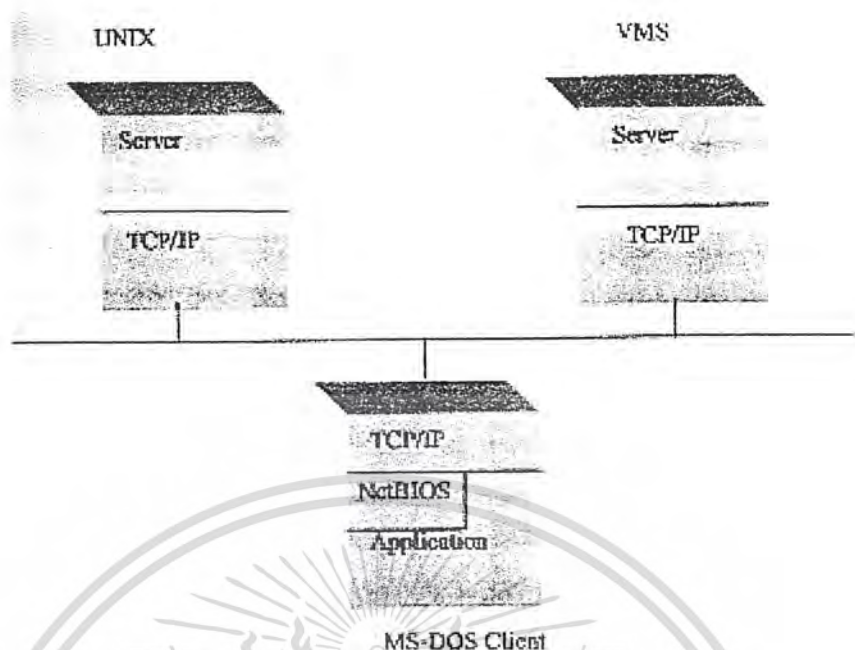
- DSM (Distributed System Management) จะรันบนทุกโหนดของระบบเน็ตเวิร์คที่เป็น ไคลเอนต์/เซิร์ฟเวอร์ จะมีมิดเดิลแวร์ SNMP (Simple Network Management Protocol), CMIP (Common Management Information Protocol) และ DME
- NOS (Network Operation System) เป็นระบบปฏิบัติการเน็ตเวิร์คซึ่งให้บริการต่างๆ ไปโดยจะมีทั้ง Directory Services, Naming Service, Security/Authentication Service, Messaging Service, Distributed file, RPC, Peer to Peer ฯลฯ ระบบปฏิบัติการเหล่านี้เช่น วินโดวส์เอ็นที เซิร์ฟเวอร์, เน็ตแวร์, Banyan Vines, OSF DCE
 - NOS จะช่วยให้การใช้ชื่อ (namespace) เพียงชื่อเดียวสามารถเข้าถึงทรัพยากรต่างๆ บนระบบเน็ตเวิร์ครวมได้
 - NOS จะทำให้ผู้ใช้งาน (User) ไม่ต้องรับรู้เกี่ยวกับความผิดพลาดที่เกิดขึ้น เช่น การรับส่งข้อมูลผิดพลาด ระบบเน็ตเวิร์คมีปัญหาหรือมีการเคลื่อนย้ายทรัพยากรจากไคลเอนต์ไปยังเซิร์ฟเวอร์ NOS จะแก้ไขและอัปเดตข้อมูลต่างๆ ให้เป็นหนึ่งเดียว
 - NOS สามารถใช้รหัสผ่านเพียง 1 ชุด เข้าสู่ระบบเน็ตเวิร์คจากเครื่องใด ที่ไหนก็ได้ โดยจะใช้ระบบรักษาความปลอดภัยแบบ DEC (Distributed Computing Environment) ในการตรวจสอบ
 - NOS จะมีระบบโกลบอลไดเรกทอรี (Global Directory) ซึ่งจะนำคน แอปพลิเคชัน โปรแกรม สิ่งต่างๆ เข้ามาทำงานร่วมกัน ทำให้ไม่ต้องขึ้นกับสถานที่ สามารถจะเปลี่ยนสถานที่ในการใช้ทรัพยากรได้
 - NOS จะจัดการในเรื่องการกระจายเวลา (Distributed time) ให้ทั้งระบบคือจะมีการซิงโครไนซ์ในเรื่องเวลาระหว่าง เซิร์ฟเวอร์และไคลเอนต์ทุกตัว
 - NOS จะจัดการในเรื่องการกระจายความปลอดภัย (Distributed Security) อย่างต่ำอยู่ในระดับ C2 ซึ่งจะต้องมีการแสดงตน (Authentication) มีการเข้ารหัสผ่าน (Encrypt) ใช้มาตรฐาน Kerberos และแอปพลิเคชันเซิร์ฟเวอร์จะมีอำนาจ (Authentication) ในการใช้ ACLs (Access Control List) เพื่อควบคุมการเข้าใช้ทรัพยากรของผู้ใช้งาน
 - NOS สามารถที่จะใช้มิดเดิลแวร์ MOM (Message Oriented Middleware) ในการช่วยจัดคิวข้อความ (Message queue) เพื่อให้ทั้งไคลเอนต์และเซิร์ฟเวอร์ยังคงทำงานได้อย่างต่อเนื่อง แม้จะมีปัญหาทางระบบสื่อสาร ลักษณะนี้อาจเรียกว่า Loosely-Coupled queue based และอีกรูปแบบคือ การใช้ RPCs (Remote Procedure Calls) ซึ่ง NOS เหล่านี้คือ OSF/DCE, ONC/SUN, เน็ตแวร์ 4.xx/โนเวล
- ทรานสปอร์ตสแต็ก (Transport stack) เป็นบริการพื้นฐานในการสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์ บนระบบ LAN และ WAN โพรโตคอล หลักๆ ในส่วนของ Transport stack

มีอยู่ด้วยกัน 4 ตัวด้วยกันคือ NetBIOS, ทีซีพี/ไอพี (TCP/IP), ไอพีเอกซ์/เอสพีเอกซ์ (IPX/SPX) และเอสเอ็นเอ (SNA)

- NetBIOS เป็น โพรโตคอล ที่ออกแบบโดยบริษัท ไอบีเอ็ม ให้ใช้งานกับเครือข่ายขนาดเล็ก ต่อมาพัฒนาเป็น NetBEUI (NetBIOS Extended ผู้ใช้งาน Interface) สามารถจะใช้งานกับระบบเครือข่ายที่มีเครื่องคอมพิวเตอร์ตั้งแต่ 20-200 เครื่อง ไม่สามารถใช้งานกับเครือข่ายขนาดใหญ่ได้ และไม่สามารถค้นหาเส้นทางได้ จะเห็นว่าใช้กับงานเวิร์คกรุ๊ป เช่น วินโดวส์ สำหรับ เวิร์คกรุ๊ป (Microsoft LAN Manager) โพรโตคอล NetBIOS จะทำงานอยู่ในชั้นของเซสชัน เลเยอร์ ตามมาตรฐานโอเอสไอ (OSI-7 Layer)
- ทีซีพี/ไอพี (TCP/IP (Transmission Control โพรโตคอล/ อินเทอร์เน็ต โพรโตคอล)) เป็น โพรโตคอล ที่ใช้งานบนระบบยูนิกซ์ (UNIX) พัฒนาขึ้นในปี 2512 โดยกระทรวงกลาโหมของสหรัฐอเมริกา มีเครือข่ายชื่อ อัลปาเน็ต (ARPANET (Advanced Research Project Agency Network)) สำหรับใช้งานกับเครือข่ายขนาดใหญ่อย่าง ระบบเครือข่ายแบบกว้าง (WANs) มีความสามารถในการค้นหาเส้นทาง และมีความยืดหยุ่นในการทำงานสูง
- ไอพีเอกซ์/เอสพีเอกซ์ (IPX/SPX (Internet work Packet Exchange/Sequenced Packet Exchange)) เป็น โพรโตคอล หลักของระบบปฏิบัติการเน็ตเวิร์ค เน็ตแวร์ มีความฉลาดในการทำงานกว่า NetBIOS คือสามารถค้นหาเส้นทางได้ทำให้ โพรโตคอล ไอพีเอกซ์/เอสพีเอกซ์ สามารถจะทำงานบนระบบเครือข่ายท้องถิ่น และ WAN ได้
- เอสเอ็นเอ (SNA (Systems Network Architecture)) เป็น โพรโตคอล ที่ออกแบบโดยบริษัท ไอบีเอ็มเพื่อใช้งานบนระบบเครือข่ายเครื่องเมนเฟรมของไอบีเอ็ม



รูปที่ 2.1 แสดงโครงสร้างระดับล่างของ Transport stack



รูปที่ 2.2 แสดงการใช้ โปรโตคอล ทีซีพี/ไอพี (TCP/IP) ในระบบ ไคลเอนต์/เซิร์ฟเวอร์

2.2.3 เซิร์ฟเวอร์ เป็นส่วนที่จะรันแอปพลิเคชันในการจัดการทรัพยากรต่างๆ สำหรับระบบ ไคลเอนต์/เซิร์ฟเวอร์สามารถแบ่งออกได้เป็น 4 แบบด้วยกันคือ

- ระบบฐานข้อมูล ซีเควล (DBMS)
 - ระบบจัดการทรานส์แอคชั่น (TP monitor)
 - ระบบกรุปแวร์ (Groupware)
 - ระบบออบเจกต์แบบกระจาย (Distributed Objects)
- ระบบฐานข้อมูล ซีเควล การประมวลผลฐานข้อมูลในระบบไคลเอนต์/เซิร์ฟเวอร์ จะมีสองส่วนคือ ส่วนหลัง และ ส่วนหน้า ส่วนหลัง จะเป็นส่วนของเซิร์ฟเวอร์ซึ่งจะมีระบบฐานข้อมูล ซีเควล เช่น ไมโครซอฟท์ ซีเควล เซิร์ฟเวอร์ ทำหน้าที่เก็บข้อมูล จัดเรียงลำดับ ค้นหา เรียกใช้ ป้องกันข้อมูล ฯลฯ และมีส่วนหน้า มีวินโดว์ส เช่น ซีเควล วินโดว์ส, เดลไฟ (Delphi) ฯลฯ ด้วยเหตุที่ ซีเควล เป็นภาษาในเชิงสอบถาม อธิบายแบบมีโครงสร้าง คำสั่งที่ใช้อ่านเข้าใจง่าย Process การทำงานเริ่มจาก ผู้ใช้งาน ส่งคำสั่งเข้าไปเพื่อขอใช้บริการผู้ให้บริการรับคำสั่งมาทำการประมวลผลเสร็จแล้วส่งผลลัพธ์กลับไปให้ ผู้ใช้งาน ผู้ขอใช้บริการ จะเห็นว่ามีเพียงผลลัพธ์ที่ ผู้ใช้งาน ต้องการเท่านั้นที่ถูกส่งออกไปบนระบบเน็ตเวิร์ค เป็นการลดทราฟฟิก (Traffic) ของระบบลงด้วย มีส่วนของชุดคำสั่งภาษา ซีเควล เรียกว่า สโตร์โปรซีเจอร์ (Stored Procedure) สำหรับจัดการข้อมูล นอกจากนี้ยังมี ทรigger (Triggers), รูล (Rules), วิว (Views) และ สครอลเคอร์เซอร์ (Scroll cursor) ช่วยทำให้ข้อมูลมีความถูกต้องและสมบูรณ์ที่สุด

- ระบบการจัดการแอปพลิเคชัน การทำงานบนเครื่องเมนเฟรมทั้งระบบ จะมีความสลับซับซ้อนของโปรแกรมมากจึงต้องมี ทรานส์แอคชั่น (TP monitor (Transaction Processing monitor)) ซึ่งเป็นระบบติดตามการประมวลผลทรานส์แอคชั่นอยู่ในระบบด้วย ซอฟต์แวร์ประเภท ทรานส์แอคชั่น (TP monitor) ที่นิยมใช้กันคือ CICS (IBM), Tuxedo (BEA) และ Encina สำหรับระบบไคลเอนต์/เซิร์ฟเวอร์ การนำ ทรานส์แอคชั่น มาใช้งานนับว่ามีประโยชน์ และเพิ่มประสิทธิภาพในการประมวลผลแบบกระจาย (Distributed Processing) ทรานส์แอคชั่น จะจัดการกับทรานส์แอคชั่น โดยการจัดเส้นทางเดินในระบบให้จาก ไคลเอนต์ไปยังเซิร์ฟเวอร์ตัวใดตัวหนึ่งบนระบบแล้วกลับมาที่เดิม ถ้าไม่ประสบความสำเร็จก็จะเริ่มต้นทำงานใหม่ งานที่ ทรานส์แอคชั่น ทำคือ การจัดการทรัพยากร และ ผู้ใช้งาน Request จัดการเรื่องของ Two phase commit เก็บ Log ของทรานส์แอคชั่นและทรัพยากรในระบบด้วย สำหรับระบบ NOS ของ วินโดว์ เอ็นที เซิร์ฟเวอร์ ของบริษัทไมโครซอฟท์ได้ออกแบบ MTS (Microsoft Transaction เซิร์ฟเวอร์) ให้ทำงานร่วมกับฐานข้อมูล ไมโครซอฟท์ ซีเควล เซิร์ฟเวอร์ อย่างมีประสิทธิภาพ นอกจากนี้เรายังสามารถใช้ซอฟต์แวร์ ทรานส์แอคชั่น ของเมนเฟรม เช่น Tuxedo ให้มาทำงานบนแพลตฟอร์ม วินโดว์ เอ็นที เซิร์ฟเวอร์ ได้อีกด้วย
- ระบบกรุปแวร์ กรุปแวร์เป็นเทคโนโลยีบนระบบเมนเฟรมซึ่งมีหลากหลายทำงานร่วมกันเช่น ระบบ อิเล็กทรอนิกส์ ระบบเวิร์คโฟล์ ระบบจัดการเอกสารแบบมัลติมีเดีย ระบบจัดการรูปภาพ ระบบจัดการเวลา ระบบการประชุม แต่สามารถนำมาใช้งานกับระบบไคลเอนต์/เซิร์ฟเวอร์ได้ ซอฟต์แวร์ประเภทนี้คือ โดตัสโน้ต (Lotus Notes), DCA OpenMind, Image Plus/2
- ระบบออบเจกต์แบบกระจาย เทคโนโลยีออบเจกต์แบบกระจาย (Distributed Object) ช่วยให้ระบบไคลเอนต์/เซิร์ฟเวอร์แบบนี้ทำหน้าที่ระบบไคลเอนต์/เซิร์ฟเวอร์ทั้ง 3 แบบ (ระบบฐานข้อมูล ซีเควล , ระบบ ทรานส์แอคชั่น และ ระบบกรุปแวร์) ด้วยการรวบรวมขั้นตอนในการทำงานและข้อมูลไว้ในออบเจกต์ซึ่งมันจะเดินทางไปบนระบบเน็ตเวิร์ค (ได้ทุกหนทุกแห่ง) และยังทำงานแบบมัลติแพลตฟอร์มอีกด้วยเช่น AIX, SUN-solaris, HP-UX, IBM-MVS, Digital UNIX Open VMS, ลินุกซ์(Linux) ฯลฯ กลุ่ม โอเอ็มจี(OMG (Object Management Group))ได้ออกแบบออบเจกต์สถาปัตยกรรม COBRA (Common Object Request Broker Architecture) ให้ใช้งานนานร่วม 8 ปี พร้อมทั้งมีบริษัทที่ใช้งาน COBRA อยู่ร่วม 600 บริษัท ทางด้านไมโครซอฟท์ก็ได้พัฒนา ดีคอม(DCOM (Distributed Common Object Model)) ออกมาโดยเริ่มจากแพลตฟอร์มบน วินโดว์ส 95 และ วินโดว์ส เอ็นที นอกจากนี้กำลังจะพอร์ตข้ามไปยัง MVS, UNIX, Solaris, AIX, SCO UNIX, HP-UX, ลินุกซ์(Linux), Open VMS

บทที่ 3

ภาษาจาวา

ภาษาจาวาเริ่มต้นในฐานะงานวิจัยเพื่อพัฒนาซอฟต์แวร์ (Software) ระดับสูงสำหรับอุปกรณ์ของเครือข่ายคอมพิวเตอร์และอุปกรณ์ที่ควบคุมโดยไมโครโปรเซสเซอร์ (Embedded system) ที่มีขนาดเล็กและเชื่อถือได้ ที่สามารถนำไปรันบนแพลตฟอร์ม (Platform) ที่แตกต่างกันได้หรือกระจายส่วนของโปรแกรมไปยังเครื่องต่างๆ อย่างปลอดภัยและประหยัดทรัพยากรของระบบ

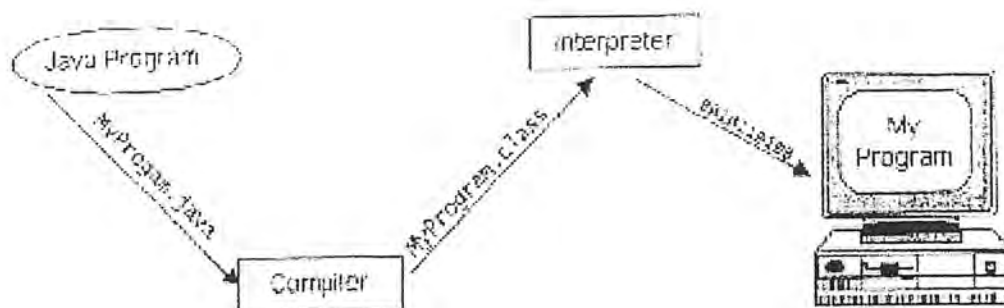
ในตอนเริ่มต้น วิศวกรของภาษาซีถูกเลือกให้เป็นวิศวกรของภาษาใหม่นี้ แต่ต่อมามีปัญหาเกิดขึ้นมากมายต้องออกแบบภาษาใหม่ขึ้นมาโดยเรียกว่าภาษาจาวา ภาษาดังกล่าวมีพื้นฐานมาจากภาษาซี (C), ซีพลัส พลัส (C++), ไอเฟล (Eiffel), สมอลล์ทอล์ค (SmallTalk), ออบเจกต์ทีฟซี (Objective C) และเซดาร์/เมซา (Cedar/Mesa) ภาษาจาวาได้รับการยอมรับว่าเป็นภาษาที่ดีที่สุดในการพัฒนาซอฟต์แวร์ที่ต้องการความปลอดภัยสูงแบบกระจาย (Distributed) ผ่านเน็ตเวิร์ค (network) โดยสามารถใช้พัฒนาได้ตั้งแต่โปรแกรมของอุปกรณ์เน็ตเวิร์คที่ควบคุมโดยไมโครโปรเซสเซอร์ (Network-embedded device) แอปพลิเคชันทั่วไปจนถึงโปรแกรมในเว็บเบราว์เซอร์ ซึ่งก็คือแอปเพลต (Applet)

3.1 ลักษณะของภาษาจาวา

จาวาเป็นภาษาระดับสูงเชิงวัตถุ (Object-Oriented High-Level Language) ซึ่งมีกลไกการแปลภาษาและเอ็กซิคิวต์ในลักษณะผสมผสานระหว่างคอมไพเลอร์ (Compiler) และตัวแปล (Interpreter) เพื่อจุดประสงค์ที่จะรับข้อดีและขจัดข้อเสียของทั้งคอมไพเลอร์และตัวแปล

คอมไพเลอร์เป็นตัวแปลภาษาที่วิเคราะห์โปรแกรมในภาษาระดับสูงเพื่อแปลงเป็นภาษาเครื่อง (Machine code) ซึ่งข้อดีของวิธีนี้คือโปรแกรมภาษาเครื่องจะทำงานเร็วมาก เพราะขั้นตอนในการแปลภาษาถูกแยกออกไปก่อน โปรแกรมทำงาน และคอมไพเลอร์ก็สามารถวิเคราะห์ทั้งโปรแกรมได้ก่อน จึงทำให้สามารถสร้างโปรแกรมภาษาเครื่องที่มีประสิทธิภาพในการทำงานและมีขนาดเล็ก แต่ข้อเสียของคอมไพเลอร์คือความไม่อิสระต่อแพลตฟอร์ม ภาษาเครื่องที่คอมไพเลอร์สร้างขึ้นมานี้จะไม่สามารถนำไปทำงานบนเครื่องที่ต่างแพลตฟอร์มได้

ส่วนตัวแปลเป็นตัวแปลภาษาที่ทำงานโดยอ่านโปรแกรมในภาษาระดับสูงทีละบรรทัด แล้วแปลโปรแกรมบรรทัดนั้นเป็นภาษาเครื่องและทำงานทันที ต่อจากนั้นก็อ่านโปรแกรมบรรทัดต่อไปเข้ามาทำเช่นเดิมอีกไปจนกว่าโปรแกรมจะหยุดหรือจบโปรแกรม วิธีนี้จะมีทั้งการแปลภาษาและทำงานโปรแกรมสลับกันไป จึงทำงานช้ากว่าคอมไพเลอร์ แต่ก็มีข้อดีคือสามารถสร้างได้ง่ายกว่า เพราะการวิเคราะห์โปรแกรมทีละบรรทัดทำได้ง่ายกว่าแปลทั้งโปรแกรม และที่สำคัญการใช้ตัวแปลจะทำให้การทำงานเป็นอิสระต่อแพลตฟอร์มได้ นั่นคือซอร์สโค้ดเดียวกัน เมื่อนำไปรันบนจาวาเวอร์ชวลแมชีนของแต่ละแพลตฟอร์มก็จะได้ผลลัพธ์เหมือนกัน



รูปที่ 3.1 การทำงานของคอมไพเลอร์และตัวแปลในภาษาจาวา

จากรูปที่ 3.1 คอมไพเลอร์จะแปลโปรแกรมในภาษาจาวาจากไฟล์ MyProgram.ภาษาจาวา ไปเป็น จาวาไบต์โค้ด (Java Bytecode) ไว้ในไฟล์ MyProgram.class ที่เป็นอิสระต่อแพลตฟอร์ม การแปลงจากโปรแกรมในภาษาไปเป็นจาวาไบต์โค้ดก่อนทำให้ตัวแปลสามารถทำงานได้รวดเร็วกว่าการอินเทอร์พรีทโดยตรงจากโปรแกรมในภาษาจาวา เพราะจาวาไบต์โค้ดมีลักษณะเป็นโค้ดที่มีความสลับซับซ้อนระหว่างภาษาจาวาและภาษาเครื่อง (Intermediate code) น้อยกว่าตัวโปรแกรมในภาษาจาวา จาวาไบต์โค้ดดังกล่าวสามารถนำไปรันบนจาวาเวอร์ชวลแมชีนของแพลตฟอร์มต่างๆ ได้ทันที เนื่องจากจาวาไบต์โค้ดมีลักษณะไม่ขึ้นต่อแพลตฟอร์ม

เราสามารถมองจาวาไบต์โค้ดเป็นคำสั่งภาษาเครื่องที่รันอยู่บนคอมพิวเตอร์จำลองที่เรียกว่า จาวาเวอร์ชวลแมชีน (Java Virtual Machine) อิมพลีเม้นเตชันของจาวาเวอร์ชวลแมชีน (Implementation) เช่น จาวาตัวแปลทั้งเครื่องมือพัฒนาซอฟต์แวร์และเว็บเบราว์เซอร์ และนอกจากนี้ยังมีอยู่ในเครื่องใช้ไฟฟ้าภายในบ้านที่รันเอ็นไวรอนเมนต์ของเพชชันนอลจาวาแอปพลิเคชัน (Personal Java™ Application Environment), สมาร์ทการ์ด (Smart card) ที่รันชุดพัฒนาจาวาคาร์ด (ภาษาจาวา Card™ Development Kit)

วิธีการของจาวาไบต์โค้ดทำให้โปรแกรมที่เขียนขึ้นในภาษาจาวาสามารถทำงานในลักษณะ “เขียนครั้งเดียว ใช้งานได้ทุกที่” กล่าวคือ โปรแกรมภาษาจาวาสามารถถูกคอมไพล์ได้ในทุกแพลตฟอร์มที่มีคอมไพเลอร์ของภาษา และจาวาไบต์โค้ดก็สามารถรันได้ทุกๆ อิมพลีเม้นเตชันของจาวาเวอร์ชวลแมชีน

