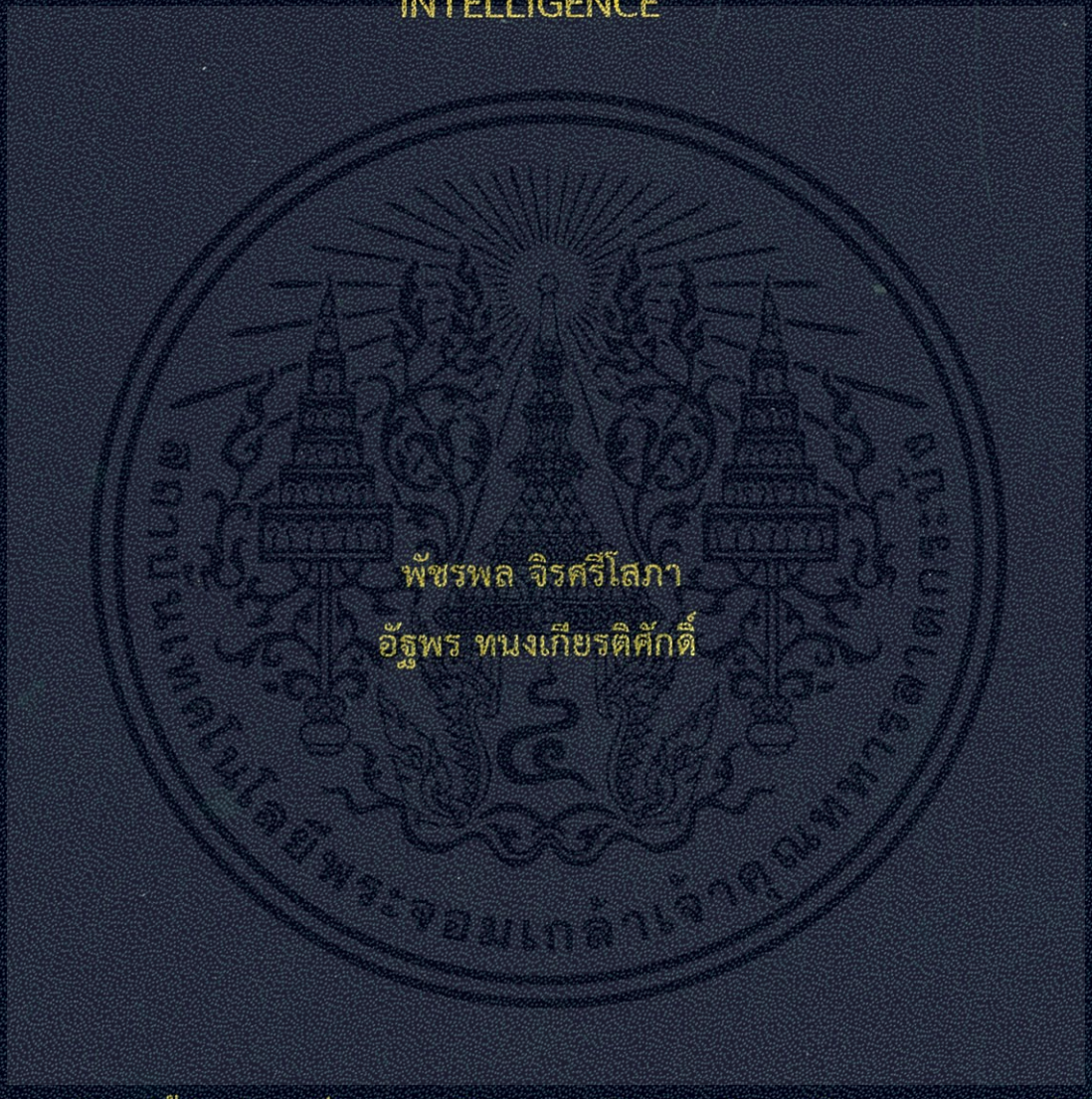
การพัฒนาเกมที่มีสภาพแวดล้อมที่เปลี่ยนแปลงตลอดเวลาด้วยปัญญาประดิษฐ์

DEVELOPING DYNAMIC ENVIRONMENT GAME BY ARTIFICIAL INTELLIGENCE



พัชรพล จิรศรีโลภา
อัฐพร ทนงเกียรติศักดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

การพัฒนาเกมที่มีสภาพแวดล้อมที่เปลี่ยนแปลงตลอดเวลาด้วยปัญญาประดิษฐ์

DEVELOPING DYNAMIC ENVIRONMENT GAME BY ARTIFICIAL INTELLIGENCE

พัชรพล จิรศรีโสภา
อัฐพร ทนงเกียรติศักดิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์

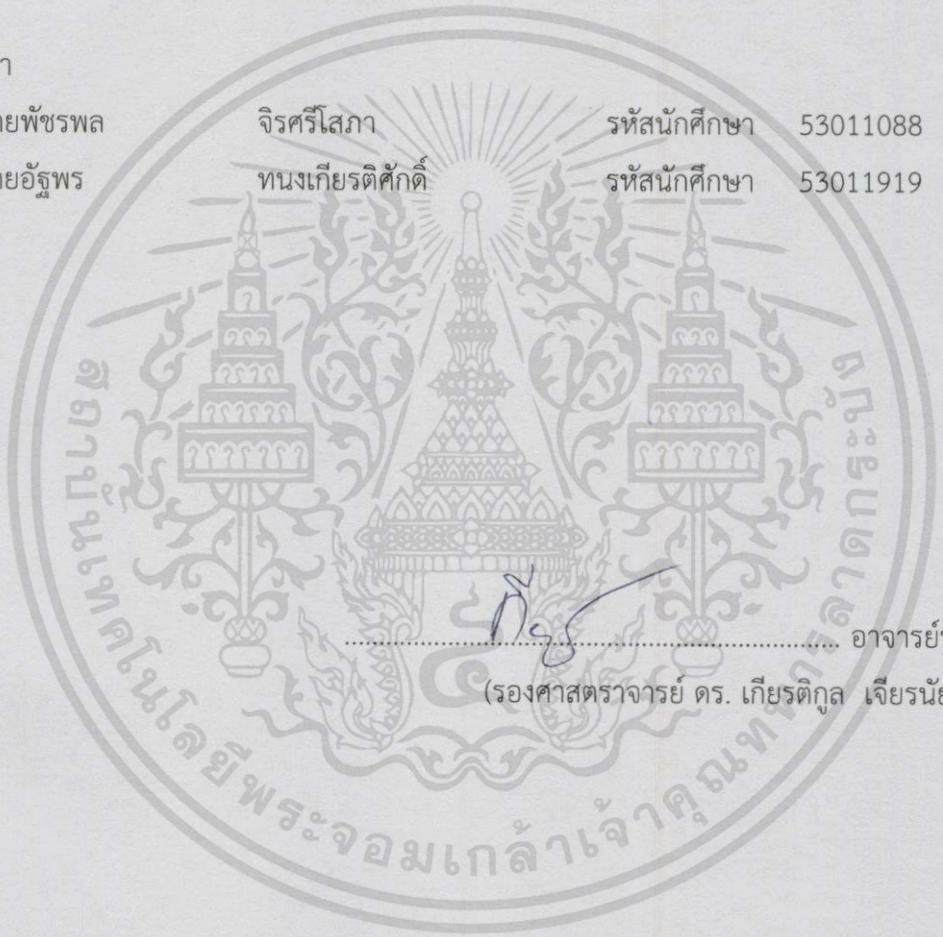
คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเกมที่มีสภาพแวดล้อมที่เปลี่ยนแปลงตลอดเวลาด้วยปัญญาประดิษฐ์

DEVELOPING DYNAMIC ENVIRONMENT GAME BY ARTIFICIAL INTELLIGENCE

ผู้จัดทำ

1. นายพัชรพล	จิรศรีโสภา	รหัสนักศึกษา	53011088
2. นายอัฐพร	ทนงเกียรติศักดิ์	รหัสนักศึกษา	53011919



..... อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ ดร. เกียรติกูล เจียรนัยธนะกิจ)

การพัฒนาเกมที่มีสภาพแวดล้อมที่เปลี่ยนแปลงตลอดเวลาด้วยปัญญาประดิษฐ์

นาย พัชรพล	จิรศรีโสภา	53011088
นาย อัฐพร	ทนงเกียรติศักดิ์	53011919
รศ.ดร. เกียรติกูล	เจียรนัยธนะกิจ	อาจารย์ที่ปรึกษา

ปีการศึกษา 2556

บทคัดย่อ

ในปัจจุบันเกมแอคชั่นแบบผู้เล่นหลายคน (MultiplayerAction Game) เป็นเกมประเภทที่ได้รับความนิยมอย่างแพร่หลาย แต่เกมเหล่านี้มักเน้นไปที่การต่อสู้ระหว่างผู้เล่น ทำให้ระบบปัญญาประดิษฐ์ในเกมแบบผู้เล่นหลายคนไม่ได้รับการพัฒนาเท่าที่ควรทางผู้จัดทำจึงเริ่มโครงการนี้ขึ้นเพื่อทดลองสร้างเกม Mahayuth Battleground ซึ่งเป็นเกมแบบผู้เล่นหลายคน โดยออกแบบให้สภาพแวดล้อมในเกมตอบสนองต่อพฤติกรรมของผู้เล่นผ่านตัวละครที่ควบคุมโดยคอมพิวเตอร์ โดยการนำหลักการของโครงข่ายประสาทเทียม (Artificial Neural Network)มาวิเคราะห์พฤติกรรมของผู้เล่นในเกมโดยอ้างอิงจากข้อมูลการเล่นจึงทำให้ความสามารถของตัวละครของฝั่งคอมพิวเตอร์มีการพัฒนาขึ้นเรื่อยๆและส่งผลต่อสภาพแวดล้อมการเล่นเกมของผู้เล่น

DEVELOPING DYNAMIC ENVIORNMENT GAME BY ARTIFICIAL INTELLIGENCE

Mr. Patcharapol	Jirasrisopa	53011088
Mr. Atthaporn	Thanongkiatisak	53011919
Assoc. Prof. Dr.Kietikul	Jearanaitanakij	Advisor

Academic Year 2013

ABSTRACT

Nowadays, Multiplayer Action Gameis very popular among gamers. But these games are mostly developed for players to compete with one another, which leave overall AI in multiplayer action games not developed enough.For these reasons, we embark on a new project to create "Mahayuth Battleground", which is a Multiplayer Action Game. We've tried to integrate Artificial Neural Network (ANN) into the game to analyze player's actions and thenchange its actionbase on player behavior. By using this technique, computer-controlled enemy will developingdynamically to player behavior.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงไปด้วยดี จากการที่ได้รับการช่วยเหลืออย่างดียิ่งจากรองศาสตราจารย์ ดร. เกียรติกูล เจียรนัยธนะกิจ ซึ่งเป็นอาจารย์ที่ปรึกษาโครงงาน ผู้ซึ่งได้เสียสละเวลาในการตรวจสอบความคืบหน้าทุกสัปดาห์ อีกทั้งยังให้คำแนะนำ ชี้แนะแนวทางทำงาน ข้อควรปรับปรุง และให้คำปรึกษาที่ดีในเรื่องต่างๆตลอดมาจนทำให้ปริญญานิพนธ์ฉบับนี้สำเร็จลุล่วงได้

ขอขอบพระคุณอาจารย์ทุกท่านที่ได้อบรมสั่งสอน และให้ความรู้เปรียบเสมือนบิดามารดาที่สองของข้าพเจ้า อีกทั้งขอขอบคุณรุ่นพี่และเพื่อนๆ ทุกคนที่ช่วยให้คำแนะนำ และพร้อมแลกเปลี่ยนความคิดเห็น ช่วยเหลือในด้านต่างๆ

นอกจากบุคคลที่ได้กล่าวมาข้างต้นแล้ว ต้องขอขอบพระคุณบิดา มารดา และครอบครัวของข้าพเจ้าที่ให้การอบรมสั่งสอน เลี้ยงดู ให้กำลังใจ ให้โอกาสทางการศึกษา และที่สำคัญคือให้การสนับสนุนข้าพเจ้าในด้านต่างๆอย่างเสมอมา

สุดท้ายขอขอบพระคุณสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ให้โอกาสข้าพเจ้าในการเข้ามาศึกษา รวมถึงให้โอกาสในการเข้าใช้เครื่องมือและทรัพยากรต่างๆ ซึ่งเป็นส่วนหนึ่งที่ทำให้โครงงานชิ้นนี้เสร็จลุล่วงไปได้ด้วยดี

คุณค่า คุณความดีและประโยชน์ใดๆ อันพึงได้จากการทำโครงงานนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกๆท่าน

นาย พัชรพล จิรศรีโสภา

นาย อัฐพร ทนงเกียรติศักดิ์

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

เนื่องจากในปัจจุบัน การใช้งานคอมพิวเตอร์ นอกจากจะใช้เพื่อทำการศึกษาค้นคว้าแล้ว การนำมาใช้เพื่อความบันเทิงก็เริ่มแพร่หลายมากขึ้น โดยเฉพาะตลาดเกมแนวสามมิติแบบผู้เล่นหลายคน (Multiplayer 3D Game) เป็นที่รู้จักกันดีและมีกลุ่มลูกค้าที่กว้าง ทำให้มีการสร้างเกมแบบผู้เล่นหลายคนเกิดขึ้นมาตลอด แต่ข้อเสียที่เห็นได้ชัดจากเกมแนวนี้ก็คือการดำเนินเรื่องหรือการปรับเปลี่ยนสภาพแวดล้อมในเกมเหล่านั้นมักจะไม่ได้อ้างอิงข้อมูลจากผู้เล่น ทำให้ผู้เล่นอาจรู้สึกเบื่อต่อสภาพแวดล้อมเดิม ส่งผลให้ผู้พัฒนาเกมต้องสร้างเนื้อเรื่องใหม่ สร้างตัวละครใหม่ หรือแม้กระทั่งสร้างเกมใหม่เพื่อรักษาระดับของผู้เล่น และไม่ให้ผู้เล่นเบื่อเกมเก่า ทางผู้จัดทำจึงเริ่มโครงการนี้ขึ้นเพื่อสร้างสภาพแวดล้อมซึ่งในที่นี้จะเน้นไปที่การต่อสู้กับทหารศัตรูที่มีพฤติกรรมที่เปลี่ยนแปลงตลอดเวลา โดยการนำหลักการของปัญญาประดิษฐ์ในเรื่องโครงข่ายประสาทเทียม (Artificial Neural Network) เข้ามาประยุกต์ เพื่อทำให้การเล่นทุกของผู้เล่นแต่ละครั้งมีความยืดหยุ่นตามสถานการณ์จากผู้เล่น

1.2 วัตถุประสงค์ของโครงงาน

1) เพื่อศึกษา ออกแบบ และสร้างเกมประเภทเกมแอ็คชั่นที่มีผู้เล่นหลายคน (Multiplayer Action Game) ให้เป็นแนวทางและเป็นเกมต้นแบบ ที่นำความรู้เกี่ยวกับปัญญาประดิษฐ์เข้ามาประยุกต์ในเกม
2) เพื่อศึกษา ออกแบบ และสร้างโครงข่ายประสาทเทียม (Artificial Neural Network) ให้สามารถจดจำและเรียนรู้ข้อมูลผลการเล่นที่เกิดจากพฤติกรรมของผู้เล่น เพื่อนำมาปรับเปลี่ยนสภาพแวดล้อมตามข้อมูลที่ได้จากผู้เล่นได้

1.3 ขอบเขตของโครงงาน

โครงการนี้มีเป้าหมายเพื่อศึกษาและออกแบบโครงข่ายประสาทเทียม พัฒนาเกมผู้เล่นหลายคนแบบสามมิติ และทั้งสองอย่างมาประยุกต์เข้าด้วยกัน เพื่อให้เกิดสภาพแวดล้อมในเกมที่มีการเปลี่ยนแปลงอยู่ตลอด โดยมีขอบเขตการศึกษา และวิจัยดังต่อไปนี้

1) การศึกษาการใช้ Unreal Development Kit ในการพัฒนาตัวเกม และรองรับบนระบบปฏิบัติการ Microsoft Windows 7 และ Microsoft Windows 8

2) การศึกษาการใช้และออกแบบ โครงข่ายประสาทเทียม ที่มีความเหมาะสมต่อการนำมาใช้ในการเรียนรู้ เพื่อการสร้างสภาพแวดล้อมในเกมที่เปลี่ยนแปลง
3) ในส่วนการสร้างสภาพแวดล้อมในเกมที่เปลี่ยนแปลง ต้องมีช่วงเวลาให้มีการเรียนรู้พฤติกรรมของผู้เล่นก่อน

1.4 วิธีการดำเนินการ

1) ศึกษาความรู้เกี่ยวกับปัญญาประดิษฐ์ ซึ่งนำมาใช้ในการจดจำและวิเคราะห์พฤติกรรมการเล่นของผู้เล่น
2) ศึกษาการพัฒนาเกมด้วย Unreal Development Kit
3) จัดหาและติดตั้งทรัพยากรที่จำเป็นต้องใช้ในการพัฒนา
4) ออกแบบเนื้อเรื่อง ตัวละคร ฉาก และพื้นผิวที่ใช้ในเกม
5) วิเคราะห์ และออกแบบโครงสร้างและระบบเกม
6) วิเคราะห์ และออกแบบโครงข่ายประสาทเทียม
7) พัฒนาตัวเกมฝั่งผู้เล่น ตัวโปรแกรมจัดการเกมออนไลน์ และโปรแกรมซึ่งมีหน้าที่ตัดสินใจและควบคุมการเปลี่ยนแปลงสภาพแวดล้อมในเกมโดยใช้หลักการของโครงข่ายประสาทเทียมเข้ามาประยุกต์
8) เก็บผลการทดลองจากการเล่นเกม นำมาให้โครงข่ายประสาทเทียมเรียนรู้
9) วิเคราะห์ผลของโครงข่ายประสาทเทียมที่ได้จากการเรียนรู้ และแก้ไขส่วนที่ผิดพลาด
10) นำโครงข่ายประสาทเทียมที่ได้ไปผนวกกับระบบเกมเพื่อเป็นส่วนในการตัดสินของเกม
11) ทดลองเล่นเกมและเก็บข้อมูล
12) นำมาสรุปวิเคราะห์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) ได้รับความรู้ความเข้าใจรวมถึงกระบวนการพัฒนาโปรแกรมด้วยปัญญาประดิษฐ์
2) ได้รับความรู้ความเข้าใจเกี่ยวกับการออกแบบและจัดการเกมที่มีผู้เล่นหลายคน
3) ได้รับความรู้ความเข้าใจเกี่ยวกับการพัฒนาเกมด้วย Unreal Development Kit
4) เพื่อเป็นแนวทางให้แก่เกมต่างๆในปัจจุบัน ในการพัฒนาเกมเพื่อให้ผู้เล่นหรือผู้ใช้งานคอมพิวเตอร์เพื่อความบันเทิงทางด้านการเล่นเกม ได้สนุกกับการเล่นเกมที่มีการเปลี่ยนแปลงของสภาพแวดล้อมอยู่ตลอดเวลา

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง กล่าวถึงทฤษฎีต่างๆที่ใช้ในการพัฒนาโครงการ ประกอบด้วย ทฤษฎีที่เกี่ยวข้องกับโครงข่ายประสาทเทียม ทฤษฎีการส่งข้อมูลผ่านเครือข่าย และข้อมูลเกี่ยวกับการโปรแกรมที่ใช้ในการพัฒนาโครงการ

บทที่ 3 การออกแบบและพัฒนา กล่าวถึงรายละเอียดของโครงงานนี้ ส่วนประกอบต่างๆที่ได้ออกแบบและพัฒนาขึ้น การทำงานของระบบ รวมถึงการออกแบบเนื้อเรื่อง ตัวละคร ฉาก และกราฟิกอื่นๆด้วย

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึงการเตรียมการทดลองทั้งการจัดเตรียมส่วนฮาร์ดแวร์ ส่วนซอฟต์แวร์ สภาวะแวดล้อมในการทำการทดลอง ข้อมูลทดลองหรือทดสอบ การทำงานหรือการจำลองการทำงานของระบบ ผลการทดลอง ค่าสมรรถนะของระบบ การวัดประสิทธิภาพของระบบ และการวิเคราะห์ผลการทดลองหรือผลการทำงาน

บทที่ 5 บทสรุป กล่าวถึงบทสรุปของโครงงาน รวมถึงปัญหาอุปสรรคต่างๆ ของโครงงาน และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

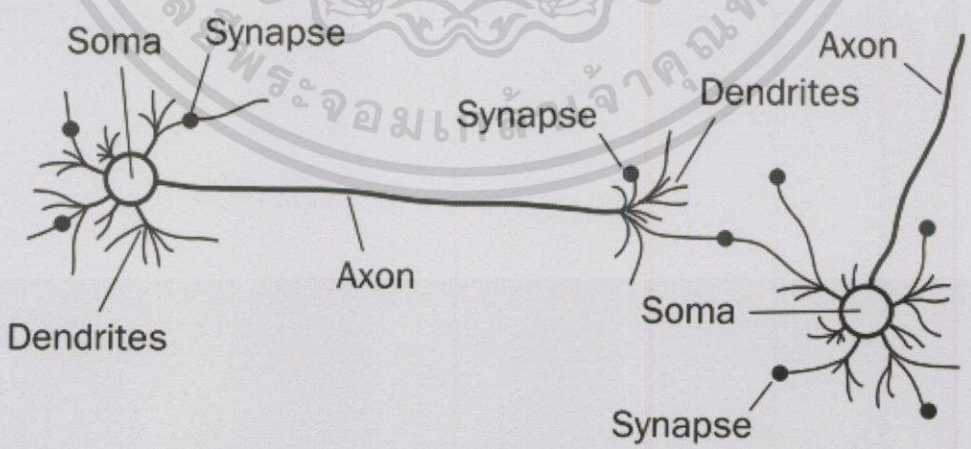
2.1 โครงข่ายประสาทเทียม (Artificial Neural Network)

2.1.1 โครงข่ายหน่วยประสาท (Neural network)

โครงข่ายหน่วยประสาทพบในสมองมนุษย์ เกิดจากการเชื่อมโยงกันของเซลล์ประสาท (Neuron) จำนวนมากและมีความซับซ้อนสูง เพื่อทำหน้าที่ในการจดจำ และทำการนำความรู้ที่ได้จากการจดจำนำมาตัดสินใจในการดำเนินการในสิ่งที่ต้องการ

สำหรับเซลล์ประสาทของมนุษย์ประกอบด้วย 4 ส่วนดังนี้คือ

1) โซมา (Soma) เป็นศูนย์กลางของเซลล์ประสาทที่ประสานงานกับส่วนประกอบอื่นๆในเซลล์ประสาท
2) เดนไดรซ์ (Dendrites) เป็นรยางค์ของเซลล์ประสาท แตกแขนงออกมาจากโซมา คล้ายรากไม้ เป็นส่วนที่รับกระแสประสาทเข้าสู่ตัวเซลล์ประสาท
3) แอกซอน (Axon) เป็นส่วนที่ช่วยถ่ายทอดและนำส่งสัญญาประสาทไปยังเซลล์ประสาทตัวอื่นๆ มีลักษณะเป็นแขนงยาวยื่นออกจากตัวเซลล์ประสาท
4) ไซแนปซิส (Synapses) เป็นจุดประสานประสาท ทำหน้าที่เชื่อมต่อเพื่อการติดต่อสื่อสารกับเซลล์ประสาทอื่น



รูปที่ 2.1 โครงสร้างของเซลล์ประสาทในมนุษย์

2.1.2 โครงข่ายประสาทเทียม (Artificial Neural Network)

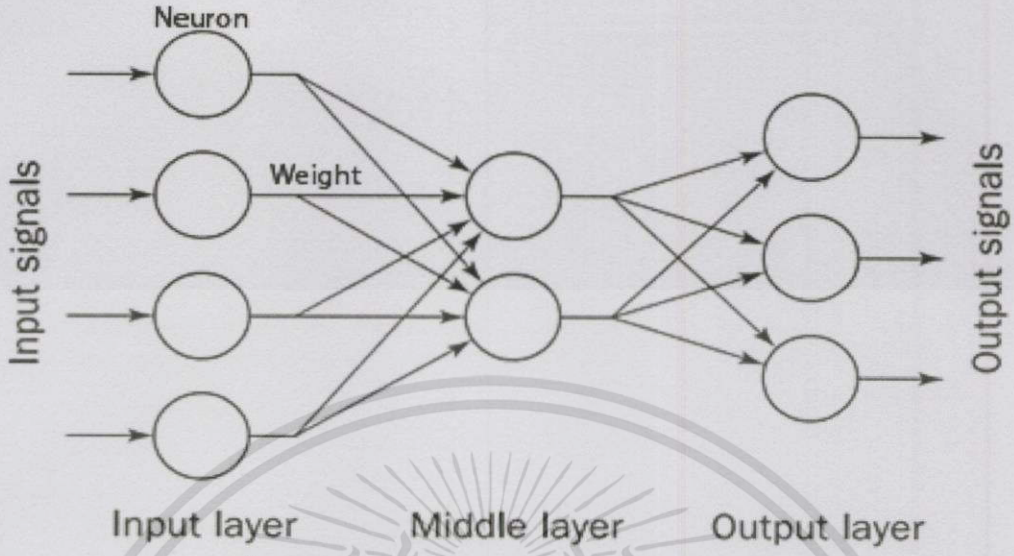
จากโครงข่ายหน่วยประสาท เราสามารถนำแนวคิดในทางชีววิทยา มาสร้างโครงข่ายประสาทเทียม เพื่อเลียนแบบลักษณะการทำงานของโครงข่ายหน่วยประสาทได้กล่าวคือ การที่เดนไดรซ์มีหน้าที่นำกระแสประสาทเข้าสู่ตัวเซลล์ประสาท ก็เปรียบเสมือนข้อมูลที่เราต้องการนำมาประมวลผล จากนั้นก็จะส่งข้อมูลต่อไปยังโซมา ก็เปรียบเสมือนเป็นส่วนที่ประมวลผลโดยดูจากส่วนประกอบรอบๆตัว และส่งข้อมูลออกไปผ่านทางแอกซอนเพื่อประมวผลเป็นผลลัพธ์ต่อไป โดยในขั้นตอนนี้ไซแนปซิสที่เป็นจุดประสานประสาท จะทำหน้าที่ส่งผลลัพธ์นี้ให้กับเซลล์ประสาทอื่นๆที่เชื่อมต่ออยู่ด้วย เพื่อช่วยในการสร้างผลลัพธ์ขั้นสุดท้าย ซึ่งเปรียบเสมือนข้อมูลที่เราต้องการให้ออกมา จากลักษณะดังกล่าวจึงสามารถสรุปส่วนประกอบของโครงข่ายประสาทเทียมได้ 4 ส่วนคือ

1) นิวรอน (Neuron) เป็นหน่วยประมวลผลที่เล็กที่สุดในโครงข่ายประสาทเทียม
2) ค่านำเข้า (Input) ทำหน้าที่รับเอาข้อมูลตัวอย่างที่ต้องการจากภายนอกมาเข้าสู่กระบวนการเรียนรู้
3) ค่าส่งออก (Output) ทำหน้าที่นำผลลัพธ์ที่ได้จากการประมวลผลส่งออกสู่ภายนอก ที่ซึ่งต้องการนำค่านี้ไปใช้ในการตีความต่อได้
4) ค่าน้ำหนัก (Weight) เป็นส่วนที่สำคัญมากของโครงข่ายประสาทเทียม โดยมีส่วนช่วยในการตัดสินใจข้อมูลส่งออก

จากส่วนประกอบทั้ง 4 เมื่อนำมาเปรียบเทียบกับแนวคิดในเชิงชีววิทยาจะสามารถเทียบส่วนประกอบได้ดังตารางที่ 2.1 และจากลักษณะดังกล่าวสามารถแสดงได้เป็นแผนภาพดังรูปที่ 2.2

ตารางที่ 2.1 การเปรียบเทียบการทำงานของโครงข่ายหน่วยประสาทกับโครงข่ายประสาทเทียม

โครงข่ายหน่วยประสาทเชิงชีววิทยา	โครงข่ายประสาทเทียม
โซมา	นิวรอน
เดนไดรซ์	ค่านำเข้า
แอกซอน	ค่าส่งออก
ไซแนปซิส	ค่าน้ำหนัก



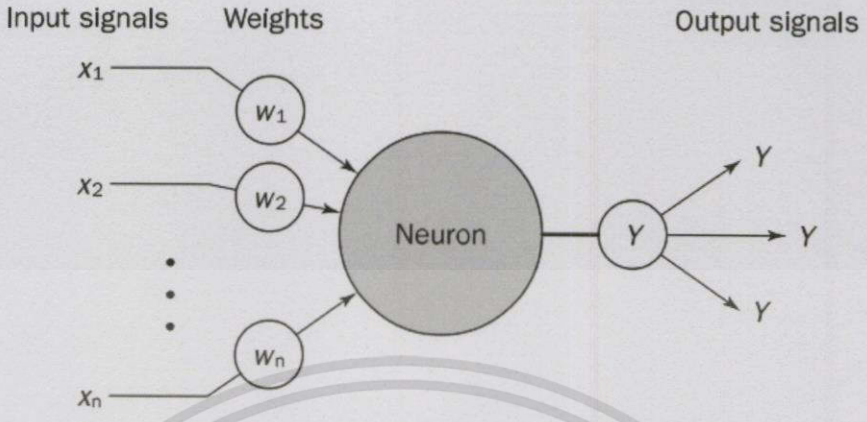
รูปที่ 2.2 ลักษณะโครงสร้างของโครงข่ายประสาทเทียม

การสร้างโครงข่ายประสาทเทียมจะต้องตัดสินใจก่อนว่าจะใช้จำนวนนิวรอนเท่าไร่ และจะเชื่อมโยงนิวรอนเหล่านั้นอย่างไร หรือกล่าวได้ว่า ก่อนอื่นเราต้องเลือกสถาปัตยกรรมของโครงข่าย (Network architecture) ก่อน จากนั้นเราจึงตัดสินใจเลือกวิธีการเรียนรู้ (Learning algorithm) ที่จะใช้ และสุดท้ายเราก็ทำการฝึก (Training) ดังนั้นจึงต้องรู้จักกับแบบจำลองโครงข่ายประสาทเทียมก่อน

2.1.3 โครงข่ายประสาทเทียมแบบชั้นเดียว (Single layer neural networks)

2.1.3.1 แบบจำลองของ McCulloch และ Walter Pitts

แบบจำลองนี้นิวรอนจะรับข้อมูลจากภายนอกผ่านทางทางเชื่อม (Link) ข้อมูลนำเข้า ไปยังนิวรอน และนิวรอนจะทำการคำนวณผลลัพธ์ในชั้นกระตุ้น (Activation level) และค่าดังกล่าวจะถูกส่งออกไปยังทางเชื่อมส่งออก (Output link) สำหรับข้อมูลนำเข้าอาจจะเป็นข้อมูลนำเข้าตรงๆเลย หรือว่าจะเป็นผลลัพธ์ที่ได้จากนิวรอนอื่นก็ได้ สำหรับข้อมูลส่งออกอาจจะเป็นคำตอบที่ต้องการ (Final solution) หรือเป็นข้อมูลนำเข้าให้อีกนิวรอนอื่นได้ แสดงให้เห็นดังรูปที่ 2.3



รูปที่ 2.3 แผนภาพนิวรอนของแบบจำลอง McCulloch และ Pitts

นิวรอนจะทำการคำนวณน้ำหนักรวมของข้อมูลนำเข้าทั้งหมดและนำมาเปรียบเทียบกับค่าขีดแบ่ง (Threshold) ใช้สัญลักษณ์แทนคือ θ ถ้าได้ค่าน้อยกว่าค่าขีดแบ่งผลลัพธ์ของนิวรอนจะได้ค่า -1 หากได้ค่ามากกว่าค่าขีดแบ่งผลลัพธ์ของนิวรอนจะได้ค่า +1 หรือกล่าวได้ว่านิวรอนนี้มีฟังก์ชั่นกระตุ้น (Activation function) คือ

$$X = \sum_{i=1}^{n} X_i W_i \tag{3.1}$$

$$Y = \begin{cases} +1 & if\ X \geq \theta \\ -1 & if\ X < \theta \end{cases} \tag{3.2}$$

