การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้

Graphic User Interface 3D RPG Game Engine

นาย ชาคริต สวัสดิรักษ์

นาย ชยานนท์ รูปประดิษฐ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้

Graphic User Interface 3D RPG Game Engine

โดย

นาย ชากริต สวัสดิรักษ์

นาย ชยานนท์ รูปประดิษฐ์

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2546

ปริญญานิพนธ์ปีการศึกษา 2546

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้ Graphic User Interface 3D RPG Game Engine

ผู้จัดทำ

1. นาย ชยานนท์ รูปประดิษฐ์ รหัสประจำตัว 43010084
2. นาย ชาคริต สวัสดิรักษ์ รหัสประจำตัว 43010098

..... อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้

นาย ชาคริต สวัสดิรักษ์
นาย ชยานนท์ รูปประดิษฐ์
ดร. วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2546

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาโปรแกรมสร้างเกมประเภท RPG 3 มิติแบบหน้าต่างติดต่อผู้ใช้ เพื่อให้นำไปใช้งานง่ายขึ้น โดยที่ผู้สร้างเกมไม่จำเป็นต้องมีความรู้ในด้านการเขียนโปรแกรมเลย ระหว่างการสร้างเกมผู้สร้างสามารถเห็นกราฟฟิกจริงๆที่จะถูกแสดงเมื่อเกมถูกนำไปเล่น สามารถสร้างเนื้อเรื่องผ่านทางการกำหนดเหตุการณ์ต่างๆให้กับเกมได้ และส่วนติดต่อผู้ใช้ของโปรแกรมนี้ถูกออกแบบให้ง่ายต่อการเรียนรู้ เป็นแอพพลิเคชั่นซึ่งผู้ใช้สามารถกำหนดสิ่งต่างๆลงไปในเกมได้ตั้งแต่การเลือกโมเดลไปจนถึงการสร้างเนื้อเรื่องให้กับเกม ซึ่งผลลัพธ์สุดท้ายก็คือเกมที่สามารถเล่นได้จริงที่มีเรื่องราวเหมือนกับที่ผู้สร้างได้วางไว้ตั้งแต่แรก

เนื้อหาของปริญญานิพนธ์ฉบับนี้ ในส่วนของ บทที่ 2 ทฤษฎีและหลักการของเกมสามมิติ และบทที่ 3 ไดเร็คเอ็กซ์ จะมีเนื้อหาตรงกันกับปริญญานิพนธ์ของโครงงาน การพัฒนาเกมสามมิติบนระบบเครือข่าย เนื่องจากเป็นโครงงานในประเภทเกมที่อ้างอิงทฤษฎีเดียวกัน

Graphic User Interface 3D RPG Game Engine

Chayanont Roopppradid

Chakrit Sawatdirak

Prof. Dr Voravat Limpoka Advisor

ABSTRACT

This project is about developing a constructional 3D RPG games tool which has user interface in order to be easy to use or learn. Creators are not necessary to have coding program skill. They can see the real graphic same as game players will see when they are playing this game. This tool can specify games detail since select model till create game's story line.

In part of lesson 2 and 3 of this document, they are same as the document from. Due to they are the same type so theories and details are the same.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือ และความร่วมมือจากหลาย ๆ ฝ่ายขอขอบพระคุณอาจารย์วรวัฒน์ ลิ้มโภคา ซึ่งอาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความใส่ใจดูแลเอาใจใส่ ให้คำปรึกษาอย่างใกล้ชิดด้วยดีตลอดมา ทำให้ผู้จัดทำรู้สึกภูมิใจเป็นอย่างมากที่ได้มีโอกาสได้ทำปริญญานิพนธ์นี้

ขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดู ให้โอกาสในการศึกษา และคอยให้กำลังใจเสมอมาในทุก ๆ ด้านอันหาที่เปรียบมิได้ นอกจากนี้ขอขอบพระคุณคณาจารย์ทุกท่านที่ประสิทธิประสาทวิชาความรู้ให้แก่ข้าพเจ้าตลอดมาตั้งแต่ได้มาศึกษาเล่าเรียนในสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังจนมีวันนี้

ขอขอบคุณทุกคนในห้องวิจัย OLALA ที่คอยซักถามความคืบหน้า คอยกระตุ้น และคอยเป็นกำลังใจในการทำงานจนสำเร็จได้

สุดท้ายขอขอบคุณเพื่อน ๆ ในแต่ละกลุ่มที่ร่วมพัฒนาเกมด้วยกันในภาควิชาตลอดระยะเวลาเกือบ 1 ปี ที่ให้ความช่วยเหลือสนับสนุนในเรื่องของการโปรแกรมซึ่งทำให้ปริญญานิพนธ์นี้เสร็จสมบูรณ์ได้ด้วยดี

นาย ชยานนท์ รูปประดิษฐ์

นาย ชาคริต สวัสดิรักษ์

สารบัญ

หน้าที่

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

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

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

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

ทางบริษัทไมโครซอฟต์ได้เล็งเห็นว่าแอพพลิเคชันทางด้านเกมนั้นมีผู้ใช้งานกันอย่างแพร่หลาย จึงมีแนวคิดที่จะให้ผู้พัฒนาแอพพลิเคชันทางด้านเกมและมัลติมีเดียหันมาพัฒนาบนระบบปฏิบัติการ Windows ดังนั้นบริษัทไมโครซอฟต์จึงพัฒนาไดเร็คเอ็ก (DirectX) ขึ้นมา ซึ่งเป็นไลบรารีทางด้านมัลติมีเดียที่มีจุดมุ่งหมายหลัก ๆ ดังต่อไปนี้

- ไดเร็คเอ็กประกอบด้วยไลบรารีที่ทำงานในระดับต่ำ ซึ่งทำงานได้รวดเร็วและไม่มีข้อจำกัดในการพัฒนาแอพพลิเคชันทางด้านเกม

- โครงสร้างของไดเร็คเอ็กต้องยกภาระในเรื่องฮาร์ดแวร์จากผู้พัฒนาแอพพลิเคชันไปสู่ผู้ผลิตฮาร์ดแวร์ ซึ่งผู้ผลิตฮาร์ดแวร์ต้องเป็นผู้สร้างไดรเวอร์สำหรับผลิตภัณฑ์ของตน และให้ผู้พัฒนาแอพพลิเคชันนั้นสามารถใช้ความสามารถล่าสุดที่มีอยู่ในฮาร์ดแวร์นั้นๆ ได้
- ไดเร็คเอ็กนั้นต้องอนุญาตให้ผู้พัฒนาแอพพลิเคชันสามารถพัฒนาแอพพลิเคชันออกมาในรูปแบบของ Windows application ที่สามารถทำงานบนเดสท็อปได้และสามารถทำงานร่วมกับฟังก์ชันต่างๆ ที่มีอยู่บนระบบปฏิบัติการ Windows ได้
- แอพพลิเคชันที่ถูกพัฒนาโดยใช้ไดเร็คเอ็กนั้นต้องมีประสิทธิภาพที่อยู่สูงกว่าหรืออย่างน้อยที่สุดต้องเทียบเท่ากับประสิทธิภาพของแอพพลิเคชันที่ทำงานบนระบบปฏิบัติการ MS-DOS

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

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการทำงานของไดเร็คเอ็กและนำไปใช้ในการพัฒนาเกมแอพพลิเคชัน
2. เพื่อพัฒนาชุดคำสั่งและนำไปสร้างเกมเอนจินเพื่อให้สามารถพัฒนาเกมได้สะดวกยิ่งขึ้น
3. สามารถแสดงให้เห็นถึงการนำไปใช้งานได้จริงของเกมเอนจินแบบ library ที่สร้างขึ้นในหลากหลายแนวทาง
4. เพื่อศึกษาและทดลองสร้างเกมเอนจินแบบ GUI (Graphic User Interface) ซึ่งเป็นเกมเอนจินที่จะติดต่อกับผู้ใช้ผ่านหน้าต่างที่ใช้ติดต่อโดยมีลักษณะเป็นกราฟฟิค

1.3 ขอบเขตของโครงงาน

โครงงานที่ได้จัดทำขึ้นนี้เป็นการพัฒนาซอฟท์แวร์ โดยรวบรวมชุดคำสั่งของไดเร็คเอ็กที่มีลักษณะการทำงานร่วมกันมารวมเข้าไว้ด้วยกัน ซึ่งจะได้ชุดคำสั่งที่ช่วยให้เกิดความสะดวกและรวดเร็วในการพัฒนาซอฟท์แวร์โดยเฉพาะอย่างยิ่งการพัฒนาเกม โดยเราเรียกชุดคำสั่งเหล่านี้ว่าเกมเอนจิน โดยเราจะทำการแบ่งฟังก์ชันหลักภายในเกมเอนจินออกเป็นส่วนย่อย เนื่องจากแต่ละส่วนนั้นมีความชัดเจนในการทำงานต่างกันออกไป ดังนั้นแต่ละกลุ่มจะทำการพัฒนาส่วนย่อยต่างๆ นี้แล้วนำมารวมเข้าไว้ด้วยกัน เป็นชุดคำสั่งไลบรารี เพื่อการทำงานในการพัฒนาเกมที่สมบูรณ์

เมื่อพัฒนาเกมเอนจินในระดับแรกเสร็จเรียบร้อยแล้ว จะมีการนำเกมเอนจินที่ได้นี้ไปประยุกต์และพัฒนาแอพพลิเคชันต่างๆ ซึ่งจะพัฒนาต่อเป็นเกมออนไลน์แบบ MMORPG คือผู้เล่นสามารถเล่นพร้อมกันได้ในเวลาเดียวกัน โดยจะทำการเชื่อมต่อผ่านระบบเครือข่ายอินเตอร์เน็ต

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ความรู้ทางคณิตศาสตร์และอัลกอลิทึมในการสร้างเกม เช่น การหมุนภาพ การสร้างภาพเคลื่อนไหว การใช้ทรีในการเพิ่มประสิทธิภาพในการแสดงผล
2. เทคนิคต่างๆ ที่ใช้ในการสร้างเกม 3 มิติ
3. การใช้ไดเร็คเอ็กในการสร้างแอพพลิเคชันใช้งาน และติดต่อกับอุปกรณ์มัลติมีเดีย
4. ความรู้ทางภาษา C++ ที่ใช้ในการพัฒนาหน้าต่างติดต่อผู้ใช้

1.5 วิธีในการดำเนินงาน

1. ทำการศึกษาและค้นคว้าความรู้ทางด้านการเขียนเกม 3 มิติ และไดเร็คเอ็ก จากหนังสือ และอินเตอร์เน็ต
2. ศึกษาแนวทางการพัฒนาและความรู้ที่ต้องใช้เพิ่มเติมจากความรู้พื้นฐาน
3. เลือกและศึกษาเครื่องมือที่ใช้ในการพัฒนา
4. ศึกษาโครงงานของปีการศึกษา 2545
5. ออกแบบและสร้างหน้าต่างติดต่อผู้ใช้ใหม่เพื่อให้ใช้ง่ายขึ้น
6. ทำการทดสอบเกมเอนจินที่ได้สร้างขึ้น
7. สร้างเกมตามโครงสร้างที่ได้ออกแบบไว้
8. ทดสอบการทำงานของเกม และหาข้อผิดพลาดของเกมที่พัฒนาขึ้น
9. ทำการรวบรวมผลทดสอบ วิเคราะห์ปัญหาที่เกิดขึ้น และหาแนวทางในการแก้ปัญหาเพื่อนำมาสรุปเป็นข้อมูลในการพัฒนาขั้นต่อไป

บทที่ 2
ไดเร็คเอ็ก(DirectX)

2.1 ไดเร็คเอ็กคืออะไร

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

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

ไดเร็คเอ็กแบ่งใหญ่ๆ ได้เป็นส่วนย่อยๆดังนี้

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

หรือนอกจากนี้ยังสามารถจัดแบ่ง API ของไดเร็คเอ็กอีกแบบเป็นสองส่วนใหญ่ๆ คือ

- **DirectX Foundation** ประกอบไปด้วยส่วน DirectDraw, DirectSound, Direct3D Immediate Mode และ DirectInput โดยจะดูแล function Low-Level ต่างๆ
- **DirectX Media** ประกอบไปด้วยส่วน DirectShow, DirectAnimation, DirectPlay และ Direct3D Retained Mode ซึ่งเป็นบริการระดับสูง ที่จะไปเรียกไดเร็คเอ็กFoundation อีกทีหนึ่ง

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

2.1.1 Immediate Mode และ Retained Mode

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

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

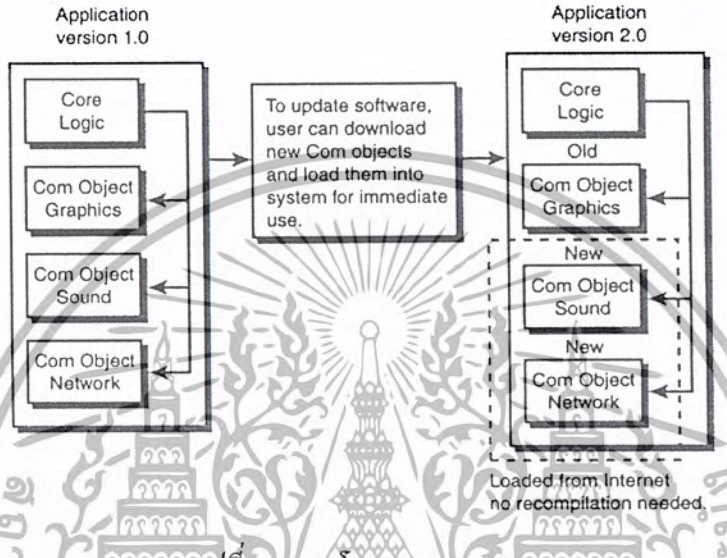
2.2 บทบาทของ COM

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

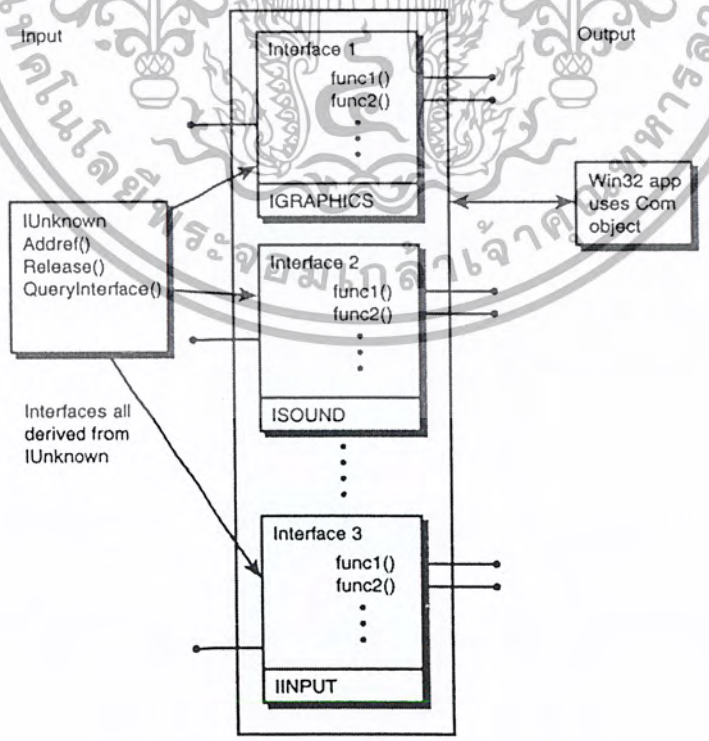
ทุกๆ COM อ็อบเจ็กต์นั้นยังจะต้องถูกสืบทอดมาจาก COM อินเตอร์เฟซมาตรฐาน ซึ่งเรียกว่า IUnknown โดย parent อินเตอร์เฟซจะมีอยู่ด้วยกัน 3 เมธอดคือ AddRef, Delete และ QueryInterface ซึ่ง 2 เมธอด แรกจะจัดการอ็อบเจ็กต์ด้วยการอ้างอิงถึงจำนวน โดยการนับจำนวนของอินสแตนซ์ตั้งแต่เริ่มต้น ส่วนเมธอดสุดท้ายจะจัดการเกี่ยวกับการสอบถามกับ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

COM อ็อบเจ็กต์ที่ทำการร้องขอนั้น เรียกว่า COM ไคลเอนต์และอ็อบเจ็กต์ที่ตอบสนองการร้องขอนั้นคือ COM เซิร์ฟเวอร์ เมื่อไคลเอนต์ได้รับการเชื่อมต่อแล้วไคลเอนต์นั้นสามารถที่จะเรียกเมธอดใดก็

ได้ใน COM เซิร์ฟเวอร์นั้น โดยใช้การอ้างอิงของอินเตอร์เฟซพอยเตอร์ ส่วนการเชื่อมต่ออื่นๆ จะออกไปเพื่อทำให้งานของอ็อบเจ็กต์ที่เรียกเพื่อเข้ามาใช้งานง่ายขึ้น ในการเชื่อมต่อนี้จะมีเมธอดที่ทำการแสดงรายชื่อของเมธอดที่สนับสนุน เพื่อที่จะช่วยให้เมธอดสามารถเรียกใช้งานได้ ก่อนที่จะรวมเข้าไว้ด้วยกัน ด้วยเทคนิคนี้จะอนุญาตให้สามารถสร้างชนิดของไลบรารีใดๆ และแอพพลิเคชัน เช่น Visual Basic ซึ่งมีข้อดีในการสร้างสมบัติของ COM โปรแกรมที่มีความทนทานต่อความผิดพลาด



รูปที่ 2-1 ภาพโดยรวมของ COM



รูปที่ 2-2 อินเตอร์เฟซของ COM อ็อบเจ็กต์

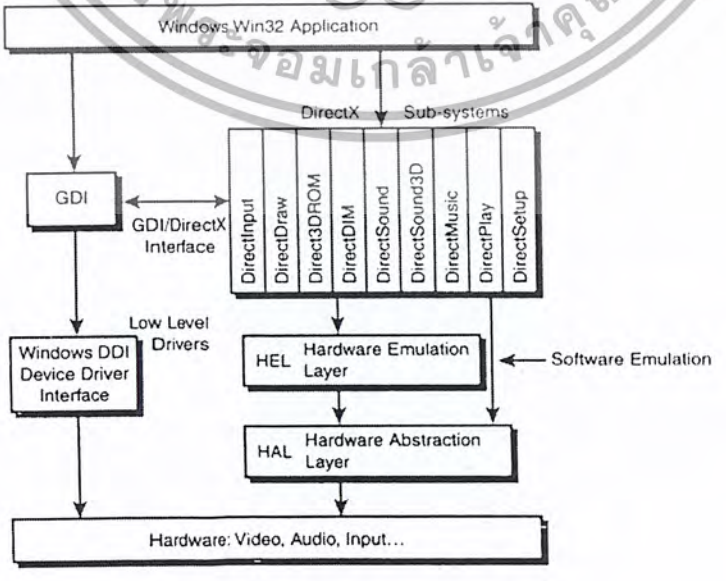
2.3 COM และไดเร็คเอ็กซ์

ทุกๆ อินเตอร์เฟซของไดเร็คเอ็กซ์นั้นจะทำการสืบทอดมาจาก IUnknown และมีพื้นฐานในการพัฒนาจาก COM ด้วยเช่นกัน การใช้ COM ของไมโครซอฟท์ทำให้ได้ชุดคอมโพเน็นต์ของไดเร็คเอ็กซ์ที่มีประสิทธิภาพเพิ่มขึ้น ซึ่งสามารถเข้ากันได้กับไดเร็คเอ็กซ์เวอร์ชั่นก่อนๆ ได้อย่างสมบูรณ์ ด้วยความอิสระไม่ขึ้นกับภาษาที่ใช้เขียนโปรแกรมของ COM ทำให้นักพัฒนาสามารถที่จะตัดสินใจเลือกภาษาที่ใช้เขียนโปรแกรมตามที่เหมาะสมกับนักพัฒนาเองมากที่สุด ซึ่งอาจเป็นภาษาที่ถนัด และยังทำให้ได้ประสิทธิภาพตามที่ต้องการ นอกจากนั้นนักพัฒนายังสามารถที่จะเลือกภาษาอะไรก็ตามที่ COM สนับสนุน และสามารถพัฒนาสภาพแวดล้อมได้ตามที่นักพัฒนาต้องการ ด้วยโปรแกรมมิ่งโมเดลของ COM โปรแกรมเมอร์จะแน่ใจได้ว่าแม้ว่าผู้ใช้จะอัพเกรดหรือเปลี่ยนติดตั้งฮาร์ดแวร์ใหม่ หรือจะเปลี่ยนไดเร็คเอ็กเวอร์ชั่นใหม่ก็ตาม ซอฟต์แวร์ก็ยังคงจะสามารถใช้งานได้ตามปกติ โดยไม่จำเป็นต้องทำการติดตั้งใหม่

