การพัฒนาเกม 3 มิติแบบออนไลน์ด้วยไดเร็กเอ็กซ์

3D ONLINE GAME DEVELOPMENT WITH DIRECTX

นายปริญญา พิทธยาพิทักษ์
นางสาวปรีชญา ศรีสมบัติ
นายพิพัฒน์ ประทีปอมรกุล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

การพัฒนาเกม 3 มิติแบบออนไลน์ด้วยไดเร็กเอ็กซ์

3D ONLINE GAME DEVELOPMENT WITH DIRECTX

โดย

นายปริญญา พิทธยาพิทักษ์

นางสาวปรีชญา ศรีสมบัติ

นายพิพัฒน์ ประทีปอมรกุล

อาจารย์ที่ปรึกษา

ดร. วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเกม 3 มิติแบบออนไลน์ด้วยไดเร็กเอ็กซ์

3D ONLINE GAME DEVELOPMENT WITH DIRECTX

ผู้จัดทำ

1. นาย ปริญญา พิทยาพิทักษ์ รหัสประจำตัว 42010193
2. นางสาว ปรีชญา ศรีสมบัติ รหัสประจำตัว 42010195
3. นาย พิพัฒน์ ประทีปอมรกุล รหัสประจำตัว 42010234

_____ อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การพัฒนาเกม 3 มิติแบบออนไลน์ด้วยไดเร็กเอ็กซ์

นายปริญญา พิทธยาพิทักษ์ 42010193
นางสาวปรีชญา ศรีสมบัติ 42010195
นายพิพัฒน์ ประทีปอมรกุล 42010234
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

การพัฒนาเกม 3 มิติ เป็นโครงงานที่กล่าวถึงการพัฒนาโปรแกรมคอมพิวเตอร์ประเภทเกม 3 มิติ โดยใช้ไดเร็กเอ็กซ์มาเป็นส่วนประกอบของโปรแกรมในด้านมัลติมีเดีย ซึ่งเริ่มต้นตั้งแต่การพัฒนาเกมเอนจิน ซึ่งเป็นหัวใจสำคัญสำหรับการสร้างเกม สำหรับการเขียนโปรแกรมเพื่อพัฒนาเกม 3 มิติ ทั้งหมดนี้อาศัยการโปรแกรมในเชิงวัตถุ ซึ่งใช้ภาษา C++ ในการพัฒนา

ขอบเขตการทำงานของโครงงานนี้ ในขั้นแรกจะทำการศึกษาถึงสถาปัตยกรรมของไดเร็กเอ็กซ์ เพื่อนำมาช่วยในการพัฒนาโปรแกรมเกมในด้านต่างๆ อาทิ การแสดงผลทางด้านกราฟิก, เสียง และการติดต่อกับอุปกรณ์อินพุต รวมถึงศึกษาทฤษฎีพื้นฐานที่จะนำไปสู่การพัฒนาเกมเอนจิน จากนั้นจึงทำการออกแบบวางแนวความคิดโครงสร้างและส่วนประกอบของเกมเอนจิน และเริ่มพัฒนาส่วนประกอบของเอนจินพร้อมทั้งศึกษาถึงผลลัพธ์ของวิธีการต่างๆ แล้วเลือกใช้อย่างเหมาะสม โดยเริ่มจากส่วนกราฟิก, เสียง, อุปกรณ์อินพุตของเกม และส่วนประกอบอื่นๆ ที่จำเป็นในตอนเริ่มต้นของการพัฒนาเกมเอนจิน

ในขั้นตอนถัดมา จะนำเกมเอนจินที่พัฒนาขึ้นมา นำมาพัฒนาเกม 3 มิติแบบออนไลน์ เพื่อทำการทดสอบความสามารถของเกมเอนจิน และความเป็นไปได้ในการนำเกมเอนจินไปใช้งานได้จริงในการพัฒนาเกม

3D ONLINE GAME DEVELOPMENT WITH DIRECTX

Prinya Pithayapitak	42010193
Preechaya Srisombut	42010195
Pipat Prateepamornkul	42010234
Dr. Worawat Limpoka	Advisor

ABSTRACT

3D game development is the project that describes the use of DirectX Application Programming Interface (API) as the multimedia component to develop 3D game which emphasize on game engine development that is the core of game creating. C++ is The object oriented programming language that is used for the development of project.

This project covers, first of all, studying the architecture of DirectX and the way to implement them for game development in many ways such as a graphics system, sound effects and interfacing with input devices and also studying the basic theory of game engine development. Then use the concept to design the game engine components and beginning to develop that component for learning the result of many methods to choose the appropriate way. The programming will begin from graphics, sound, input interfacing and the other components that are necessary in the beginning step of game engine development and also 3D modeling field.

In the last step, 3D online game development is developed by game engine so that game engine is tested and test the possibility of using game engine.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษาปริญญานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่ และยังให้กำลังใจ เอาใจใส่เสมอมา ในทุกๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณ และขอกราบขอบพระคุณมา ณ ที่นี้

ปริญญา พิทธยาพิทักษ์
ปรีชญา ศรีสมบัติ
พิพัฒน์ ประทีปอมรกุล

สารบัญ

	หน้าที่
บทคัดย่อภาษาไทย	I
บทคัดย่อภาษาอังกฤษ	II
กิตติกรรมประกาศ	III
สารบัญ	IV
สารบัญภาพ	VII
บทที่ 1 บทนำ	1
1.1 ความสำคัญและที่มา	1
1.2 วัตถุประสงค์ของโครงงาน	2
1.3 ขอบเขตของโครงงาน	2
1.4 ประโยชน์ที่คาดว่าจะได้รับ	3
1.5 วิธีในการดำเนินงาน	3
บทที่ 2 แนะนำไดเร็กเอ็กซ์	4
2.1 ไดเร็กเอ็กซ์คืออะไร	4
2.2 จุดมุ่งหมายของไดเร็กเอ็กซ์	4
2.3 สถาปัตยกรรมของไดเร็กเอ็กซ์	4
2.4 ส่วนประกอบของไดเร็กเอ็กซ์	8
บทที่ 3 ทฤษฎีและหลักการของเกม 3 มิติ	10
3.1 ระบบพิกัด 3 มิติ	10
3.2 เวอร์เท็กซ์	12
3.3 เวกเตอร์	12
3.4 เมทริกซ์และทรานส์ฟอร์มเมชัน 3 มิติ	14
3.5 การแสดงผลภาพ 3 มิติด้วย Direct3D	17
บทที่ 4 เกมเอนจิน	23
4.1 เกมเอนจินคืออะไร	23
4.2 รูปแบบของเกมเอนจิน	23
4.3 ส่วนประกอบของเกมเอนจินที่พัฒนา	24
บทที่ 5 เอนจินส่วนแอพพลิชัน	26
5.1 คลาส cApplication	26
บทที่ 6 เอนจินส่วนกราฟิก	28
6.1 คลาส cGraphics	28
6.2 คลาส cTexture	29

สารบัญ

(ต่อ)

	หน้าที่
6.3 คลาส cMaterial	30
6.4 คลาส cLight	30
6.5 คลาส cFont	31
6.6 คลาส cVertexBuffer	32
6.7 คลาส cWorldPosition	33
6.8 คลาส cCamera	34
6.9 คลาส cMesh	35
6.10 คลาส cObject	35
บทที่ 7 เอนจินส่วนอินพุต	37
7.1 คลาส cInput	37
7.2 คลาส cInputDevice	37
บทที่ 8 เอนจินส่วนเสียง	40
8.1 คลาส cSound	40
8.2 คลาส cSoundData	41
8.3 คลาส cSoundChannel	41
8.4 คลาส cMusicChannel	43
8.5 คลาส cDLS	43
บทที่ 9 เอนจินส่วนเน็ตเวิร์ค	45
9.1 คลาส cNetworkAdapter	45
9.2 คลาส cNetworkServer	45
9.3 คลาส cNetworkClient	46
บทที่ 10 MMORPG	48
10.1 MMORPG คืออะไร	48
10.2 ลักษณะของเกมแบบ MMORPG	48
10.3 สถาปัตยกรรมของเกมแบบ MMORPG	50
บทที่ 11 การทำงานส่วนเซิร์ฟเวอร์ และการสื่อสารกับไคลเอนต์	52
11.1 ขั้นตอนการทำงานส่วนเซิร์ฟเวอร์	52
11.2 แพ็คเกจที่ใช้ในแต่ละขั้นตอน	53
บทที่ 12 การทำงานส่วนไคลเอนต์	61

สารบัญ
(ต่อ)

หน้าที่

12.1 การออกแบบคลาสส่วนไคลเอนต์ 61

12.2 คลาส cFrustum, cNodeTreeMesh และ cLandscape 61

12.3 คลาส cRelativeCamera และ cNumber 62

12.4 คลาส cCreature, cMonster และ cPlayer 63

12.5 คลาส cMonsterArray และ cPlayerList 64

บทที่ 13 เทคนิคในการทำงานฝั่งไคลเอนต์ 66

13.1 Viewing Frustum และ Node Tree Engine 66

13.2 Billboard, Alpha blending และ Alpha testing 71

13.3 Third-person View 74

บทที่ 14 บทวิจารณ์และสรุป 77

บรรณานุกรม 78

สารบัญภาพ

หน้าที่

รูปที่ 2-1 สถาปัตยกรรมของไดเร็กเอ็กซ์ 5

รูปที่ 2-2 ภาพโดยรวมของ COM 6

รูปที่ 2-3 อินเตอร์เฟสของ COM อ็อบเจ็กต์ 6

รูปที่ 3-1 แสดงระบบพิกัดคาร์ทีเซียนแบบมือซ้ายและแบบมือขวา 10

รูปที่ 3-2 แสดงระบบพิกัดทรงกระบอก 11

รูปที่ 3-3 แสดงระบบพิกัดทรงกลม 12

รูปที่ 3-4 แสดงตัวอย่างการนำเวอร์เท็กซ์มาใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติ 12

รูปที่ 3-5 แสดงตัวอย่างเวกเตอร์ 13

รูปที่ 3-6 แสดง Pipeline การทำงานของ Direct3D 17

รูปที่ 3-7 Viewing Frustum 19

รูปที่ 3-8 Viewing Frustum มองจากแนวแกน X 20

รูปที่ 3-9 การแปลงจาก Viewing Frustum 20

รูปที่ 3-10 Direct3D Viewport 21

รูปที่ 4-1 ระดับของเกมเอนจิน 24

รูปที่ 5-1 คลาสไดอะแกรมของ cApplication 26

รูปที่ 6-1 คลาสไดอะแกรมของ cGraphics 28

รูปที่ 6-2 คลาสไดอะแกรมของ cTexture 29

รูปที่ 6-3 คลาสไดอะแกรมของ cMaterial 30

รูปที่ 6-4 คลาสไดอะแกรมของ cLight 31

รูปที่ 6-5 คลาสไดอะแกรมของ cFont 32

รูปที่ 6-6 คลาสไดอะแกรมของ cVertexBuffer 32

รูปที่ 6-7 คลาสไดอะแกรมของ cWorldPosition 33

รูปที่ 6-8 คลาสไดอะแกรมของ cCamera 34

รูปที่ 6-9 คลาสไดอะแกรมของ cMesh 35

รูปที่ 6-10 คลาสไดอะแกรมของ cObject 36

รูปที่ 7-1 คลาสไดอะแกรมของ cInput 37

รูปที่ 7-2 คลาสไดอะแกรมของ cInputDevice 38

รูปที่ 8-1 คลาสไดอะแกรมของ cSound 40

รูปที่ 8-2 คลาสไดอะแกรมของ cSoundData 41

รูปที่ 8-3 คลาสไดอะแกรมของ cSoundChannel 42

สารบัญภาพ
(ต่อ)

หน้าที่

รูปที่ 8-4 คลาสไดอะแกรมของ cDLS 44

รูปที่ 9-1 คลาสไดอะแกรมของ cNetworkAdapter 45

รูปที่ 9-2 คลาสไดอะแกรมของ cNetworkServer 46

รูปที่ 9-3 คลาสไดอะแกรมของ cNetworkClient 47

รูปที่ 10-1 แสดงการทำงานแบบไคลเอนต์ – เซิร์ฟเวอร์ 50

รูปที่ 11-1แสดงขั้นตอนการทำงานของเซิร์ฟเวอร์ 53

รูปที่ 12-1 คลาสไดอะแกรมของคลาส cFrustum, cNodeTreeMesh และ cLandscape 61

รูปที่ 12-2 คลาสไดอะแกรมของ cRelativeCamera และ cNumber 62

รูปที่ 12-3 การแสดงตัวเลขโดยใช้ cNumber 63

รูปที่ 12-4 คลาสไดอะแกรมของคลาส cCreature, cMonster และ cPlayer 64

รูปที่ 12-5 คลาสไดอะแกรมของ cMonsterArray 65

รูปที่ 12-6 คลาสไดอะแกรมของ cPlayerList 65

รูปที่ 13-1 Viewing Frustum 67

รูปที่ 13-2 ตัวแปร a, b, c และ d ของระนาบ 68

รูปที่ 13-3 การแบ่งโหนดแบบ quadtree และ octree 69

รูปที่ 13-4 การแบ่งโหนดออกเป็นโหนดย่อยๆ 70

รูปที่ 13-5 โหนดที่มีบางส่วนอยู่ในฟรัสทัมจะถูกเรนเดอร์ 71

รูปที่ 13-6 การทำ Billboarding 72

รูปที่ 13-7 การทำ Alpha Testing 73

รูปที่ 13-8 ตัวแปรมุมกล้อง Length, Zeta และ Alpha 74

รูปที่ 13-9 ภาพยืนในทิศต่างๆ 75

รูปที่ 13-10 Vertex ที่ใช้ทำรูปสี่เหลี่ยมเพื่อแสดงภาพ 75

รูปที่ 13-11 การเก็บค่าทิศทางของตัวละคร 76

รูปที่ 13-12 การคำนวณหาภาพตัวละครในทิศทางที่ควรเห็น 76

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

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

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

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

ทางบริษัทไมโครซอฟท์ได้เล็งเห็นว่าแอพพลิเคชันทางด้านเกมนั้นมีผู้ใช้งานกันอย่างแพร่หลาย จึงมีแนวคิดที่จะให้ผู้พัฒนาแอพพลิเคชันทางด้านเกมและมัลติมีเดียหันมาพัฒนาบนระบบปฏิบัติการ Windows ดังนั้นบริษัทไมโครซอฟท์จึงพัฒนาไดเร็กเอ็กซ์ (DirectX) ขึ้นมา ซึ่งเป็นไลบรารีทางด้านมัลติมีเดียที่มีจุดมุ่งหมายหลักๆ ดังต่อไปนี้

- ไดเร็กเอ็กซ์ประกอบด้วยไลบรารีที่ทำงานในระดับต่ำ ซึ่งทำงานได้รวดเร็วและไม่มีข้อจำกัดในการพัฒนาแอพพลิเคชันทางด้านเกม

- โครงสร้างของไดเร็กเอ็กซ์ต้องยกภาระในเรื่องฮาร์ดแวร์จากผู้พัฒนาแอพพลิเคชันไปสู่ผู้ผลิตฮาร์ดแวร์ ซึ่งผู้ผลิตฮาร์ดแวร์ต้องเป็นผู้สร้างไดรเวอร์สำหรับผลิตภัณฑ์ของตน และให้ผู้พัฒนาแอพพลิเคชันนั้นสามารถใช้ความสามารถล่าสุดที่มีอยู่ในฮาร์ดแวร์นั้นๆ ได้
- ไดเร็กเอ็กซ์นั้นต้องอนุญาตให้ผู้พัฒนาแอพพลิเคชันสามารถพัฒนาแอพพลิเคชันออกมาในรูปแบบของ Windows Application ที่สามารถทำงานบนเดสท็อปได้และสามารถทำงานร่วมกับฟังก์ชันต่างๆ ที่มีอยู่บนระบบปฏิบัติการ Windows ได้
- แอพพลิเคชันที่ถูกพัฒนาโดยใช้ไดเร็กเอ็กซ์นั้นต้องมีประสิทธิภาพที่อยู่สูงกว่าหรืออย่างน้อยที่สุดต้องเทียบเท่ากับประสิทธิภาพของแอพพลิเคชันที่ทำงานบนระบบปฏิบัติการ MS-DOS

ซึ่งในปัจจุบันแอพพลิเคชันทางด้านเกมและมัลติมีเดียได้หันมาพัฒนาบนระบบปฏิบัติการ Windows กันหมดแล้ว เพราะไดเร็กเอ็กซ์นั้นทำให้แอพพลิเคชันมีประสิทธิภาพมาก เช่น สามารถใช้ความสามารถของการ์ดเร่งความเร็วกราฟิกแบบ 3 มิติที่ติดตั้งอยู่บนเครื่องคอมพิวเตอร์และชุดคำสั่งพิเศษของไมโครโปรเซสเซอร์ได้ อีกทั้งยังช่วยให้ผู้พัฒนาแอพพลิเคชันไม่ต้องมายุ่งเกี่ยวในส่วนของฮาร์ดแวร์ที่มีอยู่มากมายหลายยี่ห้ออีกต่อไป โดยที่ยังสามารถเข้าถึงการทำงานในระดับต่ำของฮาร์ดแวร์ได้ ต่อมามีบริษัทซอฟต์แวร์ส่วนใหญ่ได้เล็งเห็นถึงการใช้งานไดเร็กเอ็กซ์ในการพัฒนาโปรแกรม 3 มิติ ว่าเป็นงานที่ต้องใช้เวลาสูง ทั้งในด้านการศึกษาและพัฒนา หลายๆ บริษัทจึงพัฒนา 3D เอนจินของตัวเองออกมา ซึ่งช่วยลดเวลาในการพัฒนาโปรแกรม 3 มิติลงได้มาก

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานของไดเร็กเอ็กซ์และนำไปใช้ในการพัฒนาเกมแอพพลิเคชัน
2. เพื่อพัฒนาชุดคำสั่งและนำไปสร้างเกมเอนจินเพื่อให้สามารถพัฒนาเกมได้สะดวกยิ่งขึ้น
3. สามารถแสดงให้เห็นถึงการนำไปใช้งานได้จริงของเกมเอนจินแบบไลบรารีที่สร้างขึ้น ในหลากหลายแนวทาง
4. เพื่อศึกษาและทดลองสร้างเกมตามแนวเกมแบบใหม่ คือ MMORPG (Massively-Multiplayer Online Role-playing Game)

1.3 ขอบเขตของโครงงาน

โครงงานที่ได้จัดทำขึ้นนี้เป็นการพัฒนาซอฟท์แวร์ โดยรวบรวมชุดคำสั่งของไดเร็กเอ็กซ์ที่มีลักษณะการทำงานร่วมกันมารวมเข้าไว้ด้วยกัน ซึ่งจะได้ชุดคำสั่งที่ช่วยให้เกิดความสะดวกและรวดเร็วในการพัฒนาซอฟท์แวร์โดยเฉพาะอย่างยิ่งการพัฒนาเกม โดยเราเรียกชุดคำสั่งเหล่านี้ว่าเกมเอนจิน โดยเราจะทำการแบ่งฟังก์ชันหลักภายในเกมเอนจินออกเป็นส่วนย่อย เนื่องจากแต่ละส่วนนั้นมีความชัดเจนในการทำงานต่างกันออกไป ดังนั้นแต่ละกลุ่มจะทำการพัฒนาส่วนย่อยต่างๆ นี้แล้วนำมารวมเข้าไว้ด้วยกัน เป็นชุดคำสั่งไลบรารี เพื่อการทำงานในการพัฒนาเกมที่สมบูรณ์

เมื่อพัฒนาเกมเอนจินในระดับแรกเสร็จเรียบร้อยแล้ว จะมีการนำเกมเอนจินที่ได้นี้ไปประยุกต์และพัฒนาแอพพลิเคชันต่างๆ ซึ่งจะพัฒนาต่อเป็นเกมออนไลน์แบบ MMORPG คือผู้เล่นสามารถเล่นพร้อมกันได้ในเวลาเดียวกัน โดยจะทำการเชื่อมต่อผ่านระบบเครือข่ายอินเตอร์เน็ต

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทักษะและความสามารถในการพัฒนาเกม โดยสามารถสร้างเกม 3 มิติได้
2. ความรู้ทางคณิตศาสตร์และอัลกอริทึมในการสร้างเกม เช่น การหมุนภาพ การสร้างภาพเคลื่อนไหว การใช้ Tree ในการเพิ่มประสิทธิภาพในการแสดงผล
3. เทคนิคต่างๆ ที่ใช้ในการสร้างเกม 3 มิติ
4. การออกแบบโปรแกรมเชิงคอมโพเน็นต์
5. การใช้ไดเร็กเอ็กซ์ในการสร้างแอพพลิเคชันใช้งาน และติดต่อกับอุปกรณ์มัลติมีเดีย
6. ความรู้ทางภาษา C++ ที่ใช้ในการพัฒนาเกมแอพพลิเคชัน

1.5 วิธีในการดำเนินงาน

1. ทำการศึกษาและค้นคว้าความรู้ทางด้านการเขียนเกม 3 มิติและไดเร็กเอ็กซ์ จากหนังสือและอินเตอร์เน็ต
2. ศึกษาแนวทางการพัฒนาและความรู้ที่ต้องใช้เพิ่มเติมจากความรู้พื้นฐาน
3. เลือกและศึกษาเครื่องมือที่ใช้ในการพัฒนา
4. ออกแบบโครงสร้างและแบ่งงานตามกลุ่มตามงานที่ได้รับมอบหมาย
5. สร้างเกมเอนจินในแต่ละส่วนตามโครงสร้างที่ออกแบบไว้
6. ทำการทดสอบเกมเอนจินที่ได้สร้างขึ้น
7. ทำการรวบรวมส่วนต่างๆ ของเกมเอนจินจากแต่ละกลุ่มให้กลายเป็นเกมเอนจินที่สามารถนำไปใช้งานได้
8. วางโครงสร้างเกมโดยนำเอาเกมเอนจินที่ได้พัฒนาไปใช้ในการพัฒนาเกมต่อ ซึ่งจะนำไปสร้างเกมแบบ MMORPG
9. สร้างโมเดล 3 มิติและสร้างภาพกราฟิกที่ต้องใช้งานในเกม
10. ศึกษาวิธีการทำงานของเกมแบบ MMORPG และศึกษาอัลกอริทึมที่ใช้ในเกมเพื่อเพิ่มประสิทธิภาพในการทำงาน
11. หาทฤษฎีและแบบจำลองการทำงานของเกม
12. สร้างเกมตามโครงสร้างที่ได้ออกแบบไว้
13. ทดสอบการทำงานของเกม และหาข้อผิดพลาดของเกมที่พัฒนาขึ้น
14. ทำการรวบรวมผลทดสอบ วิเคราะห์ปัญหาที่เกิดขึ้น และหาแนวทางในการแก้ปัญหาเพื่อนำมาสรุปเป็นข้อมูลในการพัฒนาขั้นต่อไป

บทที่ 2
แนะนำไดเร็กเอ็กซ์

2.1 ไดเร็กเอ็กซ์คืออะไร

ไดเร็กเอ็กซ์คือชุดของคำสั่ง API (Application Programming Interface) ที่ช่วยให้ผู้พัฒนาสามารถสร้างแอพพลิเคชันที่ทำงานทางด้านมัลติมีเดียได้อย่างสะดวกและรวดเร็ว โดยไม่จำเป็นต้องติดต่อกับฮาร์ดแวร์โดยตรง และสามารถทำงานได้กับอุปกรณ์ทุกตัวที่สนับสนุนการทำงานของไดเร็กเอ็กซ์ และอยู่บนระบบปฏิบัติการ Windows โดยผู้พัฒนาไม่จำเป็นต้องคำนึงถึงความเข้ากันได้กับแอพพลิเคชัน

2.2 จุดมุ่งหมายของไดเร็กเอ็กซ์

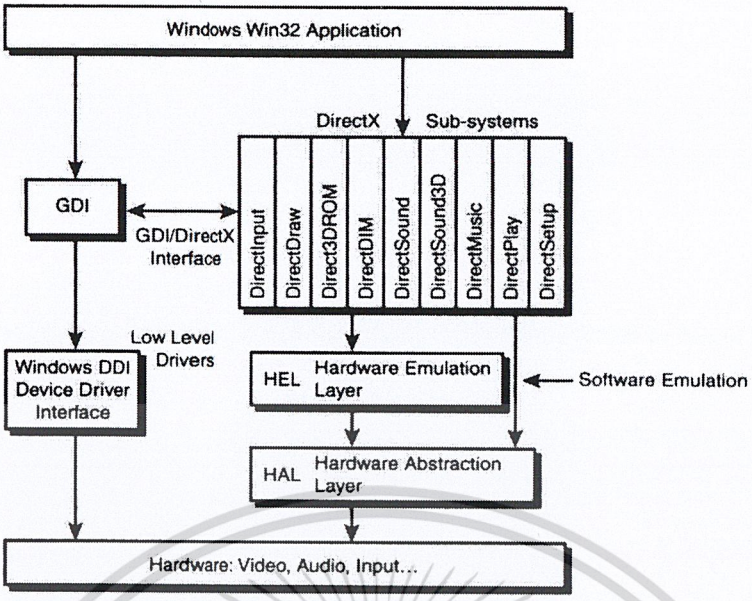
เป้าหมายที่สำคัญที่สุดของไดเร็กเอ็กซ์ คือการจัดหาชุดคำสั่งและเครื่องมือให้แก่นักพัฒนา เพื่อให้นักพัฒนาสามารถพัฒนาแอพพลิเคชันที่ทำงานเกี่ยวกับมัลติมีเดียและเกม ซึ่งใช้เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows เป็นมาตรฐาน โดยไม่จำเป็นต้องตระหนักถึงฮาร์ดแวร์ที่รองรับชุดคำสั่งของไดเร็กเอ็กซ์ ซึ่งจะรวมถึงการสนับสนุนหน้าที่การทำงานทางมัลติมีเดียอย่างหลากหลาย

ไมโครซอฟท์ได้พัฒนาไดเร็กเอ็กซ์ โดยมีเป้าหมายพื้นฐานอยู่ 2 อย่างคือ

- เพื่อให้นักพัฒนาแน่ใจได้ว่าแอพพลิเคชั่นทางมัลติมีเดียนั้นสามารถที่จะทำงานบนเครื่องคอมพิวเตอร์ใดๆ ก็ตามที่มี Windows เป็นระบบปฏิบัติการ โดยไม่จำเป็นต้องใส่ใจถึงฮาร์ดแวร์ที่ติดตั้งอยู่ในเครื่องคอมพิวเตอร์
- เพื่อให้แน่ใจว่าผลิตภัณฑ์ที่ผลิตออกมานั้น จะมีข้อดีของการใช้ความสามารถของฮาร์ดแวร์ที่ติดตั้งอยู่ในเครื่องคอมพิวเตอร์ได้อย่างมีประสิทธิภาพสูงสุด เพื่อบรรลุเป้าหมายของการใช้งานอย่างมีประสิทธิภาพและคุณภาพสูงสุด

2.3 สถาปัตยกรรมของไดเร็กเอ็กซ์

ไดเร็กเอ็กซ์ได้สร้างขึ้นมาบนพื้นฐานของคอมโพเน็นต์ โดยใช้โครงสร้างของ HAL (Hardware Abstraction Layer) ซึ่งสามารถซ่อนลักษณะเฉพาะของดีไวซ์ (Device) ที่เกี่ยวพันอยู่กับฮาร์ดแวร์ และเนื่องจากไดเร็กเอ็กซ์ได้รับการออกแบบให้มีความสามารถในการพัฒนาได้เพิ่มมากขึ้นในอนาคต ดังนั้นจึงสามารถรับรองความสามารถของฮาร์ดแวร์เร่งความเร็วชนิดใหม่ที่เข้ามา โดยใช้การทำงานผ่าน HEL (Hardware Emulation Layer)



รูปที่ 2-1 สถาปัตยกรรมของไดเร็กเอ็กซ์

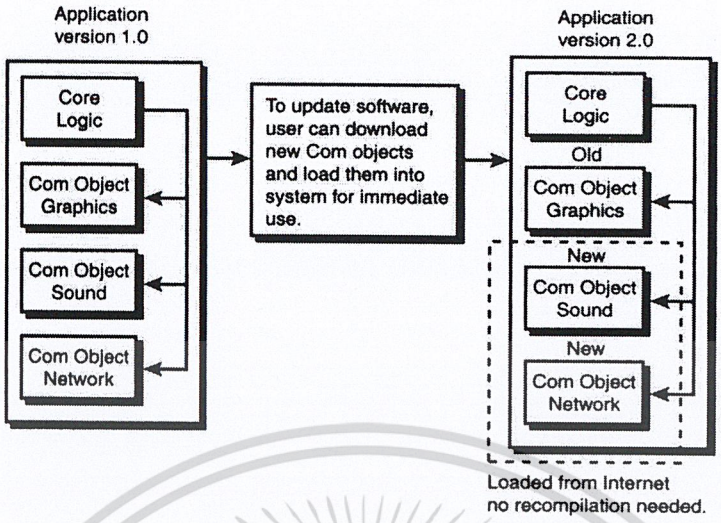
2.3.1 บทบาทของ COM

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

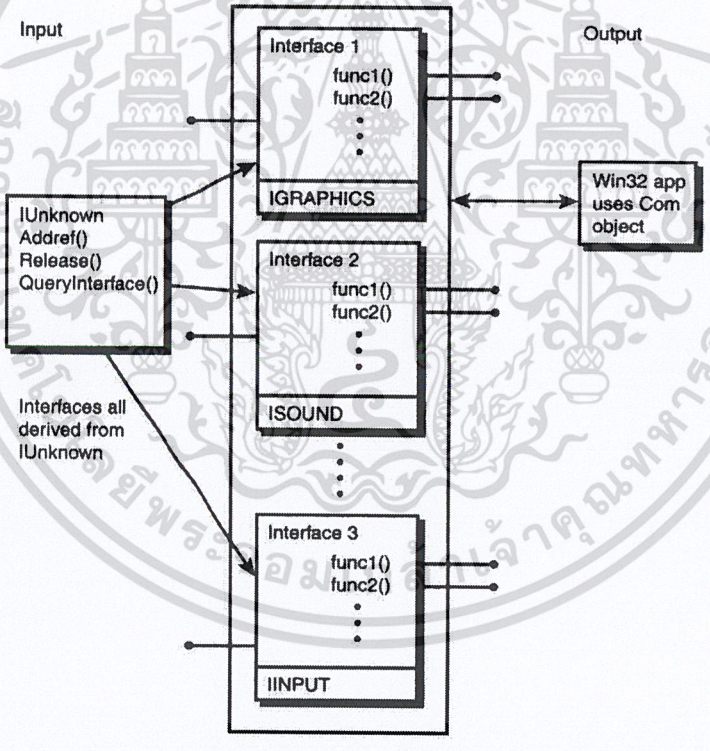
ทุกๆ COM อ็อบเจ็กต์จะต้องยึดติดกันจนเป็นโครงสร้างแบบไบนารี ซึ่งมีความคล้ายคลึงกับโครงสร้างของ Virtual Table ของ C++ ที่ถูกคอมไพล์แล้ว ดังนั้นจะทำให้ทุกภาษาสามารถที่จะสร้าง COM อ็อบเจ็กต์ได้ด้วยการจัดโครงสร้างให้เหมือนกับแบบไบนารีหลังจากการคอมไพล์