โดยที่ X คือ ค่านำเข้าถ่วงน้ำหนัก (Weighted input) ที่เข้าไปยังนิวรอน

X_i คือ ค่านำเข้า (Input) ตัวที่ i

w_i คือ ค่าน้ำหนัก (Weight) ของค่านำเข้าตัวที่ i

n คือ จำนวนของค่านำเข้า

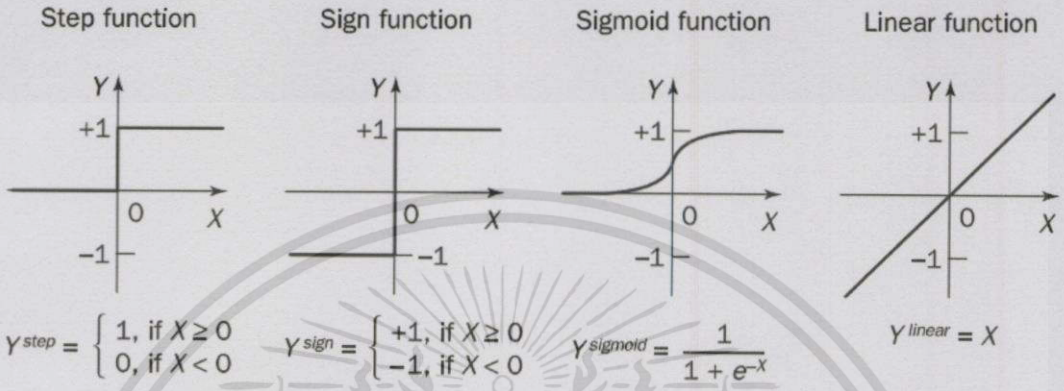
Y คือ ค่าส่งออก

θ คือ ค่าขีดแบ่ง (Threshold)

จากฟังก์ชั่นกระตุ้นที่ได้กล่าวมาจัดอยู่ในประเภทฟังก์ชั่นเครื่องหมาย (Sign function) สามารถเขียนได้เป็น

$$Y = sign\left[\sum_{i=1}^{n} x_i w_i - \theta\right] \tag{2.3}$$

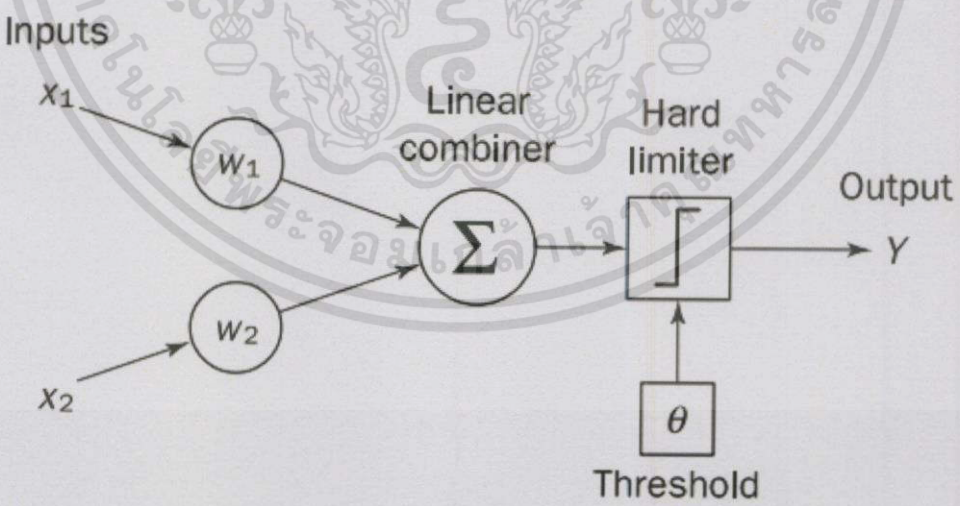
สำหรับฟังก์ชั่นกระตุ้นมีหลายชนิด เช่น ฟังก์ชั่นเครื่องหมาย, ฟังก์ชั่นแบบขั้น (Step function), ฟังก์ชั่นซิกมอยด์ (Sigmoid function) และฟังก์ชั่นเชิงเส้น (Linear function) ดังแสดงให้เห็นดังรูปที่ 2.4



รูปที่ 2.4 ฟังก์ชั่นกระตุ้นของนิวรอน

2.1.3.2 แบบจำลองของ Rosenblatt

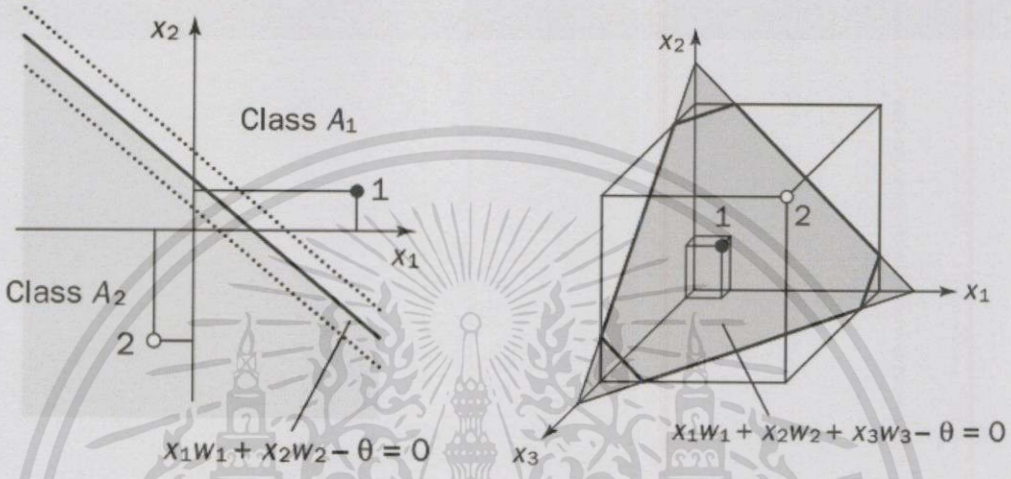
แบบจำลองนี้พัฒนามาจากพื้นฐานของแบบจำลอง McCulloch และ Pitts ซึ่งประกอบด้วยการรวมกันของฟังก์ชั่นเชิงเส้น และการปรับค่านำหนักด้วยฟังก์ชั่นฮาร์ดลิมิต (Hard limit function) ดังรูปที่ 2.5



รูปที่ 2.5 แผนภาพนิวรอนของแบบจำลอง Rosenblatt

และจะมีไฮเปอร์แพลน ซึ่งเป็นระนาบที่เกิดจากฟังก์ชั่นเชิงเส้นด้านล่างต่อไปนี้ จนทำให้เป็นระนาบที่แบ่งออกเป็นการตัดสินใจสองส่วนดังรูปที่ 2.6

$$\sum_{i=1}^{n} x_i w_i - \theta = 0 \tag{2.4}$$



(ก) (ข)

รูปที่ 2.6 ลักษณะของการแบ่งเชิงเส้นโดยที่มีจำนวนเพอเซ็ปตรอนต่างกัน

(ก) มีเพอเซ็ปตรอน 2 ข้อมูล

(ข) มีเพอเซ็ปตรอน 3 ข้อมูล

จะสังเกตว่าค่าขีดแบ่ง (θ) สามารถใช้เลื่อนเส้นขอบการแบ่งเขตของการตัดสินใจได้ ด้วยเหตุนี้ทำให้เพอเซ็ปตรอนสามารถเรียนรู้และถูกปรับค่าน้ำหนักได้ เพื่อเป็นการลดความแตกต่างระหว่างค่าจริงและค่าผิดพลาด (Error) ซึ่งสามารถหาค่าผิดพลาดได้จากสมการต่อไปนี้

$$e(p) = Y_d(p) - Y(p) \quad \text{where } p = 1, 2, 3, \ldots \tag{2.5}$$

โดยที่ e คือ ค่าความผิดพลาด

p คือ หมายเลขข้อมูลนำเข้าตัวอย่างในลำดับที่ p โดยที่ p = 1,2,3, ...

Y_d คือ ผลลัพธ์ที่ได้จากเพอเซ็ปตรอน p

Y คือ ผลลัพธ์ที่เกิดขึ้นจริง

และนำค่าความผิดพลาดที่ได้นี้มาคำนวณ เพื่อปรับค่าน้ำหนักของรอบถัดไป โดยใช้สมการดังต่อไปนี้

$$w_i(p+1) = w_i(p) + \alpha \times x_i(p) \times e(p) \tag{2.6}$$

โดยที่เครื่องหมาย α คือ อัตราการเรียนรู้ (Learning rate) ซึ่งเป็นค่าคงที่ค่าบวกที่อยู่ระหว่าง 0 ถึง 1

สำหรับขั้นตอนในการเรียนรู้ของเพอเซ็ปตรอนมีดังนี้คือ

1) การกำหนดค่าตั้งต้น (Initialization)

ทำได้โดยการสุ่มค่าน้ำหนักเริ่มต้นทุกค่า (w_1, w_2, ..., w_n) รวมถึงค่าขีดแบ่ง (θ) ให้ค่าอยู่ในช่วง -0.5 ถึง 0.5

2) การกระตุ้น (Activation)

ทำการใส่ค่านำเข้า $x_1(p)$, $x_2(p)$, ..., $x_n(p)$ รวมถึงค่าส่งออกที่ต้องการ $Y_d(p)$ ใส่ในสมการดังต่อไปนี้

$$Y(p) = step\left[\sum_{i=1}^{n} x_i(p)w_i(p) - \theta\right] \tag{2.7}$$

3) การปรับค่าน้ำหนัก (Weight training)

ปรับค่าน้ำหนักของเพอเซ็ปตรอนโดยใช้สมการต่อไปนี้

$$w_i(p+1) = w_i(p) + \Delta w_i(p) \tag{2.9}$$

โดยที่ ΔW_i คือค่าน้ำหนักที่ถูกปรับให้ถูกต้องในรอบที่ p และได้มาจากสมการ

$$\Delta w_i(p) = \alpha \times x_i(p) \times e(p) \tag{2.10}$$

4) การวนซ้ำ (Iteration)

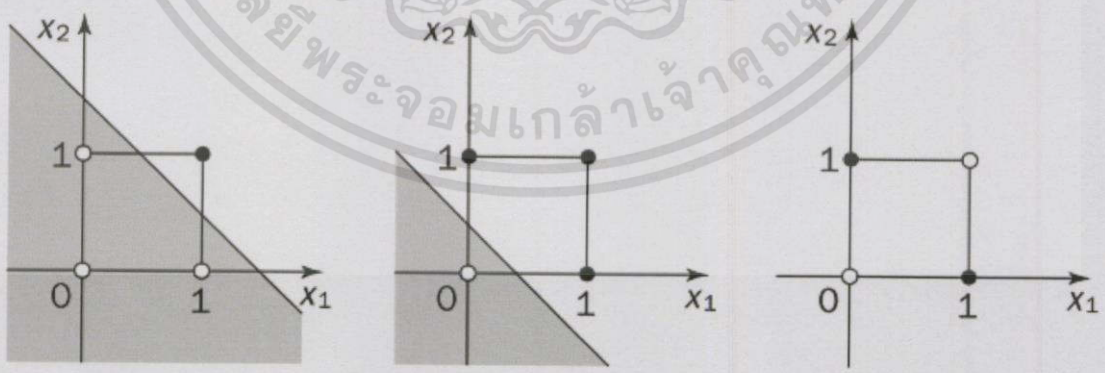
ทำการเพิ่มค่า p อีก 1 จากนั้นกลับไปที่ขั้นตอนที่ 2 และทำแบบเดิมจนกว่าค่าผิดพลาดเข้าใกล้ 0 (Convergence)

จากที่ได้กล่าวมา จะเห็นว่าข้อจำกัดของเพอเซ็ปตรอนจะอยู่ที่สามารถแก้ปัญหาได้เฉพาะปัญหาแบบเชิงเส้นทั้น เช่น ปัญหาโอเปอร์เรชันแอนด์ (AND Operation) จะสังเกตได้จากไฮเปอร์เพลนดังรูปที่ 2.7 หรือ ปัญหาโอเปอร์เรชันออร์ (OR Operation) เป็นต้น ซึ่งไม่สามารถแก้ไขปัญหาเช่น โอเปอร์เรชันเอ็กคลูซีฟออร์ (Exclusive-OR operation) ได้ จะสังเกตได้จากไฮเปอร์เพลนดังรูปที่ 2.8

Epoch	Inputs x_1	Inputs x_2	Desired output Y_d	Initial weights w_1	Initial weights w_2	Actual output Y	Error e	Final weights w_1	Final weights w_2
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	1	−1	0.2	−0.1
	1	1	1	0.2	−0.1	0	1	0.3	0.0
2	0	0	0	0.3	0.0	0	0	0.3	0.0
	0	1	0	0.3	0.0	0	0	0.3	0.0
	1	0	0	0.3	0.0	1	−1	0.2	0.0
	1	1	1	0.2	0.0	1	0	0.2	0.0
3	0	0	0	0.2	0.0	0	0	0.2	0.0
	0	1	0	0.2	0.0	0	0	0.2	0.0
	1	0	0	0.2	0.0	1	−1	0.1	0.0
	1	1	1	0.1	0.0	0	1	0.2	0.1
4	0	0	0	0.2	0.1	0	0	0.2	0.1
	0	1	0	0.2	0.1	0	0	0.2	0.1
	1	0	0	0.2	0.1	1	−1	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1
5	0	0	0	0.1	0.1	0	0	0.1	0.1
	0	1	0	0.1	0.1	0	0	0.1	0.1
	1	0	0	0.1	0.1	0	0	0.1	0.1
	1	1	1	0.1	0.1	1	0	0.1	0.1

Threshold: $\theta = 0.2$; learning rate: $\alpha = 0.1$.

รูปที่ 2.7 ตัวอย่างการเรียนรู้ของเพอเซ็ปตรอนในการแก้ไขปัญหาโอเปอร์เรชันแอนด์



(a) AND ($x_1 \cap x_2$) (b) OR ($x_1 \cup x_2$) (c) Exclusive-OR ($x_1 \oplus x_2$)

รูปที่ 2.8 การพล็อต (Plot) สมการเชิงเส้นสองมิติของโอเปอร์เรชันพื้นฐาน

จะเห็นว่าการใช้แบบจำลองโครงข่ายประสาทเทียมแบบชั้นเดียวนั้นสามารถแก้ได้เฉพาะปัญหาซึ่งสามารถใช้สมการเชิงเส้นในการแบ่งปัญหาออกมาได้ ซึ่งทำให้ไม่สามารถแก้ไขได้ทุกปัญหา ดังนั้นการพัฒนาโครงข่ายประสาทเทียมแบบหลายชั้นจึงได้เกิดขึ้นเพื่อแก้ไขปัญหาดังกล่าว

2.1.4 โครงข่ายประสาทเทียมแบบหลายชั้น (Multilayer neural networks)

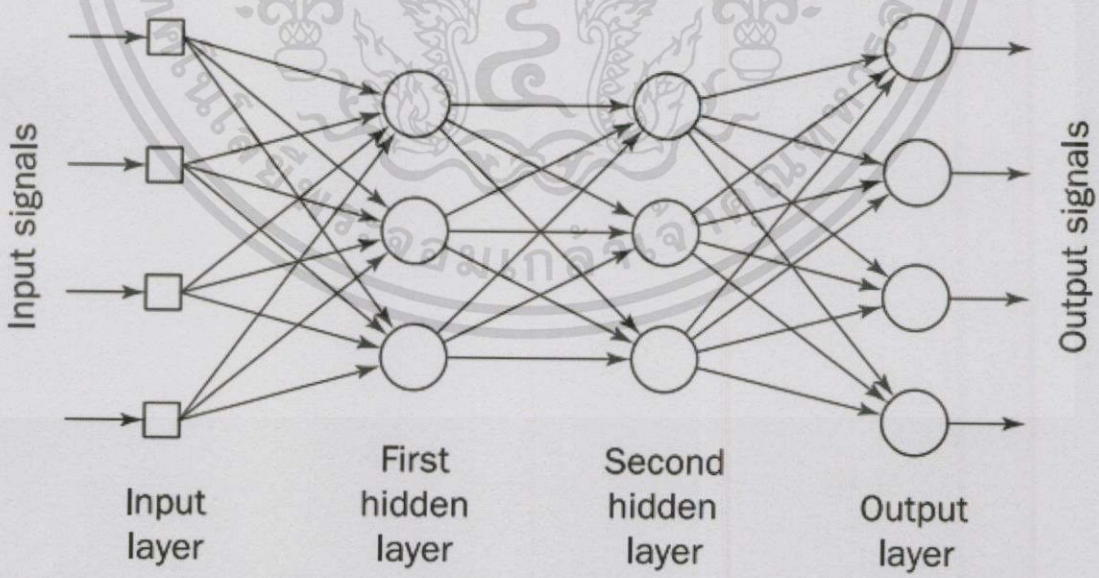
2.1.4.1 แบบจำลองโครงข่ายประสาทเทียมแบบหลายชั้น

แบบจำลองของโครงข่ายประสาทเทียมแบบหลายชั้น จะมีการเพิ่มชั้นซ่อน (Hidden layer) หรือเรียกว่าชั้นกลาง (Middle layer) ขึ้นมาได้ตั้งแต่ 1 ชั้นขึ้นไป ซึ่งโดยทั่วไปแล้วจะนิยมสามชั้นคือ ชั้นการนำเข้าข้อมูล (Input layer) ชั้นซ่อน 1 ชั้น และชั้นการส่งข้อมูลออก (Output layer) การที่มีชั้นซ่อนเพิ่มขึ้นมาส่งผลให้สามารถสร้างไฮเปอร์แพลน ได้มากกว่า 1 ไฮเปอร์เพลน ซึ่งสามารถนำมาแก้ไขปัญหาที่มีความซับซ้อนมากขึ้นได้ แต่การที่เพิ่มจำนวนชั้นหรือจำนวนนิวรอน ก็ส่งผลต่อเวลาในการคำนวณเช่นเดียวกัน ดังนั้นจึงต้องเลือกใช้ให้เหมาะสมด้วย สำหรับส่วนประกอบหลักของโครงข่ายประสาทเทียมแบบหลายชั้นมีสามส่วนได้แก่

1) ชั้นนำเข้า ทำหน้าที่ส่งข้อมูลผ่านเข้าสู่เพอเซ็ปตรอน แต่จะไม่มีการประมวลผลหรือคำนวณใดๆในชั้นนี้

2) ชั้นซ่อน ทำหน้าที่หาค่าน้ำหนักที่เหมาะสม ซึ่งซ่อนอยู่ในรูปแบบของข้อมูลที่นำเข้ามา

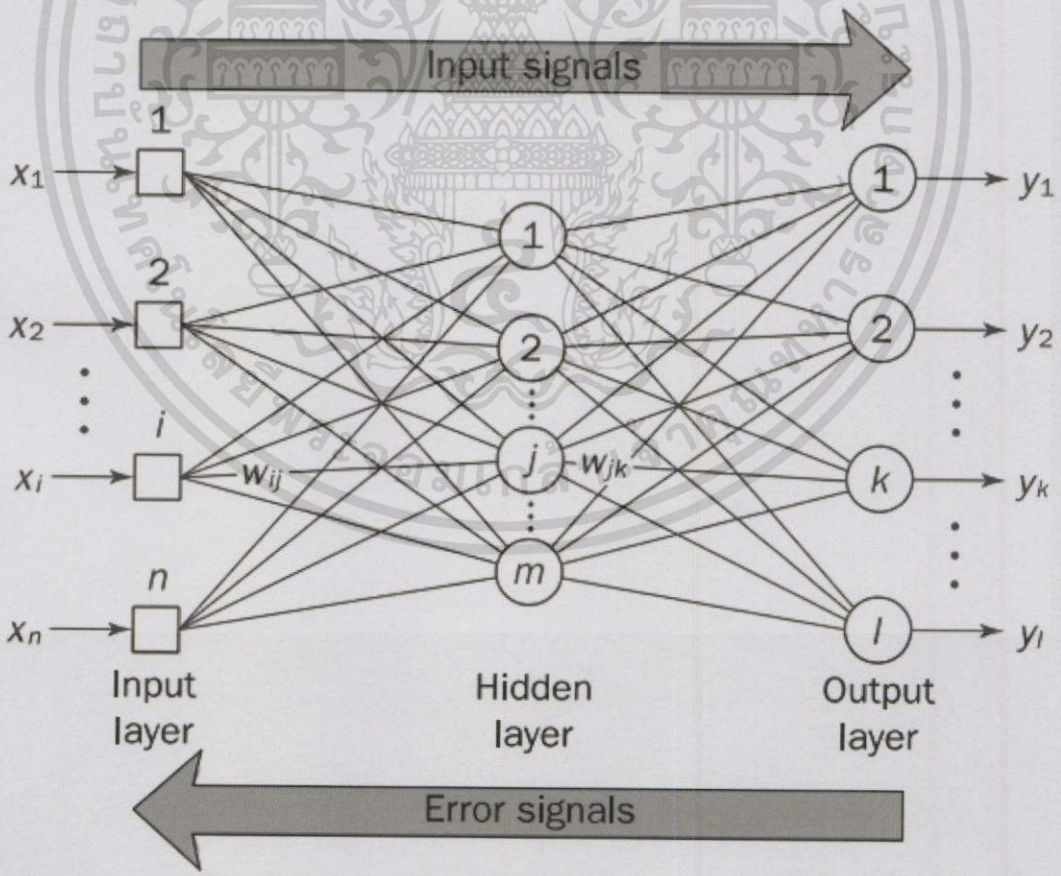
3) ชั้นส่งออก ทำหน้าที่ในการแสดงข้อมูลจากการประมวลผลทั้งหมดออกไปภายนอก



รูปที่ 2.9 ตัวอย่างของแบบจำลองโครงข่ายประสาทเทียมแบบหลายชั้นและมีสองชั้นซ่อน

2.1.4.2 ขั้นตอนวิธีการเรียนรู้แบบการถ่ายทอดย้อนกลับ (Back-propagation learning algorithm)

โครงข่ายประสาทเทียมแบบหลายชั้นมีขั้นตอนวิธีการเรียนรู้มากมายที่สามารถใช้ได้ แต่หนึ่งในขั้นตอนวิธีที่ได้รับความนิยมมากที่สุดนั่นก็คือ การถ่ายทอดสัญญาณย้อนกลับ (Back-propagation) ซึ่งขั้นตอนวิธีในการเรียนรู้แบ่งเป็นสองช่วงหลักๆโดยการส่งผ่านครั้งแรกจะเป็นไปในรูปแบบของการคำนวณไปข้างหน้า คือข้อมูลนำเข้าจะถูกป้อนจากชั้นนำเข้าผ่านชั้นต่างๆจากซ้ายไปขวาและคำนวณค่า ได้ผลลัพธ์คือข้อมูลส่งออกมาที่ชั้นส่งออก หากผลลัพธ์ที่ออกมาแตกต่างจากผลลัพธ์ที่คาดหวังไว้ (Desired output) สัญญาณความผิดพลาด (Error signal) จะถูกคำนวณและส่งผ่านจากขวาไปซ้าย หรือจากต้นทางคือชั้นส่งออกย้อนกลับไปยังชั้นนำเข้านั่นเอง และในระหว่างนั้นค่าน้ำหนักก็จะถูกเปลี่ยนแปลงไปด้วย และลักษณะสำคัญอีกหนึ่งประการคือ ชั้นต่างๆจะต้องมีการเชื่อมต่อกันแบบเต็ม (Fully connected) นั่นหมายถึงว่าทุกนิวรอนในแต่ละชั้นจะต้องเชื่อมต่อกับนิวรอนอื่นทุกอันในชั้นที่ติดกัน ดังแสดงในรูปที่ 2.10



รูปที่ 2.10 ขั้นตอนวิธีการเรียนรู้แบบการถ่ายทอดย้อนกลับ

สำหรับวิธีการคำนวณค่าน้ำหนักของข้อมูลนำเข้าสามารถหาได้จากสมการต่อไปนี้

$$X = \sum_{i=1}^{n} x_i w_i - \theta \tag{2.11}$$

สำหรับค่าข้อมูลที่นำเข้ามาจะผ่านฟังก์ชั่นกระตุ้น (Activation function) จะใช้ซิกมอยด์ฟังก์ชั่น ซึ่งแสดงให้เห็นตามสมการดังนี้ ซึ่งสมการดังกล่าวรับประกันได้ว่าค่าส่งออกมีค่าอยู่ระหว่าง 0 กับ 1

$$Y^{sigmoid} = \frac{1}{1+e^{-X}} \tag{2.12}$$

สำหรับสมการอื่นๆที่ใช้ในการคำนวณประกอบด้วย

1) สัญญาณความผิดพลาด หาได้จาก

$$e_k(p) = y_{d,k}(p) - y_k(p) \tag{2.13}$$

โดยที่ $y_{d,k}(p)$ คือ ผลลัพธ์ที่ถูกคาดหวังไว้ของนิวรอน k ในรอบที่ p

$Y_k(p)$ คือ ผลลัพธ์ที่คำนวณได้จริงของนิวรอน k ในรอบที่ p

2) กลไกที่ใช้ในการปรับค่าน้ำหนัก (Updating weight)

$$w_{jk}(p+1) = w_{jk}(p) + \Delta w_{jk}(p) \tag{2.14}$$

โดยที่ ΔW_{jk} คือค่าน้ำหนักที่แตกต่าง (Weight correction)

3) ค่าน้ำหนักที่แตกต่างที่ชั้นส่งออก

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p) \tag{2.15}$$

โดยที่ $\delta_k(p)$ คือระดับการเปลี่ยนแปลงของความผิดพลาด (Error gradient) ของนิวรอน k ที่ชั้นส่งออกในรอบที่ p ซึ่งระดับการเปลี่ยนแปลงของความผิดพลาดได้มาจากการหาอนุพันธ์ของฟังก์ชั่นกระตุ้น จากนั้นทำการคูณกับค่าความผิดพลาดของนิวรอนที่ชั้นส่งออก

4) ค่าน้ำหนักที่แตกต่างที่ชั้นส่งออก

$$\delta_k(p) = \frac{\partial y_k(p)}{\partial X_k(p)} \times e_k(p) \tag{2.16}$$

โดยที่ $y_k(p)$ คือ ผลลัพธ์ของนิวรอน k ในรอบที่ p

$X_k(p)$ คือ ค่านำเข้ารวมน้ำหนัก (Net weighted input) ของนิวรอน k ในรอบที่ p

สามารถเขียนได้ว่า

$$\delta_k(p) = \frac{\partial\left\{\dfrac{1}{1+\exp[-X_k(p)]}\right\}}{\partial X_k(p)} \times e_k(p)$$

$$= \frac{\exp[-X_k(p)]}{\{1+\exp[-X_k(p)]\}^2} \times e_k(p)$$

ซึ่งจะได้

$$= y_k(p) \times [1 - y_k(p)] \times e_k(p) \qquad (2.17)$$

โดยที่

$$y_k(p) = \frac{1}{1+\exp[-X_k(p)]} \qquad (2.18)$$

5) ค่าน้ำหนักที่แตกต่างที่ชั้นซ่อน

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p) \qquad (2.19)$$

โดยที่ $\delta_j(p)$ คือ ระดับการเปลี่ยนแปลงของความผิดพลาดของนิวรอน j ที่ชั้นซ่อนในรอบที่ p

6) ระดับการเปลี่ยนแปลงของความผิดพลาดที่ชั้นซ่อน

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) w_{jk}(p) \qquad (2.20)$$

โดยที่ $y_j(p)$ คือ จำนวนของนิวรอนในชั้นส่งออก

2.1.4.3 ขั้นตอนวิธีการเรียนรู้แบบการถ่ายทอดย้อนกลับ (Back-propagation learning algorithm)

สำหรับขั้นตอนในการเรียนรู้แบบการถ่ายทอดย้อนกลับมีดังนี้คือ

1) การกำหนดค่าตั้งต้น (Initialization)

กำหนดค่าตั้งต้นของค่าน้ำหนักของค่าขีดแบ่งทุกค่าโดยสุ่มจากช่วงตัวเลขที่แคบ

2) การกระตุ้น (Activation)

ใส่ค่าข้อมูลนำเข้าและผลลัพธ์ที่ถูกคาดวังไว้ให้กับโครงข่ายประสาทเทียมจากนั้นก็ดำเนินการดังต่อไปนี้

2.1) คำนวณหาผลลัพธ์ของนิวรอนในชั้นซ่อนจากสมการ

$$y_j(p) = sigmoid\left[\sum_{i=1}^{n} x_i(p) \times w_{ij}(p) - \theta_j\right] \quad (2.21)$$

โดยที่ n คือ จำนวนของค่านำเข้าของนิวรอน j ในชั้นซ่อน

sigmoid คือ การใช้ฟังกชั่นซิกมอยด์เป็นฟังก์ชั่นกระตุ้น

2.2) คำนวณหาผลลัพธ์ของนิวรอนในชั้นส่งออกจากสมการ

$$y_k(p) = sigmoid\left[\sum_{j=1}^{m} x_{jk}(p) \times w_{jk}(p) - \theta_k\right] \quad (2.22)$$

โดยที่ m คือ จำนวนของค่านำเข้าของนิวรอน k ในชั้นส่งออก

3) การปรับค่าน้ำหนัก (Weight training)

ทำการปรับค่าน้ำหนักในการถ่ายทอดสัญญาณควาผิดพลาดย้อนจากชั้นส่งออกไปสู่ชั้นนำเข้า ประกอบไปด้วย

3.1) ทำการคำนวณที่ชั้นส่งออก คำนวณระดับการเปลี่ยนของความผิดพลาดในชั้นส่งออกจากสมการ

$$\delta_k(p) = y_k(p) \times [1 - y_k(p)] \times e_k(p) \quad (2.23)$$

คำนวณค่าน้ำหนักที่แตกต่างจากสมการ

$$\Delta w_{jk}(p) = \alpha \times y_j(p) \times \delta_k(p) \quad (2.24)$$

3.2) ทำการคำนวณที่ชั้นซ่อน

คำนวณระดับการเปลี่ยนของความผิดพลาดในชั้นซ่อนจากสมการ

$$\delta_j(p) = y_j(p) \times [1 - y_j(p)] \times \sum_{k=1}^{l} \delta_k(p) \times w_{jk}(p) \quad (2.25)$$

