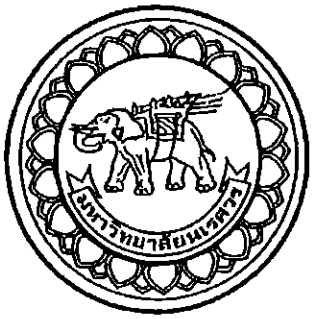


เกมไก่ชน

Cock Fighter Game

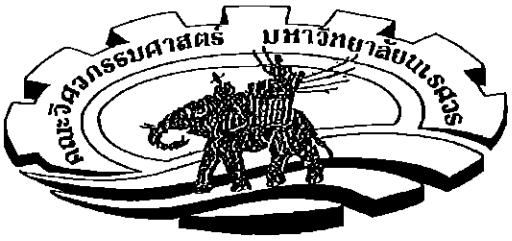
นายศิริโชค	ปวงเหมือง	รหัส 47380046
นายแสวงศักดิ์	นาถกร	รหัส 47380059

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2550



ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	เกมไก่ชน		
ผู้ดำเนินโครงงาน	นายศิริโชค	ป่วงเหมือง	รหัส 47380046
	นายแสวงศักดิ์	นาถกร	รหัส 47380059
อาจารย์ที่ปรึกษา	ดร.ไพศาล	มุณีสว่าง	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2550		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.ไพศาล มุณีสว่าง)

..... กรรมการ

(อาจารย์ศิริพร เดชะศิลารักษ์)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มเม่น)

หัวข้อโครงงาน	เกมไก่ชน
ผู้ดำเนินโครงงาน	นายศิริโชค ปวงเหมือง รหัส 47380046
	นายแสวงศักดิ์ นาถกร รหัส 47380059
อาจารย์ที่ปรึกษา	ดร.ไพศาล มุณีสว่าง
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาการพัฒนาโปรแกรมสามมิติ ซึ่งทำงานบนระบบปฏิบัติการ Windows XP โดยใช้โปรแกรม Microsoft Visual C++ 2005 และ DirectX 9.0c เป็นเครื่องมือในการพัฒนา ซึ่งจะช่วยในการจัดการเกี่ยวกับการแสดงผลภาพ 2 มิติ, 3 มิติ และระบบการรับอินพุตจากผู้เล่นด้วยเมาส์ นอกจากนี้ยังมีการใช้โปรแกรม Autodesk 3ds Max 9, Adobe Photoshop CS3 และ Adobe Illustrator CS3 เพื่อช่วยในออกแบบตัวละครสามมิติ, ฉากสำหรับการต่อสู้ และการตกแต่งภาพต่างๆ เพื่อใช้ในเกม

เกมไก่ชนที่พัฒนาขึ้นมานี้ เป็นเกมในรูปแบบสามมิติ ที่มีรูปแบบการต่อสู้ของไก่เป็นระบบปัญญาประดิษฐ์ทั้งสองฝ่าย โดยผู้เล่นจะต้องสวมบทบาทเป็นผู้เลี้ยงไก่ชน เพื่อนำไก่ของตนไปแข่งขัน หากไก่ผู้เล่นสามารถชนะการแข่งขันในการแข่งขัน จะถือว่าเป็นฝ่ายชนะ

Project title	Cock fighter game.		
Name	Mr Sirichoke	Puangmeung	ID. 47380046
	Mr. Sawangsak	Narttakorn	ID. 47380059
Project advisor	Paisarn	Muneesawang, Ph.D.	
Major	Computer Engineering.		
Department	Electrical and Computer Engineering.		
Academic year	2007		

.....

ABSTRACT

This project is studying and developing a 3D game for Windows XP operating system. Microsoft Visual C++ 2005 and DirectX 9.0c are used to develop this game. They have functions to manage displaying system of images in 2 dimension, 3 dimension and input system from mouse. Moreover Autodesk 3ds Max 9, Adobe Photoshop CS3 and Adobe Illustrator CS3 are used to design character model, create 3D objects, fighting scene, design texture and button for the game.

The cock fighter game is a 3D game which use artificial intelligence theory in fighting system. Player will be act as a cock trainer. The main goal for this game is to train a cock to fight and win in the competition.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์สำเร็จได้ด้วยดี เนื่องด้วยความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษา ดร.ไพศาล มุณีสว่าง, ผู้ช่วยศาสตราจารย์ ดร.สุชาติ แย้มแม่น และอาจารย์ศิริพร เดชะศิลารักษ์ ที่กรุณาให้คำปรึกษา แนะนำวิธีการในการทำงาน ตลอดถึงการตรวจสอบการทำงานพร้อมทั้งเสนอแนวทางการแก้ไขตลอดระยะเวลาการทำโครงงาน สุดท้ายต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ พี่ ทุกคนที่ยังไม่ได้เอ่ยนามที่คอยสนับสนุนในการทำโครงงานครั้งนี้

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญ

เนื่องจากกระแสเกมประเภทดำเนินตัวละครตามเนื้อเรื่อง หรือ RPG (Role Playing Game) นั้นได้รับความนิยม อีกทั้งตัวเกมประเภทนี้จะมีระบบการเล่นที่น่าสนใจ บางส่วนต้องใช้ความคิดอย่างรอบคอบเพื่อที่จะก้าวไปสู่ชัยชนะ อีกทั้งมีความหลากหลายของระบบการเล่นจนทำให้สามารถแยกย่อยออกเป็นอีกหลายประเภทอาทิเช่น Action RPG ซึ่งเป็นรูปแบบที่ต้องบังคับตัวละครตลอดเวลาหรือเป็นแบบ Real-Time ซึ่งผู้เล่นจะต้องคอยแก้สถานการณ์ในเกมตลอดเวลา หรือที่กำลังเป็นที่นิยมมากที่สุดในตอนนี้คือ MMORPG (Massive Multi-player Online Role Playing Game) ซึ่งจะเป็นระบบการเล่นแบบออนไลน์โดยผู้เล่นสามารถสนทนาหรือแก้ปัญหาร่วมกันได้ดังจะพบผู้เล่นได้ทั่วไป

ในส่วนของโครงงานนี้รูปแบบที่จะพัฒนาขึ้นมาจะเป็นระบบที่ใช้รูปแบบของ Simulation RPG ซึ่งเป็นระบบที่จะจำลองรูปแบบการเล่น, เหตุการณ์ หรือสิ่งที่มีในเกมมาจากของจริง โดยอาจจะแต่งเติมบางส่วนเพื่อที่จะให้ตัวเกมนั้นมีสีสัน มีความน่าเล่น และน่าสนใจ โดยส่วนที่น่าศึกษาในเกมที่จะพัฒนาขึ้นนี้ก็คือระบบปัญญาประดิษฐ์ (Artificial Intelligence) ที่ใช้ในเกมโดยในระบบต่อสู้นั้นจะใช้ระบบปัญญาประดิษฐ์ เพื่อแสดงการต่อสู้และตัดสินหาผู้ชนะ โดยระบบปัญญาประดิษฐ์จะเป็นส่วนตัดสินใจด้วยตัวเองทั้งหมด โดยผู้เล่นจะมีหน้าที่เพียงเลือกรูปแบบที่เหมาะสมเพื่อเป็นการวางแผนและคำนวณว่าระบบปัญญาประดิษฐ์ของฝ่ายผู้เล่นจะสามารถชนะระบบปัญญาประดิษฐ์ ที่เป็นของตัวเกมได้หรือไม่เท่านั้น และโครงงานนี้ยังเป็นช่วยในการศึกษาการเขียน Application บน Windows โดยใช้ความสามารถของ DirectX SDK มาพัฒนาอีกด้วย

โดยเกมที่จะพัฒนาขึ้นนั้นคือ เกม Thai Cock Fighter (เกมไก่ชนไทย) จะมีรูปแบบการเล่นโดยให้ผู้เล่นฝึกสอนไก่ของตนเอง แล้วให้ไก่ต่อสู้กับคู่ต่อสู้ โดยอาศัยทักษะที่ผู้เล่นได้ฝึกสอนมา ซึ่งในขณะที่ไก่ต่อสู้กันนั้น ผู้เล่นจะไม่มีสิทธิ์บังคับไก่ได้เลย ไก่จะต่อสู้เองโดยใช้ระบบปัญญาประดิษฐ์ ควบคุมตนเองและตัดสินใจด้วยตนเอง และใช้ชุดคำสั่งของ DirectX SDK เข้ามาช่วยเพื่อให้สามารถเขียนเกมที่มีประสิทธิภาพมากกว่าการใช้โปรแกรมสำเร็จรูปที่มีอยู่ทั่วไป ความสามารถหลายๆอย่างของ DirectX นั้นมีความสะดวกในการเรียกใช้และนำมาปรับปรุงเข้ากับตัวเกมที่จะพัฒนาขึ้น อาทิเช่นการเรียกไฟล์รูปภาพ, ภาพเคลื่อนไหว, โมเดลสามมิติ และความสามารถอื่นๆที่จะนำมาใช้ได้

คณะผู้จัดทำจึงมีแนวคิดที่พัฒนาเกมรูปแบบนี้ขึ้นมา เพื่อที่จะเพิ่มความหลากหลายของระบบเกมที่มีในปัจจุบัน และด้วยการศึกษาการเขียนโปรแกรมโดยใช้ความสามารถของ DirectX

SDK นี้จะให้มีประสบการณ์และเพิ่มความสามารถในการเขียนโปรแกรม เพื่อที่จะสามารถพัฒนาเป็นต้นแบบสำหรับการพัฒนาในสถาปัตยกรรมอื่นได้อีกต่อไป และอาจสามารถใช้ในเชิงธุรกิจในอนาคตอีกด้วย

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อเขียนเกมที่มีระบบการต่อสู้โดยให้ระบบปัญญาประดิษฐ์มีการตอบสนองกันเอง

1.2.2 เพื่อศึกษาการเขียน Application บนระบบปฏิบัติการ Windows XP

1.2.3 เพื่อศึกษาการประยุกต์ใช้ DirectX SDK เข้ากับซอฟแวร์ Microsoft Visual C++

1.2.4 เพื่อศึกษาการสร้างระบบปัญญาประดิษฐ์

1.3 ขอบเขตของโครงงาน

1. สร้างเกมที่มีการต่อสู้ในรูปแบบ 3 มิติ ที่จำลองบทบาทให้ผู้เล่นเป็นผู้เลี้ยงไก่ชน โดยเริ่มต้นผู้เล่นจะได้รับไก่มาหนึ่งตัวเพื่อที่จะฝึกฝนให้เก่งขึ้นแล้วนำไปต่อสู้ในการแข่งขันที่จำลองขึ้น ผู้เล่นสามารถใช้ไอเท็มเพื่อที่จะเพิ่มความสามารถให้ไก่ได้ สามารถนำไก่ไปชนเพื่อเพิ่มค่าประสบการณ์เพื่อให้ไก่เก่งขึ้น และเป็นการสะสมเพื่อนำไปใช้ในการซื้อไอเท็มโดยเป้าหมายหลักคือจะต้องสู้กับไก่ชนที่ได้เตรียมไว้ โดยเริ่มจากตัวที่เก่งน้อยที่สุดก่อนหากชนะก็จะสามารถที่จะไปสู้กับตัวต่อไปที่เก่งกว่าได้ หากสามารถชนะได้ทั้งหมด ก็จะสามารถชนะเกมได้

2. ปัญญาประดิษฐ์ในเกมสามารถควบคุมไก่ได้ดีในระดับหนึ่ง

1.4 แผนการดำเนินงาน

ในการดำเนินงานของโครงการมีระยะเวลาในการจัดทำโครงการนี้ทั้งหมด 7 เดือน โดยการพัฒนาจะเริ่มในเดือนตุลาคม พ.ศ. 2550 สรุปผลของโครงการในเดือนมีนาคม พ.ศ. 2551 สำหรับการดำเนินงานมีขั้นตอนดังต่อไปนี้

ตารางที่ 1.1 แสดงขั้นตอนการดำเนินโครงการ

กิจกรรม	พ.ศ. 2550			พ.ศ. 2551			
	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. ศึกษาการเขียนโปรแกรมเชิงวัตถุ	←	—	—	→			
2. ศึกษาการสร้างโมเดล 3 มิติ		←	—	—	→		
3. ศึกษาการใช้ DirectX SDK กับ Microsoft Visual C++			←	—	—	→	
4. ศึกษาระบบปัญญาประดิษฐ์				←	—	→	
5. ออกแบบการทำงานของเกม				←	—	→	
6. พัฒนาเกม					←	—	→
7. ทดสอบและปรับปรุงตัวเกม						←	→
8. สรุปผลและจัดทำคู่มือ						←	— →

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 ได้รับความรู้จากการเขียน Application บนระบบปฏิบัติการ Windows XP

1.5.2 สามารถประยุกต์ใช้ DirectX SDK เข้ากับ Visual C++

1.5.3 ได้รับความรู้จากการศึกษาออกแบบระบบปัญญาประดิษฐ์

1.5.4 ได้รับความรู้จากการศึกษาการออกแบบโมเดลแบบ 3 มิติด้วย Autodesk 3dsMax 9

1.5.5 ทำให้ได้มีโอกาสประยุกต์ความรู้ที่เรียนมาในการศึกษาออกแบบโปรแกรมทางด้านคอมพิวเตอร์

1.5.6 ได้เกมที่สร้างความสนุกสนานเพลิดเพลินแก่ผู้เล่น และได้ฝึกการวางแผนอย่างรอบคอบ

1.5.7 เกมที่ได้สามารถนำไปพัฒนาต่อในสถาปัตยกรรมอื่นได้

1.6 รายละเอียดงบประมาณของโครงงาน

1.6.1 ค่าใช้จ่ายในการซื้ออุปกรณ์ทางคอมพิวเตอร์	1,000 บาท
1.6.2 ค่าเอกสารและคู่มือ	500 บาท
1.6.3 ค่าใช้จ่ายในการทำรายงาน	500 บาท
รวมทั้งสิ้น	2,000 บาท

บทที่ 2
หลักการและทฤษฎี

ในบทที่ 2 นี้จะเป็นการกล่าวถึงทฤษฎีต่างๆ ที่ได้นำมาใช้ในการพัฒนาโปรแกรม เกม 3มิติ ซึ่งได้แก่ทฤษฎีการใช้งาน DirectX การใช้งาน Microsoft Visual C++ 2005 ในการสร้างหน้าต่างวินโดว์ และทฤษฎีที่ใช้ในการออกแบบโมเดล 3 มิติ สำหรับใช้ในเกม และหลักการปัญญาประดิษฐ์ (Artificial intelligence หรือ AI) ที่นำมาใช้ในเกม

ขั้นตอนการศึกษาทฤษฎี

1. ศึกษาการสร้างหน้าต่างวินโดว์ ด้วย Microsoft Visual C++ 2005
2. ศึกษาโครงสร้างและทฤษฎีการใช้
3. ศึกษาทฤษฎีเกี่ยวกับการออกแบบโมเดล 3 มิติ เพื่อใช้ในเกม
4. ศึกษาหลักการปัญญาประดิษฐ์ที่นำมาใช้ในเกม

2.1 การเขียนโปรแกรมด้วย Microsoft Visual C++

Visual C++ สร้างขึ้นโดยบริษัทไมโครซอฟต์ (Microsoft) เพื่อใช้ในการพัฒนาโปรแกรมบนวินโดว์ ซึ่งมีรูปแบบการพัฒนาเป็นแบบ Visual คือ ออกแบบลักษณะหน้าตาของโปรแกรมได้ง่าย มี MFC (Microsoft Foundation Class) ซึ่งเป็น Class library ให้ใช้งาน ซึ่งจะทำให้การพัฒนาโปรแกรมสามารถทำได้ง่ายยิ่งขึ้น

ฟังก์ชั่นหลักๆที่ใช้ในการสร้างหน้าต่างวินโดว์มีดังนี้

- WinMain : จะต้อง include <windows.h>
- WndProc : จะต้อง include <windows.h>

2.1.1 ฟังก์ชั่น WinMain ฟังก์ชั่นนี้ เป็นฟังก์ชั่นหลักของโปรแกรมทั้งหมด เมื่อโปรแกรมทำงานฟังก์ชั่นนี้จะถูกเรียกเป็นอันดับแรก

รูปแบบฟังก์ชั่น WinMain()

```
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow)
```

พารามิเตอร์

- hInstance : Type คือ HINSTANCE เป็นตัวที่ชี้ไปยังโปรแกรมหรือหมายเลขของโปรแกรม ซึ่งโปรแกรมสามารถทำงานได้พร้อมกันหลายๆโปรแกรมโดยใช้ โค้ดเดียวกัน ดังนั้นจึงต้องมีตัวที่อ้างถึงโปรแกรม

- hPrevInstance : เหมือนกับ hInstance แต่เป็นตัวที่ชี้ไปยังโปรแกรมที่ทำงานก่อนหน้า ถ้าหากโปรแกรมไม่ได้ทำงานด้วยโค้ดเดียว จะคืนค่า 0

- lpCmdLine : Type คือ LPSTR เป็นตัวชี้ที่ชี้ไปยังข้อความที่ได้จากการใช้ Command line

- nCmdShow : Type คือ Integer เป็นตัวบอกลักษณะหน้าต่างเช่น minimize , maximize

ขั้นตอนที่เกิดขึ้นในฟังก์ชั่น WinMain

1. การสร้างตัวแปร โครงสร้างของคลาส WNDCLASS (เป็นการกำหนดลักษณะของหน้าต่าง)
2. ทำการ Register Class เข้าไปในระบบ ด้วยฟังก์ชั่น RegisterClass()
3. ทำการสร้างหน้าต่างด้วยฟังก์ชั่น CreateWindow()
4. จัดการเกี่ยวระบบ Message (ตรวจจับ Message)

2.1.2 ฟังก์ชั่น WndProc จะทำหน้าที่ในการตอบสนองต่อ Message ที่ถูกส่งมาจากฟังก์ชั่น WinMain หากต้องการให้โปรแกรมทำอะไรเมื่อมีการส่ง Message เกิดขึ้น ก็จะทำในฟังก์ชั่นนี้ ฟังก์ชั่น WinMain และฟังก์ชั่น WinProc จะมีส่วนเกี่ยวข้องกันก็ต่อเมื่อฟังก์ชั่น WinMain มีการตรวจสอบการเจอ Message

รูปแบบฟังก์ชั่น WinProc()

LRESULT CALLBACK WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam);

พารามิเตอร์

- hWnd : Type คือ HWND เป็นตัวที่อ้างถึงหน้าต่างโปรแกรมที่เราสร้างขึ้น
- message : Type คือ UINT (Unsigned Integer) จะเก็บหมายเลข Message ที่จะนำไปใช้ในฟังก์ชั่นนี้
- wParam : Type คือ WPARAM (word parameter) จะเก็บค่าเหตุการณ์ที่มากับ Message
- lParam : Type คือ LPARAM (long parameter) จะเก็บเหตุการณ์ที่มากับ Message เช่นกัน ซึ่งจะสอดคล้องเหตุการณ์ที่ wParam

ใน WndProc จะทำการตรวจสอบ message ที่รับมา ว่าตรงกับ case ไหน ก็จะทำตาม case นั้น ซึ่งเราสามารถสั่งให้โปรแกรมทำอะรก็ได้ในแต่ละ case ตามเหตุการณ์ที่เราต้องการ เช่น ถ้า Message ที่ส่งตรงกับ WM_DESTROY โปรแกรมก็จะทำการคืนค่าหน่วยความจำแล้วฟังก์ชั่น PostQuitMessage ก็จะทำการกำหนดค่า exit code เป็น 0 ซึ่งจะทำให้ฟังก์ชั่น GetMessage คืนค่า False กัลบมาทำให้จบ loop การทำงานแล้วก็ออกจากโปรแกรม ถ้าหากไม่เข้า case ไหนเลย โปรแกรมก็จะเรียกใช้ฟังก์ชั่น DefWinProc ซึ่งฟังก์ชั่นนี้จะจัดการค่า Default ของ Windows ให้เอง