ทุกๆ COM อ็อบเจ็กต์นั้นยังจะต้องถูกสืบทอดมาจาก COM อินเตอร์เฟสมาตรฐาน ซึ่งเรียกว่า IUnknown โดย Parent อินเตอร์เฟสจะมีอยู่ด้วยกัน 3 เมธอดคือ AddRef, Delete และ QueryInterface ซึ่ง 2 เมธอดแรกจะจัดการอ็อบเจ็กต์ด้วยการอ้างอิงถึงจำนวน โดยการนับจำนวนของอินสแตนซ์ตั้งแต่เริ่มต้น ส่วนเมธอดสุดท้ายจะจัดการเกี่ยวกับการสอบถามกับ COM อ็อบเจ็กต์ตัวอื่นๆ สำหรับอินเตอร์เฟสที่ต้องการและอินเตอร์เฟสที่มีอยู่ จากนั้นจะคืนค่าพอยเตอร์ที่ชี้ไปที่อินเตอร์เฟสนั้น

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



รูปที่ 2-2 ภาพโดยรวมของ COM



รูปที่ 2-3 อินเตอร์เฟสของ COM อ็อบเจ็กต์

2.3.2 COM และไดเร็กเอ็กซ์

ทุกๆ อินเตอร์เฟสของไดเร็กเอ็กซ์นั้นจะทำการสืบทอดมาจาก IUnknown และมีพื้นฐานในการพัฒนาจาก COM ด้วยเช่นกัน

การใช้ COM ของไมโครซอฟท์ทำให้ได้ชุดคอมโพเน็นต์ของไดเร็กเอ็กซ์ที่มีประสิทธิภาพเพิ่มขึ้น ซึ่งสามารถเข้ากันได้กับไดเร็กเอ็กซ์เวอร์ชั่นก่อนๆ ได้อย่างสมบูรณ์ ด้วยความอิสระ ไม่ขึ้นกับภาษาที่ใช้เขียนโปรแกรมของ COM ทำให้นักพัฒนาสามารถที่จะตัดสินใจเลือกภาษาที่ใช้เขียนโปรแกรมตามที่เหมาะสมกับนักพัฒนาเองมากที่สุด ซึ่งอาจเป็นภาษาที่ถนัด และยังทำให้ได้ประสิทธิภาพตามที่ต้องการ นอกจากนั้นนักพัฒนายังสามารถที่จะเลือกภาษาอะไรก็ตามที่ COM สนับสนุน และสามารถพัฒนาสภาพแวดล้อมได้ตามที่นักพัฒนาต้องการ ด้วยโปรแกรมมิ่งโมเดลของ COM โปรแกรมเมอร์จะแน่ใจได้ว่าแม้ว่าผู้ใช้จะอัพเกรดหรือเปลี่ยนติดตั้งฮาร์ดแวร์ใหม่ หรือจะเปลี่ยนไดเร็กเอ็กซ์เวอร์ชั่นใหม่ก็ตาม ซอฟต์แวร์ก็ยังคงจะสามารถใช้งานได้ตามปกติ โดยไม่จำเป็นต้องทำการติดตั้งใหม่

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

2.3.3 HAL (Hardware Abstraction Layer)

HAL เป็นส่วนที่อยู่ล่างสุดของไดเร็กเอ็กซ์ ซึ่งประกอบไปด้วยส่วนควบคุมของฮาร์ดแวร์ที่ถูกทำโดยผู้ผลิตฮาร์ดแวร์ ที่จะควบคุมฮาร์ดแวร์โดยตรง โดยชั้นนี้จะให้ประสิทธิภาพอย่างมาก เพราะว่าสามารถที่จะติดต่อกับฮาร์ดแวร์ได้โดยตรง ในการทำงานจริงจะไม่สามารถติดต่อกับ HAL ได้เอง แต่ไดเร็กเอ็กซ์จะทำการจัดการให้โดยอัตโนมัติ

2.3.4 HEL (Hardware Abstraction Layer)

HEL จะอยู่บนชั้น HAL โดยทั่วไปไดเร็กเอ็กซ์จะถูกออกแบบให้สามารถใช้ข้อดีของฮาร์ดแวร์ต่างๆ ได้ แต่ถึงอย่างไรไดเร็กเอ็กซ์ก็ไม่สามารถที่จะทำงานได้กับอุปกรณ์ทุกชนิด ตัวอย่างเช่น การเขียนโค้ดแสดงผลกราฟิก สมมติว่าฮาร์ดแวร์ที่กำลังทำงานอยู่สนับสนุนการหมุน (Rotate) และปรับขนาด (Scale) ภาพบิตแมป ซึ่งการเรียกใช้งานไดเร็กเอ็กซ์ เพื่อที่จะปรับขนาดและหมุนภาพบิตแมปได้นั้นจะต้องมีฮาร์ดแวร์ที่สนับสนุนการปรับขนาดและหมุนภาพ ซึ่งถ้าฮาร์ดแวร์สนับสนุนการทำงาน ก็จะทำงานได้เต็มความสามารถ และใช้งานฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ แต่ถ้าฮาร์ดแวร์ไม่สนับสนุนการปรับขนาดและหมุนภาพ การทำงานของ HEL จะเข้ามาทำงานแทนที่ โดย HEL จะทำการจำลองหน้าที่การทำงานของ HAL ด้วยอัลกอริทึมทางซอฟต์แวร์ โดยเราจะไม่ทราบถึงความแตกต่างเลย แต่ถึงอย่างไรก็ตามโค้ดที่ได้ทำงานนั้น ก็จะทำงานได้ช้าลง เพราะว่าเป็นการจำลองการทำงาน จึงไม่สามารถที่จะทำงานได้เหมือนอย่างฮาร์ดแวร์ร้อยเปอร์เซ็นต์

ไดเร็กเอ็กซ์ไดรเวอร์นี้จะรวมเข้าด้วยกันกับไดเร็กเอ็กซ์ API ผ่านลำดับของบัฟเฟอร์อ็อบเจ็กต์ ผลลัพธ์ส่วนมากจะถูกสร้างโดยคอมโพเน็นต์ ไดเร็กเอ็กซ์ API จะถูกเขียนขึ้นเพื่อตอบสนองบัฟเฟอร์อ็อบเจ็กต์ โดยบัฟเฟอร์นี้จะถูกจัดการ และถูกทำให้มีประสิทธิภาพขึ้นตามฮาร์ดแวร์ที่มีอยู่ โดย 2 เลเยอร์ของ

สถาปัตยกรรมของไดเร็กเอ็กซ์และแปลงไปยังเอาท์พุตดีไวซ์ที่เหมาะสมอย่างเป็นระเบียบ ในระดับของภาษาที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้

คุณสมบัติของการจำลองการทำงานของฮาร์ดแวร์ ก็คือชุดของฟังก์ชันที่ทำการจำลองคำสั่งทาง 3 มิติ โดยใช้ซอฟต์แวร์ ถ้าฟังก์ชันที่ต้องการโดยเกมนั้น ไม่สามารถที่จะเข้าถึงฮาร์ดแวร์ได้โดยผ่าน HAL ไดเร็กเอ็กซ์นั้นจะสลับการทำงานมายัง HEL เพื่อที่จะจำลองการทำงานโดยใช้ซอฟต์แวร์แทน ด้วยคุณสมบัตินี้จะช่วยให้นักพัฒนาสามารถที่จะทดสอบหรือตรวจสอบคุณสมบัติทาง 3 มิติในโหมดของซอฟท์แวร์ได้ด้วยตนเอง

สิ่งเหล่านี้จะทำให้ไดเร็กเอ็กซ์มีความยืดหยุ่นในการใช้งานอย่างมาก ซึ่งจะช่วยสนับสนุนการติดตั้งใช้งานฮาร์ดแวร์ใดๆ ไม่ว่าจะเป็นรุ่นเก่าหรือใหม่ ได้อย่างหลากหลาย เช่น ถ้าเรามีแอพพลิเคชันที่ต้องการคุณสมบัติใหม่ๆ ที่อยู่ในฮาร์ดแวร์รุ่นใหม่ ก็ยังจะสามารถที่จะทำงานในฮาร์ดแวร์รุ่นเก่าได้ โดยผ่านชั้น HEL ของไดเร็กเอ็กซ์เพื่อที่จะจำลองความสามารถใหม่ที่ต้องการ ดังนั้น ถ้าเรายิ่งติดตั้งไดเร็กเอ็กซ์รุ่นใหม่เท่าใด จะช่วยในการทำงานของฮาร์ดแวร์ในคอมพิวเตอร์ ดังนั้นเกมที่ถูกเขียนมาในรูปแบบหนึ่ง จะได้รับข้อดีของคุณสมบัติใหม่ของ ฮาร์ดแวร์ที่ติดตั้งอยู่ในเครื่อง โดยการรวมเข้าด้วยกันกับไดเร็กเอ็กซ์

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

ดังนั้นเหตุผลหลักของข้อดีเหล่านี้ เป็นเพราะว่าการออกแบบสถาปัตยกรรมให้เป็น COM ซึ่งสามารถช่วยให้คอมโพเน็นต์อ็อบเจ็กต์สามารถติดต่อสื่อสารกันได้

2.4 ส่วนประกอบของไดเร็กเอ็กซ์

ไดเร็กเอ็กซ์จะประกอบไปด้วยชุดของคำสั่ง API ซึ่งเตรียมไว้ให้นักพัฒนาสามารถที่จะเข้าถึงการทำงานของฮาร์ดแวร์ที่ติดตั้งอยู่บนเครื่องคอมพิวเตอร์อย่างประสิทธิภาพได้โดยตรง ซึ่งชุดคำสั่ง API เหล่านี้จะทำหน้าที่ควบคุมการทำงานของอุปกรณ์ต่างๆ ซึ่งแบ่งออกเป็นส่วนประกอบต่างๆ ดังต่อไปนี้

- DirectX Graphics

  ทำหน้าที่ดูแลและจัดการการแสดงผลทั้งภาพ 2 มิติและ 3 มิติ ช่วยในการให้แสง จัดวางมุมกล้อง และทำภาพเคลื่อนไหว ซึ่งจะทำการควบคุมอุปกรณ์แสดงผล และดึงความสามารถของฮาร์ดแวร์ให้เต็มประสิทธิภาพ

- DirectX Audio

  ทำหน้าที่ดูแลและจัดการเกี่ยวกับเสียง ไม่ว่าจะเป็นเสียงแบบ 3 มิติหรือเสียงแบบเซอร์ราวด์ นอกจากนั้นยังช่วยในการทำเสียงเอฟเฟ็คท์ต่างๆ อีกด้วย

- DirectX Input

  ทำหน้าที่ดูแลและจัดการเกี่ยวกับอุปกรณ์อินพุตที่ใช้ควบคุมต่างๆ เช่น คีย์บอร์ด เมาส์ หรือจอยสติ๊ก เป็นต้น และยังสามารถทำงานกับอุปกรณ์ที่เป็นแบบป้อนกลับ (Force feedback) ได้อีกด้วย

- DirectX Play

  ทำหน้าที่ดูแลและจัดการเกี่ยวกับการติดต่อระหว่างคอมพิวเตอร์ผ่านระบบเครือข่าย ไม่ว่าจะเป็นระบบแลนหรือผ่านโมเด็ม

- DirectX Show

  ทำหน้าที่ดูแลและจัดการเกี่ยวกับการแสดงผลภาพวีดีโอ

- DirectX Setup

  ทำหน้าที่รวบรวมแอพพลิเคชันต่างๆ ที่พัฒนาขึ้นมาด้วยไดเร็กเอ็กซ์ เพื่อให้สามารถแจกจ่ายไปยังบุคคลอื่นๆ ได้ง่าย

บทที่ 3
ทฤษฎีและหลักการเกม 3 มิติ

ในการพัฒนาโปรแกรมเกม 3 มิติ จำเป็นอย่างยิ่งที่จะต้องมีความเข้าใจถึงทฤษฎีพื้นฐานที่จำเป็นสำหรับนำไปประยุกต์ใช้ โดยเฉพาะทฤษฎีทางด้านคอมพิวเตอร์กราฟิก ซึ่งเป็นพื้นฐานสำหรับการพัฒนาโปรแกรมเกม 3 มิติ

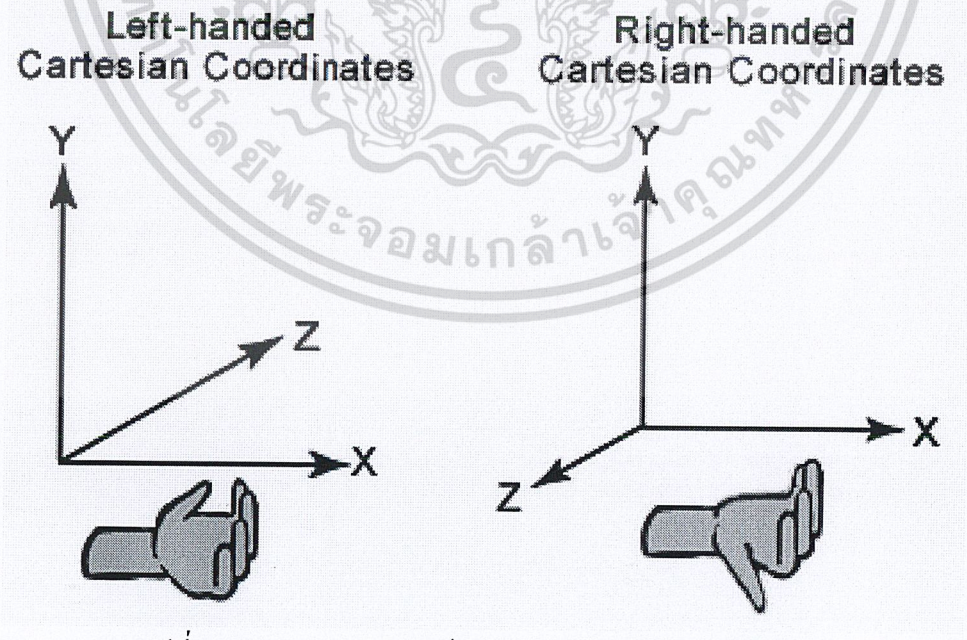
3. ระบบพิกัด 3 มิติ

3.1.1 ระบบพิกัดคาร์ทีเซียน (Cartesian Coordinate System)

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

1. ระบบพิกัดคาร์ทีเซียนแบบมือซ้าย (Left-handed Cartesian Coordinate System)
2. ระบบพิกัดคาร์ทีเซียนแบบมือขวา (Right-handed Cartesian Coordinate System)

โดยแสดงได้ดังภาพดังต่อไปนี้

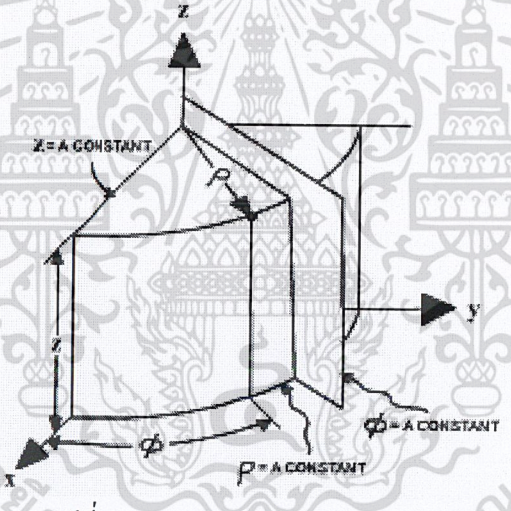


รูปที่ 3-1 แสดงระบบพิกัดคาร์ทีเซียนแบบมือซ้ายและแบบมือขวา

โดยทั่วไป DirectX Graphics API นั้นระบบพิกัดที่ใช้จะเป็นระบบมือซ้าย ในการแสดงผลภาพ 3 มิตินั้นเราจะต้องส่งเวอร์เท็กซ์อาเรย์ที่เราต้องการแสดงผลเข้าไปใน Geometry Pipeline สุดท้ายเราก็จะได้ภาพ 2 มิติออกมาแสดงบนจอภาพ เหตุที่เราต้องแปลงข้อมูลที่เป็น 3 มิติให้เป็น 2 มิตินั้นเพราะหน้าจอของเราไม่สามารถแสดงภาพ 3 มิติตรงๆ ได้จึงต้องทำการส่งข้อมูลเข้า Geometry Pipeline เพื่อให้ได้ผลลัพธ์เป็นภาพ 2 มิติที่สอดคล้องกับข้อมูลนั้นบนมุมมองที่ต้องการ

3.1.2 ระบบพิกัดทรงกระบอก (Cylindrical Coordinate System)

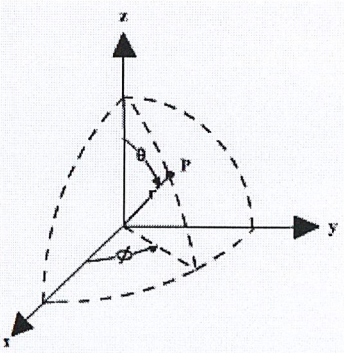
ในระบบพิกัดนี้ เราสามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด (Origin) ซึ่งแทนด้วยสัญลักษณ์ ρ มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเชียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแกนอ้างอิงในแนวระดับ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



รูปที่ 3-2 แสดงระบบพิกัดทรงกระบอก

3.1.3 ระบบพิกัดทรงกลม (Spherical Coordinate System)

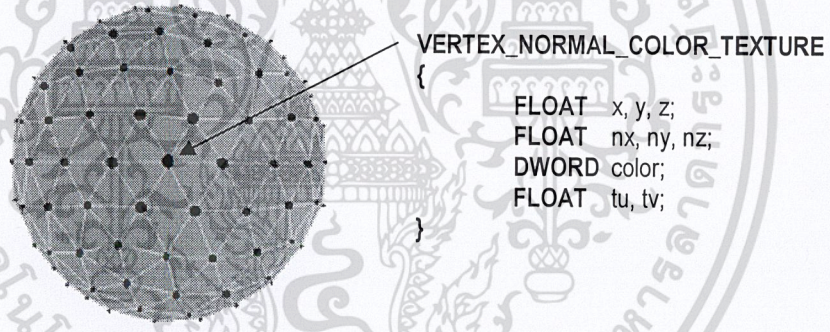
ในระบบพิกัดนี้ เราสามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด ซึ่งแทนด้วยสัญลักษณ์ r มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเชียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแนวระดับ (ซึ่งสัมพันธ์กับระนาบ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



รูปที่ 3-3 แสดงระบบพิกัดทรงกลม

3.2 เวอร์เท็กซ์ (Vertex)

เวอร์เท็กซ์เป็นจุดพิกัดในระบบ 3 มิติ ซึ่งใช้ในการอ้างอิงตำแหน่ง ซึ่งในการประมวลผลทางด้านกราฟิกนั้นเรามักจะกำหนดคุณสมบัติเพิ่มเติมให้กับเวอร์เท็กซ์ เช่น Color, Normal, Texture Coordinate เป็นต้น ซึ่งคุณสมบัติเหล่านี้เมื่อประกอบกับตำแหน่งของเวอร์เท็กซ์แล้วจะนำไปใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติได้ โดยแสดงได้ดังรูปตัวอย่าง

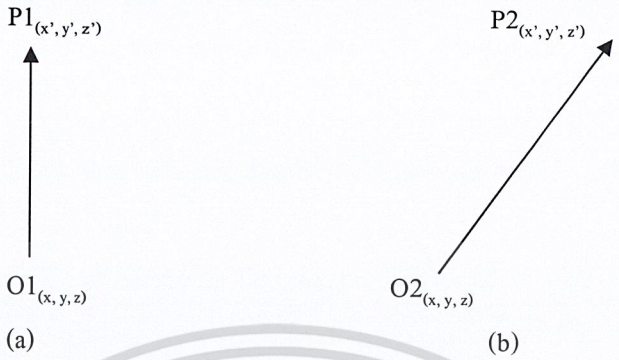


รูปที่ 3-4 แสดงตัวอย่างการนำเวอร์เท็กซ์มาใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติ

3.3 เวกเตอร์ (Vector)

เวกเตอร์เป็นปริมาณที่มีทั้งขนาดและทิศทาง เป็นองค์ประกอบทางคณิตศาสตร์ที่มีความสำคัญมากด้านการประยุกต์ใช้ในงานการประมวลผลกราฟิก ซึ่งจะต่างกับปริมาณสเกลาร์ตรงที่เวกเตอร์นั้นจะมีทิศทาง หรือกล่าวได้อีกอย่างหนึ่งว่าเวกเตอร์มีองค์ประกอบมาจากปริมาณสเกลาร์ n ตัว เพื่อแทนขนาดและทิศทางในระบบ n มิติ

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



รูปที่ 3-5 แสดงตัวอย่างเวกเตอร์

หากเราทำการบวกเวกเตอร์ทั้งสองจะได้ค่าดังนี้

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

เมื่อทำการคูณเวกเตอร์ด้วยปริมาณสเกลาร์ ซึ่งทำได้โดยการคูณปริมาณสเกลาร์กับทุกคอมโพเน็นต์ของเวกเตอร์นั้น ๆ จะได้

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

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

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

เวกเตอร์ที่ขนานกัน (Parallel Vector) หมายถึง เวคเตอร์คู่ใด ๆ ที่มีทิศทางเดียวกันหรือทิศทางตรงข้ามกัน

เวกเตอร์ร่วมระนาบ (Coplanar Vector) หมายถึง เวกเตอร์ตั้งแต่ 3 ตัวขึ้นไปที่อยู่ในระนาบเดียวกัน

3.4 เมทริกซ์และทรานส์ฟอร์มเมชัน 3 มิติ

3.4.1 เมทริกซ์ (Matrix)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$AB = I_n = BA$$

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

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

และถ้า A ไม่มีอินเวิร์ส แล้วจะเรียก A ว่าเป็นเมทริกซ์เอกฐาน (Singular Matrix)

3.4.2 Translation

เป็นการเคลื่อนย้ายตำแหน่งของวัตถุ ซึ่งใช้ Translation Matrix

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

Translation Matrix

3.4.3 Rotation

เป็นการหมุนวัตถุรอบแกน X, Y หรือ Z ซึ่งใช้ Rotation Matrix

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

Rotation Around X Axis Matrix

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

Rotation Around Y Axis Matrix

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

Rotation Around Z Axis Matrix

3.4.4 Scaling

เป็นการย่อหรือขยายวัตถุ ซึ่งใช้ Scaling Matrix

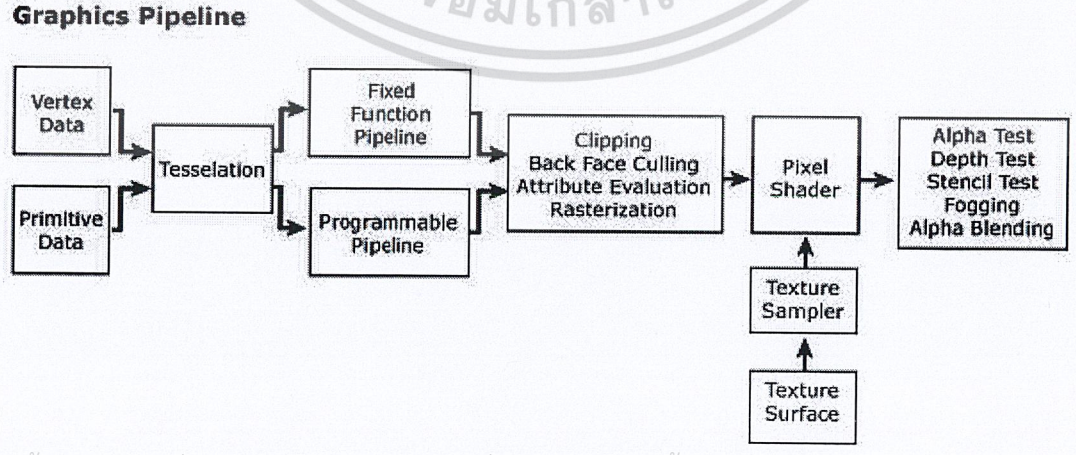
$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Scaling Matrix

3.5 การแสดงผลภาพ 3 มิติด้วย Direct3D

3.5.1 สถาปัตยกรรมของ Direct3D

สถาปัตยกรรมของ Direct3D ประกอบด้วยหน้าที่การทำงานต่างๆ ซึ่งจะรวมกันทำงานแบบ Pipeline ดังที่แสดงตามรูป



รูปที่ 3-6 รูปแสดง Pipeline การทำงานของ Direct3D

3.5.2 ทรานส์ฟอร์มเมชัน (Transformation)

3.5.2.1 การแปรไปสู่พิกัดเวิลด์ (World Transformation)

ขั้นตอนนี้จะเป็นขั้นตอนในการแปลง Local Coordinate ไปเป็น World Coordinate ซึ่งจะเป็นตำแหน่งจริงๆ ในปริภูมิ 3 มิติ ซึ่งวิธีการนั้นเราจะใช้การคูณตำแหน่งในเวอร์เท็กซ์ด้วย Transformation Matrix ต่าง ๆ เพื่อให้ได้ตำแหน่งที่เราต้องการ

ในการแปลงเวอร์เท็กซ์จาก Local Coordinate ให้เป็น World Coordinate นั้นในบางครั้งถ้าการแปลงของเรามีความซับซ้อนมากเราอาจจะต้องใช้เมทริกซ์หลายตัวในการแปลงคูณต่อกันไป (Matrix Concatenation) เพื่อสร้างเมทริกซ์สุดท้ายสำหรับใช้เป็น Transformation Matrix ก่อนแล้วจึงจะเอามาคูณกับตำแหน่งในเวอร์เท็กซ์จริงเพื่อให้ได้ผลลัพธ์สุดท้ายก็ได้

การคูณ Matrix กับตำแหน่งของ Vertex จะเป็นดังต่อไปนี้

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

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

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

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

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

3.5.2.2 การแปรไปสู่พิกัดวิว (View Transformation)

ขั้นตอนนี้ใช้สำหรับแปลง World Coordinate ให้สอดคล้องกับการเห็นของผู้สังเกต เนื่องจาก World Coordinate บอกเพียงตำแหน่งจริงๆ ในพิกัด 3 มิติเท่านั้น ยังไม่มีการอ้างอิงจากผู้สังเกตเลย ดังนั้นจึงต้องมีการบอก ว่าผู้สังเกตอยู่ที่ไหน และมองไปทางไหน ขั้นตอนนี้มักใช้ในการปรับมุมมองของกล้อง โดยการนำวิวเมทริกซ์มาคูณกับ World Coordinate จากการแปรไปสู่พิกัดเวิลด์

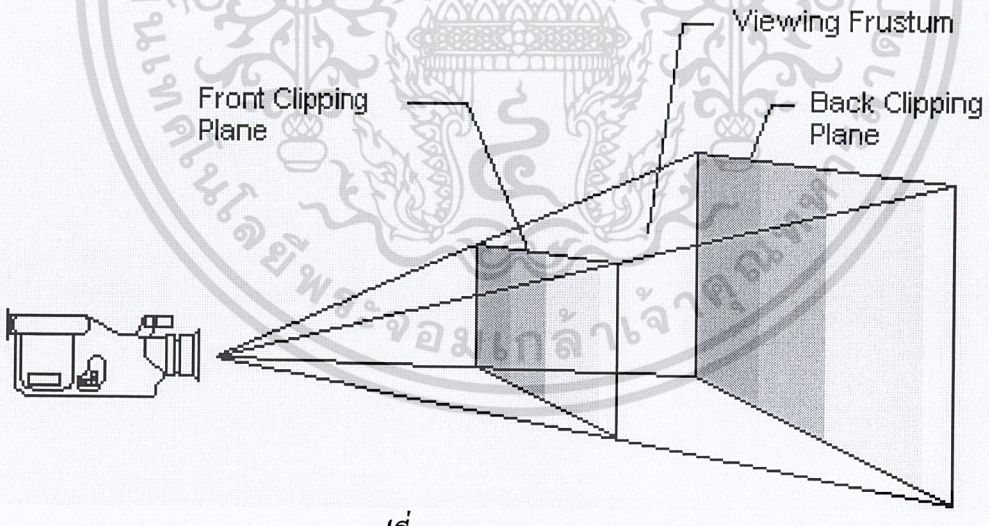
$$\begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ -(u \cdot c) & -(v \cdot c) & -(n \cdot c) & 1 \end{bmatrix}$$

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

วิวเมทริกซ์นั้นให้ข้อมูลที่จำเป็นในการปรับมุมมองของกล้องในเกมเอนจินได้ ซึ่งใช้ในการบอกตำแหน่งและทิศทางของกล้อง

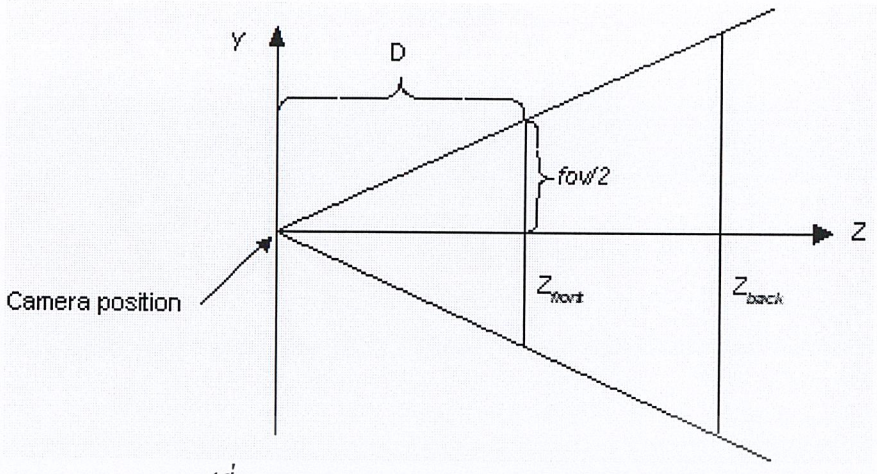
3.5.2.3 การแปรไปสู่พิกัดโปรเจ็กต์ชัน (Projection Transformation)

