# การพัฒนาเกมต่อสู้แบบตัวต่อตัว มุมมองบุคคลที่สาม

# 3D THIRD PERSON FIGHTING

นราธิป บุญญาวานิชย์

วรวิช เสวตไอยาราม

สรพล บัวสุวรรณ



ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

คณะวิทยาศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

# 3D THIRD PERSON FIGHTING

**NARATHIP BOONYAVANICH**

**WRORAWISH SAWET-AI-YARAM**

**SORAPON BUASUWAN**

**A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT**

**OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE**

**DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE**

**FACULTY OF SCIENCE**

**KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG**

**ACADEMIC YEAR 2006**

| | | |
|---|---|---|
| **หัวข้อปัญหาพิเศษ** | การพัฒนาเกมต่อสู้แบบตัวต่อตัว มุมมองบุคคลที่สาม<br>3D THIRD PERSON FIGHTING | |
| **ชื่อนักศึกษา** | นายนราธิป บุญญาวานิชย์ | 46050297 |
| | นายวรวิช เสวตไอยาราม | 46050314 |
| | นายสรพล บัวสุวรรณ | 46050319 |
| **ภาควิชา** | คณิตศาสตร์และวิทยาการคอมพิวเตอร์ | |
| **สาขาวิชา** | วิทยาการคอมพิวเตอร์ | |
| **อาจารย์ที่ปรึกษา** | รศ.ธีรวัฒน์ ประกอบผล | |

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2549

| คณะกรรมการสอบ | | ลายมือชื่อ |
|---|---|---|
| ประธานกรรมการ | อ.ธีระ ฟักอ่อน | |
| กรรมการ | ดร.นวลสวาท หิรัญสกลวงศ์ | |
| กรรมการและอาจารย์ที่ปรึกษา | รศ.ธีรวัฒน์ ประกอบผล | |

(รองศาสตราจารย์ ดร.วีระ บุญจริง)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

| | | |
|---|---|---|
| **หัวข้อปัญหาพิเศษ** | การพัฒนาเกมต่อสู้แบบตัวต่อตัว มุมมองบุคคลที่สาม | |
| **ชื่อนักศึกษา** | นายนราธิป บุญญาวานิชย์ | 46050297 |
| | นายวรวิช เสวตไอยาราม | 46050314 |
| | นางสรพล บัวสุวรรณ | 46050319 |
| **ปริญญา** | วิทยาศาสตรบัณฑิต | |
| **ภาควิชา** | คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ | |
| **สาขาวิชา** | วิทยาการคอมพิวเตอร์ | |
| **ปีการศึกษา** | 2549 | |
| **อาจารย์ที่ปรึกษา** | รศ.ธีรวัฒน์ ประกอบผล | |


## บทคัดย่อ


ปัญหาพิเศษนี้เป็นการพัฒนาเกมต่อสู้แบบตัวต่อตัว มุมมองบุคคลที่สาม ในลักษณะที่ผู้เล่นจะต้องทำการต่อสู้กับศัตรูที่ควบคุมโดยระบบปัญญาประดิษฐ์ โดยระบบของเกมมีอาวุธให้เลือกใช้ได้หนึ่งอย่าง คือ ปืนเลเซอร์ ผู้เล่นจะต้องอาศัยอาวุธนี้ในการต่อสู้ ไม่ว่าจะเป็นการแอบซ่อน การหลบหลีก การยิงกระสุน และการวางแผนในการต่อสู้ กับระบบปัญญาประดิษฐ์ที่มีความฉลาดไม่ต่างไปจากผู้เล่นคนหนึ่ง โดยรูปแบบการเล่นนั้น ผู้เล่นสามารถเล่นได้คนเดียว ซึ่งเกมต่อสู้แบบตัวต่อตัว มุมมองบุคคลที่สามนี้ พัฒนาขึ้นมาโดยใช้ Quest3D เอนจิ้น และในส่วนของตัวละครกับฉากนั้น จะสร้างขึ้นจากโปรแกรม Maya ซึ่งช่วยให้เกมมีภาพสามมิติที่สวยงามและลงตัว โดยภายในเกมยังประกอบไปด้วยเสียงประกอบต่างๆ ซึ่งจะช่วยเพิ่มความสนุกสนานและความตื่นเต้นให้กับผู้เล่นมากยิ่งขึ้น

ในการพัฒนาปัญหาพิเศษนี้ ทางทีมงานได้ศึกษาและออกแบบระบบของเกมขึ้นมาด้วยแรงบันดาลใจจากแนวเกมที่ชื่นชอบ โดยทางทีมงานได้ใช้ระยะเวลาทั้งหมดในการสร้างปัญหาพิเศษชิ้นนี้เป็นระยะเวลาทั้งสิ้นหนึ่งปีการศึกษา ตั้งแต่การออกแบบเนื้อหาเกม การออกแบบกติกาในการเล่น การออกแบบตัวละครและฉาก การเรียนรู้ถึงวิธีสร้างวัตถุสามมิติเพื่อใช้ในเกม การพัฒนาตัวเกมและการพัฒนาระบบให้มีประสิทธิภาพ จนถึง การทดสอบและแก้ปัญหาที่เกิดขึ้นกับตัวเกมทั้งหมด ตัวเกมจะมีการตอบสนองต่อผู้เล่นตามสถานการณ์ต่างๆผ่านหน้าจอคอมพิวเตอร์ รวมถึงมีการรับคำสั่งจากผู้เล่นด้วยเป็นพิมพ์และเมาส์ ซึ่งจะสร้างความสนุกสนานให้กับผู้เล่นได้เป็นอย่างดี

| | | |
|---|---|---|
| **Special Project Title** | 3D THIRD PERSON FIGHTING | |
| **Students** | Mr. Narathip Boonyavanich | 46050297 |
| | Mr. Wrorawish Sawet-ai-yaram | 46050314 |
| | Mr. Sorapon Buasuwan | 46050319 |
| **Degree** | Bachelor of Science | |
| **Department** | Mathematics and Computer Sciences, Faculty of Science | |
| **Programme** | Computer Science | |
| **Academic Year** | 2006 | |
| **Special Project Advisor** | Assoc.Prof.Teerawat Prakobphon | |


## ABSTRACT


This special problem focuses on developing a third-person fighting game in 3D. A robot under control of a player has to fight with an enemy controlled by artificial intelligence. The system of this game offers one weapon, a laser gun, using for fighting. The player must use all of his or her abilities for hiding, seeking, dodging, shooting and making plan for reaching the highest point of the game to be the winner. However, it won't be easy to take an advantage from the enemy that is no different to a human being. Program supports only single player mode. Development tool of this project, the Quest3D engine is one of the best engines available today. The 3D objects were created by the most powerful 3D animation tool in the world called Maya. Many sound effects can be found inside game, by this, the player will absorb a good time while playing, spending it with no regret.

Our inspiration of this special problem came from a style of game that we like. We spent days and nights on the game, looking forward to complete it with the greatest quality. One year of hard works, created rules for the game, created our own models, designed the system and functionalities, bug fixing, and more. We hope that what we have been engrossed in will make all people that will be playing our game happy.

# กิตติกรรมประกาศ

ในการทำปัญหาพิเศษเรื่องเกมต่อสู้แบบตัวต่อตัว มุมมองบุคคลที่สาม (3D Third Person Fighting) สามารถสำเร็จลุล่วงไปได้ด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์ธีรวัฒน์ ประกอบผล อาจารย์ผู้รับผิดชอบปัญหาพิเศษฉบับนี้ที่กรุณาให้คำแนะนำ และเป็นที่ปรึกษาในการแก้ปัญหาต่างๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนการทำปัญหาพิเศษครั้งนี้สำเร็จไปด้วยดี รวมทั้งเพื่อนๆ และน้องๆ ที่ให้ความช่วยเหลือในด้านต่างๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2550

# สารบัญ

หน้า

# สารบัญ ( ต่อ )

หน้า

# สารบัญ ( ต่อ )

หน้า

# สารบัญ ( ต่อ )

หน้า

# สารบัญ ( ต่อ )

หน้า

# สารบัญ ( ต่อ )

หน้า

# สารบัญตาราง

# สารบัญรูป

# สารบัญรูป ( ต่อ )

# สารบัญรูป ( ต่อ )

# สารบัญรูป ( ต่อ )

# บทที่ 1
# บทนำ

## 1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันเกมคอมพิวเตอร์นั้นนับได้ว่ามีพัฒนาการที่ก้าวไกลเป็นอย่างมาก นอกจากพัฒนาการด้านประสิทธิภาพความสวยงามของภาพแล้ว ระบบปัญญาประดิษฐ์ที่ทำหน้าที่ตอบโต้กับผู้เล่นนั้นยังมีความฉลาดเพิ่มขึ้นเป็นอย่างยิ่งอีกด้วย สามารถสังเกตได้จากพฤติกรรมการตัดสินใจที่มีความหลากหลายในแต่ละสถานการณ์ โดยจะเปลี่ยนแปลงไปตามลักษณะการควบคุมตัวละครหรือวัตถุต่างๆภายในเกมของผู้เล่น จึงเป็นที่มาของความสนใจในการที่จะวิจัย และศึกษาหลักการเขียนอัลกอริทึมของระบบปัญญาประดิษฐ์ในเกมคอมพิวเตอร์

เนื่องจาก ในปัจจุบันแนวเกมต่อสู้แบบมุมมองบุคคลที่สามกำลังได้รับความนิยมอย่างมาก อีกทั้งยังเป็นแนวเกมที่ฝึกฝนประสาทสัมผัสของผู้เล่นให้มีความฉับไวในการตัดสินใจมากขึ้นด้วย จึงได้ตัดสินใจเลือกที่จะศึกษาและทดลองเขียนระบบปัญญาประดิษฐ์ของเกมชนิดนี้ขึ้น ซึ่งมีจุดสนใจตรงที่ การเขียนระบบปัญญาประดิษฐ์ชนิดนี้นั้น จะต้องมีการควบคุมพฤติกรรมตามหน่วยเวลาอย่างเข้มงวด เพื่อให้สามารถตอบโต้กับผู้เล่นที่บังคับตัวละครได้อย่างอิสระภายใต้โลกเสมือนได้อย่างทันท่วงที อีกทั้งเกมแนวนี้นั้นยังต้องมีการนำความรู้ทางคณิตศาสตร์เข้ามาเพื่อช่วยกำหนดลักษณะการเคลื่อนที่อีกด้วย ทำให้นอกจากจะได้ศึกษาถึงหลักการเขียนปัญญาประดิษฐ์แล้วก็ยังจะช่วยให้มีโอกาสได้นำความรู้ทางคณิตศาสตร์ที่ได้ผ่านการศึกษา มาเพื่อใช้ในโครงงานนี้อีกด้วย

ทั้งนี้แรงบันดาลใจส่วนหนึ่งนั้นเกิดขึ้นมาจาก การที่ได้มีโอกาสศึกษาการเขียนภาพประมวลผลสามมิติในวิชา Advanced Graphic โดยใช้ OpenGL และความประทับใจในเกมต่างๆที่ได้เล่นมา จึงได้หวังผลของงานชิ้นนี้ไว้ว่า ผู้ที่ได้มีโอกาสทดลองเล่นจะมีความรู้สึกสนุกและได้แรงบันดาลใจที่จะสร้างสรรค์งานเกมของตนเองในอนาคตบ้าง หรือแม้กระทั่งเป็นการกระตุ้นให้บุคคลต่างๆได้เห็นถึงความสำคัญของเกมว่าไม่ได้เป็นแค่เพียงสื่อบันเทิงเท่านั้น แต่เป็นความสำเร็จของการที่พิสูจน์ให้เห็นได้ว่า คณิตศาสตร์นั้นสามารถอยู่ร่วมกับจินตนาการได้ รวมทั้งเป็นแรงดลใจในการที่จะคิดค้น หลักของคณิตศาสตร์ในการคำนวณของกลไกธรรมชาติชั้นสูงขึ้นไปกว่านี้ ไม่ว่าจะเป็น การเคลื่อนที่ การตกกระทบของแสง หรือแม้กระทั่งการคำนวณเพื่อสร้างรูปร่างของวัตถุ

## 1.2 ความมุ่งหมายและวัตถุประสงค์ ของการศึกษา

1.2.1 เพื่อศึกษาและทดลองหลักการเขียนโปรแกรมปัญญาประดิษฐ์ เชิงระบบต่อสู้แบบตัวต่อตัว ในมุมมองบุคคลที่สาม

1.2.2 เพื่อศึกษาและทดลองนำหลักคณิตศาสตร์มาประยุกต์ในเชิงคำนวณกลศาสตร์การเคลื่อนที่ภายในเกม

1.2.3 เพื่อฝึกการออกแบบระบบเกม การจัดความสมดุลให้กับระบบเกม และฝึกฝนจินตนาการในการสร้างเกม

1.2.4 เพื่อเป็นแนวทางแก่ผู้ที่สนใจจะพัฒนาและศึกษาการทำงานของปัญญาประดิษฐ์

## 1.3 ขอบเขตของการศึกษา

1.3.1 มีการพัฒนาปัญญาประดิษฐ์ในลักษณะของการต่อสู้กับผู้เล่นแบบตัวต่อตัว ซึ่งมีการเคลื่อนไหวจำกัดรูปแบบในสภาวะแวดล้อมที่กำหนด

1.3.2 มีรูปแบบเกมเป็นลักษณะการต่อสู้แบบตัวต่อตัวแนวมุมมองบุคคลที่สาม

1.3.3 มีการสร้างตัวละครที่เป็นตัวเอก ซึ่งมีความแตกต่างทั้งในด้านความสามารถและรูปร่าง ไว้ให้ผู้ เล่นหรือปัญญาประดิษฐ์เลือกใช้ รวมถึงมีการใส่เสียงเพลงประกอบเพื่อเพิ่มความสนุกสนานในการเล่นเกม

## 1.4 ประโยชน์ที่ได้รับจากการศึกษา

1.4.1 ได้เรียนรู้และทดสอบการเขียนปัญญาประดิษฐ์สำหรับเกม

1.4.2 ได้เรียนรู้วิธีการใช้โปรแกรม “Maya” ในการสร้างตัวละครและวัตถุสามมิติเพื่อใช้ในเกม

1.4.3 ได้เรียนรู้วิธีการใช้โปรแกรม “Quest3D” ซึ่งเป็นโปรแกรมระดับมืออาชีพ ซึ่งสามารถสร้างเกมสามมิติขึ้นมาได้อย่างมีประสิทธิภาพ

1.4.4 ได้มีโอกาสแสดงและสร้างสรรค์จินตนาการ จากที่เป็นแค่เพียงผู้บริโภค กลายมาเป็นผู้ผลิต นอกจากนั้นยังเป็นการเปิดโลกทัศน์ในมุมมองต่างๆด้วย

## 1.5 ขั้นตอนของการศึกษา

1.5.1 ออกแบบระบบการเล่นของเกม ไม่ว่าจะเป็น กำหนดจำนวนตัวละครที่จะใช้ ความสามารถของตัวละคร ฉาก และระบบติดต่อกับผู้เล่นเบื้องต้น

1.5.2 ศึกษาโปรแกรมพัฒนาเกมและคัดสรรตัวที่เหมาะสมมาเพื่อใช้ในการสร้าง

1.5.3 ศึกษาโปรแกรมพัฒนาเกมที่เลือกมาอย่างละเอียด เพื่อกำหนดฟังก์ชันที่จะต้องใช้ในการพัฒนาระบบเกม

1.5.4 ศึกษาโปรแกรมที่ใช้สร้างวัตถุสามมิติ

1.5.5 ทดสอบระบบเบื้องต้นโดยการนำวัตถุสามมิติที่สร้างขึ้น ไปใส่ลงในโปรแกรมพัฒนาเกม ตรวจสอบคุณภาพและข้อผิดพลาดที่อาจเกิดขึ้น

1.5.6 จัดสรรทรัพยากรต่างๆในเชิงผนวก เช่น ใส่เสียง สร้างหน้าเมนู ใส่การเคลื่อนไหวและเงื่อนไขให้กับระบบและวัตถุ

1.5.7 พัฒนาโปรแกรมเกมให้เป็นรุ่นทดลอง เพื่อทดสอบข้อผิดพลาด และดูความสมดุลของตัวเกม

1.5.8 แก้ข้อผิดพลาด และเพิ่มเติมระบบในส่วนที่ขาด ทำในขั้นตอนนี้จนกว่าเกมจะสมบูรณ์ถึงเกณฑ์มาตรฐานที่ได้ตั้งไว้

1.5.9 พัฒนาโปรแกรมจนสามารถกำหนดให้เป็นรุ่นสมบูรณ์ได้

# บทที่ 2
# หลักการและทฤษฎีที่เกี่ยวข้อง

## 2.1 Quest3D

Quest3D เป็นโปรแกรมที่มีชุดคำสั่งที่มีไว้สำหรับในการสร้างงานที่มีการตอบสนองในรูปแบบ 3มิติ โดยลักษณะพิเศษของโปรแกรมQuest3Dจะเป็นรูปแบบของภาษาโปรแกรมที่มีลักษณะเฉพาะแทนการเขียนโค้ดโปรแกรมที่มีความซับซ้อน โดยการทำให้อยู่ในรูปคำสั่งที่เป็นแบบบล็อคกับบล็อคด้วยเส้นเชื่อม เพื่อให้มีความยืดหยุ่นและง่ายต่อการใช้งาน การทำงานของโปรแกรมQuest3Dในลักษณะการตอบสนองทันที โดยจะขึ้นอยู่กับผลลัพธ์สุดท้ายที่ได้ทำการเปลี่ยนแปลงโครงสร้างของกราฟ ซึ่งจะไม่สูญเสียเวลาในการคอมไพเลอร์หรือการประมวลผลภาพ3มิติของแสงให้กับโมเดล

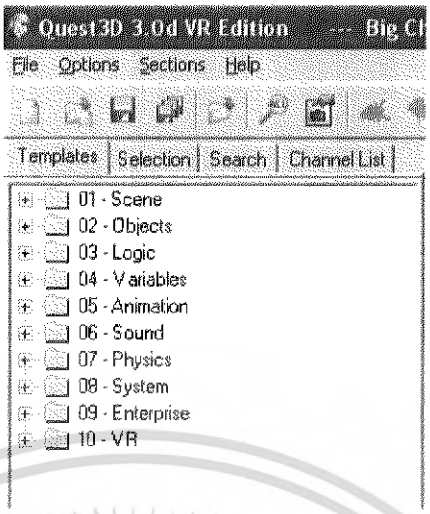
### 2.1.1 วิธีการใช้งานของโปรแกรมQuest3D

Quest3D พยายามที่จะจัดหน้าจอการทำงานของโปรแกรม เพื่อให้สามารถใช้ประโยชน์จากความสามารถของโปรแกรมได้อย่างเต็มประสิทธิภาพ ดังนั้นเราจึงทำการแบ่งหน้าจอการทำงานเป็นส่วนๆ เรียกว่าSection เพื่อที่จะนำเสนอโครงสร้างของเครื่องมือให้ได้มากที่สุดเท่าที่จะเป็นไปได้ โดยมีส่วนที่สำคัญที่สุด3ส่วนดังรายการต่อไปนี้

- **Channel Section** เป็นหัวใจหลักของQuest3D Quest3DจะแสดงChannel Section ในตอนเริ่มต้นสำหรับในส่วนนี้เป็นพื้นฐานในการสร้างเกม
- **Animation Section** เป็นส่วนที่ใช้ในการวางตำแหน่งของวัตถุสามมิติ,กล้อง และแสง รวมถึงการกำหนดการเคลื่อนที่ ซึ่งในส่วนนี้จะมีส่วนของการแสดงผลที่มีขนาดใหญ่ เพื่อใช้ในการทดสอบการทำงานของโปรแกรมง่ายยิ่งขึ้น
- **Object Section** เป็นส่วนที่ใช้สำหรับจัดการกับวัตถุสามมิติ เช่นการจัดการกับสีและพื้นผิวของวัตถุ

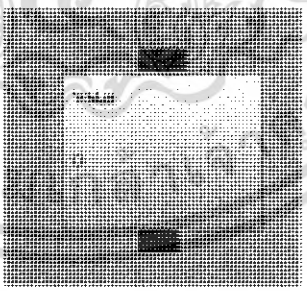
#### 2.1.1.1 การเพิ่ม Channel ในโปรแกรม

Template List ใช้สำหรับเก็บBuilding Blocksที่มีทั้งหมดของQuest3D



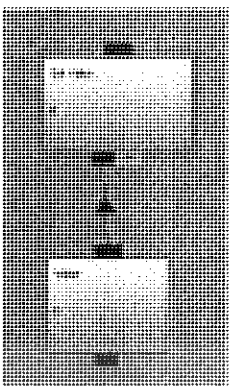

**รูปที่ 2.1** แสดงฟังก์ชันต่างๆภายในQuest3D

ในการเพิ่ม Channelใหม่เข้าไปในโปรแกรม สามารถทำได้โดยการลากชื่อจากTemplate List ไปไว้บนChannel Group โดยการลากหมายถึงการกดเมาส์ปุ่มซ้ายค้างไว้แล้วย้ายลูกศรไปไว้ที่ตำแหน่งใหม่ จากนั้นก็ปล่อยปุ่มเมาส์ Template เป็นChannelที่ถูกสร้างไว้ก่อนแล้ว การใช้Templateสามารถทำให้งานที่สร้างเร็วขึ้น โดยTemplateสามารถเพิ่มเข้าไปได้เหมือนกับการเพิ่มChannelเดียว โปรแกรมที่ถูกสร้างขึ้นจากQuest3Dนั้น ประกอบไปด้วยBuilding Blocks โดยเราจะเรียกว่าbuilding blockเหล่านี้ว่า Channel และในแต่ละส่วน Channel ก็จะมีคุณสมบัติเฉพาะตัวไม่เหมือนกัน ในQuest3D Channelคือโหนดที่มีลักษณะเป็นสี่เหลี่ยม Channelที่อยู่ในภาพข้างล่างนี้คือ Value Channel สำหรับเก็บค่าตัวเลข



**รูปที่ 2.2** แสดงโหนดValue Channel

สี่เหลี่ยมขนาดเล็กสีดำที่อยู่ด้านบนและด้านล่างของChannel เรียกว่า Link Squares ซึ่งสามารถเชื่อมต่อChannelเข้าด้วยกันได้โดยการลากเส้นจากLink Squaresเชื่อมต่อกัน



**รูปที่ 2.3** แสดงการเชื่อมเส้นระหว่างโหนดหนึ่งไปอีกโหนดหนึ่ง

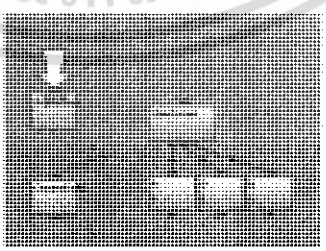
ในภาพข้างบนChannel ที่อยู่ด้านบนเรียกว่า โหนดแม่และChannel ที่อยู่ด้านล่างเรียกว่าโหนดลูก โดยโหนดลูกมักจะถูกใช้เป็นข้อมูลเข้าหรือข้อมูลออกสำหรับโหนดแม่ สำหรับLink Squareที่เป็นข้อมูลเข้าที่ไม่มีการเชื่อมต่อกับโหนดลูกจะถือว่าไม่มีข้อมูลเข้า โดยปกติแล้วจะมีการกำหนดค่าเริ่มต้นให้เป็น0

**2.1.1.2 Channel Group**

โครงสร้างที่สร้างขึ้นมาจากการเชื่อมต่อกันของChannelเรียกว่าChannel Group ในโปรเจคของQuest3Dประกอบไปด้วย Channel Group หนึ่งหรือหลายๆส่วนประกอบกัน ตัวอย่างเช่น กลุ่มหนึ่งอาจจะสร้างห้องเพื่อแสดงผลทางหน้าจอในขณะที่กลุ่มอื่นอาจจะสร้างตัวละคร โดยChannel Group ทั้งสองนี้อาจจะนำมารวมกันเพื่อแสดงผลให้เกิดตัวละครที่อยู่ในห้องก็ได้

**2.1.1.3 การเชื่อมเส้นเชื่อม**

เมื่อเลื่อนเมาส์มาวางไว้บน Link Square ด้านบนของ Channel caller จะเห็นว่าลิงค์ถูกเปลี่ยนเป็นสีส้ม



**รูปที่ 2.4** แสดงลักษณะการเชื่อมเส้นเชื่อม

ระหว่างที่มีการลิงค์ จะเห็นว่า Link Square ทุกๆอันจะเปลี่ยนสีไปเป็นสีแดงหรือสีเขียว โดยที่สีเขียวจะหมายถึงสามารถเชื่อมต่อกันได้ และสีแดงหมายถึงไม่สามารถเชื่อมต่อกันได้ เมื่อปล่อยปุ่มซ้ายของเมาส์บนLink Square จะเป็นการสร้างเชื่อมต่อ ถ้าหากการเชื่อมต่อนั้นถูกต้อง หรือ สามารถลากลิงค์ไปปล่อยบนChannelนั้นๆก็ได้ Quest3Dจะทำการเชื่อมต่อChannelเข้ากับlinkที่สามารถเชื่อมต่อกันได้เองโดยอัตโนมัติ

**2.1.1.4 การลบเส้นเชื่อม**

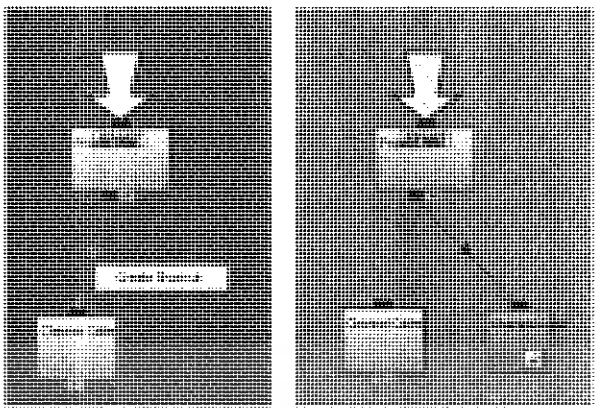
เลื่อนเมาส์ไปวางไว้บนลูกศรของลิงค์ที่เชื่อมต่อระหว่างChannel แล้วกดปุ่มซ้ายของเมาส์ จะสามารถเลือกlinkได้ จากนั้นถ้าต้องการลบลิงค์ ให้กดปุ่มDeleteบนแป้นพิมพ์

รูปที่ 2.5 แสดงการลบเส้นเชื่อม

ถ้าหากมีการลบเส้นเชื่อมออกจากChannelที่มีการสร้างเส้นเชื่อมขึ้นมาใหม่ Link Squareที่ว่างทางด้านซ้าย
จะเปลี่ยนเป็นสีน้ำเงิน โดยChannelจะทำการเว้นให้เป็นลิงค์ว่าง จนกว่าจะทำการสร้างเส้นเชื่อมใหม่ให้กับChannel

**2.1.1.5 การสร้าง Shortcut**

Channel Shortcut มีหน้าที่เหมือนกันกับShortcutทั่วๆไปในโปรแกรมวินโดว์ ใน Quest3D สามารถสร้างShortcutให้กับChannelเท่านั้น โดยการคลิกขวาบนลูกศรของเส้นที่เชื่อมต่อระหว่างChannel ทั้งสองจะปรากฏรายการขึ้นมาให้เลือก



**รูปที่ 2.6** แสดงการสร้างShortcut

Shortcut จะทำหน้าที่อ้างอิงการเชื่อมต่อChannelต้นแบบ เพื่อทำให้การจัดเรียงตัวของChannel Group ดูไม่รกจนเกินไป โปรแกรมที่ถูกสร้างขึ้นจากQuest3Dจะเริ่มต้นจากStart Channelซึ่งจะเป็น Channelที่มีเครื่องหมายลูกศรขนาดใหญ่ชี้อยู่ด้านบนของChannel ในการกำหนดChannel ใดๆให้เป็น Channelเริ่มต้น สามารถทำได้โดยการเลือกรายการ Set as start Channel จาก Channel ที่ต้องการ โปรเจคของQuest3Dจะทำงานก็ต่อเมื่อได้มีการเปิดหน้าต่างAnimation 3D Viewเท่านั้น

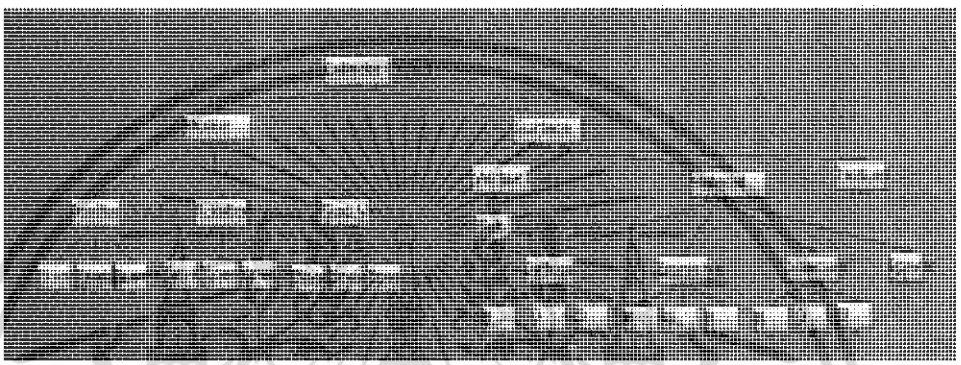
**2.1.1.6 Frame**

Quest3Dมีการทำงานแบบตอบสนองทันที ซึ่งหมายความว่าเมื่อมีการเปลี่ยนแปลงใดๆในโปรเจค ผลลัพธ์ก็จะเปลี่ยนไปและแสดงผลในทันที ซึ่งในหนึ่งรอบของการคำนวณที่เสร็จสมบูรณ์ของChannel structureเรียกว่าFrameและผลที่ได้จากการคำนวณทั้งหมดเรียกว่าRendering Framerate หมายถึง จำนวนครั้งในการคำนวณของโปรแกรมเพื่อใช้ในการแสดงผลต่อหนึ่งวินาที โดยที่ Framerateนั้นจะขึ้นอยู่กับความซับซ้อนของโปรเจ็คและประสิทธิภาพของเครื่องคอมพิวเตอร์ที่ใช้ ในการที่Channelหนึ่งๆจะรับผลลัพธ์ที่ได้จากการคำนวณของChannel ที่เป็นโหนดลูกของมันเพื่อใช้ในการประมวลผลต่อในตัวมันเองนั้น เราจะเรียกว่า Calling ซึ่งการรับผลลัพธ์จากโหนดลูกเข้ามานั้นขึ้นอยู่กับประเภทของแต่ละChannelด้วย

**2.1.2 ฟังก์ชันต่างๆในQuest3D**

วัตถุสามมิติในคอมพิวเตอร์ ประกอบไปด้วย วัตถุเสมือนจริง ซึ่งวัตถุเหล่านี้ก็จะถูกสร้างขึ้นจากองค์ประกอบต่างๆ ดังนี้ Vertex เป็นจุดๆหนึ่งที่อยู่ในพื้นที่3มิติ ถูกกำหนดตำแหน่งด้วยระบบพิกัด X ,Y และ Z สำหรับวัตถุทรงพื้นฐาน เช่น กล่องทรงลูกบาศก์นั้นประกอบไปด้วย vertex จำนวนไม่มากนัก แต่สำหรับวัตถุที่มีรูปร่างซับซ้อน เช่น รถยนต์ อาคาร หรือ ตัวละคร จะประกอบไป

ด้วยจุดจำนวนมาก Surface ถูกสร้างขึ้นมาจาก Polygon โดย Polygon นั้นจะประกอบไปด้วย vertex หลายๆจุด ซึ่งสำหรับโปรแกรมกราฟฟิกแบบตอบสนองทันที นั้นมักจะใช้รูปแบบของสามเหลี่ยมหรือสี่เหลี่ยมในการสร้างเป็นรูปทรงของวัตถุสำหรับในQuest3Dนั้น จะสร้างsurface ที่ประกอบไปด้วยจุด vertex3จุด Texture เป็นภาพที่สามารถนำไปใช้ในการห่อหุ้มวัตถุ3มิติเพื่อให้แสดงพื้นผิววัตถุ สำหรับการแสดงผลกราฟฟิกแบบตอบสนองทันที นั้น textureมักจะถูกใช้ในการกำหนดสี และ ความโปร่งใสของวัตถุสามมิติในQuest3D ถูกแบ่งเป็นสองส่วนใหญ่ๆ คือ motion และ surface



รูปที่ 2.7 แสดงโหนดของวัตถุทรงกระบอก

ลิงค์ทั้งสามที่เชื่อมต่ออยู่กับ Motion Channel คือ Vector ใช้สำหรับกำหนดค่าของ ตำแหน่ง การหมุน และ ขนาด ในแต่ละตัวของVector จะประกอบไปด้วย Value อีกตัวละ3ค่า แต่สำหรับในการทำอนิเมชัน สามารถเปลี่ยนจาก Value ไปเป็น Envelope ได้ ในส่วนของรูปร่างของวัตถุสามมิติในQuest3Dนั้น จะถูกกำหนดจากในส่วนที่สองของโครงสร้างนี้ ซึ่งลิงค์ที่เชื่อมเข้ากับSurface Channel ประกอบไปด้วย 3D ObjectData,Material และ Texture 3D ObjectData จะเก็บค่า vertex และ polygon ของวัตถุ ซึ่งจะใช้เป็นข้อมูลในการกำหนดวิธีการห่อหุ่มtextureเข้ากับรูปร่างของวัตถุ

Material ประกอบไปด้วยองค์ประกอบต่างๆ คือ

- Diffuse vector เก็บค่าสีของวัตถุ มีสามองค์ประกอบคือ สีแดง,สีเขียว และสีฟ้า หรือ RGB
- Emissive vector เป็นค่าความสว่างที่เกิดขึ้นจากตัวของวัตถุ เมื่อไม่มีแสงจากภายนอกมากมาย วัตถุที่มีค่าEmissive vectorเป็น(0,0,0)จะเป็นสีดำ
- Specular vector กำหนดค่าสีและความสว่างของไฮไลท์ โดยมีค่าPower ในการควบ

คุมขนาดของจุดไฮไลท์

- Alpha Diffuse กำหนดค่าความโปร่งใสของวัตถุ
- Texture Blend Factor กำหนดค่าการผสมกันระหว่างtexture
- Texture Channel ใช้สำหรับกำหนดภาพให้กับ surface ซึ่งจะมีค่า diffuse(color)และบางครั้งอาจจะมีค่า alphaด้วย

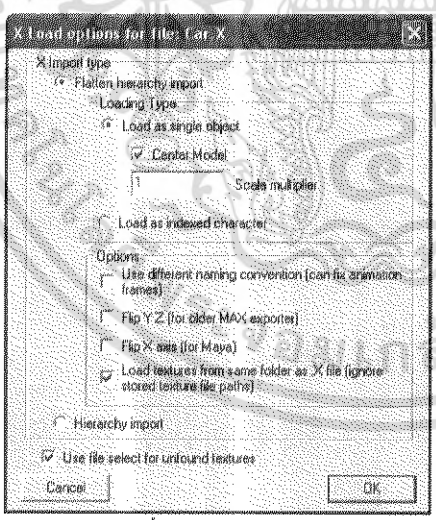
ขั้นตอนการนำเข้าวัตถุ3มิติ มี2แบบ คือ

1.วัตถุ3มิติที่เป็นแบบไม่เคลื่อนไหว

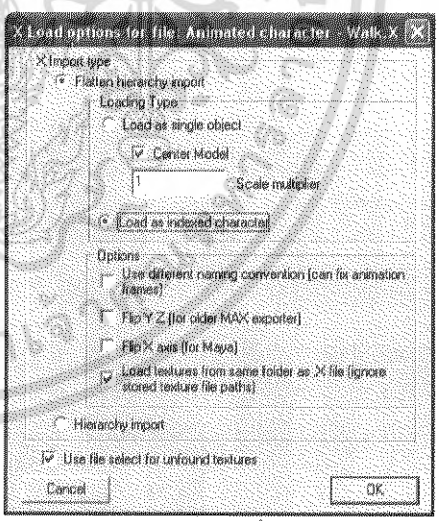
2.วัตถุ3มิติที่เป็นแบบเคลื่อนไหว

ขั้นตอนการนำเข้าวัตถุ3มิติ

1. ในQuest3D ไปที่ Application menu เลือก File->Import
2. เลือกไฟล์.xที่สร้างขึ้น จะปรากฏหน้าต่างเพื่อให้พิมพ์ชื่อ แล้วกดOK
3. ให้ใช้ค่าที่กำหนดให้ในหน้าต่าง.x Object Importer Options
   - ถ้าเป็นวัตถุ3มิติที่เป็นแบบไม่เคลื่อนไหวให้เลือกหน้าต่างดังรูปที่ a
   - ถ้าเป็นวัตถุ3มิติที่เป็นแบบเคลื่อนไหวให้เลือกหน้าต่างดังรูปที่ b




รูปภาพที่ a




รูปภาพที่ b

**รูปที่ 2.8** แสดงรูปแบบการนำเข้าวัตถุ3มิติ

4. สังเกตว่าวัตถุสามมิติจาก.X file ได้ถูกนำเข้ามาในQuest3D โดยจะถูกจัดเก็บในรูปแบบของ3D Object Channel structure

5. ลาก Simple Scene template ไปวางบน Channel Group เหนือ Object Channel

- **อนิเมชัน**

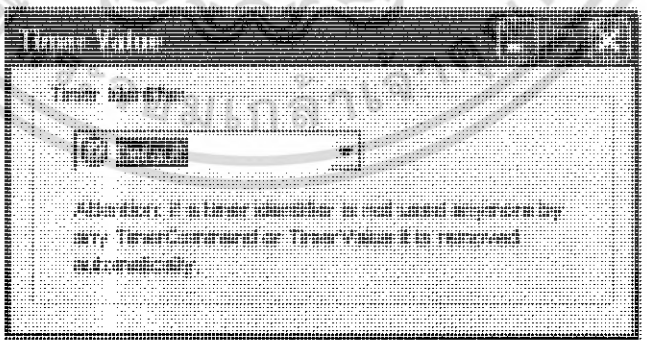
ในQuest3D สามารถที่จะกำหนดการเคลื่อนไหวให้กับสิ่งต่างๆในฉากได้ ยกตัวอย่าง เช่น วัตถุ3มิติ สามารถเคลื่อนที่ หมุน และย่อ/ขยายได้ตามเวลาที่กำหนด นอกจากนั้น คุณสมบัติทั้งหมดของsurface ก็สามารถกำหนดชนิดอนิเมชั่นได้ด้วยเช่นกัน เช่น ค่าสี และค่าความโปร่งใส เป็นต้น

- **Envelope**

ในQuest3D ส่วนใหญ่แล้ว อนิเมชัน จะหมายถึง การเปลี่ยนแปลงค่าต่างๆเมื่อเวลาผ่านไปตามปกติแล้ว Value จะสามารถเก็บค่าได้เพียงหนึ่งค่าเท่านั้น ส่วนการควบคุมของEnvelope จะดูเหมือนกับแผนภูมิทางคณิตศาสตร์ แกนนอนหมายถึง ข้อมูลเข้า โดยจะเชื่อมต่อเข้ากับ envelope โดยการลิงค์ แกนตั้งหมายถึง ข้อมูลออก

- **Timer**

Timer Value และ Timer Command Channel ใช้สำหรับควบคุมการทำอนิเมชันแบบง่ายๆ เช่น Timer Command จะมีคำสั่งPlay,StopและPlay&Loop ในภาพข้างล่าง Timer Value Channel ถูกลิงค์เป็นข้อมูลเข้าให้กับEnvelope Channel สำหรับ Timer ก็สามารถแสดงได้โดยใช้ Timer Command Channel ซึ่งสามารถอยู่ในภาพเดียวกัน



**รูปที่ 2.9** แสดงฟังก์ชันของTimer Value

Logic ในQuest3Dนั้น ในการทำโปรแกรมก็คือการเขียนFlowchart ของโปรแกรมLogic จะเป็นตัวอธิบายว่าจะมีสิ่งใดเกิดขึ้นและจะเกิดขึ้นเมื่อไหร่ในโปรแกรมChannelหลายๆอันของQuest3D ได้ถูกออกแบบมาโดยเฉพาะสำหรับการเพิ่มLogicเข้าไปในโปรแกรม ซึ่งมีดังนี้

- **If**

คล้ายกับChannel Callerตรงที่ If Channel จะเป็นตัวเรียกChannelที่ลิงค์เข้ามา โดยมีกาเพิ่มเงื่อนไขเข้าไปในลิงค์แรก โดยเงื่อนไขนั้นจะต้องเป็นChannelที่มีประเภทเป็นvalueเท่านั้น เมื่อค่าของvalueไม่เท่ากับ0(เงื่อนไขเป็นจริง)Channel ที่ถูกเชื่อมต่อกับIf Channelจะถูกเรียกใช้งาน

- **If Else**

If Else Channelมีลักษณะเหมือนกันกับ If Channel แต่จะเพิ่มการกระทำในกรณีที่เงื่อนไขไม่ถูกต้อง(value มีค่าเป็น0) Channel จะทำการเรียกลิงค์ที่สามมาใช้งาน ถ้าเงื่อนไขที่กำหนดเป็นจริงก็จะทำการสลับไปเรียกChannelที่ต่อกับลิงค์ที่สองมาใช้งานแทน

- **Trigger**

Trigger Channel มีการทำงานคล้ายกันกับIf Channel นั่นคือ Trigger จะทำการเรียกใช้งานChannelถูกตามเงื่อนไขที่กำหนด

สิ่งที่แตกต่างกันระหว่าง If และTrigger Channel มีอยู่สองประเด็นหลักๆคือ

1. เมื่อเงื่อนไขเป็นจริง Trigger จะกระทำกับ Channel ถูกเพียงครั้งเดียวเท่านั้น ในการที่จะให้Trigger ทำงานอีกครั้ง จะต้องให้เป็นเงื่อนไขเปลี่ยนเป็นfalseก่อน แล้วเปลี่ยนเป็นtrueอีกครั้ง ยกตัวอย่างเช่น การกำหนดค่าของTrigger โดยการกดปุ่ม Space bar เมื่อกดปุ่มเพื่อให้ Trigger ทำตามเงื่อนไขแล้ว หากต้องการทำอีกครั้ง จำเป็นต้องกดปุ่ม Space bar ก่อนหนึ่งครั้งเพื่อเปลี่ยนสถานะของTrigger แล้วกดปุ่มSpace barอีกครั้งเพื่อทำตามเงื่อนไข

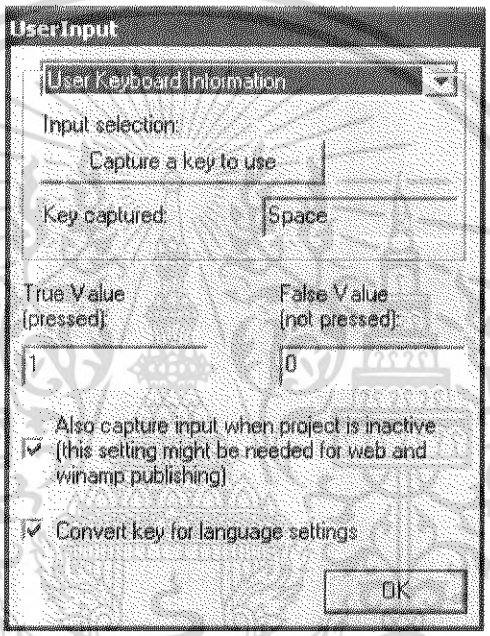
2. สามารถกำหนดตัวเลือกอื่นๆจากรายการในหน้าต่างpropertiesได้ด้วย

- **User Input**

User Input Channel สามารถใช้ในการกำหนดอุปกรณ์รับสัญญาณหลายๆชนิดในการกำหนดค่าให้กับโปรแกรม เช่น เมาส์ หรือ แป้นพิมพ์เป็นต้น สำหรับในQuest3Dนั้น ค่าที่รับมาจากอุปกรณ์เหล่านี้มีอยู่ด้วยกันสามรูปแบบ คือ

1. ค่าไบนารีของ0และ1 เช่น การกดปุ่มบนแป้นพิมพ์หรือเมาส์
2. ค่าแบบ analog จาก0ถึง1.0 เช่น การเคลื่อนที่ของจอยสติ๊กส์
3. ค่าพิกัดของหน้าจอ เช่น การเคลื่อนไหวของเมาส์เคอร์เซอร์

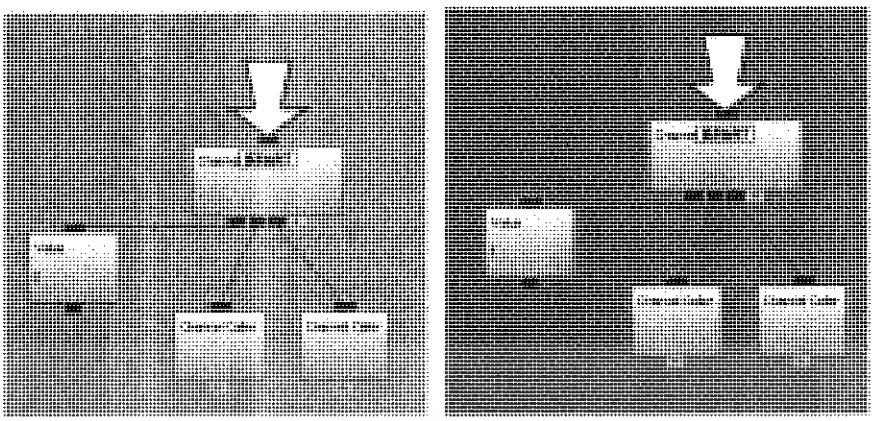
โดยที่ User Input ทั้งหมดสามารถนำมาเป็นเงื่อนไขได้ทั้งสำหรับ If(Else) และ Trigger Channel ในการกำหนดการกดปุ่มบนแป้นพิมพ์ สามารถกำหนดผ่านหน้าต่างPropertiesของ User Input หลังจากที่คลิกปุ่ม Capture a key to use ให้กดปุ่มที่ต้องการกำหนด สำหรับอุปกรณ์อื่น เช่น เมาส์ หรือ จอยสติ๊ก ก็ใช้วิธีการเดียวกัน




**รูปที่ 2.10** แสดงฟังก์ชันของUser Input

- **Channel Switch**

Channel Switch ใช้สำหรับกำหนดเงื่อนไขโดยอนุญาตให้สามารถกำหนดชนิดของข้อมูลที่จะมาลิงก์เข้ากับChannel Switch ได้ โดยจำเป็นต้องกำหนดชนิดของข้อมูลก่อนที่จะมีการทำงาน จากนั้น โหนดลูกอันแรกจะเป็นตัวกำหนดการใช้งานของโหนดลูก ในลำดับต่างๆ ค่า0หมายถึง โหนดลูกที่เป็นตัวดำเนินการแรก ,1เป็นตัวดำเนินการที่สอง



**รูปที่ 2.11** แสดงการทำงานของChannel Switch

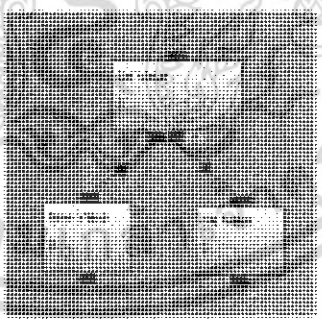
Value เป็นหัวใจหลักของโปรเจคต่างๆในQuest3D ซึ่งใช้สำหรับการกำหนดค่าสีของวัตถุ ตำแหน่งและขนาด หรือ ใช้ในการบอกคะแนน เป็นต้น

- **Set Value**

  รูปแบบคำสั่งทั่วไปในการกำหนดค่าตัวแปรเป็นดังนี้

  Old value = New Value

ใน Quest3D ประโยคข้างต้นจะหมายความว่า ให้คัดลอกค่าของNew value มาใส่ในOld value ในการคัดลอกค่าหนึ่งๆให้กับตัวแปรใดๆ เราจะใช้ Set Value Channel เป็นตัวกำหนด

