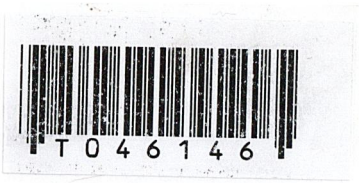
การพัฒนาเกมสามมิติโดยใช้ไดเร็กเอ็กซ์

3D GAME DEVELOPMENT USING DIRECTX



นาย ปาณิก เทศนิยม

นาย ปานบดี เมฆไพบูลย์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

การพัฒนาเกมสามมิติโดยใช้ไดเร็กเอ็กซ์

3D GAME DEVELOPMENT USING DIRECTX

โดย

นาย ปาณิก เทศนิยม

นาย ปานบดี เมฆไพบูลย์

อาจารย์ที่ปรึกษา

อ. สมเกียรติ วังศิริพิทักษ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเกมสามมิติโดยใช้ไดเร็กเอ็กซ์

3D GAME DEVELOPMENT USING DIRECTX

ผู้จัดทำ

1. นาย ปาณิก เทศนิยม รหัสประจำตัว 41014268
2. นาย ปานบดี เมฆไพบูลย์ รหัสประจำตัว 41014269

อาจารย์ที่ปรึกษา

(อ. สมเกียรติ วังศิริพิทักษ์)

การพัฒนาเกมสามมิติโดยใช้ไดเร็กเอ็กซ์

นาย ปาณิก เทศนิยม
นาย ปานบดี เมฆไพบูลย์
อ.สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

การพัฒนาเกมสามมิติ เป็นโครงงานในการศึกษาและพัฒนาการสร้างเกมสามมิติโดยใช้ไดเร็กเอ็กซ์ ซึ่งเป็นเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ทางมัลติมีเดีย และการออกแบบเกมคอมพิวเตอร์ของบริษัท ไมโครซอฟต์ จำกัด ไดเร็กเอ็กซ์มีคุณสมบัติในการดึงความสามารถของฮาร์ดแวร์ออกมาใช้งานอย่างมีประสิทธิภาพ โดยไม่คำนึงถึงชนิดของฮาร์ดแวร์

ทฤษฎีสำคัญที่ใช้ในการพัฒนาโครงงานนี้ ได้แก่ ทฤษฎีของคอมพิวเตอร์กราฟฟิก ซึ่งใช้ในส่วนของการแสดงผลสามมิติต่างๆ โดยผ่านคอมโพเน็นท์ที่เรียกว่า Direct3D และทฤษฎีการเขียนโปรแกรมผ่านระบบเครือข่ายแบบเพียร์ทูเพียร์ในส่วนของการเล่นผ่านระบบเครือข่ายโดยใช้คอมโพเน็นท์ที่เรียกว่า DirectPlay

ทั้งนี้ โครงงานพัฒนาเกมสามมิติโดยใช้ไดเร็กเอ็กซ์เป็นการสร้างเกมในรูปแบบของเกมชูทติ้งบนระบบปฏิบัติการวินโดวส์ ซึ่งมีลักษณะการเล่นสามโหมด คือ ผู้เล่นคนที่หนึ่งสู้กับคอมพิวเตอร์ ผู้เล่นคนที่หนึ่งสู้กับผู้เล่นคนที่สอง และผู้เล่นสองคนช่วยกันเล่นสู้กับคอมพิวเตอร์ โดยในสองโหมดหลัง ผู้เล่นทั้งสองคนสามารถเล่นผ่านระบบเครือข่ายร่วมกันได้โดยไม่จำเป็นต้องอยู่ด้วยกัน

3D GAME DEVELOPMENT USING DIRECTX

Panik Tesniyom

Panbodee Mekpaiboon

Mr. Somkiat Wangsiripitak Adviser

Academic Year 2001

ABSTRACT

The DirectX-based 3D Game Development Project uses DirectX SDK (Software Development Kit) as a developing tool. DirectX is an application created by Microsoft for developing multimedia software as well as computer games. One of its significant features is the capability to obtain the optimized performance of hardware regardless of hardware vendors.

The output of this project is displayed by using the theory of computer graphics via the Direct3D component. And the theory of network programming such as peer-to-peer network programming is used to implement the network communication part via a DirectPlay component.

This project is a shooting game on Windows platform with three modes: one player versus a computer, one player versus another player, and two players versus the computer. Both of the players can play together from remote locations via the network communication system.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายๆ ท่าน โดยเฉพาะอาจารย์ สมเกียรติ วังศิริพิทักษ์ อาจารย์ที่ปรึกษาโครงงาน ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างยิ่ง

ขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ นั่นก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้ให้การเลี้ยงดูมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มกำลัง และยังให้กำลังใจ พร้อมทั้งเอาใจใส่เสมอมาในทุกๆด้าน ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

ปาณิก เทศนิยม
ปานบดี เมฆไพบูลย์

สารบัญ

สารบัญตาราง

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เกมเป็นแอปพลิเคชั่นที่ถูกสร้างขึ้นเพื่อความบันเทิง ซึ่งเครื่องคอมพิวเตอร์ส่วนบุคคลนั้นก็มีแอปพลิเคชั่นทางด้านเกมอยู่มากมาย ตั้งแต่สมัยที่เครื่องคอมพิวเตอร์ส่วนบุคคลยังใช้ระบบปฏิบัติการดอสจนมาถึงปัจจุบันระบบปฏิบัติการวินโดวส์ ซึ่งเกมนั้นถือได้ว่าเป็นแอปพลิเคชั่นที่ต้องการประสิทธิภาพการประมวลผลที่สูง โดยการพัฒนาของเครื่องคอมพิวเตอร์ส่วนบุคคลส่วนหนึ่งก็ได้รับแรงกระตุ้นมาจากแอปพลิเคชั่นทางด้านเกม ตามความเป็นจริงแล้วแอปพลิเคชั่นทางด้านเกมถือได้ว่าเป็นแอปพลิเคชั่นทางด้านมัลติมีเดียอย่างหนึ่ง เนื่องจากมีการประมวลผลทั้งทางด้านภาพ เสียงและส่วนที่ติดต่อกับผู้ใช้งานซึ่งเป็นลักษณะของโปรแกรมทางด้านมัลติมีเดีย

ในอดีตนั้นเกมได้ถูกพัฒนาบนระบบปฏิบัติการดอส ซึ่งมีข้อดีคือผู้พัฒนานั้นสามารถที่จะเข้าถึงการทำงานของฮาร์ดแวร์ในระดับต่ำได้โดยตรง ซึ่งทำให้แอปพลิเคชั่นมีประสิทธิภาพการทำงานที่สูง แต่มีข้อเสียคือ ผู้พัฒนาแอปพลิเคชั่นต้องพัฒนาให้แอปพลิเคชั่นของตนสามารถรองรับการทำงานกับอุปกรณ์ของบริษัทต่างๆที่มีอยู่มากมายในท้องตลาดเช่น การ์ดแสดงผล และการ์ดเสียง ที่มีอยู่มากมายหลายยี่ห้อเพื่อให้ครอบคลุมกลุ่มผู้ใช้งานให้ได้มากที่สุด ซึ่งเป็นงานที่ยากลำบากและสิ้นเปลืองงบประมาณมาก

ต่อมาบริษัทไมโครซอฟต์ได้เปิดตัวระบบปฏิบัติการวินโดวส์ออกมาซึ่งระบบปฏิบัติการนี้เป็นระบบปฏิบัติการแบบ GUI(Graphical User Interface) มีข้อดีคือผู้พัฒนาไม่จำเป็นต้องพัฒนาแอปพลิเคชั่นของตนให้สนับสนุนอุปกรณ์ของแต่ละบริษัทอีกต่อไป เพราะเมื่ออุปกรณ์เหล่านั้นถูกพัฒนามาเพื่อใช้งานกับระบบปฏิบัติการวินโดวส์แล้ว ผู้พัฒนาแอปพลิเคชั่นเพียงแต่พัฒนาโดยยึดรูปแบบมาตราฐานของระบบปฏิบัติการวินโดวส์ก็พอ ซึ่งเป็นคุณสมบัติของการไม่ขึ้นกับอุปกรณ์(Device Independent) แต่ระบบปฏิบัติการวินโดวส์นั้นเป็นระบบปฏิบัติการที่มีการแสดงผลทางด้านกราฟฟิกช้า ดังนั้นในยุคแรกๆ ผู้พัฒนาแอปพลิเคชั่นทางด้านเกมจึงยังคงยึดติดอยู่กับการพัฒนาเกมบนดอสอยู่ ซึ่งเกมที่ทำงานอยู่บนระบบปฏิบัติการวินโดวส์นั้น ก็พอมีอยู่บ้างแต่จะเป็นพวกที่ไม่ต้องการการแสดงผลทางด้านกราฟฟิกอย่างรวดเร็ว เช่น หมากกระดานเป็นต้น

ทางด้านบริษัทไมโครซอฟต์นั้นได้เล็งเห็นว่าแอปพลิเคชั่นทางด้านเกมนั้นมีผู้ใช้งานกันอย่างแพร่หลายจึงมีแนวความคิดที่จะให้ผู้พัฒนาแอปพลิเคชั่นทางด้านเกมและมัลติมีเดียหันมาพัฒนาเกมบนระบบปฏิบัติการวินโดวส์ ดังนั้นทางบริษัทไมโครซอฟต์จึงพัฒนาไดเร็กเอ็กซ์ขึ้นมา ซึ่งเป็น libralies ทางด้านมัลติมีเดียที่มีจุดมุ่งหมายหลักๆดังนี้

1. ไดเร็กเอ็กซ์ประกอบด้วย libralies ที่ทำงานในระดับต่ำ ซึ่งทำงานได้รวดเร็วและไม่มีข้อจำกัดในการพัฒนาแอปพลิเคชั่นทางด้านเกม

2. โครงสร้างของไดเร็กเอ็กซ์ จะทำการยกภาระทางด้านฮาร์ดแวร์ไปให้ผู้ผลิตฮาร์ดแวร์ โดยจะต้องเป็นผู้สร้างไดรเวอร์สำหรับผลิตภัณฑ์ของตน และให้พัฒนาสามารถใช้ความสามารถล่าสุดของฮาร์ดแวร์นั้นๆได้
3. แอปพลิเคชั่นที่ถูกพัฒนาโดยไดเร็กเอ็กซ์นั้นต้องมีประสิทธิภาพที่อยู่สูงกว่าหรืออย่างน้อยที่สุดต้องเทียบเท่ากับประสิทธิภาพของแอปพลิเคชั่นที่ทำงานบนระบบปฏิบัติการดอส

ในปัจจุบันแอปพลิเคชั่นทางด้านเกมและมัลติมีเดียได้หันมาพัฒนาบนระบบปฏิบัติการวินโดวส์กันหมดแล้วเพราะไดเร็กเอ็กซ์นั้นทำให้แอปพลิเคชั่นมีประสิทธิภาพมาก เช่นสามารถใช้ความสามารถของการ์ดเร่งการแสดงผลสามมิติที่ติดตั้งอยู่ในเครื่องคอมพิวเตอร์ได้ อีกทั้งยังช่วยกำจัดปัญหาของยี่ห้อของฮาร์ดแวร์อีกด้วย โดยที่ยังสามารถเข้าถึงการทำงานในระดับต่ำของฮาร์ดแวร์ได้

1.2 จุดประสงค์ของโครงงาน

1. เพื่อศึกษาและทำความเข้าใจการพัฒนาเกมสามมิติด้วยการใช้ไดเร็กเอ็กซ์
2. เพื่อศึกษาและทำความเข้าใจทฤษฎีคอมโพเน็นท์
3. พัฒนาแอปพลิเคชั่นด้วยภาษาเชิงวัตถุ
4. ศึกษาการเขียนโปรแกรมผ่านระบบเครือข่ายโดยใช้ไดเร็กเพลย์

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการพัฒนาเกมสามมิติที่ทำงานบนระบบปฏิบัติการวินโดวส์ โดยมีแนวเกมเป็นเกม Shooting โดยที่สามารถเลือกเล่นได้สามโหมด รวมทั้งสามารถเล่นแบบผ่านระบบเครือข่ายได้ คือ ผู้เล่นคนที่หนึ่งสู้กับคอมพิวเตอร์ ผู้เล่นคนที่หนึ่งสู้กับผู้เล่นคนที่สอง และผู้เล่นสองคนช่วยกันเล่นสู้กับคอมพิวเตอร์ โดยในสองโหมดหลัง ผู้เล่นทั้งสองคนสามารถเล่นผ่านระบบเครือข่ายร่วมกันได้โดยไม่จำเป็นต้องอยู่ด้วยกัน

1.4 แนวทางการดำเนินงาน

1. ค้นคว้าหาข้อมูลจากอินเตอร์เน็ตตามเวปไซต์ต่างๆที่สนับสนุนการพัฒนาแอปพลิเคชั่นชนิดนี้ เช่น www.thaigamedevx.com, จากหนังสือ หรือเอกสารของไมโครซอฟต์ที่นำมาติดตั้ง
2. เลือกเครื่องมือที่ใช้ในการพัฒนาแอปพลิเคชั่น และศึกษาวิธีการใช้งาน
3. ออกแบบโครงสร้างและรูปแบบของเกม รวมทั้งกำหนดแนวทางและกติกาของเกม
4. จัดหาสิ่งที่เป็นองค์ประกอบต่างๆของเกมเช่น วัตถุสามมิติ และเสียงประกอบเกม
5. พัฒนาเกมตามรูปแบบที่ออกแบบไว้
6. ตรวจสอบและแก้ไขข้อผิดพลาด
7. ปรับแต่งความสวยงามของเกม
8. สรุปการพัฒนา

1.5 สาเหตุที่เลือกไดเร็กเอ็กซ์เป็นเครื่องมือในการพัฒนา

1. ให้ประสิทธิภาพสูงสุดเท่าที่ฮาร์ดแวร์ของระบบจะรองรับได้
2. มีแหล่งข้อมูลในการให้การศึกษา และค้นหาข้อมูลรองรับมากมาย
3. เป็นเครื่องมือที่นิยมในการพัฒนาแอปพลิเคชั่นสามมิติในปัจจุบัน
4. เนื่องจากขอบข่ายของงานเป็นเกมที่ทำงานบนระบบปฏิบัติการวินโดวส์ซึ่งเป็นระบบปฏิบัติการของไมโครซอฟต์ ดังนั้นการใช้งานไดเร็กเอ็กซ์ซึ่งเป็นผลิตภัณฑ์ของไมโครซอฟต์จะสามารถทำงานเข้ากันได้เป็นอย่างดี
5. มีการใช้งานกันอย่างแพร่หลาย

บทที่ 2
ทฤษฎีคอมโพเน็นท์และไดเร็กซ์เอ็กซ์

2.1 COM (Component Object Model)

COM เป็นวิธีที่ไมโครซอฟต์ (Microsoft) คิดขึ้นมาเพื่อทำให้ออบเจ็กต์ต่างๆ ที่อาจจะไม่มีความเกี่ยวข้องกัน สามารถติดต่อสื่อสารแลกเปลี่ยนข้อมูลกันได้ โดยไม่ขึ้นกับภาษาที่ใช้เขียน, ไม่ขึ้นกับระบบปฏิบัติการ, ไม่ขึ้นกับระบบของเครื่องคอมพิวเตอร์ ซึ่งมีวิธีการทำโดยออกแบบออบเจ็กต์ให้เป็นคอมโพเน็นท์ (component)คือ รวมเอาเมธอด (method) ย่อยๆ ต่างๆ ที่มีหน้าที่การทำงานในแบบเดียวกัน มารวมกันเป็นคอมโพเน็นท์ โดยเรียกเมธอด ย่อยๆ เหล่านั้นใหม่ว่าอินเตอร์เฟซ (Interface)

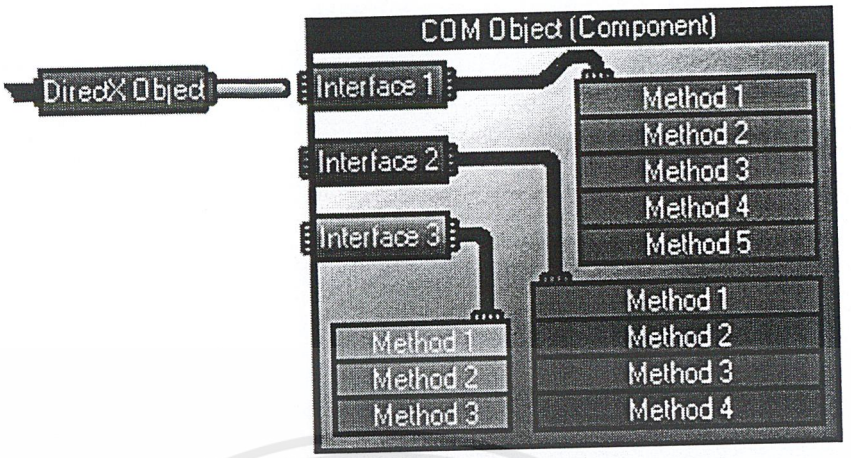
- **Component-Based Development (CBD)** คือวิธีการพัฒนาซอฟต์แวร์วิธีหนึ่งโดยการรวม, พัฒนาและเชื่อมต่อคอมโพเน็นท์ที่มีอยู่เป็นรูปแบบของระบบที่ต้องการ

- **Component** เป็นการปะติดปะต่อแพ็กเกจ (package) ของซอฟต์แวร์ที่สามารถทำการพัฒนาได้โดยไม่ขึ้นต่อกัน รวมทั้งสามารถแก้ไขได้โดยไม่กระทบต่อกัน และนำมาประกอบรวมกันเป็นแอปพลิเคชั่นที่ใหญ่ขึ้น

ในไดเร็กเอ็กซ์เวอร์ชั่น 8.0 ซึ่งเป็นเวอร์ชั่นล่าสุด จะประกอบด้วยคอมโพเน็นท์ 6 ตัว คือ DirectX Graphics, DirectX Audio, DirectInput, DirectPlay, DirectShow และ DirectSetup ซึ่งจะไม่มีคอมโพเน็นท์ที่ชื่อ DirectDraw, Direct3D, DirectSound และ DirectMusic อีกต่อไป โดยจะรวมเอา DirectDraw และ Direct3D ไปเป็นอินเตอร์เฟซ อันเดียวคือ Direct3D อินเตอร์เฟซ ซึ่งอยู่ในคอมโพเน็นท์ของ DirectX Graphics และย้าย DirectSound กับ DirectMusic ไปเป็นอินเตอร์เฟซที่อยู่ในคอมโพเน็นท์ของ DirectX Audio แทน

2.2 ไดเร็กเอ็กซ์ออบเจ็กต์, COM ออบเจ็กต์,อินเ ตอร์เฟซ และ เมธอด

อินเตอร์เฟซแต่ละอินเตอร์เฟซก็จะมีเมธอดหรือฟังก์ชั่นการทำงานต่างๆ การจะเรียกใช้เมธอดนั้นเราจะต้องสร้างไดเร็กเอ็กซ์ออบเจ็กต์ที่ชี้ไปยังอินเตอร์เฟซที่เราต้องการ โดยการสร้างไดเร็กเอ็กซ์ออบเจ็กต์นี้สามารถทำได้โดยการเรียกใช้ฟังก์ชันหรือเมธอดในการสร้างที่ไดเร็กเอ็กซ์เตรียมไว้ให้



ภาพที่ 2-1: ภาพจำลองคอมโพเน็นท์,อินเ ตอร์เฟซ และเมธอด

จากรูป COM ออบเจ็กต์เปรียบเสมือนกล่องที่มีรูหรือช่องให้เสียบหลายๆ รู แต่ละรูก็เปรียบเสมือนอินเตอร์เฟซโดยแต่ละอินเตอร์เฟซก็มีเมธอด ซึ่งก็คือหน้าที่การทำงานแบบต่างๆ เมื่อเรานำแจ็คซึ่งเปรียบเสมือนไดเร็กเอ็กซ์ออบเจ็กต์เข้าไปเสียบที่อินเตอร์เฟซใด เราก็จะสามารถเรียกใช้งานเมธอดในอินเตอร์เฟซนั้นได้ ถ้าจะใช้เมธอดอื่นๆ ที่ไม่ได้อยู่ในอินเตอร์เฟซที่เสียบอยู่ ก็ให้หาแจ็คอื่นมาเสียบเพิ่ม และถ้าเลิกใช้อินเตอร์เฟซใด เราก็ต้องถอดแจ็คนั้นออก

2.3 IUnknown อินเตอร์เฟซ

ทุกอินเตอร์เฟซ จะสืบทอดมาจากคลาสที่ชื่อ IUnknown โดยคลาสนี้ใช้สร้าง IUnknown อินเตอร์เฟซซึ่งเป็นอินเตอร์เฟซมาตรฐานคอยควบคุมไดเร็กเอ็กซ์ออบเจ็กต์ที่เข้ามาใช้อินเตอร์เฟซต่างๆ โดยอินเตอร์เฟซนี้จะมีเมธอด 3 ตัวคือ AddRef(), QueryInterface() และ Release() ดังนั้น อินเตอร์เฟซตัวอื่นๆที่สืบทอดไปจึงมี เมธอดทั้ง 3 นี้ด้วยเสมอ

ทุกครั้งที่เราสร้างไดเร็กเอ็กซ์ออบเจ็กต์ก็จะหมายถึง เราต้องการจะติดต่อกับอินเตอร์เฟซตัวใดในคอมโพเน็นท์ของไดเร็กเอ็กซ์โดย COM ออบเจ็กต์จะเรียก QueryInterface() เพื่อตรวจดูว่ามี อินเตอร์เฟซที่เราต้องการหรือไม่ ถ้ามี มันจะเตรียมพื้นที่ในหน่วยความจำแล้วส่งแอดเดรสกลับมา พร้อมทั้งเรียกใช้ AddRef() เพื่อเพิ่มค่าการนับแก่ Reference Count อีก 1 ดังนั้นถ้าไดเร็กเอ็กซ์ออบเจ็กต์ตัวแรกมีการใช้อินเตอร์เฟซของ COM ออบเจ็กต์ Reference Count จะมีค่าเป็น 1 (กลไกภายใน COM จะโหลด COM ออบเจ็กต์ที่ใช้โดยอัตโนมัติ) หลังจากนั้น ถ้ามีไดเร็กเอ็กซ์ออบเจ็กต์ตัวอื่นๆ มาใช้อินเตอร์เฟซอื่นๆ หรือใช้อินเตอร์เฟซ ซ้ำกันกับไดเร็กเอ็กซ์ออบเจ็กต์ตัวอื่น Reference Count ก็จะถูกบวกค่าไปเรื่อยๆ

หน่วยความจำอินเตอร์เฟซที่ไดเร็กเอ็กซ์ออบเจ็กต์ใช้นี้จะยังคงอยู่ตลอด แม้ว่าเราจะปิดโปรแกรมไปแล้ว ดังนั้นหลังเลิกใช้อินเตอร์เฟซใดๆ เราจะต้องเรียกใช้ Release() เสมอเพื่อให้มันคืนหน่วยความจำ ไม่เช่นนั้นจะมีปัญหาเรื่อง Low Resource นอกจากนี้เมธอด Release() มันยังจะไปลด Reference Count ลงทีละ 1 ต่อการเลิกใช้ อินเตอร์เฟซ 1 ครั้ง เมื่อลดจนเหลือ 0 ก็แสดงว่า ไม่มีไดเร็กเอ็กซ์ออบเจ็กต์ตัวใดใช้

COM ออบเจ็กต์อีกแล้ว กลไกภายใน COM มันก็จะลบ COM ออบเจ็กต์ตัวนั้นออกไปจากหน่วยความจำโดยอัตโนมัติ

นอกจากนี้ QueryInterface() ยังสามารถใช้สร้างไดเร็กเอ็กซ์ออบเจ็กต์ตัวใหม่ (ที่ติดต่อกับอินเตอร์เฟซ) จากไดเร็กเอ็กซ์ออบเจ็กต์เดิมที่มีอยู่ได้ด้วย โดยมันจะไปสอบถาม COM ออบเจ็กต์ซึ่งเป็นเจ้าของอินเตอร์เฟซที่ไดเร็กเอ็กซ์ออบเจ็กต์ตัวเดิมมันกำลังใช้อยู่ว่ามีอินเตอร์เฟซที่ไดเร็กเอ็กซ์ออบเจ็กต์ตัวที่เราจะสร้างใหม่ต้องการหรือไม่

2.4 ที่มาของไดเร็กเอ็กซ์

Microsoft DirectX SDK(Software Development Kit) นั้นเป็นเครื่องมือที่ใช้ในการพัฒนาซอฟต์แวร์ทางมัลติมีเดีย ของทางบริษัทไมโครซอฟต์นั่นเอง ซึ่งรวมถึงสามารถพัฒนาเกมคอมพิวเตอร์ได้อีกด้วย โดยเริ่มแรกนั้นเราจะเห็นเพียงเฉพาะ Header File และ Library ของ Ddraw เท่านั้นหลังจากนั้นงานด้าน 3 มิติเริ่มเข้ามามีบทบาทมากขึ้น ทางบริษัทไมโครซอฟต์ได้ทำการติดต่อกับบริษัทจากอังกฤษที่ชื่อว่าเรียลริตี้แล็บ (Reality Lab) เพื่อขอซื้อ Rendermorphics มาจัดการในงานกราฟิก 3 มิตินั่นเอง ทำให้ทางไมโครซอฟต์มีความคิดที่จะเพิ่มเติมในส่วนอื่นๆอีกเพื่อความสมบูรณ์ในตัว SDK จึงได้ออก Microsoft DirectX SDK version 2 ออกมาเพื่อทำการแนะนำตลาด ซึ่งก็ได้รับการตอบรับจากโปรแกรมเมอร์ทั้งหลายเป็นอย่างดี ทำให้เกิดการพัฒนาเวอร์ชัน ใหม่ขึ้นเรื่อยๆอันได้แก่ Microsoft DirectX SDK version 2,3,5,6,7 และ 8(ในขณะนี้) ทำให้มี อินเตอร์เฟซ ใหม่ๆขึ้นมา รวมถึงการเพิ่มเมธอดบางอย่างเข้าไปอีกเพื่อตอบสนองความต้องการให้มากที่สุดนั่นเอง

2.5 คอมโพเน็นท์ของไดเร็กเอ็กซ์

- **Direct Draw** เป็นส่วนประกอบหลักของ DirectX SDK ทีเดียวเนื่องจากมีหน้าที่จัดการกับกแสดงผลทางจอภาพ, ควบคุมข้อมูลบิตแมป, หน่วยความจำที่ออกนอกพื้นที่สกรีน, สร้างการติดต่อที่รวดเร็วให้กับฟีเจอร์ของฮาร์ดแวร์ เช่น การทำบลิตติง (Blitting) และ เพจฟลิปปิง (Page Flipping) ซึ่งเป็นฟีเจอร์พื้นฐานที่ Direct3D สามารถทำได้ รวมทั้งให้ความเร็วในการประมวลผลเร็วกว่าเกมบน Dos หรือ GDI (Graphic Device อินเตอร์เฟซ) ของ Window เนื่องจากการส่งข้อมูลที่ทำได้โดยตรงนั่นเอง อีกทั้งประกอบไปด้วยฟังก์ชันต่างๆที่สนับสนุนการพัฒนาเกมสองมิติอย่างมีประสิทธิภาพ

- **Direct 3D** เป็นเอพีไอ (APT: Application Programming อินเตอร์เฟซ) ที่มีความสามารถใช้เขียนไปรแกรมกราฟฟิกสามมิติและติดต่อใช้งานฮาร์แวร์เร่งความเร็วสามมิติ Direct3D ในยุคแรกมีอยู่สองโหมดคืออิมมิเดียทโหมด (IM: Immediate Mode) และรีเทนด์โหมด (RM: Retained Mode)

ในอิมมิเดียทโหมดเป็นโหมดที่ใช้งานยากแต่มีความยืดหยุ่นสูง เป็นเอพีไอในระดับล่างสำหรับใช้เขียนเกมที่ทำงานได้เร็วและมีประสิทธิภาพเท่าที่จะเป็นไปได้ ในระบบ ในรีเทนด์โหมด เป็นโหมดที่สร้างขึ้นมาเป็นเลเยอร์ที่อยู่บนสุดของอิมมิเดียทโหมด โดยรีเทนด์โหมดนี้จะจัดเตรียมบริการต่างๆเช่น การจัด

การพื้นผิว การโหลดออบเจ็กต์ไฟล์ และการทำออบเจ็กต์เคลื่อนไหว การศึกษาและใช้งานรีเทนด์โหมดจะใช้งานง่ายกว่าเมื่อเทียบกับอิมมิเดียทโหมด แต่ถ้าโปรแกรมเมอร์คนใดต้องการความยืดหยุ่นและประสิทธิภาพที่สูงกว่าก็มักจะเลือกใช้การทำงานในโหมดแบบอิมมิเดียทโหมด ดังนั้นการพัฒนาการทำงานในรีเทนด์โหมดจึงได้หยุดลงใน DirectX 6.0 และมุ่งพัฒนาการทำงานในโหมดอิมมิเดียดโหมดให้มีความสามารถและใช้งานง่ายในเวอร์ชันต่อๆมา ด้วยเหตุนี้การทำงานในโหมดรีเทนด์ จึงไม่สนับสนุนเทคโนโลยีใหม่ๆ เช่น Multitexturing, Bump Mapping, Hardware Transformation และ Lighting

