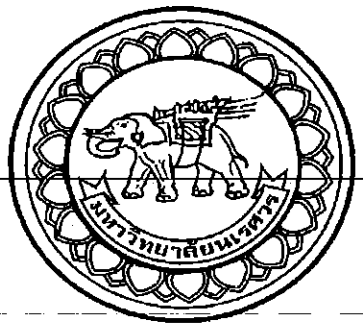


เกมหมากฮอส 3 มิติ ด้วยหลักการปัญญาประดิษฐ์

Artificial Intelligent based 3D Checker Games

นายยงเกียรติ อังประภาพรชัย รหัส 46360087

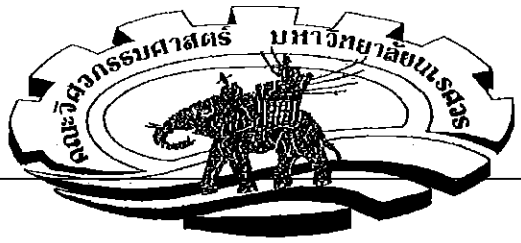
นางสาวสุปรียา เฮือนทา รหัส 46360178

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2549



ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน เกมหมากฮอส 3 มิติ ด้วยหลักการปัญญาประดิษฐ์

ผู้ดำเนินโครงงาน นายยงเกียรติ อังประภาพรชัย รหัส 46360087

นางสาวสุปรียา เสือนทา รหัส 46360178

อาจารย์ที่ปรึกษา ดร. สมยศ เกียรติวนิชวิไล

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2549

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร. สมยศ เกียรติวนิชวิไล)

.....กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ

(อาจารย์จิราพร พุกสุข)

หัวข้อโครงงาน	เกมหมากฮอส 3 มิติ ด้วยหลักการปัญญาประดิษฐ์
ผู้ดำเนินโครงงาน	นายยงเกียรติ อังประภาพรชัย รหัส 46360087
	นางสาวสุปรียา เฮือนทา รหัส 46360178
อาจารย์ที่ปรึกษา	ดร. สมยศ เกียรติวนิชวิไล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2549

บทคัดย่อ

โครงงานนี้เป็นการศึกษาและพัฒนาโปรแกรมเกมหมากฮอส แบบ 3 มิติ ให้มีความฉลาดในการเดินหมาก โดยใช้หลักการของ Game Tree Search, Minimax Algorithm และ Alpha-Beta Algorithm ซึ่งเป็นหลักการทางปัญญาประดิษฐ์ (Artificial Intelligent) ช่วยในการเดินหมากของเกมหมากฮอส การพัฒนาโปรแกรมนี้ทดสอบและรันบนระบบปฏิบัติการไมโครซอฟท์วินโดวส์ XP ใช้ภาษาวิชวลซี พลัสพลัสเวอร์ชั่น 6.0 (Visual C++ 6.0) และ OpenGL ซึ่งเป็น Open source ผลที่ได้จากการทดลองพบว่า เกมหมากฮอส แบบ 3 มิติ มีความฉลาดในการเดินหมากแต่ละตาและสามารถชนะคู่ต่อสู้ได้

โปรแกรมนี้ออกแบบขึ้นมา สามารถเลือกระดับได้ 3 ระดับ และสมารถนำไปแข่งขันกับคนได้จริง

Project title	Artificial Intelligent based 3D Checker Games		
Name	Mr. Yongkiat	Angpraporncha	ID. 46360087
	Miss Supreeya	Huantha	ID. 46360178
Project advisor	Dr. Somyot	Kaitwanidvilai	
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic year	2006		

Abstract

This project has studies and develops 3d Checker game to achieve a higher intelligent game by using Game Tree Search, Minimax Algorithm and Alpha-Beta Algorithm. These algorithms are based on the concept of artificial intelligent which can be used to identify the movement opponent of Checker. Our program is developed and run on Microsoft Windows XP using Visual C++ version 6 and Open GL as the developer tool. The experiment has been shown that 3D Checker game has been furnished with the intelligent algorithms for making a move in each turn and efficiently conquers the game.

The developed game has 3 level and is applicable for general competition with human

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์สำเร็จได้ด้วยดี ก็เนื่องจากความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษา อาจารย์สมยศ เกียรติวนิชวิไล ที่กรุณาให้คำปรึกษา แนะนำวิธีการในการทำงาน ตลอดถึงการตรวจสอบการทำงานพร้อมทั้งเสนอแนวทางการแก้ไขตลอดระยะเวลาการทำโครงงาน สุดท้ายต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ ทุกคนที่ยังไม่ได้เอ่ยนามที่คอยสนับสนุนในการทำโครงงานครั้งนี้

สารบัญ

หน้า

สารบัญ

สารบัญรูป

สารบัญรูป

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันคอมพิวเตอร์เป็นเครื่องมือหรือเป็นปัจจัยสำคัญอย่างยิ่งในการพัฒนาผลงานจากความคิดสร้างสรรค์ ไม่ว่าจะเป็นผลงานศิลปะ วีดิทัศน์ สื่อเว็บไซต์ ภาพยนตร์ หรือแม้กระทั้งเกมต่าง ๆ ซึ่งเรามีให้เราสามารถเลือกเล่นได้มากมาย

เกมแต่ละเกมก็มีจุดที่น่าสนใจแตกต่างกันออกไป เช่น วิธีการเล่น การแสดงออกของตัวละคร ภาพพื้นหลัง เป็นต้น ดังนั้นการสร้างเกมให้มีจุดเด่นและเป็นที่น่าสนใจของผู้เล่น จึงทำได้หลายวิธี ในโครงงานนี้จะพัฒนาเกมหมากฮอส ให้มีความน่าสนใจและมีความฉลาดมากขึ้น โดยให้หลักการของคอมพิวเตอร์กราฟฟิกและปัญญาประดิษฐ์

เกมหมากฮอสในโครงงานนี้จะถูกทำในรูปแบบเป็นภาพ 3 มิติ ทำให้เป็นที่ดึงดูดความสนใจและผู้เล่นมีความอยากเล่นมากยิ่งขึ้น เสมือนกับได้เข้าไปเล่นเกมนั้นจริง ๆ อีกหลักการที่ใช้ในโครงงานนี้คือ หลักการปัญญาประดิษฐ์ (Artificial Intelligence) คือทำให้เกมมีความฉลาดขึ้นเมื่อมีการแข่งขันกับผู้เล่น หรือมนุษย์ที่มีความชำนาญในเกมนั้น ๆ ด้วยหลักการนี้จะทำให้คอมพิวเตอร์มีความคิดและค้นหาลำดับการเดิน เพื่อที่จะเอาชนะคู่ต่อสู้ได้

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อสร้างเกมหมากฮอส 3 มิติ ที่มีภาพ 3 มิติ สวยงามและมีความฉลาดตามหลักการของปัญญาประดิษฐ์
2. เพื่อสร้างเกมที่ปรับแผนการเดินจากประสบการณ์ของการแข่งขันได้
3. เพื่อประยุกต์ใช้โปรแกรมโดยภาษาซี (C) และ OpenGL ในการสร้างเกม

1.3 ขอบข่ายของโครงงาน

1. สร้างเกมหมากฮอส โดยใช้หลักการปัญญาประดิษฐ์ (Artificial Intelligence) ให้มีการเดินที่ฉลาด สามารถเอาชนะคู่ต่อสู้ได้ดีขึ้น
2. สร้างเกมหมากฮอส 3 มิติ

1.4 ขั้นตอนของการดำเนินงาน

1. ศึกษาและค้นคว้าข้อมูลเกี่ยวกับคอมพิวเตอร์กราฟฟิก (Computer Graphics)
2. ออกแบบรูปร่างของตัวหมากฮอส และตารางการเดินหมากฮอส
3. เขียนโปรแกรมเพื่อจำลองการเคลื่อนไหวของตัวหมากฮอส
4. เขียนโปรแกรมในส่วนของตัวหมากฮอสด้วยหลักการปัญญาประดิษฐ์ (Artificial Intelligence)
5. ทดสอบการทำงาน
6. สรุปการทดลองและจัดทำรูปเล่ม

1.5 แผนการดำเนินงาน

กิจกรรม	ปี 2548		ปี 2549									
	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
1. ศึกษาและค้นคว้าข้อมูลเกี่ยวกับ Computer Graphics และ OpenGL	←	→										
2. ออกแบบรูปร่างของตัวหมากฮอส และตารางการเดินหมากให้เป็น 3 มิติ		←	→									
3. เขียนโปรแกรมเพื่อจำลองการเคลื่อนไหวของตัวหมากฮอส			←	→								
4. เขียนโปรแกรมในส่วนของตัวหมากฮอส ด้วยหลักการปัญญาประดิษฐ์				←	→							
5. ทดสอบการทำงาน					←	→						

6. สรุปและจัดทำรูปเล่มโครงงาน								←	—	—	—	→

1.6 ผลที่คาดว่าจะได้รับ

1. ได้เกมภาพ 3 มิติ
2. เกมที่มีหลักการปัญญาประดิษฐ์ (Artificial Intelligence) ในส่วนควบคุมตัวหมากฮอสได้

1.7 งบประมาณของโครงงาน

1.	ค่าวัสดุสำนักงาน	เป็นเงิน	500	บาท
2.	ค่าวัสดุคอมพิวเตอร์	เป็นเงิน	500	บาท
3.	ค่าถ่ายเอกสาร	เป็นเงิน	800	บาท
4.	ค่าวัสดุอื่น ๆ	เป็นเงิน	200	บาท
		รวมเป็นเงินทั้งสิ้น	2,000	บาท (สองพันบาทถ้วน)

บทที่ 2
หลักการและทฤษฎี

หลักการและทฤษฎีในการที่จะสร้างเกมหมากฮอส 3 มิตินั้น เราจำเป็นที่จะต้องรู้ถึงหลักการของคอมพิวเตอร์กราฟฟิก ว่าทำงานอย่างไรจึงสามารถแสดงภาพออกมาให้เป็นภาพ 3 มิติ และการใช้งานของ OpenGL ที่ในการสร้างภาพ รวมไปถึงวิธีการคิดเพื่อที่จะเอาชนคู่ต่อสู้ โดยใช้หลักการทางปัญญาประดิษฐ์ (artificial intelligent)

2.1 คอมพิวเตอร์กราฟฟิก (Computer Graphics)

กราฟฟิก คือ ศิลปะอย่างหนึ่ง ที่แสดงออกด้วยความคิดอ่าน โดยใช้เส้น รูปภาพ ภาพเขียน ไดอะแกรม และอื่นๆ การสื่อความหมายด้วยการใช้ภาพวาด ภาพสเกต แผนภาพ ภาพถ่าย และอื่นๆ ที่ต้องอาศัยศิลปะ และศาสตร์ เข้ามาช่วย เพื่อให้ผู้ดูเกิดความคิด และตีความหมายได้ตรงตามที่ผู้สร้างสรรค์ต้องการสื่อ เช่น แผนภูมิ แผนภาพโฆษณา การ์ตูน เป็นต้น โสตทัศนวัตถุที่ผลิตขึ้นเพื่อแสดงสัญลักษณ์ หรือความหมายของสิ่งหนึ่งสิ่งใด ทำให้คนได้มองเห็นความจริง หรือความคิดอันถูกต้องชัดเจนจากวัสดุกราฟฟิกนั้นๆ

งานกราฟฟิกไม่ได้จำกัดอยู่ที่การวาดภาพ หรือการสร้างสรรค์ศิลปะด้วยการวาดเท่านั้น ในปัจจุบันงานกราฟฟิก มีขอบเขตกว้างมาก เช่นการวาดภาพ เขียนภาพ การออกแบบสัญลักษณ์ ป้ายโฆษณา งานโทรทัศน์ การสร้างเกม เป็นต้น

คอมพิวเตอร์กราฟฟิก (computer graphics) เป็นการสร้างและจัดการภาพกราฟฟิกโดยคอมพิวเตอร์ ได้มีการนำเอาความรู้ทางด้านคอมพิวเตอร์กราฟฟิกมาประยุกต์ในงานในหลายด้าน อาทิ เช่น

- Computer aided design (CAD) เป็นงานที่เกี่ยกับการออกแบบ เช่น แบบอาคาร ยานพาหนะ ชิ้นส่วนอุปกรณ์เครื่องมือต่าง ๆ เป็นต้น
- Presentation graphics เป็นงานที่เกี่ยวกับการนำ เสนอข้อมูลในรูปของแผนภูมิ หรือแผนภาพมักใช้ในการสรุปข้อมูลเพื่อประกอบการรายงาน
- Computer art การใช้คอมพิวเตอร์ในการสร้างภาพศิลป์ ทั้งภาพนิ่งและภาพเคลื่อนไหว
- Entertainment การนำเทคนิคคอมพิวเตอร์กราฟฟิกมาใช้ในภาพยนต์หรือละคร รวมทั้งสื่อโฆษณา และเกมส์ต่าง ๆ
- Education and training เป็นการนำเอาคอมพิวเตอร์กราฟฟิกมาใช้เป็นตัวผลิตสื่อการเรียนการสอน เช่น CAI หรือ Simulation หรือ Visual Reality

- Image Processing เป็นการทำงานที่ตรงข้ามกับคอมพิวเตอร์กราฟฟิก กล่าวคือ คอมพิวเตอร์กราฟฟิกเป็นลักษณะการสร้างภาพ ส่วน Image processing เป็นลักษณะการวิเคราะห์ข้อมูลจากภาพ หรือปรับปรุงภาพให้ดีขึ้น

คอมพิวเตอร์กราฟฟิก (Computer Graphics) แบบ 3 มิติ ประกอบด้วยการเปลี่ยนแปลงรูปภาพการเปลี่ยนภาพ 2 มิติที่มีเส้นขอบและการแรเงาให้แทนด้วยพื้นผิว ด้วยคำสั่งภาษาโปรแกรมและส่วนอื่น ๆ ที่เกี่ยวข้องกับการออกแบบ

2.2 OpenGL

OpenGL คือ software ที่เชื่อมต่อกับ hardware ที่แสดงผลทางด้านกราฟฟิก มีโครงสร้างเป็น hardware-independent interface และสามารถใช้ได้กับ OS หลายๆ แบบ ซึ่งจะมีคำสั่งวาดภาพพื้นฐาน คือ จุด เส้น และรูปเหลี่ยมต่างๆ และการแสดงภาพ Raster

ภาษาที่สามารถใช้กับ OpenGL มีดังนี้ C/C++ (VC++, Borland C++, C++ Builder, C compiler on UNIX) , Delphi , Visual Basic , Java , Perl , Python , Fortran และ Ada

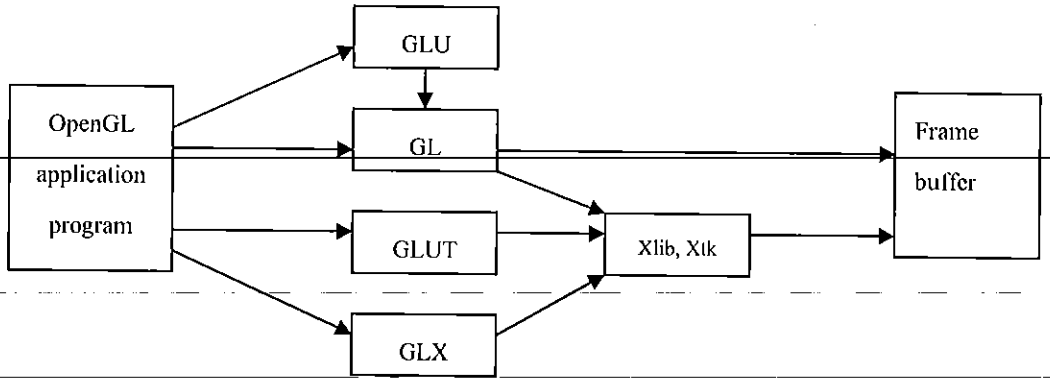
เหตุผลที่นิยมใช้ OpenGLในระบบกราฟฟิกมีประสิทธิภาพสูงในการเร่งความเร็ว application 3 มิติ และเกมต่างๆ ในปัจจุบันทำให้สามารถใช้ข้อมูลจำนวนมากและสร้าง effect 3 มิติในแบบ real-time การเพิ่มการสนับสนุนอุปกรณ์ใหม่ๆ ลงไปใน OpenGL ทำได้ง่ายและรวดเร็ว สามารถทำงานได้บนหลายแพลตฟอร์ม (platform) ทำให้การย้ายโปรแกรมประยุกต์ระหว่างแต่ละแพลตฟอร์มนั้นทำได้ง่ายและประหยัด มีเสถียรภาพในการทำงานสูงสามารถใช้งานกับเครื่อง High End 3D Workstation และ Supercomputer ได้

2.2.1 The OpenGL Interface

ฟังก์ชัน OpenGL จะขึ้นต้นด้วย gl และถูกเก็บไว้ที่ Library ที่เรียกว่า GL ซึ่งจะมี Library อื่นที่เกี่ยวข้องอีกไม่มาก

- Graphics utility library (GLU) เป็น library ที่ใช้เฉพาะ GL ฟังก์ชัน แต่ประกอบไปด้วยฟังก์ชันสำหรับการสร้างวัตถุพื้นฐาน เช่น ฟังก์ชันในการสร้างทรงกลม ผู้ใช้ไม่จำเป็นต้องรู้ว่าการสร้างทรงกลมสร้างอย่างไรแต่สามารถเรียกใช้ฟังก์ชันได้เลย GLU เป็น library ที่มีในทุก OpenGL implementation

- OpenGL Utility Toolkit (GLUT) คือ library ระบบกราฟฟิก เช่น การติดต่อกับการแสดงผลทางหน้าจอต่างๆ ช่วยให้เข้าใจการทำงานของระบบกราฟฟิกมากขึ้น



รูปที่ 2.1 แสดง library organization [1]

2.3 ลักษณะของ OpenGL

2.3.1 The OpenGL API

OpenGL API (Application Programing Interface) คือ โปรแกรมที่ทำหน้าที่เชื่อมต่อระหว่างโปรแกรม Appication กับ Hardware โดยที่โปรแกรมเมอร์ที่เขียน Application ต่างๆ ไม่จำเป็นต้องเขียนให้รู้จักไปถึง Hardware แค่เขียนให้รู้จัก API คล้ายๆ เป็นล่ามระหว่าง Software กับ Hareware

โครงสร้างของ OpenGL มีลักษณะคล้ายกับ API อื่นๆ รวมทั้ง PHIG (Programmer's Hierarchical Interactive Graphic System) และ GKS (Graphic Kernal System)ความรู้ทางด้าน OpenGL สามารถนำไปใช้กับ software อื่นๆ ได้เช่นกัน แม้ว่า OpenGL จะง่ายในการเรียนรู้เมื่อเปรียบเทียบกับ API อื่นๆ แต่มีความสามารถสูงมาก OpenGL จะสนับสนุนทั้งโปรแกรม 2 และ 3 มิติ รวมทั้งสนับสนุนเทคนิคการแสดงภาพระดับสูงด้วย

