การพัฒนาเกมสามมิติบนระบบเครือข่าย

3D GAME DEVELOPMENT (ONLINE)

นายเจนวัฒน์ แซ่ก๊อ

นายวิศิษฏ์ คุณากรพล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การพัฒนาเกมสามมิติบนระบบเครือข่าย

3D GAME DEVELOPMENT (ONLINE)

โดย

นายเจนวัฒน์ แซ่ก๊อ

นายวิศิษฏ์ คุณากรพล

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ ปีการศึกษา 2546

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเกมสามมิติบนระบบเครือข่าย

3D GAME DEVELOPMENT (ONLINE)

คณะผู้จัดทำ นายเจนวัฒน์ แซ่กือ รหัส 43010073

นายวิศิษฏ์ คุณากรพล รหัส 43010405

..... อาจารย์ที่ปรึกษา

(ดร.วรวัฒน์ ลิ้มโภคา)

การพัฒนาเกมสามมิติบนระบบเครือข่าย

นายเจนวัฒน์ แซ่กือ รหัส 43010073
นายวิศิษฏ์ คุณากรพล รหัส 43010405
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาโปรแกรมคอมพิวเตอร์ประเภทเกมบนระบบเครือข่ายโดยเป็นเกมออนไลน์แบบ MMORPG(Massive Multiplayer Online Role Playing Game) โดยมีสถาปัตยกรรมแบบ Client-Server ทำงานบนโปรโตคอล TCP/IP โดยจำลองให้ผู้เล่นเข้าไปแก้ปัญหาต่างๆตามเนื้อเรื่องที่จำลองว่าเกิดขึ้นบนระบบเครือข่ายแห่งหนึ่ง ฝั่งไคลเอนท์จะมีหน้าที่แสดงผลและรับคำสั่งจากผู้เล่นเท่านั้น การประมวลผลเกมจะทำบนเซิร์ฟเวอร์ทั้งหมดโดยอาศัยข้อมูลที่ส่งมาจากฝั่งไคลเอนท์ มีการใช้ระบบฐานข้อมูลทำให้ผู้เล่นไม่จำเป็นต้องบันทึกข้อมูล แม้ในกรณีที่ระบบเครือข่ายมีปัญหา ข้อมูลล่าสุดของผู้เล่นที่อยู่บนฐานข้อมูลก็จะไม่สูญหาย การทำงานของเซิร์ฟเวอร์จะอาศัยระบบสคริปต์ซึ่งช่วยให้ทำงานได้ง่ายขึ้น ระบบสคริปต์จะทำการอ่านไฟล์ที่ระบุค่ากำหนดต่างๆเอาไว้แล้วนำไปแปลงเป็นคำสั่ง ช่วยให้ผู้ควบคุมเกมไม่ต้องเสียเวลาคอมไพล์โปรแกรมใหม่อีกครั้ง และยังสามารถปรับปรุงข้อมูลให้ตรงกับความต้องการได้ตลอดเวลา

3D GAME DEVELOPMENT (ONLINE)

Mr.Chenwat Saekue
Mr.Visit Kunakornpoln
Dr. Voravat Limpoka Advisor
Academic Year 2003

ABSTRACT

This thesis is a 3D online game development or knows as MMOPRG (Massive Multiplayer Online Role Playing Game). Game architecture is Client-Server and works on TCP/IP Protocol. Story is about problem on Network. Client programs work in Graphic display and send game command from a player to game server. All game processing calculate on server by data come from every client. Players don't necessary to save game because database of the game work all of times receive data from client and save it time by time. When network's problem occurs, latest data from the player will live on database. Server uses script system to ease work of data changing. Script interpreter will read file and translate from natural language to command. Help game administrators save time cost of recompile every change of any data and update task

กิตติกรรมประกาศ

ขอขอบคุณภาควิชาคอมพิวเตอร์ที่ให้การสนับสนุนเป็นอย่างดี ขอบคุณสำหรับอุปกรณ์อำนวยความสะดวกต่างๆ พื้นที่ทำงานในห้องโปรเจคและอินเตอร์เนตความเร็วสูงที่ช่วยให้การทำโครงงาน และปริญญานิพนธ์นี้สำเร็จลงได้ด้วยดี

ขอบคุณ ดร.วรวัฒน์ ลิ้มโภคา และอาจารย์ทุกท่านสำหรับคำแนะนำและคำปรึกษาที่ช่วยให้ความรู้และข้อคิดเห็นสำหรับปรับปรุงโครงงานนี้ให้สมบูรณ์ยิ่งขึ้น

ขอบคุณเพื่อนๆ พี่ๆและน้องๆทุกคนที่ให้ความช่วยเหลือ และสาระคำแนะนำต่างๆมากบ้างน้อยบ้างจนงานชิ้นนี้บรรลุถึงจุดประสงค์ได้เป็นอย่างดี

ขอบคุณเนคเทคสำหรับงานแข่งที่ช่วยกระตุ้นการทำงานได้เป็นอย่างดี

และขอขอบคุณ คุณพ่อ คุณแม่ และทางครอบครัว สำหรับการสนับสนุนด้วยดี จนทำให้โครงงานนี้สำเร็จไปด้วยดี

นายเจนวัฒน์ แซ่กือ

นายวิศิษฏ์ คุณากรพล

สารบัญ

สารบัญตาราง

หน้าที่

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เกมเป็นแอพพลิเคชันที่ถูกสร้างขึ้นเพื่อความบันเทิง ซึ่งเครื่องคอมพิวเตอร์ส่วนบุคคลนั้นก็มีแอพพลิเคชันทางด้านเกมอยู่มากมาย ตั้งแต่สมัยที่เครื่องคอมพิวเตอร์ส่วนบุคคลยังใช้ระบบปฏิบัติการ MS-DOS จนมาถึงปัจจุบันบนระบบปฏิบัติการ Windows ซึ่งเกมนั้นถือได้ว่าเป็นแอพพลิเคชันที่ต้องการประสิทธิภาพการประมวลผลที่สูง โดยการพัฒนาของเครื่องคอมพิวเตอร์ส่วนบุคคลส่วนหนึ่งก็ได้รับแรงกระตุ้นจากแอพพลิเคชันทางด้านเกม ตามความเป็นจริงแล้วแอพพลิเคชันทางด้านเกมถือได้ว่าเป็นแอพพลิเคชันทางด้านมัลติมีเดียอย่างหนึ่งเพราะมีการประมวลผลทั้งทางด้านภาพ เสียง และส่วนที่ติดต่อกับผู้ใช้งาน

ในอดีตนั้นเกมถูกพัฒนาบนระบบปฏิบัติการ MS-DOS ซึ่งมีข้อดีคือผู้พัฒนานั้นสามารถที่จะเข้าถึงการทำงานของฮาร์ดแวร์ในระดับต่ำ (low-level) ได้โดยตรง ซึ่งทำให้แอพพลิเคชันมีประสิทธิภาพการทำงานที่สูง แต่มีข้อเสียคือ ผู้พัฒนาแอพพลิเคชันต้องพัฒนาให้แอพพลิเคชันของตนสามารถรองรับการทำงานกับอุปกรณ์ของบริษัทต่างๆ ที่มีอยู่มากมายในท้องตลาด เช่น การ์ดแสดงผลและการ์ดเสียงที่มีอยู่มากมายหลายยี่ห้อ เพื่อให้ครอบคลุมกลุ่มผู้ใช้งานให้ได้มากที่สุด ซึ่งเป็นงานที่ยากลำบากและสิ้นเปลืองงบประมาณเป็นอย่างมาก

ต่อมาบริษัทไมโครซอฟต์ได้เปิดตัวระบบปฏิบัติการ Windows ออกมา ซึ่งระบบปฏิบัติการ Windows นี้เป็นระบบปฏิบัติการแบบ GUI (Graphical User Interface) มีข้อดีคือผู้พัฒนาแอพพลิเคชันไม่จำเป็นต้องพัฒนาแอพพลิเคชันของตนให้สนับสนุนอุปกรณ์ของบริษัทต่างๆ อีกต่อไป เพราะเมื่ออุปกรณ์เหล่านั้นถูกพัฒนามาเพื่อใช้งานกับระบบปฏิบัติการ Windows แล้วผู้พัฒนาแอพพลิเคชันเพียงแต่พัฒนาโดยยึดรูปแบบมาตรฐานของระบบปฏิบัติการ Windows ก็เพียงพอ ซึ่งเป็นสมบัติแบบ “device-independent” แต่ระบบปฏิบัติการ Windows นั้นเป็นระบบปฏิบัติการที่มีการแสดงผลด้านกราฟิกที่ช้า ดังนั้นในยุคแรกๆ ของระบบปฏิบัติการ Windows ผู้พัฒนาแอพพลิเคชันทางด้านเกมจึงยังคงยึดติดอยู่กับระบบปฏิบัติการ MS-DOS อยู่ ซึ่งแอพพลิเคชันทางด้านเกมที่ทำงานอยู่บนระบบปฏิบัติการ Windows นั้นก็พอมีอยู่บ้าง แต่จะเป็นพวกที่ไม่ต้องการการแสดงผลทางด้านกราฟฟิกที่รวดเร็ว เช่น เกมหมากกระดาน และเกมแนวผจญภัย

ทางบริษัทไมโครซอฟต์ได้เล็งเห็นว่าแอพพลิเคชันทางด้านเกมนั้นมีผู้ใช้งานกันอย่างแพร่หลาย จึงมีแนวคิดที่จะให้ผู้พัฒนาแอพพลิเคชันทางด้านเกมและมัลติมีเดียหันมาพัฒนาบนระบบปฏิบัติการ Windows ดังนั้นบริษัทไมโครซอฟต์จึงพัฒนาไดเร็คเอ็ก(DirectX)ขึ้นมา ซึ่งเป็นไลบรารีทางด้านมัลติมีเดียที่มีจุดมุ่งหมายหลัก ๆ ดังต่อไปนี้

1. ไดเร็คเอ็กประกอบด้วยไลบรารีที่ทำงานในระดับต่ำ ซึ่งทำงานได้รวดเร็วและไม่มีข้อจำกัดในการพัฒนาแอพพลิเคชันทางด้านเกม
2. โครงสร้างของไดเร็คเอ็กต้องยกภาระในเรื่องฮาร์ดแวร์จากผู้พัฒนาแอพพลิเคชันไปสู่ผู้ผลิตฮาร์ดแวร์ ซึ่งผู้ผลิตฮาร์ดแวร์ต้องเป็นผู้สร้างไดรเวอร์สำหรับผลิตภัณฑ์ของตน และให้ผู้พัฒนาแอพพลิเคชันนั้นสามารถใช้ความสามารถล่าสุดที่มีอยู่ในฮาร์ดแวร์นั้นๆ ได้
3. ไดเร็คเอ็กนั้นต้องอนุญาตให้ผู้พัฒนาแอพพลิเคชันสามารถพัฒนาแอพพลิเคชันออกมาในรูปแบบของ Windows application ที่สามารถทำงานบนเดสท็อปได้และสามารถทำงานร่วมกับฟังก์ชันต่างๆ ที่มีอยู่บนระบบปฏิบัติการ Windows ได้
4. แอพพลิเคชันที่ถูกพัฒนาโดยใช้ไดเร็คเอ็กนั้นต้องมีประสิทธิภาพที่อยู่สูงกว่าหรืออย่างน้อยที่สุดต้องเทียบเท่ากับประสิทธิภาพของแอพพลิเคชันที่ทำงานบนระบบปฏิบัติการ MS-DOS

ซึ่งในปัจจุบันแอพพลิเคชันทางด้านเกมและมัลติมีเดียได้หันมาพัฒนาบนระบบปฏิบัติการ Windows กันหมดแล้ว เพราะไดเร็คเอ็กนั้นทำให้แอพพลิเคชันมีประสิทธิภาพมาก เช่น สามารถใช้ความสามารถของการ์ดเร่งความเร็วกราฟิกแบบ 3 มิติที่ติดตั้งอยู่บนเครื่องคอมพิวเตอร์และชุดคำสั่งพิเศษของไมโครโปรเซสเซอร์ได้ อีกทั้งยังช่วยให้ผู้พัฒนาแอพพลิเคชันไม่ต้องมายุ่งเกี่ยวในส่วนของฮาร์ดแวร์ที่มีอยู่มากมายหลายยี่ห้ออีกต่อไป โดยที่ยังสามารถเข้าถึงการทำงานในระดับต่ำของฮาร์ดแวร์ได้

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานโปรแกรมและการพัฒนา โปรแกรมเกมประเภท 3 มิติ
2. เพื่อพัฒนาชุดคำสั่งของไดเร็คเอ็กซ์และนำไปสร้างเกมเอนจินเพื่อให้สามารถพัฒนาเกมได้สะดวกยิ่งขึ้น
3. เพื่อศึกษาและทดลองการสร้างเกมออนไลน์แบบ MMORPG (Massively Multiplayer Online Role-playing Game) โดยศึกษาการทำงานและติดต่อส่งข้อมูลกันระหว่างผู้เล่น(CLIENT) และผู้ให้บริการในการเล่น (SERVER)
4. เพื่อศึกษาและทดลองสร้างฐานข้อมูลที่ใช้กับผู้ให้บริการในการเล่น (SERVER)

1.3 ขอบเขตของโครงงาน

เกม 3 มิติที่จะพัฒนานี้ได้มีข้อจำกัดในได้ต่างๆบางอย่างคือสามารถใช้งานได้เพียงระบบ ปฏิบัติการวินโดว์เท่านั้น เพราะว่าซอฟแวร์ที่นำมาใช้พัฒนาทั้งไดเรกซเอกซ์และ MS VC++ ต่างก็เป็นโปรดักซ์ของไมโครซอฟท์ ดว้ยกันทั้งนั้นและจำเป็นต้องใช้เครื่องที่มีประสิทธิสูง โดยต้องมีการ์ดแสดงผล 3 มิติได้ดว้ย เนื่องจากได้พัฒนาเกมเป็น 3 มิติทั้งหมด ของเขตของโครงงานมีดังนี้

ในส่วนของไคลเอนท์

1. โปรแกรมเกมที่พัฒนาจะมีรูปแบบเป็นเกม MMORPG (Massively-Multiplayer Online Role-Playing Game) โดยจะเป็นเกมออนไลน์ที่มีรูปแบบเป็น 3 มิติและเป็นแบบมุมมองบุคคลที่ 3 (THIRD-PERSON VIEW) คือ มองเห็นตัวละครที่เล่นอยู่

2. การเชื่อมต่อจะเป็นแบบเรียลไทม์(REAL-TIME) คือ ต้องเชื่อมต่อกับเครื่องเซิฟเวอร์เพื่อที่จะเล่นเกมกับคนอื่นได้
3. โปรแกรมเกมสามารถที่จะสนทนากับผู้เล่นคนอื่นในขณะเล่นเกมได้ โดยติดต่อผ่านเครื่องเซิฟเวอร์
4. โปรแกรมเกมที่จะพัฒนามีการกำหนดความสามารถตัวละคร สามารถที่จะให้มีเก็บประสบการณ์จากการต่อสู้ได้ มีการใช้ไอเท็มเพื่อเพิ่มความสามารถของตัวละคร
5. โปรแกรมต้องมีการใส่ID เพื่อเข้าเล่นเกมโดยแต่ละคนต้องมี ID เป็นของตัวเอง

ในส่วนของเซิฟเวอร์

1. โปรแกรมสามารถรับข้อมูลและประมวลผลจากผู้เล่นหลายๆคนได้
2. โปรแกรมมีความสามารถในการทำงานในส่วนของค้นหาข้อมูลในฐานข้อมูลได้
3. โปรแกรมจะต้องตรวจสอบการเข้ามาเล่นเกมของเครื่องไคลเอนต์ว่ามีการ LOG IN หรือไม่ และตรวจสอบ ID และ PASSWORD ของผู้เล่นคนนั้นๆ
4. โปรแกรมสามารถรับการส่งไปหาไคลเอนท์แบบBroadcastได้

1.4 วิธีในการดำเนินงาน

1. ทำการศึกษาและค้นคว้าความรู้ทางด้านการเขียนเกม 3 มิติและไดเร็คเอ็กจากหนังสือและอินเตอร์เน็ต
2. ออกแบบโครงสร้างและแบ่งงานตามกลุ่มตามงานที่ได้รับมอบหมาย
3. สร้างเกมเอนจินในแต่ละส่วนตามโครงสร้างที่ออกแบบไว้
4. ทำการทดสอบเกมเอนจินที่ได้สร้างขึ้น
5. ทำการรวบรวมส่วนต่างๆ ของเกมเอนจินจากแต่ละกลุ่มให้กลายเป็นเกมเอนจินที่สามารถนำไปใช้งานได้
6. วางโครงสร้างเกม โดยนำเอาเกมเอนจินที่ได้พัฒนาไปใช้ในการพัฒนาเกมต่อ ซึ่งจะนำไปสร้างเกมแบบ MMORPG
7. สร้างโมเดล 3 มิติและสร้างภาพกราฟิกที่ต้องใช้งานในเกม
8. ศึกษาวิธีการทำงานของเกมแบบ MMORPG และศึกษาอัลกอลิทึมที่ใช้ในเกมเพื่อเพิ่มประสิทธิภาพในการทำงาน
9. สร้างเกมตามโครงสร้างที่ได้ออกแบบไว้
10. ทดสอบการทำงานของเกม และหาข้อผิดพลาดของเกมที่พัฒนาขึ้น
11. ทำการรวบรวมผลทดสอบ วิเคราะห์ปัญหาที่เกิดขึ้น และหาแนวทางในการแก้ปัญหาเพื่อนำมาสรุปเป็นข้อมูลในการพัฒนาขั้นต่อไป

บทที่ 2
ไดเร็คเอ็ก(DirectX)

2.1 ไดเร็คเอ็กคืออะไร

ไดเร็คเอ็ก คือ กลุ่มเทคโนโลยีที่ออกแบบโดยไมโครซอฟต์เพื่อให้แอพพลิเคชันบนวินโดวส์สามารถใช้งานอุปกรณ์มัลติมีเดีย เช่นภาพกราฟิก วิดีโอ ภาพเคลื่อนไหว หรือแม้แต่ระบบเสียงรอบทิศทางได้เต็มประสิทธิภาพ สำหรับวินโดวส์ 98 และวินโดว์ 2000 รวมไปถึง Internet Explorer จะมีไดเร็คเอ็กเป็นส่วนประกอบมาเรียบร้อยแล้ว แต่อย่างไรก็ดี ยังสามารถติดตั้งไดเร็คเอ็กโดยอัตโนมัติ ด้วยการติดตั้งเกมส์ หรือแอพพลิเคชันมัลติมีเดียส่วนใหญ่ที่ออกแบบมาให้ทำงานกับไดเร็คเอ็ก

ไดเร็คเอ็กช่วยให้ผู้พัฒนาโปรแกรมมีโอกาสใช้อุปกรณ์ทันสมัยใหม่ๆได้โดยอัตโนมัติ ไม่ต้องกังวลว่าโปรแกรมหรือเกมส์ที่เขียนขึ้นมาจะได้กับอุปกรณ์มัลติมีเดียตัวใหม่ๆ ได้หรือไม่ ตราบใดที่ยังใช้ไดเร็คเอ็กอยู่ หลังจากที่ติดตั้งไดเร็คเอ็กสิ่งที่เกิดขึ้นคือ จะมีไฟล์จำนวนหนึ่งเพิ่มขึ้นใน Windows/System จำนวนหนึ่ง ซึ่งขณะที่เล่นเกม ไฟล์เหล่านี้ก็จะถูกโหลดขึ้นมา Link กับโปรแกรมเกม จากนั้นเกมก็จะสามารถเรียกใช้ความสามารถของไดเร็คเอ็กได้ ไฟล์ .dll เหล่านี้จะคอยติดต่อกับไดร์เวอร์ของการ์ดจอ ซาวน์การ์ด โมเดม และส่วนประกอบอื่นๆ ของเครื่อง อีกทีนึง สรุปได้ว่า โปรแกรมจะเรียกใช้ฮาร์ดแวร์ได้ผ่านไดเร็คเอ็กซึ่งจะเรียกผ่านไดร์เวอร์ที่ติดตั้งไว้อีกทีนึง ตรงนี้จะเห็นได้ว่าผู้เขียนโปรแกรมไม่จำเป็นต้องไปเรียกใช้ไดร์เวอร์ หรือ(ที่แย่กว่านั้น) เรียกใช้ฮาร์ดแวร์โดยตรง เพียงแต่เรียกผ่านคำสั่งต่างๆ ใน ไดเร็คเอ็กเท่านั้นเอง แล้ว ไดเร็คเอ็กก็จะติดต่อกับฮาร์ดแวร์ให้อีกทีนึง ตรงนี้เป็นจุดสำคัญที่ทำให้ไดเร็คเอ็กประสบความสำเร็จได้ เพราะหมดปัญหาเรื่องความเข้ากันได้ของฮาร์ดแวร์และยังทำให้ผู้เขียนโปรแกรมไม่ต้องไปยุ่งกับพวกคำสั่งในระดับล่างต่างๆ ด้วย

ไดเร็คเอ็กแบ่งใหญ่ๆ ได้เป็นส่วนย่อยๆดังนี้

1. **DirectGraphics** ทำหน้าที่ดูแลและจัดการการแสดงผลทั้งภาพ 2 มิติและ 3 มิติ ช่วยในการให้แสง จัดวางมุมกล้อง และทำภาพเคลื่อนไหว ซึ่งจะทำการควบคุมอุปกรณ์แสดงผล และดึงความสามารถของฮาร์ดแวร์ให้เต็มประสิทธิภาพ ในนี้จะประกอบด้วยส่วนประกอบย่อยๆ เช่น Direct3D และ DirectDraw
2. **DirectSound** ทำหน้าที่ดูแลและจัดการเกี่ยวกับเสียง ไม่ว่าจะเป็นเสียงแบบ 3 มิติหรือเสียงแบบเซอร์ราวด์ นอกจากนั้นยังช่วยในการทำเสียงเอฟเฟ็คท์ต่างๆ อีกด้วย
3. **DirectInput** ทำหน้าที่ดูแลและจัดการเกี่ยวกับอุปกรณ์อินพุตที่ใช้ควบคุมต่างๆ เช่น คีย์บอร์ด เมาส์ หรือจอยสติ๊ก เป็นต้น และยังสามารถทำงานกับอุปกรณ์ที่เป็นแบบป้อนกลับ (Force feedback) ได้อีกด้วย
4. **DirectPlay** ทำหน้าที่ดูแลและจัดการเกี่ยวกับการติดต่อระหว่างคอมพิวเตอร์ผ่านระบบเครือข่าย ไม่ว่าจะเป็นระบบแลนหรือผ่านโมเดม

5. **DirectShow**ทำหน้าที่ดูแลและจัดการเกี่ยวกับการแสดงผลภาพวีดีโอหรือนอกจากนี้ยังสามารถจัดแบ่ง API ของไดเร็คเอ็กอีกแบบเป็นสองส่วนใหญ่ๆ คือ
6. **DirectX Foundation** ประกอบไปด้วยส่วน DirectDraw, DirectSound, Direct3D Immediate Mode และ DirectInput โดยจะดูแล function Low-Level ต่างๆ
7. **DirectX Media** ประกอบไปด้วยส่วน DirectShow, DirectAnimation, DirectPlay และ Direct3D Retained Mode ซึ่งเป็นบริการระดับสูง ที่จะไปเรียกไดเร็คเอ็กFoundation อีกทีหนึ่ง

สำหรับเกมทั่วไปจะเรียกใช้ไดเร็คเอ็กFoundation เป็นหลัก และเรียกใช้ไดเร็คเอ็กMedia เมื่อต้องการได้รับบริการบางอย่าง DirectAnimation เป็น API(Application Programming Interface) สำหรับใช้ในหน้าเวปซึ่ง Internet Explorer ของ microsoft สนับสนุน สำหรับ DirectShow นั้นเป็น API สำหรับเรียกไฟล์ภาพเคลื่อนไหว (เช่น .avi) ได้อย่างสะดวก

2.1.1 Immediate Mode และ Retained Mode

อีกจุดหนึ่งที่อาจจะสงสัยคือ Direct3D Immediate Mode(IM) และ Retained Mode(RM) ต่างกันอย่าไร ถ้าดูจากว่า IM อยู่ใน Foundation และ RM อยู่ใน Media ก็พอจะบอกได้ว่า RM จะประกอบด้วยฟังค์ชั่นการทำงานระดับสูงกว่า IM ก็คือ Direct3D IM จะประกอบด้วย Low-Level Drawing Functions มีความสามารถในการวาด Low-Level 3D Objects เท่านั้น ส่วน RM จะวาด Object ที่ High-Level กว่า อย่างเช่น IM จะวาดได้แค่ 3 เหลี่ยม ในขณะที่ RM จะมีความสามารถในการวาด Polygon ได้เลย แต่เกมทั่วๆ ไปจะใช้ Immediate Mode เนื่องจาก Retained Mode Microsoft นั้นทำไว้ไม่ค่อยดีนัก ซึ่ง Immediate Mode ของ Direct3D ก็อยู่ในระดับเดียวกับ OpenGL คือเป็นการทำงานระดับล่างเหมือนกัน

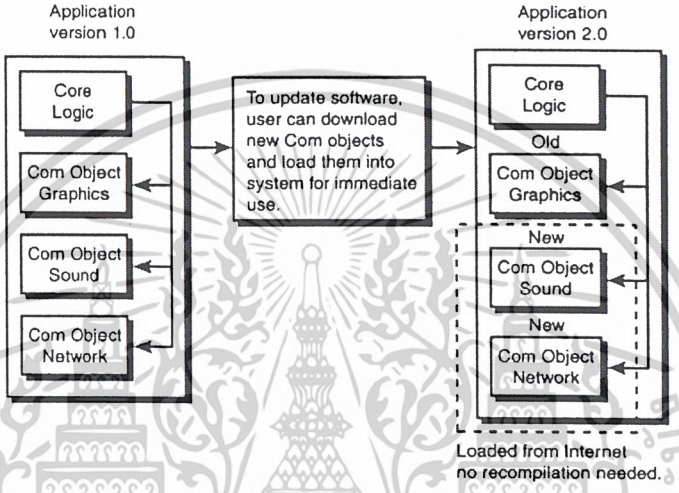
ไฟล์ที่จำเป็นก็คือไฟล์ .lib และ .h ต่างๆ การที่จะเขียนโปรแกรมเรียกใช้ไดเร็คเอ็กได้นั้น ก็จะต้องนำ .lib มา link กับโปรแกรมของตอน Compile แล้วก็ include .h ไว้ใน Source File ของด้วย เช่นถ้าจะใช้ DirectDraw ก็ต้องเพิ่ม ddraw.lib ลงใน Project ของ และ #include ไว้

2.2 บทบาทของ COM

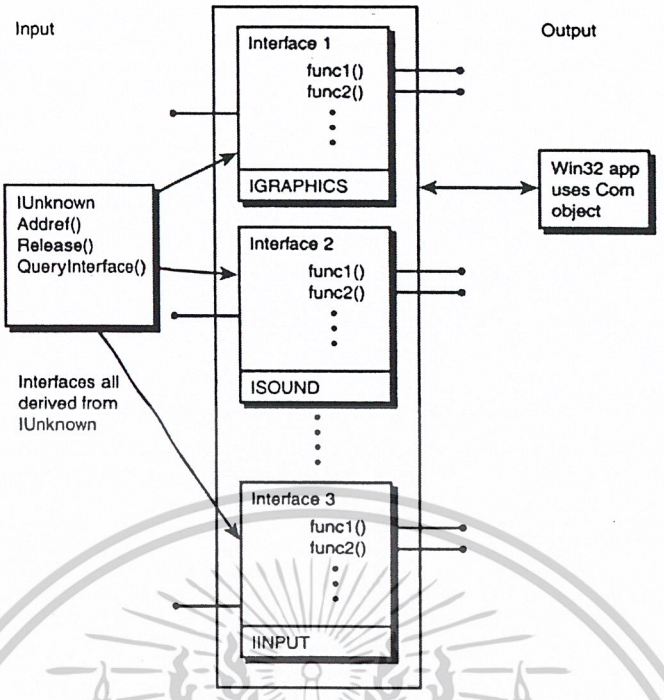
COM(Component Object Model) ได้กล่าวถึงรายละเอียดเกี่ยวกับวิธีการติดต่อสื่อสารกันระหว่างอ็อบเจ็กต์ โดยข้ามกันระหว่างโพรเซสและข้ามผ่านเน็ตเวิร์ค ระบบที่มีการเรียกใช้ COM ใช้งานจะถูกรับรองเป็นอ็อบเจ็กต์อินสแตนซ์ (Object Instance)

ทุกๆ COM อ็อบเจ็กต์นั้นยังจะต้องถูกสืบทอดมาจาก COM อินเตอร์เฟซมาตรฐาน ซึ่งเรียกว่า IUnknown โดย parent อินเตอร์เฟซจะมีอยู่ด้วยกัน 3 เมธอดคือ AddRef, Delete และ QueryInterface ซึ่ง 2 เมธอด แรกจะจัดการอ็อบเจ็กต์ด้วยการอ้างอิงถึงจำนวน โดยการนับจำนวนของอินสแตนซ์ตั้งแต่เริ่มต้น ส่วนเมธอดสุดท้ายจะจัดการเกี่ยวกับการสอบถามกับ COM อ็อบเจ็กต์ตัวอื่นๆ สำหรับอินเตอร์เฟซที่ต้องการและอินเตอร์เฟซที่มีอยู่ จากนั้นจะคืนค่าพอยเตอร์ที่ชี้ไปที่อินเตอร์เฟซนั้น

COM อ็อบเจ็กต์ที่ทำการร้องขอนั้น เรียกว่า COM ไคลเอนต์และอ็อบเจ็กต์ที่ตอบสนองการร้องขอนั้นคือ COM เซิร์ฟเวอร์ เมื่อไคลเอนต์ได้รับการเชื่อมต่อแล้วไคลเอนต์นั้นสามารถที่จะเรียกเมธอดใดก็ได้ใน COM เซิร์ฟเวอร์นั้น โดยใช้การอ้างอิงของอินเตอร์เฟซพอยเตอร์ ส่วนการเชื่อมต่ออื่นๆ จะออกไปเพื่อทำให้งานของอ็อบเจ็กต์ที่เรียกเพื่อเข้ามาใช้งานง่ายขึ้น ในการเชื่อมต่อนี้จะมีเมธอดที่ทำการแสดงรายชื่อของเมธอดที่สนับสนุน เพื่อที่จะช่วยให้เมธอดสามารถเรียกใช้งานได้ ก่อนที่จะรวมเข้าไว้ด้วยกัน ด้วยเทคนิคนี้จะอนุญาตให้สามารถสร้างชนิดของไลบรารีใดๆ และแอพพลิเคชัน เช่น Visual Basic ซึ่งมีข้อดีในการสร้างสมบัติของ COM โปรแกรมที่มีความทนทานต่อความผิดพลาด



รูปที่ 2-1 ภาพโดยรวมของ COM



รูปที่ 2-2 อินเตอร์เฟซของ COM อ็อบเจ็กต์

2.3 COM และไดเร็คเอ็กซ์

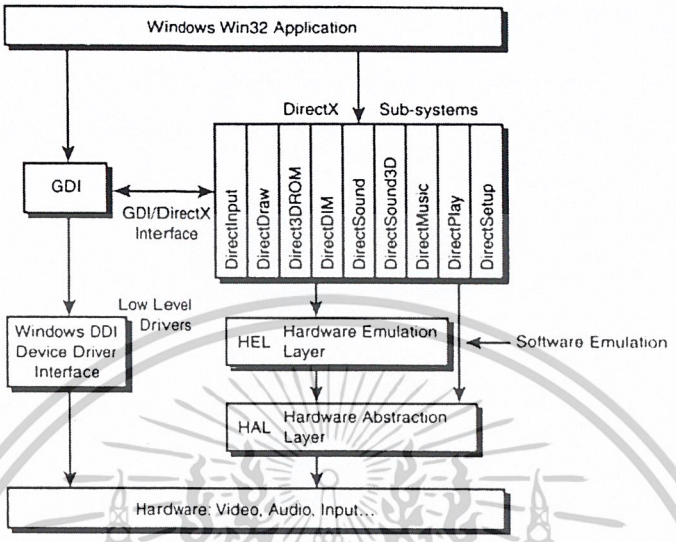
ทุกๆ อินเตอร์เฟซของไดเร็คเอ็กซ์นั้นจะทำการสืบทอดมาจาก IUnknown และมีพื้นฐานในการพัฒนาจาก COM ด้วยเช่นกัน การใช้ COM ของไมโครซอฟท์ทำให้ได้ชุดคอมโพเน็นต์ของไดเร็คเอ็กซ์ที่มีประสิทธิภาพเพิ่มขึ้น ซึ่งสามารถเข้ากันได้กับไดเร็คเอ็กเวอร์ชั่นก่อนๆ ได้อย่างสมบูรณ์ ด้วยความอิสระไม่ขึ้นกับภาษาที่ใช้เขียนโปรแกรมของ COM ทำให้นักพัฒนาสามารถที่จะตัดสินใจเลือกภาษาที่ใช้เขียนโปรแกรมตามที่เหมาะสมกับนักพัฒนาเองมากที่สุด ซึ่งอาจเป็นภาษาที่ถนัด และยังทำให้ได้ประสิทธิภาพตามที่ต้องการ นอกจากนั้นนักพัฒนายังสามารถที่จะเลือกภาษาอะไรก็ตามที่ COM สนับสนุน และสามารถพัฒนาสภาพแวดล้อมได้ตามที่นักพัฒนาต้องการด้วย โปรแกรมมิ่งโมเดลของCOM โปรแกรมเมอร์จะแน่ใจได้ว่าแม้ว่าผู้ใช้จะอัพเกรดหรือเปลี่ยนติดตั้งฮาร์ดแวร์ใหม่ หรือจะเปลี่ยนไดเร็คเอ็กเวอร์ชั่นใหม่ก็ตาม ซอฟต์แวร์ก็ยังคงจะสามารถใช้งานได้ตามปกติ โดยไม่จำเป็นต้องทำการติดตั้งใหม่

ในความเป็นจริงไดรเวอร์ของอุปกรณ์บน Windows ทุกตัวจะรวมกันเป็น WDM (Windows Driver Model) ซึ่งโครงสร้างภายในก็เป็น COM คอมโพเน็นต์ ดังนั้นจึงมีการโต้ตอบที่ดีกว่าระหว่างไดรเวอร์โมเดลกับหน้าที่ของไดรเวอร์ ในสถาปัตยกรรมของไดเร็คเอ็กนั้นทำให้แน่ใจได้ว่าการพัฒนาสำหรับนักพัฒนาเกมส์จะมีความสะดวกมากขึ้น และลดเวลาในการทดสอบลงด้วย

2.4 สถาปัตยกรรมของไดเร็คเอ็ก

ไดเร็คเอ็กได้สร้างขึ้นมาบนพื้นฐานของคอมโพเน็นต์ โดยใช้โครงสร้างของ HAL (Hardware Abstraction Layer) ซึ่งสามารถซ่อนลักษณะเฉพาะของดีไวซ์ (Device) ที่เกี่ยวพันอยู่กับฮาร์ดแวร์ และ

เนื่องจากไดเร็คเอ็กได้รับการออกแบบให้มีความสามารถในการพัฒนาได้เพิ่มมากขึ้นในอนาคต ดังนั้นจึงสามารถรับรองความสามารถของฮาร์ดแวร์เร่งความเร็วชนิดใหม่ที่เข้ามา โดยใช้การทำงานผ่าน HEL (Hardware Emulation Layer)