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

2.4 สถาปัตยกรรมของไดเร็คเอ็ก

ไดเร็คเอ็กได้สร้างขึ้นมาบนพื้นฐานของคอมโพเน็นต์ โดยใช้โครงสร้างของ HAL (Hardware Abstraction Layer) ซึ่งสามารถซ่อนลักษณะเฉพาะของดีไวซ์ (Device) ที่เกี่ยวพันอยู่กับฮาร์ดแวร์ และเนื่องจากไดเร็คเอ็กได้รับการออกแบบให้มีความสามารถในการพัฒนาได้เพิ่มมากขึ้นในอนาคต ดังนั้นจึงสามารถรับรองความสามารถของฮาร์ดแวร์เร่งความเร็วชนิดใหม่ที่เข้ามา โดยใช้การทำงานผ่าน HEL (Hardware Emulation Layer)



รูปที่ 2-3 สถาปัตยกรรมของไดเร็คเอ็ก

2.4.1 HAL (Hardware Abstraction Layer)

HAL เป็นส่วนที่อยู่ล่างสุดของไดเร็คเอ็กซึ่งประกอบไปด้วยส่วนควบคุมของฮาร์ดแวร์ที่ถูกทำโดยผู้ผลิตฮาร์ดแวร์ ที่จะควบคุมฮาร์ดแวร์โดยตรง โดยชั้นนี้จะให้ประสิทธิภาพอย่างมาก เพราะว่าสามารถที่จะติดต่อกับฮาร์ดแวร์ได้โดยตรง ในการทำงานจริงจะไม่สามารถติดต่อกับ HAL ได้เอง แต่ไดเร็คเอ็กจะทำการจัดการให้โดยอัตโนมัติ

2.4.2 HEL (Hardware Abstraction Layer)

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

ไดเร็คเอ็กไดรเวอร์นี้จะรวมเข้าด้วยกันกับไดเร็คเอ็กAPI ผ่านลำดับของบัฟเฟอร์อ็อบเจ็กต์ ผลลัพธ์ส่วนมากจะถูกสร้างโดยคอมโพเน็นต์ ไดเร็คเอ็กAPI จะถูกเขียนขึ้นเพื่อตอบสนองบัฟเฟอร์อ็อบเจ็กต์ โดยบัฟเฟอร์นี้จะถูกจัดการ และถูกทำให้มีประสิทธิภาพขึ้นตามฮาร์ดแวร์ที่มีอยู่ โดย 2 เลเยอร์ของสถาปัตยกรรมของไดเร็คเอ็กและแปลงไปยังเอาท์พุตดีไวซ์ที่เหมาะสมอย่างเป็นระเบียบ ในระดับของภาษาที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้

สมบัติของการจำลองการทำงานของฮาร์ดแวร์ ก็คือชุดของฟังก์ชันที่ทำการจำลองคำสั่งทาง 3 มิติโดยใช้ซอฟต์แวร์ ถ้าฟังก์ชันที่ต้องการโดยเกมนั้น ไม่สามารถที่จะเข้าถึงฮาร์ดแวร์ได้โดยผ่าน HAL ไดเร็คเอ็กนั้นจะสลับการทำงานมายัง HEL เพื่อที่จะจำลองการทำงานโดยใช้ซอฟต์แวร์แทน ด้วยสมบัตินี้จะช่วยให้นักพัฒนาสามารถที่จะทดสอบหรือตรวจสอบสมบัติทาง 3 มิติในโหมดของซอฟแวร์ได้ด้วยตนเอง

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

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

ดังนั้นเหตุผลหลักของข้อดีเหล่านี้ เป็นเพราะว่าการออกแบบสถาปัตยกรรมให้เป็น COM ซึ่งสามารถช่วยให้คอมโพเน็นต์อ็อบเจ็กต์สามารถติดต่อสื่อสารกันได้

บทที่ 3
ทฤษฎีและหลักการเกม 3 มิติ

ในการพัฒนาโปรแกรมเกม 3 มิติ จำเป็นอย่างยิ่งที่จะต้องมีความเข้าใจถึงทฤษฎีพื้นฐานที่จำเป็นสำหรับนำไปประยุกต์ใช้ โดยเฉพาะทฤษฎีทางด้านคอมพิวเตอร์กราฟิก ซึ่งเป็นพื้นฐานสำหรับการพัฒนาโปรแกรมเกม 3 มิติ

3.1 ระบบพิกัด 3 มิติ

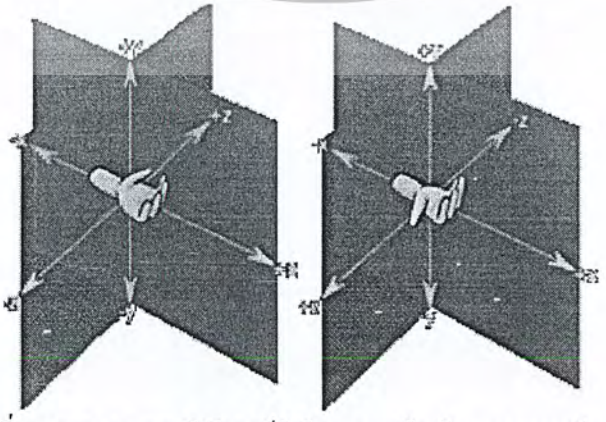
หมายถึงทิศทางของค่าในแนวแกน x, y และ z ของจุดกำเนิด (Origin) ในระบบสามมิติ ซึ่งจุดกำเนิดจะมีค่า x, y และ z เป็น 0, 0 และ 0 ตามลำดับ และโดยปกติแล้วทิศทางของค่า x จะเพิ่มขึ้นในทิศทางไปทางด้านขวา และลดลงไปทางด้านซ้ายของจุดกำเนิด ส่วนค่า y จะเพิ่มขึ้นไปทางด้านบน และลดลงไปทางด้านล่าง แต่ค่า z จะมีให้เลือกอยู่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

3.1.1 ระบบพิกัดคาร์ทีเซียน (CartesianSystem)

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

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

โดยแสดงได้ดังภาพดังต่อไปนี้

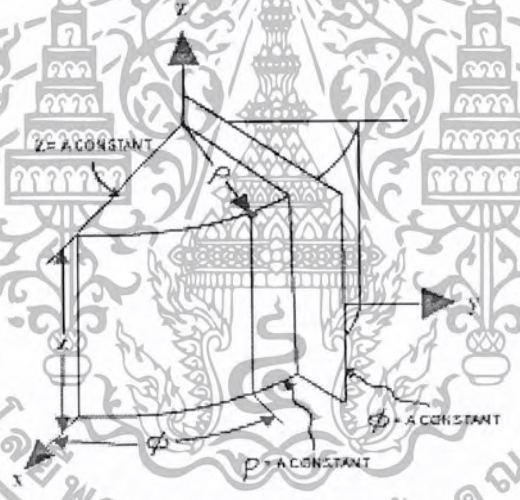


รูปที่ 3-1 แสดงระบบพิกัดคาร์ทีเซียนแบบมือซ้ายและแบบมือขวา

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

3.1.2 ระบบพิกัดทรงกระบอก (CylindricalSystem)

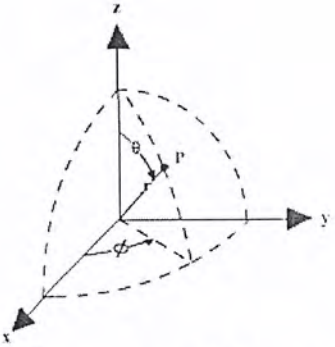
ในระบบพิกัดนี้ สามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด (Origin) ซึ่งแทนด้วยสัญลักษณ์ ρ มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเซียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแกนอ้างอิงในแนวระดับ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



รูปที่ 3-2 แสดงระบบพิกัดทรงกระบอก

3.1.3 ระบบพิกัดทรงกลม (SphericalSystem)

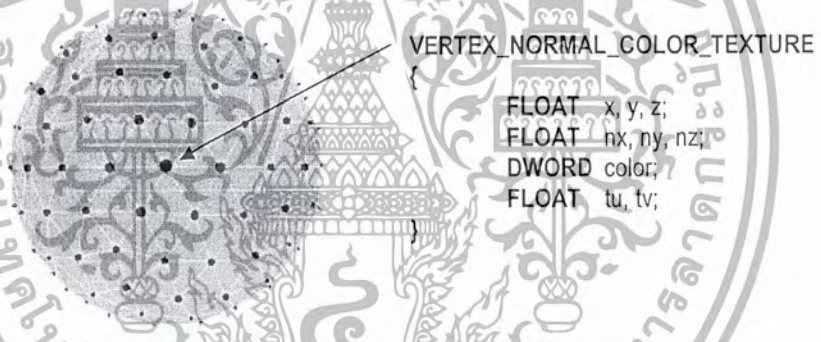
ในระบบพิกัดนี้ สามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด ซึ่งแทนด้วยสัญลักษณ์ r มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเซียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแนวระดับ (ซึ่งสัมพันธ์กับระนาบ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



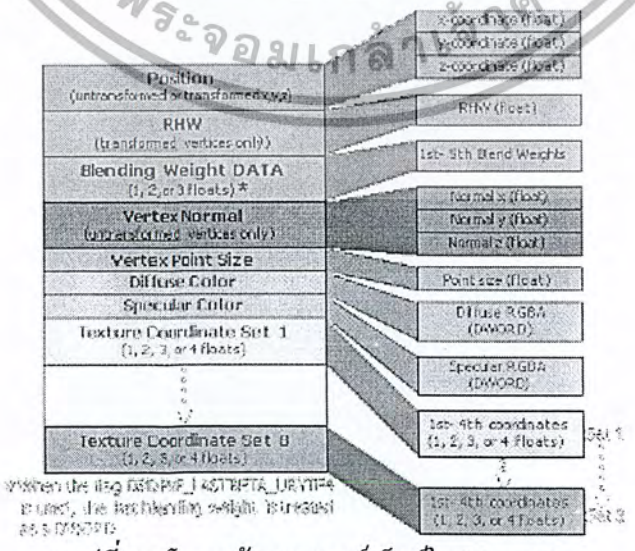
รูปที่ 3-3 แสดงระบบพิกัดทรงกลม

3.2 เวอร์เท็กซ์ (Vertex)

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



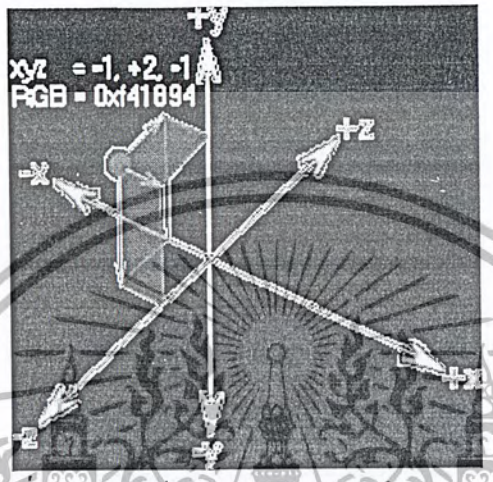
รูปที่ 3-4 แสดงตัวอย่างการนำเวอร์เท็กซ์มาใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติ



รูปที่ 3-5 โครงสร้างของเวอร์เท็กซ์ใน Direct3D

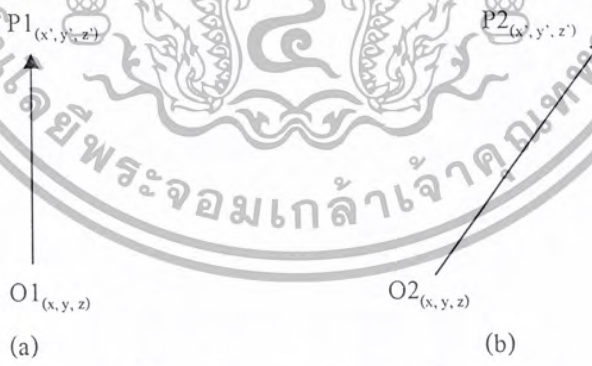
3.3 เวกเตอร์ (Vector)

เวกเตอร์เป็นปริมาณที่มีทั้งขนาดและทิศทาง เป็นองค์ประกอบทางคณิตศาสตร์ที่มีความสำคัญมากด้านการประยุกต์ใช้ในงานการประมวลผลกราฟิก ซึ่งจะต่างกับปริมาณสเกลาร์ตรงที่เวกเตอร์นั้นจะมีทิศทาง หรือกล่าวได้อีกอย่างหนึ่งว่าเวกเตอร์มีองค์ประกอบมาจากปริมาณสเกลาร์ n ตัว เพื่อแทนขนาดและทิศทางในระบบ n มิติ



รูปที่ 3-6 เวกเตอร์ของ RGB และ XYZ ใน Direct3D

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



รูปที่ 3-7 แสดงตัวอย่างเวกเตอร์

หากทำการบวกเวกเตอร์ทั้งสองจะได้ค่าดังนี้

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

เมื่อทำการคูณเวกเตอร์ด้วยปริมาณสเกลาร์ ซึ่งทำได้โดยการคูณปริมาณสเกลาร์กับทุกคอมโพเน็นต์ของเวกเตอร์นั้น ๆ จะได้

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

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

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

เวกเตอร์ที่ขนานกัน (Parallel Vector) หมายถึง เวคเตอร์คู่ใด ๆ ที่มีทิศทางเดียวกันหรือทิศทางตรงข้ามกัน

เวกเตอร์ร่วมระนาบ (Coplanar Vector) หมายถึง เวกเตอร์ตั้งแต่ 3 ตัวขึ้นไปที่อยู่ในระนาบเดียวกัน

3.4 เมทริกซ์และทรานส์ฟอร์มเมชัน 3 มิติ

3.4.1 เมทริกซ์ (Matrix)

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

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

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

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

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

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

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

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 $p \neq q$

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

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

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

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

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

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

รูปที่ 3-8 Translation Matrix

3.4.3 Rotation

เป็นการหมุนวัตถุรอบแกน X, Y หรือ Z ซึ่งใช้ Rotation Matrix

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

รูปที่ 3-9 Rotation around X Axis Matrix

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

รูปที่ 3-10 Rotation around Y Axis Matrix

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

รูปที่ 3-11 Rotation around Z Axis Matrix

3.4.4 Scaling

เป็นการย่อหรือขยายวัตถุ ซึ่งใช้ Scaling Matrix

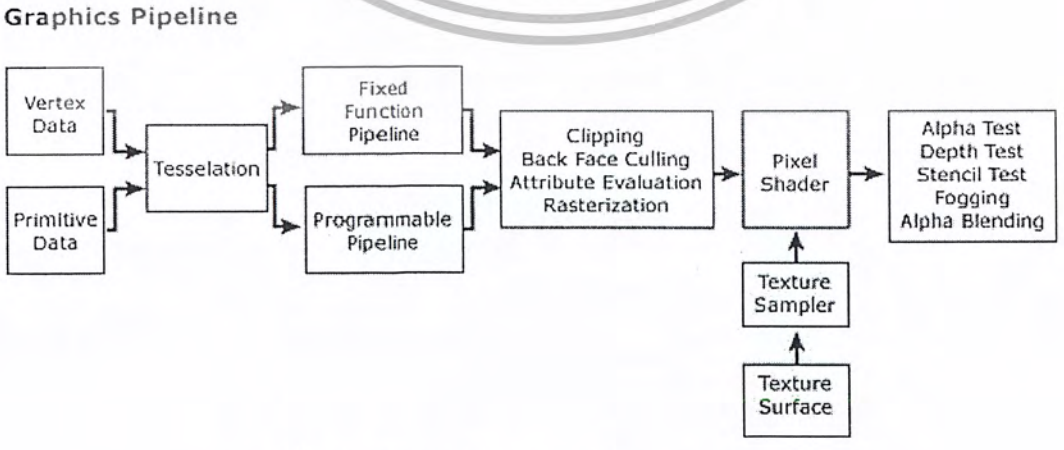
$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

รูปที่ 3-12 Scaling Matrix

3.5 การแสดงผลภาพ 3 มิติด้วย Direct3D

3.5.1 สถาปัตยกรรมของ Direct3D

สถาปัตยกรรมของ Direct3D ประกอบด้วยหน้าที่การทำงานต่างๆ ซึ่งจะรวมกันทำงานแบบ Pipeline ดังที่แสดงตามรูป



รูปที่ 3-13 รูปแสดง Pipeline การทำงานของ Direct3D

3.5.2 ทรานส์ฟอร์มเมชัน

3.5.2.1 การแปรไปสู่พิกัดเวิลด์ (World Transformation)

ขั้นตอนนี้จะเป็นขั้นตอนในการแปลง Localไปเป็น Worldซึ่งจะเป็นตำแหน่งจริงๆ ในปริภูมิ 3 มิติ ซึ่งวิธีการนั้นจะใช้การคูณตำแหน่งในเวอร์เท็กซ์ด้วย Transformationเมทริกต่าง ๆ เพื่อให้ได้ตำแหน่งที่ต้องการ

ในการแปลงเวอร์เท็กซ์จาก Localให้เป็น Worldนั้นในบางครั้งถ้าการแปลงของมีความซับซ้อนมากอาจจะต้องใช้เมทริกซ์หลายตัวในการแปลงคูณต่อกันไป (Matrix Concatenation) เพื่อสร้างเมทริกซ์สุดท้ายสำหรับใช้เป็น Transformationเมทริกก่อนแล้วจึงจะเอามาคูณกับตำแหน่งในเวอร์เท็กซ์จริงเพื่อให้ได้ผลลัพธ์สุดท้ายก็ได้

การคูณเมทริกกับตำแหน่งของเวอร์เท็กจะเป็นดังต่อไปนี้

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

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

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

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

รูปที่ 3-14 การคูณเมทริกกับตำแหน่งเวอร์เท็ก

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

3.5.2.2 การแปรไปสู่พิกัดวิว (View Transformation)

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

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

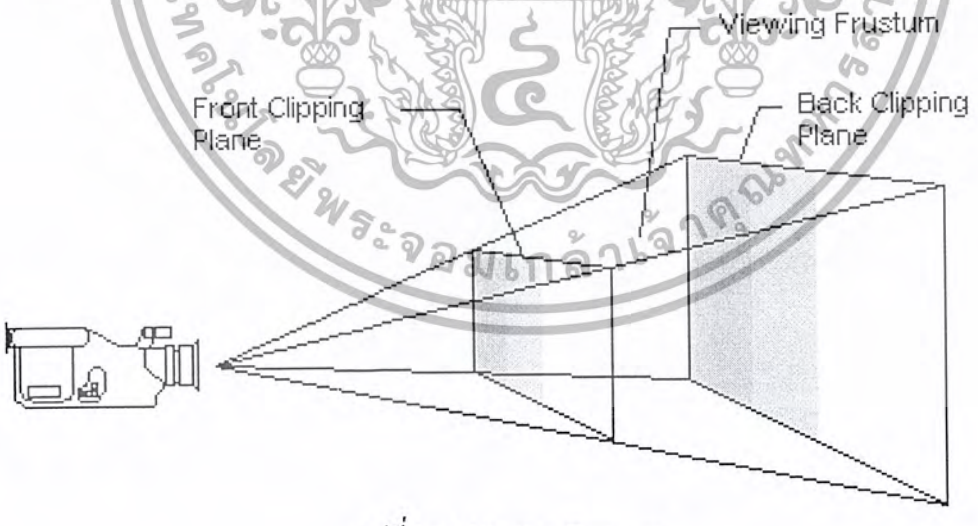
รูปที่ 3-15 การแปรไปสู่พิกัดวิว

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

วิวเมทริกซ์นั้นให้ข้อมูลที่จำเป็นในการปรับมุมมองของกล้องในเกมเอนจินได้ ซึ่งใช้ในการบอกตำแหน่งและทิศทางของกล้อง