2.3.2 Graphic Functions

การทำงานของระบบกราฟิก ผู้ใช้ไม่สามารถรู้ว่ามีการทำงานอย่างไร รู้แต่เพียงว่า input และ output คืออะไร ในระบบกราฟฟิกจะมีฟังก์ชันที่ถูกเรียกใช้ มีการรับค่าของ input จากอุปกรณ์ต่างๆ เช่น mouse หรือ keyboard หรือจาก input อื่นๆ เช่น message จาก operation system จะได้ผลลัพธ์จากการทำงานและแสดงผลยังอุปกรณ์ในการแสดงผล output อาจจะมอง input คือฟังก์ชันที่ผู้ใช้เรียกใช้ และ output คือ ผลของการทำงานที่แสดงยังหน้าจอ CRT ดังรูป 2.2 และแม้เราจะใช้ OpenGL ในการทำงานแต่ระบบกราฟิก API อื่นๆ ก็สามารถนำมาใช้งานร่วมกันได้

รูปที่ 2.2 แสดงระบบกราฟิกที่เปรียบเสมือนกล่องดำ (Black Box) [1]

API ประกอบด้วยฟังก์ชันจำนวนมาก สามารถแบ่งฟังก์ชันออกเป็น 6 กลุ่ม ดังนี้

1. Primitive functions จะทำการนิยาม low – level object หรือ entities ขนาดเล็กมากที่ระบบสามารถแสดง primitive รวมทั้ง points, line segment, polygons, pixels, test, curves และ surfaces ขึ้นอยู่กับว่าเป็น API ประเภทใด

2. Attribute function จะกำหนดวิธีในการแสดงของ primitive บนจอ attribute function จะอนุญาตให้เลือกสีในการแสดง line segment เลือกรูปแบบที่นำไปเติมใน polygon หรือเลือกชนิดของพื้นผิวที่จะระบายในกราฟ

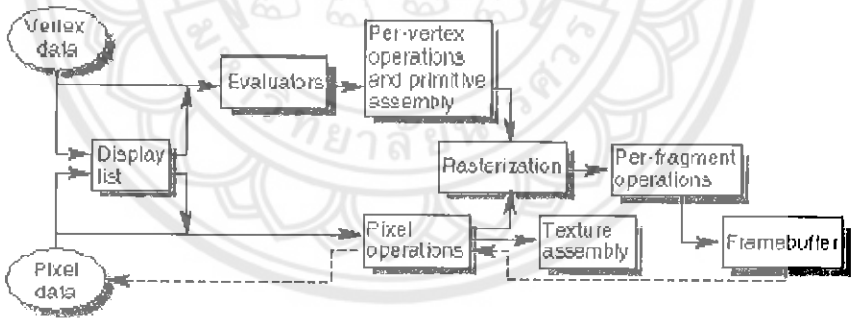
3. Viewing function จะอนุญาตให้ผู้ใช้สามารถเลือกตำแหน่งในการมองภาพได้

4. Transformation function จะอนุญาตให้สามารถทำการแปลงภาพได้ เช่น การย้ายภาพ การย้ายตำแหน่ง หรือการย่อขยาย

5. Input function อนุญาตให้ทำการ input ข้อมูลได้หลากหลายมากขึ้น ซึ่งต้องการฟังก์ชันที่ทำการจัดการกับอุปกรณ์ต่างๆ เช่น Keyboard, mouse และ data tablets

6. Control Function ทำให้สามารถทำการติดต่อกับระบบ window เมื่อเริ่มมีการทำงานของระบบและจัดการกับ error ต่างๆ ที่เกิดขึ้นระหว่างการทำงานของโปรแกรม

2.4 การทำงานของ OpenGL



รูปที่ 2.3 การทำงานของ OpenGL [1]

2.4.1 Display List คำสั่งต่างๆ ที่เรียกใช้ ส่วนใหญ่จะสั่งต่อไปให้บล็อกถัดไปแบบไปป์ไลน์ แต่เราสามารถเก็บกลุ่มคำสั่งเหล่านี้ไว้ใน Display List เพื่อนำมาประมวลผลทีหลังได้

2.4.2 ตัวประเมินผล (Evaluator) จะเป็นตัวประเมินคำสั่งที่ป้อนเข้ามา การประมาณค่าของเส้นโค้ง และพื้นผิวทางเรขาคณิตด้วยฟังก์ชันโพลิโนเมียล เพื่อสร้างเป็นเวอร์เท็กซ์ นอร์มอลล์ ตำแหน่งของเท็กซ์เจอร์ และสี

2.4.3 Per-Vertex Operations and Primitive assembly จะทำการดำเนินการกับแต่ละจุด ทำรูปทรง Primitive เรขาคณิต OpenGL จะประมวลผลหน่วยพื้นฐานทางเรขาคณิตที่ประกอบด้วยจุด เส้น และโพลีกอน ให้อยู่ในรูปของเวอร์เท็กซ์ และเวอร์เท็กซ์เหล่านี้จะถูกแปลง แล้วกำหนดจุดบนจอแสดงภาพ นอกจากนี้ในการดำเนินการขั้นพื้นฐานจะมีการตัดแต่ง (Clip) เวอร์เท็กซ์ให้เหมาะสมกับวินโดว์ที่ใช้แสดงภาพ

2.4.4 Rasterization ในขั้นตอนนี้จะเป็นการสร้างที่เก็บจุดภาพ โดยสร้างเป็นลำดับของเฟรมบัฟเฟอร์ ซึ่งจะอยู่ในรูป 2 มิติ ไม่ว่าค่านั้นจะเป็น จุด เส้น หรือรูปหลายเหลี่ยมก็ตาม เมื่อสร้างที่เก็บหรือจองหน่วยความจำให้กับข้อมูลเรียบร้อยแล้ว เฟรมบัฟเฟอร์ที่สร้างขึ้นจะถูกแบ่งออกเป็นส่วนๆ เพื่อส่งต่อให้กับบล็อกถัดไป

2.4.5 Per-Fragment operations ในขั้นตอนนี้จะเป็นการดำเนินการขั้นสุดท้ายกับข้อมูลก่อนที่จะเก็บสิ่งที่ประมวลผลได้ในรูปแบบของจุดภาพ ซึ่งจะนำไปเก็บไว้ในบัฟเฟอร์ โดยในขั้นตอนนี้จะรวมเอาเงื่อนไขในการปรับปรุงการแสดงผลไว้ด้วย โดยอาศัยข้อมูลที่ป้อนเข้ามาใหม่ และข้อมูลที่มีอยู่เดิมในเฟรมบัฟเฟอร์ โดยจะเก็บไว้ในค่า z และทำการผสมผสานสีของจุดภาพที่ป้อนเข้ามาใหม่กับสีที่เก็บอยู่เดิม

2.5 การแปลงในระบบ 3 มิติ (Three-Dimensional Transformations)

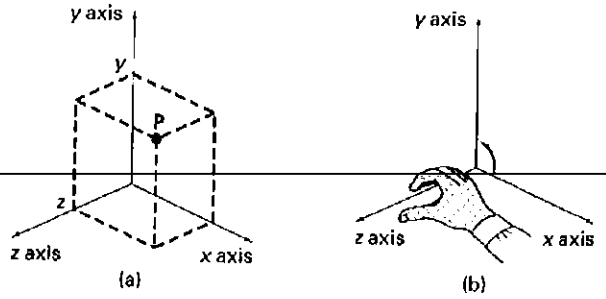
การแปลงในระบบ 3 มิติ มีหลักการเหมือนกับการแปลงในระบบ 2 มิติ แต่จะมีการคิดแนวแกน Z เพิ่มขึ้นมา ดังนั้นก่อนที่จะเริ่มต้นเกี่ยวกับการแปลง จะขอแนะนำ เกี่ยวกับเรื่องระบบพิกัด 3 มิติ (3D – Coordinate system) และเรื่องเวคเตอร์ (vector) ซึ่งเป็นพื้นฐานในการทำงานกับภาพในระบบ 3 มิติ

2.5.1 ระบบพิกัด 3 มิติ (3d-Coordinate System)

ในระบบ 2 มิติ มีแกนเพียง 2 แกนเท่านั้น คือแกน X และแกน Y ดังนั้นการกำหนดจุดต่างๆ จะอ้างถึงพิกัดเพียงแค่ 2 แกนดังกล่าว แต่ในระบบ 3 มิติ จะมีความลึกเข้ามาเกี่ยวข้อง นั่นคือจะต้องพิจารณาแกนเพิ่มอีก 1 แกน คือ แกน Z

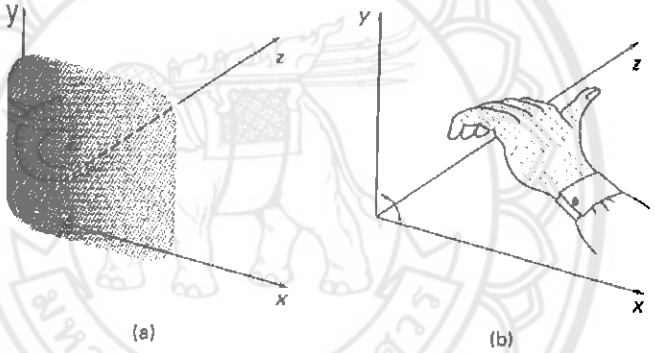
การกำหนดทิศทางของแกน Z มี 2 แบบ คือ แบบระบบมือขวา และแบบระบบมือซ้าย

ระบบมือขวา : แกน Z จะพุ่งออกตามทิศทางของหัวแม่มือ ระบบนี้มักใช้ในงานทางคณิตศาสตร์ หรือการใช้งานในระบบภูมิศาสตร์



รูปที่ 2.4 แสดงระบบมือขวา [2]

ระบบมือซ้าย : แกน Z จะพุ่งเข้าในทิศทางตรงข้ามกับระบบมือขวา เป็นระบบที่ใช้ในคอมพิวเตอร์กราฟฟิกเนื่องจาก ระนาบจอภาพที่อ้างถึงคือ ระนาบ XY ส่วนระยะความลึกจากจอภาพเข้าไป คือแกน Z มีค่าเป็นบวก



รูปที่ 2.5 แสดงระบบมือซ้าย [2]

การกำหนดจุดในระบบ 3 มิติ ต้องใช้การอ้างถึงทั้ง 3 แกน เช่น จุด (1, 2, 3) จะเป็นจุดที่ห่างจากจุดกำเนิดไปตามแนวแกน X 1 หน่วย แกน Y 2 หน่วย และแกน Z 3 หน่วย

2.5.2 การย้าย (Translation)

การย้ายในระบบ 3 มิติ สามารถเขียนให้อยู่ในรูปของ homogeneous coordinate matrix ได้ดังนี้

$$\begin{bmatrix} x' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & Tx \\ 0 & 1 & 0 & Ty \\ 0 & 0 & 1 & Tz \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

รูปที่ 2.6 แสดงมตริกซ์การย้าย

เมื่อ (X, Y, Z) คือ โคออร์ดิเนตของวัตถุก่อนการย้าย

(X', Y', Z') คือ โคออร์ดิเนตของวัตถุหลังการย้าย

Tx, Ty, Tz เป็นระยะห่างในการย้ายตามแนวแกน X, Y และ Z ตามลำดับ

เทียบเท่ากับสมการการย้าย ดังนี้

$$X' = X + tx$$

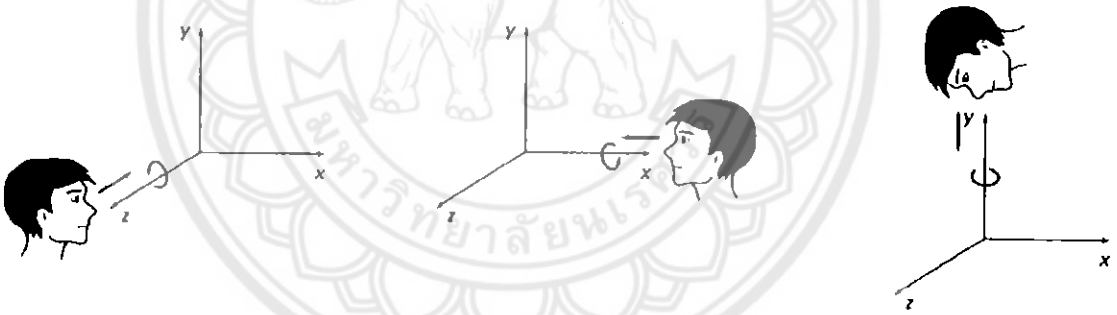
$$Y' = Y + ty$$

$$Z' = Z + tz$$

หลักการการย้ายในระบบ 3 มิติ ทำได้โดยการย้ายจุดแต่ละจุดของวัตถุนั้นๆ ไปยังจุดที่ต้องการ จากนั้นจึงทำ การลากเส้นเชื่อมจุดเหล่านั้น เพื่อสร้างรูปวัตถุนั้นในตำแหน่งใหม่ที่ต้องการ

2.5.3 การหมุน (Rotation)

การหมุนในระบบ 3 มิติ จำเป็นต้องมีการกำหนดแกนหมุนว่าจะเป็นแกน X, Y หรือ Z



รูปที่ 2.7 แสดงการหมุนรอบแกนต่างๆ [2]

1. การหมุนแนวแกน Z

การหมุนแนวแกน Z นี้ จะมีลักษณะคล้ายกับการหมุนใน 2 มิติ ดังแสดงในเมตริกซ์ของการหมุนแนวแกน Z ดังนี้

$$\begin{bmatrix} x' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

รูปที่ 2.8 แสดงเมตริกซ์ของการหมุนแนวแกน Z

ซึ่งเทียบเท่ากับสมการดังนี้

$X' = X \cos\theta - y \sin\theta$

$Y' = X \sin\theta + y \cos\theta$

$Z' = Z$

2. การหมุนแนวแกน X

การหมุนแนวแกน X ค่าที่เปลี่ยนแปลงคือค่าตามแนวแกน Y และ Z

$$\begin{bmatrix} x' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

รูปที่ 2.9 แสดงเมตริกซ์ของการหมุนแนวแกน X

สมการของการหมุนตามแนวแกน X สามารถเขียนได้ดังนี้

$Y' = Y \cos\theta - Z \sin\theta$

$Z' = Y \sin\theta + Z \cos\theta$

$X' = X$

3. การหมุนแนวแกน Y

การหมุนแนวแกน Y ค่าที่เปลี่ยนแปลงคือค่าตามแนวแกน Z และ X ดังแสดงในเมตริกซ์การหมุนแนวแกน Y ดังนี้

$$\begin{bmatrix} x' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

รูปที่ 2.10 แสดงเมตริกซ์ของการหมุนแนวแกน Y

ซึ่งสามารถเขียนให้อยู่ในรูปสมการได้ดังนี้

$$Z' = Z \cos\theta - X \sin\theta$$

$$X' = Z \sin\theta + X \cos\theta$$

$$Y' = Y$$

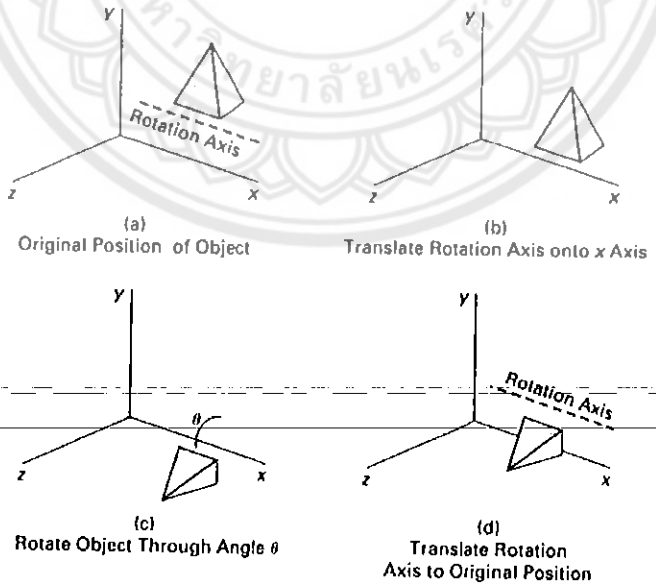
จะเห็นว่าสมการการหมุนแต่ละแกน สามารถทำ ได้โดยการแทนค่าแกน จากแกนหนึ่งเป็นอีกแกนหนึ่งดังแสดงในรูปของ cyclic permutation of the Cartesian-coordinate axes ดังนี้

$$X \longrightarrow Y \longrightarrow Z \longrightarrow X$$

4. การหมุนเทียบกับแกนสมมติซึ่งขนานกับแกนหลัก

การหมุนแบบนี้ มีหลักการคล้ายกับการย่อ- ขยายวัตถุที่มีจุดอ้างอิงที่ไม่ใช่จุดกำเนิด (origin coordinate) ดังนี้

1. ย้ายแกนสมมติไปยังแกนหลักที่แกนสมมตินั้นขนาน
2. หมุนวัตถุตามแกนนั้นๆ
3. ย้ายแกนสมมติกลับไปยังที่เดิม



รูปที่ 2.11 แสดงการหมุนเทียบแกนสมมติ [2]

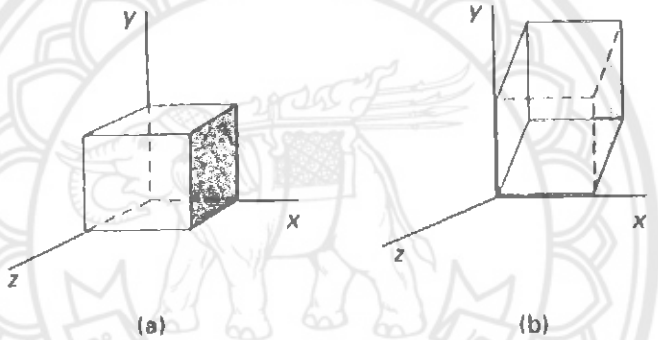
2.5.4 การบิดภาพ (Shear)

การบิดภาพถูกใช้ในการปรับเปลี่ยนรูปร่างของวัตถุ ซึ่งมีประโยชน์มากในระบบ 3 มิติ การบิดภาพในแนวแกน Z สามารถเขียนในรูปของเมตริกซ์ ได้ดังนี้

$$\begin{bmatrix} x' \\ Y' \\ Z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & a & 0 \\ 0 & 1 & b & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

รูปที่ 2.12 แสดงเมตริกซ์ของการบิดภาพ

parameter a และ b เป็นค่าคงที่จำ นวนจริง



รูปที่ 2.13 รูป (a) เป็นรูปก่อนการแปลง
รูป (b) เป็นรูปหลังการบิดตามแนวแกน Z โดยค่า a = b = 1 [2]

2.6 หลักการในการนำ เสนอวัตถุ 3 มิติ (3D-Concept)