รูปที่ 2-3 สถาปัตยกรรมของไดเร็คเอ็ก

2.4.1 HAL (Hardware Abstraction Layer)

HAL เป็นส่วนที่อยู่ล่างสุดของไดเร็คเอ็กซึ่งประกอบไปด้วยส่วนควบคุมของฮาร์ดแวร์ที่ถูกทำโดยผู้ผลิตฮาร์ดแวร์ ที่จะควบคุมฮาร์ดแวร์โดยตรง โดยชั้นนี้จะให้ประสิทธิภาพอย่างมาก เพราะว่าสามารถที่จะติดต่อกับฮาร์ดแวร์ได้โดยตรง ในการทำงานจริงจะไม่สามารถติดต่อกับ HAL ได้เอง แต่ไดเร็คเอ็กจะทำการจัดการให้โดยอัตโนมัติ

2.4.2 HEL (Hardware Emulation Layer)

HEL จะอยู่บนชั้น HAL โดยทั่วไปไดเร็คเอ็กจะถูกออกแบบให้สามารถใช้ข้อดีของฮาร์ดแวร์ต่างๆ ได้ แต่ถึงอย่างไรไดเร็คเอ็กก็ไม่สามารถที่จะทำงานได้กับอุปกรณ์ทุกชนิด ตัวอย่างเช่น การเขียนโค้ดแสดงผลกราฟิก สมมติว่าฮาร์ดแวร์ที่กำลังทำงานอยู่สนับสนุนการหมุน (Rotate) และปรับขนาด (Scale) ภาพบิตแมพ ซึ่งการเรียกใช้งานไดเร็คเอ็กเพื่อที่จะปรับขนาดและหมุนภาพบิตแมพได้นั้น จะต้องมีฮาร์ดแวร์ที่สนับสนุนการปรับขนาดและหมุนภาพ ซึ่งถ้าฮาร์ดแวร์สนับสนุนการทำงาน ก็จะทำงานได้เต็มความสามารถ และใช้งานฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ แต่ถ้าฮาร์ดแวร์ไม่สนับสนุนการปรับขนาดและหมุนภาพ การทำงานของ HEL จะเข้ามาทำงานแทนที่ โดย HEL จะทำการจำลองหน้าที่การทำงานของ HAL ด้วยอัลกอริทึมทางซอฟต์แวร์ โดยจะไม่ทราบถึงความแตกต่างเลย แต่ถึงอย่างไรก็ตามโค้ดที่ได้ทำงานนั้น ก็จะทำงานได้ช้าลง เพราะว่าเป็นการจำลองการทำงาน จึงไม่สามารถที่จะทำงานได้เหมือนอย่างฮาร์ดแวร์ร้อยเปอร์เซ็นต์

ไดเร็คเอ็กไดรเวอร์นี้จะรวมเข้าด้วยกันกับไดเร็คเอ็ก API ผ่านลำดับของบัพเฟอร์อ็อบเจ็กต์ ผลลัพธ์ส่วนมากจะถูกสร้างโดยคอมโพเน็นต์ ไดเร็คเอ็กAPI จะถูกเขียนขึ้นเพื่อตอบสนองบัพเฟอร์อ็อบเจ็กต์ โดยบัพเฟอร์นี้จะถูกจัดการ และถูกทำให้มีประสิทธิภาพขึ้นตามฮาร์ดแวร์ที่มีอยู่ โดย 2 เลเยอร์ของสถาปัตยกรรมของไดเร็คเอ็กและแปลงไปยังเอาท์พุตดีไวซ์ที่เหมาะสมอย่างเป็นระเบียบ ในระดับของภาษาที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้

สมบัติของการจำลองการทำงานของฮาร์ดแวร์ ก็คือชุดของฟังก์ชันที่ทำการจำลองคำสั่งทาง 3 มิติ โดยใช้ซอฟต์แวร์ ถ้าฟังก์ชันที่ต้องการโดยเกมนั้น ไม่สามารถที่จะเข้าถึงฮาร์ดแวร์ได้โดยผ่าน HAL ไดเร็คเอ็กนั้นจะสลับการทำงานมายัง HEL เพื่อที่จะจำลองการทำงานโดยใช้ซอฟต์แวร์แทน ด้วยสมบัตินี้จะช่วยให้นักพัฒนาสามารถที่จะทดสอบหรือตรวจสอบสมบัติทาง 3 มิติในโหมดของซอฟแวร์ได้ด้วยตนเอง

สิ่งเหล่านี้จะทำให้ไดเร็คเอ็กมีความยืดหยุ่นในการใช้งานอย่างมาก ซึ่งจะช่วยสนับสนุนการติดตั้งใช้งานฮาร์ดแวร์ใดๆ ไม่ว่าจะเป็นรุ่นเก่าหรือใหม่ ได้อย่างหลากหลาย เช่น ถ้ามีแอพพลิเคชันที่ต้องการสมบัติใหม่ๆ ที่อยู่ในฮาร์ดแวร์รุ่นใหม่ ก็ยังจะสามารถที่จะทำงานในฮาร์ดแวร์รุ่นเก่าได้ โดยผ่านชั้น HEL ของไดเร็คเอ็กเพื่อที่จะจำลองความสามารถใหม่ที่ต้องการ ดังนั้น ถ้ายิ่งติดตั้งไดเร็คเอ็กรุ่นใหม่เท่าใด จะช่วยในการทำงานของฮาร์ดแวร์ในคอมพิวเตอร์ ดังนั้นเกมที่ถูกเขียนมาในรูปแบบหนึ่ง จะได้รับข้อดีของสมบัติใหม่ของ ฮาร์ดแวร์ที่ติดตั้งอยู่ในเครื่อง โดยการรวมเข้าด้วยกันกับไดเร็คเอ็ก

ตัวอย่าง เช่น ถ้ามีไดเร็คเอ็กที่เวอร์ชั่นใหม่กว่า ที่ออกแบบมาให้สามารถทำงานกับคอมโพเน็นต์ได้อย่างมีประสิทธิภาพโดยผ่าน MMX ดังนั้นซอฟต์แวร์ทั้งหมดที่ถูกเขียนขึ้นมาก่อนที่ MMX จะออกจะสามารถใช้งาน MMX ได้ แม้ว่าแนวความคิดของ MMX ในสมัยนั้น จะยังไม่เกิดขึ้นในตอนที่ซอฟต์แวร์นั้นถูกเขียนขึ้น ในทางกลับกันซอฟต์แวร์ที่ถูกเขียนขึ้น ในตอนหลังที่มี MMX แล้ว ก็ยังสามารถทำงานได้ในโปรเซสเซอร์รุ่นเก่าก่อนที่มี MMX ได้ โดย HEL ไดรเวอร์นั้นจะสามารถจำลองหน้าที่การทำงานของ MMX ให้มากที่สุดเท่าที่จะเป็นไปได้ โดยผ่านซอฟต์แวร์โดยอนุญาตให้ผู้ใช้ได้ใช้ความสามารถนั้นด้วย

ดังนั้นเหตุผลหลักของข้อดีเหล่านี้ เป็นเพราะว่าการออกแบบสถาปัตยกรรมให้เป็น COM ซึ่งสามารถช่วยให้คอมโพเน็นต์อ็อบเจ็กต์สามารถติดต่อสื่อสารกันได้

บทที่ 3
ทฤษฎีและหลักการเกม 3 มิติ

ในการพัฒนาโปรแกรมเกม 3 มิติ จำเป็นอย่างยิ่งที่จะต้องมีความเข้าใจถึงทฤษฎีพื้นฐานที่จำเป็นสำหรับนำไปประยุกต์ใช้ โดยเฉพาะทฤษฎีทางด้านคอมพิวเตอร์กราฟิก ซึ่งเป็นพื้นฐานสำหรับการพัฒนาโปรแกรมเกม 3 มิติ

3.1 ระบบพิกัด 3 มิติ

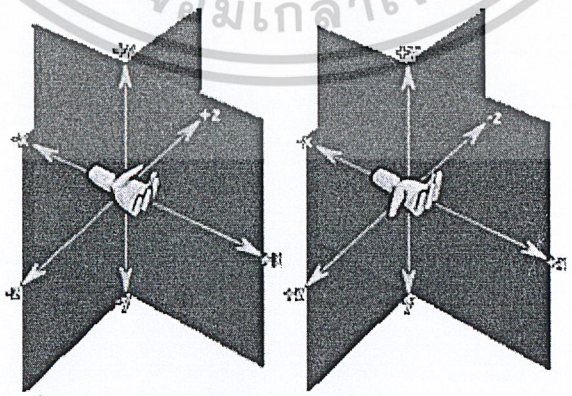
หมายถึงทิศทางของค่าในแนวแกน x, y และ z ของจุดกำเนิด (Origin) ในระบบสามมิติ ซึ่งจุดกำเนิดจะมีค่า x, y และ z เป็น 0, 0 และ 0 ตามลำดับ และโดยปกติแล้วทิศทางของค่า x จะเพิ่มขึ้นในทิศทางไปทางด้านขวา และลดลงไปทางด้านซ้ายของจุดกำเนิด ส่วนค่า y จะเพิ่มขึ้นไปทางด้านบน และลดลงไปทางด้านล่าง แต่ค่า z จะมีให้เลือกอยู่ 2 แบบ แบบแรกจะเพิ่มขึ้นในทิศทางเข้าไปในจอภาพ และแบบที่สองจะเพิ่มขึ้นในทิศทางออกจากจอภาพ โดยแบบแรกเรียกว่า Left-HandSystem ส่วนแบบที่สองจะเรียกว่า Right-HandSystem

3.1.1 ระบบพิกัดคาร์ทีเซียน (CartesianSystem)

เป็นระบบพิกัดที่ทำความเข้าใจได้ง่ายที่สุด เพราะเป็นระบบพิกัดที่คนส่วนใหญ่คุ้นเคย และมีใช้อย่างแพร่หลายที่สุด อีกทั้งยังเหมาะสมในการใช้สำหรับสร้างโปรแกรม ระบบพิกัดคาร์ทีเซียนประกอบด้วยสามแกนที่ตั้งฉากซึ่งกันและกันสำหรับกำหนดพิกัด โดยมักจะตั้งชื่อแกนดังกล่าวให้เป็น X, Y และ Z ระบบพิกัดนี้โดยทั่วไปมักมีการกำหนดแนวแกนได้ 2 แบบคือ

1. ระบบพิกัดคาร์ทีเซียนแบบมือซ้าย (Left-handed CartesianSystem)
2. ระบบพิกัดคาร์ทีเซียนแบบมือขวา (Right-handed CartesianSystem)

โดยแสดงได้ดังภาพดังต่อไปนี้

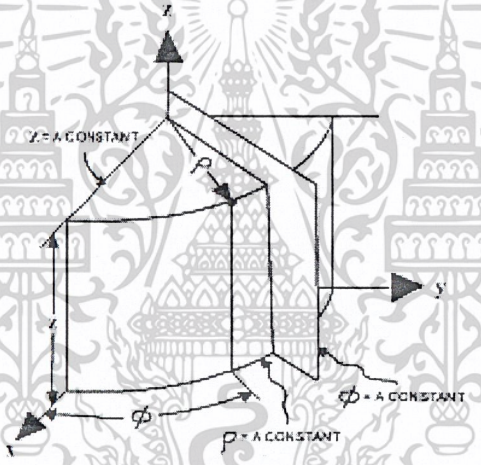


รูปที่ 3-1 ระบบพิกัดคาร์ทีเซียนแบบมือซ้ายและแบบมือขวา

โดยทั่วไปไดเร็คเอ็กGraphics API นั้นระบบพิกัดที่ใช้จะเป็นระบบมือซ้าย ในการแสดงผลภาพ 3 มิตินั้นจะต้องส่งเวอร์เท็กซ์อาเรย์ที่ต้องการแสดงผลเข้าไปใน Geometry Pipeline สุดท้ายก็จะได้ภาพ 2 มิติออกมาแสดงบนจอภาพ เหตุที่ต้องแปลงข้อมูลที่เป็น 3 มิติให้เป็น 2 มิตินั้นเพราะหน้าจอของไม่สามารถแสดงภาพ 3 มิติตรงๆ ได้จึงต้องทำการส่งข้อมูลเข้า Geometry Pipeline เพื่อให้ได้ผลลัพธ์เป็นภาพ 2 มิติที่สอดคล้องกับข้อมูลนั้นบนมุมมองที่ต้องการ

3.1.2 ระบบพิกัดทรงกระบอก (CylindricalSystem)

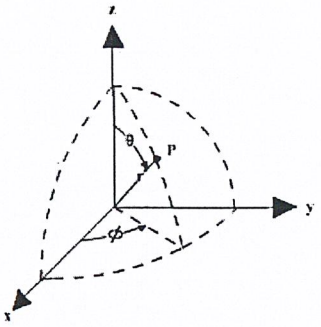
ในระบบพิกัดนี้ สามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด (Origin) ซึ่งแทนด้วยสัญลักษณ์ ρ มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเซียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแกนอ้างอิงในแนวระดับ (ซึ่งสัมพันธ์กับระนาบ XY ในระบบพิกัดคาร์ทีเซียน) ซึ่งแทนด้วยสัญลักษณ์ ϕ และระดับความสูงของจุดนั้นๆ ซึ่งมักแทนด้วย Z ดังแสดงในรูป



รูปที่ 3-2 ระบบพิกัดทรงกระบอก

3.1.3 ระบบพิกัดทรงกลม (SphericalSystem)

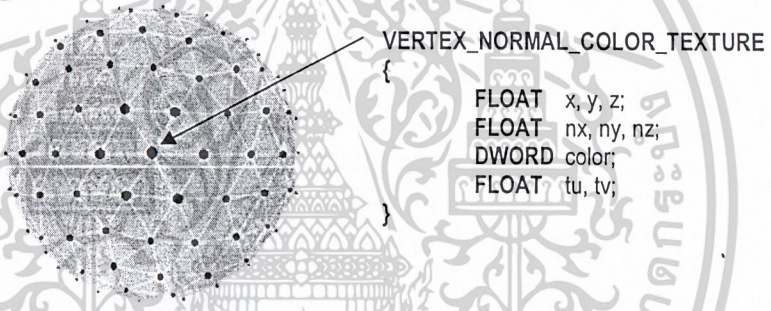
ในระบบพิกัดนี้ สามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด ซึ่งแทนด้วยสัญลักษณ์ r มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเซียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแนวระดับ (ซึ่งสัมพันธ์กับระนาบ XY ในระบบพิกัดคาร์ทีเซียน) ซึ่งแทนด้วยสัญลักษณ์ ϕ และมุมระหว่างแกนตั้งฉากแนวระดับ (แกน Z) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ดังแสดงในรูป



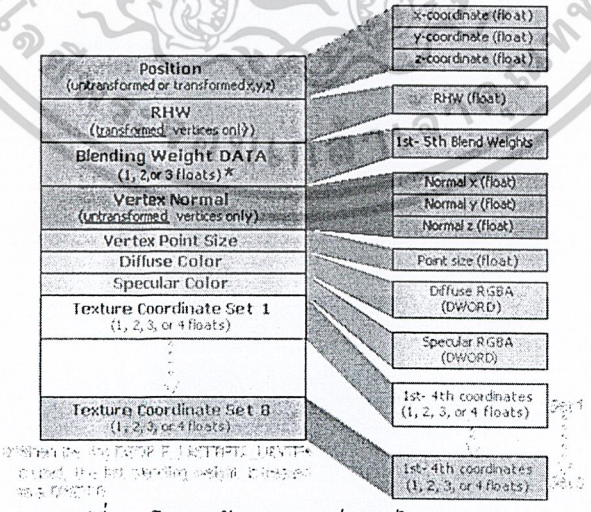
รูปที่ 3-3 ระบบพิกัดทรงกลม

3.2 เวอร์เท็กซ์ (Vertex)

เวอร์เท็กซ์เป็นจุดพิกัดในระบบ 3 มิติ ซึ่งใช้ในการอ้างอิงตำแหน่ง ซึ่งในการประมวลผลทางด้านกราฟิกนั้นมักจะกำหนดสมบัติเพิ่มเติมให้กับเวอร์เท็กซ์ เช่น Color, Normal, Textureเป็นต้น ซึ่งสมบัติเหล่านี้เมื่อประกอบกับตำแหน่งของเวอร์เท็กซ์แล้วจะนำไปใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติได้ โดยแสดงได้ดังรูปตัวอย่าง



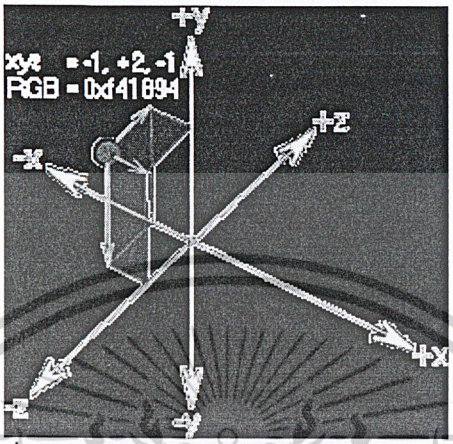
รูปที่ 3-4 ตัวอย่างการนำเวอร์เท็กซ์มาใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติ



รูปที่ 3-5 โครงสร้างของเวอร์เท็กซ์ใน Direct3D

3.3 เวกเตอร์ (Vector)

เวกเตอร์เป็นปริมาณที่มีทั้งขนาดและทิศทาง เป็นองค์ประกอบทางคณิตศาสตร์ที่มีความสำคัญมากด้านการประยุกต์ใช้ในงานการประมวลผลกราฟิก ซึ่งจะต่างกับปริมาณสเกลาร์ตรงที่เวกเตอร์นั้นจะมีทิศทาง หรือกล่าวได้อีกอย่างหนึ่งว่าเวกเตอร์มีองค์ประกอบมาจากปริมาณสเกลาร์ n ตัว เพื่อแทนขนาดและทิศทางในระบบ n มิติ



รูปที่ 3-6 เวกเตอร์ของ RGB และ XYZ ใน Direct3D

มักจะแทนเวกเตอร์โดยใช้สัญลักษณ์ $\overrightarrow{OP}$ โดยหมายถึงเวกเตอร์นี้มีทิศทางจากจุด O ไปยังจุด P และมีขนาดเท่ากับระยะห่างระหว่างจุด O และจุด P

รูปที่ 3-7 ตัวอย่างเวกเตอร์

หากทำการบวกเวกเตอร์ทั้งสองจะได้ค่าดังนี้

$$R = V1 + V2$$
$$R = (V1_X + V2_X, V1_Y + V2_Y, V1_Z + V2_Z)$$

เมื่อทำการคูณเวกเตอร์ด้วยปริมาณสเกลาร์ ซึ่งทำได้โดยการคูณปริมาณสเกลาร์กับทุกคอมโพเน็นต์ของเวกเตอร์นั้น ๆ จะได้

$$V * s = (V_X * s, V_Y * s, V_Z * s)$$

สำหรับขนาด (Magnitude/length) ของเวกเตอร์ใดๆ นั้น สามารถเขียนแทนด้วย $|V|$ ซึ่งสามารถหาได้โดยใช้กฎของพิทากอรัส ได้สมการดังนี้

$$|V| = \sqrt{V_x^2 + V_y^2 + V_z^2}$$

เวกเตอร์ที่ขนานกัน (Parallel Vector) หมายถึง เวคเตอร์คู่ใด ๆ ที่มีทิศทางเดียวกันหรือทิศทางตรงข้ามกัน

เวกเตอร์ร่วมระนาบ (Coplanar Vector) หมายถึง เวกเตอร์ตั้งแต่ 3 ตัวขึ้นไปที่อยู่ในระนาบเดียวกัน

3.4 เมทริกซ์และทรานส์ฟอร์มเมชัน 3 มิติ

3.4.1 เมทริกซ์ (Matrix)

เมทริกซ์มิติ m × n ประกอบด้วยจำนวนจริงที่เขียนเรียงเป็นแถว (Row) m แถว และเขียนในแนวดิ่ง n หลัก (Column) โดยปิดล้อมจำนวนจริงเหล่านี้ด้วยเครื่องหมาย [ ] หรือ ( ) จำนวนแต่ละจำนวนในเมทริกซ์ เรียกว่า สมาชิกของเมทริกซ์

$\begin{bmatrix} 2 & 1 & 0 \\ 5 & 3 & 2 \end{bmatrix}$ เป็นเมทริกซ์มิติ 2 × 3 $\begin{bmatrix} 1 \\ 0 \\ 2 \\ 4 \end{bmatrix}$ เป็นเมทริกซ์มิติ 4 × 1

ถ้าเมทริกซ์ที่มีจำนวนแถวและจำนวนหลักเท่ากันเท่ากับ n จะเรียกเมทริกซ์นั้นว่า เมทริกซ์จัตุรัสมิติ n และเรียก $a_{11}, a_{12}, ..., a_{nn}$ว่าเป็นสมาชิกในแนวทแยงของ A เมื่อ A เป็นเมทริกซ์จัตุรัสมิติ n

a_{11}, a_{22}, a_{33} เป็นสมาชิกในแนวทแยงของ A หากเมทริกซ์จัตุรัส $A = [a_{ij}]$ ซึ่งมีสมาชิกทุกตัวนอกจากแนวทแยงเป็นศูนย์ทั้งหมด (นั่นคือ $a_{ij} = 0$ ถ้า $I \neq j$) เรียก A ว่าเป็น เมทริกซ์ทแยง (Diagonal Matrix)

ให้ $A = [a_{ij}]_{m \times n}$ และ $B = [b_{ij}]_{m \times n}$ เป็นเมทริกซ์ที่มีมิติเท่ากัน เรียก A เท่ากับ B ก็ต่อเมื่อ $a_{ij} = b_{ij}$ สำหรับ $1 \leq i \leq m, 1 \leq j \leq n$ และเขียนแทนด้วย A = B

ถ้า $A=[a_{ij}]_{m\times n}$ และ $B=[b_{ij}]_{m\times n}$ แล้วผลบวกของ A และ B เขียนแทนด้วย A + B หมายถึง $C=[c_{ij}]_{m\times n}$ ซึ่ง

$$c_{ij}=a_{ij}+b_{ij}\ (1\le I\le m,\ 1\le j\le n)$$

ให้ $A=[a_{ij}]_{m\times n}$ เป็นเมทริกซ์ และ k เป็นจำนวนจริงใดๆ แล้ว ผลคูณสเกลาร์ kA จะเป็นเมทริกซ์ $[ka_{ij}]_{m\times n}$ เช่น

ให้ $A=\begin{bmatrix}2 & 1 & 0\\ 5 & 3 & 2\end{bmatrix}$ และ $k=3$

ดังนั้น $kA=\begin{bmatrix}3(2) & 3(1) & 3(0)\\ 3(5) & 3(3) & 3(2)\end{bmatrix}=\begin{bmatrix}6 & 3 & 0\\ 15 & 9 & 6\end{bmatrix}$

ให้ $A=[a_{ij}]_{m\times p}$ และ $B=[b_{ij}]_{p\times n}$ แล้ว ผลคูณของ A และ B เขียนแทนด้วย AB หมายถึง

$c=[a_{ij}]_{m\times n}$ โดย $c_{ij}=a_{i1}b_{1j}+a_{i2}b_{i2}+\ldots+a_{ip}b_{ip}$

$$c=\sum_{k=1}^{p}a_{ik}\,b_{kj}\quad (1\le i\le m,\ 1\le j\le n)$$

เช่น

$$AB=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}\begin{bmatrix}b_{11} & b_{12} & b_{13}\\ b_{21} & b_{22} & b_{23}\end{bmatrix}$$

$$=\begin{bmatrix}a_{11}b_{11}+a_{12}b_{21} & a_{11}b_{12}+a_{12}b_{22} & a_{11}b_{13}+a_{12}b_{23}\\ a_{21}b_{11}+a_{22}b_{21} & a_{21}b_{12}+a_{22}b_{22} & a_{21}b_{13}+a_{22}b_{23}\end{bmatrix}$$

ผลคูณ AB ไม่มีหรือไม่นิยาม (Undefined) ถ้า A เป็นเมทริกซ์ขนาด $m\times p$ และ B เป็นเมทริกซ์ขนาด $q\times n$ เมื่อ $p\neq q$

การคูณเมทริกซ์นั้นไม่มีสมบัติการสลับที่ซึ่งหมายถึง $A\times B\neq B\times A$ เมื่อ A และ B เป็นเมทริกซ์จัตุรัส

เมทริกซ์เอกลักษณ์ (Identity Matrix) คือ เมทริกซ์จัตุรัสที่มีสมาชิกในแนวทแยงเป็น 1 ทั้งหมด ซึ่งเขียนแทนด้วย I หรือ I_n แทนเมทริกซ์เอกลักษณ์มิติ n เช่น

$$\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

ถ้า A เป็นเมทริกซ์มิติ $m\times n$ แล้วจะได้ว่า $AI_n=A$ และ $I_mA=A$

B เป็นอินเวิร์สการคูณของเมทริกซ์ A (B เป็นอินเวิร์สของ A) เมื่อ B เป็นเมทริกซ์ซึ่ง $AB=BA=I$ โดยที่ A เป็นเมทริกซ์จัตุรัสใดๆ

ให้ A เป็นเมทริกซ์จัตุรัสมิติ n จะกล่าวว่า A มีตัวผกผัน (Invertible) หรือ A เป็นเมทริกซ์ซึ่งมิใช่เอกฐาน (Non-Singular Matrix) ถ้าสามารถหาเมทริกซ์จัตุรัส B ได้ ซึ่งทำให้

$$AB = I_n = BA$$

และเรียกเมทริกซ์ B ว่าเป็นอินเวิร์สการคูณ ของ A เขียนแทนด้วยสัญลักษณ์ A^{-1} นั่นคือ ถ้า A เป็นเมทริกซ์ซึ่งมิใช่เมทริกซ์เอกฐานมิติ n แล้ว จะได้ว่า

$$AA^{-1} = A^{-1}A = I_n$$

และถ้า A ไม่มีอินเวิร์ส แล้วจะเรียก A ว่าเป็นเมทริกซ์เอกฐาน (Singular Matrix)

3.4.2 Translation

เป็นการเคลื่อนย้ายตำแหน่งของวัตถุ ซึ่งใช้ Translation Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ T_x & T_y & T_z & 1 \end{bmatrix}$$

รูปที่ 3-8 Translation Matrix

3.4.3 Rotation

เป็นการหมุนวัตถุรอบแกน X, Y หรือ Z ซึ่งใช้ Rotation Matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

รูปที่ 3-9 Rotation around X Axis Matrix

$$\begin{bmatrix} \cos\theta & 0 & -\sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

รูปที่ 3-10 Rotation around Y Axis Matrix

$$\begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

รูปที่ 3-11 Rotation around Z Axis Matrix

3.4.4 Scaling

เป็นการย่อหรือขยายวัตถุ ซึ่งใช้ Scaling Matrix

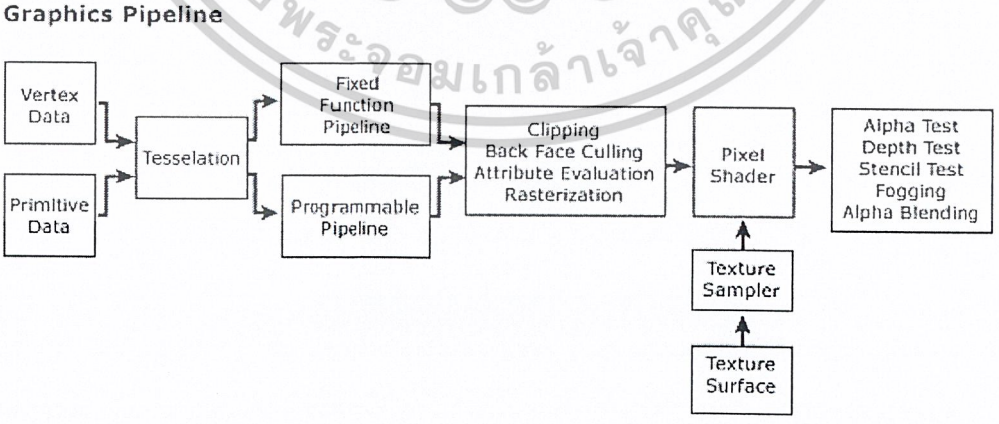
$$\begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

รูปที่ 3-12 Scaling Matrix

3.5 การแสดงผลภาพ 3 มิติด้วย Direct3D

3.5.1 สถาปัตยกรรมของ Direct3D

สถาปัตยกรรมของ Direct3D ประกอบด้วยหน้าที่การทำงานต่างๆ ซึ่งจะรวมกันทำงานแบบ Pipeline ดังที่แสดงตามรูป



รูปที่ 3-13 Pipeline การทำงานของ Direct3D

3.5.2 ทรานส์ฟอร์มเมชัน

3.5.2.1 การแปรไปสู่พิกัดเวิลด์ (World Transformation)

ขั้นตอนนี้จะเป็นขั้นตอนในการแปลง Localไปเป็น Worldซึ่งจะเป็นตำแหน่งจริงๆ ในปริภูมิ 3 มิติ ซึ่งวิธีการนั้นจะใช้การคูณตำแหน่งในเวอร์เท็กซ์ด้วย Transformationเมทริกต่าง ๆ เพื่อให้ได้ตำแหน่งที่ต้องการ

ในการแปลงเวอร์เท็กซ์จาก Localให้เป็น Worldนั้นในบางครั้งถ้าการแปลงของมีความซับซ้อนมากอาจจะต้องใช้เมทริกซ์หลายตัวในการแปลงคูณต่อกันไป (Matrix Concatenation) เพื่อสร้างเมทริกซ์สุดท้ายสำหรับใช้เป็น Transformationเมทริกก่อนแล้วจึงจะเอามาคูณกับตำแหน่งในเวอร์เท็กซ์จริงเพื่อให้ได้ผลลัพธ์สุดท้ายก็ได้

การคูณเมทริกกับตำแหน่งของเวอร์เท็กจะเป็นดังต่อไปนี้

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix} = \begin{bmatrix} x & y & z & 1 \end{bmatrix} \begin{bmatrix} M_{11} & M_{12} & M_{13} & M_{14} \\ M_{21} & M_{22} & M_{23} & M_{24} \\ M_{31} & M_{32} & M_{33} & M_{34} \\ M_{41} & M_{42} & M_{43} & M_{44} \end{bmatrix}$$

$$x' = (x \times M_{11}) + (y \times M_{21}) + (z \times M_{31}) + (1 \times M_{41})$$

$$y' = (x \times M_{12}) + (y \times M_{22}) + (z \times M_{32}) + (1 \times M_{42})$$

$$z' = (x \times M_{13}) + (y \times M_{23}) + (z \times M_{33}) + (1 \times M_{43})$$

รูปที่ 3-14 การคูณเมทริกกับตำแหน่งเวอร์เท็ก

จากรูป x, y, z คือ ตำแหน่งของเวอร์เท็กซ์พอคูณกับเมทริกซ์ดังรูปแล้ว จะได้ผลลัพธ์ เป็น x', y', z' ซึ่งเป็นผลลัพธ์ของการแปลงพิกัด

3.5.2.2 การแปรไปสู่พิกัดวิว (View Transformation)

ขั้นตอนนี้ใช้สำหรับแปลง Worldให้สอดคล้องกับการเห็นของผู้สังเกต เนื่องจาก Worldบอกเพียงตำแหน่งจริงๆ ในพิกัด 3 มิติเท่านั้น ยังไม่มีการอ้างอิงจากผู้สังเกตเลย ดังนั้นจึงต้องมีการบอก ว่าผู้สังเกตอยู่ที่ไหน และมองไปทางไหน ขั้นตอนนี้มักใช้ในการปรับมุมมองของกล้อง โดยการนำวิวเมทริกซ์มาคูณกับ Worldจากการแปรไปสู่พิกัดเวิลด์

$$\begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ -(u \cdot c) & -(v \cdot c) & -(n \cdot c) & 1 \end{bmatrix}$$

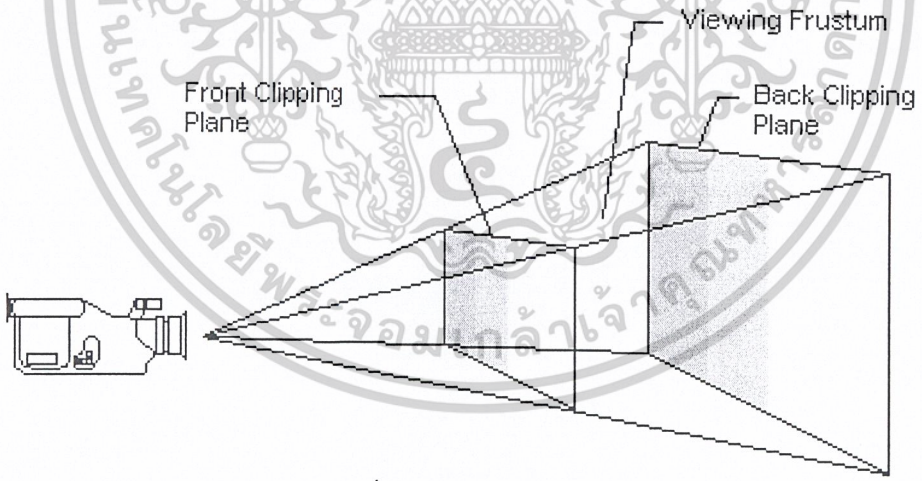
รูปที่ 3-15 การแปรไปสู่พิกัดวิว

จากรูปวิวเมทริกซ์ประกอบด้วยเวกเตอร์ 3 ตัวบอกทิศทางของผู้สังเกตว่าหันหน้าไปทางใด ซึ่งมีเวกเตอร์ u, v, n บอกถึง Up, Right, และ View Direction ตามลำดับ และเวกเตอร์ c ซึ่งบอกตำแหน่งของผู้สังเกตในพิกัด World

วิวเมทริกซ์นั้นให้ข้อมูลที่จำเป็นในการปรับมุมมองของกล้องในเกมเอนจินได้ ซึ่งใช้ในการบอกตำแหน่งและทิศทางของกล้อง

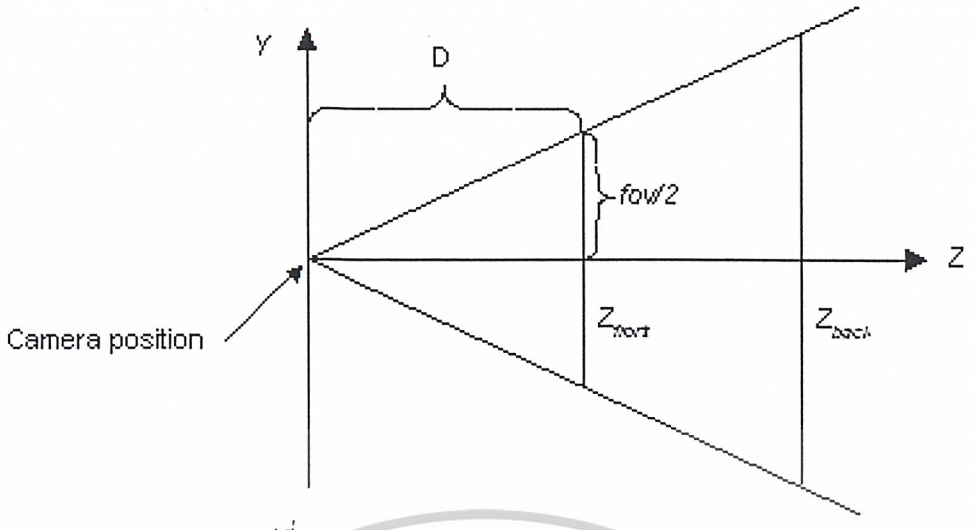
3.5.2.3 การแปรไปสู่พิกัดโปรเจ็กต์ชัน (Projection Transformation)

เมื่อได้ตำแหน่งซึ่งอ้างอิงกับผู้สังเกตเรียบร้อยแล้ว ขั้นตอนต่อไปก็คือการแปลง Viewมาเป็นพิกัด 2 มิติ ซึ่งสามารถนำมาวาดได้จริงๆ บนจอภาพ ซึ่งโดยทั่วไปแล้วต้องการให้ได้ภาพที่สมจริง คือ วัตถุที่อยู่ไกลจากสายตาจะมีขนาดเล็กลง และวัตถุที่อยู่ใกล้สายตาจะมีขนาดใหญ่ขึ้น ซึ่งเรียกว่า Perspective Projection Transformation ซึ่งเป็นการสร้างภาพฉายมาตกกระทบฉาก คล้ายๆ กับการฉายภาพจากโปรเจคเตอร์

