เกมการ์ดบนเน็ตเวิร์ก

Card Game on Network

นางสาวพรทิพย์ จันทเวช

นายพุทธิชัย สัมมา

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

เกมการ์ดบนเน็ตเวิร์ก

Card Game on Network

โดย

นางสาวพรทิพย์ จันทเวช

นายพุทธิชัย สัมมา

อาจารย์ที่ปรึกษา

อ.เกียรติกูล เจียรนัยธนะกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

ปริญญานิพนธ์ปีการศึกษา 2544

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เกมการ์ดบนเน็ตเวิร์ก

(Card Game on Network)

ผู้จัดทำ

1. นางสาวพรทิพย์ จันทเวช รหัสประจำตัว 41014293
2. นายพุทธิชัย สัมมา รหัสประจำตัว 41014318

_____ อาจารย์ที่ปรึกษา

(อ.เกียรติกูล เจียรนัยธนะกิจ)

เกมการ์ดบนเน็ตเวิร์ก

นางสาวพรทิพย์ จันทเวช
นายพุทธิชัย สัมมา
อ.เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2544

บทคัดย่อ

ปัจจุบันนี้เกมประเภทหลายผู้เล่นได้มีการแพร่หลายออกไปเป็นอย่างมาก และผู้คนก็ให้ความสนใจในเกมประเภทนี้เป็นอย่างดี ไม่ว่าจะเป็นเกมที่เล่นผ่านเครือข่าย หรือผ่านระบบเครือข่ายภายใน หรือ ผ่านอินเตอร์เน็ต เกมที่ออกวางตลาดในระยะหลังนี้โดยเฉพาะใน 3 ถึง 5 ปีที่ผ่านมาจำเป็นต้องมีการทำให้ผู้เล่นสามารถเล่นแบบหลายผู้เล่นได้ เพื่อตอบสนองความต้องการของผู้เล่น ใน 5 ปีที่ผ่านมานี้ผู้เล่นส่วนมากให้ความสนใจกับเกมประเภท หลายผู้เล่นมากกว่าเกมที่เล่นคนเดียวเสียอีก ดังนั้น จุดประสงค์ของเราก็เพื่อทำเกมแบบหลายผู้เล่นเพื่อตอบสนองความต้องการของผู้เล่นเหล่านี้ โดยเกมที่นำเสนอจะเป็นเกมประเภทเกมการ์ด ซึ่งผู้เล่นจำเป็นต้องอาศัยการคิด การวางแผน และการตัดสินใจที่ดีเพื่อชนะในเกม แต่ไม่ใช่แค่นั้นที่จะทำให้ชนะเกมนี้ได้ ผู้เล่นจำเป็นต้องมีชุดการ์ดที่ดี รวมทั้งโชคด้วยในบางครั้ง ซึ่งจะเห็นได้ว่าผู้เล่นที่วางแผนดี ตัดสินใจดี และความคิดเยี่ยมอาจจะไม่ชนะในเกมนี้ก็ได้ ซึ่งจะทำให้เกมนี้มีความสนุกยิ่งขึ้น แต่โดยรวมแล้ว ผู้เล่นที่เก่งในเรื่องของความคิด และการวางแผนก็จะได้เปรียบอีกฝ่ายหนึ่งอย่างแน่นอน จึงเป็นส่วนที่กระตุ้นให้ผู้ที่เล่นเกมได้ใช้ความคิดพิจารณา และการวางแผนที่รอบคอบ

Card Game on Network

Porntip Chantawech
Bhuttichai Summar
Kietikul Jearanaitanakij Advisor

ABSTRACT

Multi-player Game is very interesting for all people. Many games have issued in last 3 to 5 years need to made it in multiple player modes. So people can play game with another player through network LAN or Internet. In last 5 years there was evidence that people tend to play in multi-player game than single game. The purpose of this project is to make network multi-player game. Type of this game is card game. Player can play this game is several mode that is story mode, free dual mode -which is single mode- and multiple mode. This game encourage player to think a lot for playing, choosing his strategy, arranging his Deck and selecting best each turn of play. Although to win this game player must have not only good thinking and planning but also player must have a good deck and luck too. So a best player may lose in a game, however a good thinking and planning will have an advantage.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้คงไม่อาจเสร็จได้ด้วยดี หากไม่ได้รับความช่วยเหลือและร่วมมือจากหลายๆ ฝ่ายด้วยกัน บุคคลแรกที่ต้องกล่าวถึงเพราะเป็นส่วนสำคัญที่ทำให้วิทยานิพนธ์นี้เสร็จลงได้ก็คือ อาจารย์ เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่ให้ความเอาใจใส่ แนะนำ และช่วยเหลือเสมอมา ซึ่งต้องขอขอบพระคุณเป็นอย่างมาก

และต้องขอขอบพระคุณบุคคลสำคัญที่สุดที่ทำให้ข้าพเจ้ามีวันนี้ ก็คือ บิดา มารดา อันเป็นที่เคารพรักยิ่ง ซึ่งได้เลี้ยงดูผู้เขียนมาเป็นอย่างดี พร้อมทั้งให้โอกาสในการศึกษาอย่างเต็มที่และยังให้กำลังใจ เอาใจใส่เสมอมาในทุก ๆ ด้านอันหาที่เปรียบมิได้ ข้าพเจ้าขอระลึกในพระคุณอันสุดประมาณและขอกราบขอบพระคุณมา ณ ที่นี้

นางสาวพรทิพย์ จันทเวช

นายพุทธิชัย สัมมา

สารบัญ

สารบัญรูปภาพ

หน้าที่

สารบัญตาราง

บทที่ 1
บทนำ

1.1 ที่มา และความสำคัญของโครงงาน

เนื่องจากในปัจจุบันนี้ระบบคอมพิวเตอร์มีการพัฒนาไปอย่างรวดเร็ว จากเครื่องคอมพิวเตอร์ขนาดใหญ่ก็กลายเป็นเครื่องคอมพิวเตอร์แบบกระเป๋าหิ้ว ที่สามารถพกพาได้ จากการเล่นคอมพิวเตอร์แบบเล่นคนเดียวก็กลายมาเป็นการระบบคอมพิวเตอร์บนเครือข่าย และบนอินเตอร์เน็ต ซึ่งระบบเครือข่าย และระบบอินเตอร์เน็ตในปัจจุบันนี้ มีความก้าวหน้า และมีความต้องการใช้เป็นอย่างมาก ทั้งในเรื่องของธุรกิจก็มีความต้องการการติดต่อแบบทันทีทันใด และการส่งข้อมูลภายในอย่างรวดเร็ว มีการพัฒนาส่วนช่วยส่งเสริมระบบเครือข่าย รวมทั้งโปรแกรมที่ใช้ทำงานบนเครือข่ายขึ้นมามากมาย ทั้งนี้เพื่อให้ธุรกิจสามารถดำเนินไปได้อย่างราบรื่น ถูกต้อง และรวดเร็ว

เกมก็เป็นโปรแกรมอีกประเภทหนึ่งที่มีวัตถุประสงค์เพื่อให้ผู้ที่ใช้งานได้รับความบันเทิง ผ่อนคลาย และบางเกมอาจเป็นการกระตุ้นความคิดสร้างสรรค์ และความสามารถของผู้เล่นให้มากขึ้น โปรแกรมเกมก็มีอยู่มากมายหลายประเภท ทั้งเกมประเภทวางแผน เกมประเภทเกมภาษา เกมประเภทกีฬา เกมประเภทต่อสู้ ฯลฯ ทั้งนี้ในปัจจุบัน โปรแกรมเกมได้มีการเขียน และวางจำหน่ายเป็นจำนวนมาก ผู้คนมีทางเลือกมากมายในการเลือกเล่นเกม เกมประเภทหลายผู้เล่น ทำให้ผู้เล่นสามารถเล่นพร้อมๆ กันได้หลายๆ คน อาจจะเล่นผ่านระบบเครือข่าย หรือเล่นผ่านระบบอินเตอร์เน็ตก็ตามที ปัจจุบันนี้ได้รับความนิยมเป็นอย่างมาก เกมที่วางจำหน่ายในปัจจุบันจำเป็นต้องเพิ่มทางเลือก ในการเล่นแบบหลายผู้เล่นไว้ให้ผู้เล่นด้วย เพื่อเป็นการรองรับความต้องการของผู้เล่น

การเขียนโปรแกรมประเภทเกมผ่านเครือข่ายเป็นสิ่งที่จะต้องคำนึงถึงผู้เล่นทุกๆ คน โปรแกรมจะต้องสามารถติดต่อ และสื่อสารให้ผู้เล่นทุกๆ คนเข้าใจสถานะของเกมเป็นหนึ่งเดียวกัน รวมทั้งข้อมูลส่วนบุคคลของแต่ละบุคคลก็ต้องมีการจัดการที่ดี รวมไปจนถึงข้อมูลที่เป็นข้อมูลส่วนกลางก็ต้องจัดการล็อกไว้ไม่ให้มีคนเข้ามาใช้พร้อมกันมากกว่า 1 คน โดยที่คนหนึ่งต้องการเปลี่ยนแปลงค่า ทั้งๆ ที่คนอื่นกำลังอ่านอยู่

ความฉลาดของเกมก็เป็นอีกเรื่องที่ต้องคำนึงถึง ในทุกๆ เกมจะเล่นสนุกได้ผู้เล่นที่เป็นคอมพิวเตอร์ต้องมีความสามารถและมีความฉลาดพอที่จะต่อสู้กับคนได้อย่างทัดเทียม แต่ก็ไม่ใช่ว่าจะเก่งมากจนผู้เล่นไม่สามารถชนะได้ โดยเฉพาะเกมที่เป็นเกมประเภทใช้ความคิดจำเป็นจะต้องมีการวางวิธีการเล่นเกมที่มีความสามารถให้กับคอมพิวเตอร์ อาจจะใช้การหาวิธีที่ดีที่สุดที่เป็นไปได้ หรือถ้าหาไม่ได้ก็อาจจะต้องใช้การสุ่มร่วมเข้ามาด้วย ขึ้นอยู่กับสถานการณ์

เกมประเภทเกมการ์ดก็เป็นอีกประเภทหนึ่งซึ่งปัจจุบันมีผู้เล่นให้ความสนใจเป็นจำนวนมาก เนื่องจากมีการเล่นเกมการ์ดที่เป็นเกมจริงๆ ใช้การ์ดจริงๆ กันมาก ผู้เล่นที่เล่นเกมการ์ดแบบนี้ต้องการจะเล่นเกมผ่านระบบเครือข่าย หรือผ่านคอมพิวเตอร์ เพื่อที่เขาจะได้ไม่ต้องพกการ์ดไปมา แค่เขาเปิดคอมพิวเตอร์แล้วดึงชุดการ์ดของตัวเองออกมา แล้วต่อเข้าไปในเครือข่ายก็สามารถเล่นได้แล้ว

โครงงานนี้เป็นเกมประเภทเกมการ์ดซึ่งจะมีการเชื่อมต่อให้สามารถเล่นผ่านระบบเครือข่าย ซึ่งจะสามารถเล่นได้หลายผู้เล่น

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อศึกษาการเขียนโปรแกรมด้วยวิชวลเบสิก (Visual Basic)
2. เพื่อศึกษาการเขียนโปรแกรมแบบหลายผู้เล่นผ่านเครือข่ายโดยใช้ไมโครซ็อฟวินซ็อค (Microsoft Winsock)
3. นำเสนอระบบปัญญาประดิษฐ์ของเกมประเภทเกมการ์ดเพื่อให้คอมพิวเตอร์สามารถเล่นกับผู้เล่นได้

1.3 ขอบเขตของโครงงาน

โครงงานนี้จะมีการศึกษาเกี่ยวกับการเขียนโปรแกรมด้วยวิชวลเบสิกโดยเป็นการเขียนโปรแกรมผ่านเครือข่ายซึ่งใช้วินซ็อคคอนโทรลสำหรับการจัดการและส่งข้อมูลผ่านเครือข่าย ผู้เล่นก็สามารถจะเลือกเล่นเกมนี้ได้ทั้งในแบบเล่นคนเดียวแบบสู้กับคอมพิวเตอร์หรือจะเล่นตามเนื้อเรื่องและเล่นแบบหลายผู้เล่นผ่านระบบเครือข่าย โดยจะเน้นที่การวางความสามารถให้กับคอมพิวเตอร์ให้สามารถค้นหาเส้นทางที่ดีที่สุดในการเล่นเพื่อสู้กับผู้เล่นคนอื่นๆ โดยในที่นี้เราจำเป็นต้องใส่ค่าของแต่ละสถานการณ์เพื่อให้คอมพิวเตอร์สามารถทราบได้ว่า จะเลือกทางไหนดี

1.4 วิธีการดำเนินงาน

- ศึกษาทฤษฎีพื้นฐานในเรื่องต่างๆ ทั้งการเขียนโปรแกรมด้วยวิชวลเบสิก, การทำงานของวินซ็อคคอนโทรลและปัญญาประดิษฐ์
- เลือกวิธีการหาทางที่ดีที่สุดในการเล่นของคอมพิวเตอร์ รวมทั้งให้ค่าความได้เปรียบของแต่ละสถานการณ์ให้ถูกต้อง
- ออกแบบโครงสร้างของข้อมูลที่จะจัดเก็บ
- จัดทำเกี่ยวกับเรื่องของรูปภาพ, รูปการ์ดและส่วนต่างๆ ในเกม รวมทั้งเสียงและมูฟวี่สั้นๆอีกด้วย
- ออกแบบการโปรแกรมทั้งหมด
- เริ่มต้นเขียนโปรแกรม และสรุป

1.5 กติกาการเล่น

1. ผู้เล่นทั้ง 2 ฝ่ายจะต้องจัดชุดการ์ดของตัวเอง ในที่นี้จะเรียกว่าชุดการ์ด (Deck) จำนวน 50 ใบเพื่อนำมาใช้ต่อสู้กัน โดยการ์ดแต่ละใบห้ามจัดเข้ามาซ้ำกันมากกว่า 2 ใบ
2. การเล่นแบ่งเป็นเทิร์น (Turn) แต่ละเทิร์นผู้เล่นจะสามารถเล่นได้ 1 ครั้ง ผลัดกัน
3. ชนิดของการ์ดแบ่งเป็นการ์ดสิ่งมีชีวิต (Creature) และการ์ดเวทมนตร์ (Magic)

4. การ์ดสิ่งมีชีวิตแต่ละใบจะมีค่าพลังโจมตี พลังป้องกัน ค่าการบูชายัญ และค่าการเรียกใช้ เป็นของตัวเอง
5. การเล่นก็คือลงการ์ดไปสู้กันในสนาม (Field)
6. เมื่อเริ่มเล่นผู้เล่นจะได้จั่วการ์ดขึ้นมาคนละ 5 ใบ แล้วลงการ์ดไป เมื่อจบเทิร์นใช้การ์ดไปเท่าไรก็ตาม ก่อนขึ้นเทิร์นใหม่ต้องจั่วขึ้นมาให้ครบ 5 ใบ ก่อนเริ่มเทิร์นของตัวเอง
7. วิธีการลงการ์ดไปสู้กัน มีข้อกำหนดดังนี้
   - การ์ดสิ่งมีชีวิตระดับต่ำ (Level 1-3) สามารถลงไปบนสนามได้ทันที
   - การ์ดสิ่งมีชีวิตระดับสูง (Level 4 up) การจะลงจำเป็นต้องทิ้งการ์ดบนสนามเพื่อทำการบูชายัญเป็นค่าเรียกสิ่งมีชีวิตขึ้นมา ตามค่าเรียกของการ์ดใบนั้น และค่าการบูชายัญของการ์ดที่ทิ้งไป ซึ่งอาจต้องทิ้งมากกว่า 1 ใบ เพื่อเรียกการ์ดที่มีความสามารถสูงมากๆ ลงมา
   - การ์ดเวทมนตร์แต่ละใบสามารถใช้ได้เลย ไม่มีค่าการเรียก หรือการ์ดเวทมนตร์ประเภทกับดักก็สามารถใช้ได้เช่นกัน โดยวางลงมาบนสนามเพื่อวางกับดักฝ่ายตรงข้าม
   - แต่ละเทิร์นผู้เล่นสามารถลงการ์ดเวทมนตร์ได้ไม่เกิน 3 ใบ ส่วนการ์ดสิ่งมีชีวิตนั้นลงได้เทิร์นละ 1 ใบ
8. วิธีการโจมตีกัน มีดังนี้
   - ใช้การ์ดสิ่งมีชีวิตที่อยู่บนสนามโจมตีใส่การ์ดสิ่งมีชีวิตของฝ่ายตรงข้าม เมื่อโจมตีจะนำพลังโจมตีของการ์ดที่โจมตี มาเปรียบเทียบกับพลังป้องกันของการ์ดที่ถูกโจมตี หากใครมากกว่ากัน การ์ดอีกฝ่ายก็ต้องทิ้งไป และค่าความแตกต่างของพลังดังกล่าวนี้จะถูกนำไปลบออกจากพลังชีวิตของผู้เล่นฝ่ายที่แพ้
   - การ์ดแต่ละใบในสนามจะโจมตีได้เพียง 1 ครั้งในแต่ละเทิร์น ดังนั้นหากมีการ์ดบนสนามมากย่อมได้เปรียบ
   - หากฝ่ายตรงข้ามไม่มีการ์ดบนสนามไว้ป้องกัน ผู้เล่นฝ่ายโจมตีสามารถใช้การ์ดโจมตีใส่พลังของผู้เล่นโดยตรงได้เลย
   - สามารถใช้การ์ดเวทมนตร์ที่เป็นการ์ดโจมตีใส่พลังของผู้เล่นโดยตรง โจมตีใส่ผู้เล่นเลยก็ได้ ค่าที่โจมตีจะถูกนำไปหักลบออกจากพลังของฝ่ายตรงข้าม
   - การ์ดเวทมนตร์เพิ่มพลัง ใช้เพื่อเพิ่มพลังชีวิตของผู้ใช้ให้เพิ่มขึ้นตามที่กำหนด
9. ผู้เล่นแต่ละฝั่งจะมีพลังชีวิตคนละ 30 แต้ม ใครพลังชีวิตหมดก่อนเป็นฝ่ายแพ้
10. หากผู้เล่นคนใดไม่สามารถจั่วการ์ดขึ้นมาให้ครบ 5 ใบในมือก่อนเริ่มเทิร์น เนื่องมาจากใช้การ์ดหมดก็จะแพ้ทันที
11. เมื่อจบแต่ละเกม ผู้ที่ชนะจะได้การ์ดจากชุดการ์ดของผู้แพ้ 1 ใบ แบบสุ่ม

1.6 คุณสมบัติของโปรแกรม (Specification)

1. ผู้เล่นสามารถเล่นได้ตั้งแต่ 1 และ 2 คนผ่านเครือข่าย
2. การเล่นจะมีชนิดให้เลือกว่าจะเล่นตามเนื้อเรื่อง เล่นสู้เพื่อหาการ์ด สู้กับผู้เล่นอื่น รวมทั้งยังมีร้านขายการ์ดให้ซื้อการ์ดได้
3. การเล่นแต่ละครั้งไม่ว่าจะแพ้ หรือชนะนอกจากจะได้การ์ดแล้วยังจะได้คะแนนมาด้วย ซึ่งจะแตกต่างกันสำหรับผู้ชนะ และผู้แพ้ คือ ผู้ชนะจะได้คะแนนมากกว่า และผู้เล่นสามารถสะสมคะแนนนี้เพื่อใช้ในการซื้อการ์ดได้
4. การเล่นแบบหลายผู้เล่นจะใช้วิธีให้มีผู้เล่นคนหนึ่งเป็นผู้เปิดห้องไว้ ผู้เล่นที่เข้ามาทีหลังจะเข้ามาร่วมในห้อง หรือไม่อย่างนั้นก็ต้องเปิดห้องใหม่
5. ดังนั้นหากจะเล่นก็ต้องมีการลงโปรแกรมนี้ไว้ที่เครื่องนั้นๆ จึงจะเล่นได้ และโปรแกรมต้องทำงานได้ใน 2 แบบ คือในกรณีเป็นไคลเอ็นท์ (Client) และในกรณีเป็นเซิร์ฟเวอร์
6. โปรแกรมจะมีการเก็บข้อมูลของผู้เล่นทั้งหมดรวมทั้งตัวโปรแกรมที่ใช้ทำงานไว้บนเครื่องของตัวเอง โดยเครื่องที่เปิดห้องจะทำหน้าที่เป็นเซิร์ฟเวอร์ (Server) โดยมีหน้าที่เพียงแค่จัดเทิร์นให้แต่ละการเล่นและการจัดการสุ่มการ์ดของผู้แพ้ให้ผู้ชนะ
7. หากผู้เล่นต้องการนำชุดการ์ดของตัวเองที่มีอยู่บนเครื่องตัวเองไปเล่นบนเครื่องอื่นๆ ก็ต้องคัดลอกไฟล์ข้อมูลของตัวเองไปยังเครื่องที่จะเล่น

บทที่ 2
การแก้ปัญหาโดยค้นหา (Searching)

