การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้ (1)

Graphic User Interface 3D RPG Game Engine (1)

นาย พงษ์ศักดิ์ ธรรมวิริยะกุล
นาย ปฐมโชค จรูญพรพงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2545

การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้ (1)

Graphic User Interface 3D RPG Game Engine (1)

โดย

นาย พงษ์ศักดิ์ ธรรมวิริยะกุล

นาย ปฐมโชค จรูญพรพงศ์

อาจารย์ที่ปรึกษา

ดร.วรวัฒน์ ลิ้มโภคา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2545

ปริญญานิพนธ์ปีการศึกษา 2545

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้ (1)

Graphic User Interface 3D RPG Game Engine (1)

ผู้จัดทำ

1. นาย พงษ์ศักดิ์ ธรรมวิริยะกุล รหัสประจำตัว 42010215
2. นาย ปฐมโชค จรูญพรพงศ์ รหัสประจำตัว 43015368

..... อาจารย์ที่ปรึกษา

(ดร. วรวัฒน์ ลิ้มโภคา)

การพัฒนาเครื่องมือสร้างเกมประเภท RPG 3 มิติ แบบเป็นหน้าต่างติดต่อกับผู้ใช้ (I)

นาย พงษ์ศักดิ์ ธรรมวิริยะกุล 42010215
นาย ปฐมโชค จรูญพรพงศ์ 43015368
ดร.วรวัฒน์ ลิ้มโภคา อาจารย์ที่ปรึกษา
ปีการศึกษา 2545

บทคัดย่อ

ในการพัฒนาเกมปัจจุบันจำเป็นต้องอาศัยความรวดเร็วในการพัฒนา เนื่องจากในธุรกิจประเภทนี้เกิดการแข่งขันกันสูงมาก ดูได้จากข่าวคราวเกมใกม่ๆที่ออกมาแทบทุกสัปห์ดา การที่สร้างเกมได้อย่างรวดเร็วเช่นนี้จำเป็นที่จะต้องมีเครื่องมือที่มีประสิทธิภาพที่ทำให้ผู้พัฒนาลดเวลาในการเขียนโปรแกรมหรืออาจจะไม่ต้องเขียนโปรแกรมเลย ซึ่งทำให้สร้างเกมโดยเน้นที่ไปที่เนื้อหาของเกมแทน

โครงการนี้จึงเกิดขึ้นโดยมีเป้าหมายเพื่อการสร้างเกมโดยที่ผู้สร้างไม่จำเป็นต้องมีความรู้ในด้านการเขียนโปรแกรมเลย โดยจะสร้างเป็นแอพพลิเคชั่นซึ่งผู้ใช้สามารถกำหนดสิ่งต่างๆลงไปในเกมได้ตั้งแต่การเลือกโมเดลไปจนถึงการสร้างเนื้อเรื่องให้กับเกม ซึ่งผลลัพธ์สุดท้ายก็คือเกมที่สามารถเล่นได้จริงที่มีเรื่องราวเหมือนกับที่เราได้วางไว้ตั้งแต่แรก

Graphic User Interface 3D RPG Game Engine (1)

Pongsak thamviriyakul 42010215
Pathomchoke Jaroonponpong 43015368
Dr. Worawat Limpoka Advisor

ABSTRACT

Now, Game development is high competition business. Then Game developers have require
efficiency tools which fast create game in short time such as motion capture is tool for create animation model in 3D game by simulation motion of human.

This project is aim to create application tool for create game 3D RPG by graphics user interface which doesn't require programming ability from user. User will see real graphics which user want to see in his game and can create own story of game with this application.

User interface of application is designed to easy to learn but users should have sense of gamer for create game. Final product of project is game 3D RPG that has story follow by user requires.

สารบัญ

สารบัญ (ต่อ)

หน้าที่

สารบัญ (ต่อ)

สารบัญภาพ

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

สารบัญภาพ (ต่อ)

หน้าที่

บทที่1
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 เพื่อให้กรอบคลุมกลุ่มผู้ใช้งานให้ได้มากที่สุด 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

ทางบริษัทไมโครซอฟต์ได้เล็งเห็นว่าแอพพลิเคชันทางด้านเกมนั้นมีผู้ใช้งานกันอย่างแพร่หลาย จึงมีแนวคิดที่จะให้ผู้พัฒนาแอพพลิเคชันทางด้านเกมและมัลติมีเดียหันมาพัฒนาบนระบบปฏิบัติการ Windows ดังนั้นบริษัทไมโครซอฟต์จึงพัฒนาไดเร็กเอ็กซ์ (DirectX) ขึ้นมา ซึ่งเป็นไลบรารีทางด้านมัลติมีเดียที่มีจุดมุ่งหมายหลัก ๆ 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. เพื่อศึกษาการทำงานของไดเร็กเอกซ์และนำไปใช้ในการพัฒนาเกมแอพพลิเคชัน
2. เพื่อพัฒนาชุดคำสั่งและนำไปสร้าง เกมเอนจินเพื่อให้สามารถพัฒนาเกมได้สะดวกยิ่งขึ้น
3. สามารถแสดให้เห็นถึงการนำไปใช้งานได้จริงของเกมเอนจินแบบ library ที่สร้างขึ้นในหลากหลายแนวทาง
4. เพื่อศึกษาและทดลองสร้างเกมเอนจินแบบ GUI (Graphic User Interface) ซึ่งเป็นเกมเอนจินที่จะติดต่อกับผู้ใช้ผ่านหน้าต่างที่ใช้ติดต่อโดยมีลักษณะเป็นกราฟฟิค

1.3 ขอบเขตของโครงงาน

โครงงานที่ได้จัดทำขึ้นนี้เป็นการพัฒนาซอฟท์แวร์ โดยรวบรวมชุดคำสั่งของไดเร็กเอ็กซ์ที่มีลักษณะการทำงานร่วมกันมารวมเข้าไว้ด้วยกัน ซึ่งจะได้ชุดคำสั่งที่ช่วยให้เกิดความสะดวกและรวดเร็วในการพัฒนาซอฟท์แวร์โดยเฉพาะอย่างยิ่งการพัฒนาเกม โดยเราเรียกชุดคำสั่งเหล่านี้ว่าเกมเอนจิน โดยเราจะทำการแบ่งฟังก์ชันหลักภายในเกมเอนจินออกเป็นส่วนย่อย เนื่องจากแต่ละส่วนนั้นมีความชัดเจนในการทำงานต่างกันออกไป ดังนั้นแต่ละกลุ่มจะทำการพัฒนาส่วนย่อยต่างๆ นี้แล้วนำมารวมเข้าไว้ด้วยกัน เป็นชุดคำสั่งไลบรารี เพื่อการทำงานในการพัฒนาเกมที่สมบูรณ์

เมื่อพัฒนาเกมเอนจินในระดับแรกเสร็จเรียบร้อยแล้ว จะมีการนำเกมเอนจินที่ได้นี้ไปประยุกต์และพัฒนาแอพพลิเคชันต่างๆ ซึ่งจะพัฒนาต่อเป็นเกมเอนจินประเภท GUI (Graphic User Interface) โดยจะมี Interface ที่ใช้ติดต่อกับผู้ใช้ในรูปแบบ Graphic ซึ่งจะเป็นเครื่องมือที่ช่วยให้ผู้ที่ต้องการพัฒนาเกมสามารถใช้ในการพัฒนาเกมได้โดยสะดวก

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทักษะและความสามารถในการพัฒนาเกม โดยสามารถสร้างเกม 3 มิติได้
2. ความรู้ทางคณิตศาสตร์และอัลกอลิทึมในการสร้างเกม เช่น การหมุนภาพ การสร้างภาพเคลื่อนไหว การใช้ทรีในการเพิ่มประสิทธิภาพในการแสดงผล
3. เทคนิคต่างๆ ที่ใช้ในการสร้างเกม 3 มิติ
4. การออกแบบโปรแกรมเชิงคอมโพเน็นต์
5. การใช้ไดเร็กเอ็กซ์ในการสร้างแอพพลิเคชันใช้งาน และติดต่อกับอุปกรณ์มัลติมีเดีย
6. ความรู้ทางภาษา C++ ที่ใช้ในการพัฒนาเกมแอพพลิเคชัน

1.5 วิธีในการดำเนินงาน

1. ทำการศึกษาและค้นคว้าความรู้ทางด้านการเขียนเกมด้วยไดเร็กเอ็กซ์จากหนังสือและอินเตอร์เน็ต
2. ศึกษาแนวทางการพัฒนาและความรู้ที่ต้องใช้เพิ่มเติมจากความรู้พื้นฐาน
3. เลือกและศึกษาเครื่องมือที่ใช้ในการพัฒนา
4. ออกแบบโครงสร้างและแบ่งงานตามกลุ่มตามงานที่ได้รับมอบหมาย
5. สร้างเกมเอนจินในแต่ละส่วนตามโครงสร้างที่ออกแบบไว้
6. ทำการทดสอบเกมเอนจินที่ได้สร้างขึ้น
7. ทำการรวบรวมส่วนต่างๆ ของเกมเอนจินจากแต่ละกลุ่มให้กลายเป็นเกมเอนจินที่สามารถนำไปใช้งานได้
8. วางโครงสร้างเกมเอนจินแบบ GUI โดยนำเอาเกมเอนจินที่ได้พัฒนาไปใช้ในการพัฒนา GUI
9. สร้างโมเดล 3 มิติและสร้างภาพกราฟิกที่ต้องใช้งานในเกมเอนจิน
10. ศึกษาวิธีการทำงานของเกมเอนจินแบบ GUI และศึกษาอัลกอลิทึมที่จะนำมาใช้
11. หาทฤษฎีและออกแบบเกมเอนจินในลักษณะ GUI
12. สร้างเกมเอนจินในลักษณะ GUI ตามโครงสร้างที่ได้ออกแบบไว้
13. ทดสอบการทำงานของเกมเอนจินที่ได้สร้างขึ้นมา และหาข้อผิดพลาดของเกมเอนจินที่พัฒนาขึ้น
14. ทำการรวบรวมผลทดสอบ วิเคราะห์ปัญหาที่เกิดขึ้น และหาแนวทางในการแก้ปัญหาเพื่อนำมาสรุปเป็นข้อมูลในการพัฒนาขั้นต่อไป

บทที่ 2
แนะนำไดเร็กเอ็กซ์

2.1 ไดเร็กเอ็กซ์คืออะไร

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

2.2 จุดมุ่งหมายของไดเร็กเอ็กซ์

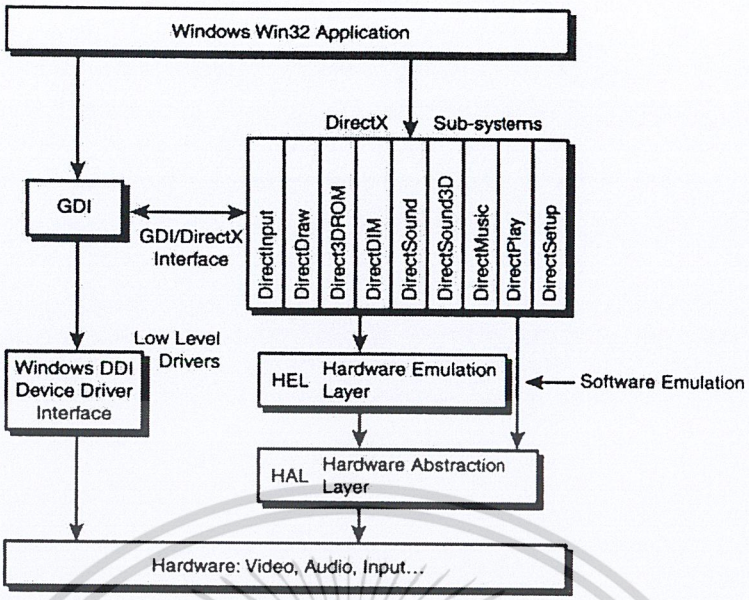
เป้าหมายที่สำคัญที่สุดของไดเร็กเอ็กซ์ คือการจัดหาชุดคำสั่งและเครื่องมือให้แก่นักพัฒนาเพื่อให้นักพัฒนาสามารถพัฒนาแอพพลิเคชันที่ทำงานเกี่ยวกับมัลติมีเดียและเกม ซึ่งใช้เครื่องคอมพิวเตอร์ที่มีระบบปฏิบัติการ Windows เป็นมาตรฐาน โดยไม่จำเป็นต้องตระหนักถึงฮาร์ดแวร์ที่รองรับชุดคำสั่งของไดเร็กเอ็กซ์ ซึ่งจะรวมถึงการสนับสนุนหน้าที่การทำงานทางมัลติมีเดียอย่างหลากหลาย

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

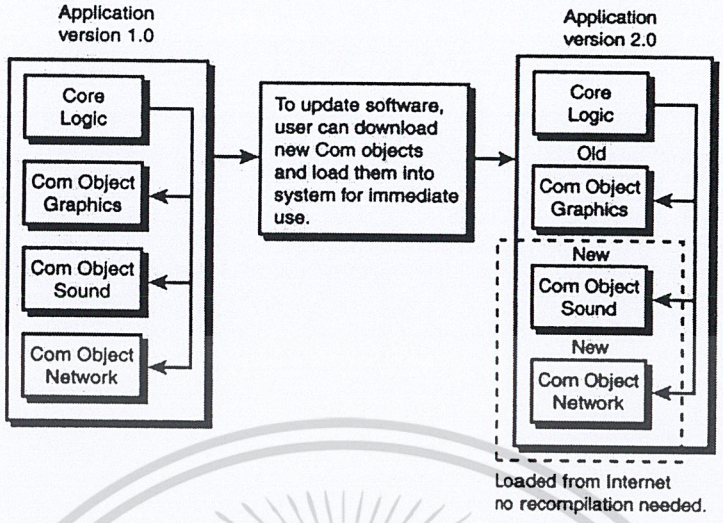
2.3.1 บทบาทของ COM

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

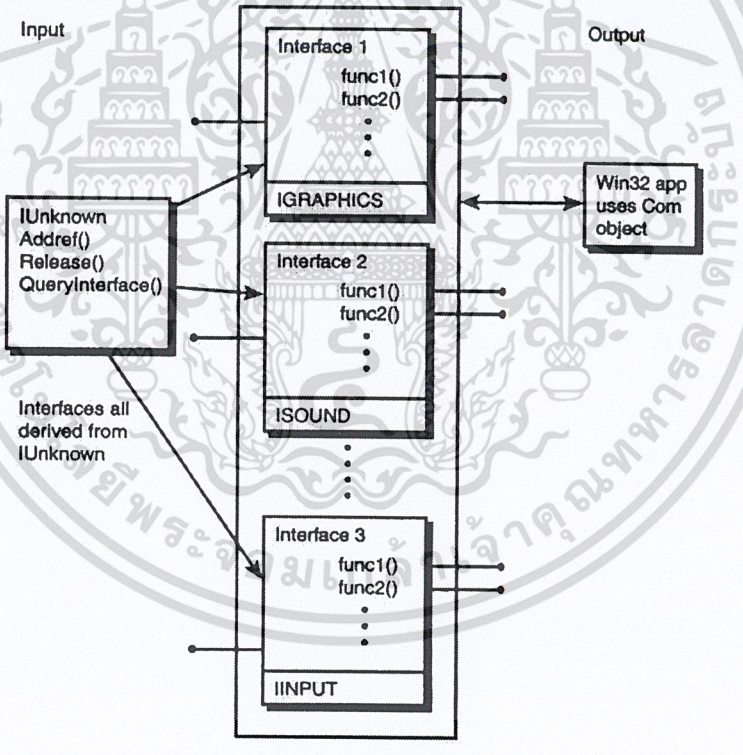
ทุกๆ COM อ็อบเจ็กต์จะต้องยึดติดกันจนเป็นโครงสร้างแบบไบนารี ซึ่งมีความคล้ายคลึงกับโครงสร้างของ virtual table ของ C++ ที่ถูกคอมไพล์แล้ว ดังนั้นจะทำให้ทุกภาษาสามารถที่จะสร้าง COM อ็อบเจ็กต์ได้ด้วยการจัดโครงสร้างให้เหมือนกับแบบไบนารีหลังจากการคอมไพล์

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

COM อ็อบเจ็กต์ที่ทำการร้องขอนั้น เรียกว่า COM ไคลเอนต์และอ็อบเจ็กต์ที่ตอบสนองการร้องขอนั้นคือ COM เซิร์ฟเวอร์ เมื่อไคลเอนต์ได้รับการเชื่อมต่อแล้วไคลเอนต์นั้นสามารถที่จะเรียกเมธอดใดก็ได้ใน COM เซิร์ฟเวอร์นั้น โดยใช้การอ้างอิงของอินเตอร์เฟซพอยเตอร์ ส่วนการเชื่อมต่ออื่นๆ จะออกไปเพื่อทำให้งานของอ็อบเจ็กต์ที่เรียกเพื่อเข้ามาใช้งานง่ายขึ้น ในการเชื่อมต่อนี้จะมีเมธอดที่ทำการแสดงรายชื่อของเมธอดที่สนับสนุน เพื่อที่จะช่วยให้เมธอดสามารถเรียกใช้งานได้ ก่อนที่จะรวมเข้าไว้ด้วยกันด้วยเทคนิคนี้จะอนุญาตให้สามารถสร้างชนิดของไลบรารีใดๆ และแอพพลิเคชัน เช่น Visual Basic ซึ่งมีข้อดีในการสร้างคุณสมบัติของ COM โปรแกรมที่มีความทนทานต่อความผิดพลาด



รูปที่ 2-2 ภาพโดยรวมของ COM



รูปที่ 2-3 อินเตอร์เฟซของ COM อ็อบเจ็กต์

2.3.2 COM และไดเร็กเอ็กซ์

ทุกๆ อินเตอร์เฟซของไดเร็กเอ็กซ์นั้นจะทำการสืบทอดมาจาก 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) ภาพบิตแมพ ซึ่งการเรียกใช้งานไดเร็กเอ็กซ์ เพื่อที่จะปรับขนาดและหมุนภาพบิตแมพได้นั้นจะต้องมีฮาร์ดแวร์ที่สนับสนุนการปรับขนาดและหมุนภาพ 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 มิติในโหมดของซอฟแวร์ได้ด้วยตนเอง

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 ไม่ว่าจะเป็นระบบแลนหรือผ่านโมเดม

- DirectX Show

  ทำหน้าที่ดูแลและจัดการเกี่ยวกับการแสดงผลภาพวีดีโอ

- DirectX Setup

  ทำหน้าที่รวบรวมแอพพลิเคชันต่างๆ ที่พัฒนาขึ้นมาด้วยไดเร็กเอ็กซ์ เพื่อให้สามารถแจกจ่ายไปยังบุคคลอื่นๆ ได้ง่าย

บทที่ 3
ทฤษฎีและหลักการเกม 3 มิติ

ในการพัฒนาโปรแกรมเกม 3 มิติ จำเป็นอย่างยิ่งที่จะต้องมีความเข้าใจถึงทฤษฎีพื้นฐานที่จำเป็นสำหรับนำไปประยุกต์ใช้ โดยเฉพาะทฤษฎีทางด้านคอมพิวเตอร์กราฟิก ซึ่งเป็นพื้นฐานสำหรับการพัฒนาโปรแกรมเกม 3 มิติ

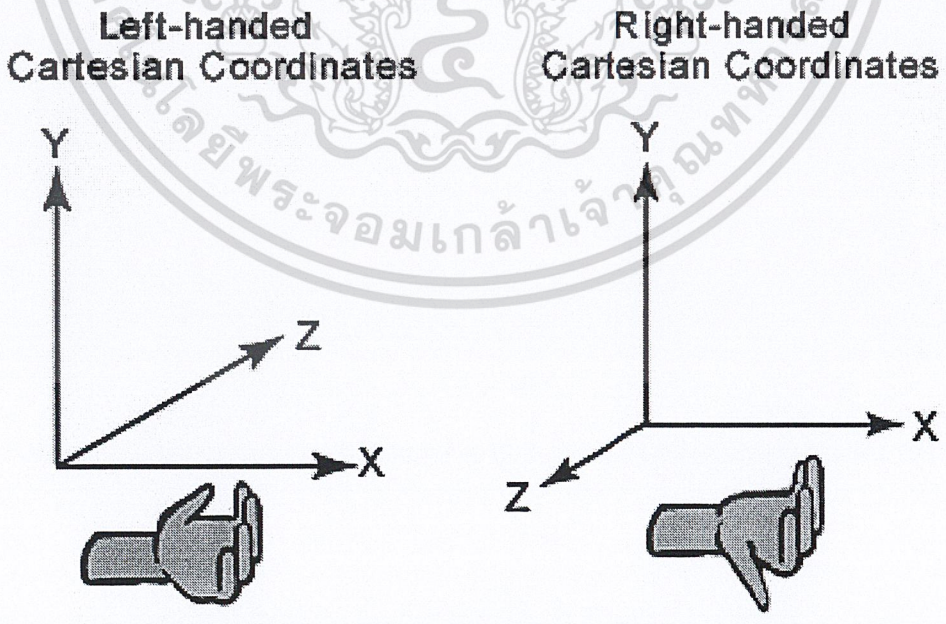
3. ระบบพิกัด 3 มิติ

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

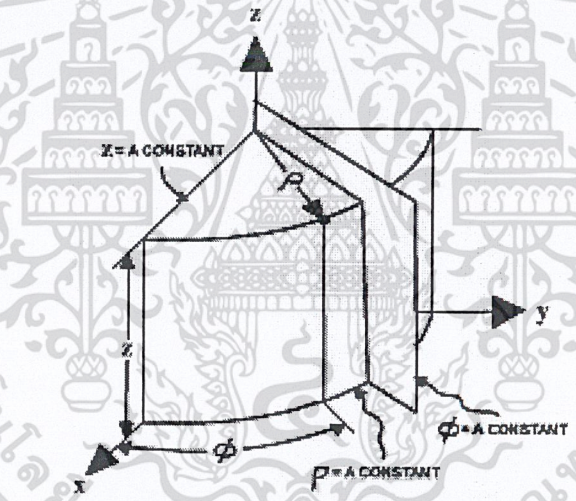


รูปที่ 3-1 แสดงระบบพิกัดคาร์ทีเซียนแบบมือซ้ายและแบบมือขวา

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)

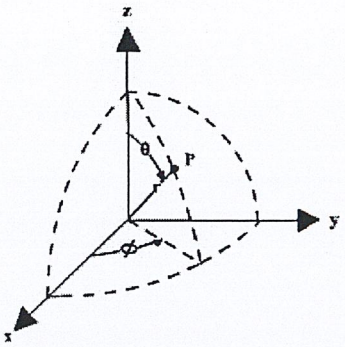
ในระบบพิกัดนี้ เราสามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด (Origin) ซึ่งแทนด้วยสัญลักษณ์ ρ มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเซียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแกนอ้างอิงในแนวระดับ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



รูปที่ 3-2 แสดงระบบพิกัดทรงกระบอก

3.1.3 ระบบพิกัดทรงกลม (Spherical Coordinate System)

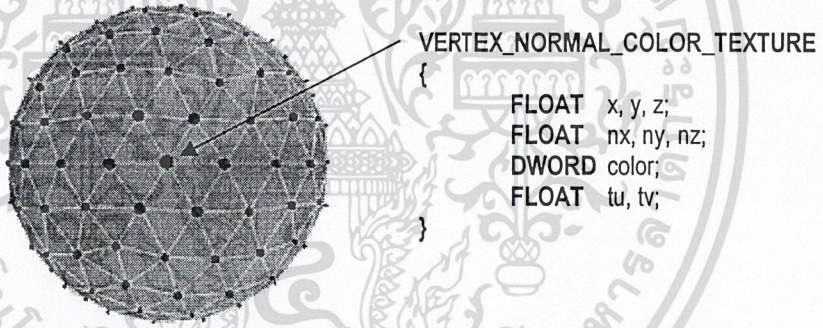
ในระบบพิกัดนี้ เราสามารถระบุพิกัดของจุดแต่ละจุดได้โดยใช้ระยะห่างระหว่างจุดนั้นๆ กับจุดกำเนิด ซึ่งแทนด้วยสัญลักษณ์ r มุมระหว่างแกนอ้างอิงในแนวระดับ (มักสัมพันธ์กับแกน X และระนาบ XY ในระบบพิกัดคาร์ทีเซียน) กับเส้นตรงที่ลากจากจุดกำเนิดไปยังตำแหน่งของจุดนั้นๆ ในระนาบของจุดกำเนิดกับแนวระดับ (ซึ่งสัมพันธ์กับระนาบ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



รูปที่ 3-3 แสดงระบบพิกัดทรงกลม

3.2 เวอร์เท็กซ์ (Vertex)

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

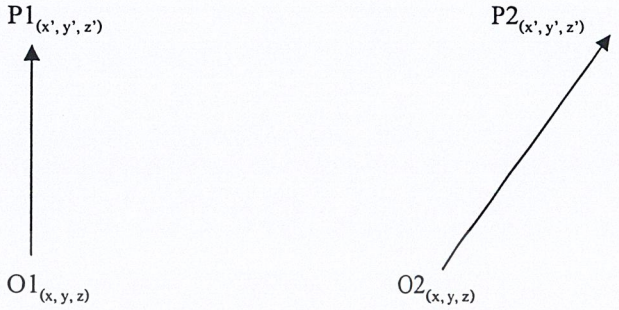


รูปที่ 3-4 แสดงตัวอย่างการนำเวอร์เท็กซ์มาใช้ในการเก็บข้อมูลที่แทนแบบจำลอง 3 มิติ

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

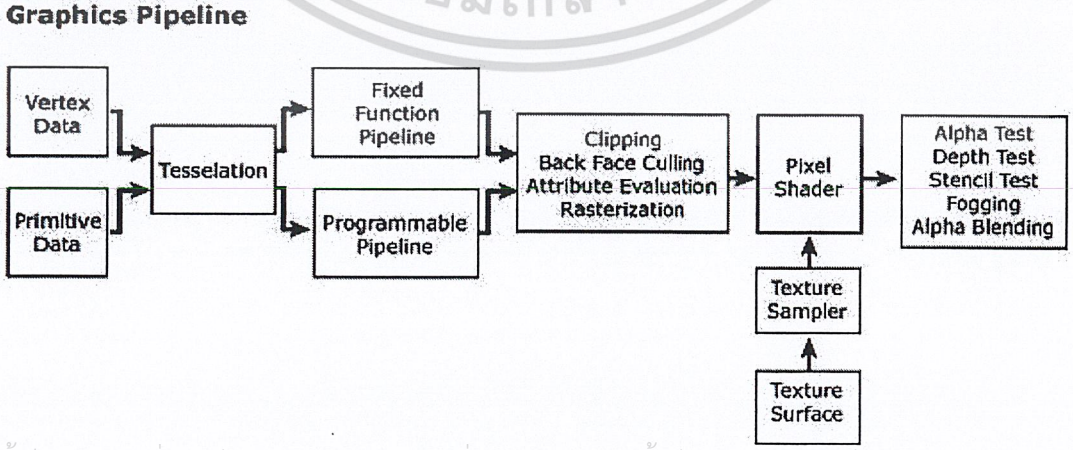
$$\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