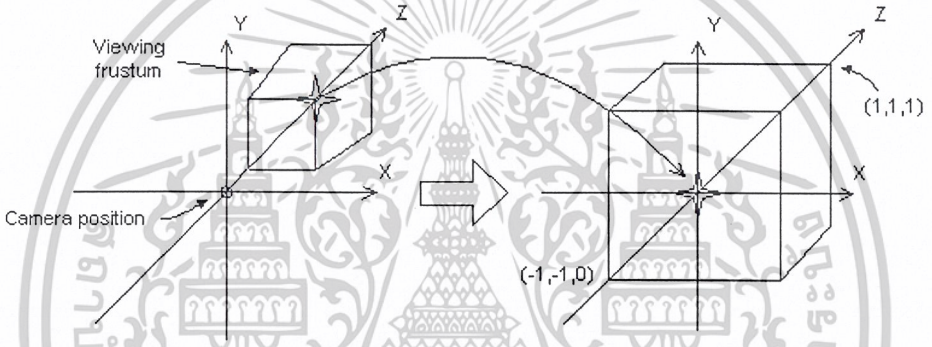


รูปที่ 3-16 Viewing Frustum

วัตถุที่อยู่ใน Viewing Frustum จะถูกนำมาแสดงผล เนื่องจากเป็นส่วนที่จอภาพสามารถเห็นได้ ส่วนวัตถุที่อยู่ภายนอก Viewing Frustum จะถูกตัดออกระหว่างการประมวลผลก่อนที่จะนำไปแสดงผล



รูปที่ 3-17 Viewing Frustum มองจากแนวแกน X



รูปที่ 3-18 การแปลงจาก Viewing Frustum ไปเป็น Cuboid shape

ซึ่งจะทำให้วัตถุที่อยู่ไกลมีขนาดเล็กลง และวัตถุที่อยู่ใกล้มีขนาดใหญ่ขึ้นตามสัดส่วนของระยะห่างจากสายตาผู้สังเกต และสามารถทำ Perspective Projection Transformation ได้โดยการใช้ Perspective Projectionเมทริกดังนี้

$$\begin{bmatrix} w & 0 & 0 & 0 \\ 0 & h & 0 & 0 \\ 0 & 0 & Q & 1 \\ 0 & 0 & -QZ_n & 0 \end{bmatrix}$$

$$w = \cot\left(fov_w/2\right)$$

$$h = \cot\left(fov_h/2\right)$$

$$Q = \frac{Z_f}{Z_f - Z_n}$$

รูปที่ 3-19 Perspective Projection Metrix

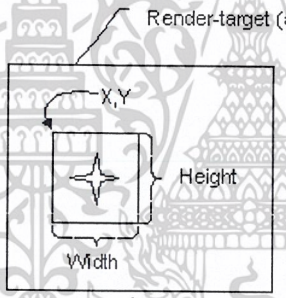
โดยทั่วไปแล้วกราฟิกเอนจินนั้นมักจะมีฟังก์ชันที่ใช้ทำการคำนวณหา Perspective Projection Matrix โดยให้กำหนดค่า Field of View (fov) ตามแนวความสูงของฉาก (โดยทั่วไปมักเป็นแกน Y) ค่า Aspect Ratio (w/h) ค่า Near Clipping Plane (Z_n) และค่า Far Clipping Plane (Z_f)

นอกจากการทำ Perspective Projection Transformation แล้วยังมีการทำ Projection Transformation แบบอื่นๆ อีก เช่น Isometric Projection Transformation, Oblique Projection Transormation เป็นต้น

3.5.3 Clipping และ Viewport Scaling

Viewport ในที่นี้หมายถึงพื้นที่สี่เหลี่ยมผืนผ้าที่ใช้ในการฉายภาพที่เกิดจากการแปรไปสู่พิกัดโปรเจ็กต์ชัน ซึ่งเป็นขั้นตอนสุดท้ายใน Geometry Pipeline และทำให้ได้ภาพฉายตกลงบน Viewport เพื่อทำกระบวนการ Rasterization ต่อไปในการสร้างภาพเสมือนจริงบนหน้าจอ

Viewport อาจหมายถึงหน้าจอทั้งหน้าจอหรือบางส่วนของหน้าจอก็ได้ ทั้งนี้ขึ้นอยู่กับโปรแกรม ซึ่งโดยทั่วไปแล้วเกมมักจะแสดงผลแบบเต็มจอ แต่การใช้ Viewport อาจมีประโยชน์เมื่อต้องการการแสดงผล 3 มิติหลายหน้าต่าง



รูปที่ 3-20 Direct3D Viewport

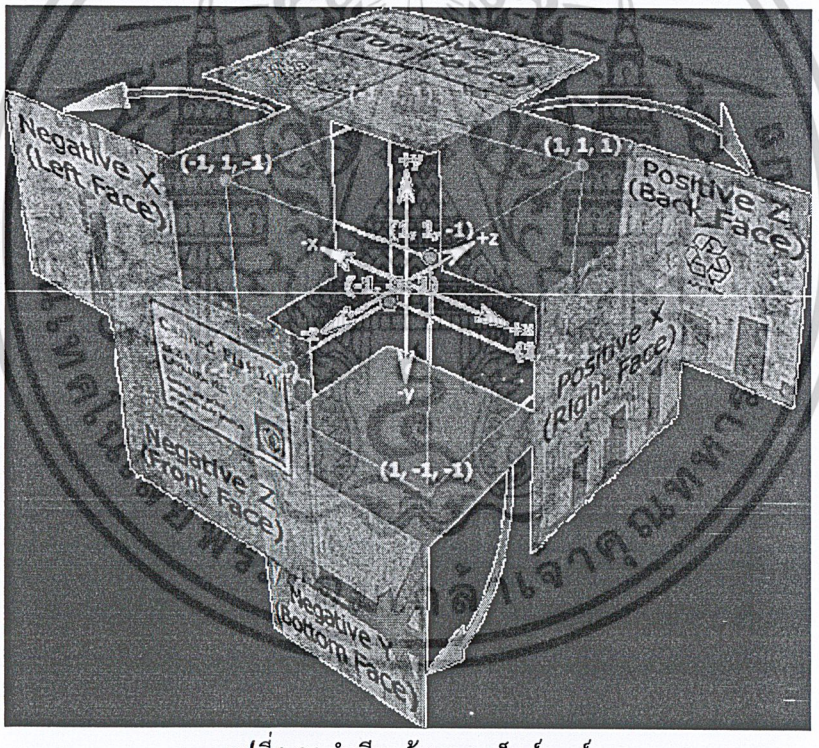
โดยทั่วไปแล้วจะทำการขริบ (Clipping) สิ่งที่มองไม่เห็นบนหน้าจอออกไป ในกรณีที่ Viewport แสดงถึงบางส่วนของหน้าจอนั้น จะขริบส่วนของภาพที่อยู่นอกกรอบ Viewport ออกไป ทั้งนี้รวมถึงการขริบในแนวแกน Z ด้วย (จะตัดทอนส่วนของภาพที่อยู่ไกลเกิน Viewport ที่กำหนด หรือ ส่วนของภาพที่อยู่ด้านหลัง Near Clipping Plane ของ Viewing Frustum ออก)

กระบวนการนี้โดยมากมักจัดการโดยกราฟิกเอนจินโดยกำหนดค่าขอบเขตของ Viewport และระยะการตัดทอนตามแนวแกน Z ก็เพียงพอแล้ว

3.6 เสปซ

3.6.1 โมเดลเสปซ

โมเดลเสปซ คือพิกัด (3D Coordinate) ที่ใช้ภายในโมเดลหรือวัตถุแต่ละอัน ซึ่งมีขอบเขตเป็นสี่เหลี่ยม โมเดลจะประกอบขึ้นมาจากรูปสามเหลี่ยมหลายๆ ชิ้น และรูปสามเหลี่ยมแต่ละชิ้นก็เกิดขึ้นจากเวอร์เท็กจำนวน 3 จุด โดยแต่ละจุดก็จะอ้างอิงถึงจุดกำเนิด (Origin) จุดเดียวกัน (จุดกำเนิดจะมีค่า x, y, x เป็น 0, 0, 0) ซึ่งปกติแล้ว จุดกำเนิดนี้จะอยู่ที่กลางโมเดลแต่ละอัน ซึ่งพิกัดต่างๆ ที่โมเดลแต่ละอันใช้นี้จะเรียกว่า Local ขอบเขตของ โมเดลเสปซ จะกว้าง, ยาว และสูงแค่ไหนก็ขึ้นอยู่กับขนาดของโมเดล จากรูปจะสร้างโมเดลเป็นกล่องกระดาษรูปทรงสี่เหลี่ยมหรือลูกบาศก์ ขนาดกว้าง 2 หน่วย, ยาว 2 หน่วย และสูง 2 หน่วย โดยจะมีจุด Origin อยู่ที่ใจกลางกล่อง ซึ่งโดยพื้นฐานของกล่องสี่เหลี่ยมแล้ว จะประกอบไปด้วยด้านสี่เหลี่ยม 6 ด้าน และจะทำให้ด้านแต่ละด้านของกล่องๆ นี้มีเท็กซ์เจอร์ไม่เหมือนกัน โดยภาพเท็กซ์เจอร์ทั้ง 6 ที่จะนำมา Map เข้ากับด้านแต่ละด้านนั้น จะถูกเรียงตามรูปแบบดังภาพด้านขวามือ ซึ่งวิธีเรียกด้านแต่ละด้านนี้จะใช้แกน x, y และ z เป็นตัวกำหนด โดยเรียงค่าตามลำดับในแนวแกน x, y และ z ดังนี้

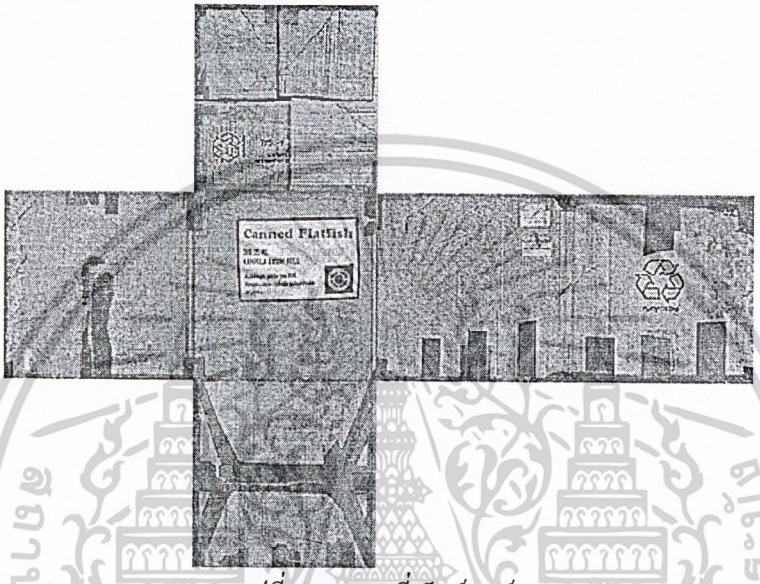


รูปที่ 3-21 คำเรียกด้านของเท็กซ์เจอร์

1. ด้านที่อยู่ในแกน x ทางด้านบวก จะเรียกว่า ด้าน Positive X หรือด้านขวา (Right Face)
2. ด้านที่อยู่ในแกน x ทางด้านลบ จะเรียกว่า ด้าน Negative X หรือด้านซ้าย (Left Face)
3. ด้านที่อยู่ในแกน y ทางด้านบวก จะเรียกว่า ด้าน Positive Y หรือด้านบน (Top Face)

4. ด้านที่อยู่ในแกน y ทางด้านลบ จะเรียกว่า ด้าน Negative Y หรือด้านล่าง (Bottom Face)
5. ด้านที่อยู่ในแกน z ทางด้านบวก จะเรียกว่า ด้าน Positive Z หรือด้านหลัง (Back Face)
6. ด้านที่อยู่ในแกน z ทางด้านลบ จะเรียกว่า ด้าน Negative Z หรือด้านหน้า (Front Face)

และด้านล่างนี้เป็นภาพเท็กซ์เจอร์จำนวน 6 ภาพ ซึ่งแต่ละภาพมีขนาด 128 x 128 เมื่อนำมาเรียงต่อกันตามรูปแบบข้างบน ก็จะดูเหมือนกล่องกระดาษที่ถูกคลี่ออกมา หรือดูเป็นกระดาษที่ยังไม่ได้ถูกพับให้เป็นกล่อง



รูปที่ 3-22 การคลี่เท็กซ์เจอร์

จะใช้ขนาดเท็กซ์เจอร์เท่าไรก็ได้ แต่ถ้ายิ่งขนาดใหญ่มาก ก็จะยิ่งทำให้เฟรมเรทลดลงด้วย (จะทำให้ภาพกระตุก) บางทีก็ต้องดูด้วยว่า การ์ด 3D รองรับขนาดเท็กซ์เจอร์สูงสุดได้แค่ไหน เช่น ถ้าเป็นการ์ด Voodoo Graphics หรือ Voodoo รุ่นแรก จะ Support ที่ 256 x 256 Texel แต่ถ้าเป็นการ์ด Voodoo5 5500 ที่ใช้อยู่ปัจจุบัน จะ Support ได้ถึง 2048 x 2048 Texel ซึ่งการจะรองรับได้ขนาดไหนนั้น จะอยู่ที่ขนาดของหน่วยความจำของการ์ด 3D ด้วย (Voodoo Graphics มี RAM ขนาด 4 MB ส่วน Voodoo5 5500 มี 64 MB)

เกม 3D จะมีการระบุว่าต้องการการ์ด 3D ที่มี RAM เท่าไรเป็นอย่างต่ำ เพื่อที่จะได้รองรับขนาดเท็กซ์เจอร์สูงสุดที่ใช้ในการเขียนเกมนั้นๆ ได้ เพราะการ Map รูปภาพเท็กซ์เจอร์ลงไปบนโมเดล จะต้องมีการ Copy รูปภาพจากหน่วยความจำปกติ ไปไว้ในหน่วยความจำของการ์ด 3D ถ้ามีการเปลี่ยนเท็กซ์เจอร์อันใหม่ ก็จะต้อง Copy ลงไปใหม่ ดังนั้นถ้าโปรแกรมที่เขียนมีการเปลี่ยนเท็กซ์เจอร์เพื่อใช้ในการ Map บ่อยๆ จะทำให้ Frame Rate ลดลง วิธีที่ถูกก็คือ ควรวาดภาพโดยเรียงตามเท็กซ์เจอร์ที่ใช้ คือ Group รูปสามเหลี่ยมหรืออาจจะ Group จุดเวอร์เท็กที่ใช้เท็กซ์เจอร์เดียวกันเข้าด้วยกัน ตัวอย่างเช่น ถ้าต้องการวาดกล่องสี่เหลี่ยมที่เหมือนกันจำนวน 10 กล่องลงไปในฉาก โดยแต่ละกล่องมีการใช้เท็กซ์เจอร์จำนวน 6 ภาพที่ต่างกันสำหรับใช้ Map ลงไปที่ด้านแต่ละด้านของกล่อง ถ้าใช้วิธี Group ตามจำนวนกล่อง หรือวาดทีละ

กล่อง คือวาดกล่องแรกจนเสร็จ แล้ววาดกล่องต่อๆ ไป ซึ่งการทำแบบนี้จะต้องมีการเปลี่ยนเท็กซ์เจอร์ถึง 60 ครั้ง (ทำวนรอบนับจำนวนกล่อง 10 รอบ และแต่ละรอบมีวรรอบเปลี่ยนเท็กซ์เจอร์อีก 6 รอบ รวมจำนวนรอบของ Loop ทั้งหมดเป็น 10 x 6 ซึ่งเท่ากับ 60)

แต่ถ้าใช้วิธีนี้ กล่องไหนมีด้านที่ต้องใช้เท็กซ์เจอร์ที่กำลังใช้อยู่ ก็วาดด้านนั้นให้เสร็จก่อน แล้วจึงเปลี่ยนเท็กซ์เจอร์เป็นด้านต่อไป และวาดด้านอื่นของกล่องทั้งหมดที่ใช้เท็กซ์เจอร์อันใหม่นี้ ทำอย่างนี้ไปเรื่อยๆ จะมีการเปลี่ยนเท็กซ์เจอร์เพียง 6 ครั้งเท่านั้น ทั้งๆ ที่ใช้จำนวนรอบรวมของการวนรอบเท่ากัน (วนรอบนับจำนวนเท็กซ์เจอร์6 รอบ และแต่ละรอบมีการวนรอบภายในเปลี่ยนกล่องที่จะวาด 10 รอบ รวมจำนวนรอบของ Loop ทั้งหมดเป็น 6 x 10 ซึ่งเท่ากับ 60) ซึ่งในตัวอย่างโปรแกรมจะวาดโดย Group ด้านที่ใช้เท็กซ์เจอร์เดียวกันเข้าด้วยกัน

ตามคำแนะนำในคู่มือไดเร็คเอ็กSDK นั้นบอกว่า ควรใช้ขนาดเท็กซ์เจอร์ที่เล็กที่สุดเท่าที่จะทำได้ นอกจากนี้ควรใช้เท็กซ์เจอร์ที่เป็นรูปสี่เหลี่ยมจตุรัสที่มีความกว้างและความสูงเท่ากัน (ปกติอาจจะใช้แบบไม่เท่ากันก็ได้ เช่น 256 x 128 หรือ 100 x 50 เป็นต้น) โดยขนาดของเท็กซ์เจอร์ที่เหมาะสมที่สุดก็คือ 256 x 256 และควรพยายามทำให้เท็กซ์เจอร์เล็กหลายๆ อัน มารวมกันให้เป็นขนาด 256 x 256 เช่น สมมุติว่ามีเท็กซ์เจอร์ขนาด 128 x 128 จำนวน 4 รูป ก็ควรนำรูปทั้ง 4 มาต่อกันให้เป็นขนาด 256 x 256

สี่เหลี่ยมที่มองเห็นจาก Direc3D จะต้องประกอบขึ้นมาจากรูปสามเหลี่ยม 2 รูป (การ์ด 3D ส่วนมากจะ Render หรือทำงานกับรูปสามเหลี่ยม, เส้นตรง และจุดเป็นหลัก) และสามเหลี่ยมแต่ละรูปก็ประกอบขึ้นมาจากเวอร์เท็กจำนวน 3 จุด แต่ไม่จำเป็นว่า รูปสี่เหลี่ยม 1 รูปที่เกิดจากรูปสามเหลี่ยม 2 รูปมาประกบกันจะมีเวอร์เท็กจำนวน 6 จุดเสมอไป ขึ้นอยู่กับ Draw Primitive ที่จะใช้ ซึ่ง Draw Primitive ก็คือ "การวาดรูปทรงต่างๆ จากการเรียงของเวอร์เท็กแต่ละจุด" โดยรูปแบบของ Draw Primitive ใน Direct3D จะมีอยู่ 6 แบบดังนี้

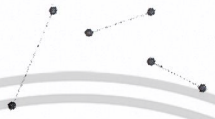
สมมุติว่าใช้เวอร์เท็กจำนวน 6 จุด ในการวาด Primitive แต่ละแบบ โดยเวอร์เท็กจะเรียงกันดังนี้

- Vertex จุดที่ 1 = (-5.0f, -15.0f, 0.0f)
- Vertex จุดที่ 2 = (0.0f, 5.0f, 0.0f)
- Vertex จุดที่ 3 = (3.0f, 0.0f, 0.0f)
- Vertex จุดที่ 4 = (10.0f, 5.0f, 0.0f)
- Vertex จุดที่ 5 = (10.0f, -5.0f, 0.0f)
- Vertex จุดที่ 6 = (15.0f, -10.0f, 0.0f)

1. Point List จะวาดจุด (Point) ตามลำดับการเรียงของเวอร์เท็กแต่ละจุด (1 Point ต่อ 1 Vertex) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_POINTLIST และผลลัพธ์จะเป็นไปตามรูปด้านล่างนี้

รูปที่ 3-23 Point List

2. Line List จะวาดเส้นตรง (Line) เรียงตามลำดับการเรียงของเวอร์เท็กทุกๆ 2 จุด โดยเส้นแต่ละเส้นจะแยกออกจากกัน (1 Line ต่อ 2 Vertex) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_LINELIST



รูปที่ 3-24 Line List

3. Line Strip จะวาดเส้นตรงเรียงตามลำดับการเรียงของเวอร์เท็กโดยที่เวอร์เท็กที่ 2 ของ Line ที่เพิ่งวาดไปจะกลายเป็นเวอร์เท็กจุดที่ 1 ของ Line ถัดไป คือใช้เวอร์เท็กร่วมกัน 1 จุด ผลลัพธ์ที่ได้คือ Line ที่ต่อกันไปเรื่อยๆ ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_LINESTRIP

รูปที่ 3-25 Line Strip

4. Triangle List จะวาดสามเหลี่ยม (Triangle) เรียงตามลำดับการเรียงของเวอร์เท็กทุกๆ 3 จุด โดยสามเหลี่ยมแต่ละรูปจะแยกออกจากกัน (1 Triangle ต่อ 3 Vertex) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLELIST

รูปที่ 3-26 Triangle List

5. Triangle Strip จะวาดสามเหลี่ยมเรียงตามลำดับการเรียงของเวอร์เท็กโดยเวอร์เท็กที่ 2 และ 3 ของรูปสามเหลี่ยมอันที่เพิ่งวาดไป จะกลายเป็นเวอร์เท็กที่ 1 และ 2 ของรูปสามเหลี่ยมรูปถัดไป คือใช้เวอร์เท็กร่วมกัน 2 จุด ผลลัพธ์ที่ได้ก็คือ รูปสามเหลี่ยมที่ต่อกันไปเรื่อยๆ (ถ้าจะ

ใช้วิธีนี้ ควรจะเรียงเวอร์เท็กแบบซิกแซ็ก) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLESTRIP

รูปที่ 3-27 Triangle Strip

6. Triangle Fan จะวาดสามเหลี่ยมเรียงตามลำดับการเรียงของเวอร์เท็กโดย มีเวอร์เท็กแรกสุดเป็นเวอร์เท็กที่ 1 ของสามเหลียมทุกรูป และเวอร์เท็กที่ 3 ของรูปสามเหลี่ยมอันที่เพิ่งวาดไปจะกลายเป็นเวอร์เท็กที่ 2 ของรูปสามเหลี่ยมรูปถัดไป ผลลัพธ์ก็คือ จะได้รูปสามเหลี่ยมที่เรียงตัวต่อกันเป็นรูปที่คล้ายพัด ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLEFAN

รูปที่ 3-28 Triangle Fan

เหตุที่ต้องมี Draw Primitive หลายๆ แบบให้เลือกใช้ก็เพื่อประหยัดหน่วยความจำ และเพิ่ม Frame Rate เพราะยิ่งใช้เวอร์เท็กน้อยเท่าไร ก็ยิ่งดีเท่านั้น ยกตัวอย่างรูปกล่องสี่เหลี่ยมที่อธิบายไว้แล้ว แต่ละด้านจะเป็นรูปสี่เหลี่ยม ซึ่งเกิดจากรูปสามเหลี่ยมสองรูปมาประกบกัน ถ้าจะวาดกล่องโดยใช้ Draw Primitive เป็น Triangle List จะต้องใช้เวอร์เท็กจำนวน 36 จุด เพราะ สามเหลี่ยมแต่ละรูปเกิดขึ้นมาจากเวอร์เท็กจำนวน 3 จุด ดังนั้นจึงต้องใช้เวอร์เท็ก6 จุดต่อการวาดรูปสี่เหลี่ยม 1 รูป และเนื่องจากกล่องมี 6 ด้าน ดังนั้นต้องใช้เวอร์เท็กทั้งสิน 6 x 6 ซึ่งเท่ากับ 36 จุด แต่ถ้าใช้ Draw Primitive แบบ Triangle Strip จะใช้เวอร์เท็กแค่ 4 จุดต่อ 1 ด้านเท่านั้น รวมแล้วก็ 4 x 6 ซึ่งเท่ากับ 24 จุด

สำหรับ เมธอด ที่ใช้ในการวาดรูปก็คือ IDirect3DDevice8::DrawPrimitive() โดยมีพารามิเตอร์ตัวแรกเป็นค่าคงที่ของ Draw Primitive ที่จะใช้ (ให้ดูค่าคงที่ในหัวข้อ Draw Primitive ทั้ง 6 แบบข้างบน) Parameter ตัวที่สองจะเป็นตำแหน่งของเวอร์เท็กที่จะเริ่มวาดรูปสามเหลี่ยม และ Parameter ตัวสุดท้ายจะบอกให้รู้ว่า จะให้วาดสามเหลี่ยมจำนวนกี่รูป จากตัวอย่างโปรแกรม จะทำ Loop ไล่ตั้งแต่เท็กซ์เจอร์รูปแรกไปจนถึงรูปที่ 6 ที่ต้องทำ Loop ก็เพราะ แต่ละด้านใช้เท็กซ์เจอร์ต่างกัน แล้วใช้ เมธอด IDirect3DDevice8::SetTexture() เพื่อสั่งให้ Load ภาพเท็กซ์เจอร์ที่จะใช้ Map ด้านนั้นๆ เข้าไปไว้ในหน่วยความจำของการ์ด 3D แล้วก็วาดด้านตามการเรียงเวอร์เท็กตามที่กำหนดไว้ในตัวแปร g_vtVertex

3.6.2 World Space

World Space นั้น ถ้าพูดง่ายๆ ก็คือโลก 3 มิติ มันจะมีขนาดใหญ่กว่าโมเดลเสปซเพราะโมเดลต่างๆ ที่สร้างจะถูกแปลงให้มาปรากฏอยู่ใน World Space โดยหลังจากที่โมเดลได้ถูกนำมาใส่ไว้ใน World Space นี้แล้ว พวกมันจะมีจุด Origin เป็นจุดเดียวกัน สำหรับขนาดความกว้างของ World Space จะเป็น 2 ยกกำลัง 32 และความสูงจะเป็น 2 ยกกำลัง 32 สำหรับความลึกก็ขึ้นอยู่กับหน่วยความจำในแนวลึก (Depth Buffer หรือบางทีก็เรียก Z Buffer) ใน World Space นี้ สามารถทำอะไรกับโมเดลก็ได้ เช่น ย้ายตำแหน่ง (Translation), หมุน (Rotation) และขยายหรือลดขนาด (Scaling) เป็นต้น

3.6.3 Camera Space

เป็นมุมมองของกล้อง หรือดวงตา เหมือนกับกรณีที่มองไปที่สิ่งต่างๆ รอบๆ ตัว ซึ่งจะเห็นเฉพาะสิ่งที่ปรากฏอยู่ในสายตาเท่านั้น สิ่งที่อยู่ในสายตานี่แหละเรียกว่า Camera Space ดังนั้นสรุปได้ว่า Camera Space จะเป็นพื้นที่แบบสามมิติที่ดึงมาจากบางส่วนของ World Space

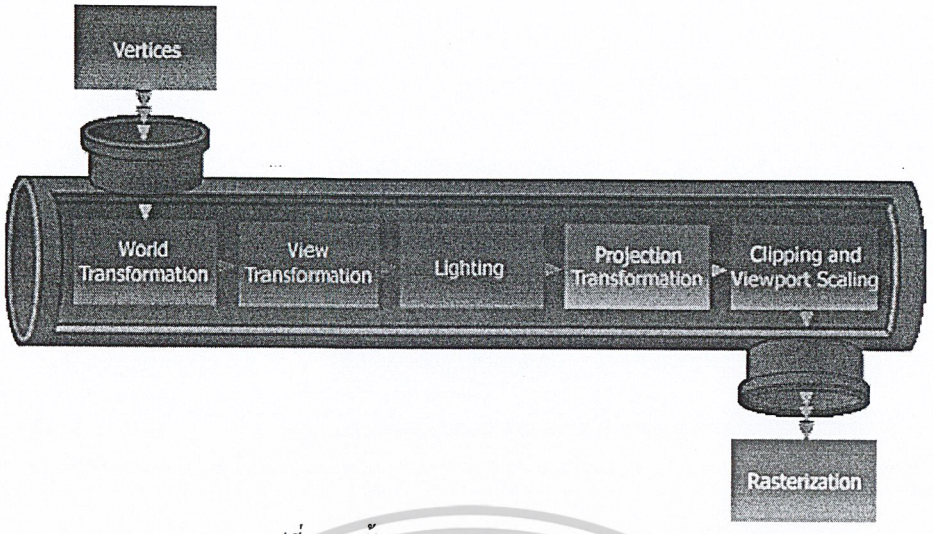
3.6.4 Projection Space

Screen Space คือพื้นที่สองมิติแบนๆ แบบภาพที่อยู่บนจอภาพ โดยพื้นที่นี้จะถูกแปลงมาจาก Projection Space เพื่อเตรียมเอาไปแสดงให้เห็นบนจอภาพ

3.6.5 Transformation and Lighting (T&L) และ Rasterization

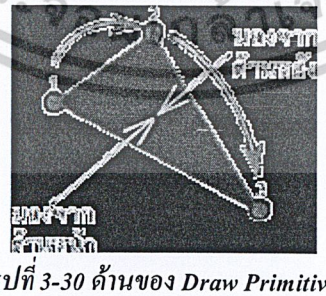
Transformation ก็คือขั้นตอนในส่วนของการคำนวณเพื่อแปลงหรือตำแหน่งของ Vertex ต่างๆ ที่ประกอบขึ้นเป็นโมเดลที่อยู่ใน Model Space ให้เป็นใน World Space (เรียกขั้นตอนนี้ว่า World Transformation) หลังจากนั้นก็ทำขั้นตอนการคำนวณเพื่อแปลง ตำแหน่งของ Vertex ต่างๆ บางส่วนของ World Space ให้ไปเป็นใน Camera Space (เรียกขั้นตอนนี้ว่า View Transformation) หลังจากนั้นจะเป็นการทำ Lighting โดยมันจะคำนวณเพื่อใส่สีต่างๆ ให้กับ Vertex เช่นสีจาก Texel ของเท็กซ์เจอร์ที่นำมา Map, สีของจุด Vertex นั้นๆ เอง หรือสีจากแหล่งกำเนิดแสงต่างๆ เป็นต้น

ขั้นต่อมาก็จะแปลงสิ่งที่อยู่ใน Camera Space ให้ไปอยู่ใน Projection Space (เรียกขั้นตอนนี้ว่า Projection Transformation) โดยที่ขั้นตอนนี้จะมีการคำนวณตำแหน่งและขนาดของโมเดลให้เพี้ยนไปจากเดิม เพื่อที่ว่าเมื่อถูกแปลงให้เห็นเป็นภาพ 2 มิติแล้ว จะได้เห็นเป็นแบบ Perspective คือสิ่งที่อยู่ใกล้ตา จะใหญ่กว่าสิ่งที่อยู่ไกลออกไป ซึ่งจะทำให้มองเหมือนว่า ภาพ 2 มิตินั้นดูมีความลึก ซึ่งการเขียนโปรแกรมเพื่อทำงานในส่วน Transformation ทั้ง 3 แบบที่กล่าวมานี้จะต้องใช้เมทริก (ซึ่งจะอธิบายเรื่องเมทริกในหัวข้อถัดไป)



รูปที่ 3-29 ขั้นตอนการทำ Transfromation

เมื่อทำงานในส่วน Transformation and Lighting เสร็จแล้ว ยังมีการทำงานอันหนึ่งแทรกก่อนจะไปถึงการทำงานในส่วน Rasterization นั่นคือ การตัดส่วนของโมเดลที่ถูกโมเดลอื่นบัง หรือตัดส่วนที่อยู่ด้านหลังออกไปด้วยแล้วก็ลดหรือขยายขนาดโมเดลต่างๆ ให้พอดีกับขนาดของ Resoluion ของหน้าจอที่ใช้ (เรียกขั้นตอนนี้ว่า Clipping and Viewport Scaling) ซึ่งขั้นตอนส่วนนี้จะช่วยลดการทำงานของขั้นตอนการ Rasterization คือไม่ต้องไปสนใจสิ่งที่มองไม่เห็นนั่นเอง จริงๆ แล้วการ Clipping และ Viewport Scaling นี่จะทำก่อน T&L ก็ได้ แต่ใน Direct3D มันมาทำตอนนี้ ถึงตอนนี้ เรื่องการเรียงตำแหน่ง Vertex ที่จะใช้การวาดรูปหรือ Draw Primitive จะเกี่ยวกับการตัดส่วนที่อยู่ด้านหลังที่ว่านี้(ด้านหลังของรูปสามเหลี่ยมจะไม่ถูก Render) ซึ่งโดย Default แล้ว Direct3D จะถือว่า ด้านหลังคือด้านที่ Vertex เรียงกันแบบทวนเข็มนาฬิกา (D3DRS_CULLMODE เป็น D3DCULL_CCW) ดังนั้น ต้องเรียง Vertex แบบตามเข็มนาฬิกา (ตามเข็มจะเรียงจากล่างขึ้นบนแล้วไปทางขวา หรือจากบนลงล่างขวา ส่วนทวนเข็มจะเรียงกลับกัน คือล่างขึ้นบนแล้วไปทางซ้าย หรือจากบนลงล่างซ้าย) จะสังเกตเห็นเหตุการณ์ที่ไม่ Render ด้านที่อยู่ข้างหลังได้โดยการรันโปรแกรมตัวอย่างแล้วใช้ปุ่ม S เลื่อนกล่องเข้ามาใกล้ๆ จนเข้าไปอยู่ในกล่อง จะเห็นว่ามันมืดสนิท



รูปที่ 3-30 ด้านของ Draw Primitive

ขั้นสุดท้ายจะเป็นการทำ Rasterization โดยตัวที่ทำหน้าที่นี้จะเรียกว่า Rasterizer จะแปลงสิ่งที่อยู่ใน Projection Space ให้ไปอยู่ใน Screen Space คือแปลงพิกัดของ Texel ต่างๆ ที่เป็นแบบ 3 มิติ ให้เป็น 2 มิติเมื่อเทียบกับขนาดของ Back Buffer หรือจอภาพ หลังจากนี้ก็จะได้ภาพปรากฏอยู่ใน Back Buffer

เพื่อที่จะได้สั่งให้ Direct3D Device Object จัดการนำข้อมูลใน Back Buffer นี้ไปสลับหรือ Copy เป็น Front Buffer เพื่อให้ภาพไปปรากฏให้เห็นบนจอภาพต่อไป

การคำนวณเรื่อง Transformation และ Lighting นี้จะใช้เวลานาน และกินแรง CPU มาก เพราะต้องใช้ส่วน FPU (Floating Point Unit) มาช่วยคำนวณเรื่อง Vertex ต่างๆ ซึ่งถ้าหากการ์ดเร่งความเร็วสามมิติใด Support การทำงานในส่วนนี้ก็จะทำให้ช่วยแบ่งเบาการทำงานของ CPU ไปได้เยอะ ทำให้ CPU มีเวลาไปคำนวณในส่วนอื่นๆ ผลก็คือโปรแกรมทำงานได้เร็วขึ้นมาก

3.6.6 World Transformation Matrix

เมทริกใน Direct3D จะเป็น Array ขนาด 4 x 4 ใช้สำหรับจัดการกับ Vector ต่างๆ ของ Vertex ซึ่งค่าที่อยู่ใน Array ของเมทริกนี้ไม่จำเป็นต้องรู้หรอก (ถ้าอยากรู้ก็ศึกษาเพิ่มเติมเอาเองนะ อิอิ) เพราะในไดเร็คเอ็กSDK มันมีฟังก์ชั่นง่ายๆ ในชุด D3DX Library ให้เรียกใช้ ซึ่งการคำนวณเมทริกจะต้องสร้าง Object ซึ่งเป็นตัวแปรแบบ Structure ขึ้นมาเสียก่อน (Structure ที่ใช้ก็คือ D3DXMATRIX)

