การสร้างเกมให้เรียนรู้ได้

Game that can learn

นาย กีรเดช ศรีเมือง

นาย กุลธวัช สุธรรมบุตร

นาย ณัฐวัฒน์ ประจวบศรีรัตน์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2549

การสร้างเกมให้เรียนรู้ได้

Game that can learn

โดย

นาย กีรเดช ศรีเมือง
นาย กุลธวัช สุธรรมบุตร
นาย ณัฐวัฒน์ ประจวบศรีรัตน์

อาจารย์ที่ปรึกษา
ผศ. เกียรติกูล เจียรนัยธนะกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2549

ปริญญานิพนธ์ปีการศึกษา 2549

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การสร้างเกมให้เรียนรู้ได้

Game That Can Learn

ผู้จัดทำ

1. นาย กีรเดช ศรีเมือง รหัสประจำตัว 46010053
2. นาย กุลธวัช สุธรรมบุตร รหัสประจำตัว 46010055
3. นาย ณัฐวัฒน์ ประจวบศรีรัตน์ รหัสประจำตัว 46010221

_____ อาจารย์ที่ปรึกษา

(ผศ. เกียรติกูล เจียรนัยธนะกิจ)

การสร้างเกมให้เรียนรู้ได้

นาย กีรเดช ศรีเมือง 46010053
นาย กุลธวัช สุธรรมบุตร 46010055
นาย ณัฐวัฒน์ ประจวบศรีรัตน์ 46010221
ผศ. เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2549

บทคัดย่อ

ในแวดวงของการพัฒนาซอฟต์แวร์นั้นผลงานที่ได้ใช่ว่าจะมีแต่เรื่องเคร่งเครียดและดูจริงจังนั่นเพราะด้านหนึ่งของการพัฒนาซอฟต์แวร์ที่มีเสน่ห์น่าหลงใหลอยู่ในตัวของมันเองก็คือซอฟต์แวร์ด้านความบันเทิงซึ่งเกมก็คือหนึ่งในนั้น

องค์ประกอบของการพัฒนาซอฟต์แวร์ด้านความบันเทิงนั้นขาดไม่ได้เลยคือ จินตนาการของผู้สร้างในการที่จะผูกโยงเรื่องราวต่างๆ ให้เร้าความสนใจ ขณะที่อีกด้านหนึ่งคือ ความสามารถทางด้านเทคนิควิศวกรรมซอฟต์แวร์ ซึ่งทั้งสองด้านจำเป็นที่จะต้องดำเนินควบคู่กันไปเพื่อถึงยังความสำเร็จที่ต้องการ

ในปัจจุบันเกมส่วนใหญ่ยังคงออกแบบมาให้ผู้เล่นสามารถเล่นเพียงคนเดียวได้ แต่รูปแบบเกมจะเป็นการแข่งขันหรือเป็นการร่วมมือกันระหว่างผู้เล่นจึงต้องให้คอมพิวเตอร์เล่นแทนผู้เล่นคนอื่น ซึ่งการพัฒนาให้คอมพิวเตอร์สามารถเล่นแทนได้เหมือนมีมนุษย์เป็นผู้เล่นนั้นไม่ใช่เรื่องง่าย

จากการศึกษาค้นคว้า จึงเกิดวิธีการสร้างเอไอมากมายหลายแบบ ซึ่งบางแบบสามารถให้คำตอบได้ดีแต่ก็มีจุดอ่อนเรื่องค่าใช่จ่ายเช่นหน่วยความจำและเวลา อีกทั้งการให้คอมพิวเตอร์สวมบทบาทแทนมนุษย์แล้วมาเล่นเกมกับมนษย์ก็ไม่ใช่ว่าจะดีเสมอไปเนื่องจากในบางครั้งอาจถูกคาดเดาพฤติกรรมได้เพราะวิธีที่ดีที่สุดย่อมไม่หลากหลาย ซึ่งในความเป็นจริงแล้วคู่ต่อสู้ทางความคิดที่รับมือยากที่สุดก็คือมนุษย์นั่นเองเพราะฉะนั้นน่าจะเลือกใช้เอไอที่สามารถเลียนแบบมนุษย์ได้ใกล้เคียงมากที่สุด เพื่อให้ได้ผลลัพธ์เสมือนมีเพื่อนมาเล่นเกมกับเรา

Game That Can Learn

Keradate	SriMuang	46010053
Kultawat	Suthambutr	46010055
Nattawat	Prajuabsrirat	46010221
Kietkul	Jearanaitanakij	Advisor

Academic Year 2006

Abstract

In the community of software development, software product is not always stressful and troublesome. Because, there is another branch of software development intimating which is entertainment software and game develops is one of them.

The composite of entertainment software development cannot be without imagination of the creator who bind the story together and make it interesting. As well as the skill of software implementation which both must be achieving to acquire the goal.

Mostly, games in the present are design to play with only single player. But the game requires another person to interact with player and that make us require AI. To make computer be able to act as human this is not an easy task.

By resent studies an amount of AI technique was being developed. Some of them make a good performance to solving problem but suffer varieties of cost such as memory and time and using the computer as human is not always the best choice because that sometime they are predictable, hence the best choice not variant and that made the toughest opponent is still be human. So we choose selected the AI which identical the most to human to acquire the result that is like having friend playing with us.

กิตติกรรมประกาศ

ในการจัดทำปริญญานิพนธ์ฉบับนี้คณะผู้จัดทำขอกราบขอบพระคุณบิดามารดาที่ช่วยอบรมสั่งสอนเลี้ยงดูและส่งเสริมด้านการศึกษาหาความรู้ต่างๆ รวมถึงกำลังใจอันหาที่เปรียบมิได้จนกระทั่งช่วยให้คณะผู้จัดทำสำเร็จการศึกษาด้วยดี

ขอขอบพระคุณอาจารย์เกียรติกูล เจียรนัยธนะกิจ ที่คอยให้คำปรึกษาและคำแนะนำต่างๆ เกี่ยวกับปริญญานิพนธ์ฉบับนี้ ทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยดี ตลอดจนคณาจารย์ทุกท่านผู้ประสิทธิประสาทวิชาให้แก่คณะผู้จัดทำทุกท่าน

ขอบคุณเพื่อนๆ และพี่ๆ ที่คอยให้คำปรึกษาต่างๆ เกี่ยวกับปัญหาที่เกิดขึ้น และเสนอแนะความคิดเห็นดีๆ ที่ประกอบขึ้นมาเป็นปริญญานิพนธ์นี้สำเร็จขึ้นมา

นาย กีรเดช ศรีเมือง

นาย กุลธวัช สุธรรมบุตร

นาย ณัฐวัฒน์ ประจวบศรีรัตน์

สารบัญ

หน้าที่

บทคัดย่อภาษาไทย I

บทคัดย่อภาษาอังกฤษ II

กิตติกรรมประกาศ III

สารบัญ IV

สารบัญภาพ VII

บทที่ 1 บทนำ 1

1.1 ความสำคัญและที่มา 1

1.2 วัตถุประสงค์ของโครงงาน 2

1.3 ขอบเขตของโครงงาน 2

1.4 ประโยชน์ที่คาดว่าจะได้รับ 2

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง 3

2.1 ไดเร็คเอ็กซ์ 3

2.1.1 Immediate Mode และ Retained Mode 4

2.1.2 บทบาทของ COM 5

2.1.3 COM และไดเร็คเอ็กซ์ 5

2.1.4 สถาปัตยกรรมของไดเร็คเอ็กซ์ 6

2.1.4.1 Hardware Abstraction Layer 6

2.1.4.2 Hardware Emulation Layer 7

2.2 ทฤษฎีการสร้างภาพสามมิติทั่วไป 8

2.2.1 โครงสร้างของโมเดลแบบต่างๆ 8

2.2.2 ส่วนประกอบของโมเดล 3 มิติ 8

2.2.3 ลวดลายของพื้นผิว 9

2.2.4 แสง 10

2.2.5 กล้อง 11

2.2.6 อุปกรณ์ 11

2.2.7 Viewport 11

2.2.8 Rendering 12

สารบัญ(ต่อ)

หน้าที่

2.2.9 Biped 12

2.2.10 Animation 12

2.2.11 Three-Dimensional Mathematics 13

2.2.11.1 ระบบพิกัด 13

2.2.11.1.1 ระบบพิกัดหนึ่งมิติ 13

2.2.11.1.2 ระบบพิกัดสองมิติ 13

2.2.11.1.3 ระบบพิกัดสามมิติ 14

2.2.11.2 เวกเตอร์ 15

2.3 โครงข่ายใยประสาทเทียม 16

2.3.1 เปรียบเทียบระหว่างสมองมนุษย์กับโครงข่ายใยประสาทเทียม 16

2.3.2 พื้นฐานของนิวรอล 18

2.3.3 นิวรอลในยุคต่อมา 18

2.3.4 โครงสร้างของโครงข่ายใยประสาทเทียม 20

2.3.4.1 แบบฟีดฟอเวิร์ด 20

2.3.4.2 แบบย้อนกลับ 21

2.3.5 ชั้นของโครงข่าย 22

2.3.6 มัลติเลเยอร์นิวรอลเน็ตเวิร์ค 23

2.3.7 เพอเซปตรอน 24

2.3.8 ระบบการเรียนรู้ 28

2.3.9 ทรานเฟอร์ฟังก์ชั่น 30

2.3.10 แบคพรอพพาเกชั่นอัลกอริทึม 30

บทที่ 3 การออกแบบและพัฒนา 31

3.1 โครงสร้างโปรแกรม 31

3.2 การแสดงภาพ 34

3.3 การควบคุม 35

3.4 ปัญญาประดิษฐ์ 36

3.5 การคำนวณ 36

บทที่ 4 การทดลองและผลการทดลอง 39

สารบัญ(ต่อ)

หน้าที่

4.1 ชนิดตัวละคร 39

4.2 ข้อมูลผู้เล่น 39

4.3 การใช้เอไอ 39

4.4 ผลที่ได้ 40

บทที่ 5 บทสรุปและวิจารณ์ 41

5.1 ปัญหาที่พบในการพัฒนา 41

5.2 การพัฒนาต่อ 41

5.3 ข้อเสนอแนะ 41

5.4 สรุป 41

ภาคผนวก 42

บรรณานุกรม 44

เว็บไซต์อ้างอิง 44

สารบัญภาพ

หน้าที่

รูปที่ 2-1 สถาปัตยกรรมของไดเร็คเอ็กซ์ 6

รูปที่ 2-2 A number line 13

รูปที่ 2-3 Cartesian coordinate system 14

รูปที่ 2-4 Left-Hand System 14

รูปที่ 2-5 Right-Hand System 14

รูปที่ 2-6 Components of a neuron 17

รูปที่ 2-7 The synapse 17

รูปที่ 2-8 The neuron model 18

รูปที่ 2-9 A simple neuron 18

รูปที่ 2-10 An MCP neuron 19

รูปที่ 2-11 Activation function สามารถใช้ได้หลาย function 20

รูปที่ 2-12 activation function ต่างๆ 20

รูปที่ 2-13 An example of a simple feed forward network 21

รูปที่ 2-14 An example of a complicated network 21

รูปที่ 2-15 Flow of Information 23

รูปที่ 2-16 โครงสร้างของนิวรอนที่ออกแบบ 26

รูปที่ 3-1 class diagram 32

รูปที่ 3-2 ตัวอย่างการแสดงผลของเกมส์ 34

รูปที่ 3-3 แสดงกระบวนการเทียบพิกัดจาก 2 มิติไปสู่พิกัด 3 มิติ 37

รูปที่ 3-4 แสดงการคำนวณมุมในการวางภาพ 37

รูปที่ 3-5 แสดงการคำนวณมุมเพื่อเปลี่ยนภาพตามทิศทางที่ตัวละครหัน 38

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มา

เกมแนวแอ็คชั่นอาร์พีจี (Action Role Playing Game) เป็นเกมที่มีผู้ชื่นชอบอยู่ไม่น้อยทั่วโลก ด้วยลักษณะพิเศษของเกมที่เป็นการสมมติให้ผู้เล่นเป็นตัวละครหนึ่งในโลกที่มีความลึกลับเต็มไปด้วยปริศนาที่ยังไม่มีผู้ให้คำตอบได้ ผู้เล่นจะได้พบกับเหตุการณ์ต่างๆ ที่จะเป็นเงื่อนงำไปสู่การไขปริศนาค้นพบความจริงของสิ่งต่างๆ มีเรื่องราวของการต่อสู้, เวทย์มนต์, สงครามระหว่างอาณาจักร, สิ่งมีชีวิตต่างเผ่าพันธุ์, ความสามารถพิเศษ และเรื่องราวแปลกพิสดารอีกมากมาย

ด้วยลักษณะพิเศษดังกล่าว จึงทำให้ผู้เล่นมีความรู้สึกผ่อนคลายความเหนื่อยล้าจากโลกแห่งความจริง แต่ถึงแม้จะเป็นเช่นนั้นเกมแนวนี้ก็ยังมีการดัดแปลงเรื่องราวให้สอดคล้องกับเหตุการณ์จริงอยู่บ่อยครั้ง จึงยิ่งเพิ่มความสมจริงยิ่งขึ้น ประหนึ่งมีจุดเชื่อมต่อระหว่างโลกแห่งความจริงกับโลกแห่งความฝันอยู่ที่เกม ที่ที่ทุกคนสามารถเป็นได้ดั่งใจ

ในแวดวงของการพัฒนาซอฟต์แวร์นั้นผลงานที่ได้ใช่ว่าจะมีแต่เรื่องเคร่งเครียด และดูจริงจัง นั่นเพราะด้านหนึ่งของการพัฒนาซอฟต์แวร์ที่มีเสน่ห์น่าหลงใหลอยู่ในตัวของมันเองนั่นก็คือ ซอฟต์แวร์ด้านความบันเทิง ซึ่งเกมก็คือหนึ่งในนั้น

องค์ประกอบของการพัฒนาซอฟต์แวร์ด้านความบันเทิงนั้นขาดไม่ได้เลยคือ จินตนาการของผู้สร้างในอันที่จะผูกโยงเรื่องราวต่างๆ ให้เร้าความสนใจ ขณะที่อีกด้านหนึ่งคือ ความสามารถทางด้านเทคนิควิศวกรรมซอฟต์แวร์ ซึ่งทั้งสองด้านจำเป็นที่จะต้องเดินควบคู่กันไป เพื่อถึงยังความสำเร็จที่ต้องการ

ในด้านเทคนิคการพัฒนานั้นมีหลายด้านที่ต้องนำมาประกอบกัน ทั้งด้านการแสดงภาพที่สวยงามสมจริง หรือเสียงที่มีคุณภาพเหมือนเสียงตามธรรมชาติ อีกด้านหนึ่งที่ไม่ได้แสดงออกมาให้ผู้เล่นรับรู้โดยตรงแต่มีความสำคัญมากคือ การประมวลผลเพื่อตัดสินใจเรื่องต่างๆ ในเกม เช่น การให้คอมพิวเตอร์เล่นแทนผู้เล่นคนอื่นโดยใช้ปัญญาประดิษฐ์ (Artificial Intelligent) หรือ เอไอนั่นเอง

ในปัจจุบัน เกมส่วนใหญ่ยังคงออกแบบมาให้ผู้เล่นสามารถเล่นเพียงคนเดียวได้ แต่จะเป็นการแข่งขันหรือเป็นการร่วมมือกันระหว่างผู้เล่น จึงต้องให้คอมพิวเตอร์เล่นแทนในส่วนของผู้เล่นคนอื่น การพัฒนาให้คอมพิวเตอร์สามารถเล่นแทนได้เหมือนมีมนุษย์เป็นผู้เล่นไม่ใช่เรื่องง่าย

จากการศึกษาค้นคว้า จึงเกิดวิธีการสร้างเอไอมากมายหลายแบบ บางแบบสามารถให้คำตอบได้ดีไร้ที่ติแต่ก็มีจุดอ่อนเรื่องค่าใช้จ่ายเช่นหน่วยความจำและเวลา อีกทั้งการให้คอมพิวเตอร์สวมบทบาทแทนมนุษย์ แล้วมาเล่นเกมกับมนุษย์ ก็ไม่ใช่ว่าจะดีเสมอไป ในบางครั้งอาจถูกคาดเดา

ได้เพราะวิธีที่ดีที่สุดย่อมไม่หลากหลาย ซึ่งในความเป็นจริงแล้วคู่ต่อสู้ทางความคิดที่รับมือยากที่สุดคือมนุษย์นั่นเอง เพราะฉะนั้นจึงน่าจะเลือกใช้เอไอแบบที่สามารถเลียนแบบมนุษย์ได้ใกล้เคียงที่สุด เพื่อให้ได้ผลลัพธ์เสมือนมีเพื่อนมาเล่นเกมกับเรา

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาขั้นตอนการพัฒนาซอฟต์แวร์เกมสามมิติ โดยใช้ DirectX, 3DMax

1.2.2 ศึกษาการทำงานของไดเร็กเอ็กซ์ในการติดต่อกับอุปกรณ์ฮาร์ดแวร์

1.2.3 ออกแบบและพัฒนาเกมเอนจินต์ โดยใช้ไดเร็กเอ็กซ์ และ ภาษาซีพลัสพลัส

1.2.4 เพื่อนำทฤษฎีนิวรอลเน็ตเวิร์คมาใช้ในหารูปแบบและวิธีการที่เหมาะสม

1.2.5 เพื่อศึกษาการพัฒนาเอไอที่สามารถเลียนแบบมนุษย์ได้

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการพัฒนาเกมสามมิติที่ทำงานบนระบบวินโดวส์ โดยมีแนวเกมเป็นเกมแอ็คชั่นอาร์พีจี (Action Role Playing Game) โดยมีเอไอเป็นระบบโครงข่ายใยประสาทเทียมซึ่งเรียนรู้ด้วยตัวเองได้ เพื่อให้ตัวละครที่บังคับโดยคอมพิวเตอร์ในเกมสามารถพัฒนาการเล่นของตัวเองได้

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 ทักษะในการพัฒนาเกม โดยนำเสนอในรูปแบบ 3 มิติ เช่นการสร้างฉาก การวางโครงเรื่องของเกม การสร้างภาพ 3 มิติ

1.4.2 ความรู้ในด้านการประมวลผลภาพ 3 มิติ เช่นการหมุนภาพ การสร้างภาพเคลื่อนไหว การใช้คณิตศาสตร์เข้ามาช่วยจัดการกับภาพ 3 มิติเป็นต้น

