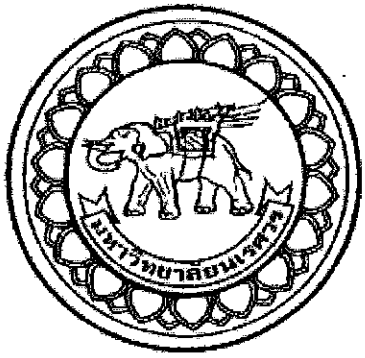


ต้นแบบเกม 3 มิติประเภทต่อสู้ด้วยตนเองโดยใช้เทคนิคปัญญาประดิษฐ์

A PROTOTYPE FOR 3D GAME WITH BOT USING AI TECHNIQUE

นายศรัณย์ พิริยโยธา รหัสนิสิต 49361966

นายยุทธนา เรืองนุกูล รหัสนิสิต 49364530

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต 2552

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2552

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	ต้นแบบเกม 3 มิติประเภทต่อสู้ด้วยตนเองโดยใช้เทคนิคปัญญาประดิษฐ์
ผู้ดำเนินโครงงาน	นายศรัณย์ พิริยโยธา รหัสนิสิต 49361966
	นายยุทธนา เรืองนุกูล รหัสนิสิต 49364530
อาจารย์ที่ปรึกษา	ดร. สุรเดช จิตประไพกุลศาล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2552

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ดร. สุรเดช จิตประไพกุลศาล)

.....กรรมการ
(อาจารย์ภาณุพงศ์ สอนคม)

.....กรรมการ
(อาจารย์เศรษฐา ตั้งค้าวานิช)

หัวข้อโครงงาน	ต้นแบบเกม 3 มิติประเภทต่อสู้ด้วยตนเองโดยใช้เทคนิคปัญญาประดิษฐ์		
ผู้ดำเนินโครงงาน	นายศรัณย์	พิริยโยธา	รหัสนิสิต 49361966
	นายยุทธนา	เรืองนุกูล	รหัสนิสิต 49364530
อาจารย์ที่ปรึกษา	ดร. สุรเดช	จิตประไพกุลศาล	
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2552		

.....

บทคัดย่อ

โครงงานนี้เป็นการพัฒนาเกมที่มีระบบปัญญาประดิษฐ์เข้ามามีส่วนช่วยในการประมวลผล ซึ่งจะทำงานบนระบบปฏิบัติการ Windows โดยใช้โปรแกรม Microsoft Visual Studio 2005 ภาษา C++ และ Irrlicht Game Engine เป็นเครื่องมือในการพัฒนา ซึ่งช่วยในการจัดการเกี่ยวกับการแสดงผลภาพ 3 มิติ การติดต่อการทรัพยากรภายในเครื่องคอมพิวเตอร์ และระบบการรับ input จากผู้เล่นด้วย mouse และ keyboard นอกจากนี้ยังมีโปรแกรม Autodesk 3ds Max 2009 และ Adobe Photoshop CS2 เพื่อช่วยในการออกแบบตัวละคร 3 มิติ ฉากต่าง ๆ และตกแต่งภาพภายในเกม

เกม 3 มิติที่พัฒนาขึ้นมานี้ มีรูปแบบการต่อสู้โดยใช้ระบบปัญญาประดิษฐ์ในการวิเคราะห์การกระทำของมอนสเตอร์ โดยมอนสเตอร์จะมีการเรียนรู้เพื่อหาว่าการกระทำใดจะส่งผลดีที่สุดในแต่ละสถานการณ์ ซึ่งผู้เล่นจะต้องสวมบทบาทเป็นผู้เลี้ยงมอนสเตอร์ นำมอนสเตอร์ไปต่อสู้ เพื่อสะสมประสบการณ์ให้มอนสเตอร์ของตนเองแข็งแกร่งขึ้นต่อไป

Project title	A prototype for 3D game with bot using AI technique		
Name	Mr. Sarun	Piriyayotha	ID. 49361966
	Mr. Yutthana	Roungnukul	ID. 49364530
Project advisor	Suradet	Jitprapaikulsarn, Ph.D.	
Major	Computer Engineering.		
Department	Electrical and Computer Engineering.		
Academic year	2009		

.....

ABSTRACT

In this project, we develop a 3D game with the learning capability using artificial intelligence. The program was developed using Microsoft Visual Studio, C++ language and Irrlicht The Characters and scenes were developed using Autodesk 3ds Max 2009 and Adobe Photoshop CS2.

The program uses artificial intelligence to analyze the monsters action. Monsters will learn to find the best action for each situation. The player acts as monster trainer and releases them to fight with other monsters. The collecting experiences will make monsters strong in the future.

กิตติกรรมประกาศ

โครงงานนี้สามารถสำเร็จมาได้ด้วยดี เนื่องด้วยความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษา ดร. สุรเดช จิตประไพกุลศาล อาจารย์ภาณุพงศ์ สอนคม และอาจารย์เศรษฐา ตั้งค้าวานิช ที่กรุณาให้คำปรึกษา แนะนำวิธีการในการทำงาน ตลอดจนถึงการตรวจสอบการทำงานพร้อมทั้งเสนอแนะแนวทางแก้ไขตลอดระยะเวลาทำโครงงาน สุดท้ายต้องขอขอบคุณอาจารย์ทุกท่าน พี่ ๆ และเพื่อน ๆ ทุกคนที่ยังไม่ได้เอ่ยนาม ที่คอยสนับสนุนในการทำโครงงานครั้งนี้

ผู้จัดทำ

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันนี้เทคโนโลยีด้านฮาร์ดแวร์ของเครื่องคอมพิวเตอร์ส่วนบุคคล (Personal Computer) ได้พัฒนาให้มีประสิทธิภาพมากขึ้นอย่างรวดเร็ว โดยมีปัจจัยหลักมากจากการพัฒนาเกมที่มีความสมจริงมากขึ้น จึงเกิดการแข่งขันในกลุ่มผู้พัฒนา ทั้งผู้พัฒนาฮาร์ดแวร์ และผู้พัฒนาเกม โดยเหตุนี้จึงเกิดเป็นตลาดที่ใหญ่ที่มีเงินลงทุนมากและทำรายได้เป็นจำนวนมาก

ประกอบกับปัจจุบันมีเกมที่เป็นของคนไทยยังมีน้อยมาก อีกทั้งในหมู่ผู้เล่นเกมนั้น ยังไม่ให้ความสำคัญและสนับสนุนเกมที่สร้างโดยคนไทยด้วยกันเองเท่าที่ควร เนื่องจากยังขาดเงินลงทุน และการประชาสัมพันธ์ที่ดี เพราะการสร้างเกมที่ดีนั้นยังต้องใช้เงินลุงทุนค่อนข้างมาก ทางผู้จัดทำจึงได้ริเริ่มที่จะสร้างเกมที่ใช้ต้นทุนในการสร้างไม่สูงนัก เพื่อเป็นแบบอย่างให้นักพัฒนาเกมชาวไทยได้นำไปศึกษาและพัฒนาเกมต่อไป

ในอดีตการพัฒนาเกมนั้น ค่อนข้างมีความยุ่งยากสลับซับซ้อน ทำให้มีการพัฒนาเกมเป็นไปได้ยาก แต่ปัจจุบันมีการพัฒนาตัวช่วยในการสร้างเกม หรือ Game Engine ขึ้น โดย Game Engine จะทำหน้าที่ช่วยจัดการติดต่อกับอุปกรณ์ต่าง เช่น mouse keyboard และทำหน้าที่จัดการทรัพยากรต่างๆในเครื่องคอมพิวเตอร์ เป็นตัวช่วยทำให้นักพัฒนาเกม สามารถพัฒนาเกมได้อย่างสะดวกและรวดเร็วมากขึ้น ซึ่งผู้จัดทำได้เลือกจะใช้ Game Engine ที่ชื่อว่า Irrlicht Game Engine ซึ่งเป็น Open Source ที่ใช้กันแพร่หลาย

ในโครงงานนี้จะเป็นการพัฒนาเกมแนว RPG (Role Playing Game) ในรูปแบบ 3 มิติ ที่มีระบบการเล่นโดยผู้เล่นมีการเคลื่อนไหวได้อย่างอิสระในฉากทั่วไป และผู้เล่นจะมีการเลี้ยงมอนสเตอร์เพื่อนำไปต่อสู้กับศัตรู และในการต่อสู้นั้นจะใช้ ระบบปัญญาประดิษฐ์ (Artificial Intelligent) ในการตัดสินใจโดยเรียนรู้จากข้อมูลที่ได้เก็บรวบรวมไว้ผ่านมาหลาย ๆ ครั้ง เพื่อหาว่าทักษะใดที่มอนสเตอร์ใช้นั้น ทำให้เกิดประสิทธิภาพได้สูงสุด โดยผู้เล่นสามารถเพิ่มความสามารถของมอนสเตอร์ได้จากการกินผลไม้ชนิดต่าง ๆ และจากการเรียนรู้ผ่านการต่อสู้กับศัตรู ซึ่งเป็นแนวเกมที่ยังไม่พบเห็นในปัจจุบัน

ซึ่งในโครงงานนี้พัฒนาด้วยภาษา C++ โดย Microsoft Visual Studio 2005 ร่วมกับ Irrlicht Game Engine ซึ่งเป็น 3D Game Engine ที่ค่อนข้างสมบูรณ์ และเป็น Open Source ที่สามารถใช้งานร่วมกับโปรแกรมด้านกราฟฟิกเกี่ยวกับการสร้างตัวละครและฉากหลัง ตัวอย่างเช่น 3D Studio Max และในด้านเสียงภายในตัวเกมได้ใช้ IrrKlang ซึ่งเป็น Library เสริมที่มีผู้พัฒนาเดียวกับ Irrlicht Game Engine

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อสร้างต้นแบบเกม 3 มิติ ที่มีรูปแบบการต่อสู้ด้วยตนเองโดยใช้ระบบปัญญาประดิษฐ์ในการตัดสินใจได้

1.2.2 เพื่อสร้างระบบปัญญาประดิษฐ์ในรูปแบบการเรียนรู้จากข้อมูลต่าง ๆ ที่รวบรวมมาไว้ได้

1.3 ขอบเขตของโครงงาน

1.3.1 แสดงผลได้เฉพาะบนวินโดว์เท่านั้น

1.3.2 ต้องใช้ Library Irrlicht.dll และ IrrKlang.dll ในการแสดงผลเท่านั้น

1.3.3 ตัวเกมสามารถเล่นได้เพียงคนเดียว

1.3.4 เกมมีระบบปัญญาประดิษฐ์ที่สามารถตัดสินใจได้เองในฉากต่อสู้ และสามารถเรียนรู้ได้

1.4 ขั้นตอนการดำเนินโครงงาน

ขั้นตอนการดำเนินงานสามารถแบ่งเป็นขั้นตอนได้ดังนี้

1.4.1 ออกแบบตัวโครงสร้างของเกมทั้งหมด

1.4.2 ศึกษาวิธีการสร้างเกม โปรแกรมที่เกี่ยวข้อง และปัญญาประดิษฐ์ที่ใช้

1.4.3 ศึกษาการวิธีใช้ Irrlicht Game Engine

1.4.4 ศึกษาการวิธีใช้ 3DS Max เพื่อใช้ในการสร้างภาพ 3D

1.4.5 ศึกษาการวิธีการใช้ A.I. และหลักการเขียน A.I. algorithm

1.4.6 เริ่มลงมือการเขียนโปรแกรม

1.4.7 ตรวจสอบแก้ไขและทดลองเกม

1.4.8 จัดทำเอกสารและรายงาน

1.5 แผนการดำเนินงาน

การดำเนินงานของโครงงานนั้นได้วางรูปแบบการดำเนินงานออกไว้เป็นหัวข้อต่าง ๆ โดยในแต่ละหัวข้อนั้นได้วางแผนเวลาในการศึกษาและการลงมือปฏิบัติตามตารางที่ 1.1

ตารางที่ 1.1 แผนการดำเนินโครงงาน

ขั้นตอนการกำเนินงาน	ระยะเวลาในการดำเนินงาน								
	มิ.ย. 52	ก.ค. 52	ส.ค. 52	ก.ย. 52	ต.ค. 52	พ.ย. 52	ธ.ค. 52	ม.ค. 53	ก.พ. 53
1.5.1 ออกแบบตัวโครงสร้างของเกมทั้งหมด	←	—	—	—	—	→			
1.5.2 ศึกษาวิธีการสร้างเกม โปรแกรมที่เกี่ยวข้อง และปัญญาประดิษฐ์ที่ใช้	←	—	—	—	—	→			
1.5.3 ศึกษาการวิธีใช้ Irrlicht Game Engine	←	—	—	—	—	→			
1.5.4 ศึกษาการวิธีใช้ 3DS Max เพื่อใช้ในการสร้างภาพ 3D						←	—	→	
1.5.5 ศึกษาการวิธีการใช้ A.I. และหลักการเขียน A.I. algorithm						←	—	→	
1.5.6 เริ่มลงมือการเขียนโปรแกรม								←	→
1.5.7 ตรวจสอบแก้ไขและทดลองเกม								←	→
1.5.8 จัดทำเอกสารและรายงาน									← →

1.6 ผลที่คาดว่าจะได้รับ

1.6.1 สามารถนำเกม 3 มิตินี้ไปพัฒนาต่อ หรือเป็นพื้นฐานในการสร้างเกมอื่น ๆ ต่อไปได้

1.6.2 เพิ่มความสามารถในกระบวนการคิด วิเคราะห์ การแก้ปัญหาและการเขียนโปรแกรมในระบบปัญญาประดิษฐ์

1.7 งบประมาณของโครงงาน

งบประมาณที่ใช้ในโครงงานนี้จะเกิดการซื้อหนังสือเอกสารสำหรับเรียนรู้

ค่าหนังสือต่าง ๆ	1600 บาท	
ค่าเบ็ดเตร็ด	400 บาท	
รวม	2,000 บาท	(สองพันบาทถ้วน)

หมายเหตุ ถัวเฉลี่ยทุกรายการ

บทที่ 2
ทฤษฎีเบื้องต้น

หลักการและทฤษฎีที่ใช้ในโครงงานนี้ เป็นการรวบรวมเอาหลักการและข้อมูลต่าง ๆ ที่เกี่ยวข้องในการปฏิบัติการของโครงงาน โดยสามารถแบบออกเป็น 4 หัวเรื่อง คือ Irrlicht Game Engine ระบบปัญญาประดิษฐ์ ภาษา C++ และระบบภาพ 3 มิติ ซึ่งมีรายละเอียดของแต่ละหัวข้อดังต่อไปนี้

2.1 Irrlicht Game Engine

2.1.1 คุณลักษณะของ Game Engine

Game Engine คือ โปรแกรมด้านการแสดงผลภาพที่มีการตอบสนองแบบทันทีทันใด โดยทำหน้าที่ช่วยจัดการติดต่อกับอุปกรณ์พื้นฐาน เช่น Mouse Keyboard เป็นตัวช่วยในการพัฒนาให้เกมง่ายขึ้น และช่วยให้พัฒนาบนระบบปฏิบัติการที่แตกต่างกัน เช่น Linux MacOS Windows หรือบนเครื่องเกมคอนโซล เป็นต้น คุณสมบัติหรือความสามารถหลักที่ Game Engine โดยทั่วไปคือสร้างประมวลผลภาพ 2 มิติ และ 3 มิติ การประมวลผลทางด้านฟิสิกส์ ตรวจสอบการตอบสนองต่อการชน ระบบเสียง ภาพเคลื่อนไหว ตัวอักษร ปัญญาประดิษฐ์ การถ่ายโอนข้อมูล รวมทั้งจัดการและบริหารหน่วยความจำ ซึ่ง ณ ปัจจุบันแนวทางในการพัฒนาเกมของอุตสาหกรรมเกมนั้น ส่วนใหญ่จะนำ Game Engine มาใช้เป็นส่วนประกอบหลักในการสร้างเกม

เดิมทีนั้นซอฟต์แวร์พัฒนาเกมแบบ Open Source จะเป็นการสร้างในรูปแบบของ SDK หรือ Software Development Kit ซึ่งเป็นการเพิ่มคำสั่งหรือคลาสให้กับตัว Compiler ของภาษาคอมพิวเตอร์ทั่ว ๆไป ซึ่งผนวกการใช้ API ที่มีอยู่เช่น DirectX หรือ OpenGL แต่หลังจาก John D. Carmac โปรแกรมเมอร์ผู้พัฒนาเกมสามมิติชื่อดังอย่างเกม Quake และ Doom ได้นำ Game Engine ที่เคยมีอยู่มาเผยแพร่เป็น Open Source ภายใต้สัญญาอนุญาตแจกจ่ายแบบ GNU ซึ่งทำให้นักพัฒนาเกมทั่วโลกที่เคยซื้อ Game Engine อย่าง Id Tech มาให้ในราคาหลายล้านก็ต้องตกใจ เพราะราคาสูงแพงหายไปเหลือเพียงของฟรีเท่านั้น แต่ก็ทำให้เกม Open Source ได้เพิ่มเทคนิคขึ้นมาอีกหลายอย่างทำให้เกิดการนำ OpenSource ไปใช้ในอุตสาหกรรมเกมมากขึ้น

สำหรับ Game Engine บางตัวนั้นจะทำหน้าที่ในการสร้าง หรือประมวลผลภาพ 2 มิติ และ 3 มิติเพียงอย่างเดียวเท่านั้น โดยนำมาใช้กับเกมที่มีความต้องการความสามารถในการทำงานที่หลากหลาย โดย Engine ประเภทนี้ผู้พัฒนาเกมจะต้องเพิ่มเติมความสามารถขึ้นมาเองหรือทำการ

รวมความสามารถจาก Game Middleware Components อื่น ๆ บางครั้งเราจะเรียก Engine เหล่านี้ว่า "Graphics Engine" หรือ "Rendering Engine" หรือ "3D Engine" ซึ่งเราสามารถเรียกรวม ๆ ว่าเป็น Game Engine ได้เช่นเดียวกัน และเนื่องจาก 3D Game Engine ที่มีคุณสมบัติครบถ้วนนั้นก็มีรากฐานมาจาก 3D Engine พื้นฐาน ตัวอย่างของ Graphics Engine ได้แก่ IrrlichtEngine OgreEngine jMonkeyEngine RealmForge Power Render Crystal Space Genesis3D เป็นต้น

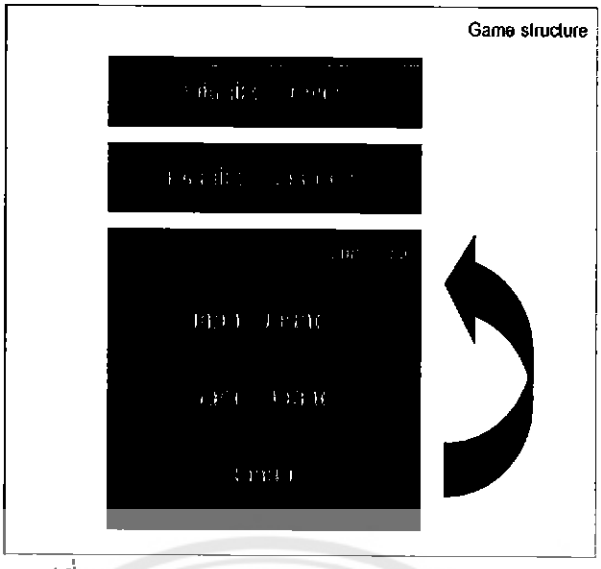
2.1.2 ประวัติ Irrlicht Game Engine

Irrlicht 3D Engine พัฒนาโดย ITSC จุดประสงค์เพื่อกลุ่มนักพัฒนาเกมที่มีความสนใจด้านการพัฒนาเกมได้นำไปศึกษา โดย Irrlicht Engine เป็นซอฟท์แวร์ประเภท Open Source ซึ่งสามารถนำมาใช้งานได้โดยไม่มีค่าใช้จ่าย Irrlicht Engine ถูกพัฒนาขึ้นด้วยภาษา C++ โดยรองรับการนำไปพัฒนาด้วยภาษา C++ และภาษาอื่น ๆ เช่น VC# .Net VB .Net Java เป็นต้น อีกทั้งสามารถใช้งานได้บนหลายระบบปฏิบัติการ อาทิเช่น Windows Linux และ Mac OS เป็นต้น Irrlicht Engine ถูกพัฒนามาให้สามารถแสดงผลภาพโดยใช้ทั้ง DirectX OpenGL ซึ่ง Irrlicht Engine ถูกออกแบบมาให้มีประสิทธิภาพได้ค่อนข้างสูงในขณะที่มีขนาดเล็ก และยังสามารถเรียนรู้เข้าใจได้ง่ายเมื่อเปรียบเทียบกับ Engine อื่น ๆ

ตั้งแต่ดั้งเดิม Irrlicht Engine ถูกออกแบบมาเป็น Graphics Engine ที่เพิ่มความสามารถทำให้มีสะดวกสบายในการใช้งานมากขึ้น เช่น การตรวจสอบการชน การรับสัญญาณข้อมูลจากคีย์บอร์ด-เมาส์ แต่ก็ยังขาดในส่วนของระบบปัญญาประดิษฐ์ Sound และ ระบบ Network หรือสิ่งอื่นที่เราต้องการสำหรับการสร้างเกม ถึงแม้ว่า Irrlicht Engine จะยังไม่มีส่วนประกอบเหล่านี้แต่มันก็เป็น Graphics Engine ที่มีประสิทธิภาพสูงสำหรับการสร้างเกม สำหรับ Irrlicht Engine นั้นสามารถทำงานได้ทั้งบนระบบปฏิบัติการ Windows MacOS หรือ Linux โดยสามารถใช้ Direct3D OpenGL หรือตัวโปรแกรมของ Irrlicht Engine สำหรับการ Render

2.1.3 โครงสร้างการทำงานของ Irrlicht Game Engine

หลักการทำงานของ Irrlicht Game Engine จะเป็นลักษณะของลูปโปรแกรม โดยเริ่มจากส่วนของการกำหนดค่าทรัพยากรต่าง ๆ ของเครื่อง แล้วจึงเข้าส่วนของลูปโปรแกรมซึ่งประกอบไปด้วยการรับค่า Input จากผู้ใช้ แล้วนำมา Update ค่าตัวแปรต่าง ๆ แล้วจึงเป็นการ Render ภาพออกมา และจะเป็นการทำงานไปเรื่อย ๆ จนกว่าจะจบการทำงาน



รูปที่ 2.1 การทำงานของ Irrlicht Game Engine [1]

2.1.4 ชุดคำสั่งของ Irrlicht Game Engine

ใน Irrlicht Game Engine จะมีชุดคำสั่งพื้นฐานต่าง ๆ มากมาย ทั้งการโหลดโมเดล 3 มิติ การสร้างพื้นผิว การตรวจจับการชนกันของวัตถุ เป็นต้น โดยสามารถอธิบายถึงชุดคำสั่งที่จำเป็นในการพัฒนาโครงงานนี้ได้ดังนี้.

