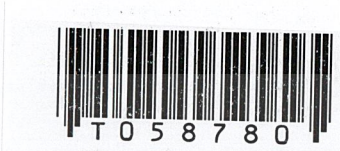
การพัฒนาเกมสามมิติสำหรับการเล่นเป็นทีม

3D SHOOTING TEAM GAME



กรวิภา ชานนนารถ
กัมพล เสวกวงศ์พระ
ชัยพร เรือนทองมา

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

3D SHOOTING TEAM GAME

KORNVIPA CHANONNART

KAMPOL SAWEKVONGPRA

CHAIYAPORN RUANTONGMA

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCES

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2004

หัวข้อปัญหาพิเศษ	การพัฒนาเกมสามมิติสำหรับการเล่นเป็นทีม 3D SHOOTING TEAM GAME	
ชื่อนักศึกษา	นางสาวกรวิภา ชานนนารถ	44050395
	นายกัมพล เสวกวงศ์พระ	44050398
	นายชัยพร เรือนทองมา	44050408
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	อ.กฤษฎา บุศรา	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2547

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	รศ.ธีรวัฒน์ ประกอบผล	
กรรมการ	อ.ศังกรศรัณย์ ล่องชูผล	
กรรมการและอาจารย์ที่ปรึกษา	อ.กฤษฎา บุศรา	

(รองศาสตราจารย์ ดร.วีระ บุญจริง)

หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ การพัฒนาเกมสามมิติสำหรับการเล่นเป็นทีม

ชื่อนักศึกษา นางสาวกรวิภา ชานนนารถ 44050395

นายกัมพล เสวกวงศ์พระ 44050398

นายชัยพร เรือนทองมา 44050408

ปริญญา วิทยาศาสตรบัณฑิต

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สาขาวิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2547

อาจารย์ที่ปรึกษา อ.กฤษฎา บุศรา

บทคัดย่อ

ปัจจุบันในวงการอุตสาหกรรมการผลิตโปรแกรมเกมคอมพิวเตอร์ ได้มีการปรับตัวอย่างรวดเร็ว และมีผู้ผลิตมากมายในตลาด เพราะผู้ผลิตต่างๆได้นำเทคโนโลยีชั้นสูงเข้ามาช่วยในการพัฒนาคุณลักษณะและเทคนิคในเกม ดังนั้นทำให้คณะศึกษาได้เลือกศึกษาโครงการเกี่ยวกับการสร้างโปรแกรมเกมสำหรับคอมพิวเตอร์ ซึ่งในมุมมองของทีมงาน โปรแกรมเกมจะไม่ได้วัดกันด้วยแง่ธุรกิจ หรือยอดขาย หรือความนิยมของผู้เล่นเพียงอย่างเดียว แต่จะต้องวัดที่ความท้าทาย, ความแปลกใหม่ และการตอบสนองระหว่างผู้เล่นกับผู้เล่นอื่น ซึ่งคณะผู้ศึกษาได้ทำการ ศึกษา วิจัยเทคนิคต่าง ๆ จากโปรแกรมอื่น ๆ ในท้องตลาด

โครงงานโปรแกรมเกมนี้จะนำเสนอความท้าทายในการตอบสนองระหว่างผู้เล่นหลายคน ผ่านทางระบบเครือข่ายในลักษณะของไคล์เอนท์ เซิร์ฟเวอร์ และมีการติดต่อกับฐานข้อมูล ซึ่งจะมีการตอบสนองของโปรแกรมในสถานการณ์ต่างๆจากการสั่งงานคีย์บอร์ดและเมาส์ โดยที่เกมส์จะแบ่งผู้เล่นเป็นทีม และมีมุมมองบุคคลที่ 1 รวมทั้งมีเสียงเอฟเฟ็คท์ที่ช่วยจำลองสถานการณ์ให้เหมือนสนามรบจริง นอกจากนี้ผู้เล่นยังสามารถเลือกซื้ออาวุธในการต่อสู้ และติดต่อสื่อสารกับผู้เล่นภายในทีมเดียวกันได้

Special Project Title	3D SHOOTING TEAM GAME	
Students	Miss Kornvipa Chanonnart	44050395
	Mr. Kampol Sawekvongpra	44050398
	Mr. Chaiyaporn Ruantongma	44050408
Degree	Bachelor of Science	
Department	Mathematics and Computer Sciences, Faculty of Science	
Programme	Computer Sciences	
Academic Year	2004	
Special Project Advisor	Kridsada Budsara	

ABSTRACT

The game software industry had fast development and more competitors in the market. Because they adapt high technology for develop feature and technique in game. With this reason, it had more challenge for us for choose project about build game software. With our focus, The Game software did not performance by business or sales volume or popular game only. But we will performance with the new challenge, new style, and interactive of game that we have research technique from other products in market.

Our game software project will offers players a new challenge style in the interaction between multiple players via LAN in Client/server architecture and connect to database for storing information of players .The game will have interactive playing with mouse and keyboard . Players will be separated into teams giving first person view and actual sound effects, simulating real combat play. Moreover players can select to buy weapons for shooting and can have a conversation within teams.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องการพัฒนาเกมสามมิติสำหรับการเล่นเป็นทีมสามารถสำเร็จลุล่วงได้ด้วยดี ด้วยความช่วยเหลือ และความร่วมมือจากหลายๆท่าน คณะผู้จัดทำต้องขอขอบพระคุณบุคคลที่มีส่วนช่วยให้การทำงานครั้งนี้เสร็จไปได้ด้วยดี คือ อ.กฤษฎา บุศรา อาจารย์ที่ปรึกษาปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำแนวคิดต่างๆในการทำปัญหาพิเศษนี้ ดูแลเอาใจใส่ และให้การสนับสนุนทางด้านซอฟท์แวร์ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของโครงงานปัญหาพิเศษฉบับนี้และทุกๆท่านที่ช่วยตอบกระทู้ทางอินเตอร์เนต

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนการทำปัญหาพิเศษนี้สำเร็จลุล่วงไปด้วยดี รวมทั้งเพื่อนๆ และพี่ๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2548

สารบัญ

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

ภาพที่ **หน้า**

ภาพที่ **หน้า**

บทที่1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

เนื่องจากปัจจุบันอุตสาหกรรมเกมได้มีการพัฒนาและเติบโตอย่างรวดเร็ว โดยอุตสาหกรรมเกมเป็นอุตสาหกรรมที่มีการแข่งขันทางด้านการตลาดที่มีมูลค่าสูง ทั้งรัฐบาลยังได้ให้การสนับสนุนในการส่งออก software ต่างๆ เพราะฉะนั้นการศึกษาวิธี และหลักในการสร้างระบบเกมจึงเป็นสิ่งที่น่าสนใจ โดยที่ผู้ศึกษาจะได้รับความรู้เกี่ยวกับวิธีการสร้างระบบเกม วิธีการติดต่อกับฮาร์ดแวร์ ในส่วนของการอินพุท / เอาท์พุท รวมถึงวิธีการใช้ เครื่องมือต่างๆ ในการสร้างรูปภาพภาพเคลื่อนไหว อีกทั้งยังสามารถเขียนโปรแกรมเกม เพื่อตอบโต้กับผู้เล่นได้ในหลายๆรูปแบบอีกด้วย โดยหวังว่า จะเป็นประโยชน์ต่อการนำไปศึกษาต่อ เพื่อเป็นแนวทางเบื้องต้นสำหรับผู้ที่สนใจการพัฒนาเกม และยังเป็นแรงผลักดันให้เกิดอุตสาหกรรมการผลิตเกมในประเทศไทย

1.2 วัตถุประสงค์ของการศึกษา

1.2.1. เพื่อศึกษาระบบและวิธีการสร้างเกมแบบ 3D สำหรับผู้เล่นหลายคน

1.2.2. เพื่อพัฒนาโปรแกรมบน Windows ด้วยเทคโนโลยีการแสดงภาพและประมวลผลภาพโดยใช้โปรแกรม Microsoft DirectX

1.2.3. เพื่อพัฒนาโปรแกรมแบบ Object Oriented Programming (OOP)

1.2.4. เพื่อพัฒนาระบบที่มีสถาปัตยกรรมแบบ client/server ซึ่งสนับสนุนการเล่นเป็นทีมผ่านทางเครือข่าย Local Area Network (LAN)

1.3 ขอบเขตของการศึกษา

1.3.1. พัฒนาซอฟต์แวร์เกม โดยใช้โปรแกรม Microsoft Visual C++ version 6.0

1.3.2. ใช้ Microsoft DirectX เป็น library สำคัญในการพัฒนา

1.3.3. เป็นซอฟต์แวร์ที่ทำงานได้บน Windows 98 ขึ้นไป และในเครื่องที่มีการติดตั้งการ์ด 3D

1.3.4. พัฒนาซอฟต์แวร์ที่มีการติดต่อกันแบบ client/server ในเครือข่าย LAN ทางผ่าน port

1.4 ขั้นตอนในการดำเนินงาน

1.4.1. ศึกษาเกี่ยวกับระบบการสร้างเกม

1.4.2. ศึกษาและรวบรวมข้อมูลภาษาในการโปรแกรม และเครื่องมือต่างๆ

ศึกษาถึง Functionต่างๆ และวิธีการเขียนโปรแกรม Microsoft DirectX และ Microsoft Visual C++ 6.00 ในการ

- ตรวจสอบการแสดงผลการแสดงผล
- การแสดงผล
- การติดต่อกันผ่านระบบเครือข่าย
- การตรวจสอบ event ต่างๆ

1.4.3. วิเคราะห์และออกแบบระบบ

ทำการวิเคราะห์ปัญหา รวบรวมแนวทางที่จะทำให้บรรลุถึงเป้าหมาย วิเคราะห์ถึงข้อดีและข้อเสียของแต่ละแนวทาง และเลือกแนวทางที่ดีที่สุด เพื่อทำการออกแบบระบบ จากนั้นวิเคราะห์ระบบที่ได้ออกแบบว่ามีข้อบกพร่องตรงไหน และจัดการแก้ไข

1.4.4. พัฒนาโปรแกรม

พัฒนาโปรแกรมตามที่ได้วิเคราะห์ และออกแบบไว้ โดยจะทำการพัฒนาโปรแกรมทีละส่วน(component) โดยแบ่งได้ดังนี้

- การแสดงฉากหลัง จะใช้ SkyBox ซึ่งเป็นสภาพแวดล้อมที่อยู่รอบๆด่านที่ตัวละครกำลังยืนอยู่ และเขียนโปรแกรมโดยใช้ฟังก์ชั่นที่มีอยู่ใน DirectX มาช่วยในการ Transformation , การ set texture และ การ render เป็นต้น
- การแสดงมุมมองของแต่ละผู้เล่น ผู้เล่นจะมีมุมมองเป็นแบบ 3 มิติ คือสามารถหมุนได้ 360 องศา ในแนวแกน y และสามารถก้ม และเงยได้ ซึ่งจะต้องมีการเขียนโปรแกรมในการ render แต่ละ model ใน world space ใหม่ทุกครั้งเมื่อเกิด event จาก mouse และ keyboard ขึ้น และมีส่วนแสดงผลที่ช่วยสนันสนุนการเล่น คือ Map จะใช้แสดงตำแหน่งปัจจุบันของแต่ละตัวละครที่อยู่ในเกม และ ส่วนแสดงสถานะของตัวละครที่จะแสดง พลังชีวิต , พลังเสื้อเกราะ , จำนวนกระสุนในปืน และ จำนวนกระสุนสำรองที่เหลืออยู่ของผู้เล่น
- การคำนวณการชนของวัตถุต่างๆ จะใช้ Vector ในการเก็บค่าตำแหน่ง และทิศทางของวัตถุในปัจจุบันที่กำลังเคลื่อนที่ไปเรื่อยๆ
- การติดต่อกันของผู้เล่นแต่ละคนผ่านระบบเครือข่าย จะใช้ direct playที่เป็นส่วนประกอบที่ประกอบที่ใช้ติดต่อกับอุปกรณ์ทางด้าน Network การอ้าง Port ต่างๆ หรือการสร้างเกมที่สามารถเล่นผ่านเครือข่ายได้
- การแสดงผลเมื่อได้รับ event ต่างๆ จะมีการเขียนโปรแกรมส่วนที่เป็น event handler เอาไว้เพื่อดักจับ event ต่างๆเช่น การคลิกmouse หรือ keyboard แล้วทำงานในส่วนที่กำหนดไว้ เช่น ส่ง message ไปยังเครื่องอื่นเพื่อให้แสดงผลตาม event ที่เกิด และมีการ render เพื่อเปลี่ยนมุมมองของผู้เล่นใหม่

1.4.5.ทำการทดสอบและปรับปรุงโปรแกรม

ทำการทดสอบเกมโดยตรวจสอบ

1.4.5.1 การแสดงผล

- ฉากหลัง
- view ของผู้เล่น
- การแสดงผลเมื่อเกิด event ต่างๆ

1.4.5.2 การตอบสนองต่อการทำงาน

- การตอบสนองเมื่อมีการ click mouse , เลื่อน mouse
- การตอบสนองเมื่อมีการ กด keyboard
- การตอบสนองเมื่อมี event อื่น มากระทบ

เมื่อพบข้อผิดพลาด จึงทำการปรับปรุงโปรแกรมและทำการทดสอบต่อไป เพื่อให้ตรงกับเป้าหมายที่ตั้งไว้มากที่สุด

1.4.6. จัดทำเอกสารประกอบรายงานพิเศษ

1.5 ข้อตกลงเบื้องต้น

1.5.1 โปรแกรมรัน เฉพาะระบบปฏิบัติการ Windows ตั้งแต่ เวอร์ชั่น XP ขึ้นไป

1.5.2 ในระบบปฏิบัติการต้องมี Microsoft DirectX software ตั้งแต่เวอร์ชั่น 8

1.5.3 ต้องการคอมพิวเตอร์ที่มีโปรเสสเซอร์ตั้งแต่ระดับ pentium 4 ความเร็ว 1.80 GHz ขึ้นไป

1.5.4 มีหน่วยความจำอย่างน้อย 256 Mbyte

1.5.5 ต้องมี ฮาร์ดแวร์อินพุท ประเภท Mouse และ Keyboard

1.6 ข้อจำกัดของการศึกษา

1.6.1 ระยะเวลาที่ทำการศึกษา

1.6.2 ความรู้และความเข้าใจที่ได้จากการศึกษา

1.6.3 ทฤษฎี ที่กำหนดขึ้นมาแล้วในการศึกษาปัญหา

1.6.4 เทคโนโลยีการศึกษาปัญหาพิเศษ

1.7 ประโยชน์ที่คาดว่าจะได้รับ

1.7.1. ได้มีทักษะในการออกแบบระบบและการเขียนโปรแกรมเชิง object

1.7.2. ได้มีทักษะการเขียนโปรแกรมเพื่อติดต่อกันระหว่างเครื่องหลายๆเครื่องในเครือข่าย LAN ทางผ่านportโดยใช้ DirectPlay

1.7.3. ได้มีทักษะในการพัฒนาระบบแบบ client/server

1.7.4. ได้มีทักษะในการเขียนโปรแกรมแบบ Event Driven Programming ที่มีการประมวลผลการทำงานต่างๆ เมื่อเกิด event จาก mouse และ keyboard

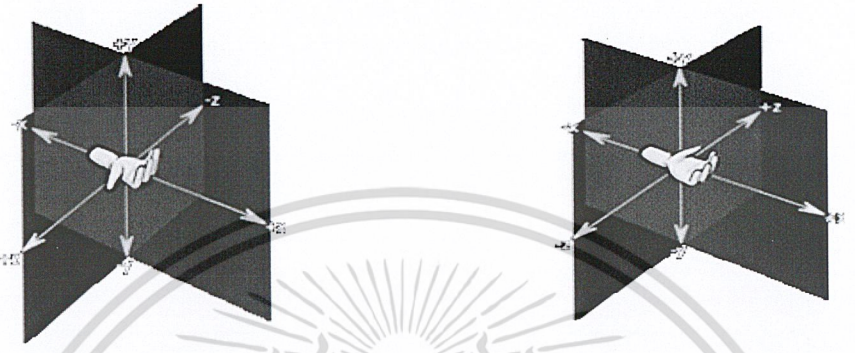
1.7.5. ได้นำความรู้เกี่ยวกับคณิตศาสตร์ และฟิสิกส์มาใช้ เช่นการคำนวณการชนกันของวัตถุในแบบ 3 มิติ

1.7.6. ได้รับความรู้เกี่ยวกับการใช้โปรแกรม Graphic 3D เพื่อสร้างภาพเคลื่อนไหวของโมเดลตัวละคร เช่น 3Dstudio MAX และ Character Studio

1.7.7. ได้มีทักษะในการแก้ปัญหา และศึกษาเกี่ยวกับสาเหตุและวิธีการแก้ปัญหา

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 3-D Coordinate System

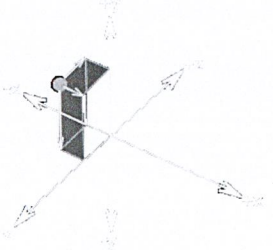


Right-Hand Coordinate System Left-Hand Coordinate System

หมายถึงทิศทางของค่าในแนวแกน x, y และ z ของจุดกำเนิด (Origin) ในระบบสามมิติ ซึ่งจุดกำเนิดจะมีค่า x, y และ z เป็น 0, 0 และ 0 ตามลำดับและโดยปกติแล้วทิศทางของค่า x จะเพิ่มขึ้นในทิศทางไปทางด้านขวาและลดลงไป ทางด้านซ้ายของจุดกำเนิด ส่วนค่า y จะเพิ่มขึ้นไปทางด้านบน และลดลงไปทางด้านล่าง แต่ค่า z จะมีให้เราเลือกอยู่ 2 แบบ คือแบบแรกจะเพิ่มขึ้นในทิศทางเข้าไปในจอภาพ และแบบที่สองจะเพิ่มขึ้นในทิศทางออกจากจอภาพ โดยแบบแรกเขาเรียกว่า Left-Hand Coordinate System ส่วนแบบที่สองจะเรียกว่า Right-Hand Coordinate System จากรูปจะแทนทิศทางที่นิ้วหัวแม่มือชี้ไปด้วยค่า +z, แทนทิศทางที่มือหันไปด้วย +x และแทนทิศทางของนิ้วอื่นๆ ที่ตั้งขึ้นด้วยค่า +y ซึ่งใน Direct3D จะนิยมใช้แบบ Left-Hand สำหรับฟังก์ชั่นที่จะใช้บอก Direct3D ว่าเราจะใช้ 3-D Coordinate แบบไหนนั้น ก็คือ D3DXMatrixLookAtLH() กับ D3DXMatrixLookAtRH()

2.1.1 เวกเตอร์ (Vector)

ใน Direct3D มักจะใช้เวกเตอร์อยู่ 3 ประเภท คือ 2-D Vector, 3-D Vector และ 4-D Vector ตัวอย่างของ 3-D Vector เช่นตำแหน่งของจุดในโลก 3 มิติ จะมีค่า 3 ค่าคือ ค่าในแนวแกน x, y และ z



นอกจากนี้ ค่าของสีของ Pixel ที่มีรูปแบบเป็น RGB ก็ถือว่าเป็น 3-D Vector ด้วย เพราะมีค่า 3 ค่ามาประกอบกันเป็นสี คือ แดง (R), เขียว (G) และน้ำเงิน (B) โดยค่าแต่ละค่าจะมีทิศทางเริ่มจาก 0 ไปจนถึง 255 สำหรับ 4-D Vector ก็ได้แค่ตำแหน่งของจุดแบบ Transformed Vertex ซึ่งประกอบด้วยค่าในแนวแกน x, y, z และค่า RHW (RHW หรือ Reciprocal of Homogeneous W ก็คือตำแหน่งของจุดโฟกัสของสายตาหรือของกล้องถ่ายรูป ซึ่งค่านี้มันจะอยู่ในแนวแกน z และวัดจากตำแหน่งของวัตถุเข้าหาจุดโฟกัสถ้ามีค่าเป็น 1.0 จะหมายถึงคือจุดเดียวกับตำแหน่ง z (โฟกัสที่วัตถุนั้นพอดี))นอกจากนี้ ค่าของสีของ Pixel ที่มีรูปแบบเป็น ARGB ก็ถือว่าเป็น 4-D Vector เพราะมีค่า Alpha หรือความโปร่งใสเพิ่มเข้ามา

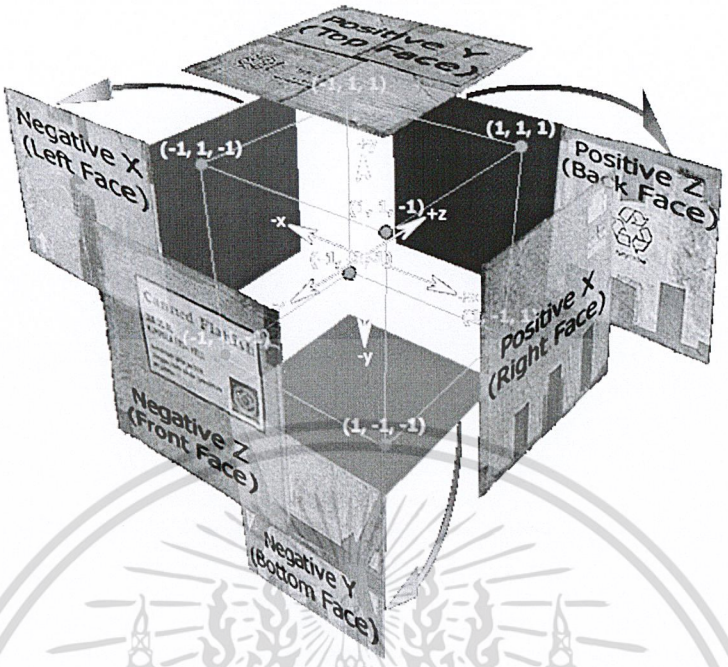
2.1.2 **เวอร์เท็กซ์** (Vertex)

คือจุดในโลก 3 มิติ ซึ่งเวอร์เท็กซ์นี้จะประกอบไปด้วยเวกเตอร์จำนวน 1 เวกเตอร์เป็นอย่างน้อย เช่น Vertex อาจจะ ประกอบด้วย 3-D Vector จำนวน 2 อัน โดยที่อันแรกจะใช้แสดงถึงตำแหน่งของจุดในแนวแกน x, y และ z ส่วน 3-D Vector อันที่สองก็คือสี RGB เพื่อใช้สำหรับกำหนดว่า จะให้จุดนั้นมีสีอะไร

การสร้าง Vertex ใน Direct3D จะใช้วิธีมาตรฐานเหมือนอย่างที่ใช้กันใน OpenGL API และ Glide API คือจะใช้วิธีสร้าง Flexible Vector Format หรือเรียกย่อๆ ว่า FVF ซึ่งเป็น Structure ที่ประกอบไปด้วยเวกเตอร์ที่เราต้องการจะใช้งานเท่านั้น เวกเตอร์ไหนที่เราไม่ต้องการใช้ก็ให้ข้ามมันไป โดยจะต้องเรียงลำดับเวกเตอร์และใช้ประเภทของตัวแปรตามรูปแบบในภาพข้างล่างนี้ครับ โดยเรียงจากบนลงล่าง ช่องด้านซ้ายเป็นชื่อของเวกเตอร์ ส่วนช่องด้านขวาเป็นรายละเอียดข้อมูลที่อยู่ในเวกเตอร์นั้นๆ

2.1.3 Model Space

Model Space คือพิกัด (3-D Coordinate) ที่ใช้ภายในโมเดลหรือวัตถุ แต่ละอัน ซึ่งมีขอบเขตเป็นสี่เหลี่ยม โมเดลจะประกอบขึ้นมาจากรูปสามเหลี่ยมหลายๆ ชิ้น และรูปสามเหลี่ยมแต่ละชิ้นก็เกิดขึ้นจาก Vertex จำนวน 3 จุด โดยแต่ละจุดก็จะอ้างอิงถึงจุดกำเนิด (Origin) จุดเดียวกัน (จุดกำ เนิดจะมีค่า x, y, z เป็น 0, 0, 0) ซึ่งปกติแล้ว จุดกำเนิดนี้จะอยู่ที่กลางโมเดลแต่ละอัน ซึ่งพิกัดต่างๆ ที่โมเดลแต่ละอันใช้นี้จะเรียกว่า Local Coordinate ขอบเขตของ Model Space จะกว้าง, ยาว และสูงแค่ไหนขึ้นอยู่กับขนาดของโมเดล



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

การวาดรูปรูปกล่องสี่เหลี่ยมลูกบาศก์ แต่ละด้านของกล่องก็จะเป็นรูปสี่เหลี่ยม ซึ่งสี่เหลี่ยมนี้จะต้องประกอบขึ้นมาจากรูปสามเหลี่ยม 2 รูป (การ์ด 3-D ส่วนมากจะ Render หรือทำงานกับรูปสามเหลี่ยม, เส้นตรง และจุดเป็นหลัก) และสามเหลี่ยมแต่ละรูปก็ประกอบขึ้นมาจาก Vertex จำนวน 3 จุด แต่ไม่จำเป็นว่า รูปสี่เหลี่ยม 1 รูปที่เกิดจากรูปสามเหลี่ยม 2 รูปมาประกบกันจะมี Vertex จำนวน 6 จุดเสมอไป มันขึ้นอยู่กับ Draw Primitive ที่เราจะใช้ ซึ่ง Draw Primitive ก็คือ "การวาดรูปทรงต่างๆ จากการเรียงของ Vertex แต่ละจุด" โดยรูปแบบของ Draw Primitive ใน Direct3D จะมีอยู่ 6 แบบดังนี้

สมมุติว่าใช้ Vertex จำนวน 6 จุด ในการวาด Primitive แต่ละแบบ โดย Vertex เรียงกันดังนี้

- Vertex จุดที่ 1 = (-5.0f, -15.0f, 0.0f)
- Vertex จุดที่ 2 = (0.0f, 5.0f, 0.0f)
- Vertex จุดที่ 3 = (3.0f, 0.0f, 0.0f)
- Vertex จุดที่ 4 = (10.0f, 5.0f, 0.0f)
- Vertex จุดที่ 5 = (10.0f, -5.0f, 0.0f)
- Vertex จุดที่ 6 = (15.0f, -10.0f, 0.0f)
- 1. Point List จะวาดจุด (Point) ตามลำดับการเรียงของ Vertex แต่ละจุด (1 Point ต่อ 1 Vertex) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_POINTLIST และผลลัพธ์จะเป็นไปตามรูปด้านล่างนี้

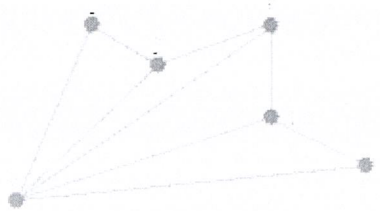
2. Line List จะวาดเส้นตรง (Line) เรียงตามลำดับการเรียงของ Vertex ทุกๆ 2 จุด โดยเส้นแต่ละเส้นจะแยกออกจากกัน (1 Line ต่อ 2 Vertex) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_LINELIST

3. Line Strip จะวาดเส้นตรงเรียงตามลำดับการเรียงของ Vertex โดยที่ Vertex ที่ 2 ของ Line ที่เพิ่งวาดไปจะกลายเป็น Vertex จุดที่ 1 ของ Line ถัดไป คือใช้ Vertex ร่วมกัน 1 จุด ผลลัพธ์ที่ได้คือ Line ที่ต่อกันไปเรื่อยๆ ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_LINESTRIP

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

5.Triangle Strip จะวาดสามเหลี่ยมเรียงตามลำดับการเรียงของ Vertex โดย Vertex ที่ 2 และ 3 ของรูปสามเหลี่ยมอันที่เพิ่งวาดไป จะกลายเป็น Vertex ที่ 1 และ 2 ของรูปสามเหลี่ยมรูปถัดไป คือ ใช้ Vertex ร่วมกัน 2 จุด ผลลัพธ์ที่ได้ก็คือ รูปสามเหลี่ยมที่ต่อกันไปเรื่อยๆ (ถ้าจะใช้วิธีนี้ ควรจะเรียง Vertex แบบซิกแซ็ก) ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLESTRIP

6.Triangle Fan จะวาดสามเหลี่ยมเรียงตามลำดับการเรียงของ Vertex โดย มี Vertex แรกสุดเป็น Vertex ที่ 1 ของสามเหลียมทุกรูป และ Vertex ที่ 3 ของรูปสามเหลี่ยมอันที่เพิ่งวาดไป จะกลายเป็น Vertex ที่ 2 ของรูปสามเหลี่ยมรูปถัดไป ผลลัพธ์ก็คือ จะได้รูปสามเหลี่ยมที่เรียงตัวต่อกันเป็นรูปที่คล้ายพัด ซึ่งค่าคงที่ที่ใช้ในกรณีนี้จะเป็น D3DPT_TRIANGLEFAN



ที่ต้องมี Draw Primitive หลายๆ แบบให้เราเลือกใช้ ก็เพื่อประหยัดหน่วยความจำ และเพิ่ม Frame Rate เพราะยิ่งเราใช้ Vertex น้อยเท่าไร ก็ยิ่งดีเท่านั้น ถ้าเราจะวาดกล่องโดยใช้ Draw Primitive เป็น Triangle List เราจะต้องใช้ Vertex จำนวน 36 จุด เพราะ สามเหลี่ยมแต่ละรูปเกิดขึ้นมาจาก Vertex จำนวน 3 จุด ดังนั้นจึงต้องใช้ Vertex 6 จุดต่อการวาดรูปสี่เหลี่ยม 1 รูปและกล่องมันมี 6 ด้าน ดังนั้นต้องใช้ Vertex ทั้งสิน 6 x 6 ซึ่งเท่ากับ 36 จุด แต่ถ้าเราใช้ Draw Primitive แบบ Triangle Strip เราจะใช้ Vertex แค่ 4 จุดต่อ 1 ด้านเท่านั้น รวมแล้วก็ 4 x 6 ซึ่งเท่ากับ 24 จุด

2.1.4World Space

ก็คือโลก 3 มิติ ซึ่งจะมีขนาดใหญ่กว่า Model Space เพราะโมเดลต่างๆ ที่เราสร้างจะถูกแปลงให้มาปรากฏอยู่ใน World Space โดยหลังจากที่โมเดลได้ถูกนำมาใส่ไว้ใน World Space นี้แล้ว พวกมันจะมีจุด Origin เป็นจุดเดียวกัน สำหรับขนาดความกว้างของ World Space จะเป็น 2 ยกกำลัง 32 และความสูงจะเป็น 2 ยกกำลัง 32 สำหรับความลึกก็ขึ้นอยู่กับหน่วยความจำในแนวลึก (Depth Buffer หรือบางทีก็เรียก Z Buffer) ใน World Space นี้ เราสามารถทำอะไรกับโมเดลก็ได้ เช่น ย้ายตำแหน่ง (Translation), หมุน (Rotation) และขยายหรือลดขนาด (Scaling) เป็นต้น

2.1.5Camera Space

เป็นมุมมองของกล้อง หรือดวงตา เหมือนกับกรณีที่เรามองไปที่สิ่งต่างๆ รอบๆ ตัว ซึ่งเราจะเห็นเฉพาะสิ่งที่ปรากฏอยู่ในสายตาเท่านั้น และสิ่งที่อยู่ในสายตาเราก็จะเรียกว่า Camera Space ดังนั้นสรุปได้ว่า Camera Space จะเป็นพื้นที่แบบสามมิติที่ดึงมาจากบางส่วนของ World Space

2.1.6 Projection Space

เป็นมุมมองในแบบ Perspective โดยโมเดลสิ่งที่อยู่ใน Camera Space จะถูกปรับขนาด และยืดขยาย คือส่วนที่อยู่ใกล้ตาหรือกล้อง จะมีขนาดใหญ่กว่าส่วนที่อยู่ไกลตาออกไป

2.1.7Screen Space

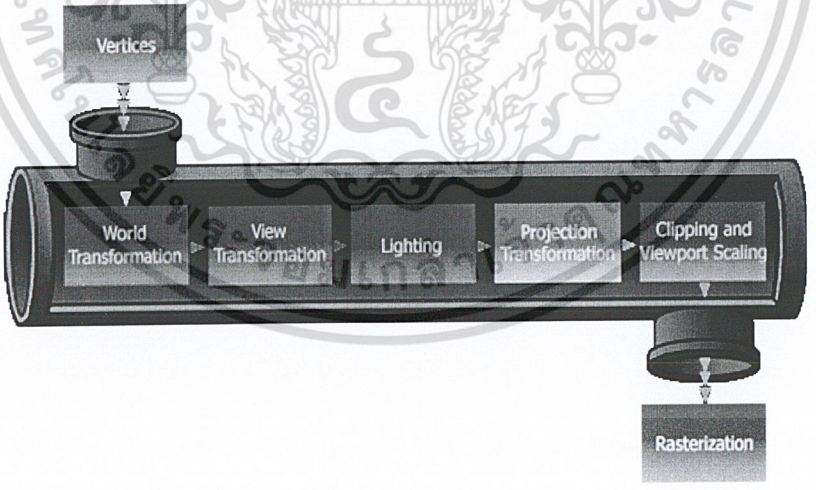
คือพื้นที่สองมิติแบนๆ แบบภาพที่อยู่บนจอภาพ โดยพื้นที่นี้จะถูกแปลงมาจาก Projection Space เพื่อเตรียมเอาไปแสดงให้เห็นบนจอภาพ

2.2 Transformation and Lighting (T&L) และ Rasterization

Transformation คือขั้นตอนในส่วนของการคำนวณเพื่อแปลง Coordinate หรือตำแหน่งของ Vertex ต่างๆ ที่ประกอบขึ้นเป็นโมเดลที่อยู่ใน Model Space ให้เป็น Coordinate ใน World Space (เรียกขั้นตอนนี้ว่า World Transformation) หลังจากนั้นก็ทำขั้นตอนการคำนวณเพื่อแปลง ตำแหน่งของ Vertex ต่างๆ บางส่วนของ World Space ให้ไปเป็น Coordinate ใน Camera Space (เรียกขั้นตอนนี้ว่าViewTransformation)

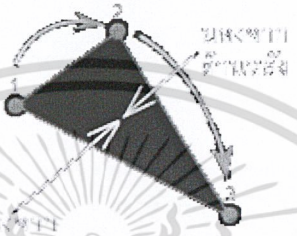
หลังจากนั้นจะเป็นการทำ Lighting ครับ โดยมันจะคำนวณเพื่อใส่สีต่างๆ ให้กับ Vertex เช่นสีจาก Texel ของ Texture ที่นำมา Map, สีของจุด Vertex นั้นๆ เอง หรือสีจากแหล่งกำเนิดแสงต่างๆ เป็นต้น

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



เมื่อทำงานในส่วน Transformation and Lighting เสร็จแล้ว ยังมีการทำงานอันหนึ่งแทรกก่อนจะไปถึงการทำงานในส่วน Rasterization นั่นคือ การตัดส่วนของโมเดลที่ถูกโมเดลอื่นบัง หรือตัดส่วนที่อยู่ด้านหลังออกไปด้วย แล้วก็ลดหรือขยายขนาดโมเดลต่างๆ ให้พอดีกับขนาดของ Resoluion

ของหน้าจอที่ใช้ (เรียกขั้นตอนนี้ว่า Clipping and Viewport Scaling) ซึ่งขั้นตอนส่วน นี้จะช่วยลดการทำงานของขั้นตอนการ Rasterization คือไม่ต้องไปสนใจสิ่งที่มองไม่เห็นนั่นเอง จริงๆ แล้วการ Clipping และ Viewport Scaling นี่จะทำก่อน T&L ก็ได้ แต่ใน Direct3D มันมาทำตอนนี้ และโดย Default แล้ว Direct3D จะถือว่า ด้านหลังคือด้านที่ Vertex เรียงกันแบบทวนเข็มนาฬิกา (D3DRS_CULLMODE เป็น D3DCULL_CCW) ดังนั้น เราจะต้องเรียง Vertex แบบตามเข็มนาฬิกา



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

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

2.3 World Transformation Matrix

Matrix ใน Direct3D จะเป็น Array ขนาด 4 x 4 ใช้สำหรับจัดการกับ Vector ต่างๆ ของ Vertex World Transformation Matrix เป็น Matrix ที่ใช้สำหรับนำโมเดลต่างๆ ไปไว้ใน World Space คือย้ายจาก Model Space ไปสู่ World Space ซึ่งขณะที่ทำการย้ายนั้น เราสามารถหมุน, ย่อ, ขยาย หรือวางโมเดลแต่ละอันไว้ที่ตำแหน่งใดใน World Space ก็ได้ โดยที่เราไม่ต้องไปแตะต้องค่าของตำแหน่งดั่งเดิมของ Vertex ต่างๆ ที่คุณกำหนดไว้ในโมเดลแต่ละอัน

หลักการแปลงจาก Model Space มา World Space ที่ดีก็คือ ทำแต่ละอย่างโมเดลแต่ละอันเอาไว้ก่อน แล้วเก็บค่าต่างๆ ไว้ใน Matrix ที่เราสร้างขึ้นไว้ จากนั้นจึงค่อยรวม Matrix แต่ละอันนั้น ให้เป็น Matrix ผลลัพธ์รวมอันเดียว แล้วจึงสั่งให้ทำกระบวนการ World Transformation ด้วยการใช้ Metrix ผลลัพธ์นี้เป็นหลัก ซึ่งทำได้โดยการเรียกใช้ Method ชื่อ IDirect3DDevice8::SetTransform() โดยส่ง Paremeter ตัวแรกเป็นค่าคงที่ชื่อ D3DTS_WORLD และ Parameter ตัวสุดท้ายเป็น Address ของ Matrix ผลลัพธ์ดังกล่าว

ดังนั้น ขั้นแรกเราต้องสร้าง Matrix สำหรับใช้เก็บผลลัพธ์การหมุนกล่องในแนวแกน x ขึ้นมาก่อน สมมติชื่อว่า matRotationX แล้วจัดการหมุนกล่องในแนวแกน x ไปอยู่ที่ตำแหน่งมุมองศาแบบเรเดียนที่ต้องการด้วยฟังก์ชั่นที่อยู่ใน D3DX library ที่ชื่อ D3DXMatrixRotationX(&matRotationX, มุมแบบ Radian) ค่อไปก็สร้าง Matrix อีกตัวขึ้นมาเพื่อใช้เก็บผลลัพธ์ของการหมุนกล่องในแนวแกน y ชื่อ matRotationY แล้วจัดการหมุนกล่องในแนวแกน y ด้วยฟังก์ชั่น D3DXMatrixRotationY (&mat RotationY , มุมแบบ Radian) จากนั้นก็นำ Matrix ของทั้ง 2 แกนมารวมกันเป็น Matrix ตัวเดียวชื่อ matRotation ดังนี้ D3DXMatrixMultiply(&matRotation, &matRotationY, &matRotationX) คือเป็นการนำ matRotationX เป็นตัวตั้ง (ตัวตั้งจะเป็น Parameter ตัวสุดท้าย) แล้วคูณด้วย matRotationY และผลที่ได้จะเป็นการรวม Matrix แล้วเก็บไว้ใน matRotation ซึ่งการที่เราใช้ matRotationX เป็นตัวตั้งจะทำให้แกนyเปลี่ยนตำแหน่งโดยหมุนอยู่รอบแกนxด้วย(ตัวคูณจะแปรเปลี่ยนไปตามตัวตั้ง) ต่อมาเราก็จะเลื่อนกล่องตามแนวแกน z โดยสร้าง Matrix ขึ้นมาอีกตัวสมมติว่าชื่อ matMoveZ แล้วสั่งให้มันย้ายตำแหน่งด้วย D3DXMatrixTranslation(&matMoveZ, 0.0f, 0.0f, จำนวนค่าในแนวแกน z ที่จะเลื่อนไป) ถ้าค่าติดลบก็จะเลื่อนเข้าหาตัวเรา ถ้าค่าเป็นบวกก็จะเลื่อนเข้าไปในจอ (ตามแบบ Left-Hand Coordinate System) ซึ่งที่ Parameter ตัวที่ 2 และ 3 เป็น 0.0f นั้นจะหมายความว่า ไม่มีการเคลื่อนที่ในแนวแกน x และ y จากนั้นก็รวม Matrix ทั้งหมด คือ ผลลัพธ์จากการหมุนทั้ง 2 แกน กับผลลัพธ์จากการย้ายตำแหน่งในแกน z โดยใช้ method D3DXMatrixMultiply(&matWorld, &mat Rotation, &matMoveZ) ซึ่งจะทำให้การหมุนนั้นเคลื่อนที่แปรเปลี่ยนไปตามแกน z ด้วย และผลลัพธ์ที่ได้จะถูกเก็บไว้ใน Matrix ที่ชื่อ matWorld

ขั้นสุดท้ายจึงนำ Matrix ผลลัพธ์ที่ได้ไปทำให้เป็น World Transformation Matrix ด้วย Method ที่ว่า g_pd3dDevice->SetTransform(D3DTS_WORLD, &matWorld)

2.4 View Transformation Matrix

เป็น Matrix ที่ใช้สร้างกล้อง (Camera) ซึ่งมันจะจับภาพบางส่วนใน World Space หรือกล่าว

ได้ว่าเป็น แปลง Coordinate หรือตำแหน่งของโมเดลต่างๆ ที่อยู่ใน World Space ตรงตำแหน่งที่กล้องจับภาพอยู่ มาเป็น Coordinate ของ Camera Space

หลักการและฟังก์ชั่นที่ใช้ในการหมุน, เคลื่อนย้าย หรือปรับขนาดของมุมมองของกล้องก็เหมือนกับที่ใช้กับWorld Transformation Matrixยกเว้นว่าต้องมีการบอกให้Direct3Dรู้ว่าเราจะใช้3-D Coordinate System แบบ Left-Hand หรือ Right-Hand โดยใช้ฟังก์ชั่น D3DXMatrix LookAt LH() หรือ D3DXMatrixLookAtRH() ซึ่งมี Parameter ที่ต้องส่งไปให้ฟังก์ชั่นนี้มีอยู่ 4 ตัว ตัวแรกเป็น Address ของ Matrix ที่จะใช้เก็บผลลัพธ์ที่ได้ ตัวที่สองเป็น 3-D Vector เพื่อบอกว่า ตำแหน่งของกล้องอยู่ที่ไหน ตัวที่สามจะเป็นตำแหน่งจุดโฟกัสของกล้อง หรือตำแหน่งที่กล้องจับอยู่ ตัวสุดท้ายเป็นตัวบอกว่า กล้องตั้งขึ้นในแนวไหน (ด้านบนของกล้องหันไปที่แกนไหน)

ต่อไปก็สร้าง View Transformation Matrix โดยการเรียกใช้ Method IDirect3DDevice8:: SetTransform() และกำหนดค่าคงที่สำหรับใช้เป็น Parameter ตัวแรกว่า D3DTS_VIEW แล้วส่ง Address ของ Matrix ผลลัพธ์ที่ได้จาก D3DXMatrixLookAtLH() เป็น Parameter ตัวสุดท้าย

2.5 Projection Transformation Matrix

เป็น Matrix ที่ใช้สร้างมุมมองแบบ Perspective จากมุมมองของกล้องหรือ Coordinate ของ Vertex ต่างๆ ที่อยู่ใน Camera Space โดยจะแปลงให้มาอยู่ใน Projection Space ซึ่งวิธีสร้างจะต้องใช้ฟังก์ชั่นแบบ Left-Handหรือ Right-Handคือ D3DXMatrixPerspectiveFovLH() กับ D3DX Matrix PerspectiveFovRH() ตามลำดับ โดย Parameter ตัวแรกที่ส่งให้ฟังก์ชั่นนี้ก็คือ Address ของ Matrix ที่จะใช้เก็บผลลัพธ์ที่ได้ ส่วน Parameter ตัวที่สองจะเป็นมุมแบบเรเดียนซึ่งหมายถึงระยะกางของมุมมองที่จะทำ Perspective หรือที่เรียกว่า Field of View (FOV) ถ้ามุมกว้างมาก จะหมายถึงการดึงภาพกว้างมาก มาบีบให้แสดงในพื้นที่ของหน้าจอ ผลก็คือเกิด Perspective มากเมื่อมองโมเดลแบบใกล้ๆ ปกติมักจะใช้ 45 องศา หรือ 1 / 4 เรเดียน

Parameter ตัวที่ 3 เป็นอัตราส่วนของหน้าจอในแนวนอนกับแนวตั้ง (Aspect Ratio) เช่น ถ้ารันแบบ Full-Screen Mode แล้ว Resolutionหรือขนาดของ Back Buffer ที่สร้างตอนเรียกใช้ Method ชื่อ IDirect3D8::CreateDevice() เป็น 1024 x 768 ก็ให้ระบุ Parameter เป็น 1024 / 768 แต่ถ้ารันแบบ Windowed Mode จะต้องใช้ขนาดของวินโดว์ของโปรแกรมแทน

Parameter ตัวที่ 4 และ 5 เป็น ขอบเขตในแนวแกน z ที่จะแสดง Perspective ซึ่งเป็นระยะใกล้ (Near View-Plane) และไกล (Far View-Plane) ตามลำดับ