- **Direct Input** ทำให้การสร้างเกมเพื่อรองรับการรับการสั่งงานจากผู้เล่นไม่ว่าจะเป็น Joy Stick, Mouse หรือว่า Keyboard ทำได้ง่ายและมีประสิทธิภาพ

- **Direct Music** เป็นเอพีไอที่ทำงานร่วมกับข้อมูลประเภท Message-Based Musical Data ซึ่งเป็นข้อมูลที่แปลงมาจาก Wave Sample ด้วยซินทิไซเซอร์ (Synthesizer) ทั้งแบบซอฟต์แวร์หรือฮาร์ดแวร์ และรองรับระบบ Playing Music และ DLS-Based Sound Effect ได้อย่างสมบูรณ์ในขณะ Runtime

- **D irect Sound** เอพีไอชุดนี้เป็นเครื่องมือที่ใช้จัดการเสียงแบบสเตอริโอและระบบสามมิติอย่างมีประสิทธิภาพ ประกอบด้วยความสามารถในการจัดการหน่วยความจำ และการผสมเสียงของฮาร์ดแวร์ Direct Sound ได้รับการออกแบบมาเพื่อดึงความสามารถของฮาร์ดแวร์บนระบบ การรวมเสียงแบบสามมิติเข้าไปในเกม ทำให้เกมของคุณให้เสียงได้สมจริง

- **Direct Play** ทำให้การสร้างเกมระบบผู้เล่าหลายๆคนทำได้ง่ายขึ้น และทำให้สามารถเชื่อมต่อแบบ Transport-Independent รวมทั้งยังให้บริการ Messaging Service ด้วย

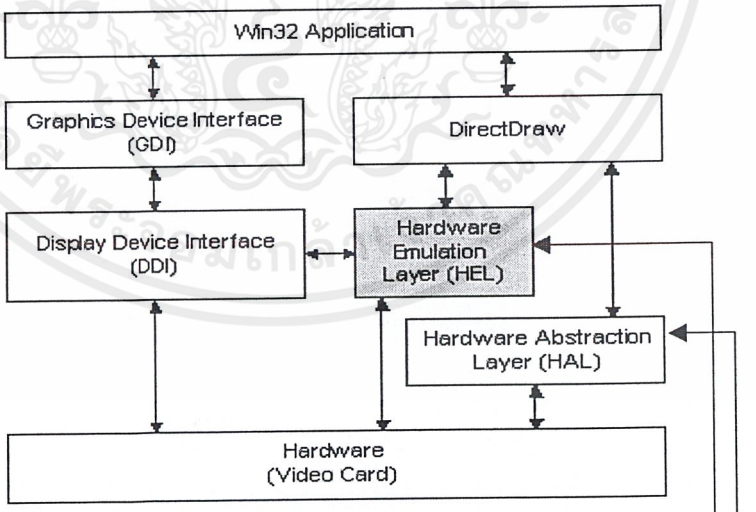
- **Direct Setup** ช่วยในการติดตั้งส่วนประกอบที่จำเป็นต้องใช้ของ DirectX (DirectX Runtime) ให้กับผู้ใช้อย่างอัตโนมัติ ถ้าหากระบบดังกล่าวยังไม่ได้รับการติดตั้ง

2.6 ลักษณะการทำงานของไดเร็กเอ็กซ์ขณะเล่นเกม

หลังจากที่ติดตั้งไดเร็กเอ็กซ์ให้กับเครื่อง สิ่งที่เกิดขึ้นคือ จะมี ddraw.dll, dsound.dll, etc. ไปปรากฏที่ Windows/System directory ขณะที่รันเกม ไฟล์เหล่านี้ก็จะถูกโหลดขึ้นมาลิงค์กับโปรแกรมเกม จากนั้นเกมก็จะสามารถเรียกใช้ความสามารถของไดเร็กเอ็กซ์ได้ ไฟล์ .DLL เหล่านี้จะคอยติดต่อกับไดรเวอร์ (Driver)ของการ์ดแสดงผล,การ์ดเสียง, โมเด็ม และส่วนประกอบอื่นๆของเครื่องอีกครั้งหนึ่ง สรุปได้ว่าโปรแกรมจะเรียกใช้ฮาร์ดแวร์ได้ผ่านไดเร็กเอ็กซ์ซึ่งจะเรียกผ่านไดรเวอร์ที่ติดตั้งไว้อีกทีหนึง ตรงนี้จะเห็นได้ว่าผู้เขียนโปรแกรมไม่จำเป็นต้องไปเรียกใช้ไดรเวอร์หรือเรียกใช้ฮาร์ดแวร์โดยตรง เพียงแต่เรียกผ่านคำสั่งต่างๆ ในไดเร็กเอ็กซ์เท่านั้นเอง แล้วไดเร็กเอ็กซ์ก็จะติดต่อกับฮาร์ดแวร์ให้เอง ซึ่งตรงนี้เป็นจุดสำคัญ ที่ทำให้ไดเร็กเอ็กซ์ประสบความสำเร็จได้ เพราะหมดปัญหาเรื่องความ Compatible และยังทำให้ผู้เขียนโปรแกรมไม่ต้องไปยุ่งกับพวกคำสั่งระดับล่างต่างๆด้วย โดยถ้าเป็นเกมบนดอสนั้น เวลาจะเขียนเกม แต่ละครั้งโปรแกรมเมอร์ต้องทำส่วนติดต่อกับการ์ดแสดงผล ส่วนติดต่อกับการ์ดเสียง ส่วนติดต่อกับ

โมเด็ม ส่วนติดต่อกับ Joystick และส่วนติดต่ออื่นๆ อีกตั้งมากมาย ด้วยตัวเอง แต่เมื่อไดเร็กเอ็กซ์ได้รับการพัฒนาขึ้นมา จะเห็นได้ว่าปัญหาเก่าๆ พวกนี้ก็หมดไป

รายละเอียดการทำงานของไดเร็กเอ็กซ์มีขั้นตอนในการทำงานดังนี้ ไดเร็กเอ็กซ์แบ่งใหญ่ๆ ได้เป็น 2 ส่วน คือ DirectX Foundation และ DirectX Media, DirectX Foundation นั้นประกอบไปด้วยส่วน DirectDraw, DirectSound, Direct3D อิมมิเดียทโหมดและ DirectInput โดยจะดูแลฟังก์ชันระดับล่างต่างๆ ส่วน DirectX Media นั้น ประกอบไปด้วย DirectShow, DirectAnimation, DirectPlay และ Direct3D รีเทนด์โหมดซึ่งเป็นบริการระดับสูง ที่จะไปเรียก DirectX Foundation อีกทีหนึ่ง สำหรับเกมทั่วไปจะเรียกใช้ DirectX Foundation เป็นหลัก และเรียกใช้ DirectX Media เมื่อต้องการได้รับบริการบางอย่าง DirectAnimation เป็นเอพีไอสำหรับใช้ในเว็บเพจซึ่ง IE4 ของไมโครซอฟท์สนับสนุน สำหรับ DirectShow นั้นเป็นเอพีไอสำหรับเรียกไฟล์หนัง(เช่น .AVI) ได้อย่างสะดวก ซึ่ง Final Fantasy 7 บนเครื่องพีซีก็ใช้ตัวนี้ อีกจุดหนึ่งที่อาจจะสงสัยคือ Direct3D อิมมิเดียทโหมด(IM) และรีเทนด์โหมด(RM) ต่างกันอย่าไร ถ้าดูจากว่าอิมมิเดียทโหมดอยู่ใน Foundation และรีเทนด์โหมดอยู่ใน Media ก็พอจะบอกได้ว่ารีเทนด์โหมดจะประกอบด้วยฟังก์ชันระดับสูงกว่าอิมมิเดียทโหมดนั่นก็คืออิมมิเดียทโหมดจะประกอบด้วยฟังก์ชันการวาดรูประดับล่างมีความสามารถในการวาดออบเจ็กต์สามมิติระดับต่ำเท่านั้น ส่วนรีเทนด์โหมดจะวาดออบเจ็กต์ที่ระดับสูงกว่ากว่า อย่างเช่นอิมมิเดียทโหมดจะวาดได้แค่ 3 เหลี่ยม ในขณะที่รีเทนด์โหมดจะมีความสามารถในการวาดโพลีกอนเลย แต่เกมทั่วๆ ไปจะใช้อิมมิเดียทโหมดเนื่องจากรีเทนด์โหมดนั้นไมโครซอฟท์ทำไว้ไม่ค่อยดีนัก และอิมมิเดียทโหมดของ Direct3D ก็อยู่ในระดับเดียวกับ OpenGL คือเป็นระดับล่างเหมือนกัน



HAL เป็นส่วนที่ฮาร์ดแวร์สนับสนุนฟังก์ชันบางฟังก์ชันของ Direct Draw
HEL เป็นส่วนที่จัดการกับฟังก์ชันต่างๆที่ไม่ได้รับการสนันสนุนจากฮาร์ดแวร์

ภาพที่ 2-2: ภาพแสดงการทำงานการเรียกใช้ฮาร์ดแวร์ในระดับล่าง(Low Level)

ส่วนของ DirectX Foundation ในส่วน DirectDraw ซึ่งเป็นส่วนประกอบหลัก จากรูปจะเห็นว่า Win32 Application สามารถเขียนภาพบนหน้าจอได้ 2 ทาง นั่นคือ ผ่านทาง GDI และผ่าน DirectDraw ในโปรแกรมบนวินโดวส์ทั่วๆไปนั้น ไม่ว่าจะเป็น Netscape, IE, Photoshop, Visual C++ หรือแม้แต่วินโดวส์เอง การเขียนกราฟฟิกบนหน้าจอจะเขียนผ่าน GDI ทั้งสิ้น ซึ่ง Win32 API นั้นจะเรียกใช้ GDI ในทุกคำสั่งที่มีการวาดบนหน้าจอ เช่น MessageBox ฯลฯ จะมีก็พวกเกมและโปรแกรมมัลติมีเดียต่างๆ เท่านั้นที่จะเรียกใช้ DirectDraw แล้วทำไมโปรแกรมเหล่านี้ถึงไม่ใช้ GDI เหตุผลหลักเนื่องจากการใช้ GDI จะช้า ในเกมทั่วๆ ไปต้องการ Frame Rate สูงๆ ซึ่งถ้าใช้ GDI ความเร็วที่ได้จะไม่เร็วพอ อีกสาเหตุหนึ่งที่ GDI ไม่เหมาะสมก็คือ ไม่สนับสนุนการสลับเพจซึ่ง การสลับเพจก็คือ การเขียนหน้าจอใน Background Page แล้วพอเขียนเสร็จค่อยสลับมาข้างหน้า ผลที่ได้ก็คือการ update จอจะเร็วมาก และจะไม่เกิดอาการแบบไล่ภาพจากบนลงล่าง

จะเห็นว่า DirectDraw จะติดต่อกับ HEL และ HAL, HAL (Hardware Abstraction Layer) ก็คือส่วนที่จะติดต่อกับฮาร์ดแวร์โดยตรง ส่วนนี้คือส่วนที่ผู้ผลิตการ์ดจอจะต้องทำมาให้ โดยอาจจะอยู่ในรูปไดรเวอร์หรือเป็นส่วนที่ติดต่อกับไดรเวอร์อีกทีหนึ่ง โดย DirectDraw จะทำหน้าที่ในการติดต่อกับ HAL เองแอปพลิเคชั่นไม่ต้องเข้ามาทำการติดต่อเลย HAL นั้นจะรายงานความสามารถของอุปกรณ์ (ในที่นี้คือการ์ดแสดงผล) ให้ DirectDraw ทราบ ซึ่ง DirectDraw ก็สามารถรายงานความสามารถต่อให้กับแอปพลิเคชั่นก็ได้ แต่ถ้าแอปพลิเคชั่นเรียกใช้ฟังชั่นที่ต้องการความสามารถที่อุปกรณ์ที่ใช้ขณะนั้นไม่สามารถทำได้ DirectDraw จะไปเรียกใช้ HEL (Hardware Emulation Layer) แทน ซึ่ง HEL นี้จะจำลองความสามารถของฮาร์ดแวร์ให้กับ DirectDraw และ Application ได้ใช้ ยกตัวอย่างเช่นการ Stretch Blit ซึ่งคือการแสดงบิตแมปที่ขยายจากบิตแมปที่เก็บในหน่วยความจำ บนหน้าจอ การ์ดแสดงผลบางตัวจะสนับสนุนความสามารถนี้ผ่านฮาร์ดแวร์แต่ถ้าเป็นการ์ดแสดงผลที่ไม่สนับสนุนความสามารถนี้ DirectDraw จะต้องไปเรียกใช้ HEL แทน ซึ่ง HEL จะทำการจำลองคุณสมบัตินี้ด้วยซอฟต์แวร์ซึ่งผลที่ได้ก็คือสามารถทำงานเหมือนการ์ดจอที่ไม่มีคุณสมบัตินี้ แต่ทำงานได้ช้ากว่าเพราะเป็นการใช้ซอฟต์แวร์แทนฮาร์ดแวร์ สำหรับ HEL นั้น จะเป็นส่วนที่อยู่ในไดเร็กเอ็กซ์คือไมโครซอฟต์เขียนให้แล้ว และจะทำงานอย่างอัตโนมัติ ผู้เขียนโปรแกรมไม่ต้องสั่งให้มีการใช้

ในส่วนอื่นๆ ของ DirectX Foundation เช่น DirectSound และ Direct3D Immediate Mode ก็จะมีลักษณะการเรียกใช้ HEL และ HAL คล้ายๆ กับตัวอย่างของ DirectDraw ข้างบนนี้

บทที่ 3
ไดเร็กเอ็กซ์กราฟฟิก

3.1 ไดเร็กเอ็กซ์กราฟฟิก

ไมโครซอฟต์ Direct3D ถูกออกแบบเพื่อสร้างเกมที่โต้ตอบกันได้แบบสามมิติ บนเครื่องคอมพิวเตอร์ภายใต้ระบบปฏิบัติการวินโดวส์ ซึ่งใช้อินเตอร์เฟซการแสดงผลที่รองรับการเข้าถึงอุปกรณ์แสดงผลแบบสามมิติโดยไม่ขึ้นกับอุปกรณ์

Direct3D เป็นเอพีไอสามมิติระดับล่าง ซึ่งเป็นเครื่องมือสำหรับนักพัฒนาที่ต้องการสร้างเกมมัลติมีเดียประสิทธิภาพสูงที่ทำงานภายใต้ระบบปฏิบัติการวินโดวส์ เนื่องจากการใช้งานเอพีไอซึ่งทำงานอยู่ที่ระดับล่างนี้มีความยืดหยุ่นในการใช้งานสูง

ข้อดีของ Direct3D มีดังต่อไปนี้:

- สามารถเปลี่ยน depth buffering ได้ (โดยใช้ z-buffers หรือ w-buffers).
- ทำ Shading ได้ทั้งแบบ Flat และ Gouraud
- สร้างแหล่งกำเนิดแสงได้หลายตัวและหลายชนิด
- รองรับการทำวัสดุและพื้นผิวรวมทั้งรองรับการทำ Mipmapping.
- รองรับการใช้ซอฟต์แวร์จำลองการทำงานของฮาร์ดแวร์(Software Emulator)
- รองรับการแปลงพิกัดวัตถุ(Transformation) และการขริบภาพ(Clipping)
- ทำงานโดยไม่ขึ้นกับชนิดของฮาร์ดแวร์
- รองรับสถาปัตยกรรมแบบ Intel MMX , Intel Streaming Single-Instruction, Multiple-Data (SIMD) Extensions (SSE), และสถาปัตยกรรม AMD's 3D architecture.
- รองรับ Hardware Abstraction Layer (HAL)
- รองรับการทำ Page Flipping ด้วย Back Buffers หลายๆตัวในแอปพลิเคชั่นแบบเต็มหน้าจอ(full-screen Application)
- รองรับการขริภาพของแอปพลิเคชั่นทั้งวินโดวส์โหมดและโหมดแสดงผลแบบเต็มหน้าจอ
- รองรับการทำ 3D z-buffers.

- เข้าถึงหน่วยความจำของอุปกรณ์แสดงผล(display-device memory)แบบ Standard และ Enhance ได้ในเวลาเดียวกัน

ข้อดีดังกล่าวนี้มีไว้เพื่อให้ใช้พัฒนาแอปพลิเคชั่นแบบ GDI-Based ที่ทำงานภายใต้วินโดวส์หรือแม้กระทั่งดอสได้อย่างง่ายดาย

การพัฒนาแอปพลิเคชั่นที่เกี่ยวกับ Direct3D นั้นต้องทำการจัดการเกี่ยวกับ Vertice, Polygon และคำสั่งที่ใช้ควบคุมมันรวมทั้งการเข้าถึงการแปลงพิกัดวัตถุ, การให้แสง(Lighting) และขั้นตอนการแสดงผลแบบท่อส่งน้ำ(Rasterization 3D Graphics Pipeline)

Direct3D สนับสนุนวิธีการเรนเดอร์สกรีนสามมิติอย่างง่าย โดยการอ้างถึงเมธอด **DrawPrimitive** ซึ่งเมธอดดังกล่าวนี้สามารถรองรับการเรนเดอร์วัตถุในหนึ่งสกรีนได้มากกว่าหนึ่งวัตถุ โดยทำการเรียกเมธอดเพียงครั้งเดียว

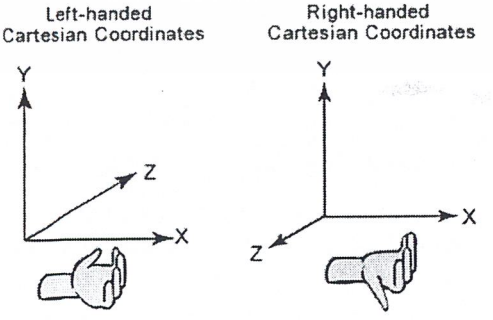
3.2 ระบบพิกัดสามมิติและรูปทรงเรขาคณิต

การเขียน Program Direct 3D ต้องทำความเข้าใจกับเรขาคณิต 3มิติให้คุ้นเคยก่อน โดยที่หัวข้อนี้จะกล่าวถึงภาพรวมของเรขาคณิตสามมิติที่สำคัญสำหรับการสร้างสกรีนสามมิติ ซึ่งมีหัวข้อดังนี้

- ระบบพิกัดสามมิติ(3-D Coordinate Systems)
- 3-D Primitives

3.2.1 ระบบพิกัดสามมิติ

ระบบพิกัดที่ใช้ใน Direct3D เป็นระบบพิกัดมือซ้าย ทั้งระบบพิกัดมือซ้ายและระบบมือขวา เป็นระบบพิกัดที่มีแกน +X อยู่ทางด้านขวา และแกน +Y อยู่ทางด้านบน แต่แกน Z จะอยู่ในทิศทางใดขึ้นอยู่กับระบบพิกัดแต่ละระบบ คุณสามารถจดจำทิศทางของแกน +Z ได้โดยการหงายมือขึ้นและชี้นิ้วมือไปทางแกน +X จากนั้นงอนิ้วขึ้นด้านบนซึ่งเป็นทิศทางของแกน +Y ทิศทางที่นิ้วโป้งชี้อยู่จะเป็นทิศทางของแกน +Z ดังรูปด้านล่างนี้



ภาพที่ 3-1: ภาพแสดงระบบพิกัดสามมิติทั้งแบบมือซ้ายและมือขวา

Microsoft Direct3D จะใช้ระบบพิกัดฉากแบบมือซ้าย ซึ่งถ้ามีการใช้ระบบพิกัดฉากแบบมือขวาพัฒนาแอปพลิเคชั่นมาก่อน ต้องทำการเปลี่ยนข้อมูลดังนี้

- เปลี่ยนลำดับของจุดของสามเหลี่ยมแต่ละรูปในออบเจ็กต์ โดยสลับตามเข็มนาฬิกาจากด้านหน้า เช่นถ้าลำดับของจุดเป็น v0, v1, v2 ต้องทำการส่งค่าไปให้ Direct3D คือ v0, v2, v1.
- ใช้การแมทริกซ์วิวเพื่อปรับพิกัดเวิลด์ด้วย -1 ในทิศทางของแกน Z ด้วยการสลับเครื่องหมายของแถวที่สามในแมทริกซ์

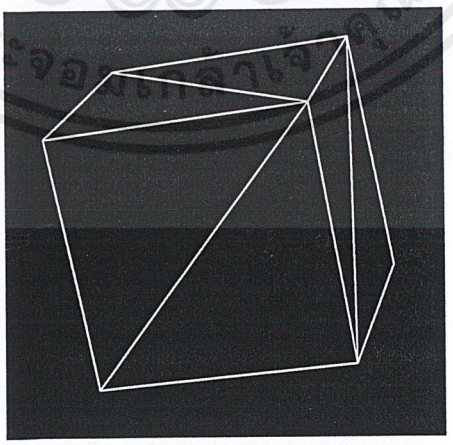
Operation ที่สำคัญที่กระทำกับวัตถุที่กำหนดไว้ในระบบพิกัดฉากคือ การเปลี่ยนตำแหน่งวัตถุ, การหมุนวัตถุ และการปรับขนาดวัตถุ

3.2.2 3D Primitives

3D Primitive คือชุดของจุด(Vertices) ของวัตถุ 3 มิติ และ Primitive ที่ซับซ้อนน้อยที่สุดคือชุดของจุดในระบบ พิกัดซึ่งถูกเรียกว่า PointList

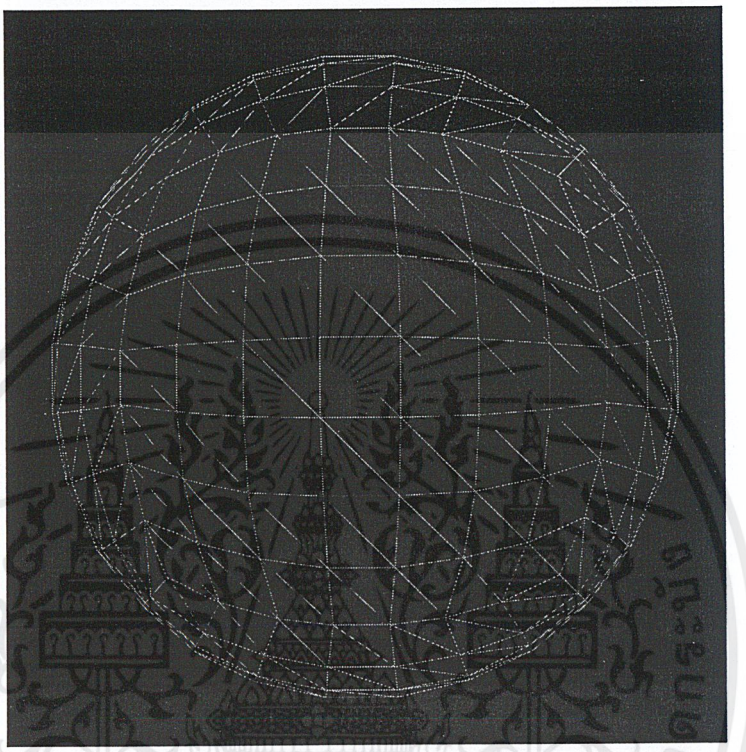
โดยทั่วๆไป 3D Primitive จะหมายถึงโพลีกอน ซึ่งก็คือรูปปิดที่มีจุดอย่างน้อยสามจุด โพลีกอนพื้นฐานทั่วไปจะเป็นรูปสามเหลี่ยม Direct3D ใช้สามเหลี่ยมเป็นหลักเนื่องจากจุดสามจุดในรูปสามเหลี่ยมนั้นจะรับประกันได้ว่าจะอยู่ในระนาบเดียวกัน เพราะถ้าหากว่าจุดไม่อยู่ในระนาบเดียวกันจะทำให้การเรนเดอร์ไม่มีประสิทธิภาพเท่าที่ควร และเราสามารถจะประกอบรูปสามเหลี่ยมเป็นโพลีกอนที่ซับซ้อนหรือวัตถุได้

รูปด้านล่างนี้เป็นรูปที่แสดงรูปลูกบาศก์ที่แต่ละด้านของลูกบาศก์ประกอบด้วยสามเหลี่ยมสองรูป โดยที่รูปสามเหลี่ยมทั้งหมดประกอบเป็นลูกบาศก์เพียงวัตถุเดียวเท่านั้น และเราสามารถจะใช้พื้นผิว(Texture) หรือ วัสดุ(Materials) แสดงผลไปที่ด้านของลูกบาศก์เพื่อให้เห็นเป็นลูกบาศก์ตันเพียงหนึ่งวัตถุได้



ภาพที่ 3-2: ภาพแสดงการประกอบกันของโพลีกอนเป็นรูปลูกบาศก์

เราสามารถใช้สามเหลี่ยมเพื่อสร้างรูปทรงที่มีพื้นผิวราบเรียบได้ โดยรูปด้านล่างนี้จะแสดงรูปของวัตถุทรงกลมสามมิติที่ประกอบด้วยสามเหลี่ยม และเมื่อทำการปะพื้นผิวไปก็จะเห็นเป็นทรงกลมเรียบในเวลาเรนเดอร์ออกมา



ภาพที่ 3-3: ภาพแสดงทรงกลมสามมิติที่ประกอบจากโพลีกอนสามเหลี่ยม

3.3 เทคนิคการให้แสงเงา

ในส่วนนี้จะอธิบายถึงเทคนิคที่ Direct3D ใช้ในการจัดการแสงเงาของโพลีกอนสามมิติซึ่งมีหัวข้อย่อยดังนี้

- โหมดการให้แสงเงา(Shading Mode)
- การเปรียบเทียบโหมดการให้แสงเงา(Comparing Shading Mode)
- การกำหนดโหมดการให้แสงเงา(Setting the Shading Mode)
- ผิวหน้าและเวคเตอร์ปกติ(Face and Vertex Normal Vectors)
- Triangle Interpolants

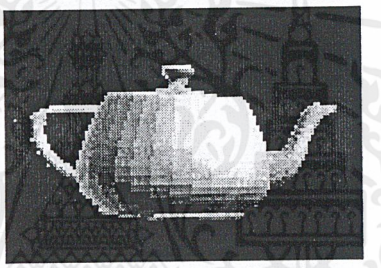
3.3.1 โหมดการให้แสงเงา(Shading Modes)

การให้แสงเงาใช้สำหรับการเรนเดอร์โพลีกอนให้มีการแสดงผลแสงเงา คือ ข้อกำหนดความเข้มของแสงและสีที่จุดใดบนโพลีกอนซึ่ง Direct3D รองรับโหมดการให้แสงเงาสองโหมดคือ

- **การให้แสงเงาแบบแบนราบ(Flat Shading)**

ในการให้แสงเงาแบบแบนราบ Direct3D จะทำการเรนเดอร์โพลีกอนโดยใช้สีวัสดุของโพลีกอนเป็นสีทั้งหมดของโพลีกอนนั้น และวัตถุสามมิติที่เรนเดอร์แบบการให้แสงเงาแบบแบนราบจะเห็นขอบของโพลีกอนถ้าหากว่าไม่ได้อยู่ในระนาบเดียวกัน

รูปด้านล่างนี้แสดงวัตถุกาน้ำชาที่ทำการเรนเดอร์แบบการให้แสงเงาแบบแบนราบซึ่งจะเห็นขอบของโพลีกอนที่อยู่คนละระนาบได้ การทำการให้แสงเงาแบบแบนราบนี้จะเสียเวลาในการคำนวณน้อยที่สุด

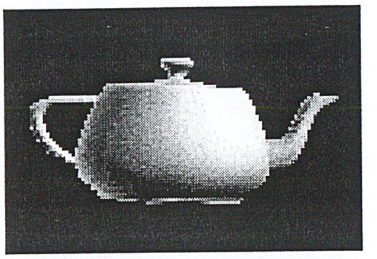


ภาพที่ 3-4 : ภาพแสดงวัตถุที่ใช้การให้แสงเงาแบบแบนราบ

- **การให้แสงเงาแบบ Gouraud (Gouraud Shading)**

เมื่อ Direct3D เรนเดอร์โพลีกอนโดยใช้การให้แสงเงาแบบ Gouraud จะต้องทำการคำนวณสีสำหรับแต่ละจุดจากจุดปกติ(Normal Vertex) และตัวแปรของแสงและทำการเพิ่มสีไปที่ผิวหน้าของโพลีกอนอย่างเป็นเชิงเส้น ตัวอย่างเช่น ถ้าส่วนประกอบของสีแดงสำหรับจุดที่หนึ่งมีค่าเป็น 0.8 และส่วนประกอบของสีแดงสำหรับจุดที่สองมีค่าเป็น 0.4 ด้วยการใช้การให้แสงเงาแบบ Gouraud และโหมดสีแบบแดงเขียวน้ำเงิน(RGB) Direct3D จะทำการกำหนดค่าส่วนประกอบของสีแดงเท่ากับ 0.6 ไว้ที่จุดกึ่งกลางระหว่างจุดที่หนึ่งและจุดที่สอง ซึ่งจะเห็นได้ว่าผลการคำนวณเป็นแบบเป็นเชิงเส้น