2.1.4.1 Namespace irr

Namespace irr จะเป็น Namespace หลักของ Irrlicht Game Engine ซึ่งยังประกอบไปด้วย Namespace ที่จำเป็นต่าง ๆ ตามลักษณะการทำงาน ประกอบด้วย

1. Namespace irr::core เป็น Namespace ที่เก็บส่วนของคลาสทางคณิตศาสตร์ต่าง ๆ เช่น Vector หรือ Matrix เป็นต้น
2. Namespace irr::gui เป็น Namespace ที่เก็บส่วนของคลาสที่เกี่ยวกับ GUI
3. Namespace irr::io เป็น Namespace ที่เก็บส่วนของคลาสที่เกี่ยวกับ Input และ Output
4. Namespace Irr::scene เป็น Namespace ที่เก็บส่วนของคลาสที่เกี่ยวกับโมเดล 3 มิติ
5. Namespace irr::video เป็น Namespace ที่เก็บส่วนของคลาสที่ทำงานเกี่ยวกับ Render

2.1.4.2 คลาสที่สำคัญใน Irrlicht Game Engine

ใน Irrlicht Game Engine จะมีคลาสต่าง ๆ ไว้เพื่อความสะดวกในการพัฒนามากมายโดยจะครอบคลุมการทำงานในส่วนต่าง ๆ ซึ่งทางผู้จัดทำได้ยกตัวอย่างคลาสที่สำคัญไว้ดังนี้

IrrlichtDevice เป็นคลาสหลักที่ใช้จัดการส่วนของการแสดงผลในหน้าต่างเกมทั้งหมด โดยส่วนใหญ่จะใช้สำหรับการกำหนดค่าของหน้าต่างเกม เช่น กำหนดขนาดหน้าต่างเกมกำหนดลักษณะการแสดงผล (เต็มจอหรือไม่เต็มจอ) รวมทั้งการกำหนดชุดคำสั่ง (Library) ที่ใช้จัดการด้านกราฟฟิก ซึ่งตัว Irrlicht Game Engine สามารถรองรับได้หลายชุดคำสั่งไม่ว่าจะเป็น OpenGL หรือ Direct3D

scene::ISceneManager เป็นคลาสที่ใช้จัดการส่วนโมเดลต่าง ๆ โดยส่วนใหญ่จะใช้สำหรับการโหลดโมเดล 3 มิติ และการสร้างกล้องจำลองเพื่อกำหนดมุมมอง

gui::IGUIEnvironment เป็นคลาสที่ใช้จัดการส่วน GUI โดยส่วนใหญ่จะใช้สำหรับการแสดงภาพสองมิติต่าง ๆ

IEventReceiver เป็นคลาสที่ใช้จัดการส่วนของการรับค่า Input โดยส่วนใหญ่จะใช้ในส่วนของการแสดงผลในแต่ละฉาก เพื่อเป็นเงื่อนไขของฉากนั้น ๆ

ISoundEngine เป็นคลาสที่ใช้จัดการส่วนของเสียง

video::IVideoDriver เป็นคลาสที่ใช้จัดการส่วนที่เกี่ยวกับการ Render ภาพ

2.2 ระบบปัญญาประดิษฐ์ (Artificial Intelligence)

ปัญญาประดิษฐ์ (Artificial Intelligence) หรือ เอไอ (AI) หมายถึงความฉลาดเทียมที่สร้างขึ้นให้กับสิ่งที่ไม่มีชีวิต ปัญญาประดิษฐ์เป็นสาขาหนึ่งในด้านวิทยาการคอมพิวเตอร์ และวิศวกรรมเป็นหลัก แต่ยังรวมถึงศาสตร์ในด้านอื่นๆอย่างจิตวิทยา ปรัชญา หรือชีววิทยา ซึ่งสาขาปัญญาประดิษฐ์เป็นการเรียนรู้เกี่ยวกับกระบวนการการคิด การกระทำ การให้เหตุผล การปรับตัว หรือการอนุมาน และการทำงานของสมอง แม้ว่าดังเดิมนั้นเป็นสาขาหลักในวิทยาการคอมพิวเตอร์ แต่แนวคิดหลาย ๆ อย่างในศาสตร์นี้ได้มาจากการปรับปรุงเพิ่มเติมจากศาสตร์อื่น ๆ เช่น

การเรียนรู้ของเครื่อง นั้นมีเทคนิคการเรียนรู้ที่เรียกว่า การเรียนรู้ต้นไม้ตัดสินใจ ซึ่งประยุกต์เอาเทคนิคการอุปนัยของ จอห์น สจวร์ต มิลล์ นักปรัชญาชื่อดังของอังกฤษ มาใช้

เครือข่ายประสาทเทียมก็นำเอาแนวคิดของการทำงานของสมองของมนุษย์ มาใช้ในการแก้ปัญหาการแบ่งประเภทของข้อมูล และแก้ปัญหาอื่นๆ ทางสถิติ เช่น การวิเคราะห์ความถดถอย หรือ การปรับเส้นโค้ง

อย่างไรก็ตาม เนื่องจากปัจจุบันวงการปัญญาประดิษฐ์มีการพัฒนาส่วนใหญ่โดยนักวิทยาศาสตร์คอมพิวเตอร์ อีกทั้งวิชาปัญญาประดิษฐ์ ก็ต้องเรียนที่ภาควิชาคอมพิวเตอร์ของคณะวิทยาศาสตร์หรือคณะวิศวกรรมศาสตร์ จึงถือเอาง่าย ๆ ว่า ศาสตร์นี้เป็นสาขาของวิทยาการคอมพิวเตอร์นั่นเอง

2.2.1 นิยามของปัญญาประดิษฐ์

สามารถจัดแบ่งออกเป็น 4 ประเภทโดยมองใน 2 มิติ ได้แก่

1. นิยามที่เน้นระบบที่เลียนแบบมนุษย์ กับ นิยามที่เน้นระบบที่ระบบที่มีเหตุผล
2. นิยามที่เน้นความคิดเป็นหลัก กับ นิยามที่เน้นการกระทำเป็นหลัก

ปัจจุบันงานวิจัยหลัก ๆ ของระบบปัญญาประดิษฐ์จะมีแนวคิดในรูปที่เน้นเหตุผลเป็นหลัก เนื่องจากการนำระบบปัญญาประดิษฐ์ไปประยุกต์ใช้แก้ปัญหา ไม่จำเป็นต้องอาศัยอารมณ์หรือความรู้สึกของมนุษย์ อย่างไรก็ตามนิยามทั้ง 4 ไม่ได้ต่างกันโดยสมบูรณ์ นิยามทั้ง 4 ต่างก็มีส่วนร่วมที่คาบเกี่ยวกันอยู่

2.2.1.1 ระบบที่คิดเหมือนมนุษย์ (Systems that think like humans)

1. [ปัญญาประดิษฐ์คือ] ความพยายามใหม่อันน่าตื่นเต้นที่จะทำให้คอมพิวเตอร์คิดได้ เครื่องจักรที่มีสติปัญญาอย่างครบถ้วนและแท้จริง

2. [ปัญญาประดิษฐ์คือ กลไกของ] กิจกรรมที่เกี่ยวข้องกับความคิดมนุษย์ เช่น การตัดสินใจ การแก้ปัญหา การเรียนรู้

หมายเหตุ ก่อนที่จะทำให้เครื่องคิดอย่างมนุษย์ได้ ต้องรู้ก่อนว่ามนุษย์มีกระบวนการคิดอย่างไร ซึ่งการวิเคราะห์ลักษณะการคิดของมนุษย์ เป็นศาสตร์ด้าน Cognitive science เช่น ศึกษาการเรียงตัวของเซลล์สมองในสามมิติ ศึกษาการถ่ายเทประจุไฟฟ้า และวิเคราะห์การเปลี่ยนแปลงทางเคมีไฟฟ้าในร่างกาย ระหว่างการคิด ซึ่งจนถึงปัจจุบัน (พ.ศ. 2548) เราก็ยังไม่รู้แน่ชัดว่า มนุษย์เรา คิดได้อย่างไร

2.2.1.2 ระบบที่กระทำเหมือนมนุษย์ (Systems that act like humans)

1. [ปัญญาประดิษฐ์คือ] วิชาของการสร้างเครื่องจักรที่ทำงานในสิ่งซึ่งอาศัยปัญญาเมื่อกระทำโดยมนุษย์

2. [ปัญญาประดิษฐ์คือ] การศึกษาวิธีทำให้คอมพิวเตอร์กระทำในสิ่งที่มนุษย์ทำได้ดีกว่าในขณะนั้น

หมายเหตุ การกระทำเหมือนมนุษย์ เช่น

สื่อสารได้ด้วยภาษาที่มนุษย์ใช้ เช่น ภาษาไทย ภาษาอังกฤษ ตัวอย่างคือ การแปลงข้อความเป็นคำพูด และ การแปลงคำพูดเป็นข้อความ

มีประสาทรับสัมผัสคล้ายมนุษย์ เช่น คอมพิวเตอร์รับภาพได้โดยอุปกรณ์รับสัมผัส แล้วนำภาพไปประมวลผล

เคลื่อนไหวได้คล้ายมนุษย์ เช่น หุ่นยนต์ช่วยงานต่าง ๆ อย่างการ ดูดฝุ่น เคลื่อนย้ายสิ่งของ

เรียนรู้ได้ โดยสามารถตรวจจับรูปแบบการเกิดของเหตุการณ์ใด ๆ แล้วปรับตัวสู่สิ่งแวดล้อมที่เปลี่ยนไปได้

2.2.1.3 ระบบที่คิดอย่างมีเหตุผล (Systems that think rationally)

1. [ปัญญาประดิษฐ์คือ] การศึกษาความสามารถในด้านสติปัญญาโดยการใช้โมเดลการคำนวณ

2. [ปัญญาประดิษฐ์คือ] การศึกษาวิธีการคำนวณที่สามารถรับรู้ ใช้เหตุผล และกระทำ

หมายเหตุ คิดอย่างมีเหตุผล หรือคิดถูกต้อง เช่น ใช้หลักตรรกศาสตร์ในการคิดหาคำตอบอย่างมีเหตุผล เช่น ระบบผู้เชี่ยวชาญ

2.2.1.4 ระบบที่กระทำอย่างมีเหตุผล (Systems that act rationally)

1. ปัญญาประดิษฐ์คือการศึกษาเพื่อออกแบบเอเจนต์ที่มีปัญญา

2. ปัญญาประดิษฐ์เกี่ยวข้องกับพฤติกรรมที่แสดงปัญญาในสิ่งที่มนุษย์สร้างขึ้น

หมายเหตุ กระทำอย่างมีเหตุผล เช่น เอเจนต์ (โปรแกรมที่มีความสามารถในการกระทำหรือเป็นตัวแทนในระบบอัตโนมัติต่าง ๆ) สามารถกระทำอย่างมีเหตุผลเพื่อบรรลุเป้าหมายที่ได้ตั้งไว้ เช่น เอเจนต์ในระบบขับรถอัตโนมัติ ที่มีเป้าหมายว่าต้องไปถึงเป้าหมายในระยะทางที่สั้นที่สุด ต้องเลือกเส้นทางที่ไปยังเป้าหมายที่สั้นที่สุดที่เป็นไปได้ จึงจะเรียกได้ว่า เอเจนต์กระทำอย่างมีเหตุผล อีกตัวอย่างเช่น เอเจนต์ในเกมหมากรุก ที่มีเป้าหมายว่าต้องเอาชนะคู่ต่อสู้ ก็ต้องเลือกเดินหมากที่จะทำให้คู่ต่อสู้แพ้ให้ได้ เป็นต้น

2.2.2 ประเภทของปัญญาประดิษฐ์

ในปัจจุบันการศึกษาค้นคว้าเกี่ยวกับปัญญาประดิษฐ์มักจะครอบคลุมอยู่ในประเภทต่างๆ ดังต่อไปนี้ คือ

การประมวลภาษาธรรมชาติ (Natural Language Processing) เป็นกระบวนการที่พัฒนาให้ระบบคอมพิวเตอร์สามารถรับรู้และเข้าใจภาษาธรรมชาติของมนุษย์ ซึ่งหมายถึงภาษาพูดและภาษาเขียนที่ใช้อยู่ในชีวิตประจำวันได้ เช่น ภาษาอังกฤษ จีน ไทย เป็นต้น ซึ่งภาษาเหล่านี้ทำให้เรากับคอมพิวเตอร์สามารถติดต่อสื่อสารกันระหว่างการทำงานได้ และสิ่งที่สำคัญก็คือ ภาษาธรรมชาติจะช่วยให้การทำงานที่อาศัยเทคโนโลยีมาช่วยนั้นสามารถเติบโตและกระจายทั่วไปในโลกปัจจุบัน สำหรับปัญหาของการพัฒนาในด้านการประมวลภาษาธรรมชาติ คือ บางครั้งอาจมีการใช้คำผิด คำบางคำมีความหมายกำกวม หรือคำหลายคำมีความหมายเดียวกัน ซึ่งปัญหาเหล่านี้อาจทำให้ระบบคอมพิวเตอร์ไม่สามารถวิเคราะห์ความหมายของประโยคนั้นได้

ระบบภาพ (Vision System) เป็นระบบคอมพิวเตอร์ที่ปฏิบัติการเหมือนกับการมองเห็นด้วยสายตามนุษย์ แต่ที่จริงแล้วเป็นการลอกเลียนแบบการมองเห็นของมนุษย์นั่นเอง สำหรับลักษณะการทำงานจะมีการจดจำรูปแบบต่างๆ ที่บกพร่องหรือเกิดการเสียหาย และจะนำไปเก็บไว้ในฐานข้อมูล ตัวอย่างเช่น การพัฒนาระบบคอมพิวเตอร์ในการตรวจสอบความบกพร่องของชิ้นส่วนต่างๆ โดยคอมพิวเตอร์จะบันทึกภาพที่มีข้อบกพร่องของชิ้นส่วนนั้นไว้ในฐานข้อมูลเพื่อนำมาเปรียบเทียบกับภาพที่เห็นจริงในขณะนั้น

ระบบเครือข่ายเส้นประสาท (Neural Network) เป็นระบบคอมพิวเตอร์ที่สามารถบ่งบอกวัตถุ หรือรูปแบบในทางที่คล้ายกันกับที่กระบวนการที่สมองทำงานในการเรียนรู้สิ่งต่างๆ โดยระบบนี้จะถูกพัฒนาให้มีการจำลองการทำงานของสมองและระบบเส้นประสาทของมนุษย์ ซึ่งจะมีความสามารถในการสังเกต การเรียนรู้ การจดจำสิ่งต่างๆ ระบบคอมพิวเตอร์ชนิดนี้จะช่วยพัฒนาการปฏิบัติงานให้มีประสิทธิภาพการขึ้น จึงทำให้หน่วยงานต่างๆ โดยเฉพาะหน่วยงานทางธุรกิจเข้ามาสนใจในระบบคอมพิวเตอร์ชนิดนี้มากขึ้น

หุ่นยนต์ (Robotics) เป็นการออกแบบและสร้างสิ่งประดิษฐ์ที่เรียกว่า หุ่นยนต์ที่เลียนแบบมนุษย์ให้สามารถเข้าใจ และสั่งให้ทำงานตามที่ต้องการ โดยการสร้างหุ่นยนต์นี้ได้จำลองการทำงานของมนุษย์ โดยมีการออกแบบและกำหนดคำสั่งให้หุ่นยนต์สามารถทำตามได้ ปัจจุบันหุ่นยนต์มีการใช้กันมากในโรงงานอุตสาหกรรมโดยเฉพาะกับงานที่เสี่ยงอันตรายมาก หรือใช้กับงานที่ต้องเสียค่าใช้จ่ายแพงมากถ้าใช้มนุษย์ทำ

ระบบผู้เชี่ยวชาญ (Expert System) เป็นระบบคอมพิวเตอร์ที่สามารถรับรู้และทำงานเฉพาะด้านได้อย่างผู้เชี่ยวชาญ หรือเป็นโปรแกรมซึ่งประกอบไปด้วยตัวแทนของความรู้ของผู้เชี่ยวชาญในแบบซึ่งคนทั่วไปสามารถใช้ได้ เช่น ผู้เชี่ยวชาญในด้านการรักษาโรค วินิจฉัยโรค เป็นต้น

2.2.3 ระบบปัญญาประดิษฐ์ที่ใช้ในเกม

ระบบปัญญาประดิษฐ์ที่ใช้ในเกมจะแบ่งออกเป็น 2 ส่วน คือ ระบบปัญญาประดิษฐ์ที่ใช้การตัดสินใจ (Supervised Learning) เพื่อตอบสนองต่อเงื่อนไขภายใต้สถานการณ์ที่กำหนดไว้ และระบบปัญญาประดิษฐ์ที่ใช้การเรียนรู้จากค่าตอบแทนที่ได้ (Reinforcement Learning) เพื่อหาว่าการกระทำใดจะได้ผลดีที่สุด

การเรียนรู้แบบเสริมกำลัง (Reinforcement Learning) เป็นอีกหนึ่งสาขาย่อยของ Machine Learning ซึ่งมีส่วนประกอบพื้นฐานดังนี้

1. ผู้เรียน (Agent)
2. ชุดการกระทำ (Action)
3. สภาพแวดล้อม (Environment)

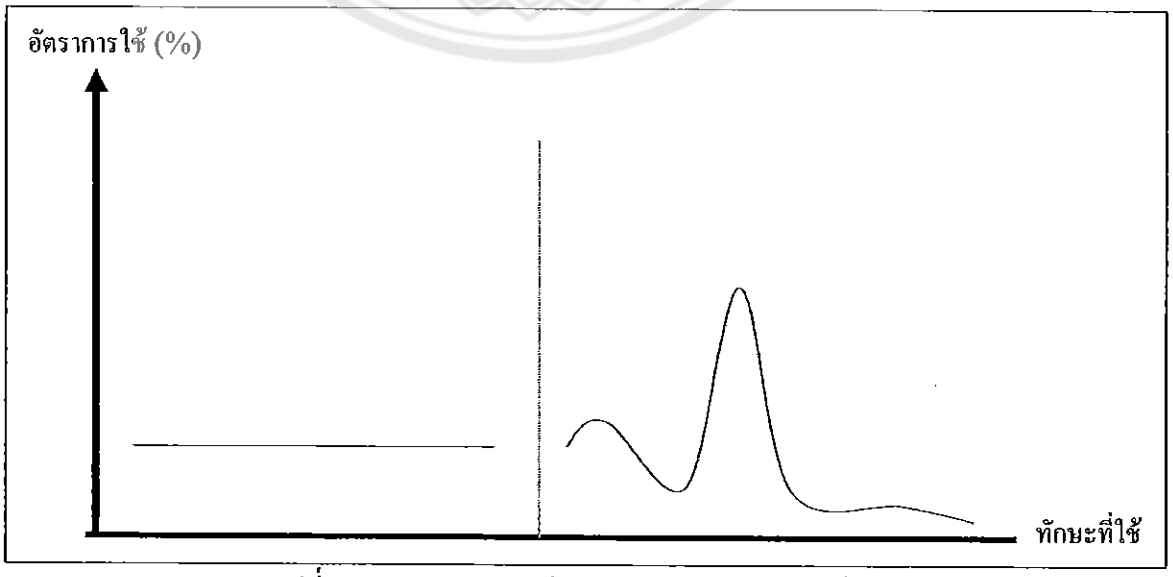
ในการเรียนรู้แบบเสริมกำลังนั้น กระบวนการส่วนใหญ่จะเกี่ยวข้องกับความสัมพันธ์ระหว่าง ผู้เรียน และ สภาพแวดล้อม โดยผู้เรียนจะพยายามเลือก การกระทำใด ๆ ในชุดของการกระทำที่เป็นไปได้ทั้งหมด ในสภาพแวดล้อมปัจจุบัน (State) หลังจากนั้น การกระทำที่ถูกเลือก จะมีผลทำให้ สภาพแวดล้อม เปลี่ยนแปลงไปและผู้เรียนก็จะได้รับ ค่าตอบแทน (Reward) ซึ่งขึ้นอยู่ว่าการกระทำดังกล่าวมีผลให้สภาพแวดล้อมเปลี่ยนแปลงไปในทางใด (ดีขึ้นหรือแย่ลง ถ้าดีขึ้นค่าตอบแทนอาจจะมาก แต่ถ้าแย่ลงค่าตอบแทนอาจจะน้อย)

จากค่าตอบแทนที่ได้รับ ผู้เรียนจะพยายามค้นหา นโยบาย (Policy) ในการเลือก การกระทำในสภาพแวดล้อมใด ๆ เพื่อให้ ค่าตอบแทน ในระยะยาวนั้นมีค่ามากที่สุด

สภาพแวดล้อม ในการเรียนรู้แบบเสริมกำลังนั้นมักจะอยู่ในรูปแบบของ Markov decision process (MDP) และการวิเคราะห์พฤติกรรมของการเรียนรู้ประเภทนี้ส่วนใหญ่จะใช้เทคนิคของ Dynamic Programming และเพื่อให้ง่ายต่อการวิเคราะห์และการนำไปใช้งาน ในบางครั้งการเปลี่ยนแปลงของสภาพแวดล้อมและการกระจายของค่าตอบแทนมักจะถูกกำหนดให้คงที่

การเรียนรู้แบบเสริมกำลังนี้แตกต่างจากการเรียนรู้แบบมีผู้สอน (Supervised learning) ตรงที่ตัวผู้เรียนเองจะไม่ได้เรียนรู้จากชุดของข้อมูลตัวอย่าง (Training set) แต่จะทำการโต้ตอบกับสภาพแวดล้อมที่ผู้เรียนกำลังทำงานอยู่โดยตรง ดังนั้นข้อมูลเดียวที่ผู้เรียนสามารถใช้ในการเรียนรู้ได้ก็คือค่าตอบแทนที่ได้รับเมื่อมีการเลือกการกระทำใดการกระทำหนึ่ง ซึ่งเราสามารถมองการเรียนรู้แบบนี้ได้เป็น การเรียนรู้แบบลองผิดลองถูก (trial-and-error)

โดยส่วนใหญ่แล้วจะมีการนำการเรียนรู้แบบนี้ไปใช้ในงานที่เกี่ยวข้องกับการควบคุมหรือเกม เช่น การควบคุมหุ่นยนต์ หมากรุก เป็นต้น



รูปที่ 2.2 แสดงการเรียนรู้ของระบบปัญญาประดิษฐ์

2.3 ภาษา C++

2.3.1 ประวัติและความเป็นมา

ภาษาซีพลัสพลัส (C++) เป็นภาษาโปรแกรมคอมพิวเตอร์อเนกประสงค์ มีโครงสร้างภาษาที่มีการจัดชนิดข้อมูลแบบสแตติก (Statically typed) และสนับสนุนรูปแบบการเขียนโปรแกรมที่หลากหลาย (Multi-paradigm language) ได้แก่ การโปรแกรมเชิงกระบวนคำสั่ง การนิยามข้อมูลการโปรแกรมเชิงวัตถุ และการโปรแกรมแบบเจเนริก (Generic programming) ภาษาซีพลัสพลัสเป็นภาษาโปรแกรมเชิงพาณิชย์ที่นิยมมากภาษาหนึ่งนับตั้งแต่ช่วงทศวรรษ 1990