1.4.3 การใช้ไดเร็กเอ็กซ์ ติดต่อกับอุปกรณ์มัลติมีเดีย

1.4.4 เทคนิคในการเขียนโปรแกรมประมวลผลภาพ 3 มิติ

1.4.5 การออกแบบโปรแกรมเชิงวัตถุ

1.4.6 การเขียนโปรแกรมภาษาซีพลัสพลัสเพื่อติดต่อกับไดเร็กเอ็กซ์

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

2.1 ไดเร็คเอ็กซ์

ไดเร็คเอ็กซ์ คือ กลุ่มเทคโนโลยีที่ออกแบบโดยไมโครซอฟต์เพื่อให้แอพพลิเคชั่นบนวินโดวส์สามารถใช้งานอุปกรณ์มัลติมีเดีย เช่น กราฟิก วิดีโอ ภาพเคลื่อนไหว หรือแม้แต่ระบบเสียงรอบทิศทางได้เต็มประสิทธิภาพ สำหรับวินโดวส์ 98 และ วินโดวส์ 2000 รวมไปถึง Internet Explorer จะมีไดเร็คเอ็กซ์เป็นส่วนประกอบมาเรียบร้อยแล้ว แต่อย่างไรก็ดี ยังสามารถติดตั้งไดเร็คเอ็กซ์ได้โดยอัตโนมัติ ด้วยการติดตั้งเกมหรือแอพพลิเคชันมัลติมีเดียส่วนใหญ่ที่ออกแบบมาให้ทำงานกับไดเร็คเอ็กซ์

ไดเร็คเอ็กซ์ช่วยให้ผู้พัฒนาโปรแกรมมีโอกาสใช้อุปกรณ์ทันสมัยใหม่ๆได้โดยอัตโนมัติ ไม่ต้องกังวลว่าโปรแกรมหรือเกมที่เขียนขึ้นมาจะใช้กับอุปกรณ์มัลติมีเดียตัวใหม่ๆได้หรือไม่ ดราบใดที่ยังใช้ไดเร็คเอ็กซ์อยู่หลังจากที่ติดตั้งไดเร็คเอ็กซ์จะมีไฟล์จำนวนหนึ่งเพิ่มขึ้นใน Windows/System จำนวนหนึ่ง ซึ่งขณะที่เล่นเกมไฟล์เหล่านี้จะถูกโหลดขึ้นมาลิงค์กับโปรแกรมเกม จากนั้นเกมก็จะสามารถเรียกใช้ความสามารถของไดเร็คเอ็กซ์ได้ ไฟล์ .dll จะคอยติดต่อกับไดร์เวอร์ของการ์ดจอ ซาวน์การ์ด โมเดม และส่วนประกอบอื่นๆของเครื่องอีกทีหนึ่ง สรุปได้ว่า โปรแกรมจะเรียกใช้ ฮาร์ดแวร์ได้ผ่านไดเร็คเอ็กซ์ซึ่งจะเรียกผ่านไดเวอร์ที่ติดตั้งไว้อีกทีหนึ่ง ซึ่งจะเห็นได้ว่าผู้เขียนโปรแกรมไม่จำเป็นต้องไปเรียกใช้ไดร์เวอร์หรือเรียกใช้ฮาร์ดแวร์โดยตรง เพียงแต่เรียกผ่านคำสั่งในไดเร็คเอ็กซ์เท่านั้นเองแล้วไดเร็คเอ็กซ์จะไปติดต่อกับฮาร์ดแวร์ให้อีกทีหนึ่ง ซึ่งตรงนี้คือจุดสำคัญที่ทำให้ไดเร็คเอ็กซ์ประสบความสำเร็จได้ เพราะหมดปัญหาเรื่องความเข้ากันได้ของฮาร์ดแวร์และยังทำให้ผู้เขียนโปรแกรมไปยุ่งกับพวกคำสั่งในระดับล่างๆ ด้วย

ไดเร็คเอ็กซ์แบ่งใหญ่ๆได้ดังนี้

- DirectGraphic จะทำหน้าที่ดูแลและจัดการแสดงผลภาพทั้ง 2 และ 3 มิติช่วยในการให้แสง การจัดวางมุมกล้อง และทำภาพเคลื่อนไหว ซึ่งจะทำการควบคุมอุปกรณ์แสดงผลและดึงความสามารถของฮาร์ดแวร์ได้เต็มประสิทธิภาพ ในนี้จะประกอบด้วยส่วนประกอบย่อยๆ เช่น Direct3D และDirectDraw
- DirectSound ทำหน้าที่คอยดูแลและจัดการเกี่ยวกับเสียง ไม่ว่าจะเป็นเสียงแบบ 3 มิติ หรือเสียงแบบเซอร์ราวด์ นอกจากนั้นยังช่วยในการทำเลียงเอฟเฟ็คท์ต่างๆอีกด้วย
- DirectInput ทำหน้าที่ดูแลและจัดการเกี่ยวกับอุปกรณ์อินพุตที่ใช้ควบคุมต่างๆ เช่น คีย์บอร์ด เมาส์ หรือจอยสติ๊ก เป็นต้น และยังสามารถทำงานกับอุปกรณ์ที่เป็นแบบป้อนกลับ (Force feedback)ได้อีกด้วย

- DirectPlay ทำหน้าที่คอยดูแลและจัดการเกี่ยวกับการติดต่อระหว่างคอมพิวเตอร์ผ่านระบบเครือข่ายไม่ว่าจะเป็นระบบแลนหรือผ่านโมเดม
- DirectShow ทำหน้าที่ดูแลและจัดการกับการแสดงผลภาพวีดีโอ ซึ่งสามารถเล่นไฟล์ประเภท .MPG และ .MP3 เป็นต้น

นอกจากนี้ยังสามารถแบ่ง API ของไดเร็คเอ็กซ์อีกแบบเป็นสองส่วนใหญ่ๆ คือ

- DirectX Foundation ประกอบด้วยส่วน DirectDraw , DirectSound , Direct3D Immediate Mode และ DirectInput โดยจะไปดูแล function Low-Level ต่างๆ
- DirectX Media ประกอบไปด้วยส่วน DirectShow , DirectAnimation , DirectPlay และ Direct3D Retained Mode ซึ่งเป็นบริการระดับสูง ที่จะไปเรียกไดเร็คเอ็กซ์ Foundation อีกทีหนึ่ง

สำหรับเกมทั่วไปจะเรียกใช้ได้ DirectX Foundation เป็นหลัก และเรียกใช้ DirectX Media เมื่อต้องการได้รับบริการบางอย่าง DirectAnimation เป็น API (Application Programming Interface) สำหรับใช้ในเวปซึ่ง Internet Explorer ของ Microsoft สนับสนุนสำหรับ DirectShow นั้นเป็น API สำหรับเรียกไฟล์ภาพเคลื่อนไหว (เช่น .avi) ได้อย่างสะดวก

2.1.1 อิมมีเดียดโหมด (Immediate Mode) และ รีเทรนโหมด (Retained Mode)

ความแตกต่างระหว่าง Direct3D Immediate Mode (IM) และ Retained Mode (RM) ต่างกันอย่างไร ถ้าดูจากว่า IM อยู่ใน Foundation และ RM อยู่ใน Media ก็พอจะบอกได้ว่า RM จะประกอบด้วยฟังก์ชั่นการทำงานระดับสูงกว่า IM ก็คือ Direct3D IM จะประกอบด้วย Low-Level Drawing Functions มีความสามารถในการวาด Low-Level 3D Objects เท่านั้น ส่วน RM จะวาดอ็อบเจ็กต์ที่ High-Level กว่าอย่างเช่น IM จะวาดได้แค่ 3 เหลี่ยมในขณะที่ RM จะมีความสามารถในการวาด Polygon ได้เลย แต่เกมทั่วๆไปจะใช้ Immediate Mode เนื่องจาก Retained Mode นั้น Microsoft ทำไว้ไม่ค่อยดีนัก ซึ่ง Immediate Mode ของ Direct3D ก็อยู่ในระดับเดียวกับ OpenGL คือเป็นการทำงานระดับล่างเหมือนกัน

ไฟล์ที่จำเป็นก็คือไฟล์ .lib และ .h ต่างๆ การที่จะเขียนโปรแกรมเรียกใช้ไดเร็คเอ็กซ์ได้นั้นก็จะต้องนำ .lib มาลิงค์กับโปรแกรมตอน Compile แล้วก็ include .h ไว้ใน Source File ด้วย เช่นถ้าจะใช้ DirectDraw ก็ต้องเพิ่ม ddraw.lib ลงใน Project ของและ #include ไว้

2.1.2 บทบาทของ COM

COM (Component Object Model) ได้กล่าวถึงรายละเอียดเกี่ยวกับการติดต่อสื่อสารกันระหว่างอ็อบเจ็กต์ โดยข้ามกันระหว่างโปรเซสและข้ามผ่านเน็ตเวิร์ค ระบบที่มีการเรียกใช้ COM ใช้งานจะถูกรับรองเป็นอ็อบเจ็กต์อินแตนซ์ (Object Instance)

ทุกๆ COM อ็อบเจ็กต์นั้นยังจะต้องถูกสืบทอดมาจาก COM อินเตอร์เฟสมาตรฐาน ซึ่งเรียกว่า IUnknonw โดย parent อินเตอร์เฟสจะมีอยู่ด้วยกัน 3 เมธอดคือ AddRef , Delete และ QueryInterface ซึ่ง 2 เมธอดแรกจะจัดการอ็อบเจ็กต์ด้วยการอ้างอิงถึงจำนวน โดยการนับจำนวนอินสแตนซ์ตั้งแต่เริ่มต้นส่วนเมธอดสุดท้ายจะจัดการเกี่ยวกับการสอบถามกับ COM อ็อบเจ็กต์ตัวอื่นๆ สำหรับอินเตอร์เฟสที่ต้องการและอินเตอร์เฟสที่มีอยู่ จากนั้นจะคืนค่าพอยเตอร์ที่ชี้ไปที่อินเตอร์เฟสนั้น

COM อ็อบเจ็กต์ที่ทำการร้องขอนั้นเรียกว่า COM ไคลเอนต์ และอ็อบเจ็กต์ที่ตอบสนองการร้องขอนั้นคือ COM เซิร์ฟเวอร์ เมื่อไคลเอนต์ได้รับการเชื่อมต่อแล้วไคลเอนต์นั้นสามารถที่จะเรียกเมธอดใดก็ได้ใน COM เซิร์ฟเวอร์นั้น โดยใช้การอ้างอิงของอินเตอร์เฟสพอตเอตอร์ ส่วนการเชื่อมต่ออื่นๆ จะออกไปเพื่อทำให้งานของอ็อบเจ็กต์ที่เรียกเพื่อเข้ามาใช้งานง่ายขึ้น ในการเชื่อมต่อนี้จะมีเมธอดที่ทำกานแสดงรายชื่อของเมธอดที่สนับสนุน เพื่อที่จะช่วยให้เมธอดสามารถเรียาใช้งานได้ ก่อนที่จะรวมเข้าไว้ด้วยกันด้วยเทคนิคนี้จะอนุญาตให้สามารถสร้างชนิดของไลบรารีใดๆ และแอพพลิเคชัน เช่น Vistual Basic ซึ่งมีข้อดีในการสร้างสมบัติของ COM โปรแกรมที่มีความทนทานต่อความผิดพลาด

2.1.3 COM และไดเร็คเอ็กซ์

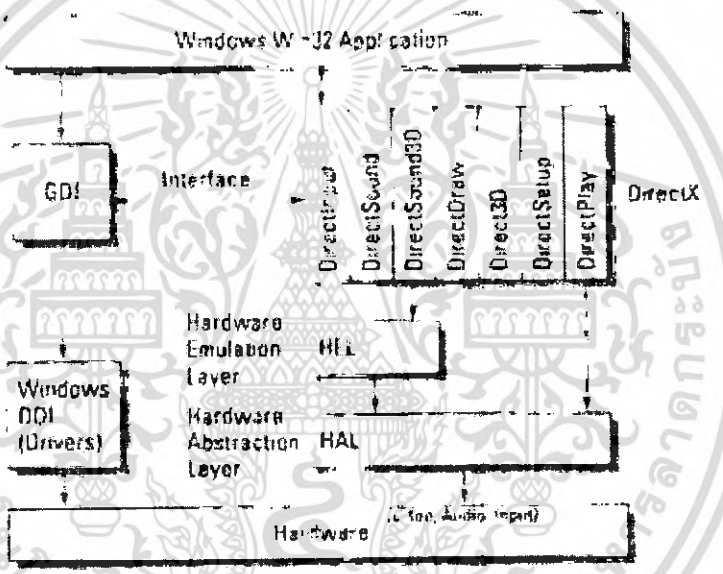
ทุกๆ อินเตอร์เฟสของไดเร็คเอ็กซ์นั้นจะทำการสืบทอดมาจาก IUnknown และมีพื้นฐานการพัฒนาจาก COM ด้วยเช่นกัน การใช้ COM ของไมโครซอฟท์ให้ได้ชุดคอมโพเน็นต์ของไดเร็คเอ็กซ์ที่มีประสิทธิภาพเพิ่มขึ้น ซึ่งสามารถเข้ากันได้กับไดเร็คเอ็กซ์รุ่นก่อนๆ ได้อย่างสมบูรณ์ ด้วยความอิสระที่ไม่ขึ้นกับภาษาที่ใช้เขียนโปรแกรมของ COM ทำให้นักพัฒนาสามารถที่จะตัดสินใจเลือกภาษาที่ใช้เขียนโปรแกรมตามที่เหมาะสมกับนักพัฒนาเองมากที่สุด ซึ่งอาจเป็นภาษาที่ถนัดและยังทำให้ได้ประสิทธิภาพตามที่ต้องการ นอกจากนั้นนักพัฒนายังสามารถที่จะเลือกภาษาอะไรก็ได้ตามที่ COM สนับสนุนและสามารถพัฒนาสภาพแวดล้อมได้ตามที่ต้องการ ด้วยโปรแกรมมิ่งโมเดลของ COM โปรแกรมเมอร์จะแน่ใจได้ว่าแม้ว่าผู้ใช้จะอัพเกรดหรือเปลี่ยนฮาร์ดแวร์ใหม่ หรือเปลี่ยนไดเร็คเอ็กซ์เวอร์ชั่นใหม่ก็ตาม ซอฟต์แวร์ก็ยังคงสามารถใช้งานได้ตามปกติ โดยไม่จำเป็นต้องทำการติดตั้งใหม่

ในความเป็นจริงไดรเวอร์ของอุปกรณ์บนวินโดว์สทุกตัวจะรวมกันเป็น WDM (Windows Driver Model) ซึ่งโครงสร้างภายในก็เป็น COM คอมโพเน็นต์ ดังนั้นจึงมีการโต้ตอบที่ดีระหว่าง

ไดรเวอร์โมเดลกับหน้าที่ของไดรเวอร์ ในสถาปัตยกรรมของไดเร็คเอ็กซ์นั้นทำให้แน่ใจได้ว่าการพัฒนาสำหรับนักพัฒนาเกมจะมีความสะดวกมากขึ้นและลดเวลาในการทดสอบลงด้วย

2.1.4 สถาปัตยกรรมของไดเร็คเอ็กซ์

ไดเร็คเอ็กซ์ได้สร้างขึ้นมาบนพื้นฐานของคอมโพเน็นต์ โดยใช้โครงสร้างของ HAL (Hardware Abstraction Layer) ซึ่งสามารถซ่อนลักษณะเฉพาะของดีไวซ์ (Device) ที่เกี่ยวพันอยู่กับฮาร์ดแวร์และเนื่องจากไดเร็คเอ็กซ์ได้รับการออกแบบให้มีความสามารถในการพัฒนาได้เพิ่มมากขึ้นในอนาคต ดังนั้นจึงสามารถรับรองความสามารถของฮาร์ดแวร์เร่งความเร็วชนิดใหม่ที่เข้ามาโดยใช้การทำงานผ่าน HEL (Hardware Emulation Layer)



รูปที่ 2-1 สถาปัตยกรรมของไดเร็คเอ็กซ์

2.1.4.1 HAL (Hardware Abstraction Layer)

HAL เป็นส่วนที่อยู่ล่างสุดของไดเร็คเอ็กซ์ซึ่งประกอบไปด้วยส่วนควบคุมของฮาร์ดแวร์ที่ถูกทำโดยผู้ผลิตฮาร์ดแวร์ ที่จะควบคุมฮาร์ดแวร์โดยตรง โดยชั้นนี้จะให้ประสิทธิภาพอย่างมากเพราะว่าสามารถที่จะติดต่อกับฮาร์ดแวร์ได้โดยตรง ในการทำงานจริงจะไม่สามารถติดต่อกับ HAL ได้เอง แต่ไดเร็คเอ็กซ์จะทำการจัดการให้โดยอัตโนมัติ

2.1.4.2 HEL (Hardware Emulation Layer)

HEL จะอยู่บนชั้น HAL โดยทั่วไปไดเร็คเอ็กซ์จะถูกออกแบบให้สามารถใช้ข้อดีของฮาร์ดแวร์ต่างๆได้ แต่ถึงอย่างไรไดเร็คเอ็กซ์ก็ไม่สามารถที่จะทำงานได้กับอุปกรณ์ทุกชนิด ตัวอย่างเช่น การเขียนโค้ดแสดงผลกราฟิก สมมติว่าฮาร์ดแวร์ที่กำลังทำงานอยู่สนับสนุนการหมุน (Rotate) และการปรับขนาด (Scale) ภาพบิดแมพ ซึ่งการการเรียกใช้งานไดเร็คเอ็กซ์เพื่อที่จะปรับขนาดและหมุนภาพบิตแมพได้นั้น จะต้องมีฮาร์ดแวร์ที่สนับสนุนการปรับขนาดและหมุนภาพ ซึ่งถ้าฮาร์ดแวร์สนับสนุนการทำงาน ก็จะทำงานได้เต็มความสามารถ และใช้งานฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ แต่ถ้าฮาร์ดแวร์ไม่สนับสนุนการปรับขนาดและหมุนภาพ การทำงานของ HEL จะเข้ามาทำงานแทนที่โดย HEL จะเข้ามาทำงานแทนที่ โดย HEL จะทำการจำลองหน้าที่การทำงานของ HAL ด้วยอัลกอริทึมทางซอฟแวร์ โดยจะไม่ทราบถึงความแตกต่างเลย แต่ถึงอย่างไรก็ตามโค้ดที่ได้ทำงานนั้นก็จะทำงานได้ช้าลง เพราะว่าเป็นการจำลองการทำงาน จึงไม่สามารถที่จะทำงานได้เหมือนอย่างฮาร์ดแวร์ร้อยเปอร์เซ็นต์

ไดเร็คเอ็กซ์ไดรเวอร์นี้จะรวมเข้าด้วยกันกับไดเร็คเอ็กซ์ API ผ่านลำดับของบัฟเฟอร์อ็อบเจ็กต์ผลลัพธ์ส่วนมากจะถูกสร้างโดยคอมโพเน็นต์ ไดเร็คเอ็กซ์ API จะถูกเขียนขึ้นเพื่อตอบสนองบัฟเฟอร์อ็อบเจ็กต์ โดยบัฟเฟอร์นี้จะถูกจัดการ และถูกทำให้มีประสิทธิภาพขึ้นตามฮาร์ดแวร์ที่มีอยู่ โดย 2 เลเยอร์ของสถาปัตยกรรมของไดเร็คเอ็กซ์และแปลงไปยังเอาท์พุตดีไวซ์ที่เหมาะสมอย่างเป็นระเบียบ ในระดับของภาษาที่เครื่องคอมพิวเตอร์สามารถเข้าใจได้