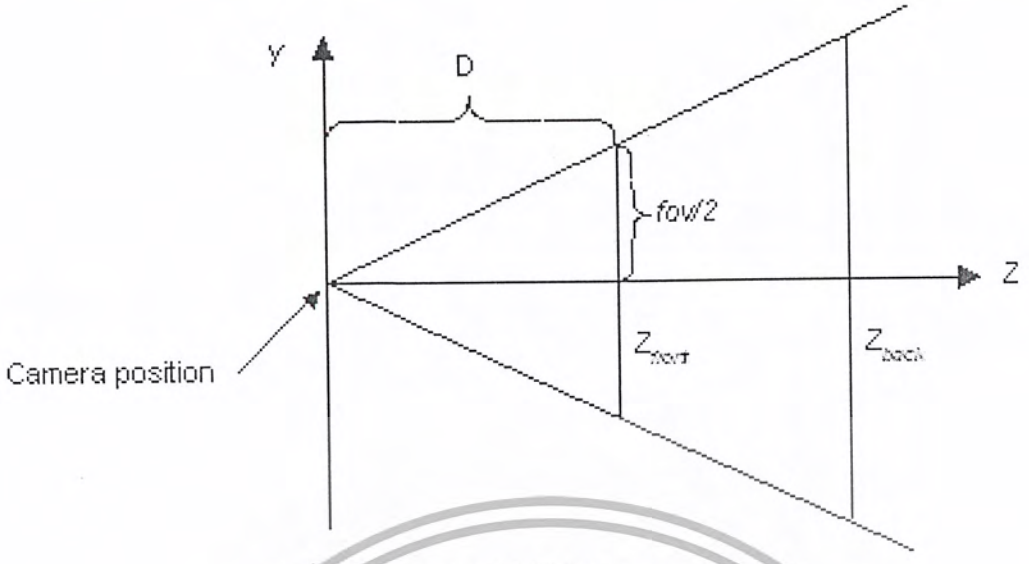
3.5.2.3 การแปรไปสู่พิกัดโปรเจ็กต์ชัน (Projection Transformation)

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

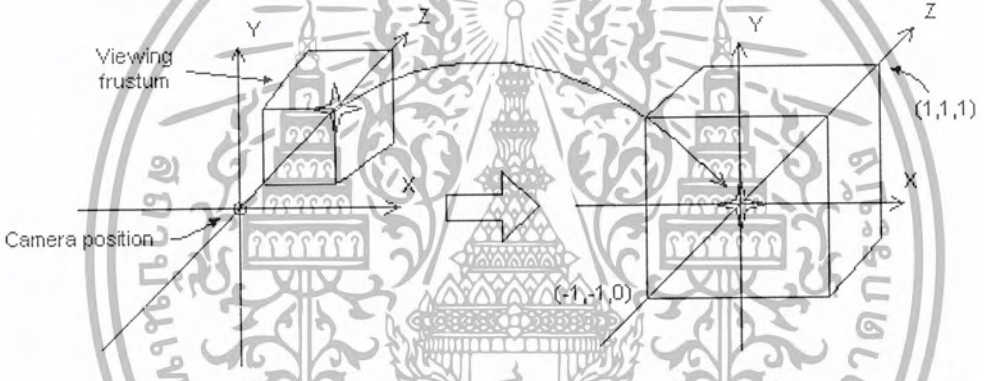


รูปที่ 3-16 Viewing Frustum

วัตถุที่อยู่ใน Viewing Frustum จะถูกนำมาแสดงผล เนื่องจากเป็นส่วนที่จอภาพสามารถเห็นได้ ส่วนวัตถุที่อยู่ภายนอก Viewing Frustum จะถูกตัดออกระหว่างการประมวลผลก่อนที่จะนำไปแสดงผล



รูปที่ 3-17 Viewing Frustum มองจากแนวแกน X



รูปที่ 3-18 การแปลงจาก Viewing Frustum (จากภาพด้านซ้าย) ไปเป็น Cuboid shape (จากภาพด้านขวา) ซึ่งทำให้วัตถุที่อยู่ไกลมีขนาดเล็กลง และวัตถุที่อยู่ใกล้มีขนาดใหญ่ขึ้นตามสัดส่วนของระยะห่างจากสายตาผู้สังเกต

สามารถทำ Perspective Projection Transformation ได้โดยการใช้ Perspective Projectionเมทริก ดังนี้

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

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

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

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

รูปที่ 3-19 Perspective Projection Metrix

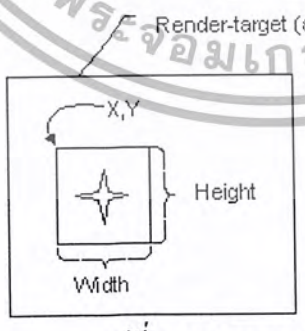
โดยทั่วไปแล้วกราฟิกเอนจินนั้นมักจะมีฟังก์ชันที่ใช้สำหรับทำการคำนวณหา Perspective Projection Matrix โดยให้กำหนดค่า Field of View (fov) ตามแนวความสูงของฉาก (โดยทั่วไปมักเป็นแกน Y) ค่า Aspect Ratio (w/h) ค่า Near Clipping Plane (Z_n) และค่า Far Clipping Plane (Z_f)

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

3.5.3 Clipping และ Viewport Scaling

Viewport ในที่นี้หมายถึงพื้นที่สี่เหลี่ยมผืนผ้าที่ใช้ในการฉายภาพที่เกิดจากการแปรไปสู่พิกัดโปรเจ็กต์ชัน ซึ่งเป็นขั้นตอนสุดท้ายใน Geometry Pipeline และทำให้ได้ภาพฉายตกลงบน Viewport เพื่อทำกระบวนการ Rasterization ต่อไปในการสร้างภาพเสมือนจริงบนหน้าจอ

Viewport อาจหมายถึงหน้าจอทั้งหน้าจอหรือบางส่วนของหน้าจอก็ได้ ทั้งนี้ขึ้นอยู่กับโปรแกรม ซึ่งโดยทั่วไปแล้วเกมมักจะแสดงผลแบบเต็มจอ แต่การใช้ Viewport อาจมีประโยชน์เมื่อต้องการการแสดงผล 3 มิติหลายหน้าต่าง



รูปที่ 3-20 Direct3D Viewport

โดยทั่วไปแล้วจะทำการขริบ (Clipping) สิ่งที่มองไม่เห็นบนหน้าจอออกไป ในกรณีที่ Viewport แสดงถึงบางส่วนของหน้าจอนั้น จะขริบส่วนของภาพที่อยู่นอกกรอบ Viewport ออกไป ทั้งนี้รวมถึงการ

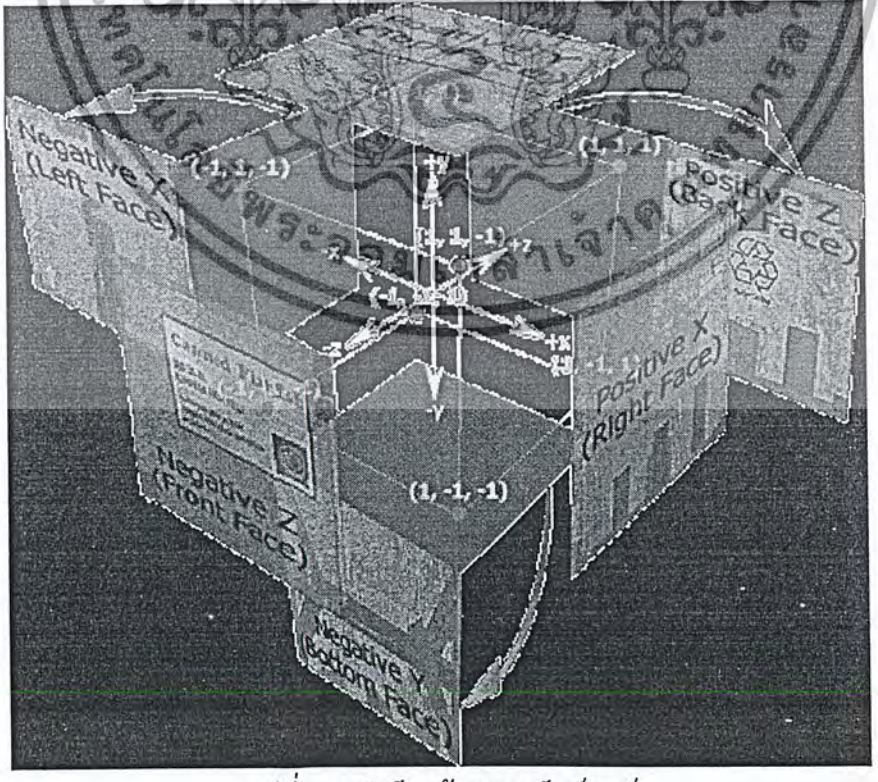
ขริบในแนวแกน Z ด้วย (จะตัดทอนส่วนของภาพที่อยู่ไกลเกิน Viewport ที่กำหนด หรือ ส่วนของภาพที่อยู่ด้านหลัง Near Clipping Plane ของ Viewing Frustum ออก)

กระบวนการนี้โดยมากมักจัดการโดยกราฟิกเอนจินโดยกำหนดค่าขอบเขตของ Viewport และระยะการตัดทอนตามแนวแกน Z ก็เพียงพอแล้ว

3.6 สเปซ

3.6.1 โมเดลสเปซ

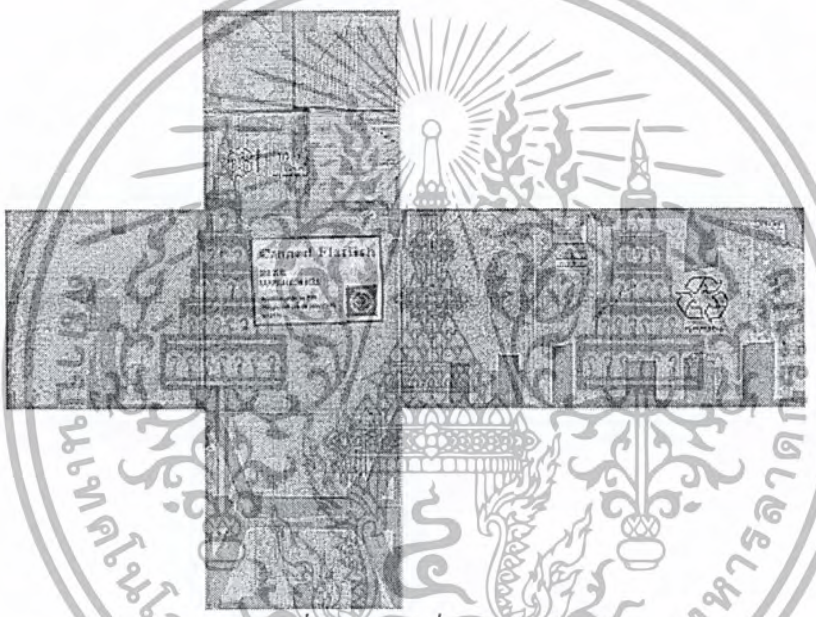
โมเดลสเปซ คือพิกัด (3D Coordinate) ที่ใช้ภายในโมเดลหรือวัตถุแต่ละอัน ซึ่งมีขอบเขตเป็นสี่เหลี่ยม โมเดลจะประกอบขึ้นมาจากรูปสามเหลี่ยมหลายๆ ชิ้น และรูปสามเหลี่ยมแต่ละชิ้นก็เกิดขึ้นจากเวอร์เท็กจำนวน 3 จุด โดยแต่ละจุดก็จะอ้างอิงถึงจุดกำเนิด (Origin) จุดเดียวกัน (จุดกำเนิดจะมีค่า x, y, x เป็น 0, 0, 0) ซึ่งปกติแล้ว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



รูปที่ 3-21 คำเรียกด้านของเท็กซ์เจอร์

1. ด้านที่อยู่ในแกน x ทางด้านบวก จะเรียกว่า ด้าน Positive X หรือด้านขวา (Right Face)
2. ด้านที่อยู่ในแกน x ทางด้านลบ จะเรียกว่า ด้าน Negative X หรือด้านซ้าย (Left Face)
3. ด้านที่อยู่ในแกน y ทางด้านบวก จะเรียกว่า ด้าน Positive Y หรือด้านบน (Top Face)
4. ด้านที่อยู่ในแกน y ทางด้านลบ จะเรียกว่า ด้าน Negative Y หรือด้านล่าง (Bottom Face)
5. ด้านที่อยู่ในแกน z ทางด้านบวก จะเรียกว่า ด้าน Positive Z หรือด้านหลัง (Back Face)
6. ด้านที่อยู่ในแกน z ทางด้านลบ จะเรียกว่า ด้าน Negative Z หรือด้านหน้า (Front Face)

และด้านล่างนี้เป็นภาพเท็กซ์เจอร์จำนวน 6 ภาพ ซึ่งแต่ละภาพมีขนาด 128 x 128 เมื่อนำมาเรียงต่อกันตามรูปแบบข้างบน ก็จะดูเหมือนกล่องกระดาษที่ถูกคลี่ออกมา หรือดูเป็นกระดาษที่ยังไม่ได้ถูกพับให้เป็นกล่อง



รูปที่ 3-22 การคลี่เท็กซ์เจอร์

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

เกม 3D จะมีการระบุว่าต้องการการ์ด 3D ที่มี RAM เท่าไรเป็นอย่างต่ำ เพื่อที่จะได้รองรับขนาดเท็กซ์เจอร์สูงสุดที่ใช้ในการเขียนเกมนั้นๆ ได้ เพราะการ Map รูปภาพเท็กซ์เจอร์ลงไปบนโมเดล จะต้องมีการ Copy รูปภาพจากหน่วยความจำปกติ ไปไว้ในหน่วยความจำของการ์ด 3D ถ้ามีการเปลี่ยนเท็กซ์เจอร์อันใหม่ ก็จะต้อง Copy ลงไปใหม่ ดังนั้นถ้าโปรแกรมที่เขียนมีการเปลี่ยนเท็กซ์เจอร์เพื่อใช้ในการ Map บ่อยๆ จะทำให้ Frame Rate ลดลง วิธีที่ถูกก็คือ ควรวาดภาพโดยเรียงตามเท็กซ์เจอร์ที่ใช้ คือ Group รูป

สามเหลี่ยมหรืออาจจะ Group จุดเวอร์เท็กที่ใช้เท็กซ์เจอร์เดียวกันเข้าด้วยกัน ตัวอย่างเช่น ถ้าต้องการวาดกล่องสี่เหลี่ยมที่เหมือนกันจำนวน 10 กล่องลงไปในฉาก โดยแต่ละกล่องมีการใช้เท็กซ์เจอร์จำนวน 6 ภาพที่ต่างกันสำหรับใช้ Map ลงไปที่ด้านแต่ละด้านของกล่อง ถ้าใช้วิธี Group ตามจำนวนกล่อง หรือวาดทีละกล่อง คือวาดกล่องแรกจนเสร็จ แล้ววาดกล่องต่อๆ ไป ซึ่งการทำแบบนี้จะต้องมีการเปลี่ยนเท็กซ์เจอร์ถึง 60 ครั้ง (ทำวนรอบนับจำนวนกล่อง 10 รอบ และแต่ละรอบมีวรรอบเปลี่ยนเท็กซ์เจอร์อีก 6 รอบ รวมจำนวนรอบของ Loop ทั้งหมดเป็น 10 x 6 ซึ่งเท่ากับ 60)

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

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

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

สมมุติว่าใช้เวอร์เท็กจำนวน 6 จุด ในการวาด Primitive แต่ละแบบ โดยเวอร์เท็กจะเรียงกันดังนี้

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

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

รูปที่ 3-23 Point List

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

รูปที่ 3-24 Line List

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

รูปที่ 3-25 Line Strip

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

รูปที่ 3-26 Triangle List

5. Triangle Strip จะวาดสามเหลี่ยมเรียงตามลำดับการเรียงของเวอร์เท็กโดยเวอร์เท็กที่ 2 และ 3 ของรูปสามเหลี่ยมอันที่เพิ่งวาดไป จะกลายเป็นเวอร์เท็กที่ 1 และ 2 ของรูปสามเหลี่ยมรูปถัดไป คือใช้เวอร์เท็กร่วมกัน 2 จุด ผลลัพธ์ที่ได้ก็คือ รูปสามเหลี่ยมที่ต่อกันไปเรื่อยๆ (ถ้าจะใช้วิธีนี้ ควรจะเรียงเวอร์เท็กแบบซิกแซ็ก) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLESTRIP

รูปที่ 3-27 Triangle Strip

6. Triangle Fan จะวาดสามเหลี่ยมเรียงตามลำดับการเรียงของเวอร์เท็กโดย มีเวอร์เท็กแรกสุดเป็นเวอร์เท็กที่ 1 ของสามเหลี่ยมทุกรูป และเวอร์เท็กที่ 3 ของรูปสามเหลี่ยมอันที่เพิ่งวาดไป จะกลายเป็นเวอร์เท็กที่ 2 ของรูปสามเหลี่ยมรูปถัดไป ผลลัพธ์ก็คือ จะได้รูปสามเหลี่ยมที่เรียงตัวต่อกันเป็นรูปที่คล้ายพัด ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLEFAN

รูปที่ 3-28 Triangle Fan

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

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

3.6.2 World Space

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

3.6.3 Camera Space

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

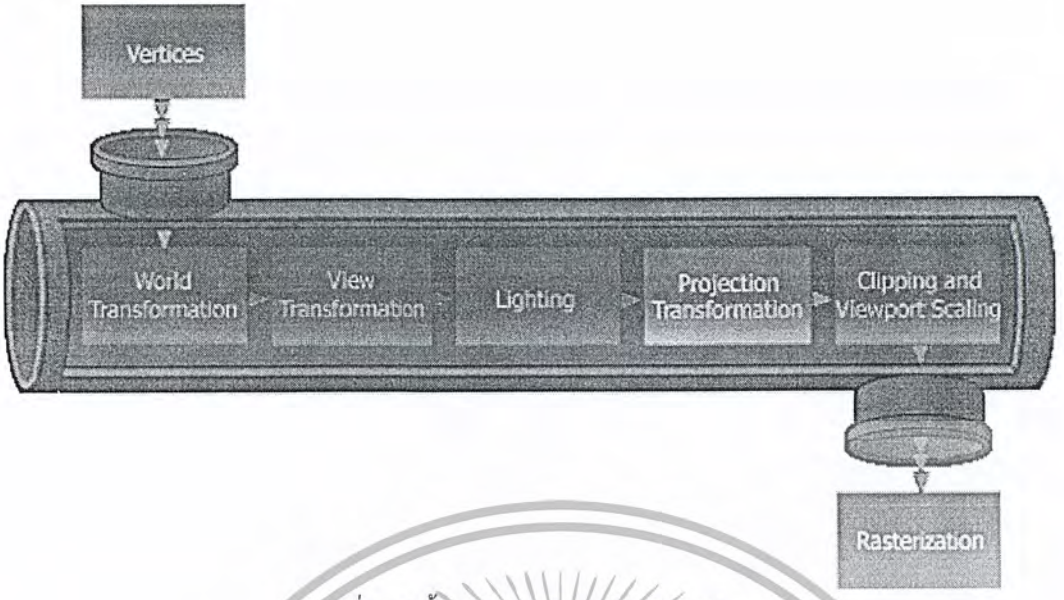
3.6.4 Projection Space

Screen Space คือพื้นที่สองมิติแบนๆ แบบภาพที่อยู่บนจอภาพ โดยพื้นที่นี้จะถูกแปลงมาจาก Projection Space เพื่อเตรียมเอาไปแสดงให้เห็นบนจอภาพ

3.6.5 Transformation and Lighting (T&L) และ Rasterization

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

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



รูปที่ 3-29 ขั้นตอนการทำ Transfromation

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

รูปที่ 3-30 ด้านของ Draw Primitive

ขั้นสุดท้ายจะเป็นการทำ Rasterization โดยตัวที่ทำหน้าที่นี้จะเรียกว่า Rasterizer จะแปลงสิ่งที่อยู่ใน Projection Space ให้ไปอยู่ใน Screen Space คือแปลงพิกัดของ Texel ต่างๆ ที่เป็นแบบ 3 มิติ ให้เป็น 2 มิติเมื่อเทียบกับขนาดของ Back Buffer หรือจอภาพ หลังจากนี้ก็จะได้ภาพปรากฏอยู่ใน Back Buffer เพื่อที่จะได้สั่งให้ Direct3D Device Object จัดการนำข้อมูลใน Back Buffer นี้ไปสลับหรือ Copy เป็น Front Buffer เพื่อให้ภาพไปปรากฏให้เห็นบนจอภาพต่อไป

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

3.6.6 World Transformation Matrix

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

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

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

3.6.7 View Transformation Matrix

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

และกฎที่สำคัญ รวมทั้งฟังก์ชั่นที่ใช้ในการหมุน, เคลื่อนย้าย หรือปรับขนาดของมุมมองของกล้องก็เหมือนกับที่ใช้กับ World Transformation เมทริกยกเว้นเรื่องที่ว่า ต้องมีการบอกให้ Direct3D รู้ด้วยว่า จะใช้ 3DSystem แบบ Left-Hand หรือ Right-Hand โดยใช้ฟังก์ชั่น D3DXMatrixLookAtLH() หรือ D3DXMatrixLookAtRH() ตามลำดับ ซึ่งจะใช้แบบมาตรฐานของ Direct3D คือ Left-Hand สำหรับ

