การพัฒนาเกมสามมิติโดยเอนจินไอริทช์และเอนจินออเดียร์

3D GAME DEVELOPMENT BY IRRLICHT ENGINE AND AUDIERE ENGINE

ธีรชาติ แอบวนาวงศ์
นพอนนต์ อนันต์วิชัยชาญ
นิชา พู่วณิชย์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

3D GAME DEVELOPMENT BY IRRLICHT ENGINE AND AUDIERE ENGINE

TEERACHAT AEBWANAWONG

NOPANON ANANVICHAICHAN

NICHA POOVANICH

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2006

หัวข้อปัญหาพิเศษ	การพัฒนาเกมสามมิติโดยเอนจินไอริทช์และเอนจินออเดียร์ 3D GAME DEVELOPMENT BY IRRLICHT ENGINE AND AUDIERE ENGINE	
ชื่อนักศึกษา	นายธีรชาติ แอบวนาวงศ์	46050295
	นายนพอนนต์ อนันต์วิชัยชาญ	46050296
	นางสาวนิชา พู่วณิชย์	46050299
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล	

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2549

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ.ดร.กรกช ประชุมรักษ์	
กรรมการ	อ.สันธนะ อู่อุดมยิ่ง	
กรรมการและอาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล	

(รองศาสตราจารย์ ดร.วีระ บุญจริง)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ	การพัฒนาเกมสามมิติโดยเอนจินไอริทช์และเอนจินออเดียร์	
ชื่อนักศึกษา	นายธีรชาติ แอบวนาวงศ์	46050295
	นายนพอนนต์ อนันต์วิชัยชาญ	46050296
	นางสาวนิชา พู่วณิชย์	46050299
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2549	
อาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล	

บทคัดย่อ

ปัญหาพิเศษนี้เป็นการพัฒนาเกมสามมิติเดินยิงมุมมองบุคคลที่สาม ในลักษณะการดิ้นรนเอาชีวิตรอดจากศัตรูที่จะเข้ามารุมทำร้าย โดยเกมนี้เป็นเกมคอมพิวเตอร์ที่มีภาพแบบสามมิติ มีมุมมองในการเล่นเปิดกว้างมองเห็นตัวละครได้ทั้งหมด เดินถืออาวุธ เพื่อโจมตีศัตรูที่เข้ามารุมทำร้าย ซึ่งผู้เล่นต้องทำการผจญภัยไปในห้องต่าง ๆ เพื่อหาสิ่งของมาไขปริศนา โดยรูปแบบการเล่นผู้เล่นสามารถเล่นได้คนเดียว โดยเกมสามมิติเดินยิงมุมมองบุคคลที่สามนี้ พัฒนาขึ้นมาโดยใช้ Microsoft Visual C++ 6.0 ร่วมกับเกมเอนจิน ตัวละครต่าง ๆ และฉากจะนำเข้ามาจากโปรแกรม 3D Studio Max และการใช้เสียงเพลงประกอบเพื่อเพิ่มอรรถรสในการเล่น

ในการพัฒนาปัญหาพิเศษนี้ได้ศึกษาและออกแบบตัวเกมขึ้นมาใหม่ โดยได้ทำการพัฒนาในส่วนต่าง ๆ เช่น การเคลื่อนที่ของตัวละคร การตรวจสอบการชนกันระหว่างตัวละครในเกม การตรวจสอบการยิงศัตรู การเก็บไอเท็ม และการใช้ไอเท็ม เป็นต้น

Special Project Title	3D GAME DEVELOPMENT BY IRRLICHT ENGINE AND AUDIERE ENGINE
Students	Mr. Teerachat Aebwanawong 46050295
	Mr. Nopanon Ananvichaichan 46050296
	Miss Nicha Puvanich 46050299
Degree	Bachelor of Science
Department	Mathematics and Computer Sciences, Faculty of Science
Programme	Computer Science
Academic Year	2006
Special Project Advisor	Assoc.Prof.Teerawat Prakobphon

ABSTRACT

This special problem concerns with developing third-person shooting game in 3D. The player could try to escape form attacking of the enemies. This game has aspect that can see everything in game environment. In this game, the player has weapons to fight with the enemies and adventure in the rooms. Sometimes, the player must find the key to fix the puzzle. The software project is powered by Microsoft Visual C++ 6.0 including two engines : Irrlicht engine and Audiere engine. It entertains player with realistic graphics and sound effects and supports only one player mode.

This special problem , we study and design new game that developed in many functions such as character movement , collision detection , collecting and using items.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องการพัฒนาเกมสามมิติโดยเอนจินไอริทช์และเอนจินออเดียร์ สามารถสำเร็จลุล่วงได้ด้วยดีด้วยความช่วยเหลือ และความร่วมมือจากหลาย ๆ ท่าน คณะผู้จัดทำต้องขอขอบพระคุณบุคคลที่มีส่วนช่วยให้การทำงานครั้งนี้เสร็จไปได้ด้วยดี คือ รศ.ธีรวัฒน์ ประกอบผล อาจารย์ที่ปรึกษาปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำและแนวคิดต่าง ๆ ในการทำปัญหาพิเศษนี้ ดูแลเอาใจใส่ และให้การสนับสนุนทาง ด้านซอฟต์แวร์ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของโครงงานปัญหาพิเศษฉบับนี้และทุก ๆ ท่านที่ช่วยตอบกระทู้ทางอินเตอร์เน็ต

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณบุคคลที่เกี่ยวข้องทุกฝ่าย ที่ทำให้การทำปัญหาพิเศษนี้สำเร็จลุล่วงไปได้ด้วยดี รวมทั้งเพื่อนๆ และพี่ๆ ทุกคนที่ให้ความ ช่วยเหลือในด้านต่างๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2550

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเกมคอมพิวเตอร์ในยุคโลกโลกาภิวัตน์ กลายเป็นส่วนหนึ่งของวิถีชีวิตของคนยุคใหม่ จนกลายเป็นอุตสาหกรรมใหญ่ที่ทำรายได้ให้กับบริษัทซอฟต์แวร์ด้านนี้อย่างเป็นล่ำเป็นสัน ประเทศไทยเป็นประเทศหนึ่งที่ได้รับอิทธพลจากเกมคอมพิวเตอร์อย่างมาก โดยเฉพาะกลุ่มนิสิตนักศึกษา เยาวชนของชาติมีการบริโภคซอฟต์แวร์เหล่านี้อย่างฟุ่มเฟือย ประเทศไทยกลายเป็นผู้เสพเทคโนโลยีติดอันดับโลก ด้วยเหตุนี้จึงมีความตั้งใจที่จะเป็นส่วนหนึ่งให้ผู้ที่เกี่ยวข้องและเยาวชนได้เล็งเห็นความสำคัญในการสร้างบุคลากรที่เป็น "ผู้สร้าง" มากกว่า "ผู้เสพ"

ด้วยเหตุนี้จึงมีความตั้งใจที่จะนำความรู้ทางด้านคอมพิวเตอร์กราฟฟิกและภาษาคอมพิวเตอร์ที่ได้ศึกษามาในระดับหนึ่งมาประยุกต์ใช้ในการพัฒนาเกมคอมพิวเตอร์ขึ้นเอง โดยเกมที่จะพัฒนานั้นเป็นเกมแนวเดินยิงกันแบบมุมมองผู้เล่นที่สาม โดยให้เกมที่พัฒนานี้มีชื่อว่า สแตนซ่า(Ztanza)

1.2 ความมุ่งหมายและวัตถุประสงค์ ของการศึกษา

เพื่อศึกษาการพัฒนาเกม 3 มิติ ในหัวข้อต่าง ๆ ซึ่งได้แก่

1.2.1 ศึกษาการพัฒนาเกม 3 มิติ โดยใช้ Irrlicht Engine และ Audiere Engine ร่วมกับการเขียนโปรแกรมภาษา C++

1.2.2 เข้าใจการนำโปรแกรม 3ds max มาใช้ในการออกแบบโมเดลรูปภาพ

1.3 ขอบเขตของการศึกษา

1.3.1 พัฒนาเกม 3 มิติจาก Irrlicht Engine และ Audiere Engine

1.3.2 สร้างเกมแนว Action Adventure ส่วนรูปแบบมุมมองของเกมเป็นแบบ Third-Person Shooter (มุมมองผู้เล่นที่ 3)

1.4 ประโยชน์ที่ได้รับจากการศึกษา

ปัญหาพิเศษเรื่องนี้ทำให้ได้รับทราบและเรียนรู้ถึงขั้นตอนต่างๆ และโครงสร้างในการผลิตเกมว่ามีการทำงานอย่างไร วิธีการสร้างภาพตัวละครและฉาก 3 มิติจากโปรแกรม 3ds max และได้รับรู้ถึงชุดคำสั่งของ Irrlicht Engine และ Audiere Engine และเรียนรู้การเขียนโปรแกรมด้วยภาษา C++ ที่ใช้ในการเขียนเกม ได้แก่ การสร้างหน้าต่างWindows , การติดต่อกับฮาร์ดแวร์(hardware) เช่น Sound Card , คีย์บอร์ด (keyboard) และเมาส์ (mouse) , การโหลดตัวละคร , การเคลื่อนไหวของตัว

ละคร รวมถึงการเรนเดอร์ภาพ อีกทั้งยังมีการนำปัญญาประดิษฐ์มาประยุกต์ใช้ในการสร้างเกมด้วยเช่นกัน

1.5 ขั้นตอนของการศึกษา

1.5.1 ออกแบบเกมและเนื้อเรื่องของเกม

1.5.2 ออกแบบตัวละครของเกม

1.5.3 ศึกษาการสร้าง 3ds max และสร้างตัวละคร

1.5.4 ศึกษาการใช้เอนจิน (engine) และทำการเขียนโปรแกรมสร้างเกม

1.5.5 ทำการทดสอบเกม

1.5.6 ปรับปรุงและแก้ไขข้อผิดพลาดของเกม

1.5.7 จัดทำส่วนของการติดตั้งโปรแกรม และบันทึกลงสื่อ

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 Microsoft Visual C++ 6.0

Microsoft Visual C++ 6.0 เป็นภาษาการโปรแกรมมิงเชิงวัตถุ (Object – Oriented Programming) แบบ GUI (Graphic User Interface) ตัวหนึ่งจากทางบริษัทไมโครซอฟต์ เป็นเครื่องมือพัฒนาโปรแกรมที่มีความสามารถสูงในยุคนี้ Microsoft Visual C++ ได้รับการพัฒนาให้มีความยืดหยุ่น และมีประสิทธิภาพสูงขึ้นมาจากภาษา C++ และได้รับการพัฒนาโปรแกรมในหลายๆ ด้านไม่ว่าจะเป็นการสร้างโปรแกรมทั่วไป , การสร้างโปรแกรมการจัดการฐานข้อมูล , การสร้างโปรแกรมบนระบบเครือข่าย หรือมัลติมีเดีย อย่างครบครัน

ในปัจจุบัน Microsoft Visual C++ ได้รับการพัฒนาจนถึงเวอร์ชัน.NET มีลักษณะเป็น IDE(Integrated Development Environment) คือเป็นโปรแกรมซึ่งมีไว้สำหรับเพิ่มความสะดวกในการสร้าง และแก้ไขโปรเจคให้ง่ายขึ้น โดยจะมีเครื่องมืออำนวยความสะดวกในการพัฒนาโปรแกรมต่างๆ ให้เรียกใช้งานใน IDE เช่นโปรแกรมที่ใช้ในการดีบัก, คอมไพล์เลอร์ และลิงค์เกอร์ เป็นต้น นอกจากนี้ยังมีส่วนรองรับการพัฒนาโปรแกรมบนวินโดว์ โดยมี MFC (Microsoft Foundation Class) เป็นไลบรารีที่ช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดว์

นอกจากนี้ Microsoft Visual C++ ยังมีส่วนที่สนับสนุนทางด้านของมัลติมีเดีย เกี่ยวกับรูปภาพ , การเล่นไฟล์ภาพเคลื่อนไหวรวมถึงทางด้านของเสียง (Sound) ทั้งยังมีส่วนของเครื่องมือที่สนับสนุนการใช้งานที่เกี่ยวข้องกับบัพเฟอร์อินพุต เอาท์พุต และริฟฟ์ไฟล์ (Riff files) ที่เรียกกันว่า มัลติมีเดียไฟล์อินพุตเอาท์พุต เซอร์วิสด์ (Multimedia file I/O services) อีกด้วย โดยมีความสามารถ และเครื่องมือต่างๆให้ใช้งานมากมายหลายคอมโพเนนต์ ซึ่งใน Visual C++ นั้นประกอบไปด้วยคอมโพเนนต์ต่างๆ ดังนี้

2.1.1 VC++ Developer Studio

ทำหน้าที่เป็นศูนย์กลางในการสร้างแอพพลิเคชัน ซึ่งจะประกอบไปด้วยองค์ประกอบย่อยๆ ภายในดังนี้

Project Manager ทำหน้าที่จัดการกับการสร้างแอพพลิเคชันในลักษณะของโปรเจค (แอพพลิเคชันหนึ่งๆ ประกอบด้วยองค์ประกอบหลายๆ ส่วนซึ่งรวมเรียกว่าโปรเจค)

Text Editor ทำหน้าที่ในการเขียนโปรแกรม โดยจะเรียกโปรแกรมที่เขียนว่า Source Code

Resource Editor	ทำหน้าที่ออกแบบหน้าตาโปรแกรม เช่น เมนู ไดอะล็อกต่างๆ และไอคอน
Wizard	เพื่อสร้างแอพพลิเคชันในลักษณะวิซาร์ดต่างๆ เช่น AppWizard และ ClassWizard จะช่วยให้สร้างโค้ดพื้นฐานแก่แอพพลิเคชันอย่างรวดเร็ว โดยเพียงแต่กำหนดคลาส C++ จัดการกับวินโดว์ แมสเสจ และกระทำงานอื่น ๆ เพิ่มเติมอีก
Compiler	จะทำการคอมไพล์อย่างอัตโนมัติ มีการลิงค์ (Link) ไฟล์ต่าง ๆ เข้าด้วยกัน
Debugger	เพื่อแก้ไขความผิดพลาดของแอพพลิเคชันโดยผ่าน Debugger
Online Help	รายละเอียดความช่วยเหลือ (กรณีติดตั้ง MSDN แล้ว)

2.1.2 VC++ Runtime Libraries

คอมโพเนนต์นี้ทำหน้าที่เก็บฟังก์ชันมาตรฐานต่างๆของ ANSI C เช่น ฟังก์ชัน sin ซึ่งสามารถเรียกใช้ภายในโปรแกรม C หรือ C++

2.1.3 VC++ MFC และ Template Libraries

MFC เป็นไลบรารีคลาส C++ ที่ถูกสร้างมาโดยเฉพาะสำหรับการสร้างแอพพลิเคชันเพื่อใช้งานกับ Windows ซึ่งมีรูปแบบการติดต่อกับผู้ใช้แบบกราฟิก หรือที่เรียกว่า GUI (Graphic User Interface) ซึ่งคลาส MFC ทำให้การเขียนโปรแกรมง่ายขึ้น และยังช่วยประหยัดเวลาในการเขียนโค้ต

นอกจากนี้ยังติดตั้ง ATL (Active Template Libraries) ซึ่งเป็นชุดของคลาส C++ ที่เป็นเทมเพลตหรือต้นแบบ ที่จะช่วยอำนวยความสะดวกในการสร้าง ActiveX Control และรวมทั้งรูปแบบการสร้างแอพพลิเคชันอื่นๆ ที่เป็นออปเจกต์ COM (Component Object Model)

2.1.4 VC++ Build Tools

ประกอบด้วยคอมไพล์เลอร์ C/C++ , Linker , คอมไพล์เลอร์ Resource (สำหรับเตรียมการเกี่ยวกับ Resource โปรแกรมต่างๆ เช่น เมนู ไดอะล็อกต่างๆ และไอคอน เป็นต้น) รวมทั้งเครื่องมืออื่นๆ ที่จำเป็นต่อการสร้างแอพพลิเคชันสำหรับวินโดว์ (การสร้างแอพพลิเคชันแบบ Win32)

2.1.5 Active X

เป็นซอฟต์แวร์ย่อย ๆ (Software Component) หรือองค์ประกอบย่อยๆ ที่เพิ่มเข้าไปในแอพพลิเคชันที่สร้างขึ้น ซึ่ง ActiveX นั้นทำให้ไม่จำเป็นต้องสร้างทุกส่วนของแอพพลิเคชันเอง เพียงแต่เลือกใช้องค์ประกอบย่อยๆ ที่เหมาะสมกับงานเพื่อสร้างเป็นแอพพลิเคชันที่สมบูรณ์

2.1.6 Data Access

เป็นคอมโพเนนต์ที่รวมไดรเวอร์ฐานข้อมูลชนิดต่างๆ ไว้ให้สามารถสร้างแอพพลิเคชันที่เชื่อมต่อกับฐานข้อมูลได้อย่างสะดวก และนอกจากไดรเวอร์แล้วยังประกอบไปด้วยคอนโทรล และเครื่องมืออื่นๆ ที่ช่วยในการสร้างแอพพลิเคชันกับฐานข้อมูลทำได้อย่างรวดเร็ว

2.1.7 Enterprise Tools

เป็นคอมโพเนนต์ที่ประกอบด้วยคอมโพเนนต์ย่อยๆ อีกดังต่อไปนี้

- Microsoft Visual SourceSafe 6.0 Client
- Application Performance Explorer
- Repository
- Visual Component Manager
- Self-installing .exe redistributable files
- Visual Basic Enterprise Components
- VC++ Enterprise Tools
- Microsoft Visual Modeler
- Visual Studio Analyzer

2.1.8 Graphics

เป็นคอมโพเนนต์เกี่ยวกับรูปภาพในรูปแบบต่างๆ เช่น metafile, bitmap, cursor และ icon รวมทั้ง video clip ซึ่งคอมโพเนนต์เหล่านี้ใช้ในการเขียนโปรแกรมเกี่ยวกับกราฟิกต่างๆ

2.1.9 Tools

เป็นคอมโพเนนต์ที่เป็นเครื่องมือเสริมการทำงานชนิดต่างๆของ Visual C++ ดังนี้

- API Text Viewer
- MS Info
- MFC Trance Utility
- Spy++
- Win32 SDK Tools
- OLE/COM Object Viewer
- ActiveX Control Test Container

2.2 การเขียนโปรแกรมด้วย Visual C++ บน Window

ภาษา C++ ซึ่งใช้โครงสร้างภาษาเหมือนกับภาษาซีทุกอย่าง แต่ได้เพิ่มความรัดกุมมากขึ้นในเรื่องของการใช้คำสั่งและฟังก์ชัน พร้อมกับได้เพิ่มความสามารถในการประกาศคลาส ซึ่งเป็นตัวแปรในแบบการเขียนโปรแกรมเชิงวัตถุ และหลังจากที่ภาษา C++ และแนวคิดของการเขียนโปรแกรมเชิงวัตถุ เป็นที่ยอมรับ และได้นำไปใช้ในการพัฒนาโปรแกรมต่างๆ มากมาย ไม่ว่าจะเป็นตัวระบบปฏิบัติการหรือโปรแกรมประยุกต์ ทั้งบนวินโดวส์และบนระบบปฏิบัติการอื่นๆ เช่น ลีนุกซ์ (Linux) และก็ยังมีการพัฒนาตัวแปลภาษาให้มีความสามารถสูงขึ้นๆ ตามความสามารถของระบบปฏิบัติการ เช่น Visual C++ ,Visual Studio, Visual Studio .NET เป็นต้น แต่อย่างไรก็ตามพื้นฐานของภาษาซีและการเขียนโปรแกรมแบบการเขียนโปรแกรมเชิงวัตถุใน C++ ก็ยังเป็นรากฐานในการพัฒนาภาษาใหม่ๆ ขึ้นมาอีกหลายภาษา เช่น จาวา (Java) , ซีชาร์ป (C#) เป็นต้น

การที่เราจะพัฒนาโปรแกรมภาษา C++ ได้นั้น จะต้องมีโปรแกรมสำหรับแปลภาษาก่อน นั่นก็คือ Turbo C++ ,Borland C++ ,Microsoft C/C++ ,Visual C++ ฯลฯ โปรแกรมแปลภาษาเหล่านี้ สามารถแปลโปรแกรมได้ทั้งภาษาซีและภาษา C++ ซึ่งก็ขึ้นอยู่กับว่าเรากำหนดให้นามสกุลของไฟล์เป็นภาษาอะไร ถ้าเขียนโปรแกรมภาษาซี ให้กำหนดนามสกุลของไฟล์โปรแกรมเป็น .C แต่ถ้าเขียนโปรแกรมที่มีการประกาศคลาสหรือใช้ความสามารถของการเขียนโปรแกรมเชิงวัตถุด้วย ให้กำหนดนามสกุลของไฟล์โปรแกรมนั้นให้เป็น .CPP

2.2.1 ฟังก์ชันและวิธีการสร้างฟังก์ชัน

- **การประกาศฟังก์ชัน**

ฟังก์ชันต้นแบบ หรือ Prototype Function นี้มีไว้เพื่อบอกให้ตัวแปลภาษารู้ว่า ในโปรแกรมนี้ยังมีฟังก์ชันชื่อว่า call() อยู่ แต่อยู่ด้านล่างของ main() ดังนั้น เราเขียนโปรแกรมให้เป็นดังนี้

```
void call();
void main(){
        call();
}
void call(){
        //
}
```

หรืออีกวิธีหนึ่งก็คือ ให้ประกาศฟังก์ชัน call() เอาไว้ด้านบนก่อน main() ดังนี้

```
void call(){
        //
}
void main(){
        call();
}
```

- **การคืนค่าฟังก์ชัน**

เราสามารถกำหนดให้ฟังก์ชันส่งค่ากลับมายังจุดที่เรียกใช้คำสั่ง return แต่ก่อนที่จะใช้คำสั่ง return นั้นต้องตรวจสอบว่า จะส่งค่าประเภทใดกลับไป ถ้าเป็นค่าจำนวนเต็ม เราจะต้องเปลี่ยนคำว่า void ที่อยู่หน้าฟังก์ชันให้เป็น int ดังตัวอย่างต่อไปนี้

```
int call();
void main(){
        int a;
        a = call();
}
int call()
        return 5+6
}
```

จากตัวอย่าง จะเห็นได้ว่าฟังก์ชันแบบ int และคืนค่าให้ 5+6 (ซึ่งก็คือ 11) กลับไปยังจุดที่เรียกใช้ ดังนั้น ถ้าเราไม่เปลี่ยนจาก void ให้มาเป็น int เราก็จะไม่สามารถส่งค่ากลับไปได้

- **การส่งค่าไปให้ฟังก์ชัน**

เราสามารถส่งค่าให้กับฟังก์ชันได้ เพื่อให้ฟังก์ชันนำเอาค่าที่ส่งไปนี้ไปประมวลผล เราเรียกค่าที่ส่งนี้ว่า "พารามิเตอร์" หรือจะเรียกว่า "อากิวเมนต์" ก็น่าจะได้เหมือนกัน ดังตัวอย่างต่อไปนี้

```
#include <stdio.h>
int call(int a, int b);
void main(){
        printf("answer = %d\\n", call(10,20));
}
int call(int a, int b){
        return a+b;
}
```

จากตัวอย่างจะเห็นได้ว่าฟังก์ชัน call รับพารามิเตอร์ 2 ตัวคือ a กับ b เราเรียกโดยผ่านฟังก์ชัน printf ส่งค่า 10 และ 20 ไปให้กับ call() และมันก็จะคืนค่ากลับมาโดยเอาเลขทั้งสองบวกกัน เพราะฉะนั้น ฟังก์ชัน call() จึงคืนค่ากลับมาเป็น 30 แสดงออกทางจอภาพโดย printf

2.2.2 การเขียนโปรแกรมเชิงวัตถุ

ธรรมชาติของวัตถุ และทฤษฎีของการเขียนโปรแกรมเชิงวัตถุ

นิยามในเรื่องของการเขียนโปรแกรมเชิงวัตถุ คือ "การมองทุกอย่างให้เป็นวัตถุ" ซึ่ง "วัตถุ" แต่ละอย่างต่างก็มีชื่อเรียกที่แตกต่างกัน เช่น

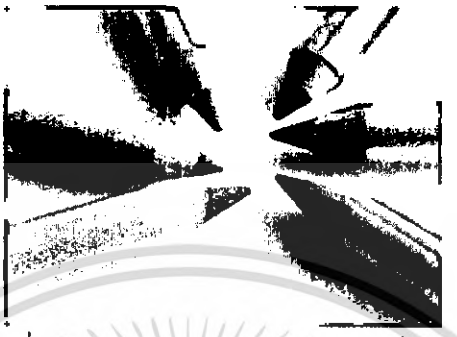
- โทรศัพท์ จะยี่ห้ออะไร แบบไหน เราเรียกวัตถุนี้ว่า โทรศัพท์
- แก้วน้ำ จะเป็นแก้วกระเบื้องหรือแก้วพลาสติก วัตถุนี้ก็คือแก้ว
- กระดาษ จะเป็นเศษกระดาษ หรือ A4 วัตถุนี้ก็คือ กระดาษ
- คน ไม่ว่าจะเป็นอาชีพใด สูงต่ำดำขาว ก็คือคนด้วยกันทั้งสิ้น
- รถ ไม่ว่ายี่ห้อใด แบบใด ประเภทใด ก็เรียกว่ารถ เช่น รถยนต์ รถบรรทุก ฯลฯ

ดังนั้นสามารถสรุปได้ว่า ในโลกนี้ ทุก ๆ อย่างล้วนแล้วแต่มี "ชื่อที่บอกว่าสิ่งนั้นคืออะไรหรือเป็นอะไร" ชื่อที่เราสมมติขึ้นมาเพื่อให้เรียกวัตถุต่างๆนั้นก็คือ "คลาส" (Class) นั่นเอง ถ้าเรามองสิ่งของรอบ ๆ ตัว ก็มีหลายคลาส เช่น

- คลาสของโทรศัพท์
- คลาสของดินสอ
- คลาสของยางลบ
- คลาสของคอมพิวเตอร์
- คลาสของเมาส์

- คลาสของพัดลม

สาเหตุที่มีคลาส หรือมีชื่อที่เรียกวัตถุตามลักษณะของวัตถุนั้นเพราะว่า ถ้าไม่มีชื่อที่เรียกในเวลาเราจะอ้างถึงสิ่งๆนั้นก็จะทำได้ยาก พิจารณารูปต่อไปนี้



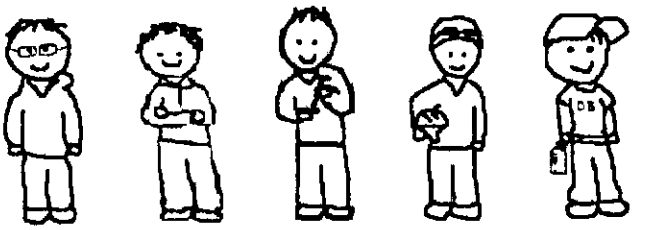
รูปที่ 2.1 แสดงตัวอย่างของออปเจกต์

จากรูป จะเห็นว่าเป็นวัตถุที่รูปร่างยาว ๆ เรียว ๆ มีไส้เป็นถ่าน ปลายด้านหนึ่งจะแหลมๆ ใช้เขียนบนวัสดุรองรับ เช่น กระดาษได้ วัตถุชิ้นนี้เรียกว่า "ดินสอ" และในคลาสของวัตถุใดๆ จะมีอยู่ 2 องค์ประกอบที่ใช้บอกถึงความเป็นวัตถุนั้น คือ

1. คุณลักษณะ (Attribute หรือ Properties) ก็คือ ลักษณะที่บอกถึงความเป็นสิ่ง ๆ นั้น
2. การกระทำ (Method หรือ Functions) ก็คือมันทำอะไรได้บ้าง

ยกตัวอย่างเช่น ดินสอ คุณลักษณะของมันก็คือ เรียว ยาว มีไส้เป็นถ่าน การกระทำของดินสอก็คือ เขียน หัก ปา จิ้ม แทง พูดง่ายๆ ก็คือ ดินสอทำอะไรได้ และเราก็ทำอะไรกับมันได้บ้าง เราสามารถใช้มันทำอะไรได้ตั้งแต่เขียน ทิ่มแทง ไปจนถึงใช้ทัดหู ก็ขึ้นอยู่กับการนำไปใช้งานนั่นเอง เพราะฉะนั้น ถ้าเกิดเราไปพบวัตถุใด ๆ ที่มีคุณลักษณะตรงกับคุณลักษณะของดินสอ และยังมีการกระทำที่เหมือนกัน เราจะสรุปได้เลยว่า วัตถุสิ่งนั้นก็คือ "ดินสอ" นั่นเอง

ในการเขียนโปรแกรมคอมพิวเตอร์ก็เช่นกัน เราจะต้องมองสิ่งที่อยู่โปรแกรมของเราให้เป็นเชิงวัตถุเพื่อที่จะได้ออกแบบวิธีการเก็บและจัดการข้อมูลที่เป็นอิสระต่อกัน แต่ก่อนที่จะเข้าสู่การเขียนโปรแกรมเชิงวัตถุ ในภาษา C++ นั้น คล้ายกับตัวแปรแบบโครงสร้าง (Structure) ในภาษาซี การประกาศตัวแปรแบบโครงสร้างนี้ เป็นแนวคิดพื้นฐานที่จะนำมาใช้ในการสร้างคลาสต่อไป ลักษณะการทำงานของตัวแปรแบบโครงสร้างจะช่วยแก้ไขปัญหาในเรื่องของความซับซ้อนของตัวแปรได้ เพราะเป็นการเปลี่ยนวิธีการเขียนโปรแกรม โดยอาศัยชุดของข้อมูลเป็นหลัก จากรูปเป็นตัวอย่างปัญหาในการเก็บข้อมูลของคน 5 คน ดังรูป



รูปที่ 2.2 แสดงตัวอย่างของแอททริบิวต์ รูปที่ 1

สมมติว่าเราจะต้องเขียนโปรแกรมให้เก็บข้อมูลของพวกเขาเหล่านั้นเอาไว้ ถ้าเราใช้การเขียนโปรแกรมภาษาซี ธรรมดา สามารถทำได้โดยใช้อะเรย์แบบ 1 มิติ โดยการประกาศตัวแปรตามสิ่งที่เราต้องการจะเก็บ คือ อายุ ส่วนสูง น้ำหนัก จำนวน 5 คน ดังนี้

```
int age[5];
double height[5];
double weight[5];
```

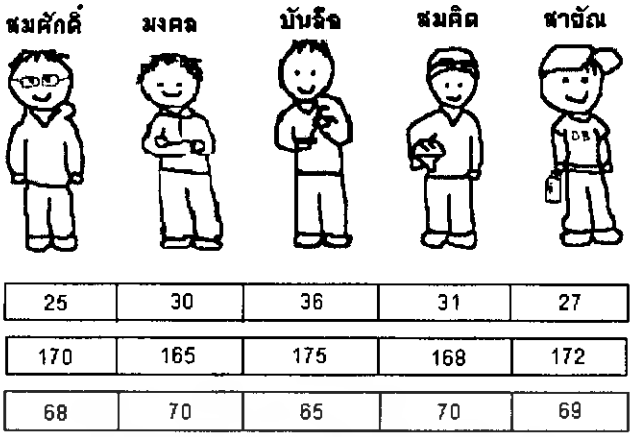
ถ้าเราจะนำเอาข้อมูลมาใส่ลงในตัวแปร age ,height และ weight นั้นเราก็จะอาศัยลำดับในอะเรย์เข้ามาช่วย นั่นคือ คนที่ 1 ก็จะเก็บลงไปในอะเรย์ช่องที่ 0 ในทุกตัวแปร ดังนี้

```
age[0] = 25;
height[0] = 170;
weight[0] = 68;
```

สำหรับคนที่ 2 ก็จะเก็บลงไปในอะเรย์ช่องที่ 1 ของทุกตัวแปร เช่นกัน ดังนี้

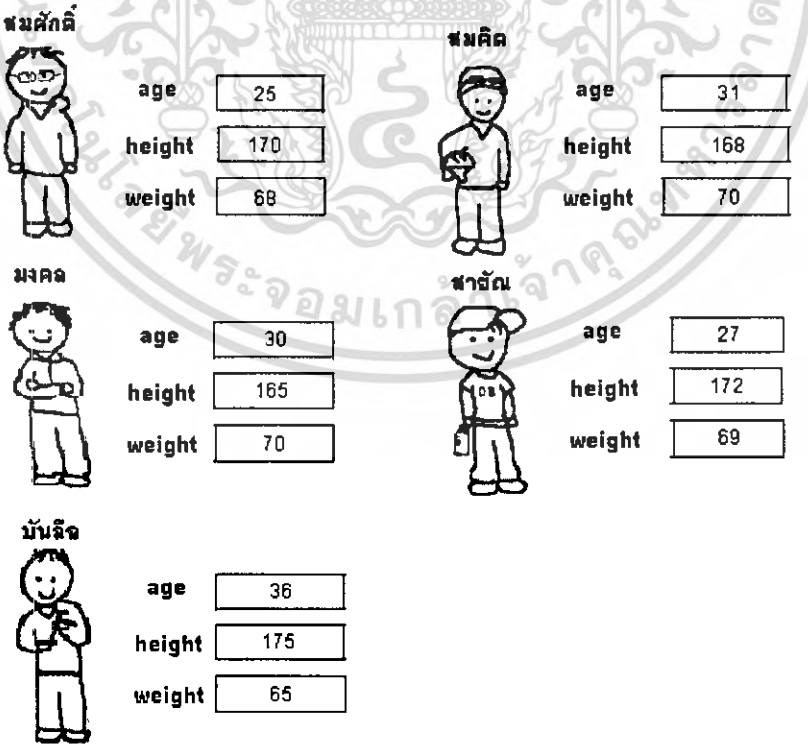
```
age[1] = 30;
height[1] = 165;
weight[1] = 70;
```

สำหรับคนที่ 3 ,4 และ 5 ก็จะเก็บลงไปในอะเรย์ช่องที่ 2,3 และ 4 ของทุกๆตัวแปรตามลำดับ เช่นกัน จากตัวอย่างนี้ เราจะสรุปได้ว่า การใช้อะเรย์ในการเก็บข้อมูลที่มีจำนวนมากๆ และมีตัวแปรหลายๆตัวค่อนข้างจะยุ่งยากพอสมควร เพราะเราใช้ลำดับของอะเรย์เป็นตัวกำหนดลำดับการเก็บข้อมูล ถ้าเราต้องการเรียกดูข้อมูลของคนที่ 1 เราก็จะต้องแสดงข้อมูลที่อยู่ในช่องที่ 0 ของอะเรย์ทุกตัวออกมา รูปแบบของการเก็บข้อมูลในลักษณะนี้ เป็นดังรูป



รูปที่ 2.3 แสดงตัวอย่างของแอททริบิวต์ รูปที่ 2

จะเห็นว่า อะเรย์ 1 ตัวที่เก็บอายุ ก็จะเก็บอายุทุก ๆ ช่อง ตัวแปรที่เก็บส่วนสูง ก็จะเก็บแต่ส่วนสูงทั้งหมด วิธีการลักษณะนี้เสี่ยงต่อการผิดพลาดอย่างมากในเรื่องของลำดับ เพราะตัวแปรแต่ละตัวมีความอิสระต่อกัน การอ้างลำดับผิดในกรณีที่มีปริมาณข้อมูลมากๆ เช่นเก็บข้อมูลเป็นร้อยๆ ข้อมูลขึ้นไป การไล่ลำดับจะต้องไล่ให้ตรงกัน ซึ่งจากปัญหาที่ได้กล่าวไปนี้ เราจะต้องใช้ตัวแปรแบบโครงสร้างมาช่วยแก้ปัญหา การใช้งานตัวแปรโครงสร้างนี้ จะคล้าย ๆ กับการที่เราสร้างตัวแปรชนิดใหม่ขึ้นมาเป็นของเราเอง โดยพิจารณาจากข้อมูลที่เราต้องการจัดเก็บ รูปแบบของการเก็บข้อมูลแสดงได้ดังรูป



รูปที่ 2.4 แสดงตัวอย่างของแอททริบิวต์ รูปที่ 3

การประกาศตัวแปรโครงสร้าง เราจะใช้คำสั่ง typedef struct ดังนี้

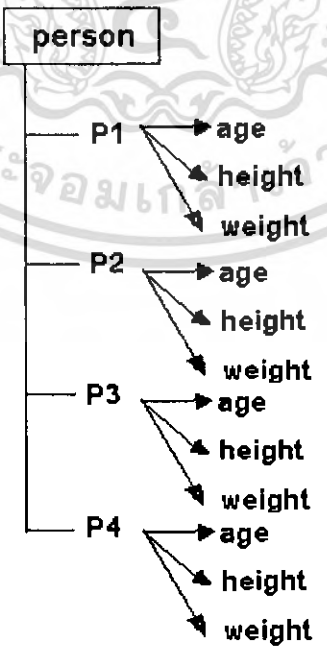
```
typedef struct_person{
        int age;
        int height;
        int weight;
}person;
```

จากข้างต้น เป็นการสร้างตัวแปรแบบโครงสร้างชนิดใหม่ขึ้นมา ชื่อว่า person เราจะถือว่าเป็นการสร้างตัวแปรแบบใหม่ และตัวแปรนี้สามารถเก็บค่าได้ 3 ค่า คือ อายุ (age) , น้ำหนัก (weight) และส่วนสูง (height) เมื่อเราประกาศตัวแปรแบบใหม่นี้เอาไว้ในโปรแกรมของเราแล้ว เราก็สามารถสร้างตัวแปรแบบ person นี้ได้ทันที โดยเขียนโค้ดดังนี้

```
person p1,p2,p3,p4
```

ตอนนี้เราได้ตัวแปร p1,p2,p3 และ p4 เป็นตัวแปรแบบ person แล้ว

สำหรับตัวอย่างนี้ก็เช่นกัน คือ เมื่อเราได้สร้างตัวแปรโครงสร้างชนิดใหม่ขึ้นมาแล้ว โดยชื่อว่า person เราก็สามารถที่จะสร้างตัวแปรแบบ person ขึ้นมาได้ และตัวแปรที่ถูกสร้างขึ้นมานี้ จะมีคุณลักษณะความเป็นตัวแปรแบบ person ทุกอย่าง คือ ภายในตัวมันเอง สามารถเก็บค่าย่อยๆได้อีก 5 ค่าดังรูป