World Transformation เมทริกเป็นเมทริกที่ใช้สำหรับนำโมเดลต่างๆ ไปไว้ใน World Space คือ ย้ายจาก Model Space ไปสู่ World Space ซึ่งขณะที่ทำการย้ายนั้น สามารถหมุน, ย่อ, ขยาย หรือวางโมเดลแต่ละอันไว้ที่ตำแหน่งใดใน World Space ก็ได้ โดยที่ไม่ต้องไปแตะต้องค่าของตำแหน่งดั้งเดิมของ Vertex ต่างๆ ที่กำหนดไว้ในโมเดลแต่ละอันเลย

กฎหรือหลักการแปลงจาก Model Space มา World Space ที่ดีก็คือ ถ้าอยากทำอะไรกับโมเดลแต่ละอันก็ทำไปก่อนแล้วเก็บค่าต่างๆ ไว้ในเมทริกธรรมดาที่สร้างขึ้นไว้ จากนั้นจึงค่อยรวมเมทริกแต่ละอันนั้น ให้เป็นเมทริกผลลัพธ์รวมอันเดียว แล้วจึงสั่งให้ทำกระบวนการ World Transformation ด้วยการใช้เจ้าเมทริกผลลัพธ์นี้เป็นหลัก ซึ่งทำได้โดยการเรียกใช้ เมธอด ชื่อ IDirect3DDevice8::SetTransform() โดยส่ง Paremeter ตัวแรกเป็นค่าคงที่ชื่อ D3DTS_WORLD และ Parameter ตัวสุดท้ายเป็น Address ของเมทริกผลลัพธ์ดังกล่าว

3.6.7 View Transformation Matrix

เป็นเมทริกที่ใช้สร้างกล้อง (Camera) ซึ่งมันจะจับภาพบางส่วนใน World Space หรืออีกในหนึ่งก็คือ แปลงหรือตำแหน่งของโมเดลต่างๆ ที่อยู่ใน World Space ตรงตำแหน่งที่กล้องจับภาพอยู่ มาเป็นของ Camera Space

และกฎที่สำคัญ รวมทั้งฟังก์ชั่นที่ใช้ในการหมุน, เคลื่อนย้าย หรือปรับขนาดของมุมมองของกล้องก็เหมือนกับที่ใช้กับ World Transformation เมทริกยกเว้นเรื่องที่ว่า ต้องมีการบอกให้ Direct3D รู้ด้วยว่า จะใช้ 3DSystem แบบ Left-Hand หรือ Right-Hand โดยใช้ฟังก์ชั่น D3DXMatrixLookAtLH() หรือ D3DXMatrixLookAtRH() ตามลำดับ ซึ่งจะใช้แบบมาตรฐานของ Direct3D คือ Left-Hand สำหรับ Parameter ที่ต้องส่งไปให้ฟังก์ชั่นนี้มีอยู่ 4 ตัว ตัวแรกเป็น Address ของเมทริกที่จะใช้เก็บผลลัพธ์ที่ได้ ตัวที่สองเป็น 3D Vector เพื่อบอกว่า ตำแหน่งของกล้องอยู่ที่ไหน

จากนั้นสร้าง View Transformationเมทริกโดยการเรียกใช้ เมธอด เหมือนที่ใช้กับ World Transformationเมทริกคือ IDirect3DDevice8::SetTransform() แต่ให้ใช้ค่าคงที่สำหรับใช้เป็น Parameter ตัวแรกว่า D3DTS_VIEW แทน แล้วส่ง Address ของเมทริกผลลัพธ์ที่ได้จาก D3DXMatrixLookAtLH() เป็น Parameter ตัวสุดท้าย

จะเห็นว่า เอาส่วนการสร้าง View Transformationเมทริกไปไว้ในฟังก์ชั่น fnInitGeometry() แทนที่จะเอาไปไว้ใน fnRender() ที่เป็นเช่นนี้ก็เพราะกล้องมันอยู่กับที่ตลอดเวลา ไม่เหมือนกับกรณี World Transformation ที่อาจมีการเปลี่ยนตำแหน่งของโมเดลในแต่ละเฟรม (นอกเสียจากว่าอยากเคลื่อนย้ายกล้องไปทั่วฉาก) ดังนั้นจึงไม่มีความจำเป็นที่จะต้องสั่งให้มัน Set ค่าเพื่อทำ View Transformation ทุกๆ เฟรม จึงทำครั้งแรกครั้งเดียวก่อนที่จะเข้า Loop การ Render

3.6.8 Projection Transformation Matrix

เป็นเมทริกที่ใช้สร้างมุมมองแบบ Perspective จากมุมมองของกล้องหรือของ Vertex ต่างๆ ที่อยู่ใน Camera Space โดยจะแปลงให้มาอยู่ใน Projection Space ซึ่งวิธีสร้างจะต้องใช้ฟังก์ชั่นแบบ Left-Hand หรือ Right-Hand คือ D3DXMatrixPerspectiveFovLH() กับ D3DXMatrixPerspectiveFovRH() ตามลำดับ แต่จะใช้แบบ Left-Hand โดย Parameter ตัวแรกที่ส่งให้ฟังก์ชั่นนี้ก็คือ Address ของเมทริกที่จะใช้เก็บผลลัพธ์ที่ได้ ส่วน Parameter ตัวที่สองจะเป็นมุมแบบเรเดียนซึ่งหมายถึงระยะกางของมุมมองที่จะทำ Perspective หรือที่เรียกว่า Field of View (FOV) ถ้ามุมกว้างมาก จะหมายถึงการดึงภาพกว้างมาก มาบีบให้แสดงในพื้นที่ของหน้าจอ ผลก็คือเกิด Perspective มากเมื่อมองโมเดลแบบใกล้ๆ (ภาพจะเพี้ยนมาก ทำให้เวียนหัวและคลื่นไส้อยากจะอ้วก) ปกติมักจะใช้ 45 องศา หรือ p / 4 เรเดียน (สามารถใช้ค่าคงที่ D3DX_PI แทนค่า p ได้) ในเกม Half-Life หรือ Half-Life: Counter-Strike จะใช้ถึง 72 องศา หรือ p / 2.5 เรเดียน

บทที่ 4
ระบบเครือข่ายในเกม

4.1 Winsock คืออะไร

winsock คือ API ที่ใช้ในการสร้างการเชื่อมต่อเพื่อรับส่งข้อมูลผ่านระบบเครือข่าย ที่เรียกว่า winsockเพราะเป็นการปรับปรุงของ API Socket จากระบบ UNIX ให้เป็น API ที่ทำงานบนระบบปฏิบัติการวินโดว์

socket ก็คือ การเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์สองเครื่องทั้งระหว่างอินเตอร์เนตหรือระหว่างแลนด้วยกันเอง การเชื่อมต่อของ socket จะเป็นแบบสองทาง คือสามารถรับและส่งข้อมูลได้โดยสร้าง socket เพียงครั้งเดียว การเชื่อมต่อผ่าน socket จะอาศัย IP Address เป็นตัวระบุเป้าหมายของการส่งข้อมูล แต่เนื่องจากเครื่องคอมพิวเตอร์เครื่องหนึ่งจำเป็นต้องมีการเชื่อมต่อมากกว่าหนึ่งชนิดสำหรับการทำงานของโปรแกรมต่างๆ ดังนั้นจึงต้องใช้พอร์ตเพื่อระบุ socket ร่วมกันกับ IP Address สำหรับแต่ละ socket เช่น การเชื่อมต่อของ Internet Explorer จะต้องใช้ IP ของเครื่องเวปเซิร์ฟเวอร์และพอร์ต 80 ซึ่งเป็นพอร์ตมาตรฐานสำหรับการเชื่อมต่อเพื่อเรียกใช้บริการเวปเพจ การสร้าง socket สำหรับการเชื่อมต่อนี้จึงต้องอาศัยข้อมูลสองอย่างนี้เพื่อให้โปรแกรม IE สามารถรับส่งข้อมูลได้อย่างถูกต้อง แม้ว่าขณะนั้นในเครื่องจะเปิดบริการอื่นๆ เช่น ดาวน์โหลดจาก FTP อยู่ก็ตาม โดยหมายเลขพอร์ตตั้งแต่ 1 – 1000 จะสงวนไว้สำหรับการใช้งานในโปรโตคอลมาตรฐาน

การทำงานของwinsockจะแบ่งเป็น 5 ขั้นตอน คือ

1. Create เป็นการสร้าง socket ขึ้นมาโดยอาศัย IP และหมายเลขพอร์ต
2. Connect สั่งให้ socket ที่สร้างขึ้นเริ่มทำงาน
3. Listen สั่งให้เครื่องหยุดการทำงานและรอรับการเชื่อมต่อจากคอมพิวเตอร์อีกเครื่องหนึ่ง โปรแกรมที่เรียกคำสั่งนี้และหยุดรออยู่จะเรียกว่า Daemon
4. Disconnect สั่งให้ socket หยุดทำงาน
5. Close ปิดการติดต่อระหว่างพอร์ตที่ระบุใน socket จัดการลบ socket และคืนทรัพยากรสู่ระบบ

winsockจะทำหน้าที่เป็นตัวกลางที่ติดต่อระหว่างโปรแกรมอื่นๆกับมาตรฐาน TCP/IP ในระบบปฏิบัติการแต่ละระบบก็มีการจัดการ socket และwinsockที่ไม่เหมือนกัน ขณะที่วินโดว์ 98มีwinsockมาให้เลยตั้งแต่ติดตั้ง ระบบปฏิบัติการวินโดว์รุ่นหลังๆกลับต้องลงwinsockเองในภายหลัง ในขณะที่ระบบ UNIX ไม่ต้องการการติดตั้ง API socket เพิ่มเติมเพราะถูกรวมมากับระบบปฏิบัติการอยู่แล้ว

การจะใช้ฟังค์ชั่นการทำงานของwinsockได้เวอร์ชั่นของwinsockจะต้องรองรับมาตรฐาน TCP/IP ของระบบเครือข่ายนั้นๆ รวมถึงเวอร์ชั่นที่ถูกต้องถ้าหากต้องการนำไปติดตั้งในระบบปฏิบัติการอื่นๆนอกจากวินโดว์

4.2 Socket Modes

winsock จะมีทำงานกับ I/O ใน 2 โหมด ดังนี้ โหมด Blocking ในโหมดนี้ winsock จะจัดการกับ I/ Oโดยเมื่อมีการ เรียก send หรือ recv จะรอจนกระทั่งการดำเนินงานเสร็จก่อนที่จะส่งค่าบอกว่าทำเสร็จกลับมายังโปรแกรมโหมดNon-blocking ในโหมดนี้ Winsock จะส่งค่ากลับไปทันที โปรแกรมที่รันบน Window ce และ Window 95 (กับ Winsock 1) ที่สนันสนุนรูปแบบ I/O ต้องการให้เลือกใช้ให้เหมาะสม

Blocking Mode

Blocking socket ทั้งหมดจะมีความเกี่ยวข้องกัน เนื่องจาก Winsock API ที่เรียก Blocking socket สามารถที่จะป้องกันไม่ให้ใช้งาน (Block) นานช่วงเวลาหนึ่ง แต่เมื่อใช้โหมดนี้จะมีผลเสีย คือ การสื่อสารมากกว่า 1 การสื่อสารจะทำได้ยาก แต่ก็มีวิธีจัดการ คือ โปรแกรมจะต้องจัดการให้มีการสื่อสาร 1 การสื่อสารรวมทั้งการรับหรือส่งข้อมูลต่อ 1 Thread แต่การทำเช่นนี้จะไม่สามารถที่จะจัดการกับการติดต่อสื่อสารจำนวนมาก ซึ่งต้องใช้Socket จำนวนมากได้

Non-blocking Mode

เป็นอีกโหมดหนึ่งของ Winsock Non-blocking socket จะไม่ต้องรอให้มีการรับหรือส่งข้อมูลก่อนแล้วส่งค่ากลับมาบอก ซึ่งมีข้อดีมากกว่าแบบ Blocking Mode เมื่อ Socket ถูกเรียกใช้งานในโหมดนี้ Winsock จะมีการส่งหรือรับข้อมูลหรือการติดต่อแล้วจะจัดการส่งค่ากลับมาในทันที

ในแต่ละโหมดทั้ง Blocking และ non-blocking มีทั้งข้อดีและข้อเสียแต่ต่างกันไป ในโหมด Blocking จะมีการเรียกใช้งานที่ง่ายกว่า แต่ยากที่จะจัดการกับการเชื่อมต่อด้วยหลายๆ socket หรือเมื่อมีข้อมูลที่ส่งหรือรับจำนวนมากในเวลาเดียวกัน แต่ในโหมด non-blocking จะมีการใช้งานที่ยากกว่าเพราะจะเขียนโปรแกรมขึ้นมาจัดการตรวจสอบว่ามีข้อมูลเข้ามากับทุกๆ socket จึงได้มี Socket I/O Model ช่วยในการจัดการการติดต่อระหว่าง socket ในเวลาเดียวกัน

ข้อดีของการเชื่อมต่อแบบ Non-Blocking จะเหมาะสำหรับการทำเซิร์ฟเวอร์ที่ต้องทำงานกับไคลเอนท์จำนวนมากในเวลาเดียวกัน เช่น เซิร์ฟเวอร์ของเกมออนไลน์ แต่เซิร์ฟเวอร์แบบ Non-Blocking ก็มีข้อเสียอยู่บ้าง เนื่องจากต้องใช้ทรัพยากรของระบบส่วนหนึ่งไปกับการทำงานของ Daemon ที่รอรับการร้องขอ และถ้าหากมีไคลเอนท์เชื่อมต่อเข้ามาเป็นจำนวนมากอย่างต่อเนื่องก็จะทำให้การทำงานในส่วนนี้มีมากขึ้น ทำให้เซิร์ฟเวอร์ต้องรับภาระหนักมากขึ้นเช่นกัน

นอกจากการแบ่งเป็น Blocking และ Non-Blocking แล้ว เซิร์ฟเวอร์ยังแบ่งได้อีกสองประเภท เป็น Synchronous และ Asynchronous โดย Synchronous จะทำงานคล้ายกับ Blocking และ Asynchronous จะทำงานคล้ายกับ Non-Blocking โดยจะแตกต่างกันในลักษณะการเรียกใช้งานคำสั่ง Listen ของ socket Asynchronous และ Synchronous จะใช้การทำงานแจ้งเตือน(Notify)โดยส่งเมสเสจมาให้กับโปรแกรมของเซิร์ฟเวอร์โดยที่ผู้เขียนไม่ต้องลงไปควบคุมคำสั่ง Listen ด้วยตัวเอง ถึงแม้การทำงานจะคล้ายกันแต่การเรียกใช้งานคำสั่ง Listen จะขึ้นอยู่กับการทำงานของโปรแกรมและแพลตฟอร์มที่เรียกใช้งานด้วย

4.3 Socket I/O Models

ใน Socket I/O Modelsจะมีอยู่ 6 แบบ ดังนี้

4.3.1 The blocking Model

นักพัฒนาโปรแกรมส่วนใหญ่จะเริ่มต้นด้วยรูปแบบนี้ เนื่องจากเป็นวิธีที่ง่ายและตรงไปตรงมา โปรแกรมที่ใช้รูปแบบนี้ ส่วนใหญ่จะใช้ Thread 1 หรือ 2 ตัวในการจัดการ ต่อ 1 การเชื่อมต่อ

4.3.2 The select Model

The select Model เป็นอีกรูปแบบหนึ่งที่ใช้กันมาก โดยการใช้ select model ก็เพราะว่ามีการใช้ฟังก์ชัน select ในการจัดการ I/O ปกติจะใช้ในคอมพิวเตอร์ที่ใช้งาน Unix ในรูปแบบนี้จะรวมเข้าไปใน Winsock 1.1 ซึ่งให้โปรแกรมสามารถใช้งานโดยปราศจากการ block เพื่อจัดการกับ socket จำนวนมากได้

ในฟังก์ชัน select นั้นสามาถที่จะตรวจสอบได้ว่ามีข้อมูลเข้ามาที่ socket หรือ สามารถที่จะส่งข้อมูลได้ เหตุผลที่ใช้ฟังก์ชันนี้ป้องกันโปรแกรมจากการ Block หรือรอที่จะใช้งาน I/O มีรูปแบบดังนี้

```
int select(
  int nfds,
  fd_set FAR * readfds,
  fd_set FAR * writefds,
  fd_set FAR * exceptfds,
  const struct timeval FAR * timeout
);
```

ข้อดีของการใช้งาน select คือสามารถที่จะจัดการการเชื่อมต่อที่ซับซ้อนของหลายๆ socket ได้ใน 1 Thread ซึ่งจะป้องกันการสร้าง Thread ออกมามากอย่างเช่น ใน blocking socket แต่ข้อเสียคือปกติจะรับการเชื่อมต่อผ่าน FD_SET ซึ่งมีขนาดมากที่สุดเป็น 1024 เท่านั้น และเรายังจะต้องมีการตรวจสอบทั้ง 1024 socket ถ้ามีการเชื่อมต่อทั้งหมด

4.3.3 The WSAAsyncSelect Model

Winsock มีรูปแบบการจัดการเกี่ยวกับ I/O ผ่าน Window message โดยการเรียกผ่านฟังก์ชัน WSAAsyncSelect หลังจากสร้าง socket ซึ่งในการใช้งานจะต้องมีการสร้างหน้าต่าง window ขึ้นมาก่อน และใช้ window procedure ในการจัดการ นอกจากการสร้างหน้าต่างwindow แล้ว ยังสามารถที่จะสร้าง Dialog ขึ้นมาแทนได้ เพราะ Dialog ก็เป็น window

รูปแบบนี้ มีข้อดี คือ สามารถรับการเชื่อมต่อได้หลายๆการเชื่อมต่อในทันทีโดยปราศจากการสูญเสีย (Overhead) ไม่เหมือนกับแบบ select ซึ่งจะต้องมีการตรวจสอบเองตลอดเวลา แต่ก็มีข้อเสียคือต้องใช้ window ถ้าในโปรแกรมไม่ได้มีการสร้าง window ก็จะทำให้ใช้งานไม่ได้ และนอกจากนี้ในการใช้ window จัดการกับการเชื่อมต่อทั้งหมดจะทำให้เกิดปัญหาคอขวด(bottleneck)

4.3.4The WSAEventSelect Model

นอกจาก winsock จะมี I/O Model แบบWSAAsyncSelect แล้วซึ่งจัดการเมื่อมีเหตุการณ์เข้ามาโดยผ่าน window ยังมีอีกรูปแบบหนึ่งคือ the WSAEventSelect Model ในรูปแบบนี้ ไม่ต้องมีการใช้ window แต่จะมีการสร้าง network event ขึ้นมาแทน โดยมีรูปแบบดังนี้

int WSAEventSelect(SOCKET s, WSAEVENT hEventObject, long lNetworkEvents);

ในรูปแบบนี้มีข้อดีอยู่หลายอย่าง คือ ง่ายและไม่ต้องการ window แต่ก็มีข้อจำกัดของการรอเหตุการณ์ 64 อย่างในเวลาเดียวกัน ซึ่งจำเป็นต้องมี Thread pool จัดการกับหลายๆsocket

4.3.5The Overlapped Model

ใน winsock ได้เสนอวิธีจัดการระบบให้มีประสิทธิภาพมากขึ้น คือ The Overlapped Model รูปแบบนี้จะให้โปรแกรมสามารถร้องขอผ่าน I/O ได้หลายๆการร้องขอในเวลาเดียวกัน โดยการใช้ Overlapped data รูปแบบนี้จะใช้ได้บน window ทุกplatform ยกเว้นบน window ce รูปแบบนี้จะจัดการกับ I/O โดยใช้รูปแบบฟังก์ชัน ReadFile และ WriteFile แต่ใน winsock 2 นี้ Overlapped I/O ได้รวมเข้าไปในฟังก์ชันของ Winsock เช่น WSASend และ WSARecv

ในการใช้ Overlapped I/O กับ socket จะต้องสร้าง socket ที่ตั้งค่า Overlapped flag หลังจากสร้าง socket ขึ้นมาแล้วต้องทำการ bind การทำงานของ Overlapped I/O เริ่มต้นโดยการเรียกฟังก์ชัน ดังนี้

- WSASend
- WSASendTo
- WSARecv
- WSARecvFrom
- WSAIoctl
- WSARecvMsg
- AcceptEx
- ConnectEx
- TransmitFile
- TransmitPackets
- DisconnectEx
- WSANSPIoctl

รูปแบบนี้ จะทำให้มีการจัดการ socket I/O ได้อย่างมีประสิทธิภาพ เนื่องจากโปรแกรมจะต้องส่ง buffer ที่ต้องการส่งหรือรับข้อมูลไปให้ระบบจัดการ เช่นถ้าโปรแกรมส่ง buffer ขนาด 10 KB เพื่อไปรับข้อมูลที่เข้ามายัง socket ระบบก็จะทำการcopy ข้อมูลเข้ามาใน bufferให้ โดยในรูปแบบก่อนๆ เมื่อข้อมูลที่มาถึง จะรับข้อมูลไปเก็บใน buffer ของ socket เท่านั้น

ส่วนข้อเสียของรูปแบบนี้อยู่ที่ event อีก คือมีข้อจำกัดรับ ได้เพียง 64 event ในเวลาหนึ่งๆ

4.3.6 The Completion Port Model

รูปแบบนี้มีความซับซ้อนมากเมื่อเปรียบเทียบกับรูปแบบก่อนๆ เพราะจะมีเพิ่ม socket เข้าไปใน completion port แต่รูปแบบนี้จะมีการจัดการได้อย่างมีประสิทธิภาพมากที่สุด เมื่อมีการจัดการกับหลายๆ socket ในเวลาเดียวกัน รูปแบบนี้จะใช้ได้แต่ใน Windows NT, Windows 2000 และ Windows XP แต่ถึงอย่างนั้น Completion Port Model ก็เป็นวิธีที่ดีที่สุดในกับจัดการกับ socket จำนวนมากๆได้ดี

4.4 การวิเคราะ เซิร์ฟเวอร์

เซิร์ฟเวอร์ สามารถแบ่งส่วนวิเคราะห์ออกเป็น 2 แบบใหญ่ ดังนี้

1. High Throughput

ใน FTP server เป็นตัวอย่างหนึ่งของ เซิร์ฟเวอร์ ที่ให้อัตราการส่งข้อมูล (Throughput) ที่สุง ในกรณีนี้ เซิร์ฟเวอร์ จะต้องประมวลผลแต่ละการเชื่อมต่อให้มีเวลาน้อยที่สุด โดย เซิร์ฟเวอร์ ต้องจำกัดจำนวนของการเชื่อมต่อในเวลาเดียวกันให้มีจำนวนน้อยๆเพราะการที่มีการเชื่อมต่อในเวลาเดียวกันมากในเวลาเดียวกันทำให้มี Throughput ของแต่ละการเชื่อมต่อที่ต่ำลง

เซิร์ฟเวอร์ ควรที่จะจัดการเกี่ยวกับรับหรือส่งข้อมูลให้ดีเพื่อให้ได้ Throughput ที่มาก เนื่องจากแต่ละ Overlapped I/O ต้องการหน่วยความจำที่กำหนดเอาไว้ตายตัวแล้วสำหรับการสร้างการเชื่อมต่อ นอกจากนี้ เซิร์ฟเวอร์ ควรที่จะตรวจดูการทำงานของของแต่ละการเชื่อมต่อ เพื่อป้องกันการโจมตีจากไคลเอนท์ อีกด้วย

2. Maximizing Connections

การเชื่อมต่อจาก ไคลเอนท์ หลายๆคนในเวลาเดียวกันนั้น เป็นสิ่งที่ยากต่อการออกแบบและจัดการ เซิร์ฟเวอร์ ไม่สามารถที่จะจัดการรับส่งให้กับแต่ละการเชื่อมต่อได้โดยตรง เพราะจะทำให้มีการใช้หน่วยความจำเป็นจำนวนมาก ดังนั้นเซิร์ฟเวอร์ จึงต้องจัด Buffer ที่ใช้ในการรับข้อมูลเมื่อมีการรับข้อมูลแล้ว เซิร์ฟเวอร์ จะต้องทำการ nonblocking receive จนกว่าจะค่าของ WSAEWOUDLBLOCK จะส่งมาซึ่งจะให้ เซิร์ฟเวอร์ รับข้อมูลทั้งหมดได้ทันที

I/O Model	*Attempted /Connected*	*Memory Used (KB)*	*Non-Paged Pool*	*CPU Usage*	*Thread*	*Throughput (Send/ Receive BPS)*
Blocking	7000/ 1008	25,632	36,121	10–60%	2016	2,198,148/ 2,198,148
	12,000/ 1008	25,408	36,352	5– 40%	2016	404,227/ 402,227
Non- blocking	7000/ 4011	4208	135,123	95–100%*	1	0/0
	12,000/ 5779	5224	156,260	95–100%*	1	0/0
WSA- Async Select	7000/ 1956	3640	38,246	75–85%	3	1,610,204/ 1,637,819
	12,000/ 4077	4884	42,992	90–100%	3	652,902/ 652,902
WSA- Event Select	7000/ 6999	10,502	36,402	65–85%	113	4,921,350/ 5,186,297
	12,000/ 11,080	19,214	39,040	50–60%	192	3,217,493/ 3,217,493
	46,000/ 45,933	37,392	121,624	80–90%	791	3,851,059/ 3,851,059
Over- lapped (events)	7000/ 5558	21,844	34,944	65–85%	66	5,024,723/ 4,095,644
	12,000/12,000	60,576	48,060	35–45%	195	1,803,878/ 1,803,878
	49,000/48,997	241,208	155,480	85–95%	792	3,865,152/ 3,834,511
Over- lapped (completion port)	7000/ 7000	36,160	31,128	40–50%	2	6,282,473/ 3,893,507
	12,000/12,000	59,256	38,862	40–50%	2	5,027,914/ 5,027,095
	50,000/49,997	242,272	148,192	55–65%	2	4,326,946/ 4,326,496

ตารางที่ 4-1 เปรียบเทียบประสิทธิภาพของ I/O Method

4.5 สถาปัตยกรรมของการเชื่อมต่อ

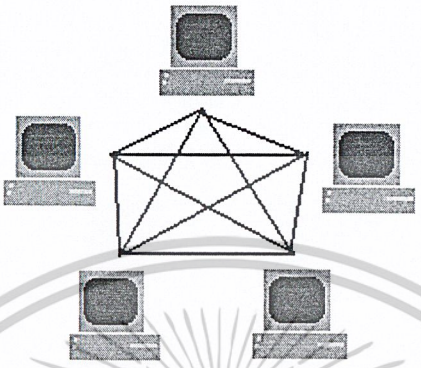
สิ่งสำคัญสำหรับอีกอย่างหนึ่งก็คือการเลือกสถาปัตยกรรมที่จะใช้ในการเชื่อมต่อ สภาปัตยกรรมส่วนใหญ่แบ่งออกเป็นสองชนิดหลักๆก็คือ Client-Client และ Client-Server

4.5.1 Client-Client

เป็นการสร้างการเชื่อมต่อระหว่างคอมพิวเตอร์สองเครื่อง ซึ่งสถาปัตยกรรมแบบนี้จะเรียกอีกแบบว่า peer-to-peer การเชื่อมต่อแบบนี้เครื่องคอมพิวเตอร์ที่อยู่ทั้งสองฝั่งจะต้องรู้ IP ของฟอีกฝั่งหนึ่งจึงจะสามารถสร้างการเชื่อมต่อขึ้นมาได้ ซึ่งผู้ที่จะสร้างการเชื่อมต่ออาจจะแลก IP ผ่านทาง Mail หรือโปรแกรมแชทอื่นๆ หรือสร้างเซิร์ฟเวอร์ที่ทำหน้าที่เก็บ IP ของแต่ละคนไว้สำหรับแจกจ่ายให้เครื่องอื่นๆ สำหรับเกมที่ใช้การเชื่อมต่อแบบนี้จะใช้ CPU ในการประมวลผลสูงมาก เนื่องจากแต่ละเครื่องจะต้อง

ประมวลผลการทำงานทั้งหมดของเกมและต้องจัดการข้อมูลที่รับส่งไปมาให้ตรงกันในช่วงเวลาที่ใกล้เคียงกัน ซึ่งการประมวลผลบางส่วนอาจจะทำเหมือนๆกันจึงเป็นการทำงานที่ซ้ำซ้อนโดยไม่จำเป็น

การเชื่อมแต่ในแบบ Client-to-Client ไม่จำเป็นว่าต้องสร้างขึ้นระหว่างเครื่องสองเครื่องเท่านั้น แต่ถ้าจำนวนเครื่องมากขึ้น การเชื่อมต่อก็ต้องมากขึ้นด้วย ในระบบของเกมออนไลน์ที่มีผู้เล่นจำนวนมาก การเชื่อมต่อแบบนี้จึงไม่สะดวกและทำได้ยาก



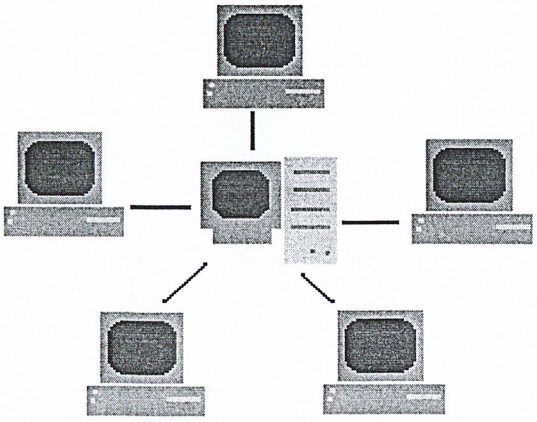
รูปที่ 4-1 การเชื่อมต่อแบบ Client-to-client

จากรูปจะเห็นได้ว่าถ้าหากมีจำนวนเครื่องจะมีผลต่อจำนวนของการเชื่อมต่อที่ต้องสร้างขึ้น ถ้าหากให้จำนวนเครื่องเท่ากับ N จำนวนการเชื่อมต่อจะเท่ากับ N*(N-1) ซึ่งถ้าหากมีเครื่องที่ต้องการเชื่อมต่อมากกว่าสิบเครื่องขึ้นไปจะต้องสร้างการเชื่อมต่อขึ้นมาเป็นจำนวนมากทำให้สิ้นเปลืองทรัพยากรของระบบ

4.5.2 Client-Server

วิธีนี้จะเป็นวิธีที่เหมาะสมกับการเชื่อมต่อจำนวนมาก โดยยกภาระในการคำนวณและประมวลผลไปให้กับเครื่องที่มีกำลังกำลังการประมวลผลและมีการเชื่อมต่อกับระบบเครือข่ายที่มีประสิทธิภาพ โดยจะเรียกเครื่องนี้ว่าเซิร์ฟเวอร์ แล้วให้เครื่องอื่นๆสร้างการเชื่อมต่อเข้ามาที่เซิร์ฟเวอร์เพียงเครื่องเดียว เซิร์ฟเวอร์จะทำหน้าที่กระจายข้อมูลที่แต่ละเครื่องส่งมาไปยังเครื่องอื่นๆที่เหลือโดยไม่ต้องสร้างการเชื่อมต่อเพิ่มเติม

ข้อแตกต่างจากการเชื่อมแต่แบบ Client-Client อีกอย่างหนึ่งก็คือจำนวนของ socket เนื่องจากการเชื่อมต่อจะเริ่มจากไคลเอนท์เป็นผู้ร้องขอการเชื่อมต่อเข้ามาเท่านั้น ดังนั้นจำนวนการเชื่อมต่อจึงเท่ากับจำนวนเครื่องที่ใช้งานอยู่ในขณะนั้น การรับส่งข้อมูลจะอาศัยเซิร์ฟเวอร์กระจายข้อมูลต่างๆออกไปยัง socket ของไคลเอนท์แต่ละตัว ถ้าหากมีไคลเอนท์จำนวนมาก จากไคลเอนท์จำนวนเท่ากันจะเห็นได้ว่าจำนวนของการเชื่อมต่อจะน้อยกว่าแบบ Client-Client มาก

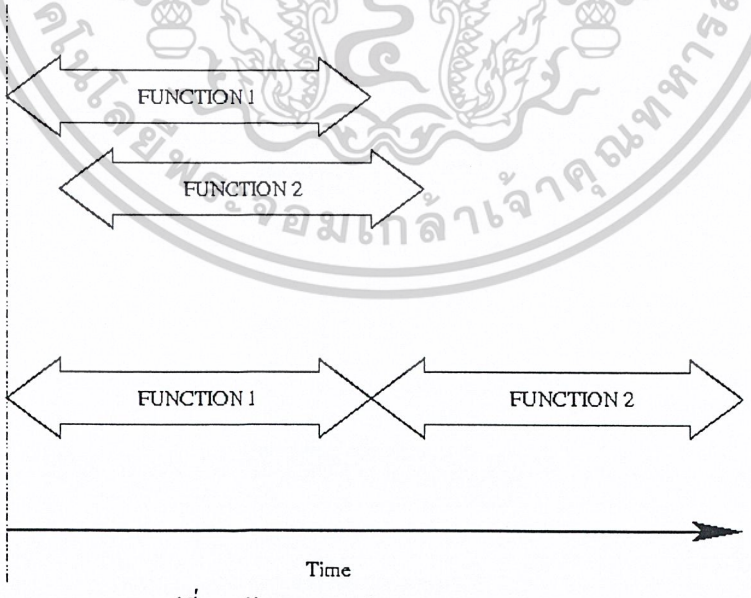


รูปที่ 4-2 การเชื่อมต่อแบบ Client-Server

แต่อย่างไรก็ตาม ความเร็วที่ใช้ในการเชื่อมต่อก็ยังคงขึ้นกับระยะทาง ความยาวของแพคเก็จที่ใช้ และในการสร้างเกมยังขึ้นกับลักษณะการทำงานของเกมอีกด้วย ดังนั้นการแบ่งงานระหว่างไคลเอนท์และเซิร์ฟเวอร์จึงเป็นสิ่งจำเป็น เซิร์ฟเวอร์ควรจะทำงานที่ซ้ำซ้อนกันในไคลเอนท์แต่ละเครื่อง โดยปล่อยให้ไคลเอนท์เป็นตัวรับส่งข้อมูลกับผู้ใช้และทำงานที่ต้องการตอบสนองอย่างรวดเร็ว

4.6 Multithreading

การสร้างเทรดคือการกำหนดให้ CPU ทำงานสองอย่างพร้อมๆกัน อันที่จริงแล้ว CPU จะแบ่งการทำงานออกเป็นส่วนย่อยๆลงไปในแต่ละโปรแกรม โดยเรียกส่วนย่อยๆว่าเทรด แต่ละเทรดจะมีทรัพยากรที่ CPU แบ่งให้เป็นของตัวส่วนหนึ่ง เพื่อให้การทำงานในแต่ละเทรดสามารถดำเนินไปได้โดยไม่ต้องขึ้นกับเทรดอื่นๆ



รูปที่ 4-3 ลักษณะการทำงานแบบ Multithread

FUNCTION1 และ FUNCTION2 ด้านบนเป็นการทำงานในลักษณะที่เรียกว่า มัลติเทรดโดยแต่ละฟังก์ชันทำงานในคนล่ะเทรดและทำงานขนานกันไป ในขณะที่ FUNCTION1 และ FUNCTION2 ด้านล่าง เป็นการทำงานของโปรแกรมทั่วไปซึ่งมีการทำงานต่อกันไม่สามารถทำงานขนานกันไปได้ จากรูปจะเห็นว่าการทำงานแบบมัลติเทรดสามารถทำงานโดยใช้เวลาที่สั้นกว่าได้ อย่างไรก็ตามสำหรับระบบที่มี CPU เพียงตัวเดียวอาจจะไม่เห็นประโยชน์ของการเขียนโปรแกรมแบบมัลติเทรดนี้มากนัก นอกจากลักษณะงานที่ต้องการนั้นจะมีหลายๆ งานที่สามารถทำงานขนานกันไปได้และในแต่ล่ะงานที่ว่านี้มีการรอค่าหรือรอสัญญาณจากภายนอกอย่างอื่นก่อนเริ่มทำงานต่อไป ซึ่งอาจทำให้ CPU ว่างและสามารถมาทำงานอื่นก่อนได้เป็นต้น