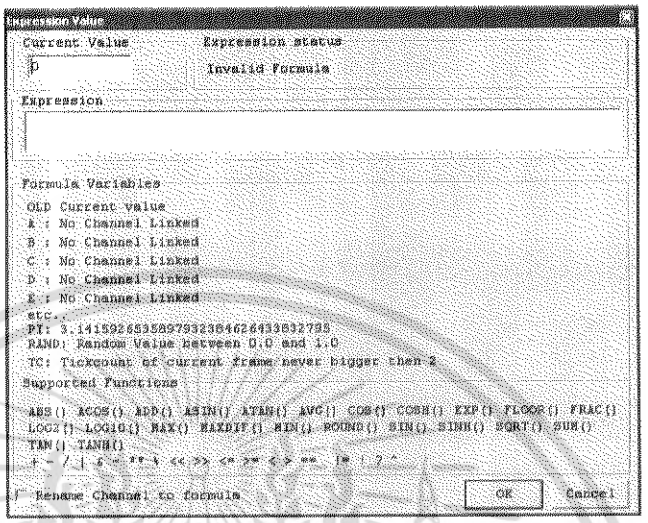


**รูปที่ 2.12** แสดงการทำงานของSet Value

- **Expression Value**

ในการกำหนดสมการทางคณิตศาสตร์ในQuest3Dนั้น สามารถทำได้โดยใช้Expression

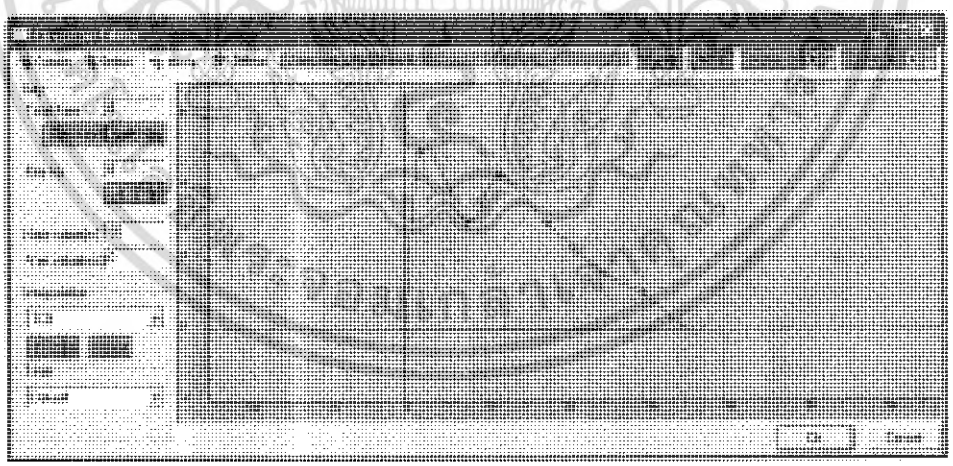
Value Channel ในหน้าต่าง propertiesของ Channel นี้จะมีช่องให้กำหนดสูตรและมีข้อความอธิบายตัวแปรและตัวดำเนินการต่างๆ




**รูปที่ 2.13** แสดงฟังก์ชันของExpression Value

- **Envelope**

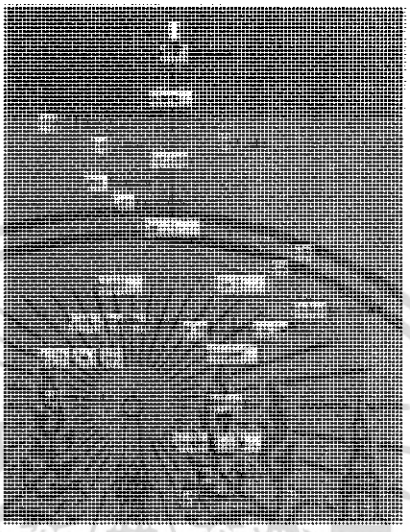
กราฟใน Envelope Channel เป็นแบบสองมิติ ใช้สำหรับการกำหนดสูตรสามารถทำได้โดยการเพิ่มจำนวนของพิกัดเข้าไปในกราฟ



**รูปที่ 2.14** แสดงการทำงานของ Envelope

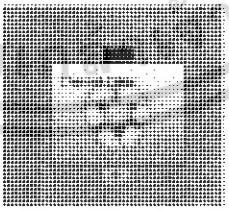
- **For Loop**

ในQuest3Dจะมีฟังก์ชันที่เรียกว่า For Loop ซึ่งมีลักษณะเหมือนกันกับในโปรแกรมภาษาอื่นๆ For Loop สามารถทำการประมวลบางส่วนของโปรแกรมซ้ำหลายๆครั้งได้ในแต่ละเฟรม



**รูปที่ 2.15** แสดงโหนดของ For Loop

ตามปกติ Quest3D จะทำการอัพเดต Channel เพียงหนึ่งครั้งต่อหนึ่งเฟรม เหตุผลก็เนื่องจากต้องรักษาประสิทธิภาพในการประมวลผล แต่สำหรับ Channel ที่อยู่ในFor Loop จะมีการประมวลผลหลายครั้งต่อเฟรม โดยการกำหนดให้กับแต่ละ Channel ให้เป็นContinuous โดยคำสั่งนี้จะอยู่ในหน้าต่าง General Properties ของแต่ละ Channel Channelที่ถูกกำหนดให้เป็น Continuous จะมีสัญลักษณ์ดังภาพข้างล่าง



**รูปที่ 2.16** แสดงรูปแบบContinuous ของวัตถุ

ค่า Max Iteration ของ For Loop Channel เป็นตัวกำหนดจำนวนครั้งทั้งหมดของโปรแกรมย่อยที่กำหนดในFor Loopสำหรับค่า Current Iteration จะใช้ในการสร้างความหลากหลายให้กับ Instance ใน For Loop

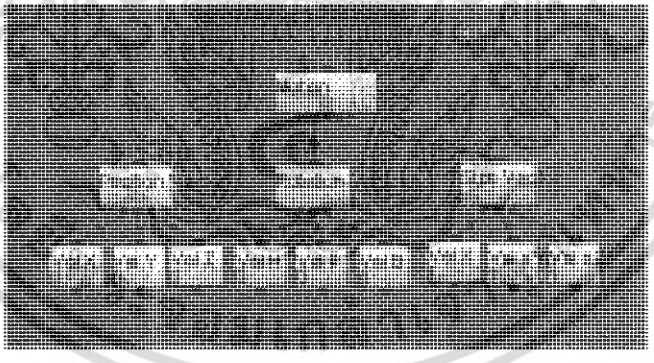
- **Rendering scene**

ในการทำงานจะเป็นการดีถ้าหากเราพยายามลดจำนวนของChannel ที่อยู่ในFor Loop ให้มีจำนวนน้อยที่สุดเท่าที่จะเป็นไปได้ ยกตัวอย่างเช่น การประมวลผลภาพ3มิติ ซึ่งประกอบไปด้วย กล้อง ,แสง และ วัตถุ3มิติ จะดีกว่าถ้าเราจะกำหนดให้มีการทำ For Loop เฉพาะกับวัตถุ แทนที่จะให้ทำ For Loop ทั้งหมด

- **Mathematical Operator**

วัตถุ3มิติ สามารถที่จะย้าย หมุน และ ปรับขนาดได้โดยการใช้พื้นฐานจากพีชคณิตเชิงเส้น ซึ่งมีดังนี้

Matrix and Motion จะทำการเก็บค่า ตำแหน่ง,การหมุน และ ขนาด อยู่ใน Matrix หรือ Motion Channel ในMotion Channel จะแบ่งออกเป็น Vector Channel 3อัน และแต่ละ Vector ก็จะแบ่งออกเป็น 3 Values



รูปที่ 2.17 แสดงโหนดของMatrix

- **Operator**

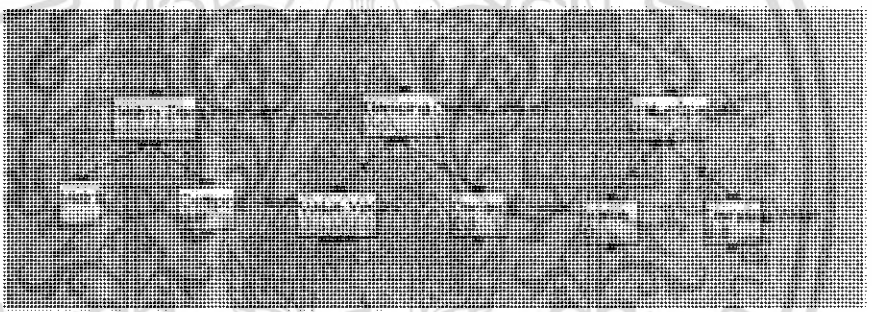
Operator Channel คือ Channel ที่สามารถปรับเปลี่ยนค่าได้ในขณะรัน โดยหลักการทั่วไป Operator Channel จะถูกเรียงลำดับโดยชนิดของChannelของผลลัพธ์ ตัวอย่างเช่น ถ้าผลลัพธ์ของOperator เป็น Value ก็จะสามารถค้นหาได้จากใน Value Operator Base Channel ฟังก์ชันส่วนใหญ่

ของ Value Operator Base Channel จะใช้ในการเข้าถึงข้อมูลจากChannelชนิดอื่นๆ ตัวอย่างเช่น Value Operator ที่ชื่อว่า Get Distance(Vector,Vector)

Vector Operator มักจะใช้ในการ Transform Vector ในการเคลื่อนย้ายตำแหน่ง ประกอบไปด้วยพื้นฐาน การบวก การลบ การคูณ และ การหาร ของVector 2ค่า

Matrix Operator อาจนำไปใช้ในการเปลี่ยนจาก Vector มาเป็น Matrix เช่น Create Transition Matrix  Matrix Operator สามารถใช้ในการรับข้อมูลเช่น Get Current Camera Matrix

Operator ชนิดพิเศษที่เรียกว่า Damping จะมีทั้งชนิดที่ใช้กับค่า Value,Vectorและ Matrix ใช้ในการหน่วงเวลาในการเปลี่ยนค่าตัวแปร มีผลในการเปลี่ยนค่าจากค่าหนึ่งไปเป็นอีกค่าหนึ่ง สามารถปรับค่าความเร็วในการหน่วงได้โดยการเปลี่ยนค่า Value ในลิงค์ที่สอง ค่าความหน่วงจะมีค่าตั้งแต่0ถึง1.0

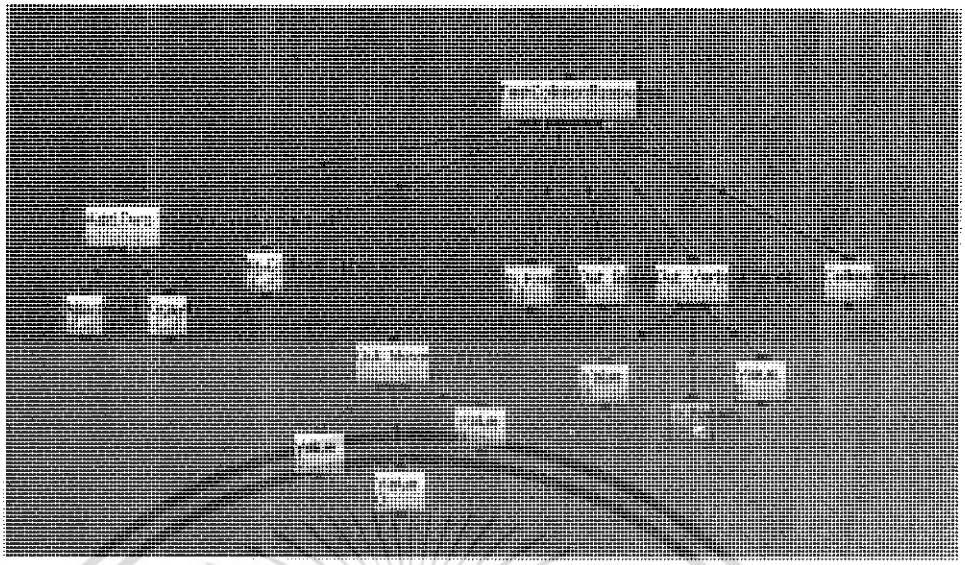


**รูปที่ 2.18** แสดงรูปแบบOperatorแต่ละประเภท

- **Collision Response**

Fast Collision Channel เป็นระบบการจำลองการชนกันระหว่างวัตถุ โดยการตรวจสอบรัศมีของตำแหน่ง ปัจจุบันกับจำนวนของ Collision Objects

โครงสร้างของ Fast Collision Response Channel มีลักษณะดังภาพตัวอย่างข้างล่าง



รูปที่ 2.19 แสดงโหนดของ Fast Collision Channel

- **Graphic user interface**

ส่วนใหญ่แล้ว Graphic user interfaceจะเป็นเลเยอร์2มิติซ้อนอยู่ด้านบนสุดของscene โดยมีลูกศรชี้ที่ปุ่มต่างๆ และข้อมูลเข้า เป็นส่วนประกอบ

- **Z buffer**

ในแต่ละเฟรม เมื่อวัตถุ3มิติ ถูกประมวลผลภาพ3มิติ จึงจะตามมาด้วย GUIและHUD เลเยอร์ ในระหว่างที่ Z buffer จะถูกลบเพื่อป้องกันวัตถุ3มิติหนึ่งซ้อนทับวัตถุอีกตัวหนึ่ง

- **Mouse input**

เมาส์จะตอบโต้กับGUI โดยการใช้ Mouse Overและ Mouse Input Channel

- **Lighting and shadow**

แสงเป็นสิ่งสำคัญที่ขาดไม่ได้สำหรับการสร้างโปรเจค ซึ่งถ้าขาดแสงและเงาแล้วจะทำให้ภาพที่ออกมาดูแข็งกระด้างและแบนราบดูไม่สมจริง การจัดแสงโดยใช้วัตถุกำเนิดแสงนั้น จะต้องมีตัวกำเนิดแสงสร้างแสงไปกระทบกับวัตถุสามมิติที่อยู่ในฉาก ในQuest3D แสงก็เป็น Channelหนึ่ง และสามารถกำหนดคุณสมบัติได้จาก Animation Section

แสงในQuest3D มีอยู่ด้วยกันสามชนิดคือ point,spot และ directional

Point light จะกระจายแสงไปกระทบกับวัตถุที่อยู่รอบตัวมัน

Spot light จะถูกกำหนดขอบเขตของแสงด้วยองศาของกรวยด้านนอกและด้านใน ซึ่งทำให้ผลที่ได้จะเป็นลักษณะแบบ fall-off effect

Directional light เป็นแสงที่ขนานที่มีทิศทางเพียงทิศทางเดียวตามทิศทางของลูกศรของวัตถุกำเนิดแสง

Shadow เงาสำหรับวัตถุที่มีการเคลื่อนที่ เช่น รถ หรือตัวละครที่เดินได้ สามารถจำลองได้โดยใช้ texture square Quest3D รองรับการสร้างแสงเงา โดยใช้ Stencil Shadow Channel เพื่อให้คำนวณการเกิดเงาแบบพิเศษ ในขณะประมวลผลภาพ3มิติ Stencil Shadow Channel ต้องการตำแหน่งของแหล่งกำเนิดแสงในการทำงาน วัตถุสามมิติจะทำการเชื่อมต่อเข้ากับ Stencil Shadow ด้วย SoftwareStencil ShadowObject

- **Sound and music**

เสียงสามารถถูกบันทึกไว้ในรูปแบบของดิจิตอลในคอมพิวเตอร์ได้ในหลายๆรูปแบบ Quest3Dรองรับรูปแบบของไฟล์ได้ทั้งแบบ .wavและ.mp3

- **Sound File Channel**

Sound File Channel ถูกใช้ในการเก็บ .wav ไว้ใน Quest3D โดยมีหน้าต่างProperties ที่มีตัวเลือกหลายๆอย่าง เช่น load,playและpause เป็นต้น ความเร็ว มีค่าตั้งแต่0%ถึง200% และ ระดับเสียงมีค่าตั้งแต่ 0%ถึง100% ทั้งสองสามารถปรับเปลี่ยนผ่านโหนดลูก ได้

- **Sound Command**

Sound Command Channel ใช้สำหรับการควบคุมทุกๆอย่างของตัวอย่างเสียง เช่น การปรับระดับเสียง อีกทั้งยังสามารถใช้โหลดไฟล์.wavจากฮาร์ดดิสก์ได้ Sound Command จะถูกเรียกใช้เพียงครั้งเดียวเมื่อมีการทำงานของโปรแกรม และจะมีผลกับทุกๆโหนดลูกของSound File Channel

- **3D Positioning**

นอกจากการเล่นเสียงแล้ว ยังสามารถกำหนดตำแหน่งของเสียงในพื้นที่3มิติได้เช่นกัน

3D Posiotion sound จำเป็นต้องใช้ Listener Channel สำหรับการเชื่อมต่อเข้ากับตำแหน่งของกล้อง และทำการปรับความถี่ ให้โดยอัตโนมัติโดยการคำนวณตำแหน่งของเสียงให้สัมพันธ์กับระยะทาง เฉพาะ Mono sound samping เท่านั้นที่สามารถจะกำหนดตำแหน่งได้

- **.MP3 format**

Quest3D สามารถรองรับไฟล์.mp3 formatได้ โดยใช้ MP3 File Channel ในการเก็บตัวอย่างเสียงชนิดนี้ โดยผ่านทางหน้าต่างProperties ในการโหลดไฟล์สำหรับ Save File in Channelgroup option เป็นตัวเลือกในการเก็บตัวอย่างเวียงไว้ในตัวของChannel ซึ่งจะทำให้ไฟล์มีขนาดใหญ่ MP3 File มี โหนดลูก เพียงหนึ่งตัวเท่านั้น ใช้สำหรับเก็บชื่อไฟล์ในรูปแบบของข้อความ MP3 Control Channel สามารถใช้ควบคุมการปรับระดับเสียง และใช้ในการรับค่าความยาวไฟล์เสียง และตำแหน่งปัจจุบันที่ไฟล์เสียงทำงานและการข้ามไปยังตำแหน่งที่ระบุไว้ในsample MP3 ไม่สามารถกำหนดตำแหน่งได้

## 2.2 Maya6.5 Unlimited Software

### 2.2.1 ข้อจำกัดในการสร้างตัวละครหรือฉากสามมิติในโปรแกรม Maya

ในการทำโมเดลสามมิติสำหรับเกมนั้นจะมีความแตกต่างกับการทำโมเดลสำหรับใช้ในหนังหรือภาพยนต์เป็นอย่างมาก อันเนื่องมาจากการแสดงผลในเกมเป็นการแสดงผลเชิงเวกเตอร์ ภาพที่เราเห็นขณะเล่นเกมนั้นเป็นภาพที่ถูกสร้างขึ้นใหม่และต่อเนื่องอยู่ตลอด เพื่อตอบสนองในแต่ละท่าทางที่เปลี่ยนไปอย่างรวดเร็วขณะเล่น จึงทำให้โมเดลที่จะใช้สำหรับเกมจะต้องมีขนาด polygon ที่ไม่มากจนเกินไป ถ้าหาก polygon ของโมเดลมีขนาดที่มากเกินไป ก็จะทำให้คอมพิวเตอร์ไม่สามารถประมวลภาพที่จะทำการแสดงผลต่อไปได้ทันตามอัตราการตอบสนองต่อผู้เล่นที่ควรจะเป็น ทำให้ผู้เล่นอาจประสบปัญหาอาการกระตุกหรือเกิดความรู้สึกไม่ต่อเนื่องในขณะเล่น ทั้งนี้เราต้องพิจารณาอีกด้วยว่าเกมที่เราสร้างนั้นจะเลือกสมรรถนะขั้นต่ำของคอมพิวเตอร์ที่จะใช้งานเป็นเท่าใด โดยเฉพาะที่เราต้องพิจารณาเป็นปัจจัยหลักคือกราฟฟิกการ์ด ซึ่งจะมีสมรรถนะในการแสดงผลจำนวน polygon ต่อหนึ่งหน่วยเวลาที่จะทำได้แตกต่างกันออกไป หากสามารถแสดงจำนวน polygon ได้มากก็หมายความว่าเราสามารถออกแบบตัวโมเดลให้มีความละเอียดได้มากขึ้นตามไปด้วย

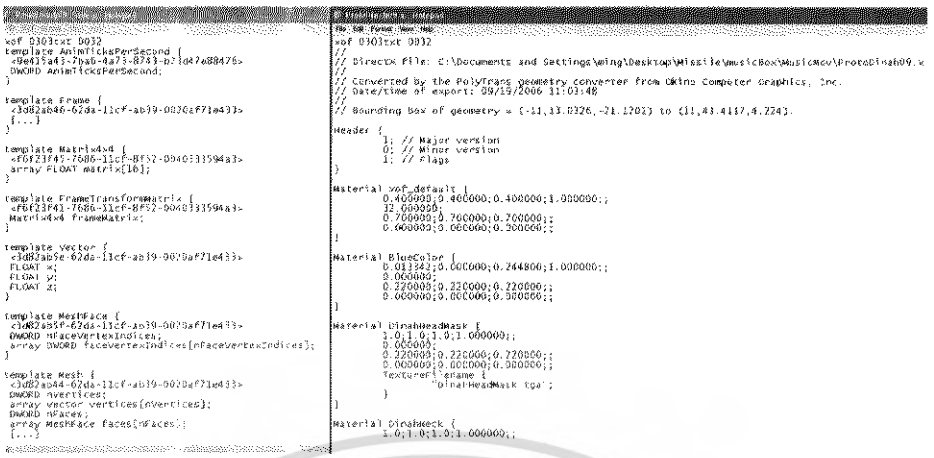
ตัว Maya เองก็มีข้อจำกัดในการทำงานสร้างโมเดลสามมิติสำหรับเกมเช่นกัน อันเนื่องมาจากว่าไฟล์โมเดลสามมิติที่ใช้กับเอนจิน นั้นเป็นประเภท DirectX (x) ซึ่งมีข้อจำกัดในการเก็บข้อมูลเครื่องมือบางอย่างใน Maya ที่เราใช้ในการสร้างและกำหนด attribute ของโมเดล ไม่สามารถที่จะ

export เพื่อบันทึกลงไปยัง X file ได้ เพราะโครงสร้างของ X file จะไม่รองรับข้อมูลของเครื่องมือประเภทนั้นๆใน Maya เราจึงต้องมีการทราบข้อกำหนดในการใช้เครื่องมือต่างๆที่มีอยู่ใน Maya เพื่อที่จะทำงานกับโมเดลสำหรับเกมสามมิติ ในส่วนของ DirectX (X) ไฟล์นั้นจะเก็บข้อมูลในส่วนขององค์ประกอบโมเดล อาทิ จุด(vertices) เส้น(edge) แกนนอร์มอล(normal) การเคลื่อน(translate) การหมุน(rotate) การลดขยายขนาด(scale) กระดูก(skeleton) การเชื่อมกระดูกกับผิว(binding) น้ำหนักระหว่างกระดูกกับผิว(skin weight) ตำแหน่งการปะเท็กซ์เชอร์(texture placement)

จุดยากของการทำโมเดลสามมิติจะอยู่ที่เราต้องทราบข้อจำกัดที่จะพึงมีได้ทั้งหมด ถ้าหากทำผิดพลาดแม้แต่ประการเดียว ก็อาจทำให้โมเดลที่เราสร้างไม่สามารถถูกอ่านจากเอนจินได้ หรือกระทั่งมีข้อผิดพลาดในกระบวนการ ไม่ว่าจะเป็น รูปร่างของโมเดลผิดไป การเคลื่อนที่ไปผิดจากที่กำหนดไว้ ซึ่งบางครั้งอาจมีปัญหาถึงขั้นที่ทำให้ต้องทำการสร้างโมเดลขึ้นใหม่เลยทีเดียว ข้อจำกัดที่เราต้องทราบนั้นมีอยู่มาก และมีบางตัวที่มีความสำคัญมาก หากไม่ทราบแล้วละก็ คงเป็นการยากที่จะสร้างโมเดลให้มีความสมบูรณ์ได้

***ข้อจำกัดด้านโครงสร้างของ DirectX (X) ไฟล์***

โครงสร้างของ X ไฟล์นั้นไม่ได้มีมาตรฐานเหมือนกันสำหรับแต่ละเอนจิน โครงสร้างของ X ไฟล์นั้นไม่ได้มีมาตรฐานกลาง ขึ้นอยู่กับว่าเอนจินไหนสนับสนุนโครงสร้างแบบใด นั่นเป็นเพราะว่า X ไฟล์มีลักษณะไม่ต่างจากไฟล์ text ธรรมดา (สามารถเปิดดูด้วย notepad ได้) โดยภายในนั้นมีการบันทึกข้อมูลของจุดและการเคลื่อนที่ต่างๆ ลงไปเก็บแยกเป็น frame ซึ่งบันทึกตำแหน่งการเลื่อน การหมุน การย่อขยาย หรือแม้กระทั่งข้อมูลของกระดูก แต่ว่าข้อมูลเหล่านี้ภายใน X ไฟล์นั้นจะมีการจัดเรียงโครงสร้างที่แตกต่างกันออกไป ขึ้นอยู่กับว่า X ไฟล์นั้นถูกบันทึกโดยโปรแกรมใด โครงสร้างก็จะมีมาตรฐานตามที่โปรแกรมนั้นทำการบันทึก เราต้องตรวจสอบให้ดีก่อนว่าเอนจินที่เราใช้นั้น มีความสามารถในการที่จะรองรับโครงสร้างของ X ไฟล์ที่เราจะทำการใช้ได้หรือไม่



**รูปที่ 2.20** แสดงการเปรียบความแตกต่างด้านโครงสร้าง directX ไฟล์

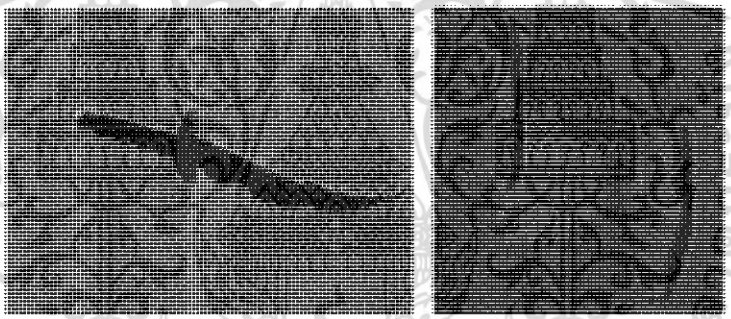
ภาพสองภาพนี้เป็น X ไฟล์ที่ให้ผลลัพธ์ในการแสดงตัวละครสามมิติที่เป็นตัวเดียวกัน ทางด้านซ้ายเป็น X ไฟล์ที่ได้จากการ export ออกมาจากโปรแกรม maya โดยตรง ส่วนภาพทางด้านขวาเป็นไฟล์เดียวกันเพียงแต่นำมาบันทึกใหม่ในโปรแกรม Mesh Viewer ซึ่งจะเห็นได้ว่าทั้งๆที่เป็นไฟล์เดียวกัน แต่กลับพบว่าโครงสร้างในการเก็บข้อมูลภายใน X ไฟล์นั้นมีลักษณะต่างกัน ดังที่ได้กล่าวมาด้านบน เราต้องตรวจสอบว่า X ไฟล์ที่เราจะใช้นั้นถูกรองรับด้วยเอนจินหรือไม่ ถ้าหากไม่ได้มีการรองรับ การแสดงผลไฟล์ที่เกิดขึ้นในเอนจินนั้นจะเกิดข้อผิดพลาดขึ้น

**รูปที่ 2.21** แสดงโมเดลที่มีความสมบูรณ์(ภาพซ้าย) และแสดงโมเดลที่มีความผิดพลาด(ภาพขวา)

ภาพทางด้านซ้ายนั้นเป็นภาพการแสดงผลโมเดลสามมิติในโปรแกรม quest ซึ่งสามารถทำได้อย่างสมบูรณ์ ในขณะที่ภาพทางขวาเป็นโมเดลไฟล์เดียวกันที่นำไปแสดงผลยัง DirectX Viewer SDK April2006ซึ่งพบข้อผิดพลาด จะเห็นได้ว่ามีจุดบางส่วนแถวด้านหลังที่ไม่ได้ขยับตามการเคลื่อนที่ ทำให้เกิดแถบสีดำขึ้นดังภาพ

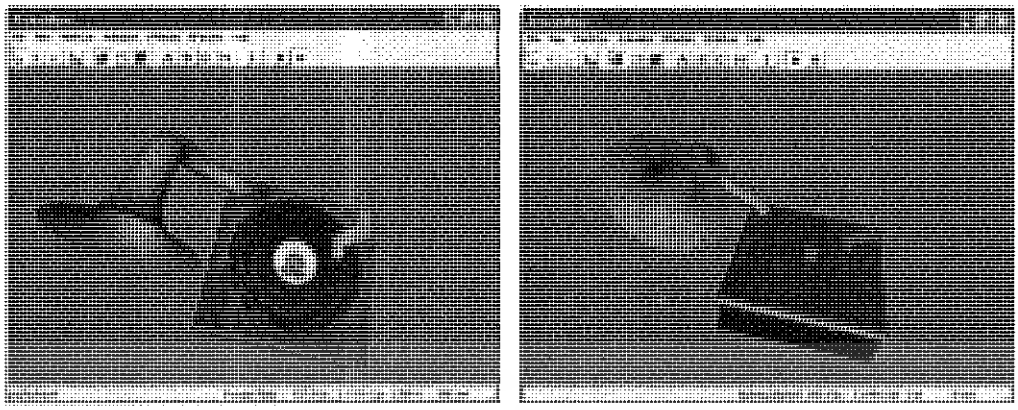
ข้อผิดพลาดที่เกิดขึ้นนี้ยากที่จะแก้ไข ดังนั้นควรตรวจสอบให้แน่ใจก่อน การทดสอบทำได้โดยลองการนำเข้าโมเดลที่ได้สร้างขึ้นที่มีความหลากหลายเข้าไปตรวจสอบการแสดงผลว่ามีข้อผิดพลาดไหม โดยปกติแล้วข้อผิดพลาดส่วนใหญ่ที่จะเกิดขึ้น จะอยู่ในส่วนของโมเดลที่มีการเคลื่อนที่หรือการใช้กระดูก แต่ถึงอย่างนั้นโมเดลที่ไม่ได้มีการเคลื่อนที่ก็มีข้อผิดพลาดได้เช่นกัน

- กรณีที่เป็นโมเดลที่ไม่มีการเคลื่อนที่ ส่วนใหญ่โมเดลชนิดนี้จะเกิดข้อผิดพลาดขึ้น เช่น พื้นผิวของวัตถุไม่แสดงสี(เป็นสีดำ) ตำแหน่งของส่วนประกอบของโมเดลอยู่ผิดตำแหน่งไม่ว่าจะเป็นจุดของโมเดลหรือจะเป็นชิ้นส่วนของโมเดล มีส่วนของโมเดลที่ไม่แสดงผลหรือไม่สามารถโหลดได้



**รูปที่ 2.22** แสดงให้เห็นถึงวัตถุที่แสดงสีไม่ขึ้น(ภาพซ้าย) และแสดงให้เห็นถึงส่วนของวัตถุที่หายไป (ภาพขวา)

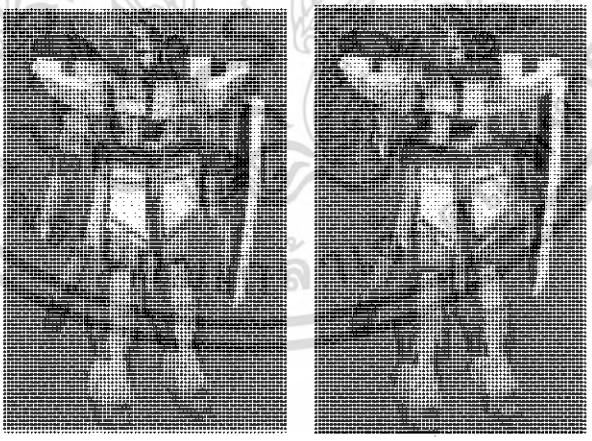
- กรณีที่เป็นโมเดลที่มีการเคลื่อนที่ นั้นโมเดลอาจไม่สามารถเคลื่อนที่ได้หรือกระทั่งเคลื่อนที่แค่เพียงบางส่วนแต่บางส่วนไม่ยอมเคลื่อนตาม หรือในขณะทำการเคลื่อนที่อาจมีการเคลื่อนที่ผิดตำแหน่งและไม่พร้อมเพียงอย่างที่ควรจะเป็น



**รูปที่ 2.23** แสดงวัตถุขณะยังไม่มีการเคลื่อนไหว(ภาพซ้าย) และแสดงข้อผิดพลาดหลังทำการเคลื่อนไหว(ภาพขวา) ของเครื่องเล่นแผ่นเสียง

จากภาพทางด้านซ้าย วัตถุซึ่งยังไม่ได้มีการแสดงภาพเคลื่อนไหวนั้นมีองค์ประกอบครบถ้วนดี แต่หลังจากเราทำการสั่งให้วัตถุแสดงการเคลื่อนไหวในภาพขวา ซึ่งในที่นี้ก็คือให้แผ่นหมุน แผ่นกลับหายไป โดยเกิดจากการที่แผ่นเสียงลงไปหมุนอยู่ด้านล่าง ซึ่งซ้อนอยู่ภายในตำแหน่งของฐาน

- กรณีที่เป็นโมเดลที่มีการใช้โครงกระดูก โมเดลนั้นอาจโหลดไม่ขึ้นหรือไม่สามารถแสดงผลได้ กระทั่งไม่สามารถเคลื่อนไหวตามที่สร้างไว้ได้



**รูปที่ 2.24** แสดงวัตถุขณะยังไม่มีการเคลื่อนไหว(ภาพซ้าย) และแสดงข้อผิดพลาดหลังทำการเคลื่อนไหว(ภาพขวา) ของตัวละคร

จากภาพจะเห็นได้ว่าเมื่อให้หุ่นทำการเคลื่อนไหวแล้วเกิดข้อผิดพลาดตรงที่แขนนั้นหดกลับเข้าไป โดยแขนควรจะทำท่าแกว่งตามที่ได้กำหนดไว้ เมื่อเอนจินไม่สามารถรองรับ X ไฟล์ที่เราใช้ได้นั้น เรามีทางเลือกอยู่สามทางคือ ทำการหาโปรแกรมที่สามารถช่วยในการแปลงโครงสร้างของ X ไฟล์ที่เราใช้อยู่ให้กลายเป็นโครงสร้างที่เอนจินรองรับ หรือเปลี่ยนไปใช้โปรแกรมสร้างโมเดลสามมิติที่มีเครื่องมือที่สนับสนุนการ export X ไฟล์ที่มีโครงสร้างตรงกับเอนจิน หรือไม่ก็ต้องเปลี่ยนเอนจินมาเป็นตัวที่สนับสนุนการทำงานของ X ไฟล์ที่เราใช้แทน ถ้าเราสามารถแก้ปัญหาจุดนี้ไปได้ตั้งแต่ต้น จะทำให้เราไม่ต้องกลับมาเริ่มต้นทำใหม่หลายครั้ง

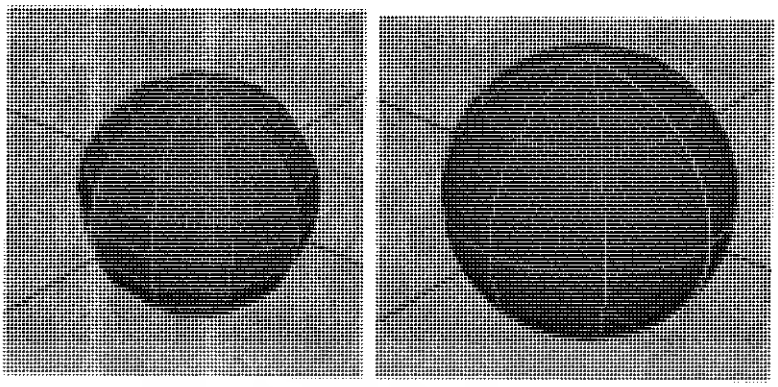
สำหรับการ export ที่ใช้ในโปรแกรม Maya นั้นเราจะใช้ตัว Okino.Polytrans.v4.2.1 เป็นตัว exporter และจะใช้โปรแกรม Mesh Viewer 9.8.299.0 ของบริษัท Microsoft Corporation เป็นตัวแปลงไฟล์อีกทีเพื่อให้ไฟล์สามารถใช้กับโปรแกรม Quest3D ได้

***ข้อจำกัดในด้านการสร้างโมเดลสามมิติ***

จุดประสงค์ที่ใช้ทำงานจริงของโปรแกรม maya นั้นเกิดมาจากความต้องการที่จะสร้างวัตถุสามมิติในการทำไฟล์ภาพยนตร์ ฉะนั้นเครื่องมือต่างๆที่มีอยู่ในโปรแกรมนั้นจึงสามารถใช้ได้เพียงแค่บางส่วนเท่านั้นในการทำโมเดลสำหรับเกม เพราะบางเครื่องมือนั้นอาศัยการประมวลผลเฉพาะในโปรแกรม maya เท่านั้น ไม่สามารถนำไปบันทึกลงใน Direct X (X) ไฟล์ได้ เราจึงต้องทราบขอบเขตการใช้เครื่องมือว่า ต้องใช้เครื่องมือใดจึงจะสามารถ export ออกไปได้

- **ใช้เฉพาะการสร้างโมเดลแบบ polygon เท่านั้นสำหรับการทำงาน**

โปรแกรม maya นั้นจะมีรูปแบบการสร้างโมเดลหลักอยู่สองรูปแบบ คือ การสร้างโมเดลโดยใช้polygon และการสร้างโมเดลโดยใช้ nurbs ซึ่งการสร้างวัตถุเพื่อที่จะ export ไปเป็น X ไฟล์นั้นจะอนุมัติแค่การสร้างแบบ polygon เท่านั้น ทั้งนี้เพราะ nurbs ซึ่งเป็นเส้นโค้งนั้นอาศัย maya software ในการคำนวณรูปร่างโดยตรงจึงไม่สามารถบันทึกลงไปใน X ไฟล์ได้



รูปที่ 2.25 โมเดล polygon(ภาพซ้าย) และโมเดล nurbs(ภาพขวา)

- **ไม่ควรสร้างโมเดลแบบ nurbs แล้วมาแปลงเป็น polygon**

โปรแกรม maya นั้นมีเครื่องมือที่สามารถแปลงโมเดลจากชนิด nurbs ไปเป็นชนิด polygon ได้ หากแต่ว่าโมเดลที่ได้จากการแปลงนั้นจะควบคุมจำนวน polygon ได้ยาก อีกทั้งยังมีโอกาสเกิดความเหลื่อมของพื้นผิววัตถุได้ง่าย นอกจากนั้นแกน normal ที่เป็นตัวระบุหน้าของพื้นผิวจะไม่มีความต่อเนื่อง หรืออาจอยู่ผิดด้านของพื้นผิว ทำให้ไม่เหมาะสมเป็นอย่างยิ่งที่จะใช้กระบวนการนี้

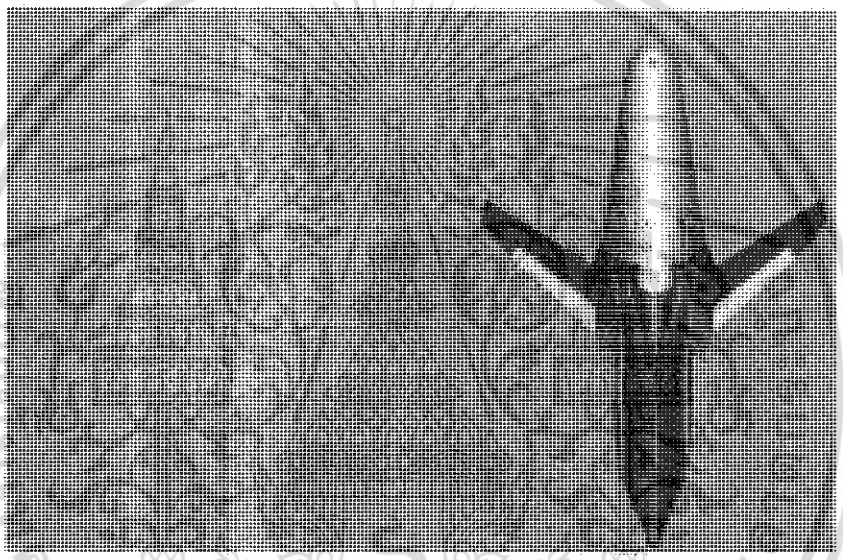
**รูปที่ 2.26** แสดงวัตถุที่เกิดจากการแปลง nurbs มาเป็น polygon

ภาพแรกสุดนั้นเป็นภาพโมเดลที่สร้างขึ้นจาก nurbs เมื่อทำการแปลงมาเป็น polygon ในภาพที่สอง เราจะเห็นได้ว่าโมเดลถูกซอยออกเป็น polygon จำนวนมากซึ่งเป็นการสิ้นเปลือง แล้วเมื่อเราลอง

ขยายขนาดเข้าไปเพื่อดูรายละเอียดก็จะพบว่าผิวของวัตถุนั้นมีการเหลื่อมกันอยู่ การแปลงจาก nurbs เป็น polygon จึงไม่เหมาะสมสำหรับการทำโมเดลเพื่อใช้กับเกม

- **เช็คแกน Normal ของพื้นผิวให้อยู่ในด้านที่ถูกต้อง**

ปกติแล้วพื้นผิวของโมเดลจะมีสิ่งที่เรียกว่าแกน normal ตั้งอยู่บนพื้นผิว แกนนี้จะเป็นตัวที่ไว้ใช้คำนวณแสงของวัตถุ ซึ่งโดยทั่วไปนั้นจะมีอยู่เพียงด้านเดียวของพื้นผิว ฉะนั้นส่วนพื้นผิวที่อยู่คนละด้านกันกับแกนนี้นั้นจะไม่ถูกแสดงผลในการมองเห็นขณะถูกโหลดขึ้นบนเอนจิน เราจึงต้องมั่นใจก่อนว่า แกน normal นี้ได้หันไปอย่างถูกทิศทาง คือต้องตั้งอยู่บนด้านพื้นผิวที่เราต้องการให้ถูกมองเห็น



**รูปที่ 2.27** แสดงให้เห็นถึงแกน Normal ที่จะมีอยู่ทุกๆพื้นผิวของโมเดล โดยจะชี้ออกมาจากพื้นผิว

จะเห็นได้ว่าการที่เรามองเห็นพื้นผิวของวัตถุนั้นก็เพราะว่ามีแกน normal ชี้ออกมายังทิศทางที่เรามองไปยังพื้นผิววัตถุนั้น เพราะฉะนั้นจึงจำเป็นอย่างมากที่เราจะต้องตรวจสอบให้ถี่ถ้วนว่าแกนเหล่านี้ได้ชี้ไปอย่างถูกทิศทางไหม เพื่อที่ว่าโมเดลจะถูกแสดงได้อย่างสมบูรณ์ในขณะที่ถูกโหลดอยู่บนเอนจิน

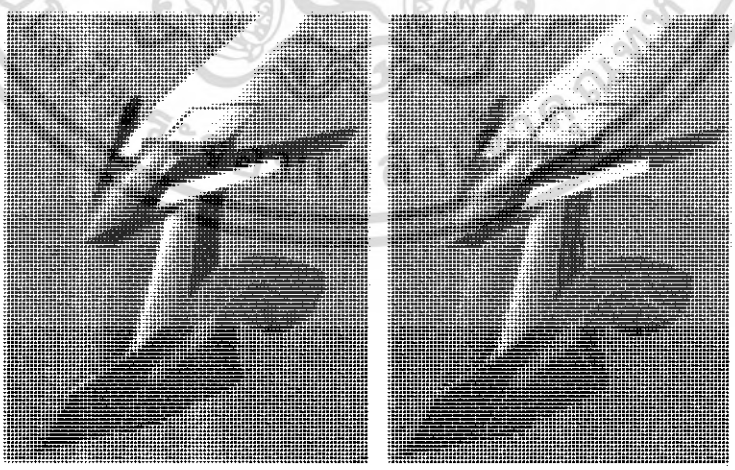
- **เมื่อเราปั้นโมเดลเสร็จในแต่ละส่วน ต้องทำการตรวจสอบข้อผิดพลาดของวัตถุชิ้นนั้นเสียก่อนที่จะปั้นชิ้นต่อไปด้วย clean up tool**

ชิ้นส่วนแต่ละชิ้นที่เราสร้างขึ้นนั้นอาจมีข้อผิดพลาดเกิดขึ้นมากมาย อาทิ non-planar face หรือ polygon ผิวที่เป็นรูปสี่เหลี่ยมแต่จุด(vertice)บางมุมไม่สมมาตรในแนวระนาบ แม้กระทั่งเกิดการซ้อนทับของพื้นผิวหรือเส้นภายในวัตถุชิ้นเดียวกันอย่างพอดี คือ face zero หรือ edge zero ซึ่งอาจจะทำให้เมื่อ export ออกไปแล้วนั้นเอนจิ้นบางตัวจะไม่ยอมอ่าน X ไฟล์ที่มีข้อมูลในลักษณะนี้ จึงควรทำการซ่อมแซมข้อผิดพลาดจำพวกนี้ก่อน โดย maya นั้นมีอุปกรณ์ช่วยแจ้งเตือน หรือช่วยขจัดข้อผิดพลาดที่มีอยู่นั่นก็คือ clean-up tool ซึ่งอยู่ในส่วนของ model ที่ Polygons->Cleanup ให้เราตรวจสอบจนแน่ใจแล้วว่าส่วนนั้นปราศจากซึ่งข้อผิดพลาดแล้ว จากนั้นจึงค่อยเริ่มทำงานในส่วนต่อๆไป จะช่วยให้การทำงานง่ายและมีประสิทธิภาพมากขึ้น

ทั้งนี้นั้นพื้นผิวของวัตถุทุกชนิดควรจะถูกแปลงเป็นพื้นผิวแบบสามเหลี่ยม(Triangular) ทั้งหมดด้วย เพราะพื้นผิวแบบสี่เหลี่ยมก็จะต้องถูกประมวลผลเป็นพื้นผิวแบบสามเหลี่ยม การแปลงเป็นพื้นผิวสามเหลี่ยมตั้งแต่ต้นนั้นจะช่วยลดข้อผิดพลาดของข้อมูลไฟล์ในอนาคตได้

- **วัตถุทุกชิ้นควรถูก combine ไว้ทั้งหมดก่อนที่จะทำการ bind skin ลงบนกระดูก**

เวลาที่เราสร้างโมเดลใน maya นั้น เราอาจต้องทำการสร้างโมเดลขึ้นจากวัตถุสามมิติหลายชิ้น เช่นอาจนำทรงสี่เหลี่ยมมาดัดแล้วนำมาต่อกัน หรือนำแผ่นผิวมาทำการดัดและเชื่อมกัน นั่นทำให้วัตถุในโมเดลนั้นถูกแบ่งออกเป็นหลายๆชิ้น ยามเมื่อเราเสร็จขั้นตอนในการขึ้นรูปโมเดลแล้ว ก็ควรนำองค์ประกอบทุกชิ้นของโมเดลมาทำการรวมให้เป็นชิ้นเดียวด้วยการ combine มิฉะนั้นแล้ว อาจเกิดปัญหาในขณะที่เราจะทำงานกับกระดูกได้ เพราะว่าตำแหน่งของวัตถุใน maya นั้นจะมีโอกาสเกิดข้อผิดพลาดได้สูงยามเมื่อ export ออกไป การรวมวัตถุให้เป็นชิ้นเดียวกันก่อนจะช่วยแก้ปัญหาตรงจุดนี้ด้วย



**รูปที่ 2.28** แสดงการ combine วัตถุ

ภาพทางด้านซ้ายนั้นจะเห็นได้ว่าโมเดลยังสามารถเลือกได้เป็นชิ้นๆอยู่ แต่หลังจากที่เราทำการ combine องค์ประกอบของโมเดลทั้งหมดแล้วนั้น โมเดลจะกลายเป็นวัตถุเพียงชิ้นเดียว เวลาเลือกก็จะ s เลือกเป็นทั้งชิ้น

- **วัตถุทุกชิ้นควรถูก Freeze Transformation ไว้ก่อนที่จะทำการ bind skin ลงบนกระดูก**

วัตถุต่างๆที่เราทำการสร้างในโปรแกรม maya นั้นจะมี ตำแหน่งเป็นของตัววัตถุ เมื่อเราทำการปรับเปลี่ยนรูปร่างหรือคุณลักษณะของวัตถุ อาจทำให้ตำแหน่งของวัตถุถูกเปลี่ยนโยกย้ายตามไปด้วย เมื่อเสร็จสิ้นกระบวนการปั้นทั้งหมด จึงควรใช้คำสั่งในหัวข้อ model คือ modify-> freeze transformation เพื่อตั้งค่าตำแหน่งของวัตถุใหม่ โดยตำแหน่งหรือมุมหมุนหรือขนาดของวัตถุจะถูกตั้งค่าใหม่ให้เป็น 0 ซึ่งมีข้อดีตรงที่ เวลาที่เราจะทำอนิเมชันกับโมเดลนั้น เราจะได้ทราบถึงตำแหน่งเริ่มต้นของมัน นั่นก็คือ 0 แต่เนื่องจากบางเอนจินอาจไม่รองรับการ Freeze Transformation ฉะนั้น การกำหนดขนาดของโมเดลที่จะทำให้มีความถูกต้องในด้านของขนาดตั้งแต่ตอนขึ้นรูปเลย