เมื่อเรียกใช้ฟังก์ชั่นดังกล่าวเสร็จแล้ว ก็ให้นำ Matrix ผลลัพธ์ที่ได้มาใช้ในการสร้าง Projection

Transformation Matrix โดยการเรียกใช้ Method IDirect3DDevice8::SetTransform() และกำหนดค่าคงที่สำหรับใช้เป็น Parameter ตัวแรกว่า D3DTS_PROJECTION แล้วส่ง Address ของ Matrix ผลลัพธ์ดังกล่าวเป็น Parameter ตัวสุดท้าย

2.6 โปรแกรม Microsoft Direct X

2.6.1 ความเป็นมา

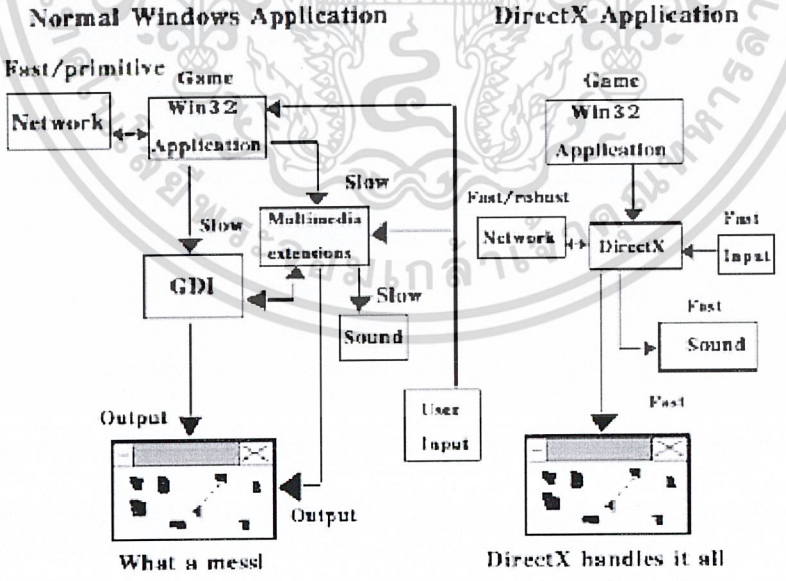
ในสมัยก่อนการเขียนโปรแกรมให้ติดต่อกับอุปกรณ์ต่างๆเป็นเรื่องยุ่งยากโปรแกรมเมอร์จำเป็นที่จะต้องทำการศึกษาถึงคำสั่งต่างๆของ Hardware แต่ละรุ่นแต่ละยี่ห้อซึ่งมีจำนวนมากและเป็นเรื่องปริมาณที่ทำให้เสียเวลาและ เมื่อ Hardware ตัวเก่าล้าสมัยและ Hardware ตัวใหม่ถูกผลิตออกมาโปรแกรมเมอร์ก็จำเป็นต้องทำการศึกษาถึงวิธีการใช้คำสั่งต่างๆของ Hardware ตัวใหม่ซึ่งอาจจะแตกต่างจากตัวเก่าโดยสิ้นเชิงหรือมีการเพิ่มเทคโนโลยีใหม่ๆเข้ามาใน Hardware ซึ่งเป็นเรื่องที่ทำความลำบากแก่ผู้พัฒนาเป็นอย่างมากใน Windows (ตั้งแต่ version 3.1 เป็นต้นมา) มีการใช้ Driverของ Hardwareเพื่อที่จะเข้าถึงอุปกรณ์เป็นเหตุทำให้นักโปรแกรมเมอร์พบกับความลำบากในการเขียนโปรแกรม เพื่อติดต่อกับ Hardwareในการพัฒนาโปรแกรมโปรแกรมเมอร์คาดหวังว่าใน Windowsรุ่นใหม่ๆจะมีชุดคำสั่งที่จะช่วย ให้การเขียนโปรแกรมติดต่อกับอุปกรณ์ สามารถทำผ่าน Interface ตัวหนึ่งเพียงตัวเดียวได้ ซึ่ง Interface ตัวนี้จะทำการ เรียกใช้ Driver ต่างๆของอุปกรณ์นั้นๆเอง ซึ่งจะทำให้การเขียนโปรแกรมเป็นไปอย่างสะดวกเพราะเพียงทำการศึกษา ถึงคำสั่งการเรียกใช้ Interface ก็เพียงพอที่จะติดต่อกับอุปกรณ์ต่างๆรุ่นใดๆ ยี่ห้อใดๆ ได้เลยทันที

บริษัท Microsoft Corporation เล็งเห็นถึงความต้องการของโปรแกรมเมอร์ตรงจุดนี้จึงได้ทำการผลิตชุดพัฒนา SDK สำหรับ Microsoft Windows ขึ้นมา โดยมีจุดประสงค์ ที่จะช่วยให้การเข้าถึง Hardware ต่างๆ เป็นเรื่องง่ายยิ่งขึ้นอีกซึ่งไม่จำเป็นต้องไปทำการศึกษาการทำงานของ Hardware ตัวนั้นๆเหมือนสมัยก่อนโดยระยะแรกมีวัตถุประสงค์ เพื่อใช้พัฒนาเกมคอมพิวเตอร์โดยเฉพาะ จึงได้ตั้งชื่อว่า “ Game kit ” ต่อมาได้เพิ่มความสามารถที่ช่วยทางด้านอื่นๆด้วยเช่น multimedia ดังนั้นจึงได้พัฒนาและถูกเรียกใหม่ว่า Microsoft DirectX และชุดพัฒนาจะมี SDK ต่อ ท้าย โดยปัจจุบันได้ออกมาถึงรุ่นที่ 9.0 แล้ว

2.6.2 **หลักการของ** Direct X

สิ่งที่ Microsoft ได้คำนึงถึงเป็นสิ่งแรกของ Direct X ก็คือจะต้องเป็นชั้นระหว่าง Application กับHardware ที่ต้องมีความสัมพันธ์กันมากเสมือนเป็นชั้นเดียวกันเพราะต้องการให้Application สามารถทำงานกับHardwareได้อย่างรวดเร็ว ความจริงแล้วก่อนที่ Direct X จะถูกพัฒนาขึ้นมาได้มีชุดคำสั่งของ Windows ซึ่งใช้ติดต่อกับการ์ดแสดงผลอยู่ ก่อนแล้วซึ่งเรียกว่า GDI ซึ่งมีลักษณะของการทำงานคล้าย Direct X ซึงก็คือสามารถติดต่อกับอุปกรณ์ Hardware ต่างๆ ได้โดยตรงโดยไม่ต้องทำการเรียกใช้Driverของการ์ดจอเองแต่ความสามารถจะช้ากว่าและความสามารถบางอย่างยังไม่สามารถทดแทนได้ เช่น หากโปรแกรมเมอร์ ทำการเรียกคำสั่งที่ติดต่อกับ Hardware แสดงผลซึ่งมีผลทำให้การทำงานเร็วมากขึ้น เพราะทำงานบน Hardware หากว่าเครื่องของผู้ใช้ Application นั้นไม่มี Hardware ดังกล่าว ถ้าเป็น GDI จะไม่สามารถทำงานหรือ ใช้งาน Application นั้นได้เลย แต่หากโปรแกรมเมอร์เรียกใช้ Hardware ผ่านทาง Direct X เมื่อ Direct X ทำการตรวจสอบหา Hardware ดังกล่าวและหาไม่เจอ ก็จะทำการจำลอง Hardware ตัวนั้นเหมือนกับว่ามีHardware ตัวนั้นอยู่และใช้ Algorithm ที่กำหนดไว้ทำการประมวลผลและทำการแสดงผลแทนHardwareตัวนั้นซึ่งผู้ใช้สามารถเรียกใช้ Software แทน Hardware และยังสามารถใช้งานได้เหมือนเดิม

Direct X คือชุดของ Dynamic Link Library (DLL) ซึ่งสามารถทำให้โปรแกรมเมอร์เข้าถึง Hardware ได้อย่างรวดเร็ว โดยไม่ผ่านทาง Hardware ตามหลักการของ Windows ดังรูปภาพที่ 2.1



รูปที่ 2.1 Direct X **สนับสนุน เกมที่รันบน** Windows **ด้วยประสิทธิภาพที่สูง**

Direct Xยังเป็นมาตราฐานของsoftware และ Hardware ในปัจจุบันโดยบริษัท Software ส่วนมากที่มีอุปกรณ์ที่เกี่ยวข้องกับ Direct X จะเขียนโปรแกรมให้เรียก ใช้ Direct X และรวมไปถึงบริษัทที่ผลิต Hardware ก็ยังสนับสนุน โดย Direct X จะทำการเรียกใช้คำสั่งของ Direct X เพราะฉะนั้นเราสามารถที่จะมั่นใจได้ว่าDirect Xจะเป็นที่แพร่หลายในอนาคตMicrosoft ได้ทำให้Direct X มีคุณสมบัติพิเศษอย่างมากเช่น เกมที่เขียนโดยใช้ Direct X 1.0 จะสามารถ run บนคอมพิวเตอร์ ที่ติดตั้ง Direct X 3.0 หรือ 6.0 และยิ่งไปกว่านั้น Microsoft รู้ว่าเทคโนโลยีDirect X จะต้องหลุดออกจากหลักการ หากว่าถูกเขียนโดยไม่มีการมองการณ์ไกลและการวางแผน ดังนั้นจึงจำเป็นที่จะต้องใช้การใช้การโปรแกรมลักษณะของ Object Oriented ซึ่งสามารถปรับปรุงให้ดีขึ้นได้ และสามารถทำงานได้กับหลายๆภาษา และเป็นลักษณะของ Black box เทคนิคเช่นนี้ถูกเรียกว่าComponent Object Model (COM)

2.6.3 **ส่วนประกอบของ Direct X**

2.6.3.1 HAL (The Hardware Abstraction Layer)

HAL คือ Software ระดับล่างสุดใน Direct X ซึ่งบรรจุ Driver ของ Hardwareจากผู้ผลิต Hardwareรายนั้นๆ เพื่อที่จะทำการควบคุม Hardware ได้โดยตรง Softwareในชั้นนี้ให้ประสิทธิภาพสูงสุดเพราะเป็นชั้นที่ทำการติดต่อกับ Hardware โดยตรง

2.6.3.2 HEL (The Hardware Emulation Layer)

HEL จะวางอยู่ชั้นก่อนหน้า HAL โดยปกติ Direct X ถูกออกแบบมาให้ใช้ Hardware ให้ได้เต็มประสิทธิภาพและด้วยความเร็วสูง หากว่ามี Hardware ชนิดนั้นๆ อยู่ แต่ Direct X ก็ยังสามารถทำงานได้อยู่หากว่าHardware ชนิดนั้นไม่มีอยู่ ตัวอย่างเช่นหากเราต้องการสร้างภาพๆหนึ่งขึ้นมาซึ่งจำเป็นต้องเรียกใช้ Function บน Hardware ชนิดหนึ่ง โดยทำการเรียกผ่าน Direct X ซึ่งถ้ามี Hardware ตัวนั้นอยู่ก็จะทำการเรียกใช้ Function บน Hardware ชนิดนั้นได้โดยตรง แต่หากว่า Hardware ชนิดนั้นไม่มีอยู่ในเครื่อง HEL จะเข้ามาทำงานในส่วนตรงนี้ โดยการจำลองตัวเอง เป็น Hardware ตัวนั้นๆ และทำงานเคียงคู่กับ HAL ซึ่งแน่นอนว่าทำให้การทำงานช้าลง เพราะเป็นการจำลอง Hardware แต่ก็ยังทำให้ Application นั้นทำงานได้ตามปกติได้

2.6.3.3 Direct Draw

อาจจะเรียกได้ว่าส่วนนี้เป็นส่วนที่สำคัญที่สุดของ Direct X เพราะ Direct Draw สามารถที่จะทำการเรียกใช้งาน Video card ด้วยความเร็วสูง โดยที่ Direct Draw รู้จักการ set mode ของจอภาพทุกๆ mode ไม่ว่าจะเป็น high-resolution หรือจะเป็น true color modes แล้วยังสนับสนุนเรื่องการจัดการสี การกำหนดขอบเขตจอภาพ และการทำ Animation

2.6.3.4 Direct 3D

ส่วนประกอบของ Direct X ชนิดนี้เป็นส่วนประกอบส่วนเดียวที่สามารถติดต่อกับอุปกรณ์แสดงผล 3มิติได้ ทำให้สามารถเขียนโปรแกรมประเภท 3 มิติได้ เช่น เกมขับเครื่องบิน Simulation ต่างๆ

2.6.3.5 Direct Sound

ก่อนที่ยังไม่มี Direct X การเขียนโปรแกรมติดต่อกับการ์ดเสียงเป็นเรื่องที่ยากมาก ซึ่งการเขียนโปรแกรมที่อาศัยการ์ดเสียง และสร้างเสียงที่มีคุณภาพในสมัยก่อนมักจะว่าจ้างให้บริษัทที่ทำงานด้านนี้โดยเฉพาะเป็นผู้ออกแบบและทำการ Coding เช่นบริษัทเกมอาจเป็นบริษัทหนึ่ง แล้วจ้างบริษัทที่ทำระบบเสียงให้ทำการเพิ่มเสียงเพลงเข้าไปในตัวเกม เป็นต้น แต่ปัจจุบันเมื่อ Direct Sound ถือกำเนิดขึ้นสถานการณ์เช่นนั้นจึงหมดลงตามยุค เพราะ Direct Sound ลดขั้นตอนการเข้าถึง Sound card และทำงานได้กับ Sound card ทุกชนิด

2.6.3.6 Direct Sound 3D

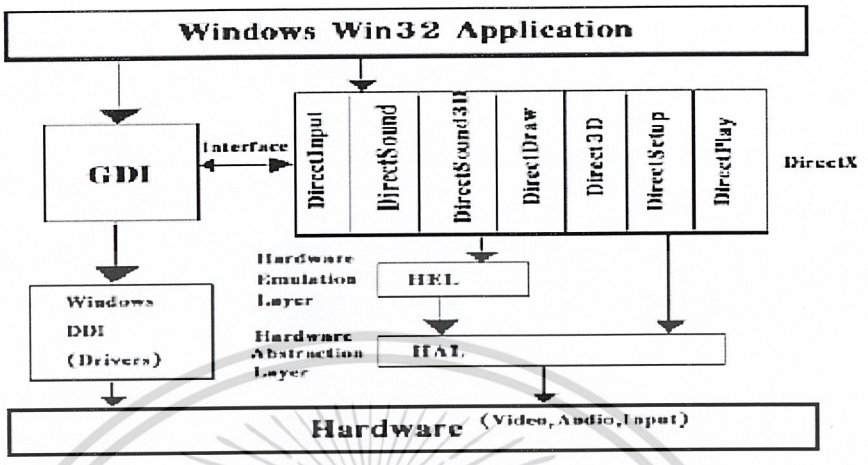
Direct Sound 3D อยู่บนพื้นฐานของ Direct Sound และทำงานได้กับระบบเสียง 3 มิติ ซึ่งจำเป็นต้องมีการคำนวณทางคณิตศาสตร์ Direct 3D จะทำการคำนวณ และพิกัดตำแหน่งของเสียงในระบบ 3 มิติ ซึ่งจะทำให้เสียงที่ออกมาฟังสมจริงมากขึ้น

2.6.3.7 Direct Input

เป็นความสามารถใหม่ที่เพิ่งจะถูกเพิ่มเข้ามาใน Direct X 3.0 ซึ่งจะทำให้การรับข้อมูลทำได้หลากหลายขึ้นและทำได้โดยง่าย โดยรับจาก Keyboard , Mouse และ Joystick ซึ่งใน เวอร์ชั่น 5.0 ยังสนับสนุน Joystick แบบ Force Feed Back (สั่นสะเทือนได้) ในสมัยก่อน Joystick มีปุ่มเพียงเล็กน้อยแต่ในปัจจุบันมีมากมายหลายปุ่ม และมีหลากหลายยี่ห้อ ทำให้การติดต่อกับ Joystick ในแต่ละปุ่มต้องทำการศึกษาถึง Joystick ชนิดนั้นๆ อย่างลึกซึ้ง

2.6.3.8 Direct Play

เป็นส่วนประกอบที่ใช้ติดต่อกับอุปกรณ์ทางด้าน Network การอ้างPort ต่างๆหรือการสร้างเกมที่สามารถเล่น ผ่านเครือข่ายได้



รูปที่ 2.2 Direct X Component

2.7 การเขียนโปรแกรมบน Windows

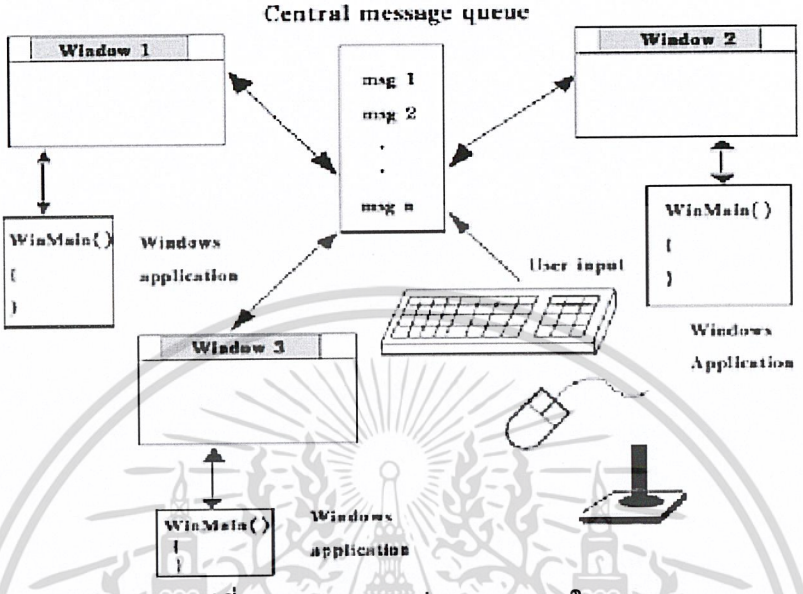
การเขียนโปรแกรม Windows อาจจะเป็นเรื่องที่ง่ายหรือยาก ขึ้นอยู่กับขนาดของโปรแกรมที่ต้องการทำ การเขียน Word processor อาจต้องการความรู้ทางด้านการเขียนโปรแกรมที่มากพอสมควร แต่สำหรับการเขียนเกมโดยใช้ Direct X ต้องการเพียงเล็กน้อยเท่านั้น ในส่วนนี้จะอธิบายถึงการเขียนโปรแกรมบน Windows และพื้นฐานที่ควรจะรู้สำหรับการเขียนโปรแกรมบน Windows เพื่อนำไปใช้ในการเขียนเกม

Windows เป็นระบบปฏิบัติการแบบ event-based โดย Message เป็นเสมือนสัญญาณที่จะบอกให้Windows ต้องทำอะไร ทำอย่างไร หรือเกิดอะไร ตัวอย่าง

Windows event เมื่อผู้ใช้ต้องการเปลี่ยนขนาดของ Window Application Windows ก็จะทำการส่งMessage ไปยัง Application ตัวนั้น ว่าจะเปลี่ยนขนาดอย่างไร

Keyboard event เมื่อผู้ใช้กด แป้นพิมพ์บน keyboard message ของตัวแป้นพิมพ์ที่กด ก็จะถูกส่งไปยัง application เพื่อกระทำการประมวลผลข้อมูลนั้น และแสดงผลข้อมูลนั้นออกมา เช่น แสดงเมนูย่อย จากเมนูบาร์หลัก

Drawing event ถ้า window ตัวใหม่ถูกแสดงออกมาทับตัวเก่าที่แสดงอยู่ หน้าจอจะต้องทำการวาดใหม่อีกครั้ง โดย Repaint message จะกระทำการในส่วนนี้



รูปที่ 2.3 ระบบการส่ง message ใน Windows

เพราะฉะนั้นอาจจะกล่าวได้ว่า ทุกสิ่งทุกอย่างที่เกิดขึ้นใน Windows ก็คือ message นั้นเอง ดังรูปที่2.3 ต่อไปจะกล่าวถึงส่วนประกอบโดยทั่วไป ในการสร้าง windows application ที่ต้องมี ได้แก่

2.7.1 Win32 Libraries

เมื่อเราสร้าง Windows application ขึ้นมา Compiler จะเป็นตัวจัดการเกี่ยวกับ ส่วนของ Librariesให้เองทั้งหมด ยกเว้นเสียแต่ Libraries ของ Application ที่เราต้องการไม่ได้ถูก ตั้งค่าไว้ตั้งแต่เริ่มต้น เช่นเมื่อต้องการสร้าง Application เกี่ยวกับ Multimedia ซึ่งต้องการ Winmm.Lib Library แต่ใน Directoryของ LIB ในตัวของ Compiler ไม่มี หรือไม่ได้ถูกกำหนดไว้ ผู้ใช้ก็ต้องเพิ่ม หรือกำหนดเข้าไปเองเมื่อสร้าง Application ที่ใช้ Direct X ผู้ใช้ก็จะต้องเพิ่ม Libraries ของ direct X เข้าไปในApplication นั้นด้วย

2.7.2 The Include files

โปรแกรมเมอร์ต้อง include files ที่จำเป็นสำหรับ application นั้นไว้ในตัวโปรแกรมด้วยเสมอ และโดยทั่วไป ใน Windows application จะต้องเริ่มต้นด้วย

```
# define win32_LEAN_AND_MEAN
#include <windows.h>
#include<windowsx.h>
```

และอาจจะเพิ่ม include files ที่จำเป็นสำหรับ applicationเข้าไปอีกได้เช่นStdio.hหรือ Math.hเป็นต้น

2.7.3 WinMain

เป็น function หลักที่ Windows application ทุก application จำเป็นต้องกำหนดไว้ คล้ายๆ กับfunction main() ในภาษา C และ ภาษา C++ การประกาศ Prototype ของ WinMain( ) จะเป็นดังนี้

```
int WINAPI WinMain( HINSTANCE hinstance,
HINSTANCE hprevinstance,
LPSTR lpcmline,
Int ncmdshow );
```

2.7.4 Windows class

โปรแกรมเมอร์ที่ต้องการสร้าง Windows application จำเป็นต้อง กำหนด Windows class ขึ้นมาด้วย ซึ่ง windows class จะอธิบายถึงคุณลักษณะโดยทั่วไปของ windows ทั้งหมด โดยหลังจากโปรแกรมเมอร์ทำการสร้าง windows class ขึ้นมาแล้วโปรแกรมเมอร์ ก็จะสามารถสร้าง window ได้ตามwindows class ที่กำหนดขึ้นมาได้ตามต้องการ และต่อไปนี้คือโครงสร้างของ Windows class

```
typedef struct _WNDCLASS
{
UINT style; // style flags
WNDPROC lpfnWndProc; // pointer to event handler
Int cbClsExtra; // extra byte
Int cbWndExtra; //extra byte
HANDLE hInstance; //handle of instance
HICON hIcon; //handle to Icon
HCURSOR hCursor; //handle to cursor
HBRUSH hbrBackground;//handle to background brush
LPCTSTR lpszMenuName; //name of Menu
LPCTSTR lpszClassName;//name of class
} WNDCLASS;
WNDCLASS wndclass // a declaration of a window class
```

2.7.5 Registering the Windows class

หลังจากที่ทำการสร้าง windows class ขึ้นมาแล้วขั้นต่อไปจะทำการ register class การ registerจะทำให้ Windows รู้ว่า มี windows class ที่สร้างขึ้นมา และ สามารถให้โปรแกรมเมอร์สร้าง window ที่มีลักษณะตาม windows class ได้โดยชื่อ ASCII ของ class เพียงอย่างเดียว การ register classกระทำได้โดย

```
if (RegisterClass(&wndclass)==0)
{
//error
} // end if
```

2.7.6 **การสร้าง** Window

การจะสร้างwindowขึ้นมาต้องทำการเรียกfunction CreateWindow ( )ที่มีค่า parameters 11ค่า การประกาศ prototype จะทำได้ดังนี้

```
HWND CreateWIndow (
LPCTSTR lpClassName, //pointer to class name (string)
LPCTSTR lpWindowName, //pointer to window title (string)
DWORD dwStyle, // windows style flags
int x, // ตำแหน่งแนวแกน x ของวินโดวส์
int y, //ตำแหน่งแนวแกน y ของ วินโดวส์
int nWidth, // ความกว้างของวินโดวส์
int nHeight, // ความสูงของวินโดวส์
HWND hWndParent, //handle to parent ( usually NULL )
HMENU hMenu, //handle to menu(usually NULL)
HANDLE hInstance, //handle to application instance
LPVOID lpParam //pointer to startup creation data
(NULL)
);
```

2.7.7 **การแสดง** Window

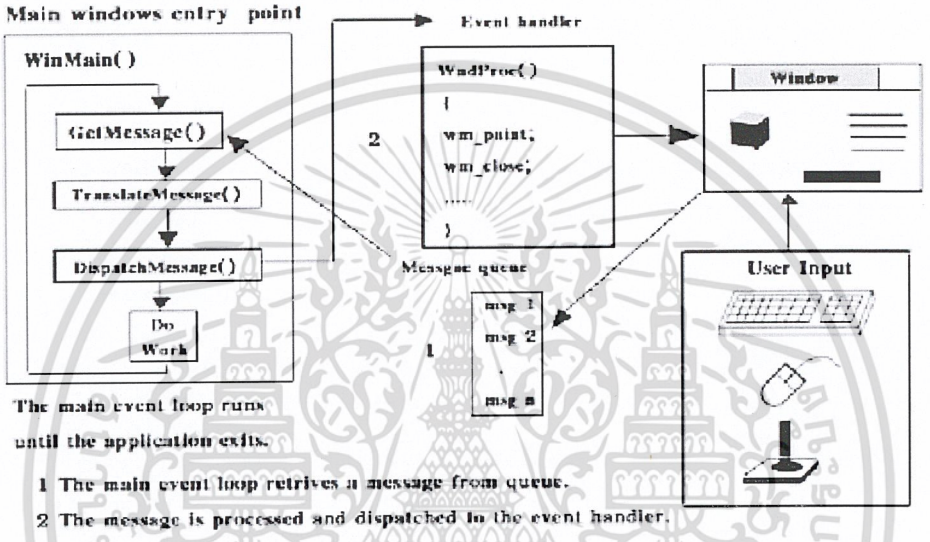
หลังจากที่ทำการ create window เสร็จแล้ว และต้องการ แสดงผลออกมา (ในกรณีที่ ตั้งค่า parameter ของ dwStyle ไม่ใช้ WS_VISIBLE ต้องทำการเรียก function ShowWindow() เสมอ

BOOL ShowWindow(HWND hWnd, // handle of window

Int nCmdshow); // show state of window

2.7.8 Event Loop

main event loop คือ loop ที่คอยรับ message record ที่เก็บไว้ จาก message queue แล้วทำการแปลง แล้วส่งต่อไปยัง event handler ดังรูป 2.4



รูปที่ 2.4 The main event loop

โดยมี Function ที่ทำหน้าที่ต่างกัน อยู่ 3 function ใน event loop นี้ได้แก่

2.7.8.1GetMessage

เป็น function ที่ทำหน้าที่รับ message จาก message queue ถ้ากรณี message queue ว่าง function ก็จะทำการรอจนกว่าจะมี message เข้ามา

BOOL GetMessage(LPMSG lpMsg, //pointer to message structure

HWND hWnd, //handle of window

UINT wMsgFilterMin, //first message

UINT wMsgFilterMax); //last message

2.7.8.2 TranslateMessage

เป็น function ที่ใช้ แปลง message event ให้อยู่ในรูปพื้นฐาน ของ event เพื่อจะนำไปใช้ในส่วนของ event handler

BOOL TranslateMessage(CONST MSG *lpmsg); //pointer to message structure

2.7.8.3 DispatchMessage

ทำการส่ง message หลังจาก ทำการ Translate Message ไปยัง event handler เพื่อทำการประมวลผลต่อไป

LONG DispatchMessage(CONST MSG *lpmsg); //pointer to message structure

2.7.9 Event handler

event handler หรือ WinProc คือ function ที่ต้องเขียนเพื่อ handle ทุก message ที่ application นั้นสนใจ Windows อาจจะส่ง messageมามากมายหลายอย่าง แต่อาจมีเพียง 1 message ที่เราสนใจจะ handle การประกาศต้นแบบ ทำได้ดังนี้

LRESULT CALLBACK WindowProc(HWND hwnd, //the window

UINT msg , //the message itself

WPARAM wParam, //more info on message

LPARAM lParam); //more info on message

2.8 Event Driven Programming

ในการเขียนโปรแกรมบน Windows อย่างแรกที่ควรจะเข้าใจคือ ลักษณะของ Event DrivenProgramming ลักษณะของ event driven คือ โปรแกรมจะทำงานตาม Event ที่เกิดขึ้นโดยระบบปฏิบัติการ จะเป็นตัวคอยบอก Program ว่ามี Event อะไรเกิดขึ้นบ้าง หาก Program ได้รับ event ที่ Programสนใจก็จะทำงานตามที่โปรแกรมเมอร์ได้กำหนดไว้

สำหรับ event ในที่นี้ได้รวมถึงทุกอย่างที่เกิดขึ้น ทั้งการได้รับ Input เช่นการกดปุ่ม การปล่อยปุ่ม การclick mouse เป็นต้น และ event เกี่ยวกับ Windows เช่น การที่ window ได้รับ focus การสั่งปิดwindow หรือแม้แต่การ redraw window ดังที่ได้กล่าวมาแล้ว ดังนั้น Event driven programming ก็คือการเขียนโปรแกรมที่จะทำงานโดยการ check message ที่ได้รับไปเรื่อยๆ แล้วทำงานตามที่กำหนดเมื่อมี message ที่ Program สนใจ

หากเปรียบเทียบกับการเขียนโปรแกรมโดยทั่วไป การทำงานจะทำงานแบบเรียงลำดับตามผู้เขียน โปรแกรมได้ เขียนไว้ แต่สำหรับ event driven programming นั้นคือ ลักษณะที่ผู้ใช้เป็นผู้ควบคุมโปรแกรมมากกว่าที่โปรแกรมจะควบคุมให้ผู้ใช้ทำตาม ยกตัวอย่างเช่น ในการเขียนโปรแกรมบน Dos ถ้าโปรแกรมจะรับ Input 1 key โปรแกรมเมอร์อาจจะใช้ getc() แต่ถ้าเป็น event driven programmingเมื่อมีการกดปุ่ม ระบบปฏิบัติการจะส่ง message บอก program ว่าได้มีการกดแป้นพิมพ์ขึ้น

2.9 วิธีการแสดงภาพ

2.9.1 ความรู้ความเข้าใจเบื้องต้นของการแสดงภาพ

DirectDraw เป็น component ส่วนหนึ่งใน Microsoft DirectX ซึ่งเป็น component ส่วนที่สำคัญที่สุดใน DirectX เพราะว่า DirectDraw มีหน้าที่สำคัญในส่วนที่ใช้วาดหรือติดต่อโดยตรงกับ Hardwareส่วนแสดงภาพ ซึ่งหากขาดส่วน DirectDraw ไป โปรแกรมจะไม่สามารถแสดงภาพได้เลย โดยต่อไปนี้จะกล่าวถึงส่วนที่สำคัญDirectDraw คือ ไลบราลี่ตัวหนึ่งนั่นเอง หรือจะอาจจะเรียกได้ว่าเป็น Include file ในภาษาซี ขึ้นอยู่กับเวอร์ชั่นของ DirectX SDK (Software Delveloper Kit) ซึ่งจะถูกสร้างขึ้นมาให้รองรับภาษาต่างๆ โดยหากเป็นของภาษาซี ก็จะเป็นดังนี้DDRAW.H และ DDRAW.LIB เวลาเรียกใช้ก็ทำการ Include เข้าไปดังนี้

```
#include <ddraw.h>
```

2.9.2 การสร้าง DirectDraw Object

Window และ DirectDraw สามารถทำงานเข้ากันได้เป็นอย่างดี แต่การที่จะนำมาทำงานร่วมกัน จำเป็นต้องมีวิธีการที่มีรูปแบบเฉพาะ โดยการที่เราจะสามารถนำ DirectDraw มาใช้ได้ เราต้องสร้างDirectDraw Object ขึ้นมาเพื่อเป็นส่วนที่ใช้ติดต่อสื่อสารกับ Hardware แต่ก่อนที่เราจะสามารถสร้างDirectDraw Object ได้นั้น เราต้องทำการสร้าง Window application ขึ้นมาเป็น working window เสียก่อน เมื่อมี window ที่ใช้ทำงานแล้ว เราจึงจะสามารถสร้าง DirectDraw Object ได้ โดยใช้ฟังก์ชั่นHRESULT DirectDrawCreate(GUID FAR *lpGUID,LPDIRECTDRAW FAR *lplpDD, IUnknown FAR *pUnkOuter);

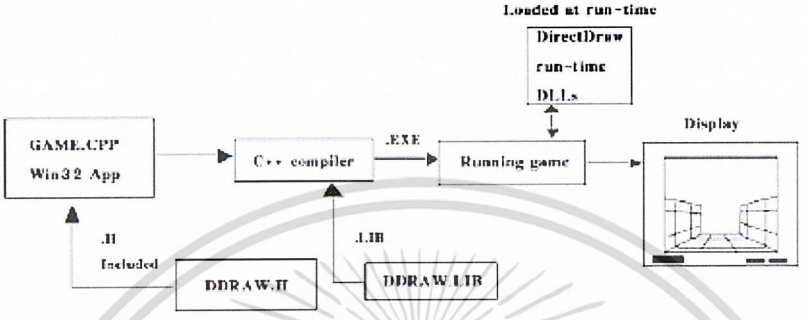
ฟังก์ชั่นนี้จะหมายถึง DirectDrawCreate จะสร้าง DirectDraw Object ไว้ที่ตำแหน่งแอดเดรสlplpDD และจะได้การแสดงผลแบบ lpGUID (ถ้าใส่ค่าเป็น NULL จะได้การแสดงผลแบบ Default) ฟังก์ชั่นนี้จะทำงานสำเร็จหรือไม่นั้น จะแจ้งผ่านค่า HRESULT มาโดย HRESULT เป็นชนิดข้อมูลชนิดหนึ่งของ DirectX ซึ่งใช้สำหรับตรวจสอบการทำงานของฟังก์ชั่น ซึ่งฟังก์ชั่นส่วนใหญ่ของ DirectX จะทำการคืนค่านี้กลับมา เพื่อตรวจสอบค่าที่บ่งบอกถึงความสำเร็จของฟังก์ชั่นก็คือ DD_OK

หากฟังก์ชั่นคืนค่าอื่นๆที่ไม่ใช่ค่านี้ หมายถึงการทำงานของฟังก์ชั่นมีความผิดพลาด ซึ่งจะแจ้งข่าวสารข้อผิดพลาดมาทาง HRESULT เช่นกัน (แทนที่จะเป็น DD_OK)

ตัวอย่าง

```
LPDIRECTDRAW lpdd; // ทำการสร้างตัวแปรที่ใช้เก็บค่าแอดเดรสของ
DirectDraw Object
//ทำการสร้าง Object และ ตรวจสอบข้อผิดพลาด
```

```
if (DirectDrawCreate(NULL,&lpdd,NULL)!=DD_OK)
{
//ทำส่วนนี้เมื่อพบข้อผิดพลาด}
```




```
}
```

รูปที่ 2.5 **เครือข่ายของ** Direct Draw

เมื่อได้ทำการสร้าง Object ของ DirectDraw เรียบร้อยแล้ว วิธีการเรียกใช้ฟังก์ชั่น มีวิธีดังนี้

lpdd-> ฟังก์ชั่น( )

เมื่อเราทำงานกับฟังก์ชั่นนั้นๆเสร็จเรียบร้อยแล้ว เราต้องทำการลบ Object นี้ออกจากหน่วยความจำได้โดย

lpdd->Release( )

2.9.3 **การกำหนดค่า** Cooperation level **ของ** DirectDraw

การที่จะเขียนโปรแกรมเพื่อทำงานบนวินโดวส์ เราจำเป็นจะต้องทำการขอแบ่งทรัพยากรต่างๆ จากสิ่งแวดล้อมบนวินโดวส์ ซึ่ง DirectDraw ก็ได้ให้ฟังก์ชั่นเพื่อใช้ในการกำหนดค่าความสำคัญและกำหนดระบบที่จะใช้ของโปรแกรมของเรา ดังนี้

HRESULT SetCooperativeLevel (HWND hWnd, DWORD dwFlags);

HWND hWnd คือ handle ของวินโดวส์หรือโปรแกรมที่กำลังทำงานด้วย โดยปกติแล้วเราเพียงแต่ทำการกำหนดค่าตัวแปรนี้แล้ว compiler จะทำการสร้างให้เองโดยอัตโนมัติ

DWORD dwFlags คือ ค่าที่จะกำหนดให้กับฟังก์ชั่นโดยมีดังนี้

ค่า Flags	ความหมาย
DDSCL_ALLOWMODEX	อนุญาตให้ใช้ ModeX ได้ (320x200,320x240) โดยค่านี้จะต้องใช้คู่กับ DDSCL_EXCLUSIVE และDDSCL_FULLSCREEN
DDSCL_ALLOWREBOOT	อนุญาตให้ใช้ Ctrl+Alt+Delete ในขณะที่อยู่ใน mode Exclusive (fullscreen)
DDSCL_EXCLUSIVE	ทำการกำหนดให้โปรแกรมเป็นแบบ Exclusive(ให้โปรแกรมมีลำดับความสำคัญสูงสุด) โดยจะต้องทำงานคู่กับ DDSCL_FULLSCREEN
DDSCL_FULLSCREEN	ค่านี้จะบอกให้วินโดวส์ อนุญาตให้เฉพาะDirectDraw เท่านั้นที่เป็นผู้เขียนข้อมูลลงบนหน่วยความจำแสดงผลทั้งหมด
DDSCL_NORMAL	บ่งบอกให้วินโดวส์ทราบว่าโปรแกรมนี้จะเป็นเหมือน Application ทั่วๆไป (เช่น Microsoft Word หรือ ACDSee) โดยหากเรากำหนดค่านี้แล้วเราจะไม่สามารถกำหนดค่า DSCL_ALLOWMODEX, DDSCL_EXCLUSIVE หรือ DDSCL_FULLSCREEN ได้
DDSCL_NOWINDOWCHANGES	บอกว่าโปรแกรมนี้ไม่สามารถที่จะ minimize หรือ restore โปรแกรมอื่นๆในขณะที่โปรแกรมนี้ทำงานอยู่ได้

ตารางที่ 2.1 แสดงค่า Cooperative level ของ DirectDraw

ตัวอย่าง

lpdd -> SetCooperativeLevel(hwnd,DDSCL_NORMAL)

หมายถึงกำหนดให้โปรแกรมเราเป็นแบบ Application ทั่วๆไป

lpdd->SetCooperativeLevel(hwnd,DDSCL_ALLOWMODEX

| DDSCL_FULLSCREEN

| DDSCL_EXCLUSIVE

| DDSCL_ALLOWREBOOT);

หมายถึงโปรแกรมเราสามารถใช้ ModeX ได้ และจะเป็นแบบ FullScreen ซึ่งแสดงผลเต็มหน้าจอและมีความสำคัญสูงสุด และยังสามารถออกจากโปรแกรมได้โดยใช้ปุ่ม Ctrl+Alt+Delete

2.9.4 การกำหนดโหมดการแสดงผล

ความสามารถในการเปลี่ยนโหมดการแสดงผลคือความสามารถที่สำคัญที่สุดของ DirectDraw ฟังก์ชั่นที่ใช้เปลี่ยนโหมดการแสดงผลคือ

```
HRESULT SetDisplayMode(
DWORD dwWidth,
DWORD dwHeight,
DWORD dwBPP,
DWORD dwRefreshRate,
DWORD dwFlags);
```

โดยส่วนที่จำเป็นต้องกำหนดค่า มีสามส่วนด้วยกัน

1. dwWidth กำหนดความกว้างของการแสดงผลโดยมีหน่วยเป็นพิกเซล
2. dwHeight กำหนดความยาวของการแสดงผลโดยมีหน่วยเป็นพิกเซล
3. dwBPP กำหนดจำนวนข้อมูลสีต่อหนึ่งพิกเซล (8,16,32 bit)

นอกนั้นให้กำหนดค่าเป็น 0 ทั้งหมด เพราะ compiler จะทำการตรวจสอบค่าที่เหมาะสมให้เองทั้งหมด ที่กล่าวมานี้เป็นส่วนที่ใช้สำหรับเริ่มการใช้งาน DirectDraw โดยสรุปได้ดังนี้

การเริ่มการใช้งาน DirectDraw มีสามขั้นตอน ดังนี้

- ทำการสร้าง DirectDraw Object
- ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากรโดย SetCoopeeativeLevel
- ทำการกำหนดขนาดการแสดงผล โดยใช้ SetDisplayMode

ตัวอย่าง

```
LPDIRECTDRAW lpdd;
if (DirectDrawCreate(NULL,&lpdd,NULL)!=DD_OK)
{ //ERROR //}
lpdd->SetCooperativeLevel(hwnd,DDSCL_ALLOWMODEX |
DDSCL_FULLSCREEN | DDSCL_EXCLUSIVE | DDSCL_ALLOWREBOOT);
if ((lpdd->SetDisplayMode(640,480,8))!=DD_OK)
{ //ERROR}
```

2.9.5 การประยุกต์ใช้ DirectDraw

DirectDraw สามารถนำมาใช้ได้ในหลายๆด้าน แต่ด้านที่ถือว่านิยมที่สุดและนำ DirectDraw มาใช้มากที่สุดก็คือด้านการสร้างเกมส์คอมพิวเตอร์ ในปัจจุบัน เกมส์ส่วนมากนำ DirectDraw มาใช้ในการสร้าง Animation ซึ่ง DirectDraw สามารถเข้าถึง Hardware ด้านการแสดงผลได้โดยตรง ดังนั้นการทำงานต่างๆจึงรวดเร็วมาก และผู้ใช้ไม่จำเป็นต้องเข้าไปยุ่งเกี่ยวกับคำสั่งในระดับ Hardware เลย

2.9.6 การวาดภาพด้วย DirectDraw

วิธีที่ใช้ในการวาดภาพ ก็คือการนำภาพที่เราได้วาดไว้แล้วจากโปรแกรมวาดภาพต่างๆ เช่น Adobellustrator หรือจะ Adobe Photoshop หรือเราอาจนำภาพมาจากโปรแกรมสร้างภาพสามมิติ เช่น3Dstudio ก็ได้ แต่ภาพจะต้องเป็นประเภท BMP โดยเราจะนำภาพที่เราได้สร้างไว้แล้วเก็บไว้ในหน่วยความจำ หรือที่เรียกว่า Surface แล้วทำการถ่ายเทข้อมูลไปยังหน่วยความจำที่การ์ดแสดงผล หลังจากนั้น จึงนำแสดงขึ้นจอภาพ โดยรายละเอียดของหลักการจะกล่าวในหัวข้อถัดไป

DirectDraw นิยามหน่วยความจำที่ไว้ใช้แสดงผลด้วยคำว่า Surface โดยเราสามารถนำภาพต่างๆเก็บไว้ ณ ที่นี้ได้ ไม่ว่าจะเป็นภาพสองมิติ หรือภาพสามมิติก็ตาม ตามปกติเราต้องมีอย่างน้อย 1 Surface คือ Primary Surface ซึ่งเป็น Surface ที่เรามองเห็นขณะนั้นๆบนจอภาพ โดยเราสามารถมี Surface ได้มากแค่ไหนก็ได้โดยขึ้นอยู่กับหน่วยความจำที่มีอยู่

2.9.7 การสร้าง Surface

หลักการสร้าง Surface มีดังต่อไปนี้

2.9.7.1 Surface ที่สร้างนั้นมีขนาดเท่าใดก็ได้ ขึ้นอยู่กับภาพที่เราจะนำมาใส่ แต่ Surface ที่เป็น Primary Surface จะต้องมีขนาดเท่ากับขนาดของความละเอียดของจอภาพขณะนั้น เช่นหากเรากำหนดให้โปรแกรมเรามีความละเอียดระดับ 800x600 เราก็ต้องสร้างให้ Primary Surface มีขนาด 800x600 ด้วยเช่นกัน

2.9.7.2 การสร้าง Surface เราสามารถเลือกว่าจะใช้หน่วยความจำที่การ์ดแสดงผล (VRAM) หรือจะใช้หน่วยความจำหลัก (System Memory) ซึ่งถ้าเราสามารถใช้ VRAM ได้จะทำให้การแสดงผลมีความเร็วสูงขึ้น แต่ VRAM มีขนาดที่ไม่ใหญ่นัก ดังนั้นหากเราไม่สามารถใช้ VRAM ได้อีก เราสามารถไปใช้ System Memory แทนได้