3.2 ลักษณะการเขียนโปรแกรมด้วยภาษาจาวา

เป็นการเขียนโปรแกรมแบบเชิงวัตถุ (Object Oriented Programming) โดยจะมององค์ประกอบต่างๆ ให้เป็นวัตถุ (Object) ดังนี้

3.2.1 คลาส (Classes) คลาสนั้นเปรียบเสมือนตัวแบบที่นำไปใช้สร้างวัตถุ โดยแต่ละคลาสจะประกอบไปด้วยลักษณะเฉพาะของข้อมูลในวัตถุพร้อมทั้งรายละเอียดของการกระทำ (Action) ที่วัตถุนั้นๆ

โดยส่วนมากจะใช้เมธอด (Methods) ดังนั้นวัตถุที่อยู่ในคลาสเดียวกันจะมีรายละเอียดต่างๆ เหมือนกัน มีประโยชน์อยู่ 2 ประการที่สำคัญในการใช้คลาสดังกล่าวคือ

- ทำให้โปรแกรมที่สร้างขึ้นมามีความง่ายต่อการแก้ไข เนื่องจากแต่ละคลาสจะไม่เกี่ยวข้องกัน ดังนั้นการแก้ไขสิ่งใดๆ ในคลาสจะไม่มีผลกระทบต่อคลาสอื่นๆ
- เนื่องจากการแก้ไขในแต่ละคลาสไม่มีผลต่อคลาสอื่น ดังนั้นจึงทำให้โปรแกรมอื่นๆ สามารถเรียกใช้งานคลาสต่างๆ ได้โดยที่ไม่มีผลกระทบต่อคลาสต่างๆ ได้โดยที่ไม่มีผลกระทบต่อคลาสอื่น ซึ่งความสามารถนี้เรียกว่าการนำกลับมาใช้ใหม่ (Reusability)

3.2.2 เมธอด (Method)

ความหมายของเมธอด ก็คือ ระเบียบการทำงานของงานใดงานหนึ่ง ซึ่งภายในของเมธอด จะมีขั้นตอน

การทำงานที่แตกต่างกัน ขั้นตอนต่างๆ เหล่านี้ก็คือคำสั่งต่างของภาษาจาวาที่นำมาใช้นั่นเอง

3.2.3 การซ่อนรายละเอียดข้อมูล (Data Encapsulation)

มีความสามารถในการซ่อนข้อมูลภายในแต่ละวัตถุไว้เพื่อไม่ให้วัตถุอื่นมองเห็นได้ ดังนั้นถ้ามีความต้องการใช้ข้อมูลเกิดขึ้น วัตถุที่ต้องการใช้ข้อมูลจะทำการร้องขอไปยังวัตถุอื่นที่มีข้อมูลเพื่อให้ส่งข้อมูลกลับมาให้ตามที่ต้องการเท่านั้น โดยขบวนการต่างๆ จะถูกซ่อนไว้ทั้งหมด ประโยชน์ของวิธีการนี้ก็คือในการแก้ไขข้อมูลหรือกระทำการใดๆ ในวัตถุหนึ่งจะไม่ผลต่อวัตถุอื่น ซึ่งทำให้ผู้อื่นที่จะนำโปรแกรมไปใช้ไม่จำเป็นต้องรู้วิธีการเขียนโปรแกรมนั้นๆ ก็สามารถเข้าใจการทำงานของโปรแกรมนั้นได้ ซึ่งเป็นคุณสมบัติอย่างหนึ่งที่ทำให้ภาษาจาวาเป็นภาษาที่มีประสิทธิภาพมากกว่าภาษาอื่น

3.2.4 การสืบทอดคุณสมบัติ (Inheritance)

ลักษณะหนึ่งที่สำคัญอย่างหนึ่งของการเขียนโปรแกรมเชิงวัตถุก็คือ การสัมพันธ์กับโครงสร้างแบบชั้น (Hierarchy) หรือก็คือ การสืบทอดคุณสมบัติ โดยสิ่งที่อยู่ในคลาสแม่จะสืบทอดไปยังคลาสลูกอย่างเช่น A มีคุณสมบัติอยู่ 3 อย่างแล้วก็ถ่ายทอดไปยัง B และ C ซึ่ง B และ C อาจจะมีคุณสมบัติอื่นๆ เพิ่มเติมก็ได้ แต่อย่างไรก็ต้องมีคุณสมบัติทั้ง 3 ประการของ A อยู่ด้วย

3.3 ประเภทของโปรแกรมจาวา

มีอยู่ด้วยกัน 2 ประเภทต้องทำการคอมไพล์จากไฟล้นามสกุล .ภาษาจาวา ซึ่งเป็นซอร์สโค้ด (Source code) ให้เป็นไบต์โค้ด (Byte code) ที่มีนามสกุล .class มีดังนี้

3.3.1 จาวาแอปพลิเคชัน (Java Application)

เป็นโปรแกรมที่สามารถนำมาใช้งานได้โดยอิสระ (Stand Alone Program) ด้วยภาษาจาวาโดยจะทำงานภายใต้ตัวแปลภาษาจาวา (Java interpreter) ทั้งนี้สามารถนำโปรแกรมที่ได้ไปใช้งานกับคอมพิวเตอร์ต่างแพลตฟอร์มกันได้

3.3.2 จาวาแอปเพล็ต (Java Applets)

เป็นโปรแกรมขนาดเล็กที่สร้างขึ้นด้วยภาษาจาวา โดยจะถูกเรียกจากเอกสาร HTML ซึ่งจะสามารถทำงานได้ภายใต้แอปเพล็ตวิวเวอร์ (Applet Viewer) หรือเว็บเบราว์เซอร์ (Web Browser) ที่มีตัว

แปลภาษาจาวาบนเครื่องที่ได้รับเอกสาร HTML นั้นไป โดยโปรแกรมที่เป็นแอปพลิเคชันนี้ไม่สามารถควบคุมการดำเนินงานด้วยตัวเองได้ทั้งหมด ซึ่งคลาสแอปพลิเคชันจะอยู่ในแฟ้มगे ภาษาจาวา.แอปพลิเคชัน ใช้สำหรับสร้างแอปพลิเคชัน



บทที่ 4

จาวา เนทีฟ อินเทอร์เฟซ

Java Native Interface

จาวาเนทีฟ อินเทอร์เฟซ (Java Native Interface) เป็นการเขียนโปรแกรมภาษาจาวาเริ่มแรกสำหรับจาวาที่เป็น เจดีเค (JDK) ซึ่งจะทำให้มั่นใจได้ว่าโค้ดต่าง ๆ นั้นสมบูรณ์ตามรูปแบบที่เขียน

เจเอ็นไอ (JNI) สามารถใช้ได้ ใน เวอร์ชวลแมชีน (Virtual Machine) เพื่อทำงานร่วมกับแอปพลิเคชันและไลบรารีต่างๆที่เขียนขึ้นด้วยภาษาอื่นๆ เช่น ซี(C) ซีพลัสพลัส (C++) แอสเซมบลี (Assembly)

นักเขียนโปรแกรมใช้ เจเอ็นไอ (JNI) ในการเขียน เมธอด (Method) ในสถานการณ์ที่แอปพลิเคชันไม่สามารถเขียนด้วยภาษาจาวาได้ ตัวอย่างเช่น

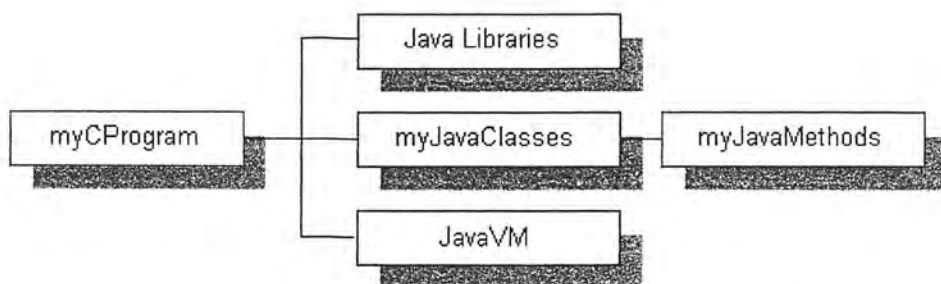
- สแตนด์อโลนคลาสไลบรารีไม่สนับสนุนความต้องการรูปแบบที่ขึ้นอยู่กับแอปพลิเคชัน
- เมื่อมีไลบรารีหรือแอปพลิเคชันที่เขียนขึ้นด้วยภาษาอื่นๆและต้องการที่จะใช้ร่วมกับภาษาจาวา
- เมื่อต้องการแสดงเวลาในไทม์คริติคัลโค้ด (time-critical code) ในการเขียนโปรแกรมระดับล่างและต้องการให้จาวาเรียกใช้งาน

การเขียนโปรแกรมผ่านเจเอ็นไอเฟรมเวิร์กจะช่วยให้ใช้เนทีฟเมธอดในการทำงานหลายๆ ระบบการทำงานอาจจะแทนแอปพลิเคชันหรือแก้ปัญหาที่เกิดจากการเขียนโปรแกรมด้วยจาวา

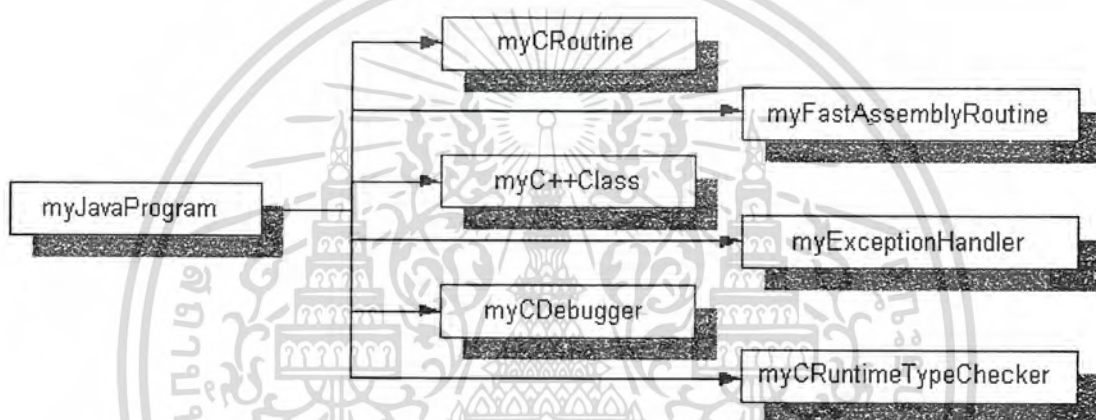
เจเอ็นไอเฟรมเวิร์ก (JNI Framework) สามารถให้เนทีฟเมธอดใช้ประโยชน์จาวาออบเจกต์ (Java object) เหมือนกับที่จาวาใช้ เนทีฟเมธอดสามารถสร้างจาวาออบเจกต์ รวมทั้งอาร์เรย์ (array) และข้อความ (String) และยังสามารถใช้ออบเจกต์ในการทำงานได้ด้วยถึงแม้ว่าจะเป็นโค้ดที่เขียนขึ้นด้วยจาวาก็ตาม และยังสามารถปรับปรุงจาวาออบเจกต์ที่สร้างขึ้นมาเองหรือออบเจกต์ที่ผ่านเข้ามา ไม่ว่าจะเป็นจาวาโค้ดหรือเนทีฟในแอปพลิเคชันก็สามารถสร้าง ปรับปรุง และเข้าถึงจาวาออบเจกต์และใช้งานร่วมกันได้ทั้งคู่

การเรียกใช้งานจาวาเมธอดในเนทีฟเมธอดนั้นทำได้ง่าย โดยไม่จำเป็นต้องเตรียมการในการเรียกใช้จาวาเมธอด นั่นคือสามารถเรียกใช้ผ่านทางพารามิเตอร์และรับค่ากลับมาเมื่อการทำงานเสร็จสิ้น

เจเอ็นไอยังสามารถใช้ความได้เปรียบในการเขียนโปรแกรมด้วยภาษาจาวาได้จากเนทีฟเมธอด นั่นคือจับหรือขว้างข้อยกเว้นและต้องการใช้ข้อยกเว้นนั้นในแอปพลิเคชัน เนทีฟเมธอดยังรับข้อมูลจากจาวาคลาสโดยการเรียกใช้เจเอ็นไอฟังก์ชัน (JNI Function) โหลดคลาสและรับข้อมูลของคลาส และยังสามารถใช้เจเอ็นไอในการตรวจสอบการทำงานได้ด้วย

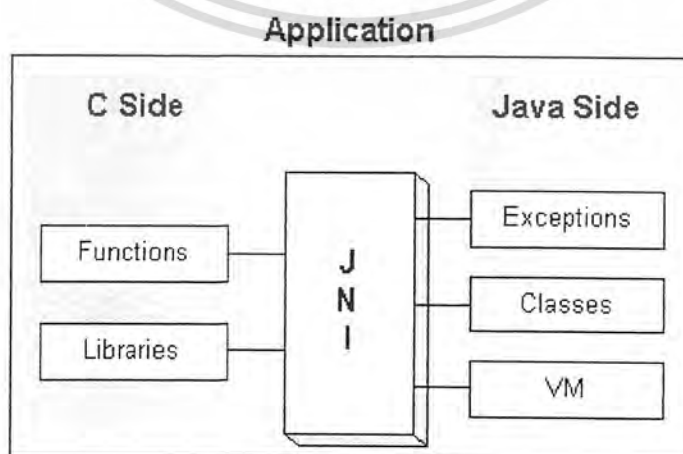


รูปที่ 4.1 แสดงให้เห็นถึงการที่โปรแกรมที่เขียนด้วยภาษาซี ใช้เงื่อนไขในการเชื่อมต่อกับจาวาไลบรารี
การเรียกใช้จาวาเมธอด การใช้จาวาคลาส



รูปที่ 4.2 อธิบายถึงการเรียกใช้ฟังก์ชันภาษาซีที่มาจากจาวาแอปพลิเคชัน

ไดอะแกรมนี้แสดงถึงความเป็นไปได้ในการใช้เงื่อนไขแทนจากโปรแกรมภาษาจาวา รวมทั้ง
การเรียกใช้ซี รูทีน (routines) ใช้ซิปล์สพลัสคลาส (C++ Classes)



รูปที่ 4.3 แสดงให้เห็นถึงความสัมพันธ์ของแอปพลิเคชันภาษาซีและจาวา

4.1 การเขียนเนทีฟเมธอด

สำหรับโปรแกรมจาวาการเขียนเนทีฟเมธอดมีหลายขั้นตอน ดังนี้

- 1) เริ่มต้นด้วยการเขียนโปรแกรมจาวา สร้างจาวาคลาสที่ประกาศเนทีฟเมธอด คลาสนี้ประกอบไปด้วยการประกาศหรือสัญลักษณ์สำหรับเนทีฟเมธอด รวมทั้งเมธอดหลักที่ใช้เรียกเนทีฟเมธอดด้วย
- 2) คอมไพล์จาวาคลาสที่ประกาศเนทีฟเมธอดและเมธอดหลักสร้างไฟล์ส่วนหัวสำหรับเนทีฟเมธอดโดยใช้ javah โดยใช้ flag -jni เมื่อสร้างไฟล์ส่วนหัวแล้วจะได้สัญลักษณ์สำหรับเนทีฟเมธอด
- 3) เขียนคำอธิบายของเนทีฟเมธอดในภาษาที่ต้องการ
- 4) คอมไพล์ไฟล์ส่วนหัวและส่วนคำอธิบายในไลบรารีเดียวกัน
- 5) เรียกใช้โปรแกรม

ขั้นที่ 1 เขียนจาวาโค้ด

สร้างจาวาคลาสชื่อ HelloWorld ที่ประกาศเนทีฟเมธอด ซึ่งรวมถึงเมธอดหลัก

ขั้นที่ 2 คอมไพล์จาวาโค้ด

ใช้ javac คอมไพล์โค้ดที่ได้จากขั้นที่ 1

ขั้นที่ 3 สร้างไฟล์ .h

ใช้ javah ในการสร้าง ไฟล์ส่วนหัวในรูปแบบเจเอ็นไอ จากคลาส HelloWorld ไฟล์ส่วนหัวเตรียมฟังก์ชันสำหรับอธิบายเนทีฟเมธอด DisplayHello()

ขั้นที่ 4 เขียนคำอธิบายเนทีฟเมธอด

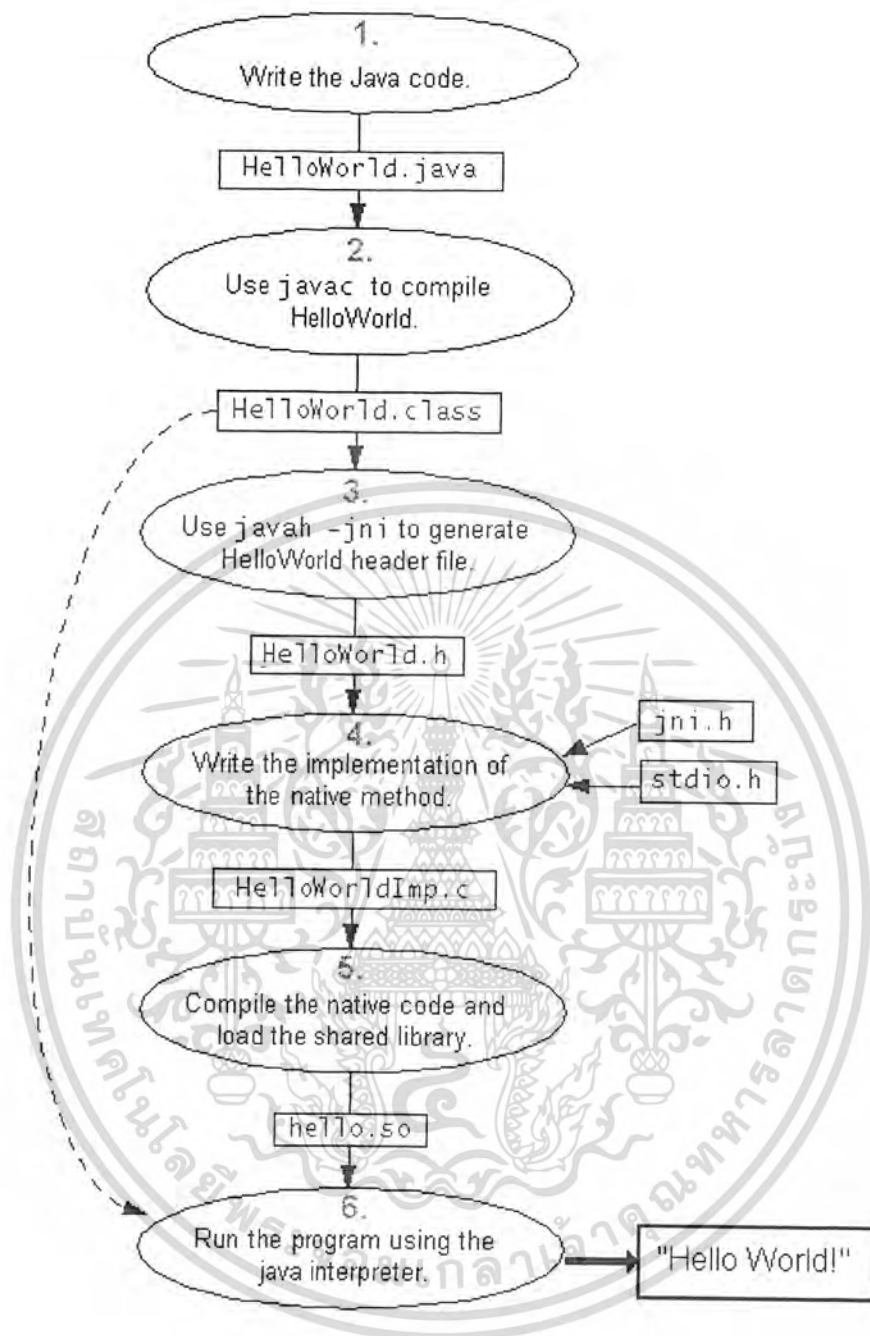
คำอธิบายต้องเป็นฟังก์ชันธรรมดาที่สามารถรวมกับจาวาคลาสได้

ขั้นที่ 5 สร้างไลบรารีรวม

ใช้ ซี คอมไพเลอร์ ในการคอมไพล์ไฟล์ .h และ .c ที่สร้างขึ้นมาในขั้นที่ 3 และ 4 ลงในไลบรารีเดียวกัน

ขั้นที่ 6 เรียกใช้โปรแกรม

สุดท้ายใช้ จาวา จาวาอินเทอร์พรีเตอร์ในการรันโปรแกรม



รูปที่ 4.4 แสดงให้เห็นถึงขั้นตอนของโปรแกรม HelloWorld

บทที่ 5

พื้นฐานมอร์ฟิท

5.1 โคออดิเนต 2 มิติ

การที่จะเข้าใจระบบ 3 มิติ ต้องเข้าใจระบบโคออดิเนต 2 มิติก่อน จากรูปที่ 5.1 แกน x ค่าบวก คือ เส้นตรงที่เริ่มจากจุดออริจินผ่านจุด $(1,0)$ หรือสามารถเขียนได้อีกอย่างว่า เวกเตอร์ $[1,0]$ ส่วนแกน y ค่าบวก คือ เส้นตรงที่เริ่มจากจุดออริจินผ่านจุด $(0,1)$ หรือสามารถเขียนได้อีกอย่างว่า เวกเตอร์ $[0,1]$ ส่วนแกน x ค่าลบจะอยู่ใต้จุดออริจิน และแกน y ค่าลบจะอยู่ทางด้านซ้ายของจุดออริจิน



รูปที่ 5.1 แกน x - y ของระบบโคออดิเนต 2 มิติ

5.1.1 โลก กำหนดให้ โลก ระบบ โคออดิเนต (World Space) มีแกน x ด้านบวกชี้ขึ้นไปเสมอ และ แกน y ด้านบวกชี้ไปทางขวาเสมอ

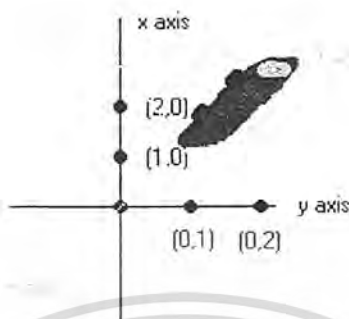
เพื่อให้เข้าใจง่ายขึ้น ยกตัวอย่างให้มีรถคันหนึ่งจอดอยู่ที่จุดออริจิน และหันหน้าไปทางแกน y ด้านลบ หรือหันหน้าไปด้านล่าง เพราะฉะนั้นทิศทางของรถ คือ $[-1,0]$ สมมุติรถคันนั้นเลี้ยวขวา รถก็จะหันหน้าขึ้นข้างบนทิศทางของรถก็จะเปลี่ยนเป็น $[1,0]$



รูปที่ 5.2 ระบบ โคออดิเนตของวัตถุเทียบกับ โลก (World Space)

5.1.2 ออบเจกต์ระบบโคออดิเนต (Object Space) เป็นระบบโคออดิเนตของออบเจกต์นั้นเมื่อรถเคลื่อนที่ไปข้างหน้า แสดงว่ามันมีทิศทางในแกน x เสมอ การที่จะทำเช่นนี้จึงต้องมีแกนอีกแกนที่เป็น

อิสระจากแกนอื่นๆ กำหนดให้แกนนั้นเป็นแกนเฉพาะของรถเท่านั้น กำหนดให้แกน x ด้านบวกชี้ออกจากทางด้านหลังของรถ และแกน y ชี้ออกจากทางซ้ายของรถดังรูปที่ 5.3 และทิศทางที่รถหันหน้าไปคือ $[-1,0]$ เสมอ