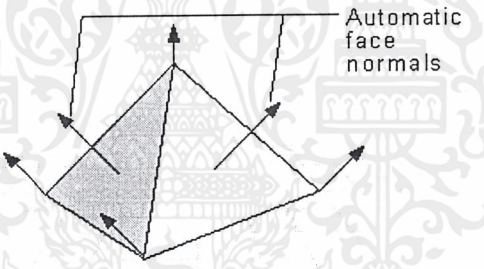
รูปด้านล่างนี้แสดงภาพของวัตถุที่ใช้การให้แสงเงาแบบ Gouraud ซึ่งกาน้ำชานี้ประกอบขึ้นจากโพลีกอนแบบสามเหลี่ยมหลายๆรูป ซึ่งมีผลให้พื้นผิวของวัตถุโค้งและเรียบ



ภาพที่ 3-5 : ภาพแสดงวัตถุที่ใช้การให้แสงเงาแบบ Gouraud

3.3.2 การเปรียบเทียบโหมดการให้แสงเงา(Comparing Shading Modes)

ในการให้แสงเงาแบบแบนราบปิรามิดด้านล่างจะถูกแสดงผลด้วยขอบที่ชัดเจนระหว่างด้านที่ติดกัน ส่วนในการให้แสงเงาแบบ Gouraud ค่าแสงเงาจะถูกเพิ่มเข้าไประหว่างขอบ และท้ายที่สุดจะเห็นเป็นด้านโค้ง

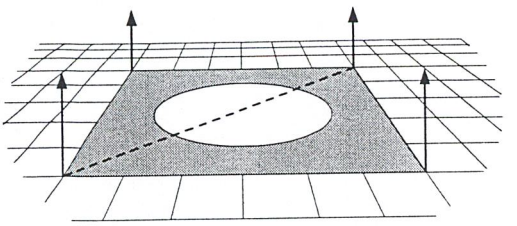


ภาพที่ 3-6 : ภาพปิรามิดแสดงขอบการให้แสงเงาวัตถุ

ในการให้แสงเงาแบบ Gouraud แสงที่ตกกระทบวัตถุจะสมจริงกว่าการให้แสงเงาแบบแบนราบ เนื่องจากสีในการให้แสงเงาแบบแบนราบ จะเป็นสีแบบเดียวกันทั้งหมด(Uniform) แต่ การให้แสงเงาแบบ Gouraud จะสามารถควบคุมแสงให้ตกกระทบที่พื้นผิวได้อย่างถูกต้อง

การให้แสงเงาแบบ Gouraud จะทำให้ขอบระหว่างโพลีกอนเรียบ ซึ่งตรงกันข้ามกับการให้แสงเงาแบบแบนราบที่จะเห็นขอบวัตถุที่อยู่คนละระนาบได้อย่างชัดเจน อย่างไรก็ตามการให้แสงเงาแบบ Gouraud ก็อาจแสดงผลแบบ Mach Bands(คือสีและแสงระหว่าง Polygon ที่ติดกันไม่ราบเรียบ) ได้ ซึ่งสามารถแก้ไขได้ด้วยการเพิ่มโพลีกอนให้วัตถุนั้นๆ, เพิ่มความละเอียดของหน้าจอแสดงผล หรือเพิ่ม ความลึกของสี(Color Depth)

การให้แสงเงาแบบ Gouraud อาจจะทำให้ส่วนประกอบปลีกย่อยผิดพลาดได้ ตัวอย่างดังเช่นรูปด้านล่างนี้ที่สปอร์ตไลท์อยู่ภายในพื้นที่ของโพลีกอนทั้งหมด



ภาพที่ 3-7 : ภาพแสดงสปอร์ตไลท์ที่อยู่ภายในโพลีกอนอื่นๆ

ในกรณีนี้การให้แสงเงาแบบ Gouraud ซึ่งทำการเพิ่มค่าต่างๆระหว่างจุด จะไม่แสดงผลของสปอร์ตไลท์พื้นผิวจะถูกเรนเดอร์โดยเหมือนว่าไม่มีแสงสปอร์ตไลท์อยู่

3.3.3 การกำหนดโหมดการให้แสงเงา(Setting the Shading Mode)

Direct3D สามารถเลือกโหมดของการให้แสงเงาได้ในเวลาใดก็ได้ โดยมีค่าเริ่มต้น(Default) เป็นการให้แสงเงาแบบ Gouraud ในภาษา C++ สามารถเปลี่ยนโหมดการให้แสงเงาได้โดยเรียกใช้เมธอด **Idirect3Ddevice8 ::SetRenderState** โดยกำหนดค่า *state* พารามิเตอร์เป็น D3DRS SHADEMODE

โค้ดด้านล่างนี้เป็นตัวอย่างที่แสดงถึงวิธีการกำหนดโหมดการให้แสงเงาในแอปพลิเคชั่นแบบสามมิติ

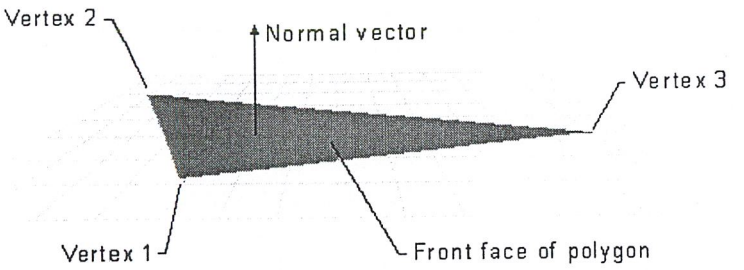
```
// Set to flat shading.
// This code example assumes that pDev is a valid pointer to
// an IDirect3DDevice8 interface.
hr = pDev->SetRenderState(D3DRS_SHADEMODE, D3DSHADE_FLAT);
  if(FAILED(hr)) continue
{
   // Code to handle the error goes here.
}

//set to gouraud shading. This is the default for Direct3D.
hr = pDev -> SetRenderState(D3DRS_SHADINGMODE,
                            D3DSHADE_GOURAUD);
If(FAILED(hr))
{
   // Code to handle the error goes here.
}
```

ตาราง 3-1 : ตารางแสดงโค้ดที่ใช้กำหนดโหมดของการให้แสงเงา

3.3.4 เวคเตอร์ผิวหน้าปกติและเวคเตอร์จุดปกติ (Face and Vertex Normal Vectors)

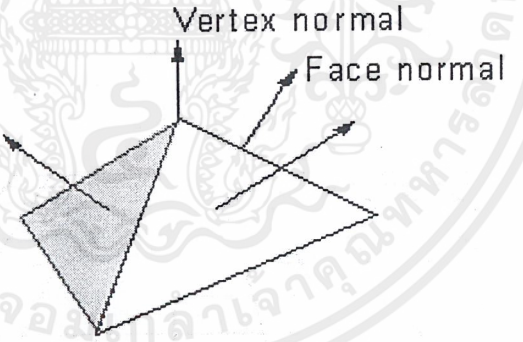
แต่ละผิวหน้าของโพลีกอนจะมีเวคเตอร์ผิวหน้าปกติตั้งฉากอยู่ และทิศทางของเวคเตอร์จะถูกกำหนดโดยลำดับของจุดที่กำหนด และระบบพิกัดฉากว่าเป็นแบบมือซ้ายหรือมือขวา



ภาพที่ 3-8 : ภาพแสดงเวคเตอร์ผิวหน้าปกติ

ด้านที่เหลือของด้านที่ไม่ใช่ด้านหน้าจะเป็นด้านหลังและ Direct3D จะไม่ทำการเรนเดอร์ด้านหลังของพื้นผิวของโพลีกอน ดังนั้นด้านหลังของพื้นผิวจะถูกเรียกว่า Culled (การเรนเดอร์ที่ไม่เรนเดอร์ด้านหลังของพื้นผิว) แต่เราสามารถเปลี่ยนแปลงการเรนเดอร์แบบ Culling Mode ให้ทำการเรนเดอร์ด้านหลังของพื้นผิวได้ถ้าต้องการ

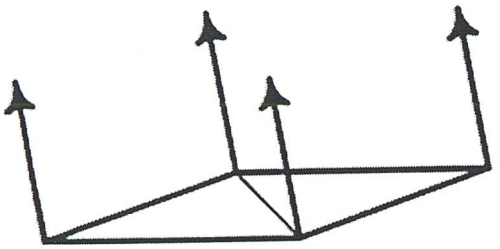
Direct3D ไม่ต้องทำการกำหนดเวคเตอร์ผิวหน้าปกติ เนื่องจากระบบสามารถคำนวณได้เองถ้าหากต้องการใช้งาน เวคเตอร์ผิวหน้าปกติจะใช้ในการให้แสงเงาแบบแบนราบแต่ในการให้แสงเงาแบบ Gouraud จะใช้ เวคเตอร์จุดยอดปกติเพื่อควบคุมแสงและเอ็ฟเฟ็คของพื้นผิววัตถุ



ภาพที่ 3-9 : ภาพแสดงเวคเตอร์ผิวหน้าปกติและเวคเตอร์จุดยอดปกติ

เมื่อใช้การกำหนดแสงเงาแบบ Gouraud กับโพลีกอน Direct3D จะใช้เวคเตอร์จุดยอดปกติในการคำนวณมุมระหว่างแหล่งกำเนิดแสงและพื้นผิวหน้าของวัตถุ โดยจะทำการคำนวณ สี และค่าความเข้มสำหรับแต่ละจุด และทำการกำหนดค่าเหล่านี้ไปที่จุดของแต่ละพื้นผิว Direct3D ทำการคำนวณค่าความเข้มของแสงจากมุม โดยที่มุมยิ่งกว้างแสงจะตกลงที่พื้นผิวน้อย

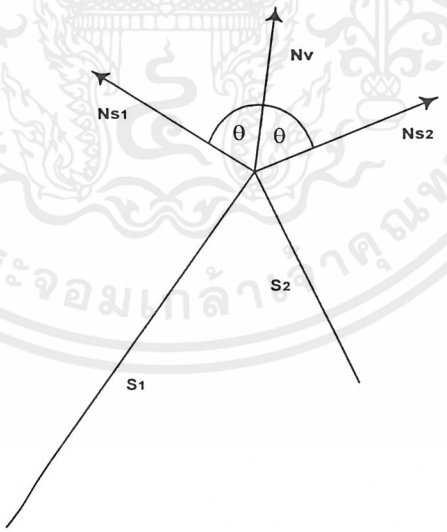
ถ้าหากทำการสร้างวัตถุที่แบนเรียบและตั้งค่าเวคเตอร์ของจุดยอดปกติดังรูป พื้นผิวดังกล่าวจะประ กอบไปด้วยสามเหลี่ยมสองรูป ดังนี้



ภาพที่ 3-10 : ภาพแสดงโพลีกอนและเวคเตอร์จุดยอดปกติ

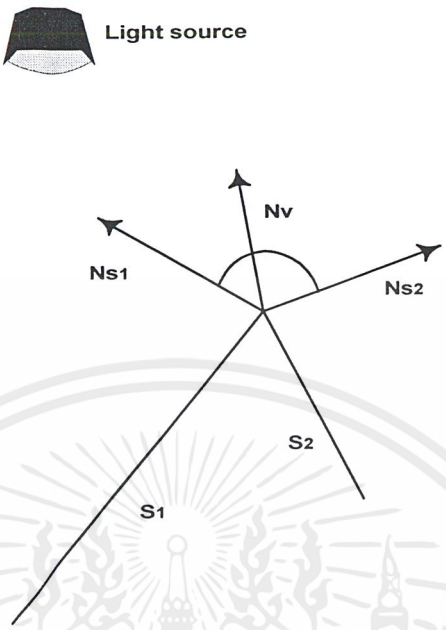
อย่างไรก็ตามวัตถุที่สร้างขึ้นก็จะถูกสร้างขึ้นด้วยสามเหลี่ยมและอยู่คนละระนาบ และวิธีง่ายๆวิธีหนึ่งที่สามารถทำให้การให้แสงเงาระหว่างสามเหลี่ยมราบเรียบคือการคำนวณเวคเตอร์ผิวหน้าปกติของแต่ละพื้นผิวของแต่ละโพลีกอนจากจุดต่างๆ เวคเตอร์จุดยอดปกติสามารถกำหนดให้มุมระหว่างเวคเตอร์ปกติเท่ากันได้ แต่วิธีนี้ก็ไม่มีประสิทธิภาพเพียงพอสำหรับวัตถุที่มีความซับซ้อนมากๆ

วิธีดังกล่าวนี้สามารถแสดงได้ดังรูปด้านล่าง ซึ่งแสดงพื้นผิว 2 พื้นผิว S1 และ S2 ซึ่งมองเห็นขอบจากมุมมองด้านบน และเวคเตอร์ปกติของทั้งสองพื้นผิวเป็นสีน้ำเงิน และเวคเตอร์จุดยอดปกติเป็นสีแดง จะเห็นว่า N เวคเตอร์จุดยอดปกติจะแบ่งมุมระหว่างพื้นผิวทั้งสองให้เท่ากัน และเมื่อพื้นผิวทั้งสองนี้ได้รับการทำการให้แสงเงาแบบ Gouruad ก็จะเกิดความเรียบของแสงและขอบของโพลีกอน



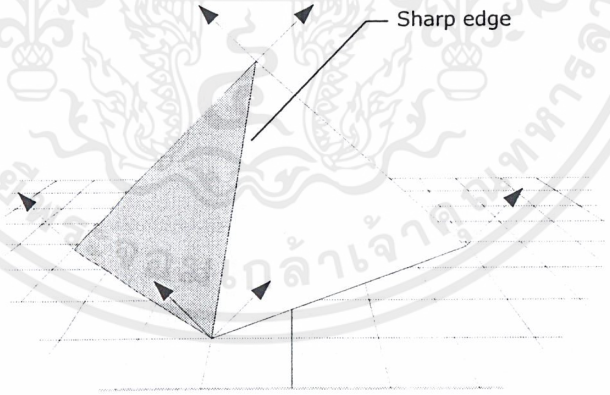
ภาพที่ 3-11 : ภาพแสดงเวคเตอร์จุดยอดปกติที่ทำการแบ่งมุมเป็นสองส่วนเท่าๆกัน

ถ้าหากว่าเวคเตอร์จุดยอดปกติเอียงไปทางด้านใดด้านหนึ่ง จะทำให้ความเข้มของแสงเปลี่ยนไปโดยอาจจะเพิ่มขึ้นหรือน้อยลง โดยขึ้นอยู่กับขนาดของมุม ตัวอย่างดังรูปด้านล่างนี้ ถ้าหากว่าเวคเตอร์จุดยอดปกติเอียงไปทางด้านของ S1 ทำให้มุมมทางด้าน Ns1 มีขนาดน้อยกว่าการแบ่งแบบปกติ



ภาพที่ 3-12 : ภาพแสดงเวคเตอร์จุดยอดปกติที่ทำการแบ่งมุมเป็นสองส่วนไม่เท่ากัน

เราสามารถใช้การสร้างแสงเงาแบบ Gouraud แสดงผลวัตถุบางวัตถุให้มีขอบอย่างเห็นชัดได้ โดยการสร้างเวคเตอร์จุดยอดปกติขึ้นมาอีกหนึ่งเวคเตอร์ที่ทุกๆจุดที่ด้านทั้งสองด้านที่ต้องการให้เห็นขอบมาเจอกัน ดังรูปด้านล่างนี้



ภาพที่ 3-13 : ภาพแสดงโพลีกอนและเวคเตอร์จุดยอดปกติที่มีมากกว่าหนึ่งเวคเตอร์

3.3.5 Triangle Interpolants

เมื่อทำการเรนเดอร์พื้นผิว ระบบจะทำการเพิ่มคุณสมบัติต่างๆเหล่านี้เข้าไปในโพลีกอนสามเหลี่ยม

- สี (Color)
- เงา (Specular)

- ความโปร่งใส (Alpha)

โดยที่การจัดการทั้งหมดจะขึ้นอยู่กับโหมดของการให้แสงเงาดังตาราง

Shading mode	Description
Flat	ไม่มีการทำ interpolation
Gouraud	มีการทำ interpolation อย่างเป็นเชิงเส้นระหว่างจุดสามจุด

ตาราง 3-2 : ตารางแสดงโหมดการให้แสงเงากับการทำ Triangle Interpolants

3.4 แมทริกซ์ และการแปลงพิกัด (Transformations)

Direct 3D ใช้แมทริกซ์ในการกำหนดการแปลง แบบวิว(view), แบบเวิลด์(world) และแบบโปรเจ็กชัน (projection)

- แมทริกซ์
- การแปลงวัตถุสามมิติ (3D tranformation)

3.4.1 แมทริกซ์

แมทริกซ์สำหรับไมโครซอฟต์ Direct3Dจะใช้อยู่ในรูปแบบแมทริกซ์โฮโมจีเนียส 4:4 และมีโครงสร้างเป็นแบบ D3DMATRIX

3.4.2 การแปลงวัตถุสามมิติ

ไมโครซอฟต์ Direct3D ใช้แมทริกซ์ในการทำการแปลงวัตถุสามมิติ ในหัวข้อนี้จะอธิบายเกี่ยวกับการใช้แมทริกซ์กับการแปลงพิกัดวัตถุสามมิติ, อธิบายการแปลงพิกัดวัตถุสามมิติเบื้องต้น, และรายละเอียดเกี่ยวกับการรวมแมทริกซ์หลายๆแมทริกซ์เป็นแมทริกซ์เดียวเพื่อลดการแปลงวัตถุหลายๆครั้ง โดยมีหัวข้อดังนี้

- การเปลี่ยนตำแหน่งวัตถุ(Translation)
- การหมุนวัตถุ(Rotation)
- การเปลี่ยนขนาดวัตถุ(Scaling)
- การรวมแมทริกซ์(Matrix Concatenation)

การแปลงวัตถุในที่นี้จะหมายถึงการจัดการกับวัตถุโดยใช้แมทริกซ์ ซึ่งมีสามชนิดด้วยกันคือ การเปลี่ยนตำแหน่งวัตถุ,กา รหมุนวัตถุ และการเปลี่ยนขนาดวัตถุ

เราสามารถแปลงจุดจุดหนึ่งเป็นจุดใดๆได้โดยการใช้แมทริกซ์ 4x4 โดยในตัวอย่างด้านล่างนี้แสดงการแปลงจุดจาก (x, y, z) ไปเป็นจุด (x', y', z')

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

ภาพที่ 3-14 : ภาพแสดงแมทริกซ์ที่ใช้ในการแปลงพิกัดจุด

ซึ่งมีการทำ operation แบบแมทริกซ์ดังนี้

$$x' = (x \times M_{11}) + (y \times M_{21}) + (z \times M_{31}) + (1 \times M_{41})$$
$$y' = (x \times M_{12}) + (y \times M_{22}) + (z \times M_{32}) + (1 \times M_{42})$$
$$z' = (x \times M_{13}) + (y \times M_{23}) + (z \times M_{33}) + (1 \times M_{43})$$

ภาพที่ 3-15 : ภาพแสดง Operation ของแมทริกซ์กับการแปลงพิกัด

อย่างที่กล่าวมาแล้วข้างต้นว่าการแปลงวัตถุหมายถึง การเปลี่ยนตำแหน่งวัตถุ, การหมุนวัตถุ และการเปลี่ยนขนาดวัตถุ เราสามารถรวมแมทมริกซ์ที่ทำการแปลงทั้งหมดนี้เป็นแมทริกซ์เพียงแมทริกซ์เดียวได้ เพื่อการคำนวณเพียงครั้งเดียว (matrix concatenation)

ในภาษา C++ ไมโครซอฟต์ Direct3D กำหนดแมทริกซ์เป็นอาเรย์สองมิติ โดยใช้โครงสร้างของ D3DMATRIX ตัวอย่างด้านล่างนี้แสดงการกำหนดค่าเริ่มต้นของ D3DMATRIX

```
// In this example, s is a variable of type float.
D3DMATRIX scale ={    s,      0.0f,    0.0f,    0.0f,
                      0.0f,   s,       0.0f,    0.0f,
                      0.0f,   0.0f,    s,       0.0f,
                      0.0f,   0.0f,    0.0f,    1.0f1.0f
                  };
```

ตาราง 3-3 : ตารางแสดงแมทริกซ์ D3DMATRIX

3.4.2.1 การเปลี่ยนตำแหน่งวัตถุ(Translation)

การแปลงต่อไปนี้เป็นการเปลี่ยนตำแหน่งของจุด จากจุด (x, y, z) ไปยังจุด (x', y', z').

$$[x'y'z'1]=[x\ y\ z\ 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ T_x & T_y & T_z & 1\end{bmatrix}$$

ภาพที่ 3-16 : ภาพแสดงแมทริกซ์ที่ใช้ในการเปลี่ยนตำแหน่งวัตถุ

เราสามารถทำการสร้างฟังก์ชันการย้ายจุดได้เองใน C++ โดยตัวอย่างด้านล่างนี้แสดงโค้ดสำหรับฟังก์ชันที่ทำการเปลี่ยนตำแหน่งของจุด

```
D3DXMATRIX Translate(const float dx, const float dy, const float dz)
{
    D3DXMATRIX ret;

    D3DXMatrixIdentity(&ret);  //Implemented by Direct3DX
    ret(3, 0)=dx;
    ret(3, 1)=dy;
    ret(3, 2)=dz;
    return ret;
}    //End of Translate
```

ตาราง 3-4 : ตารางแสดงโค้ดการเปลี่ยนตำแหน่งวัตถุ

3.4.2.2 การหมุนวัตถุ(Rotation)

การแปลงวัตถุในหัวข้อนี้จะอ้างอิงระบบพิกัดสามมิติมือซ้าย โดยตัวอย่างการหมุนด้านล่างนี้เป็นการหมุนจุด (x, y, z) รอบแกน X โดยได้เป็นจุดใหม่ (x', y', z')

$$[x'y'z'1]=[x\ y\ z\ 1]\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos\theta & \sin\theta & 0\\ 0 & -\sin\theta & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

ภาพที่ 3-17 : ภาพแสดงแมทริกซ์ที่ใช้ในการหมุนวัตถุรอบแกน X

ส่วนตัวอย่างนี้เป็นการหมุนจุดรอบแกน Y

$$[x'y'z'1]=[x\ y\ z\ 1]\begin{bmatrix}\cos\theta & 0 & -\sin\theta & 0\\ 0 & 1 & 0 & 0\\ \sin\theta & & \cos\theta & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

ภาพที่ 3-18 : ภาพแสดงแมทริกซ์ที่ใช้ในการหมุนวัตถุรอบแกน Y

และตัวอย่างนี้เป็นการหมุนจุดรอบแกน Z

$$[x'y'z'1]=[x\ y\ z\ 1]\begin{bmatrix}\cos\theta & \sin\theta & 0 & 0\\ -\sin\theta & \cos\theta & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

ภาพที่ 3-19 : ภาพแสดงแมทริกซ์ที่ใช้ในการหมุนวัตถุรอบแกน Z

ตัวอย่างทั้งหมดด้านบนนั้น ตัวอักษร θ จะหมายถึงมุมของการหมุนเป็นเรเดียน โดยการหมุนจะวัดตามเข็มนาฬิกา ในภาษา C++ จะใช้ฟังก์ชัน D3DMatrixRotationX, D3DMatrixRotationY และ D3DMatrixRotationZ เพื่อทำการสร้างแมทริกซ์ โค้ดด้านล่างนี้แสดงฟังก์ชันของ D3DMatrixRotationX

```
D3DXMATRIX*WINAPI D3DXMatrixRotationX(D3DXMATRIX *pOut, float angle)
{
   #if DBG
       if(!pOut)
           return NULL;
   #endif

       float sin, cos;
       sincosf(angle, &sin, &cos);  //Determine sin and cos of angle.

       pOut->_11=1.0f; pOut->_12=0.0f; pOut->_13=0.0f; pOut->_14=0.0f;
       pOut->_21=0.0f; pOut->_22=cos;  pOut->_23=sin;  pOut->_24=0.0f;
       pOut->_31=0.0f; pOut->_32=-sin; pOut->_33=cos;  pOut->_34=0.0f;
       pOut->_41=0.0f; pOut->_42=0.0f; pOut->_43=0.0f; pOut->_44=1.0f;

       return pOut;
}
```

ตาราง 3-5 : ตารางแสดงโค้ดการหมุนวัตถุรอบแกน X

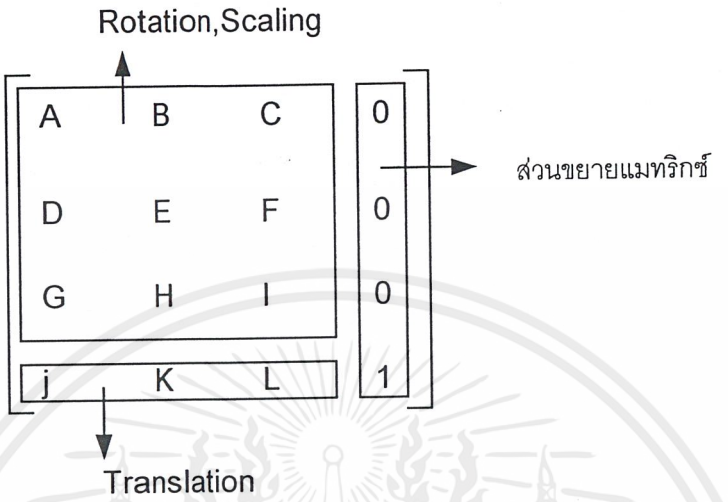
3.4.2.3 การเปลี่ยนขนาดวัตถุ(Scaling)

การแปลงต่อไปนี้เป็นการเปลี่ยนขนาดของจุด (x, y, z) ไปเป็นจุดใหม่ (x', y', z')

$$[x'y'z'1]=[x\ y\ z\ 1]\begin{bmatrix}s_x & 0 & 0 & 0\\ 0 & s_y & 0 & 0\\ 0 & 0 & s_z & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

ภาพที่ 3-20 : ภาพแสดงแมทริกซ์ที่ใช้ในการเปลี่ยนขนาดวัตถุ

จากหัวข้อต่างๆข้างต้นจะเห็นว่าการแปลงพิกัดวัตถุต่างๆนั้นไม่ว่าจะเป็นการเปลี่ยนพิกัดวัตถุ, การหมุนวัตถุ หรือแม้กระทั่งการย่อ/ขยายวัตถุนั้นสามารถกระทำได้โดยใช้แมทริกซ์ทั้งสิ้น โดยสามารถเลือกชนิดของการแปลงพิกัดได้โดยเปลี่ยนค่าต่างๆภายในแมทริกซ์ดังรูป



ภาพที่ 3-21 : ภาพสรุปแมทริกซ์ที่ใช้ในการทำการเปลี่ยนพิกัดวัตถุ

3.5 การรวมแมทริกซ์

ข้อดีอย่างหนึ่งของการใช้แมทริกซ์ คือสามารถรวมการแปลงวัตถุที่มากกว่าสองอย่างได้เป็นเพียงหนึ่งแมทริกซ์ ซึ่งหมายความว่าการทำการหมุนวัตถุและการเปลี่ยนตำแหน่งวัตถุสามารถทำได้โดยไม่ต้องใช้แมทริกซ์สองแมทริกซ์ แต่สามารถทำได้โดยการใช้เพียงแมทริกซ์เพียงแมทริกซ์เดียว ซึ่งวิธีการนี้เรียกว่า Matrix Concatenation โดยสามารถเขียนเป็นสูตรได้ดังนี้

$$C = M_1 \cdot M_2 \cdot M_{n-1} \cdot M_n$$

ภาพที่ 3-22 : ภาพแสดงสูตรการทำ Matrix Concatenation

ในสูตรนี้ C จะหมายถึงแมทริกซ์ที่ต้องการทำการ concatenation และ $M_1 \ldots M_n$ คือแมทริกซ์ที่ใช้ในการทำการแปลงแต่ละอัน

3.6 การสลับเพจ(Page Flipping) และ Back Buffering

การสลับเพจเปรียบเสมือนเป็นเป็นหัวใจหลักของการทำมัลติมิเดีย, อนิเมชัน และเกม โดยการทำการสลับเพจนี้จะเหมือนกับการทำอนิเมชันจากกระดาษ โดยที่เมื่อนักวาดรูปจะทำการเปลี่ยนหน้ากระดาษ

อย่างรวดเร็ว เราจะเห็นเป็นหนังอนิเมชันขึ้นมา ซึ่งการทำการสลับเพจในซอฟต์แวร์ก็เหมือนกับวิธีการที่กล่าวมานี้