Parameter ที่ต้องส่งไปให้ฟังก์ชั่นนี้มีอยู่ 4 ตัว ตัวแรกเป็น Address ของเมทริกที่จะใช้เก็บผลลัพธ์ที่ได้ ตัวที่สองเป็น 3D Vector เพื่อบอกว่า ตำแหน่งของกล้องอยู่ที่ไหน

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

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

3.6.8 Projection Transformation Matrix

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

บทที่ 4
เกมเอนจิน

4.1 เกมเอนจินคืออะไร

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

4.2 รูปแบบของเกมเอนจิน

เกมเอนจินโดยทั่วไป สามารถแบ่งตามลักษณะการทำงานของเกมเอนจินได้ 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 และการใช้งานเกมเอนจินลักษณะนี้ส่วนของเกมจะมีขนาดเล็ก เพราะจะแยกส่วนของเกมเอนจินออกไปเป็นอีกส่วนนึง ซึ่งจะไม่เข้าไปรวมกับส่วนของเกม และเป็นลักษณะที่แพร่หลายในการใช้งาน

4.2.2 เกมเอนจินแบบกราฟิก (GUI Game Engine)

เกมเอนจินแบบกราฟิกนั้นจะมีลักษณะเป็นแอพพลิเคชันที่ให้ผู้พัฒนาเกมสามารถสร้างเกมได้โดยไม่จำเป็นต้องรู้ถึงวิธีในการเขียนโปรแกรม และสามารถพัฒนาเกมออกมาออกมาได้ในระยะเวลาอันสั้น ผู้พัฒนาเกมเพียงแค่กำหนดรูปแบบของเกม เนื้อเรื่องของเกมที่ต้องการ จากนั้นใช้เกมเอนจินในการกำหนดส่วนต่างๆ ของเกมให้เป็นไปตามรูปแบบที่กำหนดไว้ จากนั้นก็สามารถทำงานได้แล้ว

บทที่ 5
หลักการออกแบบและการสร้างเกมเอนจินประเภท GUI

5.1 นิยามของเกมเอนจินประเภท GUI

เกมเอนจิน ประเภท GUI (Graphic User Interface) คือ เครื่องมือที่ช่วยในการสร้างเกมที่มีส่วนติดต่อกับผู้ใช้ลักษณะเป็นกราฟฟิก ทำให้ผู้ใช้สามารถใช้งานได้ง่ายและไม่จำเป็นต้องมีความรู้ความชำนาญด้านการเขียนโปรแกรม ผู้ใช้จะสามารถทำการกำหนดประเภทของเกมที่จะสร้าง, เนื้อเรื่อง แล้วจึงเลือกเกมเอนจินประเภท GUI ที่สามารถสร้างเกมประเภทเดียวกับที่ผู้ใช้ต้องการจะสร้างเท่านั้น

5.2 นิยามของเกมประเภท RPG

เกมประเภท RPG (Role Playing Game) เป็นเกมประเภทที่ผู้เล่นต้องสวมบทบาทเป็นตัวละครภายในเกมและดำเนินตามเนื้อเรื่องของเกมที่มีการออกแบบไว้ ซึ่งภายในเกมจะมีเหตุการณ์ต่างๆ เช่น มีการพูดคุยกับ NPC (Non Player Control) ซึ่งเป็นตัวละครอื่นๆเพื่อหาข่าวสารที่เป็นประโยชน์ในการดำเนินตามเนื้อเรื่อง มีการต่อสู้ กับ NPC ประเภทศัตรู เช่น สัตว์ประหลาด (Monster) ตัวละครที่ผู้เล่นสวมบทบาทอาจมีการพัฒนาตัวเองขึ้นได้ ซึ่งอาจมาจากการที่ได้สู้กับศัตรู หรือเหตุการณ์อื่นๆตามแต่เกมนั้นจะกำหนด มีการแก้ปริศนาต่างๆจากคำบอกเล่าของ NPC ในเกม ซึ่งจะทำให้ผู้เล่นรู้สึกสนุกสนาน ไปกับการดำเนินเนื้อเรื่องและความสมจริงของเกมแนวนี้

5.3 แนวคิดของการออกแบบเกมเอนจินประเภท GUI

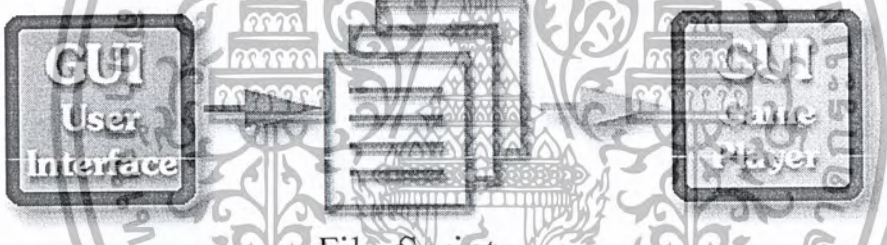
เกมเอนจินประเภท GUI นี้มีขอบเขตในการสร้างจำเพาะแนวเกมเดียว เช่นหากต้องการสร้างเกมเอนจินประเภท GUI ที่ใช้สร้างเกมแนวไล่ยิงกันแบบมุมมองบุคคลที่หนึ่ง เกมเอนจินที่สร้างออกมาจะไม่สามารถนำไปใช้สร้างเป็นเกมแนวอื่นๆได้ ในการออกแบบเกมเอนจินจึงควรคิดหาแนวเกมที่เหมาะสมในการที่จะนำมาทำเกมเอนจินประเภท GUIเป็นอันดับแรก

เกมเอนจินประเภท GUI ที่ถูกสร้างขึ้นในปริญญานิพนธ์เล่มนี้ เป็นเกมเอนจินประเภท GUI เพื่อใช้ในการสร้างเกมประเภทเกม RPG เนื่องจากเกมแนวนี้กำลังเป็นที่นิยมในปัจจุบันไม่ว่าในประเทศไทยหรือในต่างประเทศ และความสนุกของเกมประเภทนี้ คือการที่ผู้เล่นได้สวมบทบาทเข้าไปเป็น ตัวละครภายในเกมเพื่อดำเนินเนื้อเรื่อง ดังนั้นหากเกมมีเนื้อเรื่องที่ดีก็จะทำให้มีความสนุกมากยิ่งขึ้น แต่ผู้ที่ใฝ่ฝันจะสร้างเกมและมีความสามารถในการแต่งเนื้อเรื่องที่ดีก็อาจจะไม่มีความรู้ในการเขียนโปรแกรม ทำให้ไม่สามารถสร้างสรรค์เกมที่ดีออกมา ดังนั้นเกมเอนจินประเภท GUI ที่ใช้พัฒนาเกมแนวนี้น่าจะสามารถทำให้ความฝันของคนกลุ่มนี้เป็นจริงขึ้นมาได้

เกมเอนจินประเภท GUI ที่ได้ออกแบบ จะแบ่งออกได้เป็นสองส่วนใหญ่ๆ คือ

- ส่วนที่เป็นหน้าต่างติดต่อกับผู้ใช้ในการสร้างเกม (GUI User Interface) ส่วนนี้จะเป็นส่วนที่ผู้ใช้ตัว GUI ทำการสร้างฉากของเกม ตามโครงเรื่องที่ได้วางเอาไว้และจะสามารถแก้ไขค่าต่างๆ ภายในเกมได้ที่เกมเอนจินส่วนนี้เท่านั้น
- ส่วนที่ใช้เล่นเกม (GUI Game Player) ส่วนนี้จะเป็นส่วนที่ผู้ใช้นำเกมที่ได้สร้างขึ้นมาแล้วมาเล่นได้ และจะสามารถเล่นได้โดยผ่านเกมเอนจินส่วนนี้เท่านั้น

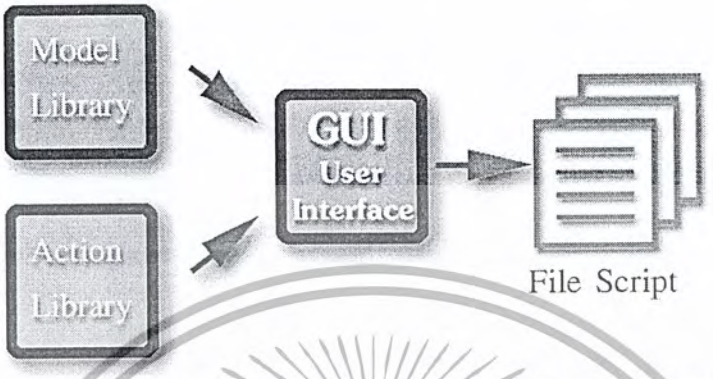
จะเห็นได้ว่า ทั้งสองส่วนแยกออกจากกัน แต่จะมีตัวกลางที่ทำให้เกมเอนจินทั้งสองส่วนสามารถติดต่อกันได้ ส่วนนี้คือ File Script โดยในส่วนของ File Script นี้จะมีหน้าที่เก็บข้อมูลเกมตามที่ผู้ใช้ได้สร้างขึ้นมา โดยเมื่อผู้ใช้ทำการสร้างเกมผ่าน GUI User Interface แล้วจะได้ File Script นี้มา และเมื่อผู้ใช้ทำการเล่นเกมที่ได้สร้างขึ้นมาผ่าน GUI Game Player ตัว GUI Game Player จะทำการเรียกข้อมูลเกม จาก File Script เพื่อประมวลแสดงผลออกมาในรูปของเกมตามที่สร้างไว้ ลักษณะการทำงานของเกมเอนจินโดยรวมดังที่ได้กล่าวมาแล้วข้างต้นจะสามารถแสดงได้ดังรูป 5-1



ภาพที่ 5-1 แสดงการทำงานโดยรวมของเกมเอนจินประเภท GUI

5.3.1 เกมเอนจินประเภท GUI ในส่วนที่เป็นหน้าต่างติดต่อผู้เล่น

ในส่วนนี้จะมีหน้าที่ รับ Input จาก Library ที่ได้กำหนดไว้ในตัวของเกมเอนจิน เพื่อมาสร้างให้เป็นเกมประเภท RPG ตามเนื้อเรื่องที่ได้กำหนดไว้ โดยจะสามารถแสดงหลักการทำงานได้ดัง รูปที่ 5-2



รูปที่ 5-2 แสดงการทำงานของเกมเอนจินประเภท GUI ส่วนที่เป็นหน้าต่างติดต่อกับผู้เล่น

การทำงานคือ โปรแกรมจะรับ Input มาจาก Library สองส่วน คือ ส่วน Model Library และ Action Library ผ่านตัว GUI User Interface แล้วจึงสร้างออกมาเป็น File Script ซึ่ง Library ทั้ง 2 ส่วนมีรายละเอียดดังนี้

5.3.1.1 Model Library

เป็น Library ที่เก็บโมเดลต่างๆที่มีอยู่ในเกม โดยจะแบ่งออกเป็น 2 ประเภท คือ

- Library ที่รวบรวมแผนที่ภายในเกม (Map Library) จะเก็บโมเดลในส่วนที่เป็นแผนที่ ซึ่งจะสามารถนำไปใช้งานในเกมได้ทั้งหมด
- Library ที่รวมรวมตัวละครต่างๆภายในเกม (Character Library) ในส่วนนี้จะเก็บโมเดลในส่วนที่เป็นตัวละครต่างๆ รวมไปถึงค่าพลังของตัวละคร ซึ่งจะสามารถนำไปใช้งานในเกมได้ทั้งหมด
- Library ที่รวบรวมสิ่งก่อสร้างต่างๆ (Environment Library) ในส่วนนี้จะเก็บโมเดลในส่วนที่เป็นสิ่งก่อสร้าง เช่น บ้าน ต้นไม้ เก้าอี้ เตียง เป็นต้น ซึ่งจะสามารถนำไปใช้งานประกอบฉากตกแต่งภายในเกมได้ทั้งหมด

5.3.1.2 Action Library

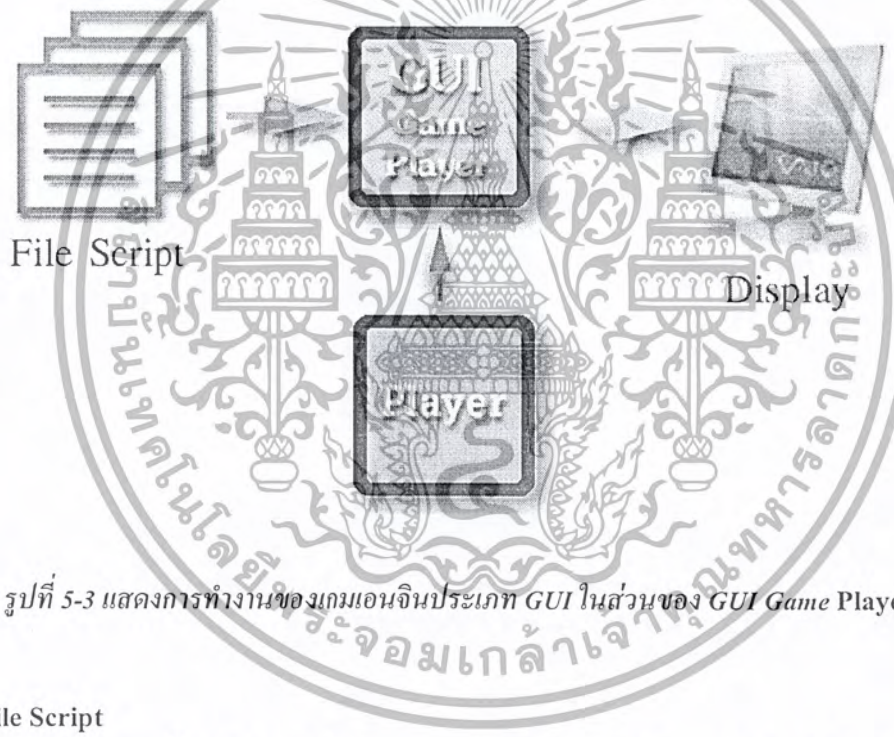
เป็น Library ที่รวบรวมคำสั่งที่เป็น Action ที่จะต้องใช้ภายในเกมส์เอาไว้เพื่อให้ผู้ใช้สามารถเลือกมาใช้กับเหตุการณ์ภายในเกมได้ เช่น Action ที่ใช้ในการพูด, Action ที่ใช้ในการฟื้นพลัง เป็นต้น

5.3.1.3 GUI User Interface

ส่วนนี้จะเป็นส่วนที่ประมวลผลจาก Input ที่ผู้ใช้ได้ทำการเลือกมาจาก Library มาจัดเก็บเอาไว้ตามโครงสร้างของแต่ละส่วนซึ่งจะแสดงที่ Class Diagram และเมื่อผู้ใช้ได้ทำการจัดเก็บ (Save) ส่วนนี้จะทำการเขียนข้อมูลลงใน File Script

5.3.2 เกมเอนจินประเภท GUI ในส่วนที่ใช้เล่นเกม

ในส่วนนี้จะทำหน้าที่ติดต่อกับ File Script ที่ได้มาจาก GUI User Interface เพื่อนำมาประมวลผลและจัดวางสิ่งต่างๆภายในเกม ที่ได้ออกแบบไว้ในเกมเอนจินในส่วนแรกแล้ว และเป็นส่วนที่ตอบสนอง Input จากผู้เล่นเกม ซึ่งได้ออกแบบให้ใช้ Mouse เป็นตัวควบคุมภายในเกมเสียเป็นส่วนใหญ่ แล้วจึงนำมาแสดงผล โดยการทำงานดังที่กล่าวมานี้จะแสดงได้ดังรูปที่ 5-3



รูปที่ 5-3 แสดงการทำงานของเกมเอนจินประเภท GUI ในส่วนของ GUI Game Player

5.3.3 File Script

ส่วนนี้เป็นส่วนที่ใช้เก็บข้อมูลที่ใช้ในการควบคุมสิ่งต่างๆภายในเกมส์ในส่วนที่ผู้ใช้ได้ทำการกำหนดไว้โดย ซึ่งในรายละเอียดของ File Script จะกล่าวถึงในหัวข้อต่อๆไป

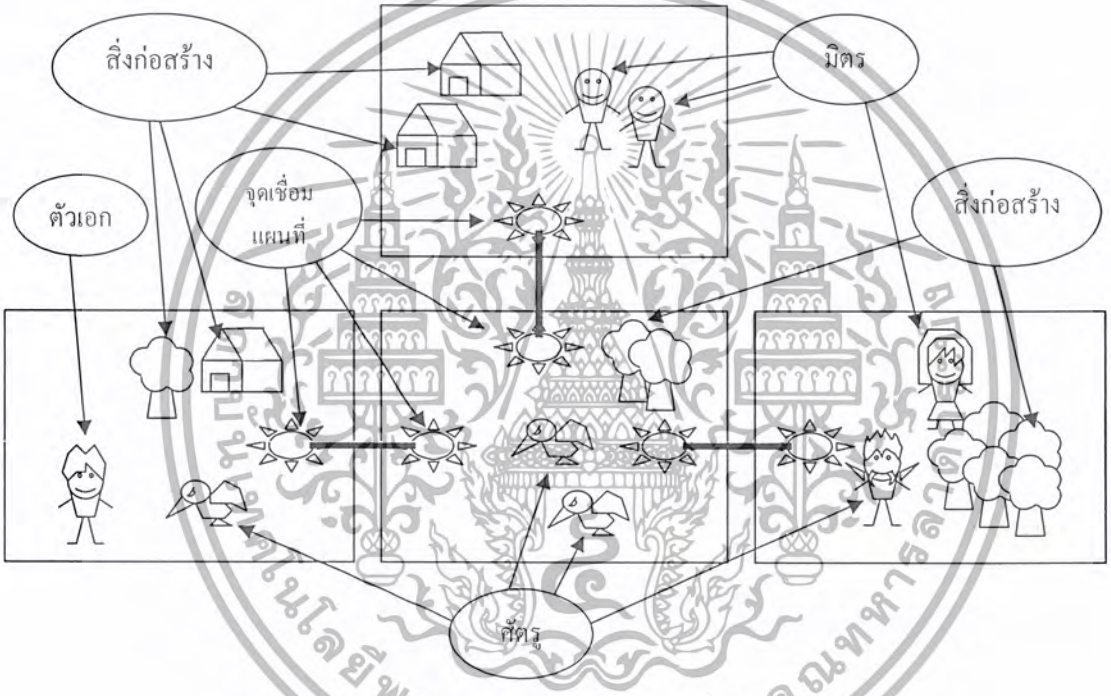
5.4 โครงสร้างและการออกแบบ GUI 3D RPG Game Engine

จากแนวคิดของการออกแบบที่กล่าวมาได้แบ่งโครงสร้างของเกมเอนจินประเภท GUI ออกเป็น 3 ส่วน ทำให้สามารถกำหนดกระบวนการออกแบบได้ดังรูปที่ 5 -4

รูปที่ 5-4 แสดงกระบวนการออกแบบและสร้าง GUI 3D RPG Game Engine

5.4.1 ขั้นตอนการกำหนดขอบเขตของเกมที่ GUI Game Engine สามารถสร้างได้

ขั้นตอนนี้จะทำการค้นคว้ารูปแบบของเกมที่เหมาะสม และสรุปออกมาเป็นผลลัพธ์ คือ ขอบเขตของเกมที่ GUI สามารถสร้างได้ ซึ่งข้อสรุปของโครงงานนี้คือ เกมในแนว RPG ซึ่งผู้เล่นต้องสวมบทบาทเป็นตัวละครตัวใดตัวหนึ่งซึ่งเป็นตัวเอก แล้วควบคุมตัวเอกนั้นดำเนินไปตามเนื้อเรื่องจนกว่าจะจบเกม โดยรูปแบบของเนื้อเรื่อง ก็มีได้หลายรูปแบบแล้วแต่ผู้สร้างเกมจะออกแบบมา แต่ลักษณะโดยพื้นฐาน ที่มีเช่น มีการพบศัตรูแล้วกำจัดเพื่อแก้ไขปริศนา หรือเพิ่มความสามารถของตัวเอก หรือมีการไปคุยกับผู้คนให้แก้ไขปริศนาหรือบอกปริศนา หรือแก้ไขปริศนาได้หมด แล้วจบเกม โดยปริศนาในแต่ละขั้นนั้น อาจจะอยู่ในฉากหรือแผนที่ซึ่งแตกต่างกันไป จะเห็นได้ว่า การออกแบบเกมแนว RPG นี้สามารถกำหนดเนื้อเรื่องได้หลากหลาย จึงต้องกำหนดขอบเขตต่างๆ แสดงให้เห็นจากภาพรวมของเกม ดังนี้