การนำ เสนอวัตถุในระบบ 3 มิติ บนระบบภาพ 2 มิติ นั้นจะต้องผ่านขั้นตอนการทำงานหลายขั้นตอนด้วยกัน กล่าวคือ หลังจากการ model วัตถุ 3 มิติ แล้ว จำเป็นต้องมีการตัดส่วนที่อยู่นอกกรอบ (clipping) ที่กำหนดโดยกรอบดังกล่าวเรียกว่า ปริมาตรภาพ (view volume) หลังจากนั้นตำแหน่งต่างๆ ของวัตถุจะถูกแปลงจากพิกัดในระบบ 3 มิติ ให้เป็นระบบ 2 มิติ เพื่อนำเสนอทางจอภาพต่อไป ทั้งนี้วิธีการแปลงจะขึ้นอยู่กับลักษณะ projection ที่ใช้ด้วย

2.6.1 Projection

เป็นวิธีการหนึ่งที่สามารถสร้างภาพ 2 มิติได้ โดยเทคนิคนี้จะเสมือนกับการฉายเงาของวัตถุไปตกบนระนาบหนึ่ง ระนาบนี้เรียกว่า ระนาบภาพ (view plane or projection plane)

ระนาบภาพ คือ ระนาบที่เป็นฉากสำหรับเงาของวัตถุ ภาพของเงาที่เกิดบนระนาบ จะเป็นภาพ 2 มิติของวัตถุและเป็นภาพที่จะถูกวาดออกทางจอภาพ หรืออุปกรณ์พิมพ์ภาพ ระนาบภาพสามารถเป็น

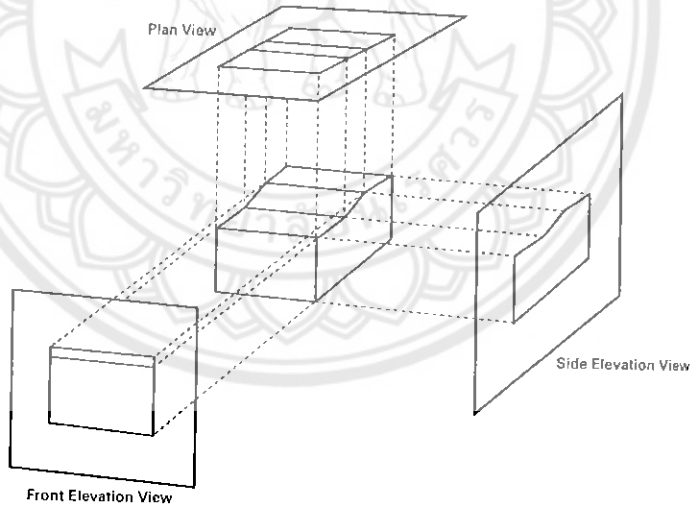
ระนาบใดๆ ก็ได้ที่ตั้งอยู่บนระบบ coordinate 3 มิติ (object coordinate) จะมองระนาบภาพนี้เสมือนกับระนาบ XY ธรรมดา (view plane coordinate)

อาจเปรียบเทียบระนาบภาพกับฟิล์มของกล้องถ่ายรูป วัตถุที่ต้องการบันทึกภาพ คือวัตถุในระบบ object coordinate เงาของวัตถุจะตกลงบนแผ่นฟิล์มในกล้อง เกิดเป็นภาพบนฟิล์มในระบบ view plane coordinate ขณะเดียวกัน ตำแหน่งของฟิล์ม คือ ตำแหน่งที่เรายืนมองวัตถุด้วย เราสามารถย้ายตำแหน่ง หรือเปลี่ยนมุมของกล้องถ่ายรูป เพื่อให้ได้ภาพถ่ายที่ออกมามีลักษณะที่ต่างกันออกไปภาพที่เกิดขึ้นบนระนาบ จะปรากฏในลักษณะใดนั้น ขึ้นอยู่กับการวางตัวของระนาบภาพ กับทิศทางของแสงที่ฉายเงาลงบนระนาบนั้นรวมไปถึงองค์ประกอบอื่นๆ

การทำ Projection ที่นิยมใช้จะมี 2 ประเภท คือ

1. Parallel Projection ในการ project ภาพแบบ parallel นั้นสามารถแบ่งออกได้เป็น 2 ประเภท คือ

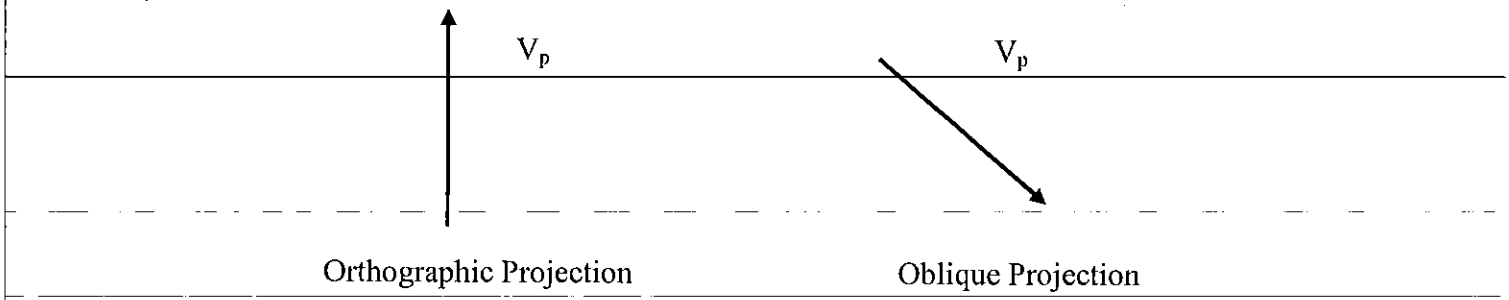
- Orthographic Parallel Projection เป็นการ Project ที่ตั้งฉากกับระนาบภาพ หากขนานกับแกน X Y หรือ Z ด้วย ในลักษณะนี้ การหาระนาบบนพิกัด 2 มิติ (ซึ่งเปรียบเสมือนระนาบ XY บนจอภาพ) ทำได้โดยการตัดพิกัดบนแกนนั้นออกเท่านั้น Orthographic Parallel Projection มีประโยชน์มาก เพราะจะทำให้สามารถแสดงภาพในมุมมองจากด้านบน ด้านข้าง ด้านหน้า หรือแม้กระทั่งด้านหลัง



รูปที่ 2.14 Orthographic parallel project [3]

อย่างไรก็ตาม หากใช้วิธี orthographic parallel projection แสดงภาพของวัตถุ เพื่อสามารถมองเห็นในเชิง 3 มิตินั้น (มองเห็นได้มากกว่า 1 ด้านบนระนาบภาพ) จะเรียกว่า *axonometric orthographic projection*

- Oblique parallel projection เป็นการ project ที่ขนานแต่ไม่ตั้งฉากกับระนาบภาพ

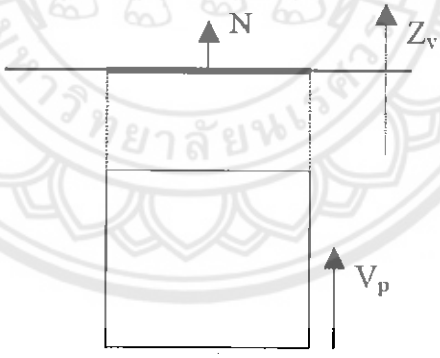


รูปที่ 2.15 แสดงทิศทางของ Orthographic และ Oblique Projection

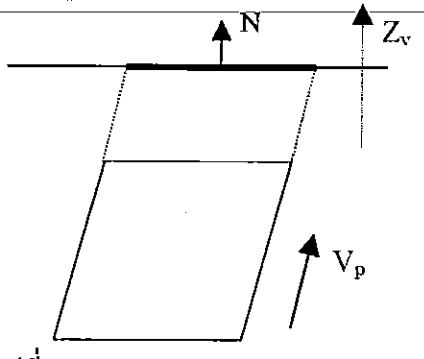
ดังที่ได้กล่าวมาแล้วว่า หากทิศทางของการ project ขนานกับแกน X Y หรือ Z พิกัดของวัตถุบนระนาบ 2 มิติ ทำ ได้โดยการตัดค่าของแกนที่ขนานนั้นออก แต่หากทิศทางของการ project เป็นไปในทิศทางอื่นเช่น เป็นไปตามเวคเตอร์ [px py pz] การคำ นวณหาพิกัดสามารถทำ ได้ดังนี้

$$x' = x - z(px / pz)$$

$$y' = y - z(py / pz)$$

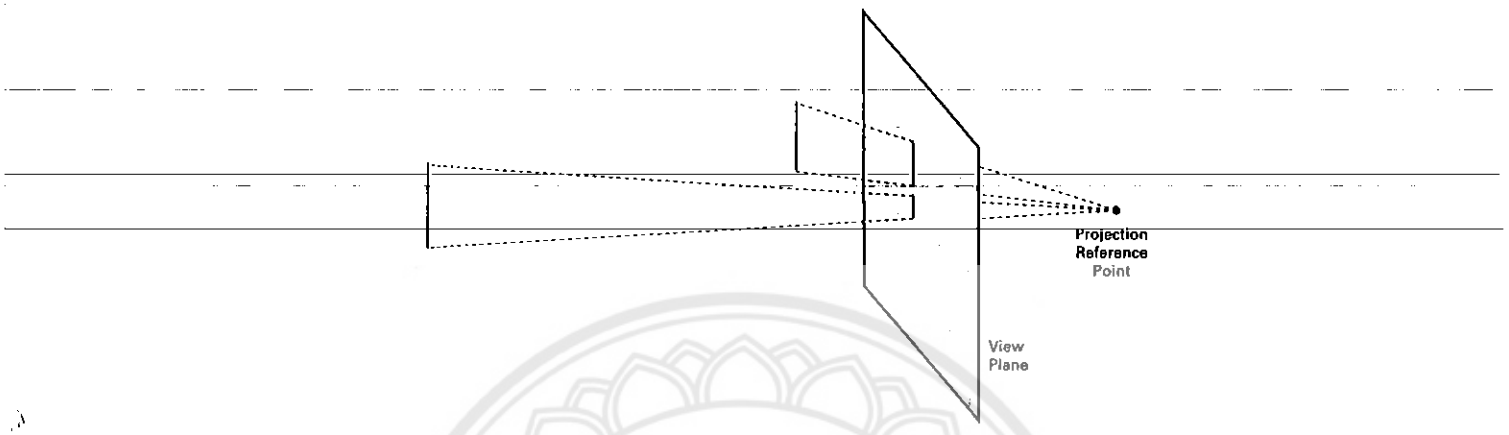


รูปที่ 2.16 ตัวอย่าง Orthographic projection



รูปที่ 2.17 ตัวอย่าง Oblique projection

2. Perspective projection ภาพที่เกิดจาก perspective projection นั้นเปรียบได้กับภาพที่เกิดจากการยิงลำแสงพุ่งเข้าหาจุดๆ หนึ่ง เรียกว่า projection reference point ดังนั้นภาพที่ปรากฏจะมีลักษณะอย่างไร ขึ้นกับระยะของจุดนี้และระยะของระนาบภาพ



รูปที่ 2.18 Perspective projection [3]

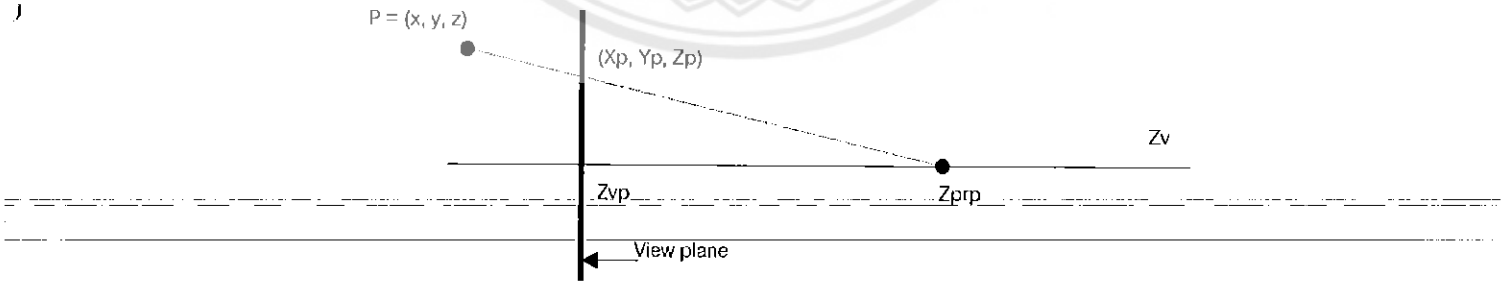
สมมติกำหนดให้ projection reference point อยู่ที่จุด Zprp ตามแนวแกน **Z** และกำหนดตำแหน่งของฉาก (view plane) ไว้ที่ Zvp ตำแหน่งของพิกัดตามแนว projection สามารถคำนวณได้ดังนี้

$Xp = x(dp / (Zprp - z))$

$Yp = y(dp / (Zprp - z))$

$Zp = Zvp$

เมื่อ $dp = Zprp - Zvp$ ซึ่งเป็นระยะห่างระหว่างฉากกับจุด projection reference point



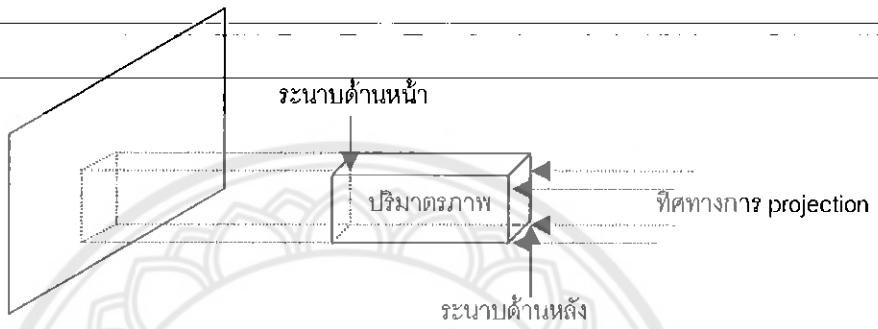
รูปที่ 2.19 แสดงพิกัดการ Perspective projection [3]

2.6.2 ลักษณะของปริมาตรภาพ

วัตถุต่างๆ ที่จะปรากฏบนจอภาพ จะต้องปรากฏอยู่ภายในปริมาตรภาพ ลักษณะของปริมาตรภาพ มี 2 ลักษณะ ตามลักษณะของ projection เช่นกัน คือ

1. Parallel Projection

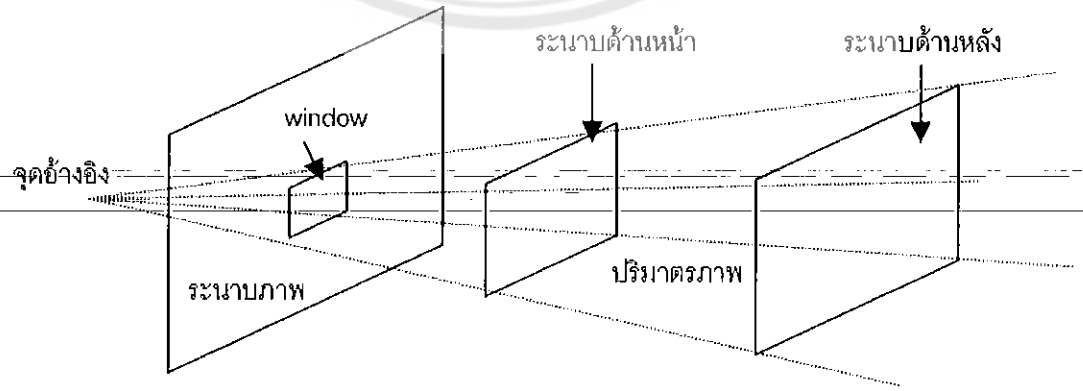
ปริมาตรภาพมีรูปทรงเป็นกล่องสี่เหลี่ยม กำหนดขอบเขตของ window บนระนาบภาพ จากขอบเขตแต่ละด้านของ window เมื่อลากเส้นจากขอบแต่ละด้านในทิศทางของ projection ทำ ให้ได้ระนาบด้านข้าง 2 ด้าน ด้านบนและด้านล่าง รวมทั้งสิ้นมีระนาบที่จะบอกขอบเขตของการแสดงวัตถุ 4 ด้าน จากนั้นต้องมีการกำ หนดอีก 2 ระนาบ คือ ระนาบด้านหน้า และหลังวัตถุที่อยู่ภายในปริมาตรที่กำหนดนี้ จะปรากฏที่จอภาพ



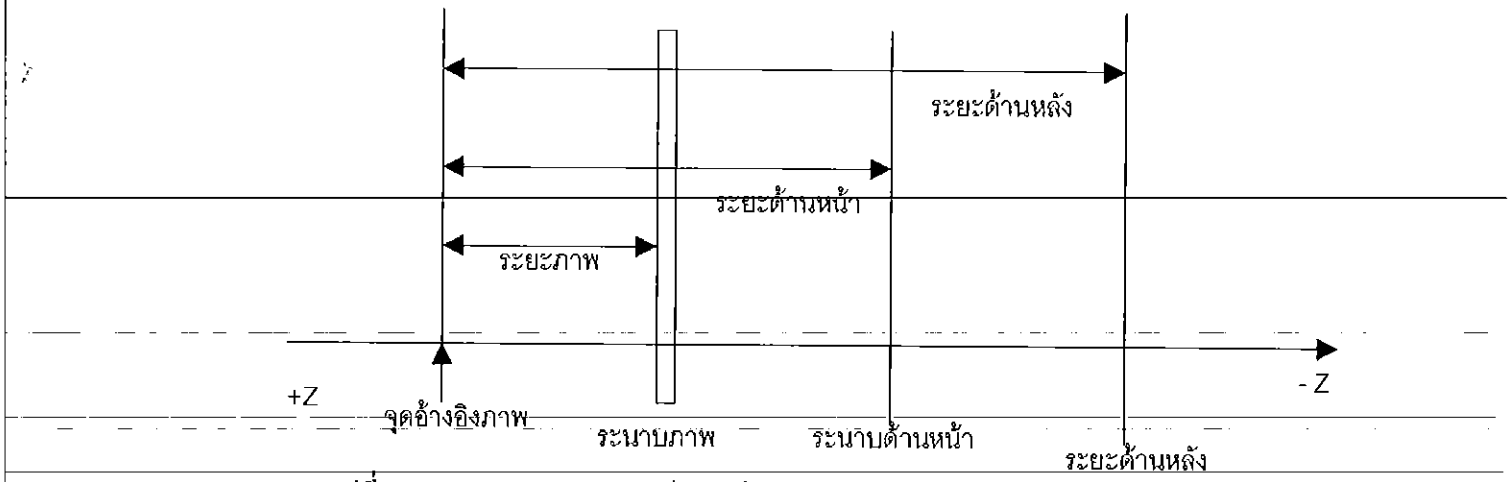
รูปที่ 2.20 แสดงปริมาตรภาพของ Parallel Projection [3]