กระบวนการนี้จะมองปัญหาเป็นอินพุทและให้ผลออกมาเป็นคำตอบในรูปแบบของลำดับของการกระทำ โดยทั่วไปแล้วปัญหามีอยู่ 4 ลักษณะ ดังนี้

1. ปัญหาแบบสถานะเดียว (single-state problem) เป็นลักษณะของปัญหาที่มีข้อมูลเพียงพอจนสามารถบอกได้ว่าตอนนี้อยู่ที่สถานะใดและควรกระทำอย่างไรต่อไป
2. ปัญหาแบบหลายสถานะ (multiple-state problem) เป็นลักษณะของปัญหาที่มีข้อมูลบอกเพียงกลุ่มของการกระทำที่สมควรทำเมื่ออยู่ในสถานะใดๆ
3. ปัญหาแบบคอนทิเจนซี่ (contingency problem) เป็นลักษณะของปัญหาที่ทราบว่าตอนนี้อยู่ที่สถานะใดและถ้ากระทำการใดต่อไปแล้ว ผลที่จะเกิดขึ้นเป็นอย่างไร
4. ปัญหาแบบเอ็กซ์โพลเรชั่น (exploration problem) เป็นลักษณะของปัญหาที่ไม่มีข้อมูล ต้องทดลองทำและหาผลที่เกิดขึ้นเอง

2.1 นิยามของปัญหาและคำตอบ

ที่จริงแล้วปัญหาคือกลุ่มของข้อมูลที่ใช้ในการตัดสินใจว่าจะทำอย่างไร โดยทั่วไปแล้วปัญหาหนึ่งๆจะประกอบด้วย

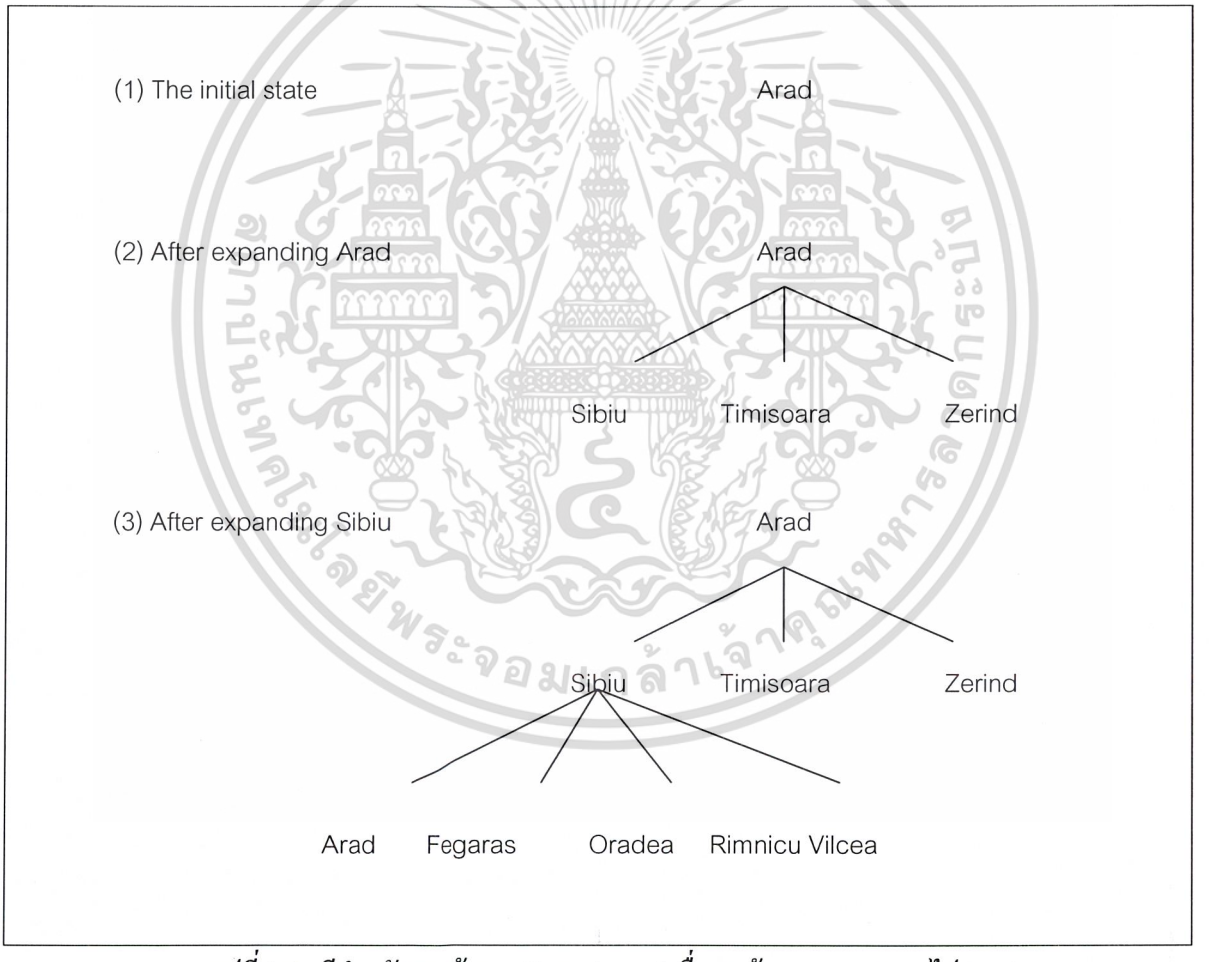
- สถานะเริ่มต้น (initial state) ซึ่งบอกว่าตอนนี้อยู่ที่ไหน
- กลุ่มของการกระทำ (operation) ที่เป็นไปได้
- พื้นที่ของสถานะ (state space) คือกลุ่มของสถานะที่สามารถไปถึงได้จากสถานะเริ่มต้น โดยที่เส้นทาง (path) ในพื้นที่ของสถานะหมายถึง ลำดับของการกระทำที่ทำให้เกิดการย้ายจากสถานะหนึ่งไปยังอีกสถานะหนึ่ง
- การตรวจสอบจุดหมาย (goal test) ที่สามารถนำไปสร้างเป็นสถานะที่นำมาพิจารณาได้ว่าที่สถานะนั้นเป็นสถานะที่เป็นจุดหมาย (goal state) หรือไม่
- ฟังก์ชันคำนวณค่าของเส้นทาง (path cost function) เป็นฟังก์ชันที่ให้ค่าของแต่ละเส้นทาง โดยการรวมค่าของแต่ละการกระทำที่อยู่ในเส้นทางเดียวกัน

จากส่วนประกอบของปัญหาข้างต้นสามารถแสดงในรูปแบบของดาต้าไทป์ (datatype) ได้ดังนี้

datatype Problem **components:** INITIAL_STATE,OPERATION,GOAL_TEST,PATH_COST_FUNCTION

โดยที่ดาต้าไทป์นี้จะให้เป็นอินพุทสำหรับกระบวนการค้นหา (search algorithm) และการค้นหานั้นจะให้คำตอบออกมาเป็นเส้นทางจากสถานะเริ่มต้นไปยังสถานะที่เป็นจุดหมาย

2.2 การค้นหาตำตอบ



รูปที่ 2.1 ทรีสำหรับการค้นหา (Search tree)เพื่อหาเส้นทางจาก Arad ไป Bucharest

วิธีการค้นหา (search strategy) คือการพิจารณาว่าจะเลือกกระจาย (expand) สถานะใดต่อไป วิธีที่มีประโยชน์มากคือนำกระบวนการค้นหามาสร้างเป็นทรีสำหรับการค้นหา โดยที่รูทโหนด (root node) ของทรีสำหรับการค้นหาจะเป็นไปตามสถานะเริ่มต้นและลีฟโหนด (leaf node) จะเป็นสถานะที่ไม่มีสถานะอื่นๆต่อไปอีก ซึ่งมี 2 กรณีคือยังไม่ถูกกระจายหรือถูกกระจายแล้วแต่สร้างสถานะที่ว่างเปล่า ในแต่ละขั้นตอน ขั้นตอนการค้นหาจะเลือกลีฟโหนดขึ้นมา 1 โหนดเพื่อกระจาย รูปที่ 2.1 เป็นตัวอย่างการกระจายบางส่วนของทรีสำหรับการค้นหาสำหรับการหาเส้นทางจากเมืองหนึ่งไปอีกเมืองหนึ่ง

```
function GENERAL_SEARCH(problem,strategy) returns a solution, or failure
        initialize the search tree using the initial state of problem
        loop do
                if there are no candidates for expansion then return failure
                choose a leaf node for expansion according to strategy
                if the node contains a goal state then return the corresponding solution
                else expand the node and add the resulting node to the search tree
end
```

รูปที่ 2.2 ขั้นตอนการค้นหาแบบพื้นฐาน

2.2.1 โครงสร้างข้อมูลของทรีสำหรับการค้นหา

ในที่นี้สมมุติว่าโหนดเป็นโครงสร้างที่ประกอบด้วยส่วนประกอบ 5 ส่วนดังนี้

1. สถานะ (state) ซึ่งอยู่ในพื้นที่ของสถานะที่โหนดนั้นอยู่
2. โหนดที่สร้างโหนดนี้ขึ้นมา เรียกว่าโหนดพ่อ (parent node)
3. กระบวนการที่ใช้ในการสร้างโหนด
4. จำนวนโหนดบนเส้นทางจากรูทโหนดมายังโหนดนี้ เรียกว่าความลึก (depth) ของโหนด
5. ค่าของเส้นทางจากรูทโหนดมาที่โหนดนั้นๆ

ดาต้าไทป์ของแต่ละโหนดมีลักษณะดังนี้ :

```
datatype node
components: STATE,PARENT_NODE,OPERATION,PATH_COST
```

2.3 วิธีการค้นหา

วิธีการประเมินค่ากระบวนการค้นหามี 4 ประการคือ

1. ความสมบูรณ์ (completeness) กระบวนการนั้นรับประกันได้หรือไม่ว่าสามารถหาคำตอบได้หากมีคำตอบอยู่ในทรีสำหรับการค้นหา
2. เวลามากที่สุดที่ใช้ในการหาคำตอบ (time complexity)
3. พื้นที่มากที่สุดที่ใช้ในการคำตอบ (space complexity) คือจำนวนหน่วยความจำที่ใช้สำหรับการค้นหา
4. ในกรณีที่มีคำตอบมากกว่า 1 คำตอบอยู่ในทรีสำหรับการค้นหาแล้วกระบวนการนั้นสามารถหาคำตอบที่ดีที่สุดได้หรือไม่ (optimality)

Criterion	Breadth-first	Uniform-cost	Depth-first	Depth-limited	Iterative Deepening	Bidirection
Time	b^d	b^d	b^d	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	Bl	bd	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complete ?	Yes	Yes	No	Yes, if 1>=1=d	Yes	Yes

ตารางที่ 2.1 แสดงการเปรียบเทียบการค้นหาประเภทยูนิฟอร์มในแบบต่างๆ

2.3.1 ประเภทของการค้นหา

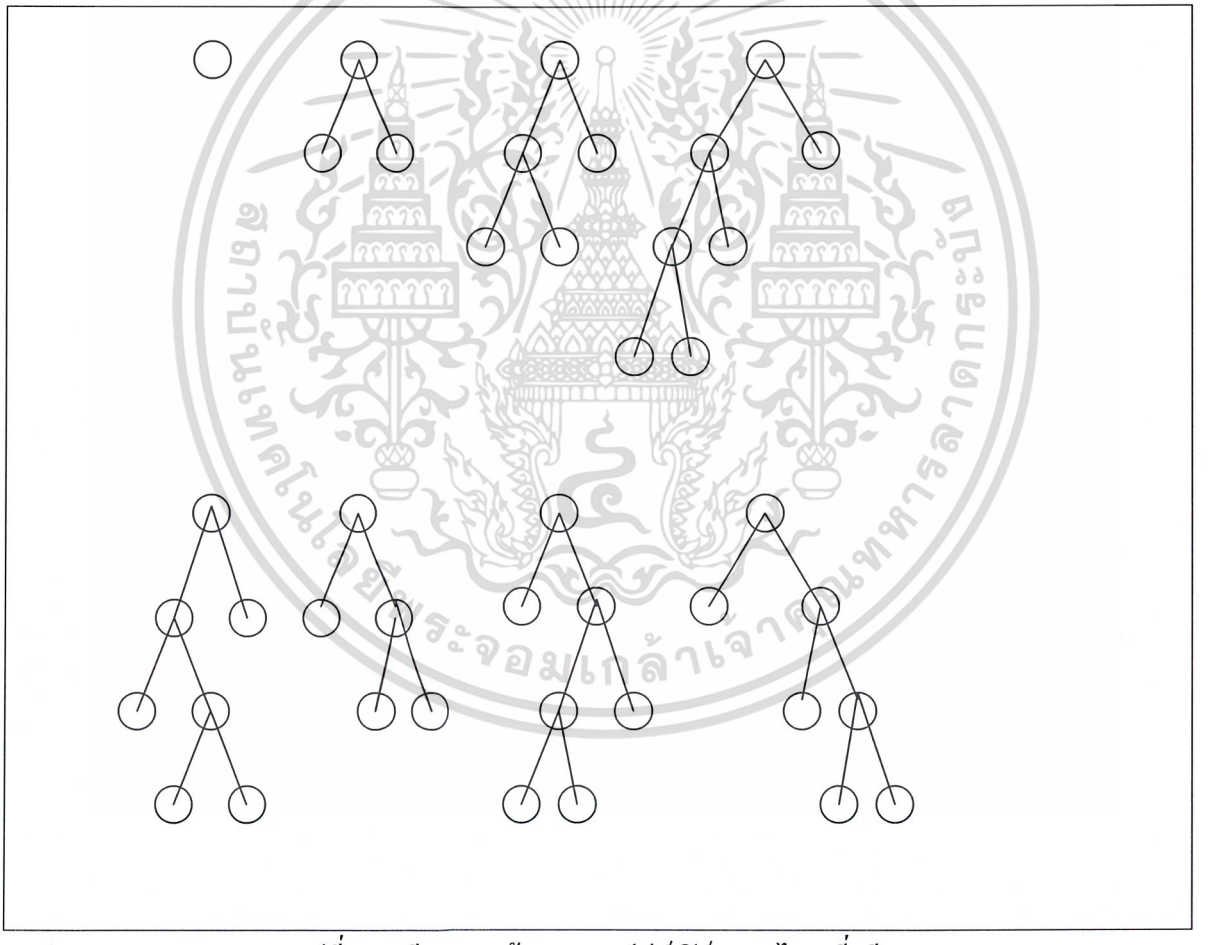
โดยทั่วไปแล้วสามารถจำแนกประเภทของการค้นหาได้ 2 ประเภทใหญ่ๆคือ

1. การค้นหาแบบยูนิฟอร์ม (uninformed search หรือ blind search) คือ การค้นหาที่ไม่มีข้อมูลที่เกี่ยวกับจำนวนขั้นตอนหรือค่าของเส้นทางจากสถานะปัจจุบันไปยังสถานะที่เป็นจุดหมาย สิ่งเดียวที่สามารถทำในสำหรับการค้นหาวิธีนี้คือแยกแยะความแตกต่างระหว่างสถานะที่เป็นจุดมุ่งหมายกับสถานะที่ไม่ใช่จุดมุ่งหมายให้ได้ การค้นหาประเภทนี้มีทั้งสิ้น 6 วิธีดังนี้
   - การค้นหาแบบเบรดท์เฟิร์ท (Breadth-first search)
   - การค้นหาแบบยูนิฟอร์ม (Uniform cost search)
   - การค้นหาแบบเด็ปท์เฟิร์ท (Depth-first search)
   - การค้นหาแบบเด็ปท์ลิมิท (Depth-limited search)
   - การค้นหาแบบไอเทอเรทีฟดีฟเพนนิ่ง (Iterative deepening search)

- การค้นหาแบบไบไดเร็คชั่นแนล (Bidirectional search)

2. การค้นหาแบบอินฟอร์ม (informed search หรือ heuristic search) คือ การค้นหาที่พิจารณาถึงสิ่งต่างๆที่เกี่ยวข้องกับการค้นหาและสิ่งแวดล้อมที่มีอยู่ การค้นหาประเภทนี้มีทั้งสิ้น วิธีดังนี้
   - การค้นหาแบบเบสเฟิร์ท (Best-first search)
   - การค้นหาแบบมินิไมซิ่งค่าพาทคอสหรือเอสตาร์ (Minimizing the total path cost : A* search)

2.4 การค้นหาแบบเด็ปท์เฟิร์ท



รูปที่ 2.3 ทรีของการค้นหาแบบเด็ปท์เฟิร์ทแบบไบนาริ่ทรี (binary tree)

การค้นหาแบบเด็ปท์เฟิร์ทจะกระจายโหนดที่อยู่ระดับที่ลึกที่สุดในทรีจนกระทั่งไม่สามารถกระจายต่อไปได้อีก จึงย้อนกลับมากระจายโหนดที่อยู่ระดับที่ตื้นกว่า การค้นหาวิธีนี้สามารถนำไปประยุกต์เป็นการค้นหาแบบทั่วไป (General search) โดยใช้คิวลิ่งฟังก์ชัน (queuing function) ซึ่งมักจะสร้างสถานะใหม่ด้านหน้าของคิว กระบวนการค้นหาแสดงไว้ดังรูปที่ 2.3

การค้นหาแบบเด็ปท์เฟิร์ทใช้พื้นที่ของหน่วยความจำค่อนข้างน้อย ดังแสดงในรูปที่ 2.3 เพราะจะเก็บเฉพาะเส้นทางเพียงเส้นทางเดียวจากรูทโหนดไปยังลีฟโหนดโดยไม่มีการกระจายโหนดข้างเคียง (sibling node) ของแต่ละโหนดระหว่างเส้นทางนั้นๆเลย

การค้นหาแบบเด็ปท์เฟิร์ทต้องการพื้นที่สำหรับสถานะเพียง bm โหนดสำหรับ tree ที่มีบรานชิ่งแฟ็กเตอร์ b และความลึกสูงสุด m ในขณะที่การค้นหาแบบเบรดท์เฟิร์ทต้องการพื้นที่สำหรับสถานะถึง b^d และเมื่อเทียบกับตารางที่ 2.2 การค้นหาแบบเด็ปท์เฟิร์ทต้องการของหน่วยความจำเพียง 12 ไบท์ในขณะที่การค้นหาแบบเบรดท์เฟิร์ทต้องการถึง 111 เทระไบท์ (terabytes) ที่จำนวน d=12

Depth	Nodes	Times	Memory
0	1	1 millisecond	100 bytes
2	111	.1 seconds	11 kilobytes
4	11111	11 seconds	1 megabytes
5	10^6	18 minutes	111 megabytes
6	10^8	31 hours	11 gigabytes
10	10^{10}	128 days	1 terabytes
12	10^{12}	35 years	111 terabytes
14	10^{14}	3500 years	11111 terabytes

ตารางที่ 2.2 เวลาและจำนวนหน่วยความจำที่ต้องการสำหรับการค้นหาแบบเบรดเฟิร์ทโดยสมมุติว่าบรานชิ่งแฟ็กเตอร์ b = 10 ; 1000 nodes/second ; 100 bytes/node.