2.2 การใช้งาน DirectX SDK

2.2.1 DirectX เป็นชุดคำสั่งที่ใช้ในการเขียนเกมที่มีประสิทธิภาพสูงทั้ง เกม 2 มิติ และ 3 มิติ พัฒนาโดยบริษัท Microsoft ซึ่งเป็นชุดคำสั่งที่อำนวยความสะดวกในการเขียนเกม ทำให้ผู้พัฒนาโปรแกรมเรียกใช้ความสามารถด้านฮาร์ดแวร์ของระบบได้อย่างเต็มประสิทธิภาพไม่ว่าจะเป็นการแสดงผลภาพ เสียง และการควบคุมเกมด้วยอุปกรณ์ต่างๆ เช่น mouse keyboard หรือ joystick เป็นต้น อีกทั้งยังมีชุดคำสั่งที่สนับสนุนการเล่นเกมแบบหลายผู้เล่น (Multiplayer Game) บนเครือข่ายเดียวกัน และบนอินเตอร์เน็ตอีกด้วย โดยมีข้อดีคือสามารถเข้าถึงระบบฮาร์ดแวร์ได้โดยตรงไม่ต้องผ่านทางระบบปฏิบัติการจึงมีชื่อว่า DirectX ทำให้มีความเร็วสูงกว่าเกมหรือโปรแกรมทางด้าน multimedia ที่ต้องควบคุมฮาร์ดแวร์ผ่านระบบปฏิบัติการ และข้อดีอีกประการก็คือโปรแกรมที่สร้างขึ้นสามารถรันได้บนเครื่องคอมพิวเตอร์ที่มีระบบฮาร์ดแวร์แตกต่างกันได้โดยไม่มีปัญหา แต่ฮาร์ดแวร์เหล่า นั้นต้องสนับสนุน DirectX ด้วยเช่นกัน และผลลัพธ์ที่ได้ในเครื่องหนึ่งอาจจะไม่เหมือนกับอีกเครื่องหนึ่งทุกประการ

DirectX แบ่งเป็น 2 ส่วนดังนี้

1. DirectX Runtime Library เป็นส่วนหนึ่งของ DirectX ที่ทำให้สามารถเล่นเกมที่เขียนขึ้นโดย DirectX SDK ได้ ซึ่งจะถูกติดตั้งลงไปในระบบปฏิบัติการของ Windows ให้โดยอัตโนมัติ

2. DirectX SDK (Software Development Kit) เป็นชุดคำสั่งที่สร้างขึ้นมาเพื่อใช้ในการพัฒนาโปรแกรม ซึ่งถูกออกแบบมาให้ใช้ร่วมกับตัวแปรภาษาได้หลายภาษา เช่น C หรือ C++ Delphi และ Visual basic เป็นต้น ซึ่งสามารถศึกษาการพัฒนาเกมด้วย DirectX SDK ได้ด้วย document ที่มีมาพร้อมกับโปรแกรม

DirectX ประกอบไปด้วยชิ้นส่วนย่อย ๆ มากมาย ลักษณะพิเศษของ DirectX คือ ถึงแม้ว่าบาง Component ของ DirectX จะถูกประกาศว่าไม่สนับสนุน (Deprecated) แล้ว แต่ทุก ๆ Component และทุก ๆ เวอร์ชั่นที่ผ่านมาของ Component ทุก Component จะยังคงสามารถเข้าถึงได้ ซึ่งเป็นลักษณะเด่นของ API ที่สร้างขึ้นจากพื้นฐานของ COM โดย COM มีอยู่ด้วยกัน 2 ส่วนคือ COM Interface และ COM Object โดยโครงสร้างต่างๆ จะคล้ายกับคลาสของภาษา C++ คือจะต้องสร้าง COM Object ก่อนถึงจะใช้งานได้

ตัวอย่าง Component ที่มีใน DirectX

- DirectGraphic หรือ Direct3D ประกอบด้วยคำสั่งทางด้านการประมวลผลภาพ 3 มิติ ออกสู่หน้าจอคอมพิวเตอร์
- DirectSound ประกอบด้วยคำสั่งทางด้านการประมวลผลเสียงต่างๆ
- DirectInput ประกอบด้วยคำสั่งในการรับข้อมูลจากผู้ใช้ผ่านทาง เมาส์ คีย์บอร์ด และจอยสติก เป็นต้น

2.2.2 หลักการใช้งาน DirectX และ Direct3D9 ในขั้นแรกจะต้องทำการประกาศ ไฟล์เฮดเดอร์ของ Direct3D และ D3DX Library เข้ามาเสียก่อนที่ส่วนหัวของโปรแกรม

```
#include <d3d9.h>
#include <d3dx9.h>
```

โดยไฟล์เฮดเดอร์ d3d9.h จะเป็นไฟล์เฮดเดอร์ของ Direct3D และ d3dx9.h จะเป็นไฟล์เฮดเดอร์ของ D3DX Library เมื่อประกาศ ไฟล์เฮดเดอร์เสร็จเรียบร้อยแล้วจะต้องทำการบอกให้ C++ ทราบว่าจะเรียกใช้งานฟังก์ชั่นจากไลบรารี d3d9.lib และ d3dx9.lib

```
#pragma comment(lib,"d3d9.lib")
#pragma comment(lib,"d3dx9.lib")
```

เมื่อถึงจุดนี้แล้วก็จะเริ่มใช้งาน Direct3D และ D3DX Library ได้แล้ว ในขั้นแรกจะต้องประกาศและสร้าง COM Object หลักของ Direct3D เพื่อจะเรียกใช้งานเมธอดต่างๆ และสร้าง COM Interface ตัวอื่นๆ จาก COM Object หลักของ Direct3D ต่อไป การประกาศ COM Object ของ Direct3D9 ทำได้โดย

```
LPDIRECT3D9 g_lpD3D = NULL;
IDirect3D9 * Direct3DCreate9(
UINT SDKVersion
);
```

IDirect3D9 คือ Object ของ Direct3D9 ที่ประกาศไว้ในตอนแรก

SDKVersion เป็นค่าคงที่ซึ่งมีการประกาศไว้ในไฟล์ d3d9.h แล้วเพียงใส่ค่าคงที่ D3D_SDK_VERSION นี้ลงไปเท่านั้น

ถ้าหากเมธอดนี้ทำงานสำเร็จก็จะคืนค่าพอยเตอร์มายัง Object ของ Direct3D9 ที่ประกาศไว้และคืนค่า NULL มาให้เมื่อไม่สำเร็จ ตัวอย่างการเรียกใช้เมธอดมีดังนี้

```
if(NULL == (g_lpD3D = Direct3DCreate9(D3D_SDK_VERSION)))return ERROR;
```

หลังจากสร้าง Direct3D9 Object เรียบร้อยแล้วก็จะสามารถที่จะเรียกใช้งานเมธอดที่อยู่ภายใน Object ตัวนี้ได้ การเรียกใช้งานเมธอดนั้นโดยหลักการก็คล้ายๆ กับการเรียกใช้เมธอดของ Class ของ C++ นั้นเอง

```
COM Interface -> ชื่อเมธอดที่จะเรียกใช้(พารามิเตอร์);
```

หลังจากทำการสร้าง COM Object ของ Direct3D ต่อมาจะต้องสร้าง COM Interface หลักอีกตัวหนึ่งซึ่งใช้กันมากและใช้สร้าง COM Interface ตัวอื่นๆ อีกหลายตัวนั้นคือ Direct3DDevice9 โดย COM Interface ตัวนี้จะทำหน้าที่ในการติดต่อและควบคุม Device ต่างๆ ที่อยู่ในเครื่องเช่น การ์ดจอ ให้แสดงผลออกมาตามที่เราต้องการ แต่การจะสร้าง COM Interface ตัวนี้ได้ก็จำเป็นที่จะต้องเลือก Device ที่เหมาะสมให้แก่มัน โดยเราประกาศ COM Interface ของ Direct3DDevice9 ไว้เป็นตัวแปรแบบ Global ไว้ก่อน

```
LPDIRECT3DDEVICE9 g_lpd3dDevice;
```

โหมดการแสดงผลที่ DirectX และ Direct3D9 สามารถที่จะแสดงผลได้นั้นสามารถแบ่งออกเป็น 2 โหมดคือ

- FullScreen Mode คือโหมดเต็มจออย่างที่เกมทั่วไปๆ ใช้กัน หากเลือกใช้โหมดนี้จะต้องทำการกำหนดความละเอียดของหน้าจอ (Resolution) และจำนวนบิทต่อพิกเซล (bit per pixel) เอง

- Windowed Mode ก็คือโหมดทั่วไปของวินโดวส์ที่หน้าต่างหนึ่งๆ จะสามารถซ้อนทับอีกหน้าต่างหนึ่งและ สามารถ Minimized หรือ Maximized ได้นั่นเอง

เมื่อสร้าง COM Interface ของ Direct3DDevice9 สำเร็จแล้ว ก็จะสามารถเรียกใช้เมธอดภายใน COM Interface ตัวนี้ได้แล้ว นั่นคือจะสามารถควบคุม Device เพื่อที่จะแสดงผลออกสู่หน้าจอได้

การแสดงผลออกสู่หน้าจอนั้นจะเกี่ยวข้องกับการเขียนข้อมูลลงไปบน BackBuffer ซึ่งขั้นแรกจะต้องเคลียร์ Buffer ทั้งหมดไม่ว่าจะเป็น BackSurface, Z-Buffer หรือ stencil buffer เพื่อให้พร้อมต่อการประมวลผลภาพ (Render) จากนั้นจึงเริ่มเขียนข้อมูลลงไปบน BackBuffer โดยมีข้อแม้ว่าจะต้องทำอยู่ระหว่างเมธอด BeginScene และ EndScene เมื่อประมวลผลภาพเสร็จแล้วจึงทำการสลับ Surface จาก Back Surface ไปเป็น Front Surface (แสดงผลออกสู่หน้าจอนั่นเอง) โดยใช้เมธอด Present

```
HRESULT Clear(
        DWORD Count,
        CONST D3DRECT * pRects,
        DWORD Flags,
        D3DCOLOR Color,
        float Z,
        DWORD Stencil
        );
```

- Count กำหนดเป็น 0
- pRects กำหนดเป็น NULL เพราะเราจะเคลียร์ทั้งหน้าจอ
- Flags ค่าซึ่งจะบอกว่าจะทำการเคลียร์Buffer ใดบ้างซึ่งมีให้เลือกดังนี้
    - D3DCLEAR_STENCIL เคลียร์ stencil buffer
    - D3DCLEAR_TARGET เคลียร์หน้าจอปัจจุบัน
    - D3DCLEAR_ZBUFFER เคลียร์ depth buffer หรือ Z-Buffer

ซึ่งค่าเหล่านี้สามารถนำมารวมกันโดยใช้เครื่องหมาย | หากต้องการเคลียร์มากกว่าหนึ่ง Buffer อย่างเช่น D3DCLEAR_TARGET |D3DCLEAR_STENCIL| D3DCLEAR_ZBUFFER เป็นการเคลียร์ Buffer ทั้งหมด

- Color สีที่จะใช้ในการเคลียร์หน้าจอ สามารถใส่ค่าตัวเลขจำนวนเต็มไม่มีเครื่องหมายแทนค่าสีเข้าไปได้เลยหากทราบรหัสแทนค่าสี หรืออาจใช้ D3DCOLOR_XRGB (ค่าสีแดง, ค่าสีเขียว, ค่าสีน้ำเงิน) ก็ได้ ค่าสีนี้แต่ละแม่สีอยู่ในช่วง 0 ถึง 255
- Z ค่าที่ใช้ในการเคลียร์ depth buffer หรือ Z-Buffer มักกำหนดให้เป็น 1.0f
- Stencil ค่าที่ใช้ในการเคลียร์ stencil buffer มักกำหนดให้เป็น 0

```
HRESULT BeginScene(); HRESULT EndScene(); HRESULT Present(
        CONST RECT * pSourceRect,
        CONST RECT * pDestRect,
        HWND hDestWindowOverride,
        CONST RGNDATA * pDirtyRegion
);
```

พารามิเตอร์ในเมธอด Present กำหนดให้ทุกค่าเท่ากับ NUL

```
g_pd3dDevice->Clear( 0, NULL, D3DCLEAR_TARGET,
D3DCOLOR_XRGB(0,0,255), 1.0f, 0 );
g_pd3dDevice->BeginScene();
        // Rendering of scene objects happens here
g_pd3dDevice->EndScene();
g_pd3dDevice->Present( NULL, NULL, NULL, NULL );
```

เมื่อถึงเวลาปิดโปรแกรมหรือเลิกใช้งาน COM ของ Direct3DDevice9 และ Direct3D จะต้องถูกปิดการทำงานโดยการเรียกเมธอด Release ซึ่งมีข้อกำหนดว่า COM Object ของ Direct3D จะต้องถูก Release หลังสุดทุกครั้งเสมอ

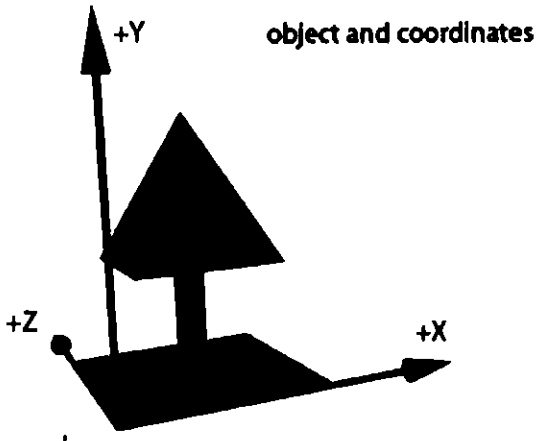
2.3 ทฤษฎีเกี่ยวกับการออกแบบโมเดล 3 มิติ

2.3.1 ระบบเมทริกซ์แบบ 3 มิติ (3D Matrix Math)

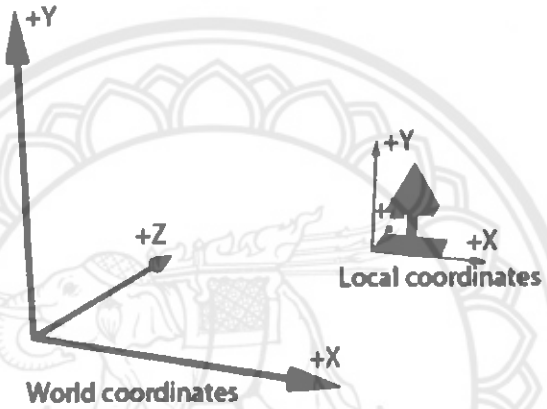
ในระบบเมทริกซ์แบบ 3 มิติ จะใช้เมทริกซ์เพื่อช่วยคำนวณกระบวนการสร้างภาพ 3D โดยจะใช้คำนวณตั้งแต่ขั้นตอน World Transformation, View Transformation และ Projection transformation

2.3.1.1 การแปลงระยะพิกัด (World Transformation)

เป็นการแปลงระยะพิกัดของตัวโมเดลสู่ระยะพิกัดจริงรูปแบบเมทริกซ์ของ World Transform ซึ่งก็คือการรวมรูปแบบ การย้าย (translation), การหมุน (rotation), และการย่อ-ขยาย (scaling) นั่นเอง



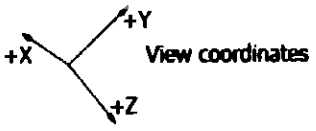
รูปที่ 2.1 แสดงตัวอย่างวัตถุและพิกัดของวัตถุ

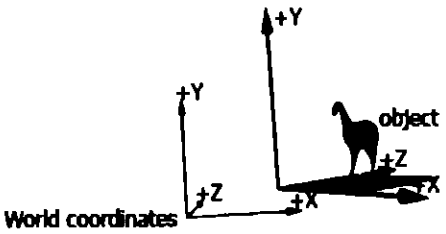


รูปที่ 2.2 แสดงการแปลงวัตถุลงสู่ พิกัดจริง (World coordinates) โดยเป็นการย้ายวัตถุ และแสดงพิกัดของวัตถุนี้ซึ่งเป็น พิกัดย่อย (local coordinates) เมื่ออยู่ในระบบพิกัดจริง

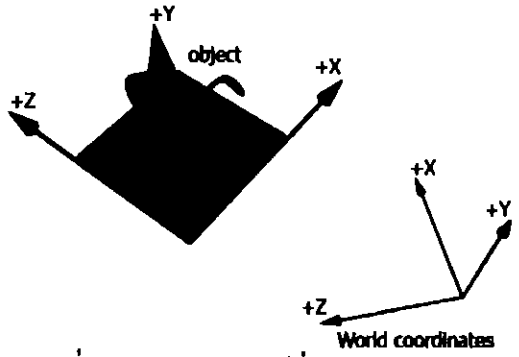
2.3.1.2 การวางตำแหน่งกล้อง (View Transformation)

View transformation เป็นเหมือนตำแหน่งของกล้อง ข้อมูลทิศทางต่าง ๆ ก็เหมือนทิศที่กล้องฉายไป หรือเปรียบได้กับตาของเราที่มองไปยังโลก 3 มิติ (ภาพในจอภาพที่จะแสดงนั่นเอง)





รูปที่ 2.3 แสดงตำแหน่งพิกัดของ view



รูปที่ 2.4 แสดงผลลัพธ์ที่แสดงในจอภาพ

ในการกำหนด view matrix ให้ฉายไปยังวัตถุในมุมมองและตำแหน่งของกล้องตามต้องการนั้น ทำได้หลายทาง รูปแบบที่กระชับคือ จาก view matrix ที่อยู่ในตำแหน่งและทิศทางเดียวกับ world space ก็ย้ายไปยังจุดที่ต้องการ และหมุนแกนทั้ง 3 (x, y, z) ให้ได้ทิศทางที่ต้องการ ในรูปแบบนี้จะได้ตามสมการ

$$V = T \cdot R_z \cdot R_y \cdot R_x \qquad (2.1)$$

จากสมการ V คือค่า view matrix ของผลลัพธ์ ส่วนค่า T คือ matrix ของระยะที่ย้ายไป (translation matrix) และ Rx ถึง Rz คือ matrix การหมุนเพื่อให้ได้ทิศทางที่ต้องการ (rotation matrices) ซึ่งได้จากการหาค่ามุมที่หมุนกลับสู่ทิศทางเริ่มต้น ถ้าตามสมการข้างบนนั้นก็จะเริ่มหาไล่จากแกน x -> y -> z

ใน Direct3D นั้นมีฟังก์ชั่นเพื่อช่วยเหลือการคำนวณค่า view matrix ให้ได้รูปแบบข้างต้น โดยกำหนด 3 ค่าคือ

1. ตำแหน่งกล้อง (eye) [x, y, z]
2. จุดที่กล้องมองไป (look-at) [x, y, z]
3. ความเอียงของกล้อง (up) เป็นทิศทางและกำหนดด้วยค่า vector [x, y, z] กล้องที่ไม่เอียงทิศของ vector จะตั้งขึ้น ซึ่งค่าก็คือ [0, 1, 0] นั่นเอง

ค่าทั้งสามนี้จะนำไปแปลงเป็น Vector 3 ตัวคือ

1. Vector ทางขวา (right) เปรียบเหมือนพิกัดแกน X กำหนดอักษรย่อเป็น u
2. Vector แนวตั้ง (up) เปรียบเหมือนพิกัดแกน Y เราสามารถหาความเอียงกล้องได้จาก vector นี้ กำหนดอักษรย่อเป็น v
3. Vector ทิศที่มอง (view-direction) เปรียบเหมือนพิกัดแกน Z การมองซ้ายขวาหน้าหลังก้มเงยก็ดูจากข้อมูล vector นี้ กำหนดอักษรย่อเป็น n

และจะได้คุณสมบัติของทั้ง 3 vector เป็น

1. ยาว 1 หน่วย นั่นคือ $\sqrt{(x^2+y^2+z^2)}=1$ (ได้เป็นค่าความยาว)
2. ทั้ง 3 Vector มีทิศทางตั้งฉากกัน เหมือนพิกัดแกนในโลก 3D

Vector u, v, n กับตำแหน่งกล้อง c นำมาใส่ใน view matrix ได้ดังนี้

$$\begin{bmatrix} u_X & v_X & n_X & 0 \\ u_Y & v_Y & n_Y & 0 \\ u_Z & v_Z & n_Z & 0 \\ -(u\cdot c) & -(v\cdot c) & -(n\cdot c) & 1 \end{bmatrix} \quad (2.2)$$

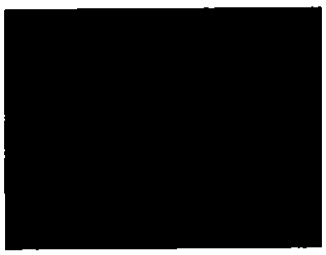
2.3.1.3 Projection transformation

เป็นการกำหนดรูปแบบลักษณะที่จะแสดงผลบนจอภาพ ซึ่งมี 2 ประเภทคือ Perspective และ Orthographic