Bjarne Stroustrup จากเบลล์แล็บส์ (Bell Labs) เป็นผู้พัฒนาภาษาซีพลัสพลัส (จากเดิมใช้ชื่อ "C with classes") ในปี ค.ศ. 1983 เพื่อพัฒนาภาษาซีดั้งเดิม สิ่งที่พัฒนาขึ้นเพิ่มเติมนั้นเริ่มจากการเพิ่มเติมการสร้างคลาสจากนั้นก็เพิ่มคุณสมบัติต่างๆ ตามมา ได้แก่ เวอร์ชวลฟังก์ชัน การโอเวอร์โหลดโอเปอเรเตอร์ การสืบทอดหลายสาย เทมเพลต และการจัดการเอกเซพชัน มาตรฐานของภาษาซีพลัสพลัสได้รับการรับรองในปี ค.ศ.1998 เป็นมาตรฐาน ISO/IEC 14882:1998 เวอร์ชันล่าสุดคือเวอร์ชันในปี ค.ศ.2003 ซึ่งเป็นมาตรฐาน ISO/IEC 14882:2003 ในปัจจุบันมาตรฐานของภาษาในเวอร์ชันใหม่ (รู้จักกันในชื่อ C++0x) กำลังอยู่ในขั้นพัฒนา

2.3.2 แนวคิดที่สำคัญของการเขียนโปรแกรมเชิงวัตถุ

คลาส (Class) ประเภทของวัตถุ เป็นการกำหนดว่าวัตถุจะประกอบไปด้วย ข้อมูล (Data) หรือคุณสมบัติ (Property) และ พฤติกรรม (Behavior) หรือการกระทำ (Method) อะไรบ้าง ซึ่งคลาส (เช่น มนุษย์) เป็นโครงสร้างพื้นฐานของการเขียนโปรแกรมเชิงวัตถุ

วัตถุ (Object) โดยมากจะเรียกว่า อ็อบเจกต์ คือ ตัวตน (Instance) ของ คลาส (เช่น นาย. ก นาย. ข) ซึ่งจะเกิดขึ้นระหว่าง run-time โดยแต่ละ อ็อบเจกต์ จะมีข้อมูลเฉพาะของตัวเอง ทำให้อ็อบเจกต์ แต่ละ อ็อบเจกต์ ของ คลาส ซึ่งใช้ source code เดียวกันมีคุณลักษณะและคุณสมบัติที่แตกต่างกัน

Encapsulation การปิดบังข้อมูล เป็นวิธีการกำหนดสิทธิในการเข้าถึงข้อมูล หรือการกระทำกับ อ็อบเจกต์ ของ คลาสนั้น ๆ ทำให้แน่ใจได้ว่าข้อมูลของอ็อบเจกต์นั้นจะถูกเปลี่ยนแปลงแก้ไขผ่านทาง Methods หรือ Properties ที่อนุญาตเท่านั้น (เช่น การกำหนดตำแหน่งทางการเมือง เป็น Public method ที่ผู้อื่นสามารถกระทำได้ ส่วนการลาออกจากตำแหน่ง เป็น Private method ที่มีแต่ object ของ คลาส เท่านั้นที่จะสามารถทำได้ แต่การกดดันและการขับไล่สามารถสร้าง Data ที่อาจจะส่งผลเกิดการลาออกได้เช่นกัน)

Inheritance การสืบทอดคุณสมบัติ เป็นวิธีการสร้าง คลาสย่อย ที่เรียกว่าซับคลาส (subclass) ซึ่งจะเป็นกำหนดประเภทของวัตถุให้จำเพาะเจาะจงขึ้น ซึ่ง ซับคลาส จะได้รับถ่ายทอดคุณสมบัติต่าง ๆ มาจากคลาสหลักด้วย (เช่น คลาส มนุษย์ สืบทอดมาจาก คลาส สิ่งมีชีวิต)

Abstraction นามธรรม เป็นการแสดงถึงคุณลักษณะและพฤติกรรมของ object เท่าที่จำเป็นต้องรับรู้และใช้งาน โดยซ่อนส่วนที่เหลือเอาไว้เพื่อไม่ให้เกิดความสับสน เช่น ตามปกติแล้ว นาย ก จัดเป็นตัวตนของ คลาส มนุษย์ ซึ่งจะมีพฤติกรรม การกระทำทุกอย่างที่ตามที่กำหนดไว้ตามโครงสร้างของ คลาส มนุษย์ แต่ในบางกรณีที่นำไปใช้งาน เราไม่ต้องการให้เกิดการสับสนต่อการใช้งานหรือการจัดประเภทมากเราสามารถจัดการหรือใช้งาน object นาย ก ให้อยู่ในรูปของสิ่งมีชีวิตได้

Polymorphism ภาวะที่มีหลายรูปแบบ เป็นวิธีการกำหนดรูปแบบการกระทำที่เหมือนกัน แต่ได้ผลที่แตกต่างกัน เช่น การเปล่งเสียง เป็น method หลักของ คลาส สิ่งมีชีวิต ซึ่งมีคลาส มนุษย์ และคลาสสุนัข เป็น ซับคลาส แต่ผลของการเปล่งเสียงของอ็อบเจกต์จากคลาสทั้งสองจะออกมาไม่เหมือนกัน

หลักการของ Object Oriented Programming (OOP) คือ การมองหน่วยต่าง ๆ ของภาษาโปรแกรม (Entity) ให้เป็นวัตถุ (Object) จะเห็นได้ว่าวัตถุแต่ละตัว จะมีลักษณะเฉพาะตัว ที่ไม่เหมือนกัน เช่น Button จะมีลักษณะเป็นปุ่มกด รูปสีเหลี่ยม แตกต่างจาก Radio Button ซึ่งมีลักษณะเป็นปุ่มวงกลม มีข้อความด้านขวามือ และหน้าที่ก็ยังแตกต่างกันเป็นต้น

2.3.3 การกำหนดคุณลักษณะของโปรแกรมเชิงวัตถุ

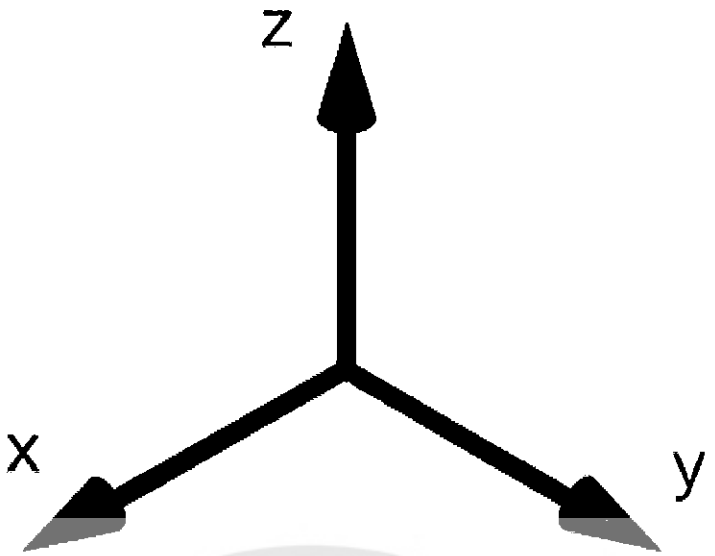
จะเห็นได้ว่า วัตถุ แต่ละตัว จะมีลักษณะเฉพาะตัว ที่ไม่เหมือนกัน เช่น Button จะมีลักษณะเป็นปุ่มกด รูปสี่เหลี่ยม แตกต่างจาก Radio Button ซึ่งมีลักษณะเป็นปุ่มวงกลม มีข้อความด้านขวามือ และหน้าที่ก็ยังแตกต่างกัน เป็นต้น

เมื่อสนใจที่ วัตถุ ชนิดเดียวกันจะเห็นว่าลักษณะเฉพาะอย่างเดียวกัน ก็อาจแตกต่างกันได้ใน วัตถุ คนละตัว เช่น Button 2 ตัว อาจมีข้อความที่ไม่เหมือนกัน ได้ เราเรียกลักษณะต่างๆ ที่มีในตัว วัตถุ นี้ว่า Property เช่น Button จะมี Poperty ที่ชื่อว่า Caption เป็นตัวเก็บข้อความที่แสดงบนปุ่ม ซึ่ง วัตถุ ชนิด Button จะมี Poperty ชนิดเดียวกัน แต่อาจมีค่าที่เก็บใน Poperty ที่แตกต่างกันได้

2.4 ระบบภาพ 3 มิติ

ระบบแกน 3 มิติจะมีอยู่ 2 แบบ คือแบบ มือซ้าย และแบบ มือขวา ในระบบทั้ง 2 อย่าง แกน X จะชี้ไปทางด้านขวา และแกน Y จะชี้ขึ้นบน เราสามารถช่วยจำเรื่องแกน Z ได้โดยเอาแขนซ้ายหรือขวา ยื่นขนานไปกับแกน X ทำและหมุนมือไปทางแกน Y นิ้วโป้งจะเป็นตัวชี้แกน Z

โดยค่าในเมทริกซ์ที่แสดงสำหรับลักษณะการหมุนตามเข็มนาฬิกา ในตำแหน่งที่เรามองตรง ๆ ไปที่แกนหมุน (มุมมองที่แกนหมุนตั้งตรงกับจุด Origin) และมองเหนือแกน + ทอดยาวไปสู่แกน - โดยที่ระบบแกนเป็นระบบมือซ้าย (Left-handed coordinate systems)



รูปที่ 2.3 แสดงรูประบบแกน 3 มิติ

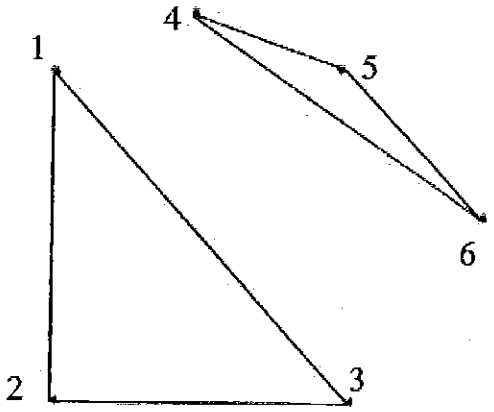
2.4.1 โพลีกอน (Polygon)

โพลีกอน คือรูปร่างแบบปิดในระบบพื้นที่ 3 มิติ ลองสมมุติถึงจุดที่อยู่ตามที่ต่าง ๆ ในอากาศ เราลากเส้นจากจุดที่ 1 ไปยังจุดที่ 2 และจากจุดที่ 2 ไปยังจุดที่ 3 ไปเรื่อย ๆ จนจุดสุดท้ายลากกลับมายังจุดที่ 1 เราจะได้ Polygon รูปปิดมา 1 ชิ้น ที่ภายใน Polygon นั้นจะมีพื้นที่ปิดจำนวนหนึ่งนั่นเอง เพราะฉะนั้นจำนวนจุด (Vertices) ที่น้อยที่สุดที่จะสามารถสร้าง Polygon 1 ชิ้นได้ก็คือ 3 จุดนั่นเอง (3 Vertices) นั่นคือโพลีกอนพื้นฐานจะเป็นรูปสามเหลี่ยม (จุด 3 จุด เชื่อมต่อกัน)

ใน Direct3D และ OpenGL จะวาดได้แต่ Polygon 3 เหลี่ยมเท่านั้น นั่นเพราะต้องการลดความซับซ้อน และ 3 Vertices นั่นจะได้เป็นระนาบที่เรียบแน่นอน ซึ่งในรูปแบบอื่น ๆ นั้นก็สามารถทำได้โดยการ เอา Polygon 3 เหลี่ยม มาต่อกัน

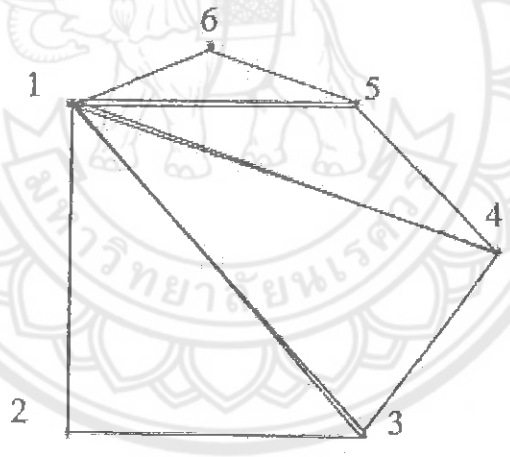
วิธีการวาดโพลีก้อนของทั้ง DirectX และ OpenGL จะนับ Vertex ที่เก็บไว้ แล้วเชื่อมแต่ละจุดในรูปแบบต่างๆกัน ดังนี้

TriangleList จะนับ Vertex 3 จุด แล้ววาดสามเหลี่ยม 1 รูป วาด 1 รูป ทุก ๆ 3 Vertex จนกว่าจะหมด ผลที่ได้จะได้สามเหลี่ยมที่ไม่ต่อกัน ดังรูป จะวาดสามเหลี่ยมแรก จากจุด {1 2 3} และสามเหลี่ยมต่อมา จากจุด {4 5 6}



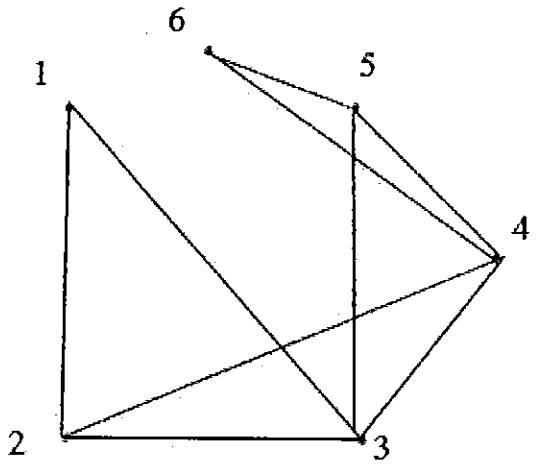
รูปที่ 2.4 แสดงรูปสามเหลี่ยมที่ไม่ต่อกัน

TriangleFan จะนับ Vertex แรกสุดเป็นจุดศูนย์กลาง แล้วจะวาดสามเหลี่ยมจากจุดถัดไปทีละ 2 จุด แต่จะเริ่มจากจุดแรกก่อน และกลับไปปิดสามเหลี่ยมที่จุดแรกเสมอ ดังรูป จะวาดสามเหลี่ยมจากจุด {1 2 3} {1 3 4} {1 4 5} และ {1 5 6}



รูปที่ 2.5 แสดงรูปสามเหลี่ยมที่ติดต่อกัน

TriangleStrip จะนับ Vertex ซ้อนกัน ทีละ 3 จุด โดยเลื่อนชุดสามเหลี่ยมที่จะวาดสามเหลี่ยมต่อไป ไปทีละ 1 จุด ดังรูป จะวาดสามเหลี่ยมจากจุด {1 2 3} {2 3 4} {3 4 5} {4 5 6}



รูปที่ 2.6 แสดงรูปสามเหลี่ยมที่เกิดจาก vertex ที่ซ้อนทับกัน

บทที่ 3
ขั้นตอนการดำเนินงาน

ในบทนี้จะกล่าวถึงขั้นตอนการดำเนินงานต่าง ๆ ตั้งแต่ศึกษาเครื่องมือที่ใช้ในการสร้างเกม การออกแบบโครงสร้างของเกม ออกแบบระบบปัญญาประดิษฐ์ จนถึงการเขียนโปรแกรมเพื่อสร้างเกมขึ้นมา โดยสามารถแบ่งออกเป็นขั้นตอนย่อยๆ ได้ดังนี้

- ขั้นตอนการรวบรวมข้อมูลในการสร้างเกม
- ขั้นตอนการศึกษาวิธีการสร้างเกมและโครงสร้างของเกม
- ขั้นตอนการออกแบบโครงสร้างของเกม
- ขั้นตอนการออกแบบเงื่อนไขของระบบปัญญาประดิษฐ์
- ขั้นตอนการพัฒนาเกม

3.1 ขั้นตอนการรวบรวมข้อมูลในการสร้างเกม

จากการศึกษาและรวบรวมข้อมูลของการสร้างเกมในปัจจุบัน โดยทั่วไปผู้พัฒนาเกมส่วนใหญ่จะใช้ Game Engine มาเป็นเครื่องมือช่วยในสร้างเกม เพื่อจัดการกับการทำงานพื้นฐานต่าง ๆ เช่น การรับค่า Input Output การแสดงภาพทั้งสองมิติ และสามมิติ ระบบเสียง เป็นต้น ซึ่งในปัจจุบันมี Game Engine อยู่มากมาย แต่ที่ผู้พัฒนาเกมนิยมใช้ในการสร้างเกมสามมิติได้แก่ Irrlicht Game Engine และ CDX Game Engine

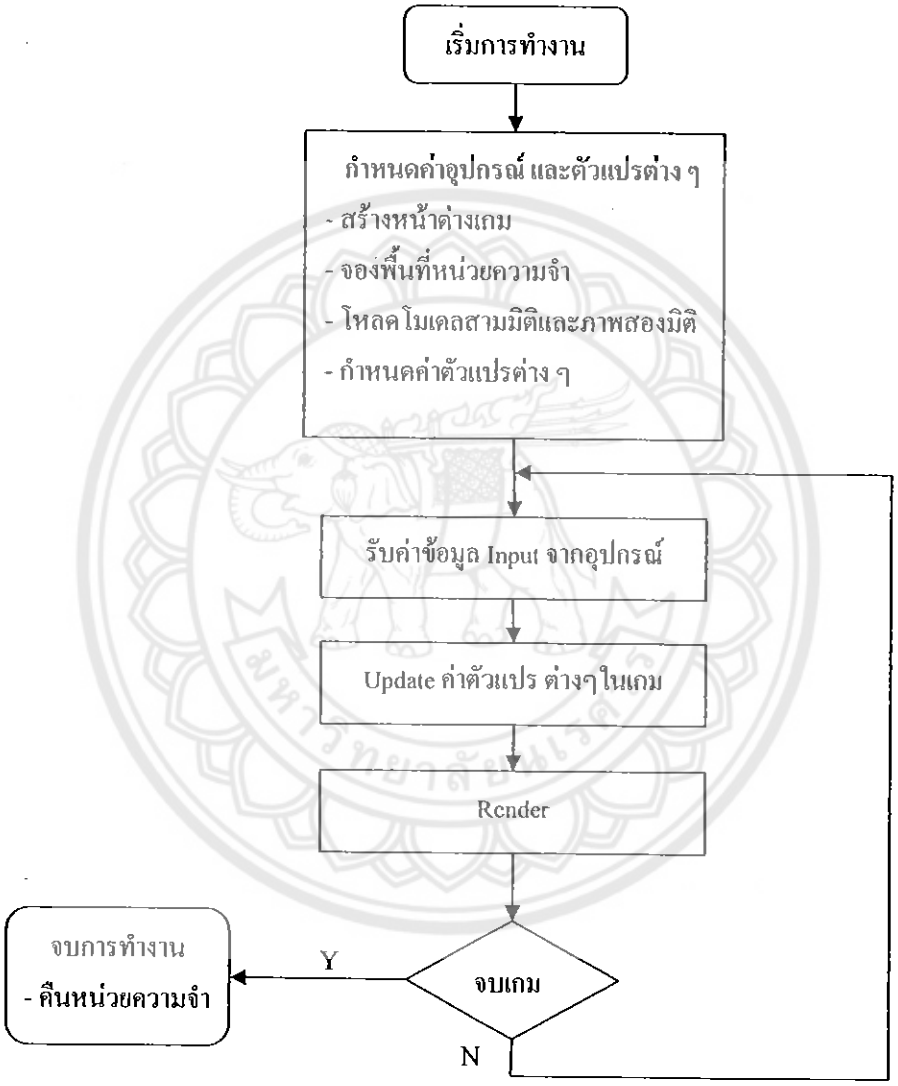
ตารางที่ 3.1 ตารางเปรียบเทียบ Irrlicht Game Engine และ CDX Game Engine

	ข้อดี	ข้อเสีย
Irrlicht Game Engine	- เป็น Open Source - สามารถทำงานได้หลายระบบปฏิบัติการ - มีการพัฒนาอยู่ตลอดเวลา - ความสวยงามในการแสดงผลสูง	- ไม่มีตัวช่วยในการเชื่อมต่อ Network - ค้นหาความรู้เพิ่มเติมได้ยาก เพราะเป็น Engine ใหม่ - ตัวช่วยในการแสดงภาพสองมิติมีน้อย - ตัวช่วยในการทำเกมแนว RPG มีน้อย
CDX Game Engine	- เป็น Open Source - เป็น Engine ที่สามารถเรียนรู้ได้ง่าย - สามารถประมวลภาพสองมิติได้ดีและ - มีเครื่องมือช่วยในการแสดงภาพสองมิติที่ดี	- เป็น Engine เก่าที่ไม่มีการพัฒนาต่อแล้ว - ตัวช่วยในการประมวลผลภาพสามมิติมีน้อย

3.2 ขั้นตอนการศึกษาวิธีการสร้างเกมและโครงสร้างของเกม

3.2.1 การศึกษาการทำงานของ Irrlicht Game Engine

จากข้อมูลข้างต้นทางผู้จัดทำได้เลือกใช้ Irrlicht Game Engine เป็นตัวช่วยในการสร้างเกม และได้ศึกษาระบบการทำงานในส่วนต่างๆของ Irrlicht Game Engine ซึ่งสามารถสรุปโครงสร้างการทำงาน ได้ดังนี้



รูปที่ 3.1 Flowchart แสดงโครงสร้างการทำงานของ Irrlicht Game Engine

หลักการทำงานของ Irrlicht Game Engine จะเป็นลักษณะของลูปโปรแกรม โดยจะเริ่มจัดการทรัพยากรต่างๆในเครื่อง แล้วสร้างหน้าต่างเกมและกำหนดค่าตัวแปรต่าง ๆ โดยใน Irrlicht Game Engine จะมีคลาสให้เรียกเพื่อเก็บตัวแปรชนิดต่าง ๆ อยู่แล้ว ต่อมาจะเข้าส่วนของลูปโปรแกรมโดยจะจัดการกับการรับค่า Input ต่าง ๆ แล้วนำมา Update ค่าตัวแปรต่าง ๆ ตามเงื่อนไขของเกม โดยมากจะเป็นตัวแปรที่เป็นภาพสองมิติ ภาพสามมิติ และตัวแปรต่าง ๆ ในเกม แล้วจึง

แสดงผล และจะทำงานไปเรื่อย ๆ จนกว่าผู้ใช้จะทำการจบการทำงาน และเมื่อจบการทำงานก็จะมีการคืนค่าหน่วยความจำที่ใช้

3.2.2 การศึกษาโครงสร้างของเกม

โครงสร้างเกม คือองค์ประกอบภายในเกมต่าง ๆ ที่นำมาแสดงให้ผู้เล่นเข้าใจ และดึงดูดให้ผู้เล่นมีอารมณ์ร่วม ซึ่งจะต้องมีการการจัดการกับองค์ประกอบต่างๆ ให้เสมือนเป็นการจำลองสถานการณ์ต่าง ๆเพื่อให้ผู้เล่นได้แก้ไขปัญหา ตามกฎเกณฑ์ และเป้าหมายของตัวเกม โดยสามารถแบ่งองค์ประกอบต่าง ๆ ตามลักษณะการทำงานได้ดังนี้