รูปที่ 2.5 แสดงคุณสมบัติของแอททริบิวต์

ลักษณะของตัวแปรแบบโครงสร้างนี้ จะจัดเก็บข้อมูลเป็นหมวดหมู่ และแบ่งแยกชัดเจนมากขึ้นอย่างเห็นได้ชัดว่า age ,height และ weight ก็เป็นของคน ๆ นั้นเลย เพราะฉะนั้นมันจะไม่ปะปนกันคือ age ของ p1 จะเป็นของ p1 โดยเฉพาะ อาจจะไม่เท่ากับ age ของ p2 ก็ได้ เพราะฉะนั้นด้วยหลักการของโครงสร้างนี้ เมื่อเรานำมาใช้ในการเก็บข้อมูลอายุ , น้ำหนัก และส่วนสูงของคน 5 คน ดังที่ได้ยกตัวอย่างไว้ในข้างต้น ถ้าเขียนด้วยภาษาซีก็จะเขียนลักษณะดังนี้

```
typedef struct  _person{
        int age;
        int height;
        int weight;
}person;
person p[5];

p[0].age = 25;
p[0].height = 170;
p[0].weight = 68;

p[1].age = 30;
p[1].height = 165;
p[1].weight = 70;

p[2].age = 36;
p[2].height = 175;
p[2].weight = 65;

p[3].age = 31;
p[3].height = 168;
p[3].weight = 70;
```

จากตัวอย่าง จะเหมือนกับว่าเรามีตัวแปรทั้งหมด 15 ตัว และแต่ละตัวก็ถูกจัดหมวดหมู่แบ่งออกเป็นคน เป็นข้อมูลเฉพาะส่วนๆ ไป เราสามารถเข้าถึงข้อมูลที่เป็นอายุของคนที่ต้องการได้โดยอ้างอิงลำดับของคน ๆ นั้นและตามด้วยชื่อตัวแปรที่เราต้องการ เช่น ต้องการอายุของนายมงคล ซึ่งลำดับที่สอง ก็คือ p[1].age เป็นต้น จะเห็นได้ว่า วิธีการนี้ เป็นการเปลี่ยนแนวคิดและมุมมองจากอะเรย์ให้มาเป็นแนวคิดที่คล้าย ๆ กับเชิงวัตถุเลย คือ เรามองแต่ละคน เป็นวัตถุมากขึ้น 1 คน ก็จะมีอายุ น้ำหนัก และส่วนสูง หรือมีข้อมูลมากกว่านี้ใน 1 โครงสร้าง ทำให้การจัดการข้อมูลทำได้ง่ายกว่าการใช้อะเรย์แบบในวิธีแรกที่ได้นำเสนอไป

- **การสร้างคลาส**

การสร้างคลาสใน C++ เราจะเขียนคล้ายๆ กับการประกาศตัวแปรแบบโครงสร้างดังที่ได้กล่าวไปแล้ว คลาสจะเป็นโครงสร้างที่รวมการทำงานทั้งหมดเข้าด้วยกัน ให้เป็นวัตถุก้อนเดียวกัน ดูตัวอย่างโปรแกรม EASY.CPP ต่อไปนี้

```
#include

class Person{
        int age;
        void setAge(int a){
                age = a;
        }
        void showAge(){
                printf("age = %d\\n",age);
        }
};


void main(){
        Person p;
}
```

โปรแกรม EASY.CPP ในข้างต้น มีการประกาศคลาส Person เอาไว้ ภายในคลาสมีสมาชิกดังนี้

1. ตัวแปร age

2. ฟังก์ชัน setAge(int a)

3. ฟังก์ชัน showAge()

คลาส Person ที่ยกตัวอย่างมานี้ แสดงให้เห็นถึงการจัดเก็บข้อมูล ก็คือ อายุ และมีฟังก์ชัน setAge ที่ใช้ในการกำหนดค่าอายุโดยรับพารามิเตอร์จากภายนอก และมีฟังก์ชัน showAge ที่ใช้ในการแสดงค่าอายุออกมาทางจอภาพโดยฟังก์ชัน printf และในฟังก์ชัน main() ก็ได้มีการประกาศออปเจกต์ของคลาส Person นี้ชื่อว่า p เอาไว้ จากนั้นก็จบโปรแกรม โปรแกรมนี้เป็นตัวอย่างที่แสดงให้เห็นถึงการประกาศคลาส ซึ่งมีรูปแบบดังนี้

```
class ชื่อคลาส{
    //ประกาศตัวแปร
    //ประกาศฟังก์ชัน
}
```

ในคลาส หนึ่ง ๆ จะมีแค่การสร้างตัวแปรหรือฟังก์ชันเพียงอย่างเดียวก็ได้ หรือเขียนแค่คลาสเพียงอย่างเดียว มีแต่ {...} ไม่ต้องมีตัวแปรหรือฟังก์ชันไว้เลยก็ได้

จากโปรแกรม EASY.CPP ในข้างต้นนั้น เราจะมาทำความเข้าใจกันดังนี้ ในคลาส Person มีตัวแปร age ตัวหนึ่งและมีฟังก์ชัน setAge ฟังก์ชันนี้รับค่าพารามิเตอร์ 1 ตัวคือ int a และเมื่อรับเข้ามาแล้ว ก็จะทำการกำหนดให้กับตัวแปร age ที่อยู่ภายในคลาส และอีกฟังก์ชันหนึ่งก็คือฟังก์ชัน showAge ซึ่งเป็นฟังก์ชันที่ใช้ในการแสดงค่า age ออกทางจอภาพ จะเห็นได้ว่าการทำงานจะอยู่ภายในคลาสเท่านั้น จบแค่ภายในคลาส เมื่อเราประกาศออปเจกต์ของคลาสดังนี้

```
Person p;
```

ก็เท่ากับว่า ตอนนี้คลาส Person มีทายาทแล้วนั่นคือออปเจกต์ p นั่นเอง และ p นี้ก็มีคุณลักษณะทุกอย่างเช่นเดียวกับคลาส Person นี้เลย นั่นก็คือ p สามารถเรียกใช้ตัวแปร a ,ฟังก์ชัน setAge และ showAge ได้ ให้เราแก้ไขโปรแกรม EASY.CPP ให้เป็นดังนี้

```
void main(){
        Person p;
        p.setAge(25);
        p.showAge();
}
```

จากข้างต้น เป็นการสั่งให้ p เรียกฟังก์ชัน setAge โดยใส่พารามิเตอร์ 25 ให้กับฟังก์ชัน เป็นการเรียกฟังก์ชันที่อยู่ในคลาส Person เอง จากนั้นก็เรียกฟังก์ชัน showAge ตามลำดับ เพื่อแสดงค่าอายุในตัวแปร age ออกมาปัญหานั่นก็คือเมื่อเราแปลโปรแกรมนี้โดยใช้ตัวแปรภาษา C++ ใด ๆ จะพบกับข้อความผิดพลาดลักษณะนี้

```
\'setAge\' : cannot access private member declared in class \'Person\'
\'showAage\' : cannot access private member declared in class \'Person\'
```

จากข้อความที่ปรากฏในข้างต้น แปลแล้วได้ความว่า ฟังก์ชัน setAge และฟังก์ชัน showAge ไม่สามารถเรียกใช้ได้ ข้อความผิดพลาดจะฟ้องออกมาในบรรทัดนี้

```
p.setAge(25);
p.showAge();
```

ที่เป็นเช่นนี้ก็เพราะว่ากฎของ OOP ข้อหนึ่งก็คือ คลาสจะมีการปกป้องข้อมูลไม่ให้กระบวนการภายนอกคลาสใดๆ เข้าถึงสมาชิกในคลาสได้เลย เมื่อเราประกาศออปเจกต์ p ซึ่งออปเจกต์นี้เป็นตัวแปรที่อยู่ภายนอกคลาส ดังนั้น p จึงไม่สามารถเข้าถึงตัวแปร age และฟังก์ชัน setAge กับ showAge ได้ คุณสมบัติดังกล่าวเป็นคุณสมบัติของการเขียนโปรแกรมเชิงวัตถุที่ช่วยให้สมาชิกภายในคลาสไม่ถูกรบกวน ทำให้คลาสมีคุณสมบัติปกป้องข้อมูล หรือ Encapsulation ได้นั่นเอง ซึ่งเป็นสิ่งที่แตกต่างออกไปจากตัวแปรแบบโครงสร้างที่เห็นได้ชัด ถ้าเราจะทำให้ออปเจกต์ p สามารถเรียกใช้สมาชิกในคลาสได้ ผู้เขียนโปรแกรมจะต้องอนุญาตให้สมาชิกในคลาสตัวที่ต้องการ มีคุณสมบัติเป็น "สาธารณะ" หรือ public ดังนี้

```
class Person{
        int age;
public:
        void setAge(int a){
                age = a;
        }
        void showAge(){
                printf("age = %d\\n",age);
        }
};
```

```
void main(){
        Person p;
        p.setAge(25);
        p.showAage();
}
```

จากข้างต้น ในตอนนี้ฟังก์ชันทั้งสองในคลาสได้กลายเป็น public แล้ว ดังนั้น จึงสามารถเรียกใช้ p.setAage และ p.showAge ได้อย่างสมบูรณ์

จากตัวอย่าง EASY.CPP ในข้างต้น จะเห็นได้ว่า ไม่สามารถเข้าถึงออปเจกต์ภายในคลาสได้ทันที ออปเจกต์จะเรียกใช้ฟังก์ชันหรือตัวแปรในคลาสได้จะต้องกำหนดให้เป็นแบบ public ซึ่งตรงจุดนี้เป็นความสามารถของการเขียนโปรแกรมเชิงวัตถุที่จะช่วยให้เกิดความปลอดภัยของข้อมูลในคลาสมากขึ้น ระดับในการเข้าถึงข้อมูลในคลาสมี 3 ระดับ คือ

- ส่วนตัว หรือ Private
- สาธารณะ หรือ Public
- ปกป้อง หรือ Protect

ถ้าเราสร้างคลาสขึ้นมา โดยไม่ได้ระบุว่าเป็น private, public หรือ protect ใดๆ สมาชิกภายในคลาสก็จะเป็นแบบ private ทั้งหมด

พิจารณาคลาสต่อไปนี้

```
class Person{
        int age;
        void setAge(int a){
                age = a;
        }
        void showAge(){
                printf("age = %d\\",age);
        }
};
```

สมาชิกทุกตัวในคลาสเป็นแบบ private หมด สมาชิกแบบ private นั้น จะอนุญาตให้กระบวนการที่ดำเนินการอยู่ภายในคลาสเดียวกันเรียกใช้งานกันได้ ในคลาส Person จะเห็นว่ามีฟังก์ชัน setAge และ showAge ซึ่งต่างก็เรียกใช้ตัวแปร age ได้ พูดง่ายๆ ก็คือ ตัวแปรและฟังก์ชันนั้นสามารถเรียกใช้กันภายในคลาสได้อย่างปกติทุกอย่าง แต่ถ้าเกิดเราสร้างออปเจกต์ของคลาสขึ้นมา และเรียกใช้ตัวแปรที่อยู่ในคลาส

```
p.age = 10;
```

ไม่สามารถเรียกใช้แบบนี้ได้ ทั้งนี้เป็นเพราะว่าออปเจกต์ p เป็นตัวแปรที่ถูกสร้างขึ้นมาใหม่อยู่ภายนอกคลาส สำหรับการประกาศให้เป็น public นั้น เราจะต้องเขียนคำว่า public: ไว้ที่บรรทัดก่อนที่จะทำการประกาศตัวแปรหรือฟังก์ชันนั้น เช่น

```
class Person{
public:
        int age;
        void setAge(int a){
                age = a;
        }
        void showAge(){
                printf("age =  %d\\n",age);
        }
};
```

2.3 Direct X

2.3.1 DirectX คืออะไร

ก่อนที่จะมี DirectX เกิดขึ้นนั้น นักพัฒนาเกมคอมพิวเตอร์สำหรับ DOS หรือ วินโดวส์ จะต้องเขียนเกม คอมพิวเตอร์ให้รู้จักกับฮาร์ดแวร์ ซึ่งมีอยู่มากมายในท้องตลาด ซึ่งในกรณีที่มีฮาร์ดแวร์ตัวใหม่เกิดขึ้น อาจเกิดปัญหาความไม่สนับสนุนกันระหว่างเกมคอมพิวเตอร์ กับฮาร์ดแวร์ตัวใหม่นั้น นักเล่นเกมจะต้องรอจนกว่านักพัฒนาเกมจะทำการอัพเดตเกมนั้นๆ ให้ใช้ความสามารถของฮาร์ดแวร์ตัวใหม่ได้ เมื่อเป็นเช่นนี้ไมโครซอฟต์จึงได้ทำการพัฒนาเทคโนโลยีที่มีความสามารถในการเป็นสื่อกลางติดต่อระหว่างเกมคอมพิวเตอร์ หรือโปรแกรมมัลติมีเดียต่างๆ กับ ฮาร์ดแวร์ขึ้นมาโดยใช้ชื่อว่า DirectX

DirectX ตามความหมายจะหมายถึง ไลบรารีคำสั่ง (Run Time Library) ที่ช่วยทำงานด้านมัลติมีเดีย Graphic โดยตัว DirectX Foundation จะมีส่วนประกอบที่เรียกว่า HAL (Hardware Abstraction Layer) จะใช้ซอฟต์แวร์ในการตรวจสอบความสามารถของฮาร์ดแวร์ที่อยู่ในเครื่องคอมพิวเตอร์นั้นอย่างอัตโนมัติ แล้วนำมากำหนดพารามิเตอร์ของแอพลิเคชันให้ตรงตามความเหมาะสมระหว่างเกมคอมพิวเตอร์ หรือโปรแกรมมัลติมีเดียกับไดร์เวอร์ของฮาร์ดแวร์ที่มีส่วนเกี่ยวข้องกับเกมหรือโปรแกรมนั้น ทำให้การประมวลผลโปรแกรมทำได้เร็วขึ้น เนื่องจากขั้นตอนต่างๆ จะถูกนำไปประมวลผลโดยตรง ไม่ต้องอาศัยตัวกลางอย่างเช่น GDI (Graphic Device Interface) ก่อน ทำให้นักพัฒนาเกมคอมพิวเตอร์สามารถเขียนโปรแกรมให้สื่อสารกับ DirectX เท่านั้นก็เพียงพอ นอกจากนี้ DirectX Foundation ยังมีส่วนประกอบที่เรียกว่า HEL (Hardware emulation Layer) ทำให้สามารถใช้โปรแกรมมัลติมีเดีย หรือโปรแกรมที่เกี่ยวข้องกับ 3D บน Hardware ที่ไม่สนับสนุนการใช้งานทางด้าน 3 มิติ โดยจะทำการจำลองความสามารถบางอย่างที่ฮาร์ดแวร์ตัวนั้นไม่มี ให้สามารถใช้งานได้กับโปรแกรมที่ต้องการ แม้จะมีข้อเสียอยู่บ้างตรงที่ อาจทำให้ช้าลงบ้างก็ตาม แต่ก็คุ้มค่ากับความสามารถของ DirectX ที่มีอยู่ในปัจจุบัน

DirectX ถูกใส่ไว้ให้เป็นส่วนหนึ่งตั้งแต่ Windows 98 และ Windows 2000 และส่วนประกอบอื่น ๆ ของ DirectX จะสามารถติดตั้งเพิ่มเติมได้ในภายหลัง ด้วยจุดประสงค์ที่จะให้ผู้พัฒนาโปรแกรม (developers) มีชุดคำสั่ง และส่วนประกอบต่าง ๆ ร่วมกัน เพื่อที่จะ

- ทำให้โปรแกรม multimedia ที่ผู้พัฒนาผลิตขึ้นมา สามารถทำงานได้กับ PC ที่ใช้ windows โดยไม่คำนึงถึงระบบ hardware ที่ใช้ และยืนยันว่า โปรแกรมจะสามารถใช้ประสิทธิภาพสูงสุดที่ hardware มี เพื่อให้ได้ผลงานที่มีคุณภาพสูงสุด
- ทำให้ผู้พัฒนา สามารถใช้อุปกรณ์การพัฒนาช่วยเหลือการเขียนโปรแกรม ทำให้การเขียนโปรแกรม multimedia สามารถทำได้โดยง่าย และยังสามารถผสมสื่อหลากชนิดเข้าไว้ด้วยกันได้

2.3.2 ประวัติความเป็นมาของ DirectX

ในช่วงที่ระบบปฏิบัติการไมโครซอฟต์วินโดวส์เริ่มต้นขึ้นมานั้น สิ่งที่เป็นเอกลักษณ์ของวินโดวส์ ก็คือเรื่องของ GUI (Graphical User Interface) โดยมี GDI (Graphical Device Interface) เป็นเครื่องมือสำหรับการจัดการทางด้านภาพ และเกมทั่วไปก็ยังคงพัฒนาภายใต้ระบบปฏิบัติการเดิมอยู่ ซึ่งก็คือระบบปฏิบัติการ "DOS" นั่นเอง

การเริ่มต้นครั้งแรกสำหรับไมโครซอฟต์ คือ "เครื่องมือ Win-G" สำหรับ Window95 และดูเหมือนว่าเครื่องมือตัวนี้จะไม่ได้รับการตอบรับที่ดีนักจากผู้พัฒนาเกมทั่วโลก ยังผลให้ไมโครซอฟต์เองต้องปรับปรุงรูปแบบใหม่ให้กับเครื่องมือตัวนี้อีกครั้ง

ไมโครซอฟต์ได้ทำการจัดตั้งทีมงานขึ้นมาใหม่เพื่องานกราฟิก งานมัลติมีเดีย งานเน็ตเวิร์ก งานด้านการรับข้อมูล Input และ งานกราฟิก 3 มิติ (ชื่อเทคโนโลยี "RenderGraphic") โดยใช้ชื่อว่า "DirectX"

DirectX 2.0 สามารถเข้ามาจัดการงานด้านเกมได้เป็นอย่างดีการทำงานเร็วขึ้นกว่าระบบปฏิบัติการ DOS ก็เลยทำให้โปรแกรมเมอร์ทั่วโลกได้เริ่มหันมามองเทคโนโลยีตัวนี้ใหม่อีกครั้ง และจากการพัฒนาอย่างต่อเนื่องของไมโครซอฟต์เพื่อเพิ่มประสิทธิภาพของ DirectX ให้มีความสามารถมากขึ้น ทำให้เกิด DirectX 2.0 ,DirectX 3.0 ,DirectX 5.0 ,DirectX 6.0,DirectX 7.0 ,DirectX 8.0 และ DirectX 9.0 ซึ่งเป็นเวอร์ชันใหม่ล่าสุดที่ไมโครซอฟต์ผลิตขึ้น และยังมีการพัฒนาเวอร์ชันใหม่ ๆ ต่อไปจากเทคโนโลยีนี้เองเป็นผลให้เกมบนระบบปฏิบัติการ DOS ค่อยๆ เลือนหายไปในที่สุด

ทุกวันนี้ DirectX 9.0 ได้นำเทคโนโลยี COM (Component object Model) มาใช้งานทำให้ได้รับความสะดวกสบายในการใช้งานเป็นอย่างยิ่ง ผู้พัฒนาเกมสามารถควบคุมและสร้างการทำงานได้อย่างอิสระ

2.3.3 ความรู้เบื้องต้นเกี่ยวกับ DirectX

DirectX เป็น API ของไมโครซอฟท์พัฒนาเพื่อใช้จัดเตรียมอินเตอร์เฟส สำหรับควบคุมฮาร์ดแวร์มัลติมีเดียบนระบบ Microsoft Windows ได้อย่างมีประสิทธิภาพ และเป็นเครื่องมือให้โปรแกรมเมอร์ทำงานกับคำสั่ง และโครงสร้างข้อมูลในระดับใกล้ชิดฮาร์ดแวร์ โดยไม่ต้องสร้างโค้ดติดต่อในระดับล่างซึ่งวิธี ติดต่อจะแตกต่างกันไปตามประเภทของอุปกรณ์ การเขียนโค้ดที่เป็นอิสระจากอุปกรณ์ในลักษณะนี้ ช่วยให้โปรแกรมเมอร์สามารสร้างซอฟต์แวร์เพื่อทำงานดังกล่าวได้อย่างดี แม้ผู้ใช้จะปรับเปลี่ยนอุปกรณ์ตัวใหม่ และเพิ่มการ์ดเร่งความเร็วแบบสามมิติ เสียง อุปกรณ์อินพุต และ อื่นๆ ก็ตาม

DirectX ได้รับการออกแบบให้นักพัฒนามีสภาพแวดล้อมคล้ายคลึงกับสภาพแวดล้อมที่มีประสิทธิภาพของ MS-DOS ซึ่งทำงานได้เร็วกว่าโค้ดที่ทำงานบนวินโดวส์ เนื่องจากไม่ต้องสูญเสียประสิทธิภาพจาก API สำหรับจัดการงานมัลติมีเดียของ วินโดวส์รุ่นก่อน แต่อย่างไรก็ตามการสนับสนุนความสามารถในการทำงานของฮาร์ดแวร์ที่มีอยู่บนระบบ โค้ดที่เขียนขึ้นด้วย DirectX สามารถรันได้เร็วกว่าแอพพลิเคชันของ MS-DOS

เมื่อใดก็ตามที่สร้างออปเจกต์ DirectX ให้กับดีไวซ์นั้น DirectX จะเข้าไปซักถามฮาร์ดแวร์ผ่าน HAI เพื่อดึงเอาข้อมูลเกี่ยวกับดีไวซ์ออกมานี้มีอยู่ในตาราง Cap Bits (Capability Bits) ข้อมูลที่มีอยู่ใน Cap Bits เป็นข้อมูลที่ใช้บอกความสามารถที่ฮาร์ดแวร์สามารถทำได้หรือความสามารถใดที่ HEL ต้องจำลอง

ไมโครซอฟท์จัดเตรียม Cap Bits ให้รับทราบฟีเจอร์ของฮาร์ดแวร์ที่มีอยู่ใน HAL และฟีเจอร์ที่ต้องจำลองขึ้นโดยซอฟต์แวร์ HEL ดังนั้นวิธีที่ดีที่สุดก็คือเขียนแอพพลิเคชันโดยใช้ค่าระบบ หรือฟีเจอร์ต่ำสุดที่ยอมรับได้ และ Optimize โค้ดที่เขียนให้รันได้เร็วและมีประสิทธิภาพมากที่สุด อีกทางหนึ่งควรเขียนโค้ดเผื่อการทำงานของ HEL ในอนาคตไว้ด้วย การสนับสนุนการใช้ฟีเจอร์เหล่านี้เป็นฟีเจอร์พิเศษที่จะมีให้เลือกบนเกมได้ เช่น การทำงาน Texture ขั้นสูง การทำโพลีกอนที่มีความซับซ้อนสูง หรือแม้แต่การสร้างแสงแบบไดนามิก เผื่อว่าใครก็ตามที่มีฮาร์ดแวร์ที่มีความสามารถสูงจะสามารถใช้งานฟีเจอร์ดังกล่าวได้ ดังนั้นควรออกแบบให้รองรับฟีเจอร์ต่างๆ ที่มีได้ทั้งหมด

DirectX จะใช้หลักการของ COM (Component Object Model) ซึ่งเป็นวิธีในการนำเอา component ที่มีประโยชน์ มาใช้ใหม่ใช้ในโปรแกรมต่าง ๆ โดยส่วนใหญ่แล้ว COM จะถูกสร้างเอาไว้ในไฟล์ .dll ซึ่งการที่เราจะสามารถใช้เมทธอดต่าง ๆ ที่มีใน COM ตัวนั้นได้ เราจะต้องทำการสร้าง COM object ขึ้นมา และเนื่องจาก COM จะใช้คุณสมบัติ Encapsulation อย่างเข้มงวด ทำให้เราไม่สามารถเรียกใช้เมทธอดต่าง ๆ จาก COM object ที่สร้างขึ้นมาได้ เราต้องสร้าง Interface จาก COM object อีกทีหนึ่ง เราถึงจะเรียกใช้เมทธอดที่ต้องการจาก Interface ได้ โดยใน Interface หนึ่งๆ ก็จะมีเมทธอดตามประเภทของ Interface ที่สร้างขึ้นเท่านั้น

DirectX จะประกอบขึ้นมาจากส่วนหลัก ๆ 7 ส่วน ตามนี้

- DirectDraw จัดการกับกราฟิก 2 มิติ และการติดต่อกับการ์ดแสดงผล
- Direct3D จัดการกับกราฟิก 3 มิติ และการติดต่อกับฮาร์ดแวร์ 3 มิติ
- DirectSound จัดการกับเสียง และการติดต่อกับการ์ดเสียง
- DirectPlay ควบคุมการติดต่อกับเครือข่ายสำหรับการเล่นเกมแบบ Multi-Player
- DirectInput ควบคุมการรับข้อมูลจากอุปกรณ์นำข้อมูลเข้า (Input Device)
- DirectMusic จัดการกับการเล่นเพลง

- DirectSetup สำหรับการติดตั้งส่วนประกอบต่าง ๆ ของ DirectX

2.3.4 หลักการเบื้องต้นเกี่ยวกับ DirectX

ก่อนที่จะมี DirectX เกิดขึ้นนั้น นักพัฒนาเกมคอมพิวเตอร์สำหรับ ดอสหรือวินโดวส์ จะต้องเขียนเกมคอมพิวเตอร์ให้รู้จักกับฮาร์ดแวร์ ซึ่งมีอยู่มากมายในท้องตลาด ซึ่งในกรณีที่มีฮาร์ดแวร์ตัวใหม่เกิดขึ้น อาจจะเกิดปัญหาความไม่สนับสนุนกันระหว่างเกมคอมพิวเตอร์กับฮาร์ดแวร์ตัวใหม่นั้น นักเล่นเกมจะต้องรอจนกว่านักพัฒนาเกมจะทำการอัพเดทเกมนั้นๆ ให้ใช้ความสามารถของฮาร์ดแวร์ใหม่ได้ เมื่อเป็นเช่นนี้ไมโครซอฟต์จึงได้ทำการพัฒนาเทคโนโลยีที่มีความสามารถในการเป็นสื่อกลางติดต่อระหว่างเกมคอมพิวเตอร์ หรือโปรแกรมมัลติมีเดียต่างๆ กับฮาร์ดแวร์ขึ้นมาโดยใช้ชื่อว่า "DirectX"

DirectX ตามความหมายจะหมายถึง ไลบรารีคำสั่ง (Run Time Library) ที่ช่วยทำงานด้านมัลติมีเดีย Graphic โดยตัว DirectX Foundation จะมีส่วนประกอบที่เรียกว่า HAL (Hardware Abstraction Layer) จะใช้ซอฟต์แวร์ในการรวจสอบความสามารถของฮาร์ดแวร์ที่อยู่ในเครื่องคอมพิวเตอร์นั้นอย่างอัตโนมัติ แล้วนำมากำหนดพารามิเตอร์ของแอพลิเคชันให้ตรงตามความเหมาะสมระหว่างเกมคอมพิวเตอร์ หรือโปรแกรมมัลติมีเดียกับไดร์เวอร์ของฮาร์ดแวร์ที่มีส่วนเกี่ยวข้องกับเกมหรือโปรแกรมนั้น ทำให้การประมวลผลโปรแกรมทำได้เร็วขึ้น เนื่องจากขั้นตอนต่างๆ จะถูกนำไปประมวลผลโดยตรง ไม่ต้องอาศัยตัวกลางอย่างเช่น GDI (Graphic Device Interface) ก่อน ทำให้นักพัฒนาเกมคอมพิวเตอร์สามารถเขียนโปรแกรมให้สื่อสารกับ DirectX เท่านั้นก็เพียงพอ นอกจากนี้ DirectX Foundation ยังมีส่วนประกอบที่เรียกว่า HEL (Hardware Emulation Layer) ทำให้สามารถใช้โปรแกรมมัลติมีเดีย หรือโปรแกรมที่เกี่ยวข้องกับ 3D บน Hardware ที่ไม่สนับสนุนการใช้งานทางด้าน 3 มิติ โดยจะทำการจำลองความสามารถบางอย่างที่ฮาร์ดแวร์ตัวนั้นไม่มี ให้สามารถใช้งานได้กับโปรแกรมที่ต้องการ แม้จะมีข้อเสียอยู่บ้างตรงที่ อาจทำให้ช้าลงบางก็ตาม แต่ก็คุ้มค่ากับความสามารถของ DirectX ที่มีอยู่ในปัจจุบัน

2.3.5 API ของ DirectX

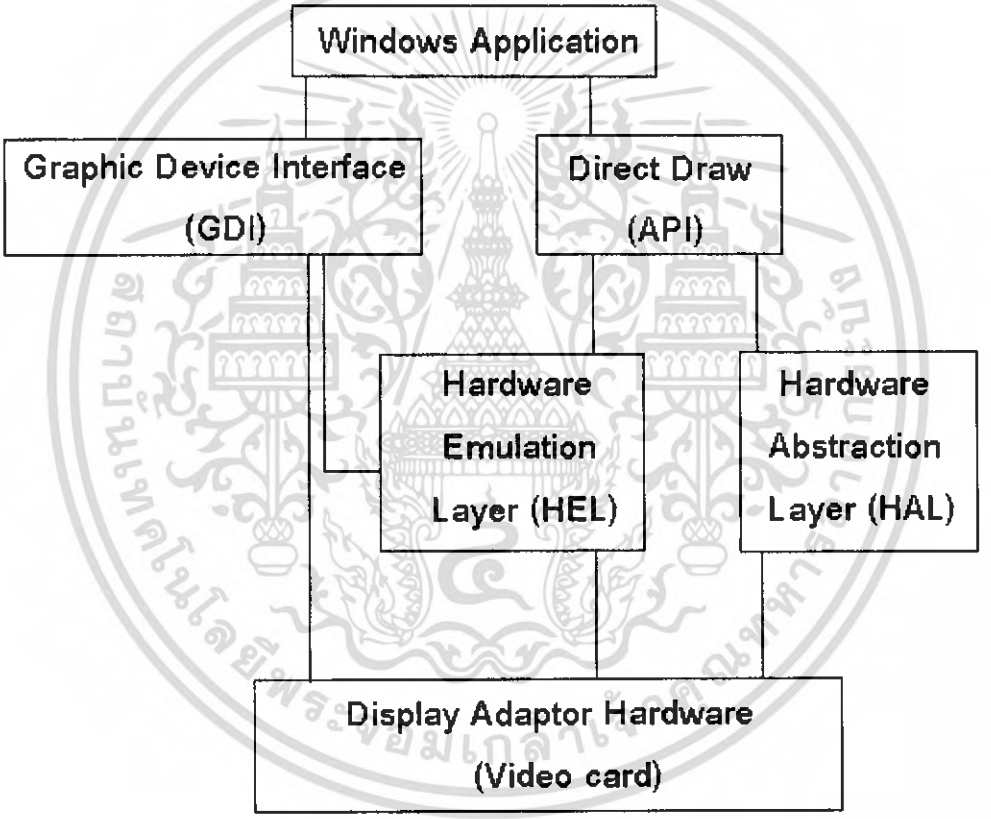
2.3.5.1 DirectDraw

DirectX ประกอบไปด้วย API มากมายที่ได้รับการออกแบบเพื่อใช้พัฒนาเกม และการจำลองแบบแบบสามมิติ (3D Simulation) (ส่วนใหญ่ยังไม่ได้พัฒนาให้เป็นแบบสามมิติ) โดย DirectX มี ไลบรารีที่เก็บฟังก์ชันที่ใช้ในการเรนเดอร์แบบสองมิติและสามมิติ , สร้างเสียงแบบปกติและแบบสามมิติ , ดนตรี , ติดต่อแป้นพิมพ์ , จอยสติ๊ก และอุปกรณ์อินพุตชนิดต่างๆ รวมทั้ง

ฮาร์ดแวร์ที่มีความสามารถสร้างปฏิกิริยาสะท้อนกลับ (จอยสติ๊กแบบสั่น) และการเล่นเกมผ่านเครือข่าย สามารถใช้ไลบรารีของคำสั่งต่าง ๆ ที่รวมเข้ามาเพื่อสร้างเกม และทำซิมูเลชันที่งดงาม

ชุด API ที่มีอยู่ใน DirectX คือ DirectDraw ,Direct3D ,DirectMusic ,DirectSound ,DirectPlay ,DirectInput และ DirectSetup

DirectDraw เป็นชุด API สำหรับใช้จัดการอุปกรณ์แสดงผล ควบคุมข้อมูลบิตแมปหน่วยความถี่ ออกนอกพื้นที่สกรีน และสร้างการติดต่อที่รวดเร็วให้กับฟีเจอร์ของฮาร์ดแวร์ เช่น Blitting และ Page Flipping ซึ่งเป็นฟีเจอร์พื้นฐานที่ Direct3D สามารถทำได้



รูปที่ 2.6 แสดงแผนภาพการทำงานของ Direct X

- **Cooperative level และ Display mode**

ก่อนที่จะคำสั่งต่าง ๆ ใน DirectX ให้แสดงผลภาพได้นั้น เราจะต้องเซต Cooperative level และ Display mode ก่อน โดยการเซต Cooperative level ก็คือการบอก DirectDraw ว่าเราจะทำงานกับ DirectDraw ในลักษณะใด เช่น window mode หรือ full screen mode สามารถ interrupt ด้วยการกดปุ่ม Ctrl-Alt-Del ได้หรือไม่ และสามารถที่จะเปลี่ยนขนาดของ window ได้หรือไม่ (ใน window mode) เป็นต้น หลังจากเซต Cooperative level แล้วจะต้องเซต Display mode เป็นลำดับ

ถัดมา เพื่อระบุ screen Resolution, color depths และ refresh rate ที่ต้องการโดยจะต้องระบุความกว้างความยาวของ resolution เป็นหน่วย pixel และจะใช้สีแบบใด 6 ,8 ,24 หรือ 32 บิต หรือจะใช้ตารางสี (palette) ทั้งนี้ video hardware ที่ใช้จะต้องสนับสนุนกับ mode ที่เลือกด้วย มิฉะนั้นจะเกิด error Display mode

- **Surface**

Surface เป็นพื้นที่บน memory ใช้สำหรับเก็บข้อมูลภาพ bitmap เพื่อเตรียมสำหรับการนำมาแสดงผลบนจอภาพต่อไป โดยพื้นที่นี้อาจอยู่บน video memory หรือ system memory ก็ได้ขึ้นอยู่กับค่า flags ที่ส่งให้ตอนสร้าง โดย surface แบ่งออกเป็น 2 ชนิด คือ Primary Surface และ Off-screen surface

Primary surface นั้นเป็น surface หลักที่จะแสดงผลออกสู่จอภาพ โดยจะต้องมีขนาดเท่ากับ Display mode ที่เลือกไว้ ภาพใด ๆ ก็ตามที่ต้องการแสดงออกทางจอภาพต้องนำมาวาดไว้บน Primary surface เสมอ แต่การนำภาพมาวาดลงบน Primary surface โดยตรงจะทำให้ภาพที่ได้กระพริบไม่ราบรื่น จึงต้องใช้การทำ flipping มาช่วย (จะกล่าวรายละเอียดในหัวข้อถัดไป)

Off-screen surface คือ surface ใช้สำหรับเก็บข้อมูลภาพต่างๆ เพื่อ Blitting ลงใน buffer หรือ surface อื่นเพื่อเตรียมแสดงผลต่อไป

- **Blitting และ Flipping**

Blitting คือ การนำภาพต่างๆ จาก source ไปวาดบนบริเวณ destination ที่กำหนด โดยอาจมีการย่อขยายภาพให้พอดีกับขนาด destination ที่กำหนด และสามารถทำการ transparent สีบางสีได้ด้วย การทำ Blitting สามารถทำได้หลายครั้ง โดยภาพที่ทำ Blitting จะเรียงซ้อนทับกันเป็นชั้นๆ

Flipping เป็นการสลับที่กันของ buffer โดยก่อนการทำ Flipping จะต้องมีการสร้าง buffer ขึ้นมาอย่างน้อย 2 buffer คือ Front buffer กับ Back buffer เพิ่มขึ้นมาก็ได้ โดย Front buffer คือ buffer ที่แสดงภาพออกที่หน้าจอ หรือ Primary surface นั่นเอง ส่วน Back buffer กับ Third buffer เป็นที่พักของภาพที่จะรอแสดงผลบนหน้าจอ buffer ทั้ง 3 ชนิดนี้จะมีขนาดเท่ากับ Display mode ที่เลือกไว้

การวาดหรือ Blitting ลงบน Front buffer โดยตรงจะให้ภาพกระพริบ ควรเตรียมภาพที่จะแสดงผลไว้ให้พร้อมบน Back buffer แล้วทำการ Flipping โดย DirectDraw จะสลับภาพระหว่าง Front buffer กับ Back buffer เพื่อนำภาพบน Back buffer มาแสดงบนจอภาพ โดยการสลับในที่นี้ไม่ได้หมายถึงการสลับที่ข้อมูลภาพแต่เป็นการสลับเฉพาะ pointer ที่ชี้พื้นที่ที่เก็บข้อมูลอยู่เท่านั้น ซึ่งทำได้อย่างรวดเร็ว และไม่ทำให้เกิดการกระพริบ หรือการกระตุกของภาพ