เวลามากที่สุดที่ใช้ในการค้นหาสำหรับการค้นหาแบบเด็ปท์เฟิร์ทคือ $O(b^m)$ และสำหรับปัญหาที่มีหลายคำตอบนั้น การค้นหาวิธีนี้จะใช้เวลาน้อยกว่าการค้นหาแบบเบรดท์เฟิร์ทเพราะมีโอกาสที่ดีในการหาคำตอบหลังจากสำรวจเพียงบางส่วนของทั้งหมด

จุดบกพร่องของการค้นหาแบบเด็ปท์เฟิร์ทนั้นจะเกิดขึ้นในกรณีที่ไปค้นหาคำตอบผิดเส้นทาง หลายๆ ปัญหาจะมีทรีที่ยาวมากทำให้เกิดทรีสำหรับการค้นหาที่ไม่สิ้นสุด (infinite search tree) ทำให้ไม่สามารถหาคำตอบที่ต้องการได้ เพราะจะค้นหาคำตอบในแนวดิ่งเพียงอย่างเดียวโดยไม่ย้อนกลับมากระจายโหนดที่อยู่ตื้นกว่า จึง

แสดงให้เห็นว่าการค้นหาด้วยวิธีนี้จะไม่สามารถรับประกันได้ว่าหากมีคำตอบที่ต้องการอยู่ในทรีนั้นแล้วต้องพบคำตอบนั้นๆ และไม่สามารถให้คำตอบที่ดีที่สุดหากมีคำตอบต้องการหลายคำตอบอยู่บนทรี

สามารถประยุกต์การค้นหาแบบเด็ปท์เฟิร์ทกับการค้นหาแบบทั่วไปได้ดังนี้

```
function DEPTH_FIRST_SEARCH(problem) returns a solution, or failure
        GENERAL_SEARCH(problem,ENQUEUE_AT_FRONT)
```

2.5 การค้นหาแบบเด็ปท์ลิมิท

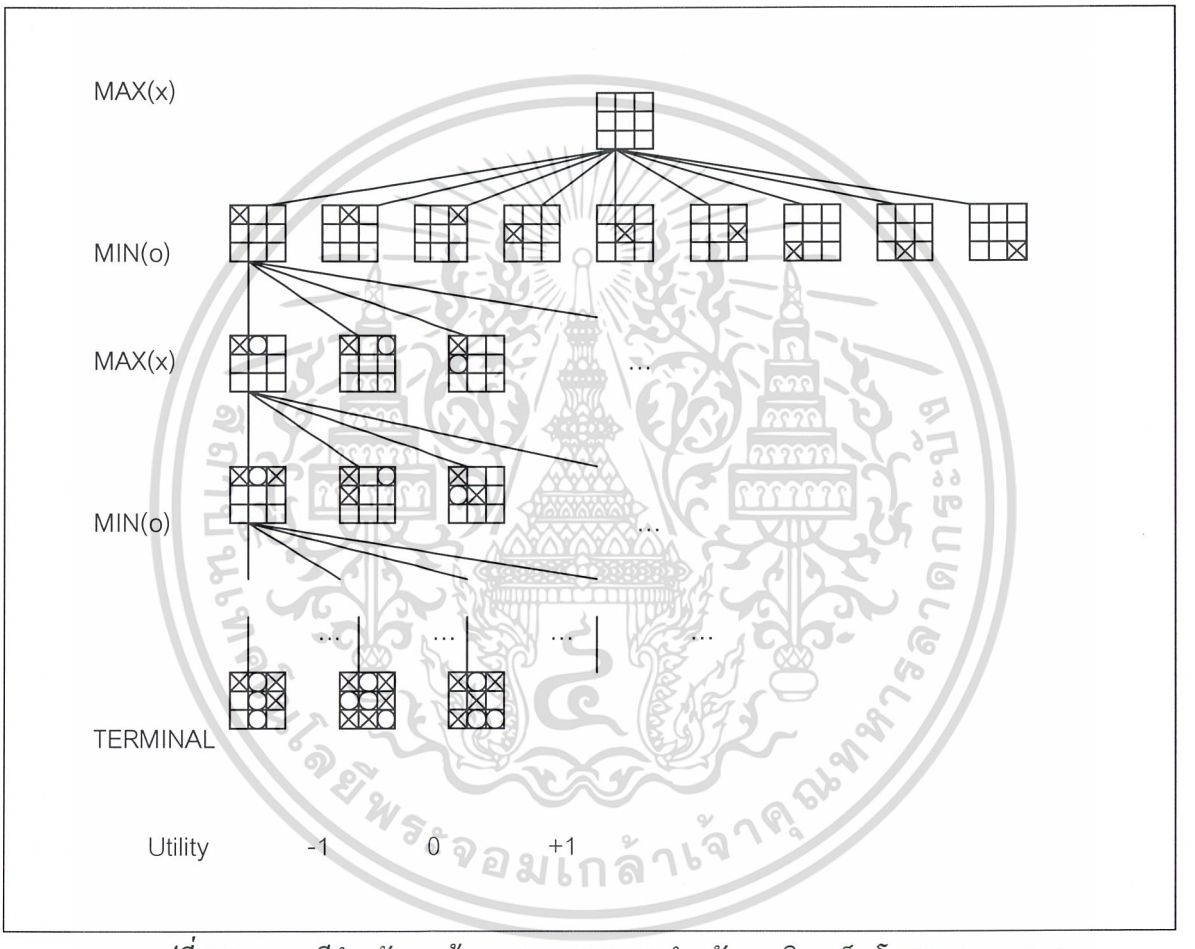
การค้นหาแบบเด็ปท์ลิมิทช่วยแก้ไขข้อบกพร่องของการค้นหาแบบเด็ปท์เฟิร์ท โดยการกำหนดค่าคัทอ็อฟ (Cutoff) สำหรับความลึกสูงสุดที่จะทำการค้นหา ค่าคัทอ็อฟนี้สามารถประยุกต์ใช้กับการค้นหาแบบเด็ปท์เฟิร์ทแบบพิเศษหรือ การค้นหาแบบทั่วไปโดยใช้การเก็บค่าความลึกไว้ สำหรับการค้นหาวิธีนี้รับประกันว่าสามารถหาคำตอบที่มีอยู่ได้ แต่ไม่รับประกันว่าจะได้คำตอบที่อยู่ใกล้ที่สุด ดังนั้นจึงสมบูรณ์แต่ไม่สามารถหาคำตอบที่ดีที่สุดหากมีมากกว่า 1 คำตอบในทรีได้ (ถ้าหากเลือกค่าคัทอ็อฟน้อยเกินไปอาจทำให้ไม่สมบูรณ์ได้เช่นกัน)

- พื้นที่สำหรับสถานะ = bm โหนดสำหรับทรีที่มีบรานชิ่งแฟ็กเตอร์ b และความลึกสูงสุดเท่ากับ m
- เวลามากที่สุดที่ใช้ในการค้นหาเท่ากับ $O(b^m)$
- ไม่สามารถหาคำตอบที่ดีที่สุดหากมีมากกว่า 1 คำตอบในทรีได้แต่สมบูรณ์

บทที่ 3

การเล่นเกม (Game playing)

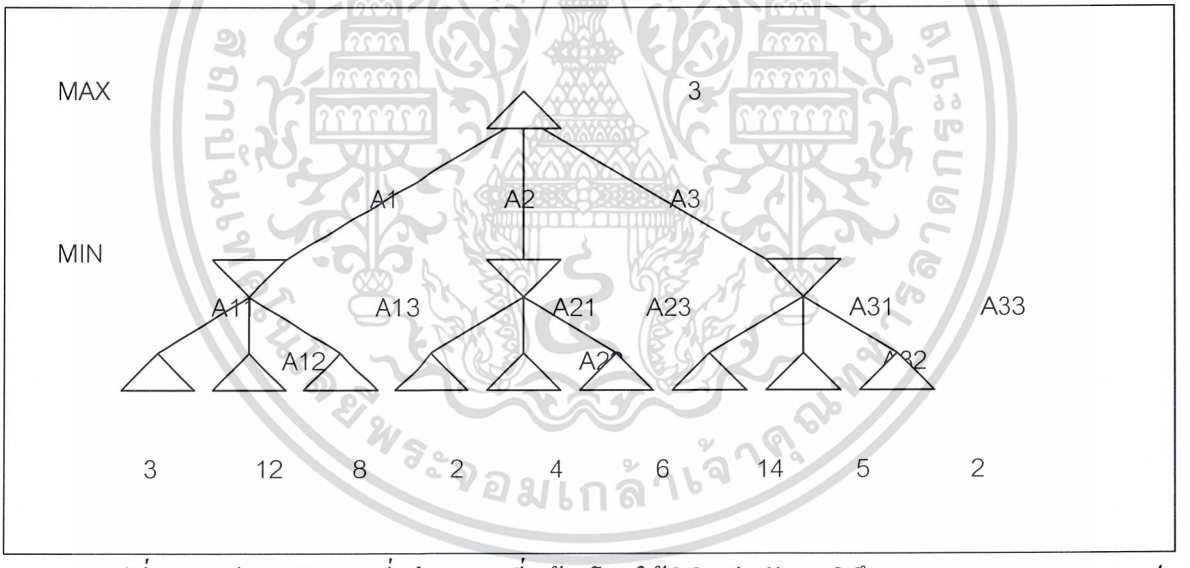
3.1 การตัดสินใจที่สมบูรณ์แบบในการเล่นเกม (Perfect decisions in Two-Person games)



รูปที่ 3.1 แสดงทรีสำหรับการค้นหา (search tree) สำหรับเกมทิค-แท็ก-โท (Tic-Tac-Toe) โหนดบนสุดเป็นสถานะเริ่มต้นฝ่ายแม็ก (MAX) เป็นฝ่ายเล่นก่อนโดยเขียน x ลงในตาราง แล้วเปลี่ยนให้ฝ่านมิน (MIN) เล่นโดยการเขียน o ตาราง ดังรูปบางส่วนของทรีสำหรับการค้นหาที่ให้ฝ่ายมินและฝ่ายแม็กเล่นไปจนถึงสถานะสิ้นสุดเกมซึ่งให้ค่ายูทิลิตี้ (utility) ตามกฎของเกม

เมื่อพิจารณากรณีทั่วไปของเกมสำหรับเล่น 2 คนที่เรียกผู้เล่นว่าฝ่ายแม็กและฝ่ายมิน ฝ่ายแม็กจะเริ่มเล่นก่อน หลังจากนั้นมีการเปลี่ยนกันเล่นจนเกมจบ เมื่อจบเกมแล้วจะใช้คะแนนที่ได้ในการตัดสินว่าจะให้ฝ่ายไหนเป็นฝ่ายชนะ (หรือบางทีใช้ข้อผิดพลาดในการตัดสินว่าฝ่ายไหนเป็นฝ่ายแพ้) จึงอาจกล่าวได้ว่าเกมเป็นปัญหาของการค้นหาชนิดหนึ่งที่ประกอบด้วย

- สถานะเริ่มต้นที่ประกอบด้วยกระดานที่บอกตำแหน่งและเครื่องหมายที่ชี้ได้ว่าเป็นฝ่ายไหน
- กลุ่มของการกระทำ ที่อธิบายถึงการเล่นที่ถูกต้องตามกฎ
- การตรวจสอบการสิ้นสุด (terminal test) เป็นการตรวจสอบว่าเกมจะสิ้นสุดตอนไหนและเรียกสถานะที่เกมจบลงว่าเป็นสถานะสิ้นสุดเกม
- ยูทิลิตี้ฟังก์ชันหรือเพย์ออฟฟังก์ชัน (utility function or payoff function) เป็นค่าทางตัวเลขที่แสดงถึงผลของการเล่นเกม ตัวอย่างเช่นการเล่นหมากรุกผลจากการเล่นเกมมี 3 กรณีคือ ชนะ, เสมอ หรือแพ้ ซึ่งสามารถแสดงได้ด้วยค่า +1, 0 หรือ –1 หรือสำหรับเกมที่มีผลที่เป็นไปได้หลายค่าอย่างเช่นเกมแบ็กแกมม่อน (backgammon) นั้นจะมีค่ายูทิลิตี้ฟังก์ชันในช่วง +192 ถึง –192 เป็นต้น



รูปที่ 3.2 ทรีสำหรับเกมที่เล่น 2 คนที่สร้างโดยใช้มินิแม็กอัลกอริทึม (Minimax algorithm) รูปสามเหลี่ยมหงายจะเลื่อนโดยฝ่ายแม็กและสามเหลี่ยมคว่ำจะเลื่อนโดยฝ่ายมิน โหนดสุดท้ายแสดงค่ายูทิลิตี้สำหรับฝ่ายแม็กที่คำนวณโดยใช้ยูทิลิตี้ฟังก์ชัน (Utility function)

ถ้ามีปัญหาที่ต้องการการค้นหาเกิดขึ้นแล้ว สิ่งที่ฝ่ายแม็กควรจะทำคือค้นหาลำดับของการเล่นที่สามารถไปถึงสถานะสิ้นสุดเกมของผู้ชนะ (ตามค่ายูทิลิตี้ของเกมนั้นๆ) หลังจากนั้นจึงทำตามลำดับการกระทำแรกตามที่ค้นหามาได้ แต่ในการเล่นเกมจริงๆแล้วฝ่ายแม็กจะต้องพิจารณาลำดับการเล่นของตัวเองโดยพิจารณาลำดับการเล่นที่เป็น

ไปได้ของฝ่ายมินด้วย ดังรูปที่ 3.1 แสดงทรีสำหรับการค้นหาของเกมทิค-แท็ก-โท จากสถานะเริ่มต้นฝ่ายแม็กมีทางเลือกที่เป็นไปได้ 9 ทางเลือก โดยฝ่ายแม็กจะเขียน x ลงในกระดานขณะที่ฝ่ายมินเขียน o ลงในการดานสลับกันจนกว่าจะไปถึงลีฟโหนดซึ่งเป็นสถานะสิ้นสุดเกม ซึ่งเป็นสถานะที่ฝ่ายใดฝ่ายหนึ่งสามารถเขียนสัญลักษณ์ของฝ่ายตัวเองได้ 3 แถวเป็นแนวตรงหรือแนวทแยง ตัวเลขที่ปรากฏที่แต่ละลีฟโหนดแสดงค่ายูทิลิตี้ของทุกๆสถานะสิ้นสุดเกมในมุมมองของฝ่ายแม็กสำหรับในที่นี้สมมุติว่าค่ายูทิลิตี้มากเป็นผลดีต่อฝ่ายแม็กหน้าที่ของฝ่ายแม็กคือใช้ทรีสำหรับการค้นหาเพื่อพิจารณาลำดับการเล่นที่ดีที่สุด

รูปที่ 3.2 แสดงการเล่นเกม 3 รอบ ฝ่ายแม็กสามารถเลือกที่จะเล่นได้ 3 วิธีคือไปทาง A1, A2 และ A3 ทางที่ฝ่ายมินสามารถเล่นได้เมื่อฝ่ายแม็กเลือกเล่นทาง A1 คือ A11, A12 และ A13 ค่ายูทิลิตี้ที่สถานะสิ้นสุดเกมมีค่าอยู่ในช่วง 2 ถึง 14

3.1.1 มินิแม็กอัลกอริทึม

```
Function MINIMAX_DECISION(game) returns an operation
        for each op in OPERATORS[game] do
                VALUE[op] – MINIMAX_VALUE(APPLY(op,game),game)
        End
        Return the op with the highest VALUE[op]
--------------------------------------------------------------------------------
Function MINIMAX_VALUE(state,game) returns a utility value
        If TERMINAL_STATE[game](state) then
                Return UTILITY[game](state)
        Else if MAX is to move in the state then
                Return the highest MIMIMAX_VALUE of SUCCESSORS(state)

Else
                Return the lowest MINIMAX_VALUE of SUCCESSORS(state)
```

รูปที่ 3.3 แสดงขั้นตอนการคำนวณการตัดสินใจแบบมินิแม็ก

มินิแม็กอัลกอริทึมถูกออกแบบขึ้นมาเพื่อพิจารณาหาวิธีการที่ดีที่สุดสำหรับฝ่ายแม็กและตัดสินใจได้ว่าควรจะเริ่มเล่นอย่างไร ขั้นตอนนี้มี 5 ขั้นตอนดังนี้

1. สร้างทรีสำหรับเกมนั้นๆทั้งหมด โดยที่ทุกๆเส้นทางต้องสิ้นสุดที่สถานะสิ้นสุดเกม
2. ใช้ยูทิลิตี้ฟังก์ชันเพื่อหาค่ายูทิลิตี้สำหรับแต่ละสถานะสิ้นสุดเกม
3. ใช้ค่ายูทิลิตี้ของโหนดสุดท้ายเพื่อพิจารณาค่ายูทิลิตี้ของโหนดที่อยู่ระดับชั้นที่สูงขึ้นไปในทรีสำหรับการค้นหาพิจารณาโหนดที่อยู่ทางซ้ายสุดด้านล่างในรูปที่ 2 ที่โหนด▽ เหนือขึ้นไปฝ่ายมินมีทางเลือกที่จะเล่นที่ดีที่สุดคือเลือกเล่นทาง A11 ซึ่งจะทำให้ได้ค่าที่น้อยที่สุดคือ 3 ดังนั้นยูทิลิตี้ฟังก์ชัน จึงไม่เหมาะสมที่จะใช้ที่โหนด ▽นี้เราจึงให้ค่ายูทิลิตี้ที่โหนดนี้มีค่าเท่ากับ 3 เช่นเดียวกันกับที่โหนดที่สองที่มีค่ายูทิลิตี้เท่ากับ 2
4. พิจารณาค่ายูทิลิตี้ย้อนกลับจากลีฟโหนดไปยังรูทโหนด
5. เมื่อหาค่ายูทิลิตี้ของแต่ละโหนดได้แล้ว ที่จุดนี้ ฝ่ายแม็กจะเลือกเส้นทางการเล่นที่ทำให้มีค่ามากที่สุดที่โหนด △บนสุดในรูปที่ 2 ฝ่ายแม็กเลือกเส้นทางการเล่นที่จะทำให้ได้ค่า 3, 2 และ 2 ตามลำดับ ดังนั้นทางเลือกที่ดีที่สุดสำหรับฝ่ายแม็กคือ A1 เรียกสิ่งนี้ว่าการตัดสินใจแบบมินิแม็ก (minimax decision)

ถ้าความลึกสูงสุด (maximum depth) ของทรีคือ m และมีบรานชิ่งแฟ็กเตอร์ (branching factor) b แล้วเวลามากที่สุดที่ใช้ (time complexity) สำหรับมินิแม็กอัลกอริทึมจะมีค่าเท่ากับ $O(b^m)$ ในส่วนของพื้นที่ที่ต้องการจะมีค่าใกล้เคียงกับ m และ b

3.2 การตัดสินใจที่ไม่สมบูรณ์ (Imperfect decisions)

มินิแม็กอัลกอริทึมอยู่บนสมมุติฐานที่ว่าโปรแกรมมีเวลามากพอที่จะค้นหาทุกเส้นทางที่จะไปยังสถานะสิ้นสุดเกม ซึ่งเป็นไปไม่ได้ในทางปฏิบัติ ในบทความดั้งเดิมของชานนอน (Shannon's original paper) กล่าวไว้ว่าแทนที่จะไปยังทุกเส้นทางที่ไปถึงสถานะสิ้นสุดได้โดยใช้ยูทิลิตี้ฟังก์ชันโปรแกรมควรตัดการค้นหาออกให้เร็วขึ้นแล้วประยุกต์ใช้อิวาลูเอทฟังก์ชัน (evaluate function) ช่วยแก้ปัญหาที่ส่วนลีฟของทรี หรือกล่าวอีกนัยหนึ่งได้ว่าเอกสารดังกล่าวแนะนำว่าให้เปลี่ยนแปลงมินิแม็กอัลกอริทึม 2 ส่วนคือ ใช้อีวาลูเอทฟังก์ชัน EVAL แทนยูทิลิตี้ฟังก์ชันและใช้การตรวจสอบค่าคัทอ็อฟ (cutoff test) CUT_OFF แทนการตรวจสอบการสิ้นสุด

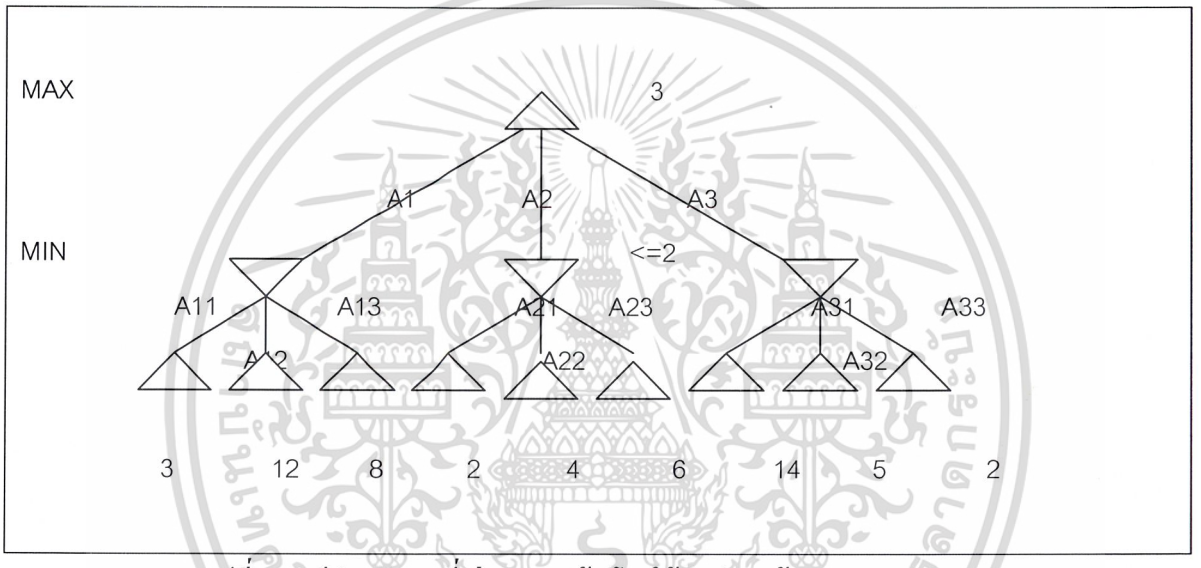
3.2.1 อิวาลูเอทฟังก์ชัน

ประสิทธิภาพของโปรแกรมสำหรับเล่นเกมนั้นขึ้นอยู่กับคุณภาพของอีวาลูเอทฟังก์ชันของโปรแกรมนั้นๆ ถ้าอีวาลูเอทฟังก์ชันของโปรแกรมขาดความละเอียดและถูกต้อง แล้วอาจทำให้โปรแกรมทำในสิ่งที่อาจก่อให้เกิดความเสียหายได้ สิ่งที่ใช้วัดคุณภาพของอีวาลูเอทฟังก์ชันมีดังนี้