1. ภาพสองมิติ
2. โมเดลสามมิติ
3. ระบบปัญญาประดิษฐ์
4. ฐานข้อมูล
5. เสียง
6. ข้อความหรือตัวอักษร

3.3 ขั้นตอนการออกแบบโครงสร้างของเกม

3.2.1 การวิเคราะห์การออกแบบโครงสร้างของเกม

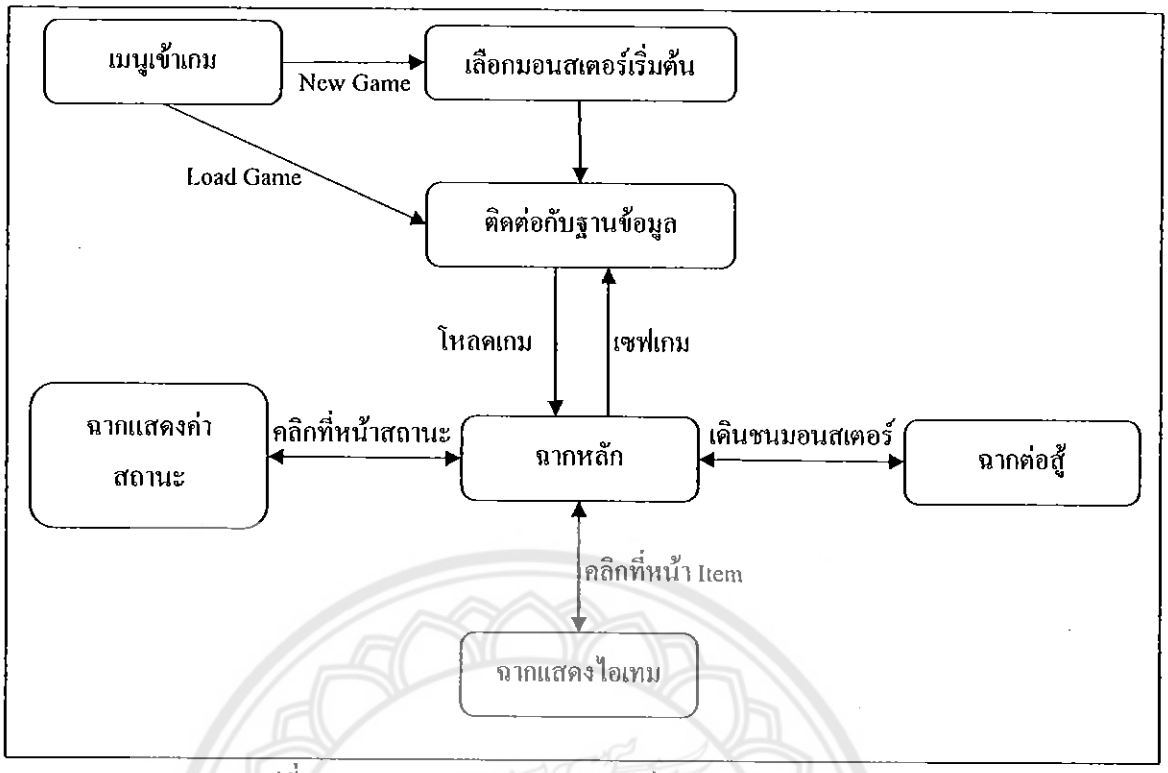
เกม 3 มิติที่สร้างเป็นเกมประเภท RPG (Role Playing Game) ที่มีระบบการเล่นโดยผู้เล่นมีการเคลื่อนไหวได้อย่างอิสระในฉากทั่วไป และผู้เล่นจะมีการเลี้ยงมอนสเตอร์เพื่อนำไปต่อสู้กับศัตรู และในการต่อสู้นั้นจะใช้ ระบบปัญญาประดิษฐ์ (Artificial Intelligent) ในการตัดสินใจโดยเรียนรู้จากข้อมูลที่ได้เก็บรวบรวมไว้ผ่านมาหลาย ๆ ครั้ง เพื่อหาว่าทักษะใดที่มอนสเตอร์ใช้นั้น ทำให้เกิดประสิทธิภาพได้สูงสุด โดยผู้เล่นสามารถเพิ่มความสามารถของมอนสเตอร์ได้จากการกินผลไม้ชนิดต่าง ๆ และจากการเรียนรู้ผ่านการต่อสู้กับศัตรู

3.2.2 การออกแบบโครงสร้างของเกมโดยรวม

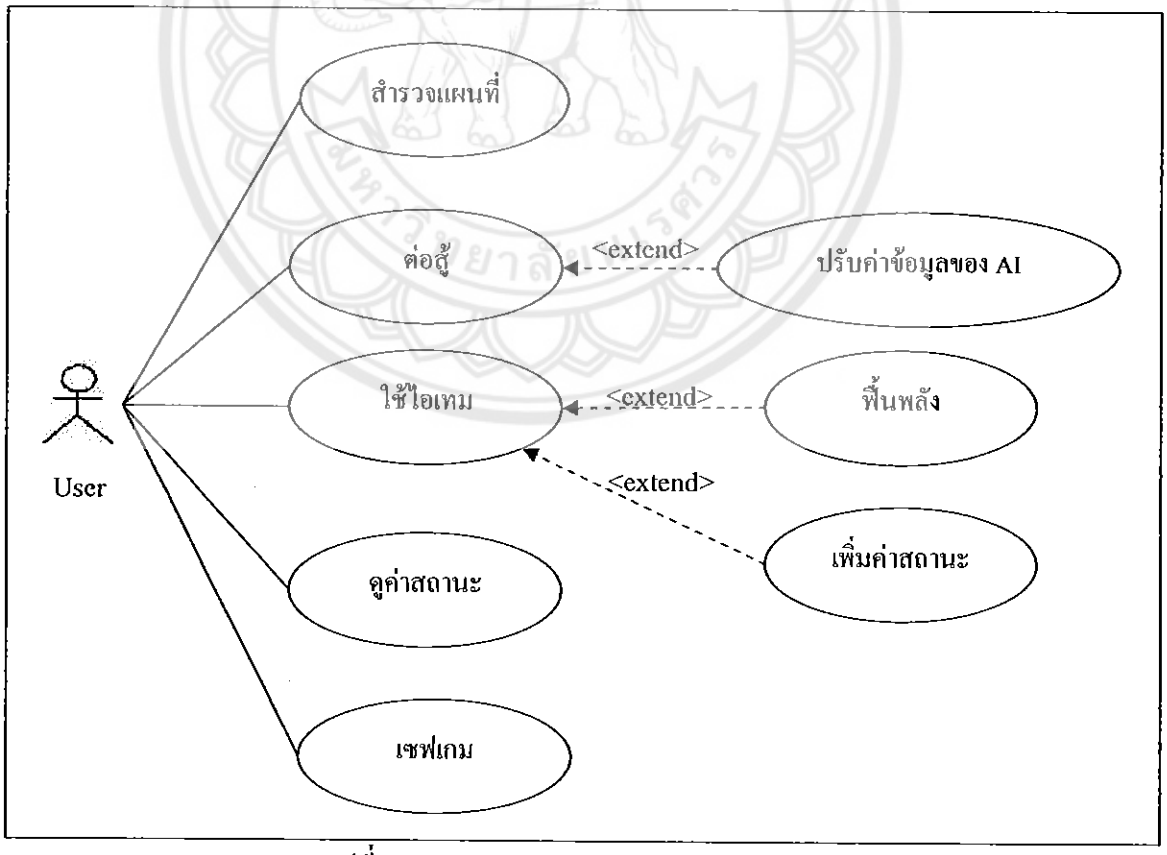
โครงสร้างของเกมโดยรวมได้แบ่งออกเป็น 4 ฉาก ดังนี้

- ฉากหน้าเมนูเข้าเกม เป็นฉากเริ่มต้นที่แสดงตอนเกม
- ฉากหลัก เป็นฉากที่ผู้เล่นเคลื่อนไหวไปมาได้อย่างอิสระ
- ฉากต่อสู้ เป็นฉากที่นำมอนสเตอร์มาต่อสู้โดยใช้เงื่อนไขของระบบปัญญาประดิษฐ์
- ฉากแสดงสถานะและไอเทม เป็นฉากที่ผู้เล่นสามารถดูค่าสถานะ และใช้ไอเทมได้

ซึ่งแต่ละฉากมีความสัมพันธ์กันดังนี้



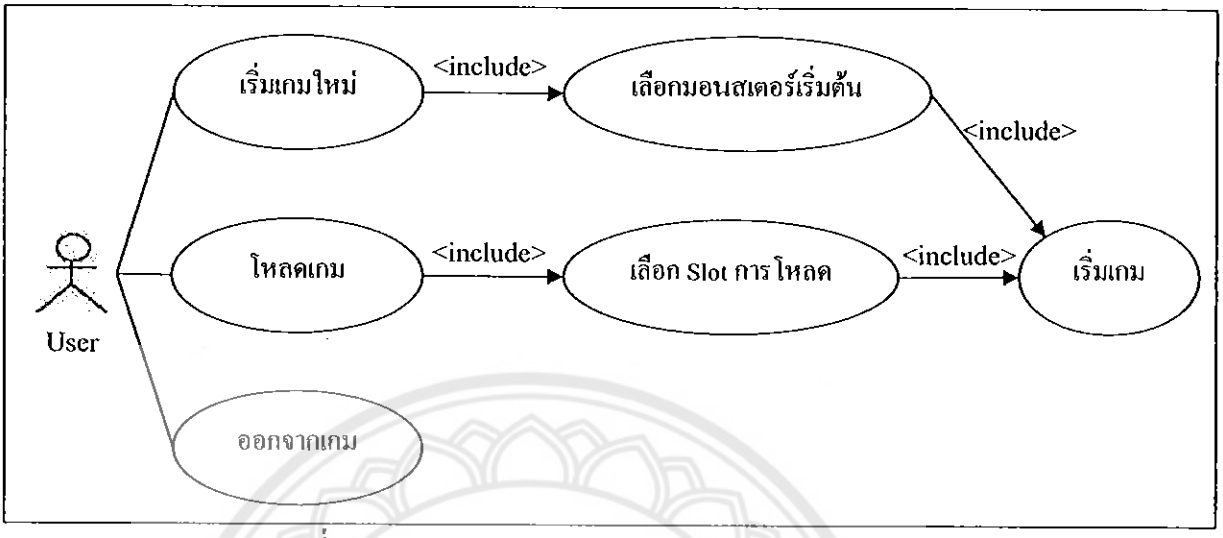
รูปที่ 3.2 รูปภาพแสดงความสัมพันธ์ของฉากภายในเกม



รูปที่ 3.3 Use Case Diagram ของฉากหลัก

3.2.3 การออกแบบส่วนหน้าเมนูเข้าเกม

สามารถเขียนอยู่ในรูป Use Case Diagram ได้ดังนี้



รูปที่ 3.4 Use Case Diagram แสดงหน้าเมนูเข้าเกม

ในส่วนของการเข้าเกม โดยจะมีตัวเลือกให้ผู้เล่นเลือก ได้แก่ New Game, Load Game และ Exit

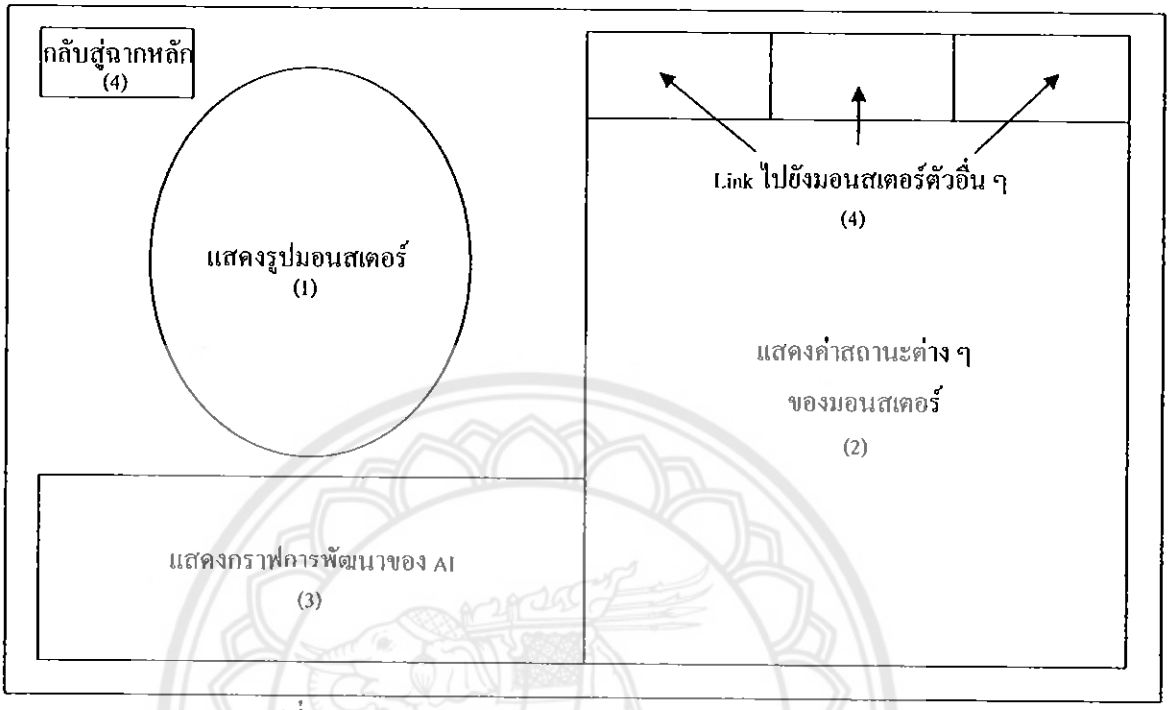
ในส่วนของ New Game โปรแกรมจะทำการโหลดค่าตั้งต้นต่าง ๆ จากไฟล์ Default แล้วผู้เล่นจะเข้าสู่ฉากเริ่มเกม ผู้เล่นจะต้องทำการเลือกมอนสเตอร์ 3 ตัวจากทั้งหมด 4 ตัวเพื่อใช้ในการเล่น หลังจากนั้นผู้เล่นจะสามารถเคลื่อนไหวได้อย่างอิสระ และเข้าต่อสู้กับศัตรูที่อยู่ทั่ว ๆ ไปตามฉากได้

ในส่วนของ Load Game โปรแกรมจะทำการโหลดค่าตั้งต้นต่าง ๆ จากไฟล์ที่ผู้เล่นได้ทำการบันทึกไว้ หลังจากนั้นผู้เล่นจะสามารถทำการเล่นต่อจากจุดเริ่มต้น โดยใช้ค่าต่าง ๆ ที่ได้ทำการบันทึกไว้

ในส่วนของ Exit จะเป็นการปิดโปรแกรมเพื่อออกจากเกม

3.2.4 การออกแบบส่วนแสดงค่าสถานะและไอเทม

ออกแบบส่วนแสดงค่าสถานะ ไว้ดังนี้



รูปที่ 3.5 รูปภาพแสดงรูปแบบของฉากแสดงค่าสถานะ

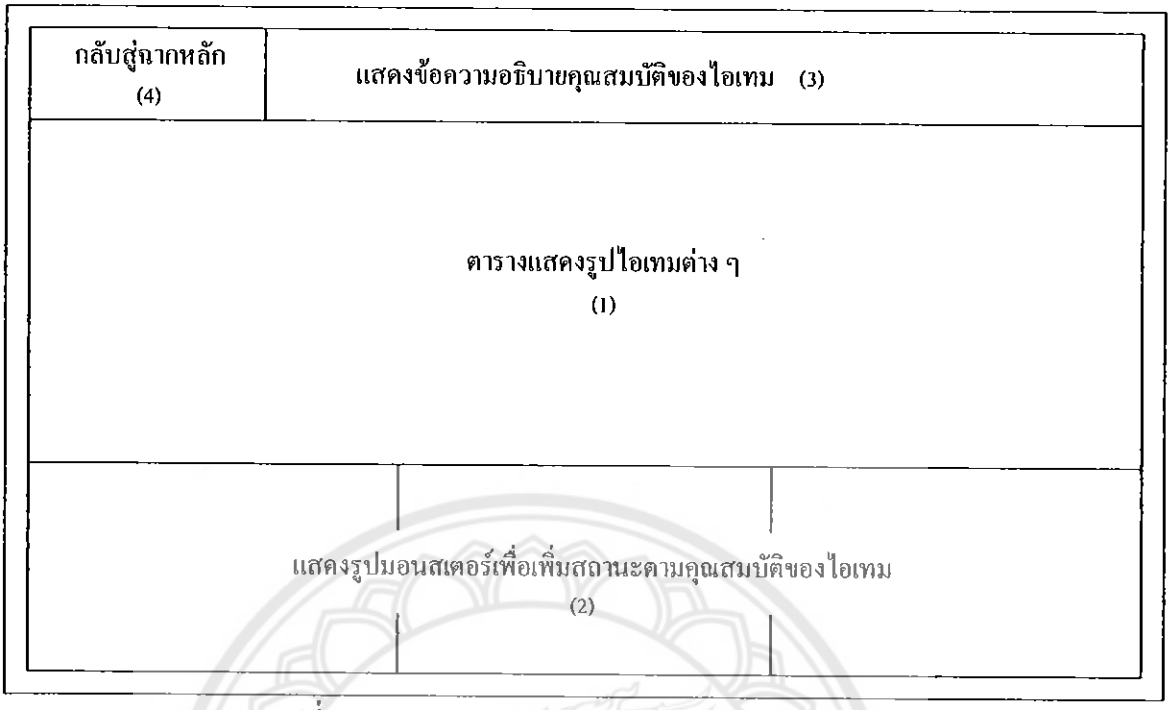
เมื่อคลิกที่รูปมอนสเตอร์ที่ฉากหลักจะเข้าสู่ฉากแสดงค่าสถานะ ซึ่งประกอบไปด้วย

- รูปแสดงมอนสเตอร์ (1) จะแสดงมอนสเตอร์ 3 มิติที่ผู้เล่นเลือก
- ส่วนแสดงค่าสถานะของมอนสเตอร์ (2) จะแสดงค่าสถานะของมอนสเตอร์ที่เลือก
- ส่วนแสดงกราฟการพัฒนาของระบบปัญญาประดิษฐ์ (3)
- ส่วนเชื่อมต่อไปยังมอนสเตอร์ตัวอื่น ๆ และกลับสู่ฉากหลัก (4)

ค่าสถานะของมอนสเตอร์ จะมีดังนี้

- HP (Health Point) คือ ค่าพลังชีวิต ถ้าเหลือเท่ากับ 0 มอนสเตอร์ตัวนั้นจะตาย
- SP (Special Point) คือ ค่าพลังพิเศษ เพื่อใช้ทักษะพิเศษอื่น ๆ
- AT (Attack) คือ ค่าพลังโจมตี
- DF (Defend) คือ ค่าพลังป้องกัน
- SD (Speed) คือ ค่าความเร็ว
- S.AT (Special Attack) คือ ค่าพลังโจมตีพิเศษ
- S.DF (Special Defend) คือ ค่าพลังป้องกันพิเศษ
- ELE (Element) คือ ธาตุของมอนสเตอร์ ได้แก่ ดิน น้ำ ลม ไฟ

ออกแบบส่วนแสดงไอเทม ไว้ดังนี้



รูปที่ 3.6 รูปภาพแสดงรูปแบบของฉากแสดงไอเทม

เมื่อคลิกที่รูปกระเป๋าที่ฉากหลักจะเข้าสู่ฉากแสดงไอเทม ซึ่งประกอบไปด้วย

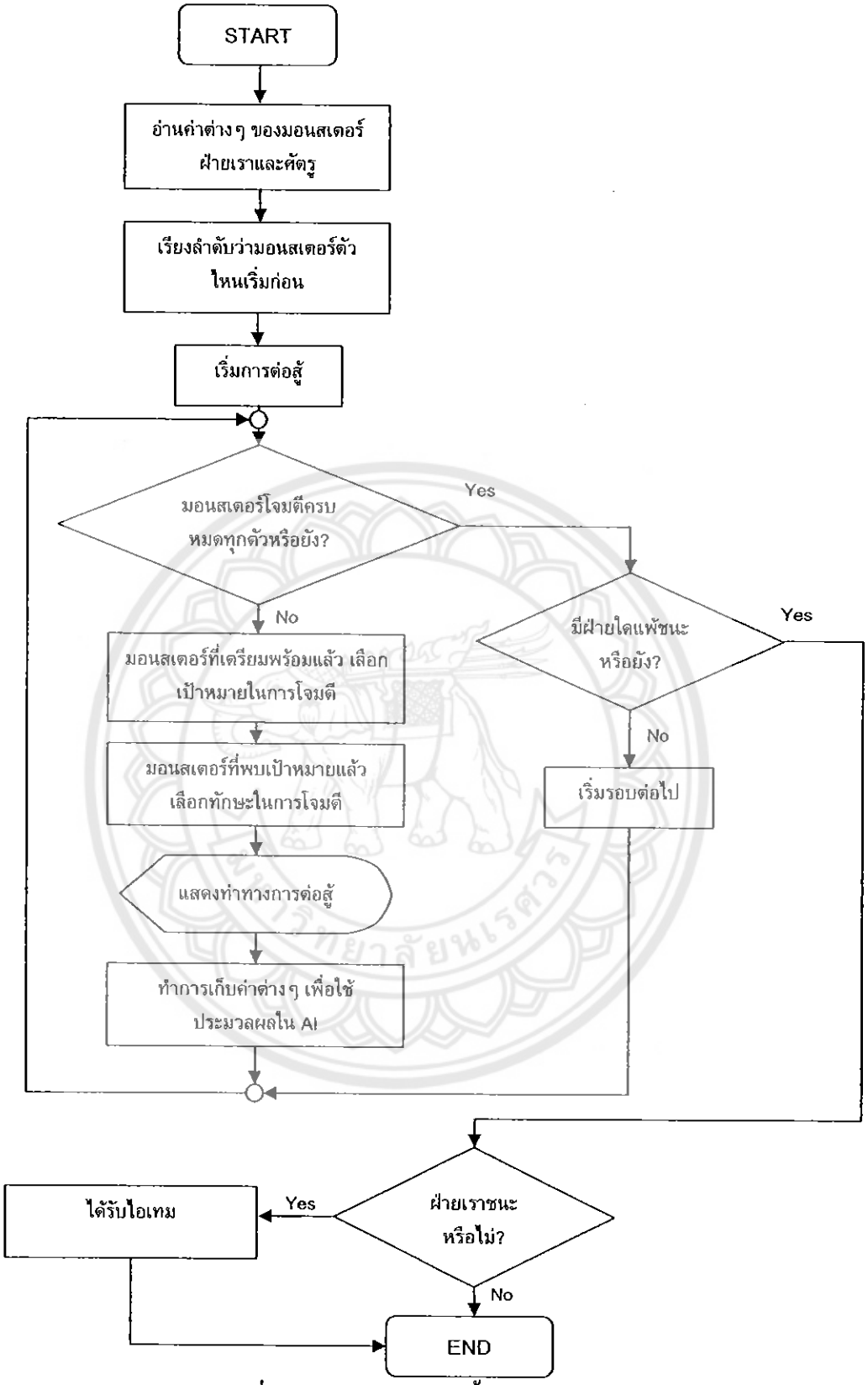
- ตารางแสดงรูปไอเทมต่าง ๆ (1) จะแสดงรูปไอเทม และจำนวนที่มีอยู่
- ส่วนแสดงรูปมอนสเตอร์ (2) เพื่อเลือกมอนสเตอร์ที่ต้องการจะเพิ่มสถานะตามคุณสมบัติ
- ส่วนแสดงข้อความอธิบายคุณสมบัติของไอเทมที่เลือก (3)
- ส่วนเชื่อมต่อกลับสู่ฉากหลัก (4)