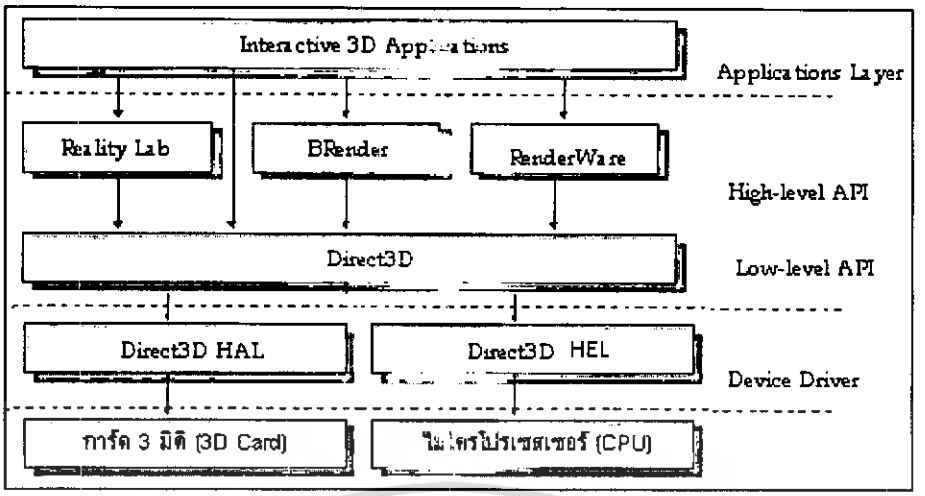
2.3.5.2 Direct3D

Direct3D คือ API ที่สามารถใช้เขียนโปรแกรมการฟิก 3 มิติ และติดต่องานฮาร์ดแวร์แต่งความเร็วสามมิติ การ์ดแสดงผลส่วนใหญ่ที่มีขายในท้องตลาดสนับสนุนความสามารถในการเร่งการแสดงผลสามมิติ และเกมสามมิติส่วนใหญ่ที่มีอยู่ในปัจจุบันก็มักจะรันบนวินโดว์ซึ่งใช้ Direct3D

Direct3D ในยุคแรกมี API อยู่สองโหมดคือ โหมด Immediate (IM) และ โหมด Relation (RM) ในโหมด IM (การเรนเดอร์ออปเจกต์ทำได้โดยฉับพลันได้ตามความต้องการของโปรแกรมเมอร์) เป็นโหมดที่ใช้งานยากแต่มีความยืดหยุ่นสูงเป็น API ในระดับล่างสำหรับใช้เขียนเกมที่ทำงานได้เร็ว และมีประสิทธิภาพเท่าที่จะเป็นไปได้บนระบบ ในโหมด RM (API จะเก็บซีนลงในฐานข้อมูล แล้วนำมาเรนเดอร์ทั้งหมดในคราวเดียวกัน) เป็นโหมดที่สร้างขึ้นมาเป็นเลเยอร์ที่อยู่บนสุดของโหมด IM โดยโหมดนี้จะจัดเตรียมบริการต่างๆ เช่น การจัดการ Texture, การโหลด Object File, การจัดลำดับเฟรม และการทำออปเจกต์เคลื่อนไหว การศึกษาและใช้งานโหมด RM นั้นง่ายกว่าเมื่อเทียบกับ โหมด IM แต่ในโหมด IM เป็นโหมดที่มีประสิทธิภาพและความยืดหยุ่นสูงกว่า ดังนั้นการพัฒนาการทำงานในโหมด RM จึงได้หยุดลงใน DirectX 6.0 และมุ่งพัฒนาการทำงานในโหมด IM ให้มีความสามารถและใช้งานง่ายในเวอร์ชันต่อมา ด้วยเหตุนี้เองการทำงานในโหมด RM จึงไม่สนับสนุนเทคโนโลยีใหม่เช่น Multitexturing ,Bump Mapping, Hardware Transformation และ Lighting ดังนั้นความสามารถทั้งหมดของโปรแกรม 3 มิติควรเขียนขึ้นมาด้วยการใช้โหมด IM

หลักการทำงานของ Driect3D คือ คอยเป็นตัวกลางประสานงานระหว่างซอฟต์แวร์ 3 มิติจำพวกเกมส์และ Application (ที่ออกแบบให้ใช้งานบน Windows 95) และการ์ดแสดงผลแบบ 3 มิติ ซึ่งเป็นฮาร์ดแวร์ โดยการที่โปรแกรมเมอร์ที่เขียนโปรแกรมนั้น สามารถเขียนโค้ดเพื่อเรียกใช้งานการแสดงผล 3 มิติ ที่ระดับใดๆ ก็ได้ และเนื่องจาก Direct3D นั้นเป็น API ที่ทำงานในระดับของ OS โดยไม่ได้ยุ่งเกี่ยวกับฮาร์ดแวร์ ดังนั้นการเขียนโปรแกรมจึงยืดหยุ่น สามารถกำหนดทางเลือกให้กับโปรแกรมเมื่อให้การประมวลผลภาพ 3 มิติบนคอมพิวเตอร์ได้อย่างมีประสิทธิภาพมากที่สุด

สำหรับการ์ด 3 มิตินั้น จะเห็นได้ว่าเทคนิคและกลไกในการแสดงภาพ 3 มิติอย่างที่กล่าวไปแล้วว่ามีหลากหลายมาก (Texturing, Fog, Alpha Bending, Z Buffering ฯลฯ) และผู้ผลิตการ์ดแต่ละราย ต่างก็มีมาตรฐานในการผลิตและการออกแบบชิปสำหรับการ์ด 3 มิติแตกต่างกันออกไป เพราะการ์ด 3 มิติเป็นเทคโนโลยีใหม่ในคอมพิวเตอร์ ในมุมของผู้ผลิตแล้วการออกแบบการ์ด 3 มิติให้ทำงานกับ Direct3D ไม่ใช่เรื่องง่ายนัก เป็นที่คาดการณ์ว่าประสิทธิภาพการทำงานของการ์ด 3 มิติแต่ละยี่ห้อนั้นจะแตกต่างกันออกไป



รูปที่ 2.7 ภาพแสดงสรุปการทำงานของ Direct3D

เมื่อ Direct3D นั้นมีอัลกอริทึมในการควบคุมฟังก์ชัน 3 มิติ ออกเป็น 2 ระดับใหญ่ คือ HEL (Hardware Emulation Layer) ซึ่งจะถูกเรียกใช้ในกรณีที่ตรวจพบว่าไม่มีการ์ดแสดงผล 3 มิติในคอมพิวเตอร์ เพื่อให้คอมพิวเตอร์สามารถทำงานกับกราฟิก 3 มิติได้ก็จะต้องจำลองฮาร์ดแวร์ 3 มิติ ซึ่งปกติตัวที่ถูกนำมาจำลองเป็นผู้ประมวลผลกราฟิกก็คือซีพียู ภาระหนักจะตกอยู่ที่ซีพียูซึ่งผลที่ได้จากการประมวลผลจะถูกส่งไปที่ Frame Buffer ของ DirectDraw แทน และระดับที่สองคือ HAL (Hardware Abstraction Layer) Direct3D จะควบคุมการแสดงภาพแบบ 3 มิติโดยผ่านฮาร์ดแวร์ที่มีความสามารถในระบบ 3 มิติ ซึ่งในที่นี้คือ การ์ด 3 มิติ Direct3D โดยหลักการแล้ว เมื่อมีการรันซอฟต์แวร์ที่ต้องการใช้งานฟังก์ชัน 3 มิติ Direct3D ก็จะมีอัลกอริทึมในการตรวจสอบหาช่องทางหรือเครื่องมือที่จะให้การแสดงผล 3 มิติ ในคอมพิวเตอร์นั้นๆ ดีที่สุด ซึ่งในที่สุดก็จะเป็นตัวบ่งบอกว่า Application นั้นต้องใช้ HEL หรือ HAL ในการควบคุมการแสดงผล 3 มิติ นั่นเอง

Direct3D ไม่ได้ทำงานเป็นตัวเชื่อมประสานระหว่าง 3D Application อย่างเดียวแต่ยังเปิดช่องให้ผู้ผลิต 3D Application API รายอื่นๆ ได้พัฒนาแอพพลิเคชันหรือฟังก์ชันในการเรียกใช้ Direct3D ด้วย ซึ่งการเปิดกว้างนี้เป็นช่องทางหนึ่งที่ทำให้ Direct3D กลายเป็นเครื่องมือสอนคอมพิวเตอร์ให้รู้จักกราฟิก 3 มิติ โดยแท้จริง

ถึงแม้ว่าประสิทธิภาพของ Direct3D ในส่วนของซอฟต์แวร์เพียงอย่างเดียวจะไม่เลวร้ายมากนัก แต่ DirectX นั้นถูกออกแบบมาให้มีความสามารถของฮาร์ดแวร์เร่งความเร็ว ซึ่งหมายถึงผู้ผลิตการ์ดแสดงผลสามารถที่จะสร้างการ์ดที่สนับสนุนฟังก์ชันของ DirectDraw และ Direct3D ได้จากตัวฮาร์ดแวร์โดยตรง พร้อมกับให้ไดร์เวอร์สำหรับการ์ดของคนที่สนับสนุน DirectX มาด้วยกัน เมื่อมีการเรียกใช้ฟังก์ชันที่การ์ดสนับสนุน การทำงานดังกล่าวก็จะถูกส่งไปให้การ์ดแสดงผลเป็นผู้จัดการแทน ซึ่งจะเพิ่มเวลาว่างให้กับซีพียูในการประมวลผลงานอื่นๆ ได้มากขึ้น ปกติแล้วฮาร์ดแวร์สำหรับ Direct3D โดยทั่วไปจะมีหน่วยความจำบนการ์ดอย่างน้อย 2 MB

2.4 OpenGL

- เป็นภาษาระดับล่างใช้ในการ rendering และ imaging library
- เป็นส่วนติดต่อระหว่าง graphic hardware กับ application program
- เป็น API ที่ใช้สร้าง color image ของ 3D objects
- เป็นภาษาทางด้านกราฟิกในเชิงของการทำงานมากกว่าเชิงอธิบาย
- ไม่ขึ้นกับ OS และ Hardware platform ที่ใช้

2.4.1 OpenGL Features

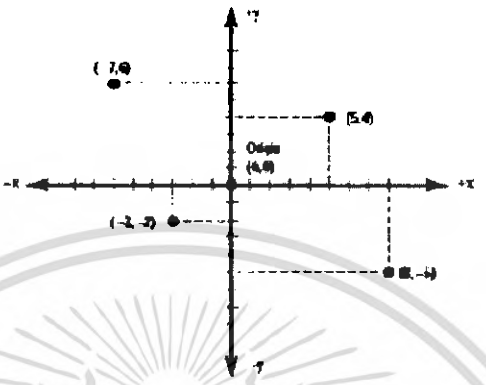
Texture Mapping	เป็นการนำภาพ Mapping เข้ากับ พื้นผิวของ Object
Z-buffering	ความสามารถในการคำนวณระยะทางจากตำแหน่งของ viewer
Double buffering	ทำให้ animation มีความ smooth
Lighting effects	สามารถคำนวณผลกระทบของแสงเมื่อแสงกระทบกับพื้นผิวของวัตถุ จากแหล่งกำเนิดแสง 1 จุด หรือ มากกว่า
Smooth shading	สามารถคำนวณผลกระทบของ shading ที่เกิดขึ้นเมื่อแสงกระทบกับพื้นผิว ของวัตถุ โดย มุมตกกระทบที่ต่างกัน จะให้ผลออกมาที่ต่างกัน
Material properties	ความสามารถในการระบุลักษณะของวัตถุว่า แข็ง หรือ ที่อ เช่น เหล็ก หรือ พลาสติก
Alpha blending	สามารถระบุ alpha หรือ opacity เพิ่มเติมจาก RGB ได้
Transformation matrices	สามารถเปลี่ยน ตำแหน่ง , ขนาด และ perspective ของ ออปเจกต์ใน 3D coordinate

2.4.2 ระบบพิกัดของเบื้องต้นของ OpenGL

- 2D Cartesian Coordinate
- Coordinate Clipping
- ViewPorts
- 3D Coordinate System
- Projections
- Orthographic Projections
- Perspective Projections

2D Cartesian Coordinate

ระบบพิกัดที่เราคุ้นเคยที่สุดและเป็นมาตรฐานที่สุด ก็คือระบบพิกัดคาร์ทีเซียน (Cartesian) ซึ่งระบบโดยคู่พิกัด (x,y) ตามแนวนอนและแนวตั้งตามลำดับ โดยมีจุดกำเนิด (origin) ที่ (0,0)



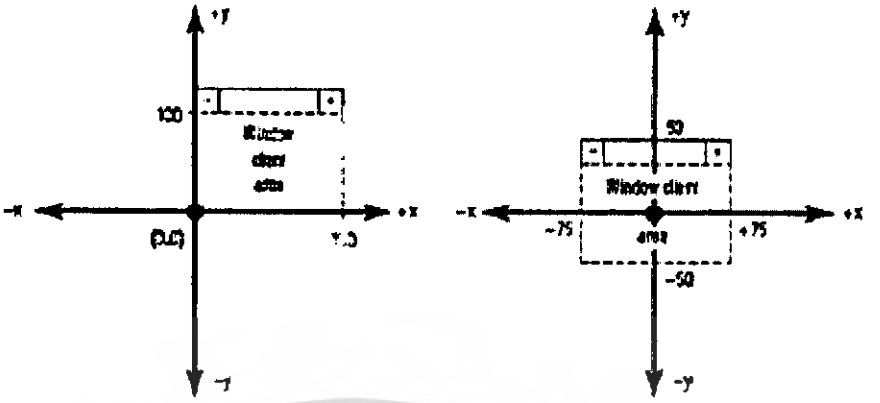
รูปที่ 2.8 ระบบพิกัดคาร์ทีเซียนสองมิติ

โดยแกนทั้งสอง (x และ y) นั้นตั้งฉากซึ่งกันและกันและรวมเรียกว่า xy plane หรือระนาบ xy โดยระนาบคือ พื้นที่ราบ นั่นเอง ซึ่งโดยทั่วไปแล้ว เมื่อเส้นตรงสองเส้นตัดกันทำมุม ๆ หนึ่ง จะเกิดระนาบขึ้นมาโดยปริยาย ซึ่งในระบบที่มีแกนเพียงแกนนั้น จะมีระนาบเพียงหนึ่งเดียวเท่านั้น

Coordinate Clipping

ก่อนที่เราจะสามารถพล็อตจุด (หรือวาดรูปใดๆ) ลงไปในหน้าต่างได้ เราต้องทำการบอก OpenGL ก่อนว่าจะทำการ "แปลง" คู่พิกัดของสิ่งที่เราต้องการวาด ไปเป็นคู่พิกัดของจอ (physical pixel coordinate) ได้อย่างไร ซึ่งสามารถทำได้โดยการระบุขอบเขตในระนาบคาร์ทีเซียนที่ใช้หน้าต่าง ซึ่งเรียกว่า clipping region ซึ่งในระนาบสองมิตินั้น clipping region ก็คือว่า x,y ต่ำสุดและสูงสุดที่อยู่ภายในหน้าต่างนั่นเอง อีกวิธีหนึ่งที่ทำได้ก็คือ การระบุตำแหน่งของจุดกำเนิดในเชิงสัมพัทธ์กับหน้าต่าง

รูปต่อไปนี้แสดงถึง clipping region แบบที่นิยมใช้กันสองแบบ



รูปที่ 2.9 clipping region ในระนาบคาร์ทีเซียนสองมิติ

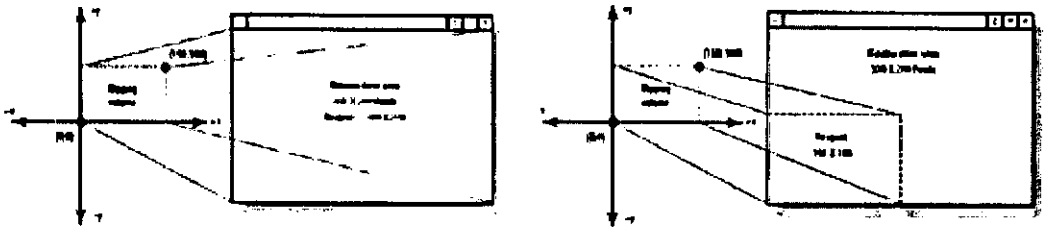
รูปซ้ายนั้น มีพิกัด x ตั้งแต่ 0 ถึง 150 จากซ้ายไปขวา และ y ตั้งแต่ 0 ถึง 100 จากล่างขึ้นบน จุดกลางจอมีตำแหน่ง (75,50) ส่วนในรูปซ้ายนั้น x มีค่าตั้งแต่ -75 ไปถึง 75 และ y มีค่าตั้งแต่ -50 ไปถึง 50 และจุดกลางมีค่า (0,0)

เราสามารถที่จะใช้ OpenGL หรือว่าไลบรารีตัวอื่นๆในการปรับค่านี้กลับหัวกลับหางหรือซ้ายเป็นขวาได้ ซึ่งในความเป็นจริงแล้ว ค่าปริยายของพิกัดหน้าต่างนั้น มักจะให้ค่า y วิ่งจากบนลงล่าง มากกว่าล่างขึ้นบน ทั้งนี้เนื่องจาก เป็นลักษณะที่เป็นธรรมชาติมากกว่าสำหรับการแสดงข้อความ (text) แต่ว่าไม่เหมาะสมเท่าใดนักกับการใช้งานหรือวาดกราฟิก

Viewports (การ map ค่าจาก drawing coordinate ไปยัง window coordinate)

โดยปกติแล้ว clipping area ที่เราต้องการกำหนดนั้น มักไม่ค่อยตรงกับขนาดจริงของหน้าต่าง (เช่นเรา อาจกำหนดหน้าต่างขนาด 640x480 แต่ว่าต้องการ clipping area จาก x : [-75,75] และ y : [-50,50] ก็ได้) ดังนั้นค่าของจุดใน clipping area ที่เราต้องการวาดนั้นต้องได้รับการ map ไปยังค่าพิกัดจริงของหน้าต่าง และไปยังค่าจริงของอุปกรณ์แสดงผลต่อไป ซึ่งเราเรียกกว่า map เช่นนี้ว่า Viewport

Viewport คือพื้นที่ในส่วนของหน้าต่างโปรแกรมที่ใช้ในการวาด clipping area ซึ่งอันที่จริงแล้วก็เป็นเพียง mapping ของ clipping area ไปยัง window coordinate อย่างง่าย ๆ เท่านั้น ซึ่งโดยปกติแล้ว Viewport จะมีค่าเป็นพื้นที่ทั้งหมดของหน้าต่าง แต่ว่านี่ไม่ใช่สิ่งที่จำเป็นแต่อย่างใด เราสามารถระบุ Viewport เป็นอะไรก็ได้ที่เราต้องการ เช่น ล่างซ้ายของ clipping area เป็นต้น



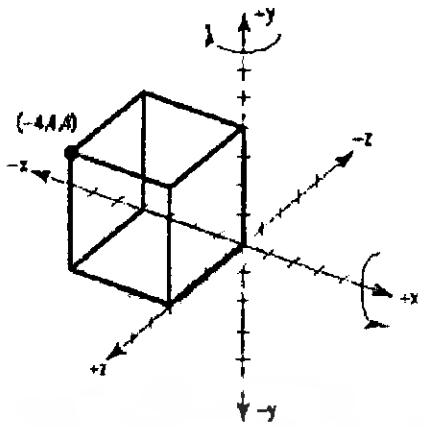
รูปที่ 2.10 ตัวอย่างวิวพอร์ต (viewport)

จากรูป ซ้ายมือ เรากำหนดขนาดของหน้าต่างไว้เป็น 300x200 และกำหนดให้ Viewport มีขนาดเป็น 300x200 ด้วยเช่นกัน ซึ่งถ้าหากเรากำหนด clipping area ไว้เป็น x : [0,150] และ y : [0,100] ระบบก็จะทำการ map ค่าพิกัดไปยังค่าพิกัดของหน้าต่างจริงให้เอง (ซึ่งนั่นหมายความว่า การเปลี่ยนแปลงพิกัดใน clipping area ก็จะถูก map โดยการเพิ่มเป็นสองเท่า และถูกวาดลงหน้าต่าง) ส่วนรูปขวามือนั้น เรากำหนดขนาดหน้าต่างไว้เป็น 300x200 และ clipping area เป็น x : [0,150] และ y : [0,100] เท่ากันกับภาพซ้าย แต่ว่าเรากำหนด view เป็นขนาด 150x100 ซึ่งผลที่ได้นั้นทำให้ภาพที่เราเห็นเต็ม clipping area นั้น จะปรากฏบนมุมล่างซ้ายของหน้าต่างเท่านั้น (เพราะว่าการเปลี่ยนแปลงขนาด 1 pixel ใน clipping area ก็จะมีค่าเท่ากับ 1 pixel ในหน้าต่างด้วย)

เราสามารถใช้ viewport นี้ในการเพิ่ม / ลดขนาดภาพในหน้าต่างได้และสามารถใช้ในการแสดงเฉพาะส่วนของ clipping area ได้เช่นกัน

3D Coordinate System

สำหรับระบบพิกัดสามมิตินั้นก็เหมือนกับสองมิติ เพียงแค่เพิ่มแกน z ที่ตั้งฉากกับระนาบ xy เข้ามาอีกแกนหนึ่ง ซึ่งในระบบการแสดงผลของภาพนั้น ก็คือระยะใกล้ไกล (+z ใช้ออกจากจอภาพมาหาผู้ใช้) ดังรูป

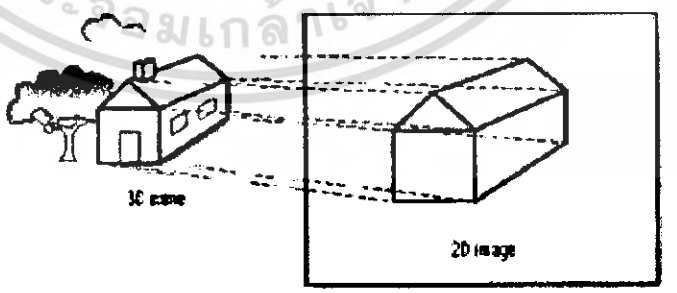


รูปที่ 2.11 ระบบพิกัดสามมิติ (แกน z ชี้ออกจากจอภาพเข้าหาตัวผู้มอง)

Projection: จาก 3D สู่ 2D

แม้ว่าเราจะสามารถกำหนดและวาดสิ่งที่ต้องการได้โดยใช้พิกัดสามมิติ แต่ว่าจอภาพก็ยังคงเป็นสองมิติอยู่นั่นเอง ดังนั้นการที่จะแสดงผลออกให้เราเห็นทางหน้าต่างของโปรแกรมได้นั้น ต้องทำการแปลงพิกัดมาเป็นสองมิติเสียก่อนถึงจะระบุตำแหน่ง pixel ได้ ซึ่งสามารถทำได้โดยใช้วิธีการทางคณิตศาสตร์ (ตรีโกณมิติและวิธีการทางเมตริกซ์) ซึ่งแม้ว่าเราจะไม่ต้องเข้าใจคณิตศาสตร์ตรงนี้มากเท่าไหร่นัก ความเข้าใจที่ลึกขึ้นของ 3D maths จะช่วยให้เราใช้งาน OpenGL ได้อย่างมีประสิทธิภาพมากขึ้น กับงานหลายประเภทมากขึ้นด้วย

สำหรับคอนเซปท์แรกที่เราต้องทำความเข้าใจ ก็คือสิ่งที่เรียกว่า projection ซึ่งเป็นการนำเอาพิกัดสามมิติที่เราได้กระทำการระบุตอนที่วาดไปแปลง (project) ลงบนระนาบสองมิติ



รูปที่ 2.12 Projection: 3D image --> 2D surface

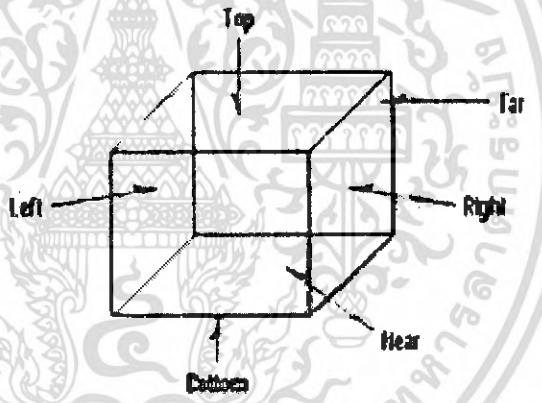
ซึ่งใน OpenGL นั้น จะมี projection อยู่สองแบบที่เราต้องทำความคุ้นเคย นั่นก็คือ orthographic และ perspective ซึ่งเราทำการระบุ projection หมายความว่า

เราได้ทำการระบุ viewing volume และกำหนดว่า viewing volume นี้จะถูก transform ยังไงลงในระนาบสองมิติ

- **Orthographic Projections**

ใช้เมื่อทำการระบุ square (หรือ rectangular) viewing volume ซึ่งอะไรก็ตามนอกเหนือจาก volume นี้จะไม่ถูกวาด นอกเหนือจากนั้น วัตถุทุกอย่างที่มีขนาดเท่ากันจะถูกวาดด้วยขนาดเท่ากัน ไม่ว่ามันจะอยู่ใกล้หรือไกลต่างกันยังไงก็ตาม ซึ่ง projection ในลักษณะนี้ มักจะถูกใช้ในงานจำพวกงานออกแบบเขียนแบบหรือกราฟสองมิติ ตัวอย่างในการใช้งานกับโปรแกรมสามมิติก็เช่นการเพิ่มตัวอักษร (เช่น frame rate) ลงไปบนหน้าจอ เป็นต้น

เราทำการระบุ orthographic projection โดยการระบุระนาบสำหรับ clipping ดังนี้ :far, near, left, right, top, และ bottom ดังรูป

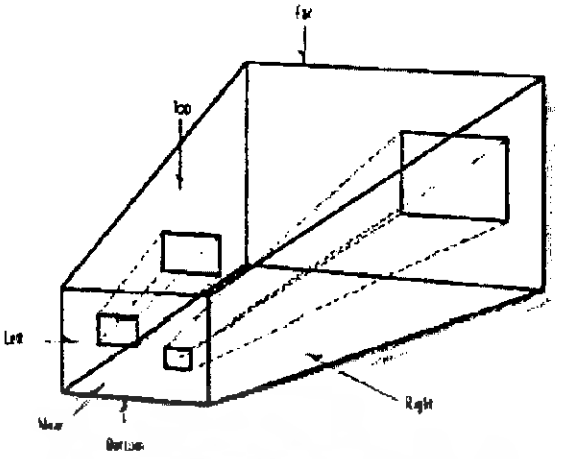


รูปที่ 2.13 Orthographic projection clipping volume

ซึ่งวัตถุทุกอย่างที่ระบุตำแหน่งให้อยู่ภายใน clipping volume นี้จะถูก project ไปเป็นพิกัดสองมิติ (ตาม orientation) สำหรับแสดงผลต่อไป

- **Perspective Projections**

Projection แบบที่สอง และเป็นแบบที่คุ้นเคยกันมากกว่า ก็คือแบบ Perspective ซึ่งจะเพิ่มระยะใกล้ไกลของวัตถุในรูปเข้าไปด้วย ซึ่งเหมือนกับการคิดว่า clipping volume มีลักษณะเป็นพีระมิดหัวตัด (เรียกว่า frustum) ซึ่งทำให้วัตถุที่อยู่ใกล้กว่า มีขนาดใกล้เคียงกับขนาดที่ถูกกำหนดไว้มากกว่าวัตถุที่อยู่ไกล ดังรูป



รูปที่ 2.14 Perspective projection clipping volume

2.5 Irrlicht Engine

Irrlicht Engine เป็น 3D เอนจินที่ถูกพัฒนาขึ้นโดยใช้ภาษา C++ ซึ่งมีความสามารถในการทำงานได้ทุกแพลตฟอร์ม (platform) ซึ่งมี API ระดับสูงที่ใช้สำหรับสร้างเกมได้ทั้ง 3D และ 2D หรือสามารถนำไปใช้ในงานด้านกราฟิกอื่น ๆ ได้ ซึ่งเอนจินได้รวบรวมเอาความสามารถต่าง ๆ ไว้ด้วยกัน อันได้แก่ การทำงานแบบ

- Dynamic Shadow
- Particle Systems
- Character Animation
- Indoor & Outdoor Technology
- Collision Detection

ซึ่งเข้าควบคุมและออกแบบได้ง่ายผ่านทางอินเตอร์เฟส ซึ่งรายละเอียดความสามารถต่าง ๆ มีดังนี้

- มีประสิทธิภาพในการเรนเดอร์ภาพทั้งใน Direct3D และ OpenGL
- ทำงานได้โดยไม่ขึ้นกับแพลตฟอร์ม เช่น Window95 98 NT 2000 XP หรือ Linux
- มี Library ให้ใช้เป็นจำนวนมาก ทั้งยังสามารถเพิ่ม Library อื่น ๆ ได้อย่างยืดหยุ่นทั้งยังสนับสนุนการทำงานของ vertex & pixel shader อีกด้วย
- สามารถทำงานผสมผสานระหว่าง feather ตัวเองและ feather นอก ได้อย่างมีประสิทธิภาพ เช่น สามารถ import utilities ต่าง ๆ มาใช้ได้
- มีระบบ Character animation คือ สามารถนำแอนิเมชันต่าง ๆ มาประกอบเข้ากันได้
- มี effect หลากหลายให้ใช้

- สนับสนุนการพัฒนาโดยใช้ VC++ และ .NET ทุกภาษาเช่น C# , VisualBasic และ Delphi.NET
- มีความเป็นอิสระทั้งในด้านแพลตฟอร์มและไดรเวอร์ทั้งยังมี software ที่ช่วยสำหรับการเรนเดอร์อย่างมีประสิทธิภาพ (software renderer)ซึ่งมีความสามารถในการทำ z-buffer gouraud shading, alpha-blending และ transparency ทั้งยังวาดภาพ 2D ได้อย่างรวดเร็ว
- มีระบบ GUI ให้ผู้ใช้สามารถใช้งานได้ง่าย
- มี Function ช่วยในการวาดภาพแบบ 2 มิติ เช่น alpha blending และการผสมผสานระหว่างภาพ 3D และ 2D
- มี Tuturial ที่สอนในเรื่องการใช้ API ที่เข้าใจได้ง่าย
- สนับสนุนการเขียนโดยใช้ C++ และภาษาเชิง object ได้เกือบทุกภาษา
- สามารถ import ไฟล์ข้อมูลโดยตรงจาก Maya (.obj) , 3DStudeio(.3ds) COLLADA(.dae) , DelcD(.dmf), Milkshape(.ms3d) Quake 3 levels (.bsp) , Quake 2 models(.md2) , Microsoft DirectX (.X) ได้
- สามารถ import texture ได้ทั้งในสกุล
  - Windows Bitmap (.bmp)
  - Portable Network Graphics (.png)
  - Adobe Photoshop (.psd)
  - JPEG File Interchange Format (.jpg)
  - Truevision Targe (.tga)
  - ZSoft Painbrush (.pcx)
- มีระบบ collision detection ที่ตอบสนองได้รวดเร็วและใช้ง่าย
- มี Library สนับสนุน function ทางคณิตศาสตร์ ที่ทำงานกับภาพ 3D เพื่อให้ทำงานได้รวดเร็วยิ่งขึ้น
- สามารถอ่านข้อมูลจากไฟล์ ที่ถูกบีบอัดไว้ได้โดยตรงเช่น .zip
- มี XML parser ที่รวดเร็ว
- สนับสนุน ข้อมูลแบบ Unicode
- ทำงานได้กับทั้ง Microsofts VisualStudio6.0, VisualStudio.NET 7.0-8.0, Metrowerks Codewarrior, and Bloodshed Dev-C++ with g++3.2-4.0.
- ตัวเอนจินเองจะ open source และฟรี ซึ่งสามารถเข้าไป debug หรือแก้ไขได้ตามใจ

ชอบ ซึ่ง licence จะเป็นของ zlib-licence ไม่ใช่ของ GPL หรือ LGPL ซึ่งทั้ง GPL และ LGPL เป็น licence ที่ยอมให้คนอื่นเข้าไปปรับปรุงแก้ไขได้ แต่หากมีการแก้ไขจำเป็นต้องแจ้งให้เจ้าของทราบ

2.5.1 Special effects

ใน Irrlicht engine จะมี special effect มากมายให้เลือกใช้ ซึ่งใช้ได้ง่าย ส่วนใหญ่แค่ switch ไปมาว่าจะเปิดใช้หรือไม่แค่นั้น ซึ่ง effect ที่รองรับมีดังต่อไปนี้

- Realistic water surfaces
- Dynamic lights
- Dynamic shadows using the stencil buffer
- Billboards
- Bump mapping
- Parallax mapping
- Transparent objects
- Light maps
- Customizeable Particle systems for snow, smoke, fire, ...
- Sphere mapping
- Texture animation
- Skyboxes
- Fog

2.5.2 Driver

Irrlicht Engine สนับสนุน 5 API ซึ่งในนั้นมี 4 ตัวที่เอนจินนี้รองรับมากกว่าเอนจินอื่นๆ

- Direct3D 8.1
- Direct3D 9.0
- OpenGL 1.2
- The Irrlicht Engine software renderer
- A null device

ลักษณะเด่นอื่นจุดที่เอนจินนี้สนับสนุนคือ ผู้พัฒนาโปรแกรม ไม่จำเป็นต้องรู้ว่า API ที่ เอนจินใช้ติดต่อเป็นเอนจินตัวใด มันสามารถจัดการได้เอง ซึ่งผู้พัฒนาเพียงแต่เป็นผู้ระบุว่าจะใช้ API ตัวไหนแค่นั้นพอ และนี่คือเหตุผล 3 ประการที่ทำไมเอนจินนี้ ไม่สนใจ API ตัวใดตัวหนึ่งเท่านั้น

1) Performance

เพื่อประสิทธิภาพ เนื่องจาก API แต่ละตัวจะเก่งคนละด้าน เช่น OpenGL จะแสดงผลภาพได้ดีกว่า แต่ถ้าใช้ Direct3D จะทำให้แสดงผลได้รวดเร็วกว่า เป็นต้น

2) Platform independence

เนื่องจาก Direct3D จะไม่สามารถทำงานได้บน Mac และ Linux ในขณะที่ OpenGL สามารถทำงานได้ ซึ่งในกรณีของ OpenGL ก็เช่นเดียวกัน ซึ่งเพราะเหตุนี้เอง Irrlicht Engine จึงได้พยายามรองรับ API ที่หลากหลายเพื่อให้ทำงานได้บนทุกแพลตฟอร์ม

3) Driver problems

เป็นปัญหาบ่อย ๆ ที่เกิดขึ้นที่ user จะต้องเจอเมื่อใช้ 3D software ซึ่งปัญหาส่วนใหญ่เกิดจากการไม่ยืดหยุ่นของ driver และไม่สนับสนุน driver version เก่า ๆ ซึ่งในเอนจินนี้จะให้ผู้ใช้ สามารถเลือก driver ที่จะใช้ได้เอง เมื่อเกิดปัญหาขึ้น

2.5.3 Platforms

Irrlicht Engine ไม่ขึ้นกับแพลตฟอร์มใดแพลตฟอร์มหนึ่งมันจึงสามารถทำงานได้บนแพลตฟอร์มมากกว่า 1 แพลตฟอร์มทั้ง

- Windows 98, ME, NT 4, 2000, XP, XP64
- Linux

แพลตฟอร์มอื่นๆ เช่น MacOS ได้ถูกออกแบบไว้แล้ว แต่ยังไม่ได้นำไป implement โปรแกรมเมอร์จะเขียนโค้ดสำหรับเกมของเค้าเพียงแค่ครั้งเดียวเท่านั้น มันก็สามารถรันได้บน ทุก ๆ แพลตฟอร์มโดยไม่จำเป็นต้องเปลี่ยนแปลงโค้ดแม้แต่บรรทัดเดียว

2.5.4 Materials and Shaders

มันมีความสามารถในการสร้าง environment ที่มีความสมจริงได้อย่างรวดเร็ว ซึ่งในเอนจินนี้จะมี material จำนวนมากใช้เลือกใช้

ซึ่ง material บางตัวจะมาจาก fixed function pipeline (ตัวอย่างเช่นการทำ light mapped) และบางอย่างต้องอาศัย programmable pipeline (normal mapped/parallax per pixel lighted

materials for example) ซึ่งในปัจจุบัน 3D Hardware ได้มีให้แล้ว มันยังสามารถที่จะผสม materials ทั้ง 2 ประเภท ใน scene ได้โดยไม่มีปัญหาใดๆ และเมื่อมีการใช้ material ที่ต้องการ feature ที่ Hardware ไม่สามารถทำได้

เอนจินจะจัดหา fall back materials ไว้ให้ แต่อย่างไรก็ตาม ถ้า built in materials มีไม่เพียงพอ มันเป็นไปได้ที่จะทำการเพิ่ม material ใหม่ ๆ เข้าไปใน irricht engine ในตอน runtime โดยไม่ต้องทำการ modify หรือ recompile ตัวเอนจินใดๆ ทั้งสิ้น shader languages ที่รองรับในปัจจุบัน ได้แก่

- Pixel and Vertex Shaders 1.1 to 3.0
- ARB Fragment and Vertex Programs
- HLSL
- GLSL

2.5.5 Scene Management

การเรนเดอร์ใน Irrlicht Engine จะทำโดยการใช้ hierarchical scene graph ซึ่ง Scene nodes จะถูกยึดติดกับ nodes อื่น ๆ ตามการเคลื่อนที่ของมัน และคัดเลือก children node ของพวกมันไปยัง the viewing frustum และสามารถทำการ collision detection บาง scene node สามารถถูกใช้เป็นตัวอย่างของระดับชั้นของกล้อง, ภายในอาคาร, ภายนอกอาคาร ลักษณะโครงกระดูก การเคลื่อนไหวของน้ำ เป็นต้น

ในกรณีนี้ Irricht engine สามารถทำการผสม indoor scene กับ outdoor scene ได้อย่างกลมกลืน และให้ programmer สามารถควบคุมสิ่งต่างๆ ใน scene ได้อย่างเต็มที่ อีกทั้ง programmer ยังสามารถทำการเพิ่มเติมสิ่งต่างๆ ได้เอง เพราะว่าโปรแกรมเมอร์สามารถ เพิ่ม node ของเค้าได้เอง รวมถึง mesh and texture loaders, GUI elements อีกด้วย

2.5.6 Character Animation

ในปัจจุบันมี Character Animation ที่ถูกใช้อยู่ 2 ประเภท คือ

Morphing target animation : Meshes เป็นการ interpolated แบบ linear จาก frame หนึ่ง ไปยัง frame ถัดไป ซึ่งวิธีนี้ถูกใช้ในเกมตระกูล Quake ส่วนใน irricht engine จะใช้มันโดยการ import file สกุล .md2

Skeletal animation : พื้นผิวจะถูกจัดการโดยการเชื่อมต่อกัน ซึ่ง Irricht Engine จะทำโดยการโหลด file สกุล .ms3d หรือ .x ซึ่งมันง่ายมากที่เราจะยึดติด objects ใดๆ เข้ากับโมเดลที่เรา