เมื่อเราได้ตำแหน่งซึ่งอ้างอิงกับผู้สังเกตเรียบร้อยแล้ว ขั้นตอนต่อไปก็คือการแปลง View Coordinate มาเป็นพิกัด 2 มิติ ซึ่งสามารถนำมาวาดได้จริงๆ บนจอภาพ ซึ่งโดยทั่วไปแล้วเราต้องการให้ได้ภาพที่สมจริง คือ วัตถุที่อยู่ไกลจากสายตาจะมีขนาดเล็กลง และวัตถุที่อยู่ใกล้สายตาจะมีขนาดใหญ่ขึ้น ซึ่งเรียกว่า Perspective Projection Transformation ซึ่งเป็นการสร้างภาพฉายมาตกกระทบฉาก คล้ายๆ กับการฉายภาพจากโปรเจคเตอร์

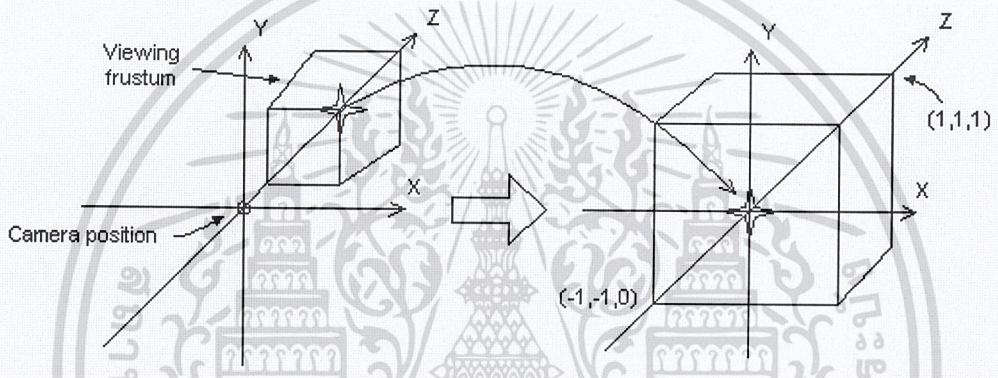


รูปที่ 3-7 Viewing Frustum

วัตถุที่อยู่ใน Viewing Frustum จะถูกนำมาแสดงผล เนื่องจากเป็นส่วนที่จอภาพสามารถเห็นได้ ส่วนวัตถุที่อยู่ภายนอก Viewing Frustum จะถูกตัดออกระหว่างการประมวลผลก่อนที่จะนำไปแสดงผล



รูปที่ 3-8 Viewing Frustum มองจากแนวแกน X



รูปที่ 3-9 การแปลงจาก Viewing Frustum (จากภาพด้านซ้าย) ไปเป็น Cuboid shape (จากภาพด้านขวา) ซึ่งทำให้วัตถุที่อยู่ไกลมีขนาดเล็กลง และวัตถุที่อยู่ใกล้มีขนาดใหญ่ขึ้นตามสัดส่วนของระยะห่างจากสายตาผู้สังเกต

เราสามารถทำ Perspective Projection Transformation ได้โดยการใช้ Perspective Projection Matrix ดังนี้

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

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

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

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

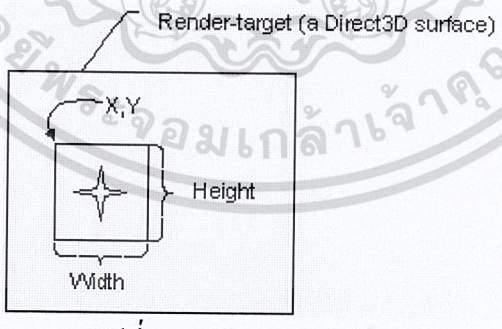
โดยทั่วไปแล้วกราฟิกเอนจินนั้นมักจะมีฟังก์ชันที่ใช้สำหรับทำการคำนวณหา Perspective Projection Matrix โดยให้เรากำหนดค่า Field of View (fov) ตามแนวความสูงของฉาก (โดยทั่วไปมักเป็นแกน Y) ค่า Aspect Ratio (w/h) ค่า Near Clipping Plane (Z_n) และค่า Far Clipping Plane (Z_f)

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

3.5.3 Clipping and Viewport Scaling

Viewport ในที่นี้หมายถึงพื้นที่สี่เหลี่ยมผืนผ้าที่ใช้ในการฉายภาพที่เกิดจากการแปรไปสู่พิกัดโปรเจ็กต์ชัน ซึ่งเป็นขั้นตอนสุดท้ายใน Geometry Pipeline และทำให้ได้ภาพฉายตกลงบน Viewport เพื่อทำกระบวนการ Rasterization ต่อไปในการสร้างภาพเสมือนจริงบนหน้าจอ

Viewport อาจหมายถึงหน้าจอทั้งหน้าจอหรือบางส่วนของหน้าจอก็ได้ ทั้งนี้ขึ้นอยู่กับโปรแกรม ซึ่งโดยทั่วไปแล้วเกมมักจะแสดงผลแบบเต็มจอ แต่การใช้ Viewport อาจมีประโยชน์เมื่อเราต้องการการแสดงผล 3 มิติหลายหน้าต่าง



รูปที่ 3-10 Direct3D Viewport

โดยทั่วไปแล้วเราจะทำการขริบ (Clipping) สิ่งที่มองไม่เห็นบนหน้าจอออกไป ในกรณีที่ Viewport แสดงถึงบางส่วนของหน้าจอนั้น เราจะขริบส่วนของภาพที่อยู่นอกกรอบ Viewport ออกไป ทั้งนี้รวมถึงการขริบในแนวแกน Z ด้วย (เราจะตัดทอนส่วนของภาพที่อยู่ไกลเกิน Viewport ที่กำหนด หรือ ส่วนของภาพที่อยู่ด้านหลัง Near Clipping Plane ของ Viewing Frustum ออก)

กระบวนการนี้โดยมากมักจัดการโดยกราฟิกเอนจินโดยเราเพียงแค่กำหนดค่าขอบเขตของ Viewport และระยะการตัดทอนตามแนวแกน Z ก็เพียงพอแล้ว

บทที่ 4
เกมเอนจิน

4.1 เกมเอนจินคืออะไร

เกมเอนจินคือเครื่องมือที่ช่วยให้ผู้พัฒนาสามารถสร้างเกมได้สะดวก และรวดเร็วมากขึ้น ซึ่งช่วยดูแลและจัดการในเรื่องของการแสดงผล การควบคุมอุปกรณ์อินพุต เสียงเพลง เสียงเอฟเฟ็คต์ต่างๆ และการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ผ่านระบบเครือข่าย โดยไม่จำเป็นต้องศึกษาการทำงานอย่างละเอียด นอกจากนั้นเกมเอนจินยังจะช่วยให้ผู้พัฒนาไม่ต้องเสียเวลาในการพัฒนาเกม เพราะเกมเอนจินจะช่วยจัดการการทำงานบางส่วนให้กับผู้พัฒนาเกมแล้ว เช่น การทำงานในส่วนแสดงผลอาจใช้เพียงแค่คำสั่งเดียวก็สามารถทำงานตามที่ต้องการได้แล้ว

4.2 รูปแบบของเกมเอนจิน

เกมเอนจินโดยทั่วไป จะสามารถแบ่งตามลักษณะของรูปแบบการทำงานของเกมเอนจินได้ 2 ชนิด คือ

4.2.1 เกมเอนจินแบบไลบรารี (Library Game Engine)

จะเป็นเกมเอนจินแบบที่จะรวบรวมชุดคำสั่งที่จำเป็นในการพัฒนาเกมเข้าไว้ด้วยกัน โดยจะแบ่งการทำงานออกเป็นส่วนๆ ตามการทำงาน ซึ่งในการใช้งานจะต้องทำการเพิ่มส่วนของเกมเอนจินเข้าไปรวมกับส่วนของโปรแกรม เพื่อให้ส่วนของโปรแกรมสามารถเรียกใช้งานชุดคำสั่งที่เกมเอนจินได้จัดเตรียมไว้ ซึ่งลักษณะของเกมเอนจินแบบไลบรารีจะสามารถแบ่งได้ออกเป็น 2 ลักษณะ คือ

4.2.1.1 สเตติกไลบรารี

ลักษณะของเกมเอนจินแบบนี้ คือ เวลาใช้งานเกมเอนจินจะรวมส่วนของเกมเอนจินเข้าไปกับส่วนของเกม ดังนั้นถ้ามีการเปลี่ยนแปลงส่วนใดส่วนหนึ่งของเกมเอนจินนั้นจะต้องทำการคอมไพล์ส่วนของเกมใหม่ เพื่อที่จะปรับปรุงการทำงานส่วนของเกมเอนจิน และการใช้งานเกมเอนจินลักษณะนี้จะทำให้ส่วนของเกมมีขนาดที่ใหญ่ เพราะรวมเอาส่วนของเกมเอนจินเข้าไปในส่วนของเกมด้วย

4.2.1.2 ไดนามิกไลบรารี

ลักษณะของเกมเอนจินแบบนี้ คือ เวลาใช้งานจะทำการรวมเข้ากับส่วนของเกม และเวลาเกมเริ่มทำงาน เมื่อใดที่ส่วนของเกมต้องการใช้งานเกมเอนจิน ก็จะทำการไปเรียกส่วนการทำงานของเกมเอนจินขึ้นมาใช้งานในขณะนั้น ดังนั้นเมื่อทำการเปลี่ยนแปลงส่วนใดส่วนหนึ่งของเกมเอนจิน ก็ไม่จำเป็นต้องคอมไพล์ส่วนของเกมใหม่ จึงทำให้ง่ายในการปรับปรุงการทำงานส่วนของเกมเอนจิน และการใช้งานเกม

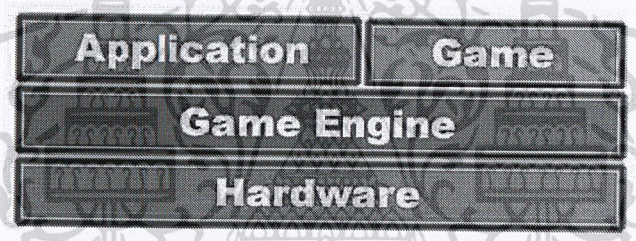
เอนจินลักษณะนี้ส่วนของเกมจะมีขนาดเล็ก เพราะจะแยกส่วนของเกมเอนจินออกไปเป็นอีกส่วนนึง ซึ่งจะไม่เข้าไปรวมกับส่วนของเกม และเป็นลักษณะที่แพร่หลายในการใช้งาน

4.2.2 เกมเอนจินแบบกราฟิก (GUI Game Engine)

เกมเอนจินแบบกราฟิกนั้นจะมีลักษณะเป็นแอพพลิเคชันที่ให้ผู้พัฒนาเกมสามารถสร้างเกมได้โดยไม่จำเป็นต้องรู้ถึงวิธีในการเขียนโปรแกรม และสามารถพัฒนาเกมออกมาออกมาได้ในระยะเวลาอันสั้น ผู้พัฒนาเกมเพียงแค่กำหนดรูปแบบของเกม เนื้อเรื่องของเกมที่ต้องการ จากนั้นใช้เกมเอนจินในการกำหนดส่วนต่างๆ ของเกมให้เป็นไปตามรูปแบบที่กำหนดไว้ จากนั้นก็สามารถทำงานได้แล้ว

4.3 ส่วนประกอบของเกมเอนจินที่พัฒนา

เกมเอนจินที่พัฒนานั้นจะมีลักษณะการทำงาน ซึ่งรวบรวมขึ้นมาเป็นชุดคำสั่ง มีการใช้งานที่ง่าย ซึ่งเกมเอนจินที่พัฒนานี้พัฒนาโดยใช้ไดเร็กเอ็กซ์เป็นพื้นฐาน จะมีส่วนประกอบต่างๆ ซึ่งจะแบ่งตามลักษณะของการทำงาน โดยจะแบ่งออกเป็นส่วนประกอบหลักๆ ดังนี้



รูปที่ 4-1 แสดงระดับของเกมเอนจิน

4.3.1 เกมเอนจินส่วนแอพพลิเคชัน

เกมเอนจินส่วนแอพพลิเคชันเป็นส่วนเริ่มต้นและควบคุมการทำงานของโปรแกรมเกม โดยจะควบคุมกระบวนการทำงานของเกม

4.3.2 เกมเอนจินส่วนกราฟิก

เกมเอนจินส่วนกราฟิกจะช่วยในการแสดงผล การสลับหน้าจอ การให้แสง การโหลดโมเดล กำหนดตำแหน่งกล้อง การทำงานกับวัตถุ 3 มิติ

4.3.3 เกมเอนจินส่วนอินพุต

เกมเอนจินส่วนอินพุตเป็นส่วนติดต่อกับอุปกรณ์อินพุตต่างๆ เช่น คีย์บอร์ด เมาส์ จอยสติ๊ก ซึ่งสามารถที่จะควบคุมการทำงานของอุปกรณ์ต่างๆ

4.3.4 เกมเอนจินส่วนเสียง

เกมเอนจินส่วนเสียงจะดูแลการทำงานในการเอาท์พุตเสียงออกลำโพง และสามารถเล่นเสียงได้หลายช่องทาง ใส่เสียงเอฟเฟ็คต์ และเสียงแบบ 3 มิติ

4.3.5 เกมเอนจินส่วนเน็ตเวิร์ค

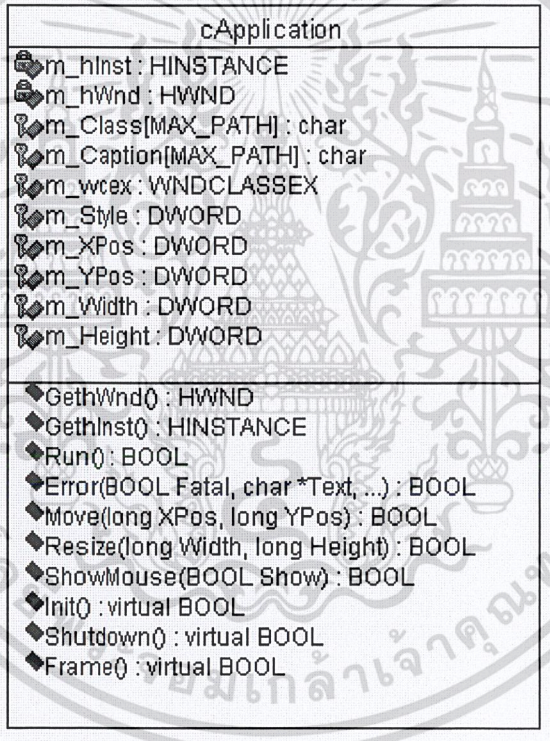
เกมเอนจินส่วนเน็ตเวิร์คจะจัดการติดต่อกับการ์ดเน็ตเวิร์ค โดยจะควบคุมกระบวนการทำงานในการติดต่อสื่อสารกันระหว่างเครื่องคอมพิวเตอร์

บทที่ 5
เอนจินส่วนแอพพลิเคชัน

เอนจินส่วนแอพพลิเคชันเป็นส่วนเริ่มต้นและควบคุมการทำงานของโปรแกรมเกม โดยจะควบคุมกระบวนการทำงานของเกม โดยจะเป็นตัวจัดการสร้างหน้าต่างขึ้นมาให้ โดยมีคลาสที่เกี่ยวข้องดังนี้

5.1 คลาส cApplication

เป็นคลาสที่สร้างและควบคุมการทำงานของโปรแกรม คลาสนี้จะทำการ Register วินโดว์คลาสและสร้างหน้าต่างขึ้นมาให้ และคอยควบคุมดูแลเมสเสจของแอพพลิเคชันที่เราสร้างขึ้นมา



รูปที่ 5-1 คลาสไดอะแกรมของ cApplication

การใช้งานคลาสนี้จำเป็นต้องทำการสืบทอดมาเป็นคลาสใหม่ และจะมีอินสแตนซ์ของคลาสนี้ได้เพียงอินสแตนซ์เดียว เพราะว่าคลาสนี้จะเป็นคลาสหลักในการสร้างแอพพลิเคชันโปรแกรม ซึ่งจะกำหนดขนาดของหน้าต่าง ตำแหน่งของหน้าต่าง ซึ่งจะกำหนดใน Constructor ของคลาสที่สืบทอดจากคลาส cApplication และยังทำการ Register วินโดว์คลาสและทำการสร้างหน้าต่างของแอพพลิเคชันขึ้นมา ถ้าต้องการเปลี่ยนขนาดของหน้าต่างภายหลัง จะใช้เมธอด Resize() และถ้าต้องการย้ายตำแหน่งของหน้าต่างก็จะใช้เมธอด Move() ไปยังตำแหน่งที่ต้องการ

ในการสืบทอดคลาส cApplication นั้น เราจะทำการ Override เมธอดหลักๆ 3 เมธอด คือ Init() ซึ่งจะถูกเรียกทำงานโดยอัตโนมัติ ซึ่งจะถูกเรียกเป็นเมธอดแรกหลังจากทำงานในส่วน Constructor แล้ว เมธอดถัดมาจะเป็นเมธอด Shutdown() ซึ่งจะเป็นเมธอดสุดท้ายที่ถูกเรียกหลังจากเลิกใช้งานแอพพลิเคชัน โดยส่วนใหญ่กำหนดการทำงานในส่วนของเมธอด Init() จะทำเพื่อกำหนดค่าเริ่มต้น หรือทำการจองทรัพยากรที่ต้องการก่อนการใช้งานแอพพลิเคชัน ส่วนเมธอด Shutdown() จะทำเพื่อคืนทรัพยากรที่ทำการจองไว้คืนแก่ระบบ ส่วนเมธอดสุดท้ายคือ Frame() จะเป็นเมธอดที่ถูกเรียกทุกครั้งในการทำงาน ซึ่งจะถูกเรียกใช้งานเมื่อไม่มีเมสเสจเข้ามา โดยจะวนทำงานไปจนกระทั่งผู้ใช้ยกเลิกการทำงานของแอพพลิเคชัน เมธอดนี้จะถูกเรียกใช้ภายในเมธอด Run() ซึ่งอยู่ภายในคลาส cApplication ซึ่งส่วนใหญ่การทำงานของเมธอด Frame() นั้นจะถูกใช้งานเพื่อทำการเรนเดอร์ภาพ 3 มิติ และแสดงผลกราฟิกต่างๆ

นอกจากนั้นยังสามารถที่จะกำหนดการประมวลผลเมสเสจของแอพพลิเคชันได้โดยการ Override เมธอด MsgProc() ซึ่งจะเป็นเมธอดในการจัดการเมสเสจ โดยผู้ใช้สามารถกำหนดการทำงานของเมสเสจได้ตามที่ผู้ใช้ต้องการ

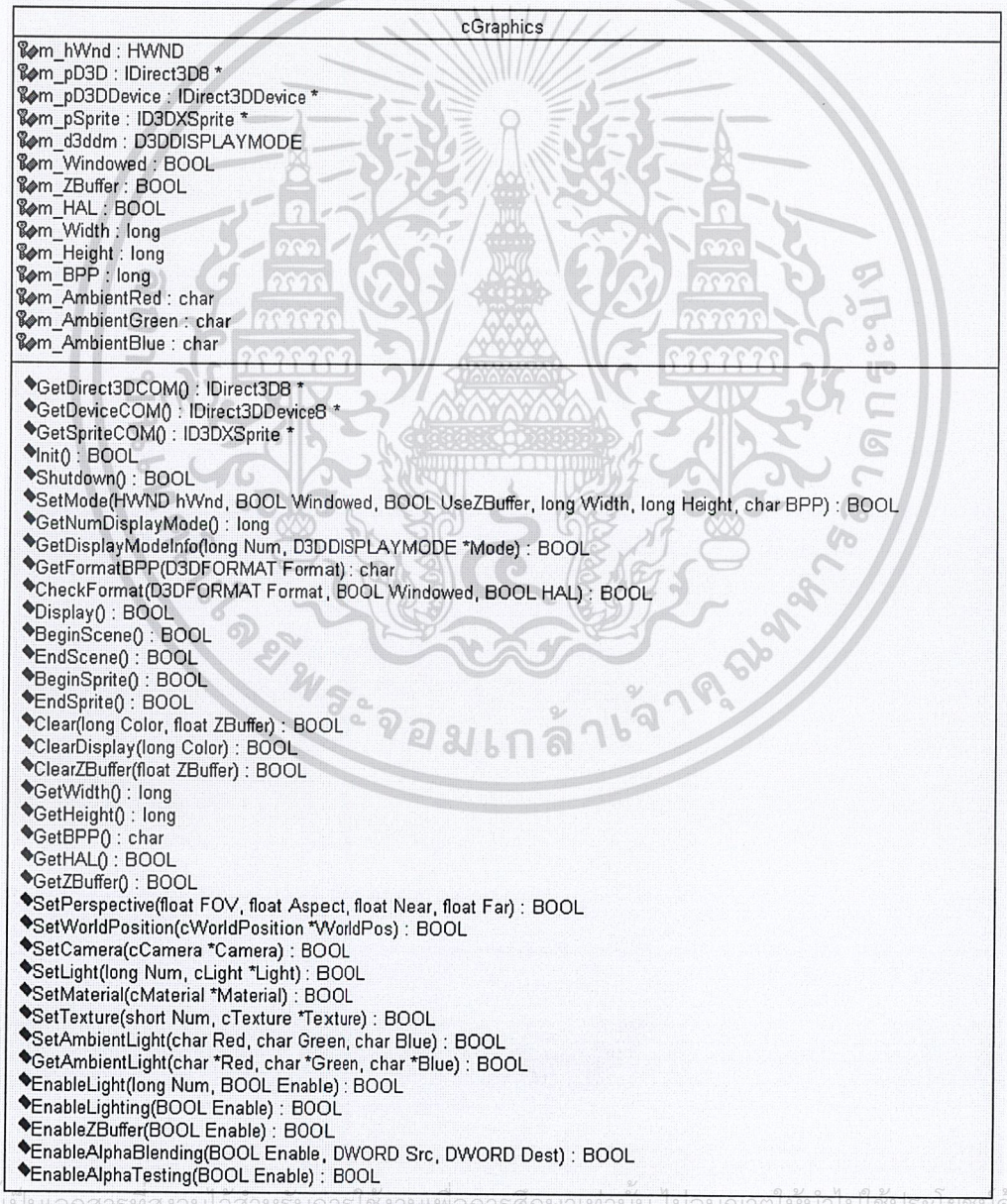
การทำงานของคลาสที่สืบทอดมาจากคลาส cApplication นั้นจะมีแค่เพียงการสร้างอินสแตนซ์ของคลาสที่สืบทอดมา และทำการเรียกเมธอด Run() เพื่อเริ่มการทำงานของแอพพลิเคชัน

บทที่ 6
เอนจินส่วนกราฟิก

เป็นส่วนที่ใช้ควบคุมอุปกรณ์แสดงผล และแสดงผลภาพ 3 มิติ จัดการโมเดล 3 มิติ และการจัดวางมุมกล้องต่างๆ ทำให้การแสดงผลมีประสิทธิภาพ โดยจะแบ่งออกเป็นคลาสดังต่อไปนี้

6.1 คลาส cGraphics

คลาสนี้ทำหน้าที่สร้างสภาพแวดล้อมของแอพพลิชันให้สามารถแสดงผลภาพ 3 มิติได้อย่างมีประสิทธิภาพ และจัดการการแสดงผลภาพ 3 มิติ



รูปที่ 6-1 คลาสไดอะแกรมของ cGraphics

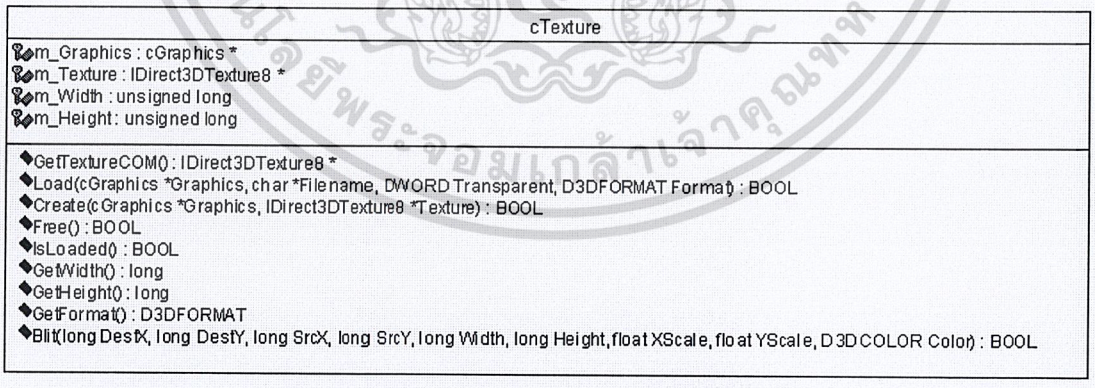
คลาสนี้จะทำการติดต่อกับอุปกรณ์แสดงผล เช่น การ์ดเร่งความเร็วกราฟิก เป็นต้น ซึ่งจะจัดการการแสดงผลในรูปแบบ 3 มิติ ซึ่งจะเริ่มการใช้งานโดยเรียกเมธอด Init() ซึ่งจะกำหนดค่าต่างๆ ให้กับสภาพแวดล้อมที่จะต้องใช้ในการแสดงผลภาพ 3 มิติ จากนั้นจะเริ่มทำงานในการแสดงผล โดยใช้เมธอด SetMode() ซึ่งจะมีลักษณะการแสดงผลอยู่ 2 แบบ คือแบบวินโดว์ (Window) และแบบเต็มหน้าจอ (Full Screen)

การแสดงผลภาพ 3 มิติจะทำโดยใช้เมธอด BeginScene() เพื่อทำการเรนเดอร์ภาพลงใน BackBuffer เมื่อทำการเรนเดอร์ภาพเสร็จเรียบร้อย จะต้องทำการเรียกเมธอด EndScene() เพื่อทำการจบการทำงานในส่วนการเรนเดอร์ภาพลงใน BackBuffer จากนั้นจะทำการแสดงผลภาพ 3 มิติโดยใช้เมธอด Display() ในการสลับ BackBuffer มายัง FrontBuffer เพื่อให้ภาพที่ทำการเรนเดอร์ที่ BackBuffer แสดงผลออกทางจอภาพ ซึ่งในก่อนการเรนเดอร์ภาพใน BackBuffer จะต้องมีการลบภาพที่อยู่ใน BackBuffer ออกเสียก่อนโดยใช้เมธอด Clear() เพื่อที่จะได้เรนเดอร์ภาพเฟรมถัดไป

นอกจากนั้นยังสามารถเปลี่ยนลักษณะการเรนเดอร์ได้โดยใช้เมธอด EnableLighting() ซึ่งจะทำการเปิดปิดการเรนเดอร์แสง เมธอด EnableZBuffer() จะทำการเปิดปิดลักษณะการเรนเดอร์ว่าให้เรนเดอร์ภาพแบบมีความลึกหรือไม่ ส่วนเมธอด EnableAlphaTesting() จะทำการเปิดปิดลักษณะการเรนเดอร์ภาพให้มีลักษณะโปร่งใส (Tranparent) หรือไม่

6.2 คลาส cTexture

คลาสนี้จะทำหน้าที่ในการเก็บ Texture รวมถึงรายละเอียดต่างๆ ของ Texture เช่น ความกว้าง ความสูง เป็นต้น ซึ่งอินสแตนซ์ของคลาสนี้ จะใช้แทน 1 Texture



รูปที่ 6-2 คลาสไดอะแกรมของ cTexture

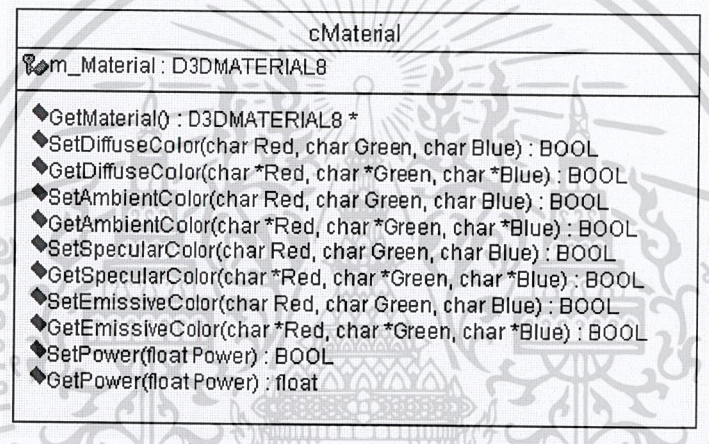
การใช้งานคลาสนี้จะมีอยู่ 2 วิธีคือถ้าต้องการโหลดภาพ Texture จากไฟล์จะใช้เมธอด Load() ซึ่งจะทำการโหลดไฟล์ขึ้นมาเป็น Texture ส่วนอีกวิธีคือ ถ้ามีการโหลด Texture ขึ้นมาแล้ว ซึ่งเก็บอยู่ในอินเตอร์เฟสของ Direct3DTexture8 ก็จะใช้เมธอด Create() ในการสร้าง Texture

คลาสนี้ยังสามารถที่จะวาด Texture ลงในส่วนแสดงผลได้โดยใช้เมธอด Blit() ซึ่งสามารถที่จะย่อขยาย และวาดยังตำแหน่งใดก็ได้ โดยกำหนดค่าลงในพารามิเตอร์ของเมธอดนี้ นอกจากนั้นยังสามารถทำให้วาดแบบโปร่งใสได้ โดยกำหนดสีที่จะเป็นสีที่เป็นสีโปร่งใส

โดยส่วนใหญ่การใช้งานคลาสนี้จะใช้ร่วมกับคลาส cGraphics ในการแปะภาพ Texture ลงบนส่วนของโพลีกอน โดยใช้เมธอด SetTexture() ของคลาส cGraphics ทำให้โพลีกอนที่ได้ดูเหมือนจริงยิ่งขึ้น

6.3 คลาส cMaterial

คลาสนี้จะทำการเปลี่ยนสีที่ปรากฏอยู่บนพื้นผิวของการเรนเดอร์อีอบเจ็กต์ ซึ่งจะทำให้อีอบเจ็กต์ที่ถูกเรนเดอร์นั้นมีลักษณะสมจริงยิ่งขึ้น