ในการสร้างเซิร์ฟเวอร์แบบ Non-Blocking จะต้องมีการสร้างเทรด(Thread) ขึ้นเพื่อจัดการกับ socket อย่างน้อยสอง socket พร้อมกัน คือ Listening Socket ที่ทำหน้าที่รอรับการร้องขอการเชื่อมต่อจากไคลเอนท์ กับ Client Socket หรือ Working Socket ที่เป็นการติดต่อจริงๆระหว่างเซิร์ฟเวอร์กับไคลเอนท์

และเพราะแต่ละเทรดจะจัดการกับเมมโมรีในส่วนของตัวเอง เทรดจึงมีปัญหาบางอย่างกับการทำงานของเซิร์ฟเวอร์ เช่น การเข้าถึงฐานข้อมูล แต่ละเทรดจะต้องมีการควบคุมให้เข้าถึงฐานข้อมูลได้ทีละเทรดเท่านั้น เพื่อให้ไม่เกิดความผิดพลาดจากการเขียนและอ่านข้อมูลผิดลำดับ ซึ่งจะใช้วิธีของ Critical Section ที่จะจำกัดให้เทรดที่เข้าไปทำงานมีเพียงเทรดเดียว

4.7 Connection Orient และ Connection less

ระบบของ OSI โมเดลในส่วนของการจัดเตรียมข้อมูลเกี่ยวกับการเชื่อมต่อ ชั้นเนตเวิร์คเลเยอร์จะทำหน้าที่เรียกใช้หรือกำหนดช่องทางในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet ,Token Ring หรือ FDDI รวมถึงการควบคุมลำดับและอัตราการรับส่งข้อมูล สถานที่ที่จะส่งข้อมูลไป (Address) ทั้งนี้ชั้นดาต้าลิงค์เลเยอร์ เป็นชั้นแรกที่จัดการแปลงข้อมูลจากบิตให้อยู่ในรูปของแพคเกจโดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้องในกรณีส่งข้อมูลออกไป หรือในกรณีอ่านข้อมูลเข้ามาตรวจสอบส่วนเช็คซัมเพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับแพคเกจข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งานต่อและบอกไปยังต้นทางให้ส่งมาใหม่

การทำงานในชั้นดาต้าลิงค์เลเยอร์ จะมีการทำงานสองแบบ แบบแรกคือ Connection Orient ซึ่งการเชื่อมต่อแบบนี้จะต้องมีการสร้างคอนเนคชั่นก่อนที่จะส่งข้อมูลโดยเมื่อสร้างคอนเนคชั่นขึ้นแล้วจะส่งข้อมูลเท่าใดก็ได้จนกว่าจะปิดการเชื่อมต่อหรือเกิดปัญหาขึ้นกับระบบเครือข่าย ส่วนการเชื่อมต่อแบบ Connectionless คือการส่งข้อมูลเป็นแพคเกจไปยังเป้าหมายโดยที่ไม่ต้องสร้างคอนเนคชั่น แต่การส่งข้อมูลแบบนี้จะมีปัญหาในเรื่องของความปลอดภัยและการสูญหายของข้อมูล

บทที่ 5

MMORPG

5.1 MMORPG คืออะไร

MMORPG ย่อมาจาก Massive Multiplayer Online Role Playing Game หมายถึงเกมจำลองบทบาทที่เล่นผ่านระบบเครือข่ายโดยสามารถรองรับผู้เล่นจำนวนมากได้ MMORPG พัฒนาขึ้นมาจากเกม RPG ทั่วไปที่ให้ผู้เล่นจำลองตัวเองเข้าไปเล่นเป็นบทบาทสมมุติในเกม มีการจัดการเงื่อนไขต่างของแต่ละคนหรือเป็นกลุ่มเพื่อดำเนินเนื้อเรื่องในเกมต่อไป ภายในเกมจะเน้นที่การพัฒนาตัวละครเหมือนกับเกม RPG ทั่วไป แต่จะมีส่วนของการสร้างความสัมพันธ์กับผู้เล่นคนอื่นๆ ให้ความร่วมมือ จนกลายเป็นสังคมจำลองในระบบออนไลน์ขนาดใหญ่ที่มีผู้เล่นหลายร้อยคน

เนื่องจากปัจจุบันระบบเครือข่ายได้พัฒนาไปมาก อินเตอร์เนตความเร็วสูงก็มีประสิทธิภาพมากขึ้นขณะที่ราคาถูกลง การใช้งานระบบเครือข่ายเพื่อความบันเทิงจึงเป็นไปได้ง่ายขึ้น และเกมออนไลน์จะมีความเปลี่ยนแปลงในตัวเกมสูงทำให้เกิดความหลากหลายและค่าในการเล่นซ้ำมีสูง ทำให้เกิดการลงทุนและขยายตัวในตลาดเกมออนไลน์สูงขึ้นอย่างมากในช่วงสองปีที่ผ่านมา มีบริษัทผู้ผลิตหลายแห่งส่งซอร์ฟแวร์เกมออนไลน์เข้ามาในตลาดอย่างต่อเนื่อง รวมถึงจำนวนของร้านอินเตอร์เนตคาเฟ่ที่เพิ่มขึ้นอย่างรวดเร็ว แสดงให้เห็นถึงการเติบโตของเกมออนไลน์ที่มีขึ้นอย่างก้าวกระโดดภายในเวลาไม่นาน

5.2 ลักษณะเฉพาะของ MMORPG

นอกจากการออนไลน์และผู้เล่นจำนวนมากที่เป็นเอกลักษณ์ของเกม MMORPG แล้ว ข้อแตกต่างระหว่าง MMORPG และ เกม RPG ที่เป็นเกมออฟไลน์ธรรมดาก็คือเรื่องของสังคมในเกม (Community) ซึ่งจะมีส่วนให้ผู้เล่นย้อนกลับมาเล่นอยู่เรื่อยๆ ซึ่งเป็นเหตุผลหลักที่ทำให้เกมแบบ MMORPG สามารถทำค่าทางธุรกิจได้มหาศาล ลักษณะเด่นของเกม MMORPG แยกออกมาเป็นข้อย่อยๆได้ดังนี้

5.2.1 สถานะถาวร

การมีสถานะถาวรหมายถึงการที่เก็บสถานะของตัวละครไว้ตลอดเวลาแม้ว่าผู้เล่นจะออกจากเกมไปแล้วก็ตาม ถึงแม้ว่าผู้เล่นจะเข้าและออกจากเกมตลอดเวลาสถานะของทุกคนจะถูกเก็บไว้ในฐานข้อมูลโดยแยกจากกัน ซึ่งจะแตกต่างจากเกมทั่วไปที่การเก็บข้อมูลจะกระทำเมื่อผู้เล่นหยุดการเล่นแล้วออกจากโปรแกรมเท่านั้น

5.2.2 ข้อมูลถูกต้องตรงกัน(Data Integrity)

เนื่องจากเกม MMORPG จะมีผู้เล่นเชื่อมต่อเข้ามาจากที่ต่างๆกันและเล่นพร้อมกันได้หลายๆคน ดังนั้นข้อมูลทั้งหมดจึงควรจะเก็บไว้ที่เซิร์ฟเวอร์เพื่อให้เกิดความเท่าเทียมกันต่อผู้เล่นทุกคน การจัดเก็บ

ข้อมูลของผู้เล่นไว้ที่เซิร์ฟเวอร์จะทำให้ข้อมูลที่เก็บไว้ตรงกับความเป็นจริงสำหรับทุกคน ซึ่งถ้าหากจัดเก็บไว้บนไคลเอนท์ ผู้เล่นบางคนอาจจะทำการแก้ไขข้อมูลทำให้เกิดความเสียหายหรือไม่เท่าเทียมกันได้

5.2.3 ผู้เล่นสามารถมีส่วนในการครอบครอง

เกม MMORPG ที่ดีจะเว้นพื้นที่บางส่วนในเกมเพื่อให้ผู้เล่นสร้างเพิ่มเติม เช่นปราสาทที่สามารถให้ผู้เล่นสามารถต่อเติมได้ เพื่อสร้างความรู้สึกเป็นเจ้าของและสร้างสังคมจำลองขึ้นในกลุ่มคนที่เป็นเจ้าของร่วมกัน แต่การทำเช่นนี้ได้ต้องกำหนดสิทธิการเข้าถึงเซิร์ฟเวอร์เอาไว้ระดับหนึ่งเพื่อป้องกันความเสียหายจากการที่ผู้เล่นกำหนดบางสิ่งบางอย่างซึ่งอาจเป็นผลร้ายต่อเกมโดยรวม

อัตราส่วนของการครอบครองที่ผู้เล่นทำได้ในแต่ละเกมจะแตกต่างกันออกไปซึ่งส่งผลให้รูปแบบของเกมแตกต่างกันออกไปอย่างเห็นได้ชัด ในเกมผู้เล่นมีสิทธิสูงจะเกิดสังคมขึ้นอย่างรวดเร็วและกว้างขวาง เพราะทรัพย์สินในเกมเหล่านี้มักจะต้องใช้ความพยายามสูงในการครอบครองซึ่งทำได้ยากถ้าหากใช้คนเพียงคนเดียว จึงมีการร่วมมือและช่วยกันของผู้เล่นหลายๆคนในการรักษาทรัพย์สินภายในเกมเอาไว้ซึ่งจะพัฒนากลายเป็นสังคมจำลองได้ในเวลาอันรวดเร็ว

5.2.4 การพัฒนาความสามารถต่างๆ

ในเกม RPG ทั่วไป ผู้เล่นจะพัฒนาเลเวลของความสามารถด้วยการดำเนินเกมไปจนถึงเงื่อนไขที่กำหนด ในเกม MMORPG ผู้เล่นก็สามารถพัฒนาความสามารถได้เช่นกัน แต่จะต่างออกไปในเรื่องของลำดับชั้นของเลเวล ค่าประสบการณ์ที่ใช้เป็นเงื่อนไขของเกม MMORPG จะเป็นแบบ Exponential ที่สูงมากเพื่อให้ผู้เล่นใช้เวลานานขึ้นในการขึ้นสู่เลเวลสูงสุด

นอกจากนี้ในเกม MMORPG ส่วนใหญ่จะมีการเปรียบเทียบหรือจัดลำดับผู้เล่นที่มีความสามารถสูงสุดเพื่อให้ผู้เล่นแข่งขันและพยายามที่จะเลื่อนอันดับขึ้นไปซึ่งเป็นการดึงดูดให้ผู้เล่นกลับมาเล่นอีกบ่อยครั้งขึ้น

5.2.5 การดำเนินเนื้อเรื่อง

เนื้อเรื่องของเกม RPG จะมีเส้นทางให้เลือกจำกัดเนื่องจากใช้พื้นที่ในเครื่องจัดเก็บข้อมูลเนื้อเรื่องซึ่งเกม MMORPG จัดเก็บลงบนฐานข้อมูลที่มีขนาดใหญ่กว่า ดังนั้นเนื้อเรื่องใน MMORPG จะไม่จำกัดเส้นทางโดยจะอยู่ในลักษณะ Event ย่อยๆ ถ้าหากผู้เล่นมีเงื่อนไขครบก็สามารถทำได้ทันทีโดยไม่ต้องเคลียร์เงื่อนไขไปตามลำดับตั้งแต่ต้นจนจบ

การจัดการเนื้อเรื่องแบบนี้จะสะดวกในการแก้ไข ถ้าหากเกิดปัญหาขึ้นกับข้อมูลบางส่วนก็สามารถแก้ไขได้โดยที่ใช้เวลาไม่นานเพราะส่วนของเกมและข้อมูลแยกจากกัน ถ้าโปรแกรมของเซิร์ฟเวอร์ยังไม่เสียหายก็สามารถป้อนข้อมูลที่เก็บแบ็คอัพเอาไว้กลับลงไปใหม่ได้ตลอดเวลา

5.2.6 สังคมจำลอง

จุดมุ่งหมายของเกมออนไลน์โดยเฉพาะแบบ MMORPG จะอยู่ที่การเล่นซ้ำมากกว่าการเล่นเพื่อให้จบเนื้อเรื่อง ดังนั้นเกม MMORPG ส่วนใหญ่จึงไม่มีจุดสิ้นสุดที่ชัดเจน สามารถเพิ่มเติมฉากใหม่ๆ หรือเนื้อเรื่องใหม่ๆลงไปได้ตลอดเวลา ผู้เล่นที่เข้าไปในโลกของ MMORPG จึงจำเป็นต้องแลกเปลี่ยนข้อมูลกับผู้เล่นคนอื่นๆเพื่อติดตามการเปลี่ยนแปลงนี้ให้ทันอยู่ตลอดเวลา

ระบบของเกมที่มีการสื่อสารของผู้เล่นทำให้ภายในเกมเกิดความร่วมมือกัน ภายใต้กรอบหลวมๆ ของเกมแบบ MMORPG ผู้เล่นสามารถติดต่อกับผู้เล่นคนอื่นๆเพื่อสอบถามเนื้อเรื่องบางส่วนที่ตนเองไม่รู้ หรือสามารถร่วมมือกับผู้เล่นกลุ่มอื่นในการต่อสู้หรือการดำเนินเนื้อเรื่องนั้นๆได้ การพัฒนาตัวละครก็อาจจะให้ผู้เล่นคนอื่นๆช่วยได้เช่นกัน และในระบบเกม MMORPG ส่วนใหญ่จะมีการจำลองระบบการค้าไว้ภายใน ไอเทมภายในเกมจึงอาจจะซื้อขายกันระหว่างผู้เล่นโดยที่ไม่ต้องติดต่อกับ NPC (Non-Playable Character) ซึ่งควบคุมโดยคอมพิวเตอร์แต่เพียงอย่างเดียว ระบบของสังคมจำลองจึงทำให้เกมออนไลน์สามารถดึงดูดให้ผู้เล่นเข้าสู่ระบบได้มากครั้งกว่า

บทบาทของผู้เล่นในเกม RPG ทั่วไปจะอยู่ที่ตัวเอก แต่ใน MMORPG แล้ว ผู้เล่นจะมีความสามารถในการร่วมทีมโดยที่ให้ผู้เล่นคนอื่นๆมีหน้าที่ต่างๆกันไปสำหรับแต่ละทีม ผู้เล่นมีสิทธิแสดงบทบาทของตนเองได้ทั้งในแง่ดีและไม่ดี โดยผู้เล่นคนหนึ่งอาจจะก่อความเดือดร้อนให้กับผู้เล่นคนอื่นๆได้เช่นเดียวกับสังคมทั่วไป

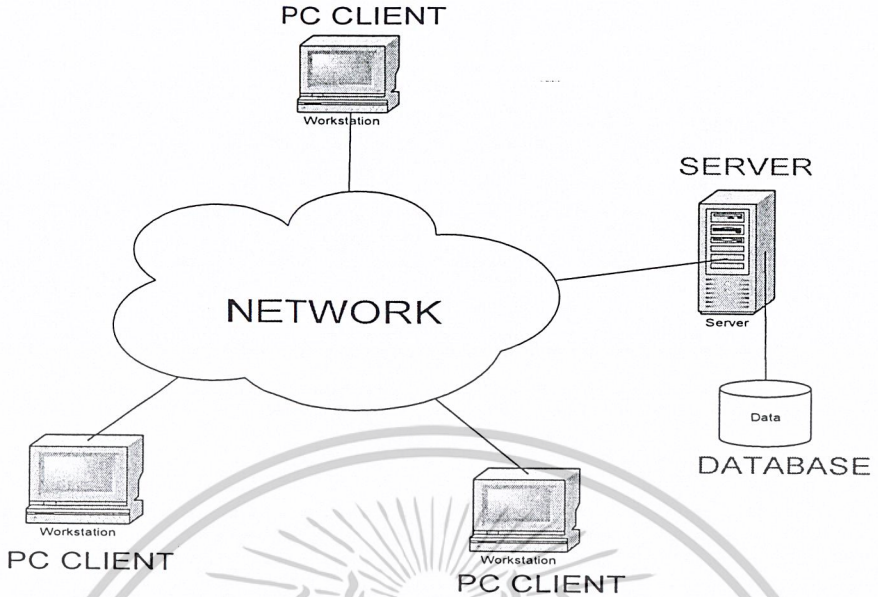
5.3 ฐานข้อมูลใน MMORPG

เนื่องจากข้อมูลของผู้เล่นจำเป็นจะต้องถูกบันทึกอยู่ตลอดเวลา ทุกครั้งที่มีการรับส่งข้อมูลจากเครื่องไคลเอนท์ หลังจากที่แปลความหมายของแพคเก็จนั้นๆได้แล้ว ข้อมูลที่ได้จะต้องถูกบันทึกอยู่ตลอดเวลา แม้แต่การเคลื่อนที่ของตัวละครก็จำเป็นต้องเก็บเอาไว้เพื่อทำการบันทึกตำแหน่งล่าสุดของแต่ละตัวละครเอาไว้ การบันทึกลงไฟล์ไม่สะดวกเนื่องจากการบันทึกข้อมูลลงอุปกรณ์ช้าเกินไป จึงต้องจัดเก็บข้อมูลที่ส่งมาจากไคลเอนท์ไว้ในฐานข้อมูลรวม เมื่อถึงระยะเวลาหนึ่งเซิร์ฟเวอร์ก็จะทำการกระจายข้อมูลเพื่ออัพเดทตำแหน่งของตัวละครต่างๆให้กับทุกไคลเอนท์

เนื่องจากงานของฐานข้อมูลดังกล่าวเป็นงานที่หนักและต้องการความเร็วสูง ระบบฐานข้อมูลในเกม MMORPG จึงมีความสำคัญเป็นอย่างมาก ทุกๆครั้งที่มีการส่งข้อมูลจากในไคลเอนท์ เซิร์ฟเวอร์จำบันทึกลงในฐานข้อมูล หลังจากนั้นเมื่อไคลเอนท์เครื่องใดร้องขอก็จะจัดส่งไปให้ หรือถ้าหากถึงรอบเวลาที่ต้องอัพเดตข้อมูลให้กับไคลเอนท์ก็จะกระจายข้อมูลตำแหน่งไปให้กับทุกๆเครื่อง รอบเวลาอัพเดตต้องอยู่ในช่วงที่ไม่นานเกินไปจนทำให้ไคลเอนท์แต่ละคนแสดงผลไม่เหมือนกัน

นอกจากนี้ฐานข้อมูลจะต้องจัดเก็บเสตทของผู้เล่นที่กำลังดำเนินเนื้อเรื่องต่างๆ รายชื่อและรายละเอียดของไอเทม บทสนทนาของตัวละคร ซึ่งทั้งหมดจำเป็นต้องเก็บในฐานข้อมูลเพราะเกม MMORPG สามารถจะเปลี่ยนแปลงรายละเอียดภายในเกมได้ตลอดเวลาเพื่อขยายเกมและจัดอีเวนท์ประจำช่วงเวลาสำคัญต่างๆ

5.4 สถาปัตยกรรมของเกม MMORPG



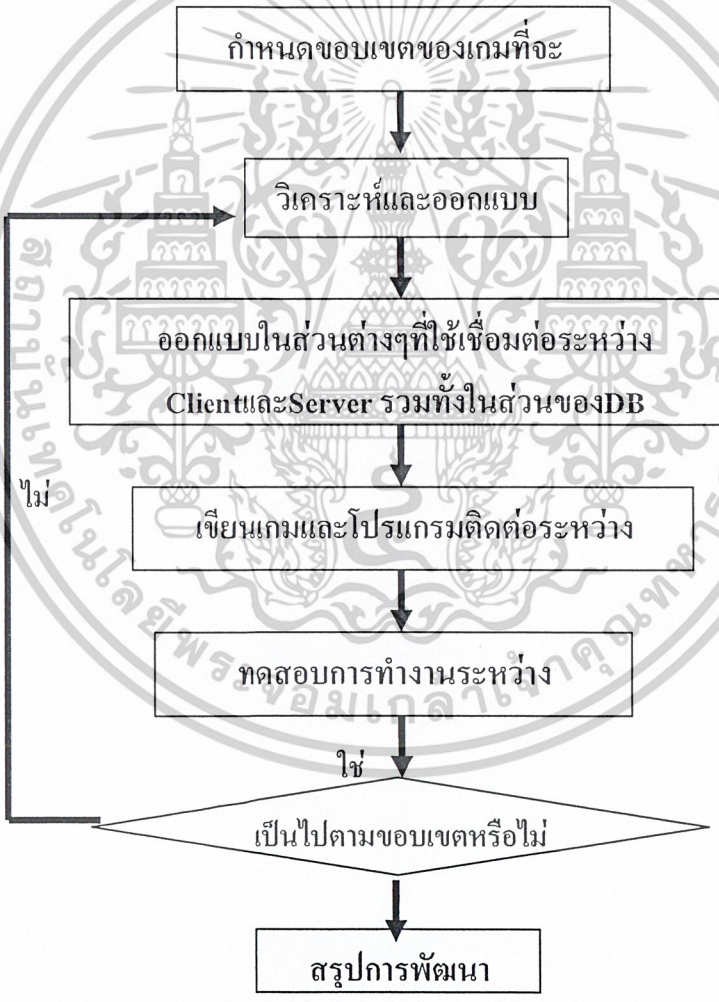
รูปที่ 5-1 สถาปัตยกรรมของเกม MMORPG

จากรูปแบบการดำเนินการที่ต้องติดต่อกับผู้เล่นหลายๆคนพร้อมกันและต้องมีการกระจายข้อมูลจากเครื่องหนึ่งไปยังเครื่องอื่นๆที่เหลือ สถาปัตยกรรมของเกม MMORPG จึงเป็นแบบ Client-Server ที่สะดวกในการควบคุมการเชื่อมต่อและรับส่งข้อมูลมากกว่า โดยเมื่อมีไคลเอนท์เข้าสู่เกม จะต้องมีการยืนยันตนกับเซิร์ฟเวอร์ ถ้าหากยืนยันตนสำเร็จเซิร์ฟเวอร์จึงจะสร้างคอนเนคชั่นสำหรับให้ไคลเอนท์รับส่งข้อมูลโดยผ่านโปรโตคอล TCP/IP ซึ่งเป็นโปรโตคอลมาตรฐานบนอินเตอร์เนต หลังจากนั้นเมื่อไคลเอนท์เริ่มทำการรับส่งข้อมูลกับซิร์ฟเวอร์ ตัวเซิร์ฟเวอร์เองจะมีหน้าที่จัดส่งข้อมูลให้กับไคลเอนท์อื่นๆเพื่อให้ทุกเครื่องดำเนินการไปพร้อมๆกันและแสดงผลเหมือนกันในทุกๆเครื่อง

บทที่ 6
ภาพรวมของการออกแบบและพัฒนา

6.1 แนวคิดการออกแบบ

เริ่มจากการวิเคราะห์ระบบเกมแล้วทำการออกแบบการทำงานในส่วนต่างๆ เช่น ระบบเน็ตเวิร์ค การแสดงผล การคำนวณของฝั่งเซิร์ฟเวอร์ และการต่อต่อกับฐานข้อมูล หลังจากที่ออกแบบเรียบร้อยแล้ว จึงลงมือเขียนโปรแกรมเพื่อทดสอบการทำงานในขั้นต้น แล้วทำการทดสอบว่าตรงตามขอบเขตที่ได้วางไว้หรือไม่ ถ้าหากถูกต้องก็จะนำรวบรวมโปรแกรมขั้นต้นที่ทดสอบแล้วมาทำการรวมเป็นโปรแกรมที่สมบูรณ์แล้วทดสอบในขั้นตอนสุดท้ายจึงสรุปผล



รูปที่ 6-1 แนวทางการออกแบบ

6.2 การทำงานของไคลเอนท์

ไคลเอนท์คือเครื่องที่ทำการร้องขอการเชื่อมต่อไปยังเซิร์ฟเวอร์ ไคลเอนท์อาจจะมีจำนวนตั้งแต่หนึ่งเครื่องขึ้นไปโดยทุกครั้งที่เริ่มเกมจะร้องขอการเชื่อมต่อไปยังเซิร์ฟเวอร์ ถ้าหากเชื่อมต่อสำเร็จจึงจะรับข้อมูลล่าสุดที่เซิร์ฟเวอร์ดึงขึ้นมาจากฐานข้อมูลแล้วส่งมาให้ และเมื่อผู้เล่นสั่งการใดๆ ไคลเอนท์จะส่งการตัดสินใจของผู้เล่นไปให้เซิร์ฟเวอร์จัดการทุกครั้ง การติดต่อกันเช่นนี้จะมีอยู่ตลอดเวลา ถ้าผู้เล่นสั่งปิดโปรแกรม หรือหากการเชื่อมต่อมีปัญหาก็จะทำให้โปรแกรมหยุดลงและไคลเอนท์จะต้องออกจากเกมในที่สุด

เครื่องไคลเอนท์จะทำหน้าที่ส่งข้อมูลจากผู้ใช้ไปให้เซิร์ฟเวอร์เป็นหลัก แต่ถึงแม้การประมวลผลเกมทั้งหมดจะทำบนเซิร์ฟเวอร์ แต่ก็มีหน้าที่บางอย่างที่สามารถทำบนไคลเอนท์ได้เลยโดยไม่ต้องส่งไปให้เซิร์ฟเวอร์ประมวลผลก่อน การทำเช่นนี้จะช่วยลดภาระการทำงานของเซิร์ฟเวอร์ลงส่วนหนึ่งโดยมีข้อแม้ว่าการประมวลผลนั้นจะต้องไม่มีผลต่อไคลเอนท์อื่นโดยตรง หรือไม่ก็ต้องมีการวิธีการคำนวณที่สามารถให้ผลตรงกันในทุกๆไคลเอนท์ได้แม้ว่าจะไม่มีการส่งข้อมูลไปมาก็ตาม เช่น การหาเส้นทางเดินของตัวละครจากจุดหนึ่งไปยังจุดหนึ่ง การโจมตีศัตรูแบบอัตโนมัติ แต่ทั้งนี้ต้องคำนึงถึงความปลอดภัยของข้อมูลที่คำนวณบนไคลเอนท์เป็นหลัก

6.3 การทำงานของเซิร์ฟเวอร์

เซิร์ฟเวอร์ คือ เครื่องที่ใช้เป็นศูนย์กลางการติดต่อระหว่างไคลเอนท์แต่ละเครื่อง เป็นเครื่องที่รับหน้าที่ประมวลผลข้อมูลที่ไคลเอนท์แต่ละเครื่องส่งมา เครื่องเซิร์ฟเวอร์จะเป็นฝ่ายดำเนินเกม ทำหน้าที่ควบคุมการทำงานของเกมต่างๆและกระจายการประมวลผลหรือคำสั่งควบคุมไปยังไคลเอนท์ นอกจากนี้เซิร์ฟเวอร์ยังทำหน้าที่ติดต่อกับฐานข้อมูลสำหรับไคลเอนท์ทุกเครื่องอีกด้วย การเข้าถึงข้อมูลของไคลเอนท์แต่ละเครื่องจะต้องกระทำผ่านโปรแกรมของเซิร์ฟเวอร์แต่เพียงอย่างเดียว ดังนั้นจำนวนของไคลเอนท์จึงมีผลต่อประสิทธิภาพการทำงานของเซิร์ฟเวอร์โดยตรง ถ้าหากต้องการให้สามารถรองรับไคลเอนท์ได้เป็นจำนวนมาก ก็จะต้องจัดการให้เซิร์ฟเวอร์มีประสิทธิภาพมากขึ้นตามไปด้วย

นอกจากนี้ เนื่องจากเครื่องเซิร์ฟเวอร์ จะทำหน้าที่เป็นศูนย์กลางการทำงานของระบบ ดังนั้นเซิร์ฟเวอร์จะต้องมีความมั่นคงสูงและมีความปลอดภัยในระดับหนึ่ง ต้องสามารถรองรับการเชื่อมต่อของไคลเอนท์ได้อย่างมีประสิทธิภาพและสามารถประมวลผลข้อมูลได้อย่างถูกต้องและมีความรวดเร็ว เพราะถ้าหากเครื่องเซิร์ฟเวอร์เกิดปัญหาขึ้นหรือประมวลผลไม่ทัน เครื่องไคลเอนท์ทั้งหมดจะได้รับผลกระทบทันที

6.4 การจัดเก็บข้อมูล

เซิร์ฟเวอร์จำเป็นต้องมีระบบฐานข้อมูลที่มีเสถียรภาพเพราะข้อมูลเป็นส่วนหลักที่จะดำเนินเกม เมื่อผู้เล่นเชื่อมต่อเข้ามาจะเริ่มมีการเปลี่ยนแปลงข้อมูล จากนั้นเมื่อผู้เล่นออกจากเกมจะต้องมีการเซฟข้อมูลสุดท้ายของผู้เล่นเอาไว้ ถ้าหากระหว่างนั้นข้อมูลสูญหายหรือเซิร์ฟเวอร์ที่ใช้เก็บข้อมูลหยุดทำงานจะต้องสามารถกู้ข้อมูลที่สำรองไว้ล่าสุดกลับมาได้(roll back) ดังนั้นระบบสำรองข้อมูลจึงเป็นสิ่งจำเป็น

จะต้องสามารถดึงข้อมูลทั้งหมดออกมาเป็นเป็นข้อมูลสำรองได้ในทุกๆช่วงเวลาหนึ่ง โดยจะกำหนดเอาจากสัดส่วนผู้เล่นและขนาดข้อมูลของเกม

นอกจากนี้การเข้าถึงข้อมูลก็เป็นส่วนที่แตกต่างจากเกมอื่นๆ เกมออนไลน์มีผู้เล่นเชื่อมต่อเข้ามาเป็นจำนวนมาก ผู้เล่นแต่ละคนก็มีข้อมูลของตัวเองเป็นจำนวนมาก เช่น รายชื่อไอเทม ค่าต่างๆของตัวละคร ในเกมอื่นๆส่วนใหญ่จะใช้ไฟล์ข้อมูลที่กำหนดเอง (Customize file) เพื่อความสะดวกในการเขียนโปรแกรม แต่สำหรับเกมออนไลน์ การใช้ไฟล์ข้อมูลทำได้ยากเนื่องจากมีการเข้าถึงข้อมูลจำนวนมากอยู่ตลอดเวลา โปรแกรมฐานข้อมูลต่างๆ เช่น MySQL, Microsoft SQL 2000 จึงเหมาะสมกว่าเพราะสามารถจัดลำดับการเข้าถึงข้อมูลได้ดีกว่า และสามารถจัดการกับข้อมูลได้โดยตรงผ่านทางอินเตอร์เฟซของตัวโปรแกรมเอง

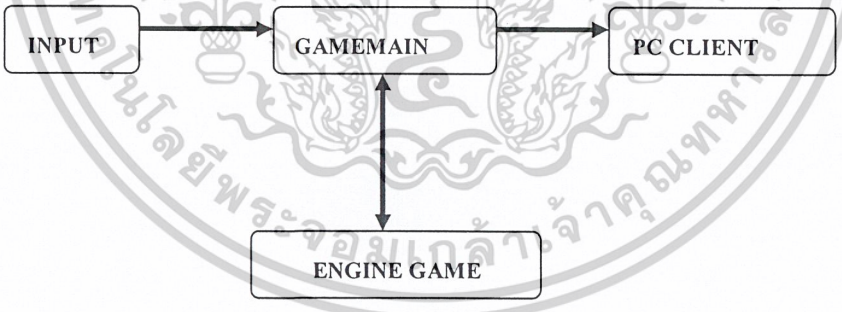
บทที่ 7
ไคลเอนท์เกม

7.1 วิเคราะห์การทำงานของไคลเอนท์

ในระบบเกม MMORPG ไคลเอนท์เป็นส่วนที่ติดต่อกับผู้ใช้และคอยรับข้อมูลจากเซิร์ฟเวอร์ไปแสดงผลโดยที่มีส่วนคำนวณประมวลผลเพียงเล็กน้อยเท่านั้น ต่างจากเกมทั่วไปที่ทำการประมวลผลทั้งหมดในเครื่องตัวเอง ไคลเอนท์จะถูกจำกัดให้มีการคำนวณเฉพาะเท่าที่จำเป็นเท่านั้น เช่น การคำนวณเกี่ยวกับการแสดงผลบนหน้าจอ การหาเส้นทางระยะสั้นๆสำหรับตอบสนองต่อผู้เล่น สำหรับการคำนวณในส่วนที่เหลือจะถูกผลักภาระไปทำบนเซิร์ฟเวอร์ เหตุที่ต้องทำเช่นนี้เนื่องมาจากรูปแบบเกมที่สามารถเล่นพร้อมกันได้หลายคน การแสดงผลต้องปรากฏเหมือนๆกันในแต่ละเครื่อง จึงสะดวกมากกว่าถ้าหากให้เซิร์ฟเวอร์เป็นคนประมวลผลแล้วกระจายข้อมูลออกไป แทนที่จะให้ไคลเอนท์แต่ละเครื่องประมวลผลกันเองซึ่งอาจจะทำให้การแสดงผลออกมาไม่เหมือนกันได้

7.2 การออกแบบไคลเอนท์

รายละเอียดการทำงานของเกม มีอยู่ 3 ส่วน คือ ส่วนแรกจะเป็น Game Main จะรับค่าอินพุทจากผู้เล่นและประมวลผลโดยเรียกใช้ ส่วนที่สองคือส่วนของตัวเกม (Engine Game) และ แล้วจะส่งผลของการประมวลกลับไปให้ตัว Game Main โดยที่ตัวเกมจะมีการเรียกใช้ เอนจิน อีกครั้งหนึ่ง เมื่อประมวลผลเสร็จแล้ว ก็นำค่าที่ได้มาตรวจสอบเพื่อแสดงผลออกมา



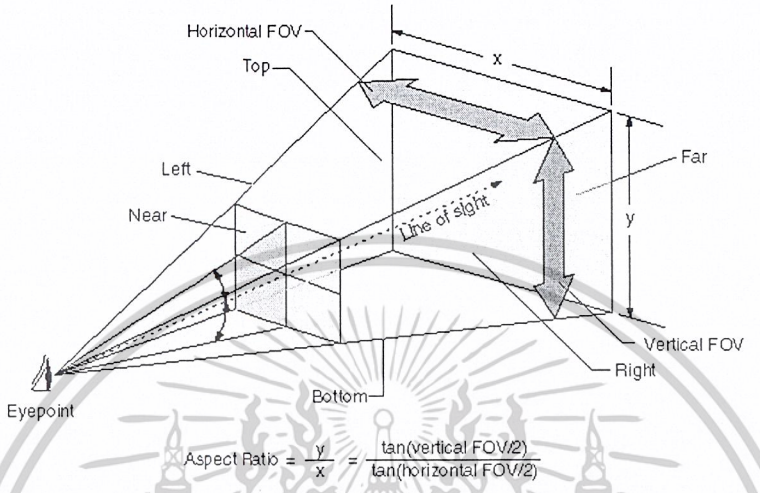
รูปที่ 7-1 การทำงานของเกม

7.2.2 เทคนิคประมวลผลของไคลเอนท์

เทคนิคที่นำมาใช้ในส่วนของกราฟิกนี้ เป็นส่วนที่มีไว้เพื่อจัดการกับเกมให้มีประสิทธิภาพในด้านเรื่องต่างๆเช่น การลดการแสดงผลในส่วนที่ไม่จำเป็น การทำภาพโปร่งแสง เป็นต้น ซึ่งจะมีหัวข้อต่างๆที่ใช้ดังนี้

7.2.2.1 Viewing Frustum

เป็นวิธีการกำหนดขอบเขตของการแสดงผลโดยการคำนวณจากขนาดของความละเอียดและตำแหน่งของกล้อง เพื่อตรวจสอบว่าส่วนใดไม่อยู่ภายในขอบเขตนี้ ถ้าส่วนนั้นไม่อยู่ภายในขอบเขตจะไม่ถูกแสดงผล(render) ซึ่งวิธีการทำ frustum นั้นปกติจะสร้างเป็น ปิรามิดฐานสี่เหลี่ยม ดังรูป