1. Perspective Projection ภาพแบบ Perspective คือ ทัศนียภาพเหมือนที่ตาเห็น ยิ่งไกลวัตถุจะยิ่งดูเล็กลง เป็นต้น

รูปที่ 2.5 แสดงตัวอย่างภาพแบบ Perspective

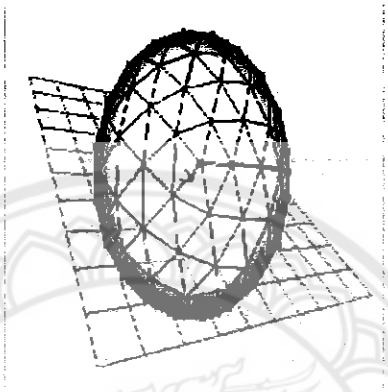
2. Orthographic Projection สำหรับภาพแบบ Orthographic นั้นวัตถุใกล้ไกลจะยังคงดูเท่าเดิม



รูปที่ 2.6 แสดงตัวอย่างภาพแบบ Orthographic

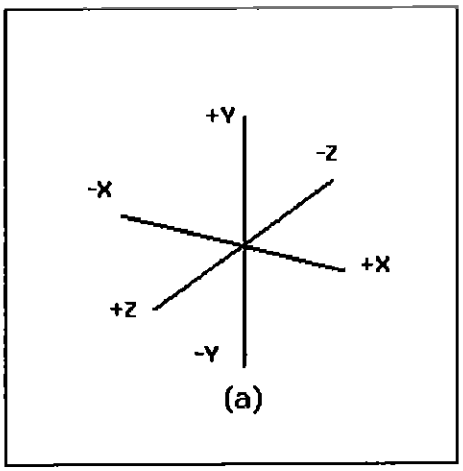
2.3.2 การสร้างโลก 3 มิติและ DirectX

โลก 3 มิติเกิดจากค่าพิกัด 3 ค่ารวมกัน (ความกว้าง, ความสูง, ความลึก หรือ แกน X, แกน Y, แกน Z) เรียกค่าพิกัดเหล่านี้ว่าจุด Vertex เมื่อนำจุด Vertex สามจุดมารวมกัน และใช้หลักการทางคณิตศาสตร์เรื่องเวคเตอร์ (Vector) และระบบระนาบ (Plane) เข้ามาเกี่ยวข้องก็จะทำให้เกิดรูป 3 เหลี่ยม (Triangle) หรือมากกว่า ซึ่งเรียกรวมๆ กันว่า Polygon (บางทีก็เรียกว่า Face) เมื่อนำ Polygon ทั้งแต่สอง Polygon มารวมเข้าด้วยกันจะทำให้เกิดวัตถุ 3 มิติ (3D geometry)

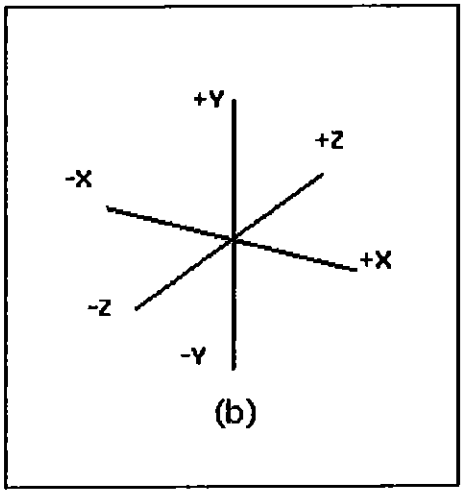


รูปที่ 2.7 แสดงรูปแบบของวัตถุ 3 มิติ ซึ่งเกิดจากการรวมตัวกันของ จุด Vertex และPolygon

ค่าพิกัดในแต่ละแกนนั้นมีค่าได้ทั้งลบและบวกส่วนค่าที่เป็น 0 แสดงว่าเป็นจุดกำเนิด ซึ่งจุดที่พิกัดในแกน X ที่เท่ากับ 0 แกน Y ที่เท่ากับ 0 และแกน Z ที่เท่ากับ 0 นั้นเราจะเรียกว่าจุดกำเนิด (Origin) ซึ่งจะอยู่ตรงจุดกึ่งกลางของโลก 3 มิติ ส่วนระบบพิกัดสามารถแบ่งออกได้เป็น 2 แบบคือระบบพิกัดมือซ้าย (Left-Hand Coordinate System) และระบบพิกัดมือขวา (Right-Hand Coordinate System) สำหรับ DirectX คุณสามารถเลือกใช้ระบบพิกัดได้ทั้งระบบพิกัดมือซ้ายและมือขวาแต่โดยพื้นฐานแล้ว DirectX จะใช้ระบบพิกัดมือซ้าย (เป็นค่าdefault) ซึ่งต่างจาก OpenGL ที่ใช้ระบบพิกัดมือขวา



รูปที่ 2.8 แสดงระบบพิกัดมือขวา



รูปที่ 2.9 แสดงระบบพิกัดมือซ้าย

โดยปรกติแล้วจุด Vertex ไม่ได้ประกอบด้วยจุดพิกัดในแกน X, แกน Y และแกน Z เพียงอย่างเดียวเพราะหากมีแค่นี้ก็ไม่อาจจะสร้างวัตถุ 3 มิติในโลก 3 มิติได้อย่างสมบูรณ์ มันยังประกอบด้วยอะไรๆ อีกหลายอย่างซึ่งก็แล้วแต่ผู้ออกแบบระบบจะต้องการให้มีและงานที่ต้องการจะใช้ แต่โดยส่วนใหญ่แล้วไม่ว่าจะเป็น DirectX หรือ OpenGL ก็จะมีลักษณะคล้ายกันค่าที่เสริมเข้ามานี้อาจจะเป็นค่าสีของวัตถุในแต่ละจุด Vertex (diffuse color หรือ specular color) ค่าพิกัดของลวดลาย (Texture) ที่จะปะ (Mapping) ลงไปบนวัตถุ 3 มิติซึ่งเรียกว่า Texture coordinate หรือพิกัด UV (เพราะใช้ค่า u และ v ในการระบุตำแหน่ง)

สำหรับการกำหนดจุด Vertex ของ DirectX จะใช้หลักการเดียวกับ OpenGL นั้นก็คือ Flexible Vertex Format (FVF) ซึ่งเป็นโครงสร้างข้อมูลที่จะบอก DirectX ว่าหนึ่งจุด Vertex ประกอบด้วยส่วนประกอบอะไรบ้าง โดยการกำหนดค่าคงที่ให้แก่ FVF ดังตัวอย่าง

```
#define CUSTOM_FVF D3DFVF_XYZ|D3DFVF_DIFFUSE|D3DFVF_TEX1
```

ค่า CUSTOM_FVF เป็นชื่อของค่าคงที่ที่ผู้ใช้เป็นผู้กำหนดเองขึ้นมา ส่วนค่า flags ต่างๆ ที่ตามมานั้นเป็นค่าคงที่ที่ DirectX กำหนดไว้ให้เพื่อเป็นตัวกำหนดว่า Vertex หนึ่งๆ ประกอบด้วยค่าอะไรบ้าง ซึ่งผู้ใช้จะต้องนำมากำหนดอีกทีโดยค่า Flags นั้นมีความหมายดังนี้

D3DFVF_XYZ เป็นการกำหนดให้จุด Vertex ประกอบด้วย ค่าพิกัด X Y และ Z

D3DFVF_NORMAL เป็นการกำหนดให้จุด Vertex ประกอบด้วยค่าพิกัด Normal ซึ่งจะต้องนำไปใช้คำนวณค่าแสงที่ตกกระทบบนวัตถุ

D3DFVF_DIFFUSE เป็นการกำหนดให้จุด Vertex ประกอบด้วยค่าสี Diffuse

D3DFVF_SPECULAR เป็นการกำหนดให้จุด Vertex ประกอบด้วยค่าสี Specular

D3DFVF_TEX1 ถึง D3DFVF_TEX8 เป็นการกำหนดให้จุด Vertex ประกอบด้วยค่าพิกัด u และ v ซึ่งเป็นตำแหน่งของลวดลายที่จะปะลงไปบนวัตถุ

```
#define CUSTOM_FVF
D3DFVF_XYZ|D3DFVF_NORMAL|D3DFVF_DIFFUSE|D3DFVF_TEX1
struct Vertex
{
        float x,y,z;    // D3DFVF_XYZ
        float nx,ny,nz; // D3DFVF_NORMAL
        DWORD dif;      // D3DFVF_DIFFUSE
        float u,v;      // D3DFVF_TEX1
};
```

หรือ

```
struct Vertex
 {
        D3DXECTOR3 Position;  // float x,y,z;
        D3DXVECTOR3 Normal;   sp; // float nx,ny,nz;
 };
```

2.3.3 การเปลี่ยนแปลงวัตถุ 3 มิติโดยใช้ Matrix (3D model transformation)

รูปแบบของการเปลี่ยนแปลงในโลก 3 มิตินั้นสามารถแบ่งการเปลี่ยนแปลงของวัตถุ 3 มิติออกเป็น 3 แบบหลักๆ คือ

- การเคลื่อนย้ายตำแหน่ง (Translation) เป็นการเปลี่ยนแปลงตำแหน่งของวัตถุ 3 มิติเมื่อเทียบกับจุดกำเนิด (Origin) ใน world space
- การหมุน (Rotation) เป็นการหมุนวัตถุ 3 มิติรอบแกนสมมติทั้ง 3 แกน คือการหมุนรอบแกน X (เป็นการหมุนในแนวนอน หน้าไปหลังหรือหลังไปหน้า) การหมุนรอบแกน Y (เป็นการหมุนในแนวตั้ง ซ้ายไปขวาหรือขวาไปซ้าย) และการหมุนรอบแกน Z (เป็นการหมุนในแนวนอนเช่นกันแต่หมุนจาก ซ้ายไปขวาหรือขวาไปซ้าย)
- การปรับขนาด (Scaling) เป็นการปรับขนาดให้แก่วัตถุ3 มิติทั้งในแนวแกน X, Y และ Z โดยค่าน้อยกว่า 1.0f หมายถึงการลดขนาด ในทางกลับกันค่าที่มากกว่า 1.0f ก็จะหมายถึงเป็นการขยายขนาดให้แก่วัตถุ 3 มิติ

2.3.3.1 การปรับขนาดของวัตถุ 3 มิติ

ในการปรับขนาดของวัตถุ 3 มิตินั้นผมได้พูดเกริ่นนำไปแล้วในตอนต้นว่า จะต้องกำหนดค่าทั้งในแกน X, Y และ Z โดยค่านี้เป็นทศนิยมบวกแบบ float โดยค่าน้อยกว่า 1.0f (0.0f – 1.0f) หมายถึงการลดขนาด ในทางกลับกันค่าที่มากกว่า 1.0f จะเป็นการขยายขนาดให้แก่รูปทรง 3 มิติ ที่จริงแล้วการปรับขนาดนั้นมีหลักการง่ายๆ คือเป็นการคูณ ค่าพิกัดในแกนนั้นๆ ของจุด Vertex กับค่าที่ใช้ปรับขนาดในแกนเดียวกัน ซึ่งถ้าเขียนเป็น Code ในภาษา C จะได้ดังนี้

```
for (int i = 0; i< CountOfVertex;i++)
{
        VertexBuffer[i].x = VertexBuffer[i].x * scalingX;
        VertexBuffer[i].y = VertexBuffer[i].y * scalingY;
        VertexBuffer[i].z = VertexBuffer[i].z * scalingZ;
}
```

เมื่อ scalingX, scalingY, และ scalingZ เป็นค่ากำหนดขนาดซึ่งเรากำหนดขึ้นและ VertexBuffer เป็นพอย์เตอร์ที่ชี้ไปยังหน่วยความจำที่เก็บข้อมูลของจุด Vertex เอาไว้ สำหรับฟังก์ชั่นที่ใช้ในการปรับขนาดของวัตถุ 3 มิติใน DirectX คือ

```
D3DXMATRIX * D3DXMatrixScaling(
        D3DXMATRIX * pOut,
        FLOAT sx,
        FLOAT sy,
        FLOAT sz
);
```

pOut คือพอย์เตอร์ที่ชี้ไปยังตัวแปรโครงสร้างข้อมูล D3DXMATRIX เป็นผลลัพธ์ที่ฟังก์ชั่นนี้จะคืนค่ากลับมา

sx , sy และ sz คือค่าที่กำหนดขนาดของวัตถุ 3 มิติทั้งในแนวแกน X , Y และ Z

2.3.3.2 การเคลื่อนย้ายตำแหน่งของวัตถุ 3 มิติ

ในการเคลื่อนย้ายตำแหน่งของวัตถุ 3 มิตินั้นคุณจะต้องกำหนดตำแหน่งใหม่ให้แก่วัตถุทั้งแกน X, Y และ Z ใน World Space โดยปรกติแล้วหากเราไม่ได้กำหนดตำแหน่งให้แก่วัตถุ 3 มิติ DirectX จะเหมาว่าคุณจะให้วัตถุนั้นอยู่ในตำแหน่งจุดกำเนิด (origin) ซึ่งมีค่าพิกัดในแกนทั้ง 3 แกนเป็น 0.0f (จุดกำเนิดคือจุดที่มีค่าแกนทั้ง 3 แกนเป็น 0.0 ซึ่งก็คือจุดกึ่งกลางของ World Space นั้นเอง) ที่จริงแล้วการเคลื่อนย้ายตำแหน่งนั้นมีหลักการง่ายๆ ก็คือเป็นการบวกค่าพิกัดในแกนนั้นๆ ของจุด Vertex กับค่าตำแหน่งใหม่ในแกนเดียวกันซึ่งถ้าเขียนเป็น Code ในภาษา C จะได้ดังนี้

```
for (int i = 0; i< CountOfVertex;i++)
{
        VertexBuffer[i].x = VertexBuffer[i].x + NewPosition.x;
        VertexBuffer[i].y = VertexBuffer[i].y + NewPosition.y;
        VertexBuffer[i].z = VertexBuffer[i].z + NewPosition.z;
}
```

เมื่อ NewPosition เป็นค่าพิกัดของตำแหน่งใหม่ของวัตถุ 3 มิติใน World Space และ VertexBuffe เป็นพอย์เตอร์ที่ชี้ไปยังหน่วยความจำที่เก็บข้อมูลของจุด Vertexเอาไว้ สำหรับฟังก์ชั่นที่ใช้ในการเคลื่อนย้ายตำแหน่งของวัตถุ 3 มิติใน DirectX คือ

```
D3DXMATRIX * D3DXMATRIXTranslation(
        D3DXMATRIX * pOut,
        FLOAT x,    FLOAT y,    FLOAT z
);
```

pOut คือพอย์เตอร์ที่ชี้ไปยังตัวแปรโครงสร้างข้อมูล D3DXMATRIX ซึ่งผลลัพธ์ที่ฟังก์ชั่นที่จะคืนค่ากลับมา

x, y และ z คือตำแหน่งใหม่ของวัตถุ 3 มิติใน World Space

2.3.3.3 การหมุนของวัตถุ 3 มิติ

การหมุนในโลก 3 มิติของ DirectX นั้นมีด้วยกัน 3 แบบคือหมุนในแนวแกน X หมุนในแนวแกน Y และหมุนในแนวแกน Z การหมุนนั้นคุณจะต้องทำการหมุนไปทีละแกน ฟังก์ชั่นที่ใช้ในการคำนวณการหมุนของวัตถุ 3 มิติใน DirectX คือ D3DXMatrixRotationX ซึ่งเป็นการหมุนใน

แนวแกนX D3DXMatrixRotationYเป็นฟังก์ชั่นที่คำนวณการหมุนในแนวแกนY และ D3DXMatrixRotationZ เป็นฟังก์ชั่นที่คำนวณการหมุนในแนวแกน Z ซึ่งพารามิเตอร์ของฟังก์ชั่นทั้ง 3 นี้จะเหมือนกันคือ

```
D3DXMATRIX * D3DXMatrixRotationX(
        D3DXMATRIX * pOut,
        FLOAT Angle
);
```

pOut คือพอย์เตอร์ที่ชี้ไปยังตัวแปรโครงสร้างข้อมูล D3DXMATRIX ซึ่งผลลัพธ์ที่ฟังก์ชั่นที่จะคืนค่ากลับมา

Angle คือค่ามุมที่จะต้องหมุนไปค่าเป็น radians ซึ่งอาจกำหนดเป็นองศาได้โดยใช้สูตรเปลี่ยนจากองศาเป็น radians คือ องศา* ค่าพาย (Pi) / 180 หรืออาจใช้เมธอดจาก D3DX library D3DXToRadian

2.4 หลักการปัญญาประดิษฐ์ที่นำมาใช้ในเกม

2.4.1 ความหมายของปัญญาประดิษฐ์

ปัญญาประดิษฐ์ (Artificial Intelligence) หรือ เอไอ (AI) หมายถึงความฉลาดเทียมที่สร้างขึ้นให้กับสิ่งที่ไม่มีชีวิต ปัญญาประดิษฐ์เป็นสาขาหนึ่งในด้านวิทยาการคอมพิวเตอร์ และวิศวกรรมเป็นหลัก แต่ยังรวมถึงศาตร์ในด้านอื่นๆอย่างจิตวิทยา ปรัชญา หรือชีววิทยา ซึ่งสาขาปัญญาประดิษฐ์เป็นการเรียนรู้เกี่ยวกับกระบวนการการคิด การกระทำ การให้เหตุผล การปรับตัวหรือการอนุมาน และการทำงานของสมอง แม้ว่าดังเดิมนั้นเป็นสาขาหลักในวิทยาการคอมพิวเตอร์แต่แนวคิดหลายๆ อย่างในศาสตร์นี้ได้มาจากการปรับปรุงเพิ่มเติมจากศาสตร์อื่นๆ

ปัญญาประดิษฐ์จะประกอบด้วย Performance, Environment, Actuators, Sensors หรือ P.E.A.S. โดยแต่ละอย่างมีความสำคัญดังนี้

- Performance เป็นการวัดระดับประสิทธิภาพของสิ่งที่ปัญญาประดิษฐ์ได้ทำลงไป
- Environment สิ่งแวดล้อมที่ปัญญาประดิษฐ์ใช้เป็นข้อมูลในการตัดสินใจ
- Actuators เป็นส่วนที่ปัญญาประดิษฐ์ใช้ในการกระทำหรือแสดงออก
- Sensors คือสิ่งที่ปัญญาประดิษฐ์ใช้เพื่อการรับข้อมูลเข้ามาเพื่อตัดสินใจ

นิยามของปัญญาประดิษฐ์มากมาย ซึ่งสามารถจัดแบ่งออกเป็น 4 ประเภท โดยมองใน 2 มิติ ได้แก่

- ระหว่าง นิยามที่เน้นระบบที่เลียนแบบมนุษย์ กับ นิยามที่เน้นระบบที่ระบบที่มีเหตุผล (แต่ไม่จำเป็นต้องเหมือนมนุษย์)
- ระหว่าง นิยามที่เน้นความคิดเป็นหลัก กับ นิยามที่เน้นการกระทำเป็นหลัก

ปัจจุบันงานวิจัยหลักๆ ของ AI จะมีแนวคิดในรูปที่เน้นเหตุผลเป็นหลัก เนื่องจากการนำ AI ไปประยุกต์ใช้แก้ปัญหา ไม่จำเป็นต้องอาศัยอารมณ์หรือความรู้สึกของมนุษย์ อย่างไรก็ตาม นิยามทั้ง 4 ไม่ได้ต่างกันโดยสมบูรณ์ นิยามทั้ง 4 ต่างก็มีส่วนร่วมที่คาบเกี่ยวกันอยู่
นิยามดังกล่าวคือ

1. **ระบบที่คิดเหมือนมนุษย์** (Systems that think like humans)
    1. AI คือ ความพยายามใหม่อันน่าตื่นเต้นที่จะทำให้คอมพิวเตอร์คิดได้ ... เครื่องจักรที่มีสติปัญญาอย่างครบถ้วนและแท้จริง
    2. AI คือ กลไกของ]กิจกรรมที่เกี่ยวข้องกับความคิดมนุษย์ เช่น การตัดสินใจ การแก้ปัญหา การเรียนรู้