สมบัติของการจำลองการทำงานของฮาร์ดแวร์ ก็คือชุดของฟังก์ชันที่ทำการจำลองคำสั่งทาง 3 มิติโดยใช้ซอฟแวร์ ถ้าฟังก์ชันที่ต้องการโดยเกมนั้น ไม่สามารถที่จะเข้าถึงฮาร์ดแวร์ได้โดยผ่าน HAL ไดเร็คเอ็กซ์นั้นจะสลับการทำงานมายัง HEL เพื่อที่จะจำลองการทำงานโดยใช้ซอฟแวร์ ด้วยสมบัตินี้จะช่วยให้นักพัฒนาสามารถที่จะทดสอบหรือตรวจสอบสมบัติทาง 3 มิติในโหมดของซอฟแวร์ได้ด้วยตนเอง

สิ่งเหล่านี้จะทำให้ไดเร็คเอ็กซ์มีความยืดหยุ่นในการใช้งานอย่างมาก ซึ่งจะสนับสนุนการติดตั้งใช้งานฮาร์ดแวร์ใดๆ ไม่ว่าจะเป็นรุ่นเก่าหรือใหม่ ได้อย่างหลากหลายเช่น ถ้ามีแอพพลิเคชันที่ต้องการสมบัติใหม่ๆ ที่อยู่ในฮาร์ดแวร์รุ่นใหม่ ก็ยังจะสามารถที่จะทำงานในฮาร์ดแวร์รุ่นเก่าได้โดยผ่านชั้น HEL ของไดเร็คเอ็กซ์เพื่อที่จะจำลองความสามารถใหม่ที่ต้องการ ดังนั้น ถ้ายิ่งถ้ายิ่งติดตั้งไดเร็คเอ็กซ์รุ่นใหม่เท่าใด จะช่วยในการทำงานของฮาร์ดแวร์ในคอมพิวเตอร์ ดังนั้นเกมที่ถูกเขียนมาในรูปแบบหนึ่ง จะได้รับข้อดีของสมบัติใหม่ของฮาร์ดแวร์ที่ติดตั้งอยู่บนเครื่องโดยการรวมเข้ากันกับไดเร็คเอ็กซ์ ตัวอย่างเช่น ถ้ามีไดเร็คเอ็กซ์ที่เวอร์ชันใหม่กว่า ที่ออกมาให้สามารถทำงานกับคอมโพเน็นต์ได้เช่างมีประสิทธิภาพโดยผ่าน MMX ดังนั้นซอฟต์แวร์ทั้งหมดที่ถูกเขียนขึ้นมาก่อนที่ MMX จะออกมาก็จะสามารถใช้งาน MMX ได้แม้ว่าแนวความคิดของ MMX ในสมัยนั้นจะยังไม่เกิดขึ้นในตอนที่ซอฟเวร์นั้นถูกเขียนขึ้น ในทางกลับกันซอฟต์แวร์ที่ถูกเขียนขึ้น ในตอน

หลังที่มี MMX แล้วก็ยังสามารถทำงานได้ในโปรเซสเซอร์รุ่นเก่าก่อนที่มี MMX ได้โดย HEL ไดรเวอร์นั้นจะสามารถจำลองหน้าที่การทำงานของ MMX ให้มากที่สุดเท่าทีจะเป็นไปได้ โดยผ่านซอฟต์แวร์โดยอนุญาตให้ผู้ใช้ได้ใช้ความสามารถนั้นด้วย

ดังนั้นด้วยเหตุผลหลักของข้อดีเหล่านี้ เป็นเพราะว่าการออกแบบสถาปัตยกรรมให้เป็น COM ซึ่งสามารถช่วยให้คอมโพเน็นต์อ็อบเจ็กต์สามารถติดต่อสื่อสารกันได้

2.2 ทฤษฎีการสร้างภาพสามมิติทั่วไป

2.2.1 โครงสร้างของโมเดลแบบต่างๆ

- โครงสร้างแบบ Mesh และ Poly เป็นโครงสร้างที่ประกอบขึ้นจากจุดและเส้นตรงที่เชื่อมต่อกันจนเกิดเป็นแผ่นระนาบขึ้นมา และเมื่อนำแผ่นระนาบมาต่อกันเป็นรูปทรงต่างๆก็จะเกิดเป็นชิ้นงาน 3 มิติ โครงสร้างแบบนี้ถือเป็นโครงสร้างที่นิยมใช้มากที่สุดในปัจจุบันเพราะความง่ายในการขึ้นโมเดลเป็นรูปทรงต่างๆและในปัจจุบันโครงสร้างแบบ Mesh และ Poly ยังถูกพัฒนาไปจนถึงขั้นที่สามารถขึ้นโมเดลที่เป็นรูปแบบฟรีฟอร์มได้ โครงสร้างแบบนี้เหมาะกับการนำไปสร้างโมเดลสำหรับการทำเกม 3 มิติเนื่องจากเกม 3 มิตินั้นต้องการการประมวลผลภาพที่เร็ว โมเดลที่มีขนาดของ Poly น้อยก็จะทำให้ประมวลผลภาพได้รวดเร็วขึ้น

- โครงสร้างแบบ Spline เป็นโครงสร้างที่มีส่วนประกอบเช่นเดียวกับ Mesh และ Poly แต่แตกต่างกันที่ลักษณะของเส้นที่สามารถดัดโค้งได้เป็นอิสระก้อนที่จะสร้างพื้นผิว ซึ่งเหมาะสำหรับการสร้างรูปทรงแบบฟรีฟอร์ม

- โครงสร้างแบบ NURBS คล้ายกับโครงสร้างแบบ Spline แต่มีความยืดหยุ่นมากกว่าเหมาะสำหรับการสร้างรูปทรงแบบฟรีฟอร์ม

2.2.2 ส่วนประกอบของโมเดล 3 มิติ

- Vertex คือ จุด (point) ซึ่งวางเรียงกันเป็นรูปทรงต่างๆ

- Vertices คือ เซตของจุดที่ใช้แทนตำแหน่งของวัตถุ เช่น Face หรือ Mesh ใน 3D Space โดยที่แต่ละ Vertex อาจจะแทนด้วยค่า 3 ค่าซึ่งมีลักษณะเฉพาะของจุดหรือ coordinate ในรูปทรง 3 มิติ

- Edge คือ เส้นที่เชื่อมต่อระหว่าง Vertex หนึ่งไปยังอีก Vertex หนึ่ง

- Polygon คือระนาบที่เกิดจากการนำ Edge มาวางเรียงต่อกัน โดยที่ 1 Polygon จะต้องมีอย่างน้อย 3 Edge การทำภาพที่มีลักษณะโค้งจะใช้หลายๆ Polygon มาเรียงต่อกันให้ดูโค้งแทนการทำเป็นเส้นโค้งจริงเพื่อประหยัดเวลาในการคำนวณ ภาพ 3 มิติที่มีลักษณะโค้งที่มีจำนวนของ Polygon มากจะทำให้ภาพมีความโค้งมนสมจริงมากขึ้น แต่ก็ต้องใช้เวลาในการประมวลผลภาพมากยิ่งขึ้น

- Face คือ ส่วนประกอบที่อยู่ใน Mesh หรือ Poly ที่ถูกแบ่งครึ่งนั่นเอง หรือก็คือ Vertices ที่มีตั้งแต่ 3 จุดขึ้นไปมาเชื่อมต่อกันเป็นรูปทรงต่างๆในแนวระนาบ Vertex เป็นตัวกำหนดมุมของ Face ทำให้ทุกๆ Vertex ใน Face จะต้องถูกกำหนดให้อยู่ในแนวระนาบ

- Mesh เกิดจากการรวมกันของ Face ที่เชื่อมต่อกัน ซึ่ง 1 Mesh สามารถมี Face ได้ตั้งแต่ 1 Face ขึ้นไป การรวมของ Face นี้ทำให้ง่ายต่อการจัดการวัตถุในการทำ Animation, Material และ Texture ชนิดการรวมกันของ Face มีดังนี้คือ

- Fan คือ กลุ่มของรูปทรงสามเหลี่ยม ซึ่งทุกรูปมีการใช้ Vertex ร่วมกัน 1 จุด โดยการกำหนดให้ Vertex นั้นๆอยู่ระหว่างกลางของสามเหลี่ยมเหล่านั้นชนิดของการรวม Face แบบ Fan นี้คล้ายกับแบบ Strip คือมีการกำหนดค่า Vertices 3 ค่าแรกสำหรับสามเหลี่ยมแรก จากนั้นในสามเหลี่ยมต่อไปก็เพียงแค่เพิ่มขึ้นมารูปละ 1 Vertex เท่านั้น

- Strip คือ กลุ่มของสามเหลี่ยม ซึ่งแต่ละรูปจะมีการใช้เส้นร่วมกันกับสามเหลี่ยมส่วนหน้าซึ่งหมายความว่า หลังจากที่มีการกำหนดค่า Vertices 3 ค่าแรกสำหรับสามเหลี่ยมแรก จากนั้นในสามเหลี่นมต่อไปก็เพียงเพิ่มขึ้นมารูปละ 1 Vertex นั่นเอง

- List คือ กลุ่มของสามเหลี่ยม ซึ่งทุกรูปไม่มีการใช้เส้นหรือ Vertex ร่วมกันเลย กล่าวคือ ค่า 3 ค่าของสามเหลี่ยมทุกรูปต้องกำหนดเองทั้งหมด

2.2.3 Texture (ลวดลายของพื้นผิว)

- Shading คือการทำรายละเอียดของตัวพื้นผิว เช่น ความมันวาว การสะท้อนของพื้นผิว หรือความโปร่งแสง ทึบแสงของวัตถุ เป็นการให้สีเป็นลำดับชั้น

- Flat Shading Lighting เป็นการรายละเอียดพื้นผิวที่มีสีเสมอกันทั่วทั้ง polygon

- Vertex Shading หรือ Gouraud Shading เป็นการให้สีแก่ vertex แต่ละจุดตามสีที่ได้กำหนดเอาไว้แล้ว

- Phong Shading มีลักษณะคล้ายกับ Gouraud Shading แต่จะให้แสงที่นวลกว่า แต่ก็ใช้เวลาในการ render นานกว่าด้วย

- - Texture คือลวดลายของพื้นผิว โดยที่จะเป็น Bitmap ที่เป็น Pattern หรือ Image มักจะเก็บในรูปแบบไฟล์ BMP, PCX หรือ GIF เพื่อเป็นการใส่รายละเอียดให้แก่พื้นผิวของวัตถุ ทำให้วัตถุมีความสมจริง

- Texture Coordinate ใช้กำหนดการเชื่อมต่อกันระหว่าง Vertices ของ Face กับ Pixel ของ Bitmap โดย Texture Coordinate นี้ใช้แทน 2 มิติของ Coordinate System

- Texture Mapping คือ การวาดรูปลงบนพื้นผิวของ Face หรือ Polygon และในการทำ Texture Mapping นี้ต้องคำนึงถึงการคำนวณค่าต่างๆด้วย จึงต้องมีการกำหนดค่าของ Vertices

ด้วย จากการที่ Texture Coordinate กำหนด Pixel ของ Texture ที่จะวาดลงในส่วนของ Face แล้วก็จะมีการ Wrapping เพื่อ Generate Texture Coordinate สำหรับ Object นั้นซึ่งการ Wrapping นั้นมี 4 ชนิดคือ

- Flat Warp จะทำการวาดลงบน Face โดยตรง
- Cubical Warp จะทำการ Warp Texture ใน Cube รอบๆ Object เหมือนกับการแปะลวดลายบนวัตถุที่มีลักษณะเป็นกล่อง
- Cylindrical Warp จะทำการ Warp Texture ในทรงกระบอกรอบๆ Object คือจะนำลวดลายในการ Map รอบโมเดลในลักษณะการห่อเหมือนทรงกระบอก ซึ่งเหมาะกับวัตถุที่มีลักษะเป็นทรงกระบอกหรือเป็นแท่งๆ
- Spherical Warp จะทำการ Warp Texture ใน Sphere รอบๆ Object คือจะนำภาพมาแปะในลักษณะห่อรอบทรงกลม เหมาะกับการแปะลวดลายบนวัตถุทรงกลม
- Shrink Wrap คล้ายกับการ Map แบบ Spherical แต่โปรแกรมจะรวมจุดปลายของลวดลายที่ นำมา Map เข้าหากันเป็นจุดเดียว

2.2.4 Light (แสง)

แสงทำให้เกิดการเปลี่ยนแปลงของสีของ Vertices โดย Module ทำให้เกิด Vertex Normal เพราะสิ่งนี้ขึ้นอยู่กับมุมของแหล่งกำเนิดแสงตามปกติจะมีแสงสีขาว เพราะเป็น การรวมกันอย่างหนาแน่นของสีทุกสีและโดยมากมักใช้รูปแบบของ RGB ในการกำหนดสีของแหล่งกำเนิดแสง

ใน 3D นั้นมีการตั้งค่า RGB ของแม่สีต่างๆดังนี้

แสงสีขาว เป็น 1, 1, 1

แสงสีแดง เป็น 1, 0, 0

แสงสีน้ำเงิน เป็น 0, 0, 1

นอกจากนี้เรายังสามารถใช้ 3 สี นี้ในการผสมสีได้ ซึ่งแหล่งกำเนิดแสงมี 4 ชนิด ได้แก่

- Ambient Light คือแหล่งกำเนิดแสงที่ง่ายที่สุด เพราะไม่ต้องมีการกำหนดตำแหน่งของแหล่งกำเนิดแสง และยังให้ความสว่างทั่วทุก Object
- Point Light เป็นแหล่งกำเนิดแสงที่ทำการกระจายแสงไปทุกทิศทาง แต่ต้องระบุตำแหน่งของแหล่งกำเนิดแสง โดยไม่ต้องกำหนดทิศทางของแสง
- Directional Light เป็นแหล่งกำเนิดแสงที่มีประสิทธิภาพมากที่สุดเพราะ เป็นแหล่งกำเนิดแสงที่มีทิศทาง โดยต้องระบุตำแหน่งของแหล่งกำเนิดแสง
- Spot Light เป็นแหล่งกำเนิดแสงที่ต้องมีการระบุทั้งทิศทางและตำแหน่งของแหล่งกำเนิดแสง โดยการผลิดแสงจะเป็นรูปร่าง

2.2.5 กล้อง

Camera คือ มุมกล้องที่ทำให้เราสามารถมองเห็นวัตถุจากทิศทางต่างๆ เปรียบเสมือนว่าเรากำลังนำกล้องไปตั้งไว้บริเวณพื้นที่ที่มีวัตถุดังกล่าว ดังนั้นการเปลี่ยนมุมกล้องก็จะทำให้ภาพที่ปรากฏบนจอมีลักษณะแตกต่างกันไป การสร้างวัตถุเพียงชิ้นเดียวอาจจะไม่มีปัญหาอะไรในการมองรูปภาพ แต่ถ้าเราสร้างวัตถุหลายๆชิ้นหลายๆรูปทรงใน 1 ภาพ การลำดับตำแหน่งของวัตถุจึงเป็นสิ่งจำเป็นที่เราจะต้องฝึกให้คุ้นเคยและสามารถมองภาพให้เป็นมิติและสมจริงได้ ไม่ว่าภาพนั้นจะแสดงผลในรูปแบบใด ดังนั้นลำดับเป็นเรื่องที่เราต้องแยกแยะให้ออก หากว่าเราไม่สามารถที่จะมองภาพที่เราสร้างขึ้นให้เป็นมิติได้ ก็ยากที่จะรู้ได้ว่าภาพหรือวัตถุชิ้นไหนอยู่ข้างหน้าหรืออยู่ข้างหลัง และอาจทำให้เราเกิดความสับสนในเรื่องของการลำดับวัตถุ และทำให้การสร้างภาพหรือวัตถุมีความยุ่งยากซับซ้อนมากยิ่งขึ้น จนไม่สามารถสร้างเกมที่มีความสมบูรณ์และมีประสิทธิภาพได้

2.2.6 อุปกรณ์

Device คือ Object ที่สร้างขึ้นเพื่อใช้ในการวาดภาพจากมุมกล้องโดย Device มี 2 ชนิดหลักๆ คือ

- Software Device ซึ่งจะอนุญาตให้โปรแกรมสามารถวาดภาพโดยไม่ต้องนำไป คำนวณที่การ์ดจอแบบ 3D Accelerated

- Hardware Device สามารถทำงานบนคอมพิวเตอร์ที่มี 3D Hardware เท่านั้น และจะทำให้ Direct3D สามารถเรียกใช้คุณสมบัติต่างๆในการเรนเดอร์ภาพได้อย่างเต็มที่ ทำให้จำนวนเฟรม/วินาทีมีค่ามากขึ้น

2.2.7 Viewport

อยู่ในส่วนของ Camera โดย Viewport นั้นจะรวมไปถึงตำแหน่งและทิศทางของ Scene จากการมองเห็นซึ่งเรามักใช้ Viewport ร่วมกับ Field of View Front & Back Clipping และ Perspective Transformation

Viewport ประกอบด้วย

- Eyepoint คือ จุดที่ตั้งกล้อง
- Lookat คือ จุดที่กล้องมอง
- Upvector คือ Vector ด้านบนของกล้อง

การ View ดู Object มี 2 ชนิด

- Perspective เป็นวิธีที่ทำให้รูปที่วาดลงบน Screen มีลักษณะที่ใกล้เคียงกับ Object จริงๆมากที่สุดเพราะจะสนใจทั้งจุด x, y, z ทำให้เกิดความลึกของภาพนั่นเอง

- Orthographic เป็นวิธีที่ทำให้รูปที่วาดลงบน Screen แบบง่ายๆจึงละเลยในส่วนของจุด z ไปทำให้ภาพที่ได้คล้ายกับ 2 มิติ

2.2.8 Rendering

การ Render คือการสั่งให้โปรแกรมประมวลผลทุกสิ่งทุกอย่างไม่ว่าจะเป็น Material หรือแสงเงาของภาพ 3 มิติให้ออกมาเป็นภาพ 2 มิติสำหรับการนำไปใช้งาน