คุณสมบัติของไอเทม จะมีดังนี้

- ไอเทมเพิ่มพลังชีวิต โดยจะเพิ่มค่า HP
- ไอเทมเพิ่มพลังพิเศษ โดยจะเพิ่มค่า SP
- ไอเทมเพิ่มสถานะต่าง ๆ ของมอนสเตอร์ โดยจะเพิ่มค่า AT DF SD S.AT และ S.DF

3.2.5 การออกแบบส่วนต่อสู้

สามารถเขียนอยู่ในรูป Flowchart ได้ดังนี้



รูปที่ 3.7 Flowchart แสดงขั้นตอนการต่อสู้

เมื่อทำการเดินชนมอนสเตอร์แล้วตัวเกมจะตัดเข้าสู่ฉากต่อสู้ ซึ่งผู้เล่นจะไม่สามารถบังคับมอนสเตอร์ฝ่ายผู้เล่นได้ จนกระทั่งรู้ผลแพ้ชนะของมอนสเตอร์ฝ่ายผู้เล่นหรือมอนสเตอร์ฝ่ายศัตรูเสียก่อน ถ้ามอนสเตอร์ฝ่ายผู้เล่นเป็นผู้ชนะจะได้รับไอเทมมาจำนวนหนึ่ง แต่ถ้าเป็นฝ่ายแพ้มอนสเตอร์ฝ่ายผู้เล่นจะถูกสุ่มลดค่าสถานะอย่างใดอย่างหนึ่ง

ค่าสถานะของมอนสเตอร์ฝ่ายศัตรูจะได้มาจากการสุ่มระดับ และการสุ่มค่าสถานะของมอนสเตอร์ฝ่ายผู้เล่น ดังนี้

- มอนสเตอร์ระดับต้น จะมีค่าสถานะน้อยกว่ามอนสเตอร์ฝ่ายผู้เล่นอยู่มาก
- มอนสเตอร์ระดับกลาง จะมีค่าสถานะใกล้เคียงกับมอนสเตอร์ฝ่ายผู้เล่น
- มอนสเตอร์ระดับสูง จะมีค่าสถานะมากกว่ามอนสเตอร์ฝ่ายผู้เล่น

รูปแบบการต่อสู้ กำหนดไว้ดังนี้

เมื่อมอนสเตอร์ตัวใด ยังไม่มีคู่ต่อสู้ จะถูกกำหนดสถานะเป็น “Alone” เมื่อมอนสเตอร์ตัวใด ได้เลือกคู่ต่อสู้แล้ว ทั้งตัวมอนสเตอร์และคู่ต่อสู้จะถูกกำหนดสถานะเป็น “Lock”

เมื่อมีคู่มอนสเตอร์ที่มีสถานะ “Lock” ต่อสู้กันอยู่ และมีมอนสเตอร์ตัวใหม่เข้ามาต่อสู้ด้วย จะเรียกสถานะมอนสเตอร์ตัวใหม่ว่า “Flank” ถ้ามอนสเตอร์ตัวใดมีสถานะนี้ จะได้ค่าความเสียหายจากการโจมตีเพิ่มมากขึ้น

มอนสเตอร์ทุกตัวจะมีธาตุของตัวเองเสมอ ได้แก่ ธาตุไฟ ธาตุลม ธาตุน้ำ ธาตุดิน ถ้าเกิดมีมอนสเตอร์ที่มีธาตุที่แข็งแกร่งกว่ามาโจมตี จะทำให้ค่าความเสียหายจากการโจมตีเพิ่มมากขึ้น โดยเรียงความแข็งแกร่งของธาตุไว้ดังนี้ ธาตุไฟชนะธาตุลม ธาตุลมชนะธาตุดิน ธาตุดินชนะธาตุน้ำ และธาตุน้ำชนะธาตุไฟ

ค่าความเสียหายจากการโจมตีทั่วไป จะคำนวณได้จาก

$$Damage = (AT * S.AT) - (DF * S.DF)$$

มอนสเตอร์ทุกตัวจะมีโอกาสโจมตีแบบ Critical ได้ โดยค่าความเสียหายจะไม่หักลบกับค่าพลังป้องกันของมอนสเตอร์ศัตรู โดยคำนวณได้จาก

$$Damage = AT * S.AT$$

ถ้ามอนสเตอร์มีธาตุที่แข็งแกร่งกว่า หรือมีสถานะ “Flank” ค่าความเสียหายจะคำนวณจาก

$$Damage = Damage * 1.2$$

มอนสเตอร์ทุกตัวจะมีอัตราการหลบหลีก โดยคิดจากค่า SD ของมอนสเตอร์ตัวนั้น ๆ ซึ่งถ้ามอนสเตอร์ตัวใดหลบหลีกการโจมตีได้ ค่าความเสียหายจะมีค่าเท่ากับศูนย์

3.3 ขั้นตอนการออกแบบเงื่อนไขของระบบปัญญาประดิษฐ์

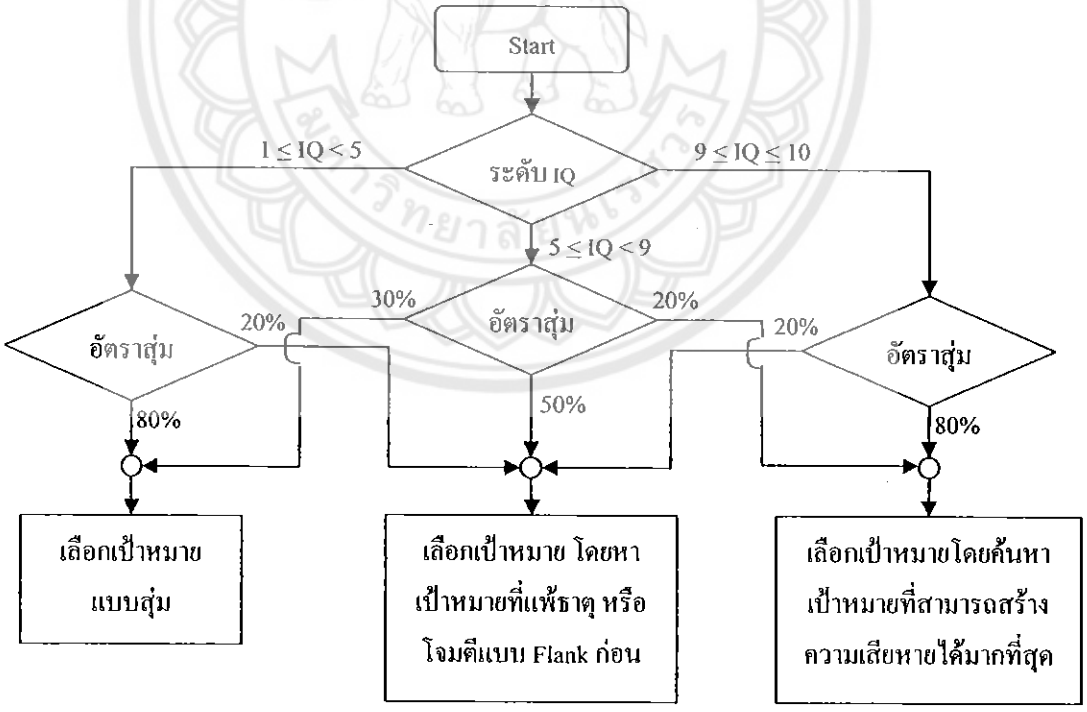
ระบบปัญญาประดิษฐ์ที่ใช้ได้แบ่งออกเป็น 2 ส่วนดังนี้

3.3.1 ระบบปัญญาประดิษฐ์ของมอนสเตอร์ฝ่ายศัตรู

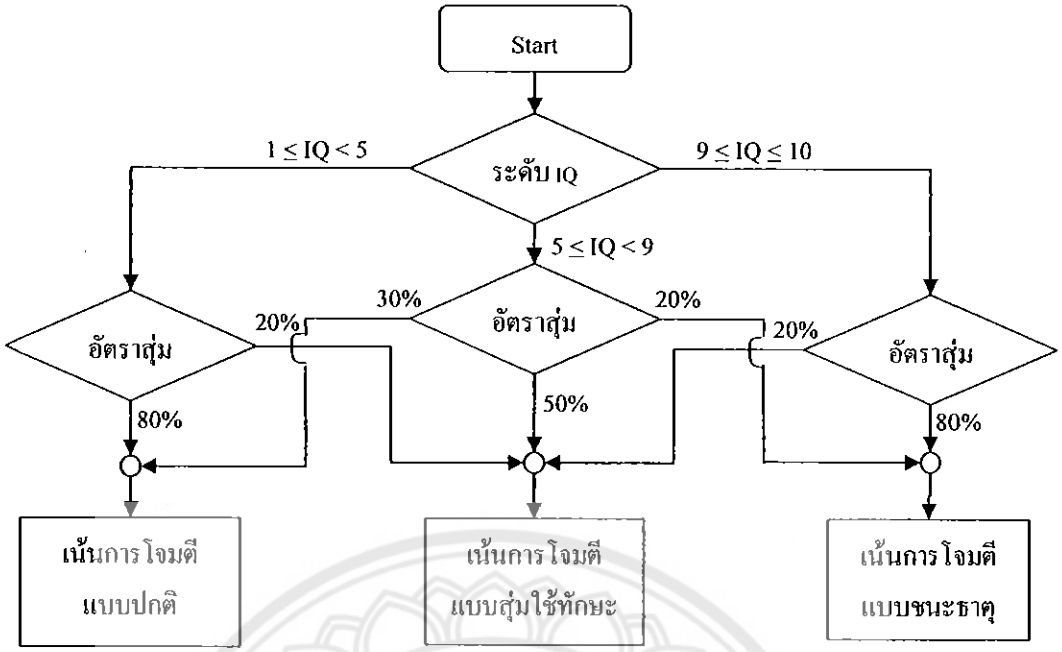
เลือกใช้ระบบปัญญาประดิษฐ์แบบ Searching โดยตั้งค่าความสามารถของมอนสเตอร์ศัตรูไว้ 10 ระดับ และใช้ค่า IQ เป็นตัววัดระดับของมอนสเตอร์ กำหนดค่า IQ และระดับของมอนสเตอร์ไว้ดังนี้

- มอนสเตอร์ระดับต้น จะมีระดับ IQ ตั้งแต่ 1 – 4
- มอนสเตอร์ระดับกลาง จะมีระดับ IQ ตั้งแต่ 5 – 8
- มอนสเตอร์ระดับสูง จะมีระดับ IQ ตั้งแต่ 9 – 10

สามารถแสดงเป็น Flowchart อย่างคร่าว ๆ ได้ดังนี้



รูปที่ 3.8 Flowchart แสดงการเลือกเป้าหมายของมอนสเตอร์ฝ่ายศัตรู



รูปที่ 3.9 Flowchart แสดงการเลือกทักษะของมอนสเตอร์ฝ่ายศัตรู

3.3.2 ระบบปัญญาประดิษฐ์ของมอนสเตอร์ฝ่ายผู้เล่น

เลือกใช้ระบบปัญญาประดิษฐ์แบบ Reinforcement โดยตั้งค่าความเสียหายเป็นตัวแปรในการเปลี่ยนแปลงการพัฒนาของระบบปัญญาประดิษฐ์ ถ้าการเลือกเป้าหมาย และการเลือกทักษะแบบใดสร้างความเสียหายได้มาก กราฟการพัฒนาของระบบปัญญาประดิษฐ์จะเพิ่มขึ้น ถ้าการเลือกเป้าหมาย และการเลือกทักษะแบบใดสร้างความเสียหายได้น้อย กราฟการพัฒนาของระบบปัญญาประดิษฐ์จะลดลง

การใช้ระบบปัญญาประดิษฐ์ของมอนสเตอร์ฝ่ายผู้เล่นแบ่งออกเป็น 2 ขั้นตอน

1. ขั้นตอนการเก็บรวบรวมข้อมูล

จะเก็บรวบรวมข้อมูลในระหว่างการต่อสู้ โดยเก็บค่าต่าง ๆ คือ ค่าความเสียหายที่ทำได้ ทักษะที่ใช้ในการโจมตี สถานะและธาตุของเป้าหมายที่เลือกในการโจมตี เพื่อนำไปใช้เปลี่ยนแปลงของกราฟการพัฒนาต่อไป

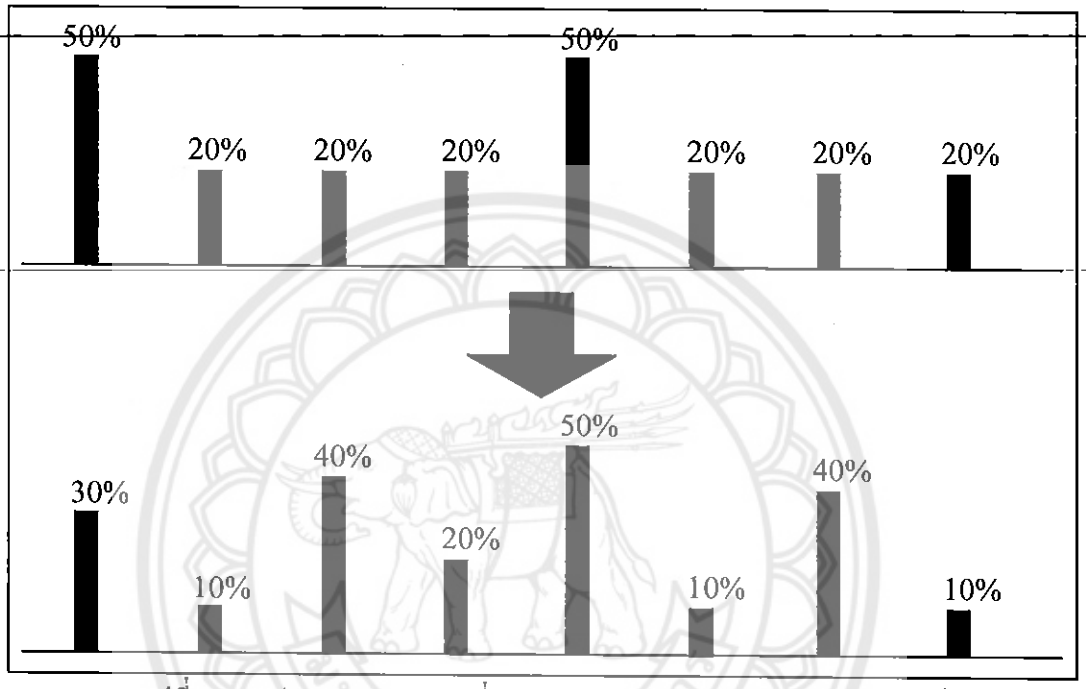
2. ขั้นตอนการเปลี่ยนแปลงกราฟการพัฒนา

กราฟจะไม่มีการเปลี่ยนแปลง ถ้ามอนสเตอร์ฝ่ายผู้เล่นไม่มีการโจมตี (ถูกโจมตีฝ่ายเดียว) หรือโจมตีโดยใช้ทักษะเดิมและเลือกเป้าหมายเดิมตลอดการต่อสู้

ค่าความเปลี่ยนแปลงจะวัดจากค่าน้ำหนักของความเสียหาย ดังนี้

- ค่าความเสียหายที่ระหว่าง 0 ถึง 200 กราฟจะเพิ่มขึ้น 5%
- ค่าความเสียหายที่ระหว่าง 200 ถึง 1000 กราฟจะเพิ่มขึ้น 10%
- ค่าความเสียหายที่ระหว่าง 1000 ถึง 5000 กราฟจะเพิ่มขึ้น 15%

- ค่าความเสียหายที่มากกว่า 5000 ขึ้นไป กราฟจะเพิ่มขึ้น 20%
- ค่าความเสียหายที่ระหว่าง -200 ถึง 0 กราฟจะลดลง 5%
- ค่าความเสียหายที่ระหว่าง -1000 ถึง -200 กราฟจะลดลง 10%
- ค่าความเสียหายที่ระหว่าง -5000 ถึง -1000 กราฟจะลดลง 15%
- ค่าความเสียหายที่น้อยกว่า -5000 ลงไป กราฟจะลดลง 20%



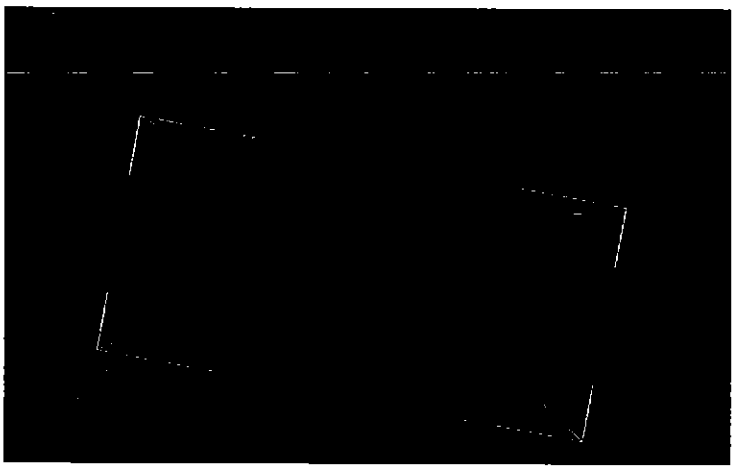
รูปที่ 3.10 รูปภาพแสดงการเปลี่ยนแปลงของกราฟระบบปัญญาประดิษฐ์

3.4 ขั้นตอนการพัฒนาเกม

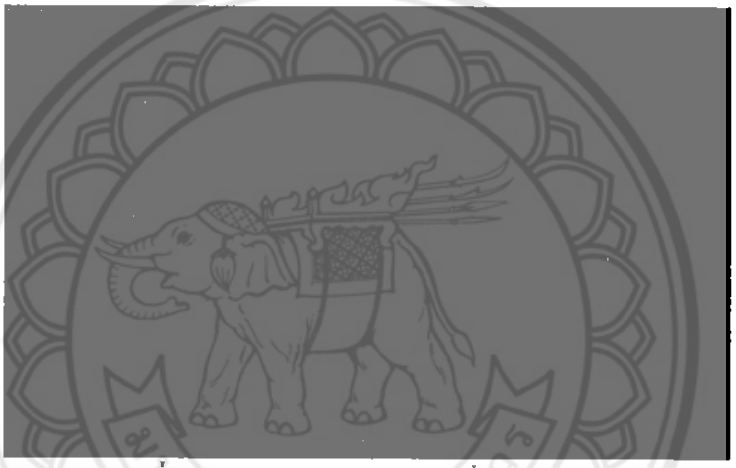
3.4.1 ขั้นตอนการสร้างโมเดล 3 มิติ

3.4.1.1 ขั้นตอนการสร้างโมเดลด้วย Autodesk 3ds Max 2009

การสร้างโมเดลมอนสเตอร์ ฉาก และตัวละครจะเริ่มขึ้นรูปโดยใช้รูปทรงเรขาคณิตต่าง ๆ เป็นหลัก เช่น Box Cone Sphere และ Cylinder เป็นต้น จากนั้นทำการแปลงเป็นรูปแบบ Poly และจัดการตกแต่งให้เป็นรูปทรงที่ต้องการด้วยการดัด Vertex และ Border พอได้รูปทรงที่ต้องการแล้วจึงนำมาประกอบเป็นรูปร่าง และจัดโมเดลเป็นกลุ่มเพื่อทำเป็น animation เคลื่อนไหวต่อไป ตามรูปที่ 3.11 ถึงรูปที่ 3.14

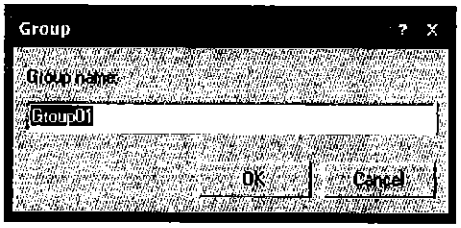


รูปที่ 3.11 รูปภาพแสดงการตกแต่งให้เป็นรูปทรงด้วยการดัด Vertex และ Border



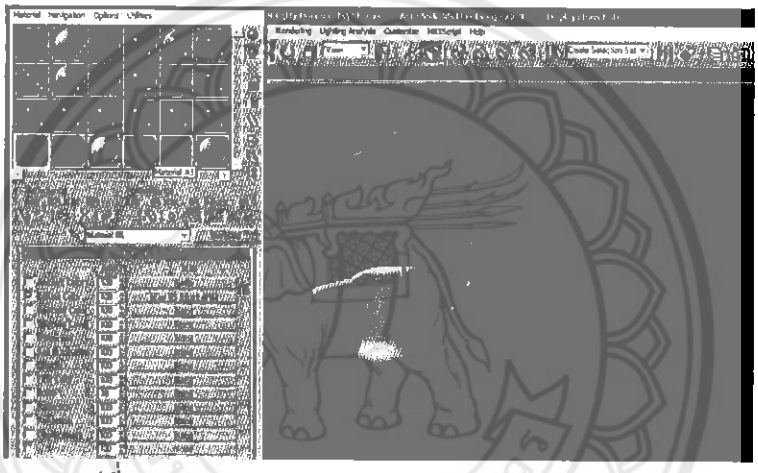
รูปที่ 3.12 รูปภาพแสดงรูปทรงที่ดัดเสร็จแล้ว

รูปที่ 3.13 รูปภาพแสดงการประกอบชิ้นส่วนต่าง ๆ เป็นรูปทรงที่ต้องการ



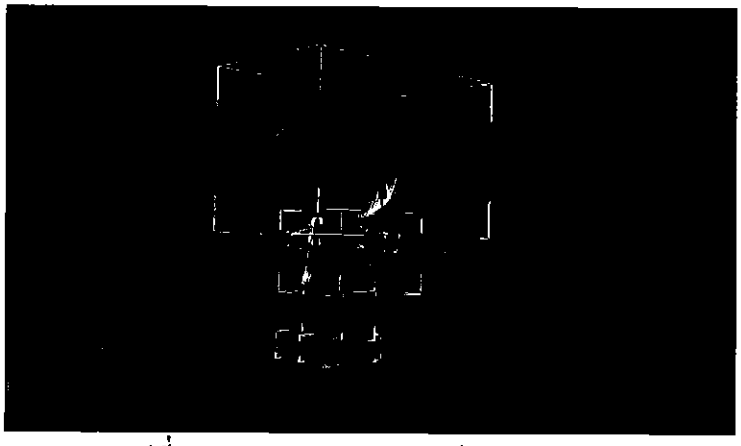
รูปที่ 3.14 รูปภาพแสดงการจัดกลุ่มโมเดลเพื่อง่ายต่อการทำ Animation