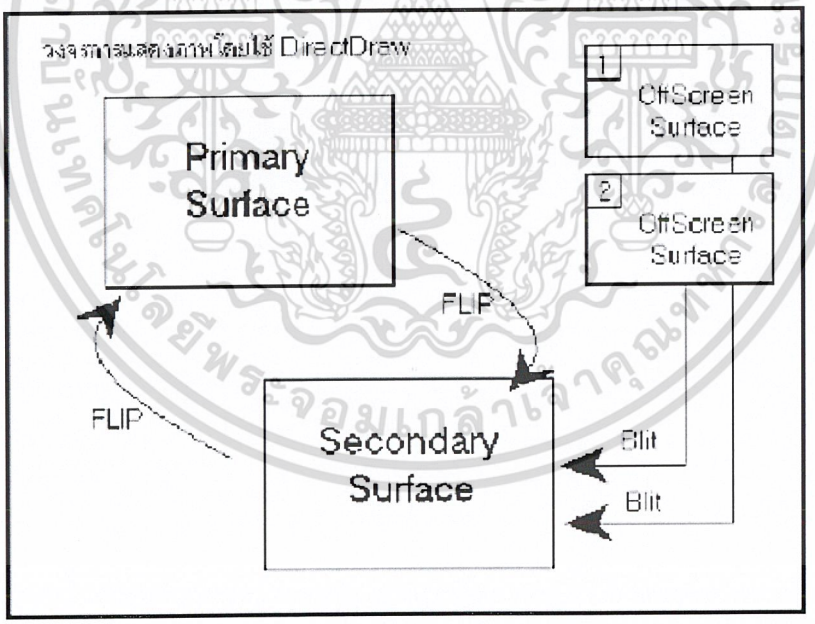
2.9.7.3 เมื่อเราสร้าง Surface ขึ้นมา ระดับความลึกของสีของ Surface นี้จำเป็นต้องเข้ากันได้กับระดับสีของ Primary Surface แต่ตามปกติเมื่อเราสร้าง Surface ขึ้นมาด้วย DirectDraw Objectตัวเดียวกัน ระดับสีของแต่ละ Surface จะเป็น ระดับเดียวกันหมดอยู่แล้ว แต่หากเรามี DirectDrawObject สองตัว การที่จะมีระดับสีของแต่ละ Surfaceไม่เท่ากันก็เป็น ไปได้ และเราก็ไม่

สามารถทำการถ่ายข้อมูลระหว่าง Surface ที่มีระดับสีต่างกันได้โดยทั่วไปเกมส์ที่สร้างขึ้นจะใช้ Surface ดังนี้

1. Primary Surface หรือ FrontBuffer ใช้แสดงภาพที่เป็นหน้าจอปัจจุบัน โดยมีโครงสร้างและขนาดเท่ากับความละเอียดของหน้าจอ

2. Secondary Surface ใช้สำหรับวาดภาพหน้าจอต่อไปขณะที่ Primary Surface กำลังแสดงผลหน้าจอปัจจุบัน หรืออาจเรียกว่า BackBuffer โดย Surface นี้มีโครงสร้างทุกอย่างเหมือน Primary Surface เพื่อที่จะทำงานสลับกับ Primary Surface

3. OffScreen Surface เป็น Surface ที่เราใช้สำหรับเก็บภาพต่างๆที่ใช้ในโปรแกรม ไม่ว่าจะเป็นภาพสำหรับทำภาพเคลื่อนไหว (Animation) หรือภาพนิ่งก็ตาม เพื่อที่จะเพิ่มประสิทธิภาพเมื่อต้องการจะแสดงผลจะได้ไม่ต้องไปอ่านภาพจากไฟล์รูปภาพ ซึ่งหากมีภาพหลายภาพ จะทำให้เสียเวลาในการเข้าถึงข้อมูลในฮาร์ดดิสก์ แต่เราจะเก็บไว้ในหน่วยความจำแทน ทำให้เข้าถึงได้เร็วกว่า ส่วนมากแล้ว เกมจะมีจำนวน Surface นี้มากที่สุด เพื่อใช้เก็บรูปภาพต่างๆ ลักษณะของการแสดงภาพจะเป็นวัฏจักรดังนี้



รูปที่ 2.6 การแสดงภาพโดย DirectDraw

เริ่มจากเรานำภาพที่เราต้องการใช้งานเก็บไว้ใน OffScreenSurface โดยอาจมีได้หลาย OffscreenSurface ต่อ ไปจึงนำภาพที่เก็บไว้ใน Offscreen ทำการ "Blit" (Bit Block Transfer) ซึ่งก็คือการถ่ายข้อมูลจากใน Offscreen ไปยัง Secondary Surface ณ ตำแหน่งที่อยากให้แสดงผลโดยอ้างอิงจากจอภาพเมื่อทำการจัดวางภาพต่างๆอย่างดีแล้วก็พร้อม ที่จะแสดงผลต่อผู้ใช้ ซึ่งเราจะใช้วิธี "FLIP"หรือการสลับหน้าจอกับ PrimarySurfaceโดยจะทำการสลับเฉพาะตำแหน่ง ในหน่วยความจำ จากนั้น จึงทำการวาดภาพซึ่งจะใช้แสดงผลต่อไปใน Secondary Surface ในตำแหน่งที่เพิ่งสลับมา และจะเป็นเช่นนี้ไปจนกว่าจะจบโปรแกรม

การสร้าง Surface จะต้องทำการใส่ข้อมูลลงในโครงสร้างข้อมูลแบบ DDSURFACEDESC แล้วจึงทำการเรียกฟังก์ชั่น CreateSurface( ) เพื่อทำการสร้าง Surface นั้นๆ โดยฟังก์ชั่น CreateSurface ( ) มีการรับค่าต่างๆดังต่อไปนี้

```
HRESULT CreateSurface(LPDDSURFACEDESC lpDDSurfaceDesc,
LPDIRECTDRAWSURFACE FAR *lplpDDSurface,
Iunknown FAR *pUnkOuter);
```

1. lpDDSurfaceDesc เป็นข้อมูลประเภทพอยเตอร์ชี้ไปยังโครงสร้างข้อมูล DDSURFACEDESC ซึ่งมีการใส่ข้อมูลลงไปเรียบร้อยแล้ว
2. lplpDDSurface ถ้าฟังก์ชั่นทำการสร้าง Surface ได้สำเร็จ จะคืนค่านี้กลับมาเป็นพอยเตอร์ ซึ่งชี้ไปยัง Surface ที่สร้างขึ้น
3. pUnkOuter เป็นพารามิเตอร์ขั้นสูง โดยปกติจะกำหนดให้เป็น NULLดังเช่นฟังก์ชั่นอื่นๆ หากฟังก์ชั่นนี้ทำงานสำเร็จจะคืนค่า DD_OK กลับมา แต่ก่อนที่เราจะทำการใช้ฟังก์ชั่น CreateSurface( ) ได้ เราจำเป็นต้องทำการใส่ค่าลงไปยังโครงสร้างข้อมูลที่ชื่อDDSURFACEDESC (Direct Draw Surface Description) ก่อน โดยมีรายละเอียดของโครงสร้างดังนี้

```
typedef struct _DDSURFACEDESC
{
DWORD dwSize;
DWORD dwFlags;
DWORD dwHeight;
DWORD dwWidth;
Union
{
```

```
LONG lPitch;
DWORD dwLinearSize;
};
DWORD dwBackBufferCount;
union
{
DWORD dwMipMapCount;
DWORD dwZBufferBitDepth;
DWORD dwRefreshRate;
};
DWORD dwAlphaBitDepth;
DWORD dwReserved;
LPVOIDlpSurface;
DDCOLORKEY ddckCKDestOverlay;
DDCOLORKEY ddckCKDestblt;
DDCOLORKEY ddckCKSrcOverlay;
DDCOLORKEY ddckCKSrcBlt;
DDPIXELFORMAT ddpfPixelFormat;
DDSCAPS ddsCaps;
} DDSURFACEDESC;
```

จะเห็นได้ว่าเป็นโครงสร้างข้อมูลที่ใหญ่ แต่เราไม่จำเป็นต้องกำหนดทุกค่า มีเพียงบางค่าเท่านั้นที่จะถูกใส่ค่า ลงไป ดังนี้

- dwSize คือค่าขนาดของโครงสร้าง DDSURFACEDESC
- dwFlags คือส่วนที่ใช้กำหนดให้ CreateSurface ทราบว่าจะสร้าง Surface แบบใดซึ่งมีหลายค่า ดังนี้

DDSD_ALL คือการกำหนดให้สามารถเรียกใช้การทำงานได้ทุกๆส่วนของ Surface

หากเรากำหนด Flag ค่าไหน การทำงานในส่วนนั้นจะสามารถเรียกใช้งานได้ แต่ถ้าเราเรียก DDSD_ALL เพียงตัวเดียว ทุกตัวจะสามารถทำงานได้ (แต่เราใช้งานจริงๆเพียงบางตัวเท่านั้น) เราจะเพียงบอกให้ทราบว่ามี Flag อะไรบ้าง แต่เราจะอธิบายเพียงบาง Flag ที่ใช้งาน

DDSD_ALPHABITDEPTH , DDSD_BACKBUFFERCOUNT , DDSD_CAPS,
DDSD_CKDESTBLT , DDSD_CKDESTOVERLAY , DDSD_CKSRCBLT,
DDSD_ CKSRCOVERLAY, DDSD_HEIGHT , DDSD_LINEARSIZE ,
DDSD_LPSURFACE , DDSD_MIPMAPCOUNT , DDSD_PITCH,
DDSD_PIXELFORMAT, DDSD_REFRESHRATE, DDSD_WIDTH,
DDSD_ZBUFFERBITDEPTH

-dwBackBufferCount คือส่วนที่ใช้กำหนดว่าเราจะมี BackBuffer กี่อัน (เราสามารถมีBackbuffer ได้มากกว่า 1 Backbuffer แต่จะทำให้เปลืองหน่วยความจำไปอีก โดยทั่วไปมีเพียงหนึ่งSurface ก็เพียงพอแล้ว)-ddsCaps คือส่วนที่ใช้กำหนดคุณสมบัติของ Surface โดยมีโครงสร้างดังนี้

typedef struct _DDSCAPS
{
DWORD dwCaps;
} DDSCAPS.FAR* LPDDSCAPS;

-dwcaps เป็นส่วนที่ใช้เก็บค่าคุณสมบัติของ Surface ที่เราต้องการให้มี โดยส่วนใหญ่จะใช้ในระดับสูง แต่เราจะยกขึ้นมาเพียงเท่าที่จำเป็นต้องใช้

ค่า Flags	ความหมาย
DDSCAPS_BACKBUFFER	Surface ถูกกำหนดให้เป็น BackBuffer
DDSCAPS_COMPLEX	บอก DirectDrawSurface ว่าเราจะใช้หลาย Surface มากกว่า Primary Surface อันเดียว
DDSCAPS_FLIP	บอกว่า Surface นี้สามารถ Filp ได้
DDSCAPS_FRONTBUFFER	บอกว่า Surface นี้คือ frontbuffer
DDSCAPS_MODEX	กำหนดให้ Surface เป็น ModeX หรือ 320x200 หรือ 320x240
DDSCAPS_OFFSCREENPLAIN	กำหนดให้ Surface เป็น Offscreen ซึ่งไม่ใช้ Frontbuffer, Backbuffer หรือSurface ประเภทอื่นๆ โดยปกติจะใช้สำหรับเก็บรูปภาพที่ใช้แสดงภาพเคลื่อนไหวต่างๆ
DDSCAPS_SYSTEMMEMORY	กำหนดให้ Surface นี้จองหน่วยความจำในหน่วยความจำหลัก

ตารางที่ 2.2 Surface Capabilities Flags

ตัวอย่าง ตั้งแต่เริ่มจนทำการสร้าง PrimarySurface และ SecondarySurface

```
LPDIRECTDRAW lpdd; // ประกาศตัวแปรพ๊อยเตอร์ของ DirectDraw
DDSURFACEDESC ddsd; // ประกาศตัวแปรที่ใช้เก็บค่าของโครงสร้างข้อมูล
//DDSURFACEDESC
LPDIRECTDRAWSURFACE lpddsprimary; //เมื่อ Surface ถูกสร้างขึ้นจะทำการเก็บ
//address ไว้ที่ตัวแปรนี้
//ทำการสร้าง DirectDraw Object
DirectDrawCreate(NULL,&lpdd,NULL);
//ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากร
lpdd->SetCooperatiiveLevel(hwnd, DDSCL_ALLOWREBOOT |
DDSCL_ALLOWMODEX |
DDSCL_FULLSCREEN | DDSCL_EXCLUSIVE);
//ทำการกำหนดขนาดการแสดงผล
lpdd->SetDisplayMode(SCREEN_WIDTH,SCREEN_HEIGHT,SCREEN_BPP);
//สร้าง Primary Surface
//เริ่มต้นโดยทำการกำหนด size
ddsd.dwSize = sizeof(ddsd);
//หลังจากนั้นจึงกำหนด Flag
ddsd.dwFlags = DDSD_CAPS | DDSD_BACKBUFFERCOUNT);
//เมื่อ DDSD_CAPS สามารถเรียกใช้งานได้หมายถึงสามารถกำหนด Primary Surface ได้แล้ว
ddsd.dwCaps.dwCaps = DDSCAPS_PRIMARYSURFACE | DDSCAPS_FLIP |
DDSCAPS_COMPLEX;
ddsd.dwBackBufferCount = 1; //มี BackBuffer 1 อัน
lpdd->CreateSurface(&ddsd,&lpddsprimary,NULL); //สร้าง Primary Surface
ddscaps.dwCaps = DDSCAPS_BACKBUFFER; //ทำการกำหนด BackBuffer
//กำหนดให้ lpddsback เป็น Backbuffer ซึ่งใช้ทำงานร่วมกับ lpddsprimary
lpddsprimary->GetAttachedSurface(&ddscaps,&lpddsback);
```

2.9.8 การนำภาพขึ้นแสดง

เมื่อเราทำการวาดภาพลงบน BackBuffer เรียบร้อยแล้ว จึงทำการ Flip สลับกับ FrontBuffer หรือ PrimarySurface โดยใช้คำสั่ง Flip()

HRESULT Flip(LPDIRECTDRAWSURFACE lpDDSurfaceOverride, DWORD dwFlags);

lpDDSurfaceOverride โดยปกติจะกำหนดให้เป็น NULL

dwFlags โดยปกติจะกำหนดให้เป็น DDFLIP_WAIT เพราะเนื่องจากบางครั้ง

SecondarySurface ยังอยู่ในสถานะที่กำลังวาดอยู่ และยังไม่

สามารถ Flip ได้ โดยค่านี้จะทำให้ฟังก์ชั่นทำการลอง Flip ไปจนกว่าจะ Flip ได้วิธีใช้เป็นดังนี้

lpddsprimary->Flip(NULL,DDFLIP_WAIT);

2.9.9 การวาดภาพลงบน Surface และการทำ Transparency

แต่ก่อนอื่นที่เราจะทำการ Flip เราจะต้องทำการวาดลงบน Surface ให้ได้ก่อน โดยเราสามารถใช้ฟังก์ชั่น Blt ในการคัดลอกข้อมูล โดยเราสามารถคัดลอกข้อมูลจาก Surface ใดไป Surface ใดก็ได้โดยมีโครงสร้างของฟังก์ชั่นดังนี้

HRESULT Blt(LPRECT lpDestRect,LPDIRECTDRAWSURFACE lpDDSrcSurf,

DWORD dwFlags, LPDDBLTFX lpDDBltFx);

lpDestRect คือพ๊อยเตอร์ที่ชี้ไปยังตัวแปรประเภท RECT ของ Surfaceจุดหมายซึ่งตัวแปรประเภท RECTนี้มีโครงสร้างดังนี้

typedef struct _RECT

{

LONG top; //ตำแหน่ง พิกเซล ที่อยู่บนสุดของรูปภาพที่ต้องการวางหรือตำแหน่งที่ต้องการวาง

LONG left; //ตำแหน่ง พิกเซล ซ้ายสุด

LONG right; //ตำแหน่ง พิกเซล ขวาสุด

LONG bottom; //ตำแหน่ง พิกเซล ล่างสุด

}RECT

จากโครงสร้างนี้แสดงให้เห็นว่า lpDestRect ก็คือตำแหน่งที่เราจะวางรูปภาพลงไปนั่นเอง แต่หากค่านี้กำหนดเป็น NULL จะหมายถึงใช้ Surfaceปลายทางทั้ง SurfacelpDDSrcSurf คือ Surface ที่เป็นรูปภาพต้นทางซึ่งอาจเป็น OffScreen ก็ได้

lpSrcRect คือตำแหน่งของรูปภาพที่ต้นทาง หากกำหนดให้เป็น NULL จะหมายถึงใช้รูปทั้งรูป

dwFlags เป็นส่วนที่ใช้ควบคุมการวาดภาพ

lpDDBltFx คือ พ๊อยเตอร์ที่ใช้ชี้ไปที่โครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนดความสามารถพิเศษต่างๆ ของฟังก์ชั่น (เช่นการหมุนภาพ หรือระบายสี)

dwFlags มีค่าดังนี้

ค่า Flags	ความหมาย
DDBLT_COLORFILL	ใช้ร่วมกับ dwFillColor ในโครงสร้างข้อมูล DDBLTFX โดยใส่เป็นสีรูปแบบ RGB (Red,Green,Blue) ซึ่งจะทำการระบายลงบน RECT ที่ surface ปลายทาง
DDBLT_DDFX	ใช้ร่วมกับ dwDDFX ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนด ลักษณะพิเศษบางอย่างในการแสดงผล
DDBLT_DDROPS	ใช้ร่วมกับ dwDDROPS ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX โดยใช.กับการทำงานชั้นสูง
DDBLT_KEYDEST	เมื่อกำหนดแล้วจะสามารถกำหนด color key ที่ surface ปลายทางได้
DDBLT_KEYSRC	เมื่อกำหนดแล้วจะสามารถกำหนด color key ที่ surface ต้นทางได้
DDBLT_ROP	ใช้ร่วมกับ dwROP โดยกำหนดในโครงสร้างข้อมูล DDBLTFX โดยทำงานตรงกันข้ามกับ DDBLT_DDROPS
DDBLT_ROTATIONANGLE	ใช้ร่วมกับ dwRotationAngle ในโครงสร้างข้อมูล DDBLTFX โดยใช้หมุน ภาพที่จะ blit ไปใส่ที่ surface (มีหน่วยเป็น 1/100 องศา)
DDBLT_WAIT	สั่งใช้ทำการ blit จนกว่าจะ blit สำเร็จเมื่อใช้คำสั่งนี้ฟังก์ชั่น blit จะไม่ส่งสารข้อผิดพลาดกลับมาถึงแม้จะยังทำการblit ไม่ได้ก็ตาม

ตารางที่ 2.3 แสดงค่า Control flags ของ Blit( )

ในโครงสร้างข้อมูล DDBLTFX มีข้อมูลจำนวนมากต้องกำหนด แต่ข้อมูลส่วนใหญ่จะใช้สำหรับการทำงานขั้นสูง เช่น Z-buffer (ใช้กับภาพสามมิติ) ดังนั้นเราจะกล่าวถึงเพียงเท่าที่จำเป็นเท่านั้น

```
typedef struct _DDBLTFX
{
```

```
DWORD dwSize;
DWORD dwDDFX;
DWORD dwROP;
DWORD dwDDROP
DWORD dwRotationAngle;
DWORD dwZBufferOpCode;
DWORD dwZbufferLow;
DWORD dwZbufferHigh;
DWORD dwZBufferBaseDest;
DWORD dwZDestConstBitDepth;
union
{
DWORD dwZDestConst;
LPDIRECTDRAWSURFACE lpDDSZBufferDest;
};
DWORD dwZSrcConstBitDepth;
union
{
DWORD dwzSrcConst;
LPDIRECTDRAWSURFACE lpDDSZBufferSrc;
};
DWORD dwAlphaEdgeBlendBitDepth;
DWORD dwAlphaEdgeBlend;
DWORD dwReserved;
DWORD dwAlphaDestConstBitDepth;
union
{
DWORD dwAlphaDestConst;
LPDIRECTDRAWSURFACE lpDDSAlphaDest;
```

```
};
DWORD dwAlphaSrcConstBitDepth;
union
{
DWORD dwAlphaSrcConst;
LPDIRECTDRAWSURFACE lpDDSAlphaSrc;
};
union
{
DWORD dwFillColor;
DWORD dwFillDepth;
DWORD dwFillPixel;
LPDIRECTDRAWSURFACE lpDDSPattern;
};
DDCOLORKEY ddckDestColorkey;
DDCOLORKEY ddckSrcColorkey;
} DDBLTFX.FAR* LPDDBLTFX;
```

ทั้งหมดนี้คือโครงสร้างข้อมูล DDBLTFX แต่เราจะกำหนดใช้เพียงส่วนน้อยเท่านั้น

dwSize ขนาดของโครงสร้างข้อมูล DDBLTFX นี้ โดยกำหนดเป็นไบท์

dwFillColor สีที่ต้องการระบายลง surface

ddckDestColorkey เป็น color key ของทาง surface ปลายทาง

ddckSrcColorkey เป็น color key ของทาง surface ต้นทาง

dwRotationAngle กำหนดเป็นองศาเพื่อจะใช้หมุนภาพ

การใช้ฟิลด์ที่เกี่ยวข้องกับ color key จำเป็นจะต้องทราบความหมายของ color key เสียก่อน โดยcolor key ก็คือ สีที่เรากำหนดให้เป็นสีโปร่งใส (Transparency) โดยจะใช้มากในเกมส์คอมพิวเตอร์ทั่วๆไป โดยปกติ รูปภาพที่สร้างขึ้นจะมีขอบสี่เหลี่ยม ซึ่งเป็นส่วนเกินเมื่อนำภาพไปสร้างเกมส์ เราจำเป็นต้องตัดส่วนเกินนี้ออก โดยการกำหนดสีที่เราจะใช้เป็น color key แล้วเราก็ใช้สีเดียวกันนี้ระบายลงไปที่ขอบหรือส่วนเกินของภาพ เมื่อทำการ Blit แล้วคอมไพเลอร์จะทำการจับคู่สีที่เป็น color key แล้วจะไม่แสดงสีนั้นลงบนจอภาพ

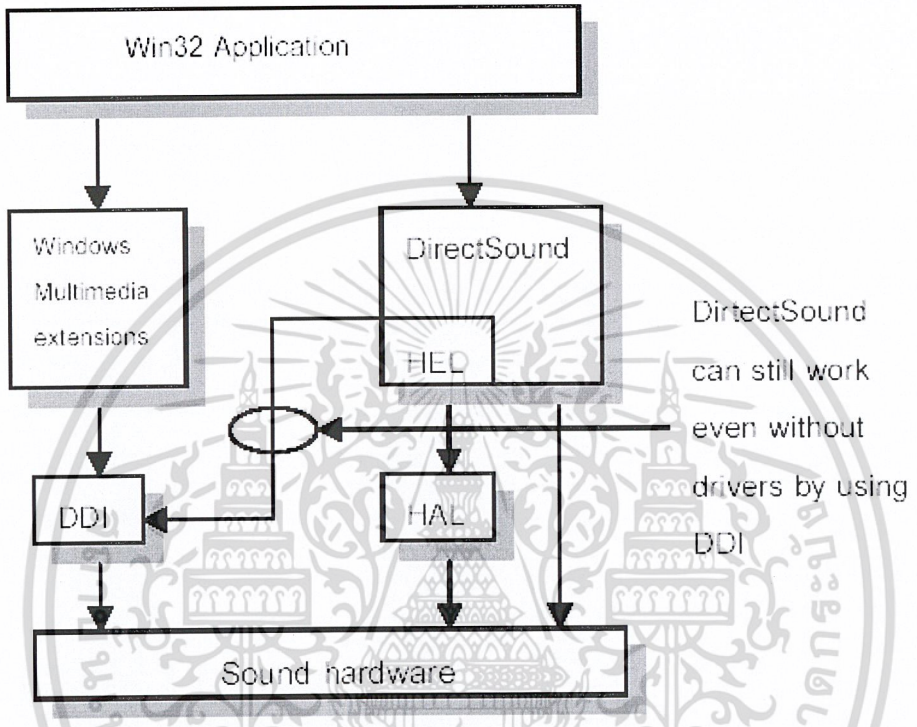
ตัวอย่าง การ blit โดยใช้ color key เป็นสีดำ

```
DDBLTFX ddbltfx; //ประกาศโครงสร้างข้อมูล ddbltfx
RECT blit_area,src_area; //ประกาศ RECT ที่จะวางภาพปลายทาง และภาพต้นทาง
memset(&ddbltfx,0,sizeof(DDBLTFX));
ddbltfx.dwSize=sizeof(DDBLTFX);
DDCOLORKEY col_key; //กำหนดตัวแปรแบบ DDCOLORKEY
col_key.dwColorSpaceLowValue = 0; // ทำการ set ค่าสีจากสีหนึ่ง (ต้องมีค่าสีต่ำกว่า)
col_key.dwColorSpaceHighValue = 0; //ไปยังอีกสีหนึ่ง ทำให้เราสามารถมี transparent
// ได้หลายสี
lpddsback->SetColorKey(DDCKEY_SRCBLT,&col_key); //ทำการ setcolorkey โดยเป็น Source
Colorkey ซึ่งก็คือ สี //ที่ไม่ต้องการ ณ ภาพต้นทางโดยต้องทำการกำหนดที่BackBufferเพื่อบอกให้
BackBufferทราบว่าจะไม่สนใจสีนี้และ //ไม่ทำการ blit ลงไป แต่หากเป็น Destination Colorkey จะ
หมายถึงว่าเป็นสีที่จะไม่ให้วาดทับแทน
blit_area.top = top;
blit_area.left = left;
blit_area.bottom = bottom;
blit_area.right = right;
src_area.top=top_src;
src_area.left=left_src;
src_area.bottom=bottom_src;
src_area.right=right_src;
lpddsprimary->Blt(&blit_area,lpddsback,,&src_area,DDBLT_KEYSRC | DDBLT_WAIT,
NULL);
```

การใช้DirectDrawเราสามารถนำไปใช้ได้ หลากหลายงานไม่ เฉพาะเจาะจงว่าต้องเป็นเกมส์คอมพิวเตอร์ อยู่ที่ความสามารถในการประยุกต์ใช้ของผู้พัฒนา โดยข้อได้เปรียบต่างๆทำให้ DirectDrawเป็น library ที่น่าสนใจเป็นอย่างยิ่ง

2.10 วิธีการแสดงเสียง

ในส่วนนี้จะกล่าวถึง Direct Sound ที่ใช้ในการสร้างเสียง และมีประโยชน์อย่างมากในกรณีที่ไม่มีDriver ของ การ์ด เสียง ตัวนั้น โดย Direct Sound จะติดต่อกับการ์ดเสียงได้โดยตรง ไม่ต้องอาศัย



Driver ของ การ์ดเสียง ดังรูป2.7

รูปที่ 2.7 แสดงการทำงานของ Direct Sound

ส่วนประกอบพื้นฐานของ Direct Sound มีอยู่ สามส่วนได้แก่

- ขณะ run-time .DLL จะถูกโหลดขณะเรียกใช้ Direct Sound
- ขณะ compile-time Dsound.Lib จะทำการสร้าง Direct sound application
- ขณะ compile-time Dsound.h จะทำการสร้าง Direct sound application

ดังนั้นจะต้องทำการเพิ่ม file เหล่านี้เข้าไปใน application ที่ต้องการสร้างด้วยขั้นตอนการสร้าง Direct sound Object

2.10.1 **การสร้าง** Direct Sound Object

การที่จะสร้าง Direct sound object จะต้องทำการเรียก Function DirectSoundCreate() ซึ่งกำหนด prototype ดังนี้

```
HRESULT DirectSoundCreate(
LPGUID lpGuid, // GUID ของ การ์ดเสียง
LPDIRECTSOUND *lpDS, //interface pointer to obj
lunknow FAR *pUnkOuter ) //ปกติ NULL
```

โดยจะต้องประกาศ pointer เพื่อใช้ ชี้ Direct Sound Object ดังนี้

```
LPDIRECTSOUND lpds;
```

เมื่อทำการสร้าง Object ของ Direct Sound ขึ้นมาแล้ว และเมื่อเรียกใช้ เสร็จก็ต้องทำการคืนค่า Objectนั้นด้วยคำสั่ง

```
lpds - > Release( );
```

2.10.2.**การตั้งค่าของ** cooperation level

จะคล้ายกับการ set cooperation ของ Direct Draw โดยการเรียก Function SetCooperativeLevel( )โดยมี prototype ดังนี้

```
HRESULT SetcooperativeLevel ( HWND hwnd, // window handle
DWORD dwLevel ); // ตั้งค่า Cooperation
```

ค่าของ cooperative level	ความหมาย
DSSCL_NORMAL	ตั้งเป็นแบบปกติ โดยมีขนาด pprimary bufferสำหรับ ความถี่เสียง 22 kHz Stereo แบบ 8bit เป็นที่นิยมที่dfgสุด
DSSCL_PRIORTY	อนุญาติให้เปลี่ยนขนาดของ primary buffer ได้ตามต้องการ และสามารถเปลี่ยนรูปแบบเสียงเป็นแบบ 16 bit ได้
DSSCL_EXCLUSIV	เหมือนกับ priority แต่จะได้ยินเสียงเพียงเสียงเดียว
DSSCL_WRITEPRIMARY	เป็นระดับสูงสุด จะควบคุมทุกอย่างของ primary buffer เหมาะสำหรับการที่จะสร้างเสียงขึ้นมาเอง

ตารางที่ 2.4 **แสดงค่าของ** cooperation level DirectSound

2.10.3 Primary และ secondary buffers

Primary buffer จะทำการสร้างเสียงขึ้นมาบนการ์ดเสียง โดยที่ตัวโปรแกรมไม่ต้องไปสนใจ DirectSound จะเป็นตัวจัดการแทนให้ และไม่จำเป็นต้องสร้าง primary buffer โดยเพียงแค่ set ค่า cooperative level ต่ำสุด Direct sound ก็จะจัดการให้ส่วนที่จะต้องสนใจจริงๆ ก็คือ Secondary buffer ซึ่งส่วนนี้จะทำการเล่นเสียงที่โปรแกรมต้องการจะมีขนาดเท่า ไรก็ได้ ขึ้นอยู่กับหน่วยความจำที่มีอยู่ อย่างไรก็ตาม SRAM ที่อยู่บนการ์ดเสียง ก็เพียงพอที่จะเก็บเสียงได้มากตามที่ต้องการแล้ว

2.10.4 การสร้าง Secondary Buffer

การจะสร้าง secondary sound buffer ต้องทำการเรียก Function CreateSoundBuffer() โดยกำหนด prototype ไว้ดังนี้

```
HRESULT CreateSoundBuffer(
LPCDSBUFFERDESC lpcDSBufferDesc, // ตัวชี้ไปยัง DSBUFFERDESC
LPLPDIRECTSOUNDBUFFER lplpDSBuff, //ตัวชี้ไปยัง sound buffer
IUnknown FAR *pUnkOuter ); // ปกติใช้ NULL
```

ก่อนที่จะสร้าง secondary sound buffer จะต้องกำหนดโครงสร้างของ DirectSoundbuffer เสียก่อน โดยมีรูปแบบดังนี้

```
typedef struct {
DWORD dwSize; // ขนาดของโครงสร้างนี้
DWORD dwFlags; // ค่า control flags
DWORD dwBufferBytes; // ขนาดของ sound buffer ในรูป byte
DWORD dwReserved; // ไม่ใช้
LPWAVEFORMATEX lpwfxFormat; // รูปแแบบ wave format
} DSBUFFERDESC *LPDSBUFFERDESC;
```

ค่าของ control Flags	ความหมาย
DSBCAPS_CTRLALL	BUFFER ต้องมีการควบคุมขนาด
DSBCAPS_CTRLDEFAULT	เป็นการตั้งค่า default
DSBCAPS_CTRLFREQUENCY	ควบคุมความถี่ของเสียง
DSBCAPS_CTRLPAN	ควบคุมความ balance ของเสียง
DSBCAPS_CTRLVOLUME	ควบคุมความดังของเสียง

DSBCAPS_STATIC	บอกว่า buffer จะใช้สำหรับข้อมูลเสียงแบบ static
DSBCAPS_LOCHARDWARE	ใช้ Hardware ทำการสร้างเสียงขึ้นมาและใช้หน่วยความจำแทน buffer
DSBCAPS_LOCSOFTWARE	บังคับให้ buffer ใช้หน่วยความจำของsoftware เป็นตัวเก็บ
DSBCAPS_PRIMARYBUFFER	บอกให้รู้ว่า buffer นี้แป็นแบบ primary buffer ตั้งค่านี้ก็ต่อเมื่อต้องการสร้าง primary buffer ด้วยตัวเอง

ตารางที่ 2.5 แสดงค่า control flags ของโครงสร้าง DirectSound Buffer

โดยส่วนใหญ่ จะกำหนดค่า Control flag เป็น DSBCAPS_CTRLDEFAULT | DSBCAPS_STATIC | DSBCAPS_LOCSOFTWARE

ต่อไปเป็นการ กำหนดรายละเอียดของ เสียง โดยสร้าง รูปแบบ ของ Waveformatex ขึ้นมาดังนี้

```
typedef struc {
WORD wFormatTag ; // ปกติใช้ WAVE_FORMAT_PCM
WORD nChannels ; // จำนวนของช่องสัญญาณ จะใช้ 1 สำหรับเสียงแบบ mono และ 2 สำหรับ stereo
DWORD nSamplesPerSec; // samples per second
DWORD nAvgBytesPerSec ; // average data rate
WORD nBlockAlign ; // nchannels * bytespersample
WORD wBitsPerSample ; // bits per sample
WORD cbSize ; // ตั้งเท่ากับ 0
} WAVEFORMATEX;
```

2.10.5 การเขียนข้อมูลลงบน Secondary Buffer

ใน Direct Draw Surface จะต้องทำการ lock surface memory ก่อน แล้วจึงเขียน ใน DirectSound ก็เช่นเดียวกัน แต่ต่างกันที่ Direct sound มี 2 พอยเตอร์ ที่ใช้ในการเขียน โดยพอยเตอร์แรก จะใช้เขียนข้อมูล ในส่วนแรก และข้อมูลที่เหลือ สำหรับพอยเตอร์ที่ สอง โดยมีการกำหนด prototype Lock() ไว้ดังนี้

```
HRESULT Lock(
DWORD dwWriteCursor,
DWORD dwWriteBytes, // ขนาดของ Buffer ที่ต้องการ lock
```

LPVOID lplpvAudioPtr1,

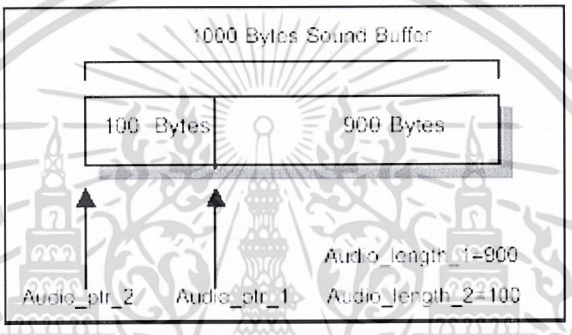
LPDWORD lpdwAudioBytes1,

LPVOID lplpvAudioPtr2,

LPDWORD lpdwAudioBytes2,

DWORD dwFlags);

ตัวอย่าง เช่น ต้องการสร้าง Sound buffer ขนาด 1000 bytes เมื่อทำการ lock buffer เพื่อจะทำการเขียน จะต้องใช้ สองพอยเตอร์ในการอ้างอิง โดยขนาดของพอยเตอร์ตัวแรก อาจจะมีขนาด 900 bytes ส่วนที่เหลือ 100 bytes จะเป็นของพอยเตอร์ตัวที่สอง ดังรูปภาพที่ 2.8



รูปที่ 2.8 การเขียนข้อมูลเสียงลง Sound Bufer

หลังจากทำการสร้าง Sound Buffer และ โหลดsoundทั้งหมดขั้นต่อไปก็จะเป็นการนำ sound ที่พร้อมนำมาแสดง โดย Direct X มี Function มากมายในการแสดงเสียงที่จำเป็นได้แก่

2.10.6 การแสดงเสียง

ใช้ ฟังก์ชั่น Play() ในการแสดงเสียงจาก sound buffer โดย มีตัวต้นแบบดังนี้

HRESULT Play(

DWORD dwReserved1, DWORD dwReserved2, // กำหนดเป็น 0 ทั้งสองค่า

DWORD dwFlags); // ควบคุม flags ในการเล่น ถ้าต้องการเล่นวนรอบ ตั้งค่าเป็น

//DSBPLAY_LOOPING แต่ถ้าต้องการเล่นเพียงครั้งเดียว ตั้งค่า เป็น 0

2.10.7 การหยุดเสียง

ถ้าต้องการหยุดเสียงที่กำลัง เล่นอยู่ ใช้ ฟังก์ชั่น Stop ( ) โดยกำหนดตัวต้นแบบดังนี้

HRESULT Stop( ) ;

2.10.8 การตั้งระดับความดังของเสียง

ถ้าต้องการปรับความดังเสียง จะเรียกใช้ ฟังก์ชั่น SetVolume( ) มีการกำหนด ต้นแบบดังนี้

HRESULT SetVolume(LONG VOLUME) ;

2.11 วิธีการรับข้อมูล จาก Mouse และ Keyboard

DirectInput เป็นไลบรารีที่ช่วยให้โปรแกรมเมอร์ทำการติดต่อและเรียกใช้อุปกรณ์ในการรับอินพุทชนิด ต่างๆได้อย่างมีประสิทธิภาพ direct input ครอบคลุมอุปกรณ์ในการรับอินพุทมาตรฐานทุกชนิด อันประกอบไปด้วย

- คีย์บอร์ดมาตรฐาน
- เมาส์ ทั้ง 2 และ 3 ปุ่ม
- จอยสติ้ก จอยโยก ทั้งที่เป็นแบบอนาลอค และ ดิจิตอล
- อุปกรณ์ควบคุมการขับเคลื่อน ใช้สำหรับจำลองการขับรถทั้งที่เป็นแบบอนาลอค และดิจิตอล
- อุปกรณ์ที่ตอบสนองแรงสั่นสะเทือน จอยสติ้กหรืออุปกรณ์อื่นๆที่สามารถคำนวณเพื่อทำให้เกิดแรงสั่นโต้ตอบกับผู้ใช้
- virtual headgear

2.11.1 การใช้ DirectInput

direct input ประกอบไปด้วยรันไทม์ไลบรารี และ 2 คอมไพล์ไฟล์ DINPUT.LIB DINPUT.H สำหรับโปรแกรมหรือโปรเจคที่ต้องการใช้ความสามารถของ direct input จะต้องเพิ่มไฟล์ตามที่ได้กล่าวมาข้างต้นไว้ในโปรเจคนั้นๆด้วย

ขั้นตอนในการใช้ direct input- สร้างออปเจค direct input หลักด้วย DirectInputCreate( ).