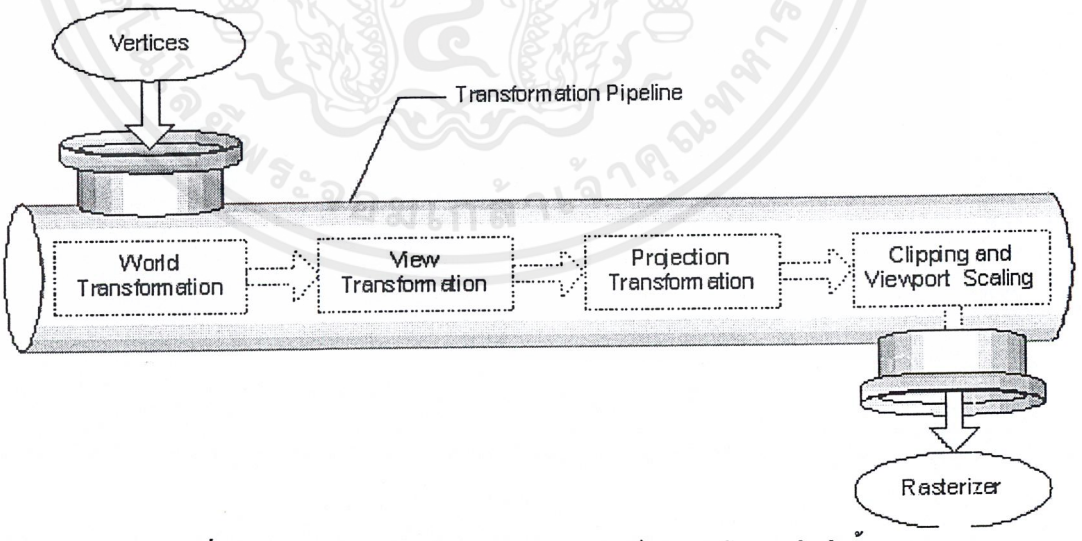
ไมโครซอฟต์ Direct3D ทำการสลับเพจนี้เป็นห่วงโซ่ต่อๆกันไป คือ เริ่มต้นเราต้องทำการกำหนดลำดับของภาพที่ต้องการแสดงออกทางจอภาพ

buffer แรกจะหมายถึง front buffer และ buffer ที่ยังไม่แสดงผลจะหมายถึง back buffer แอปพลิเคชั่นจะทำการวาดภาพลงไปที่ back buffer และทำการสลับมาเป็น front buffer และสลับ front buffer ไปเป็น back buffer ในขณะที่ระบบกำลังแสดงภาพอยู่นั้นเราก็จะทำการวาดภาพลงไปที่ back buffer อีกครั้ง และการทำงานนี้จะทำต่อไปเรื่อยๆจนกว่าแอปพลิเคชั่นจะสั่งให้หยุด

3.7 กระบวนการเรนเดอร์แบบโมเดลท่อส่งน้ำ(3D Rendering Pipeline)

ส่วนที่ทำหน้าที่ส่งอินพุทต่างๆผ่านไปยังฟังก์ชันการทำงานในรูปแบบของท่อส่งน้ำ(pipeline)ของ Direct3D เรียกว่า Transformation Engine ซึ่งฟังก์ชันที่ว่านี้จะหมายถึงการทำให้วัตถุเสมือนมีอยู่จริงในโลก, ทำการแปลงให้สามารถแสดงผลได้ในฉาก, การทำการขริบภาพ(clipping), ปรับขอบเขตการมอง (วิวพอร์ต) รวมทั้งกระบวนการให้แสงและคำนวณแสงเงาด้วย

รูปแบบของท่อส่งน้ำที่ว่านี้จะมีเวอร์เท็กซ์เป็นอินพุท และ Transformation Engine จะทำการแปลงพิกัดสามชนิดคือ การแปลงพิกัดเวิลด์, การแปลงพิกัดวิว, การแปลงพิกัดโปรเจ็กชัน หลังจากนั้นจะทำการขริบแต่งภาพที่ได้ และส่งผลลัพธ์ที่ได้ไปที่ส่วนการจัดการแสดงผลดังแสดงในรูปภาพด้านล่าง



ภาพที่ 3-23 : ภาพแสดงการประมวลผลแสดงเอาท์พุทแบบโมเดลท่อส่งน้ำ

ในส่วนเริ่มต้นของท่อน้ำ อินพุทที่รับเข้ามาจะยังไม่มีการแปลงพิกัดใดๆเกิดขึ้น ดังนั้นจุดของวัตถุต่างๆจะถูกกำหนดอยู่ในรูปแบบของพิกัดของแต่ละโมเดลเอง ซึ่งจะเรียกว่าพิกัดแบบนี้ว่าพิกัดแบบท้องถิ่น (Local Coordinator)

ขั้นตอนแรกของการทำกระบวนการการแปลงพิกัดและให้แสงนี้คือ การแปลงพิกัดแบบท้องถิ่นของแต่ละวัตถุให้มาเป็นระบบพิกัดที่ทุกๆวัตถุใช้ ซึ่งกระบวนการนี้เรียกว่า การแปลงพิกัดแบบเวิลด์ ในขั้นตอนถัดไปเป็นการแปลงพิกัดเวิลด์ไปสู่พิกัดที่มองออกมาจากกล้อง(พิกัดมุมมองกล้อง) ซึ่งเรียกว่าพิกัดวิว หลังจากนั้นคือการทำการแปลงพิกัดโปรเจ็กชัน ในส่วนนี้จะเป็นการกำหนดสัดส่วนและขนาดของวัตถุให้มีความสัมพันธ์กับระยะทางจากผู้มอง โดยที่วัตถุที่อยู่ใกล้จะมีขนาดใหญ่กว่าวัตถุที่อยู่ไกล

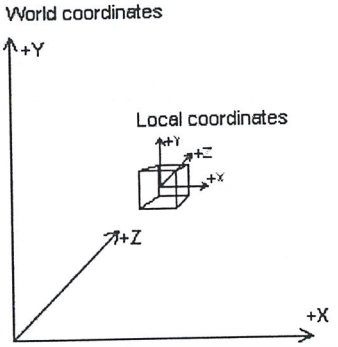
ในส่วนสุดท้ายของท่อน้ำ จุดต่างๆที่มองไม่เห็นจะถูกตัดออกไปเพื่อที่ส่วนแสดงผลจะได้ไม่ต้องทำการคำนวณแสงและสีให้เสียเวลาซึ่งขั้นตอนดังกล่าวนี้เรียกว่าการขริบภาพ(clipping)

3.7.1 การแปลงพิกัดแบบเวิลด์(World Transformation)

ในระหว่างกระบวนการแปรพิกัดและให้แสง พิกัดของวัตถุทั้งหมดที่นำมาเรนเดอร์ต้องแปลงให้มาอยู่ในระบบพิกัดเดียวกัน นั่นคือแปลงมาให้อยู่ในระบบพิกัดเวิลด์ แต่เพื่อความสะดวกของโปรแกรมวัตถุทุกตัวจึงมีพิกัดพิเศษของตนเอง ระบบพิกัดนี้เรียกว่าพิกัดท้องถิ่น ดังนั้นก่อนการเรนเดอร์ต้องทำการแปลงพิกัดก่อนโดยใช้แมทริกซ์ที่มีชื่อว่า แมทริกซ์เวิลด์

การใช้พิกัดท้องถิ่นช่วยให้สามารถทำงานได้ง่ายมากเช่นการเคลื่อนย้ายวัตถุ โดยการแปลงวัตถุจากพิกัดท้องถิ่นไปสู่พิกัดเวิลด์ทำได้ง่ายและเร็วกว่าการแปลงพิกัดทั้งหมดของวัตถุเข้ามาสู่พิกัดเวิลด์ในคราวเดียว นอกจากนี้ยังอนุญาตให้สร้างอินสแตนซ์ได้ด้วย นั่นคือสามารถวาดวัตถุเช่นรูปทรงกลมไว้ที่ตำแหน่งใดตำแหน่งหนึ่ง และยังสามารถวากรูปทรงกลมนี้ได้อีกในตำแหน่งอื่น เพียงแต่จะมีพิกัดของตนเองไม่ใช่พิกัดเดียวกัน ดังนั้นกระบวนการแปลงจากพิกัดท้องถิ่นไปสู่พิกัดเวิลด์จึงต่างกัน การเปลี่ยนแปลงวัตถุภายในพิกัดท้องถิ่นจะทำได้ง่ายมากอย่างเป็นธรรมชาติ เช่นการหมุนวัตถุรอบศูนย์กลางภายในพิกัดท้องถิ่นจะสามารถทำได้อย่างง่ายดาย โดยไม่ต้องคำนึงถึงพิกัดจริงในพิกัดเวิลด์

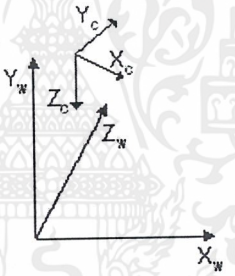
ขั้นตอนแรกในกระบวนการแปลงแบบโมเดลท่อน้ำนี้คือใช้แมทริกซ์เวิลด์ ซึ่งเป็นแมทริกซ์ที่ใช้สำหรับแปลงพิกัดตำแหน่งของเวอร์เท็กซ์ในวัตถุจากพิกัดท้องถิ่นไปสู่พิกัดเวิลด์ นอกจากนี้ยังนำมาใช้ในการหมุน(หมุนวัตถุรอบแกน X, Y หรือ Z), การเปลี่ยนตำแหน่ง(เปลี่ยนตำแหน่งตามแกน X, Y หรือ Z), และการเปลี่ยนขนาด ร่วมกันอีกด้วย รูปภาพด้านล่างนี้แสดงความสัมพันธ์ของระบบพิกัดเวิลด์และระบบพิกัดท้องถิ่น



ภาพที่ 3-24 : ภาพแสดงความสัมพันธ์ระหว่างพิกัดท้องถิ่นและพิกัดเวิลด์

3.7.2 การแปลงพิกัดแบบวิว(View Transformation)

ระบบพิกัดเวิลด์ไม่มีตัวตนให้คุณสัมผัสได้ ไม่ว่าจะเป็นตำแหน่งของจุดกำเนิดและแกนภายในพื้นที่ของระบบพิกัดล้วนถูกควบคุมตามความต้องการของโปรแกรมเมอร์ ดังนั้นจึงต้องมีการแปลงพิกัดแบบวิว การแปลงพิกัดแบบวิวเป็นการสร้างมุมมองเข้าไป โดยการแปลงพิกัดเวิลด์เป็นพิกัดที่มองออกมาจากกล้อง กล้องที่ว่านี้คือกล้องจำลองที่ตั้งอยู่ที่ตำแหน่งจุดกำเนิด นอกจากนี้ในขั้นตอนการให้แสงทั้งหมดก็จะถูกแปลงเข้ามาในระบบพิกัดวิวด้วยดังรูป



ภาพที่ 3-25 : ภาพแสดงความสัมพันธ์ระหว่างพิกัดพิกัดเวิลด์และพิกัดวิว

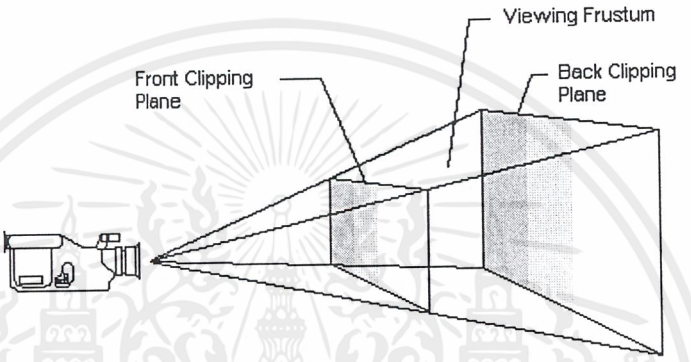
ในขั้นตอนนี้เอฟเฟ็กต์ของแสงแต่ละแบบจะถูกคำนวณผลที่มีต้อเวอร์เท็กซ์ทั้งหมด โดยโค้ดในส่วนของการให้แสงจะพิจารณาจากเวคเตอร์ผิวปกติ, สี และคุณสมบัติของวัสดุที่แปะอยู่บนเวอร์เท็กซ์ ซึ่งเป็นปัจจัยร่วมกับคุณสมบัติของแสงสำหรับคำนวณผลของเอฟเฟ็กต์ที่มีต่อเวอร์เท็กซ์ หลังจากนั้นงานนี้จะเสร็จสิ้นเมื่อจัดเก็บสีของเวอร์เท็กซ์แต่ละตัวที่สำเร็จกระบวนการเหล่านี้แล้วลงบนโครงสร้างข้อมูลของตนเอง ในงานต่อไปของโมเดลท่อน้ำนี้จะไม่ยุ่งเกี่ยวกับแสงและวัสดุของเวอร์เท็กซ์อีก

3.7.3 การแปลงพิกัดแบบโปรเจ็กชัน(Projection Transformation) และ Viewing Frustum

งานต่อไปของขั้นตอนการทำ T&L จะทำการปรับขนาดออบเจ็กต์ในซีนตามความห่างจากจุดที่มองซึ่งเราเรียกว่า การแปลงไปสู่โปรเจ็กชัน(Projection Transformation) กระบวนการนี้ทำให้วัตถุปรากฏตามความลึกภายในซีน โดยการทำให้วัตถุที่อยู่ไกลมีขนาดเล็กกว่าวัตถุที่อยู่ใกล้ หลังเสร็จสิ้นกระบวนการนี้แล้วเวอร์เท็กซ์ทั้งหมดจะถูกแปลงมาอยู่ในพิกัดโปรเจ็กชัน

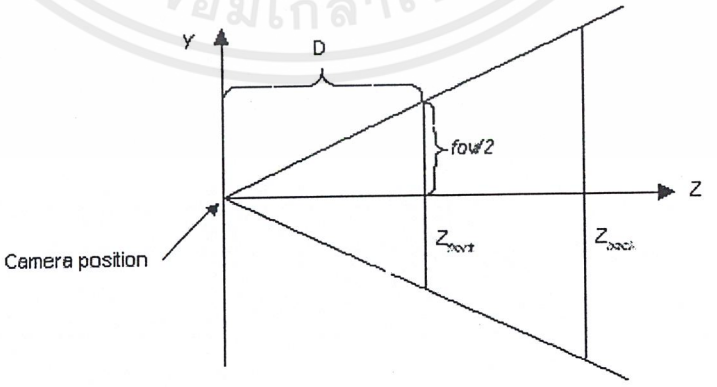
● **Viewing Frustum**

Viewing Frustum จะมองเสมือนว่าเป็นพีรามิด โดยมีส่วนที่เป็นยอดแหลมชนกับกล้องพอดี เมื่อทำการเคลื่อนมุมกล้องจากจุดยอดเข้าไปยังใจกลางพีรามิด กำแพงทั้งสี่ด้านของพีรามิดจะเป็นด้านทั้งสี่ของสกรีนและมีระนาบด้านหน้า(ส่วนยอดพีรามิด) และระนาบด้านหลัง(ส่วนฐานของพีรามิด) ปริมาตรที่อยู่ระหว่างสองระนาบและกำแพงทั้งสี่ด้านนี้เองที่เรียกว่า Viewing Frustum และวัตถุจะถูกมองเห็นได้เมื่ออยู่ในบริเวณนี้เท่านั้น



ภาพที่ 3-26 : ภาพแสดง Viewing Frustum

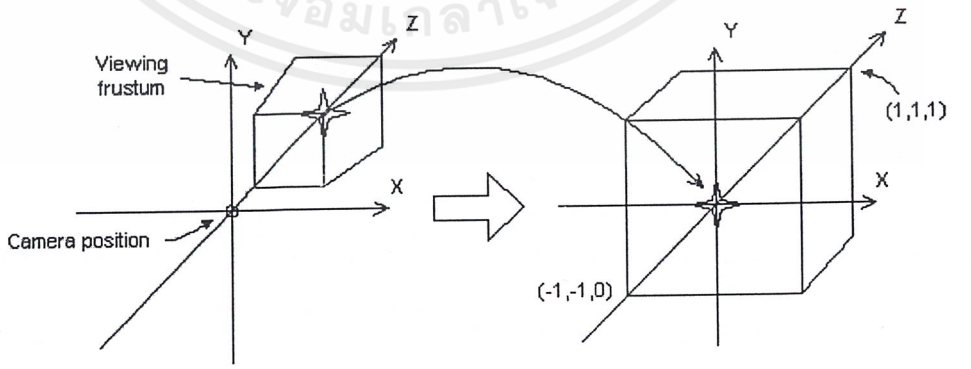
ให้ลองคิดว่าถ้าเรายืนอยู่ในห้องและมองออกไปนอกหน้าต่าง สิ่งที่เราเป็นนั่นคือ Viewing Frustum โดยที่ระนาบด้านหน้าคือหน้าต่าง และระนาบด้านหลังจะเป็นอะไรก็ตามที่อยู่ด้านหลังสุดที่ขัดขวางการมองเห็น เช่นท้องฟ้า หรือภูเขา สิ่งที่คุณมองเห็นจะอยู่ระหว่างหน้าต่างและระนาบด้านหลังนี้เท่านั้น Viewing Frustum จะถูกกำหนดโดย fov(field of view) และระยะทางระหว่างระนาบด้านหน้าและระนาบด้านหลัง



ภาพที่ 3-27 : ภาพแสดง Viewing Frustum และ Front และ Back Plane

ในรูปภาพด้านบนจะเห็นระนาบตัดด้านหน้าและด้านหลังแสดงอยู่ ระนาบตัดทั้งสองด้านใช้กำหนดสิ่งที่จะปรากฏแก่ผู้มองเมื่อเราทำการเรนเดอร์ซีน โดยระนาบด้านหน้าเป็นตัวกำหนดระยะทางใกล้ที่สุดที่วัตถุจะถูกรวบเข้ามาเรนเดอร์ในซีน และระนาบด้านหลังจะเป็นระยะทางไกลที่สุด วัตถุใดๆที่อยู่นอกเหนือพื้นที่ Viewing Frustum จะไม่นำมาเรนเดอร์ ดังนั้นระนาบตัดทั้งสองจึงเป็นสิ่งจำเป็น เพราะเรานำมากำหนดค่าน้อยที่สุดและค่ามากที่สุดให้แก่ Z-Buffer ซึ่งหากไม่มีระนาบในระยะไกลแล้ว อุปกรณ์เรนเดอร์จะไม่สามารถรู้ค่าที่จะแมปเข้ากับค่า Z ที่สูงที่สุด จำไว้ว่าระยะห่างที่กำหนดให้กับระนาบไกล (ระนาบตัดด้านหลัง)จะเป็นสิ่งที่มีผลต่อความเร็วและคุณภาพของซีน ถ้ากำหนดระยะห่างไว้ไกลเกินไปก็จะพบกับอาการสะดุดของภาพ ทำให้วัตถุที่มองเห็นจะมีการกระโดดเข้ามาทันทีมากกว่าจะเห็นว่าวัตถุดังกล่าวเคลื่อนที่เข้ามาหาจากระยะไกล แต่ถ้ากำหนดให้ระนาบไกลมีระยะมากเกินไป จะทำให้เกิดการตัดวัตถุระยะไกลออกไปอย่างไม่มีประสิทธิภาพ ซึ่งก็เป็นสาเหตุของการใช้เวลาในการเรนเดอร์นานด้วยเหมือนกัน

รูปภาพด้านล่างนี้แสดงกระบวนการแปลงไปสู่โปรเจ็กชันที่ใช้ในการแปลง Viewing Frustum ไปเป็นระบบพิกัดใหม่ จะเห็นได้ว่า Frustum กลายเป็นรูป Cuboid(ปริมาตรที่มีลักษณะคล้ายลูกบาศก์ แต่มีด้านไม่เท่ากัน) ที่เป็นเช่นนี้เพราะโปรเจ็กชันที่นำมาใช้คือชนิด Perspective ซึ่งระยะห่างจากวัตถุถึงฉากจะมีผลต่อภาพที่ปรากฏที่ฉากด้วย(ถ้าทำการโปรเจ็กชันแบบ Parallel ระยะห่างจากวัตถุถึงฉากจะไม่มีผล ต่อการโปรเจ็กชัน นั่นคือวัตถุที่มีรูปร่างเหมือนกันแต่ระยะห่างต่างกัน จะโปรเจ็กชันมาได้รูปที่ฉากเท่ากันและเหมือนกัน) หลังจากกระบวนการนี้เสร็จสิ้นลง ขอบเขตของด้านทางแกน X จะมีค่าเท่ากับ –1 สำหรับระนาบด้านซ้าย และ 1 สำหรับระนาบด้านขวา, ขอบเขตของด้านทางแกน Y จะมีค่าเท่ากับ –1 สำหรับระนาบด้านล่าง และ 1 สำหรับระนาบด้านบน และขอบเขตของด้านทางแกน Z จะมีค่าเท่ากับ 0 สำหรับระนาบด้านหน้า และ 1 สำหรับระนาบด้านหลัง



ภาพที่ 3-28 : ภาพแสดงการแปลง Viewing Frustum ไปสู่พิกัดโปรเจ็กชัน

3.7.4 การขริบภาพ (Clipping)

กระบวนการนี้เป็นขั้นตอนที่จะสร้างความมั่นใจว่าวัตถุที่อยู่นอกพื้นที่ของ Viewing Frustum จะไม่ถูกเรนเดอร์ และวัตถุที่อยู่ภายใน Viewing Frustum จะถูกวาดโดยไม่มีไม่มีพิกเซลใดเล็ดลอดออกจากพื้นที่สี่เหลี่ยมที่กำหนดขึ้นโดยวิวพอร์ต(ขอบเขตการมอง)

การขริบภาพนี้เป็นงานหนึ่งในขั้นตอนการทำ T&L ซึ่งต้องการข้อมูลของชิ้นงานกราฟิกที่ต่อเชื่อมกับเวอร์เท็กซ์ เช่นถ้ามีเวอร์เท็กซ์จุดหนึ่งของรูปสามเหลี่ยมหลุดลอดขอบเขตของ Viewing Frustum ออกมา โค้ดที่ใช้ในการขริบภาพจะต้องสามารถกำหนดจุดสองจุด ซึ่งเป็นขอบของรูปสามเหลี่ยมที่อินเตอร์เซ็ก(การนำค่าที่ซ้ำกันมาใช้) อยู่กับ Viewing Frustum และตัดแบ่งรูปสามเหลี่ยมออกเป็นสามเหลี่ยมสองรูป เพราะว่าเส้นร่างของรูปสามเหลี่ยมที่ถูกตัดขณะนี้มีด้านอยู่สี่ด้าน ดังนั้นถ้าหากมีข้อมูลชิ้นส่วนอิมเมจป้อนให้แก่ T&L Pipeline การขริบภาพก็จะสามารถกระทำได้โดยสมบูรณ์ ด้วยเหตุนี้ถ้า T&L Pipeline แปลงพิกัดและให้แสงใน Vertex Buffer (โดยไม่ไปยุ่งเกี่ยวกับข้อมูลของอิมเมจ) ด้วยการเรียกใช้ Process Vertices จะทำให้ T&L Pipeline สามารถระบุและบันทึกเวอร์เท็กซ์ภายนอกระนาบของ Viewing Frustum จากนั้นก็จะนำข้อมูลนี้มาใช้ภายหลังจากเรียกใช้ฟังก์ชัน DrawPrimitive เพื่อใช้ขริบภาพร่วมกับการใช้ Vertex Buffer

3.7.5 การปรับขนาดวิวพอร์ต

กระบวนการนี้เป็นกระบวนการสุดท้ายของการทำการเรนเดอร์แบบโมเดลท่อส่งน้ำนี้ เป็นการปรับเวอร์เท็กซ์ให้มีขนาดหมาะสมกับวิวพอร์ต(ขอบเขตการมอง)ที่ต้องการ วิวพอร์ตอนุญาตให้สามารถกำหนดวิธีที่ใช้ในการแมปภาพลงบนพื้นที่เป้าหมายได้โดยสามารถกำหนดให้มีทั้งการโยกย้ายและการปรับขนาด แต่โดยทั่วๆไปแล้วมักจะเป็นการเติมภาพลงบนสิ่งที่ต้องการเรนเดอร์ ดังนั้นจึงกำหนดให้ไม่มีการโยกย้ายใดๆ จากนั้นก็ปรับขนาดพิกัดของเวอร์เท็กซ์ให้ค่า –1 ของพิกัด X แมปอยู่กับขอบด้านซ้ายและค่า 1 แมปอยู่กับขอบด้านขวา, ค่า –1 ของพิกัด Y แมปอยู่กับขอบด้านบนสุดและค่า 1 แมปอยู่กับขอบด้านล่างสุด, ค่า –1 ของพิกัด Z แมปอยู่กับขอบด้านหน้า และค่า 1 แมปอยู่กับขอบด้านหลัง แต่เราสามารถปรับขนาดในพิกัด Z ได้ ถ้าต้องการเรนเดอร์ให้มีระยะความลึก

บทที่ 4
ไดเร็กเพลย์และระบบเครือข่าย

4.1 การติดต่อสื่อสารผ่านระบบเครือข่ายเบื้องต้น

ในการสร้างเกมแบบหลายผู้เล่น(Multiplayer)นั้น สิ่งสำคัญที่ต้องทำความเข้าใจก่อนอื่นนั่นคือเรื่องของการติดต่อสื่อสารกันระหว่างคอมพิวเตอร์ผ่านระบบเครือข่าย ถึงแม้ว่าเราจะทราบกันแล้วว่าระบบเครือข่ายคืออะไร แต่ในเรื่องของการสร้างเกมแอปพลิเคชั่นแบบหลายผู้เล่นนั้น สิ่งที่เราต้องสนใจมากกว่านั้นก็คือ ระบบเครือข่ายมีผลต่อการพัฒนาเกมอย่างไร ภายในบทนี้จะอธิบายทฤษฎีของระบบเครือข่ายแบบคร่าวๆ ซึ่งมีหัวข้อดังต่อไปนี้

4.1.1 Sockets

Sockets จะหมายถึงการติดต่อผ่านระบบเครือข่ายซึ่งอาจจะเป็น LAN หรืออินเตอร์เน็ตก็ได้ ซึ่ง Sockets Library โดยส่วนมากที่ได้ยินกันคุ้นๆหูก็คือ WinSock WinSock เป็น Library ที่ไมโครซอฟต์จัดเตรียมไว้สำหรับให้โปรแกรมเมอร์สามารถใช้ Sockets ในโปรแกรมต่างๆได้ ซึ่งเกมที่พัฒนาบนระบบปฏิบัติการวินโดวส์กันโดยส่วนมากก็จะใช้ Sockets Library ดังกล่าวนี้ เมื่อทำการใช้งาน Sockets จะมีกฎการควบคุมการใช้งานเรียกว่าโปรโตคอล ซึ่งโปรโตคอลนี้จะเป็นโปรโตคอลอะไรนั้น ก็ขึ้นอยู่กับว่าผู้ใช้งาน Sockets ต้องการใช้งานในรูปแบบใด

สำหรับการพัฒนาเกมแอปพลิเคชั่นนั้นโปรโตคอล TCP/IP ก็เป็นโปรโตคอลที่ใช้งานกันอย่างแพร่หลาย เนื่องจากเป็นโปรโตคอลที่ยืนยันการรับส่ง Packet เมื่อติดตั้ง TCP/IP บนเครื่องแล้ว เครื่องนั้นๆก็จะได้รับหมายเลข IP ไว้

4.1.2 หมายเลข IP

หมายเลข IP จะเป็นตัวกำหนดและบอกให้ทราบว่าคอมพิวเตอร์เครื่องหนึ่งๆนั้นตั้งอยู่ที่ใดภายในระบบเครือข่ายเช่นระบบเครือข่าย LAN หรือระบบเครือข่ายอินเตอร์เน็ต โดยที่หมายเลข IP นั้นจะประกอบด้วยหมายเลขต่างๆสี่ชุดขั้นด้วยเครื่องหมายจุดดัง และแต่ละชุดของหมายเลขจะมีช่วงระหว่าง 0-255 ดังรูป

100.100.100.100

ภาพที่ 4-1 : ภาพแสดงตัวอย่างหมายเลข IP

4.1.3 Packets

Packets คือข้อมูลที่คอมพิวเตอร์ทำการส่งและรับผ่านระบบเครือข่ายกับเครื่องคอมพิวเตอร์อื่นๆ หรืออาจจะกล่าวได้ว่า Packets เป็นตัวกำหนดโครงสร้างของข้อมูลที่ทำการส่งและรับผ่านระบบเครือข่ายนั่นเอง ในการส่งข้อมูลระหว่างเครื่องคอมพิวเตอร์นั้นจะต้องมีการกำหนดรูปแบบของโครงสร้างข้อมูล

ก่อนเพื่อที่เครื่องคอมพิวเตอร์ที่เป็นผู้รับนั้นจะได้ทราบว่าข้อมูลส่วนใดเป็นเนื้อของข้อมูลหรือว่าเป็นส่วนของ Header ซึ่งอาจจะใช้การกำหนดโครงสร้างข้อมูลดังรูป ซึ่งเป็นตัวอย่างการกำหนดโครงสร้างข้อมูลในเกม Tic-Tac-Toe แบบผ่านเครือข่าย

```
#define PACKET_TYPE_MOVE 1

//standard header packet
struct pacStandardHeader
{
   int iPacketType;
   int iPacketLength;
};

//move packet
struct pacMove
{
   pacStandardHeader pacHeader;
   int iHorizontalPosition;
   int iVerticalPosition;
}
```