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

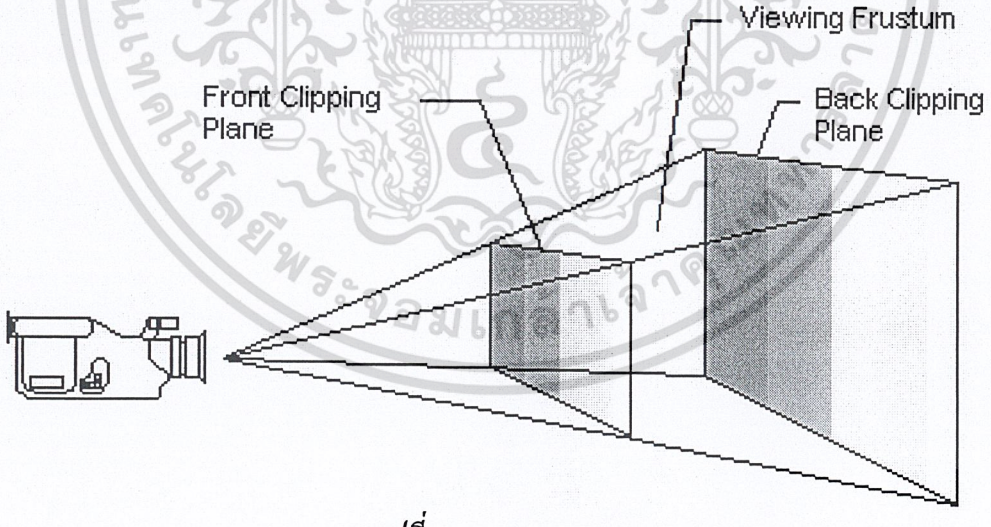
$$\begin{bmatrix} u_x & v_x & n_x & 0 \\ u_y & v_y & n_y & 0 \\ u_z & v_z & n_z & 0 \\ -(u \cdot c) & -(v \cdot c) & -(n \cdot c) & 1 \end{bmatrix}$$

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

วิวเมทริกซ์นั้นให้ข้อมูลที่จำเป็นในการปรับมุมมองของกล้องในเกมเอนจินได้ ซึ่งใช้ในการบอกตำแหน่งและทิศทางของกล้อง

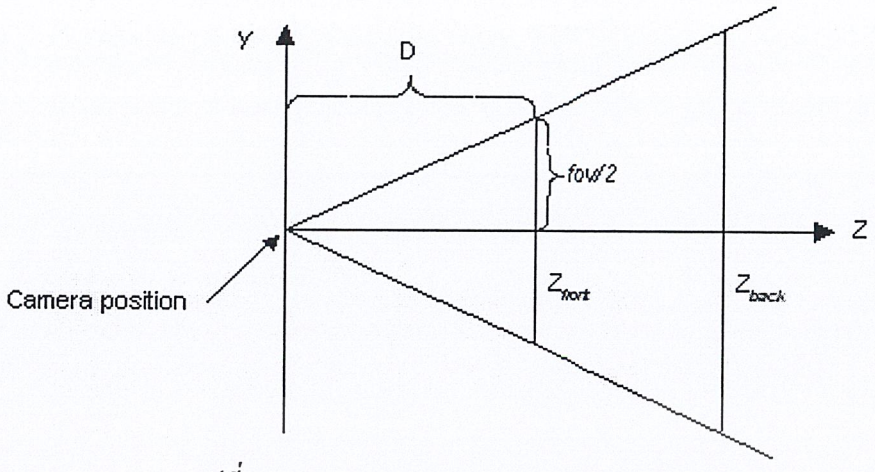
3.5.2.3 การแปรไปสู่พิกัดโปรเจ็กต์ชัน (Projection Transformation)

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

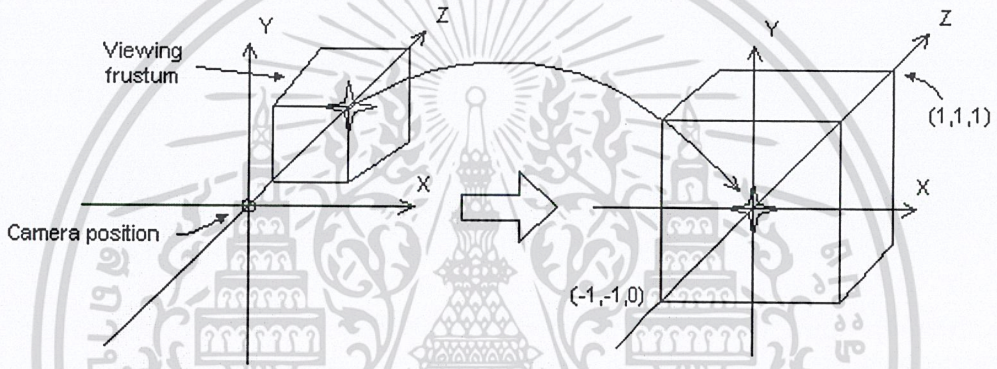


รูปที่ 3-7 Viewing Frustum

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

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

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

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

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

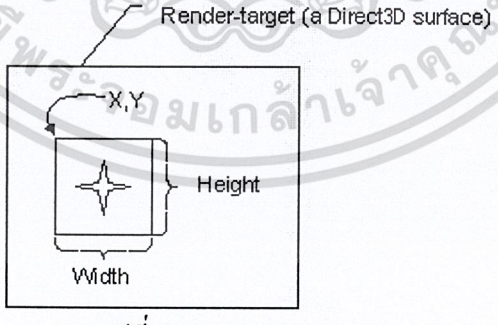
โดยทั่วไปแล้วกราฟิกเอนจินนั้นมักจะมีฟังก์ชันที่ใช้สำหรับทำการคำนวณหา Perspective Projection Matrix โดยให้เรากำหนดค่า Field of View (fov) ตามแนวความสูงของฉาก (โดยทั่วไปมักเป็นแกน Y) ค่า Aspect Ratio (w/h) ค่า Near Clipping Plane (Z_n) และค่า Far Clipping Plane (Z_f)

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 เสียงเอฟเฟคต์ต่างๆ 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

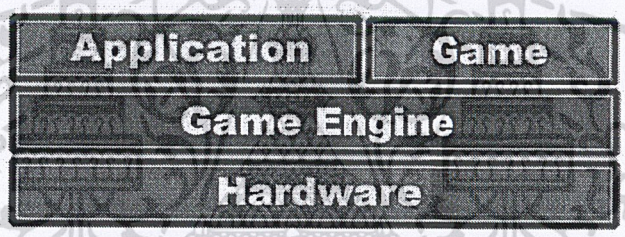
เอนจินลักษณะนี้ส่วนของเกมจะมีขนาดเล็ก เพราะจะแยกส่วนของเกมเอนจินออกไปเป็นอีกส่วนนึง ซึ่งจะไม่เข้าไปรวมกับส่วนของเกม และเป็นลักษณะที่แพร่หลายในการใช้งาน

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 ใส่เสียงเอฟเฟ็ค และเสียงแบบ 3 มิติ

4.3.5 เกมเอนจินส่วนเน็ตเวิร์ค

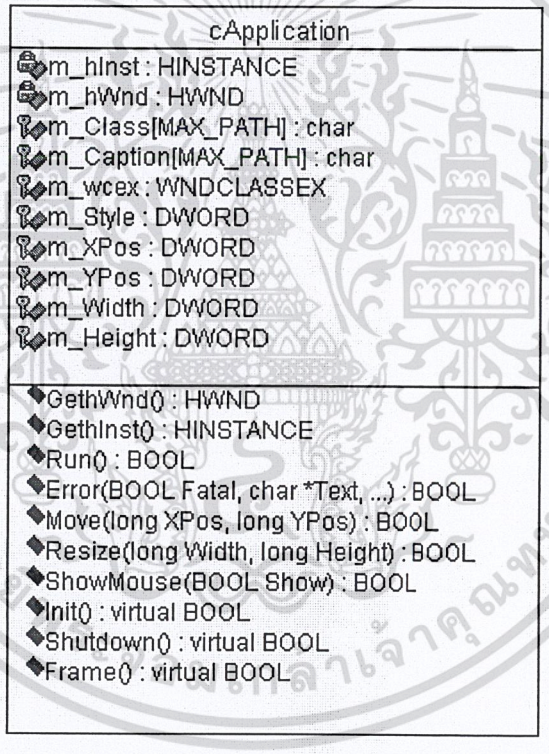
เกมเอนจินส่วนเน็ตเวิร์คจะจัดการติดต่อกับการ์ดเน็ตเวิร์ค โดยจะควบคุมกระบวนการทำงานในการติดต่อสื่อสารกันระหว่างเครื่องคอมพิวเตอร์

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

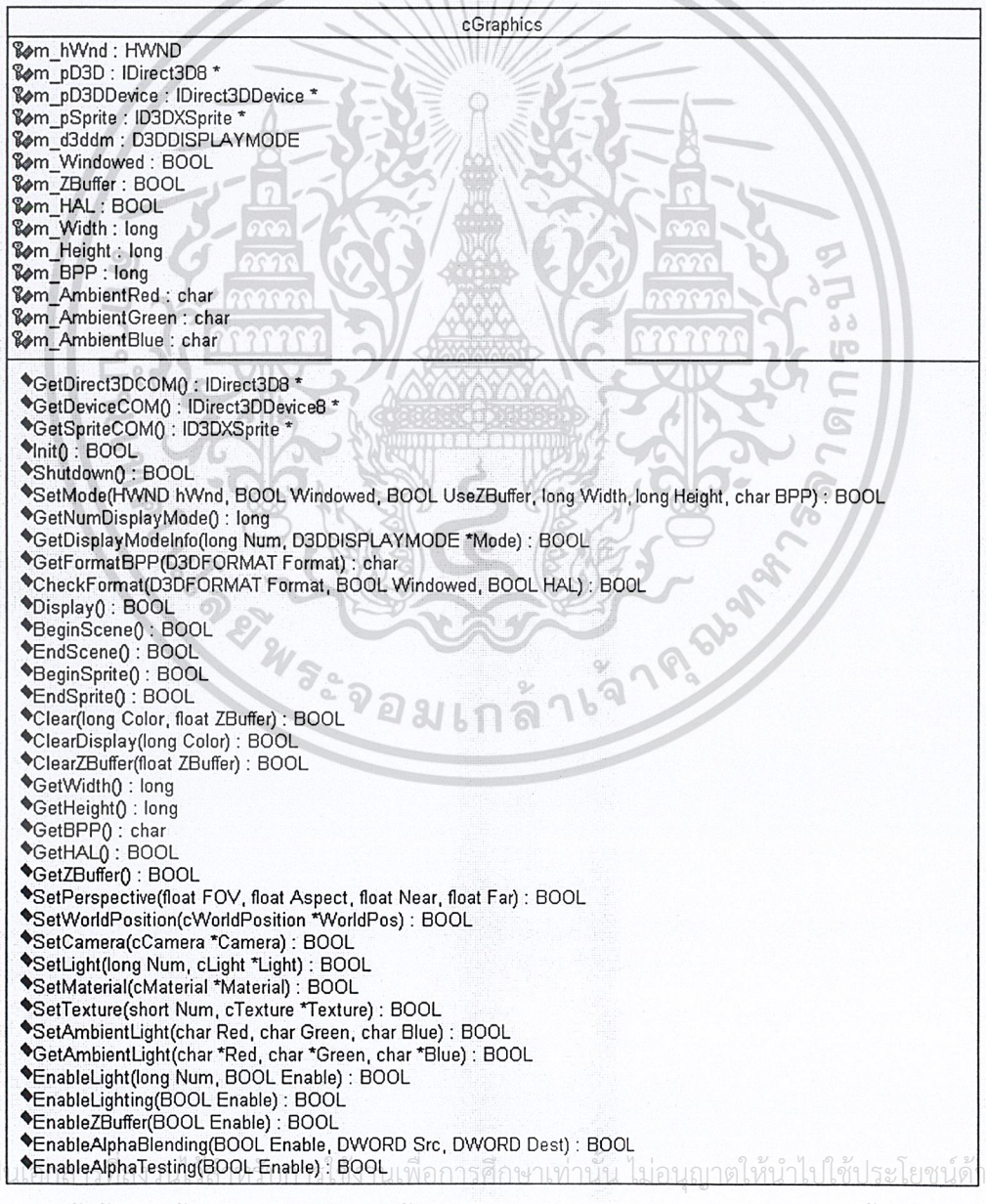
การทำงานของคลาสที่สืบทอดมาจากคลาส cApplication นั้นจะมีแค่เพียงการสร้างอินสแตนซ์ของคลาสที่สืบทอดมา และทำการเรียกเมธอด Run() เพื่อเริ่มการทำงานของแอพพลิเคชัน

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

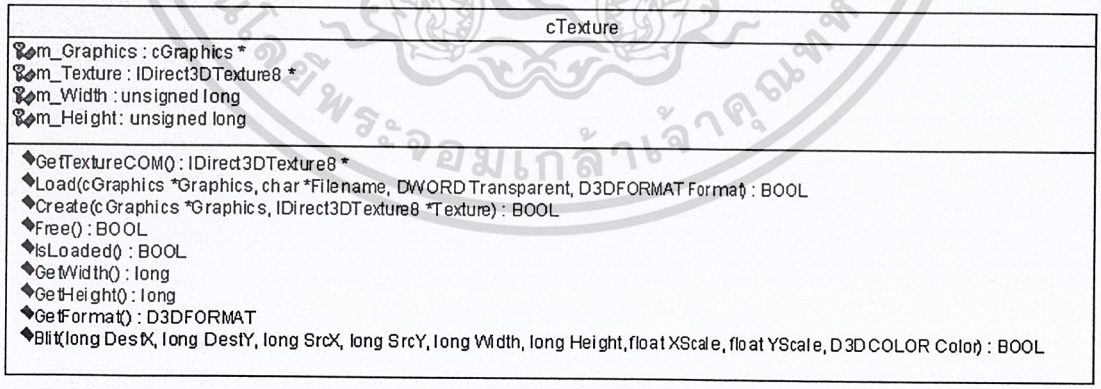
คลาสนี้จะทำการติดต่อกับอุปกรณ์แสดงผล เช่น การ์ดเร่งความเร็วกราฟิก เป็นต้น ซึ่งจะจัดการการแสดงผลในรูปแบบ 3 มิติ ซึ่งจะเริ่มการใช้งานโดยเรียกเมธอด Init() ซึ่งจะกำหนดค่าต่างๆ ให้กับสภาพแวดล้อมที่จะต้องใช้ในการแสดงผลภาพ 3 มิติ จากนั้นจะเริ่มทำงานในการแสดงผล โดยใช้เมธอด SetMode() ซึ่งจะมีลักษณะการแสดงผลอยู่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

การแสดงผลภาพ 3 มิติจะทำโดยใช้เมธอด BeginScene() เพื่อทำการเรนเดอร์ภาพลงใน BackBuffer เมื่อทำการเรนเตอร์ภาพเสร็จเรียบร้อย จะต้องทำการเรียกเมธอด EndScene() เพื่อทำการจบการทำงานในส่วนการเรนเดอร์ภาพลงใน BackBuffer จากนั้นจะทำการแสดงผลภาพ 3 มิติโดยใช้เมธอด Display() ในการสลับ BackBuffer มายัง FrontBuffer เพื่อให้ภาพที่ทำการเรนเดอร์ที่ BackBuffer แสดงผลออกทางจอภาพ ซึ่งในก่อนการเรนเดอร์ภาพใน BackBuffer จะต้องมีการลบภาพที่อยู่ใน BackBuffer ออกเสียก่อน โดยใช้เมธอด 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

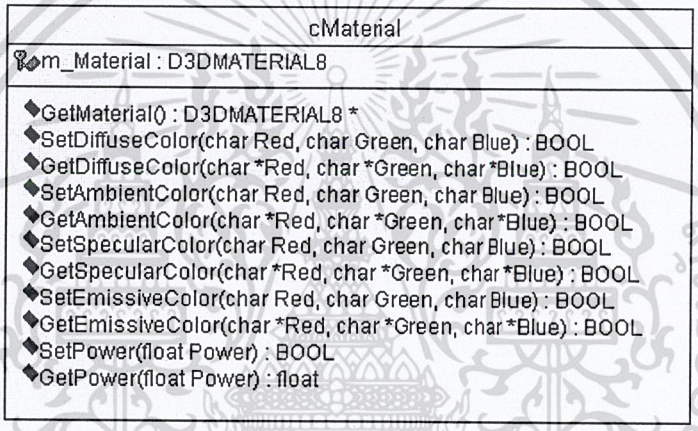
การใช้งานคลาสนี้จะมีอยู่ 2 วิธีคือถ้าต้องการโหลดภาพ Texture จากไฟล์จะใช้เมธอด Load() ซึ่งจะทำการโหลดไฟล์ขึ้นมาเป็น Texture ส่วนอีกวิธีคือ ถ้ามีการโหลด Texture ขึ้นมาแล้ว ซึ่งเก็บอยู่ในอินเตอร์เฟสของ Direct3DTexture8 ก็จะใช้เมธอด Create() ในการสร้าง Texture

คลาสนี้ยังสามารถที่จะวาด Texture ลงในส่วนแสดงผลได้โดยใช้เมธอด Blit() ซึ่งสามารถที่จะย่อขยาย และวาดยังตำแหน่งใดก็ได้ โดยกำหนดค่าลงในพารามิเตอร์ของเมธอดนี้ นอกจากนั้นยังสามารถทำให้วาดแบบโปร่งใสได้ โดยกำหนดสีที่จะเป็นสีที่เป็นสีโปร่งใส

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

6.3 คลาส cMaterial

คลาสนี้จะทำการเปลี่ยนสีที่ปรากฏอยู่บนพื้นผิวของการเรนเดอร์อ็อบเจ็กต์ ซึ่งจะทำให้อ็อบเจ็กต์ที่ถูกเรนเดอร์นั้นมีลักษณะสมจริงยิ่งขึ้น



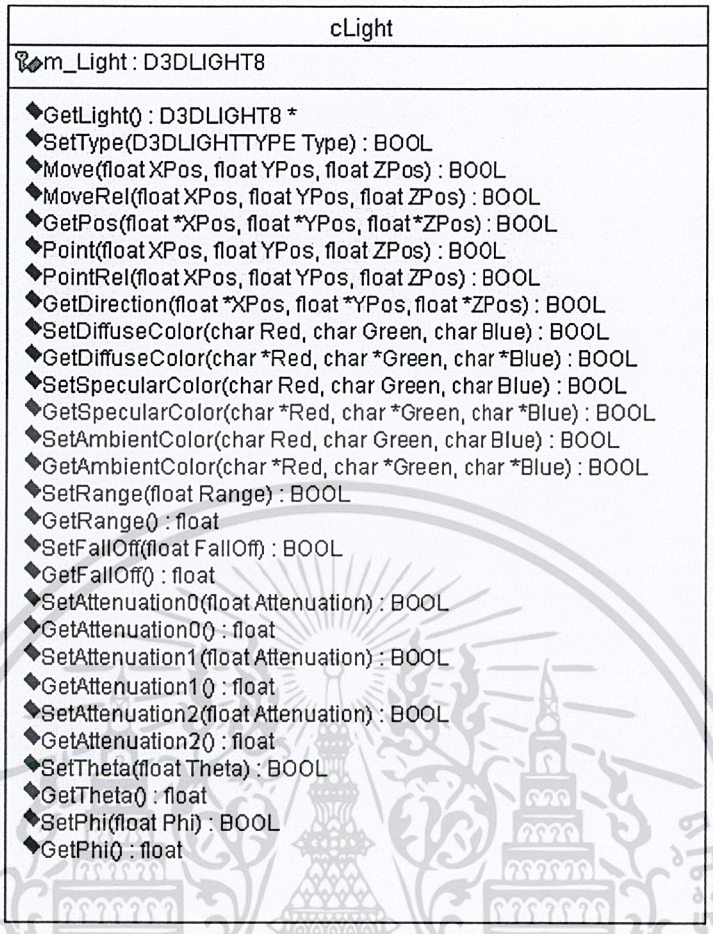
รูปที่ 6-3 คลาสไดอะแกรมของ cMaterial

คลาสนี้เพียงอินสแตนซ์เดียวจะเก็บได้เพียงโครงสร้างของ D3DMATERIAL เดียวเท่านั้น และจะมีเมธอดที่ใช้งานในการกำหนดลักษณะของสีของพื้นผิวที่จะเปลี่ยนไป โดยค่าของแต่ละสีจะมีค่าอยู่ระหว่าง 0 ถึง 255

คลาสนี้ไม่ค่อยมีการใช้งานมากนัก เนื่องจากจะใช้คลาส cTexture แทน เพราะการใช้ภาพแปะลงบนพื้นผิวของวัตถุจะดูสมจริงมากกว่าใช้สีวาดลงบนพื้นผิวของวัตถุ ซึ่งจะดูไม่สมจริงเท่า

6.4 คลาส cLight

คลาสนี้จะใช้สำหรับสร้างแสง สำหรับการเรนเดอร์ภาพให้ดูเหมือนจริงยิ่งขึ้น ซึ่งจะมีลักษณะของการสร้างแสงอยู่หลายลักษณะ



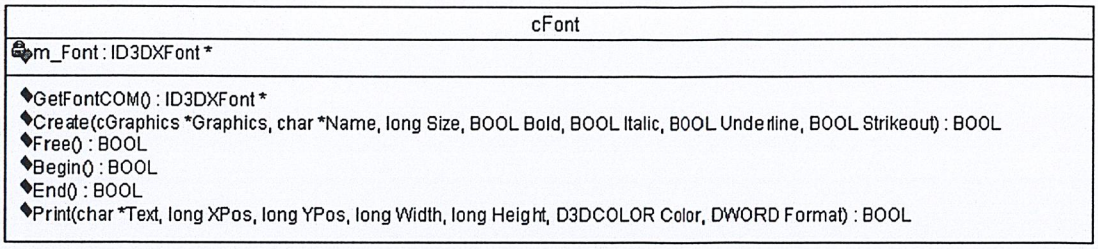
รูปที่ 6-4 คลาสไดอะแกรมของ cLight

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

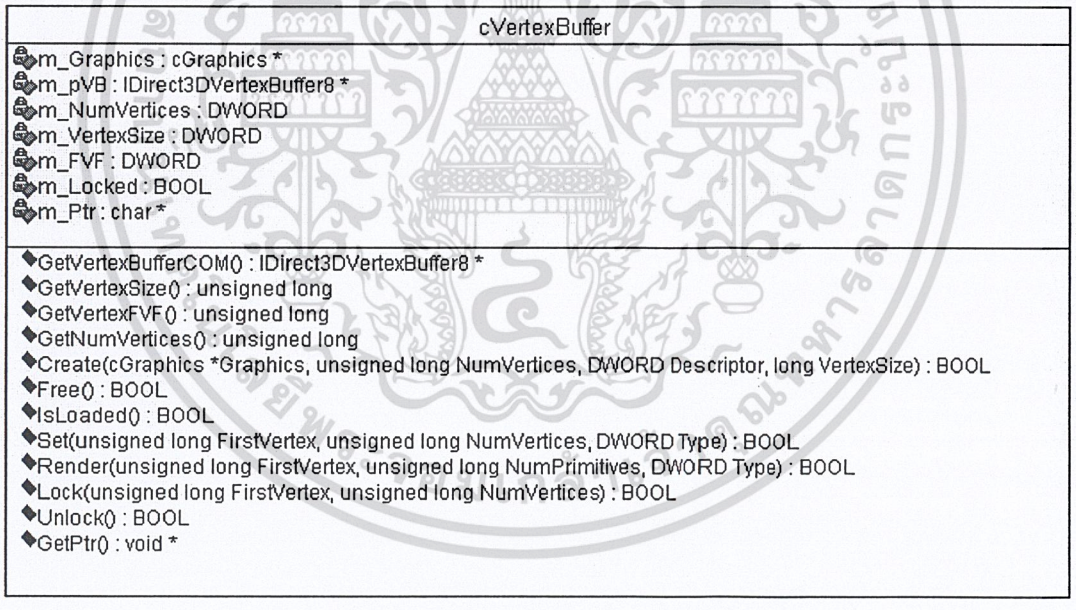


รูปที่ 6-5 คลาสไดอะแกรมของ cFont

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

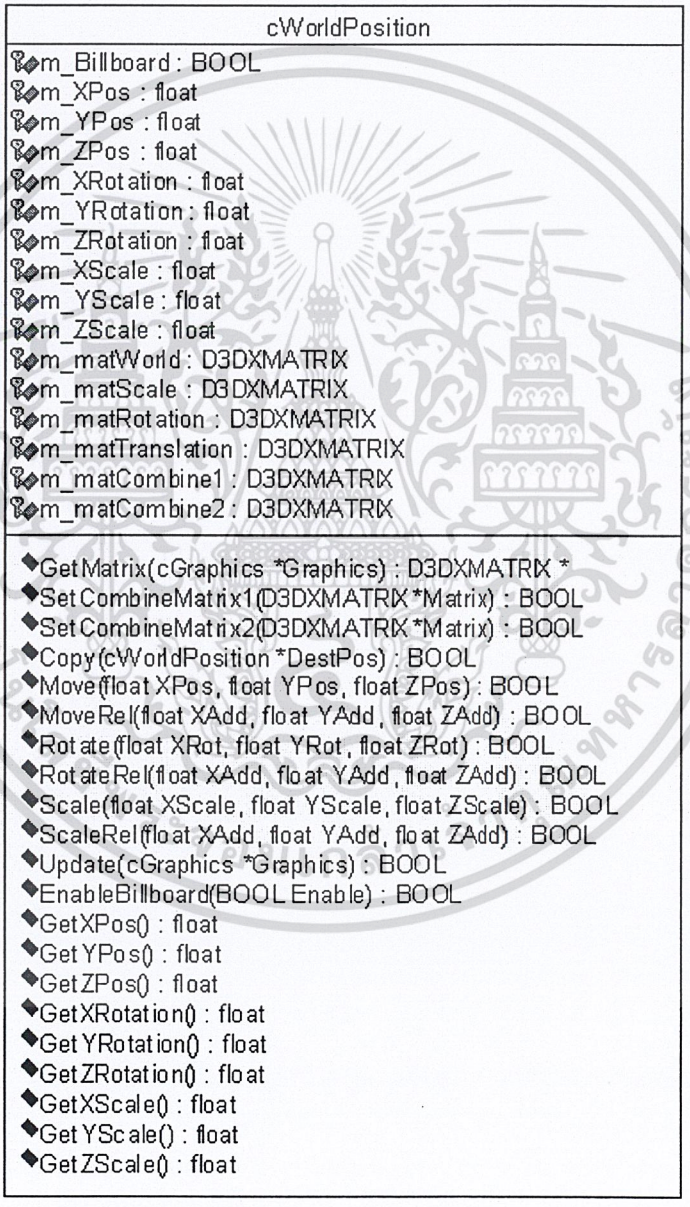
ในการใช้งาน จะทำการเรียกเมธอด Create() เป็นเมธอดแรก เพื่อทำการสร้างเวอร์เท็กซ์บัฟเฟอร์ ซึ่งเป็นหน่วยความจำที่ใช้เก็บลักษณะของเซตของจุด และเมื่อเลิกการใช้งานจะทำการคืนทรัพยากรให้กับระบบโดยใช้เมธอด Free()