หลังจากได้รูปทรงที่ต้องการแล้ว เรียกหน้าต่าง Material edition ออกมาแล้วใส่พื้นผิว (Texture) ให้แก่โมเดล หลังการนั้นให้ทำการ Mapping ลวดลายให้แสดงตรงตามที่ต้องการด้วย Unwrap UVW ตามรูปที่ 3.15



รูปที่ 3.15 รูปภาพแสดงการใส่สีและลวดลายบนโมเดล

เริ่มทำการสร้าง Animation ด้วยการกำหนด key frame แล้วขยับชิ้นส่วนต่างๆให้เป็นท่าทางที่ต้องการ โดยแบ่งท่าทางต่าง ๆ ตามจำนวนเฟรมที่กำหนดเพื่อให้ง่ายต่อการนำไปเรียกใช้ในโปรแกรม หลังจากนั้นก็ Export ไฟล์ออกมาเป็น file สกุล .X



รูปที่ 3.16 รูปภาพแสดงการสร้าง Animation

3.4.1.2 ขั้นตอนการ Export files จาก Autodesk 3ds Max 2009

ในส่วนนี้เป็นการ Export files จาก 3ds Max 2009 เพื่อนำในใช้ในเกม โดย Irrlicht Game Engine นั้นสามารถรองรับ File Animation ได้หลายสกุล แต่ทางผู้จัดทำได้เลือกใช้ File Animation สกุล .X มาใช้ในตัวเกม (สามารถดูรายละเอียดได้ที่ภาคผนวก ในส่วนของการ Export files .X)

3.4.2 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนเป็นส่วนของการนำเอาองค์ประกอบต่าง ๆ มารวมให้เป็นโครงร่างขึ้นมา และต้องมีการจัดการกับอุปกรณ์ต่าง ๆ ของเครื่อง โมเดลสามมิติ ภาพสองมิติ รวมทั้งการรับค่า Input จากผู้เล่น และต้องจัดการส่วนที่อาจจะเกิดความผิดพลาดขึ้นระหว่างการทำงาน ซึ่งจะเป็นการเขียนโปรแกรมบนพื้นฐานโครงสร้างของ Irrlicht Game Engine โดยสามารถแบ่งขั้นตอนการเขียนโปรแกรมได้ดังนี้

1. การเขียนโปรแกรมส่วนการสร้างหน้าต่างเกม
2. การเขียนโปรแกรมส่วนการโหลดโมเดล 3 มิติ
3. การเขียนโปรแกรมส่วนการโหลดภาพ 2 มิติ
4. การเขียนโปรแกรมส่วนการรับค่า Input จากผู้เล่น
5. การเขียนโปรแกรมส่วนของรูปแบบภายในเกม

3.4.2.1 การเขียนโปรแกรมส่วนการสร้างหน้าต่างเกม

ในส่วนนี้จะเป็นการสร้างหน้าต่างเกม รวมถึงการจัดการเกี่ยวกับอุปกรณ์แสดงผลและการแสดงผลในรูปแบบต่าง ๆ โดยการเขียนโปรแกรมจะเป็นการกำหนดค่าต่าง ๆ ในคลาส IrrlichtDevice ซึ่งเป็นคลาสใน Irrlicht Game Engine ที่ทำงานเกี่ยวกับหน้าต่างเกมทั้งหมด ซึ่งประกอบไปด้วยการกำหนดชุดคำสั่ง (Library) จัดการด้านกราฟฟิกโดยได้เลือกใช้ชุดคำสั่งของ OpenGL การกำหนดขนาดของหน้าต่าง และกำหนดการแสดงผลแบบไม่เต็มจอ (สามารถดูรายละเอียดได้ที่ภาคผนวก ในส่วนของคลาส IrrlichtDevice)

3.4.2.2 การเขียนโปรแกรมส่วนการโหลดโมเดล 3 มิติ

ในส่วนนี้จะเป็นการโหลดโมเดล 3 มิติที่จะนำเข้ามาใช้ในเกม ทำให้เกมดูสวยงาม น่าเล่นยิ่งขึ้น โดยในส่วนของการเขียนโปรแกรมจะเป็นการกำหนดค่าต่าง ๆ ในส่วนของ Namespace irr::scene ซึ่งเป็นส่วนที่ทำงานเกี่ยวกับการ Render ภาพทั้งหมด โดยจะมีการกำหนดค่าต่าง ๆ ให้กับตัวแปรที่จัดการเกี่ยวกับโมเดล 3 มิติ เช่นตำแหน่งที่โมเดล (Position) ขนาด (Scale) การหมุนภาพ (Rotation) รวมทั้งการกำหนดค่าของการเคลื่อนไหวในส่วนของโมเดลที่มีการเคลื่อนไหว เช่น ลูปของการแสดงภาพ (FrameLoop) ความไวของการแสดงภาพ (Animation speed) นอกจากนี้ยังมี

ส่วนของการกำหนดจุดกำเนิดแสง การให้เงาของโมเดลต่าง ๆ (สามารถดูรายละเอียดได้ที่ภาคผนวก ในส่วนของคลาส scene::IAnimatedMeshSceneNode)

3.4.2.3 การเขียนโปรแกรมส่วนการโหลดภาพ 2 มิติ

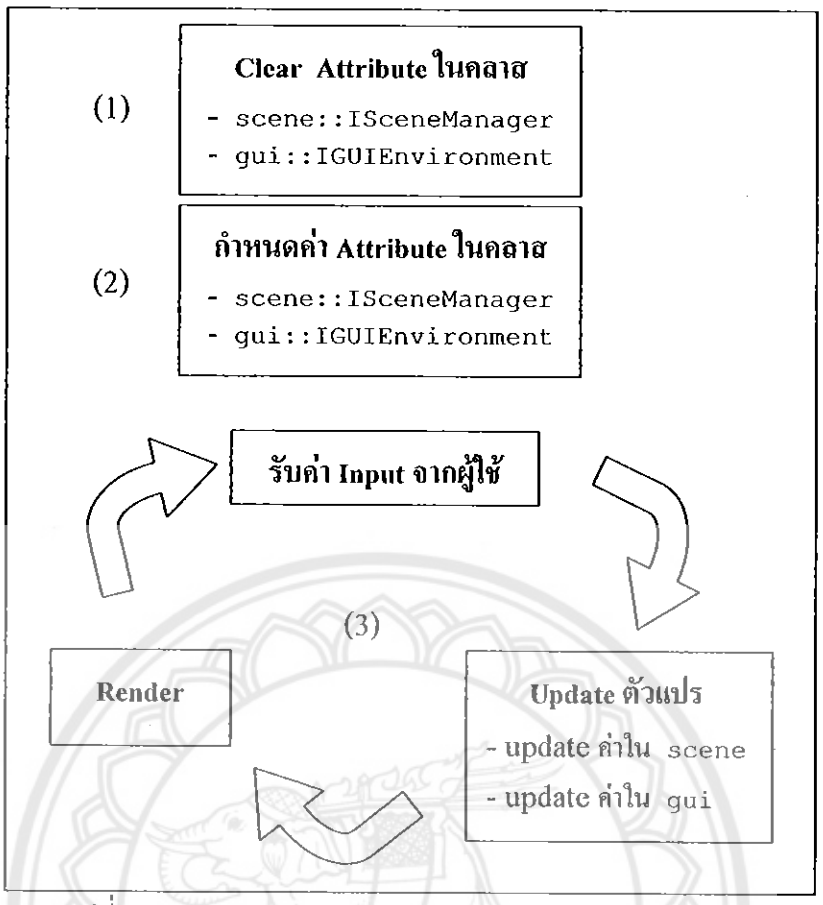
ในส่วนนี้จะเป็นการแสดงภาพ 2 มิติทั้งภาพเมนูต่าง ๆ รวมถึงข้อความหรือตัวอักษรต่างๆ ซึ่งจะช่วยเพิ่มสีสันให้กับตัวเกมมากขึ้น และยังเป็นการช่วยให้ผู้เล่นสามารถเข้าใจตัวเกมได้มากยิ่งขึ้นด้วย โดยการเขียนโปรแกรมจะเป็นการกำหนดค่าต่าง ๆ ในส่วนของ namespace irr::gui ซึ่งเป็นส่วนที่ทำงานเกี่ยวกับ GUI (Graphic User Interface) ซึ่งจะต้องกำหนดค่าพิกัดในการแสดงภาพ รวมทั้งขนาดของพื้นที่ที่จะแสดงภาพ (สามารถดูรายละเอียดได้ที่ภาคผนวก ในส่วนของคลาส gui::IGUIEnvironment)

3.4.2.4 การเขียนโปรแกรมส่วนการรับค่า Input จากผู้เล่น

ในส่วนนี้จะเป็นการรับค่า Input จากผู้เล่นเพื่อที่ให้ผู้เล่นควบคุมตัวเกม ซึ่งประกอบไปด้วยการรับค่าจาก Keyboard เพื่อควบคุมตัวละครในฉากหลัก และการรับค่า Mouse เพื่อเข้าสู่เมนูหรือคำสั่งต่าง ๆ โดยการเขียนโปรแกรมจะเป็นการกำหนดค่าต่าง ๆ ในคลาส IEventReceiver ซึ่งเป็นคลาสของ Irrlicht Game Engine ที่ทำงานเกี่ยวกับการรับค่า Input ต่าง ๆ (สามารถดูรายละเอียดได้ที่ภาคผนวก ในส่วนของคลาส IEventReceiver)

3.4.2.5 การเขียนโปรแกรมส่วนของรูปแบบภายในเกม

ในส่วนนี้จะเป็นการเขียนเงื่อนไขของการแสดงผลในแต่ละฉากภายในเกม เพื่อให้การทำงานเป็นไปอย่างสมบูรณ์และใช้ทรัพยากรให้คุ้มค่าที่สุด โดยการแบ่งช่วงการทำงานเป็นส่วนแล้วเมื่อมีการเปลี่ยนฉากก็จะทำการ เคลียร์ค่าในคลาส scene::ISceneManager และคลาส gui::IGUIEnvironment แล้วจึง Set ค่าใหม่ ซึ่งจะทำให้ใช้ทรัพยากรได้น้อยลง โดยในแต่ละฉากจะประกอบไปด้วยการทำงาน 3 ส่วนคือ การโหลดและกำหนดค่าของโมเดล 3 มิติและภาพ 2 มิติ การแสดงภาพ 2 มิติ และการรับค่า Input จากผู้เล่นซึ่งสามารถแสดงการทำงานได้ดังนี้



รูปที่ 3.17 รูปภาพแสดงการทำงานของโปรแกรมในแต่ละฉาก

นอกจากนี้ผู้จัดทำได้แบ่งฉากต่าง ๆ ออกเป็นส่วน ๆ ตามลักษณะการแสดงผลได้ดังนี้

1. ฉากเมนูเริ่มเกม
2. ฉากเลือกมอนสเตอร์เริ่มต้น
3. ฉากหลัก
4. ฉากต่อสู้
5. ฉากแสดงไอเทม
6. ฉากแสดงสถานะของมอนสเตอร์

โดยในแต่ละฉากจะมีการทำงานของโปรแกรมดังที่กล่าวมา ซึ่งแต่ละฉากจะมีคลาส Receiver ที่มีการทำงานเกี่ยวกับการรับค่า Input ที่แตกต่างกันออกไปตามลักษณะการทำงานของฉากนั้นๆ เช่น ฉากหลักมีการรับ Input ทั้งทาง Keyboard และ Mouse ฉากแสดงไอเทมและฉากแสดงสถานะของมอนสเตอร์จะมีการรับ Input ทาง Mouse เท่านั้น เป็นต้น นอกจากนั้นยังต้องกำหนดให้แต่ละฉากสามารถเชื่อมโยงไปมาได้ตามรูปแบบของตัวเกม

บทที่ 4

ผลการทดลอง

หลังจากการศึกษาการสร้างเกม ออกแบบส่วนประกอบต่าง ๆ และได้ดำเนินการการพัฒนาเกมเป็นอันเสร็จสิ้นแล้ว ในบทนี้จะเป็นการนำโครงงานที่ทำมาทดสอบ เพื่อตรวจสอบว่าโครงงานได้เป็นไปตามที่ออกแบบไว้หรือไม่ และค้นหาข้อผิดพลาดต่าง ๆ มาใช้วิเคราะห์และหาทางแก้ไขให้โปรแกรมมีความผิดพลาดน้อยที่สุด

4.1 ผลการทดสอบตัวเกม

4.1.1 ทดสอบการโหลดโมเดล 3 มิติจาก 3ds Max เข้าสู่เกม

การทดสอบครั้งที่ 1

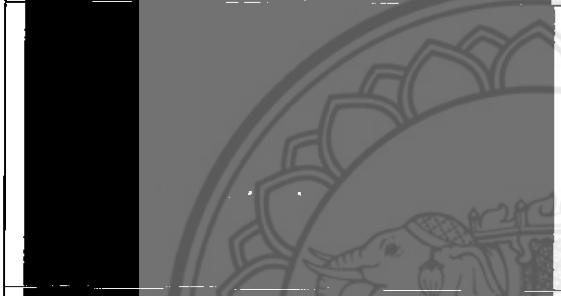
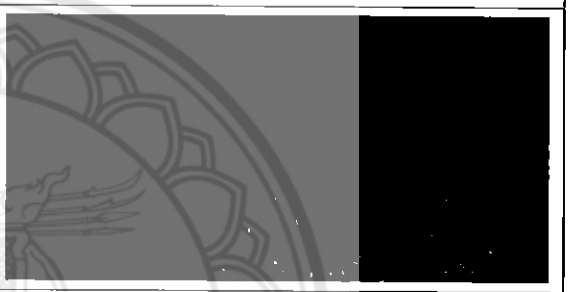
วัตถุประสงค์ เพื่อทดสอบการแสดงผลของโมเดล 3 มิติที่สร้างขึ้นเอง

รายละเอียด ทำการสร้างโมเดล 3 มิติ ด้วยโปรแกรม Autodesk 3ds Max 2009 แล้วทำการโหลดโมเดล 3 มิติที่สร้างขึ้นลงไปภายในเกม

ผลที่คาดหวัง โมเดลที่สร้างขึ้นด้วย 3ds Max จะต้องแสดงผลได้ดีภายในเกม

ผลที่ได้ โมเดลที่ได้มีความแตกต่างกันอยู่ค่อนข้างมาก ทั้งเรื่องแสงเงา ส่วน Animation และสีที่แสดงผลออกมา

ตารางที่ 4.1 ตารางเปรียบเทียบภาพที่สร้างจากโปรแกรม **3ds Max** และภาพภายในเกม

ภาพจากโปรแกรม 3ds Max	ภาพภายในเกม
	
	
	
	

4.1.2 ทดสอบฉากภายในเกม

4.1.2.1 ทดสอบฉากเมนูเข้าเกม

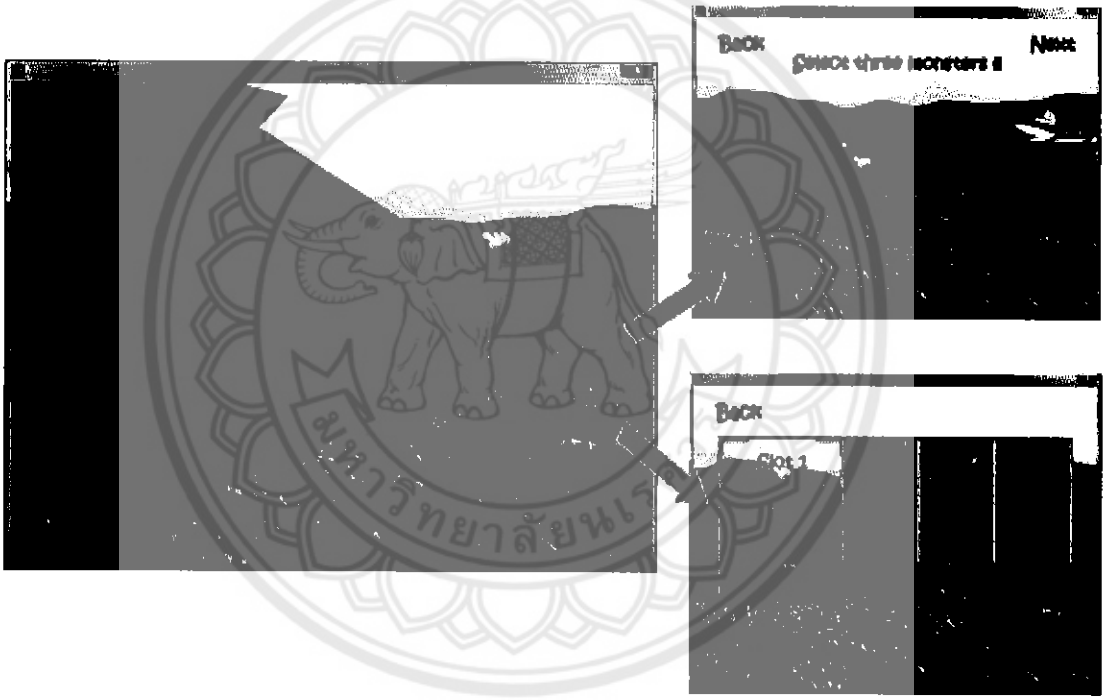
การทดสอบครั้งที่ 2

วัตถุประสงค์ เพื่อทดสอบการแสดงผลและการรับ Input ภายในฉาก

รายละเอียด เมื่อเริ่มเกมตัวเกมจะเข้าสู่หน้าเมนูเข้าเกม ซึ่งจะสามารถรับ Input ผ่านทาง Mouse เพื่อเชื่อมต่อไปยังฉากเลือกมอนสเตอร์ผ่านทาง New Game หรือเชื่อมต่อไปยังฉากโหลดเกมผ่านทาง Load Game และออกจากเกมโดยคลิกที่ Exit

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.1 รูปภาพการแสดงฉากเมนูเข้าเกม

4.1.2.2 ทดสอบฉากเลือกมอนสเตอร์เริ่มต้น

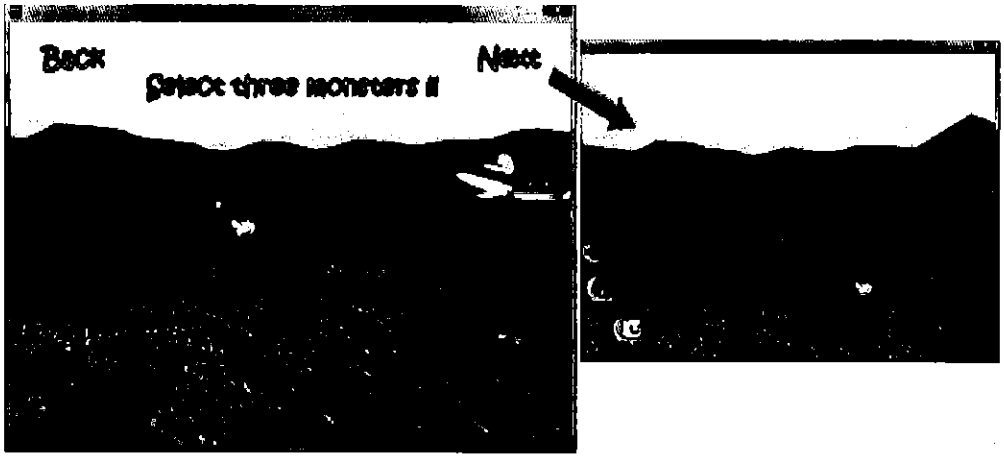
การทดสอบครั้งที่ 3

วัตถุประสงค์ เพื่อทดสอบการแสดงผลและการรับ Input ภายในฉาก

รายละเอียด เมื่อเข้าสู่ฉากเลือกมอนสเตอร์เริ่มต้น ผู้เล่นจะต้องเลือกมอนสเตอร์ 3 ตัวที่ต้องการแล้ว กดที่ Next เพื่อเข้าสู่ฉากหลัก

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.2 รูปภาพการแสดงฉากเลือกมอนสเตอร์เริ่มต้น

4.1.2.2 ทดสอบฉากหลัก

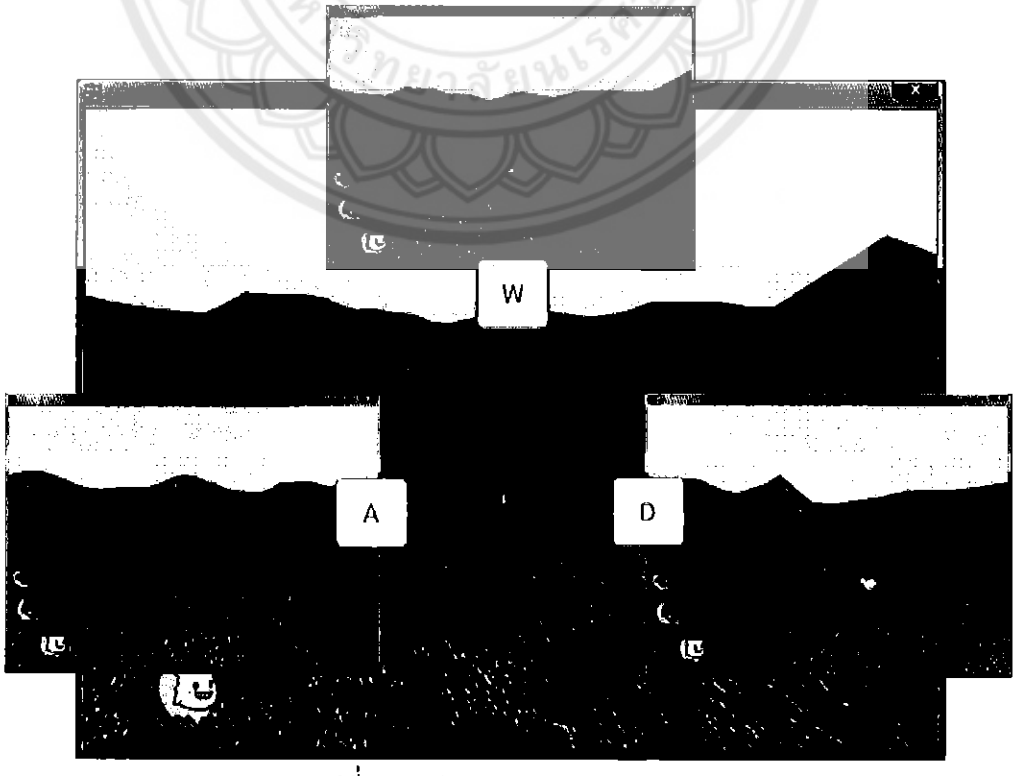
การทดสอบครั้งที่ 4

วัตถุประสงค์ เพื่อทดสอบการแสดงผลและการรับ Input ภายในฉาก

รายละเอียด เมื่อเข้าสู่ฉากหลัก ผู้เล่นจะสามารถเคลื่อนไหวตัวละครได้โดยกดปุ่ม W A และ D บน Keyboard