ตาราง 4-1 : ตารางแสดงโค้ดการกำหนดโครงสร้างของข้อมูลของเกม Tic-Tac-Toe

จากตัวอย่างการกำหนดโครงสร้างข้อมูลดังกล่าว Stream ข้อมูลที่ส่งกันจริงๆสำหรับ Packet นี้จะมีรูปแบบคือ

1 16 2 2

ซึ่งจากข้อมูลข้างต้นอธิบายได้ว่าหมายเลข 1 จะหมายถึงชนิดของ Packets นั่นก็คือเมื่อเครื่องผู้รับได้รับข้อมูลนี้ก็จะทำการมองที่ส่วน Header ก่อนซึ่งผู้รับจะทราบว่า Packets นี้เป็น Packets ที่มีชื่อว่า Move Packets ในส่วนต่อไปของข้อมูลซึ่งก็คือเลข 16 ซึ่งเป็นหมายเลขบอกขนาดของ Packets ดังนั้นเมื่อเครื่องผู้รับได้รับข้อมูล 16 ไบต์แล้ว ก็จะทราบว่าข้อมูลที่ถัดจากนี้เป็นข้อมูล Packets ใหม่ ซึ่งเลข 16 ในที่นี้หมายความว่า Packets Move นี้ประกอบไปด้วยข้อมูลชนิด integer จำนวนสี่ตัว(ในระบบปฏิบัติการแบบ 32 บิต ข้อมูลชนิดจำนวนเต็มมีขนาดสี่ไบต์) ส่วนหมายเลข 2 อีกสองตัวถัดไปจะหมายถึงตำแหน่งของตารางในแนวตั้งและแนวนอนตามลำดับ

4.2 ไดเร็กเพลย์คืออะไร

ไมโครซอฟต์ไดเร็กเพลย์เอพีไอ สามารถรองรับนักพัฒนาให้สามารถขียนโปรแกรมแบบมัลติเพลย์ เช่นเกม หรือห้องสนทนา(Chat Room)ได้ แอปพลิเคชันแบบผู้เล่นหลายคน(Multiplayer Application)นี้มีลักษณะเบื้องต้น 2 ประการคือ

1. มีผู้เล่นสองคนขึ้นไปเล่นอยู่บนเครื่องของตนเอง
2. มีการต่อผ่านระบบเกรือข่ายเชื่อมต่อระหว่างเครื่อง เพื่อให้มีการติดต่อระหว่างกัน โดยอาจจะติดต่อผ่านเครื่องเซิร์ฟเวอร์กลาง

ไดเร็กเพลย์จะทำการสร้างเลเยอร์ขึ้นมาหนึ่งเลเยอร์ ซึ่งแบ่งตัวอย่างเด็ดขาดจากส่วนพื้นฐานของเครือข่ายโดยมีจุดประสงค์หลักเพื่อให้แอปพลิเคชั่นสามารถเรียกใช้งานโดยไม่สนใจชั้นของการติดต่อจริงได้อย่างง่ายดาย และให้หน้าที่ในการจัดการในส่วนปลีกย่อยของการติดต่อระหว่างเครือข่ายเป็นหน้าที่ของไดเร็กเพลย์ ไดเร็กเพลย์มีข้อดีและความง่ายดายในการจัดการ ในส่วนของการพัฒนาแอปพลิเคชั่นแบบมัลติเพลย์ ดังนี้

- นักพัฒนาแอปพลิเคชั่นไม่ต้องสนใจว่าการติดต่อนั้นจะเป็นการติดต่อระหว่างโมเด็มกับโมเด็ม หรือว่าเป็นการติดต่อระหว่างอินเตอร์เน็ตเคเบิล หรืออาจจะเป็นอุปกรณ์ใหม่ๆเนื่องจากไดเร็กเพลย์จะสร้างชั้นของการติดต่อแบบ Abstract ขึ้นมา
- รองรับการสร้างและจัดการกับเซชชันทั้งแบบ peer-to-peer และแบบ client/server
- บริหารและจัดการกับผู้ใช้ที่อยู่ในเซชชัน
- บริหารและจัดการข่าวสาร(Message) ระหว่างสมาชิกในเซชชัน บนระบบเครือข่ายที่ต่างกัน และมีข้อจำกัดของเครือข่ายที่ต่างกัน
- ทำให้แอปพลิเคชั่นสามารถติดต่อกันกับล็อบบี้(Lobby) ได้
- ทำให้ผู้ใช้สามารถติดต่อสื่อสารระหว่างกันโดยใช้เสียงได้

4.3 การสร้างและการจัดการกับเซชชัน

เซชชันเป็นอินสแตนซ์ (Instance) ของเกมแบบมัลติเพลเยอร์ เซชชันจะมีผู้ใช้งานสองคนขึ้นไปทำการเล่นเกมอยู่ในเวลาเดียวกัน โดยทำการเล่นเกมเดียวกัน โดยเกมนั้นๆต้องอยู่บนเครื่องของตนเอง ผู้เล่น (Player) เป็นส่วนประกอบหนึ่ง และถูกกำหนดขึ้นโดยตัวเกม ผู้ใช้งานแต่ละคนอาจจะเป็นผู้เล่นมากกว่าหนึ่งคนในเกมหนึ่งๆ ก็ได้ ซึ่งตัวเกมก็มีหน้าที่ที่จะต้องจัดการเกี่ยวกับผู้เล่นแต่ละคนเอง โดยใช้อินเตอร์เฟซของไดเร็กเพลย์หรือออบเจ็กต์ สำหรับแต่ละผู้เล่น

ขั้นตอนแรกในการสร้างเซชชัน คือการทำการเก็บ(Collect)และจัดตั้งกลุ่มของผู้ใช้งาน โดยมีทฤษฎีเบื้องต้นสองทฤษฎี คือ

1. เกมเซชชันบางเกม ถูกจัดเตรียมโดยล็อบบี้แอปพลิเคชั่น(Lobby Application) ซึ่งรันอยู่ที่รีโมทคอมพิวเตอร์ ซึ่งวิธีนี้จะใช้มากในเกมแบบ Internet-Based
2. อาจจะทำการเตรียมเกมเซชชันโดยการให้เครื่องผู้ใช้แต่ละเครื่องในเครือข่ายติดต่อกัน ซึ่งวิธีนี้จะมีข้อจำกัดอยู่ที่ผู้ใช้ต้องอยู่บน LAN วงเดียวกัน

เมื่อเกมเซชชันได้ถูกสร้างขึ้นมาครั้งหนึ่งแล้ว เกมก็จะเริ่มขึ้น และเมื่อเริ่มเกมไปแล้วอาจจะมีผู้เล่นออกจากเซชชัน หรืออาจจะมีผู้เล่นใหม่เพิ่มเข้ามาในเซชชันก็ได้ ซึ่งรายละเอียดเหล่านี้ขึ้นอยู่กับลักษณะของเกมแต่ละเกม

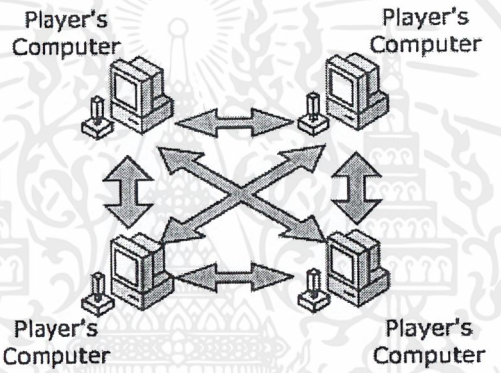
ในเกมแบบมัลติเพลเยอร์ ผู้เล่นแต่ละคนจะต้องมีการเล่นไปพร้อมๆ กับผู้เล่นคนอื่นๆ ในเซชชันเดียวกัน การจัดการเซชชันในเกมมัลติเพลเยอร์จึงต้องการข่าวสารที่ต่อเนื่องสำหรับแต่ละผู้ใช้งาน เช่นทุก

ครั้งที่ผู้เล่นคนหนึ่งมีการเคลื่อนที่ ต้องมีการส่งข่าวสารไปเพื่อเปลี่ยนแปลงตำแหน่งที่อยู่ของผู้เล่นคนนั้นบนหน้าจอของผู้เล่นคนอื่นๆ หน้าที่หลักของไดเร็กเพลย์คือเป็นส่วนหนึ่งของเอพีไอที่ช่วยให้การส่งข่าวสารระหว่างคอมพิวเตอร์ในเซชชันเดียวกันเป็นไปอย่างมีประสิทธิภาพ และมีความยืดหยุ่น

การแบ่งโครงสร้างของวิธีการส่งข่าวสารระหว่างเซชชันได้แบ่งไว้เป็นสองแบบ ซึ่งก็คือชนิด Peer-to-Peer และแบบ Client/Server โดยที่แต่ละแบบนั้นก็มีข้อดีและข้อด้อยที่แตกต่างกับไป ดังนั้นจึงต้องมีการพิจารณาว่าวิธีไหนถึงจะเหมาะกับเกมแอปพลิเคชั่นที่ต้องการพัฒนา

4.3.1 เทคนิคแบบ Peer-to-Peer

เกมแบบ Peer-to-Peer นั้นจะประกอบด้วยเครื่องคอมพิวเตอร์ของผู้เล่นแต่ละคน เชื่อมต่อกันในเครือข่าย โดยมีรูปแบบดังรูป 4-2



ภาพที่ 4-2: การเชื่อมต่อแบบ Peer-to-Peer โดยมีผู้เล่นจำนวนสี่คน

เกมและการเปลี่ยนแปลงต่างๆ จะถูกจัดการโดยการติดต่อกันระหว่างแต่ละไคลเอ็นท์โดยตรง ตัวอย่างเช่น ถ้าหากว่าไคลเอ็นท์คนหนึ่งในเกมมีการเปลี่ยนสเตท[1]ของเกมเช่นมีการเปลี่ยนตำแหน่ง ไคลเอ็นท์เครื่องนั้นจะส่งข่าวสารสามข่าวสารไปบอกเครื่องอีกสามเครื่องในระบบเพื่อบอกให้ทราบ และปรับเปลี่ยนสถานะตาม ข้อดีของเกมแบบ Peer-to-Peer นั้นก็คือ ถ้าหากว่ามีผู้เล่นคนใดคนหนึ่งหยุดการติดต่อเกมนั้นๆจะสามารถเล่นต่อไปได้ เนื่องจากไม่ได้มีผู้เล่นคนใดคนหนึ่งทำการควบคุมเกมไว้ ซึ่งต่างกับเกมแบบ Client/Server

เกมแบบ Peer-to-Peer ปกติจะมีการเรียก และการจัดแบบล็อบบี้ไคลเอ็นท์บนแต่ละเครื่องของผู้ใช้เอง มีหลักเบื้องต้นในการที่ล็อบบี้ไคลเอ็นท์จัดการกับเซชชัน สองวิธีดังนี้

- ล็อบบี้ไคลเอ็นท์ติดต่อโดยตรงล็อบบี้ไคลเอ็นท์ของผู้ใช้คนอื่นๆ ลักษณะแบบนี้สามารถนำไปใช้ได้ เช่น ใช้ในการจัดเกมบน LAN Subnet เดียวกัน

[1] สเตทของเกมจะหมายถึงสถานะของสภาพแวดล้อมของผู้เล่นแต่ละคน ณ เวลาใดเวลาหนึ่ง

- ล็อบบี้ไคลเอ็นท์ทำหน้าที่คล้ายเป็นตัวเชื่อมต่อไปล็อบบี้เซิร์ฟเวอร์ซึ่งทำงานอยู่บนเครื่องคอมพิวเตอร์ปลายทาง ซึ่งลักษณะนี้เป็นลักษณะของเกมบนอินเตอร์เน็ต

เมื่อเซชชันได้มีการสร้าง และทำงานขึ้น ข่าวสารที่มีการส่งส่วนมากจะเป็นระหว่างเครื่องผู้ใช้ไปยังผู้ใช้ด้วยกันเอง ถ้ามีลักษณะเป็นล็อบบี้เซิร์ฟเวอร์ ส่วนของล็อบบี้เซิร์ฟเวอร์จะมีหน้าที่แค่จัดการเมื่อมีผู้เล่นออกจากเกม หรือเมื่อมีผู้เล่นคนใหม่เข้ามาเท่านั้น นอกเหนือจากนี้แล้วก็ไม่ต้องสนใจข่าวสารที่ส่งผ่านไปมา

เนื่องจากว่าเซิร์ฟเวอร์นั้นเหมือนกับว่าไม่มีอยู่ หรืออย่างน้อยที่สุดก็ไม่ได้มีความเกี่ยวข้องกับเกม ผู้เล่นคนหนึ่งอาจจะมีหน้าที่เป็นโฮสต์หรือเป็นเซิร์ฟเวอร์โดยมีหน้าที่แค่นำผู้เล่นคนใหม่เข้ามาสู่เกมเท่านั้น เมื่อผู้เล่นที่ทำหน้าที่เสมือนเป็นโฮสต์นี้ได้ทำการเลิกการติดต่อไป ผู้เล่นคนใหม่ที่ยังอยู่ในเซชชันนี้จะมาทำหน้าที่เป็นโฮสต์ใหม่

เกมแบบ Peer-to-Peer นี้จะมีข้อได้เปรียบที่ความง่ายในการทำ สิ่งที่จำเป็นต้องทำก็คือรวมผู้เล่นเข้ามาแล้วสร้างเซชชันสำหรับผู้เล่นเหล่านี้ ส่วนข้อเสียของลักษณะนี้คือ เมื่อผู้ใช้มีมากขึ้น จำนวนของข่าวสารที่มีการส่งไปส่งมาก็จะมากขึ้น ซึ่งจำนวนที่มากขึ้นนี้ก็จะถูกจำกัดโดยตัวเกมเอง หรือไม่ก็โดยความสามารถของระบบเครือข่าย ซึ่งโดยมากจะไม่เกิน 8-10 คน โดยประมาณ

ข้อดี	ข้อเสีย
1.เกมแอปพลิเคชั่นสามารถทำงานต่อไปได้แม้บางผู้เล่นจะเลิกการติดต่อแล้ว	1.จำนวนผู้เล่นจะถูกจำกัดโดยความสามารถของระบบเครือข่าย
2.สามารถพัฒนาได้ง่าย	2.เกิดความคับคั่งในระบบเครือข่าย
3.ไม่ต้องการเซิร์ฟเวอร์ที่มีความเร็วสูงมาเป็นส่วนหนึ่งของระบบ	3.มีความเสี่ยงต่อข้อมูลที่ได้รับสูง

ตาราง 4-2 : ตารางเปรียบเทียบข้อดีข้อเสียของเกมแอปพลิเคชั่นแบบ Peer-to-Peer

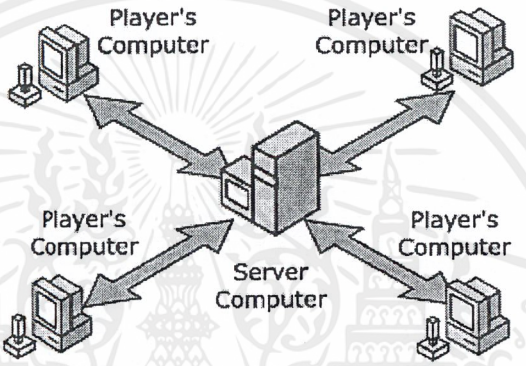
<u>ขั้นตอนการทำงานของเกมแบบ Peer-to-Peer มีขั้นตอนดังนี้</u>

1. ผู้เล่น A ทำการเริ่มเล่นเกมแบบหลายผู้เล่น และมีหมายเลข ID เริ่มต้นเป็นเลข “0”
2. ผู้เล่น B ทำการเริ่มเล่นเกมโดยขอร่วมเล่นกับผู้เล่น A และจะทำการร้องขอข้อมูลหมายเลข ID และข้อมูลต่างๆที่เกี่ยวกับผู้เล่น A และเกม
3. ผู้เล่น A ส่งข้อมูลหมายเลข ID หมายเลข “1” ให้ผู้เล่น B รวมทั้งข้อมูลต่างๆที่เกี่ยวกับผู้เล่น A เพียงแค่นั้นเนื่องจากไม่มีผู้เล่นคนอื่นอีก
4. ผู้เล่น C ทำการเข้าร่วมเกมกับผู้เล่น A และทำการร้องขอหมายเลข ID
5. ผู้เล่น A ทำการส่งหมายเลข ID ซึ่งมีค่าเป็น “2” ให้กับผู้เล่น C รวมทั้งข้อมูลเกี่ยวกับผู้เล่น A และผู้เล่น B

6. ผู้เล่น C ทำการส่งข้อมูลเกี่ยวกับตัวเองไปที่ผู้เล่น A และ B ดังนั้นผู้เล่น B จะสามารถรู้ได้ว่ามีผู้เล่น C เข้ามาขอร่วมเกมแล้ว
7. ผู้เล่นทั้งสามคนสามารถส่งข้อมูลไปมากันได้เนื่องจากทราบหมายเลขและข้อมูลของแต่ละเครื่องแล้ว

4.3.2 เทคนิคแบบ Client/Server

เทคนิคแบบ Client/Server นี้ จะประกอบด้วยเครื่องคอมพิวเตอร์ของผู้เล่นแต่ละคน และทำการเชื่อมต่อเข้ากับเครื่องคอมพิวเตอร์เซิร์ฟเวอร์ ดังรูป 4-3



ภาพที่ 4-3: การเชื่อมต่อแบบ Client/Server โดยมีผู้เล่นจำนวนสี่คน

การเปลี่ยนแปลงของเกม ซึ่งรวมถึงสเตทต่างๆ จะกระทำโดยผู้เล่นแต่ละคนจะติดต่อกันผ่านเครื่องเซิร์ฟเวอร์ และเซิร์ฟเวอร์มีหน้าที่ผ่านข่าวสารต่างๆไปยังทุกคน เมื่อมีผู้เล่นคนหนึ่งเคลื่อนที่ จะมีการส่งข่าวสารไปยังเซิร์ฟเวอร์ และเซิฟเวอร์จะส่งข่าวสารนี้ไปให้ผู้เล่นคนอื่นๆ โดยเซิร์ฟเวอร์ จะทำหน้าที่ต่างๆ ดังนี้

- ทำหน้าที่เหมือนเป็นตัวส่งผ่านข้อมูล โดยหากว่ามีเครื่องคอมพิวเตอร์เครื่องใดต้องการส่งข่าวสารก็เพียงแต่ทำการส่งข่าวสารดังกล่าวไปที่เครื่องเซิร์ฟเวอร์ แล้วเครื่องเซิร์ฟเวอร์จะทำการกระจายข้อมูลไปยังเครื่องอื่นให้ วิธีดังกล่าวนี้จะช่วยลดความคับคั่งของข้อมูลได้ โดยเฉพาะกับเกมใหญ่ๆ
- ทำหน้าที่เสมือนเป็นโฮสต์ในเทคนิคแบบ Peer-to-Peer
- ทำหน้าที่จัดการส่วนแสดงผลของเกม เกมที่มีขนาดใหญ่ส่วนมากการจัดการ และ Process ต่างๆ จะถูกนำไปไว้ยังเครื่องเซิร์ฟเวอร์ ส่วนเครื่องไคลเอ็นท์จะทำหน้าที่แสดงผลเท่านั้น

เกมแบบ Client/Server นั้นโดยปกติแล้วจะถูกจัดการ และเรียกใช้งานผ่านล็อบบี้ไคลเอ็นท์ ที่อยู่ในเครื่องของผู้ใช้งาน ล็อบบี้ไคลเอ็นท์นั้นจะทำหน้าที่เชื่อมต่อไปยังล็อบบี้เซิร์ฟเวอร์ ซึ่งโดยปกติแล้วจะทำงานอยู่บนเครื่องเดียวกับเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็นโฮสต์ของเกม เมื่อเกมถูกเรียกขึ้นมาทำงานเครื่องเซิร์ฟเวอร์จะกลายเป็นโฮสต์ของเกม และทำงานต่างๆ เช่นการจัดการเมื่อมีผู้เล่นเข้ามาใหม่เป็นต้น

ข้อดี	ข้อเสีย
1. ประสิทธิภาพของการทำงานดีกว่า โดยเฉพาะอย่างยิ่งกับเกมที่มีขนาดใหญ่ เนื่องจากผู้เล่นที่เพิ่มเข้ามาจะทำให้เกิดการเพิ่มขึ้นของ Message ในระบบอย่างเป็นเชิงเส้น(Linear Increase)	1.ฮาร์ดแวร์ที่ใช้เป็นเซิร์ฟเวอร์ต้องใช้เครื่องที่มีประสิทะภาพสูง ซึ่งมีราคาแพง
2. เกมจะไม่ถูกจำกัดโดยประสิทธิภาพของเครื่องคอมพิวเตอร์ของผู้ใช้ โดยอาจจะมีการกำหนดให้ Process ส่วนใหญ่ของเกมไปทำงานบนเครื่องที่ประสิทธิภาพสูง และให้เครื่องของผู้ใช้ทำหน้าที่แค่แสดงผล	2.ต้องใช้ความเร็วในระบบเครือข่ายมากกว่าแบบ Peer-to-Peer
3.สามารถกำหนดลักษณะของเกมได้บนเซิร์ฟเวอร์ เช่น อาจจะมีการแก้ไขปรับปรุงเกมอยู่ตลอด ซึ่งจะเป็นการยากถ้าต้องแก้ให้ทุกเครื่องของผู้ใช้ ดังนั้นถ้าแก้ไขแค่ที่เครื่องเซิร์ฟเวอร์ ก็จะสามารถทำได้ง่ายกว่า	3.ถ้าหากว่าเครื่องเซิร์ฟเวอร์หลุดการติดต่อกับระบบไป เกมนั้นๆจะไม่สามารถเล่นได้เลย

ตาราง 4-3 : ตารางเปรียบเทียบข้อดีข้อเสียของเกมแอปพลิเคชั่นแบบ Client/Server

อย่างไรก็ตาม เมื่อได้มีการพัฒนาเกมเป็นแบบ Peer-to-Peer แล้ว ก็ถือว่าประสบความสำเร็จแล้ว เนื่องจากความสามารถของเครื่องผู้ใช้ก็พอเพียงแล้ว แต่แบบ Client/Server นั้นจำเป็นจะต้องมีการตกลงกับผู้ใช้ทุกๆ คน ต้องมีการจัดการทั้งหมดบนเครือข่าย เพื่อที่จะมีการจัดการทุกๆข่าวสาร ถ้าเกมเป็นเกมใหญ่มากๆ แล้วอาจจะต้องมีการทำงานโดยไม่มีการหยุดเลยก็เป็นได้ และยังต้องมีการควบคุมความผิดพลาดที่ทำโดยผู้ใช้อีกด้วย

<u>ขั้นตอนการทำงานของเกมแบบ Peer-to-Peer มีขั้นตอนดังนี้</u>

1. ผู้เล่น A ทำการส่ง Packets ของการกระทำ(Action Packets) เช่นตัวละครมีการเลี้ยวขวาไปยังเครื่องเซิร์ฟเวอร์
2. เครื่องเซิร์ฟเวอร์ได้รับ Packets ของการกระทำจากผู้เล่น A และส่งข่าวสารยืนยันการได้รับ Packets ไปบอกผู้เล่น A
3. เครื่องเซิร์ฟเวอร์ทำการปรับเปลี่ยนเกมสเตทด้วยข้อมูลที่ได้รับมา
4. เครื่องเซิร์ฟเวอร์ทำการส่ง Packets ไปยังเครื่องไคลเอ็นท์ทุกเครื่องที่ทำการเชื่อมต่ออยู่กับเซิร์ฟเวอร์ เพื่อบอกให้ทราบว่าผู้เล่น A ทำการเลี้ยวขวา
5. เครื่องไคลเอ็นท์ทั้งหมด(ไม่รวมเครื่องของผู้เล่น A) จะทำการเปลี่ยนเกมสเตทด้วยข้อมูลที่ไคลเอ็นท์ส่งมา

4.4 ไดเร็กเพลย์กับการสื่อสารในระบบเครือข่าย

หน้าที่หลักของไดเร็กเพลย์คือ การรองรับการจัดการส่งข่าวสารได้อย่างยืดหยุ่นและมีประสิทธิภาพ ถ้าหากว่าต้องการส่งข่าวสารไปทำการเปลี่ยนสถานะ(Status Update) ก็สามารถทำได้อย่างง่ายดาย โดยการเรียกผ่านเอพีไอของไดเร็กเพลย์ โดยไม่สนใจว่าใช้ระบบเครือข่ายชนิดใด ไดเร็กเพลย์นั้นรองรับการติดต่อทั้งบน TCP/IP, IPX, Modem และการติดต่อโดยตรง

4.4.1 โปรโตคอลการสื่อสารของไดเร็กเพลย์ (DirectPlay Transportation Protocol)

หัวใจของความสามารถของไมโครซอฟต์ไดเร็กเพลย์คือไดเร็กเพลย์โปรโตคอล ซึ่งชั้นของโปรโตคอลการสื่อสาร(Transport-Layer Protocol) นี้ได้รับการปรับปรุงใหม่ทั้งหมดในไดเร็กเอ็กซ์เวอร์ชัน 8 นี้ และสามารถใช้ได้กับการสื่อสารทุกรูปแบบ ไดเร็กเพลย์โปรโตคอลนี้มีจุดประสงค์เพื่อทำให้การส่งผ่านข้อมูลระหว่างต้นทางกับปลายทางสามารถทำได้อย่างง่ายดายโดยไม่ต้องสนใจว่ามีการกระทำอย่างไรในขั้นตอนเหล่านั้น โปรโตคอลดังกล่าวนี้จะมีหลายๆทางเลือกซึ่งจำเป็นต่อเกมประเภทหลายผู้เล่นให้เลือกได้แก่

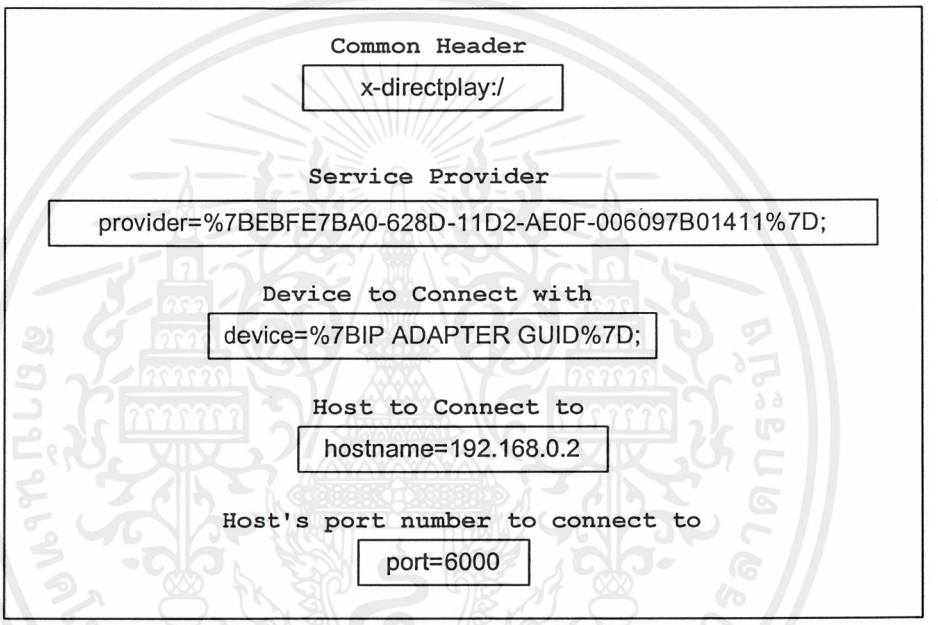
- การส่งข่าวสารแบบเชื่อถือได้(Reliable) และเชื่อถือไม่ได้ (Unreliable) การส่งแบบเชื่อถือได้จะมีการรับรองจนกว่าปลายทางจะได้รับข้อมูล
- การสื่อสารแบบต่อเนื่อง (Sequential) และแบบไม่ต่อเนื่อง (non-Sequential) การส่งแบบต่อเนื่องจะส่งไปยังปลายทางตามลำดับที่ได้รับมา
- การแบ่งข่าวสารออก (Fragmentation) และการประกอบเป็นแบบเดิม (Reassemble) ถ้าขนาดของข่าวสารเกินกว่าความสามารถของเครือข่ายที่จะส่งไปได้ไดเร็กเพลย์จะทำการแยกออกเป็นส่วนๆ ให้ แล้วจึงนำมาประกอบกันใหม่เมื่อถึงปลายทาง
- การควบคุมความคับคั่งของข้อมูล (Congestion Control) ไดเร็กเพลย์จะทำการควบคุมข่าวสารที่ส่งออกไปให้อยู่ในระดับที่เครื่องปลายทางสามารถจะรับได้ ลักษณะนี้จะช่วยป้องกันไม่ให้ข่าวสารล้นเกินกว่าจะสามารถประมวลผลทันได้ที่เครื่องปลายทาง
- ระดับความสำคัญของข้อมูล (Send Prioritization) เพื่อเป็นการรับประกันว่าข่าวสารที่สำคัญมากจะได้รับการส่งไปก่อน ไดเร็กเพลย์จะให้คุณสามารถจัดระดับความสำคัญของข่าวสารที่ส่งไปเป็นแบบ ต่ำ ปานกลาง และสูงได้ ข่าวสารที่สำคัญจะได้รับการส่งไปก่อน
- ระยะเวลาในการส่งข้อมูล (Message Timeouts) เพื่อป้องกันไม่ให้เราต้องรอข่าวสารตอบรับจากเครื่องปลายทาง หากว่ามีการผิดพลาดเนื่องจากข้อมูลไปไม่ถึงปลายทาง ไดเร็กเพลย์จะอนุญาตให้มีการกำหนดระยะเวลาในการรอไว้ได้ หากเกินเวลาแล้วข่าวสารที่จะส่งจะถูกนำออกจาก Queue โดยไม่สนใจว่าส่งไปแล้วหรือยัง