รูปที่ 5-4 แสดงความสัมพันธ์ของแผนที่และสิ่งต่างๆในเกม

ความสัมพันธ์ของสิ่งต่างๆภายในเกมจะถูกกำหนดดังนี้

- ลักษณะแผนที่ในเกมจะเป็นแผนที่ย่อยต่อๆกัน
- จุดเชื่อมต่อระหว่างแผนที่จะมีหรือไม่ก็ได้และมีเฉพาะแผนที่ที่อยู่ติดกัน

- **ตัวละคร** ในเกมที่สามารถเคลื่อนไหวและโต้ตอบกับผู้เล่นได้ สามารถนำมาประกอบใน โมเดลแผนที่เพื่อให้ฉากแต่ละฉาก มีตัวละครแตกต่างกันไป โดยแบ่งเป็นประเภทย่อยๆตามลักษณะเกมแนว RPG ดังนี้

ตัวเอก ตัวละครประเภทนี้เป็นตัวเอกในการเล่นเกม คือเป็นตัวละครที่มีได้ตัวเดียว และ เป็นตัวละครที่ผู้เล่นเกมจะใช้ บังคับเพื่อดำเนินเกม โดยตัวเอกนั้นจะมีความสามารถต่างๆติดตัว และมีค่าพลังชีวิต

ตามแต่ผู้สร้างเกมกำหนดว่าให้มีเท่าใด หรือกำหนดว่า มีอาวุธอะไรติดตัวอยู่บ้าง มีพลังในการโจมตีศัตรู ความว่องไวในการเคลื่อน ความฉลาดในการหลบหนี พลังชีวิตสามารถเพิ่มได้หรือไม่ โดยตัวเอกนี้จะไม่สร้างไว้ในเกมไม่ได้ เพราะเกมจะไม่สามารถดำเนินได้ และตัวเอกนี้หากเล่นแล้วพลังชีวิตหมด หรือโดนฆ่าตายก็คือ เกมจบ โดยหากต้องการสร้างเกมให้เนื้อเรื่องหลากหลาย ต้องกำหนดให้เลือกตัวพระเอกได้หลากหลายด้วย

ศัตรู ตัวละครประเภทนี้เป็นศัตรูกับตัวเอก คือเมื่อมีการเจอกับตัวเอกต้องมีการต่อสู้กัน โดยตัวศัตรูนั้นมีความสามารถแตกต่างกันไป แล้วแต่ผู้สร้างเกมจะกำหนดว่ามีความสามารถอย่างไร และมีการผูกปริศนาไว้กับตัวละครตัวนี้หรือไม่ เพื่อให้เนื้อเรื่องเปลี่ยนแปลงไปตามต้องการ เช่นเมื่อมีการสร้างตัวศัตรูแล้วกำหนดให้ปริศนาเกมสิ้นสุดเมื่อฆ่ามันตาย โดยตัวศัตรูนั้นก็มีพลังชีวิตเช่นเดียวกับตัวเอก และสามารถกำหนดได้ว่าเป็นเท่าใด เพื่อให้เกิดความหลากหลายในเกม เช่น เมื่อตัวพระเอกมีพลังชีวิตน้อยๆ ก็ต้องเลือกต่อสู้กับตัวศัตรูที่มีพลังชีวิตใกล้เคียงกัน โดยตัวศัตรูนั้นก็สามารถกำหนดความสามารถแต่ละอย่างได้ต่างกัน เช่น ความว่องไวในการเคลื่อนไหว ค่าพลังชีวิต เป็นต้น

มิตร ตัวละครประเภทนี้เป็นตัวละครทั่วๆไป ที่ไม่ทำร้ายตัวเอก โดยลักษณะของมิตร คือช่วยตัวเอกในการแก้ไขปริศนา เพิ่มความสามารถของตัวเอกเพื่อประโยชน์ในการแก้ไขปริศนาในขั้นต่อไป ผู้สร้างเกมสามารถนำ GUI RPG ไปกำหนดเนื้อเรื่องจาก ความหลากหลายของ ตัวละครนี้ได้

- ตัวละครอื่นๆที่ไม่ใช่ตัวเอก ซึ่งผู้เล่นไม่สามารถควบคุมได้เรียกอีกอย่างว่า NPC (None Player Control)
- ตัวละครพวก NPC สามารถเดินไปมาในฉากเองได้
- ตัวละครจะมีการระยะเวลาคอยการโจมตี ขึ้นอยู่กับค่าความว่องไวของตัวละครแต่ละตัวเมื่อโจมตีไปครั้งหนึ่งหากมีความว่องไวสูงกว่าก็จะสามารถโจมตีครั้งต่อไปได้เร็วกว่า
- ตัวเอกจะต่อสู้กับศัตรูได้และเกมจะจบถ้า ค่าพลังชีวิตเหลือ 0
- ค่าพลังของตัวละครแต่ละตัวสามารถปรับแต่งได้ สามารถกำหนดตำแหน่งและการกระทำได้

- **สิ่งแวดล้อม** ที่นำมาประกอบใน โมเดลแผนที่ เพื่อให้มีภูมิประเทศแตกต่างกัน แม้ว่าจะมีแผนที่คล้ายๆกัน เช่น โมเดลแผนที่ ที่เป็นทะเล อาจจะมีความแตกต่างของ โมเดลสิ่งแวดล้อมเช่น ทะเลแห่งแรกมีแต่โขดหิน ส่วนทะเลอีกแห่งอาจประกอบด้วยไม้ เป็นป่าโกงกาง เป็นต้น โดย แบ่งเป็น 3 ประเภท ดังนี้

สิ่งก่อสร้าง คือสิ่งก่อสร้างโดยทั่วไป ที่มีความใหญ่โต เมื่อนำมาประกอบกับ โมเดลแผนที่ แล้วทำให้ฉากในแต่ละฉากเปลี่ยนแปลงไปมาก เช่น โมเดลแผนที่ ถนน นั้นอาจจะมี ถนนสายหนึ่งประกอบด้วย ตึกสูงๆหลายๆตึกรวมกันเป็น ถนนเศรษฐกิจ ส่วนถนนอีกสายหนึ่งอาจจะประกอบด้วยบ้านคนกลายเป็นถนนในหมู่บ้าน เป็นต้น จากความหลากหลายของสิ่งก่อนสร้าง นี่เอง ทำให้เกม RPG นั้นมีภูมิประเทศที่แตกต่างกันโดยสิ้นเชิง

ต้นไม้ คือต้นไม้ เป็นส่วนประกอบฉากหนึ่งๆ ที่สร้างมาเพื่อให้ฉากนั้นๆ สวยงามขึ้นและเปลี่ยนภูมิประเทศ เป็นแนวป่าไม้ แทนเมือง ในแบบแรก เป็นต้น ทำให้ฉากแต่ละฉากในเกม ดูหลากหลาย ผู้เล่น

สามารถกำหนดเนื้อเรื่องเป็นอีกประเภทได้ เช่นจากเมืองที่มีตึกมีตัวศัตรูโจร เป็น ฉากป่าไม้ที่มีตัวศัตรูเป็นสัตว์ เป็นต้น

สิ่งแวดล้อมทั่วไป คือสิ่งก่อสร้างเล็กๆ หรือ สิ่งของจุกจิกโดยทั่วไป เช่นโต๊ะ เก้าอี้ ก้อนหิน

- **เหตุการณ์** จะเป็นสิ่งที่กำหนดเนื้อเรื่องของเกม แบ่งออกเป็น 2 ประเภท ดังนี้

เหตุการณ์ของแผนที่ เป็น Script ที่กำหนดเหตุการณ์ให้กับฉากโดยตรง เช่น เมื่อเดินเข้าไปในฉากแต่ละฉากแล้วกำหนดให้เสียงดนตรี เป็นอย่างไร หากมีเสียงจังหวะระทึกแสดงว่าเป็นฉากสำคัญที่มีการต่อสู้กับตัวศัตรูตัวสำคัญ เช่นหัวหน้าโจร หากเป็นเสียงดนตรี หรือเสียงคนมากมาย ก็แสดงว่าเป็นฉากที่มีพ่อค้าแม่ค้า หรือมีพลเมืองมากๆ หรือ หากเป็นเสียงดนตรีธรรมดา ก็อาจจะเป็นการดำเนินเรื่องทั่วไป เป็นต้น หรือกำหนดให้ บางฉากเข้าไปแล้วมีหมอกปกคลุมจนหาปริศนาลำบาก หรือมีหมอกสีแดงเพื่ออำพรางศัตรูตัวสำคัญในฉากการต่อสู้ หรือกำหนดให้ จุดเปลี่ยนฉากไปยังฉากอื่นนั้นทำงานหรือไม่ โดยอาจจะมีเงื่อนไขในการทำงานตัวอย่าง เช่น ฉากชายทะเลจะกำหนดให้จุดเปลี่ยนฉาก ทำงานคือเปลี่ยนฉากไปยัง ฉากบนเรือได้เมื่อตัวพระเอก นั้นมีเรือ หรือไปซื้อเรือมาได้จึงเปลี่ยนฉากได้ เป็นต้น หรือกำหนดว่าจะเข้าไปในฉากบ้านได้เมื่อตัวพระเอกมีสิ่งของคือกุญแจ เป็นต้น จะเห็นได้ว่า ส่วนของ เหตุการณ์แผนที่นี้สามารถนำไปสร้างเป็นปริศนาเพื่อให้เกมเนื้อเรื่องตรงตามต้องการและให้เกมจบได้ตามต้องการ เช่นให้เกมจบเมื่อตัวพระเอกหากุญแจมาเพื่อเปลี่ยนฉากไปยังฉาก ห้องลับเพื่อช่วยตัวพลเมืองได้ เป็นต้น

เหตุการณ์ของตัวละคร เป็นการกำหนดกำหนดเหตุการณ์ให้กับตัวละครโดยตรง ว่าเหตุการณ์ในแต่ละอย่างที่เกิดขึ้นมานั้น เกิดขึ้นเพราะอะไร และทำงานอย่างไร ตัวละครที่ได้รับเหตุการณ์ จะเป็นอย่างไร ตัวละครที่สร้างเหตุการณ์สร้างอย่างไร เหตุการณ์ที่เปลี่ยนแปลงความสามารถของตัวละคร เหตุการณ์ที่เกิดขึ้นเพื่อแก้ไขปริศนาของเกม เช่น

เหตุการณ์การพูดคุย ของเกม ตัวอย่างคือตัวพระเอกไปพบกับตัวพลเมืองตำรวจเพื่อสอบถามว่าตัวศัตรูโจรอยู่ที่ใดบ้าง เพื่อไปกำจัด หรือไปพบกับตัวพลเมืองพ่อค้าเพื่อสอบถามว่าสิ่งของที่ต้องการหา เช่นรองเท้าวางอยู่ที่ใดบ้าง เป็นต้น

เหตุการณ์การเก็บสิ่งของ เช่น เมื่อตัวพระเอกพบกับสิ่งของคือยารักษา ก็จะเพิ่มพลังชีวิตให้กับตัวพระเอก เป็นต้น หรือหากตัวพระเอกเก็บสิ่งของคือเสื้อเกราะก็จะเพิ่มค่าในการป้องกันการโจมตีเอาไว้

เหตุการณ์การให้สิ่งของ เช่น เมื่อตัวพระเอกมีสิ่งของคือแผนที่ แล้วตัวพระเอกนำไปให้กับตัวพลเมืองคือตำรวจ แล้วตัวตำรวจก็จะแก้ไขปริศนาให้อีกขั้น เป็นต้น

เหตุการณ์การเปลี่ยนตำแหน่งตัวละคร เช่น เมื่อตัวพระเอกได้พบกับตัวพลเมืองคนนำทาง คนนำทางก็จะพาตัวพระเอกไปยังฉาก หรือไปยังตำแหน่งที่ กำหนดเงื่อนไขเอาไว้ เป็นต้น

5.4.2 ขั้นตอนการออกแบบ Script

ดังที่ได้กล่าวมาแล้วว่าเกมเอนจินประเภท GUI ที่ได้ออกแบบมาจะมีส่วนที่ทำหน้าที่เป็นเสมือนตัวกลางเชื่อมการทำงานระหว่าง ส่วนที่เป็นหน้าต่างติดต่อกับผู้ใช้ และ ส่วนที่ใช้เล่นเกม โดยจะเป็นส่วนที่กำหนดสิ่งต่างๆภายในเกม ซึ่ง File script ที่ออกแบบมาใช้ภายในเกมเอนจิน มีทั้งหมด 9 ประเภท คือ

1. MFS (Map File Script) ทำหน้าที่เก็บข้อมูลเกี่ยวกับฉากที่ผู้ใช้ได้ทำการเลือกออกมาใช้ เช่น ได้เลือกใช้ฉากใด ใช้ Model ชื่ออะไร ใช้ Texture ชื่ออะไร เป็นต้น และเป็น File Script แรกที่จะถูกอ่าน ในเกมเอนจินส่วนที่ใช้เล่นเกม

2. Cdef (Character Definition) ทำหน้าที่เก็บข้อมูลค่าพลังต่างๆของตัวละครทั้งหมดที่ผู้ใช้เลือกมาใช้ภายในเกมที่สร้างขึ้น เช่น ค่าพลังชีวิต ค่าพลังโจมตี ของตัวละคร

3. Ch (Character) ทำหน้าที่เก็บข้อมูลในส่วนที่เกี่ยวกับตัวละครภายในแต่ละฉาก เช่น รหัสตัวละครทั้งหมดในฉาก ตำแหน่งของตัวละครภายในฉาก และตัวละครจะอ้างอิงค่าพลัง File Script ประเภท Cdef และ อ้างอิงถึง Model ที่ใช้ จาก File Script ประเภท Cmod

4. Cmod (Character Model) ทำหน้าที่เก็บข้อมูลในส่วนของ Model ของตัวละครต่างๆที่ใช้ภายในฉากต่างๆของเกม เช่น ชื่อ File Model ชื่อ File Texture

5. Env (Environment) ทำหน้าที่เก็บข้อมูลของสิ่งแวดล้อมต่างๆ ภายในฉาก และจะอ้างอิงถึง File Script ประเภท Emod ซึ่งเก็บข้อมูล Model ที่สิ่งแวดล้อมในฉากนั้นๆต้องใช้

6. Emod (Environment Model) ทำหน้าที่เก็บข้อมูลในส่วนของ Model ของสิ่งแวดล้อมต่างๆที่ใช้ภายในฉากต่างๆของเกม เช่น ชื่อ File Model ชื่อ File Texture

7. Trp (Transport) เก็บข้อมูลของจุดที่ใช้บอกตำแหน่งที่เชื่อมกันระหว่างฉากแต่ละฉาก เพื่อใช้ในการกำหนดตำแหน่งเมื่อตัวละครมีการเดินทางข้ามฉากภายในเกม

8. Scp (Script) จะเป็น File ที่ใช้เก็บคำสั่งที่ใช้ควบคุมเหตุการณ์ต่างๆภายในเกมโดยเหตุการณ์ภายในเกมจะแบ่งส่วนออกเป็น 2 ส่วน คือ

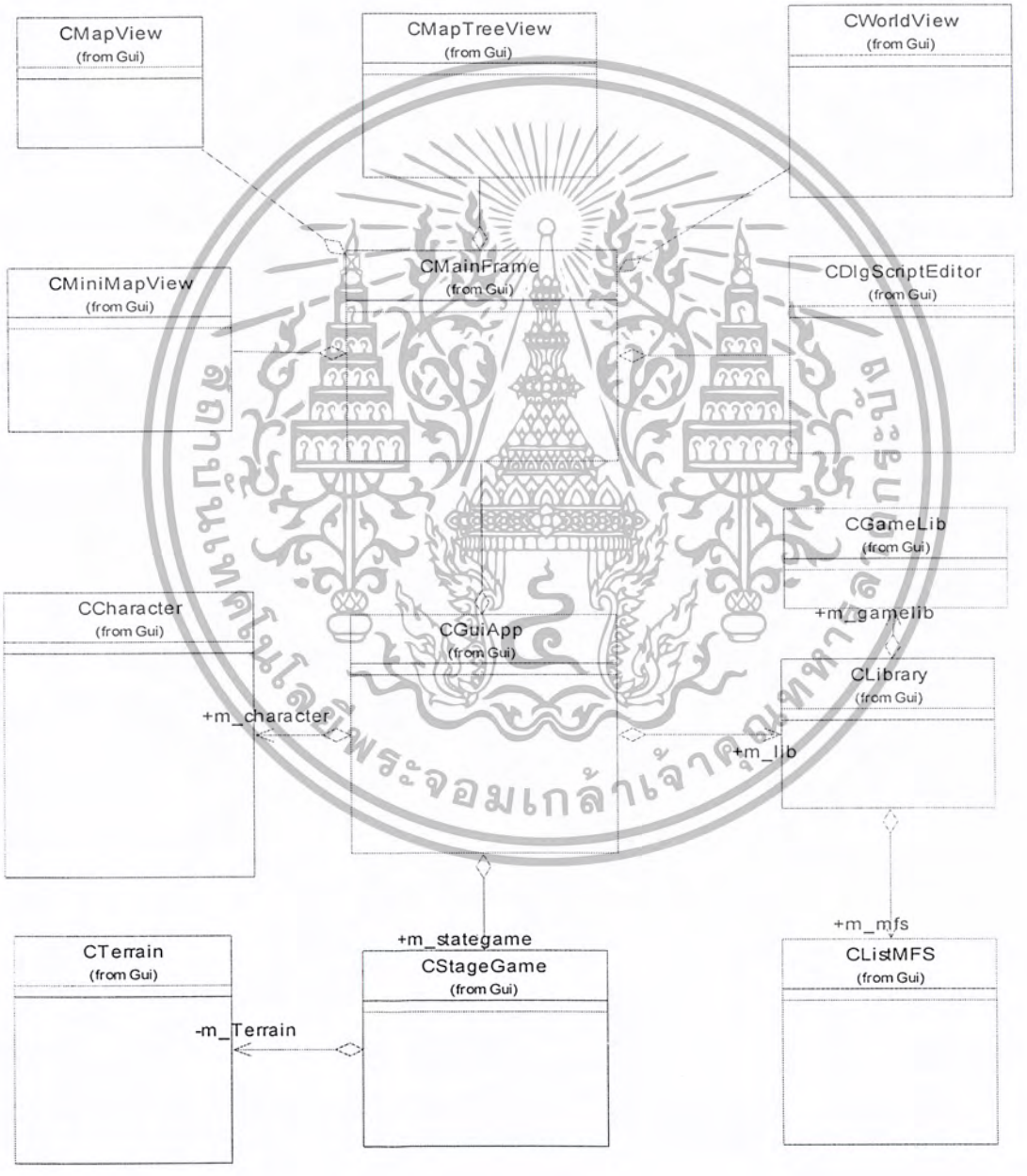
- เหตุการณ์ที่เกิดขึ้นกับแผนที่ (Map Script) จะใช้ควบคุมเหตุการณ์ที่เกิดขึ้นเมื่อเข้ามาในแผนที่นี้ เช่นเมื่อมาที่แผนที่นี้แล้ว เปลี่ยนเพลง มีหมอก เป็นต้น
- เหตุการณ์ที่เกิดขึ้นกับตัวละคร (Character Script) จะใช้ควบคุมเหตุการณ์ที่เกิดขึ้นจากวัตถุต่างๆเช่น เมื่อสัมผัสกับตัวละครในเกม เป็นต้น

9. Rt (route point) เป็นไฟล์ที่ใช้ควบคุมทิศทางการเดินของตัวละครที่ไม่ใช่ตัวเอก หรือ NPC โดยจะให้ผู้ใช้ได้กำหนดทิศทางและระยะทาง เพื่อให้ตัวละครเหล่านี้เคลื่อนที่ได้

5.4.3 ขั้นตอนการออกแบบโครงสร้างของ GUI และ เกม

ในขั้นตอนนี้จะสามารถแยกการทำงานออกเป็น 2 ส่วน คือ การออกแบบโครงสร้างของ GUI ที่ใช้ในการสร้าง File Script กับ การออกแบบโครงสร้าง ตัวเกม RPG 3 มิติ ที่สามารถอ่าน File Script นั้นและนำไปแสดงผลออกมาเป็นเกมได้