- อิวาลูเอทฟังก์ชันต้องสอดคล้องกับยูทิลิตี้ฟังก์ชันที่สถานะสิ้นสุดเกม

- อิวาลูเอทฟังก์ชัน ต้องไม่ใช้เวลานานเกินไป
- อิวาลูเอทฟังก์ชัน ต้องถูกต้องและแสดงให้เป็นถึงการชนะได้

ข้อแนะนำเพิ่มเติมคือควรกำหนดอิวาลูเอทฟังก์ชันโดยใช้กฎความน่าจะเป็นดังนี้ ถ้าที่ตำแหน่ง A มีโอกาสที่จะชนะ 100% แล้วที่ตำแหน่งนั้นควรมีค่าอิวาลูเอทฟังก์ชันเท่ากับ 1.00 และ ถ้าที่ตำแหน่ง B มีโอกาสที่จะชนะ 50%, โอกาสที่จะแพ้ 25% และโอกาสเสมอ 25% แล้วที่ตำแหน่งนั้นควรมีค่าอีวาลูเอทฟังก์ชันเท่ากับ (+1 * .50) + (-1 * 0.25) + (0 * 0.25) = 0.25 แต่ในความเป็นจริงแล้วไม่จำเป็นที่ต้องทำเช่นนั้น เพราะค่าทางตัวเลขที่แน่นอนนั้นไม่สำคัญตราบใดที่ A มีค่ามากกว่า B



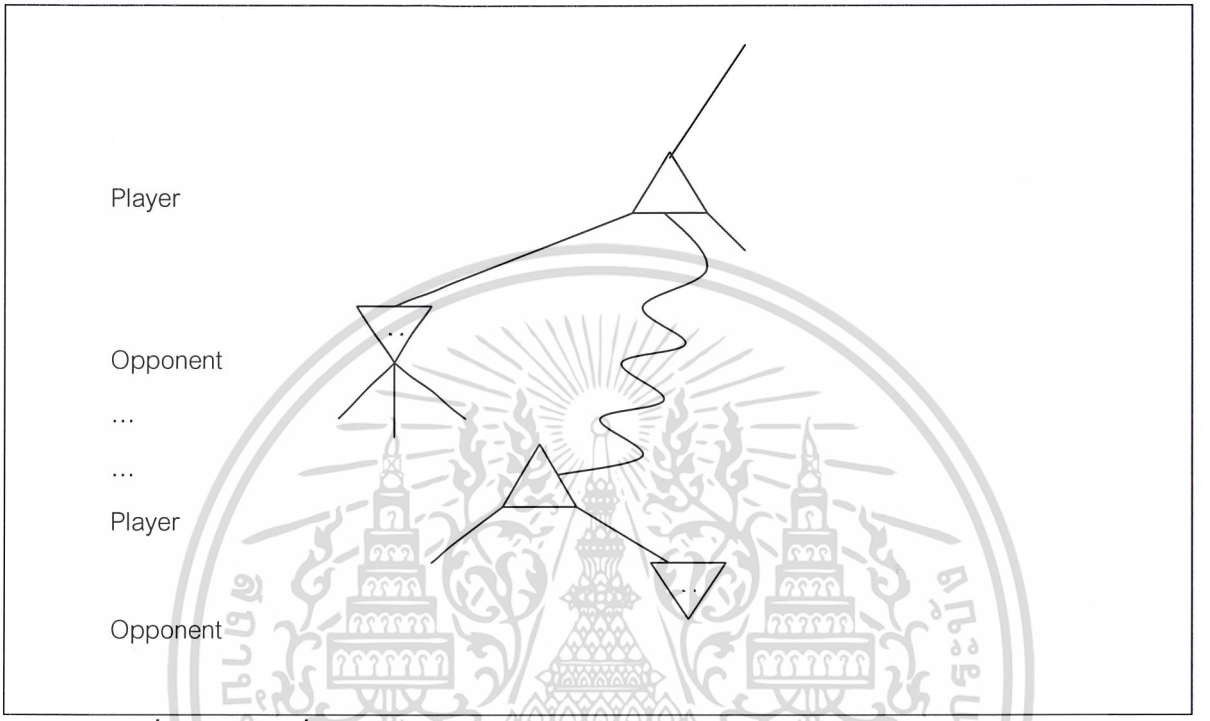
รูปที่ 3.4 ทรีสำหรับเกมที่เล่น 2 คนสร้างโดยใช้อัลฟา-เบต้า (Alpha-Beta)

3.2.2 การพรุนแบบอัลฟา-เบต้า (Alpha-Beta Pruning)

การพรุน (Pruning) คือกระบวนการกำจัดกิ่ง (branch) ของทรีสำหรับการค้นหาโดยการพิจารณาที่ปราศจากการตรวจสอบ เทคนิคที่เป็นพื้นฐานสำหรับการพรุนเรียกว่า "การพรุนแบบอัลฟา-เบต้า" เมื่อนำไปประยุกต์ใช้กับมินิแม็กทรี (minimax tree) ก็จะได้ผลเหมือนเดิมทุกประการเพียงแต่ว่าจะมีการตัดทอนส่วนของกิ่งของทรีที่ไม่มีอิทธิพลต่อการตัดสินใจขั้นสุดท้ายออกไป

พิจารณาทรีของเกมที่เล่น 2 คนจากรูปที่ 3.2 และรูปที่ 3.4 การค้นหาเริ่มต้นก่อนที่ A1 หลังจากนั้น A11, A12, A13 และ โหนดที่อยู่ใต้ A1 ได้ค่ามินิแม็ก = 3 หลังจากนั้นไปดูที่โหนด A2 และ A12 ซึ่งมีค่าเท่ากับ 2 จากจุดนี้จะเห็นว่า ถ้าฝ่ายแม็กเล่นทาง A2 แล้วฝ่ายมินจะมีทางเลือกให้ไปยังตำแหน่งที่มีค่าที่ดีที่สุดคือ 2 และทางเลือกอื่นๆอีก ดังนั้นสามารถกล่าวได้ว่าเอาทาง A2 ออกจะมีค่าที่ดีที่สุดสำหรับฝ่ายแม็กคือ 2 เนื่องจากเราสามารถรู้ได้ทันทีว่าถ้าเอาทาง A1 ออกจะมีค่าที่ดีที่สุดคือ 3 จึงไม่มีความจำเป็นที่จะต้องมองถึงรายละเอียดในระดับชั้นที่อยู่ใต้

A2 หรือกล่าวได้อีกอย่างหนึ่งคือ เราสามารถพรุนทรีได้ที่ตำแหน่งนี้และมั่นใจได้ว่าส่วนที่พรุนออกไปจะไม่มีผลกระทบต่อผลลัพธ์ที่ได้



รูปที่ 3.5 รูปแบบทั่วไปของการพรุนแลลอัลฟา-เบต้า ถ้า m ดีกว่า n แล้วในการเล่นจะไม่มีทางไปถึง n

หลักเกณฑ์โดยทั่วไปคือ พิจารณาโหนด n ใดๆในทรี (ดูในรูปที่ 3.5) อย่างเช่น โหนดที่ผู้เล่นมีทางเลือกที่จะเดินไปที่โหนดนั้นๆ ถ้าผู้เล่นมีทางเลือก m หรือที่โหนดพ่อ (parent node) ของ n หรือทางเลือกอื่นๆในระดับชั้นต้นๆที่ดีกว่า n แล้ว ในทางปฏิบัติผู้เล่นจะไม่มีทางไปถึงโหนด n ได้ ดังนั้นจึงสามารถพรุนทิ้งได้

มินิแม็กเป็นการค้นหาแบบเด็ปท์เฟิร์ส ดังนั้น ณ เวลาใดๆจะพิจารณาเฉพาะโหนดที่อยู่ในเส้นทางเดียวในทรีให้ α เป็นทางเลือกที่ดีที่สุดที่เจอในบรรดาหลายๆทางเลือกบนเส้นทางของฝ่ายแม็กและβ เป็นค่าที่ดีที่สุด (ค่าที่ต่ำที่สุด) ที่เจอในบรรดาหลายๆทางเลือกบนเส้นทางของฝ่ายมินแล้วการค้นหาแบบอัลฟา-เบต้า (Alpha-Beta search) จะแก้ไขข้อมูลให้ใหม่อยู่เสมอเมื่อพรุนส่วนย่อยของทรีและพบว่าค่า α และ β แย่กว่าค่าปัจจุบัน

ขั้นตอนการทำงานอธิบายไว้ในรูปที่ 3.6 แบ่งออกเป็นฟังก์ชัน MAX_VALUE และ MIN_VALUE สำหรับประยุกต์ใช้กับโหนดของแม็กและโหนดของมินตามลำดับแต่ทั้งสองฟังก์ชันให้ผลเหมือนกันคือเป็นค่า

```
Function MAX_VALUE(state,game,α,β) returns the minimax value of state
 inputs:        state, current state in game
        game, game description
        α, the best score for MAX along the path to state
        β, the best score for MIN along the path to state
 if CUTOFF_TEST(state) then returns EVAL(state)
            for each s in SUCCESSORS(state) do
        α -- MAX(α,MIN_VALUE(s,game,α,β))
        if α >= β then return β
 end
 return α
        ---------------------------------------------------------------------
Function MIN_VALUE(state,game,α,β) returns the minimax value of state
 if CUTOFF_TEST(state) then returns EVAL(state)
 for each s in SUCCESSORS(state) do
        β -- MIN(β,MIN_VALUE(s,game,α,β))
        if β <= α then return α
 end
 return β
```

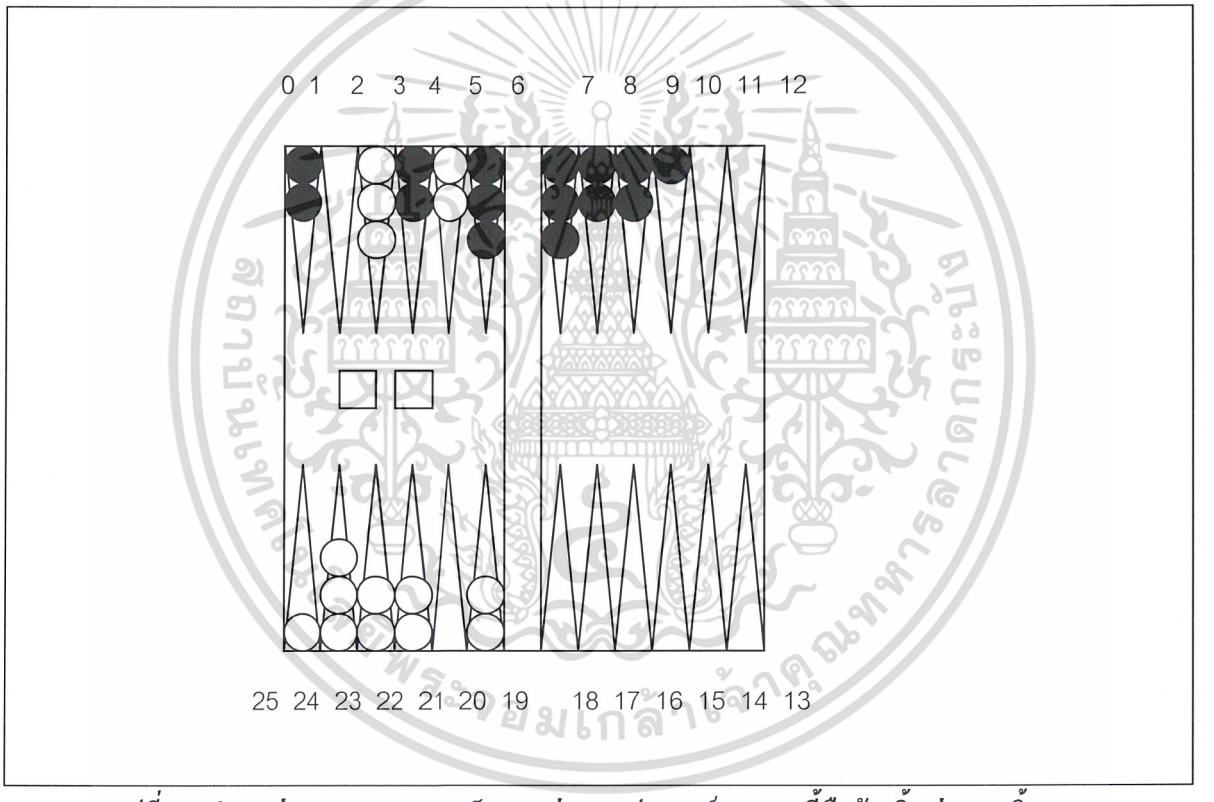
รูปที่ 3.6 แสดงกระบวนการค้นหาแบบอัลฟา-เบต้า (alpha-beta search algorithm)
มีลักษณะเหมือนกับการคำนวณมินิแม็กทั่วไปแต่มีการพรุน

3.3 เกมที่ประกอบด้วยปัจจัยโอกาส (Games that include an element of chance)

ในชีวิตจริงมักจะมีเหตุการณ์ที่ไม่คาดคิดเกิดขึ้นเสมอๆ ซึ่งเหตุการณ์เหล่านั้นอาจนำเราไปสู่สถานการณ์ที่คาดไม่ถึง หลายๆเกมสะท้อนสิ่งนี้โดยใช้ปัจจัยสุ่มอย่างเช่น การทอดลูกเต๋า

เกมแบ็กแกมม่อนเป็นตัวอย่างเกมง่ายๆที่ผสมผสานระหว่างความฉนัดและโชคชะตา ในตอนเริ่มต้นแต่ละเทิร์นของผู้เล่นแต่ละคนจะต้องทอดลูกเต๋าเพื่อกำหนดว่าจะเคลื่อนที่ได้ทั้งหมดกี่ครั้ง ตามตำแหน่งในรูปที่ 3.7 ฝ่ายสีขาวทอดลูกเต๋าได้ 6-5 จึงมี 4 ทางเลือกคือ (5-10,5-11), (5-11,19-24), (5-10,10-16) และ (5-11,11-16)

ถึงแม้ว่าฝ่ายสีขาวจะทราบว่าจะเลือกเล่นอย่างไรแต่ก็ไม่ทราบว่าฝ่ายสีดำจะทอดลูกเต๋าได้เท่าไร ดังนั้นจึงไม่มีทางทราบได้ว่าฝ่ายสีดำจะเล่นอย่างไร ซึ่งหมายความว่าฝ่ายสีขาวไม่มีทางสร้างทรีที่สมบูรณ์อย่างเช่นที่เห็นในเกมทิค-เท็ก-โท สำหรับเกมนี้ได้ ดังนั้นทรีสำหรับเกมเบ็กแกมม่อนจึงต้องเพิ่มเติมโหนดโอกาส (chance nodes) ให้กับทั้งฝ่ายแม็กและฝ่ายมินด้วย ดังแสดงในรูปที่ 3.8 แต่ละกิ่งที่ออกมาจากแต่ละโหนดโอกาสแสดงถึงความเป็นไปได้ของการทอดลูกเต๋าและมีเครื่องหมายบอกหน้าของลูกเต๋าที่ทอดได้และโอกาศที่อาจเกิดขึ้นด้วย สำหรับการทอดลูกเต๋าโดยทั่งไปแล้วจะมีโอกาสที่เป็นไปได้ทั้งหมด 36 วิธี (แต่เนื่องจากผลการทอดลูกเต๋า 6-5 และ 5-6 ถือว่าเหมือนกัน ดังนั้นจึงมีวิธีที่ไม่ซ้ำจำนวน 21 วิธี) การทอดลูกเต๋าที่ออกมาหน้าเหมือนกันทั้งสองลูกมี 6 วิธีคือ 1-1,2-2,3-3,...,6-6 ลักษณะนี้มีโอกาสเกิดขึ้นเท่ากับ 1/36 ส่วนแบบอื่นๆมีโอกาสเกิดขึ้นเท่ากับ 1/18



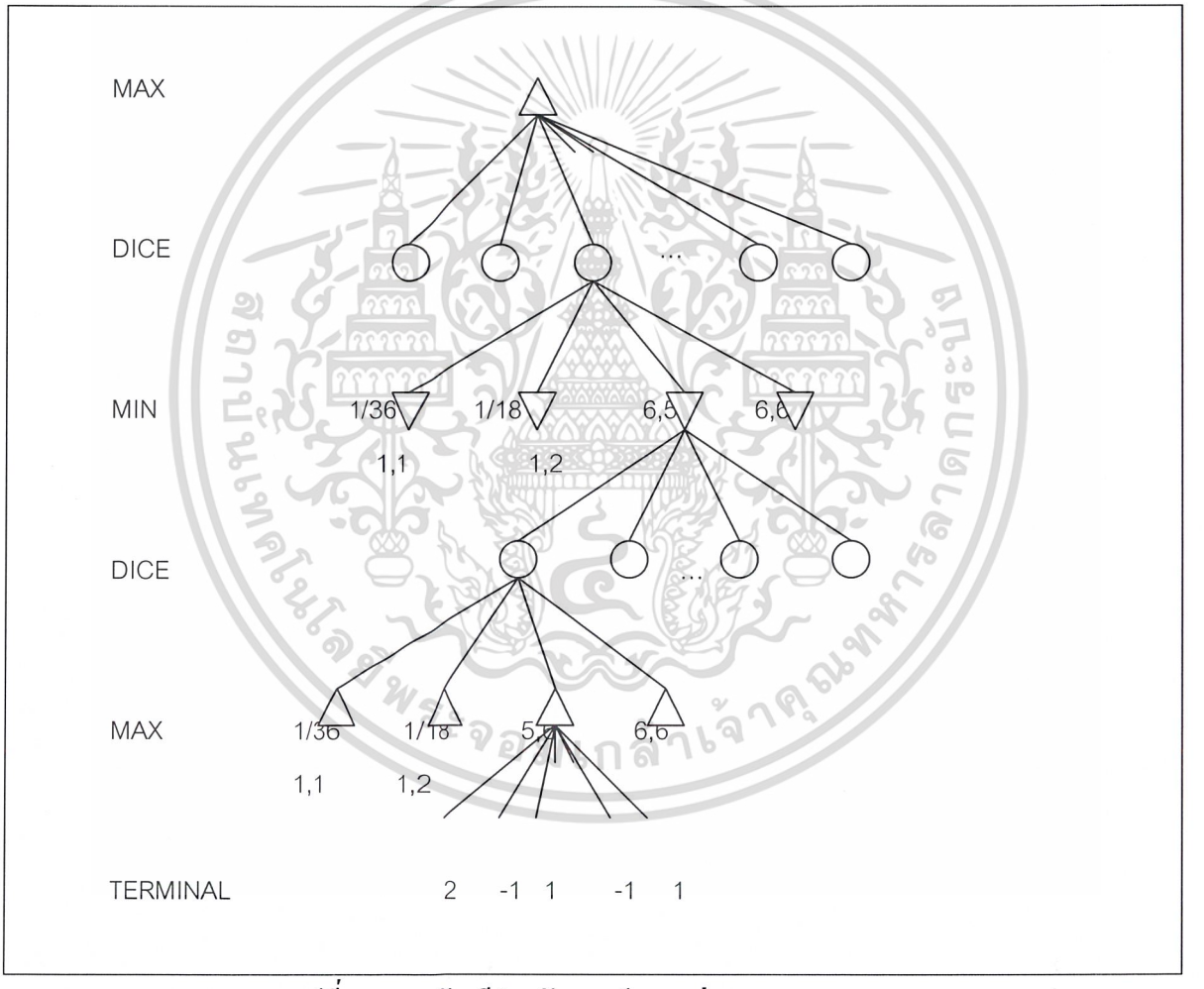
รูปที่ 3.7 ตำแหน่งการวางของเกมเบ็กแกมม่อนจุดประสงค์ของเกมนี้คือย้ายชิ้นส่วนทุกชิ้นของตัวเองออกมานอกกระดาน ฝ่ายสีขาวจะหมุนตามเข็มนาฬิกาไปยังหมายเลข 25 ส่วนฝ่ายสีดำจะหลุนทวนเข็มนาฬิกาไปยังเลข 0 แต่ละชิ้นส่วนสามารถย้ายไปที่ใดก็ได้ยกเว้นตำแหน่งที่มีชิ้นส่วนของฝ่ายตรงข้ามอยู่มากกว่า 2 ชิ้น

ขั้นตอนต่อไปคือการทำความเข้าใจถึงวิธีการตัดสินใจที่ถูกต้อง ที่จริงแล้วเราสามารถเพียงแค่คำนวณค่าโดยประมาณ (expected value) ซึ่งเป็นค่าประมาณของเหตุการณ์ที่เป็นไปได้ทั้งหมดในการทอดลูกเต๋า สำหรับโหนดสิ้นสุด (terminal node) นั้นจะใช้ยูทิลิตี้ฟังก์ชัน