2. Perspective Projection

การกำหนดปริมาตรภาพแบบนี้ ต้องกำหนดขอบเขต หรือกรอบของ window บนระนาบภาพก่อนขั้นตอนนี้ คล้ายกับการกำหนด window ในระบบ 2 มิติ window บนระนาบภาพจะเป็นตัวกำหนดระนาบด้านข้างทั้ง 4 (บน – ล่าง – ซ้าย ขวา) ของปริมาตรภาพ ส่วนระนาบด้านหน้าและหลัง กำหนดได้ด้วยระยะห่างระหว่างระนาบทั้ง 2 กับระนาบภาพ ทั้งระนาบด้านหน้าและระนาบด้านหลัง จะขนานกับระนาบภาพ ดังนั้นเราเพียงกำ หนดระยะห่างระหว่างระนาบทั้งสอง กับระนาบภาพก็เพียงพอแล้ว



รูปที่ 2.21 ลักษณะของปริมาตรภาพของ projection แบบ perspective [3]



รูปที่ 2.22 การกำหนดระนาบด้านหน้า- หลังของปริมาตรภาพ แบบ perspective [3]

2.6.3 การตัดภาพ (Clipping)

ในระบบภาพ 2 มิติ การตัดภาพทำ ได้โดยการตรวจสอบวัตถุกับเส้นขอบของ window แต่ สำ หรับระบบ 3 มิติ เราต้องตรวจสอบตำ แหน่งของจุดหรือวัตถุกับระนาบ แทนการตรวจสอบกับเส้น ระนาบที่ใช้ในการตรวจสอบ คือระนาบทั้ง 6 ของปริมาตรภาพนั่นเอง เทคนิคในการตัดภาพ อาจใช้วิธี polygon clip ก็ได้ โดยการ clip ออกทีละด้าน โดยเริ่ม clip ด้านหน้าและหลังเข้าไป และต้องเปลี่ยนการเปรียบเทียบของเส้นขอบของ window เป็นการเปรียบเทียบกับระนาบของปริมาตรภาพแทน

สมการระนาบ $Ax + By + Cz + D = 0$

เช่น P1 (x1, y1, z1) : $Ax1 + By1 + Cz1 + D = 0$

ถ้าผลลัพธ์ของสมการระนาบ มีค่าเท่ากับ 0 จุด P นี้อยู่ภายในระนาบ

ถ้ามีค่ามากกว่า 0 จุด P อยู่ด้านบวก (ด้านหน้าระนาบ)

ถ้ามีค่าน้อยกว่า 0 จุด P อยู่ด้านลบ (ด้านหลังระนาบ)

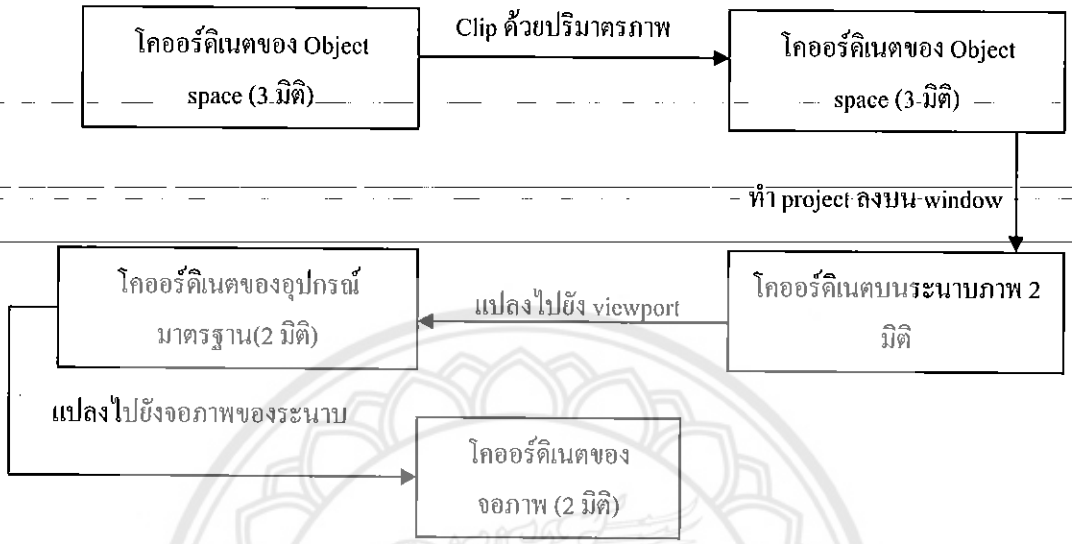
กรณีที่ทำ การ projection แบบขนาน โดยใช้ระนาบ XY เป็นระนาบภาพ ทิศทางของ projection ขนานกับแกน Z ภาพต่างๆ จะเกิดขึ้นบนระนาบ XY จากที่กล่าวข้างต้นว่าการทำ projection แบบนี้จะง่ายมาก คือ ตัดเอาค่า coordinate ทางแกน Z ทิ้ง เหลือแต่ coordinate แกน X และ Y ดังนั้นอาจนำเอาวิธี clip-a-line มาใช้กับระนาบด้านซ้าย – ขวา – บน – ล่าง ของปริมาตรภาพได้เลย

2.6.4 สรุปขั้นตอนการแปลงภาพจาก object space ในระบบ 3 มิติ ให้อยู่ใน image space

วัตถุต่างๆ ที่ถูกสร้างขึ้นในระบบ 3 มิติ เป็นการสร้างใน object space หมายความว่าการกำหนดตำแหน่ง และขนาดต่างๆ เป็นไปตามขนาดของวัตถุนั้นจริงๆ นอกจากนี้ระนาบภาพและปริมาตรภาพใช้โคออร์ดิเนตเดียวกันกับวัตถุ แต่ภาพที่เราต้องการจะต้องถูกสร้างขึ้นบนจอภาพ 2 มิติ

ซึ่งเป็นโคออร์ดิเนตของจอภาพนั่นเอง ดังนั้นจึงต้องแปลงภาพจากโคออร์ดิเนตใน object space ไปยังโคออร์ดิเนตของจอภาพ

ลำดับในการแปลงภาพในระบบ 3 มิติจาก object space ไปยังจอภาพระบบ 2 มิติ สรุปได้ดังนี้

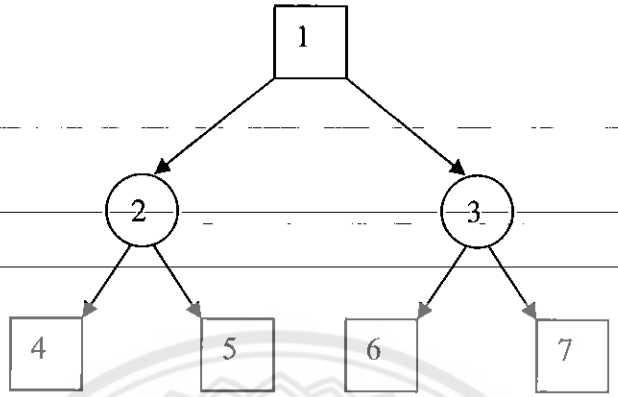


รูปที่ 2.23 แสดงลำดับการแปลงภาพ

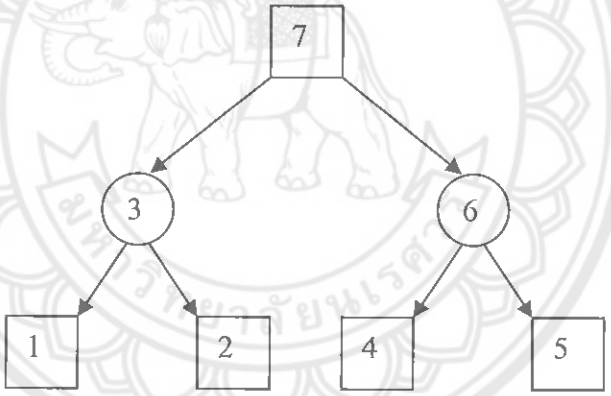
2.7 หลักการทางปัญญาประดิษฐ์ (Artificial Intelligence)

อาจกล่าวได้ว่าหลักการGame Tree Search, Minimax, Negamax, Alpha-Beta in Negamax Function คือหัวใจของโปรแกรมหมากฮอสก็ได้ การทำให้คอมพิวเตอร์รู้จักกติกานั้นไม่ยาก แต่จะทำให้คิดเพื่อแข่งกับคนตรงนี้ยากกว่า ลองพิจารณาดูว่า ณ ตำแหน่งใดตำแหน่งหนึ่ง มีตาเดินที่เป็นไปได้หลายตา หากลองเลือกเดินดูตาหนึ่งก็จะพบว่าตาต่อไปคู่ต่อสู้ก็สามารถเลือกเดินได้หลายตาเช่นกัน และหากต้องการคิดที่ระดับลึกหลายชั้น ความเป็นไปได้ของรูปแบบที่เป็นไปได้ก็จะยิ่งมีหลากหลายรูปที่เกิดขึ้น ลักษณะที่มีการแตกกิ่งขยายไปอื่นมากมายเป็นลักษณะทวีคูณ จำนวนรูปต่างๆ ที่เกิดขึ้นนี้จึงถูกเรียกคล้ายกับกิ่งของต้นไม้ คือ Game Tree และวิธีที่นำมาลดทอนการขยายของจำนวนแขนงของ Game Tree ก็เรียกว่า Pruning (แปลว่าการตัด หรือเล็ม) วิธีที่นิยมนำมาใช้ในการคิดคำนวณโดยทั่วไปก็คือ Depth First Procedure คือเหมือนให้โปรแกรมลองเดินดูโดยเลือกตาเดินตาหนึ่งจากตาเดินที่เดินได้ทั้งหมด เมื่อเดินแล้วมีการเปลี่ยนแปลงข้อมูลบนกระดานแล้ว จากนั้นก็ลองดูอีกว่า ณ ตานั้นอีกฝ่ายหนึ่งจะสามารถเดินอะไรอีก เลือกตาเดินและสลับข้างอีก จากนั้นก็ทำการเดินพร้อมกับสลับฝ่ายลึกลงไปเรื่อยๆ จึงเรียกวิธีนี้ว่า Depth First Procedure จนกระทั่งถึงระดับความลึกในการคิดที่ต้องการ ก็ใช้ Static Evaluation Function คำนวณอย่างคร่าวๆ ให้ค่าออกมาเป็นตัวเลขว่าฝ่ายใดได้เปรียบฝ่ายใดเสียเปรียบเท่าไร รายละเอียดของ Static Evaluation Function เมื่อได้ค่าจาก Evaluation Function ก็ส่งค่าย้อนขึ้นกลับมาเป็นชั้นๆ จากนั้นก็พิจารณาตาเดินอื่นต่อ จนหมดตามต้องการ ดังรูปจะแสดงถึงลำดับในการ

Generate Move และลำดับในการ Evaluate Move นั้น โดยในรูปเสมือนหนึ่งว่าแต่ละตาเดินมีตาเลือกเดินได้เพียงสองตา ซึ่งในข้อเท็จจริงในเกม หากไม่ใช่ตากินซึ่งเป็นตาบังคับเดินแล้วมักจะมีตาเดินให้เลือกประมาณ 10 ตา ทั้งนี้เพื่อความเข้าใจง่ายในการแสดงภาพให้เห็น



รูปที่ 2.24 Diagram แสดงลำดับของการ Generate Move แบบ Depth First Procedure



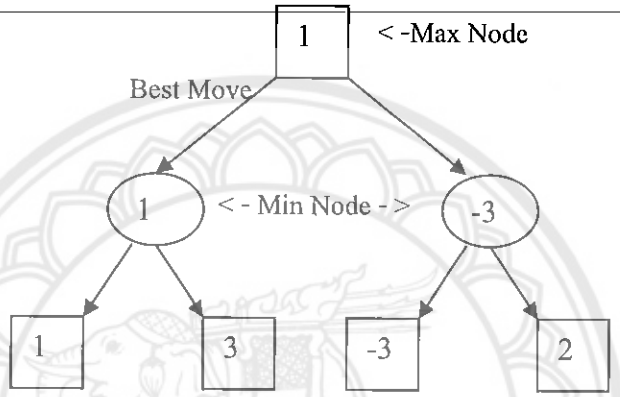
รูปที่ 2.25 Diagram แสดงถึงลำดับการ Evaluate ของแต่ละ Node แบบ Depth First Procedure

สังเกตว่า Node แรกที่ถูก Evaluate คือ Node ลึกที่สุดที่ได้รับการ Generate ก่อน และ Node สุดท้ายก็คือ Root Node

2.7.1 Minimax Function

โดยหลักการก็คือ ต้องถือว่าโปรแกรมต้องเลือกตาเดินที่ดีที่สุด ในขณะเดียวกันคู่ต่อสู้ก็ต้องเลือกตาเดินที่ดีที่สุดสำหรับฝ่ายเขาเช่นเดียวกัน จึงเรียกว่าเป็น Minimax Function โดยที่โปรแกรมของเราขณะนี้ไม่มีความรู้เรื่องหมากฮอสอะไรเลย วิธีคิดก็อย่างที่กล่าวมาใช้ Depth First Procedure เมื่อได้ค่าคำนวณได้ของแต่ละตาเดิน หรือ Node นั้นๆ พิจารณาดูว่าที่ระดับความลึกนั้นเป็นเลขคู่ (Even Depth) หรือ เลขคี่ (Odd Depth) หากเป็น Even Depth ก็ให้โปรแกรมเลือก Node ปลายทาง (Leaf Node)

ที่มีค่ามากที่สุด (Maximum) เพื่อส่งค่าย้อนขึ้นไปยัง Node ที่เหนือกว่าขึ้นไป เหตุผลก็คือเมื่อโปรแกรมเป็นฝ่ายเดินก็ต้องหาตาเดินที่ทำแล้วได้เปรียบมากที่สุด หรืออีกนัยหนึ่งเลือกตาเดินที่ได้คะแนนสูงสุด หากเป็น Odd Depth หรือฝ่ายตรงข้ามเดินก็ให้เลือกตาเดินที่ให้คะแนนที่ต่ำที่สุด (Minimum) เพื่อส่งกลับขึ้นไปในระดับที่เหนือกว่า เหตุผลก็คือคู่ต่อต้องสู้เลือกตาเดินที่ทำให้โปรแกรมได้คะแนนน้อยที่สุด อย่างนี้ไปเรื่อยๆ จนกว่าจะถึง Root Node คือ Node ที่อยู่บนสุด ก็จะได้คำตอบว่าตาเดินใดจะดีที่สุดในตำแหน่งนั้นๆ ดังรูป เป็นรูปแสดงถึงวิธีการคิดคำนวณโปรแกรมหมากฮอสของเราและส่งค่าคืนกลับมา

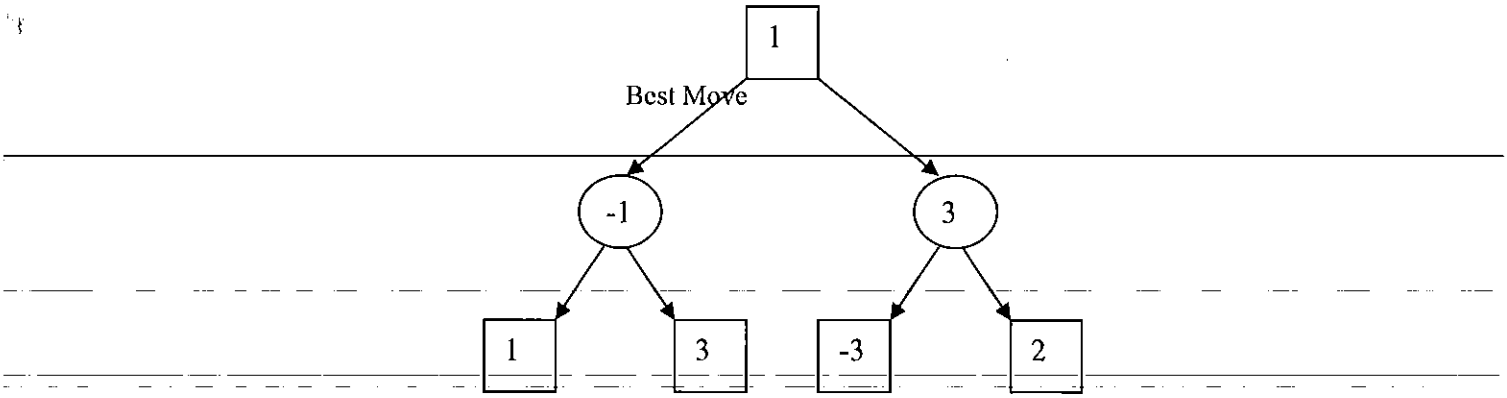


รูปที่ 2.26 Diagram แสดงการทำงานของ Minimax Function

จาก Diagram แสดงการทำงานของ Minimax Function ข้างบนแสดงผลของ Evaluation Function ที่ leaf node จะเห็นว่าที่ Odd Depth หรือ Min Node จะเลือกค่าตัวเลขที่ต่ำที่สุดที่ได้จาก Node ที่ต่ำกว่าแล้วส่งค่าขึ้นไป Node ที่สูงกว่า ส่วนที่ Max Node ก็ตรงกันข้าม และเมื่อถึง Root Node คือ Node ที่สูงสุดใน Tree ตาเดินที่ให้คะแนนที่สูงสุดก็คือ ตาเดินที่ดีที่สุดที่โปรแกรมควรเลือกเดิน

2.7.2 Negamax Function

เป็นการดัดแปลงจาก Minimax Function แทนที่จะแยกว่าเป็น Odd Depth (Min Node) หรือ Even Depth (Max Node) ก็ใช้คุณสมบัติของ -1 มาช่วยกล่าวคือ ที่ Min Node แทนที่จะเลือกตาเดินที่ให้ค่ากลับมาน้อยที่สุด ก็เพียงเปลี่ยนเป็น นำ -1 คูณกับทุกค่า จากนั้น return ค่าที่มากที่สุดแทน เมื่อค่าดังกล่าวมาถึง Max Node ก็นำ -1 มาคูณ เพื่อให้กลับเป็นค่าเดิม (-1 x -1 = 1) เหมือนวิธี Minimax แล้วก็ return ค่าที่มากที่สุดเหมือนเดิม วิธีนี้จะดีกว่าคือไม่ต้องมาแยกแยะว่าเป็น Max Node หรือ Min Node ก็จะได้ค่าที่ Root Node เหมือนเดิม หากแต่สามารถเขียนเป็นคำสั่งได้ง่ายกว่ามาก เพราะเพียงแต่ นำค่า -1 มาคูณแล้วก็เลือกค่าที่มากที่สุด return ขึ้นไป ซึ่งสามารถทำให้เขียนโปรแกรมในรูปของ recursive function ได้จึงเรียกวิธีนี้ว่า Recursive Negamax Function คือเลือกค่ามากที่สุดจาก ค่า Negative Value ของตัวนั้น ดังรูป