ในส่วนของการออกแบบโครงสร้าง GUI นั้นแบ่งการออกแบบหลักได้เป็น 2 ส่วนใหญ่ๆ คือ ส่วนที่เป็นการสร้างคลาสเพื่อการสร้างหน้าต่างติดต่อกับผู้ใช้งาน กับอีกส่วนคือ ส่วนในการทำงานด้านสามมิติ ดังรูปข้างล่าง



รูปที่ 5-5 แสดงแผนภูมิคลาสของส่วน GUI

- ส่วนที่เป็นคลาสเพื่อการสร้างหน้าต่างติดต่อกับผู้ใช้งาน

CGuiApp เป็นคลาสหลักที่สืบทอดมาจากคลาส CWinApp ซึ่งทำหน้าที่หลักคือเรียกโปรแกรมขึ้นมาทำงานซึ่งภายในมีการสร้าง อินแสตนท์ ออบเจ็กต์ของ CMainFrame ไว้โดย CGuiApp มีหน้าที่ในการเริ่มต้นและปิดโปรแกรม โดยมีฟังก์ชัน InitInstance() ทำหน้าที่กำหนดรูปแบบโปรแกรมว่าเป็นโปรแกรมแบบชนิดใดโดยได้กำหนดให้เป็นโปรแกรมแบบเปิดได้หลายหน้าต่างพร้อมกัน และมีฟังก์ชันหลักคือ OnIdle() ซึ่งเป็นฟังก์ชันที่ถูกเรียกใช้งานเมื่อโปรแกรมไม่มีการรับอินพุต โดยทำหน้าที่ในกรณีที่โปรแกรมถูกเรียกใช้โหมดการแสดงผลภาพสามมิติด้วยการเรียกใช้คลาส CWorldView

CMainFrame เป็นคลาสที่สืบทอดมาจากคลาส CMDIFrameWnd ของ MFC ซึ่งถูกเรียกใช้งานหลังจากเปิดโปรแกรมทำหน้าที่เป็นตัวสร้างกรอบหน้าต่างหลักว่าจะมี Frame ย่อยๆอะไรบ้าง ซึ่งประกอบด้วยคลาสย่อยๆ 5 คลาส คือ CDlgScriptEditor , CMapView , CWorldView , CMiniMapView , CMapTreeView ซึ่งทั้ง 5 คลาสนี้ เป็นคลาสในการสร้าง Frame ย่อยๆ ที่ทำหน้าที่ในการแสดงผลต่างกันไป และคลาส CMainFrame นี้ถูกสร้างอินแสตนท์ขึ้นมาหลังจากสร้างหน้าต่างหลักที่ทำหน้าที่ติดต่อกับผู้ใช้ดังนั้นหน้าที่หลักคือเลือกสร้าง โครงงานเกมใหม่ เปิดโครงงานเกมที่สร้างไว้แล้ว และ บันทึกข้อมูลโครงงานเกมที่กำลังจัดทำอยู่ โดยจะมีการสร้างเมนูในการทำงานหลายรูปแบบ และมี ไดอะล็อคติดต่อกับผู้ใช้เพื่อรายงานผลข้อมูลของ GUI RPG ทั้งหมดไม่ว่าจะเป็นไลบรารี ที่เก็บโมเดล หรือ เรียกหน้าต่างย่อยๆเพื่อการทำงานในแต่ละส่วน

CMapTreeView เป็นคลาสที่สืบทอดมาจาก CTreeView ของ MFC ทำหน้าที่แสดงรายละเอียดทั้งหมดของเกมที่ผู้ใช้ได้สร้างขึ้นมาในรูปแบบของ แผนภูมิต้นไม้ เพื่อให้ผู้สร้าง สามารถตรวจสอบรายละเอียดของเกมได้โดยสะดวก โดยคลาสจะเริ่มทำงานเมื่อมีการเปิดโครงงานเก่า ขึ้นมาโดยจะนำแผนที่ทั้งหมดในโครงงานนั้นมาแสดง และจะนำทั้ง ตัวละคร, สิ่งแวดล้อมและจุดเชื่อมแผนที่ทั้งหมดที่อยู่ในแต่ละแผนที่ออกมาแสดงในรูปแบบเป็นรากของแต่ละแผนที่นั้น

CMapView เป็นคลาสที่สืบทอดมาจาก CView ของ MFC ทำหน้าที่ในการเลือกแสดงผลภาพแผนที่ซึ่งกำลังถูกเลือกใช้งานอยู่ หรือเป็นแผนที่ ที่มีตัวละครที่กำลังเรียกใช้งานอยู่ ฟังก์ชันหลัก คือ OnDraw() ทำหน้าที่วาดรูปแผนที่บนวิวที่กำหนด โดยมี คลาส CDIBSectionLite m_bitmap ทำหน้าที่ในช่วยการแสดงผลรูปภาพแผนที่เพิ่มเข้ามาโดยสามารถอ่านไฟล์ในรูปแบบ Bitmap ที่มีขนาดแตกต่างกันได้

CWorldView เป็นคลาสที่สืบทอดมาจาก CScrollView ของ MFC ทำหน้าที่ 2 รูปแบบการทำงาน คือ การทำงานในการแสดงการเชื่อมต่อกันของแผนที่ทั้งหมด กับ แสดงผลแผนที่เดียวในรูปแบบ 3 มิติ ในส่วนรูปแบบการทำงานในการแสดงการเชื่อมต่อกันของแผนที่ทั้งหมดนั้น จะมีคลาส CDIBSectionLite *m_pBitmap[] ทำหน้าที่ในการแสดงผลรูปแผนที่ทั้งหมด ออกมาเป็นรูปแบบคล้ายแผนที่โลก และมีฟังก์ชัน MouseMapMove() ทำหน้าที่ในการเปลี่ยนตำแหน่งแผนที่ที่เลือกว่าต่อกันอย่างไร โดยมีการรับ Input จากเมาส์โดยฟังก์ชัน OnMouseMove() และฟังก์ชัน OnInitialUpdate() ทำหน้าที่ในการเริ่มต้นวาด

หน้าจอภาพ สำหรับส่วนรูปแบบการแสดงผลแผนที่เดียวในรูปแบบ 3 มิตินั้น จะการแสดงผลข้อมูลในฉากแต่ละฉาก และเพิ่มหรือเปลี่ยนแปลงตำแหน่ง ของ โมเดลวัตถุ ต่างๆในฉาก ดังนั้นต้องมีฟังก์ชันในการควบคุมการเคลื่อนไหวของกล้อง การแสดงผลการเคลื่อนไหวของโมเดล รวมทั้งแสดงคุณสมบัติต่างๆของโมเดลวัตถุ เช่น ตำแหน่งตัวละคร ค่าพลังชีวิตต่างๆ ค่าของสิ่งของแต่ละอย่าง เป็นต้น ฟังก์ชันที่จำเป็นต่อการแสดงผลภาพสามมิติ คือ ฟังก์ชัน Render() ซึ่งในฟังก์ชันนี้มีการเรียกใช้ส่วนที่เป็นกราฟฟิก เช่น CStageGame , CCharacter เป็นต้น ฟังก์ชันหลักที่ใช้ในการควบคุมกล้องให้เคลื่อนที่ตามเมาส์มี สี่ส่วนคือ MouseCameraMove() ใช้ในการเคลื่อนที่กล้องไปตามที่ที่ลากไป ฟังก์ชัน MouseCamerraZoom() ใช้ในการเลื่อนกล้องเข้าออกจากจุดโฟกัส ฟังก์ชัน MouseCameraYaw() เป็นฟังก์ชันที่ใช้ในการหมุนกล้องตามแนวนอน และฟังก์ชัน MouseCameraPtich() เป็นฟังก์ชันที่ใช้ในการหมุนกล้องตามแนวตั้ง ซึ่งทั้ง 3 ฟังก์ชันมีประโยชน์ในการกำหนดตำแหน่งที่จะวางวัตถุโมเดลลงไปบนแผนที่ เพราะหากแผนที่มีขนาดใหญ่ก็จำเป็นต้องเลื่อนกล้อง ไปตามจุดต่างๆตามต้องการ ฟังก์ชันถัดมาอีกสี่ฟังก์ชันเป็นฟังก์ชันที่ใช้ในการเพิ่มวัตถุโมเดลลงไปในแผนที่ รวมทั้งเปลี่ยนแปลงแก้ไข ตำแหน่งของวัตถุโมเดลต่างๆในแผนที่ตามต้องการ คือ ฟังก์ชัน MouseTransportMove() ทำหน้าที่ในการเลือกจุดเชื่อมแผนที่ใหม่มาวางบนแผนที่รวมทั้งเคลื่อนย้ายตำแหน่งจุดเชื่อมแผนที่ไปตามต้องการ ฟังก์ชัน MouseCharacterMove() ทำหน้าที่ในการเลือกตัวละครมาวางบนแผนที่ รวมทั้งเปลี่ยนแปลงตำแหน่งตัวละครในแผนที่ตามต้องการ ฟังก์ชัน MouseEnvironmentMove ทำหน้าที่ในการเลือกสิ่งแวดล้อมมาวางบนแผนที่ รวมทั้งแก้ไขเปลี่ยนแปลงตำแหน่งสิ่งแวดล้อมในแผนที่ ตามต้องการ

CMiniMapView เป็นคลาสที่สืบทอดมาจาก CFormView ของ MFC ทำหน้าที่ในการแสดงผลเมนูคำสั่งการจัดการ แผนที่ , ตัวละคร , สิ่งแวดล้อม , จุดเชื่อมแผนที่ที่กำลังเลือกทำงานอยู่ คลาสนี้จึงเป็นคลาสหลักในการสร้างภูมิประเทศของแผนที่ หรือ เพิ่มและแก้ไขตัวละครในแต่ละฉากได้ตามต้องการ โดยจะส่วนใหญ่จะเป็นทำงานร่วมกับคลาส CWorldView ที่ใช้แสดงการเชื่อมต่อกันของแผนที่ทั้งหมด และ แสดงรายละเอียดแผนที่แบบ 3 มิติ ซึ่งจะทำให้การจัดการเปลี่ยนแปลงรายละเอียดต่างๆเป็นไปโดยสะดวก

จากคลาสที่อธิบายมาทั้งหมดตัว GUI RPG สามารถสร้างแผนที่ต่อเนื่องกันได้ สามารถวางตัวละครและสิ่งแวดล้อม ได้แล้วแต่ยังไม่สามารถกำหนดเหตุการณ์ ของตัวละคร ปริศนาเงื่อนไขของเกมได้ ดังนั้นจึงต้องมีส่วนของเหตุการณ์ที่เกิดกับวัตถุประเภทต่างๆ เช่น การพูดคุยกันระหว่างตัวละคร จากส่วนที่กล่าวมาทั้งการเปลี่ยนแปลงแก้ไขแผนที่ และการวางแผนที่แล้ว ต่อไปคือการกำหนด เงื่อนไข ปริศนาและเหตุการณ์ต่างๆที่จะเกิดกับตัวละคร นั่นคือการกำหนดเนื้อเรื่องให้กับตัวเกม ซึ่งได้ออกแบบคลาสหลักๆ ดังนี้

CDlgScriptEditor เป็นคลาสที่สืบทอดมาจาก CDialog ของ MFC ซึ่งมีหน้าที่ในการกำหนดเหตุการณ์ให้กับวัตถุโมเดลต่างๆเช่น ตัวละคร แผนที่ เป็นต้น เมื่อผ่านขั้นตอนสองขั้นแรกคือ สร้างแผนที่ทั้งหมดและแก้ไขแผนที่แต่ละแผนที่แล้ว ก็สามารถเลือกกำหนดเนื้อเรื่องและปริศนาให้กับเกม ได้ โดย

เรียกฟังก์ชันแรกในการทำงานคือ OnInitDialog() เพื่อสร้างไดอะล็อกขึ้นมา จากนั้นโปรแกรม จะเรียกฟังก์ชัน LoadAction() ไว้สำหรับเปิด ไฟล์ แอคชันซึ่งจะมีการเรียกแอคชันต้นแบบที่ได้สร้างไว้แล้วขึ้นมาทั้งหมด จากนั้นจะเรียกใช้ LoadScript() เป็นการโหลดแอคชันสคริปที่ผู้ใช้เคยสร้างไว้แล้ว จากนั้นก็เป็นขั้นตอนการสร้างเหตุการณ์ต่างๆตามต้องการ โดยเหตุการณ์ต่างๆมีได้หลายรูปแบบ และมีฟังก์ชันที่สนับสนุนการทำงานดังนี้

การเกิดขึ้นของเหตุการณ์ต่างๆไม่ว่าจะเป็นการพูดคุย การมีเสียงดนตรี นั้น เหตุการณ์จะแบ่งได้เป็น 2 ประเภทหลักๆด้วยกัน คือเหตุการณ์ที่เกิดกับแผนที่ เหตุการณ์ที่เกิดกับตัว คลาส CDlgScriptEditor จึงสร้างเหตุการณ์พื้นฐานขึ้นมาสนับสนุนการสร้างเหตุการณ์ดังนี้

ฟังก์ชัน OnPlayMusic() ใช้สำหรับสร้างสคริปสำหรับเล่นเสียงเพลงโดยการทำงานคือ เมื่อเลือกแอคชัน Play Music จะเกิดไดอะล็อกให้เลือก ไฟล์ Midi และ ให้ใส่ค่า Volumn ผลลัพธ์คือจะได้ สคริปที่อธิบายว่า “Play Music Midi (ชื่อไฟล์เพลง) Volumn(ความดัง)” เป็นต้น

ฟังก์ชัน OnSetflag() ใช้สำหรับกำหนดค่าความจริงให้กับ Flag ซึ่งเป็นตัวแปรที่ใช้อ้างถึงในบางเงื่อนไข ลักษณะของ Script คือ “Set flag (หมายเลข) is (จริงหรือเท็จ)”

ฟังก์ชัน OnIfflag() ไว้สำหรับกำหนดค่าว่าจริงหรือเท็จในเพื่อประโยชน์ในการสร้างเงื่อนไขของเกม โดยฟังก์ชันนี้มีค่าที่ต้องกำหนดสองค่าคือ หมายเลขแฟค และ ค่าจริงหรือเท็จ โดยลักษณะสคริปคือ “if flag(หมายเลข) is (จริงหรือเท็จ) then” ฟังก์ชันนี้ต้องใช้คู่กับฟังก์ชัน OnSetflag() ในการกำหนดหมายเลขแฟคเริ่มต้นว่าให้จริงหรือเท็จ ตัวอย่างการใช้งานเช่น เมื่อเริ่มต้นเกมเรากำหนด ตัวโจร มีแฟคหมายเลข 1 เป็นจริงและกำหนดให้เมื่อตัวโจรตายแฟคหมายเลข 1 เป็นเท็จเป็นต้น

ฟังก์ชัน OnElse() เป็นการเขียนเงื่อนไขของสคริปในลักษณะ ที่ว่า “ถ้า .... แล้ว “ เป็นต้น ดังนั้นฟังก์ชันนี้จริงเป็นการกำหนดเงื่อนไขขั้นที่สอง หรือขั้นถัดไปดังนั้น ฟังก์ชันนี้ต้องใช้คู่กับฟังก์ชันที่กำหนดเงื่อนไขเริ่มต้น อย่าง OnIfflag() ลักษณะสคริปคือ “else”

ฟังก์ชัน OnEndif() เป็นการจบเงื่อนไขสคริปที่สร้างไว้ ในกรณีที่เงื่อนไขยังไม่สิ้นสุด โดยการทำงานคือต้องกำหนดค่าหลังจากวางเงื่อนไขไว้แล้ว เช่น “ ถ้า ... แล้ว ... จบเงื่อนไข” ลักษณะเงื่อนไขคือ “endif” เป็นต้น ฟังก์ชันนี้ต้องใช้คู่กับ OnIfflag() และ OnElse() เพื่อให้เงื่อนไขสมบูรณ์แบบ

ฟังก์ชัน OnEbtransport() เป็นการกำหนดว่า ให้จุดเปลี่ยนฉากทำงานหรือไม่ ลักษณะของสคริปคือ “enable transport (หมายเลขจุดเปลี่ยนฉาก หรือ ชื่อจุดเปลี่ยนฉาก) is (ทำงาน หรือ ไม่ทำงาน) ” เป็นต้น

ฟังก์ชัน OnVisiblechar() เป็นการกำหนดการมีอยู่ของตัวละคร ลักษณะสคริปคือ “visible character (หมายเลขตัวละคร หรือ ชื่อตัวละคร) is (จริงหรือเท็จ)” จะเห็นได้ว่าฟังก์ชันนี้สามารถนำมาสร้างเงื่อนไข หรือปริศนา ต่อจากฟังก์ชัน OnIfflag() ให้ทำงานได้

ฟังก์ชัน OnSpeak() เป็นฟังก์ชันที่สร้างการสนทนาระหว่างตัวละครให้พูดกันตามที่กำหนด โดยฟังก์ชันนี้ต้องกำหนดค่าสองอย่าง คือตัวละครอะไรและคำพูดที่จะพูด “character (หมายเลขตัวละคร หรือ ชื่อตัวละคร) speak (ประโยคที่จะพูด) ” จะเห็นได้ว่าฟังก์ชันนี้สามารถนำไปสร้างปริศนา หรือบอกปริศนาให้ผู้เล่นได้รู้

ฟังก์ชัน OnEndscript() เป็นฟังก์ชันการสั่งให้จบการทำงานของสคริป ลักษณะคำสั่งคือ “end script”

ฟังก์ชัน OnInchp() เป็นฟังก์ชันที่ทำงานกับตัวเอกเป็นหลัก คือใช้เพิ่มค่าพลังชีวิตให้กับตัวเอก ลักษณะสคริปคือ “increase hp (จำนวนที่เพิ่ม) to hero”

ฟังก์ชัน OnIfchardie() เป็นฟังก์ชันที่ใช้กำหนดเงื่อนไขไว้กับตัวละครกำหนดว่า ถ้าตัวละครที่กำหนดตาย โดยฟังก์ชันนี้ต้องใช้ฟังก์ชันอย่าง OnMoveChar() หรือ OnInchp() ควบคู่เพื่อกำหนดเงื่อนไขต่อไป เช่นหากตัวศัตรูตายแล้ว ตัวพระเอกจะมีพลังเพิ่มขึ้นตามที่กำหนดไว้ เป็นต้น ลักษณะสคริปคือ คือ “if character (ชื่อตัวละคร หรือหมายเลขตัวละคร) on map (ชื่อแผนที่หรือหมายเลขแผนที่) die”

ฟังก์ชัน OnShowfog() เป็นฟังก์ชันที่ใช้สร้างหมอกในฉากตามเงื่อนไขที่กำหนด ลักษณะสคริปคือ “fog enable (ทำงานหรือไม่ทำงาน) color red (ค่าสีแดงของหมอก) green (ค่าสีเขียวของหมอก) blue (ค่าสีน้ำเงินของหมอก) ” เป็นต้น ตัวอย่างการใช้งานเช่น เมื่อตัวพระเอกกำจัดตัวศัตรูได้แล้วก็ให้เกิดหมอกในฉากเป็นต้น

ฟังก์ชัน OnEndgame() เป็นฟังก์ชันสำคัญที่จะกำหนดว่าเงื่อนไข เกมจะจบเมื่อใด ซึ่งอาจจะควบคู่กับฟังก์ชันที่ผ่านมาได้หลายอย่าง เพื่อกำหนดรูปแบบ ของการจบเกมต่างกันไป เช่น เกมจบเมื่อตัวพระเอกหาสิ่งของที่ต้องการพบ หรือ จบเกมเมื่อตัวพระเอกกำจัดตัวศัตรูได้ตามที่กำหนด ลักษณะของสคริป คือ “ end game”

จะเห็นได้ว่าฟังก์ชันที่กำหนดมาให้เป็นฟังก์ชันพื้นฐานโดยทั่วไปของเกมแนว RPG ทำให้ผู้ใช้สามารถนำไปสร้างเกม ให้มีเนื้อเรื่องตามต้องการได้

- ส่วนที่เป็นคลาสในการทำงานด้าน 3 มิติ

เกมแนว RPG นั้นมีข้อมูลที่จำเป็นต้องจัดการหลายส่วน ไม่ว่าจะเป็นตัวละครซึ่งต้องเก็บค่าต่างๆ เช่นตำแหน่งตัวละครบนแผนที่ ทั้งแกน x ,y และ z หรือค่าพลังต่างๆ หรือ จุดเปลี่ยนฉากซึ่งต้องกำหนดว่า จะเปลี่ยนจากฉากใดไปยังฉากใด จุดเปลี่ยนฉากอยู่ที่ตำแหน่งใดของแผนที่ หรือมีการกำหนด ค่าว่าทำงานหรือไม่ทำงาน หรือ ไฟล์หลักที่กำหนดว่าแผนที่รวมนั้นประกอบด้วยแผนที่ย่อยๆกี่แผนที่ และใช้ไฟล์อะไรเป็นตัวแสดงผลเป็นต้น