***ทั้งนี้ยังมีข้อจำกัดที่ควรรู้อยู่อีกมาก ซึ่งหากใครมีความต้องการที่จะทำโมเดลแล้วละก็ ก็ควรที่จะหาข้อมูลด้านข้อจำกัดนั้นเอาไว้ให้ครบเสียก่อน มิเช่นนั้น ปัญหาที่จะเกิดขึ้นในระหว่างการดำเนินงานนั้นอาจจะร้ายแรงถึงขนาดต้องเริ่มต้นใหม่ หรือเลิกล้มไปเลยทีเดียว***

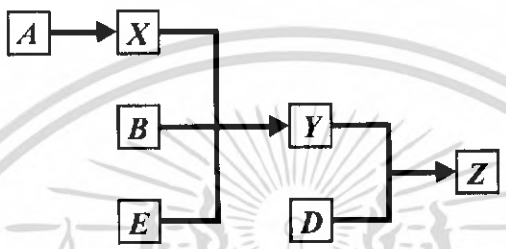
**2.3 ระบบการแสดงความรู้ด้วยกฎ(Rule Base System)**

การใช้กลุ่มของเงื่อนไขซึ่งเก็บสะสมจากการทำงานของหน่วยความจำ และกลุ่มของกฎที่ได้ระบุถึงวิธีการแสดงบนกลุ่มของเงื่อนไข ระบบการแสดงความรู้ด้วยกฎสามารถถูกสร้างขึ้นมาได้ โดยระบบการแสดงความรู้ด้วยกฎจะมีรูปแบบที่ค่อนข้างง่าย ซึ่งจะประกอบไปด้วยกลุ่มข้อความที่เป็นเงื่อนไขหลายเงื่อนไขในรูปแบบของif-then แต่การจัดการขั้นพื้นฐานจะเรียกว่า ระบบผู้เชี่ยวชาญ ซึ่งได้ถูกนำมาใช้อย่างกว้างขวางในหลายสาขา แนวคิดของระบบผู้เชี่ยวชาญ คือ ความรู้ของผู้เชี่ยวชาญที่ได้ถูกเข้ารหัสในกลุ่มของกฎ เมื่อได้รับข้อมูลเดียวกัน ระบบผู้เชี่ยวชาญจะแสดงในรูปแบบกติกาที่คล้ายกันต่อผู้เชี่ยวชาญ

ระบบการแสดงความรู้ด้วยกฎเป็นแบบจำลองง่ายๆที่มีความสัมพันธ์กันที่สามารถที่จะถูกปรับ

ไปเป็นปัญหาใดก็ได้ ด้วยการใช้ปัญญาประดิษฐ์ใดก็ได้ ระบบการแสดงความรู้ด้วยกฎจะมีจุดแข็งที่ดีเหมือนกับการมีขอบเขตที่ต้องถูกพิจารณาก่อนที่จะทำการตัดสินใจ ซึ่งเป็นเทคนิคที่ดีที่ใช้สำหรับการแก้ปัญหา ระบบการแสดงความรู้ด้วยกฎเป็นระบบที่เหมาะสำหรับปัญหาต่างๆเพียงอย่างเดียว ซึ่งจะใช้ความรู้ทั้งหมดในบริเวณพื้นที่ของปัญหาในรูปแบบที่อยู่ในรูปของกฎ If-Then และบริเวณพื้นที่ของปัญหาจะไม่ใหญ่ ถ้ามีกฎต่างๆมากเกินไป ระบบจะทำการซ่อมบำรุงยาก

ตัวอย่าง




**รูปที่ 2.29** ยกตัวอย่างการทำงานของระบบการแสดงความรู้ด้วยกฎ

Rule1: If Y is true
AND D is true
THEN Z is true

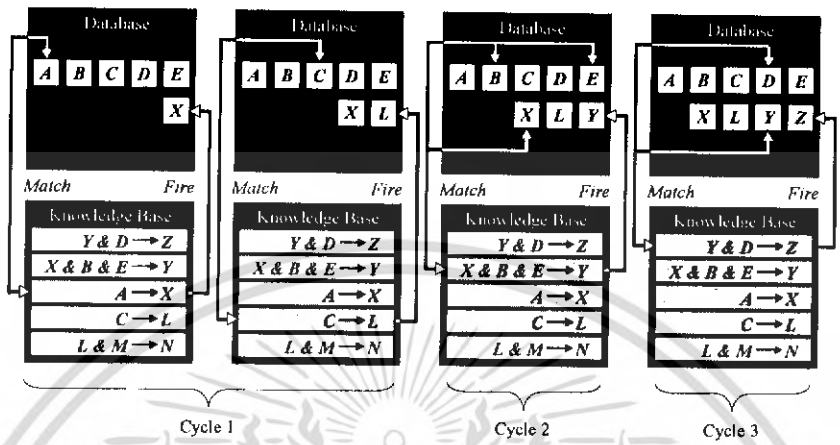
Rule 2: IF X is true
AND B is true
AND E is true
THEN Y is true

Rule 3: IF A is true
THEN X is true

วิธีของระบบการแสดงความรู้ด้วยกฎ(Methods of Rule-Based Systems) มี 2 แบบ คือ

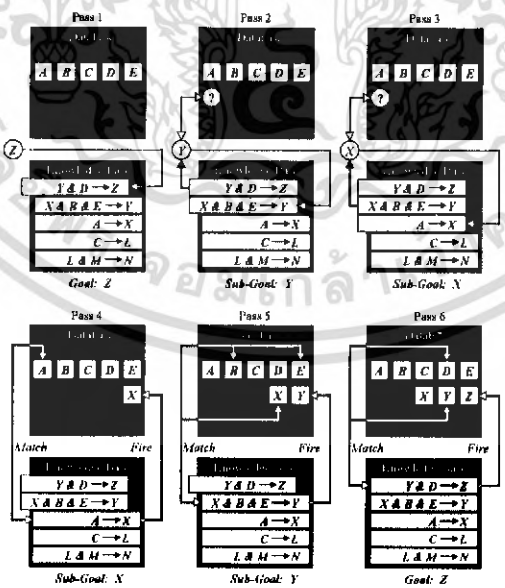
1. **การอนุมานแบบไปข้างหน้า (Forward chaining)** เป็นวิธีการอนุมาน โดยเริ่มการตรวจสอบข้อมูล กับกฎเกณฑ์ที่มีอยู่ในระบบจนกว่าสามารถหากฎเกณฑ์ ที่สอดคล้องกับสถานการณ์แล้วจึงดำเนินงานตามเหมาะสม กฎเกณฑ์นั้นอาจได้มาจากการกำหนดจากผู้สร้างหรือเก็บสถิติจากการเล่นของผู้เล่น เพื่อพัฒนาเงื่อนไขเพิ่มเติม ในขั้นตอนถัดมาเครื่องอนุมานจะนำเอาค่าความจริงมา

เปรียบเทียบกับเงื่อนไขแต่ละกฎ หากกฎใดมีเงื่อนไขถูกต้องเครื่องอนุมานก็จะปฏิบัติตามส่วนกระทำของกฎนั้น จากนั้นเครื่องอนุมานจะเปรียบเทียบเงื่อนไขของกฎถัดไปจนกว่าจะหมดข้อมูลเงื่อนไข




**รูปที่ 2.30** กระบวนการทำงานของ Forward chaining

**2. การอนุมานแบบย้อนกลับ(Backward-Chaining)** เป็นวิธีการอนุมานอีกวิธีหนึ่ง โดยเริ่มต้นจากเป้าหมาย ( Goals ) ที่ต้องการแล้วดำเนินการย้อนกลับเพื่อหาสาเหตุการอนุมาน ในลักษณะนี้มักนำมาใช้ในการพัฒนาระบบความฉลาดที่มีความเข้าใจ




**รูปที่ 2.31** กระบวนการทำงานของ Backward chaining

# บทที่3
# ขั้นตอนการดำเนินงานวิจัย

ขั้นตอนในการดำเนินงานวิจัยทั้งหมดสามารถแบ่งได้เป็น 3 ส่วน ได้แก่ ส่วนของการวิเคราะห์และออกแบบเกม ส่วนของการลงมือปฏิบัติจริง และส่วนของการทดสอบเกม

1) ขั้นตอนการวิเคราะห์และออกแบบเกม

เป็นการวางรูปแบบของเกมให้มีรูปแบบการเล่นอย่างไร กฎกติกาของเกม ระบบการเล่น หน้าต่างติดต่อกับผู้เล่น ตัวละครในเกม ท่าทางของตัวละครแต่ละตัว การตอบสนองของหุ่นฝ่ายศัตรู ฉาก เสียงตอบสนองและดนตรีประกอบ

2) ขั้นตอนการลงมือปฏิบัติจริง

เมื่อได้แบบหุ่นและฉากที่ต้องการแล้ว ขั้นต่อมาจึงทำการสร้างโมเดลหุ่นและฉากด้วยโปรแกรม Maya ซึ่งเมื่อทำการสร้างเสร็จแล้วจึงทำการใส่ท่าทางต่างๆ ให้มีการเคลื่อนไหว ตามที่ได้ออกแบบไว้ จากนั้นจึงนำเข้ามาให้กับโปรแกรม Quest3D ด้วยคำสั่ง Import

3) ปัญหาและอุปสรรคของการพัฒนาโปรแกรม

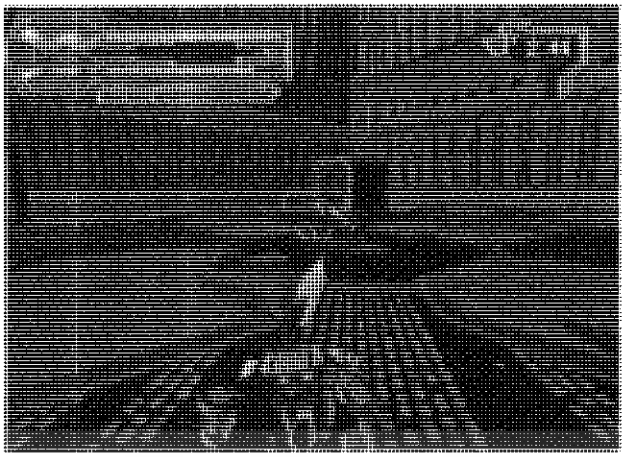
ในระหว่างการสร้างโปรแกรมนั้นมักพบเจอปัญหาต่างๆซึ่งต้องมีการหาสาเหตุของปัญหาและวิธีการแก้ปัญหานั้นๆให้ได้เสียก่อนจึงสามารถเนินการทำงานในส่วนต่อไปได้

## 3.1 ขั้นตอนการออกแบบและวิเคราะห์เกม

### 3.1.1 การออกแบบรูปแบบ กติกาการเล่นและระบบเกม

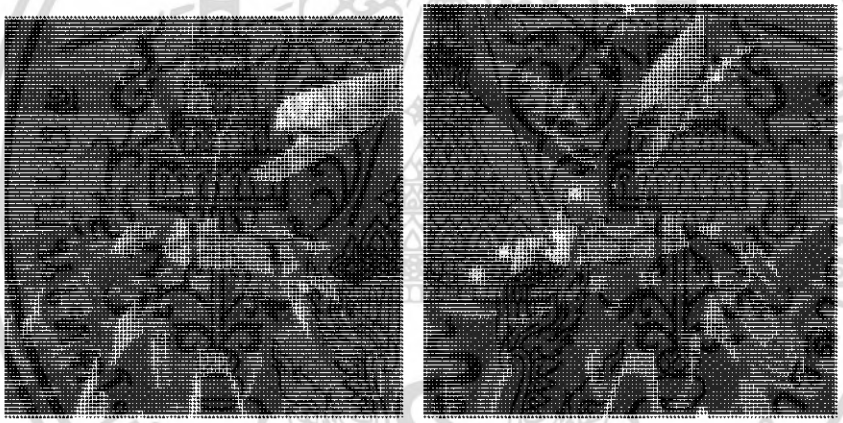
เกม Mebius Combat นั้นเป็นเกมแนวต่อสู้ของหุ่นยนต์แบบมุมมองบุคคลที่สาม โดยมีรูปแบบการต่อสู้แบบตัวต่อตัว ซึ่งการที่จะเป็นผู้ชนะได้นั้น ผู้เล่นจะต้องทำความเสียหายจนกระทั่งพลังชีวิตของหุ่นยนต์ที่เป็นคู่ต่อสู้นั้นหมดลง หรือหากสามารถทำความเสียหายให้กับคู่ต่อสู้ ซึ่งเมื่อเปรียบเทียบกับความเสียหายที่เราได้รับแล้วถือว่าสูงกว่าก็จะเป็นผู้ชนะเช่นกัน

***3.1.1.1 รูปแบบการต่อสู้*** - เราจะทำการบังคับตัวละครหุ่นยนต์หนึ่งตัว เพื่อใช้ในการต่อสู้กับฝ่ายตรงข้าม ด้วยการเคลื่อนไหวและปืนเลเซอร์

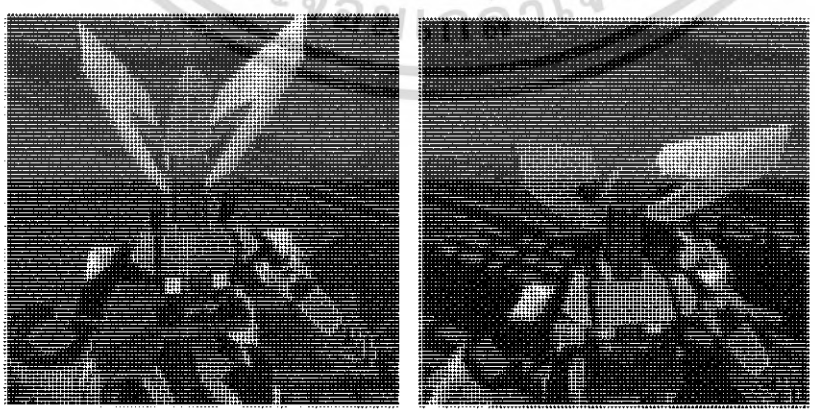


**รูปที่ 3.1** รูปแบบการต่อสู้

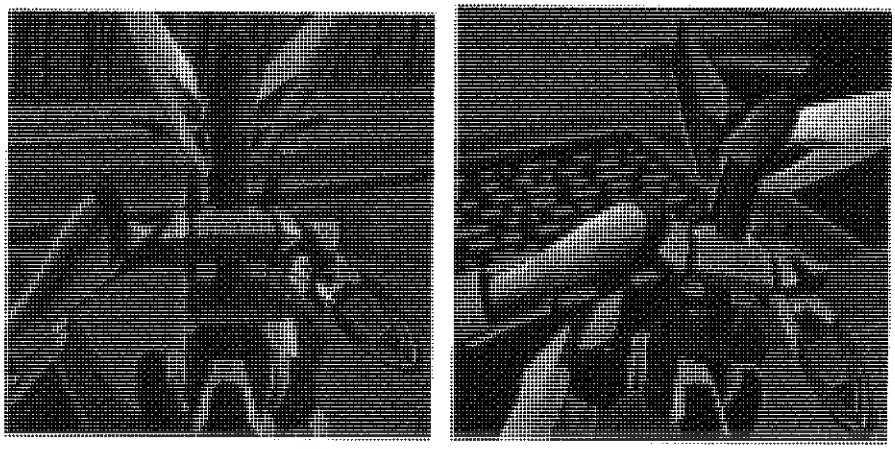
***3.1.1.2 การเคลื่อนไหว*** - การเคลื่อนไหวที่ผู้เล่นสามารถที่จะกระทำได้ในเกมคือ การเคลื่อนไหวไปข้างหน้า ข้างหลัง ข้างซ้ายและข้างขวา การกระโดด การพุ่งด้วยความเร็ว



**รูปที่ 3.2** การเดินไปทางซ้าย และขวาของตัวละคร



**รูปที่ 3.3** การเดินไปข้างหน้าและถอยหลัง

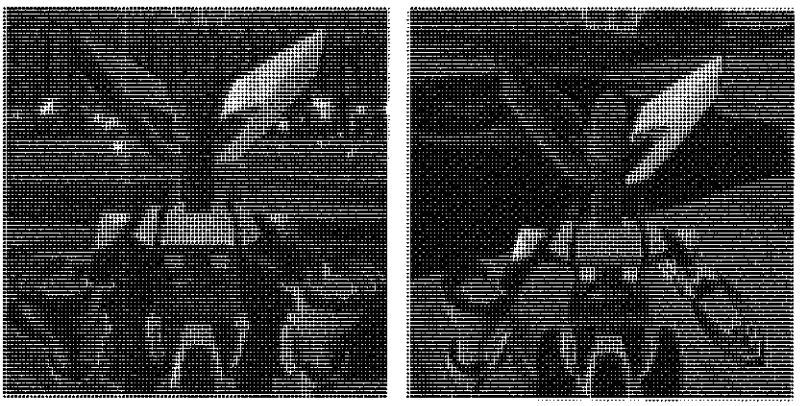


**รูปที่ 3.4** การกระโดดและการพุ่งด้วยความเร็ว

*3.1.1.3 มุมมองของตัวละคร* – มุมมองของตัวละครนั้นจะอยู่ที่ตำแหน่งกลางจอเสมอ โดยจะมีรูปเป้าไว้สำหรับเล็งเป้าหมายของกระสุน หากมีการเลื่อนเมาส์ก็จะทำให้มุมการมองนั้นเลื่อนตามไปด้วย

**รูปที่ 3.5** รูปภาพเป้าที่อยู่กึ่งกลางของหน้าจอ และ มุมมองบนหน้าจอ

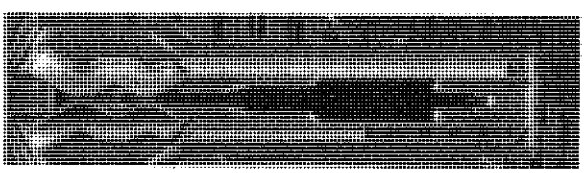
*3.1.1.4 การใช้อาวุธและระบบโหมด* – ระบบการใช้อาวุธนั้นจะประกอบไปด้วยโหมด ซึ่งโหมดจะเป็นตัวกำกับการใช้อาวุธของผู้เล่น เช่น โหมดถือปืนก็สามารถที่จะยิงกระสุนได้ หรือโหมดปกติที่ทำท่าป้องกันได้แต่ไม่สามารถยิงกระสุนได้ สำหรับระบบที่พัฒนาขึ้นนั้นจะมีโหมดอยู่เพียงสองโหมด โดยมีอาวุธให้ใช้อยู่เพียงแค่อาวุธเดียว นั่นก็คือโหมดอาวุธปืนเลเซอร์ และโหมดปกติที่จะทำให้พลังงานของหุ่นยนต์ขึ้นเร็วขึ้นและทำท่าป้องกันได้ แต่จะไม่สามารถยิงกระสุนได้



**รูปที่ 3.6** ตัวละครผู้เล่น โดยภาพทางซ้ายเป็นภาพขณะอยู่ในโหมดปกติ และภาพทางขวาเป็นภาพขณะอยู่ในโหมดถืออาวุธ

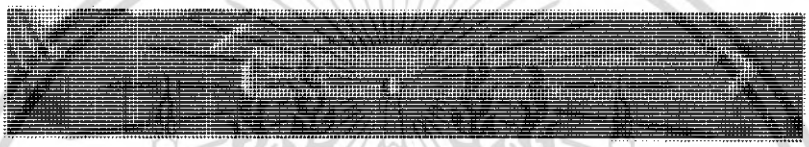
**รูปที่ 3.7** การยิงกระสุนและการป้องกันของตัวละครผู้เล่น

***3.1.1.5 พลังชีวิตของหุ่นยนต์*** – ในการต่อสู้นั้น ทั้งสองฝ่ายจะมีพลังงานชีวิตของตนเอง ซึ่งถ้าหากพลังงานชีวิตนี้หมดลง ก็จะเท่ากับเป็นฝ่ายที่พ่ายแพ้ในทันที ซึ่งพลังชีวิตของตัวละครทั้งสองฝ่ายนั้นจะถูกแสดงอยู่บนแถบพลังงานชีวิตที่อยู่ทางซ้ายบนของหน้าจอ โดยหลอดที่อยู่ทางด้านบนจะเป็นของศัตรู และหลอดที่อยู่ทางด้านล่างจะเป็นของผู้เล่น ซึ่งหลอดทั้งสองนี้จะแสดงพลังชีวิตที่เหลืออยู่ของผู้เล่นและของศัตรูในการเปรียบเทียบเชิงเปอร์เซ็นต์



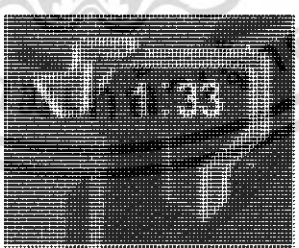
**รูปที่ 3.8** แถบวัดพลังงานชีวิตของผู้เล่นและศัตรู

***3.1.1.6 พลังงานของหุ่นยนต์*** – การใช้อาวุธ และการพุ่งด้วยความเร็วนั้นจะต้องใช้พลังงานของหุ่นยนต์ที่มีอยู่ ซึ่งหากว่าพลังงานตรงนี้หมดไปแล้วนั้นก็จะทำให้ไม่สามารถยิงกระสุน หรือพุ่งด้วยความเร็วได้ โดย*ค่า*พลังงานของหุ่นยนต์นี้จะเพิ่มขึ้นตามเวลาที่ผ่านไป และจะเพิ่มได้เร็วขึ้นไปอีกขั้นในขณะที่ผู้เล่นอยู่ในโหมดปกติ พูดให้ฟังเข้าใจง่ายคือหากหลอดพลังงานอาวุธนี้มีการถูกใช้ไปและไม่ได้อยู่ในสถานะเต็มหลอดนั้น ทุกๆหน่วยเวลาที่ผ่านไปหลอดพลังงานนี้ก็จะถูกชาร์จเพิ่มขึ้นทีละนิดๆ ตัวอย่างเช่น เราทำการยิงปืนเลเซอร์ที่ใช้พลังงาน 10% ของหลอดพลังงาน ทำให้พลังงานของหลอดเหลือเพียง 90% เมื่อเวลาผ่านไป 5 วินาทีหากไม่ได้มีการใช้ พลังงานของหลอดที่เหลืออยู่ก็จะขึ้นมาอยู่ที่ 95% นั่นหมายความว่าหากเราไม่ได้ใช้ พลังงานในหลอดก็จะถูกสะสมขึ้นเรื่อยๆจนเต็มได้ใหม่ ทั้งนี้นั้นพลังงานในหลอดที่มีอยู่ไม่สามารถเก็บได้เกิน 100%



**รูปที่ 3.9** แถบวัดพลังงานหุ่นยนต์ของผู้เล่น

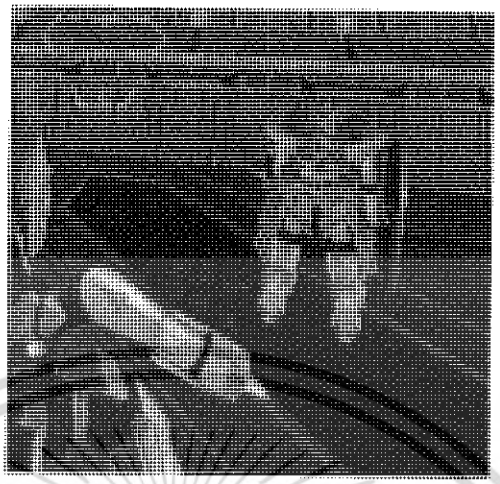
***3.1.1.7 เวลาที่ใช้กำกับการต่อสู้*** – การต่อสู้จะถูกกำกับด้วยเวลาในแต่ละรอบการต่อสู้ ซึ่งกำหนดไว้ที่ 4 นาที หากเวลาที่กำกับการต่อสู้นั้นได้หมดลงโดยที่ยังไม่สามารถตัดสินผลแพ้ชนะกันได้ ระบบก็จะทำการตัดสินผลแพ้ชนะกันที่พลังงานชีวิต ซึ่งฝ่ายไหนมีมากกว่าก็จะเป็นฝ่ายชนะ ในกรณีที่พลังชีวิตของทั้งสองฝ่ายเหลือเท่ากัน ก็จะตัดสินให้ฝั่งศัตรูเป็นฝ่ายชนะไป




**รูปที่ 3.10** ตัวนับเวลาที่ปรากฏอยู่บริเวณขวาบนของหน้าจอ

***3.1.1.8 คู่ต่อสู้ของผู้เล่น*** – เนื่องจากว่าระบบเกมนั้นถูกออกแบบมาเพื่อให้เล่นได้เพียงแค่คนเดียว ทำให้คู่ต่อสู้ของผู้เล่นนั้นจะถูกบังคับด้วยระบบปัญญาประดิษฐ์ โดย

ระบบปัญญาประดิษฐ์นั้นจะทำหน้าที่เหมือนกับเป็นผู้เล่นอีกคนซึ่งมีระบบการควบคุมที่ไม่ได้แตกต่างไปจากตัวผู้เล่น



**รูปที่ 3.11** ตัวละครของศัตรู

**3.1.2 การออกแบบวัตถุสามมิติเพื่อใช้ภายในเกม**

วัตถุสามมิติที่จะใช้ในเกมนั้น เป็นวัตถุที่ได้ทำการสร้างขึ้นเองหมดโดยใช้โปรแกรม Maya เพื่อสร้างวัตถสามมิติ และ Okino-Exporter เพื่อแปลงให้เป็น X file ผนวกเข้ากับเอฟเฟกต่างๆที่เอนจิน Quest3D มีให้ ซึ่งจะประกอบไปด้วยวัตถุสามมิติหลักๆสามประเภท

***3.1.2.1 วัตถุสามมิติประเภทตัวละคร***

ตัวละครที่ใช้ภายในเกมนั้นจะมีอยู่ทั้งหมด 2 ตัวละคร คือ

- ***ES Dinah***

ตัวละครตัวนี้จะถูกบังคับโดยผู้เล่น

**รูปที่ 3.12** หุ่นยนต์ ES Dinah

- ***Hi-nu Gundam***

ตัวละครตัวนี้จะถูกบังคับโดยระบบปัญญาประดิษฐ์

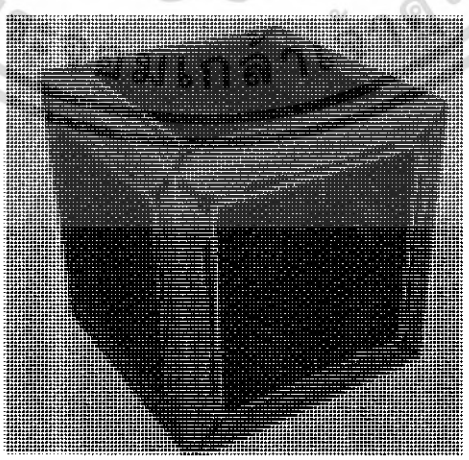
**รูปที่ 3.13** หุ่นยนต์ Hi-nu Gundam

หุ่นยนต์ทั้งสองชนิดนั้นจะกำหนดให้มีสมรรถนะที่เหมือนกัน ไม่ว่าจะเป็นความเร็วในการเดิน ปริมาณพลังงานที่ใช้ ความเสียหายที่สามารถกระทำได้ในหนึ่งกระสุน ทั้งนี้ก็เพื่อความสมดุลและความสูสีในการต่อสู้

### *3.1.2.2 วัตถุสามมิติประเภทฉาก และสิ่งแวดล้อม*

สำหรับฉากนั้นจะมีลักษณะเป็นพื้นที่สี่เหลี่ยมล้อมรอบด้วยกำแพงสูง ประกอบกับมี กล่องสี่เหลี่ยมสี่กล่อง และขอบฉากที่จะมีพื้นที่ ที่เป็นลักษณะเนินยกสูงขึ้น เพื่อกีดขวางการโจมตีและการมองเห็นของทั้งผู้เล่นและระบบปัญญาประดิษฐ์ การที่ได้ออกแบบลักษณะฉากเป็นเช่นนี้นั้น เพราะจะช่วยให้ระบบปัญญาประดิษฐ์สามารถทำงานได้ค่อนข้างดีและไม่จำเป็นที่จะต้องมีฟังก์ชันการตรวจจับที่ซับซ้อนมากจนเกินไปนัก

**รูปที่ 3.14** ฉากที่ใช้ในการต่อสู้



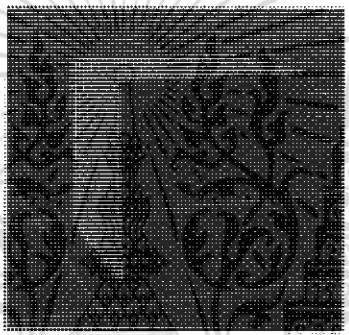
**รูปที่ 3.15** กล่องที่จะใช้เป็นสิ่งกีดขวางในการต่อสู้

### *3.1.2.3 วัตถุสามมิติประเภทเอฟเฟกและวัตถุสามมิติที่มีให้ใน Quest3D*

ในเอนจิน Quest3D มีวัตถุพื้นฐานให้เราได้เลือกใช้ รวมถึงเอฟเฟกต่างๆ ซึ่งจะช่วยอำนวยความสะดวกให้กับการสร้างระบบเกมได้ไม่น้อยเลยทีเดียว

- ***วัตถุพื้นฐานในเอนจิน Quest3D***

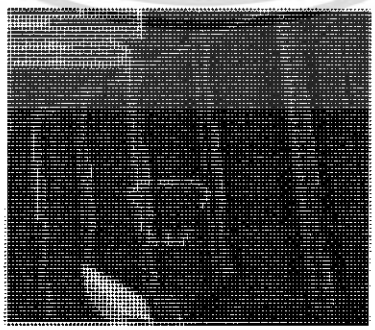
  วัตถุพื้นฐานในเอนจิน Quest3D นั้นประกอบไปด้วย สี่เหลี่ยม วงกลม แผ่นราบ โคน รูปทรงกระบอก เส้น หรือวงแหวน ซึ่งวัตถุเหล่านี้นั้นอาจไม่ค่อยมีประโยชน์ในด้านการแสดงรูปร่างสักเท่าไหร่เพราะมีลักษณะเรียบง่ายจนเกินไป หากแต่จะนำมาใช้ในกระบวนการตรวจสอบการชนซะเป็นส่วนใหญ่ ซึ่งทำประโยชน์ให้กับผู้จัดสร้างได้มากกว่า



**รูปที่ 3.16** วัตถุพื้นฐานที่เป็นกล่องสี่เหลี่ยม

- ***เอฟเฟกต่างๆที่มีอยู่ในเอนจิน Quest3D***

  ในเอนจิน Quest3D มีเอฟเฟกต่างๆให้เลือกใช้ได้อย่างหลากหลาย ซึ่งเอฟเฟกเหล่านี้นั้นจะถูกใช้เพื่อตกแต่งฉากให้มีความสวยงาม รวมถึงถูกใช้เป็นแสงของกระสุนปืนที่ใช้ในการโจมตีกันอีกด้วย

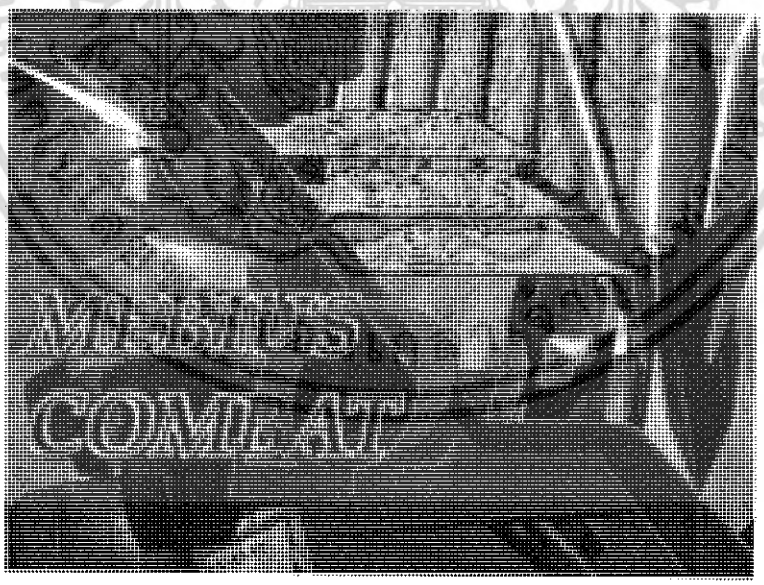


**รูปที่ 3.17** เอฟเฟกแสงที่วิ่งอยู่บนผนัง

**รูปที่ 3.18** เอฟเฟกแสงที่ใช้ในการยิงกระสุน

**3.1.3 การออกแบบหน้าจอหลัก**

หน้าจอเมนูถูกออกแบบให้มีปุ่มอยู่ทั้งหมดสามปุ่ม โดยปุ่มแรกจะเป็นปุ่ม Start ซึ่งปุ่ม Start นั้นจะเป็นการเริ่มเล่นเกมในระบบต่อสู้ ปุ่มที่สองจะเป็นปุ่ม Credits ซึ่งเมื่อกดแล้วจะแสดงหน้าต่างรายชื่อผู้จัดทำ และปุ่มสุดท้ายเป็นปุ่ม Exit ซึ่งเมื่อกดแล้วจะเป็นการออกจากเกม



**รูปที่ 3.19** หน้าจอเมนูของเกม

รูปที่ 3.20 หน้าจอผู้จัดทำของเกม

**3.1.4 การออกแบบด้านเสียงประกอบเกม**

อันเนื่องมาจากว่าทางกลุ่มผู้จัดทำนั้นไม่มีผู้เชี่ยวชาญในด้านการแต่งเพลง จึงจำเป็นที่จะต้องนำเพลงจากที่อื่นมาใช้ประกอบลงในเกมแทน

ไฟล์เสียงที่จะทำมาใช้นั้นเป็นสกุลไฟล์ชนิด Wave เพราะว่า Quest3D ไม่รองรับการบรรจุเสียงสกุล mp3 ลงภายในไฟล์ โดยจะบรรจุแค่พาธตำแหน่งลงไป ถ้าหากเกิดการเปลี่ยนตำแหน่งของไฟล์ ชนิด mp3 ตัวเอนจิน Quest3D จะหาไฟล์นั้นไม่เจอในขณะทำงานทำให้ไม่สามารถเล่นเสียงได้ ผิดกับไฟล์ชนิด Wave ซึ่งจะถูกบรรจุลงไปในไฟล์ Quest3D จริงๆเลย ทำให้ไม่เกิดปัญหาด้านการเรียกไฟล์

**3.1.5 การออกแบบระบบปัญญาประดิษฐ์**

การออกแบบระบบปัญญาประดิษฐ์นั้นจะทำให้มีลักษณะคล้ายกับการตัดสินใจของมนุษย์มากที่สุด โดยจะออกแบบส่วนตัดสินใจสำหรับการกดปุ่มแต่ละปุ่มเหมือนการตัดสินใจในการกดแป้นพิมพ์ของผู้เล่น หลักในการคิดนั้นเริ่มจากจุดที่ว่า ทำไมมนุษย์จำเป็นที่จะต้องตัดสินใจแบบนั้นไปในรูปแบบของสถานการณ์ที่เกิดขึ้น ถ้าเราสามารถเรียนรู้และเข้าใจหลักการนี้ได้ ก็จะสามารถนำมาประยุกต์สร้างเป็นระบบปัญญาประดิษฐ์แบบพื้นฐานได้ โดยการออกแบบนั้นจะอิงหลักการ rule based system โดยทำการกำหนดเงื่อนไขในสถานการณ์ต่างๆ เสริมเข้ากับการสุ่มทางตัวเลข ทำให้เกิดการตัดสินใจที่หลากหลายและเหมาะสมในแต่ละสถานการณ์ จึงได้มีการคิดลักษณะการออกแบบเชิงกระจายโครงสร้าง ส่วนรับผิดชอบการทำงานจะถูกแบ่งออกเป็นส่วนต่างๆ ทำงานใน

ลักษณะแยกกันทำงาน แต่จะมีตัวแปรควบคุมที่เชื่อมต่อถึงกัน เพราะการทำงานบางส่วนนั้นจำเป็นต้องขึ้นต่อการทำงานของอีกส่วน เช่น หากส่วนที่รับผิดชอบการทำงานในการที่จะเปลี่ยนโหมดของหุ่นเกิดตัดสินใจขึ้นมาได้ว่าควรจะมีการเปลี่ยนโหมด ส่วนการทำงานเปลี่ยนโหมดก็จะต้องแจ้งเตือนให้ส่วนรับผิดชอบการยิงรู้ว่าสถานะกำลังจะเปลี่ยนไป หากว่าหุ่นยนต์ไม่ได้อยู่ในสถานะถือปืน ก็จะไม่สามารถทำการยิงได้ ส่วนการทำงานด้านการยิงจะต้องหยุดตัวเองเพื่อรอให้ส่วนการทำงานการเปลี่ยนโหมดเปลี่ยนกลับมาเป็นโหมดถือปืนจึงจะทำงานต่อไปได้ ด้วยลักษณะการทำงานที่ได้กล่าวไป ทำให้ต้องคิดกรณีให้ครอบคลุมที่สุดเท่าที่จะทำได้ ว่าส่วนไหนจะทำงานคลุมอยู่บนส่วนใดบ้าง

หลักๆแล้ว ระบบปัญญาประดิษฐ์จะรับข้อมูลเข้าสำหรับการประมวลผลเป็นค่าของตัวแปรต่างๆที่อยู่ในระบบ อาทิ เลือดของระบบปัญญาประดิษฐ์ เลือดของผู้เล่น พลังงานของผู้เล่น พลังงานของตัวเอง ส่วนต่างของเลือด ระยะห่างกับวัตถุต่างๆ และอื่นๆ แต่ก็จะมีหลายส่วนการทำงานที่จะต้องรอรับผลลัพธ์การทำงานจากอีกส่วน อาทิ การประเมินการยืนระยะ ซึ่งเราต้องทราบก่อนว่าส่วนการทำงานที่ตัดสินใจว่าระบบปัญญาประดิษฐ์ควรจะอยู่ที่ระยะห่างเท่าไหร่ก่อนจะประมวลผลการกดปุ่มเดินหน้าหรือถอยหลังเพื่อสร้างระยะตามที่เหมาะสม

เมื่อเราออกแบบการทำงานแต่ละส่วนและความสัมพันธ์ของแต่ละส่วนแล้ว จึงทำการรวมระบบและทดสอบตัวแปรควบคุมต่างๆว่าทำงานได้อย่างถูกต้อง การทำงานของระบบปัญญาประดิษฐ์ที่ได้ออกแบบไว้นั้นไม่ได้ยากจนเกินไป แต่มีความซับซ้อนเชิงโครงสร้างค่อนข้างสูง จึงต้องทดสอบการทำงานให้มั่นใจก่อนที่จะสรุปว่าสามารถทำงานได้จริง

## 3.2 ขั้นตอนการลงมือปฏิบัติจริง

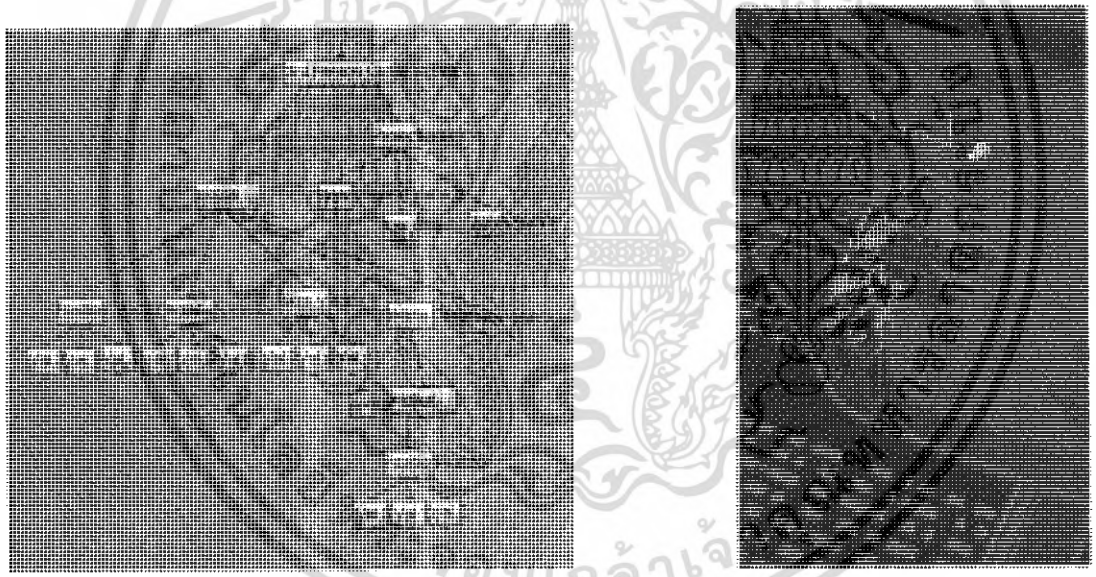
### 3.2.1 ระบบภายในเกม

#### 3.2.1.1 กล้องและหน้าจอการมอง

กล้องเป็นตัวที่ทำหน้าที่ในการแสดงภาพในระยะขอบเขตที่สามารถมองเห็นได้ โดยสามารถปรับระยะความใกล้-ไกลในการมองเห็น และสามารถที่จะทำการกำหนดตำแหน่งและมุมองศาของกล้อง

ประเภทของกล้องที่ใช้ภายในระบบเกม

- Basic Camera เป็นกล้องที่ใช้แสดงเกี่ยวกับหน้าเมนู และ แถบที่ใช้แสดงพลังชีวิตกับพลังงานของตัวละคร
- 3rd Person Camera เป็นกล้องที่ใช้แสดงตัวละครที่เคลื่อนไหว โดยจะวางไว้ด้านหลังของตัวละคร ทำให้เสมือนว่าเป็นมุมมองบุคคลที่สาม



**รูปที่ 3.21** แสดงโหนดของ3rd Person Camera(ซ้าย) และการวางตำแหน่งของกล้อง(ขวา)

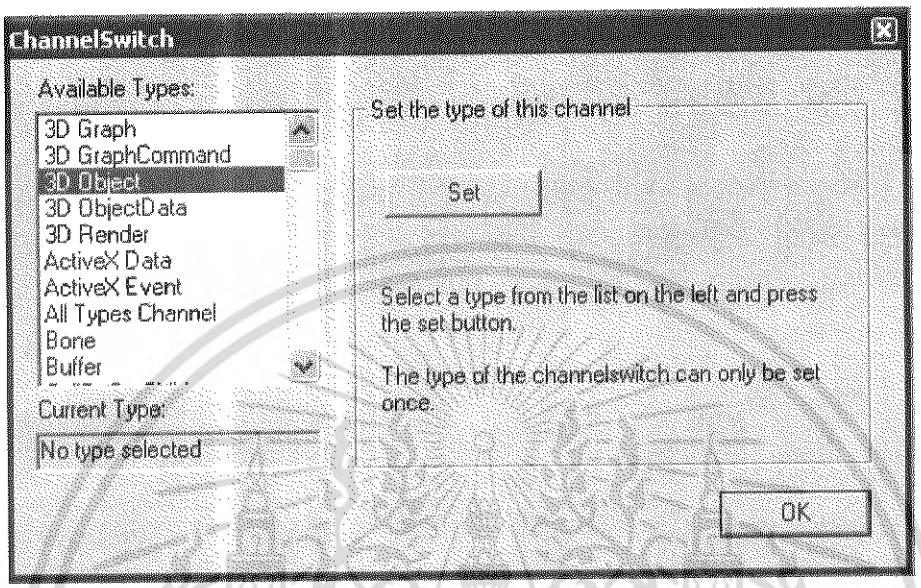
#### 3.2.1.2 อนิเมชันและวัตถุ

วัตถุที่นำมาใช้ในระบบเกมมีอยู่2แบบ คือ

- อนิเมชัน เป็นวัตถุที่สามารถแสดงท่าทางเคลื่อนไหวได้ โดยสามารถที่จะกำหนดเวลาในแต่ละเฟรมให้สามารถเคลื่อนไหวได้ เช่น ตัวละคร
- วัตถุเป็นสิ่งที่ไม่สามารถขยับได้ เช่น ฉาก สิ่งกีดขวาง

**3.2.1.3 การสลับวัตถุ**

ฟังก์ชันที่ใช้ในการสลับวัตถุจะเรียกว่า Channel Switch โดยChannel Switch ใช้สำหรับกำหนดเงื่อนไขโดยอนุญาติให้สามารถกำหนดชนิดของข้อมูลที่จะมาเชื่อมเข้ากับChannel Switchได้ โดยจำเป็นต้องกำหนดชนิดของข้อมูลที่จะมีการใช้งาน

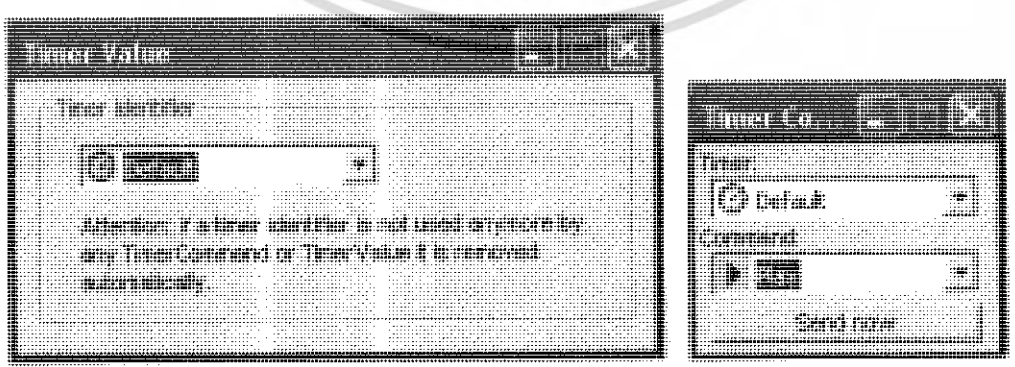



**รูปที่ 3.22** แสดงการใช้งานChannel Switch

**3.2.1.4 คำสั่งการจัดการด้านเวลา**

คำสั่งที่ใช้จัดการด้านเวลาจะมี 2 คำสั่ง คือ

- Timer Value เป็นการกำหนดชื่อของเวลาที่จะใช้ควบคุมการเคลื่อนไหวของตัวละคร
- Timer Command Channel เป็นการควบคุมการเคลื่อนไหวของตัวละครให้แสดงในเวลาที่กำหนด