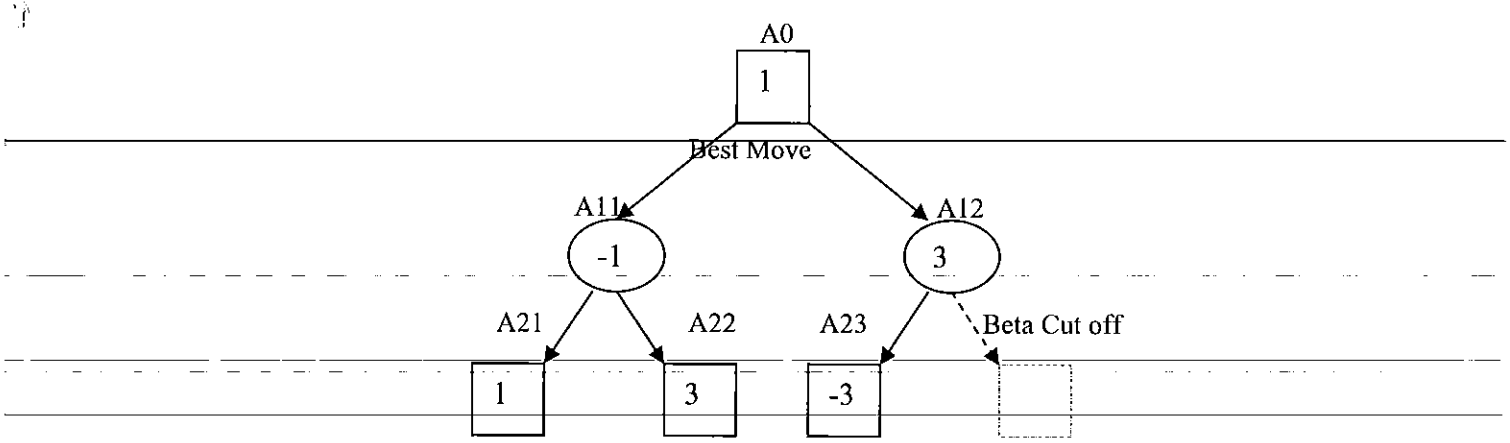
รูปที่ 2.27 Diagram แสดงการทำงานของ Negamax Function

จาก Diagram ข้างบนแสดงถึง การทำงานของ Negamax Function คล้ายกับ Minimax Function เพียงแต่ที่ Node ใดๆ ก็ตามให้เลือกตาเดินที่ให้ค่า Negative Value ที่สูงสุดแทน ก็จะได้ผลเช่นเดียวกันกับ Minimax Function เพียงแต่การเขียน Algorithm ง่ายกว่า เมื่อนำ Negamax Function มาประยุกต์ใช้กับโปรแกรมโดยเขียนเป็น recursive function มีข้อที่ต้องให้ความสำคัญก็คือ ในหมากฮอสมีการกินต่อเนื่องต้องระวังจุดนี้ด้วย หากเป็นการกินต่อเนื่อง (Double Jump) คะแนนที่ได้จากการเดินในชั้นต่อไปเป็นคะแนนของฝ่ายเดียวกัน ดังนั้น recursive function ต้องใส่เครื่องหมายให้ถูกต้องด้วย

2.7.3 Alpha-Beta in Negamax Function

เมื่อพิจารณาจาก Minimax Function จะเห็นว่าเป็นการวิเคราะห์ทุกตาที่เป็นไปได้อย่างละเอียดถี่ถ้วนทุกตา แต่ข้อเสียคือจะใช้เวลานานมาก ตัวอย่างเช่น สมมติว่า ณ ตำแหน่งใดๆ มีตาเดินที่เป็นไปได้ 10 ตาเดิน หากเราต้องการคิดลึกลงไป 6 ชั้น เท่ากับเราต้องคิดคำนวณ 10 x 10 x 10 x 10 x 10 x 10 = 1,000,000 Node จึงมีผู้ริเริ่มใช้ Alpha-Beta Algorithm ซึ่งจะทำให้สามารถลดจำนวน Node ลงไปได้เป็นอย่างมาก

พิจารณาดูตามรูปตัวอย่างที่ข้างล่าง จะเห็นว่า เมื่อเราได้ค่า A21 A22 เราก็จะได้ค่าของ A11 เท่ากับ -1 ขณะที่พิจารณา Node ที่ A12 เพื่อหาค่ามาเทียบกับ A11 เมื่อได้ค่าจาก A23 ครั้งแรกก็ทำให้ A12 มีค่าอย่างต่ำเท่ากับ 3 หรือมากกว่านั้น ทำให้ไม่มีความจำเป็นต้องหาค่าของ A24 ก็ได้ เนื่องจากสมมติว่า ค่า A24 มากกว่า -3 (คือตั้งแต่ -2 ขึ้นไป) ก็ไม่สามารถเปลี่ยนแปลงค่าของ A12 ได้ เพราะเราต้องเลือกค่าที่สูงสุดของ Negative value ซึ่งเท่ากับ 3 ขณะเดียวกันหากค่าของ A24 น้อยกว่า -3 (คือตั้งแต่ -4 ลงไป) จะทำให้ค่าของ A12 มากขึ้นไปอีก แต่กรณีนี้ค่า Negative value มากกว่าค่า Negative value ของ A11 ซึ่งเท่ากับ -1 หรือ beta ในขณะนั้น ก็ไม่สามารถเปลี่ยนแปลงค่าของ A0 ได้อยู่ดี จึงไม่จำเป็นต้องหาค่าของ A24 แต่อย่างใด เรียกการตัดการคิดคำนวณต้องนี้ออกว่า Beta cut off ซึ่งจะทำให้มีการประหยัดเวลาในการทำงานได้มากกว่า Negamax Function ธรรมดาอย่างมาก



รูปที่ 2.28 Diagram แสดงการทำงานของ Alpha-Beta in Negamax Function

บทที่ 3
วิธีการดำเนินการ

วิธีการสร้างเกมหมากฮอส 3 มิตินั้นเราจะดำเนินการโดยใช้ OpenGL ที่ลงใน Microsoft Visual Studio 6 เป็นตัวหลักในการสร้างเกม โดยในบทนี้จะอธิบายถึงขั้นตอนและวิธีการคิดของตัวเกม ว่าทำอย่างไรถึงมีความสามารถในการที่จะเอาชนะคู่ต่อสู้ได้

3.1 การใช้ OpenGL ใน Microsoft Visual Studio 6

ขั้นตอนการเริ่มใช้ OpenGL ใน Microsoft Visual Studio 6 มีดังนี้

3.1.1 ทำการ Download GLUT ของ OpenGL

3.1.2 ทำการ Copied GLUT ดังนี้

glut32.dll to C:\Windows\system

glut32.lib to \..\..\VC98\lib

glut.h to \..\..\VC98\include\GL

3.1.3 เปิด Microsoft Visual C++ เวอร์ชัน 6

3.1.4 จากนั้นไปที่ File > New

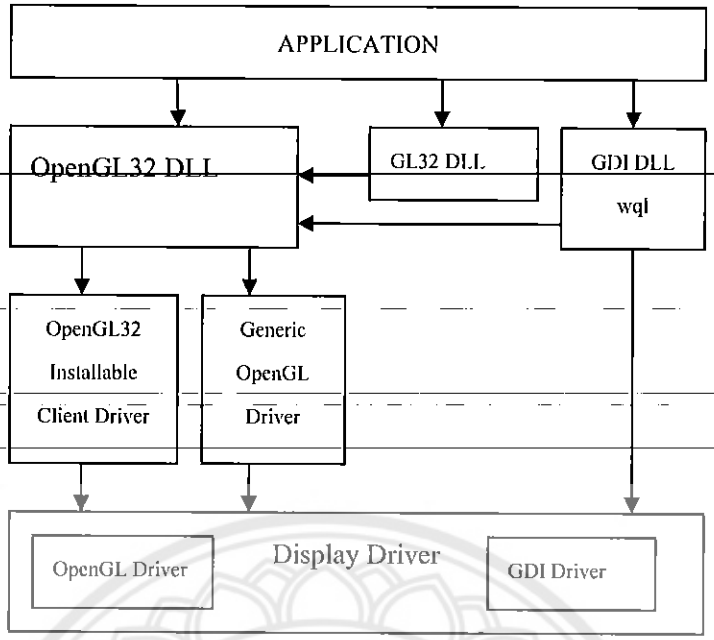
3.1.5 เลือก Win 32 Console Application

3.1.6 ตั้งชื่อ Project Filename ที่ต้องการ

3.1.7 คลิก OK

3.1.8 เมื่อเริ่มทำงานจะต้องประกาศชนิด #include“GL\glut.h” ประกาศเพื่อเรียกใช้ Library ของ OpenGL

3.1.9 หรือประกาศเป็น #include <windows.h>, #include <GL\gl.h>, #include<GL\glu.h> ก็ได้



รูปที่ 3.1 โครงสร้างของ OpenGL

3.2 การสร้างกระดานหมากฮอส

ขั้นตอนการสร้างกระดานหมากฮอส มีดังนี้

3.2.1 การสร้างกระดานตารางเพื่อให้ง่ายต่อการสร้างจึงต้องให้ใช้เป็น Array Board ซึ่งได้สร้างเป็น Board[35] เพื่อเก็บข้อมูลทั้งหมดของกระดานเพื่อเก็บข้อมูลทั้งหมดของกระดาน 32 ตารางว่าแต่ละตารางมีตัวอะไรตั้งอยู่บนนั้น

จะเห็นว่าทำไมไม่ใช้ Board[32] ทั้งที่กระดานหมากฮอสมี 32 ตาราง ดังนั้นมาพิจารณาความแตกต่างของ Board[35] กับ Board[32]

3.2.2 พิจารณา Board[32]

00		01		02		03	
	04		05		06		07
08		09		10		11	
	12		13		14		15
16		17		18		19	
	20		21		22		23
24		25		26		27	
	28		29		30		31

รูปที่ 3.2 ตัวอย่าง Board[32]

ในขั้นตอนต่อไปที่จะหาตาเดินทั้งหมด (Move Generation Function) ที่เป็นไปได้สำหรับตำแหน่งนั้น ๆ ค่าความห่างระหว่างตาที่เบี้ยตั้งอยู่กับตาที่ต้องการเดิน ในแต่ละแถวจะไม่เท่ากันมีความแตกต่างระหว่างแถวคู่ กับแถวคี่ ไปสมมติว่ามีเบี้ยฝ่ายขาว (WHITE BIA) อยู่ที่ตาที่ 29 หรือแถวที่หนึ่งเราจะบอกโปรแกรมว่าเบี้ยตัวนี้เดินขึ้นไปยังช่องที่ 24, 25 ได้ ค่าความแตกต่างเท่ากับ 4, 5 แต่ถ้าหากเป็นแถวคู่ เช่นตาที่ 24 ค่าความต่างจะเท่ากับ 4, 3 จะเห็นว่าเวลาคำนวณการเดินจะเกิดความยุ่งยาก

3.2.3. พิจารณา Board[35]

00		01		02		03		
	04		05		06		07	08
09		10		11		12		
	13		14		15		16	17
18		19		20		21		
	22		23		24		25	26
27		28		29		30		
	31		32		33		34	

รูปที่ 3.3 ตัวอย่าง Board[35]

จะเห็นว่าถ้าเพิ่มตาขึ้นมาอีก 3 ตาเป็นปกติที่ไม่มีอยู่จริงและไม่ได้เก็บข้อมูลอะไร คือ ตาที่ 8, 17, 26 ผลที่ได้คือค่าความห่างระหว่างตาที่ตั้งต้นกับตาที่จะเดินไปจะคงที่ กล่าวคือ เมื่อเราจะเขียน Function กำหนดตาเดินให้โปรแกรมค่าความห่าง จะเป็น 4, 5, -4, -5 เสมอ ซึ่งทำให้ง่ายต่อการเขียนโปรแกรมการเดิน

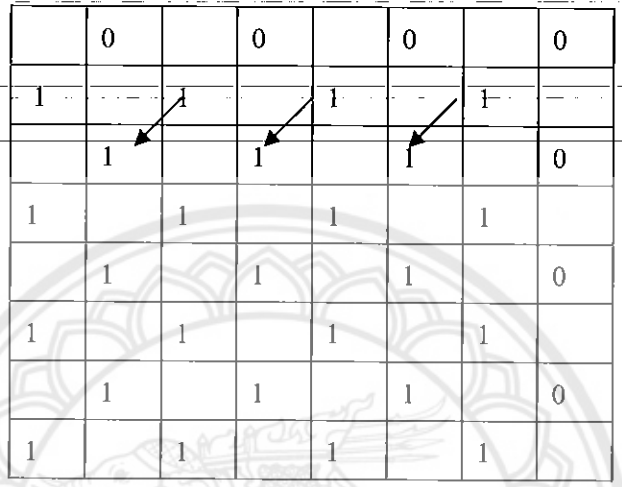
3.2.4 การวาดกระดานใช้คำสั่ง GL_QUADS ซึ่งเป็นการวาดกล่องสี่เหลี่ยม จากนั้นใช้การ Rotates ตำแหน่งให้ครบตามจำนวนช่องของตารางหมากฮอส

3.2.5 การสร้างตัวเบี้ยสร้างเป็นทรงกลมโดยใช้คำสั่ง glutSolidSphere() โดยสร้างเป็น 2 ข้าง โดยเบี้ยแต่ละข้างมี 8 ตัว และข้างละสี การวางเบี้ยก็วางตามลักษณะการเล่นของเกมหมากฮอสซึ่งเป็นที่ทราบกันดีอยู่แล้ว

3.3 การเริ่มต้นการเดินของตัวหมากฮอส

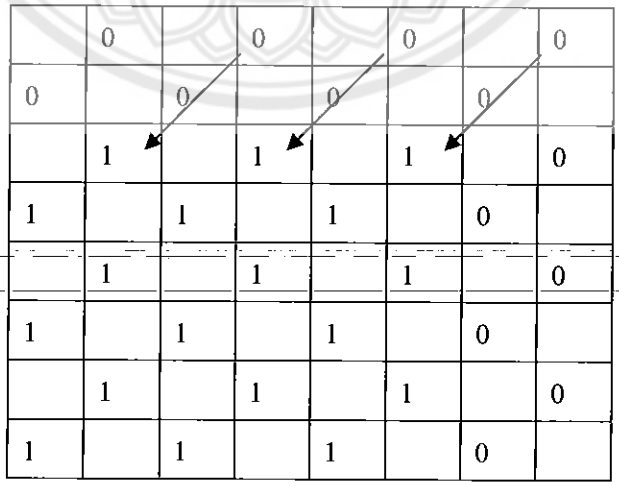
การเดินตัวเบี้ยสามารถกำหนดความตัวแปรต่างๆ ในช่วงเริ่มต้นเกมโดยรวมๆ ก็มี 3 กลุ่มหลักๆ คือ

3.3.1 ตัวแปรต่างๆ ที่ใช้ใน Move Generation Function เช่น Mask Array คือกลุ่มของ array ที่กำหนดว่าตารางใดบ้างที่สามารถเดินไปในทิศทางเช่น



รูปที่ 3.4 แสดงทิศทางการเดินของหมากฮอส

หมายถึงหากมีเบี้ยในตาเดินในช่องที่เป็นเลข 1 สามารถเดินไปในทิศทางขึ้นบนและเฉียงขวาได้ ส่วนตัวเลข 0 หมายถึงตานั้นไม่สามารถเดินไปในทิศทางดังกล่าวได้ เนื่องจากจะเป็นการเดินผิดกติกาหรือเดินไปในตาที่ไม่มีอยู่จริง



รูปที่ 3.5 แสดงทิศทางการกินของตัวหมากฮอส

คล้ายกันก็คือมีไว้เพื่อตรวจสอบว่าหากเรามีเบี้ยขาวในตาใดก็ตาม จะสามารถกินไปในทิศทางขึ้นบนและเฉียงขวาได้หรือไม่เพื่อไม่ให้เบี้ยเดินออกมานอกกระดานนั่นเอง ทั้งหมดนี้เพื่อจะได้นำไปใช้ใน Move Generation Function

3.3.2 ตัวแปรกลุ่มที่ต้องกำหนดใหม่เมื่อเริ่มเกมใหม่ หรือค่าคงที่ที่จำเป็นในส่วนต่างๆ ของโปรแกรมโดยทั่วไปก็เป็นตัวแปรเกี่ยวกับข้อมูลทั้งหมดของกระดานเช่น จำนวนเบี้ยของแต่ละฝ่ายเมื่อเริ่มต้นเล่นมีฝ่ายละ 8 ตัว และกำหนดฝ่ายเดินก่อน ตรงนี้มีส่วนที่เป็นเทคนิคอย่างหนึ่งซึ่งจะเป็นประโยชน์มาก คือควรจะกำหนดตัวเลขที่ใช้แทนตัวเบี้ยหรือตัวฮอสในลักษณะที่แยกจากกันได้ง่าย เมื่อข้อมูลทุกอย่างที่อยู่บนกระดานเป็นตัวเลขที่มี bit ไม่ตรงกันจะเป็นประโยชน์มากใน Move Generation Function เพราะเราจะสามารถแยกแต่ละตัวออกจากกันได้เร็ว ใน Source Code ตัวอย่างโปรแกรมจะเรียกช่องฝ่ายเลขมากกว่าเป็นฝ่ายขาว และเป็นฝ่ายเดินก่อน และกำหนดจุดเริ่มต้นของ Move list ในส่วนตัวแปรที่เกี่ยวข้องนี้ต้องทำการ Reset ใหม่ทุกครั้งในหากมีการเริ่มเล่นเกมใหม่โดยไม่ได้ปิดโปรแกรม

3.3.3 กลุ่มตัวแปรที่ใช้ใน Evaluation Function ตัวอย่างเช่น ค่าของหมากแต่ละตัวในโปรแกรมจะกำหนดอย่างคร่าวๆ ให้น้ำหนัก ฮอสหนึ่งตัวเท่ากับเบี้ยสองตัว ในกรณีแลกกัน ค่านี้จะคงที่ตลอดทั้งเกมไม่มีการเปลี่ยนแปลง ไม่ต้องมีการ Reset ใหม่แต่อย่างใด

3.4 การคิดลำดับการเดินแต่ละตำแหน่ง

เมื่อเราเก็บข้อมูลของกระดานหมากฮอสทุกอย่างอยู่ในรูปของ array[35] และใช้ตัวเลข Integer แทนตัวหมากต่างๆ และช่องว่าง จากนั้นต้องทราบว่าในตำแหน่งดังกล่าว ฝ่ายใดเป็นฝ่ายเดิน และมีตาเดินอะไรได้บ้าง ต้องเตรียมตัวแปรจำนวนหนึ่งเพื่อเก็บข้อมูลของ ตาเดินแต่ละตา ข้อมูลที่สำคัญของตาเดินก็จะมีดังต่อไปนี้

```
typedef struct
{
int from;        // ตาเดินนั้นเดินจากตาไหน
int to;          // เดินไปตาไหน
int over;        //หากเป็นการกิน ตัวที่ถูกกินอยู่ที่ตาไหน
int cap;         //หากเป็นการกิน ตัวที่ถูกกินคืออะไร
int type;        //เพื่อเป็นการแยกว่าเป็นการเดินปกติ หรือการกิน หรือกินต่อเนื่อง
} movetype;
```