2. **ระบบที่กระทำเหมือนมนุษย์** (Systems that act like humans)
    1. AI คือ วิชาของการสร้างเครื่องจักรที่ทำงานในสิ่งซึ่งอาศัยปัญญาเมื่อกระทำโดยมนุษย์
    2. AI คือ การศึกษาวิธีทำให้คอมพิวเตอร์กระทำในสิ่งที่มนุษย์ทำได้ดีกว่าในขณะนั้น

3. **ระบบที่คิดอย่างมีเหตุผล** (Systems that think rationally)
    1. AI คือ การศึกษาความสามารถในด้านสติปัญญาโดยการใช้โมเดลการคำนวณ
    2. AI คือ การศึกษาวิธีการคำนวณที่สามารถรับรู้ ใช้เหตุผล และกระทำ

4. **ระบบที่กระทำอย่างมีเหตุผล** (Systems that act rationally)
    1. ปัญญาประดิษฐ์คือการศึกษาเพื่อออกแบบเอเจนต์ที่มีปัญญา
    2. AI เกี่ยวข้องกับพฤติกรรมที่แสดงปัญญาในสิ่งที่มนุษย์สร้างขึ้น

2.4.2 ระบบปัญญาประดิษฐ์ที่ใช้ในเกม

ระบบปัญญาประดิษฐ์ที่ใช้ในเกมจะเป็นการตัดสินใจ และแก้ไขปัญหาสำหรับการต่อสู้ในขณะนั้นๆ มีการออกแบบตัว Agent ที่ตอบสนองภายใต้เงื่อนไขที่กำหนดไว้ ซึ่งเรียกว่า การตอบสนองแบบ Simple reflex agent เป็นระบบที่ไม่มีการวางแผนหรือกำหนดจุดมุ่งหมายไว้ แต่จะ

สามารถตอบสนองต่อสภาพแวดล้อมในขณะนั้น ว่าควรจะกระทำเช่นใดต่อสถาณการณ์นั้น เพื่อให้ได้ผลที่ดีต่อตัว Agent เอง สำหรับรูปแบบฟังก์ชั่นการทำงานของ Agent นั้นแสดงได้ดังนี้

Function SIMPLE-REFLEX-AGENT(*percept*) return *action*

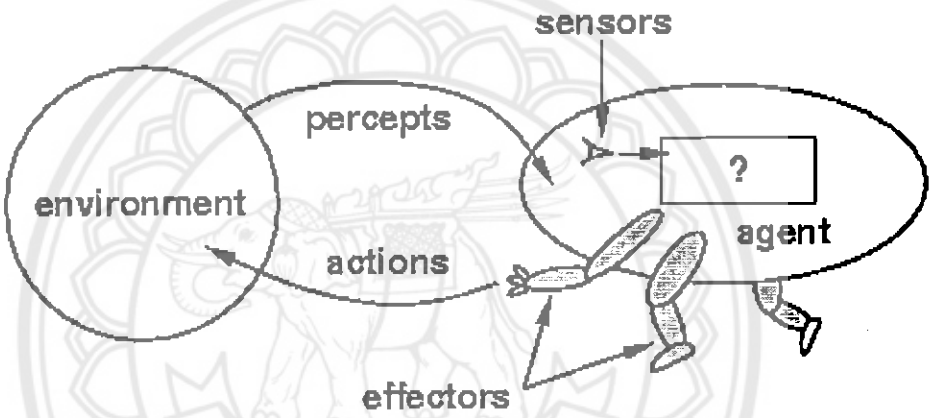
Static: *rules*, a set of condition-actions rules

State <- INTERPRET-INPUT(*percept*)

Rules <- RULE-MATCH(*state*, *rules*)

Action <- RULE-ACTION[*rule*]

Return *action*



รูปที่ 2.10 แสดงระบบ Simple reflex agent

บทที่ 3
วิธีการดำเนินการ

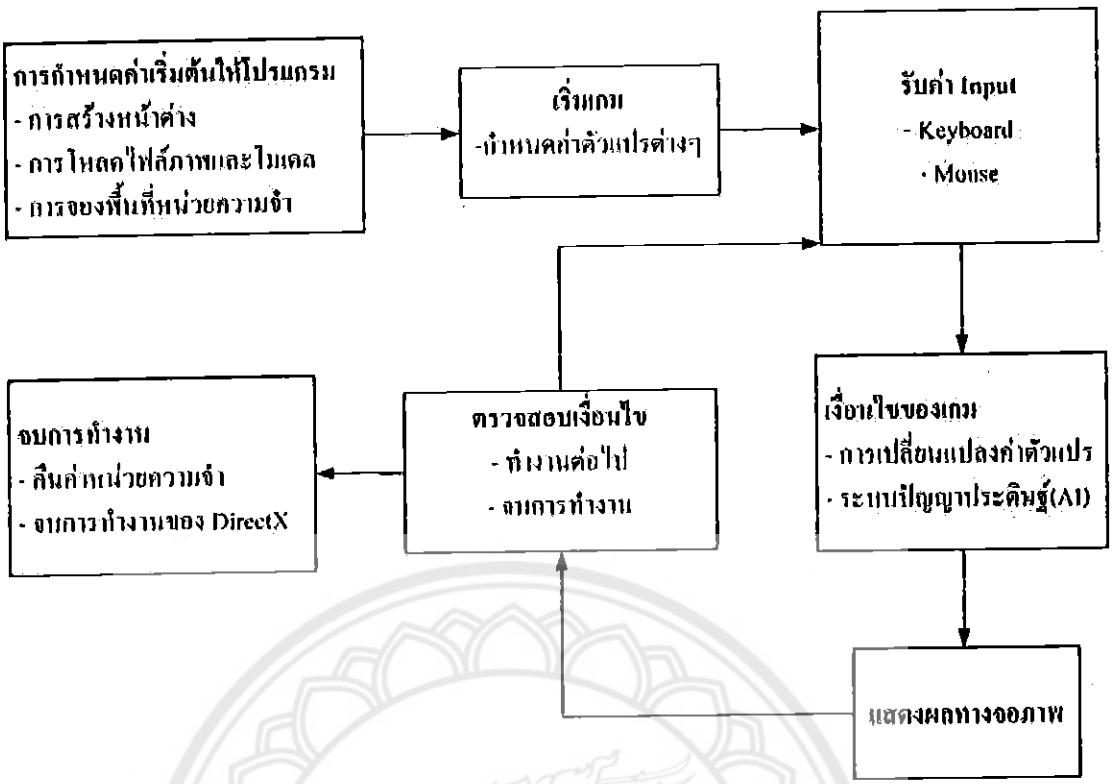
โครงงานนี้แบ่งขั้นตอนการพัฒนาเกมออกเป็น 4 ส่วนคือ

1. ขั้นตอนการศึกษาวิธีการสร้างเกม
2. ขั้นตอนการออกแบบโมเดลที่จะใช้ในเกม
3. ขั้นตอนการออกแบบเงื่อนไขต่างๆ ของเกม
4. ขั้นตอนการเขียนโปรแกรมเพื่อให้เกมเป็นไปตามเงื่อนไขที่ได้ออกแบบไว้

3.1 การศึกษาวิธีการสร้างเกม

หลักการทำงานโดยทั่วๆไปของเกมที่มีในปัจจุบันนั้น จะเป็นลักษณะของลูปโปรแกรม ซึ่งแท้ที่จริงแล้วเกมก็คือโปรแกรมที่จัดการกับเหตุการณ์ต่างๆ ที่เกิดขึ้นตามที่กำหนดไว้นั่นเอง โดยทั่วๆไปแล้ว โปรแกรมเกมมีขั้นตอนในการทำงานอยู่ 7 ขั้นตอน คือ

1. **Initialize** ส่วนเริ่มต้นโปรแกรม ส่วนนี้จะทำการสร้างวินโดว์ ตั้งค่า DirectX โหลดรูปภาพและอนิเมชั่น รวมถึงการจองหน่วยความจำ
2. **Start** ส่วนเริ่มต้นเกมส์ ส่วนนี้เป็นพื้นฐานการตั้งค่าการเล่นเกมส์ เป็นส่วนที่เลือกแผนที่ เลือกสถานที่ของตัวละคร หรือ สุ่มค่าต่าง ๆ หลังจากนี้ไป คุณจะเข้าสู่ลูปของเกมส์
3. **Input** ส่วนรับค่า เป็นส่วนรับข้อมูลจากคีย์บอร์ด เมาส์ จอยสติ๊ก หรือ อุปกรณ์อื่น ๆ ขั้นตอนนี้จะอยู่ในส่วนของ Direct Input
4. **Logic and AI** ส่วนคำนวณ ส่วนนี้จะทำการคำนวณสิ่งที่เกิดขึ้นในโลกของเกมส์ หรืออื่น ๆ ที่จะถูกกำหนดในขั้นตอนนี้
5. **Render** ส่วนแสดงผล ส่วนนี้จะทำการเรนเดอร์ เป็นส่วนที่ DirectX จะถูกใช้งานมากที่สุดในการประมวลผล 3D (รวมถึง 2D) และแสดงผล
6. **Restart** ให้โปรแกรมทำงานต่อไป หรือ ให้โปรแกรมจบการทำงานจบการทำงาน
7. **Cleanup** ส่วนจัดการสุดท้าย ที่จะทำหน้าที่คืนหน่วยความจำ เกมส์ทั้งหมดจะจบในขั้นตอนนี้



รูปที่ 3.1 แสดงหลักการทำงานของเกมโดยทั่วๆไป

3.2 การออกแบบโมเดลที่จะใช้ในเกม

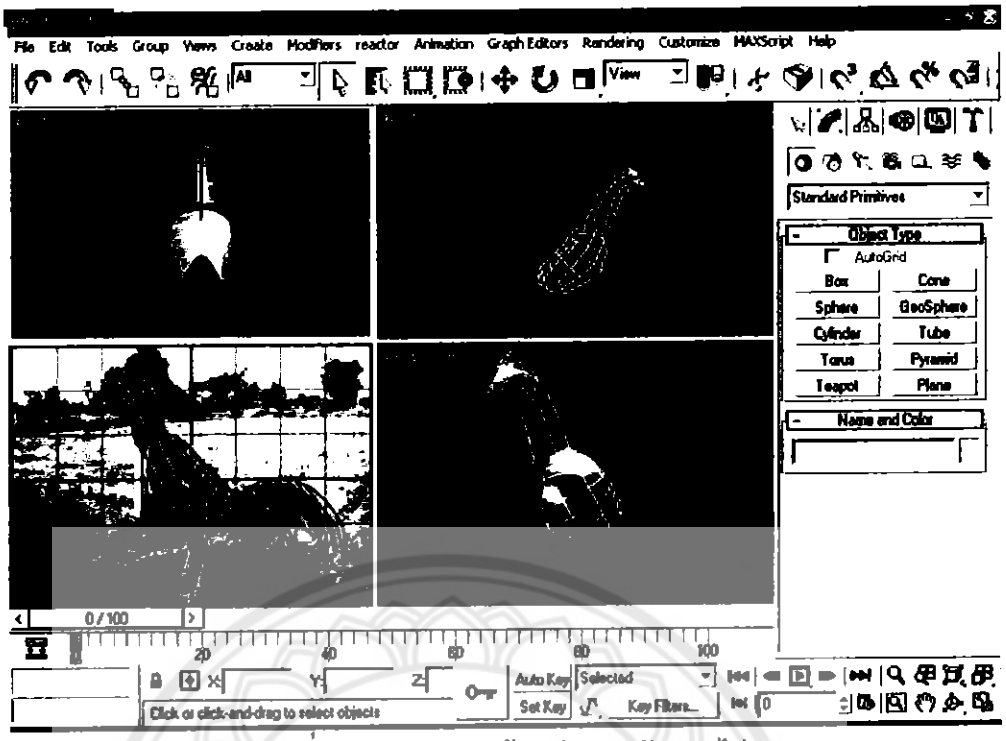
โมเดลที่ใช้ในเกมนี้จะแบ่งเป็น 2 ส่วนดังนี้

1. โมเดลที่เป็นตัวละครไก่
2. โมเดลที่ใช้เป็นฉากสำหรับการต่อสู้

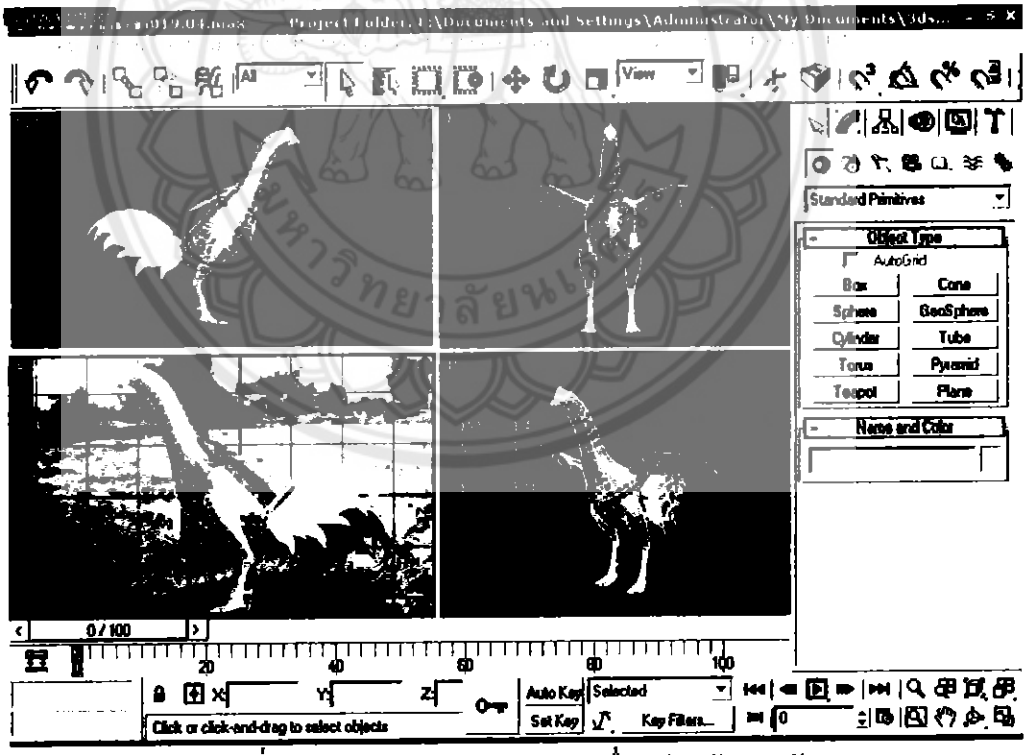
ซึ่งโมเดลเหล่านี้สร้างโดยโปรแกรมสร้างโมเดล 3 มิติ คือ 3DsMax 9

3.2.1 โมเดลที่เป็นตัวละครไก่ (Building Cock model)

การสร้างโมเดลไก่นั้นจะเริ่มจากการสร้าง polygon แบบ Box แล้วทำการแก้ไขรูปทรงให้เหมือนไก่ด้วยการ เพิ่มเติมเส้น edege และจุด vertex ในอวัยวะหลักๆก่อน โดยการแยกเป็นส่วนๆ แล้วนำมาเชื่อมต่อกัน ตามรูปที่ 3.2 และรูปที่ 3.3



รูปที่ 3.2 แสดงการสร้างอวัยวะหลักของไก่

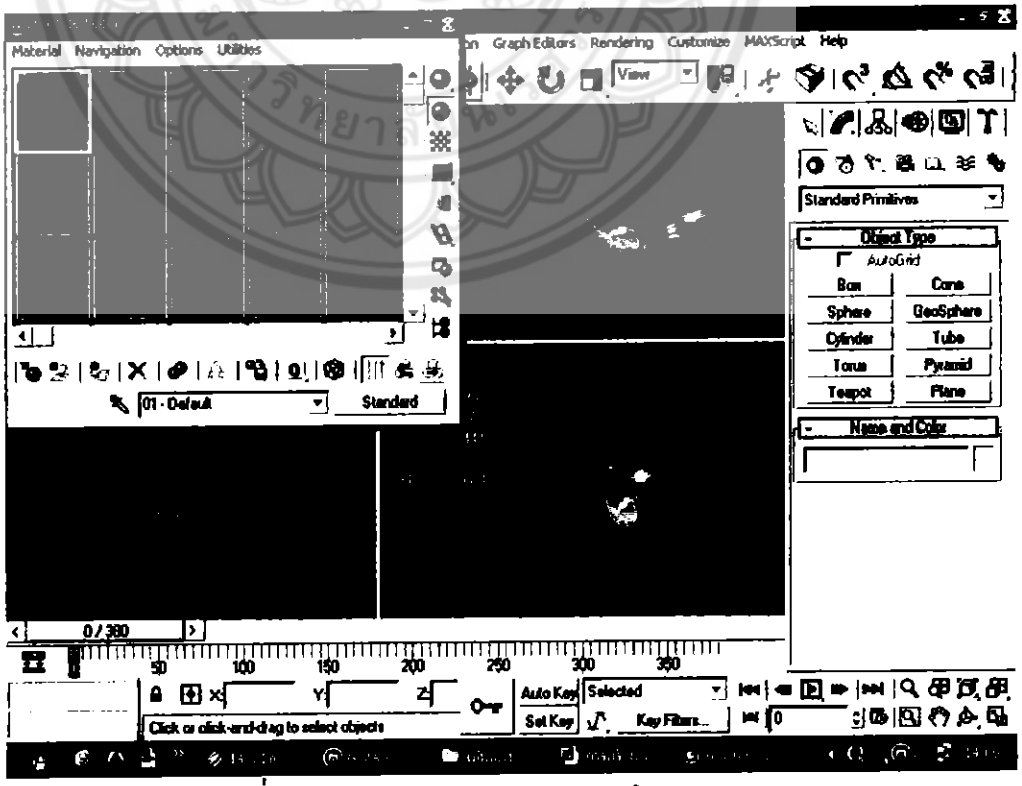


รูปที่ 3.3 แสดงโมเดลหลังจากการเชื่อมต่ออวัยวะหลัก

สร้าง texture เพื่อใช้เป็นลวดลายของผิวหนังโมเดลไก่ ด้วยโปรแกรม Photoshop

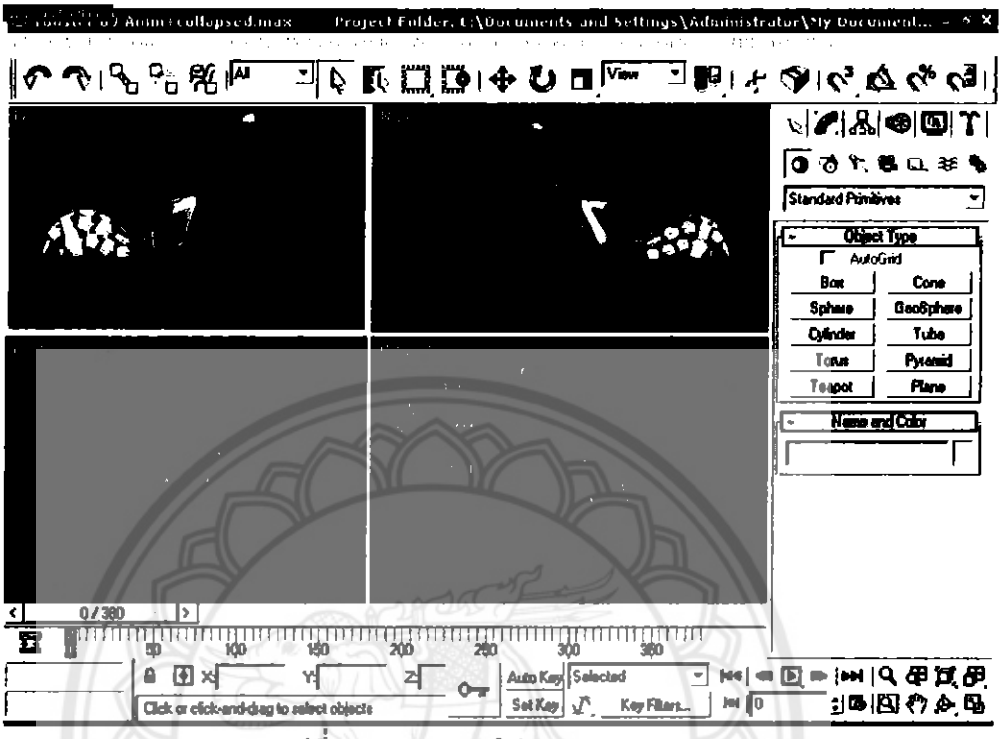
รูปที่ 3.4 แสดงพื้นผิวที่สร้างจากโปรแกรม Photoshop