สร้างขึ้นมา อย่างเช่น การติดอาวุธเข้ากับมือของไมเคล ซึ่งมันจะเคลื่อนที่ตามการเคลื่อนที่ของมือด้วย ซึ่งการยึดติดออปเจกต์นี้สามารถทำได้โดยการเขียนโค้ดเพียงแค่บรรทัดเดียวเท่านั้น

โปรแกรมเมอร์ไม่จำเป็นต้องรู้เกี่ยวกับการทำงานเหล่านี้ ถ้าเค้าไม่ต้องการ เค้าเพียงแค่ทำการโหลดไฟล์เข้าไปในเอนจินและใช้จินตนาการเพื่อวาดมันก็เพียงพอแล้ว

2.5.7 Supported Formats

มี file formats มากมายที่ได้รับการรองรับและ สามารถถูก load เข้าสู่เอนจินได้โดยตรง โดยไม่ต้องถูก Convert เพื่อเปลี่ยนเป็น format ใดๆ สำหรับการใช้งานกับ irricht engine ซึ่งเป็นการช่วยประหยัดเวลาที่ใช้ในการพัฒนาเกมได้ทางหนึ่ง จำนวน file format ที่ irricht engine รองรับได้เพิ่มมากขึ้นเรื่อยๆ หากคุณต้องการให้ irricht engine สามารถโหลด file format ใดๆ ที่ยังไม่มีการรองรับในปัจจุบัน

Textures file formats ที่ Irrlicht Engine รองรับในปัจจุบัน คือ

- Adobe Photoshop (.psd)
- JPEG File Interchange Format (.jpg)
- Portable Network Graphics (.png)
- Truevision Targa (.tga)
- Windows Bitmap (.bmp)
- Zsoft Paintbrush (.pcx)

Mesh file formats ที่ Irrlicht Engine รองรับในปัจจุบัน คือ

- 3D Studio meshes (.3ds)
- Alias Wavefront Maya (.obj)
- Cartography shop 4 (.csm)
- COLLADA (.xml, .dae)
- DeleD (.dmf)
- FSRad oct (.oct)
- Microsoft DirectX (.x)
- Milkshape (.ms3d)
- My3DTools 3 (.my3D)
- Pulsar LMTools (.lmts)
- Quake 3 levels (.bsp)

- Quake 2 models (.md2)

นอกจากนั้น ยังมีตัว exporter สำหรับ 3D Packages ที่เป็นที่นิยม (เช่น Blender, 3DSMax, Gile[s], ..) ซึ่งถูกรวมไว้อยู่ใน SDK แล้ว

2.6 Audiere Engine

ออเดียร์ (Audiere) คือ แอพพลิเคชันโปรแกรมอินเตอร์เฟส (Application Program Interface) ทางด้านเสียงระดับสูง มันสามารถเล่นไฟล์ได้หลายรูปแบบ ไม่ว่าจะเป็นไฟล์

- Ogg Vorbis
- MP3
- FLAC
- uncompressed WAV
- AIFF
- MOD
- S3M
- XM
- IT

สำหรับ ออดิโอ้ เอาต์พุท (audio output) ของออเดียร์สนับสนุนไดเรคซาวด์ (DirectSound) หรือ วินเอ็มเอ็ม (WinMM) บน ระบบปฏิบัติการ วินโดว์ (Windows), โอเอสเอส (OSS) บนลีนุกซ์ (Linux) และ ซิกวิน(Cygwin) และ เอสจีแอล เอแอล (SGI AL) บน อิลิก (IRIX)

ออเดียร์เป็นโอเพน ซอส (Open Source) ที่มีลิขสิทธิ์แบบแอลจีพีแอล (LGPL) ซึ่งหมายถึง สามารถใช้ในทางธุรกิจได้ แต่ไม่อนุญาตให้เปลี่ยนแปลงซอสโค้ด (Source Code) ถ้าต้องการเปลี่ยนแปลงจะต้องชำระเงิน

ออเดียร์สามารถทำงานได้บนหลายระบบ ไม่ว่าจะเป็นวินโดว์, ลีนุกซ์ ไอสามแปดหก (Linux-i386), ซิกวิน และ อิลิก ด้วย คอมไพล์เลอร์ (compiler) หลักๆ อย่างน้อย 3 ตัว

ออเดียร์ ไม่ขึ้นกับระบบปฏิบัติการ ดังนั้นการทำงานในสถาปัตยกรรมที่ต่าง ๆ กัน ทำได้โดยการเปลี่ยนแปลงโค้ด (code) เพียงเท่านั้น

คุณสมบัติ

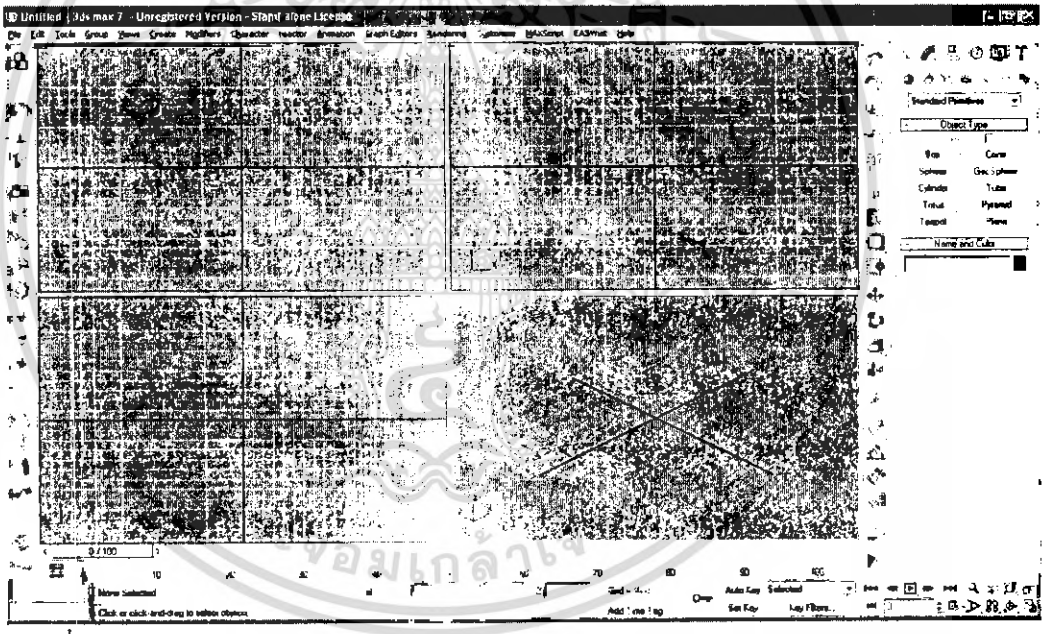
- เป็น แอพพลิเคชันโปรแกรมอินเตอร์เฟส ที่ใช้งานได้ง่าย

- สนับสนุนรูปแบบไฟล์ แบบ Uncompressed WAV*, Uncompressed AIFF*, Ogg Vorbis*, FLAC*, MP3, MOD, S3M, IT, XM (* สนับสนุนการ seeking)
- สนุบสนุนการสตรีมมิง (Streaming) และ บัฟเฟอร์ (buffered) เกี่ยวกับเสียง
- สนุบสนุนการปรับแต่ง เปลี่ยนแปลง และ ขยายความกังวานของเสียง
- สนับสนุนสตรีมของไฟล์ ที่กำหนดขึ้นเอง
- สนับสนุน Python, Delphi, Java, XPCOM (JavaScript in Mozilla) ไบน์ดิ้ง (bindings)

2.7 การใช้งานเกี่ยวกับ 3D Studio Max7

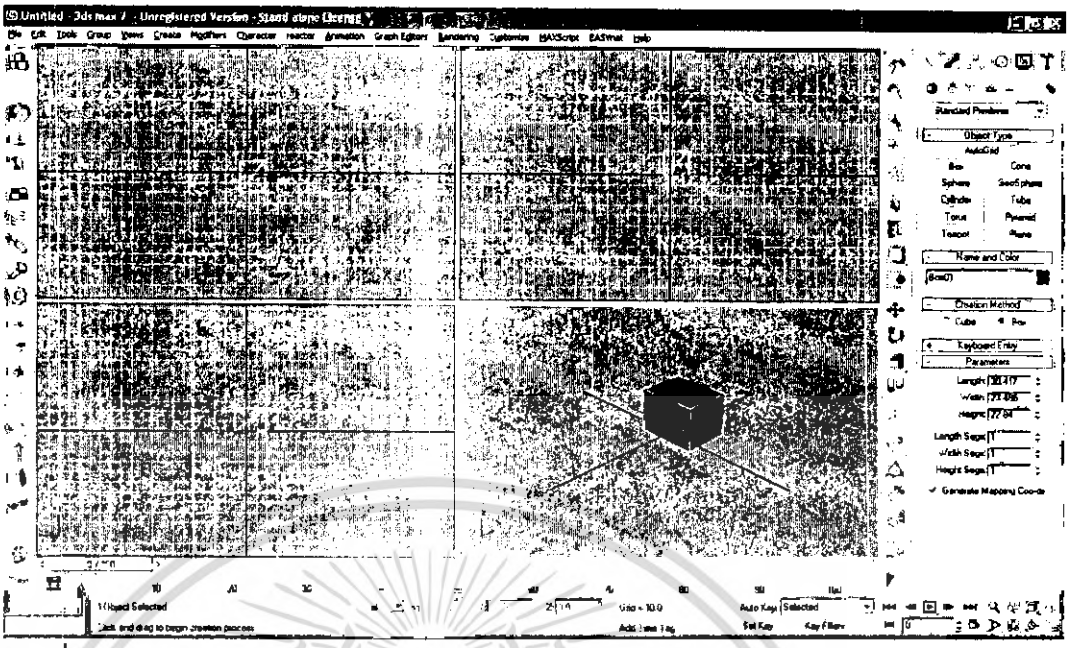
2.7.1 การกำหนดการเลื่อนไหว

1. สร้างโมเดลที่ใช้ในเกม โดยเปิดโปรแกรม 3d studio max 7 ขึ้นมาแล้วที่ FILE -> New จะได้หน้าต่างดังรูป



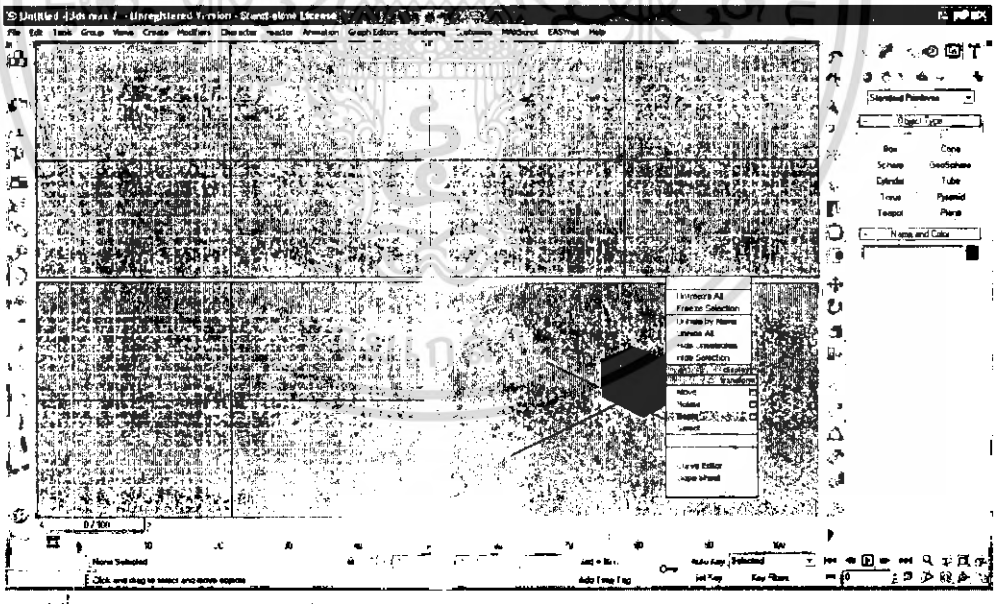
รูปที่ 2.15 แสดงโปรแกรม3D MAX

2. เลือกกลุ่มของวัตถุที่ต้องการจะสร้างในที่นี้ทำการเลือกชุดวัตถุมาตรฐาน หลังจากนั้นเราก็สามารถเลือกวัตถุต่าง ๆ ของชุดวัตถุมาตรฐานมาสร้างเป็นโมเดลที่ต้องการได้



รูปที่ 2.16 การสร้างวัตถุ

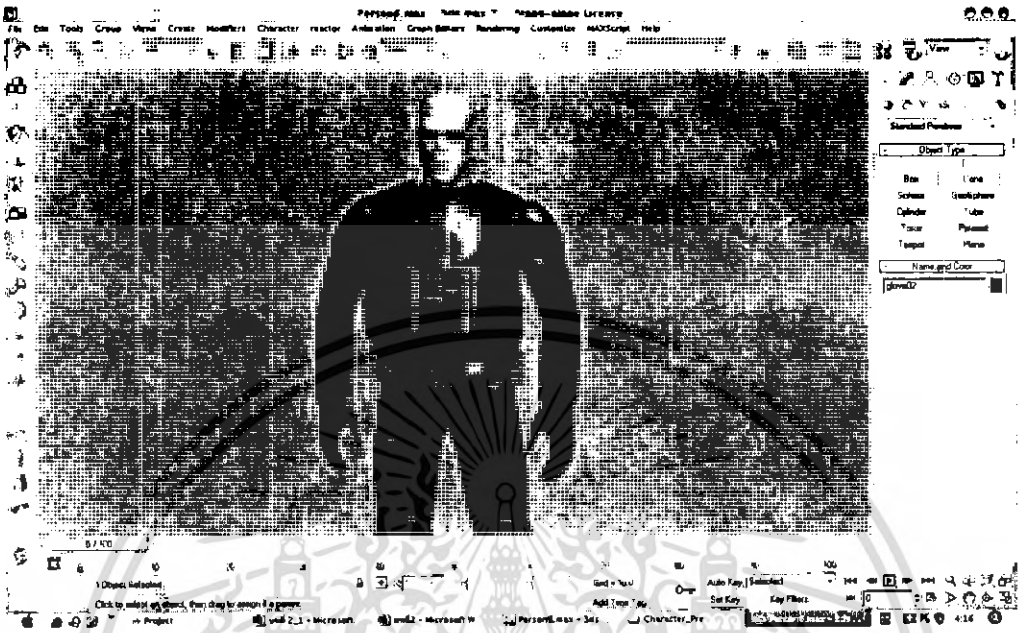
3. เราสามารถดำเนินการย่อขยาย เลื่อนพิกัด หรือหมุนวัตถุได้ตามต้องการ โดยการเลือกที่แถบเครื่องมือ หรือการคลิ๊กขวาที่ตัวออปเจกต์แล้วเลือกสิ่งที่เราต้องการแก้ไข



รูปที่ 2.17 การขยายขนาดวัตถุ

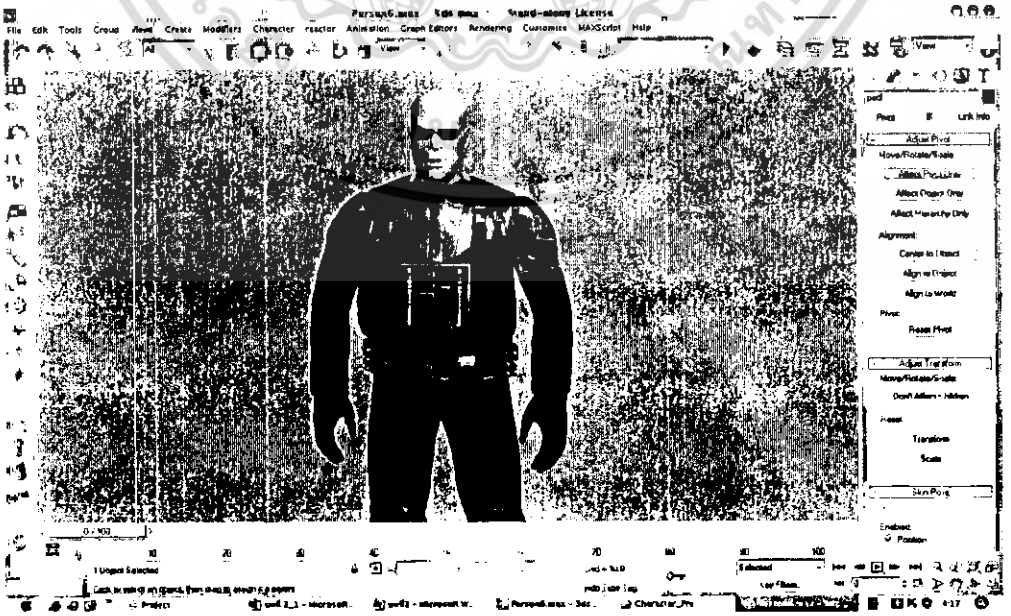
4. หลังจากทำการสร้างโมเดลได้ตามต้องการแล้ว เราต้องทำการเชื่อมต่อชิ้นส่วนแต่ละชิ้นส่วนเข้าด้วยกันเพื่อให้ชิ้นส่วนแต่ละชิ้นมีความสัมพันธ์กัน โดยทำการเลือกวัตถุลูกที่จะทำการเชื่อมต่อหลังจากนั้นให้กดปุ่มเลือกและ

เชื่อมในแถบเครื่องมือ แล้วทำการลากจากวัตถุลูกไปยังวัตถุแม่ที่เราต้องการ จะทำให้เราได้ต้นไม้ที่แสดงลำดับชั้นของวัตถุที่เราต้องการ



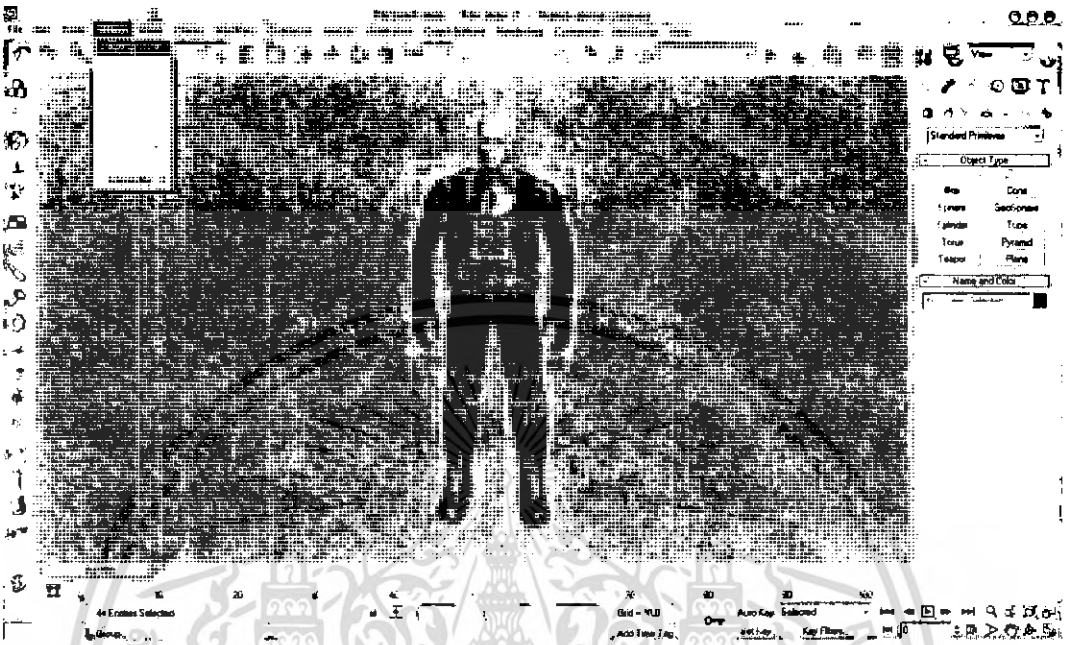
รูปที่ 2.18 การเชื่อมต่อวัตถุ

5. เมื่อเชื่อมวัตถุได้ตามต้องการแล้วเราต้องกำหนดจุดหมุนของแต่ละโมเดลให้ถูกต้อง โดยเลือกวัตถุที่ต้องการให้หมุนได้ก่อน จากนั้นให้ไปที่หน้าต่างลำดับชั้นต่อจากนั้นกดปุ่ม Affect Pivot Only เพื่อเลือกทำงานกับจุด Pivot เท่านั้น



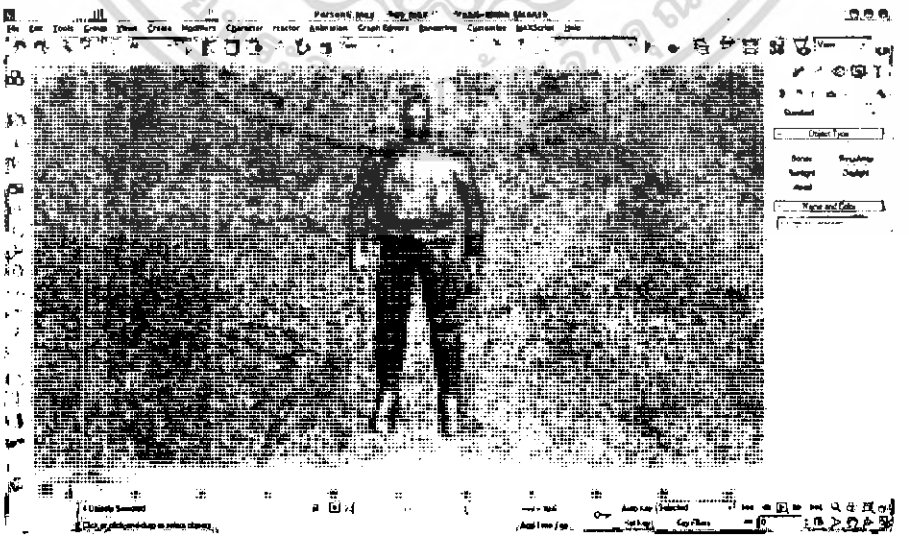
รูปที่ 2.19 การกำหนดจุดหมุน

6. ก่อนที่จะทำการใส่กระดูกให้กับโมเดลจะต้องรวมทุกชิ้นส่วนของโมเดลให้เป็นชิ้นเดียวกัน โดยการ group



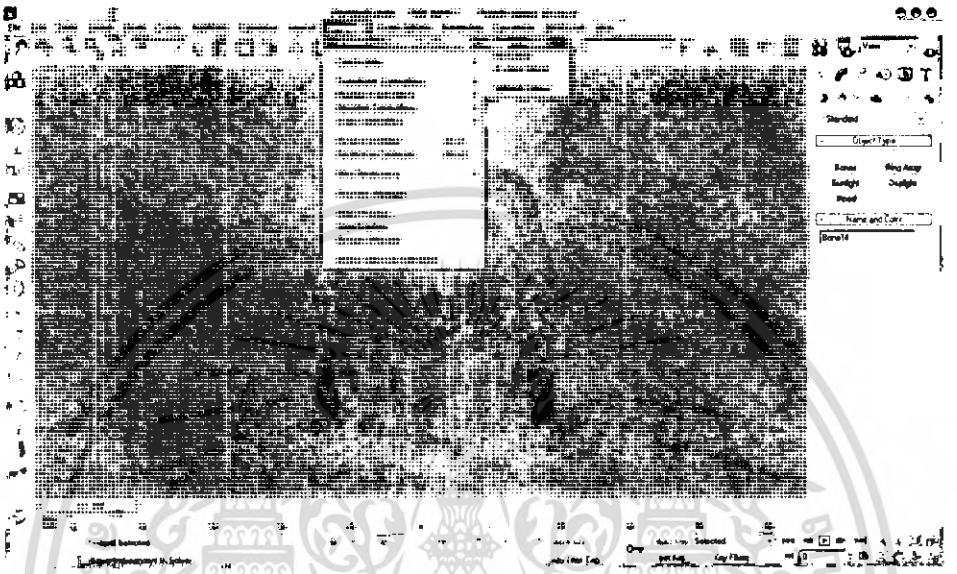
รูปที่ 2.20 การ group โมเดล

7. หลังจาก group โมเดลเรียบร้อยแล้วจะทำการใส่กระดูกให้กับโมเดลเพื่อให้การเคลื่อนไหวดูสมจริงโดย และให้เป็นการง่ายก็จะเลือกที่ตัวโมเดลแล้วกด Alt x เพื่อเป็นการล็อกตัวโมเดลเพื่อง่ายต่อการใส่กระดูก จากนั้นก็เลือก bone แล้วนำมาใส่ในโมเดล



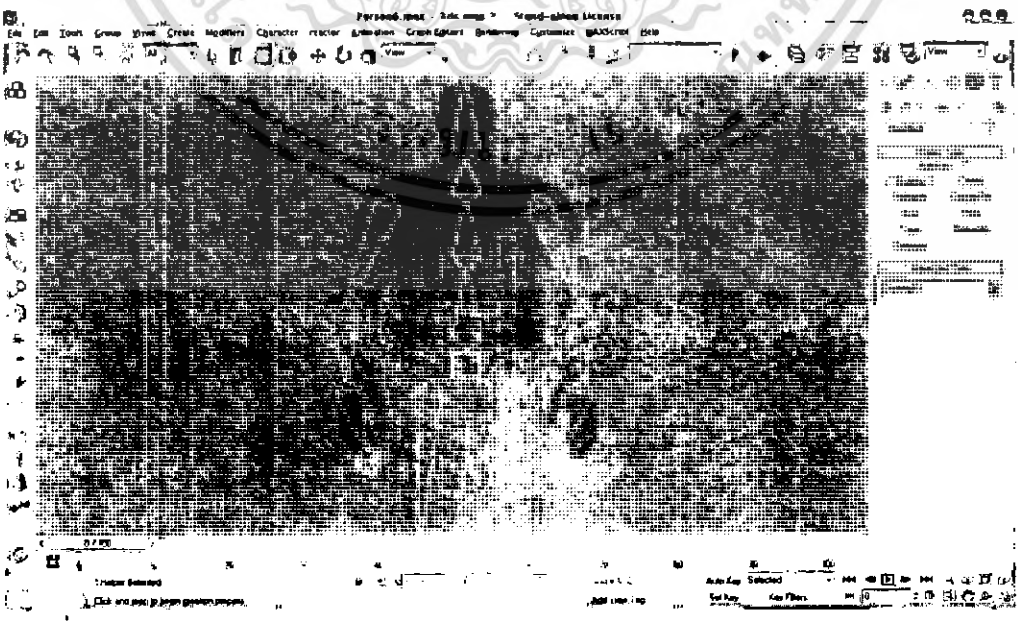
รูปที่ 2.21 ใส่กระดูกให้กับโมเดล

8. เมื่อใส่กระดูกจนครบทุกส่วนแล้วจะเชื่อมต่อการเคลื่อนไหวในแต่ละส่วนโดยใช้ IK Solver โดยเริ่มจากเลือกที่กระดูกส่วนต้นแล้วเลือก IK Solver แล้วไปเลือกที่ส่วนปลาย ก็จะทำให้กระดูกเคลื่อนไหวได้สมจริงแล้ว



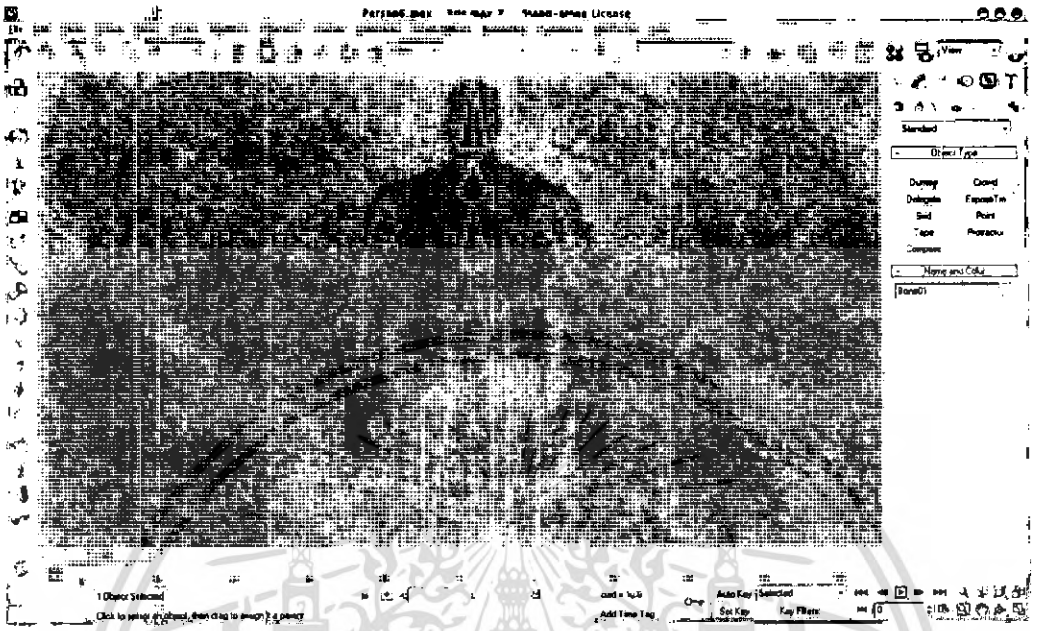
รูปที่ 2.22 แสดงการเชื่อมต่อกระดูก

9. ต่อมาจะทำการเชื่อมต่อกระดูกแต่ละส่วนเข้าด้วยกัน โดยเริ่มจากการใส่ Dummy ไว้กลางลำตัวที่จะทำการเชื่อมต่อ



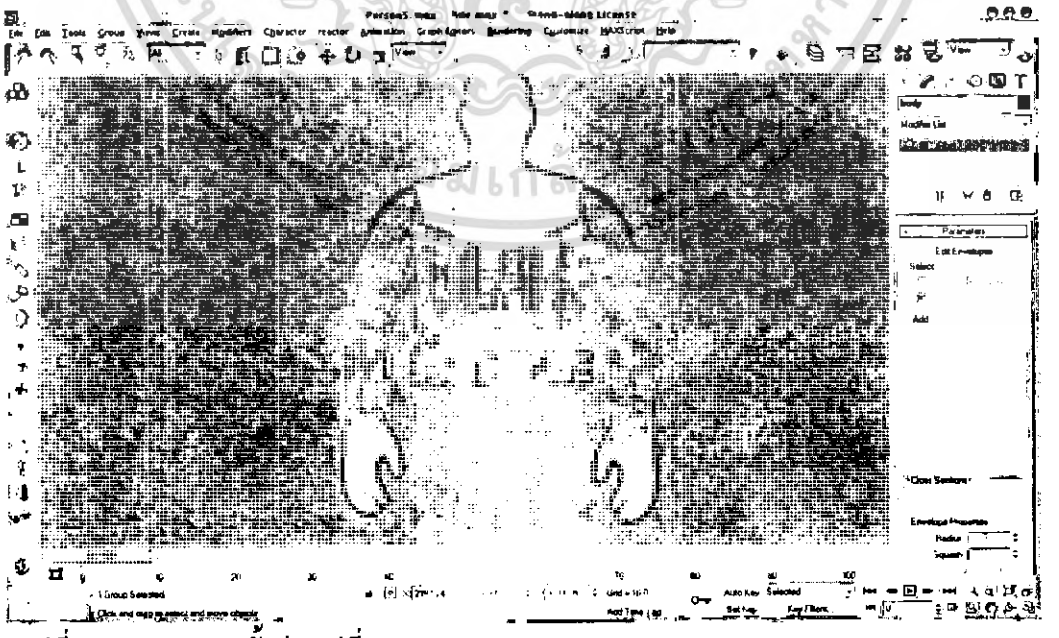
รูปที่ 2.23 แสดงสร้างDummy

10. จะโยงส่วนลำตัวเข้ากับ Dummy แล้วนำส่วนแขนและขาโยงกับส่วนลำตัวโดยกดปุ่มดังภาพแล้วลากโยงไปที่ๆต้องการจะเชื่อม



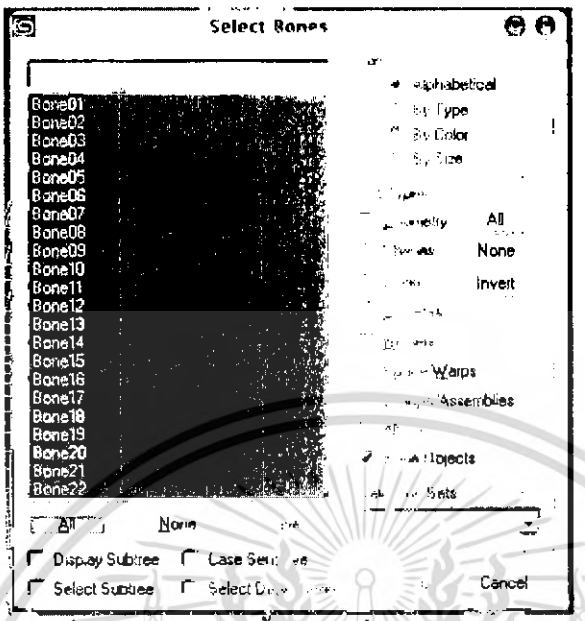
รูปที่ 2.24 แสดงสร้างเชื่อมโยง

11. ทำการใส่กระดูกเข้ากับตัวโมเดลโดยเลือกที่ตัวโมเดลแล้วไปที่ Modify แล้วเลือกแถบ skin แล้วกดปุ่ม Add



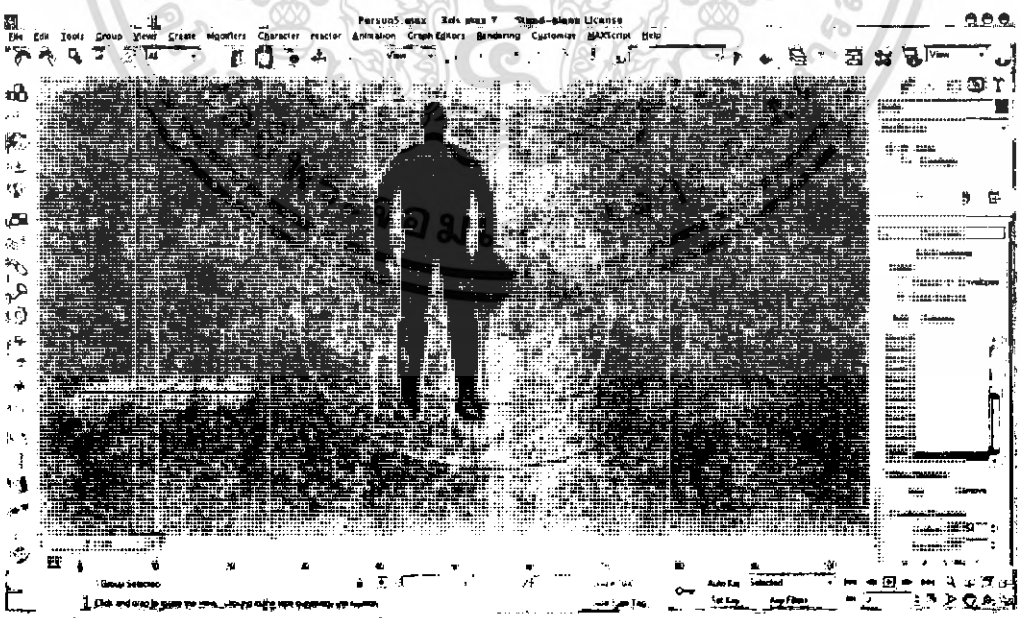
รูปที่ 2.25 แสดงการตั้งค่า รูปที่1

12. แล้วใส่กระดูกทั้งหมดที่ใช้



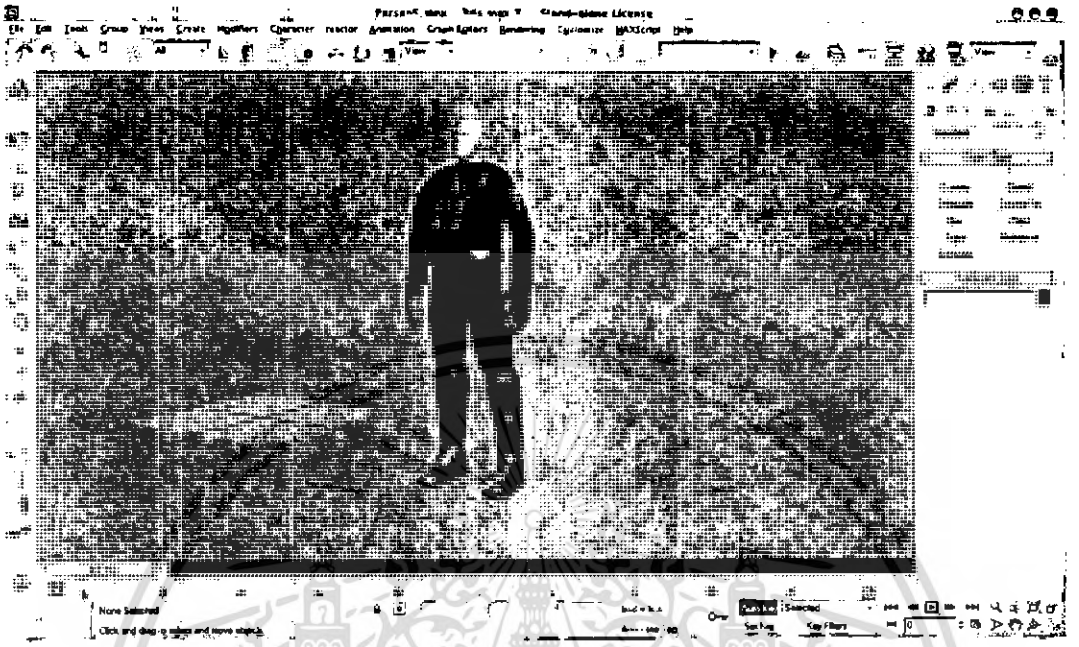
รูปที่ 2.26 แสดงการตั้งค่า รูปที่2

13. จะได้ดังภาพ และสามารถปรับแต่งกล้ามเนื้อได้ จากนั้นก็สามารถนำโมเดลไปทำ animation ที่สมจริงได้