- สร้างออบเจคของอุปกรณ์อินพุทด้วย CreateDevice( ).
- กำหนดลักษณะในการใช้งานของแต่ละอุปกรณ์ด้วย SetCooperativeLevel( ).
- กำหนดรูปแบบข้อมูลของแต่ละอุปกรณ์ด้วย SetDataFormat( ). และทำการกำหนดคุณสมบัติพิเศษของอุปกรณ์ต่างๆ ด้วย SetProperty( ).
- ทำการจองอุปกรณ์ต่างๆด้วย Acquire( ).
- รับข้อมูลจากแต่ละอุปกรณ์ที่ได้ทำการจองไว้แล้วด้วย GetDeviceState( ).การอ่านข้อมูลของแต่ละอุปกรณ์จะถูกอ่านเป็นเรคคอร์ด แต่ละเรคคอร์ดจะถูกแปลงเป็นข้อมูล เพื่อให้โปรแกรมเมอร์นำข้อมูลต่างๆเหล่านี้ไปใช้ แต่ละอุปกรณ์ก็จะมีรูปแบบของเรคคอร์ดที่แตกต่างกัน ถึงแม้ว่าจะมีรูปแบบที่แตกต่างกัน แต่เรคคอร์ดเหล่านี้ก็มีลักษณะใกล้เคียงกันมาก ทำให้เกิดการสร้างข้อมูลอินพุทที่เป็นรูปแบบหลักๆขึ้นโปรแกรมเมอร์สามารถร้องขอข้อมูลอินพุทจาก DirectInput ได้ 2 วิธี

1. **แบบทันทีทันใด**(immediate input) ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของอุปกรณ์อินพุทขณะนั้น

2.**แบบมีการพักข้อมูล**(buffered input) ข้อมูลที่ถูกเก็บเป็นดาต้าเบสของเหตุการณ์ที่

เกิดขึ้นตั้งแต่การร้องขออินพุทครั้งสุดท้ายในการใช้ DirectInput โปรแกรมเมอร์ไม่ต้องกังวลเกี่ยวกับแมสเสทหรือการตอบสนองกับผู้ใช้ว่าจะเป็น ไปอย่างเชื่องช้า เพราะ DirectInput จะทำงานโดยตรงกับไดร์เวอร์ของอุปกรณ์อินพุทนั้นๆ

2.11.2 การติดต่อกับคีย์บอร์ดผ่านทาง DirectInput

2.11.2.1 DirectInput Object จะสร้างออปเจคหลักเพื่อใช้ในการจัดการ ติดต่อกับอุปกรณ์อินพุทชนิดอื่น โดยออปเจคหลักนี้ถูกอ้างถึงด้วย

LPDIRECTINPUT lpdi;

ทำการสร้างออปเจคหลักด้วยฟังก์ชั่น DirectInputCreate( ) โดยมีรูปแบบการของฟังก์ชั่นดังนี้

HRESULT DirectInputCreate(

HINSTANCE hinst,

DWORD dwversion,

LPDIRECTINPUT * lplpDirectInput,

LPUNKNOWN punkOuter);

ฟังก์ชั่นจะคืนค่า DI_OK(DirectInput Okey) ถ้าการเรียกใช้ฟังก์ชั่นนี้ผ่านและจะคืนค่าเป็นค่าผิดพลาดอื่นๆถ้าฟังก์ชั่นนี้ไม่ ผ่าน สำหรับค่าพารามิเตอร์ต่างๆของ DirectInputCreate มีดังนี้

hinst ใช้อ้างถึงแอพพลิเคชั่นที่ต้องการใช้งาน DirectDraw ให้แน่ใจว่าค่า hinst นี้ถูกส่งค่ามาจากค่า hinstance ใน WinMain( )

dwversion กำหนดเวอร์ชั่นของ DirectInput ว่าจะใช้เวอร์ชั่นใด มีไว้เพื่อให้โปรแกรมเมอร์ที่ต้องการใช้ DirectInput เวอร์ชั่นเก่า แต่โดยทั่วไปจะถูกกำหนดค่าเป็น

DIRECTINPUT_VERSION ซึ่งหมายความว่า DirectInput ที่จะใช้เป็นเวอร์ชั่นล่าสุด

lplpDirectInput เป็นพอยเตอร์ที่อ้างอิงไปยังออปเจคหลักของ DirectInput

punkOuter ถูกกำหนดเป็น NULL เสมอ

ตัวอย่าง การสร้างออปเจคหลักของ DirectInput

LPDIRECTINPUT lpdi;

DirectInputCreate(hinstance, DIRECTINPUT_VERSION, &lpdi, NULL);

เมื่อไม่ต้องการใช้งาน ออปเจคหลักของ DirectInput ก็ทำการเคลียร์ค่าด้วยฟังก์ชั่น Release( ) เช่น

lpdi->Release( );

2.11.2.2 **ทำการสร้างออปเจคของคีย์บอร์ด** ด้วยฟังก์ชั่น CreateDevice( ) ฟังก์ชั่นนี้จะใช้เมื่อต้องการ สร้างออปเจคของอุปกรณ์อินพุทใดๆ รูปแบบการใช้งานฟังก์ชั่น CreateDevice( ) มีดังนี้

HRESULT CreateDevice(

REFGUI rguid,

LPDIRECTINPUTDEVICE* lplpDIDev,

LPUNKNOWN punkOuter);

ค่าพารามิเตอร์ต่างๆของ DirectInput มีดังนี้

rguid คือค่า GUID(Globally Unique Identifier) เป็นหมายเลขของอุปกรณ์อินพุทที่ต้องการสร้าง โดยปกติจะใช้ฟังก์ชั่น EnumDevice() เพื่อหาค่า GUID ของอุปกรณ์อินพุทนั้นๆ แต่ใน DirectInput มีการกำหนดค่า GUID ของคีย์บอร์ดและเมาส์ไว้ให้เรียบร้อยแล้ว โดยจะมีค่าเป็น

GUID_SysKeyboard สำหรับคีย์บอร์ดมาตรฐาน

GUID_SysMouse สำหรับเมาส์มาตรฐาน

อย่างไรก็ตามการใช้ค่า GUID ที่ถูกประกาศไว้แล้วนี้จำเป็นต้องเพิ่ม#define INITGUID ไว้ส่วนบนของทุกไฟล์ C/C++ ที่มีการประกาศOBJBASE.H เพื่อคอมไพเลอร์จะรวมข้อมูลเกี่ยวกับCOM และค่า GUID ที่ได้ถูกประกาศไว้แล้ว

lplpDiDev เป็นตำแหน่งที่พอยเตอร์ชี้เพื่ออ้างอิงถึงอุปกรณ์อินพุทชนิดนั้นๆ

punkOuter ถูกกำหนดให้เท่ากับ NULL เสมอ

ตัวอย่าง การสร้างออปเจคของคีย์บอร์ด

```
#define INITGUID
#include <OBJBASE.H>
#include "DINPUT.H"
//ทำการสร้างออปเจคหลักตามที่ได้กล่าวมาในขั้นก่อนหน้านี้หลังจากสร้างออปเจคหลักแล้วจึงทำ
//การสร้างออปเจคของคีย์บอร์ด
LPDRECTINPUTDEVICE lpdikey;
lpdi->CreateDevice(GUID_SysKeyboard, &lpdikey, NULL);
```

2.11.2.3 **กำหนดลักษณะการใช้งานของคีย์บอร์ด** แต่ละอุปกรณ์ก็จะมีลักษณะการใช้งานเป็นของตัวเอง SetCooperativeLevel( ) มีรูปแบบการเรียกใช้ดังนี้

HRESULT SetCooperativeLevel(

HWND hwnd,

DWORD dwFlag);

ค่าของ dwFlags	ความหมาย
DISCL_BACKGROUND	กำหนดให้แอพพลิเคชั่ นสามารถใช้อุปกรณ์อินพุทนั้นๆได้แม้ว่าแอพพลิเคชั่นจะอยู่ในโหมด background
DISCL_FOREGROUND	แอพพลิเคชั่นไม่สามารถเรียกใช้อุปกรณ์อินพุทนั้นๆได้ในโหมด background เพราะอุ ปกรณ์อินพุทจะถูกปล่อยทันทีเมื่อ แอพพลิเคชั่ นเปลี่ยนเป็นโหมด background
DISCL_EXCLUSIVE	หลังจากทำการจองอุปกรณ์อินพุทนั้นๆแล้ว จะมีได้เพียงแอพพลิเคชั่นเดียวที่ ร้องขอ DISCL_EXCLUSIVE ส่วนแอพพลิเคชั่นอื่นจะต้องรร้องขอการใช้งานอุปกรณ์อินพุทแบบ DISCL_NONEXCLUSIVE ได้
DISCL_NONEXCLUSIVE	แอพพลิเคชั่นสามารถใช้งานอุ ปกรณ์อินพุทร่วมกันได้

ตารางที่ 2.6 แสดงค่า dwFlag ลักษณะการใช้งานของ Keyboard

ค่าพารามิเตอร์ต่างๆของ SetcooperativeLevel( ) มีดังนี้

hwnd ใช้อ้างถึงวินโดวส์ของแอพพลิเคชั่นนั้นๆ

dwFlag เป็นค่าที่กำหนดลักษณะของอุปกรณ์อินพุทนั้นๆมีทั้งหมด 4 ค่า

สำหรับเกมที่ใช้โหมดแบบ full screen จะมีการกำหนดลักษณะของอุปกรณ์อินพุทเป็นแบบ DISCL_BACKGROUND และ DISCL_NONEXCLUSIVE

ตัวอย่าง การใช้ฟังก์ชั่น SetCooperativeLevel( )

lpdikey->SetCooperativeLevel(hwnd, DISCL_BACKGROUND |

DISCL_NONEXCLUSIVE);

2.11.2.4 **กำหนดรูปแบบข้อมูลของคีย์บอร์ด** เพื่อกำหนดว่าข้อมูลที่จะรับเข้ามาจากคีย์บอร์ดมีรูปแบบอย่างไรด้วยฟังก์ชั่น SetDataFormat( ) โดยมีรูปแบบการเรียกใช้ดังนี้

HRESULT SetDataFormat(LPCDIDATAFORMAT lpdf);

SetDataFormat มีพารามิเตอร์เพียงตัวเดียวคือ lpdf พารามิเตอร์นี้จะใช้โครงสร้างของ

DIDATAFORMAT ตามที่ได้แสดงด้านล่างนี้

Typedef struct {

DWORD dwSize; //ขนาดของโครงสร้างรูปแบบนี้ หน่วยเป็นไบต์

DWORD dwObjSize; //ขนาดของ DIOBJECTDATAFORMAT เป็นไบต์

DWORD dwFlags; //กำหนดการรายงานผลว่าเป็นรูปแบบ Reletive

//หรือว่า Absolute

DWOR dwDataSize; //ขนาดของ packetข้อมูล

DWORD dsNumObjs; //จำนวนออปเจคกำหนดให้เป็นขนาดของอาร์เรย์

LPDIOBJECTDATAFORMAT rgodf; //พอยเตอร์ชี้ไปยังอาร์เรย์ออปเจค

} DIDATAFORMAT, *LPDIDATAFORMAT;

แต่ DirectInput ได้กำหนดรูปแบบของข้อมูลสำหรับอุปกรณ์อินพุทมาตรฐานไว้เรียบร้อยแล้วนั่นคือ

c_dfDIKeyboard กำหนดรูปแบบข้อมูลสำหรับคีย์บอร์ดมาตรฐาน

c_dfDIMouse กำหนดรูปแบบข้อมูลสำหรับเมาส์มาตรฐาน

c_dfDIJoystick กำหนดรูปแบบข้อมูลสำหรับจอยสติ้กมาตรฐาน

ดั งนั้นโปรแกรมเมอร์สามารถใช้รูปแบบข้อมูลที่ ถูกกำหนดขึ้นไว้แล้วถ้าแอพพลิเคชั่นนั้นใช้ อุปกรณ์อินพุทมาตรฐาน

ตัวอย่าง การใช้ฟังก์ชั่น SetDataFormat

lpdikey->SetDataFormat(&c_dfDIKeyboard);

2.11.2.5 **ทำการจองคีย์บอร์ดด้วยฟังก์ชั่น** Acquire( )มีรูปแบบการเรียกใช้ดังนี้

HRESULT Acquire( );

ตัวอย่าง การใช้ฟังก์ชั่น Acquire( );

lpdikey->Acquire( );

2.11.2.6 **ทำการอ่านข้อมูลจากคีย์บอร์ด** สำหรับลักษณะข้อมูลของคีย์บอร์ดที่ถูกกำหนดโดย DirectInput มีลักษณะเป็น state data นั่นหมายความว่า เมื่ออ่านข้อ มูลขึ้นมาจากคีย์บอร์ด ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของคีย์บอร์ด สำหรับการอ่านข้อมูลจากคียบอร์ด จะใช้ฟังก์ชั่น GetDeviceState( ) ฟังก์ชั่นนี้จะอ่านค่าสถานะ ปัจจุบันของคีย์บอร์ดขึ้นมา หรือสิ่งที่คีย์บอร์ดได้ทำครั้งสุดท้ายในอินพุทลูปโดยมีรูปแบบการเรียกใช้ฟังก์ชั่นดังนี้

HRESULT GetDeviceState (

DWORD cbData, //ขนาดของเรคคอร์ดข้อมูล

LPVOID lpvData); // พอย์เตอร์ชี้ไปยังตำแหน่งที่เห็บข้อมูล

ตัวอย่าง การใช้ ฟังก์ชั่น GetDeviceState

UCHAR keystate[256];

lpdikey ->GetDeviceState(256, keystate)

DirectInput จะมีค่าคงที่ของแต่ละคีย์บนคีย์บอร์ดเพื่อให้โปรแกรมเมอร์สามารถตรวจสอบสถานะของคีย์บนคีย์บอร์ด ได้ว่าเป็นอย่างไร โดยค่าคงที่นี้จะขึ้นต้นด้วย DIK_ แล้วตามด้วยค่าคีย์ ตามตารางด้านล่างนี้

ค่า	ความหมาย
DIK_ESCAPE	ปุ่ม ESC
DIK_0 – DIK_9	หมายเลข 0 ถึง 9
DIK_A – DIK_Z	A ถึง Z
DIK_RETURN	ปุ่ม Enter
DIK_LCONTROL	ปุ่ม Ctrl ทางซ.ายมือ
DIK_RCONTROL	ปุ่ม Ctrl ทางขวามือ
DIK_SPACE	ปุ่ม Spacebar
DIK_F1 – DIK_F12	ปุ่ม F1 ถึง F12
DIK_UP	ปุ่มลูกศรชี้ขึ้น
DIK_DOWN	ปุ่มลูกศรชี้ลง
DIK_LEFT	ปุ่มลูกศรชี้ไปทางซ.าย
DIK_RIGHT	ปุ่มลูกศรชี้ไปทางขวา
DIK_PRIOR	ปุ่ม PageUp
DIK_NEXT	ปุ่ม PageDown

ตารางที่ 2.7 แสดงค่าคงที่ของแต่ละคีย์บนคีย์บอร์ด

หลังจากทำการอ่านค่าสถานะของคีย์บอร์ดแล้วจะสามารถตรวจสอบได้ว่าคีย์ที่สนใจมีสถานะเป็นอย่าง ไรด้วยการตรวจสอบค่าบิตของคีย์นั้นๆถ้าคีย์ถูกกดค่าบิต ox80 จะมีค่าเป็น1แต่ถ้าคีย์นั้น ๆ ไม่ได้ถูกกด หรือคีย์นั้นๆถูกปล่อยไปแล้วค่าบิต 0x80 จะมีค่าเป็น 0

ตัวอย่าง การตรวจสอบค่าของคีย์

int ship_x = 100, ship_y = 100;

```
UCHARkeystate[256];
lpdikey->GetDeviceState(256, keystate)
if (keystate[DIK_RIGHT] & 0x80) ship_x++;
if (keystate[DIK_DOWN] & 0x80) ship_y++;
```

จากตัวอย่างเป็นการตรวจสอบว่าคีย์ลูกศรชี้ลง หรือ ลูกศรชี้ไปด้านขวาถูกกดหรือไม่

2.11.3 การรับข้อมูลจากเมาส์

ขั้นตอนต่างๆในการใช้ DirectInput เพื่อติดต่อกับเมาส์นั้นจะคล้ายกับขั้นตอนในการใช้ DirectInput เพื่อติดต่อกับคีย์ บอร์ดจะต่างกันเพียง การอ่านค่าตำแหน่งของเมาส์เท่านั้น ขั้นตอนในการสร้างออปเจคเพื่อใช้ในการติดต่อและอ่านค่า จากเมาส์มีดังนี้

1. สร้างออปเจคเมาส์
2. ทำการกำหนดลักษณะการใช้งานของเมาส์
3. กำหนดรูปแบบข้อมูลที่ได้จากการอ่านค่าสถานะของเมาส์
4. ทำการจองเมาส์

2.11.3.1 สร้างออปเจคเมาส์

สำหรับการสร้างออปเจคเมาส์นี้ จะใช้ฟังก์ชั่น CreateDevice( ) พารามิเตอร์ GUID ที่จะใช้เป็นหมายเลขของเมาส์จะใช้ค่า GUID ที่ถูกประกาศไว้แล้วโดย DirectInput นั่นคือใช้ค่า GUID_SysMouse

ตัวอย่าง การเรียกใช้ฟังก์ชั่น CreateDevice( )

```
LPDIRECTINPUTDEVICE lpdimouse;
//ทำการสร้างเมาส์ออปเจค
lpdi->CreateDevice(GUID_SysMouse, &lpdimouse, NULL);
```

2.11.3.2 ทำการกำหนดลักษณะการใช้งานของเมาส์

หลังจากการ CreateDevice ( ) ได้สำเร็จ ขั้นต่อไปก็จะทำการกำหนดลักษณะการใช้งานของอุปกรณ์อินพุทกับแอพพลิเคชั่นหนึ่งๆ อุปกรณ์อินพุทในที่นี้คือเมาส์ จะทำการกำหนดลักษณะการใช้งานได้ด้วยฟังก์ชั่น SetCooperativeLevel( ) สำหรับพารามิเตอร์ที่ใช้ในฟังก์ชั่นนี้จะเหมือนกับพารามิเตอร์ที่ใช้ในการกำหนดลักษณะการใช้งานให้กับคีย์บอร์ด โดยมีค่าแฟลก 4 ค่า นั่นคือ

- DISCL_BACKGROUND
- DISCL_FOREGROUND

- DISCL_EXCLUSIVE

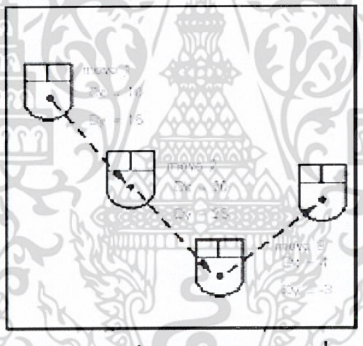
- DISCL_NONEXCLUSIVE

ค่าแฟลกทั้ง 4 ค่านี้มีความหมายเหมือนกับค่าแฟลกที่ใช้กำหนดลักษณะการใช้งานของคีย์บอร์ดสำหรับเกมที่ใช้ในโหมด Full Screen จะกำหนดค่าแฟลกเป็น DISCL_BACKGROUND และ DISCL_NONEXCLUSIVE เพื่อหลีกเลี่ยงข้อผิดพลาดจากการจองอุปกรณอินพุทชนิดเดียวกัน

2.11.3.3 ทำการกำหนดรูปแบบของข้อมูลที่ได้รับ

ขั้นตอนนี้เหมือนกับขั้นตอนในการกำหนดรูปแบบของข้อมูลให้กับคีย์บอร์ด จะมีการใช้รูปแบบของข้อมูลที่ได้ถูกประกาศไว้แล้วคือ c_dfDIMouseแต่รูปแบบของข้อมูลที่ได้จากการใช้c_dfDIMouse จะเป็นแบบสัมพันธ์(relative) นั่นหมายความว่าทุกๆครั้งที่มีการร้องขอข้อมูลจากเมาส์ ค่าตำแหน่งของเมาส์จะอยู่ในรูปผลต่าง (เปลี่ยนจากตำแหน่งที่เมาส์ชี้อยู่เท่าไหร่)จะใช้ฟังก์ชั่น SetDataFormat( ) เพื่อทำการกำหนดรูปแบบของข้อมูลที่ได้จากการอ่านค่าสถานะของเมาส์

lpdimouse->SetDataFormat(&c_dfDIMouse);



รูปที่ 2.9 **แสดงการเก็บค่าตำแหน่งเมื่อทำการเลื่อนเมาส์**

2.11.3.4 ทำการจองเมาส์

DirecInput จะทำการจองเมาส์ไว้ให้แอพพลิเคชั่นนั้นๆด้วยฟังก์ชั่น Acquire( )

lpdimouse->Acquire( );

2.11.3.5 การอ่านค่าจากเมาส์

จะสามารถอ่านค่าจากเมาส์ได้ด้วยฟังก์ชั่น GetDeviceState( ) แต่ข้อมูลที่ส่งคืนกลับมาจะต่างจากข้อมูลที่ถูกส่งคืนจากคีย์บอร์ด ข้อมูลที่ได้จากการอ่านค่าเมาส์จะถูกส่งคืนมาในรูปของ DIMOUSESTATE

typedef struct {

LONG lx; //ค่าแกน x ของเมาส์

LONG ly; //ค่าแกน y ของเมาส์

LONG lz; //ค่าแกน z ของเมาส์ หมายถึงปุ่มที่วงล้อ

BYTE rgbButtons[4] //ปุ่มของเมาส์ บิตสูงจะหมายถึงปุ่มนั้นถูกกด

} DIMOUSESTATE, *LPDIMOUSESTATE

จากโครงสร้างข้อมูลแบบนี้ ค่าตำแหน่งของแกน x y และค่าของวงล้อจะถูกส่งมายังพารามิเตอร์ lx ly และ lz และค่าสถานะของปุ่มจะถูกส่งมายัง rgbButtons[4] เป็นอาร์เรย์ขนาด 4 ไบต์ สำหรับวิธีการอ่านค่าสถานะจากเมาส์จะใช้คำสั่ง GetDeviceState( )

DIMOUSESTATE mouse_state;

lpdimouse->GetDeviceState(sizeof(DIMOUSESTATE),(LPVOID)&mouse_state);

เมื่อได้ข้อมูลมาแล้วการตรวจสอบว่าข้อมูลที่ได้มานั้นมีค่าเช่นไรสามารถตรวจสอบได้ดังนี้// เป็นการตรวจสอบว่ามีการกดเมาส์ปุ่มซ้าย ถ้าพบว่าปุ่มถูกกดก็ให้ทำงานตามหลังวงเล็บปากกา

if (mouse_state.rgbButtons[0])

{

}

// เป็นการตรวจสอบว่ามีการกดเมาส์ปุ่มขวา ถ้าพบว่าปุ่มถูกกดก็ให้ทำงานตามหลังวงเล็บปีกกา

if (mouse_state.rgbButtons[1])

{

}

เมื่อต้องการยกเลิกการใช้เมาส์จะต้องทำการคืนเมาส์ให้สู่ระบบ ด้วยฟังก์ชั่น Unacquire( ) และทำการลบออปเจคของเมาส์ทิ้งไปด้วยฟังก์ชั่น Release( )

lpdimouse->Unacquire( );

lpdimouse->Release( );

2.12 วงจรเกม

หลักการโดยทั่วไปจะทำการสร้างวงจรของเกม(Game Loop)ขึ้นมา เพื่อรอรับ Input จากผู้เล่นประมวลผล แล้วทำการแสดงผล และกลับไปรอรับ Input ประมวลผล และแสดงผล เป็นเช่นนี้เรื่อยๆ จนกว่าผู้เล่นจะสามารถทำให้เงื่อนไขการวน Loop เป็นจริง เป็นต้นว่าในปัญหาพิเศษนี้ จะต้องบรรลุเป้าหมาย หรืออาจกดปุ่ม exit

2.13 โครงร่างการเขียนเกม

หลังจากที่ได้ศึกษาในส่วนของ Window Programming ไปบ้างแล้ว ในส่วนนี้ จะกล่าวถึงโครงร่างการเขียนเกม (Game console) ที่ถูกออกแบบมาเพื่อใช้ในการเขียน เกมโดยตรง โดย game consoleจะทำงานภายใน Winmain จุดประสงค์ของการสร้าง Game Console ขึ้นมาก็เพื่อจะสร้างเกมที่รันบนระบบปฏิบัติการ วินโดวส์และใช้ Direct X ได้ด้วยความสะดวก ดังนั้น ในปัญหาพิเศษนี้ จะกระทำบน Game Console ซึ่ง GameConsole จะแบ่ง ออก เป็น 3 ฟังก์ ชั่น ได้แก่ Game_Init( ) , Game_Shutdown( ) และ Game_Main( )

2.13.1 Game_Init

มีหน้าที่กำหนดค่าเริ่มต้นให้แก่ระบบ เช่น ปรับความละเอียดหน้าจอ เตรียม back buffer , Frontbuffer ทำการตั้งค่าการ์ดเสียง เป็นต้น โดยฟังก์ชันนี้จะถูกเรียกหลังจาก วินโดวส์ ได้ตั้งระบบเริ่มต้นเรียบร้อยแล้ว

2.13.2 Game_Shutdown

มีหน้าที่คืนทรัพยากรต่างๆให้แก่ระบบ เมื่อพร้อมที่จะจบการทำงานของระบบ จะถูกเรียกหลังจากระบบทำงานเสร็จ เตรียมจะปิด

2.13.3 Game_Main

จะคล้ายกับ ฟังก์ชั่น main( ) ในภาษา C ทั่วไป โดยตัวโปรแกรม จะอยู่ในส่วนนี้ และ ฟังก์ชั่น Game_main จะอยู่ใน event loop เพื่อที่จะทำการประมวลผลไปเรื่อยๆ จนกว่าโปรแกรมจะทำงานเสร็จพร้อมที่จะปิด

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

3.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบมีรายละเอียดดังนี้

3.1.1 การออกแบบระบบเกม

การออกแบบระบบเกมเป็นส่วนที่กำหนดการดำเนินเรื่องต่างๆของเกม ตามแนวคิดที่ได้วางไว้ โดยการออกแบบมีรายละเอียดต่างๆดังนี้

- การออกแบบทั่วไป
- การออกแบบคำสั่งต่างๆและการรับข้อมูล
- การออกแบบการติดต่อกันระหว่างเครื่อง

3.1.1.1 การออกแบบทั่วไป

การออกแบบทั่วไปเป็นการออกแบบรูปลักษณ์ของเกม โดยมีสิ่งต่างๆดังนี้

- ตัวละครที่เห็นในเกม มี 2 แบบ เพื่อให้สามารถแบ่งได้เป็น 2 ทีม
- เกมมีลักษณะเป็น 3 มิติ แนว Shooting แบบเรียลไทม์ (Real time)
- มุมมองที่ใช้เป็นมุมมองบุคคลที่ 1 (First-Person View) คือตำแหน่งการมองที่ผู้เล่นแต่ละคนเห็น จะเป็นมุมมองที่แต่ละตัวละครเห็น
- จุดที่ใช้เล็งยิงจะอยู่ที่กลางจอภาพเสมอ
- ผู้เล่นสามารถเลือกซื้อปืนได้ โดยปืนแต่ละชนิดจะมีราคา และประสิทธิภาพในการโจมตีต่างกัน
- ผู้เล่นสามารถมีปืนสั้น และปืนยาวได้อย่างละ 1 กระบอก
- ผู้เล่นสามารถทิ้งปืนที่มีอยู่เพื่อเก็บปืนใหม่ได้
- ผู้เล่นสามารถทำอันตรายฝ่ายเดียวกันได้
- เมื่อผู้เล่นสามารถทำลายตัวละครฝ่ายตรงข้ามได้ จะได้เงินเพิ่ม 800 ทันที และเมื่อเริ่มรอบใหม่จะได้เงินเพิ่ม 1000 ต่อ 1 ตัวละครที่ทำลายได้

3.1.1.2 การออกแบบคำสั่งต่างๆและการรับข้อมูล

เป็นการออกแบบคำสั่งในการสั่งให้ตัวละครปฏิบัติ ซึ่งตัวละครแต่ละตัวมีคำสั่งให้ดำเนินการ ดังนี้

- เคลื่อนที่
- โจมตี
- ติดต่อสื่อสารกันในทีม

- การแสดงเมนู

โดยที่แต่ละคำสั่งมีรายละเอียดดังต่อไปนี้

การเคลื่อนที่

เป็นคำสั่งที่ใช้สั่งให้ตัวละครเคลื่อนที่ไปยังจุดต่างๆ โดยรายละเอียดคำสั่งในการเคลื่อนที่ดังนี้

การเคลื่อนที่	*คำสั่ง*
เดินหน้า	กดปุ่ม W
ถอยหลัง	กดปุ่ม S
สไลด์ไปด้านซ้าย	กดปุ่ม A
สไลด์ไปด้านขวา	กดปุ่ม D
ย่อตัวอยู่กับที่	กดปุ่ม Ctrl
ย่อตัวแล้วเดินไปข้างหน้า	กดปุ่ม Ctrl + W
ย่อตัวแล้วเดินถอยหลัง	กดปุ่ม Ctrl + S
ย่อตัวแล้วสไลด์ไปด้านซ้าย	กดปุ่ม Ctrl + A
ย่อตัวแล้วสไลด์ไปด้านขวา	กดปุ่ม Ctrl + D
กระโดด	กดปุ่ม Space bar
หมุนตัวไปทางซ้าย	เลื่อนเมาส์ไปทางซ้าย
หมุนตัวไปทางขวา	เลื่อนเมาส์ไปทางขวา
มองขึ้นข้างบน	เลื่อนเมาส์ไปข้างหน้า
มองลงข้างล่าง	เลื่อนเมาส์ถอยหลัง

ตาราง 3.1 คำสั่งในการเคลื่อนที่

การโจมตี

เป็นคำสั่งในการโจมตีด้วยการยิงปืน โดยแต่ละคนจะมีกระสุนสำรองเมื่อยิงจนลูก

กระสุนหมดก็สามารถบรรจุกระสุนใหม่ได้จนกว่ากระสุนสำรองจะหมด

การโจมตี	*คำสั่ง*
ยิง	**คลิกเมาส์ปุ่มซ้าย**
บรรจุกระสุนใหม่	**กดปุ่ม R**

ตาราง 3.2 คำสั่งในการโจมตี

การแสดงเมนู

เป็นคำสั่งในการแสดงเมนูเมื่อผู้เล่นต้องการซื้อของในเกม ทำได้โดยการกดปุ่ม B ซึ่งจะประกอบด้วยเมนูดังนี้

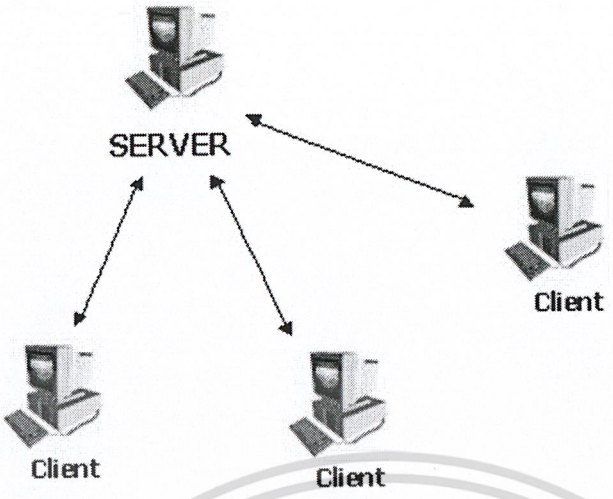
1. ซื้อปืนสั้น
2. ซื้อปืนยาว
3. ซื้อกระสุนปืนสั้น
4. ซื้อกระสุนปืนยาว
5. ซื้อเสื้อเกราะ
6. ยกเลิกการแสดงเมนู

การติดต่อสื่อสารกันในทีม

เป็นการติดต่อสื่อสารกันในกลุ่มโดยการพิมพ์ข้อความ โดยเมื่อต้องการพิมพ์ข้อความให้กดปุ่ม Y แล้วพิมพ์ข้อความที่ต้องการ จากนั้นเมื่อพิมพ์เสร็จให้กด Enter เพื่อส่งข้อความ ซึ่งข้อความที่ส่งนั้นจะปรากฏอยู่ในหน้าจอของผู้เล่นในทีมเท่านั้น

3.1.1.3 การออกแบบการติดต่อกันระหว่างเครื่อง

การติดต่อกันระหว่างเครื่องจะเป็นลักษณะ Client – Server ดังนี้



รูปที่ 3.1 การสื่อสารแบบ Client / Server

โดยเครื่องผู้เล่นจะเป็น Client ส่วนเครื่องที่เป็น Server จะมีหน้าที่คอยจัดการการสื่อสารภายในเกม

3.1.2 การออกแบบตัวละคร

การออกแบบตัวละครประกอบด้วยการออกแบบตัวละคร และท่าทางต่างๆของตัวละคร โดยที่โปรแกรมที่ใช้ในการออกแบบนี้คือโปรแกรม 3DMax เวอร์ชั่น 5 โดยท่าทางต่างๆในเกมประกอบด้วย

- ท่าวิ่งหรือเคลื่อนที่
- ท่ายิง
- ท่าตาย
- ท่ายืน
- ท่านั่งย่อ
- ท่ากระโดด

โดยในท่าเคลื่อนไหวประกอบด้วยทิศทางต่างๆดังนี้

- วิ่งตรงไปข้างหน้า
- วิ่งถอยหลัง
- วิ่งสไลด์ไปข้างซ้าย
- วิ่งสไลด์ไปข้างขวา

3.1.3 การออกแบบอินเทอร์เฟซติดต่อกับผู้ใช้

ฉาก

ฉากในเกมนี้เป็นฉาก 3 มิติ โดยโปรแกรมที่ใช้ในการออกแบบคือโปรแกรม 3DMax เวอร์ชั่น 5 โดยเมื่อออกแบบฉากเสร็จแล้ว จะทำการเอ็กซ์พอร์ทฉากดังกล่าวให้มีนามสกุลเป็น .x

ส่วนแสดงสถานะของตัวละคร

สถานะของตัวละครจะแสดงอยู่ทางด้านล่างของจอภาพ ดังรูป

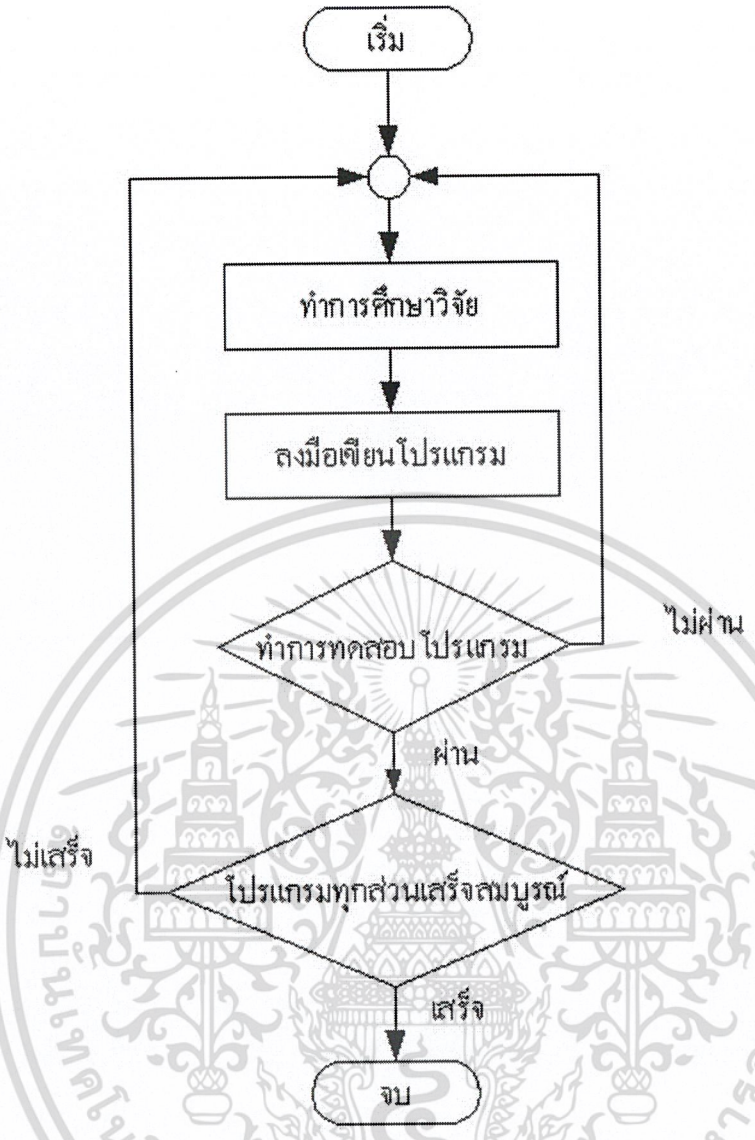
รูปที่ 3.2 ส่วนแสดงสถานะของตัวละคร

- พลังชีวิต คือ พลังชีวิตของตัวละครที่เหลืออยู่ โดยเริ่มเล่นตัวละครแต่ละตัวจะมีพลังชีวิตเท่ากับ 100
- พลังเสื้อเกราะ คือ พลังเสื้อเกราะที่เหลืออยู่ โดยเริ่มเล่นตัวละครแต่ละตัวจะมีพลังเสื้อเกราะเท่ากับ 0
- จำนวนกระสุนในปืน คือ จำนวนกระสุนที่มีเหลืออยู่ในปืน
- จำนวนกระสุนสำรอง คือจำนวนกระสุนสำรองที่ตัวละครแต่ละตัวมี โดยจะเอาไว้สำหรับบรรจุกระสุนใหม่ เมื่อจำนวนกระสุนในปืนเท่ากับ 0

3.1.4 **การออกแบบการตอบโต้ของคอมพิวเตอร์**

3.2ขั้นตอนการเขียนโปรแกรม

ส่วนใหญ่แล้วขั้นตอนต่างๆจะมีการทำงานเป็นเส้นตรง หมายถึงเมื่อทำเสร็จงานหนึ่งก็จะทำงานต่อไปโดยไม่มีการวนมาที่งานนั้นใหม่ แต่ขั้นตอนการเขียนโปรแกรมมีการทำงานที่ต้องวนมาศึกษาเรื่อยๆและทำการเขียนโปรแกรม และหากยังเขียนไม่สำเร็จ ก็จะต้องทำการศึกษาเพิ่มเติมใหม่ ดังแผนภาพต่อไปนี้



รูปที่ 3.3 ขั้นตอนการเขียนโปรแกรม

โปรแกรมเกมจะถูกแบ่งเป็นส่วนย่อย ๆ หลายส่วนดังนี้

- ส่วนของการแสดงผล จะอยู่ในClass Direct3d.h
- ส่วนของการติดต่อกับอุปกรณ์รับข้อมูลจะอยู่ในClass DirectInput.h
- ส่วนของการติดต่อการ์ดเสียงและการสร้างเสียง
- ส่วนของตัวผู้เล่น จะอยู่ในClass mycharecter.h
- ส่วนของฉากจะอยู่ ในClass Scene.h
- ส่วนของปืนจะอยู่ ในClass Gun.h

เหตุผลหลักของผู้ทำโครงงานแบ่งการทำงานออกเป็นหลายๆส่วนเพื่อต้องการจัดแบ่ง หน้าที่การทำงานให้เป็นประเภทเดียวกัน ทำให้ง่ายต่อการตรวจสอบ แก้ไข หรือเพิ่มเติม หลักการ แนว ความคิดและรูปแบบของข้อมูลในแต่ละส่วนการทำงาน

3.2.1 ส่วนการแสดงผล

ในส่วนการแสดงผลนี้จะรวมฟังก์ชั่นการทำงานที่เกี่ยวกับการแสดงผล การติดต่อกับ Hardware โดยผ่านทาง DirectDraw ทำการสร้าง Primary surface ,Secondary Surface ,OffScreen Surface

การทำภาพเคลื่อนไหว การแสดงเคอร์เซอร์ของเมาส์ และ การแสดงรูปยูนิตฟังก์ชั่นการทำงานในส่วนของการแสดงผล ใน class Direct3d.hมีดังนี้

- Direct3dInit(HWND hwnd)
- Direct3dInitGeometry(char namescene[ ])
- Direct3dRender( )
- Direct3dTerminate ( )

หลักการและการทำงานของฟังก์ชั่น

3.2.1.1 Direct3dInit(HWND hwnd)

ฟังก์ชั่นนี้จะทำการสร้าง object ของ DIRECT3D8 สำหรับอ่านรูปแบบการแสดงผลขณะปัจจุบันเพื่อกำหนดรูปแบบการแสดงผลใน Game และใช้สำหรับการสร้าง object ของ DIRECT3DDEVICE8 เพื่อนำมาใช้ในการ Render

3.2.1.2 Direct3dInitGeometry(char namescene [ ])

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับกำหนดการสร้าง model ฉาก , model ปืน และ animation ของตัวละครที่จะถูก Render

3.2.1.3 Direct3dRender ( )

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับเรียกการ Render Model ได้แก่ ฉาก,ปืนและ Animation ใน 3d Space และกำหนดการ Render ของตัวอักษรบน screen

3.2.1.4 Direct3dTerminate( )

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับ clear object ใน Class Direct3d.h ออกจาก Memory

3.2.2 ส่วนการติดต่ออุปกรณ์เพื่อรับอินพุท

ในส่วนนี้จะรวมฟังก์ชั่นที่ทำการติดต่อกับอุปกรณ์อินพุทโดยผ่านทาง Direct Input โปรแกรมเกมที่พัฒนาขึ้นมานี้ใช้ทั้ง เมาส์และคีย์บอร์ดในการรับอินพุท

ฟังก์ชั่นการทำงานในส่วนการติดต่ออุปกรณ์เพื่อรับอินพุต

- DirectInputInit(HWND hwnd,HINSTANCE hinstance)
- DirectKeyboard(Hawnda hwnd)
- DirectMouse(HWND hwnd)

- DirectInputTerminate ( )

หลักการและการทำงานของฟังก์ชั่น

3.2.2.1 DirectInputInit(HWND hwnd,HINSTANCE hinstance)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับสร้าง object เพื่อใช้สำหรับการรับ Input จาก Mouse และ Keyboard

3.2.2.2 DirectKeyboard(Hawnda hwnd)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับตรวจสอบการเกิด Input จาก Keyboard

3.2.2.3 DirectMouse(HWND hwnd)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับตรวจสอบการเกิด Input จาก Mouse

3.2.2.4 DirectInputTerminate ( )

ฟังก์ชั่นนี้ดป็นฟังก์ชั่นสำหรับ Clear Object ใน Class DirectInput.h ออกจาก memory

3.2.3 ส่วนการติดต่อกับผู้เล่น

ส่วนนี้จะอยู่ใน class my Charecter.h เป็น class ของผู้เล่นซึ่งประกอบด้วย Attribute ที่สำคัญดังนี้

- Name คือ ชื่อผู้เล่น
- Team คือ ทีมของผู้เล่น
- Health คือ พลังชีวิตของผู้เล่น (พลังเต็ม = 100)
- Armor คือ พลังเสื้อเกราะของผู้เล่น (พลังเต็ม = 100)
- Action คือ Action ของผู้เล่น ใช้สำหรับกำหนดการแสดง Animation ของผู้เล่น ในเครื่องผู้เล่นอื่น เช่น

  0.....ตาย

  1.....ยืนตรง

  2.....วิ่ง เป็นต้น

- Money คือ เงินของผู้เล่น
- Num_kill คือ จำนวนตัวละครที่ทำลายได้ (ในการ Join Game 1 ครั้ง)
- Num_kill_All คือ จำนวนตัวละครที่ทำลายได้ทั้งหมด (เริ่มนับตั้งแต่ผู้เล่นลงทะเบียน)
- Num_Dead คือ จำนวนครั้งที่ตาย (ในการ Join Game 1 ครั้ง)
- Num_Dead_All คือ จำนวนครั้งที่ตายทั้งหมด(เริ่มนับตั้งแต่ผู้เล่นลงทะเบียน)
- Xpos คือ ตำแหน่งของผู้เล่นในแกน X

- Ypos คือ ตำแหน่งของผู้เล่นในแกน Y
- Zpos คือ ตำแหน่งของผู้เล่นในแกน Z
- Xangle คือ มุมของทิศทางที่ผู้เล่นหันไป ที่กระทำกับแกน X
- Yangle คือ มุมของทิศทางที่ผู้เล่นหันไป ที่กระทำกับแกน Y
- G_Gun คือ Object Gun ของ ผู้เล่น (มี 2 Object)

3.2.4 ส่วนของปืน

ส่วนนี้จะอยู่ใน Class Gun.h เป็น Class สำหรับเก็บ Attribute ของปืนของผู้เล่นแต่ละคนซึ่งประกอบด้วย Attribute ที่สำคัญดังนี้

- Name คือ ชื่อปืน ใช้สำหรับเปรียบเทียบชื่อในการ Render Model ปืน ใน Class Direct3d.h
- Type คือ ประเภทของปืน (0 = ปืนสั้น ,1 = ปืนยาว)
- Bullet_per_mag คือ จำนวนกระสุนต่อ magazine
- Bullet_spare คือ จำนวนกระสุนสำรองที่มีได้
- Price_per_mag คือ ราคากระสุนต่อ magazine
- Price คือ ราคาปืน
- Power_per_bullet คือ ความแรงของกระสุนต่อนัด
- Bullet คือ จำนวนกระสุนใน magazine ปัจจุบัน
- Bullet_int_space คือ จำนวนกระสุนสำรอง

3.2.5 ส่วนของฉาก

ส่วนนี้จะอยู่ใน Class Scene.h เป็น class ของฉากซึ่งจะถูกเรียกใช้โดย Object ของ Class Direct3d.h ซึ่งจะประกอบด้วยฟังก์ชั่นที่สำคัญดังนี้

- InitGeometry (LDDIRECT3DDEVICE8 pd3dDevice,char name[ ])
- Render(LDDIRECT3DDEVICE8 pd3dDevice,float ts ,float tx ,float ty , float tz)
- Cleanup( )

หลักการและการทำงานของฟังก์ชั่น

3.2.5.1 InitGeometry (LDDIRECT3DDEVICE8 pd3dDevice,char name[ ])

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับสร้าง model ฉากซึ่งถูกเรียกใช่โดยฟังก์ชั่น Direct3dInitGeometry(name)

3.2.5.2Render(LDDIRECT3DDEVICE8 pd3dDevice,float ts,float tx,float ty,float tz)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับ Render Model ฉาก ซึ่งถูกเรียกใช้โดยฟังก์ชั่น

direct3dRender( )

3.2.5.3 Cleanup( )

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับคืนหน่วยความจำให้ระบบ ซึ่งถูกเรียกใช้โดยฟังก์ชั่น Direct3d Terminate( )

3.2.6 ส่วนการติดต่อกับผู้เล่นคนอื่น

ส่วนนี้จะอยู่ใน Class otherPlayer.h เป็น Class ของผู้เล่นอื่นในเกม ซึ่งประกอบด้วย Attribute ที่สำคัญ คือ

- g_fXangle_DS
- g_fYangle_DS
- g_fDistance

หลักการและการทำงานของฟังก์ชั่น

3.2.6.1 g_fXangle_DS

คือความแตกต่างระหว่างมุมของทิศทางที่ผู้เล่นหันไปกระทำกับแกน X กับ มุมที่ผู้เล่นนี้กระทำกับเราในแกน X

3.2.6.2 g_fYangle_DS

คือ ความแตกต่างระหว่างมุมของทิศทางที่ผู้เล่นหันไปกระทำกับแกน Y กับ มุมที่ผู้เล่นนี้กระทำกับเราในแกน Y

3.2.6.3 g_fDistance

คือ ระยะทางระหว่างเรากับผู้เล่นนี้ซึ่งทั้ง 3 Attribute นี้ใช้สำหรับการคำนวณการยิง ส่วน Attribute อื่นๆจะเหมือนกับ Attribute ของ Class Charecter.h

3.2.7 ส่วนของ Model ปืน

ส่วนนี้จะอยู่ใน Class Gunmodel.h เป็น Class สำหรับสร้าง Model ปืนที่จะถูก Render ใน Class Direct3d.h ประกอบด้วยฟังก์ชั่นที่สำคัญ คือ

- InitGeometrry(LDDIRECT3DDEVICE8 pd3dDevice , char name[ ])
- Render(LDDIRECT3DDEVICE8 pd3dDevice,float tx ,float ty ,float tz , float radx , float rady)
- Cleanup( )

หลักการและการทำงานของฟังก์ชั่น

3.2.7.1 InitGeometrry(LDDIRECT3DDEVICE8 pd3dDevice ,char name[ ])

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับสร้าง Model ปืน ซึ่งจะถูกเรียกใช้โดยฟังก์ชั่น Direct3dInitGeometry ( )

3.2.7.2 Render(LDDIRECT3DDEVICE8 pd3dDevice,float tx,float ty,float tz,float radx,float rady)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับ Render model ปืน ซึ่งจะถูกเรียกใช้โดยฟังก์ชั่น Direct3dRender( )

3.2.7.3 Cleanup( )

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับคืนหน่วยความจำให้ระบบ ซึ่งถูกเรียกใช้โดยฟังก์ชั่น Direct3dTerminate( )

3.2.8 ส่วนของ Animation

ส่วนนี้จะอยู่ใน class Animation.h เป็น class สำหรับสร้าง Animation ของตัวละครในท่าทางต่างๆ ที่จะถูก Render ใน Class Direct3d.h ซึ่ง Attribute Action ของ Class otherPlayer.h จะเป็นตัวกำหนดว่าผู้เล่นคนนั้นจะถูก Render ด้วย Animation ใด ซึ่งประกอบด้วยฟังก์ชั่นที่สำคัญ คือ

- AnimationInit(LDDIRECT3DDEVICE8 g_pd3dDevice)
- showAnimation (int action , LDDIRECT3DDEVICE8 g_pd3dDevice , float s , float x , float y , float z, float Rx , float Ry)
- AnimationTerminate( )

หลักการและการทำงานของฟังก์ชั่น

3.2.8.1 AnimationInit(LDDIRECT3DDEVICE8 g_pd3dDevice)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นสำหรับสร้าง Animation ทั้งหมดของตังละคร ซึ่งจะถูกเรียกใช้โดยฟังก์ชั่น Direct3dInitGeometry( )

3.2.8.2 showAnimation (int action , LDDIRECT3DDEVICE8 g_pd3dDevice , float s , float x , float y , float z, float Rx , float Ry)

ฟังก์ชั่นนี้เป็นฟังก์ชั่น สำหรับแสดง Animation ของตัวละครโดยจะใช้ parameter action เป็นตัวกำหนดว่างจะแสดง Animation ใด ซึ่งจะถูกเรียกใช้โดยฟังก์ชั่น Direct3dRender( )

3.2.8.3 AnimationTerminate( )

ฟังก์ชั่นนี้เป็นฟังก์ชั่นคืนค่าหน่วยความจำให้ระบบ ซึ่งจะถูกเรียกใช้โดยฟังก์ชั่น Direct3dTerminate( )