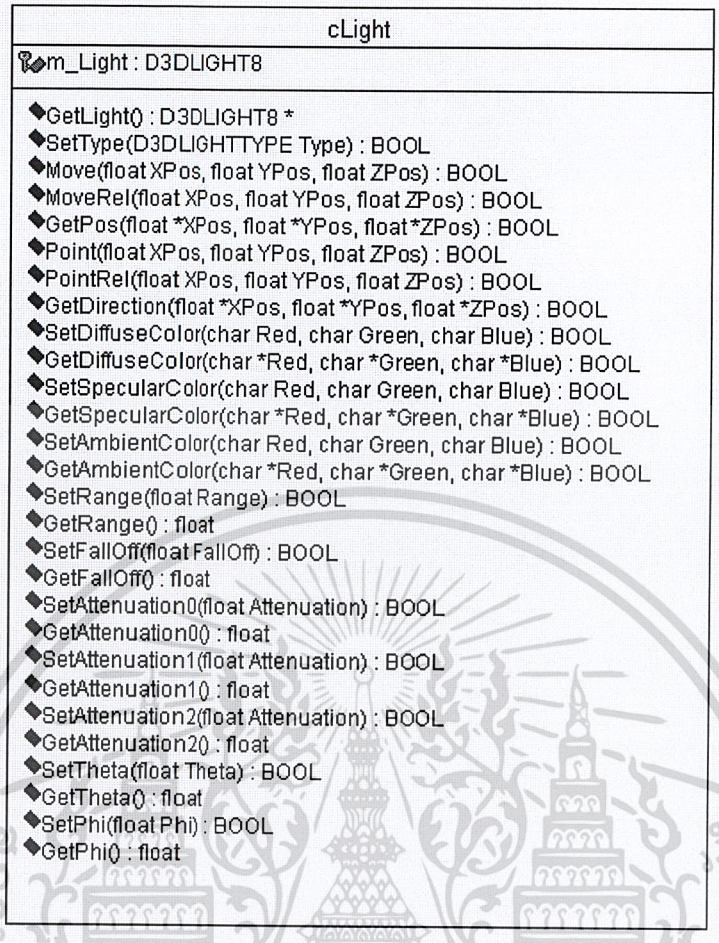
รูปที่ 6-3 คลาสไดอะแกรมของ cMaterial

คลาสนี้เพียงอินสแตนซ์เดียวจะเก็บได้เพียงโครงสร้างของ D3DMATERIAL เดียวเท่านั้น และจะมีเมธอดที่ใช้งานในการกำหนดลักษณะของสีของพื้นผิวที่จะเปลี่ยนไป โดยค่าของแต่ละสีจะมีค่าอยู่ระหว่าง 0 ถึง 255

คลาสนี้ไม่ค่อยมีการใช้งานมากนัก เนื่องจากจะใช้คลาส cTexture แทน เพราะการใช้ภาพแปะลงบนพื้นผิวของวัตถุจะดูสมจริงมากกว่าใช้สีวาดลงบนพื้นผิวของวัตถุ ซึ่งจะดูไม่สมจริงเท่า

6.4 คลาส cLight

คลาสนี้จะใช้สำหรับสร้างแสง สำหรับการเรนเดอร์ภาพให้ดูเหมือนจริงยิ่งขึ้น ซึ่งจะมีลักษณะของการสร้างแสงอยู่หลายลักษณะ



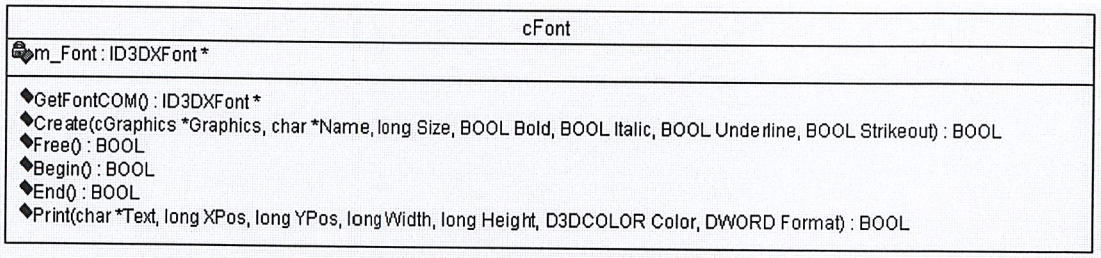
รูปที่ 6-4 คลาสไดอะแกรมของ cLight

ในการใช้งานคลาสนี้ จะใช้เมธอด SetType() ในการกำหนดรูปแบบของแสงที่ต้องการ ซึ่งสามารถกำหนดลักษณะออกเป็น 3 แบบ คือเป็นแบบจุด แบบกระจายออก และแบบทิศทาง และทำการกำหนดคุณสมบัติต่างๆ ของแสง เช่น สี รัศมีของแสง ความเข้ม โดยใช้เมธอดที่มีอยู่

คลาสนี้จะมีการใช้งานร่วมกับคลาส cGraphics ในการกำหนดให้มีการเรนเดอร์ลักษณะของแสงตามที่ได้กำหนดในคลาส cLight โดยใช้เมธอด SetLight() ของคลาส cGraphics

6.5 คลาส cFont

คลาสนี้จะใช้ทำการแสดงผลข้อความลงบนหน้าจอ โดยทำการเรนเดอร์ข้อความลงบน BackBuffer ก่อนที่จะมีการแสดงผล

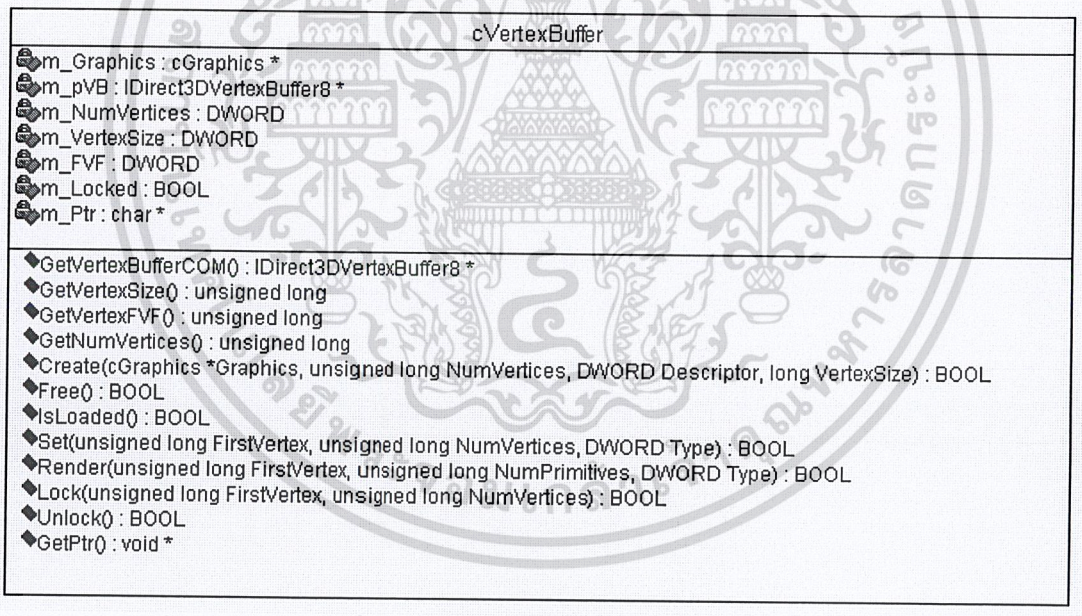


รูปที่ 6-5 คลาสไดอะแกรมของ cFont

ในการใช้งานคลาสนี้ จะทำโดยเรียกเมธอด Create() โดยกำหนดรูปแบบของตัวอักษรที่ต้องการ ขนาดของตัวอักษร และกำหนดลักษณะอื่นๆ เช่น ตัวหนา ตัวเอียง เป็นต้น และจะทำการเรนเดอร์ตัวอักษร โดยใช้เมธอด Print() โดยกำหนดข้อความที่ต้องการและตำแหน่งของข้อความที่ต้องการแสดงผล นอกจากนั้นยังกำหนดสีของข้อความที่ต้องการแสดงได้ด้วย

6.6 คลาส cVertexBuffer

คลาสนี้จะใช้ทำการสร้างเซตของจุด และสามารถเรนเดอร์ออกมาเป็นรูปต่างๆ ได้



รูปที่ 6-6 คลาสไดอะแกรมของ cVertexBuffer

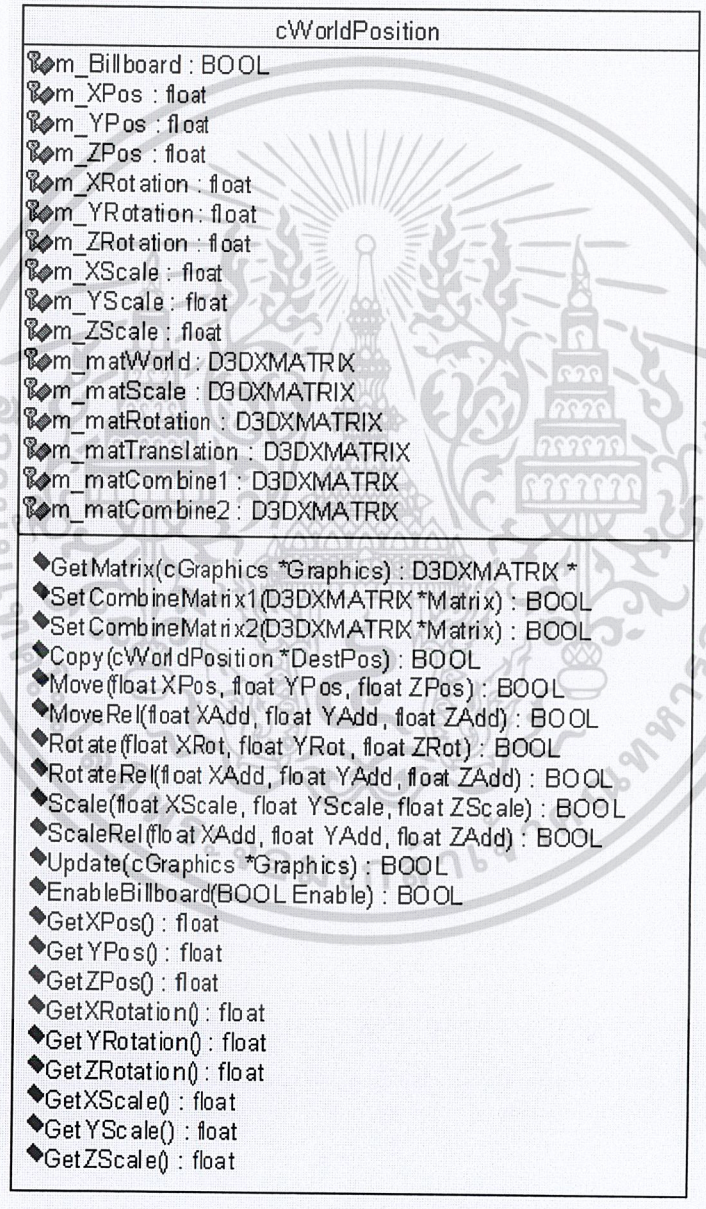
ในการใช้งาน จะทำการเรียกเมธอด Create() เป็นเมธอดแรก เพื่อทำการสร้างเวอร์เท็กซ์บัฟเฟอร์ ซึ่งเป็นหน่วยความจำที่ใช้เก็บลักษณะของเซตของจุด และเมื่อเลิกการใช้งานจะทำการคืนทรัพยากรให้กับระบบโดยใช้เมธอด Free()

เมื่อทำการสร้างบัฟเฟอร์ ก็จะทำการนำข้อมูลของจุดไปเก็บไว้ในบัฟเฟอร์ โดยใช้เมธอด Set() ในการกำหนดลักษณะของเซตของจุด จากนั้นจะทำการเรนเดอร์เซตของจุดโดยใช้เมธอด Render() ซึ่งจะ

ทำการวาดเซตของจุดให้เป็นโพลีกอน ซึ่งจะมีลักษณะของการวาดอยู่ 6 ลักษณะ คือ วาดแบบเป็นจุด วาดต่อกันเป็นเส้น วาดแต่ละจุดเชื่อมกันกับจุดก่อนหน้า วาดสามเหลี่ยมด้วยจุด 3 จุด วาดสามเหลี่ยมโดยใช้ 2 จุดก่อนหน้า และวาดสามเหลี่ยมโดยใช้จุดศูนย์กลางร่วมกัน

6.7 คลาส cWorldPosition

คลาสนี้ทำหน้าที่กำหนดตำแหน่งต่างๆ ของโพลีกอนในพิกัด 3 มิติ ซึ่งสามารถเปลี่ยนพิกัดตำแหน่งของโพลีกอนให้กลายเป็นพิกัดของโลก 3 มิติ



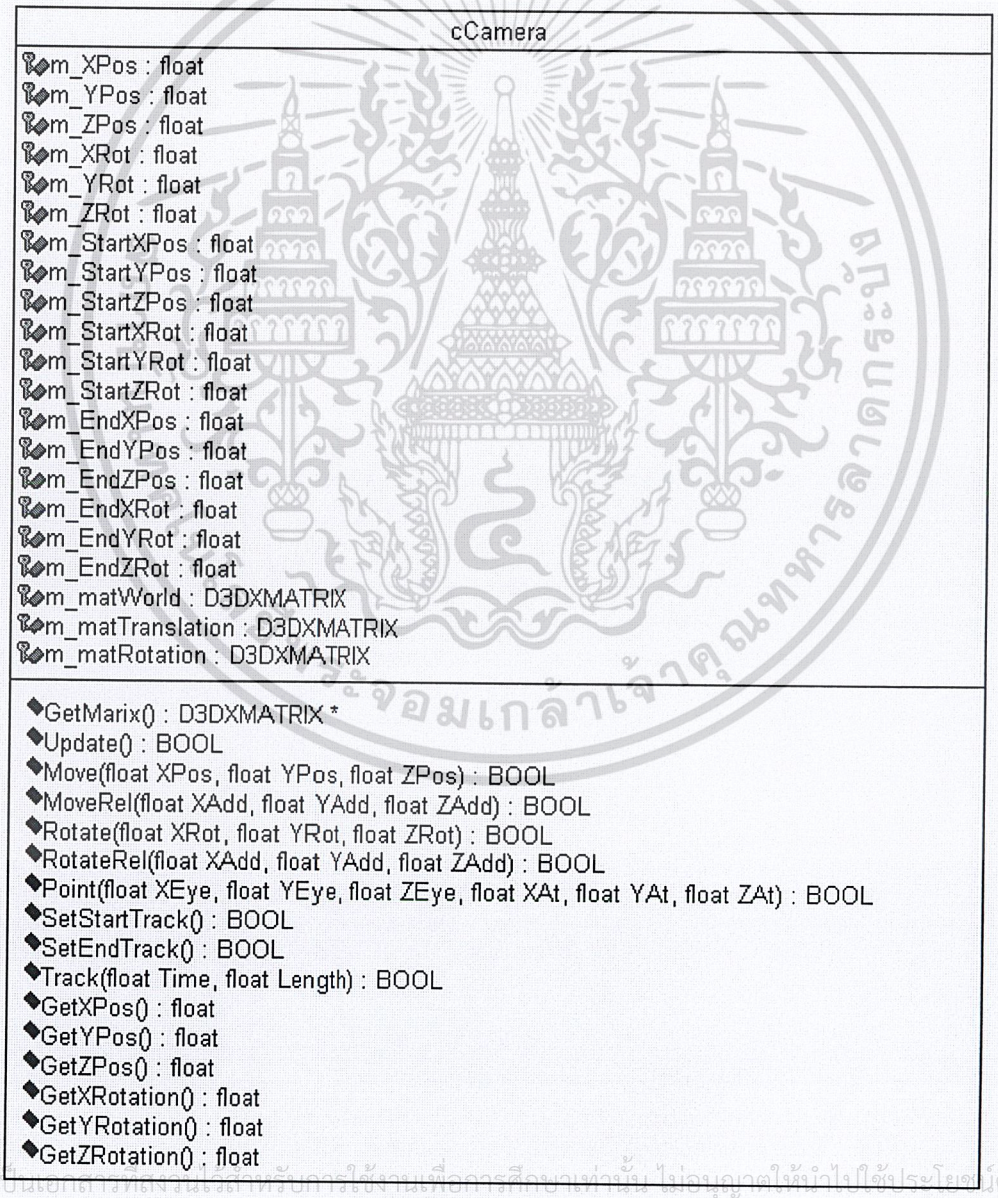
รูปที่ 6-7 คลาสไดอะแกรมของ cWorldPosition

คลาสนี้จะใช้งานโดยการใช้เมธอด Move() เพื่อทำการย้ายตำแหน่งของโพลีกอนไปยังตำแหน่งที่ต้องการในพิกัด 3 มิติ ถ้าต้องการหมุนโพลีกอนตามแกน X, Y และ Z ก็จะใช้เมธอด Rotate() โดยกำหนดค่าตามแกนที่ต้องการ และถ้าต้องการปรับเปลี่ยนขนาดของโพลีกอน ก็จะใช้เมธอด Scale() ในการปรับขนาด

นอกจากนั้นคลาสนี้ยังสามารถทำ Billboard ได้โดยการใช้เมธอด EnableBillboard() เพื่อให้โพลีกอนที่แสดงผล แสดงผลแบบ Billboard ได้

6.8 คลาส cCamera

คลาสนี้ใช้ในการจัดการเกี่ยวกับกล้อง เช่น การเปลี่ยนตำแหน่ง หรือการหมุนกล้องตามแกนต่างๆ ซึ่งจะทำให้มุมมองของเกมเปลี่ยนตามไปด้วย

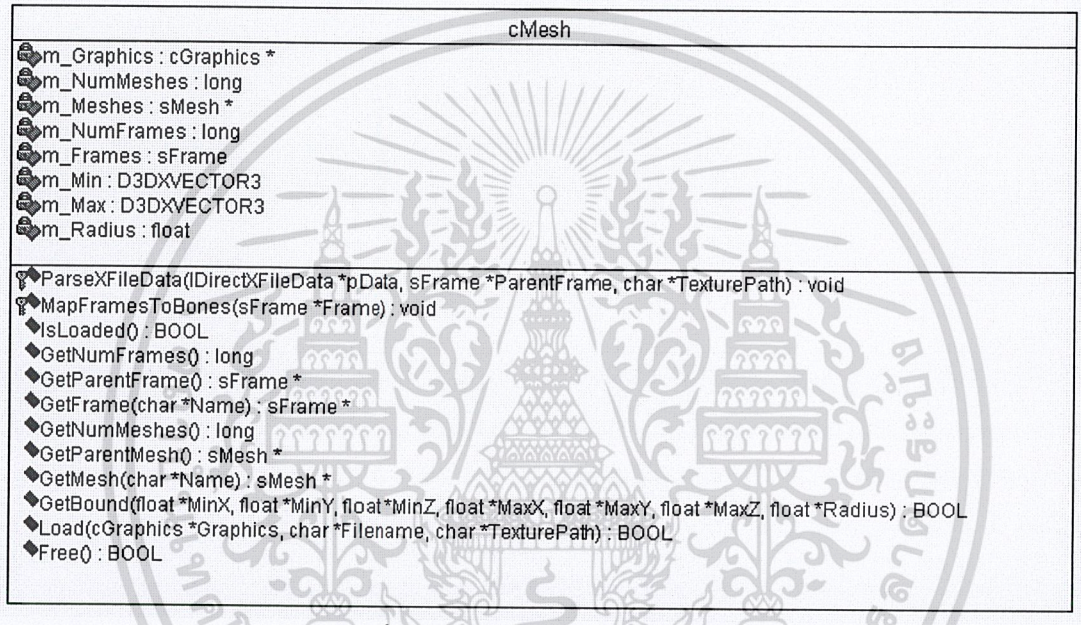


รูปที่ 6-8 คลาสไดอะแกรมของ cCamera

ในการใช้งานคลาสนี้ จะใช้เมธอด Move() เพื่อทำการเคลื่อนย้ายตำแหน่งของกล้องไปยังตำแหน่งที่ต้องการ ใช้เมธอด Rotate() เพื่อทำการหมุนกล้องตามแกนที่ต้องการ นอกจากนั้นยังสามารถกำหนดตำแหน่งกล้องและจุดที่กล้องโฟกัสได้โดยใช้เมธอด Point()

6.9 คลาส cMesh

คลาสนี้ช่วยในการจัดการ โมเดล 3 มิติ ซึ่งจะใช้ตามรูปแบบไฟล์ที่มีนามสกุล X ซึ่งเป็นมาตรฐานของไดเร็กเอ็กซ์



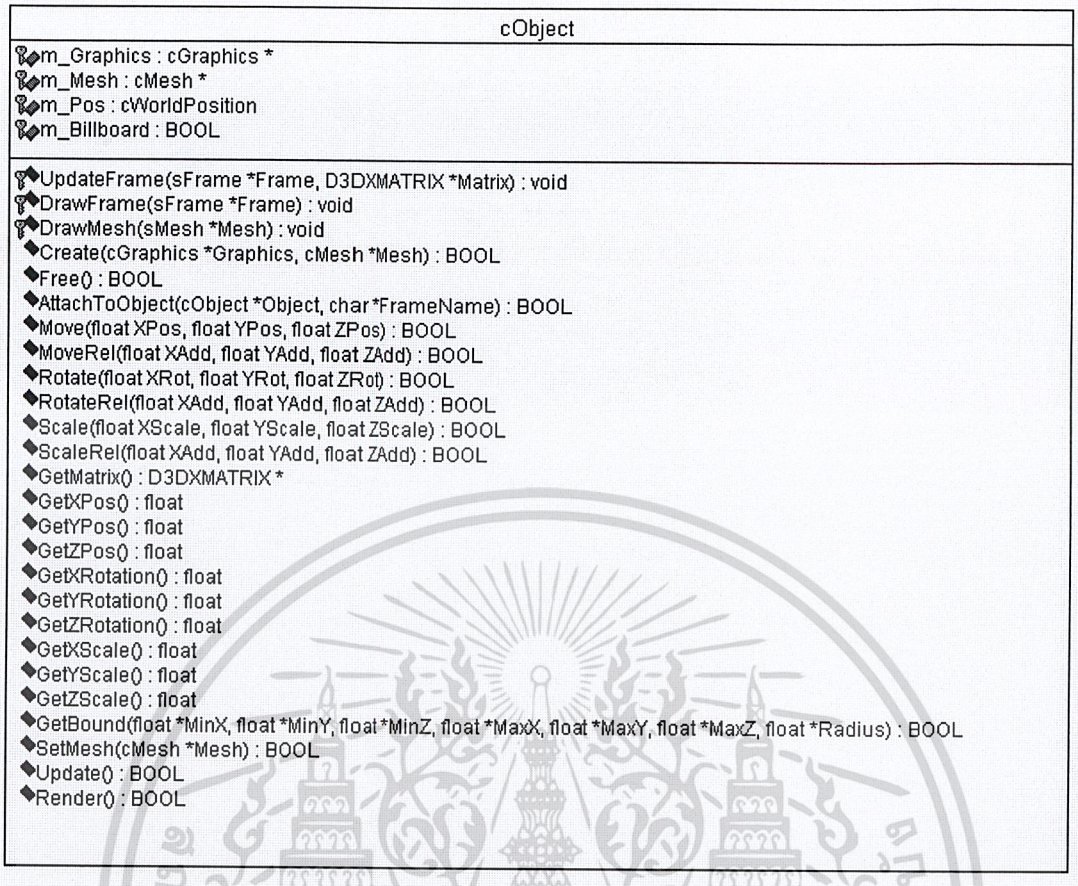
รูปที่ 6-9 คลาสไดอะแกรมของ cMesh

การใช้งานคลาสนี้จะใช้งานโดยเรียกเมธอด Load() ซึ่งจะทำการโหลดข้อมูลของโมเดล เช่น จำนวนเฟรมของโมเดล ชื่อของไฟล์ Texture ที่ใช้ในโมเดล ข้อมูลของจุดที่ใช้ในโมเดล เป็นต้น นอกจากนี้ยังมีเมธอด GetBound() ซึ่งใช้ในการหาขอบเขตของโมเดล ซึ่งส่วนมากจะใช้หาขนาดของโมเดลว่ามีขนาดเท่าใด

การใช้งานคลาส cMesh ส่วนใหญ่จะทำงานร่วมกันกับ cObject เพื่อใช้ในการแสดงโมเดล 3 มิติออกทางส่วนแสดงผล

6.10 คลาส cObject

คลาสนี้จะใช้ในการเรนเดอร์โมเดล 3 มิติให้ออกทางหน้าจอ ซึ่งจะช่วยให้สามารถใช้งานโมเดล 3 มิติได้อย่างมีประสิทธิภาพ และใช้งานหน่วยความจำในการเก็บโมเดลน้อยที่สุด



รูปที่ 6-10 คลาสไดอะแกรมของ cObject

คลาสนี้จะมีการใช้งานโดยเรียกใช้เมธอด Create() โดยใช้คลาส cMesh ในการสร้างอินสแตนซ์ของคลาสนี้ และจะทำการเปลี่ยนตำแหน่งโดยใช้เมธอด Move() ถ้าต้องการหมุนโมเดลตามแกนต่างๆ ก็จะใช้เมธอด Rotate() ส่วนถ้าต้องการปรับขนาดของโมเดล ก็จะใช้เมธอด Scale() เพื่อทำการปรับขนาดของโมเดล และเมื่อต้องการแสดงผลโมเดลออกทางส่วนแสดงผลก็จะใช้เมธอด Render() เพื่อแสดงผลออกทางหน้าจอ

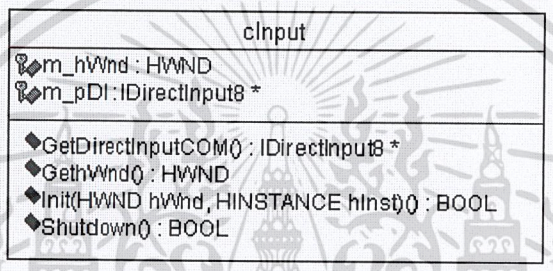
ข้อดีของการคลาสนี้ในการเรนเดอร์ก็คือ สามารถที่จะสร้างคลาส cObject ซึ่งใช้ Mesh เดียวกันได้ ทำให้ไม่เปลืองการใช้หน่วยความจำในการเก็บ Mesh และสามารถที่จะเปลี่ยนตำแหน่งโมเดล หมุนโมเดล หรือปรับขนาดได้โดยไม่กระทบอ็อบเจ็กต์อื่น

บทที่ 7
เอนจินส่วนอินพุต

เอนจินส่วนอินพุตเป็นส่วนติดต่อกับอุปกรณ์อินพุตต่างๆ เช่น คีย์บอร์ด เมาส์ จอยสติ๊ก ซึ่งสามารถที่จะควบคุมการทำงานของอุปกรณ์ต่างๆ โดยจะแบ่งออกเป็นคลาส ดังต่อไปนี้

7.1 คลาส cInput

เป็นคลาสที่จะทำการกำหนดค่าต่างๆ ที่จำเป็นต้องใช้ในการติดต่อกับอุปกรณ์อินพุต และเริ่มต้นติดต่อกับอุปกรณ์อินพุตที่ติดตั้งอยู่

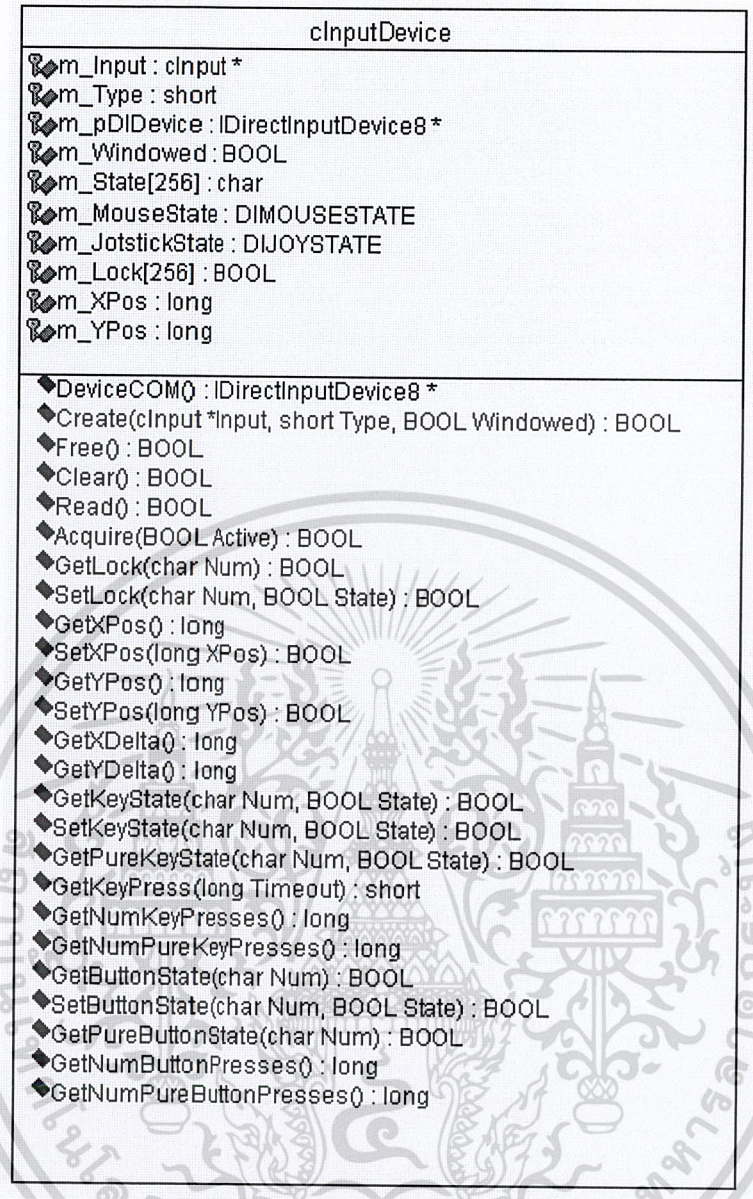


รูปที่ 7-1 คลาสไดอะแกรมของ cInput

เราจะทำการใช้งานโดยเรียกใช้เมธอด Init() ซึ่งจะทำการเริ่มต้นการติดต่อกับอุปกรณ์ต่างๆ โดยจะไปทำการกำหนดค่า และเริ่มต้นการติดต่อกับอุปกรณ์ผ่านอินเตอร์เฟสของ DirectInput และเมื่อต้องการยกเลิกการติดต่อกับอุปกรณ์จะทำการเรียกใช้งานเมธอด Shutdown() เพื่อยกเลิกการติดต่อกับอุปกรณ์อินพุตต่างๆ

7.2 คลาส cInputDevice

เป็นคลาสที่จะจำลองการทำงานของอุปกรณ์อินพุต ซึ่งแต่ละอินสแตนซ์ของคลาสจะแทนอุปกรณ์เพียงอันเดียว ซึ่งถ้าต้องการใช้หลายอุปกรณ์พร้อมกัน จำเป็นต้องสร้างหลายอินสแตนซ์เพื่อทำการใช้งานอุปกรณ์ต่างๆ



รูปที่ 7-2 คลาสไดอะแกรมของ cInputDevice

เราจะใช้งานโดยใช้เมธอด Create() โดยใช้คลาส cInput ที่ทำการกำหนดค่าแล้วมาติดต่อกับอุปกรณ์อินพุต และทำการกำหนดชนิดของอุปกรณ์ที่ต้องการ เช่น คีย์บอร์ด เมาส์ หรือจอยสติ๊ก เป็นต้น และเมื่อต้องการอ่านค่าจากอุปกรณ์อินพุต ก็จะใช้เมธอด Read() เพื่อทำการอ่านค่าสถานะของอุปกรณ์อินพุต ซึ่งจะไปทำการเก็บสถานะของอุปกรณ์ภายในตัวแปร m_State และเมื่อต้องการเลิกใช้อุปกรณ์นั้นก็จะทำการคืนทรัพยากรแก่ระบบโดยใช้เมธอด Free()