รูปที่ 2.27 แสดงการใส่กระดูกที่เสร็จสมบูรณ์

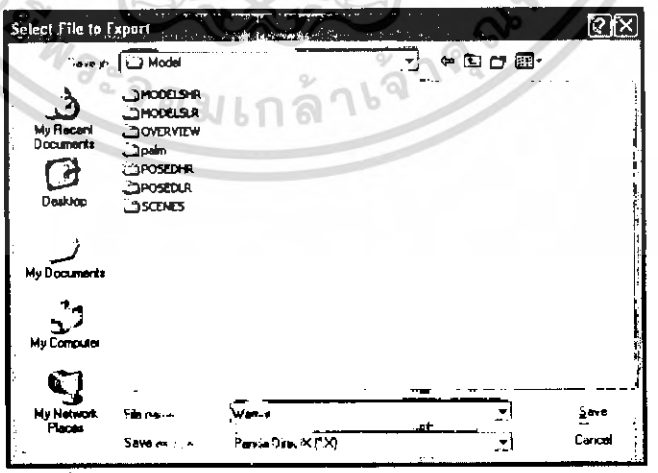
14. เริ่ม animation โดยกดที่ Auto Key หลังจากนั้นก็ขยับโมเดลไปตามแต่ละเฟรม เมื่อเสร็จเรียบร้อยก็เอาปุ่มออก



รูปที่ 2.28 แสดงการใส่เฟรม

2.7.2 การส่งออกโมเดลเป็นไฟล์ .X โดยใช้โปรแกรม Panda DirectX

1. ทำการก็อปปี้ไฟล์ PandaDXExport6.dle ไปไว้ที่โฟล์ดเดอร์ 3dMax/lib
2. จากเมนูบาร์ทำการเลือก File->Export จะได้ดังภาพ ให้ทำการเลือกชนิดของไฟล์ที่จะทำการบันทึกเป็น Panda DirectX(*.X)



รูปที่ 2.29 การบันทึกไฟล์ .X

3. ทำการกำหนดค่าให้กับโมเดลก่อนที่จะทำการส่งออกไป โดยในกรณีโมเดลที่จะส่งออกไปเป็นแอนิเมชันสิ่งสำคัญที่จะต้องกำหนดให้กับโมเดลเสมอคือเช็คบอกซ์ Include Animation (requires frames)ที่อยู่ในแท็บ 3DS Max Object ส่วนรายละเอียดของเช็คบอกซ์ต่าง ๆ มีดังนี้

Mesh definition เป็นการปรับกรอบให้กับตัวเลือกเมช

Material เป็นการกำหนดแมททีเรียลให้กับโมเดล

Include Animation (requires frames) กรณีส่งออกไฟล์แบบมีเลือกไว้ ถ้าไม่มีเฟรมหรือแอนิเมชันไม่ต้องเลือก

Bones ทำการเลือกถ้าเราใช้วัตถุกระดูกที่ใช้ modifier Skin หรือ Physique

Optimize mesh ทำการเลือก None หรือ Normal ไว้กรณี Optimized แล้วใช้ไม่ได้

Geometric เป็นการกำหนดให้กับโมเดลที่เป็นทรงเรขาคณิต

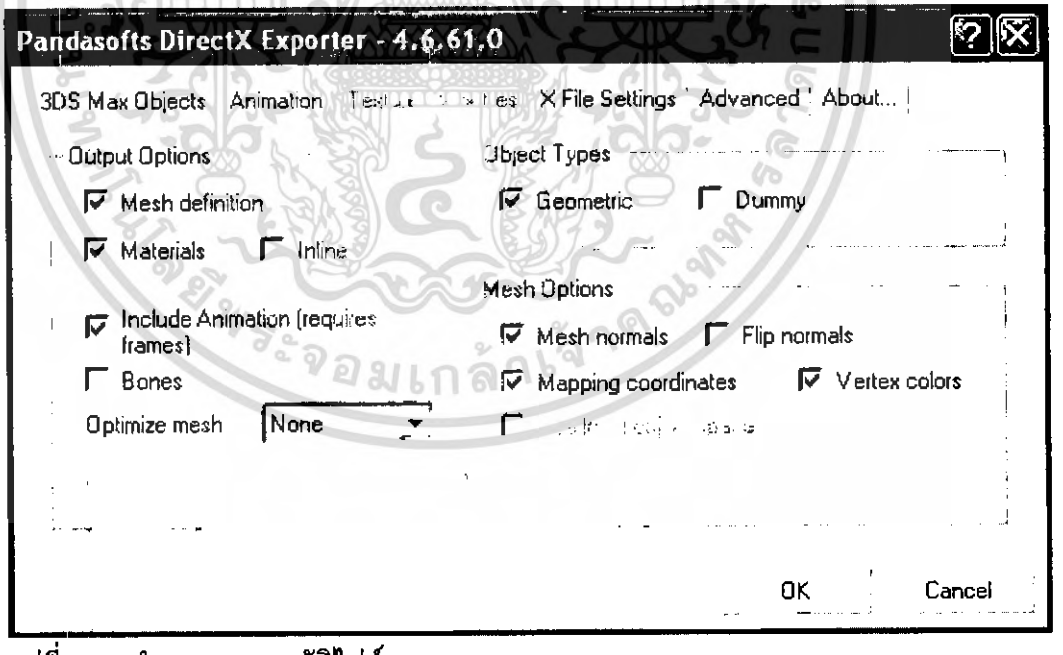
Dummy เป็นการกำหนดสำหรับโมเดลที่มีการใช้ดัมมี่

Mesh normals เป็นการกำหนดเมชให้เป็นแบบปกติ

Mapping coordinates เป็นการกำหนดค่า UV เพื่อไม่ให้ลายเสีย

Flip normals ไว้กลับด้านนอมอล

Vertex colors กำหนดแพนดาลงสีตามจุดยอด



รูปที่ 2.30 กำหนดคุณสมบัติไฟล์ .X

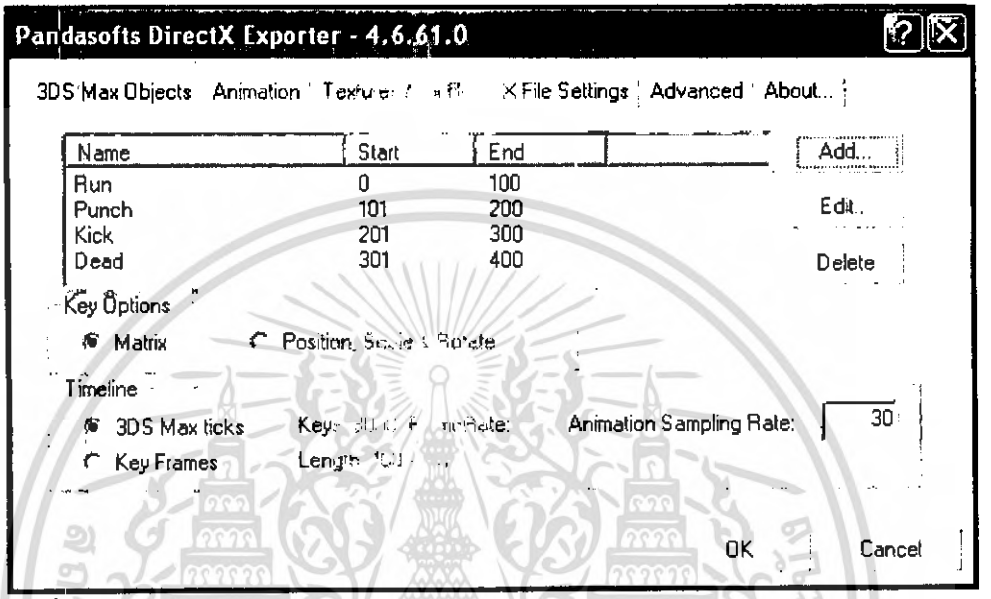
4. ที่แท็บ Animation ทำการกำหนดจำนวนแทร็คและกำหนดจำนวนเฟรมให้กับแต่ละแทร็คตามต้องการ โดยจำนวนแทร็คมักเท่ากับจำนวนพฤติกรรมที่โมเดลนั้นสามารถกระทำได้ ส่วนรายละเอียดของตัวเลือกต่าง ๆ มีดังนี้

Matrix กำหนดให้คีย์รวมกันเป็นเมตริกซ์

Position Scale Rotate กำหนดให้คีย์ต่างๆแยกออกจากกัน

3DS Max ticks กำหนดให้ช่วงเวลาเป็นแบบเฟรม

Key Frames กำหนดช่วงเวลาเป็นหน่วยวินาที



รูปที่ 2.31 การกำหนดแทรค

5. ที่แทบ X Files Setting ในส่วนของเฟรม X Files Animation Option ให้ทำการเช็คถูกที่เช็คบอกซ์ Include Animation Option เพื่อเป็นการบอกว่าจะมีการรวมคุณลักษณะของแอนิเมชันเข้าไปกับไฟล์ที่ทำการส่งออก จากนั้นให้กดตกลงเราก็จะได้ไฟล์ .X ตามต้องการ ส่วนรายละเอียดของตัวเลือกต่าง ๆ มีดังนี้

Text ทำการส่งออก .x เป็นข้อความที่อ่านได้ขนาดใหญ่

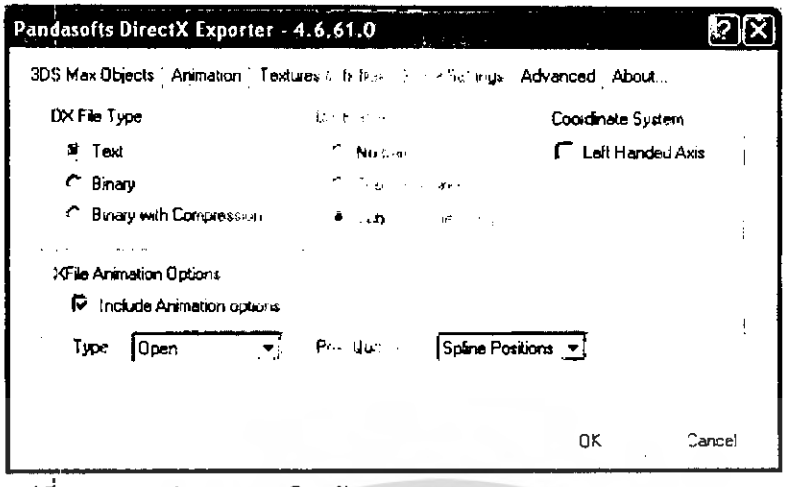
Binary ทำการส่งออก .x เป็นขนาดธรรมดาทำให้อ่านได้เร็ว

Binary with Compression ทำการส่งออก .x เป็นขนาดเล็กทำให้อ่านได้ช้า

No Frames เลือกไว้ถ้าไม่มีโมเดลไม่มีแอนิเมชัน

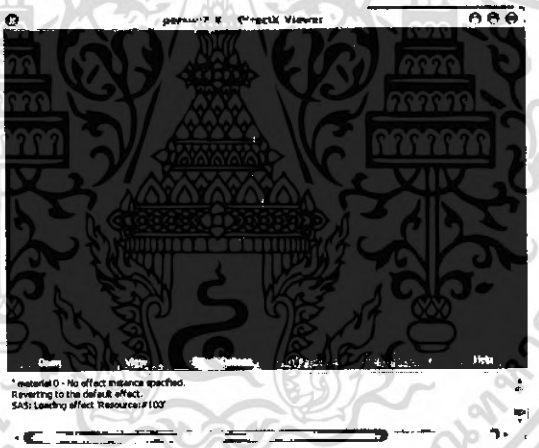
Top Frame only เลือกถ้าโมเดลมีแอนิเมชันแบบไม่ซับซ้อน

Sub frame hierarchy ถ้าโมเดลมีการใช้กระดูกให้เลือก



รูปที่ 2.32 การกำหนดแอนิเมชัน

6. ทดสอบผลลัพธ์ของไฟล์ .X โดยทำการเปิดโปรแกรม Mesh Viewer ที่มากับ DirectX SDK จากนั้นทำการเลือกไฟล์ที่ต้องการทดสอบจะได้ผลลัพธ์ดังภาพ



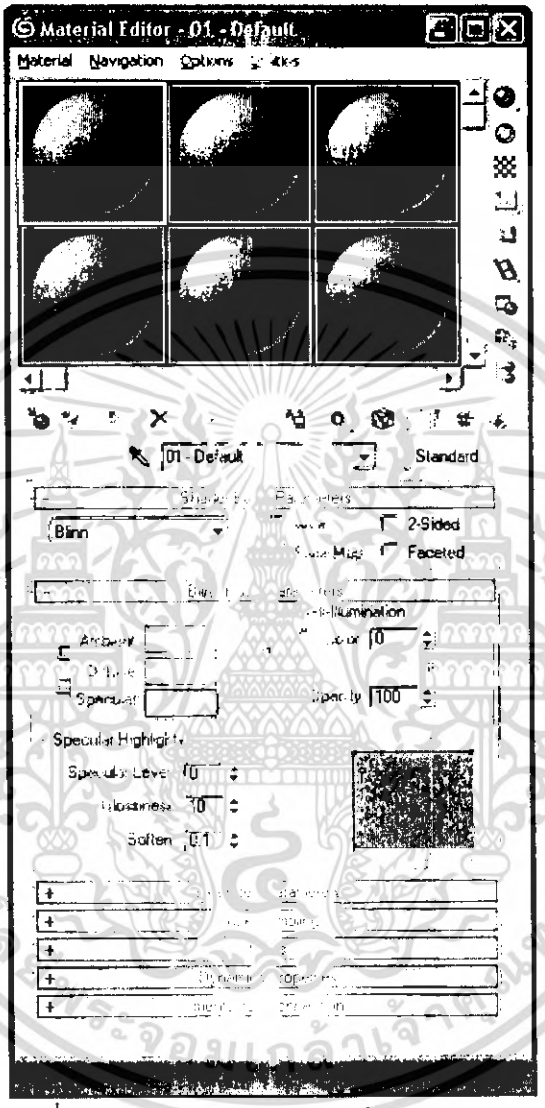
รูปที่ 2.33 การทดสอบแอนิเมชัน

2.7.3 การสร้างและควบคุมพื้นผิวให้กับโมเดล

การกำหนดพื้นผิวให้โมเดลที่สร้างขึ้นจะถูกแบ่งออกเป็นเรื่องใหญ่ ๆ 2 เรื่องคือ Shading และ Texture

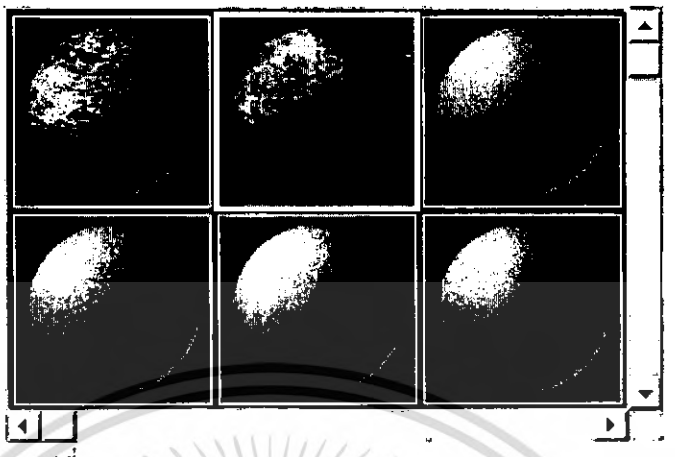
- **Shading** คือการกำหนดรายละเอียดของตัวพื้นผิว เช่น ความมันวาว หรือการสะท้อนของผิว
- **Texture** คือการกำหนดลวดลายของพื้นผิด เช่น พื้นมีลายเป็นไม้ รวมทั้งการแปะรูปภาพหรือลวดลายที่สร้างขึ้นเองลงบนพื้นผิว

สำหรับ 3d studio max 7 การควบคุมรายละเอียดทั้งสองข้อที่กล่าวมานี้สามารถทำได้ด้วย การกำหนดค่าต่างๆ จากหน้าต่าง Material Editor โดยไปที่ Rendering -> Material Editor จากโปรแกรม 3d studio max 7 ก็จะขึ้นหน้าต่าง Material Editor มาให้ดังรูป



รูปที่ 2.34 ส่วนประกอบของหน้าต่างควบคุมการสร้างพื้นผิว (Material Editor)

- **Material Slot** ช่องแสดงตัวอย่างของ Material ที่สร้างขึ้น



รูปที่ 2.35 Material Slot

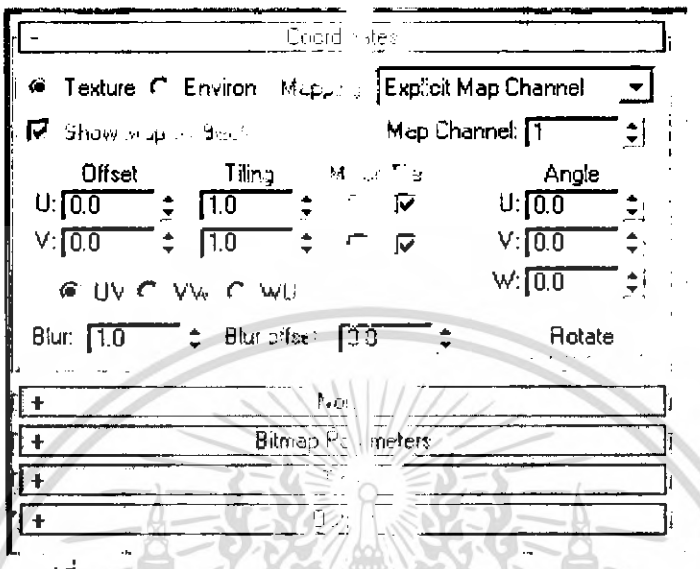
- **Material Toolbar** ชุดเครื่องมือสำหรับควบคุมหน้าต่าง Material Editors ประกอบไปด้วยเครื่องมือสำหรับเลือก ลบ และเรียก Material ที่เก็บไว้ในโปรแกรมขึ้นมาใช้งาน

รูปที่ 2.36 Material Toolbar

- **Material Editors Options** ชุดเครื่องมือควบคุม Option การแสดงผลในหน้าต่าง Material Editors

รูปที่ 2.37 Material Editors Options

- **Rollouts** เป็นส่วนที่เราจะเข้าไปกำหนดค่าต่าง ๆ เพื่อเปลี่ยนแปลงรายละเอียดของพื้นผิวให้เป็นไปตามที่เราต้องการ ในการทำงานการกำหนดค่าต่าง ๆ



รูปที่ 2.38 Rollouts

การปรับคุณสมบัติของพื้นผิวให้กับโมเดล

- เปิดหน้าต่าง Material Editors ขึ้นมา
- แดรกเมาส์ลาก Material จาก Slot มายังโมเดลที่ต้องการ
- ภายในหน้าต่าง Material Editors > Blinn Basic Parameters Rollout คลิกที่ช่องสีหลัง Diffuse เพื่อเปลี่ยนสีให้กับ Material

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

ลำดับขั้นตอนในการดำเนินงานวิจัยทั้งหมดแบ่งออกเป็น 3 ส่วน ได้แก่ ส่วนของการวิเคราะห์และออกแบบเกม ส่วนของการลงมือปฏิบัติจริง และส่วนของการทดสอบเกม

1) ขั้นตอนการวิเคราะห์และออกแบบเกม

เป็นการวิเคราะห์ความต้องการว่าต้องการจะสร้างเกมในรูปแบบใด ไม่ว่าจะเป็นเนื้อหาภายในเกม กติกาการเล่น ระบบการเล่น แนวทางการเล่น รูปแบบตัวละคร ฉาก เสียง เนื้อหาของเกม

2) ขั้นตอนการลงมือปฏิบัติการจริง

จะเป็นการลงมือปฏิบัติในการเขียนโค้ดของส่วนเกมขึ้นมาจริงๆ โดยเขียนคำสั่งต่างๆ ที่มีการนำ ฉาก ตัวละคร และเสียงประกอบ ที่ได้สร้างไว้มาใช้ อีกทั้งยังวางโครงสร้างของโปรแกรม และการทำงานของเกม

3) ขั้นตอนการทดสอบเกม

ขั้นตอนนี้จะทำการนำโปรแกรมเกมที่ได้เขียนขึ้นเสร็จแล้วมาทำการตรวจสอบเพื่อหาข้อผิดพลาดต่างๆ โดยนำไปให้ผู้อื่นช่วยทดสอบและประเมิน จากนั้นจึงนำกลับมาปรับปรุงและจัดทำเป็นเวอร์ชันสมบูรณ์

3.1 ขั้นตอนการวิเคราะห์และออกแบบเกม

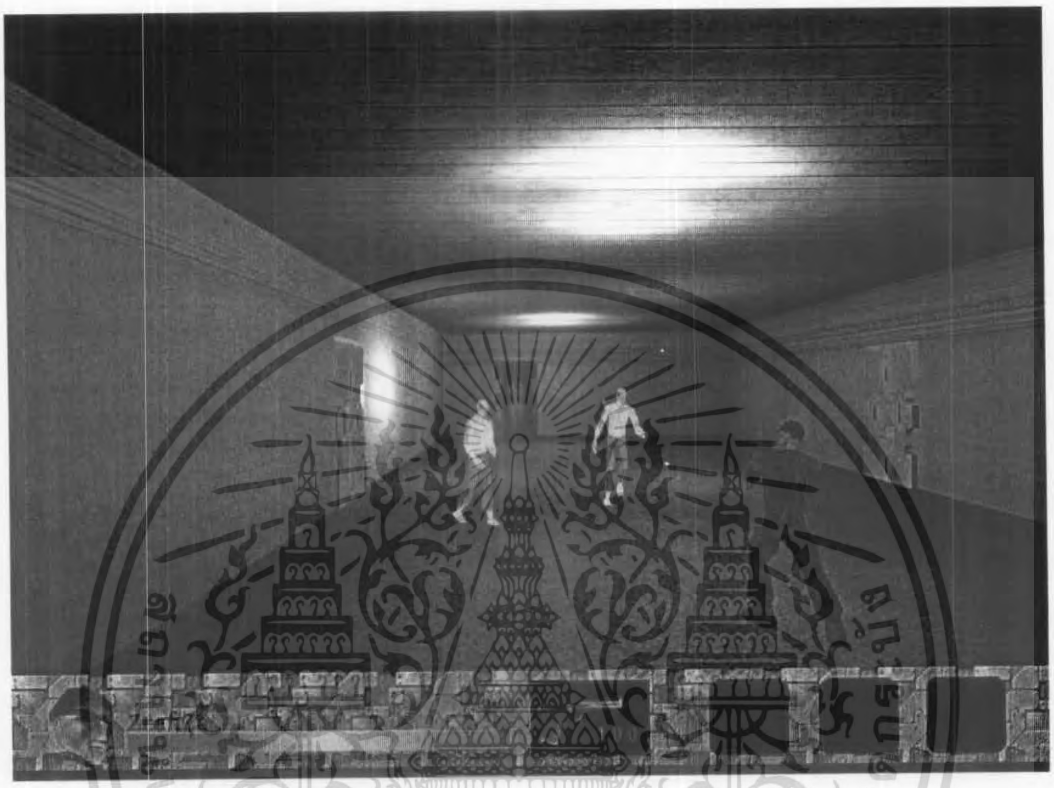
3.1.1 การออกแบบรูปแบบ กติกาการเล่นและระบบเกม

"ZtanZa" เป็นเกมแนวเดินยิงมุมมองบุคคลที่ 3 โดยจะมีลักษณะเป็นการเล่นตะลุยด่านเพียงคนเดียว โดยมีจุดประสงค์ คือการทำลายศัตรูต่าง ๆ ที่จะเข้ามารุมทำร้ายเพื่อที่จะเอาชีวิตรอดให้ได้ อีกทั้งยังต้องทำภารกิจต่าง ๆ ให้สำเร็จอีกด้วย โดยการบังคับจะเป็นการใช้ คีย์บอร์ด เพื่อให้เป็นเกมแนวแอคชันสมจริงมากขึ้น โดยเมื่อเข้าสู่เกมแล้วจะมีสิ่งที่ต้องทำต่าง ๆ ดังนี้

- การต่อสู้กับศัตรูเพื่อเอาตัวรอดให้ได้
- การเก็บไอเท็มต่างๆภายในเกม
- การนำไอเท็มต่างๆที่ได้ในเกมมาไขปริศนาเพื่อหาคำตอบและนำไปสู่ปริศนาต่อไป
- การจบเกม

3.1.1.1 การต่อสู้กับศัตรูเพื่อเอาตัวรอดให้ได้

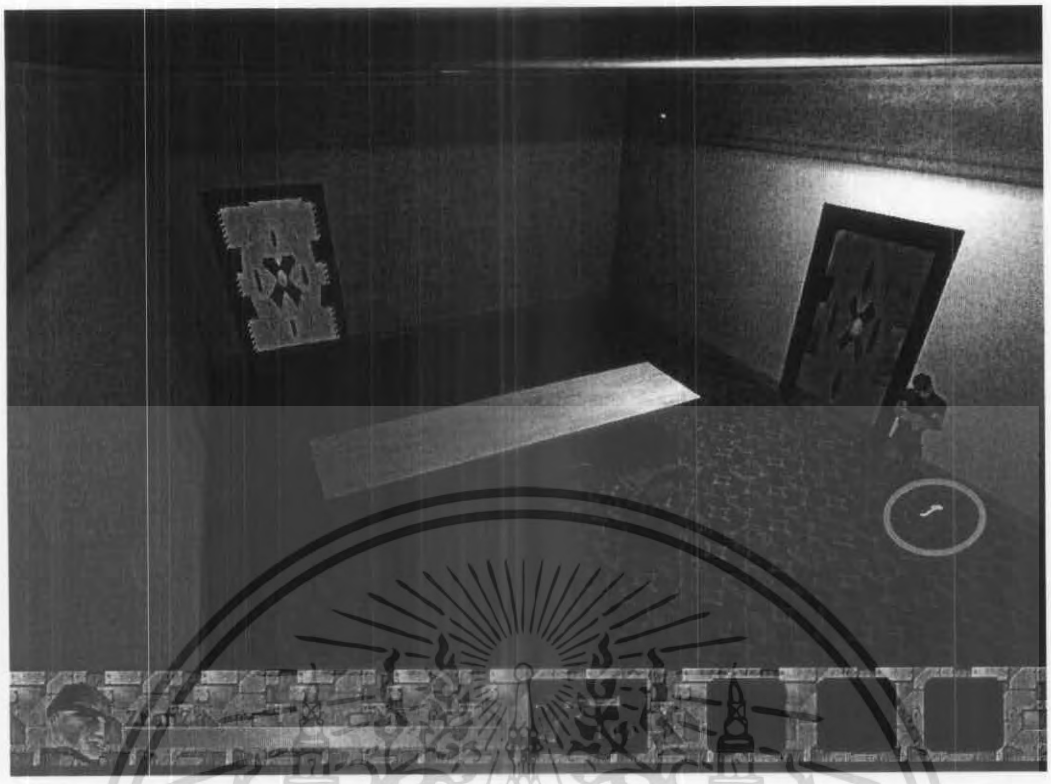
การต่อสู้กับศัตรูจะเป็นการที่เรานำอาวุธต่าง ๆ ที่มีอยู่ ซึ่งก็คือปืนสั้น มาทำการต่อสู้กับเหล่าศัตรูต่าง ๆ ภายในเกมที่จะเข้ามารุมทำร้ายเรา เพื่อจะสามารถมีชีวิตรอดต่อไปในเกมได้



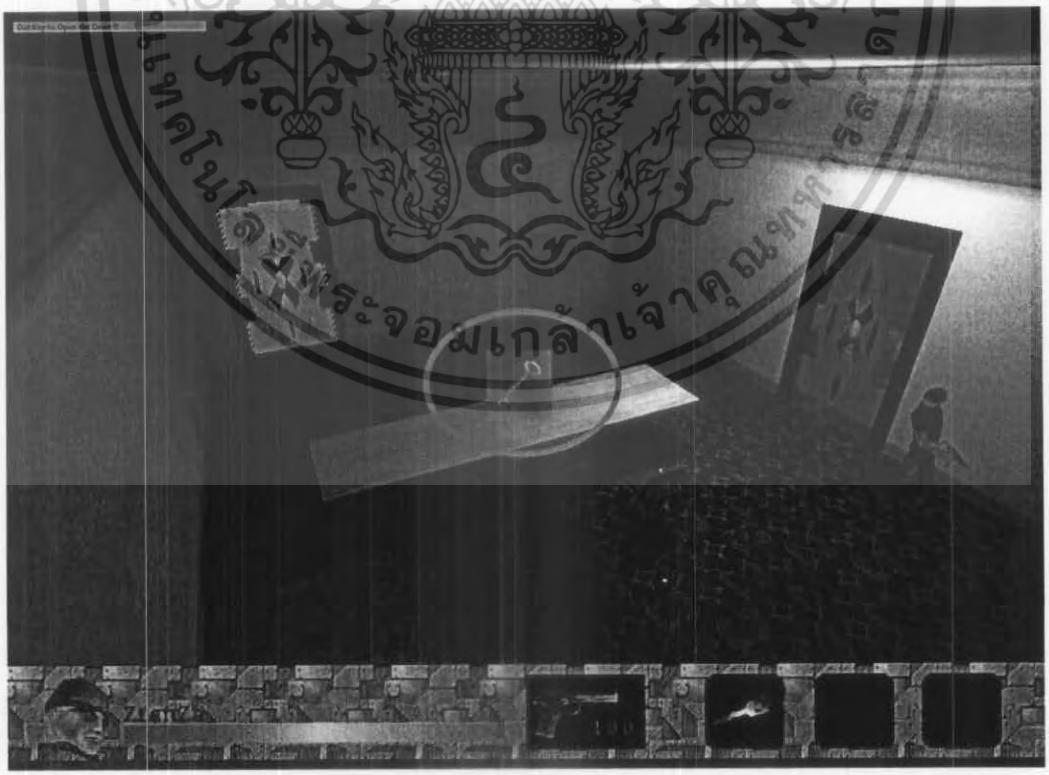
รูปที่ 3.1 แสดงการต่อสู้กับศัตรู

3.1.1.2 การเก็บไอเท็มต่าง ๆ ภายในเกม

ภายในเกมจะมีไอเท็มต่าง ๆ ให้เก็บ ไม่ว่าจะเป็นลูกกระสุนปืนที่ต้องใช้ ยาเพิ่มพลังชีวิตหรือแม้กระทั่งกุญแจต่าง ๆ ที่จะใช้ในการไขปริศนาต่าง ๆ ภายในเกมให้สามารถดำเนินเกมต่อไปได้เรื่อย ๆ



รูปที่ 3.2 แสดงไอเท็ม



รูปที่ 3.3 แสดงการเก็บไอเท็ม

3.1.1.3 การนำไอเท็มต่าง ๆ ที่ได้ในเกมมาไขปริศนาเพื่อหาคำตอบ

ภายในเกมจะมีปริศนาต่างๆมากมายเพื่อรอการหาคำตอบ ซึ่งการหาคำตอบจะต้องนำไอเท็มต่างๆที่เก็บได้มาไขปริศนา และนำไปสู่ปริศนาต่อไป โดยผู้เล่นจะต้องทำการหาคำตอบของปริศนาไปเรื่อย ๆ จนกระทั่งปริศนาทั้งหมดถูกไขจนกระจ่าง

3.1.1.4 การจบเกม

เมื่อผู้เล่นสามารถเอาชีวิตรอดมาได้จากการรุมทำร้ายของเหล่าศัตรูและสามารถไขปริศนาภายในเกมได้ทั้งหมดก็เป็นอันจบเกม

3.1.2 การออกแบบภาพและกราฟิก

รูปแบบของภาพในเกม มีลักษณะเป็น 3 มิติ โดยจะพยายามใช้แสงเงา และ เอฟเฟกต์ต่างๆที่ Irrlicht Engine มีเตรียมมาให้ เพื่อให้ภาพที่ออกมานั้นสมจริงที่สุด ซึ่งตัวละครอาวุธ แผนที่ และองค์ประกอบอื่นๆจะใช้โปรแกรม 3ds max 7 ช่วยในการออกแบบ แต่ในส่วนการออกแบบหน้าจอจะใช้โปรแกรม Adobe Photoshop CS เข้าช่วย

3.1.3 การออกแบบเสียงและซาวน์เอฟเฟกต์ประกอบ

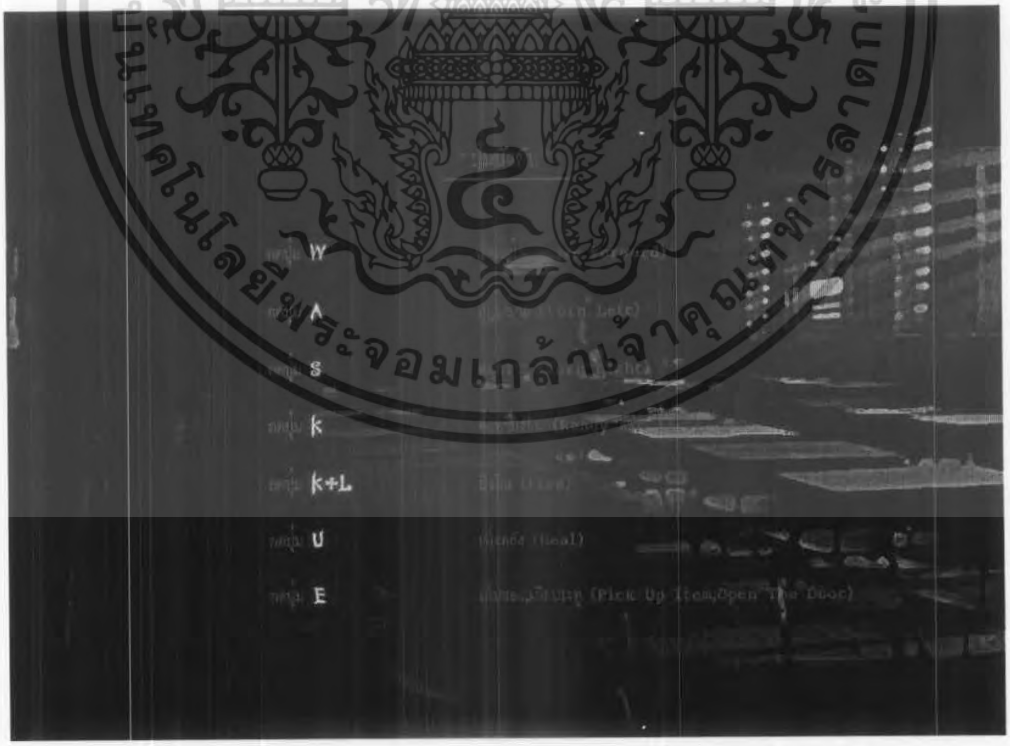
ไฟล์เสียงที่ใช้เป็นเอฟเฟกต์ในเกมมีหลายประเภทเช่น WAVE, MP3, OGG เป็นต้น ซึ่งแบ่งเป็นสองส่วนด้วยกันคือเสียงแบ็คกราวน์ซึ่งเป็นเสียงประกอบของฉาก และเสียงเอฟเฟกต์ซึ่งเป็นเสียงประกอบของอิริยาบถต่าง ๆ ภายเกม เช่น เสียงยิงปืน เสียงตะโกน เสียงย่ำเท้า เสียงระเบิด เป็นต้น ซึ่งเสียงใส่ภายในเกมสามารถเพิ่มอรรถรสในการเล่นเกมของผู้เล่นได้ โดยโปรแกรมที่นำมาใช้สำหรับการสร้างและแปลงไฟล์เสียงและซาวน์เอฟเฟกต์ ได้แก่ Nero Wave Editor , Sound Recorder และ Jet Audio

3.2 ขั้นตอนการทำงานจริง

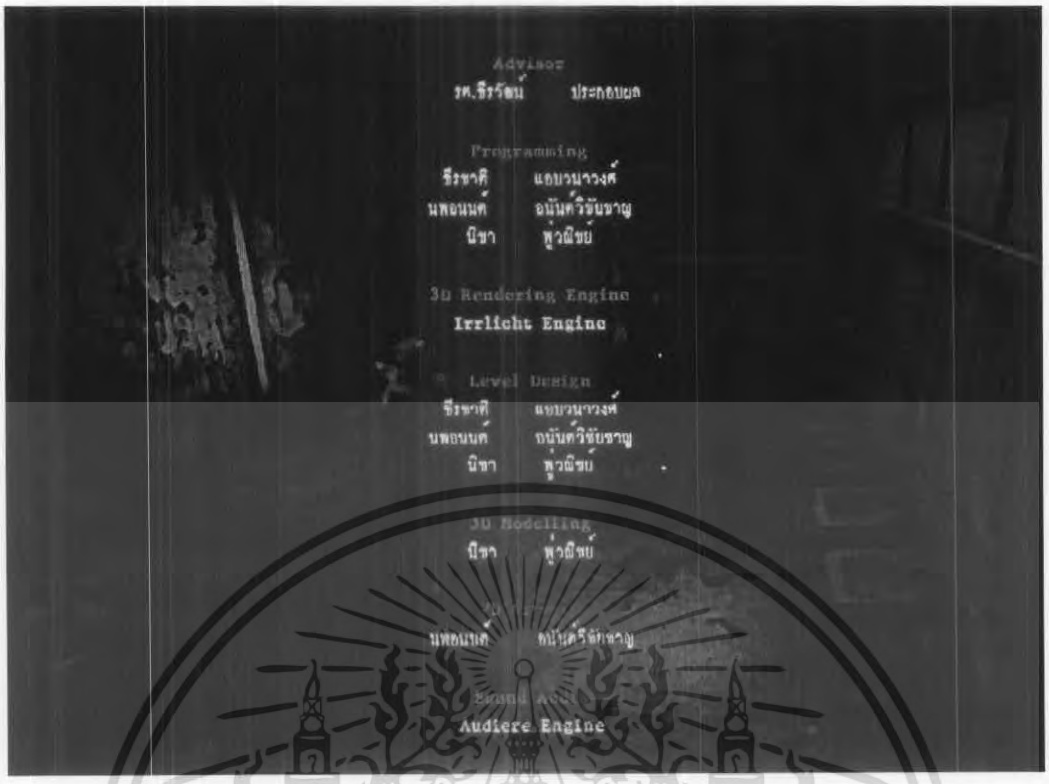
3.2.1 การออกแบบหน้าจอภายในเกม

โดยการออกแบบหน้าจอหลักต่างๆไม่ว่าจะเป็น หน้าจอเข้าเกม หน้าจอดูวิธีการเล่น และหน้าจอแสดงผลอื่นๆ