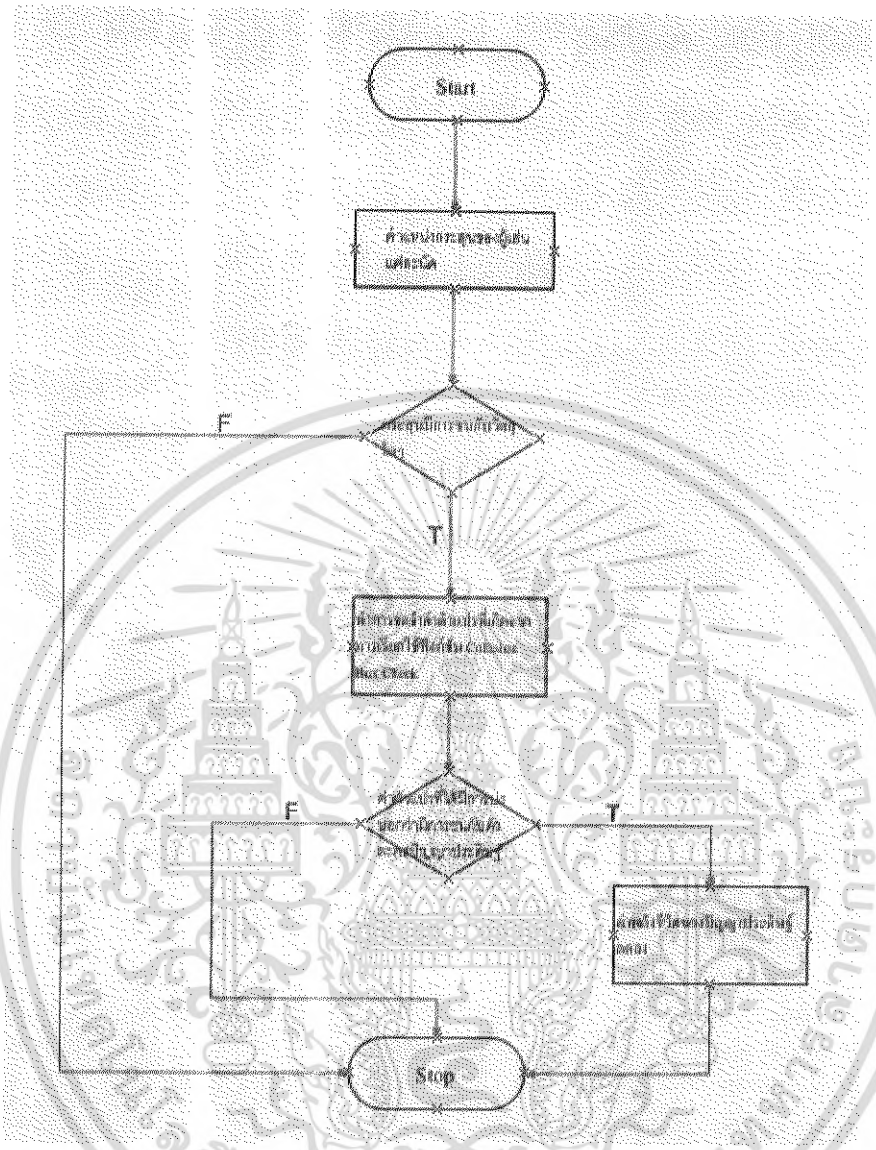


**รูปที่ 3.23** แสดงการใช้งานTime Valueและ Timer Command Channel

**3.2.1.5 การตรวจสอบการชนระหว่างวัตถุ**

ฟังก์ชันที่ใช้ตรวจสอบการชนระหว่างวัตถุจะมีอยู่ 2 แบบ คือ

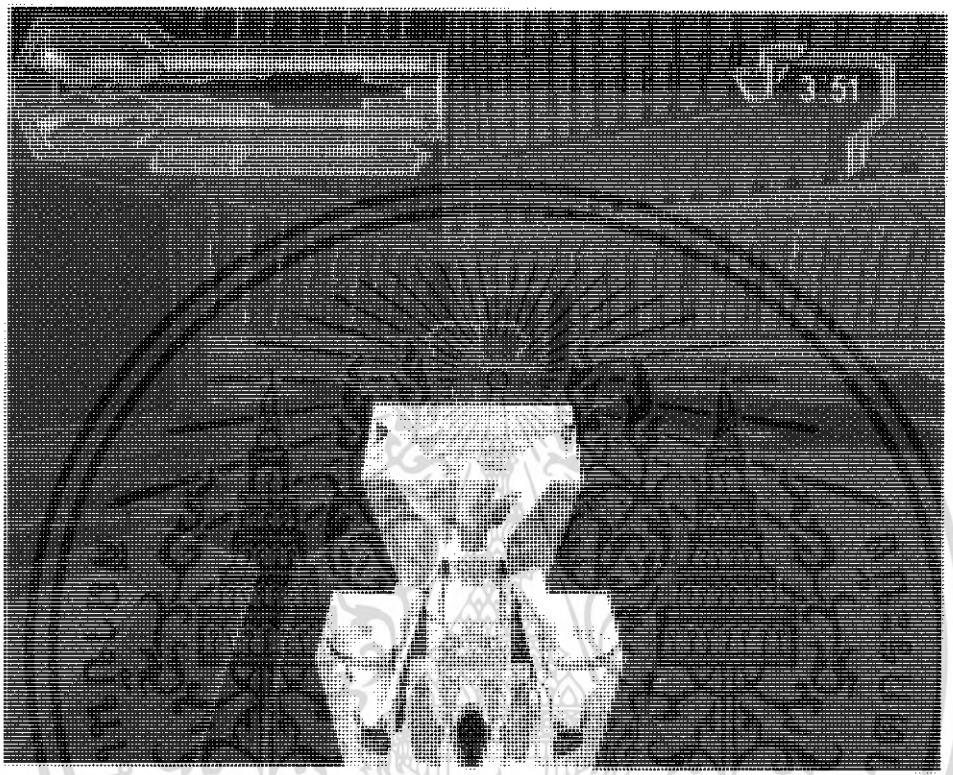
- ฟังก์ชันการตรวจสอบการชนโดยอัตโนมัติ เป็นฟังก์ชันที่ไม่แสดงค่าตัวเลขขณะที่ชนกับวัตถุใดวัตถุหนึ่ง โดยส่วนใหญ่จะใช้ตรวจสอบการชนระหว่างตัวละครกับฉากหรือระหว่างตัวละครทั้ง2ตัว ฟังก์ชันดังกล่าวได้แก่ Collision Object, Fast Collision Respondเป็นต้น
- ฟังก์ชันการตรวจสอบการชนโดยบอกค่าตัวเลข เป็นฟังก์ชันที่แสดงค่าตัวเลขขณะที่ชนกับวัตถุใดวัตถุหนึ่ง โดยส่วนใหญ่จะใช้ตรวจสอบการชนระหว่างกระสุนกับตัวละครหรือระหว่างกระสุนกับฉาก โดยจะนำค่าตัวเลขดังกล่าวไปทำการคำนวณเพื่อใช้ลดค่าพลังชีวิตของอีกฝ่ายหรือเกิดการระเบิดขึ้น ฟังก์ชันดังกล่าวได้แก่ Collision Box Check เป็นต้น




**รูปที่ 3.24** flowchart diagram แสดงการเขียนโปรแกรมเพื่อตรวจสอบการชนศัตรู

**3.2.1.6 วัตถุเสมือนเพื่อตรวจสอบการชน**

วัตถุเสมือนที่ใช้ตรวจสอบการชนเป็นวัตถุที่มีลักษณะเป็นกล่องสี่เหลี่ยม โดยกล่องแต่ละกล่องจะถูกสร้างขึ้นมาเพื่อครอบแต่ละส่วนของตัวละครและทำการผูกติดไปกับตัวละคร ซึ่งกล่องที่ถูกสร้างขึ้นมานั้นจะสร้างให้มีลักษณะที่ใกล้เคียงกับรูปร่างของตัวละครให้มากที่สุด เพื่อให้สามารถตรวจสอบการชนแต่ละส่วนของตัวละครได้



**รูปที่ 3.25** แสดงภาพวัตถุเสมือนของตัวละคร

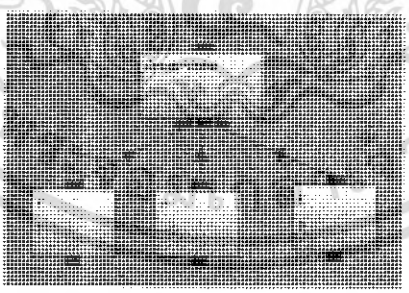
**3.2.1.7 การกำหนดเส้นทางการเคลื่อนที่ของวัตถุ**

ฟังก์ชันที่ใช้ในการกำหนดเส้นทางการเคลื่อนที่ของวัตถุจะเรียกว่า Envelope โดยสามารถกำหนดจุดพิกัดตำแหน่งเส้นทางแต่ละจุดให้กับวัตถุ เพื่อให้สามารถเคลื่อนที่และหมุนได้ตามที่ต้องการซึ่งEnvelopeจะทำการเปลี่ยนแปลงค่าต่างๆเมื่อเวลาผ่านไป

**รูปที่ 3.26** แสดงการกำหนดเส้นทางการเคลื่อนที่ของวัตถุ

**3.2.1.8 แรงโน้มถ่วงสำหรับวัตถุ**

ฟังก์ชันที่แสดงค่าแรงโน้มถ่วงสำหรับวัตถุจะเรียกว่า Gravity โดยสามารถที่จะกำหนดค่าของแรงโน้มถ่วงขณะที่กำลังจะตกลงสู่พื้น ซึ่งสามารถปรับความเร็วให้วัตถุตกลงสู่พื้นเร็วหรือช้าได้



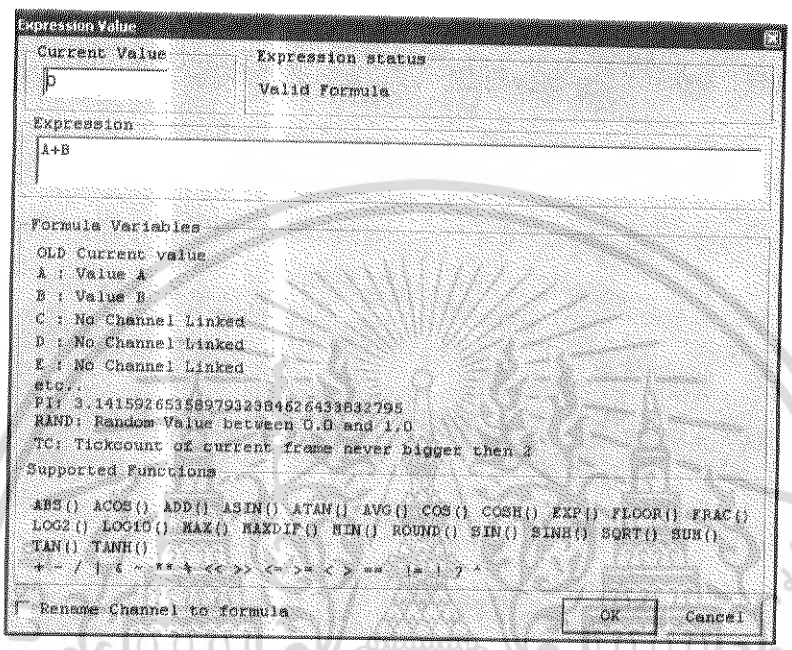
**รูปที่ 3.27** แสดงฟังก์ชันแรงโน้มถ่วงของวัตถุ

**3.2.1.9 การสร้างตรรกะทางคณิตศาสตร์**

ฟังก์ชันที่ใช้สำหรับการสร้างตรรกะทางคณิตศาสตร์จะเรียกว่าExpression Value โดยสามารถสร้างคำสั่งการคำนวณทางคณิตศาสตร์และคำสั่งการเปรียบเทียบของข้อมูล ซึ่งจะมีช่องให้กำหนดสูตรและมีข้อความอธิบายตัวแปรและตัวดำเนินการต่างๆ

ฟังก์ชันที่ใช้ในการสร้างตรรกะทางคณิตศาสตร์ มีดังนี้

- การบวก(+) , การลบ( -) , การคูณ(*)
- หาร(/) , การสุ่มค่า(Rand)
- ฟังก์ชันตรีโกณมิติ ได้แก่ ฟังก์ชัน Sin , Cos , Tan
- ฟังก์ชันการเปรียบเทียบ ได้แก่ AND(&) , OR(||) , < , <= , = = , > , >=



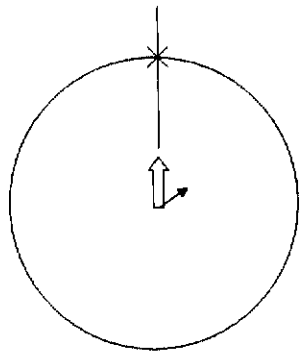

**รูปที่ 3.28** แสดงฟังก์ชันการสร้างตรรกะทางคณิตศาสตร์

### 3.2.1.10 การยิงตัดตำแหน่งเป้าหมาย

เป็นการยิงทำมุมให้ผ่านจุดเป้าหมายในระยะทางที่กำหนด สาเหตุที่ต้องมีการคำนวณในส่วนนี้นั้น เนื่องมาจากในการยิงนั้นระบบเกมต้องการให้มีการยิงโดนเมื่อตัวละครหันหน้าตรงเป้าหมายแต่ในความเป็นจริงนั้นตัวละครต้องมีการถืออาวุธไว้ที่มือข้างหนึ่ง ทำให้ไม่สามารถเล็งเป้าหมายได้จากกลางตัวละคร

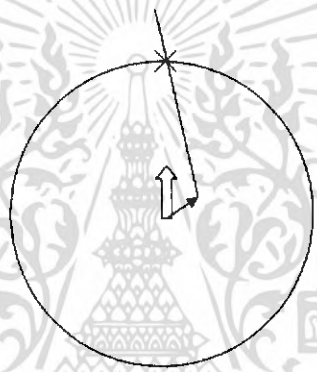
อัลกอริทึม

- ทำการหาเป้าหมายซึ่งเป็นตัวศัตรูก่อนว่าอยู่ห่างไปเป็นระยะทางเท่าใดในระนาบแกน XZ โดยใช้สูตรคำนวณหาระยะทางคือ $p_1p_2=SQRT((x_1-x_2)^2+(z_1-z_2)^2)$
- เมื่อได้ระยะห่างมาแล้วจะทำให้รู้จุดตัดในทิศทางที่หุ่นหันไป



**รูปที่ 3.29** แสดงการคำนวณหาตำแหน่งเป้าหมาย

- หลังจากที่ได้พิกัดการยิงแล้วขั้นต่อมาคือต้องทำการหาองศาการยิงตัวใหม่โดยนับจากปากกระบอกปืนหรือจุดที่กระสุนเริ่มทำการยิง



**รูปที่ 3.30** แสดงการคำนวณการยิงกระสุนตัดตำแหน่งที่เป้าหมาย

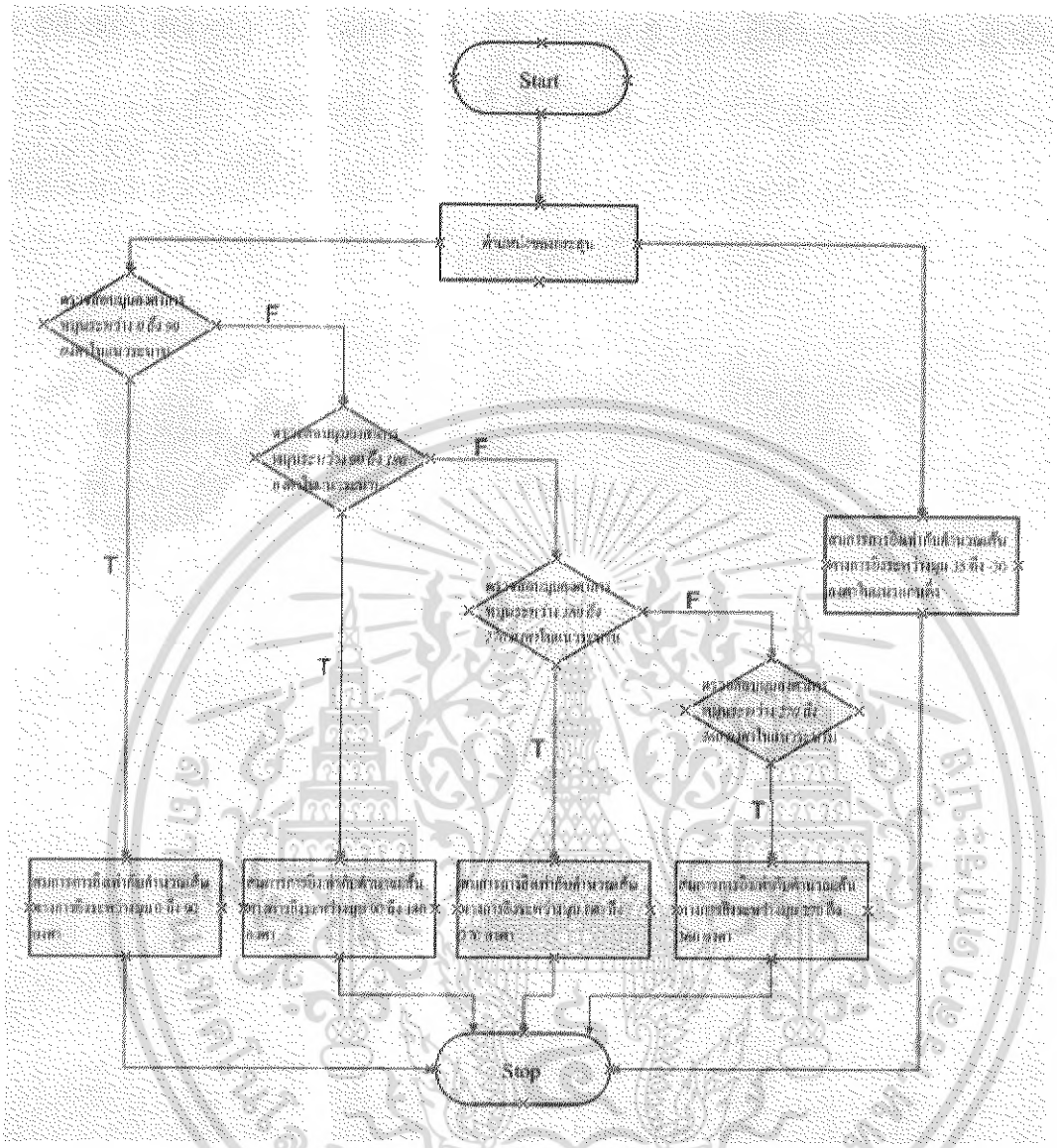
- เมื่อได้องศาการยิงใหม่และพิกัดเป้าหมายเรียบร้อยแล้วขั้นสุดท้ายคือต้องทำการย้ายแกนเพราะระยะทางและองศาการยิงตัวใหม่นั้นเราพิจารณาที่จุด (0,0,0) ดังนั้นจึงต้องมีการนำค่าตำแหน่ง ณ จุดที่ตัวละครทำการยืนอยู่มาคำนวณโดยใช้สูตร

$$X_{bullet} = X_{position} + (v*t*\sin(NewDegreeHor*pi/180)$$

$$Y_{bullet} = Y_{position} + (v*t*\sin(DegreeVer*pi/180)$$

$$Z_{bullet} = Z_{position} + (v*t*\cos(NewDegreeHor*pi/180)$$

โดย v คือค่าความเร็วของกระสุน , t คือ เวลา และ pi มีค่าเป็น 3.1428 โดยประมาณ




**รูปที่ 3.31** flowchart diagram แสดงการเขียนโปรแกรมเพื่อตรวจสอบการยิงกระสุน

**3.2.1.11 การเคลื่อนย้ายวัตถุด้วยพิกัดเชิงขั้ว**

เป็นการกำหนดตำแหน่งของวัตถุหนึ่งไปไว้ยังตำแหน่งของตัวละคร และให้สามารถเคลื่อนที่ไปพร้อมกับตัวละคร โดยเสมือนว่าวัตถุดังกล่าวเป็นส่วนหนึ่งของตัวละคร ซึ่งจะใช้หลักการพิกัดเชิงขั้วทำการคำนวณทั้งแกนระนาบและแกนในแนวดิ่ง โดยในที่นี้ จะกล่าวถึงการย้ายตำแหน่งกระสุนไปไว้ยังตำแหน่งที่ปลายมือของตัวละคร

อัลกอริทึม

การคำนวณในแกนแนวดิ่ง(ระนาบ YZ)

- คำนวณหาค่ารัศมีจากกระสุนที่จะย้ายไปวางยังตำแหน่งปลายมือของตัวละคร ซึ่งคำนวณค่ารัศมีจากตำแหน่งกระสุนเทียบกับจุดศูนย์กลางของตัวละคร โดยใช้สูตรคำนวณหาระยะทางคือ

$$R_{yz}=SQRT((y_1-y_2)^2+(z_1-z_2)^2)$$

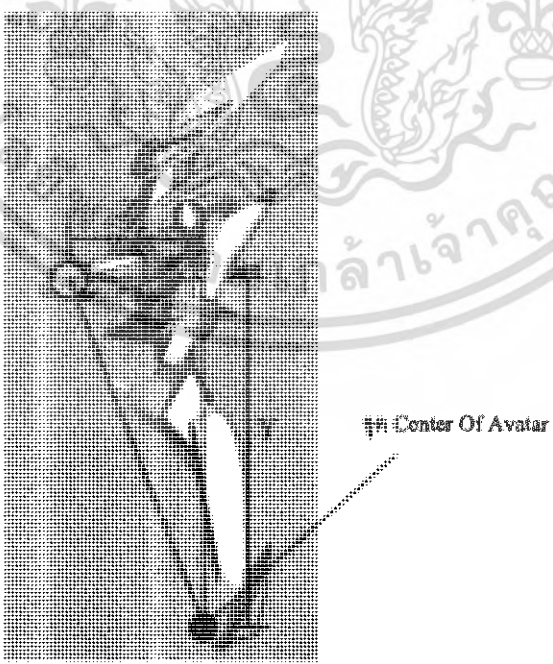
- คำนวณหาค่ามุมองศาในแนวแกนดิ่ง โดยนำค่าผลต่างระหว่างตำแหน่งกระสุนกับจุดศูนย์กลางของตัวละคร โดยใช้สูตรคำนวณหามุมองศาในแนวแกนดิ่งคือ

$$DegreeVer=180*(\tan^{-1}((y_1-y_2)/(z_1-z_2)))/\ pi$$

- นำมุมในแนวแกนดิ่งของตัวละครมารวมกับมุมที่ได้จากการคำนวณในแนวแกนดิ่ง จากนั้นทำการหาค่าY และ Z ค่าใหม่ของตำแหน่งกระสุน โดยใช้สูตรคำนวณพิกัดเชิงขั้ว

$$Z_{ใหม่} = Z_{เก่า} + R_{yz} * COS(NewDegreeVer)$$

$$Y_{ใหม่} = Y_{เก่า} + R_{yz} * SIN(NewDegreeVer)$$




**รูปที่ 3.32** แสดงการคำนวณพิกัดเชิงขั้วในแกนแนวดิ่ง

การคำนวณในแกนแนวระนาบ(ระนาบ XZ)

- คำนวณหาค่ารัศมีจากกระสุนที่จะย้ายไปวางยังตำแหน่งปลายมือของตัวละคร ซึ่งคำนวณค่ารัศมีจากตำแหน่งกระสุนเทียบกับจุดศูนย์กลางของตัวละคร โดยใช้สูตรคำนวณหาระยะทางคือ

$$R_{xz}=SQRT((x_1-x_2)^2+(z_1-z_2)^2)$$

โดยค่าZที่นำมาคำนวณ เป็นค่าZใหม่ที่เกิดจากการการคำนวณในแนวแกนดิ่ง

- คำนวณหาค่ามุมองศาในแนวแกนระนาบ โดยนำค่าผลต่างระหว่างตำแหน่งกระสุนกับจุดศูนย์กลางของตัวละคร โดยใช้สูตรคำนวณหามุมองศาในแนวแกนดิ่งคือ

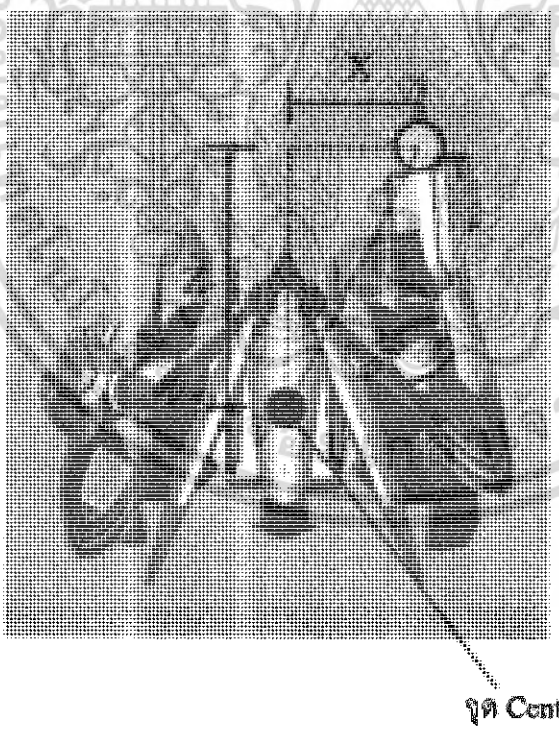
$$DegreeHor=180*(\tan^{-1}((z_1-z_2)/(x_1-x_2)))/\ pi$$

โดยค่าZที่นำมาคำนวณ เป็นค่าZใหม่ที่เกิดจากการการคำนวณในแนวแกนดิ่ง

- นำมุมในแนวแกนระนาบของตัวละครมารวมกับมุมที่ได้จากการคำนวณในแนวแกนดิ่ง จากนั้นทำการหาค่าX และ Z ค่าใหม่ของตำแหน่งกระสุน โดยใช้สูตรคำนวณพิกัดเชิงขั้ว

$$Z_{ใหม่} = Z_{เก่า} + R_{xz} * SIN(NewDegreeHor)$$

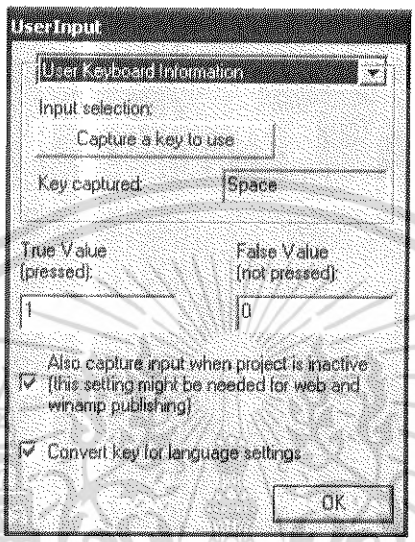
$$X_{ใหม่} = X_{เก่า} + R_{xz} * COS(NewDegreeHor)$$




**รูปที่ 3.33** แสดงการคำนวณพิกัดเชิงขั้วในแกนระนาบ

**3.2.1.12 การรับค่าจากแป้นพิมพ์และเมาส์**

ฟังก์ชันที่ใช้ในการติดต่อกับแป้นพิมพ์และเมาส์เพื่อที่จะทำการรับค่าขณะที่ผู้เล่นได้ทำการกดปุ่มต่างๆ จะเรียกว่า UserInput โดยสามารถเลือกรูปแบบการรับค่าข้อมูลได้หลายรูปแบบ เช่น การรับค่าข้อมูลจากแป้นพิมพ์, การรับค่าข้อมูลจากเมาส์, การรับค่าข้อมูลจากแท่งหรรษา เป็นต้น



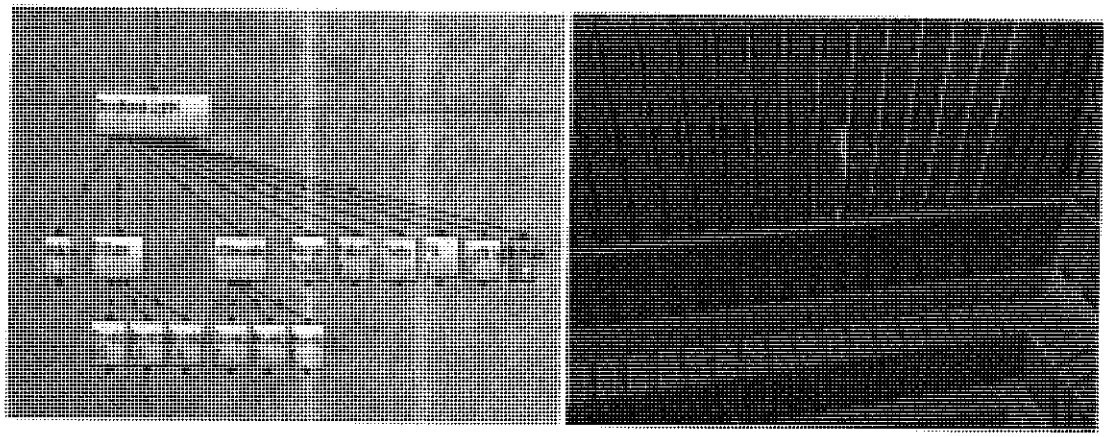

**รูปที่ 3.34** แสดงฟังก์ชันการรับค่าจากแป้นพิมพ์และเมาส์

**3.2.1.13 การกำหนดทิศทางของแสง**

การจัดแสงโดยใช้วัตถุกำเนิดแสง จะต้องมีตัวกำเนิดแสงสร้างแสงไปกระทบกับวัตถุสามมิติที่อยู่ในฉาก ทำให้สามารถมองเห็นวัตถุสามมิติดังกล่าวได้ เราสามารถย้ายหรือหมุนตัววัตถุกำเนิดแสงไปวางไว้ที่ตำแหน่งใดตำแหน่งหนึ่งได้

ประเภทของแสง

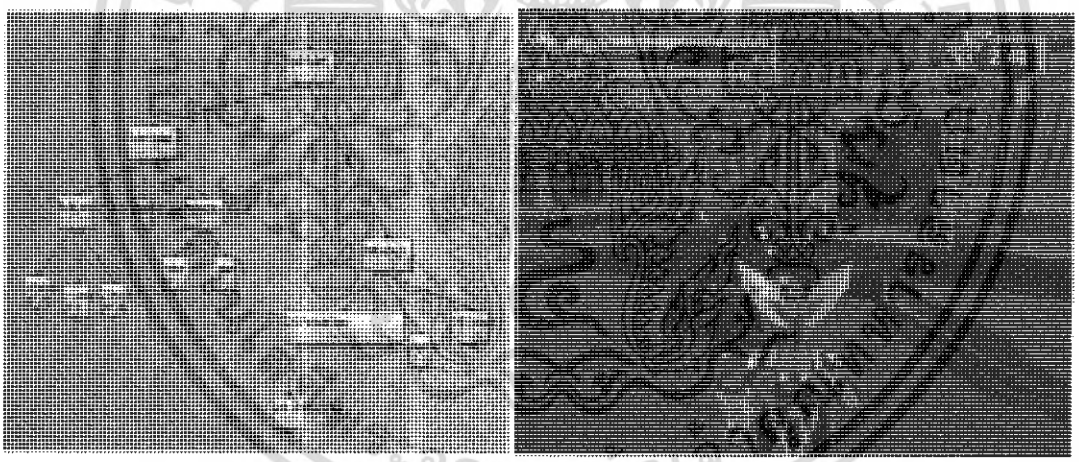
- Point Light ทำหน้าที่กระจายแสงไปกระทบวัตถุที่อยู่รอบตัวของแสง
- Spot Light จะถูกกำหนดขอบเขตของแสงด้วยองศาของกรวยด้านนอกและด้านใน
- Directional Light เป็นแสงที่ขนานที่มีทิศทางเดียวตามทิศทางของลูกศรของวัตถุ



**รูปที่ 3.35** แสดงโหนดของแสงแบบDirectional Light (ซ้าย) และการกำหนดทิศทางของแสงแบบ Directional Light (ขวา)

**3.2.1.14 การใส่เอฟเฟก**

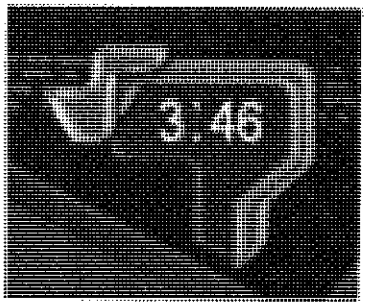
วัตถุที่นำมาใช้เป็นเอฟเฟกภายในเกม เช่น กระสุน ระเบิด เป็นต้น จะเป็นเอฟเฟกที่มีอยู่ภายในQuest3D เอนจิน โดยเราสามารถที่จะกำหนดสีของเอฟเฟกที่จะแสดง และกำหนดขนาดรูปร่างของวัตถุเอฟเฟกได้ตามที่ต้องการ



**รูปที่ 3.36** แสดงโหนดของกระสุนของตัวละคร (ซ้าย) และกำหนดขนาดรูปร่างและสีของกระสุน (ขวา)

**3.2.1.15 การกำหนดเวลาภายในเกม**

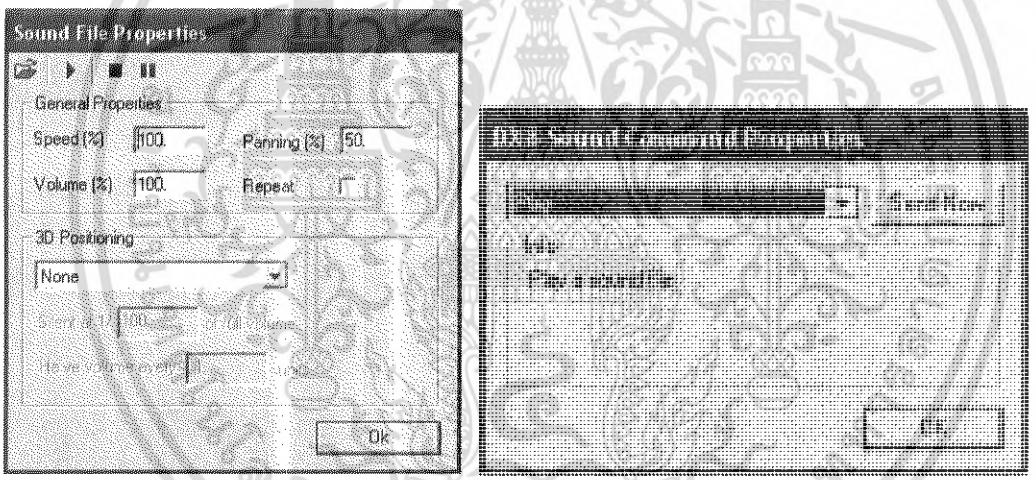
เวลาภายในเกมเป็นปัจจัยสำคัญที่จะเป็นตัวตัดสินว่าผู้เล่นจะชนะหรือแพ้ โดยเวลาที่สร้างขึ้นมานั้นจะใช้ฟังก์ชันExpression Value และทำการแปลงจากค่าผลลัพธ์ที่เป็นตัวเลขให้แสดงในรูปแบบของตัวอักษร




**รูปที่ 3.37** การกำหนดรูปแบบเวลาภายในเกม

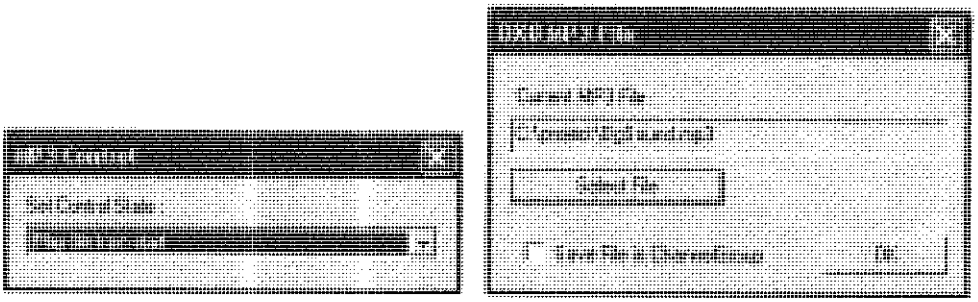
**3.2.1.16 การใส่เสียงและไฟล์ภาพยนตร์**

เสียงเอฟเฟกที่ใช้ภายในระบบเกม จะใช้ไฟล์เสียงที่เป็นแบบ.wav โดยจะใช้ฟังก์ชันที่เรียกว่าSound File Channel เพื่อทำการนำไฟล์เสียงที่เป็นรูปแบบ.wavเข้ามาในQuest3D เอนจิน และฟังก์ชันที่ชื่อว่า Sound Command channel เป็นคำสั่งที่ใช้ควบคุมการทำงานไฟล์เสียงที่เป็นรูปแบบ.wav



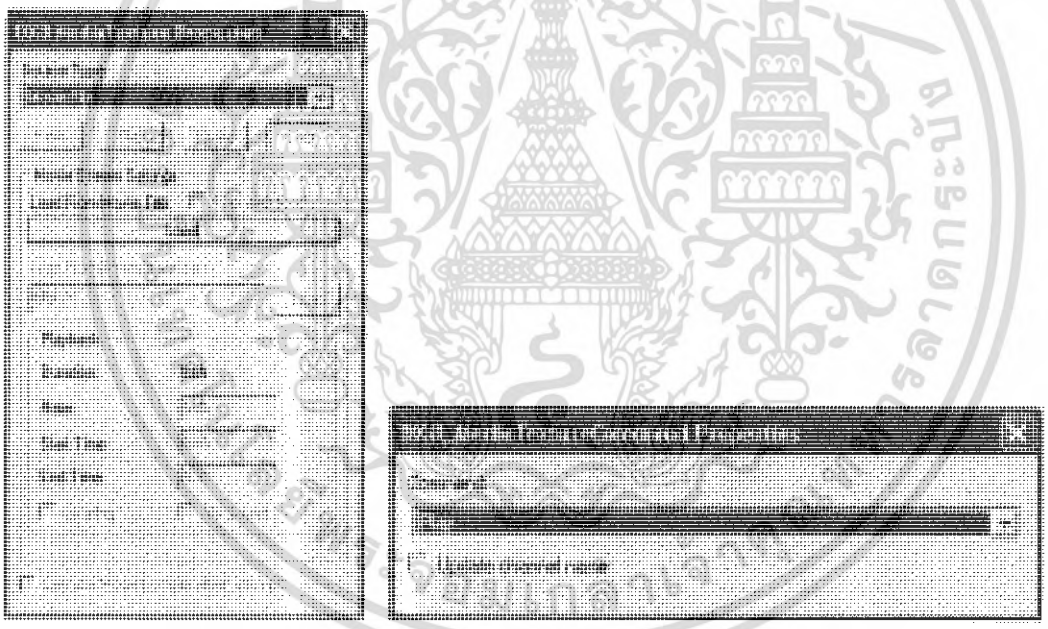

**รูปที่ 3.38** แสดงฟังก์ชันSound File Channel (ซ้าย) และฟังก์ชัน Sound Command channel (ขวา)

เสียงประกอบฉากที่ใช้ภายในระบบเกม จะใช้ไฟล์เสียงที่เป็นแบบ.mp3 โดยจะใช้ฟังก์ชันที่เรียกว่า MP3 File เพื่อทำการนำไฟล์เสียงที่เป็นรูปแบบ.wavเข้ามาในQuest3D เอนจิน และฟังก์ชันที่ชื่อว่า MP3 Control เป็นคำสั่งที่ใช้ควบคุมการทำงานไฟล์เสียงที่เป็นรูปแบบ.mp3



**รูปที่ 3.39** แสดงฟังก์ชัน MP3 Control (ซ้าย) และฟังก์ชัน MP3 File (ขวา)

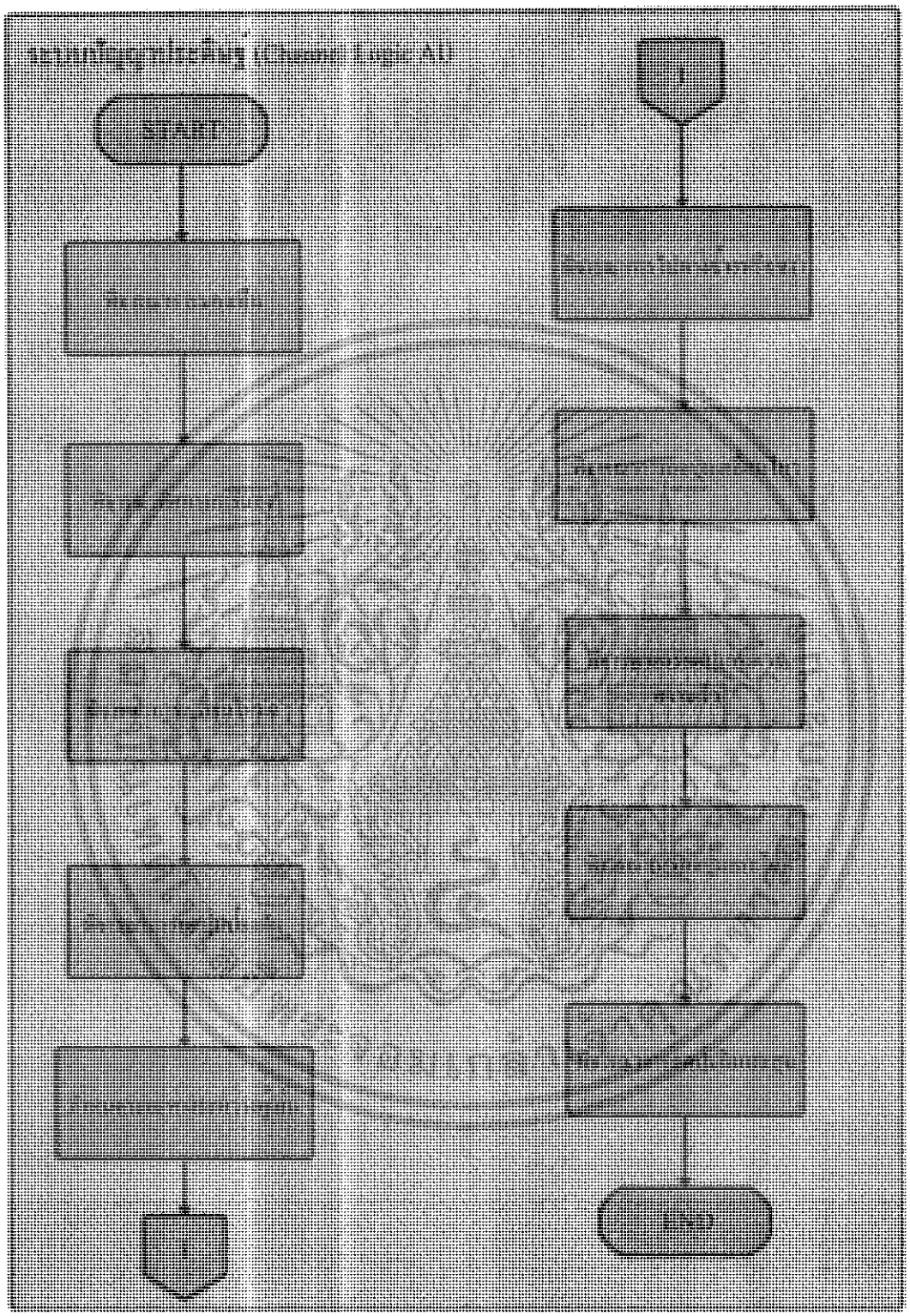
ไฟล์ภาพยนตร์ที่นำเข้ามาภายในระบบเกม จะใช้สกุลไฟล์.avi เพราะQuest3D เอนจิน สามารถรองรับไฟล์ภาพยนตร์.aviและคุณภาพของไฟล์ภาพยนตร์แบบ.avi จะให้ภาพที่คมชัดและดูเหมือนจริงฟังก์ชันที่ใช้ในการนำเข้าไฟล์ภาพยนตร์จะใช้ฟังก์ชันที่ชื่อว่าMedia Textureและฟังก์ชันที่ใช้ในการควบคุมการทำงานของไฟล์ภาพยนตร์ให้แสดงภายในเวลาที่กำหนดจะใช้ฟังก์ชันที่ชื่อว่า Media Texture Command



**รูปที่ 3.40** แสดงฟังก์ชันMedia Texture (ซ้าย) และฟังก์ชันMedia Texture Command (ขวา)

### 3.2.2 การสร้างระบบปัญญาประดิษฐ์

ระบบปัญญาประดิษฐ์แบ่งการทำงานออกเป็น 10 ส่วนหลัก



**รูปที่ 3.41** การทำงานต่างๆของระบบปัญญาประดิษฐ์

ซึ่งได้แก่

1. การพิจารณาการมองเห็น
2. การพิจารณาทิศทางการมอง
3. การพิจารณาการเปลี่ยนโหมด
4. การพิจารณาการกดปุ่มป้องกัน
5. การกำหนดระยะห่างระหว่างผู้เล่น
6. การพิจารณาการไปทางซ้ายหรือขวา
7. การพิจารณาการกดปุ่มเคลื่อนไหว
8. การพิจารณาการกดปุ่มพุ่งด้วยความเร็ว
9. การพิจารณาการกดปุ่มกระโดด
10. การพิจารณาการกดปุ่มยิงกระสุน

การทำงานในแต่ละส่วนของระบบปัญญาประดิษฐ์นั้นเป็นการทำงานเชิงโครงสร้างไล่จากข้างบนลงมาสู่ข้างล่าง ที่ได้จัดลำดับการทำงานไว้ดังนี้ก็เพื่อความเหมาะสมต่อระบบงาน การทำงานบางส่วนอาจจำเป็นที่จะต้องใช้ผลลัพธ์จากส่วนที่ทำงานอยู่เหนือกว่า ทำให้ส่วนนั้นๆจำเป็นที่จะต้องทำงานในลำดับที่ต่ำกว่า นอกจากนี้ ระบบการทำงานแต่ละส่วนยังจะประกอบไปด้วยตัวแปรควบคุมและตัวแปรเงื่อนไข ซึ่งถือได้ว่าเป็นหัวใจในการคำนวณผลลัพธ์ของระบบอีกด้วย

ตัวแปรควบคุมนั้นโดยปกติแล้วจะเชื่อมโยงกับส่วนการทำงานมากกว่าหนึ่งส่วน ซึ่งจะคอยทำหน้าที่ให้การควบคุมการทำงาน เช่น บอกให้ทราบว่าส่วนนี้ทำงานได้ หรือให้หยุดการทำงานลงก่อน ส่วนตัวแปรเงื่อนไขนั้นจะทำหน้าที่เป็นสภาวะให้กับระบบ ซึ่งระบบแต่ละส่วนจะมีตัวดักจับสภาวะของตัวแปรเงื่อนไขเพื่อที่จะตัดสินใจว่าควรจะทำเช่นไรในสภาวะนั้นๆ

ต่อไปจะได้แจกแจงการทำงานในแต่ละส่วนของระบบปัญญาประดิษฐ์

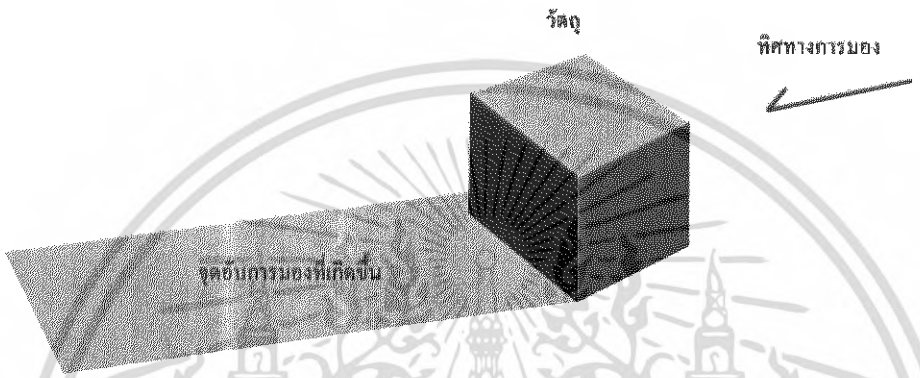
**3.2.2.1 การพิจารณามองเห็น**

***จุดประสงค์การทำงาน:*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถวิเคราะห์ได้ว่ามองเห็นตัวละครของผู้เล่นอยู่หรือไม่

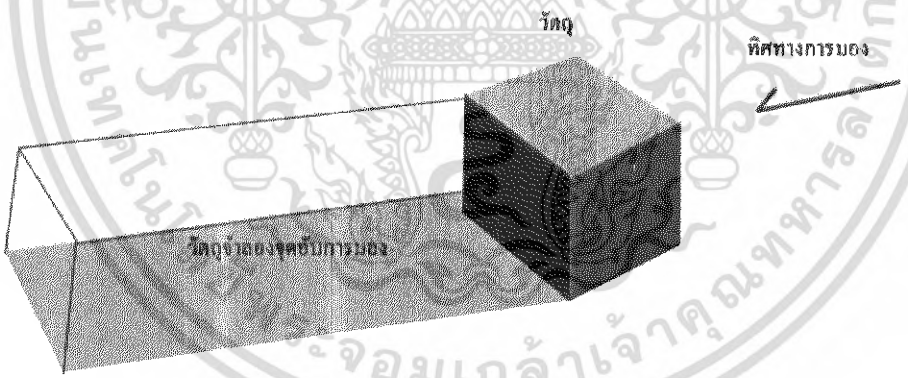
***ช่วงเวลาที่ระบบจะทำงาน:*** ระบบจะทำงานตลอดเวลา

เป็นส่วนที่สร้างขึ้นเพื่อให้ระบบปัญญาประดิษฐ์ มีความใกล้เคียงกับมนุษย์มากขึ้น โดยระบบปัญญาประดิษฐ์ เมื่อถูกสิ่งกีดขวางบังการมองไปยังผู้เล่น จะทำให้ระบบปัญญาประดิษฐ์ เข้าสู่สถานะการมองไม่เห็น ทั้งนี้ก็เพื่อให้ผู้เล่นสามารถแอบอ้อมไปดักหลังตัวละครของระบบปัญญาประดิษฐ์ได้ ไม่ต่างไปจากการต่อสู้กับบุคคลจริงๆ

ตามหลักการมองเห็นโดยทั่วไป เมื่อมีการมองวัตถุชิ้นหนึ่งที่ด้านหน้า ก็จะทำให้เกิดจุดอับของสายตาในที่บริเวณด้านหลังของวัตถุนั้นเป็นแนวยาวออกไป หรือในทางกลับกันหากมองวัตถุจากทางด้านหลังก็จะทำให้เกิดจุดอับบริเวณด้านหน้าของวัตถุชิ้นนั้น หลักการทำงานของส่วนนี้จึงได้ใช้การจำลองวัตถุ ซึ่งทำหน้าที่เหมือนจุดอับของกล่องแต่ละกล่องขึ้นมา โดยวัตถุจำลองจุดอับการมองนั้นจะหมุนไปตามทิศทางการมองของตัวละครปัญญาประดิษฐ์ที่มุ่งไปยังกล่องแต่ละกล่อง ซึ่งวัตถุจำลองจุดอับนั้นจะมีจุดหมุนอยู่ที่ตำแหน่งกึ่งกลางวัตถุซึ่งเป็นตัวกล่องจริงๆ