ในการใช้งาน เราจะแบ่งเมธอดออกเป็น 2 ส่วนใหญ่ๆ ซึ่งจะแบ่งตามชนิดการทำงานของอุปกรณ์ คือ คีย์บอร์ดกับเมาส์และจอยสติ๊ก ซึ่งส่วนของคีย์บอร์ดจะมีการใช้งานเมธอดหลักๆ คือ GetKeyState() โดยจะทำการอ่านค่าของปุ่มของคีย์บอร์ดที่ถูกกดอยู่ในขณะนั้น และส่วนของเมาส์และจอยสติ๊กนั้นจะมีการใช้เมธอดร่วมกัน เพราะมีการทำงานที่คล้ายกัน ซึ่งจะใช้เมธอดหลักๆ คือ GetXPos(), GetYPos() โดย

จะทำการอ่านค่าตำแหน่งของค่า X และ Y ที่เกิดจากการเลื่อนเมาส์ หรือกดปุ่มที่อยู่บนจอยสติ๊ก นอกจากนั้นยังมีเมธอด GetButtonState() จะทำการอ่านค่าปุ่มที่ถูกกดจากเมาส์หรือจอยสติ๊ก

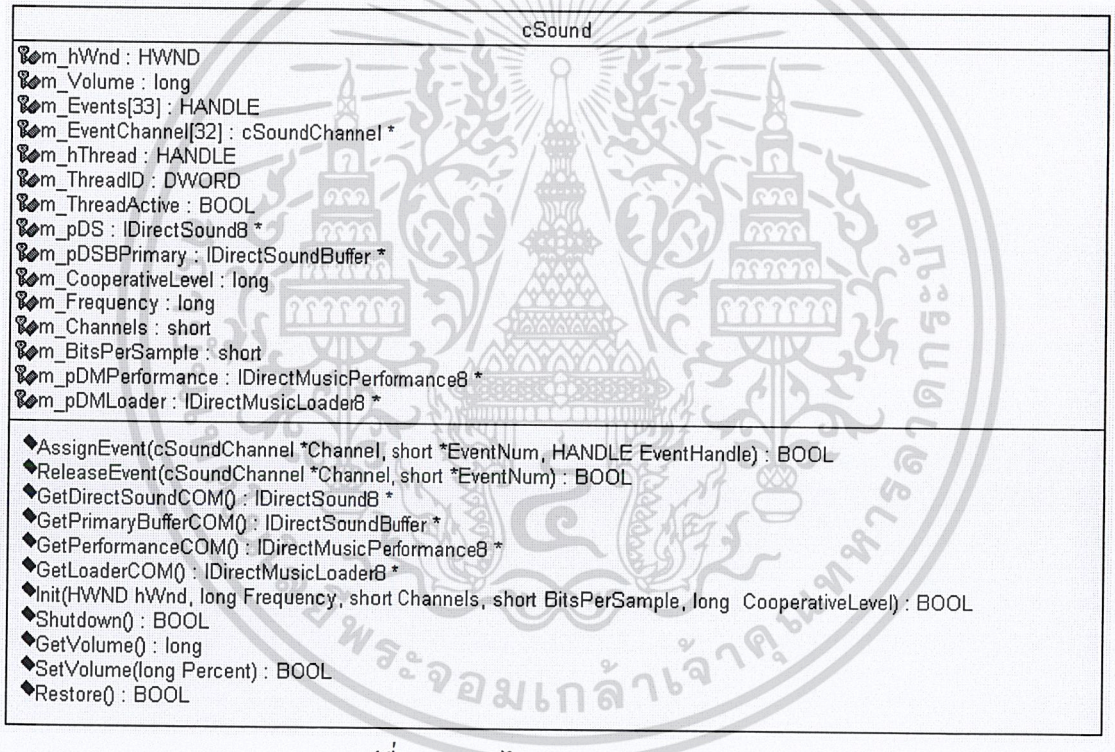
ในการอ่านค่าของปุ่มที่ถูกกด เราจะนำมาเทียบกับค่าคงที่ ซึ่งจะเป็นค่าของปุ่มต่างๆ ซึ่งค่าคงที่ของคีย์บอร์ดจะขึ้นต้นด้วย KEY_ แล้วตามด้วยชื่อของปุ่มที่ต้องการ เช่น ปุ่ม W ก็จะมีค่าคงที่เป็น KEY_W หรือถ้าต้องการปุ่ม ESC ก็จะมีค่าคงที่เป็น KEY_ESC ส่วนค่าคงที่ของเมาส์ จะขึ้นต้นด้วย MOUSE_ แล้วตามด้วยปุ่มของเมาส์ที่ต้องการ เช่น เมื่อกดเมาส์ปุ่มซ้าย จะได้ค่าคงที่เป็น MOUSE_LBUTTON โดยเราจะนำค่าคงที่เหล่านี้ส่งเข้าไปยังเมธอด เพื่อทำการเปรียบเทียบ จากนั้นเมธอดจะทำการคืนค่าผลลัพธ์ที่ได้ออกมาเป็น Boolean คือ TRUE เมื่อค่าคงที่ที่ส่งไปเป็นค่าเดียวกับปุ่มที่กด และจะเป็น FALSE เมื่อไม่ได้กดปุ่มตรงกับค่าคงที่ที่ส่งเข้าไป

บทที่ 8
เอนจินส่วนเสียง

เป็นส่วนที่ใช้ควบคุมและจัดการกับเสียงที่เราจะใส่เข้าไปในเกมของเรา โดยสามารถใส่เสียงเอฟเฟ็คต์เข้าร่วมได้ ซึ่งประกอบด้วยคลาส ดังนี้

8.1 คลาส cSound

คลาส cSound ทำหน้าที่ควบคุมการทำงานของอ็อบเจ็กต์ของ DirectSound และ DirectMusic และสามารถกำหนดระดับความดังของเสียง ซึ่ง cSound จะทำการสร้างอ็อบเจ็กต์ของ DirectSound ขึ้นมาให้โดยอัตโนมัติ

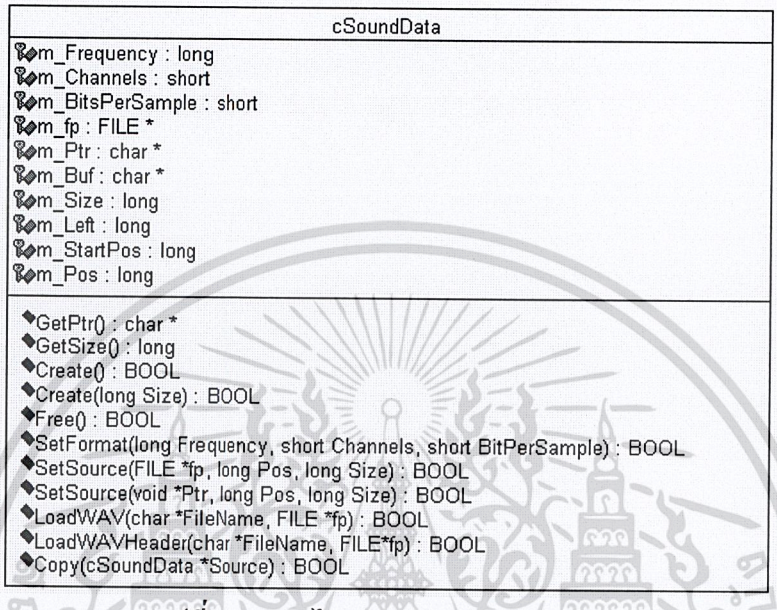


รูปที่ 8-1 คลาสไดอะแกรมของ cSound

ในการใช้งานจะมีเมธอดหลักในการใช้งานอยู่ 3 เมธอดด้วยกัน โดยจะเริ่มด้วยการเรียกเมธอด Init() เพื่อทำการกำหนดค่าเริ่มต้นให้กับเสียงที่ต้องการเล่น ซึ่งถ้าไม่กำหนดค่าใดจะมีค่าปกติคือ มีความถี่ 22,020 Hz ระบบเสียงเป็นแบบ Mono มีอัตราการสุ่มที่ 16 bit เมื่อใดที่ต้องการปรับความดังของเสียงที่เล่นจะใช้เมธอด SetVolume() ในการปรับความดังของเสียง ซึ่งจะใช้หน่วยเป็นเปอร์เซ็นต์แทนความดังของเสียง และมีค่าอยู่ระหว่าง 0 ถึง 100 และเมื่อทำการปิดแอพพลิเคชันต้องมีการเรียกเมธอด Shutdown() เพื่อคืนทรัพยากรที่เอามาจากระบบ

8.2 คลาส cSoundData

คลาสนี้ทำหน้าที่เก็บข้อมูลเสียง เช่น ความถี่เสียง, bit-per-sample, จำนวนของ Channel, ขนาดของเสียง โดยจะมีความสัมพันธ์กับคลาส cSoundChannel ซึ่งจะใช้คลาส cSoundData ในการเล่นเสียง โดยข้อมูลของเสียงจะได้มาจากที่เก็บข้อมูล 2 ที่ คือ ไฟล์และหน่วยความจำ



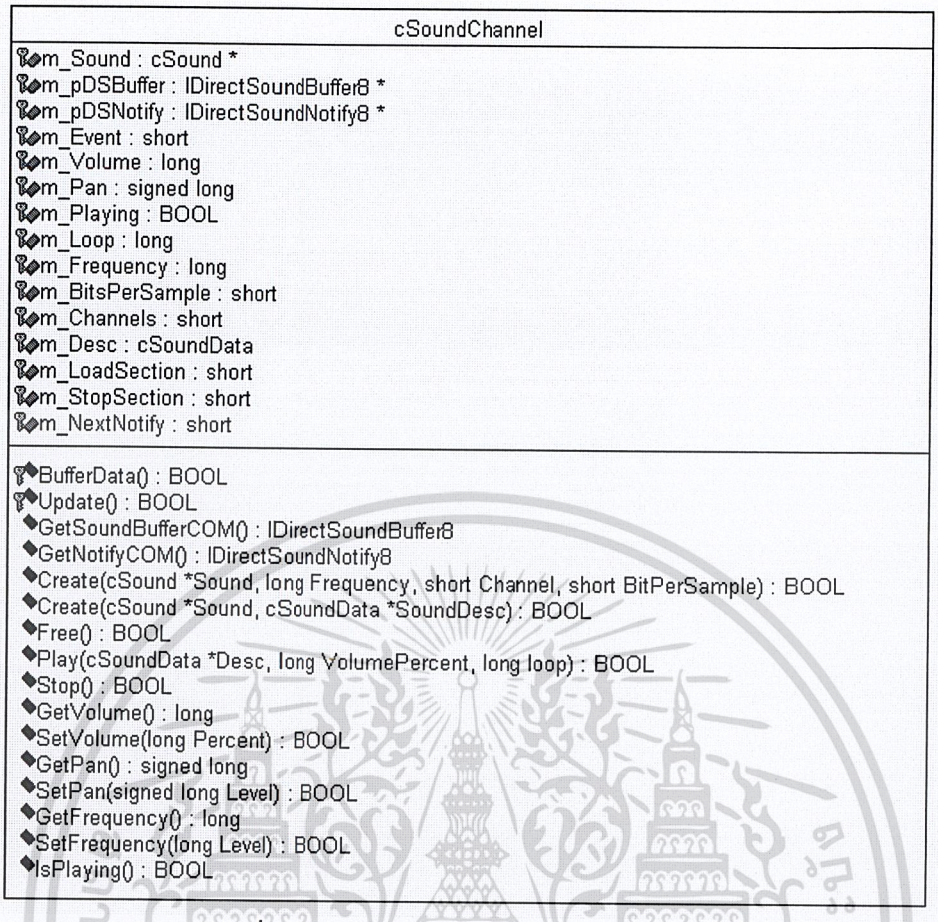
รูปที่ 8-2 คลาสไดอะแกรมของ cSoundData

การใช้งานคลาสนี้จะไม่ค่อยซับซ้อน เพราะว่าคลาสนี้เป็นคลาสที่ทำการโหลดไฟล์เสียงขึ้นมาเก็บไว้เท่านั้น โดยทำงานแค่เพียงกำหนดรูปแบบของเสียงที่ต้องการ ซึ่งจะมีเมธอดที่จะทำการโหลดไฟล์เสียงที่มีนามสกุลเป็น WAV อย่างง่ายคือเมธอด LoadWAV() โดยสามารถโหลดได้ 2 วิธีคือ โหลดเสียงจากไฟล์ที่มีนามสกุล WAV โดยตรง หรือโหลดเสียงผ่านไฟล์พอยเตอร์

ถ้ามีการใช้เสียงที่มาจากที่เก็บข้อมูลแบบหน่วยความจำนั้น จะต้องมีการสร้างบัฟเฟอร์ โดยใช้เมธอด Create() ซึ่งจะกำหนดขนาดของบัฟเฟอร์ที่ต้องการ ซึ่งจะรู้ขนาดของบัฟเฟอร์ได้โดยการเรียกใช้เมธอด LoadWAVHeader() และจะอ้างอิงถึงบัฟเฟอร์ที่สร้างขึ้นมา โดยใช้เมธอด GetPtr()

8.3 คลาส cSoundChannel

คลาสนี้จะทำหน้าที่เล่นไฟล์เสียง ซึ่งเก็บอยู่ในออบเจ็กต์ของคลาส cSoundData โดยสามารถปรับรูปแบบการเล่นได้ตามที่ผู้ใช้ต้องการ



รูปที่ 8-3 คลาสไดอะแกรมของ cSoundChannel

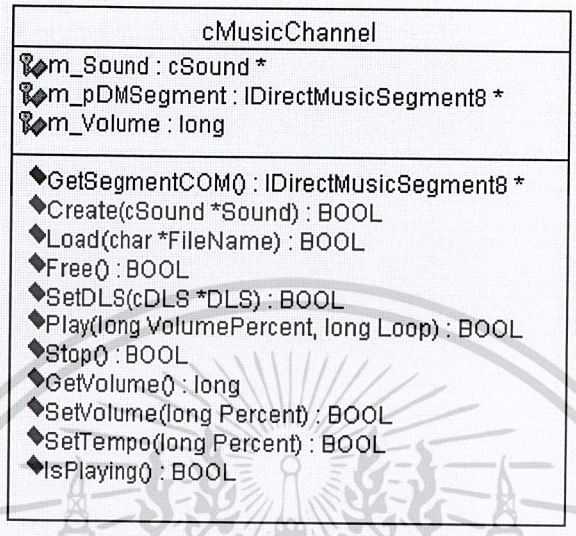
ในการใช้งานคลาส cSoundChannel จะใช้งานร่วมกับคลาส cSoundData ซึ่งใช้ในการเล่นเสียง ซึ่งเราสามารถสร้างอินสแตนซ์ของคลาสนี้ได้สูงสุด 32 อินสแตนซ์ ดังนั้นจึงสามารถสร้างเสียงได้ทั้งหมด 32 Channel ซึ่งสามารถเล่นได้พร้อมๆ กัน การใช้งานจะทำโดยเรียกเมธอด Create() ทำการสร้างรูปแบบของเสียงที่ต้องการเล่น หรือถ้าสร้างอินสแตนซ์ของคลาส cSoundData ไว้แล้วก็สามารถใช้ตามรูปแบบที่กำหนดไว้แล้วในอินสแตนซ์ โดยไม่ต้องกำหนดรูปแบบของเสียงที่จะเล่นใหม่

การทำงานส่วนใหญ่ของคลาสนี้จะเกี่ยวกับการเล่นเสียง ดังนั้นเมธอดหลักที่ใช้จะมีอยู่ 4 เมธอด คือ เมธอด Play() จะทำการเล่นเสียงให้ออกทางลำโพง ซึ่งเมื่อต้องการที่จะหยุดการเล่นก็จะใช้เมธอด Stop() เพื่อทำการหยุด ถ้าต้องการปรับความดังของเสียงที่ออกจะใช้เมธอด SetVolume() ซึ่งจะใช้หน่วยเป็นเปอร์เซ็นต์ โดยมีค่าอยู่ระหว่าง 0 ถึง 100 และถ้าต้องการตรวจสอบว่ากำลังเล่นเสียงอยู่หรือไม่โดยใช้เมธอด IsPlaying() เพื่อทำการตรวจสอบ

นอกจากนั้นยังสามารถปรับให้เสียงที่เล่นออกทางลำโพงข้างใดข้างหนึ่งดังกว่ากันก็ได้ โดยใช้เมธอด SetPan() โดยจะใช้หน่วยเป็นเปอร์เซ็นต์ ซึ่งมีค่าอยู่ระหว่าง –100 ถึง +100 ซึ่งค่าติดลบจะหมายความว่าให้เสียงออกลำโพงทางซ้ายมากกว่าลำโพงทางขวา ส่วนค่าบวกจะหมายความว่าให้เสียงออกลำโพงทางขวามากกว่าลำโพงทางซ้าย

8.4 คลาส cMusicChannel

คลาสนี้จะใช้ทำการเล่นไฟล์เสียงที่มีนามสกุล MID และ SGT ซึ่งเป็นรูปแบบของ DirectMusic native song

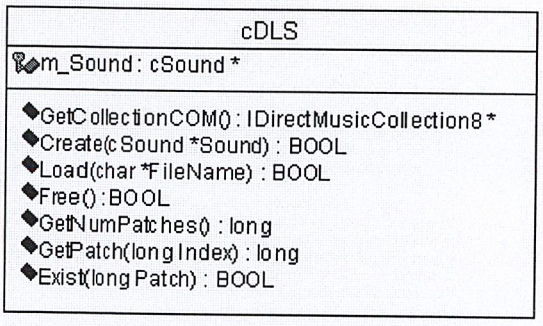


รูปที่ 8-4 คลาสไดอะแกรมของ cMusicChannel

ในการใช้งานคลาสนี้ จะทำการกำหนดค่าเริ่มต้นเพียงครั้งเดียวเท่านั้น โดยใช้เมธอด Create() และเมื่อเราต้องการเล่นไฟล์เสียงใด ก็จะใช้เมธอด Load() โดยจะเล่นได้กับไฟล์เสียงที่มีนามสกุล MID และ SGT เท่านั้น เมื่อต้องการเล่นไฟล์เสียงที่ทำการโหลดแล้วโดยใช้เมธอด Play() โดยกำหนดความดังของเสียงและกำหนดว่าต้องการให้เล่นวนใหม่อีกครั้งหรือไม่เมื่อเล่นจบ เมื่อต้องการยกเลิกการเล่น ก็จะใช้เมธอด Stop() เพื่อทำการหยุดเล่น เมื่อต้องการที่จะเลิกเล่นเสียงก็จะทำการคืนทรัพยากรให้กับระบบโดยใช้เมธอด Free() และถ้าต้องการตรวจสอบว่ามีการเล่นไฟล์เสียงอยู่หรือไม่ ก็จะทำการตรวจสอบโดยใช้เมธอด IsPlaying()

8.5 คลาส cDLS

คลาสนี้จะใช้ในการเก็บ DLS (Downloadable Sounds) ซึ่งจะถูกนำไปใช้โดยคลาส cMusicChannel



รูปที่ 8-5 คลาสไดอะแกรมของ cDLS

ในการใช้งานจะต้องสร้างอินสแตนซ์ของคลาสนี้ขึ้นมาก่อน โดยใช้เมธอด Create() จากนั้นจะทำการโหลดเซตของ DLS ขึ้นมาโดยใช้เมธอด Load() และเมื่อต้องการยกเลิกการใช้งานคลาส cDLS ก็จะทำการเรียกเมธอด Free() เพื่อทำการคืนทรัพยากรให้กับระบบ

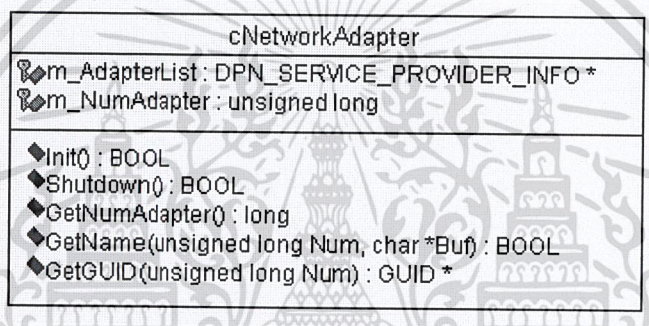
นอกจากนั้นยังมีเมธอดที่ใช้กับ Instrument ของเซตของ DLS โดยจะมีอยู่ทั้งหมด 3 เมธอดหลัก คือ เมธอด GetNumPatch() ซึ่งจะบอกว่ามี Instrument อยู่ภายในเซตของ DLS เท่าใด เมธอด GetPatch() ใช้หาหมายเลข Patch ของ Instrument ที่อยู่ในเซตของ DLS และเมธอดสุดท้ายคือ Exist() ซึ่งจะทำการตรวจสอบหมายเลข Patch ว่ามีอยู่ในเซตของ DLS หรือไม่

บทที่ 9
เอนจินส่วนเน็ตเวิร์ค

เอนจินส่วนเน็ตเวิร์คจะจัดการติดต่อกับการ์ดเน็ตเวิร์ค โดยจะควบคุมกระบวนการทำงานในการติดต่อสื่อสารกันระหว่างเครื่องคอมพิวเตอร์ และจะจัดการการทำงานในส่วนเซิร์ฟเวอร์ และไคลเอนต์ โดยมีคลาสที่เกี่ยวข้องดังนี้

9.1 คลาส cNetworkAdapter

เป็นคลาสที่จะทำการติดต่อกับการ์ดเน็ตเวิร์ค และจัดการกำหนดลักษณะการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์ เพื่อให้สามารถติดต่อสื่อสารกันได้



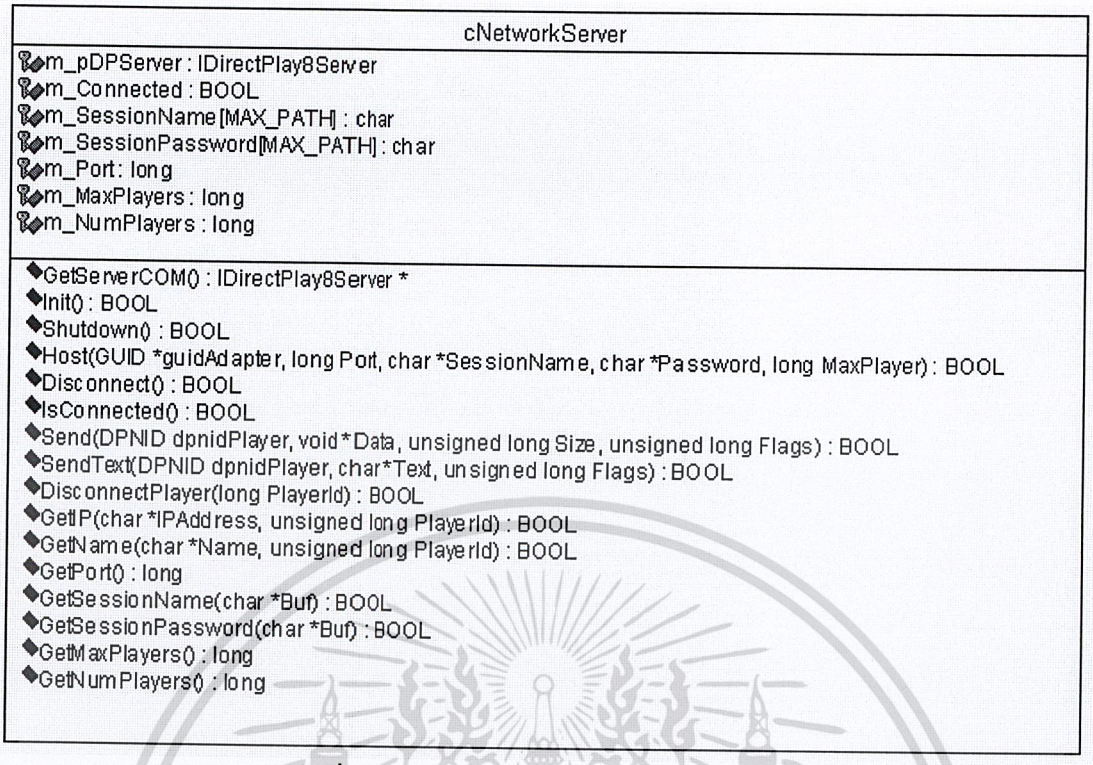
รูปที่ 9-1 คลาสไดอะแกรมของ cNetworkAdapter

คลาสนี้จะทำการติดต่อกับอุปกรณ์เน็ตเวิร์ค โดยผ่านโปรโตคอล TCP/IP ในการใช้งาน การติดต่อกันระหว่างคอมพิวเตอร์นั้น เราจะต้องรู้ GUID โดยเราจะทำการเรียกเมธอด Init() ซึ่งจะทำการกำหนดค่าเริ่มต้นที่จำเป็น และลักษณะการเชื่อมต่อต่างๆ โดยผ่านอินเตอร์เฟสของ DirectPlay ซึ่งเมื่อนำการเรียกเมธอดนี้ ก็จะได้ชื่อของ Adapter และ GUID ซึ่งถ้ามีหลาย Adapter ก็จะทำเก็บจำนวนของ Adapter ที่ตัวแปรชื่อ m_NumAdapter เราจะรู้ GUID ได้โดยใช้เมธอด GetGUID() โดยส่งหมายเลขของ Adapter ที่ต้องการ แล้วจะได้ผลลัพธ์เป็น GUID ของ Adapter ที่ต้องการ และเมื่อใดที่ต้องการเลิกการติดต่อ จะทำการเรียกเมธอด Shutdown() เพื่อทำการยกเลิกการเชื่อมต่อกับระบบเน็ตเวิร์ค

คลาสนี้จะเป็นคลาสหลักที่ใช้เริ่มต้นในการจะทำการติดต่อสื่อสารกันผ่านระบบเน็ตเวิร์ค ซึ่งจำเป็นต้องสร้างขึ้นมาก่อนที่จะทำการกำหนดว่าคอมพิวเตอร์เครื่องใดจะทำหน้าที่เป็นเซิร์ฟเวอร์ หรือไคลเอนต์

9.2 คลาส cNetworkServer

เป็นคลาสที่จะทำหน้าที่เป็นเซิร์ฟเวอร์ ซึ่งจะคอยควบคุมการเชื่อมต่อกันระหว่างไคลเอนต์ ดูแล และจัดการเมสเสจที่เข้าและออก เป็นต้น



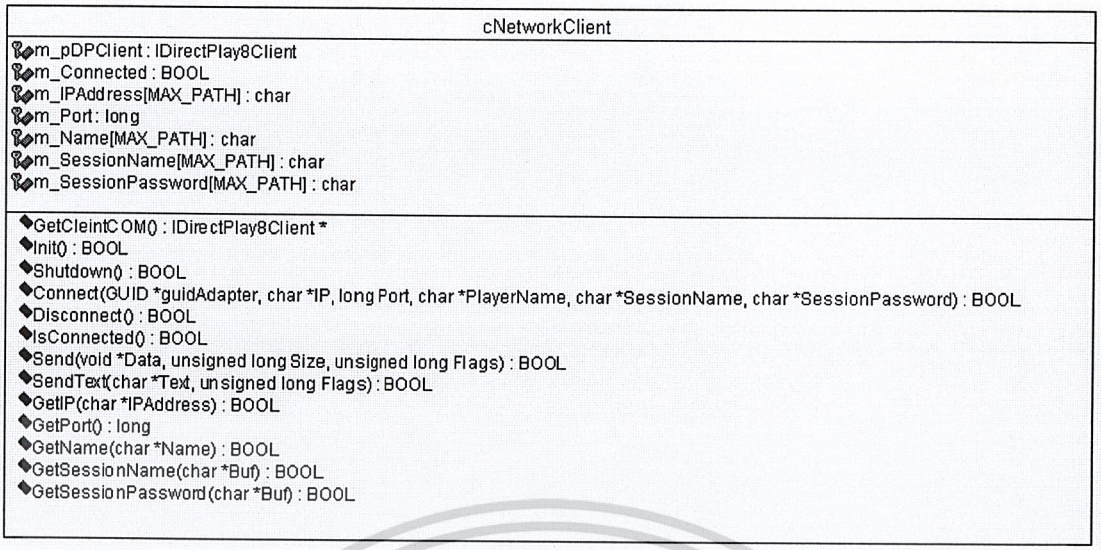
รูปที่ 9-2 คลาสไดอะแกรมของ cNetworkServer

คลาสนี้จะเริ่มใช้งานโดยเรียกเมธอด Init() ซึ่งจะทำการกำหนดค่าเริ่มต้นที่จำเป็นในการใช้งานในการทำงานเป็นเซิร์ฟเวอร์ จากนั้นจะทำการเรียกเมธอด Host() โดยจะส่ง GUID ของ Adapter ที่ต้องการ พอร์ตที่ต้องการใช้ติดต่อ ชื่อและรหัสของ Session ที่ต้องการ ซึ่งเมธอดนี้จะทำการคอยส่งและรับเมสเสจเพื่อทำการประมวลผลเมสเสจที่เข้ามา และทำการทำงานตามลักษณะของเมสเสจที่รับมา เราสามารถตรวจสอบว่ามีการเชื่อมต่ออยู่หรือไม่ โดยใช้เมธอด IsConnected() และเราสามารถส่งเมสเสจไปยังคอมพิวเตอร์เครื่องอื่นได้โดยใช้เมธอด Send() โดยเราจำเป็นต้องรู้ DPNID ของคอมพิวเตอร์ที่จะทำการส่ง ซึ่งเราจะทำรู้ได้จาก Header ของเมสเสจที่รับเข้ามา และเมื่อเราต้องการยกเลิกการทำงานของเซิร์ฟเวอร์ก็จะใช้เมธอด Shutdown() เพื่อยกเลิกการเชื่อมต่อกับคอมพิวเตอร์อื่นๆ

การใช้งานนั้นจะมีการกำหนดจำนวนผู้ใช้ไว้ เพื่อไม่ให้มีผู้ใช้มากเกินที่เซิร์ฟเวอร์จะรับได้ ซึ่งจะช่วยให้ไม่เกิดความล่าช้าในการประมวลผลเมสเสจ และสามารถตอบรับกลับไปยังไคลเอนต์ได้อย่างทันที เรายังสามารถดูจำนวนของผู้ใช้ที่ทำการเชื่อมต่อได้โดยใช้เมธอด GetNumPlayer() ซึ่งจะคืนค่าจำนวนผู้ใช้ที่เชื่อมต่ออยู่ในขณะนั้น