รูปที่ 5.3 Object Space

ตามรูปที่ 5.2 และ 5.3 จะพบว่าในระบบ โคออดิเนตของวัตถุ รถจะเคลื่อนที่ไปข้างหน้าตามทิศทาง $[-1,0]$ แต่ในโคออดิเนตของโลก รถมีทิศทาง $[1,1]$

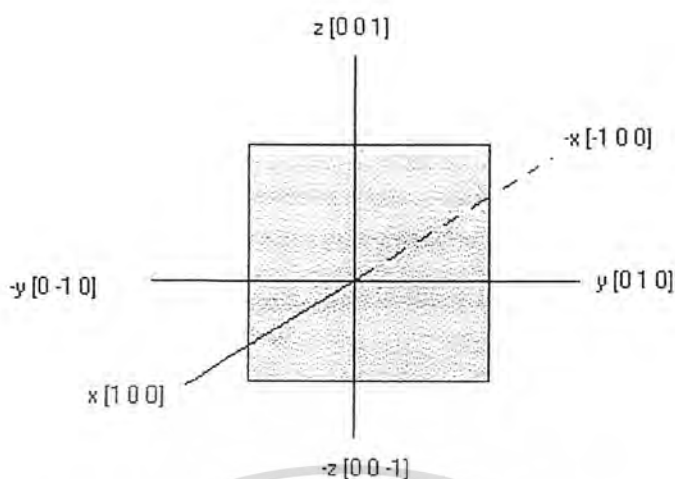
ระบบโคออดิเนตของโลกแกน x และ y จะแทนถึง เวกเตอร์ $[1,0]$ และ $[0,1]$ ตามลำดับ และแกนของวัตถุก็จะเคลื่อนตามวัตถุไปเสมอจากรูปที่ 5.3 แกน x - y ของรถจะเท่ากับ เวกเตอร์โคออดิเนตของโลก $[-1,-1]$ และ $[-1,1]$ ตามลำดับ

ในเอนจินมอร์ฟนั้นจะสามารถเปลี่ยนเวกเตอร์ระหว่าง 2 ระบบนี้ได้ทำให้ผู้โปรแกรมสามารถเลือกใช้ระบบโคออดิเนตที่เหมาะสมต่อสถานการณ์ได้

5.2 ระบบโคออดิเนต 3 มิติ

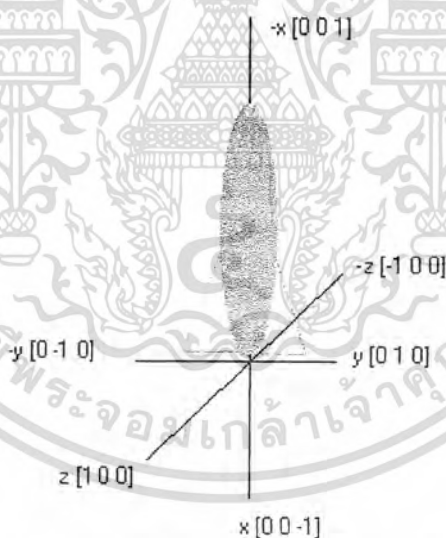
ในระบบโคออดิเนต 3 มิติ จะต้องตั้งอยู่บนกฎมือซ้าย หรือ มือขวอย่างใดอย่างหนึ่ง ใน ไคเร็กซ์ เอกซ์ จะใช้ระบบโคออดิเนตตามกฎมือซ้าย แต่ในมอร์ฟิท และใน โปรแกรมสร้างภาพ 3 มิติ โดยทั่วไปจะยึดระบบโคออดิเนต 3 มิติตามกฎมือขวา

ตามกฎมือขวถ้าสมมติให้จุดกลางจอมอนิเตอร์เป็นจุดออริจิน จุดที่อยู่บนแกน x ด้านบวก คือ จุดที่อยู่หลังหน้าจอมอนิเตอร์ จุดที่อยู่บนแกน x ด้านลบ คือ จุดที่ชี้ออกมาด้านหน้า แกน y ด้านบวก คือ จุดที่ชี้ไปทางขวาของจอ ในแกนลบจะชี้ไปทางซ้าย ส่วนแกน z ด้านบวก คือ จุดที่อยู่ด้านบนของจอ ในด้านลบก็จะอยู่ด้านล่างของจอ



รูปที่ 5.4 ระบบ โคออดิเนต 3 มิติตามกฎมือขวา

ถ้าเป็นระบบ โคออดิเนตนี้ รถที่วิ่งไปข้างหน้าจะมีทิศ $[-1,0,0]$ และรถที่วิ่งไปทางขวาจะมีทิศ $[0,1,0]$ เกล็ดเครื่องบินขึ้นจากพื้นไปด้านบนของจอจะมีทิศ $[0,0,1]$ นี่เป็นการอ้างอิงทิศทางเทียบกับระบบโคออดิเนตของโลก



รูปที่ 5.5 ระบบ โคออดิเนต 3 มิติตามกฎมือขวาเทียบกับวัตถุ

ถ้าเปรียบเทียบกับระบบโคออดิเนตของโลกแกน x จะชี้ออกมาจากด้านหลัง แกน y จะชี้ไปทางขวา แกน z จะชี้ขึ้นข้างบน ยกตัวอย่างเช่น จรวดที่ยิงขึ้นสู่ท้องฟ้าจะมีค่าแกน x, y และ z เป็น $[0,0,-1]$ $[0,1,0]$ $[1,0,0]$ ตามลำดับ

5.2.1 การบวกเวกเตอร์ ให้โคออดิเนตของวัตถุด้านหน้าห่างออกไป n ในระบบโคออดิเนตของวัตถุ จุดๆนั้น จะมีค่าเป็น $[-n,0,0]$ และให้วัตถุด้านหน้ามีทิศทางเดียวกับคือ มีทิศทาง $[-1,0,0]$ ตำแหน่งในระบบโคออดิเนตของโลกของวัตถุที่อยู่ข้างหน้าจะเป็น $[current_x_location - n, current_y_location,$

current_z_location] ส่วนทิศทางก็จะเป็น $[\text{dir}_x, \text{dir}_y, \text{dir}_z] / (x^2 + y^2 + z^2)^{1/2}$ ถ้า $[\text{dir}_x, \text{dir}_y, \text{dir}_z]$ เป็นทิศทางของวัตถุ สาเหตุที่ต้องหารด้วย $(x^2 + y^2 + z^2)^{1/2}$ เพราะค่านี้เป็นค่า magnitude ของเวกเตอร์ การหารด้วยค่านี้นี้จะทำให้ค่า $(\text{dir}_x^2 + \text{dir}_y^2 + \text{dir}_z^2)^{1/2} = 1$ (ค่า dir_x คือค่า dir_x ที่ถูกหารด้วยค่า magnitude แล้ว) เพราะฉะนั้น

$$\text{location_in_front} = \text{current_location} + n * \text{unit_direction}$$

แต่ถ้าจะคำนวณตำแหน่งถัดไปของวัตถุโดยมีตำแหน่งปัจจุบันกับความเร็วของวัตถุ

$$\text{New_location} = \text{current_location} + \text{speed} * \text{direction}$$

5.2.2 การลบเวกเตอร์ การเลื่อนวัตถุไปข้างหน้าเป็นตัวอย่างของการเพิ่มเวกเตอร์ (การบวกผลคูณของเวกเตอร์ทิศทางกับตำแหน่งปัจจุบัน)

แต่ในการหาเวกเตอร์ระหว่าง 2 วัตถุ ต้องใช้การลบเวกเตอร์ สมมุติให้กล้องอยู่ที่ $\text{cam_pos} = [x, y, z]$ และวัตถุอยู่ที่จุด $\text{obj_pos} = [x, y, z]$ ถ้าต้องการหาเวกเตอร์ที่กล้องชี้ลงมาที่วัตถุ เพื่อที่จะแสดงผลวัตถุนั้น ทำได้โดยเอาตำแหน่งวัตถุนั้นตั้งแล้วลบด้วยตำแหน่งวัตถุ

$$\text{Direction} = \text{obj_pos} - \text{cam_pos}$$

หรือถ้านำไปเขียนแยกทีละแกนได้ดังนี้

$$\text{Dir}_x = x - \text{cx}$$

$$\text{Dir}_y = y - \text{cy}$$

$$\text{Dir}_z = z - \text{cz}$$

ค่าระยะห่างระหว่างวัตถุกับกล้องก็คือค่า magnitude ของเวกเตอร์หรือ $(\text{dir}_x^2 + \text{dir}_y^2 + \text{dir}_z^2)^{1/2}$ นั่นเอง

5.2.3 ระนาบ เรื่องนี้เป็นเรื่องสุดท้ายเกี่ยวกับระบบ โคออดิเนต 3 มิติ ที่จะกล่าวถึง ระนาบ 3 มิติ จะแทนด้วยสมการ

$$Ax + By + Cz + D = 0$$

ถ้าเป็นระนาบที่ทุกๆบนระนาบมีค่า $x = -4$ ก็จะสามารถเขียนได้เป็น

$$1x + 0y + 0z + 4 = 0 \text{ หรือ } x + 4 = 0$$

ในระนาบโดยทั่วไปจะดูอ้างอิงโดยค่าเวกเตอร์ที่ตั้งฉากกับระนาบนั้นๆ โดยนิยามให้ระนาบเกิดจากเส้นตรงจำนวนอนันต์มารวมกัน ยกตัวอย่างเช่นในระนาบ xy ที่แกน z มีค่าเป็น 0 ก็จะมีแกน z เป็นเวกเตอร์ตั้งฉากนั่นเอง $[(0, 0, 1)]$ หรือ $[0, 0, -1]$ โดยทั่วไปจะหาสมการระนาบได้โดยแทนค่าเวกเตอร์ตั้งฉาก $[A, B, C]$ ในสมการ $Ax + By + Cz + D = 0$

5.3 แชนเดล (Handles)

ในการแยกแยะวัตถุทุกอย่างในมอร์ฟิทเอพีไอจำเป็นต้องรู้แชนเดลของวัตถุนั้นๆ วัตถุทุกอย่างและวัตถุทั้งหมดจะต้องมีแชนเดลเป็นของตัวเอง ไม่ว่าจะเป็นบิตแมป (Bitmaps) หรือโพลีกอนก็ตาม

แชนเดลในมอร์ฟิทเป็นค่า DWORD (หรือ unsigned int) นั่นก็หมายความว่าถ้าส่งค่า unsigned int แทนแชนเดลของวัตถุผ่านฟังก์ชันไป คอมไพเลอร์ก็จะทำงานตามปกติเพราะไม่รู้ข้อแตกต่าง แต่ในช่วง

รันใหม่ มอร์ฟิทจะ แสดงข้อความขึ้นมาแสดงข้อผิดพลาด และค่าแฮนเคิลเป็นนูล (NULL) หรือแฮนเคิลของวัตถุถูกทำลาย มอร์ฟิทจะ แสดงข้อผิดพลาดเช่นกัน

ถ้าสงสัยว่าค่าแฮนเคิลไม่ถูกต้องสามารถตรวจสอบได้โดยใช้ฟังก์ชัน “is” เช่น

`Morfit_camera_is_camera()` , `Morfit_object_is_object()` , ฯลฯ

ส่วนการอ่านค่าแฮนเคิลของวัตถุทำได้โดยใช้ฟังก์ชัน เช่น แฮนเคิลของกล้อง

`Morfit_camera_get_default_camera()` หรือในกรณีที่จะอ่านค่าแฮนเคิลของกล้องที่มีชื่อตามที่เรากำลังต้องการก็ให้ใช้ฟังก์ชัน `Morfit_camera_get_using_name("my_camera")` ในการเขียนโปรแกรมควรเรียกฟังก์ชันอ่านค่าแฮนเคิลเพียงครั้งเดียวก็พอไม่ควรเรียกทุกครั้งก่อนจะทำอะไรกับวัตถุนั้น

5.4 ภาพรวมของเกมเอนจินมอร์ฟิท

มอร์ฟิทเอพีไอเป็นกลุ่มของฟังก์ชัน โดยในแต่ละกลุ่มของฟังก์ชันจะมีหน้าที่ต่าง ๆ กันในการกระทำกับโลก 3 มิติ กลุ่มที่สำคัญได้แก่ เอพีไอของกล้อง (camera API) , ออบเจกต์เอพีไอ (object API) , เอนจินเอพีไอ (engine API) , โพลีกอนเอพีไอ (polygon API) และ บิตแมปเอพีไอ(bitmap API)

โดยฟังก์ชันต่างๆของมอร์ฟิทจะนำหน้าด้วยคำว่า มอร์ฟิทเช่น `Morfit_camera_get_location()` , `Morfit_engine_load_world()` เป็นต้น ส่วนฟังก์ชันต่างๆของมอร์ฟิทได้มี Help อธิบายหน้าที่ของฟังก์ชันไว้หมดแล้ว เรียกใช้โดยการทำเครื่องหมายที่ “Use Extension Help” ใน Help menu ของคอมไพเลอร์ และในเฮดเดอร์ไฟล์ `morfit_api.h` ของมอร์ฟิทก็มีคำอธิบายหน้าที่ของฟังก์ชันเช่นกัน

ในส่วนต่อไปจะอธิบายถึงหน้าที่ของเอพีไอสำคัญๆ ของมอร์ฟิท

ออบเจกต์เอพีไอ (Object API)

ทำหน้าที่ควบคุมวัตถุในโลก 3 มิติที่สร้างขึ้น ทุกๆวัตถุไม่ว่าคน , สิ่งก่อสร้าง , พาหนะ ฯลฯ โดยแต่ละวัตถุจะถูกสร้าง ทำลาย หมุน หรือเคลื่อนที่โดยใช้เอพีไอนี้ นอกจากนี้เอพีไอนี้ยังมีการเตรียมฟังก์ชันสำหรับตรวจสอบการชน ว่าจะสามารถเคลื่อนวัตถุไปได้หรือไม่ และยังสามารถกำหนดรายละเอียดทางฟิสิกส์ให้วัตถุแต่ละตัวได้ด้วย เช่น ความยืดหยุ่น , ความเสียดทาน และ แรง เป็นต้น โดยแต่ละวัตถุต้องอยู่ภายใต้กฎฟิสิกส์ และยังมีฟังก์ชันที่ทำหน้าที่เปลี่ยนรูปร่างวัตถุเพื่อนำมาทำแอนิเมชันด้วย

เอพีไอของกล้อง (Camera API)

ทำหน้าที่ควบคุมกล้องในโลกที่สร้างขึ้นมา ไม่ว่าจะเป็นการย้าย , ขยายภาพ , ย่อภาพ หรือโฟกัสไปที่จุดใด โดยกล้องแต่ละตัวสามารถควบคุมโดยโปรแกรมอย่างอิสระ หรือควบคุมให้เคลื่อนไปตาม Track หรือ Chase ก็ได้

กรุ๊ปเอพีไอ (Group API)

เป็นเอพีไอที่อนุญาตให้รวมโพลีกอนต่างๆเข้าไว้เป็นกลุ่มเดียวกัน ทำให้สามารถหมุน , เลื่อน , ย้าย หรือลดขนาดของกลุ่มของวัตถุนั้นพร้อมๆกันได้ เราสามารถอ่านขนาดของกลุ่มวัตถุได้ โดยในมอร์ฟิทจะเรียกว่า “Bounding Box” และประกอบด้วยฟังก์ชันอื่น เช่น ฟังก์ชันทำหน้าที่กำหนดจุดกึ่งกลางของ

กลุ่ม และสามารถกำหนดรายละเอียดทางฟิสิกส์ได้เหมือน ออบเจกต์เอพีไอ (object API), เอนจินเอพีไอ (engine API)

เป็นเอพีไอที่ประกอบด้วยฟังก์ชันต่างๆ ที่จัดการกับปรากฏการณ์ทั้งหมดในโลกที่สร้างขึ้น และการแสดงผลออกทางหน้าจอ ไม่ว่าจะเป็นฟังก์ชันโหนดโลก 3 มิติ ที่สร้างขึ้นหลายฟังก์ชันทำหน้าที่ในการเพิ่มประสิทธิภาพของเกม หรือ เพิ่มคุณภาพของภาพ เช่น ฟังก์ชันในการเรียกใช้ Z-buffer, ฟังก์ชันในการเปลี่ยนความละเอียดที่จะแสดงผล

บิตแมปเอพีไอ (Bitmap API)

เป็นเอพีไอที่ทำหน้าที่โหลด หรือ เปลี่ยนบิตแมปที่อยู่ในเกม บิตแมปที่อ่านทำหน้าที่ในการให้สี และ เทกซ์เจอร์กับวัตถุ

แอนิเมชันเอพีไอ (Animation API)

เป็นเอพีไอที่ทำหน้าที่แสดงแอนิเมชันของชุดโพลีกอน ที่เปลี่ยนไปตามเวลาต่างๆแอนิเมชันสามารถใช้สร้างเอฟเฟคต่างๆเช่น แสดงวัตถุคนวิ่งในโลกที่สร้างขึ้น

โพลีกอนเอพีไอ (Polygon API)

วัตถุทุกวัตถุประกอบขึ้นมาจากโพลีกอน สำหรับเอพีไอนี้จะทำหน้าที่ควบคุมรูปร่าง ,สี ,และ บิตแมปของโพลีกอน เราสามารถตั้งค่าคุณสมบัติเกี่ยวกับแสง ความใส หรือ ความสว่าง เราสามารถใช้ฟังก์ชันนี้ขยับโพลีกอนเพิ่มจุดเข้าไปในนั้น หมุน ,ย้าย หรือขยายโพลีกอนนั้น หรือแม้กระทั่งหาจุดตัดในโพลีกอนหาเส้นตรงที่โพลีกอนตัดกัน และยังสามารถนำไปใช้แสดงผลแสงและเงาให้กับวัตถุด้วย

เอพีไอ อื่นๆ

เช่น พอยนต์เอพีไอ (Point API) , เอพีไอฉากหลัง(Background API) ,แทร็คเอพีไอ(Track API) และ การ์ด 3 มิติเอพีไอ เป็นเอพีไอที่มีความสำคัญรองลงมาแต่ก็เป็นเอพีไอที่ยังมีประโยชน์อยู่มาก

บทที่ 6

สร้างออบเจกต์เพื่อใช้ Morfit

Create Object To Use With Morfit

6.1 Viewer Mode และ Dynamic Objects

ในการที่จะแสดง โลก ที่เราสร้างขึ้น เราสามารถกำหนดรูปแบบการแสดงผล โดยใช้คำสั่งดังนี้

```
Morfit_engine_load_world(world_file_name , world_directory_path , bitmaps_directory_path, world_mode);
```

ซึ่งรูปแบบของการแสดงผลจะขึ้นอยู่กับค่า world_mode โดยจะเป็น

- Viewer Mode เมื่อเรากำหนดค่านี้เป็น USER_DEFINE_BEHAVIOR โดยจะ load และคำนวณ โลก ทั้งหมดในที่เดียวเลย
- Editor Mode เมื่อเรากำหนดค่านี้เป็น EDITOR_MODE ไม่จำเป็นต้องสร้างและคำนวณ โลก ทั้งหมดตั้งแต่เริ่มต้น

6.2 การตั้งชื่อและการเข้าใช้วัตถุ

มีขั้นตอนดังนี้

- 1) ใน เวิลด์บิวเดอร์(World Builder) เราสามารถตั้งชื่อวัตถุได้ใน dialog box “ Properties of Object “
- 2) จากนั้นเราสามารถเข้าถึงวัตถุได้โดยอ้างแฮนเดิล (handle) ซึ่งสร้างจากการกำหนดค่าดังต่อไปนี้

```
DWORD Morfit_object_get_object_using_name(char *object_name);
```

6.3 การเคลื่อนย้ายวัตถุ มี 4 ประเภทดังนี้

6.3.1 การควบคุมโดยตรง

สามารถใช้ควบคุมวัตถุได้อย่างมีประสิทธิภาพที่สุด มีการทำงานที่สำคัญๆ ได้แก่

- การกำหนดตำแหน่งให้วัตถุ คือ การทำการวางวัตถุลงบนตำแหน่งตามค่าในแกน [x,y,z] ดังนี้

```
Morfit_object_set_location( DWORD object_handle , double x , double y , double z);
```

- การกำหนดทิศทางให้วัตถุ จะใช้คำสั่งดังนี้

```
Morfit_object_set_direction(DWORD object_handle , double x , double y , double z);
```

- การหาตำแหน่งของวัตถุในขณะนั้น

Morfit_object_set_location(DWORD object_handle , double * x , double *y , double *z);

- การเคลื่อนที่โดยอ้างอิงตำแหน่งสัมพัทธ์ คือ การเคลื่อนวัตถุโดยอ้างอิงจากตำแหน่งปัจจุบันของวัตถุ เช่น

Morfit_object_move(my_object , world_SPACE ,0,0,10);

จุดสำคัญคือการเลือกใช้ระบบโคออร์ดิเนต โดย

- 1) โลก space จะอ้างอิงกับ โลก เป็นหลักเช่น [-1,0,0] จะทำการเคลื่อนวัตถุตามทิศทางเข้าหาหน้าจอตรงๆ (ตามแกน X ลบ)
 - 2) object space จะอ้างอิงกับวัตถุนั้นๆ เป็นหลักเช่น [-1,0,0] จะทำการเคลื่อนวัตถุตามทิศทางด้านหน้าของวัตถุ ไม่ว่าวัตถุจะหันหน้าไปทางใดอยู่ก็ตาม
- การหมุนวัตถุ จะหมุนวัตถุไปตามแกนที่เราต้องการ ดังนี้

Morfit_object_rotate_x(DWORD object_handle , double degrees , int space_flag);

โดย degrees คือ องศาที่ต้องการหมุนไป และเลือกค่าของ flag_space จาก world_SPACE หรือ OBJECT_SPACE

6.3.2 การกำหนดให้วัตถุเคลื่อนที่ไปตามเส้นทาง (Track)

คือ เมื่อเราไม่ต้องการกำหนดการเคลื่อนไหวนั้นๆด้วยตนเอง เราสามารถกำหนดวัตถุให้เคลื่อนที่ไปตามสิ่งอื่นๆได้ โดยการกำหนดเส้นทาง (Track) จากนั้นจึงใช้คำสั่งด้านล่างเพื่อให้วัตถุเดินตามทางที่ได้กำหนดไว้

Morfit_object_set_track(object_handle , track_handle);

6.3.3 การกำหนดให้เคลื่อนที่ไปตามวัตถุ

เราสามารถกำหนดให้ออบเจกต์ที่สามารถเคลื่อนที่ได้ (dynamic object) ใดๆ เคลื่อนที่ไปตามออบเจกต์ที่สามารถเคลื่อนที่ได้ (dynamic object) ใดๆ ก็ได้ โดยจะมี

ปัจจัยหลัก 4 ประการ ได้แก่

- วัตถุที่กำหนดให้ถูกเคลื่อนที่ตาม (Chase Object) การกำหนดวัตถุให้เคลื่อนที่ตามสิ่งต่างๆทำได้โดย

Morfit_object_set_object_to_chase(DWORD object_handle , DWORD object_to_chase_handle);

Morfit_object_set_camera_to_chase(DWORD object_handle , DWORD camera_to_chase_handle);

Morfit_object_set_group_to_chase(DWORD object_handle , DWORD group_to_chase_handle);

การหยุดวัตถุไม่ให้เคลื่อนไหวนั้นๆตามวัตถุอื่น ทำได้โดย

Morfit_object_set_chase_type(object_handle , NO_CHASE);

- ระยะการเคลื่อนที่ตาม (Chase Offset) เป็นการกำหนดว่าวัตถุที่กำลังเคลื่อนที่ตามอยู่ห่างวัตถุที่ถูกเคลื่อนที่ตามเท่าไร เช่น ถ้า ออฟเซตที่ไล่ตาม (chase offset) เป็น [0,20,0] แล้ววัตถุที่เคลื่อนที่ตามจะเคลื่อนที่ไปพร้อมกับวัตถุที่ถูกตาม แต่จะอยู่เหนือวัตถุที่ถูกตามอยู่ 20 หน่วย เป็นต้น

การกำหนด ออฟเซตของออบเจกต์ที่ต้องการตาม ทำได้โดย

Morfit_object_set_chase_offset(DWORD object_handle , double offset[3]);