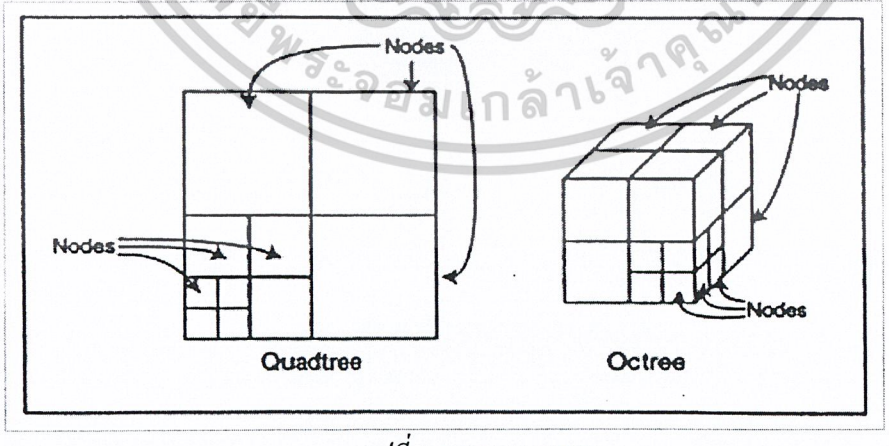


รูปที่ 7-2 Viewing Frustum

7.2.2.2 NodeTree

เป็นวิธีการลดเวลาในการแสกนผ่านทุกๆโพลีกอนว่าอยู่ใน Frustum หรือไม่ ซึ่งมีอยู่ 2 แบบ คือ

1. **QUADTREE** - เป็นการแบ่งส่วนของวัตถุออกเป็น 4 ส่วน โดยวิธีนี้จะเหมาะกับเกมที่ไม่มีการเปลี่ยนแปลงค่ามุมของกล้องที่ทำกับแกน Y มากนัก
2. **OCTREE** - เป็นการแบ่งส่วนของวัตถุออกเป็น 8 ส่วน โดยส่วนมากจะใช้วิธีนี้กับวัตถุที่มีขนาดใหญ่ เช่นเกมที่มีมุมมองบุคคลที่ 1 ซึ่งสามารถมองเห็นได้รอบทิศ

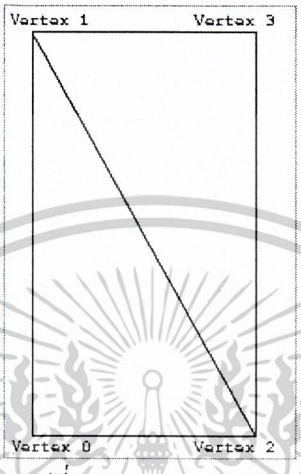


รูปที่ 7-3 NODE TREE

7.2.2.3 BillBoard

เป็นเทคนิคที่ใช้ในการแสดงผลจากภาพ 2 มิติ ในโลก 3 มิติ ซึ่งภาพที่แสดงจะขนานกับกล้องตลอดเวลา ทำให้ดูเหมือนกับว่า เป็นภาพ 2 มิติตลอดเวลา ในการจัดวางรูป 2 มิติในจะต้องมีความเข้าใจในเรื่อง coordinate ก่อนดังนี้

ในการจะวางรูป 2 มิติในโลก 3 มิติเราจะต้องกำหนดจุด 4 จุดวางในโลก 3 มิติ โดยตำแหน่งในการวางด้วย UV coordinate ดังรูป



รูปที่ 7-4 UV coordinate

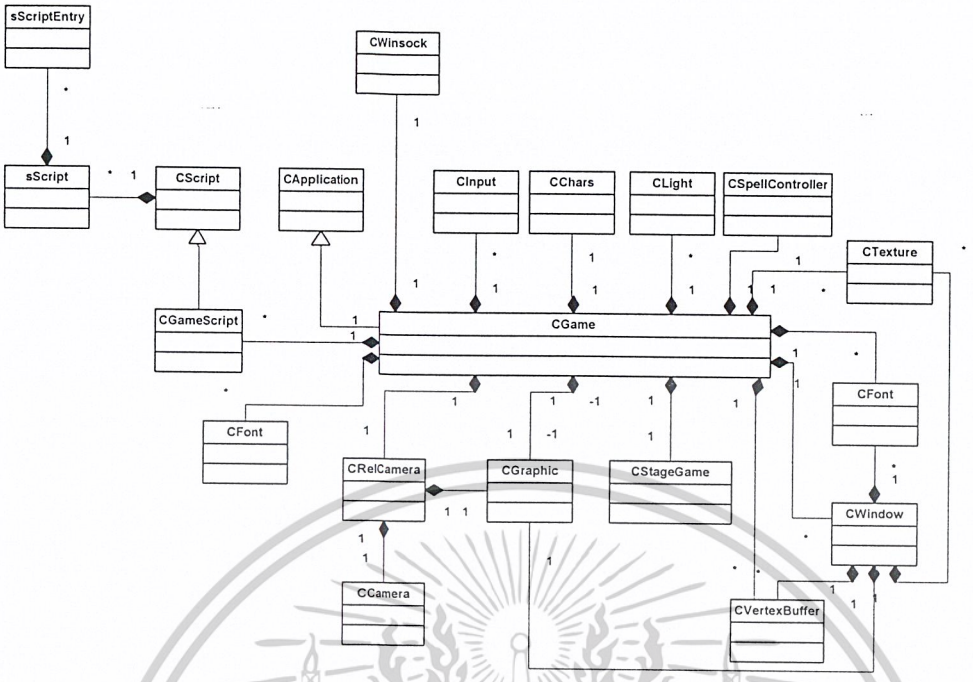
จากรูปถ้าเราต้องการวางภาพให้หันออกหน้าจอเราจะต้องเราแมปภาพโดยกำหนดจุดเริ่มต้นที่ Vertex 0 และวางลงไปเรื่อยๆ ที่ Vertex 1, Vertex 2 และสิ้นสุดที่ Vertex 3

7.2.2.4 AlphaTesting & AlphaBlending

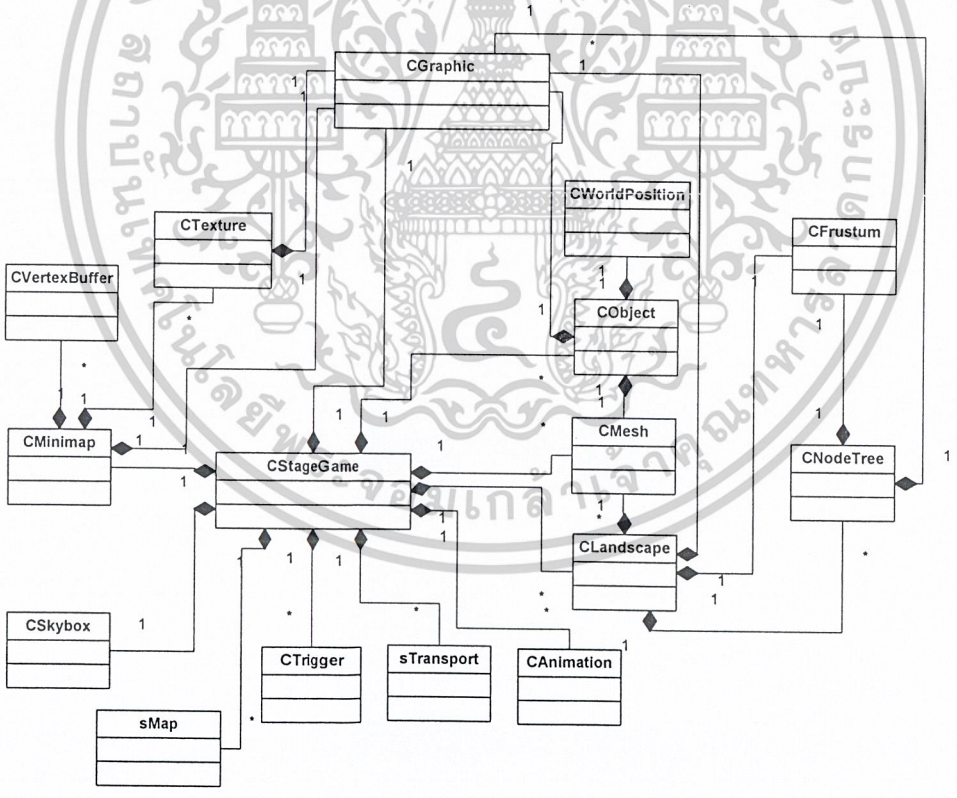
Alpha Testing เป็นการทำให้ส่วนที่ไม่ต้องการโปร่งแสง โดยการ set color key ที่ไม่ต้องการให้แสดง Alpha Blending ส่วนสีที่ขอบที่ไม่ดำสนิท จะต้องทำการ blend สีให้เข้ากับพื้นหลังโดยใช้ Alpha blending

7.3 โครงสร้างคลาสของไคลเอนท์

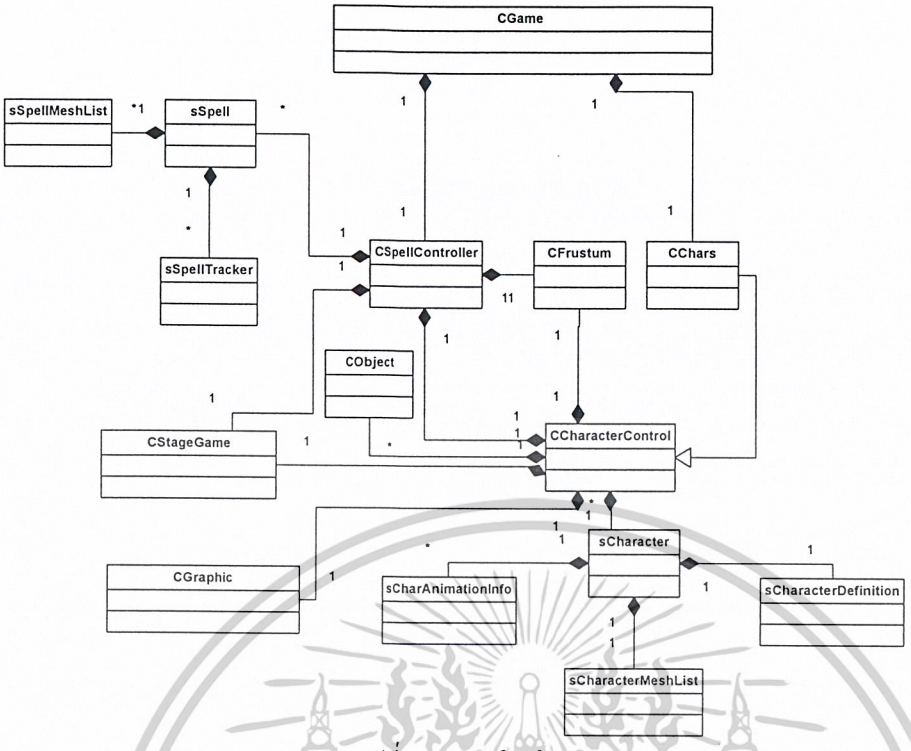
ในไคลเอนท์เกมจะมีคลาสหลักที่ใช้ในการประมวลผลเกม โดยจะเรียกคลาสอื่นๆ เข้ามาใช้งาน โดยรวมแล้วจะมีรูปแบบของไดอะแกรมดังรูป



รูปที่ 7-5 ภาพรวมของคลาสบนไคลเอนท์



รูปที่ 7-6 คลาสในส่วนของ StageGame

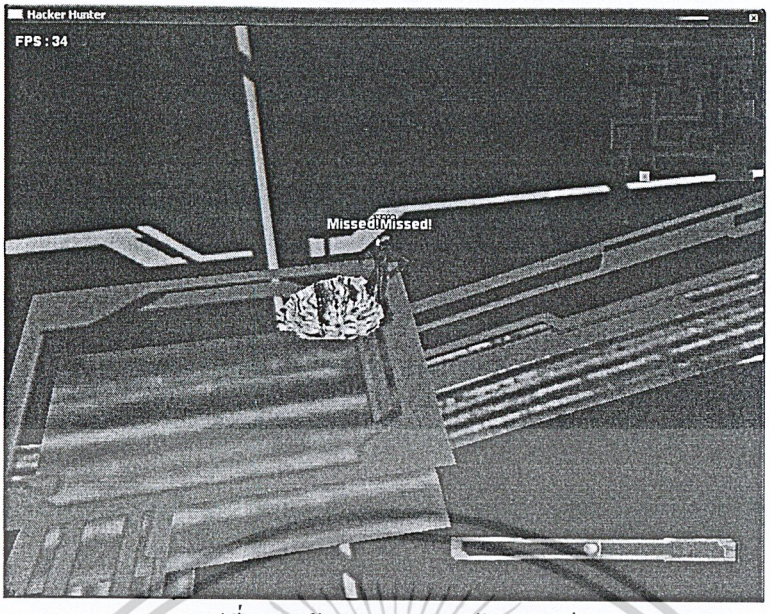


รูปที่ 7-7 คลาสในส่วน Character

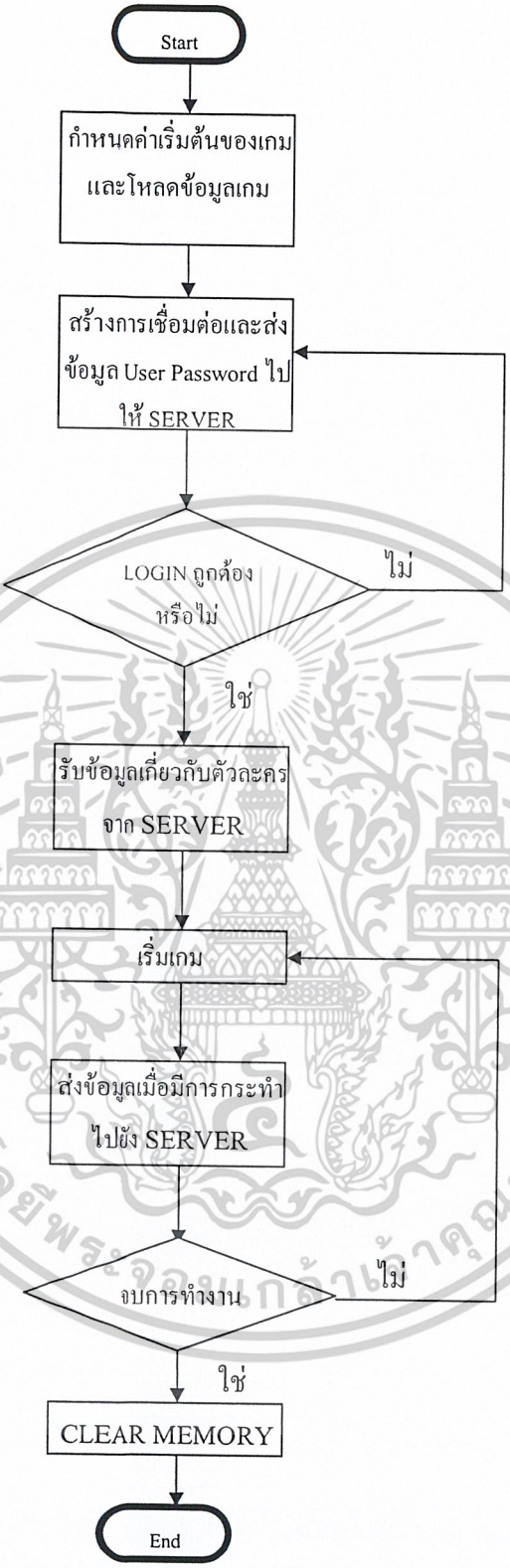
7.4 อินเตอร์เฟซของไคลเอนท์

หน้าจอที่ปรากฏบนไคลเอนท์ จะแสดงผลกราฟิคเกมในลักษณะของวินโดว์ ข้อมูลจะถูกประมวลผลแล้วแสดงเป็นภาพสามมิติ โดยมีเมนูควบคุมเกมบางส่วนแสดงบนหน้าจอ เช่น แผนที่ขนาดเล็กบนมุมบนขวาของหน้าจอ เพื่อให้ผู้เล่นมีพื้นที่มากที่สุดจึงจัดให้เมนูทั้งหมดเรียงอยู่บริเวณขอบจอ เมื่อผู้เล่นสั่งการใดๆผ่านคีย์บอร์ดหรือเมาส์ โปรแกรมจะทำการประมวลผลแล้วแสดงบนหน้าจอก่อนที่จะจัดส่งข้อมูลไปยังเซิร์ฟเวอร์เพื่อให้ส่งต่อไปยังไคลเอนท์ของตัวละครอื่นๆที่อยู่บริเวณใกล้เคียง ในกรณีที่มีการส่งข้อมูลจากเซิร์ฟเวอร์เพื่อเปลี่ยนตำแหน่งของตัวละครที่อยู่ในฉาก ไคลเอนท์จะทำการคำนวณเส้นทางการเดินของตัวละครแต่ละตัวแล้วแสดงผลขึ้นบนหน้าจอเช่นกัน

การแสดงผลจะเป็นแบบสามมิติ(3D) ซึ่งจะวาดตัวละครในระบบโพลีกอนที่แบ่งพื้นผิวออกเป็นชิ้นสามเหลี่ยมเล็กๆประกอบกัน ผู้เล่นสามารถหมุนหน้าจอเพื่อเปลี่ยนมุมมองภายในเกมได้ การควบคุมจะทำผ่านคีย์บอร์ดและเมาส์ ใช้เมาส์คลิ๊กบนตำแหน่งของฉากในจอเพื่อสั่งให้ตัวละครเดินไปในทิศทางนั้นหรือสั่งโจมตีตัวละครที่ผู้เล่นคลิ๊ก เมื่อตัวละครเคลื่อนที่ หน้าจอเกมจะเลื่อนโดยที่ตัวละครอยู่ตรงกลางของหน้าจอตลอดเวลา



รูปที่ 7-8 หน้าจอแสดงผลบนไคลเอนท์



รูป 7-9 แผนภาพแสดงขั้นตอนการทำงานของไคลเอนท์

บทที่ 8
เซิร์ฟเวอร์เกม

8.1 วิเคราะห์การทำงานของเซิร์ฟเวอร์

เซิร์ฟเวอร์จะต้องสามารถประมวลผลการติดต่อกับไคลเอนท์ผ่านระบบเครือข่ายได้โดยที่ไม่เกี่ยวข้องหรือมีผลกับการประมวลผลเกม ดังนั้นส่วนควบคุมระบบเนตเวิร์คจะต้องแยกเป็นอิสระหรือทำงานคนละเทรดกันกับระบบเกม ในระบบเกมเองจะต้องมีการสร้างตัวละครซึ่งเป็นตัวละครที่ผู้เล่นควบคุมไม่ได้หรือ NPC การทำงานของ NPC แต่ละตัวจะต้องเป็นอิสระจากกันเพื่อให้ NPC แต่ละตัวสามารถแสดงการทำงานได้อย่างเป็นธรรมชาติ ดังนั้นเซิร์ฟเวอร์จะต้องสร้างเทรดสำหรับแต่การทำงานของ NPC แต่ละตัวขึ้นในของระบบเกม โดยทั้งส่วนของระบบเนตเวิร์คและระบบเกมจะติดต่อร่วมกันโดยการดึงและเขียนข้อมูลลงในฐานข้อมูล

เซิร์ฟเวอร์ของเกมออนไลน์มีลักษณะเหมือนเซิร์ฟเวอร์ของโปรแกรมที่ต้องทำงานแบบ Client-Serverโดยทั่วไป จะต่างออกไปที่เรื่องของการเชื่อมต่อเป็นจำนวนมาก และการส่งข้อมูลจำนวนมากในเวลาที่จำกัด เพื่อให้ทันกับการประมวลผลของไคลเอนท์

ปัจจัยสำคัญที่มีผลก็คือ

1. ความเร็วในการประมวลผลของCPU
2. ความเร็วที่ได้ในการเชื่อมต่อของไคลเอนท์แต่ละเครื่อง
3. จำนวนเครื่องไคลเอนท์ที่เชื่อมต่อในขณะนั้น
4. การทำ Load Sharing ของเซิร์ฟเวอร์

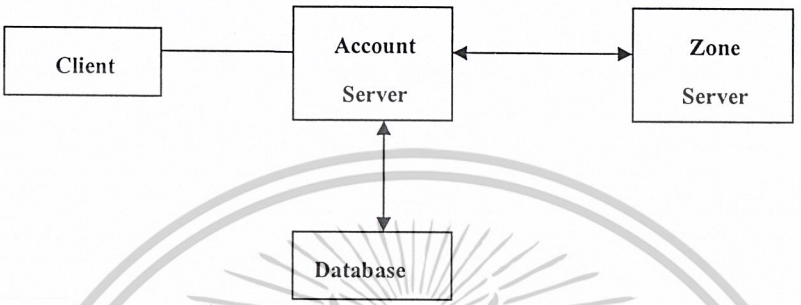
นอกจากนี้เซิร์ฟเวอร์จะต้องมีการทำงานอื่นๆเพื่ออำนวยความสะดวกในการพัฒนาเกม เช่น ความสามารถในการอ่านไฟล์สคริปต์ การเก็บล็อกไฟล์ การแสดงผลการทำงานต่างๆขึ้นบนหน้าจอแบบเรียลไทม์ รวมถึงการแจ้งเตือนข้อผิดพลาดต่างๆที่เกิดขึ้นภายในเกมอีกด้วย

8.2 การออกแบบเซิร์ฟเวอร์

การทำงานของเซิร์ฟเวอร์จะเริ่มขึ้นโดยการสร้างเทรดของ NPC ต่างๆขึ้นมาตามค่าที่กำหนดไว้ในสคริปต์แล้วสั่งให้แต่ละเทรดเริ่มทำงาน หลังจากนั้นจึงสร้างการเชื่อมต่อขึ้นเพื่อรอการร้องขอจากไคลเอนท์ เมื่อมีการร้องขอการเชื่อมต่อเข้ามา ไคลเอนท์จะต้องยืนยันตนเองกับเซิร์ฟเวอร์ด้วยรหัสผ่าน ถ้ารหัสผ่านถูกต้องเซิร์ฟเวอร์จึงจะสร้างการเชื่อมต่อระหว่างไคลเอนท์เครื่องนั้นๆแล้วจึงเริ่มรับส่งข้อมูล ระหว่างนั้นจะมีการส่งข้อมูลของตำแหน่งตัวละครจากไคลเอนท์อื่นๆและตำแหน่งของ NPC กระจายออกไปยังไคลเอนท์เพื่อให้การแสดงผลในแต่ละเครื่องออกมาเหมือนหรือใกล้เคียงกันมากที่สุด โดยข้อมูลตำแหน่งของ NPC จะมาจากเทรดที่ทำการสร้างไว้ตั้งแต่แรก คำสั่งต่างๆที่ถูกส่งเข้ามาจากไคลเอนท์จะถูกประมวลผลแล้วจึงกระจายออกไปยังไคลเอนท์อื่นๆ ถ้าหากมีคำสั่งออกจากเกมหรือระบบเนตเวิร์คมีปัญหา เซิร์ฟเวอร์จะตัดการติดต่อและคืนทรัพยากรของไคลเอนท์นั้นๆคืนสู่ระบบ

8.2.1 การเชื่อมต่อระหว่างไคลเอนท์และเซิร์ฟเวอร์

การเชื่อมต่อของเกมออนไลน์จะเป็นแบบ Connection Orient เพื่อความสะดวกในการตรวจสอบผู้เล่น ในการเชื่อมต่อของผู้เล่นแต่ละครั้งจะต้องยืนยันชื่อและรหัสก่อนเป็นอันดับแรก การเชื่อมต่อแบบ Connection Orient จะกันไม่ให้แพ็คเก็ตที่ไม่มีที่มา (จากการเชื่อมต่อแบบ Connection less) เข้ามาทำงานในระบบเกม นอกจากนี้การเชื่อมต่อแบบ Connection Orient จะมีความน่าเชื่อถือในการส่งข้อมูลมากกว่า



รูปที่ 8-1 การเชื่อมต่อระหว่าง Client กับ Server

8.2.2 การออกแบบฐานข้อมูล

ฐานข้อมูลภายในเกมประกอบด้วยตารางดังต่อไปนี้

- Account - เก็บข้อมูลรายละเอียดส่วนตัวของผู้เล่นแต่ละคนเอาไว้
  - AID ID ของแต่ละแอคเคาท์
  - Aname ชื่อเจ้าของแอคเคาท์
  - Addr ที่อยู่ของเข้าของแอคเคาท์
  - Phone เบอร์โทรศัพท์เจ้าของแอคเคาท์
  - Hphone เบอร์โทรศัพท์เคลื่อนที่(ถ้ามี)ของเจ้าของแอคเคาท์
  - Sex เพศ
  - Email ระบุอีเมลล์ที่ใช้งานอยู่

- Login - เก็บรหัสผ่านสำหรับการล๊อกอินเข้าใช้งานของผู้เล่น
  - AID ID แอคเคาท์เจ้าของล็อกอิน
  - ID ID ของคาแรคเตอร์
  - Passwd พาสเวิร์ด

- Map - เก็บรายละเอียดของแผนที่ต่างๆในเกม
  - MapID ID ของแต่ละแผนที่

- MapName ชื่อของแผนที่
- Col จำนวนบล็อกของแผนที่ในแกนนอน
- Row จำนวนบล็อกของแผนที่ในแกนตั้ง

* บล็อกหนึ่งๆมีขนาดเท่ากับตัวละครของผู้เล่นโดยประมาณ

- Monster เก็บข้อมูลของ NPC ที่เป็นมอนเสตอร์
  - MonID ID ของมอนเสตอร์
  - MonName ชื่อของมอนเสตอร์
  - Class ประเภทของมอนเสตอร์
  - LV ระดับของมอนเสตอร์
  - EXP ค่าประสบการณ์ที่ได้เมื่อฆ่ามอนเสตอร์ตัวนั้นได้
  - HP จำนวนพลังชีวิต
  - MP จำนวนพลังเวทมนต์
  - Str ค่าความแข็งแกร่ง มีผลต่อพลังโจมตี
  - Vit ค่าความทนทาน มีผลต่อพลังป้องกันและพลังชีวิต
  - Int ค่าความฉลาด มีผลต่อความแรงของเวทมนต์
  - Agi ค่าความรวดเร็ว มีผลต่ออัตราการหลบหลีก
  - Mental สภาพของตัวละครในขณะนั้น
  - ToHit อัตราการโจมตีสำเร็จ
  - Money จำนวนเงินที่มี
- UserLogin - เก็บรายชื่อผู้เล่นที่กำลังออนไลน์อยู่ในขณะนั้น
  - TU_AID ID ของแอคเคาท์เจ้าของคาแรคเตอร์
  - TU_ID ยูสเซอร์เนมของคาแรคเตอร์
  - TU_IP IP ของคาแรคเตอร์ที่ล็อกอินใช้งาน
- CharInfo - เก็บรายละเอียดของตัวละครของผู้เล่นแต่ละคน
  - GID ID ของคาแรคเตอร์
  - Gname ชื่อของตัวละคร
  - Class อาชีพของตัวละคร
  - LV ระดับของตัวละคร
  - EXP ค่าประสบการณ์ที่มีอยู่
  - HP ระดับพลังชีวิตขณะนั้น
  - MP ระดับพลังเวทย์ในขณะนั้น

- MaxHP ระดับพลังชีวิตสูงสุด
- MaxMP ระดับพลังเวทย์สูงสุด
- ToHit อัตราการโจมตีสำเร็จ
- Str ค่าความแข็งแกร่ง มีผลต่อพลังโจมตี
- Vit ค่าความทนทาน มีผลต่อพลังป้องกันและพลังชีวิต
- Int ค่าความฉลาด มีผลต่อความแรงของเวทมนต์
- Agi ค่าความรวดเร็ว มีผลต่ออัตราการหลบหลีก
- Mental สภาพของตัวละครในขณะนั้น
- ToHit อัตราการโจมตีสำเร็จ
- Money จำนวนเงินที่มี
- AID ID แอคเคาท์เข้าของคาแรคเตอร์
- MapID ID ของแผนที่ที่ตัวละครอยู่ล่าสุด
- Xpos ตำแหน่งขณะนั้นของตัวละครในแกนตั้ง
- Zpos ตำแหน่งขณะนั้นของตัวละครในแกนนอน
- SXPos ตำแหน่งเริ่มต้นของตัวละครในแกนตั้ง
- SZPos ตำแหน่งเริ่มต้นของตัวละครในแกนนอน
- StartMap แผนที่ตั้งต้นของตัวละคร

* ทุกครั้งที่ระดับพลังของตัวละครเป็น 0 ตัวละครนั้นจะกลับมายังจุดเริ่มต้นใหม่อีกครั้ง

8.2.3 การออกแบบแพ็คเกจข้อมูลในการรับส่งข้อมูล

1. **Packet** เป็น Packet หลักที่กำหนดรูปแบบ Packet ของ Packet เอาไว้ว่าเป็นชนิดใด(type) และทำอะไร (action)

```
typedef struct sPacket{
  unsigned char type;
  unsigned char action;
}sPacket;
```

2. **PacketLogin** เป็น Packet ที่ใช้ส่งข้อมูลที่เป็นการเข้าเล่นเกม โดยจะส่งข้อมูลทั้ง username และ password ไปตรวจสอบ

```
typedef struct sPacketLogin  : public sPacket {
  char user[20];
  char pass[20];
}sPacketLogin;
```

3. **PacketPlayer** เป็น Packet ที่ใช้สำหรับบอกถึงสถานะของตัวละคร

```
typedef struct sPacketPlayer : public sPacket {
  char name[20];
  int HP;
  int HPMAX;
  int MP;
  int MPMAX;
  int EXP;
  int Level;
  int STR;
  int VIT;
  int INT;
  int Point;
}sPacketPlayer;
```

4. **PacketGetStatus** เป็น Packet ที่ใช้สำหรับส่งไปร้องขอให้ส่งสถานะของตัวละครมาให้

```
typedef struct sPacketGetStatus : public sPacket {
}sPacketGetStatus;
```

5. **PacketWalk** เป็น Packet ที่ใช้สำหรับส่งไปบอกให้ตัวละครเดินไปยังตำแหน่งที่กำหนด

```
typedef struct sPacketWalk : public sPacket {
  unsigned int ID;
  unsigned char xpos;
  unsigned char zpos;
  unsigned char xdest;
  unsigned char zdest;
  float direction;
}sPacketWalk;
```

6. **PacketCreateSpawnPlayer** เป็น Packet ที่ใช้สำหรับส่งไปบอกให้ไคลเอนต์สร้างตัวละครด้วยข้อมูลที่ส่งมา

```
typedef struct sPacketCreateSpawnPlayer : public sPacket {
  unsigned int ID;
  char name[20];
  unsigned char xpos;
```

```
    unsigned char zpos;
    float direction;
}sPacketCreateSpawnPlayer;
```

7. **PacketUpdateSpawnPlayer** เป็น Packet ที่ใช้สำหรับส่งข้อมูลไปบอกถึงตำแหน่งหรือสถานะปัจจุบันที่ตัวละครทำอยู่

```
typedef struct sPacketUpdateSpawnPlayer : public sPacket {
    unsigned int ID;
    unsigned char xpos;
    unsigned char zpos;
    float direction;
    unsigned char act;
    unsigned char ai;
    unsigned int  time;
}sPacketUpdateSpawnPlayer;
```

8. **PacketCreateMonster** เป็น Packet ที่ใช้สำหรับส่งไปบอกให้ไคลเอนต์สร้างศัตรูขึ้นมาในฉาก

```
typedef struct sPacketCreateMonster : public sPacket {
    unsigned int ID;
    unsigned char xpos;
    unsigned char zpos;
    float direction;
}sPacketCreateMonster;
```

9. **PacketUpdateMonster** เป็น Packet ที่ใช้สำหรับส่งข้อมูลไปบอกถึงตำแหน่งหรือสถานะปัจจุบันที่ศัตรูทำอยู่

```
typedef struct sPacketUpdateMonster : public sPacket {
    unsigned int ID;
    unsigned char xpos;
    unsigned char zpos;
    float direction;
    unsigned char act;
    unsigned char ai;
```

```
    unsigned int  time;
  }sPacketUpdateMonster;
```

10. **PacketRemoveSpawnPlayer** เป็น Packet ที่ใช้สำหรับที่ส่งไปบอกให้ไคลเอนต์ลบตัวละครออกจากเกม

```
  typedef struct sPacketRemoveSpawnPlayer : public sPacket {
    unsigned int ID;
  }sPacketRemoveSpawnPlayer;
```

11. **PacketRebornMonster** เป็น Packet ที่ส่งไปบอกให้ไคลเอนต์เซ็ตให้ศัตรูที่ตายเกิดขึ้นมาอีกครั้ง

```
  typedef struct sPacketRebornMonster : public sPacket {
    unsigned int ID;
  }sPacketRebornMonster;
```

12. **PacketToChar** เป็น Packet ที่ส่งไปบอกว่าผู้เล่นคลิกเลือกไปที่ตัวละครใด

```
  typedef struct sPacketToChar : public sPacket {
    unsigned int IDAttacker;
    unsigned int IDVictim;
  }sPacketToChar;
```

13. **PacketMessage** เป็น Packet ที่ใช้สำหรับบอกให้ไคลเอนต์แสดงข้อความเกี่ยวกับเหตุการณ์ของผู้เล่นเช่น ถูกโจมตีด้วยพลังเท่าใด ประสบการณ์เพิ่มขึ้นเท่าใด

```
  typedef struct sPacketMessage : public sPacket {
    unsigned int ID;
    D3DCOLOR color;
    long timer;
    char text[50];
  }sPacketMessage;
```

14. **PacketChat** เป็น Packet ที่ใช้สำหรับการสนทนา

```
  typedef struct sPacketChat : public sPacket {
    char name[20];
    char msg[70];
```

}sPacketChat;

8.2.4 การคำนวณหาจำนวนผู้เล่น

ถ้าให้ M เป็นจำนวนของศัตรู

N เป็นจำนวนของผู้เล่น

และให้ช่วงเวลาที่ใช้ในการรับส่งข้อมูลแต่ละครั้งโดยเฉลี่ยเป็น 100 ms และกำหนดไว้ว่าเครือข่ายที่ใช้กับเซิร์ฟเวอร์ไม่ได้ใช้งานเนตเวิร์กอย่างอื่นเลย

กรณี Worstcase ในการส่งข้อมูลแต่ละครั้ง

ชนิดของแพ็กเกต	*ขนาดของแพ็กเกต*	*จำนวนครั้งในการรับและส่ง*
Packet	2	0
PacketLogin	40+2	N
PacketPlayer	60+2	N
PacketGetStatus	0+2	N
PacketWalk	12+2	(M*N)+(N*N)+N
PacketCreateSpawnPlayer	30+2	N/A
PacketUpdateSpawnPlayer	16+2	N*N
PacketCreateMonster	10+2	N/A
PacketUpdateMonster	16+2	M*N
PacketRemoveSpawnPlayer	4+2	N/A
PacketRebornMonster	4+2	N/A
PacketToChar	8+2	(M*N)+(N*N)+N
PacketMessage	66+2	N
PacketChat	90+2	2*N*N