9.3 คลาส cNetworkClient

ทำหน้าที่เป็นไคลเอนต์ ซึ่งจะทำการส่งและรับเมสเสจจากเซิร์ฟเวอร์ โดยจะทำการเชื่อมต่อกับเซิร์ฟเวอร์ เพื่อสื่อสารกับไคลเอนต์อื่นๆ หรือเพื่อต้องการให้เซิร์ฟเวอร์ทำงานให้



รูปที่ 9-3 คลาสไดอะแกรมของ cNetworkClient

คลาสจะทำการเริ่มใช้งาน โดยทำการเรียกเมธอด Init() เหมือนคลาส cNetworkServer และจะเริ่มทำการเชื่อมต่อไปยังเซิร์ฟเวอร์โดยใช้เมธอด Connect() ซึ่งจะต้องบอกหมายเลข IP ของเซิร์ฟเวอร์และบอกหมายเลขพอร์ตที่จะทำการเชื่อมต่อ จากนั้นบอกชื่อของ Session ที่ต้องการเข้าร่วมและถ้า Session นั้นมีรหัสในการเข้าร่วมก็ให้ใส่เข้าไปด้วย จากนั้นก็จะทำการเชื่อมต่อไปยังเซิร์ฟเวอร์นั้น ถ้าต้องการตรวจสอบว่ามีการเชื่อมต่ออยู่หรือไม่ จะใช้เมธอด IsConnected() เพื่อช่วยในการตรวจสอบ และถ้าต้องการยกเลิกการเชื่อมต่อกับเซิร์ฟเวอร์ ก็จะทำการเรียกใช้เมธอด Disconnect()

ในการส่งเมสเสจจะใช้เมธอดอยู่ 2 เมธอดก็คือ ถ้าต้องการส่งข้อมูลที่เป็นรูปแบบตามที่ต้องการ ก็จะใช้เมธอด Send() ซึ่งจำเป็นต้องบอกขนาดของเมสเสจด้วย แต่ถ้าต้องการส่งเพียงข้อความอย่างเดียวจะใช้เมธอด SendText() ซึ่งจะทำให้สะดวกต่อการส่งมากกว่าใช้เมธอด Send()

ในการใช้งานคลาส cNetworkServer และคลาส cNetworkClient นั้น จะต้องมีการสืบทอดคลาสดังกล่าวมาเป็นคลาสใหม่ เนื่องจากจำเป็นต้องมีการประมวลผลจัดการกับเมสเสจตามที่ผู้ใช้ต้องการ ดังนั้นการสืบทอดคลาสนั้นจะต้องทำการสร้างเมธอดใหม่ขึ้นมา เพื่อที่จะจัดการกับรูปแบบของเมสเสจที่ผู้ใช้ต้องการ ซึ่งส่วนใหญ่จะสร้างโดยให้มีชื่อเมธอดเป็น Receive() โดยจะรับเมสเสจมา และทำการตรวจสอบว่าเป็นเมสเสจชนิดใด และทำการประมวลผลเมสเสจ จากนั้นอาจทำการตอบสนองเมสเสจหรือทำการอย่างใดอย่างหนึ่งตามที่ผู้พัฒนาต้องการตามชนิดของเมสเสจนั้น

บทที่ 10
MMORPG

10.1 MMORPG คืออะไร

MMORPG (Massively-Multiplayer Online Role-Playing Game) เป็นเกมประเภทหนึ่ง ที่มีลักษณะเป็นแบบ Role-Playing คือผู้เล่นจะเสมือนจำลองเป็นบุคคลหนึ่งในเกม ซึ่งจะควบคุมบทบาทตัวละครให้เป็นไปตามที่ผู้เล่นต้องการ แต่เนื่องจากเป็นเกมแบบ MMORPG ซึ่งจะเป็นเกมแบบออนไลน์ คือ สามารถเล่นเกมได้พร้อมกันหลายๆ คนในเวลาเดียวกัน โดยผู้เล่นแต่ละคนจะสามารถพบเจอและพูดคุยกันได้ภายในเกม เสมือนเจอกันจริง นอกจากนี้ผู้เล่นยังสามารถที่จะผจญภัย หรือแก้ไขปริศนา ซึ่งจะสามารถเล่นไปพร้อมกับผู้เล่นคนอื่นได้ ทำให้เกมลักษณะนี้ได้รับความนิยมอย่างมาก

10.2 ลักษณะของเกมแบบ MMORPG

เกมแบบ MMORPG มีลักษณะที่เด่นกว่าเกมประเภทอื่นทั่วไป คือ เป็นเกมที่เล่นผ่านอินเตอร์เน็ต และสามารถเล่นพร้อมกันได้หลายคน ดังนั้นเกมแบบ MMORPG จึงมีลักษณะดังต่อไปนี้

10.2.1 มีสถานะที่ถาวร (Persistent Worlds)

การมีสถานะที่ถาวร หมายถึงตัวเกมจะมีการเก็บสถานะต่างๆ ของผู้เล่นไว้ตลอด แม้ว่าผู้เล่นจะออกจากเกมแล้วก็ตาม แต่การดำเนินการเล่นของเกมยังคงมีอยู่ต่อไป เนื่องจากเป็นเกมแบบออนไลน์ ดังนั้นจึงยังมีผู้เล่นคนอื่นที่ยังดำเนินการเล่นอยู่ ซึ่งต่างกับเกมทั่วไปประเภทอื่นๆ ที่เกมจะหยุด เมื่อเราทำการออกจากเกม และจะเก็บข้อมูลไว้

ในการทีสถานะที่ถาวรนั้น จะต้องมีการเก็บสถานะต่างๆ ของผู้เล่นไว้ ซึ่งโดยทั่วไปการเก็บข้อมูลต่างๆ จะเก็บไว้ที่ฝั่งเซิร์ฟเวอร์ เพื่อป้องกันการเปลี่ยนแปลงสถานะต่างๆ ของผู้เล่น ซึ่งอาจจะเกิดความไม่เท่าเทียมกันกับผู้เล่นคนอื่นๆ และข้อดีอีกอย่าง คือผู้เล่นไม่จำเป็นต้องเก็บข้อมูลของผู้เล่นคนอื่นๆ ไว้ ซึ่งถ้ามีผู้เล่นเป็นจำนวนมาก ก็จะใช้พื้นที่ในการเก็บมากด้วยเช่นกัน

การเก็บข้อมูลสถานะของผู้เล่นนั้น จะมีการเก็บได้ 2 ลักษณะ คือ

1. การเก็บข้อมูลลงไฟล์

   เป็นวิธีการเก็บข้อมูลอย่างง่าย ซึ่งผู้ใช้จะต้องทำการเขียนการทำงานส่วนการจัดเก็บ การค้นหา การเพิ่ม การลบข้อมูลเอง ซึ่งถ้าผู้พัฒนาใช้แนวคิดในการจัดการการทำงานในส่วนนี้ไม่ดี ก็จะทำให้การทำงานของเกมช้าได้ ส่งผลให้ประสิทธิภาพของเกมลดลง

นอกจากนั้นไม่สามารถที่จะเก็บความสัมพันธ์กันระหว่างข้อมูลต่างๆ ที่จัดเก็บ ซึ่งผู้ใช้จะต้องทำการจัดการความสัมพันธ์กันของข้อมูลเอง แต่วิธีนี้เป็นวิธีเบื้องต้นในการเก็บข้อมูลสถานะของผู้เล่น

2. การเก็บข้อมูลลงฐานข้อมูล

วิธีการเก็บข้อมูลลงฐานข้อมูลนั้น มีวิธีการทำค่อนข้างยาก เพราะจำเป็นต้องออกแบบตาราง และความสัมพันธ์กันระหว่างข้อมูล แต่วิธีการนี้ค่อนข้างมีประสิทธิภาพ เพราะว่าการจัดการในการเก็บข้อมูล ค้นหาข้อมูล การเพิ่ม การลบ จะถูกจัดการ โดย DBMS (Database Management System) ซึ่งจะช่วยในการทำงานเหล่านี้อย่างมีประสิทธิภาพ ผู้พัฒนาจำเป็นต้องมีความรู้เกี่ยวกับภาษา SQL (Structured Query Language) ซึ่งเป็นภาษาใช้ในการจัดการกับข้อมูลที่เก็บอยู่ในฐานข้อมูล วิธีนี้เป็นวิธีที่ค่อนข้างนิยมกันในการใช้งาน เพราะสะดวก แต่ต้องอาศัยความรู้ในการจัดการฐานข้อมูล

10.2.2 มีข้อมูลตรงกัน (Data Integrity)

เนื่องจากเกมลักษณะแบบ MMORPG นั้น ผู้เล่นสามารถเล่นพร้อมกันได้หลายคน ดังนั้นข้อมูลของผู้เล่นต่างๆ ในขณะที่เล่นอยู่นั้นต้องตรงกัน แต่เพราะมีการเก็บข้อมูลสถานะต่างๆ ของผู้เล่นไว้ที่ฝั่งเซิร์ฟเวอร์ ดังนั้นข้อมูลที่มีอยู่จึงตรงกันกับข้อมูลของผู้เล่นที่มีอยู่จริง เพราะถ้ามีการเก็บข้อมูลไว้ที่ฝั่งไคลเอนต์ ผู้เล่นอาจทำการแก้ไขข้อมูล ทำให้ข้อมูลที่มีอยู่ไม่ตรงตามความเป็นจริง และเกิดความไม่เท่าเทียมกันในการเล่นเกม

10.2.3 การมีสมาคมกันของผู้เล่น (Player Communities)

ข้อดีอย่างหนึ่งของเกมแบบ MMORPG นั้นก็คือ ผู้เล่นสามารถที่จะทำการรวมกลุ่มตั้งขึ้นมาเป็นสมาคม เพื่อทำการเล่นเกมร่วมกัน ช่วยกันกันเล่น หรือช่วยกันแก้ปัญหาต่างๆ ที่มีอยู่ภายในเกม ที่ทำเช่นนี้ได้ก็เพราะว่าผู้เล่นสามารถเล่นเกมพร้อมกันได้หลายคน และผู้เล่นก็จำลองเหมือนเป็นบุคคลในเกม ซึ่งทำให้เกมได้ความนิยมอย่างมาก เพราะผู้เล่นสามารถเข้ามาพูดคุย เข้ามาพบเจอกัน เหมือนอยู่ในโลกความเป็นจริง

10.2.4 การมีความก้าวหน้าของผู้เล่น (Player Advancement)

ผู้เล่นสามารถที่จะพัฒนาความสามารถของผู้เล่นได้เอง จากการเล่นเกม โดยทำการผจญภัย และแก้ไขปัญหาต่างๆ ซึ่งระหว่างการเล่นอาจจะเจอสัตว์ประหลาด หรือตัวละครที่เป็นคอมพิวเตอร์ โดยที่ผู้เล่นจะต้องกำจัด ดังนั้นผู้เล่นจะได้ค่าประสบการณ์ หรืออาจจะมีความสามารถพิเศษของผู้เล่นเพิ่มมากขึ้น ดังนั้นผู้เล่นจะพยายามเล่นเพื่อเพิ่มระดับความสามารถของตัวผู้เล่นเอง ดังนั้นเกมลักษณะนี้จึงเป็นที่นิยมอย่างมาก

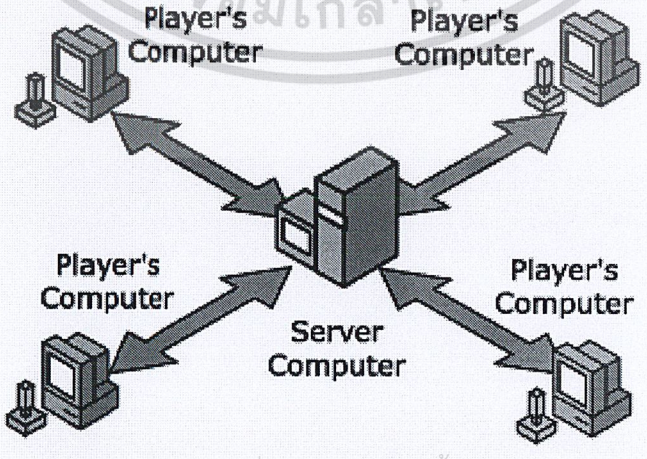
10.2.5 ความรู้สึกเป็นเจ้าของสิ่งของในเกม (Item Owner)

ในการเล่นเกมนั้น ผู้เล่นอาจจะได้รับอุปกรณ์ หรือสิ่งของต่างๆ ที่มีอยู่ในภายในเกม ซึ่งสิ่งของเหล่านี้จะมีผู้เล่นที่เป็นคนเก็บสิ่งของเหล่านี้เป็นเจ้าของ ดังนั้นผู้เล่นจะรู้สึกเหมือนเป็นเจ้าของสิ่งของนั้นจริง ซึ่งผู้เล่นสามารถที่จะกระทำใดๆ กับสิ่งของนั้นได้ เหมือนมีสิ่งของนั้นอยู่จริง เช่น อาจจะให้สิ่งของนั้นแก่ผู้อื่น หรือใช้สิ่งของนั้น หรือทำการขายสิ่งของนั้น

10.2.6 ผู้เล่นดำเนินการเล่นพร้อมกัน (Player Synchronization)

เนื่องจากเกมจะมีผู้เล่นเล่นพร้อมกันหลายๆ คนในเวลาเดียวกัน จึงต้องมีการทำการ Synchronize ระหว่างผู้เล่น โดยที่เซิร์ฟเวอร์จะเป็นผู้จัดการการทำการ Synchronize กับฝั่งของไคลเอนต์ ซึ่งจะทำให้ผู้เล่นเกมเห็นการเปลี่ยนแปลงของเกมเหมือนที่ผู้เล่นคนอื่นๆ เห็น

10.3 สถาปัตยกรรมของเกมแบบ MMORPG



รูปที่ 10-1 รูปแสดงการทำงานแบบไคลเอนต์ – เซิร์ฟเวอร์

สถาปัตยกรรมของเกมแบบ MMORPG จะใช้แบบไคลเอนต์ - เซิร์ฟเวอร์ ในการติดต่อสื่อสารระหว่างเครื่องคอมพิวเตอร์ โดยไคลเอนต์จะทำการติดต่อสื่อสารกับเซิร์ฟเวอร์ และจะใช้เซิร์ฟเวอร์ในการติดต่อสื่อสารผ่านไปยังผู้เล่นคนอื่น ซึ่งไคลเอนต์จะเชื่อมต่อกับเซิร์ฟเวอร์ผ่านระบบอินเตอร์เน็ต โดยผ่านโปรโตคอลที่ใช้สื่อสารแบบ TCP/IP ซึ่งเป็นโปรโตคอลมาตรฐานที่ในใช้สื่อสารผ่านระบบอินเตอร์เน็ต

บทที่ 11
การทำงานส่วนเซิร์ฟเวอร์ และการสื่อสารกับไคลเอนต์

11.1 ขั้นตอนการทำงานส่วนเซิร์ฟเวอร์

ขั้นตอนการทำงานในส่วนเซิร์ฟเวอร์จะแบ่งเป็น 5 ขั้นตอนดังนี้

1. Initialization

   เป็นขั้นตอนที่ทำการกำหนดค่าเริ่มต้นต่างๆ ของเซิร์ฟเวอร์ โดยจะเริ่มตั้งแต่การจัดการฐานข้อมูล, เน็ตเวิร์ค และทำการเริ่มโปรแกรม

2. Wait Packet

   ทำการรอรับชื่อผู้ใช้ และรหัสผ่านจากผู้ใช้ เมื่อผู้เล่นได้ทำการป้อนข้อมูล และทำการกดปุ่มตกลง เพื่อยืนยัน ก็จะมีการส่งแพ็คเกจไปยังเซิร์ฟเวอร์ เพื่อทำการตรวจสอบก่อนเข้าไปเล่นในเกม

3. Process Packet

   ทำการประมวลผลแพ็คเกจต่างๆ ที่ได้มาจากไคลเอนต์ และทำการปรับปรุงข้อมูลต่างๆ ของผู้เล่น ตามแพ็คเกจที่ส่งมา

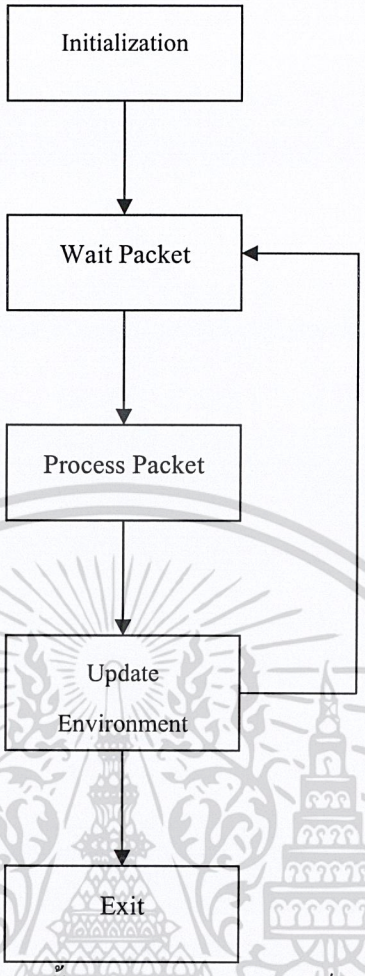
4. Update Environment

   ทำการปรับปรุงข้อมูลต่างๆ แล้วทำการส่งไปยังผู้เล่นคนอื่นทุกคน เพื่อปรับปรุงข้อมูลของผู้เล่นคนอื่นๆ ให้มีข้อมูลตรงกัน (Synchronization) โดยใช้แพ็คเกจหลักๆ ดังนี้

   - Update Player
   - Update Spawn Player
   - Update Monster
   - Update Player Attack
   - Update Monster Attack

5. Exit

   ทำการออกจากโปรแกรม และคืนทรัพยากรต่างๆ ที่ใช้ให้กับระบบ



รูปที่ 11-1 รูปแสดงขั้นตอนการทำงานของเซิร์ฟเวอร์

11.2 แพ็คเกจที่ใช้ในแต่ละขั้นตอน

11.2.1 ขั้นตอนล็อคอิน

มีแพ็คเกจที่ใช้อยู่ 4 แพ็คเกจคือ

Packet Login	
ต้นทาง :	ไคลเอนต์
ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากไคลเอนต์เพื่อส่งข้อมูลของผู้เล่น ซึ่งก็คือชื่อผู้เล่นและรหัสผ่าน
รูปแบบ :	Name - ชื่อผู้เล่น ขนาด 16 ไบท์ Password - รหัสผ่าน ขนาด 16 ไบท์ รวมทั้งหมดขนาด 32 ไบท์

Packet User Invalid	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เพื่อบอกว่าชื่อผู้ใช้ไม่มีในระบบ
รูปแบบ :	ไม่มี

Packet Password Invalid	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เพื่อบอกว่ารหัสผิด
รูปแบบ :	ไม่มี

Packet Player	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์ เมื่อตรวจสอบชื่อผู้เล่นและรหัสผ่านถูกต้อง แล้วจะดึงข้อมูลของผู้เล่นคนนั้นจากฐานข้อมูล และส่งให้แก่ไคลเอนต์
รูปแบบ :	Name - ชื่อตัวละคร ขนาด 16 ไบท์ Mapname - ชื่อแผนที่ที่ตัวละครอยู่ ขนาด 16 ไบท์ ID - เน็ตเวิร์ก ID ของผู้เล่น ขนาด 4 ไบท์ Level - ระดับของตัวละคร ขนาด 4 ไบท์ Exp - ค่าประสบการณ์ของตัวละคร ขนาด 4 ไบท์ Money - เงินที่ตัวละครมีอยู่ ขนาด 4 ไบท์ Head - ลักษณะหัวของตัวละคร ขนาด 4 ไบท์ Body - ลักษณะตัวของตัวละคร ขนาด 4 ไบท์ X - พิกัดแกน x ของตัวละคร ขนาด 4 ไบท์ Z - พิกัดแกน z ของตัวละคร ขนาด 4 ไบท์ HP - พลังชีวิตของตัวละคร ขนาด 4 ไบท์ HPMax - พลังชีวิตสูงสุดของตัวละคร ขนาด 4 ไบท์ MP - พลังเวทย์มนต์ของตัวละคร ขนาด 4 ไบท์ MPMax - พลังเวทย์มนต์สูงสุดของตัวละคร ขนาด 4 ไบท์ STR - ค่าความแข็งแกร่งของตัวละคร ขนาด 4 ไบท์

	AGI - ค่าความคล่องแคล่วจองตัวละคร ขนาด 4 ไบท์ VIT - ค่าความอดทนของตัวละคร ขนาด 4 ไบท์ INT - ค่าความฉลาดของตัวละคร ขนาด 4 ไบท์ DEX - ค่าความแม่นยำของตัวละคร ขนาด 4 ไบท์ LUK - ค่าความโชคดีของตัวละคร ขนาด 4 ไบท์ รวมทั้งหมดขนาด 104 ไบท์

11.2.2 ขั้นตอนโหลดเกม

จะมีแพ็คเกจที่ใช้อยู่ 3 แพ็คเกจคือ

Packet Init Complete	
ต้นทาง :	ไคลเอนต์
ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็คเกจที่ส่งจากไคลเอนต์ไปยังเซิร์ฟเวอร์ เมื่อโหลดตัวละครและฉากเรียบร้อยแล้ว เพื่อให้เซิร์ฟเวอร์รับรู้สถานะ ของเราและจะดำเนินการต่อไป
รูปแบบ :	ไม่มี

Packet Create Monster	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เพื่อส่งข้อมูลของศัตรูทั้งหมดที่อยู่ในฉาก ซึ่งจะมีมากสุดที่ 30 ตัวโดยที่จะมีจำนวนบอกว่ามีทั้งหมดกี่ตัว จากนั้นจะเป็นข้อมูลของแต่ละตัวจำนวน 30 ตัว
รูปแบบ :	Num - จำนวนของศัตรูที่อยู่ในแผนที่นี้ ขนาด 4 ไบท์ ใน 1 แพ็กเกจ จะมีข้อมูลดังนี้ ID - หมายเลขของศัตรู โดยศัตรูแต่ละตัวจะมีหมายเลขไม่ซ้ำกัน ขนาด 4 ไบท์ X - พิกัดในแกน x ที่ศัตรูอยู่ ขนาด 4 ไบท์ Z - พิกัดในแกน z ที่ศัตรูอยู่ ขนาด 4 ไบท์ State - สถานะของศัตรู เช่น เดิน โจมตี เป็นต้น ขนาด 4 ไบท์ Type - ประเภทของศัตรู ขนาด 4 ไบท์ รวมแพ็กเกจศัตรู 1 ตัว 20 ไบท์ 30 ตัว = 600 ไบท์ รวมทั้งหมด ขนาด 600+4 = 604 ไบท์

Packet Create Monster Complete	
ต้นทาง :	ไคลเอนต์
ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากไคลเอนต์ไปยังเซิร์ฟเวอร์ เมื่อโหลดตัวศัตรูเรียบร้อยแล้ว เพื่อให้เซิร์ฟเวอร์รับรู้สถานะของเรา และจะดำเนินการต่อไป
รูปแบบ :	ไม่มี

11.2.3 ขั้นตอนเหตุการณ์ต่างๆ

จะแบ่งออกเป็น 2 ฝั่งคือ ฝั่งไคลเอนต์จะมีอยู่ 4 แพ็คเกจ และฝั่งเซิร์ฟเวอร์จะมี 10 แพ็คเกจ

11.2.3.1 ฝั่งไคลเอนต์

Packet Request Move	
ต้นทาง :	ไคลเอนต์
ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากไคลเอนต์ เพื่อส่งข้อมูลตำแหน่ง X, Z ที่ผู้เล่นได้ทำการเดินไป
รูปแบบ :	X - ขนาด 4 ไบท์ พิกัด X ที่ต้องการจะเดินไป Z - ขนาด 4 ไบท์ พิกัด Z ที่ต้องการจะเดินไป รวมทั้งหมดขนาด 8 ไบท์

Packet Chat	
ต้นทาง :	ไคลเอนต์
ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากไคลเอนต์ เมื่อผู้เล่นทำการพิมพ์ในกล่อง Chat Message
รูปแบบ :	Chat - ข้อความที่ต้องการจะพูดคุย ขนาด 128 ไบท์ รวมทั้งหมดขนาด 128ไบท์

Packet Attack	
ต้นทาง :	ไคลเอนต์

ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็กเกจ ที่ส่งจากไคลเอนต์ เมื่อผู้เล่นทำการคลิ๊กที่ศัตรู เพื่อจะโจมตี
รูปแบบ :	ID - หมายเลขของศัตรูที่ต้องการจะโจมตี ขนาด 4 ไบท์ Mode - โหมดในการต่อสู้ มีด้วยกัน 2 โหมดคือ โจมตี 1 ครั้ง หรือ โจมตีต่อเนื่อง ขนาด 4 ไบท์ รวมทั้งหมดขนาด 8 ไบท์

Packet Request Name	
ต้นทาง :	ไคลเอนต์
ปลายทาง :	เซิร์ฟเวอร์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากไคลเอนต์ เมื่อผู้เล่นเลื่อนเมาส์ไปยังผู้เล่นคนอื่นที่ยังไม่รู้จักชื่อ
รูปแบบ :	ID - หมายเลขของผู้เล่นที่ต้องการรู้ชื่อ ขนาด 4 ไบท์ รวมทั้งหมดขนาด 4ไบท์

11.2.3.2 ฝั่งเซิร์ฟเวอร์

Packet Chat Broadcast	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์ ให้ผู้เล่นทุกคนทราบว่าขณะนี้มีใครที่กำลังพูดอยู่
รูปแบบ :	Chat - ข้อความที่ผู้เล่นคนนั้นพูด ขนาด 128 ไบท์ Name - ชื่อของผู้เล่นที่ทำการพูด ขนาด 16 ไบท์ ID - หมายเลขของผู้เล่นที่ทำการพูด ขนาด 4 ไบท์ รวมทั้งหมดขนาด 148ไบท์

Packet Name	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์ ไปยังผู้ที่ทำการร้องขอว่าหมายเลขนั้นชื่ออะไร

รูปแบบ :	Name - ชื่อของผู้เล่น ที่ id นั้น ขนาด 16 ไบท์ ID - หมายเลขของผู้เล่น ขนาด 4 ไบท์ รวมทั้งหมดขนาด 20ไบท์

Packet Player Died

ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์ เพื่อบอกว่าผู้เล่นคนนั้นได้ตายแล้ว
รูปแบบ :	ID - หมายเลขของผู้เล่นที่ตาย ขนาด 4 ไบท์ รวมทั้งหมดขนาด 4ไบท์

Packet Remove Spawn Player

ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์ เพื่อบอกว่าผู้เล่นคนนั้น ได้ออกจากเกม
รูปแบบ :	ID - หมายเลขของผู้เล่นที่ออกจากเกม ขนาด 4 ไบท์ รวมทั้งหมดขนาด 4ไบท์

Packet Remove Monster

ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์ เพื่อบอกว่าศัตรูตัวนั้นได้ตายแล้ว
รูปแบบ :	ID - หมายเลขของศัตรูที่ตาย ขนาด 4 ไบท์ รวมทั้งหมดขนาด 4ไบท์

Packet Update Player

ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เรื่อยๆ เพื่อบอกถึงสถานะตัวละครของผู้เล่น
รูปแบบ :	HP - พลังชีวิตปัจจุบันของตัวละครเรา ขนาด 4 ไบท์ MP - พลังเวทย์มนต์ปัจจุบันของตัวละครเราขนาด 4 ไบท์ Money - เงินปัจจุบันของตัวละครเรา ขนาด 4 ไบท์

	Exp - ค่าประสบการณ์ปัจจุบันของตัวละครเรา ขนาด 4 ไบท์ X - พิกัดแกน X ปัจจุบันของตัวละครเรา ขนาด 4 ไบท์ Z - พิกัดแกน Z ปัจจุบันของตัวละครเราขนาด 4 ไบท์ State - สถานะของตัวละครเรา ขนาด 4 ไบท์ รวมทั้งหมดขนาด 28ไบท์

Packet Update Spawn Player	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เรื่อยๆ เพื่อบอกถึงสถานะตัวละครผู้เล่นคนอื่นๆ ที่ไม่ใช่ตัวเรา
รูปแบบ :	ID - หมายเลขของผู้เล่น ขนาด 4 ไบท์ State - สถานะของผู้เล่นคนนั้น ขนาด 4 ไบท์ X - พิกัดแกน X ปัจจุบันของผู้เล่นคนนั้น ขนาด 4 ไบท์ Z - พิกัดแกน Z ปัจจุบันของผู้เล่นคนนั้น ขนาด 4 ไบท์ รวมทั้งหมดขนาด 16ไบท์

Packet Update Monster	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เรื่อยๆ เพื่อบอกถึงสถานะศัตรู
รูปแบบ :	ID - หมายเลขของศัตรู ขนาด 4 ไบท์ State - สถานะของศัตรูตัวนั้น ขนาด 4 ไบท์ X - พิกัดแกน X ปัจจุบันของศัตรูตัวนั้น ขนาด 4 ไบท์ Z - พิกัดแกน Z ปัจจุบันของศัตรูตัวนั้น ขนาด 4 ไบท์ รวมทั้งหมดขนาด 16ไบท์

Packet Update Player Attack	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เรื่อยๆ เมื่อผู้เล่นทำการจมตีศัตรูอยู่ ให้รู้ว่าผู้เล่นคนนั้นจมตีใส่ศัตรูตัวไหนและสร้างความเสียหายแก่ศัตรูตัวนั้นเท่าไหร่ โดยจะส่งไปให้ผู้เล่นทุกคนรับทราบ

รูปแบบ :	ID_Attacker - หมายเลขของผู้เล่นที่ทำการโจมตี ขนาด 4 ไบท์ ID_Defender - หมายเลขของศัตรูที่ถูกโจมตี ขนาด 4 ไบท์ Damage - ค่าความเสียดายที่ได้รับ ขนาด 4 ไบท์ รวมทั้งหมดขนาด 12ไบท์

Packet Update Monster Attack	
ต้นทาง :	เซิร์ฟเวอร์
ปลายทาง :	ไคลเอนต์
รายละเอียด :	เป็นแพ็กเกจที่ส่งจากเซิร์ฟเวอร์เรื่อยๆ เมื่อศัตรูทำการจมตีผู้เล่นอยู่ให้รู้ว่าศัตรูตัวนั้นโจมตีใส่ผู้เล่นคนไหนและสร้างความเสียหายแก่ผู้เล่นคนนั้นนั้นเท่าไหร่ โดยจะส่งไปให้ผู้เล่นทุกคนรับทราบ
รูปแบบ :	ID_Attacker - หมายเลขของศัตรูที่ทำการโจมตี ขนาด 4 ไบท์ ID_Defender - หมายเลขของผู้เล่นที่ถูกโจมตี ขนาด 4 ไบท์ Damage - ค่าความเสียหายที่ได้รับ ขนาด 4 ไบท์ รวมทั้งหมดขนาด 12ไบท์