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



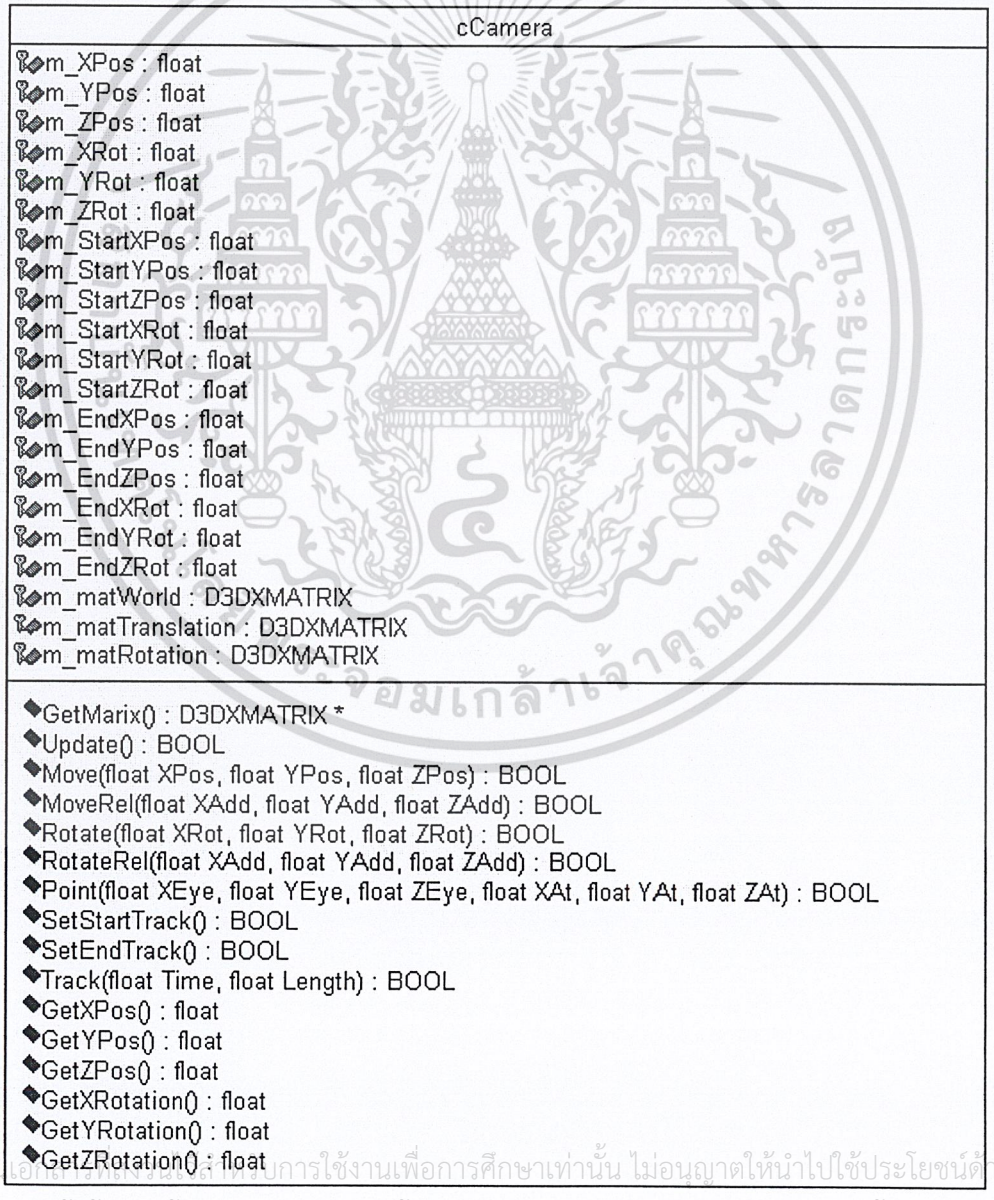
รูปที่ 6-7 คลาสไดอะแกรมของ cWorldPosition

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

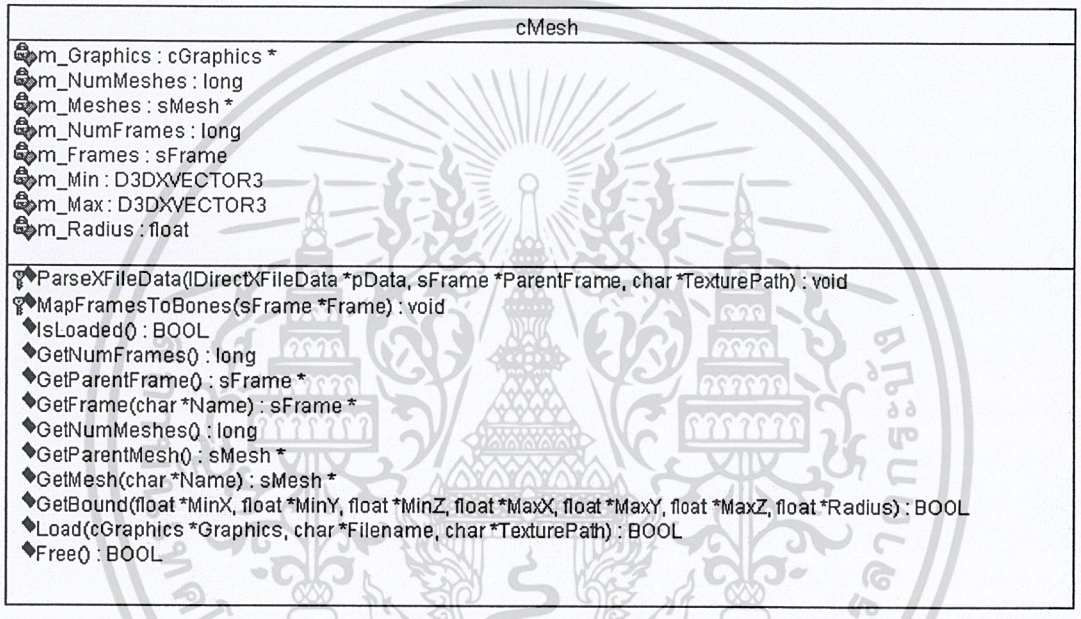


รูปที่ 6-8 คลาสไดอะแกรมของ cCamera

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

6.9 คลาส cMesh

คลาสนี้ช่วยในการจัดการโมเดล 3 มิติ ซึ่งจะใช้ตามรูปแบบไฟล์ที่มีนามสกุล X ซึ่งเป็นมาตรฐานของไดเร็กเอ็กซ์



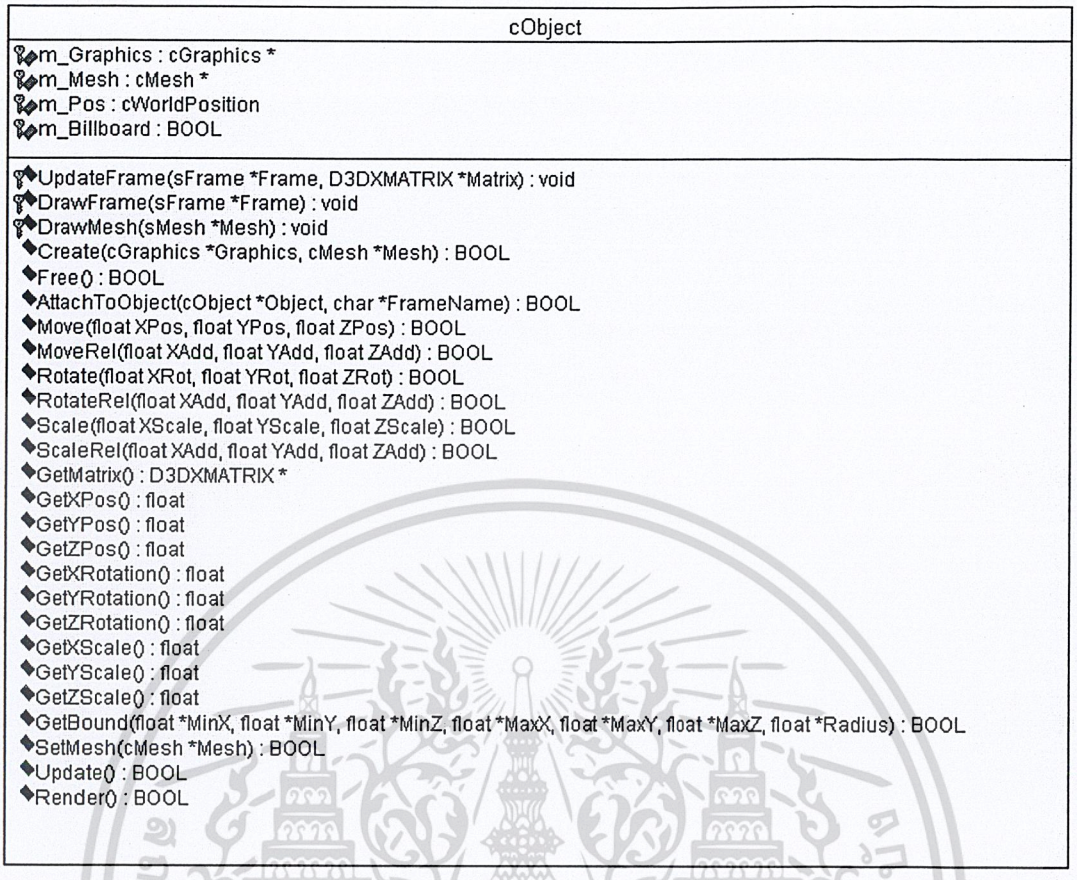
รูปที่ 6-9 คลาสไดอะแกรมของ cMesh

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

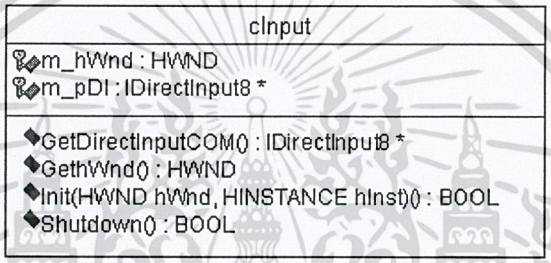
ข้อดีของการคลาสนี้ในการเรนเดอร์ก็คือ สามารถที่จะสร้างคลาส cObject ซึ่งใช้ Mesh เดียวกันได้ ทำให้ไม่เปลืองการใช้หน่วยความจำในการเก็บ Mesh และสามารถที่จะเปลี่ยนตำแหน่งโมเดล หมุนโมเดล หรือปรับขนาดได้โดยไม่กระทบอ็อบเจ็กต์อื่น

บทที่ 7
เอนจินส่วนอินพุต

เอนจินส่วนอินพุตเป็นส่วนติดต่อกับอุปกรณ์อินพุตต่างๆ เช่น คีย์บอร์ด เมาส์ จอยสติ๊ก ซึ่งสามารถที่จะควบคุมการทำงานของอุปกรณ์ต่างๆ โดยจะแบ่งออกเป็นคลาส ดังต่อไปนี้

7.1 คลาส cInput

เป็นคลาสที่จะทำการกำหนดค่าต่างๆ ที่จำเป็นต้องใช้ในการติดต่อกับอุปกรณ์อินพุต และเริ่มต้นติดต่อกับอุปกรณ์อินพุตที่ติดตั้งอยู่

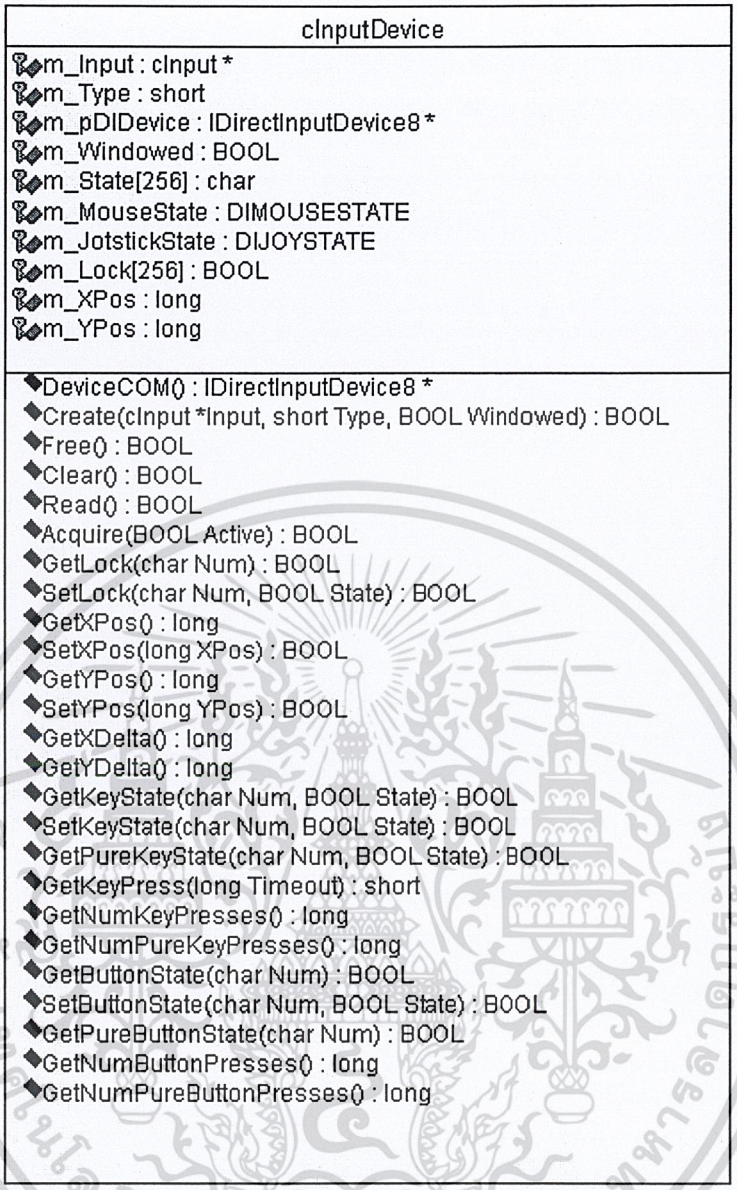


รูปที่ 7-1 คลาสไดอะแกรมของ cInput

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

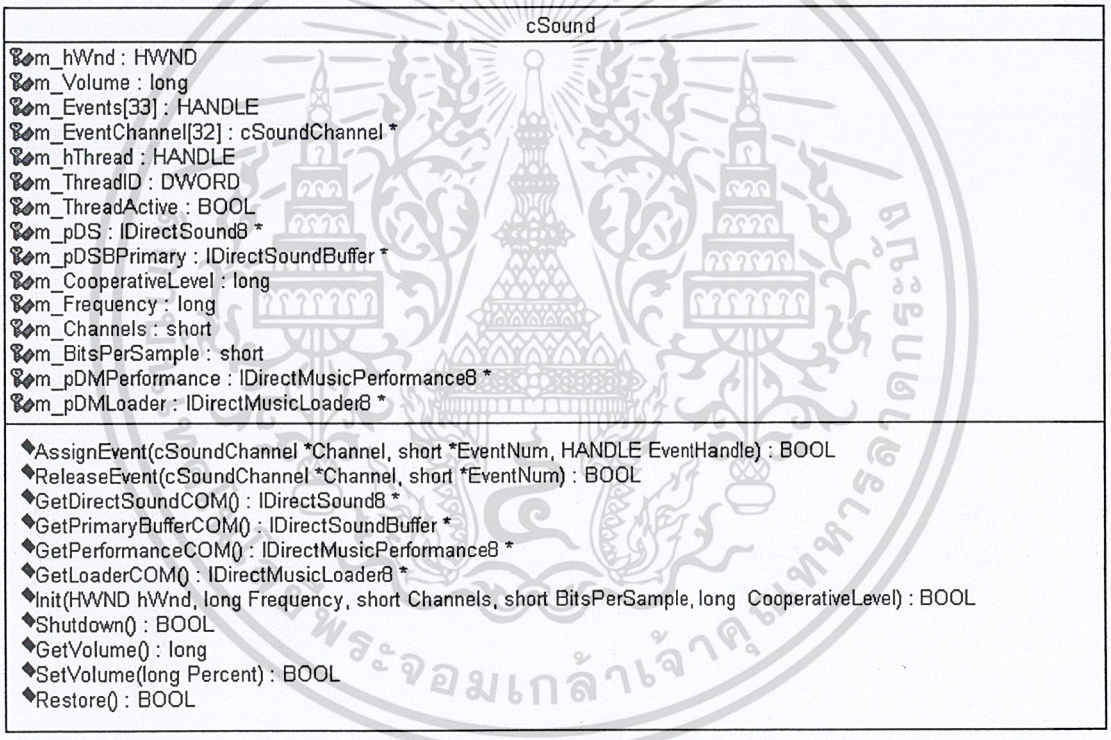
ในการอ่านค่าของปุ่มที่ถูกกด เราจะนำมาเทียบกับค่าคงที่ ซึ่งจะเป็นค่าของปุ่มต่างๆ ซึ่งค่าคงที่ของคีย์บอร์ดจะขึ้นต้นด้วย KEY_ แล้วตามด้วยชื่อของปุ่มที่ต้องการ เช่น ปุ่ม W ก็จะมีค่าคงที่เป็น KEY_W หรือถ้าต้องการปุ่ม ESC ก็จะมีค่าคงที่เป็น KEY_ESC ส่วนค่าคงที่ของเมาส์ จะขึ้นต้นด้วย MOUSE_ แล้วตามด้วยปุ่มของเมาส์ที่ต้องการ 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

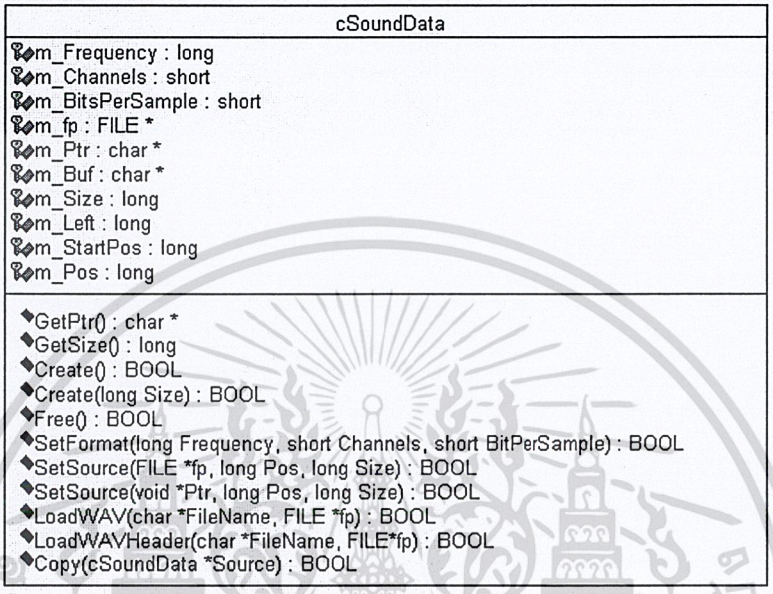


รูปที่ 8-1 คลาสไดอะแกรมของ cSound

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



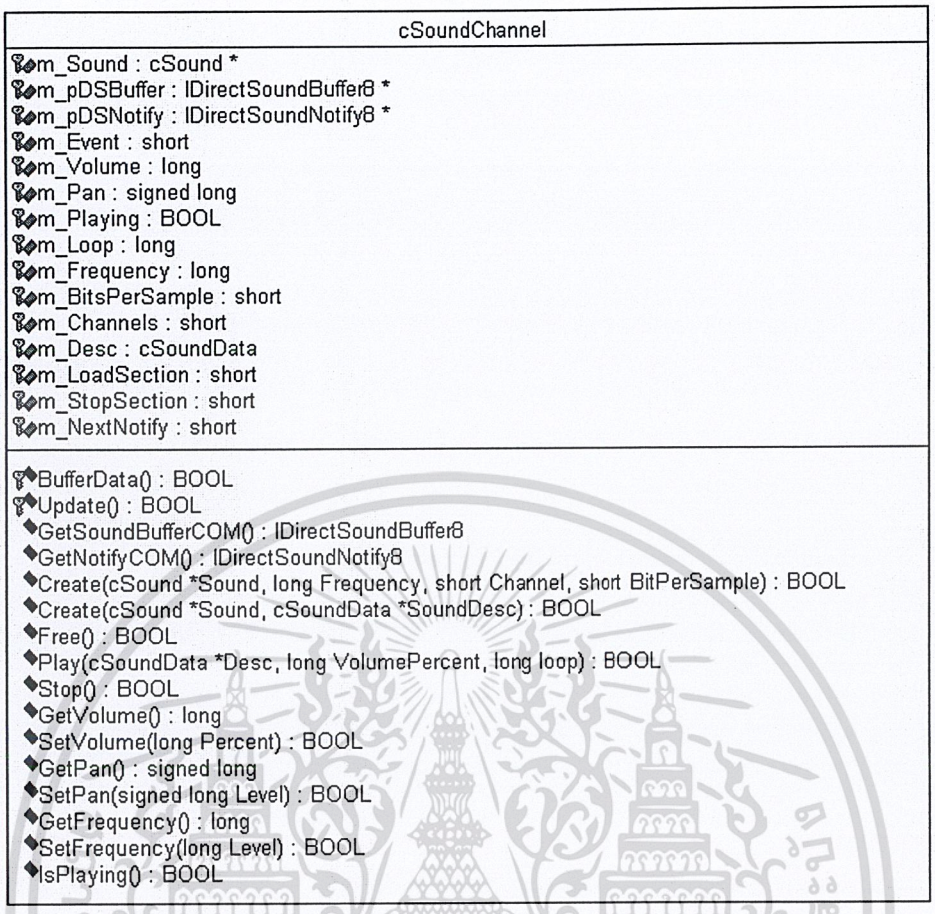
รูปที่ 8-2 คลาสไดอะแกรมของ cSoundData

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

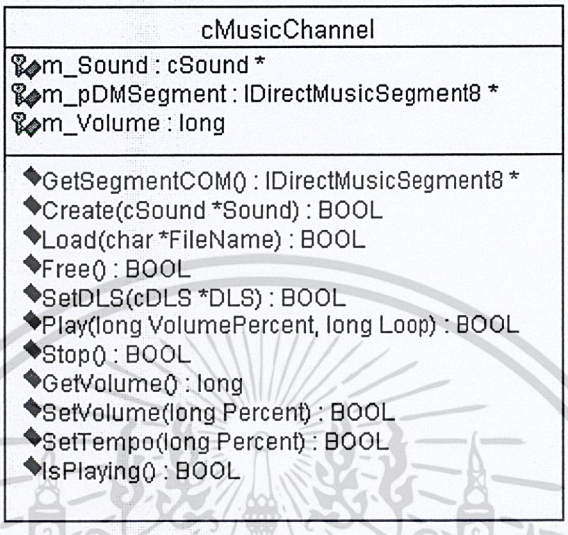
ในการใช้งานคลาส cSoundChannel จะใช้งานร่วมกับคลาส cSoundData ซึ่งใช้ในการเล่นเสียง ซึ่งเราสามารถสร้างอินสแตนซ์ของคลาสนี้ได้สูงสุด 32 อินสแตนซ์ ดังนั้นจึงสามารถสร้างเสียงได้ทั้งหมด 32 Channel ซึ่งสามารถเล่นได้พร้อมๆ กัน การใช้งานจะทำโดยเรียกเมธอด Create() ทำการสร้างรูปแบบของเสียงที่ต้องการเล่น หรือถ้าสร้างอินสแตนซ์ของคลาส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

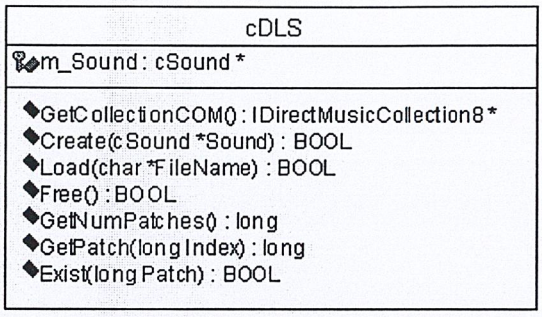


รูปที่ 8-4 คลาสไดอะแกรมของ cMusicChannel

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

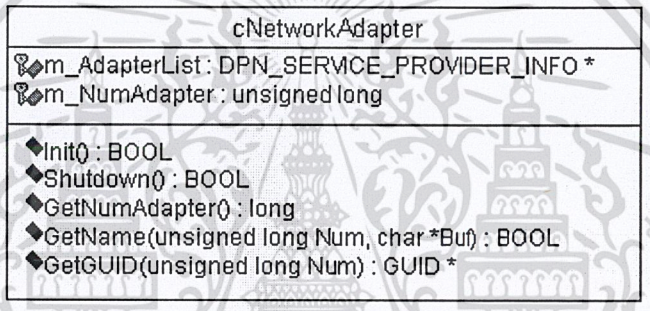
นอกจากนั้นยังมีเมธอดที่ใช้กับ Instrument ของเซตของ DLS โดยจะมีอยู่ทั้งหมด 3 เมธอดหลัก คือ เมธอด GetNumPatch() ซึ่งจะบอกว่ามี Instrument อยู่ภายในเซตของ DLS เท่าใด เมธอด GetPatch() ใช้หาหมายเลข Patch ของ Instrument ที่อยู่ในเซตของ DLS และเมธอดสุดท้ายคือ Exist() ซึ่งจะทำการตรวจสอบหมายเลข Patch ว่ามีอยู่ในเซตของ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

บทที่ 9
เอนจินส่วนเน็ตเวิร์ค

เอนจินส่วนเน็ตเวิร์คจะจัดการติดต่อกับการ์ดเน็ตเวิร์ค โดยจะควบคุมกระบวนการทำงานในการติดต่อสื่อสารกันระหว่างเครื่องคอมพิวเตอร์ และจะจัดการการทำงานในส่วนเซิร์ฟเวอร์ และไคลเอนต์ โดยมีคลาสที่เกี่ยวข้องดังนี้

9.1 คลาส cNetworkAdapter

เป็นคลาสที่จะทำการติดต่อกับการ์ดเน็ตเวิร์ค และจัดการกำหนดลักษณะการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์ เพื่อให้สามารถติดต่อสื่อสารกันได้