2.2.9 Biped

Biped คือ ชุดโครงกระดูกสัตว์สองเท้า (มนุษย์) สำเร็จรูปที่ Character Studio ได้เตรียมเอาไว้ให้ใช้งานแทนใช้งานกระดูก (Bone) แบบเดิมๆ ที่มีใช้งานใน 3D Studio Max เมื่อเราต้องการให้ตัวละครของเราทำงานโดยอาศัยความสามารถของ Character Studio เราก็จะต้องใช้ Biped เป็นกระดูกให้กับตัวละครนั้นๆ ซึ่ง Biped เองนั้นจะช่วยให้เราลดขั้นตอนในการสร้างกระดูก เชื่อมต่อกระดูกชิ้นต่างๆ เข้ามาเป็นรูปร่างตัวละคร การทำงานที่เกี่ยวข้องกับการ Setup IK รวมทั้งการกำหนดชื่อให้กับกระดูกแต่ละชิ้นด้วย (ซึ่งบางท่านอาจจะไม่ให้ความสำคัญกับขั้นตอนการตั้งชื่อให้กับตัวละครเท่าไรนัก แม้แต่ตัวผมเองในบางโอกาสของการทำงานก็ตาม) ซึ่งการทำงานในขั้นตอนเหล่านี้เป็นเรื่องที่น่าเบื่อมากๆ โดยเฉพาะกับงานที่ต้องมีตัวละครมากๆ

ในหุ่นของ Biped หนึ่งตัวจะประกอบด้วยชิ้นส่วนต่างๆ มากมายหลายชิ้นตามจำนวนชิ้นส่วนข้อต่อหลักของมนุษย์ในส่วนของสะโพกเราเห็นรูปสี่เหลี่ยมเล็กๆ ฝังอยู่ด้านใน (มองเห็นชัดใน Wireframe) ชิ้นส่วนนี้เรียกว่า Center of Mass ทำหน้าที่เป็นศูนย์กลาง (Root) ของ Biped

2.2.10 Animation

Keyframe Animation - เป็นที่ทราบกันทั่วไปว่า หลักการทำ Animation ขั้นพื้นฐานนั้นก็คือการนำเอาภาพนิ่งที่ต่อเนื่องกันจำนวนมากๆมาเปิดทีละภาพต่อกันด้วยความเร็วสูง ภาพชุดนั้นๆ ก็จะดูเป็นภาพ เคลื่อนไหวขึ้นมาได้ ภาพแต่ละภาพที่ถูกเปิดขึ้นมาทีละภาพนั้นเรียกว่า Frame โดยหน่วยที่ ใช้วัดคุณภาพของ Animation ก็จะใช้การนับจำนวนของภาพหรือ Frame ที่จะถูกเปิดขึ้นมาในช่วง เวลา 1 วินาที เช่น Animation แบบ 8 Frame ต่อวินาทีนั้นหมายถึง ในหนึ่งช่วงเวลา 1 วินาทีจะต้องใช้ภาพนิ่งจำนวน 8 ภาพ ซึ่งการทำ Animation ในลักษณะนี้จะเห็นว่ายุ่งยากมาก เพราะถ้าเป็น Animation เรื่องยาวๆ ก็เป็นอันต้องเขียนภาพจำนวนมากๆ อย่างหลีกหนีไปไม่ได้ แต่วิธีการทำ Animationแบบนี้ก็ถือว่าเป็นวิธีพื้นฐานที่สุดแม้ทุกวันนี้การทำ Animation หลายๆ เรื่องก็ยังคงใช้การทำงานแบบนี้อยู่

สำหรับการทำงาน Animation ในงาน 3D นั้นก็ยังอาศัยหลักการเดียวกันกับการทำ Animation ที่ได้กล่าวมาแล้ว แต่ก็ได้มีการลดขั้นตอนการทำงานให้ย่นย่อสะดวกสบายยิ่งขึ้น

เรียกว่าการทำ Animation แบบ In-Between Frame ซึ่งการทำ Animation ใน ลักษณะนี้เราจะกำหนดให้มี Frame บาง Frame ทำหน้าที่เป็นตัวบันทึกจังหวะในการ เคลื่อนที่ในแต่ละช่วงเวลาของวัตถุใน Scene ซึ่งเราจะเรียก Frame ที่ทำหน้าที่บันทึกการ เคลื่อนที่ของวัตถุนั้นว่า Key Frame สำหรับการทำ Animation ในแบบของการกำหนด Key Frame นั้นจะทำโดยการนำเอาตำแหน่งของในแต่ละ Key Frame มาคำนวณหาความ เป็นไปได้ของตำแหน่งใน Frame ที่อยู่ ระหว่าง Key Frame แบบอัตโนมัติ ซึ่งการทำงาน ในลักษณะนี้จะประหยัดเวลากว่ามากสำหรับการทำ Animation แบบเดิมๆ โดยเฉพาะกับ Animation ที่มีความยาวมากๆ แต่กลับใช้คนน้อยๆ ในการทำงานซึ่งโปรแกรม คอมพิวเตอร์ที่ใช้ทำงานด้าน Animation เกือบจะทั้งหมดต่างก็ใช้หลักการทำ Animation แบบ Key Frame เป็นหลักทั้งสิ้น

2.2.11 Three-Dimensional Mathematics

2.2.11.1 ระบบพิกัด (Coordinate Systems)

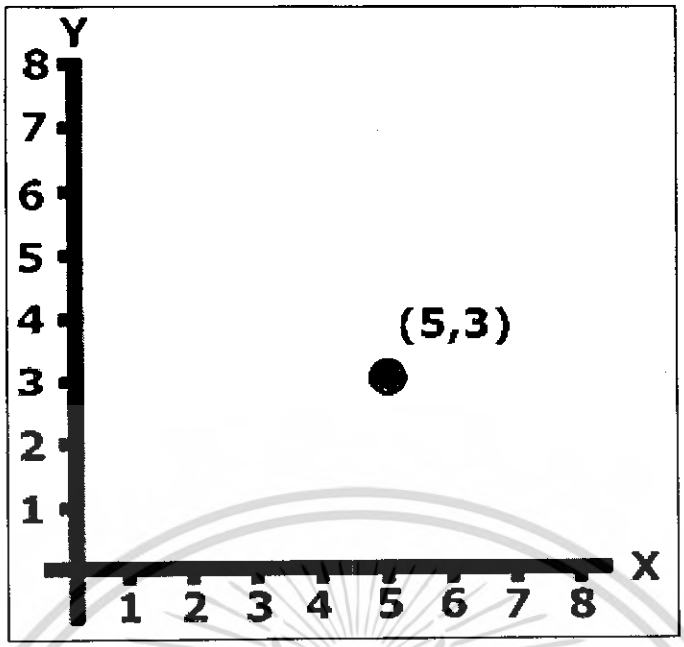
2.2.11.1.1 ระบบพิกัดหนึ่งมิติ

พิกัด 1 มิติเป็นพิกัดที่ง่ายที่สุดในการศึกษาระบบพิกัดเช่น ถ้าคุณต้องการเดินไปข้างหน้าหรือเดินถอยหลัง โดยให้จุดที่คุณยืนอยู่ในตอนเริ่มต้นเป็นจุดกำเนิด (จุดกำเนิดคือจุดที่พิกัดศูนย์เสมอ) จุดที่อยู่ด้านขวาของจุดเริ่มต้นจะเป็นจุดที่มีค่าเป็นบวกเสมอ เช่น 1,2,3,4 เป็นต้น ส่วนจุดที่อยู่ด้านซ้ายของจุดเริ่มต้นจะเป็นจุดที่มีค่าเป็นลบเสมอ เช่น -1,-2,-3,- 4 เป็นต้น

รูปที่ 2-2 A number line

2.2.11.1.2 ระบบพิกัดสองมิติ

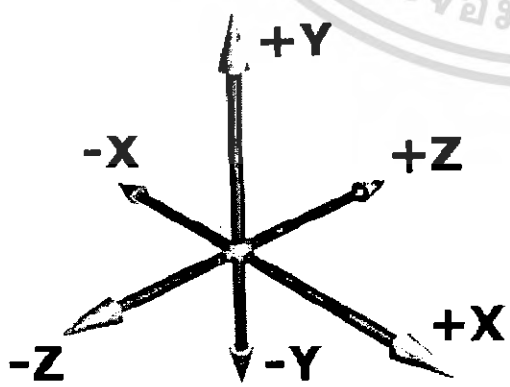
มาตรฐานกราฟ xy ที่เป็นรูปแบบดั้งเดิมของของระบบภาพ 2 มิติ ซึ่งสามารถเคลื่อนที่ขึ้น ,ลง,ซ้าย,ขวา ซึ่งระบบนี้เราเรียกว่า ระบบคาร์ทีเซียนโคออดิเนต (Cartesian Coordinate System) ซึ่งก็คือการนำเอาพิกัด 1 มิติสองอันมาไขว้กันโดยแต่ละอันจะเรียกว่าแกน โดยแกนในแนวนอนซึ่งจะมีค่าเพิ่มขึ้นจากซ้ายไปขวา (เหมือนในระบบพิกัด 1 มิติ) เรียกว่าแกน X และแกนในแนวตั้งซึ่งจะมีค่าเพิ่มขึ้นจากล่างขึ้นบนจะเรียกว่าแกน Y โดยจุดตัดระหว่างแกน X กับแกน Y จะเรียกว่าจุดกำเนิด ดังรูปที่ 2-3



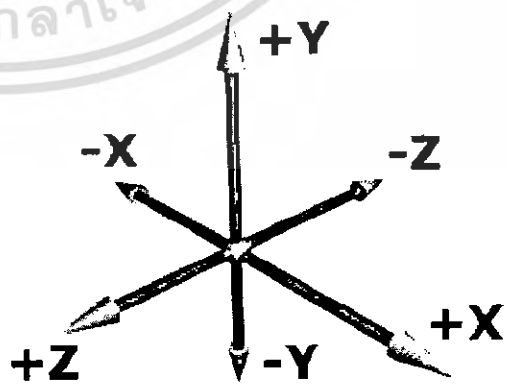
รูปที่ 2-3 Cartesian coordinate system

2.2.11.1.3 ระบบพิกัดสามมิติ

หมายถึงทิศทางของค่าในแนวแกน x , y และ z ของจุดกำเนิด (origin) ในระบบสามมิติ ซึ่งจะมีค่า x , y และ z เป็น 0 , 0 และ 0 ตามลำดับ และโดยปกติแล้วทิศทางของค่า x จะมีค่าเพิ่มขึ้นในทิศทางไปทางด้านขวา และลดลงเมื่อไปทางด้านซ้ายมือของจุดกำเนิด ส่วนค่า y จะเพิ่มขึ้นไปในทางด้านบนและลดลงเมื่อไปทางด้านล่าง แต่ค่า z จะมีให้เลือกอยู่ 2 แบบ แบบแรกจะเพิ่มขึ้นในทิศทางเข้าไปในจอภาพ และแบบที่สองจะเพิ่มขึ้นในทิศทางออกจากจอภาพ โดยแบบแรกเรียกว่า Left-Hand System ดังรูปที่ 2-4 ส่วนแบบสองเรียกว่า Right-Hand System ดังรูปที่ 2-5



รูปที่ 2-4 Left-Hand System



รูปที่ 2-5 Right-Hand System

2.2.11.2 เวกเตอร์ (Vector)

เวกเตอร์เป็นปริมาณที่มีทั้งขนาดและทิศทางซึ่งเป็นองค์ประกอบทางคณิตศาสตร์ที่มีความสำคัญมากด้านการประยุกต์ใช้ในงานการประมวลผลกราฟิก ซึ่งจะต่างกับปริมาณสเกลาร์ตรงที่เวกเตอร์นั้นมีทิศทาง หรือกล่าวได้อีกอย่างหนึ่งว่าเวกเตอร์มีองค์ประกอบมาจากปริมาณสเกลาร์ n ตัว เพื่อแทนขนาดและทิศทางในระบบ n มิติมักจะแทนเวกเตอร์โดยใช้สัญลักษณ์ $\overrightarrow{OP}$ โดยจะหมายถึงเวกเตอร์นี้มีทิศทางจากจุด O ไปยังจุด P และมีขนาดเท่ากับระยะห่างระหว่างจุด O กับจุด P

- การบวกเวกเตอร์

$$R = V1 + V2 \tag{2.1}$$

$$R = (V1_x + V2_x, V1_y + V2_y, V1_z + V2_z) \tag{2.2}$$

- การลบเวกเตอร์

$$R = V1 - V2 \tag{2.3}$$

$$R = (V1_x - V2_x, V1_y - V2_y, V1_z - V2_z) \tag{2.4}$$

- การคูณเวกเตอร์ด้วยปริมาณสเกลาร์ ซึ่งทำได้โดยการคูณปริมาณสเกลาร์กับทุกคอมโพเน็นต์ของเวกเตอร์นั้นๆ

$$V*s = (V_x * s, V_y * s, V_z * s) \tag{2.5}$$

- สำหรับขนาดของเวกเตอร์นั้นสามารถเขียนแทนด้วยสัญลักษณ์ |V| ซึ่งสามารถหาได้โดยการใช้ กฎของพิทากอรัส ดังสมการ

$$|V| = \sqrt{v_x^2 + v_y^2 + v_z^2} \tag{2.6}$$

- เวกเตอร์ที่มีขนาดเท่ากัน (Parallel Vector) หมายถึง เวกเตอร์คู่ใดๆ ที่มีทิศทางเดียวกันหรือทิศทางตรงกันข้าม
- เวกเตอร์ร่วมระนาบ หมายถึงเวกเตอร์ตั้งแต่ 3 ตัวขึ้นไปที่อยู่ในระนาบเดียวกัน

2.3 โครงข่ายใยประสาทเทียม

โครงข่ายใยประสาทเทียม(An Artificial Neural Network หรือANN) เป็นการประมวลผลข้อมูลที่เลียนแบบรูปแบบการทำงานของ cell ประสาทของมนุษย์ ดั่งการทำงานระบบสมองของมนุษย์ ซึ่งจุดสำคัญของอัลกอริทึม (Algorithm) นี้คือโครงสร้างข้อมูลของ processing system ที่ประกอบด้วยค่าตัวเลขของการติดต่อระหว่างโปรเซสซิ่งอิลิเมน (processing elements) ที่ทำงานสอดคล้องกันเพื่อแก้ปัญหาต่างๆ

โครงข่ายใยประสาทเทียมเหมือนกับมนุษย์คือการเรียนรู้จากตัวอย่าง โดย โครงข่ายใยประสาทเทียม จะเปลี่ยนลักษณะจำเพาะของตัวเองบางอย่างเพื่อที่จะจดจำรูปแบบของข้อมูลในการเรียนรู้ดั่งเช่นในระบบประสาทที่จะปรับไซแนปติกคอนเน็คชั่น (synaptic connections) ที่อยู่ระหว่าง นิวรอล

นิวรอลเน็ตเวิร์คมีคุณสมบัติในการตีความหมายรูปแบบข้อมูลของ นิวรอลเน็ตเวิร์ค ดังที่ได้กล่าวไปแล้วนั้นจะเห็นได้ว่า นิวรอลเน็ตเวิร์คสามารถคาดการณ์รูปแบบข้อมูลได้ดังนั้นมันจึงถูกใช้ในการอนุมานหรือวิเคราะห์แนวโน้มรูปแบบของข้อมูลที่มีความซับซ้อน ซึ่งด้วยความสามารถสูงส่งที่สามารถคาดเดาคำตอบจากข้อมูลยากๆ หรือไม่สมบูรณ์ได้นี้มันจึงถูกใช้สกัดรูปแบบ และค้นหาเทรดที่ซับซ้อนเกินมนุษย์หรือ เทคนิคทางคอมพิวเตอร์ อื่นๆจะทำได้ และ นิวรอลเน็ตเวิร์คที่ได้ถูกฝึกฝนแล้วสามารถมองเป็น " ผู้เชี่ยวชาญ (expert) " ในสาขาข้อมูลที่มันถูกสั่งให้วิเคราะห์ได้ โดย "ผู้เชี่ยวชาญ" นี้ สามารถให้เหตุผลจากแนวคิดหรือสถานการณ์ใหม่ๆ ได้

สรุปแล้ว นิวรอลเน็ตเวิร์คมีข้อได้เปรียบกว่าระบบเรียนรู้อื่นอื่นๆ คือ

1. มีการเรียนรู้แบบอแดปทีฟ(Adaptive learning): สามารถเรียนรู้ได้จากข้อมูลและการฝึกฝนที่ผ่านมา

2. มีการจัดการตัวเอง(Self-Organization): โครงข่ายใยประสาทเทียม สามารถจัดข้อมูลและนำข้อมูลที่ได้เรียนรู้มาแสดงได้

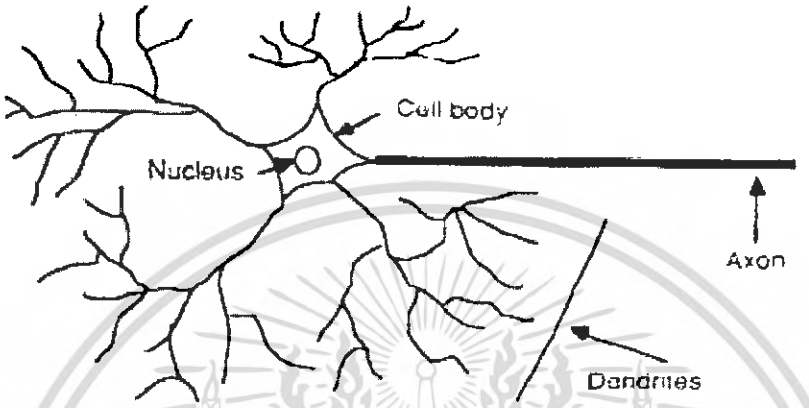
3. มีการทำงานแบบเรียลไทม์(Real Time Operation): โครงข่ายใยประสาทเทียมสามารถทำงานในแบบขนาน ซึ่งฮาร์ดแวร์หลายๆ อย่างสามารถนำข้อดีนี้มาพัฒนาเพิ่มประสิทธิภาพได้

4. มีความทนทานต่อการเสียหายของเน็ตเวิร์ค(Fault Tolerance via Redundant Information Coding): การทำลายบางส่วนของเน็ทเวิร์คนั้นทำให้ประสิทธิภาพรวมลดลง แต่ก็ยังสามารถกู้คืนความสามารถได้บางส่วนแม้เสียหายหนัก

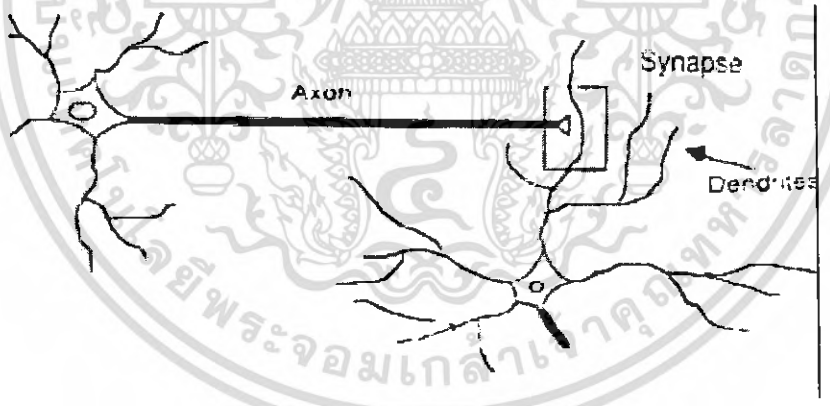
2.3.1 เปรียบเทียบระหว่างสมองมนุษย์กับโครงข่ายใยประสาทเทียม