**รูปที่ 3.42** จุดอับการมองที่เกิดขึ้นจากทิศทางการมอง

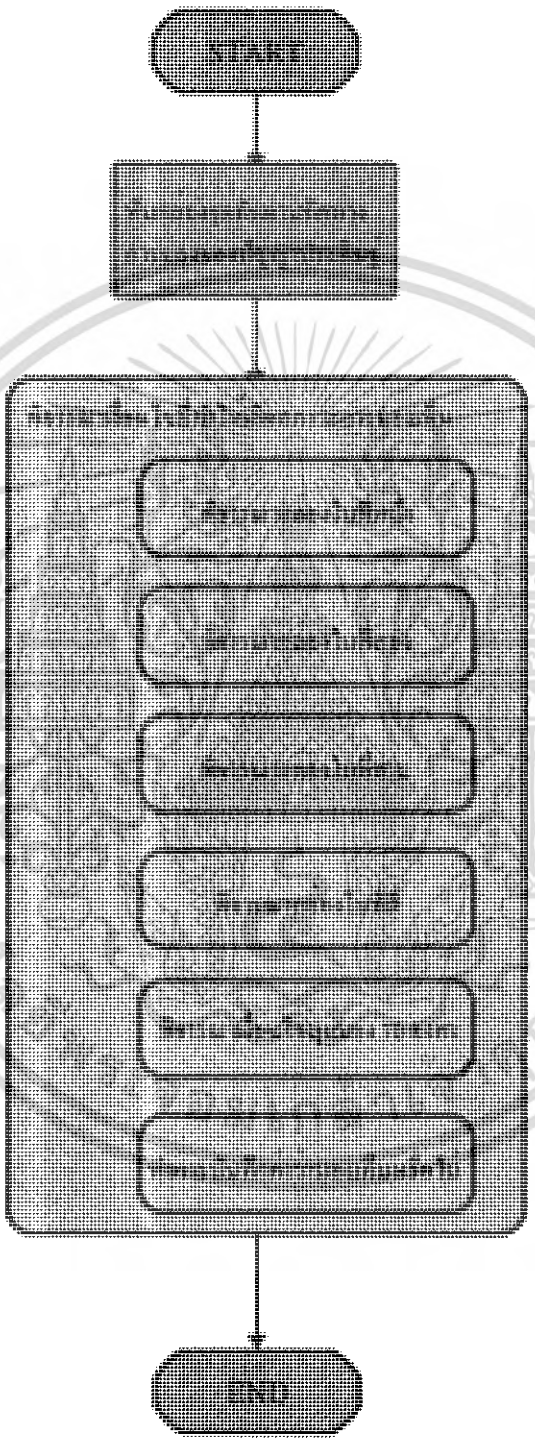



**รูปที่ 3.43** วัตถุจำลองจุดอับการมอง

จากภาพ 3.2.2 (3) จะเห็นได้ว่าพื้นที่จุดอับการมองนั้นจะถูกจำลองด้วยวัตถุสี่เหลี่ยม ถ้าหาก ผู้เล่นมีการชน หรือเข้ามาอยู่ภายในพื้นที่ของวัตถุจำลองจุดอับนี้ ระบบปัญญาประดิษฐ์ก็จะตรวจสอบแล้วทราบได้ว่า ไม่สามารถที่จะมองเห็นผู้เล่นในขณะนั้นได้ ก็จะทำการสร้างจุดคาดการณ์ตำแหน่งที่ตัวละครผู้เล่นน่าจะยืนอยู่ขึ้นมา แทนที่จะตรวจจับตำแหน่งจริงๆที่ตัวละครผู้เล่นยืนอยู่ในขณะนั้น ซึ่งจะได้กล่าวถึงในส่วนการทำงานถัดไป

ฉะนั้นการตรวจสอบว่าระบบปัญญาประดิษฐ์นั้นจะมองเห็นผู้เล่นหรือไม่ จึงจะตรวจสอบจากจุดอับนี้เป็นสำคัญ

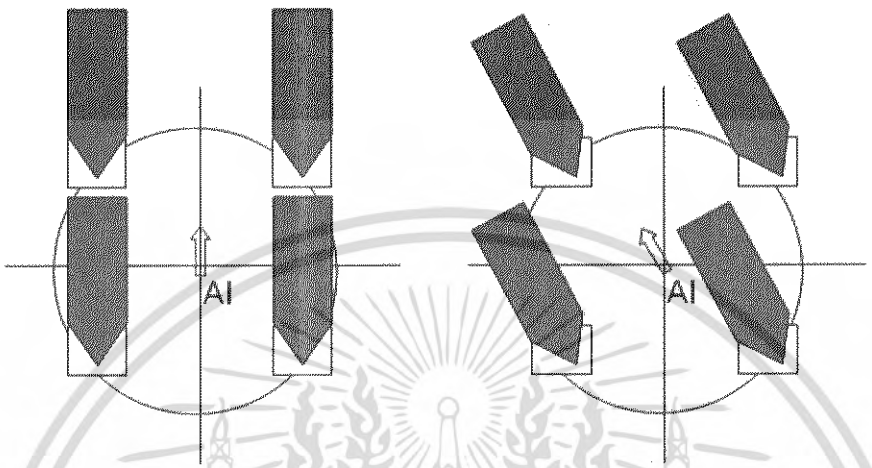
**ส่วนพิจารณาการมองเห็น**



**รูปที่ 3.44** ระบบภายในของการพิจารณาการมองเห็น

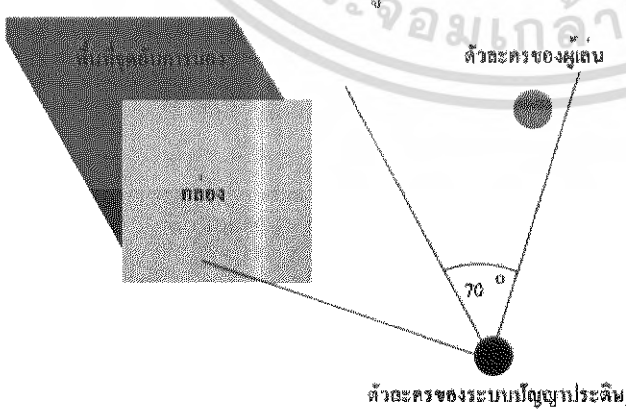
***การทำงาน***

- การทำงานนั้นเริ่มจาก การหันองศาของวัตถุจำลองจุดอับการมองให้ไปในทิศทางเดียวกันกับทิศทางการมองของตัวละครระบบปัญญาประดิษฐ์ เพื่อที่จะทราบถึงพื้นที่ที่เป็นจุดอับในการมอง

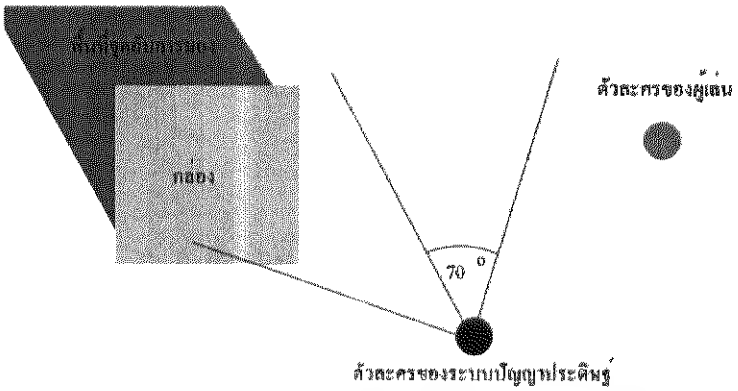



**รูปที่ 3.45** การหมุนวัตถุจำลองจุดอับการมองเห็น ไปในทิศทางเดียวกันกับที่ระบบปัญญาประดิษฐ์มองไป

- จากนั้นจึงทำการพิจารณาว่า ตัวละครของผู้เล่นนั้นมีการชน หรือเข้าไปอยู่ในวัตถุจำลองจุดอับการมองเห็นหรือไม่ โดยจะทำการตรวจสอบจุดอับของกล่องทั้งสี่กล่อง พิจารณาร่วมกันกับปัจจัยที่ว่า มีกล่องใดบ้างที่อยู่ภายใต้ขอบเขตการมองเห็นของระบบปัญญาประดิษฐ์
- หากทราบว่าผู้เล่นไม่ได้ถูกบังโดยกล่อง และไม่ได้ยืนอยู่ในองศาที่มากกว่า 70 องศา เมื่อเทียบกับทิศทางการมองของระบบปัญญาประดิษฐ์ จึงจะสรุปได้ว่าสามารถมองเห็นผู้เล่นได้




**รูปที่ 3.46** ตำแหน่งของผู้เล่นที่อยู่ไม่เกิน 70 องศา การมองของระบบปัญญาประดิษฐ์




**รูปที่ 3.47** ตำแหน่งของผู้เล่นที่อยู่เกิน 70 องศา การมองของระบบปัญญาประดิษฐ์

- หากไม่ตรงตามกรณีที่ได้กำหนดไว้ ก็สามารถสรุปได้ว่าไม่สามารถมองเห็นผู้เล่นในขณะนั้นได้
- เมื่อทราบแล้วว่ามองเห็นผู้เล่นอยู่หรือไม่ก็จะทำการเก็บผลลัพธ์ไว้ในตัวแปรที่ชื่อว่า StatusVisible เพื่อให้ส่วนอื่นได้นำผลลัพธ์ที่ได้นี้ไปประมวลผลต่อไป

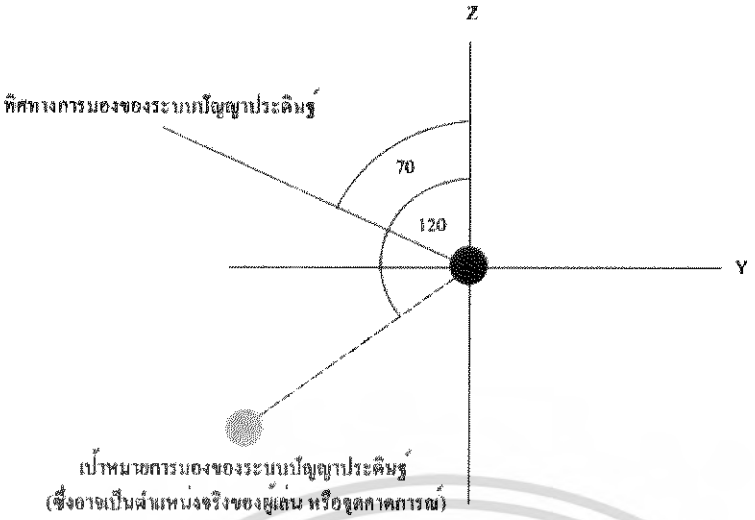
**3.2.2.2 การพิจารณาทิศทางการมอง**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์หันทิศทางไปยังตำแหน่งของผู้เล่นได้อย่างถูกต้อง

***ช่วงเวลาที่ระบบจะทำงาน :*** ระบบทำงานตลอดเวลา

การที่เราจะทราบได้ว่า ระบบปัญญาประดิษฐ์ควรจะหันไปยังทิศทางใดนั้น จำเป็นที่จะต้องคำนวณจากตำแหน่งที่ผู้เล่นอยู่ เพราะปกติแล้วระบบปัญญาประดิษฐ์ควรจะหันไปหาผู้เล่นเสมอ เพื่อที่จะสามารถหลบหลีกกระสุนที่ผู้เล่นยิงมา หรือกระทั่งสามารถยิงกระสุนตอบโต้ กลับไปสู่ตำแหน่งที่ผู้เล่นอยู่ได้อย่างถูกต้อง แต่จะมีกรณีที่ระบบปัญญาประดิษฐ์ไม่สามารถที่จะมองเห็นผู้เล่นในขณะนั้นได้ จึงจะกำหนดให้จุดคาดการณ์เป็นตำแหน่งของผู้เล่นแทน ซึ่งส่วนการทำงานนี้ก็จะรับผิดชอบในการกำหนดจุดคาดการณ์แทนด้วย

การทำงานในส่วนนี้จะทำการเพิ่มหรือลดค่าองศาการมองของระบบปัญญาประดิษฐ์ เพื่อให้เกิดความใกล้เคียงหรือตรงกับตำแหน่งผู้เล่น โดยจะต้องมีการคำนวณองศาของตำแหน่งผู้เล่น องศาของตำแหน่งผู้เล่นก็คือค่าของมุมซึ่ง ถ้าระบบปัญญาประดิษฐ์หันด้วยทิศทางที่มีค่าองศาเท่ากับองศาของตำแหน่งผู้เล่นแล้วละก็ จะมองเห็นตรงกับตัวละครของผู้เล่นพอดี



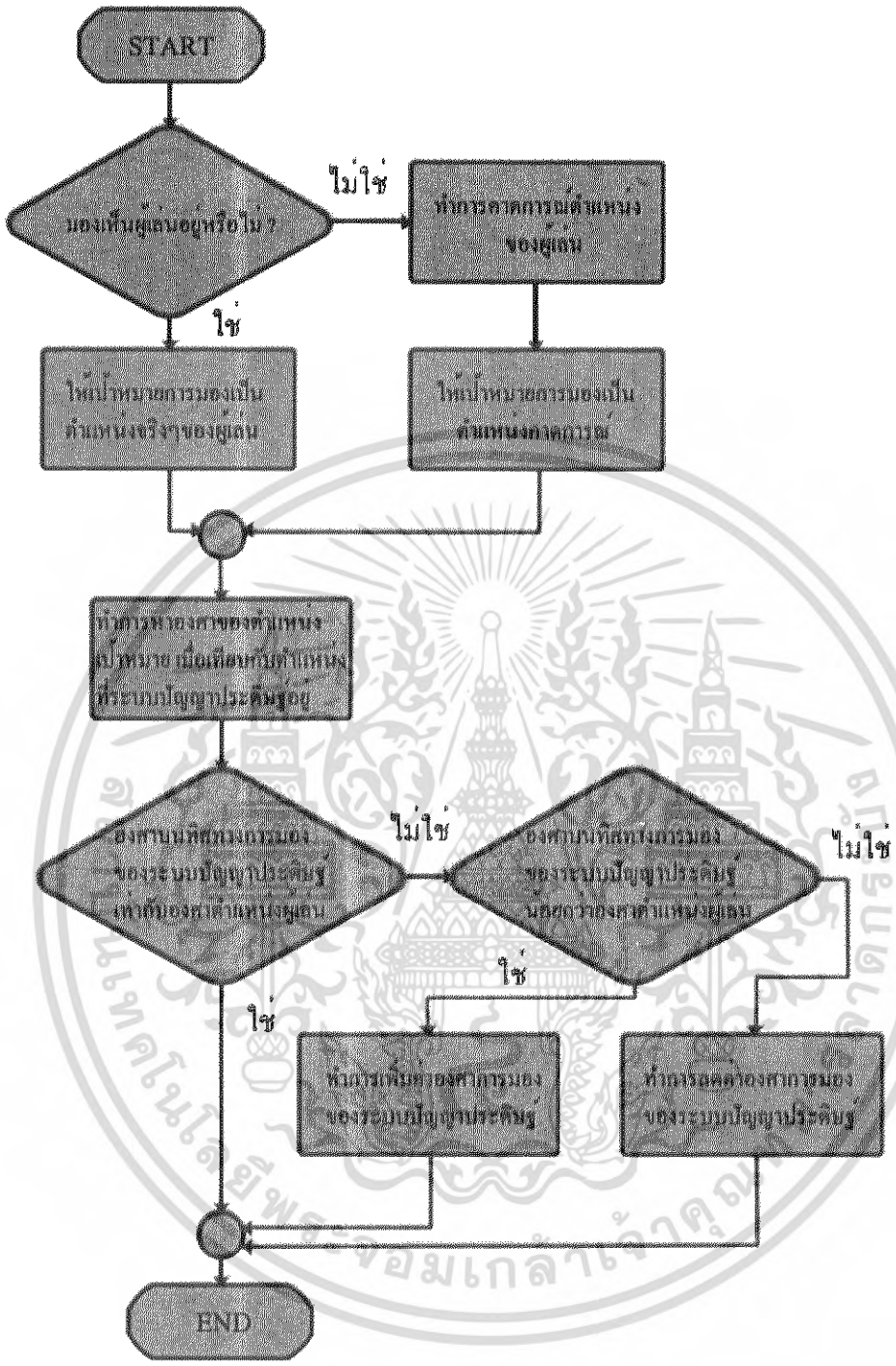

**รูปที่ 3.48** แสดงองศาการหันของระบบปัญญาประดิษฐ์ ซึ่งมีค่าเท่ากับ 70 องศา และองศาตำแหน่งของผู้เล่นซึ่งมีค่าเท่ากับ 120 องศา

จากภาพ 3.2.2 (8) เราสามารถบอกได้ว่า ตัวละครของระบบปัญญาประดิษฐ์ยังหันไปในทิศทางที่ไม่ตรงกันกับตัวละครของผู้เล่น ซึ่งระบบปัญญาประดิษฐ์จะต้องทำการหันเพิ่มขึ้นอีก 50 องศา ฉะนั้นระบบควบคุมการหันก็จะทำการบวกองศาการหันเพิ่มเข้าไปให้กับตัวละครของระบบปัญญาประดิษฐ์จนกว่าจะมีค่าเทียบเท่า 120 องศา

ทั้งนี้การเพิ่มค่าองศาการมองของระบบปัญญาประดิษฐ์นั้นจะไม่เพิ่มขึ้นให้เท่ากับองศาเป้าหมายในทีเดียว แต่จะค่อยๆเพิ่มขึ้นไปเรื่อยๆ ก็เพื่อให้เกิดการหันที่ราบเรียบและสวยงาม ทั้งนี้ ความเร็วในการหันได้ถูกกำหนดไว้ ให้มีความเร็วมากเกินกว่าที่ผู้เล่นจะหนีได้ทัน ฉะนั้น ภายในเวลาที่ไม่มากจนเกินไปนัก ระบบปัญญาประดิษฐ์ก็จะสามารถหันทิศทางการมองให้ตรงกับตัวละครของผู้เล่นได้อย่างสมบูรณ์

ต่อไปจะได้อธิบายถึงการทำงานของระบบ

ส่วนพิจารณาทิศทางการมอง

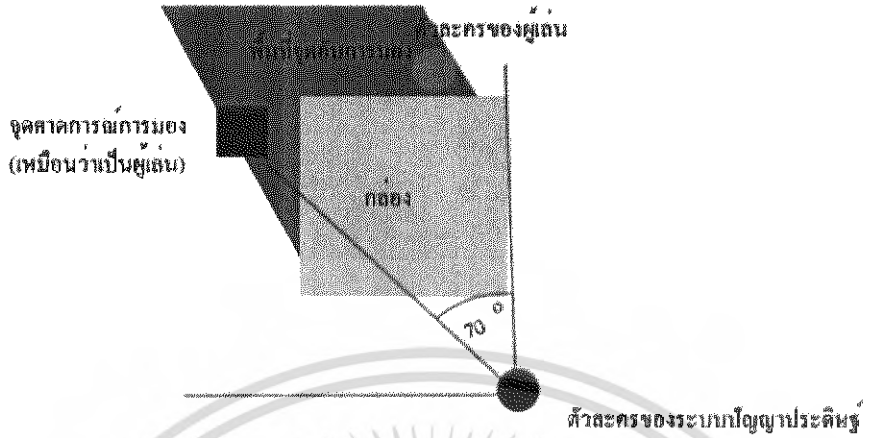



**รูปที่ 3.49** ระบบภายในของการกำหนดทิศทางการมอง

*การทำงาน*

- การทำงานเริ่มจากการตรวจสอบว่า ระบบปัญญาประดิษฐ์อยู่ในสถานะมองเห็นหรือมองไม่เห็น

- ถ้าอยู่ในสถานะมองไม่เห็น ก็ให้สร้างจุดคาดการณ์ตำแหน่งผู้เล่นขึ้นแทนจุดที่ตัวละครผู้เล่นอยู่จริงๆ




**รูปที่ 3.50** การมองไปยังตำแหน่งคาดการณ์แทนที่การมองไปยังตำแหน่งของผู้เล่น ขณะอยู่ในสถานะมองไม่เห็นตัวผู้เล่น

- ถ้าหากอยู่ในสถานะที่มองเห็นตัวละครของผู้เล่น ก็ให้ใช้ตำแหน่งจริงๆของผู้เล่น
- ทำการหาองศาที่ผู้เล่นยืนอยู่ เมื่อเทียบกับตำแหน่งตัวละครของระบบปัญญาประดิษฐ์
- ทำการเปรียบเทียบองศาของตำแหน่งผู้เล่น กับองศาบนทิศทางการมองของระบบปัญญาประดิษฐ์
- ถ้าองศาบนทิศทางการมองของระบบปัญญาประดิษฐ์มีค่ามากกว่าองศาของตำแหน่งผู้เล่น ให้ลดองศาบนทิศทางการมองให้มีค่าน้อยลง
- ถ้าองศาบนทิศทางการมองของระบบปัญญาประดิษฐ์มีค่าน้อยกว่าองศาของตำแหน่งผู้เล่น ให้เพิ่มองศาบนทิศทางการมองให้มีค่ามากขึ้น

**3.2.2.3 การพิจารณาการเปลี่ยนโหมด**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์มีความสามารถในการตัดสินใจว่าควรเปลี่ยนโหมดในเวลาและสถานการณ์เช่นไร เพื่อให้เกิดประโยชน์สูงสุดในขณะต่อสู้

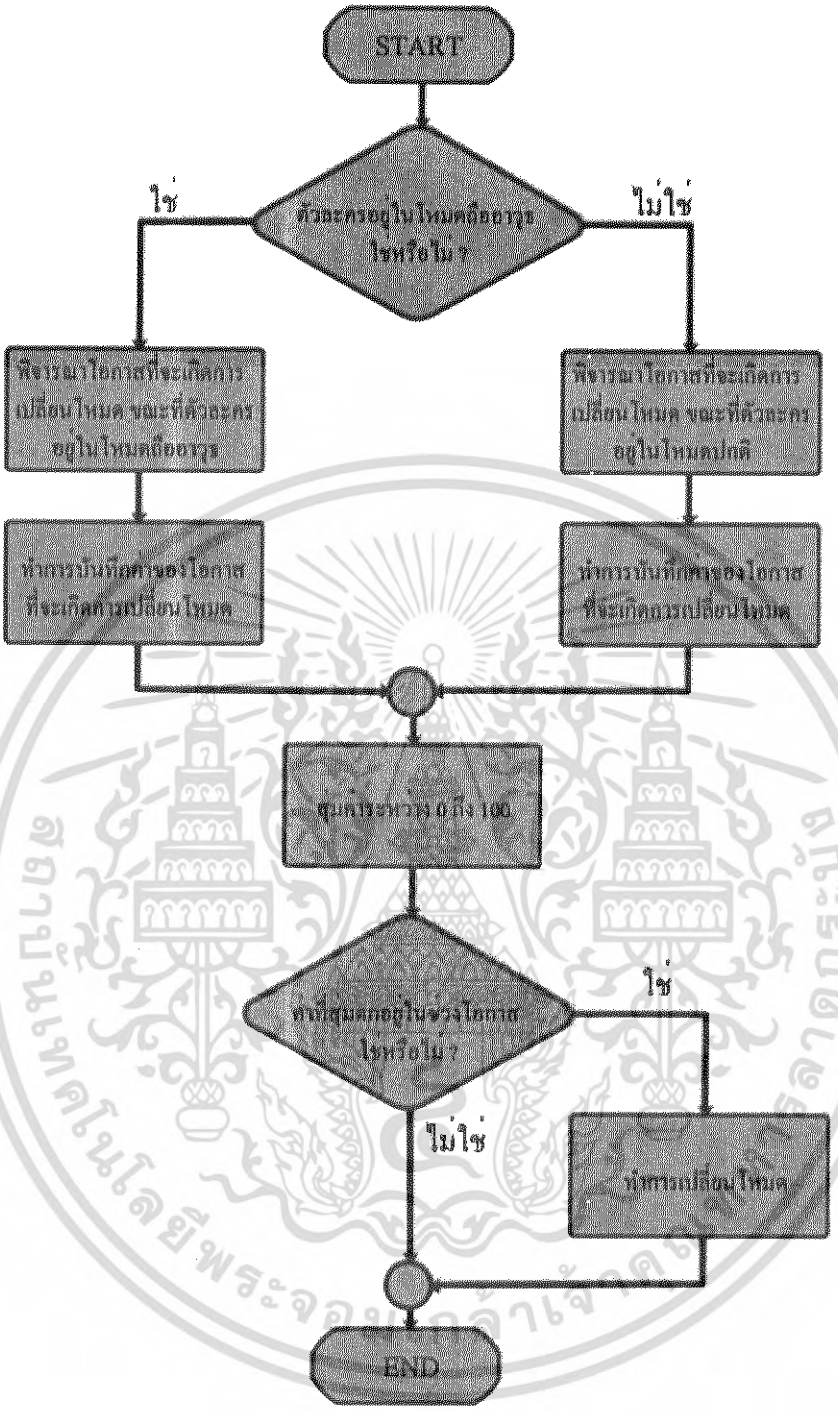
***ช่วงเวลาที่ระบบจะทำงาน :*** ระบบจะทำงานทุกๆ 10 วินาที หรือราว 5 วินาทีในขณะที่ระบบปัญญาประดิษฐ์เหลือพลังชีวิตน้อย

ระบบภายในเกมได้ถูกออกแบบมาให้มีการเปลี่ยนโหมดได้ นั่นคือ โหมดปกติซึ่งไม่ได้ถืออาวุธ และโหมดถืออาวุธ ซึ่งทั้งสองโหมดนั้นจะมีข้อดีและข้อเสียที่แตกต่างกัน โหมดปกตินั้นจะไม่สามารถโจมตีได้ แต่พลังงานของหุ่นยนต์จะเพิ่มขึ้นเร็วกว่าโหมดถืออาวุธถึง 2 เท่า ทำให้ถ้าหากพลังงานของหุ่นยนต์เหลืออยู่น้อย ก็ควรจะเปลี่ยนโหมดของหุ่นยนต์กลับไปเป็นโหมดปกติเพื่อเร่งอัตราการเพิ่มของพลังงาน และถ้าหากว่าพลังงานเหลืออยู่มาก ก็ควรจะเปลี่ยนโหมดกลับไปเป็นโหมดถืออาวุธเพื่อที่จะต่อสู้ได้ต่อไป จึงต้องมีการกำหนดรูปแบบและวิธีการพิจารณาให้กับระบบปัญญาประดิษฐ์ว่าควรจะมีการเปลี่ยนโหมดในสถานการณ์และเวลาเช่นใดบ้าง

หลักการทำงานนั้นพิจารณาจากสภาวะพลังงานที่เหลืออยู่ของระบบปัญญาประดิษฐ์เป็นสำคัญ การเปลี่ยนโหมดจึงมีความสัมพันธ์โดยตรงกับพลังงาน ยิ่งพลังงานเหลือมาก แล้วตัวละครอยู่ในโหมดอาวุธ โอกาสที่จะเปลี่ยนโหมดมาเป็นโหมดปกติก็จะมีน้อย ขณะที่ ถ้าตัวละครอยู่ในโหมดปกติ โอกาสที่จะเปลี่ยนโหมดไปเป็นโหมดถืออาวุธนั้นก็จะมีสูง ในทางกลับกัน ยิ่งพลังงานเหลือน้อย โอกาสที่ตัวละครจะเปลี่ยนจากโหมดปกติไปสู่โหมดถืออาวุธนั้นก็จะมีน้อย และโอกาสที่ตัวละครจะเปลี่ยนจากโหมดถืออาวุธไปสู่โหมดปกตินั้นก็จะมีมาก

ต่อไปจะได้อธิบายถึงการทำงานของระบบ

การพิจารณาการเปลี่ยนโหมด




**รูปที่ 3.51** การพิจารณาการเปลี่ยนโหมด

***การทำงาน***

- การทำงานเริ่มจากการตรวจสอบว่าตัวละครของระบบปัญญาประดิษฐ์นั้นอยู่ในโหมดอะไร

- ถ้าหากอยู่ในโหมดปกติ ก็จะทำการพิจารณาโดยอิงหลักที่ว่า ยิ่งพลังงานเหลืออยู่มากเท่าไหร่ โอกาสที่จะเกิดการเปลี่ยนโหมดก็ยิ่งมากขึ้นเท่านั้น แล้วจึงทำการบันทึกค่าโอกาสที่จะเกิดการเปลี่ยนโหมดเอาไว้
- ถ้าหากอยู่ในโหมดถืออาวุธ ก็จะทำการพิจารณาโดยอิงหลักที่ว่า ยิ่งพลังงานเหลืออยู่น้อยเท่าไหร่ โอกาสที่จะเกิดการเปลี่ยนโหมดก็ยิ่งมากขึ้นเท่านั้น แล้วจึงทำการบันทึกค่าโอกาสที่จะเกิดการเปลี่ยนโหมดเอาไว้
- ทำการสุ่มค่า หากตกในช่วงค่าที่กำหนดไว้ ก็แสดงว่า ต้องทำการเปลี่ยนโหมดตัวละครของระบบปัญญาประดิษฐ์

**3.2.2.4 การพิจารณาการกดปุ่มป้องกัน**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถที่จะป้องกันกระสุนของฝ่ายผู้เล่น ที่กำลังจะโดนตัวละครของระบบปัญญาประดิษฐ์ได้

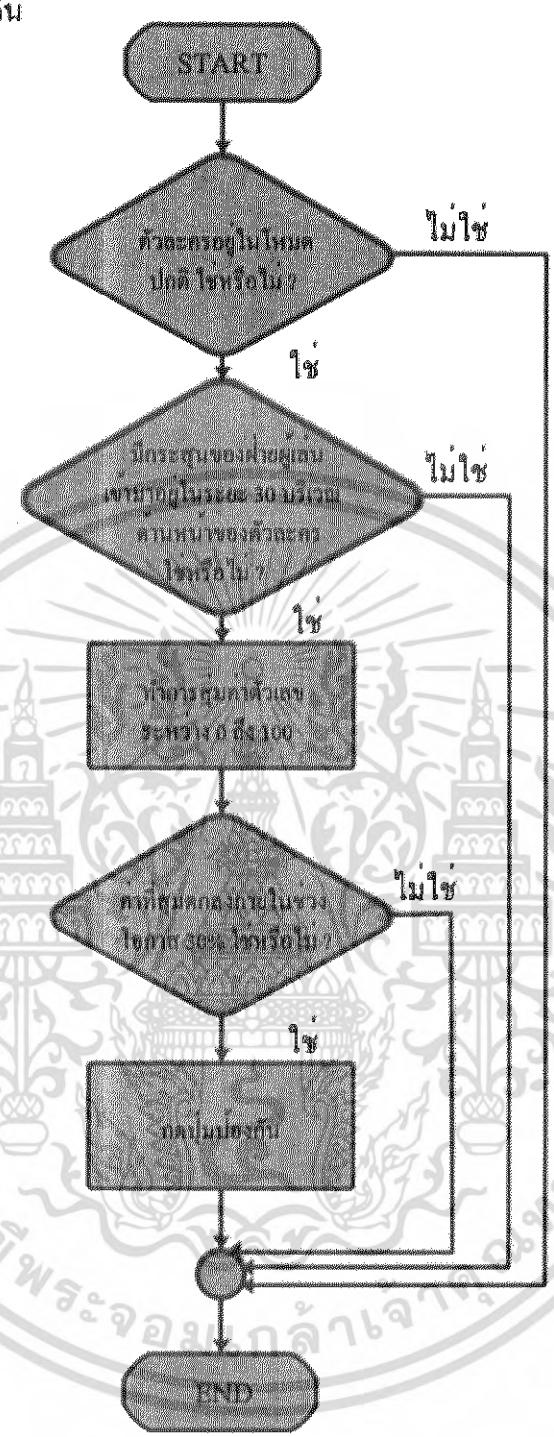
***ช่วงเวลาที่ระบบจะทำงาน :*** ระบบจะทำงานก็ต่อเมื่อตัวละครของระบบปัญญาประดิษฐ์อยู่ในโหมดปกติ และมีกระสุนของฝ่ายผู้เล่นเข้ามาในทิศทางข้างหน้าเท่านั้น

การป้องกันนั้นมีจุดอ่อนอยู่ที่ ในขณะที่ทำการป้องกันนั้นพลังงานของหุ่นยนต์จะไม่เพิ่มขึ้น จึงอาจไม่ใช่ทางเลือกที่ดีสักเท่าไหร่ในการที่จะป้องกันกระสุน แทนที่จะหลบหลีกกระสุน แต่กระนั้นการป้องกันก็ยังมีประโยชน์ในกรณีที่มีโอกาสที่จะโดนกระสุนสูง แต่ในที่นี้จะกำหนดระบบป้องกันที่เป็นพื้นฐานให้แก่ระบบปัญญาประดิษฐ์เสียก่อน เนื่องจากว่าการพิจารณาถึงการป้องกันจริงๆนั้นมีความซับซ้อนสูงมาก อาจต้องใช้การคำนวณทางคณิตศาสตร์เข้ามาช่วยเพื่อคำนวณทิศทางของกระสุนที่เข้ามา และความเป็นไปได้ที่จะโดน ซึ่งจะเป็นการเสียเวลาจนเกินไป จึงได้พัฒนาส่วนของการหลบหลีกที่ดูจะมีประสิทธิภาพมากกว่าไว้เป็นตัวทำงานหลักแทน ซึ่งจะได้กล่าวถึงในส่วนการทำงานอื่นๆ

หลักการทำงานของการป้องกัน คือ เมื่อมีกระสุนเข้ามาในระยะ 30 หน่วย บริเวณช่วงด้านหน้าของตัวละคร จะมีโอกาส 30% ที่จะเกิดการป้องกันเป็นเวลาราว 1 วินาที

ต่อไปจะได้อธิบายถึงวิธีการทำงานของระบบ

**การพิจารณาการป้องกัน**




**รูปที่ 3.52** การพิจารณาการป้องกัน

***การทำงาน***

- การทำงานเริ่มจากการตรวจสอบก่อนว่าตัวละครของระบบปัญญาประดิษฐ์นั้นอยู่ในโหมดปกติหรือไม่ ถ้าไม่ได้อยู่ในโหมดปกติ ระบบส่วนนี้จะไม่ทำงาน

- ถ้าหากอยู่ในโหมดปกติ ก็จะทำการพิจารณาต่อว่ามีกระสุนของฝ่ายผู้เล่นเข้ามาอยู่ในระยะ 30 หน่วย ที่หน้าด้านหน้าของตัวละครหรือไม่
- ถ้ามี จึงทำการสุ่มค่าตัวเลขออกมาภายใต้โอกาส 30 %
- ถ้าค่าที่สุ่มออกมา ตกลงในช่วงโอกาส 30% ก็จะทำการกดปุ่มป้องกันตัวเอง
- ถ้าค่าที่สุ่มออกมา ไม่ได้ตกอยู่ในช่วงโอกาส 30% ก็จะทำการข้ามส่วนการทำงานในการกดปุ่มป้องกันไป

**3.2.2.5 การกำหนดระยะห่างระหว่างผู้เล่น**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถเว้นระยะห่างระหว่างตัวละครของตัวเอง กับผู้เล่น ได้อย่างเหมาะสมต่อสถานการณ์ในสภาวะต่างๆ

***ช่วงเวลาที่ระบบจะทำงาน :*** ระบบจะทำงานทุกๆ 10 วินาที หรือ ทุกๆ 5 วินาที ในกรณีที่พลังชีวิตของระบบปัญญาประดิษฐ์เหลือน้อย

การกำหนดระยะห่างระหว่างตัวละครของระบบปัญญาประดิษฐ์และผู้เล่นนั้นมีความสำคัญมาก เนื่องจากระยะห่างนั้นมีผลต่อความแม่นยำในการยิง และการติดตามเป้าหมายเพื่อต่อสู้ นั่นหมายความว่ายิ่งมีระยะห่างที่มากเท่าไหร่ ผู้เล่นและระบบปัญญาประดิษฐ์ก็ยิ่งที่จะมีโอกาสในการยิงโดนกันได้ยากเท่านั้น นอกจากนี้ ระยะห่างยังมีผลต่อการซ่อน หลบหนี หรือกระทั่งการเข้าถึงตัวอีกด้วย ฉะนั้นระบบปัญญาประดิษฐ์ต้องทราบว่า ในรูปแบบสถานการณ์ที่เป็นอยู่นั้น ตัวละครของตนควรจะยืนห่างจากตัวผู้เล่นประมาณเท่าไหร่ เช่น ถ้าหากเลือดของตัวละครเหลือน้อย ก็ควรที่จะยืนในระยะห่างที่ค่อนข้างไกล เพื่อหลีกเลี่ยงโอกาสที่จะถูกยิงได้ง่าย หรือถ้าผู้เล่นเหลือพลังชีวิตน้อย ระบบปัญญาประดิษฐ์ก็ควรพยายามที่จะเข้าใกล้ตัวผู้เล่น เพื่อกดดันและเพิ่มโอกาสการยิงโดนให้สูงขึ้น เป็นต้น

การที่ระบบปัญญาประดิษฐ์จะสามารถเพิ่มหรือลดระยะห่างระหว่างตัวเองกับผู้เล่นได้นั้น กระทำโดยการกดเดินหน้าหรือถอยหลัง เนื่องจากในส่วนการทำงานกำหนดทิศทางการมองนั้น จะทำให้ระบบปัญญาประดิษฐ์หันหน้าไปยังเป้าหมายเสมอ หมายความว่าถ้าต้องการร่นระยะ ก็คือการกดเดินไปข้างหน้า และถ้าต้องการเพิ่มระยะนั่นก็คือการกดถอยไปข้างหลัง ด้วยหลักการนี้จะทำให้เราสามารถรักษาระยะที่ต้องการไว้ได้

ความกว้างและยาวของฉากจะมีขนาดเท่ากันคือที่ 200 หน่วย โดยจะมีขนาดเส้นแทยงมุมราว 270 หน่วย ฉะนั้นระยะห่างต่างๆนั้นจะกำหนดให้มีอยู่ด้วยกัน 5 ระยะ คือ ใกล้มาก ใกล้ ปานกลาง ไกล และ ไกลมาก โดยแต่ละระยะจะมีช่วงค่าที่กำหนดให้อยู่ คือ

ใกล้มาก อยู่ในช่วงความห่างราว 5 ถึง 25 หน่วย

ใกล้ อยู่ในช่วงความห่างราว 40 ถึง 60 หน่วย

ปานกลาง อยู่ในช่วงความห่างราว 80 ถึง 110 หน่วย

ไกล อยู่ในช่วงความห่างราว 130 ถึง 160 หน่วย

ไกลมาก อยู่ในช่วงความห่างราวตั้ง 180 หน่วยขึ้นไป

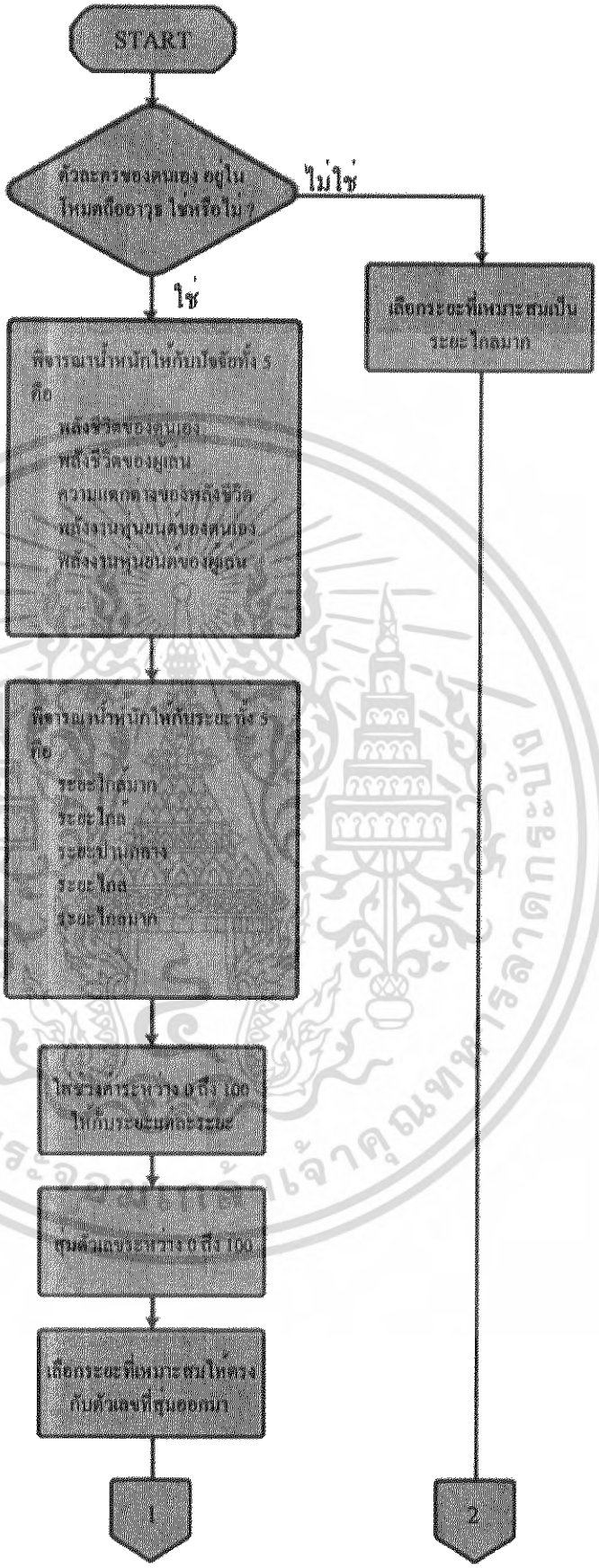
ถ้าหากระบบปัญญาประดิษฐ์รู้แล้วว่าควรจะยืนที่ระยะไหน ก็จะต้องสุ่มค่าภายในช่วงระยะนั้นออกมา แล้วจดจำให้เป็นระยะห่างเป้าหมายที่ควรจะทำระยะห่างให้ได้ตามนั้น ซึ่งระยะห่างเป้าหมายที่เหมาะสมนี้เองจะถูกอ่านโดยส่วนที่ควบคุมการเคลื่อนไหวเพื่อที่จะตัดสินใจว่า ควรจะเดินหน้าหรือถอยหลังในขณะนั้น

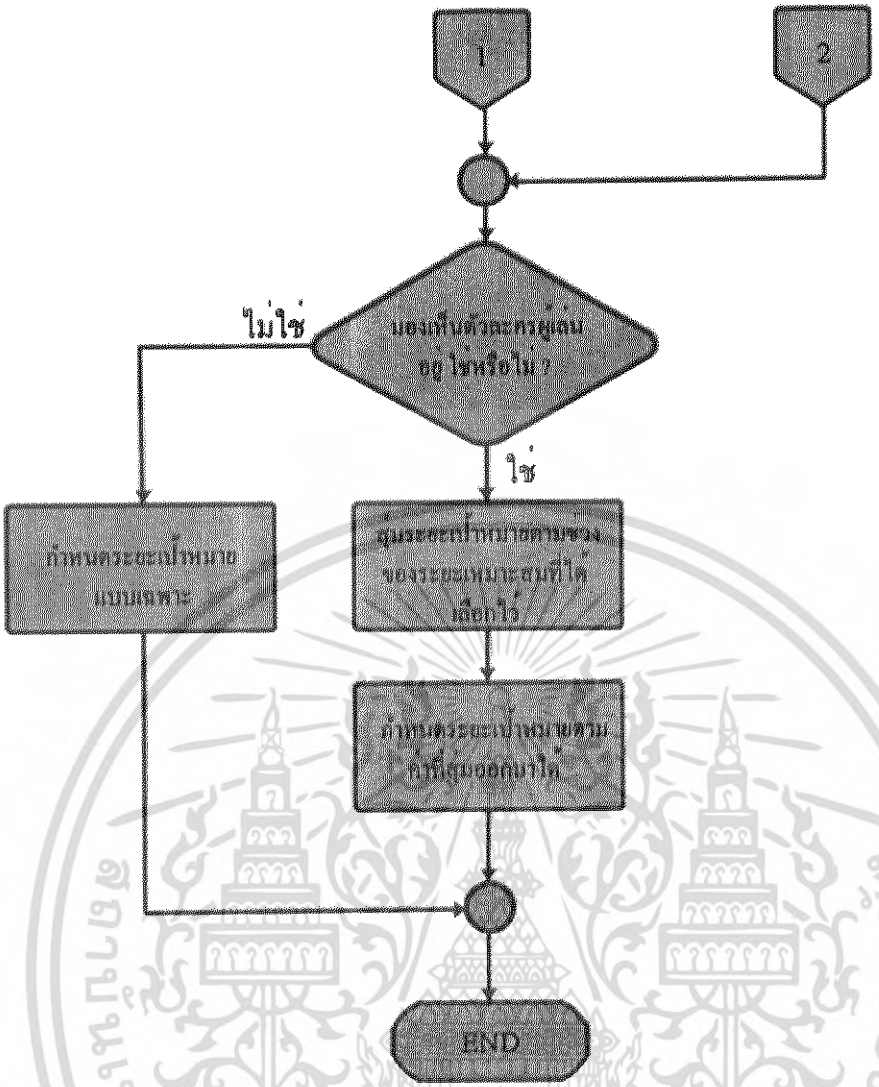
การพิจารณาว่าระบบปัญญาประดิษฐ์ควรจะอยู่ที่ระยะห่างระหว่างตัวเองกับผู้เล่นเป็นเท่าไหร่นั้นจะพิจารณาจากปัจจัยหลัก 8 อย่าง คือ พลังงานชีวิตของตัวเอง พลังงานชีวิตของผู้เล่น ความแต่งต่างของพลังชีวิต พลังงานของหุ่นยนต์ตัวเอง พลังงานของหุ่นยนต์ผู้เล่น โหมดของตัวเอง โหมดของผู้เล่น และสถานะการมองเห็น โดยผลลัพธ์ที่ได้จะเกิดจากการพิจารณาปัจจัยร่วมทั้งหมดแล้วประเมินออกเป็นช่วงน้ำหนักให้แก่ระยะทั้ง 5 ระยะ และทำการกระจายช่วงความน่าจะเป็นตั้งแต่ 0 จนถึง 100 ให้เป็น 5 ช่วง ตามความมากหรือน้อยของน้ำหนักแต่ละระยะที่ได้ จากนั้นจึงทำการสุ่มตัวเลขออกมา หากตกลงบนช่วงค่าของระยะใดนั้น ก็จะตัดสินใจเลือกระยะนั้นให้เป็นระยะที่เหมาะสมสำหรับสถานการณ์ในขณะนั้น จากนั้นจึงสุ่มค่าที่อยู่ภายในช่วงค่าของระยะนั้นออกมาเพื่อกำหนดให้เป็นระยะห่างเป้าหมาย

ในกรณีที่สถานะอยู่ในสภาพมองไม่เห็นผู้เล่น การพิจารณาก็จะมีความแตกต่างออกไป โดยจะพิจารณาจากระยะที่เหมาะสมขณะที่อยู่ในสถานะที่มองเห็นมาเป็นตัวข้อมูลตั้งต้น โดยระยะ ใกล้มาก และใกล้ จะถูกเปลี่ยนระยะห่างเป้าหมายเป็น 5 หน่วยในทันที ทั้งนี้ก็เพื่อให้ระบบปัญญาประดิษฐ์พยายามที่จะเข้าประชิดตัวผู้เล่นในทันที ไม่ให้ผู้เล่นสามารถที่จะหนีไปหลบหลังกล่องได้นานจนเกินไป หรือระยะไกล และระยะไกลมาก ก็จะถูกเปลี่ยนระยะเป้าหมายให้เป็น 210 หน่วยในทันทีเพื่อถอยห่างจากผู้เล่นให้มากที่สุด เนื่องจากในขณะเริ่มเกม ทั้งผู้เล่นและระบบปัญญาประดิษฐ์จะเห็นซึ่งกันและกัน ทำให้ระบบปัญญาประดิษฐ์จะมีโอกาสคำนวณระยะห่างที่เหมาะสมไว้ก่อนอยู่แล้ว จึงสามารถมั่นใจได้ว่า ในขณะที่เกิดการเข้าสู่สถานะมองไม่เห็นตัวละครผู้เล่นนั้น ระบบปัญญาประดิษฐ์จะมีระยะห่างที่เหมาะสมสำหรับการประมวลผลอย่างแน่นอน