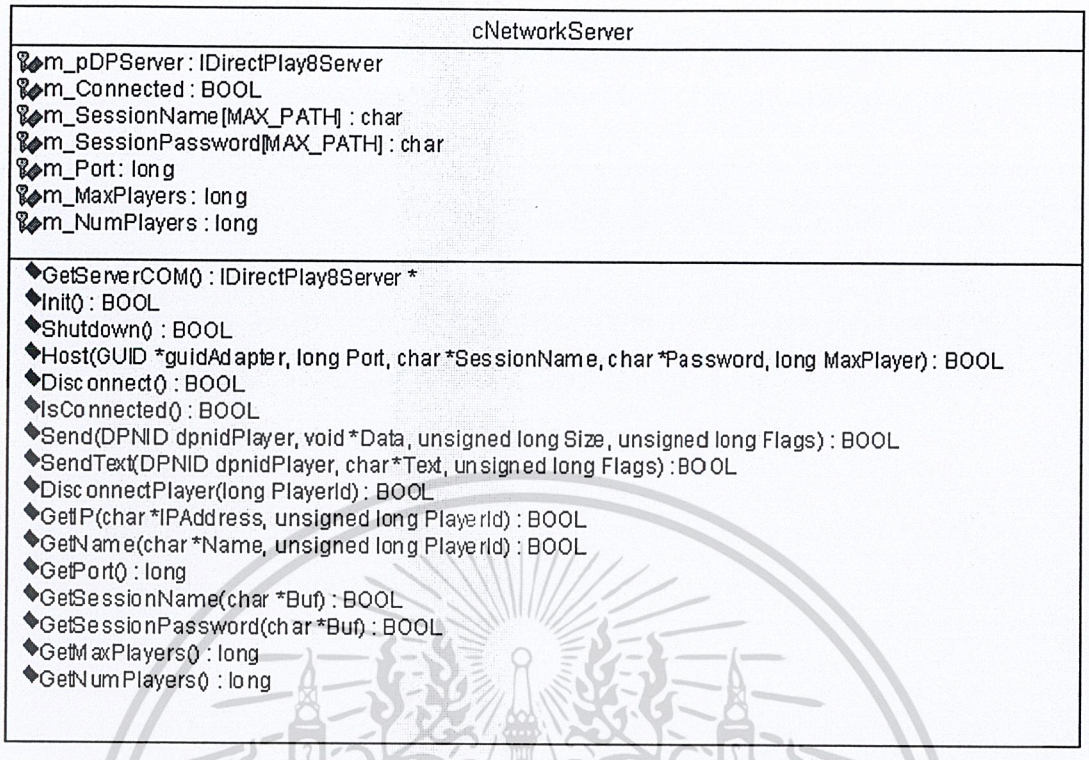
รูปที่ 9-1 คลาสไดอะแกรมของ cNetworkAdapter

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 ดูแลและจัดการเมสเสจที่เข้าและออก เป็นต้น



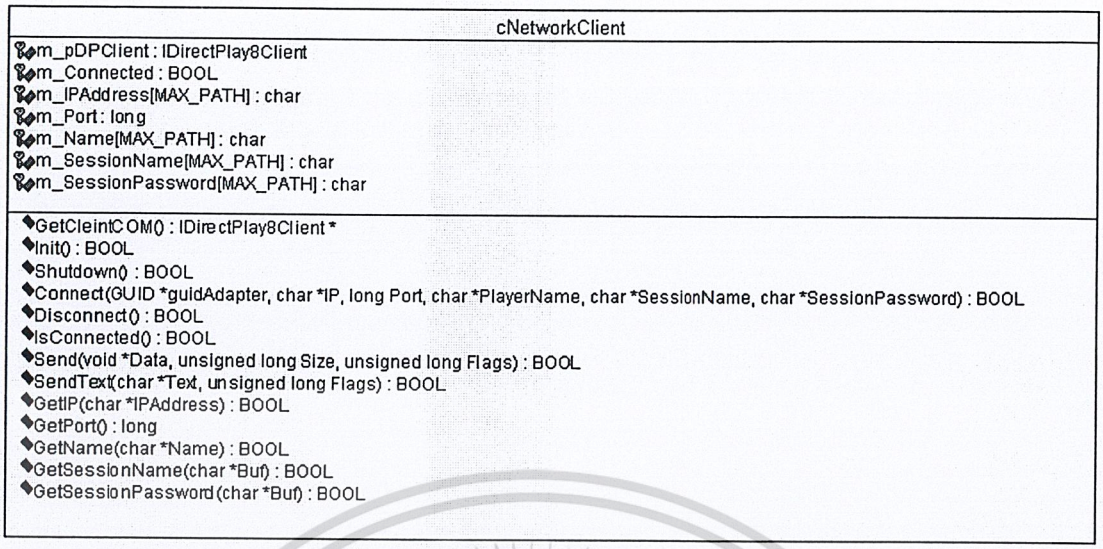
รูปที่ 9-2 คลาสไดอะแกรมของ cNetworkServer

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

คลาสจะทำการเริ่มใช้งานโดยทำการเรียกเมธอด 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
หลั กการออกแบบและการสร้างเกมเอนจินประเภท GUI

10.1 นิยามของเกมเอนจินประเภท GUI

เกมเอนจิน ประเภท GUI (Graphic User Interface) คือ เครื่องมือที่ช่วยในการสร้างเกม ที่ส่วนที่ติดต่อกับผู้ใช้มีลักษณะ เป็น Graphic โดยที่ผู้ใช้ไม่จำเป็นต้อง มีความรู้ความชำนาญด้านการเขียนโปรแกรม โดยผู้ใช้ได้ทำการกำหนดประเภทของเกม ที่จะสร้าง เนื้อเรื่อง และจากนั้นจึงเลือกประเภทของเกมส์เอนจิน ประเภท GUI ให้ตรงกับประเภทของเกมส์ที่ผู้ใช้ต้องการจะสร้าง เพียงเท่านั้น

10.2 นิยามของเกมประเภท RPG

เกมประเภท RPG (Role Playing Game) เป็นเกมส์ ประเภทที่เราต้องสวมบทบาทเพื่อดำเนินตามเนื้อเรื่อง ที่เราได้วางเอาไว้ โดยจะมีการพูดคุยกับ NPC (Non Player Control) เพื่อหาข่าวสารที่เป็นประโยชน์ในการดำเนินตามเนื้อเรื่องที่เราได้วางเอาไว้ มีการต่อสู้ กับ NPC ประเภท สัตว์ประหลาด (Monster) และ ตัวละครที่เราสวมบทบาทจะมีการพัฒนา ตัวเองอยู่ตลอด เวลา จากการที่ได้สู้กับ สัตว์ประหลาด และจะมีการแก้ปริศนาต่างๆจากคำบอกเล่าของ NPC ในเกมส์ ซึ่งจะทำให้ตัวเรารู้สึกสนุกสนาน และสมจริงสมจังกับการได้เล่นเกมส์แนวนี้

10.3 แนวคิดของการออกแบบเกมเอนจิน ประเภท GUI

เนื่องจากเกมเอนจินประเภท GUI นี้มีขอบเขตในการสร้าง จำเพาะแนวเกมใดแนวเกมนึง เช่น หากต้องการสร้างเกมเอนจินประเภท GUI ที่ใช้สร้างเกมแนวไล่ยิงกันแบบมุมมองบุคคลที่หนึ่ง เกมเอนจินที่เราได้ออกมาจะไม่สามารถนำไปสร้างเป็นเกมแนวอื่นๆได้นอกจากแนวเกมที่เราได้กำหนดไว้แต่แรก จึงควรคิดหาแนวเกมที่เหมาะสมในการที่จะนำมาทำเกมเอนจินประเภท GUI

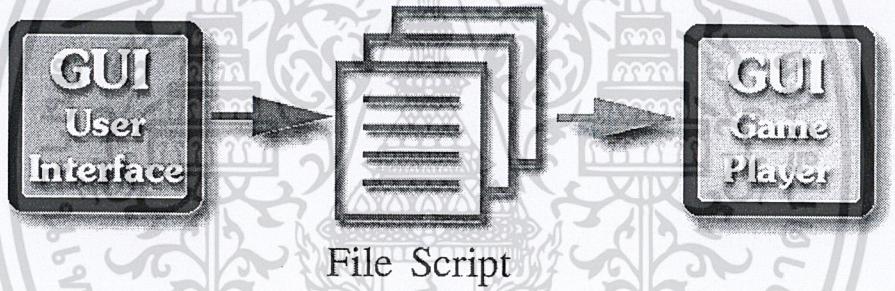
เกมเอนจินประเภท GUI ที่เรากำลังจะสร้าง จะสร้างขึ้นเป็นเกมเอนจินประเภท GUI เพื่อใช้ในการสร้างเกมประเภทเกม RPG เนื่องจากเกมแนวนี้กำลังเป็นที่นิยมในปัจจุบันไม่ว่าไทยหรือต่างประเทศ และความสนุกของเกมนี้คือการที่เราได้สวมบทบาทเข้าไปเป็น ตัวเอกของเกมเพื่อดำเนินเรื่องราว และเกมประเภทนี้ยังมีความสนุกอยู่ที่เรื่องราวที่เกิดขึ้น จึงคิดว่าน่าจะทำเกมเอนจินประเภท GUI เพื่อที่ใช้พัฒนาเกมแนวนี้โดยเฉพาะ

เกมเอนจินประเภท GUI ที่ได้ออกแบบ จะแบ่งออกเป็นสองส่วนใหญ่ๆ คือ

1.ส่วนที่เป็นหน้าต่างติดต่อกับผู้ใช้ในการสร้างเกม(GUI User Interface)โดยส่วนนี้จะเป็นส่วนที่ผู้ใช้ตัว GUI ทำการ สร้างฉากของเกม ตามโครงเรื่องที่ได้วางเอาไว้โดยจะสามารถแก้ไขค่าต่างๆ ภายในเกมส์ ได้ที่เกมเอนจินส่วนนี้เท่านั้น

2.ส่วนที่ใช้เล่นเกม(GUI Game Player) โดยส่วนนี้จะเป็นส่วนที่ผู้ใช้นำเกมส์ที่ได้สร้างมาแล้ว มาเล่นได้โดยเล่นผ่านเกมเอนจินส่วนนี้เท่านั้น

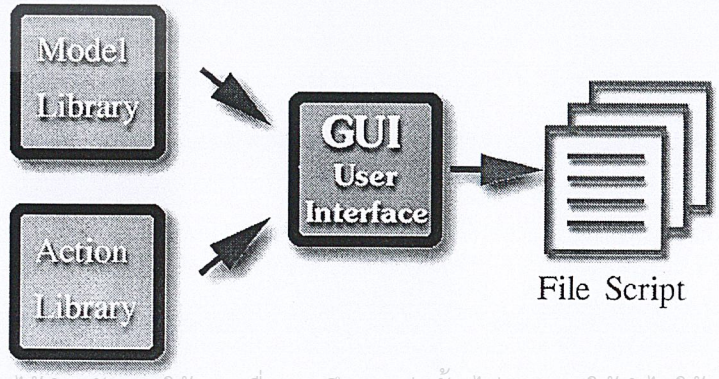
จะเห็นได้ว่า ทั้งสองส่วนแยกออกจากกัน แต่ว่าจะมีตัวกลางที่ทำให้เกมเอนจินทั้งสองส่วนสามารถติดต่อกันได้ ส่วนนี้คือ File Script โดยในส่วนของ File Script นี้จะมีหน้าที่เก็บข้อมูลที่ใช้ในการสร้างเกมตามที่ผู้ใช้ได้กำหนดขึ้นมา โดยเมื่อผู้ใช้ทำการสร้างเกมผ่าน GUI User Interface แล้วจึงจะได้ File Script นี้ขึ้นมา และเมื่อผู้ใช้ได้ทำการเล่นเกมที่ได้สร้างขึ้นมาผ่าน GUI Game Player โดยตัว GUI Game Player จะทำการเรียกข้อมูลที่ผู้ใช้ได้สร้างขึ้นมาจาก File Script เข้ามาเพื่อประมวลแสดงออกมาในรูปของเกมส์ ลักษณะการทำงานของเกมเอนจินโดยรวมดังที่ได้กล่าวมาแล้วข้างต้นจะสามารถ แสดงได้ดังรูป 10-1



รูปที่ 10-1 แสดงการทำงานโดยรวมของเกมเอนจินประเภท GUI

10.3.1 เกมเอนจินประเภท GUI ในส่วนที่เป็นหน้าต่างติดต่อผู้เล่น

ส่วนนี้จะมีหน้าที่ รับ Input จาก Library ที่ได้กำหนดไว้ในตัวของเกมเอนจิน เพื่อมาสร้างให้เป็นเกมประเภท RPG ตามเนื้อเรื่องที่ได้กำหนดไว้ โดยจะสามารถแสดงหลักการทำงานได้ดัง รูปที่ 10-2



รูปที่ 10-2 แสดงการทำงานของเกมเอนจินประเภท GUI ส่วน User Interface

การทำงานคือจะรับ Input มาจาก Library สองส่วน คือ ส่วนที่เป็น Model Library

และ Action Library โดยจะผ่าน ตัว GUI User Interface แล้วจึงสร้างออกมาเป็น File Script โดย

10.3.1.1 Model Library

เป็น library ที่เก็บ พวกโมเดลต่างๆที่มีอยู่ในเกมส์ โดยจะแบ่งออกเป็น 4 ประเภท คือ

- Library ที่รวบรวมแผนที่ภายในเกมส์ (Map Library) จะเก็บโมเดลในส่วนที่เป็นที่เป็นแผนที่ทั้งหมดที่จะสามารถนำมาใช้งาน
- Library ที่รวมรวมตัวละครต่างๆภายในเกมส์ (character Library) ในส่วนนี้จะเก็บโมเดลในส่วนที่เป็น ตัวละครต่างๆเพื่อให้ผู้ใช้นำไปใช้งาน
- Library ที่รวบรวมสิ่งก่อสร้างต่างๆ (Environment Library) ในส่วนนี้จะเก็บโมเดลในส่วนที่เป็นสิ่งก่อสร้าง เช่น บ้าน ต้นไม้ เก้าอี้ เตียง เป็นต้น เพื่อให้ผู้ใช้สามารถนำไปใช้ประกอบฉากตกแต่งภายในเกม
- Library ที่รวบรวมสิ่งของต่างๆ (Item Library) จะรวบรวมโมเดลที่เป็น สิ่งของต่างๆภายในเกมส์ เช่น อาวุธ ยาเพิ่มพลัง ในส่วนนี้แตกต่างจากสิ่งก่อสร้างตรงที่ สิ่งของต่างๆภายในเกมส์นี้จะสามารถ เก็บได้ และนำมาตรวจสอบเงื่อนไขภายในเกมได้ เช่น หากตัวละครของเราไม่ได้เก็บ ขวานมา ก็จะไม่สามารถผ่านเข้าเมืองได้ เป็นต้น โดยจะต่างจาก สิ่งก่อสร้างตรงที่ไม่ได้ประดับฉากเพียงอย่างเดียว

10.3.1.2 Action Library

เป็น library ที่รวบรวมคำสั่งที่เป็น action ที่จะต้องใช้ภายในเกมส์เอาไว้เพื่อให้ผู้ใช้ สามารถเลือกมาใส่ภายในเกม เช่น action ที่ใช้ในการพูด, action ที่ใช้ในการให้สิ่งของ, action ที่ใช้ในการเพิ่มพลัง เป็นต้น

10.3.1.3 GUI User Interface

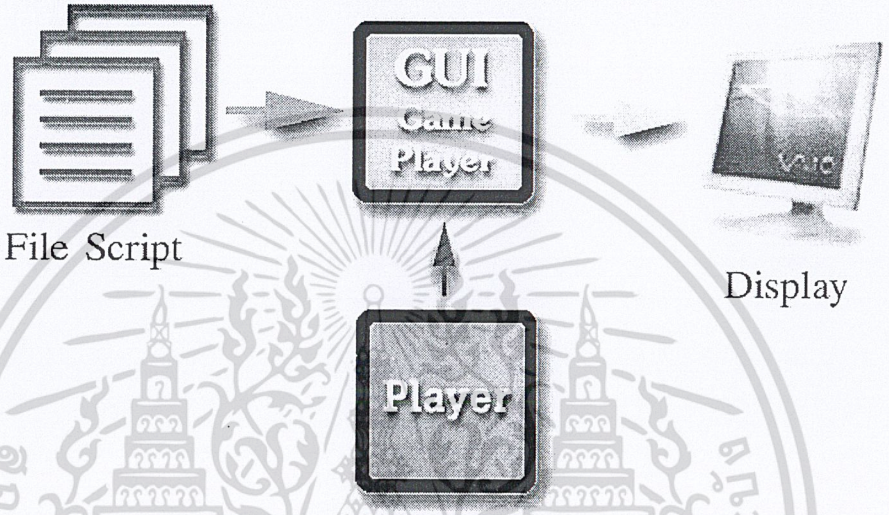
ส่วนนี้จะเป็นส่วนที่ประมวลผลจาก input ที่ผู้ใช้ได้ทำการเลือกมาจาก library มาจัดเก็บเอาไว้ตามโครงสร้างของแต่ละส่วนซึ่งจะแสดงที่ Class diagram และเมื่อผู้ใช้ได้ทำการจัดเก็บ(save) ส่วนนี้จะทำการเขียนข้อมูลลงใน File Script

10.3.1.4 File Script

ส่วนนี้เป็นส่วนที่ใช้เก็บข้อมูลที่ใช้ในการควบคุมสิ่งต่างๆภายในเกมส์ในส่วนที่ผู้ใช้ได้ทำการกำหนดไว้โดย เมื่อผู้ใช้ได้ทำการจัดเก็บข้อมูลที่ได้ทำการสร้างจาก GUI User Interface แล้วตัว GUI User Interface ก็จะทำการสร้าง File Script เหล่านี้ออกมา

10.3.2 เกมเอนจินประเภท GUI ในส่วนที่ใช้เล่นเกมส์ (GUI Game Player)

ในส่วนนี้จะทำหน้าที่ติดต่อกับ File Script ที่ได้มาจาก GUI User Interface เพื่อนำมาประมวลผลและจัดวางสิ่งต่างๆตามเกม ที่ได้ออกแบบไว้แล้วในเกมเอนจินในส่วนแรกแล้วในส่วนนี้จะเป็นส่วนที่ตอบสนอง input จากผู้ใช้ ซึ่งจะได้ออกแบบให้ใช้ mouse เป็นตัวควบคุมภายในเกมเสียเป็นส่วนใหญ่ แล้วจึงนำมาแสดงผลโดยการทำงานดังที่กล่าวมานี้จะแสดงได้ดังรูปที่ 10-3



รูปที่ 10-3 แสดงการทำงานของเกมเอนจินประเภท GUI ในส่วนของ GUI Game Player

10.3.3 File Script

ดังที่ได้กล่าวมาแล้วว่าเกมเอนจิน ประเภท GUI ที่ได้ออกแบบมาจะมีส่วนที่ทำหน้าที่เป็นเสมือนตัวกลาง เชื่อมการทำงานระหว่างส่วนที่เป็น หน้าต่างติดต่อกับผู้ใช้ และส่วนที่ใช้เล่นเกม โดยจะเป็นตัวที่จะกำหนดส่วนต่างๆภายในเกม ซึ่ง file script ที่ได้ออกแบบมาใช้ภายในเกมเอนจิน จะมีทั้งหมด 12 ประเภท คือ

1. mfs (map file script) เก็บข้อมูลเกี่ยวกับฉากที่ผู้ใช้ได้ทำการเลือกออกมาใช้ เช่น ได้เลือกใช้ฉากไหน ใช้ model ชื่ออะไร ใช้ texture ชื่ออะไร เป็นต้น และจะ file script แรกที่จะถูกอ่านเข้าไปในเกมเอนจิน ในส่วนที่ใช้เล่นเกม
2. env (environment) เก็บข้อมูลของสิ่งก่อสร้างต่างๆ ภายในฉาก ซึ่งจะระบุตำแหน่ง ระบุ ชื่อของ file นามสกุล emod ที่สิ่งก่อสร้างในฉากนั้นๆต้องใช้
3. emod (environment model) จะเก็บข้อมูลในส่วนของ model ของสิ่งก่อสร้างต่างๆที่ใช้ในเกมส์ เช่น ชื่อ model ชื่อ texture
4. cdef (character definition) เก็บข้อมูลของตัวละครที่ ผู้ใช้ได้ทำการเลือกมาใช้ เช่น ค่าพลังชีวิต ค่าประสบการณ์ตัวละคร model ที่ตัวละครใช้

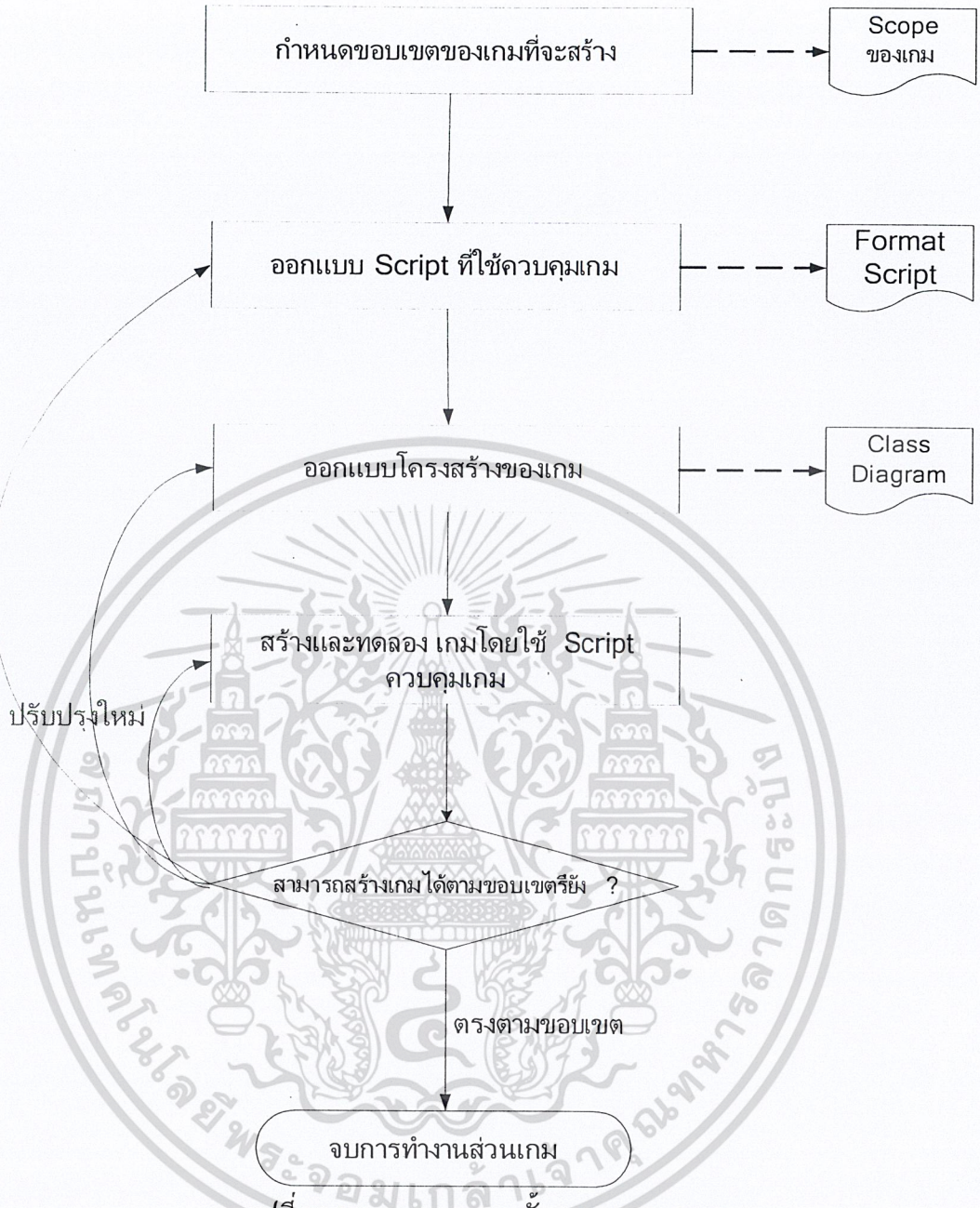
5. ch (character) เก็บข้อมูลในส่วนที่เกี่ยวกับตัวละครภายในฉาก เช่น ฉากนี้มีตัวละครอะไรบ้าง เก็บตำแหน่งของตัวละครภายในฉาก โดยตัวละครจะอ้างถึง definition ของตัวเองซึ่งจะอยู่ใน file นามสกุล cdef
6. cmod (character model) ข้อมูลจะมีลักษณะคล้ายๆกับ emod คือจะเก็บข้อมูลในส่วนที่เป็น model ในส่วนของตัวละคร
7. trp (transport) เก็บข้อมูลของจุดที่ใช้บอกตำแหน่ง เมื่อมีการเปลี่ยนแปลงฉากระหว่างฉาก
8. idef (item model) เก็บข้อมูลของสิ่งของต่างๆภายในเกมส์ โดยข้อมูลที่เก็บจะเป็นคุณสมบัติของสิ่งของนั้นๆ เช่น สิ่งของใช้เพิ่มพลังจะเพิ่มพลังให้ 200 เป็นต้น
9. imod (item model) เก็บข้อมูลในส่วนของ model ของ สิ่งของต่างๆภายในเกม จะเหมือนกับ emod และ cmod คือจะระบุ ชื่อของ model และ texture ที่จะนำมาใช้
10. inv (inventory) file ชนิดนี้จะมีการใช้งานสองประเภท คือ เมื่ออ้างอิงจาก file mfs ก็จะเป็นการอ้างอิงถึงตำแหน่งของสิ่งของในฉากแต่ละฉากเพื่อจะแสดงว่าในฉากนี้มีสิ่งของ สิ่งใดอยู่ที่ตำแหน่งไหนบ้าง และเมื่ออ้างอิงจาก file ch หรือ อ้างอิงจากตัวละครแต่ละตัวก็จะหมายถึงตัวละครนี้มีสิ่งของอะไรอยู่ภายในตัวละครบ้างเปรียบเสมือนเป็นกล่องใส่ไอเท็ม
11. scp (script) จะเป็น file ที่ใช้เก็บคำสั่งที่ใช้ควบคุมการเกิด เหตุการณ์ ต่างๆภายในเกมโดยเหตุการณ์ภายในเกมจะแบ่งส่วนออกเป็นสามส่วน คือ

- เหตุการณ์ที่เกิดขึ้นกับแผนที่ (map script) จะใช้ควบคุมเหตุการณ์ที่เกิดขึ้นเมื่อเข้ามาในแผนที่นี้ เช่นเมื่อมาที่แผนที่นี้แล้ว เปลี่ยนเพลง มีหมอก เป็นต้น
- เหตุการณ์ที่เกิดขึ้นกับวัตถุ (object script) จะใช้ควบคุมเหตุการที่เกิดขึ้นจากวัตถุต่างๆเช่น เมื่อคุยกับตัวละคร ในเกมส์ เมื่อเก็บสิ่งของที่ตกบนพื้น เป็นต้น
- เหตุการที่เกิดขึ้นในบริเวณที่กำหนด (region script) จะใช้ควบคุมเหตุการณ์ที่เกิดขึ้นในบริเวณที่ต้องการให้เกิดเหตุการณ์ต่างๆ