หมายเหตุ - N/A คือ มีผลต่อแบนวิดท์น้อยมาก

ตารางที่ 8-1การคำนวณแบนวิดท์ของเครือข่าย

จากตารางสามารถคำนวณได้ว่าในการรับส่งทุกๆ 100 ms จะใช้ข้อมูลแบนวิดท์ทั้งหมด

= 42*N + 62*N + 2*N + 14*((M*N) + (N*N) + N) + 18*N*N + 18*M*N + 10*((M*N) + (N*N) + N) + 68*N + 92*2*N*N

= (42+62+2+14+10+68)*N + (14+18+10+)*M*N + (14+18+10+184)*N*N

= 198*N + 42*M*N + 226*N*N

= 198N + 42MN + 226N2

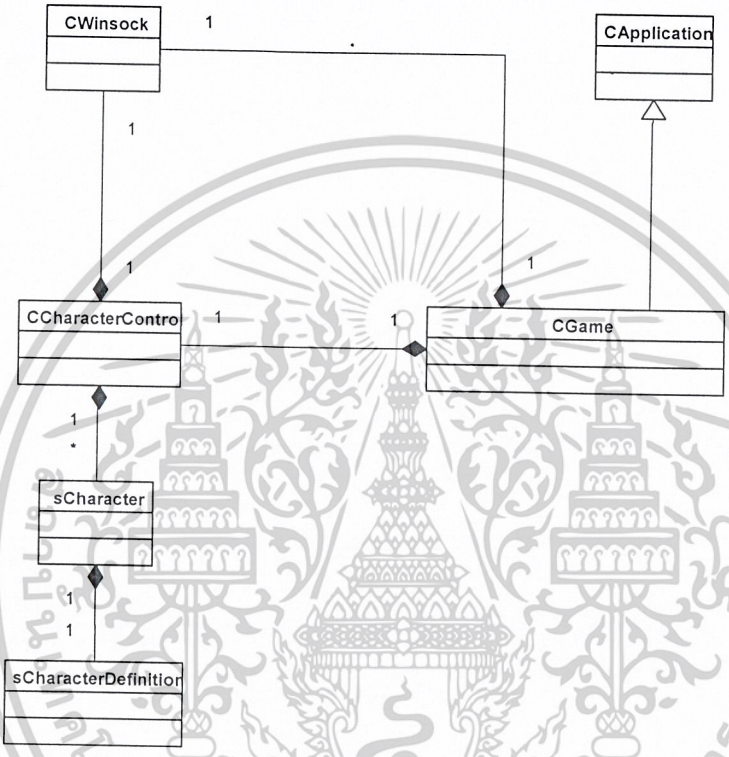
ดังนั้น ใน 1 วินาทีจะใช้แบนวิดท์ทั้งหมด

= 10*(198N + 42MN + 226N2) B/s

เมื่อเรารู้ว่าเครือข่ายมีความเร็วเท่าใด และ กำหนดว่าเกมมีศัตรูอยู่จำนวนเท่าใดสามารถคำนวณได้ว่าจำนวนผู้เล่นที่เหมาะสมในสำหรับเครือข่ายเป็นเท่าใดได้จาก

จำนวนผู้เล่น = ความเร็วของเครือข่าย(MB/s) / แบนวิดท์(B/s)

8.2.5 โครงสร้างคลาสของเซิร์ฟเวอร์

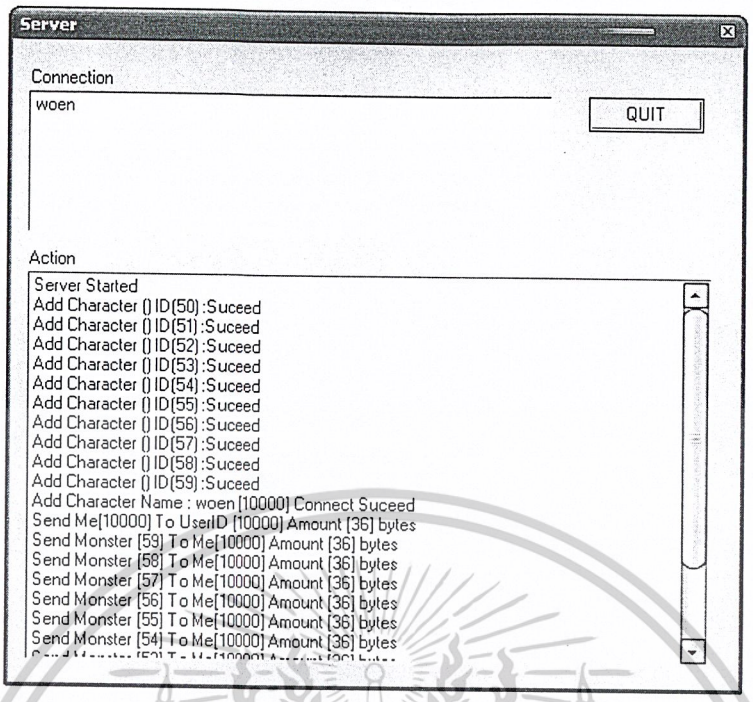


รูปที่ 8-2 โครงสร้างของคลาสบนเซิร์ฟเวอร์

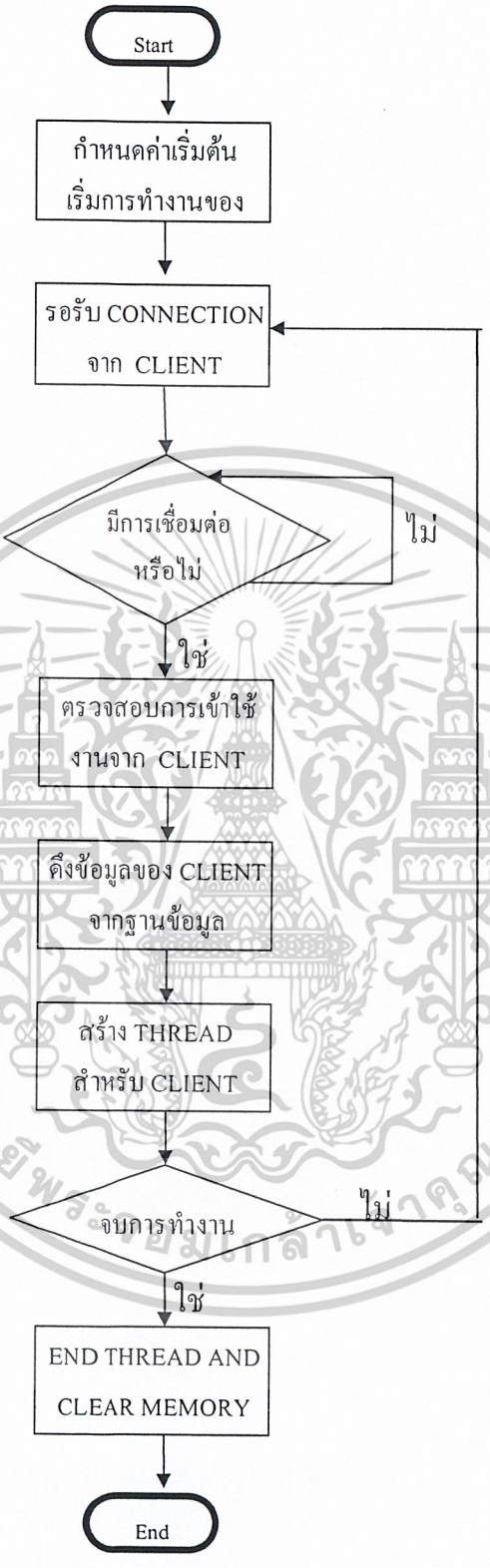
8.3 อินเตอร์เฟซของเซิร์ฟเวอร์

หน้าจอแสดงผลของเซิร์ฟเวอร์จะเป็นหน้าจอของวินโดว์โดยจะแสดงผลเฉพาะรายชื่อของผู้เล่นที่เชื่อมต่ออยู่ในขณะนั้น และรายการดำเนินการของเซิร์ฟเวอร์ที่ผ่านไป รายการนี้จะประกอบด้วยข้อความแสดงการเริ่มต้นทำงานของเซิร์ฟเวอร์ การเชื่อมต่อจากไคลเอนท์ การรับส่งข้อมูลจากไคลเอนท์แต่ละเครื่องโดยแสดงหมายเลขระบุไคลเอนท์ว่าเป็นของเครื่องใด และข้อความแจ้งเมื่อไคลเอนท์จบการเชื่อมต่อกับเซิร์ฟเวอร์ ข้อความในส่วนนี้จะมีสครอลบาร์ช่วยให้ผู้ดูแลระบบสามารถเลื่อนขึ้นไปดูรายละเอียดการทำงานของเซิร์ฟเวอร์ที่ผ่านไปแล้วได้

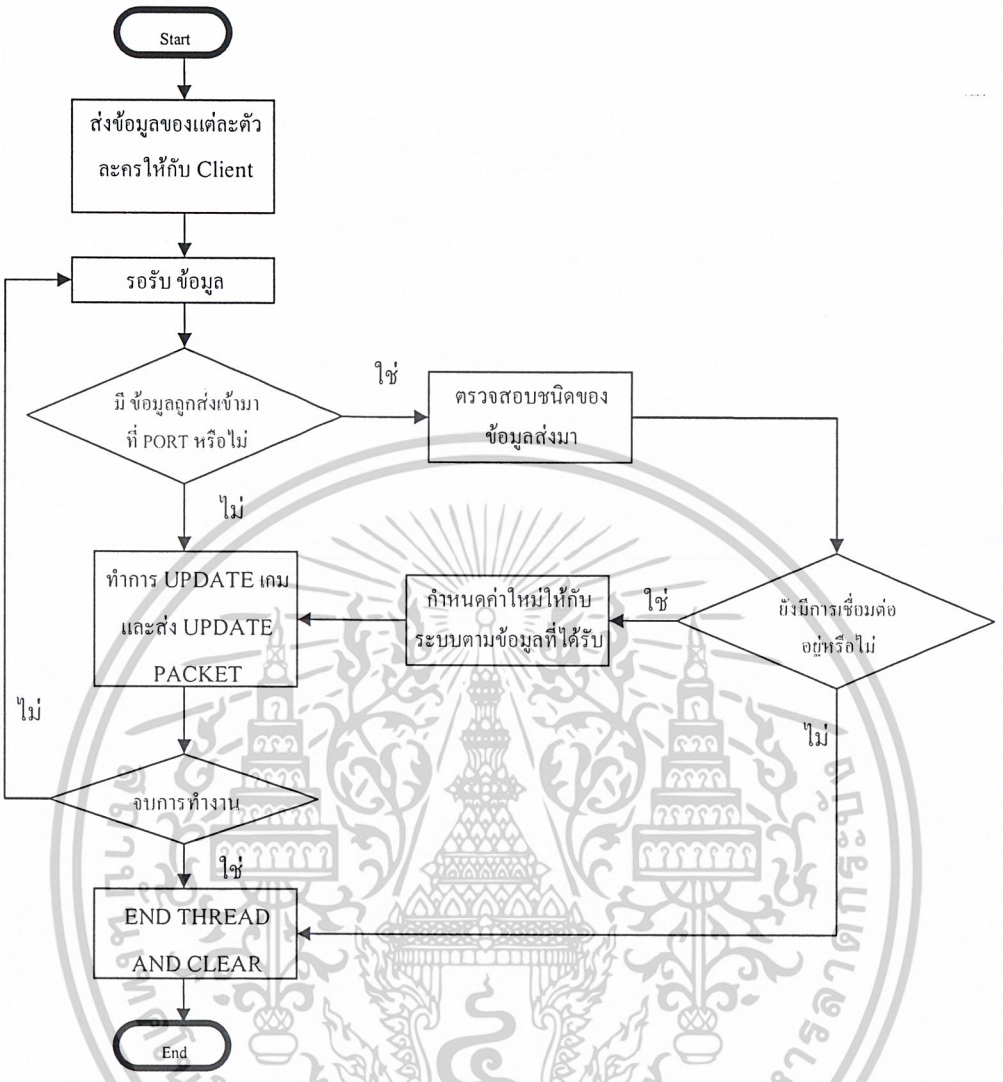
นอกจากเท็กซ์บ๊อกที่แสดงการทำงานของเซิร์ฟเวอร์แล้ว ในหน้าจอจะมีปุ่มที่ใช้ปิดเซิร์ฟเวอร์อยู่ที่มุมบนขวา ถ้าหากปิดการทำงาน ข้อความที่อยู่ในเท็กซ์บ๊อกจะถูกจัดเก็บไว้ในล๊อกไฟล์เพื่อให้ผู้ดูแลระบบสามารถเรียกขึ้นมาดูได้ภายหลัง



รูปที่ 8-3 หน้าจอแสดงผลบนเซิร์ฟเวอร์



รูปที่ 8-4 แผนภาพแสดงขั้นตอนการทำงานของเซิร์ฟเวอร์



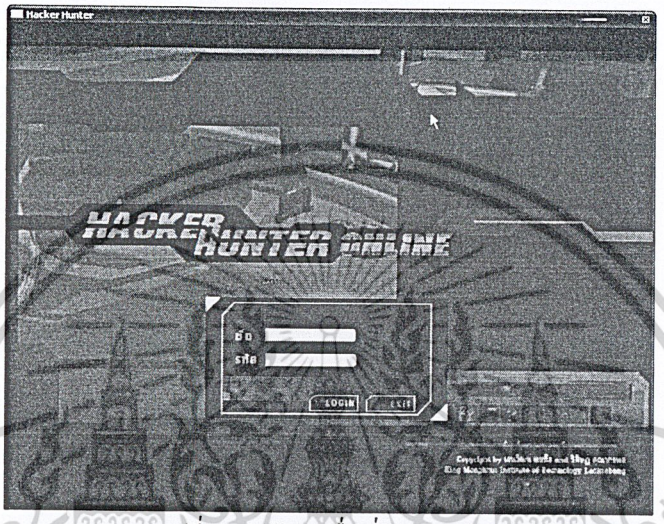
รูปที่ 8-5 แผนภาพแสดงขั้นตอนการทำงานของเซิร์ฟเวอร์เมื่อจัดการกับผู้ใช้

บทที่ 9
ผลการวิจัย

9.1 ผลการทดสอบโปรแกรม

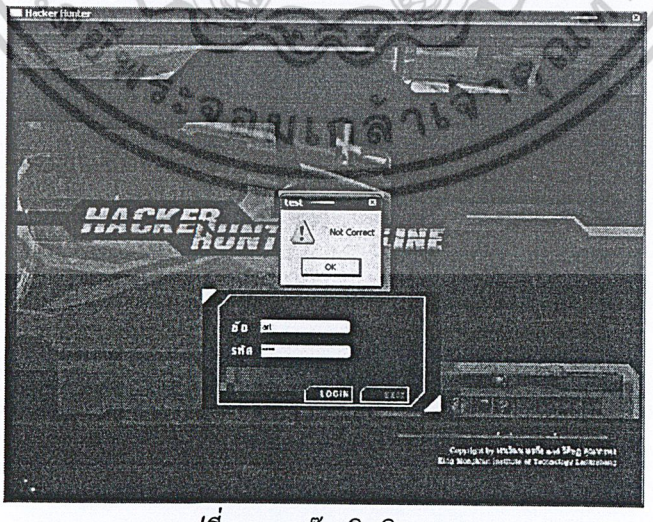
9.1.1 การทดสอบไคลเอนท์

เมื่อโปรแกรมเริ่มทำงาน โปรแกรมจะโหลดทรัพยากรทั้งหมดก่อนแล้วจึงแสดงหน้าจอล็อกอินให้ผู้ใช้กรอกชื่อและรหัสเพื่อเชื่อมต่อกับเซิร์ฟเวอร์



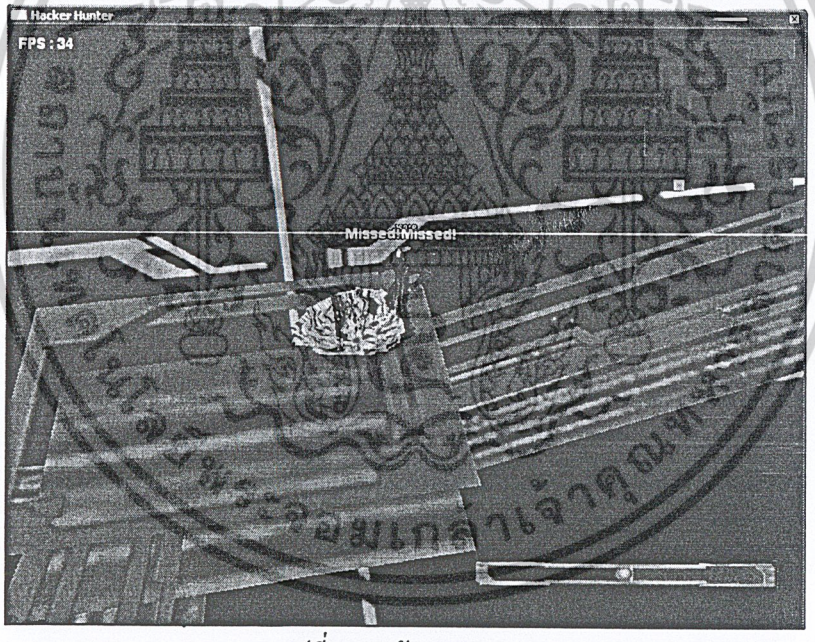
รูปที่ 9-1 หน้าจอเมื่อเริ่มการทำงาน

ถ้าหากยูสเซอร์กรอกรหัสผิดพลาดก็จะมีการแจ้งเตือนขึ้นมา โดยจะแจ้งเตือนว่า Not Connect และบังคับให้ผู้ใช้ต้องกรอกรหัสใหม่อีกครั้ง ถ้าหากชื่อและรหัสถูกต้อง โปรแกรมก็จะสร้างการเชื่อมต่อกับเซิร์ฟเวอร์ ดึงข้อมูลล่าสุดของผู้เล่นออกมาจากฐานข้อมูลเพื่อนำมาใช้เป็นค่าตั้งต้นของเกม เมื่อเสร็จเรียบร้อยแล้วก็จะรอคำสั่งจากผู้เล่นต่อไป

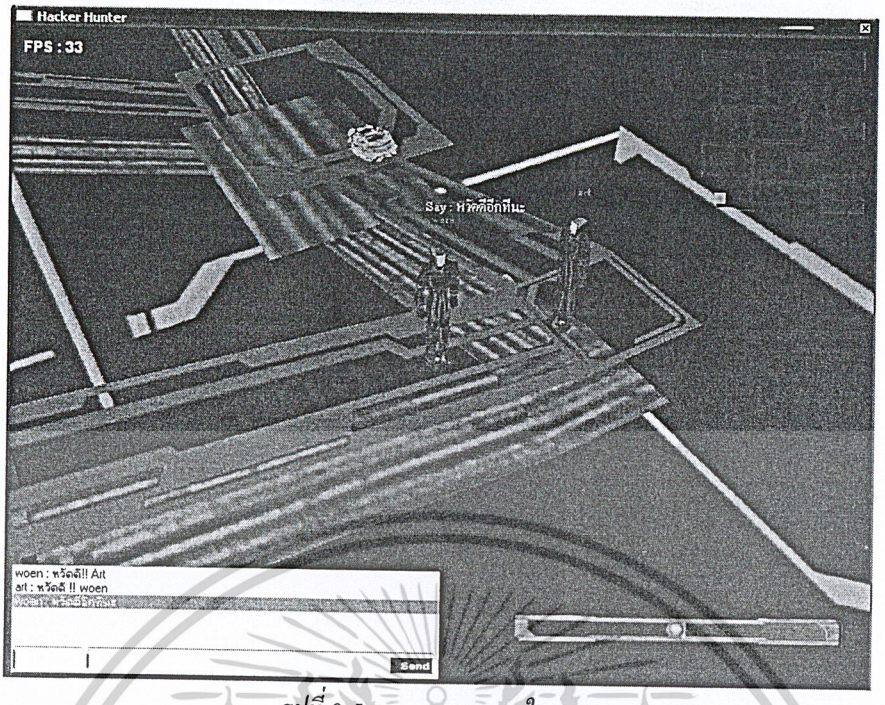


รูปที่ 9-2 การล็อกอินผิดพลาด

รูปที่ 9-3 หน้าจอระหว่างที่ทำการโหลดข้อมูลตั้งต้น



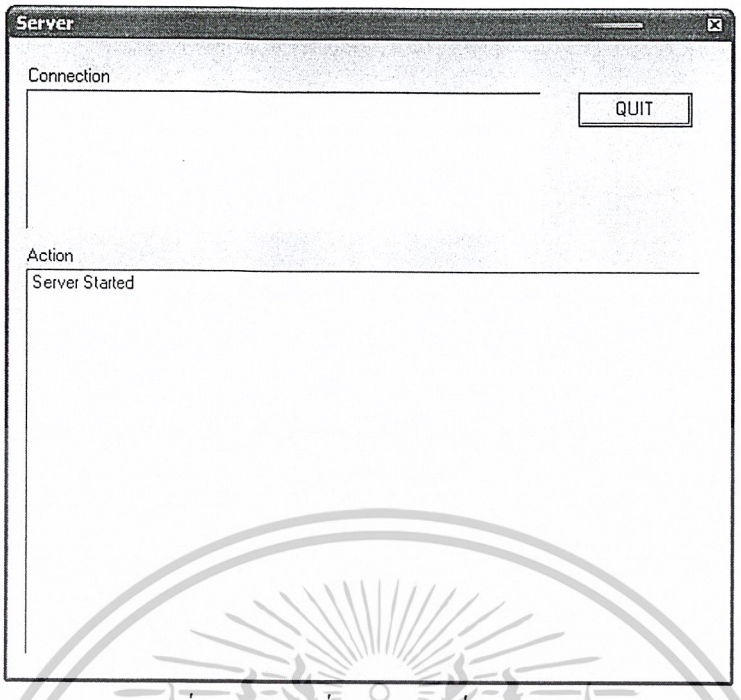
รูปที่ 9-4 หน้าจอของเกม



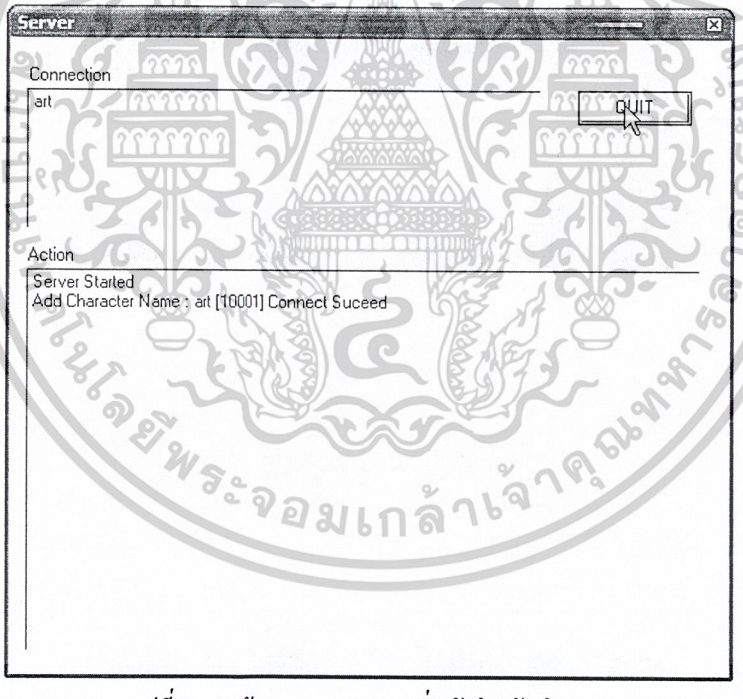
รูปที่ 9-5 การสนทนาภายในเกม

9.1.2 การทดสอบเซิร์ฟเวอร์

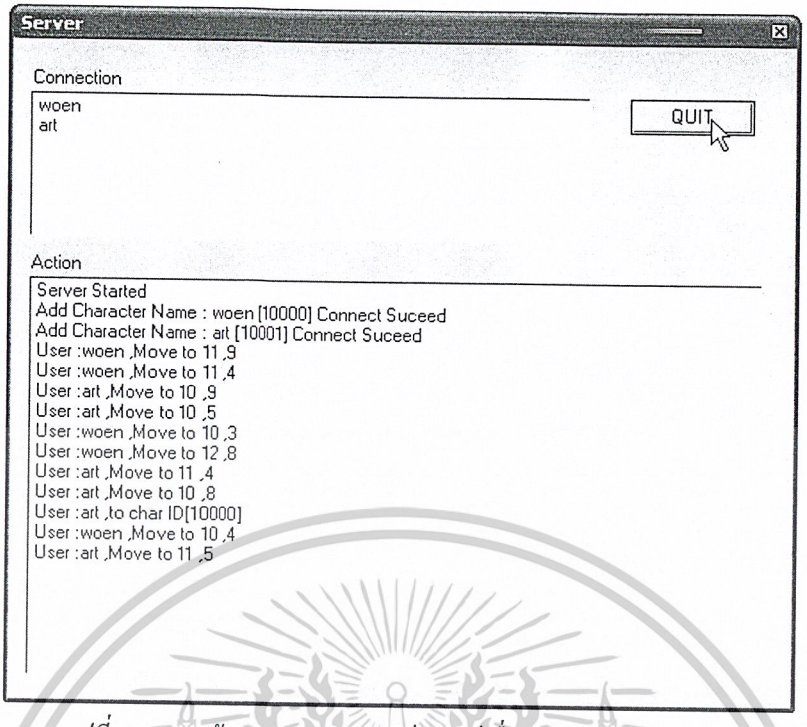
เมื่อเซิร์ฟเวอร์เริ่มทำงาน โปรแกรมจะแสดงผลหน้าจอดังรูป หลังจากที่ตั้งค่าพอร์ตและสร้าง Environment ของเกมเรียบร้อยแล้ว โปรแกรมจะเริ่มรอรับการเชื่อมต่อจากไคลเอนท์ เมื่อมีการเชื่อมต่อหรือมีการรับข้อมูลเข้ามา ก็จะตรวจสอบการเข้าเล่นเกมจากฐานข้อมูล ถ้าหากถูกต้องก็จะแจ้งเตือนขึ้นบนหน้าจอ พร้อมทั้งระบุว่า มีผู้เล่นชื่ออะไรเชื่อต่ออยู่ในขณะนี้ หรือขณะนี้เซิร์ฟเวอร์กำลังทำอะไรให้กับไคลเอนท์ตัวไหนบ้าง โดยจะมีรายละเอียดเกี่ยวกับการเดินบนแผนที่ โดยจะระบุตำแหน่งของตัวละครบนแผนที่ หรือถ้ามีการโจมตีก็จะระบุว่าใครโจมตีใคร หรือมีการส่งข้อความถึงกันก็จะระบุว่ามีใครส่งข้อความอะไรถึงใคร



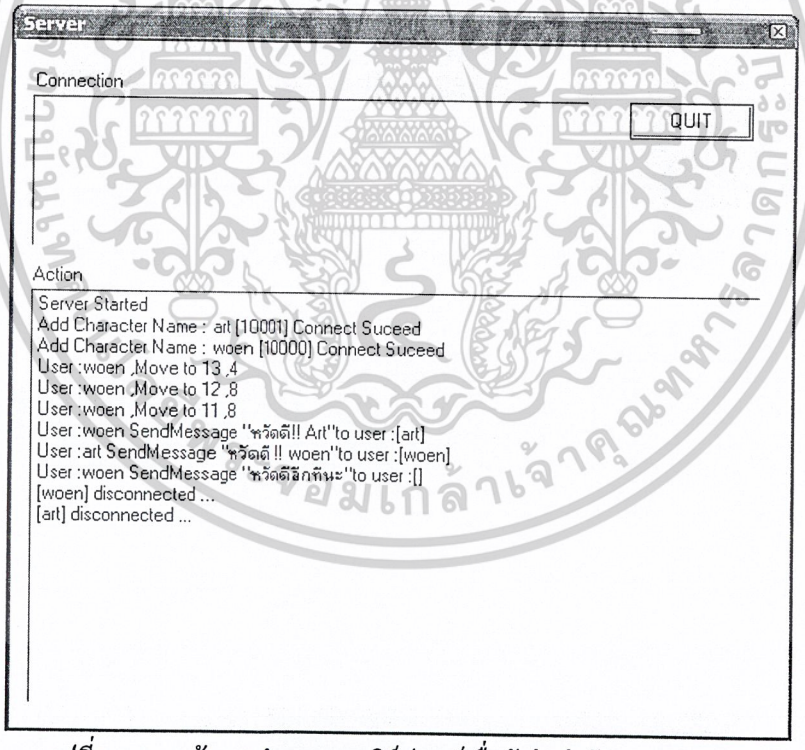
รูปที่ 9-6 หน้าจอเมื่อเซิร์ฟเวอร์เริ่มการทำงาน



รูปที่ 9-7 หน้าจอขณะทำการเพิ่มผู้เล่นเข้าสู่ระบบ



รูปที่ 9-8 การแจ้งการทำงานของเซิร์ฟเวอร์เมื่อผู้เล่นคลิกเดินบนฉาก



รูปที่ 9-9 การแจ้งการทำงานของเซิร์ฟเวอร์เมื่อผู้เล่นส่งข้อมูลสนทนามา

9.2 สรุปผลการทดสอบ

9.2.1 การกำหนดความต้องการ

ความต้องการของระบบที่ได้ออกแบบไว้จะมีเฉพาะฟังค์ชั่นการทำงานหลักๆของเกมออนไลน์ เช่น การทำ Synchronization หรือ การจัดการไคลเอนท์แบบ Non-Blocking ทำให้ได้เซิร์ฟเวอร์ที่ทำงานได้ใกล้เคียงกับเกม MMORPG จริงๆที่อยู่ในท้องตลาด ส่วนไคลเอนท์ที่ทำการพัฒนาขึ้นมีความสามารถครบถ้วนเช่นกัน โดยสามารถแสดงผลเป็นกราฟิคสามมิติและสามารถรับส่งข้อมูลกับเซิร์ฟเวอร์ได้

9.2.2 การวิเคราะห์และออกแบบ

การวิเคราะห์และออกแบบระบบจะยึดหลักของ Object Orient เป็นหลักทำให้วิเคราะห์ความต้องการและสามารถออกแบบได้อย่างรวดเร็ว ครอบคลุมความต้องการขั้นต้นได้เกือบทั้งหมด ระบบที่ได้สามารถปรับปรุงหรือเพิ่มเติมได้ทำให้มีความยืดหยุ่นพอสมควร

แต่เนื่องจากผู้พัฒนาขาดประสบการณ์ในการพัฒนาหแอพลิเคชั่นขนาดใหญ่ ทำให้ฟังค์ชั่นการทำงานที่ออกแบบไว้บางอย่างไม่สามารถใช้งานได้จริง จึงเสียเวลาบางส่วนไปกับการปรับแก้ต้นแบบและศึกษาความรู้เพิ่มเติมและพัฒนาส่วนที่ขาดไปขึ้นมาให้สมบูรณ์

9.2.3 การสร้างระบบ

สามารถสร้างระบบเกมที่มีความสมบูรณ์ใกล้เคียงกับจุดประสงค์ที่ตั้งเอาไว้โดยแยกเป็นส่วนๆได้ดังนี้

1. การเรียกใช้ไลบรารีพื้นฐานของไดเร็คเอ็กบนไคลเอนท์ทำได้แต่ยังไม่ครบถ้วนตามจุดประสงค์ เนื่องจากขาดความรู้และความเข้าใจอย่างเพียงพอ เช่น ไม่สามารถลดค่า Opacity ของเท็กซ์เจอร์บนวัตถุสามมิติ การทำอนิเมชั่นไม่สามารถควบคุมแต่ละเฟรมได้โดยตรง
2. การปรับตัวโปรแกรมให้ไปอยู่บนระบบ COM ทำได้ยาก เพราะอินเตอร์เฟซของโปรแกรมที่ออกแบบไว้รองรับการทำงานของ COM ได้ไม่สมบูรณ์ บางคลาสต้องใช้อินเตอร์เฟซร่วมกันทำให้เวลาปรับเป็น COM แล้วไม่สามารถทำงานได้
3. การแสดงผลบนหน้าจออยู่ในระดับที่น่าพอใจ เฟรมเรทขณะตัวละครเคลื่อนที่มากพอที่สายตาจะไม่สังเกตเห็นอาการกระตุก การโหลดโมเดลและเท็กซ์เจอร์ทำได้สมบูรณ์ แต่พบปัญหาเกี่ยวกับการเรียกใช้งานภาษาไทยในการแสดงผลบนหน้าจอ เนื่องจากการเรียกใช้งานภาษาไทยจะต้องทำผ่านคอมโพเนนต์ของวินโดว์ ทำให้เรียกใช้งานอินเตอร์เฟซต่างๆก่อนจึงจะใช้ได้ซึ่งเป็นการซับซ้อนและทำให้โปรแกรมไม่มีประสิทธิภาพเท่าที่ควร
4. ฐานข้อมูลที่ออกแบบไว้ครอบคลุมกับความต้องการของระบบ เนื่องจากเกม MMORPG จะต้องการความรวดเร็วในการทำงานเป็นหลัก ดังนั้นการออกแบบฐานข้อมูลจึงไม่อาจยึดตามทฤษฎีได้เต็มที่ ต้องมีการเพิ่มเติมบางส่วนที่ทำให้เกิดความซ้ำซ้อนขึ้นเพื่อให้สามารถดึงข้อมูลได้อย่างรวดเร็ว

9.2.4 การทดสอบระบบ

โดยภาพรวมสามารถสร้างเกมออนไลน์แบบ MMORPG ขึ้นมาได้โดยมีฟังค์ชั่นการทำงานหลักๆ ที่สามารถใช้การได้ครบถ้วน คือเกมสามารถรองรับผู้เล่นได้หลายคนโดยต้องมี ID เป็นของตัวเองเพื่อใช้เข้ามาเล่นเกมและมีตัวละครให้ สามารถที่ต่อสู้กับศัตรูเพื่อเก็บประสบการณ์เพื่อพัฒนาตัวละครให้เก่งขึ้น และสามารถที่จะสนทนากันภายในเกมได้ แต่เนื่องจากขาดแรงงานและเวลาในการพัฒนาทำให้รายละเอียดที่เป็นความสมบูรณ์เช่น เนื้อเรื่อง หรือ แผนที่ต่างๆ ทำได้ไม่มากนัก อีกทั้งทรัพยากรที่จำกัดทำให้ไม่สามารถทดสอบการเชื่อมต่อเป็นจำนวนมากในสถานการณ์จริงได้

บทที่ 10
วิจารณ์และสรุปผล

10.1 ปัญหาที่พบ

ในด้านการทำกราฟิค การทำโมเดลจะมีข้อควรระวังที่ต้องตั้งให้ Pivot(จุดตั้งต้นของโมเดล) อยู่ในตำแหน่ง(0,0,0) ตลอดเวลาและจำต้องเปลี่ยนค่าที่ใช้ในการย่อขยายขนาดให้เป็น 1/1 อยู่เสมอ ถ้าหากไม่ทำเช่นนี้แล้วโปรแกรมจะไม่สามารถคำนวณขนาดที่แท้จริงของวัตถุได้ การทำอนิเมชันบางครั้งทำให้โมเดลเกิดการบิดหรือเพี้ยนไปจากต้นแบบ เนื่องจากความผิดพลาดในการ Map ตัวละครเข้ากับ Bone ที่ใช้ในการควบคุมอนิเมชัน

การเชื่อมต่อระหว่างกราฟิกและโปรแกรมยังมีปัญหาค่อนข้างมาก เช่น เมื่อทำการออก แบบเสร็จแล้วนำไปใช้งานจะเกิดการผิดเพี้ยนของขนาดสี เมื่อมีการรวมโค้ดต่างๆเข้าด้วยกัน ด้านของการเขียนโปรแกรมยังมีปัญหาในการโหลดไฟล์ที่มีขนาดใหญ่จะทำได้ค่อนข้างช้าในเครื่องที่มีสมรรถนะของเครื่องค่อนข้างต่ำ และยังมีปัญหาในการคำนวณจุดต่าง ๆ ในระนาบสามมิติ เพื่อที่จะนำไปใช้งานนั้น เกิดความคลาดเคลื่อนค่อนข้างมาก ทำให้รูปเกิดความผิดเพี้ยนในบ้าง และเนื่องจากใช้ตัวละครแบบ Xfile ทำให้มีปัญหาในการเลือก frame ของแต่ละ action เนื่องจากไม่ได้เลือกเป็น frame แต่จะเลือกเวลาของแต่ละ frame จึงเป็นปัญหาสำหรับการเลือก action