ส่วนใหญ่เรายังไม่รู้วิธีที่สมองคนใช้ในการเรียนรู้ ดังนั้นในทางทฤษฎีนิวรอลจะเก็บสัญญาณ จากโครงสร้างที่เรียกว่า เดนไดรท์นิวรอล จะส่งสัญญาณไฟฟ้าไปทางเส้นยาวบากที่เรียกว่าแอกซอนที่แตกออกไปเป็นพันสาขาและที่ปลายสาขานั้นโครงสร้างชื่อไซแนป(synapse)

จะเปลี่ยนสัญญาณออกเป็นสัญญาณไฟฟ้าที่จะไปกระตุ้นหรือยับยั้ง นิวรอล ที่ต่ออยู่ เมื่อได้รับสัญญาณกระตุ้นที่มีค่าสูงกว่าค่ายับยั้งพอสมควร มันจะส่งสัญญาณไฟฟ้าไปทางแอกซอน การเรียนรู้เกิดขึ้นจากการปรับปรุงเส้นทางของโนด(Node) ซึ่งส่งผลต่ออันอื่นๆ ด้วย

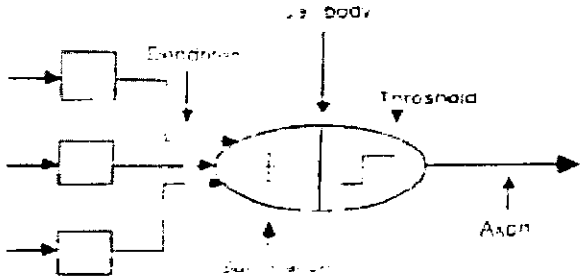


รูปที่ 2-6 Components of a neuron



รูปที่ 2-7 The synapse

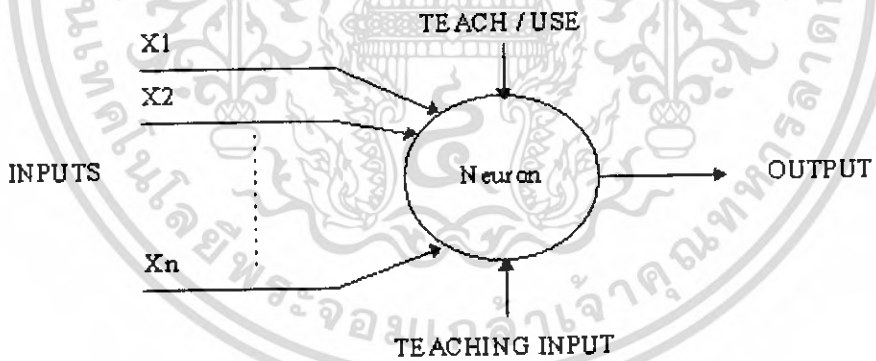
เราทดลองสร้างนิวรอลเน็ตเวิร์ค(Neural networks) โดยการจำลองโครงสร้างที่สำคัญของนิวรอล และสิ่งเชื่อมต่อ และสร้างโปรแกรมเพื่อจำลองการทำงานของมัน แต่เนื่องจากความรู้จำกัดและความสามารถทางเทคโนโลยี แบบจำลองของเราจึงเป็นเพียงรูปแบบแนวความคิดคร่าวๆ ของนิวรอล เท่านั้น



รูปที่ 2-8 The neuron model

2.3.2 พื้นฐานของนิวรอล (Simple neuron)

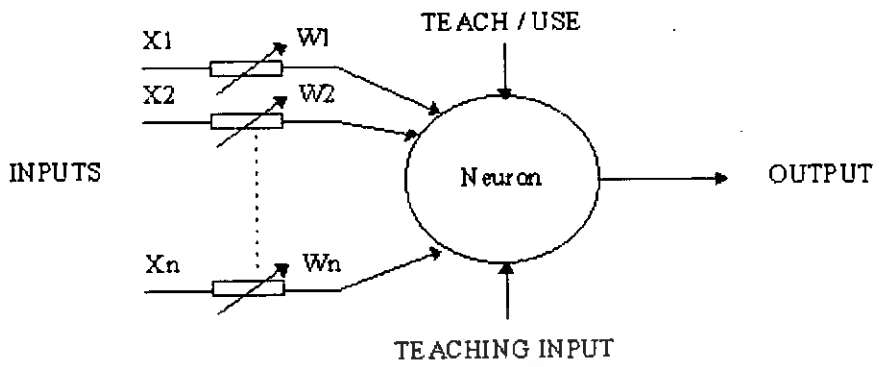
ตามหลักการแล้ว โครงข่ายใยประสาทเทียมมีหลายอินพุตส่วน เอาต์พุต มีเพียงเอาต์พุตเดียว การทำงานมี 2 ลักษณะคือการฝึกฝน(training mode) และ การใช้งานจริง (using mode) ในการฝึกฝน(training mode) นั้น นิวรอล สามารถฝึกฝนที่จะตอบสนอง สำหรับอินพุตที่มีลักษณะจำเพาะ ส่วน การใช้งานจริง (using mode) เมื่อได้รับอินพุตที่เป็นลักษณะเดียวกับที่ได้เรียนรู้มา เอาต์พุต ที่ได้จะเป็น เอาต์พุต ที่เกี่ยวเนื่องกับอินพุตนั้น แต่ถ้ารูปแบบอินพุต นั้นไม่เคยเรียนรู้มากฎการตัดสินใจ(firing rule)จะถูกใช้เพื่อทำการตัดสินใจแทน



รูปที่ 2-9 A simple neuron

2.3.3 นิวรอลในยุคต่อมา

นิวรอลรุ่นเดิม จะไม่ทำอะไรที่คอมพิวเตอร์ ทั่วไปไม่ได้ทำอยู่แล้ว นิวรอลแบบที่พัฒนาขึ้นแล้ว(รูปที่ 2) คือ McCulloch and Pitts model (MCP). ความแตกต่างกันรุ่นเก่าคืออินพุตจะมีการถ่วงน้ำหนัก คือ ผลของการที่แต่ละอินพุตจะตัดสินใจว่าจะแสดงผลอย่างไรจะขึ้นอยู่กับน้ำหนักของอินพุตนั้นๆ ซึ่ง น้ำหนักของอินพุตจะเป็นตัวเลขซึ่งเมื่อคูณกับอินพุตจะทำให้เกิดอินพุตถ่วงน้ำหนัก ซึ่งอินพุตถ่วงน้ำหนักนี้จะถูกรวมเข้าด้วยกัน และถ้ามันเกินค่าเทรสโฮลด (threshold value) ที่ตั้งไว้ นิวรอล จะ ตอบสนอง และในกรณีอื่น นิวรอล จะไม่ ตอบสนอง



รูปที่ 2-10 An MCP neuron

ในเชิงคณิตศาสตร์นิวรอลจะ ตอบสนอง ก็ต่อเมื่อ

$$X_1W_1 + X_2W_2 + X_3W_3 + \ldots > T$$

การรวมกันของน้ำหนักอินพุตและของเทรสโฮลด(threshold) ทำให้ นิวรอล นี้ยืดหยุ่นและมีความสามารถสูง MCP neuron มีความสามารถที่จะปรับตัวให้เข้ากับสถานการณ์เฉพาะอย่างโดยการเปลี่ยนน้ำหนักและ/หรือเทรสโฮลด(threshold) อัลกอริทึมมากมายมีขึ้นเพื่อให้ นิวรอลสามารถ 'ปรับตัว' ได้อันที่ใช้บ่อยที่สุดคือกฎเดลต้า (Delta rule) และ แบคเออเรอร์พรอพพาเกชั่น (back error propagation) แบบต้นๆ จะถูกใช้ในฟีดฟอเวิร์ดเน็ตเวิร์ค(feed-forward networks)

นิวรอลคำนวณ เอาต์พุตจากอินพุตได้โดยการหาผลรวม ของอินพุตที่ถ่วงน้ำหนักแล้ว (netweighted) มาเทียบกับค่า เทรสโฮลด(threshold) ถ้ามากกว่าหรือเท่ากับ จะมี เอาต์พุต เป็น 1 แต่ถ้าน้อยกว่าจะเป็น 0 สามารถเขียนเป็นฟังก์ชั่น ได้เป็น

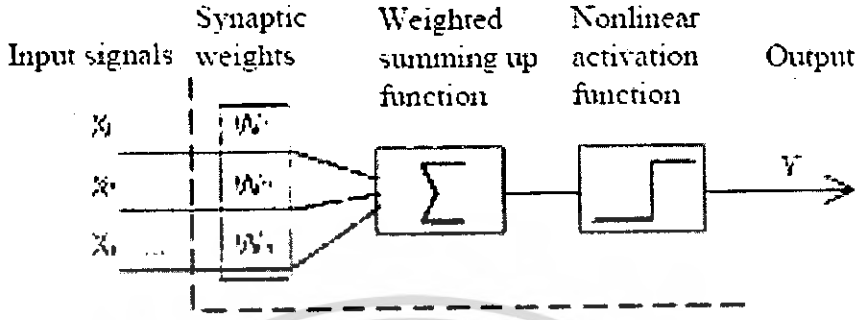
$$X = \sum_{i=1}^{n} x_i w_i$$
$$Y = \begin{cases} 1 \mid X \geq \theta \\ 0 \mid X < \theta \end{cases} \qquad (2.7)$$

เมื่อ X เป็นผลรวมอินพุตที่ถ่วงน้ำหนักแล้ว, x_i เป็นค่าของแต่ละอินพุต I, w_i เป็นน้ำหนักถ่วงของแต่ละอินพุต i, n เป็นจำนวนของ input, Y เป็น เอาต์พุตของ นิวรอลที่ผ่านแอกทิเวชั่นฟังก์ชั่น(activation function)แล้ว และ θ เป็น เทรสโฮลด(threshold)

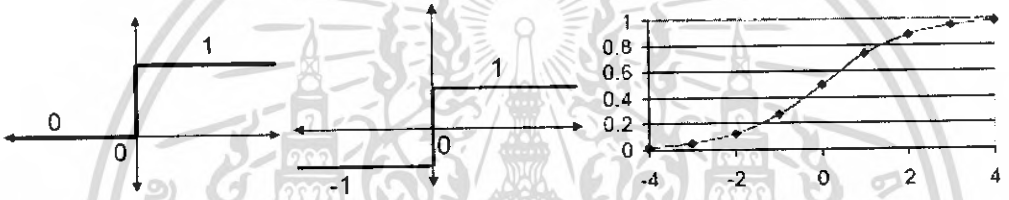
เราสามารถเขียนในรูปของสเตปฟังก์ชั่น(step function) ได้เป็น

$$Y = step\left[\sum_{i=1}^{n} x_i w_i - \theta\right] \qquad (2.8)$$

เวท(Weight) และเทรสโฮลด (threshold) เริ่มต้นจะกำหนดโดยสุ่มค่าขึ้นมา



รูปที่ 2-11 Activation function สามารถใช้ได้หลาย function



$$step: Y = \begin{cases} 1 \mid X \geq 0 \\ 0 \mid X < 0 \end{cases} \quad sign: Y = \begin{cases} +1 \mid X \geq 0 \\ -1 \mid X < 0 \end{cases} \quad sigmoid: Y = \frac{1}{1+e^{-X}}$$

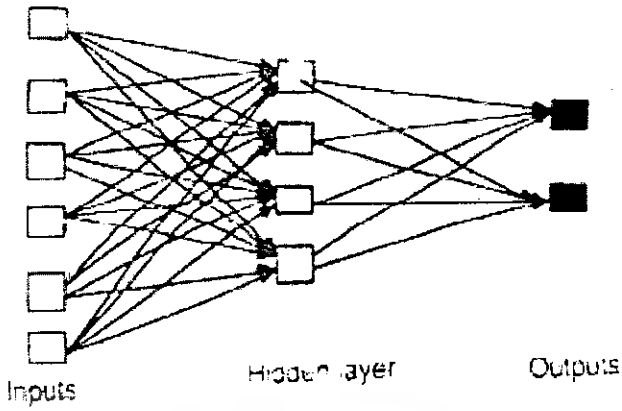
รูปที่ 2-12 activation function ต่างๆ

แต่ละฟังก์ชั่นข้อดีแตกต่างกันในเรื่องความสะดวกในการใช้งาน ความละเอียดของเอาต์พุต และการเรียนรู้ เป็นต้น

2.3.4 โครงสร้างของโครงข่ายใยประสาทเทียม

2.3.4.1 แบบฟีดฟอเวิร์ด (Feed-forward networks)

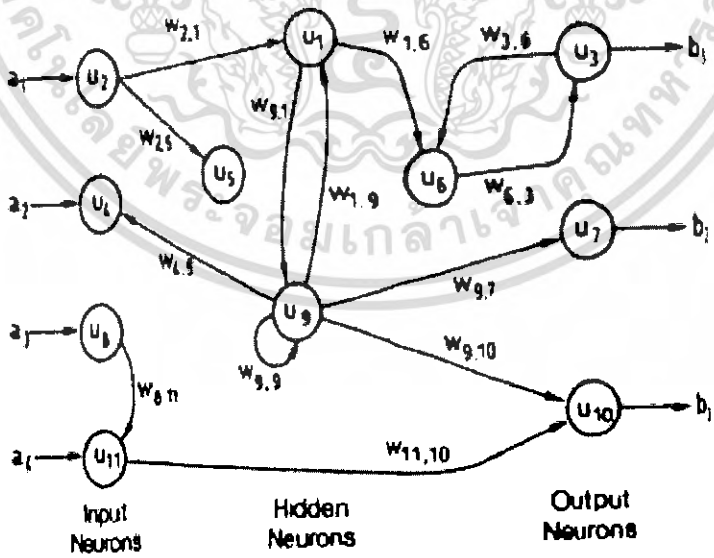
ฟีดฟอเวิร์ด(รูป 4.18) ยอมให้สัญญาณผ่านได้ทางเดียวจากอินพุตสู่ เอาต์พุตโดยไม่มีฟีดแบค(feedback หรือ loops)เช่น เอาต์พุตของเลเยอร์ใดๆ จะไม่มีผลกับเลเยอร์เดิม ฟีดฟอเวิร์ดนั้นจะเป็น เน็ตเวิร์ค ที่เป็นเส้นตรง เชื่อมอินพุตกับ เอาต์พุต เข้าด้วยกัน.ตัวมันเองจะถูกใช้บ่อยในการวิเคราะห์รูปแบบข้อมูล การจัดรูปแบบของมันนี้ถูกเรียกว่าแบบ Bottom-up หรือ Top-down.



รูปที่ 2-13 An example of a simple feed forward network

2.3.4.2 แบบย้อนกลับ (Feedback networks)

ฟีดแบค(รูป 1) ยอมให้สัญญาณผ่าได้ทั้งสองทิศทางโดยการทำให้มีวนซ้ำ(loop) ใน ฟีดแบคนั้นมีความสามารถสูงมากและสามารถที่จะซับซ้อนมากได้ด้วย ฟีดแบคนั้นเป็นแบบไดนามิค เพราะ 'สถานะ' ของมันเปลี่ยนแปลงตลอดเวลาจนกว่ามันจะถึงจุดสมดุล. มันจะคงอยู่ที่จุดสมดุลจนกว่าอินพุตจะเปลี่ยนและต้องหาจุดสมดุลใหม่ สถาปัตยกรรมฟีดแบคอาจถูกเรียกได้เป็นอินเตอร์แรคทีฟ(Interactive) หรือ รีเคอร์เรนท์(Recurrent) ถึงแม้ว่ารูปแบบหลังมักจะถูกใช้เรียกแทนฟีดแบคคอนเนคชั่น(feedback connections) ใน single-layer organizations.



รูปที่ 2-14 An example of a complicated network

2.3.5 เน็ตเวิร์คเลเยอร์ (Network layers)

รูปแบบธรรมดาที่สุดของ โครงข่ายใยประสาทเทียม(An Artificial Neural Network หรือ ANN) ประกอบด้วย 3 กลุ่มหรือ เลเยอร์ของยูนิต: เลเยอร์ ของ "อินพุต" ยูนิต จะเชื่อมต่อกับ เลเยอร์ ของ "ฮิดเดน(hidden)"ยูนิต ซึ่งจะไปเชื่อมต่อกับเลเยอร์ของ "เอาต์พุต" ยูนิต (ดูรูป 2.13)

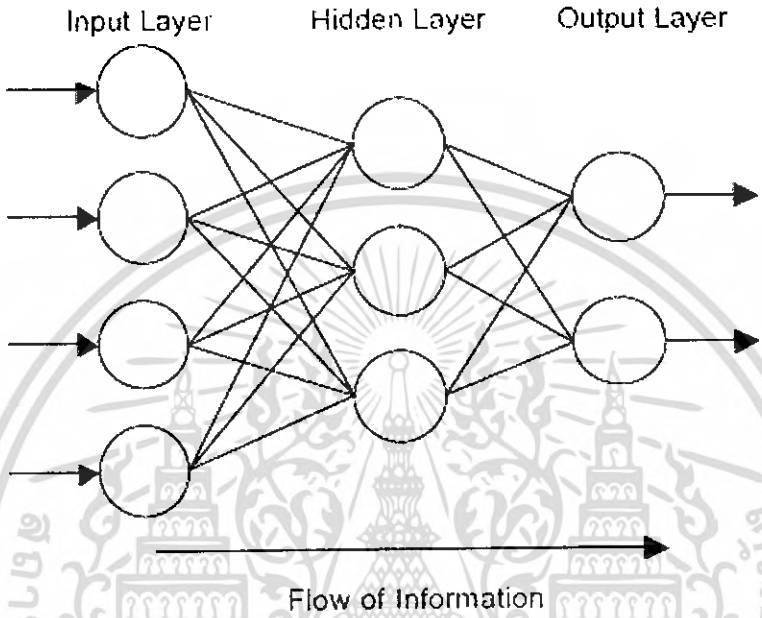
- ความเคลื่อนไหวของ อินพุตยูนิต จะแสดงข้อมูลดิบที่ถูกป้อนเข้าในเน็ตเวิร์ค
- ความเคลื่อนไหวของแต่ละฮิดเดนยูนิต (hidden unit) จะถูกตัดสินใจจากความเคลื่อนไหวของอินพุตยูนิตและน้ำหนักในการเชื่อมต่อระหว่างอินพุตและฮิดเดนยูนิต (hidden unit).
- พฤติกรรมของ เอาต์พุตยูนิต ขึ้นอยู่กับความเคลื่อนไหวของ ฮิดเดนยูนิต (hidden unit) และน้ำหนักระหว่างฮิดเดน (hidden) และเอาต์พุตยูนิต

รูปแบบของ เน็ตเวิร์คอย่างง่ายนี้ มีความน่าสนใจเพราะ ฮิดเดนยูนิต (hidden unit) นั้นมีอิสระในการสร้างวิธีการนำเสนอของอินพุต น้ำหนักระหว่างอินพุตและ ฮิดเดนยูนิต (hidden unit) จะตัดสินใจว่าเมื่อใด ฮิดเดนยูนิต (hidden unit) จะ ทำงานและดังนั้น ด้วยการแก้ไขค่า เวท, ฮิดเดนยูนิต (hidden unit) สามารถเลือกได้ว่าจะแสดงผลอะไร.