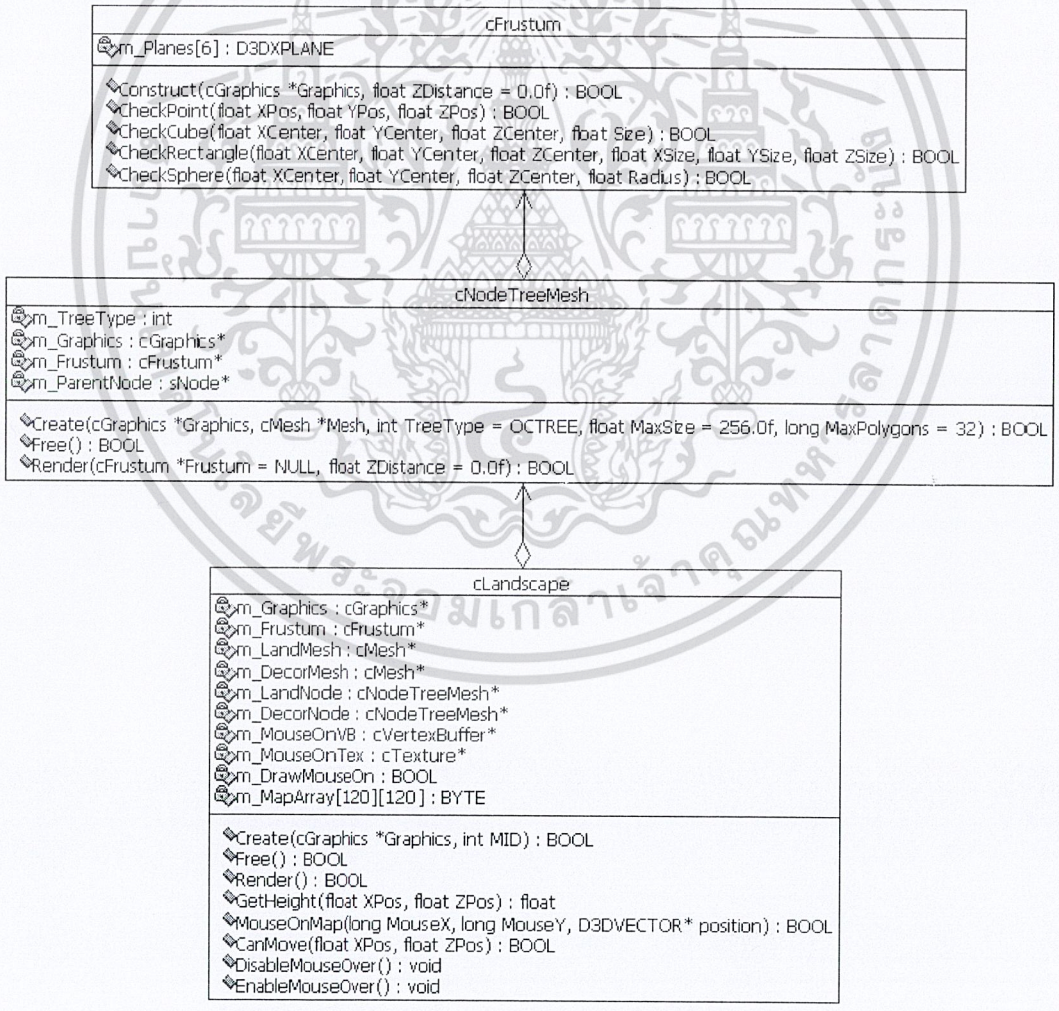
บทที่ 12
การทำงานส่วนไคลเอนต์

12.1 การออกแบบคลาสในส่วนไคลเอนต์

ในส่วนไคลเอนต์นั้น ประกอบด้วยคลาสต่างๆ มากมายหลายคลาส ที่ใช้ในการแสดงผลกราฟิก ซึ่งในที่นี้จะขอยกมาอธิบายเพียงบางคลาสที่สำคัญเท่านั้น

12.2 คลาส cFrustum, cNodeTreeMesh และ cLandscape

เป็นกลุ่มคลาสที่ดูแลเกี่ยวกับการโหลดและแสดงผลโลกจำลองของเกม โดยคลาส cFrustum เป็นการ Implement เทคนิค Viewing Frustum ตามที่ได้กล่าวไปแล้ว จะเห็นว่าภายในคลาสประกอบด้วยระนาบหกระนาบ มีฟังก์ชัน Construct() ที่ใช้ในการคำนวณหา Viewing Frustum และฟังก์ชันอื่นที่ใช้ในการตรวจสอบว่ารูปร่างหรือรูปทรงที่กำหนด มีส่วนหนึ่งส่วนใดอยู่ในฟรัสทัมหรือไม่



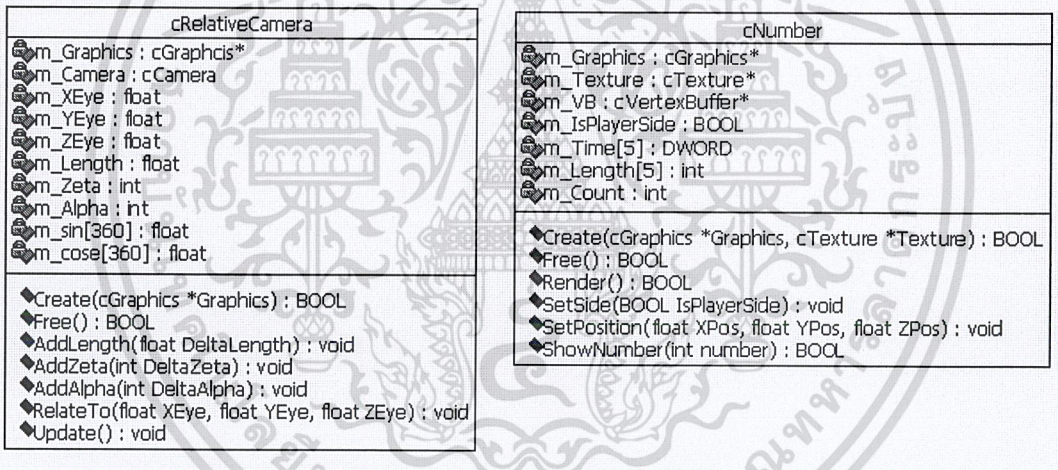
รูปที่ 12-1 คลาสไดอะแกรมของคลาส cFrustum, cNodeTreeMesh และ cLandscape

ส่วน cNodeTreeMesh นั้นเป็นการ Implement เทคนิค Node Tree โดยคำสั่ง Create() จะเป็นการสร้าง Tree จะใช้ครั้งเดียวเท่านั้น ส่วนฟังก์ชัน Render() จะรับฟรัสทัมมาเช็คดูก่อนว่าตรงกับของเดิม (การเรนเดอร์ครั้งที่แล้ว) หรือไม่ ถ้าไม่ตรงจะทำการสแกนโหนด Tree ใหม่ว่า มีโหนดใดอยู่ในฟรัสทัมมากจากนั้นจะทำการเรนเดอร์เฉพาะโหนดเหล่านั้น

ส่วน cLandscape นั้นเป็นตัวแทนโลกจำลอง ซึ่งเราจะแบ่ง Mesh หรือโมเดลออกเป็นสองชุด จึงต้องใช้ cNodeTreeMesh 2 อ็อบเจ็กต์ ซึ่งเป็นตัวพื้นที่ตัวละครเดินได้หนึ่งโมเดล และเป็นส่วนตกแต่งเช่น บ้าน ต้นไม้ อีกหนึ่งโมเดล นอกจากนี้จะมีส่วน Vertex Buffer และ Texture จะรูปที่จะวางบนพื้นผิวขณะที่เมาส์วางอยู่บนนั้นด้วย และสุดท้ายจะมีอาเรย์ที่บอกว่าช่องไหนสามารถเดินได้ ช่องไหนเดินไม่ได้

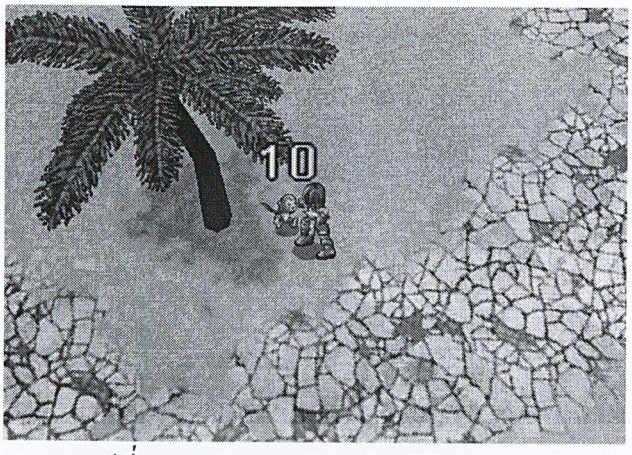
12.3 คลาส cRelativeCamera และ cNumber

คลาส cRelativeCamera คือลักษณะกล้องที่เป็น Third Person View ตามที่ได้อธิบายไปในข้างต้นแล้ว โดยจะประกอบด้วย Length, Alpha และ Zeta ที่ใช้ในการซูม ก้มเงย และหมุนกล้อง ส่วน XEye, YEye, ZEye นั้นเป็นจุดที่ผู้เล่นหลักอยู่ คือเราจะให้กล้องเคลื่อนที่ตามผู้เล่นหลัก โดยผู้เล่นหลักจะอยู่กลางหน้าจอตลอดเวลา



รูปที่ 12-2 คลาสไดอะแกรมของ cRelativeCamera และ cNumber

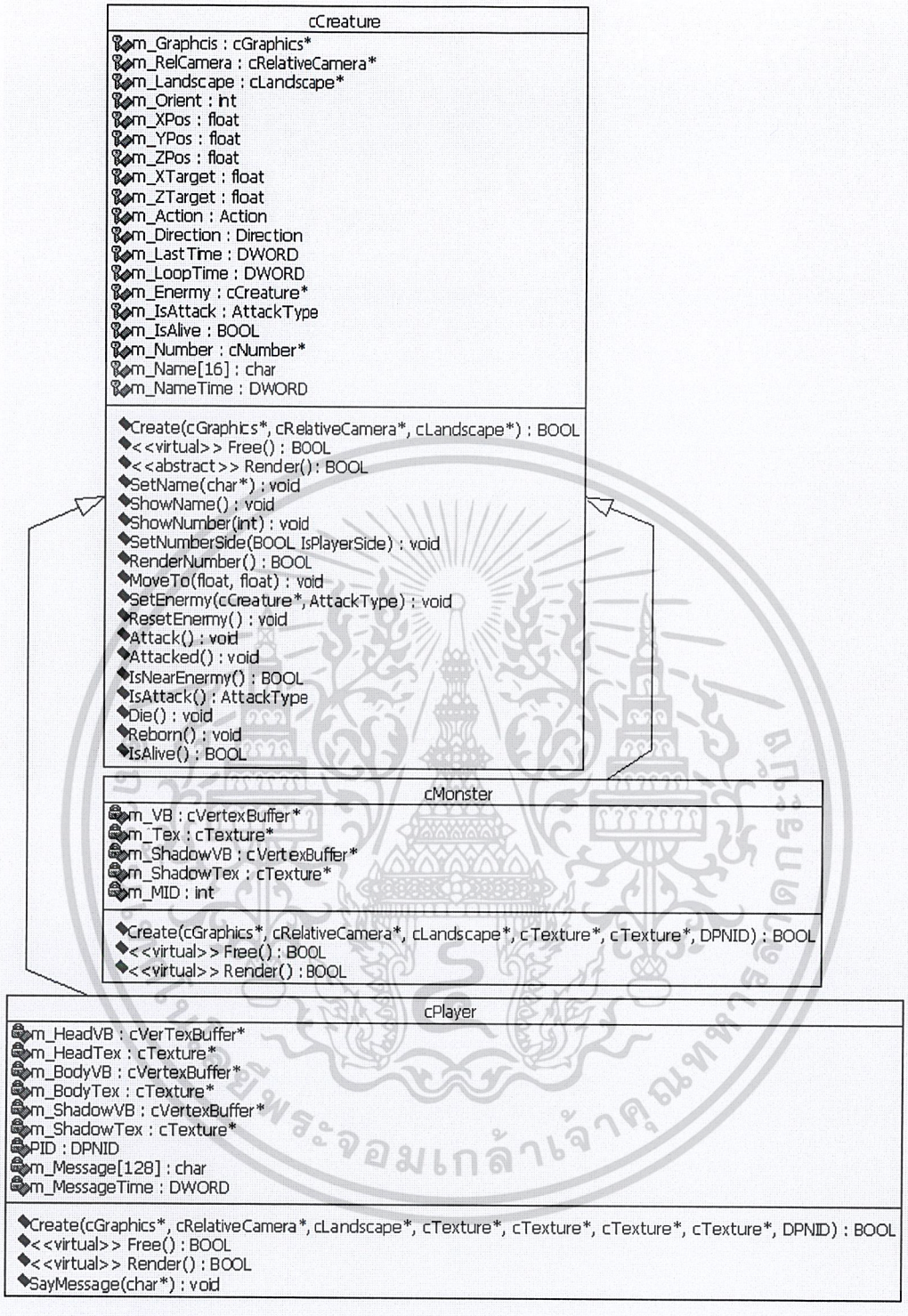
คลาส cNumber เป็นการแสดงผลตัวเลข โดยถ้าผู้เล่นตีศัตรูได้ จะมีค่าที่ตีได้ขึ้นเป็นสีขาวที่ฝั่งศัตรู และถ้าผู้เล่นโดนตีจะมีสีแดงขึ้นที่ตัวละครของผู้เล่น ตัวอย่างดังรูป



รูปที่ 12-3 การแสดงตัวเลขโดยใช้ cNumber

12.4 คลาส cCreature, cMonster และ cPlayer

คลาส cCreature เป็นคลาสแม่ของ cMonster และ cPlayer ทั้งสามคลาสเป็นตัวแทนของตัวละครตัวหนึ่งเหมือนกัน แต่สิ่งที่แตกต่างกันระหว่าง cMonster และ cPlayer คือ cPlayer มีการแยกส่วนหัวและส่วนตัวทำให้ต้องใช้ Vertex Buffer และ Texture 2 ชุด นอกจากนั้น จำนวนภาพและทิศทางไม่เท่ากัน ทำให้การคำนวณหา UV ไม่เหมือนกัน นอกจากนี้ cPlayer ยังสามารถแสดงข้อความพูดคุยได้ด้วย



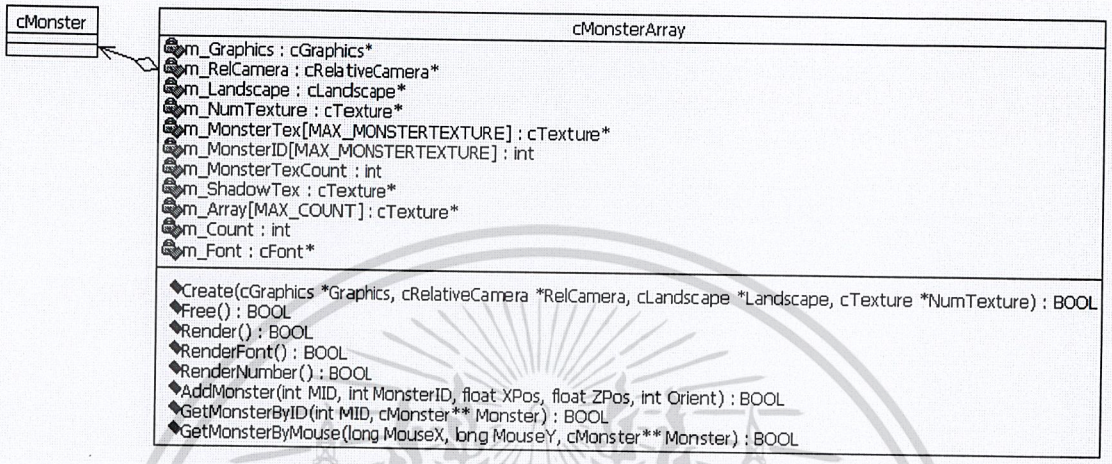
รูปที่ 12-4 คลาสไดอะแกรมของคลาส cCreature, cMonster และ cPlayer

12.5 คลาส cMonsterArray และ cPlayerList

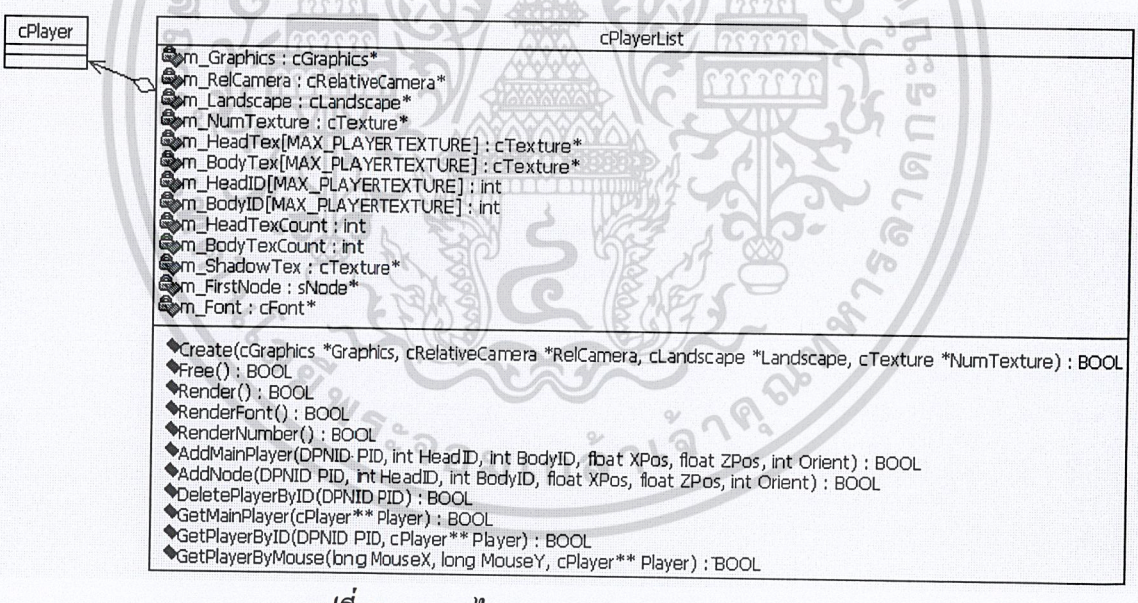
เป็น Contain Class เหมือนกัน โดย cMonsterArray ทำหน้าที่เก็บตัวศัตรูทั้งหมดบนแผนที่ cPlayerList ใช้เก็บผู้เล่นทั้งหมดบนแผนที่ แต่ที่ต่างกันคือ cMonsterArray ใช้ลักษณะอาเรย์ในการเก็บข้อมูล แต่ cPlayerList ใช้ลักษณะแบบ Link List ในการเก็บ ที่เป็นเช่นนี้เนื่องจากบนแผนที่หนึ่งๆ นั้นมี

จำนวนตัวศัตรูคงที่ ถ้าตายจะทำการแรนดอมไปเกิดใหม่บนจุดอื่นๆ ในแผนที่เดียวกัน แต่จำนวนผู้เล่นนั้นไม่คงที่ขึ้นอยู่กับจำนวนผู้เล่นในขณะนั้น ดังนั้นเราจึงเก็บตัวผู้เล่นในลักษณะ List แทน

ซึ่งทั้งสองคลาสก็มีลักษณะบางอย่างที่เหมือนกันคือ จะมีการเก็บ Texture Array ไว้เพื่อส่งให้ตัวละครใช้ในการเรนเดอร์ และไม่จำเป็นต้องโหลดใหม่ถ้าตัวละครนั้นใช้ texture ที่เคยโหลดมาแล้ว



รูปที่ 12-5 คลาสไดอะแกรมของ cMonsterArray



รูปที่ 12-6 คลาสไดอะแกรมของ cPlayerList

บทที่ 13
เทคนิคในการทำงานฝั่งไคลเอนต์

ในส่วนฝั่งไคลเอนต์นั้น จะรับผิดชอบในการรับข้อมูลต่างๆจากเซิร์ฟเวอร์ มาแสดงผลกราฟิกให้สวยงาม และทำการรับคำสั่งจากผู้เล่น ส่งให้เซิร์ฟเวอร์ประมวลผลต่อไป ฉะนั้นงานที่ไคลเอนต์รับบทหนักจริงๆ นั้น คือการแสดงผลกราฟิกนั่นเอง

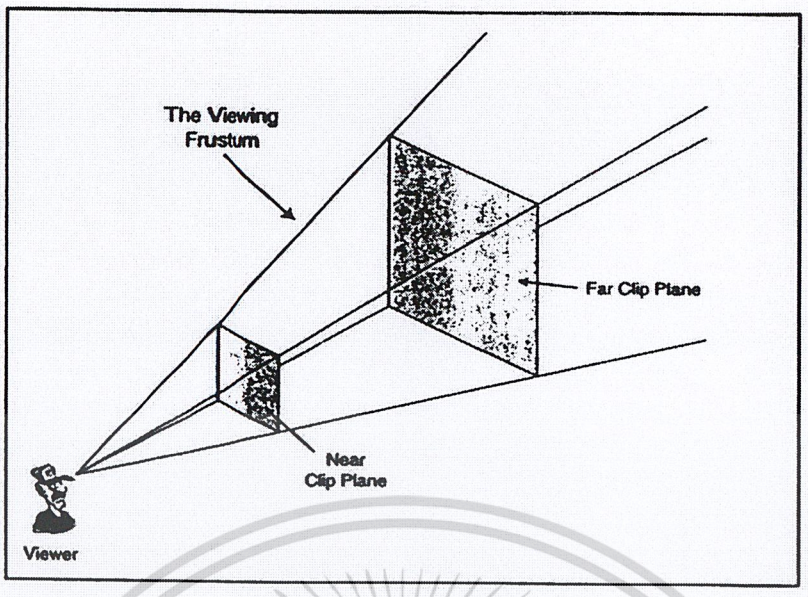
การทำงานในส่วนนี้ ก็ใช้ความสามารถของ Direct Graphic เป็นหลัก ซึ่งเราได้รวมไว้ในไลบรารีเกมเอนจินแล้ว แต่เนื่องจากเกมแบบ MMORPG นั้น มีลักษณะที่เด่นเป็นพิเศษเลยก็คือ ผู้เล่นจะแทนตัวละครตัวหนึ่งในโลกจำลอง และจะมีผู้เล่นคนอื่นๆ มากมายเข้ามาอยู่ร่วมในโลกจำลองนี้ ทำให้เราจำเป็นต้องนำอัลกอริทึมต่างๆ มาใช้ในส่วนไคลเอนต์ของเรา โดยจะอธิบายต่อไปตามหัวข้อต่อไปนี้

- Viewing Frustum and Node Tree Engine
- Billboarding, Alpha Blending and Alpha Testing
- Third Person View

13.1 Viewing Frustum and Node Tree Engine

เนื่องจากโลกจำลองหรือแผนที่ในส่วนไคลเอนต์ของเกมของเรามีขนาดค่อนข้างใหญ่และมีรายละเอียดมาก ทำให้จำนวนโพลีกอนมากตาม ซึ่งการเรนเดอร์ทั้งแผนที่ทุกๆ ลูปเกมนั้นเป็นไปไม่ได้ เพราะภาพที่ได้จะกระตุกมาก เนื่องจากหน่วยประมวลผลทางกราฟิกในการ์ดจอ (GPU : Graphics Processing Unit) ไม่สามารถทำงานได้ทัน ดังนั้นเราจึงต้องนำเทคนิค Viewing Frustum และ Node Tree มาใช้เพื่อให้การแสดงผลโลกที่มีจำนวนโพลีกอนมากๆ ของเราได้

Viewing Frustum คือ ปริมาตรสามมิติที่ประกอบด้วยระนาบ 6 ระนาบ โดยปริมาตรดังกล่าวจะมีความสัมพันธ์กับหน้ากล้อง หรือมุมมองผู้เล่น ดังรูปที่ 13-1



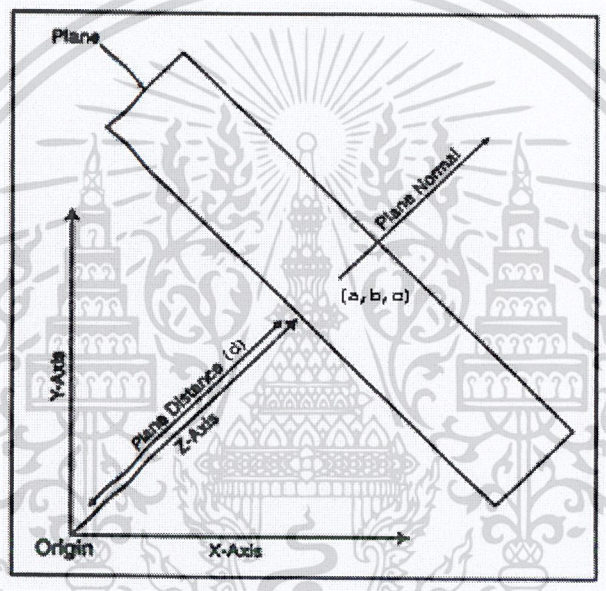
รูปที่ 13-1 Viewing Frustum

โดยระนาบทั้งหกด้านนี้เรียกว่า Clipping Planes ซึ่งเพลนหรือระนาบดังกล่าว จะทำให้เราพิจารณาได้ว่าอ็อบเจ็กต์ใดบ้างที่ผู้ใช้เห็นในขณะนั้นและจำเป็นจะต้องทำการเรนเดอร์ ซึ่งเราสามารถคำนวณหาระนาบทั้งหกได้จากสูตร

```
D3DXMATRIX Matrix, matView, matProj;

m_Graphics->GetDeviceCOM()->GetTransform(D3DTS_PROJECTION, &matProj);
m_Graphics->GetDeviceCOM()->GetTransform(D3DTS_VIEW, &matView);
D3DXMatrixMultiply(&Matrix, &matView, &matProj);

// Calculate the planes
m_Planes[0].a = Matrix._14 + Matrix._13; // Near
m_Planes[0].b = Matrix._24 + Matrix._23;
m_Planes[0].c = Matrix._34 + Matrix._33;
m_Planes[0].d = Matrix._44 + Matrix._43;
D3DXPlaneNormalize(&m_Planes[0], &m_Planes[0]);

m_Planes[1].a = Matrix._14 - Matrix._13; // Far
m_Planes[1].b = Matrix._24 - Matrix._23;
m_Planes[1].c = Matrix._34 - Matrix._33;
m_Planes[1].d = Matrix._44 - Matrix._43;
D3DXPlaneNormalize(&m_Planes[1], &m_Planes[1]);

m_Planes[2].a = Matrix._14 + Matrix._11; // Left
m_Planes[2].b = Matrix._24 + Matrix._21;
m_Planes[2].c = Matrix._34 + Matrix._31;
m_Planes[2].d = Matrix._44 + Matrix._41;
D3DXPlaneNormalize(&m_Planes[2], &m_Planes[2]);

m_Planes[3].a = Matrix._14 - Matrix._11; // Right
m_Planes[3].b = Matrix._24 - Matrix._21;
m_Planes[3].c = Matrix._34 - Matrix._31;
m_Planes[3].d = Matrix._44 - Matrix._41;
D3DXPlaneNormalize(&m_Planes[3], &m_Planes[3]);
```

```
m_Planes[4].a = Matrix._14 - Matrix._12; // Top
m_Planes[4].b = Matrix._24 - Matrix._22;
m_Planes[4].c = Matrix._34 - Matrix._32;
m_Planes[4].d = Matrix._44 - Matrix._42;
D3DXPlaneNormalize(&m_Planes[4], &m_Planes[4]);

m_Planes[5].a = Matrix._14 + Matrix._12; // Bottom
m_Planes[5].b = Matrix._24 + Matrix._22;
m_Planes[5].c = Matrix._34 + Matrix._32;
m_Planes[5].d = Matrix._44 + Matrix._42;
D3DXPlaneNormalize(&m_Planes[5], &m_Planes[5]);
```

ค่า a, b, c และ d นั้นเป็นตัวแปรที่แทนระนาบ โดยค่า a, b, c คือจุด x, y, z ที่รังสีจากจุกกำเนิดตัดตั้งฉากกับระนาบ และ d คือระยะทางจากจุดกำเนิดไปยังจุดตัดนั้นบนระนาบ



รูปที่ 13-2 ตัวแปร a, b, c และ d ของระนาบ

ซึ่งการพิจารณาว่าจุดหนึ่งๆ หรือเวอร์เท็กซ์หนึ่งๆ นั้นอยู่ในฟรัสทัมหรือไม่นั้น ทำได้การหา Dot Product ของจุดนั้นกับแต่ละระนาบ โดยใช้คำสั่ง

```
FLOAT D3DXPlaneDotCoord(
      CONST D3DXPLANE * pP,    //D3DXPLANE to check
      CONST D3DVECTOR* pV);    //Point to check
```

โดยฟังก์ชันนี้จะคืนค่ามาเป็นระยะทางระหว่างจุดกับระนาบ ซึ่งถ้าได้ค่าเป็นบวกแสดงว่าจุดอยู่หน้าระนาบ ถ้าเป็นลบแสดงว่าจุดอยู่หลังระนาบ เมื่อเราทำการหา Dot Product ของจุดใดๆ กับระนาบทั้งหกแล้ว เราก็จะรู้ได้ว่าจุดๆ นั้นอยู่นอกหรืออยู่ในฟรัสทัม โดยทำใช้ฟังก์ชันตรวจสอบดังนี้


```
BOOL cFrustum::CheckPoint(float XPos, float YPos, float ZPos)
{
```

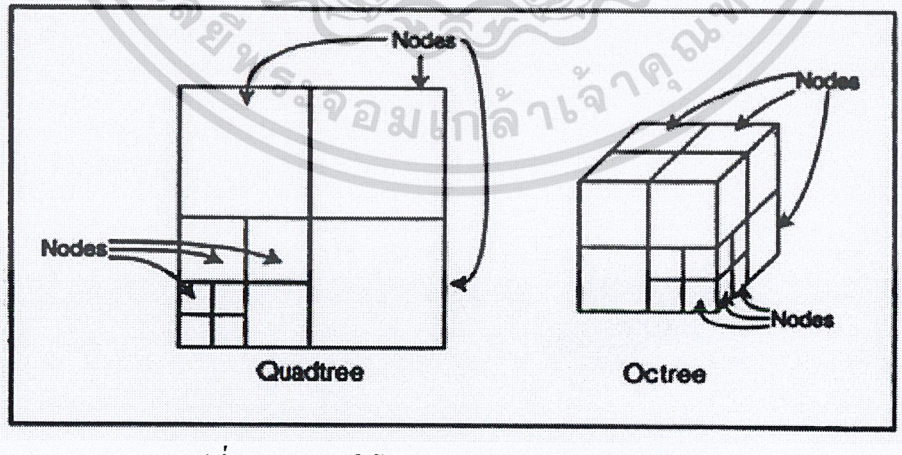
```
    short i;

    // Make sure point is in frustum
    for(i=0;i<6;i++) {

        if(D3DXPlaneDotCoord(&m_Planes[i],&D3DXVECTOR3(XPos,YPos,ZPos))<0.0f)
       return FALSE;
    }

    return TRUE;
}
```