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.3 รูปภาพการแสดงฉากหลัก

4.1.2.3 ทดสอบฉากแสดงสถานะของมอนสเตอร์

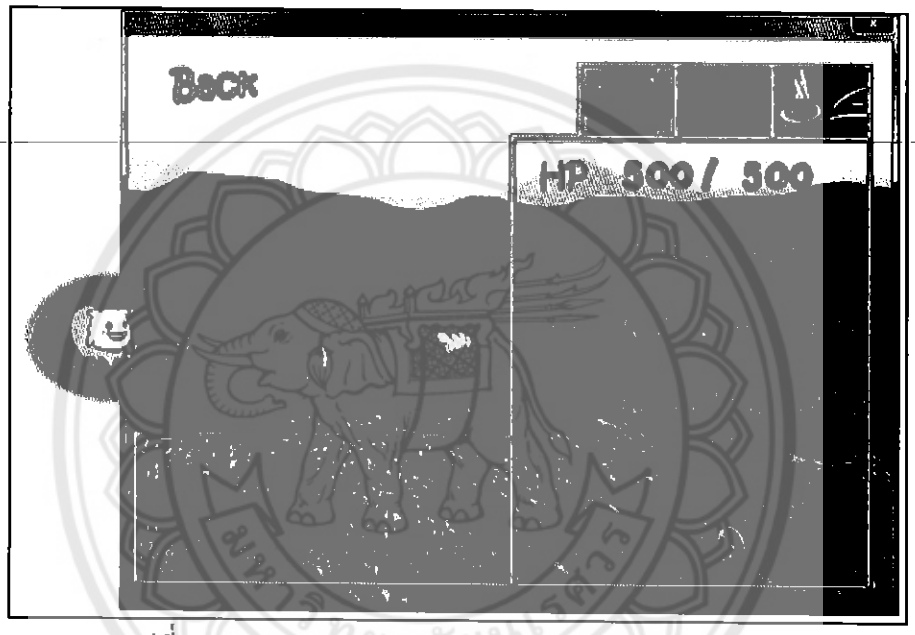
การทดสอบครั้งที่ 5

วัตถุประสงค์ เพื่อทดสอบการแสดงผลและการรับ Input ภายในฉาก

รายละเอียด เมื่อผู้เล่นคลิกที่รูปมอนสเตอร์ด้านซ้ายล่าง ผู้เล่นจะเข้าสู่ฉากแสดงสถานะของมอนสเตอร์

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.4 รูปภาพการแสดงฉากแสดงสถานะของมอนสเตอร์

4.1.2.4 ทดสอบฉากแสดงไอเทม

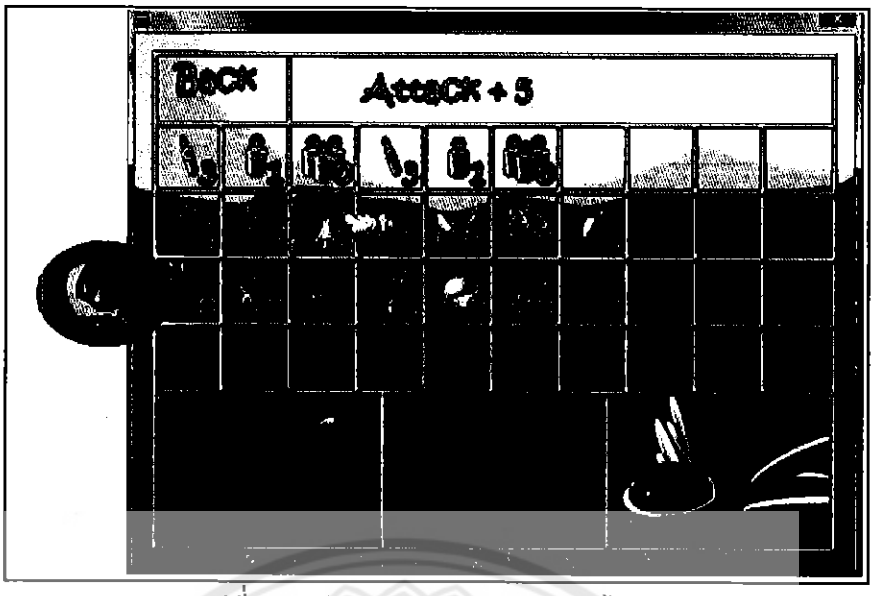
การทดสอบครั้งที่ 6

วัตถุประสงค์ เพื่อทดสอบการแสดงผลและการรับ Input ภายในฉาก

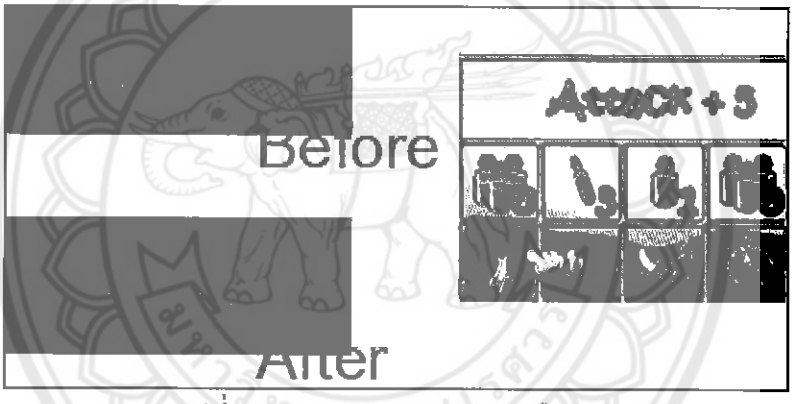
รายละเอียด เมื่อผู้เล่นคลิกที่รูปกระเป๋าด้านซ้ายล่าง ผู้เล่นจะเข้าสู่ฉากแสดงไอเทม เมื่อคลิกที่รูปไอเทม แล้วไปคลิกที่มอนสเตอร์จะเป็นการใช้ไอเทมเพื่อเพิ่มสถานะของมอนสเตอร์อย่างใดอย่างหนึ่ง

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.5 รูปภาพการแสดงฉากแสดงไอเทม



รูปที่ 4.6 รูปภาพการแสดงการใช้ไอเทม

4.1.2.5 ทดสอบฉากต่อสู้

การทดสอบครั้งที่ 7

วัตถุประสงค์ เพื่อทดสอบการแสดงผลฉากต่อสู้

รายละเอียด เมื่อผู้เล่นเดินไปชนกับมอนสเตอร์ภายในฉากหลักแล้ว จะเข้าสู่ฉากต่อสู้

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง

รูปที่ 4.7 รูปภาพการแสดงการเข้าสู่ฉากต่อสู้และการแสดงฉากต่อสู้

4.1.2.6 ทดสอบการโหลดเกมและเซฟเกม

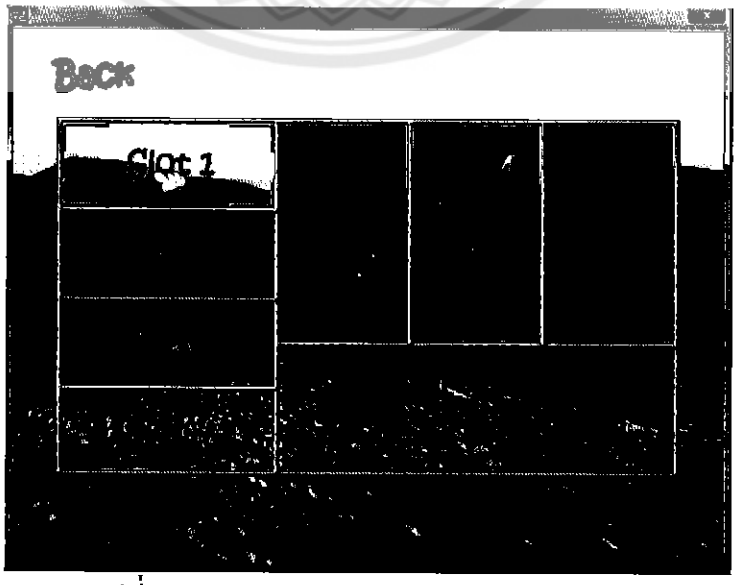
การทดสอบครั้งที่ 8

วัตถุประสงค์ เพื่อทดสอบการเชื่อมต่อข้อมูลกับฐานข้อมูล

รายละเอียด เมื่อคลิกที่ Load Game ในฉากเมนูเริ่มต้น จะเข้าสู่ฉากโหลดเกม หรือคลิกที่รูปประตูด้านซ้ายล่างในฉากหลัก เพื่อเข้าสู่ฉากเซฟเกม

ผลที่คาดหวัง สามารถแสดงผลได้ตรงตามรายละเอียด

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.8 รูปภาพการแสดงฉากโหลดและเซฟเกม

4.2 ผลการทดสอบระบบปัญญาประดิษฐ์

4.2.1 ทดสอบการแสดงผลของระบบปัญญาประดิษฐ์ภายในเกม

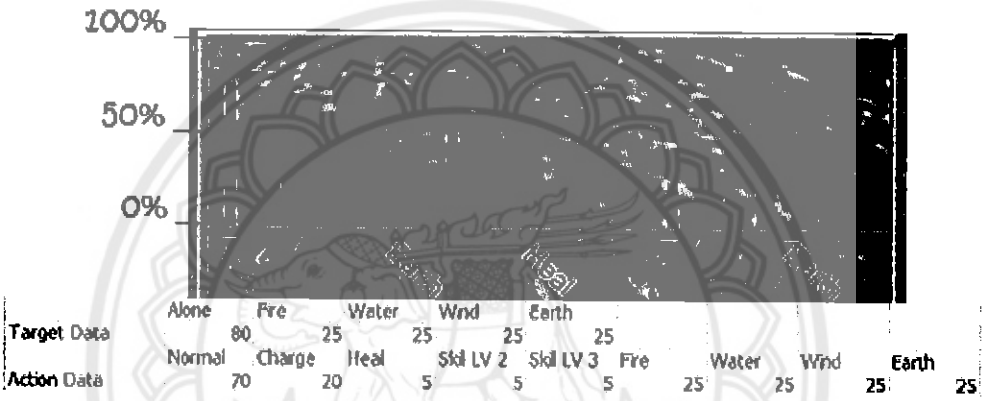
การทดสอบครั้งที่ 9

วัตถุประสงค์ เพื่อทดสอบการแสดงผลของระบบปัญญาประดิษฐ์

รายละเอียด เมื่อทำการใส่ค่าของระบบปัญญาประดิษฐ์ลงในฐานข้อมูลแล้ว ทำการแสดงผลออกมาโดยการเริ่มเกมใหม่

ผลที่คาดหวัง ค่าที่แสดงผลออกมาตรงกันข้อมูลในฐานข้อมูล

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.9 รูปภาพการแสดงค่าในการเชื่อมต่อฐานข้อมูลกับระบบปัญญาประดิษฐ์ภายในเกม

4.2.2 ทดสอบการเปลี่ยนแปลงของระบบปัญญาประดิษฐ์

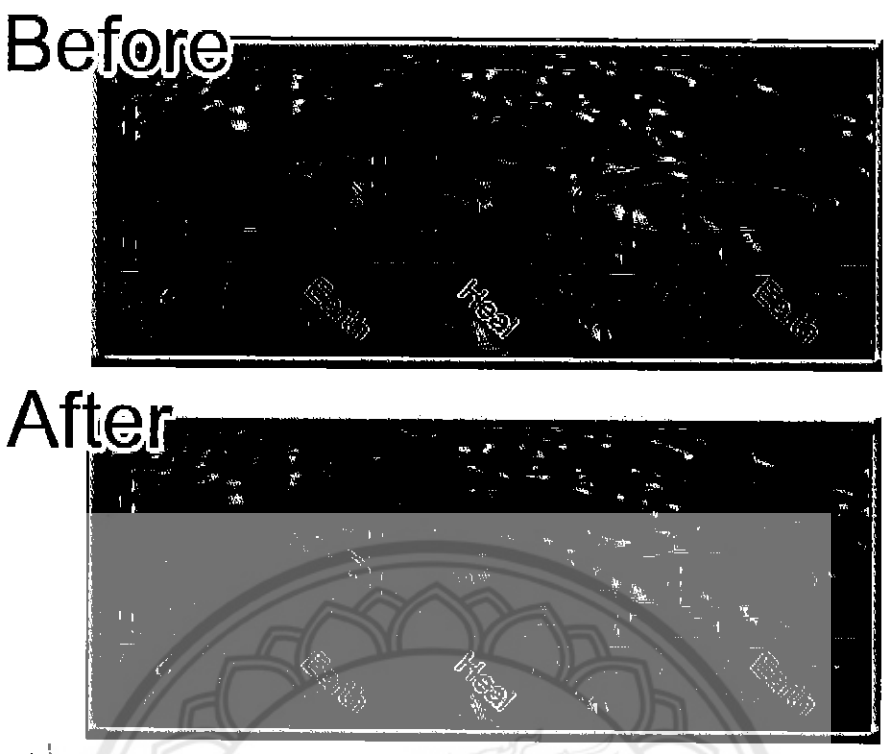
การทดสอบครั้งที่ 10

วัตถุประสงค์ เพื่อทดสอบการเปลี่ยนแปลงของกราฟระบบปัญญาประดิษฐ์

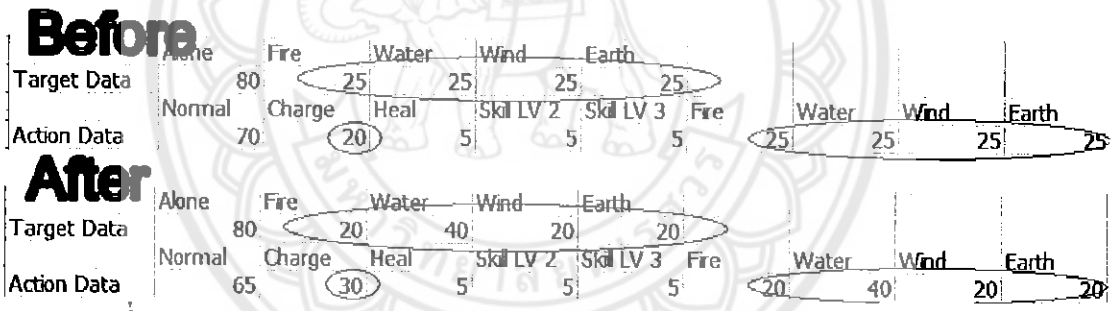
รายละเอียด เมื่อเริ่มการต่อสู้ ระบบปัญญาประดิษฐ์จะทำการเก็บข้อมูล และเมื่อการต่อสู้เสร็จสิ้น ระบบปัญญาประดิษฐ์จะทำการคิดคำนวณและเปลี่ยนแปลงกราฟการพัฒนาใหม่

ผลที่คาดหวัง สามารถแสดงค่าที่มีการเปลี่ยนแปลงได้อย่างชัดเจน

ผลที่ได้ ได้ผลตรงตามที่คาดหวัง



รูปที่ 4.10 รูปภาพแสดงความแตกต่างของกราฟการพัฒนาของระบบปัญญาประดิษฐ์



รูปที่ 4.11 รูปภาพแสดงความแตกต่างของข้อมูลการพัฒนาของระบบปัญญาประดิษฐ์

4.2.3 ทดสอบความเที่ยงตรงของระบบปัญญาประดิษฐ์

การทดสอบครั้งที่ 11

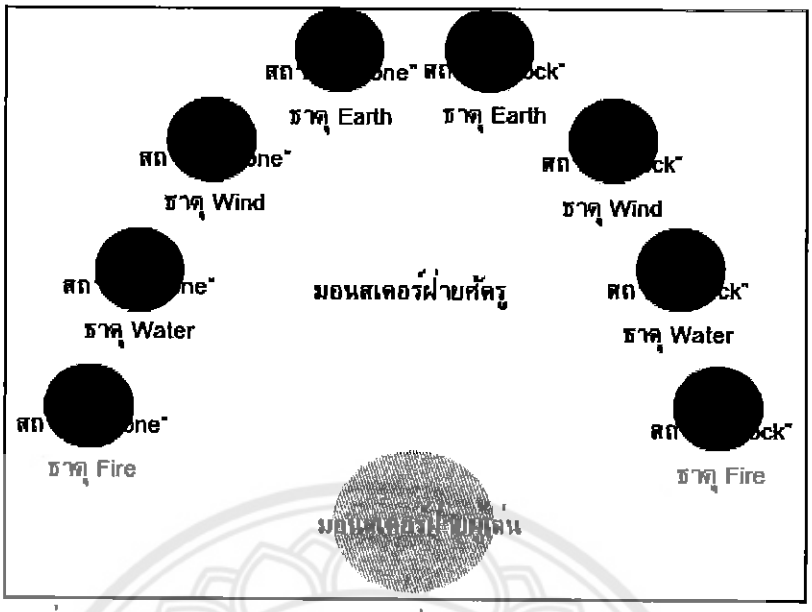
วัตถุประสงค์ เพื่อทดสอบความเที่ยงตรงของการเลือกเป้าหมาย

รายละเอียด เมื่อเริ่มต่อสู้ มอนสเตอร์ของผู้เล่นจะต้องใช้ระบบปัญญาประดิษฐ์เพื่อเลือกหาเป้าหมาย มอนสเตอร์ของผู้เล่นจะต้องมีเปอร์เซ็นต์โอกาสที่จะเลือกเป้าหมายได้ตามข้อมูลในฐานข้อมูล หรือกราฟการพัฒนานั้น ๆ

เงื่อนไข ทดสอบกับมอนสเตอร์ในสถานการณ์ที่มีการเลือกเป้าหมายได้ทุกกรณี โดยใช้กราฟการพัฒนาเริ่มต้น ตามรูปที่ 4.9

ผลที่คาดหวัง สามารถแสดงค่าได้ตามข้อมูลที่ใส่ไว้

ผลที่ได้ ค่าที่ได้คาดเคลื่อนเล็กน้อยเนื่องจากความเบี่ยงเบนจากการสุ่มค่า



รูปที่ 4.12 รูปภาพแสดงสถานการณ์ที่สามารถเลือกเป้าหมายได้ทุกกรณี

ตารางที่ 4.2 ตารางการทดสอบการเลือกเป้าหมายของระบบปัญญาประดิษฐ์

ทดสอบครั้งที่	"Alone" Fire	"Alone" Water	"Alone" Wind	"Alone" Earth	"Lock" Fire	"Lock" Water	"Lock" Wind	"Lock" Earth	รวม
1	17%	22%	19%	20%	5%	4%	4%	5%	96%
	173	224	198	207	57	47	41	53	1000
2	22%	22%	20%	18%	4%	4%	4%	3%	97%
	220	223	200	188	47	48	41	33	1000
3	19%	22%	21%	17%	5%	4%	5%	4%	97%
	199	221	215	176	52	44	50	43	1000
4	21%	20%	20%	19%	4%	4%	5%	4%	97%
	213	208	199	196	43	44	52	45	1000
5	21%	21%	19%	19%	4%	4%	4%	4%	98%
	217	217	191	198	49	44	44	40	1000

ตารางที่ 4.3 ตารางสรุปการทดสอบการเลือกเป้าหมายของระบบปัญญาประดิษฐ์

ทดสอบครั้งที่	สถานะ Alone	ธาตุ Fire	ธาตุ Water	ธาตุ Wind	ธาตุ Earth
1	78%	22%	26%	23%	25%
2	82%	26%	26%	24%	21%
3	79%	24%	26%	26%	21%
4	79%	25%	24%	25%	23%
5	80%	25%	25%	23%	23%
เฉลี่ย	**79.6%**	**24.4%**	**25.4%**	**24.4%**	**22.6%**
ค่าที่มีอยู่จริง	80%	25%	25%	25%	**25%**

การทดสอบครั้งที่ 12

วัตถุประสงค์ เพื่อทดสอบความเที่ยงตรงของการเลือกทักษะในการโจมตี

รายละเอียด เหมือนกับการทดสอบที่ 11 แต่เป็นการเลือกทักษะในการโจมตีแทนการเลือกเป้าหมาย

เงื่อนไข ทดสอบในสถานการณ์ที่มีการเลือกทักษะได้ทุกกรณี ยกเว้นทักษะ “Heal” โดยใช้กราฟการพัฒนาเริ่มต้น ตามรูปที่ 4.9

ผลที่คาดหวัง สามารถแสดงค่าได้ตามข้อมูลที่ใส่ไว้

ผลที่ได้ ค่าที่ได้คาดเคลื่อนจำนวนหนึ่งเนื่องจากความเบี่ยงเบนจากการสุ่มค่า

ตารางที่ 4.4 ตารางการทดสอบการเลือกทักษะของระบบปัญญาประดิษฐ์

ทดสอบครั้งที่	Normal	Charge	Fire Skill	Water Skill	Wind Skill	Earth Skill
1	72%	19%	2.5%	1.7%	2.2%	2.3%
	723	190	25	17	22	23
2	71%	19%	2.0%	3.5%	2.6%	1.3%
	711	195	20	35	26	13
3	71%	18%	2.5%	2.7%	2.3%	1.8%
	719	188	25	27	23	18
4	70%	20%	2.3%	3.0%	2.6%	1.8%
	700	203	23	30	26	18
5	71%	20%	2.6%	2.1%	1.9%	1.6%
	716	202	26	21	19	16

ตารางที่ 4.5 ตารางสรุปการทดสอบการเลือกทักษะของระบบปัญญาประดิษฐ์

ทดสอบครั้งที่	Normal	Charge	Fire Skill	Water Skill	Wind Skill	Earth Skill
1	72%	19%	25%	17%	22%	23%
2	71%	19%	20%	35%	26%	13%
3	71%	18%	25%	27%	23%	18%
4	70%	20%	23%	30%	26%	18%
5	71%	20%	26%	21%	19%	16%
เฉลี่ย	71%	19.2%	23.8%	26%	23.2%	17.6%
ค่าที่มีอยู่จริง	70%	20%	25%	25%	25%	25%

บทที่ 5
สรุปผลและข้อเสนอแนะ

โครงงานนี้เป็นการพัฒนาเกม ในรูปแบบเกม 3 มิติ โดยผู้เล่นจะต้องรับบทบาทเป็นผู้เลี้ยงมอนสเตอร์ และนำมอนสเตอร์ออกไปต่อสู้ ซึ่งมอนสเตอร์จะมีระบบปัญญาประดิษฐ์ในการตัดสินใจ โดยผู้เล่นจะต้องคอยสังเกตการพัฒนาในด้านต่าง ๆ ของมอนสเตอร์ เพื่อที่จะเลือกเพิ่มค่าสถานะต่าง ๆ ให้กับมอนสเตอร์ และมอนสเตอร์ก็จะเรียนรู้และสามารถพัฒนาตัวเองให้ฉลาดขึ้นไปเรื่อย ๆ

ทางผู้จัดทำได้ใช้ซอฟต์แวร์ Microsoft Visual Studio 2005 และ Autodesk 3ds Max 2009 โดยใช้ภาษา C++ ในการพัฒนาเกม นอกจากนี้ยังได้เลือก Irrlicht Game Engine มาเป็นเครื่องมือช่วยในการพัฒนา เนื่องจากมีตัวช่วยในการพัฒนาจำนวนมาก ทำให้สามารถพัฒนาได้สะดวกและรวดเร็วขึ้น