เรายังสามารถแบ่งได้อีกเป็นสถาปัตยกรรมแบบ single-layer และ multi-layer โดยรูปแบบ single-layer นั้นยูนิต ทั้งหมดจะเชื่อมต่อซึ่งกันและกัน ประกอบรวมได้ รูปแบบมาตรฐานจำนวนมาก และมีพลังในการคำนวณสูงกว่ารูปแบบ multi-layer แบบลำดับชั้นใน multi-layer networks units มักจะถูกจัดเลขโดย เลเยอร์แทนที่จะตามระบบจัดเลขแบบ โกลบอลนัมเบอร์ริ่ง(global numbering)

2.3.6 มัลติเลเยอร์นิวรอลเน็ตเวิร์ค (Multilayer Neural Network)

เนื่องจากเพียงนิวรอลเดียวไม่สามารถแก้ปัญหาที่ซับซ้อนเกินกว่าเส้นแบ่งเทรสโฮลด (threshold) เส้นเดียวได้จึงต้องนำมาเชื่อมกันหลายๆ ชั้นส่งค่าต่อกันไปเรื่อยๆ เหมือนเป็นการช่วยกันตัดสินใจในหลายๆ แง่มุม



รูปที่ 2-15 Flow of Information

นิวรอลที่ใช้ภายในเน็ตเวิร์ค สามารถใช้ต่างประเภทกันได้ แต่ใน ฮิดเดนเลเยอร์ (hidden layer) จำเป็นต้องใช้ประเภทที่มี เอาต์พุตเป็นฟังก์ชั่น ต่อเนื่อง เพราะจำเป็นต้องหาอนุพันธ์ในขั้นตอนการเรียนรู้

การเรียนรู้โดยปรับ เวท ในแต่ละเลเยอร์สามารถคำนวณเหมือน single layer ด้วยอินพุต และ เอาต์พุตของ นิวรอลนั้น แต่ค่าความผิดพลาดของเน็ตเวิร์คไม่สามารถใช้กับนิวรอลในแต่ละเลเยอร์ได้ จะต้องใช้เออเรอกราเดียน(error gradient)แทน

$$\delta_j(p) = \frac{\partial y_j(p)}{\partial X_j(p)} \times e_j(p) \qquad (2.9)$$
$$\Delta w_{ij}(p) = \alpha \times x_j(p) \times \delta_j(p)$$

เมื่อ $\delta_k(p)$ คือ เออเรอกราเดียน, $y_k(p)$ คือ เอาต์พุตของ นิวรอล k และ $X_k(p)$ คือ เนทเวท(net weighted) ของอินพุตของ นิวรอล k

โดยมากแล้ว กับปัญหาที่ไม่ซับซ้อนมากเราจะใช้ ฮิดเดนเลเยอร์เพียง 1เลเยอร์เท่านั้นคือมี อินพุตเลเยอร์, ฮิดเดนเลเยอร์และ เอาต์พุตเลเยอร์ สำหรับปัญหาที่ซับซ้อนเกินไปก็สามารถเพิ่มได้ แต่ 1เลเยอร์ที่มากขึ้นก็กินพลังและเวลาในการคำนวณมากขึ้น

2.3.7 เพอเซปตรอน (Perceptrons)

งานวิจัยที่มีอิทธิพลต่อวงการโครงข่ายใยประสาทเทียมเกิดขึ้นในยุค 60 ภายใต้หัวข้อ "perceptrons" คิดค้นขึ้นโดย Frank Rosenblatt จากนั้นมันได้ถูกเขียนเป็น MCP model นิวรอลที่มีการถ่วงน้ำหนักอินพุตโดยการปรับปรุงแก้ไขบางส่วน ยูนิต ชื่อ A1, A2, Aj , Ap จะถูกเรียกว่า แอสโซซิเอชั่นยูนิต(association units)และงานของมันคือการแยกรูปแบบที่จำเพาะและจำกัดวง จากรูปที่อินพุตเข้ามา เพอเซปตรอนเลียนแบบแนวความคิดพื้นฐานของระบบการรับภาพของสัตว์ และถูกใช้ในการทำการวิเคราะห์คุณลักษณะเป็นหลัก แต่ขีดความสามารถของมันยังสามารถทำหน้าที่ได้อีกมาก

ในปี 1969 Minsky และ Papert ได้เขียนหนังสือซึ่งได้บรรยายข้อจำกัดของ ซิงเกิลเลเยอร์เพอเซปตรอน (single layer Perceptrons) เอาไว้ มีผลกระทบอย่างใหญ่หลวงจากอิทธิพลของหนังสือเล่มนั้นและทำให้นักวิจัย โครงข่ายใยประสาทเทียมหลายคนหมดสิ้นความสนใจไปทีเดียว หนังสือเล่มนั้นอธิบายไว้อย่างสมบูรณ์และแสดงให้ดูทางคณิตศาสตร์ว่า ซิงเกิลเลเยอร์เพอเซปตรอนไม่สามารถวิเคราะห์คุณลักษณะแบบพื้นฐานบางอย่างได้ เช่น ตัดสินใจด้านความคล้ายกันของรูปทรงหรือตัดสินใจว่ารูปทรงนั้นเชื่อมต่อกันหรือไม่ ซึ่งไม่มีใครรู้มาก่อนจนกระทั่งถึงปี 80 ว่าถ้าให้การเรียนรู้ที่ถูกต้องแล้วมัลติเพิ้ลเพอเซปตรอน (multiple perceptrons)นั้นสามารถทำได้

สุดท้ายเราสามารถสรุปหลักการ เพอเซปตรอนได้คือ เป็นการให้อินพุตเป็นชุดของข้อมูลชุดหนึ่งพร้อมกับมีคำตอบที่ถูกต้องไว้แล้ว และตรวจสอบว่าเน็ตเวิร์ค นั้นสามารถให้คำตอบเป็นที่ยอมรับได้หรือไม่ ถ้าไม่ก็จะต้องทำให้เน็ตเวิร์คนั้นเรียนรู้และจดคำตอบที่ถูกต้องไว้

การเรียนรู้ทำได้โดยปรับ เวท ของแต่ละอินพุตเพื่อให้ได้ เอาต์พุตที่ถูกต้อง และนำ เอาต์พุตนั้นไปคำนวณหาค่าความผิดพลาด โดยเทียบกับค่าที่ถูกต้องสำหรับการปรับ เวท ดังสมการต่อไปนี้

$$
\begin{aligned}
e(p) &= Y_d(p) - Y(p) \\
\Delta w_i(p) &= \alpha \times x_i(p) \times e(p) \\
w_i(p+1) &= w_i(p) + \Delta w_i(p)
\end{aligned}
\qquad (2.10)
$$

เมื่อ e(p) คือ ค่าความผิดพลาด จาก เพอเซปตรอน p, Y(p) คือเอาต์พุตจริง, $Y_d(p)$ คือ เอาต์พุต (ที่ถูกต้อง (ควรจะเป็น), Δw คือค่าที่เวทจะเปลี่ยนไป, $w_i(p)$ คือ เวท ขณะที่รับอินพุตนั้น, $w_i(p+1)$ คือ เวท ใหม่หลังจากการปรับ และ α คือ อัตราการเรียนรู้ (learning rate)

ความแตกต่างของอินพุตที่ให้แก่นิวรอลตั้งแต่ [0, 1], [-1, +1], [1, 2] หรือแบบอื่นๆ มีผลต่อการปรับ เวท ของแต่ละอินพุตให้เปลี่ยนแปลงไปไม่เท่ากันจึงจะได้ค่าที่จะเป็น เวท ที่เหมาะสม เช่น [0, 1] มีผลให้อินพุตที่ได้รับค่า 1 เท่านั้นที่มีการปรับ เวท, [-1, +1] จะปรับเวทไปในทิศทางตรงกันข้ามและ [1, 2] อินพุตที่ได้ค่ามากกว่า เวท จะถูกปรับไปมากกว่า เป็นต้น

แต่จากสมการสังเกตได้ว่า ถ้า $\Delta w = 0$ เวท จะไม่มีการเปลี่ยนแปลง นั่นคือไม่เกิดการเรียนรู้ ซึ่งเกิดได้จาก 3 กรณี ไม่มีอัตราการเรียนรู้, ไม่มีค่าความผิดพลาดและอินพุตเป็น 0 สำหรับกรณีแรก $\alpha = 0$ เป็นที่แน่นอนอยู่แล้วว่าเราจะไม่เลือกกำหนดให้ learning rate เป็น 0 กรณีต่อไป $e = 0$ ก็เป็นเรื่องปกติอยู่แล้วว่าถ้าไม่เกิดความผิดพลาดก็ไม่ต้องปรับ เวท แต่กรณีสุดท้าย $x = 0$ อาจทำให้การเรียนรู้ช้าลงหรือผิดพลาดได้ เช่น ทั้งๆ ที่เกิด error ($e \neq 0$) แต่ไม่สามารถเรียนรู้ ได้เพราะผลคูณกับอินพุต(input)($x = 0$) ได้เท่ากับ 0 และค่าลบก็อาจทำให้เกิดการหักล้างกับค่าบวกได้ เพราะฉะนั้นจึงควรหลีกเลี่ยงการกำหนดรูปแบบอินพุตไม่ใช้ 0 หรือค่าลบในการแทนค่ารูปแบบหรือสัญลักษณ์ใดๆ (เฉพาะ single neuron)

ตัวอย่าง

สมมติว่าต้องการสร้าง neural network เลียนแบบการทำงานของ AND logic โดยแทนสัญลักษณ์ 0 ด้วย 1 และ 1 ด้วย 2 (เพื่อหลีกเลี่ยง input 0)

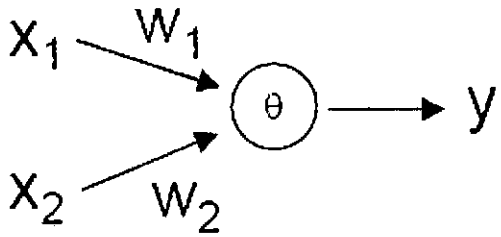
Activation function เลือกใช้ step function + 1

Training set ที่ใช้จะเหมือนกับ test set เนื่องจากมีกรณีที่เป็นไปได้น้อย

x_1	x_2	$y = x_1 \cdot x_2$
1	1	1
2	1	1
1	2	1
2	2	2

ขั้นที่ 1: สร้างและกำหนดค่า

มี 2 input, 1 output จะได้ network ดังรูป



รูปที่ 2-16 โครงสร้างของนิวรอนที่ออกแบบ

จากนั้น กำหนดค่าให้กับ w_1, w_2, θ โดยสุ่มค่าใน [-1.0, 1.0]

ขั้นที่ 2: เริ่มการทดสอบ

ส่ง input ให้กับ network แล้วเก็บ output ไว้ทีละชุด

ขั้นที่ 3: แก้ไขเพื่อเรียนรู้

คำนวณหา error ด้วย output กับ y ที่กำหนด แล้วนำไปปรับ weight

ขั้นที่ 4: เวียนซ้ำ

ทำซ้ำตั้งแต่ขั้นที่ 2 ไปเรื่อยๆ จนกว่าจะได้ผลลัพธ์ที่น่าพอใจ

ตารางแสดงการฝึกในแต่ละครั้ง โดยมี $\theta = 0.4$, $\alpha = 0.1$

Input		Desired output	Initial weights		Actual output	Error	Final weights	
x_1	x_2	Y_d	w_1	w_2	Y	e	w_1	w_2
0	0	0	0.7	-0.2	0			
0	1	0	0.7	-0.2	0	Pre-test		
1	0	0	0.7	-0.2	I			
1	1	1	0.7	-0.2	1			
0	0	0	0.7	-0.2	0	0	0.7	-0.2
0	1	0	0.7	-0.2	0	0	0.7	-0.2
1	0	0	0.7	-0.2	1	-1	0.6	-0.2
1	1	1	0.6	-0.2	0	1	0.7	-0.1
0	0	0	0.7	-0.1	0	0	0.7	-0.1

0	1	0	0.7	-0.1	0	0	0.7	-0.1
1	0	0	0.7	-0.1	1	-1	0.6	-0.1
1	1	1	0.6	-0.1	1	0	0.6	-0.1
0	0	0	0.6	-0.1	0	0	0.6	-0.1
0	1	0	0.6	-0.1	0	0	0.6	-0.1
1	0	0	0.6	-0.1	1	-1	0.5	-0.1
1	1	1	0.5	-0.1	1	0	0.5	-0.1
0	0	0	0.5	-0.1	0	0	0.5	-0.1
0	1	0	0.5	-0.1	0	0	0.5	-0.1
1	0	0	0.5	-0.1	1	-1	0.4	-0.1
1	1	1	0.4	-0.1	1	0	0.5	0.0
0	0	0	0.5	0.0	0	0	0.5	0.0
0	1	0	0.5	0.0	0	0	0.5	0.0
1	0	0	0.5	0.0	1	-1	0.4	0.0
1	1	1	0.4	0.0	1	0	0.4	0.0
0	0	0	0.4	0.0	0	0	0.4	0.0
0	1	0	0.4	0.0	0	0	0.4	0.0
1	0	0	0.4	0.0	1	-1	0.3	0.0
1	1	1	0.3	0.0	0	1	0.4	0.1
0	0	0	0.4	0.1	0	0	0.4	0.1
0	1	0	0.4	0.1	0	0	0.4	0.1
1	0	0	0.4	0.1	1	-1	0.3	0.1
1	1	1	0.3	0.1	1	0	0.3	0.1
0	0	0	0.3	0.1	0	0	0.3	0.1
0	1	0	0.3	0.1	0	0	0.3	0.1
1	0	0	0.3	0.1	0	0	0.3	0.1
1	1	1	0.3	0.1	1	0	0.3	0.1

2.3.8 ระบบการเรียนรู้

การจำรูปแบบและระบบโต้ตอบย่อยๆในตัว เน็ตเวิร์ค สามารถจัดได้เป็นสองรูปแบบต่อไปนี้:

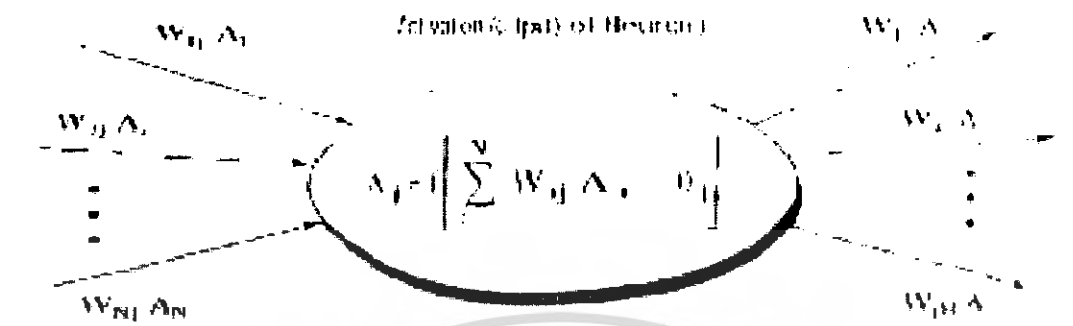
1.แอสโซซิแอทีฟแมปปิ้ง(associative mapping)ซึ่ง เน็ตเวิร์ค จะเรียนรู้ที่จะสร้างรูปแบบจำเพาะในชุดของอินพุตยูนิต และเมื่อใดที่รูปแบบ ที่คล้ายคลึงถูกป้อนเข้ามาในชุดของ อินพุตยูนิต ตัว แอสโซซิแอทีฟแมปปิ้งสามารถถูกแบ่งลงมาเป็นสองกลไกต่อไปนี้ได้

2.ออโต้แอสโซซิแอชั่น(auto-association): รูปแบบอินพุต จะเกี่ยวข้องกับตัวมันเองและสถานะของอินพุตและ เอาต์พุตยูนิต ที่เข้ากันสนิทพอดี ซึ่งจะถูกใช้ในการทำแพทเทิร์นคอมพลีชั่น(pattern completion), เช่น เพื่อสร้าง รูปแบบ(pattern) เมื่อใดก็ตามที่พบกลุ่มหรือส่วนหนึ่งของรูปแบบที่ไม่สมบูรณ์. หรือในกรณีที่สองเน็ตเวิร์คจะเก็บคู่ของแพทเทิร์นและสร้างความเกี่ยวข้องระหว่างสองเซตของแพทเทิร์น

เฮทเทอร์โรแอสโซซิแอชั่น(Hetero-association): จะเกี่ยวข้องกับกลไก สองแบบนี้:

- เนียร์เรสไนบอร์(Nearest-neighbor recall) ที่ รูปแบบเอาต์พุต จะถูกสร้างโดยตอบสนองต่อ อินพุตแพทเทิร์นที่เก็บไว้ซึ่งจะใกล้กับรูปแบบที่แสดงที่ใกล้ที่สุดและ อินเตอร์โพเลทีฟรี-คอล (Interpolative recall) ที่เอาต์พุตแพทเทิร์นที่มีการเติมโดยเป็นอิสระจากการคล้ายคลึงของรูปแบบการจัดเก็บโดยตอบสนองต่อ รูปแบบที่แสดงและอีกแบบหนึ่งซึ่งการแมปปิ้ง (Mapping) แบบเกี่ยวเนื่องกับตัวแปรจะถูกแบ่งประเภท เช่นเมื่อมีเซต จำเพาะของการจัดหมวดหมู่จะนำอินพุตยูนิตแพทเทิร์นไปแยกประเภท
- เรคกูลาริตี้ดีเทคชั่น(Regularity detection) ซึ่งยูนิตจะเรียนรู้ที่จะตอบสนองต่อคุณสมบัติของอินพุตแพทเทิร์น ด้วยเหตุแอสโซซิแอทีฟแมปปิ้งของ เน็ตเวิร์คจะจัดเก็บความสัมพันธ์ระหว่างรูปแบบซึ่งในการตรวจจับแบบปกติ การตอบสนองของแต่ละยูนิตจะมีความหมายจำเพาะกลไกการเรียนรู้แบบนี้สำคัญต่อการเรียนรู้ในอนาคตและการเผยแพร่ความรู้