หลังจากเตรียม texture เรียบร้อยแล้ว เรียกหน้าต่าง material editer ออกมาแล้วใส่ผิวแก่โมเดล หลังจากนั้นให้ทำการ mapping ลวดลายเพื่อให้วางถูกตำแหน่งด้วยเครื่องมือ Unwarp UVW

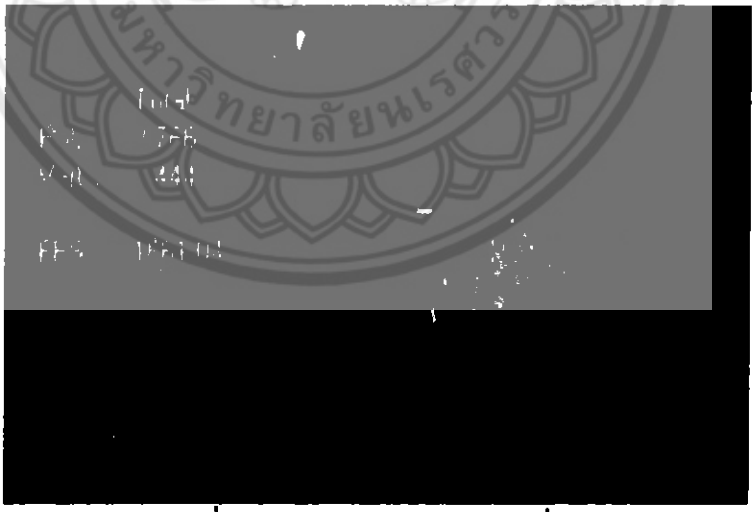


รูปที่ 3.5 แสดงการใส่ลวดลายลงบนโมเดล

สร้างโครงกระดูกเพื่อใช้เป็นแกนสำหรับการสร้าง Animation ด้วยการดัดแปลง Biped ที่เป็นโครงสร้างของสัตว์ 4 เท้า และเพิ่มเติมรายละเอียดด้วย Bone ดังรูปที่ 3.6 และรูปที่ 3.7

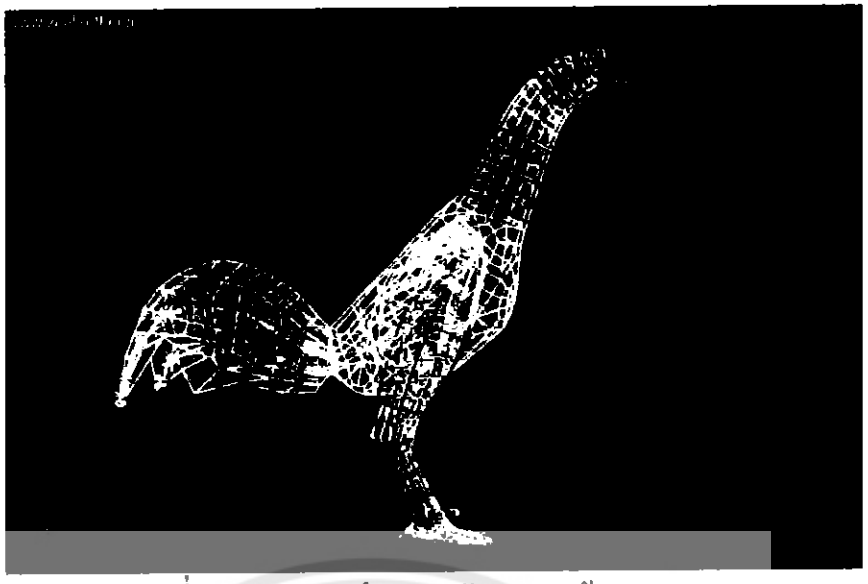


รูปที่ 3.6 แสดงการใส่ Bone และ Biped



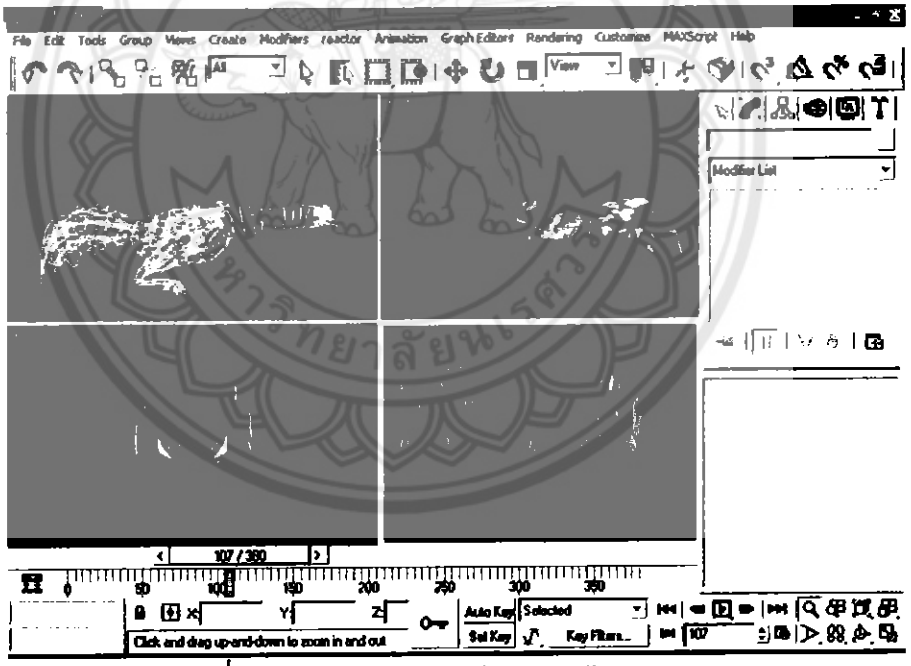
รูปที่ 3.7 แสดง bone และ biped ที่ใช้

ทำการยึดโครงสร้างของ Biped/Bone กับ Vertex ด้วยคำสั่ง Envelope ซึ่งอยู่ภายใต้คำสั่ง Physique ในการยึดโครงสร้างนั้นต้องควบคุมเส้นรัศมีของ envelope ให้ครอบคลุมส่วนที่ต้องการยึด และระวังไม่ให้มีการยึดกับส่วนที่ไม่ต้องการ



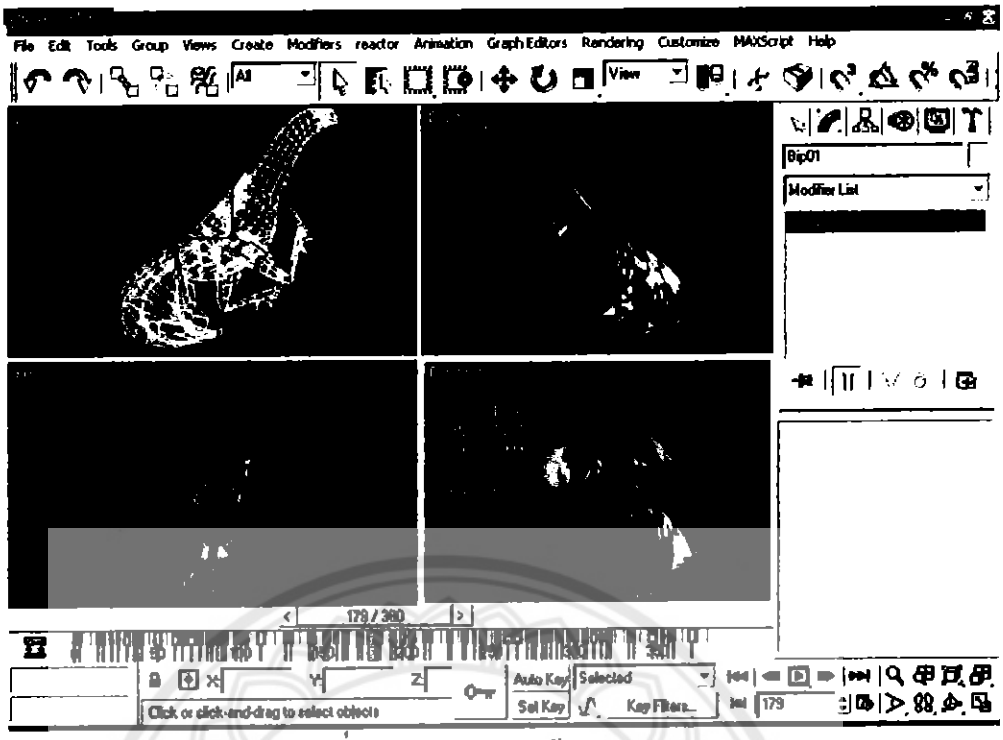
รูปที่ 3.8 แสดงการยึด bone กับ vertex ด้วย envelope

เมื่อทำการยึดโครงสร้างเสร็จให้ทดสอบด้วยการขยับ Biped/Bone ต่างๆ ดังรูปที่ 3.9



รูปที่ 3.9 แสดงผลจากการยึดโมเดลด้วย envelope

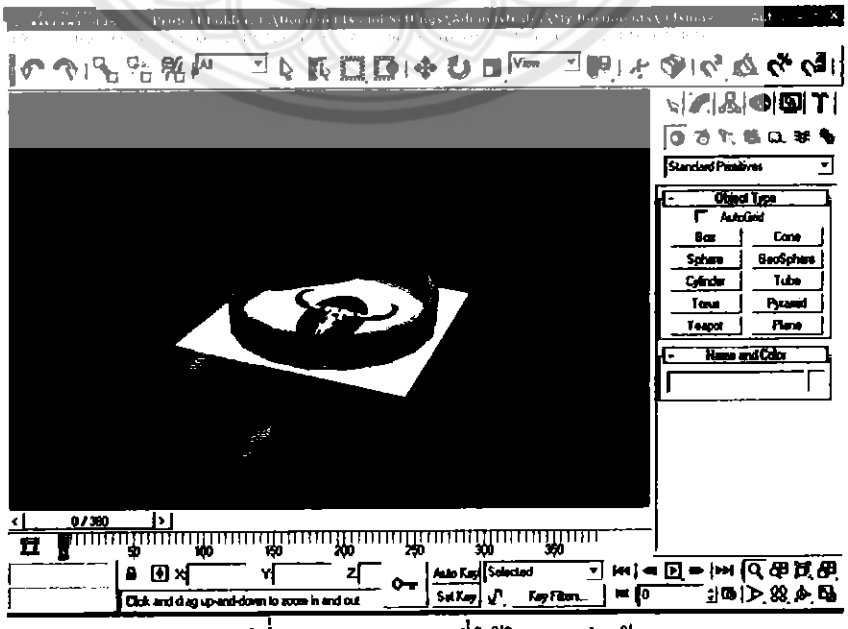
เริ่มทำการสร้าง Animation แบบ frame ต่อ frame ด้วยการขยับส่วนต่างๆ ในท่วงท่าที่เป็นธรรมชาติมากที่สุด ให้แบ่งท่าทางของโมเดลเป็นส่วนๆ เพื่อง่ายต่อการตั้งค่าสำหรับการ Export files ไปเป็น .X files แล้วเรียกใช้ทีละส่วนในขั้นตอนการแสดง animation ในตัวเกม ดังรูปที่ 3.10



รูปที่ 3.10 แสดงการสร้าง Animation

3.2.2 การสร้างฉากสำหรับการต่อสู้ (Building Environment)

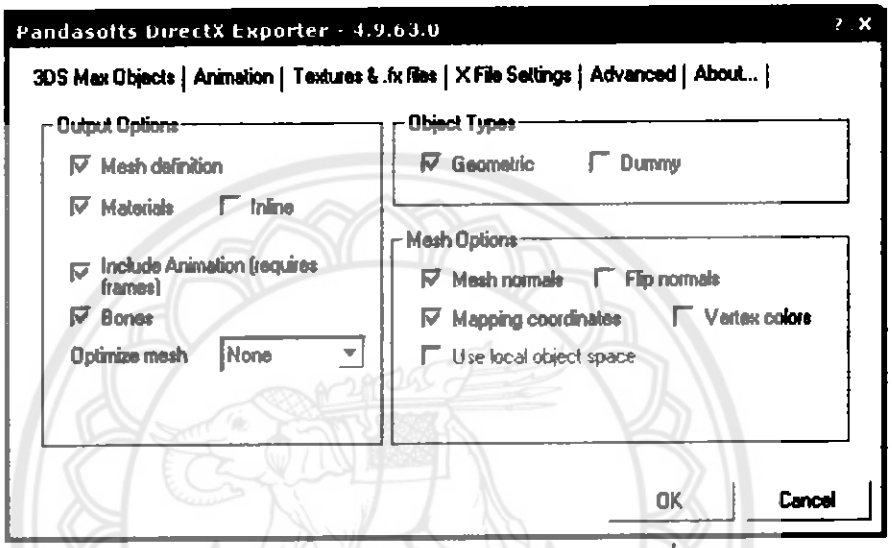
ในการสร้างฉากสำหรับการต่อสู้นั้นมีขั้นเหมือนกับการสร้างโมเดลไก่เพียงแต่ไม่มีการกำหนด Animation เท่านั้น แต่ยังมีการ Export files ออกมาในรูปของ .X อยู่ เพื่อเรียกใช้ในเกมได้ ในขั้นตอนนี้จะมีส่วนของ ท้องฟ้าที่ต้องทำการ Flip ให้พื้นผิวกลับเข้าไปอยู่ด้านในของทรงกลม เพื่อทำโลกเสมือน



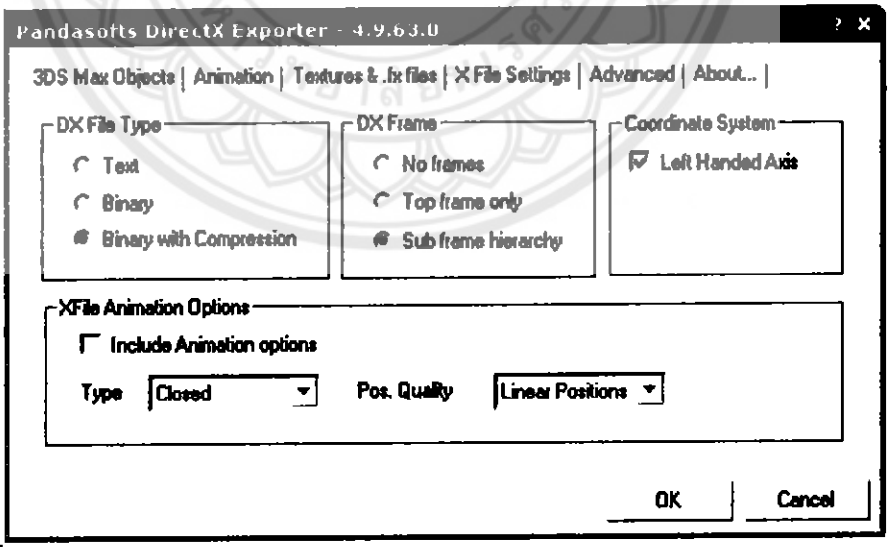
รูปที่ 3.11 แสดงฉากที่ใช้ในการต่อสู้

3.2.3 การ Export files ในรูป .X (X files Exporting)

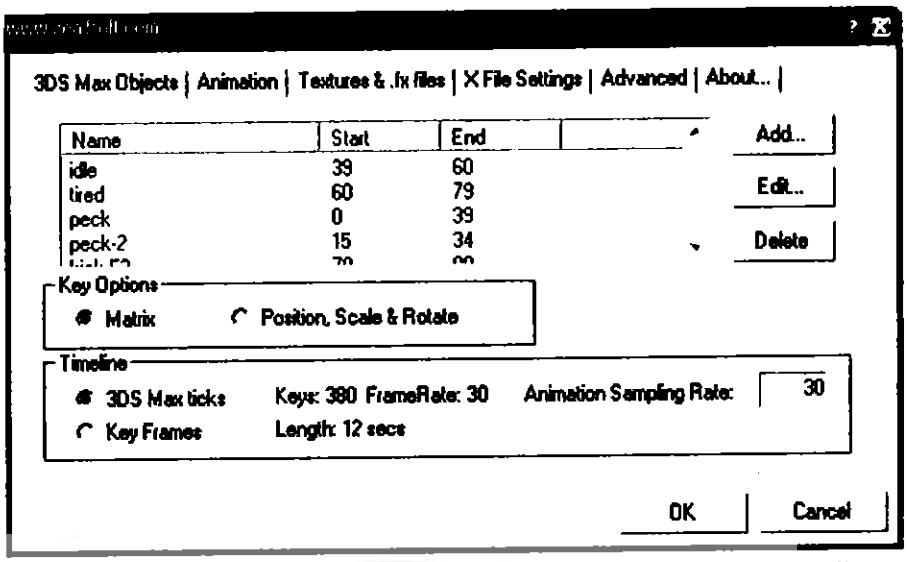
สำหรับการ Export files ในรูปของ .X เพื่อสำหรับการเรียกใช้ในเกม ต้องทำการกำหนดให้มีการ Export Animation ด้วย พร้อมทั้งกำหนดขอบเขตของ Animation เพื่อที่จะได้เรียกใช้เป็นท่าทางต่างๆ ที่ได้สร้างไว้ในโปรแกรม 3DsMax และให้กำหนด Coordinate System แบบ Left hand axis เพราะใน DirectX นั้นได้ใช้ระบบ Coordinate แบบ Left hand axis ดังรูปที่ 3.12, รูปที่ 3.13 และรูปที่ 3.14 ตามลำดับ



รูปที่ 3.12 แสดงการ Export files ด้วย PandaX ในส่วนของการเพิ่ม Animation ใน X files

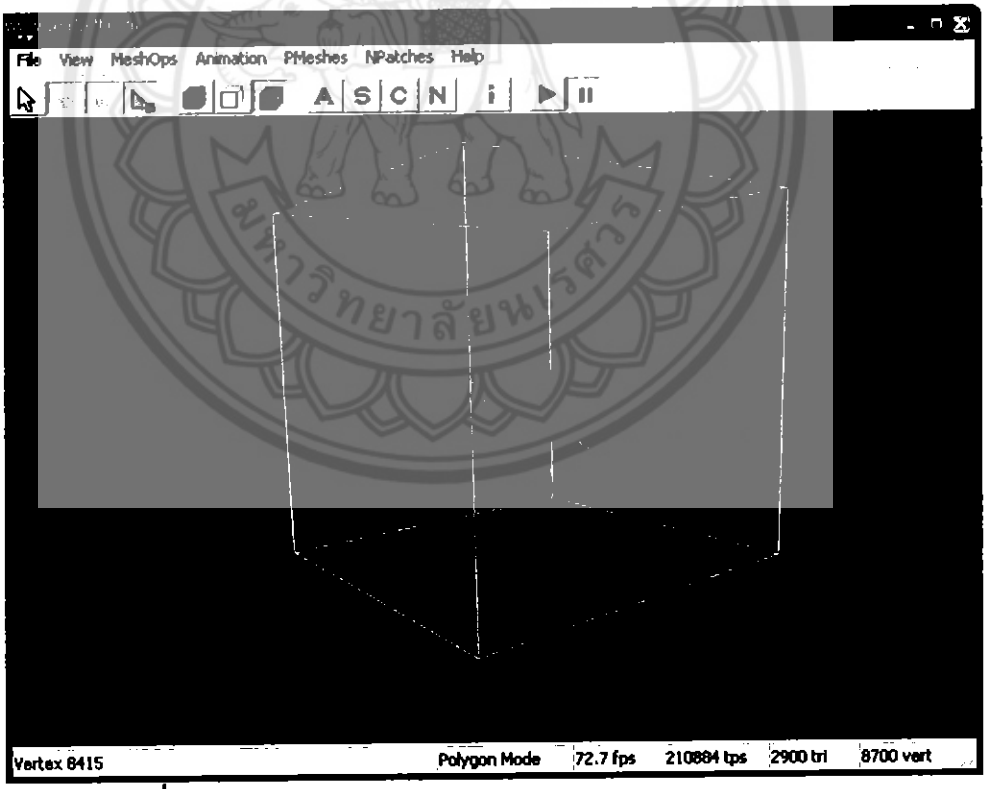


รูปที่ 3.13 แสดงการ Export files ด้วย PandaX ในส่วนของการกำหนดระบบ Coordinate



รูปที่ 3.14 แสดงการ export files ด้วย PandaX ในส่วนของการกำหนด frame สำหรับ Animation