CStageGame เป็นคลาสที่ทำหน้าที่ช่วยในการแสดงผลข้อมูลด้านกราฟฟิกทั้งหมด เช่น การแสดงผลภาพแผนที่สามมิติ และเป็นคลาสที่จัดการกับข้อมูลไฟล์ env , map , trp

CCharacter เป็นคลาสที่ทำหน้าที่ทำหน้าที่เกี่ยวกับตัวละครโดยเฉพาะเช่น เพิ่มตัวละคร วางตัวละครบนแผนที่ แล้วเก็บตำแหน่ง แสดงผลสามมิติของตัวละคร รวมทั้งกำหนด ค่าต่างๆ เช่นพลังชีวิตเป็นต้น

CLibrary ทำหน้าที่ในการเปิดโปรเจกต์เกมที่มีอยู่แล้ว สร้างโปรเจกต์เกมใหม่ รวมทั้งเก็บข้อมูล โดยภายในคลาสจะมี โดยภายในประกอบด้วย

CListMFS ทำหน้าที่จัดการเรื่องการเพิ่มแผนที่เข้าไปในเกม ไม่ว่าจะเป็นการตั้งชื่อแผนที่ การกำหนดค่าว่าแผนที่ใดต่อกับแผนที่ใด และใช้ไฟล์ชื่อ อะไรในการแสดงผลสามมิติ รวมทั้งเป็นตัวสร้างไฟล์อื่นๆที่จำเป็นต่อการสร้างเกม ไม่ว่าจะเป็น ไฟล์ trp ไฟล์ scp ไฟล์ cmod หรือ ไฟล์ env เป็นต้น

CGameLib ซึ่งทำหน้าที่ในการดึงข้อมูลจากไลบรารีมาแสดงผลไม่ว่าจะเป็น ไฟล์ mmod ไฟล์ cdef ไฟล์ cmod ไฟล์ imod ซึ่งคลาสนี้ทำหน้าที่ในการเปิดไฟล์ขึ้นมาทั้งหมด ตอนเริ่มเปิดโปรเจกต์เกม

หลังจากขั้นตอนการออกแบบโครงสร้างของ GUI เสร็จสิ้นขั้นตอนต่อไปจะเป็นขั้นตอนการทดสอบความถูกต้องของ File Script ที่ GUI สร้างขึ้นมาได้ และนำ File Script ไปใช้ร่วมกับเกมที่ได้สร้างไว้ ซึ่งจะกล่าวถึงในบทต่อๆไป

บทที่ 6
การพัฒนาเกมเพื่อทดสอบ GUI Game Engine

ประเภทเกมที่พัฒนา

สำหรับเกมที่พัฒนาขึ้นนี้เป็น เกม RPG ที่ผู้เล่นจะต้องเล่นตามเงื่อนไขของเกมที่ได้สร้างเป็นเรื่องราวไว้ ตั้งแต่ต้นจนจบ โดยเกมที่จะสร้างมีลักษณะดังนี้

- เกมจะมีเรื่องราวและเป้าหมายให้ผู้เล่นสามารถดำเนินเหตุการณ์ตามเรื่องราวของเกมได้
- เกมจะจบเมื่อผู้เล่น ตาย หรือ สามารถดำเนินตามเนื้อเรื่องที่ได้วางไว้
- ตัวละครที่เป็นNPC สามารถตอบสนองกับผู้เล่นได้ และมีการเคลื่อนไหวไปมาโดยไม่ต้องควบคุม
- Monster จะต้องเข้ามาโจมตีผู้เล่น เมื่อผู้เล่นเข้าไปใกล้
- ในเกมจะต้องมีเสียงดนตรีประกอบการเล่นด้วย
- การควบคุมตัวละครเอก และ การควบคุมกล้อง ทำได้โดยการใช้เม้าส์ควบคุมทั้งหมด

ขั้นตอนการทดสอบมีดังนี้

1. สร้างโครงเรื่องของเกม ว่ามีเป้าหมายอะไรบ้าง มีเรื่องราวเป็นยังไง ฉากทั้งหมดจะมีกี่ฉาก จะให้บรรยากาศในเกมเป็นอย่างไร
2. เริ่มการสร้างเกมด้วย GUI Game Engine
3. เมื่อสร้างเสร็จแล้วให้ไปยังโฟลเดอร์ที่อยู่ของเกมซึ่งจะบอกตอนสร้างเกมใหม่ จากนั้นทดลองรันโปรแกรม
4. สังเกตการทดลอง จากนั้นนำมาเปรียบเทียบดูกับลักษณะของเกมและดูว่าตรงกับเนื้อเรื่องที่เราได้วางไว้ตอนเริ่มต้นหรือไม่

เริ่มการทดลอง

1. สร้างโครงเรื่องและรายละเอียดของเกม

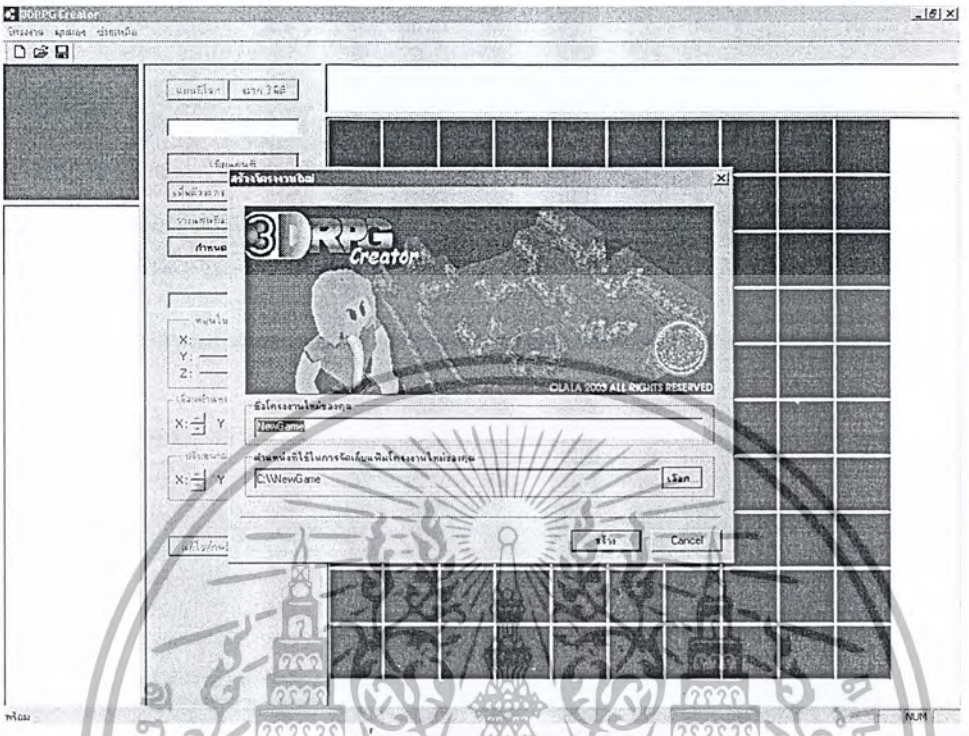
เรื่องราวของเกมที่จะพัฒนามีอยู่ว่า คุณจะสวมบทเป็นเด็กหนุ่มในหมู่บ้านแห่งหนึ่งที่ถูกเจ้า Sligg สาปให้หายตัวไป คุณต้องไปฆ่ามันเพื่อที่จะช่วยคนในหมู่บ้าน ระหว่างทางก็มีผู้ช่วยเหลือและสัตว์ประหลาดรอคอยอยู่ เมื่อคุณสามารถฆ่า เจ้า Sligg ได้หมู่บ้านก็เกิดความสงบสุขขึ้นมาอีกครั้ง

แผนที่ทั้งหมด มีด้วยกัน 3 แผนที่ คือ

1. แผนที่หมู่บ้าน
2. แผนที่เส้นทางที่จะไปยังที่อยู่ของ Sligg
3. แผนที่ที่อยู่อาศัยของเจ้า Sligg