รูปที่ 3.4 แสดงหน้าจอเมนูของเกม



รูปที่ 3.5 แสดงหน้าจอวิธีการเล่นของเกม



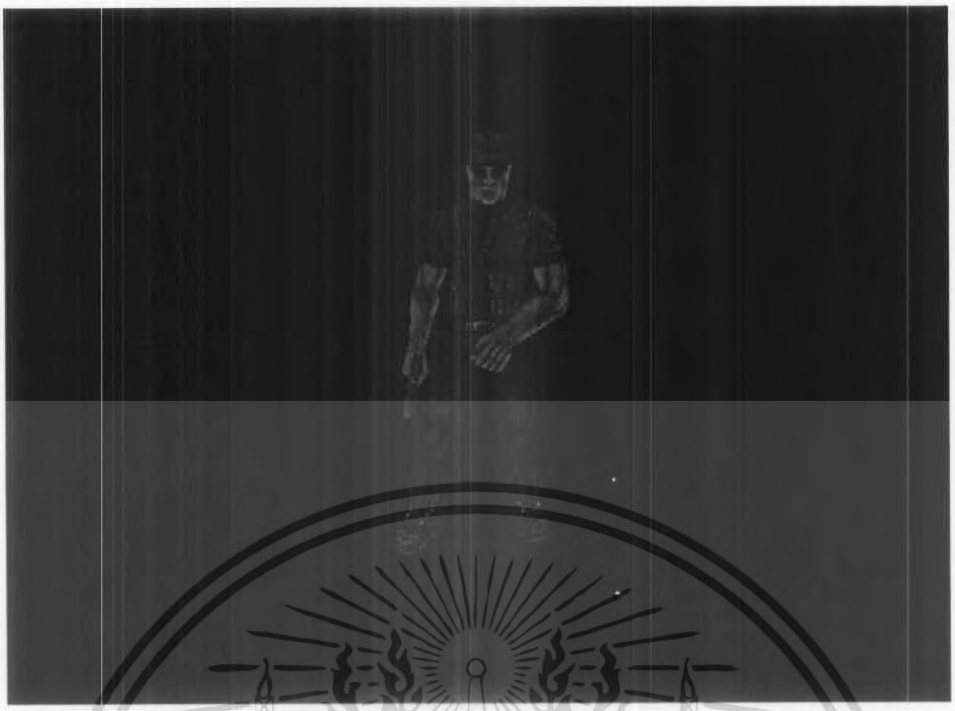
รูปที่ 3.6 แสดงหน้าจอรายชื่อผู้จัดทำ

3.2.2 สร้างโมเดลที่ใช้ในเกมทั้งหมด

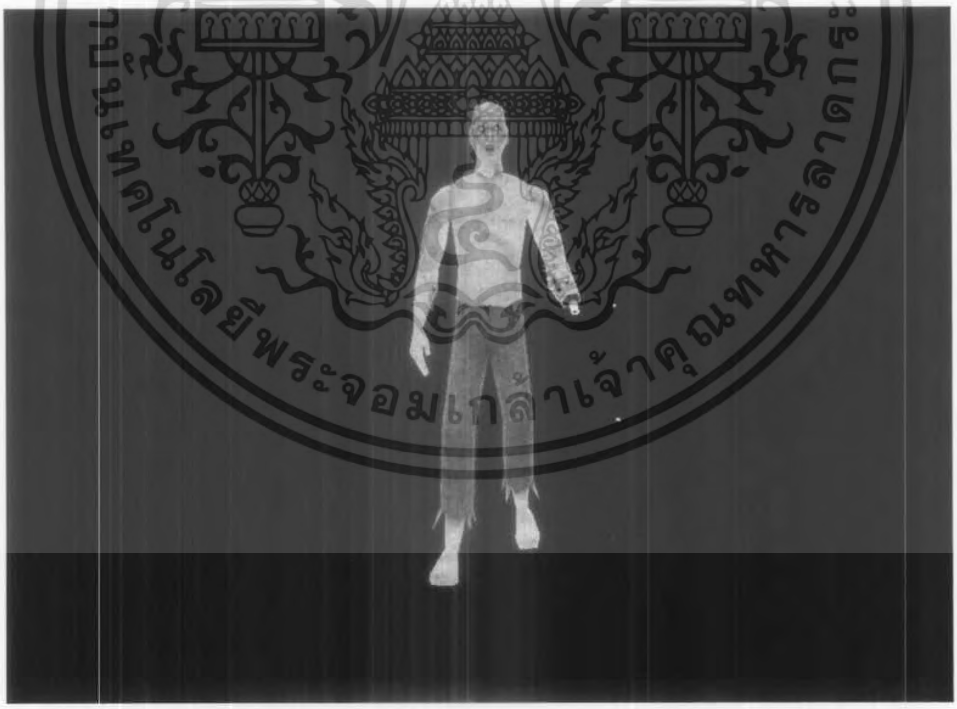
ส่วนประกอบต่าง ๆ ภายในเกมล้วนแต่โมเดลที่สร้างขึ้นทั้งสิ้น ซึ่งโมเดลเหล่านี้ส่วนใหญ่สร้างมาจากโปรแกรม 3ds max 7 ประกอบกับการนำโมเดลที่มีอยู่แล้วมาประยุกต์ใช้ โดยโมเดลภายในเกมจะประกอบด้วย 3 ส่วนหลักด้วยกัน ได้แก่ โมเดลตัวละครในเกม โมเดลแผนที่ โมเดลอาวุธและอุปกรณ์ต่าง ๆ

3.2.2.1 โมเดลตัวละครในเกม (Character)

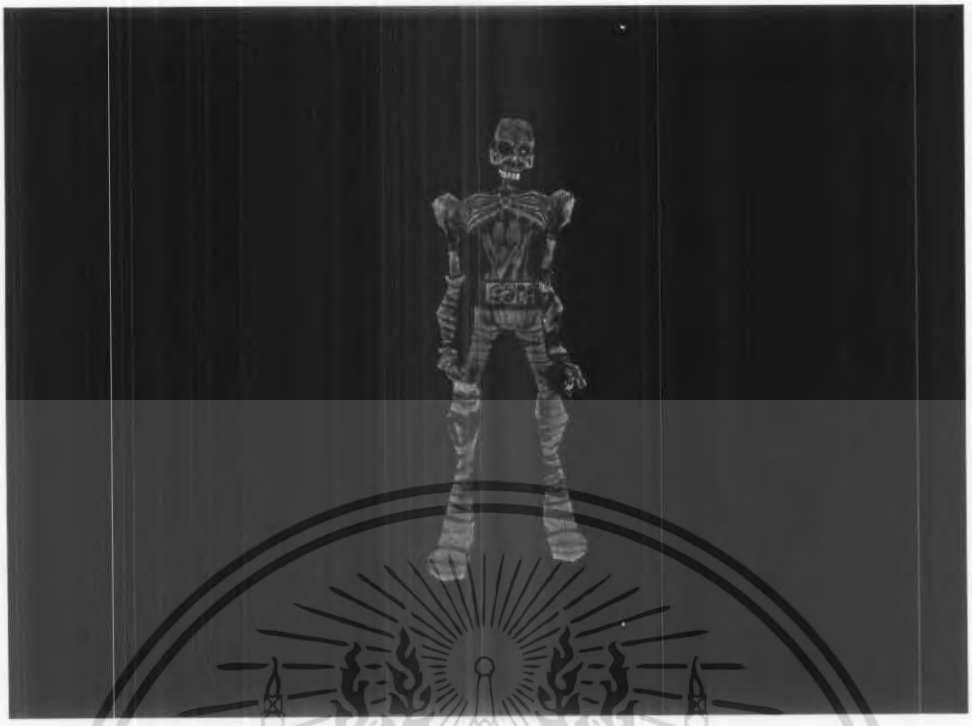
ตัวละครภายในเกมเหมือนอย่างที่กล่าวเอาไว้ ซึ่งโมเดลที่จะนำเข้ามาใช้ในเกมนี้จะใช้โมเดลที่สร้างด้วยโปรแกรม 3D Studio Max 7 (ไฟล์ .max) แล้วค่อยส่งออกไฟล์เป็น .x โดยใช้โปรแกรม PandaExport หรือการนำโมเดลที่มีอยู่แล้วมาประยุกต์ใช้เช่นพวกนามสกุล .md2 หรือ .pk3 เป็นต้น



รูปที่ 3.7 แสดงโมเดลตัวละครเอก



รูปที่ 3.8 แสดงโมเดลของศัตรูภายในเกม

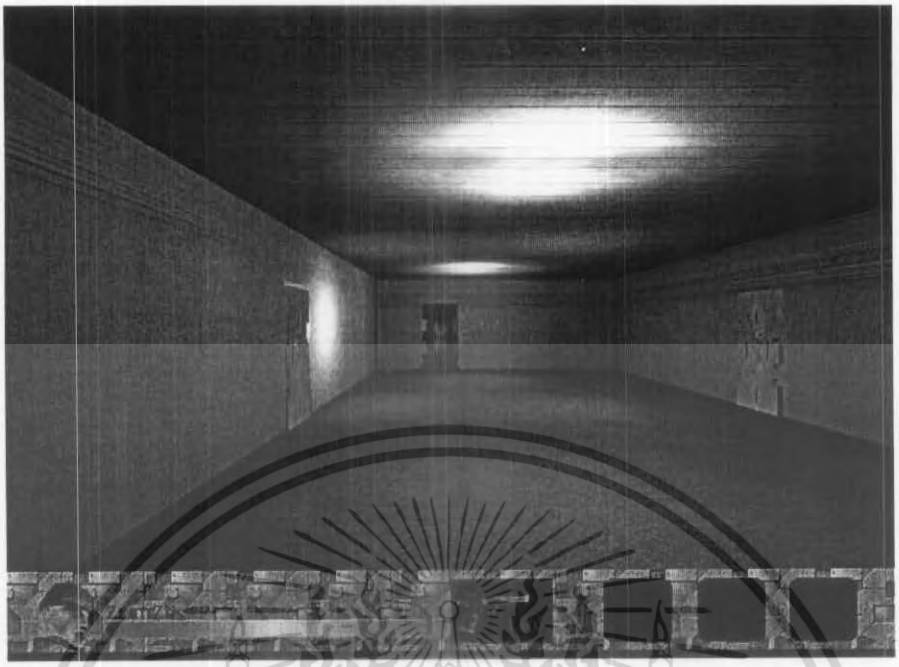


รูปที่ 3.9 แสดงโมเดลของหัวหน้าใหญ่

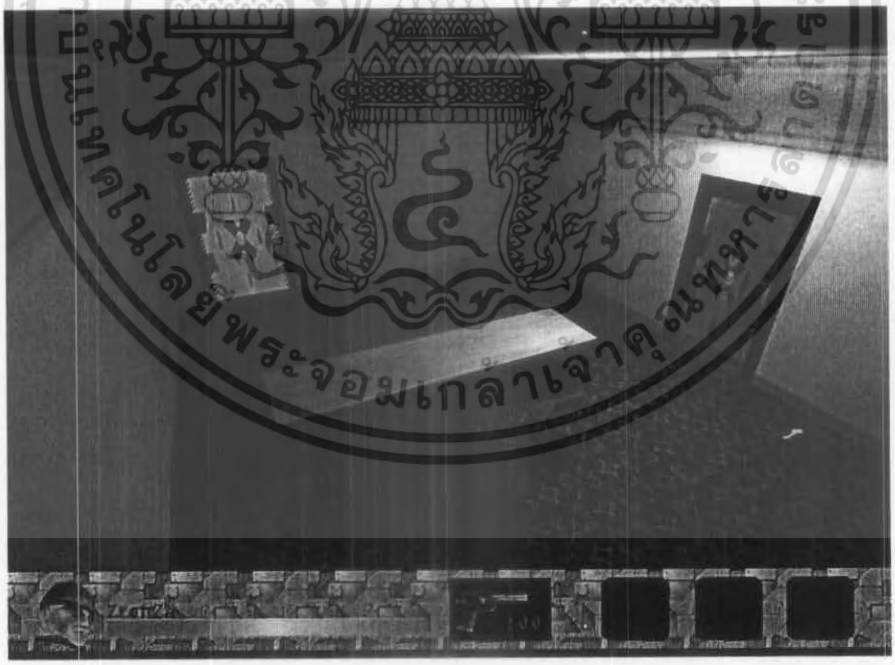
3.2.2.2 โมเดลแผนที่ (Map)

เราจะทำการโหลดเอาแผนที่ที่เราได้สร้างไว้แล้วโดยใช้โปรแกรม 3ds Max เพื่อนำเข้ามาใช้ภายในเกม ผสมผสานกับแผนที่ที่เรามีอยู่แล้วมารวมกันเป็นห้องต่างๆเพื่อให้ผู้เล่นได้ไขปริศนาในห้องต่างๆ

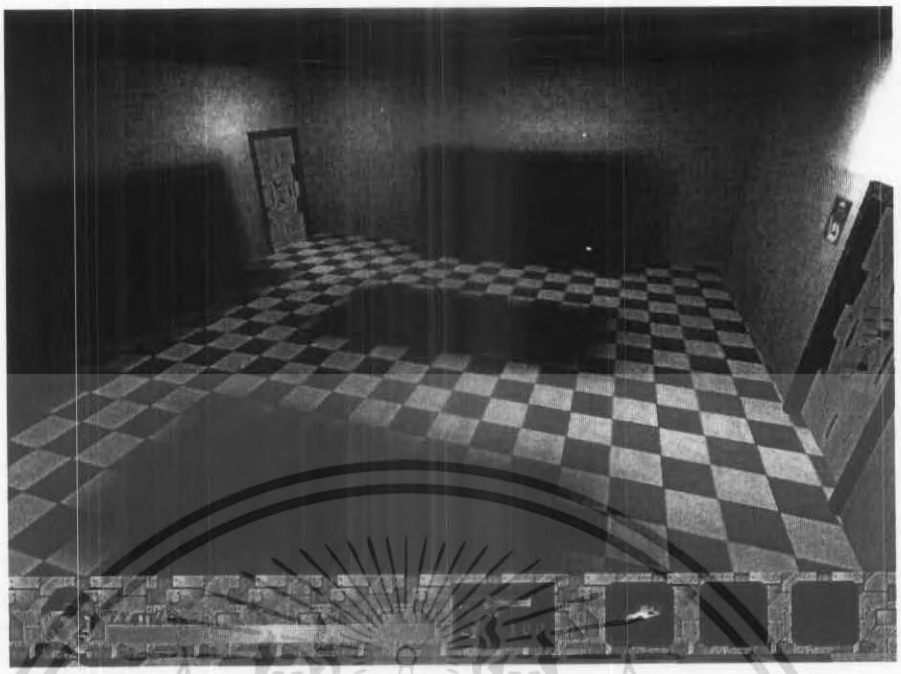
รูปที่ 3.10 แสดงโมเดลแผนที่ห้องที่หนึ่ง



รูปที่ 3.11 แสดงโมเดลแผนที่ห้องที่สอง

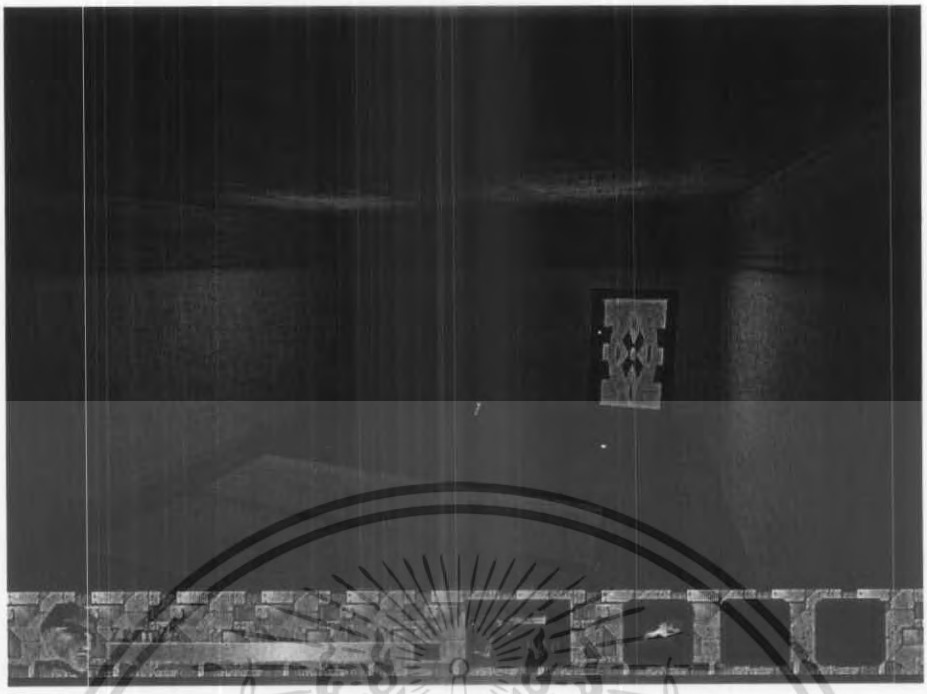


รูปที่ 3.12 แสดงโมเดลแผนที่ห้องที่สาม



รูปที่ 3.13 แสดงโมเดลแผนที่ห้องที่สี

รูปที่ 3.14 แสดงโมเดลแผนที่ห้องที่ห้า



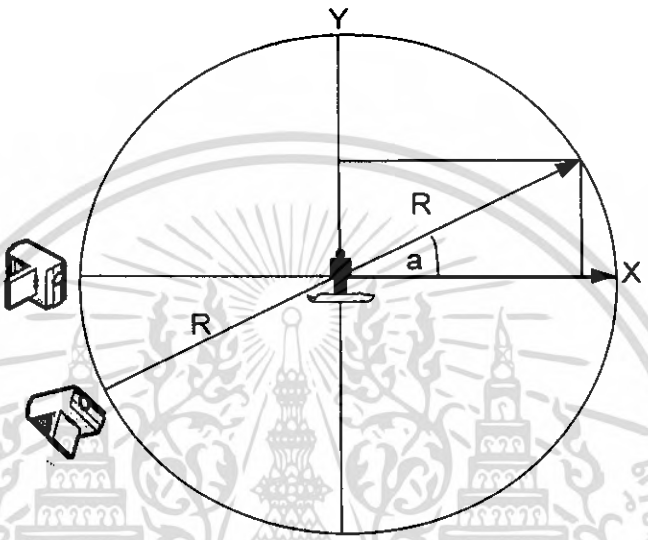
รูปที่ 3.15 แสดงโมเดลแผนที่ห้องที่หก

รูปที่ 3.16 แสดงโมเดลแผนที่ห้องที่เจ็ด

3.2.3 หลักการโดยรวมของโปรแกรม

3.2.3.1 หลักการของการเคลื่อนที่

เมื่อมีการเคลื่อนที่ในแนวแกนต่าง ๆ หากไม่มีการหมุนของตัวละครจะมีการเคลื่อนที่ ๆ ถูกต้อง แต่เมื่อตัวละครมีการหมุนไปจำนวน a องศาและเคลื่อนที่ในแนวแกน X เป็นระยะทาง R จะทำให้ตัวละครเคลื่อนไปในแกนต่าง ๆ ดังนี้

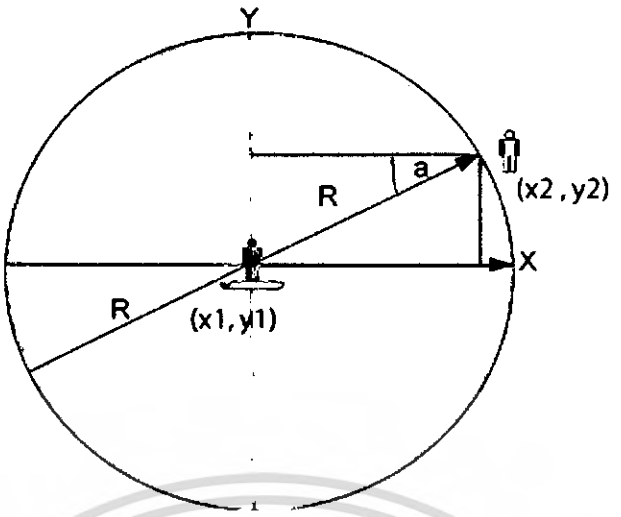


รูปที่ 3.17 แสดงการคำนวณการเคลื่อนที่ของตัวละครเอก

- แกน X จะเคลื่อนที่ไปในระยะ R*cos(a)
- แกน Y จะเคลื่อนที่ไปในระยะ R*sin(a)

3.2.3.2 หลักการเคลื่อนที่ของศัตรู

เมื่อตัวละครเอกมีการเคลื่อนที่ ศัตรูจะทำการดึงค่าตำแหน่งของตัวละครเอกเพื่อมาทำการคำนวณหาองศาที่จะทำให้ศัตรูหันหน้าไปหาตัวละครเอกได้อย่างถูกต้อง แล้วจึงนำองศาที่หาได้ไปทำการคำนวณหาเส้นทางเดินต่อไป โดยการหาค่ามุมนั้นเราจะนำระยะห่างของตัวละครเอกและตัวศัตรูมาหาค่าของ arctan แล้วได้ออกมาเป็นค่าของมุม a จากนั้นจึงนำมุม a ไปเข้าสูตรของหลักการเคลื่อนที่นั่นเอง สามารถดูรูปภาพประกอบได้ดังนี้

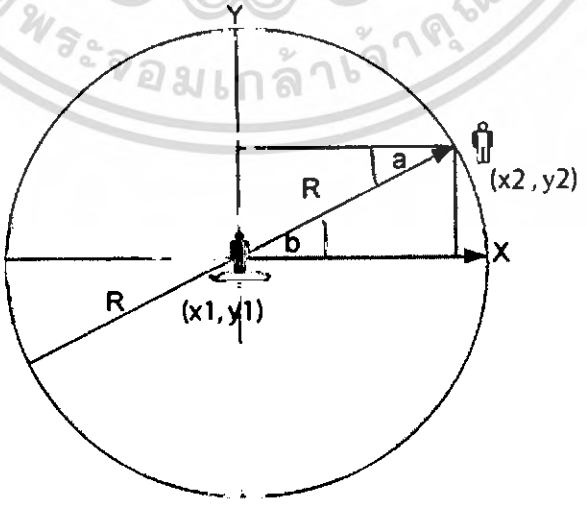


รูปที่ 3.18 แสดงการคำนวณการเคลื่อนที่ของตัวศัตรู

- มุม a = arctan((y2-y1) / (x2-x1))
- แกน X จะเคลื่อนที่ไปในระยะ R*cos(a)
- แกน Y จะเคลื่อนที่ไปในระยะ R*sin(a)

3.2.3.3 หลักการที่ใช้ในการคำนวณศัตรูถูกยิง

เมื่อทำการกดปุ่มยิงต้องทำการตรวจสอบว่ามุมของตัวละครเอกกับมุมที่ตัวศัตรูยืนทำมุมกับแนวแกนมีค่าตรงกันหรือไม่ ถ้าตรงกันก็คือศัตรูโดนยิง โดยทำการบวกลบความกว้างของแนวเส้นทางเพิ่มอีก 70 และต้องทำการตรวจสอบด้วยว่าศัตรูอยู่ด้านหน้าหรือหลังของตัวละครเอก สามารถดูรูปภาพประกอบได้ดังนี้



รูปที่ 3.19 แสดงการคำนวณการยิงปืนใส่ศัตรู

- มุม b ตัวละครต้องเท่ากับมุม a = arctan((y2-y1) / (x2-x1))
- ความกว้างของมุม a เท่ากับ ± 70
- ทำการตรวจสอบว่าศัตรูอยู่ข้างหน้าหรือหลัง

3.2.4 ขั้นตอนการเขียนโปรแกรม

ในขั้นตอนการเขียนโปรแกรมนั้น ได้ใช้ Visual C++ 6.0 สำหรับการพัฒนาโดยใช้เอนจิน ได้แก่

- Irrlicht Engine สำหรับดูแลเรื่องกราฟิกต่าง ๆ ภายในเกม(Graphic Engine)
- Audiere Engine สำหรับดูแลเรื่องเสียงต่าง ๆ ภายในเกม (Sound Effect Engine)

3.2.4.1 ฟังก์ชันการทำงานหลักในแต่ละเอนจิน

ตาราง 3.1 ฟังก์ชันการทำงานหลักของ Irrlicht Engine

IRRLICHT ENGINE	
ชื่อฟังก์ชัน	หน้าที่ การทำงาน
IrrlichtDevice::createDevice()	เป็นฟังก์ชันหลักที่จำเป็นที่สุดของการใช้ Irrlicht Engine โดยจะเป็นการกำหนดข้อมูลเบื้องต้นของการใช้งาน ไม่ว่าจะเป็นประเภทของการแสดงผล และขนาดหน้าจอ
IVideoDriver:: getVideoDriver()	เป็นฟังก์ชันเรียก Vidco Driver ของ Irrlicht ซึ่งจะจัดการการแสดงผล ทั้งยังใช้ระบุจุดเริ่มต้น และจุดสุดท้ายของการแสดงผล
ISceneManager:: getSceneManager()	เป็นฟังก์ชันเรียก Scene Manager ของ Irrlicht ซึ่งจะมีหน้าที่จัดการและควบคุมออปเจกต์อะไรก็ตามที่ใส่เข้ามาในฉาก (Scene) เช่น ตัวละคร แผนที่ หรือวัตถุต่างๆ เป็นต้น
IGUIEnvironment:: getGUIEnvironment()	เป็นฟังก์ชันเรียก Scenc Manager ของ Irrlicht ซึ่งจะมีหน้าที่จัดการและควบคุม GUI ที่ใส่เข้ามาในฉาก เช่น EditBox หรือปุ่มต่างๆ เป็นต้น

IAnimatedMesh:: getMesh()	เป็นการกำหนดโมเดลที่เราจะใช้
IAnimatedMeshSceneNode:: addAnimatedMeshSceneNode()	เป็นฟังก์ชันสร้างโหนดจาก Mesh ที่เราได้กำหนดไว้
IAnimatedMeshSceneNode::setMaterialTexture()	เป็นการเซต Texture ให้กับโหนดที่เราสร้างขึ้น
IrrlichtDevice:: getFileSystem()::addZipFileArchive()	เป็นคำสั่งเพิ่มไฟล์นั้นๆเข้าระบบไฟล์ของ Irrlicht เพื่อให้สามารถเรียกใช้ได้ภายหลัง
ISceneManager:: addCameraSceneNode()	เป็นคำสั่งกำหนดมุมกล้อง
ISceneManager:: createFlyStraightAnimator()	เป็นฟังก์ชันที่กำหนดให้โมเดลเคลื่อนไหวจากจุดหนึ่งไปยังอีกจุดหนึ่งเป็นเส้นตรง รวมทั้งกำหนดความเร็วในการเคลื่อนที่ด้วย
ISceneManager::createCollisionResponseAnimator()	เป็นฟังก์ชันที่ใช้ตรวจสอบ (Detect) วัตถุรอบข้างเกี่ยวกับการชน (Collision)
setMD2Animation ( )	สำหรับตั้งค่าท่าทางของตัวโมเดลที่โหลดเข้าไป
setFrameLoop( )	สำหรับกำหนดเฟรมที่ต้องการให้โมเดลแสดงโดยพารามิเตอร์จะกำหนดเฟรมเริ่มต้นและเฟรมสิ้นสุด
setLoopMode( )	เป็นฟังก์ชันเพื่อตั้งค่าการทำซ้ำโดยถ้าใส่ค่าพารามิเตอร์เป็น true จะให้ทำซ้ำ ถ้า false จะทำเพียงรอบเดียว
setScale( )	เป็นฟังก์ชันเพื่อกำหนดขนาดของโมเดล
setPosition( )	เป็นฟังก์ชันสำหรับกำหนดตำแหน่งของโมเดลที่ต้องการ
setRotation( )	เป็นฟังก์ชันสำหรับกำหนดมุมการหมุนของโมเดลโดยมีค่าพารามิเตอร์เป็นแกน x , y และ z
getFrameNr()	เป็นฟังก์ชันเพื่อเรียกค่าเฟรมที่โมเดลแสดงอยู่ออกมา
getTexture( )	เป็นฟังก์ชันเพื่อกำหนดภาพที่ต้องการแสดงที่เป็น ITexture

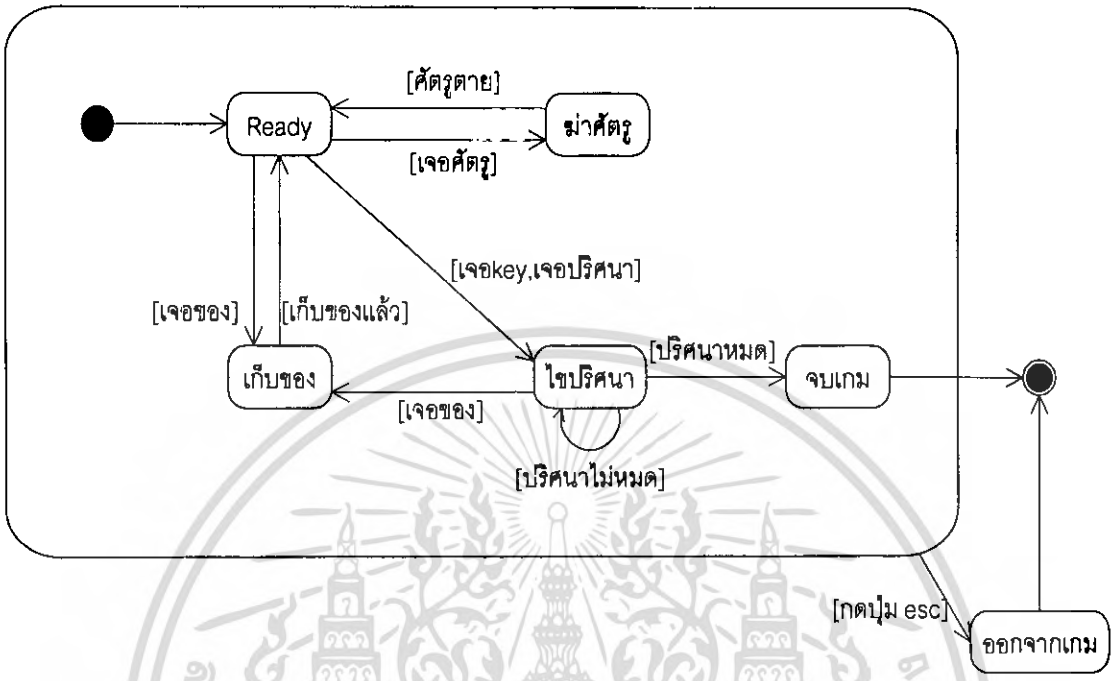
draw2DImage( )	เป็นฟังก์ชันสำหรับเรียกใช้ภาพ 2 มิติที่โหลดมาจาก getTexture( )
setAnimationSpeed( )	เป็นฟังก์ชันเพื่อกำหนดความเร็วของโมเดลที่ต้องการให้เล่น โดยใส่พารามิเตอร์เป็นค่าของความเร็ว
getMesh( )	เป็นฟังก์ชันเพื่อโหลด texture ของโมเดล 3 มิติ
setImage( )	เป็นฟังก์ชันโหลดภาพ 2 มิติเข้ามาและตั้งค่าภาพที่ต้องการ
setRelativePosition( )	เป็นฟังก์ชันตั้งค่าตำแหน่งที่ต้องการให้ภาพที่ตั้งค่าจาก setImage() โดยตั้งค่าในแกน x และ y

ตาราง 3.2 ฟังก์ชันการทำงานหลักของ Audiere Engine

AUDIERE ENGINE	
ชื่อฟังก์ชัน	**หน้าที่ การทำงาน**
AudioDevicePtr::AudioDevicePtr()	เป็นคำสั่งเริ่มการทำงานของ Audiere Engine
OutputStream::OpenSound()	เป็นคำสั่งระบุเสียงที่เราต้องการเปิด
OutputStream::setVolume()	เป็นฟังก์ชันเซตระดับเสียง
OutputStream::play()	เป็นฟังก์ชันให้เริ่มทำการเล่นเสียงที่ระบุไว้
OutputStream::setRepeat()	เป็นฟังก์ชันที่บอกว่าเสียงที่เล่นนั้น เมื่อเล่นจบให้เล่นซ้ำหรือไม่
OutputStream::stop()	เป็นฟังก์ชันเพื่อหยุดเล่นเสียง

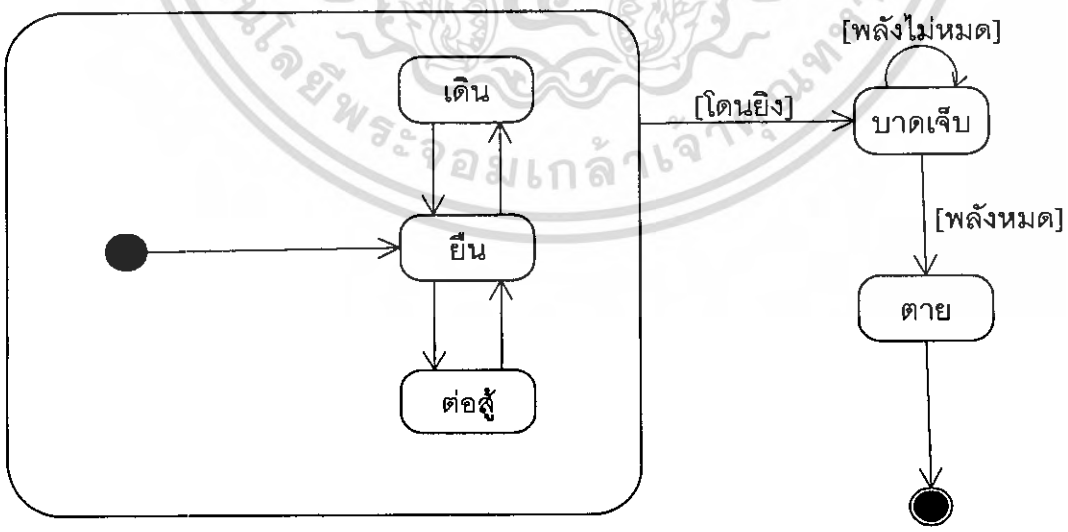
3.2.5 สถานะของเกม (State Diagram)

State Diagram ของเกมหลัก



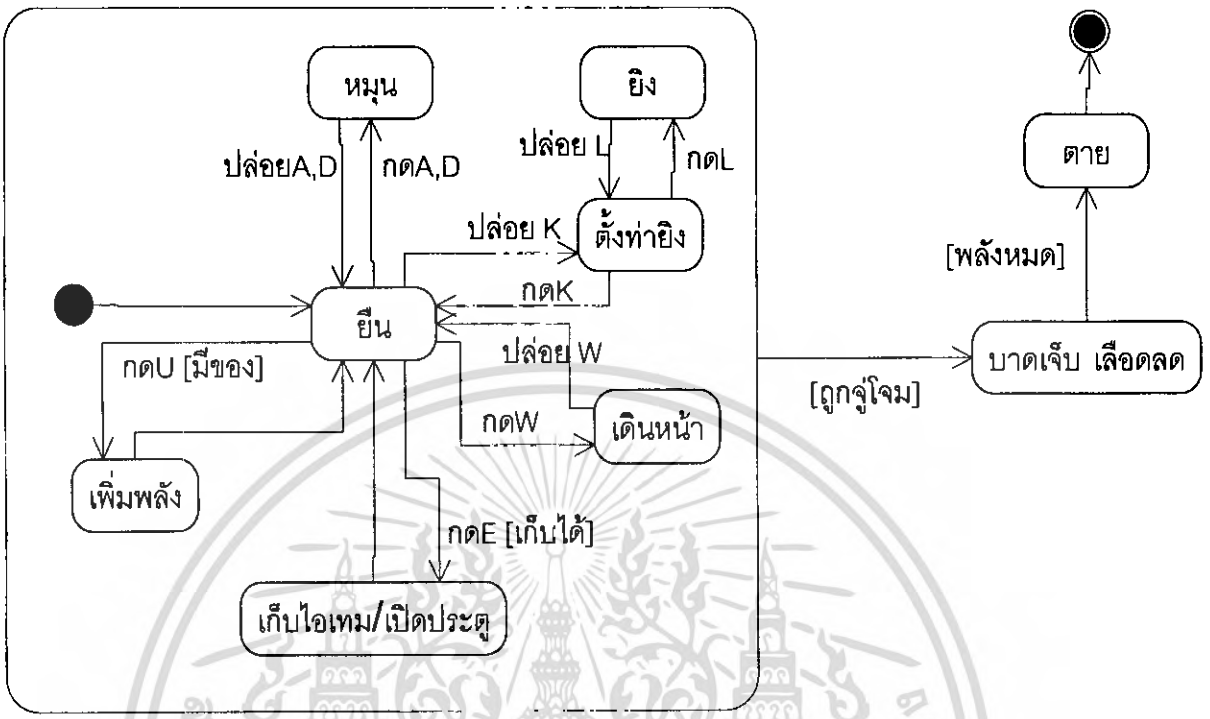
รูปที่ 3.20 แสดงState Diagram ของตัวเกม

State Diagram ของศัตรูภายในเกม



รูปที่ 3.21 แสดงState Diagram ของศัตรูในตัวเกม

State Diagram ของตัวละครเอกในเกม



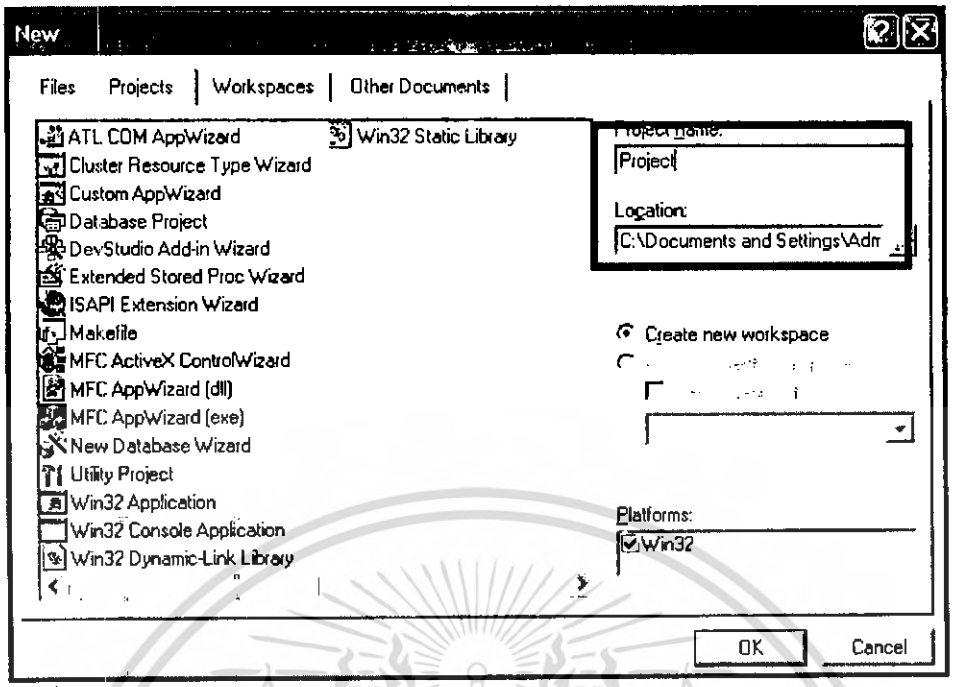
รูปที่ 3.22 แสดงState Diagram ของตัวละครเอกภายในเกม