ในทุกๆ โครงข่ายใยประสาทเทียม จะมีความรู้ซึ่งถูกบรรจุไว้ในรูปแบบของค่าของคอนเนคชั่นเวท (connections weight) การแก้ไขความรู้ที่เก็บไว้ใน เน็ตเวิร์คแบบฟังก์ชั่น ของประสบการณ์ หมายความให้เลินนิ่งรู(learning rule)เปลี่ยนแปลงค่าของเวท



รูปที่ 2-16 Weight matrix ของ neural network

ข้อมูลถูกเก็บไว้ในเวทเมตริกซ์ (Weight matrix) ของโครงข่ายใยประสาทเทียมการเรียนรู้คือการตัดสินใจของเวท ด้วยวิธีนี้จะเกิดการเรียนรู้, เราสามารถแบ่งได้เป็นสองรูปแบบหลักๆ ของโครงข่ายใยประสาทเทียมได้ดังนี้

1. ฟิกส์เน็ตเวิร์ค(fixed networks): ซึ่งไม่สามารถเปลี่ยนแปลงเวทได้
2. อแดปทีฟเน็ตเวิร์ค(adaptive networks) :ซึ่งสามารถเปลี่ยนแปลง เวทได้

การเรียนรู้ทั้งหมดที่ถูกใช้ใน โครงข่ายใยประสาทเทียมแบบปรับตัวได้นั้น สามารถจัดออกได้เป็นสองหมวดหมู่ใหญ่ๆ คือ

1. ซูเปอร์ไวซ์เลินนิ่ง(Supervised learning) ที่ต้องพึ่งการสอนจากภายนอก, ดังนั้นแต่ละเอาต์พุตยูนิต จะถูกบอกว่าให้ตอบสนองอย่างไรต่ออินพุตซิกแนล ระหว่างการเรียนรู้อาจต้องใช้โกลบอลอินฟอร์เมชั่น(global information) ด้วย รูปแบบของ ซูเปอร์ไวซ์เลินนิ่งนี้รวมถึง เออเรอคอเรคชั่นเลินนิ่ง(error-correction learning),เรนฟอร์ซเมนเลินนิ่ง(reinforcement learning) และสโตชาติกเลินนิ่ง(stochastic learning) เรื่องสำคัญเกี่ยวกับ ซูเปอร์ไวซ์เลินนิ่งคือปัญหาเรื่องจุดบรรจบของข้อผิดพลาด, เช่นการลดข้อผิดพลาดระหว่างผลที่ต้องการและค่าที่คำนวณได้ เป้าหมายคือการพิจารณาชุดของเวท ซึ่งลดค่าผิดพลาด วิธีที่รู้จักกันดีคือวิธี least mean square (LMS)

2. อันซูเปอร์ไวซ์เลินนิ่ง(Unsupervised learning) ไม่ใช้การเรียนรู้จากภายนอกและจะตัดสินใจจาก โลคอลอินฟอร์เมชั่น(Local Information) เท่านั้น บางครั้งก็ถูกเรียกว่าระบบแบบเซลฟ์ออแกไนเซชั่น(self-organization), เพราะตัวมันจัดข้อมูลที่มีใน เน็ตเวิร์คเอง และตรวจหาคอเลคทีฟพรอเพตี้(collective properties) ที่แสดงขึ้นมา วิธีอันซูเปอร์ไวซ์เลินนิ่งเช่น เฮปเบี้ยนเลินนิ่ง(Hebbian learning) และ คอมเพททิทีฟเลินนิ่ง(competitive learning) เราเรียกว่า โครงข่ายใยประสาทเทียมเรียนรู้แบบ ออฟไลน์เมื่อมีเลินนิ่งเฟส(learning phase) และ โอเปอเรชั่นเฟส

(operation phase) แยกกัน และ โครงข่ายใยประสาทเทียมจะเรียนรู้แบบออนไลน์ ถ้ามันเรียนรู้และทำงานไปพร้อมกัน โดยทั่วไปแล้วซูเปอร์ไวซ์เลินนิ่งจะเป็นแบบ ออฟไลน์, ส่วน อันซูเปอร์ไวซ์เลินนิ่งเป็นแบบออนไลน์

2.3.9 ทรานเฟอร์ฟังก์ชั่น (Transfer Function)

พฤติกรรมของ โครงข่ายใยประสาทเทียมนั้นขึ้นอยู่กับน้ำหนักและอินพุต –เอาต์พุต ฟังก์ชั่น (transfer function) ที่ถูกกำหนดให้กับยูนิต นั้นๆฟังก์ชั่น เหล่านี้ ถูกแยกออกได้เป็นสามหมวดดังนี้

1. ลิเนียยูนิต, เอาต์พุดแอกทิวิตี้(output activity) จะเป็นสัดส่วนกับเวทเอาต์พุต(weighted output) รวม
2. เทรสโฮลดยูนิต, เอาต์พุต จะถูกตั้งเป็นหนึ่งจากสองเลเวล ขึ้นอยู่กับว่าอินพุตรวมมากกว่าหรือน้อยกว่า เทรสโฮลดแวลู
3. ซิกมอยด์ยูนิต, เอาต์พุตจะแปรผันอย่างต่อเนื่องแต่ไม่ใช่เป็นลิเนีย กับการเปลี่ยนแปลงของ อินพุต, ซิกมอยด์ยูนิต จะมีความคล้ายคลึงกับ นิวรอล จริงมากกว่าลิเนีย หรือเทรสโฮลดยูนิต แต่ทั้งสามอันนั้นคิดได้แค่การประมาณอย่างคร่าวๆ

การทำให้ โครงข่ายใยประสาทเทียม สามารถทำงานจำเพาะได้นั้นเราต้องเลือกว่ายูนิตจะเชื่อมต่อกันอย่างไรและเราต้องตั้งค่าเวท ของการเชื่อมต่ออย่างถูกต้อง การเชื่อมต่อนี้จะตัดสินว่า ยูนิตแต่ละยูนิตจะมีผลซึ่งกันและกันได้หรือไม่ โดยน้ำหนักจะตัดสินค่าความสำคัญของการมีผลนั้น เราสามารถสอน ทรีเลเยอร์เน็ตเวิร์ค(Three-layer network) ให้ทำงานเฉพาะอย่างได้ด้วยวิธีต่อไปนี้

1. ให้เราใส่การเรียนรู้เบื้องต้นก่อนซึ่งประกอบไปด้วยรูปแบบของ แอกทิวิตี้(Activities) สำหรับ อินพุตยูนิต พร้อมๆ กับรูปแบบที่ต้องการของแอกทิวิตี้ สำหรับ เอาต์พุตยูนิต
2. เราตัดสินว่าจะให้ผลที่ออกมาจาก เอาต์พุตของเน็ตเวิร์คเข้ากับ เอาต์พุตที่ต้องการได้อย่างไร
3. เราเปลี่ยนน้ำหนักของแต่ละคอนเนคชั่น เพื่อให้เน็ตเวิร์คให้ผล เอาต์พุตที่เราต้องการได้ถูกต้องขึ้น

2.3.10 แบคพรอพพาเกชั่นอัลกอริทึม (The Back-Propagation Algorithm)

เพื่อจะฝึกให้ โครงข่ายใยประสาทเทียม ทำหน้าที่ได้นั้น เราต้องปรับค่าเวท ของแต่ละยูนิตให้ค่าความผิดพลาดระหว่าง เอาต์พุตที่ต้องการและ เอาต์พุตจริงนั้นน้อยที่สุด วิธีนี้ต้องให้ โครงข่ายใยประสาทเทียมคำนวณหาค่าเออเรอดีริเวทีฟ(error derivative)ของเวท (EW) หรือพูดอีกอย่างคือเราต้องการให้มันคำนวณหาว่าค่าความผิดพลาดเปลี่ยนแปลงอย่างไรเมื่อเพิ่มหรือลดเวท เล็กน้อย แบคพรอพพาเกชั่นอัลกอริทึมเป็นวิธีที่ใช้กันมากในการหาค่า EW.

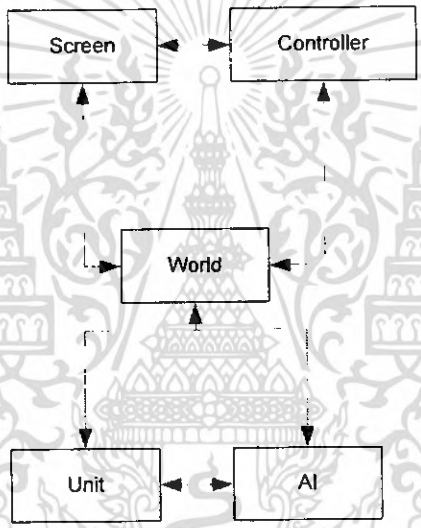
แบคพรอพพาเกชั่นอัลกอริทึมจะง่ายในการเข้าใจมากถ้า ยูนิตทั้งหมดใน เน็ตเวิร์คเป็นลิเนีย แบคอัลกอริทึมจะคำนวณแต่ละ EW โดยเริ่มต้นคำนวณหา EA, อัตราที่ความผิดพลาดเปลี่ยนเมื่อแอกทิวิตี้เลเวล(activity level)ของ ยูนิตเปลี่ยน สำหรับ เอาต์พุตยูนิต ค่า EA ก็คือผลต่างของเอาต์พุต ทั้งสอง ในการคำนวณ EA ของ ฮิดเดนยูนิตในเลเยอร์ ก่อนที่จะถึงเอาต์พุตเลเยอร์ เราต้องหาเวท ระหว่าง ฮิดเดนยูนิตและ เอาต์พุตยูนิต ที่เชื่อมต่อกันทั้งหมดก่อน แล้วคูณค่าเวทเหล่านั้นด้วย EA ของ เอาต์พุตยูนิต และบวกกับ โปรดัก(products) ผลนั้นจะเป็น EA ของ ฮิดเดนยูนิต ที่เลือก หลังจากคำนวณหา EA ในทุก ฮิดเดนเลเยอร์ ก่อนถึง เอาต์พุตเลเยอร์ เราก็จะสามารถใช้วิธีเดียวกันหา EA สำหรับเลเยอร์ อื่นๆได้, โดยการเคลื่อนจากเลเยอร์ไปเลเยอร์ในทิศทางตรงข้ามกับ activities ไปตามเน็ตเวิร์คทำให้มันถูกเรียกว่า แบคพรอพพาเกชั่นและเมื่อ EA ถูกคำนวณแล้วก็ง่ายที่จะคำนวณหาค่า EW สำหรับ unit ค่า EW เป็นผลของ EA และ แอกทิวิตี้จาก อินคัมมิ่งคอนเนคชั่น (incoming connection)

บทที่ 3

การออกแบบและพัฒนา

3.1 โครงสร้างโปรแกรม

ออกแบบให้แต่ละส่วนทำงานเสมือนพร้อมกัน โดยจัดการข้อมูลภายในตนเองและเรียกใช้จากส่วนอื่นที่ต้องการได้ เช่น ทุกส่วนสามารถแสดงข้อความผ่านจอภาพได้

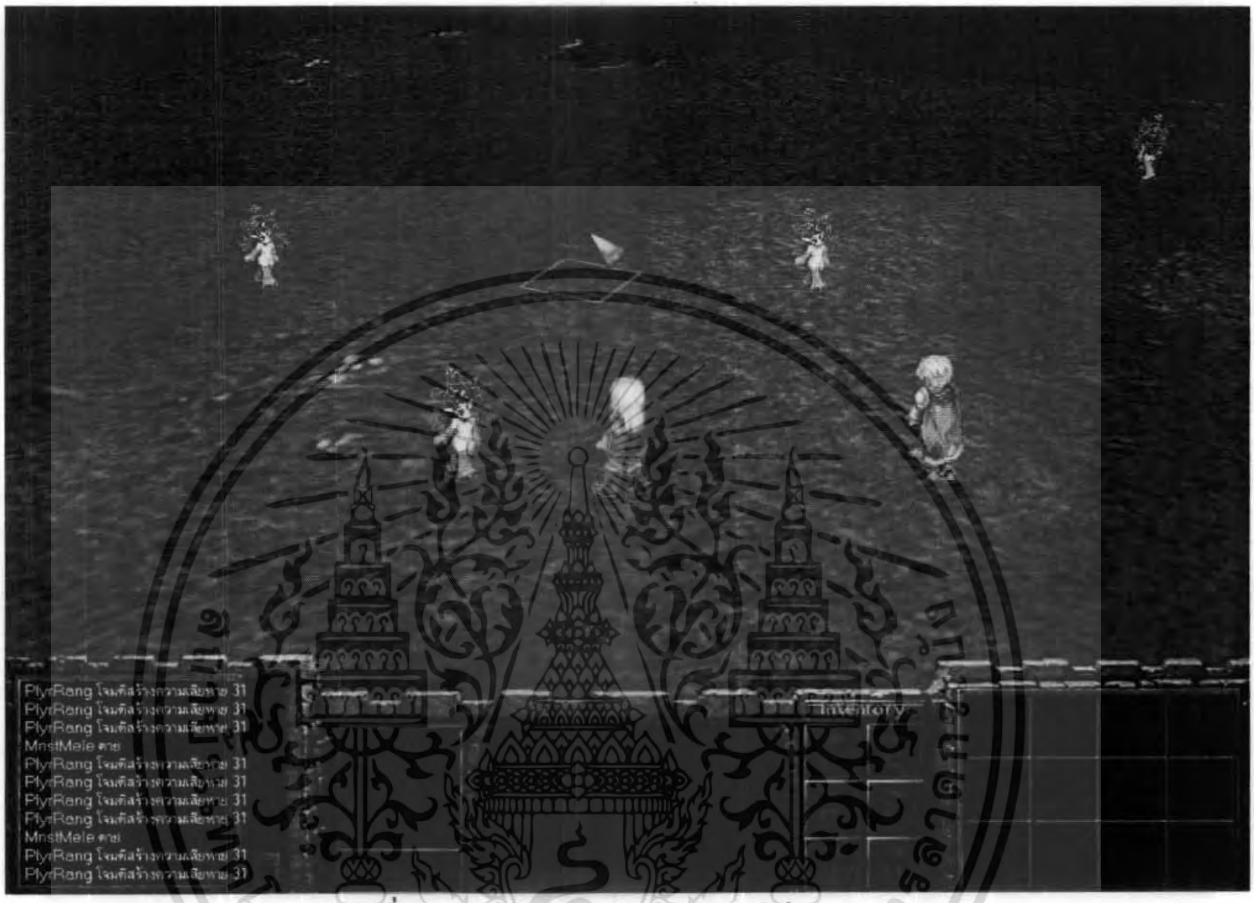


รูปที่ 3-1 class diagram

- จอภาพ (Screen)
  - เป็นส่วนแสดงภาพโดยใช้ไดเร็กต์ 3D (Direct3D)
  - ใช้การแสดงผลหลายแบบร่วมกันทั้ง 2 มิติและ 3 มิติ
  - รับค่าการเคลื่อนที่เมาส์จากอุปกรณ์ควบคุมแล้วเปลี่ยนแปลงตำแหน่งพอยเตอร์เอง
- อุปกรณ์ควบคุม (Controller)
  - เป็นส่วนรับการควบคุมจากอุปกรณ์โดยใช้ไดเร็กต์อินพุต (DirectInput)
  - รับค่าจากคีย์บอร์ด
  - รับค่าจากเมาส์
  - วนรอบรับค่าจากผู้เล่นและเก็บไว้ในบัฟเฟอร์สำหรับให้ส่วนอื่นนำไปใช้

- พื้นที่โลก (World)
    - เก็บรายละเอียดต่างๆ ที่อยู่ในระบบเกม เช่น ฉากที่ใช้, ตัวละคร เป็นต้น
    - ตัวละครเก็บเป็นชุดรวมกันทุกฝ่าย โดยจะเก็บตัวละครฝ่ายผู้เล่นไว้ที่ตำแหน่งแรกก่อน
    - เอไอเก็บเรียงกันตามชนิดของตัวละคร โดยเก็บแค่ตัวเดียวสำหรับให้ตัวละครทุกตัวที่มีชนิดนั้นใช้ร่วมกัน
    - เปลี่ยนฉากโดยการเก็บข้อมูลตัวละคร, สั่งให้เอไอเรียนรู้, เปลี่ยนข้อมูลภายในส่วนนี้และอ่านข้อมูลตัวละครและเอไอขึ้นมาใหม่
- ตัวละคร (Unit)
    - เก็บสถานะของตัวละคร เช่น พลังชีวิต, ชนิด, ระยะการโจมตี ซึ่งเป็นความแตกต่างระหว่างตัวละครแต่ละตัว โดยเก็บรายละเอียดที่พัฒนาขึ้นของตัวละครฝ่ายผู้เล่นไว้ในไฟล์เมื่อออกจากเกมไว้สำหรับเล่นต่อภายหลัง
    - เคลื่อนไหวไปตามคำสั่งจากผู้เล่นหรือเอไอ โดยผู้เล่นจะสั่งผ่านอุปกรณ์ควบคุมและเก็บไว้เป็นคำสั่งปัจจุบัน เช่น ตัวละครจะเดินไปจนถึงปลายทางในการกดเมาส์ครั้งเดียว ไม่จำเป็นต้องกดค้าง แต่ตัวละครอื่นที่ผู้เล่นไม่ได้เลือกอยู่รวมถึงศัตรูจะขอคำสั่งจากเอไอแทน
- เอไอ (AI)
    - ควบคุมตัวละครที่ผู้เล่นไม่ได้บังคับอยู่
    - ใช้นิวรอลเน็ตเวิร์คเป็นหลัก
    - แยกตามชนิดของตัวละคร

3.2 การแสดงภาพ



รูปที่ 3-2 ตัวอย่างการแสดงผลของเกมส์

ใช้วิธีการหลายอย่างในการแสดงผล แบ่งเป็น แผงควบคุม, พอยเตอร์, ตัวละคร, ฉาก และข้อความ

- แผงควบคุม
  - แสดงด้วยสไปรท์ (Sprite) เป็นภาพ 2 มิติที่วาดอยู่บนหน้าจอ
  - LPD3DXSPRITE เก็บขนาดสไปรท์
  - LPDIRECT3DTEXTURE9 เก็บภาพที่แสดงในสไปรท์
- พอยเตอร์
  - แสดงด้วยสไปรท์ วาดไว้ที่ตำแหน่งของเมาส์พอยเตอร์