คำนวณค่าน้ำหนักที่แตกต่างจากสมการ

$$\Delta w_{ij}(p) = \alpha \times x_i(p) \times \delta_j(p) \tag{2.26}$$

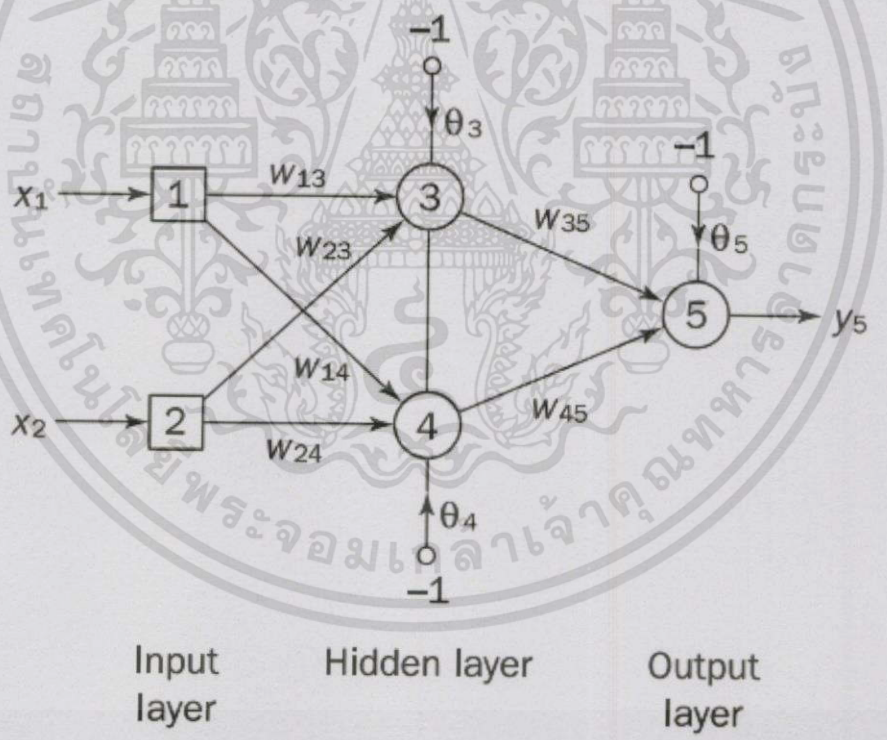
ปรับค่าน้ำหนักของทั้งเครือข่าย

$$w_{ij}(p+1) = w_{ij}(p) + \Delta w_{ij}(p) \tag{2.27}$$

4) การทำซ้ำ (Iteration)

กลับไปทำซ้ำตั้งแต่ขั้นตอนที่ 2 จนกว่าจะได้ค่าที่ตรงกับบรรทัดฐานความผิดพลาด (Error criterion) ที่ตั้งไว้

2.1.4.4 ตัวอย่างโครงข่ายประสาทเทียมสามชั้นสำหรับแก้ไขปัญหาเอ็กซ์คลูซีฟออร์ (Three-layer neural network for solving Exclusive-OR problem)



รูปที่ 2.11 โครงข่ายประสาทเทียมสำหรับแก้ปัญหาเอ็กซ์คลูซีฟออร์

ขั้นตอนแรกคือทำการให้ค่าน้ำหนักและค่าน้ำหนักของค่าขีดแบ่งจากการสุ่ม ซึ่งจากภาพด้านบนจะเห็นว่าค่าขีดแบ่งนี้ถูกตั้งค่านำเข้าถาวรแล้วคือมีค่าเท่ากับ -1 โดยสมมติให้ผลจากการสุ่มเป็นดังนี้

$w_{13} = 0.5,\ w_{14} = 0.9,\ w_{23} = 0.4,\ w_{24} = 1.0,\ w_{35} = -1.2,\ w_{45} = 1.1,$
$\theta_3 = 0.8,\ \theta_4 = -0.1$ and $\theta_5 = 0.3.$

ขั้นตอนที่สองคือ การป้อนค่านำเข้าโดยให้ $x_i = 1$ และ $x_2 = 1$ โดยที่ผลลัพธ์ที่ถูกคาดหวังไว้จะมีค่าเท่ากับ 0 หรือ $y_{d,5} = 0$ จากนั้นคำนวณหาผลลัพธ์ที่นิวรอน 3 และ 4 ในชั้นซ่อนดังนี้

$$y_3 = sigmoid\,(x_1 w_{13} + x_2 w_{23} - \theta_3) = 1/[1 + e^{-(1\times0.5+1\times0.4-1\times0.8)}] = 0.5250$$

$$y_4 = sigmoid\,(x_1 w_{14} + x_2 w_{24} - \theta_4) = 1/[1 + e^{-(1\times0.9+1\times1.0+1\times0.1)}] = 0.8808$$

และทำการคำนวณผลลัพธ์ที่นิวรอน 5 จากค่า y_3 และ y_4 ที่ได้

$$y_5 = sigmoid\,(y_3 w_{35} + y_4 w_{45} - \theta_5) = 1/[1 + e^{-(-0.5250\times1.2+0.8808\times1.1-1\times0.3)}] = 0.5097$$

ดังนั้นจะสามารถคำนวณค่าความผิดพลาดได้ดังนี้

$$e = y_{d,5} - y_5 = 0 - 0.5097 = -0.5097$$

ขั้นตอนต่อไปคือการปรับค่าน้ำหนักโดยในการปรับค่าน้ำหนักและค่าน้ำหนักของค่าขีดแบ่งในเครือข่าย ค่าความผิดพลาดจะถูกถ่ายทอดย้อนกลับจากชั้นส่งออกไปหาชั้นนำเข้าสิ่งแรกที่จะทำการคำนวณในขั้นตอนนี้คือ ระดับการเปลี่ยนแปลงของความผิดพลาดในชั้นส่งออก

$$y_5 = sigmoid\,(y_3 w_{35} + y_4 w_{45} - \theta_5) = 1/[1 + e^{-(-0.5250\times1.2+0.8808\times1.1-1\times0.3)}] = 0.5097$$

จากนั้นค่าน้ำหนักที่แตกต่างจะสามารถคำนวณได้โดยกำหนดให้อัตราการเรียนรู้มีค่าเท่ากับ 0.1

$$\Delta w_{35} = \alpha \times y_3 \times \delta_5 = 0.1 \times 0.5250 \times (-0.1274) = -0.0067$$

$$\Delta w_{45} = \alpha \times y_4 \times \delta_5 = 0.1 \times 0.8808 \times (-0.1274) = -0.0112$$

$$\Delta \theta_5 = \alpha \times (-1) \times \delta_5 = 0.1 \times (-1) \times (-0.1274) = 0.0127$$

ลำดับถัดไปจะเป็นการคำนวณการเปลี่ยนแปลงของความผิดพลาดของนิวรอน 3 และ 4 ในชั้นซ่อน

$$\delta_3 = y_3(1 - y_3) \times \delta_5 \times w_{35} = 0.5250 \times (1 - 0.5250) \times (-0.1274) \times (-1.2) = 0.0381$$

$$\delta_4 = y_4(1 - y_4) \times \delta_5 \times w_{45} = 0.8808 \times (1 - 0.8808) \times (-0.1274) \times 1.1 = -0.0147$$

ต่อไปจะเป็นการคำนวณค่าน้ำหนักที่แตกต่าง

$$\Delta w_{13} = \alpha \times x_1 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta w_{23} = \alpha \times x_2 \times \delta_3 = 0.1 \times 1 \times 0.0381 = 0.0038$$

$$\Delta \theta_3 = \alpha \times (-1) \times \delta_3 = 0.1 \times (-1) \times 0.0381 = -0.0038$$

$$\Delta w_{14} = \alpha \times x_1 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$

$$\Delta w_{24} = \alpha \times x_2 \times \delta_4 = 0.1 \times 1 \times (-0.0147) = -0.0015$$

$$\Delta \theta_4 = \alpha \times (-1) \times \delta_4 = 0.1 \times (-1) \times (-0.0147) = 0.0015$$

สุดท้ายจะเป็นการปรับค่าน้ำหนักและค่าน้ำหนักของค่าขีดแบ่งของทั้งเครือข่าย

$$w_{13} = w_{13} + \Delta w_{13} = 0.5 + 0.0038 = 0.5038$$

$$w_{14} = w_{14} + \Delta w_{14} = 0.9 - 0.0015 = 0.8985$$

$$w_{23} = w_{23} + \Delta w_{23} = 0.4 + 0.0038 = 0.4038$$

$$w_{24} = w_{24} + \Delta w_{24} = 1.0 - 0.0015 = 0.9985$$

$$w_{35} = w_{35} + \Delta w_{35} = -1.2 - 0.0067 = -1.2067$$

$$w_{45} = w_{45} + \Delta w_{45} = 1.1 - 0.0112 = 1.0888$$

$$\theta_3 = \theta_3 + \Delta \theta_3 = 0.8 - 0.0038 = 0.7962$$

$$\theta_4 = \theta_4 + \Delta \theta_4 = -0.1 + 0.0015 = -0.0985$$

$$\theta_5 = \theta_5 + \Delta \theta_5 = 0.3 + 0.0127 = 0.3127$$

กระบวนการจะวนซ้ำจนกระทั่งผลบวกของค่าความผิดพลาดยกกำลังสองมีค่าน้อยกว่า 0.001

2.2 การส่งข้อมูลผ่านเครือข่าย

2.2.1. สถาปัตยกรรมผู้ใช้บริการและผู้ให้บริการ (Client-Server Architecture)

สถาปัตยกรรมผู้ใช้บริการและผู้ให้บริการเป็นสถาปัตยกรรมสำหรับการติดต่อสื่อสาร หรือแลกเปลี่ยนข้อมูล หรือแบ่งปันทรัพยากรร่วมกัน ซึ่งสถาปัตยกรรมนี้จะมีส่วนประกอบย่อยสองส่วนคือ

1) Client หมายถึง ผู้ใช้บริการ กล่าวคือการที่ ผู้ใช้ที่เป็นทรัพยากรบุคคล ได้ใช้เครื่องมือหรืออุปกรณ์ ในการส่งคำร้องขอ (Request) ข้อมูล ไปยังผู้ให้บริการ และรอการตอบกลับจากผู้ให้บริการ
2) Server หมายถึง ผู้ให้บริการ กล่าวคือการที่อุปกรณ์ทำการรอคำขอใช้บริการจากผู้ใช้บริการ และทำหน้าที่บริหารจัดการข้อมูลหรือทรัพยากร และส่งคำตอบสนอง (Response) กลับไปยังผู้ใช้บริการที่ได้ร้องขอมา

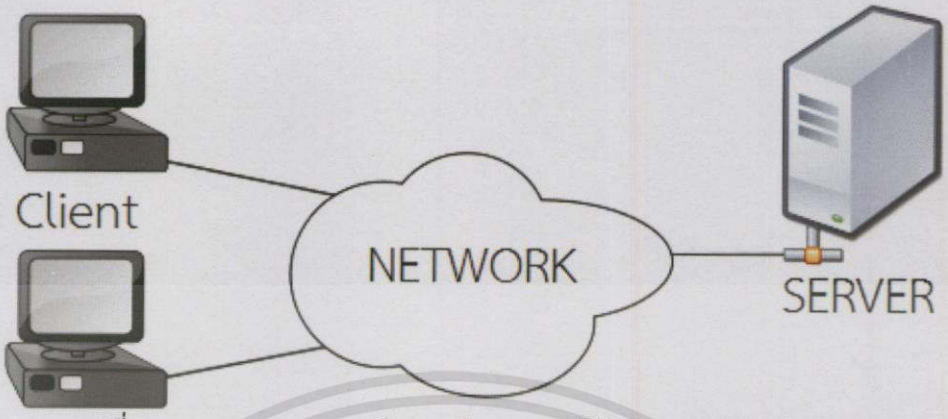
2.2.1.1 ข้อดีของการใช้สถาปัตยกรรมนี้

1) ข้อมูลหรือทรัพยากรถูกเก็บและจัดการโดยผู้ให้บริการทั้งหมด ทำให้สามารถควบคุมและจำกัดการเข้าถึงข้อมูลหรือทรัพยากรนั้นๆได้
2) สามารถทำการปรับปรุงประสิทธิภาพการให้บริการได้ง่ายเช่น การอัพเกรด (Upgrade) โปรแกรมที่ให้บริการ, การเปลี่ยนอุปกรณ์เพื่อขยายการให้บริการ เป็นต้น
3) ง่ายต่อการบริหารและจัดการ รวมถึงการตรวจสอบข้อมูลและทรัพยากรต่างๆในระบบ

2.2.1.2 ข้อเสียของการใช้สถาปัตยกรรมนี้

1) ภาระการทำงานจะตกอยู่ที่เครื่องผู้ให้บริการ และหากเกิดปัญญหาที่เครื่องของผู้ให้บริการและไม่มีระบบสำรองไว้ ก็จะไม่สามารถให้บริการ แก่ผู้ใช้บริการได้เลย
2) ความปลอดภัย หากมีการละเมิดมาตรการการปลอดภัย หรือมีการบุกรุกเข้ามาโดยมิชอบ ผู้ที่บุกรุก อาจจะได้ครบครองข้อมูลและทรัพยากรหลายๆอย่าง หรือทุกอย่างภายในการโจมตีนั้น เพื่อแค่ที่เดียวครั้งเดียว

ตัวอย่างการนำไปใช้งานที่เห็นในปัจจุบันเช่น การแบ่งปันไฟล์ร่วมกัน โดยการสร้าง FTP Server, การแลกเปลี่ยนจดหมายระหว่างกัน โดยการสร้าง Mail Server เป็นต้น



รูปที่ 2.12 แผนภาพแสดงถึงสถาปัตยกรรมผู้ใช้บริการและผู้ให้บริการ

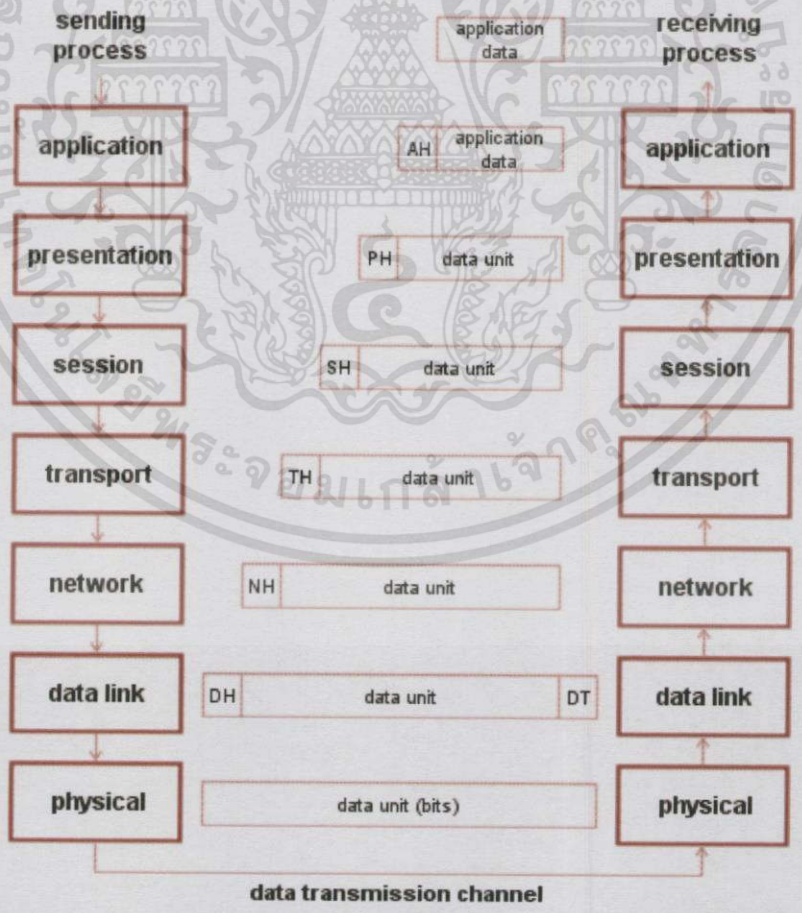
2.2.2. แบบจำลองโอเอสไอ (OSI Model)

องค์กร International Standards Organization (ISO) ได้กำหนดรูปแบบโครงสร้างมาตรฐานสากลสำหรับการติดต่อสื่อและแลกเปลี่ยนข้อมูลระหว่างกันขึ้น เพื่อเป็นโมเดลสำหรับการอ้างอิง (Reference model) นั่นก็คือ Open System Interconnection (OSI) ซึ่งต่อไปจะเรียกว่าแบบจำลองโอเอสไอ โดยแบบจำลองโอเอสไอนี้แบ่งออกเป็น 7 ชั้น (Layer) ดังรูปที่ 2.13 ซึ่งแต่ละชั้นก็จะมีหน้าที่แตกต่างกันไปดังนี้

1) Physical layer เป็นชั้นล่างสุดของการติดต่อสื่อสาร ทำหน้าที่รับและส่งข้อมูลผ่านสื่อกลาง (Media) ระหว่างอุปกรณ์หนึ่งไปยังอุปกรณ์อื่นๆ โดยมีข้อกำหนดของสื่อกลาง เช่น ใช้สัญญาณไฟกี่โวลต์, ใช้เทคนิคใดในการมัลติเพล็กซ์ เป็นต้น ตัวอย่างของสื่อกลางเช่น สายทองแดงแบบบิดคู่ตีเกลียว (Unshielded Twisted Pair), สายใยแก้วนำแสง (Fiber Optic) เป็นต้น
2) Data Link layer เป็นชั้นที่ควบคุมความการส่งข้อมูลและควบคุมความผิดพลาดของข้อมูลในระดับการส่งข้อมูลภายในเครือข่ายเดียวกัน โดยจะแบ่งข้อมูลออกเป็นเฟรม (Frame) ก่อนการส่ง และอีกหน้าที่หนึ่งคือ ป้องกันไม่ให้เครื่องส่งทำการส่งข้อมูลเร็วจนเกิดขีดความสามารถของเครื่องผู้รับจะรับข้อมูลได้
3) Network layer เป็นชั้นที่ออกแบบหรือกำหนดเส้นทางการเดินทางของข้อมูล ในการส่งผ่านข้อมูลระหว่างต้นทางและปลายทางที่มีเครือข่ายที่ต่างกันให้ดีที่สุด โดยการแบ่งข้อมูลออกมาเป็นแพ็กเกต (Packet)
4) Transport layer บางครั้งเรียกว่า ชั้นเครื่องต่อเครื่อง (Host-to-Host) ทำหน้าที่เกี่ยวกับการสื่อสารกันระหว่างต้นทางและปลายทางว่าสามารถส่งถึงปลายทางได้ครบถ้วนถูกต้องหรือไม่ ตัวอย่างการทำงานในชั้นนี้เช่น การตรวจสอบข้อมูลว่ามีการผิดพลาดระหว่างการส่ง

(Error detection and correction), การควบคุมการไหลของข้อมูล (Flow control), การรวบและแยกข้อมูล (Reassemble) เป็นต้น

5) Session layer ทำหน้าที่คอยรักษา ควบคุม สร้าง ปรับปรุง การเชื่อมต่อเฉพาะระหว่างผู้ใช้งานที่ใช้งานผ่านอุปกรณ์หนึ่งกับกับอุปกรณ์อื่นๆ ให้มีจังหวะการรับและส่งข้อมูลที่สอดคล้องกัน (Synchronization)
6) Presentation layer เป็นชั้นที่มีการจัดรูปแบบของข้อมูลเพื่อให้สามารถแสดงผลได้อย่างถูกต้อง เช่น การเข้ารหัส (Encoding), การบีบอัดข้อมูล (Compression) เป็นต้น
7) Application layer เป็นชั้นบนสุดของโมเดลจะเป็นชั้นที่ใกล้ชิดกับผู้ใช้มากที่สุด ทำหน้าที่รองรับการให้บริการแก่ผู้ใช้ หรือกล่าวได้ว่าเป็นส่วนที่ทำการติดต่อระหว่างแอปพลิเคชั่นของเครือข่ายผู้ใช้งาน ตัวอย่างแอปพลิเคชั่นบนเครือข่ายเช่น ระบบจดหมายอิเล็คทรอนิค (Electronic Mail) การโอนถ่ายแฟ้มข้อมูล (File Transfer) การควบคุมเครื่องยะไกล (Remote Desktop) เป็นต้น



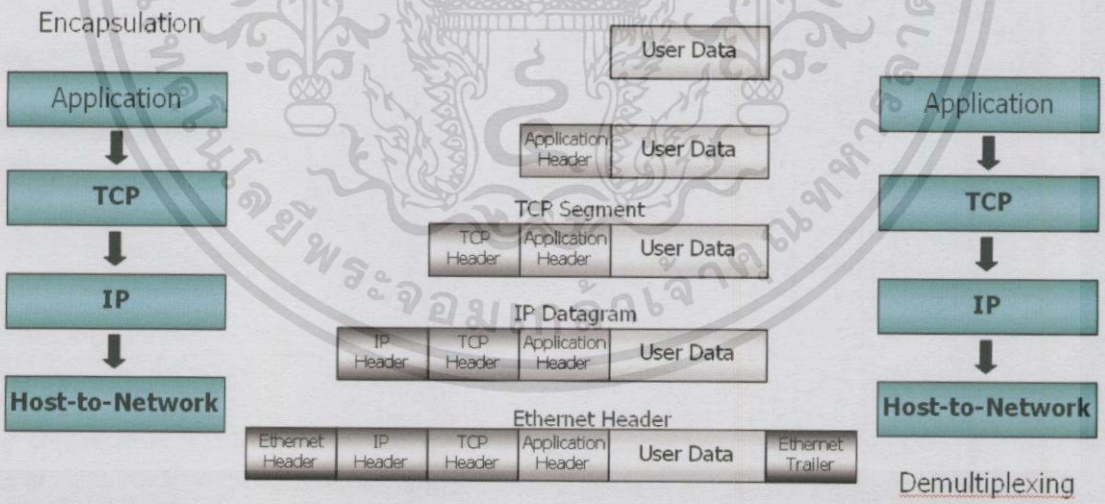
รูปที่ 2.13 แผนภาพแสดงกระบวนการรับและส่งข้อมูลผ่านแบบจำลองโอเอสไอ 7 ชั้น

2.2.3. ทีซีพี/ไอพี (TCP/IP)

2.2.3.1 TCP/IP (Transmission Control Protocol/Internet Protocol)

TCP/IP เป็นชุดของโปรโตคอลที่ถูกใช้ในการสื่อสารผ่านเครือข่ายอินเทอร์เน็ต โดยมีวัตถุประสงค์เพื่อให้สามารถใช้สื่อสารจากต้นทางข้ามเครือข่ายไปยังปลายทางได้ นิยมใช้กันอย่างแพร่หลาย ก่อนการประกาศใช้มาตรฐานแบบจำลองโอเอสไอ ปัจจุบันจึงยังใช้โพรโทคอลนี้ในการทำงานอยู่ โดยได้แบ่งชั้นการทำงานซึ่งคล้ายกับแบบจำลองโอเอสไอ แต่มีเพียงเป็น 4 ชั้นดังนี้

1) Link layer หรือเรียกว่า (Host-to-network) เป็นชั้นที่ทำหน้าที่ในการส่งข้อมูลภายในเครือข่าย คล้ายกับชั้นที่ 1-2 ตามแบบจำลองโอเอสไอ
2) Internet layer มีหน้าที่นำพาข้อมูลจากต้นทางให้ผ่านเส้นทางที่ดีที่สุด ไปยังปลายทางได้ โดยทั้งต้นทางและปลายทางจะมีไอพี (IP) เพื่อใช้ในการพิจารณา เปรียบเสมือนชั้นที่ 3 ตามแบบจำลองโอเอสไอ
3) Transport layer เป็นชั้นที่ทำหน้าที่ตรวจสอบข้อมูลและควบคุมการส่งข้อมูล มีหน้าที่เสมือนชั้นที่ 4 ตามแบบจำลองโอเอสไอ
4) Application layer เป็นชั้นที่ต้องติดต่อกับผู้ใช้งาน ทำหน้าที่เปรียบเสมือนชั้นที่ 5-7 ตามแบบจำลองโอเอสไอ

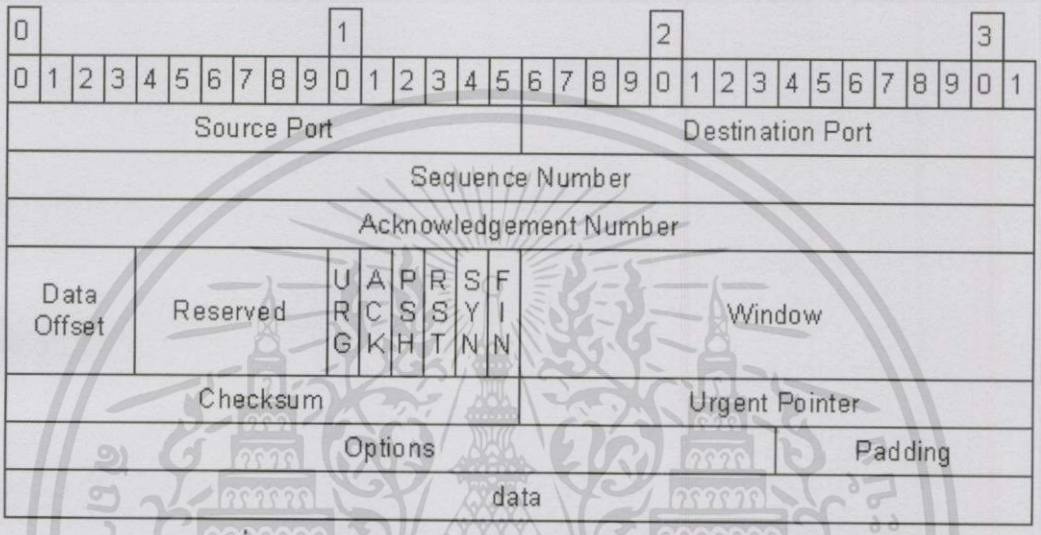


รูปที่ 2.14 แผนภาพแสดงการทำงานในภาพรวมของโพรโทคอลทีซีพี/ไอพี

2.2.3.2 โครงสร้างของ TCP

Transmission Control Protocol (TCP) เป็นแบบที่มีการกำหนดช่วงการสื่อสารตลอดระยะเวลาการสื่อสาร (connection-oriented) ซึ่งจะยอมให้มีการส่งข้อมูลเป็นแบบที่ละไบต์ (Byte stream) ที่มีกระบวนการเพื่อตรวจสอบข้อมูลที่มีความข้อผิดพลาด รวมถึงข้อมูลที่มีปริมาณมากจะ

ถูกแบ่งออกเป็นส่วนเล็กๆ เรียกว่า เซกเมนต์ (segment) ซึ่งจะถูกส่งไปยังผู้รับผ่านทางชั้นสื่อสารของอินเทอร์เน็ต ทางฝ่ายผู้รับจะนำ segment มาเรียงต่อกันตามลำดับเป็นข้อมูลตัวเดิม TCP ยังมีความสามารถในการควบคุมการไหลของข้อมูลเพื่อป้องกันไม่ให้ผู้ส่ง ส่งข้อมูลเร็วเกินกว่าที่ผู้รับจะทำงานได้ทันอีกด้วย



รูปที่ 2.15 แผนภาพแสดงโครงสร้างของโพรโทคอลทีซีพี

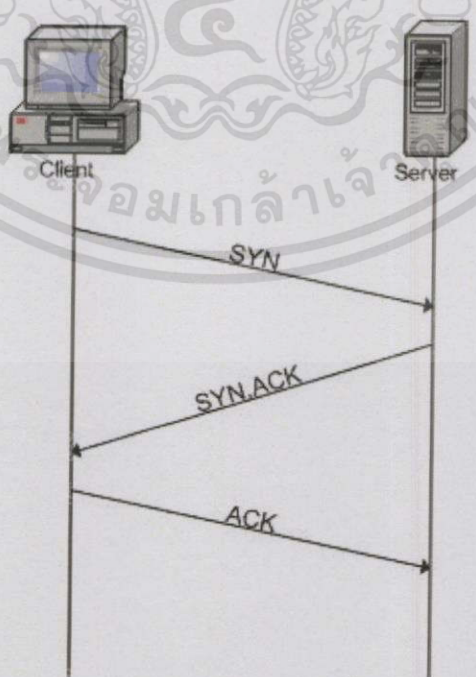
จากรูปที่ 2.15 ในส่วนของเฮดเดอร์ (Header) จะประกอบด้วยฟิลด์ (Field) ต่างๆ ซึ่งมีรายละเอียดดังนี้

1) Source Port Number เป็นหมายเลขพอร์ตต้นทางที่ส่งเซกเมนต์
2) Destination Port Number เป็นหมายเลขพอร์ตปลายทางที่จะเป็นผู้รับเซกเมนต์
3) Sequence Number เป็นหมายเลขที่ใช้ในการอ้างอิงลำดับการสื่อสารข้อมูลแต่ละครั้ง เพื่อใช้ในการแยกแยะว่าเป็นข้อมูลของชุดใด และนำมาจัดลำดับได้ถูกต้อง
4) Acknowledgment Number ทำหน้าที่เช่นเดียวกับ Sequence Number แต่จะใช้ในการตอบรับกลับมายังผู้ส่ง
5) Header Length เป็นการระบุความยาวของเฮดเดอร์ โดยปกติความยาวของเฮดเดอร์ TCP จะมีความยาว 20 ไบต์ แต่อาจจะมากกว่านั้น ถ้ามีข้อมูลในฟิลด์ตัวเลือก (Option) แต่ต้องไม่เกิน 60 ไบต์
6) Flag เป็นข้อมูลระดับบิตที่อยู่ในเฮดเดอร์ TCP โดยใช้เป็นตัวบอกคุณสมบัติของแพ็กเก็ต TCP ขณะนั้นๆและใช้เป็นตัวควบคุมจังหวะการรับส่งข้อมูลด้วย Flag ในเฮดเดอร์ของ TCP มีความสำคัญในการกำหนดการทำงานของ TCP segment เนื่องจากข้อมูลในเฮดเดอร์ของ

TCP จะมีข้อมูลครบถ้วนทั้งการรับและการส่งข้อมูล ซึ่งในการทำงานแต่ละอย่างจะมีการใช้งานฟิลด์ไม่เหมือนกัน flag จะเป็นตัวกำหนดว่าให้ใช้งานฟิลด์ไหน เช่น ฟิลด์ Acknowledgment number จะไม่ถูกใช้ในขั้นตอนการเริ่มต้นการเชื่อมต่อ แต่จะมีข้อมูลในฟิลด์ ซึ่งเป็นข้อมูลที่ไม่มีความหมายใดๆ ซึ่งถ้าไม่มี flag เป็นตัวกำหนดก้ออาจจะมีการนำข้อมูลมาใช้ และก่อให้เกิดความผิดพลาดได้

นอกจากนี้แล้วก่อนการจะเริ่มสื่อสารส่งข้อมูล จะต้องมีการการส่งสัญญาณบอกอีกฝ่ายหนึ่งที่ต้องการติดไปให้เตรียมตัวที่จะรับการติดต่อก่อน โดยกระบวนการนี้เรียกว่า กระบวนการจับมือร่วมสามขั้นตอน (Three-way Handshake) ดังรูปที่ 2.16 ซึ่งประกอบด้วย 3 ขั้นตอนดังนี้

1) เครื่องลูกข่าย (Client) จะทำการส่ง SYN Flag ระบุหมายเลขพอร์ตที่ต้องการติดต่อไปยังเครื่องให้บริการด้วยการระบุหมายเลขลำดับของข้อมูล (ISN - Initial Sequence Number)
2) เครื่องเซิร์ฟเวอร์เมื่อได้รับข้อมูลเซกเมนต์จากข้อ 1 ก็จะตอบกลับด้วยการเพิ่มค่า ISN ที่ได้รับขึ้นอีก 1 พร้อมทั้งระบุหมายเลขลำดับ (ISN) ของตนเอง และเปิด SYN กับ ACK Flag
3) เครื่องลูกข่ายเมื่อได้รับการตอบกลับจากเครื่องแม่ข่ายตามข้อ 2 ก็จะทำการตอบรับกลับไปโดยการเพิ่มค่า ISN ของเซิร์ฟเวอร์ขึ้นอีก 1 และเปิด ACK Flag เมื่อผ่านการสร้างการเชื่อมต่อทั้ง 3 ขั้นตอนแล้ว ตอนนี้ทั้งเครื่องลูกข่ายและเครื่องแม่ข่าย ก็สามารถติดต่อสื่อสารกันได้แล้ว