3.2.6 การติดตั้งเอนจินและการสร้างโปรเจคใน Microsoft Visual C++

- **การติดตั้งเอนจินไอริชท์** จะต้องทำตามขั้นตอน ดังนี้
  1. ทำการดาวน์โหลดไฟล์ irrlitcht-1.1.zip (หรือเวอร์ชันล่าสุดที่มีให้ดาวน์โหลด) ได้ที่ http://irrlicht.sourceforge.net/downloads.html
  2. ขยายไฟล์ irrlitcht-1.1.zip ออกมาไว้ที่ไดเร็คทอรีที่ต้องการ โดยจะนำไปไว้ที่ไดเร็คทอรีไหนก็ได้
  3. นำไฟล์ Irrlitch.dll ที่อยู่ในไดเร็คทอรี ...\irrlicht-1.1\bin\Win32-gcc ไปไว้ในไดเร็คทอรี C:\WINDOWS\system32
  4. นำไฟล์ .h ทั้งหมดที่อยู่ในไดเร็คทอรี ...\irrlicht-1.1\include ไปไว้ในไดเร็คทอรี C:\Program Files \Microsoft Visual Studio\VC98\Include
  5. นำไฟล์ Irrlicht.lib ที่อยู่ในไดเร็คทอรี ... \irrlicht-1.1\lib\Win32-visualstudio ไปไว้ในไดเร็คทอรี C:\Program Files \Microsoft Visual Studio\VC98\Lib

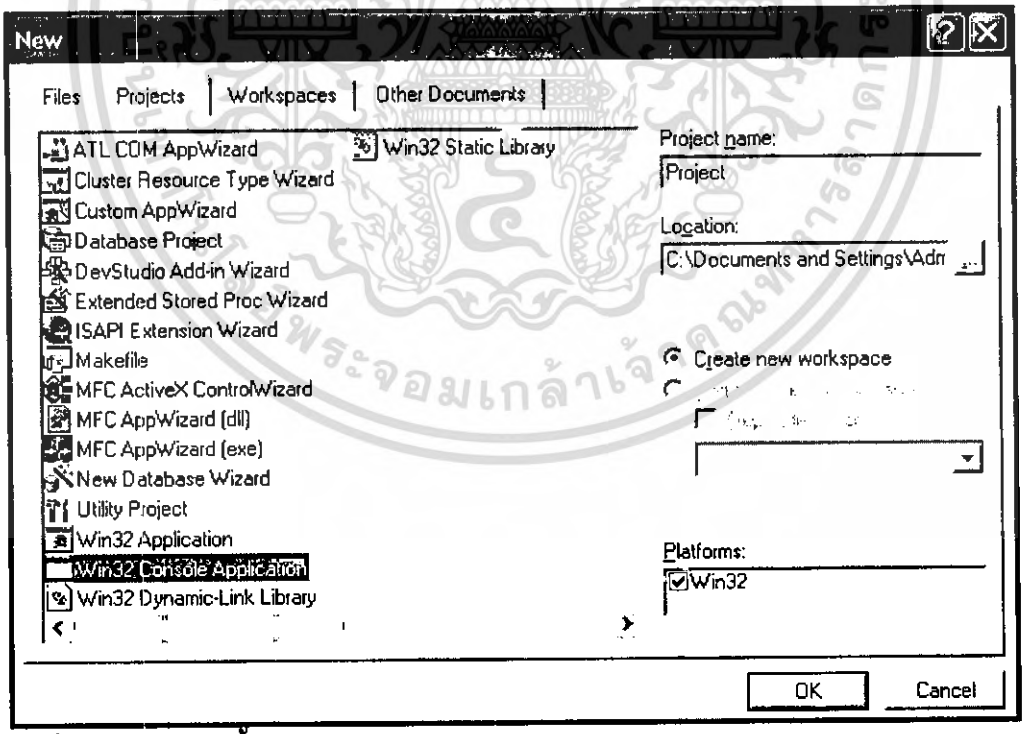
- **การติดตั้งเอนจินออร์เดียร์** จะต้องทำตามขั้นตอน ดังนี้
  1. ทำการดาวน์โหลดไฟล์ audiere-1.9.4-win32.zip (หรือเวอร์ชันล่าสุดที่มีให้ดาวน์โหลด) ได้ที่ http://audiere.sourceforge.net/download.php
  2. ขยายไฟล์ audiere-1.9.4-win32.zip ออกมาไว้ที่ไดเร็คทอรีที่ต้องการ โดยจะนำไปไว้ที่ไดเร็คทอรีไหนก็ได้
  3. นำไฟล์ audiere.dll ที่อยู่ในไดเร็คทอรี ...\audiere-1.9.4-win32\bin ไปไว้ในไดเร็คทอรี C:\WINDOWS\system32
  4. นำไฟล์ audiere.h ที่อยู่ในไดเร็คทอรี ...\audiere-1.9.4-win32\include ไปไว้ในไดเร็คทอรี C:\Program Files \Microsoft Visual Studio\VC98\Include
  5. นำไฟล์ audiere.lib ที่อยู่ในไดเร็คทอรี ...\audiere-1.9.4-win32\lib ไปไว้ในไดเร็คทอรี C:\Program Files\Microsoft Visual Studio\VC98\Lib

- **การสร้างโปรเจคใน Microsoft Visual C++** จะต้องทำตามขั้นตอน ดังนี้
  1. เปิดโปรแกรม Microsoft Visual C++6.0
  2. เลือกเมนู File>New
  3. ทำการตั้งชื่อโปรเจคที่ต้องการในกล่องข้อความ Project Name และเลือกไดเร็คทอรีที่ต้องการสร้างโปรเจคในกล่องข้อความ Location



รูปที่ 3.23 การสร้างโปรเจคใน Microsoft Visual C++

4. เลือกประเภทโปรเจคงานเป็น Win 32 Console Application

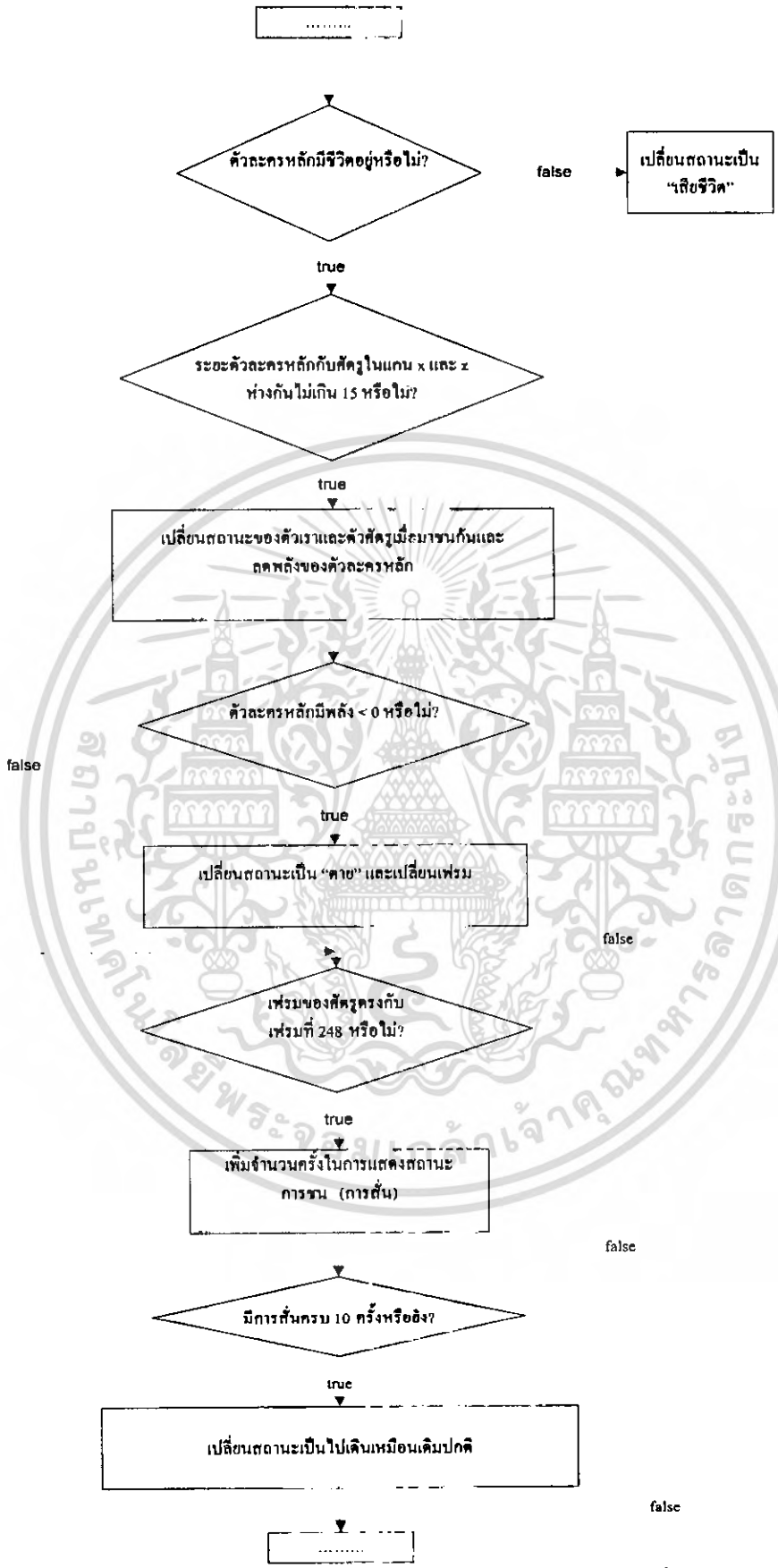


รูปที่ 3.24 การเลือกชิ้นงานโปรเจค

5. คลิกปุ่ม OK

3.2.7 การเขียนโปรแกรมเพื่อตรวจสอบการชนศัตรู

ในการตรวจสอบการชนศัตรูจะต้องทำการตรวจสอบว่าระยะห่างระหว่างตัวละครหลักกับศัตรูอยู่ในระยะที่กำหนดหรือไม่ ถ้าหากอยู่ในระยะห่างที่กำหนดจะมีการเปลี่ยนเฟรมของตัวละครหลักและตัวศัตรูให้กลายเป็นการยืนสั่นแทน และทำการลดพลังของศัตรู ซึ่งสามารถอธิบายด้วยแผนผังแสดงลำดับขั้นการดำเนินงาน (flowchart diagram) ดังนี้



รูปที่ 3.25 flowchart diagram แสดงการเขียนโปรแกรมเพื่อตรวจสอบการชนศัตรู

ในส่วนของการเขียนโปรแกรมเพื่อตรวจสอบการชนศัตรูเป็นดังนี้

```
if(en1.chkDead1 != 1)
{
    if((en1.getXEnB(1)-ac.getXAc()>-15&&en1.getXEnB(1)-ac.getXAc()<15)
&& (en1.getZEnB(1)-ac.getZAc()>-15&&en1.getZEnB(1)-ac.getZAc()<15))
    {
        if(en1.chkColEn == 0)
        {
            en1.shootEn(1); // shock
            en1.chkColEn = 1;
            controlActor.iniChkLoopEn();
            controlActor.iniChkBackLoop();

            if(ac.getHealth()>0) // decrease actor power
            {
                ac.setHealth(4);
            }

        }
        if(ac.getHealth()<=0) // Actor DEAD
        {
            if(ac.chkDead==0) // check for 1 loop
            {
                ac.deadAc();
                ac.chkDead=1;
                receiver.chkPress = 1;
            }
        }
    }
```

```
if(en1.getFrame() == 248)
{
        controlActor.setChkLoopEn();
}
if(controlActor.getChkLoopEn()   - 10)  // shock 10 loop
{
        if(controlActor.getChkBackLoop() == 0)
        {
                en1.shootBack(1); // back to walk
                controlActor.setChkBackLoop();
                en1.chkColEn = 0;
                controlActor.iniChkLoopEn(); // change count 10 loop to 0
        }
}
}
```

บรรทัดที่ 1 เป็นการตรวจสอบว่าตัวละครหลักยังมีชีวิตอยู่หรือไม่

บรรทัดที่ 3 เป็นการตรวจสอบว่าขณะนี้ตำแหน่งของตัวละครหลักกับตัวศัตรูตรงกันหรือไม่ โดยจะเห็นว่าจะนำตำแหน่งของตัวละครหลักไปลบกับตำแหน่งของศัตรู ถ้าต่างกันในแกน x และแกน z ไม่เกิน -15 และ 15 จะถือว่าตัวเราและตัวศัตรูชนกัน

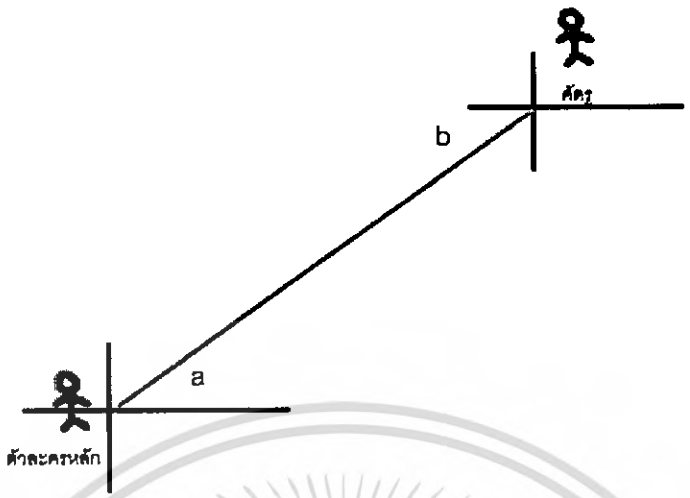
บรรทัดที่ 5-15 เป็นการเปลี่ยนสถานะของตัวละครหลักและศัตรูเมื่อมาชนกัน

บรรทัดที่ 16 เป็นการตรวจสอบว่าตอนนี้ตัวเรามีพลังอยู่หรือไม่ ถ้าไม่มีก็ถือว่าตาย

บรรทัดที่ 18-23 เป็นการเปลี่ยนสถานะของเราให้อยู่สถานะตาย

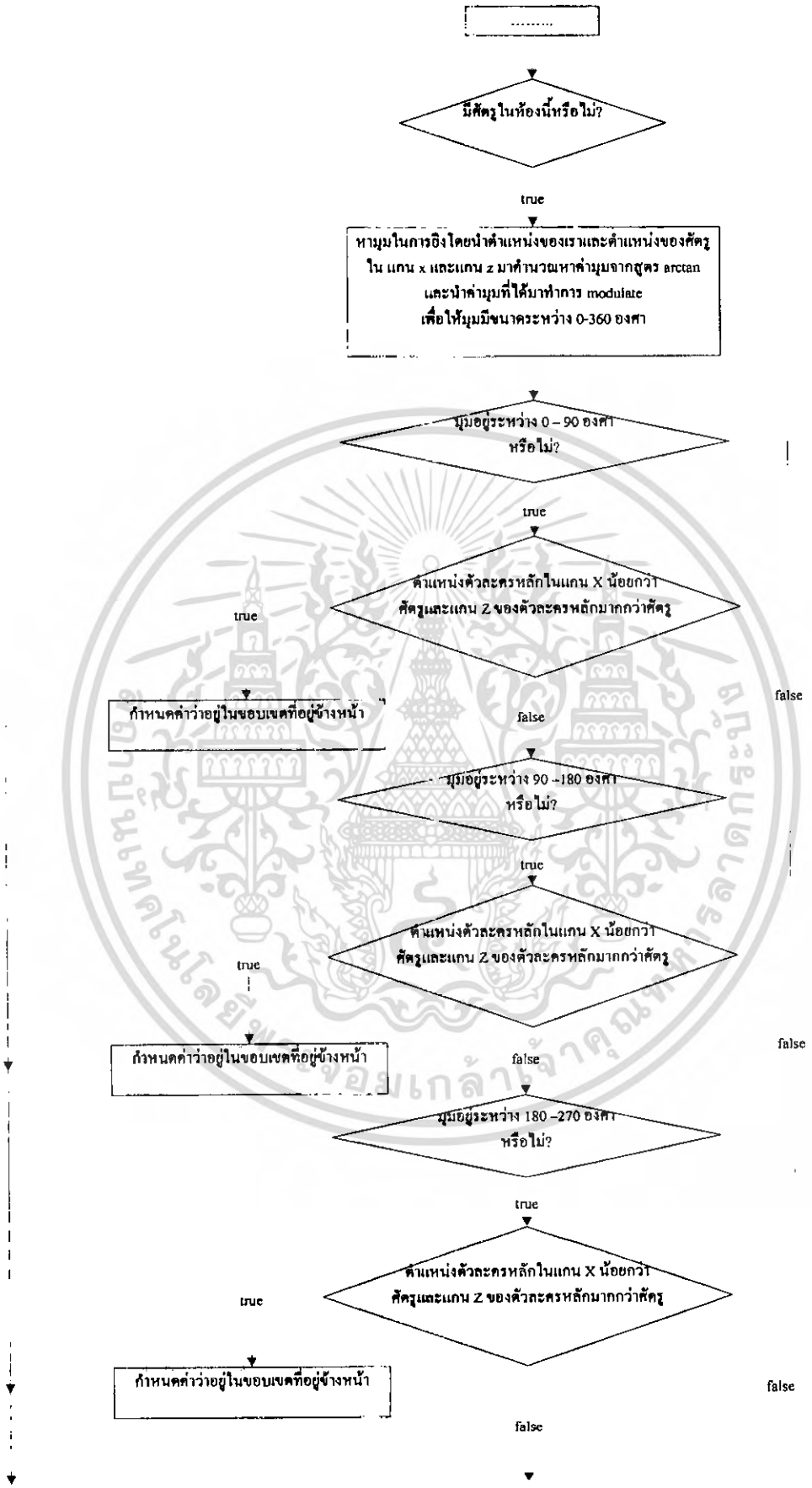
บรรทัดที่ 26-39 เป็นการตรวจสอบว่าตอนนี้ตัวศัตรูอยู่ที่เฟรมไหน ถ้ากำลังอยู่ในเฟรมที่เป็นสถานะของการชนก็จะข้ามไป แต่ถ้าเป็นสถานะเดินปกติก็จะให้เปลี่ยนสถานะเป็นสถานะของเฟรมการชนโดยจะให้ตัวศัตรูกระตุก 10 ครั้ง

3.2.8 การเขียนโปรแกรมเพื่อตรวจสอบการยิงศัตรู

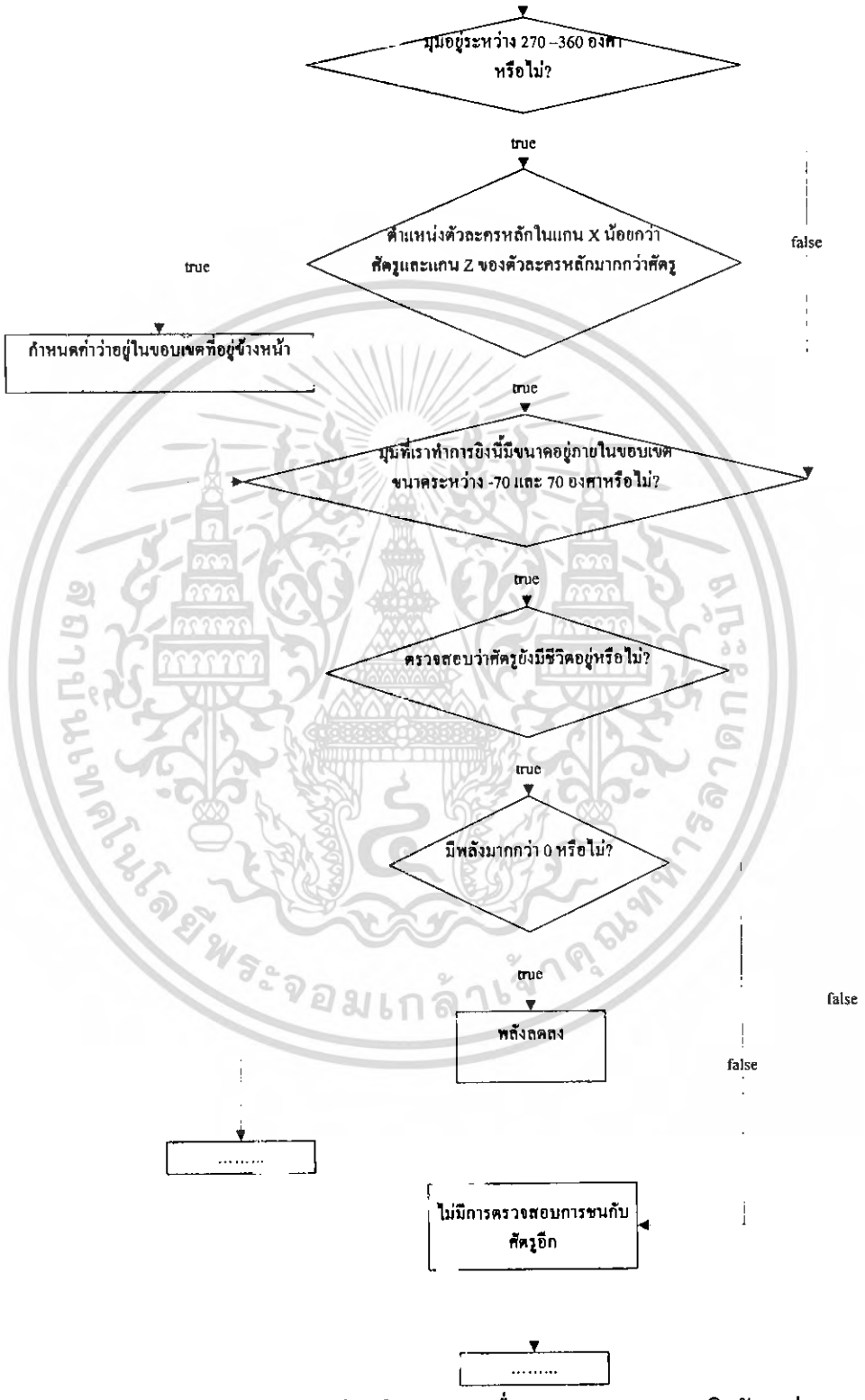


รูปที่ 3.26 การตรวจสอบการยิงศัตรู

ในการตรวจสอบการยิงศัตรูจะต้องทำการตรวจสอบว่าตัวละครเอกหันหน้าเข้าหาศัตรูหรือไม่ และทำการตรวจสอบโดยจะนำมุม a-b(ดังรูป 3.26) แล้วค่าที่ได้อยู่ในช่วงตั้งแต่ -70 องศาถึง 70 องศาหรือไม่ ถ้าหากตรงเงื่อนไขในเรื่องของมุมและการที่ศัตรูอยู่ทางด้านหน้าแล้วก็ถือว่ายิงโดนศัตรูและจะมีการเปลี่ยนเฟรมของศัตรูให้กลายเป็นการยืนสั่นแทน หากไม่ตรงตามเงื่อนไขทั้งสองเงื่อนไขก็จะถือว่ายิงไม่โดนศัตรู ซึ่งในส่วนของการเขียนโปรแกรมเป็นดังนี้



รูปที่ 3.27 flowchart diagram แสดงการเขียนโปรแกรมเพื่อการตรวจสอบการยิงศัตรู



รูปที่ 3.28 flowchart diagram แสดงการเขียนโปรแกรมเพื่อการตรวจสอบการยิงศัตรู(ต่อ)

ในส่วนของการเขียนโปรแกรมเพื่อตรวจสอบการยิงศัตรูเป็นดังนี้

```
if(levelEnemy==1)
{
    enAngle1 = atan2(ac.getXAc()-en1.getXEn(),ac.getZAc()-en1.getZEn());
    modAn = ((int)ac.getAngleAc())%360;
    if(modAn>=0 && modAn<=90 || modAn>=-360 && modAn<=-270)
    {
        if(ac.getXAc()<en1.getXEn()  &&  ac.getZAc()>en1.getZEn())
// check enemy back or front of actor // can shoot
        {   chkAn = 0;
            if(modAn<0)
            {
                modAn = modAn+360;
            }
        }
        else // can't shoot
        {
            chkLoop = 1;
        }
    }
    if(modAn>90 && modAn<=180 || modAn>-270 && modAn<=-180)
    {
        if(ac.getXAc()>en1.getXEn()  &&  ac.getZAc()>en1.getZEn())
// check enemy back or front of actor
        {   chkAn = 180;
            if(modAn<0)
            {   modAn = modAn+360;  }
        }
        else
        {   chkLoop = 1;  }
    }
```

```
if(modAn>180 && modAn<=270 || modAn>-180 && modAn<=-90)
{
        if(ac.getXAc()>en1.getXEn()    &&    ac.getZAc()<en1.getZEn())//
check enemy back or front of actor
        {       chkAn = 270;
                if(modAn<0)
                {       modAn = modAn+360;   }
        }
        else
        {       chkLoop = 1;    }
}
if(modAn>270 && modAn<360 || modAn>-90 && modAn<0)
{
        if(ac.getXAc()<en1.getXEn()    &&    ac.getZAc()<en1.getZEn())//
check enemy back or front of actor
        {       chkAn = 360;
                if(modAn<0)
                {       modAn = modAn+360;   }
        }
        else
        {       chkLoop = 1;    }
}
if((modAn-enAngle1-chkAn)<70 && (modAn-enAngle1-chkAn)>-70)
{
        if(chkLoopDead1==0) // can't shoot after dead
        {
                if(chkLoop == 0)
                {
                        if(en1.getHealth(1)>0)
                        {
```

```
                    en1.setHealth(1);
                    en1.shootEn(1); // change to shock
                    chkLoop = 1;
                    chkLoopEn = 0; // set shock 10 loop
                    chkBackLoop = 0;
                }
                else  // dead
                {
                    if(chkDead1==0) // change enemy to floor
                    {
                        en1.shootEn(2); // change to dead
                        chkDead1 = 1;
                        chkLoopDead1=1;
                        en1.chkDead1 = 1;
                    }
                }
            }
        }
        else
        {
            // dead already
        }
    }
}
```

บรรทัดที่ 3 เป็นการหามุมในการยิงโดยนำตำแหน่งของเราและตำแหน่งของศัตรูในแกน x และแกน z มาคำนวณหาค่ามุมจากสูตร arctan

บรรทัดที่ 4 เป็นการที่นำค่ามุมที่ได้จากบรรทัดที่ 3 มาทำการ modulate เพื่อให้มุมมีขนาดระหว่าง 0-360 องศา

บรรทัดที่ 5-48 เป็นการตรวจสอบว่ามุมที่หันอยู่ที่จะทำการยิงว่าศัตรูอยู่หน้าเรา หรืออยู่ด้านหลังของเราเพื่อป้องกันการที่มุมในการยิงตรงกับศัตรูแค่เราหันหลังให้กับศัตรูแล้วยิงโดน

บรรทัดที่ 49 เป็นการตรวจสอบว่ามุมที่เราทำการยิงนี้มีขนาดอยู่ภายในขอบเขตขนาดระหว่าง -70 และ 70 องศาหรือไม่

บรรทัดที่ 51 เป็นการตรวจสอบสถานะของศัตรูว่ายังมีชีวิตอยู่หรือไม่

บรรทัดที่ 55-62 เป็นการเปลี่ยนสถานะของศัตรูเป็นสถานะถูกยิงและพลังของศัตรูจะลดลง

บรรทัดที่ 63-72 ถ้าหากศัตรูตายแล้วก็จะเปลี่ยนสถานะของศัตรูให้เป็นการตาย และจะไม่มีการตรวจสอบการชนกับตัวศัตรูอีก

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

การทดลองและการวิเคราะห์ปัญหาที่จะกล่าวถึงในบทนี้ จะเป็นขั้นตอนของการทดสอบและผลที่ได้จากการทดสอบในแต่ละส่วนของเกมทั้งหมด โดยผลการทดสอบที่ได้นี้ จะถูกนำไปวิเคราะห์ถึงปัญหาและแนวทางในการแก้ปัญหาเพื่อการพัฒนาต่อไปในอนาคต โดยทางผู้จัดทำหวังเป็นอย่างยิ่งว่าผลการทดลองและแนวทางในการแก้ปัญหาที่ได้จากการทดสอบในครั้งนี้ จะเป็นประโยชน์และแนวทางต่อผู้ที่มีความสนใจจะศึกษาและค้นคว้าเพื่อจะนำไปศึกษาต่อ ให้ได้เห็นถึงปัญหา และข้อดีข้อเสียของปัญหาโดยผู้ศึกษาไม่จำเป็นต้องทำการทดสอบเพื่อหาปัญหาด้วยตนเองอีกครั้ง

คุณสมบัติของระบบที่จะนำมาทดสอบ

1. ระบบปฏิบัติการ Microsoft Windows XP
2. หน่วยประมวลผลกลางที่มีความเร็ว 1.6 GHz
4. หน่วยความจำหลักขนาด 1024 MB
5. VGA Card ของ NVIDIA Geforce GO 7300 หน่วยความจำบนการ์ดขนาด 256 MB
6. ระบบเสียง Sound Card ของ Creative Vibra 128

ขั้นตอนการดำเนินการทดสอบ

- ทำการติดตั้งตัวโปรแกรมและซอฟท์แวร์ที่จำเป็นต้องใช้
- ทำการรันและประมวลผลภายใต้ระบบที่กำหนด
- ตรวจสอบการใช้คำสั่งของปุ่มกดต่าง ๆ และการทำงานของปุ่มกดต่าง ๆ
- ตรวจสอบการทำงานของโปรแกรมโดยการใช้คีย์บอร์ด
- ตรวจสอบการหา Error

จุดประสงค์ของการดำเนินการทดสอบ

- หลังจากติดตั้งตัวโปรแกรมแล้วสามารถใช้โปรแกรมได้
- การประมวลผลของโปรแกรมจะต้องทำความเร็วในระดับที่ยอมรับได้
- การใช้คำสั่งของปุ่มกดต่างๆเป็นไปตามรูปแบบ
- การกดแป้นของคีย์บอร์ดถูกต้องตามเป้าหมาย
- จำนวน Error ที่เกิดขึ้นต้องไม่มี หรือน้อยที่สุด

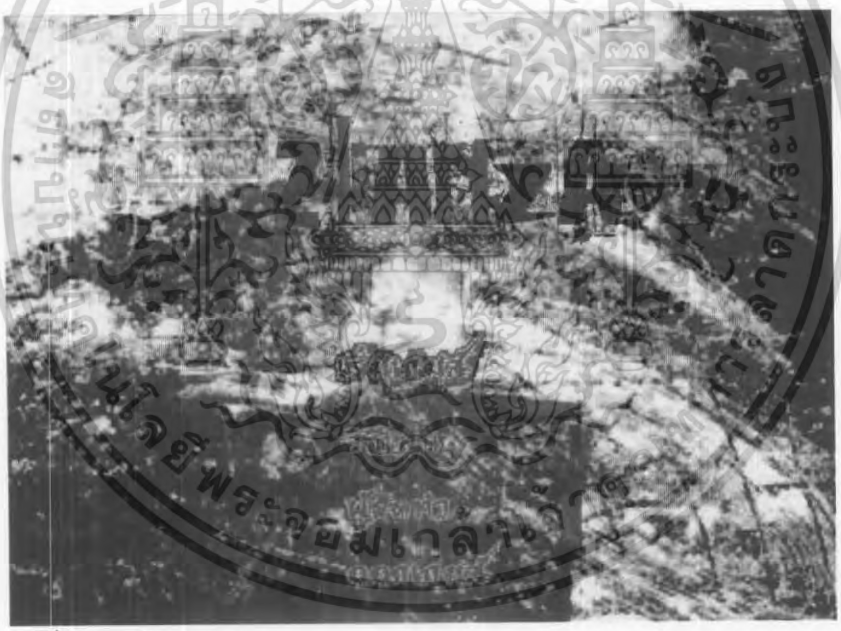
4.1 ขั้นตอนการดำเนินการทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็นต้องใช้

ตาราง 4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็น

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
การติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็น	ทำการรันโปรแกรมโดยดับเบิลคลิกที่ไฟล์ "Setup.exe" ในแผ่นซีดี	1.สามารถติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็นได้สมบูรณ์ 2. ผู้ใช้สามารถรันโปรแกรมได้

4.2 ขั้นตอนการดำเนินการทดสอบการประมวลผลภายใต้ระบบที่กำหนด

4.2.1 รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนด โดยดับเบิลคลิกที่ไฟล์ "Ztanza.EXE"



รูปที่ 4.1 หน้าจอหลักของเกม

ตาราง 4.2 ขั้นตอนการรันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนดและการทดสอบสถานะเมนู

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
รันโปรแกรมภายใต้ระบบเครื่องคอมพิวเตอร์ที่กำหนด	ทำการรันโปรแกรมโดยดับเบิลคลิกที่ไฟล์ "Ztanza.exe" ในโฟลเดอร์ปลายทางที่ได้ทำการติดตั้ง	โปรแกรมสามารถทำงานได้โดยปรากฏหน้าจอหลักของเกม

ทดสอบลำดับของการเปลี่ยนสถานะ	เลือกเมนูต่างๆ เพื่อเข้าสู่หน้าจอส่วนต่างๆ ของเกม	ลำดับของการเปลี่ยนสถานะหน้าจอถูกต้องตามที่ได้ออกแบบไว้

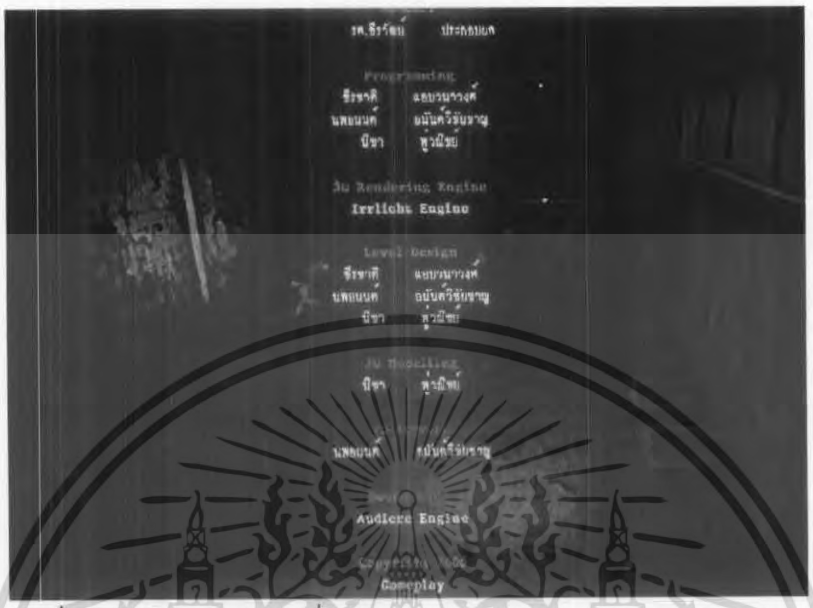
4.2.2 เมื่อเลือกที่เมนู "เริ่มเกมส์" จะเข้าสู่ภายในตัวเกม

รูปที่ 4.2 หน้าจอเริ่มต้นภายในตัวเกม

4.2.3 เมื่อเลือกที่เมนู "วิธีเล่น" จะปรากฏหน้าจอแสดงวิธีการบังคับและการใช้คีย์ต่างๆ

รูปที่ 4.3 หน้าจอแสดงวิธีการบังคับและการใช้คีย์ต่างๆ

4.2.4 เมื่อเลือกที่เมนู "ผู้จัดทำ" จะปรากฏหน้าจอแสดงรายชื่อผู้จัดทำ



รูปที่ 4.4 หน้าจอแสดงรายชื่อผู้จัดทำ

4.2.5 เมื่อเลือกที่เมนู "ออกจากเกมส์" จะเป็นการออกจากหน้าเมนูและออกจากเกม

4.3 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของเกม

ตารางที่ 4.3 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของเกมในส่วนของการเลือกเมนู

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
ทดสอบการเลือกเมนูต่างๆของหน้าจอ	กดปุ่มขึ้นหรือลงเพื่อเลือกเมนูเริ่มเกมส์, วิธีเล่น, ผู้จัดทำ, ออกเกมส์	เมนูต่างๆจะเปลี่ยนสีไปเมื่อกดปุ่มขึ้นหรือลง
ทดสอบการเลือกเมนูต่าง ๆ ของหน้าจอ	เลือกเมนูเริ่มเกมส์, วิธีเล่น, ผู้จัดทำ, ออกเกมส์	เมื่อเลือกเมนูแล้วหน้าจอจะถูกเปลี่ยนเป็นหน้าจอของแต่ละเมนู

ตารางที่ 4.4 ขั้นตอนการทดสอบการควบคุมตัวละคร

การทดสอบ	วิธีการทดสอบ	ผลการทดสอบ
การเดินหน้า	กดปุ่ม W	เกมแสดงผลของการเดินหน้า
การหมุนซ้าย	กดปุ่ม A	เกมแสดงผลของการหมุนซ้าย