3.2.9 ส่วนของการติดต่อระหว่าง Client-Server

ส่วนนี้จะประกอบด้วยฟังก์ชั่นที่ทำหน้าที่ติดต่อระหว่าง Client-Server โดยผ่านทาง DirectPlay ซึ่งประกอบด้วยฟังก์ชั่นที่มีความสำคัญ โดยจะแบ่งเป็น 2 ส่วน คือ ฟังก์ชั่นของClient และ ฟังก์ชั่นของ Server

ฟังก์ชั่นที่สำคัญของ Client ประกอบด้วย

- StartDirectPlay
- InitDirectPlay
- DirectPlayMessageHandler

หลักการและการทำงานของฟังก์ชั่น

3.2.9.1 StartDirectPlay(HINSTANCE hInst,GUID g_guidApp)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่ทำการ initialize ต่างๆ แล้ว แสดง dialog box แรก

3.2.9.2 InitDirectPlay()

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่ทำการสร้าง object ต่างๆเกี่ยวกับ directplay ที่ต้องใช้

3.2.9.3 DirectPlayMessageHandler (PVOID pvUserContext,
DWORD dwMessageId, PVOID pMsgBuffer)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยจัดการ message ต่างๆ ที่เกิดจากการติดต่อกับ Server

ฟังก์ชั่นที่สำคัญของ server ประกอบด้วย

- StartDirectPlay
- StartServer
- DirectPlayMessageHandler
- SendActionMessage
- SendChatMessage
- SendWaveMessageToAll
- SendCreatePlayerMsg
- SendWorldStateToNewPlayer
- SendDestroyPlayerMsgToAll
- StopServer

หลักการและการทำงานของฟังก์ชั่น

3.2.9.4 StartDirectPlay(HINSTANCE g_hInstance)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่ทำการ initialize ต่างๆ แล้ว แสดง dialog box แรก

3.2.9.5 StartServer(HWND hDlg)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่ทำการรับข้อมูลจาก dialog box มากำหนดค่า

3.2.9.6 DirectPlayMessageHandler (PVOID pvUserContext,
DWORD dwMessageId, PVOID pMsgBuffer)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยรับ message ต่างๆ แล้วมี switch case ว่าเป็น message อะไร

3.2.9.7 SendActionMessage (char m[],float position[])

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยส่ง action message ประกอบด้วย รหัสที่บอก idของผู้เล่น และท่าทาง (เช่น เดินหน้า, ถอยหลัง ,เลี้ยวซ้าย –ขวา ฯลฯ) , ตำแหน่งของ character ขณะนั้น เป็นต้น

3.2.9.8 SendChatMessage(char m[])

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยส่งข้อความสนทนา ซึ่งจะมี ID ของผู้ส่งระบุไว้ด้วย

3.2.9.9 SendWaveMessageToAll (PACKET rpack)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยส่ง message ที่ได้รับจาก client หนึ่ง ไปยัง ทุกๆ client

3.2.9.10 SendCreatePlayerMsg (APP_PLAYER_INFO*
pPlayerAbout, DPNID dpnidTarget,PLAYER_NAME nameofnew)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยส่ง message ที่ประกอบด้วย dpnid และชื่อของ client ใหม่

ที่เพิ่งเข้ามา ไปยังเครื่องที่ระบุ โดยจะต้องมีการวนลูปเรียกฟังก์ชั่นนี้เพื่อส่งข้อมูลของ client ใหม่ ไปยังทุกๆ client ที่connect แล้ว

3.2.9.11 SendWorldStateToNewPlayer (DPNID
dpnidNewPlayer,float NOofPlayer,PLAYER_NAME nameofall[])

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยส่งmessageไปยัง client ใหม่ ประกอบด้วย id ที่จะกำหนดให้

และ อาเรย์ชื่อของ client ทั้งหมดที่ connect แล้ว

3.2.9.12 SendDestroyPlayerMsgToAll(APP_PLAYER_INFO*
pPlayerInfo)

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่คอยส่ง message ประกอบด้วย dpnid ของ client ที่ถูกทำลายแล้ว

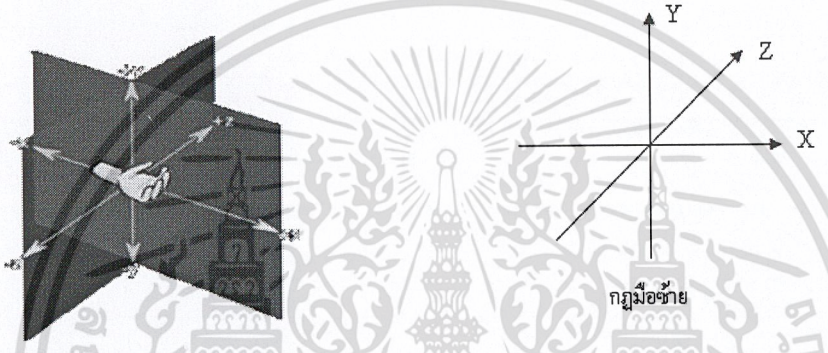
ไปยังทุกๆ client เพื่อให้ทุกๆclient ลบข้อมูลของ client นี้ออก

3.2.9.13 StopServer()

ฟังก์ชั่นนี้เป็นฟังก์ชั่นที่ทำการลบ object ต่างๆ ที่สร้างไว้

การตรวจสอบการยิง

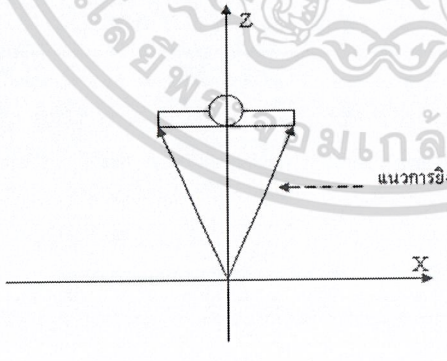
เนื่องจากเกมนี้เป็นลักษณะเกม 3 มิติ และกำหนด Coordinate System แบบ Left-Hand Coordinate System



รูปที่ 3.4 Left-Hand Coordinate System

ดังนั้นในการตรวจสอบการยิงจะมีการตรวจสอบ 2 ขั้นตอน คือ

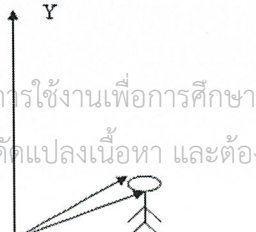
1. ตรวจสอบมุมของแนวการยิง (0) ที่กระทำกับแกน Y



รูปที่ 3.5 ตรวจสอบมุมของแนวการยิง ที่กระทำกับแกน Y

ถ้ามุมของแนวการยิงที่กระทำกับแกน 0 กับ -0 ให้ตรวจสอบมุมของแนวการยิงที่กระทำกับแกน X

2. ตรวจสอบมุมของแนวการยิงที่กระทำกับแกน X

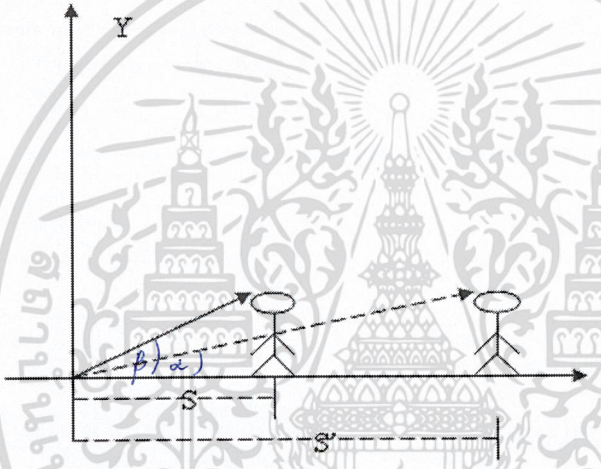


รูปที่ 3.6 ตรวจสอบมุมของแนวการยิง ที่กระทำกับแกน X

2.1 ถ้ามุมของแนวการยิงที่กระทำกับแกน X อยู่ในช่วงมุม คือ ยิงโดนหัวของคู่ต่อสู้(ยิงโดนหัวคู่ต่อสู้ พลังคู่ต่อสู้จะลดลง 10 เท่าของการลดลงของพลังจาการยิงโดนตัว)

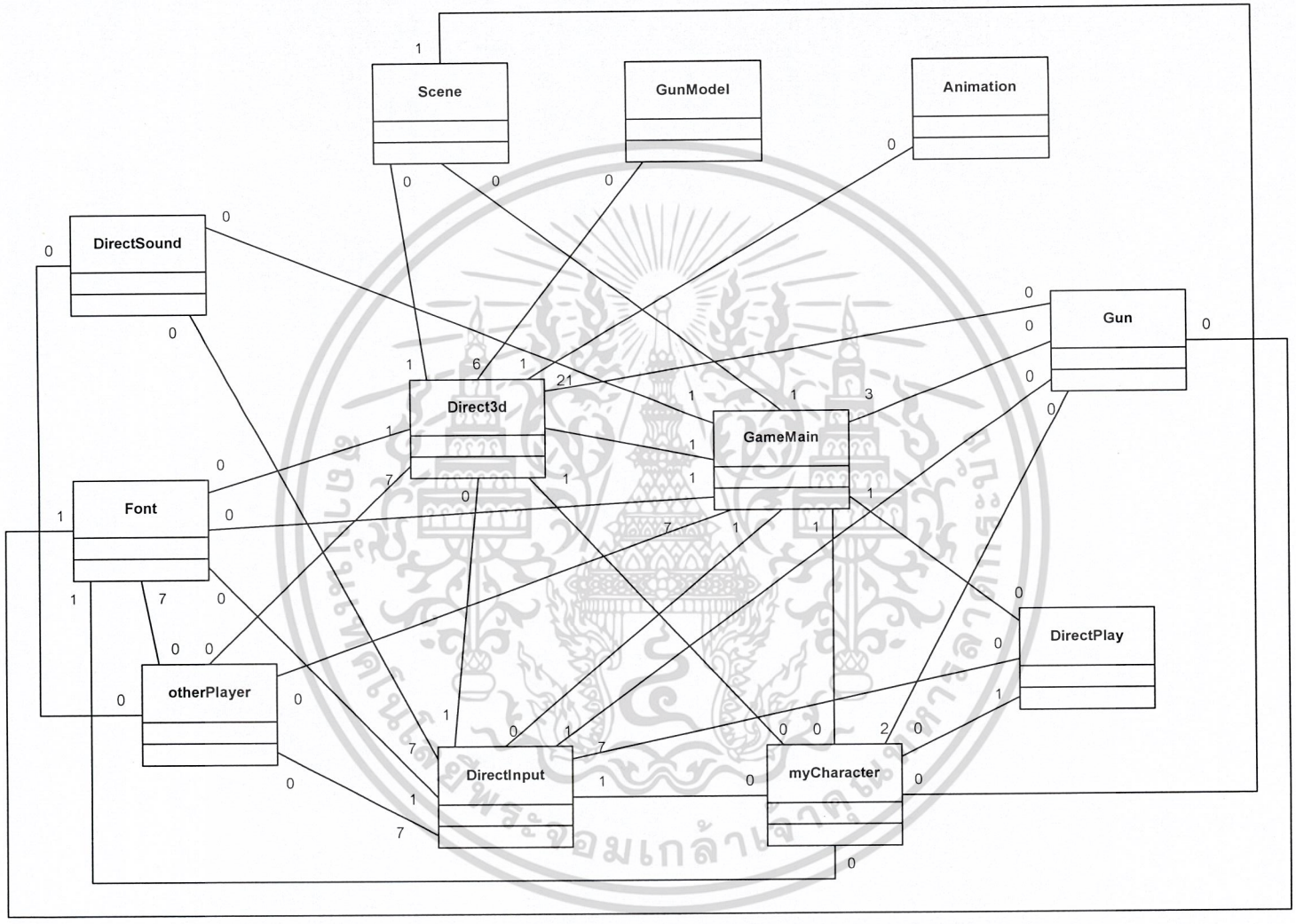
2.2 ถ้ามุมของแนวการยิงที่กระทำกับแกน X อยู่ในช่วงมุม คือยิงโดนตัวของคู่ต่อสู้

ในการตรวจสอบการยิง ระยะทางระหว่างผู้เล่น(ผู้ยิง) กับ คู่ต่อสู้ มีผลกับขอบเขตของมุม (θ , α, β) ที่ใช้ในการตรวจสอบการยิง



รูปที่ 3.7 ระยะทางและขอบเขตของมุมที่ใช้ในการตรวจสอบการยิง

จากรูปเมื่อ S' > S ได้ว่า $\alpha < \beta$ ดังนั้นในการตรวจสอบการยิงจึงต้องมีการคำนวณขอบเขตของมุม(θ , α, β)ที่ใช้ในการตรวจสอบการยิงด้วย



รูปที่ 3.8 Class Diagram Client Game

Attribute และ Method ที่สำคัญของ Class

GameMain
-g_D3d : Direct3d -g_Dinput : DirectInput -g_Dplay : DirectPlay -g_Dsound[6] : DirectSound -g_Scene : Scene -g_Font : Font -g_myCharact : myCharacter -g_Gun[3] : Gun -g_other[7] : otherPlayer
+GameInit(hWnd : HWND , hInstance : HINSTANCE , scenename : char [], team : int) : BOOL +GameRun(hWnd : HWND) : void +GameTerminate() : void

Attribute และ Method ที่สำคัญของ Class GameMain

Direct3d
-g_Scene : Scene -g_Font : Font -g_Gun[21] : Gun -g_GunModel[6] : GunModel -g_myCharact : myCharacter -g_otherPlayer[7] : otherPlayer -g_Animation : Animation
+Direct3dInit(hWnd : HWND) : BOOL +Direct3dInitGeometry(namescene : char []) : void +Direct3dRender() : void +Direct3dTerminate() : void

Attribute และ Method ที่สำคัญของ Class Direct3d

DirectInput
-g_myCharact : myCharacter -g_Gun : Gun -g_Font : Font -g_D3d : Direct3d -g_Dplay : DirectPlay -g_Dsound[6] : DirectSound -g_other[7] : otherPlayer
+DirectInputInit(hWnd : HWND, hInstance : HINSTANCE, scenename : char[]) : BOOL +CheckKeyboard(hWnd : HWND) : void +CheckMouse(hWnd : HWND) : void +DirectInputTerminate() : void

Attribute และ Method ที่สำคัญของ Class DirectInput

DirectPlay
-g_myCharact : myCharacter
+SendToPlaying() : void +SendClientMessage(msgsend : char [500]) : void +SendDropGun(namegun : char [30], x : float , z : float, yangle : float , bullet : int , bis : int) : void +SendGunSound() : void

Attribute และ Method ที่สำคัญของ Class DirectPlay

Attribute และ Method ที่สำคัญของ Class (ต่อ)

DirectSound
+SetupAudio() : HRESULT +SetupSound(pwsFileName : WCHAR*) : HRESULT +Kill() : void +Play(bLoop : bool) : HRESULT +SetFrequencyOfSound(dwFrequency : DWORD) : HRESULT +setPos(fX : float, fY : float, fZ : float) : HRESULT +setSourceSpeed(fX : float, fY : float, fZ : float) : HRESULT +setListenerPos(fX : float, fY : float, fZ : float) : HRESULT +setListenerSpeed(fX : float, fY : float, fZ : float) : HRESULT

Attribute และ Method ที่สำคัญของ Class DirectSound

Scene
-g_myCharact : myCharacter
+InitGeometry(pd3dDevice : LPDIRECT3DDEVICE8 , name : char []) : HRESULT +Cleanup() : void +SetupMatrices(pd3dDevice : LPDIRECT3DDEVICE8 , scale : float, X : float, Y : float, Z : float) : void +Render(pd3dDevice : LPDIRECT3DDEVICE8 , ts : float, tx : float, ty : float, tz : float) : void

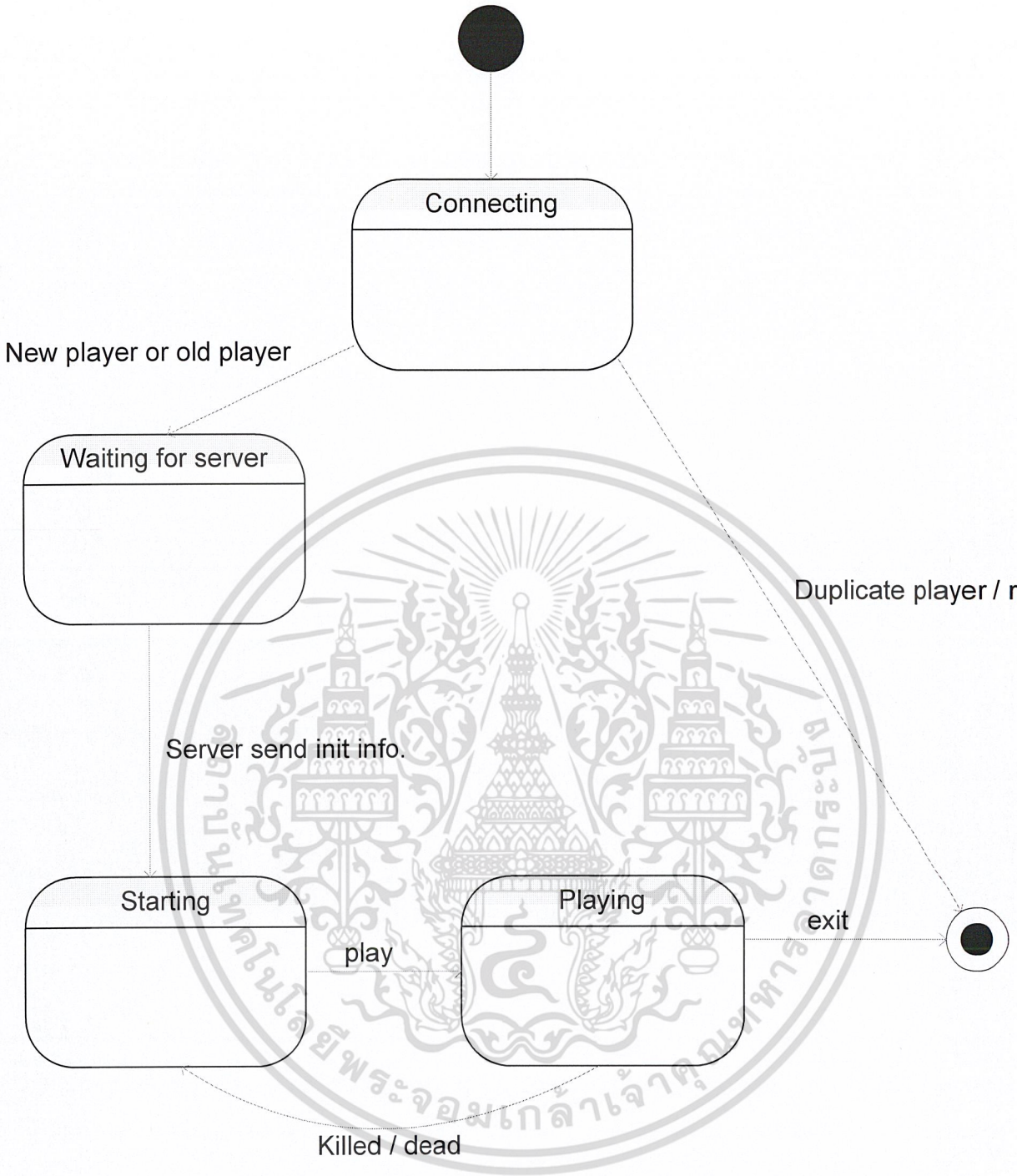
Attribute และ Method ที่สำคัญของ Class Scene

GunModel
+InitGeometry(pd3dDevice : LPDIRECT3DDEVICE8 , name : char []) : void +Cleanup() : void +CleanMaterialsAndTextures() : void +SetupMatrices(pd3dDevice : LPDIRECT3DDEVICE8 , tx : float , ty : float , tz : float , radx : float , rady : float) : void +Render(pd3dDevice : LPDIRECT3DDEVICE8 , tx : float, ty : float, tz : float, radx : float, rady : float) : void

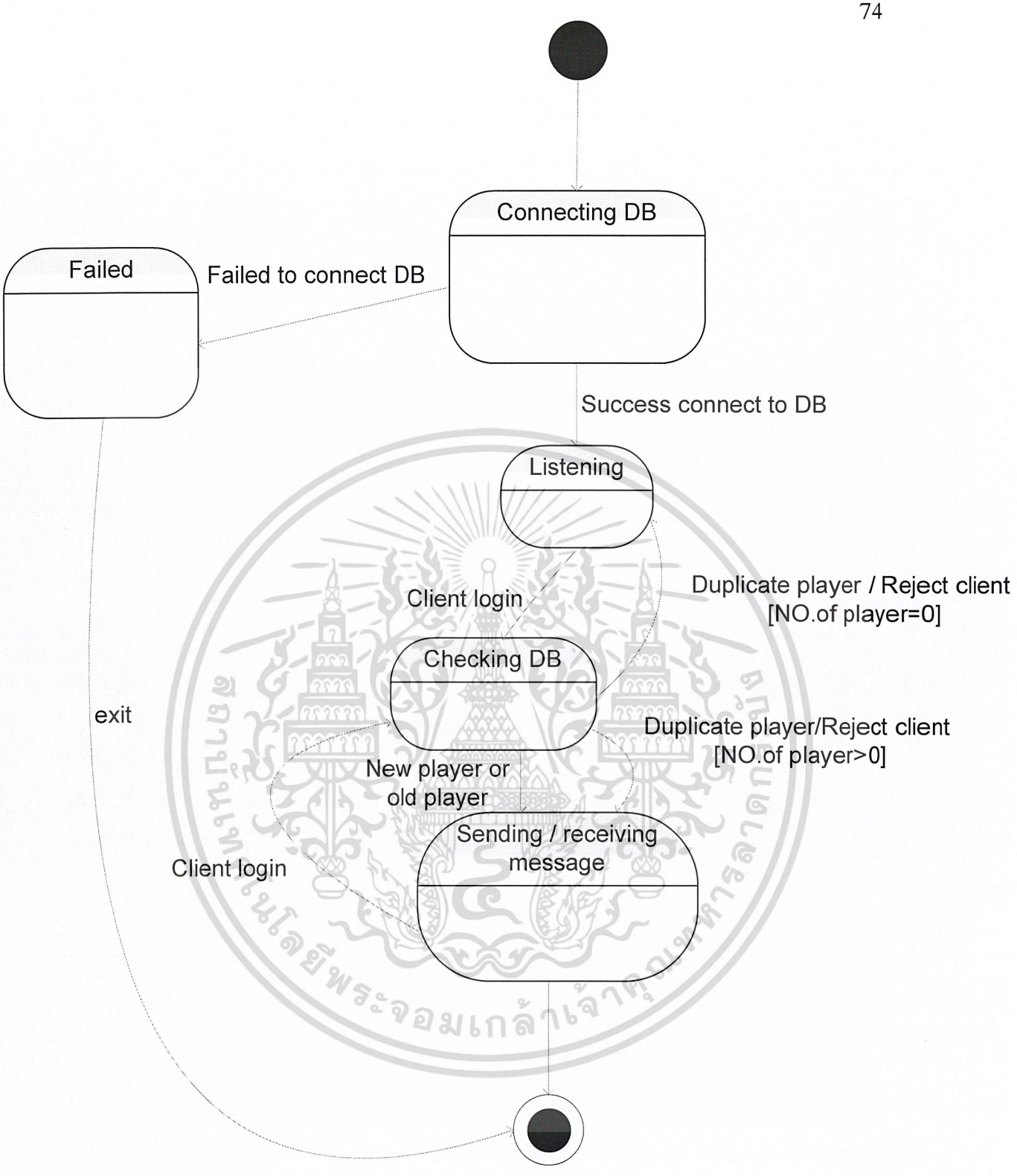
Attribute และ Method ที่สำคัญของ Class GunModel

Font
-g_myCharact : myCharacter -g_Gun : Gun -g_other[7] : otherPlayer
+build_font(g_pd3dDevice : LPDIRECT3DDEVICE8 , twidth : int , theight : int) : void +BeginFont() : void +EndFont() : void +free_font() : void +ShowTextToScreen() : void +Show_font(buffer : char [], top : int, left : int) : void

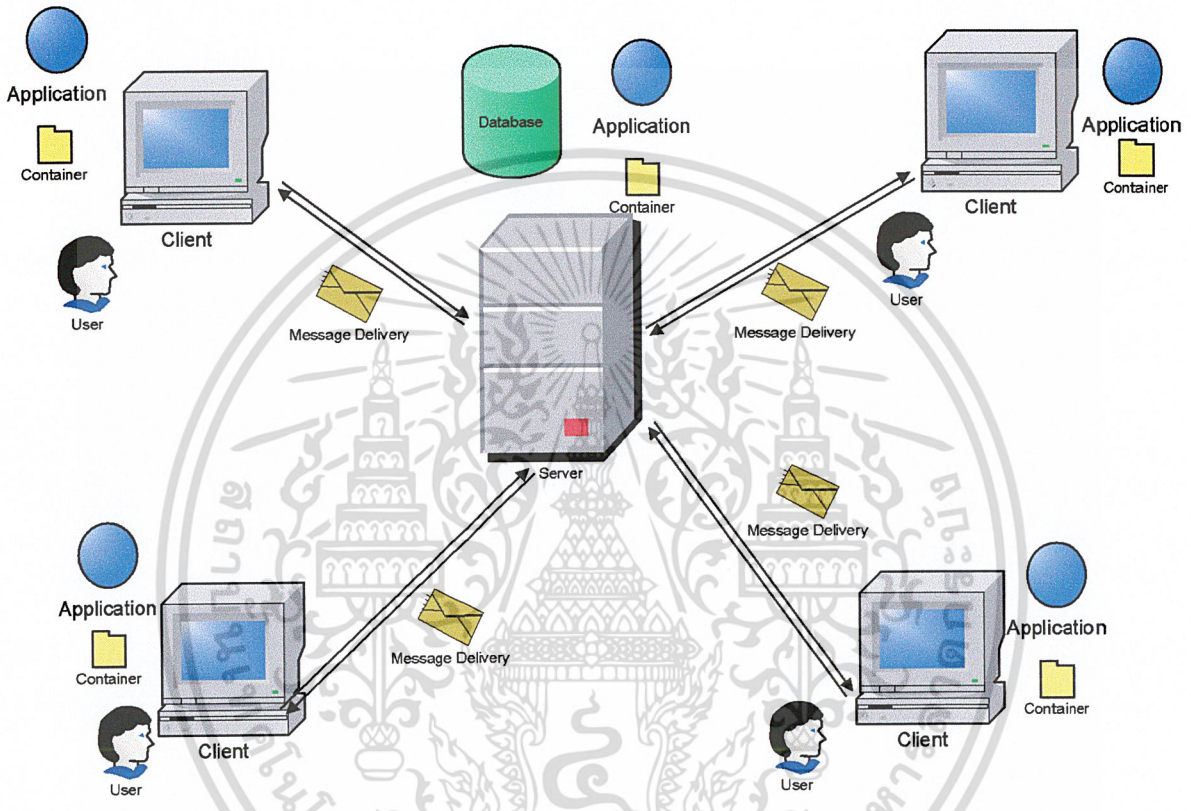
Attribute และ Method ที่สำคัญของ Class Font



รูปที่ 3.9 State Diagram Client Game



รูปที่ 3.10 State Diagram Server Game



รูปที่ 3.11 Diagram แสดงสถาปัตยกรรม

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

การทดลองและการวิเคราะห์ปัญหาที่กล่าวถึงในบทนี้ จะเป็นขั้นตอนการทดสอบและผลการทดสอบที่ได้ในแต่ละขั้น โดยผลการทดสอบนี้ จะถูกนำไปวิเคราะห์ถึงปัญหาและแนวทางในการพัฒนาต่อไปในอนาคต โดยทางผู้จัดทำหวังว่าจะเป็นประโยชน์แก่ผู้ที่จะนำไปศึกษาเพื่อพัฒนาต่อ และเพื่อให้เห็นถึงปัญหา และข้อดีข้อเสียของปัญหานี้โดยที่ผู้ศึกษาไม่จำเป็นต้องทำการทดสอบเพื่อหาปัญหา และนำไปพัฒนาต่ออีกครั้งหนึ่ง

คุณสมบัติของระบบที่นำมาทดสอบ

- ระบบปฏิบัติการ Microsoft Window XP
- ติดตั้ง Microsoft DirectX เวอร์ชั่น 8.0
- เครื่องคอมพิวเตอร์ที่มีซีพียูความเร็ว 1.8 GHz
- หน่วยความจำหลักขนาด 256 Mbytes

ขั้นตอนการดำเนินการทดสอบ

- ทำการติดตั้งตัวโปรแกรมและซอฟท์แวร์ที่จำเป็นต้องใช้
- ทำการรันและประมวลผลภายใต้ระบบที่กำหนด
- ตรวจสอบการใช้คำสั่งของปุ่มกดต่างๆและการทำงานของปุ่มกดต่างๆ
- ตรวจสอบการทำงานของโปรแกรมโดยการใช้เมาส์และคีย์บอร์ด
- ตรวจสอบการการหา Error

การประเมินผล

- หลังจากติดตั้งตัวโปรแกรมแล้วสามารถใช้โปรแกรมได้
- การประมวลผลของโปรแกรมจะต้องทำความเร็วในระดับที่ยอมรับได้
- การตอบโต้ของคอมพิวเตอร์จะต้องเป็นไปตามที่กำหนดไว้
- การใช้คำสั่งของปุ่มกดต่างๆเป็นไปตามรูปแบบ
- การคลิกเมาส์และการกดแป้นของคีย์บอร์ดถูกต้องตามเป้าหมาย
- จำนวน Error ที่เกิดขึ้นต้องไม่มี หรือน้อยที่สุด

ต่อไปจะกล่าวถึงรายละเอียดของขั้นตอนการทดสอบและผลการทดสอบที่ได้

4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและซอฟท์แวร์ที่จำเป็น

- ติดตั้ง ตัวโปรแกรมได้โดยคลิกไฟล์ “SETUP.EXE”
- ตัวโปรแกรมและซอฟท์แวร์จะถูกทำการติดตั้ง

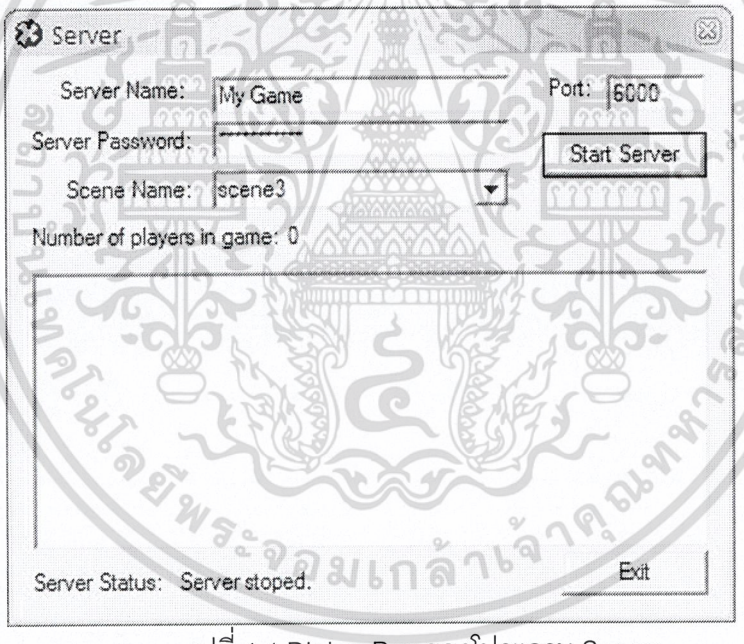
- ผู้ใช้ทำการรันโปรแกรม Server ได้โดยคลิกที่ไฟล์ “Server.EXE”
- ผู้ใช้ทำการรันโปรแกรม Game ได้โดยคลิกที่ไฟล์ “Client.EXE”

ผลการทดสอบ

- ติดตั้งโปรแกรมและ ซอฟท์แวร์ที่จำเป็นลงบนเครื่องคอมพิวเตอร์ได้
- ผู้ใช้สามารถรันโปรแกรมได้
- เนื้อที่ของ ฮาร์ดดิสก์ถูกใช้ไปประมาณ 20 Mbytes

4.2 ขั้นตอนการทดสอบการรันและประมวลผลภายใต้ระบบที่กำหนด

4.2.1 รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนด โดยดับเบิ้ลคลิกที่ไฟล์ “Server.EXE”



รูปที่ 4.1 Dialog Box ของโปรแกรม Server

ผลการทดสอบ

- สามารถรันโปรแกรมได้บนระบบที่กำหนด
- ความเร็วในการแสดงภาพหน้าจอแรกอยู่ในขั้นดี
- แสดง Dialog box ตามที่ได้ออกแบบเอาไว้

4.2.1.1 ขั้นตอนการคลิกเลือกปุ่ม Exit บน Dialog box

ผลการทดสอบ

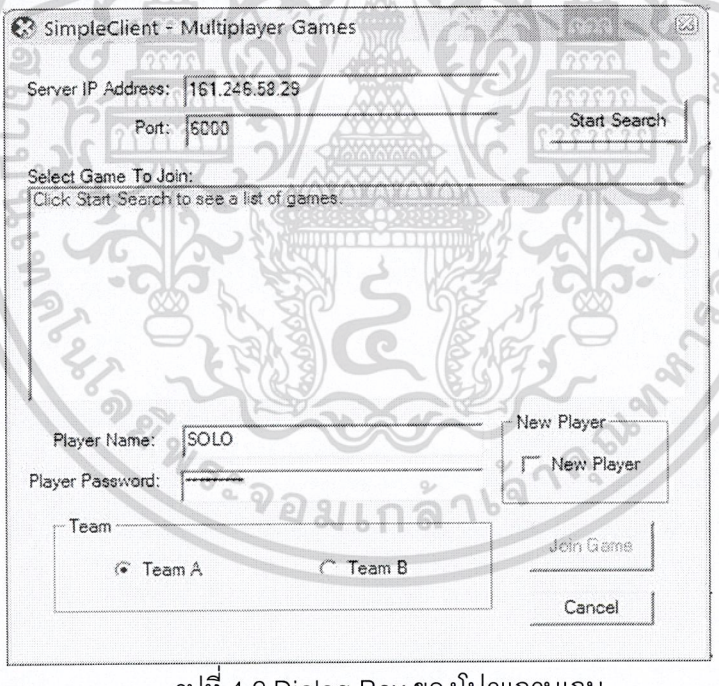
- ตัวโปรแกรมปิด
- โปรแกรมคืนค่าทรัพยากรให้สู่หน่วยความจำของระบบ
- ความเร็วที่ได้จากการออกจากเกมอยู่ในขั้นดี

4.2.1.2 ขั้นตอนการคลิกเลือกปุ่ม Start Server บน Dialog box

ผลการทดสอบ

- สามารถให้เครื่อง client เข้ามา connect ได้

4.2.2 รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนด โดยดับเบิ้ลคลิกที่ไฟล์ “Client.EXE”



รูปที่ 4.2 Dialog Box ของโปรแกรมเกม

ผลการทดสอบ

- สามารถรันโปรแกรมได้บนระบบที่กำหนด
- ความเร็วในการแสดงภาพหน้าจอแรก อยู่ในขั้นดี
- แสดง Dialog box ตามที่ได้ออกแบบเอาไว้

4.2.2.1 ขั้นตอนการคลิกเลือกปุ่ม Cancel บน Dialog box

ผลการทดสอบ

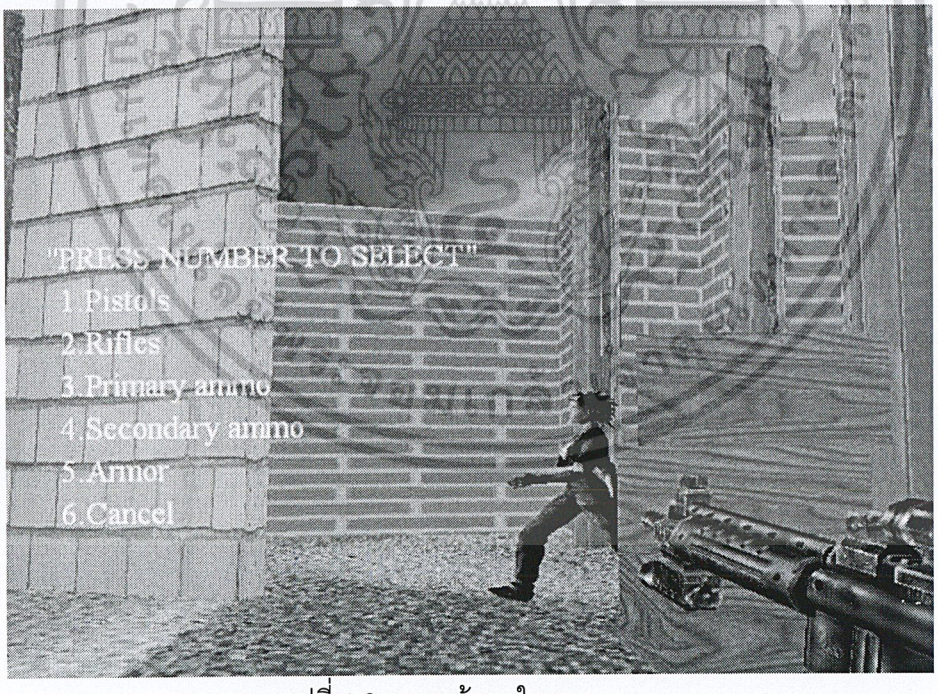
- ตัวโปรแกรมปิด
- โปรแกรมคืนค่าทรัพยากรให้สู่หน่วยความจำของระบบ
- ความเร็วที่ได้จากการออกจากเกมอยู่ในขั้นดี

4.2.2.2 ขั้นตอนการคลิกเลือกปุ่ม Start Search บน Dialog box

ผลการทดสอบ

- สามารถค้นหาเครื่องที่เป็น Server ตาม Server IP Address และ Port ที่กำหนด
- ความเร็วในการค้นหาอยู่ในขั้นดี

4.2.2.3 ขั้นตอนการคลิกเลือกปุ่ม Join Game บน Dialog box จะได้หน้าจอดังนี้



รูปที่ 4.3 ภาพหน้าจอในเกม

ผลการทดสอบ

- สามารถ connect กับเครื่อง Server ได้
- สามารถรันเกมได้
- สามารถรับ message จากเครื่อง Server และประมวลผลได้

4.3 ขั้นตอนการทดสอบการควบคุมตัวละคร

การทดสอบ วิธีการทดสอบ และผลการทดสอบจะแสดงได้ดังตารางที่ 4.1

การทดสอบ	วิธีการทดสอบ	ผลการทดสอบ
การเดินหน้า	กดปุ่ม W	เกมแสดงผลของการเดินหน้า
การถอยหลัง	กดปุ่ม S	เกมแสดงผลของการถอยหลัง
การสไลด์ไปทางซ้าย	กดปุ่ม A	เกมแสดงผลของการสไลด์ไปทางซ้าย
การสไลด์ไปทางขวา	กดปุ่ม D	เกมแสดงผลของการสไลด์ไปทางขวา
การกระโดด	กดปุ่ม Space Bar	เกมแสดงผลของการกระโดด
การย่อตัว	กดปุ่ม Left Ctrl	เกมแสดงผลของการย่อตัว
การย่อตัวเดินหน้า	กดปุ่ม Left Ctrl + W	เกมแสดงผลของการย่อตัวเดินหน้า
การย่อตัวถอยหลัง	กดปุ่ม Left Ctrl + S	เกมแสดงผลของการย่อตัวถอยหลัง
การย่อตัวสไลด์ไปทางซ้าย	กดปุ่ม Left Ctrl + A	เกมแสดงผลของการย่อตัวสไลด์ไปทางซ้าย
การย่อตัวสไลด์ไปทางขวา	กดปุ่ม Left Ctrl + D	เกมแสดงผลของการย่อตัวสไลด์ไปทางขวา
การย่อตัวกระโดด	กดปุ่ม Left Ctrl + Space bar	เกมแสดงผลของการย่อตัวกระโดด
การยืนขึ้น	ปล่อยปุ่ม Left Ctrl	เกมแสดงผลของการยืนขึ้น
การยิง	เล็งคู่ต่อสู้ แล้วคลิกเมาท์ข้าง	จำนวนกระสุนลดลง และพลัง

	ซ้าย	ของคู่ต่อสู้ลดลง
การทิ้งปืน	กดปุ่ม G	เกมแสดงปืนอยู่บนพื้น
การเก็บปืน	ทิ้งปืนให้หมด แล้วเดินผ่านปืนที่อยู่บนพื้น	ปืนบนพื้นหายไป และมาแสดงในหน้าจอผู้เล่น
การแสดงเมนู	กดปุ่ม B	แสดงเมนูตามที่กำหนด
การซื้อของ	กดปุ่ม B แล้วกดเลือกซื้อของตามเมนูที่ปรากฏ	สามารถซื้อของได้ โดยจำนวนเงินลดลง และได้ของตามที่ซื้อ
การเปลี่ยน Magazine	กดยิงให้กระสุนลดลง จากนั้นกดปุ่ม R	กระสุนใน Magazine ปัจจุบันเต็ม และจำนวนกระสุนสำรองลดลง
การสลับปืน	ซื้อปืนให้ครบสองกระบอก จากนั้นกดปุ่ม Q	สามารถสลับปืนได้

ตาราง 4.1 ขั้นตอนการทดสอบการควบคุมตัวละคร

4.4 ขั้นตอนการทดสอบการส่งข้อมูล

- เคลื่อนไหวผู้เล่นในเกม
- เล็งแล้วยิงคู่ต่อสู้

ผลการทดสอบ

- การแสดงผลของตัวผู้เล่นที่ปรากฏในเครื่องอื่นถูกต้อง
- Server ประมวลผล และส่งข้อมูลได้ถูกต้อง

4.5 ขั้นตอนการทดสอบการหาข้อผิดพลาดของโปรแกรม

ในขั้นตอนการทดสอบเพื่อหาข้อผิดพลาดจะทำการทดสอบโดยการเล่นเกมหลายๆรอบ เพื่อหาข้อผิดพลาดและจากการทดสอบพบว่า ข้อผิดพลาดที่พบจะได้แก่

- พบข้อผิดพลาดในการตรวจสอบการชนในบางส่วนของฉาก
- การตรวจสอบการยิงยังไม่แม่นยำในบางครั้ง
- การแสดงผลของปืนที่ถูกเก็บยังไม่ถูกต้องในบางครั้ง

4.6 ปัญหาและการวิเคราะห์

จากผลการทดสอบโปรแกรมทำให้ทราบถึงปัญหาดังนี้

4.6.1 **ความล่าช้าในการประมวลผลในฝั่ง** Client

เมื่อมีผู้เล่นเป็นจำนวนมากทำให้ต้องมีการแสดงภาพ Animation ของผู้เล่นอื่นมาก ทำให้การเคลื่อนไหวของผู้เล่นช้าลง

สาเหตุของความช้าในการประมวลผลอาจเนื่องมาจากการเขียนโปรแกรมที่ไม่กระชับ มีการวนลูปตรวจสอบค่ามากเกินไป และการแสดงภาพ Animation ที่ละเอียดเกินไป

4.6.2 **ความล่าช้าในการส่งข้อมูลของ** Server

เมื่อมีผู้เล่นเป็นจำนวนมากทำให้ต้องมีการส่งข้อมูลบนเครือข่ายมาก ทำให้มีความล่าช้าในการส่งข้อมูลของ Server

สาเหตุของความล่าช้าในการส่งข้อมูลเกิดจากการการส่งข้อมูลของผู้เล่นจำนวนมากบนเครือข่าย

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

การทำงานโครงงานนี้มีการวางแผนขั้นตอนการทำงานเป็นอย่างดี แต่ถึงแม้จะได้วางแผนแต่ละขั้นตอนไว้อย่างดี ก็ยังมีปัญหากับขั้นตอนบางขั้นตอน ซึ่งบางขั้นตอนเกิดจากการศึกษาการใช้ซอฟท์แวร์กราฟฟิกใหม่ๆ เนื่องจากต้องใช้ประสบการณ์และระยะเวลาพอสมควร ถึงจะเข้าใจและสามารถทำงานกับซอฟท์แวร์แต่ละตัวได้เป็นอย่างดี บางขั้นตอนเกิดจากความล่าช้าในการคิดระเบียบวิธีการโปรแกรมมิ่ง ซึ่งจำต้องศึกษาจากบทความภาษาอังกฤษตามโฮมเพจต่างๆและขอคำแนะนำจากบุคคลผู้ที่มีความรู้ทางด้านนี้

โดยขั้นตอนต่างๆสรุปได้ดังนี้

- ขั้นตอนการออกแบบระบบเกม
- ขั้นตอนการเขียนโปรแกรม
- ขั้นตอนการสร้างตัวละครและฉาก
- ขั้นตอนการสร้างท่าทาง การเคลื่อนไหวของตัวละคร