5.1 สรุปผลการทดลอง

1. จากการทดสอบเกมพบว่า ตัวเกมสามารถทำงานได้บนระบบปฏิบัติการ Windows Vista และ Windows XP ได้เป็นอย่างดี โดยต้องมีการติดตั้งชุดคำสั่งบางอย่างลงในตัวเกม

2. จากการทดสอบเกมพบว่าการพัฒนาของระบบปัญญาประดิษฐ์ของมอนสเตอร์ มีการพัฒนาที่สูงขึ้น ซึ่งจะมากหรือน้อย ขึ้นอยู่กับการเพิ่มค่าสถานะต่าง ๆ ให้มอนสเตอร์ ถ้าผู้เล่นเพิ่มค่าสถานะในกับมอนสเตอร์ได้ถูกต้องก็จะทำให้มอนสเตอร์สามารถเรียนรู้ได้เร็วขึ้น

5.2 ปัญหาและแนวทางแก้ไข

1. การ Export โมเดล 3 มิติ จากโปรแกรม Autodesk 3ds Max 2009 ให้เป็น file สกุล .X โดยใช้ Panda3D นั้นยังไม่สมบูรณ์เนื่องจากพิกัดของโมเดลบางส่วนผิดไปจากโมเดลที่สร้างขึ้น ซึ่งอาจแก้ไขโดยการหา เครื่องมือช่วยในการ Export ตัวอื่น หรือใช้โปรแกรมพัฒนาโมเดล 3 มิติอื่น ๆ มาใช้ในการพัฒนา

2. การแสดงผลในฉากต่อสู้ยังไม่สมบูรณ์ โดยบางกรณียังมีการแสดงผลที่ผิดพลาดอยู่ ซึ่งอาจแก้ไขโดยการวางรูปแบบโครงสร้างการเขียนโปรแกรมใหม่

5.3 ข้อเสนอแนะ

1. เพิ่มส่วนของเสียงประกอบภายในเกมให้มากขึ้น เพื่อเพิ่มความสมบูรณ์ให้กับเกม และให้ผู้เล่นได้รู้สึกเพลิดเพลินไปกับเกมมากขึ้น

2. เพิ่มส่วนของรูปแบบการเล่นต่าง ๆ เข้าไป เช่น ระบบเควส เนื้อเรื่องของเกม หรือระบบเกมย่อยต่าง ๆ เป็นต้น เพื่อให้ผู้เล่นได้มีรูปแบบการเล่นที่หลากหลาย

3. เพิ่มรูปแบบการเล่นให้สามารถเล่นได้หลายคน ผ่านระบบเครือข่าย

4. พัฒนาส่วนของกราฟฟิกให้สวยงามมากยิ่งขึ้นเพื่อให้ตัวเกมน่าสนใจมากขึ้น

5. สามารถปรับอัลกอลิทึมของระบบปัญญาประดิษฐ์ให้มีความซับซ้อนยิ่งขึ้น เช่น ปรับเงื่อนไขของการเพิ่มหรือลดของค่าน้ำหนักของกราฟการพัฒนา เพื่อให้ระบบปัญญาประดิษฐ์สามารถประมวลผลได้ละเอียดยิ่งขึ้น

6. เพิ่มรูปแบบการต่อสู้ การเลือกเป้าหมายและการเลือกทักษะให้มากขึ้น

7. เพิ่มระบบปัญญาประดิษฐ์อื่น ๆ ให้กับมอนสเตอร์แต่ละตัว เพื่อเพิ่มรูปแบบการต่อสู้มีหลากหลายยิ่งขึ้น

8. เพิ่มส่วนของ Option ให้มีการปรับเปลี่ยนรูปแบบการแสดงผลได้ เช่น ขนาดของหน้าต่างเกม จำนวนสีในการแสดงผล หรือสามารถเลือกรูปแบบของชุดคำสั่งในการ Render เพื่อให้ผู้เล่นสามารถปรับเปลี่ยนตามความชอบ

เอกสารอ้างอิง

[1] ไม่สามารถระบุผู้แต่ง. **"Reinforcement Learning."** [online].Available : http://researchers.in.th/blog/krikamol/229

[2] ไม่สามารถระบุผู้แต่ง. **"Artificial Intelligence."** [online].Available : http://smitch.exteen.com/20090217/entry-1

[3] ไม่สามารถระบุผู้แต่ง. **"Artificial Intelligence."** [online].Available : http://th.wikipedia.org/wiki/

[4] ไม่สามารถระบุผู้แต่ง. **"Irrlicht Engine."** [online].Available : http://irrlicht.sourceforge.net/

[5] ไม่สามารถระบุผู้แต่ง. **"Irrlicht Engine."** [online].Available : http://irrlicht.sourceforge.net/

[6] ไม่สามารถระบุผู้แต่ง. **"Irrlicht Engine."** [online].Available : http://www.exteen.com/tag/irrlicht

[7] ไม่สามารถระบุผู้แต่ง. **"C++ Library."** [online].Available : http://www.cplusplus.com/reference/clibrary/cmath/

[8] จุฑามาศ จิวะสังข์. **"สร้างงาน 3D ขั้นพื้นฐานจนถึงการใช้งานจริง 3ds Max 9."** กรุงเทพ ฯ : ซัคเซส มีเดีย 2550

ภาคผนวก

การใช้ IrrlichtDevice

IrrlichtDevice เป็นคลาสที่ใช้ในการสร้างหน้าต่างเกมขึ้น ซึ่งในส่วนการเขียนโปรแกรมมีการใช้คำสั่งดังนี้

```
IrrlichtDevice *device = createDevice(video::EDT_OPENGL,core::dimension2d<u32>(800,600),32,false);
                                      (1)               (2)                          (3) (4)
```

(1) เป็นการเลือกชุดคำสั่ง(Library) ที่ใช้ในการ Render ซึ่ง Irrlicht Game Engine สามารถรองรับได้ทั้ง OpenGL Direct3D และแบบ Software โดยจะใช้คำสั่งดังนี้

OpenGL ใช้คำสั่ง EDT_OPENGL

DirectX8 ใช้คำสั่ง EDT_DIRECT3D8

DirectX9 ใช้คำสั่ง EDT_DIRECT3D9

Software ใช้คำสั่ง EDT_SOFTWARE

(2) เป็นการกำหนดขนาดของการแสดงผล ดังตัวอย่างจะเป็นการแสดงผลที่ขนาด 800x600

(3) เป็นการกำหนดจำนวนบิตพักเซล ดังตัวอย่างจะเป็นการกำหนดที่ 32 บิต

(4) เป็นการกำหนดรูปแบบการแสดงผล ถ้า true คือแสดงเต็มจอ และ false คือแสดงไม่เต็มจอ

การโหลดโมเดลสามมิติ

```
scene::ISceneManager* smgr = device->getSceneManager();                    (1)
scene::IAnimatedMeshSceneNode* node = smgr->addAnimatedMeshSceneNode(
                    smgr->getMesh("house.x"));                             (2)
node->setPosition(core::vector3df(4550,50,5200));                          (3)
node->setScale(core::vector3df(5,5,3));                                    (4)
node->setRotation(core::vector3df(0,90,0));                                (5)
node ->setFrameLoop(0,30);                                                 (6)
node->setAnimationSpeed(10.f);                                             (7)
```

จากตัวอย่างเป็นการโหลดโมเดล 3 มิติ ที่เป็น file สกุล .X แล้วกำหนดค่าต่าง ๆ ของกับโมเดลดังนี้

(1) สร้าง ISceneManager ชื่อ smgr เพื่อให้เป็นตัวจัดการในส่วนของ Scene หรือการ Render ต่าง ๆ

(2) สร้าง IAnimatedMeshSceneNode ชื่อ node เพื่อเก็บโมเดล Animation ที่ชื่อ "house.x"

(3) เป็นการกำหนดพิกัดให้กับ node ซึ่งจะกำหนดเป็น vector สามมิติ โดยใช้ namespace irr::core มาช่วยสร้าง vector สามมิติ และค่าใน vector ก็คือ พิกัดใน map จริง ดังตัวอย่างจะเป็นการทำหนดพิกัดที่ x = 4550 y = 50 และ z = 5200

(4) เป็นการกำหนดขนาดให้กับ node ซึ่งจะกำหนดเป็น Vector สามมิติ โดยค่าใน Vector นั้นจะแสดงถึงการจำนวนเท่าของขนานเดิมทั้งสามแกน x y และ z ดังตัวอย่างเป็นการกำหนดให้ขนาดของโมเดลใหญ่ขึ้นจากเดิมตามแกน x และ y เป็นจำนวน 5 เท่า และขนาดของโมเดลใหญ่ขึ้นจากแกน z 3 เท่า

(5) เป็นการกำหนดการหมุนของโมเดล ซึ่งจะกำหนดเป็น Vector สามมิติ โดยค่าใน Vector นั้นคือ มุมที่เปลี่ยนไปตามแกน x y และ z ดังตัวอย่างจะเป็นการ หมุนโมเดลที่แกน y ไป 90 องศา

(6) เป็นการกำหนดการแสดง Frame ของโมเดลที่เป็น Animation ดังตัวอย่างจะเป็นการกำหนดให้แสดง Frame เริ่มต้นที่ Frame ที่ 0 เป็นสิ้นสุดที่ Frame ที่ 30 และให้แสดงวนไปเรื่อย ๆ

(7) เป็นการกำหนดความเร็วในการแสดงผล ซึ่งกำหนดเป็นค่า Frame ต่อวินาที

การแสดงภาพสองมิติ

```
gui::IGUIEnvironment* env = device->getGUIEnvironment();                    (1)
env->addImage(driver>getTexture("load.png"),
                                    core::position2d<s32>(100,100));        (2)
```

จากตัวอย่างเป็นการแสดงภาพสองมิติ ที่เป็น File สกุล .png แล้วกำหนดค่าต่าง ๆ ดังนี้

(1) สร้าง IGUIEnvironment ชื่อ env เพื่อให้เป็นตัวจัดการในส่วนของ GUI

(2) ให้ env เก็บรูปภาพ 2 มิติและกำหนดค่าพิกัดที่จุดแกน x และ แกน y ของหน้าต่าง ๆ เกม เป็น vector สองมิติ ดังตัวอย่างจะเป็นการแสดงรูป load.png และกำหนดพิกัดที่จุด 100 100

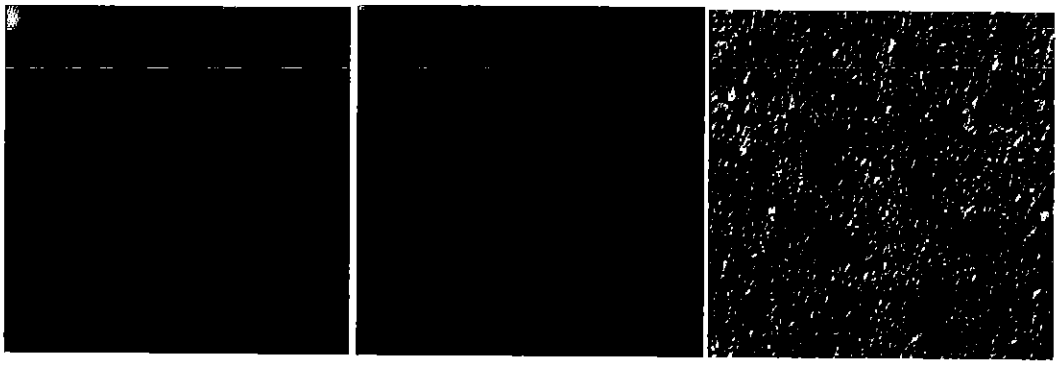
การกำหนดมุมกล้อง

```
scene::ICameraSceneNode* camera = 0;                                        (1)
camera->setPosition(core::vector3df(2200,300,2000));                        (2)
camera->setTarget(core::vector3df(2000,0,2000));                            (3)
```

(1) เป็นการสร้าง ICameraSceneNode ที่ชื่อ camera ซึ่งคือการสร้างโมเดลที่เป็นกล้องคอย กำหนดมุมมองการมองเห็น

(2) เป็นการกำหนดจุดที่เราจะเอากล้องไปวางไว้

(3) เป็นการกำหนดจุดที่เราจะให้กล้องหันไปทางจุดนั้น



รูปแสดง ลักษณะของภาพ Height Texture และ Detailmap

(1) เป็นการนำเอาภาพ height.bmp มากำหนดลักษณะของ terrain โดยการสร้างนั้นจะลักษณะการทำงานคือถ้าในภาพมีสีขาวจะสร้างเป็นพื้นที่ที่สูงขึ้นมา ส่วนที่ภาพสีเป็นสีดำก็จะเป็นพื้นที่ที่ต่ำลงไป ดังตัวอย่างก็จะเป็นการสร้าง terrain ที่ตรงกลางเป็นพื้นเรียบและขอบทั้งสี่ด้านเป็นพื้นสูงขึ้นมา

(2) เป็นการนำเอาภาพ texture.bmp มากำหนดสีของ terrain ทั้งหมด

(3) เป็นการนำเอาภาพ detailmap.jpg มากำหนดพื้นผิวของ terrain

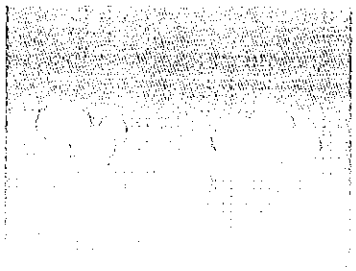
(4) เป็นการกำหนดขนาดของ terrain

การสร้างท้องฟ้า

```
scene::ISceneNode* skydome = 0;                                    (1)
skydome =
smgr->addSkyDomeSceneNode(driver->getTexture("sky.bmp"),15,7,1,1);(2)
```

(1) เป็นการสร้าง ISceneNode ที่ชื่อ skydome ขึ้นมา

(2) เป็นการสร้าง SkyDomeSceneNode โดยใช้ภาพ sky.bmp



รูปแสดง ลักษณะของภาพ sky.bmp

การตรวจจับการชนของวัตถุ

```
scene::ISceneNodeAnimator* anim = 0;                                    (1)
anim = smgr->createCollisionResponseAnimator(terrain,
      node,core::vector3df(5,5,5),core::vector3df(0,0,0));              (2)
                          (3)                  (4)
node->addAnimator(anim);                                                (5)
```

(1) คือการสร้าง ISceneNodeAnimator ที่ชื่อ anim ซึ่งคือการสร้าง Animation ของโมเดลขึ้นมา

(2) กำหนดให้ anim สร้าง Animation ชนิดตรวจจับการชนระหว่าง terrain ที่เป็น ITerrainSceneNode กับ node ที่เป็น IAnimatedMeshSceneNode

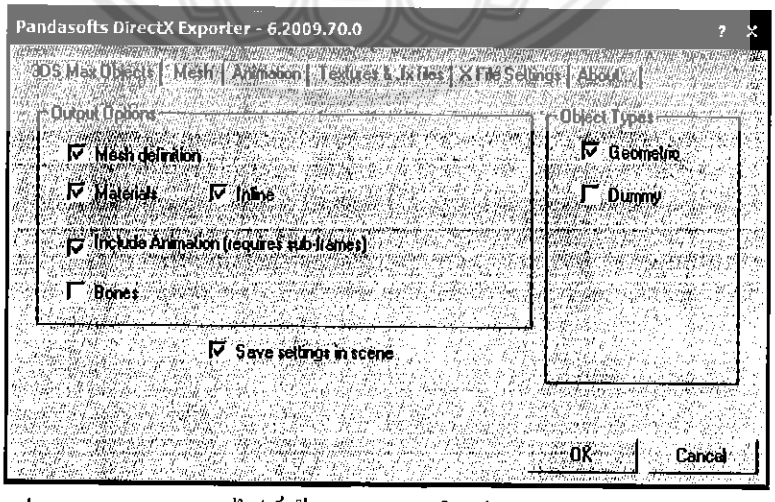
(3) เป็นการกำหนดรัศมีของการตรวจจับการชนทั้งสามแกน

(4) เป็นการกำหนดแรงโน้มถ่วงในกรณีที่ต้องการให้ Node ทั้งสองค่อย ๆ ตกมาชนกัน

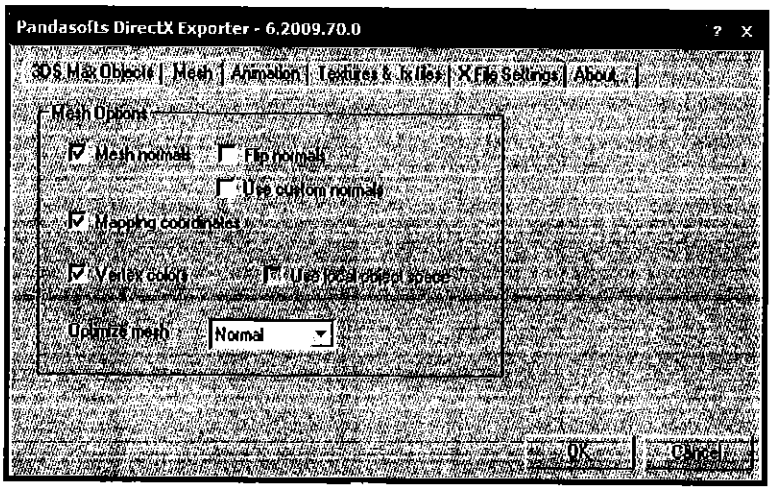
(5) เป็นการแปะ anim ซึ่งเป็น Animation ของโมเดลการชน ให้กับ node

การ Export files ในรูป .X (X file Exporting)

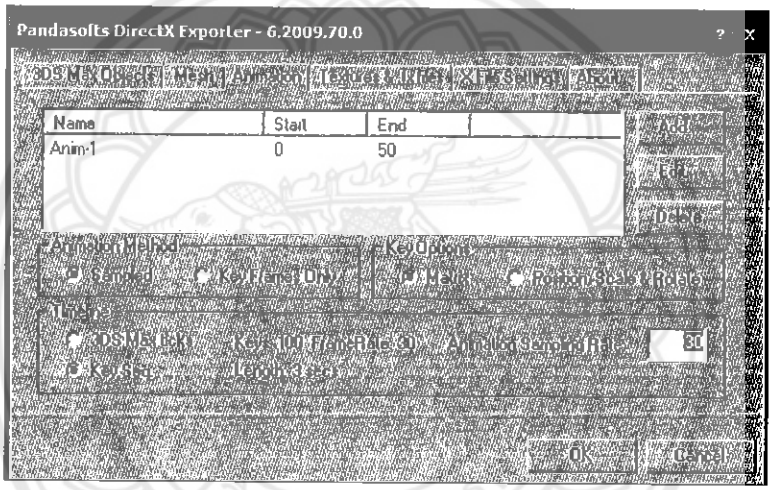
สำหรับการ Export ไฟล์ ออกมาเป็น .X file นั้น ใช้ Plug-in ของโปรแกรม 3ds max ที่ชื่อว่า Pandasofts DirectX Exporter โดยการกำหนดค่าให้มีการ export animation และ texture ที่ใช้ออกมาด้วย ดังรูป



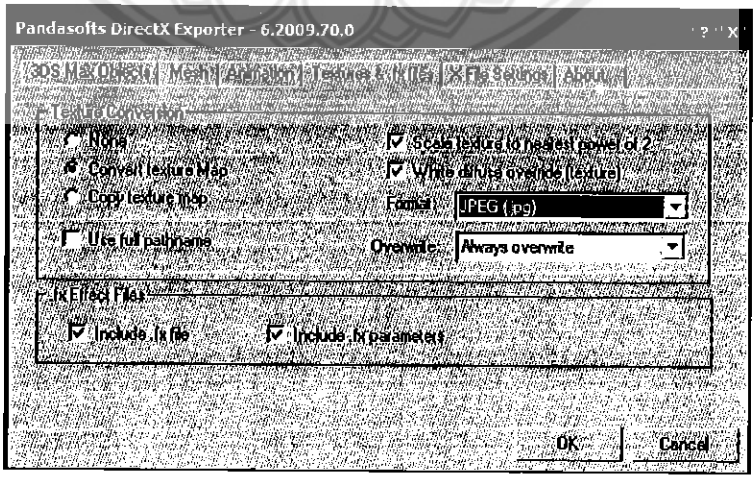
รูปแสดงการ Export ไฟล์ ด้วย PandaX ในส่วนของ 3DS Max Object



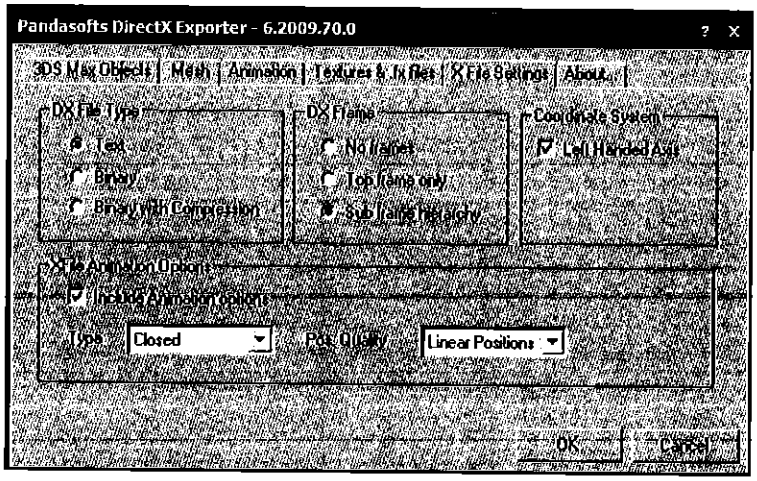
รูปแสดงการ Export ไฟล์ ด้วย PandaX ในส่วนของ Mesh



รูปแสดงการ Export ไฟล์ ด้วย PandaX ในส่วนของ Animation

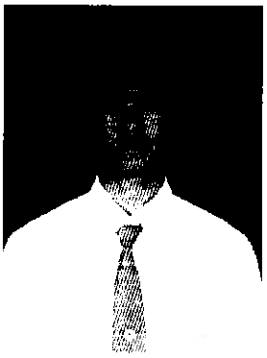


รูปแสดงการ Export ไฟล์ ด้วย PandaX ในส่วนของ Texture & .fx files



รูปแสดงการ Export ไฟล์ ด้วย PandaX ในส่วนของ X file Setting

ประวัติผู้เขียนโครงงาน



ชื่อ นายศรัณย์ พิริยโยธา

ภูมิลำเนา 152/399 ม.8 ต.ท่าโพธิ์ อ.เมือง จ.พิษณุโลก

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: **flyingsweet_pig@hotmail.com**

ชื่อ นายยุทธนา เรืองนุกูล

ภูมิลำเนา 72 ม.5 ต.ทุ่งหลวง อ.เมือง จ.สุโขทัย

ประวัติการศึกษา

- จบระดับมัธยมศึกษาจากโรงเรียนสุโขทัยวิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail: **VDa_JangWei@hotmail.com**