- รูปแบบการเคลื่อนที่ตาม (Chase Type) มีทั้งหมด 3 รูปแบบ ได้แก่
 - 1) Chase Precise จะทำให้วัตถุตามอยู่ในตำแหน่งที่สัมพันธ์กับตำแหน่งของวัตถุที่ถูกตามเสมอ เช่น เฮลิคอปเตอร์ เคลื่อนที่ตามรถยนต์ โดยอยู่เหนือกว่ารถ เมื่อรถเลี้ยวจะทำให้เฮลิคอปเตอร์วิ่งตามรถให้ไ้ระยะ offset ที่เท่าเดิมทันที เป็นต้น
 - 2) Chase Location จะทำงานเหมือน precise แต่เฮลิคอปเตอร์จะไม่หมุนตามรถ เพียงแค่รักษาระยะ offset ให้เท่าเดิมเท่านั้น
 - 3) Chase Flexible จะไม่เลี้ยวเพื่อตามวัตถุอย่างแม่นยำแต่จะค่อยๆตามวัตถุไป แม้ว่าจะทำให้ค่า offset คลาดเคลื่อนไปเล็กน้อย

การเลือกเลือกรูปแบบการไล่ตาม ทำได้โดย

Morfit_object_set_chase_type(DWORD object_handle , int chase_type);

- ความผ่อนปรนในการเคลื่อนที่ตาม (Chase Softness)
จะเป็นค่าที่ใช้ในการพิจารณาว่าวัตถุ ตามจะหักเลี้ยวตามวัตถุนำ โดยเลี้ยวสะบัดแรงเท่าไรซึ่งค่าปกติ จะมีค่าอยู่ระหว่าง 0 กับ 1 แต่ใน เวิลด์บิวเดอร์(World Builder) จะเห็นว่า เป็นค่า 0—10 โดยค่าที่ใส่เข้าไปจะถูกหารด้วย 10 เพื่อให้ได้ค่าความนุ่มนวลของภาพในการไล่ตาม (chase softness)จริง

การควบคุมการทำงานของค่าความนุ่มนวลของภาพในการไล่ตาม (chase softness) ได้โดย

Morfit_object_set_chase_softness(DWORD object_handle , double softness);

Double Morfit_object_get_chase_softness(DWORD object_handle);

6.3.4 การกำหนดให้วัตถุเคลื่อนที่ตามกฎฟิสิกส์

โดยในโปรแกรม มอร์ฟิทจะมีคุณสมบัติต่างๆ ให้กำหนดได้หลายอย่าง สามารถดูรายละเอียดได้ที่ไฟล์ “ mrft_api.h “

6.4 ตัวอย่างการสร้างวัตถุ

ในตัวอย่างที่จะยกมานี้ เราต้องการสร้างคนขึ้นมาโดยต้องการให้จุดศูนย์กลางของคนเคลื่อนที่ไปตามเส้นทาง ที่ระยะสูงจากแนวระดับ ครึ่งหนึ่งของความสูงของคนจากแนวระดับ (ระยะ $[z_{max} - z_{min}] / 2$) ซึ่งจะทำให้เท้าของคนอยู่บนพื้นในการหาความสูงของคนเราใช้

`Morfit_object_get_bounding_box(DWORD object_handle , double box[2][3]);`

ซึ่งจะคำนวณค่าสูงสุดและค่าต่ำสุดของ โคออร์ดิเนต X, Y และ Z และเก็บลงใน อาร์เรย์ของกล่อง (box array) ในรูปแบบต่อไปนี้

`box[0][0]` เป็นค่า minimum X

`box[0][1]` เป็นค่า minimum Y

`box[0][2]` เป็นค่า minimum Z

`box[1][0]` เป็นค่า max X

`box[1][1]` เป็นค่า max Y

`box[1][2]` เป็นค่า max Z

ดังนั้นความสูงของคนจะคำนวณได้ดังนี้

`Morfit_object_get_bounding_box(man,box);`

`Double height = (box[1][2] - box[0][2]);`

จากนั้นเราก็กำหนด track offset โดยกำหนด vector ขนาดสูงครึ่งหนึ่ง (ตามแกน z) ของความสูงของคนดังนี้

`double up[3] = {0,0,height/2};`

`Morfit_object_set_track_offset(man,up);`

6.5 การตรวจสอบการชน (Collision Detection)

ทำได้โดยการเรียกใช้ดังนี้

`int Morfit_object_is_movement_possible(DWORD object_handle,double startlocation[3],`

`double end_location[3], DWORD *intersected_polygon,double intersection[3],DWORD *blocking_object);`

ฟังก์ชันจะคืนค่า YES แสดงว่าเคลื่อนที่ไปได้และจะเป็น NO แสดงว่าเกิดการชน เราสามารถกำหนดให้ทุกวัตถุไม่สามารถทะลุผ่านได้โดย

`Morfit_object_make_collisional(DWORD object_handle);`

6.6 กล้อง (Camera)

ภาพที่ถูกแสดงผลออกมาผ่านกล้องจะเป็นเช่นไร ก็ขึ้นอยู่กับปัจจัยหลักๆดังนี้คือ

6.6.1 การตั้งชื่อ กล้อง (Camera) และแฮนเคิล (handles)

คำสั่งที่ใช้กับกล้องจะเหมือนกับของวัตถุโดยจะต่างกันเล็กน้อย โดยเราสามารถสร้างกล้องโดยใช้คำสั่ง

```
DWORD Morfit_camera_create(char *camera_name);
```

ส่วนการทำลายกล้องทำได้โดย

```
int Morfit_camera_delete(DWORD camera_handle); หรือ
```

```
Morfit_camera_delete_all(void);
```

นอกจากนี้ยังมีแฮนเคิลของกล้องที่สำคัญๆ ที่เราควรรู้ คือ

- ค่าปริยายของกล้อง (Default Camera) หมายถึง กล้องที่สร้างขึ้นเป็นครั้งแรกในโลกที่สร้างขึ้นโดยใช้ เวิลด์บิวเดอร์ (World Builder) หรือ ใน โปรแกรมเรียกใช้โดยใช้คำสั่ง

```
Morfit_camera_get_default_camera(void);
```

- กล้องปัจจุบัน (Current Camera) หมายถึง กล้องที่ใช้แสดงผลตัวสุดท้าย (ที่ใช้อยู่ขณะนั้นๆ) ฟังก์ชันบางตัวของมอร์ฟิทนั้นจะ ไม่สามารถใช้กับกล้องที่เรากำหนด แต่จะใช้ได้กับกล้องปัจจุบันเท่านั้น สามารถเรียกได้โดยใช้คำสั่ง

```
int Morfit_camera_set_current(DWORD camera_handle);
```

```
DWORD Morfit_camera_get_current(void);
```

6.6.2 ตำแหน่งของกล้อง

ตำแหน่งของกล้องในโลก 3 มิติ อ้างอิงโดยใช้โคออร์ดิเนต 3 มิติ [x,y,z] เดียวกับวัตถุ และสามารถเปลี่ยนแปลงได้เช่นกัน

- การกำหนดตำแหน่งกล้องจะใช้คำสั่ง

```
Morfit_camera_set_location(DWORD camera_handle, double x, double y, double z);
```

- ส่วนฟังก์ชันนี้ใช้หาตำแหน่งกล้องในขณะนั้น

```
Morfit_camera_get_location(DWORD camera_handle, double x, double y, double z);
```

- ในการเลื่อนกล้องโดยอ้างอิงจากตำแหน่งปัจจุบัน ใช้

```
Morfit_camera_move(DWORD camera_handle, double x, double y, double z);
```

6.6.3 ทิศทางของกล้อง

ทิศทางของกล้องจะเป็นตัวบอกว่ากล้องหันไปทางใดเล็งไปที่วัตถุใดอยู่ โดยสามารถกำหนดทิศทางของกล้องได้โดยใช้คำสั่ง

```
Morfit_camera_set_direction(camera,-1,0,1);
```

ในคำสั่งนี้กล้องมีทิศไปหน้าจอเป็นมุมเบย 45 องศา หรือจะสั่งให้วัตถุชี้ไปทางใดๆ เพียงใช้คำสั่ง

```
Morfit_object_get_location(object , &x , &y , &z);
```

```
Morfit_camera_point_at(camera,x,y,z);
```

นอกจากนี้ยังมีฟังก์ชันที่น่าสนใจอีกมาก เช่น หมุนกล้องไปตามมุมต่างๆ เป็นต้น ซึ่งสามารถดูรายละเอียดได้ที่ ไฟล์ “ mrfit_api.h “

6.6.4 การซูม (Zoom)

หมายถึง พื้นที่ที่เรามองเห็นนั่นเอง ยิ่งพื้นที่ที่เรามองเห็นน้อย ภาพที่เห็นก็จะใหญ่ หรืออีกนัยหนึ่งก็คือความกว้างของมุมมองที่มองเห็นนั่นเอง

- ตัวอย่างโค้ด

```
double Morfit_camera_set_zoom(DWORD camera_handle, double field_of_view_angle);
```

```
double Morfit_camera_get_zoom(DWORD camera_handle);
```

```
double Morfit_camera_modify_zoom(DWORD camera_handle, double field_of_view_change);
```

6.6.5 การควบคุมคุณภาพ

มอร์ฟิทสามารถตั้งค่าคุณภาพที่แสดงออกมาทางหน้าจอได้ ถ้าเราตั้งไว้สูง ความเร็วก็จะลดลง แต่ถ้าตั้งค่าไว้ต่ำก็จะทำให้แสดงผลเร็วขึ้น นอกจากนี้ภาพที่นำมาใช้ก็มีผลต่อคุณภาพภาพเช่นเดียวกัน ถ้าภาพมีขนาดเล็ก ความละเอียดก็จะสูง อาจทำให้ภาพแตกได้

- การอ่านกว้างและยาวของหน้าจอให้ใช้คำสั่ง

```
int Morfit_camera_get_width(DWORD camera_handle);
```

```
int Morfit_camera_get_height(DWORD camera_handle);
```

- ในการตั้งค่าความละเอียดของหน้าจอให้ใช้ค่า ความสูงและความกว้างที่ได้จากคำสั่งทั้งสอง โดยค่า คุณภาพ (quality) จะเท่ากับ ความกว้างคูณความยาว

```
Morfit_engine_set_picture_quality(int quality);
```

```
Morfit_engine_increase_picture_quality();
```

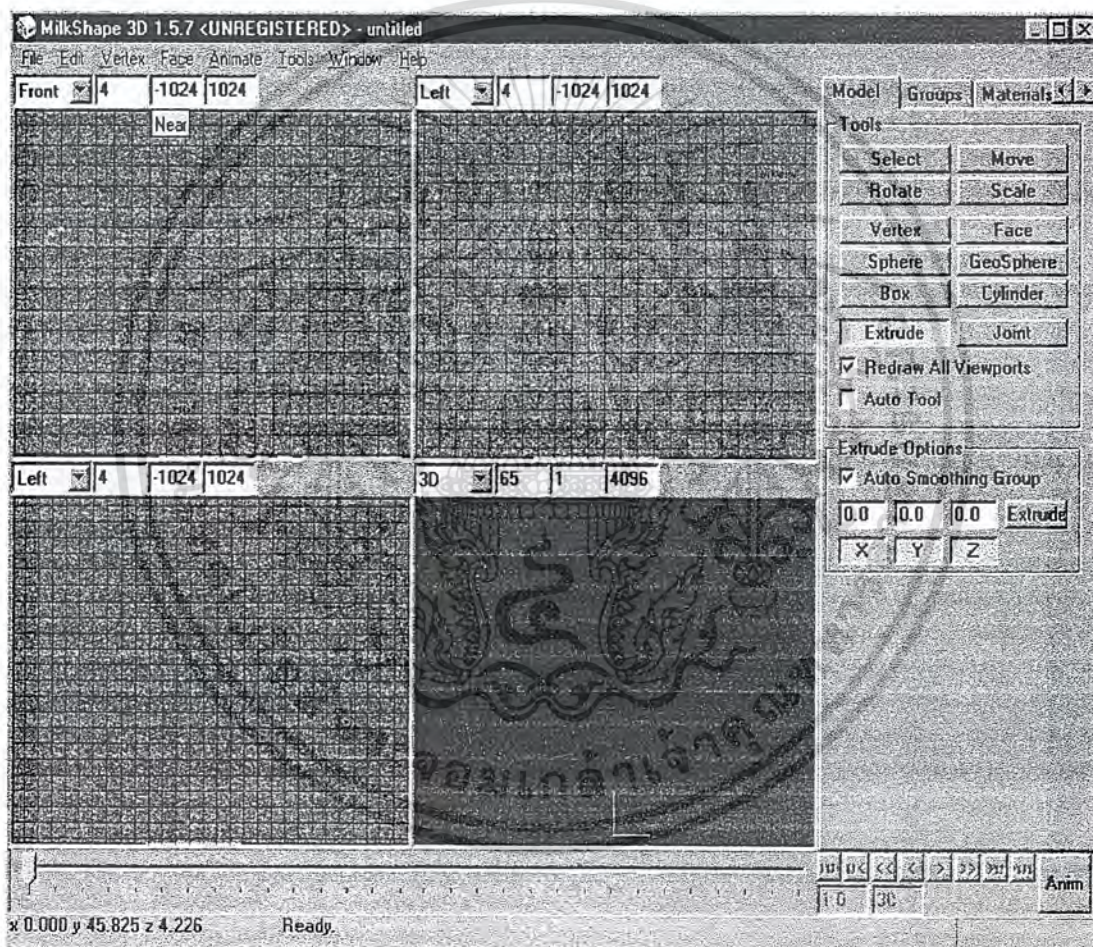
```
Morfit_engine_decrease_picture_quality();
```

บทที่ 7

โปรแกรมสร้างโมเดล 3 มิติ มิลล์เชฟ 3 มิติ

7.1 มิลล์เชฟ 3 มิติ

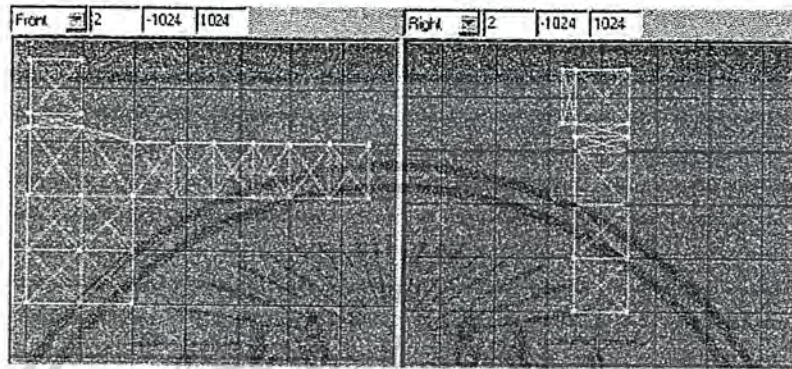
สำหรับโครงการนี้เราได้เลือกใช้โปรแกรม มิลล์เชฟ 3 มิติ ในการสร้างตัวละคร เนื่องจากโปรแกรมนี้นี้มีจุดเด่นอยู่คือการที่จะสามารถจัดการกับแอนิเมชันต่างๆของตัวละครได้เป็นอย่างดี อีกทั้งยังสามารถเอ็กซ์พอร์ตไฟล์โมเดลที่สร้างจากโปรแกรมนี้ออกมาได้หลายฟอร์แมตซึ่งสามารถนำไปใช้ในมอร์ฟิทได้ จึงเหมาะสมที่จะใช้ทำโมเดลตัวละคร



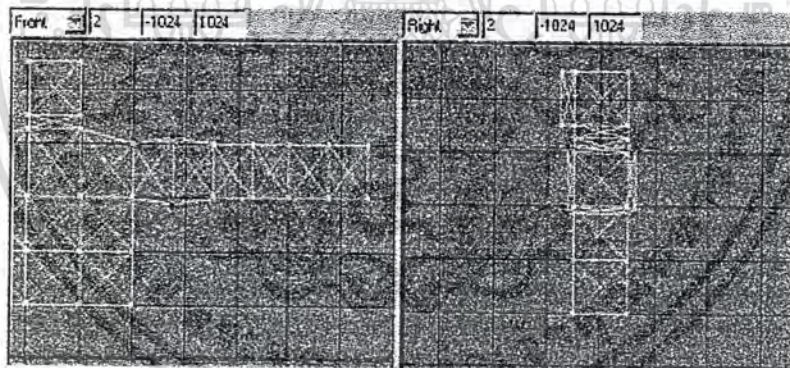
รูปที่ 7.1 แสดงหน้าจอการทำงานของโปรแกรม มิลล์เชฟ 3 มิติ

7.2 การสร้างรูปโมเดล 3 มิติ

ในการสร้างโมเดล 3 มิติสามารถสร้างได้โดยการสร้างรูปแบบอิสระโดยการวาดสามเหลี่ยมหลายๆอันมาต่อกันเป็นรูปโมเดล หรือสามารถเริ่มต้นจากรูปทรงเรขาคณิตที่ใกล้เคียงกับโมเดลที่ต้องการ แล้วค่อยทำการปรับแต่งรูปร่างให้ได้ตามต้องการ โดยการปรับขนาดสเกล , ปรับตำแหน่งจุดเวอร์เท็กซ์ เป็นต้น



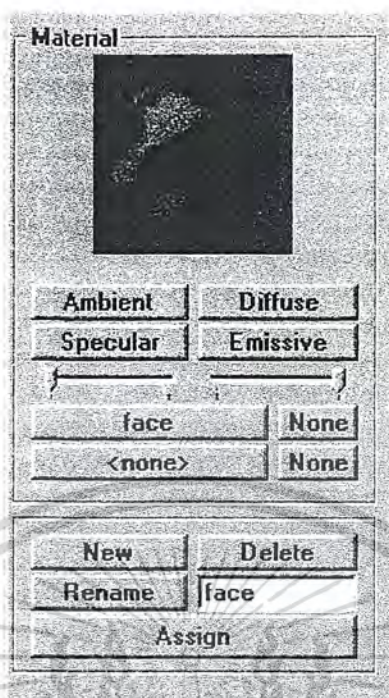
รูปที่ 7.2 แสดงการทำโมเดล 3 มิติ โดยการนำรูปสี่เหลี่ยมมาต่อเป็นรูปร่าง



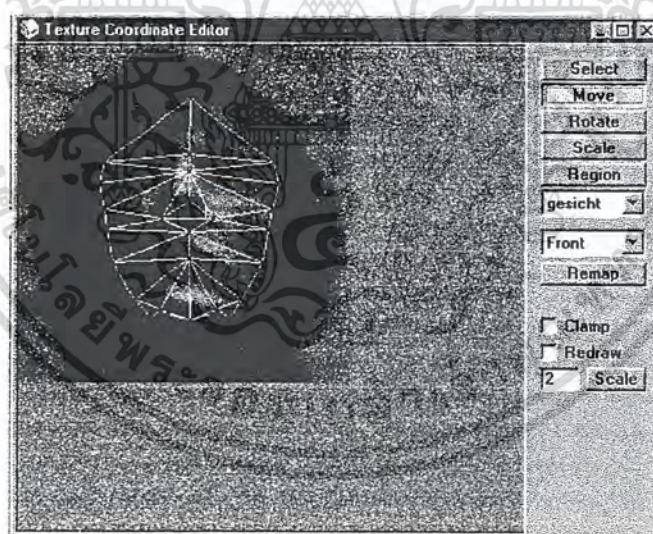
รูปที่ 7.3 แสดงการปรับแต่งขนาดให้ได้รูปตามต้องการ

7.3 การกำหนดพื้นผิว(Texture Mapping)

การกำหนดพื้นผิวในโปรแกรม มัลติเซฟ 3 มิติ นั้นสามารถทำได้อย่างเป็นอิสระเพราะแต่ละ face จะมีข้อมูลของวัตถุ (material) และ โคออดิเนต (coordinate) เป็นของตัวเอง นอกจากนี้เรายังสามารถกำหนดให้พื้นผิวแต่ละส่วนมีวัตถุ (material) มาจากภาพบิตแมปภาพเดียวกันก็ได้ โดยการจัดโคออดิเนตของพื้นผิวแต่ละส่วนลงบนภาพตามต้องการ ซึ่งเราสามารถทำได้อย่างอิสระ เพราะโปรแกรม มัลติเซฟ 3 มิติ มีเครื่องมือ "Texture Coordinate Editor" มาช่วยในการทำพื้นผิววัตถุด้วย



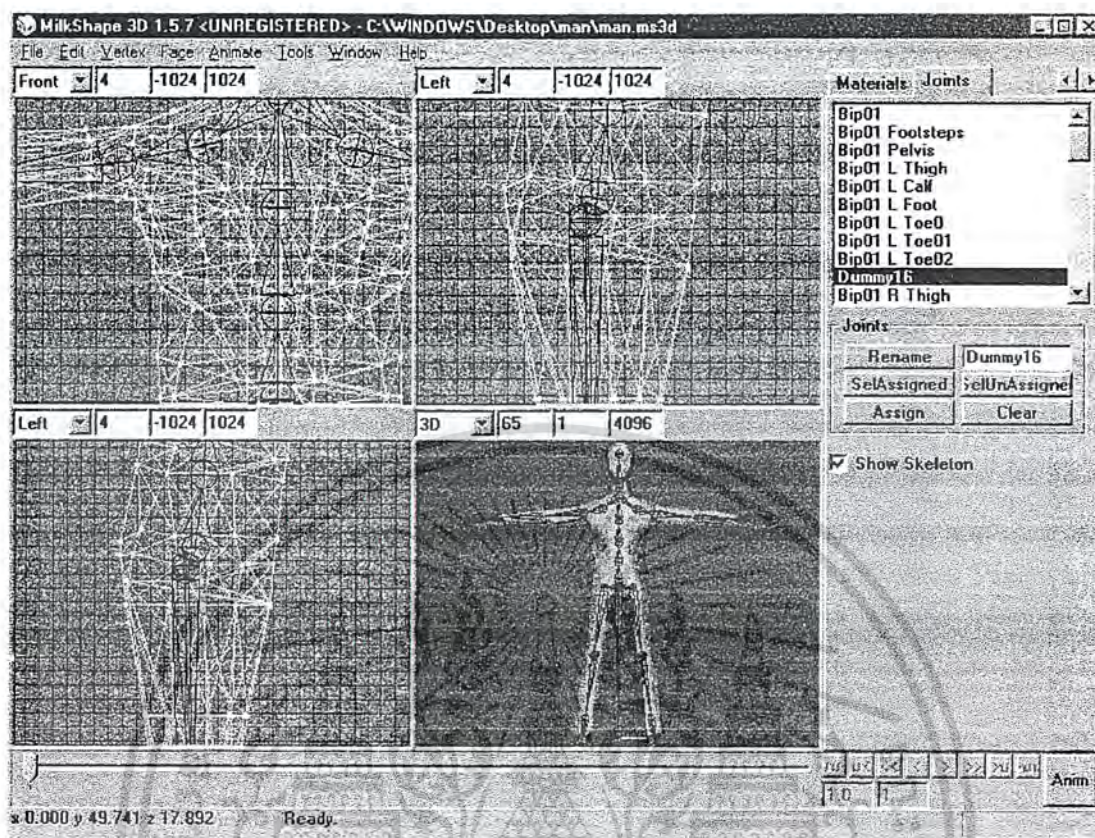
รูปที่ 7.4 แสดงส่วนที่ใช้ในการกำหนด วัสดุ (material) ใน มัลติเซฟ 3 มิติ



รูปที่ 7.5 แสดงเครื่องมือ Texture Coordinate Editor และการกำหนดโคออดิเนตให้พื้นผิว

7.4 การกำหนดกระดูกและข้อต่อให้โมเดล 3 มิติ

หากต้องการให้โมเดล 3 มิติทำการเคลื่อนไหวได้ เราต้องทำการบันทึกการเคลื่อนไหวของตัวละครเอาไว้ก่อน แล้วจึงนำมาแสดงตามเงื่อนไขต่างๆ ในโปรแกรม มัลติเซฟ 3 มิติ หากต้องการทำการเคลื่อนไหวตัวละคร 3 มิติ จำเป็นต้องทำการกำหนดกระดูกและข้อต่อให้โมเดลเสียก่อนว่ากระดูกส่วนไหนเชื่อมกับโมเดลส่วนใด