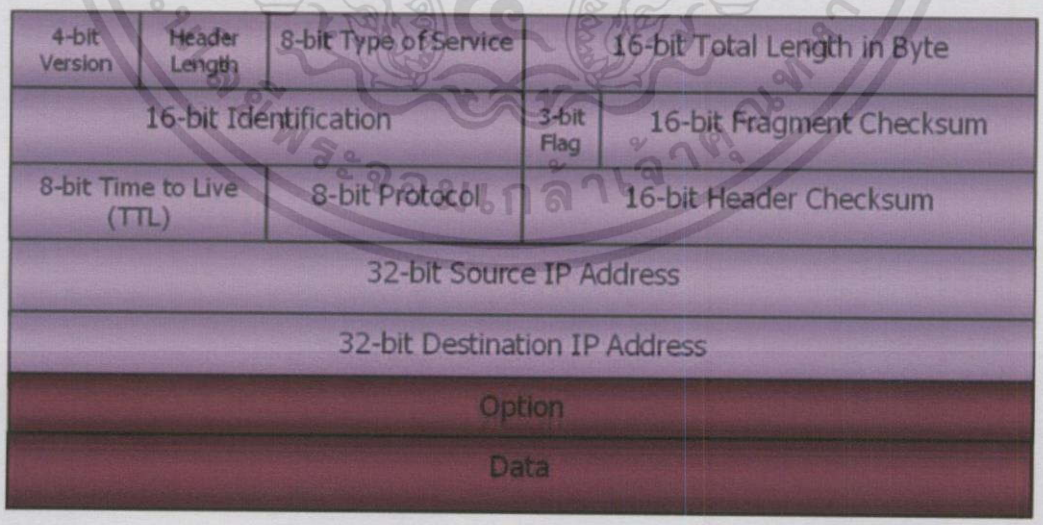


รูปที่ 2.16 การทำงานของกระบวนการจับมือสามขั้นตอน (3-Way handshake)

2.2.3.3 โครงสร้างของ IP

IP (Internet Protocol) เป็นแบบวิธีการ (protocol) ที่ใช้ในการส่งข้อมูลจากอุปกรณ์เครื่องหนึ่งไปยังอีกเครื่องหนึ่งในอินเทอร์เน็ต (Internet) แต่ละอุปกรณ์จะต้องมีตัวบ่งบอกถึงที่อยู่อย่างน้อยหนึ่งที่อยู่ (address) ซึ่งไม่ซ้ำกับอุปกรณ์อื่นในอินเทอร์เน็ต เพื่อให้สามารถรับและส่งข้อมูลข้อมูลได้ โดยในการส่งแต่ละครั้งข้อมูลจะถูกแบ่งเป็นกลุ่มข้อมูลหรือที่เรียกว่า แพ็กเกต (Packet) แต่ละชุดจัดจะเก็บที่อยู่ของผู้ส่งและผู้รับ ไปยังเกตเวย์ (Gateway) ซึ่งจะอ่านที่อยู่ปลายทาง และจัดการส่งผ่านเครือข่ายต่อไปยังที่อยู่จุดหมายปลายทางได้

IP ทำงานในระดับชั้น Internet Layer ของตามรูปแบบของทีซีพี/ไอพี ทำหน้าที่จัดการเกี่ยวกับแอดเดรสและข้อมูล และควบคุมการส่งข้อมูลบางอย่างที่ใช้ในการหาเส้นทางของแพ็กเกต ซึ่งกลไกในการหาเส้นทางของ IP จะมีความสามารถในการหาเส้นทางที่ดีที่สุด และสามารถเปลี่ยนแปลงเส้นทางได้ในระหว่างการส่งข้อมูล และมีระบบการแยกและประกอบแพ็กเกต เพื่อรองรับการส่งข้อมูลที่มีขนาด MTU (Maximum Transmission Unit) ที่แตกต่างกัน การเชื่อมต่อของ IP เพื่อทำการส่งข้อมูล จะเป็นแบบไม่กำหนดการเชื่อมต่อ (Connectionless) หรือเกิดเส้นทางการเชื่อมต่อในทุกๆ ครั้งของการส่งข้อมูล 1 แพ็กเกตโดยจะไม่ทราบถึงแพ็กเกตที่ส่งก่อนหน้าหรือส่งตามมา แต่การส่งข้อมูลในหนึ่งแพ็กเกตอาจจะเกิดการส่งได้หลายครั้งในกรณีที่มีการแบ่งข้อมูลออกเป็นส่วนย่อยๆ (fragmentation) และถูกนำไปรวมเป็นแพ็กเกตเดิมเมื่อถึงปลายทาง



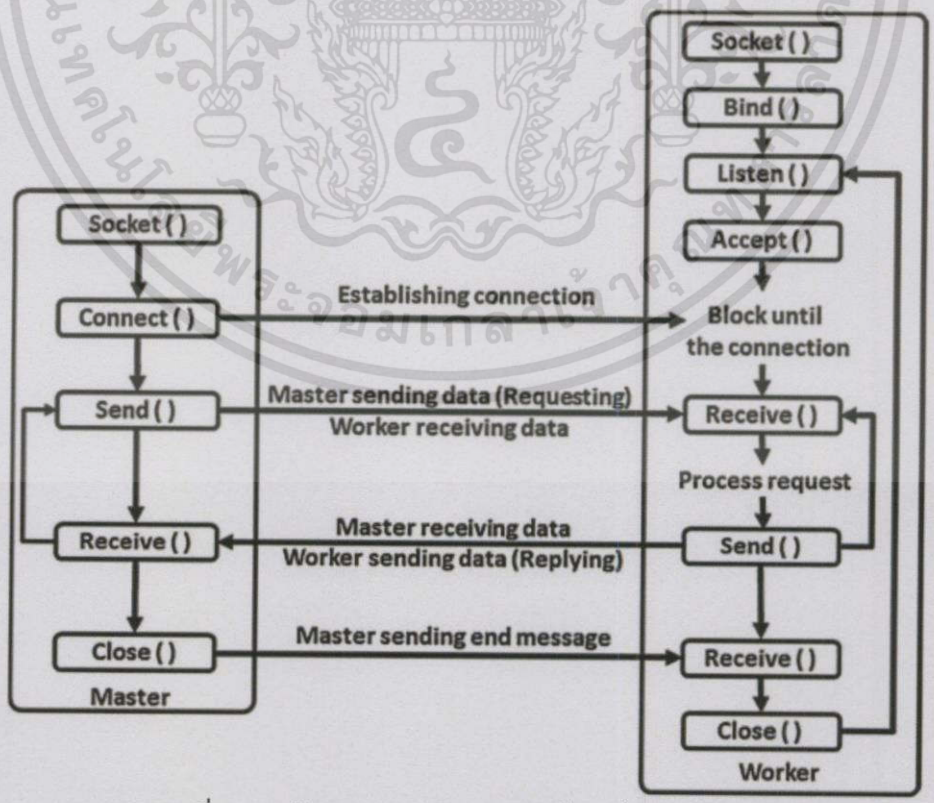
รูปที่ 2.17 โครงสร้างของโพรโทคอลไอพี

2.2.4. ซ็อกเก็ต (Socket)

ซ็อกเก็ต (Socket) คือกลุ่มของหมายเลขพอร์ตและหมายเลขไอพี ซึ่งจะเป็นตัวบ่งชี้ที่เฉพาะเจาะจง ใช้ในการสื่อสารระหว่างกระบวนการ (IPC: Inter Process Communication) ทำให้ติดต่อรับส่งข้อมูลภายในกระบวนการได้

ลักษณะการทำงานในฝั่งผู้ให้บริการ เริ่มต้นจากการสร้างซ็อกเก็ต ซึ่งจะต้องมีการระบุประเภทของซ็อกเก็ต และทำการจองทรัพยากรในระบบ จากนั้นก็ทำการยึดเหนี่ยว (Bind) กับหมายเลขพอร์ตที่ต้องการ และทำการรอรับฟัง (Listen) การขอเริ่มการเชื่อมต่อ และเมื่อได้รับคำขอเริ่มก็จะยอมรับการเชื่อมต่อ (Accept) จากนั้นจะเข้าสู่การรอรับข้อมูลจากผู้ใช้บริการ (Request) เมื่อได้รับข้อมูลแล้วก็จะทำการประมวลผล (Process) และส่งข้อมูลกลับ (Send) ไปยังผู้ใช้บริการ และจะวนแบบนี้ไปเรื่อยๆ จนกว่าจะได้รับคำขอปิดการเชื่อมต่อ ก็จะเริ่มกระบวนการปิด (Close) การเชื่อมต่อกับผู้ใช้บริการนั้นและกลับมาสู่สถานะรอรับฟังเหมือนเดิม

ลักษณะการทำงานในฝั่งผู้ใช้บริการ เริ่มต้นจากการสร้างซ็อกเก็ตเช่นเดียวกันกับผู้ให้บริการ และเมื่อสร้างเสร็จแล้วก็จะทำการติดต่อไปยังผู้ให้บริการ เมื่อกระบวนการเชื่อมต่อเรียบร้อย ก็จะเริ่มรับและส่งข้อมูลระหว่างผู้ใช้บริการและผู้ให้บริการ และเมื่อเสร็จสิ้นแล้วก็ทำการส่งคำขอเริ่มกระบวนการปิดการเชื่อมต่อไปยังผู้ให้บริการและสิ้นสุดการทำงานลง เป็นไปดังรูปที่ 2.18



รูปที่ 2.18 ลักษณะการทำงานของซ็อกเก็ต (Socket)

2.3 ข้อมูลเบื้องต้นของ Unreal Engine และ UDK

2.3.1 เกมเอนจิ้น

เกมเอนจิ้นคือ เครื่องมือที่ถูกสร้างและออกแบบมาเพื่อพัฒนาเกมโดยเฉพาะ โดยตัวเกมเอนจิ้นนั้น จะเก็บรวบรวมกรอบการทำงาน (Framework) ที่ช่วยในการสร้างเกมไว้เพื่อให้สามารถใช้งานได้สะดวก ตัวเกมเอนจิ้นนั้นส่วนมากแล้วจะประกอบไปด้วยตัวจัดการพื้นฐานในด้านต่างๆเช่น

1) ระบบประมวลผลกราฟิก
2) ระบบจัดการการประมวลผลทางด้านฟิสิกส์และตรวจสอบการชนต่างๆ
3) ระบบเสียง
4) ระบบจัดการสคริปต์ (Script) ในเกม
5) การเคลื่อนไหวของกราฟิกในเกม
6) ปัญญาประดิษฐ์
7) ระบบเครือข่าย
8) การจัดการระบบข้อมูลความจำ (Memory Management)
9) การแบ่งสายโยงใย (Thread) ในการทำงาน
10) ระบบภาษา
11) และอื่นๆ

การใช้นำเกมเอนจิ้นที่เหมาะสมมาใช้นั้น จะเป็นตัวช่วยให้การพัฒนาเกมสะดวกขึ้นเพราะมีการนำเครื่องมือที่เคยมีอยู่ก่อนแล้วมาใช้ใหม่ หรือ ปรับปรุงเพิ่มเพื่อสร้างเกมใหม่ได้โดยไม่ต้องสร้างเครื่องมือทั้งหมดใหม่ตั้งแต่ต้น และเป็นเครื่องมือที่ทำให้การพอร์ต (Port) ซึ่งก็คือ การแปลงเกมไปทำงานบนแพลตฟอร์ม (Platform) อื่นๆ เช่น PS3 หรือ PSP สะดวกขึ้นมากด้วย

ในปัจจุบัน มีเกมเอนจิ้นหลากหลายแบบให้เลือกใช้งานโดยที่แต่ละอันก็มีจุดเด่นของตัวเองแตกต่างกันไปตามประเภทของเกมที่จะพัฒนา นอกจากนี้ยังมีเกมเอนจิ้นบางค่ายที่เป็นแบบเปิดเผยโปรแกรมต้นฉบับ (Open Source) ทำให้บุคคลทั่วไปสามารถปรับปรุงระบบของตัวเอนจิ้นเพิ่มเติมได้โดยไม่ผิดลิขสิทธิ์มีผู้พัฒนาและแจกจ่ายระบบสคริปต์ต่างๆ เพื่อให้เหมาะแก่เกมแนวต่างๆที่เคยมีมาแล้ว สะดวกต่อการเริ่มต้นศึกษา

2.3.2 Unreal Engine

Unreal Engine เป็นเกมเอนจิ้นที่พัฒนาขึ้นโดยบริษัท Epic Games โดยปรากฏสู่สาธารณะเป็นครั้งแรกในปี 1998 โดยใช้ในการพัฒนาเกมยิงมุมมองบุคคลที่หนึ่งชื่อว่า Unreal (ซึ่งเป็นที่มาของชื่อเอนจิ้น) ซึ่งภายหลังถูกขยายไปใช้กับเกมแนวอื่นๆและประสบความสำเร็จหลายเกม นอกจากนี้

ภาษาที่ใช้พัฒนาเอนจิ้นขึ้นมาคือ C++ ซึ่งสามารถพอร์ทลงอุปกรณ์อื่นๆได้ง่าย ทำให้เอนจิ้นนี้เป็นที่นิยมสำหรับผู้พัฒนาเกมมากมาย

Unreal Engine เวอร์ชันล่าสุดคือ Unreal Engine 3 ซึ่งสามารถทำงานกับ Microsoft's DirectX 9 (Windows and Xbox 360), DirectX 10 (Windows Vista) and DirectX 11 (Windows 7 and later), OpenGL (OS X, Linux, PlayStation 3, Wii U, iOS, Android), Stage 3D (Adobe Flash Player 11) และ JavaScript/WebGL (HTML5)

2.3.3 Unreal Development Kit

Unreal Development Kit (UDK) คือ ชุดโปรแกรมที่ใช้ในการพัฒนา (SDK: Software Development Kit) ที่ใช้ในการพัฒนาเกม โดยใช้ Unreal Engine 3 ซึ่งเป็นเกมเอนจิ้นที่มีคุณภาพในระดับสูง ถูกพัฒนาขึ้นโดยบริษัท Epic Games และแจกจ่ายให้ใช้งานได้ฟรีเพื่อการศึกษา แต่ไม่อนุญาตให้มีการปรับแต่งตัวเอนจิ้นของเกม

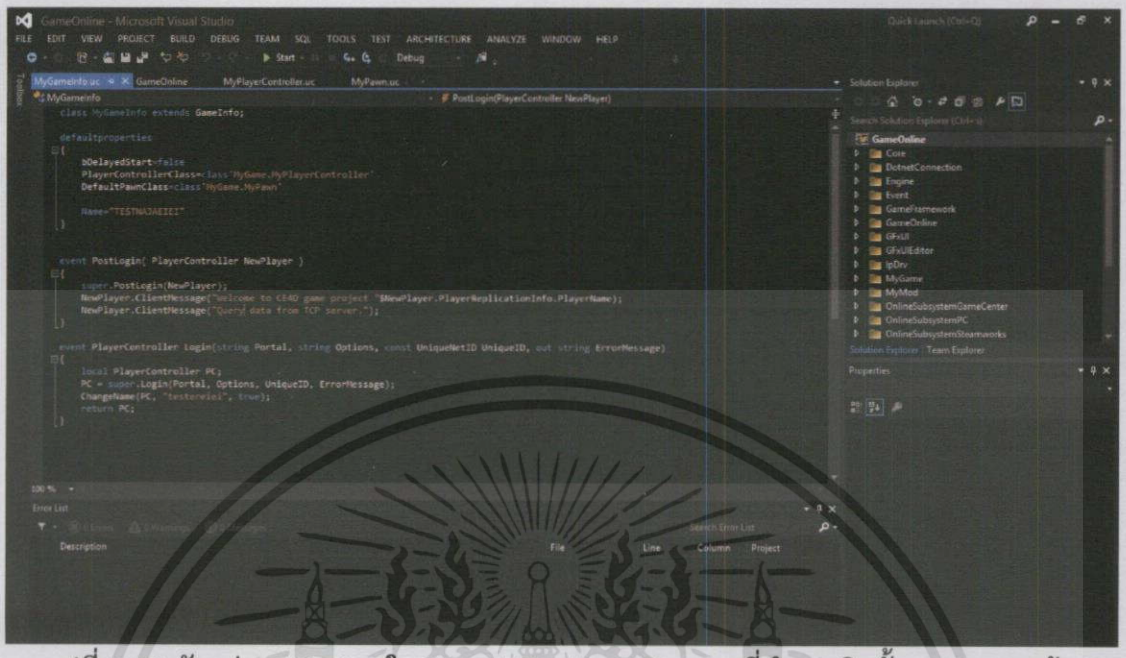
นอกจากสคริประดับพื้นฐานที่มีมาให้ระดับหนึ่งแล้ว UDK นั้นยังมีโปรแกรมสำเร็จรูปสำหรับทำงานกับองค์ประกอบศิลป์ของเกม ทำให้การพัฒนาเกมเป็นไปได้อย่างรวดเร็วและเข้าใจได้ง่ายยิ่งขึ้น นอกจากนี้ยังมีสารานุกรมออนไลน์และบทเรียนในเว็บไซต์ของทางบริษัทให้ใช้งานอีกด้วย

2.3.4 องค์ประกอบของการพัฒนาเกมด้วย UDK

2.3.4.1 Unreal script

ตัวเกมที่พัฒนาโดย UDK ใช้ภาษาที่เรียกว่า UnrealScript หรือ UScript ในการเขียนโครงสร้าง ซึ่ง UScript เป็นการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming, OOP) ที่ใช้งานในรูปแบบเดียวกันกับ Java และ C++ ตัวสคริปต์สามารถแก้ไขได้ด้วยตัวแก้ไขข้อความ (Text Editor) ใดๆก็ได้ โดย UScript นั้นสามารถนำมาพัฒนาใน Microsoft Visual Studio ได้โดยผ่านโปรแกรมเสริม (Plug-in) ที่ชื่อว่า nFringe

UScript สามารถเรียกใช้งานไลบรารี่ (Libraries) อื่นๆที่อยู่นอก UDK ได้ด้วยการใช้ฟังก์ชัน DLLBind ทำให้สามารถทำงานได้หลากหลายและยืดหยุ่น

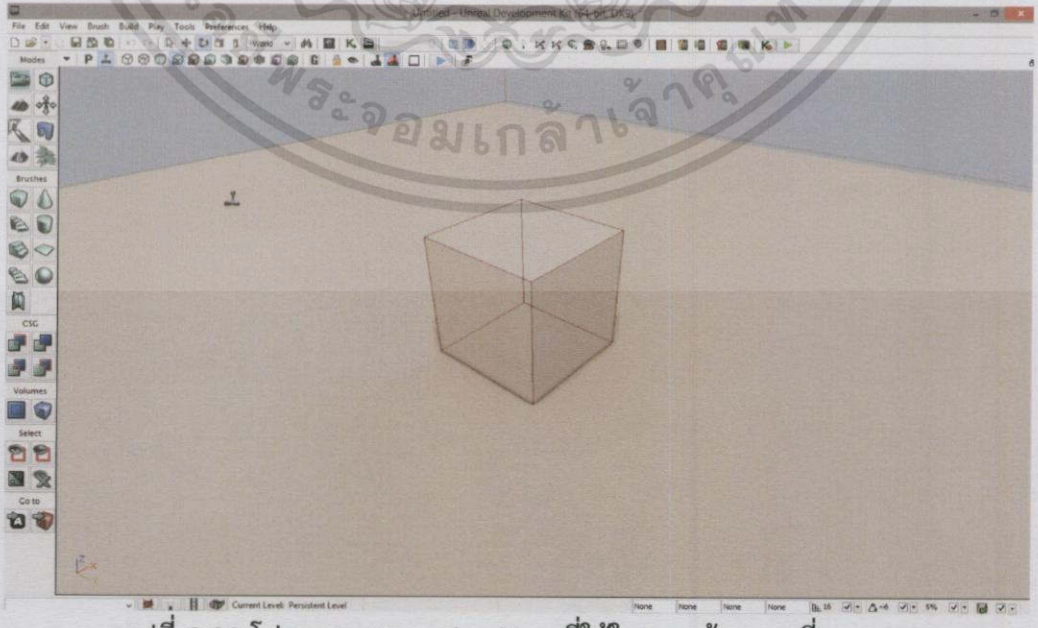


รูปที่ 2.19 ตัวอย่าง UScript ใน Visual Studio Ultimate ที่ทำการติดตั้ง nFringe แล้ว

2.3.4.2 UDK Editor

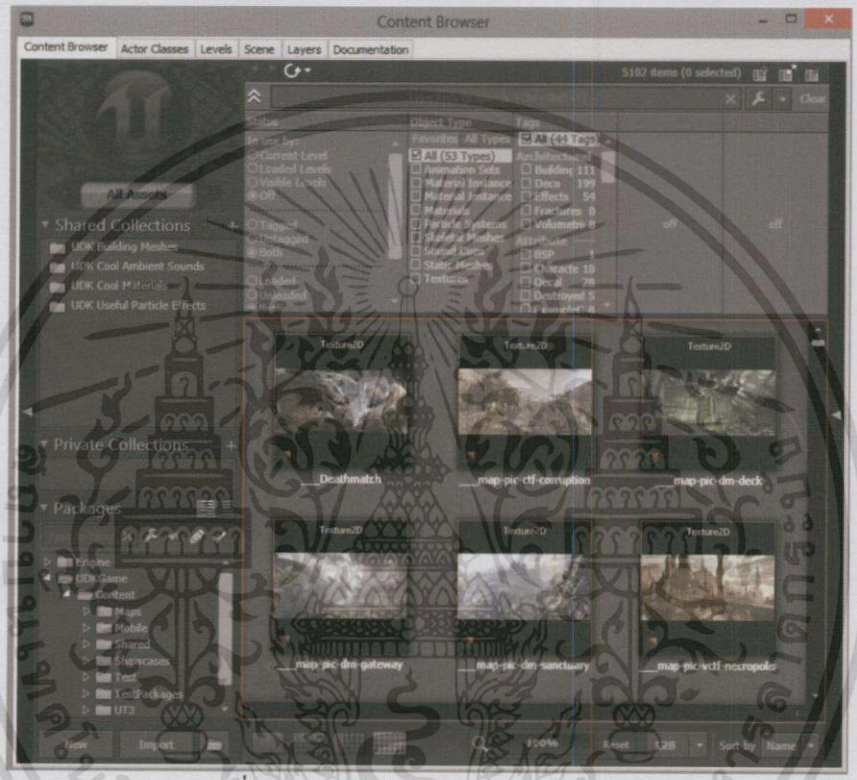
โปรแกรม UDK Editor มีขึ้นเพื่อใช้ในการจัดการและเชื่อมต่อองค์ประกอบศิลป์ในเกมเข้ากับโครงสร้างที่เขียนไว้ด้วย UScript หน้าที่หลักๆของ UDK Editor มีอยู่ 3 หน้าที่ด้วยกัน

1) การสร้างแผนที่ในเกม สามารถทำได้ผ่านหน้าต่างหลักของโปรแกรม โดยสามารถเลือกใช้เครื่องมือของตัวโปรแกรมในการสร้างแผนที่สำหรับการเล่นได้



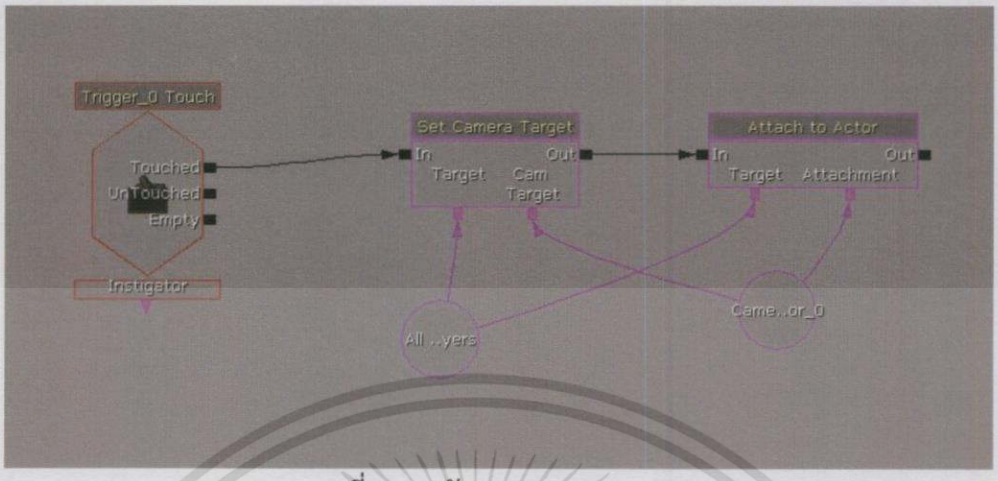
รูปที่ 2.20 โปรแกรม Unreal editor ที่ใช้ในการสร้างแผนที่ของเกม

2) การจัดการองค์ประกอบศิลป์ สามารถทำได้ด้วยการใช้การค้นดูเนื้อหา (Content Browser) ของโปรแกรมซึ่งนอกจากจะใช้องค์ประกอบศิลป์ที่มีมาให้กับตัว UDK แล้วยังสามารถ นำเข้า (Import) มาจากภายนอกได้



รูปที่ 2.21 หน้าต่าง Content Browser

3) การสร้างสคริปต์ในเกมที่อิงกับตัวแผนที่ สามารถทำได้โดยเครื่องมือที่เรียกว่า Kismet ซึ่งเป็นเครื่องมือที่ช่วยในการสร้างตัวนำเหตุการณ์ (Event Trigger) ภายในเกมด้วยรูปแบบของแผนภาพ (Diagram) ซึ่งง่ายต่อการใช้งาน เป็นส่วนสำคัญที่ใช้เชื่อมต่อ โครงสร้างของเกมที่เขียนด้วย UScript เข้ากับแผนที่ของเกมที่สร้างด้วยองค์ประกอบศิลป์ต่างๆ



รูปที่ 2.22 ลักษณะของ Kismet

2.4 ภาพรวมของเกมแบบผู้เล่นหลายคน

2.4.1 โลกต่อเนื่อง (Persistent World)

โลกต่อเนื่อง คือ โลกจำลองที่แม้ว่าจะไม่มีผู้เล่นหรือผู้ใช้งานอยู่ โลกนั้นๆก็ยังคงดำเนินต่อไป โลกต่อเนื่องถูกออกแบบขึ้นเพื่อให้ผู้ใช้งานจำนวนมากสามารถใช้บริการได้พร้อมกันและสามารถปฏิสัมพันธ์ระหว่างผู้ใช้งานได้ในรูปแบบของสภาพแวดล้อมรวม

ข้อดีของการใช้โลกต่อเนื่องก็คือสามารถควบคุมการทำงานของระบบได้โดยการใช้กฎร่วมกันระหว่างผู้ใช้ ด้วยเหตุนี้จึงสามารถลดอันตราการเกิดความผิดพลาดและตรวจสอบแก้ไขได้ง่ายขึ้น

2.4.2 แนวเกม (Genres)

MMO นั้นสามารถแบ่งออกได้หลายประเภทเช่นเดียวกับแนวเกมทั่วๆไปซึ่งมีจำนวนมาก โดยในที่นี้จะคัดมาเฉพาะแนวที่สำคัญและเหมาะสมกับเนื้อหาในรายงานเล่มนี้

2.4.2.1 เกมเล่นตามบทบาท (Role-Playing Game)

เกมเล่นตามบทบาท หรือ เกมอาร์พีจี (RPG) คือเกมประเภทหนึ่งที่ผู้เล่นสมมุติรับบทเป็นตัวละครหนึ่งในเกม โดยเล่นตามกฎกติกาของเกมผ่านการป้อนคำสั่งและเลือกเงื่อนไขที่เกมกำหนดมา โดยผลลัพธ์ที่เกิดจะแตกต่างกัน ตามเงื่อนไขที่เลือก

เกมแบบผู้เล่นหลายคนที่พบได้บ่อยที่สุดในอุตสาหกรรมเกมและค่อนข้างเป็นที่นิยม เนื่องจากเกมแนวนี้จะเน้นให้ผู้เล่นสัมผัสบรรยากาศและรู้สึกว่าตนเองเป็นส่วนหนึ่งของโลกต่อเนื่องในเกมได้ รวมถึงสามารถเห็นผู้เล่นคนอื่นๆได้อีกด้วย

เนื่องจากเนื้อเรื่องในเกมไม่ใช่ชีวิตจริง จินตนาการจึงมีบทบาทสูงในแนวทางการวางเนื้อเรื่อง ซึ่งจินตนาการนี้เป็นสิ่งสำคัญในการเชื่อมต่อตัวละครในเกมกับผู้เล่นเข้าด้วยกัน ถ้าผู้เล่นไม่สามารถเข้าใจเนื้อเรื่อง หรือ ไม่มีจินตนาการในเนื้อเรื่องนั้นๆ อาจทำให้ความรู้สึกในการดำเนินเนื้อเรื่องไม่ราบรื่น หรือทำให้ความสนุกในการเล่นเกมลดลง

2.4.2.2 เกมแอคชั่น (Action Game)

เกมแอคชั่นเป็นเกมที่เน้นการเคลื่อนไหวและความเร็วของประสาทสัมผัสในการเล่น ผู้เล่นจะได้เล่นเป็นตัวละครที่จะต้องเคลื่อนที่ไปบนสภาพแวดล้อมของเกมเพื่อทำภารกิจให้สำเร็จลุล่วง ตัวอย่างเช่น เดินทางไปให้ถึงจุดหมาย หรือ ต่อสู้กับตัวละครอื่น

เกมเดินยิง (Shooter Game) เป็นเกมแอคชั่นชนิดหนึ่งที่ได้รับความนิยมสูง ซึ่งโดนส่วนมากจะเป็นการต่อสู้กันระห่างตัวละครด้วยอาวุธต่าง เช่น ปืน แต่ก็มีบางเกมที่ใช้อาวุธชนิดอื่นๆเช่นกัน

ในเกมเดินยิงนั้นก็มีหลายประเภทด้วยกันโดยแบ่งประเภทตามมุมกล้อง เนื่องจากมุมกล้องมีผลมากต่อความรู้สึกขณะเล่น เกมเดินยิงแนวที่กำลังเป็นที่นิยมในปัจจุบันคือ เกมเดินยิงมุมมองบุคคลที่หนึ่ง (First Person Shooter) และ เดินยิงมุมมองบุคคลที่สาม (Third Person Shooter)

บทที่ 3
การออกแบบและโครงสร้างของระบบ

3.1. การออกแบบเกม

3.1.1 รายละเอียดของเกม

เกมใช้ชื่อว่า มหายุทธแบทเทิลกราวน์ (Mahayuth Battleground) มีกลุ่มผู้เล่นเป้าหมายทุกเพศอายุระหว่าง 15-28 ปี เป็นเกมแนวแอ็คชั่น (Action) และมีลักษณะการเล่นเกมที่เล่นหลายคนร่วมกับผู้เล่นบุคคลอื่นด้วย โดยผู้เล่นจะถือบทบาทเป็นแม่ทัพในการนำทหารมาเพื่อสู้รบ โดยใช้แนวความคิดในการดึงตัวละครในโลกแฟนตาซีประสานกับวรรณคดีไทยเป็นฐานในการออกแบบดำเนินการโดยการจำลองการเดินทางและการสู้รบ

3.1.2 แนวคิดหลักในการออกแบบระบบการเล่น

1) รองรับผู้เล่นได้จำนวนหลายคน
2) พัฒนาได้สะดวกบน UDK
3) สภาพแวดล้อมในเกมมีการเปลี่ยนแปลงตลอดเวลา
4) ผู้เล่นสามารถปรับเปลี่ยนวิธีเล่นไปตามสภาพแวดล้อมในเกม
5) ออกแบบเกมโดยการประยุกต์ใช้วรรณคดีไทย