12. rt (route point) เป็นไฟล์ที่ใช้ควบคุมทิศทางการเดินของตัวละครที่ไม่ใช่ตัวผู้เล่น หรือ NPC โดยจะให้ผู้ใช้ได้กำหนดทิศทางและระยะทาง เพื่อให้ตัวละครเหล่านี้เคลื่อนไหว

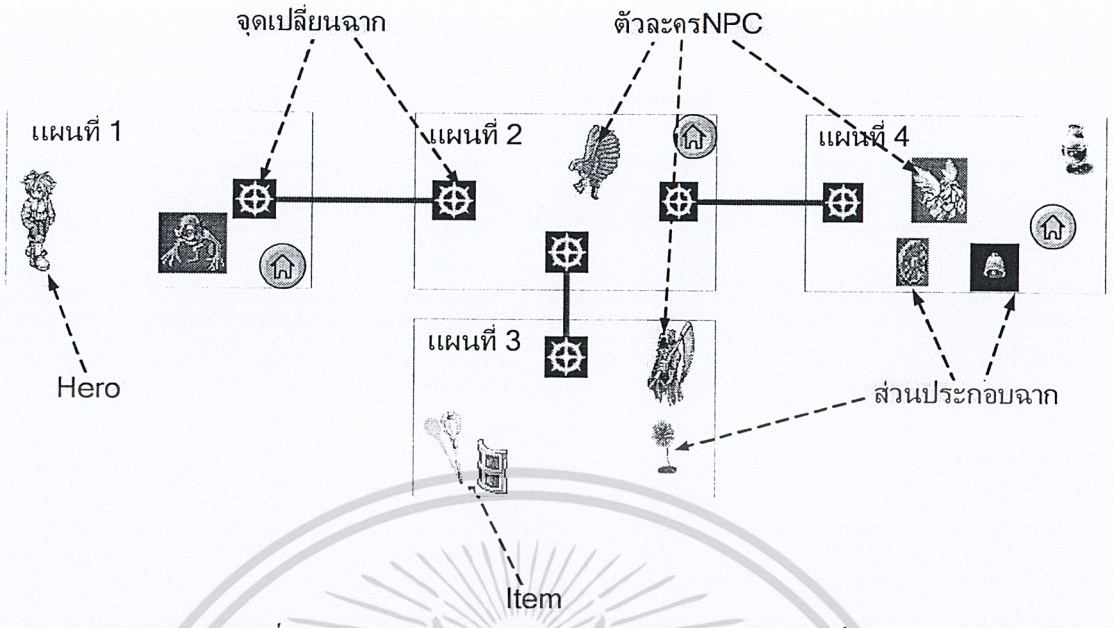
10.4 โครงสร้างและการออกแบบของ GUI Game Engine

GUI Game Engine จะให้ Output คือ File Script ควบคุมเกม เพราะฉะนั้นจึงจำเป็นต้องทำการออกแบบและสร้างเกมที่สามารถควบคุมด้วย File Script ขึ้นมาก่อน ขั้นตอนการทำงานมีดังนี้



รูปที่ 10-4 flow chart แสดงขั้นตอนการออกแบบเกม

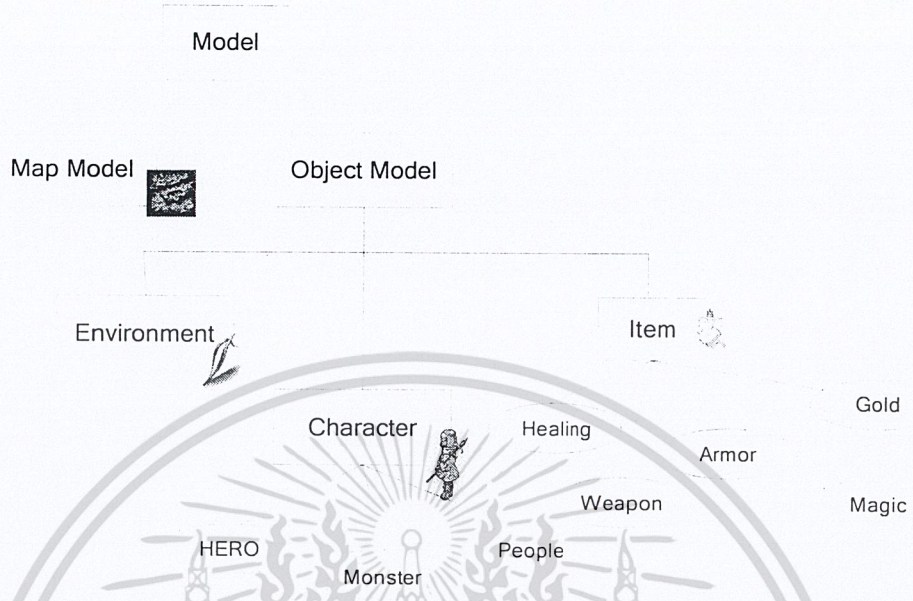
1. ในส่วนแรกคือ กระบวนการกำหนดขอบเขตของเกมเป็นการออกแบบเกมที่เราจะสร้างซึ่งจะได้ผลลัพธ์คือ ขอบเขตของเกมรวมถึงรายละเอียดต่างๆของเกม จากรูปจะอธิบายถึงภาพรวมของเกมก่อน



รูปที่ 10-5 แสดงความสัมพันธ์ระหว่าง ฉากต่างๆและส่วนอื่นๆ

- เกมจะให้ผู้ล่นควบคุมตัวละครได้ หนึ่งตัวเท่านั้น ตัวละครตัวนี้จะเรียกว่า ฮีโร่
- ลักษณะแผนที่ในเกมจะเป็นแผนที่ย่อยๆต่อๆกัน
- จุดเชื่อมต่อระหว่างฉากเรียกว่า ทรานสปอร์ต(Transport)
- สิ่งของที่ตัวละครฮีโร่เก็บได้เรียกว่าไอเท็ม(Item)
- ตัวละครอื่นๆที่ไม่ใช่ฮีโร่ ซึ่งผู้เล่นไม่สามารถควบคุมได้เรียกว่า None Player Control (NPC)
- ส่วนประกอบฉากเช่น พวกต้นไม้ บ้าน ก้อนหิน เรียกว่า environment
- ค่าคุณสมบัติของตัวละครแต่ละตัวสามารถปรับแต่งได้ และสามารกำหนดตำแหน่งและควบคุมการกระทำได้
- ตัวละครแบ่งเป็น 3 ประเภทย่อยด้วยกันคือ ฮีโร่ มอนสเตอร์ ประชาชน ความแตกต่างคือ ฮีโร่สามารถถูกบังคับได้จาก ผู้เล่น มอนสเตอร์จะโจมตีฮีโร่ได้เมื่อฮีโร่เดินเข้ามาในบริเวณของมัน ประชาชนจะไม่ทำอันตรายต่อฮีโร่และจะเป็นผู้ช่วยเหลือหรือให้ข่าวสารต่อฮีโร่
  - ตัวละครพวก NPC สามารถเดินไปมาในฉากเองได้
  - ตัวละครจะมีการระยะเวลาคอยการโจมตี ขึ้นอยู่กับค่าความว่องไวของตัวละครแต่ละตัว เมื่อโจมตีไปครั้งนึงแล้วค่าความว่องไวจะลดเหลือศูนย์ ทำให้ต้องคอยเวลาเพื่อรอให้ค่านี้เต็มอีกครั้งนึง ถึงจะโจมตีใหม่ได้
  - ฮีโร่จะต่อสู้กับสัตว์ประหลาดได้และจะตายได้ถ้า ค่าพลังชีวิตเหลือ 0

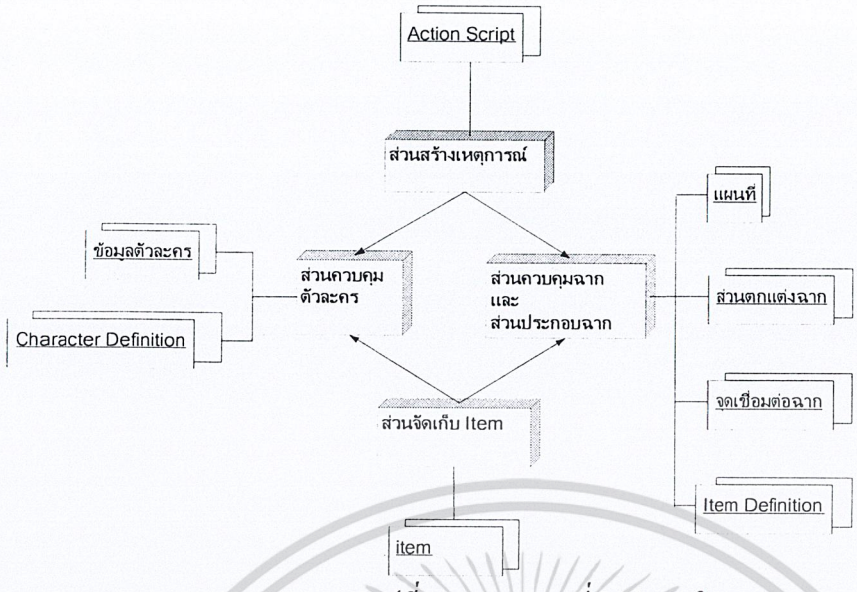
Model ที่ใช้ในเกมมีโครงสร้างดังนี้



รูปที่ 10-6 ประเภท MODEL ในเกม

2. ขั้นตอนต่อไป คือ ออกแบบScript ที่ใช้ควบคุมเกม การสร้างส่วนนี้เราต้องรู้ก่อนว่าเราจะใช้มันควบคุมเกมในส่วนไหนบ้าง จากนั้นก็สร้างรูปแบบScript ตามที่เราต้องการ ในข้อนี้ได้เคยอธิบายแล้วในหัวข้อก่อนหน้านี้

3. ขั้นตอนออกแบบโครงสร้างของเกม ขั้นนี้จะเป็นการออกแบบSoftware โดยนำสิ่งที่ได้จาก 2 ข้อแรกมาเป็นข้อมูลการออกแบบ ซึ่งจะทำให้ได้โครงสร้างที่รับรองการทำงานของ File Script ทั้งหมด Block Diagram แสดง module ต่างๆมีดังนี้



รูปที่ 10-7 Module ที่สำคัญภายในเกม

Module ที่สำคัญภายในเกมมี 4 ส่วนด้วยกันคือ

3.1 ส่วนควบคุมฉากและส่วนประกอบฉาก

ส่วนนี้จะเป็นส่วนที่ทำหน้าที่เก็บข้อมูลของฉากทั้งหมดที่มีอยู่ในเกมทำให้บอกได้ว่าในเกมนี้มีฉากอะไรบ้าง แล้วแต่ละฉากประกอบด้วยวัตถุอะไรบ้าง

โดยมันจะจัดการกับส่วนย่อยๆ 4 ส่วนคือ

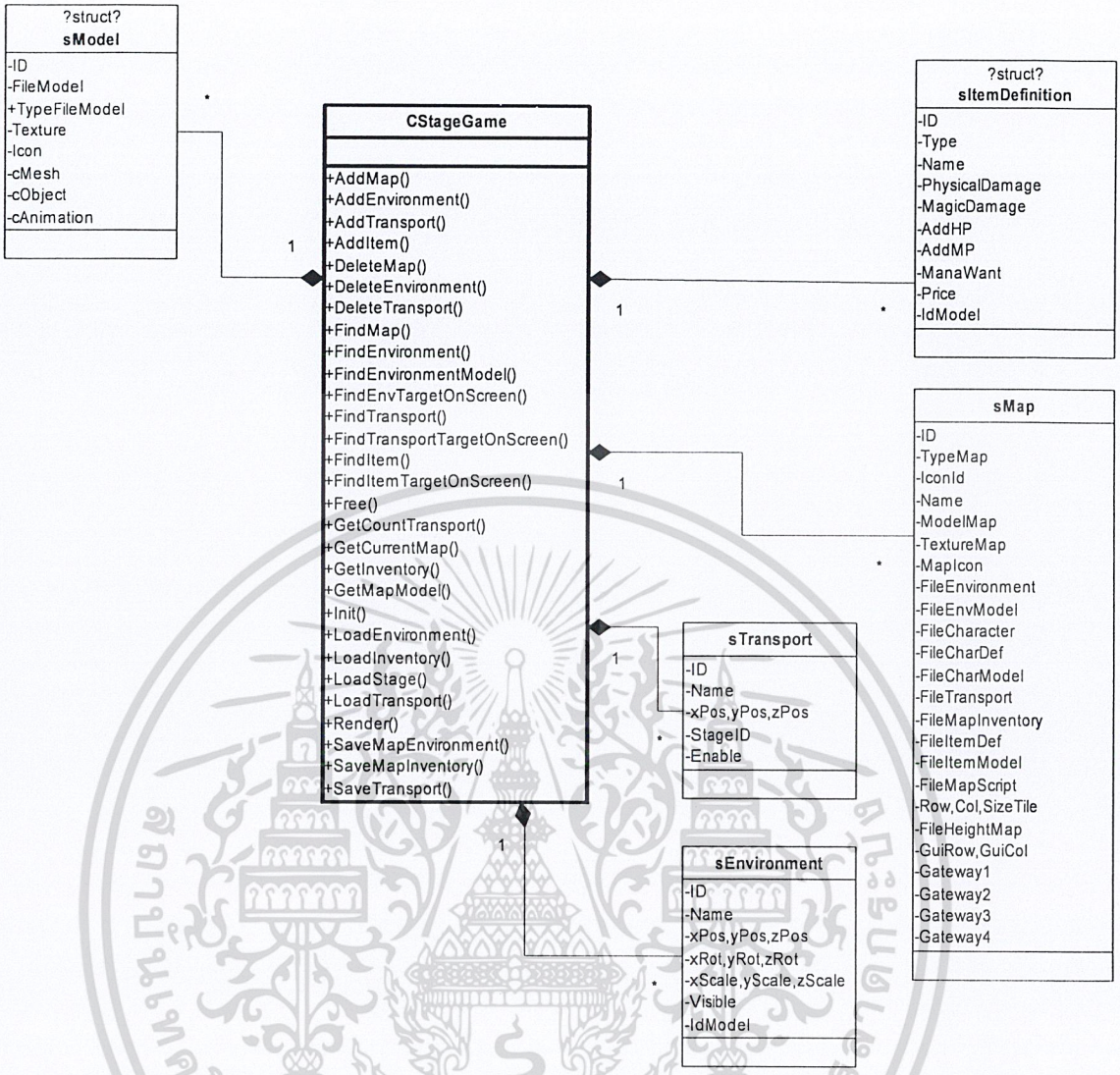
3.1.1 แผนที่ภายในเกม เช่น แผนที่ป่า ทะเลทราย

3.1.2 ส่วนตกแต่งฉาก เช่น บ้าน ต้นไม้ เป็นต้น

3.1.3 จุดเชื่อมต่อฉาก ไว้เก็บความสัมพันธ์ของแต่ละฉาก ว่าเชื่อมโยงกันยังไง และเป็นตัวที่จะบอกว่าเมื่อไหร่จะ load ฉากต่อไป

3.1.4 Item Definition เป็นตัวบอกคุณสมบัติของItem ทั้งหมดที่ใช้ภายในเกม

ในส่วนนี้จะเริ่มทำงานเป็นModule แรกเมื่อเริ่มรันเกม โดยมันจะload ข้อมูลจาก File Script ที่มันเกี่ยวข้อง จากนั้นมันก็จะload ฉากเริ่มต้นรวมถึงส่วนประกอบฉากทั้งหมดขึ้นมา และทำการrender Class และ Struct ของmodule นี้



รูปที่ 10-8 คลาส CStageGame

CStageGame จะทำหน้าที่จัดการload ข้อมูลจาก File Script มาเก็บไว้ในstruct ที่เกี่ยวข้องกับFile Script นามสกุลนั้นๆ ความสัมพันธ์ระหว่าง File Script และ Struct มีดังนี้

- sMap สัมพันธ์กับ .mfs
- sEnvironment สัมพันธ์กับ .env
- sTransport สัมพันธ์กับ .trp
- sItemDefinition สัมพันธ์กับ .idef
- sModel สัมพันธ์กับ .emod

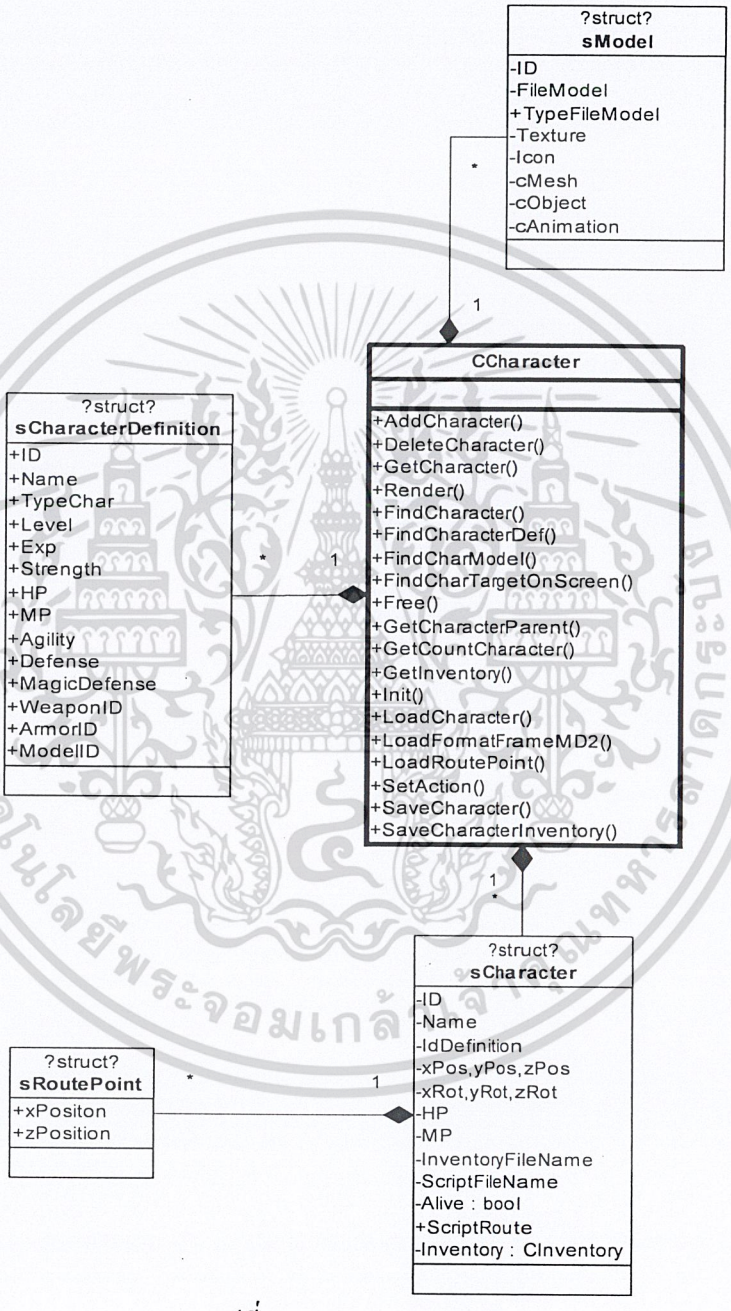
3.2 ส่วนควบคุมตัวละคร

ส่วนนี้จะทำหน้าที่จัดการตัวละครทุกตัวทุกประเภทที่อยู่ในฉากปัจจุบัน ซึ่งมันจะจัดการกับตัวละครที่อยู่ในฉากปัจจุบันเท่านั้นเมื่อมีการเปลี่ยนฉาก ตัวละครฉากเดิมจะถูกsave ลงในไฟล์จากนั้นก็จะ

ทำการload ตัวละครฉากใหม่ขึ้นมาแสดง ในส่วนนี้จะต้องรู้ว่าตัวละครแต่ละตัวกำลังทำอะไรอยู่ที่ไหน และจะต้องทำอะไรต่อไป มีส่วนประกอบย่อย 2 ส่วนคือ

3.2.1 ข้อมูลตัวละคร เป็นตัวบอกว่ามีตัวละครอยู่กี่ตัว อะไรบ้างแต่ละตัว อยู่ที่ตำแหน่งไหน กำลังกระทำอะไรอยู่

3.2.2 Character Definition เป็นตัวบอกถึงคุณสมบัติของตัวละครแต่ละตัว



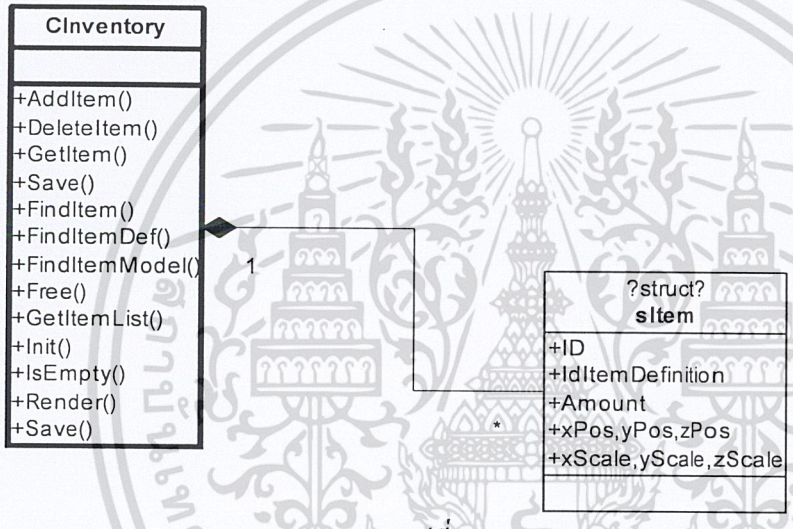
รูปที่ 10-9 คลาส CCharacter

struct ต่างๆจะเป็นที่เก็บข้อมูลที่load มาจาก File Script ซึ่งมีความสัมพันธ์กันดังนี้

- sCharacter สัมพันธ์กับ .ch
- sCharacterDefinition สัมพันธ์กับ .cdef
- sModel สัมพันธ์กับ .cmod
- sRoutePoint สัมพันธ์กับ .rt

3.3 ส่วนจัดเก็บ Item

จะเป็นส่วนที่จัดการกับ Item ทั้งหมดในเกม โดยแยกเป็น Item ของฉากและตัวละคร ในส่วนนี้มีโครงสร้างดังนี้



รูปที่ 10-10 คลาส CInvemtory

struct ต่างๆจะเป็นที่เก็บข้อมูลที่load มาจาก File Script ซึ่งมีความสัมพันธ์กันดังนี้

-sItem สัมพันธ์กับ .item

-sModel สัมพันธ์กับ .imod

3.4 ส่วนสร้างเหตุการณ์

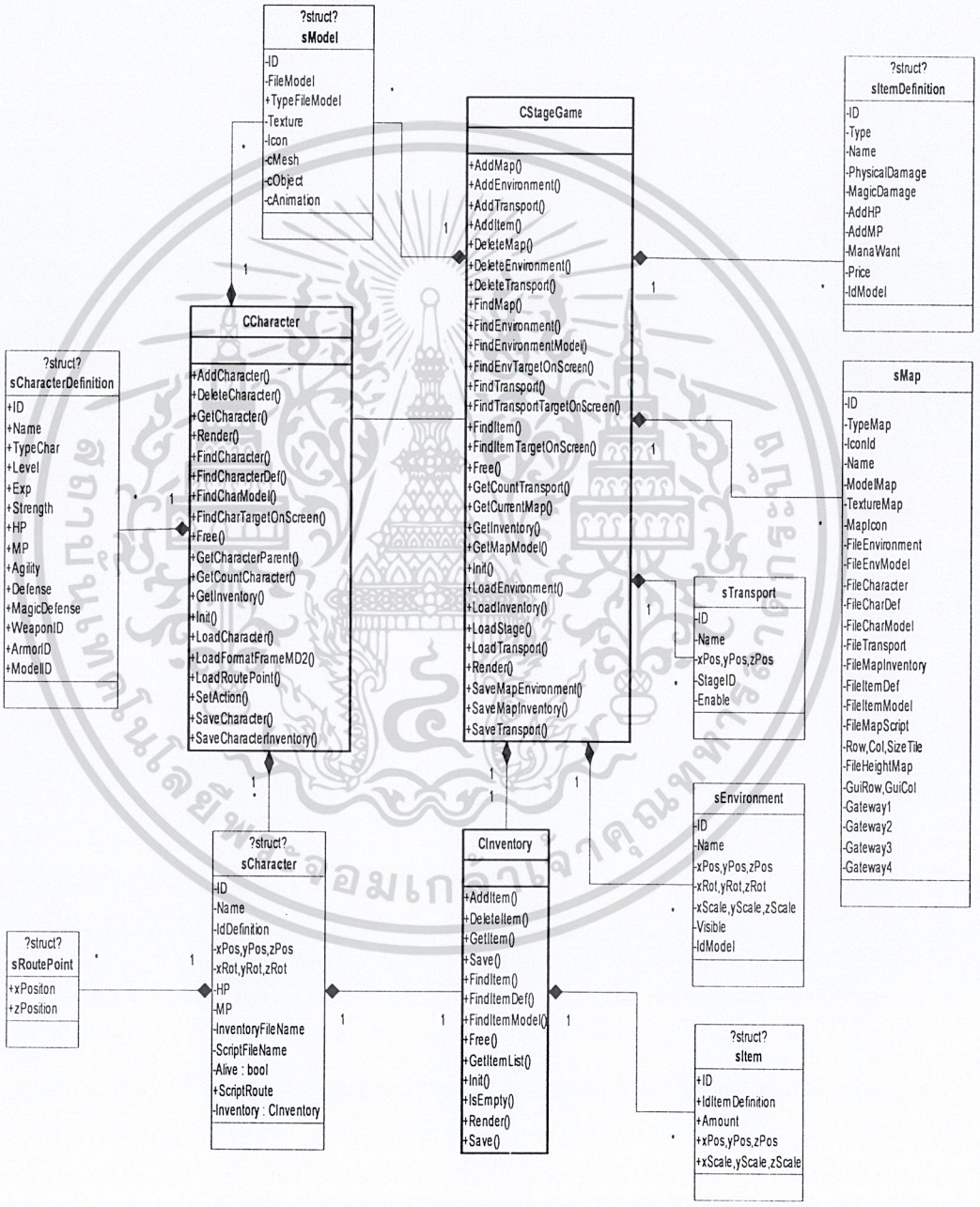
ใน module นี้เป็นการจัดการกับ event ทั้งหมดภายในเกมซึ่งเป็น module ใหญ่ดังนั้นจึงขอยกไปกล่าวในหัวข้อถัดไป

4. สร้างและทดลองเกมโดยใช้ File Script ควบคุมเกม ขั้นตอนนี้เป็นการcoding ตามไดอะแกรมที่ได้ออกแบบมาทั้งหมดเพื่อทดสอบการ ทำงานตามFile Script

5.ขั้นตอนนี้ก็เป็นการแก้ไขส่วนที่ยังทำงานไม่ตรงตามขอบเขตของเกม การกลับไปแก้ไขสามารถกลับไปแก้ไขในขั้นตอนก่อนๆได้