รูปที่ 7.6 แสดงการกำหนดข้อต่อและกระดูกใน มิลล์เชฟ 3 มิติ

7.5 การจัดการแอนิเมชันของโมเดล 3 มิติ

เมื่อเราทำการกำหนดกระดูกและข้อต่อของโมเดลทั้งหมดแล้ว เราจะสามารถบันทึกการเคลื่อนไหวของโมเดล 3 มิติได้ โดยการกำหนดจำนวนเฟรมขึ้นมาก่อนว่าจะให้มีการเคลื่อนไหวทั้งหมดกี่เฟรม ต่อจากนั้นก็ทำการกำหนดคีย์เฟรมขึ้นมาก็ได้เฟรมก็ได้แล้วแต่เรา กำหนดค่าทางลงไปในแต่ละคีย์เฟรมที่เราสร้างขึ้นมาตั้งแต่เฟรมแรกจนเฟรมสุดท้าย เมื่อทำการรันแอนิเมชัน โปรแกรมจะทำการคำนวณการเคลื่อนไหวระหว่างคีย์เฟรมให้เองโดยอัตโนมัติ

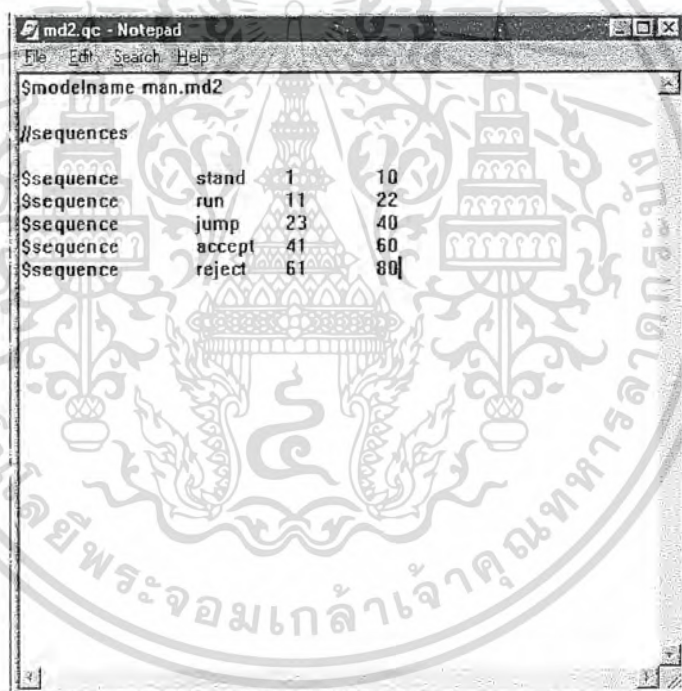


รูปที่ 7.7 ส่วนที่ใช้ในการทำแอนิเมชันใน มิลล์เชฟ 3 มิติ

7.6 การเอ็กซ์พอร์ตเพื่อนำไปใช้กับมอร์ฟิเทนจิน

มอร์ฟิเทนจินจะรู้จักไฟล์โมเดล 3 มิติที่มีนามสกุล .md2 โดยมีการบันทึกทั้งรูปร่าง โมเดลและแอนิเมชันการเคลื่อนไหวทั้งหมดไว้ภายในไฟล์ โดยแต่ละแอนิเมชันจะมีชื่อแอนิเมชันให้ทำการเรียกใช้ได้ ในความเป็นจริงแล้ว รูปแบบไฟล์ .md2 จะทำการเรียงค่อแอนิเมชันทั้งหมดเป็นแอนิเมชันเดียว โดยแต่ละเฟรมจะมีชื่อเฟรมอยู่ เฟรมที่มีชื่อนำหน้าเหมือนกันจะถูกจัดให้อยู่ในกลุ่มเดียวกัน ถือเป็นแอนิเมชันหนึ่งเช่น ทำยืน มีชื่อเรียกคือ "stand" มีแอนิเมชันในช่วงเฟรมที่ 1- 10 ของไฟล์สกุล .md2 ก็จะมีชื่อเฟรมที่ 1-10 คือ "stand01" ,"stand02" ,"stand03" ,..."stand10" หรือ ทำวิ่งมีชื่อเรียกคือ "run" มีแอนิเมชันในช่วงเฟรมที่ 11-22 ของไฟล์สกุล .md2 ก็จะมีชื่อเฟรมที่ 11-22 คือ "run01" ,"run02" , ... "run12"

ในโปรแกรม มัลล์เชฟ 3 มิติ นั้นสามารถเอ็กซ์พอร์ตโมเดล 3 มิติในรูปแบบไฟล์สกุล .md2 ได้ โดยจะต้องทำการเขียน QC ไฟล์ชื่อ md2.qc เพื่อกำหนดชื่อเฟรมแต่ละเฟรม ส่วนแอนิเมชันให้ทำการเรียงต่อกันทั้งหมดเป็นแอนิเมชันเดียว ตัวอย่างไฟล์ md2.qc แสดงไว้ดังรูปที่ 6-8



รูปที่ 7.8 ตัวอย่างไฟล์ md2.qc

สำหรับคำสั่งในไฟล์ md2.qc นั้นมีความหมายดังนี้

\$modelname <ชื่อไฟล์โมเดล>	เป็นการกำหนดชื่อไฟล์ของโมเดลที่จะทำการเอ็กซ์พอร์ต
\$sequence <ชื่อแอนิเมชัน> <เฟรมเริ่มต้น> <เฟรมสุดท้าย>	เป็นการกำหนดเฟรมให้แอนิเมชันว่ามีช่วงอยู่ในเฟรมที่เท่าไรบ้าง โดยเมื่อทำการเอ็กซ์พอร์ตเสร็จแล้วแต่ละเฟรมจะมีชื่อเป็น <ชื่อแอนิเมชัน> <หมายเลขเฟรมที่>
// <ข้อความ>	เป็นหมายเหตุ

ตารางที่ 7.1 แสดงคำสั่งในไฟล์ md2.qc



บทที่ 8

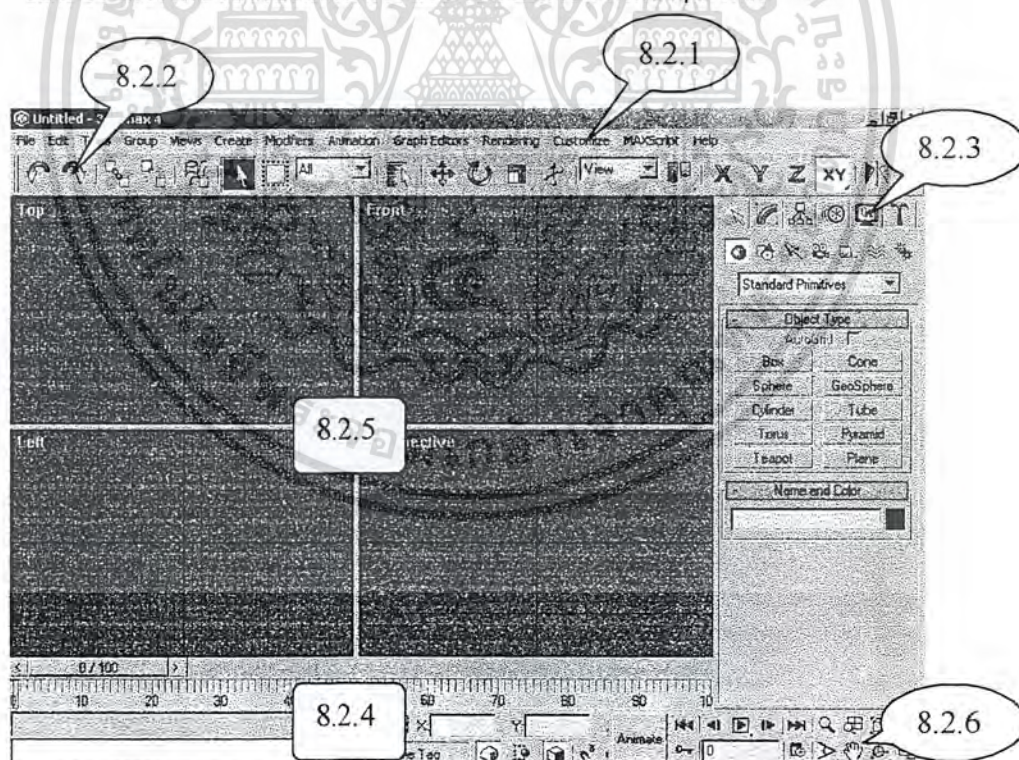
การสร้างโมเดล 3 มิติ

8.1 หลักการสร้างกราฟิก 3 มิติด้วยโปรแกรม 3 มิติ Studio Max

ในการทำงานบน 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 นั้นต้องทำความเข้าใจก่อนว่า กำลังติดต่อกับโลกเสมือนถูกสร้างขึ้นโดยคอมพิวเตอร์ ดังนั้นจึงต้องเข้าใจวิธีการแสดงและเก็บวัตถุต่างๆ ภายในโลกเสมือนนี้ด้วย คือ วัตถุทุกชิ้นมีโคออร์ดิเนตของตัวเอง และถูกสร้างขึ้นจากรูปทรงทางคณิตศาสตร์ หรือกล่าวได้ว่า วัตถุมีลักษณะเป็น 3 มิติ (สร้างและอยู่เฉพาะบนซอฟต์แวร์เท่านั้น) และทุกๆ วัตถุภายใน F โลกจำลอง มี 3 โคออร์ดิเนต คือ การแสดงความสูง ความกว้างและความลึกของจุดแต่ละ โคออร์ดิเนต จะเรียงตัวไปตามแกนเฉพาะ 1 แกน (axis) คือ ความกว้างถูกแสดงด้วยแกน X ความลึกถูกแสดงด้วยแกน Y และความสูงถูกแสดงด้วยแกน Z โดยจุดตัดของทั้ง 3 แกนนี้จะเป็นจุดเริ่มต้นของระบบแกนโคออร์ดิเนต จะมีพิกัด (0,0,0) เรานำมาระบุตำแหน่งต่างๆ ของวัตถุ 3 มิติใน โลกจำลอง ได้

8.2 โครงสร้างและหน้าจอการทำงานภายในโปรแกรม 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0

8.2.1 อินเทอร์เฟซของโปรแกรมนี้จะประกอบไปด้วยส่วนต่างๆ ดังนี้คือ



รูปที่ 8.1 แสดงหน้าจอการทำงานภายในโปรแกรม 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0

8.2.2 แถบเมนู (Menu bar) คือ แถบแสดงรายการคำสั่ง มีลักษณะเป็น แถบลงมาด้านล่าง จะเก็บคำสั่งที่สำคัญต่างๆ ทั้งหมดตามหัวข้อที่ได้แยกไว้

8.2.3 แทปเครื่องมือ (Tab Panel) เป็นส่วนที่รวบรวมและจัดหมวดหมู่ของกลุ่มคำสั่งโดยแบ่งคำสั่งออกเป็นกลุ่มๆ โดยปุ่มคำสั่งที่มีอยู่ในส่วนนี้ทั้งหมดจะมีอยู่ใน เครื่องมือควบคุม และ แถบเครื่องมือหลักด้วย

8.2.4 เครื่องมือควบคุม (Command Panel) เป็นส่วนหลักในการทำงานบน 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 ซึ่งใน เครื่องมือควบคุม จะมีการแบ่งคำสั่งออกเป็น 6 กลุ่มดังนี้

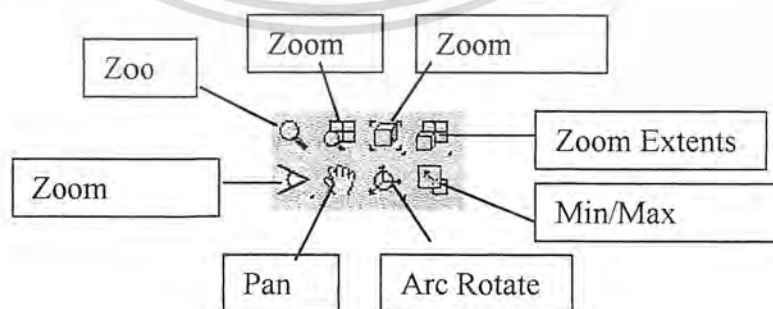
- 1) Create Command Panel
- 2) Modifier
- 3) Hierachy Command Panel
- 4) Motion Command Panel
- 5) Display Command Panel
- 6) Utility Command Panel

8.2.5 หน้าต่างแสดงสถานะ ประกอบไปด้วยส่วนของการทำแอนิเมชัน ตำแหน่งของเคอร์เซอร์ ตำแหน่งของเฟรม องศาการหมุน สถานภาพการทำงานขณะนั้น และการ snap

8.2.6 หน้าจอย่อย (Viewport) ในหน้าจอหลัก 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 สามารถเลือกที่จะแสดงหน้าจอย่อยได้พร้อมกันถึง 4 หน้าจอย่อยหรือแสดงเพียง 1 หน้าจอย่อยเท่านั้นก็ได้เช่นกัน หน้าจอย่อยใน 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 ได้แก่ Top Bottom Left Right Perspective และ camera (ถ้ามี) และคลิกเปลี่ยนให้แสดงหน้าจอย่อยใดๆ ได้โดยกดปุ่มตัวอักษรบนคีย์บอร์ด เช่น Top กด T, Bottom กด B, Left กด L, Right กด R, Front กด F, Back กด K, Perspective กด P, Camera กด C และ Spotlight กด S หรือคลิกเมาส์ขวาที่มุมบนซ้ายของหน้าจอย่อย ใดก็ได้เพื่อแสดงเมนู และสามารถเลือกหน้าจอย่อยได้

8.3 ส่วนควบคุมหน้าจอย่อย

ประกอบไปด้วยปุ่มคำสั่งควบคุมหน้าจอย่อยดังรูป



รูปที่ 8.2 แสดงส่วนควบคุมหน้าจอย่อย

Zoom

เมื่อคำสั่งย่อ – ขยายภาพบนหน้าจอย่อยที่ทำงานอยู่ คลิกเมาส์ลากขึ้นเป็นการขยายภาพ (Zoom – in) ลากลงคือย่อภาพ (Zoom – out)

Zoom all	คำสั่งนี้ เหมือนคำสั่งที่แล้ว แต่มีผลกับทุกหน้าจอย่อย
Zoom extents	เป็นของคำสั่งย่อ – ขยายภาพ แต่จะทำงานกว่าจะมองเห็นวัตถุทั้งหมดในหน้าจอย่อย และมีคำสั่ง Zoom Extents Selected เป็น flyout ขึ้นมาให้เลือก (ให้สังเกตว่าหากที่มุมล่าง – ขวาของปุ่มใดมีรูปสามเหลี่ยมเล็กๆ อยู่แสดงว่ามีปุ่ม flyout สำหรับเลือกคำสั่งเพิ่มเติม)
Zoom extents all	คำสั่งนี้ เหมือนกับคำสั่งที่แล้ว แต่มีผลกับทุกหน้าจอย่อยในเวลาเดียวกัน และมีคำสั่ง Zoom Extents All Selected เป็น flyout สำหรับเลือกเฉพาะวัตถุที่ต้องการ
Zoom Window	คำสั่งนี้คล้ายกับคำสั่ง Zoom แต่จะเป็นการปรับค่าพื้นที่ในการมองเห็นของหน้าจอย่อยเท่านั้น ซึ่งถ้าใช้คำสั่งบน Planar Viewport เช่น Top Right หรือ Front จะถูกแทนที่ด้วย Region Zoom หรือถ้าใช้คำสั่งบน Perspective หน้าจอย่อย จะถูกแทนที่ด้วย Field of View
Pan	คำสั่งเลื่อน หน้าจอย่อย โดยไม่มีผลต่อการย่อ – ขยายภาพ
Arc Rotate	คำสั่งนี้ ใช้หมุน หน้าจอย่อย ไปรอบๆ โดยใช้จุดศูนย์กลางของ หน้าจอย่อย เป็นแกนหมุนการใช้งานเมื่อคลิกเลือกคำสั่งนี้ จะปรากฏไอคอนสี่เหลี่ยมบนหน้าจอย่อย คลิกเมาส์ค้างไว้แล้วลากเมาส์ไปรอบๆทำให้ หน้าจอย่อย เปลี่ยนไปตามเมาส์ และมี Arc Rotate Selected เป็น flyout สำหรับเลือกเฉพาะวัตถุที่ต้องการเป็นแกนหมุน และ Arc Rotate SubObject เป็น flyout สำหรับเลือก material ที่กำหนดลงบนวัตถุหรือวัตถุย่อย (SubObject) เป็นแกนหมุน
Min / Max Toggle	คำสั่งนี้ ใช้ในการขยายภาพบนจอภาพย่อยที่ทำงานอยู่ให้แสดงเต็มจอภาพ เมื่อเราคลิกปุ่มนี้เข้าจอภาพที่ขยายเต็มจอภาพทำงานจะกลับสู่สภาพเดิม

8.4 การ เชื่อมต่อ (Merge) ไฟล์

การ โหลดไฟล์และรวมเข้ากับไฟล์ที่แสดงอยู่ หรือเลือกรวมเฉพาะวัตถุได้เช่นกัน โดยเลือก file (Menu Pull-Down) > Merge และ Browse ไฟล์ที่ต้องการ แล้วเลือกวัตถุที่ต้องการจากรายการ

8.5 การ อิมพอร์ต / เอ็กซ์พอร์ตไฟล์

สามารถโหลดไฟล์ในรูปแบบอื่นได้ ซึ่งใน โปรแกรม 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 สามารถ import และรองรับไฟล์ประเภท 3 D Studio Mesh (3 มิติS ,PRJ), ASE, DXF, STL, DWG, AI, IGE, SHP และ WRL และ เอ็กซ์พอร์ต เป็นไฟล์ประเภท เดียวกับที่สามารถ อิมพอร์ต เข้ามาได้ ซึ่งถ้าเราติดตั้ง ปลั๊กอินของมอร์ฟิทเข้าไปก็จะ เอ็กซ์พอร์ต เป็น WLD ได้เพิ่มอีกด้วย

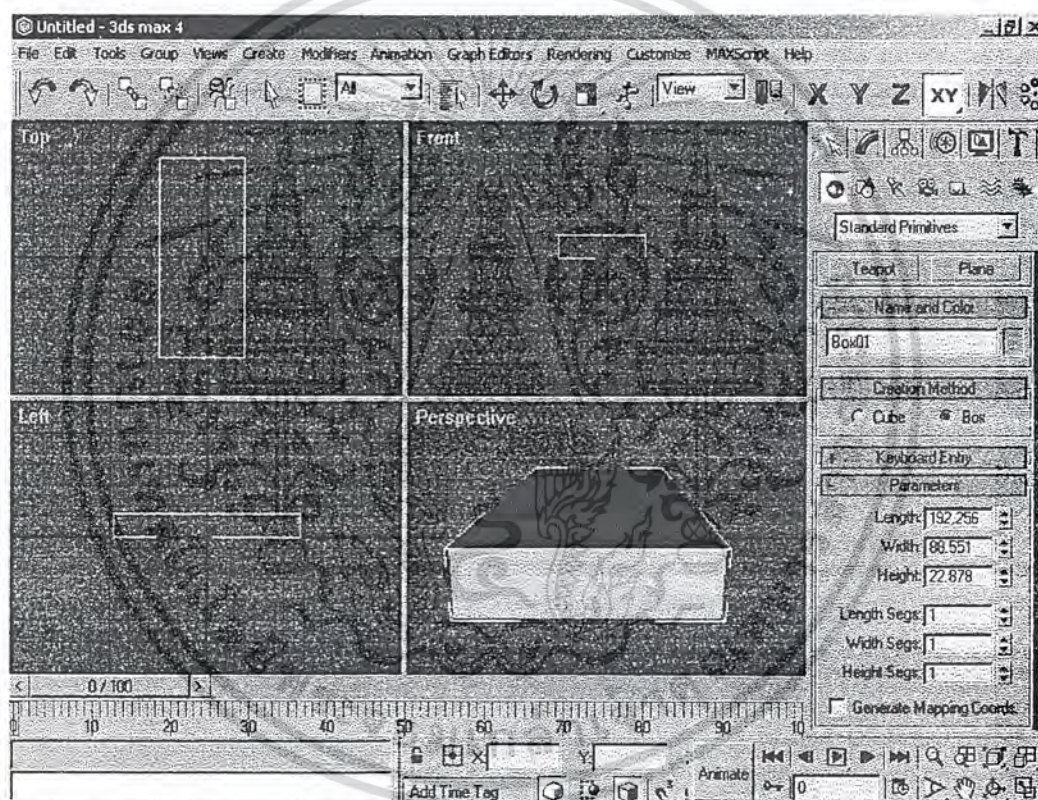
8.6 การเปลี่ยนไฟล์เป็น WLD

เราสามารถ เปลี่ยน จากไฟล์ 3 มิติS เป็น WLD ได้โดยใช้เครื่องมือที่มากับ โปรแกรมมอร์ฟิท คือ 3 มิติS2WLD โดยมันจะแปลงบิตแมปที่จำเป็นไว้ในโฟลเดอร์บิตแมปซึ่งจะอยู่ในโฟลเดอร์เดียวกับไฟล์ WLD ที่ เปลี่ยน ไป

8.7 การสร้างวัตถุ 3 มิติรูปทรงพื้นฐาน

การสร้างวัตถุ 3 มิติในโปรแกรม 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 มีวิธีการสร้างได้ 2 วิธี โดยจะ ยกตัวอย่างประกอบเป็นการสร้างวัตถุรูปกล่องสี่เหลี่ยม

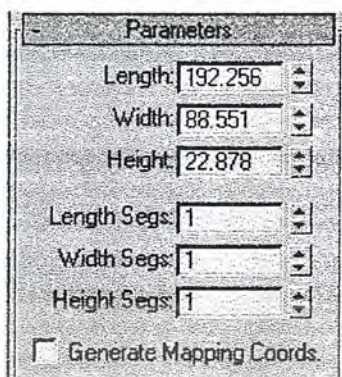
8.7.1 สร้างวัตถุโดยการใส่เมาส์ในจอภาพย่อย



รูปที่ 8.3 แสดงสร้างวัตถุโดยการใส่เมาส์ในจอภาพย่อย

- คลิกเมาส์ที่ปุ่ม Box
- เลื่อนเคอร์เซอร์มาที่จอภาพย่อยมุมมอง Top แล้วคลิกให้หน้าจอย่อยนี้ทำงาน
- คลิกเมาส์ แล้วคลิกวางไว้ ตามเคอร์เซอร์ไปยังตำแหน่งที่ต้องการ
- ปล่อยปุ่มเมาส์ แล้วเลื่อนเมาส์ขึ้นไปจนกล่องมีความสูงตามต้องการ

สามารถตั้งค่าต่างๆของกล่องได้ในแถบคำสั่ง Rollout หัวข้อ Parameters



รูปที่ 8.4 แสดงแถบคำสั่ง Rollout หัวข้อ Parameters

8.7.2 สร้างกล่องโดยการกำหนดค่าจากคีย์บอร์ด

- กำหนดจุดวางกล่องโดยกำหนดตัวเลข X , Y และ Z ในแถบคำสั่ง Rollout หัวข้อ Keyboard Entry
- กำหนดค่าความกว้าง ความลึก และความสูงของกล่องที่ต้องการสร้างใน Keyboard Entry
- คลิกปุ่ม Create วัตถุรูปกล่องสี่เหลี่ยมจะปรากฏขึ้น ณ ตำแหน่งที่เราได้กำหนดไว้