4.4.2 การอ้างที่อยู่ของไดเร็กเพลย์ (DirectPlay Addresses)

ในการส่งข่าวสาร ผู้ที่อยู่ในระบบต้องมีแอดเดรสที่ไม่ซ้ำกัน ซึ่งแอดเดรสดังกล่าวอาจจะอ้างตามเครื่องคอมพิวเตอร์ที่กำลังทำการรันแอปพลิเคชั่นนั้น (Device Address) หรือเครื่องคอมพิวเตอร์ที่ต้องการติดต่อสื่อสารด้วย (Host Address)

แอดเดรสในไดเร็กเพลย์นั้นจะอยู่ในรูปแบบของข้อความ URL โดยข้อความดังกล่าวจะประกอบไปด้วย Header ,ตัวแบ่ง และข้อความที่ประกอบด้วยข้อมูลต่างๆ ในรูปแบบข้างล่างนี้

x-DirectPlay:/ [data string]



ภาพที่ 4-4 : ส่วนต่างๆในข้อความที่ประกอบกันเพื่ออ้างถึงแอดเดรสของไดเร็กเพลย์

4.4.3 การติดต่อสื่อสารกันด้วยไดเร็กเพลย์ออบเจ็กต์ (Communication with DirectPlay Objects)

ไมโครซอฟต์ไดเร็กเพลย์นั้นประกอบไปด้วย COM ออบเจ็กต์หลายๆตัว โดยแต่ละตัวนั้นมีอินเตอร์เฟซหนึ่งอินเตอร์เฟซหรือมากกว่าเพื่อให้เราทำการจัดการกับส่วนต่างๆของไดเร็กเพลย์ได้เช่น DirectPlay peer Object(CLSID_DirectPlay8Peer) ใช้สำหรับจัดการเกมแบบ Peer-to-Peer

เราสามารถทำการติดต่อกับออบเจ็กต์ของไดเร็กเพลย์ได้โดยการเรียกเมธอดต่างๆผ่านทางอินเตอร์เฟซที่มีการจัดเตรียมไว้ให้ ตัวอย่างเช่น ถ้าหากต้องการทำการส่งข้อมูลไปหาผู้เล่นคนอื่นๆที่อยู่ในระบบเกมแบบ Peer-to-Peer นั้น เราสามารถกระทำได้โดยการเรียกเมธอด IdirectPlay8Peer: SendTc แล้วไดเร็กเพลย์จะทำการจัดการส่งข้อมูลให้

ไดเร็กเพลย์ทำการติดต่อกับเกมหรือแอปพลิเคชั่นผ่านทางสิ่งที่เรียกว่า Callback Function ที่อาจจะมีมากกว่าหนึ่งได้ ฟังก์ชันนี้จะคล้ายกับ Window Procedure เกมหรือแอปพลิเคชั่นจะมีการจัดการกับ Callback Function และทำการส่งค่า Pointer ที่ชี้ไปยังฟังก์ชันนั้นไปให้กับไดเร็กเพลย์ในระหว่างช่วงของ

การเริ่มต้น(Initialization) และเมื่อไดเร็กเพลย์ต้องการทำการติดต่อกับเกมหรือแอปพลิเคชั่นนั้น มันก็จะทำการเรียก Callback Function นี้และทำการส่งค่าข้อมูลสองชนิดดังนี้

- ID ของข่าวสาร (Message ID) ซึ่งทำหน้าที่กำหนดชนิดของข่าวสาร
- Pointer ที่ชี้ไปที่บล็อกของข้อมูล ปกติจะเป็นโครงสร้างที่มีการเก็บข้อมูลที่ต้องการไว้

ตัวอย่างเช่นเมื่อข่าวสารต่างๆตามตัวอย่างด้านบนที่กล่าวมานั้นได้ไปถึงเครื่องเป้าหมายที่ต้องการส่งข้อมูล Callback Function ของเกมหรือแอปพลิเคชั่นทางฝั่งเครื่องที่ได้รับข่าวสารจะได้รับข้อความกับ DPNMSGID_RECEIVE ซึ่งเป็น Message ID เพื่อบอกว่าข่าวสารที่มาถึงนั้นมาจากผู้เล่นคนอื่น

4.4.4 ไดเร็กเพลย์กับการรองรับการใช้ล็อบบี้

ล็อบบี้คือแอ็พพลิเคชันซึ่งมีจุดมุ่งหมายหลักคือการทำให้ผู้เล่นเกมสามารถเข้ามาเล่นเกมร่วมกันได้(A lobby is an application whose primary purpose is to enable players to meet and arrange games.) ซึ่งโดยปกติมันจะอยู่ที่เครื่องรีโมทคอมพิวเตอร์และสามารถเข้าถึงได้โดยอินเตอร์เน็ต ส่วนเครื่องล็อบบี้เซิร์ฟเวอร์นั้นมีหน้าที่ในการทำงานหลายอย่าง เช่น เป็นโฮสต์ของห้องสนทนา(Chat Room) หรือเป็นกระดานข่าวประกาศข่าวสาร

ถึงแม้ว่าการใช้งานล็อบบี้เซิร์ฟเวอร์จะเป็นที่นิยมและสะดวกสบายในการใช้งาน แต่ทว่าในการเล่นเกมแบบมัลติเพลเยอร์นั้นไม่จำเป็นต้องใช้ การเล่นเกมแบบมัลติเพลเยอร์นั้นสามารถใช้งานจัดตั้งได้โดยการติดต่อกันโดยตรงระหว่างล็อบบี้ไคลเอ็นท์

ส่วนประกอบสามอย่างที่จำเป็นสำหรับการทำให้เกมสามารถสื่อสารกับเกมล็อบบี้ได้

1. เซิร์ฟเวอร์ล็อบบี้ (Lobby Server)
2. ไคลเอ็นท์ล็อบบี้ (Lobby Client)
3. เกมที่รองรับการใช้งานล็อบบี้(Lobbyable Game)

ไมโครซอฟต์ไดเร็กเพลย์นั้นไม่ได้กำหนดว่าล็อบบี้เซิร์ฟเวอร์แอปพลิเคชั่นควรจะมีการพัฒนาอย่างไร ในทางตรงกันข้ามไดเร็กเพลย์จะมีการรองรับสำหรับล็อบบี้ไคลเอ็นท์ ซึ่งล็อบบี้ไคลเอ็นท์นี้จะถูกพัฒนาขึ้นโดยผู้ขายล็อบบี้เซิร์ฟเวอร์ และจะถูกติดตั้งลงไปในเครื่องผู้ใช้แต่ละเครื่อง โดยทำหน้าที่เชื่อมต่อระหว่างผู้ใช้กับล็อบบี้ ขณะที่เราสามารถจัดการกับการติดต่อสื่อสารได้โดยตรงนั้นโปรแกรมจำเป็นต้องรู้รายละเอียดของข้อมูลทั้งหมดที่เกี่ยวกับทุกๆล็อบบี้ที่ใช้ในการเปิดเกม

โปรแกรมในล็อบบี้ไคลเอ็นท์จะควบคุมข้อมูลเกี่ยวกับการติดต่อสื่อสารกับล็อบบี้เซิร์ฟเวอร์ของตัวเองโดยใช้โปรโตคอลที่เหมาะสม ล็อบบี้ไคลเอ็นท์จะติดต่อกับผู้เล่นและเกมแอปพลิเคชั่นผ่านอินเตอร์เฟซของไดเร็กเพลย์ ส่วนไดเร็กเพลย์จะส่งข่าวสารไปยังแอปพลิเคชั่น และแอปพลิเคชั่นก็สามารถใช้อินเตอร์เฟซของไดเร็กเพลย์ เพื่อส่งข่าวสารไปยังล็อบบี้ไคลเอ็นท์ได้

แนวโน้มในเกมหลายผู้เล่นแบบเล่นเป็นทีมต้องมีการติดต่อกันระหว่างผู้เล่นกับผู้เล่น (Player-to-Player) เป็นสำคัญในระหว่างเกม ซึ่งในสมัยก่อนมีการติดต่อกันผ่านทางระบบตัวอักษร (Text-Based Communication) ผู้เล่นจะพิมพ์ตัวอักษรเพื่อติดต่อกับเพื่อนร่วมทีม ซึ่งในลักษณะนี้จะเหมาะสำหรับเกมที่เล่นเป็นเทิร์นแต่ถ้าเป็นเกมแบบ Real-Time แล้วนอกจากบางคนพิมพ์ได้ช้าย่อมติดต่อกันได้ช้า แต่ยังทำให้

โดยรวมแล้วเกมจะช้าลงด้วย การแก้ปัญหานี้ทำได้โดยใช้คำพูดในการสื่อสารกันแทน ไม่จำเป็นต้องมีการฝึกฝน และเพิ่มการจดจ่อในเกม

วินโดวส์จะมีการจัดการเกี่ยวกับเรื่องการติดต่อแบบ Real-Time อยู่แล้วสำหรับผู้เขียนโปรแกรม แต่ผู้พัฒนาจำเป็นต้องใช้ความพยายามสูงมากในการเขียนโปรแกรม ซึ่งเป็นการสิ้นเปลือง และยากต่อการทำ และจัดการ ซึ่งในที่นี้ DirectPlay ได้จัดการทั้งหมดให้แล้ว เพื่อให้ง่ายต่อการใช้

4.5 การติดต่อกับไดเร็กเพลย์ล็อบบี้

โปรแกรมล็อบบี้เป็นโปรแกรมที่มีวัตถุประสงค์ในการช่วยผู้เล่นจัดเกมประเภทหลายผู้เล่น โดยมากล็อบบี้จะเป็นโปรแกรมที่ Deploy ไว้ที่รีโมทเซิร์ฟเวอร์ ผู้ใช้จะติดต่อไปยังล็อบบี้ดังกล่าวผ่านทางอินเตอร์เน็ต และทำการสร้างเซชชันของเกม หรือเข้าร่วมเซชชันของเกมที่ทีผู้อื่นสร้างไว้แล้ว

เนื่องจากเกมแบบหลายผู้เล่นส่วนมากจะจัดตั้งกลุ่มของผู้เล่นผ่านทางล็อบบี้แอปพลิเคชัน ดังนั้นเกมต่างๆที่ทำการพัฒนาโดยใช้งานไดเร็กเพลย์นั้นต้องสามารถติดต่อกับล็อบบี้แอปพลิเคชันได้ ในหัวข้อนี้จะทำการอธิบายถึงล็อบบี้ดังนี้

4.5.1 สถาปัตยกรรมของไดเร็กเพลย์ล็อบบี้

4.5.2 เซิร์ฟเวอร์ล็อบบี้

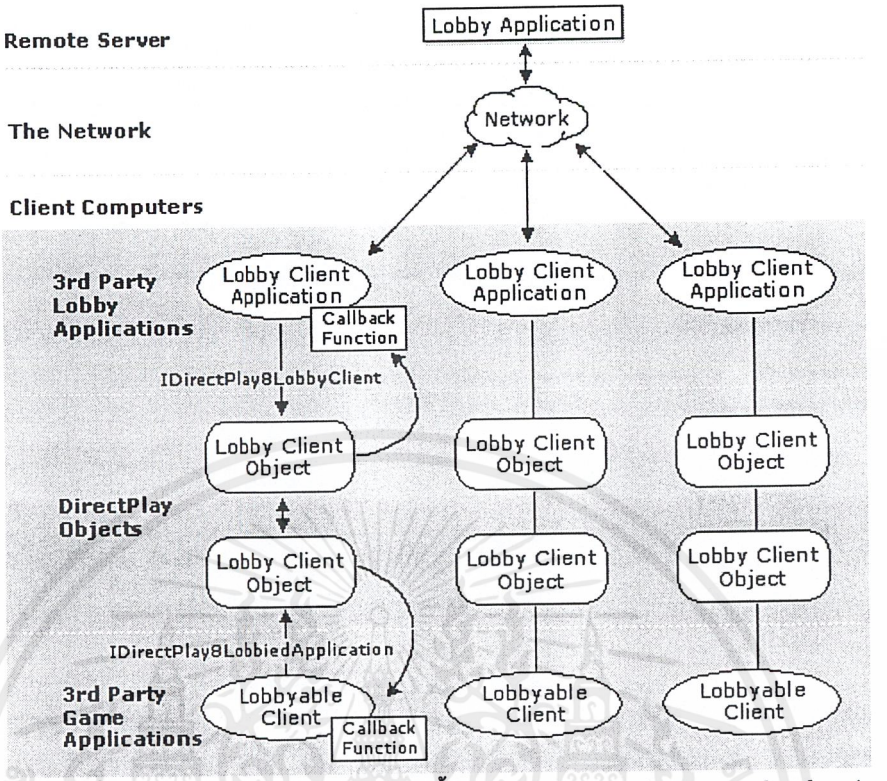
4.5.3 ไคลเอ็นท์ล็อบบี้

4.5.1 สถาปัตยกรรมของไดเร็กเพลย์ล็อบบี้

กระบวนการในการจัดตั้งและจัดการเซชชันของเกมประเภทหลายผู้เล่นซึ่งพัฒนาโดยใช้ไดเร็กเพลย์นั้นเกี่ยวเนื่องกับส่วนประกอบ 5 ส่วน โดยส่วนแรกคือโปรแกรมล็อบบี้(ซึ่งอาจจะเป็น Third Party ล็อบบี้แอปพลิเคชั่นได้) จะ Deploy ไว้ที่รีโมทเซิร์ฟเวอร์และสามารถติดต่อเข้าถึงได้ผ่านทางอินเตอร์เน็ต ส่วนอีกสี่ส่วนที่เหลือนั้นจะมีการติดตั้งไว้ที่เครื่องของผู้ใช้แต่ละเครื่องดังนี้

1. ไคลเอ็นท์ล็อบบี้ เป็นส่วนที่จะทำการติดต่อกับเซิร์ฟเวอร์ล็อบบี้
2. เกมแอปพลิเคชั่นที่รองรับการใช้งานล็อบบี้
3. ไดเร็กเพลย์ไคลเอ็นท์ล็อบบี้ออบเจ็กต์
4. ไดเร็กเพลย์ล็อบบี้แอปพลิเคชั่นออบเจ็กต์

ออบเจ็กต์สองตัวของไดเร็กเพลย์ทำหน้าที่คล้ายกับเป็นตัวเชื่อมต่อระหว่างโปรแกรม กับไคลเอ็นท์ล็อบบี้ ซึ่งมีการติดต่อระหว่างกันผ่านอินเตอร์เฟซที่กำหนดไว้โดยเฉพาะ



ภาพที่ 4-5 แสดงสถาปัตยกรรมของล็อบบี้และลักษณะการติดต่อกันของแต่ละส่วนประกอบ

4.5.2 เซิร์ฟเวอร์ล็อบบี้

เป็นแอปพลิเคชั่นที่จุดมุ่งหมายเพื่อให้ผู้เล่นแต่ละคนสามารถพบกัน และจัดตั้งเกมสำหรับเล่นได้ โดยส่วนมากแล้วเซิร์ฟเวอร์ล็อบบี้จะถูกติดตั้งไว้ที่รีโมทเซิร์ฟเวอร์ และเข้าถึงได้ผ่านทางอินเตอร์เน็ต สำหรับการจัดการเกี่ยวกับเกมแบบหลายผู้เล่นนี้ เซิร์ฟเวอร์ล็อบบี้ต้องจัดการเกี่ยวกับงานต่างๆเหล่านี้ เช่น

1. จัดการเกี่ยวกับแอดเดรสของระบบเครือข่ายของเซชชันและผู้เล่น
2. ออกคำสั่งเรียกเซชชันโดยการเรียกเกมแอปพลิเคชั่นบนเครื่องของผู้เล่น
3. เพิ่มเติมผู้เล่นเข้าไปในเซชชันที่สร้างขึ้นมาแล้วได้
4. ทำการติดต่อเครื่องต่างๆในเซชชันให้เป็นแอดเดรสของระบบเครือข่ายที่ถูกต้อง
5. จัดการกับข้อมูลต่างๆของผู้เล่นเช่นมีผู้เล่นออกจากเกม เป็นต้น

ข้อมูลของเซิร์ฟเวอร์ล็อบบี้ขึ้นอยู่กับชนิดของบริการที่ผู้ขายต้องการให้ไดเร็กเพลย์ไม่ได้กำหนดว่าจะต้องโปรแกรมส่วน เซิร์ฟเวอร์ล็อบบี้อย่างไร และต้องควรจะติดต่อกับคอมพิวเตอร์อื่นๆอย่างไร อย่างไรก็ตาม ผู้ขายส่วนล็อบบี้ต้องมีการโปรแกรมให้ติดต่อกับไดเร็กเพลย์ได้

4.5.3 ไคลเอ็นท์ล็อบบี้

ไคลเอ็นท์ล็อบบี้เป็นแอปพลิเคชั่นที่ถูกพัฒนาโดยผู้ขายเซิร์ฟเวอร์ล็อบบี้และทำการติดตั้งไว้ที่เครื่องของผู้เล่นแต่ละเครื่องโดยจะทำหน้าที่จัดการดูแลการติดต่อสื่อสารระหว่างผู้เล่นกับเกม

แอปพลิเคชั่นและเซิร์ฟเวอร์ล็อบบี้ วิธีการทั่วๆไปที่จะสามารถทำการติดตั้งไคลเอ็นท์ล็อบบี้นี้ลงไปในเครื่องของผู้เล่นได้จะใช้วิธีการให้ Download ได้จากเว็บของผู้ขายเซิร์ฟเวอร์ล็อบบี้

โปรแกรมไคลเอ็นท์ล็อบบี้ไม่จำเป็นต้องมีการเชื่อมต่อไปยังเครื่องคอมพิวเตอร์ปลายทาง ในบางลักษณะผู้เล่นจะเรียกโปรแกรมไคลเอ็นท์ล็อบบี้โดยตรง ไคลเอ็นท์ล็อบบี้ก็จะแสดงเกม และ เซชชัน ที่มีอยู่ทั้งหมดซึ่งบางครั้งอาจจะเป็นผู้เล่นที่ต่อกันอยู่ใน LAN เมื่อผู้เล่นได้เลือกเกมและเซชชันแล้ว ไคลเอ็นท์ล็อบบี้จึงจะทำการเริ่มต้นเกม

4.6 ไดเร็กเพลย์โปรโตคอล

ไดเร็กเพลย์โปรโตคอลเป็นโปรโตคอลที่จัดการกับข่าวสารในระดับของการสื่อสาร (Transport Layer) เพื่อจัดการกับข่าวสารที่ทำการส่งไปมาระหว่างเครือข่ายให้ระบบสามารถทำงานได้อย่างเรียบร้อยและมีประสิทธิภาพ โดยไดเร็กเพลย์โปรโตคอลสามารถจัดการกับข่าวสารต่างๆได้ดังนี้

- การส่งข่าวสารแบบเชื่อถือได้(Reliable) และเชื่อถือไม่ได้ (Unreliable)
- การสื่อสารแบบต่อเนื่อง (Sequential) และแบบไม่ต่อเนื่อง (non-Sequential)
- การแบ่งข่าวสารออก (Fragmentation) และการประกอบเป็นแบบเดิม (Reassemble)
- การควบคุมความคับคั่งของข้อมูล (Congestion Control)
- ระดับความสำคัญของข้อมูล (Send Prioritization)
- ระยะเวลาในการส่งข้อมูล (Message Timeouts)

4.6.1 การจัดการกับข่าวสารเบื้องต้น(Basic Message Handling)

คำว่าข่าวสารในเอกสารชุดนี้จะหมายถึงบล็อกของข้อมูลที่ต้องการส่งไปที่เครื่องคอมพิวเตอร์เครื่องอื่นๆ โดยที่โปรโตคอลของระบบเครือข่ายจะทำการสร้าง Packets โดยทำการเพิ่มข้อมูลที่ต้องการทำการส่ง(ข่าวสาร)เข้าไปยังบล็อกของข้อมูลที่มีการเก็บข้อมูลบางอย่างไว้แล้ว(ข้อมูลที่เป็นHeader) เช่นที่อยู่ของเครื่องที่ต้องการทำการติดต่อด้วย Packets นี้จะหมายถึงหน่วยของข้อมูลที่ต้องการทำการส่งผ่านระบบเครือข่าย เมื่อเครื่องปลายทางได้รับ Packet แล้ว โปรโตคอลของระบบเครือข่ายของเครื่องปลายทางจะทำการกำจัดข้อมูลส่วนที่เป็น Header ทิ้งและนำเฉพาะส่วนที่เป็นข้อมูลที่ต้องการใช้จริงๆไปใช้งานโโยส่งให้แอปพลิเคชั่นต่อไป

คำว่าข่าวสารและ Packets ไม่สามารถใช้แทนกันได้ คำว่าข่าวสารจะหมายถึงหน่วยของข้อมูลที่ทำการรับส่งกันโดยไดเร็กเพลย์ ส่วนคำว่า Packet จะหมายถึงหน่วยของข้อมูลที่มีการจัดการโดยระบบเครือข่าย สาเหตุที่ต้องทำการแบ่งแยกระหว่างคำว่าข่าวสารและ Packets นั้นเนื่องมาจากว่าระบบเครือข่ายจะทำการจำกัดขนาดของ Packet ที่ระบบเครือข่ายนั้นๆรองรับ ซึ่งขนาดดังกล่าวนี้เรียกว่า Maximum Transmission Unit(MTR) นั่นก็หมายความว่าถ้าข่าวสารมีขนาดเล็ก ก็จะสามารถทำการส่งโดยใช้ Packets เพียง Packets เดียวได้ โดยในกรณีนี้คำว่าข่าวสารและ Packets จะมีความหมายเหมือนกัน แต่ถ้าหากว่าข่าวสารที่ต้องการส่งนั้นมีขนาดใหญ่ก็อาจจะต้องทำการแบ่งข่าวสารออกเป็นสอง Packets และเครื่องปลายทางจะทำการประกอบเป็นข่าวสารเดียวกันให้เอง ซึ่งไดเร็กเพลย์โปรโตคอลจะทำการแยกและประกอบ

ข่าวสารให้โดยอัตโนมัติ ไดเร็กเพลย์สามารถส่งข่าวสารที่ขนาดใดๆ ก็ได้ แต่ถ้าขนาดใหญ่มากจะทำให้การแตกกลุ่มย่อยมีมากกลุ่ม ซึ่งอาจจะมีการผิดพลาดได้มากขึ้น ดังนั้นถ้าเป็นไปได้ไม่ควรส่งข่าวสารที่ขนาดใหญ่มากๆ

4.6.2 ชนิดของข่าวสาร(Message Categories)

ไดเร็กเพลย์โปรโตคอลได้ถูกออกแบบมาเพื่อจัดการกับข่าวสารของระบบเครือข่ายสองชนิดด้วยกันนั่นก็คือ

4.6.2.1 การส่งข่าวสารแบบเชื่อถือได้(Reliable)และแบบเชื่อถือไม่ได้ (Unreliable)

ข่าวสารที่ส่งกันผ่านระบบเครือข่ายนั้นอาจจะเกิดเสียหายหรือสูญหายในขั้นตอนของการส่งได้ แต่สำหรับการส่งข่าวสารที่เป็นแบบเชื่อถือได้นั้น จะรับรองได้ว่าทุกๆข่าวสารที่ทำการส่งนั้นจะถึงเครื่องของผู้รับแน่นอน การเลือกใช้งานข่าวสารแบบนี้ควรใช้เมื่อข่าวสารที่ต้องการส่งนั้นเป็นข่าวสารสำคัญที่ไม่อาจยอมให้สูญหายไปในเครือข่ายได้ หลักการของการส่งข่าวสารแบบนี้ก็คือ เมื่อผู้ส่งทำการส่งข่าวสารไปแล้วนั้นผู้ส่งจะรอข้อความตอบรับจากเครื่องผู้รับว่าได้รับข่าวสารแล้วภายในช่วงเวลาที่กำหนด หากว่าภายในกำหนดเวลาที่กำหนดไว้แล้วยังไม่มีข้อความตอบรับจากผู้รับ ผู้ส่งจะทำการส่งข่าวสารนั้นซ้ำไปยังผู้รับอีกครั้งหนึ่ง ขั้นตอนดังกล่าวนี้จะถูกทำซ้ำไปเรื่อยๆจนกว่าผู้ส่งจะได้รับข้อความยืนยันการได้รับข่าวสารจากผู้รับ

โปรโตคอลของไดเร็กเพลย์จะกำหนดจำนวนครั้งของการส่งข่าวสารซ้ำเอาไว้ ถ้าหากว่าผู้ส่งทำการส่งข่าวสารซ้ำครบจำนวนที่กำหนดไว้แล้ว ไดเร็กเพลย์จะถือว่าการติดต่อกับเครื่องผู้รับนั้นๆได้หยุดการติดต่อลงแล้ว และจะทำการยกเลิกการติดต่อกับเครื่องดังกล่าว

ในส่วนของข่าวสารที่เชื่อถือไม่ได้นั้น ถือว่าเป็นรูปแบบอย่างง่ายที่สุดของการติดต่อสื่อสารและมีความเร็วของการใช้งานที่สูงกว่าแบบข่าวสารที่เชื่อถือได้ เนื่องจากว่าเมื่อผู้ส่งทำการส่งข้อความไปแล้ว จะไม่ต้องรอข้อความการยืนยันการได้รับข้อความจากผู้รับ ดังนั้นหากผู้รับไม่ได้รับข่าวสารดังกล่าว ผู้ส่งก็จะไม่ทำการส่งข่าวสารนั้นซ้ำไปให้ การเลือกใช้ข่าวสารแบบนี้นั้นควรเลือกใช้งานเมื่อต้องการความเร็วของการใช้งาน และข่าวสารที่ทำการส่งนั้นไม่มีความสำคัญมากนัก ตัวอย่างเช่นแอปพลิเคชั่นประเภท Streaming Media ซึ่งแอปพลิเคชั่นประเภทนี้นั้นไม่สามารถรอการยืนยันการรับข่าวสารได้ เนื่องจากต้องการการทำงานที่รวดเร็ว ส่วนข่าวสารที่อาจจะสูญหายไปบ้างนั้นจะมีผลกระทบแค่เพียงคุณภาพเท่านั้นซึ่งถือว่ายอมรับได้

4.6.2.2 การสื่อสารแบบต่อเนื่อง (Sequential) และแบบไม่ต่อเนื่อง (non-Sequential)

เมื่อข่าวสารถูกส่งจากเครื่องเครื่องหนึ่งไปยังเครื่องปลายทางนั้น ลำดับของการส่งและการรับอาจจะไม่เหมือนกัน ตัวอย่างเช่นผู้ส่งทำการส่งข่าวสารบางอย่างไปยังเครื่องปลายทาง ระหว่างการส่งข้อมูลนั้นข่าวสารดังกล่าวเกิดการสูญหาย ผู้ส่งจะทำการส่งข่าวสารดังกล่าวซ้ำไปยังปลายทาง ดังนั้นข่าวสารที่ทำการส่งไปใหม่นี้อาจจะไปถึงปลายทางหลังจากข่าวสารบางอย่างที่ถูกส่งออกมาจากเครื่องผู้ส่งก่อนข่าวสาร

ที่ถูกทำการส่งใหม่นี้ได้ ข่าวสารแบบต่อเนื่องนี้จะทำการเพิ่มข้อมูลเกี่ยวกับลำดับการส่งบางอย่างเข้าไปยังข่าวสารที่ต้องการส่ง เพื่อใช้ตรวจสอบลำดับของการรับข้อมูลที่เครื่องปลายทาง ข่าวสารแบบต่อเนื่องนี้จะถูกใช้งานเมื่อแอปพลิเคชั่นปลายทางต้องการรับข่าวสารตามลำดับของข้อมูลที่ส่งมาจากผู้ส่ง ส่วนข่าวสารที่ลำดับของการส่งไม่ตรงกับหมายเลขที่ต้องการนั้นก็จะถูกเก็บเอาไว้ใน Buffer จนกระทั่งข่าวสารที่มีลำดับการรับถูกต้องจะมาถึงเครื่องปลายทาง

4.6.3 การควบคุมความคับคั่งของข้อมูล (Congestion Control)