สรุปงานที่ได้จากการทำส่วนนี้คือ เกมที่สามารถใช้File Script ควบคุมได้จากภายนอก และจะได้ รูปแบบของFile Script ทั้งหมดที่ใช้ควบคุมเกมด้วย

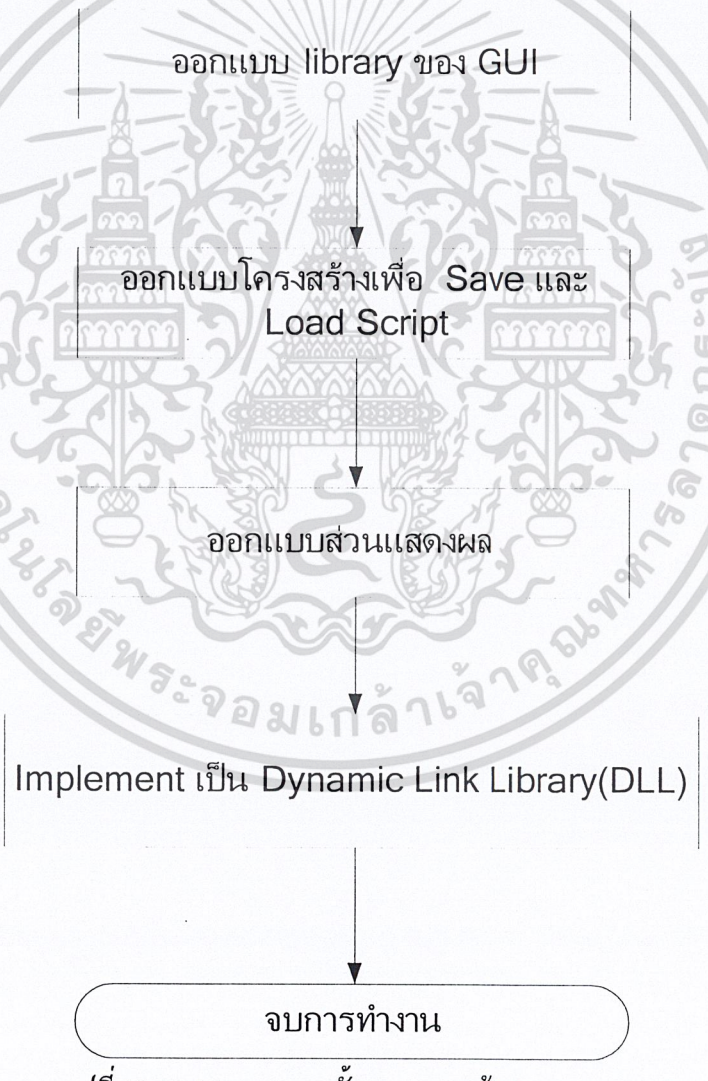
Class Diagram ของทุกmodule เป็นดังนี้



รูปที่ 10-11 คลาสไดอะแกรมสำหรับทุกโมดูล

ขั้นตอนการทำงานที่จะทำต่อไปก็คือ จะเริ่มเข้าสู่การออกแบบส่วนหลักของ GUI Game -Engine ในส่วนนี้จะสร้างมาเพื่อจะนำไปประกอบกับ User Interface อีกทีนึงเสมือนกับเป็นตัวที่ทำงานอยู่เบื้องหลังที่คอยตอบสนองต่อผู้ใช้ตามที่ผู้ใช้ต้องการ เช่น การขอข้อมูลแผนที่จากไลบรารี่ (library) ที่ผู้ใช้ต้องการมาแสดงผล

จุดประสงค์หลักของขั้นตอนนี้ คือ การสร้างส่วนที่สามารถผลิตFile Script ได้ ซึ่งจะมีขั้นตอนดังรูปข้างล่าง

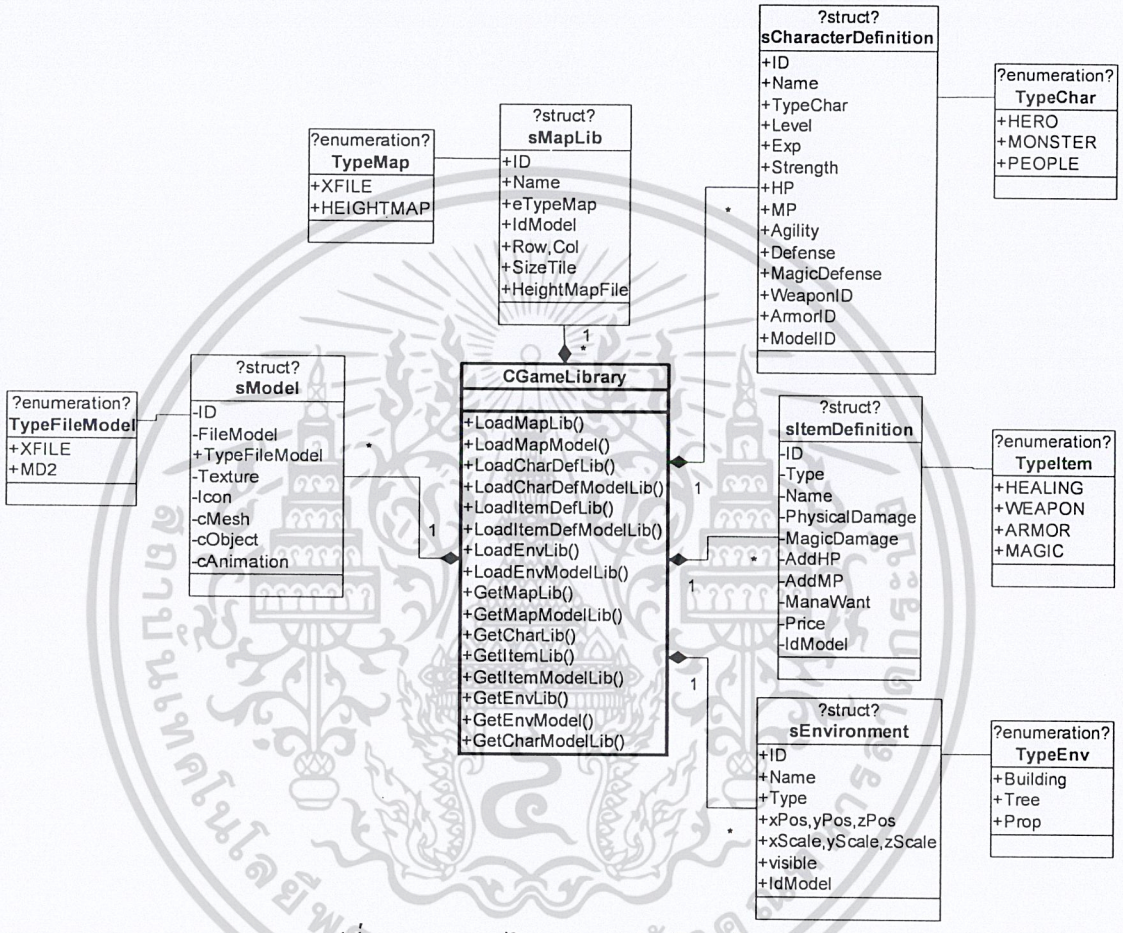


รูปที่ 10-12 Flow Chart ขั้นตอนการสร้างCore ของ GUI

10.4.1 ขั้นตอนการออกแบบส่วน library ของ GUI

เนื่องจากลักษณะของ GUI จำเป็นต้องมี library ไว้เป็นเหมือนห้องเก็บของ ซึ่งในที่นี้จะแบ่งเป็น 2 ห้อง คือ ห้องที่ทำหน้าที่เก็บโมเดล(Model Library) และ ห้องที่เก็บคำสั่งการกระทำ (Action Library) ของที่ผู้ใช้ต้องการ จะถูกดึงมาจากห้องเหล่านี้ แล้วนำมาใส่ไว้ในห้องอีกห้องนึงซึ่งจะกล่าวถึงในขั้นตอนถัดไป

ในส่วนนี้จะอธิบายถึงเรื่องการจัดการเอาของออกจากห้องเก็บของ ซึ่งมีโครงสร้างดังนี้



รูปที่ 10-13 คลาสไดอะแกรมส่วนโหลดไลบรารี

จากไดอะแกรมจะเห็นว่ามี class CGameLibrary ที่บรรจุ struct ต่างๆอยู่ struct เหล่านี้เป็นเหมือนกับชนิดของของ ที่เก็บอยู่ในห้องเก็บของที่ชื่อว่า CGameLibrary ที่มีหน้าที่หลักๆในการค้นหาสิ่งที่ต้องการออกมาให้เมื่อมีคนเรียกใช้ โดยตอนเริ่มต้นมันจะต้องนำของมาเก็บในห้องซะก่อนจึงจะสามารถเปิดให้คนมาเรียกใช้ได้ขั้นตอนนี้เป็นการโหลดข้อมูลจาก File Scriptซึ่งเป็น File Script ที่เก็บข้อมูลทั้งหมดไว้เช่น ข้อมูลของทุกๆโมเดลซึ่งอ้างถึงไฟล์โมเดลทุกๆไฟล์

จากนั้นเมื่อมีการร้องขอข้อมูลจากผู้ใช้เช่น ผู้ใช้อาจต้องการใช้ตัวละครชื่อ เอ คลาสนี้จะทำหน้าที่ค้นหาไปยังส่วนที่จัดเก็บข้อมูลที่เกี่ยวข้องกับ ตัวละครเอนี้ ข้อมูลที่ได้นี้จะถูกนำออกมาจากห้องแล้วส่งไปยังส่วนเก็บข้อมูลในส่วนถัดไปที่จะกล่าวถึง

สรุปเป้าหมายการทำส่วนนี้ก็คือการโหลดข้อมูลจาก File Script มาสร้างเป็น library และให้บริการค้นหาข้อมูลที่ต้องการได้

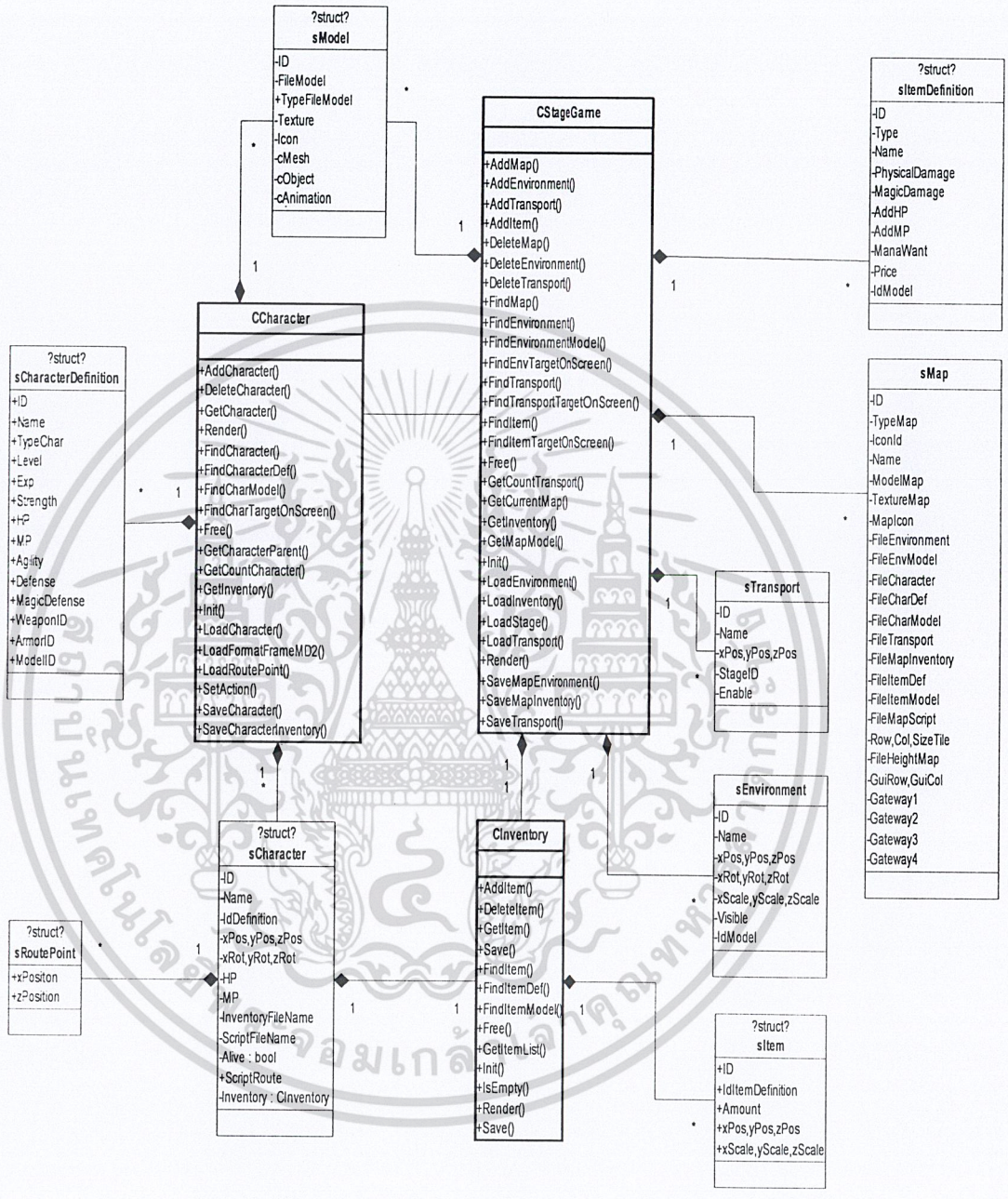
10.4.2 ขั้นตอนการออกแบบโครงสร้างการ Save และ Load Script

ส่วนนี้ถ้าจะให้เปรียบเทียบให้เข้าใจง่ายก็คือ เป็นเหมือนตะกร้าใส่ของ ซึ่งของในที่นี้ก็คือของที่ดึงมาจากห้องเก็บของจากขั้นตอนที่แล้ว เมื่อได้ของมาครบแล้วก็จะทำการเซฟลงไปยังFile Script ส่วนเมื่อต้องการนำของออกมาแก้ไขใหม่ก็ทำการ โหลดข้อมูลขึ้นมาซึ่งเป็นข้อมูลเก่าที่ได้เคยเซฟไว้ต่างกับการโหลดข้อมูลในส่วนของlibraryที่อยู่ในขั้นตอนที่แล้ว โครงสร้างจะแสดงรวมกับส่วนการแสดงผลซึ่งเป็นขั้นตอนถัดไป

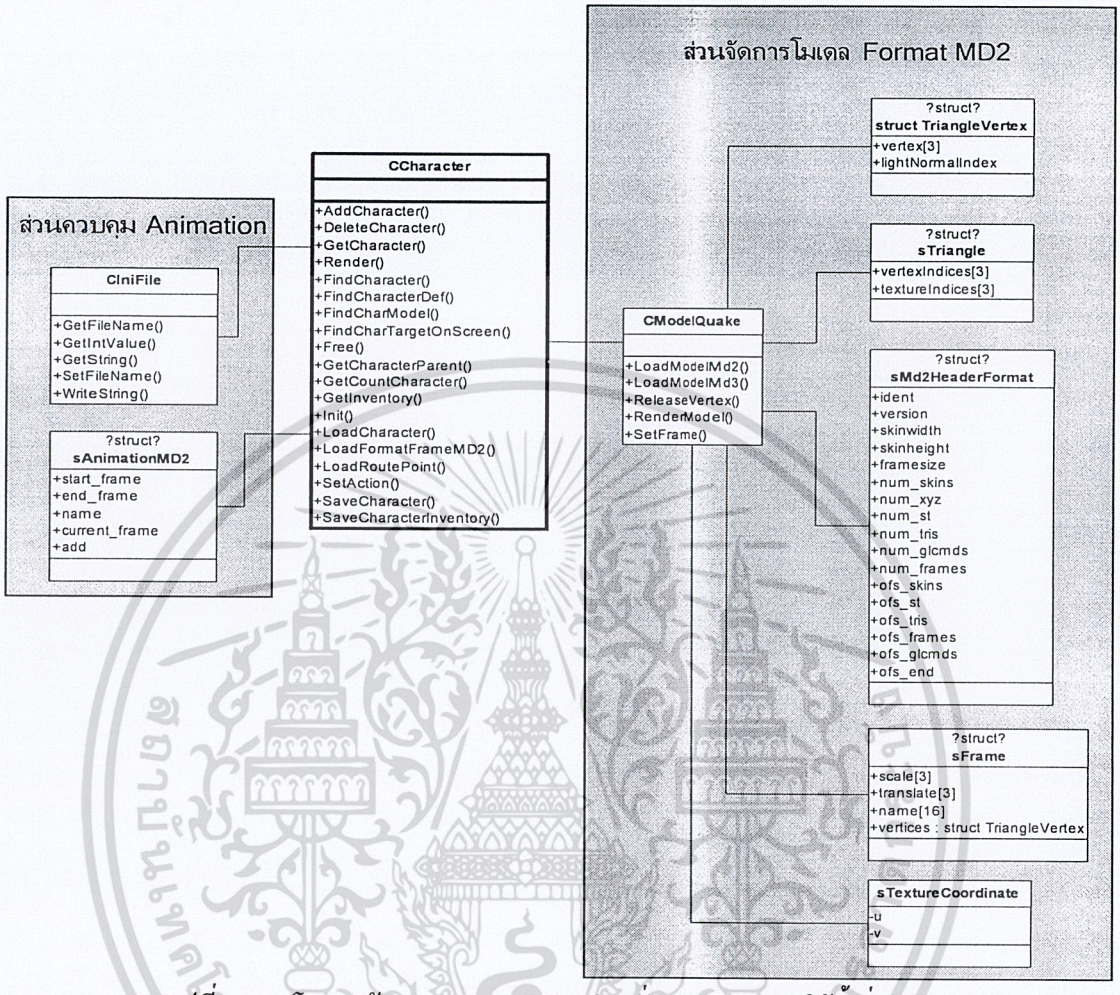
10.4.3 ส่วนแสดงผลในGUI

โมเดลทุกตัวที่ถูกเก็บอยู่ในตะกร้าจะถูกแสดงผล แต่เมื่อผู้ใช้ต้องการเปลี่ยนฉากเพื่อที่จะไปทำการแก้ไขที่ฉากอื่น โมเดลทุกตัวในตะกร้าจะถูกเซฟเก็บไว้ก่อนแล้วนำโมเดลของฉากนั้นๆ มาใส่ในตะกร้าเพื่อจะแสดงผลออกมา

โครงสร้างของส่วนที่ 2 และ 3 นี้จะถูกแสดงรวมกัน ซึ่งโครงสร้างนี้ทางทฤษฎีแล้วจะเหมือนกับโครงสร้างที่เคยได้กล่าวมาแล้วตอนสร้างเกม ซึ่งนำมาใช้ใหม่ในส่วนGUI แต่ในทางปฏิบัติแล้วจะแตกต่างกันเนื่องจากทางฝั่งเกมจะimplement code ด้วยwin32API แต่ทางฝั่งGUI จะใช้เป็นMFC เพราะต้องคำนึงถึงการสร้างuser interfaceด้วย



รูปที่ 10-14 Class Diagram ของขั้นตอนที่ 2 และ 3

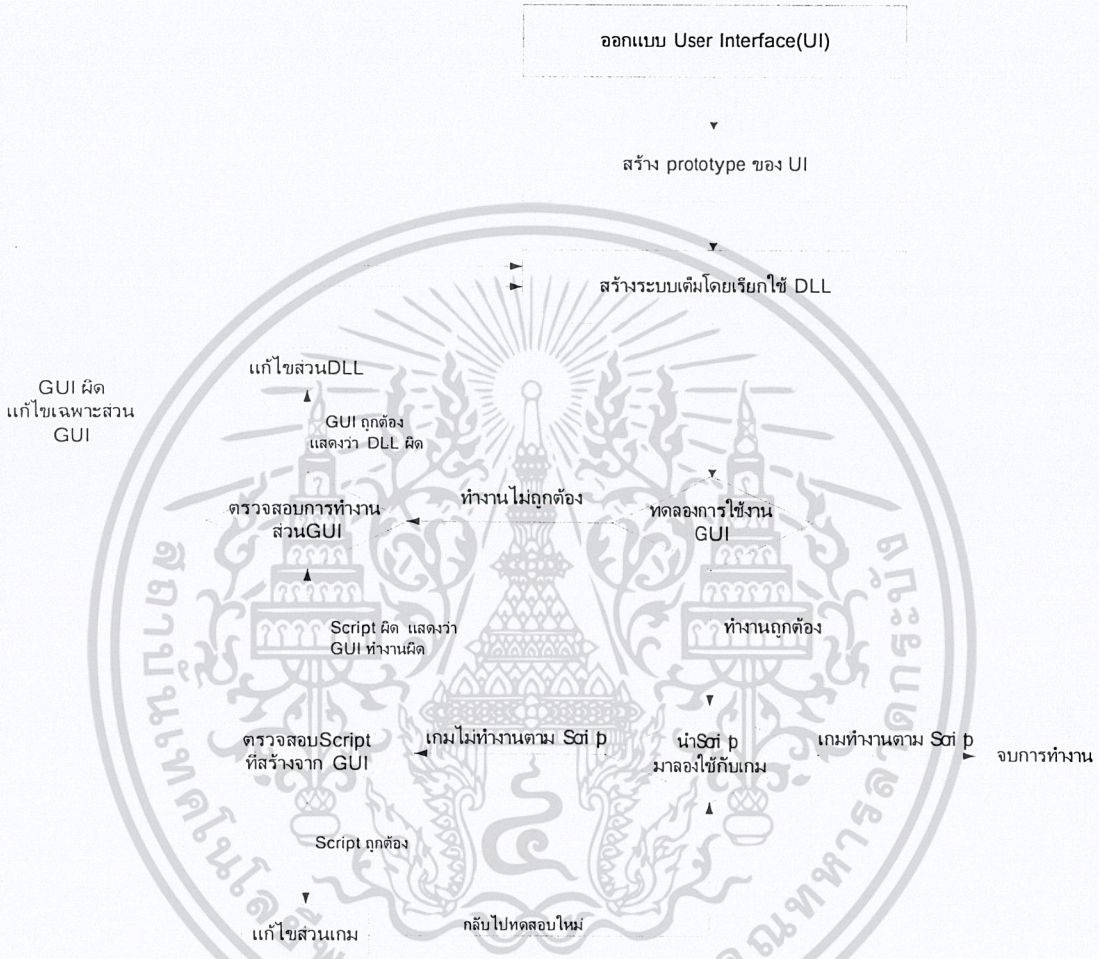


รูปที่ 10-15 โครงสร้างของการควบคุมอนิเมชั่นของตัวละคร(ใช้ทั้งฝั่ง เกม และ GUI)

10.4.4 Implement เป็น Dynamic Link Library (DLL)

จากขั้นตอนทั้งหมดที่กล่าวมาจะได้Dynamic Link Library(DLL) ที่จะนำไปประกอบกับส่วน user interface การที่ตัดสินใจสร้างเป็น DLL นี้เพราะจะต้องนำส่วนนี้ไปให้ทีมพัฒนาอีกกลุ่มเรียกใช้งาน เพื่อเป็นการป้องกันการแก้ไขและเป็นการสะดวกในการอัปเกรดอีกด้วย

การทำงานต่อมาจะเป็นส่วนที่สร้าง GUI Game Engine ซึ่งเป็นขั้นตอนการทำงานขั้นสุดท้ายเพื่อให้ได้ Script ที่ถูกต้องและมีการทำงานตรงตามขอบเขตที่ได้วางไว้ ขั้นตอนการทำงานมีดังนี้



รูปที่ 10-16 Flow Chart ขั้นตอนการสร้าง GUI

1. ขั้นตอนการออกแบบ user interface

ในขั้นตอนนี้จะทำการออกแบบหน้าต่างการใช้งานทั้งหมด โดยจะเริ่มสร้างหน้าต่างหลักๆก่อนจากนั้น ก็จะสร้างพวกไดอะลอก สำหรับแก้ไขค่าพารามิเตอร์ต่างๆ และเน้นว่าผู้ใช้จะต้องใช้งานได้ง่าย โดยไม่ต้องมีความรู้ด้านการเขียนโปรแกรม และต้องแสดงสิ่งที่ผู้ใช้ต้องการได้ถูกต้องตามที่เห็น

2. สร้างหน้าต่างต้นแบบจำลองการทำงานของ user interface

เป็นการสร้างหน้าต่างตามที่ได้ออกแบบไว้ แต่ยังไม่มีการทำงานจริงที่เป็นส่วนติดต่อกับข้อมูล

3. สร้างระบบเต็มโดยการเรียกใช้ DLL

ส่วนนี้จะทำการสร้างการเชื่อมต่อระหว่าง interface และ ตัวDLL ที่เคยได้สร้างไว้แล้วก่อนหน้านี้

4. ขั้นตอนการทดสอบ

การทดสอบจะทดสอบ 2 ส่วนคือ

- ทดสอบการใช้งาน GUI เพื่อดูว่าตัวโปรแกรมทำงานได้ตามที่ได้ออกแบบไว้รึไม่ ในส่วนนี้ถ้าเกิดความผิดพลาดก็จะรู้ได้ว่าผิดพลาดที่ส่วนGUI เอง ซึ่งยังไม่เกี่ยวข้องกับตัวเกม

- ทดลองนำ File Script ที่ได้ไปควบคุมเกมว่าเกมสามารถดำเนินเรื่องได้ถูกต้องตาม ที่ได้กำหนดไว้ใน File Script รึเปล่า ส่วนนี้ถ้าผิดก็อาจจะผิดได้ 2 กรณีคือ File Script ผิด หรืออาจผิดจากตัวเกมที่ประมวลผล script นี้ การทำงานจะสิ้นสุดเมื่อเกมสามารถดำเนินเรื่องราวได้ตามFile Script ที่ได้วางไว้ทั้งหมด

10.5 การสร้างเหตุการณ์ให้กับเกม

เหตุการณ์ที่เกิดขึ้นภายในเกมจะเกิดขึ้นจากการกระทำ(Action) ระหว่างผู้เล่น และ ออบเจ็กต์ต่างๆที่ได้กำหนดไว้ ในGUI Game Engine จะมีส่วนที่ทำให้ผู้ใช้สามารถสร้างเหตุการณ์ (Event) ให้เกิดขึ้นภายในเกมได้ โดยจะมีเซทของการกระทำซึ่งถูกเก็บอยู่ในส่วนที่เรียกว่า “Action Library” ในส่วนนี้จะมี Action Template เก็บไว้ เป็นตัวบอกว่ามีการกระทำอะไรบ้างที่ให้ผู้ใช้สามารถเลือกมาใช้ได้ การกระทำแต่ละตัวจะประกอบด้วย entry ซึ่งเป็นค่าที่ให้ผู้ใช้สามารถกำหนดได้ แบ่งเป็นประเภทของ entry ได้ดังนี้