เมื่อกำหนดลักษณะของ movetype แล้ว ต้องเตรียมตัวแปรลักษณะ array of movetype จำนวนหนึ่งเพื่อเก็บข้อมูลของตาเดินที่เป็นไปได้ทั้งหมด ดังเช่น

movetype tree[500];

ข้อสังเกตอย่างหนึ่งก็คือในการกินต่อเนื่องหลายๆ ต่อ เราจะแยกการกินแต่ละครั้งเป็นหนึ่งตา เพียงแต่ระบุว่าเป็นการกินต่อเนื่องเท่านั้น ขั้นตอนต่อไปคือการหาว่าตาเดินที่เป็นไปได้สำหรับตำแหน่งนั้น เรียงตามลำดับคือ

3.4.1 ดูข้อมูลจากตาเดินก่อนหน้านั้นว่าเป็นการกินต่อเนื่องหรือไม่ หากเป็นดังกล่าวตาเดินที่ถูกต้องก็คือตัวที่กินในตาก่อนหน้านั้นต้องกินต่อเท่านั้น ไม่ต้องพิจารณาตาเดินตาอื่น

3.4.2 ดูว่าฝ่ายใดเป็นฝ่ายเดิน จากนั้น วน loop ดูว่าบนกระดานนั้นมีตัวอะไรบ้างที่เดินได้ หากได้ตัวฝ่ายนั้นมาแล้วก็จึงมาดูว่าตัวนั้นเดินอะไรได้บ้าง เนื่องจากในหมากฮอสไทยตากินเป็นตาบังคับ ดังนั้นจึงหาตากินก่อน หากไม่มีจึงหาตาเดินปกติ

1. การกินของตัวเบี้ยยุ่งยากพอสมควรเพราะจะต้องดูว่าตัวนั้นอยู่ในตำแหน่งที่สามารถกินได้หรือไม่ และตาที่กระโดดข้ามไปมีคู่ต่อสู้อยู่หรือไม่ และตาที่จะกระโดดไปว่างหรือไม่ เมื่อครบทั้งสามส่วนแล้วยังต้องดูว่าเมื่อกระโดดข้ามไปแล้วเป็นตาเข้าฮอสด้วยหรือไม่ และทั้งหมดนี้เป็นการกินต่อเนื่องหรือไม่ เพื่อเป็นการเก็บข้อมูลอย่างครบถ้วน

2. การกินของฮอสเนื่องจากตัวฮอสเดินได้ทั้งสี่ทิศทาง และแต่ละทางยาวเท่าไรก็ได้ตราบเท่าที่ไม่ตกขอบ และไม่ติดตัวของพวกเดียวกันเอง จึงต้องใช้ while loop เข้ามาช่วยตรงนี้ มีประเด็นสำคัญตรงนี้อยู่คือว่า เมื่อพบว่าในตำแหน่งนั้นฮอสสามารถกินได้ เราต้องทำการเอาตัวฮอส และตัวที่กินได้นั้นออกไปก่อนเป็นการชั่วคราว เพื่อตรวจสอบหาว่ามีตัวที่กินได้ต่อเนื่องอีกหรือไม่เนื่องจากในบางกรณีสามารถกินในทิศทางย้อนกลับกับที่กินครั้งแรกได้

3.4.3 ตาเดินปกติมีขั้นตอนดังนี้

1. การเดินของเบี้ยค่อนข้างจะตรงไปตรงมาก็เพียงดูว่าตัวเบี้ยของฝ่ายไหนเดินไปทางซ้าย หรือขวาหากตาที่เดินไปได้นั้นว่างก็เดินได้และก็ต้องไม่ลืมที่จะตรวจสอบว่าเป็นการเดินเข้าฮอสหรือไม่

2. การเดินของตัวฮอสก็ยุ่งกว่าของตัวเบี้ยเพราะระยะเดินของตัวฮอสยาวกว่า ดังนั้นต้องใช้ while loop ช่วยคือตราบที่ยังเจอช่องว่าง ก็เพิ่มวน loop จนกว่าจะสิ้นสุดทุกตาที่เป็นไปได้

3.5 การเดินหน้าและการเดินย้อนกลับ

หมายถึงให้โปรแกรมเดินหน้าและย้อนกลับ ใน Makemove() ก็คือให้เดินไปตามตาเดินนั้นๆ ส่วน Unmakemove() คือ ย้อนตาเดินที่เพิ่งจะเดินไป อันนี้จะจำเป็นมากในขั้นตอนต่อไปของโปรแกรมเพื่อจะคิดคำนวณ ในส่วน Makemove() มีองค์ประกอบการทำงานหลักอยู่สามส่วนคือ

3.5.1 การเก็บข้อมูลทั้งหมดของตาเดินนั้น ๆ ก่อนที่จะมีการเปลี่ยนแปลงใด ๆ เพื่อที่จะได้ย้อนกลับได้อย่างถูกต้อง ซึ่งตรงนี้จะใช้ใน Search Function

3.5.2 ทำการเปลี่ยนแปลงข้อมูลในตำแหน่งต่างๆ ที่เราได้จากตาเดินนั้นๆ ใน board[35] ที่เก็บข้อมูลไว้ อาทิเช่น ตาที่เดินมีตัวนั้นอยู่ก็เปลี่ยนเป็นตาว่าง ถ้าเป็นการเข้าฮอสก็เพิ่มตัวฮอสเข้าไปแทนตัวเบี้ยในตำแหน่งนั้นๆ หากเป็นการกินก็ลบตัวที่ถูกกินออกไปจากกระดาน

3.5.3 ส่วนสุดท้ายก็คือทำหน้าสลับหรือไม่สลับฝ่ายที่จะเดินต่อไป ก็คือหากไม่ใช่การกินต่อเนื่องก็ทำการสลับฝ่ายเดิน แต่หากเป็นการกินต่อเนื่องก็ไม่ต้องสลับ ซึ่งตรงนี้ต่างกับโปรแกรมหมากรุกเพราะในหมากรุกจะไม่มีการกินอย่างต่อเนื่องต้องระวังตรงจุดนี้ให้ดี

ส่วน Unmakemove() ก็เช่นกันแต่ทำในลักษณะตรงข้ามกับ Makemove() โดยนำข้อมูลของตาเดินนั้นๆ มาจากที่เก็บไว้เดิม แล้วทำย้อนกลับซึ่งตรงนี้จำเป็นมากในการวิเคราะห์หาตาเดิน ซึ่งจะได้กล่าวต่อไป

เมื่อจบขั้นตอนนี้โปรแกรมก็พัฒนามาได้อีกระดับหนึ่งซึ่งรู้กติกาการเดินได้อย่างถูกต้อง เดินหน้าเป็น ถอยหลังได้ สามารถใช้แทนกระดานหมากฮอสได้ แต่ยังไม่สามารถคิดคำนวณหาตาเดินที่ดีด้วยตัวเองได้ ซึ่งเป็นสิ่งสำคัญที่ต้องการมากที่สุด

3.6 การ Search หาลำดับการเล่น

การ Search เป็นหัวใจของ ความคิดของคอมพิวเตอร์ หน้าที่เราก็คือทำให้ทุกอย่างเป็นคณิตศาสตร์ คอมพิวเตอร์นับเป็น บอกได้ว่าอะไรมากกว่า อะไรน้อยกว่า

3.6.1 เริ่มต้นให้คอมพิวเตอร์ นับตัวว่าใครมากกว่าจากนั้นเริ่มคิดโดยรับรู้ข้อมูลว่าบนกระดานมีตัวอะไรอยู่ที่ไหน เดินอย่างไร

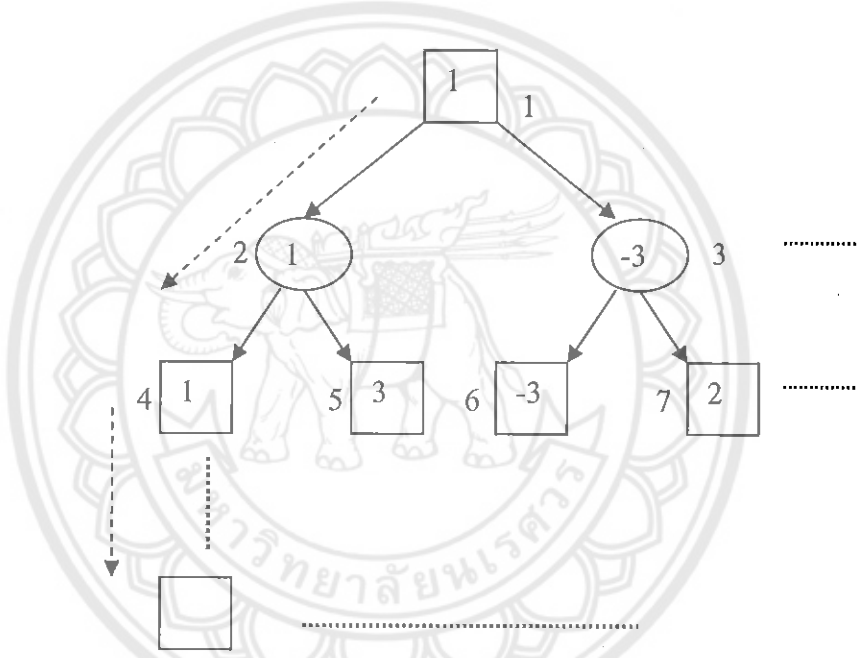
3.6.2 จากนั้นดูว่ามีตาเดินอะไรได้บ้าง แล้วก็ลองเดินดู ทีละตา เมื่อเดินตาหนึ่งแล้วก็บันทึกข้อมูล

3.6.3 การเปลี่ยนแปลงบนกระดานใหม่ จะพิจารณาตาเดินในระดับนั้นใหม่ลึกลงไป (depth first procedure) ทำอย่างนี้ลึกลงไปเรื่อย ๆ จนครบระดับที่ตั้งใจไว้จากนั้นก็นับคะแนนเทียบกับตั้งแต่เริ่มต้นว่าอะไร มากกว่าเลือกเอาอันที่มากที่สุดให้กับฝ่ายที่กำลังเดิน

3.6.4 การเดินย้อนกลับ (Unmakemove()) ทำซ้ำเรื่อย ๆ จนกว่าจะครบตาเดินทุกตาที่มีอยู่ การคำนวณอาจวิธีใช้ minimax หรือ negamax (minimax หมายถึง ถ้าเป็นตาที่เราเดินเราเลือกตาที่ได้คะแนน maximum คือสูงสุด ส่วนถ้าเป็นตาคู่ต่อสู้เดิน ก็เลือกตาเดินที่เราได้คะแนนต่ำสุด minimum ส่วน negamax หมายถึง คะแนนเรา = - คะแนนคู่ต่อสู้ ก็คล้าย ๆ กัน แต่ negamax จะประหยัดเวลาในการทำงานกว่าเล็กน้อยจึงเป็นที่นิยมกัน) ข้อสังเกต ใน program หมากฮอสจะต่างกับหมากรุกที่มีการกิน 2 ต่อ (double jump) เพราะฉะนั้นต้องดูด้วยว่าเป็นการกิน 2 ต่อหรือไม่ถ้าเป็นการกิน 2 ต่อ (double jump) score ที่ได้จากการ search ในระดับต่อไปคือ score ของฝ่ายนั้นเอง ไม่ใช่ score ของฝ่ายตรงข้าม

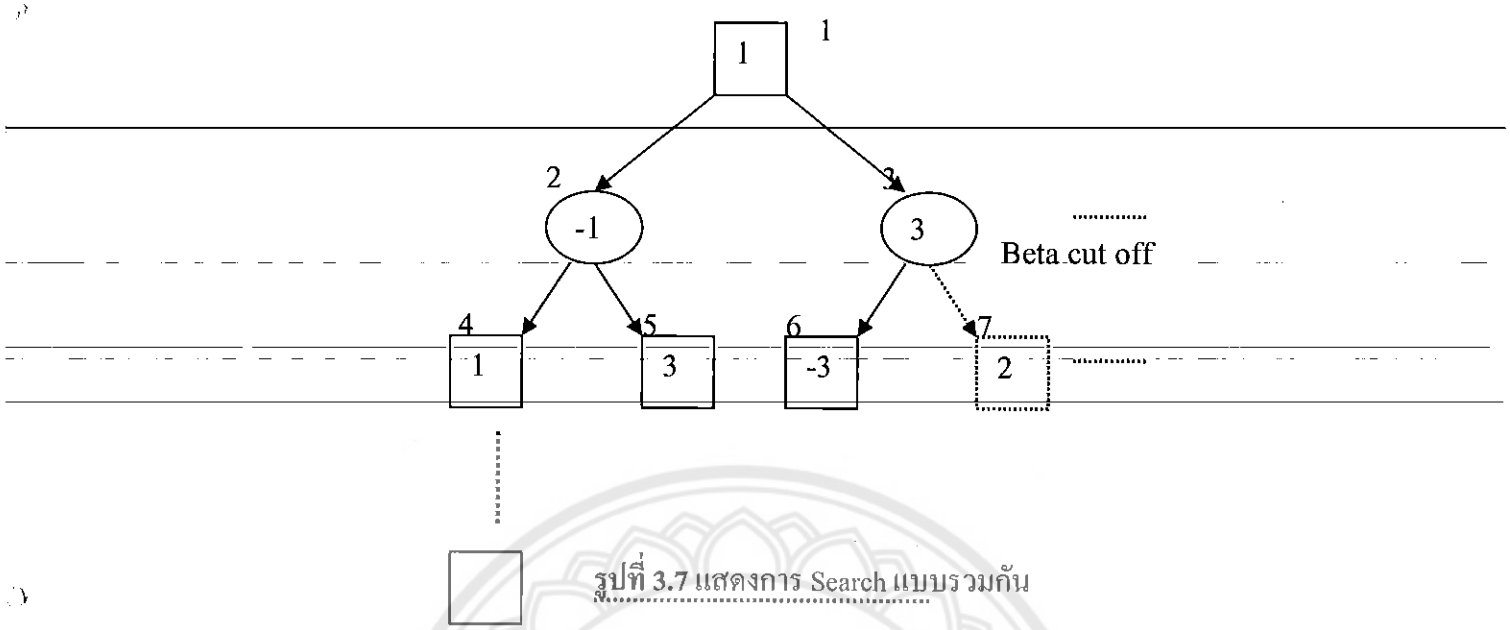
ตัวอย่างดังนี้ (สังเกตว่าเป็น Recursive negamax function) สำหรับ Quiescent Search() เมื่อ depth = 0 ก่อนที่จะ return evaluation() ในโปรแกรม chess ทั่วๆ ไป มักจะตรวจสอบว่าในตำแหน่งนั้นๆ หากมีการกินกันเกิดขึ้นฝ่ายใดจะเป็นฝ่ายได้เปรียบ ดังนั้นจึงเลือกที่จะ Generatemove เฉพาะที่จะมีการกินหรือ Gencratecapture เท่านั้น ต่างจากใน Search() ที่จะ loop ดูในทุกตาเดินที่เป็นไปได้ ในส่วนของโปรแกรมหมากฮอส เนื่องจากตาที่มีการกินเป็นการบังคับทั้งสิ้นเพราะฉะนั้นจึงไม่มีความแตกต่างกันระหว่าง Search() และ Quiescent Search()

3.6.5 การ Search หาลำดับการเล่น ลำดับแรกเมื่อผู้เล่นเลือกลำดับการเล่นนั้นจะแสดงว่าจะต้องใช้ Depth เท่าไร เมื่อทำการเลือกแล้ว จะได้ค่า Depth มาหาลำดับการเล่น ซึ่งขั้นแรกจะใช้การ Search แบบ Depth first search ซึ่งจะหาจากลำดับความลึก ดังตัวอย่าง



รูปที่ 3.6 แสดงการ search ของเกมแบบ Depth first search

จากรูป สมมติเลือกระดับการเล่นเป็นระดับง่าย ซึ่ง Depth = 2 จะทำการ Search ในระดับลึก 2 แถว จากนั้นจะทำการ Search แบบ Negamex ซึ่ง Negamax คล้ายกับ Minimax ต่างกันตรงที่ Negamex จะนำ -1 มาคูณ และหาค่ามากที่สุด จากตัวอย่างข้างบน เมื่อได้ระดับ Depth ที่ต้องการแล้วก็จะใช้หลักการนี้ในการ Search ผสมกับ Alpha Beta cut off จึงเรียกรวมกันว่า Alpha Beta in Negamax Function ดังตัวอย่างต่อไปนี้



รูปที่ 3.7 แสดงการ Search แบบรวมกัน

จากรูปข้างต้น เมื่อกำหนด Depth เท่ากับ 2 นั้นแสดงว่าจะทำการ search แค่ 2 แถว ดังนั้น ก็พิจารณาแค่ 2 แถวนี้ เมื่อเราได้ค่า Value แต่ละ node ดังตัวอย่าง ก็ทำการเปรียบเทียบตามหลักของ Negamex จากตัวอย่าง node ที่ 4 กับ 5 นำ -1 มาคูณ จากนั้นเลือกตัวที่มากที่สุด ก็นำค่ามาใส่ที่ node ที่ 2 ซึ่งค่า Value ที่ได้ เท่ากับ -1 จากนั้น พิจารณาที่ node ที่ 3 เพื่อมาเปรียบเทียบกับ node ที่ 2 เมื่อได้ค่าจาก node ที่ 6 ครั้งแรกก็ทำให้ 3 มีค่าอย่างต่ำเท่ากับ 3 หรือมากกว่านั้น ทำให้ไม่มีความจำเป็นต้องหาค่าของ 7 ก็ได้ เนื่องจาก สมมติว่า ค่า node ที่ 7 มากกว่า -3 (คือตั้งแต่ -2 ขึ้นไป) ก็ไม่สามารถเปลี่ยนแปลงค่าของ node ที่ 3 ได้ เพราะต้องเลือกค่าที่สูงสุดของ Negative value ซึ่งเท่ากับ 3 ขณะเดียวกันหากค่าของ node ที่ 7 น้อยกว่า -3 (คือตั้งแต่ -4 ลงไป) จะทำให้ค่าของ node ที่ 3 มากขึ้นไปอีก แต่กรณีนี้ค่า Negative value มากกว่าค่า Negative value ของ node ที่2 ซึ่งเท่ากับ -1 หรือ beta ในขณะนั้น ก็ไม่สามารถเปลี่ยนแปลงค่าของ node ที่ 1 หรือ root node ได้อยู่ดี จึงไม่จำเป็นต้องหาค่าของ node ที่ 7 แต่อย่างใด เราก็จะได้หมากที่ต้องการเดิน ซึ่งเป็นตัวหมากที่ดีที่สุดที่จะสามารถเอาชนะคู่ต่อสู้ได้

3.7 Static Evaluation Function

ภายใต้เทคโนโลยีปัจจุบัน ในเมื่อเราไม่สามารถให้คอมพิวเตอร์คิดตั้งแต่ตาแรกจนถึงตาสุดท้ายได้ เมื่อความลึกถึงระดับที่ต้องการ จึงต้องมี Function นี้เพื่อประเมินอย่างคร่าวๆ ถึงสภาพการณ์ว่าฝ่ายใดเป็นต่อ หรือเป็นรองประมาณเท่าไร แล้วส่งค่ากลับขึ้นมาเป็นตัวเลขเพื่อการเปรียบเทียบกับตาเดินตาอื่นๆ โดยทั่วไปหลักการอย่างง่ายที่ควรมีเป็นองค์ประกอบใน Evaluation Function คือ