ขั้นตอนถัดไปเมื่อเจอโหนดโอกาสในรูปที่ 3.8 โหนดโอกาสเป็นรูปวงกลม พิจารณาที่โหนดที่มีสัญลักษณ์ C โดยให้

d_i เป็นความน่าจะเป็นของหน้าลูกเต๋า

$P(d_i)$ เป็นโอกาสหรือความน่าจะเป็นของการทอดลูกเต๋า



รูปที่ 3.8 แผนผังทรีสำหรับเกมเบ็กแกมม่อน

สำหรับการทอดลูกเต๋าแต่ละครั้ง เราสามารถคำนวณยูทิลิตี้ของการเล่นที่ดีที่สุดของฝ่ายมินและทำการเพิ่มน้ำหนักค่ายูทิลิตี้ด้วยโอกาสของการทอดลูกเต๋า ถ้าให้ $S(C,d_i)$ แสดงถึงกลุ่มของตำแหน่งที่สร้างโดยการ

ประยุกต์การเล่นสำหรับการทอดลูกเต๋า $P(d_i)$ ที่ตำแหน่ง C แล้วสามารถคำนวณค่าประมาณที่มากที่สุด (expectimax value) ของ C ได้โดยใช้สูตรนี้

$$\text{expectimax}(C) = \sum P(d_i) \max_{s \varepsilon S(C,di)}(\text{Utility}(s))$$

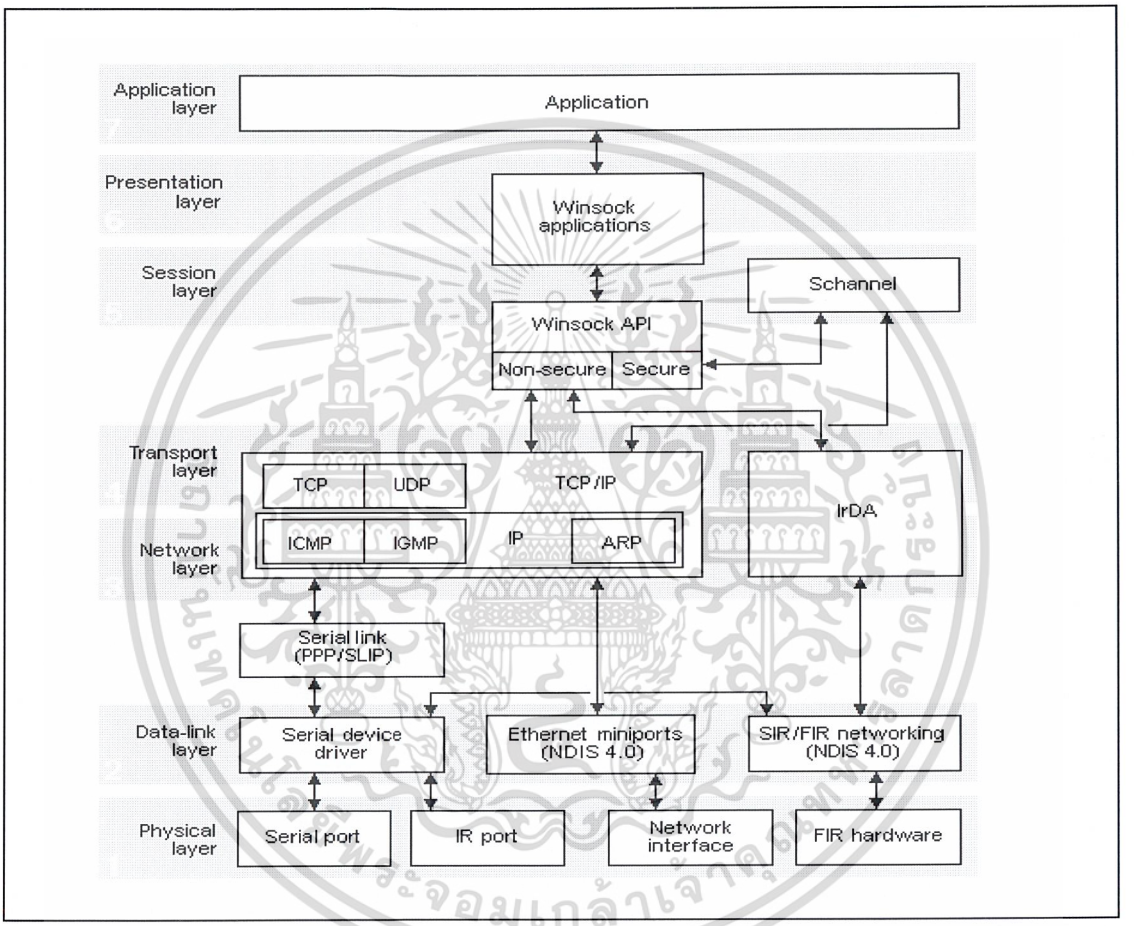
ค่าที่ได้มานี้เป็นค่ายูทิลิตี้ที่ประมาณ (expected utility) ของตำแหน่ง C

ลำดับต่อไปเมื่อไปถึงระดับโหนด MIN ($\bigtriangledown$ในรูปที่ 3) สามารถประยุกต์ใช้สูตรค่ามินิแม็ก (minimax-value) ได้เพราะได้มีการใส่ค่ายูทิลิตี้ให้กับโหนดโอกาสทั้งหมดแล้ว หลังจากนั้นเลื่อนไปที่โหนดโอกาส B เราสามารถคำนวณค่าประมาณที่น้อยที่สุด (expectimin value) โดยใช้สูตรที่คล้ายคลึงกับสูตรของการคำนวณค่าประมาณที่มากที่สุดที่ผ่านมา

บทที่ 4

วินซ็อค (Winsock)

4.1 วินซ็อคและโอเอสไอโมเดล (OSI Model)



รูปที่ 4.1 แสดงความสัมพันธ์ระหว่างวินซ็อคกับโปรโตคอลสำหรับการสื่อสารอื่นๆ

ในไอเอสโอ/โอเอสไอโมเดล (ISO/OSI Model) นั้นวินซ็อคจะทำงานที่อินเตอร์เฟสระหว่างชั้นเซสชั่น (Session Layer) กับชั้นทรานสปอร์ต (Transport Layer) โดยวินซ็อคทำหน้าที่เป็นตัวกลางระหว่างแอพพลิเคชั่นกับโปรโตคอลสำหรับการส่งข้อมูลและเป็นทางส่งข้อมูลอีกด้วย รูปที่ 4.1 แสดงความสัมพันธ์ระหว่างวินซ็อคกับโปรโตคอลวินโดว์ซีอีสำหรับการติดต่อสื่อสาร (Window CE communication protocols) อื่นๆ ในขอบเขตของไอเอสโอ/โอเอสไอโมเดล

โดยวินซ็อคจะทำให้แอพพลิเคชั่นที่พัฒนาในไอเอสโอ/โอเอสไอชั้นบนมีความง่ายขึ้นโดยจะเก็บเฉพาะรายละเอียดข้อมูลของเครือข่ายที่จะแลกเปลี่ยนกับชั้นที่ต่ำลงมากเท่านั้น นอกจากนี้วินซ็อคยังได้เพื่อความสามารถให้โปรแกรมเมอร์สามารถเขียนโปรแกรมการติดต่อระหว่างชั้นบน (5-7) กับชั้นล่าง (1-4) ได้อีกด้วย ข้อมูลของวินซ็อคแอพพลิเคชั่นจะถูกทำเป็นแพ็กเก็ตและส่งผ่านเครือข่ายโดยชั้นทรานสปอร์ต (transport layer), ดาต้าลิ้งค์ (data-link layer) และฟิสิกคอล (physical layer)

4.2 การเขียนโปรแกรมการเชื่อมต่อผ่านเครือข่ายด้วยวินซ็อค

ในปัจจุบันการสื่อสารผ่านระบบเครือข่าย (Network) รวมทั้งระบบอินเตอร์เน็ต (Internet) เป็นที่แพร่หลายมากในแง่ของการใช้งาน, ใช้บริการและอำนวยความสะดวกในชีวิตประจำวันต่างๆเนื่องจากระบบสื่อสารที่รวดเร็วและสามารถเชื่อมโยง เข้ากับหลายๆระบบได้ ทำให้สามารถนำมาประยุกต์ใช้งานผ่านระบบเครือ ข่ายได้หลากหลายรูปแบบ ซึ่งในบทความนี้จะพยายามเน้นไปประยุกต์ใช้ ในการควบคุมอุปกรณ์หรือโปรแกรมต่างๆ เรื่องที่ต้องรู้ในการที่จะเขียนโปรแกรมผ่านระบบเครือข่ายมีดังนี้

4.2.1 ทีซีพี/ไอพี (TCP/IP)

ระบบเครือข่ายที่ใช้โปรโตคอลมาตรฐานชื่อทีซีพี/ไอพีในการสื่อสารผ่านระบบเพื่อติดต่อกับเครื่องคอมพิวเตอร์อื่นๆโปรโตคอลทีซีพี/ไอพีนั้นประกอบด้วยส่วนที่สำคัญ 2 ส่วนก็คือ

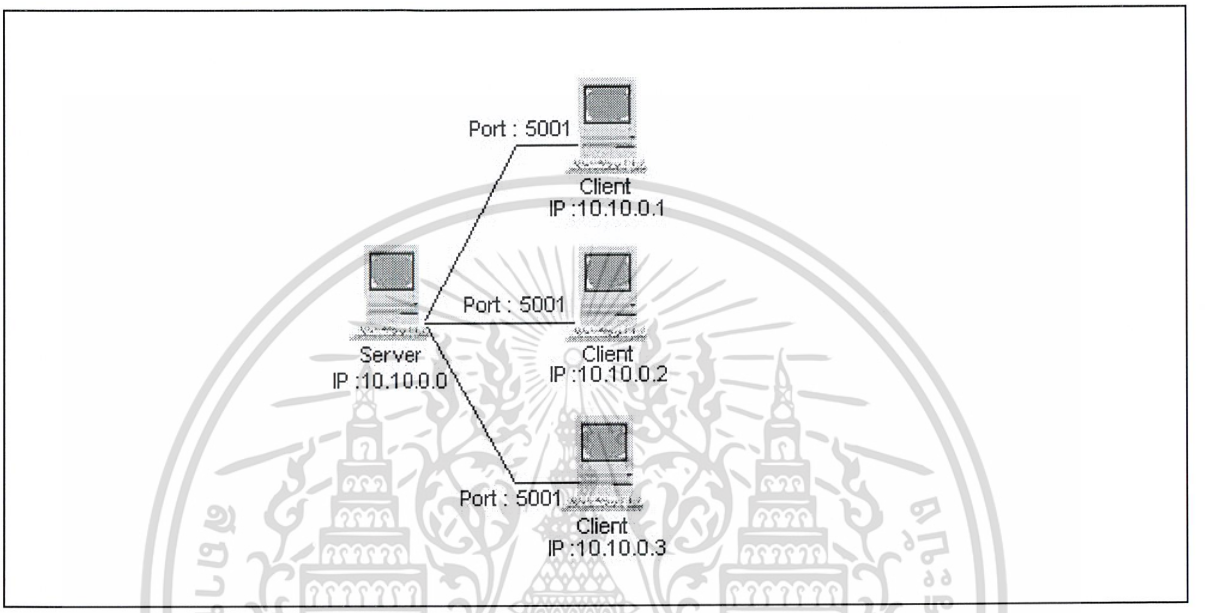
1. ทีซีพี (Transmission Control Protocal: TCP)
2. ไอพี (Internet Protocal: IP)

ในการติดต่อสื่อสารกันจริงๆแล้วคงจะไม่สามารถเห็นขั้นตอนการทำงานของระบบได้เพราะเป็นการทำงานของซอฟแวร์และฮาร์ดแวร์แต่สามารถอธิบายเพื่อให้เข้าใจโปรโตคอลทีซีพี-ไอพีซึ่งมีส่วนประกอบดังนี้

- ตำแหน่งไอพี (IP Address) สำหรับการรับส่งข้อมูลในระบบอินเตอร์เน็ตจะถูกกำหนดและอ้างอิงด้วยหมายเลขประจำเครื่องนั้นก็คือตำแหน่งไอพีซึ่งในระบบอินเตอร์เน็ตจะมีเครื่องคอมพิวเตอร์เป็นจำนวนมากที่อยู่ในระบบ ดังนั้นการที่จะใช้ตำแหน่งไอพีอาจจะไม่สะดวก จึงได้มีการเปลี่ยนมาใช้เป็นชื่อ ในความเข้ากันก็คือชื่อโดเมน (Domain name) โดยทั้งหมดนี้อยู่ในระบบบริการชื่อ
- ลักษณะเร้าติ้ง (Routing Configuration) ข้อดีของโปรโตคอลทีซีพี-ไอพีก็คือในการกำหนดเส้นทางสำหรับการรับส่ง ที่สามารถเลือกเส้นทางในการรับส่งข้อมูลได้อย่างอัตโนมัติหากถ้าเกิดเส้นทางบางเส้นทางเสียหาย ระบบกลไกในการกำหนดเส้นทางสำหรับการรับส่งข้อมูลของโปรโตคอลทีซีพี-ไอพีก็จะเลือกเส้นทางให้เหมาะสมถูกต้องให้สามารถรับส่งข้อมูลได้
- โปรโตคอล (Protocal), พอร์ท (Ports), ซ็อกเก็ต (Socketes) เป็นช่องทางสำหรับกำหนดทิศทางของการรับส่งข้อมูลนอกเหนือจากที่จะต้องกำหนดหลังจากตำแหน่งไอพี

4.2.2 เซิร์ฟเวอร์และไคลเอ็นท์ (Server and Client)

สิ่งที่เราจะพูดถึงในการเขียนโปรแกรมเพื่อใช้ในระบบเครือข่าย (Network) จุดหลักๆของระบบ จะแบ่งฝ่ายที่ต้องติดต่อรับส่งข้อมูลระหว่างกันออกเป็น 2 ส่วน คือเซิร์ฟเวอร์ (Server) และไคลเอ็นท์ (Client) ซึ่งในการใช้งานจริงอาจมีส่วนประกอบอื่นๆอีก ซึ่งจะขอไม่กล่าวถึงเนื่องจากต้องการให้เห็นภาพและเข้าใจง่ายขึ้นจึงยกแค่ 2 ส่วนนี้มากล่าว



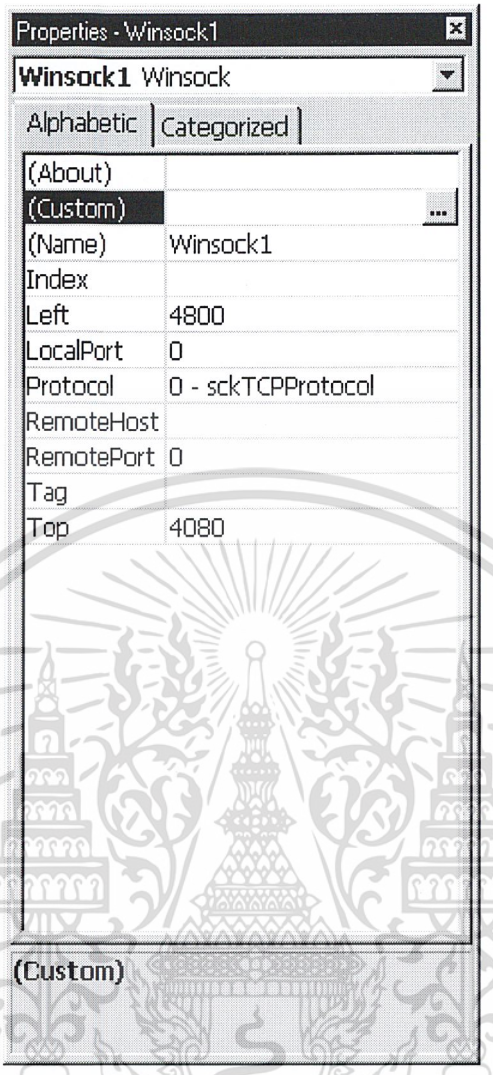
รูปที่ 4.2 โครงสร้างการทำงานระหว่างไคลเอ็นท์กับเซิร์ฟเวอร์

เซิร์ฟเวอร์จะเป็นส่วนทำหน้าที่เสมือนกองอำนวยการ, ประชาสัมพันธ์, เมสเซทเจอร์ ร่วมถึงผู้จัดให้กับระบบ ส่วนไคลเอ็นท์จะเป็นส่วนร้องขอข้อมูลจากเซิร์ฟเวอร์ โดยเซิร์ฟเวอร์ในที่นี้จะเป็นส่วนที่เก็บข้อมูล, จัดการ, บริหารข้อมูลหรือทรัพยากรของระบบ เพื่อให้ฝ่ายไคลเอ็นท์สามารถใช้บริการได้

เซิร์ฟเวอร์และไคลเอ็นท์ต่างก็จะต้องมีตำแหน่งไอพี , พอร์ท (Port) โดยทั้งสองฝ่ายจะสามารถติดต่อถึงกันได้จะต้องอยู่ในช่องทางเดียวกัน ซึ่งเราสามารถกำหนดหมายเลขของพอร์ทได้ ทั้งนี้โปรแกรมที่ติดต่อนั้นจะต้องอ้างอิงหมายเลขของพอร์ททุกครั้งเนื่องจากในระบบมีโปรแกรมมากมายที่กำลังติดต่อกันอยู่

4.3 ไมโครซอฟวินซ็อคคอนโทรล (MS Winsock Control 6)

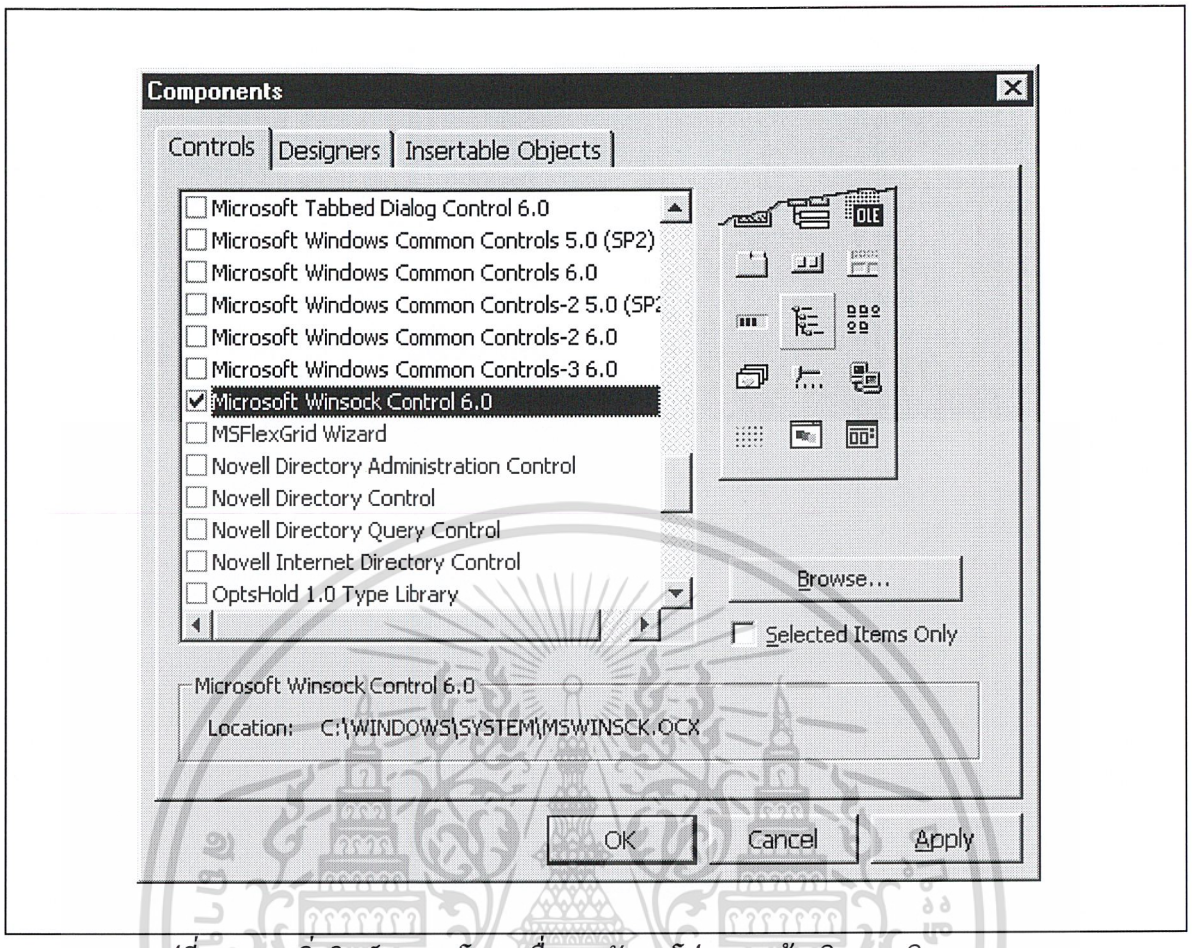
วินซ็อคคอนโทรลเป็นออฟเจ็ค (Object) ในการช่วยในการเข้าถึงบริการต่าง ๆ ของเครือข่ายทีซีพีและยูดีพี ซึ่งวินซ็อคคอนโทรลนี้สามารถ ใช้ได้กับหลายๆโปรแกรม เพื่อที่จะพัฒนาโปรแกรมทางด้านไคลเอ็นท์/เซิร์ฟเวอร์ โดยที่โปรแกรมเมอร์ไม่จำเป็นต้องมีความเข้าใจเกี่ยวกับทีซีพีหรือ การใช้งานฟังก์ชันเอพีไอของวินซ็อคเลย แค่ทำการกำหนดที่คุณสมบัติของอุปกรณ์และทำการเรียกใช้เมธอด (Method) ก็สามารถ ทำการติดต่อเครื่องระยะไกล (Remote) และแลกเปลี่ยนข้อมูลได้แล้ว