5.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบเป็นขั้นตอนที่รวดเร็วที่สุด เนื่องมาจากผู้วิจัยมีความคิดต่างๆของแต่ละคนไว้อยู่แล้ว ซึ่งก็เกิดจากประสบการณ์ซึ่งเคยเล่นเกม และมีแนวคิดของเกมที่ตัวเองชื่นชอบ จึงนำมารวบรวมและสรุปเป็นรูปแบบของเกมในงานวิจัย

5.2 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้สามารถทำควบคู่ไปได้กับขั้นตอนการสร้างตัวละครและฉาก โดยขั้นตอนนี้ใช้เวลานานที่สุดในงานวิจัย เนื่องจากเป็นเรื่องใหม่ซึ่งผู้วิจัยต้องทำการศึกษาค้นคว้านานพอสมควร แล้วจึงลงมือเขียนโปรแกรม การทำงานในส่วนนี้มีซอฟท์แวร์ที่ใช้ในการพัฒนาดังนี้

- Microsoft Visual C++ เวอร์ชั่น 6.0
- Microsoft Direct X SDK เวอร์ชั่น 8.0

5.3 ขั้นตอนการสร้างตัวละครและฉาก

ขั้นตอนนี้เป็นขั้นตอนที่ใช้ระยะเวลาหลังจากทำการเขียนโปรแกรมไปได้ระยะหนึ่งเนื่องจากการสร้างตัวละครเป็นงานที่ใช้เวลานาน และปริมาณมาก ปัญหาส่วนใหญ่อยู่ที่การทำงานกับซอฟท์แวร์กราฟฟิกต่างๆ ซึ่งต้องใช้เวลาศึกษาพอสมควร แต่หลังจากได้ศึกษาและชินกับการทำงาน ทำให้การทำงานเป็นไปอย่างราบรื่น ขั้นตอนนี้ใช้ Application ดังนี้

- 3Dstudio Max เวอร์ชั่น 5.0
- Adobe Photoshop เวอร์ชั่น 7.0

3Dstudio Max เป็นซอฟท์แวร์ที่ใช้ออกแบบภาพสามมิติต่างๆที่ใช้ในการสร้างเกม รวมถึงปืนสำหรับผู้เล่น

Adobe Photoshop เป็นซอฟท์แวร์ใช้สำหรับทำการตัดต่อภาพเพื่อนำมาใช้ตกแต่งและตัดต่อภาพของตัวละคร และภาพฉากในลักษณะต่างๆ

5.4 ขั้นตอนการสร้างท่าทาง การเคลื่อนไหวของตัวละคร

ขั้นตอนนี้เป็นขั้นตอนที่สามารถทำได้หลังจากสร้างตัวละครเสร็จแล้ว ขั้นตอนนี้เราจะนำตัวละครที่ได้มาสร้างท่าทางเคลื่อนไหวเพิ่ม ซึ่งเป็นงานที่ใช้เวลานาน และปัญหาที่เกิดส่วนใหญ่อยู่ที่การทำงานกับซอฟท์แวร์กราฟฟิกต่างๆซึ่งต้องใช้เวลาศึกษาค้นคว้าพอสมควร แต่หลังจากได้ศึกษาและชินกับการทำงาน ทำให้การทำงานเป็นไปอย่างราบรื่น โดยในขั้นตอนนี้ใช้ Application ในส่วนของ Character Studio ซึ่งอยู่ในโปรแกรม 3Dstudio MAX version 5.0

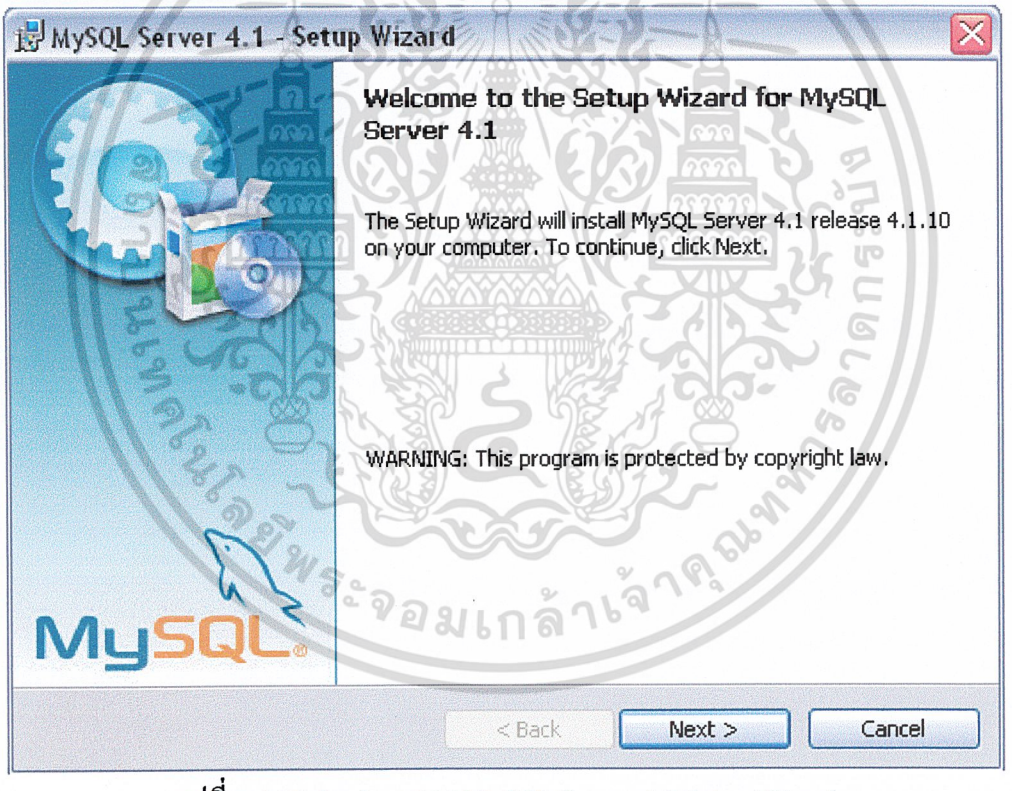
สรุปแล้วในการทำงานวิจัยเพื่อสร้างเกมโดยตอบโต้กับคอมพิวเตอร์ การทำงานต่างๆปัญหาที่เกิดขึ้นตลอดจนการได้ติดต่อกับผู้ผลิตเกมในต่างประเทศ เพื่อทำการศึกษานี้ ทำให้ผู้วิจัยทราบถึงแนวทางในการผลิตเกมในตลาดซอฟท์แวร์ไทย เกมในปัจจุบันบางเกมคนไทยเป็นผู้ผลิต แต่ไม่สามารถบอกได้ว่าเป็นคนไทย ก็เนื่องมาจากบริษัทต่างๆที่ถือลิขสิทธิ์เป็นบริษัทต่างประเทศ ดังนั้นการที่คนไทยจะก่อตั้งบริษัทซอฟท์แวร์ขึ้นก็มีความเป็นไปได้ แต่ปัญหาที่สำคัญที่สุดในตลาดซอฟท์แวร์ไทยก็คือปัญหาละเมิดลิขสิทธิ์ซอฟท์แวร์ ซึ่งเป็นสาเหตุสำคัญที่ทำให้คนไทยไม่กล้าที่จะผลิตซอฟท์แวร์คุณภาพออกมาแข่งขันกัน และปัญหาอีกอย่างในตลาดซอฟท์แวร์เกมก็คือการให้ความสนใจในวิธีการผลิตของคนไทยน้อยเกินไป ในขณะเดียวกันก็บริโภคหรือให้ความสนใจในเนื้อหาของเกมมาก ทำให้ความรู้ในการผลิตยังไม่สามารถทำได้เทียบเท่ากับชาวต่างชาติซึ่งเป็นชาติผู้ผลิต ดังนั้นโครงงานนี้น่าจะเป็นแนวทางหนึ่งที่จะแนะนำให้ผู้สนใจได้นำไปศึกษาและผลิตเกมที่สามารถนำออกมาจำหน่ายได้จริงต่อไป

ภาคผนวก ก
คู่มือการติดตั้ง

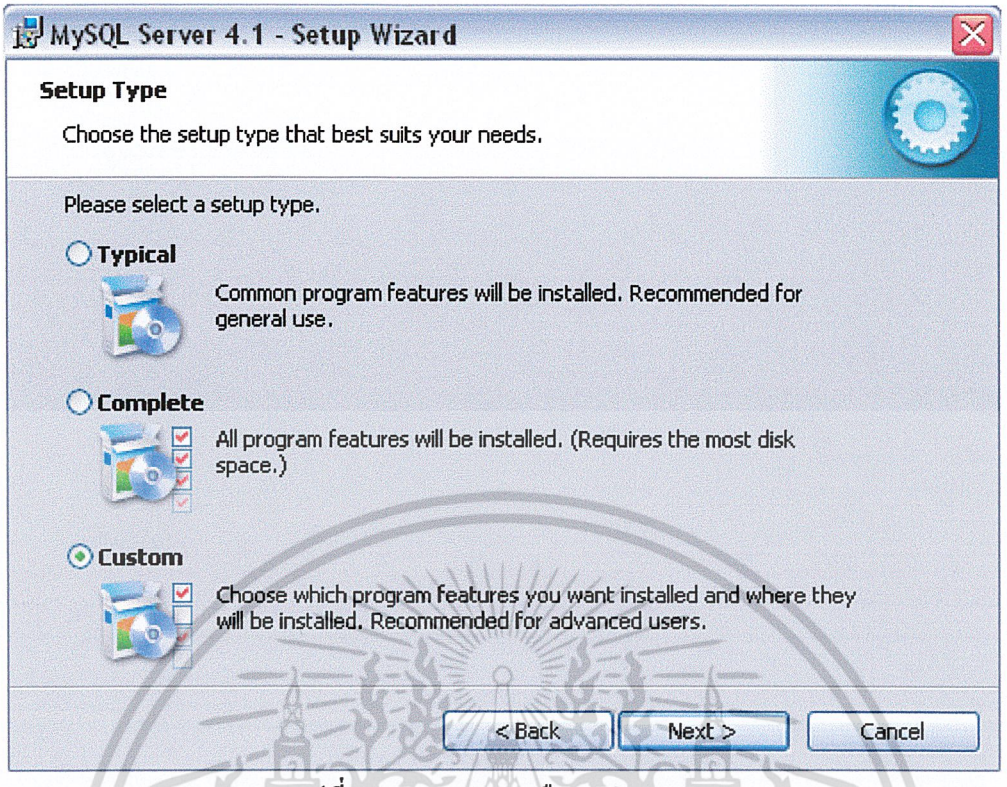
1. ติดตั้ง Microsoft directX version 8.0 ขึ้นไป
2. สำหรับเครื่องที่เป็น server ต้องติดตั้ง MySQL Server 4.1 และติดตั้ง tool ชื่อ SQLyog 4.0 เพื่อความสะดวกในการสร้าง database และ table

การติดตั้ง MySQL Server 4.1

ดับเบิ้ลคลิกที่ไฟล์ setup จะปรากฏ MySQL Server 4.1 Setup Wizard ดังรูป แล้วกดปุ่ม next

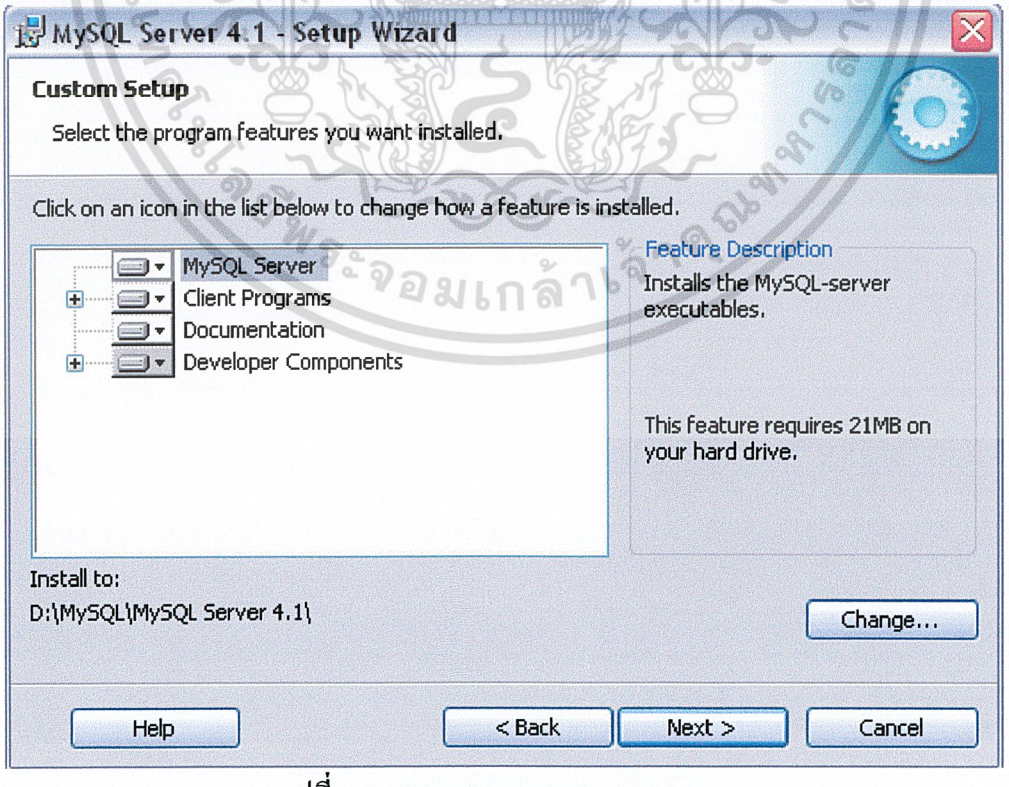


รูปที่ ก.1 Dialog box ของ MySQL Server 4.1 Setup Wizard



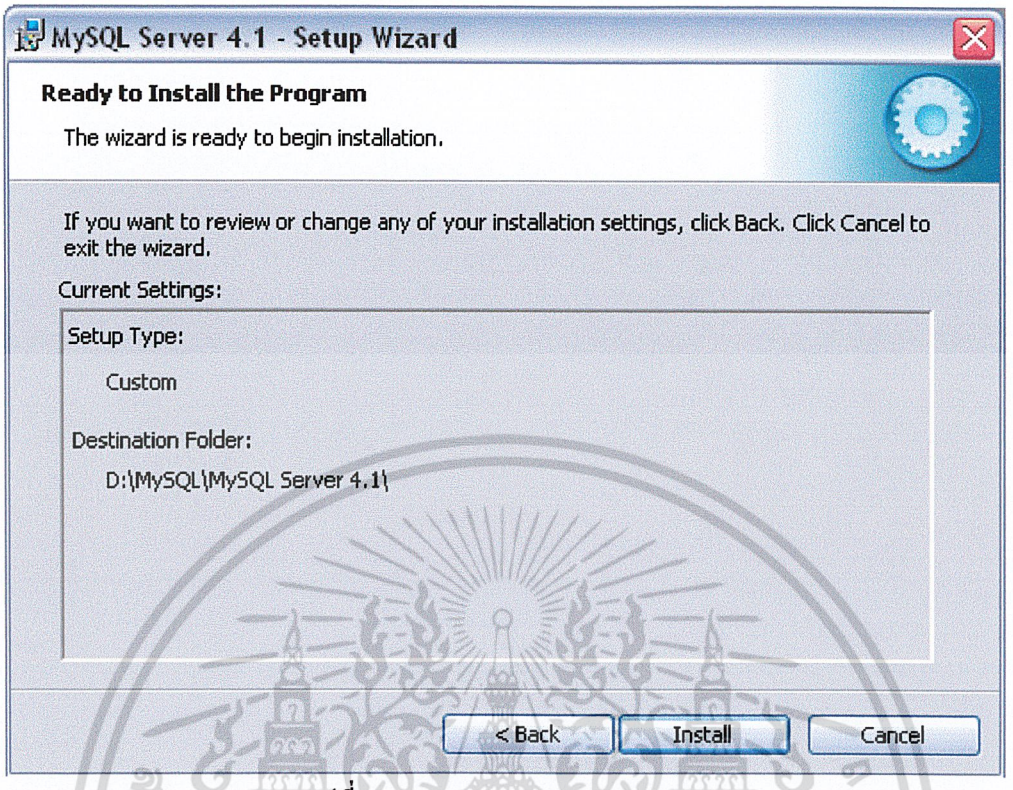
รูปที่ ก.2 Dialog box เลือก Setup type

เลือก ประเภทการติดตั้งแบบ custom แล้วกดปุ่ม next



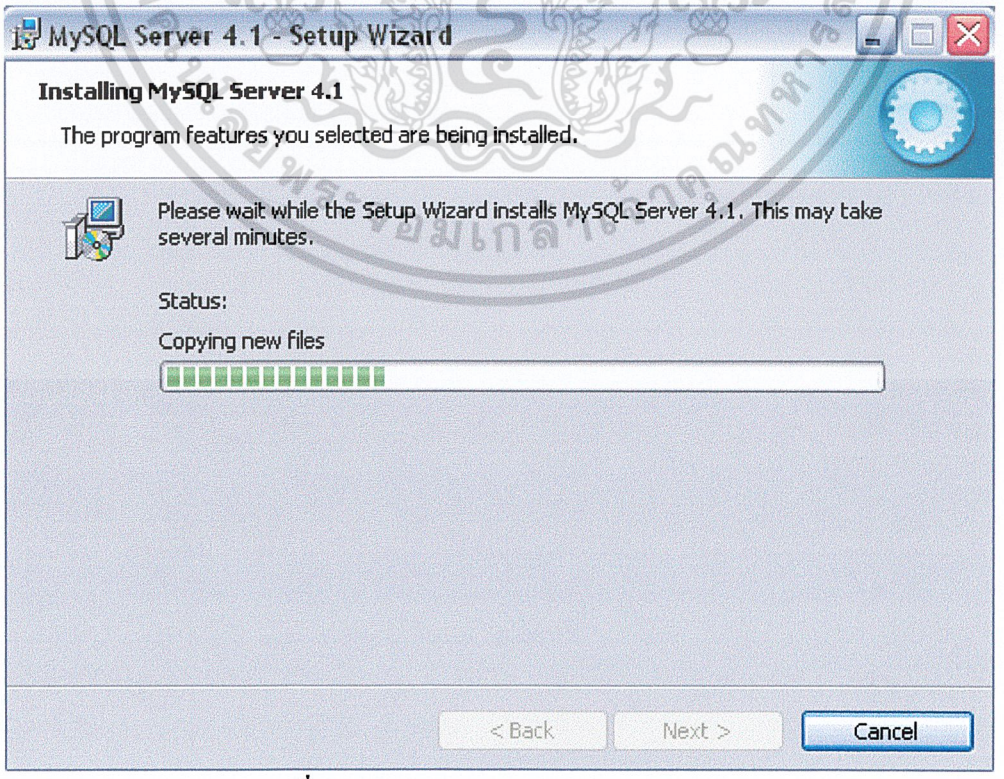
รูปที่ ก.3 Dialog box ของ Custom Setup

เลือกคุณสมบัติต่างๆที่ต้องการติดตั้งดังรูป และเลือก path ที่ต้องการ แล้วกดปุ่ม next

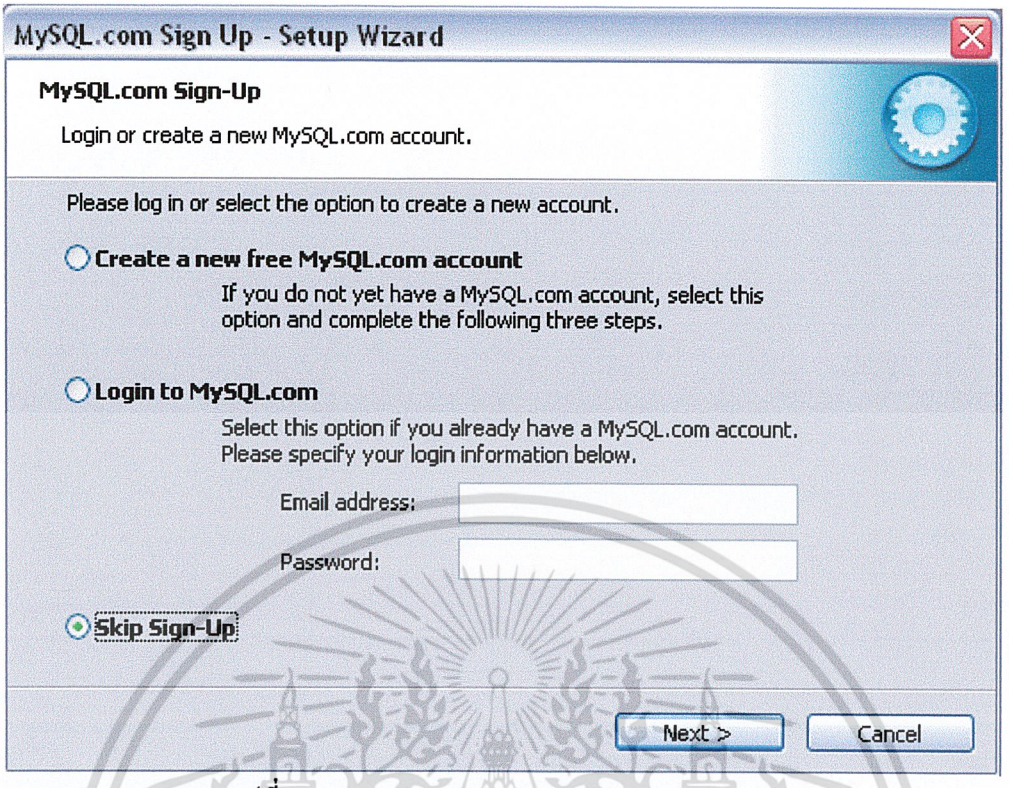


รูปที่ ก.4 Dialog box ของการ install

กดปุ่ม install

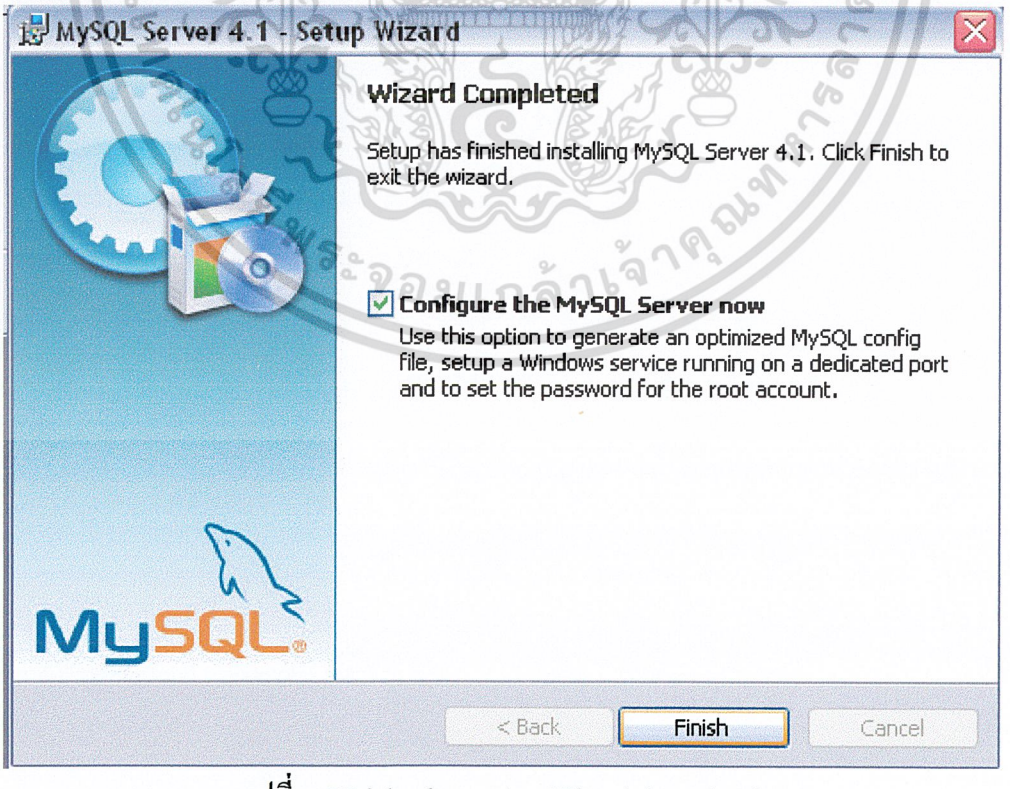


รูปที่ ก.5 Dialog box ขณะทำการ copy file



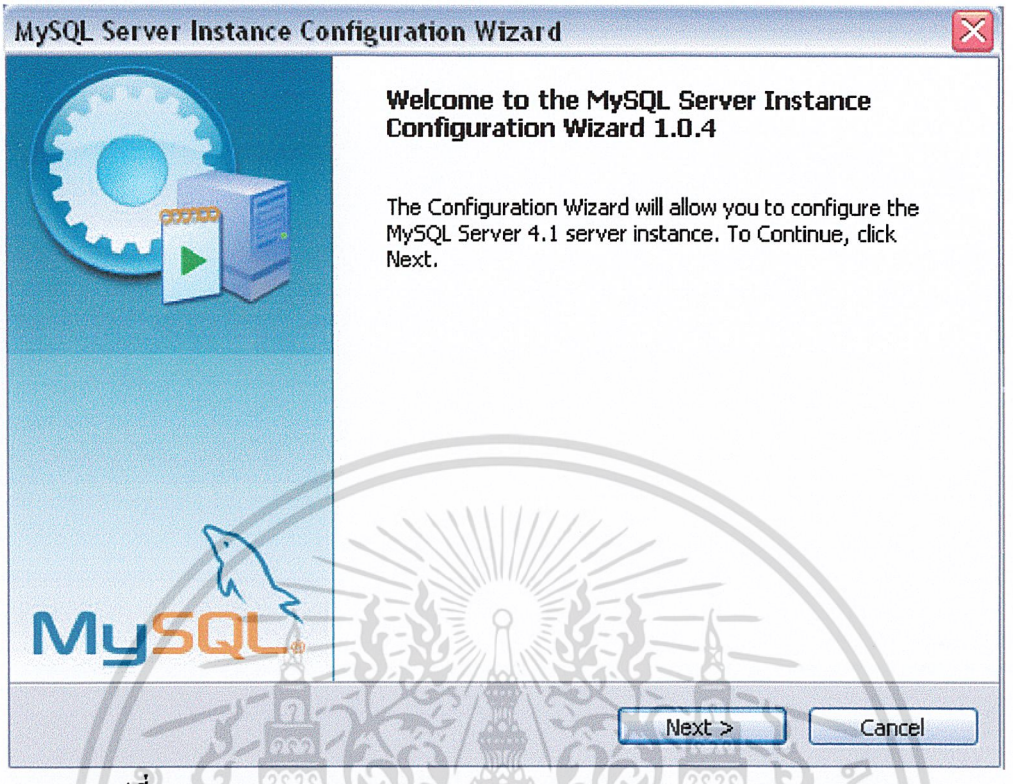
รูปที่ ก.6 Dialog box ของ MySQL.com Sign-Up

เลือก skip Sign-Up แล้วกดปุ่ม next



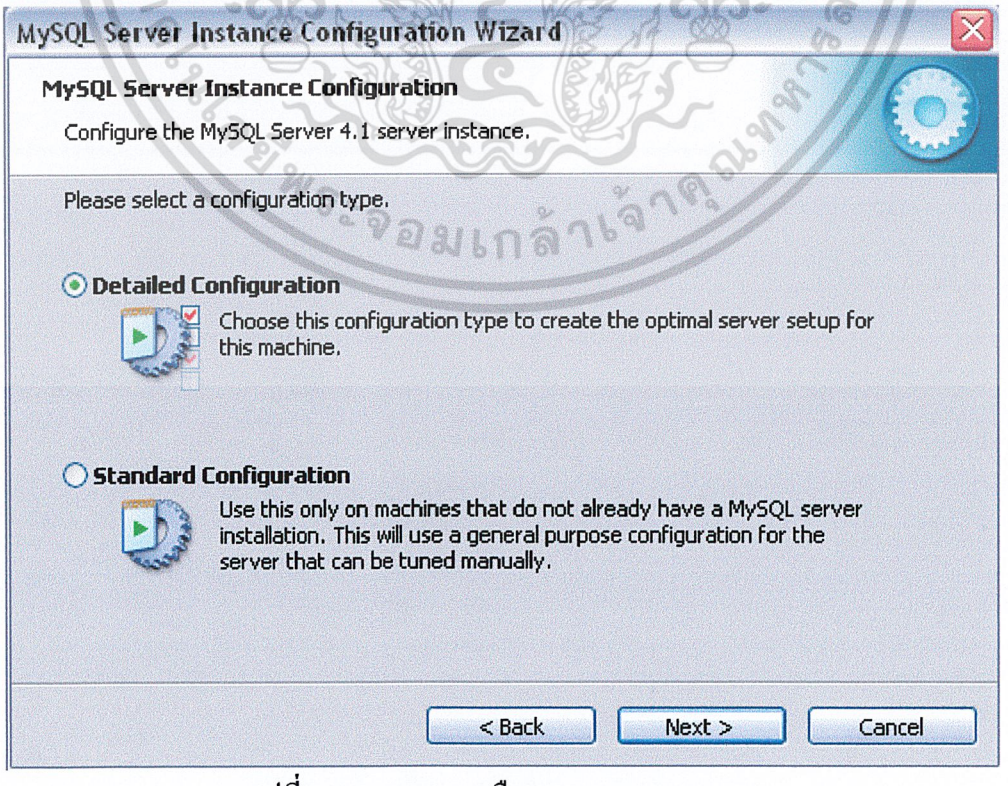
รูปที่ ก.7 Dialog box แสดง Wizard Completed

กดปุ่ม finish แล้วรอจนกว่าจะมี dialog box “MySQL Server Instance Configuration Wizard” ขึ้นมา



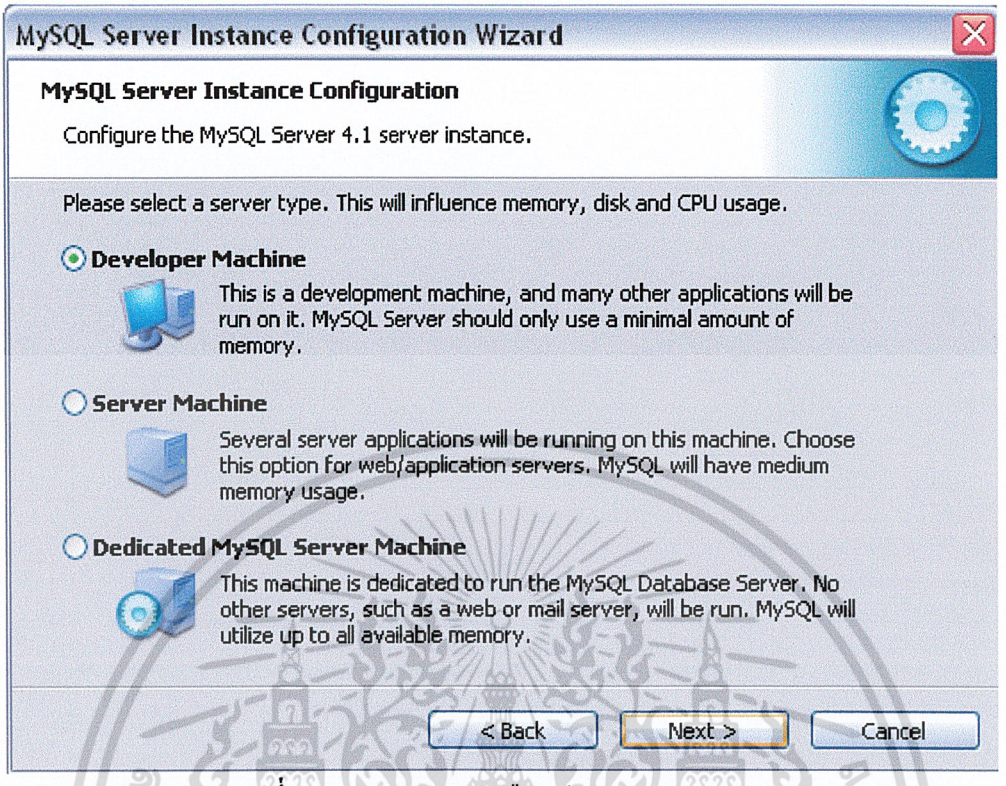
รูปที่ ก.8 Dialog box ของ MySQL Server Instance Configuration Wizard

กดปุ่ม next



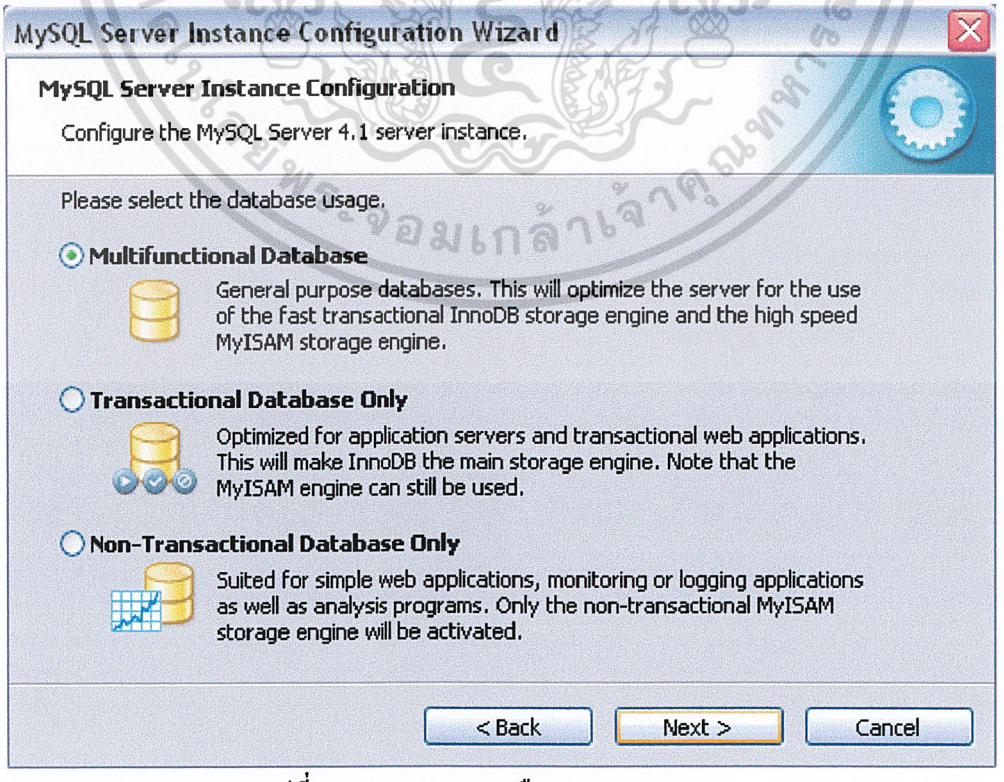
รูปที่ ก.9 Dialog box เลือก configuration type

เลือก configuration แบบ detailed แล้วกดปุ่ม next



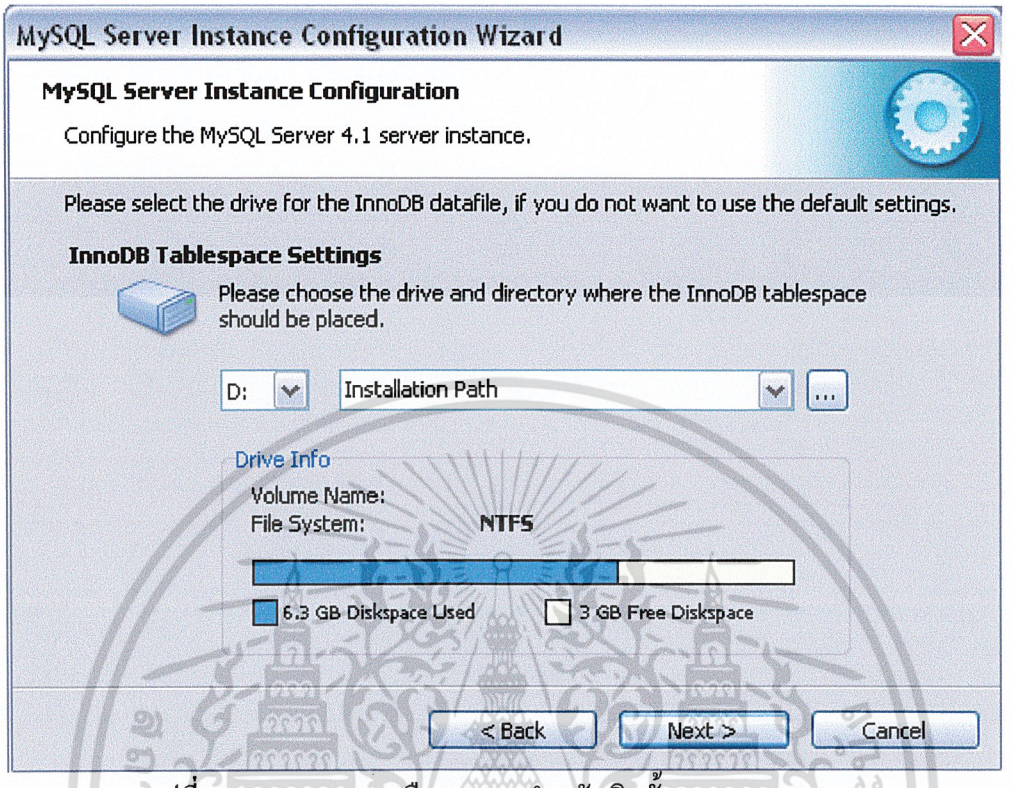
รูปที่ ก.10 Dialog box เลือกประเภทของ server

เลือก server แบบ Developer Machine แล้วกดปุ่ม next



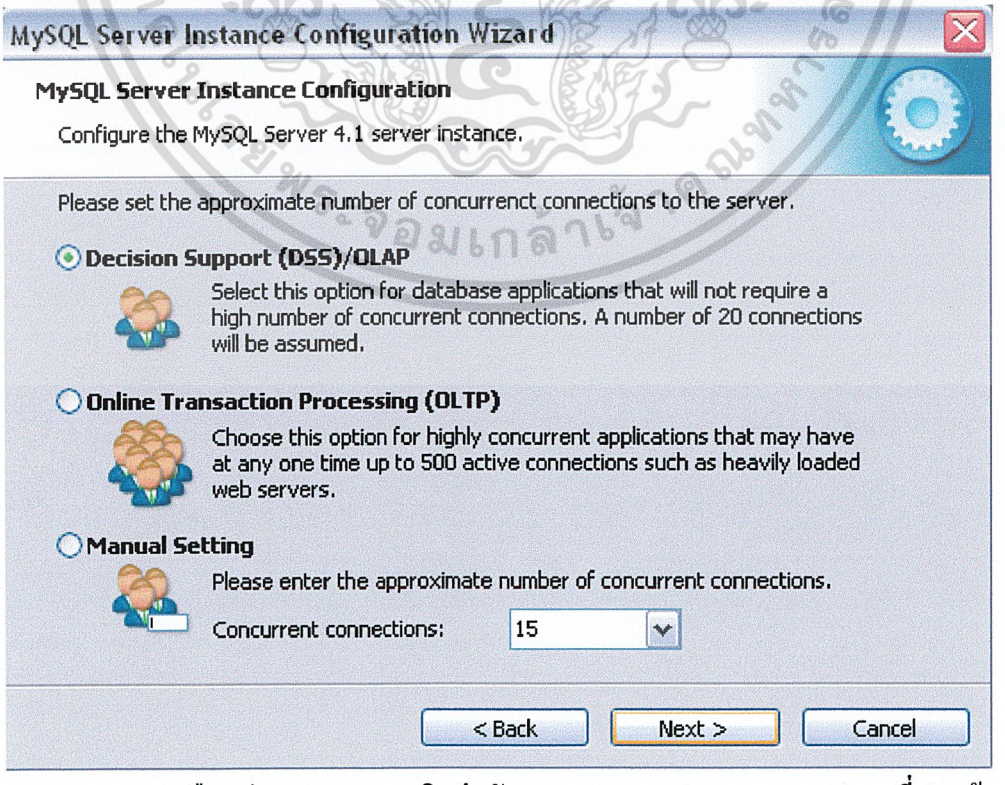
รูปที่ ก.11 Dialog box เลือก database usage

เลือก database usage แบบ Multifunctional Database แล้วกดปุ่ม next



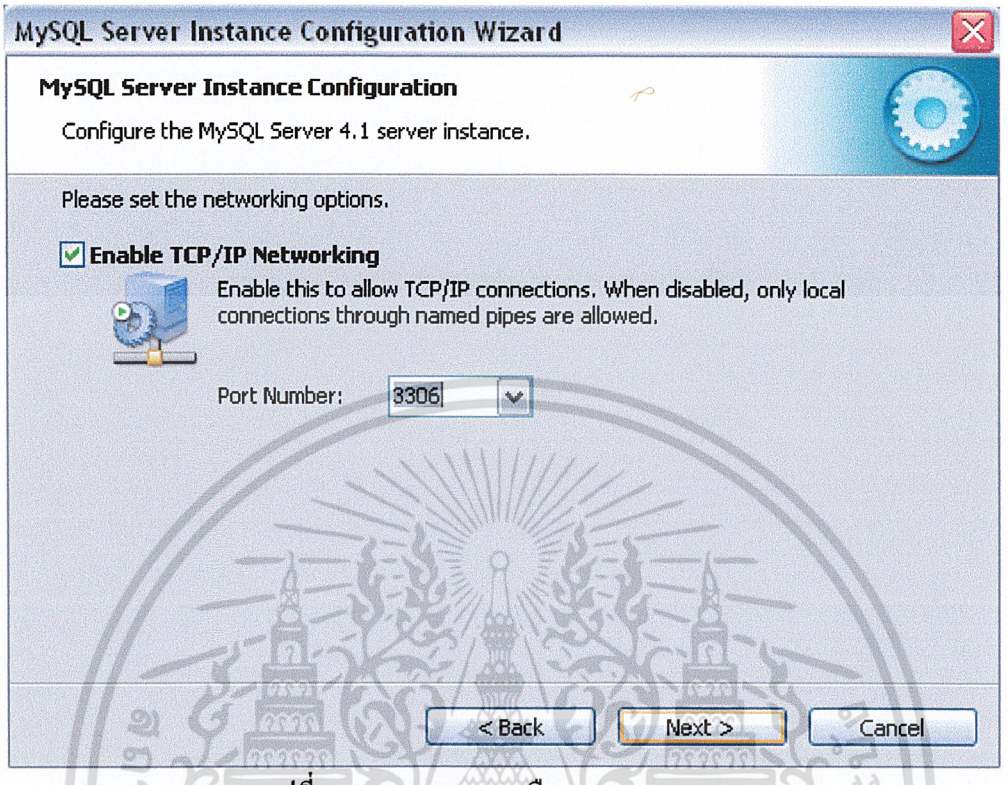
รูปที่ ก.12 Dialog box เลือก drive สำหรับติดตั้ง InnoDB datafile

เลือก drive สำหรับติดตั้ง InnoDB datafile แล้วกดปุ่ม next



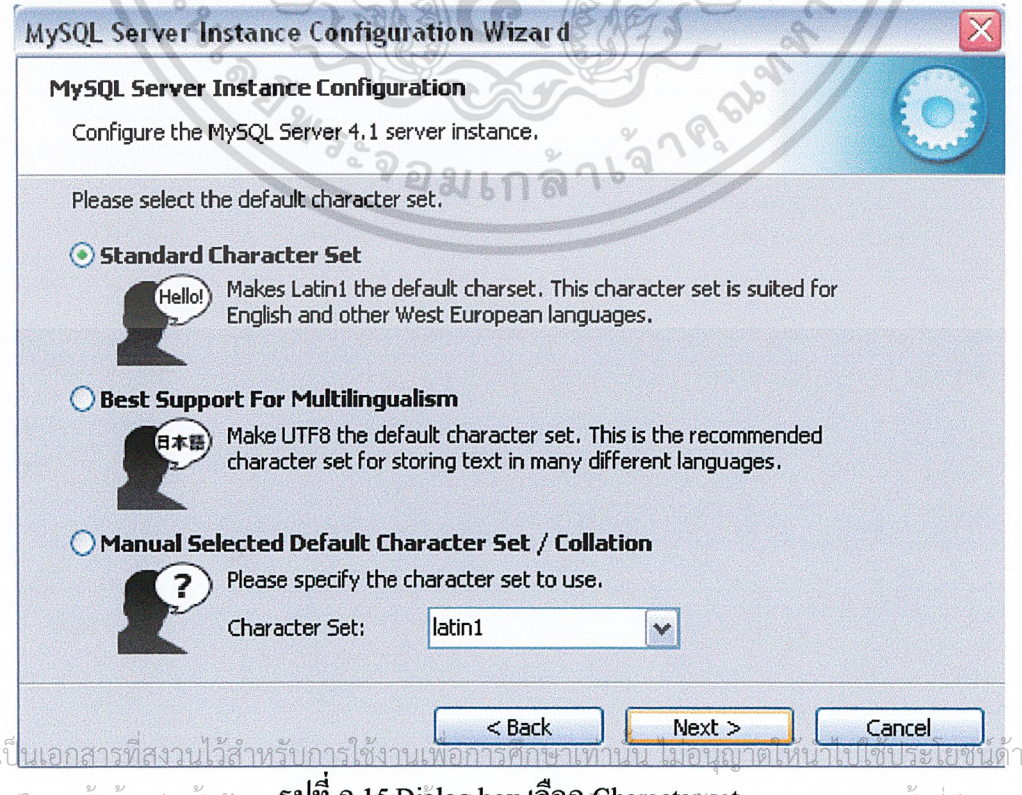
รูปที่ ก.13 Dialog box เลือกประเภทของการติดต่อกับ database ตามจำนวน connection ที่ทำพร้อมๆกัน