8.8 การกำหนดพื้นที่ผิววัตถุ

โดยปกติพื้นผิวของวัตถุ 3 มิติ โดยพื้นฐาน จะแบ่งออกเป็น 3 ส่วนดังตามที่แสดงในรูป



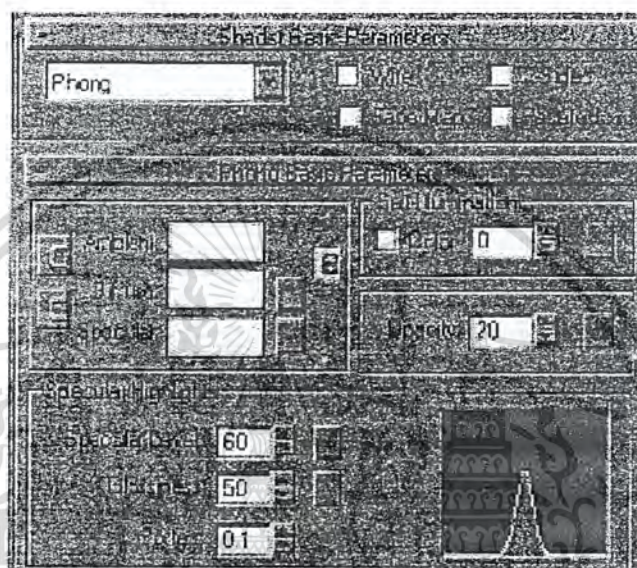
รูปที่ 8.5 แสดงคุณสมบัติพิเศษของวัตถุ

Specular

คือบริเวณพื้นผิววัตถุส่วนที่รับแสงในแนวตรงด้านหน้าสุดซึ่งบริเวณ specular นี้จะสะท้อนแสงจ้าที่สุดในพื้นผิวของวัตถุ

Diffuse	คือ พื้นผิวของสีพื้นฐานของวัตถุที่ถูกแสง และ ไล่เฉดของสีตามปริมาณแสงที่กระทบวัตถุ
Ambient	คือ บริเวณด้านเงามืดของวัตถุที่ไม่ถูกแสง หรืออยู่ด้านตรงข้ามกับแหล่งกำเนิดแสง

นอกจากนั้นยังต้องกำหนดคุณสมบัติทางกายภาพที่มีต่อแสงของวัตถุ เช่น ความเงา ความมันวาว ความโปร่งใสของวัตถุ เป็นต้น



รูปที่ 8.6 แสดงการกำหนดคุณสมบัติ

เนื่องจากมีวัตถุ 3 มิติบางชิ้นในโครงการนี้เป็นกระจก และ วัตถุที่สามารถสะท้อนแสง การกำหนดวัตถุจะช่วยให้วัตถุ 3 มิติที่สร้างขึ้นมีความเหมือนจริงมากยิ่งขึ้น ยกตัวอย่างเช่น กระจกเป็นวัตถุที่มีความโปร่งใส และสามารถสะท้อนแสงได้

บทที่ 9

การออกแบบโปรแกรมจำลองเสมือนจริง

9.1 ภาพรวมของโครงการ

โปรแกรมจำลองเสมือนจริงเป็นโปรแกรมสนทนาผ่านทางเครือข่ายเน็ตเวิร์ค โดยจะใช้การแสดงผลในรูปแบบของกราฟฟิก 3 มิติ ซึ่งผู้ใช้งานสามารถที่จะควบคุมตัวละครให้เคลื่อนไหวไปมาภายในสิ่งแวดล้อมที่สร้างขึ้นและจะทำการเลือกสนทนากับบุคคลอื่นๆได้ ซึ่งเมื่อเปรียบเทียบกับโปรแกรมสนทนาแบบเดิม เช่น Pirc, ICQ ฯลฯ จะเห็นว่าโปรแกรมเหล่านี้มันจะทำให้เพียงการสนทนาที่เป็นเท็กซ์โหมดเท่านั้น แต่ในโปรแกรมจำลองเสมือนจริงนี้ทำให้ผู้สนทนามีความรู้สึกได้เดินทางไปพูดคุยกับบุคคลอื่นเสมือนจริง นอกจากนั้นยังสามารถให้ผู้ใช้งานสามารถเลือกชุดแต่งกายของตัวละครได้ในทุกๆครั้งที่ทำการล็อกอิน ทำให้มีความรู้สึกแปลกใหม่น่าสนใจ และยังมีการออกแบบให้สามารถโฆษณาสินค้าภายในสิ่งแวดล้อมที่สร้างขึ้นได้อีกด้วย ซึ่งในจุดนี้ยังสามารถหารายได้ให้แก่ผู้พัฒนาโปรแกรมได้อีกทางด้วย

ในการออกแบบและการเขียนโปรแกรมประยุกต์ขึ้นนี้ เริ่มแรกจะทำการแยกพัฒนาโปรแกรมออกเป็น 2 ส่วน คือ

- การพัฒนาโปรแกรมประยุกต์ คลเอนต์ / เซิร์ฟเวอร์ พัฒนาโดยใช้โปรแกรมภาษา ภาษาจาวา เป็นหลักซึ่งจะทำการออกแบบ โปรแกรมสนทนาผ่านเครือข่ายเน็ตเวิร์ค
- การพัฒนาโปรแกรมกราฟฟิก 3 มิติ พัฒนาโดยใช้โปรแกรม 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0 , มิดลเซฟ 3 มิติ และใช้โปรแกรมภาษา วิซวลซี พลัส พลัส (Visual C++)

หลังจากนั้นจะทำการรวมการทำงานทั้ง 2 ส่วนเข้าด้วยกันเพื่อประยุกต์ออกมาเป็นโปรแกรมสนทนาผ่านทางเครือข่ายเน็ตเวิร์คซึ่งใช้การนำเสนอในรูปแบบกราฟฟิก 3 มิติ

9.2 การพัฒนาโปรแกรมประยุกต์ คลเอนต์ / เซิร์ฟเวอร์

การพัฒนาโปรแกรมในส่วนนี้จะใช้โปรแกรมภาษา ภาษาจาวา ในการพัฒนาในส่วนของการสนทนาผ่านทางเครือข่ายเน็ตเวิร์ค เนื่องจากกำลังเป็นที่นิยมและยังมีข้อดีกว่าภาษาอื่นๆในหลายๆด้าน ในการเขียนโปรแกรม ภาษาจาวา นี้จะทำการพัฒนาทั้งในส่วนของคลเอนต์ และ เซิร์ฟเวอร์ ซึ่งทั้งสองส่วนจะมีการทำงานแตกต่างกัน

9.2.1 ส่วนของคลเอนต์

ส่วนของคลเอนต์นี้จะเป็นส่วนการใช้งานของผู้สนทนาแต่ละคน โดยเริ่มแรกเมื่อผู้สนทนาได้เข้ามาใช้งานโปรแกรมเป็นครั้งแรก ก็จะต้องทำการ ลงทะเบียน ก่อนโดยต้องทำการกรอกข้อมูลส่วนตัวเพื่อนำข้อมูลเหล่านี้ไปเก็บยังเซิร์ฟเวอร์ หลังจากนั้นทุกๆครั้งที่เข้ามาใช้งานโปรแกรมก็จะล็อกอิน เข้าสู่ระบบ โดยจะต้องกรอก ผู้ใช้งานชื่อ และ รหัสผ่าน (ที่ได้ทำการกรอกไว้เมื่อตอน ลงทะเบียน) เมื่อเซิร์ฟเวอร์ได้ทำการเช็คค่า ผู้ใช้งานชื่อ และ รหัสผ่าน ถูกต้องผู้ใช้งานก็จะสามารถเข้าไปใช้งานโปรแกรมสนทนาได้เมื่อต้องการออกจากระบบก็จะต้อง ล็อกเอาต์เพื่อออกจากระบบ

9.2.2 ส่วนของเซิร์ฟเวอร์

ส่วนของเซิร์ฟเวอร์นี้จะเป็นส่วนที่ทำหน้าที่ในการเก็บข้อมูลต่างๆของไคลเอนต์รวมทั้งยังมีหน้าที่ในการตรวจสอบสถานะการทำงานของระบบและจัดการกับระบบได้ นอกจากนี้ยังมีหน้าที่สำคัญของเซิร์ฟเวอร์ดังนี้

- คอยจัดการติดต่อให้กับผู้ใช้โปรแกรมที่ต้องการติดต่อสนทนากับบุคคลอื่น และยังกำหนดรูปแบบในการส่งข้อมูลด้วย
- เซิร์ฟเวอร์มีความสามารถในการปรับปรุงเปลี่ยนแปลงสภาพแวดล้อมให้มีความเหมาะสมตามความต้องการได้ (Update Environment)
- เซิร์ฟเวอร์สามารถเก็บข้อมูลและจัดการกับข้อมูลของผู้ใช้โปรแกรมไว้ที่ฐานข้อมูล
- เซิร์ฟเวอร์มีความสามารถที่จะตรวจสอบและคอยดูแลประสิทธิภาพของระบบได้ (Check Performance)

9.3 การพัฒนาโปรแกรมกราฟฟิก 3 มิติ

ในขั้นตอนเริ่มต้นนั้นได้ทำการศึกษาหาข้อมูลของโปรแกรมที่เราจะนำมาใช้สร้างตัวโมเดล และนำมาจัดการกับ ไดเรกซ์เอกซ์ เปรียบเทียบประสิทธิภาพของโปรแกรมต่างๆจนคิดว่าได้โปรแกรมที่นำมาใช้ได้สะดวกและมีประสิทธิภาพมากที่สุดก่อน จากนั้นเราก็อย่เริ่มต้นออกแบบโมเดลตัวละคร 3 มิติและฉากที่เราจะใช้ขึ้นมา ซึ่งในขั้นต้นของการพัฒนาอาจยังไม่ต้องทำโมเดล 3 มิติที่ต้องการใช้จริง แต่ลองทำโมเดลง่ายๆมาทดสอบการทำงานของ มอร์ฟิเอนจิน ก่อน โดยเริ่มเขียน ฟังก์ชัน พื้นฐานของตัวละคร 3 มิติว่าผู้ใช้โปรแกรมสามารถควบคุมตัวละครได้อย่างไรบ้าง เช่น การเดิน, การกระโดด, การแสดงท่าทาง เป็นต้น เมื่อทดลองกับโมเดลง่ายๆได้แล้วจึงลงมือทำโมเดลที่ต้องการ และทำการเขียนโปรแกรมส่วนที่เหลือทั้งหมด เมื่อทำเสร็จแล้วจึงนำไปรวมกับส่วนของการพัฒนาโปรแกรมประยุกต์ไคลเอนต์/เซิร์ฟเวอร์ ส่วนในการทำโมเดล 3 มิติ เราได้เลือกใช้โปรแกรม 3 มิติในการทำดังนี้คือ

9.3.1 โปรแกรม 3 มิติ สตูดิโอ แมกซ์ เวอร์ชัน 4.0

ซึ่งโปรแกรมนี้เราได้ใช้ในการสร้างโลก 3 มิติ (โลก) เนื่องจากโปรแกรม 3 มิติ Studio Max นี้มีความสามารถในการจัดเก็บ โพลีกอนที่มีความซับซ้อนและมีจำนวนมากได้เป็นอย่างดีจึงเหมาะสมที่จะใช้ในการสร้างโลก 3 มิติซึ่งมีรายละเอียดสูง

9.3.2 โปรแกรม มิดล์เซฟ 3 มิติ

ซึ่งโปรแกรมนี้เราได้ใช้ในการสร้างโมเดลตัวละคร 3 มิติ เนื่องจากโปรแกรมนี้มีจุดเด่นอยู่ที่การที่จะสามารถจัดการกับแอนิเมชันต่างๆของตัวละครได้เป็นอย่างดี อีกทั้งยังสามารถเอ็กซ์พอร์ตไฟล์โมเดลที่สร้างจาก โปรแกรมนี้ออกมาได้หลายฟอร์แมตซึ่งสามารถนำไปใช้ในมอร์ฟิเอน

9.4 การนำโปรแกรมในส่วน 3 มิติและส่วนเน็ตเวิร์กมารวมกัน

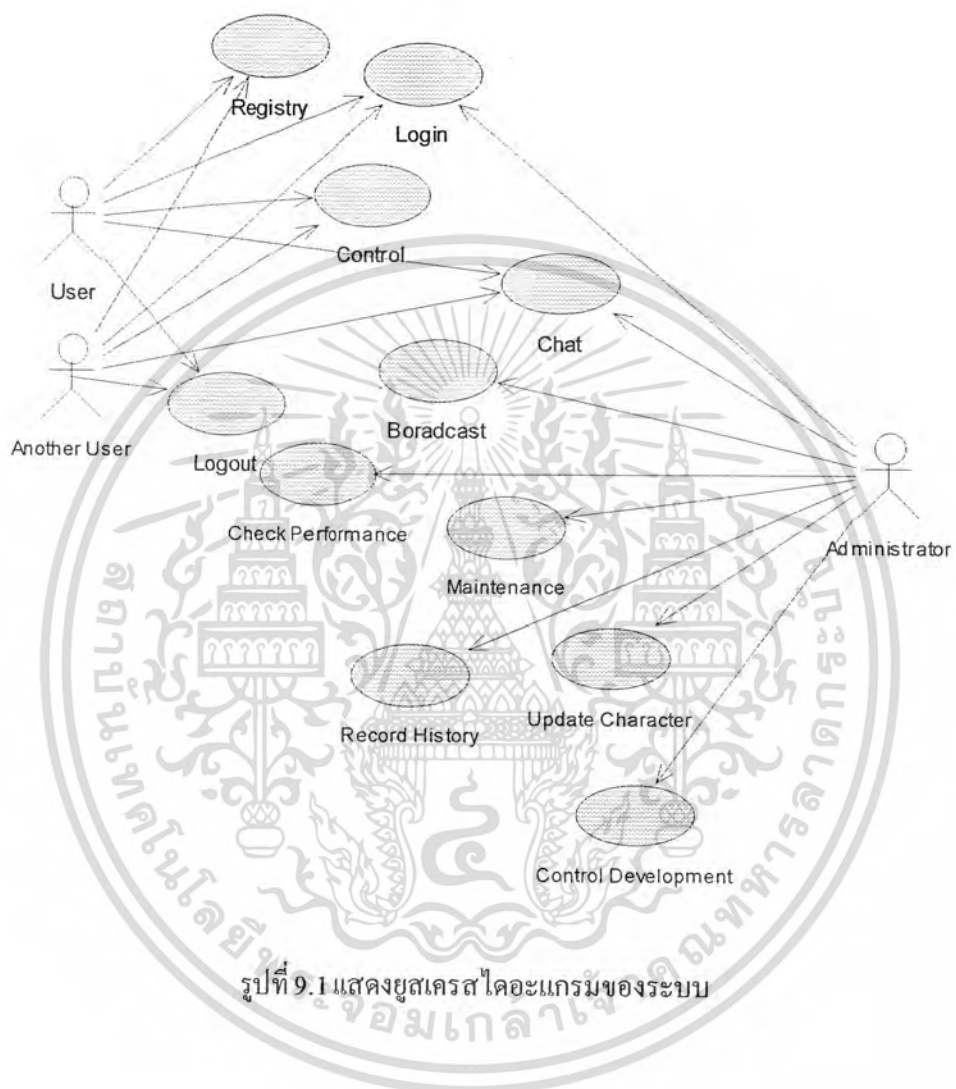
ในตอนที่ 2 เราได้เริ่มนำโปรแกรมทั้งสองส่วน คือ ส่วนที่สร้างภาพ 3 มิติและควบคุมการทำงานของตัวละคร อีกส่วนคือ ระบบติดต่อส่งข้อมูลกันบนเน็ตเวิร์กมารวมกัน โดยใช้ จาวาเน็ตไจฟอินเตอร์เฟซ (Java native Interface(JNI)) ในการรวมภาษาจาวา และภาษาซีเข้าด้วยกัน การออกแบบให้รวมกันได้โดยเราใช้ภาษาจาวาสร้างออบเจกต์สำหรับสร้างภาพ 3 มิติที่เขียนโดยภาษาซี และใช้การรอรับสัญญาณที่ส่งเข้ามาจากเครื่องอื่นๆแล้วจึงนำมาวาดลงบนเวดจ์ของผู้ใช้งานแต่ละเครื่อง

ในการเคลื่อนไหวของตัวละครแต่ละครั้งจะมีการตรวจสอบการเคลื่อนที่ว่าไปในทิศทางใดในภาษาซี จากนั้นส่งค่านั่นๆข้ามไปยังเซิร์ฟเวอร์ ตัวเซิร์ฟเวอร์จะตรวจสอบว่าการค่าที่ส่งมานั้นทำงานใด เช่น เดินหน้า ถอยหลัง คุยแบบเป็นกลุ่ม คุยแบบส่วนตัว เป็นต้น จากนั้นจึงส่งข้อมูลนี้ไปยังเครื่องปลายทาง

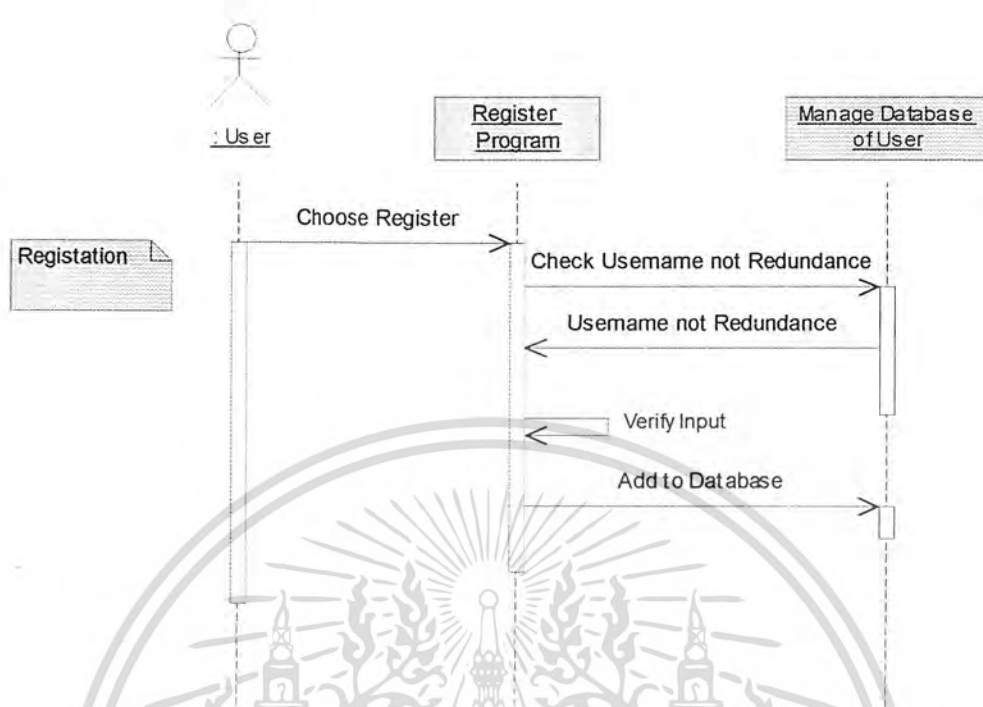
ในการส่งค่าไปทำงานแต่ละครั้งจะมีการส่งชื่อของผู้ใช้งานคนนั้นๆไปยังเครื่องต่างๆที่เข้ามาเล่น ก่อนหน้านี้ทุกเครื่อง สมมติว่ามีผู้ใช้งานคนใหม่เข้ามาในระบบ เซิร์ฟเวอร์จะส่งชื่อ , ตำแหน่ง และ รูปร่างของโมเดลของผู้ที่เข้ามาใหม่ไปยังทุกๆเครื่องที่เข้ามาเล่นก่อนหน้านี้ จากนั้นทุกๆเครื่องจะเก็บชื่อของผู้เล่น ตำแหน่ง และ สร้างออบเจกต์ของผู้ที่เข้ามาใหม่ ในภายหลังเมื่อผู้เล่นคนนั้นเคลื่อนไหวหรือทำงานใดๆ ทุกๆเครื่องก็จะแก้ไขค่าต่างๆของผู้เล่นคนนั้น และเลื่อนตำแหน่งของออบเจกต์ตัวนั้นไปยังตำแหน่งที่เปลี่ยนแปลงไป

ส่วนแนวคิดในการออกจากระบบทำได้โดยลบออบเจกต์ทันทีที่ผู้ใช้งานออกจากระบบไปแล้ว

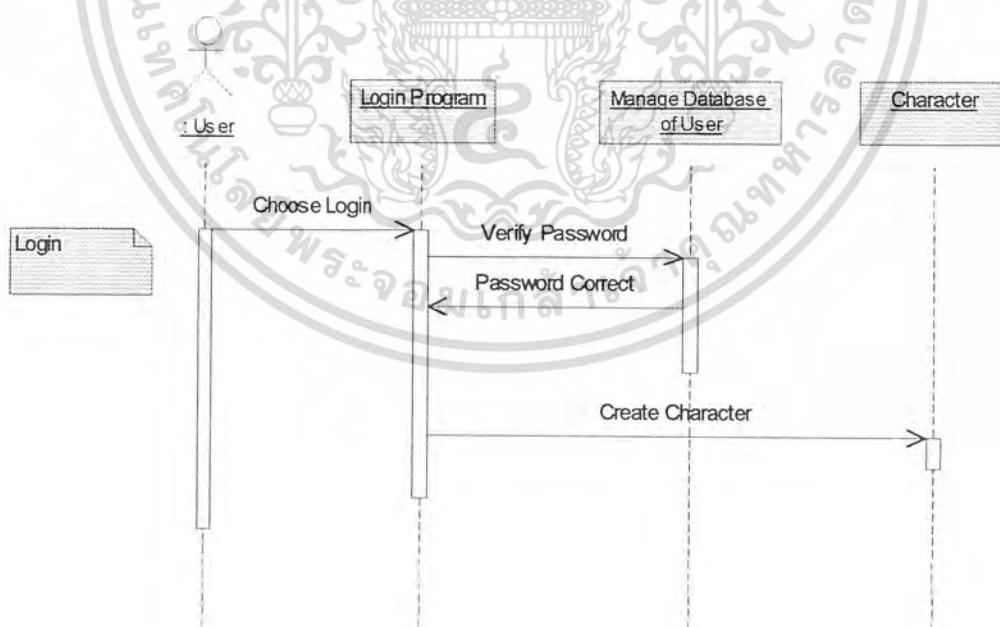
9.5 แผนภาพต่างๆ ของการออกแบบระบบ (Diagram)