รูปที่ 4.3 คุณสมบัติของไมโครซอฟวินซ็อคคอนโทรล

Properties	Detail
Name	ชื่อของคอนโทรลที่เราสามารถตั้งได้
LocalPort	Portsที่เราจะกำหนดในการติดต่อ
Protocal	เลือกชนิดของโปรโตคอลปกติเราใช้ 0-TCPProtocal
RemoteHost	IP Address ของเครื่องที่ใช้ ติดต่อ
RemotePort	Port Address ของเครื่องที่ใช้ติดต่อ

ตารางที่ 4.1 รายละเอียดของคุณสมบัติแต่ละตัวของไมโครซอฟวินซ็อคคอนโทรล



รูปที่ 4.4 การเพิ่มวินซ็อคคอนโทรลเพื่อการพัฒนาโปรแกรมด้วยวิชวลเบสิก

ในการเขียนโปรแกรมวิชวลเบสิกเพื่อการสื่อสารผ่านเครือข่ายโดยใช้ไมโครซอฟวินซ็อคนั้นมีจุดประสงค์หลักคือสามารถเป็นได้ทั้งไคลเอ็นท์และเซิร์ฟเวอร์ได้ในเครื่องเดียวกัน โดยการที่จะใช้งานวินซ็อคนั้นจำเป็นต้องมีการเพิ่มส่วนของวินซ็อคคอนโทรลซึ่งแสดงในรูปที่ 4.4

4.3.1 พื้นฐานของ TCP

ทีซีพี (Transfer Control Protocol) ช่วยในเรื่องของการสร้างและคงสถานะการติดต่อ ไปยังเครื่องระยะไกล สิ่งแรกที่ควรจะรู้เมื่อกำลังสร้างแอพพลิเคชั่นสำหรับไคลเอ็นท์คือ เราต้องรู้ตำแหน่งไอพีของเครื่องเซิร์ฟเวอร์หรือชื่อเครื่องเซิร์ฟเวอร์ (RemoteHost Property) และต้องรู้พอร์ทที่เครื่องเซิร์ฟเวอร์ใช้ในการติดต่อ (RemotePort Property)

เมื่อทำการสร้างเซิร์ฟเวอร์แอพพลิเคชั่นแล้ว เราจำเป็นจะต้องเปิดพอร์ทเพื่อทำการรับฟังการติดต่อ (LocalPort Property) เมื่อทำการกำหนดพอร์ทแล้ว จำเป็นที่จะต้องเรียกเมธอด Listen เพื่อทำการเริ่มการรับฟังการติดต่อจากฝั่งไคลเอ็นท์ (Client Side) เมื่อไคลเอ็นท์สร้างการติดต่อเข้ามา จะเกิดเหตุการณ์ ConnectionRequest ที่ฝั่งเซิร์ฟเวอร์แอพพลิเคชั่นและเพื่อจะตกลงเซิร์ฟเวอร์จะต้องเรียกใช้เมธอด Accept ภายในเหตุการณ์ ConnectionRequest

เมื่อทั้ง 2 ฝั่งสามารถติดต่อกันได้แล้วนั้น ทั้งคู่สามารถติดต่อส่งข้อมูลได้ โดยการส่งใช้เมธอด SendData และ ฝั่งรับ รับข้อมูลที่เหตุการณ์ DataArrival โดยใช้เมธอด GetData

4.3.2 ขั้นตอนการติดต่อ ระหว่างไคลเอ็นท์และเซิร์ฟเวอร์

ขั้นที่ 1 Server: Listening on port XXXX

ขั้นที่ 2 Client : Request a connection

ขั้นที่ 3 Server : Accept Connection

คราวนี้เราจะมาพูดถึงว่าทำอย่างไรที่เราจะเขียนโปรแกรมบนระบบวินโดว์ (Windows) ให้สามารถติดต่อสื่อสารบนระบบเครือข่ายได้ ถ้าเป็นเมื่อก่อนบนระบบปฏิบัติการดอส (DOS) นั้นคงจะยากทีเดียว แต่เดี่ยวนี้เครื่องคอมพิวเตอร์ใช้ระบบวินโดว์กันแล้วเนื่องจากมีเครื่องมือ (Tool) ที่ช่วยให้สามารถเขียนโปรแกรมติดต่อผ่านระบบเครือข่าย ที่กล่าวถึงก็คือไมโครซอฟวินซ็อคคอนโทรล 6 (MS Winsock Control 6) เป็นเครื่องมือที่อำนวยความสะดวกสำหรับการเขียนโปรแกรมติดต่อผ่านระบบเครือข่าย โดยโปรโตคอลทีซีพี/ไอพี ซึ่งจะทดลองเขียนด้วยวิชวลเบสิค ส่วนนอกเหนือจากการที่กล่าวมาแล้วเกี่ยวกับวินซ็อคคอนโทรลแล้วยังมีเอพีไอ (API) ที่มีนักพัฒนาเขียนขึ้นมาหลายตัวทีเดียว แต่ดูๆแล้ววินซ็อคจะใช้งานง่ายกว่า

4.3.3 วินซ็อคโพรซิเดอร์ (Winsock Procedure)

ในส่วนของวินโดว์โค้ดอ็อฟเจ็ค (Windows Code Object) หรือวินซ็อคนั้นมีโพรซิเดอร์ (Procedure) สำหรับกระทำติดต่อสื่อสารทั้งทางฝ่ายเซิร์ฟเวอร์และไคลเอ็นท์ โดยที่

- Close คือ เหตุการณ์เมื่อมีหยุดหรือยกเลิกการติดต่อสื่อสารของฝ่ายเซิร์ฟเวอร์หรือไคลเอ็นท์ โดยฟังก์ชัน Winsock.Close ซึ่งจะเราสามารถจะใช้ตรวจสอบฝ่ายตรงข้ามว่ามีการติดต่ออยู่หรือไม่ โดยอาจจะใส่ข้อความเตือนเป็นต้น
- Connect เป็นเหตุการณ์ที่ฝ่ายไคลเอ็นท์มีการส่งสัญญาณติดต่อกับมายังเซิร์ฟเวอร์ส่งผลให้โพรซิเดอร์ของฝ่ายเซิร์ฟเวอร์ทำงาน และเราสามารถนำข้อความของโค้ด (Code Massage) ไปใส่เพื่อตรวจสอบได้เช่นกัน
- ConnectionRequest เป็นเหตุการณ์เมื่อฝ่ายไคลเอ็นท์ส่งสัญญาณติดต่อกลับมายังเซิร์ฟเวอร์แล้วโพรซิเดอร์ส่วนนี้ก็จะทำงานพร้อมกับค่า requestID แบบ Long ซึ่งเป็นเครื่องหมายที่สร้างขึ้นมาในระบบโดยค่านั้นจะไม่เหมือนเดิม และให้ฝ่ายเซิร์เวอร์รับรู้ว่าใช้ไอดี (ID) จากคอนโทรลตัวใดเพื่อจะได้สื่อสารถึงต้อง
- DataArrival เหตุการณ์นี้เกิดขึ้นเมื่อมีการส่งข้อมูลระหว่างเซิร์เวอร์และไคลเอ็นท์ โดยที่โพรซิเดอร์ของเซิร์ฟเวอร์และไคลเอ็นท์นี้จะทำงานขึ้นมา พร้อมกับค่าจำนวน bytesTotal แบบ Long ที่รับเข้ามา

- Error เหตุการณ์ที่เกิดความผิดพลาดระหว่างการติดต่อสื่อสารระหว่างเซิร์เวอร์กับไคลเอ็นท์โดยจะส่งค่า Number ซึ่งเป็น Integer มาให้ว่าเป็นหมายเลขใดพร้อมทั้งรายละเอียดของการผิดพลาดในเหตุการณ์นั้นๆ คือ Description ซึ่งเป็น String
- SendProgress จะเกิดขึ้นในขณะที่มาการส่งข้อมูลอยู่เหตุการณ์นี้ก็จะทำงานเมื่อส่งข้อมูลหมดแล้วจะส่งผลทำให้เกิดเหตุการณ์ (Event) SendComplete
- SendComplete เหตุการณ์เมื่อมีการส่งข้อมูลออกไปยังฝ่ายตรงข้ามเสร็จเรียบร้อยแล้ว

4.3.4 **คุณสมบัติและเหตุการณ์ของวินซ็อค** (Winsock Properties & Events)

- Accept (requestID) คือการตกลงกันระหว่างเซิร์ฟเวอร์และไคลเอ็นท์ในการเลือกหมายเลขไอดีคอนโทรล (ID Control) ให้ตรงกันเพื่อสามารถสื่อสารได้ถูกต้อง
- Close เป็นการส่งสัญญาณยกเลิกการติดต่อระหว่างกัน จะเป็นฝ่ายเซิร์ฟเวอร์หรือไคลเอ็นท์ก็ได้ที่จะใช้ฟังก์ชันนี้ จากนี้จะทำให้โพรซิเดอร์ปิดการทำงานของฝ่ายตรงข้าม
- Connect เป็นการส่งสัญญาณว่าตอนนี้ทำการติดต่อเรียบร้อยแล้ว ซึ่งจะส่งผลให้โพรซิเดอร์ของฝ่ายตรงข้ามทำงาน
- Getdata เป็นการรับข้อมูลเมื่อฝ่ายตรงข้ามส่งมาโดยประโยคคำสั่งนี้จะอยู่ในส่วนของโพรซิเดอร์DataArrival เนื่องจากเป็นเหตุการณ์ที่การกระทำขณะเมื่อฝ่ายตรงข้ามส่ง ข้อมูลเข้ามา
- Listen การกระทำที่จะคอยตรวจสอบสัญญาณที่ส่งไปว่าฝ่ายตรงข้ามตอบรับการร้องขอการติดต่อ
- LocalHostName คำสั่งนี้จะส่งชื่อของคอมพิวเตอร์เครื่องนั้นๆ

  Debug.Print Winsock1. LocalHostName
- LocalIP คำสั่งนี้จะทำการส่งหมายเลขตำแหน่งไอพี

  Debug.Print Winsock1.LocalIP
- LocalPort คำสั่งที่จะส่งค่าของหมายเลขในการติดต่อทีซีพี/ไอพีของเครื่องนั้นๆ

  Debug.Print Winsock1.LocalPort
- RemoteHost กำหนดหรือคืนค่าชื่อคอมพิวเตอร์ของเครื่องที่จะทำการติดต่อ

  Winsock1.RemoteHost =MyServer
- RemoteHostIP กำหนดหมายเลขตำแหน่งไอพีของเครื่องที่จะทำการติดต่อ

  Winsock1. RemoteHostIP =10.10.0.0
- RemoteHostPort กำหนดหมายเลขพอร์ทที่จะใช้ในการติดต่อระหว่างกัน

  Winsock1. RemoteHostIP =5000
- SocketHandle จะคืนค่าของช่องทางที่ใช้ในการติดต่อระหว่างกันซึ่งสามารถเรียกดูได้ดังนี้

  Debug.Print Winsock1.SocketHandle

- State จะคืนค่าของสถานะของซ็อคเก็ตขณะที่ใช้ติดต่อระหว่างอยู่ โดยอาจจะใช้ตรวจสถานะโดยค่าคงที่เหล่านี้เช่น sckClosed (มีค่า=0) ซ็อคเก็ตปิดการใช้งาน, sckOpen (มีค่า= 1) ซ็อคเก็ตเปิดใช้งาน หรือ sckError (มีค่า = 9) ซ็อคเก็ตมีความผิดพลาดเกิดขึ้น เป็นต้น

4.3.5 กระบวนการติดต่อของวินซ็อค (Basic Winsock Process Connect)

จากที่อธิบายรายละเอียดต่างๆของวินซ็อคคอนโทรลมาแล้วคราวนี้ จะมาทำการเริ่มการเขียนโปรแกรมเพื่อที่จะติดต่อกับโปรโตคอลทีซีพี/ไอพีกันล่ะซึ่งการติดต่อจะต้องมีทั้งไคลเอ็นท์และเซิร์ฟเวอร์ ซึ่งก็จะมี 2 โปรแกรม แต่ความจริงแล้วก็คือโปรแกรมเดียวกันแต่เปิด 2 หน้าต่าง โดยกำหนดว่าฝั่งไหนเป็นไคลเอ็นท์หรือเซิร์ฟเวอร์ก็ได้

เหตุการณ์แรกที่ต้องทำในฐานะที่เป็น ฝ่ายเซิร์ฟเวอร์คือการตรวจสอบสัญญาณจากฝั่งไคลเอ็นท์ในที่นี้ใช้ เหตุการณ์คลิ๊ก (Event Click) ของปุ่มชื่อ cmdListen

```
Private Sub cmdListen_Click()
Winsock1.LocalPort = txtPortSvr.Text
Winsock1.Listen
End Sub
```

เหตุการณ์ที่สองในฐานะไคลเอ็นท์ก็จะตอบรับโดยส่งสัญญาณไปให้กับเซิร์ฟเวอร์

```
Private Sub cmdConnect_Click()
Winsock1.RemoteHost ="10.10.0.25"
Winsock1.RemotePort = "5000" ' กำหนดหมายเลขอื่นก็ได้ แต่ต้องให้ตรงกันทั้ง 2 ฝ่าย
Winsock1.Connect
End Sub
```

เมื่อคลิกปุ่ม cmdConnect ตอนนี้ทางวินซ็อคคอนโทรลทางฝั่งเซิร์ฟเวอร์จะเกิดเหตุการณ์ Connect ซึ่งในเหตุการณ์นี้เราอาจจะใส่ข้อความเตือนได้ดังนี้

```
Private Sub Winsock1_Connect()
MsgBox "ตอบรับการติดต่อกลับมาแล้ว", vbExclamation, "Chat by MS Winsock Control 6"
End Sub
```

ซึ่งก็เป็นอันว่าทั้งเซิร์ฟเวอร์และไคลเอ็นท์สามารถติดต่อสื่อสารกันได้แล้ว สำหรับตัวอย่างของการส่งข้อมูลเป็นดังนี้

```
Private Sub cmdSend_Click()
Winsock1.SendData "Hi! How are you ? "
End Sub
```

เมื่อฝ่ายเซิร์ฟเวอร์ส่งข้อมูลออกไปที่วินซ็อคคอนโทรล ทางฝั่งไคลเอ็นท์จะเกิดเหตุการณ์ DataArrival และรับข้อมูลนั้นด้วยคำสั่ง Getdata โดยเขียนโปรแกรมดังนี้

```
Sub Winsock1_DataArrival(ByVal bytesTotal As Long)
Dim StrValue As String
Winsock1.GetData StrValue
End Sub
```

ถ้าหากต้องการยกเลิกการติดต่อก็ใช้คำสั่ง Close เมื่อเซิร์ฟเวอร์หรือไคลเอ็นท์ใช้คำสั่งนี้ จะทำให้เกิดเหตุการณ์ Close ขึ้นกับวินซ็อคคอนโทรลของฝั่งนั้น ดังเช่นเราสร้างปุ่ม Close ไว้

```
Private Sub cmdClose_Click()
Winsock1.SendData "ยกเลิกการติดต่อแล้ว"
Winsock1.Close
End Sub
```

สมมุติว่าเมื่อฝั่งเซิร์ฟเวอร์มีการคลิ๊กปุ่มเกิดขึ้นก็จะทำให้วินซ็อคคอนโทรลฝั่งไคลเอ็นท์เกิดเหตุการณ์ Close โดยอาจจะเขียนข้อความเตือนไว้ดังต่อไปนี้

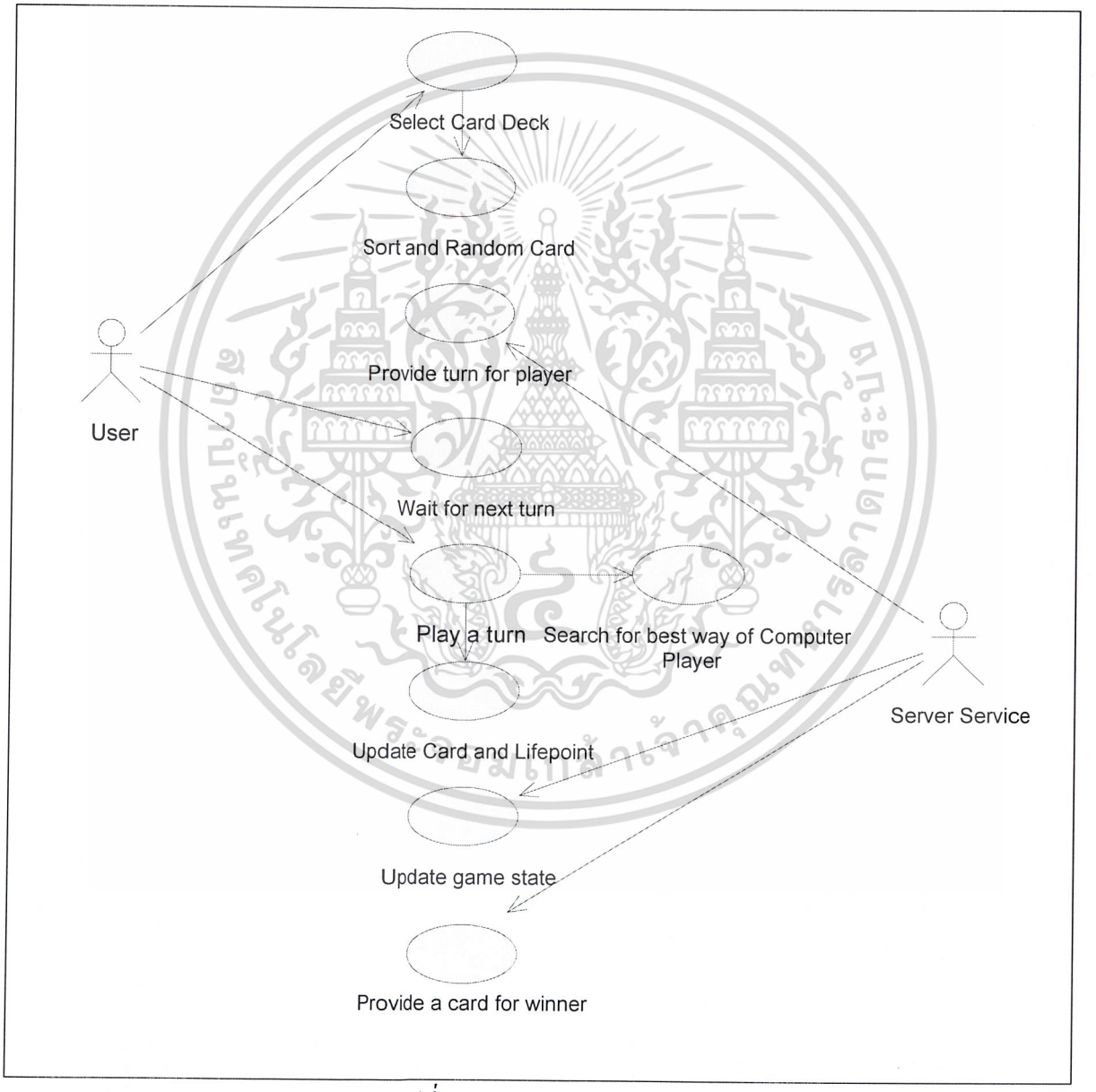
```
Private Sub Winsock1_Close()
MsgBox "ยกเลิกการติดต่อแล้ว", vbExclamation, "Chat by MS Winsock Control 6"
End Sub
```

บทที่ 5

การออกแบบ

5.1 ไดอะแกรม (Diagram)