3.7.1 Material หรือจำนวนตัวใครมีตัวมากว่ากัน และน้ำหนักของฮอสต่อเบี้ย ซึ่งอาจจะอยู่ประมาณ 2-3 : 1 แต่ทั้งนี้ก็แล้วแต่องค์ประกอบอื่นด้วย อันนี้เป็นสิ่งที่เข้าใจง่ายที่สุด ฝ่ายตัวมากกว่าย่อมดีกว่าฝ่ายตัวน้อย

3.7.2 ตำแหน่งเบี้ยสูงกว่า มักจะดีกว่าตำแหน่งเบี้ยต่ำ โดยเฉพาะปลายกระดาน แต่ก็ไม่เสมอไป

3.7.3 ตำแหน่งเบี้ยแถวล่างสุด ที่เป็นตัวกันฝ่ายตรงข้ามเข้าฮอส อันนี้ก็มีความสำคัญในการป้องกันฝ่ายตรงข้ามเข้าฮอส

3.7.4 ตำแหน่งของเบี้ยที่อยู่ติดมุม และไม่สามารถเดินออกได้ ถือว่าเสียเปรียบ ตรงข้ามตำแหน่งเบี้ยซึ่งอยู่กลางกระดาน สามารถเลือกตาเดินได้มากกว่าถือว่าเป็นการได้เปรียบเช่นกัน

ทั้งนี้องค์ประกอบทั้งหมดต้องให้น้ำหนักในส่วนต่างๆ โดยขึ้นกับระยะของเกม เช่นในช่วงการเปิดหมากต้องให้น้ำหนักกับ เบี้ยแถวล่างค่อนข้างมาก แต่พอปลายกระดานซึ่งมักจะเหลือตัวน้อยๆ ก็อาจไม่ต้องให้ความสำคัญกับเบี้ยแถวล่างมากนัก ทั้งหมดนี้ต้องเขียนรายละเอียดในระดับที่ชัดเจนพอควร เพื่อให้โปรแกรมออกมาแล้วเดินได้เหมือนคนๆ หนึ่ง แต่หากใส่รายละเอียดใน Function นี้มากก็จะทำให้โปรแกรมทำงานช้าลง ก็ต้องหาจุดที่สมดุลที่สุดซึ่งก็คือต้องทดสอบกับผู้เล่นที่เป็นคนจริงๆ แล้วนำข้อผิดพลาดมาแก้ไขจึงจะได้โปรแกรมที่เก่งสมดังใจ

บทที่ 4

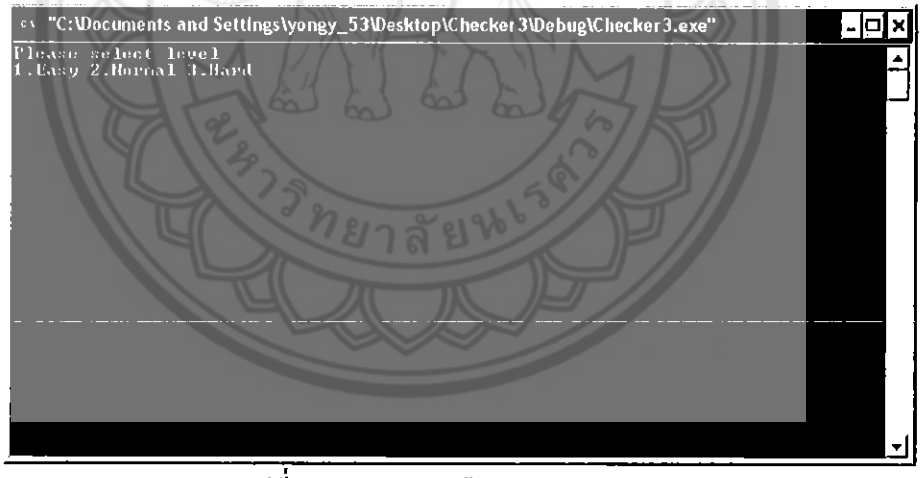
ผลการทดลอง

ในผลของการสร้างเกมหมากฮอส 3 มิตินั้น จะแสดงให้เห็นถึงรูปร่างหน้าตาขอตัวเกมว่าเป็นอย่างไร กติกาการเล่นเกมหมากฮอสและวิธีการเล่นเกมว่าใช้การบังคับอย่างไร รวมไปถึงแสดงผลจากการคิดของหลักการปัญญาประดิษฐ์(Artificial Intelligent) ว่ามันคิดอย่างไรถึงได้เดินตัวหมากตัวนั้น

4.1 ผลการรันโปรแกรมหมากฮอส

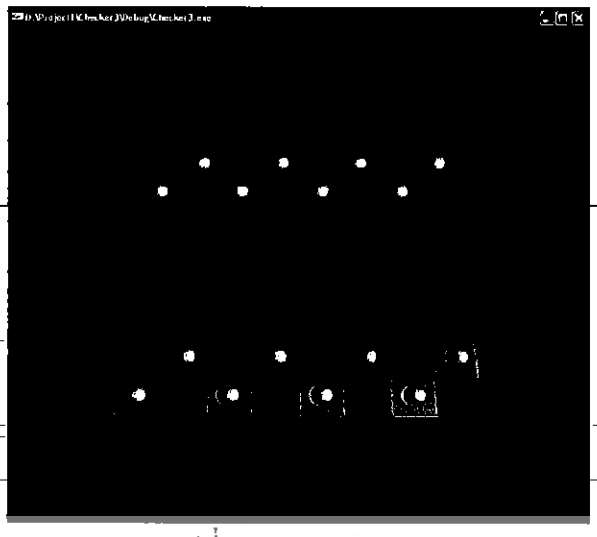
เมื่อรันโปรแกรมจะได้ผลลัพธ์ดังรูป เริ่มต้นจะเป็นการถามว่าจะเล่นในระดับความยากแค่ไหน ซึ่งจะแบ่งเป็น 3 ระดับ ดังนี้

1. Easy ระดับง่าย คือ การ Search ที่มีค่า Depth เท่ากับ 2
2. Normal ระดับกลาง คือ การ Search ที่มีค่า Depth เท่ากับ 4
3. Hard ระดับยาก คือ การ Search ที่มีค่า Depth เท่ากับ 8



รูปที่ 4.1 แสดงการเลือกระดับการเล่น

เมื่อปรากฏดังรูปข้างต้น เป็นการถามความต้องการของผู้เล่นว่าจะเลือกเล่นระดับไหน ผู้เล่นสามารถเลือกระดับได้ โดยกดเป็นตัวเลขตามระดับที่ปรากฏจากนั้นกด Enter จะแสดงหน้าต่างตารางเกมหมากฮอสพร้อมตัวหมากออกมา ดังรูป



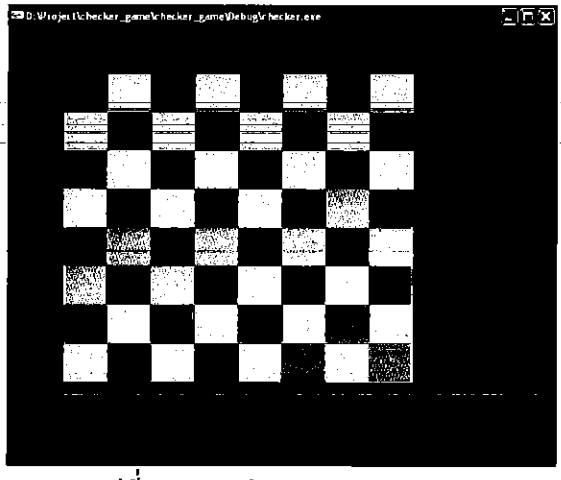
รูปที่ 4.2 เกมหมากฮอส

4.2 กติกาการเล่นเกมหมากฮอส

หมากฮอส เป็นกีฬาหมากกระดานประเภทหนึ่ง ประกอบด้วยผู้เล่น 2 ฝ่าย

อุปกรณ์การเล่น ได้แก่กระดานและตัวหมาก

กระดาน เป็นวัสดุพื้นเรียบ รูปทรงสี่เหลี่ยมจัตุรัส แบ่งแต่ละด้านออกเป็น 8 ช่องเท่ากัน จะได้ตารางย่อย 64 รูป เรียกสี่เหลี่ยมย่อยแต่ละรูปว่า ตา หรือ ช่อง กำหนดเป็นตาสีเข้ม 32 ตาและตาสีอ่อน 32 สลับกันโดยให้ตามุมซ้ายด้านบนของผู้เล่นเป็นตาสีอ่อน ตาที่ใช้เดินจะใช้เฉพาะตาสีเข้มรวม 32 ตา ในที่นี้จะกำหนดตำแหน่งของแต่ละตาด้วยเลขจำนวนนับ เพื่อให้ทราบว่ากล่าวถึงตาใดในกระดาน โดยเริ่มจากตาสีเข้มบนซ้ายสุดเป็น เลข 1 ตาสีเข้มถัดมาในแนวนอนเป็นตา 2 และในแนวนอนถัดไปเป็นตา 3 และ 4 ตามลำดับ จากนั้นในแถวต่อมาเริ่มตาสีเข้มซ้ายมือสุดเป็นตา 5 เรียงลำดับไปเรื่อย ๆ จนถึงแถวที่ 8 ตาสีเข้มขวามือล่างสุด จะเป็นตา 32



รูปที่ 4.3 ตัวอย่างกระดาน

ตัวหมาก เมื่อเริ่มต้น ทั้ง 2 ฝ่ายจะมีหมากฝ่ายละ 8 ตัว ในที่นี้กำหนดให้เป็นฝ่ายสีฟ้าเป็นฝ่ายของคอมพิวเตอร์ส่วนฝ่ายสีชมพูเป็นฝ่ายของผู้เล่น และตั้งกระดานโดยให้ตัวหมากของผู้เล่นอยู่บนสองแถวแรกที่ใกล้ตัวผู้เล่นฝ่ายนั้นแสดงดังรูป

รูปที่ 4.4 ตัวอย่างกระดานพร้อมตัวหมาก

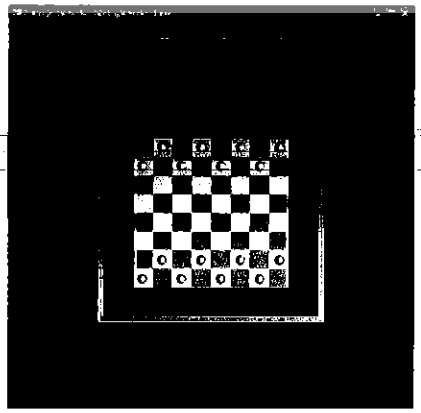
4.3 การดำเนินเกม

4.3.1 เมื่อรันโปรแกรมจะปรากฏให้ผู้เล่นเลือกระดับความสามารถของเกมหมากฮอสซึ่งมีอยู่ 3 ระดับ คือ ง่าย ปานกลาง และยาก

4.3.2 เมื่อเลือกเสร็จแล้วจะปรากฏกระดานหมากฮอสพร้อมตัวหมากที่ใช้เล่น

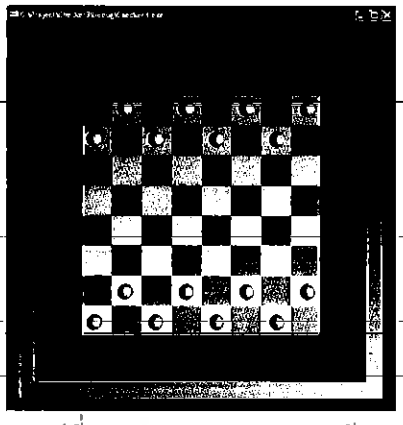
4.3.3 เนื่องจากโปรแกรมที่สร้างขึ้นเป็นแบบ 3 มิติ ดังนั้นสามารถเห็นส่วนลึก ส่วนกว้างได้ด้วยการ Zoom เข้า Zoom ออก เลื่อนขึ้นเลื่อนลง เช่น

1. เมื่อกด F1 เป็นการ Zoom ออก



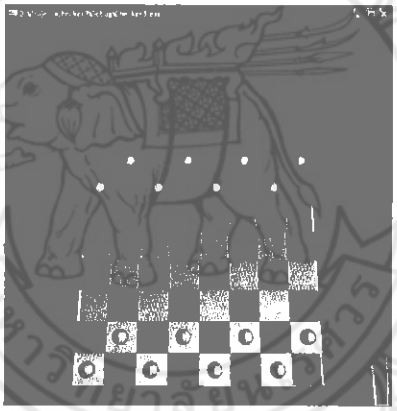
รูปที่ 4.5 แสดงการ Zoom ออก

2. เมื่อกด F2 เป็นการ Zoom เข้า



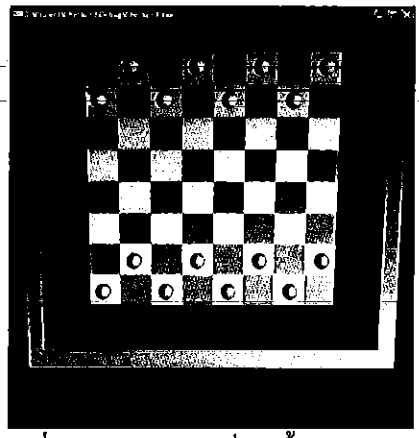
รูปที่ 4.6 แสดงการ Zoom เข้า

3. เมื่อกด F3 เป็นการเลื่อนลงแบบหมุน



รูปที่ 4.7 แสดงการเลื่อนลงแบบหมุน

4. เมื่อกด F4 เป็นการเลื่อนขึ้นแบบหมุน



รูปที่ 4.8 แสดงการเลื่อนขึ้นแบบหมุน

5. เวลา Zoom หรือ เลื่อน สามารถใช้ลูกศรช่วยในการเลื่อนก็ได้

4.3.3 ฝ่ายผู้เล่นจะเป็นสีชมพู ส่วนฝ่ายคอมจะเป็นสีฟ้า

4.3.4 ฝ่ายผู้เล่นจะเป็นผู้เริ่มเล่นก่อน และทั้ง 2 ฝ่ายสลับกันเดินจนกว่าการเล่นจะสิ้นสุด

4.3.5 การเดินหมาก

การเดินหมากแต่ละตัวบนกระดานหมากฮอส ใช้เมาส์คลิกที่ตัวหมากที่ต้องการจะเดิน จากนั้นต้องการจะวางที่ช่องในของตารางหมากฮอสก็ใช้เมาส์คลิกอีกหนึ่งครั้ง วิธีสังเกตว่าคลิกเมาส์โดนตัวหมากฮอสหรือเปล่าให้ดูที่หน้าต่างที่ปรากฏด้านข้าง ถ้าตรง Select ขึ้นมาว่า Select = 1 แสดงว่าคลิกเมาส์โดนตัวหมากฮอสที่เลือกพอวางเสร็จ Select จะเท่ากับ 0 เหมือนเดิม

1. ตัวหมากของแต่ละฝ่ายเมื่อเริ่มเล่นเรียกว่า เบี้ย

2. เบี้ยจะเดินได้ครั้งละ 1 ตา ในลักษณะเฉียงไปด้านหน้า เว้นแต่จะมีหมากตัวอื่นขวางทางหรือเมื่อไปชนริมกระดาน

3. หากเบี้ยเดินไปจนถึงแถวที่ 8 นับจากตัวผู้เล่น ตัวเบี้ยจะเปลี่ยนเป็นฮอส (เรียกว่า เข้าฮอส) แสดงให้เห็นว่าแตกต่างจากเบี้ย โดยใช้เบี้ยสีเดียวกันวางซ้อนเบี้ยตัวที่เข้าฮอสอีก 1 ตัว

4. ฮอสจะเดินกี่ตาก็ได้ในแนวเฉียงทั้งด้านหน้าและด้านหลัง

5. หากในการเดินของเบี้ยหรือฮอส มีหมากของฝ่ายตรงข้ามวางอยู่บนตาเดิน และมีตาว่างอยู่ตาถัดไปในแนวเดียวกัน เบี้ยหรือฮอสของฝ่ายที่เดิน สามารถเดินข้ามหมากฝ่ายตรงข้ามและยกหมากตัวนั้นออกจากกระดานไป และเรียกการเดินในลักษณะนี้ว่าการกิน

6. การกินสามารถทำได้อย่างต่อเนื่อง กล่าวคือ หากเมื่อกินหมากของคู่ต่อสู้แล้ว มีหมากตัวอื่นของอีกฝ่ายหนึ่งวางอยู่บนตากิน เบี้ยหรือฮอสนั้น สามารถกินต่อได้อีก เรียกว่าเป็นการกินหลายต่อโดยหมากตัวเดียว ในการเดินครั้งเดียว

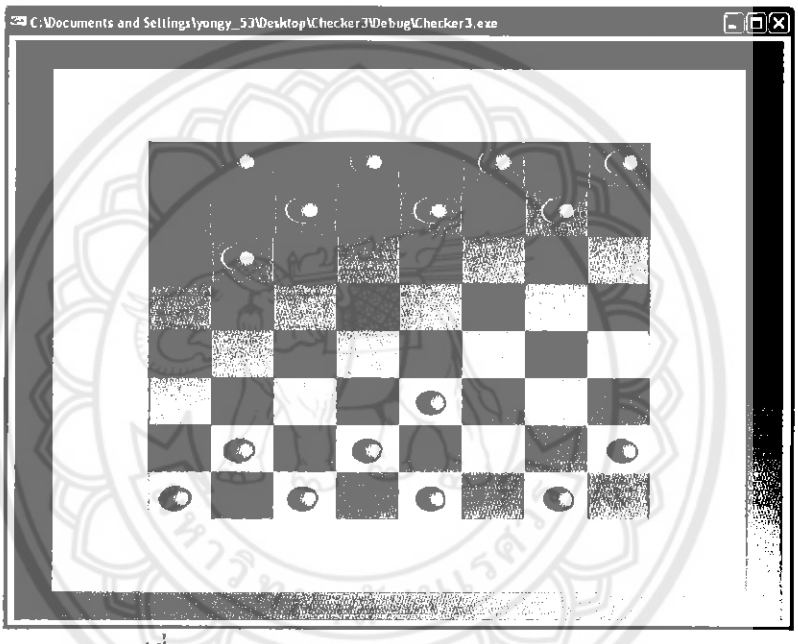
7. แม้ผู้เล่นจะมีอิสระในการเลือกเดินตัวหมากตัวใดก็ได้ แต่หากมีตัวหมากที่กินหมากของอีกฝ่ายหนึ่งได้ ผู้เล่นจะต้องเดินหมากเพื่อกินในการเดินครั้งนั้น (เว้นแต่จะมีตัวหมากที่กินคู่ต่อสู้ได้มากกว่า 1 ตัว ก็สามารถเลือกที่จะกินวิธีใดวิธีหนึ่ง) เรียกว่าไม่มีการหักขา