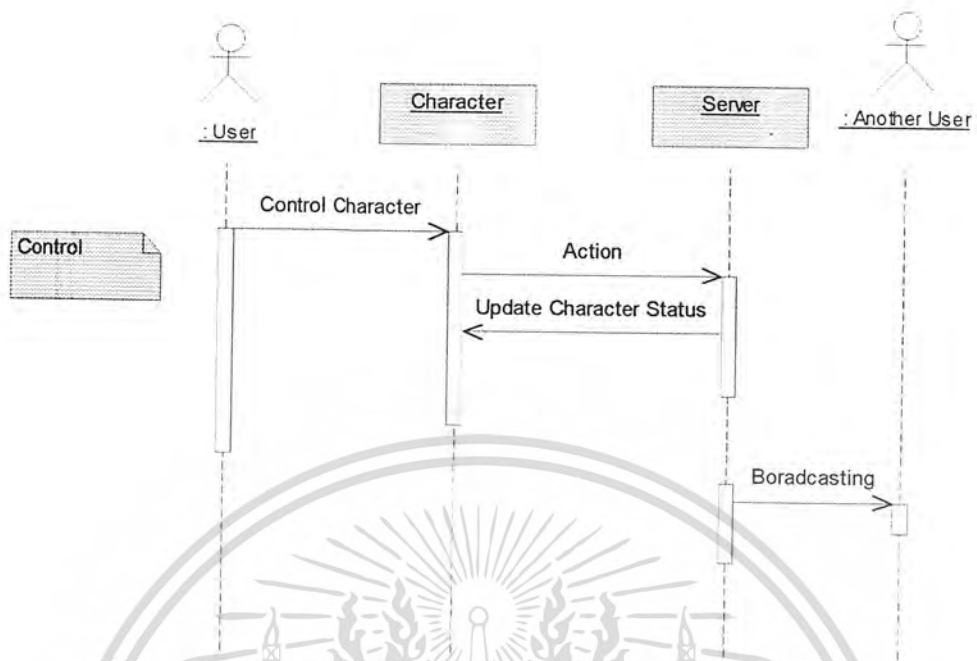
รูปที่ 9.1 แสดงยูสเคสไคอะแกรมของระบบ



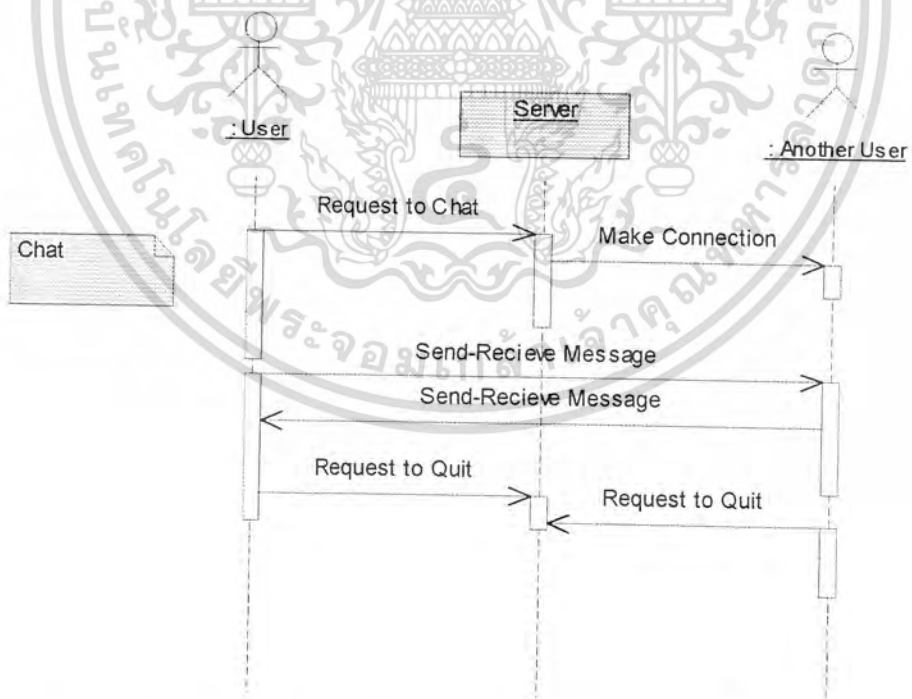
รูปที่ 9.2 แสดงซีควเอนไคอะแกรมของการสมัครเป็นสมาชิก



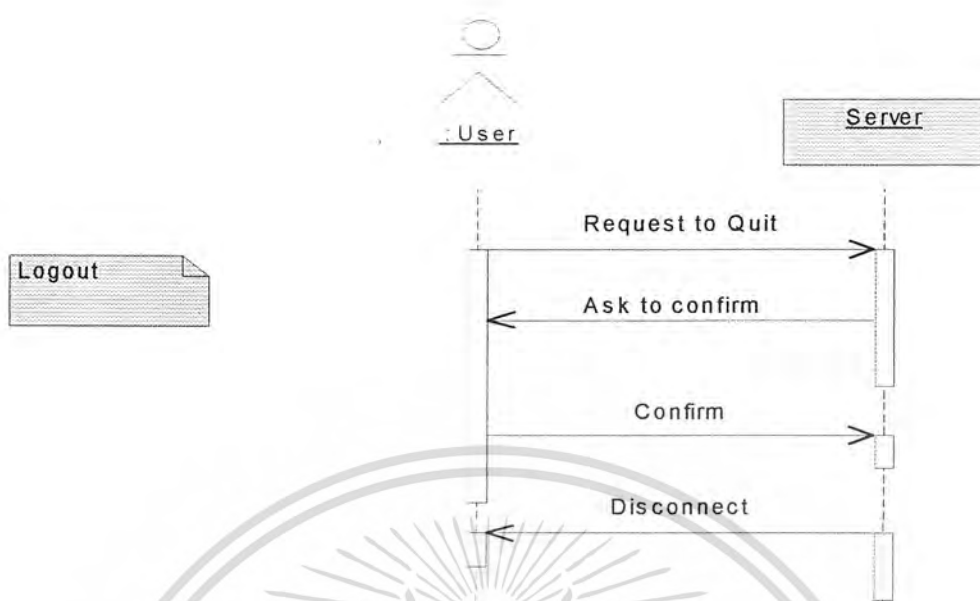
รูปที่ 9.3 แสดงซีควเอนไคอะแกรมของการล็อกอินของผู้ใช้



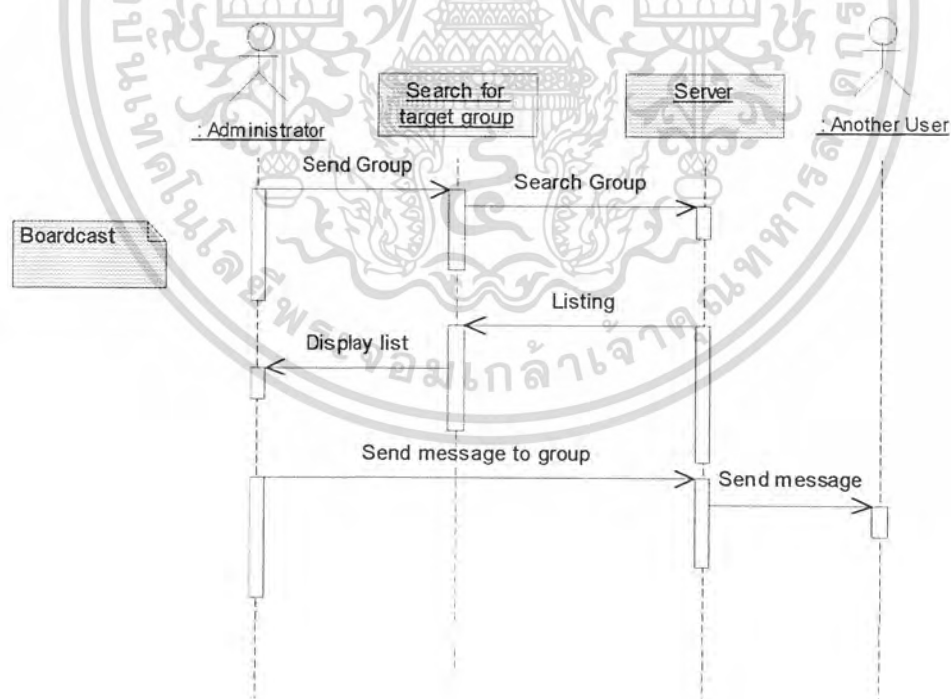
รูปที่ 9.4 แสดงซีควเอนไคอะแกรมของการควบคุมตัวละคร



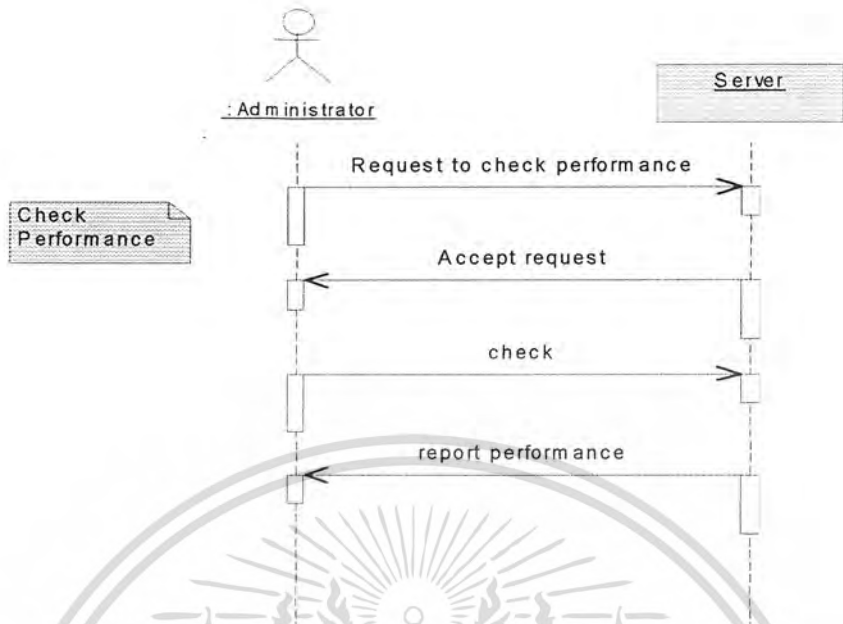
รูปที่ 9.5 แสดงซีควเอนไคอะแกรมของการสนทนาผ่านทางเครือข่ายของผู้ใช้



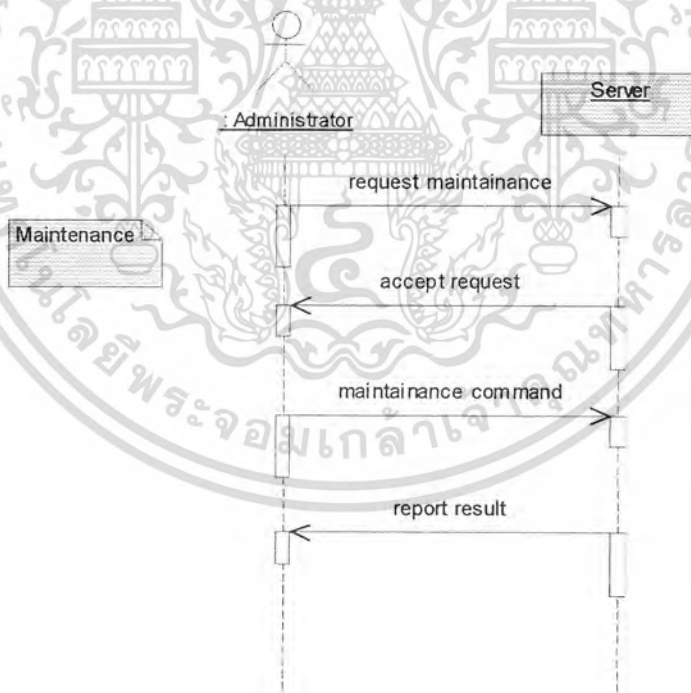
รูปที่ 9.6 แสดงซีเควนไดอะแกรมของการออกจากระบบของผู้ใช้



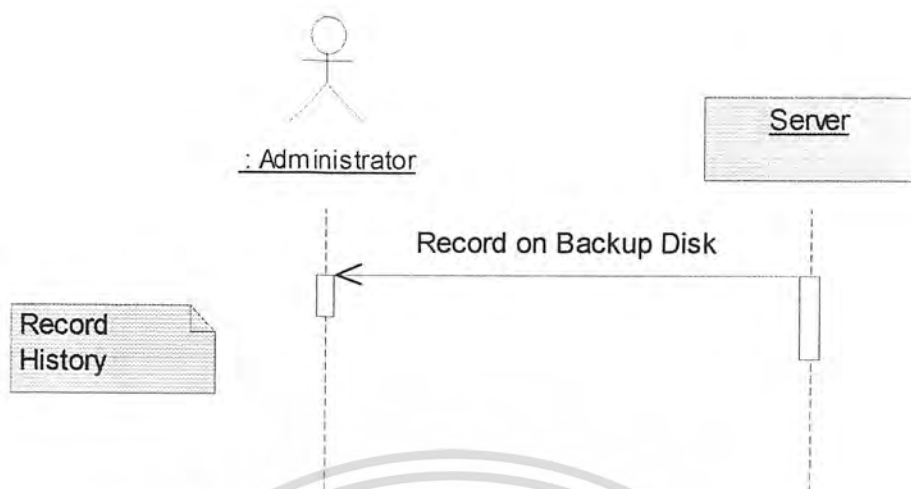
รูปที่ 9.7 แสดงซีเควนไดอะแกรมของการกระจายข่าวของผู้ควบคุมระบบ



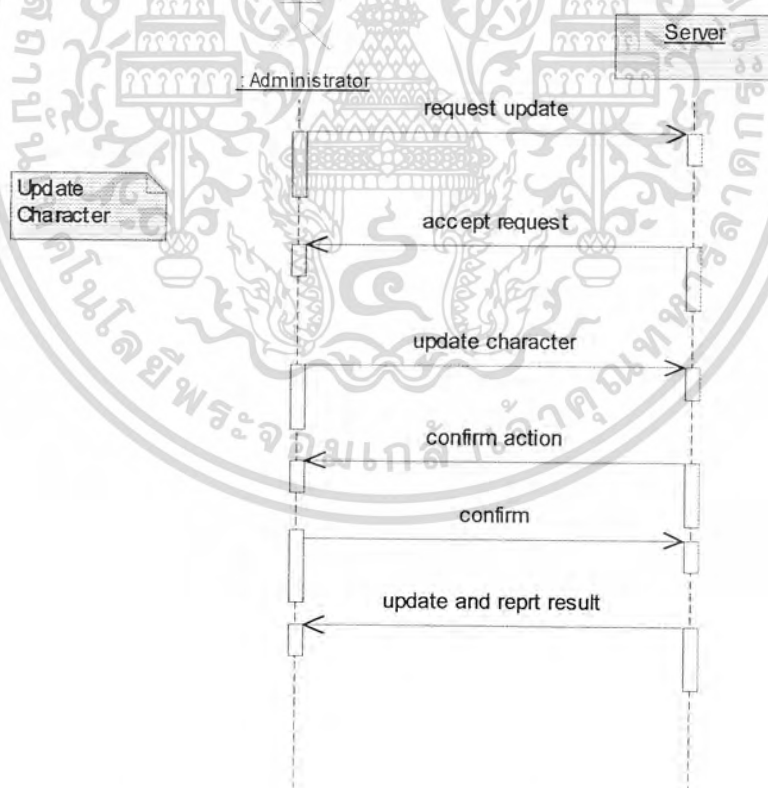
รูปที่ 9.8 แสดงซีควเอนโคอะแกรมของการตรวจสอบประสิทธิภาพระบบของผู้ควบคุมระบบ



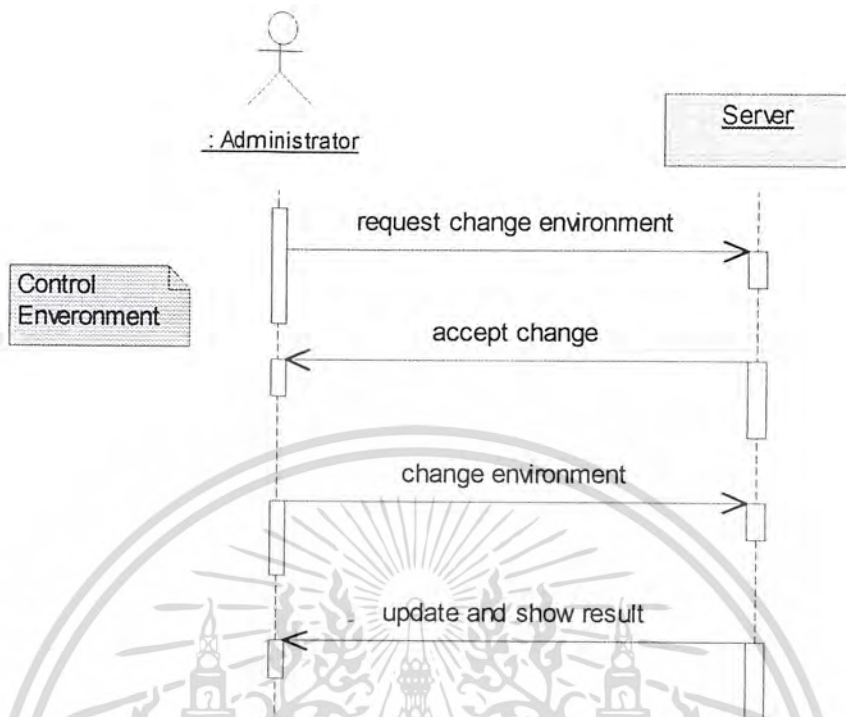
รูปที่ 9.9 แสดงซีควเอนโคอะแกรมของการดูแลรักษาระบบของผู้ควบคุมระบบ



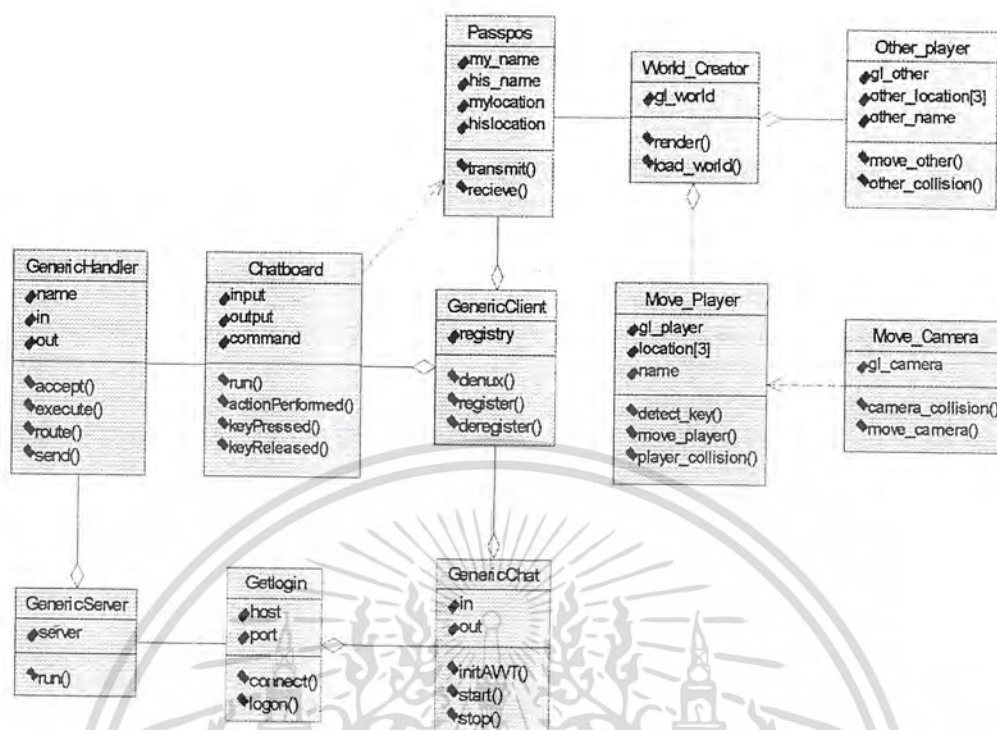
รูปที่ 9.10 แสดงซีเควนไดอะแกรมของการสำรองข้อมูลผู้ใช้ของผู้ควบคุมระบบ



รูปที่ 9.11 แสดงซีเควนไดอะแกรมของการเปลี่ยนแปลงข้อมูลตัวละครของผู้ควบคุมระบบ



รูปที่ 9.12 แสดงซีควเอนโคอะแกรมของการเปลี่ยนแปลงสิ่งแวดล้อมของผู้ควบคุมระบบ



รูปที่ 9.13 แสดงคลาสไคอะแกรมของโปรแกรมจำลองเสมือนจริง (Virtual Center Point)

บทที่ 10

ปัญหาและการแก้ไข

10.1 การใช้งานร่วมกันของภาษาจาวาและซี

เนื่องจากแนวคิดที่เกิดจากทั้งสองภาษานั้นแตกต่างกัน คือ ในภาษาจาวานั้นจะมีการคิดแบบ ออบเจกต์-โอเรียนเต้ดแต่ในภาษาซีนั้นยังคงเป็นแบบเดิมๆอยู่ แต่ก็ยังมีส่วนคล้ายกันอยู่ คือในภาษาซีนั้นยังมีการเรียกเป็นคาส ซึ่งคล้ายคลึงกับภาษาจาวา เราจึงสามารถดัดแปลงมาใช้ร่วมกันได้ ในช่วงแรกนั้นเรายังไม่ความรู้ในการรวมกันของทั้งสองภาษามากนัก เมื่อเกิดปัญหาขึ้นจึงเกิดความเบื่อหน่ายที่จะทำงาน แต่ในภายหลังเรามีความชำนาญมากขึ้นจึงสามารถผ่านอุปสรรคต่างๆมาได้ ตัวอย่างของปัญหาเช่น

เดิมในภาษาซีนั้น เราใช้การวนรับป้อนในการเคลื่อนไหว ปัญหาที่เกิดขึ้นคือ เมื่อเรารับป้อนก็จะไม่สามารถหยุดคุยกับเครื่องอื่นไปด้วย (ทางผู้จัดทำต้องการให้สามารถคุยกับเครื่องอื่นและเคลื่อนไหวไปพร้อมๆกันได้) ทางทีมงานจึงได้ปรับปรุงโดยนำการวนนั้นออก และนำภาษาจาวามาครอบไว้อีกโดยกำหนดให้กลุ่มของฟังก์ชันชุดที่ควบคุมการเคลื่อนไหวทั้งหมดเป็นหนึ่งออบเจกต์ จึงสามารถแก้ไขปัญหานี้ได้

10.2 ปัญหาเนื่องจากโพลีกอนและผู้เล่นเยอะเกินไป

เมื่อมีการเข้ามาเล่นจะเกิดการสร้างโพลีกอนใหม่ขึ้นมาบนเครื่องคอมพิวเตอร์ของผู้เล่นแต่ละคน ทำให้ความเร็วในการแสดงผลลดลงอย่างมาก โดยในปกติจะมีการแสดงผลอยู่ในช่วง 20 – 30 เฟรมต่อวินาที ถ้ามีผู้เล่นอื่นเข้ามาเล่นก็ทำให้แสดงผลได้เพียง 10 – 18 เฟรมต่อวินาที แม้เปลี่ยนความละเอียดก็มีผลต่อความเร็วของการแสดงผลน้อยมาก ทางผู้จัดทำจึงตั้งนิยามว่าน่าจะเป็นเพราะอัลกอริทึมของเอนจินที่ใช้ในการทราแซฟอรั่มโพลีกอน มีประสิทธิภาพต่ำและใช้ทรัพยากรของระบบสูงเกินไป อีกทั้งตัวโพลีกอนที่สร้างขึ้นมาเพื่อเป็นสิ่งแวดล้อมนั้นก็มีจำนวนมากอยู่แล้ว จึงยังทำให้แสดงผลช้าลงไปอีก

10.3 ขีดจำกัดของจำนวนสีของบิตแมปที่ใช้เป็นฉากหลัง

ในปัจจุบันมีการใช้เท็กซ์เจอร์ขนาด 16-24 บิตสี หรือ 65536-16.7 ล้านสี เป็นมาตรฐานในการแสดงผล แต่ในตัวเอนจินเรานี้สามารถแสดงผลได้เพียง 256 สีเท่านั้น จึงทำให้ภาพที่ออกมาไม่สมจริงเท่าที่ควร

10.4 การเข้าถึงแอนิเมชันของโมเดลเอ็มดีทู (MD2)

หลังจากเรียกใช้โมเดลเอ็มดีทูขึ้นมาแล้ว หากต้องการ โหลดโมเดลเอ็มดีทูขึ้นมาอีกตัวสามารถทำได้ แต่จะไม่สามารถเข้าถึงข้อมูลแอนิเมชันของ โมเดลเอ็มดีทูตัวที่สองได้หากโมเดลเอ็มดีทูตัวที่สองนั้นเป็น โมเดลเดียวกับตัวแรก แม้ว่าจะโหลดมาจากไฟล์ตำแหน่งที่มีชื่อต่างกันก็ตาม ดังนั้นทางผู้จัดทำจึงแก้ไขโดยสร้างฐานข้อมูลเพื่อเก็บข้อมูลของผู้เล่นแต่ละคน ซึ่งก็สามารถแก้ปัญหานี้ได้เป็นอย่างดี