หลังจากการนั้นให้ทำการตรวจสอบความถูกต้องของโมเดลด้วยโปรแกรม MeshViewer ที่มาพร้อมกับการติดตั้ง DirectX SDK ดังรูปที่ 3.15

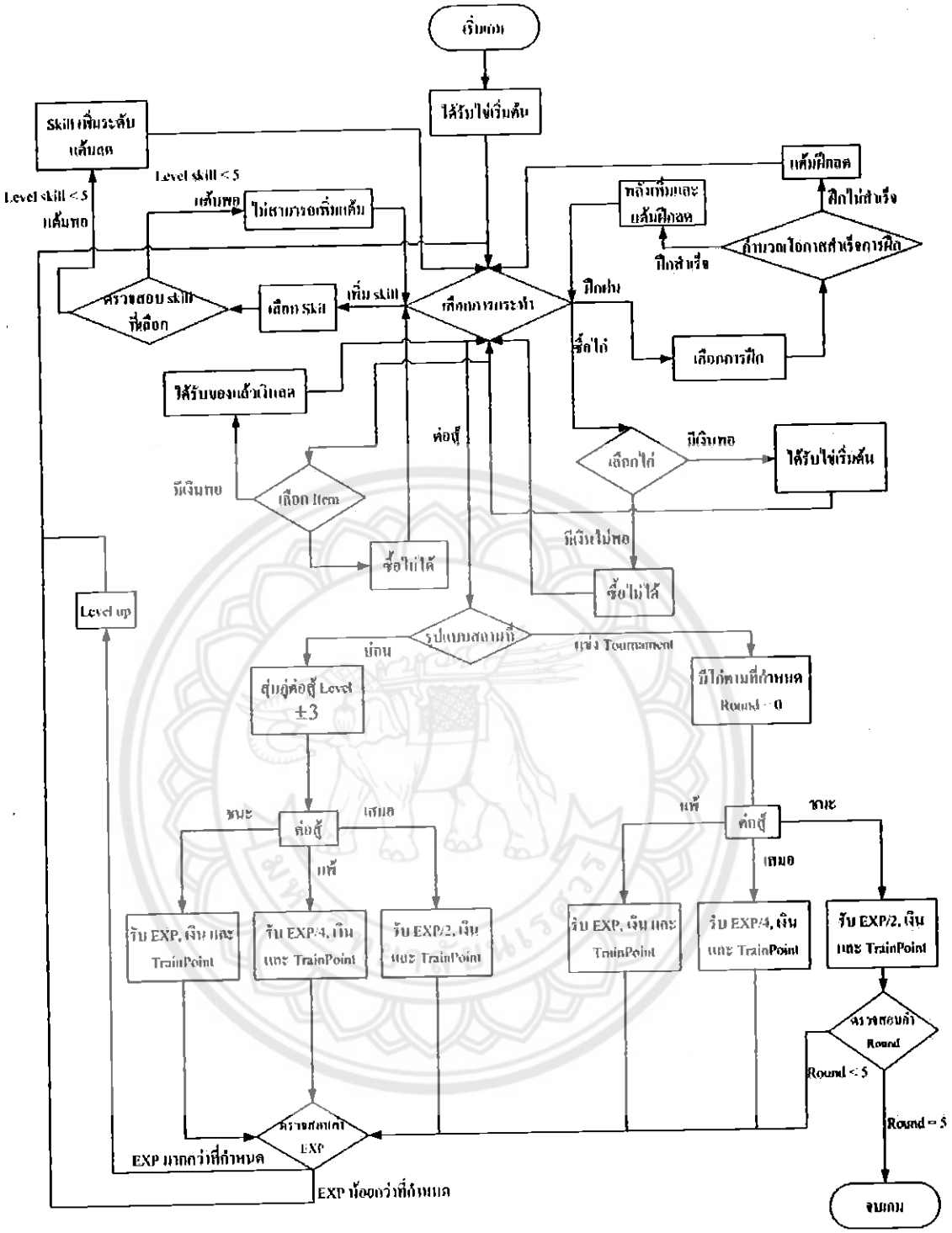


รูปที่ 3.15 แสดงการตรวจสอบโมเดล .x จากโปรแกรม meshViewer

3.3 การออกแบบเงื่อนไขต่างๆ ของเกม

ขั้นตอนนี้จะเป็นขั้นตอนที่สำคัญเพราะความสนุกสนานของเกมจะอยู่ตรงที่เงื่อนไขต่างๆ ในการเล่นเกมซึ่งเกมไก่ชนนี้ได้มีการออกแบบรูปแบบหรือเงื่อนไขการเล่นดังนี้

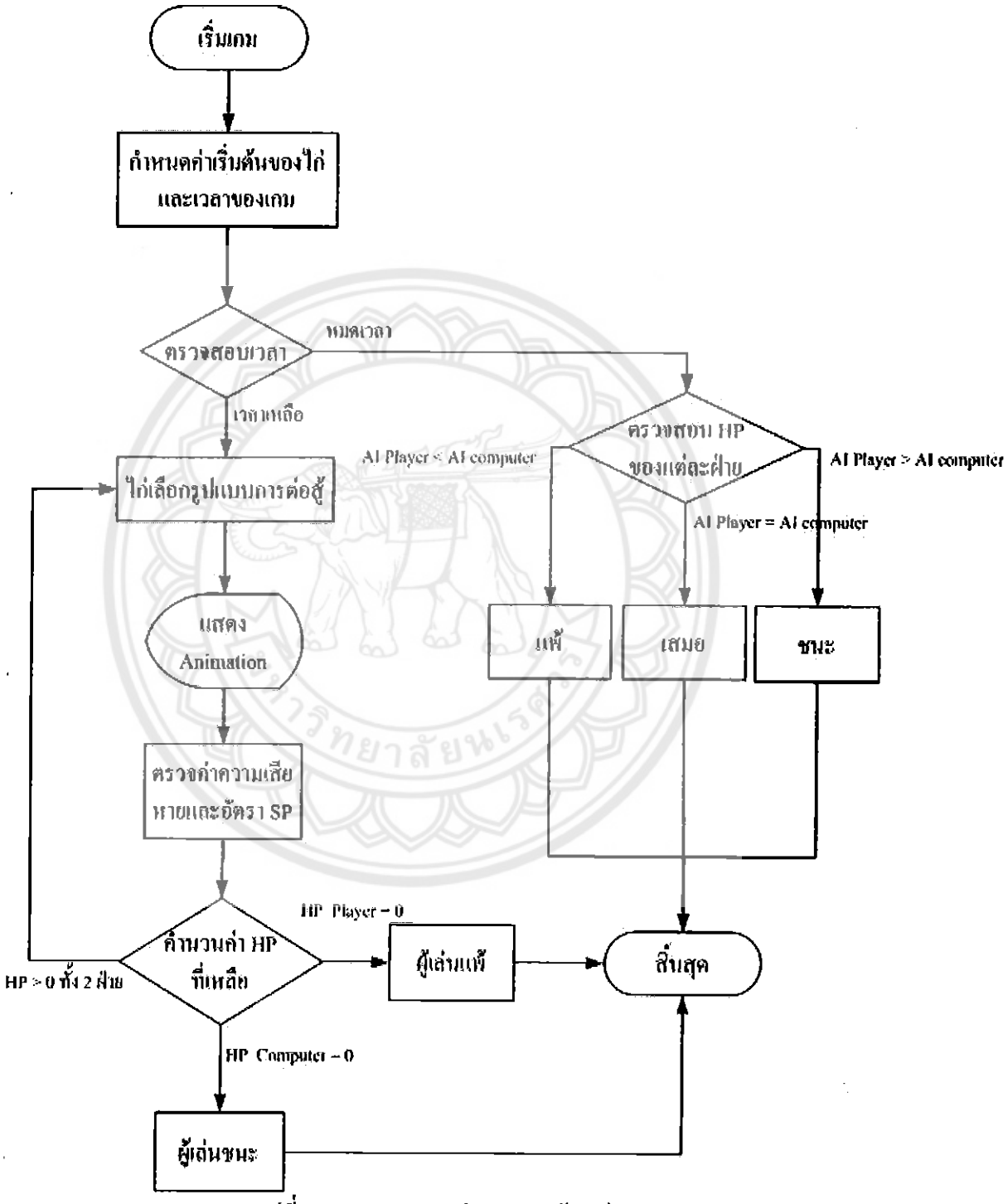
เกมไก่ชนเป็นเกมที่ให้ผู้เล่นสวมบทบาทในการเป็นผู้เลี้ยงไก่ชน โดยให้ผู้เล่นทำการฝึกฝนไก่ของตนเพื่อนำไปชนให้ชนะเลิศในการแข่งขันที่กำหนดไว้ในเกม ผู้เล่นก็จะชนะเกมนี้ โดยในการต่อสู้ของไก่ชนนั้นผู้เล่นจะไม่สามารถบังคับไก่ของตนได้ เกมจะใช้หลักปัญญาประดิษฐ์ในการควบคุมไก่ชนเพื่อที่จะให้ไก่ชนนั้นสู้กัน ซึ่งความได้เปรียบเสียเปรียบนั้นขึ้นอยู่กับการฝึกฝนไก่ชนของผู้เล่น และในเกมยังมีส่วนของการใช้ไอเท็มช่วยเหลือในการฝึกฝนไก่ชนอีกด้วย สามารถอธิบายด้วย Flow Chart ดังรูปที่ 3.16



รูปที่ 3.16 แสดงเงื่อนไขภายในเกม

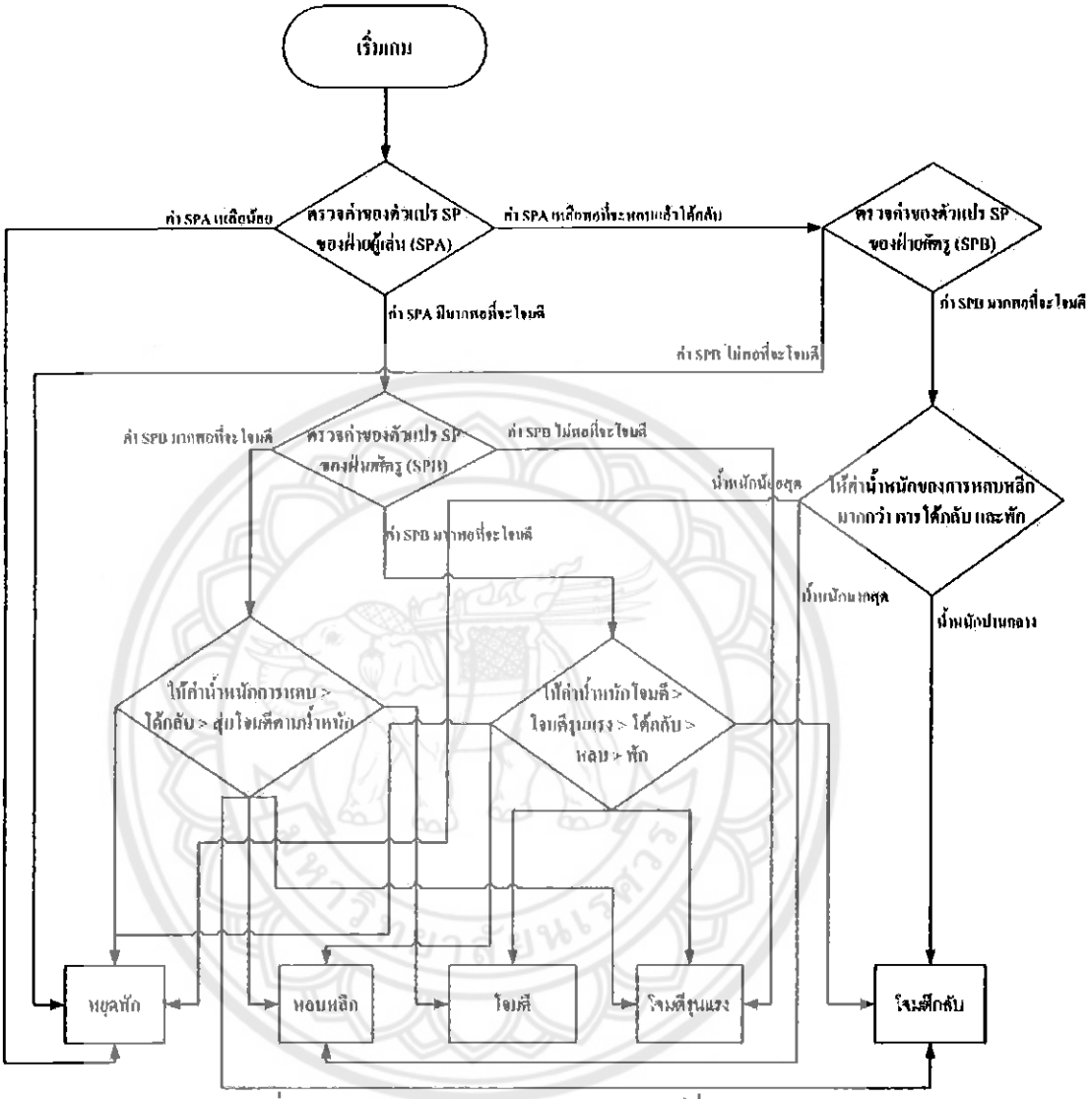
ในการต่อสู้ของไก่ชนนั้นมีกติกาในการต่อสู้ก็คือ ใช้เวลาในการต่อสู้ 60 วินาที ถ้าหากภายในเวลา 60 วินาทีนี้ ฝ่ายใดที่เหลือค่า HP (Health Point) เท่ากับ 0% จะเป็นฝ่ายแพ้ แต่ถ้าหากเวลาหมดลง ฝ่ายที่เหลือค่า HP มากกว่าจะเป็นฝ่ายชนะ แต่ถ้าเหลือเท่ากันจะถือว่าเสมอกัน ซึ่งในการต่อสู้จะมีค่าที่สำคัญอีกค่าหนึ่งก็คือ SP (Stamina Point) โดยถ้าฝ่ายใดที่มีค่า SP เหลือน้อยจะไม่

สามารถแสดงท่าทางได้จะต้องพักเพื่อให้ฟื้นฟูค่า SP ขึ้นมา ในระหว่างที่ฟื้นฟูค่า SP อยู่นี้หากโดนโจมตีจะได้รับความเสียหายมากกว่าปกติ กติกาในการต่อสู้ของไก่ชน สามารถอธิบายด้วย Flow Chart ดังรูปที่ 3.17



รูปที่ 3.17 แสดงกติกาในการต่อสู้ของไก่ชน

การต่อสู้ของไก่ชนนั้น ใช้หลักปัญญาประดิษฐ์สำหรับการตัดสินใจของไก่เพื่อที่จะแสดงท่าทางในการต่อสู้ออกมา โดยวิธีคิดของปัญญาประดิษฐ์ สามารถอธิบายด้วย Flow Chart ดังรูปที่ 3.18



รูปที่ 3.18 แสดงหลักการปัญญาประดิษฐ์ที่ควบคุมไก่ชน

3.4 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้เป็นส่วนที่สำคัยที่สุดในการสร้างเกมเพราะต้องมีการเขียนโปรแกรมเพื่อให้เกมเป็นไปตามเงื่อนไขที่ได้กำหนดไว้ และจะต้องมีการจัดการเกี่ยวกับเรื่องของโมเดล การควบคุมเกม ความผิดพลาดที่อาจเกิดขึ้นระหว่างการทำงานของโปรแกรมและอื่นๆ อีกมาก

ในส่วนการเขียนโปรแกรมจะแบ่งขั้นตอนการพัฒนาเป็นดังนี้

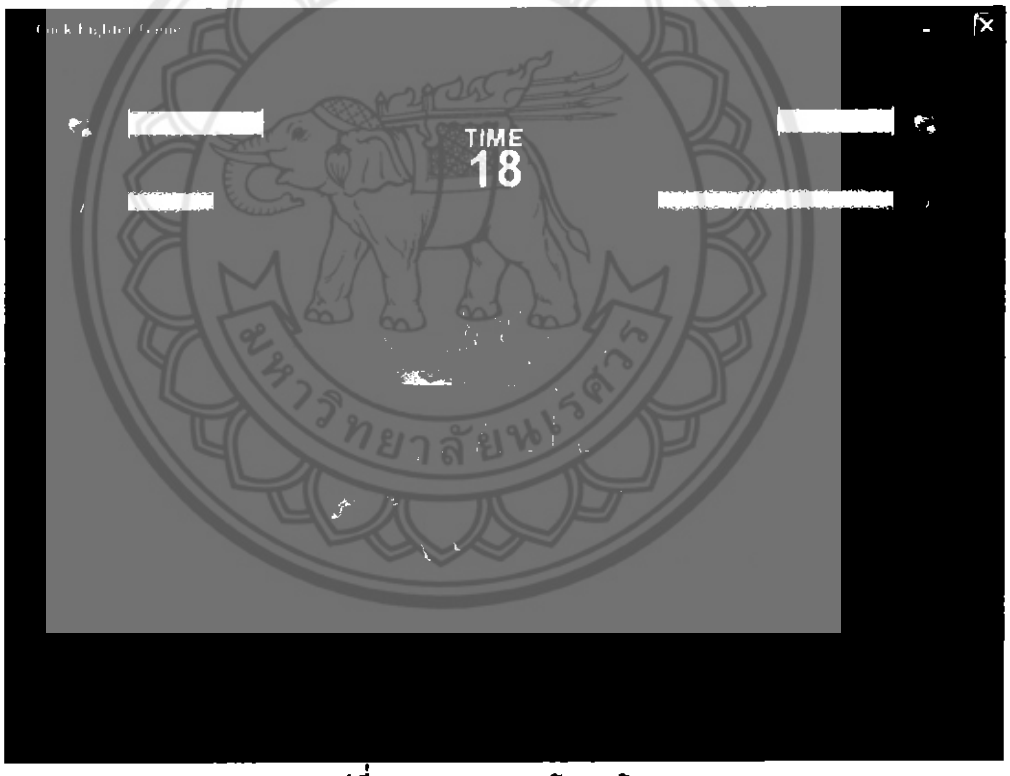
1. การเขียนโปรแกรมเพื่อสร้างหน้าต่างวินโดว์ในการเล่นเกม
2. การเขียนโปรแกรมเพื่อโหลดภาพและโมเดลเข้ามาใช้ในเกม
3. การเขียนโปรแกรมเพื่อแสดงข้อความรูปแบบ 2 มิติ ในเกม 3 มิติ

4. การเขียนโปรแกรมเพื่อควบคุมการเคลื่อนไหวของโมเดล

5. การเขียนโปรแกรมเพื่อควบคุมให้เป็นไปตามเงื่อนไขที่ออกแบบไว้

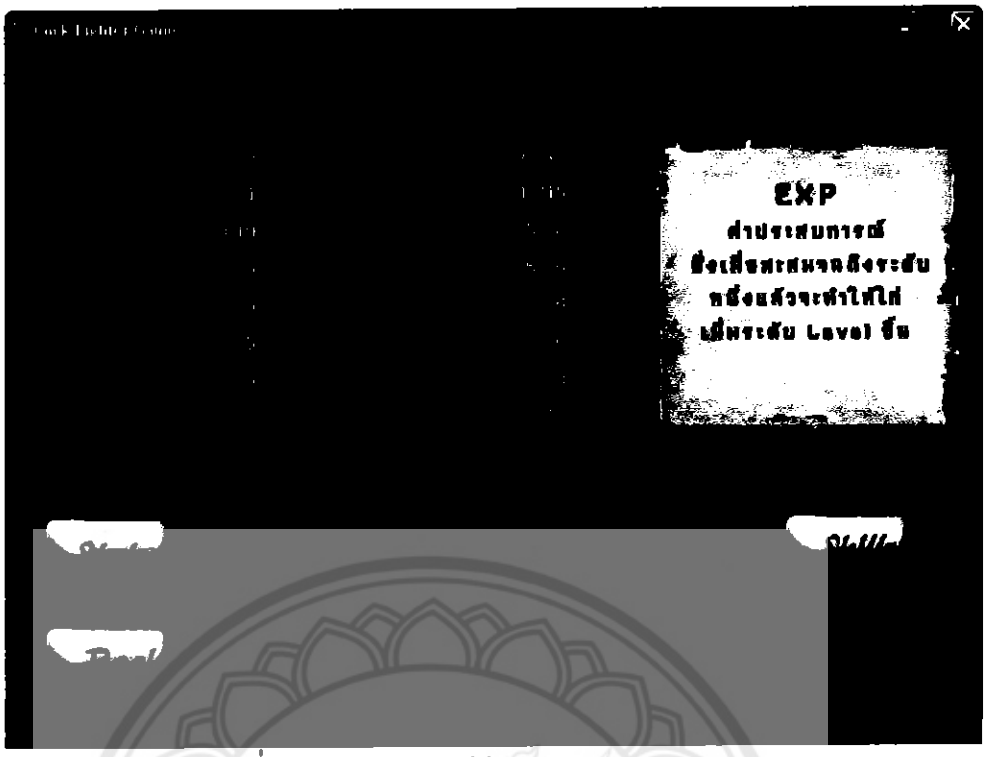
1. การเขียนโปรแกรมเพื่อสร้างหน้าต่างวินโดว์ในการเล่นเกม ในส่วนนี้จะเป็นส่วนที่ใช้ในการสร้างหน้าต่างหลักของเกม ซึ่งจะมีลักษณะเป็นแบบหน้าต่างวินโดว์ ขนาดของหน้าต่างวินโดว์คือ 800 x 600 ไม่ใช่แบบแสดงผลเต็มจอ