5.1.1 Use Case Diagram

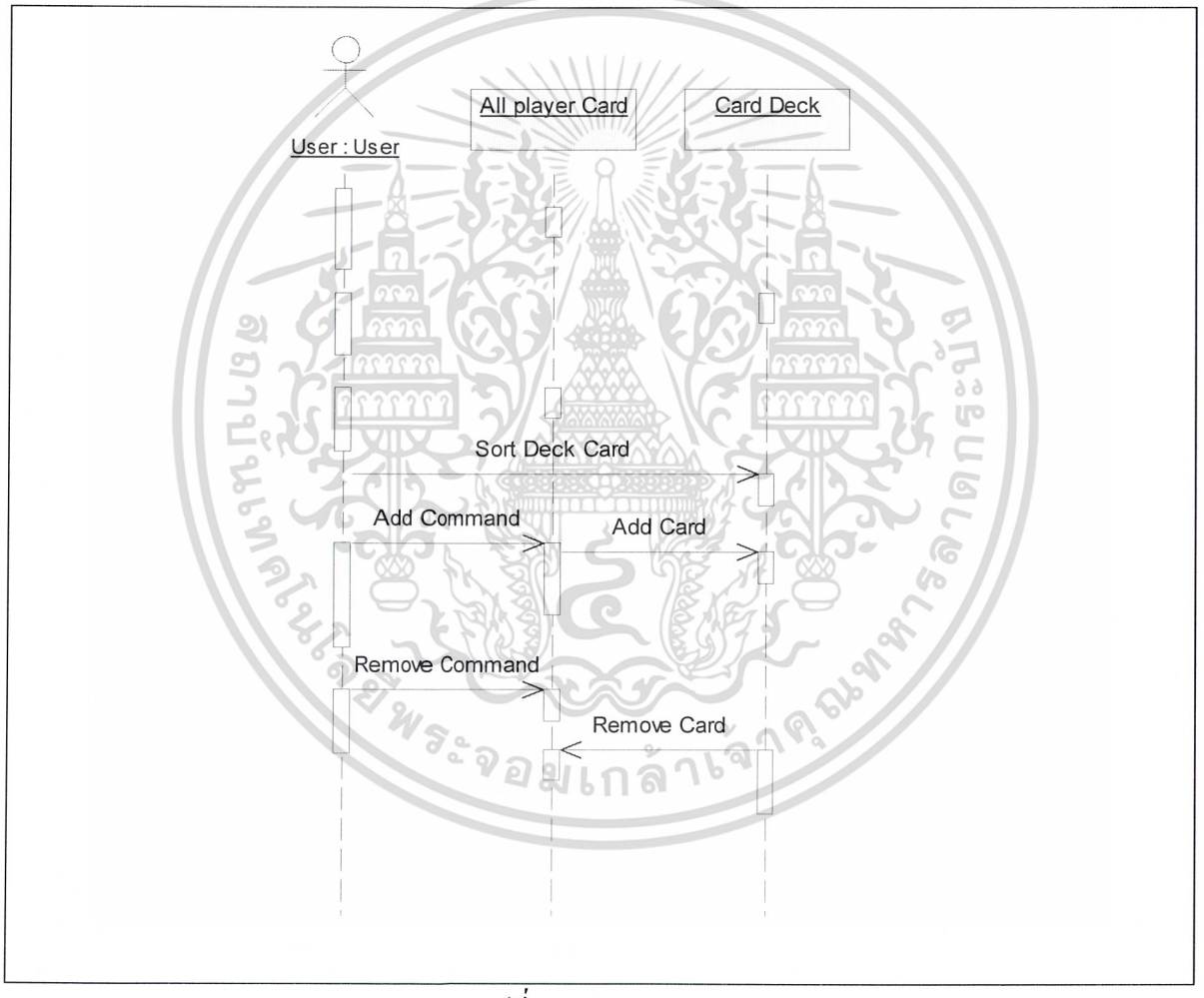


รูปที่ 5.1 Use Case Diagram

ผู้ใช้จะมีหน้าที่ในการเลือกชุดการ์ดที่จะใช้เล่น รวมไปถึงการซื้อขายการ์ด และเล่นเกมในเทิร์นนั้นๆ ที่เหลือก็คือต้องรอจนกว่าจะมีถึงเทิร์นของตัวเอง

หน้าที่อื่นๆ นอกจากนั้นเป็นหน้าที่ของคอมพิวเตอร์ ทั้งการจัดการระบบต่างๆ ในเกม การจัดการในแต่ละเทิร์นให้ผู้เล่น เปลี่ยนแปลงสถานะปัจจุบันของเกม พลังชีวิตของทั้งคอมพิวเตอร์ และผู้ใช้ และการจัดการเมื่อจบเกมหนึ่งๆ

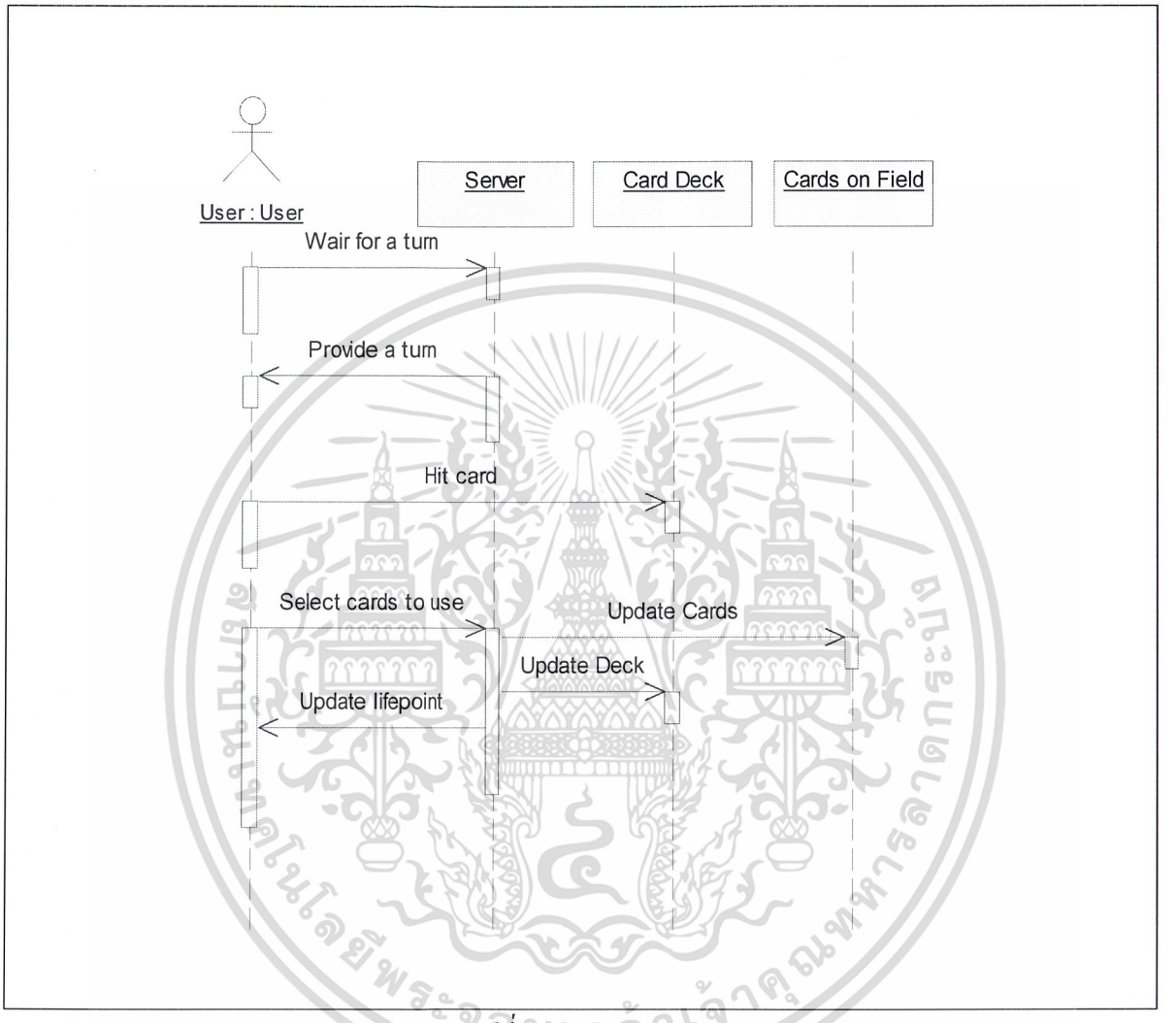
5.1.2 Sequence Diagram



รูปที่ 5.2 Select Deck

ในส่วนของการจัดชุดการ์ดนี้ เป็นส่วนของผู้เล่นที่จะสามารถเลือกได้ว่าจะเลือกเพิ่มการ์ดเข้าสู่ชุดการ์ด ก็จะส่งไปที่การ์ดทั้งหมด ก็จะมีการเพิ่มการ์ดนั้นๆ เข้าไปในชุดการ์ด

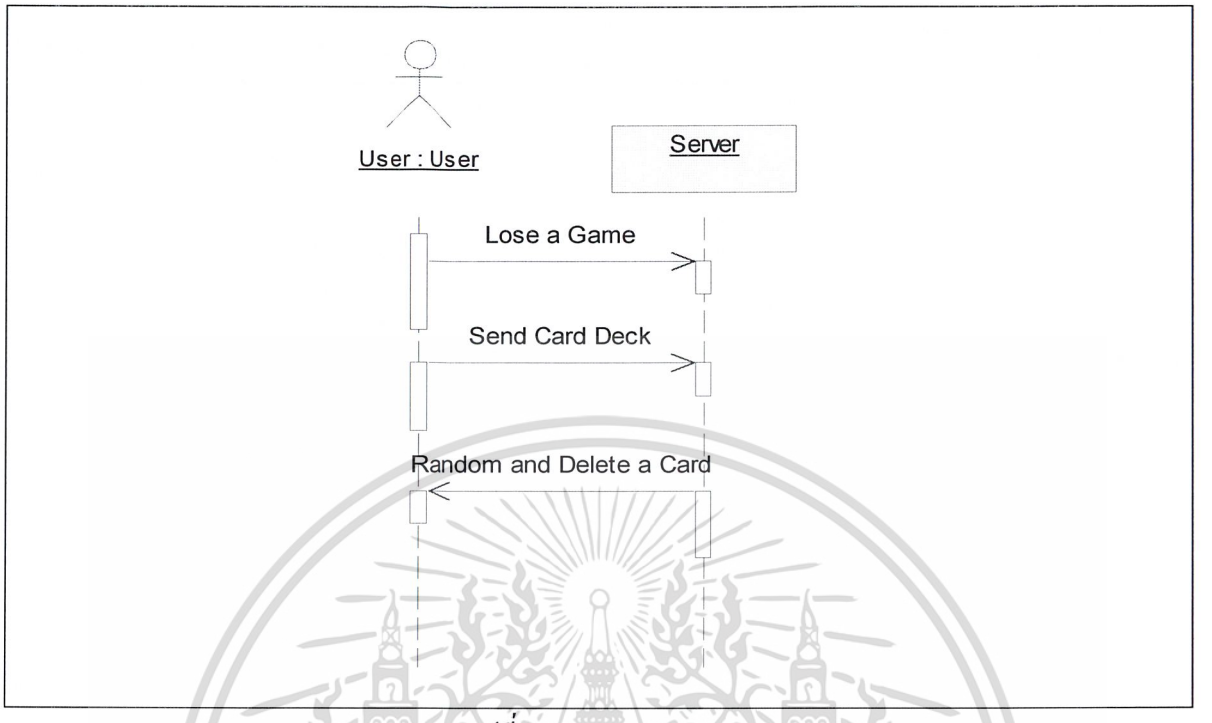
ส่วนเมื่อต้องการนำการ์ดออกมาจากชุดการ์ดก็ทำโดยส่งหมายเลขการ์ดเข้าไปที่ชุดการ์ด ชุดการ์ดก็จะส่งการ์ดนั้นออกมาให้กับการ์ดทั้งหมดเอง



รูปที่ 5.3 Play Game

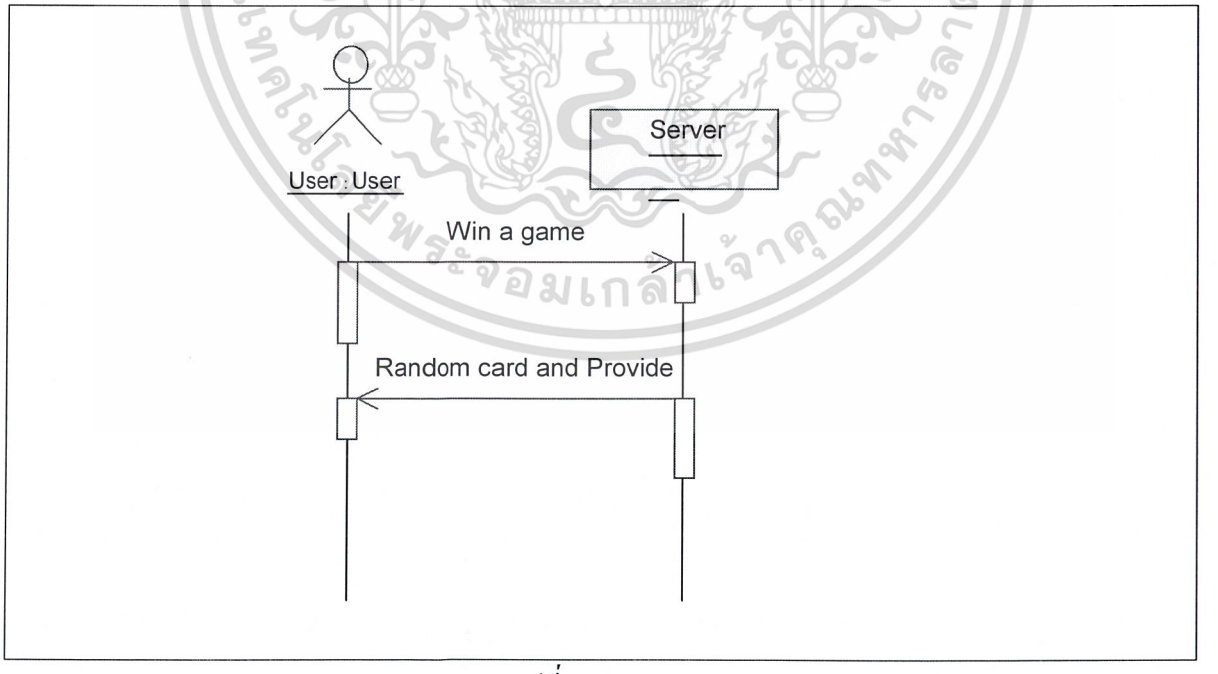
เมื่อเล่นเกมผู้เล่นจะต้องรอจนกว่าจะถึงเทิร์นของตัวเอง จึงจะมีสิทธิเล่น เมื่อ Server ส่งสัญญาณมาบอกว่าถึงเทิร์นตัวเองแล้ว ผู้เล่นก็จะจั่วไพ่ขึ้นมาเพื่อเริ่มเทิร์นใหม่

ผู้ใช้จะเลือกการ์ดที่จะลง ส่งไปยังเครื่องเซิร์ฟเวอร์ ส่วนหลังจากนี้เซิร์ฟเวอร์จะจัดการข้อมูลทั้งหมดให้เอง



รูปที่ 5.4 Remove Card

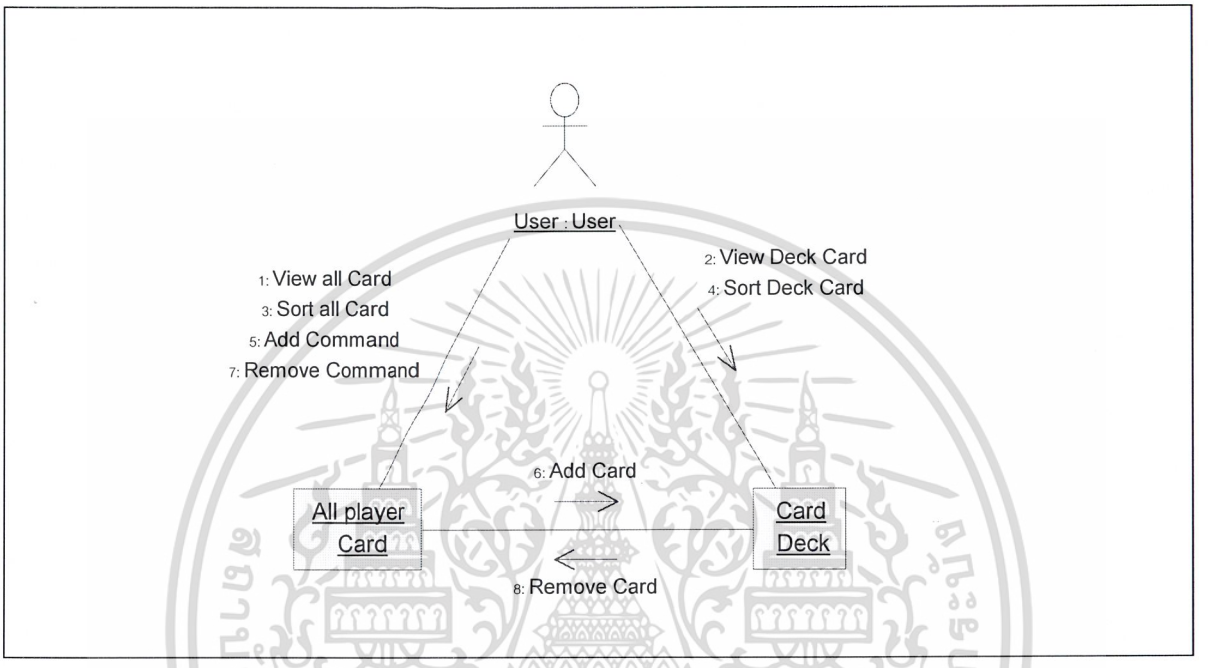
หากผู้ใช้แพ้ในการเล่นเซิร์ฟเวอร์ก็จะทำการสุ่มลบการ์ดจากผู้เล่น 1 ใบ



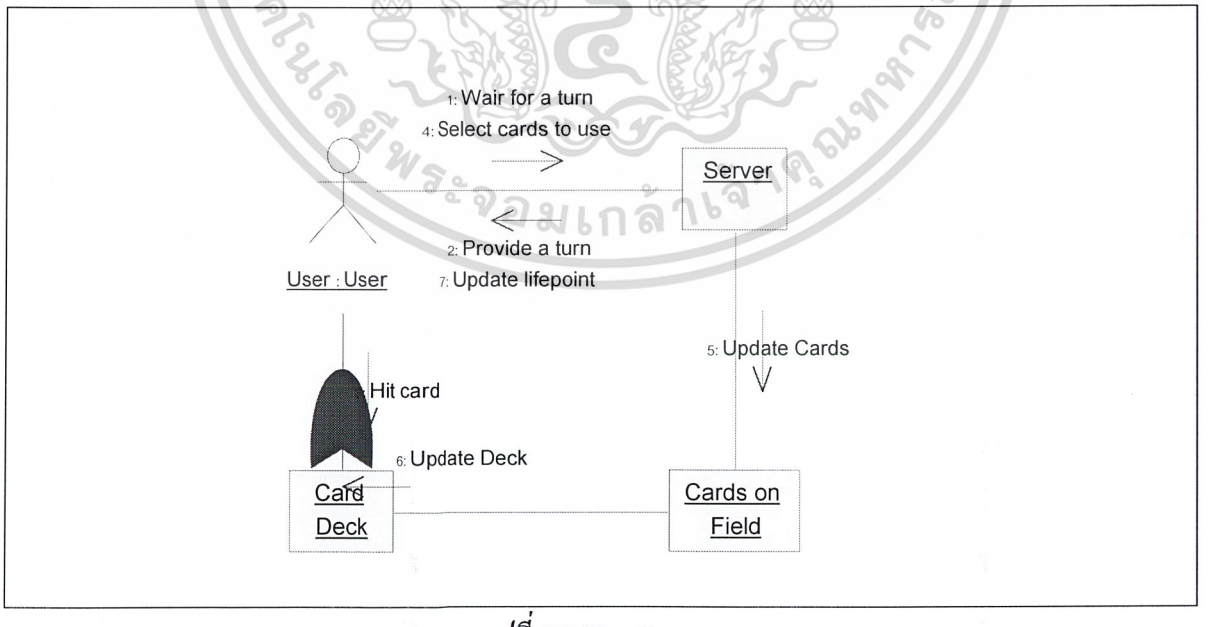
รูปที่ 5.5 Win Game

หากผู้ใช้ชนะในการเล่นเซิร์ฟเวอร์ก็จะทำการสุ่มเพิ่มการ์ดให้ผู้เล่น 1 ใบ

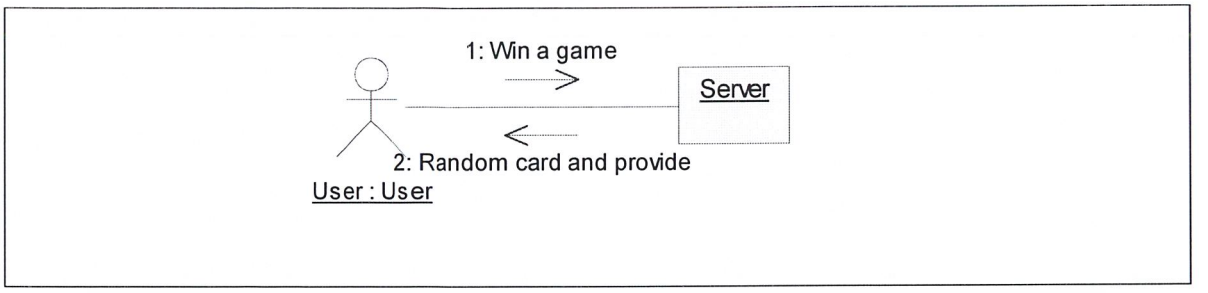
5.1.3 Collaboration Diagram



รูปที่ 5.6 Select Deck



รูปที่ 5.7 Play Game



รูปที่ 5.8 Provide Card

รูปที่ 5.9 Remove Card

5.2 หลักการออกแบบโปรแกรม

โปรแกรมเกมการ์ดบนเน็ตเวิร์กจะใช้การเก็บข้อมูลทั้งหมดในตาราง และอาร์เรย์ ประกอบด้วยเทมเพลตของการ์ดประเภทสิ่งมีชีวิตและเทมเพลตของการ์ดประเภทเวทมนตร์ซึ่งแต่ละตัวจะประกอบมีสมาชิกดังนี้