เลือกประเภทของการ connection แบบ Decision Support(DSS)/OLAP แล้วกดปุ่ม next



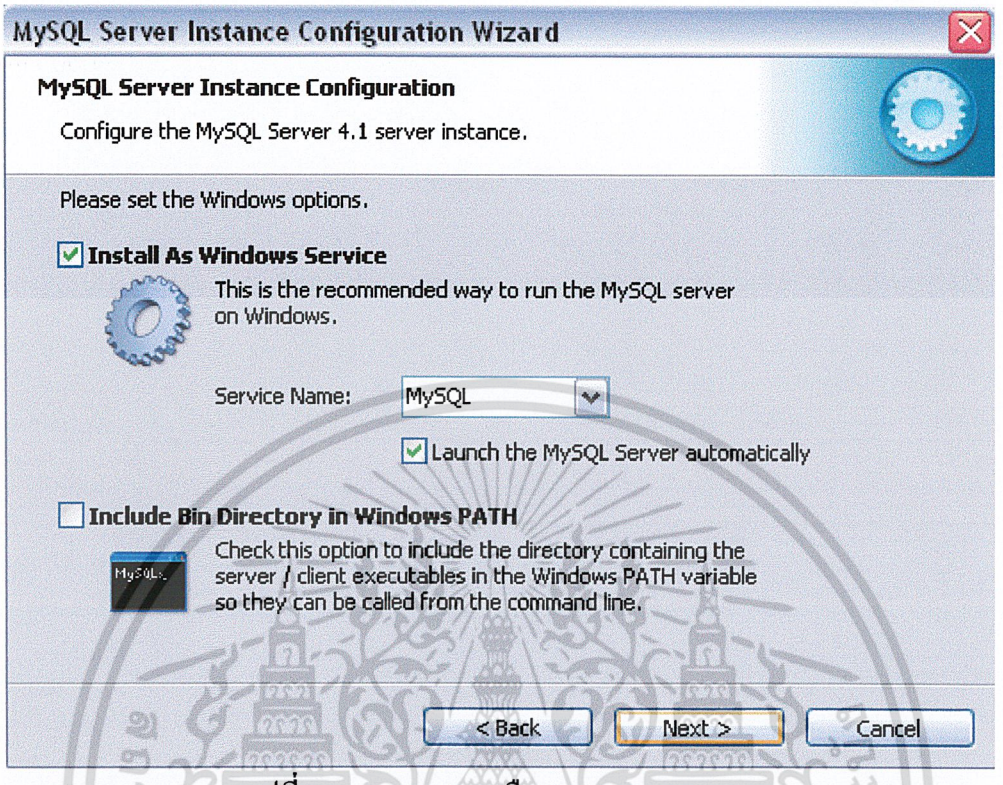
รูปที่ ก.14 Dialog box เลือก หมายเลข port

เลือก หมายเลข port เป็น 3306 แล้วกดปุ่ม next



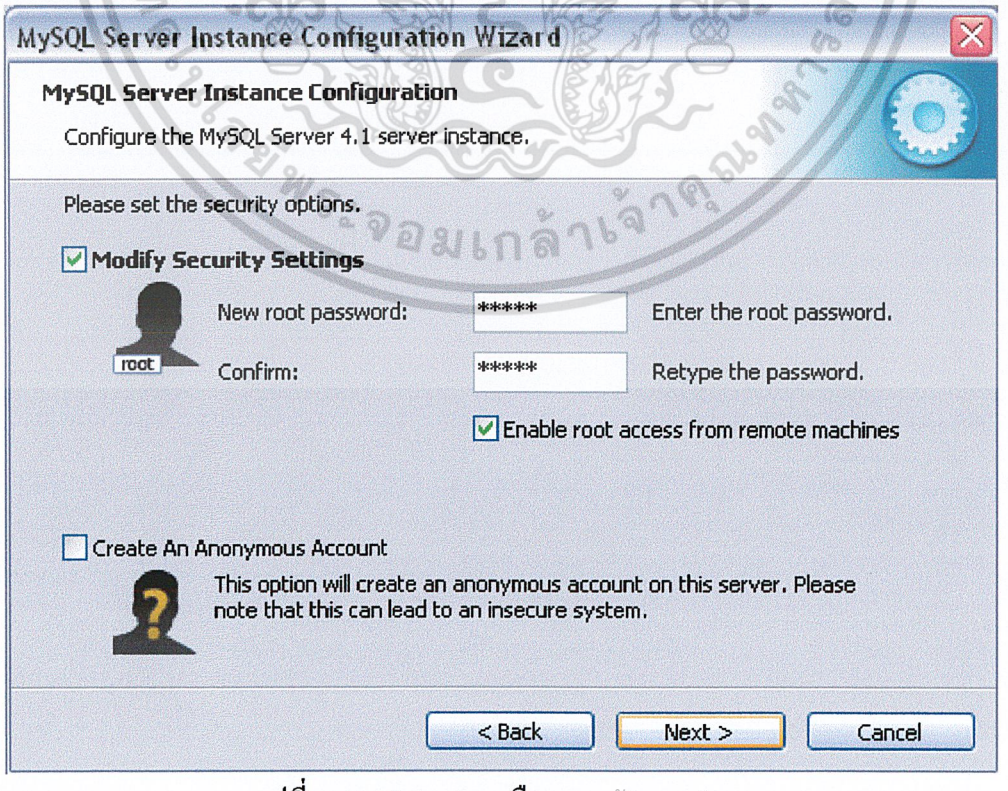
รูปที่ ก.15 Dialog box เลือก Character set

เลือก character set แบบ Standard Character Set



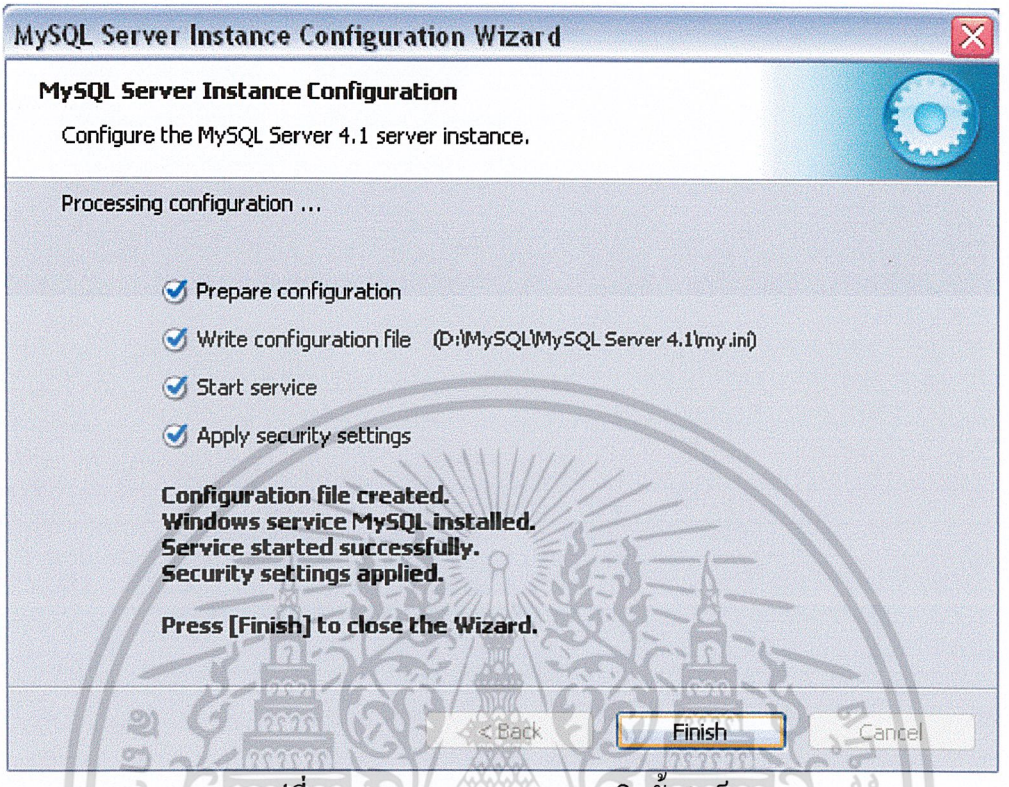
รูปที่ ก.16 Diglog box เลือก windows options

เลือก windows options แบบ Install As Windows Service



รูปที่ ก.17 Dialog box เลือก security options

เลือก security options แบบ Modify Security Settings และใส่ password เป็น “shoot”



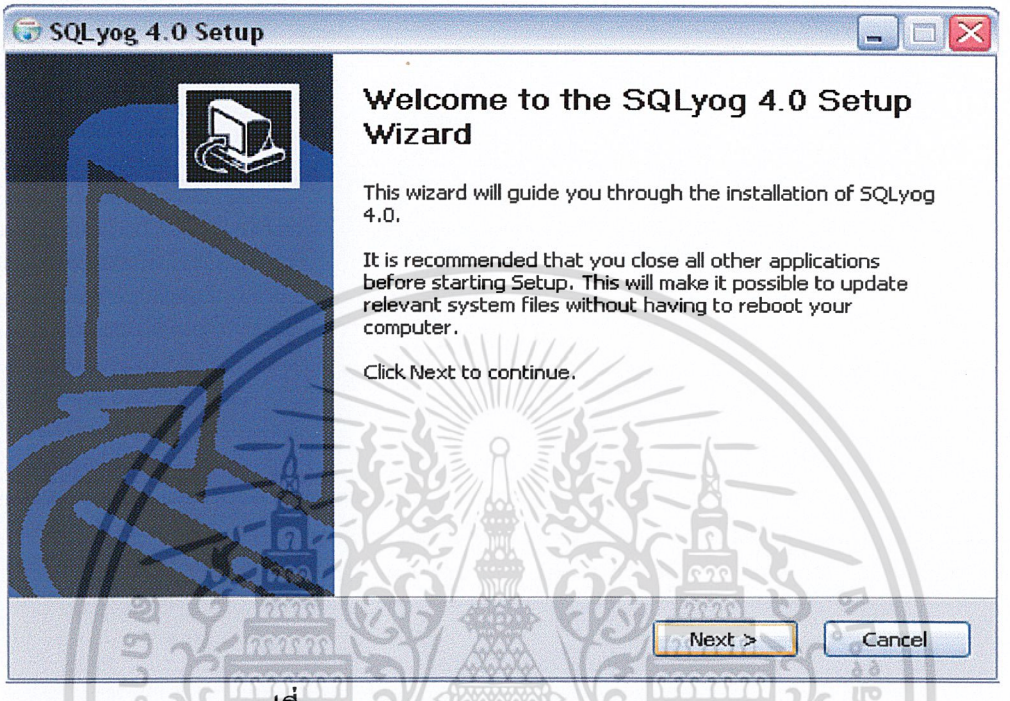
รูปที่ ก.18 Dialog box แสดงการติดตั้งสำเร็จ

กด execute แล้วรอจนกว่า จะมีเครื่องหมาย ☑ ครบทุกช่องดังรูป จึงกดปุ่ม finish

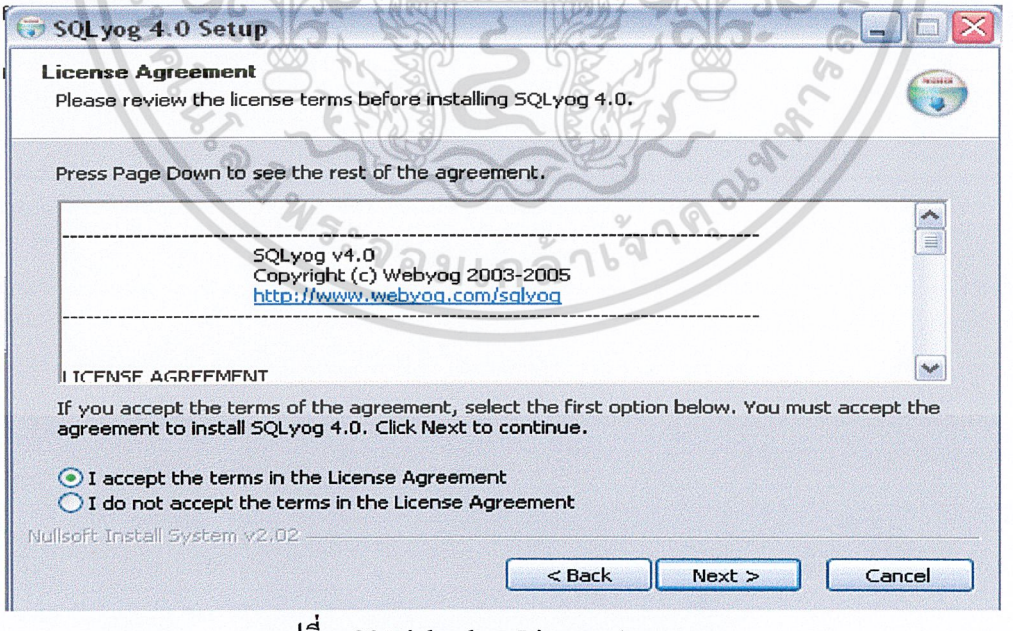
การติดตั้ง SQLyog40

ดับเบิ้ลคลิกที่ icon 

จะปรากฏ dialog box ดังรูป แล้วกดปุ่ม next

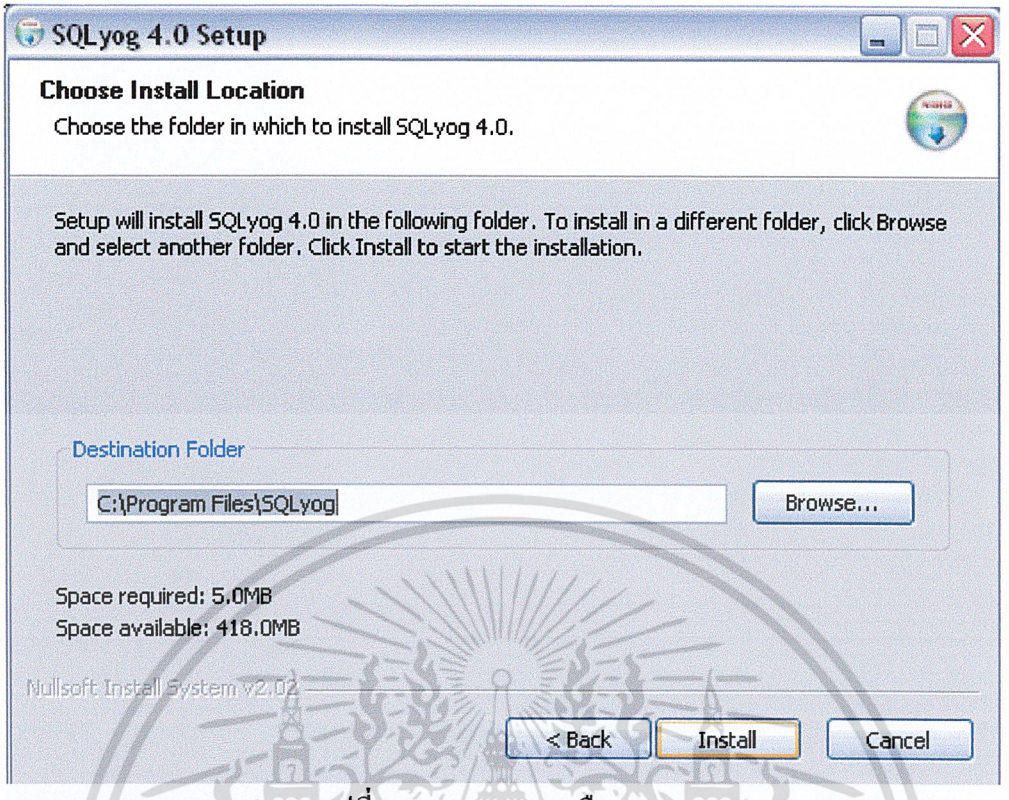


รูปที่ ก.19 Dialog box ของ SQLyog 4.0 Setup



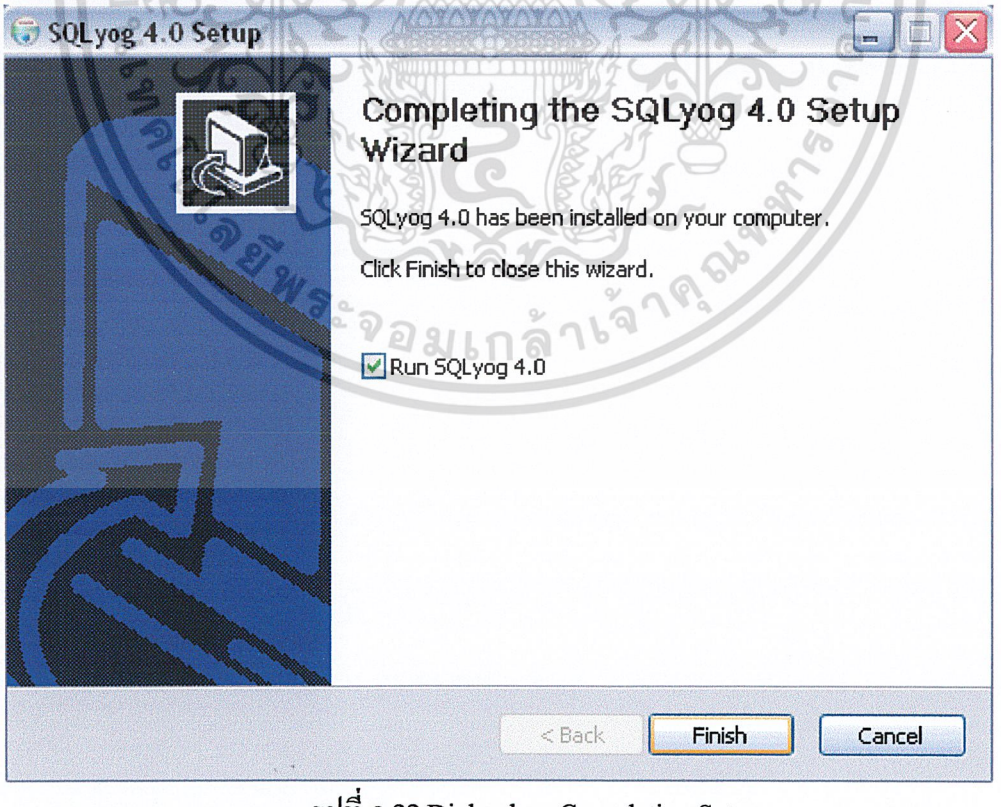
รูปที่ ก.20 Dialog box License Agreement

เลือก I accept the term in the License Agreement แล้วกดปุ่ม next



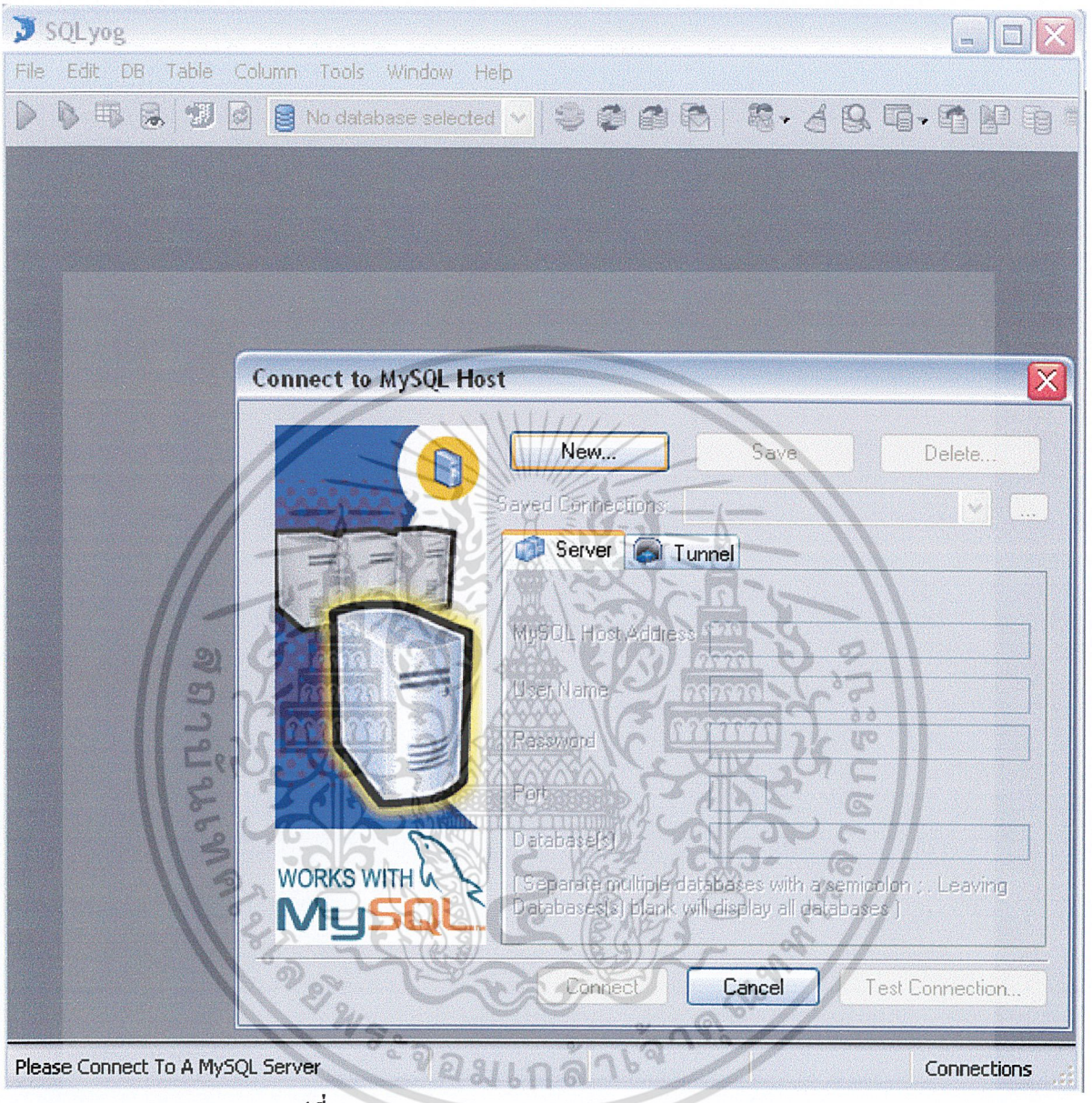
รูปที่ ก.21 Dialog box เลือก path

เลือก path ที่ต้องการติดตั้งแล้วกดปุ่ม next



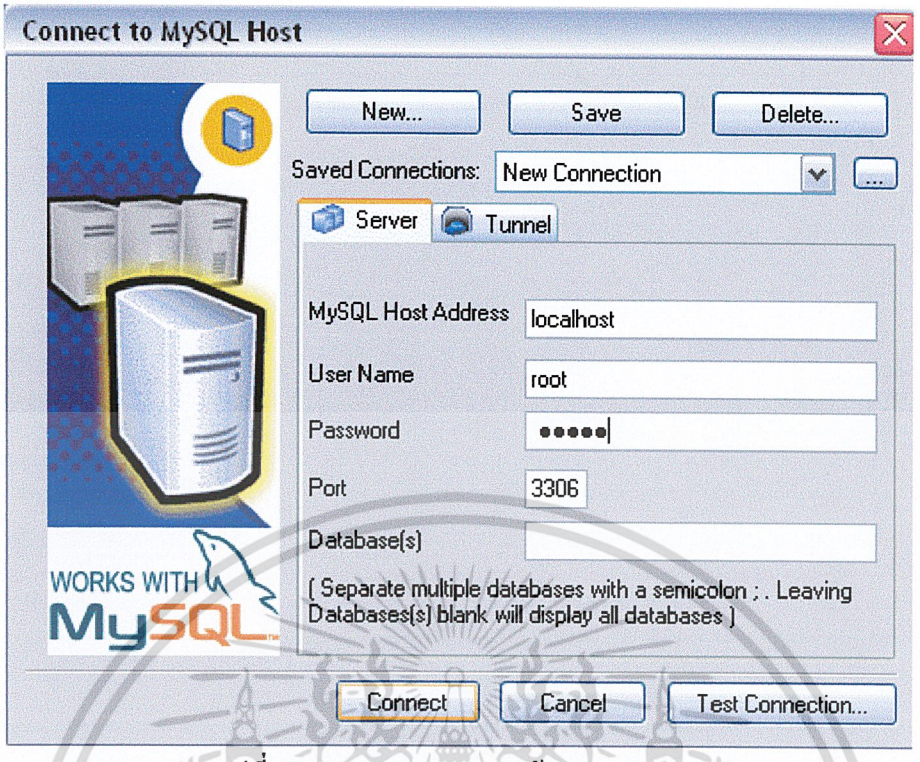
รูปที่ ก.22 Dialog box Completing Setup

กดปุ่ม finish



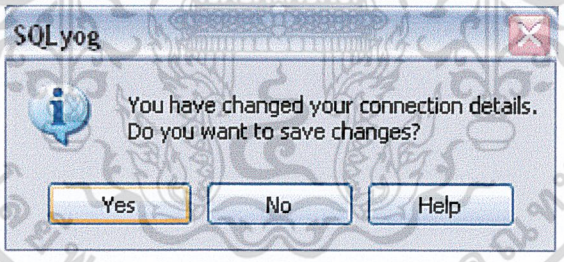
รูปที่ ก.23 Dialog box "Connect to MySQL Host"

หลังจากติดตั้ง SQLyog เรียบร้อยแล้ว โปรแกรมจะเปิดขึ้นมาและมี dialog box ดังรูป ให้กดปุ่ม new เพื่อสร้าง connection ใหม่



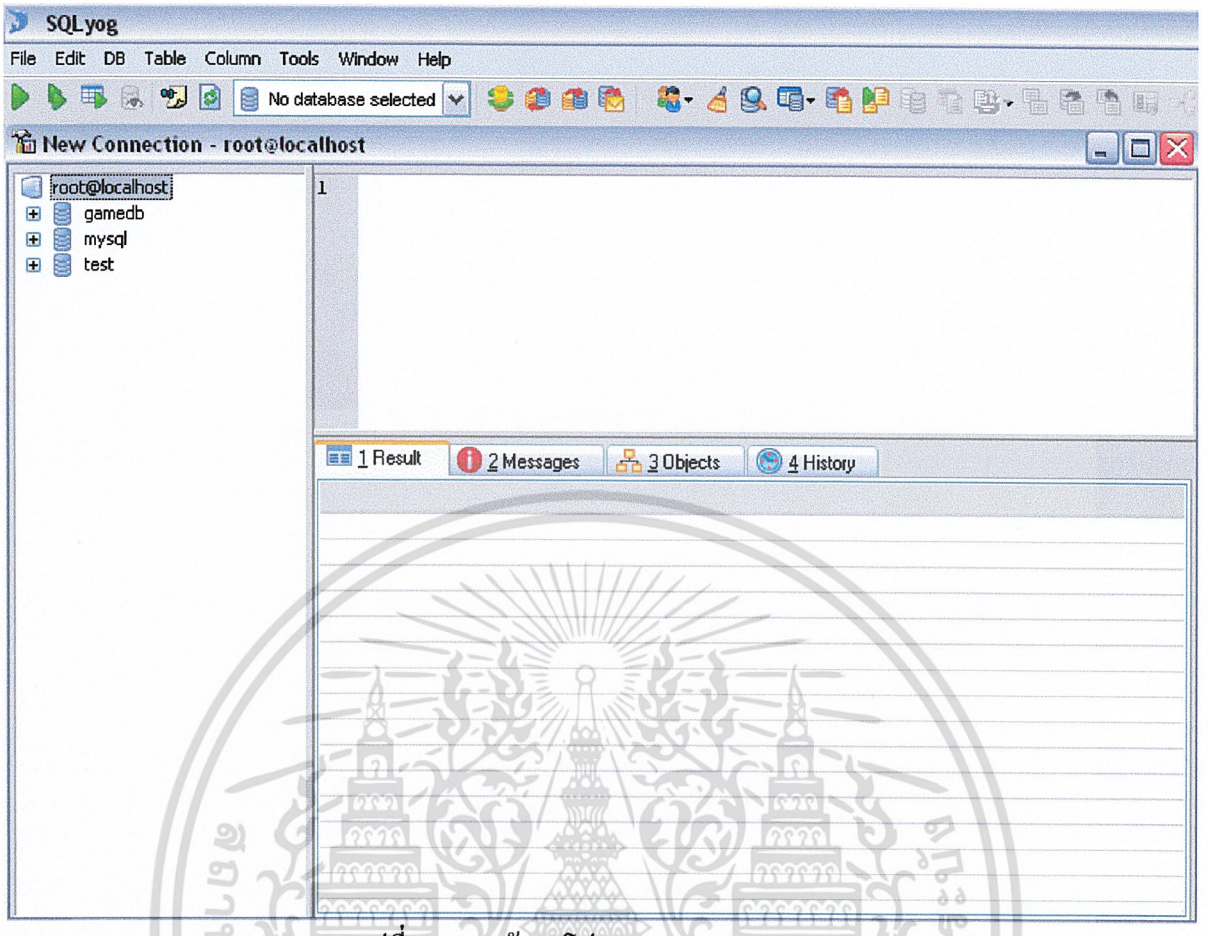
รูปที่ ก.24 Dialog box การสร้าง connection

ใส่ password เป็น “shoot” และหมายเลข port เป็น 3306 แล้วกดปุ่ม connect



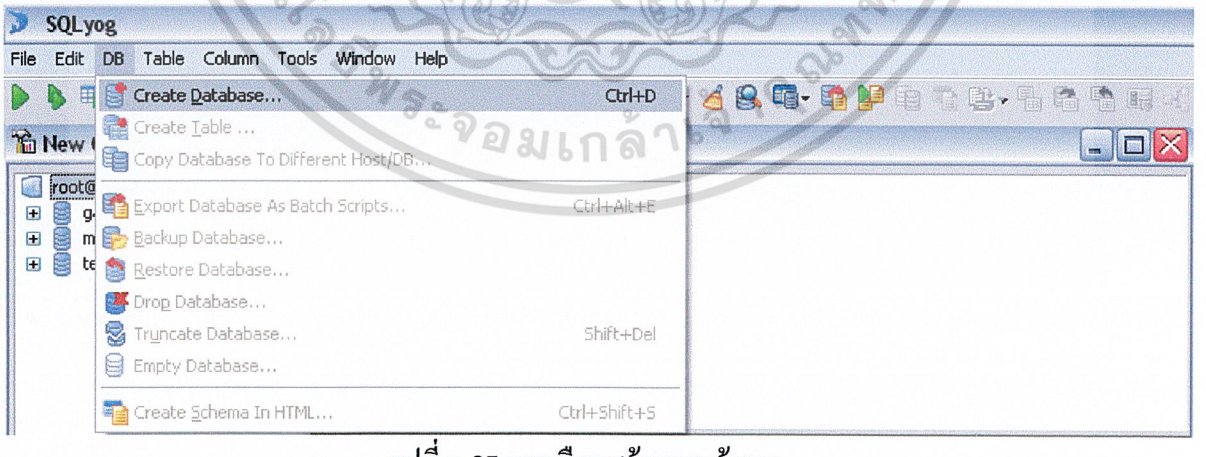
รูปที่ ก.25 Dialog box ถามว่า Do you want to save changes?

กดปุ่ม yes แล้วเข้าสู่ โปรแกรม SQLyog 4.0

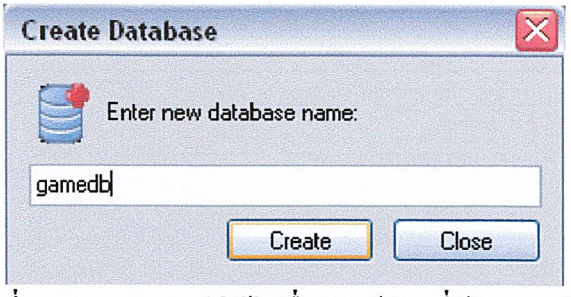


รูปที่ ก.26 หน้าจอโปรแกรม SQLyog 4.0

ทำการสร้างฐานข้อมูลโดยเลือกเมนู DB แล้วเลือก Create Database... ดังรูป

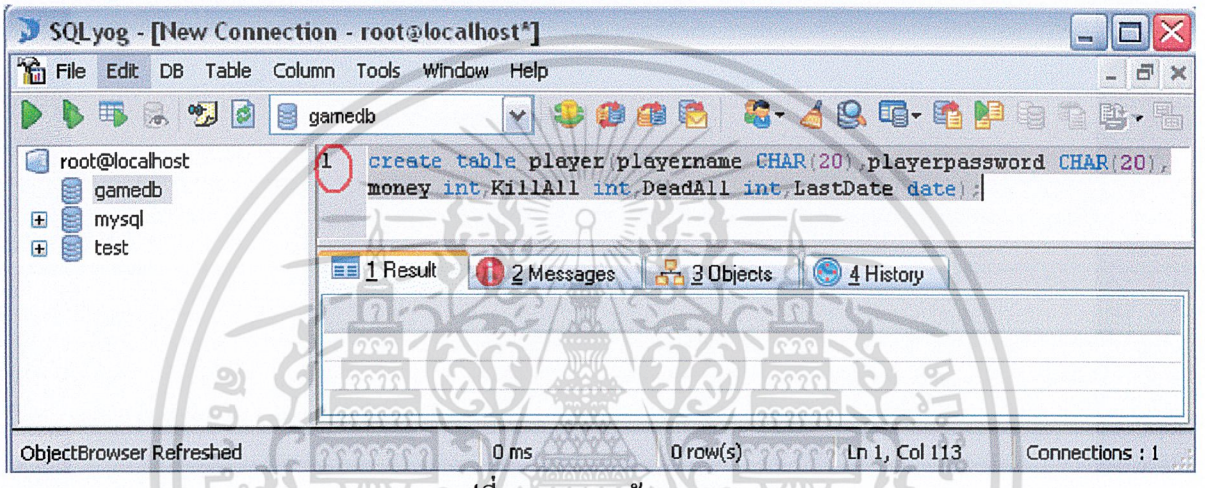


รูปที่ ก.27 การเลือกสร้างฐานข้อมูล



รูปที่ ก.28 Dialog box ให้ใส่ชื่อฐานข้อมูลที่ต้องการสร้าง

ใส่ชื่อฐานข้อมูลที่ต้องการสร้างเป็น “gamedb” แล้วกดปุ่ม create

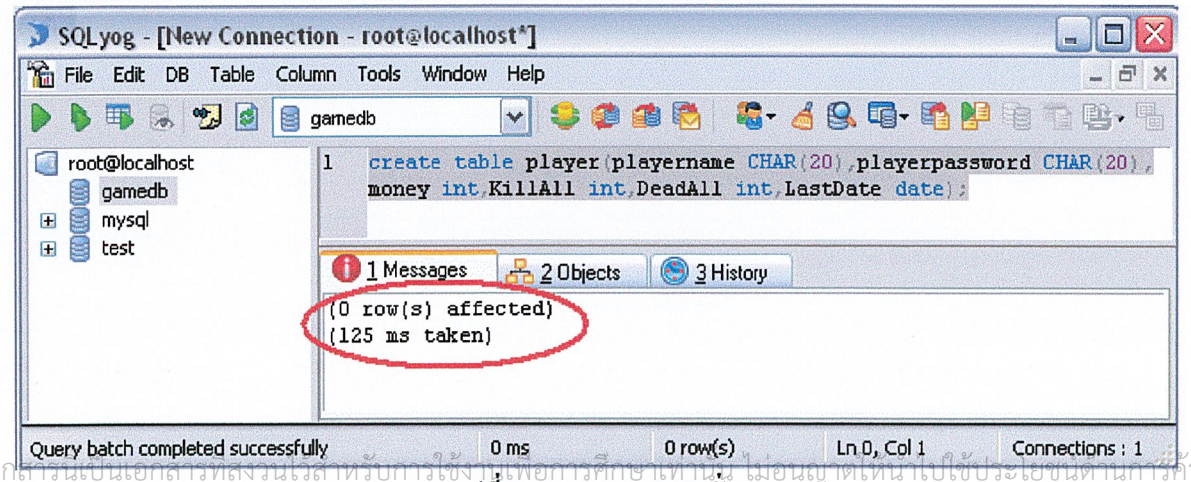


รูปที่ ก.29 การสร้าง table

เลือกฐานข้อมูล gamedb แล้วพิมพ์คำสั่ง SQL ดังนี้

create table player(playername CHAR(20),playerpassword CHAR(20),money int, KillAll int ,DeadAll int ,LastDate date);

จากนั้นคลิกที่หมายเลขบรรทัด (วงกลมสีแดง) เพื่อ highlight คำสั่ง SQL ที่พิมพ์ แล้วกด  และถ้าคำสั่งทำงานได้สำเร็จ จะแสดงผลดังรูป



รูปที่ ก.30 ผลการทำคำสั่ง

ภาคผนวก ข
คู่มือการเล่น

ความเป็นมา

เกมนี้จับเอาเรื่องราวใน Counter strike เป็นจุดดำเนินเรื่อง โดยจะเน้นถึงการทำการต่อสู้ของผู้เล่น 2 ฝ่าย โดยส่วนใหญ่เกมที่มีอยู่ในปัจจุบันจะอาศัยพลังของอาวุธที่ใช้เข้าโจมตีคู่ต่อสู้ เช่น ปืน และเสื้อเกราะสำหรับสวมใส่เพื่อปกป้องหรือลดความรุนแรงของกระสุนที่จะมากระทบตัวเอง ดังนั้นผู้เล่นก็จะเลือกซื้อเสื้อเกราะและปืนที่มีประสิทธิภาพสูงที่สุด ไว้ใช้สำหรับการเล่น ดังนั้นรูปแบบของเกมจึงกลายเป็นว่าผู้ที่มีเงินมากกว่าเป็นผู้ชนะ ในความเป็นจริงนั้น ใครมีเงินมากกว่าและสามารถเลือกซื้อเสื้อเกราะและปืนที่มีประสิทธิภาพสูงกว่าย่อมได้ชัยชนะ แต่ในบางครั้งความจริงนั้นก็ไม่จริงเสมอไป การต่อสู้ด้วยอาวุธที่มีประสิทธิภาพน้อยกว่า ในบางครั้งอาจได้รับชัยชนะหากทำการต่อสู้ด้วยสติปัญญาและการสื่อสารภายในทีมที่ดี โดยอาศัยการสื่อสารภายในทีมที่เป็นระเบียบ การวางแผนเป็นขั้นตอนและการทำการต่อสู้แบบเป็นทีม คือว่าในการเล่น ภายในทีมจะต้องวางแผนและปฏิบัติตามแผนเพื่อให้ทีมประสบกับชัยชนะ

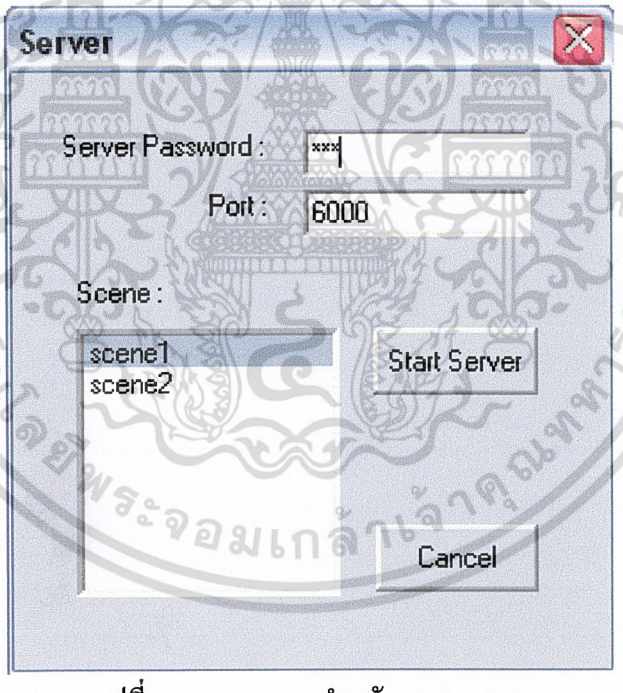
ประสิทธิภาพของระบบที่แนะนำ

1. คอมพิวเตอร์ระดับความเร็ว 1.80 GHz
2. หน่วยความจำขนาด 256 Mbyte
3. อุปกรณ์อินพุทประเภทเมาส์ และคีย์บอร์ด

เริ่มเข้าเกมส์โดยทำการ start server โดยการดับเบิ้ลคลิกเข้าไปใน folder Server Game แล้วเข้าไปใน folder debug จะปรากฏ icon ของ server program ดังรูป

รูปที่ ข.1 icon ของ server program

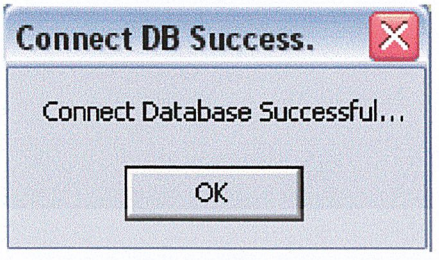
ให้ดับเบิ้ลคลิกที่ myTest-package ซึ่งเป็นไฟล์ .exe เพื่อรันโปรแกรม server จะปรากฏ dialog box ดังรูป



รูปที่ ข.2 dialog box สำหรับ start server

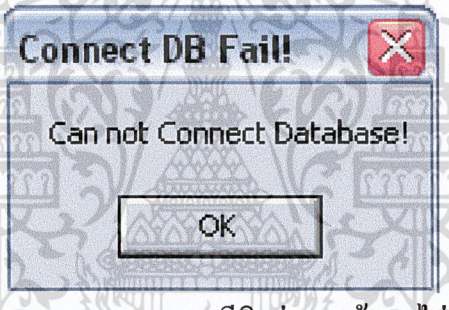
ใส่ข้อมูล Server Password (ไม่ต้องใส่ก็ได้) และเลือก scene (ฉากของเกมส์) ที่ต้องการ แล้วกดปุ่ม Start Server

ถ้า server program ติดต่อฐานข้อมูลได้สำเร็จก็จะปรากฏ message box ดังนี้



รูปที่ ข.3 message box กรณีติดต่อฐานข้อมูลสำเร็จ

แต่ถ้า server program ติดต่อฐานข้อมูลไม่สำเร็จ จะปรากฏ message box ดังนี้



รูปที่ ข.4 message box กรณีติดต่อฐานข้อมูลไม่สำเร็จ

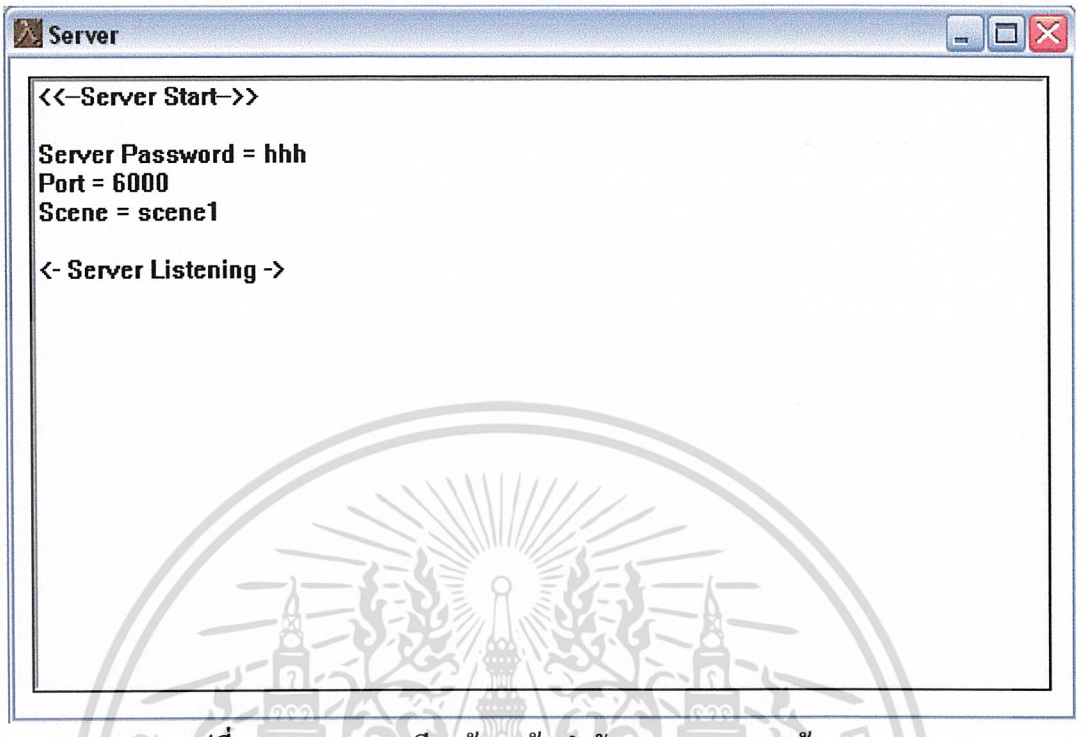
ให้ทำการตรวจสอบ username ,password และ หมายเลข port ที่ใช้ติดต่อกับ ฐานข้อมูลใหม่ ว่าเป็นดังนี้