3.1.3 ขอบเขตการออกแบบที่สรุปได้จากแนวคิดหลัก

1) ระบบการเล่นหลักๆเป็นแนว Action Shooter ในมุมมองบุคคลที่สาม เนื่องจากเอื้อเฟื้อต่อการพัฒนาเกมด้วย UDK
2) เนื่องจากสภาพแวดล้อมจะต้องมีการเปลี่ยนแปลงตลอด ระบบเกมจะต้องอนุญาตให้ผู้เล่นได้สัมผัสกับสภาพแวดล้อมอย่างเต็มที่ ดังนั้นรูปแบบเกมที่เหมาะสมจะเป็นแบบเปิดหรือ Open World ซึ่งผู้เล่นมีอิสระที่จะเดินทางไปยังที่ใดก็ได้ในโลกของเกมๆนั้น
3) ผู้เล่นจะสามารถจัดกำลังพลทหารของตนเอง เพื่อนำมาต่อสู้กับฝ่ายศัตรูได้
4) เพื่อให้สภาพแวดล้อมมีผลกับผู้เล่น ผู้เล่นและฝั่งตรงข้ามจะเผชิญหน้ากันเป็นกลุ่ม ซึ่งมีจุดเด่นจุดด้อยหักล้างกัน โดยสิ่งเหล่านี้มีผลต่อการตัดสินใจของระบบปัญญาประดิษฐ์
5) สภาพแวดล้อมที่เปลี่ยนแปลงตลอดจะต้องไม่ทำให้ผู้เล่นรู้สึกว่าไม่สมดุล ระบบปัญญาประดิษฐ์ของเกมจึงมีจุดประสงค์เพื่อปรับสภาพแวดล้อมที่ผู้เล่นสัมผัสให้เกิดความ

เปลี่ยนแปลงที่สมดุล ไม่ใช่เป็นการต่อสู้กับผู้เล่นโดยตรง ดังนั้นจึงเลือกใช้ระบบโครงข่ายประสาทเทียมในการพัฒนาระบบปัญญาประดิษฐ์ เพราะการทำให้เกิดการเปลี่ยนแปลงที่สมดุลนั้นขึ้นอยู่กับพฤติกรรมของผู้เล่นในภาพรวม

6) เนื่องจากสภาพแวดล้อมในเกมสามารถเปลี่ยนแปลงได้ตลอด จึงไม่สามารถใช้เรื่องราวจากวรรณคดีไทยได้โดยตรง ต้องทำการสร้างโลกแฟนตาซีใหม่ขึ้นมา โดยใช้วรรณคดีไทยมาเป็นเพียงแค่แรงบันดาลใจในการออกแบบ

จากแนวคิดในการออกแบบข้างต้น สามารถนำมาต่อยอดและเพิ่มรายละเอียด ในการออกแบบขั้นต่อๆไปได้

3.1.4 เนื้อเรื่อง

ครั้งหนึ่ง ณ ดินแดนสุวรรณภูมิที่อุดมสมบูรณ์และสวยงามทั้งยังเต็มไปด้วยมนตราอันลี้ลับ จักรวรรดิอยุธยาอันยิ่งใหญ่ผู้รวบรวมมวลมนุษย์มากมายหลายเชื้อชาติไว้ใต้ธงพื้นเดียว ได้ยกไพร่พลไปเข้าตีอาณาจักรลงกาของเหล่ายักษ์ องค์จักรพรรดิผู้ยิ่งใหญ่แห่งจักรวรรดิอังกอร์ได้นำชัยชนะมาสู่มวลมนุษย์ด้วยเวทมนต์ดำอันลี้ลับ ทว่าเขากลับไม่หยุดเพียงแค่นั้นและถลำลึกเข้าสู่วิถีแห่งมนต์ดำจนนำมาซึ่งความล่มสลายของจักรวรรดิ เมื่อเห็นโอกาสอันดี เหล่ายักษ์ที่เหลือรอดจึงนำกองกำลังออกล้างแค้นเหล่ามนุษย์ จนเกิดการสู้รบขึ้นทั่วดินแดนสุวรรณภูมิ

เมื่อสัญญาณแห่งสงครามใหญ่มาถึง เหล่าขุนศึกมากมายได้ปรากฏตัวขึ้น นำพายอดนักรบเข้าสู่สมรภูมิ ผู้เล่นจะได้สวมบทบาทเป็นขุนศึกของจักรวรรดิอยุธยา พากองกำลังของตนเดินทางไปในดินแดนสุวรรณภูมิและจัดกระบวนกองกำลังของตัวเอง เพื่อทำการต่อสู้กับกองกำลังของฝ่ายยักษ์

3.1.5 รูปแบบและกติกาการเล่น

3.1.5.1 แบบวิธีเดินทาง (Travel Mode)

ในแบบวิธีเดินทางนั้น จะมีตัวละครเดินทางไปบนแผนที่ของเกม โดยตัวละครเหล่านั้นมีทั้งตัวละครของผู้เล่นและตัวละครที่ควบคุมโดยปัญญาประดิษฐ์ ตัวละครแต่ละตัวสามารถเดินทางไปบนแผนที่ได้อย่างอิสระ

3.1.5.2 กองกำลัง (Warband)

ตัวละครแต่ละตัวในแบบวิธีเดินทางนั้นแทนกองกำลังหนึ่งกองกำลัง ซึ่งภายในแต่ละกองจะมีสมาชิกต่างๆกันไปเรียกว่า ยูนิต (Unit) กองกำลังที่ควบคุมโดยผู้เล่นจะมียูนิตของผู้เล่นที่เรียกว่า ขุนศึก (Warlord) รวมอยู่ในสมาชิกด้วยในขณะที่กองกำลังที่ควบคุมโดยระบบปัญญาประดิษฐ์นั้นจะมีเพียงแต่ยูนิตธรรมดาทั่วไปเท่านั้น

3.1.5.3 แบบวิธีสู้รบ (Battle Mode)

เมื่อกองกำลังใดกองกำลังหนึ่งเกิดปะทะกับกองกำลังอื่น เกมจะตัดเข้าสู่แบบวิธีสู้รบ ยูนิตของแต่ละฝั่งจะถูกสร้างขึ้นบนแผนที่ย่อยและทำการต่อสู้กัน โดยฝ่ายใดที่ยูนิตหมดสภาพการต่อสู้ทั้งกองกำลังก่อนจะเป็นฝ่ายแพ้ เมื่อได้ผลแพ้ชนะของการสู้รบ

หากกองกำลังที่แพ้ควบคุมโดยปัญญาประดิษฐ์ กองกำลังนั้นจะโดนลบหายไปจากแผนที่โลก แต่ถ้าเป็นผู้เล่น กองกำลังนั้นจะถูกสุ่มไปเกิดใหม่บนแผนที่

ในแบบวิธีสู้รบของยูนิตทุกยูนิตจะสามารถเคลื่อนไหวได้อย่างอิสระและสามารถโจมตีและป้องกันคู่ต่อสู้ได้

3.1.5.4 ยูนิต (Unit)

ยูนิต คือ ส่วนประกอบหลักของเกม มีหน้าที่แทนตัวละครหนึ่งตัว ซึ่งจะร่วมกับยูนิตฝั่งเดียวกันเข้าทำการสู้รบกับยูนิตของฝ่ายตรงข้ามเพื่อตัดสินผลแพ้ชนะ โดยยูนิตมีหลักการต่อสู้ที่ตายตัว คือจะทำการโจมตียูนิตคู่ต่อสู้หากระยะอาวุธถึง หากระยะอาวุธไม่ถึงยูนิตจะทำการเดินไปยังยูนิตคู่ต่อสู้ที่อยู่ในระยะมองเห็นและทำการโจมตี หากไม่เห็นก็จะทำการสุ่มเดินไปในทิศทางต่างๆ เช่นนี้ไปเรื่อยๆ โดยยูนิตแต่ละยูนิตมีค่าต่างๆที่เกี่ยวข้องดังนี้

1.) Health หรือค่าพลังชีวิต เมื่อยูนิตถูกโจมตีจะทำให้พลังชีวิตลด ยูนิตนั้นจะหมดสภาพการต่อสู้หากพลังชีวิตหมด
2.) Shield หรือค่าเกราะ ปริมาณของเกราะจะทำให้พลังชีวิตลดช้าลง โดยนำไปลดค่าเกราะแทน

3.1.5.4 ประเภทของยูนิต (Type of unit)

ยูนิตในเกมจะมีทั้งหมด 3 ประเภทคือ

1.) ทหารประเภทโจมตีระยะไกล (ทหารธนู) ลักษณะคือ ถืออาวุธยิงระยะไกล ความแรงต่ำ เกราะน้อย แพ้ทางนักดาบ
2.) ทหารประเภทโจมตีระยะใกล้เสริมเกราะ (ทหารดาบ) ลักษณะคือ ถืออาวุธฟันระยะใกล้ ความแรงปานกลาง เกราะค่อนข้างมาก แพ้ทางนักหอก
3.) ทหารประเภทโจมตีระยะใกล้ (ทหารหอก) ลักษณะคือ ถืออาวุธฟันระยะใกล้ ความแรงค่อนข้างแรง เกราะปานกลาง แพ้ทางนักธนู

3.1.6 ระบบการควบคุมของผู้เล่น

3.1.6.1 Launcher

ผู้เล่นทำการเข้าสู่ระบบด้วยชื่อ (Username) และรหัสผ่าน (Password) จากนั้นจะมีการตรวจสอบรุ่น (Version) ก่อนว่าเป็นรุ่นปัจจุบันหรือยัง ถ้ายังจะต้องมีการปรับปรุงโปรแกรมให้เป็นปัจจุบัน (Update) จากนั้นจึงเข้าสู่ตัวเกมได้

3.1.6.2 Travel Mode

เมื่อเข้าสู่ตัวเกม ผู้เล่นจะเข้าสู่แบบวิธีเดินทางและมองเห็นตัวละคนของตนเองจากมุมมองแบบบุคคลที่สาม ผู้เล่นสามารถควบคุมกองกำลังของตนเองได้โดยใช้ WASD Direction Control ควบคุมการเคลื่อนที่ของตัวละครโดยปุ่ม W ไปข้างข้าง A ไปทางซ้าย D ไปทางขวา และ S ถอยหลัง และเข้าปะทะกับฝ่ายยักษ์เพื่อเข้าต่อสู้โดยการเดินเข้าชนขุนศึกของฝั่งยักษ์ซึ่งเป็นตัวแทนของกองกำลังของฝ่ายศัตรู

3.1.6.3 Battle Mode

เมื่อเข้าสู่แบบวิธีสู้รบ ยูนิตทุกยูนิตรวมถึงผู้เล่นจะทำการต่อสู้กันตามกติกา Team Death match ของ Unreal Tournament โดยใช้มุมกล้องจากมุมมองบุคคลที่สามแบบเหนือไหล่ ผู้เล่นจะเป็นผู้ควบคุมขุนศึกเข้าทำการต่อสู้พร้อมกับยูนิตของตัวเอง โดยมีการควบคุมพื้นฐานดังนี้

1) WASD Direction Control ควบคุมการเคลื่อนที่ของตัวละครโดยปุ่ม W ไปข้างหน้า A ไปทางซ้าย D ไปทางขวา และ S ถอยหลัง
2) คลิกซ้ายเพื่อทำการโจมตี
3) กดปุ่มหมายเลขเพื่อเปลี่ยนอาวุธที่ใช้งาน
4) กดปุ่ม Space เพื่อกระโดด

โดยยูนิตอื่นๆนอกจากผู้เล่นในแบบวิธีสู้รบ จะถูกควบคุมด้วยปัญญาประดิษฐ์ของ Unreal Engine ตามกฎของ Unreal Tournament ซึ่งมีกฎการทำงานที่แน่นอน

3.1.7 สภาพแวดล้อมในเกม (Game Environment)

กองกำลังอื่นๆในเกมที่ไม่ได้ควบคุมโดยผู้เล่น จะมีพฤติกรรมการเกิด (Spawn) ที่ถูกควบคุมโดยระบบปัญญาประดิษฐ์ จะมีแม่ทัพซึ่งเสมือนว่ามีหน้าที่ควบคุมปริมาณและขนาดของกองกำลังที่จะถูกสร้างขึ้น รวมถึงชนิดของยูนิตและการเดินทางของกองกำลังแต่ละกองกำลังที่ถูกสร้างขึ้นด้วย สำหรับการเดินทางของกองกำลังที่ถูกสร้างโดยแม่ทัพนั้น จะถูกสั่งให้เดินทางแบบสุ่มเดิน

จำนวนยูนิตของกองกำลังแต่ละกองกำลังของยักษ์นั้นจะปรับเมื่อมีผู้เล่นจะเข้ามาทำการต่อสู้ โดยที่จะปรับก่อนเกิดการต่อสู้ กองกำลังของฝ่ายยักษ์นั้นจะนำค่าข้อมูลจำนวนทหารแต่ละประเภท

ของกองกำลังฝ่ายผู้เล่น นำไปทำนายที่โครงข่ายประสาทเทียม หากผลการทำนายปรากฏว่ากองกำลังฝ่ายยักษ์จะชนะ กองกำลังของยักษ์ก็จะไม่ปรับอะไร และทำการเริ่มต่อสู้กับผู้เล่นเลย แต่ผลการทำนายปรากฏว่ากองกำลังฝ่ายยักษ์จะแพ้ ฝ่ายยักษ์ก็จะทำการปรับจำนวนทหารตามกฎการปรับซึ่งจะกล่าวในหัวข้อถัดไป

3.1.8 การทำนายผลการต่อสู้ (Battle Result Prediction)

เมื่อเกิดการต่อสู้ขึ้น กองกำลังฝ่ายยักษ์จะทำการส่งข้อมูลไปยังระบบโครงข่ายประสาทเทียมเพื่อทำนายผลแพ้ชนะ หากผลที่ได้จากการทำนายออกมาว่าฝ่ายยักษ์แพ้ ฝ่ายยักษ์จะทำการปรับปรุงกองกำลังโดยการโยกย้ายหน่วยรบตามกฎการปรับปรุงกองกำลัง จากนั้นจึงค่อยส่งกองกำลังเข้าต่อสู้กับศัตรู

3.1.9 กฎการปรับปรุงกองกำลัง (Army Reformation Rules)

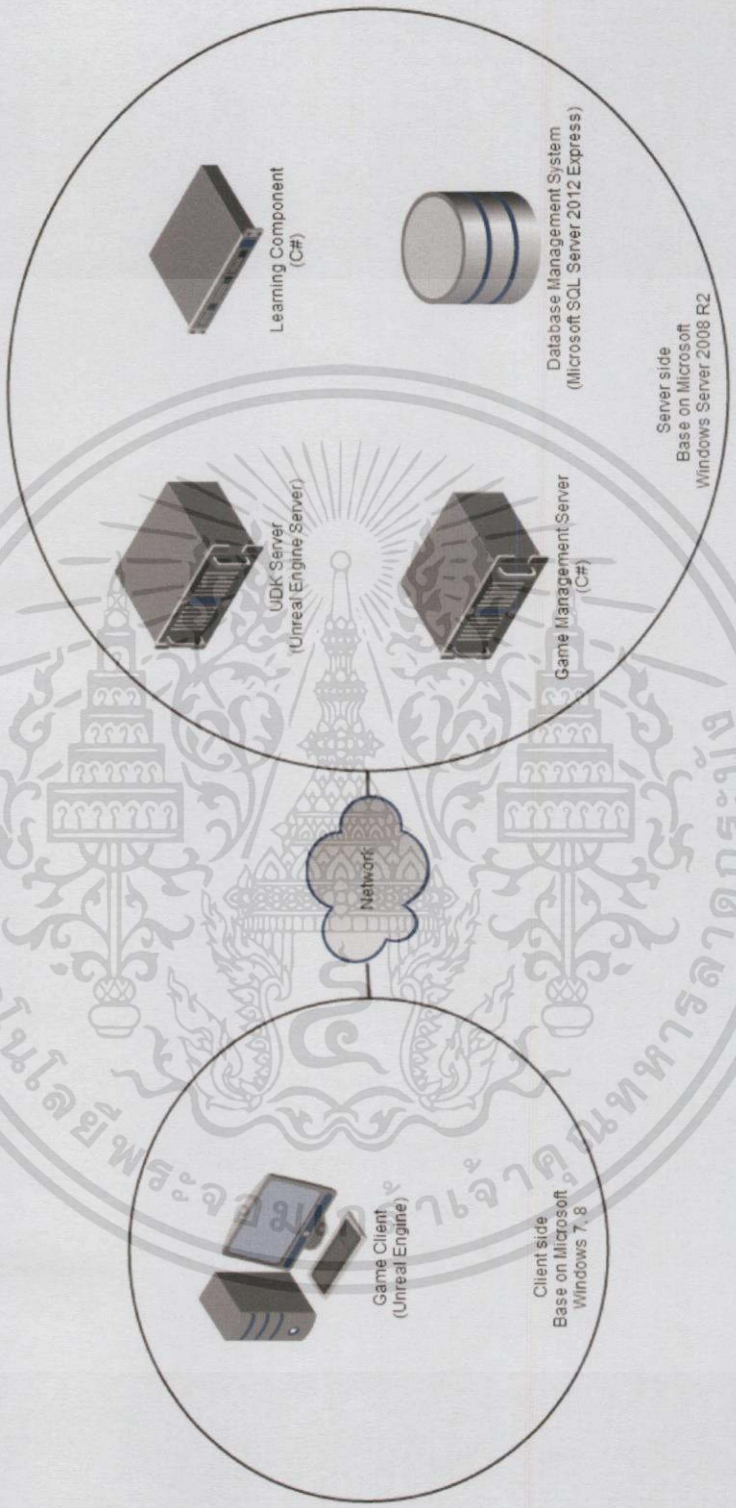
เมื่อมีการเรียกใช้กฎการปรับปรุงกองกำลัง นั่นแสดงว่ากองกำลังยักษ์ดังกล่าวถูกทำนายว่าจะแพ้การต่อสู้ ดังนั้นจึงมีความเป็นไปได้ว่าจำนวนทหารที่มีเยอะกว่าอีกฝั่งอาจจะทำให้เสียเปรียบ (เช่น มีพลดาบเยอะกว่าอีกฝ่ายหลายตัว แต่โดนพลธนูของอีกฝ่ายซึ่งมีมากกว่า ยิงจนแพ้การต่อสู้) ส่วนกองทหารที่มีน้อยกว่าอีกฝั่งอาจจะทำให้ไม่ได้เปรียบเท่าที่ควร (เช่น อีกฝั่งมีพลดาบเยอะ แต่มีพลธนูน้อยเกินไปจนพลดาบเข้าถึงระยะใกล้เสียก่อน) ดังนั้นกฎการปรับปรุงกองกำลังจะตัดสินใจปรับปรุงโดยคิดจากส่วนต่างและจำนวนของทหารแต่ละชนิดของทั้งสองฝั่ง โดยส่วนต่างดังกล่าวคำนวณโดยการนำจำนวนทหารฝั่งยักษ์ลบด้วยจำนวนทหารฝั่งคนให้ครบทุกชนิด โดยแบ่งวิธีการคิดเป็นกรณีย่อยๆดังนี้

1) กรณีผลต่างมากสุด 1 ค่า แสดงว่ามีทหารชนิดหนึ่งที่มีจำนวนมากกว่าอีกฝั่ง ให้ไปดูสองชนิดที่เหลือประกอบ ถ้าสองชนิดที่เหลือต่างกัน ให้ย้ายทหารหนึ่งคนจากกองที่มีผลต่างมากที่สุดไปยังกองที่ผลต่างน้อยที่สุด ถ้าผลต่างยังคงเท่ากันอยู่ ให้ย้ายไปกองที่จำนวนรวมน้อยกว่าแทน แต่ถ้าจำนวนรวมยังคงเท่ากันก็จะใช้การสุ่มแทน
2) กรณีผลต่างมากสุด 2 ค่า แสดงว่ามีทหารชนิดหนึ่งที่จำนวนน้อยกว่าอีกฝั่ง ถ้าสองกองดังกล่าวมีจำนวนรวมต่างกัน ให้สมาชิกกองที่มากกว่าย้ายมา แต่ถ้าจำนวนรวมยังคงเท่ากันก็จะใช้การสุ่มแทน
3) กรณีผลต่างเท่ากันหมดเลย แสดงว่าผลปัจจุบันไม่เหมาะสมแต่ข้อมูลก็ไม่พอที่จะสามารถคาดเดาได้ว่าควรต้องทำอย่างไร ดังนั้นจึงทำการสุ่มเลือกการกระทำว่าจะย้ายกองไหนไปยังกองไหน

3.2. โครงสร้างภาพรวมของระบบ

การทำงานของระบบประกอบไปด้วยโปรแกรม 5 ส่วนหลักคือ

1) โปรแกรมเกมออนไลน์ฝั่งผู้เล่น (Game Client) เป็นตัวเกมที่ผู้เล่นใช้ในการเล่นเกม พัฒนาโดยใช้ Unreal Development Kit
2) โปรแกรมให้บริการของยูดีเค (UDK Server) เป็นโปรแกรมที่เป็นตัวกลางในการควบคุมแลกเปลี่ยนเพื่อเข้าจังหวะ (Synchronize) ตำแหน่งการเดิน การเคลื่อนที่ การหมุน การหันหน้า การกระโดด หรือการกระทำอื่นๆ ของผู้เล่นในเกมทั้งหมด
3) โปรแกรมจัดการเกมออนไลน์ (Game Management Server) เป็นโปรแกรมที่มีหน้าที่รับ-ส่งข้อมูลที่ต้องการเชื่อมต่อมายังฐานข้อมูลส่วนกลาง รวมถึงจัดการเกมเช่น การเคลื่อนย้ายตำแหน่งของฝ่ายยักษ์ การควบคุมการเกิดของกองกำลัง เป็นต้น พัฒนาโดยใช้ภาษา C# (.NET Framework 4.0) และใช้โปรแกรม Visual Studio 2013
4) ระบบฐานข้อมูล (Database Management System) เป็นระบบจัดการข้อมูล โดยใช้ระบบฐานข้อมูลเชิงสัมพันธ์ (Relational Database System) พัฒนาและออกแบบโดยใช้โปรแกรม Microsoft SQL Server 2012 Express
5) โปรแกรมในการเรียนรู้ (Learning Component) เป็นส่วนประกอบเพื่อใช้ในการจดจำและเรียนรู้พฤติกรรมของผู้เล่น รวมถึงใช้ในการทำนายผลเพื่อใช้ในการตัดสินใจที่มีผลต่อการเปลี่ยนแปลงสภาพแวดล้อมในเกม พัฒนาโดยใช้ภาษา C# (.NET Framework 4.0) และใช้โปรแกรม Visual Studio 2013



รูป 3.1 การออกแบบโครงสร้างของระบบเกมในภาพรวม

3.3. โปรแกรมเกมออนไลน์ฝั่งผู้เล่น (Game Client)

ตัวเกมฝั่งผู้เล่นมีขั้นตอนการทำงานดังนี้

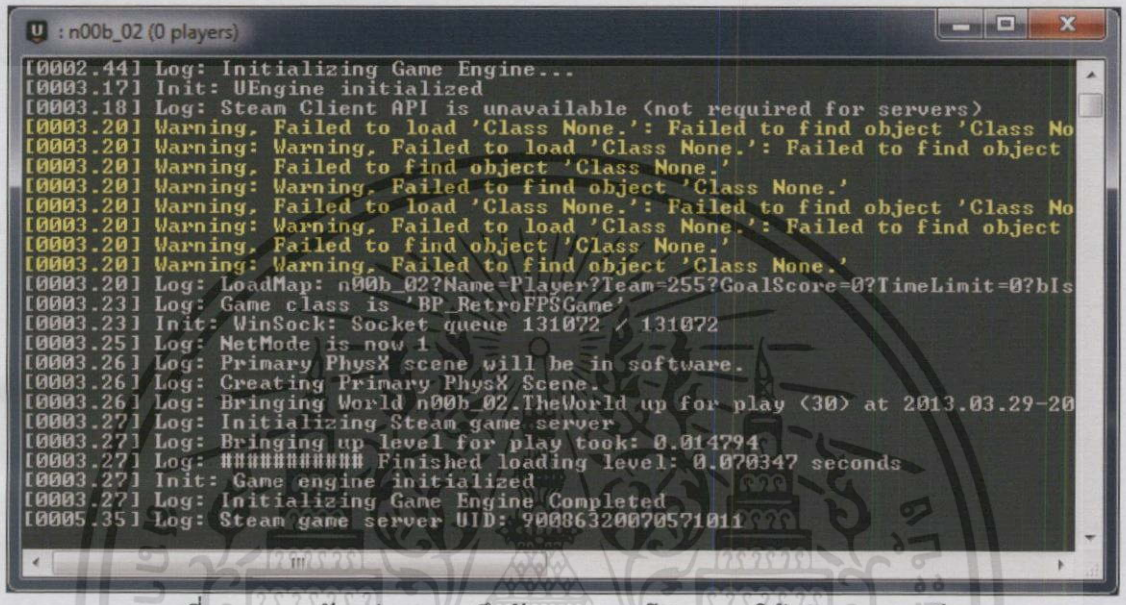
1. ผู้เล่นจะเข้าสู่เกมโดยผ่าน Game Launcher ซึ่งมีหน้าที่ให้ผู้เล่นกรอกชื่อผู้เล่นและรหัสผ่านเพื่อเข้าสู่ระบบ และปรับเปลี่ยนกองกำลังทหาร (หากต้องการ) จากนั้นก็ทำการเข้าสู่เกม
2. ผู้เล่นจะเข้าสู่เกมในแผนที่แบบเปิด (Open world) ซึ่งสามารถเดินได้อย่างอิสระ และจะสามารถเห็นผู้เล่นคนอื่นในแผนที่นี้ได้ และผู้เล่นจะเลือกเข้าปะทะกับกองกำลังฝ่ายยักษ์ (ซึ่งเป็นคอมพิวเตอร์) โดยการเข้าชน
3. เมื่อผู้เล่นเข้าปะทะจะตัดฉากเฉพาะผู้เล่นนั้นไปเริ่มการต่อสู้กับกองกำลังฝ่ายยักษ์โดยใช้กติกา Team death-match และเมื่อจบการต่อสู้ ตัวเกมก็จะส่งผลการต่อสู้ไปยังเซิร์ฟเวอร์เพื่อประมวลผล

รูปที่ 3.2 แผนภาพการทำงานของโปรแกรมฝั่งผู้เล่น

3.4. โปรแกรมให้บริการของยูดีเค (UDK Server)

เป็นโปรแกรมที่เป็นตัวกลางในการควบคุมแลกเปลี่ยนเพื่อเข้าจังหวะ (Synchronize) ตำแหน่งการเดิน การเคลื่อนที่ การหมุน การหันหน้า การกระโดด ของทั้งตัวละครที่เป็นผู้เล่น และตัวละครที่เป็นฝั่งยักษ์ รวมถึงการกระทำอื่นๆที่ส่งผลใดๆก็ตามต่อแผนที่อีกด้วย

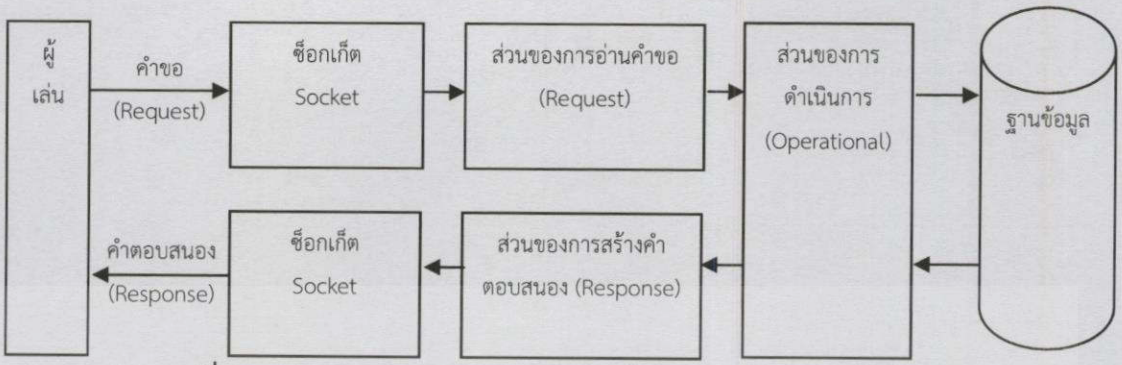
สำหรับโปรแกรมนี้ เป็นโปรแกรมที่ติดมาพร้อมกับตัว UDK ซึ่งสามารถปรับการทำงานเป็นโหมดเซิร์ฟเวอร์ได้ และได้นำโปรแกรมที่ติดมานี้ มาเขียนสคริปต์ (Script) เพื่อปรับแต่งให้สามารถใช้งานกับลักษณะของเกมที่ใช้ในการทดลองนี้อย่างเหมาะสม



รูปที่ 3.3 ภาพตัวอย่างแสดงถึงลักษณะของโปรแกรมให้บริการของยูดีเค

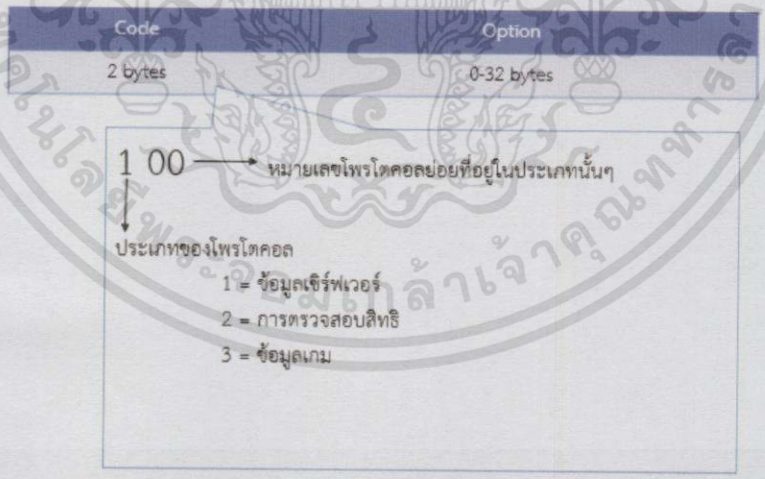
3.5. โปรแกรมจัดการเกมออนไลน์ (Game Server)

เป็นตัวโปรแกรมที่อยู่บนฝั่งผู้ให้บริการ (Server) เพื่อจัดการการร้องขอหรือรับส่งข้อมูลที่ต้องติดต่อกับฐานข้อมูล โดยมีหลักการคือ โปรแกรมจะทำการเปิดซ็อกเก็ต (Socket) เพื่อรอรับการเชื่อมต่อ เมื่อมีคำขอ (Request) เข้ามา โปรแกรมก็จะดำเนินการรับคำขอนั้นมาประมวลผล โดยส่งผ่านไปยังส่วนของการอ่านคำขอ ซึ่งจะทำการอ่านข้อมูลตามรูปแบบโพรโทคอล (Protocol) ที่ได้ระบุไว้ จากนั้นก็จะส่งต่อไปยัง ส่วนดำเนินการ (Operational) ว่าจะทำอย่างไรเช่น ติดต่อฐานข้อมูลเพื่อดึงค่าบางอย่าง เป็นต้น และหากมีคำตอบสนองกลับ (Response) ก็จะส่งต่อไปยังส่วนของการสร้างคำตอบสนอง และส่งให้ซ็อกเก็ต ส่งข้อมูลกลับไปยังผู้เล่นนั่นเอง ดังแสดงให้เห็นดังรูป



รูปที่ 3.4 แผนภาพแสดงการทำงานของโปรแกรมจัดการเกมออนไลน์

ซึ่งการติดต่อและแลกเปลี่ยนข้อมูลระหว่างเครื่องผู้ใช้บริการกับผู้ให้บริการนั้น จะมีรูปแบบของการติดต่อ (Protocol) ในระดับชั้นแอพลิเคชั่น (Application layer) โดยกำหนดเป็นโครงสร้างของข้อมูลที่จะส่งไปประกอบด้วย 2 ส่วนเป็นดังรูปที่ 3.4 ส่วนแรกคือส่วนของโค้ด (Code) ซึ่งเป็นตัวเลขที่ใช้แทนคำสั่งที่ต้องการ ค่าหลักร้อยเป็นค่าที่แสดงถึงประเภท (Type) ของคำสั่ง ค่าหลักหน่วยและหลักสิบประกอบกันเป็นลำดับของคำสั่งในประเภทนั้นๆ ส่วนแรกนี้มีขนาด 2 ไบต์ (Byte) ส่วนที่สองคือออพชั่น (Option) ซึ่งเป็นส่วนที่ใส่ข้อมูลที่ต้องการเพื่อแนบไปกับคำสั่งนั้นๆมีขนาดตั้งแต่ 0 – 32 ไบต์