การหมุนขวา	กดปุ่ม D	เกมแสดงผลของการหมุนขวา
การตั้งท่ายิง	กดปุ่ม K	เกมแสดงผลของการตั้งท่ายิง
การยิง	กดปุ่ม L	เกมแสดงผลของการยิงปืน
การเก็บไอเท็มหรือเปิดประตู	กดปุ่ม E	ไอเท็มที่เก็บหายไป(เก็บไอเท็ม)หรือเปลี่ยนห้อง(เปิดประตู)

ตารางที่ 4.5 ขั้นตอนการทดสอบตัวละครถูกโจมตี

การทดสอบ	วิธีการทดสอบ	ผลการทดสอบ
ตัวละครถูกโจมตี	ยิงปืนใส่ศัตรูหรือเดินเข้าสู่รัศมีของศัตรู	ตัวละครที่ถูกโจมตีจะเกิดอาการสั่นอยู่พักหนึ่ง

ตางราง 4.6 ขั้นตอนการทดสอบการตาย

การทดสอบ	วิธีการทดสอบ	ผลการทดสอบ
ตัวละครเอกตาย	ถูกศัตรูโจมตีจนมีพลังชีวิตน้อยกว่าหรือเท่ากับศูนย์	จะปรากฏหน้าจอจบเกม ซึ่งมีข้อความว่า "GAME OVER"
ศัตรูตาย	ยิงปืนใส่ศัตรู	ศัตรูลอยกระเด็นไปข้างหลังและตกลงสู่พื้น

รูปที่ 4.5 หน้าจอจบเกมเมื่อตัวละครเอกตาย

ตารางราง 4.7 ขั้นตอนการทดสอบการจบเกม

การทดสอบ	วิธีการทดสอบ	ผลการทดสอบ
เล่นจบเกม	ทำการฆ่าหัวหน้าใหญ่	จะปรากฏหน้าจอจบเกม ซึ่งมีข้อความว่า "YOU WIN"

รูปที่ 4.6 หน้าจอจบเกมเมื่อชนะเกม

4.4 ปัญหาข้อผิดพลาดและข้อเสนอแนะ

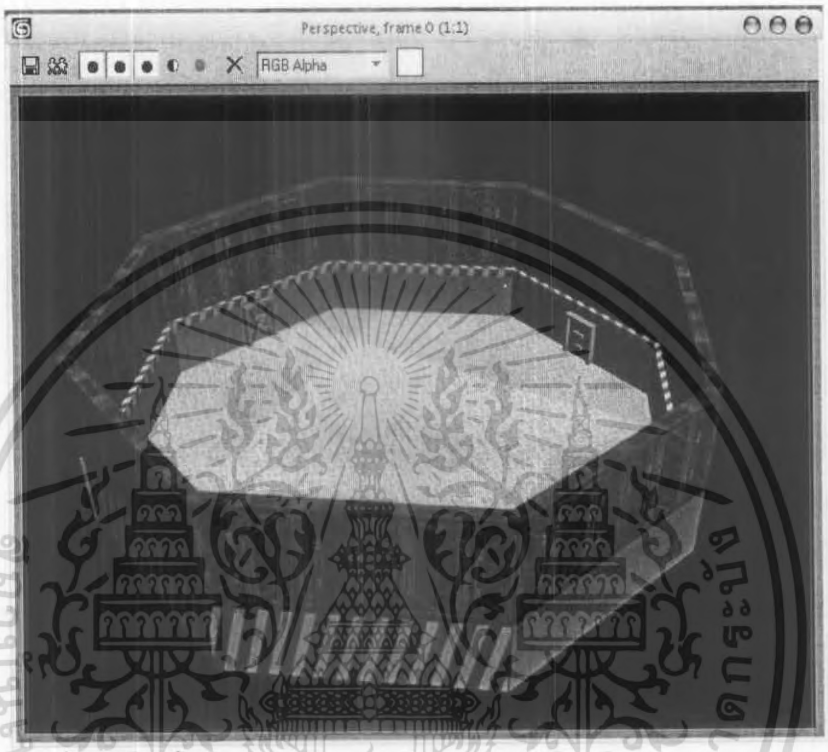
ในขั้นตอนของการทดสอบเพื่อหาข้อผิดพลาดจะไม่มีรูปแบบที่ตายตัวชัดเจน ที่จะทดสอบให้ได้ข้อผิดพลาดที่เกิดขึ้น โดยวิธีการทดสอบที่ดีที่สุดคือการทดสอบให้ครอบคลุมทุกส่วนของโปรแกรม และทดสอบหลายๆ ครั้งให้ได้ผลการทดสอบที่แน่นอน โดยจากการทดสอบทั้งหมดโปรแกรมทำให้เราสามารถหาข้อผิดพลาดได้ดังนี้

4.4.1 ปัญหาการยิงศัตรูที่ยืนซ้อนกันแล้วโดนยิงพร้อมกัน

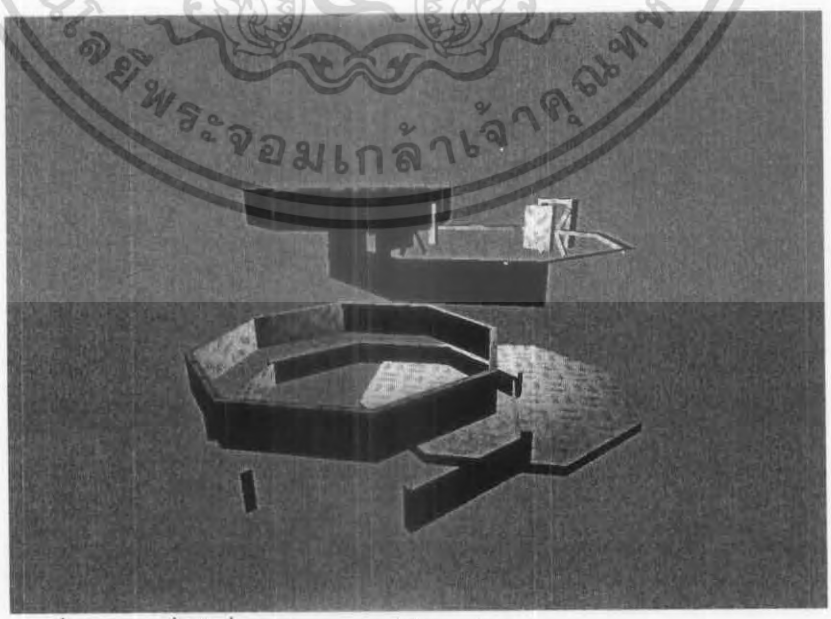
ปัญหานี้จะเกิดขึ้นเมื่อผู้เล่นทำการยิงใส่ศัตรูที่ยืนอยู่ในแนวยิงของปืนซ้อนกันตั้งแต่สองตัวขึ้นไป ซึ่งจะส่งผลให้ศัตรูโดนยิงพร้อม ๆ กัน ซึ่งสาเหตุเกิดจากการที่ผู้ออกแบบได้เช็คการยิงปืนใส่ศัตรูโดยการใช้มุมที่ตัวละครหันหน้าไปนั่นเอง

4.4.2 ปัญหาโมเดลที่ทำการเปลี่ยนฟอร์แมตไม่เหมือนกับต้นฉบับ

ในบางครั้งที่มีการแปลงฟอร์แมตจาก3d max file มาเป็น .X file ในการแสดงผลโดยตัวเกมซึ่งนำ .X file มาใช้งานนั้นผลที่ได้มีความแตกต่างจากต้นฉบับ เช่นในการ Export ไฟล์โมเดลมาใช้งานในตัวเกม ดังรูป



รูปที่4.7 ภาพที่ได้จากการ Render ด้วย 3D MAX



รูปที่4.8 ภาพที่ได้เมื่อนำโมเดลมาใช้งานจริง

4.4.3 ปัญหาการติดกันของตัวละครเอกกับวัตถุภายในห้อง

เนื่องจากการเช็คชนกันระหว่างตัวละครเอกนั้นตัวเอนจินจะทำการสร้างทรงกระบอกขึ้นมาครอบตัวละครกับตัววัตถุภายในห้อง ซึ่งปัญหานี้เกิดจากเมื่อตอนเราสร้างห้องขึ้นมาแล้วสร้างตัวละครเอกขึ้นมาซึ่งตอนสร้างตำแหน่งของตัวละครเอกกับวัตถุใกล้กันมากจนกระทั่งทรงกระบอกเกิดซ้อนทับกัน จนทำให้ตัวละครไม่สามารถเคลื่อนที่ออกมาได้

4.4.4 ปัญหาของการมีข้อความแจ้งเตือน

ปัญหานี้เกิดจากเมื่อเราทำการออกจากเกมอย่างกระทันหันโดยกดปุ่ม Alt+F4 ซึ่งจะทำให้เกิดข้อความแจ้งเตือนขึ้นมาจาก Windows ซึ่งปัญหาน่าจะเกิดจากการคืนเนื้อที่หน่วยความจำ ซึ่งแนวทางการแก้ไขอาจจะต้องมีการตรวจสอบการคืนเนื้อที่หน่วยความจำในส่วนการทำงานทั้งหมด

4.5 ประเมินประสิทธิภาพของเกม

จากการทดสอบที่ผ่านมาทั้งหมดทำให้เราสามารถประเมินประสิทธิภาพโดยรวมของเกมได้ว่า เกมนี้สามารถทำงานได้ตามที่ออกแบบไว้สมบูรณ์พอสมควร โดยมีข้อผิดพลาดเล็กน้อยที่เกิดขึ้น โดยการทดสอบได้จากระบบที่กำหนดไว้ข้างต้นซึ่งอาจจะสามารถเปลี่ยนแปลง เพิ่มหรือลดประสิทธิภาพได้ ถ้าหากมีการเปลี่ยนระบบที่ใช้ในการประมวลผลของเกม และเนื่องจากเกมนี้มีการติดต่อกับเสียงและภาพที่ค่อนข้างมากทำให้ระบบที่จะนำเกมนี้ไปประมวลผลต้องมีประสิทธิภาพสูงพอสมควร

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

5.1.1 การศึกษาและรวบรวมข้อมูล

การศึกษาและรวบรวมข้อมูลที่จำเป็นต้องใช้ในการพัฒนาเกมทำได้โดยการศึกษาและรวบรวมข้อมูลความเป็นไปได้ที่จะพัฒนาเกมนี้บนเครื่องคอมพิวเตอร์ได้หรือไม่ เมื่อทำการวิเคราะห์และและตัดสินใจได้แล้วก็ทำการรวบรวมข้อมูล เพื่อสร้างกฎและกติกาที่จะนำมาเป็นแนวทางในการพัฒนาเกม พร้อมทั้งทำการศึกษาขั้นตอนวิธีการต่างๆ ในการพัฒนาเกม โดยอาจศึกษาจากเนื้อหาวิชา หรือคู่มือวิธีใช้ของเครื่องมือที่จำเป็นต้องใช้ หรืออาจจะศึกษาจากตัวเกมต่างๆ ที่มีอยู่ จากนั้นรวบรวมความรู้ทั้งหมดเพื่อนำไปใช้เป็นโครงสร้างของเกม เพื่อง่ายต่อการออกแบบหน้าจอส่วนติดต่อผู้ใช้ ออกแบบโครงสร้างและวิธีการในการพัฒนา และการเลือกใช้เครื่องมือต่างๆ ที่เหมาะสมและสนับสนุนในการพัฒนา

5.1.2 การวิเคราะห์และการออกแบบเกม

การออกแบบเกมต้องทำให้เกมมีความสวยงามดึงดูดผู้เล่น มีกติกาที่ชัดเจน สามารถใช้งานได้ง่าย โดยการวิเคราะห์และออกแบบ จะนำข้อมูลต่างๆ ที่รวบรวมไว้มาประกอบกับความรู้ความสามารถที่มีอยู่ รวมไปถึงความรู้ที่ได้มาจากการศึกษาค้นคว้าเพิ่มเติม จากนั้นทำการออบแบบกฎและกติกา วางโครงสร้างและออกแบบหน้าจอติดต่อผู้ใช้ รวมถึงขั้นตอนวิธีต่างๆ ในการพัฒนาเกมให้เสร็จตามระยะเวลาที่กำหนด แล้วจึงทำการแบ่งหน้าที่ และมอบหมายส่วนงานต่างๆ ให้กับสมาชิก

5.1.3 การสร้างตัวละคร ภาพ และเสียงต่างๆ

ขั้นตอนนี้สามารถทำควบคู่ไปได้กับการเขียนโปรแกรม โดยเกมสแตนซ่านี้ ได้ใช้เครื่องมือหลายอย่างที่ช่วยในการออกแบบและสร้างตัวละคร ภาพ และเสียงต่างๆ รวมทั้งการสร้างภาพเคลื่อนไหวภายในเกม โดยเครื่องมือที่ใช้ในการออกแบบและสร้างงานกราฟิก ได้แก่ โปรแกรมตกแต่งรูปภาพ Adobe Photoshop ส่วนเครื่องมือที่ใช้ในการออกแบบและสร้างงานด้านเสียงเอฟเฟกต์และเสียงประกอบ ได้แก่ โปรแกรม Sound Recorder

5.1.4 การพัฒนาโปรแกรม

ขั้นตอนนี้จะเป็นขั้นตอนที่สำคัญและใช้ระยะเวลานานที่สุดในการพัฒนาเกม เพราะว่าขั้นดอนนี้จะเป็นการนำความรู้ที่เรารวบรวมได้ทั้งหมด มารวมกับโครงสร้าง รูปแบบ และกติกาของเกมที่เราจะพัฒนา รวมไปถึงภาพ และเสียงที่เราได้จัดเตรียมไว้ และเมื่อเริ่มต้นพัฒนาก็ต้องมีการค้นคว้าความรู้ใหม่ๆ มากมายนอกเหนือจากที่ค้นคว้าไว้ในตอนแรก มีการวิจัยและพัฒนาอัลกอริทึมที่ใช้ในการควบคุมกระบวนการของเกมให้เป็นไปตามกฎและกติกาที่วางเอาไว้ และทำการทดสอบการทำงานของเกมที่พัฒนาไปเรื่อยๆ จึงต้องใช้ระยะเวลาในขั้นดอนนี้มากพอสมควร ซึ่งขั้นตอนการทำงานในส่วนนี้มีซอฟแวร์ และเครื่องมือที่ใช้ในการพัฒนาดังนี้

- Microsoft Visual C++
- Irrlicht Game Engine
- Audiere Sound Engine

5.2 ข้อจำกัดของโปรแกรม

เกมสแตนซ่า เป็นเกมที่พัฒนาบนพื้นฐานโครงสร้างของ Microsoft Windows และทำงานที่ความละเอียด 1024 x 768 พิกเซลซ์ โดยที่เกมนี้ทำงานร่วมกับภาพ และเสียงค่อนข้างมาก รวมไปถึงการตรวจเงื่อนไขภายในเกมโดยขึ้นกับหน่วยประมวลผลกลาง เช่นการตรวจสอบเงื่อนไขบางเงื่อนไขภายในเกมโดยขึ้นอยู่กับหน่วยประมวลกลาง ทำให้เกมนี้ต้องทำงานอยู่บนระบบคอมพิวเตอร์ที่มีประสิทธิภาพสูงพอสมควร โดยรายละเอียดของข้อจำกัดของโปรแกรมมีดังต่อไปนี้

1. ระบบเครื่องคอมพิวเตอร์ที่ใช้รันเกมต้องมีประสิทธิภาพที่สูง เนื่องจากเกมจำเป็นต้องมีการประมวลผลของภาพกราฟิก และเสียง ค่อนข้างมาก
2. เกมสแตนซ่า ต้องการ DirectX เวอร์ชัน 9 ขึ้นไป
3. ห้ามมีการเคลื่อนย้ายโฟลเตอร์ต่าง ๆ จากตำแหน่งที่ติดตั้งเกม เพราะว่าเกมมีการเรียกใช้ข้อมูลที่อยู่ในโฟลเดอร์เหล่านั้น โดยหากเกมหาข้อมูลที่จำเป็นต้องใช้ในการทำงานไม่เจอก็จะเกิดความผิดพลาดได้
4. เกมสแตนซ่า ต้องการการ์ดจอ ที่สนับสนุนการประมวลผลด้าน 3 มิติ

5.3 ข้อเสนอแนะ

เนื่องจากเกมสแตนซ่า เป็นเกมที่มีกฎและกติกาค่อนข้างซับซ้อน และปลีกย่อยมากมาย จึงทำให้ในการพัฒนาจำเป็นต้องมีอัลกอริทึมในการควบคุมกระบวนการของเกมที่ดี และต้องมีการพัฒนาโปรแกรมที่มีโครงสร้างซับซ้อนมากขึ้น ด้วยเหตุนี้ทำให้การออกแบบอัลกอริทึมของเกมต้องทำด้วยความรอบคอบ และต้องคอยตรวจสอบผลการทำงานของเกมไปทุกระยะ เพื่อตรวจหาข้อผิดพลาดที่อาจเกิดขึ้น นอกจากนี้ยังสามารถนำเกมนี้ไปพัฒนาโดยการสร้างกฎและกติกาใหม่ๆ หรือความพิเศษใหม่ๆ เพื่อให้ดูมีเหมาะสมมากขึ้นได้ตามต้องการ

บรรณานุกรม

[1] นิรุธ อำนวยศิลป์ , สร้างโปรแกรมบน Windows ด้วย Visual C++ Version 6.0 , บริษัท ซัคเซส มีเดีย จำกัด

[2] วิวัฒน์ อุดมปีติทรัพย์ , 3ds max Reference Autodesk VIZ 2005 ,บริษัท เอส.พี.ซี บุ๊คส์ จำกัด

[3] Nikolaus Gebhardt. 2002. **Irrlicht Graphic Engine**. [Online].
Available : http://irrlicht.sourceforge.net

ภาคผนวก

คู่มือการติดตั้ง

คู่มือการเล่น

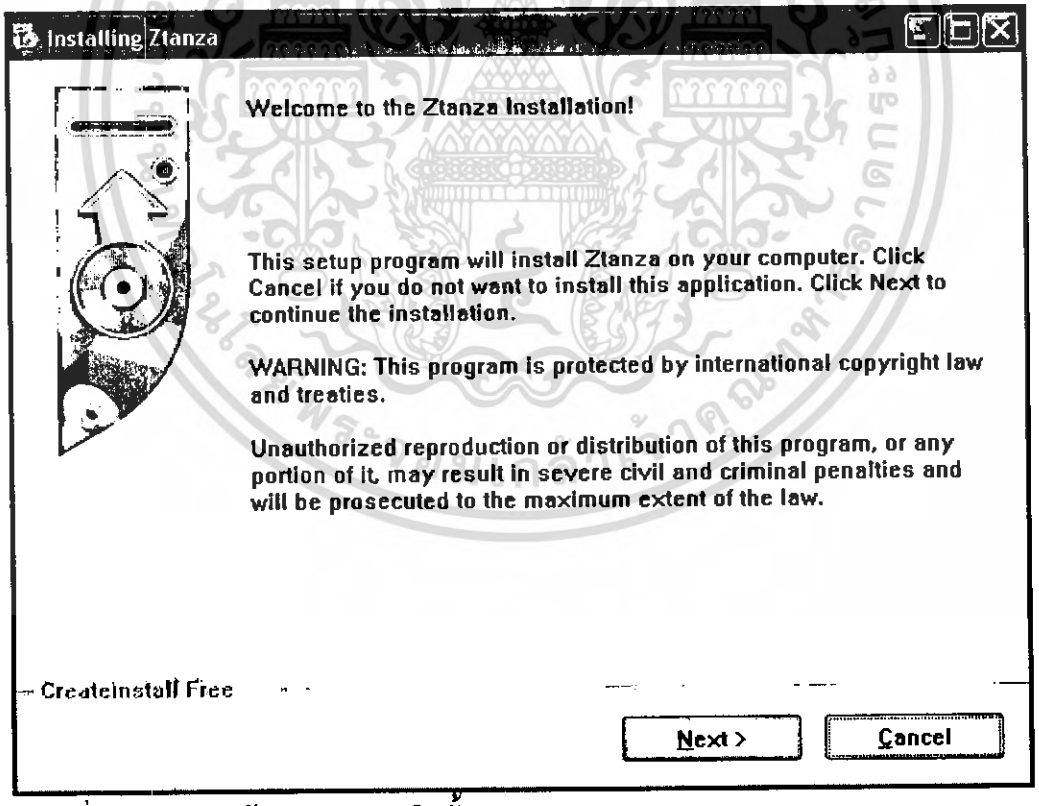
ภาคผนวก ก
คู่มือการติดตั้ง

คุณสมบัติขั้นต่ำของเครื่องคอมพิวเตอร์ที่ต้องการ

1. ระบบปฏิบัติการ Microsoft Windows XP Professional
2. หน่วยประมวล CPU 1.6 GHz Duo core
3. หน่วยความจำหลักขนาด 1,024 MB
4. VGA Card หน่วยความจำบนการ์ดขนาด 256 MB
5. หน่วยความจำสำรองต้องมีพื้นที่ว่างเหลือไม่ต่ำกว่า 170 MB
6. ต้องสนับสนุน OpenGL หรือ DirectX 9.0 ขึ้นไป

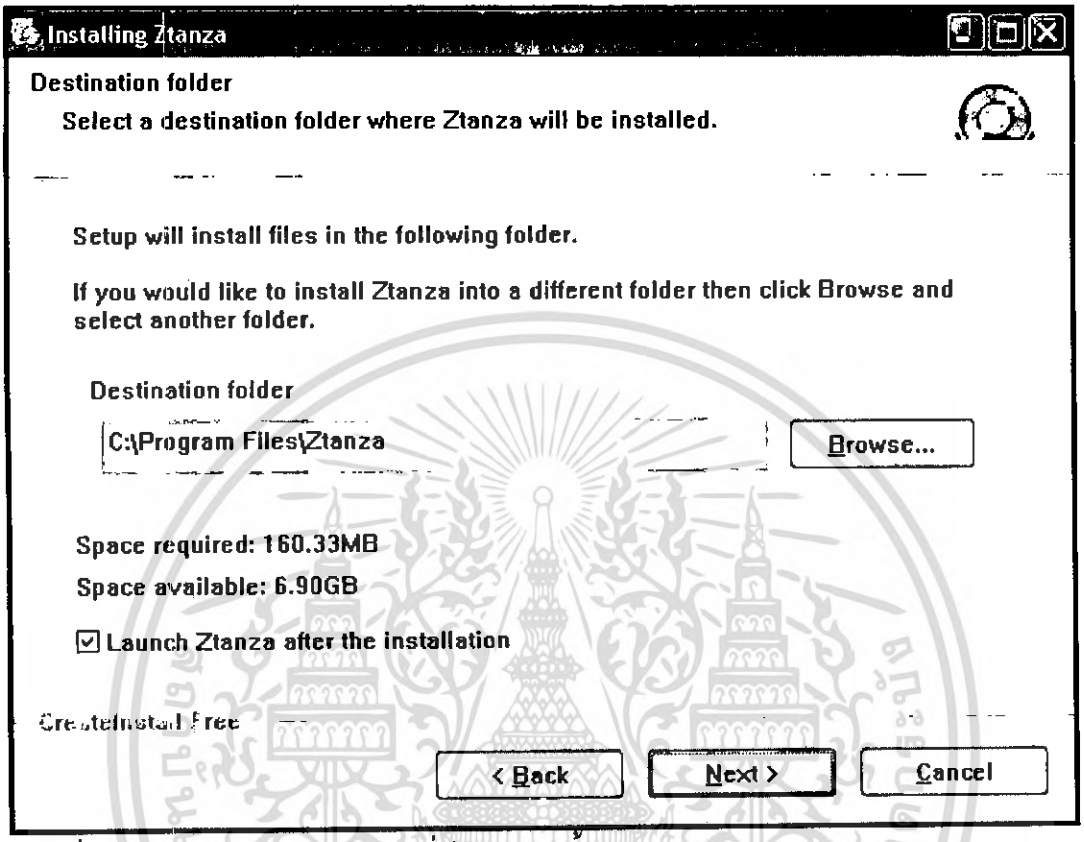
วิธีการติดตั้ง Ztanza มีดังต่อไปนี้

1. รันไฟล์ Setup.exe จะปรากฏหน้าจอดังภาพ



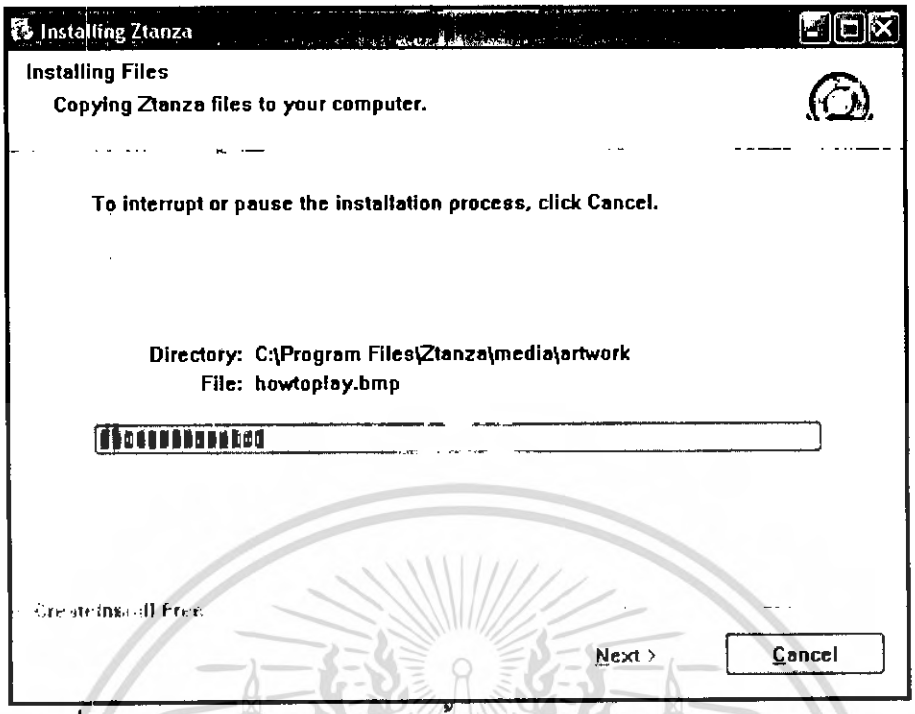
รูปที่ ก.1 แสดงหน้าแรกของการติดตั้ง

2. เมื่อป้อนข้อมูลเสร็จแล้วกด Next จะเป็นขั้นตอนการเลือก Path ที่ต้องการลงโปรแกรม โดยเนื้อที่ของโปรแกรมที่ใช้ คือ 160.33 MB



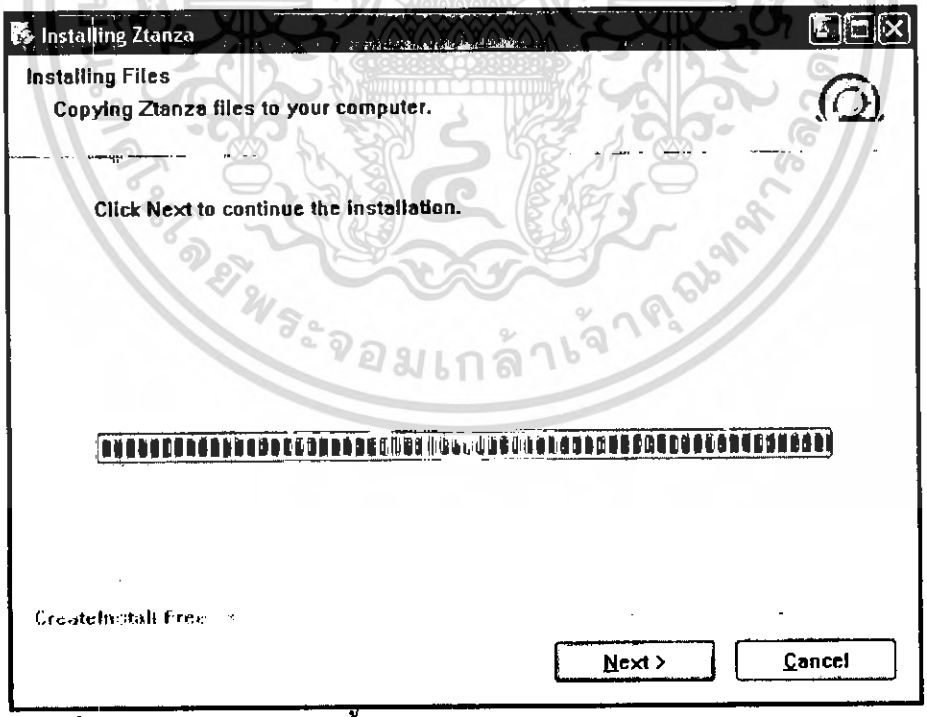
รูปที่ ก.2 แสดงหน้าการระบุ Path ที่ต้องการติดตั้ง

3. จากนั้นกดปุ่ม Next จะเป็นการติดตั้งโปรแกรมลงเครื่อง จะปรากฏแถบวิ่ง ดังรูป



รูปที่ ก.3 แสดงขณะกำลังทำการติดตั้ง

4. ขั้นตอนต่อไปเมื่อติดตั้งเสร็จให้จะปรากฏปุ่ม Next ขึ้นมา ให้ทำการกดปุ่ม Next



รูปที่ ก.4 แสดงแถบการติดตั้งข้อมูลเสร็จ

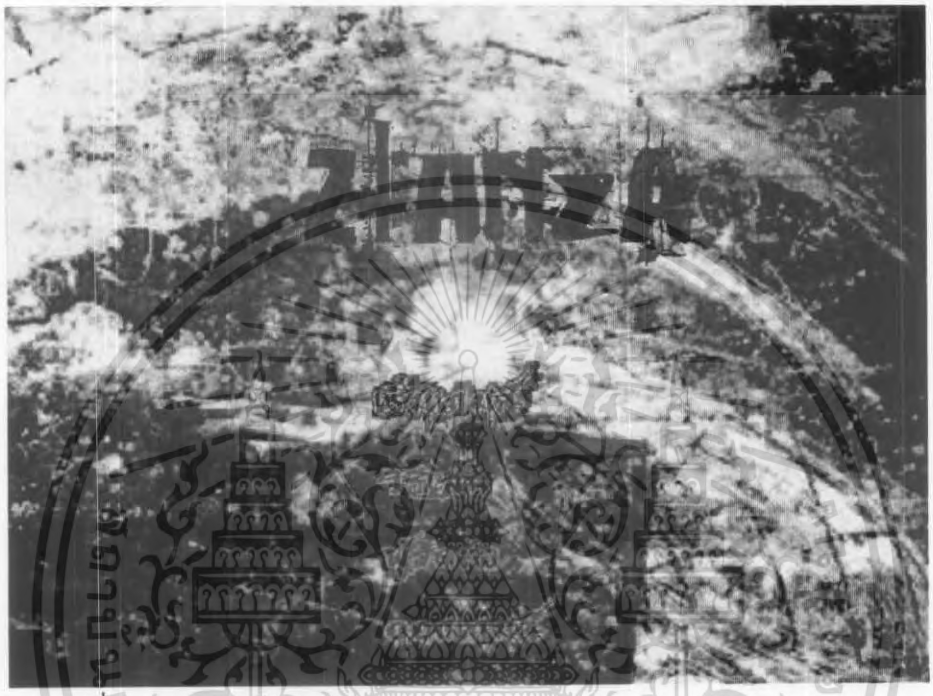
5. เสร็จสิ้นการติดตั้งเกม Ztanza โดยกดปุ่ม Finish

รูปที่ ก.5 แสดงการติดตั้งเสร็จสมบูรณ์

ภาคผนวก ข

คู่มือการเล่น

ทำการเข้าเกมจากไฟล์ที่ชื่อ Ztanza.EXE จะปรากฏหน้าจอดังรูป

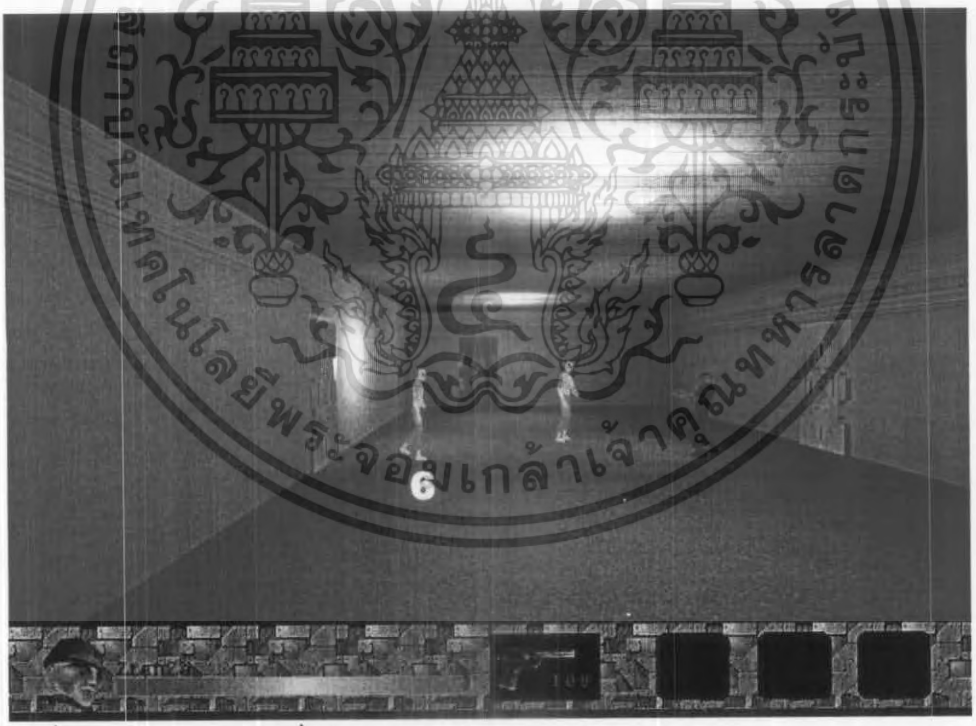


รูปที่ ข.1 แสดงหน้าจอเมนูหลักของเกม

- เลือกเมนู "เริ่มเกม" เพื่อเข้าสู่เกม
- เลือกเมนู "วิธีเล่น" เพื่อดูปุ่มบังคับต่าง ๆ
- เลือกเมนู "ผู้จัดทำ" เพื่อดูรายชื่อผู้พัฒนาเกม
- เลือกเมนู "ออกจากเกม" เพื่อออกจากเกม

หลังจากเลือกเมนู "เริ่มเกม" จะเข้าสู่เกม โดยรายละเอียดของหน้าจอต่าง ๆ เป็นดังนี้

รูปที่ ข.2 ความหมายของสิ่งต่าง ๆ ในหน้าจอขณะเล่นเกม



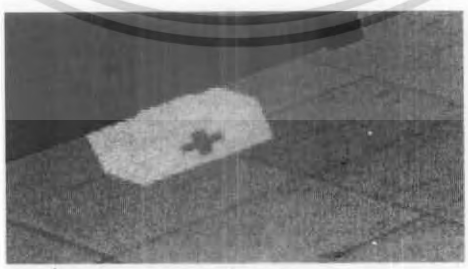
รูปที่ ข.3 ความหมายของสิ่งต่าง ๆ ในหน้าจอขณะเล่นเกม

หมายเลข 1	ตัวผู้เล่น
หมายเลข 2	พลังชีวิตของผู้เล่น
หมายเลข 3	จำนวนกระสุนที่มี
หมายเลข 4	ช่องเก็บไอเท็มต่าง ๆ
หมายเลข 5	ไอเท็มต่าง ๆ
หมายเลข 6	ศัตรู

ไอเท็มต่าง ๆ ภายในเกม จะถูกวางอยู่ในห้อง มีดังนี้

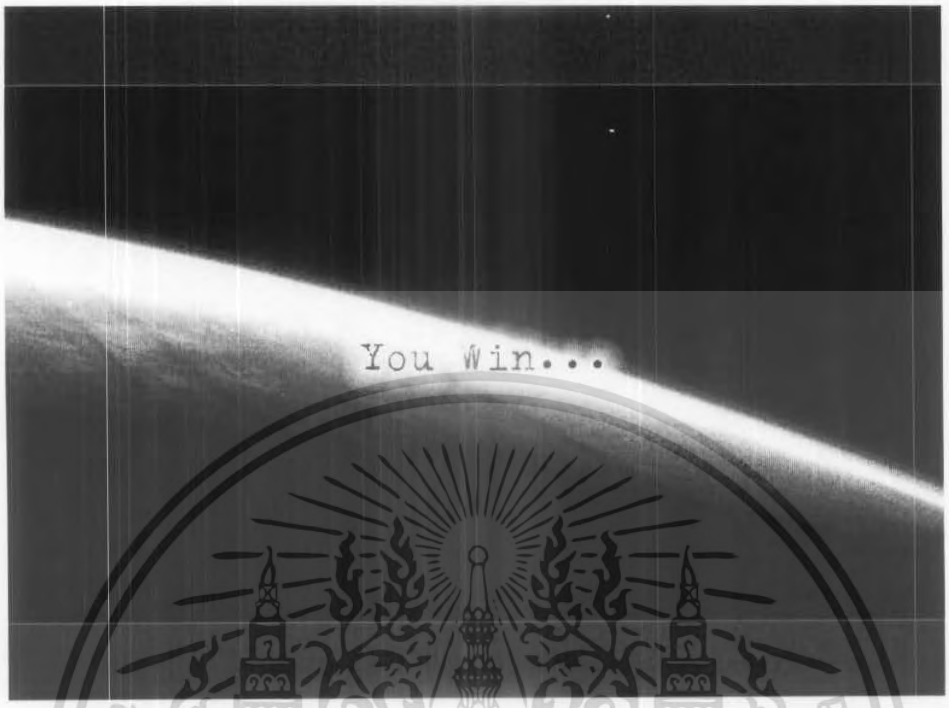
รูปที่ ข.4 แสดงไอเท็มกุญแจ

รูปที่ ข.5 แสดงไอเท็มกระสุน



รูปที่ ข.6 แสดงไอเท็มกล่องพยาบาล

เมื่อสามารถจัดการหัวหน้าได้ จะปรากฏหน้าจอดังนี้



รูปที่ ข.7 แสดงหน้าจอเมื่อชนะ

แต่ถ้าหากพลังชีวิตผู้เล่นหมด จะปรากฏหน้าจอดังนี้

รูปที่ ข.8 แสดงหน้าจอเมื่อผู้เล่นตาย