2. การเขียนโปรแกรมเพื่อโหลดภาพและโมเดลเข้ามาใช้ในเกม ในส่วนนี้จะเป็นการเขียนโปรแกรมเพื่อโหลดภาพและโมเดลเข้ามาใช้ในเกม ซึ่งภาพจะเป็นภาพแบบ 2 มิติ โดยที่โมเดลจะเป็นแบบ 3 มิติ ซึ่งโมเดลจะมีอยู่ด้วยกัน 2 ลักษณะ คือ โมเดลที่ไม่มีการเคลื่อนไหว และโมเดลที่มีการเคลื่อนไหว ในเกมนี้จะมีการใช้โมเดลทั้ง 2 แบบนี้ในการพัฒนาเกม



รูปที่ 3.19 แสดงการโหลดโมเดล

3. การเขียนโปรแกรมเพื่อแสดงข้อความรูปแบบ 2 มิติ ในเกม 3 มิติ ในขั้นตอนนี้จะเป็นส่วนในการเขียนโปรแกรมเพื่อสร้างตัวอักษรขึ้นเพื่อใช้ในเกม ตัวอักษรเหล่านี้มีความสำคัญเช่นกัน ซึ่งจะช่วยเพิ่มสีสันให้กับเกมทำให้เกมน่าเล่นมากยิ่งขึ้น และยังเป็นส่วนที่ช่วยให้ผู้เล่นสามารถสื่อสารกับเกมได้รู้เรื่อง



รูปที่ 3.20 แสดงการใช้ข้อความ 2 มิติ ในเกม 3 มิติ

4. การเขียนโปรแกรมเพื่อควบคุมการเคลื่อนไหวของโมเดล ในขั้นตอนนี้จะเป็นการเขียนโปรแกรมเพื่อควบคุมการแสดงท่าทางของไก่ชนโดยการใช้หลักการปัญญาประดิษฐ์ในการคิดและการแสดงท่าทางที่เหมาะสมออก

รูปที่ 3.21 แสดงการใช้ระบบปัญญาประดิษฐ์ควบคุมไก่

5. การเขียนโปรแกรมเพื่อควบคุมให้เป็นไปตามเงื่อนไขที่ออกแบบไว้ ขั้นตอนนี้เป็นส่วนของการรวมรวมส่วนประกอบทั้งหมดนำมาใส่เงื่อนไขของเกม และทำให้เกมสามารถที่ดำเนินไปตามรูปแบบที่ได้ออกแบบไว้ได้ และทำให้เกมเสร็จสมบูรณ์

รูปที่ 3.22 แสดงเกมที่เขียนขึ้นตามเงื่อนไขที่ได้ออกแบบไว้

3.5 วิธีการเล่นเกม

1. เริ่มต้นผู้เล่นจะได้รับไก่ระดับชั้น D ซึ่งเป็นไก่ที่มีความสามารถน้อยที่สุด ได้รับเงิน 1000 บาท เพื่อใช้ในการ ซื้อของ
2. ผู้เล่นจะต้องฝึกฝนไก่ให้เก่งขึ้น โดยสามารถฝึกได้ที่หน้า Training โดยจะต้องเสียค่าการฝึกฝนจำนวนหนึ่ง
3. ผู้เล่นสามารถฝึกไก่ให้เรียนรู้ทักษะได้ในหน้า Skills โดยจะต้องเสียค่า Skill point
4. ผู้เล่นสามารถนำไก่ไปชนที่บ่อนเพื่อสะสมค่าประสบการณ์ เงิน ค่าการฝึกฝน เพื่อนำมาพัฒนาไก่ให้เก่งขึ้น
5. ผู้เล่นสามารถซื้อไก่ที่มีระดับชั้นที่ดีกว่าเดิมได้ที่ร้านขายไข่
6. ผู้เล่นสามารถซื้อของเพื่อเพิ่มความสามารถให้กับไก่ได้ที่ร้านขายของ
7. ในการต่อสู้ผู้เล่นจะไม่สามารถควบคุมไก่ได้ ไก่จะสู้กันเอง
8. การจะชนะเกมได้ผู้เล่นจะต้องชนะการแข่งขันใน Tournament รอบสุดท้าย

บทที่ 4
ผลการทดสอบโปรแกรมและวิเคราะห์ผล

4.1 จุดประสงค์ของการทดสอบโปรแกรม

1. เพื่อทำการทดสอบโปรแกรมที่ได้สร้างขึ้นเป็ไปตามเงื่อนไขหรือตามที่ได้ออกแบบไว้หรือไม่
2. เพื่อทดสอบประสิทธิภาพการทำงานของโปรแกรม
3. เพื่อทดสอบหาข้อผิดพลาดที่เกิดขึ้นกับโปรแกรมเพื่อจะได้นำข้อผิดพลาดที่เกิดขึ้น มาวิเคราะห์และหาทางแก้ไขให้โปรแกรมมีความผิดพลาดเกิดขึ้นน้อยที่สุด

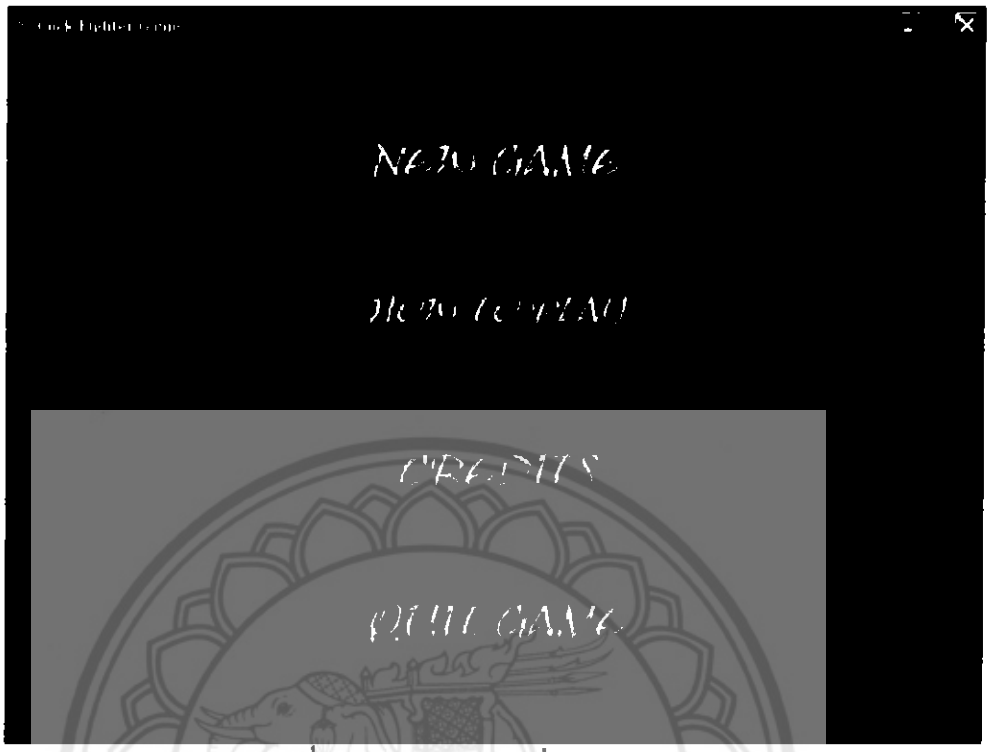
4.2 ขั้นตอนการทดสอบการทำงานของโปรแกรม

1. ติดตั้งโปรแกรม Visual C++ 2005
2. ติดตั้งโปรแกรม DirectX SDK และ DirectX Runtime
3. ตั้งค่าการทำงานเพื่อให้โปรแกรม Visual C++ กับโปรแกรม DirectX SDK สามารถทำงานร่วมกันได้
4. เปิดโปรแกรมเกมเพื่อทดสอบการทำงานของเกมว่าเป็นไปตามเงื่อนไขที่กำหนดไว้หรือไม่
5. ตรวจสอบหาข้อผิดพลาดที่เกิดขึ้นในโปรแกรม
6. นำข้อผิดพลาดที่เกิดขึ้นมาวิเคราะห์หาสาเหตุและทำการปรับปรุงแก้ไขโปรแกรม

4.3 ผลการทดสอบโปรแกรม

โปรแกรมเกมนี้ได้ทำการกำหนดให้มีการควบคุมเกมผ่านเม้าส์เท่านั้น คีย์บอร์ดจะไม่สามารถใช้ในเกมได้

เมื่อเปิดโปรแกรมขึ้นมาจะมีเมนูของเกมให้เลือกเริ่มเล่นเกม หรือออกจากเกม ดังรูปที่ 4.1



รูปที่ 4.1 แสดงเมนูเกมเมื่อเปิดโปรแกรม

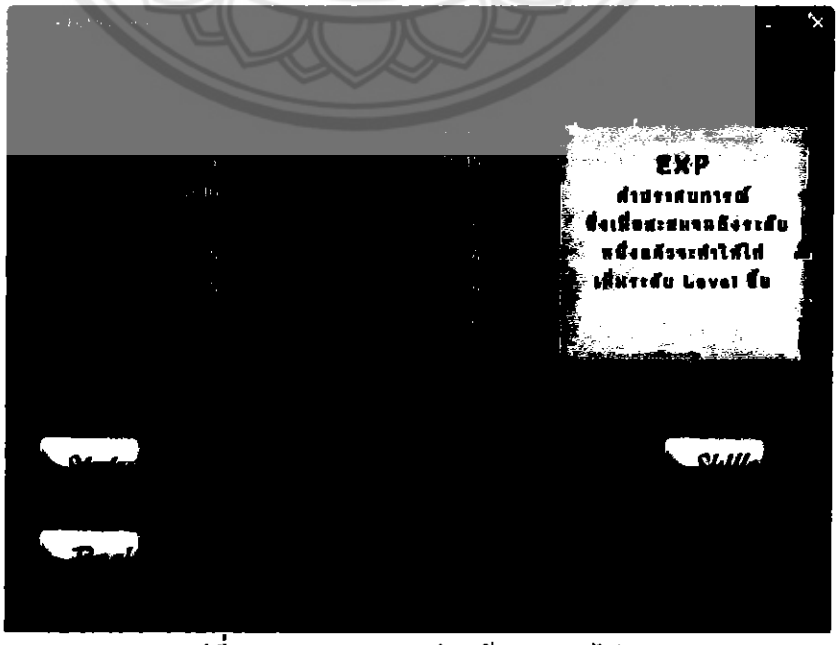
เมื่อเลือกที่เริ่มเกมจะพบกับการเกริ่นนำก่อนเริ่มเกม ดังรูปที่ 4.2

รูปที่ 4.2 แสดงข้อความเกริ่นนำก่อนเริ่มเกม

เมื่อผ่านการเกริ่นนำก่อนเริ่มเกมแล้วจะเข้าสู่หน้าหลักของเกม โดยจะสมมติว่าเป็นฟาร์มไก่ชน โดยจะมีเมนูให้เลือกคือ ดูข้อมูลไก่ชน การฝึกฝน ช่องเก็บไอเท็ม และออกไปยังหมู่บ้านดังรูปที่ 4.3

รูปที่ 4.3 แสดงเมนูหลักในการเล่นเกม

เมื่อเลือกดูข้อมูลไก่ชนในเมนูหลักของเกม จะพบกับการแสดงรายละเอียดข้อมูลของไก่ชน และคำอธิบายค่าสถานะของไก่ชน โดยนำเคอร์เซอร์ของเม้าส์ไปวางไว้บนรูปของสถานะที่ต้องการทราบรายละเอียด รายละเอียดจะปรากฏขึ้นมาบริเวณทางขวาของหน้าต่างเกม ดังรูปที่ 4.4

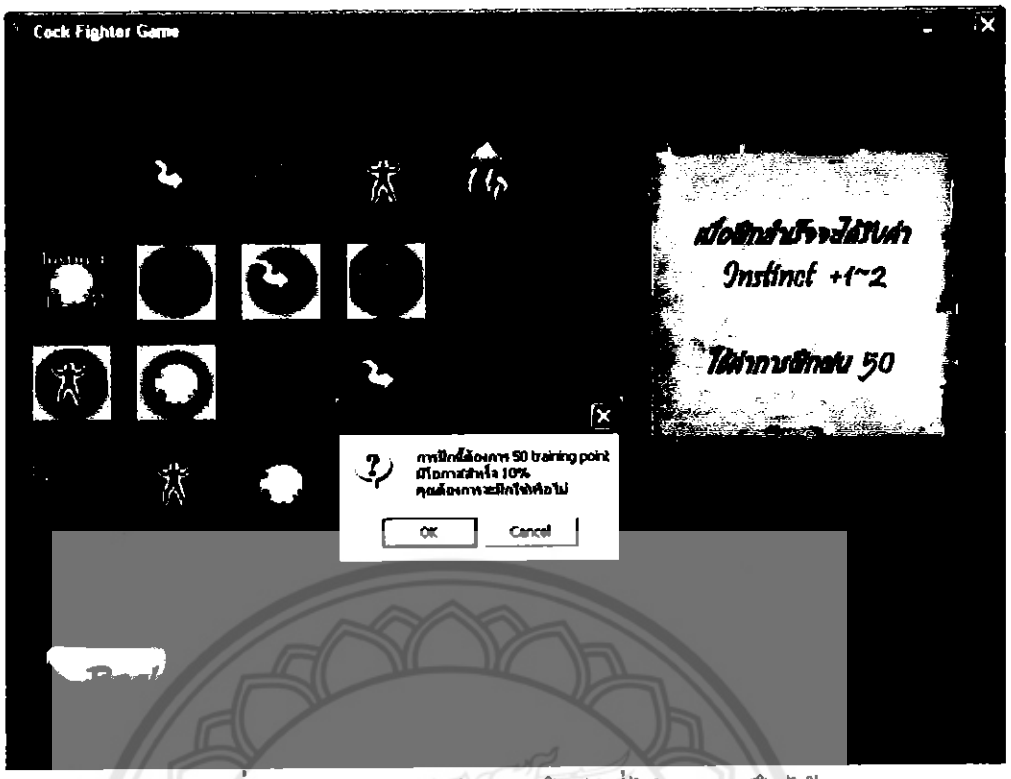


รูปที่ 4.4 แสดงรายละเอียดข้อมูลของไก่ชน

เมื่อเลือกเมนู Skills ที่หน้าจอการแสดงรายละเอียดของไก่ชน จะพบกับรายละเอียดความสามารถต่างๆที่สามารถจะให้ไก่เรียนรู้ได้ โดยสามารถรู้รายละเอียดของความสามารถต่างๆได้โดยนำเคอร์เซอร์ของเม้าส์มาวางไว้บนความสามารถที่ต้องการก็จะมีรายละเอียดแสดงขึ้นมาทางด้านขวาของหน้าจอเกม และสามารถให้ไก่เรียนรู้ความสามารถได้ดดยคลิกเม้าส์ซ้ายที่ความสามารถที่ต้องการ หากมีแต้มความสามารถเหลืออยู่ก็จะสามารถเรียนรู้ความสามารถนั้นได้ แต่หากค่าความสามารถไม่เหลือ หรือความสามารถนั้นได้เรียนรู้ถึงระดับ 5 แล้วก็จะไม่สามารถเรียนรู้ได้ ดังรูปที่ 4.5

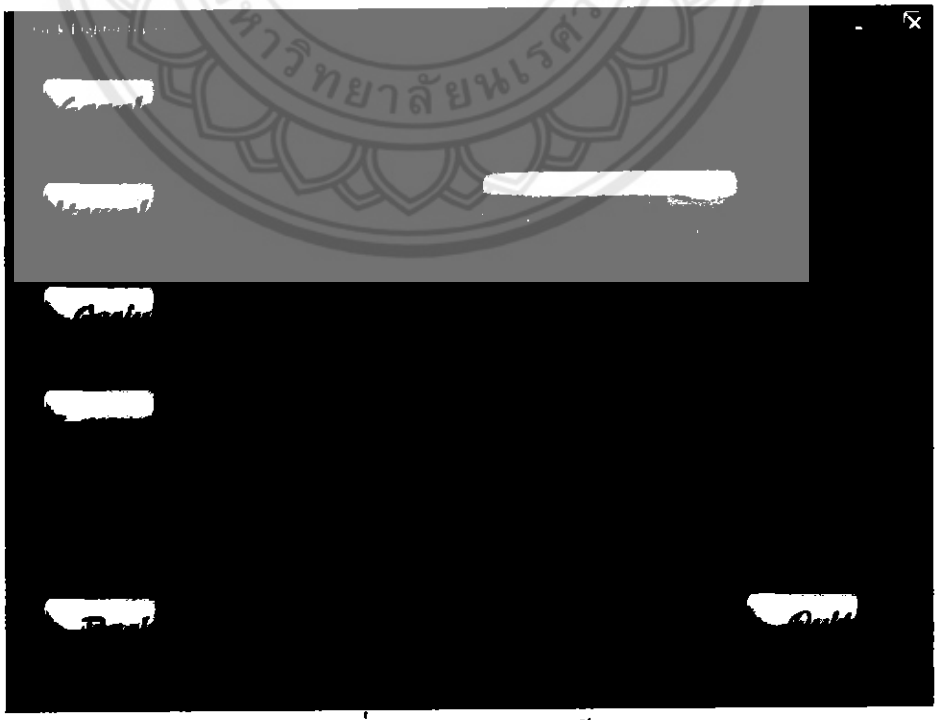
รูปที่ 4.5 แสดงรายละเอียดความสามารถที่ไก่สามารถเรียนรู้ได้

เมื่อเลือกเมนูการฝึกฝนที่หน้าจอหลักของเกม จะพบกับรายละเอียดการฝึกที่ไก่สามารถที่จะฝึกฝนเพื่อเพิ่มค่าสถานะต่างๆได้ โดยในการฝึกนั้นจะบอกโอกาสในการฝึกสำเร็จอยู่ และเมื่อฝึกสำเร็จก็จะได้รับค่าสถานะต่างๆตามที่ระบุในรายละเอียด และเสียค่าการฝึกฝนตามที่การฝึกนั้นต้องการ หากการฝึกล้มเหลวก็จะเสียค่าการฝึกฝนเล็กน้อย หากไก่มีค่าการฝึกฝนไม่เพียงพอ ก็จะไม่สามารถฝึกได้ โดยรายละเอียดของแต่ละการฝึกฝนสามารถดูได้โดยการนำเคอร์เซอร์ของเม้าส์วางไว้บนรายการฝึกที่ต้องการ รายละเอียดจะปรากฏบริเวณด้านขวาของจอเกม ดังรูปที่ 4.6



รูปที่ 4.6 แสดงรายละเอียดการฝึกฝนที่ไก่สามารถฝึกได้

เมื่อเลือกเมนูเข้าสู่หมู่บ้านในหน้าจอหลักของเกมก็จะพบเมนูของหมู่บ้านโดยจะมีเมนูให้เลือกคือ ร้านขายไข่ ร้านขายไอเท็ม บ่อน และการแข่งขัน ดังรูปที่ 4.7