Username : root

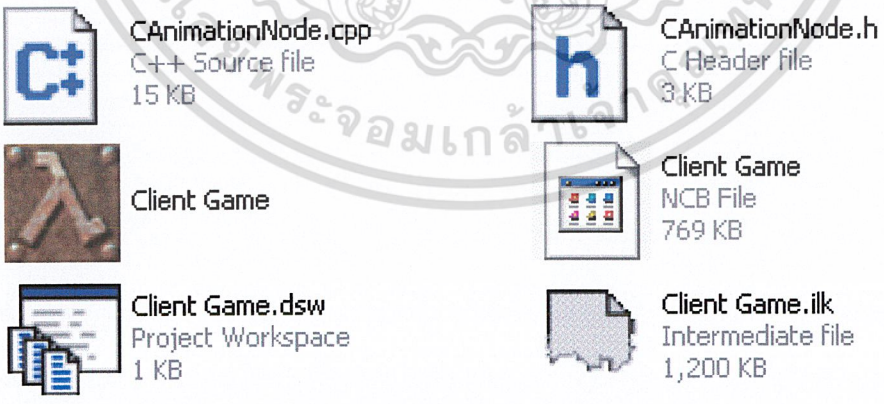
Password : shoot

Port : 3306

เมื่อติดต่อฐานข้อมูลได้แล้วกด OK จะปรากฏ dialog box ดังรูป



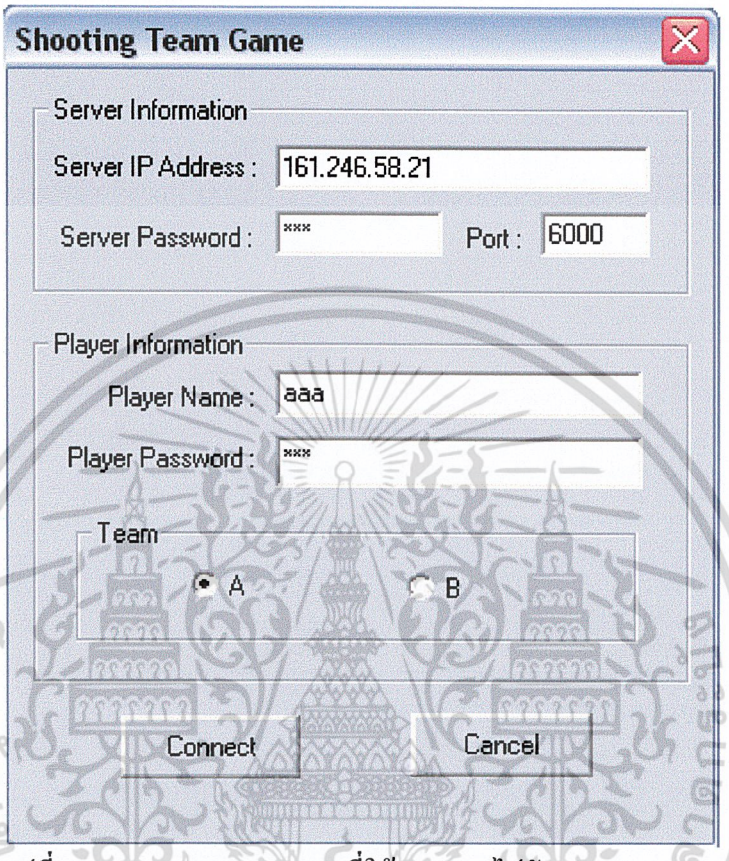
จากนั้นจึงเริ่มรัน client program โดยดับเบิ้ลคลิกเข้าไปใน folder Client Game จะปรากฏ icon ของ client game ดังรูป



รูปที่ ข.6 icon ของ client program

ให้ดับเบิ้ลคลิกที่ Client Game ซึ่งเป็นไฟล์ .exe เพื่อรันโปรแกรม client

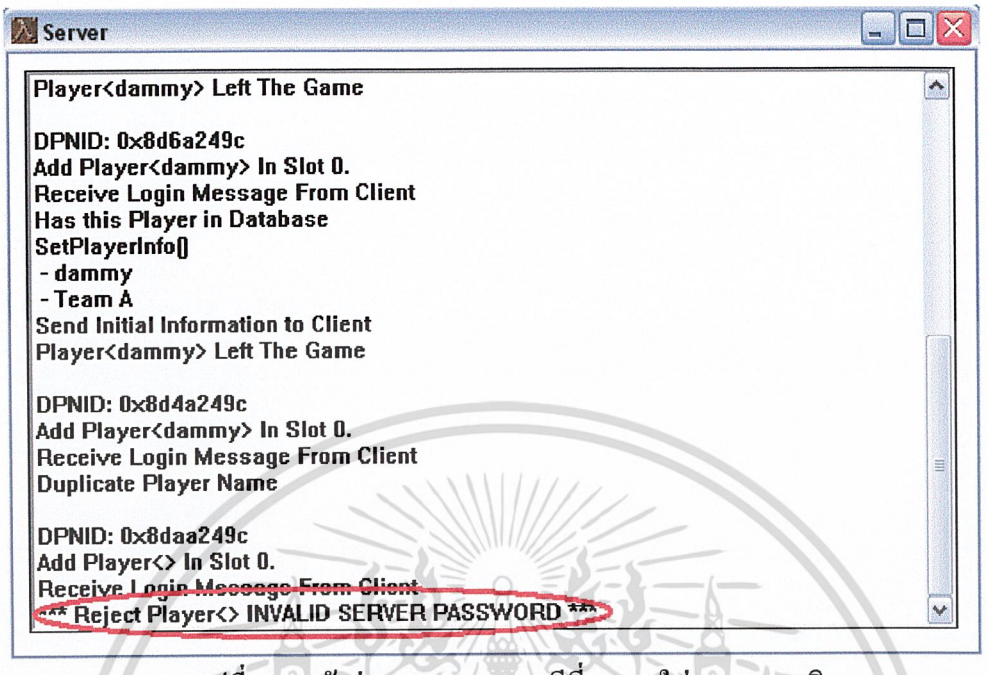
และจะปรากฏ dialog box ดังรูป



รูปที่ ข.7 dialog box ของ client ที่ใช้ connect ไปยัง server

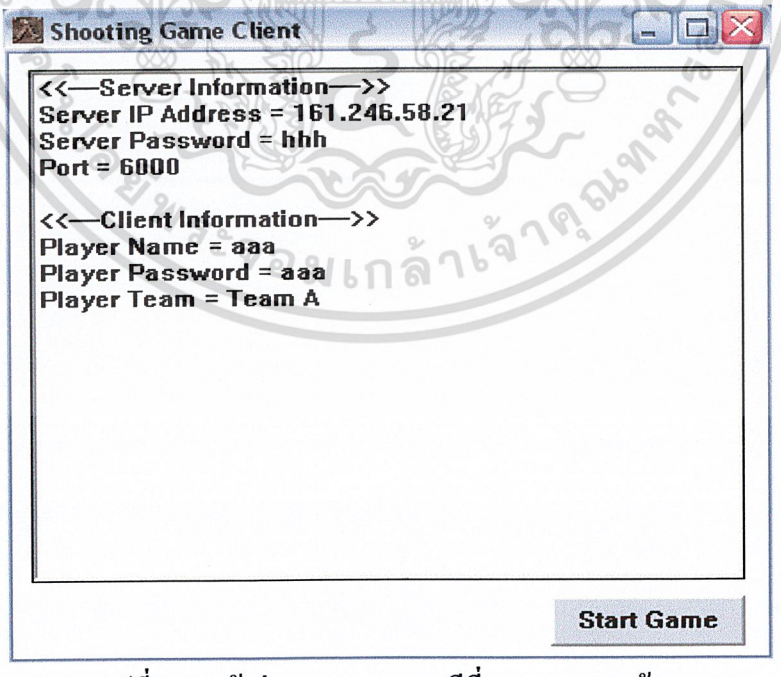
ใส่ข้อมูล IP address ของ server และserver password ที่ถูกต้อง (ถ้ามี) แล้วใส่ข้อมูลของผู้เล่น , เลือกทีม แล้วกดปุ่ม connect เพื่อติดต่อไปยัง server

ถ้าผู้เล่นใส่ password ของ server ผิด server program ก็จะ reject client program นั้น โดยอัตโนมัติ และจะมีข้อความขึ้นที่ server ดังนี้



รูปที่ ข.8 หน้าต่างของ server กรณีที่ client ใส่ password ผิด

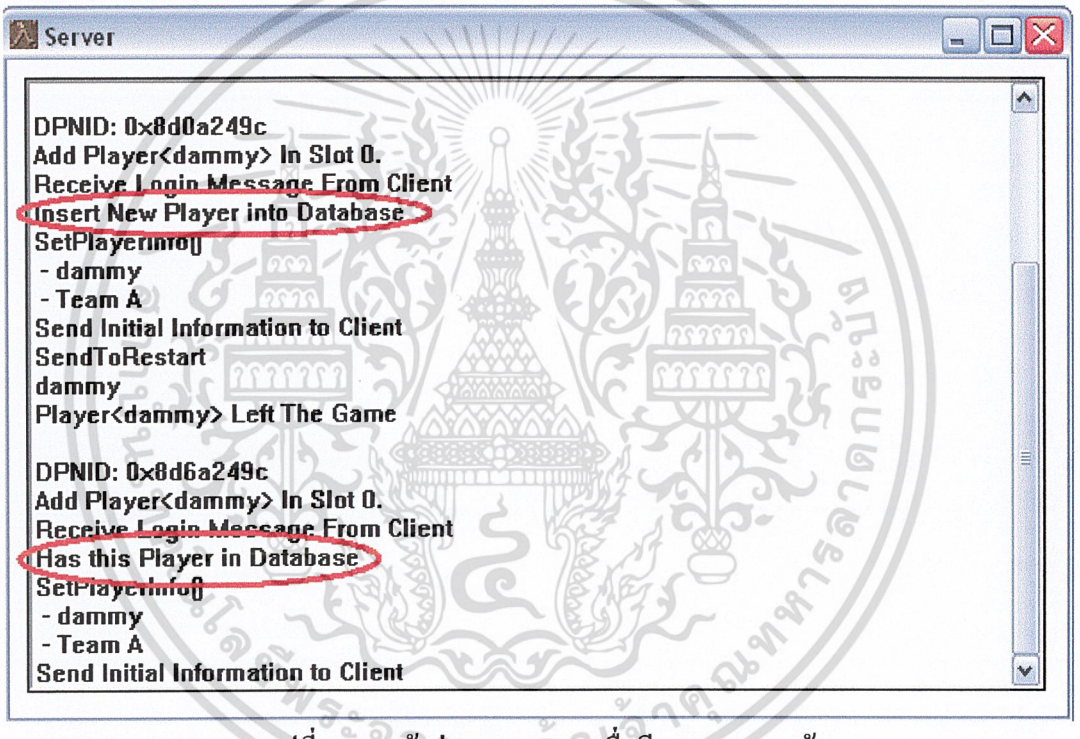
แต่ถ้าผู้เล่นใส่ password ของ server ถูกต้องก็จะสามารถเริ่มเกมส์ได้และจะปรากฏ dialog box ดังนี้



รูปที่ ข.9 หน้าต่างของ client กรณีที่ password ถูกต้อง

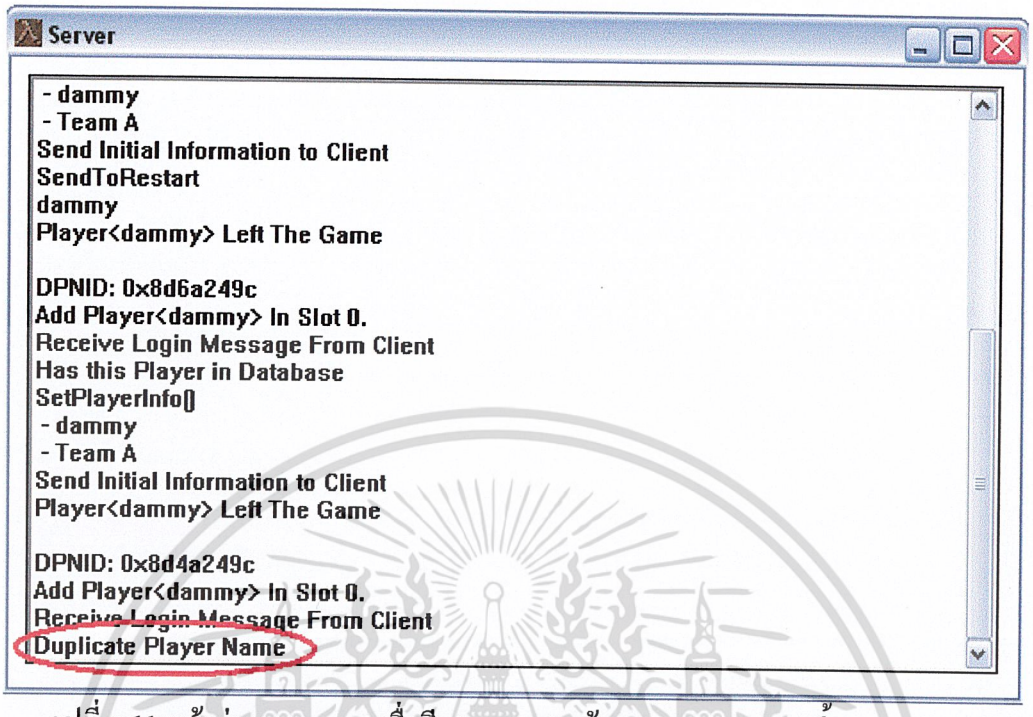
ให้ทำการขยายหน้าต่างให้เต็มหน้าจอก่อน แล้วจึงกดปุ่ม Start Game เพื่อส่ง login message ไปยัง server

server program จะตรวจสอบ username และ password ในฐานข้อมูล และถ้าพบว่าเป็นผู้เล่นใหม่ก็จะ insert ข้อมูลลงฐานข้อมูล และขึ้นข้อความว่า "Insert New Player into Database" และถ้าเป็นผู้เล่นที่เคย login เข้ามาแล้ว ก็จะขึ้นข้อความว่า "Has this Player in Database" แล้วมีข้อความว่า "Send Initial Information to Client" ขึ้นมารอไว้ให้ผู้ใช้กด enter ดังรูป



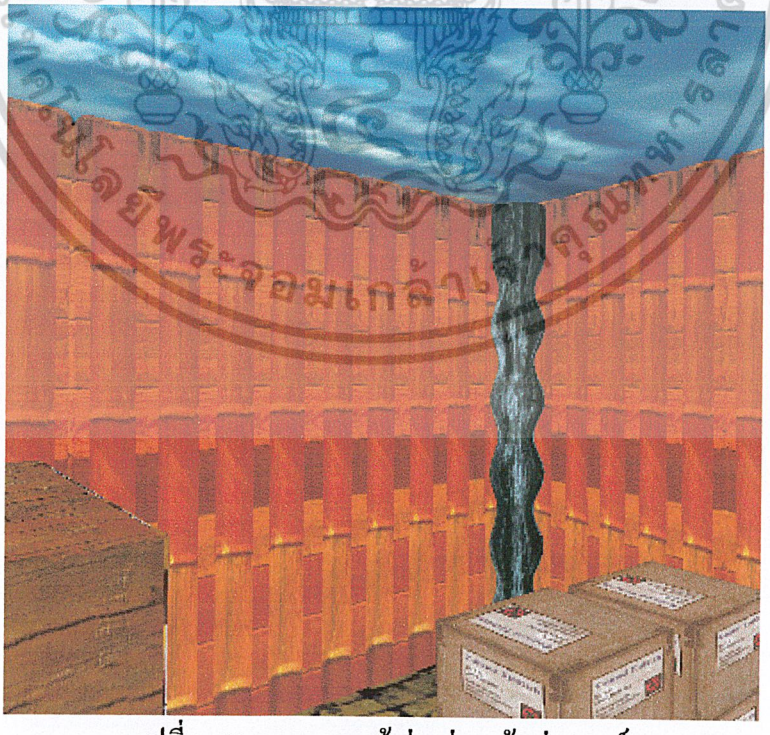
รูปที่ ข.10 หน้าต่างของ server เมื่อมี client login เข้ามา

แต่ถ้า username นั้นมีอยู่แล้ว และ password ไม่ตรง server ก็จะ reject client นั้น ทำให้โปรแกรมของ client ปิดไปและขึ้นข้อความที่ server ดังรูป



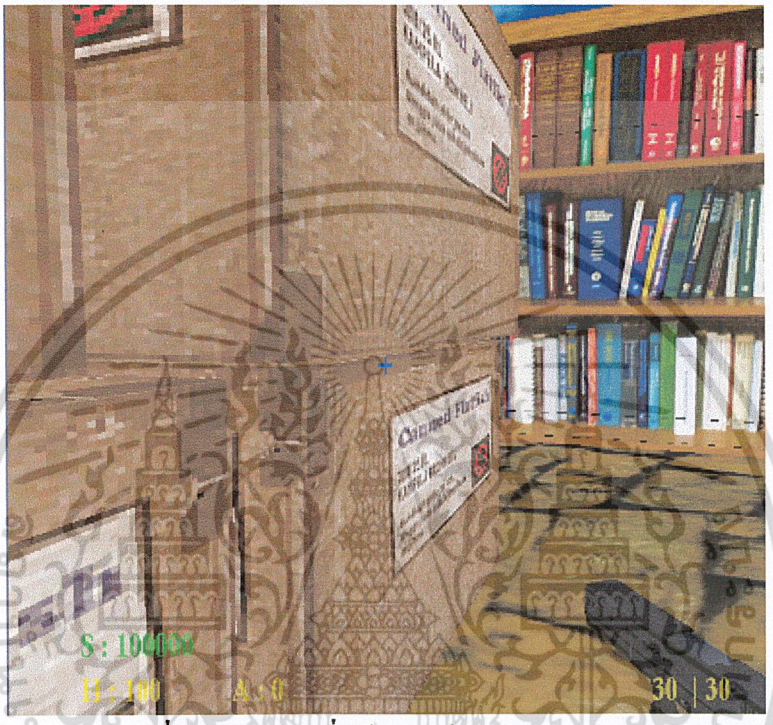
รูปที่ ข.11 หน้าต่างของ server เมื่อมี client login เข้ามา และ username ซ้ำ

ถ้าผู้เล่นสามารถ start client game ได้สำเร็จ ก็จะปรากฏเป็นฉากดังนี้

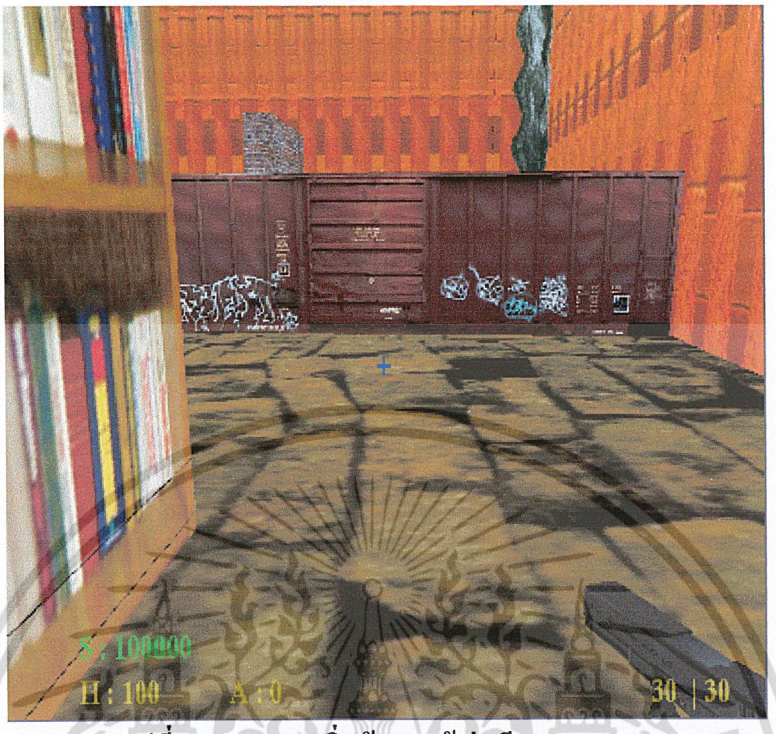


รูปที่ ข.12 มุมมองของผู้เล่น ก่อนเข้าเล่นเกมส์

ซึ่งผู้เล่นจะลอยอยู่ข้างบน ยังไม่สามารถเล่นเกมส์ได้ ต้องรอจนกว่า server จะกด enter ตามที่กล่าวข้างต้น หรือรอจนกว่าจะมีผู้เล่นอีกทีมเข้ามา client ก็จะทำการ restart โดยอัตโนมัติ และก่อนที่ผู้เล่นจะเริ่มเกมส์เล่น server จะมีการกำหนดตำแหน่งเริ่มต้นให้ ซึ่งผู้เล่นคนละทีมกัน ก็จะมีตำแหน่งเริ่มต้นอยู่คนละมุมกันในฉาก



รูปที่ ข.13 มุมมองเริ่มต้นของผู้เล่นทีม A



รูปที่ ข.14 มุมมองเริ่มต้นของผู้เล่นทีม B

เริ่มเล่นเกมส์

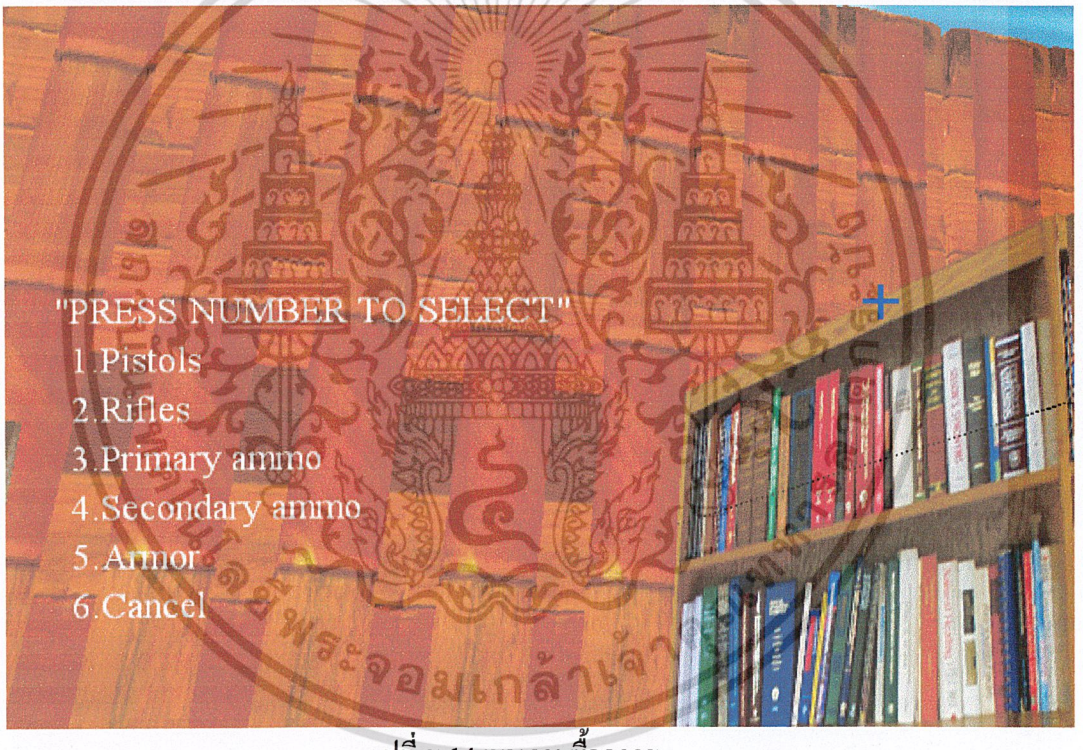
ส่วนของการกระทำ

ปุ่มกดบน keyboard / mouse	การกระทำ
ลูกศรขึ้น หรือ w	เดินหน้า
ลูกศรลง หรือ S	ถอยหลัง
ลูกศรซ้าย หรือขยับmouse ไปทางซ้าย	หันซ้าย
ลูกศรขวา หรือขยับ mouse ไปทางขวา	หันขวา
A	ลากเท้าไปทางซ้าย
D	ลากเท้าไปทางขวา
space bar	กระโดด
Ctrl	หมอบ
Ctrl+ลูกศรขึ้น หรือ w	หมอบเดินหน้า
Ctrl+ลูกศรลง หรือ S	หมอบถอยหลัง

ปุ่มกดบน keyboard / mouse	การกระทำ
Ctrl+ลูกศรซ้ายหรือขยับmouse ไปทางซ้าย	หมอบหันซ้าย
Ctrl+ลูกศรขวาหรือขยับmouse ไปทางขวา	หมอบหันขวา
Ctrl+A	หมอบลากเท้าไปทางซ้าย
Ctrl+D	หมอบลากเท้าไปทางขวา
ปุ่ม mouse ซ้าย	ยิง

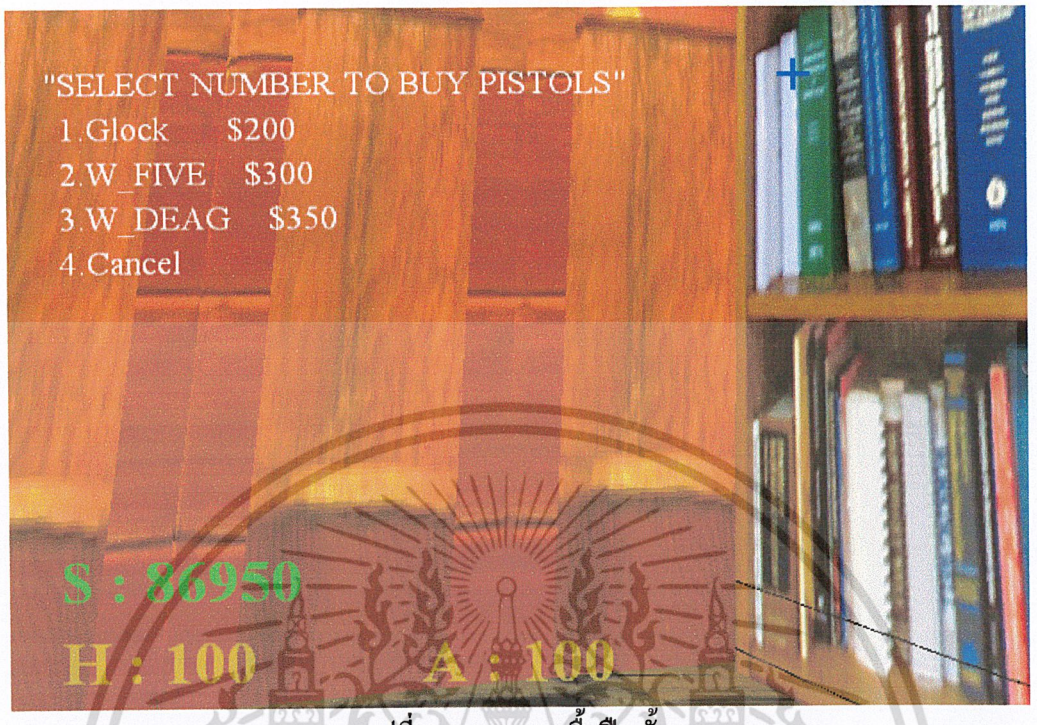
ส่วนของปืน

- การซื้อปืน ให้กด b จะปรากฏเมนูให้เลือกดังรูป



รูปที่ ข.14 เมนูการซื้ออาวุธ

ให้ใส่หมายเลขเพื่อเลือก เช่น กด 1 เพื่อซื้อปืนสั้น , กด 2 เพื่อซื้อปืนยาว , กด 5 เพื่อซื้อเสื้อเกราะ โดยเมื่อกดเลือกซื้อปืนสั้นหรือปืนยาว ก็จะมีปืน 3 แบบให้เลือกอีก ดังรูป



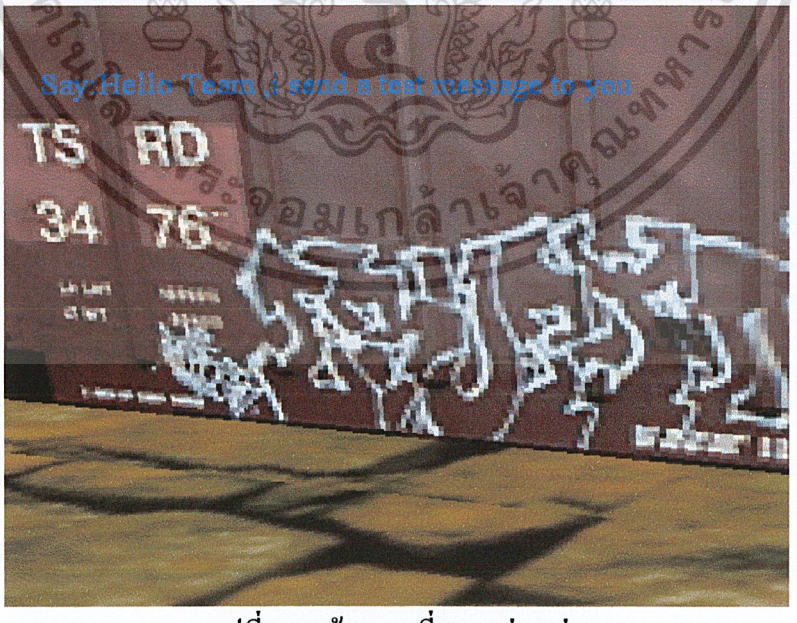
รูปที่ ข.15 เมนูการซื้อปืนสั้น

- การสลับปืนให้กด q ซึ่งถ้าผู้เล่นมีปืนประเภทหนึ่งอยู่ (ปืนสั้น-ปืนยาว) แล้วซื้อปืนประเภทเดียวกันเพิ่มก็ต้องทิ้งอันเดิม แต่ถ้าซื้อเพิ่มเป็นปืนคนละประเภทกัน ก็สามารถมีได้ 2 อันพร้อมกัน และสลับกันใช้ได้
- การทิ้งปืนให้กด g

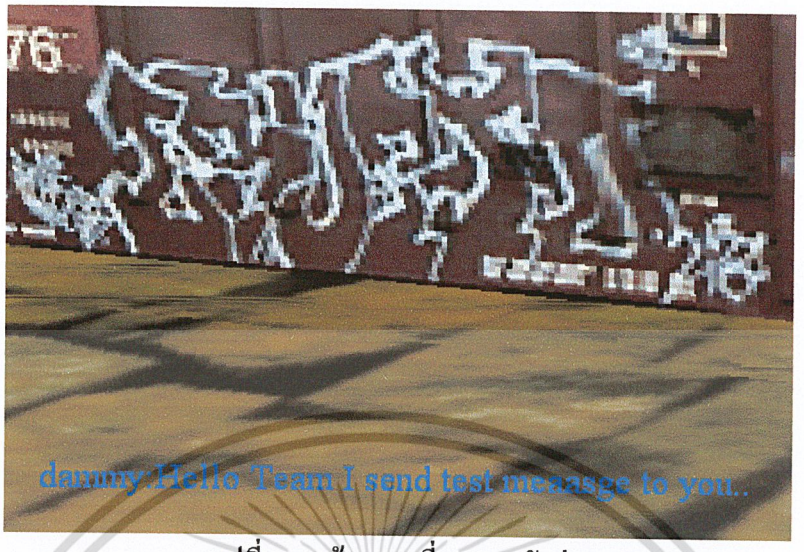
รูปที่ ข.16 ปืนที่ทิ้งแล้ววางอยู่บนพื้น

<u>ส่วนของการส่งข้อความสนทนา</u>

- การส่งข้อความสนทนา จะส่งไปยังผู้เล่นเฉพาะที่อยู่ทีมเดียวกัน โดยให้กด y หน้าจอจะปรากฏข้อความว่า “say:” รอให้ผู้เล่นพิมพ์ข้อความลงบนหน้าจอ ซึ่งจะแสดงข้อความไปขณะที่พิมพ์ แล้วให้ผู้เล่นกด enter เพื่อส่งข้อความ ซึ่งข้อความทั้งหมดจะไปปรากฏด้านล่างบนจอผู้ส่งสักพักหนึ่งแล้วหายไป เพื่อแสดงว่าส่งข้อความได้สำเร็จ

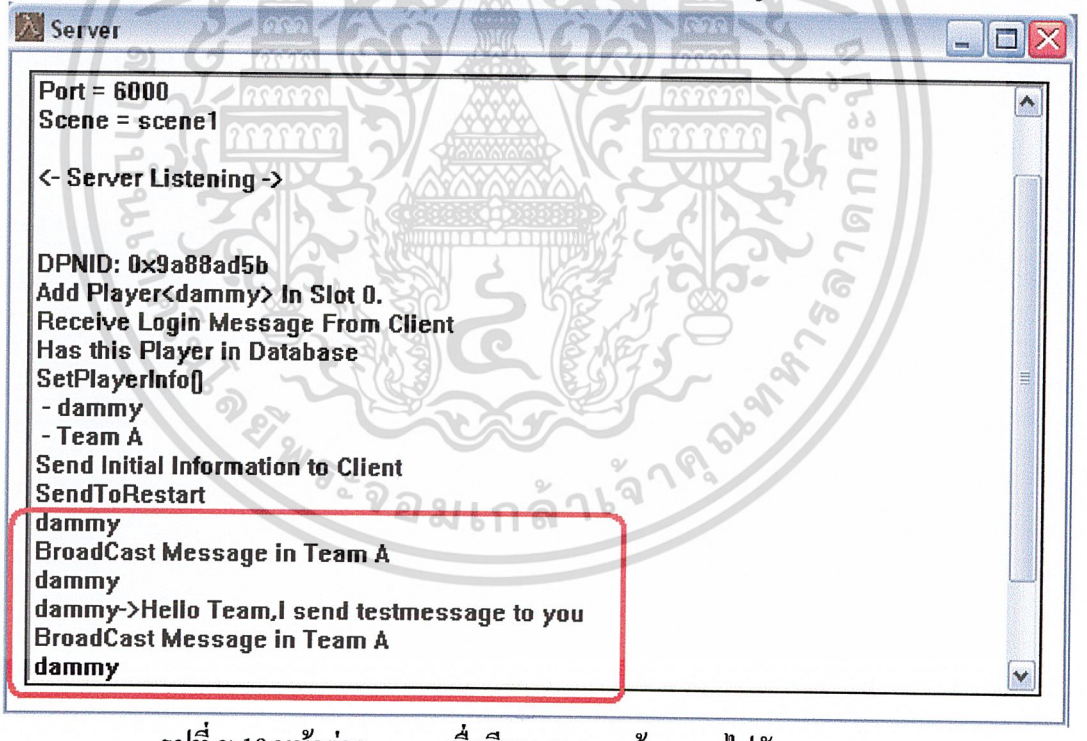


รูปที่ ข.17 ข้อความที่แสดงก่อนส่ง



รูปที่ ข.18 ข้อความที่แสดงหลังส่ง

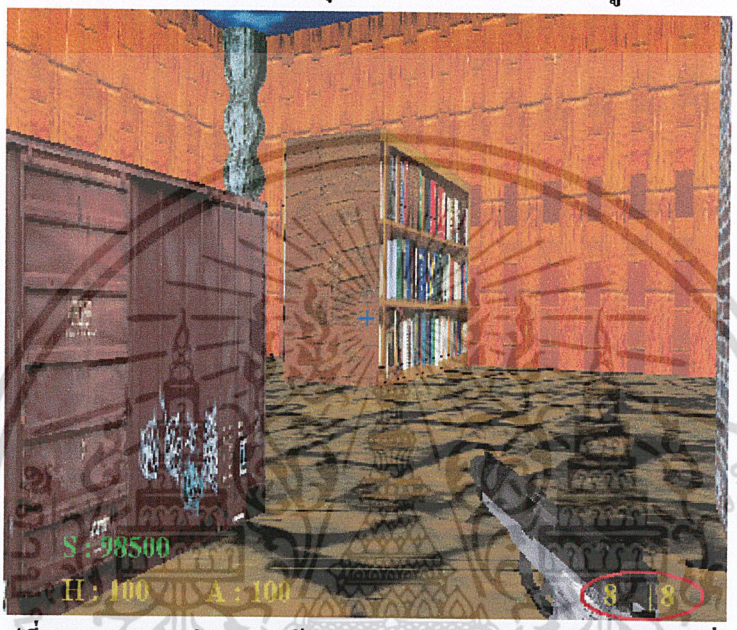
และ server ซึ่งทำหน้าที่กระจายข้อมูลจะมีข้อความขึ้นบนหน้าต่างดังรูป



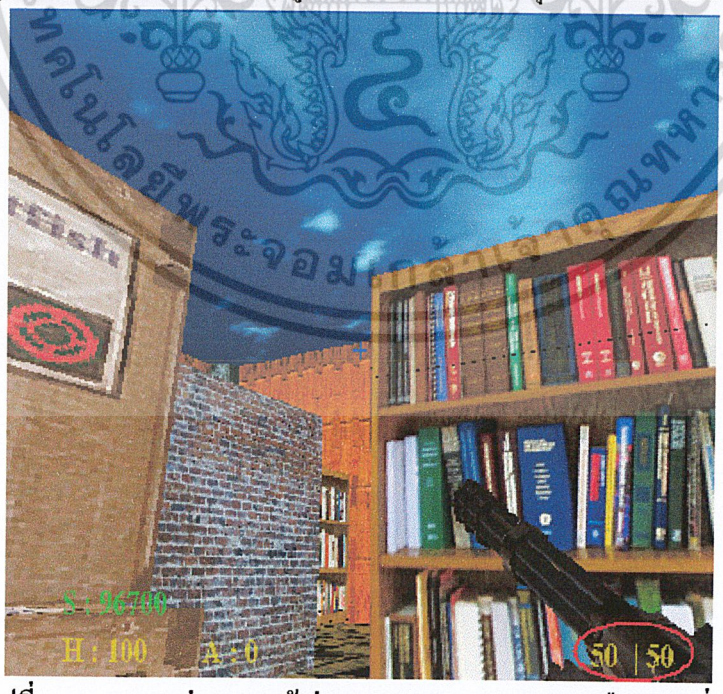
รูปที่ ข.19 หน้าต่าง server เมื่อมีการกระจายข้อความไปยัง clients

<u>ส่วนการแสดงสถานะของผู้เล่น</u>

หน้าต่างเกมส์จะแสดงสถานะของผู้เล่น ประกอบด้วย $ คือจำนวนเงินที่มี H คือพลังชีวิตที่มี และ A คือพลังเสื้อเกราะที่มี ส่วนตัวเลขทางด้านขวามือจะเป็น จำนวนกระสุนที่มีใช้อยู่ขณะนั้น กับจำนวนกระสุนสำรอง เมื่อผู้เล่นใช้กระสุนที่ใช้อยู่นั้นหมด ก็จะใช้กระสุนสำรองต่อ ซึ่งปืนแต่ละแบบก็จะมีจำนวนกระสุนสำรองไม่เหมือนกัน ดังรูป

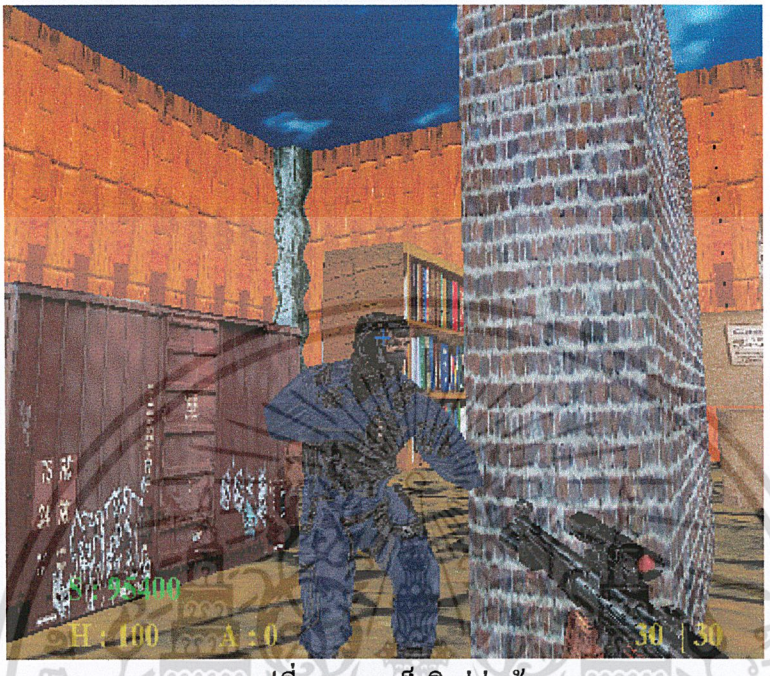


รูปที่ ข.20 สถานะต่างๆของผู้เล่น และจำนวนกระสุนของปืนแบบที่ 1



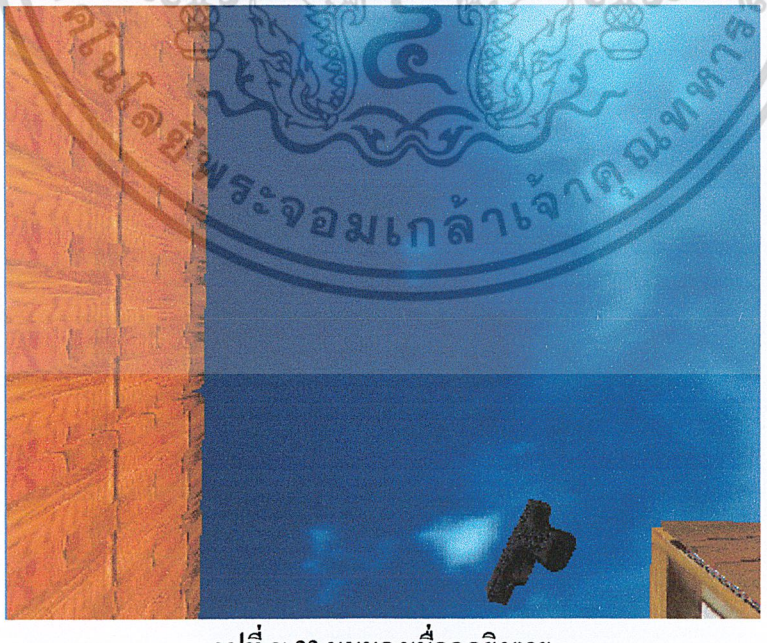
รูปที่ ข.21 สถานะต่างๆของผู้เล่น และจำนวนกระสุนของปืนแบบที่ 2

จุดเล็งยิงคู่ต่อสู้จะอยู่กลางหน้าต่างเกมส์ ดังรูป



รูปที่ ข.22 จุดเล็งยิงคู่ต่อสู้

เมื่อผู้เล่นถูกยิงตายจะปรากฏมุมมองดังรูป และถ้าเกมส์ไม่ restart โดยอัตโนมัติ ก็ให้ไปกด enter ที่ โปรแกรม server



รูปที่ ข.23 มุมมองเมื่อถูกยิงตาย

คุณสมบัติของผู้เล่น

- ชื่อ
- รหัสผ่าน
- จำนวนเงินคงเหลือ
- จำนวนที่ครั้งที่ฆ่าไปทั้งหมด
- จำนวนครั้งที่ตายไปทั้งหมด

คุณสมบัติของปืน

- ชื่อ
- ราคาปืน
- ความแรง / นัด(โดนบริเวณลำตัว)
- ความแรง / นัด(โดนศีรษะ)
- จำนวนกระสุน / Magazine
- จำนวน Magazine สำรองที่สามารถพกได้
- ราคากระสุน / Magazine

เงื่อนไขการจบเกมส์

ผู้เล่นฝ่ายใดที่ยิงอีกฝ่ายตายจนหมดก่อน จะเป็นฝ่ายชนะ และเมื่อจบเกมส์จะมีการเก็บหรือ update ข้อมูลของผู้เล่นแต่ละคนดังนี้ Money(จำนวนเงินที่เหลือ) , KillAll (ผู้เล่นนี้ยิงอีกฝ่ายตายไปอีกคนแล้ว) DeadAll (ผู้เล่นนี้เคยตายไปกี่ครั้งแล้ว)

บรรณานุกรม

- พีรภัทธ์ สว่างเพียร , เทคนิคการเขียนโปรแกรมและเกมด้วย Visual C++ , se-education public company limited
- Document ที่มากับโปรแกรม DirectX SDK
- Sarmad Kh Abdulla ,Implementing Skin Meshs with DirectX8. [online] Available : http://www.gamedev.net/reference/programming/features/skinmesh/default.asp
- นิรุธ อำนวยศิลป์ ,การเชื่อมต่อ MySQL ด้วย Visual C++ http://www.thaidev.com