- ตัวละคร
  - แสดงด้วยเวอร์เท็กซ์ (Vertex) เป็นแผ่นภาพ 2 มิติที่ตั้งอยู่บนแกนในระบบ 3 มิติ
  - LPDIRECT3DVERTEXBUFFER9 เก็บตำแหน่งมุมของเวอร์เท็กซ์ครึ่งแรก
  - LPDIRECT3DINDEXBUFFER9 เก็บส่วนที่เหลือของเวอร์เท็กซ์
  - LPDIRECT3DTEXTURE9 เก็บภาพที่แสดงในเวอร์เท็กซ์
- ฉาก
  - แสดงด้วยเมช (Mesh) เป็นโมเดล 3 มิติ
  - LPD3DXMESH เก็บโมเดล 3 มิติ
  - D3DMATERIAL9 เก็บวัตถุในโมเดล
  - LPDIRECT3DTEXTURE9 เก็บภาพที่แสดงในโมเดล
- ข้อความ
  - แสดงด้วยฟอนต์ (Font) สำหรับรายละเอียดเหตุการณ์ที่เกิดขึ้นในเกม
  - LPD3DXFONT แสดงตัวอักษร

3.3 การควบคุม

รับการควบคุมจากผู้เล่นผ่านอุปกรณ์ควบคุม คือ คีย์บอร์ดและเมาส์

- คีย์บอร์ด
  - อ่านค่าการกดปุ่มมาเก็บในบัฟเฟอร์
  - LPDIRECTINPUTDEVICE8 ติดต่อกับคีย์บอร์ด
  - BYTE[256] เก็บสถานะการกดปุ่ม
- เมาส์
  - อ่านค่าการกดปุ่มและเลื่อนมาเก็บในบัฟเฟอร์
  - LPDIRECTINPUTDEVICE8 ติดต่อกับเมาส์
  - DIMOUSESTATE2 เก็บสถานการณ์กดปุ่มและเลื่อน

ควบคุมตัวละครด้วยเมาส์เป็นหลัก และเพิ่มเติมการควบคุมอื่นๆ ด้วยคีย์บอร์ด

- เลือกตัวละคร
  - ด้วยการกดปุ่ม F1 – F4 สำหรับเลือกตัวละครของฝั่งผู้เล่นทั้ง 4 ตัว
  - หรือกด S แล้วคลิกซ้ายที่ตัวละครที่ต้องการ

- เดิน
  - คลิกซ้ายบนตำแหน่งที่ต้องการ โดยตำแหน่งนั้นจะต้องว่าง
  - หรือกด M แล้วคลิกที่ตำแหน่ง
- โจมตี
  - คลิกซ้ายบนตำแหน่งที่ต้องการ โดยตำแหน่งนั้นจะต้องมีตัวละครของศัตรูอยู่
  - หรือกด A แล้วคลิกซ้ายที่เป้าหมาย
- หมุนมุมมอง
  - คลิกขวาค้างไว้แล้วเลื่อนเมาส์

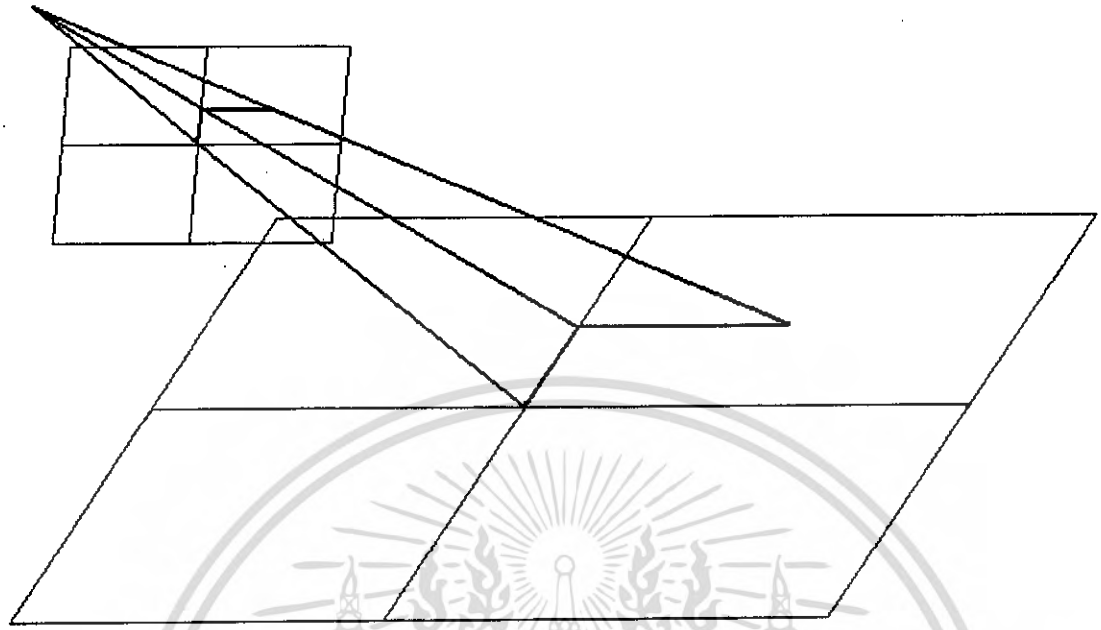
3.4 เอไอ

ใช้สำหรับควบคุมตัวละครที่ผู้เล่นไม่ได้บังคับอยู่และศัตรู

- ใช้นิวรอลเน็ตเวิร์คเป็นหลัก
- แบ่งตามชนิดของตัวละคร คือ ตัวละครแต่ละชนิดจะมีวิธีการเล่นที่เหมาะสมไม่เหมือนกัน และตัวละครหลายตัวที่มีชนิดเดียวกันสามารถใช้เอไอชุดเดียวกันได้
- เอไอเรียนรู้จากการควบคุมของผู้เล่นที่เคยควบคุมตัวละครชนิดนั้นไว้ โดยเก็บประวัติการควบคุมคู่กับสถานะตัวละครและข้อมูลในฉากรอบตัวละคร
- จังหวะการใช้งานของเอไอจะทำโดยตลอดให้เหมือนกับที่ผู้เล่นสั่งได้ตลอดเวลา

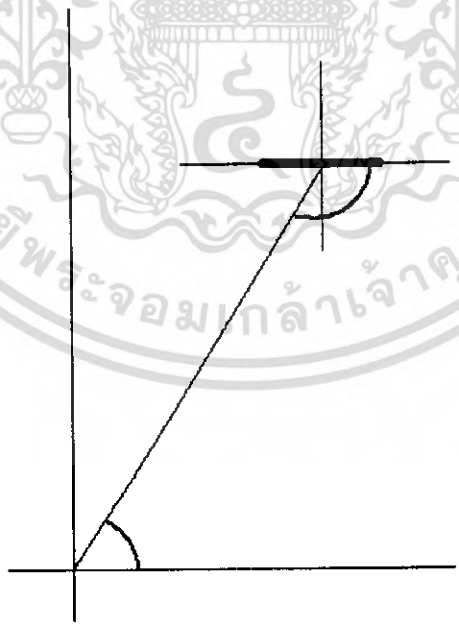
3.5 การคำนวณ

- พิกัดของพอยเตอร์ที่แสดงบนจอเป็นระบบ 2 มิติ เมื่อจะหาว่าชี้พิกัดไหนในระบบ 3 มิติ จะต้องแปลงดังรูป



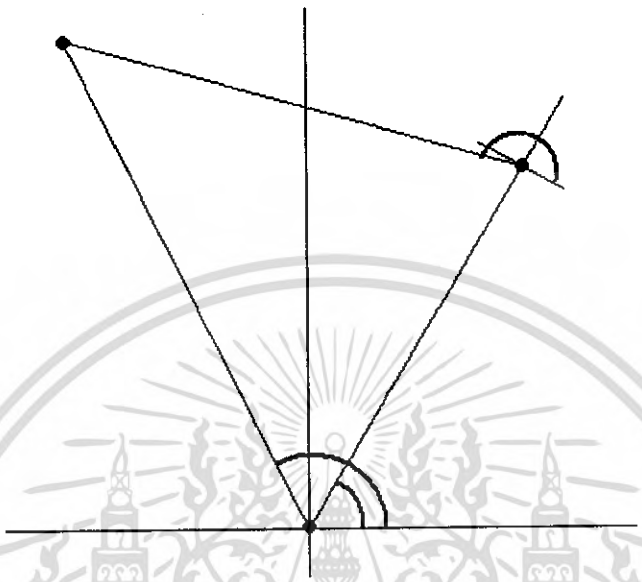
รูปที่ 3-3 แสดงกระบวนการเทียบพิกัดจาก 2 มิติไปสู่พิกัด 3 มิติ

- หมุนเวอร์เทกซ์ตัวละครให้หันมาขนานกับจอ เพื่อให้ผู้เล่นเห็นรูปชัดเจนและไม่ดูเป็นแผ่น



รูปที่ 3-4 แสดงการคำนวณมุมในการวางภาพ

- เปลี่ยนรูปตัวละครไปตามทิศทางที่ตัวละครหัน จะต้องคำนวณตามทิศของมุมมองผู้เล่น



รูปที่ 3-5 แสดงการคำนวณมุมเพื่อเปลี่ยนภาพตามทิศทางที่ตัวละครหัน

บทที่ 4

การทดลองและผลการทดลอง

4.1 ชนิดตัวละคร

แบ่งเป็น 2 ชนิด ตัวละครโจมตีระยะใกล้ และ ตัวละครโจมตีระยะไกล โดยจะใช้เอไอคนละชุดกัน เนื่องจากวิธีการเล่นที่เหมาะสมไม่เหมือนกัน

4.2 ข้อมูลผู้เล่น

เก็บข้อมูลการควบคุมผู้เล่น เพื่อนำไปใช้เป็นเทรนนิ่งเซตสำหรับให้นิวรอลเน็ตเวิร์คเรียนรู้ โดยจะเก็บการควบคุมทุกครั้งที่มีคำสั่งที่สมบูรณ์ เช่น สั่งเดินไปที่พิกัดใดหลังจากคลิกเมาส์ซ้ายแล้ว

นำไปใช้เรียนรู้หลังจากที่ผู้เล่นเปลี่ยนฉากหรือออกจากเกม และอ่านกลับเข้ามาเมื่อเริ่มฉากใหม่ หมายความว่าเอไอจะเปลี่ยนแปลงทุกครั้งที่ขึ้นฉากใหม่ทั้งฝ่ายเราและศัตรู

4.3 การใช้เอไอ

กำหนดรูปแบบอินพุตและเอาต์พุตของนิวรอลเน็ตเวิร์ค

- กำหนดอินพุต ให้นิวรอลเน็ตเวิร์คส่วนเลือกการกระทำ
  - กำหนดให้อินพุตคือข้อมูลของสถานะการณ์รอบตัวที่จำเป็นต่อการตัดสินใจของเอไอ ชนิดของวัตถุรอบตัว ขอบเขต 7 x 7 ช่อง ไม่นับตัวเอง กำหนดให้เป็นค่าตัวเลข 3 ระดับ
    0 คือ ช่องว่างหรือวัตถุที่ไม่มีผลต่อการต่อสู้
    +1 คือ เพื่อน
    -1 คือ ศัตรู
  - การกระทำปัจจุบันของตัวละคร 2 ค่า เดิน, โจมตี
  - จัดเรียงเป็นอินพุตได้ 50 ค่า
- กำหนดเอาต์พุต ให้นิวรอลเน็ตเวิร์คส่วนเลือกการกระทำ
  - กำหนดให้เอาต์พุตของนิวรอลเน็ตเวิร์คแปลเป็นคำสั่งให้ตัวละคร
  - ประเภทของคำสั่ง 2 ค่า เดิน, โจมตี ค่าใดมากกว่าถือเป็นการตัดสินใจเลือกคำสั่งนั้น
  - จัดเรียงเป็นเอาต์พุตได้ 2 ค่า

ถ้าการกระทำที่เอไอตัดสินใจจำเป็นต้องมีพิกัดด้วยให้ใช้นิวรอลเน็ตเวิร์คอีกส่วนสำหรับเลือกพิกัด

- กำหนดอินพุต ให้นิวรอลเน็ตเวิร์คส่วนเลือกพิกัด
  - กำหนดให้อินพุตคือข้อมูลของสถานการณ์รอบตัวที่จำเป็นต่อการตัดสินใจของเอไอเหมือนส่วนการกระทำ
  - การกระทำที่เลือก 2 ค่า เดิน, โจมตี
  - จัดเรียงเป็นอินพุตได้ 50 ค่า

- กำหนดเอาต์พุต ให้นิวรอลเน็ตเวิร์คส่วนเลือกการกระทำ
  - กำหนดให้เอาต์พุตของนิวรอลเน็ตเวิร์คแปลเป็นพิกัดอ้างอิงกับพิกัดของตัวละคร 2 แกน
  - จัดเรียงเป็นเอาต์พุตได้ 2 ค่า

4.4 ผลที่ได้

- เอไอที่เรียนรู้แล้วมีความแตกต่างกับเมื่อเริ่มแรก แต่ยังไม่เห็นถึงความฉลาดที่ชัดเจนนัก โดย เมื่อเริ่มสร้างตัวละครจะเดินสับสนไปมา
- เมื่อทดลองใส่ตัวละครที่ใช้นิวรอลเน็ตเวิร์คประมาณ 4 ตัวขึ้นไป พบว่าความเร็วของเกมลดลงอย่างเห็นได้ชัด อาจแก้ปัญหาได้โดยวิธี เช่น
  - ให้ทำงานเป็นเธรด (thread) เพื่อให้การแสดงผลไม่ต้องรอจนคำนวณเสร็จ
  - เว้นช่วงการใช้นิวรอลเน็ตเวิร์ค
  - เขียนโค้ดนิวรอลเน็ตเวิร์คใหม่โดยเน้นไปที่ความเร็วในการคำนวณ

บทที่ 5
บทสรุปและวิจารณ์

5.1 บทสรุป

ในการพัฒนาโครงงานนี้นั้นยังไม่ถือว่าบรรลุได้ตามจุดประสงค์ที่ตั้งไว้ทั้งหมด ทั้งส่วนของกราฟิก 3 มิติ ระบบเกม รวมทั้งการประยุกต์ใช้นิวรอลเน็ตเวิร์คด้วย

แต่อย่างไรก็ตาม โครงงานนี้ได้บรรลุตามจุดประสงค์ที่ตั้งไว้ในเบื้องต้น คือ การเข้าใจในหลักการเขียนกราฟิก 3 มิติ การประยุกต์นำกราฟิก 3 มิติมาใช้ในเกม และการนำนิวรอลเน็ตเวิร์คมาใช้ในเกม

5.2 ปัญหาที่พบในการพัฒนา

- เอไอที่เขียนขึ้นทำงานช้าไม่เป็นไปตามที่คาดการไว้เนื่องจากการบันทึกข้อมูลของผู้เล่นแล้วใช้เป็น Training set ให้กับนิวรอลเน็ตเวิร์กโดยตรงไม่เหมาะสมกับปัญหานี้
- เกิดปัญหาด้านกราฟฟิกแสดงผลผิดพลาดเล็กน้อยเนื่องจากวิธีการใช้คลาสของไดเร็กเอ็กซ์มีวิธีการใช้บางอย่างที่ค่อนข้างซับซ้อน
- โครงสร้างโปรแกรมมีการแก้ไขบ่อยเนื่องจากในบางขั้นตอนไม่สามารถเขียนโค้ดให้ได้ตามที่ออกแบบไว้

5.3 ข้อเสนอแนะ

- อาจแก้ปัญหาการกราฟิกด้วยการสร้างลำดับการแสดงผลสไปรต์
- อาจแก้ปัญหาความเร็วเอไอได้โดย
  - ให้ทำงานเป็นเธรด (thread) เพื่อให้การแสดงผลไม่ต้องรอจนคำนวณเสร็จ
  - เว้นช่วงการใช้นิวรอลเน็ตเวิร์ค
  - เขียนโค้ดนิวรอลเน็ตเวิร์คใหม่โดยเน้นไปที่ความเร็วในการคำนวณ

5.4 แนวทางการพัฒนาต่อ

ในการพัฒนาโครงงานต่อไปควรพัฒนาให้เอไอตอบสนองเร็วขึ้นกว่านี้ และพัฒนากราฟิกให้สวยงามขึ้น โดยการใช้โมเดล 3 มิติแสดงตัวละครแทนสไปรต์ เก็บรายละเอียดและขยายพื้นที่ของฉากให้มากขึ้น ในด้านระบบเกมเพิ่มฟังก์ชันให้หลากหลายขึ้นเพื่อเพิ่มอรรถรสในเกมมากขึ้น

บรรณานุกรม

[1] Alan Thorn : DirectX® 9 Graphics:The Definitive Guide to Direct3D®

[2] An introduction to neural computing. Aleksander, I. and Morton, H. 2nd edition

[3] Industrial Applications of Neural Networks (research reports Esprit, I.F.Croall, J.P.Mason)

[4] An Introduction to Computing with Neural Nets (Richard P. Lipmann, IEEE ASSP Magazine, April 1987)

[5] Alkon, D.L 1989, Memory Storage and Neural Systems, Scientific American, July, 42-50

[6] Learning internal representations by error propagation by Rumelhart, Hinton and Williams (1986).

[7] Neural computers, NATO ASI series, Editors: Rolf Eckmiller Christoph v. d. Malsburg

[8] Neural Networks, Eric Davalo and Patrick Naim.

เว็ปไซต์อ้างอิง

http://www.gamedev.net

http://www.thaigamedevx.com

http://www.sourceforge.net

http://www.lancet.mit.edu.ga

http://www.emsl.pnl.gov/

ภาคผนวก

ภาคผนวก

ก. ความต้องการของระบบ

ก.1 ฮาร์ดแวร์ที่ต้องใช้งาน

รายการฮาร์ดแวร์	รายละเอียด	คำแนะนำ
โปรเซสเซอร์ (Processor)	โปรเซสเซอร์ 80486 ขึ้นไป	Pentium อย่างต่ำ
ฮาร์ดดิสก์ (Harddisk)	1 GB ขึ้นไป	1.5 GB ขึ้นไป
หน่วยความจำหลัก (RAM)	128 MB ขึ้นไป	256 MB ขึ้นไป
การ์ดแสดงผล (Graphic Card)	สนับสนุนการทำงาน ดังนี้ - Vertex Shader Version 1.1 ขึ้นไป - Max VertexBlend Matrix Index 12 ขึ้นไป - Hardware Transform And Light (geForce3 หรือที่สูงกว่านี้)	สนับสนุน Pixcel Shader Version 2.0 ขึ้นไป (Radeon 9700 หรือที่สูงกว่านี้)

ก.2 ซอร์ฟแวร์ที่ต้องใช้งาน

รายการซอร์ฟแวร์	รายละเอียด
.NET Runtime	Version 1.1
DirectX	Version 9.0c
Direct SDK	Version 9.0 และสำหรับ C#
โปรแกรมเกม	TestAnimation

ข. วิธีการติดตั้ง

1. ติดตั้ง .NET Runtime
2. ติดตั้ง DirectX 9.0c
3. ติดตั้ง DirectX SDK
4. คัดลอกไฟล์ในโฟล์เดอร์ Execute ลงในฮาร์ดดิสก์