ต่อไปจะได้กล่าวถึงการทำงานของระบบในส่วนนี้

การกำหนดระยะห่างระหว่างผู้เล่น






**รูปที่ 3.53** การกำหนดระยะห่างระหว่างผู้เล่น

*การทำงาน*

- เริ่มต้นที่ การพิจารณาว่าตัวละครของระบบปัญญาประดิษฐ์ว่าอยู่ในโหมดถืออาวุธหรือไม่
- ถ้าตัวเองไม่ได้อยู่ในโหมดถืออาวุธ จะปรับระยะห่างที่เหมาะสมเป็นระยะไกลมากทันที
- ถ้าหากตัวเองอยู่ในโหมดถืออาวุธ จะเริ่มทำการพิจารณาน้ำหนักให้กับปัจจัยทั้ง 5 ก่อน คือ พลังชีวิตของตน พลังชีวิตของผู้เล่น ความแตกต่างของพลังชีวิต พลังงานหุ่นยนต์ของตน พลังงานหุ่นยนต์ของผู้เล่น เพราะว่าในแต่ละสถานะการณ์ ปัจจัยแต่ละอย่างมีผลความสำคัญไม่เท่ากัน เช่น ตอนที่พลังชีวิต

เหลือน้อย ก็ควรจะให้น้ำหนักกับปัจจัยด้านพลังชีวิตสูง เพราะอาจจะทำให้พ่ายแพ้ได้

- เมื่อได้น้ำหนักของปัจจัยทั้ง 5 ปัจจัย จึงทำการพิจารณาการใส่ค่าน้ำหนักให้แก่ระยะต่างๆ 5 ระยะคือ ใกล้มาก ใกล้ ปานกลาง ไกล ไกลมาก การใส่น้ำหนักให้แก่ระยะทั้ง 5 นี้อันเนื่องมาจาก การพิจารณาว่าระบบปัญญาประดิษฐ์ควรจะยืนอยู่ที่ระยะใดนั้น จะดูจากปัจจัยทั้ง 5 และความสำคัญของแต่ละปัจจัยเมื่อพิจารณาแล้ว จึงได้ออกมาว่า ระยะใดมีความเหมาะสมมากหรือน้อย ซึ่งระยะที่มีความเหมาะสมมากก็จะมีน้ำหนักมากตามไปด้วย ส่วนระยะที่มีความเหมาะสมน้อยก็จะมีน้ำหนักที่น้อยตามไปด้วย ซึ่งถ้ามีการรวมน้ำหนักจากระยะทั้ง 5 แล้ว จะมีค่าพอดีที่ 100
- ด้วยน้ำหนักของระยะทั้ง 5 นี้เอง เราจึงทำการใส่ช่วงค่าระหว่าง 0-100 ให้แก่ระยะแต่ละตัวตามน้ำหนักของมัน เช่น ระยะใกล้ มีน้ำหนักอยู่ที่ 20 ก็มีโอกาสที่จะได้ช่วงค่า 40-60 เป็นต้น
- จากนั้นจึงทำการสุ่มค่าระหว่าง 0-100 ถ้าเกิดตัวเลขที่สุ่มได้ อยู่ในช่วงค่าของระยะใด จึงเลือกระยะนั้นเป็นระยะที่เหมาะสม เช่น ถ้าค่าที่สุ่มได้ออกมา มีค่า 46 ซึ่งตกลงในช่วง 40-60 ที่เป็นของระยะใกล้ ระบบก็จะทำการเลือกระยะใกล้เป็นระยะที่เหมาะสม
- ถ้าหาก อยู่ในสถานะที่ไม่สามารถมองเห็นได้ จะเข้าสู่รูปแบบการใส่ค่าระยะเป้าหมายเฉพาะ
- โดยถ้าเป็น ระยะใกล้มาก หรือใกล้ จะใช้ระยะเป้าหมายเป็น 5 หน่วย
- ถ้าเป็น ปานกลาง จะใช้ระยะเป้าหมายที่ 70 หน่วย
- ถ้าเป็น ไกล หรือ ไกลมาก จะใช้ระยะเป้าหมายที่ 210 หน่วย
- ถ้าเป็นกรณีที่มองเห็นศัตรู เมื่อทราบถึงระยะที่เหมาะสมแล้ว จึงทำการสุ่มค่าระยะเป้าหมาย ของระยะที่เหมาะสมนั้น เช่นระยะใกล้ มีช่วงค่าอยู่ที่ 40-60 หน่วย ก็ทำการสุ่มค่าระหว่างช่วงค่านี้ เป็นระยะเป้าหมายของระบบปัญญาประดิษฐ์ที่ควรจะอยู่ห่างจากผู้เล่น

**3.2.2.6 พิจารณาการไปทางซ้ายหรือทางขวา**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถตัดสินใจที่จะเลือกไปทางซ้าย หรือทางขวา และทำการหลบกระสุนตามทิศทางที่ลูกกระสุนผ่านเข้ามาได้

***ช่วงเวลาที่ระบบจะทำงาน:*** ในส่วนของการพิจารณาการหลบกระสุนจะทำงานทุกๆ 0.3 วินาที และในส่วนของการพิจารณาการเดินไปทางซ้ายหรือขวาจะทำงานทุกๆ 2 ถึง 7 วินาที

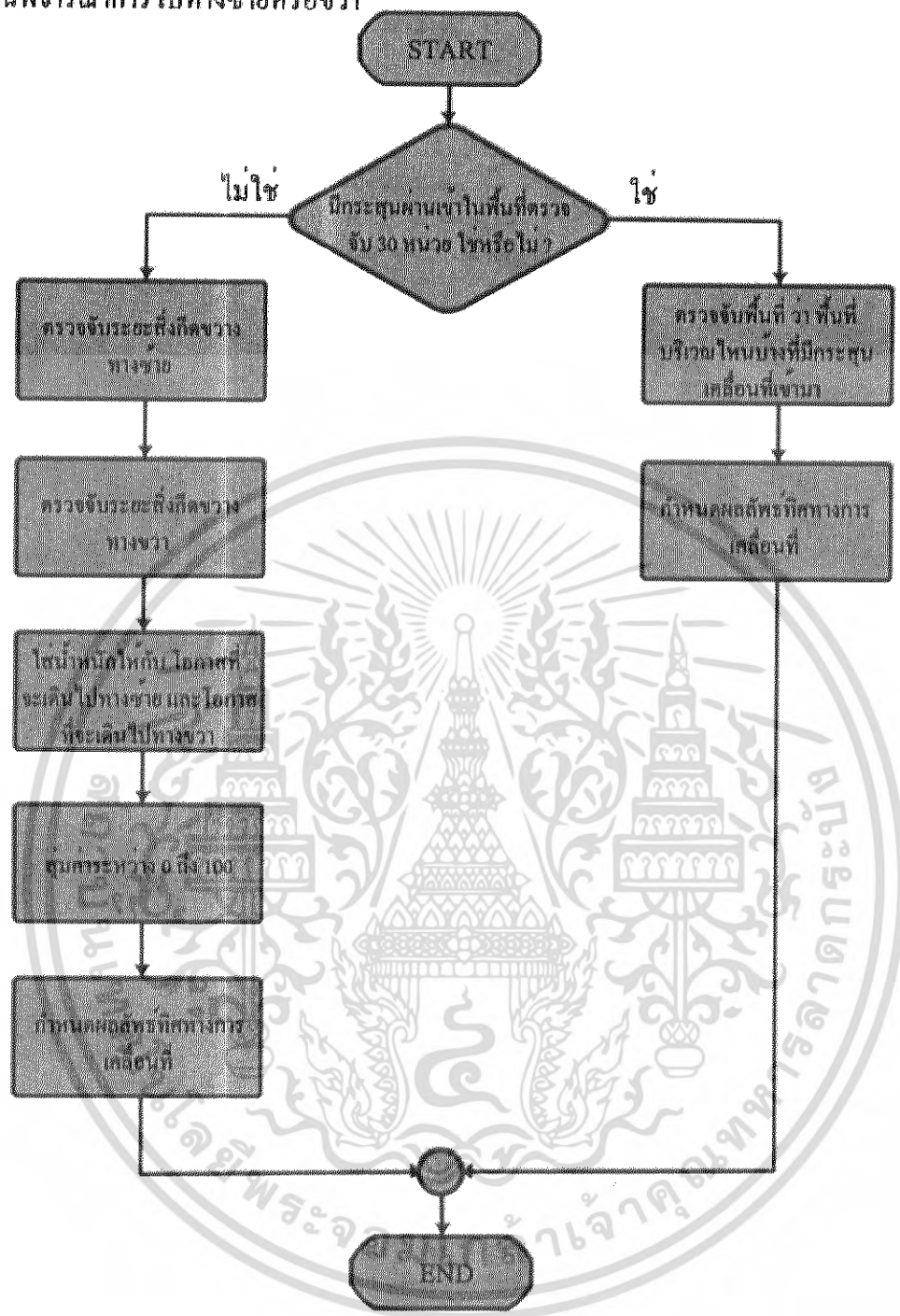
นอกจากการเคลื่อนที่ไปข้างหน้า หรือถอยหลังแล้ว ก็ยังมีการเคลื่อนที่ในทิศทางซ้ายและขวาอีก การเคลื่อนที่ไปยังซ้ายและขวานั้น จะเป็นปัจจัยสำคัญที่จะทำให้เกิดการเคลื่อนที่หลบกระสุน ล่อหลอก หรือการหลบเข้าไปแอบซ่อนหลังกล่องได้ ทำให้ระบบปัญญาประดิษฐ์จะต้องมีการตัดสินใจว่าในสถานการณ์เช่นใด ควรที่จะเลือกเดินไปยังทางซ้ายหรือขวา เช่น ถ้ามีกระสุนเข้ามาบริเวณทางซ้าย ก็ควรที่จะเดินหลบไปในทิศทางทางขวา เป็นต้น

การหลบกระสุนจะพิจารณาจากตำแหน่งของกระสุนภายใต้ระยะ 30 หน่วย โดยจะกำหนดทิศทางการหลบให้ตามความเหมาะสม ซึ่งจะมีกรณีหลักๆอยู่สามกรณี คือ หากมีกระสุนอยู่บริเวณด้านหน้าทางซ้าย ก็ให้หลบไปทางขวา หากมีกระสุนอยู่บริเวณด้านหน้าทางตรง ก็ให้สุ่มการหลบกระสุนไปทางซ้ายหรือขวา หรือถ้าหากมีกระสุนอยู่บริเวณด้านหน้าทางขวาก็ให้หลบไปทางซ้าย ด้วยการทำงานที่ต่อเนื่องของกรณีการหลบที่ได้วางเอาไว้ จะทำให้ ถ้ามีจำนวนกระสุนมากกว่าหนึ่งนัดที่เข้ามาในระยะ 30 หน่วย ระบบปัญญาประดิษฐ์ก็ยังสามารถที่จะเลือกตัดสินใจในการหลบหลีกได้อย่างเหมาะสม

การเคลื่อนที่ไปทางซ้ายหรือขวาแบบทั่วไปนั้นจะพิจารณาแยกกับส่วนของการหลบกระสุน โดยจะกำหนดให้ทิศทางที่ต้องเคลื่อนที่เพื่อหลบกระสุนมีความสำคัญมากกว่า แต่หากไม่ได้มีกระสุนถูกยิงเข้ามาสู่ตัวละครของระบบปัญญาประดิษฐ์แล้ว ก็จะทำให้การเคลื่อนที่ไปทางซ้ายหรือขวาถูกตัดสินใจโดยระบบพิจารณาสิ่งกีดขวาง โดยจะให้ความสำคัญกับทิศทางการเคลื่อนที่ ที่จะมีโอกาสจะเจอสิ่งของน้อยกว่าทิศทางการเคลื่อนที่ที่โปร่งโล่ง เช่น ต้องการพิจารณาว่าจะไปทางซ้ายหรือขวาดี ที่ทางซ้ายห่างไป 30 หน่วย พบว่า มีกล่อง ในขณะที่ทางขวา ห่างไป 80 หน่วยจึงจะพบกล่อง ก็จะให้น้ำหนักที่จะเคลื่อนที่ไปทางขวามากกว่าทางซ้าย นั่นก็เพราะทางซ้ายมีสิ่งกีดขวางอยู่ใกล้กว่าทางขวา ที่ออกแบบหลักการให้มีลักษณะนี้เนื่องจากกว่าในการต่อสู้ พื้นที่ถือว่าเป็นสิ่งสำคัญมาก หากไม่มีพื้นที่ ที่จะให้เคลื่อนที่ อาจจะทำให้ถูกยิงได้ง่าย

ทั้งนี้การทำงานในส่วนของการตัดสินนั้นจะส่งไปยังส่วนควบคุมก

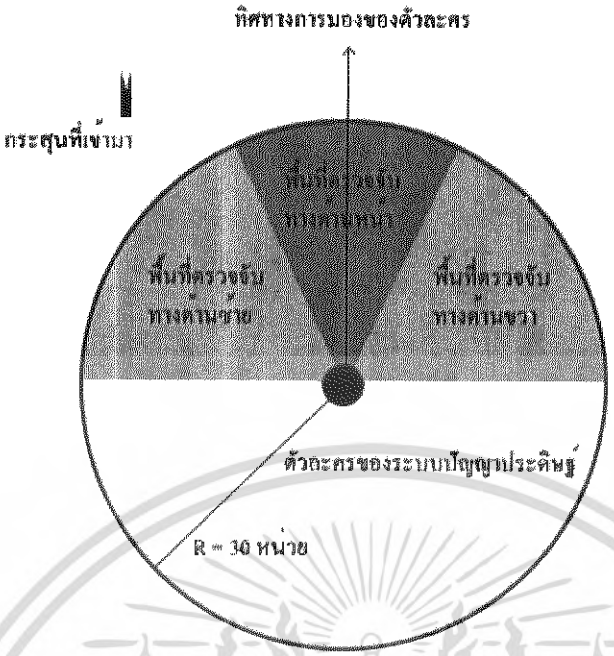
ส่วนพิจารณาการไปทางซ้ายหรือขวา




**รูปที่ 3.54** ส่วนพิจารณาการไปทางซ้ายหรือขวา

*การทำงาน*

- เริ่มต้นที่ พิจารณาก่อนว่ามีกระสุนผ่านเข้ามาในระยะ 30 หน่วย บริเวณด้านหน้าตัวละครของระบบปัญญาประดิษฐ์หรือไม่




**รูปที่ 3.55** การตรวจจับกระสุนที่เข้ามาในระยะ 30 หน่วย ขณะที่กระสุนยังไม่ได้ผ่านเข้ามา


**รูปที่ 3.56** การตรวจจับกระสุนที่เข้ามาในระยะ 30 หน่วย ขณะที่กระสุนผ่านเข้ามายังบริเวณตรวจจับทางด้านซ้าย

จากภาพ 3.2.2 (14) จะเห็นได้ว่ากระสุนยังไม่ได้ผ่านเข้ามาในระยะการตรวจสอบใดๆ ฉะนั้นระบบปัญญาประดิษฐ์จะยังไม่สามารถตรวจจับกระสุนได้ และจะใช้การพิจารณาการเคลื่อนไปยังทิศทางซ้ายหรือขวาแบบปกติ คือตรวจสอบจากสิ่งกีดขวาง แต่ว่า พอกระสุนผ่านเข้ามาสู่พื้นที่ตรวจสอบทางด้านซ้าย ระบบก็จะรู้ทันทีว่ามีกระสุนเข้ามาทางบริเวณซ้าย ซึ่งจะส่งผลให้การตัดสินใจที่จะเดินนั้นอิงกับผลการตรวจสอบการหลบกระสุน ซึ่งในกรณีนี้ก็จะหลบไปทางขวา แทนที่จะเดินไปในทางซ้าย

- ถ้ามีกระสุนเข้ามาภายในพื้นที่ตรวจจับก็จะทำการพิจารณาว่า กระสุนผ่านเข้ามายังทิศทางใด จากนั้นจึงทำการตัดสินใจว่าควรจะเคลื่อนที่ไปในทิศทางใดเพื่อทำการหลบกระสุน
- ถ้าไม่สามารถตรวจจับได้ว่ามีกระสุนผ่านเข้ามา ก็จะทำการพิจารณาว่าจะไปทางซ้ายหรือทางขวาด้วยหลักการที่ว่าทิศทางไหนมีโอกาสที่จะเจอสิ่งกีดขวางได้น้อยกว่ากัน
- ทำการสุ่มค่าระหว่าง 0 ถึง 100 ออกมา
- ถ้าตกอยู่ในช่วงโอกาสของทิศทางใด ก็จะเลือกทิศทางนั้นในการเคลื่อน ซึ่งผลลัพธ์ของการตัดสินใจนั้นจะถูกส่วนที่ควบคุมการเคลื่อนอ่านเพื่อนำไปควบคุมตัวละครอีกที

**3.2.2.7 พิจารณาการกดปุ่มการเคลื่อนไหว**

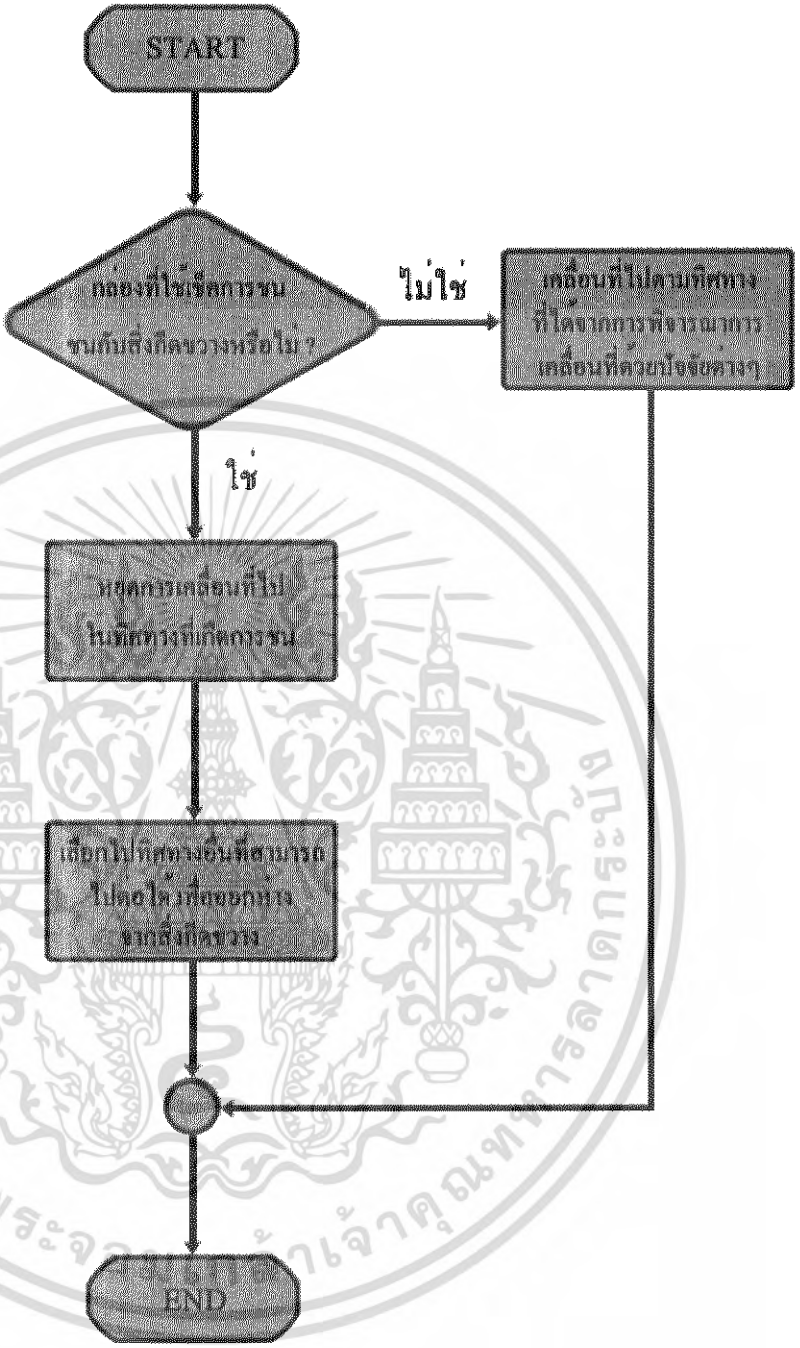
***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถกระทำการเคลื่อนไหวไปในทิศทางที่ได้ตัดสินใจไปและพิจารณาต่อว่าทิศทางที่ได้ตัดสินใจไปนั้นสามารถไปได้โดยไม่ติดสิ่งกีดขวางได้หรือไม่

***ช่วงเวลาที่ระบบจะทำงาน:*** ในส่วนของการพิจารณาการเคลื่อนที่ว่าสามารถไปได้หรือไม่จะพิจารณาตลอดเวลาว่าติดสิ่งกีดขวางใดและทิศทางใดบ้าง

โดยการเคลื่อนที่ไปในทิศทางต่างๆนั้นจะได้จากการรักษาระยะห่างที่ควรอยู่และแนวโน้มทิศทางการหลบกระสุนของระบบปัญญาประดิษฐ์เป็นปัจจัยเบื้องต้น ซึ่งการเคลื่อนที่นี้ก็จะมีขอบเขตอยู่ที่ว่าถ้าไปเจอสิ่งกีดขวางก็ไม่ควรจะไปต่อควรหยุดอยู่แค่นั้นและพิจารณาว่าควรไปทิศทางใด เช่นเดินเรียบกล่องไปเรื่อยๆ หรือถอยออกจากกกล่องหรือแม้กระทั่งการถอยไปจนติดมุม ซึ่งต้องมีการพิจารณาปัจจัยต่างๆว่าควรเดินขึ้นหน้าหรือเดินไปในทิศทางซ้าย-ขวาเพื่อออกห่างผนังหรือทั้งสองอย่าง

การพิจารณาสิ่งกีดขวางนั้นจะใช้กล่องซึ่งอยู่รอบทิศทางของระบบปัญญาประดิษฐ์โดยแบ่งออกเป็น 8 ทิศทาง เพื่อเช็คว่าเข้าใกล้สิ่งกีดขวางที่ทิศทางใดบ้าง เพื่อไม่ให้ระบบปัญญาประดิษฐ์ ได้เดินเข้าไปติดสิ่งกีดขวาง

**ส่วนควบคุมการเคลื่อนไหว**




**รูปที่ 3.57** ส่วนควบคุมการเคลื่อนไหว

***การทำงาน***

- เริ่มต้นที่ การพิจารณากล่องที่ใช้เช็คการชนของแต่ละทิศทางของระบบปัญญาประดิษฐ์ว่ามีกล่องในทิศทางใดบ้างของระบบปัญญาประดิษฐ์ที่ไปชนกับสิ่งกีดขวาง

- ถ้าไม่มีกล่องในทิศทางใดเลยที่ชนกับสิ่งกีดขวางก็ให้จบการพิจารณาในส่วนนี้ไปโดยยังคงให้เคลื่อนที่ไปตามทิศทางที่ได้จากการพิจารณาการเคลื่อนที่
- ถ้ามีกล่องใดกล่องหนึ่งที่อยู่รอบตัวละครของระบบปัญญาประดิษฐ์เกิดการชนกับสิ่งกีดขวางเข้าก็ให้ทำการกดปุ่มในทิศทางอื่นที่สามารถไปได้แทนเพื่อหลีกเลี่ยงการเดินติดสิ่งกีดขวาง

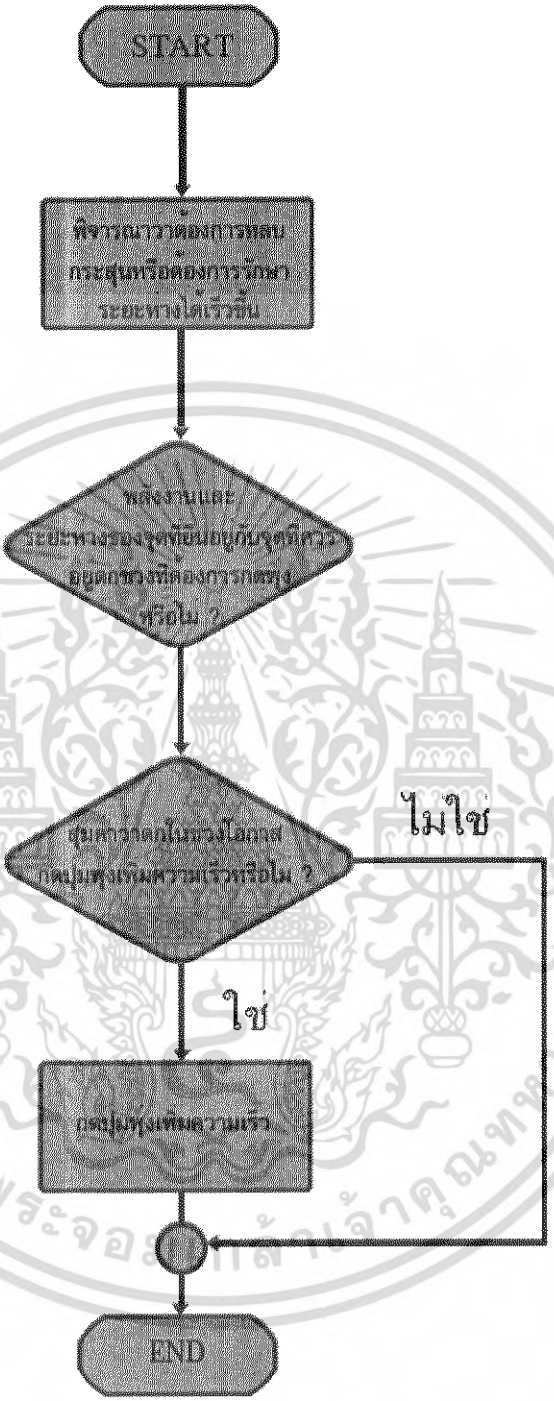
**3.2.2.8 พิจารณาการกดปุ่มเพิ่มความเร็ว**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถกระทำการเคลื่อนไหวไปด้วยความเร็วที่เพิ่มขึ้นเพื่อให้สามารถทำการหลบกระสุนได้พ้นและรักษาระยะที่ควรออกห่างจากตัวผู้เล่นได้รวดเร็วขึ้น

***ช่วงเวลาที่ระบบจะทำงาน:*** ในส่วนของการพิจารณาการการพุ่งด้วยความเร็วนี้จะพิจารณาจากปัจจัยค่าพลังงานที่เหลืออยู่และความห่างของจุดที่ยืนอยู่กับจุดที่ควรอยู่ ซึ่งปัจจัยทั้ง 2 ตัวนี้จะเป็นตัวกำหนดค่าโอกาสในการพุ่งด้วยความเร็วว่ามีโอกาสเพิ่มความเร็วมากหรือน้อยเพียงใด จากนั้นจึงทำการสุ่มค่าว่าตกในช่วงโอกาสการกดปุ่มพุ่งเพิ่มความเร็วหรือไม่

การพิจารณาการพุ่งด้วยความเร็วนี้จะมีการแบ่งคิดแยกเป็น 2 กรณี คือ ต้องการกดปุ่มพุ่งเพิ่มความเร็วเพื่อการหลบกระสุน หรือ ต้องการกดปุ่มพุ่งเพิ่มความเร็วเพื่อให้ไปอยู่ในจุดที่ควรอยู่ได้อย่างรวดเร็วขึ้น

ส่วนการกดปุ่มพุ่งเพิ่มความเร็ว




**รูปที่ 3.58** ส่วนการกดปุ่มเพิ่มความเร็ว

***การทำงาน***

- เริ่มต้นที่ การพิจารณาว่าต้องการพุ่งด้วยความเร็วเพื่อทำการหลบกระสุน หรือพุ่งด้วยความเร็วเพื่อไปยังระยะที่ควรอยู่ได้รวดเร็วขึ้น ซึ่งทั้ง 2 ปัจจัยนี้จะได้โอกาสการพิจารณาพุ่งด้วยความเร็วที่แตกต่างกัน
- ทำการเช็คค่าพลังงานและระยะห่างว่าทั้ง 2 ปัจจัยนี้จะทำให้ระบบปัญญาประดิษฐ์มีโอกาสกดพุ่งเพิ่มความเร็วกี่เปอร์เซ็นต์ และใช้เวลากดปุ่มพุ่งเพิ่มความเร็วนี้นานเท่าใดถ้าตัดสินใจกดปุ่มพุ่ง
- ทำการสุ่มค่าว่าดกอยู่ในอยู่ในช่วงขอบเขตโอกาสการกดปุ่มพุ่งด้วยความเร็วหรือไม่ถ้าตกจึงให้ทำการกดปุ่มพุ่งเพิ่มความเร็ว

**3.2.2.9 พิจารณาการกดปุ่มกระโดด**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถกระโดดหลบกระสุนในกรณีที่กระสุนเข้ามาในระยะประชิดมีมากเกินไปทำให้การเคลื่อนที่หลบแบบปกติทำได้อย่างยากลำบาก

***ช่วงเวลาที่ระบบจะทำงาน:*** ในส่วนของการพิจารณาการการกระโดดนั้นจะพิจารณาจากกระสุนที่ทำการยิงมาจากตัวผู้เล่นว่ามีมากน้อยเพียงใดและสามารถคักทางตัวระบบปัญญาประดิษฐ์ได้มาน้อยเพียงใด ซึ่งถ้ากระสุนเข้ามาหลายทิศทางและอยู่ระยะประชิดระบบปัญญาประดิษฐ์จะตัดสินใจทำการกระโดด

**ส่วนควบคุมการกระโดด**


**รูปที่ 3.59** ส่วนควบคุมการกระโดด

*การทำงาน*

- เริ่มต้นที่ พิจารณาว่าหากต้องการหลบกระสุนและกระสุนนั้นสามารถดักทางการเคลื่อนที่ของระบบปัญญาประดิษฐ์ได้หลายทิศทางและกระสุนอยู่ในระยะประชิดระบบปัญญาประดิษฐ์จะตัดสินใจทำการกระโดด

**3.2.2.10 ส่วนพิจารณาการยิงกระสุน**

***จุดประสงค์การทำงาน :*** เพื่อให้ระบบปัญญาประดิษฐ์สามารถที่จะยิงกระสุนเพื่อยู่โจมและตอบโต้ผู้เล่นได้ รวมถึงคำนวณองศาการยิงเผื่อไปยังจุดที่ได้คาดการณ์ไว้ว่าผู้เล่นจะทำการเคลื่อนที่ไป

***ช่วงเวลาที่ระบบจะทำงาน:*** เฟสการยิงจะใช้เวลาทั้งหมด 3 วินาที และ มีช่องว่างขณะที่ระบบจะไม่ทำงานอีกราว 1 ถึง 2 วินาที ต่อท้ายหลังจากจบเฟสการยิง เพื่อเป็นช่วงขั้นระหว่างเฟสการยิงกระสุนในแต่ละเฟส

ระบบยิงกระสุนนั้นมีการทำงานเป็นช่วงๆ (เฟส) ซึ่งในเฟสการยิงหนึ่งเฟสนั้นจะยิงกระสุนได้สูงสุด 5 นัด การที่ระบบจะตัดสินใจว่าควรจะยิงกระสุนทั้งหมดกี่นัดในเฟส

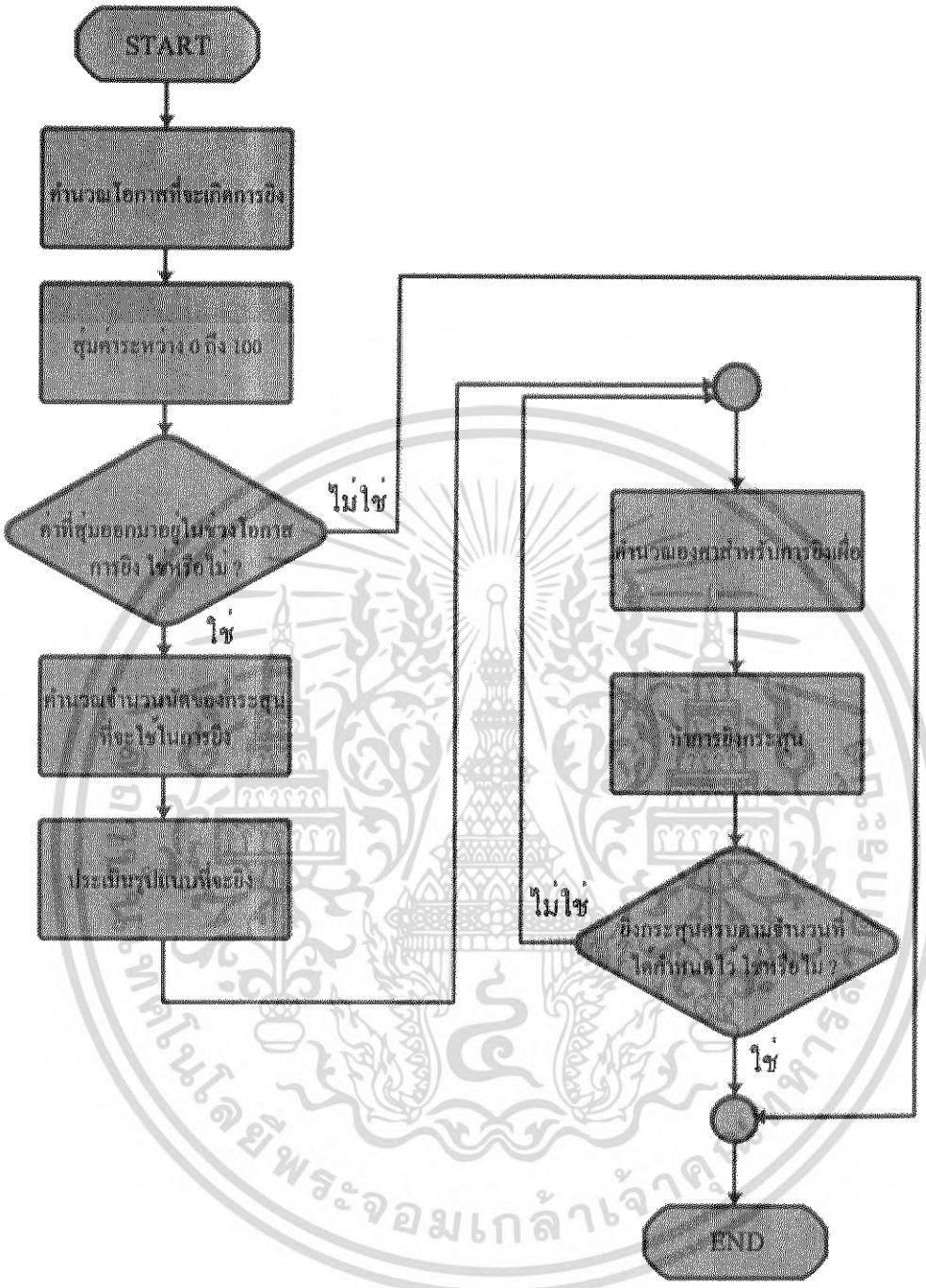
นั้นๆ จะพิจารณาจากพลังงานของหุ่นยนต์ที่เหลืออยู่ และระยะห่างระหว่างตนเองและผู้เล่น โดยยิ่งผู้เล่นอยู่ใกล้ตัวละครระบบปัญญาประดิษฐ์มากเท่าใดและพลังงานยิ่งเหลือมากเท่าไหร่ โอกาสที่จะยิงด้วยจำนวนนัดที่สูงก็จะยิ่งมากขึ้นเท่านั้น กระสุนที่จะถูกยิงออกไปในแต่ละเฟสนั้นจะมีลักษณะการกระจายตัวในการยิงอยู่หลายแบบ โดยได้กำหนดไว้ด้วยกันถึง 15 แบบ เพื่อความหลากหลายในการโจมตี

โดยปกติแล้วตัวละครของระบบปัญญาประดิษฐ์จะหันหน้าไปหาตัวละครของผู้เล่นเสมอ ฉะนั้นการโจมตีด้วยกระสุนก็ย่อมจะพุ่งไปยังตำแหน่งของผู้เล่นอย่างแน่นอน แต่เนื่องจากกระสุนต้องใช้เวลาในการเดินทาง กว่าที่จะไปถึงยังตำแหน่งของผู้เล่น ก็อาจทำให้กระสุนพลาดเป้าได้ เพราะผู้เล่นได้เคลื่อนที่ไปจากตำแหน่งนั้นแล้ว จึงได้มีการใส่หลักการทำงานในการยิงเผื่อขึ้นมา โดยระบบจะทำการคำนวณก่อนที่จะยิงว่า ผู้เล่นจะเคลื่อนที่ไปยังทิศทางใด แล้วก็จะทำการยิงกระสุนโดยเผื่อระยะที่ผู้เล่นจะเคลื่อนที่เข้าไปด้วย องศาที่ทำการยิงเผื่อนั้นจะแปรผันไปตามระยะห่างระหว่างผู้เล่นและระบบปัญญาประดิษฐ์ โดยยิ่งระยะทางไกลขึ้นก็ยิ่งต้องเผื่อมุมมากขึ้น โดยมีเกณฑ์การกำหนดจากการประมาณค่าความเร็วของกระสุน

ระบบการยิงเป็นระบบที่ยังสามารถพัฒนาไปได้อีกมาก เนื่องจากการทดสอบการยิงในความเป็นจริงนั้นยังมีข้อผิดพลาดในเรื่องการกะเกณฑ์มุมเผื่ออยู่ ถ้าหากผู้เล่นพอที่จะจับจุดการยิงได้นั้น ก็อาจทำให้ระบบปัญญาประดิษฐ์เกิดความยากลำบากที่จะยิงโดนผู้เล่นเป็นอย่างมาก ซึ่งหากจะแก้ไขข้อบกพร่องนี้ได้ก็อาจจำเป็นที่จะต้องใช้เวลาอีกพอสมควร

ต่อไปจะได้อธิบายถึงระบบการทำงาน

**ส่วนพิจารณาการยิงกระสุน**




**รูปที่ 3.60** ส่วนพิจารณาการยิงกระสุน

***การทำงาน***

- เริ่มจาก ทำการคำนวณ โอกาสที่จะเกิดการยิงจากพลังงานหุ่นยนต์ของระบบปัญญาประดิษฐ์ และระยะห่างระหว่างตนเองกับผู้เล่น

- สุ่มค่าระหว่าง 0 ถึง 100
- ถ้าค่าที่สุ่มออกมาตกลงในช่วงโอกาสที่เกิดการยิง ก็จะเริ่มต้นการทำงานของการยิง
- คำนวณจำนวนนัดที่จะยิงจากพลังงานหุ่นยนต์ของระบบปัญญาประดิษฐ์ และระยะห่างระหว่างตนเองกับผู้เล่น
- คำนวนรูปแบบการยิงกระสุน
- หลังจากทราบจำนวนนัดและรูปแบบที่จะยิงแล้ว จะเข้าสู่การวนซ้ำการยิงกระสุน
- ถ้าหากยังยิงไม่ครบตามจำนวนนัดที่ได้กำหนดไว้ ก็จะทำการกลับมาวนซ้ำเพื่อทำการยิงให้ครบ
- โดยทุกครั้งก่อนที่จะทำการยิงจะคำนวณมุมเผื่อ โดยพิจารณาจากระยะห่างของผู้เล่นและ การประเมินทิศทางการเคลื่อนที่ของผู้เล่น
- จากนั้นจึงทำการยิงกระสุน

## 3.3 ปัญหาและอุปสรรคของการพัฒนาโปรแกรม

### 3.3.1 ปัญหาด้านโครงสร้างในการทำงานของ Quest3D

- **ไม่สามารถเรียกใช้โปรแกรมเชิงการแยกส่วนการทำงานได้**

เนื่องจากการเขียนโปรแกรมเป็นรูปแบบเชิงลากโหนดซึ่งจะทำการประมวลผลจากซ้ายไปขวาและจากบนลงล่างทำให้ไม่สามารถโยงโหนดไปเพื่อเรียกใช้อีกครั้งได้

- **โปรแกรมมีความเชื่องช้า**

เนื่องมาจากการใช้โครงสร้างโหนดแทนการเขียนโปรแกรม ซึ่งโปรแกรมจะมีการเก็บค่าและตำแหน่งโหนดที่วางไว้บนแผนผังเป็นอีกไฟล์หนึ่งเสมอทำให้ไฟล์นี้มีขนาดใหญ่ขึ้นทุกครั้งที่เพิ่มโหนดเข้าไปเป็นเหตุให้ต้องมีการอ่านไฟล์ขนาดใหญ่ตลอดเวลาส่งผลให้การทำงานมีความล่าช้ามาก

- **ขาดระบบการทำงานบางอย่างที่จำเป็น**

เนื่องจากขาดฟังก์ชันบางอย่างที่จำเป็นในการสร้างเกม เช่น ฟังก์ชันนำวัตถุ3มิติมาผูกติดกับวัตถุ3มิติอีกชิ้นหนึ่งให้เคลื่อนที่ไปด้วยกัน

- **การจัดรูปแบบโหนดบางโครงสร้างมีความจำเพาะสูง**

โครงสร้างการทำงานของฟังก์ชันต่างๆนั้นไม่สามารถเข้าใจได้ง่าย เนื่องจากต้องเข้าใจโหนดอย่างถ่องแท้ว่าข้อมูลเข้าเป็นชนิดไหนและข้อมูลออกเป็นชนิดอะไรรวมทั้งต้องเข้าใจคุณสมบัติต่างๆที่มีให้ปรับค่าด้วย

### 3.3.2 ปัญหาเกี่ยวกับการนำเข้าวัตถุสามมิติ

- **ไม่มีการนิยามโครงสร้าง direct x-file ที่ชัดเจน**

เนื่องจาก direct x-file มีโครงสร้างที่ไม่เป็นมาตรฐานสากล ทำให้การส่งออกจากโปรแกรมสร้างวัตถุ3มิติ ไม่สามารถรองรับกับ เอนจิน บางตัวได้ ซึ่งต้องมีการทดลองหาส่วนเสริมสำหรับโปรแกรม ในการส่งออกให้เข้ากับ เอนจิน นั้นๆได้

- **ไม่รองรับ Freeze Transformation**

Freeze Transformation คือการกำหนดขนาดและตำแหน่งของ polygon ให้เป็นค่าเริ่มต้น เนื่องจากการแบ่งงานกันทำนั้น ผู้สร้างอาจทำขนาดของวัตถุ3มิติไม่เท่ากันหรือไม่สมดุลกัน ทำให้ต้องมีการปรับเปลี่ยนขนาดให้มีขนาดเริ่มต้นที่สมดุลกัน

- **การนำเข้าวัตถุ 3 มิติใช้ทรัพยากรเครื่องเยอะ**

โปรแกรมจะทำการสร้างโหนดต่างๆขึ้นมามากมาย เมื่อมีการนำเข้าวัตถุ3มิติ ทำให้ต้องใช้ทรัพยากรในการประมวลผลมาก

- **ไฟล์สกุลของวัตถุ 3 มิติที่รองรับของโปรแกรมมีน้อย**

ไฟล์สกุลของวัตถุ 3 มิติสามารถใช้ได้เพียงบางไฟล์สกุลเท่านั้นเช่น direct x-file 3ds ls lwo mot

**3.3.3 ปัญหาเกี่ยวกับระบบควบคุมตัวละครและปัญหาการผูกติดวัตถุ 3 มิติไปด้วยกัน**

เนื่องจากมีความจำเป็นที่บางครั้งต้องทำการผูกติดไปด้วยกันเช่นให้อาวุธติดไปด้วยกันกับมือ ทำให้ต้องมีการใช้สมการทางคณิตศาสตร์ในเรื่องพิกัดเชิงขั้วเข้ามาคำนวณตำแหน่งของกระสุนเมื่อเทียบกับจุดกึ่งกลางตัวของตัวละคร

**3.3.4 ปัญหาเกี่ยวกับการตรวจจับการชนของวัตถุ**

- **ไม่สามารถเช็คการชนระหว่างพื้นผิวของวัตถุทั้ง 2 วัตถุได้**

เนื่องจาก เอนจิน ถูกออกแบบมาให้นำค่าตำแหน่งมาเช็คการชนกับพื้นผิวของวัตถุ 3มิติได้เท่านั้น

- **ปัญหาเช็คการชนวัตถุแบบการเคลื่อนไหว ไม่สามารถกระทำได้**

เนื่องจากโปรแกรมจะเก็บค่าพื้นผิวที่สำหรับเช็คชนไว้เฉพาะเฟรมแรกเท่านั้น ทำให้ต้องมีการสร้างกล่องตรวจการชน ซึ่งเป็นกล่องที่มองไม่เห็นผูกติดไว้กับตัวละครไว้หลายๆ กล่องมาต่อกันตามรูปร่างของตัวละครในแต่ละเฟรมของแต่ละท่าทาง

**3.3.5 ปัญหาเกี่ยวกับการนำเข้าภาพยนตร์และเสียงดนตรี**

- **ไม่ระบุ Header ที่แน่ชัดในการ Encode ไฟล์ภาพยนตร์**

โปรแกรม Quest3D บอกเพียงแค่ว่าสนับสนุนไฟล์สกุลอะไรเท่านั้น แต่ไม่บอกว่าว่าไฟล์สกุลนั้นรองรับโครงสร้างข้อมูลแบบใด

- **ชนิดไฟล์ภาพยนตร์และไฟล์เสียงที่รองรับได้มีน้อย**

ไฟล์ภาพยนตร์และไฟล์เสียงที่รองรับได้แก่ .avi .mpeg .mp3 .wav .midi เท่านั้น ไม่สามารถรับไฟล์ชนิดอื่นที่นอกเหนือจากนี้ได้เช่น .mov .rm .ogm .m2v เป็นต้น

### 3.3.6 ปัญหาเกี่ยวกับโหนดภายใน Quest3D

- **ปัญหาเรื่อง shortcut ของโหนด**

เนื่องจากโปรแกรมเป็นเชิงการลากโหนดทำให้มีความวุ่นวายในการลากโยงถึงกัน โปรแกรมจึงมีฟังก์ชันสร้าง shortcut ซึ่งเสมือนว่าได้โยงไปหาโหนดนั้นจริงๆ แต่ข้อเสียนั้นคือ เมื่อทำการคัดลอกโหนดหากไม่มีโหนดตัวจริงถูกคัดลอกไปด้วย จะทำให้ shortcut ที่สร้างมาทั้งหมดนั้นหายไปทั้งหมด ซึ่งทำให้การรวมงานเป็นเรื่องที่ทำได้อย่างยากลำบากและเสียเวลาในการลากใหม่เป็นอย่างมาก

- **ปัญหาความช้าในการคัดลอกโหนด**

เนื่องจากโปรแกรมจะทำการไล่หาทุกโหนดของโปรแกรมว่ามีโหนดใดบ้างที่สามารถเชื่อมโยงไปหาโหนดตัวต้นฉบับได้ ซึ่ง shortcut ใดที่ไม่สามารถอ้างไปถึงได้โปรแกรมจะทำการตัดทิ้งไปทั้งหมด

- **ปัญหาเรื่องการ public ในการทำ execute fileใน เอนจิน**

การทำ public ต้องมีการรวมไฟล์ย่อยที่เกี่ยวข้องมาให้ครบและต้องอยู่ในแฟ้มเดียวกัน ไม่เช่นนั้นจะไม่มีการทำงานในส่วนที่ขาดไปทำให้เกมไม่สมบูรณ์

- **ปัญหา Error ZIP Compression**

จะเกิดจากการที่ไม่สามารถบีบอัดขนาดของไฟล์โปรแกรมให้มีขนาดเล็กลงได้ ซึ่งจะทำให้ไฟล์มีขนาดใหญ่เกินไป ดังนั้น จึงต้องทำการบันทึกไฟล์โปรแกรมเป็นชื่อใหม่จนกว่าจะทำการ ZIP Compression ได้

- **ปัญหาเรื่องการหา Header ไม่เจอตอน public**

จะเกิดจากการที่บันทึกไฟล์ที่เป็นตัวหลักของโปรแกรมไว้หลายๆไฟล์ภายในโฟลเดอร์เดียวกัน ทำให้ไฟล์ย่อยต่างๆที่ถูกเรียกจากไฟล์โปรแกรมหลักไม่ทราบว่าไฟล์ไหนที่เป็นตัวHeaderที่ทำหน้าที่เรียกไฟล์ย่อยต่างๆ ดังนั้น จึงต้องมีการกดเลือกไฟล์Headerและไฟล์ย่อยทั้งหมดขณะที่ทำการpublicเป็น execute file