รูปที่ 3.5 โครงสร้างของโพรโทคอลที่ใช้ในการติดต่อสื่อสาร

รายละเอียดของแต่ละคำสั่งและออพชั่นของคำสั่ง สามารถแสดงรายละเอียดได้ดังตารางที่ 3.1 สำหรับวิธีการอ่านตารางให้สังเกตดังนี้ สดมภ์หรือคอลัมน์ (Column) แรกหมายถึงหมายเลขของคำสั่งนั้นๆ คอลัมน์ที่สองมาหมายถึงผู้ส่งคำสั่งและออพชั่นนั้นออกไป หากเป็นตัวเกมฝั่งผู้เล่น (Client) ก็จะทำการร้องขอ หากเป็นตัวจัดการเกม (Server) ก็จะเป็นการตอบสนอง คอลัมน์ที่สาม

หมายถึงรายละเอียดของคำสั่งนั้นๆ และหากคำสั่งนั้นต้องการใส่ออพชั่น ก็จะอธิบายความหมายของออพชั่นในคอมลัมน์นี้ คอลัมน์ที่สี่หมายถึงรหัสหรือตัวเลขแทนคำสั่ง คอลัมน์สุดท้ายหมายถึงข้อมูลออพชั่นที่ต้องการส่งไปพร้อมกับคำสั่ง ซึ่งจะบอกภายในคอลัมน์นี้ว่าจะส่งข้อมูลในแต่ละไบต์เป็นข้อมูลอะไรบ้าง

ยกตัวอย่างการอ่านโพรโตตอลหมายเลข 9 หมายถึงตัวเกมฝั่งผู้เล่นเป็นคนส่งคำร้องขอไปยังตัวจัดการเกม เพื่อขอตำแหน่งปัจจุบันของผู้เล่น โดยการส่งคำขอด้วยโค้ด 301 และส่งออพชั่นไบต์ที่ 1-4 เป็น หมายเลขประจำตัวผู้เล่น (Player ID) ไปด้วย

ตารางที่ 3.1 ตารางแสดงรูปแบบและข้อกำหนดของโพรโทคอล

หมายเลข	ผู้ส่ง (Sender)	คำสั่ง (Command)	โค้ด (Code)	ออพชั่น (Options)
กลุ่มที่ 1: ข้อมูลเซิร์ฟเวอร์				
1	Client	ขอสถานะของเซิร์ฟเวอร์	100	
2	Server	ส่งสถานะของเซิร์ฟเวอร์ว่าเปิด	100	Byte[1] = 1
	Server	ส่งสถานะของเซิร์ฟเวอร์ว่าปิด	100	Byte[1] = 0
3	Client	ขอเวอร์ชั่นของเกม	101	
4	Server	ส่งเวอร์ชั่นของเกม - v.v = vesion	101	Byte[1-3] = v.v
กลุ่มที่ 2: การตรวจสอบสิทธิ				
5	Client	ตรวจสอบผู้ใช้งาน - x = ชื่อผู้เล่น - y = รหัสผ่าน	200	Byte[1-8] = x Byte[9] = # Byte[10-17] = y
6	Server	ส่งสถานะการเข้าสู่ระบบ (สำเร็จ) - u = หมายเลขผู้เล่น	200	Byte[1] = 1 Byte[2-5] = u
	Server	ส่งสถานะการเข้าสู่ระบบ (ล้มเหลว)	200	Byte[1] = 0
กลุ่มที่ 3: ข้อมูลเกม				
7	Client	ขอตำแหน่งของขุนศึก - w = หมายเลขขุนศึก	300	Byte[1-4] = w

ตารางที่ 3.1 (ต่อ) ตารางแสดงรูปแบบและข้อกำหนดของโพรโทคอล

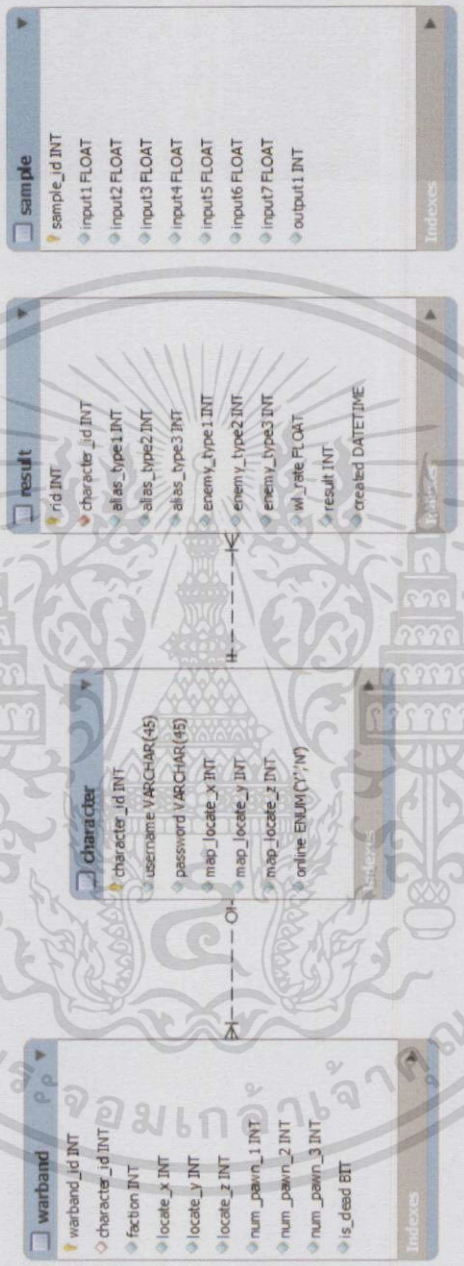
หมายเลข	ผู้ส่ง (Sender)	คำสั่ง (Command)	โค้ด (Code)	ออพชั่น (Options)
8	Server	ส่งตำแหน่งของขุนศึก - x = ตำแหน่งขุนศึกในแกน X - y = ตำแหน่งขุนศึกในแกน Y - z = ตำแหน่งขุนศึกในแกน Z	300	Byte[1-4] = x Byte[5-8] = y Byte[9-12] = z
9	Client	ขอตำแหน่งของผู้เล่น - u = หมายเลขผู้เล่น	301	Byte[1-4] = u
10	Server	ส่งตำแหน่งของผู้เล่น - x = ตำแหน่งผู้เล่นในแกน X - y = ตำแหน่งผู้เล่นในแกน Y - z = ตำแหน่งผู้เล่นในแกน Z	301	Byte[1-4] = x Byte[5-8] = y Byte[9-12] = z
11	Client	ขอจำนวนทหารธนูของผู้เล่น - u = หมายเลขผู้เล่น	302	Byte[1-4] = u
12	Server	ส่งจำนวนทหารธนูของผู้เล่น - n = จำนวนทหารธนู	302	Byte[1-4] = n
13	Client	ขอจำนวนทหารดาบของผู้เล่น - u = หมายเลขผู้เล่น	303	Byte[1-4] = u
14	Server	ส่งจำนวนทหารดาบของผู้เล่น - n = จำนวนนักดาบ	303	Byte[1-4] = n
15	Client	ขอจำนวนทหารหอกของผู้เล่น - u = หมายเลขผู้เล่น	304	Byte[1-4] = u
14	Server	ส่งจำนวนทหารหอกของผู้เล่น - n = จำนวนนักหอก	303	Byte[1-4] = n
16	Client	ขอจำนวนทหารธนูของฝ่ายยักษ์ - u = หมายเลขฝ่ายยักษ์	302	Byte[1-4] = u
17	Server	ส่งจำนวนทหารธนูของฝ่ายยักษ์ - n = จำนวนทหารธนู	302	Byte[1-4] = n

ตารางที่ 3.1 (ต่อ) ตารางแสดงรูปแบบและข้อกำหนดของโพรโทคอล

หมายเลข	ผู้ส่ง (Sender)	คำสั่ง (Command)	โค้ด (Code)	ออพชั่น (Options)
18	Client	ขอจำนวนทหารดาบของฝ่ายยักษ์ - u = หมายเลขฝ่ายยักษ์	303	Byte[1-4] = u
19	Server	ส่งจำนวนทหารดาบของฝ่ายยักษ์ - n = จำนวนนักดาบ	303	Byte[1-4] = n
20	Client	ขอจำนวนทหารหอกของฝ่ายยักษ์ - u = หมายเลขฝ่ายยักษ์	304	Byte[1-4] = u
14	Server	ส่งจำนวนทหารหอกของฝ่ายยักษ์ - n = จำนวนนักหอก	303	Byte[1-4] = n
20	Server	ส่งขุนศึกฝ่ายยักษ์ทั้งหมดที่อยู่ในแผนที่ - w = หมายเลขขุนศึกฝ่ายยักษ์ - หมายเหตุ: หากมีมากกว่า 1 ขุนศึก จะใช้เครื่องหมายจุลภาค (,) คั่น	306	Byte[1-4] = w
21	Client	แจ้งเซิร์ฟเวอร์ในเตรียมตัวเริ่มการต่อสู้	307	
22	Server	แจ้งฝั่งผู้เล่นว่าพร้อาให้เริ่มต่อสู้แล้ว	307	Byte[1] = 1

นอกจากเป็นโปรแกรมที่ใช้ในการจัดการการเชื่อมต่อที่ต้องการเข้าถึงข้อมูลในฐานข้อมูลแล้ว ยังเป็นโปแกรมที่มีหน้าที่ในการบริหารจัดการเกมในภาพรวมเช่น การเคลื่อนย้ายตำแหน่งของฝ่ายยักษ์ การควบคุมการเกิดของกองกำลัง การประสานงานกับส่วนการเรียนรู้เพื่อตัดสินใจปรับจำนวนทหารก่อนการเริ่มการต่อสู้ การสังเกตการณ์ระบบ (Monitoring) อีกด้วย

3.6. ระบบจัดการฐานข้อมูล (Database Management System)



รูปที่ 3.6 แบบจำลองความสัมพันธ์เอนทิติ (Entity Relationship Diagram)

ตารางที่ 3.2 ตารางแสดงโครงสร้างของตาราง Character

ชื่อฟิลด์ (Field)	ประเภท	คำอธิบาย	หมายเหตุ
character_id	int	หมายเลขผู้เล่น	Primary Key, auto_increasement
username	varchar(45)	ชื่อผู้ใช้สำหรับเข้าสู่ระบบ	
password	varchar(45)	รหัสผ่านสำหรับเข้าสู่ระบบ	
map_loc_x	int	ตำแหน่งปัจจุบันของตัวละครในแกน X	
map_loc_y	int	ตำแหน่งปัจจุบันของตัวละครในแกน Y	
map_loc_z	int	ตำแหน่งปัจจุบันของตัวละครในแกน Z	
online	ENUM(Y,N)	ตัวละครนี้ออนไลน์อยู่ในเกมหรือไม่	

ตารางที่ 3.3 ตารางแสดงโครงสร้างของตาราง Warband

ชื่อฟิลด์ (Field)	ประเภท	คำอธิบาย	หมายเหตุ
warband_id	int	หมายเลขของกองพล	Primary Key, auto_increasement
character_id	int	หมายเลขผู้เล่นที่เป็นเจ้าของ Warband นี้ (หากเป็นคอมพิวเตอร์จะตั้งเป็นค่าว่างเปล่า (null))	Foreign key กับ character_id กับตาราง character
faction	int	หมายเลขของฝ่ายที่สังกัดอยู่	
locate_x	int	ตำแหน่งปัจจุบันของ Warband ในแกน X	
locate_y	int	ตำแหน่งปัจจุบันของ Warband ในแกน Y	
locate_z	int	ตำแหน่งปัจจุบันของ Warband ในแกน Z	

ตารางที่ 3.3 (ต่อ) ตารางแสดงโครงสร้างของตาราง Warband

ชื่อฟิลด์ (Field)	ประเภท	คำอธิบาย	หมายเหตุ
num_pawn_1	int	จำนวนทหารธนู	
num_pawn_2	int	จำนวนทหารดาบ	
num_pawn_3	Int	จำนวนทหารหอก	
is_dead	bit	การบ่งบอกถึงการถูกฆ่าตายหรือยัง (1 หมายถึงกองกำลังนี้ตายแล้ว, 0 หมายถึงกองกำลังนี้ยังไม่ตาย)	

ตารางที่ 3.4 ตารางแสดงโครงสร้างของตาราง Result

ชื่อฟิลด์ (Field)	ประเภท	คำอธิบาย	หมายเหตุ
rid	int	หมายเลขของผลการต่อสู้	Primary Key, auto_increasement
character_id	int	หมายเลขผู้เล่นที่เป็นเจ้าของผลการต่อสู้นี้	Foreign key กับ character_id กับตาราง character
alias_type1	int	จำนวนทหารธนู ของฝั่งผู้เล่น	
alias_type2	int	จำนวนทหารดาบ ของฝั่งผู้เล่น	
alias_type3	int	จำนวนทหารหอก ของฝั่งผู้เล่น	
enemy_type1	int	จำนวนทหารธนู ของฝั่งศัตรู	
enemy_type2	int	จำนวนทหารดาบ ของฝั่งศัตรู	
enemy_type3	int	จำนวนทหารหอก ของฝั่งศัตรู	
wl_rate	float	ค่าเฉลี่ยสถิติการแพ้ชนะของผู้เล่น ก่อนหน้านี้	
result	int	ผลการต่อสู้ (1 หมายถึงผู้เล่นต่อสู้แล้วชนะ, 0 หมายถึงผู้เล่นต่อสู้แล้วแพ้)	
created	datetime	วันและเวลาที่ผลแพ้ชนะนั้นถูกบันทึกลงไป	

ตารางที่ 3.5 ตารางแสดงโครงสร้างของตาราง Sample

ชื่อฟิลด์ (Field)	ประเภท	คำอธิบาย	หมายเหตุ
sample_id	int	หมายเลขของกองพล	Primary Key, auto_increasement
input1	float	ค่าจำนวนทหารธนูฝั่งผู้เล่นที่ทำการ Re-scale แล้ว	
input2	float	ค่าจำนวนทหารดาบฝั่งผู้เล่นที่ทำการ Re-scale แล้ว	
input3	float	ค่าจำนวนทหารหอกฝั่งผู้เล่นที่ทำการ Re-scale แล้ว	
input4	float	ค่าจำนวนทหารธนูฝั่งศัตรูที่ทำการ Re-scale แล้วจา	
input5	float	ค่าจำนวนทหารดาบฝั่งศัตรูที่ทำการ Re-scale แล้ว	
input6	float	ค่าจำนวนทหารหอกฝั่งศัตรูที่ทำการ Re-scale แล้ว	
input7	float	ค่าเฉลี่ยสถิติการแพ้ชนะของผู้เล่น	
output	int	ผลการต่อสู้ (1 หมายถึงผู้เล่นชนะ, 0 หมายถึงผู้เล่นแพ้)	

3.7. โปรแกรมในการเรียนรู้ (Learning Component)

3.7.1 ขั้นตอนการพัฒนา

1) เลือกข้อมูลที่เป็นส่วนของการนำเข้า
2) เลือกวิธีการเข้ารหัส (Encode) เพื่อแปลงข้อมูลที่ได้จากการเล่น ให้มาอยู่ในรูปแบบที่โครงข่ายประสาทเทียมสามารถเรียนรู้ได้
3) เลือกฟังก์ชั่นกระตุ้น (Activation function) ที่ใช้ในโครงข่ายประสาทเทียม
4) เลือกและหาวิธีการวิเคราะห์ตัดสินค่าต่างๆ ที่ได้จากข้อมูลส่งออก
5) พัฒนาโครงสร้างของโครงข่ายประสาทเทียมตามหลักการที่และรูปแบบที่ได้ออกแบบไว้

6) ทำการทดลองเพื่อวัดประสิทธิภาพของโครงข่ายประสาทเทียมที่ใช้ และทำการปรับปรุง เพื่อให้ผลเป็นที่ยอมรับได้

3.7.2 เป้าประสงค์ของโปรแกรมในการเรียนรู้

สิ่งที่ผู้พัฒนาต้องการจากโปรแกรมในการเรียนรู้คือการทำนายผลการแพ้ชนะโดยจดจำและวิเคราะห์จากข้อมูลจากพฤติกรรมผู้เล่น ซึ่งพฤติกรรมที่กล่าวถึงคือ พฤติกรรมการจัดการกองกำลังซึ่งประกอบไปด้วยทหารสามชนิดอย่างที่ได้กล่าวไปข้างต้น โดยข้อมูลในเกมมีจำนวนมากและสามารถเพิ่มขยายได้ตลอดเวลาเนื่องจากเป็นข้อเด่นของเกมประเภทนี้ และเมื่อนำค่าดังกล่าวมาทำการปรับค่าแล้ว จะนำข้อมูลที่ปรับค่าแล้วนำไปป้อนเข้าระบบโครงข่ายประสาทเทียมต่อไป

3.7.3 การออกแบบโครงข่ายประสาทเทียม

จากที่ต้องการให้กองกำลังทำการทำนายผลการต่อสู้ว่าแพ้หรือชนะจากรูปแบบการจัดกองกำลังที่ต่างกันและมีความซับซ้อนในระดับหนึ่งนั้น ผู้พัฒนาจึงออกแบบโดยใช้โครงข่ายประสาทเทียมแบบหลายชั้น โดยมีชั้นซ่อนหนึ่งชั้น ที่มีรูปแบบการเรียนรู้แบบย้อนกลับมาใช้ และรูปแบบการตัดสินใจแบบวินเนอร์เทคออล (Winner take all)

3.7.3.1 ส่วนของการนำเข้า

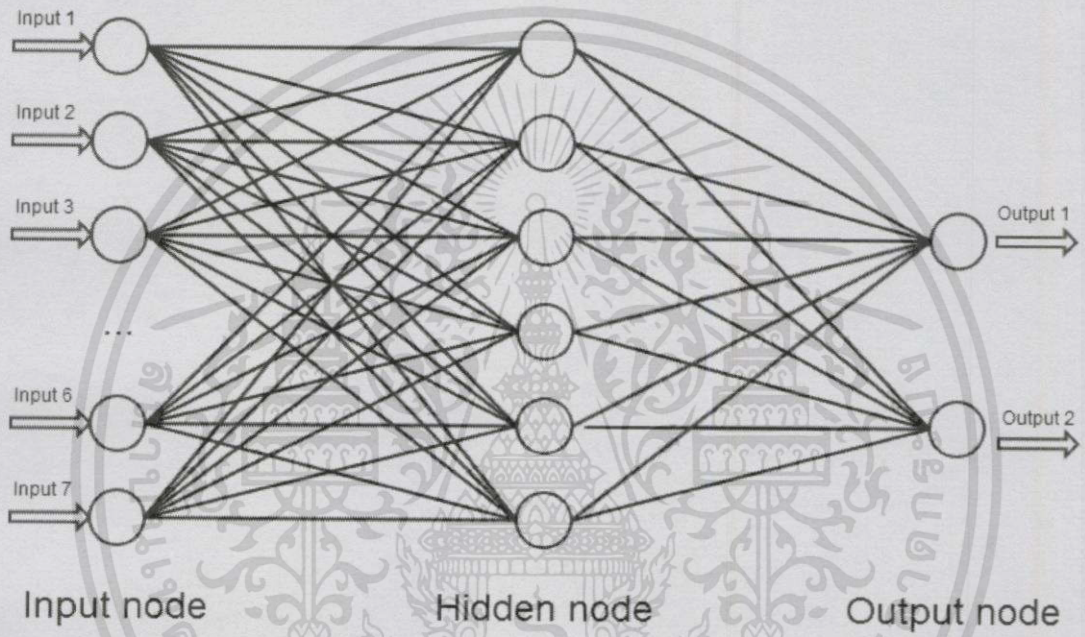
1) จำนวนทหารธนูของฝั่งผู้เล่นที่มีการปรับค่าแล้ว มีค่าอยู่ในช่วง 0 ถึง 1
2) จำนวนทหารดาบของฝั่งผู้เล่นที่มีการปรับค่าแล้ว มีค่าอยู่ในช่วง 0 ถึง 1
3) จำนวนทหารหอกของฝั่งผู้เล่นที่มีการปรับค่าแล้ว มีค่าอยู่ในช่วง 0 ถึง 1
4) จำนวนทหารธนูของฝั่งยักษ์ที่มีการปรับค่าแล้ว มีค่าอยู่ในช่วง 0 ถึง 1
5) จำนวนทหารดาบของฝั่งยักษ์ที่มีการปรับค่าแล้ว มีค่าอยู่ในช่วง 0 ถึง 1
6) จำนวนทหารหอกของฝั่งยักษ์ที่มีการปรับค่าแล้ว มีค่าอยู่ในช่วง 0 ถึง 1
7) ค่าเฉลี่ยสถิติผลแพ้ชนะของผู้เล่น มีค่าอยู่ในช่วง 0 ถึง 1

3.7.3.2. ส่วนของการส่งออก

ค่าที่ได้จากการคำนวณจะบ่งบอกถึง ผลการทำนายแพ้ชนะจากโครงข่ายประสาทเทียม ซึ่งแปลผลดังตารางที่ 3.6

ตารางที่ 3.6 ตารางการแปลผลข้อมูลส่งออกจากโครงข่ายประสาทเทียม

ค่าของข้อมูลส่งออกที่		ความหมาย
1	2	
1	0	ทำนายว่าผู้เล่นจะเป็นผู้ชนะ (ฝ่ายยักษ์แพ้)
0	1	ทำนายว่าฝ่ายยักษ์จะเป็นผู้ชนะ (ฝ่ายผู้เล่นแพ้)



รูปที่ 3.7 ลักษณะของโครงข่ายประสาทเทียมที่นำมาใช้ในการทำนายผลการต่อสู้

3.8 ขั้นตอนการออกแบบองค์ประกอบศิลป์

1) ออกแบบกรอบแนวคิด
2) ร่างแบบมโนภาพ (Concept Art)
3) นำแบบมโนภาพที่ได้ไปใช้อ้างอิงในการทำโครงร่างโมเดลสามมิติ (3d Model)
4) ทำลายพื้นผิว (Texture) ให้กับโครงร่างโมเดลที่สร้างขึ้น
5) นำโมเดลที่เสร็จแล้วมาใส่การเคลื่อนไหว (Animation) ใน UDK Editor
6) ใช้ UDK Editor สร้างแผนที่จากแบบมโนภาพที่ร่างไว้

3.9. แผนการดำเนินการ

ตารางที่ 3.7 ตารางแสดงแผนการดำเนินงานของโครงการ

แผนการดำเนินการ	เดือนที่ 1	2	3	4	5	6	7	8	9
1. ออกแบบและวางแผน									
- ออกแบบโครงสร้างของเกม									
- ออกแบบกติกาและวิธีการเล่นเกม									
- ออกแบบเนื้อเรื่องเกม									
- ออกแบบฐานข้อมูล									
- ศึกษาและออกแบบโพรโทคอลที่ใช้ในการสื่อสาร									
- ศึกษาและออกแบบโครงข่ายประสาทเทียม									
2. พัฒนาตัวจัดการเกมออนไลน์									
- พัฒนาระบบในการสื่อสาร									
- ติดตั้งและเชื่อมต่อกับฐานข้อมูล									
3. ออกแบบและพัฒนาตัวเกมฝั่งผู้เล่น									
- ศึกษาและทดลองใช้ UDK									
- ออกแบบตัวละคร แผนที่และฉาก									
- พัฒนาโปรแกรมที่ใช้เริ่มเกม									
- พัฒนาตัวเกมฝั่งผู้เล่น									

ตารางที่ 3.7 (ต่อ) ตารางแสดงแผนการดำเนินงานของโครงการ

แผนการดำเนินการ	เดือนที่								
	1	2	3	4	5	6	7	8	9
4. พัฒนาโปรแกรมในส่วนของการเรียนรู้									
- ออกแบบวิธีวิเคราะห์ข้อมูล							■		
- พัฒนาระบบดึงข้อมูลจากระบบเกม							■		
- พัฒนาโครงข่ายประสาทเทียม							■	■	
- นำข้อมูลเข้ามาให้โครงข่ายประสาทเทียมเรียนรู้และปรับค่าต่างๆให้เหมาะสม								■	
5. ทดสอบและทดลองโครงข่ายประสาทเทียมกับตัวเกม และบันทึกผลการทดลอง									■
6. สรุปผลโครงงาน									■

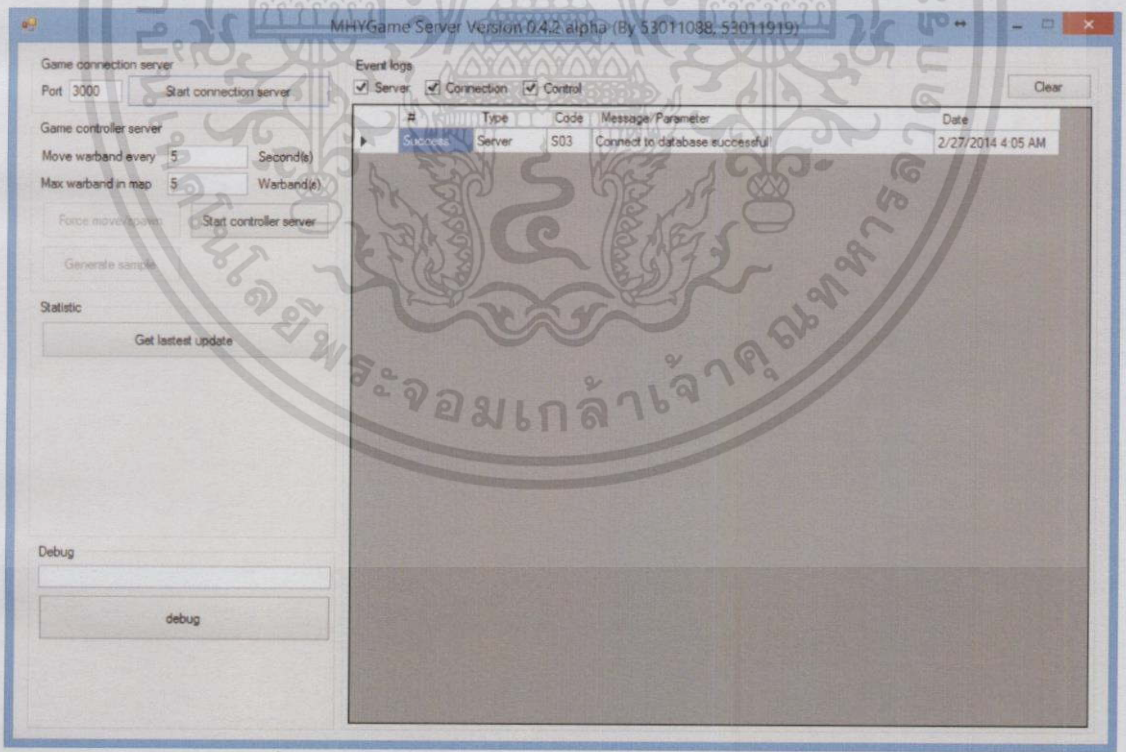
บทที่ 4
การทดลองและผลการทดลอง

4.1. การทดลองสร้างเกมสามมิติแบบผู้เล่นหลายคน

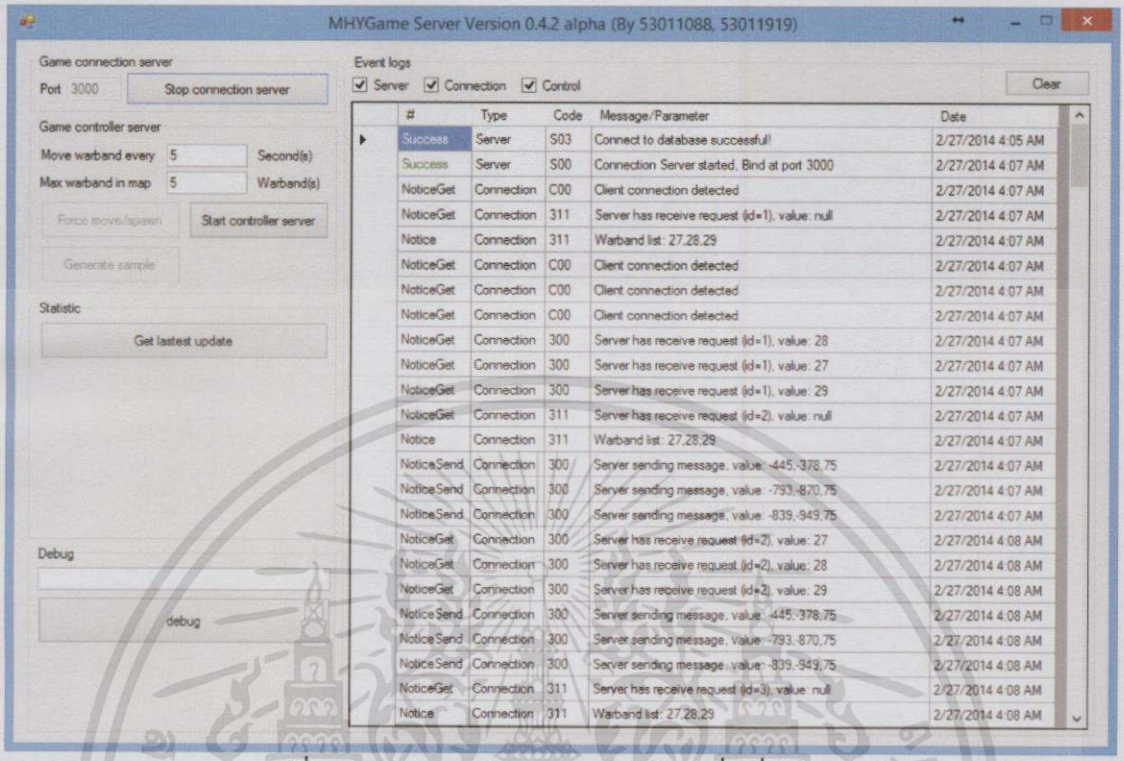
เป็นการทดลองสร้างเกมเพื่อให้ผู้เล่นสามารถเข้ามาเล่นเกมได้ และส่งผลค่าข้อมูลการต่อสู้กลับไปยังส่วนของการเรียนรู้ได้ ซึ่งจะต้องทำให้ผู้เล่นสามารถเล่นได้หลายคน

การทดลองที่ 1 เป็นการทดลองเพื่อทำการติดต่อสื่อสารแลกเปลี่ยนข้อมูลระหว่างโปรแกรมจัดการเกมออนไลน์ และตัวเกมฝั่งผู้เล่นด้วยทีซีพี (TCP) ผ่านระบบเครือข่าย

เริ่มต้นจากการที่ระบบจะต้องเปิดโปรแกรมจัดการเกมออนไลน์เพื่อเป็นศูนย์กลางการจัดการข้อมูลก่อน โดยเปิดโปรแกรมขึ้นมาจะได้ดังรูปที่ 4.1 และทำการกำหนดพอร์ตในการเชื่อมต่อ ซึ่งในการทดลองนี้คือพอร์ต 3000 และทำการกดปุ่ม Start connection server เพื่อทำการเริ่มต้นการให้บริการ ถ้าหากการทำงานทุกอย่างเป็นไปตามปกติ จะแสดงให้เห็นดังรูปที่ 4.2