4.4 ผลจากการเดินของตัวหมากฮอสและวิธีการคิด

การ Search การเดินของตัวหมากฮอสนี้ จะต้องดูที่ระดับการเล่นด้วยว่าผู้เล่นเลือกเล่นเกมที่ระดับไหนซึ่งการเลือกระดับนี้ เป็นการบอกว่าจะทำการ Search ที่ ระดับความลึกของกระดานหมากฮอสที่เท่าไร ถ้าเลือกที่ระดับง่าย depth = 2 จะทำการ Search ที่ระดับความลึกแค่ 2 แถว ของแต่ละตำแหน่งของตัวหมาก ถ้าเลือกระดับกลาง depth = 4 เกมจะทำการ Search ที่ระดับความลึก 4 แถว เลือกระดับยาก depth = 8 เกมจะทำการ Search ทั้งหมด 8 แถว ซึ่งครอบคุมทั้งตาราง ระดับนี้จึงยากสำหรับผู้เล่นที่เลือกเล่นระดับนี้

4.4.1 การเดินหมากตัวแรก

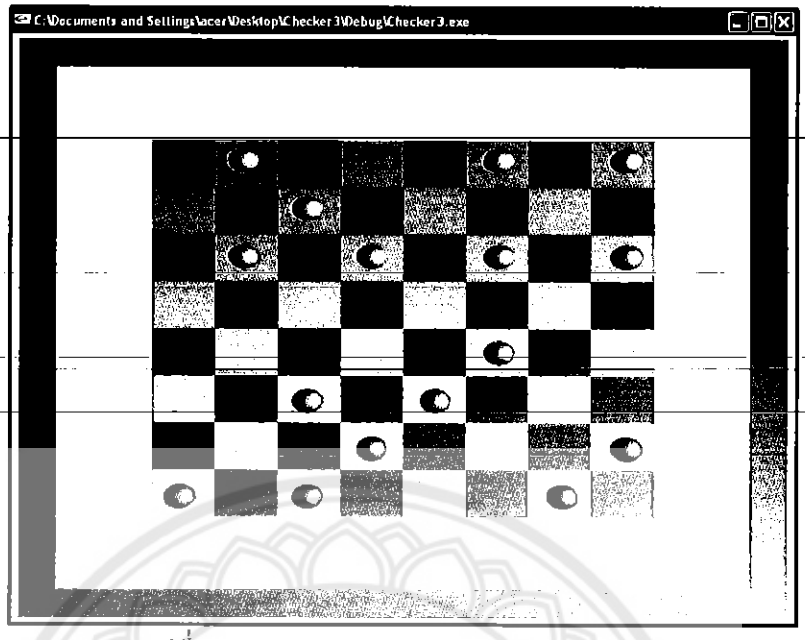
เริ่มต้นการเดินหมากตาแรกฝ่ายผู้เล่นจะทำการเริ่มเดินก่อนจากนั้นคอมพิวเตอร์ก็จะทำการ Search ตำแหน่งแรกที่ตัวหมากจะเริ่มเดิน โดยจะเริ่ม Search จากตำแหน่ง Array ตัวที่น้อยที่สุด ที่สามารถเดินได้ เป็นตัวแรก ซึ่งจากตารางหมากฮอสพบว่า ตำแหน่ง Array ที่น้อยที่สุดของการเปิดหมากที่สามารถเดินได้คือ Array[5] จานั้นก็ทำการ Search ตำแหน่งการเดินของ Array ตัวถัดไป คือ Array[6], Array[7]... ไปเรื่อย ๆ จนครบตาที่สามารถเดินได้ ซึ่งจากการ Search พบว่าค่า Value ของแต่ละตำแหน่งมีค่าเท่ากันหมดทุกตัว ดังนั้นสามารถเดินตัวไหนก็ได้เพื่อเป็นการเปิดตาราง จากตัวอย่างจึงเลือกที่ตำแหน่ง Array[4] ดังรูป



รูปที่ 4.9 แสดงการเดินหมากตัวแรกของแต่ละฝ่าย

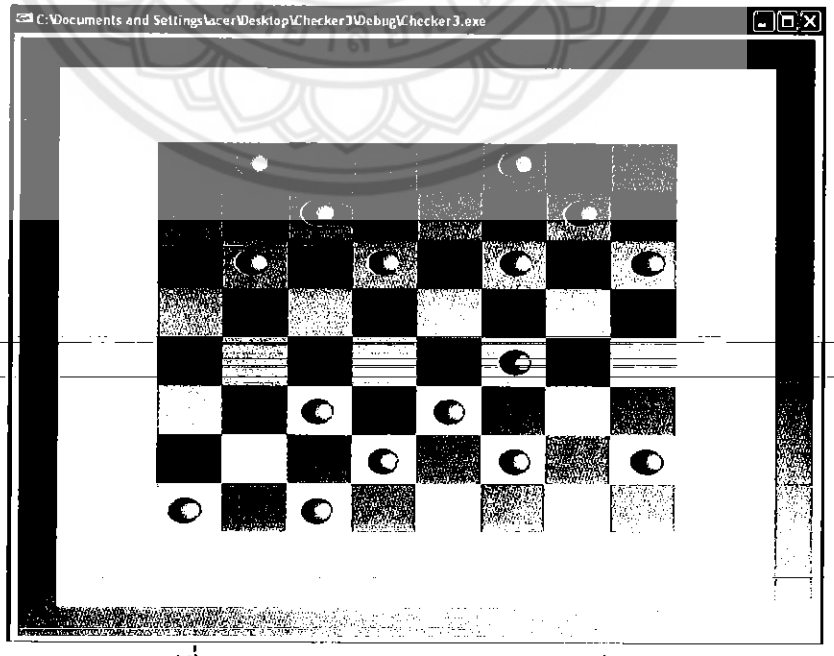
4.4.2 การเดินหมากกลางกระดาน

การ Search กลางกระดาน หมายถึงการเดินหมากไปได้ระยะหนึ่ง ซึ่งจะมีการกินกันเกิดขึ้น การ Search ระดับนี้ก็เหมือนกับการ Search แบบการเดินหมากของตัวแรก แต่การเดินหมากกลางกระดานนี้จะมีการกินกันเกิดขึ้น ดังนั้นจึงต้องดูด้วยว่าตาที่จะเดินไปนั้นโดนกินหรือไม่ ถ้าโดนกินค่าตรงตำแหน่งที่โดนกินจะออกมาเป็นลบ และในการ Search กลางกระดานนี้จะพิจารณาที่ฝ่ายตรงข้ามด้วยว่าเดินที่ตำแหน่งไหนบ้างแล้วจะมีการกินเกิดขึ้นกับเกมหรือเปล่า ทำอย่างนี้จนครบทุกตัว แล้วดูค่า Value ที่มากที่สุดว่าอยู่ตำแหน่งไหนก็เลือกเดินตานั้น เช่น



รูปที่ 4.10 แสดงเกมหมากฮอสกลางกระดาน

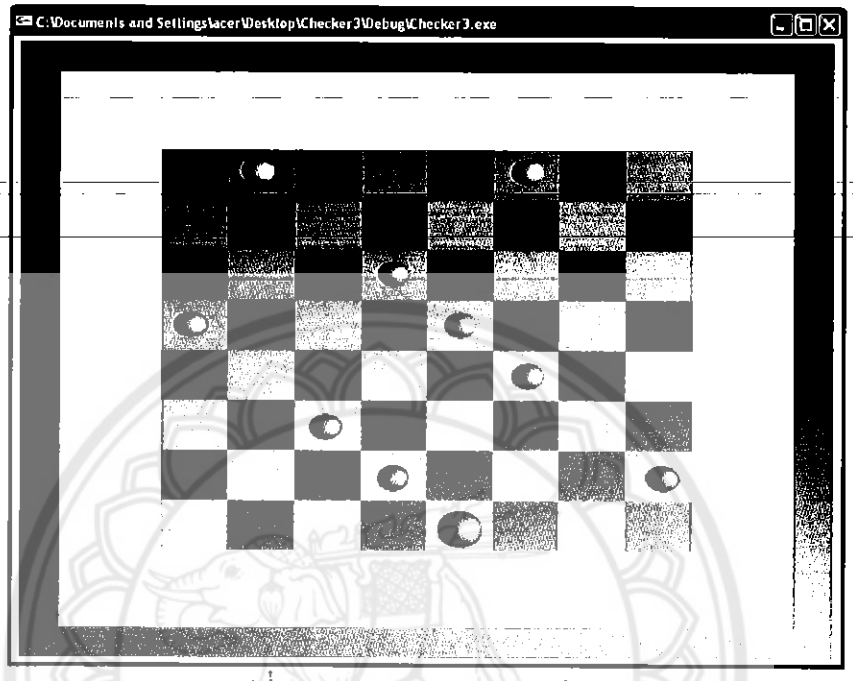
จากรูปข้างต้นจะเห็นว่า Array[11] มีค่า Value มากที่สุดแต่ทั้งนี้ทั้งนั้นก็ต้องดูที่การเดินของฝ่ายตรงข้ามด้วยว่าเดินตัวไหนและทำการ Search อีกครั้ง ซึ่งจากรูปข้างล่างจะพบว่าฝ่ายตรงข้ามได้เดินจากตำแหน่ง Array[32] เดินมาที่ตำแหน่ง Array[28] ทำให้ค่า Value ของตำแหน่ง Array[10] มีค่าน้อยลงซึ่งเท่ากับตำแหน่ง Array[4] ด้วย จึงเลือกเดินที่ตำแหน่ง Array[4] แสดงดังรูป



รูปที่ 4.11 แสดงการเดินหมากของตัวถัดไป

4.4.3 การกินเมื่อเข้าฮอส

เมื่อเข้าฮอสแล้วตัวหมากฮอสจะมีการเดินอีกแบบหนึ่งซึ่งสามารถเดินหน้าหรือย้อนกลับได้ซึ่งตรงนี้ต้องคิดด้วย ดังนั้นการเดินของตัวที่เป็นฮอสนี้จึงคิดค่าเป็น 2 เท่าของตัวธรรมดา



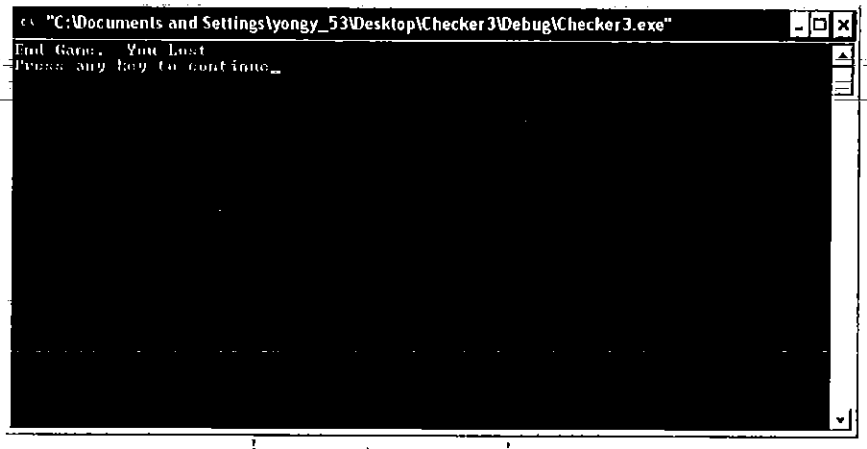
รูปที่ 4.12 แสดงตัวอย่างการเข้าฮอส

4.5 การตัดสิน

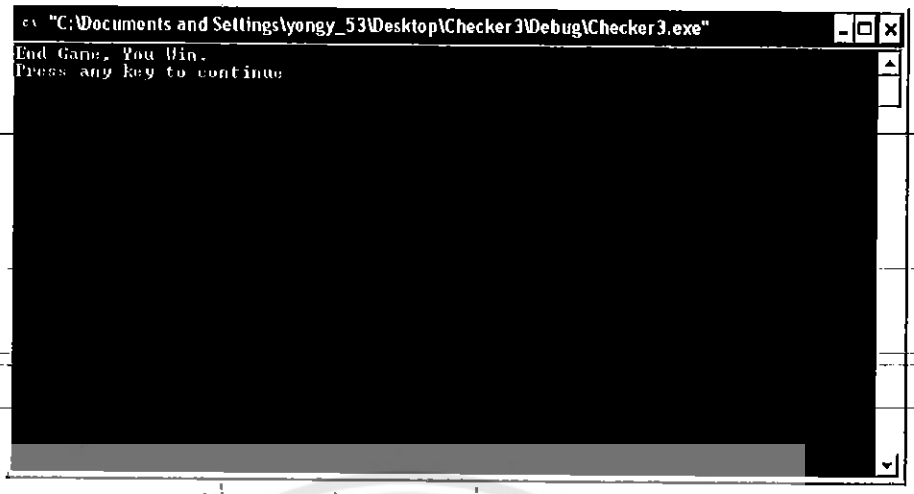
ผลของการเล่นแต่ละกระดาน จะสิ้นสุดลงโดยมีการแพ้-ชนะ ดังนี้

4.5.1 ฝ่ายที่สามารถกินหมากคู่ต่อสู้จนหมดจากกระดานเป็นฝ่ายชนะ

4.5.2 ฝ่ายที่ไม่สามารถเดินหมากตัวใดได้เลย แม้จะมีตัวหมากเหลืออยู่ เป็นฝ่ายแพ้



รูปที่ 4.13 หน้าต่างแสดงเมื่อผู้เล่นแพ้เวลาจบเกม



รูปที่ 4.14 หน้าต่างแสดงเมื่อผู้เล่นชนะเวลาจบเกม

บทที่ 5
สรุปผล

5.1 สรุปผลการทดลอง

โครงงานเกมหมากฮอส 3 มิติด้วยหลักการปัญญาประดิษฐ์ เป็นโครงงานที่จัดทำขึ้นโดยมีวัตถุประสงค์ที่จะสร้างเกมหมากฮอส 3 มิติ ที่มีภาพ 3 มิติ สวยงามและมีความฉลาดตามหลักการของปัญญาประดิษฐ์ โดยใช้หลักการของ Game Tree Search, Minimax Algorithm และ Alpha-Beta Algorithm ซึ่งเป็นหลักการทางปัญญาประดิษฐ์ (Artificial Intelligent) ช่วยในการเดินหมากของเกมหมากฮอส การพัฒนาโปรแกรมนี้ทดสอบและรันบนระบบปฏิบัติการไมโครซอฟท์วินโดวส์ XP ใช้ภาษาวิชวลซี พลัสพลัสเวอร์ชั่น 6.0 (Visual C++ 6.0) และ OpenGL ซึ่งเป็น Open source

ผลที่คาดว่าจะได้รับคือ ตัวเกมที่ได้ออกมาเป็นเกมหมากฮอส แบบ 3 มิติ ที่มีความสวยงามสะดวกในการเล่น มีความฉลาดในการเดินหมากแต่ละตาและมีความสามารถอย่างสูงในการเดินเพื่อที่จะเอาชนะได้อย่างมีประสิทธิภาพ

5.2 ปัญหาที่พบ

จากการพัฒนาโปรแกรมเกมหมากฮอส 3 มิติด้วยหลักการปัญญาประดิษฐ์ นั้นพบปัญหาดังนี้

1. ในการเริ่มต้นเดินหมากตัวแรกนั้น ฝั่งคู่ต่อสู้จะเดินหมากตัวแรกเป็นช่องเดิมทุกครั้ง
2. ในการใช้เมาส์เล่น ห้ามเปลี่ยนมุมมอง เพราะจะทำให้คลิกไม่โดนตัวหมาก

5.3 ข้อเสนอแนะ

จากการพัฒนาโปรแกรมเกมหมากฮอส 3 มิติด้วยหลักการปัญญาประดิษฐ์ นั้นมีข้อเสอนอแนะดังนี้

1. วิธีการทางปัญญาประดิษฐ์ที่นำมาใช้ในโครงงานนี้ ซึ่งคือ การนำ Minimax และ Alpha Beta มาใช้อาจจะไม่ใช่วิธีที่ดีที่สุด ดังนั้นควรทำการศึกษาค้นคว้าต่อไปเพื่อที่จะให้ตัวหมากสามารถเดินได้อย่างมีประสิทธิภาพมากที่สุด

2. การเล่นเกมนี้ยังผู้ใช้งานยังไม่มีความสะดวกในการเล่น ผู้เล่นจะต้อง ลง library ของ OpenGL ก่อน จึงจะสามารถเล่นได้

3. ตัวเกมยังมีความสวยงามไม่มากเท่าไรนักเราจึงควรศึกษาเพื่อนำไปประยุกต์หาโปรแกรมที่สามารถสร้างกราฟฟิกเพื่อให้เกมมีความสวยงามมากขึ้นต่อไป

4. ควรจะมีการเก็บสถิติการเล่นระหว่างคอมพิวเตอร์กับผู้เล่น และระหว่างคอมพิวเตอร์กับคอมพิวเตอร์ เพื่อทำการเปรียบเทียบว่าคอมพิวเตอร์แต่ระดับมีความสามารถตามที่กล่าวจริง

บรรณานุกรม

[1] ชุติมา ดังยางหวาย. "การแปลงภาพ 3 มิติเรขาคณิตด้วย OpenGL." [Online]. Available http://project.cs.kku.ac.th/2546/seminar/g24/

[2] ไม่ปรากฏชื่อผู้แต่ง, **"Computer Graphics - Chapter 10."** [Online]. Available : http://www.payap.ac.th/~geng/cs341/chapter10.pdf

[3] ไม่ปรากฏชื่อผู้แต่ง, **"Computer Graphics – Chapter 11."** [Online]. Available : http://www.payap.ac.th/~geng/cs341/chapter11.pdf

[4] Hearn Baker. **Computer Graphics with OpenGL**. Second Edition . America : Silicon Graphics, Inc. 1997.

[4] สมยศ เกียรติวนิชวิไล. **"Computer Graphic."** [Slide]. พิษณุโลก : มหาวิยาลัยนเรศวร. 2549

[5] ไม่ปรากฏชื่อผู้แต่ง, "กราฟฟิก." [Online]. Available : http://www.nectec.or.th/courseware/graphics/intro/0020.html

ประวัติผู้เขียนโครงงาน

ชื่อ นายยงเกียรติ อังประภาพรชัย

ภูมิลำเนา 511 หมู่ 9 ตำบลนครสวรรค์ตก อำเภอเมือง
จังหวัดนครสวรรค์ 60000

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนนครสวรรค์ จังหวัดนครสวรรค์
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

Email: **yongkiat_53@hotmail.com**

ชื่อ นางสาวสุปรียา เฮือนทา

ภูมิลำเนา 151 หมู่ 4 ตำบลตากตก อำเภอบ้านตาก
จังหวัดตาก 63120

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนบ้านตากประชาวิทยาคาร จังหวัดตาก
- ปัจจุบันกำลังศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4
  สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

Email: **supreeya_tuk@hotmail.com**