ในอุดมคติ เกมและแอปพลิเคชั่นสามารถส่งข่าวสารได้มากและบ่อยได้เท่าที่ต้องการ โดยที่ข่าวสารจะถึงเป้าหมายทันที และมีการทำงานในทันทีเช่นกัน ถ้าทุกๆเครื่องในเกมมีความสามารถในการประมวลผลสูงมากและติดต่อด้วยสายเชื่อมเครือข่ายที่ Bandwidth สูงมากๆ ก็อาจจะเป็นไปได้ตามอุดมคติดังกล่าว นั่นก็คือเราเราสามารถส่งข่าวสารได้มากที่สุดเท่าที่ต้องการ แต่อย่างไรก็ตามส่วนประกอบอื่นๆก็อาจจะทำให้การสื่อสารช้าลง และมีความคับคั่งของข้อมูลในเครือข่ายได้ ดังนี้

- Network Latency: ข่าวสารอาจจะต้องใช้เวลาในการวิ่งหาทางจากผู้ส่งไปยังผู้รับ โดยเฉพาะอย่างยิ่งบนอินเตอร์เน็ต ซึ่งอาจจะมีความล่าช้าของการตอบรับจากปลายทางหรือการส่ง Packet ซ้ำในกรณีที่เกิดการสูญหายของข้อมูล หรือการประกอบ Packet ที่มาไม่ตรงตามลำดับ
- Network Bandwidth: จะเป็นส่วนประกอบที่ควบคุมอัตราการรับส่งข่าวสาร การเชื่อมต่อของเครือข่ายจะมีหลายแบบ บางครั้งเครือข่ายที่มี Bandwidth สูงบางเครือข่ายก็อาจจะมีความสามารถในการสื่อสารช้าลงได้เมื่อระดับของจำนวนข้อมูลที่ส่งมีมากขึ้น หรือถ้าหากว่ามีผู้เล่นบางคนที่มี Bandwidth ต่ำ ความสามารถในการสื่อสารก็จะช้าลงไปด้วย
- ความเร็วในการประมวลผล: แม้ว่า Bandwidth ของเครือข่ายจะสูง เครื่องปลายทางก็ยังจำเป็นต้องใช้เวลาในการประมวลผลอยู่ ดังนั้นประสิทธิภาพของเครื่องก็จะส่วนประกอบหนึ่งที่มีผลต่อความเร็วในการสื่อสาร

4.6.4 การจำกัดข่าวสาร (Message Throttling)

ถ้าไม่มีการควบคุมอัตราการส่งข่าวสารที่ทำการส่งไปยังปลายทาง ผู้รับอาจจะรับข่าวสารไม่ทันได้ เพื่อป้องกันเหตุการณ์นี้ไดเร็กเพลย์จึงมีการจำกัดอัตราในการส่งข่าวสารไว้ อัตราในการส่งถูกควบคุมโดยอัตราซึ่งฝ่ายรับสามารถรับข่าวสารได้ทัน

การควบคุมอัตราการส่งนี้สามารถทำได้โดยใช้ Sliding Window วิธี Sliding Window จะใช้คิวในการจำกัดจำนวนของช่องข้อมูลซึ่งจะถูกส่งไป ข่าวสารที่ส่งออกไปทั้งหมดจะถูกใส่ลงในคิวโดยไม่สนใจประเภทของข่าวสาร เมื่อข่าวสารที่ต้องการจะส่งออกไปล้นคิว ก็จะไม่มีการรับข่าวสารที่จะถูกส่งออกไปอีกจนกว่าจะมีข่าวสารหนึ่งในคิวไปถึงผู้รับ

4.6.5 การตรวจสอบสถานะการติดต่อ

ถ้าไม่มีงานบนการเชื่อมต่อ ไดเร็กเพลย์จะทำการทดสอบสถานะการเชื่อมต่อเป็นระยะๆ โดยทำการส่ง Packet เปล่าๆ ไป ถ้าไม่มีการตอบรับมาตามเวลาที่กำหนดไว้ก็จะถือว่าการเชื่อมต่อนั้นได้หลุดไปแล้ว

4.6.6 การให้ความสำคัญระหว่างข่าวสาร

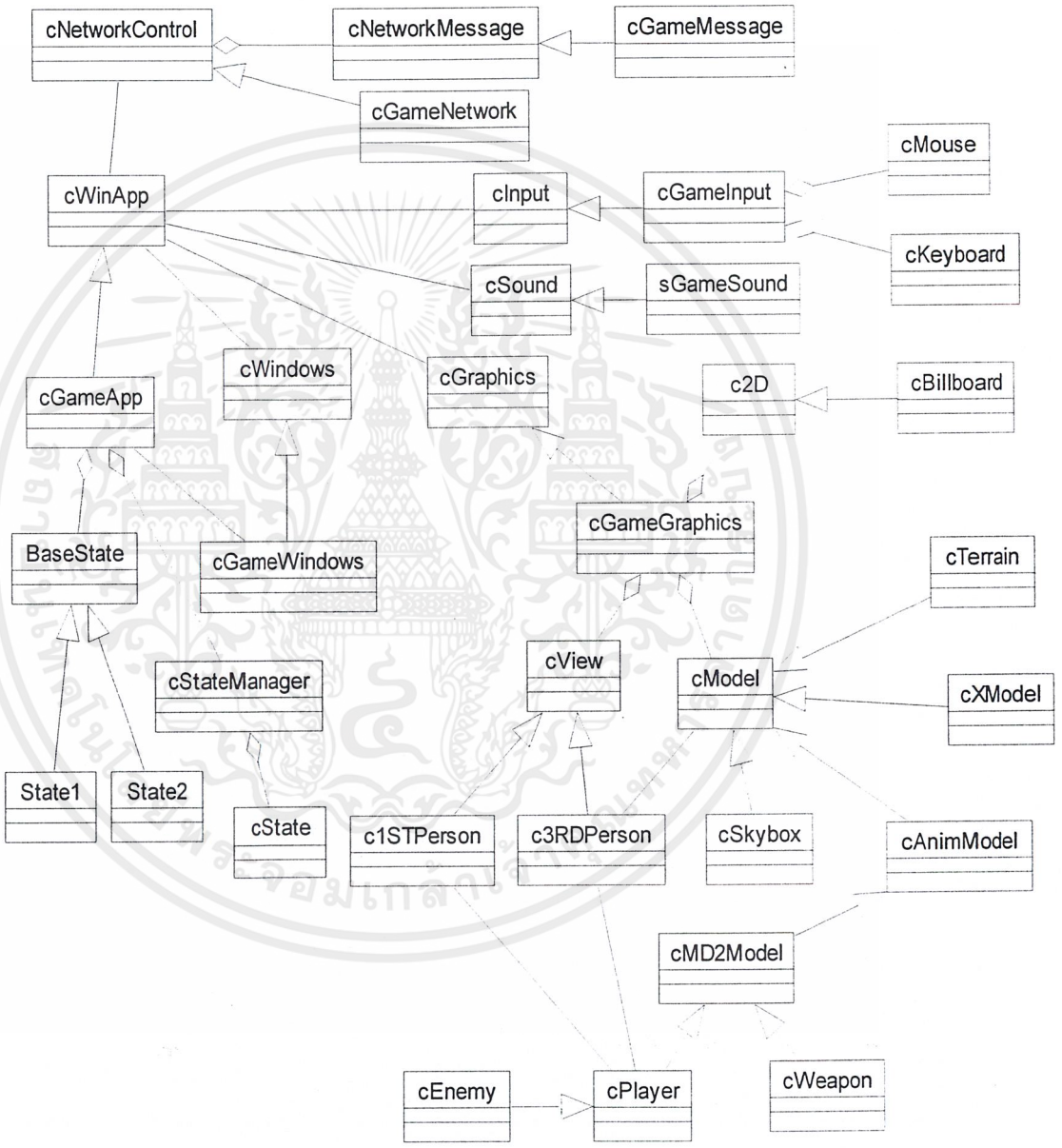
ข่าวสารในเกมนั้นมีการเปลี่ยนแปลงเป็นอย่างมาก บางข่าวสารจำเป็นต้องได้รับการส่งออกไปก่อน ในขณะที่บางข่าวสารสามารถที่จะรอได้ การเกิดความคับคั่งของข้อมูลก็อาจเกิดขึ้นได้เมื่อโปรแกรมสามารถสร้างข่าวสารได้เร็วกว่าความสามารถในการส่งข่าวสารออกไป ข่าวสารที่ยังไม่ได้รับส่งออกไปไปก็จะเก็บไว้ในคิวจนกว่าช่องการส่งจะว่าง ถ้าข่าวสารที่ยังไม่ได้ส่งมีค่าเท่ากัน ข่าวสารที่มีความสำคัญน้อยอาจจะถูกส่งออกไปก่อนข่าวสารที่มีความสำคัญมากก็ได้

โปรโตคอลของไดเร็กเพลย์ทำการแก้ปัญหาดังกล่าวนี้ด้วยการมีคิวสามระดับ ได้แก่ คิวที่มีความสำคัญต่ำ, คิวที่มีความสำคัญปานกลาง และคิวที่มีความสำคัญมาก เมื่อช่องทางการสื่อสารว่าง ไดเร็กเพลย์โปรโตคอลจะใช้หลักการดังนี้เลือกข่าวสารที่จะทำการส่งไปก่อน

1. ส่งข่าวสารที่เก่าที่สุดที่อยู่ในคิวที่มีความสำคัญมากออกก่อน
2. ถ้าไม่มีข่าวสารในคิวที่มีความสำคัญมาก ก็ให้ส่งข่าวสารที่เก่าที่สุดที่อยู่ในคิวที่มีความสำคัญปานกลาง
3. ถ้าไม่มีข่าวสารในคิวที่มีความสำคัญปานกลาง ก็ให้ส่งข่าวสารที่เก่าที่สุดที่อยู่ในคิวที่มีความสำคัญต่ำที่สุดออกไป

บทที่ 5
การออกแบบและหลักการทำงานของเกม

5.1 Class Diagram ของเกม



ภาพที่ 5-1 : ภาพแสดง Class Diagram ของโครงงาน

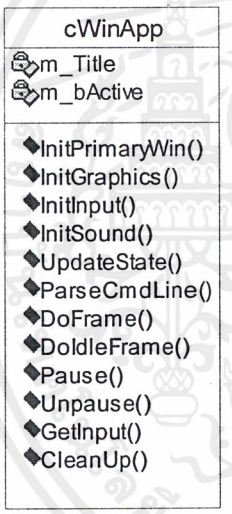
5.2 สมาชิกในคลาส และคำอธิบายคลาส

ในหัวข้อนี้จะขออธิบายคลาสต่างๆที่มีความสำคัญอย่างคร่าวๆถึง Attribute, Operation รวมทั้งหน้าที่ในแต่ละคลาส ดังนี้

1. คลาส cWinApp

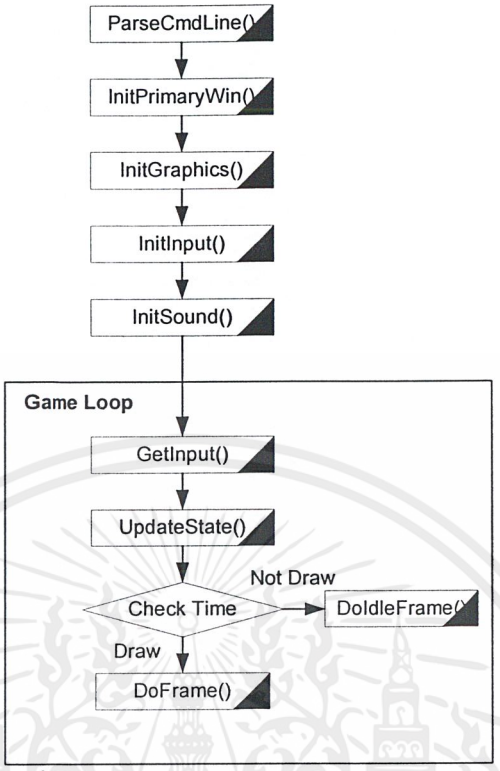
เป็นคลาสหลักซึ่งเป็น Abstract Class ทำหน้าที่ควบคุมคลาสอื่นๆในเกม ดังนั้นหน้าที่ของคลาสนี้จะทำหารควบคุมสิ่งต่างๆภายในเกมเช่น การแสดงผล, การรับอินพุทจากผู้เล่น, การให้เสียงประกอบ, รวมทั้งการจัดการการส่งข้อมูลผ่านเครือข่ายด้วย

คลาสที่สืบทอดจากคลาสนี้ไปคือ cGameApp จะเป็นคลาสเพียงคลาสเดียวที่ถูกเรียกใช้งานจาก WinMain และเมื่อคลาสนี้ได้รับการเรียกให้ทำงานแล้วจะทำการไปเรียกคลาสอื่นๆให้ทำงานต่ออีกครั้งหนึ่ง โดยมีลำดับขั้นตอนการทำงานดังรูปที่ 5-2 Attribute และ Operation ที่สำคัญของคลาสนี้ คือ



Class cWinApp	
Attribute	**หน้าที่**
m_Title	เก็บชื่อของแอปพลิเคชั่น
m_bActive	เป็น Flag ที่บ่งบอกว่าแอปพลิเคชั่น Active อยู่หรือไม่
Operation	
InitPrimaryWin()	ทำการกำหนดค่าเริ่มต้นต่างๆ และสร้างวินโดวส์หลักขึ้นมา
InitGraphics()	ทำการกำหนดค่าเริ่มต้นส่วนของการแสดงผล
InitInput()	ทำการกำหนดค่าเริ่มต้นส่วนการรับอินพุทจากผู้เล่น
InitSound()	ทำการกำหนดค่าเริ่มต้นต่างๆที่ใช้ในการให้เสียง
UpdateState()	จัดการการเปลี่ยนแปลงสเตทหลักของเกม
ParseCmdLine()	จัดการเกี่ยวกับการรับ Command Line จากภายนอก
DoFrame()	ทำงานในแต่ละลูปของเกม เพื่อทำการเรนเดอร์และแสดงผลออกทางหน้าจอ
DoIdleFrame()	ใช้จัดการส่วนต่างๆเมื่อเกมยังไม่มีการเรนเดอร์
Pause()	หยุดการทำงานของโปรแกรม
Unpause()	เริ่มการทำงานของโปรแกรม
GetInput()	รับอินพุทจากภายนอก
CleanUp()	จบการทำงานของโปรแกรม

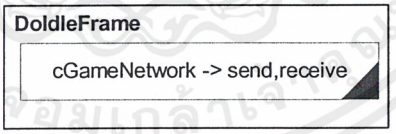
ตาราง 5-1 : ตารางแสดง Attribute และ Operation ของคลาส cWinApp



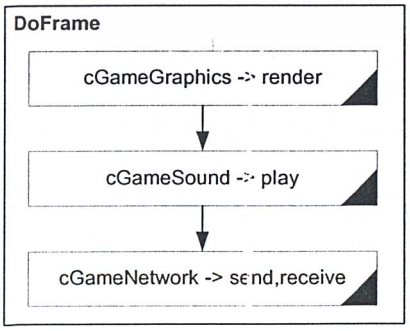
ภาพที่ 5-2 : แสดงลำดับการทำงานของคลาส cWinApp

2. คลาส cGameApp

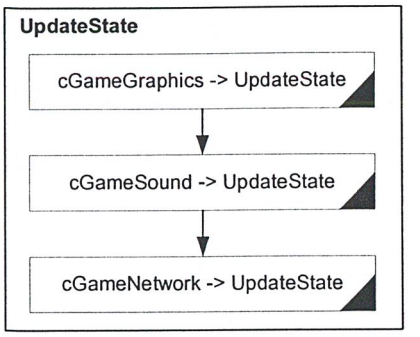
เป็นคลาสที่สืบทอดมาจากคลาส cWinApp คลาสนี้จะทำการเรียกใช้งานคลาสต่างๆเช่นคลาส cGameSound, cGameInput, cGameGraphics และ cGameNetwork โดยมีการทำงานของส่วนต่างๆที่ขยายมาจากคลาส cWinApp ดังรูปที่ 5-3,5 -4 และ 5-5 ตามลำดับ



ภาพที่ 5-3 : แสดงการทำงานในส่วนของ DoIdleFrame ของคลาส cGameApp



ภาพที่ 5-4 : แสดงการทำงานในส่วนของ DoFrame ของคลาส cGameApp



ภาพที่ 5-5: แสดงการทำงานในส่วนของ UpdateState ของคลาส cGameApp

3. คลาส cWindows

เป็น Abstract Class ที่ทำหน้าที่จัดการเกี่ยวกับ Windows ทั้งหมดไม่ว่าจะเป็นการลงทะเบียน Window ใหม่ลงไปยังระบบ หรือสร้าง Window ขึ้นมา โดยมี Attributes และ Operations หลักๆดังตารางที่ 5-2

Class cWindows		
cWindows m_width c_height m_hwnd m_bActive m_name wndProc() RegisterClass() getHWnd() InitInstance()	**Attribute**	**หน้าที่**
	m_width	กำหนดความกว้างของ Windows
	m_height	กำหนดความสูงของ Windows
	m_hwnd	เป็น Handle ของ Window
	m_bActive	เป็น Flag ที่บ่งบอกว่า Window Active อยู่หรือไม่
	m_name	ตัวแปรเก็บชื่อของ Window
	Operation	**หน้าที่**
	WndProc	เป็นเมธอดที่ใช้เชื่อมต่อกับ WindowProc ใน Win32 API เพื่อใช้ในการควบคุม Window
	RegisterClass	ใช้เพื่อลงทะเบียน Window ลงไปยัง System
	GetHWnd	เรียกใช้งาน Handle ของ Window เพื่อนำไปใช้ควบคุมและติดต่อกับ Window
	InitInstance	เป็นเมธอดที่ใช้ในการสร้าง Window ขึ้นมา

ตาราง 5-2 : ตารางแสดง Attribute และ Operation ของคลาส cWindows

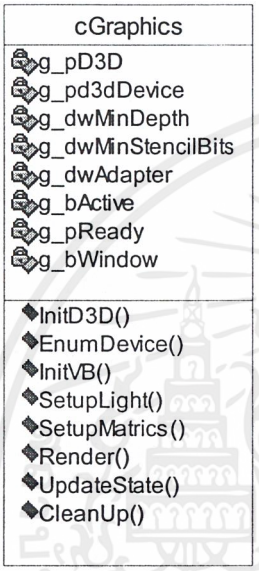
4. คลาส cGameWindows

เป็นคลาสที่จัดการกับ Window ของเกม และเป็นการที่ทำการสืบทอดมาจากคลาส cWindows โดยเพิ่มเติมส่วนของการ Implementation ให้กับฟังก์ชันที่เป็น Abstract Function

5. คลาส cGraphics

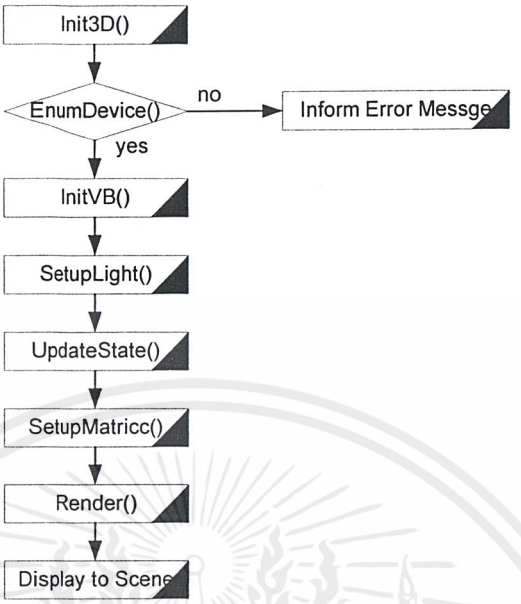
เป็น Abstract Class ที่ทำหน้าที่จัดการเกี่ยวกับกราฟฟิกทั้งหมด งส่วนของการจัดการแสง การควบคุมแมทริกซ์ที่ใช้ในการคำนวณการแสดง และการเรนเดอร์ ความสัมพันธ์ในการแสดงผลของแต่ละ Operation มีความสัมพันธ์ดังรูปที่ 5-6 มี Attributes และ Operations ที่สำคัญดังตารางด้านล่างนี้

Class cGraphics



Attribute	หน้าที่
g_pD3D	เป็นอินเตอร์เฟซของ Direct3D
g_pD3DDevice	เป็นอินเตอร์เฟซของอุปกรณ์ส่วนที่ใช้แสดงผลของ Direct3D
g_dwMinDepth	จำนวนบิตที่ใช้กำหนดสีในแต่ละจุด
g_dwMinStencil Bits	ใช้เก็บจำนวนบิต Stencil เพื่อใช้ในการจัดการ Effect การเขียนหรือไม่เขียนจุดต่างๆ
g_dwAdapter	เก็บข้อมูลของการ์ดแสดงผลที่ใช้
g_bActive	เป็น Flag ที่เก็บข้อมูลเพื่อบอกว่าพร้อมที่จะให้มีการแสดงผลหรือไม่
g_pReady	เป็น Flag ที่เก็บข้อมูลเพื่อบอกว่าพร้อมที่จะให้มีการแสดงผลหรือไม่
g_bWindow	เป็นตัวแปรที่เก็บค่าว่า Window ที่แสดงผลเป็น Window Mode หรือ Full Screen Mode
Operation	**หน้าที่**
Init3D()	เป็นเมธอดที่ใช้กำหนดค่าเริ่มต้นต่างๆให้กับ Direct3D
EnumDevice()	ใช้ในการตรวจสอบว่ามีอุปกรณ์ใดๆบ้างและรองรับกรแสดงผลที่ต้องการแสดงหรือไม่
InitVB()	ใช้ในการกำหนดค่าเริ่มต้นต่างๆให้กับ Vertex Buffer
SetupLight()	ใช้กำหนดค่าของแสงต่างๆที่ใช้ในฉาก
SetupMatrics()	ใช้กำหนด Transformation Matrix และคำนวณเกี่ยวกับตำแหน่งของส่วนต่างๆ
Render()	ใช้เพื่อคำนวณภาพที่ใช้แสดงผลที่หน้าจอ
UpdateState()	เป็นเมธอดที่ทำหน้าที่เปลี่ยนสถานะต่างๆของวัตถุ
CleanUp()	ใช้เพื่อทำการลบวัตถุในเกมทิ้ง

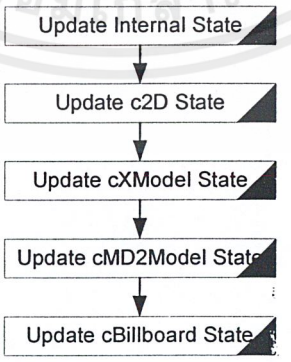
ตาราง 5-3 : ตารางแสดง Attribute และ Operation ของคลาส cGraphics



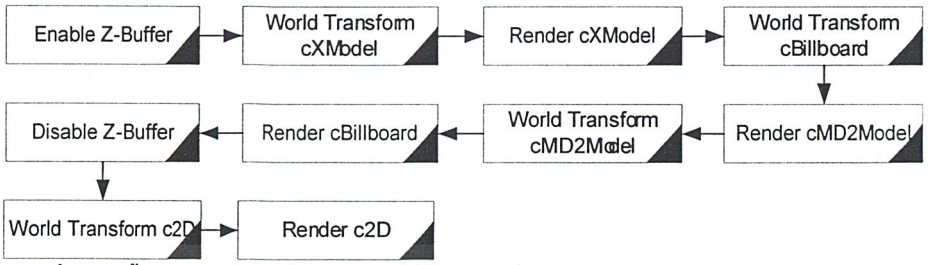
ภาพที่ 5-6 : แสดงความสัมพันธ์ของแต่ละ Operation ในการแสดงผลของคลาส cGraphics

6.คลาส cGameGraphics

เป็นคลาสที่สืบทอดมาจากคลาส cGraphics โดยเพิ่มเติมส่วนของการ Implementation ให้กับฟังก์ชันที่เป็น Abstract Function เพื่อนำไปใช้ในการแสดงผลภายในเกม ภายในคลาสนี้จะประกอบไปด้วยหลายคลาสเช่น คลาส c2D, cView, cMD2Model, cXModel, cTerrain, cBillboard เป็นต้น เพื่อใช้ในการแสดงผลเช่นการแสดงผลในส่วนของตัวละครในรูปแบบของไฟล์ตระกูล .X(DirectX File Format) หรือตัวละครในรูปแบบไฟล์ตระกูล MD2(Quake2 Model File Format) ในคลาสนี้จะมีการเปลี่ยนแปลงสเตทของตัวละครต่างๆรวมอยู่ด้วยซึ่งหลักการของการเปลี่ยนแปลงสเตทนั้นจะมีขั้นตอนดังรูปที่ 5-7 ส่วนรูปที่ 5-8 นั้นแสดงขั้นตอนการแสดงผลหลังจากที่ทำการเปลี่ยนแปลงสเตทแล้ว



ภาพที่ 5-7 : แสดงการเปลี่ยนแปลงสเตทของคลาส cGameGraphics



ภาพที่ 5-8 : ขั้นตอนการแสดงผลหลังจากที่ทำการเปลี่ยนแปลงสเตทของคลาส cGameGraphics

7. คลาส cModel

เป็น Abstract Class ที่เก็บข้อมูลเกี่ยวกับโมเดลต่างๆที่จะใช้ในการเรนเดอร์ โดยมี Operation ที่สำคัญดังตารางที่ 5-4

Class cModel		
cModel LoadFromFile() Render()	**Operation**	หน้าที่
	LoadFromFile()	ทำการโหลดโมเดลจากไฟล์ที่กำหนด
	Render()	ทำการเรนเดอร์โมเดลเป็นภาพเพื่อแสดงผล

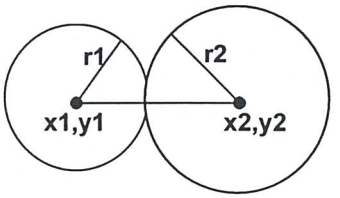
ตาราง 5-4 : ตารางแสดง Operation ของคลาส cModel

คลาส cModel นี้จะถูกสืบทอดมาเป็นคลาสอีกสองคลาสดังนี้

1. **คลาส cXModel:** เป็นคลาสที่ใช้ในการโหลดและเรนเดอร์โมเดลที่อยู่ในรูปแบบของ DirectX File Format (มีนามสกุลเป็น .x) ซึ่งเป็นมาตราฐานการเก็บโมเดลสามมิติของไมโครซอฟต์
2. **คลาส cMD2Model:** เป็นคลาสที่ใช้ในการโหลดและเรนเดอร์โมเดลที่อยู่ในรูปแบบของไฟล์ที่มีนามสกุล .md2 ซึ่งเป็นมาตราฐานการเก็บโมเดลสามมิติที่มีการเคลื่อนไหวของ INSoftware

8. คลาส cPlayer

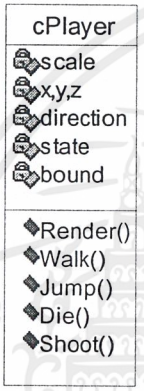
เป็นคลาสที่ใช้ในการจัดการตัวละครในเกมโดยมี Attribute และ Operation ดังตารางที่ 5-5 ซึ่งจะเห็นได้ว่ามี Attribute ที่ชื่อ Bound อยู่ด้วยซึ่ง Attribute นี้นำไปใช้ในการตรวจสอบการชนของวัตถุในเกม ซึ่งในโครงงานนี้จะใช้การตรวจสอบการชนแบบวงกลม(Bounding Sphere Checking) โดยมีหลักดังรูปที่ 5-9



ภาพที่ 5-9 : หลักการการตรวจสอบการชนของวัตถุแบบวงกลม

จากภาพจะเห็นว่าถ้า $\sqrt{(x1-x2)^2+(x1-y2)^2} < r1+r2$ นั่นหมายความว่าเกิดการชนของวัตถุขึ้น

Class *cPlayer*



Attribute	**หน้าที่**
Scale	เก็บอัตราส่วนของภาพที่แสดง กับขนาดจริงที่อยู่ในไฟล์
x,y,z	ค่าตำแหน่งของตัวละครในพิกัดเวิลด์จริง
Direction	ค่าทิศทางของตัวละครในพิกัดเวิลด์จริง
State	เก็บสเตทกริยาท่าทางของตัวละคร
Bound	ขนาดของขอบเขตที่ใช้ในการตรวจสอบการชน
Operation	**หน้าที่**
Render()	แสดงภาพตัวละครออกทางจอภาพ
Walk()	เลือกท่าทางขณะเดินของตัวละครเพื่อแสดงผล
Jump()	เลือกท่าทางขณะกระโดดของตัวละครเพื่อแสดงผล
Die()	เลือกท่าทางขณะตายของตัวละครเพื่อแสดงผล
Shoot()	เลือกท่าทางขณะยิงปืนของตัวละครเพื่อแสดงผล

ตาราง 5-5 : ตารางแสดง Attribute และ Operation ของคลาส cPlayer

9. คลาส cEnemy

เป็นคลาสที่สืบทอดมาจากคลาส cPlayer โดยมีการเพิ่มเติมในส่วนของรูปแบบของการโจมตีเข้าไป เพื่อให้ใช้เป็นศัตรูกับผู้เล่น