# บทที่4
# ผลการทดลองและการวิเคราะห์ปัญหา

ผลการทดลองและการวิเคราะห์ปัญหาที่จะเกิดขึ้นภายในบทนี้นั้นจะแสดงให้เห็นถึงปัญหาที่เป็นอุปสรรคในการทำงาน และขั้นตอนวิธีการแก้ปัญหาและวิเคราะห์ปัญหาต่างๆ โดยทางผู้จัดทำหวังว่าผลการทดลองและปัญหาที่เกิดขึ้น จะเป็นแนวทางในการนำไปพัฒนาต่อไปในอนาคต

**คุณสมบัติของระบบที่จะนำมาทดสอบ**

1. ระบบปฏิบัติการ Microsoft Windows XP version 2005
2. หน่วยประมวลผลกลางAMD x2 4400+ dual core
3. หน่วยความจำ 2GB dual channel
4. การ์ดประมวลกราฟฟิกของ Nvidia 7900 GTX-XFX RAM 512MB
5. โปรแกรม Runtime ของ DirectX ตั้งแต่เวอร์ชัน 9 ขึ้นไป

**ขั้นตอนการดำเนินการทดสอบ**

- ติดตั้งโปรแกรม Runtime DirectX 9.0
- ทำการเปิดไฟล์ Mebius_Combat.exe
- ตรวจสอบว่าฉากเปิดเกมทำงานได้ปกติ
- ตรวจสอบว่าหน้าเมนูสามารถตอบสนองต่อผู้เล่นได้อย่างถูกต้อง
- ตรวจสอบการทำงานของการเลื่อนกล้อง
- ตรวจสอบว่าโมเดลที่ได้สร้างขึ้นเองในตัวเกมนั้น แสดงผลได้อย่างถูกต้อง
- ตรวจสอบการทำงานของโปรแกรมโดยใช้ เมาส์ และ แป้นพิมพ์
- ตรวจสอบการชนระหว่างวัตถุ
- ตรวจสอบการทำงานของระบบปัญญาประดิษฐ์
- ตรวจสอบข้อผิดพลาดภายในโปรแกรม

**จุดประสงค์การดำเนินการทดสอบ**

- โปรแกรมสามารถเล่นได้อย่างไม่มีข้อผิดพลาด
- การทำงานของโปรแกรมในแต่ละเครื่องมีการประมวลผลด้วยความเร็วที่ใกล้เคียงกัน
- การตอบสนองของปัญญาประดิษฐ์เป็นไปตามเงื่อนไขที่กำหนด

- การใช้คำสั่งของปุ่มกดต่างๆมีการตอบสนองต่อผู้เล่นตามรูปแบบที่กำหนดไว้
- ระบบปัญญาประดิษฐ์สามารถตอบสนองต่อผู้เล่นได้เป็นอย่างดี
- จำนวนความผิดพลาดที่เกิดขึ้นควรจะมีน้อยที่สุด

## 4.1 ขั้นตอนการดำเนินการทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็นต้องใช้

ตาราง 4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็น

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็น | 1.ติดตั้งโปรแกรม Runtime ของ DirectX 9.0<br>2.ทำการคัดลอกไฟล์ Mebius_Combat.exe มาลงบนเครื่อง<br>3.ทำการเปิดไฟล์ Mebius_Combat.exe ว่าสามารถทำงานได้ไหม | 1.สามารถติดตั้งโปรแกรม Runtime ของ DirectX 9.0 ได้อย่างไม่มีข้อผิดพลาด<br>2.การคัดลอกไฟล์ Mebius_Combat.exe มายังเครื่องนั้นมีความสมบูรณ์พร้อม<br>3.ผู้ใช้สามารถเปิดไฟล์ Mebius_Combat.exe ได้ |

## 4.2 ขั้นตอนการดำเนินการทดสอบการประมวลผลภายใต้ระบบที่กำหนด

4.2.1 ทำการเปิดโปรแกรม Mebius_Combat.exe

ตาราง 4.2.1 ทดสอบการเปิดโปรแกรม Mebius_Combat.exe

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทำการเปิดโปรแกรมภาย Mebius_Combat.exe | ทำการ ดับเบิ้ลคลิก ไฟล์ Mebius_Combat.exe | หน้าจอโปรแกรมสามารถเปิดขึ้นมาได้อย่างถูกต้องตามที่ได้กำหนดไว้ |

4.2.2 ทดสอบฉากเปิดเกม

ตาราง 4.2.2 ทดสอบฉากเปิดเกม

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบฉากเปิดเกม โดยการรอฉากเปิดเกมจนจบ | รอจนกระทั่งการแสดงฉากเปิดเกมเสร็จสิ้น | หลังจากจบฉากเปิดแล้ว โปรแกรมสามารถเข้าสู่หน้าเมนู ได้ |
| ทดสอบฉากเปิดเกมโดยการกดปุ่มข้ามฉากเปิด | เมื่อเห็นฉากเปิดแล้วจึงทำการกดปุ่มใดๆบนแป้นพิมพ์ | หลังจากกดปุ่มใดๆบนแป้นพิมพ์แล้ว โปรแกรมสามารถเข้าสู่หน้าเมนูได้ |

4.2.3 ทดสอบหน้าเมนู

ตาราง 4.2.3 ทดสอบหน้าเมนู

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบหน้าเมนู | 1.ใช้ mouse คลิกปุ่ม Start<br>2.รอการแสดงฉากเริ่มใหม่อีกครั้ง | 1.โปรแกรมสามารถทำการเข้าสู่ระบบหน้าจอของการต่อสู้ได้<br>2.โปรแกรมสามารถกลับไปแสดงฉากเริ่มเกมใหม่ได้ |

4.2.4 ทดสอบการเลื่อนกล้องบนสนามต่อสู้

ตาราง 4.2.4 ทดสอบการเลื่อนกล้องบนสนามต่อสู้

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการเลื่อนกล้องบนสนามต่อสู้ | รอดูการเคลื่อนไหวของกล้องจนเสร็จสิ้น | กล้องสามารถเคลื่อนที่ตามทิศทางที่กำหนดไว้ได้อย่างถูกต้อง |

4.2.5 ทดสอบการกดปุ่มบนแป้นพิมพ์

ตาราง 4.2.5 ทดสอบการกดปุ่มบนแป้นพิมพ์

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการกดปุ่มบนแป้นพิมพ์ | ทดสอบกดปุ่มบนแป้นพิมพ์ | เกิดการแสดงผลลัพธ์ขึ้นกับตัวละครตามที่ได้กำหนด |

4.2.6 ทดสอบการใช้งานเมาส์

ตาราง 4.2.6 ทดสอบการใช้งานเมาส์

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการเคลื่อนเมาส์ | ทำการเคลื่อนเมาส์ และกดปุ่มบนเมาส์ | เกิดการแสดงผลลัพธ์ขึ้นกับตัวละครตามที่ได้กำหนด |

4.2.7 ทดสอบการประมวลผลของระบบปัญญาประดิษฐ์

ตาราง 4.2.7 ทดสอบการประมวลผลของระบบปัญญาประดิษฐ์

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการประมวลผลของระบบปัญญาประดิษฐ์ | ทำการโต้ตอบกับระบบปัญญาประดิษฐ์ผ่านทางแป้นพิมพ์ และเมาส์ | ระบบปัญญาประดิษฐ์มีการดอบสนองตามที่ได้กำหนดไว้ |

## 4.3 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของเกม

4.3.1 ทดสอบการควบคุมตัวละคร

ตาราง 4.3.1 ทดสอบการควบคุมตัวละคร

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| การเดินหน้า | กดปุ่ม W | ตัวละครผู้เล่นแสดงท่าเดินหน้าพร้อมกับเคลื่อนที่ไปข้างหน้า |
| การถอยหลัง | กดปุ่ม S | ตัวละครผู้เล่นแสดงท่าเดินถอยหลังพร้อมกับเคลื่อนที่ไปข้างหลัง |
| การเดินไปทางซ้าย | กดปุ่ม A | ตัวละครผู้เล่นแสดงท่าเดินไปทางซ้าย พร้อมกับเคลื่อนที่ไปทางซ้าย |
| การเดินไปทางขวา | กดปุ่ม D | ตัวละครผู้เล่นแสดงท่าเดินไปทางขวา พร้อมกับเคลื่อนที่ไปทางขวา |
| การเดินไปทางซ้ายบน | กดปุ่ม W พร้อมกับปุ่ม A | ตัวละครผู้เล่นแสดงท่าเดินไปทางซ้าย พร้อมกับเคลื่อนที่ไปทางซ้ายบน |
| การเดินไปทางขวาบน | กดปุ่ม W พร้อมกับปุ่ม D | ตัวละครผู้เล่นแสดงท่าเดินไปทางขวา พร้อมกับเคลื่อนที่ไปทางขวาบน |
| การเดินไปทางซ้ายล่าง | กดปุ่ม S พร้อมกับปุ่ม A | ตัวละครผู้เล่นแสดงท่าเดินไปทางซ้าย พร้อมกับเคลื่อนที่ไปทางซ้ายล่าง |

| | | |
|---|---|---|
| การเดินไปทางขวาล่าง | กดปุ่ม S พร้อมกับปุ่ม D | ตัวละครผู้เล่นแสดงท่าเดินไปทางขวา พร้อมกับเคลื่อนที่ไปทางขวาล่าง |
| การหันไปทางซ้าย | เคลื่อน เมาส์ไปทางซ้าย | ตัวละครผู้เล่นและมุมมองหันไปทางซ้าย |
| การหันไปทางขวา | เคลื่อน เมาส์ไปทางขวา | ตัวละครผู้เล่นและมุมมองหันไปทางขวา |
| การหันขึ้นข้างบน | เคลื่อน เมาส์ขึ้น | ตัวละครผู้เล่นและมุมมองหันขึ้นไปทางด้านบน |
| การหันลงข้างล่าง | เคลื่อน เมาส์ลง | ตัวละครผู้เล่นและมุมมองหันลงไปทางด้านล่าง |
| การกดพุ่ง | 1.กดปุ่ม Shift ซ้าย พร้อมกันกับปุ่มทิศทางการเดิน<br>2.กดปุ่ม Shift ซ้าย เฉยๆ | 1.ตัวละครผู้เล่นแสดงท่าทางการพุ่งตามทิศทางการเคลื่อนที่พร้อมทั้งเคลื่อนที่เร็วขึ้น และสูญเสียค่าพลังงานของหุ่นยนต์<br>2.ตัวละครผู้เล่นแสดงท่าปกติพร้อมทั้งไม่แสดงการเคลื่อนที่และไม่สูญเสียค่าพลังงานของหุ่นยนต์ |
| การกดกระโดด | 1.กดปุ่ม C เฉยๆ<br>2.กดปุ่ม C พร้อมกันกับกลุ่มทิศทางการเคลื่อนที่ | 1.ตัวละครของผู้เล่นแสดงท่าทางการกระโดด และตำแหน่งตัวละครจะมีการสูงขึ้นและลดลง<br>2.ตัวละครของผู้เล่นแสดงท่าทางการกระโดด และตำแหน่งตัวละครตัวละครจะมีการสูงขึ้นและลดลงพร้อมทั้งมีการเคลื่อนที่ไปยังทิศทางที่กดปุ่มการเคลื่อนที่ |

| | | |
|---|---|---|
| การยิง | คลิกเมาส์ที่ปุ่มซ้าย ขณะอยู่ในโหมดถืออาวุธ | กระสุนถูกยิงออกไปจากปลายอาวุธตัวละครของผู้เล่น |
| การเปลี่ยนโหมด | 1.คลิกเมาส์ที่ปุ่มขวา ขณะไม่ได้ทำการกดปุ่มที่เกี่ยวข้องกับการเคลื่อนที่<br>2.คลิกเมาส์ที่ปุ่มขวา ขณะที่มีการกดปุ่มที่เกี่ยวข้องกับการเคลื่อนที่ | 1.ตัวละครของผู้เล่นแสดงท่าการเปลี่ยนอาวุธ พร้อมทั้งมีการเปลี่ยนโหมด<br>2.ตัวละครของผู้เล่นจะไม่มีการเปลี่ยนโหมด |

4.3.2 ทดสอบการตอบสนองของระบบปัญญาประดิษฐ์

ตาราง 4.3.2 ทดสอบการตอบสนองของระบบปัญญาประดิษฐ์

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการมองเห็นของระบบปัญญาประดิษฐ์ | 1.บังคับตัวละครให้ระบบปัญญาประดิษฐ์เห็นโดยไม่มีสิ่งกีดขวาง<br>2.บังคับตัวละครให้ระบบปัญญาประดิษฐ์เห็นโดยมีสิ่งกีดขวาง | 1.ระบบปัญญาประดิษฐ์สามารถรับรู้ได้ว่ามองเห็น<br>2.ระบบปัญญาประดิษฐ์สามารถรับรู้ได้ว่ามองไม่เห็น |
| ทดสอบการหันมองตามผู้เล่นของระบบปัญญาประดิษฐ์ | เคลื่อนตัวละครผู้เล่นไปยังตำแหน่งต่างๆ | ระบบปัญญาประดิษฐ์หันทิศทางการมองตามตำแหน่งผู้เล่นที่เปลี่ยนไป |
| ทดสอบการเคลื่อนที่ของระบบปัญญาประดิษฐ์ | สังเกตการณ์การเคลื่อนที่ของระบบปัญญาประดิษฐ์ | ระบบปัญญาประดิษฐ์สามารถเคลื่อนที่ไปในทิศทางต่างๆ |
| ทดสอบการยิงกระสุนของระบบปัญญาประดิษฐ์ | สังเกตพฤติกรรมขณะที่ระบบปัญญาประดิษฐ์อยู่ในโหมดถืออาวุธ | ระบบปัญญาประดิษฐ์มีการยิงกระสุน |
| ทดสอบการเปลี่ยนโหมดของระบบปัญญาประดิษฐ์ | สังเกตพฤติกรรมของระบบปัญญาประดิษฐ์ในขณะที่มีพลังงานเหลือมาก และพลังงานเหลือน้อย | ระบบปัญญาประดิษฐ์มีการเปลี่ยนโหมด |

| | | |
|---|---|---|
| ทดสอบการป้องกันของระบบปัญญาประดิษฐ์ | สังเกตพฤติกรรมของระบบปัญญาประดิษฐ์ในขณะที่อยู่ในโหมดปกติ และมีกระสุนพุ่งเข้าใส่ | ระบบปัญญาประดิษฐ์มีการทำการป้องกัน |
| ทดสอบการกระโดดของระบบปัญญาประดิษฐ์ | สังเกตพฤติกรรมของระบบปัญญาประดิษฐ์ | ระบบปัญญาประดิษฐ์มีการกระโดด |

4.3.3 ทดสอบกระบวนการภายในของระบบ

ตาราง 4.3.3 ทดสอบกระบวนการภายในของระบบ

| การทดสอบ | ขั้นตอนการทดสอบ | ผลการทดสอบ |
|---|---|---|
| ทดสอบการชนกับสิ่งต่างๆของตัวละครผู้เล่น | บังคับตัวละครเข้าไปชนกับสิ่งของต่างๆ | ตัวละครผู้เล่นไม่สามารถเดินทะลุสิ่งของต่างๆได้ |
| ทดสอบการชนกับสิ่งต่างๆของตัวละครระบบปัญญาประดิษฐ์ | สังเกตตัวละครของระบบปัญญาประดิษฐ์ขณะทำการเคลื่อนที่ | ตัวละครระบบปัญญาประดิษฐ์ไม่สามารถเดินทะลุสิ่งของต่างๆได้ |
| ทดสอบการโดนโจมตีด้วยกระสุนของตัวละครผู้เล่น | สังเกตขณะที่ตัวละครของผู้เล่นโดนกระสุนกระทบลงบนตัวละคร | แถบพลังชีวิตของผู้เล่นมีการเปลี่ยนแปลงค่า คือ ลดลง |
| ทดสอบการโดนโจมตีด้วยกระสุนของตัวละครระบบปัญญาประดิษฐ์ | สังเกตขณะที่ตัวละครระบบปัญญาประดิษฐ์โดนกระสุนกระทบลงบนตัวละคร | แถบพลังชีวิตของระบบปัญญาประดิษฐ์มีการเปลี่ยนแปลงคือ ลดลง |
| ทดสอบการแพ้หรือชนะของเกม | 1.เมื่อพลังชีวิตของผู้เล่นหมดลง<br>2.เมื่อพลังชีวิตของระบบปัญญาประดิษฐ์หมดลง<br>3.เมื่อเวลาที่กำหนดไว้หมดลง และ พลังชีวิตของผู้เล่นเหลือมากกว่าระบบปัญญาประดิษฐ์<br>4.เมื่อเวลาที่กำหนดไว้หมดลง และพลังชีวิตของระบบปัญญาประดิษฐ์เหลือมากกว่า | 1.แสดงให้เห็นว่าระบบปัญญาประดิษฐ์เป็นฝ่ายชนะ<br>2.แสดงให้เห็นว่าผู้เล่นเป็นฝ่ายชนะ<br>3.แสดงให้เห็นว่าผู้เล่นเป็นฝ่ายชนะ<br>4.แสดงให้เห็นว่าระบบปัญญาประดิษฐ์เป็นฝ่ายชนะ<br>5.แสดงให้เห็นว่าระบบปัญญาประดิษฐ์เป็นฝ่ายชนะ |

| | | |
|---|---|---|
| | พลังชีวิตของผู้เล่น<br>5.เมื่อเวลาที่กำหนดหมดลง และพลังชีวิตของผู้เล่นที่เหลือมีค่าเท่ากับพลังชีวิตของระบบปัญญาประดิษฐ์ | |
| ทดสอบการเพิ่มพลังงานของตัวละครผู้เล่น | สังเกตขณะที่พลังงานของผู้เล่นไม่ได้เต็ม และไม่ได้อยู่ระหว่างการใช้งาน | หลอดพลังงานของผู้เล่นมีลักษณะค่อยๆเพิ่มขึ้นเรื่อยๆที่ละนิด |
| ทดสอบการระเบิด | สังเกตขณะกระสุนกระทบลงบนตัวละคร | เกิดสะเก็ดระเบิด ณ จุดที่กระสุนชนกับตัวละคร |

**4.4 ปัญหาข้อผิดพลาดและข้อเสนอแนะ**

ระบบของเกมที่ทำขึ้นนั้นมีขนาดค่อนข้างใหญ่ จึงมีความเป็นไปได้ว่า อาจจะยังมีข้อผิดพลาดที่อาจมองไม่เห็นอยู่ การทดสอบที่ได้กระทำไปนั้นยังคงไม่สมบูรณ์แบบ และมีการพบปัญหาบางจุดที่ยังไม่สามารถแก้ไขได้ จึงจะได้กล่าวไว้ในที่นี้

4.4.1 ปัญหาเกี่ยวกับการที่กระสุนตัดผ่านกล่องตรวจสอบการชนภายในตัวละครแต่ไม่เกิดการชน

มีหลายครั้งซึ่งเมื่อกระสุนได้เคลื่อนตัดเข้ากับกล่องตรวจสอบการชนที่ฝังไว้บนตัวละครแล้วไม่เกิดการชน อันเนื่องมาจากว่าการเพิ่มค่าของกระสุนนั้นใช้การเพิ่มค่าที่ได้มาจากการคำนวณทางสมการ ซึ่งยิ่งต้องการให้กระสุนมีความเร็วมากเท่าใดก็ยิ่งเพิ่มโอกาสที่จะทำให้กระสุนเคลื่อนที่ข้ามพื้นที่ที่กล่องดักจับการชนวางอยู่มากเท่านั้น ปัญหานี้ยังไม่พบวิธีแก้ที่แน่นอน ส่วนวิธีเสนอแนะนั้น คาดว่าหากจะแก้ให้ได้ คงจะต้องเขียนฟังก์ชันบางประการขึ้นมาเพื่อช่วงในการตรวจจับการชนให้แม่นยำ

4.4.2 ปัญหาเกี่ยวกับพฤติกรรมที่ผิดแปลกบางประการของระบบปัญญาประดิษฐ์

ระบบปัญญาประดิษฐ์ที่ได้สร้างขึ้นมีความซับซ้อนด้านโครงสร้างค่อนข้างสูง ทำให้การทำงานบางประการอาจมีข้อผิดพลาด เช่น บางครั้งหากตัวละครของระบบปัญญาประดิษฐ์นั้นเคลื่อนที่เข้าไปอยู่บริเวณมุมฉาก อาจทำให้ระบบปัญญาประดิษฐ์เดินอยู่บริเวณมุมอับนานผิดปกติ หรือกระทั่งการกลับคืนสู่การมองเห็นตัวผู้เล่นนั้นทำได้ล่าช้ากว่าที่ควรจะเป็น ล้วนแล้วแต่เป็นปัญหาที่มีโอกาสเกิดขึ้นได้ทั้งสิ้น แต่ทั้งนี้ไม่ได้เกิดขึ้นบ่อย และปัญหาก็ไม่ได้รุนแรงเกินกว่าที่จะ

ยอมรับได้ ข้อเสนอแนะเพื่อแก้ไขนั้นคือ ต้องมีเวลาในการตรวจสอบมากกว่านี้ เนื่องจากเวลาที่มีนั้นค่อนข้างน้อย ทำให้ข้อผิดพลาดที่เกิดขึ้นในส่วนที่มีความซับซ้อนสูงมากนั้นแก้ไขได้ยาก

4.4.3 ปัญหาด้านทรัพยากรฮาร์ดแวร์ที่จำเป็นในการเล่นเกม

สำหรับวัตถุสามมิติที่ใช้ภายในเกมนั้น มีข้อมูลที่ค่อนข้างใหญ่ ผนวกกับการเขียนระบบปัญญาประดิษฐ์นั้นทำให้เครื่องที่จะเล่นเกมนี้ได้ดีนั้น จำเป็นที่จะต้องมีทรัพยากรฮาร์ดแวร์ที่เหมาะสม ซึ่งถ้าเทียบตามมาตรฐานของเครื่องคอมพิวเตอร์ที่บุคคลทั่วไปพึงจะมีได้นั้นพบว่า อาจทำให้มีปัญหาขณะทำการเล่นเกมนี้ได้ เช่นอาการกระตุก หรือการประมวลเชิงอัลกอริธึ่มต่างๆที่ไม่ทันท่วงที และอาจทำให้ตัวตรวจจับเวลาที่อยู่ภายในระบบปัญญาประดิษฐ์นั้นทำงานล่าช้าตามไปด้วย ส่งผลให้เกมเสียสมดุล และผิดไปจากหลักการที่ได้วางไว้ ฉะนั้นจึงแนะนำว่า ผู้ที่จะนำเกมนี้ไปเล่นนั้นควรมีการจอที่มีหน่วยความจำไม่ต่ำกว่า 256MB และมีหน่วยประมวลผลกลางที่มีประสิทธิภาพไม่ต่ำไปกว่า pentium4

## 4.5 การประเมินประสิทธิภาพของเกม

จากการทดสอบระบบที่ผ่านมาพบว่า ระบบมีประสิทธิภาพที่ค่อนข้างน่าพึงพอใจ เนื่องจากสภาวะที่กดดันและเวลาที่มีค่อนข้างน้อยเป็นอุปสรรคสำคัญต่อคุณภาพงาน ข้อเสียของเกมนี้คงจะอยู่ที่ทรัพยากรฮาร์ดแวร์ที่ต้องใช้ ซึ่งอาจจะทำให้จำนวนคนที่จะเล่นเกมนี้ได้นั้นมีจำนวนกลุ่มที่ไม่มากเท่าที่ควร หากมีเวลาในการทำงานมากกว่านี้ คุณภาพงานที่ได้ก็น่าจะดีกว่านี้ กระนั้นโดยรวมแล้วระบบทำงานได้ค่อนข้างดี ถึงแม้ลูกเล่นด้านอาวุธของตัวละครจะน้อยไปบ้าง หากแต่ว่าระบบปัญญาประดิษฐ์ที่ได้สร้างขึ้นก็สามารถโต้ตอบได้อย่างเหมาะสมและคงจะสร้างความสนุกสนานให้คนแต่ละคนได้มาก

# บทที่ 5
# สรุปผลการดำเนินงานและข้อเสนอแนะ

## 5.1 สรุปผลการดำเนินงาน

### 5.1.1 การศึกษาข้อมูลเพื่อทำเกมและอุปสรรคในการศึกษา

การศึกษาและรวบรวมข้อมูลสำหรับการพัฒนาเกมนี้นั้นประกอบขึ้นจากกระบวนการอันหลากหลาย เริ่มจากการค้นหาข้อมูลทางอินเตอร์เนต โดยค้นคว้าว่าเอนจินตัวไหนที่จะมีความเหมาะสมสำหรับที่จะนำมาใช้ทำเกมในแนวที่ได้กำหนดไว้ และค้นคว้าวิธีการที่จะจัดสร้างวัตถุสามมิติขึ้นใช้เอง ในช่วงแรกของการค้นคว้านั้น ทางกลุ่มได้ตัดสินใจเลือก irrlicht เอนจินเป็นเอนจินหลัก และเลือกโปรแกรม maya เป็นโปรแกรมสำหรับจัดสร้างวัตถุสามมิติภายในเกม ซึ่งก็พบปัญหามากมาย เนื่องจากการจะใช้วัตถุสามมิติที่สร้างขึ้นเองนั้นทำได้ยาก เพราะแต่ละเอนจินก็รองรับโครงสร้าง directX ไฟล์ที่แตกต่างกัน ซึ่งหากไม่ทราบถึงข้อจำกัดของเอนจินในการรองรับวัตถุสามมิติที่ได้สร้างขึ้นจะเกิดข้อผิดพลาดได้ทันที ในขณะที่มีการใช้งานโปรแกรม maya เราต้องรู้ข้อจำกัดของ directX ไฟล์ โดยห้ามใช้เครื่องมือหรือกรรมวิธีใดใน maya บ้าง ข้อมูลที่หาได้ในอินเตอร์เนตจึงไม่เพียงพอ เพราะว่าการหาข้อมูลเรื่องนี้ทำได้ยาก มีคนมากมายที่ประสบปัญหาไม่สามารถที่จะนำวัตถุสามมิติที่ตัวเองสร้างเข้าใช้งานภายในเอนจินได้ ประกอบกับ irrlicht เอนจินมีข้อจำกัดในหลายๆประการที่ค่อนข้างมาก จึงได้มีการตัดสินใจกันว่าจะหาเอนจินที่เหมาะสมในขณะนั้นเอง อุทยานการเรียนรู้ TK-Park ได้มีการเปิดคอร์สอบรมผู้สร้างเกมเป็นเวลาราว 2 เดือน ซึ่งทำให้เกิดการตัดสินใจว่าจะส่งคนบางส่วนภายในกลุ่มไปเรียนที่คอร์สนี้ ในวันปฐมนิเทศการเปิดตัวคอร์ส TK-Park นั้น เราได้พบกับเอนจิน Quest3D เป็นครั้งแรก ซึ่งเป็นเอนจินที่ประมวลผลภาพ3มิติได้อย่างสวยงาม จึงได้มีการตัดสินใจภายในกลุ่มว่าจะมีการเลือกใช้เอนจินตัวนี้เป็นเอนจินสำหรับการทำโปรเจค แต่ก็ประสบปัญหาในการใช้อนิเมทโมเดลกับ Quest3D เอนจิน ทางบริษัท Imaginmax และบริษัท VR จึงต้องมีการประสานงานเพื่อหาทางแก้ไขปัญหาร่วมกัน จนในที่สุดก็สามารถหาวิธีที่จะใช้ อนิเมทโมเดลได้

การใช้โมเดลที่สร้างขึ้นเองในตอนนั้นยังไม่อาจประสบความสำเร็จได้ เนื่องจากว่าข้อจำกัดในการทำโมเดลสำหรับ Quest3D มีมากมาย และทางผู้สอนที่อยู่ในโครงการ TK-Park ทุกคนก็ไม่ทราบถึงข้อจำกัดนี้เช่นกัน ทำให้ในการดำเนินงานสร้างวัตถุสามมิติภายในกลุ่มนั้นเป็นไปด้วยความยากลำบาก เพราะต้องนั่งทดสอบในขณะดำเนินการเพื่อให้รู้ถึงข้อผิดพลาดอันมากมายที่อาจเกิดขึ้นได้ จนในที่สุดเราก็ทราบถึงข้อจำกัดที่จะทำให้เราสามารถที่จะสร้างโมเดลไว้เพื่อใช้ภายในเกมเอนจิน Quest3D ได้

Quest3D เป็นเอนจินที่ดี แต่เมื่อได้ศึกษาและใช้งานมาได้ในระดับหนึ่ง พบว่า เอนจินตัวนี้ไม่ใช่เอนจินที่เหมาะสำหรับการทำเกมที่มีโครงสร้างซับซ้อน เพราะว่า Quest3D เอนจินเป็นโปรแกรมที่เหมาะสำหรับการทำสถานที่จำลองต่างๆ ซึ่งมีการทำงานที่ไม่ซับซ้อนจนเกินไป แต่กระนั้น Quest3D เอนจินก็ยังเป็นเอนจินที่ดีพอที่จะให้ทางกลุ่มเลือกใช้งานได้ ซึ่งการเริ่มศึกษา Quest3D นั้นกระทำจากคู่มือเริ่มต้น ซึ่งเป็นไฟล์สกุล pdf ที่มากับโปรแกรมเป็นอย่างแรก ประกอบกับเอกสารการศึกษาที่ได้มาจากคอร์สการเรียนที่ TK-Park ทำให้การดำเนินการที่จะเริ่มใช้งาน Quest3D นั้นสามารถทำงานได้ง่ายขึ้น จึงเป็นที่มาของการเลือกใช้เอนจิน Quest3D และ โปรแกรมสำหรับสร้างวัตถุสามมิติ maya

**5.1.2 การสร้างองค์ประกอบที่จำเป็นต่างๆภายในเกม**

สำหรับการสร้างวัตถุสามมิตินั้นจะสร้างขึ้นจากโปรแกรม maya โดยมีการตัดสินใจกันว่าวัตถุทุกชิ้นที่ใช้ในเกมนั้นจะทำการสร้างขึ้นเองทั้งหมด ไม่ว่าจะเป็นตัวละคร หรือ ฉาก ในส่วนของเอฟเฟกต่างๆนั้นจะสร้างจาก Quest3D เอนจินซึ่งมีการรองรับอยู่ภายในตัวโปรแกรม เช่น เอฟเฟกกระสุน หรือแสงสีต่างๆ การสร้างภาพอินเทอร์เฟสที่จำเป็น หรือกระทั่ง texture ที่ต้องใช้กับโมเดลนั้นจะใช้โปรแกรม Photoshop ซึ่งสามารถจัดการเกี่ยวกับภาพได้เป็นอย่างดี ส่วนในเรื่องเกี่ยวกับเสียงจะใช้การนำเข้าเสียงจากที่อื่น เช่น เพลงจากเกมที่ชื่อชอบ และมีการสร้างฉากเริ่มเกม ซึ่งจะใช้โปรแกรม maya และโปรแกรม Adobe Premiere

**5.1.3 การดำเนินการพัฒนาเกมและอุปสรรค**

เป็นขั้นตอนที่ยากลำบากที่สุด ปัญหาที่พบจะเพิ่มขึ้นตามความก้าวหน้าของงานที่ได้ทำไป แม้ว่าการพัฒนาด้วย Quest3D นั้น ผู้พัฒนาไม่จำเป็นที่จะต้องเขียนภาษาโปรแกรม แต่ก็ไม่ได้ทำให้ความยากในการทำงานลดลง เพราะฟังก์ชันที่มีอยู่ใน Quest3D หลายตัวนั้น มีความยืดหยุ่นที่ค่อนข้างน้อย ทำให้การดึงมาใช้นั้นเกิดข้อผิดพลาดที่ไม่อาจแก้ไขได้หลายประการ จึงต้องมีการเปลี่ยนโครงสร้าง เพื่อจัดการและต้องทำการปรับปรุงใหม่ ทำให้เสียเวลาเป็นอย่างมาก มีการทำงานหลายส่วนที่ต้องนำความรู้ทางคณิตศาสตร์เข้ามาประยุกต์เพื่อแก้ปัญหากัน การวางเกมให้ออกมาเป็นได้เหมือนกฎ กติกา และระบบที่ได้วางไว้เป็นไปแบบค่อนข้างยาก อันเนื่องมาจากเวลาที่มีอยู่ค่อนข้างน้อย

เวลาทำงานเกือบทั้งหมดจะเสียไปในส่วนของการทำงานกับโปรแกรม Quest3D คือการสร้างฟังก์ชันต่างๆ และการรวมระบบในแต่ละส่วนที่ได้ช่วยกันพัฒนาแยกส่วนเอาไว้ เช่นระบบปัญญาประดิษฐ์ซึ่งประกอบไปด้วยการทำงานร่วมกันถึง 9 ส่วนเป็นต้น กระบวนการพัฒนาเริ่มจากการทำระบบเบื้องต้นให้เสร็จสิ้นก่อน เช่น การใส่ตัวละคร การใส่ฉาก การจัดแสง การทำสมการการยิง การทำสมการย้ายกระสุน การบริหารท่าทางตัวละคร การวางตำแหน่งจุดที่เกิดการชนและ

ไม่ชน จากนั้นจึงทำการพัฒนาระบบปัญญาประดิษฐ์ขึ้นมา โดยทำการแยกส่วนการพัฒนาของระบบปัญญาประดิษฐ์ออกเป็นหลายส่วน เช่น ส่วนควบคุมความประพฤติ ส่วนควบคุมการยิง ส่วนควบคุมการเปลี่ยนโหมด จากนั้นกระจายแยกกันไปพัฒนาในแต่ละส่วน แล้วจึงนำกลับมารวมกันอีกครั้งหนึ่ง เพื่อให้ระบบเกิดความสมบูรณ์

**5.1.4 การดำเนินการแก้ไขปัญหา และข้อบกพร่องที่เกิดขึ้น รวมถึงเก็บรายละเอียดงาน**

ดังที่ได้กล่าวไปว่า ขั้นตอนในส่วนของการพัฒนานั้นมีความซับซ้อนสูงมาก ทำให้เมื่อทำการรวมระบบหรือเพิ่มเติมระบบก็อาจก่อให้เกิดปัญหาใหญ่เช่นกัน หลังจากที่ทำการรวมระบบแล้วจึงต้องมีการตรวจสอบข้อบกพร่องที่เกิดขึ้นและทำการแก้ไข เนื่องด้วยว่า ส่วนพัฒนาระบบนั้นเป็นเหมือนการพัฒนาฟังก์ชันการทำงานในส่วนต่างๆ ฉะนั้น จึงมีบางส่วนที่ต้องพัฒนาภายหลังจากที่ทำโครงระบบหลักปัญญาประดิษฐ์ของเกม เช่น การแพ้ชนะ การเชื่อมหน้าจอเมนู การวนกลับไปที่ฉากเปิด หรือการเพิ่มขึ้นของพลังงานตัวละคร หรือการลดลงของพลังชีวิตตัวละครเป็นต้น ซึ่งรายละเอียดเหล่านี้จะถูกเพิ่มเติมให้สมบูรณ์ในขั้นตอนนี้

**5.1.5 การตรวจสอบข้อจำกัดของโปรแกรมและตรวจเช็คข้อผิดพลาดที่ยอมรับได้**

ในขั้นตอนนี้จะเป็นการนำระบบที่คาดว่าน่าจะสมบูรณ์ไปทดสอบเล่นจริง ว่าระบบที่ได้พัฒนามามีประสิทธิภาพจริงหรือไม่ และยังมีข้อผิดพลาดใดๆหลงเหลืออยู่อีก ในขั้นตอนนี้หากพบข้อผิดพลาดที่ไม่ร้ายแรงก็จะไม่ทำการแก้ไข โดยดูจากตัวข้อผิดพลาดว่ามันกระทบต่อประสิทธิภาพของระบบมากหรือน้อยเพียงใด ถ้าหากว่าเป็นสิ่งที่กระทบต่อระบบอย่างรุนแรงก็จะต้องย้อนกลับไปในขั้นตอนที่ผ่านมา คือ การกลับไปแก้ไขส่วนที่ผิดพลาด ซึ่งถ้าหากแก้ไขได้ตามที่ต้องการแล้วก็เท่ากับว่าได้ระบบเกมที่สมบูรณ์

**5.1.6 แนวทางการพัฒนาต่อในอนาคต**

เกม3มิติที่ถูกสร้างโดยใช้เอนจินQuest3D เป็นอีกแนวทางหนึ่งที่สามารถนำแนวคิดและหลักการทางปัญญาประดิษฐ์ที่ได้ไปใช้พัฒนาต่อให้เกมมีความสมบูรณ์มากยิ่งขึ้น โดยสามารถนำไปพัฒนาต่อในด้านปัญญาประดิษฐ์ให้มีความฉลาดมากยิ่งขึ้น มีการเรียนรู้และจดจำข้อมูลเพื่อใช้ในการวิเคราะห์รูปแบบการต่อสู้ ในส่วนทางด้านคณิตศาสตร์ สามารถนำมาใช้ในการคำนวณวิถีการยิงกระสุนในรูปแบบต่างๆ หรืออาจจะนำหลักการดังกล่าวมาใช้ในการสร้างเกมโดยใช้เอนจินเกมตัวอื่นก็ได้

สำหรับผู้ที่สนใจในการสร้างเกม3มิติ เอนจินQuest3Dเป็นเอนจินอีกตัวหนึ่งที่เหมาะสำหรับใช้ในการสร้างสรรผลงานเกมให้ดียิ่งขึ้น

# เอกสารอ้างอิง

[1] Alex J. Champandard "The Rule-Based System" [Online]. Available: http://aigamedev.com/

[2] Act-3D B.V. "Quest3D" [Online]. http://www.quest3d.com/

[3] รศ.ศรีบุตร แววเจริญ และ ผศ.ดร.ชนศักดิ์ บ่ายเที่ยง. "เรขาคณิตวิเคราะห์และการเขียนกราฟ2มิติและ3มิติ". ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์,สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ

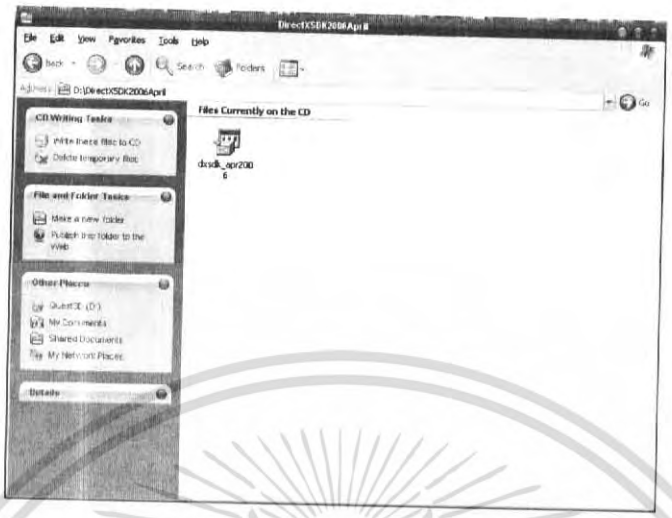
[4] สุวพาส ดวงจิตต์งาม. "การใช้โปรแกรม maya เพื่อการสร้างวัตถุสามมิติสำหรับเกม". เอกสารประกอบการอบรม โครงการสร้างเกมสร้างคนสร้างงาน ,TKpark

# ภาคผนวก ก.

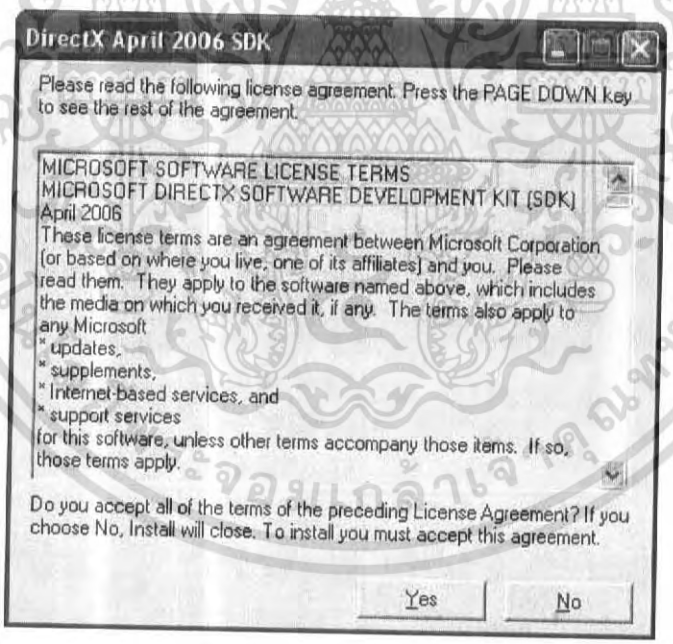
# การติดตั้งโปรแกรม DirectX April 2006 SDK

ขั้นตอนการติดตั้ง

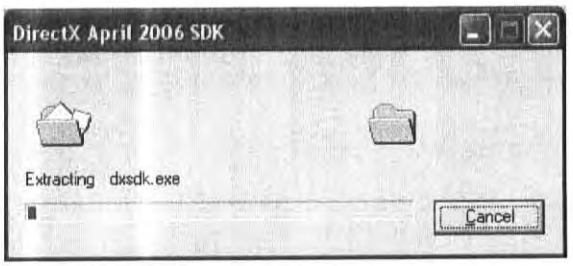
1. ใส่แผ่นโปรแกรม จากนั้นทำการคลิกที่ตัวdxsdk_apr2006.exe ดังในรูป



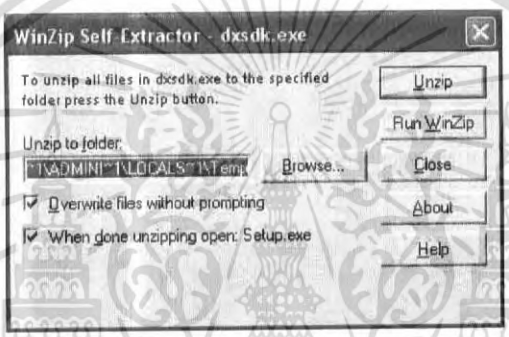

2. ระบบจะแสดงหน้าจอให้คลิกYes



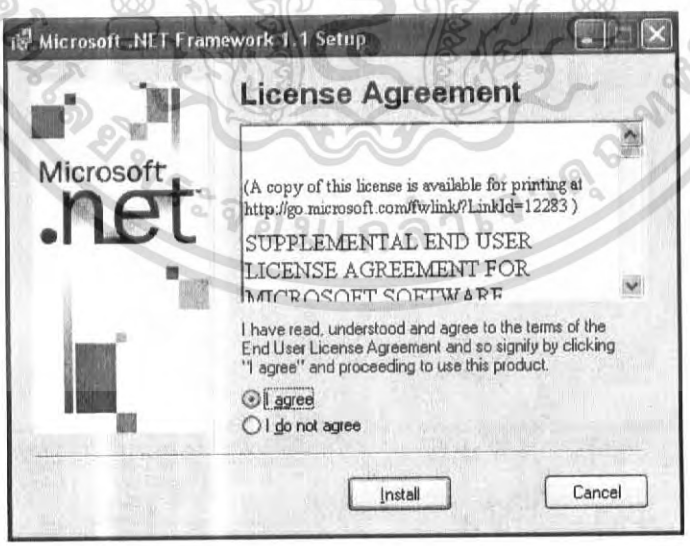
3. ขณะที่ติดตั้งDirectX April 2006 SDKจะแสดงสถานะการติดตั้ง



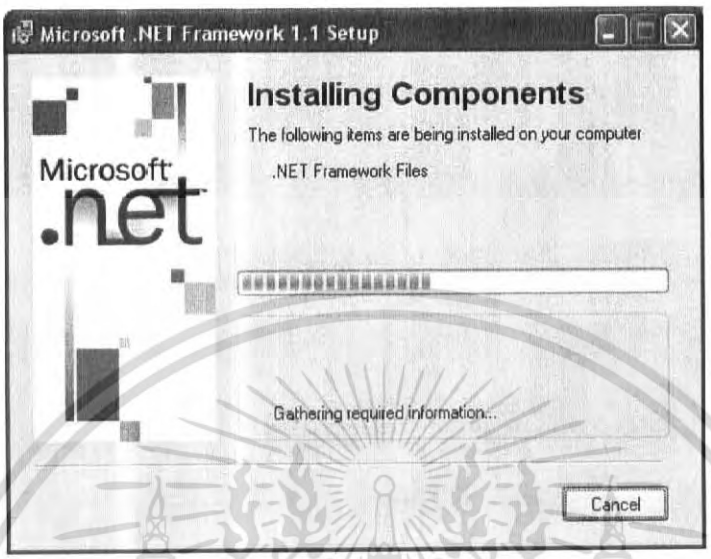

4. แสดงรายละเอียดของลักษณะที่จะติดตั้ง รวมถึงพาธที่ติดตั้งโปรแกรม คลิกUnzip




5. ให้เลือก I agree จากนั้นคลิกInstall

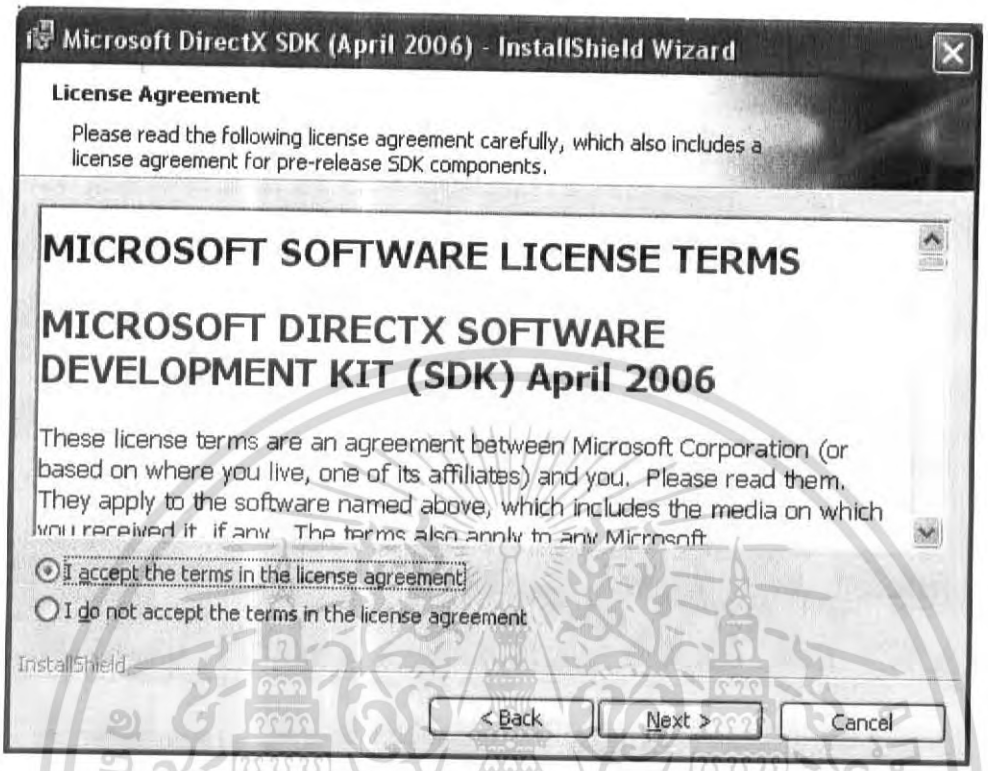


6. ขณะที่ติดตั้ง Microsoft.NET Framework 1.1จะแสดงสถานะการติดตั้ง

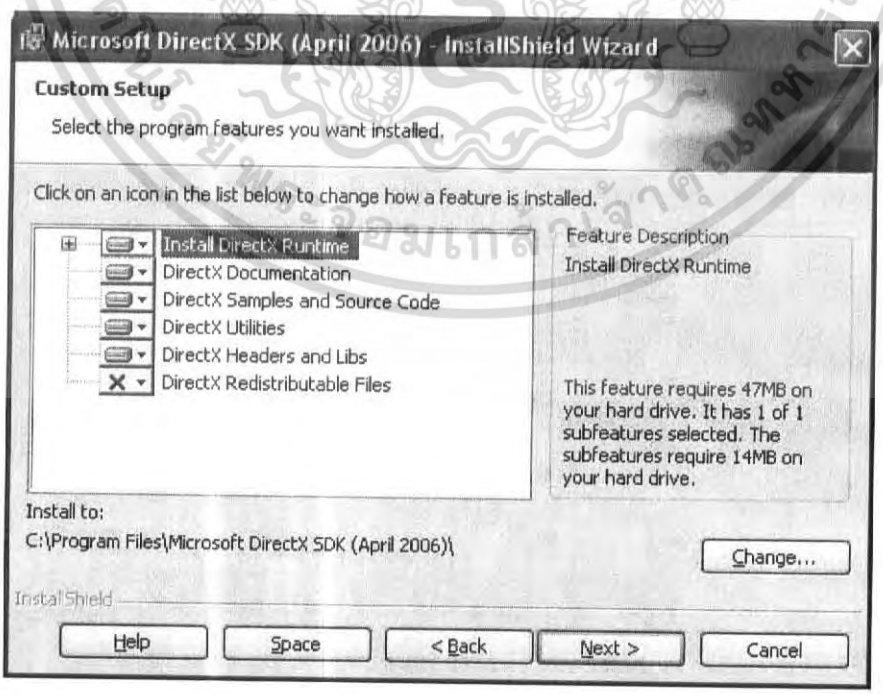



7. กดปุ่มNext เพื่อทำการInstallShield Wizard สำหรับ Microsoft DirectX SDK (April 2006)