10.5 ข้อบกพร่องของการตรวจสอบการชนโดยใช้ฟังก์ชันของมอร์ฟิท

เนื่องจากฟังก์ชันตรวจสอบการชนบนมอร์ฟิทมีลักษณะการทำงานดังนี้ คือ ให้กำหนดจุดเริ่มต้นและจุดปลายของวัตถุที่ต้องการจะตรวจสอบ โดยบอกค่าแกนเคิลของวัตถุที่ต้องการตรวจสอบ จากการที่ทราบจุดเริ่มและจุดปลายก็จะสามารถนำจุดทั้งสองมาสร้างเส้นตรงได้ แล้วนำเส้นตรงมาตรวจสอบว่ามี การตัดโพลีกอนใดหรือไม่ ถ้าไม่ฟังก์ชันจะส่งกลับค่า YES แต่ถ้าไม่ตัดจะส่งค่า NO พร้อมทั้งบอกจุดตัดกับโพลีกอน, แกนเคิลของโพลีกอน และ วัตถุที่เส้นตรงตัดกลับคืนมา แต่เนื่องจากวัตถุในโลก 3 มิติเป็น ปริมาตร ทำให้เกิดปัญหาในบางที่เมื่อวัตถุเคลื่อนเข้าหากันโดยที่มีบางส่วนซ้อนทับกันอยู่แต่จุดเส้นตรงไม่ตัดกัน โพลีกอนจึงสามารถเดินผ่านกันได้ดังนั้นจึงเกิดปัญหา

บทที่ 11

ผลลัพธ์ของโครงการ

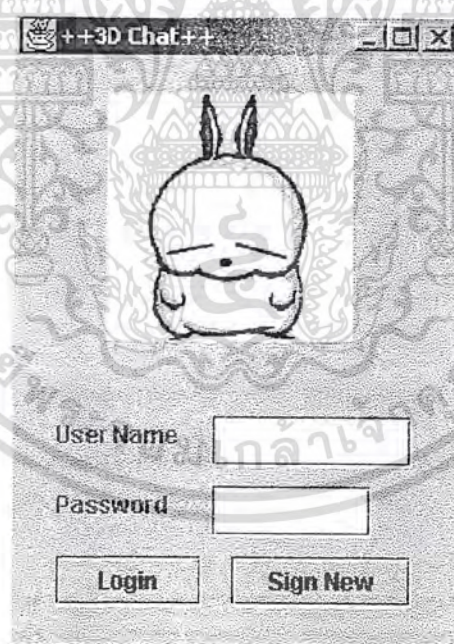
11.1 รูปแบบของโครงการ

โครงการนี้เป็นโครงการที่แสดงผลออกมาในรูปแบบของ 3 มิติการใช้งานจะเป็นการพูดคุยกันระหว่างผู้ใช้บริการ โปรแกรมผ่านบนเครือข่ายท้องถิ่น (LAN) โดยสามารถเลือกตัวละครและเปลี่ยนสภาพแวดล้อมภายในโลกจำลอง (world)

11.2 คู่มือการใช้งาน

สำหรับผู้ใช้งานทั่วไป

เมื่อเราเริ่มเปิดโปรแกรมขึ้นมาจะปรากฏหน้าจอสำหรับขอเข้าใช้บริการดังรูปที่ 11.1 โดยจะมีแบบฟอร์มสำหรับรับข้อมูลชื่อผู้ใช้กับรหัสผ่าน จากนั้นผู้ใช้จึงกดที่ปุ่ม login เพื่อทำการเข้าใช้บริการโปรแกรม แต่ถ้ายังไม่ได้ลงทะเบียนต้องทำการลงทะเบียนเพื่อขอใช้บริการก่อน โดยกดที่ปุ่ม sign new เพื่อทำการลงทะเบียนกับทางผู้ให้บริการ



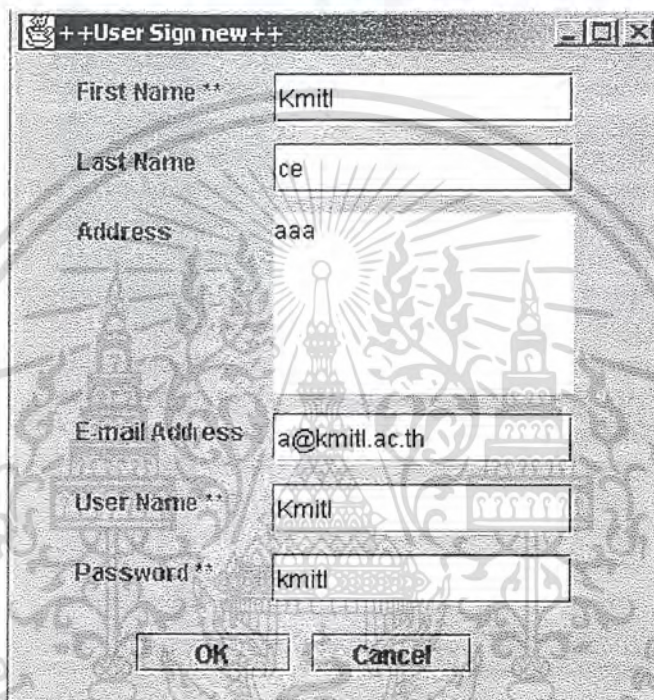
รูปที่ 11.1 แสดงหน้าจอแรกเมื่อเริ่มเปิดโปรแกรม

เมื่อกดที่ปุ่ม sign new แล้วจะปรากฏหน้าจอที่มีแบบฟอร์มสำหรับให้ผู้ขอใช้บริการ โปรแกรมกรอกข้อมูลส่วนตัวดังรูปที่ 11.2 เพื่อนำไปเก็บที่ฐานข้อมูลของเซิร์ฟเวอร์ซึ่งมีข้อมูลให้กรอกดังนี้

- ชื่อ
- นามสกุล

- ที่อยู่
- อีเมลแอดเดรส
- ชื่อผู้ใช้
- รหัสผ่าน

โดยช่องที่มีสัญลักษณ์ดอกจัน 2 คงเป็นช่องข้อมูลที่คุณขอใช้บริการต้องกรอกมิฉะนั้นจะทำการลงทะเบียนไม่ได้ เมื่อกรอกเสร็จเรียบร้อยแล้วให้กดปุ่ม ok ถ้าต้องการยกเลิกกดปุ่ม cancel



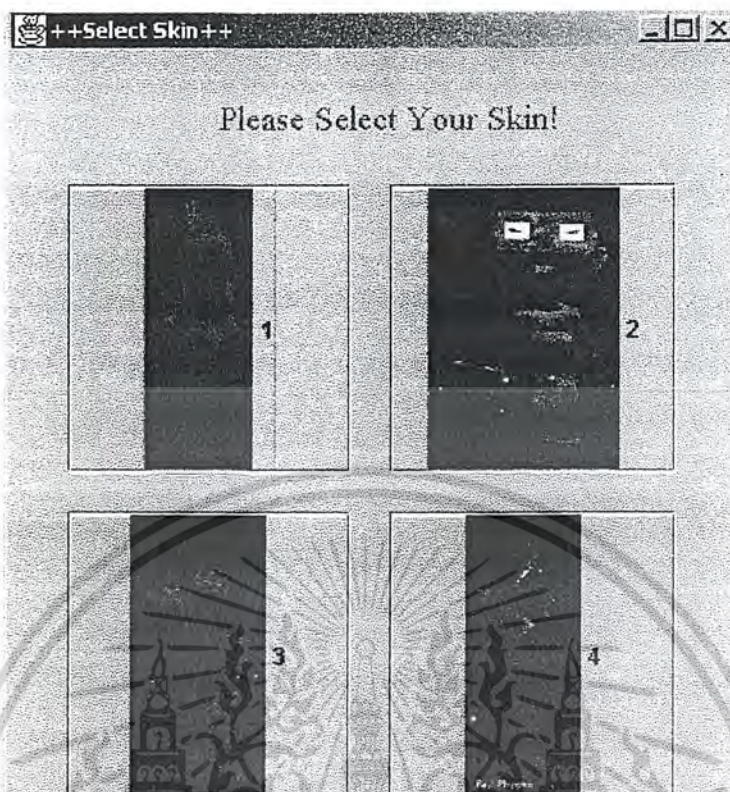
The image shows a 'User Sign new' dialog box with the following fields and values:

Field	Value
First Name **	Kmittl
Last Name	ce
Address	aaa
E-mail Address	a@kmitl.ac.th
User Name **	Kmittl
Password **	kmitl

At the bottom of the dialog box are two buttons: 'OK' and 'Cancel'.

รูปที่ 11.2 หน้าจอสำหรับลงทะเบียนขอใช้บริการ

เมื่อลงทะเบียนเสร็จแล้วจึงสามารถเข้าใช้บริการได้ ผู้ใช้บริการจะเห็นหน้าจอสำหรับการเลือกตัวละครของคนที่ใช้ในโปรแกรมดังรูปที่ 11.3 โดยจะมีให้เลือก 4 ตัว



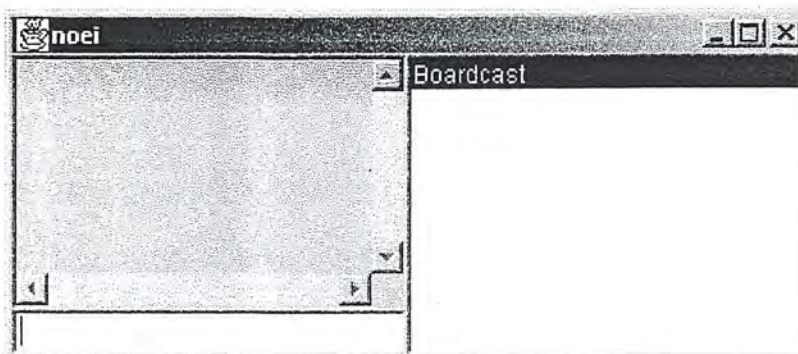
รูปที่ 11.3 หน้าจอสำหรับเลือกตัวละครของผู้ใช้

จากนั้นจะเห็นหน้าจอที่ให้ผู้ให้บริการเลือกสีเสื้อผ้าของตัวละครที่ผู้ใช้เลือก ซึ่งมีให้เลือกทั้งหมด 4 แบบดังรูปที่ 11.4

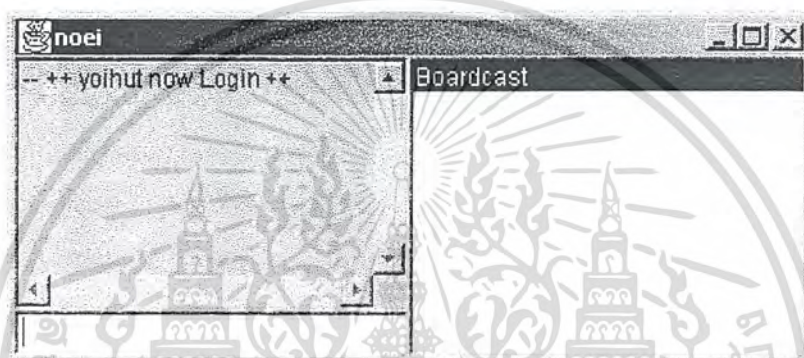


รูปที่ 11.4 แสดงหน้าจอเลือกสีเสื้อผ้าตัวละคร

ต่อมาจะเข้าสู่โปรแกรม โดยจะปรากฏหน้าต่างสำหรับการพูดคุยดังรูปที่ 11.5 และถ้ามีผู้เล่นคนอื่นเข้ามา ก็จะขึ้นข้อความมาแจ้งว่าใครเพิ่งเข้ามาใช้บริการดังรูปที่ 11.6 ตัวละครที่ผู้ใช้เลือกอยู่ในโลกจำลองบนหน้าจอ ดังรูปที่ 11.7



รูปที่ 11.5 แสดงหน้าต่างการพูดคุย



รูปที่ 11.6 แสดงการแจ้งข้อความเมื่อมีผู้เล่นคนอื่นเข้ามา



รูปที่ 11.7 หน้าต่างแสดงตัวละครบนโลกจำลอง

ในการบังคับตัวละครบนโลกจำลองนั้นมีการบังคับดังนี้

- กดปุ่มลูกศรขึ้น เดินหน้า
- กดปุ่มลูกศรลง ถอยหลัง
- กดปุ่มลูกศรซ้าย หันซ้าย
- กดปุ่มลูกศรขวา หันขวา

โดยผู้เล่นสามารถบังคับตัวละครของตัวเองเดินไปไหนก็ได้ภายในโลกจำลองและถ้าผู้เล่นต้องการที่จะพูดคุยกับผู้เล่นคนอื่นสามารถทำได้โดยการพิมพ์ข้อความลงในหน้าต่างการพูดคุยในรูปแบบที่ 11.5 โดยผู้เล่นสามารถเลือกรูปแบบการส่งข้อความได้ 2 แบบ ดังนี้

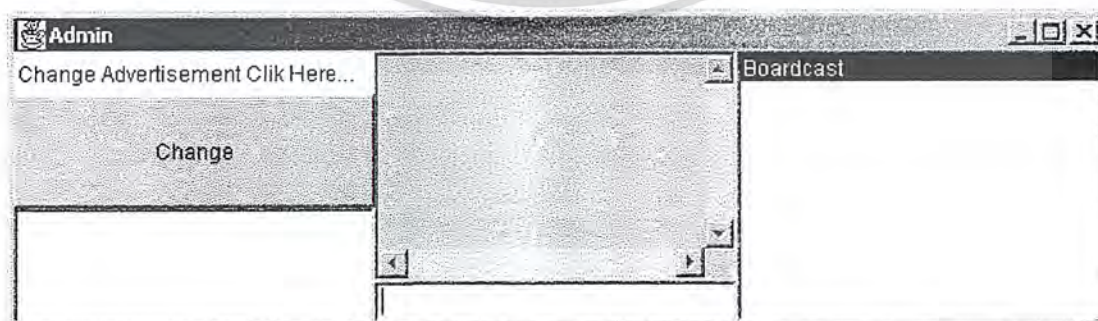
- แบบส่งให้ทุกคนที่ใช้บริการ โปรแกรมอยู่
- แบบส่งให้เป็นรายบุคคลที่ตัวละครอยู่ใกล้กัน

ในการส่งแบบแรกผู้เล่นทุกคนจะได้รับข้อความจากผู้ส่งเหมือนกัน แต่ในแบบที่สองจะได้รับเฉพาะคนที่ผู้ส่งเลือกส่งข้อความให้เท่านั้น โดยการส่งแบบนี้ผู้เล่นต้องบังคับตัวละครของตนเดินไปให้ใกล้กับตัวละครของคนที่เราต้องการจะพูดคุยด้วยเมื่อเดินเข้ามาใกล้ตัวละครของผู้เล่นคนอื่นจะปรากฏชื่อของผู้เล่นคนนั้นบนหน้าต่างการพูดคุย โดยเราสามารถเลือกส่งไปหาผู้เล่นคนอื่นได้ตามรายชื่อที่ปรากฏบนหน้าต่างหรือจะเลือกส่งหาผู้เล่นทุกคนโดยเลือก Broadcast

เมื่อผู้เล่นต้องการเลิกใช้บริการโปรแกรมสามารถทำได้โดยปิดหน้าต่างการพูดคุยของโปรแกรมก็จะออกจากโปรแกรม และผู้เล่นคนอื่นจะได้รับแจ้งการออกจากโปรแกรมของผู้เล่นบนหน้าต่างการพูดคุยของตน

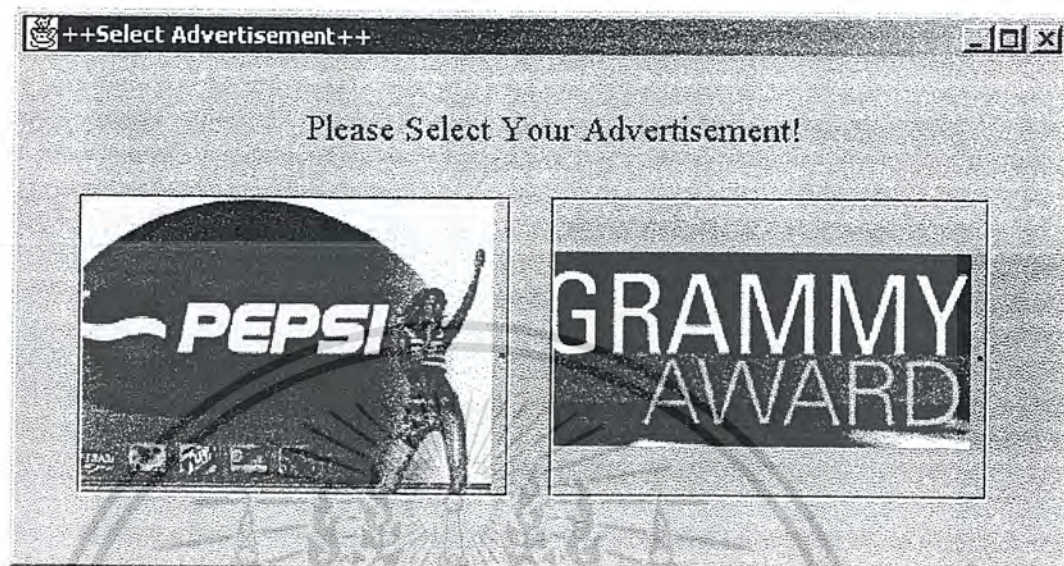
สำหรับผู้ควบคุมระบบ

ในส่วนของการใช้งานของผู้ควบคุมระบบจะมีจุดแตกต่างจากการใช้งานของผู้ใช้บริการทั่วไป คือ หน้าต่างการพูดคุยของผู้ควบคุมระบบจะมีส่วนเพิ่มเติมจากหน้าต่างของผู้เล่นทั่วไปเป็นปุ่มสำหรับแก้ไขปรับเปลี่ยนป้ายโฆษณาที่ปรากฏตามจุดต่างๆบนโลกจำลองดังรูปที่ 11.8



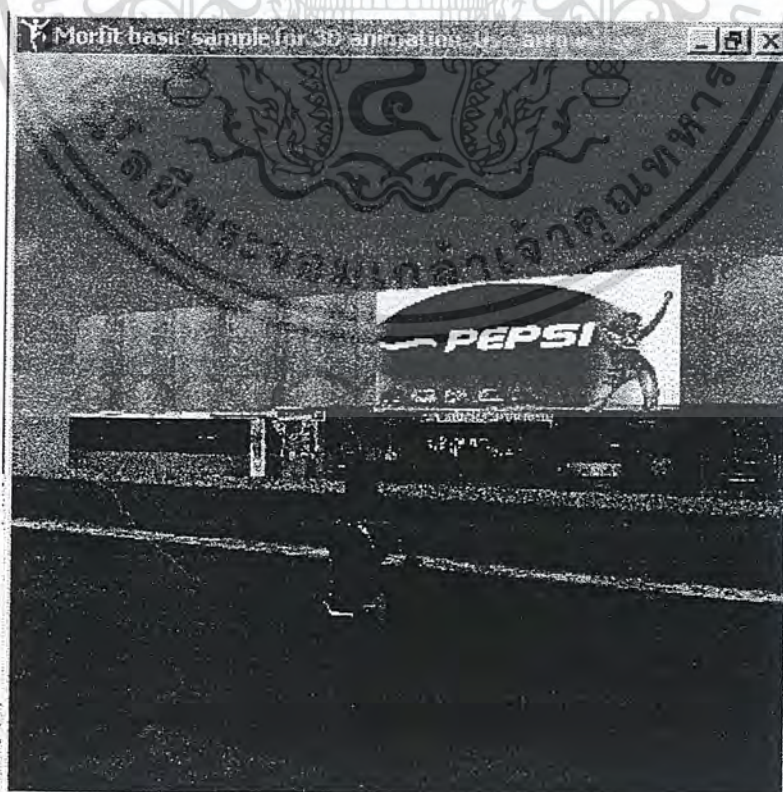
รูปที่ 11.8 หน้าต่างการพูดคุยสำหรับผู้ควบคุมระบบ

และเมื่อผู้ควบคุมระบบกดปุ่มเปลี่ยนป้ายโฆษณา หน้าต่างเลือกป้ายโฆษณาจะปรากฏบนหน้าจอให้ผู้ควบคุมระบบเลือกเปลี่ยนป้ายโฆษณาตามความต้องการดังรูปที่ 11.9



รูปที่ 11.9 หน้าต่างการเลือกป้ายโฆษณา

หลังจากกดเลือกป้ายโฆษณาที่ต้องการแล้วป้ายโฆษณาบนโลกจำลองก็จะเปลี่ยนตามที่คุณควบคุมระบบเลือกไว้ดังที่ปรากฏบนหน้าจอตามรูปที่ 11.10



รูปที่ 11.10 ผลเมื่อเลือกเปลี่ยนป้ายโฆษณา

บทที่ 12

บทวิจารณ์และสรุป

12.1 ประเมินผล

ในโครงการนี้ เราเน้นถึงการพัฒนาโปรแกรมโดยใช้ภาษาจาวาและภาษาซีร่วมกัน ในการควบคุมวัตถุ 3 มิติ ผ่านบนเน็ตเวิร์กพร้อมทั้งสามารถส่งข้อความสื่อสารกันได้อย่างมีประสิทธิภาพ อีกทั้งยังบรรลุถึงความสมจริงของฉากที่ได้จำลองมาจากสถานที่จริง ทำให้การสื่อสารกันระหว่างเครื่องคอมพิวเตอร์มีความสมจริงมากขึ้น

12.2 แนวทางการพัฒนาต่อ

ในปัจจุบันนี้มีผู้พัฒนาโปรแกรมการสื่อสารกันบนเน็ตเวิร์กแบบ 3 มิติมากขึ้น อีกทั้งเทคโนโลยีในการพัฒนาทางด้านนี้ได้พัฒนาไปอย่างรวดเร็ว เช่น เราสามารถประมวลผลภาพได้เร็วขึ้น ทำภาพให้สวยงามขึ้น เป็นต้น เนื่องด้วยเหตุผลเหล่านี้ แนวทางในการพัฒนาจึงน่าจะเป็นไปในทิศทางที่จะทำให้ภาพสมจริงขึ้น เราสามารถใส่เฟรมภาพในการทำงานมากขึ้น ทำให้ภาพออกมาสวยงามและนุ่มนวลขึ้น

นอกจากนั้นเราสามารถใส่ค่าการตอบสนองต่อการกระทำได้หลากหลายขึ้น เช่น สามารถแสดงอารมณ์ได้หลากหลาย สามารถส่งเสียงและแสดงท่าทางได้หลากหลาย เป็นต้น

อีกทางหนึ่งที่น่าสนใจ คือ การพัฒนาให้ผู้ใช้สามารถใช้งานโปรแกรมนี้ผ่านทางอินเทอร์เน็ตได้ จุดสำคัญของการพัฒนาในด้านนี้คือ ความเร็วในการรับส่งข้อมูลผ่านทางสายส่งสัญญาณ เราจะทำอย่างไรจึงสามารถให้ปริมาณการส่งข้อมูลลดลงและได้ภาพที่ดีที่สุด

12.3 สรุป

จากผลลัพธ์ของ โครงการที่ออกมานั้น ได้บรรลุถึงจุดประสงค์ของ โครงการที่ตั้งไว้เป็นอย่างดี โดยทางผู้พัฒนาได้ใช้ความรู้ที่ได้ศึกษามาอย่างเต็มที่จึงได้ผลงานออกมาอยู่ในระดับที่น่าพอใจ

บรรณานุกรม

- [1] อ.บัณฑิต จามรภูติ : “การประยุกต์ใช้ระบบ ไลคเอนส์ เซิร์ฟเวอร์”
- [2] ดร.วีระศักดิ์ ชิงฉาวร : “Java Programming Volume 1”
- [3] H.M. Deitel, P.J. Deitel : “Java How To Program”
- [4] Merlin Hughes, Michael Shoffner, Derek Hamner : “ Java Network Programming”
- [5] Morfit Staff : “3 มิติ Programming Using the Morfit 3 DEngine”
- [6] นิรุช อำนวยศิลป์ : “คู่มือการเขียน โปรแกรม Microsoft Visual C++ Version 6.0”
- [7] ศุภพงษ์ เลิศสินธวานนท์ : “ก้าวสู่โลก 3 มิติ 3 D Studio MAX”
- [8] <http://www.swissquake.ch/chumbalum.soft>, 2001 : “Milkshake 3 มิติ Tutorials”
- [9] <http://www.3 Dstate.com>
- [10] <http://www.java.sun.com>
- [11] <http://www.jcreator.com>