1. No entry
2. Text entry
3. Boolean value
4. Integer number
5. Float number
6. Multiple Choice

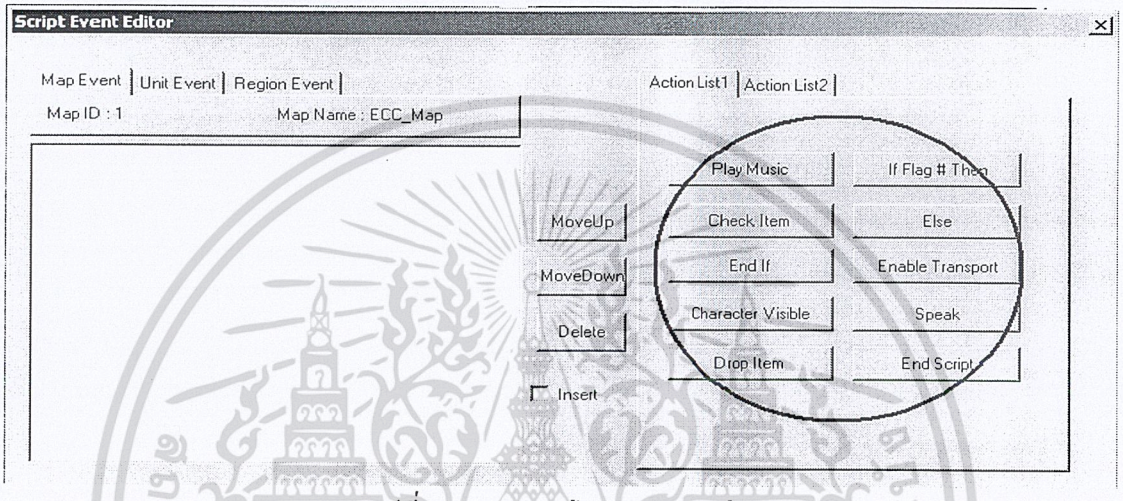
แต่ละประเภท entry จะมีลักษณะแตกต่างกันเช่น Text entry สามารถที่จะจำกัดจำนวนตัวอักษรได้ ถ้าเป็นInteger number สามารถกำหนดขอบเขตได้

ตัวอย่างของการกระทำที่เก็บอยู่ภายใน Action Template เป็นดังนี้

```
“Increase health point amount ~ to character ~”
INT 1 255
INT 1 255
```

บรรทัดแรกนั้นจะเป็นประโยคบอกถึงการกระทำว่าให้ทำอะไร โดยจะมีเครื่องหมาย ~ แทน entry ที่จะให้ผู้ใช้กรอก โดยประเภทข้อมูลที่จะใส่จะบอกที่บรรทัดถัดมา ในที่นี้มี 2 entry แต่ละ entry เป็นประเภท integer มีค่าตั้งแต่ 1 ถึง 255

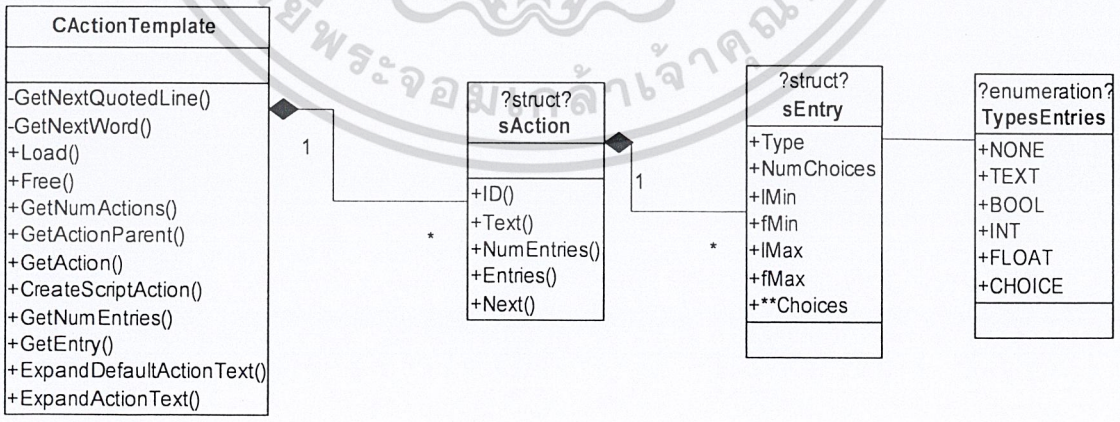
Action Template จะถูกสร้างโดยผู้พัฒนาและถูกเก็บไว้ในไฟล์นามสกุล .mla เมื่อ เปิดโปรแกรม GUI ขึ้นมาAction Template จะถูกload เข้ามาเก็บไว้ โดยGUI จะมี dialog ที่ชื่อว่าScript Event Editor เป็นdialogไว้จัดการกับเหตุการณ์ทั้งหมดภายในเกม แต่ละการกระทำภายใน Action Template จะถูกแทนด้วยปุ่มให้user เลือกดังรูป



รูปที่ 10-17 แสดง ปุ่มของ actionต่างๆ

ส่วนที่วงกลมด้วยสีแดงคือ Action ต่างๆที่นำมาจาก Action Template (ในส่วนการใช้งานdialog นั้นจะไม่อธิบายในวิทยานิพนธ์ นี้ ในที่นี้จะเน้นส่วนโครงสร้างการทำงานเป็นหลัก)

Class Diagram ในส่วนของAction Template มีดังนี้

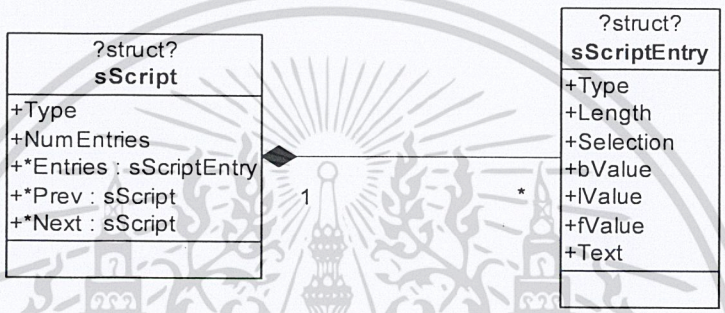


รูปที่ 10-18 โครงสร้างของ Action Template

อธิบาย Class Diagram

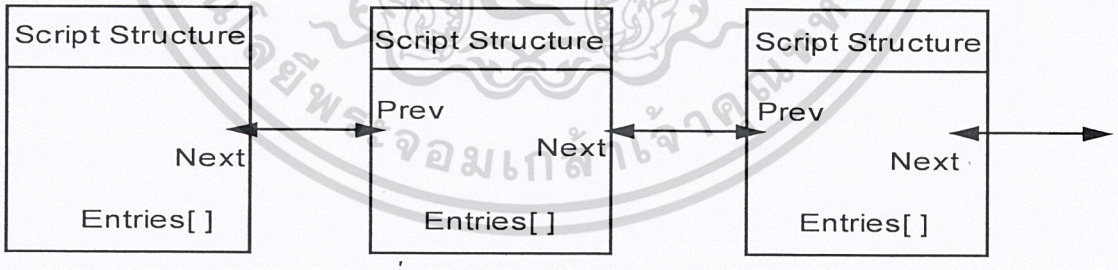
CActionTemplate ทำหน้าที่จัดการติดต่อกับไฟล์นามสกุล .mla เพื่อโหลดaction มาเก็บไว้ในsAction ซึ่งมีโครงสร้างเป็น link list ที่ชี้ไปยังaction ถัดไป ในแต่ละaction ก็จะมี entry เก็บไว้ซึ่งชนิดของ entryจะดูได้จาก Type

เนื่องจากว่าsEntry จะเก็บเฉพาะtemplate ของ actions และ entries ดังนั้นการเก็บข้อมูลของแต่ละaction ตามที่ผู้ใช้ได้แก้ไขนั้นจำเป็นต้องมีโครงสร้างอื่นมาเพิ่มอีก ดังนี้



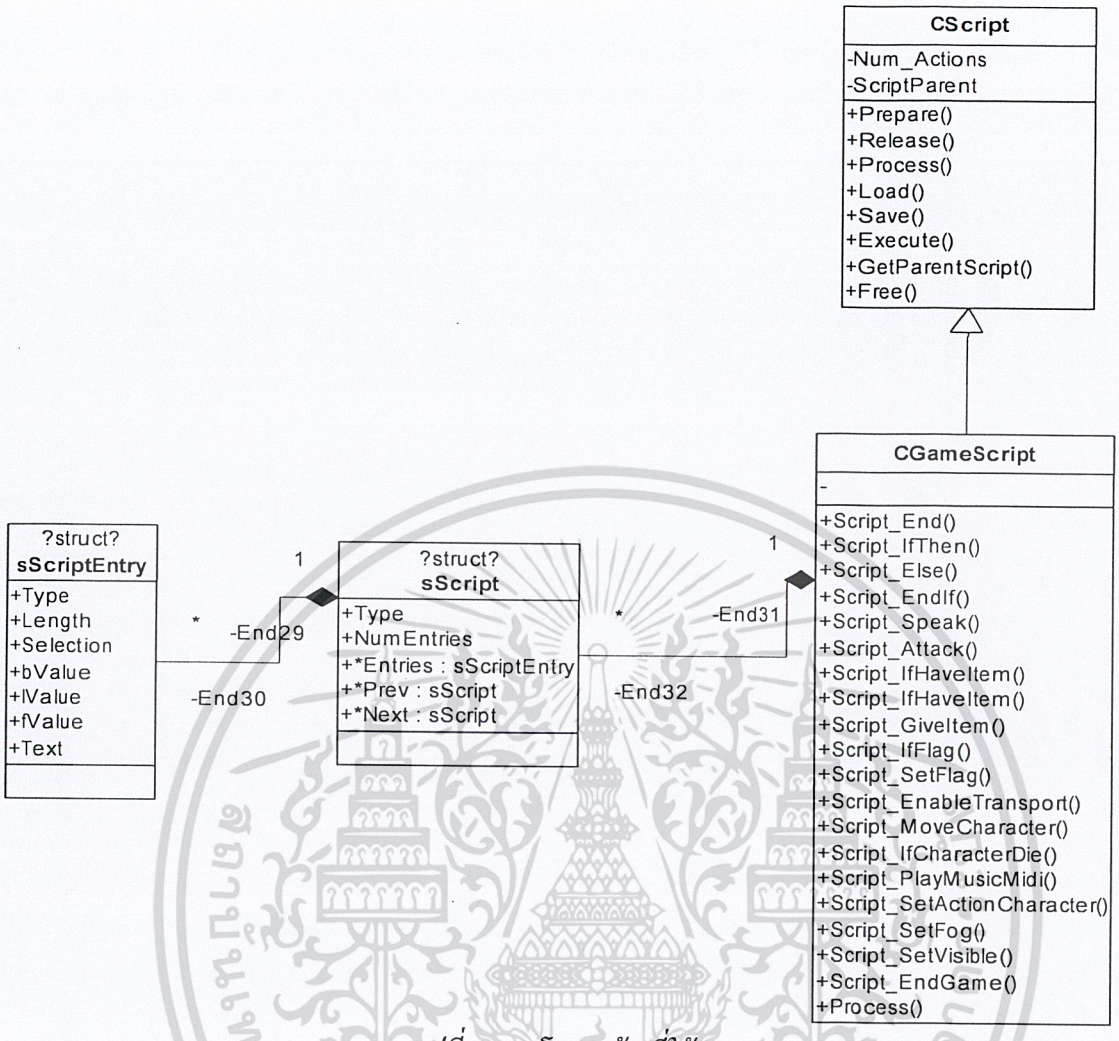
รูปที่ 10-19 Struct Script

sScriptEntry จะคล้ายกับ sEntry ต่างกันที่มันจะเก็บ ค่าจริงๆตามที่ผู้ใช้ได้กรอกเข้ามา sScript จะเป็นตัวเก็บแต่ละ action เอาไว้ซึ่งจะเชื่อมโยงไปยัง action ต่อๆไปแต่ละ script นี้จะมี array ของscript entriesของตัวมันเองเก็บไว้อยู่ ซึ่งมีโครงสร้างดังรูป



รูปที่ 10-20 โครงสร้างของ Action Script

Event Script ที่ได้จาก GUIนี้จะถูก save เป็นไฟล์นามสกุล .scp
เมื่อเราได้ไฟล์ที่เก็บข้อมูลของ event ทั้งหมดแล้วเราก็จะเรียกใช้งานมันโดยใช้ class เหล่านี้



รูปที่ 10-21 โครงสร้างที่ใช้ Process Script

การเรียกใช้งานevent script จะเรียกผ่านmethod ของCGameScript ที่ชื่อว่า Execute ซึ่งรับ parameter 1 ตัว คือชื่อ ไฟล์ event script ที่มีนามสกุลscp จากนั้นCGameScriptจะทำการloadข้อมูลของ event ที่เก็บไว้ในไฟล์เข้ามา ใส่ไว้ในโครงสร้างของ sScript และ sScriptEntry เหมือนตอนก่อนที่มันจะถูก save จากนั้นมันก็จะทำการ process แต่ละ script ไล่จากต้น list ไปยัง ท้ายlist โดยเลือกฟังก์ชันต่างๆตามประเภทของaction ขึ้นมาprocess

บทที่ 11
การพัฒนาเกมเพื่อทดสอบ GUI Game Engine

ประเภทเกมที่พัฒนา

สำหรับเกมที่พัฒนาขึ้นนี้เป็น เกม RPG ที่ผู้เล่นจะต้องเล่นตามเงื่อนไขของเกมที่ได้สร้างเป็นเรื่องราวไว้ ตั้งแต่ต้นจนจบ ซึ่งตัวอย่างเกมแนวนี้เช่น Diablo, Dungeon Siege โดยเกมที่จะสร้างมีลักษณะดังนี้

- เกมจะต้องมีเรื่องราวและเป้าหมาย ให้ผู้เล่นสามารถดำเนินเหตุการณ์ตามเรื่องราวของเกมได้
- ตัวละครของผู้เล่นสามารถพัฒนาระดับความสามารถได้
- 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

11.1 สร้างโครงเรื่องและรายละเอียดของเกม

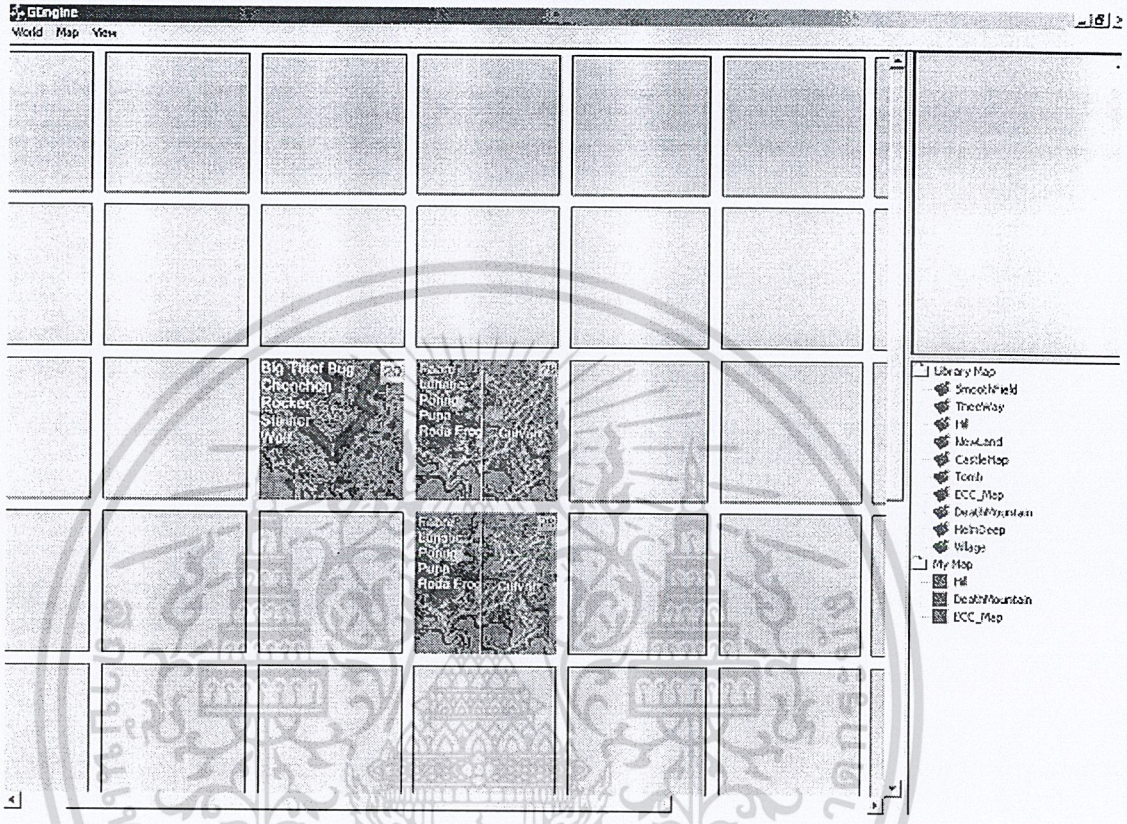
เรื่องราวของเกมที่จะพัฒนามีอยู่ว่า คุณจะสวมบทเป็นนักบินอวกาศที่ต้องเดินทางไปยังสถานีอวกาศที่ดาวเคราะห์ดวงหนึ่งแต่เมื่อเดินทางเกือบจะถึงยังสถานี ยานอวกาศเกิดเครื่องยนต์ขัดข้องและตกลงก่อนที่จะถึงสถานี คุณเป็นผู้รอดชีวิตเพียงคนเดียวและไม่สามาถติดต่อกับสถานีได้จึงต้องหาทางไปด้วยตนเอง

เป้าหมายคือ คุณจะต้องกลับไปยังสถานีให้ได้ โดยระหว่างทางคุณจะต้องเจอกับสัตว์ต่างดาวที่จะมาทำร้ายคุณ เกมจะจบเมื่อคนที่สถานีรู้ว่าคุณมีชีวิตอยู่

แผนที่ทั้งหมด มีด้วยกัน 3 แผนที่ คือ แผนที่ที่ยานตก แผนที่ก่อนถึงสถานี แผนที่ที่สถานี

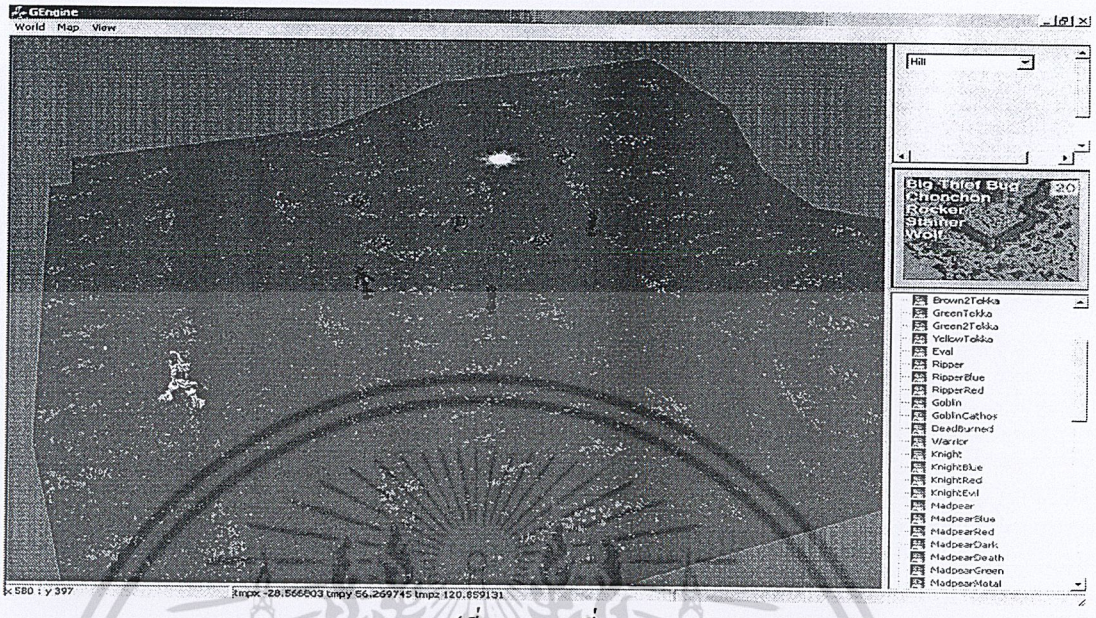
11.2 เริ่มการสร้างเกมด้วย GUI Game Engine

11.2.1 เริ่มต้นด้วยการเลือกและวางแผนที่ ขั้นตอนนี้จะเป็นการวาง ตำแหน่งแผนที่โดยรวม



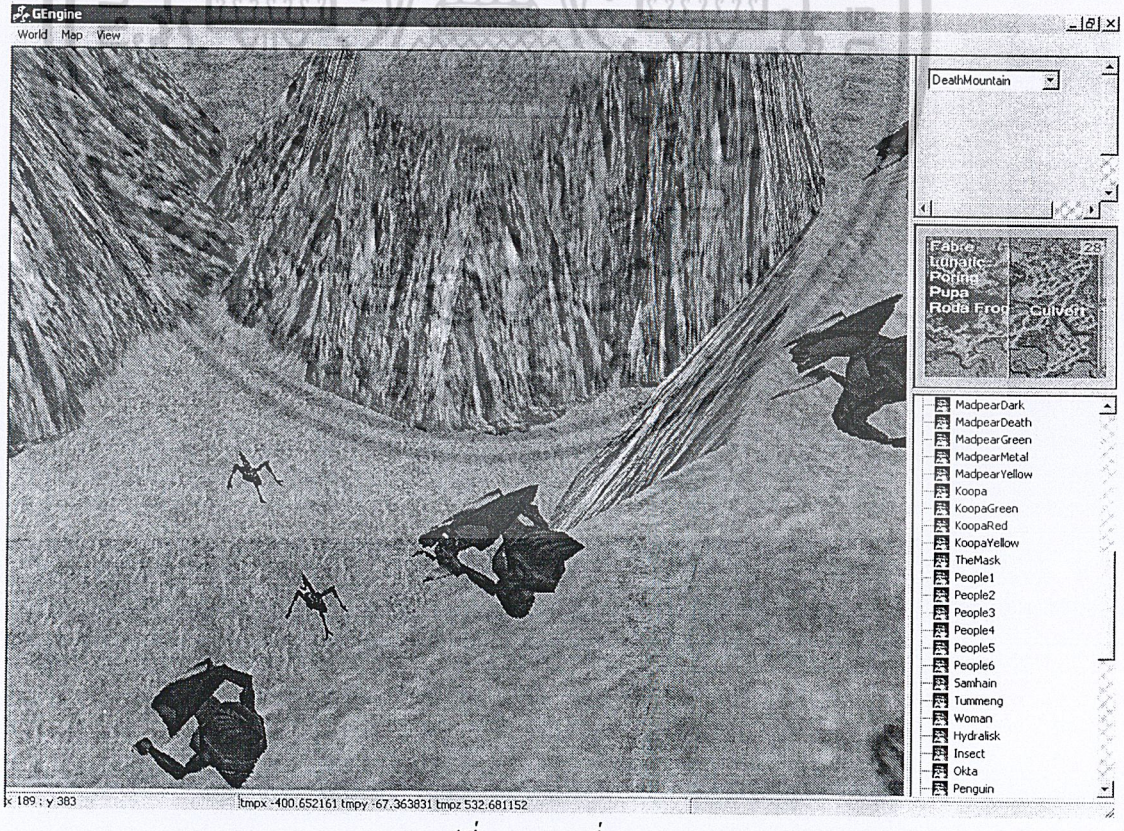
รูปที่ 11-1 แสดงการจัดแผนที่

11.2.2 เข้าไปจัดฉากแต่ละฉาก



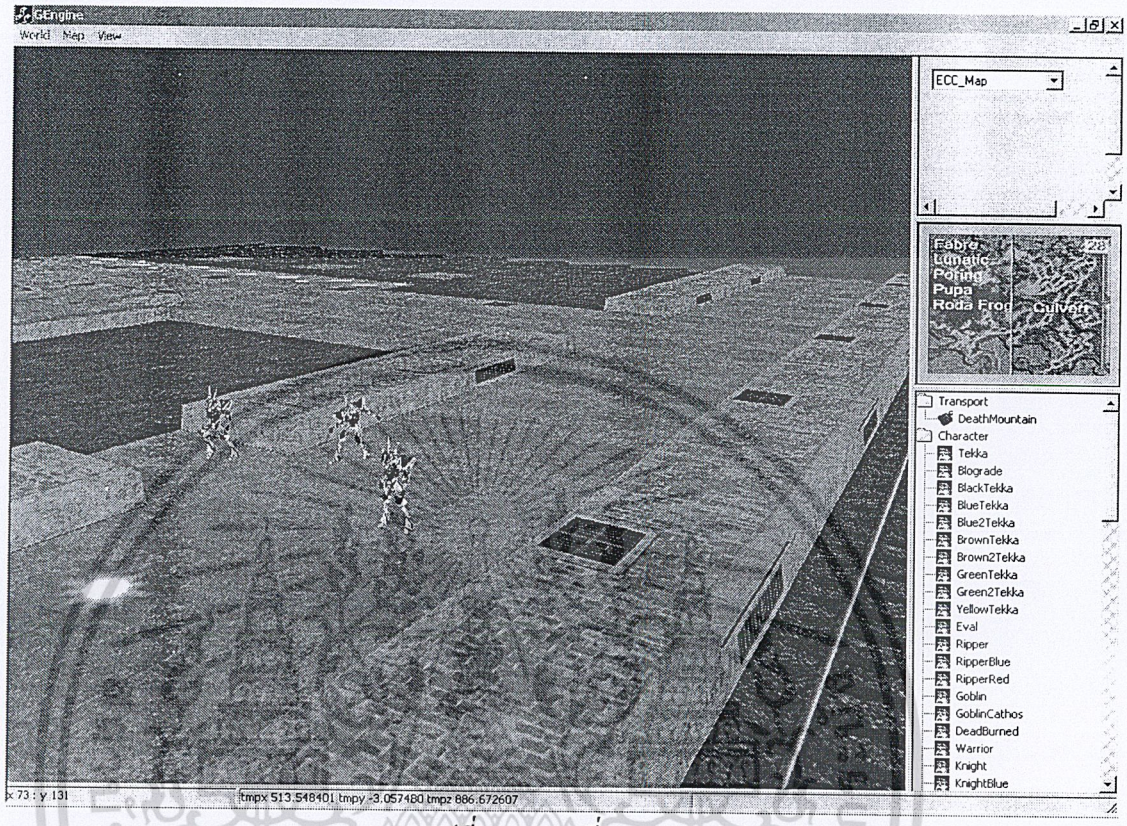
รูปที่11-2 ฉากที่1

ฉากที่ 1 ฉากยานอวกาศตกเป็นฉากที่อยู่บนทุ่งหญ้า คุณต้องทางไปยังสถานีให้ได้



รูปที่ 11-3 ฉากที่2

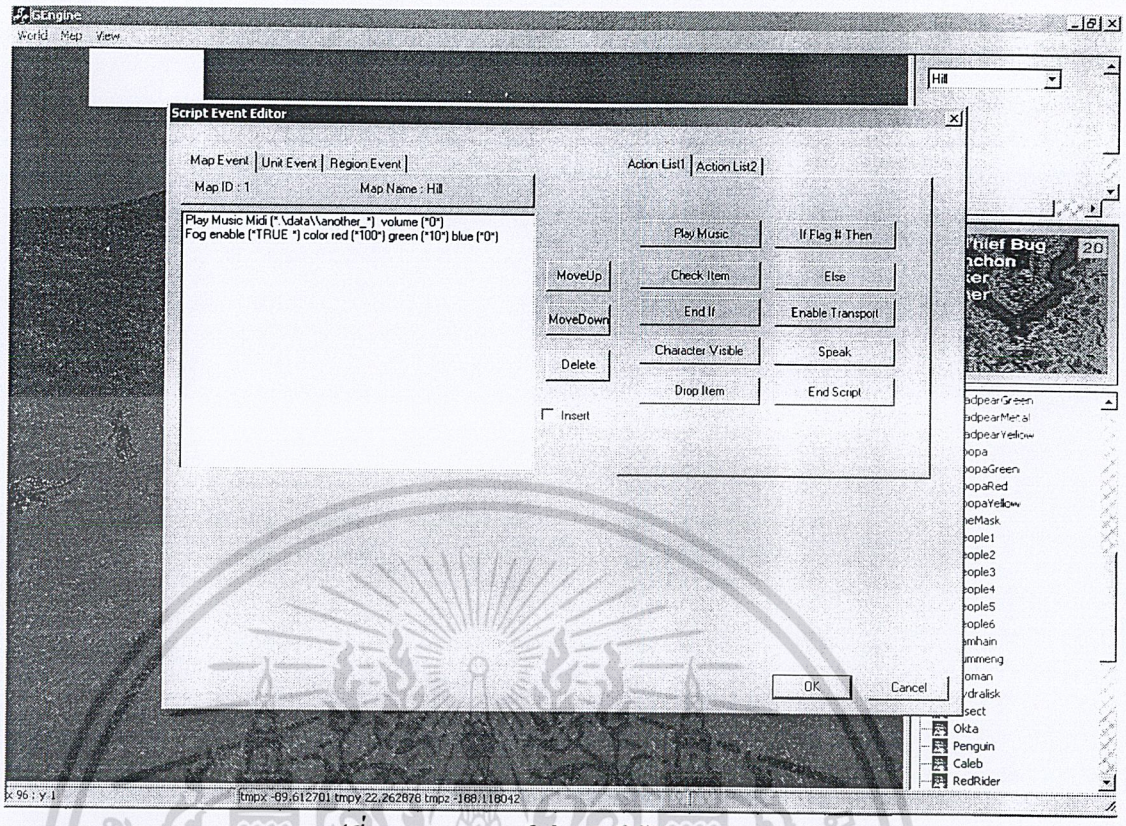
ฉากที่2 ฉากที่ต้องเดินผ่านฝูงสัตว์ประหลาดต่างดาว