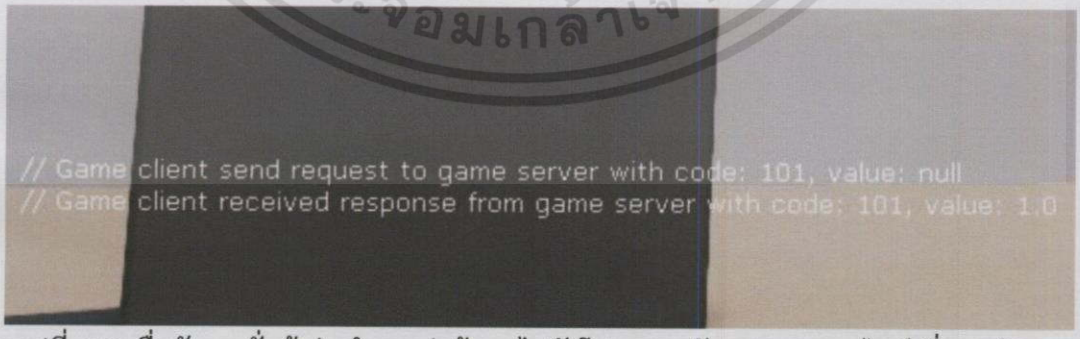


รูปที่ 4.1 โปรแกรมจัดการเกมออนไลน์ก่อนการเริ่มทำงาน



รูปที่ 4.2 โปรแกรมจัดการเกมออนไลน์เมื่อเริ่มทำงาน

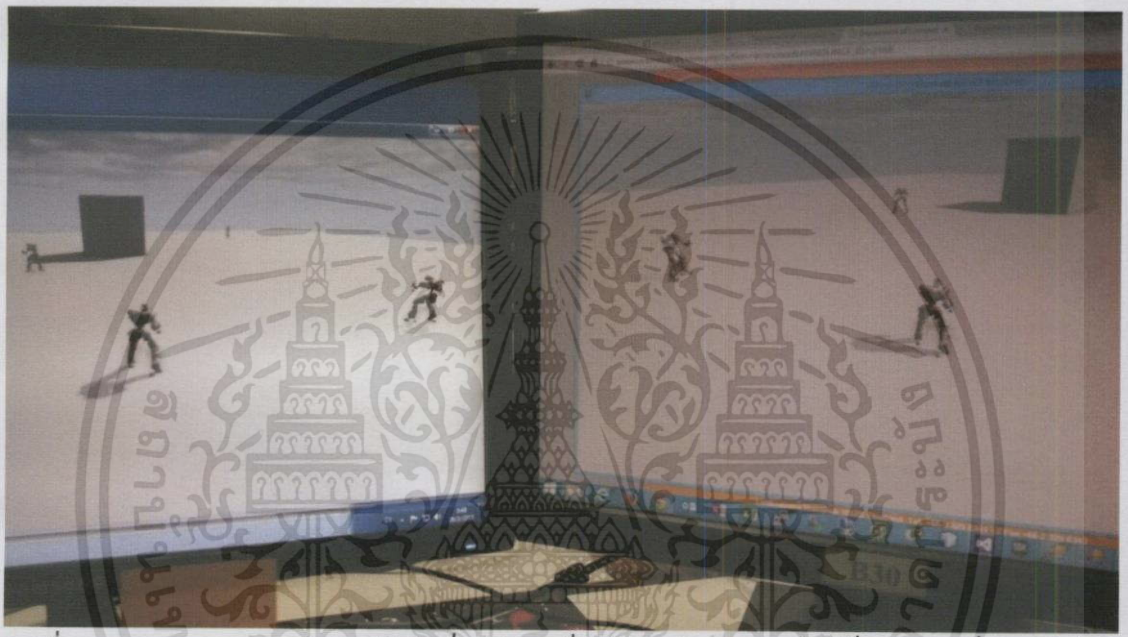
เมื่อโปรแกรมจัดการเกมออนไลน์ทำงานเป็นปกติแล้ว ตัวเกมฝั่งผู้เล่นก็สามารถส่งคำขอ (Request) มายังโปรแกรมจัดการเกมออนไลน์เพื่อติดต่อและแลกเปลี่ยนข้อมูลได้ ในการทดลองนี้ ตัวเกมฝั่งผู้เล่นจะทำการส่งคำขอด้วยโค้ด (Code) 101 ดังรูปที่ 4.3 ซึ่งตามโพรโทคอลที่กำหนดไว้ในบทที่ผ่านมาจะหมายถึงการร้องขอรุ่น (Version) ของโปรแกรมจัดการเกมออนไลน์นั่นเอง และเมื่อโปรแกรมจัดการเกมได้รับก็จะทำการตอบสนองตามที่ร้องขอมา



รูปที่ 4.3 เมื่อตัวเกมฝั่งผู้เล่นทำการส่งข้อมูลไปยังโปรแกรมจัดการเกมออนไลน์เพื่อขอข้อมูลและได้รับการตอบกลับตามที่ได้ออกแบบไว้ในโพรโทคอล

การทดลองที่ 2 เป็นการทดลองสร้างส่วนของการเดินในแผนที่หลักซึ่งผู้เล่นหนึ่งจะต้องเห็นผู้เล่นคนอื่น ไม่ว่าจะเป็นการเดิน การเคลื่อนไหว หรือการกระทำใดๆที่ส่งผลต่อแผนที่โดยรวม

ซึ่งจากการที่ได้ทดลองสร้างเกมและทดลองเล่นดูปรากฏดังรูปที่ 4.4 ผลการทดลองเป็นไปตามที่คาดหวังไว้คือ ผู้เล่นคนหนึ่งสามารถเห็นผู้เล่นคนอื่น เคลื่อนไหว หรือการกระทำใดๆที่ส่งผลต่อแผนที่โดยรวมได้



รูปที่ 4.4 ภาพแสดงการเล่นของผู้เล่นที่อยู่ต่างเครื่องกัน แต่สามารถเห็นผู้อื่นในแผนที่แบบเปิดได้

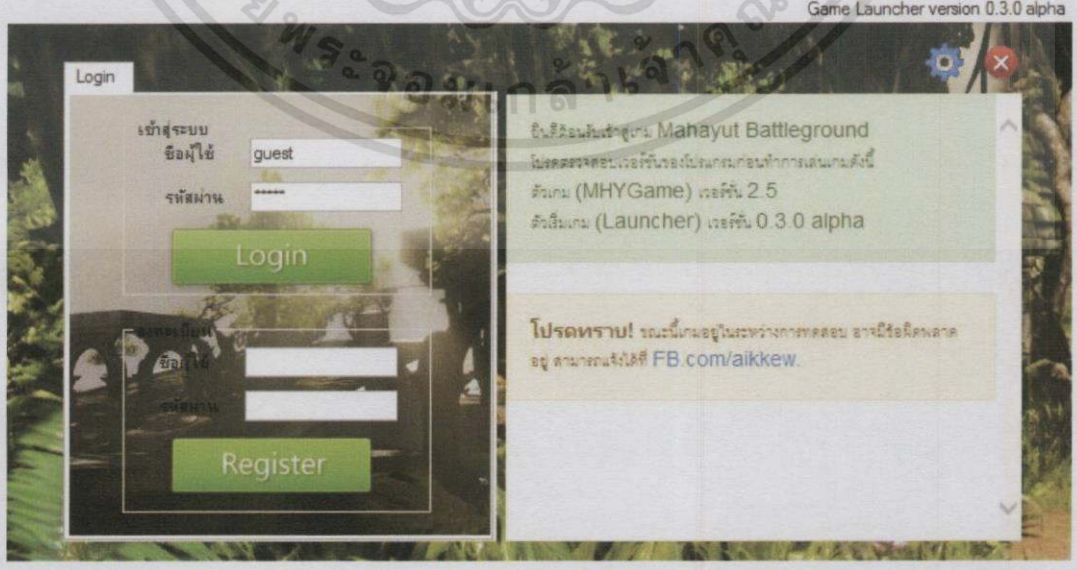
การทดลองที่ 3 เป็นการทดลองสร้างแผนที่สำหรับการต่อสู้ และระบบการต่อสู้แบบแบ่งทีม ซึ่งจะนำไปใช้ในการสร้างการต่อสู้ระหว่างฝ่ายผู้เล่นกับฝ่ายยักษ์ รวมถึงเมื่อสู้จบแล้วสามารถบันทึกลงฐานข้อมูลเพื่อจะได้นำข้อมูลไปใช้ต่อไปได้ด้วย

ซึ่งจากการที่ได้ทดลองสร้างแผนที่สำหรับการต่อสู้และระบบการต่อสู้แบบแบ่งทีม และทดลองเล่นดูปรากฏดังรูปที่ 4.5 ผลการทดลองเป็นไปตามที่คาดหวังไว้คือ ผู้เล่นสามารถเข้ามาทำการต่อสู้กับฝ่ายยักษ์และสามารถส่งผลการต่อสู้บันทึกลงฐานข้อมูลได้

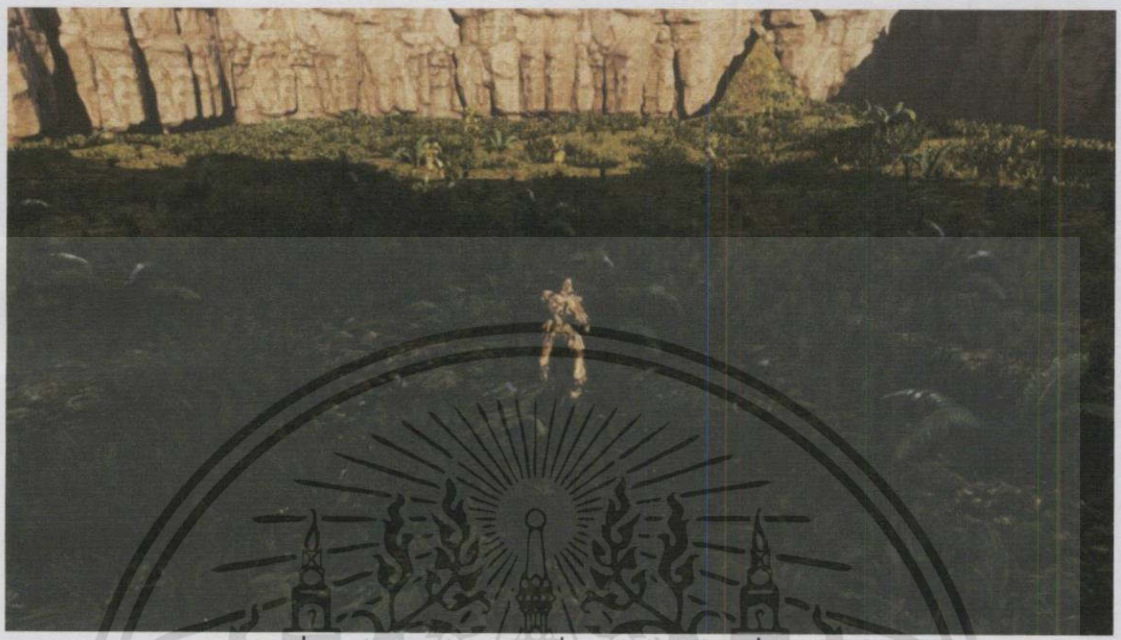
รูปที่ 4.5 ภาพแสดงการเล่นของผู้เล่นกับฝ่ายยักษ์ในระบบการต่อสู้แบบแบ่งทีม

การทดลองที่ 4 เป็นการทดลองให้ผู้เล่นเข้าเกมผ่าน Game Launcher และกรอกข้อมูลเพื่อเข้าสู่ระบบ (หากไม่ได้เป็นสมาชิกก็สามารถสมัครสมาชิกได้) และสามารถปรับเปลี่ยนกองกำลังของตนเองได้ ดังรูปที่ 4.6

ซึ่งจากการที่ได้ทดลองให้ผู้เล่นเข้าเกมผ่าน Game Launcher นั้น พบว่าผู้เล่นสามารถเข้าสู่เกมและสามารถปรับปรุงกองกำลังรวมถึง เข้าเดินในแผนที่แบบเปิดได้อย่างอิสระดังที่ต้องการ ดังรูปที่ 4.7



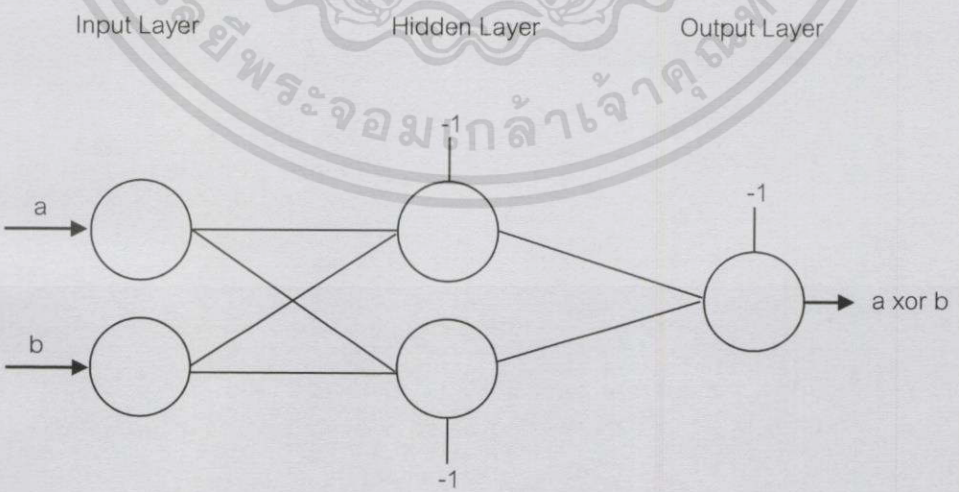
รูปที่ 4.6 ภาพแสดงโปรแกรม Game Launcher ซึ่งเป็นโปรแกรมที่ให้ผู้เล่นใช้เข้าสู่เกม



รูปที่ 4.7 ภาพแสดงผู้เล่นที่กำลังอยู่ในแผนที่แบบเปิด

4.2. การทดลองการแก้ไขปัญหาตรรกศาสตร์ XOR ด้วยโครงข่ายประสาทเทียม

ในการทดลองนี้จะใช้หลักการของโครงข่ายประสาทเทียม (Artificial Neural Network) แบบหลายชั้น (Multi-layer) รวมถึงใช้วิธีการการเรียนรู้แบบย้อนกลับ มาช่วยในการแก้ไขปัญหาตรรกศาสตร์ XOR โดยมีลักษณะเป็นดังรูปที่ 4.8



รูปที่ 4.8 ลักษณะโครงสร้างของโครงข่ายประสาทเทียมแบบหลายชั้นที่ใช้ในการแก้ไขปัญหาตรรกศาสตร์ XOR

สำหรับปัญหาตรรกศาสตร์ XOR จะมีค่าความจริงตามตารางที่ 4.1 ซึ่งจะใช้ตารางนี้เป็นข้อมูลนำเข้าในแต่ละรอบของการเรียนรู้

ตารางที่ 4.1 ตารางค่าความจริงของตรรกศาสตร์ XOR

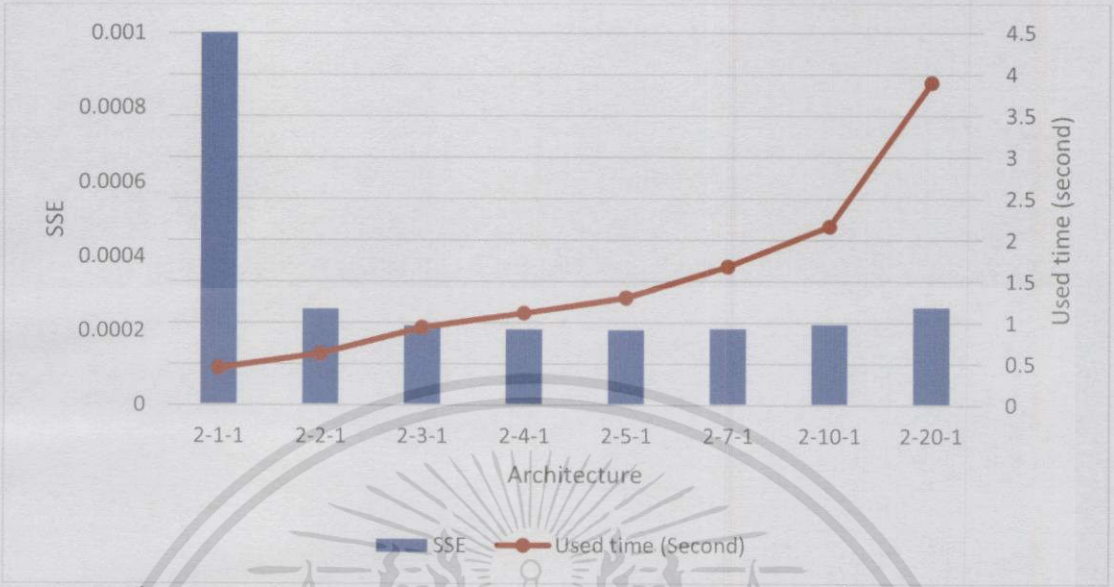
a	b	a xor b
0	0	0
0	1	1
1	0	1
1	1	0

สิ่งที่ทำการทดลองมีส่วนประกอบด้วยกันอยู่สองส่วนหลักๆคือ การเลือกใช้สถาปัตยกรรมและอัตราการเรียนรู้ (Learning rate) และในแต่ละรอบของการเรียนรู้ สิ่งที่เรากำลังพิจารณาคือผลรวมของค่าความผิดพลาดยกกำลังสอง (SSE: Sum square error)

ดังนั้นในการทดลองที่ 1 กำหนดให้ตัวแปรต้นคือ สถาปัตยกรรม ตัวแปรตามคือ ผลรวมของค่าความผิดพลาดยกกำลังสอง และเวลาที่ใช้ในการเรียนรู้ โดยมีตัวแปรควบคุมคือ จำนวนรอบของการเรียนรู้ (No. Epoch) อัตราการเรียนรู้ และน้ำหนักเริ่มต้นของแต่ละนิวรอน (ยกเว้นบางน้ำหนักที่จะเพิ่มขึ้นมาตามสถาปัตยกรรมให้มีค่าเริ่มต้นเป็น 0) โดยได้ผลการทดลองเป็นดังตารางที่ 4.2

ตารางที่ 4.2 ผลการทดลองการเรียนรู้ของโครงข่ายประสาทเทียมเพื่อแก้ไขปัญหาตรรกศาสตร์ XOR โดยมีตัวแปรต้นคือ สถาปัตยกรรม

No.	No. Epoch	Architecture	Learning rate	SSE	Used time (Second)
1	100000	2-1-1	0.2	0.660911570839732	0.453
2	100000	2-2-1	0.2	0.000260367682897	0.623
3	100000	2-3-1	0.2	0.000214528415378	0.940
4	100000	2-4-1	0.2	0.000204053134764	1.116
5	100000	2-5-1	0.2	0.000202068284420	1.302
6	100000	2-7-1	0.2	0.000206106241222	1.680
7	100000	2-10-1	0.2	0.000218137683686	2.164
8	100000	2-20-1	0.2	0.000264198603511	3.898



รูปที่ 4.9 แผนภูมิแสดงผลการทดลอง การเลือกใช้สถาปัตยกรรมที่มีผลต่อผลรวมค่าความผิดพลาดยกกำลังสองและเวลาที่ใช้ในการเรียนรู้

สรุปผลการทดลองที่ 2 ได้ว่าสถาปัตยกรรมมีผลต่อผลรวมของค่าความผิดพลาดยกกำลังสอง กล่าวคือหากมีนิวรอนที่ซ่อน (Hidden node) น้อยเกินไปถึงแม้จะใช้เวลาน้อย แต่การเรียนรู้ก็ลดน้อยลงไปด้วยสังเกตได้จากค่าผลรวมของค่าความผิดพลาดยกกำลังสอง แต่หากมากเกินไปก็ต้องอาศัยเวลามากขึ้นเพื่อต้องการจำนวนข้อมูลในการเรียนรู้เพื่อปรับน้ำหนักมากขึ้นเช่นกัน ทำให้ผลรวมของค่าความผิดพลาดยกกำลังสองนั้นไม่คุ้มกับเวลาที่เสียไป ดังนั้นในการแก้ไขปัญหาตรรกศาสตร์ XOR นี้ ดังนั้นสถาปัตยกรรมที่ควรใช้เพื่อให้คุ้มแก่ผลรวมของค่าความผิดพลาดยกกำลังสองที่ได้และเวลาที่เสียไป คือ 2-2-1

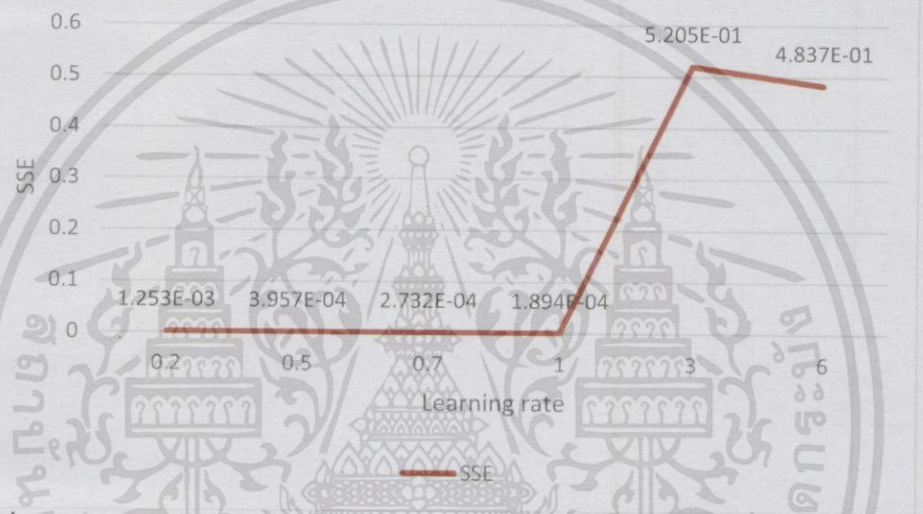
การทดลองที่ 2 กำหนดให้ตัวแปรต้นคือ อัตราการเรียนรู้ ตัวแปรตามคือ ค่าผลรวมของค่าความผิดพลาดยกกำลังสอง โดยมีตัวแปรควบคุมคือ จำนวนรอบของการเรียนรู้ (No. Epoch) สถาปัตยกรรม และน้ำหนักเริ่มต้นของแต่ละนิวรอน โดยได้ผลการทดลองเป็นดังตารางที่ 4.3

ตารางที่ 4.3 ผลการทดลองการเรียนรู้ของโครงข่ายประสาทเทียมเพื่อแก้ไขปัญหาตรรกศาสตร์ XOR โดยมีตัวแปรต้นคือ อัตราการเรียนรู้

No.	No. Epoch	Architecture	Learning rate	SSE
1	30000	2-2-1	0.2	0.00125305202973477
2	30000	2-2-1	0.5	0.000395728872339622
3	30000	2-2-1	0.7	0.000273297814968869

ตารางที่ 4.3 (ต่อ) ผลการทดลองการเรียนรู้ของโครงข่ายประสาทเทียมเพื่อแก้ไขปัญหาตรรกศาสตร์ XOR โดยมีตัวแปรต้นคือ อัตราการเรียนรู้

No.	No. Epoch	Architecture	Learning rate	SSE
4	30000	2-2-1	1.0	0.000189485828878295
5	30000	2-2-1	3.0	0.520550924776843
6	30000	2-2-1	6.0	0.483770359799246



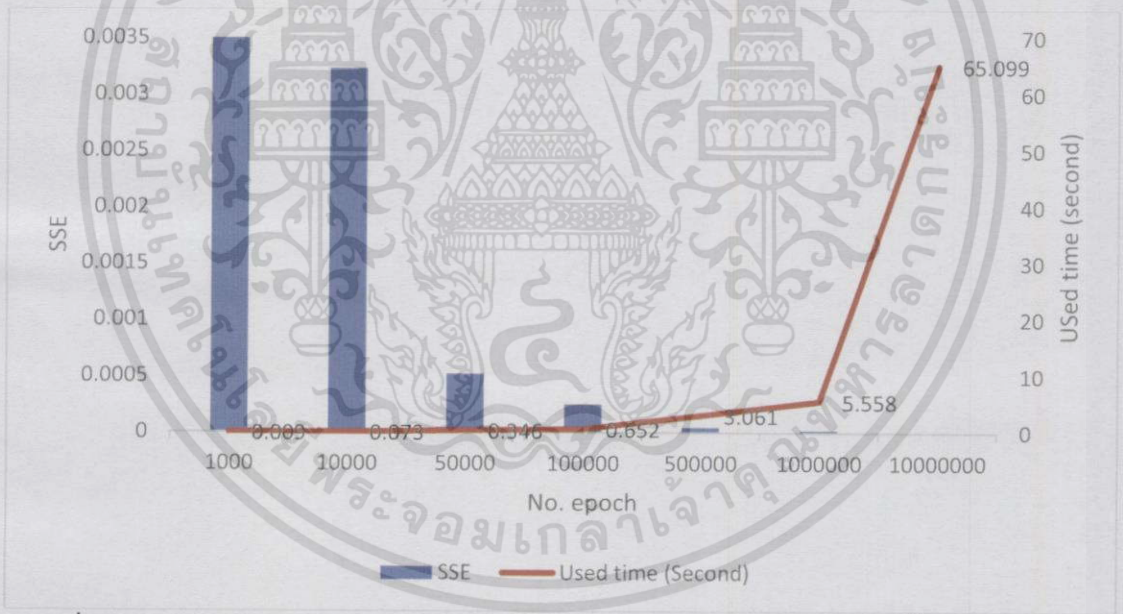
รูปที่ 4.10 แผนภูมิแสดงผลการทดลอง การเลือกใช้อัตราการเรียนรู้ที่มีผลต่อผลรวมค่าความผิดพลาดยกกำลังสอง

สรุปผลการทดลองที่ 2 ได้ว่าค่าอัตราการเรียนรู้มีผลต่อผลรวมของค่าความผิดพลาดยกกำลังสองกล่าวคือ หากอัตราการเรียนรู้มาก ทำให้การเรียนรู้ทำได้เร็วจะผลรวมของค่าความผิดพลาดยกกำลังสองน้อย แต่หากกำหนดอัตราการเรียนรู้มากเกินไป จะทำให้การเรียนรู้แต่ละครั้งนั้นถูกเชื่อถือมากกว่าข้อมูลเก่า ทำให้มีโอกาสที่จะเกิดความผิดพลาดหรือการเรียนรู้ที่มีผลรวมของค่าความผิดพลาดยกกำลังสองที่ไม่คงที่ได้

การทดลองที่ 3 เป็นการทดลองเพื่อดูค่าน้ำหนักตัวอย่างที่ได้จากการเรียนรู้ กำหนดให้ตัวแปรต้นคือจำนวนรอบในการเรียนรู้ ตัวแปรตามคือผลรวมของค่าความผิดพลาดยกกำลังสอง โดยกำหนดตัวแปรควบคุมคือการใช้สถาปัตยกรรม 2-2-1 และอัตราการเรียนรู้ที่ 0.2 โดยได้ผลการทดลองเป็นดังตารางที่ 4.4

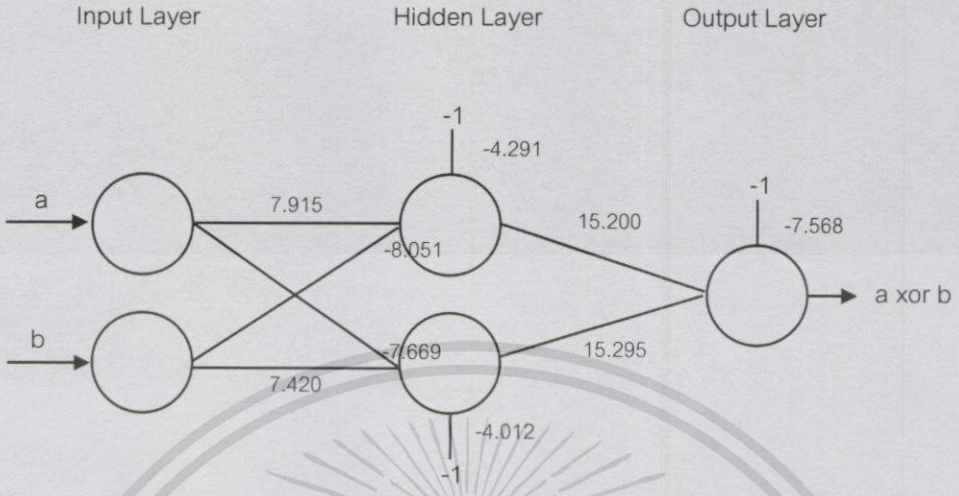
ตารางที่ 4.4 ผลการทดลองการเรียนรู้ของโครงข่ายประสาทเทียมเพื่อแก้ไขปัญหาตรรกศาสตร์ XOR โดยมีตัวแปรต้นคือ จำนวนรอบในการเรียนรู้

No.	No. Epoch	Architecture	Learning rate	SSE	Used time (Second)
1	1000	2-2-1	0.2	0.392777571604279	0.009
2	10000	2-2-1	0.2	0.003229607496425	0.073
3	50000	2-2-1	0.2	0.000522188359271	0.346
4	100000	2-2-1	0.2	0.000250814691577	0.652
5	500000	2-2-1	0.2	0.000047700231553	3.061
6	1000000	2-2-1	0.2	0.0000235636946236	5.558
7	10000000	2-2-1	0.2	0.0000023094770902	65.099



รูปที่ 4.11 แผนภูมิแสดงผลการทดลอง การเลือกใช้จำนวนรอบในการเรียนรู้ที่มีผลต่อผลรวมค่าความผิดพลาดยกกำลังสองและเวลาที่ใช้ในการเรียนรู้

สรุปผลการทดลองที่ 3 จากตารางผลการทดลองจะเห็นได้ว่าเมื่อเราใช้จำนวนรอบในการให้โครงข่ายประสาทเทียมเรียนรู้มากขึ้น ก็จะทำให้ผลรวมของค่าความผิดพลาดยกกำลังสองลดน้อยลง ซึ่งหมายถึงค่าความผิดพลาดจากข้อมูลจริงก็น้อยลงไปด้วย และจากตารางในรอบที่ 10,000,000 จะได้ค่าน้ำหนักในแต่ละนิวรอนดังรูปที่ 4.12



รูปที่ 4.12 โครงข่ายประสาทเทียมพร้อมค่าน้ำหนักที่สามารถนำไปใช้ในการแก้ไขปัญหาตรรกศาสตร์ XOR ได้

4.3. การทดลองสร้างส่วนของการเรียนรู้ข้อมูลจากพฤติกรรมการเล่นของผู้เล่น

ในการทดลองนี้จะเป็นการทดลองนำเอาโครงข่ายประสาทเทียมมาเพื่อเรียนรู้พฤติกรรมการเล่นของผู้เล่น ซึ่งพฤติกรรมการเล่นที่กล่าวถึงนี้ก็คือพฤติกรรมการจัดกองกำลังของผู้เล่นจะเป็นส่วนหนึ่งที่มีผลต่อผลการแพ้ชนะ และจะนำโครงข่ายประสาทเทียมที่ได้มาใช้ในการทำนายผลแพ้ชนะก่อนการเริ่มต่อสู้เพื่อใช้ในการปรับเปลี่ยนกองกำลังของฝ่ายยักษ์นั่นเอง

4.3.1 วิธีการทดลอง

1) เก็บข้อมูลผลการเล่นจริงๆระหว่างฝ่ายผู้เล่นและฝ่ายยักษ์จนได้ครบ 30 ตัวอย่าง
2) ทำการกำหนดค่าน้ำหนักของแต่ละโหนดในโครงข่ายประสาทเทียมแบบสุ่ม
3) นำข้อมูลที่เก็บได้มานั้น ป้อนเข้าไปยังโครงข่ายประสาทเทียม
4) รอการเรียนรู้ของโครงข่ายประสาทเทียม และจะใช้วิธีวัดจาก ผลรวมของค่าความผิดพลาดยกกำลังสอง (SSE: Sum square error) ถ้าหากไม่เกิน 0.00001 ถือว่ายอมรับการปรับน้ำหนักของแต่ละโหนดได้
5) นำโครงข่ายประสาทเทียมที่ได้เรียนรู้มาแล้ว มาทำการป้อนค่าข้อมูลเข้าไปและสังเกตผลลัพธ์จากการทำนาย

4.3.2 วิธีการประเมินและคำนวณค่า

เมื่อมีข้อมูลคือผลการเล่นจริงๆระหว่างผู้เล่นและยักษ์แล้ว ก็จะนำข้อมูลดังกล่าวมาทำการปรับ เพื่อให้เหมาะสมต่อการเรียนรู้ของโครงข่ายประสาทเทียม ประกอบไปด้วย