10. คลาส cSound

เป็นคลาสที่ใช้ในการจัดการกับเสียงในเกมโดยมี ImusicLoader8 เป็นอินเตอร์เฟซที่ใช้ในการควบคุม มี Operation ที่สำคัญดังตารางที่ 5-6

Class *cSound*		
cSound ◆Init() ◆Load() ◆Play() ◆Stop() ◆CleanUp()	**Operation**	**หน้าที่**
	Init()	ทำการกำหนดค่าเริ่มต้นต่างๆที่จำเป็นในการใช้งาน
	Load()	ใช้ในการโหลดไฟล์เสียง
	Play()	สั่งให้ทำการเล่นเสียงที่ต้องการ
	Stop()	สั่งหยุดเสียงที่กำลังทำงานอยู่
	CleanUp()	ลบไฟล์เสียงออกจากเกม

ตาราง 5-6 : ตารางแสดง Operation ของคลาส cSound

คลาส cSound นี้จะถูกสืบทอดมาเป็นอีกคลาสหนึ่งที่มีชื่อว่า cGameSound ซึ่งเป็นคลาสที่ใช้ในการควบคุมเสียงในเกมจริงๆ

11. คลาส cInput

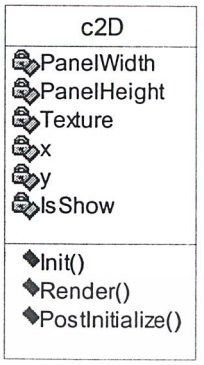
เป็น Abstract Class ที่ทำหน้าที่ในการรับอินพุทจากภายนอก คลาสนี้จะถูกสืบทอดมาเป็นคลาส cGameInput ซึ่งเป็นคลาสที่ Implement Operation ต่างๆของคลาส cInput และทำหน้าที่ควบคุมการรับอินพุทที่ใช้ในเกมจริงๆ ภายในคลาส cGameInput นี้จะมีคลาสย่อยอีกสองคลาสนั่นคือคลาส cMouse และ cKeyboard เพื่อรับค่าจาก Mouse และ Keyboard โดยที่ทั้งสองคลาสย่อยนี้มี Operation ที่สำคัญดังตาราง 5-7

Class *cMouse, cKeyboard*		
cMouse,cKeyboard ◆Init() ◆GetState() ◆CleanUp()	**Operation**	**หน้าที่**
	Init()	ทำการกำหนดค่าเริ่มต้นต่างๆที่จำเป็นในการใช้งาน
	GetState()	แปลงการรับอินพุตจากภายนอกเป็นสเตทที่เกมนำไปใช้งาน
	CleanUp()	ยกเลิกการใช้งาน

ตาราง 5-7 : ตารางแสดง Operation ของคลาส cMouse, cKeyboard

12. คลาส c2D

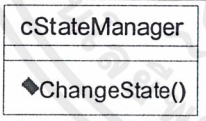
เป็นคลาสที่ใช้จัดการเกี่ยวกับภาพสองมิติ โดยใช้วิธีการนำโพลีกอนมาทำเป็นพื้นราบ(Plane) แล้วทำการวางพื้นผิวที่เป็นภาพสองมิติที่ต้องการแสดงผลทับลงไป มี Attribute และ Operation ดังตารางที่ 5-8 คลาส c2D นี้จะถูกสืบทอดไปเป็นคลาส cBillboard อีกทอดหนึ่ง

Class *c2D*		
	**Attribute**	**หน้าที่**
	PanelWidth	เก็บค่าความกว้างของวัตถุในโลกสองมิติ
	PanelHeight	เก็บค่าความสูงของวัตถุในโลกสองมิติ
	Texture	คือภาพสองมิติที่ต้องการแสดงผล
	x	เก็บค่า x ที่จุดซ้ายบนของ Panel
	y	เก็บค่า y ที่จุดซ้ายบนของ Panel
	IsShow	เป็น Flag ที่ใช้กำหนดว่าต้องการให้แสดงผลในเวลาใดๆ หรือไม่
	Operation	**หน้าที่**
	Init()	ทำการกำหนดค่าเริ่มต้นต่างๆที่จำเป็นในการใช้งาน
	Render()	ทำการเรนเดอร์ภาพเพื่อแสดงผล
	PostInitialize()	เอาไว้กำหนดการ Transform สำหรับภาพสองมิติ

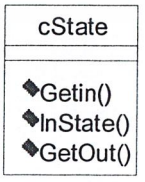
ตาราง 5-8 : ตารางแสดง Attribute และ Operation ของคลาส c2D

13. คลาส cStateManager

เป็นคลาสที่ทำหน้าที่ควบคุมฉากๆภายในเกม ภายในคลาสนี้จะมี(has) คลาสย่อยอีกหนึ่งคลาสชื่อว่า cState คลาส cStateManager มี Operation ดังตารางที่ 5-9 ส่วนคลาส cState แสดงในตารางที่ 5-10

Class cStateManager		
	**Operation**	**หน้าที่**
	ChangeState()	ทำการเรียกฟังก์ชันของคลาส cState เพื่อเปลี่ยนฉาก

ตาราง 5-9 : ตารางแสดง Operation ของคลาส cStateManager

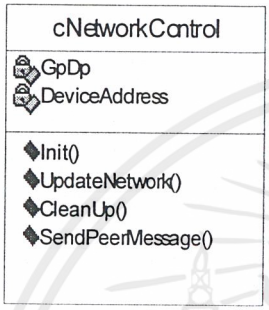
Class cState		
	**Operation**	**หน้าที่**
	GetIn()	ฟังก์ชันที่ทำการกำหนดค่าต่างๆก่อนทำการเปลี่ยนฉาก
	InState()	ฟังก์ชันที่กระทำในขณะที่ใช้งานฉากใดๆอยู่
	GetOut()	ฟังก์ชันที่ทำการ Free ต่างๆก่อนทำการเปลี่ยนฉาก

ตาราง 5-10 : ตารางแสดง Operation ของคลาส cState

14 คลาส cNetworkControl

คลาสนี้จะทำหน้าที่จัดการกับระบบเครือข่ายทั้งหมด เช่นการจัดการการเปลี่ยนแปลงในเรื่องของสเตทต่างๆ การส่งข้อมูลผ่านเครือข่าย รวมทั้งการกำหนดชนิดของข่าวสารที่ทำการส่งด้วยเช่นกัน คลาสนี้มี Attribute และ Operation ดังตารางที่ 5-11 ภายในคลาสนี้จะมี(has) คลาส cNetworkMessage ซึ่งเป็นคลาสที่กำหนดชนิดของ Packets บรรจุอยู่ภายใน

Class *cNetworkControl*



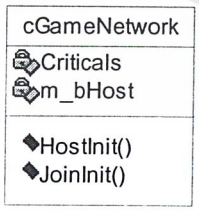
Attribute	**หน้าที่**
GpDp	เป็นอินเตอร์เฟซที่ชี้ไปยัง IdirectPlay8Peer
DeviceAddress	เป็นอินเตอร์เฟซที่ชี้ไปยัง IdirectPlay8Address
Operation	**หน้าที่**
Init()	ทำการกำหนดค่าเริ่มต้นต่างๆที่จำเป็นในการใช้งาน
UpdateNetwork()	
CleanUp()	ยกเลิกการใช้งานและฟรี Resource
SendPeerMessage()	ส่งข่าวสารผ่านระบบเครือข่ายไปยังผู้เล่นที่ Connect อยู่

ตาราง 5-11 : ตารางแสดง Attribute และ Operation ของคลาส cNetworkControl

15. คลาส cGameNetwork

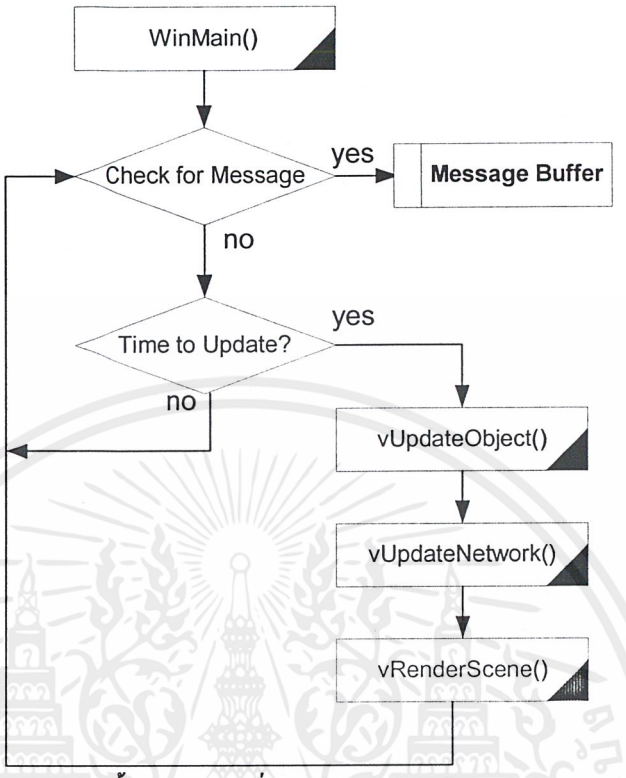
เป็นคลาสที่สืบทอดมาจากคลาส cNetworkControl เพื่อทำหน้าที่ควบคุมระบบเครือข่ายที่ใช้ในเกมจริงๆ มี Attribute และ Operation ที่เพิ่มมาจากคลาส cNetworkControl ดังตาราง 5-12 และมีขั้นตอนของการเปลี่ยนสเตทดังรูปที่ 5-10

Class *cGameNetwork*



Attribute	**หน้าที่**
Criticals	Flag ที่ใช้กำหนดส่วนที่เป็น Critical Section
M_bHost	Flag ที่ใช้ระบุและตรวจสอบว่าเครื่องใดทำหน้าที่เป็นโฮสต์
Operation	**หน้าที่**
HostInit()	ทำการกำหนดค่าเริ่มต้นต่างๆให้กับเครื่องที่เป็นโฮสต์
JoinInit()	ทำการกำหนดค่าเริ่มต้นต่างๆให้กับเครื่องที่เป็นผู้ขอ Connect กับโฮสต์

ตาราง 5-12 : ตารางแสดง Attribute และ Operation ของคลาส cGameNetwork



ภาพที่ 5-10 : แสดงขั้นตอนการเปลี่ยนแปลงสเตทภายในคลาส cGameNetwork

16 cNetworkMessage

เป็นคลาสที่บรรจุอยู่ในคลาส cNetworkControl และไม่มี Operation ทำหน้าที่เสมือน Struct ตามหน้าที่ 32 ในบทที่ 4 แต่โครงงานนี้ได้เปลี่ยน Struct ดังกล่าวมาเป็นคลาสเพื่อความง่ายของการสืบทอดต่อไป Attribute ของคลาสนี้แสดงในตารางที่ 5-13

Class cNetworkMessage		
cNetworkMessage Type Size	**Attribute**	**หน้าที่**
	Type	เก็บชนิดของ Packets ที่ต้องการส่งว่าเป็นชนิดใด
	Size	เก็บขนาดของ Packets ชนิดที่กำหนดไว้ว่ามีขนาดเท่าใด

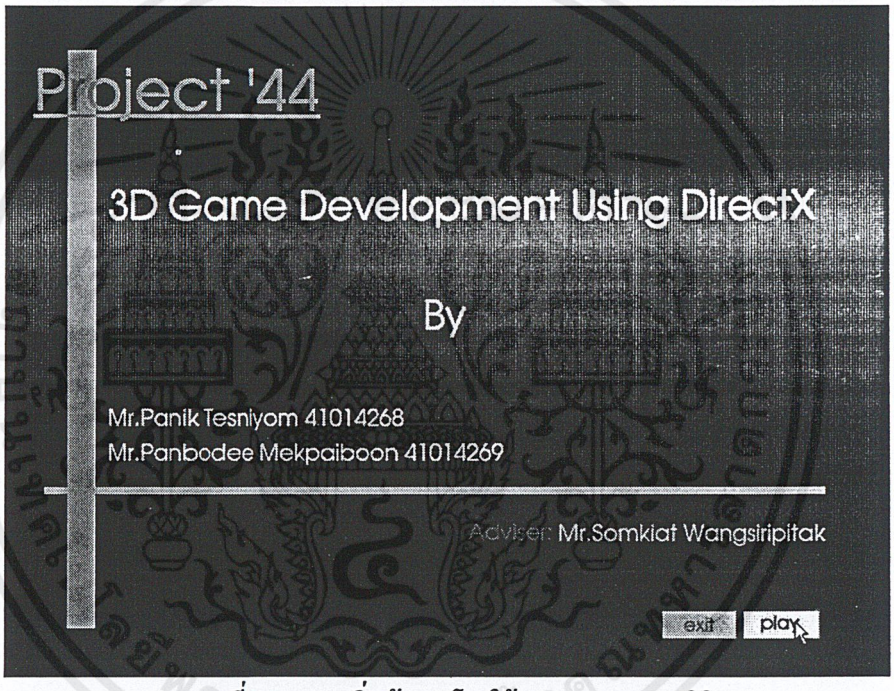
ตาราง 5-13 : ตารางแสดง Attribute ของคลาส cNetworkMessage

คลาสนี้จะถูกสืบทอดไปเป็นคลาส Message ต่างๆที่ใช้ในเกมเช่น cPacketEnemy, cPacketTypeEnemy เป็นต้น โดยคลาสต่างๆเหล่านี้จะทำการเพิ่ม Attribute ที่เป็นข้อมูลที่ต้องการใช้เข้าไป

บทที่ 6
ผลการทดลอง

6.1 การแสดงผลภาพสองมิติ

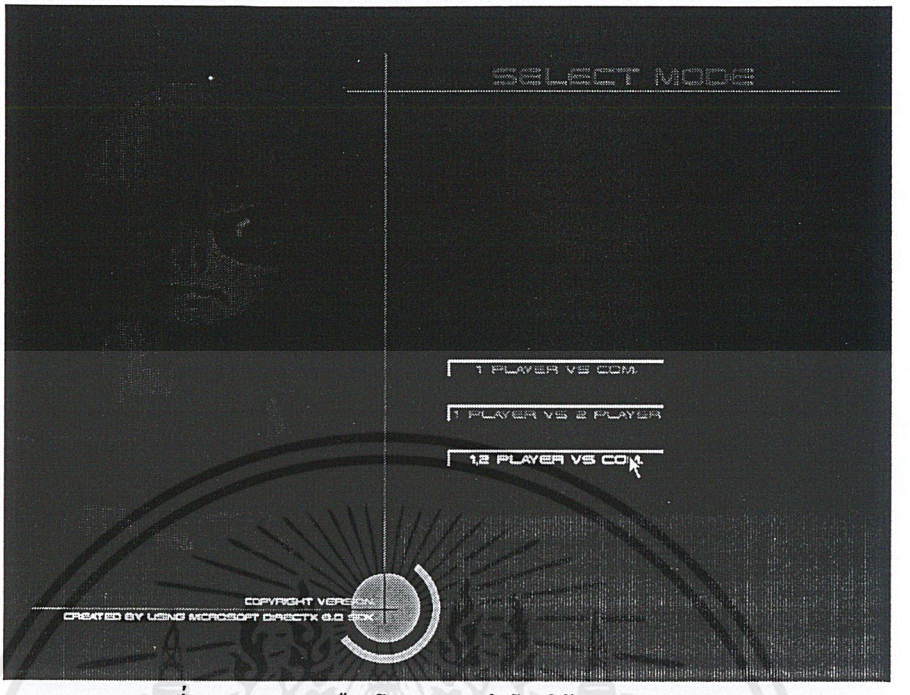
ส่วนของการแสดงผลสองมิตินั้น ได้ใช้เทคนิคการนำ Plane โพลีกอนสามมิติมาแปะภาพพื้นผิวเป็นภาพสองมิติที่ต้องการแสดงผล ซึ่งส่วนของการแสดงผลสองมิตินี้ ได้ใช้ในส่วนของเมนูการติดต่อผู้เล่น เช่น การเลือกโหมดการเล่น หรือการเลือกตัวละคร เป็นต้น ซึ่งจะเห็นได้จากรูปที่ 6-1, 6-2 และ 6-3



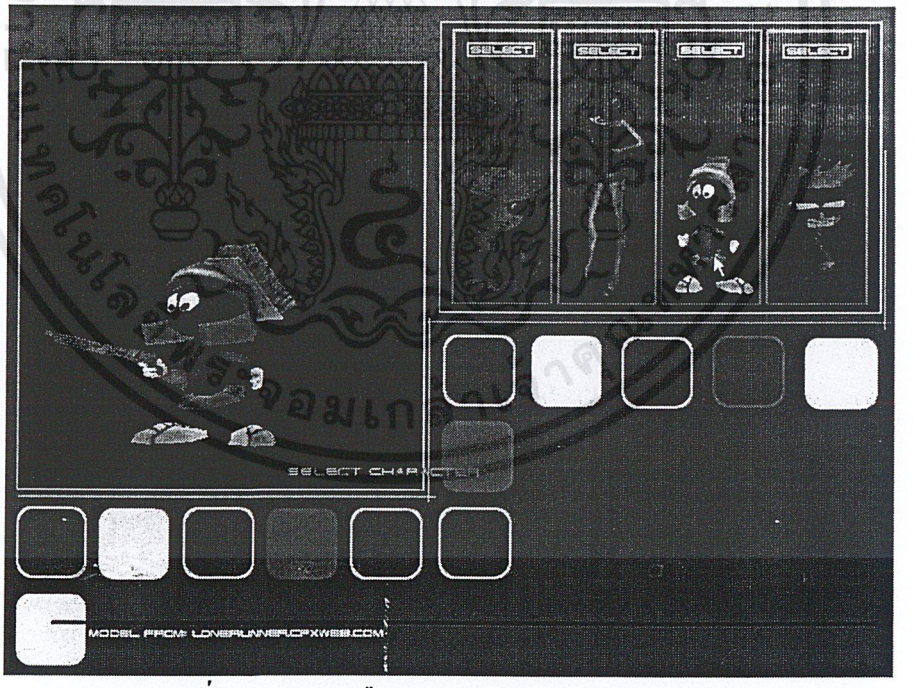
ภาพที่ 6-1 : ภาพเริ่มต้นเกมโดยใช้การแสดงผลสองมิติ

6.2 การแสดงผลภาพสามมิติ

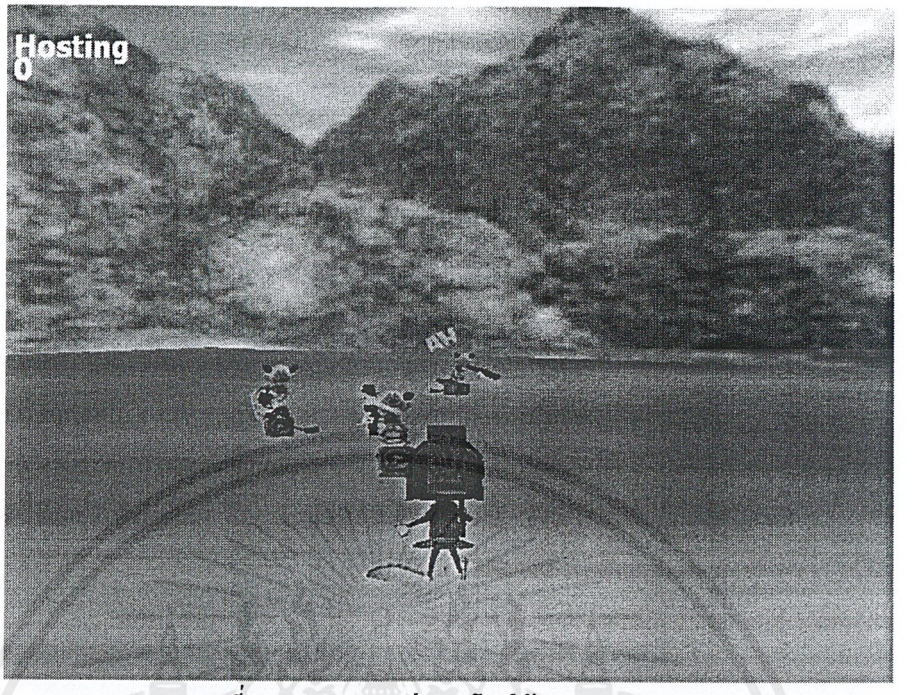
ส่วนของการแสดงผลสามมิตินั้นเป็นส่วนการแสดงผลหลักของเกมทั้งสามโหมด โดยภาพที่ได้นั้นใช้ไดเร็กเอ็กซ์กราฟฟิกเป็นเครื่องมือในการพัฒนา โดยวัตถุแต่ละวัตถุในเกมนั้นประกอบขึ้นมาจากโพลีกอนและแปะพื้นผิวที่ต้องการให้เป็นหน้าตา หรือเสื้อผ้าของตัวละคร หรือฉาก ดังภาพที่ 6-4 เป็นต้น



ภาพที่ 6-2 : ภาพการเลือกโหมดการเล่นโดยใช้การแสดงผลสองมิติ



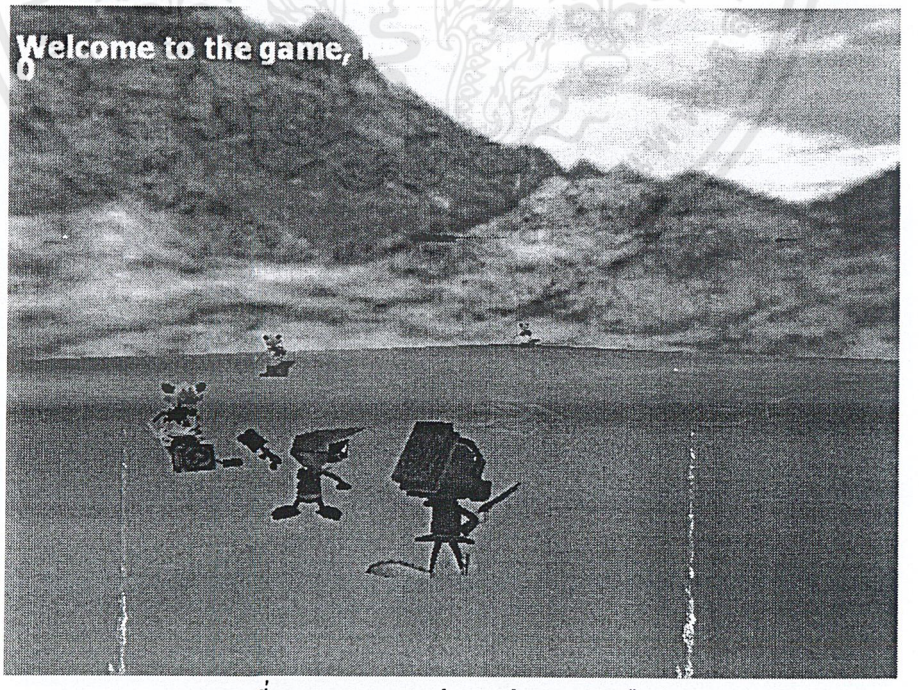
ภาพที่ 6-3 : ภาพการเลือกตัวละครโดยใช้การแสดงผลสองมิติ



ภาพที่ 6-4 : ภาพขณะเล่นเกมโดยใช้การแสดงผลสามมิติ

6.3 การเล่นผ่านระบบเครือข่าย

ในโหมดที่สองและสาม ซึ่งเป็นโหมดแบบการเล่นสองคนนั้น สามารถเล่นได้โดยใช้เครื่องคนละเครื่องผ่านระบบเครือข่าย ซึ่งผู้เล่นจะมองเห็นกันและการเปลี่ยนแปลงที่ผู้เล่นคนหนึ่งจะแสดงผลไปยังผู้เล่นอีกหนึ่งคนด้วย ดังรูปที่6-5



ภาพที่ 6-5 : ภาพขณะเล่นเกมผ่านระบบเครือข่าย

บทที่ 7
บทสรุปและวิจารณ์

7.1 ปัญหาที่พบในการพัฒนาโครงงาน

ภายในการพัฒนาโครงงานนี้ ผู้พัฒนาได้ประสบกับปัญหาต่างๆมากมาย โดยสามารถสรุปออกมาได้เป็นข้อๆ ดังนี้

1. การเรนเดอร์ภาพสามมิติ แบบ realtime นั้นมีการขึ้นกับสภาพแวดล้อมกับสิ่งต่างๆมาก เช่น จำนวนโพลีกอน หรือขนาดของวัตถุ ถ้าวัตถุต่างๆภายในฉากมีเหลี่ยมหรือจำนวนโพลีกอนที่มากเกินไป การแสดงผลอาจจะออกมาช้ามากจนถึงระดับที่ไม่สามารถยอมรับได้
2. การพัฒนาในด้านระบบเครือข่าย ประสบปัญหาการส่งข้อมูลเกิดความล่าช้าเนื่องจาก Packets ที่ส่งมีจำนวนมากและขนาดใหญ่

Vertices	Frame Rate(fps)	
	ใช้ Culling	ไม่ใช้ Culling
20	300	440
100	290	380
300	270	300
3000	130 (Vary ตาม Polygon)	25
10000	60 (Vary ตาม Polygon)	5

ตาราง 7-1: ตารางแสดงอัตราโพลีกอนกับ Frame Rate

7.2 การพัฒนาโครงงาน

ในการพัฒนาโครงงานต่อไป ควรพัฒนาให้สามารถเล่นผ่านระบบเครือข่ายได้มากกว่าสิบคนขึ้นไปโดยมีประสิทธิภาพที่สูงจนพอรับได้ และพัฒนาความเร็วในการแสดงผลหรือเฟรมเรทให้มีประสิทธิภาพมากขึ้น ความสวยงามต่างๆภายในเกม การเพิ่มฉากของการเล่นให้มีมากกว่านี้ รวมทั้งการพัฒนาส่วนติดต่อผู้ใช้งานต่างๆเช่น การกรอก IP แอดเดรสของผู้เล่นที่เป็นโฮสต์ อาจจะให้กรอกเป็นชื่อเครื่องแทนหมายเลข IP เป็นต้น

7.3 ข้อแนะนำ

ควรใช้เกมเอนจิ้นที่มีประสิทธิภาพในการพัฒนาเกมสามมิติเนื่องจากมีความง่าย สะดวก และรวดเร็วในการพัฒนามากกว่าการพัฒนาส่วนเอนจิ้นขึ้นมาเองทั้งหมด เอนจิ้นเกมในตลาดที่มีราคาถูกและน่าสนใจเช่น LithTech ของอเมริกาเป็นต้น

การพัฒนาเกมอาจจะใช้เครื่องมือในการพัฒนาที่ต่างไปเช่น OpenGL เนื่องจากการ์ดแสดงผลเกือบทั้งหมดต่างก็รองรับการใช้งาน OpenGL และ OpenGL นั้นก็มีความเร็วในการประมวลผลมากกว่าไดเร็กเอ็กซ์อีกด้วย แต่ข้อเสียของ OpenGL นั้นก็คือโครงสร้างไม่เป็นออบเจ็กต์ แต่เป็นแบบโครงสร้างนั่นเอง ดังนั้นควรจะมีการพัฒนาเกมขึ้นมาเป็นเกมเดียวกัน และใช้โมเดลการแสดงผลเป็นโมเดลเดียวกัน แต่ทำการพัฒนาโดยใช้เครื่องมือต่างกัน เพื่อเปรียบเทียบให้เห็นประสิทธิภาพ

ในส่วนของการเล่นผ่านระบบเครือข่ายนั้น ควรพัฒนาโดยใช้ระบบแบบ Client/Server ถ้าต้องการพัฒนาเกมที่รองรับผู้เล่นได้หลายคน แต่ถ้าเป็นเกมขนาดเล็กก็ควรใช้เทคโนโลยีแบบ Peer-to-Peer และทำการดึงประสิทธิภาพแบบ Peer-to-Peer ออกมาให้เต็มที่เช่นการทำ Host Migration เป็นต้น

7.4 สรุป

ในการพัฒนาโครงงานนี้นั้น ถือได้ว่าได้บรรลุตามจุดประสงค์ที่ตั้งไว้ แม้ว่าการใช้งานส่วนการติดต่อผู้ใช้จะไม่สะดวกและไม่สวยงามเท่าที่ควรจะเป็น แต่ก็ สามารถเข้าใจถึงหลักการการพัฒนาเกมสามมิติผ่านระบบเครือข่ายได้

บรรณานุกรม

[1] ชัยวัฒน์ คำรัตน์, "Game Engine เล่ม 1", พ.ศ 2542

[2] "Message Board", www.thaigamedevx.com, พ.ศ 2544

[3] "Message Board", www.gamedev.net, พ.ศ 2544

[4] Todd Barron , "MULTIPLAYER GAME PROGRAMMING",Prima Tech Publishing, 1999

[5] Adrian Parez, "Advanced 3D Game Programming Using DirectX 7.0", 2000

[6] Peter J. Kovach, "Direct3D พลังพัฒนาแห่งเกมสามมิติ", 2000

[7] Mark Deloura, "Game Programming Gems 2", 2001

[8] Andre Lamothe, "Tricks of the Game Programming Gurus", 2000