ถ้าจุดนี้อยู่ด้านหลังระนาบใดระนาบหนึ่งแสดงว่าจุดนั้นอยู่นอกฟรัสทัม หรือถ้าเราต้องการหาว่าโพลีกอนใดอยู่ในฟรัสทัมหรือไม่ เราก็ทำการตรวจสอบเช่นเดียวกันนี้กับจุดทั้ง 3 จุด ของโพลีกอนว่า ถ้ามีจุดใดจุดหนึ่งอยู่ในฟรัสทัมโพลีกอนนั้นถือว่าอยู่ในฟรัสทัมและต้องเรนเดอร์

จากแนวคิดเรื่อง Viewing Frustum นั้นทำให้อ็อบเจ็กต์หรือโพลีกอนที่อยู่ในปริมาตรนี้เท่านั้นที่จะถูกเรนเดอร์ ช่วยให้เราสามารถลดการปริมาณการประมวลผลในขณะเรนเดอร์ไปได้มากทีเดียว แต่อย่างไรก็ดี เราหลีกเลี่ยงไม่ได้ที่จะต้องสแกนผ่านทุกๆ โพลีกอนในโมเดลของเราแล้วพิจารณาว่าโพลีกอนนั้นอยู่ในฟรัสทัมหรือไม่ และจะต้องสแกนใหม่ทุกๆ ลูปการเรนเดอร์ ซึ่งนั่นก็เป็นการเสียเวลาอย่างมากทีเดียว

ดังนั้นแทนที่เราจะสแกนผ่านเพื่อพิจารณาทุกๆ โพลีกอน เราจะนำเทคนิค Node Tree เข้ามาช่วย โดยเราจะทำการแบ่งโมเดลขนาดใหญ่ของเราออกเป็นส่วนย่อยๆ เรียกว่าโหนด และจัดเรียงโหนดเหล่านั้นให้อยู่ในรูปของ Tree จากนั้นเราจะสามารถแสกนผ่าน Tree ได้โดยง่าย และทำการเรนเดอร์โหนดที่มีบางส่วนอยู่ในฟรัสทัมเลย ซึ่งช่วยเราลดเวลาในการสแกนไปได้มากทีเดียว

โดยลักษณะการแบ่งโมเดลออกเป็นโหนดย่อยๆนั้น สามารถทำได้ 2 แบบด้วยกันคือ แบบ Quadtree และแบบ Octree ดังรูป



รูปที่ 13-3 การแบ่งโหนดแบบ Quadtree และ Octree

จะเห็นว่าแบบ Quadtree จะทำการแบ่งโหนดใหญ่ออกเป็นครั้งละ 4 โหนดย่อย โดยจะแบ่งเฉพาะระนาบ XZ (2D-space) ดังนั้นแบบนี้จะไม่เหมาะกับเกมที่มีการเปลี่ยนตำแหน่งในแนวแกน Y

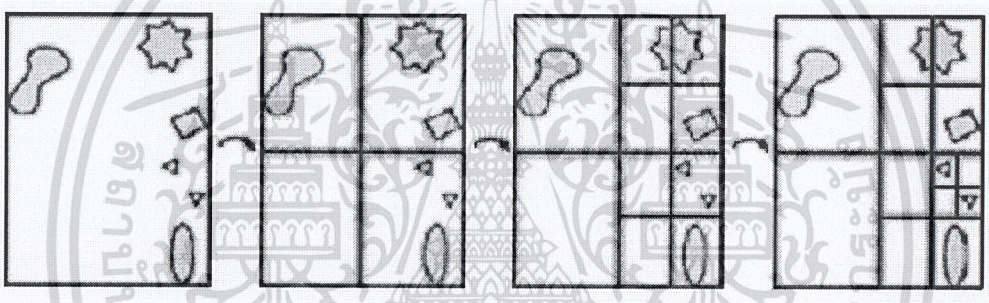
มากๆ และอีกแบบคือแบบ Octree จะแบ่งโหนดใหญ่ออกเป็นโหนดรูปทรงสี่เหลี่ยมครั้งละ 8 โหนดย่อย ในระนาบ XYZ (3D-space) ซึ่งแต่ละโหนดจะเป็นตัวแทนของกลุ่มโพลีกอนที่อยู่บริเวณที่กำหนด

โดยโหนดใหญ่จะมีความสัมพันธ์กับโหนดย่อยที่ถูกแบ่งออกนั้น ในลักษณะพ่อกับลูก (Parent-Child) ซึ่งโหนดใหญ่สุดหรือโมเดลทั้งโมเดล เรียกว่า Root Node

การสร้าง Node Tree นั้นจะทำเพียงครั้งเดียวเท่านั้น ฉะนั้นความเร็วช้าในการสร้างจึงไม่ใช่ปัญหาสำคัญ โดยวิธีการสร้างโหนดคือ

อันดับแรกเราจะกำหนด Bounding Box หรือกล่องบริเวณที่ต้องการจัดกลุ่มโพลีกอน จากนั้นจะสแกนดูว่ามีโพลีกอนใดอยู่หรือมีบางส่วนอยู่ในกล่องนี้บ้าง (หนึ่งโพลีกอนอาจอยู่ได้หลายโหนด)

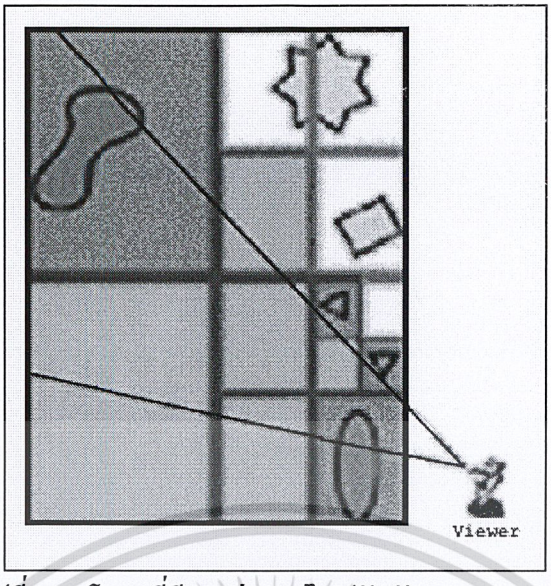
หลังจากนั้นทำการแบ่งกล่องออกเป็นกล่องย่อยๆ ทำการสแกนเช่นเดียวกันอีกครั้ง ทำการแบ่งโหนดที่ยังมีขนาดใหญ่หรือมีโพลีกอนมากกว่ากำหนดไปเรื่อยๆ จนกว่าจะได้ขนาดโหนดที่เล็ก และมีจำนวนโพลีกอนไม่มากเกินไปตามที่ต้องการ ส่วนโหนดใดที่ไม่มีโพลีกอนอยู่เลยเราจะทำการละทิ้งโหนดนั้นไป



รูปที่ 13-4 การแบ่งโหนดออกเป็นโหนดย่อยๆ

เมื่อเสร็จแล้วเราก็จะได้ Node Tree ที่มีขนาดโหนดตามที่ต้องการ จากนั้นเมื่อเราจะเรนเดอร์ เราก็จะทำการสแกนโหนด โดยจะเริ่มที่ Root Node ถ้าโหนดนั้นมีส่วนหนึ่งส่วนใดอยู่ในฟรัสทัม จะทำการสแกนลงไปในโหนดลูก ถ้าไม่มีส่วนใดอยู่ในฟรัสทัมก็จะไม่สนใจโหนดนั้น

จากนั้นจะทำการแสกนต่อๆ ไปยังโหนดลูก จนกระทั่งไม่มีลูกต่อแล้ว จะสิ้นสุดการสแกน และทำการคัดลอกข้อมูลโพลีกอนจากทุกๆ โหนดสุดท้ายที่มีส่วนอยู่ในฟรัสทัมมารวมกันไว้ในบัฟเฟอร์ จากนั้นจะทำการเรนเดอร์บัฟเฟอร์นั้น



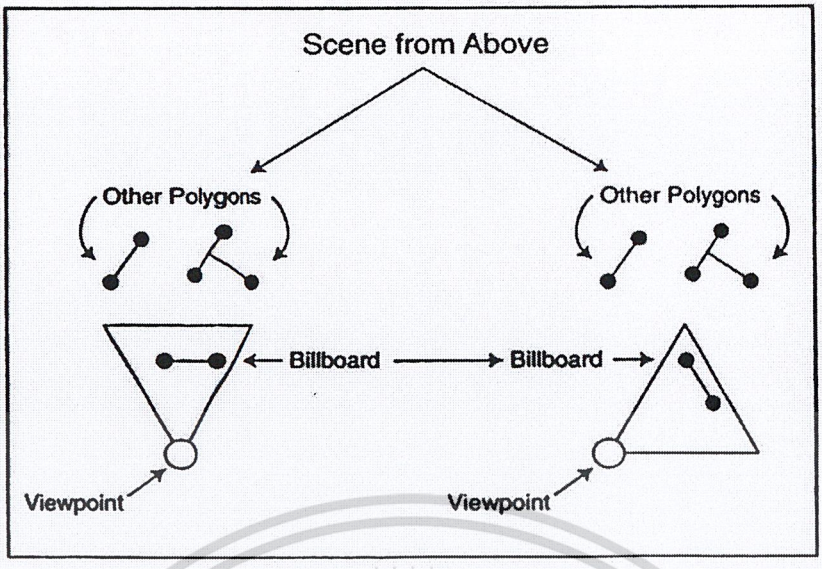
รูปที่ 13-5 โหนดที่มีบางส่วนอยู่ในฟรัสทัมจะถูกเรนเดอร์

จากการใช้เทคนิค Viewing Frustum และ Node Tree จะทำให้เราสามารถเรนเดอร์โมเดลแผนที่ที่มีขนาดใหญ่และมีโพลีกอนมากได้ โดยช่วยแก้ปัญหาโมเดลขนาดใหญ่ไปได้มากทีเดียว

13.2 Billboarding ,Alpha blending ang Alpha Testing

จากการที่เกม MMORPG นี้จะต้องมีผู้เล่นจำนวนมากอยู่ในโลกใบเดียวกัน และในหนึ่งหน้าจอที่ผู้เล่นเห็นอาจมีผู้เล่นอยู่พร้อมกันจำนวนมาก ดังนั้นการที่จะได้ภาพตัวละครสามมิติที่สวย จำเป็นต้องใช้โพลีกอนต่อหนึ่งตัวละครมากๆ อาจเป็นสิ่งเกินกำลังต่อการเรนเดอร์ ดังนั้นเราจะใช้ภาพตัวละครสองมิติเข้ามาอยู่ในโลกสามมิติ โดยสร้างโพลีกอน 2 อันประกอบกันเป็นรูปสี่เหลี่ยมเพื่อแปะภาพตัวละคร และใช้เทคนิค Billboarding เข้ามาช่วย โดยการทำ Billbaording คือ การทำให้รูปสี่เหลี่ยมของเราหันหน้าหากล้องตลอดเวลา ไม่ว่ากล้องจะไปทางไหนก็ตาม ก็จะเห็นภาพบนรูปสี่เหลี่ยมหรือตัวละครของเราได้อย่างถูกต้อง ไม่บิดเบี้ยว

โดยเราจะต้องหา World Transform Matrix มาหมุนสี่เหลี่ยมให้หันหน้าเข้ากล้อง ซึ่งคำนวณได้จาก View Transform Matrix ว่าตอนนี้กล้องหมุนทำมุมอย่างไร เพื่อคำนวณให้สี่เหลี่ยมของเราหมุนกลับก็จะได้ภาพที่หันหน้าหากล้องตลอดเวลา



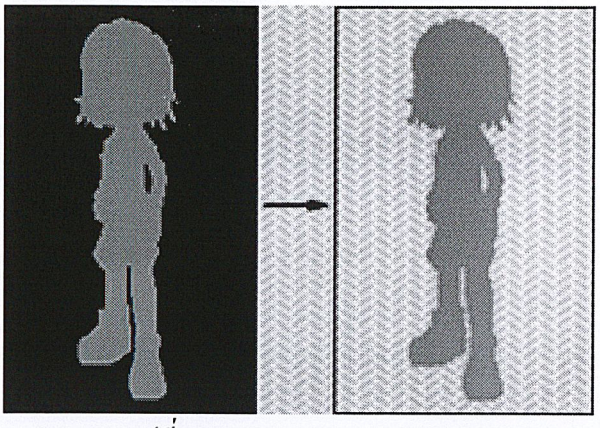
รูปที่ 13-6 การทำ Billboarding

```
// g_D3Ddevice = pre-initialized device object

D3DXMATRIX matBillboard;

D3DXMatrixIdentity(&matBillboard);

// calculate billboarding matrix
g_D3Ddevice ->GetTransform(D3DTS_VIEW, &matView);
D3DXMatrixTranspose(&matTransposed, &matView);
matTransposed._41 = matTransposed._42 = matTransposed._43 =
matTransposed._14 = matTransposed._24 = matTransposed._34 = 0.0f;

// Apply billboard matrix to world transform matrix
g_D3Ddevice->SetTransform(D3DTS_WORLD, &matTempWorld);
```

หรือถ้าใช้เกมเอนจินของเราจะใช้คำสั่งว่า

```
// m_WorldPos = pointer to pre-initialized cWorldPosition object
// m_Graphics = pointer to pre-intialized cGraphics object

m_WorldPos->EnableBillboard(TRUE);
m_Graphics-> SetWorldPosition(&m_WolrdPos);
```

และจากโพลีกอนที่ทำเป็นรูปสี่เหลี่ยมนั้น เราจะทำการแปะภาพตัวละครลงไป ซึ่งเราจะต้องทำให้ส่วนที่ไม่ใช่ตัวละครโปร่งใส เพื่อให้เห็นแต่ตัวละครจริงๆ โดยเราจะนำเทคนิค Alpha Testing เข้ามาช่วย คือทำการกำหนด Color Key ที่จะให้ทำโปร่งใส ดังในรูปเราจะให้สีดำโปร่งใสเป็นต้น



รูปที่ 13-7 การทำ Alpha Testing

วิธีการทำ Alpha Testing คือเราจะใช้คำสั่ง D3DXCreateTextureFromFileEx โดยมีลักษณะคำสั่งดังรูป

```
HRESULT D3DXCreateTextureFromFileEx(
  LPDIRECT3DDEVICE8  pDevice,
  LPCTSTR pSrcFile,
  UINT Width,
  UINT Height,
  UINT MipLevels,
  DWORD Usage,
  D3DFORMAT Format,
  D3DPOOL Pool,
  DWORD Filter,
  DWORD MipFilter,
  D3DCOLOR ColorKey,
  D3DXIMAGE_INFO* pSrcInfo,
  PALETTEENTRY* pPalette,
  LPDIRECT3DTEXTURE8* ppTexture
);
```

ซึ่งออกจะวุ่นวาย ดังนั้นเราจะใช้เป็นคำสั่งของเกมเอนจินของเราแทน โดยใช้คำสั่งคือ

```
m_Texture->Load(m_Graphics, "texture.bmp", 0xFFFFFF);
```

เป็นการสั่งโหลดภาพจากไฟล์ texture.bmp โดยกำหนด Color Key เป็นสีดำ และก่อนการเรนเดอร์เราจะต้องทำการ Enable Alpha Testing เสียก่อนโดยใช้คำสั่ง

```
m_Graphics->EnableAlphaTesting(TRUE);
```

แต่ข้อเสียของการทำ Alpha Testing คือเราสามารถกำหนดสีที่จะให้โปร่งใสได้สีเดียวเท่านั้น เช่น ถ้าเรากำหนดให้สีดำโปร่งใส ถ้าสีเกือบๆ ดำ ไม่ดำสนิทก็จะเห็นขึ้นเป็นขอบสีเกือบดำนั้น ซึ่งเราจะต้องใช้ Alpha Blending เข้ามาช่วย เพื่อเกลี่ยสีที่เกือบดำนั้นให้โปร่งแสงด้วย โดยวิธีใช้อย่างง่ายๆ นั้น เราไม่ต้องทำอะไรมากเลย เนื่องจากเกมเอนจินของเราโดยค่าเริ่มต้นแล้วได้กำหนดให้ทำการ Blend สีไปยังค่าสีของ Vertex อยู่แล้ว เราไม่ต้องใช้คำสั่งในการ Blend ด้วยตนเองเลย เพียงแค่กำหนดสีให้กับ Vertex ตามสีที่เราต้องการเท่านั้น จากนั้นก่อนการเรนเดอร์เราก็ทำการสั่ง Enable Alpha Blending ด้วย

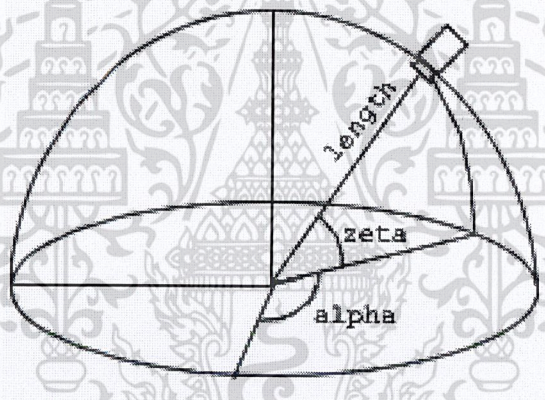
```
m_Graphics->EnableAlphaBlending(TRUE);
```

จากการใช้ Billboarding, Alpha Blending และ Alpha Testing นี้ ทำให้เราสามารถแสดงผลภาพตัวละครสองมิติ ในโลกสามมิติของเราได้แล้ว

13.3 Third- Person View

มุมมองของกล้องที่ใช้ในเกมของเรานั้นเป็นแบบมุมมองบุคคลที่สาม คือผู้เล่นเกมจะมองเห็นตัวละครของตนว่ากำลังกระทำอะไรกับใครอยู่บ้าง โดยเราสามารถปรับมุมกล้องได้ 3 ลักษณะ คือ การหมุนกล้องวนรอบแกน Y การก้มเงยกล้อง และการซูมเข้าออก โดยเราจะทำการสร้างคลาสกล้องขึ้นมาใหม่เก็บตัวแปรเบื้องต้นสามตัว คือ Length, Zeta และ Alpha โดยที่

- Length จะใช้ในกรณีซูมเข้าออก
- Zeta ใช้ในกรณีก้มเงยกล้อง
- Alpha ใช้ในกรณีหมุนกล้อง



รูปที่ 13-8 ตัวแปรมุมกล้อง Length, Zeta และ Alpha

แต่เนื่องจากภาพตัวละครของเราเป็นสองมิติ การจะหมุนมุมกล้องแล้วเจอภาพทิศทางเดิมตลอดนั้นไม่ควรเกิดขึ้น ดังนั้นเราจะต้องทำภาพตัวละครในทิศทางต่างๆ ซึ่งในที่นี้ตัวละครของเรามีแปดทิศทาง ซึ่งเราจะต้องเตรียมภาพในแปดทิศทางนี้ สำหรับทุกๆ ท่างทาง ไม่ว่าจะ เดิน ต่อสู้ บาดเจ็บ เป็นต้น จะเห็นได้ว่าเราจะต้องใช้ภาพจำนวนมหาศาลเลยทีเดียว

ซึ่งถ้าเราจะใช้วิธี สร้าง Texture Buffer จำนวนมากๆ และใช้การสลับ Texture ตลอดเวลาที่หมุนกล้องนั้นเป็นไปไม่ได้ เพราะการสลับ Texture หนึ่งครั้งนั้นใช้เวลาและทรัพยากรมาก เพื่อแก้ปัญหานี้ เราจะใช้เทคนิคที่คล้ายคลึงกับที่ใช้ในเกมสองมิติเข้ามาช่วย นั่นก็คือวิธี Sprite Animation โดยจะเป็นการรวมเอาภาพทุกภาพ ทุกทิศทางมาต่อรวมกันในภาพเดียว ทำให้เราไม่ต้องสลับ Texture และเราจะใช้การเปลี่ยนค่าใน UV-Coordinate หรือ Texture Coordinate แทนนั่นเอง

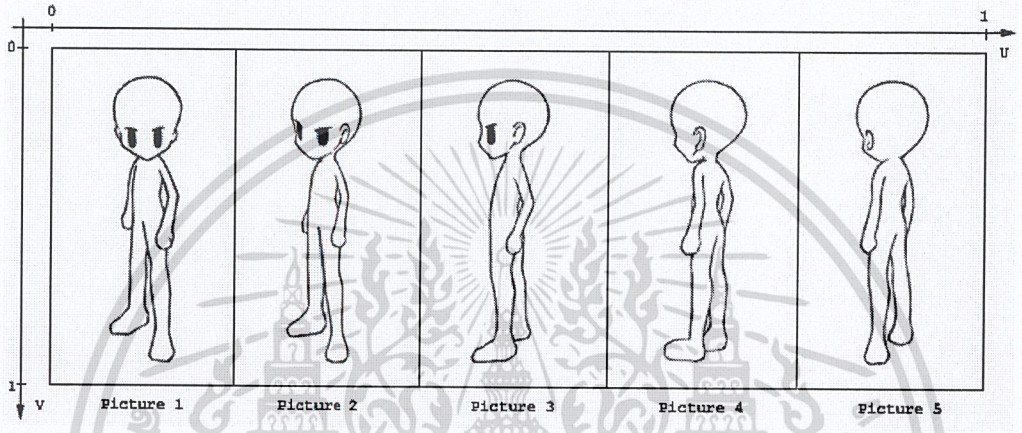
เช่นถ้าเรามีภาพดังรูปที่ 13-9 ซึ่งเป็นภาพท่ายืน 5 ทิศทาง และเราจะทำการ Mirror ภาพที่ 2, 3 และ 4 เพื่อให้ได้ภาพทั้งหมดแปดทิศ

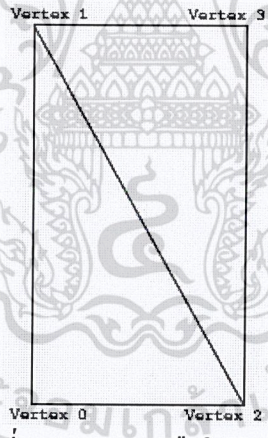
ถ้าต้องการแสดงภาพด้านหน้า เราต้องทำการกำหนดค่า UV ของ Vertex ทั้ง 4 จุดดังนี้

```
vertex0.u = 0.0f;              vertex0.v = 1.0f;
vertex1.u = 0.0f;              vertex1.v = 0.0f;
vertex2.u = 1.0f / 5.0f;       vertex2.v = 1.0f;
vertex3.u = 1.0f / 5.0f;       vertex3.v = 0.0f;
```

และถ้าต้องการแสดงภาพด้านข้างที่หันหน้าไปทางซ้าย ค่า UV จะเป็นดังนี้

```
vertex0.u = 2.0f / 5.0f;       vertex0.v = 1.0f;
vertex1.u = 2.0f / 5.0f;       vertex1.v = 0.0f;
vertex2.u = 3.0f / 5.0f;       vertex2.v = 1.0f;
vertex3.u = 3.0f / 5.0f;       vertex3.v = 0.0f;
```





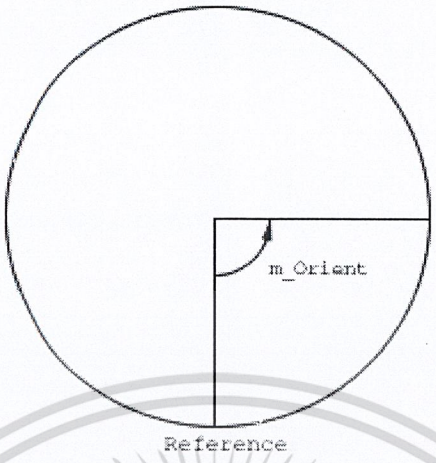
รูปที่ 13-9 บน ภาพยืนในทิศต่างๆ

รูป 13-10 ล่าง Vertex ที่ใช้ทำรูปสี่เหลี่ยมเพื่อแสดงภาพ

ซึ่งในการใช้งานจริงจะมีภาพมากกว่านี้มากนัก เนื่องจากเราต้องใช้หลายๆภาพ ทำเป็นภาพเคลื่อนไหวในหนึ่งท่าทาง แต่การกำหนดค่า UV ก็จะหลักการเดียวกันกับที่ใช้กำหนดในข้างต้นนี้

ในขั้นตอนต่อไป การที่เราจะรู้ได้ว่าในขณะนั้นจะต้องแสดงภาพิศทางไหน เราจะต้องรู้ก่อนว่าตัวละครหันด้านใดให้กับหน้ากล้อง เพื่อที่เราจะรู้ได้ว่าตัวละครหันด้านใดให้กับกล้องนั้น เราจะต้องใช้ตัวแปรอีก 2 ตัว เข้ามาช่วย ตัวแปรแรกคือ ตัวแปรที่ใช้เก็บทิศทางของคน ว่าคนนี้หันหน้าไปทิศทางไหน

เช่น ถ้าคนหันหน้าไปทางขวา ดังรูป เราจะเก็บค่ามุมเทียบกับทิศอ้างอิง จะได้ค่า m_Orient ของตัวละครตัวนี้เป็น 90 องศา



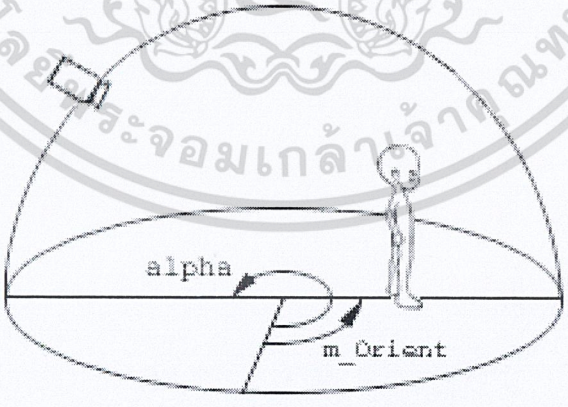
รูปที่ 13-11 การเก็บค่าทิศทางของตัวละคร

เมื่อเรารู้ว่าตัวละครของเราหันไปทางทิศใดแล้ว แต่เรายังไม่สามารถแสดงภาพที่ถูกต้องได้ เนื่องจากยังไม่ใช่ทิศทางที่หันหากล้อง ดังนั้นเราจะนำค่าทิศทางของกล้องหรือค่า Alpha มาใช้เป็นตัวแปรที่สอง โดยเราจะรู้ได้ว่าคนหันหน้าทิศใดให้กับกล้อง เราจะนำค่า Alpha ของกล้อง มาหาความแตกต่างกับทิศทาง m_Orient ของตัวละคร เช่นตัวอย่างในรูป

m_Orient = 90 เนื่องจากคนหันไปทางขวา ทำมุม 90 องศากับทิศอ้างอิง

Alpha = 270 กล้องหมุนไป 270 องศา

จะได้ว่าภาพที่เราต้องแสดงคือภาพที่มีทิศทาง 180 องศาหรือ เห็นด้านหลังของตัวละครนั่นเอง



รูป 13-12 การคำนวณหาภาพตัวละครในทิศทางที่ควรเห็น

บทที่ 14
บทวิจารณ์และสรุป

14.1 สรุปผลการวิจัย

จากการทำการศึกษาไดเร็กเอ็กซ์ ไดเร็กเอ็กซ์เป็นชุดคำสั่งที่ช่วยให้การพัฒนาเกมสะดวก และง่ายยิ่งขึ้น แต่อย่างไรก็ตามการใช้ไดเร็กเอ็กซ์ยังคงมีความซับซ้อน และยุ่งยากสำหรับผู้พัฒนามือใหม่ ดังนั้นการนำไดเร็กเอ็กซ์มาพัฒนาจนกลายเป็นเกมเอนจิน ทำให้ผู้พัฒนาสามารถใช้งานได้อย่างไม่ซับซ้อน และลดขั้นตอนที่ยุ่งยากมาเป็นส่วนที่ง่าย และไม่ซับซ้อน ซึ่งทำให้ผู้พัฒนาสามารถพัฒนาเกมได้อย่างรวดเร็ว และใช้เวลาในการศึกษาการทำงานของเกมเอนจินไม่มาก ทำให้การพัฒนาแอพพลิเคชัน หรือเกมที่ใช้ความสามารถทางมัลติมีเดียสามารถทำได้โดยง่าย สะดวก และรวดเร็ว และสามารถนำไปใช้ได้อย่างมีประสิทธิภาพ

14.2 แนวทางในการพัฒนาต่อ

แนวทางในการพัฒนาต่อนั้นจะช่วยให้เกมที่ได้มีการทำงานที่มีประสิทธิภาพมากขึ้น และใช้งานทรัพยากรได้อย่างคุ้มค่า ซึ่งจะมีแนวทางดังนี้

- ปรับปรุงการส่งข้อมูลระหว่างไคลเอนต์กับเซิร์ฟเวอร์ได้มีประสิทธิภาพเพิ่มขึ้น โดยลดข้อมูลที่ไม่จำเป็นลง
- มีการนำ DBMS มาช่วยจัดการในจัดเก็บฐานข้อมูล ซึ่งจะช่วยให้การทำงานกับข้อมูลมีประสิทธิภาพเพิ่มขึ้น
- เพิ่มความปลอดภัยการรับส่งข้อมูล ซึ่งจะช่วยรักษาข้อมูลของผู้ใช้ให้เป็นความลับ เช่น รหัสในการเข้าไปเล่นเกม เป็นต้น
- เพิ่มความสามารถของเกมให้มีความสนุกยิ่งขึ้น เช่น เพิ่มการรวมกลุ่มของผู้เล่น เพิ่มอาวุธ เพิ่มตัวละคร เพิ่มฉาก เพิ่มศัตรู เป็นต้น

บรรณานุกรม

[1] Jim Adams :"Programming Role Playing Game With Direct X ", Premier Press, 2002

[2] Mason McCuskey : "Special Effects Game Programming with Direct X", Premier Press, 2002

[3] Mark DeLoura (2000) : "Game Programming Gems", Charles River Media, 2000.

[4] พีรภัทร์ สว่างเพียร : "เทคนิคการเขียนโปรแกรมและเกมด้วย Visual C++", ซีเอ็ด 2545

[5] Todd Barron : "Multiplayer Game Programming", Prima Tech, 2000