1) การปรับค่า (Re-scale) จำนวนทหาร จากที่เกมกำหนดให้ทหารสูงสุดและต่ำสุดของทหารแต่ละประเภทที่อนุญาตให้เป็นไปได้ต่ำสุดคือ 1 ตัวและสูงสุดคือ 10 ตัว และจะนำมาปรับค่าให้อยู่ในช่วงระหว่าง 0 ถึง 1 นั้น จึงมีวิธีปรับค่าจำนวนทหารได้โดยการ นำจำนวนทหารที่มีหารด้วย 10 นั่นเอง
2) การหาค่าเฉลี่ยสถิติผลแพ้ชนะของผู้เล่น ได้มาจากผลรวมของผลการต่อสู้ (ซึ่งในที่นี้เมื่อผู้เล่นชนะจะมีค่าเป็น 1 และเมื่อผู้เล่นแพ้จะมีค่าเป็น 0) นำมาหารกับจำนวนครั้งที่ผู้เล่นได้ทำการต่อสู้ไปทั้งหมดตั้งแต่เริ่มเกมมา

4.3.3 การทดลอง

ทดลองโดยสร้างโครงข่ายประสาทเทียมแบบหลายชั้น โดยมีชั้นซ่อนหนึ่งชั้นและใช้สถาปัตยกรรมแบบ 7-20-1 ที่มีรูปแบบการเรียนรู้แบบย้อนกลับ และมีอัตราการเรียนรู้ที่ 0.2 ซึ่งมีค่าที่นำเข้าไปเรียนรู้ดังนี้

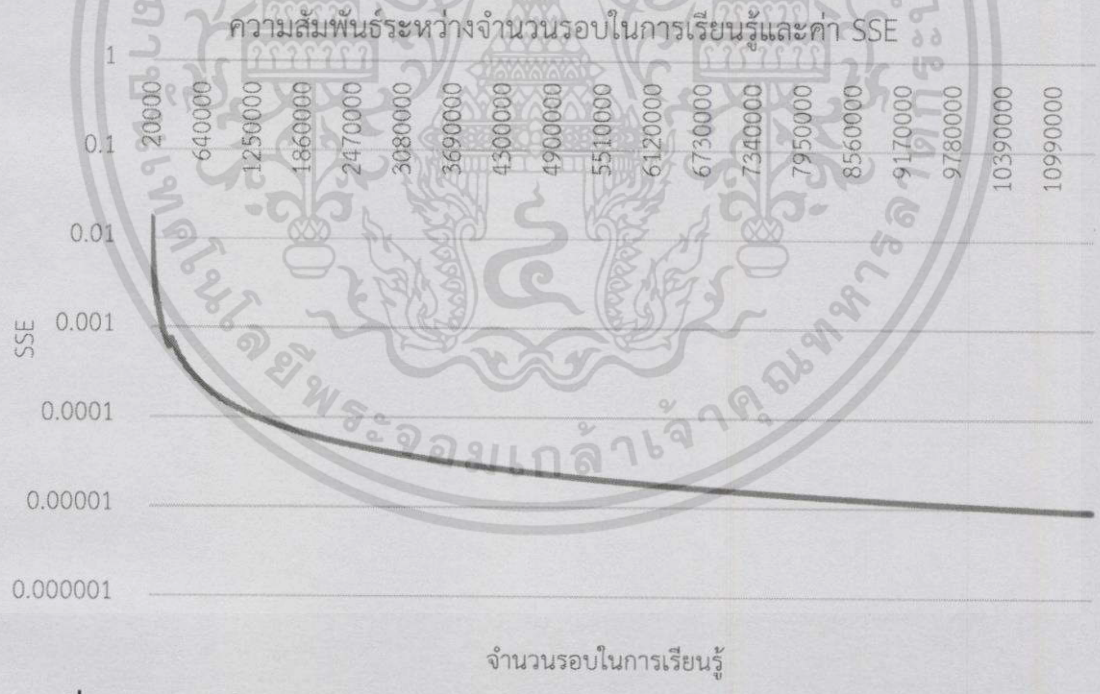
1) จำนวนทหารธนูของฝั่งผู้เล่นที่มีการปรับค่าแล้ว
2) จำนวนทหารดาบของฝั่งผู้เล่นที่มีการปรับค่าแล้ว
3) จำนวนทหารหอกของฝั่งผู้เล่นที่มีการปรับค่าแล้ว
4) จำนวนทหารธนูของฝั่งยักษ์ที่มีการปรับค่าแล้ว
5) จำนวนทหารดาบของฝั่งยักษ์ที่มีการปรับค่าแล้ว
6) จำนวนทหารหอกของฝั่งยักษ์ที่มีการปรับค่าแล้ว
7) ค่าเฉลี่ยสถิติผลแพ้ชนะของผู้เล่น
8) ผลการต่อสู้ (เป็นตัวระบุว่าผู้เล่นแพ้หรือชนะ)

4.3.4 ผลการทดลอง

ผู้พัฒนาได้นำจำนวนรอบที่ใช้ในการเรียนรู้และค่าผลรวมของค่าความผิดพลาดยกกำลังสองเป็นไปตามตารางที่ 4.5 และนำมาแสดงเป็นกราฟได้ดังรูปที่ 4.17 และหลังจากทดลองหยิบข้อมูลบางส่วนเพื่อจะมาทดลอบผลที่ได้คือ โครงข่ายประสาทเทียมสามารถทำนายผลการต่อสู้ได้ตรงตามข้อมูลต้นฉบับที่ใช้ในการเรียนรู้

ตารางที่ 4.5 ผลการทดลองการเรียนรู้ของโครงข่ายประสาทเทียมเพื่อเรียนรู้ผลการต่อสู้ซึ่งเกิดจากข้อมูลการต่อสู้ของผู้เล่นจริง

No.	No. Epoch	Architecture	Learning rate	SSE	Used time (Second)
1	20000	7-20-1	0.2	0.168777718165472	18.591
2	50000	7-20-1	0.2	0.004398016710459	46.272
3	100000	7-20-1	0.2	0.001819312043305	93.557
4	500000	7-20-1	0.2	0.000328892767292	467.936
5	1000000	7-20-1	0.2	0.000138842552359	943.288
6	5000000	7-20-1	0.2	0.000022568716894	4646.184
7	10000000	7-20-1	0.2	0.000010641989604	9251.629
8	10594135	7-20-1	0.2	0.000009999992898	9804.881



รูปที่ 4.13 แผนภูมิแสดงผลการทดลองความสัมพันธ์ระหว่างค่าความผิดพลาดยกกำลังสองและจำนวนรอบที่ใช้ในการเรียนรู้

4.4. การทดสอบโครงข่ายประสาทเทียมที่ได้ผ่านการเรียนรู้แล้ว

ในการทดลองนี้จะเป็นการนำเอาโครงข่ายประสาทเทียมที่ได้ผ่านการเรียนรู้แล้วมาทดสอบเพื่อจะสังเกตและประเมินว่าเหมาะสมต่อการนำไปใช้ในตัวเกมแล้วหรือไม่ ซึ่งในการทดลองนี้จะทำโดยการเก็บข้อมูลจากการเล่นจริงๆข้อมูลเล่นมา 10 ชุด (ซึ่งไม่ใช่ชุดเดียวกับที่ได้เรียนรู้ไปแล้ว) เป็นชุดข้อมูลที่โครงข่ายประสาทเทียมไม่เคยเห็นมาก่อน (Unseen test) และนำมาผ่านโครงข่ายประสาทเทียมและดูผลการทำนาย

ตารางที่ 4.6 ผลการทดสอบการทำนายของโครงข่ายประสาทเทียมที่ได้ผ่านการเรียนรู้แล้ว โดยทดสอบกับข้อมูลที่โครงข่ายประสาทเทียมไม่เคยเห็นมาก่อน

ชุดข้อมูลที่	ค่าที่ควรจะได้	ค่าที่ออกมา	เปอร์เซ็นต์ความผิดพลาด	เปอร์เซ็นต์ความผิดพลาดเฉลี่ยตั้งแต่เริ่มต้น
1	1	0.0205350368398	97.94649631602	97.94649631602
2	1	0.9789344708810	2.10655291190	50.02652461396
3	0	0.9999978517028	99.99978517028	66.68427813274
4	0	0.0008317696355	0.08317696355	50.03400284044
5	1	0.0018584250039	99.81415749961	59.99003377227
6	1	0.9999999999832	0.00000000168	49.99169481051
7	1	0.9999999996349	0.00000003651	42.85002412851
8	1	0.9999536733365	0.00463266635	37.49435019574
9	1	0.9997143199755	0.02856800245	33.33148550759
10	1	0.0000004641919	99.99995358081	39.99833231492

รูปที่ 4.14 แผนภูมิแสดงค่าความผิดพลาดที่ได้จากการทดสอบข้อมูลกับโครงข่ายประสาทเทียม โดยใช้ข้อมูลที่ไม่เคยเห็นมาก่อน

จากตารางที่ 4.6 และรูปที่ 4.14 แสดงให้เห็นว่าข้อมูลที่โครงข่ายประสาทเทียมไม่เคยเห็นมาก่อน 10 ชุด โครงข่ายประสาทเทียมมีอัตราการทำนายที่ผิดพลาดประมาณ 39.99% หรือกล่าวได้ว่าจากข้อมูล 10 ตัวอย่าง โครงข่ายประสาทเทียมทำนายถูกถึง 6 ตัวอย่าง ซึ่งแสดงให้เห็นว่าโครงข่ายประสาทเทียมมีความพร้อมในการนำไปใช้ทำนายได้ในระดับหนึ่ง

4.5. การทดลองนำโครงข่ายประสาทเทียมไปใช้ในเกม

ในการทดลองนี้จะเป็นการทดลองเปรียบเทียบระหว่างผลลัพธ์ของเกมที่นำโครงข่ายประสาทเทียมมาใช้ และผลลัพธ์ของเกมที่ไม่ได้นำโครงข่ายประสาทเทียมมาใช้ เพื่อนำผลลัพธ์มาวิเคราะห์ต่อไป โดยในการทดลองนี้จะมีตัวแปรควบคุมคือ จำนวนการเล่นเกมทั้งหมด 54 รอบ ใช้กฎกติกาเดียวกัน กฎการต่อสู้เหมือนกัน แผนที่เดียวกัน จำนวนทหารแต่ละประเภทของฝั่งผู้เล่นในแต่ละรอบจะถูกดึงมาจากรายการจำนวนทหารที่ถูกสุ่มไว้เสร็จเรียบร้อยแล้ว และทหารที่บังคับโดยผู้เล่นที่เป็นแม่ทัพ ในที่นี้จะนิ่งเฉยไม่ทำอะไร เพื่อตัดปัญหาเรื่องความไม่แน่นอนและความสามารถของผู้เล่นออกไป และจะทำการทดลองซ้ำ 3 ครั้ง

สำหรับการทดลองที่นำโครงข่ายประสาทเทียมมาใช้นั้นฝ่ายคอมพิวเตอร์นั้น มีวิธีการทดลองดังนี้คือ

1) ทำการปรับจำนวนทหารโจมตีระยะไกล, จำนวนทหารโจมตีระยะใกล้เสริมเกราะ, จำนวนทหารโจมตีระยะใกล้ ของคอมพิวเตอร์ในการทดลองแต่ละครั้ง ครั้งแรกเป็น 4-4-6 ครั้งที่สองเป็น 4-6-4 ครั้งที่สามเป็น 6-4-4 ตามลำดับ
2) ทำการปรับจำนวนทหารแต่ละประเภทของฝั่งผู้เล่นให้เป็นไปตามจำนวนที่ได้ถูกสุ่มไว้แล้วตามลำดับรอบของการเล่น
3) นำข้อมูลจำนวนทหารของฝั่งผู้เล่นและฝั่งคอมพิวเตอร์นำมาเป็นข้อมูลนำเข้าให้กับโครงข่ายประสาทเทียม
4) รอโครงข่ายประสาทเทียมจะส่งค่านำออก ซึ่งเป็นผลการทำนายการต่อสู้ว่าผู้เล่นมีโอกาสแพ้หรือชนะ
5) ดูผลการทำนายการต่อสู้ว่าผู้เล่นมีโอกาสแพ้หรือชนะ ถ้าผู้เล่นจะชนะ โปรแกรมควบคุมจะปรับเปลี่ยนจำนวนทหารของฝ่ายคอมพิวเตอร์ ตามกฎที่ได้กล่าวบทการออกแบบแล้วนั้น
6) เริ่มทำการเริ่มเล่นเกม
7) เมื่อจบการต่อสู้นี้นับเป็น 1 รอบ และทำการส่งผลลัพธ์และข้อมูลที่เกี่ยวข้องไปให้โครงข่ายประสาทเทียมเรียนรู้ข้อมูลใหม่
8) บันทึกผลและทำขั้นตอน 2-7 จนกว่าจะครบ 54 รอบ
9) เมื่อครบ 54 รอบแล้วถึงว่าจบการทดลอง 1 ครั้ง ทดลองแบบนี้ซ้ำจำนวน 3 ครั้ง

ซึ่งเมื่อได้ผลการทดลองแล้ว สามารถนำมาสรุปโดยมุ่งเน้นไปที่ข้อมูลจำนวนผลการแพ้ชนะของฝ่ายคอมพิวเตอร์ได้ดังตารางที่ 4.7

ตารางที่ 4.7 ผลการทดลองการเล่นเกมโดยที่นำโครงข่ายประสาทเทียมเข้ามาใช้ในเกม

การทดลองครั้งที่	จำนวนรอบทั้งหมดที่เล่นทั้งหมด (รอบ)	จำนวนรอบที่ฝ่ายคอมพิวเตอร์ชนะ (รอบ)	อัตราการชนะของฝ่ายคอมพิวเตอร์ (เปอร์เซ็นต์)
1	54	30	56.56%
2	54	30	56.56%
3	54	27	50.00%

สำหรับการทดลองที่ไม่ได้นำโครงข่ายประสาทเทียมมาใช้นั้นฝ่ายคอมพิวเตอร์นั้น มีวิธีการทดลองดังนี้คือ

1) ทำการปรับจำนวนทหารแต่ละประเภทของฝั่งผู้เล่นให้เป็นไปตามจำนวนที่ได้ถูกสุ่มไว้แล้วตามลำดับรอบของการเล่น
2) ทำการปรับจำนวนทหารแต่ละประเภทของฝั่งคอมพิวเตอร์แบบสุ่มทั้งหมด
3) เริ่มทำการเริ่มเล่นเกม
4) เมื่อจบการต่อสู้นี้นับเป็น 1 รอบ
5) บันทึกผลและทำขั้นตอน 1-4 จนกว่าจะครบ 54 รอบ
6) เมื่อครบ 54 รอบแล้วถึงว่าจบการทดลอง 1 ครั้ง ทดลองแบบนี้ซ้ำจำนวน 3 ครั้ง

ซึ่งเมื่อได้ผลการทดลองแล้ว สามารถนำมาสรุปโดยมุ่งเน้นไปที่ข้อมูลจำนวนผลการแพ้ชนะของฝ่ายคอมพิวเตอร์ได้ดังตารางที่ 4.8

ตารางที่ 4.8 ผลการทดลองการเล่นเกมโดยที่ไม่ได้นำโครงข่ายประสาทเทียมเข้ามาใช้ในเกม

การทดลองครั้งที่	จำนวนรอบทั้งหมดที่เล่นทั้งหมด (รอบ)	จำนวนรอบที่ฝ่ายคอมพิวเตอร์ชนะ (รอบ)	อัตราการชนะของฝ่ายคอมพิวเตอร์ (เปอร์เซ็นต์)
1	54	30	46.30%
2	54	24	44.44%
3	54	24	44.44%

จากการทดลองทั้งสองแบบสามารถสรุปได้ว่า การที่เกมนำโครงข่ายประสาทเทียมมาใช้ เพื่อช่วยในการทำนายผลและใช้ในการปรับจำนวนทหารแต่ละประเภทของศัตรูก่อนการต่อสู้นั้น ทำให้มีโอกาสที่ฝ่ายคอมพิวเตอร์จะมีโอกาสชนะมากขึ้น และสามารถตอบสองต่อผู้เล่นที่มีการจัดกลุ่มทหาร

อย่างหลากหลาย (ในการทดลองจำนวนทหารฝ่ายผู้เล่นจะถูกสุ่ม) ได้ดีกว่าการที่จะใช้ปรับจำนวนทหารแต่ละประเภทของฝ่ายศัตรูแบบสุ่มอย่างเดียว

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1. บทสรุป

ในการวิจัยและพัฒนาครั้งนี้ได้ทำการพัฒนาส่วนประกอบส่วนต่างๆ โดยสามารถสรุปในภาพรวมได้ดังต่อไปนี้

1) ส่วนของตัวเกมฝั่งผู้เล่น (Client) ได้มีการออกแบบตัวละคร เนื้อเรื่อง ลักษณะเกม แผนที่และฉาก รวมถึงกติกาและวิธีการเล่นของเกม อีกทั้งได้ทำการศึกษาการใช้งานและได้ทดลองใช้ส่วนประกอบต่างๆ ของ UDK (Unreal Development Kit) และได้นำมาสร้างเป็นตัวเกมขึ้นมาเป็นเกมสามมิติแบบเล่นได้หลายคนสื่อสารผ่านระบบเครือข่าย ผู้เล่นจะเข้ามาผ่านโปรแกรมเรียกเกม (Launcher) ภายในเกมจะมีสองแผนที่หลักคือแผนที่โลกซึ่งเปิดกว้างสำหรับให้ผู้เล่นทุกคนเดินอย่างอิสระและเมื่อปะทะกับกองกำลังของยักษ์ก็จะตัดเข้าแผนที่ต่อสู้อีกแผนที่หนึ่งเพื่อต่อสู้กับเฉพาะกองกำลังนั้นๆ เมื่อการต่อสู้เสร็จก็จะส่งข้อมูลที่เกี่ยวข้องไปที่เครื่องให้บริการ และผู้เล่นก็จะกลับเข้าสู่แผนที่โลกเหมือนเดิม เป็นเช่นนี้ไปเรื่อยๆ
2) ส่วนของโปรแกรมจัดการเกมและโปรแกรมที่เกี่ยวข้องนั้น ได้มีการติดตั้งระบบฐานข้อมูล รวมถึงการออกแบบฐานข้อมูล ออกแบบและพัฒนาโพรโทคอลและโปรแกรมเพื่อให้ตัวเกมสามารถติดต่อสื่อสารกับตัวโปรแกรมจัดการเกมเช่น กระบวนการในการแลกเปลี่ยนข้อมูลตำแหน่งของกองกำลัง, กระบวนการส่งผลลัพธ์ของการต่อสู้ เป็นต้น และได้พัฒนาระบบเพื่อมีหน้าที่จัดการควบคุมเกมในด้านต่างๆเช่น การควบคุมการปรับจำนวนทหารกองกำลัง การเคลื่อนย้ายกองกำลัง การสร้างกองกำลังขึ้นมาใหม่ เป็นต้นด้วย
3) ส่วนของโปรแกรมในการเรียนรู้ ได้มีการพัฒนาโครงข่ายประสาทเทียมแบบหลายชั้น โดยครั้งแรกผู้พัฒนาได้ใช้การทดลองตัวอย่างคือการแก้ไขปัญหาตรรกศาสตร์ XOR เพื่อทำการทดสอบแนวคิดในการนำโครงข่ายประสาทเทียมแบบหลายชั้น ที่มีวิธีการเรียนรู้แบบย้อนกลับ มาช่วยใช้ในการเรียนรู้ตัวแปรต่างๆในเกมเพื่อทำการเปลี่ยนสภาพแวดล้อมในเกม และต่อมาได้ทำการนำต้นแบบโครงข่ายประสาทเทียมแบบหลายชั้นนี้ มาปรับเพื่อใช้ในการเรียนรู้ข้อมูลซึ่งเป็นข้อมูลที่เกี่ยวข้องกับการเล่นจริงๆระหว่างฝ่ายผู้เล่นและยักษ์ และนำมาใช้ในการทำนายผลการแพ้ชนะในครั้งถัดๆไป เพื่อให้ยักษ์สามารถทำนายก่อนว่าจะแพ้หรือชนะ หากทำนายว่ายักษ์แพ้ ยักษ์จะปรับกองกำลังจากกฎที่ระบุไว้ตายตัวก่อนทำการต่อสู้กับผู้เล่น

5.2. ปัญหาอุปสรรคและแนวทางการแก้ไข

ในการทำงานพบปัญหาอุปสรรคดังต่อไปนี้

1) เนื่องจากกลุ่มของผู้พัฒนาเกมโดยใช้ Unreal Engine ในประเทศไทยนั้นไม่มาก และไม่เป็นที่นิยมแพร่หลาย จึงทำให้ยากที่จะหาคนช่วยเหลือในเวลาที่เกิดปัญหาหรือให้คำปรึกษาด้านการพัฒนาเกม แนวทางการแก้ไขคือ พยายามไปสอบถามกลุ่มผู้พัฒนาจากต่างประเทศแทน
2) ตัว UDK มีส่วนประกอบที่ซับซ้อน โค้ด (Code) ของโปรแกรมจะต้องพัฒนาในลักษณะของการเขียนโปรแกรมเชิงวัตถุ (Object-oriented programming) ทั้งหมด ซึ่งในบางครั้งจำเป็นต้องรู้ว่าแต่ละคลาสนั้นมีต้นตอมาจากคลาสไหน แนวทางการแก้ไขคือ การทำแผนภาพคลาส (Class diagram) ของโค้ดให้มากที่สุดเพื่อที่จะให้เห็นที่มาของคลาส
3) ตัว UDK ไม่ได้ออกแบบมาเพื่อสร้างเกมออนไลน์โดยเฉพาะดังนั้นจึงมีแค่คลาสเกี่ยวกับโพรโทคอลมาตรฐานแบบพื้นฐานมาให้คือโพรโทคอลทีซีพีเท่านั้น แนวทางการแก้ไขคือ ต้องทำการเขียนโปรแกรมพัฒนาคลาสต่อยอดเพื่อให้สามารถรองรับการแลกเปลี่ยนสื่อสารข้อมูลในแบบที่ต้องการได้ แต่อาจมีผลกระทบต่อมาคือทำให้ใช้เวลาในการศึกษามากขึ้นกว่าเดิม
4) เกมที่พัฒนาโดยใช้ UDK และมีการเปิดเผยซอสโค้ด (Source code) หาได้ยากทำให้ ไม่มีตัวอย่างในการพัฒนา หรือตัวอย่างในการใช้คลาสที่ตัว UDK มีมาให้ แนวทางการแก้ไขคือ ต้องทดลองใช้และประยุกต์ด้วยตนเอง แต่อาจมีผลกระทบต่อมาคือทำให้ใช้เวลาในการศึกษามากขึ้นกว่าเดิม
5) ลักษณะของเกมที่ออกแบบมาต้องใช้เทคโนโลยีและความรู้หลากหลายด้าน และบางอย่างเป็นสิ่งที่ค้นพบหลังจากการศึกษาวิจัยในรายละเอียดมาได้ระยะหนึ่งแล้ว เช่น การที่จะสร้างหน้าเพื่อตอบโต้กับผู้ใช้งาน (User interface) ในตัวเกมฝั่งผู้เล่นได้ ต้องอาศัย Flash ซึ่งทำให้ต้องศึกษาการใช้งานโปรแกรม Adobe Flash และภาษา Action Script ด้วย เป็นต้น ส่งผลกระทบให้ใช้เวลาในการศึกษานานมากขึ้น แนวทางการแก้ไข พยายามลดส่วนเสริมของเกมออกไปให้ได้มากที่สุด
6) เนื่องจากเป็นเกมออนไลน์ ทำให้ต้องมีเครื่องให้บริการ (Server) ที่เปิดให้บริการอยู่ตลอดเวลา รวมถึงจะต้องรองรับการเชื่อมจำนวนมาก ซึ่งในการวิจัยนี้อาจจะยังไม่มีทรัพยากรเพียงพอ แนวทางการแก้ไขคือ ใช้เครื่องคอมพิวเตอร์ของผู้พัฒนาในการเปิดเป็นเครื่องให้บริการไปก่อนและให้บริการในวงจำกัด
7) ในการติดต่อสื่อสารระหว่างตัวเกมฝั่งผู้เล่นและโปรแกรมจัดการเกมออนไลน์นั้น ข้อมูลได้ส่งผ่านโพรโทคอลทีซีพี และข้อมูลที่ส่งผ่านยังไม่ได้มีการเข้ารหัส จึงทำให้ผู้ที่ไม่ประสงค์ดีอาจใช้ช่องโหว่ในจุดนี้เพื่อทำการเปลี่ยนแปลงหรือปลอมแปลงค่าเพื่อปรับเปลี่ยนหรือโกง

ข้อมูลในเกม และอื่นๆได้ แนวทางการแก้ไขคือ หาวิธีการเข้ารหัสในการเชื่อมต่อระหว่างตัวเกมฝั่งผู้เล่นและโปรแกรมจัดการเกมในเบื้องต้นอาจใช้การเข้ารหัสแบบสมมาตรมาใช้

8) เนื่องจากผู้พัฒนาเคยใช้ระบบฐานข้อมูล MySQL มาก่อน แต่ในการพัฒนาครั้งนี้ใช้ Microsoft SQL Server 2013 Express ซึ่งมีการใช้งานที่ค่อนข้างซับซ้อนและไม่มีระบบบางอย่างเหมือนที่ผู้พัฒนาเคยใช้ใน MySQL อาจจะเป็นเพราะเป็นลักษณะเฉพาะ (Proprietary) ที่มีอยู่ในบางโปรแกรมเท่านั้น เช่นชนิดข้อมูล ENUM (Enumeration) พบใน MySQL แต่ไม่พบใน Microsoft SQL Server 2013 Express แนวทางการแก้ไขคือ ค้นคว้าการใช้งาน Microsoft SQL Server เพิ่ม ซึ่งพบว่ามีลักษณะการใช้งานแบบอื่นที่ให้ผลเหมือนกับการใช้ ENUM เหมือนกัน
9) ในการพัฒนาโปรแกรมในส่วนของการเรียนรู้ ซึ่งเราเลือกใช้โครงข่ายประสาทเทียม (Artificial Neural Network) นั้น ในขั้นตอนการดำเนินการจะต้องใช้เวลาในการเรียนรู้ (Training) เป็นระยะเวลาหนึ่ง ซึ่งจากการทดลองหากใช้จำนวนข้อมูลนำเข้า (Input) ที่มากขึ้น หรือจำนวนข้อมูลที่นำเข้าไปเรียนรู้มากขึ้น ก็จะใช้เวลาเรียนรู้ในการแก้ไขปัญหานาน แนวทางการแก้ไขคือ พยายามลดจำนวนข้อมูลนำเข้าที่ไม่จำเป็นหรือไม่เกี่ยวข้องออกไปให้ได้มากที่สุด
10) การเลือกสิ่งแวดล้อมที่จะนำมาเปลี่ยนแปลงนั้นมีหลากหลายมาก ซึ่งจะต้องทำการเลือกปัจจัยต่างๆมา และทำการทดลองว่าถ้านำปัจจัยนี้มาใช้แล้วจะสามารถช่วยให้สิ่งแวดล้อมเปลี่ยนแปลงหรือมีผลกระทบต่อตัวเกมมากน้อยเพียงใด จึงทำให้ต้องใช้เวลาในการทดลองและวิเคราะห์หาปัจจัยดังกล่าว ซึ่งจำเป็นต้องใช้เวลาในการเลือกปัจจัยมาทดลองนานระยะหนึ่ง แนวทางการแก้ไขคือ ต้องวิเคราะห์ปัจจัยที่ไม่มีผลเกี่ยวข้องออกไปก่อน และเลือกปัจจัยที่มีผลกระทบมากที่สุดมาทดลองก่อน
11) เนื่องจากพฤติกรรมของผู้เล่นที่เข้ามาเล่นเกมบางคนอาจมีฝีมือการเล่นไม่คงที่ ทำให้ส่งผลกระทบต่อผลการต่อสู้ ซึ่งเป็นสิ่งสำคัญในการนำไปใช้ประเมินผู้เล่นและเปลี่ยนแปลงจำนวนทหารในกองกำลังทหารของฝั่งยักษ์ แนวทางการแก้ไขคือ หาปัจจัยที่เกี่ยวข้องอื่นที่ได้มาจากผู้เล่น เพื่อลดความไม่แน่นอนนี้
12) การปรับความสมดุลของทหารทั้งสามประเภท ทหารแต่ละประเภทมีลักษณะที่เป็นจุดดีจุดด้อยต่างกันอยู่แล้วทำให้มีผลต่อการต่อสู้ที่ต่างกัน แต่ต้องเกิดความสมดุลกัน ไม่ให้ทหารประเภทใดได้เปรียบกว่าประเภทใด เช่น ถ้าการปรับค่าไม่สมดุลดีพอ การปรับทหารธนูมากที่สุดเท่าที่เป็นไปได้แล้วจะทำให้ได้เปรียบและมีโอกาสชนะสูงมาก เป็นต้น แนวทางการแก้ไขคือ ต้องทดลองเล่นและสังเกตความสมดุล จากนั้นปรับค่าลักษณะเฉพาะของทหารแต่ละประเภทให้เหมาะสม

5.3. แนวทางในการพัฒนาต่อ

1) พัฒนาระบบสร้างสิ่งแวดล้อมในเกมต่างๆมาประกอบเช่น การเคลื่อนที่ของฝูงสัตว์ การเปลี่ยนแปลงของสภาพอากาศ และช่วงเวลากลางวัน-กลางคืน เป็นต้น
2) พัฒนาระบบเกมให้ยูนิตสามารถใส่ไอเทมเพื่อเพิ่มความสามารถของยูนิตได้
3) พัฒนาระบบเกมโดยการเพิ่มระบบจัดการเควส (Quest) โดยให้เควสแต่ละอัน ขึ้นอยู่กับสภาพแวดล้อมของเกมในขณะนั้น เพื่อเพิ่มแรงจูงใจไม่ให้พูดเล่นเบื่อกับเกมมากขึ้น
4) พัฒนาตัวเกมฝั่งผู้เล่นให้สามารถทำการต่อสู้กันเองระหว่างผู้เล่นได้ (PVP: Player Versus Player)
5) พัฒนาโปรแกรมในส่วนของการเรียนรู้ ให้สามารถเรียนรู้ปัจจัยอื่นๆที่มีผลต่อการเปลี่ยนแปลงสภาพแวดล้อมได้มากขึ้น
6) พัฒนาการเชื่อมต่อระหว่างตัวเกมและตัวจัดการเกมออนไลน์ให้มีความปลอดภัยมากขึ้น

บรรณานุกรม

[1] Michael Negnevitsky. **Artificial Intelligence A Guide to Intelligent Systems.** 2nd Ed. Pearson Education Limited. 2005.

[2] Alex Berson. **Client/Server Architecture.** McGraw-Hill, Inc. 1994.

[3] Behrouz A. Forouzan. **TCP/IP Protocol Suite.** 3rd Ed. McGraw-Hill, Inc. 2007.

[4] Rachel Cordone. **Unreal Development Kit Game Programming with UnrealScript: Beginner's Guide.** Packt Publishing. 2011.

[5] Wikipedia. **"Berkeley sockets."** [Online]. Available : http://en.wikipedia.org/wiki/Berkeley_sockets. 2013.

[6] Epic Games, Inc. **"Characters Technical Guide."** [Online]. Available : http://udn.epicgames.com/Three/CharactersTechnicalGuide.html. 2014.

[7] Epic Games, Inc. **"Unreal Engine 3 Command-Line Arguments."** [Online]. Available : http://udn.epicgames.com/Three/CommandLineArguments.html . 2014.

[8] Epic Games, Inc. **"Unreal Development Kit (UDK) documentation."** [Online]. Available : http://www.unrealengine.com/en/udk/documentation. 2014.