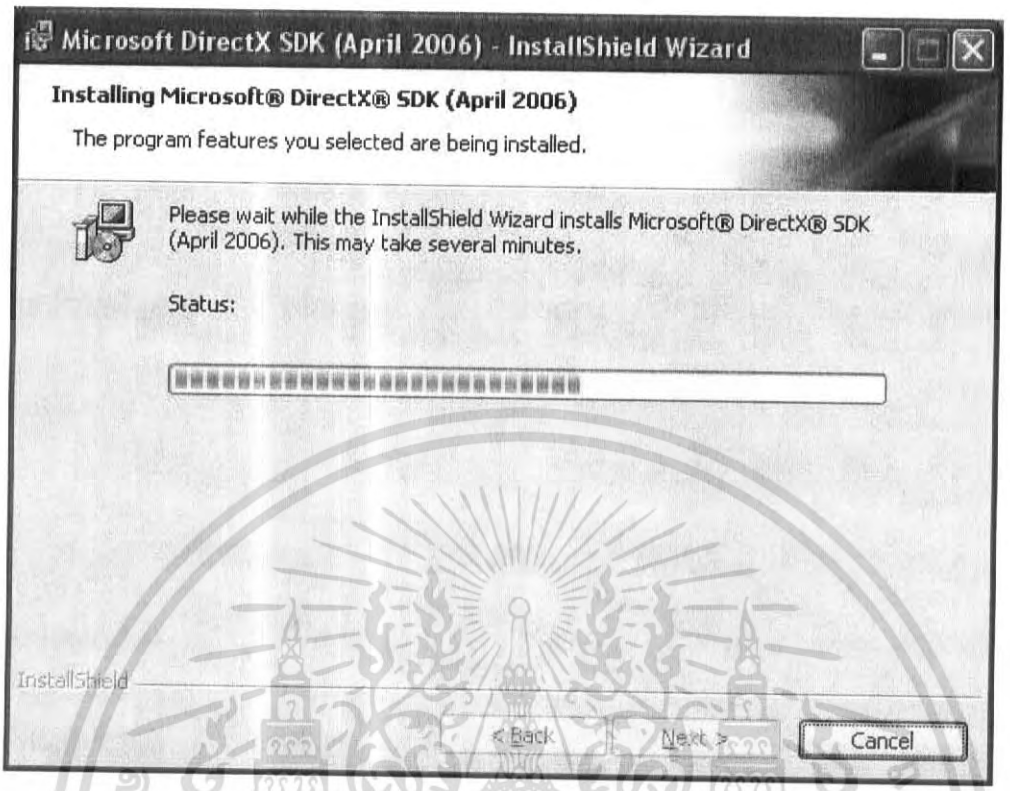
8. กดเลือก I accept the terms in the license agreement จากนั้นกดปุ่ม Next



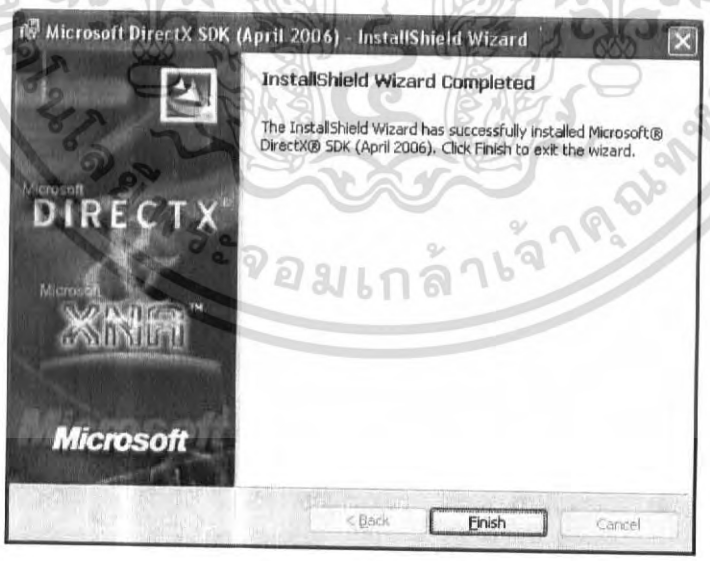

9. แสดงรายละเอียดของลักษณะที่จะติดตั้ง รวมถึงพาธที่ติดตั้งโปรแกรม คลิก Next



10. ขณะที่ติดตั้ง InstallShield Wizard จะแสดงสถานะการติดตั้ง




11. คลิกปุ่ม Finish เมื่อทำการติดตั้งDirectX SDK April 2006เสร็จเรียบร้อย

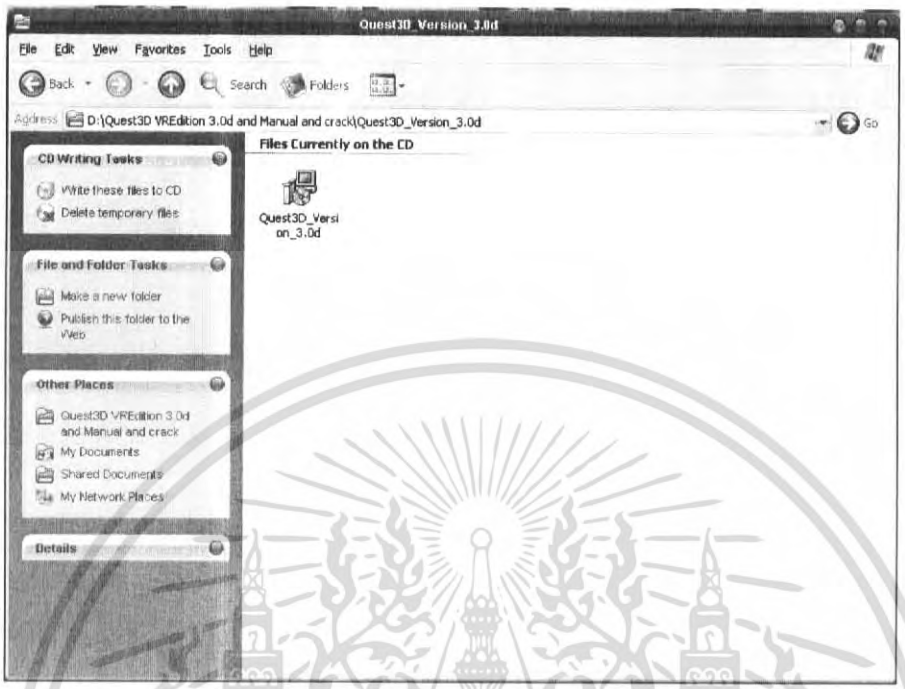


# ภาคผนวก ข.

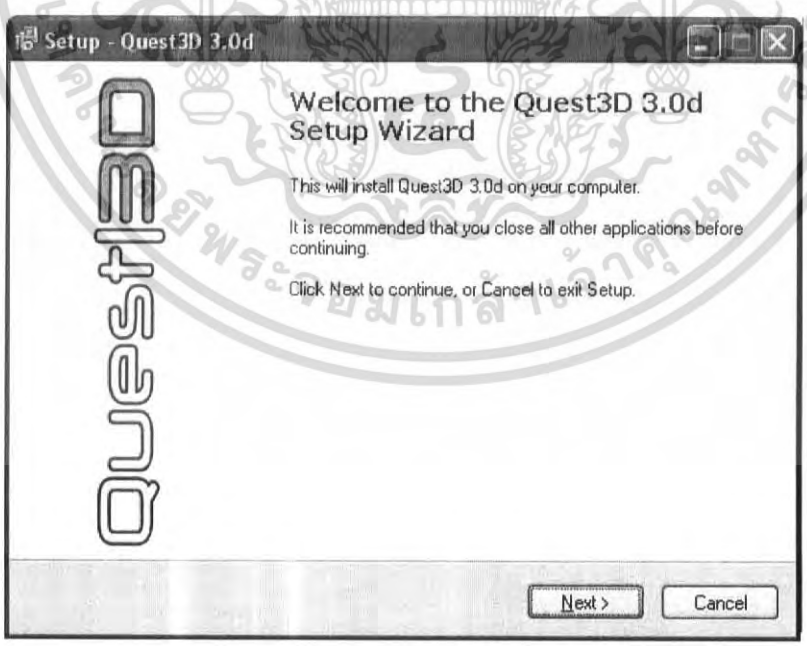
# การติดตั้งโปรแกรม Quest3D

ขั้นตอนการติดตั้ง

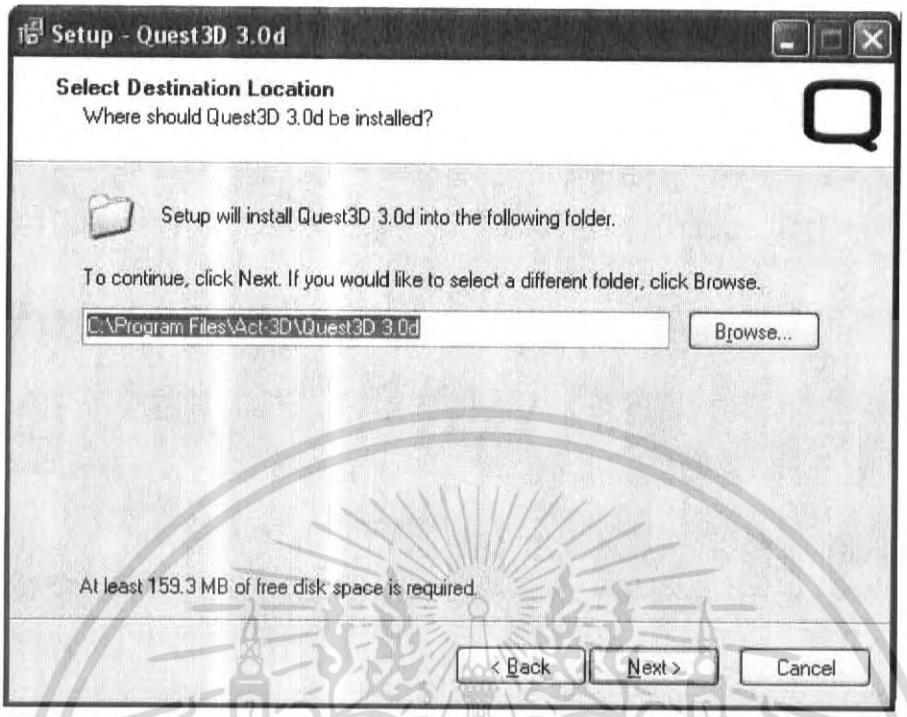
1. ใส่แผ่นโปรแกรม จากนั้นทำการคลิกที่ตัวQuest3D_Version_3.0d ดังในรูป



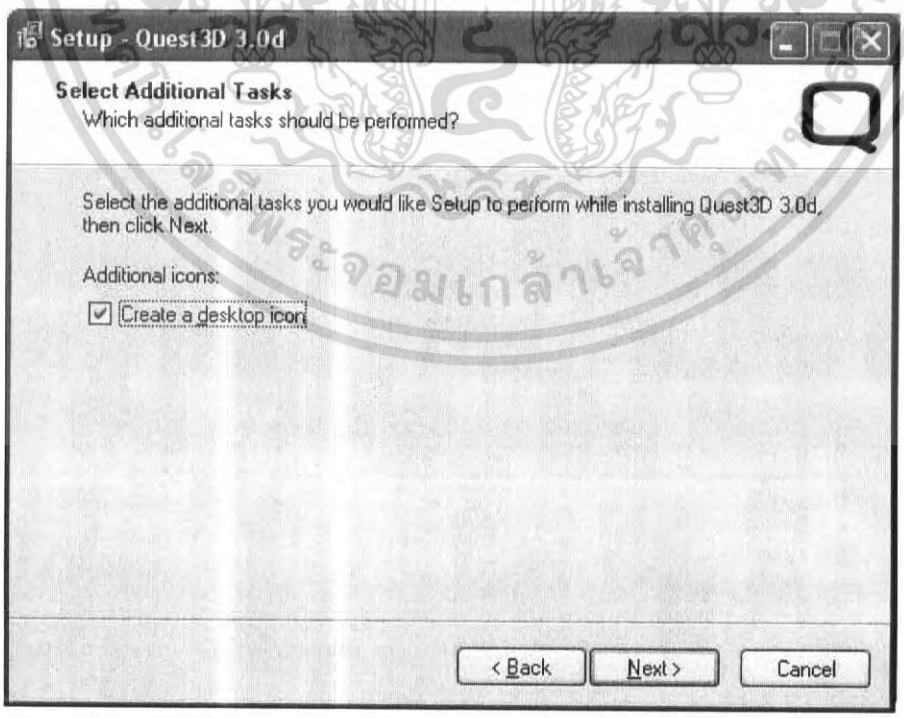

2. ระบบจะแสดงหน้าจอต้อนรับให้คลิก Next



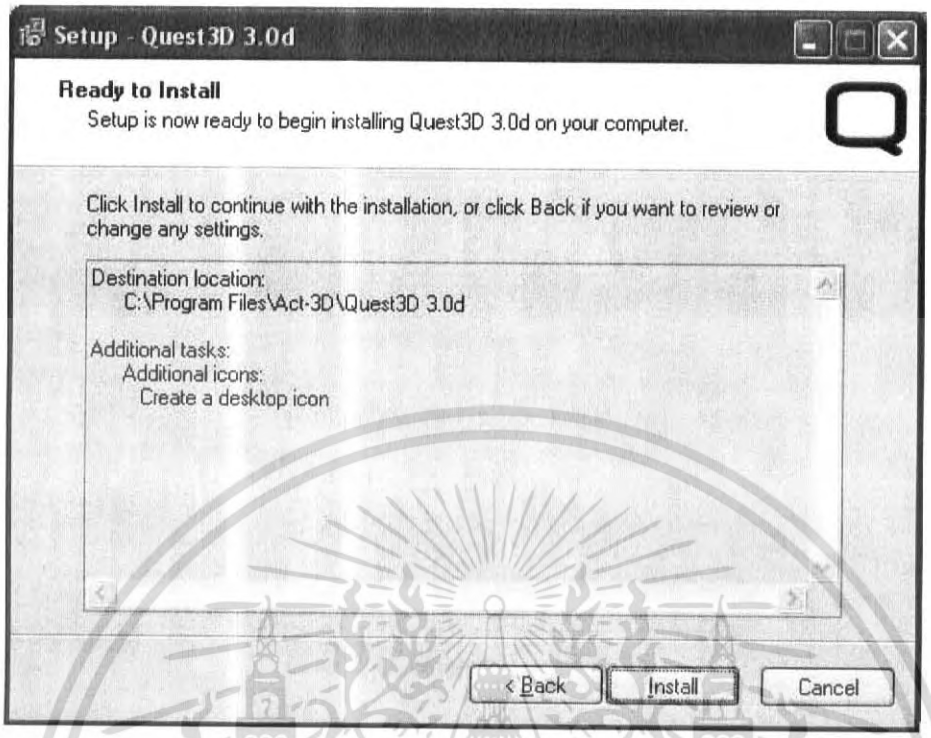
3. เลือก directory ที่จะทำการInstall โดยกดปุ่ม Browse จากนั้น กดปุ่ม Next



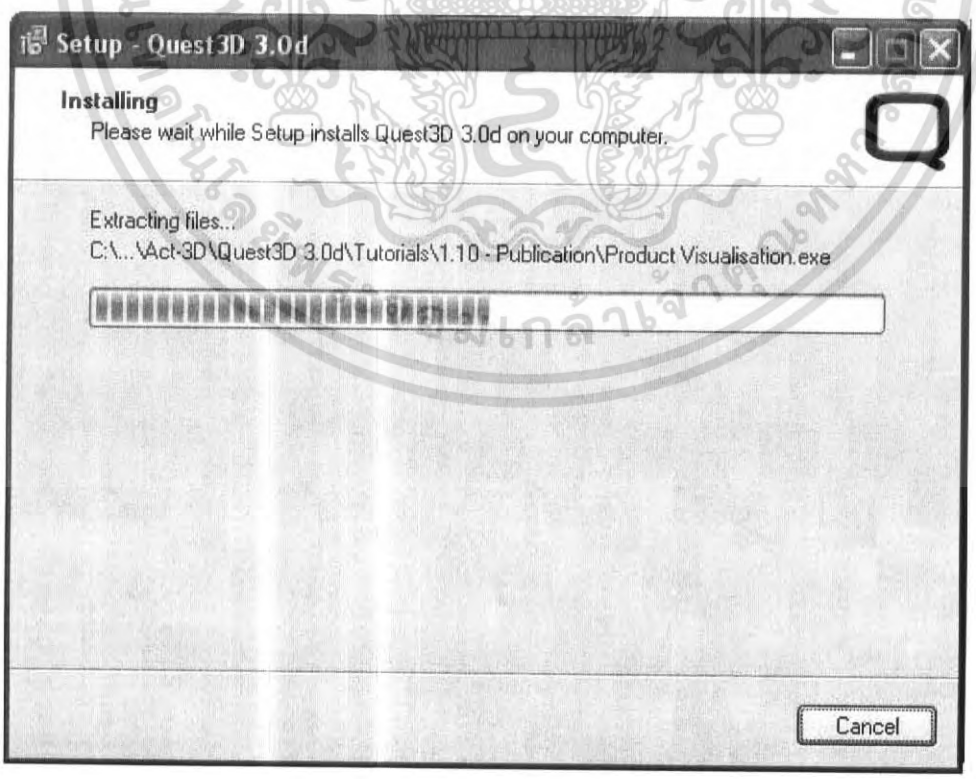

4. คลิกเครื่องหมายที่ช่อง Create a desktop icon เพื่อสร้างiconที่หน้าจอdesktop จากนั้นกดปุ่ม Next



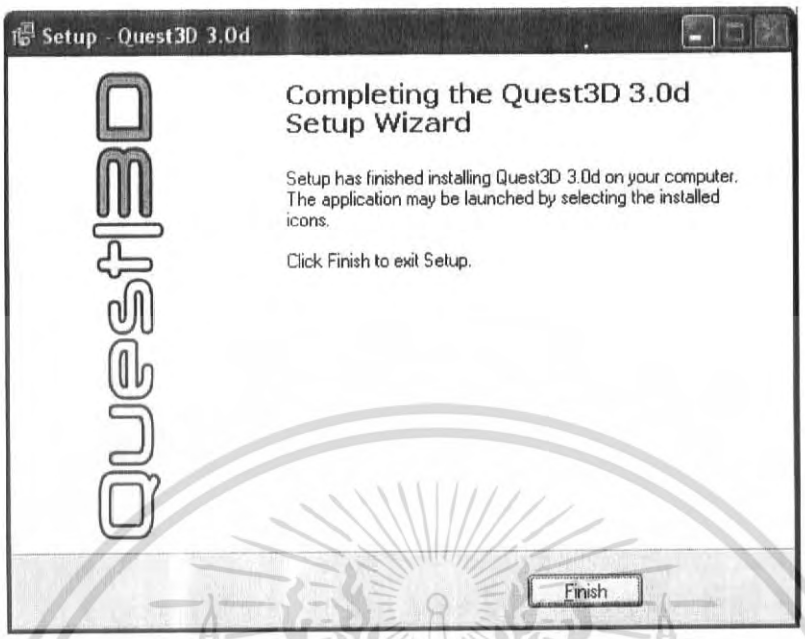
5. แสดงพาธที่จะติดตั้งโปรแกรมQuest3D คลิก Install




6. ขณะที่ติดตั้งโปรแกรมจะแสดงสถานะการติดตั้ง



7. คลิกปุ่ม Finish เมื่อทำการติดตั้งQuest3D เสร็จเรียบร้อย

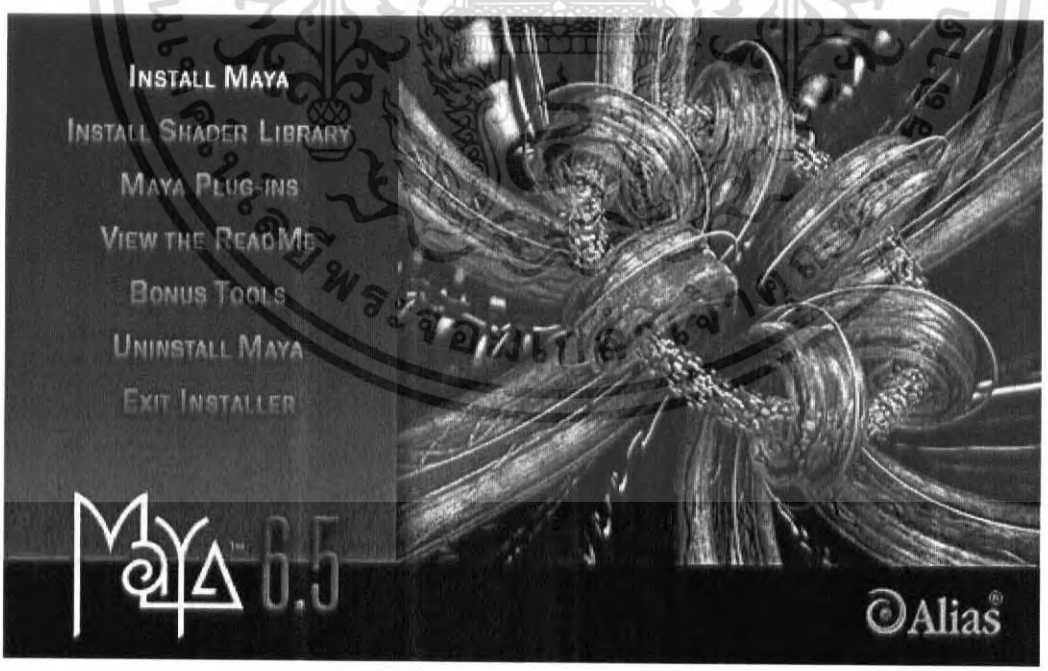


# ภาคผนวก ค.
# การติดตั้งโปรแกรม Maya

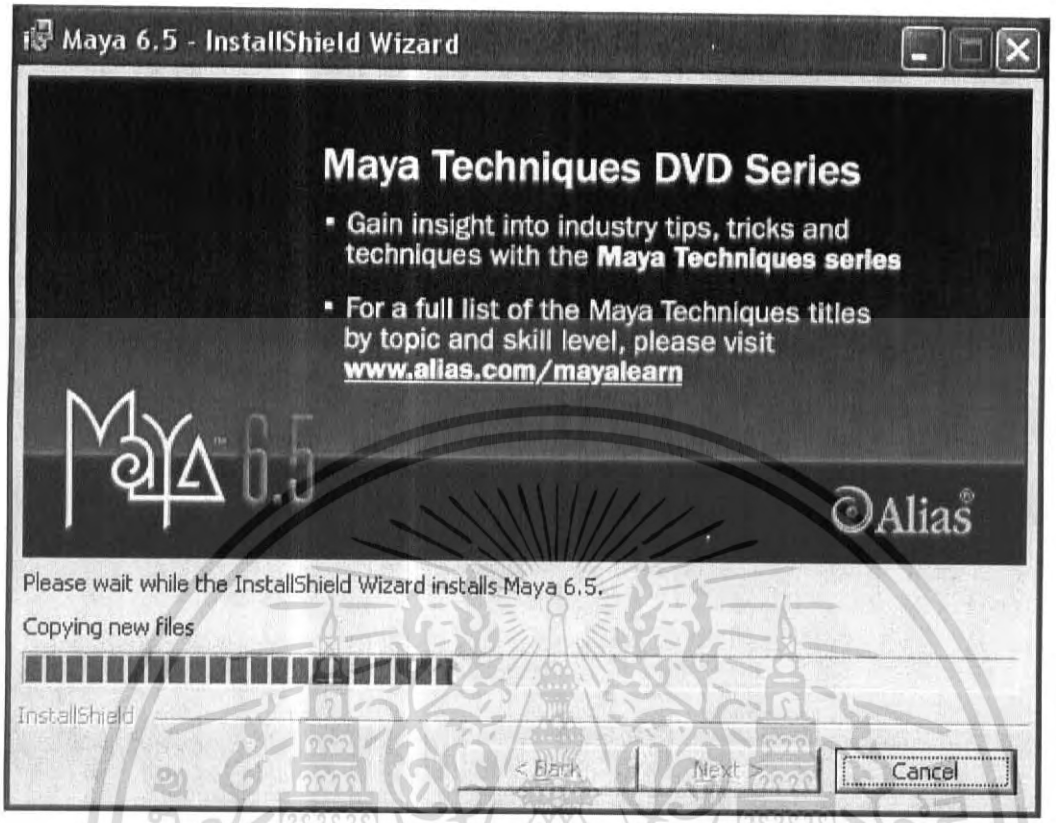
ขั้นตอนการติดตั้ง

1. แสดงหน้าจอการ Install โปรแกรม Maya ขึ้นมา

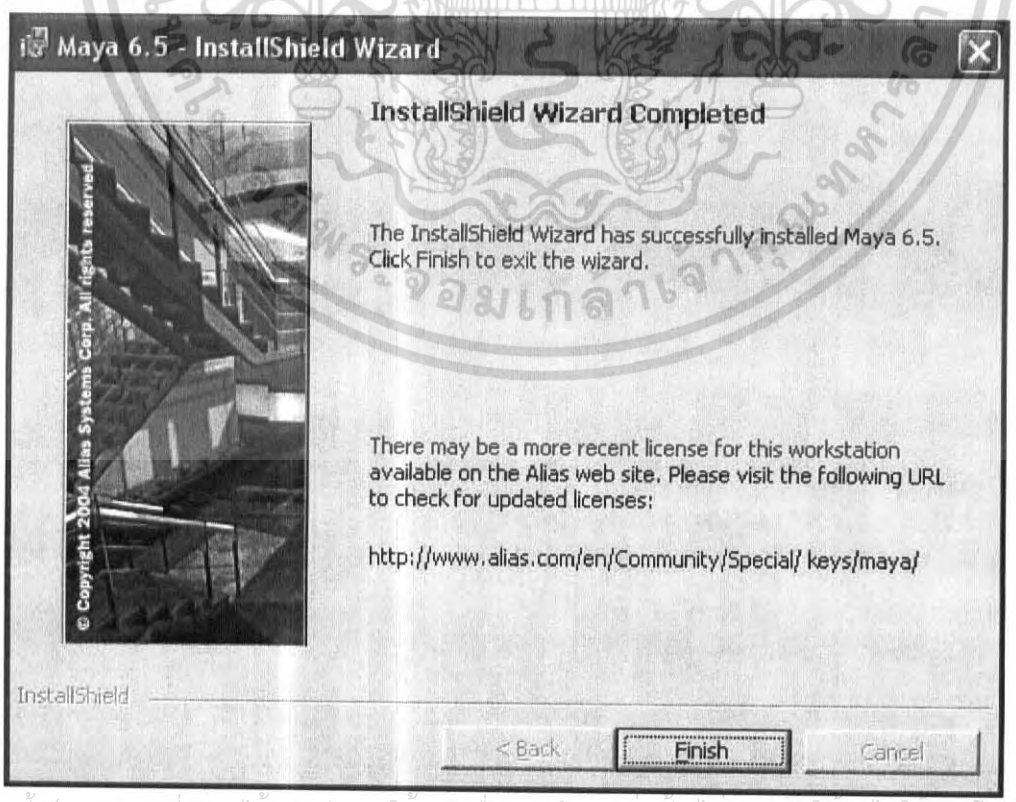

2. ทำการติดตั้งโปรแกรมโดยเลือก Install Maya



3. ขณะที่ติดตั้งโปรแกรมจะแสดงสถานะการติดตั้ง




4. คลิกปุ่ม Finish เมื่อทำการติดตั้งMaya เสร็จเรียบร้อย



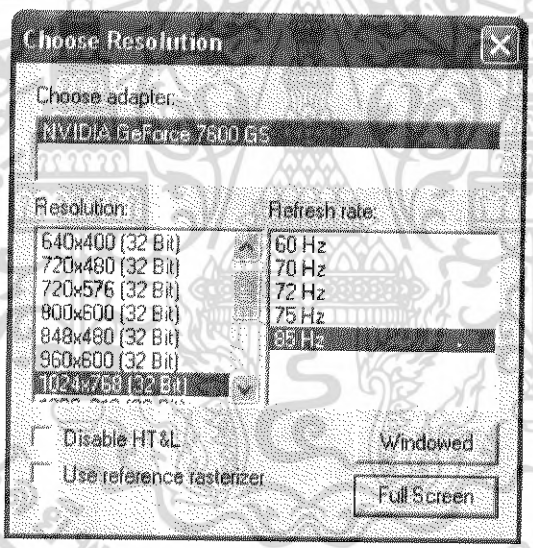
# ภาคผนวก ง.
# คู่มือการเล่น

## ความต้องการของระบบ

1. ระบบปฏิบัติการ Microsoft Windows XP Service Pack 2 ขึ้นไป
2. หน่วยประมวลผลกลาง Intel Pentium4.0 2.0GHz ขึ้นไป
3. หน่วยความจำอย่างน้อย 512MB ขึ้นไป
4. การ์ดประมวลกราฟฟิกของ Nvidia GeForce 7300 GT RAM 256MB
5. โปรแกรม Runtime ของ DirectX ตั้งแต่ version 9.0c ขึ้นไป

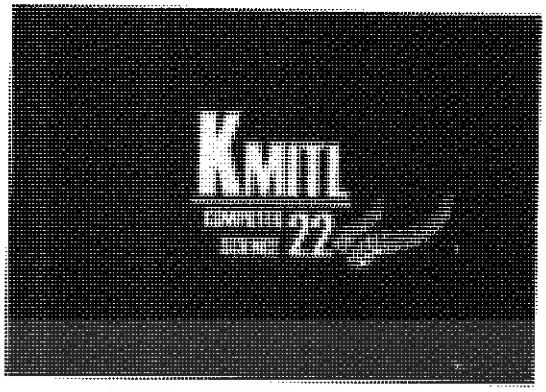
## ขั้นตอนการเล่น

1. ทำการคัดลอกโฟลเดอร์ที่ชื่อ Mebius Combat ไปวางไว้ที่หน้าจอDesktop จากนั้นเข้าไปที่ Mebius Combat.exe จะแสดงรูปแบบการเลือกการแสดงผลของเกม โดยสามารถกำหนดให้แสดงในรูปแบบWindowedที่สามารถเลือกขนาดของวินโดว์ที่ใช้แสดงผลหรือสามารถเลือกแสดงผลแบบFull Screenได้ ดังในรูป ง.1



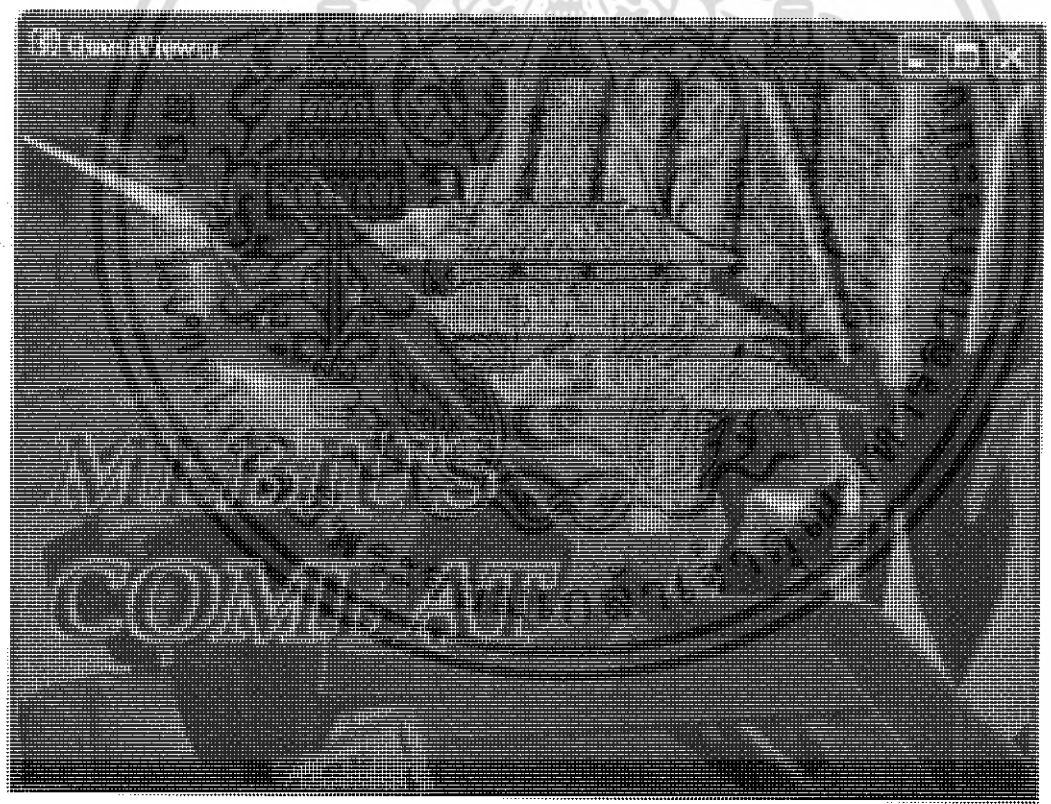

**รูปที่ ง.1** หน้าจอการเลือกแสดงผลรูปแบบของเกม

2. เมื่อเลือกรูปแบบการแสดงผลของเกม จะปรากฏหน้าจอที่แสดงเนื้อเรื่องภาพยนตร์ของเกม




**รูปที่ ง.2** หน้าจอแสดงเนื้อเรื่องภาพยนตร์ของเกม

3. เมื่อเนื้อเรื่องภาพยนตร์แสดงผลจนจบตามเวลาที่กำหนด หรือ มีการกดปุ่มใดๆขณะที่แสดงผลภาพยนตร์ จะปรากฏหน้าจอเมนูหลักของเกม



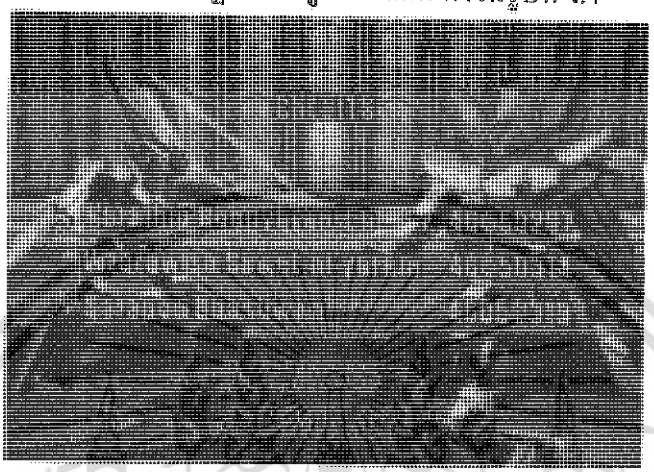
**รูปที่ ง.3** หน้าจอเมนูหลักของเกม

- เลือกเมนู "Start" เพื่อเข้าสู่เกม
- เลือกเมนู "Credits" เพื่อดูรายชื่อผู้พัฒนาเกม

- เลือกเมนู “Exit” เพื่อออกจากเกม

ถ้าไม่ได้มีการกดปุ่มเมนูใดเมนูหนึ่งภายในเวลาที่กำหนด จะกลับไปปรากฏหน้าจอแสดงเนื้อเรื่องภาพยนตร์ของเกม

4. เมื่อเลือกเมนู “Credits” จะปรากฏรายชื่อผู้พัฒนาเกม ดังในรูปที่ ง.4



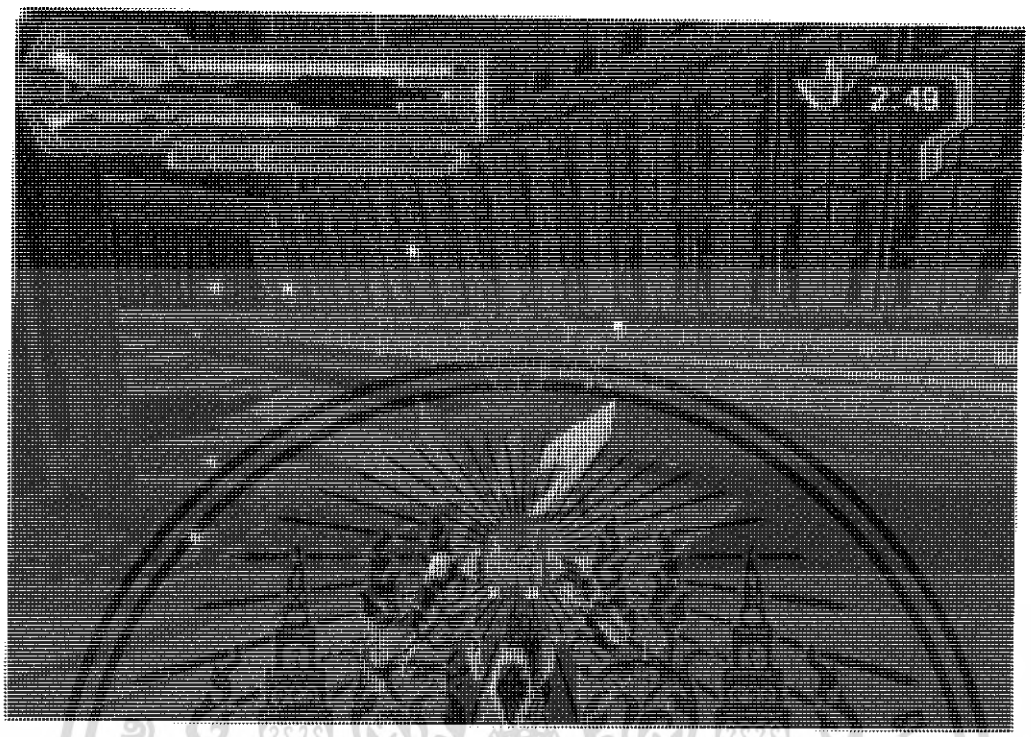
**รูปที่ ง.4** หน้าจอแสดงรายชื่อผู้พัฒนาเกม

- กดปุ่มใดๆ ถ้าต้องการกลับไปยังหน้าเมนูหลักของเกม

5. ภายหลังจากเลือกเมนู “Start” จะปรากฏหน้าจอแสดงฉากภายในเกมก่อนที่จะต่อสู้กับตัวละครดังในรูปที่ ง.5

**รูปที่ ง.5** หน้าจอแสดงฉากภายในเกมก่อนที่จะต่อสู้กับตัวละคร

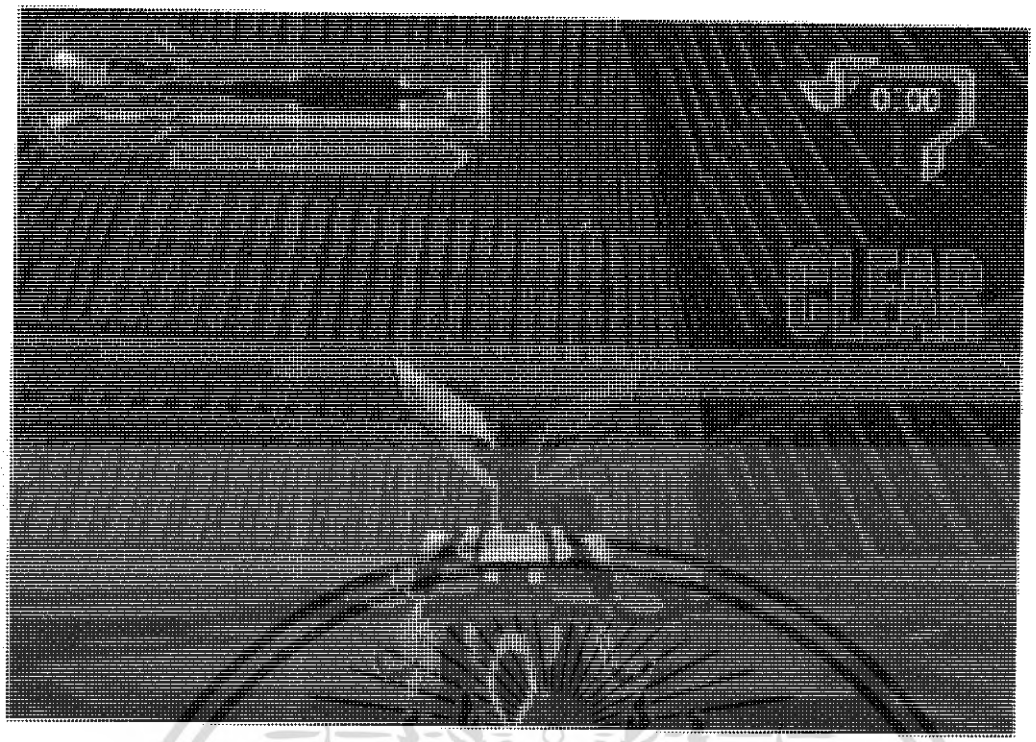
6. ภายหลังจากที่แสดงฉากภายในเกมก่อนที่จะต่อสู้กับตัวละครจนจบ ภายในเวลาที่กำหนด ก็จะเข้าสู่ฉากต่อสู่กับตัวละคร ดังในรูปที่ ง.6



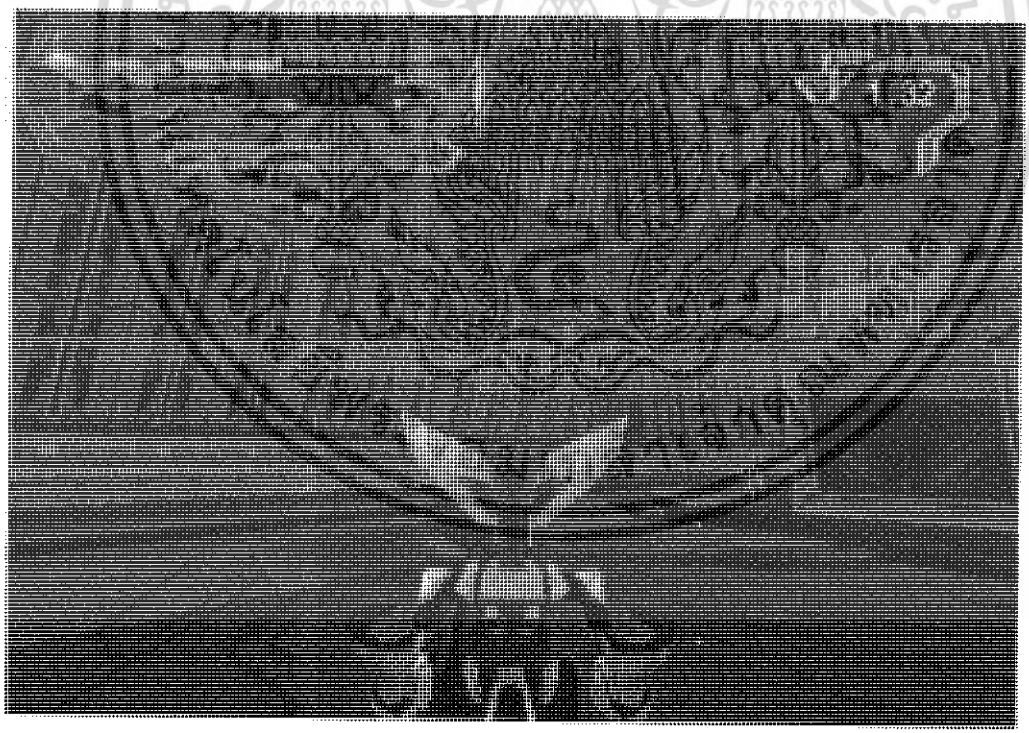
**รูปที่ ง.6** หน้าจอแสดงฉากการต่อสู้ระหว่างผู้เล่นกับตัวศัตรู

- เมื่อพลังชีวิตของศัตรูหมดหรือพลังชีวิตของผู้เล่นมากกว่าพลังชีวิตของศัตรูภายหลังจากหมดเวลา จะแสดงข้อความว่า “Clear” ตรงกลางหน้าจอ ดังในรูปที่ ง.7
- เมื่อพลังชีวิตของผู้เล่นหมดหรือพลังชีวิตของผู้เล่นน้อยกว่าพลังชีวิตของศัตรูภายหลังจากหมดเวลา จะแสดงข้อความว่า “Lose” ตรงกลางหน้าจอ ดังในรูปที่ ง.8

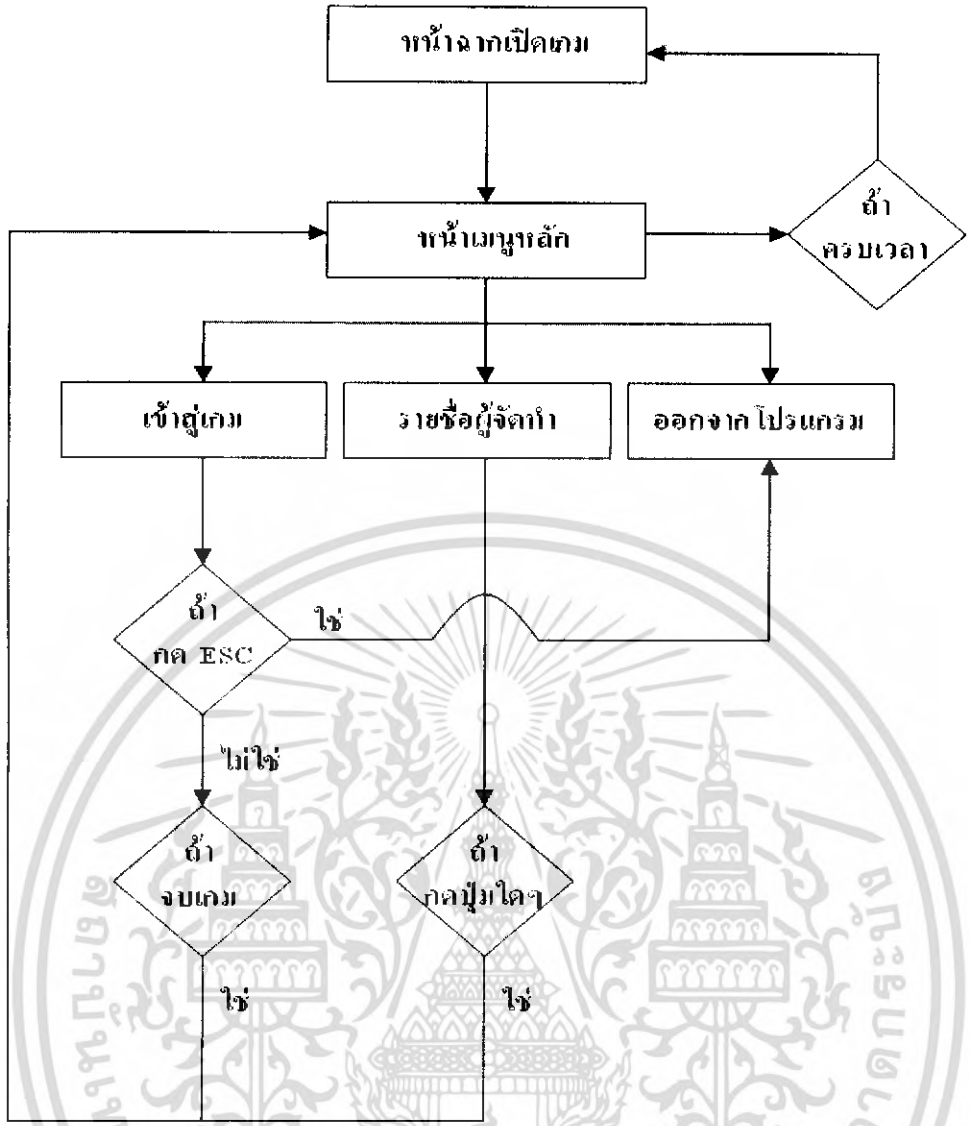
หลังจากแสดงข้อความว่า“Clear”หรือ“Lose” จะกลับไปยังหน้าจอเมนูหลักของเกม



รูปที่ ง.7 แสดงหน้าจอเมื่อผู้เล่นชนะ



รูปที่ ง.8 แสดงหน้าจอเมื่อผู้เล่นแพ้




**รูปที่ ง.9** flowchart diagram แสดงขั้นตอนการทำงานของเกม