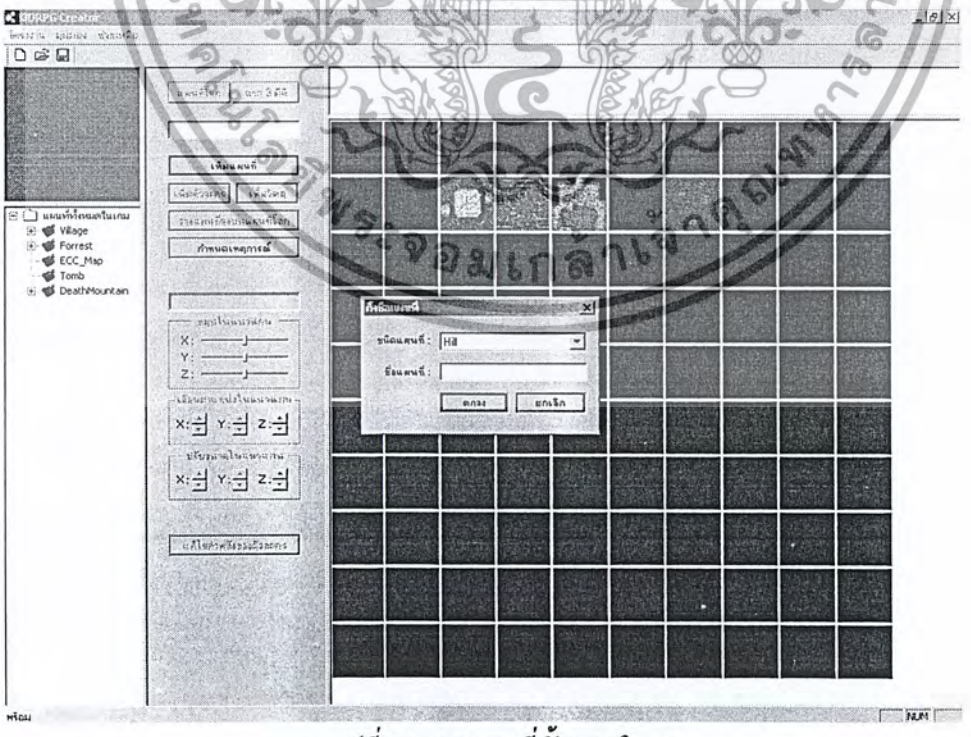
2. เริ่มการสร้างเกมด้วย GUI Game Engine

2.1 ทำการสร้างโปรเจ็กของเกมที่จะสร้างพร้อมทั้งตั้งชื่อและเลือกตำแหน่งของโปรเจ็ก



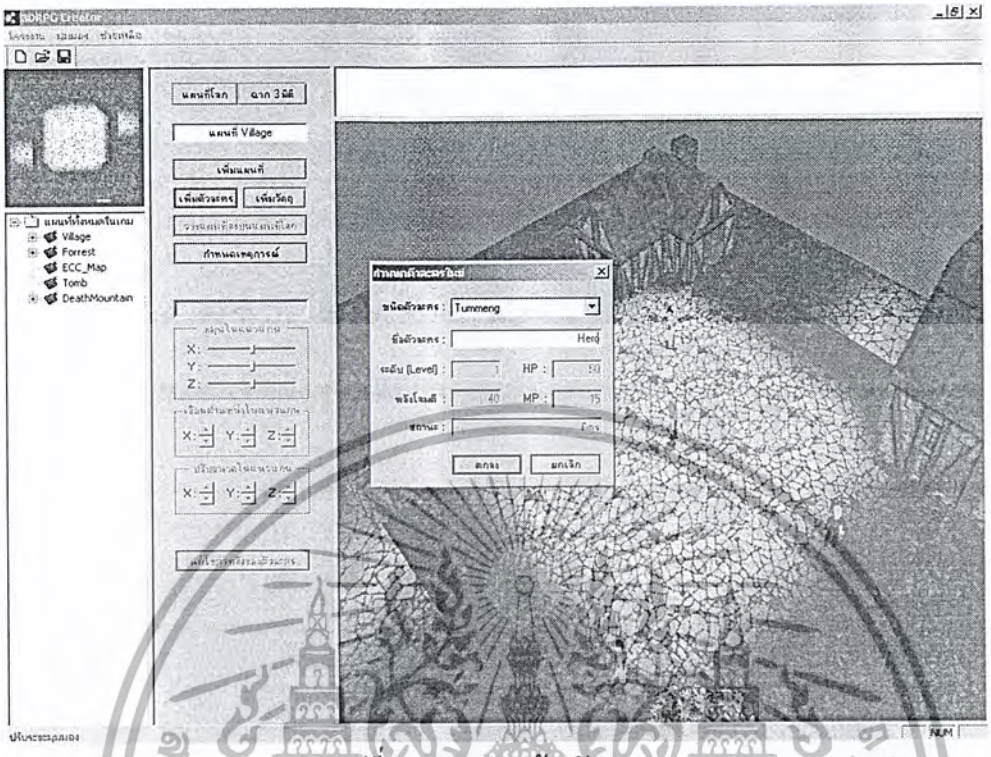
รูปที่ 6-1 เมนูการสร้างโปรเจ็ก

2.2 เริ่มต้นด้วยการเลือกและวางแผนที่ ขั้นตอนนี้จะเป็นการวาง ตำแหน่งแผนที่โดยรวม

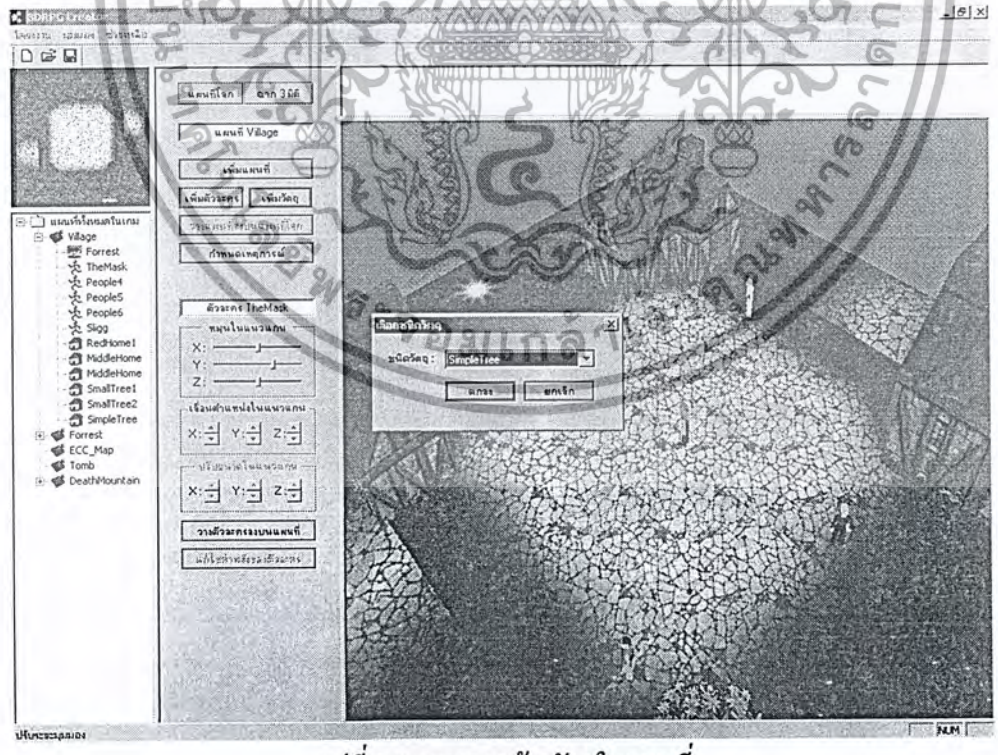


รูปที่ 6-2 ฉากแผนที่ทั้งหมดในเกม

2.3 เข้าไปจัดฉากแต่ละฉาก

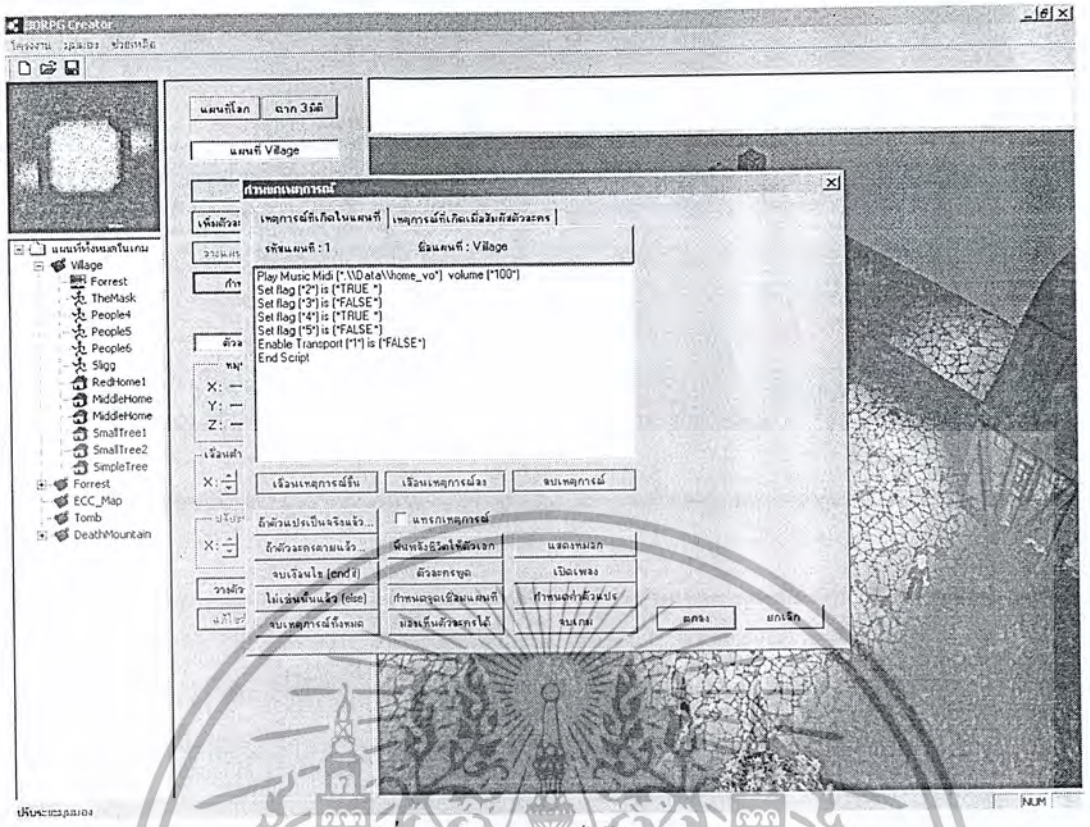


รูปที่ 6-3 เมนูการสร้างตัวละคร

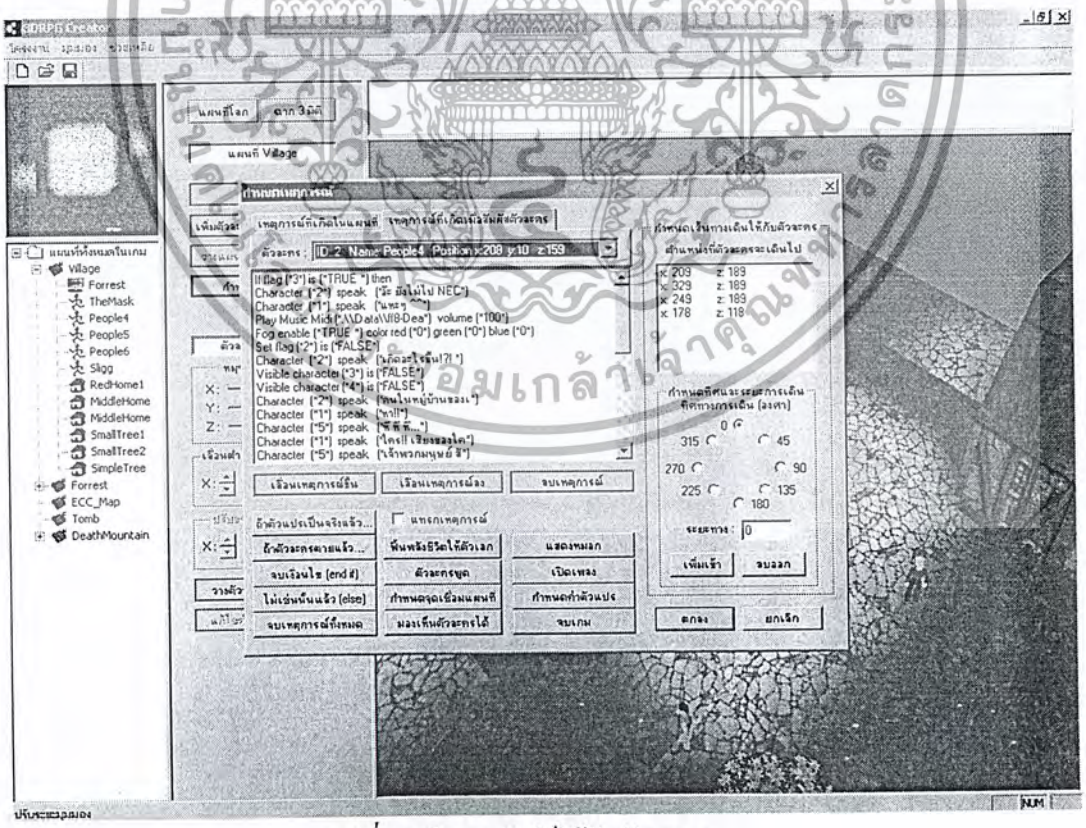


รูปที่ 6-4 เมนูการสร้างวัตถุในแผนที่

2.2 สร้างEvent ให้กับตัวละคร ด้วยส่วน Script Event Editor ของ GUI Game Engine

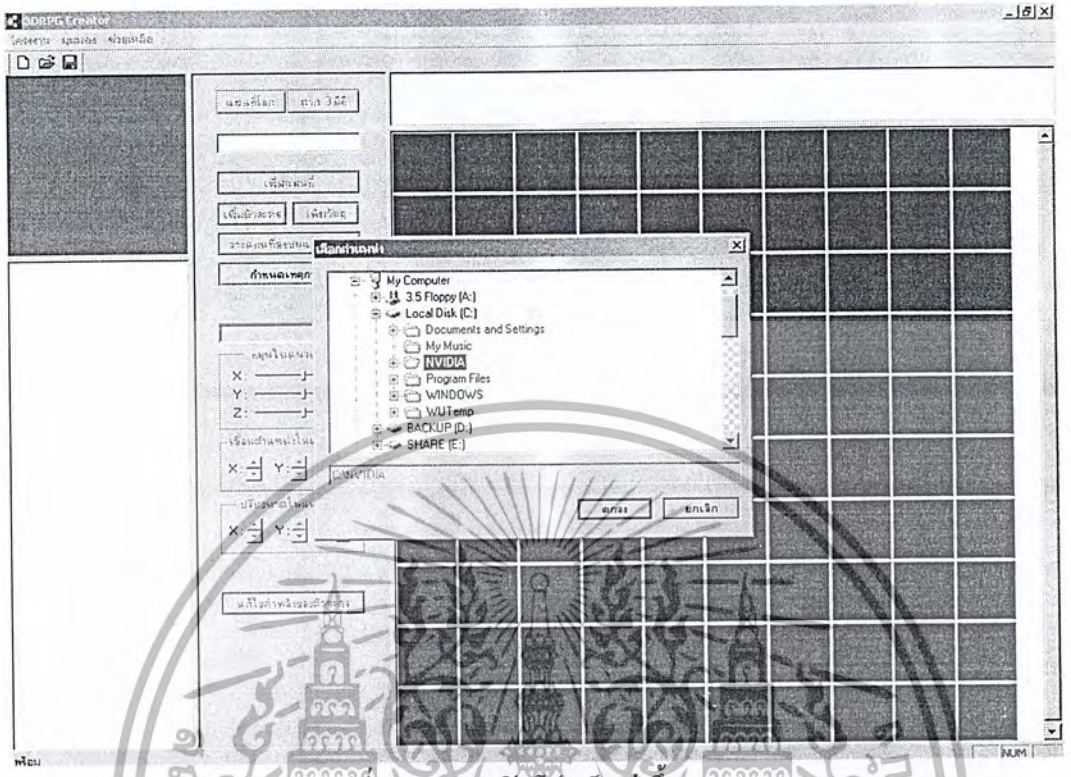


รูปที่ 6-5 เขียนสคริปต์ให้กับฉาก



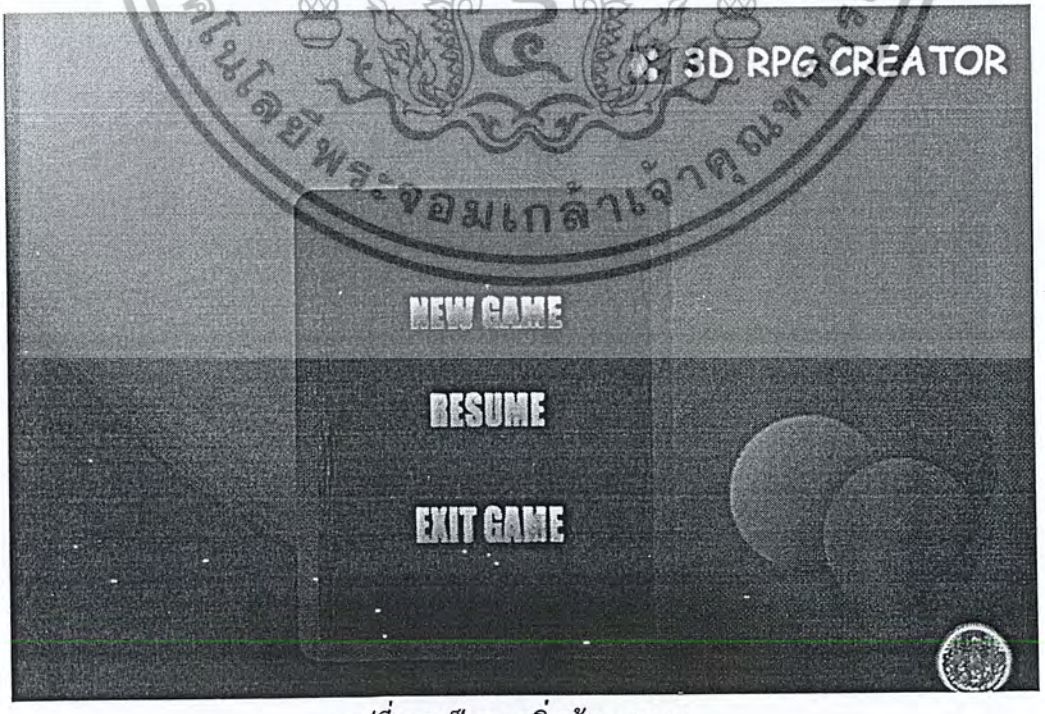
รูปที่6-6 เขียนสคริปต์ให้กับตัวละคร

3. ทำการบันทึกข้อมูลเพื่อสามารถที่นำเกมไปหรือนำไปใช้พัฒนาต่อในครั้งหน้า

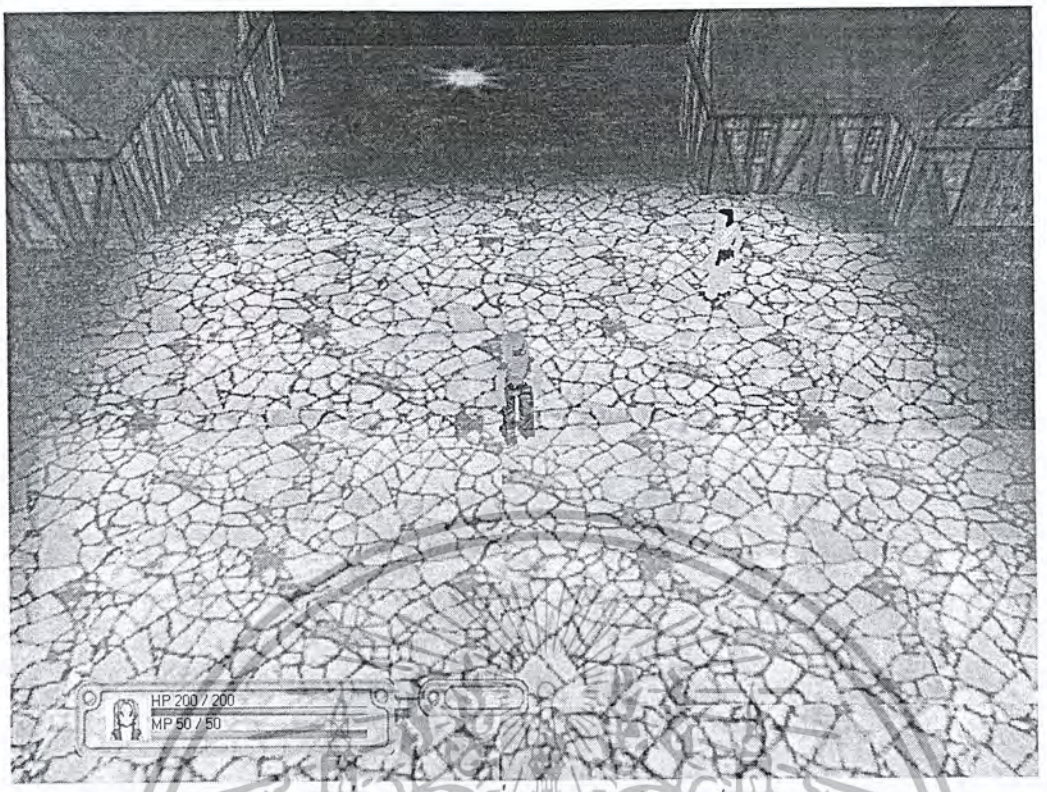


รูปที่6-7 เมนูการเปิดโปรเจ็กเก่าขึ้นมา

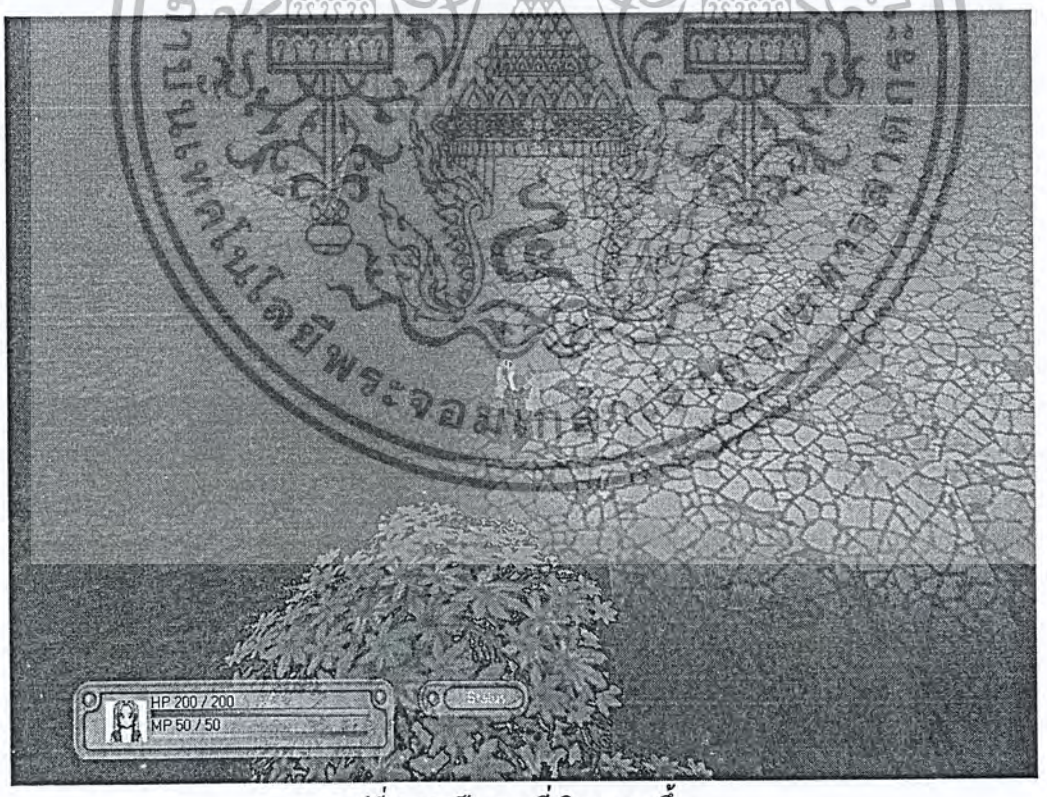
4. ทดลองรันเกมและทดลองเล่นดูว่าเป็นไปตามเนื้อเรื่องรึเปล่า



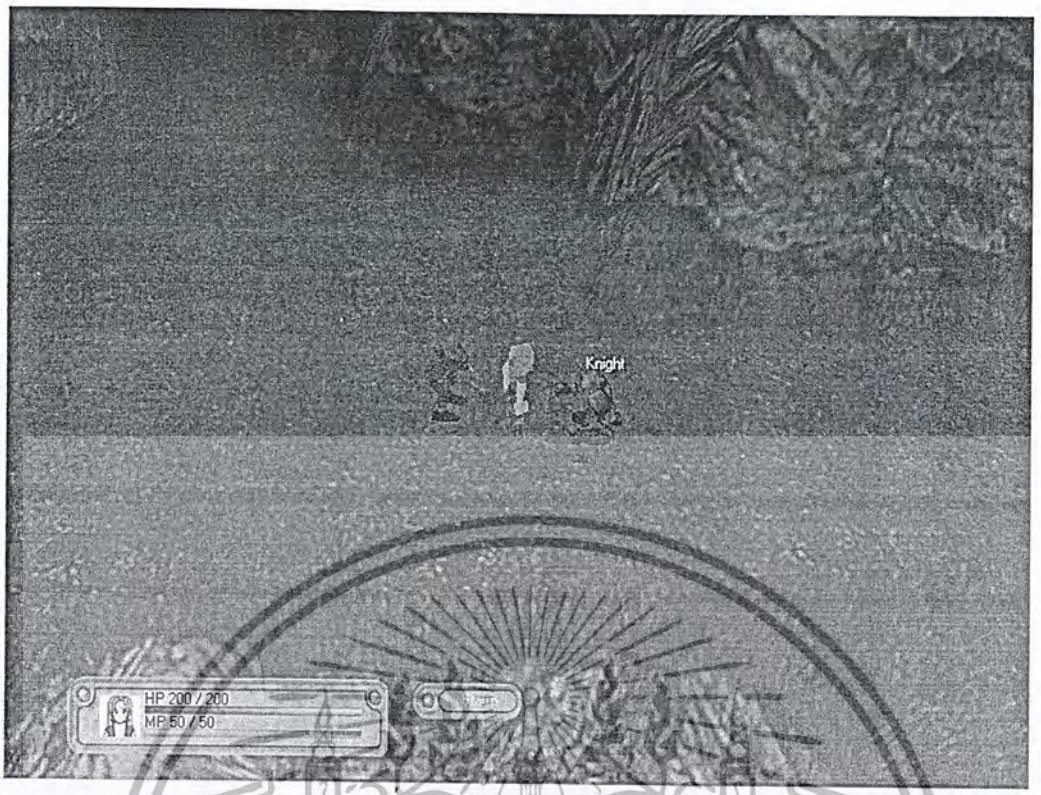
รูปที่ 6-8 เป็นฉากเริ่มต้นของเกม



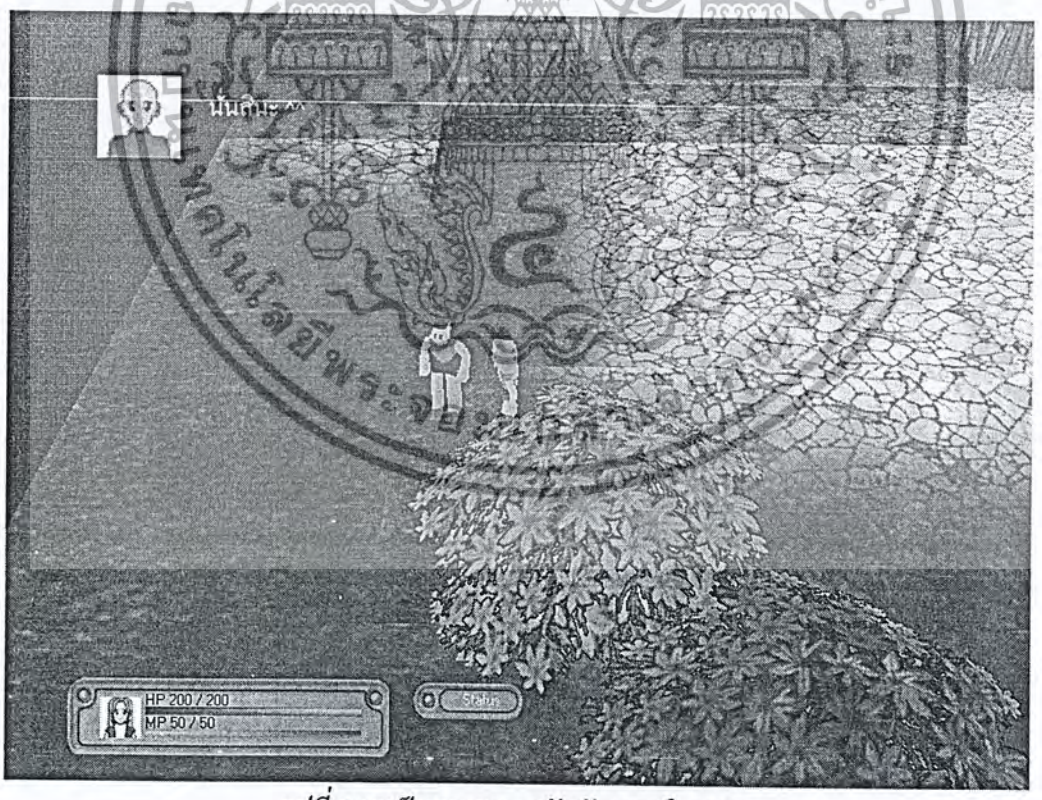
รูปที่ 6-9 เป็นฉากที่ตัวเอกเข้ามาในเกมส์



รูปที่ 6-10 เป็นฉากที่เกิดหมอกขึ้น



รูปที่ 6-11 เป็นฉากต่อสู้ในเกม



รูปที่ 6-12 เป็นฉากพูดคุยกับตัวละครในเกม

สรุปผลการทดลองที่ได้จากการทดลองเล่นเกมที่พัฒนาด้วย GUI Game Engine

เกมที่ได้ทดลองเล่นมีความสมบูรณ์ตามขอบเขตที่ได้บอกไว้ข้างต้นทุกประการ คือมันสามารถทำตามสิ่งที่ต้องการที่ได้สร้างไว้ในส่วน GUI ได้ทุกส่วน จากการทดลองพัฒนาเกมด้วยGUI นี้ ได้แสดงให้เห็นแล้วว่าสามารถสร้างเกมได้อย่างรวดเร็วโดยที่ไม่ต้องมีการcodingเลย

บทที่ 7
บทวิจารณ์และสรุป

7.1 สรุปผลการวิจัย

จากการได้ทดลองสร้างเกมจาก เอนจินนี้ เกมที่ได้สร้างสามารถที่จะเล่นได้ ผู้ใช้สามารถพัฒนาเกมได้อย่างรวดเร็ว โดยไม่ต้องเขียนโปรแกรมเอง การแก้ไขรายละเอียดต่างๆภายในเกมได้สะดวก ส่วนที่เพิ่มขึ้นมาจากปีก่อนคือ สามารถเพิ่มแผนที่ ตัวละครและวัตถุต่างใน libary ได้ สามารถที่จะปรับค่าต่างๆของตัวละครได้

ตลอดเวลาที่ได้พัฒนาเกมเอนจินต์ขึ้นมานั้นมีอุปสรรคและปัญหามากมายรวมทั้งเวลาอันจำกัด การศึกษา scipt และ code ต่างๆของรุ่นพี่ และ ข้อมูลที่มีอยู่น้อยนิด

7.2 แนวทางในการพัฒนาต่อ

- เพิ่มชุดคำสั่งใน Action Library เพื่อให้มีรูปแบบของการกระทำมากยิ่งขึ้น
- เพิ่มส่วนที่ติดต่อกับระบบ network
- เพิ่มความความหลากหลายในแนวเกมที่ GUI จะสร้างออกมาได้ เพื่อทำให้เกมดูหลากหลายมากยิ่งขึ้น
- เพิ่มค่าความสามารถของ ตัวละครต่างๆเช่น ให้ตัวละครสามารถใช้ skill ได้ ให้ตัวละครสามารถใช้ไอเทมได้

บรรณานุกรม

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

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

[3] Mark DeLoura (2000): "Game Programming Gems", Charles River Media, 8 2000.

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

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