ในส่วนของเซิร์ฟเวอร์จะมีปัญหาในเรื่องของการรับส่งข้อมูลปริมาณมากๆในช่วงเวลาน้อยๆ ซึ่งการจัดการเกี่ยวกับการรับส่งข้อมูลจะต้องมีบัฟเฟอร์เก็บข้อมูลไว้ก่อนที่จะส่งข้อมูล จึงทำได้ยาก อีกทั้งยังมีการจัดการกับไคลเอนท์ด้วยเทรด (Thread) ทำให้เกิดปัญหาเกี่ยวกับการเข้าใช้ข้อมูลตัวเดียวกันอีกด้วย

10.2 สรุปผลการวิจัย

1. การพัฒนาเอนจินเกมบนไดเร็คเอ็กสามารถทำได้
2. สามารถเขียนโปรแกรมติดต่อและจัดการเกี่ยวกับการส่งข้อมูลผ่านระบบเครือข่ายได้ระดับหนึ่ง
3. สามารถโหลดและแสดงผลโมเดลสามมิติที่สร้างไว้ได้ อนิเมชั่นแสดงออกมาอย่างถูกต้อง
4. เซิร์ฟเวอร์สามารถติดต่อกับฐานข้อมูลเพื่อนำข้อมูลมาใช้งาน และเพื่อเก็บข้อมูลของผู้เล่นได้
5. สามารถจัดการแบ่งการทำงานให้กับผู้เล่นแต่ละคนด้วยการใช้เทรดได้
6. ผู้เล่นสามารถสนทนากันภายในเกมได้ ไม่ว่าจะเป็นการบอร์ดแคสหรือส่งให้ผู้เล่นคนอื่นโดยตรง

10.3 แนวทางการพัฒนาต่อ

1. เพิ่มระบบรักษาความปลอดภัยในส่วนจัดการระบบเครือข่ายของเซิร์ฟเวอร์ เช่น การเข้ารหัสข้อมูลก่อนที่จะทำการส่ง
2. ปรับปรุงประสิทธิภาพด้วยการเพิ่มส่วนของ Load sharing และการทำ Zone เพื่อเพิ่มประสิทธิภาพของเซิร์ฟเวอร์
3. นำชุดคำสั่งที่ได้รวบรวมไปพัฒนาโดยสร้างเป็นไลบรารีเกมเอนจินต่อไป
4. เพิ่มรายละเอียดของเกมเพื่อให้สมบูรณ์ยิ่งขึ้น เช่น เนื้อเรื่องและฉากใหม่ๆ
5. จัดการเกี่ยวกับระบบการรับส่งข้อมูลให้มีความถูกต้องและไว้วางใจได้ เมื่อผู้เล่นเชื่อมต่อเกมด้วยโมเด็ม

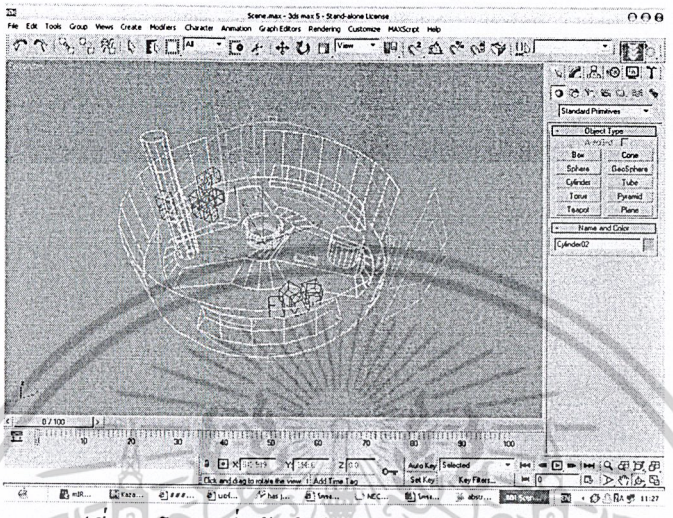
ภาคผนวก ก
แหล่งข้อมูลเพิ่มเติม

- http://www.gamedev.net เป็นแหล่งค้นหาข้อมูลสำหรับการโหลดตัวละครภายในเกมด้วยไฟล์ XFILE ซึ่งจะมีรายละเอียดของซอร์สโค้ด ที่จะโหลดตัวละครเข้าไปในเกม
- http://www.gametutorials.com เป็นแหล่งข้อมูลสำหรับผู้ที่จะเขียนเกมด้วย ไดเร็คเอ็ก และ Opengl ซึ่งจะมีตัวอย่างการเขียนให้ด้วยมากมาย
- http://www.cwinapp.com เป็นแหล่งข้อมูลสำหรับผู้เริ่มต้นเขียนเกมด้วย ไดเร็คเอ็ก เพราะจะมีการสอนเกี่ยวกับการเขียนเกมเบื้องต้นสำหรับการเขียนเกมด้วย ไดเร็คเอ็ก
- http://www.thaigamedevx.com เป็นเวบไซท์สำหับผู้เขียนเกมจะมีการถามตอบเกี่ยวกับเกมให้ผู้มีข้อสงสัยตั้งกระทู้ถามตอบกันได้
- http://www.gamasutra.com เป็นเวบไซท์อีกเวบไซท์หนึ่งที่มีบทความให้ผู้สนใจที่จะทำเกมได้มาศึกษาถึงแนวคิดในการทำเกม
- http://tangentsoft.net เป็นเวบไซท์ที่ให้ความรู้สำหรับนักพัฒนาโปรแกรมโดยใช้ Winsock โดยจะมีเรื่อง บทความ และตัวอย่างเกี่ยวกับการเขียนโปรแกรมอีกด้วย
- http://www.hal-pc.org/~johnnic2/winsock.html เป็นเวบไซท์ที่สอนการเขียนโปรแกรมเบื้องต้นด้วย Winsock

ภาคผนวก ข

ขั้นตอนการสร้างโมเดล

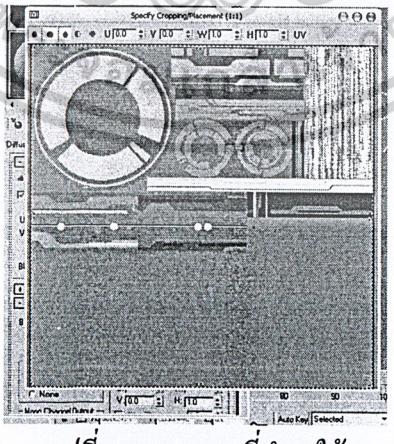
การสร้างฉากจะขึ้นโมเดลในโปรแกรม 3D Studio Max จากนั้นจึงใส่ Texture และตรวจเช็คขนาดก่อนจะนำไปทำอนิเมชั่นอีกที



รูปที่ ข-1 โมเดลที่สร้างในโปรแกรม 3D studio Max 5

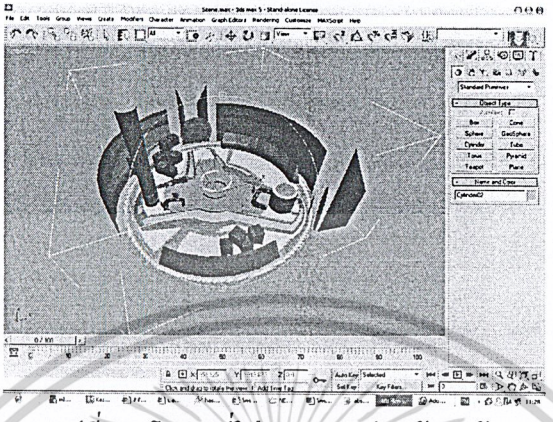
ในขั้นตอนแรกจะต้องกำหนดรูปร่างของโมเดลโดยใช้ Line วาดรูปของฉากขึ้นมาคร่าวๆก่อน การวาดจะต้องระวังให้มีแต่รูปสี่เหลี่ยมหรือสามเหลี่ยมเท่านั้นเนื่องจากโปรแกรม 3Dmax จะถือว่าเส้นขอบเขตที่มีมุมไม่เกินสี่มุมเท่านั้นที่สามารถนำมาคำนวณเป็นโพลีกอนหรือเป็นระนาบได้

การวาดจะเริ่มจากฐานก่อนแล้วจึงค่อยใส่ส่วนสูงให้กับฐานดังกล่าวจนเป็นรูปสามมิติที่สมบูรณ์ในที่สุด ขั้นตอนการวาดนี้เราสามารถใช้ออปเจ็คพื้นฐานที่มีมากับโปรแกรมวาดรูปทรงง่ายๆ เช่น ทรงกลมหรือทรงกระบอก ได้เลยทันที หลังจากวาดเสร็จแล้วจะต้องทำการรวมแต่ละออปเจ็คที่ใช้พื้นผิวเดียวกันให้กลายเป็นออปเจคเดียวกันเพื่อความสะดวกในการใส่เท็กซ์เจอร์ต่อไป



รูปที่ ข-2 Texture ที่นำมาใช้

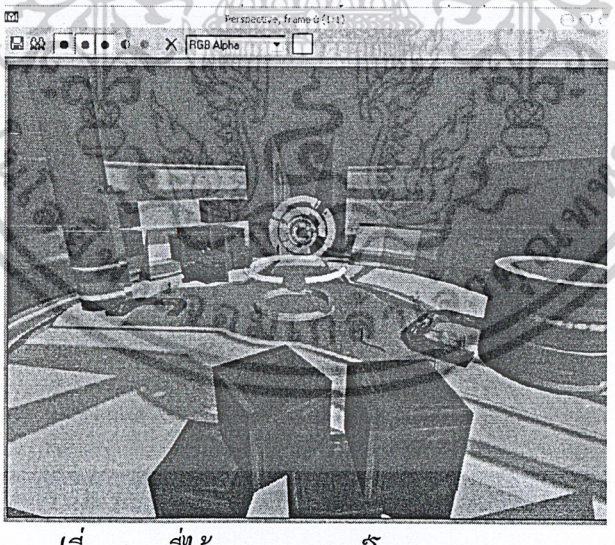
จากตัวอย่างเท็กเจอร์จะเห็นได้ว่ามีการรวมเอารูปต่างๆที่จะใช้เป็นระนาบของออปเจ็คไว้เป็นรูปเดียวกัน ทั้งนี้เพื่อการประหยัดเวลาในการโหลดข้อมูลรูปให้เสียเวลาโหลดไฟล์เดียวแทนที่จะต้องวนรอบโหลดทีละไฟล์ซึ่งเสียเวลาอย่างมาก ถ้าหากต้องการให้ส่วนไหนใสไม่มีสีก็สามารถกำหนดค่าสีให้ตรงกับในเอนจิน เช่น 255, 0, 255 ซึ่งเป็นค่าสีที่พบได้ยากในรูปทั่วไป



รูปที่ ข-3 โมเดลที่ใส่ Texture เรียบร้อยแล้ว

หลังจากนั้นจึงใช้ฟังค์ชั่น UVWmap ของโปรแกรม3Dmax กำหนดขอบเขตต่างๆของออปเจ็คว่าจะให้ตรงกับพื้นผิวใด ขั้นตอนนี้จะกินเวลานานและต้องการความถูกต้องสูงเพื่อโมเดลที่ได้จะมีความถูกต้องตรงกับที่ต้องการ เมื่อ map ค่าดังกล่าวเสร็จสิ้น โมเดลจะมีพื้นผิวและมีความเป็นวัตถุสามมิติมากขึ้น

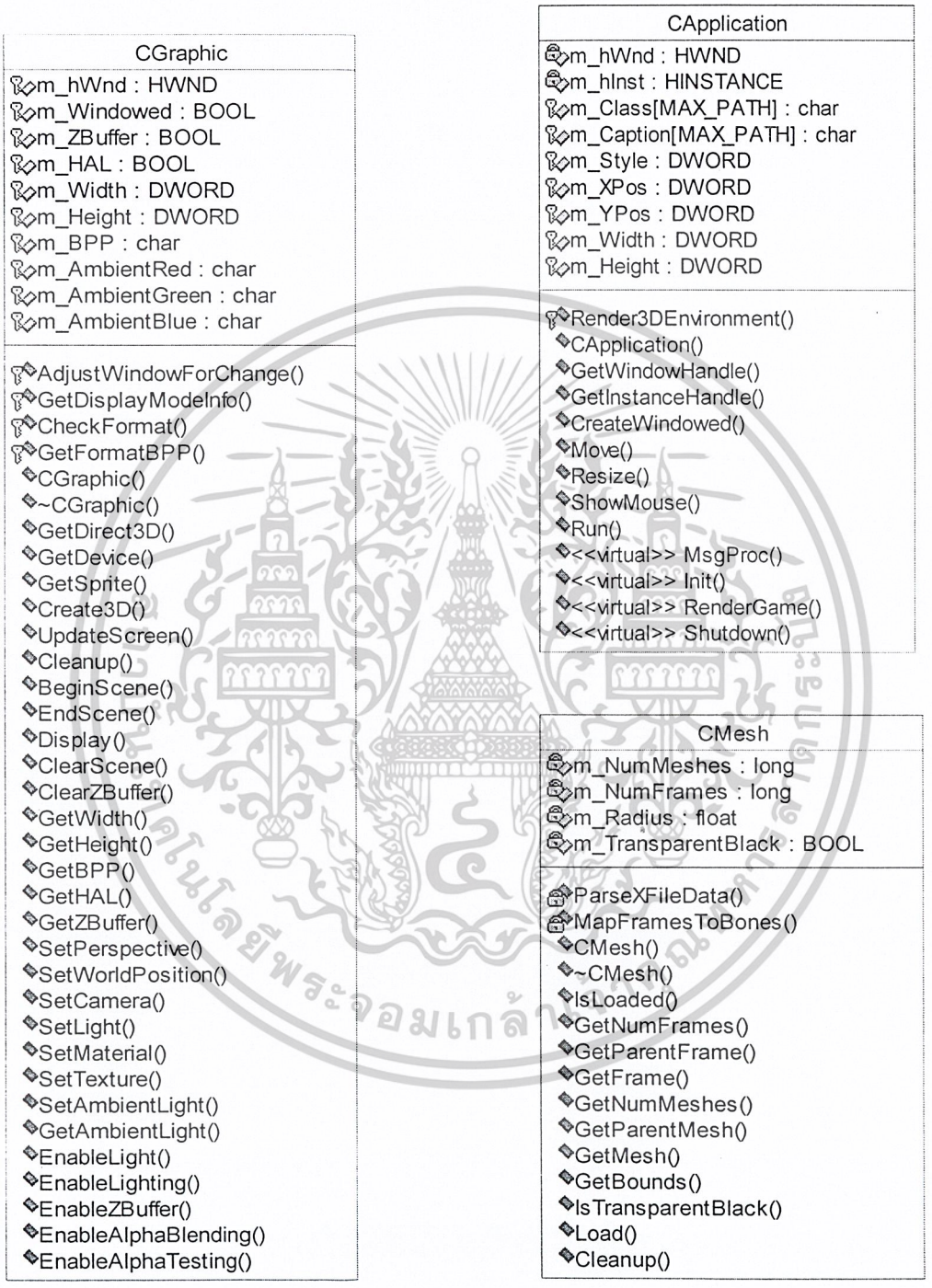
หลังจากนั้นจึงนำไปทำอนิเมชั่นเพิ่มเติมเพื่อเพิ่มการเคลื่อนไหวแล้วจึง Export ออกมาเป็นไฟล์ .X ซึ่งเอนจินจะนำไปใช้งานได้ต่อไป

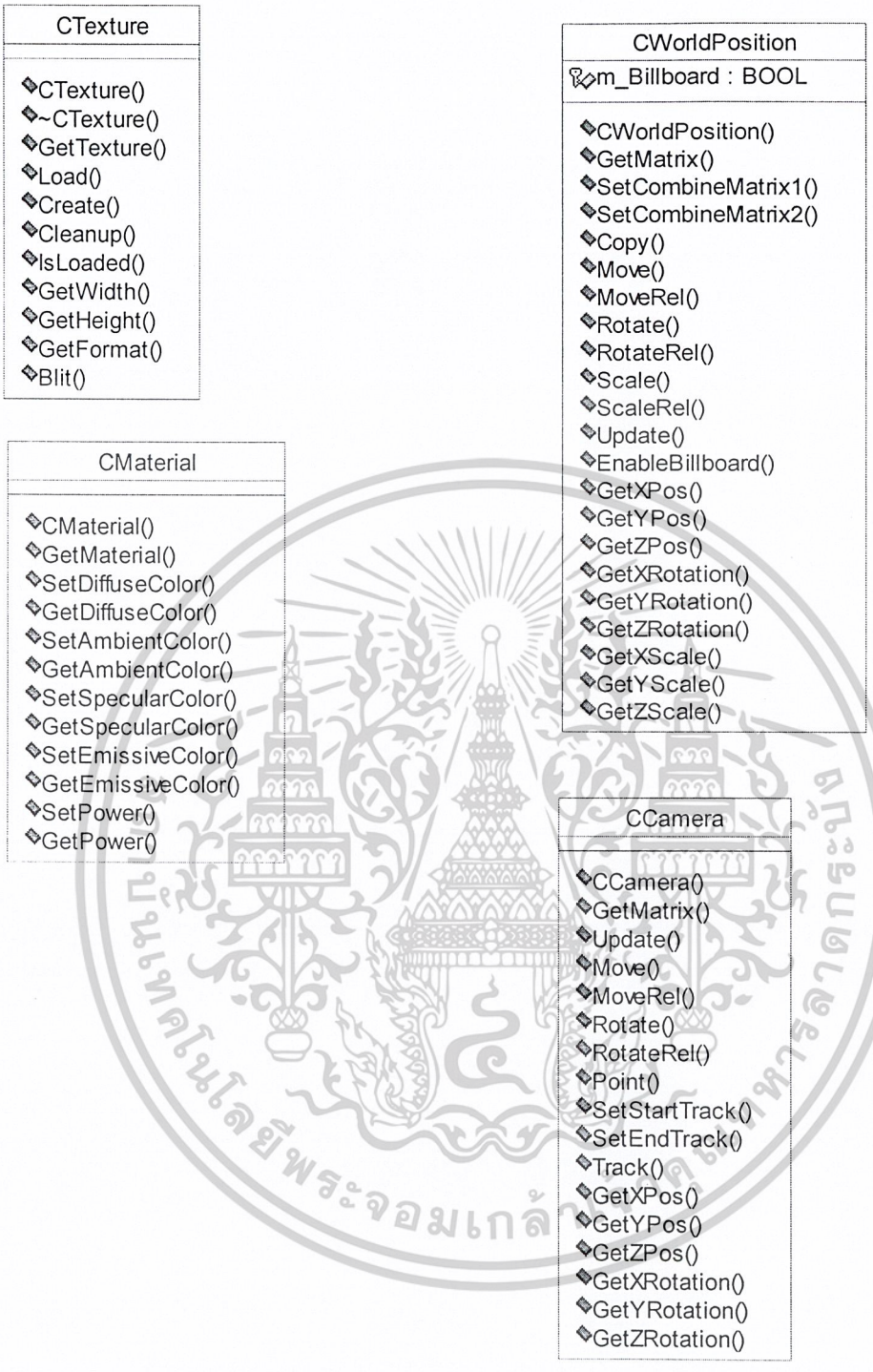


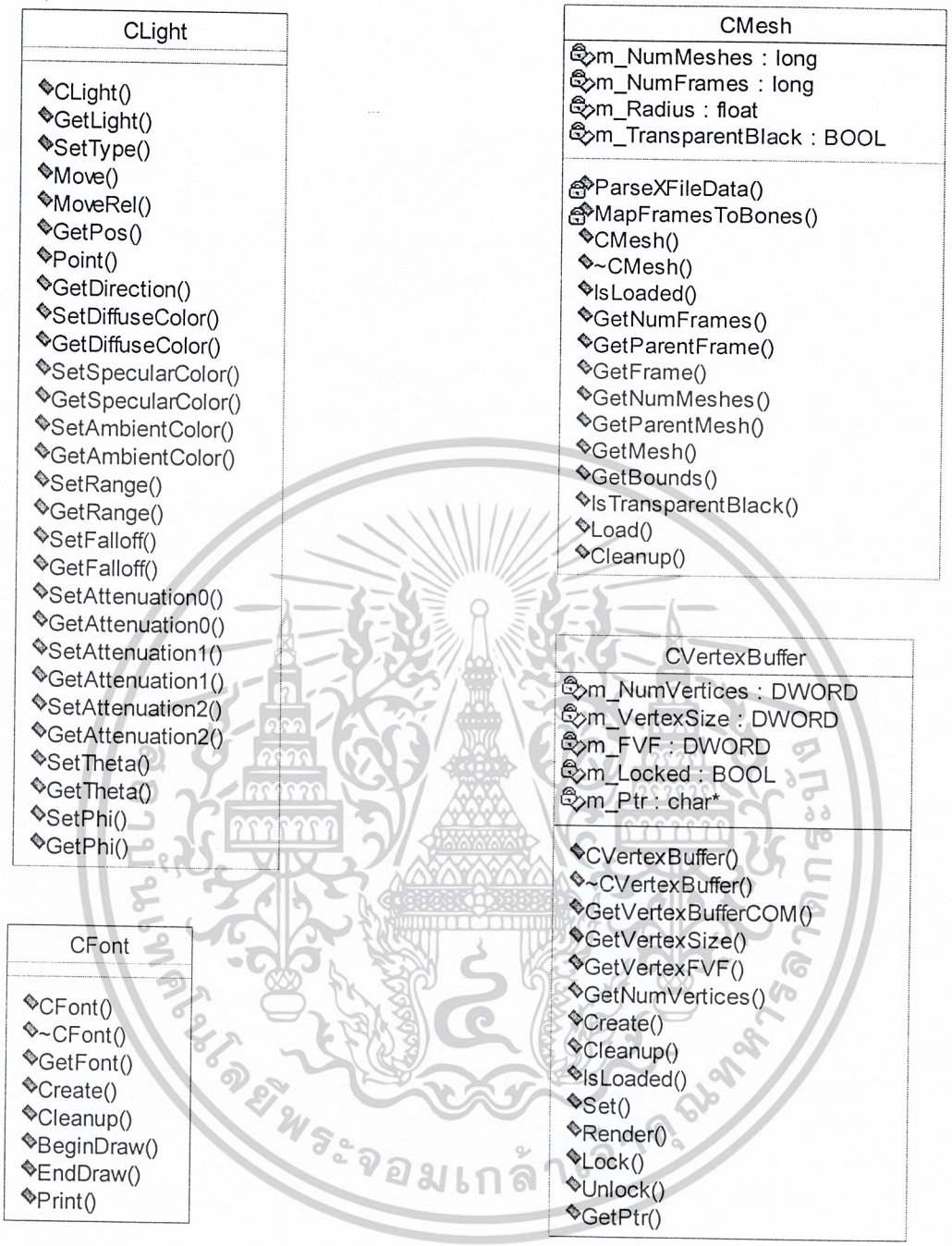
รูปที่ ข-4 ผลที่ได้จากการเรนเดอร์โดย 3D studio Max

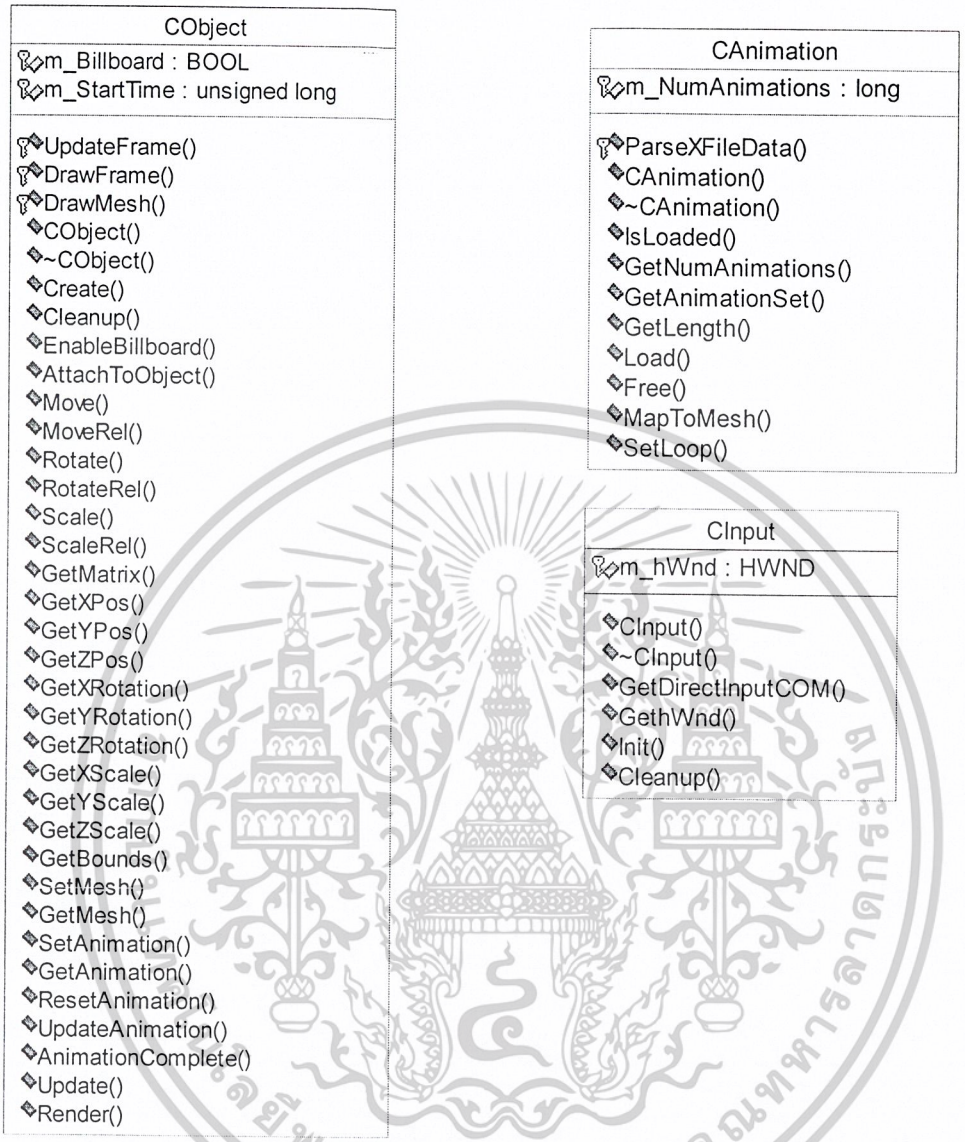
ภาคผนวก ค

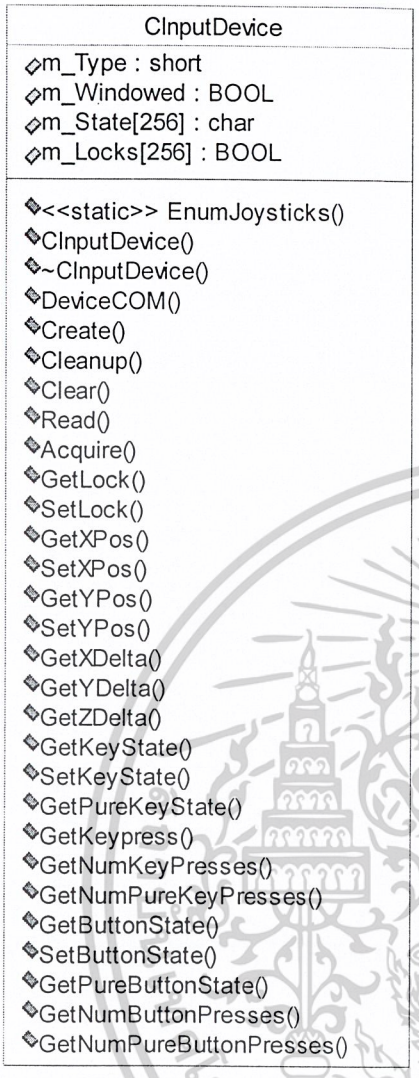
รายละเอียดของคลาสเพิ่มเติม

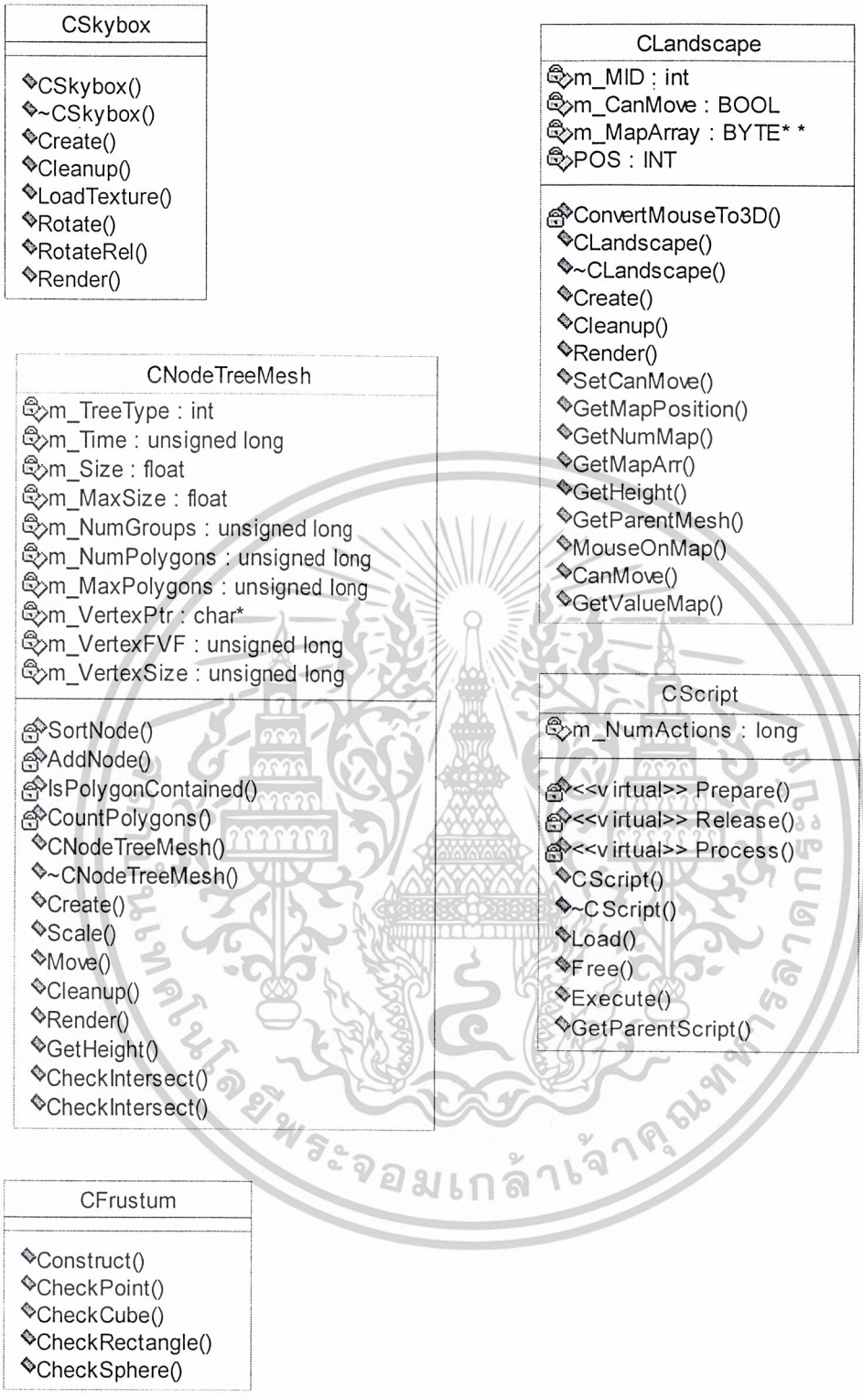


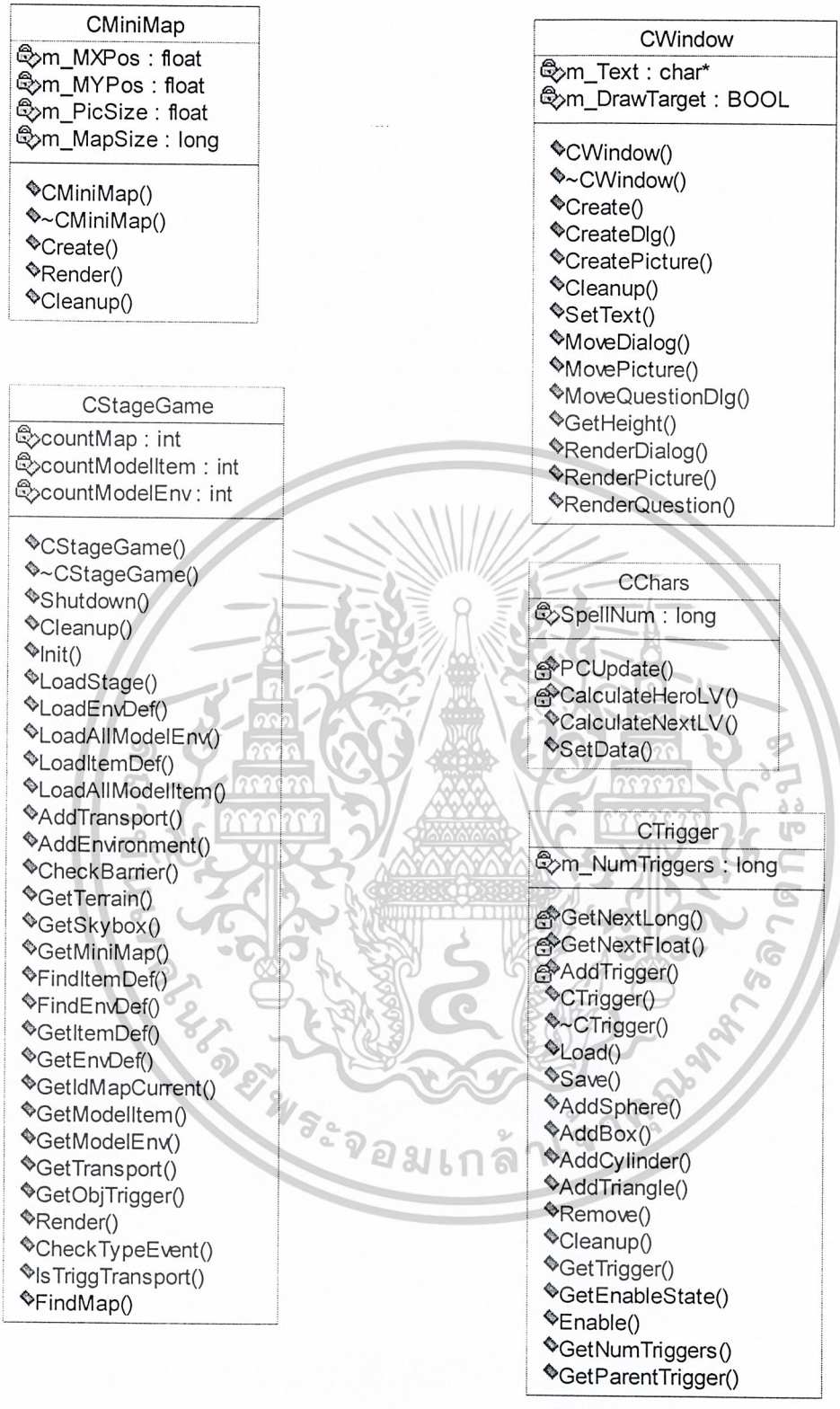












บรรณานุกรม

[1] Todd Barron (2001): " Multiplayer Game Programming " , Prama Publishing , 2001

[2] นิรุธ อำนวยการศิลป์, เขียนเกมอย่างมืออาชีพด้วย Visual C++ และไดเร็คเอ็ก สำนักพิมพ์ อินโฟเพรส , พิมพ์ครั้งที่ 1 มกราคม 2545

[3] Andre LAMOTHE : "Trick of the windows Game Programming Gurus",SAMS

[4] Jim Adams : "Programming Role Playing Games with ไดเร็คเอ็ก", Premier Press

[5] Anthony Jones, Jim Ohlund : "Network Programming for Microsoft Windows, Second Edition", Microsoft Press