รูปที่ 11-4 ฉากที่ 3

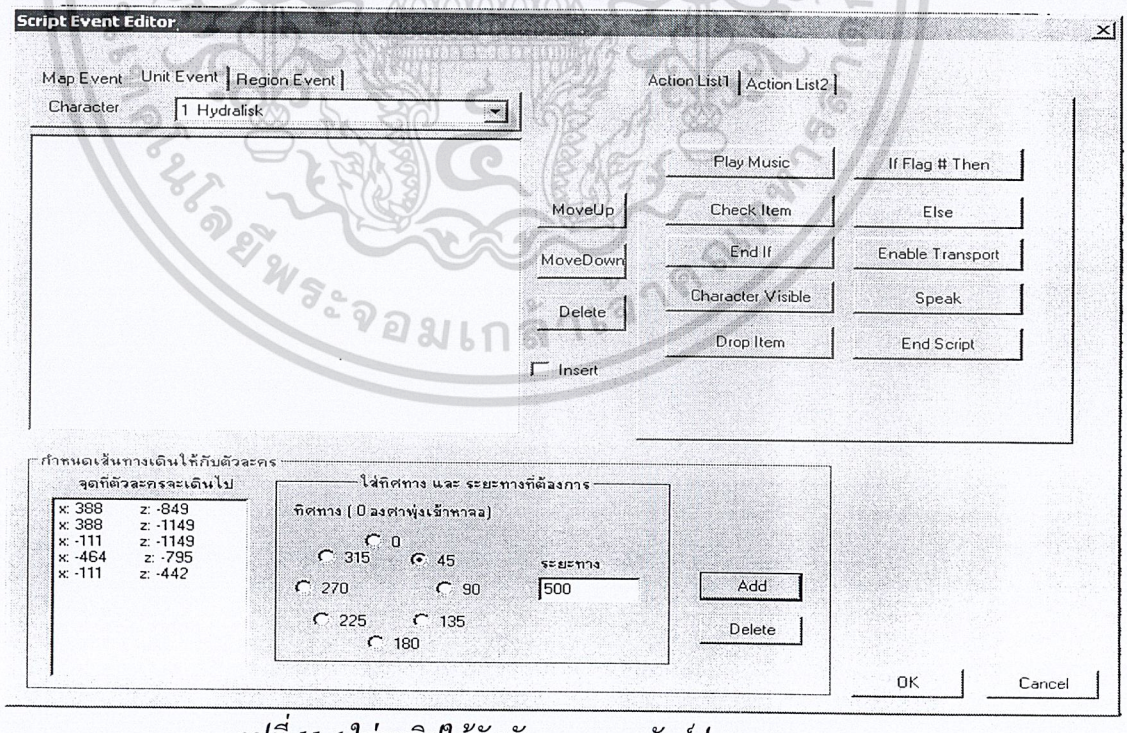
ฉากที่3 ฉากสถานียานอวกาศ คนที่สถานีกำลังรอคุณอยู่

11.2.3 สร้างEvent ให้กับตัวละคร ด้วยส่วน Script Event Editor ของ GUI Game Engine



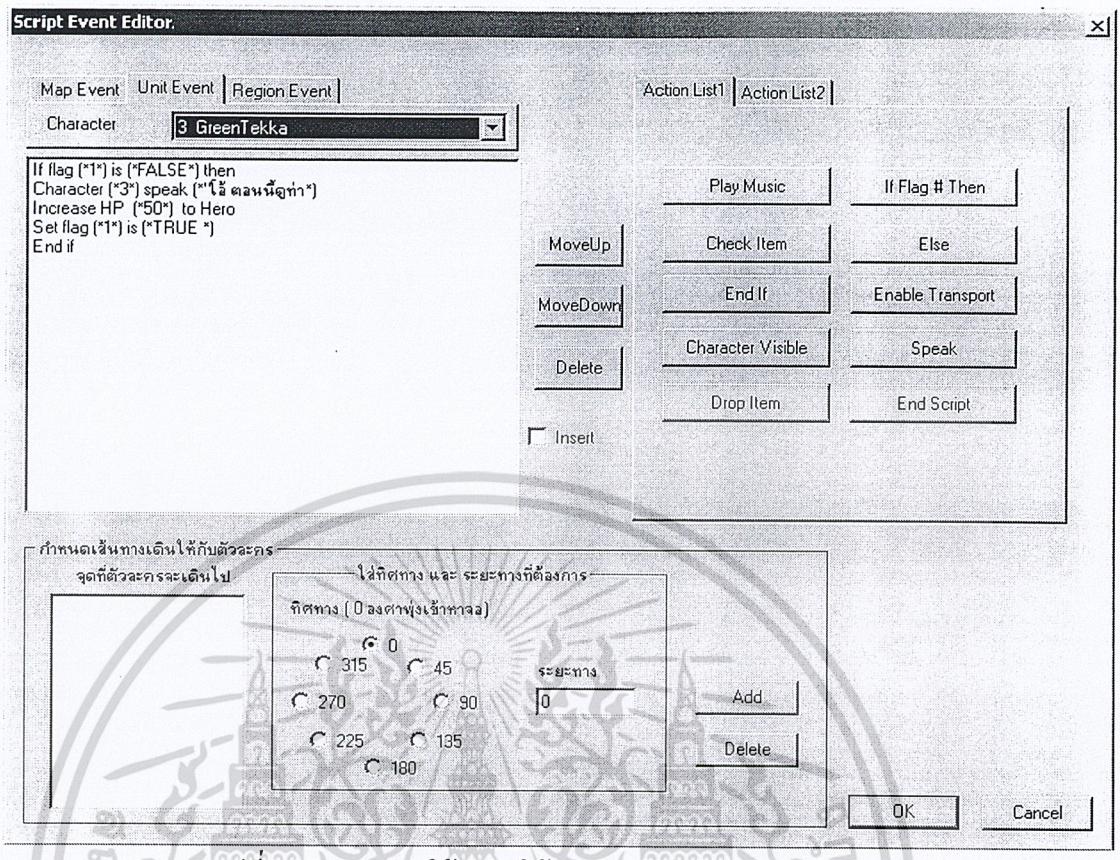
รูปที่ 11-5 แสดงการใส่สคริปให้กับฉาก

ในที่นี้จะใส่ หมอกและเสียงให้กับฉาก แรกให้มีบรรยากาศในเกมแบบอึมครึม



รูปที่ 11-6 ใส่สคริปให้กับตัวละครพวกสัตว์ประหลาด

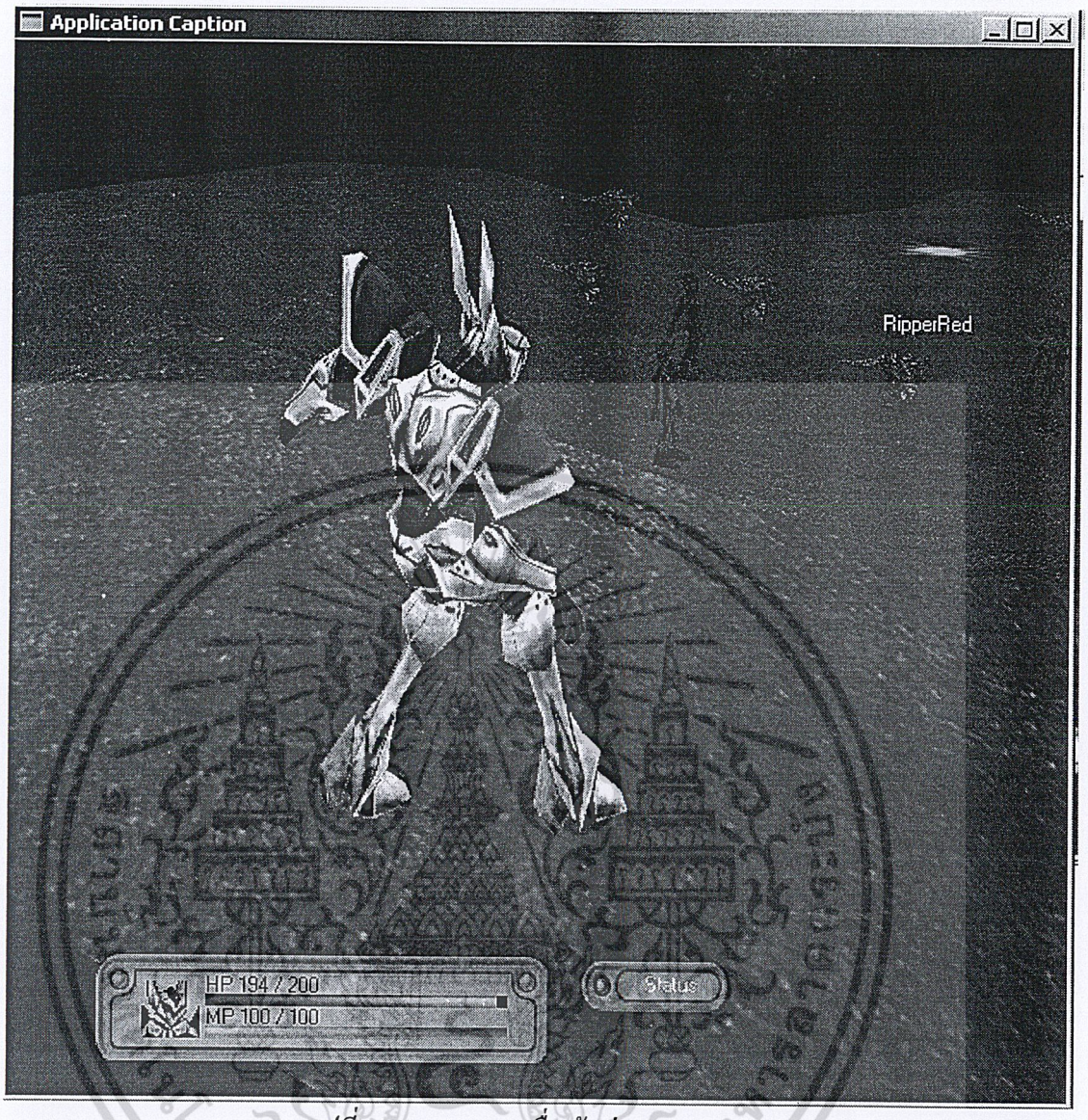
ในส่วนนี้เป็นการกำหนดเส้นทางเดินให้กับตัวละครพวกสัตว์ประหลาด



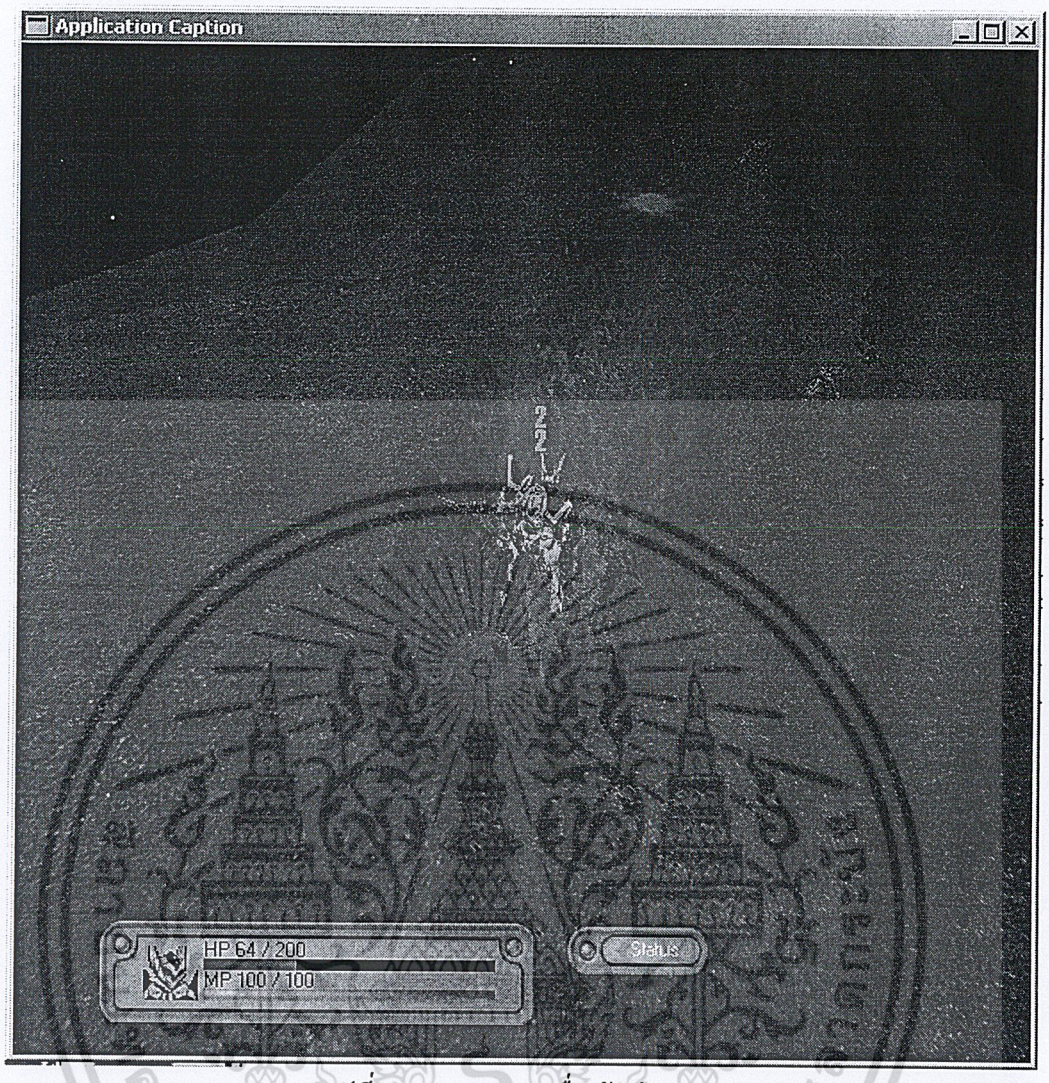
รูปที่ 11-7 แสดงการใช้สคริปให้กับตัวละครในฉากสุดท้าย

ส่วนนี้จะเป็นการใส่ Action script ให้กับตัวละครในฉากสุดท้าย ซึ่งเป็นการพูดและเพิ่มพลังให้ตัวฮีโร่

1. ไปยังโฟลเดอร์ที่ได้สร้างเกมไว้ และตรวจดูคร่าวๆ ว่ามี File Script ครบรึเปล่า
2. ทดลองรันเกมและทดลองเล่นดูว่าเป็นไปตามเนื้อเรื่องรึเปล่า
   ซึ่งได้ผลการทดลองดังนี้



รูปที่ 11-8 แสดงภาพเมื่อเข้าเล่นเกม 1

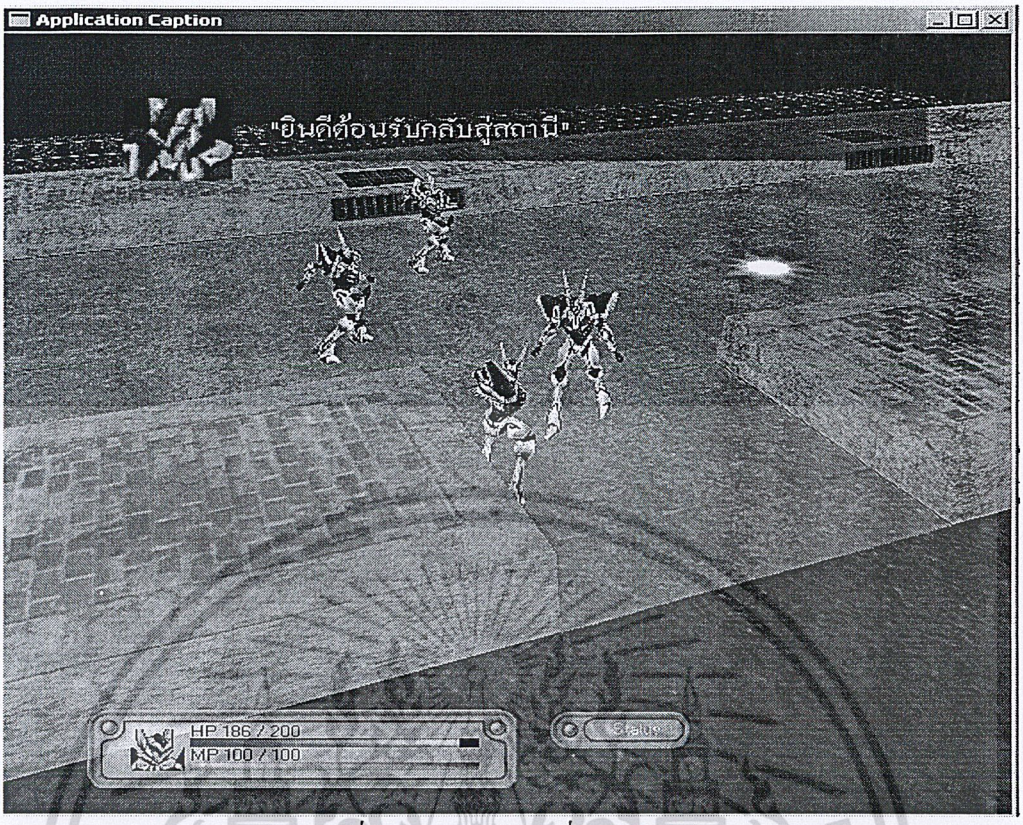


รูปที่ 11-9 แสดงภาพเมื่อเข้าเล่นเกม 2

จากรูปจะเป็นฉากแรกที่เราจะต้องหาทางไปยังสถานี ซึ่งพวก สัตว์ประหลาดที่เราได้วางไว้จะเข้ามาโจมตีเราเมื่อเราเดินเข้าไปใกล้มัน สังกตว่าจะมีหมอกที่เราใส่ไว้ใน map event ด้วย

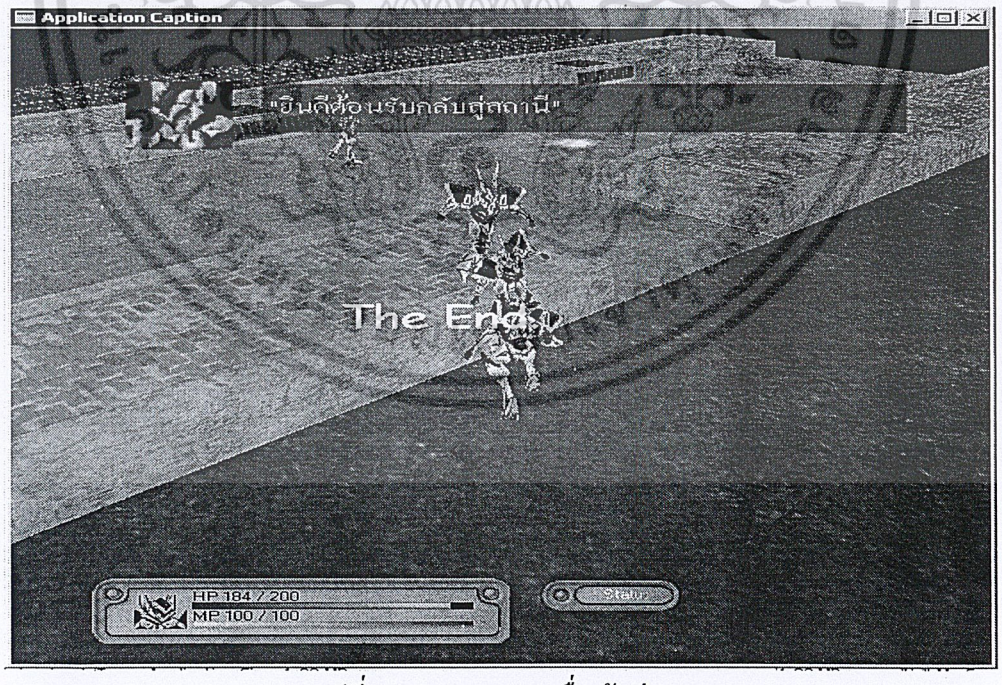
รูปที่ 11-10 แสดงภาพเมื่อเข้าเล่นเกม 3

ฉากที่ 2 นี้แสดงการเข้ามาโจมตีของตัวละครสัตว์ประหลาด และได้สังเกตผลการสร้างเส้นทางให้กับตัวละครด้วยแล้วว่ามันสามารถเดินได้ถูกต้องตามเส้นทางที่ได้กำหนดไว้



รูปที่ 11-11 แสดงภาพเมื่อเข้าเล่นเกม 4

ฉากสุดท้ายนี้เป็นการทดสอบให้ตัวละครพูดตามscript ที่ได้ใส่ไว้ให้กับตัวละคร



รูปที่ 11-12 แสดงภาพเมื่อเข้าเล่นเกม 5

เกมจะจบเมื่อเรากลับถึงสถานีและได้คุยกับคนในสถานีนี้

<u>สรุปผลการทดลองที่ได้จากการทดลองเล่นเกมที่พัฒนาด้วย GUI Game Engine</u>

เกมที่ได้ทดลองเล่นมีความสมบูรณ์ตามขอบเขตที่ได้บอกไว้ข้างต้นทุกประการ คือมันสามารถทำตามสิ่งที่ต้องการที่ได้สร้างไว้ในส่วน GUI ได้ทุกส่วน จากการทดลองพัฒนาเกมด้วยGUI นี้ได้แสดงให้เห็นแล้วว่าสามารถสร้างเกมได้อย่างรวดเร็วโดยที่ไม่ต้องมีการ codingเลย แต่ปัญหาที่พบก็คือเฟรมเรท ที่ลดลงถ้าเกิดในฉากมีการใส่ตัวละครไว้มากๆ และเป็นตัวละครที่แตกต่างกัน

บทที่ 12
บทวิจารณ์และสรุป

12.1 สรุปผลการวิจัย

จากการทดลองพัฒนาเกมจาก เกมเอนจินที่ได้สร้างขึ้นมาพบว่าผู้ใช้สามารถพัฒนาเกมได้อย่างรวดเร็ว เพราะลดเวลาที่ผู้ใช้ต้องไปทำฉาก เขียนโปรแกรม และอื่นๆ การเคลื่นไหวของตัวละครเป็นไปตามเส้นทางที่ได้วางเอาไว้อย่างถูกต้อง การแก้ไขรายละเอียดต่างๆภายในเกมได้สะดวก เช่นค่าพลังตัวละครเป็นต้น ซึ่งตรงตามขอบเขตที่ได้วางเอาไว้ แต่เกมเอนจินที่ได้พัฒนานี้จะไม่สามารถสร้างเกมแนวอื่นๆได้นอกจาก แนวเกมที่ได้วางไว้แต่แรกคือ RPG เพราะ GUI ในส่วนที่เป็น Game Player ผู้ใช้ไม่สามารถเปลี่ยนแปลงได้

ตลอดเวลาที่ได้พัฒนาเกมเอนจินต์ขึ้นมานั้นมีอุปสรรคและปัญหามากมายรวมทั้งเวลาอันจำกัดและข้อมูลที่มีอยู่น้อยนิด โดยเฉพาะที่เป็นทฤษฎีที่เกี่ยวข้องกับการออกแบบและพัฒนาเกมโดยตรงนั้นหายากมาก

12.2 แนวทางในการพัฒนาต่อ

- สร้างหน้าต่างสำหรับการเพิ่มตัวละคร ฉากต่างๆ
- เพิ่มชุดคำสั่งใน Action Library เพื่อให้มีรูปแบบของการกระทำมากยิ่งขึ้น
- เพิ่มส่วนที่ติดต่อกับระบบ network
- เพิ่มความความหลากหลายในแนวเกมที่ GUI จะสร้างออกมาได้ เพื่อทำให้เกมส์ดูหลากหลายมากยิ่งขึ้น
- ปรับปรุงหน้าต่างส่วนที่ติดต่อกับผู้ใช้ให้สะดวกมากยิ่งขึ้น
- เพิ่มส่วนที่สามารถสร้างฉากได้เองโดยไม่จำเป็นต้องสร้าง model มาจากโปรแกรมอื่น

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