รูปที่ 4.7 แสดงเมนูหมู่บ้าน

เมื่อเลือกเมนูร้านขายไข่ในเมนูหมู่บ้าน จะสามารถเลือกซื้อไข่ได้ โดยไข่แต่ละประเภทจะมีราคาต่างกัน โดยถ้าหากผู้เล่นมีเงินน้อยกว่าราคาของไข่ก็จะไม่สามารถซื้อไข่นั้นได้ แต่ถ้าหากผู้เล่นมีเงินพอก็จะสามารถซื้อไข่ได้ และไข่ก็จะกลายเป็นไก่ สามารถดูรายละเอียดต่างๆได้ในเมนูข้อมูลไก่ชน

รูปที่ 4.8 แสดงร้านขายไข่

เมื่อเลือกเมนูร้านขายไอเท็มในเมนูหมู่บ้าน จะสามารถเลือกซื้อไอเท็มได้ โดยจะแสดงรายละเอียดของไอเท็มแต่ละชนิดโดยนำเคอร์เซอร์ของเม้าส์ไปวางไว้บนไอเท็มที่ต้องการดูรายละเอียด จะแสดงรายละเอียดของไอเท็มนั้นๆและราคาของไอเท็มบริเวณด้านขวาของจอเกม โดยถ้าหากผู้เล่นมีเงินไม่พอกับราคาของไอเท็มหรือช่องใส่ไอเท็มเต็มก็จะไม่สามารถซื้อไอเท็มนั้นได้ แต่หากผู้เล่นมีเงินพอที่จะซื้อไอเท็มนั้นได้ ก็จะได้รับไอเท็มนั้นไปอยู่ในช่องไอเท็มโดยสามารถดูได้ในช่องไอเท็มโดยเลือกจากเมนูไอเท็มในเมนูหลักของเกม

รูปที่ 4.9 แสดงร้านขายไอเท็ม

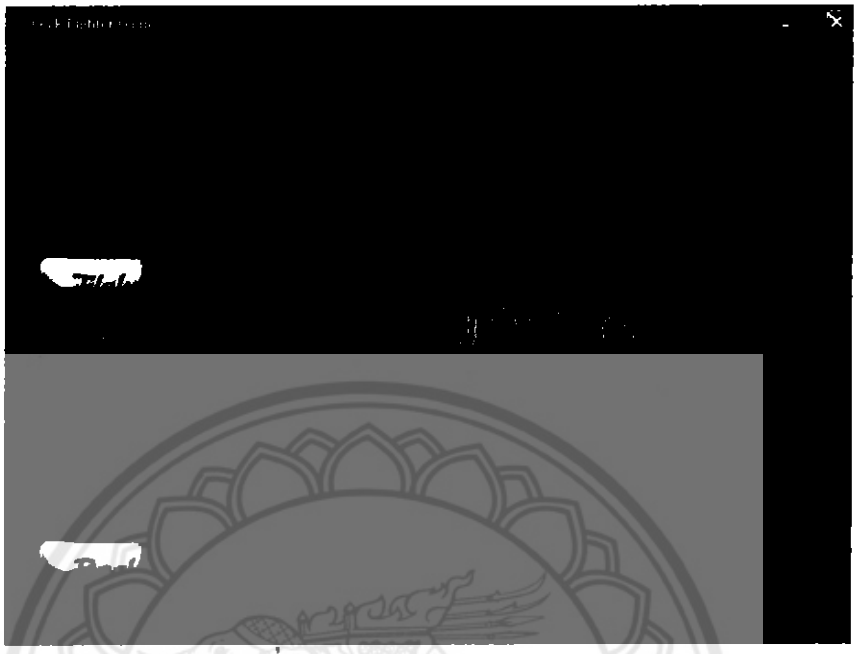
เมื่อเลือกเมนูช่องเก็บไอเท็มในเมนูหลักของเกม จะพบกับช่องเก็บไอเท็ม โดยช่องเก็บไอเท็มนี้สามารถเก็บได้สูงสุด 16 ชิ้น โดยไอเท็มสามารถหาได้จากการซื้อที่ร้านขายไอเท็ม หรือได้รับหลังจากการต่อสู้ในการแข่งขัน เมื่อคลิกเม้าส์ซ้ายที่ไอเท็มที่ต้องการใช้งานก็จะสามารถใช้งานไอเท็มนั้นได้

รูปที่ 4.10 แสดงช่องเก็บไอเท็ม

เมื่อเลือกเมนูบ่อนที่เมนูหมู่บ้านก็จะสามารถเลือกที่จะนำไก่มาต่อสู้เพื่อสะสมค่าประสบการณ์และสะสมเงินเพื่อนำไปซื้อไอเท็มหรือซื้อไข่ได้

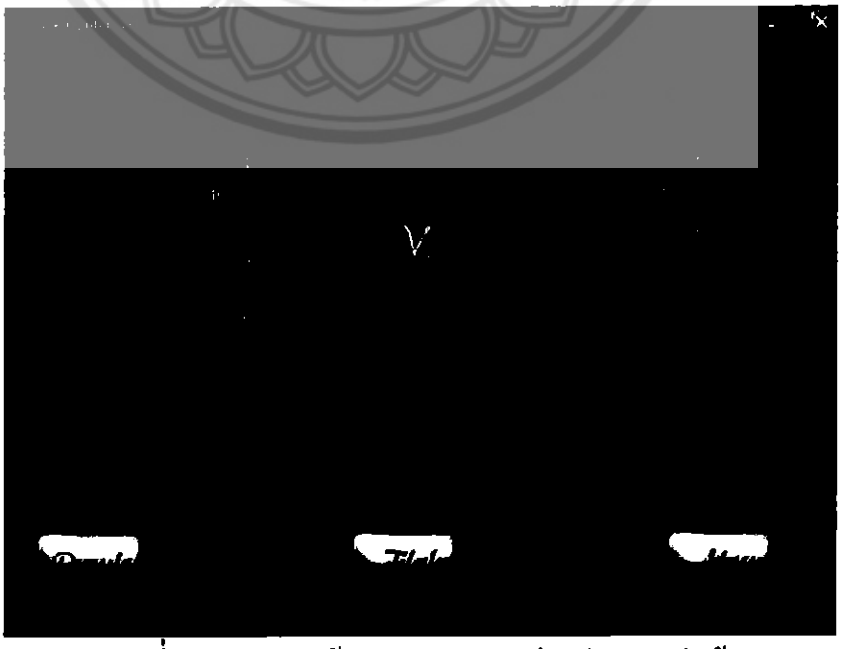
รูปที่ 4.11 แสดงหน้าจอบ่อน

เมื่อเลือกเมนูการแข่งขันที่เมนูหมู่บ้าน จะสามารถนำไก่มาแข่งขันเพื่อที่จะต่อสู้ให้ชนะเพื่อผ่านเข้ารอบต่อไป และชนะเลิศการแข่งขัน



รูปที่ 4.12 แสดงหน้าจอการแข่งขัน

เมื่อเลือกเมนูต่อสู้ที่หน้าต่างบ่อน หรือการแข่งขัน ก็จะเป็นการแสดงรายละเอียดค่าสถานะของไก่เรา และไก่ของคู่ต่อสู้ โดยข้อมูลด้านซ้ายจะเป็นข้อมูลของไก่เรา ด้านขวาจะเป็นข้อมูลของคู่ต่อสู้

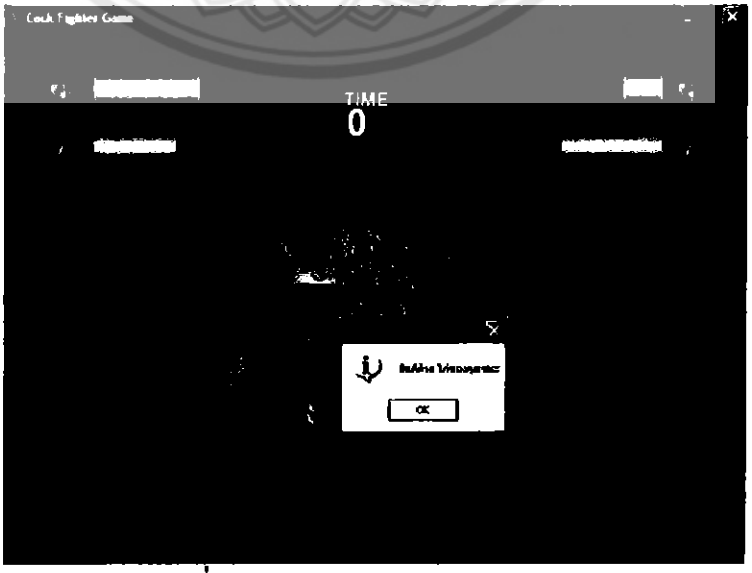


รูปที่ 4.13 แสดงหน้าจอแสดงรายละเอียดก่อนการต่อสู้

เมื่อเลือกที่การต่อสู้ในหน้าจอแสดงรายละเอียดก่อนการต่อสู้ ก็จะเข้าสู่การต่อสู้ทันที โดยจะมีการแสดงค่า HP (Health point) ของทั้งฝ่ายเราและฝ่ายคู่ต่อสู้โดยคิดเป็นเปอร์เซ็นต์โดยค่านี้จะอยู่ด้านบน ด้านซ้ายจะเป็นของเรา ส่วนด้านขวาจะเป็นของคู่ต่อสู้ และจะแสดงค่า SP (Stamina Point) ซึ่งมีไว้สำหรับใช้ในการออกท่าทางของไก่โดยคิดเป็นเปอร์เซ็นต์ และตัวเลขตรงกลางจะแสดงเวลา

รูปที่ 4.14 แสดงหน้าจอการต่อสู้ของไก่

ในระหว่างการต่อสู้หากฝ่ายศัตรูมีค่า HP เป็น 0% จะทำให้เราชนะการต่อสู้ทันที หรือหากฝ่ายเราเหลือค่า HP เป็น 0% ก็จะทำให้แพ้ทันทีเช่นกัน แต่ถ้าหากเวลาหมดลงก่อนจะตัดสินโดยการดูว่าใครเหลือ HP ซึ่งคิดเป็นเปอร์เซ็นต์ที่มากกว่าก็จะชนะไป



รูปที่ 4.15 แสดงการตัดสินผลการต่อสู้

4.4 ผลการทดลองภาคปฏิบัติ

ตารางที่ 4.1 แสดงผลการต่อสู้ระหว่างคอมพิวเตอร์ 1 กับคอมพิวเตอร์ 2 โดยที่มีค่าสถานะเท่ากันในจำนวน 20 ครั้ง

ครั้งที่	คอมพิวเตอร์ 1 ชนะ	เสมอ	คอมพิวเตอร์ 2 ชนะ
1	/		
2			/
3	/		
4	/		
5	/		
6			/
7			/
8	/		
9			/
10			/
11	/		
12	/		
13	/		
14			/
15	/		
16			/
17			/
18			/
19	/		
20	/		
รวม	11	0	9

ตารางที่ 4.2 แสดงผลการต่อสู้ระหว่างคอมพิวเตอร์ 1 กับคอมพิวเตอร์ 2 โดยที่คอมพิวเตอร์ 1 มีค่าสถานะมากกว่า คอมพิวเตอร์ 2 เล็กน้อย ในจำนวน 20 ครั้ง

ครั้งที่	คอมพิวเตอร์ 1 ชนะ	เสมอ	คอมพิวเตอร์ 2 ชนะ
1	/		
2			/
3	/		
4	/		
5	/		
6	/		
7			/
8	/		
9	/		
10	/		
11			/
12	/		
13	/		
14			/
15	/		
16	/		
17	/		
18			/
19	/		
20	/		
รวม	15	0	5

ตารางที่ 4.3 แสดงผลการต่อสู้ระหว่างคอมพิวเตอร์ 1 กับคอมพิวเตอร์ 2 โดยที่คอมพิวเตอร์ 1 มีค่าสถานะมากกว่า คอมพิวเตอร์ 2 พอสมควร ในจำนวน 20 ครั้ง

ครั้งที่	คอมพิวเตอร์ 1 ชนะ	เสมอ	คอมพิวเตอร์ 2 ชนะ
1	/		
2	/		
3	/		
4	/		
5	/		
6	/		
7	/		
8	/		
9	/		
10	/		
11	/		
12	/		
13	/		
14	/		
15	/		
16	/		
17	/		
18	/		
19	/		
20	/		
รวม	20	0	0

สรุปผลการทดลองที่ได้

- ถ้าสถานะของไก่ใกล้เคียงกันจะทำให้การต่อสู้นั้นสูสีและ ผลการแพ้ชนะที่ออกมาจะใกล้เคียงกัน
- ถ้าสถานะของไก่ไม่เท่ากัน ฝ่ายที่มีมากกว่าจะได้เปรียบในการต่อสู้ และผลการต่อสู้ฝ่ายที่มีสถานะมากกว่าจะมีโอกาสชนะมากกว่า

บทที่ 5
สรุปผลและข้อเสนอแนะ

ในบทนี้จะเป็นการสรุปผลของโครงงานนี้ ซึ่งจะกล่าวถึงการสรุปผลของโครงงาน ปัญหาในการทำงาน ข้อเสนอแนะและแนวทางในการพัฒนา เพื่อเป็นประโยชน์สำหรับผู้สนใจจะพัฒนาโครงงานนี้ต่อไป

5.1 สรุปผล

1. โปรแกรมที่พัฒนาเป็นเกม 3 มิติ ซึ่งพัฒนาโดยโปรแกรม Microsoft Visual C++ 2005 และ DirectX SDK เวอร์ชั่น 9.0c ซึ่งเป็นส่วนที่ช่วยในการแสดงผลภาพทั้ง 2 มิติ และ 3 มิติ การควบคุมการเคลื่อนไหวของตัวละครในเกมและการสร้างเคลด้วยโปรแกรมสร้างโมเดล 3 มิติ คือ 3ds max 9 การสร้างภาพและตกแต่งภาพด้วยโปรแกรม Photoshop CS3
2. โปรแกรมที่พัฒนาเป็นแบบเล่นคนเดียว เป้าหมายของเกมคือชัยชนะโดยการฝึกฝนไก่ชนของตนให้ชนะเลิศในการแข่งขันใหญ่
3. โปรแกรมที่พัฒนาในส่วนของการต่อสู้นั้นจะเป็นการทำงานของปัญญาประดิษฐ์โดยที่ผู้เล่นไม่สามารถจะควบคุมได้ การตัดสินใจจะขึ้นอยู่กับปัญญาประดิษฐ์
4. ระบบปัญญาประดิษฐ์ในส่วนของการต่อสู้ของไก่นั้นมีความสามารถใกล้เคียงกัน โดยสรุปได้จากการทดลอง ให้ไก่ที่มีค่าสถานะเท่ากันสู้กัน พบว่ามีโอกาสแพ้ชนะพอๆกัน แต่ถ้าไก่ตัวใดมีค่าสถานะมากกว่าก็จะได้เปรียบขึ้นมาอย่างเห็นได้ชัด

5.2 ปัญหาที่พบในการทำงาน

1. ผู้พัฒนามีความรู้ในการใช้โปรแกรม Visual C++ 2005 ไม่เพียงพอ จำเป็นต้องเสียเวลาในการศึกษาการใช้งาน โปรแกรม Visual C++ 2005 พอสมควร
2. ผู้พัฒนามีความรู้ในการใช้ DirectX SDK เวอร์ชัน 9.0c ไม่เพียงพอ จำเป็นต้องเสียเวลาในการศึกษาการใช้งาน DirectX SDK เวอร์ชัน 9.0c พอสมควร
3. การพัฒนาเกมต้องใช้โมเดล 3 มิติ ซึ่งเป็นส่วนที่ใช้เวลานานพอสมควร เนื่องจากต้องมีการออกแบบโมเดล และยังต้องใช้เวลาในการศึกษาการใช้งานโปรแกรม 3ds max 9 เพื่อใช้ในการสร้างโมเดล

5.3 ข้อเสนอแนะ

1. ก่อนการดำเนินงานในแต่ละขั้นตอนควรจะทำการศึกษาข้อมูลให้เข้าใจมากที่สุด เพื่อให้เกิดข้อผิดพลาดในการทำงานน้อยที่สุด

2. ควรจะดำเนินงานในแต่ละขั้นตอนให้เสร็จก่อนกำหนดจะดีที่สุด เพราะบางขั้นตอนอาจต้องใช้เวลาในการพัฒนามากกว่าเวลาที่กำหนดไว้

3. ควรจะมีการทดสอบการทำงานของโปรแกรมแต่ละส่วนเพื่อการหาข้อผิดพลาดที่เกิดขึ้นและทำการแก้ไขให้เสร็จก่อน จากนั้นนำแต่ละส่วนมาประกอบกันเป็นโปรแกรมที่มีขนาดใหญ่ขึ้น เพื่อจะได้หาข้อผิดพลาดและแก้ไขโปรแกรมได้ง่ายขึ้น

5.4 แนวทางในการพัฒนา

1. ควรจะเพิ่มรูปแบบการแข่งขันต่างๆเพื่อความสนุกของเกม

2. ใช้หลักการปัญญาประดิษฐ์อื่นๆเพื่อที่จะให้ไก่แสดงท่าทางได้เหมาะสมและสมจริงยิ่งขึ้น

3. เพิ่มรูปแบบเงื่อนไขต่างๆในเกมเพื่อให้เกมสมจริงและน่าเล่นยิ่งขึ้น

4. เพิ่มการเล่นแบบหลายคนโดยผ่านระบบเครือข่ายเพื่อเพิ่มความสนุกให้กับเกม

5. พัฒนากราฟฟิกให้สวยงามยิ่งขึ้นเพื่อให้เกมน่าสนใจยิ่งขึ้น

6. เพิ่มรายละเอียดของไก่เพื่อให้เกมน่าเล่นยิ่งขึ้น

7. เพิ่มเสียงประกอบภายในเกม เพื่อให้เกมมีความน่าสนใจยิ่งขึ้น

8. สามารถนำไปพัฒนาเกมในรูปแบบอื่นๆ โดยใช้หลักการพัฒนาเกมไก่ชนนี้ได้

เอกสารอ้างอิง

[1] ไม่ปรากฏชื่อผู้แต่ง. "DirectX 9 Tutorials." [online]. Available : http://www.directxtutorial.com/Tutorial9/tutorials.aspx

[2] นิรุธ อำนวยศิลป์. เขียนเกมอย่างมืออาชีพด้วย Visual C++ และ DirectX. นนทบุรี : สำนักพิมพ์ infopress. 2545.

[3] ไม่ปรากฏชื่อผู้แต่ง. "X File Hierarchy Loading." [online]. Available : http://www.toymaker.info/Games/html/load_x_hierarchy.html

[4] ไม่ปรากฏชื่อผู้แต่ง. "DirectX Graphics." [online]. Available : http://www.thaigamearticles.com/Resource/Articlestutorials/DirectX/directx.php

[5] อนุรักษ์ ภู่เนตร. สร้างงาน 3D ด้วย 3ds max 9. กรุงเทพฯ : พีเอ็นเอ็นกรุ๊ป 2550.

[6] อนัน วาโซะ. สร้างงาน Character Animation Character Studio. นนทบุรี : ไอดีซีฯ 2550

[7] ไม่ปรากฏชื่อผู้แต่ง. "Intelligent agent." [online]. Available : http://en.wikipedia.org/wiki/Intelligent_agents

[8] ไม่ปรากฏชื่อผู้แต่ง. "3d tutorials" [online]. Available : http://www.thai3d.net

ประวัติผู้เขียนโครงงาน

ชื่อ นายศิริโชค ปวงเหมือง

ภูมิลำเนา 147 หมู่ 1 ตำบลน้ำชำ อำเภอเมืองแพร่ จังหวัดแพร่ 54000

ประวัติการศึกษา

- จบประถมศึกษาจากโรงเรียนบ้านน้ำชำ(วิชัยชนานุเคราะห์) จังหวัดแพร่
- จบมัธยมศึกษาจากโรงเรียนห้วยม้าวิทยาคม จังหวัดแพร่
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

Email: iron_ear@hotmail.com

ชื่อ นายแสวงศักดิ์ นาถกร

ภูมิลำเนา 232 หมู่ 10 ตำบลแม่เมาะ อำเภอแม่เมาะ จังหวัดลำปาง 52220

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนบุญวาทย์วิทยาลัย จังหวัดลำปาง
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

E-mail: marijuana_nt@hotmail.com