1. การ์ดประเภทสิ่งมีชีวิต

```
Public Type Creature
    Name As String
    Att As Integer
    Def As Integer
    Lev As Integer
    Chg As Integer
    Cal As Integer
    Pts As Integer
    Cost As Integer
```

```
End Type
```

2. การ์ดประเภทเวทมนตร์

```
Public Type Magic
    Name As String
    Des As String
    MType As Integer
    MyCard As Integer
    EnCard As Integer
    MyLP As Integer
    EnLP As Integer
    TEff As Integer
    LEff As Integer
    Trap As Integer
    turn As Integer
    EfTo As Integer
    Cost As Integer
End Type
```

การ์ดประเภทสิ่งมีชีวิตจะมีทั้งหมด 300 ใบ และการ์ดประเภทการ์ดเวทมนตร์จะมีทั้งหมด 180 ใบ รวมเป็น 480 ใบ ซึ่งเทมเพลตดังกล่าวนี้จะอยู่ในไฟล์โมดูลของโปรแกรม หรือเป็นโกลบอลเทมเพลต เนื่องมีการเรียกใช้ตลอดทั้งโปรแกรม ซึ่งเทมเพลตทั้ง 480 ใบนี้จะต้องมีการเรียกใช้เกือบทุกส่วนของการทำงาน ทั้งในส่วนของการจัดการกับชุดการ์ดเบื้องต้น การซื้อขายการ์ด จนกระทั่งการเล่น

ข้อมูลที่มีการส่งไปมา และใช้ในการเล่นทั้งหมดจะมีใช้เพียงแค่หมายเลขเฉพาะของการ์ดไว้เท่านั้น ถ้าต้องการใช้ข้อมูลอื่นๆ ของการ์ดใบนั้นๆ จะต้องมีการเรียกมาจากเทมเพลตทั้ง 2 ตัวดังที่ได้กล่าวไว้แล้ว

หมายเลขของการ์ดจะมีทั้งหมด 4 ตัว การ์ดประเภทสิ่งมีชีวิตจะมีหมายเลขเฉพาะขึ้นต้นด้วยเลข 1 ส่วนการ์ดประเภทเวทมนตร์จะมีหมายเลขเฉพาะขึ้นต้นด้วยเลข 2 สำหรับการ์ดสิ่งมีชีวิตจะมีประเภททั้งหมด 6 ประเภท ได้แก่

- การ์ดมนุษย์ (Human/Man) มีหมายเลขเฉพาะตัวที่ 2 เป็น 1
- การ์ดนักเวทมนตร์ (Wizard) มีหมายเลขเฉพาะตัวที่ 2 เป็น 2
- การ์ดสัตว์ป่า (Ground Animal) มีหมายเลขเฉพาะตัวที่ 2 เป็น 3

- การ์ดสัตว์ปีก (Flying Ainmal) มีหมายเลขเฉพาะตัวที่ 2 เป็น 4
- การ์ดวิญญาณ (Soul) มีหมายเลขเฉพาะตัวที่ 2 เป็น 5
- การ์ดปีศาจ (Devil) มีหมายเลขเฉพาะตัวที่ 2 เป็น 6

ส่วนการ์ดประเภทเวทมนตร์นั้นจะแบ่งออกได้เป็น 7 ประเภท

- ประเภททำลาย
- ประเภทเพิ่ม หรือลดพลังชีวิต
- ประเภทเพิ่ม หรือลดความสามารถของการ์ดประเภทสิ่งมีชีวิต
- ประเภทการ์ดกับดัก
- ประเภทการ์ดหยุดเวทมนตร์, หยุดการบูชายัญ (Sacrifice), หยุดการโจมตี
- ประเภทการ์ดขโมยการ์ด
- ประเภทการ์ดชุบชีวิต

โปรแกรมจะมีการเก็บข้อมูลของผู้ใช้ไว้ทั้งหมด 3 ตัวด้วยกัน ได้แก่

Dim AllC(480, 3) As Integer

ตัวแปร AllC ชนิดอาร์เรย์ 2 มิติของตัวเลขนี้ในช่องแรกจะเก็บหมายเลขเฉพาะของการ์ดทั้งหมดไว้ ไม่ว่าผู้ใช้จะมีการ์ดใบนั้นหรือไม่ก็ตาม เนื่องจากมีการ์ดอยู่ทั้งหมด 480 ใบ ดังนั้นจึงใช้ทั้งหมด 480 แถวในการเก็บ

ช่องที่สองจะเก็บจำนวนการ์ดใบที่มีหมายเลขตรงกับช่องแรกที่ผู้ใช้มีอยู่

ช่องที่สามสุดท้ายจะเก็บจำนวนการ์ดว่าอยู่ในชุดการ์ด (Deck) ของผู้ใช้กี่ใบ การที่ต้องเก็บเช่นนี้อาจจะมองว่าเป็นการเปลืองเนื้อที่มากกว่า แต่ความเป็นจริงๆ แล้วหากมีการเล่นมากครั้งมากๆ จำนวนไพ่เพิ่มขึ้นมากๆ วิธีดังกล่าวจะเป็นการประหยัดจำนวนการเก็บได้อย่างมาก

Public money As Integer

ตัวแปรนี้จะทำหน้าที่เก็บเงินของผู้ใช้ไว้ เนื่องจากในเกมมีการให้ซื้อขายการ์ดได้ที่ร้านขายการ์ด (Card Shop) ซึ่งอยู่ในเกม ดังนั้นผู้ใช้แต่ละคนจะต้องมีจำนวนเงินของเขาอยู่ ในการเล่นแต่ละครั้งเมื่อผู้ใช้ชนะ จะได้รับเงินจำนวน 5 แต้ม แพ้จะได้ 2 แต้ม

Public status As Integer

ตัวแปรนี้ทำหน้าที่ในการเก็บเนื้อเรื่องของเกมสำหรับผู้ใช้คนนั้นๆ เอาไว้ เนื่องจากเนื้อเรื่องที่จำเป็นต้องใช้นั้นเป็นแค่ว่าสู้ถึงตัวไหนแล้วในเกม ดังนั้นจึงใช้เป็นชนิดตัวเลขจะง่ายที่สุด

เกมการ์ดบนเน็ตเวิร์กสามารถบันทึก และโหลดข้อมูลเก่าของผู้ใช้ได้ โดยข้อมูลจะเก็บ และดึงมาจากไฟล์ซึ่งบันทึกเป็นไฟล์ข้อความธรรมดานั่นเอง การจะบันทึกข้อมูลลงไปจะต้องมีการใส่ Keyword หรือ Password ให้

กับไฟล์ด้วย เพื่อเป็นการยืนยันตัวตนของเจ้าของไฟล์จริงๆ ไม่เช่นนั้นจะไม่สามารถโหลดข้อมูลขึ้นมาได้ ข้อมูลที่มีการบันทึกลงในไฟล์จะเป็นดังนี้

- ขั้นแรก Passwaord จะมีการเข้ารหัสอย่างง่ายๆ คือ แปลงตัวอักษรแต่ละตัวเป็นรหัสแอสกี (ASCII) แล้วจับคูณด้วย 2 แล้วค่อยแปลงกลับมาเป็นตัวอักษรอีกครั้ง
- ต่อมาจำนวนเงินจะถูกเขียนลงในไฟล์ต่อ
- ต่อมาเนื้อเรื่องของผู้ใช้จะถูกเขียนลงในไฟล์ต่อ
- ต่อมาจะเป็นตัวเลขที่ทำหน้าที่ตรวจสอบความถูกต้องของไฟล์ หาได้จากการนำเอาผลรวมของค่าแอสกีของรหัสผ่านผู้ใช้ บวก จำนวนเงินของผู้ใช้ รวมมารวมกับผลคูณระหว่าง หมายเลขเฉพาะของการ์ดกับผลบวกของจำนวนการ์ดนั้นที่ผู้ใช้มีกับที่มีอยู่ในชุดการ์ด สุดท้ายบวกด้วยค่าเนื้อเรื่องอีกที

ส่วนขั้นตอนในการโหลดข้อมูลขึ้นมาก็จะทำการคำนวณค่าในไฟล์ตามลำดับเดิมใหม่ แล้วนำผลลัพธ์ที่ได้มาเทียบกับเลขรหัสความถูกต้องของไฟล์ ถ้าไม่ตรงกันแสดงว่าไฟล์มีการเปลี่ยนแปลงไป จะยกเลิกการโหลดข้อมูลทั้งหมด

ในส่วนของร้านขายการ์ดนั้นจะมีการดึงเอาข้อมูลมาจากเทมเพลตหลักเช่นกัน ทั้งในส่วนซื้อ และส่วนการขายการ์ด การซื้อการ์ดก็จะมีการตรวจสอบว่าราคามากเกินเงินที่มีหรือไม่ หากซื้อการ์ดมาแล้วก็จะทำการบวกค่าช่องที่ 2 ของตัวแปร AllC ขึ้น 1 ส่วนเวลาขายก็ลบออก 1 ในเวลาขายต้องมีการตรวจสอบด้วยว่าหากการ์ดที่ขายอยู่ในชุดการ์ดก็ต้องลบค่าช่องที่ 3 ของตัวแปรออกไปด้วย การ์ดที่ผู้ใช้ขายออกไปจะนำราคามาหารสองเป็นเงินให้ผู้ใช้ เพื่อให้การ์ดมีคุณค่ามากขึ้น

ส่วนการจัดชุดการ์ดก็เช่นกัน ก็จะใช้การบวก และลบค่าจากช่องที่ 3 ของตัวแปร AllC นั่นเอง

สำหรับการแสดงผลรูปภาพของการ์ดแต่ละใบนั้นจะใช้หมายเลขเฉพาะของการ์ดแต่ละใบเป็นตัวอ้างอิง ซึ่งรูปภาพของการ์ดจะถูกเก็บไว้ต่างหากในไดเร็กทอรีชั้นใน แล้วอ้างอิงจากไดเร็กทอรีปัจจุบันเข้าไป ดังนั้นจะสามารถแสดงรูปภาพของการ์ดได้ทั้งโปรแกรม เนื่องจากต้องมีการแสดงรูปภาพในหลายๆ ช่วงของโปรแกรม ทั้งตอนเล่นเกม ตอนจัดชุดการ์ด ตอนร้านขายการ์ด

ในช่วงเล่นเกมเริ่มต้นก่อนเข้าเกมจะมีการตรวจสอบ ก่อนโดยนำเอาค่าช่องสุดท้ายทั้งหมดของตัวแปร AllC มารวมกันทั้งหมด หากไม่เท่ากับ 50 ก็จะไม่สามารถเข้าได้ คือ ผู้ใช้ต้องเข้าไปจัดชุดการ์ดให้ครบทั้ง 50 ใบก่อนที่จะเข้าเล่นเกม เมื่อเข้าไปแล้วจะทำการตรวจสอบค่าเนื้อเรื่องของผู้ใช้ เพื่อเลือกชุดการ์ดมาเป็นชุดการ์ดของคอมพิวเตอร์ เมื่อได้แล้วจะการสับไพ่ทั้งหมด โดยใช้ทฤษฎีง่ายๆ คือสุ่มค่าตำแหน่งของชุดการ์ดขึ้นมา 2 ค่า แล้วจับสลับกัน ไปเรื่อยๆ มากๆ ครั้งเข้าการ์ดจะเหมือนมีการสุ่มการจัดเรียงเอง

สำหรับตัวแปรที่ใช้ในการเล่นนั้นจะประกอบด้วย

- u1card เป็นอาร์เรย์ ชนิดตัวเลขเก็บชุดการ์ดของผู้ใช้
- u2card เป็นอาร์เรย์ ชนิดตัวเลขเก็บชุดการ์ดของคอมพิวเตอร์
- Index1 เป็นตัวเลข เก็บตำแหน่งอ้างอิงปัจจุบันของชุดการ์ดของผู้ใช้

- Index2 เป็นตัวเลข เก็บตำแหน่งอ้างอิงปัจจุบันของชุดการ์ดของคอมพิวเตอร์
- Lp1 เป็นตัวเลข เก็บพลังชีวิตของผู้ใช้
- Lp2 เป็นตัวเลข เก็บพลังชีวิตของคอมพิวเตอร์
- field เป็นอาร์เรย์ชนิดตัวเลข เก็บไพ่ที่อยู่บนสนามการต่อสู้ (Battle Field) ของเกม มีจำนวน 16 ตัว 8 ตัวแรกเป็นของผู้ใช้ 7 ตัวแรกเป็นการ์ดสิ่งมีชีวิตบนสนามการต่อสู้ ตัวที่ 8 เป็นการ์ดกับดักของผู้ใช้ อีก 8 ตัวหลังเป็นของคอมพิวเตอร์ 7 ตัวแรกเป็นการ์ดสิ่งมีชีวิต ตัวที่ 16 เป็นการ์ดกับดักของคอมพิวเตอร์เช่นกัน
- tfield เป็นอาร์เรย์ชนิดตัวเลข มี 16 ช่องตามตัวแปร field ปกติจะเป็น 0 ทุกช่อง เมื่อมีการลงไพ่ใดๆ ก็ตามลงมาในสนามเป็นครั้งแรกในเกมจะไม่สามารถใช้โจมตีได้ ตัวแปร tfield จะถูกเปลี่ยนเป็น 1 เวลาที่จะมีการโจมตีก็จะมาดูค่าที่ตัวแปร tfield หากเป็น 1 ก็จะไม่สามารถโจมตีได้ เมื่อจบเทิร์นใดๆ แล้วจะมีการเปลี่ยนทุกๆ ช่องในตัวแปร tfield กลับเป็น 0 อีกครั้ง
- udfield เป็นอาร์เรย์ชนิดตัวเลข มี 16 ช่องตามตัวแปร field เช่นเดียวกัน ทำหน้าที่เก็บระดับที่มีการเปลี่ยนแปลงไปของไพ่แต่ละใบในสนามการต่อสู้ เนื่องมาจากการใช้การ์ดเวทมนตร์ชนิดเพิ่ม หรือลดพลังของการ์ด ค่าพลังที่แท้จริงของไพ่บนสนามการต่อสู้จะมาการค่าพลังของการ์ดใบนั้น รวมกับค่าจาก udfield ช่องที่ไพ่ใบนั้นอยู่
- stfield เป็นอาร์เรย์ชนิดตัวเลข มี 16 ช่องตามตัวแปร field ทำหน้าที่ เก็บในการที่มีการใช้การ์ดชนิดขโมยการ์ด เนื่องจากการ์ดขโมยการ์ดบางใบจะมีการขโมยการ์ดตามจำนวนเทิร์น ตัวแปร stfield นี้จะทำหน้าที่เป็นเหมือนกับตัวจับเวลา เมื่อหมดค่าแล้วจะมีการส่งไพ่ใบที่อยู่บนตำแหน่งนั้นๆ บนสนามการต่อสู้กลับคืนไปยังอีกฝ่ายหนึ่งตามเดิม
- OnHand1 เป็นอาร์เรย์ชนิดตัวเลข มี 5 ตัว เก็บค่าไพ่ที่อยู่บนมือของผู้ใช้ ในขณะนั้นๆ เนื่องจากผู้เล่นจะมีไพ่อยู่บนมือในขณะใดๆ ได้ไม่เกิน 5 ใบ เมื่อจะขึ้นเทิร์นของตัวเองก็จะมีการส่งค่าจากตัวแปร u1card ที่ตำแหน่งที่ตัวแปร Index1 ชี้อยู่ แล้วบวกค่า Index1 ขึ้น 1 ทำเช่นนี้จนกว่าจะมี ตัวแปร OnHand1 จะไม่มีช่องใดเป็น 0 และหาก Index1 มากกว่า 50 ก็จะให้แพ้เกมนั้นทันที
- OnHand2 เป็นอาร์เรย์ชนิดตัวเลข มี 5 ตัว เก็บค่าไพ่ที่อยู่บนมือของคอมพิวเตอร์ ในขณะนั้นๆ การจั่วไพ่ขึ้นมาบนมือก็เช่นเดียวกันกันของผู้ใช้ แต่จะใช้อ้างอิงจากตัวแปร u2card และ Index2 แทน
- Bin1 เป็นอาร์เรย์ชนิดตัวเลข มี 50 ตัวใช้เก็บการ์ดประเภทสิ่งมีชีวิตที่ถูกโจมตีตายไป การ์ดที่ถูกทิ้งการ์ดไป และการ์ดที่ถูกใช้บูชายัญของผู้ใช้ จะถูกเรียกมาใช้งานเมื่อมีการใช้การ์ดประเภทชุบชีวิต
- Bin2 เป็นอาร์เรย์ชนิดตัวเลข มี 50 ตัวใช้เก็บการ์ดประเภทสิ่งมีชีวิตที่ถูกโจมตีตายไป การ์ดที่ถูกทิ้งการ์ดไป และการ์ดที่ถูกใช้บูชายัญของคอมพิวเตอร์

- B1 เป็นตัวเลข ใช้อ้างตำแหน่งของตัวแปร Bin1 เมื่อมีการ์ดมาลง Bin1 จะมีการเขียนลงที่ตำแหน่ง B1 แล้วบวกค่า B1 ขึ้น 1
- B2 เป็นตัวเลข ใช้อ้างตำแหน่งของตัวแปร Bin2 เมื่อมีการ์ดมาลง Bin2 จะมีการเขียนลงที่ตำแหน่ง B2 แล้วบวกค่า B2 ขึ้น 1
- Ssac1 เป็นตัวเลข ใช้เก็บในกรณี คอมพิวเตอร์ใช้การ์ดประเภทหยุดการบูชายัญ ค่า ssac1 จะเพิ่มขึ้นตามจำนวนเทิร์นในการ์ดเวทมนตร์ที่ใช้ ผู้ใช้เมื่อค่านี้มากกว่า 0 จะทำให้ไม่สามารถบูชายัญได้ เมื่อมีการจบเทิร์นแต่ละครั้งค่า ssac1 จะลดลง 1 จนกว่าจะเป็น 0
- Ssac2 เป็นตัวเลข ใช้เก็บในกรณี ผู้ใช้ใช้การ์ดประเภทหยุดการบูชายัญ ค่า ssac2 จะเพิ่มขึ้นตามจำนวนเทิร์นในการ์ดเวทมนตร์ที่ใช้ คอมพิวเตอร์เมื่อค่านี้มากกว่า 0 จะทำให้ไม่สามารถบูชายัญได้ เมื่อมีการจบเทิร์นแต่ละครั้งค่า ssac2 จะลดลง 1 จนกว่าจะเป็น 0
- Satt1 เป็นตัวเลข ใช้เก็บในกรณี คอมพิวเตอร์ใช้การ์ดประเภทหยุดการโจมตี ค่า satt1 จะเพิ่มขึ้นตามจำนวนเทิร์นในการ์ดเวทมนตร์ที่ใช้ ผู้ใช้เมื่อค่านี้มากกว่า 0 จะทำให้ไม่สามารถโจมตีได้ เมื่อมีการจบเทิร์นแต่ละครั้งค่า satt1 จะลดลง 1 จนกว่าจะเป็น 0
- Satt2 เป็นตัวเลข ใช้เก็บในกรณี ผู้ใช้ใช้การ์ดประเภทหยุดการโจมตี ค่า satt2 จะเพิ่มขึ้นตามจำนวนเทิร์นในการ์ดเวทมนตร์ที่ใช้ คอมพิวเตอร์เมื่อค่านี้มากกว่า 0 จะทำให้ไม่สามารถโจมตีได้ เมื่อมีการจบเทิร์นแต่ละครั้งค่า satt2 จะลดลง 1 จนกว่าจะเป็น 0
- Smag1 เป็นตัวเลข ใช้เก็บในกรณี คอมพิวเตอร์ใช้การ์ดประเภทหยุดเวทมนตร์ ค่า smag1 จะเพิ่มขึ้นตามจำนวนเทิร์นในการ์ดเวทมนตร์ที่ใช้ ผู้ใช้เมื่อค่านี้มากกว่า 0 จะทำให้ไม่สามารถใช้การ์ดเวทมนตร์ใดๆ ได้ เมื่อมีการจบเทิร์นแต่ละครั้งค่า smag1 จะลดลง 1 จนกว่าจะเป็น 0
- Smag2 เป็นตัวเลข ใช้เก็บในกรณี ผู้ใช้ใช้การ์ดประเภทหยุดเวทมนตร์ ค่า smag2 จะเพิ่มขึ้นตามจำนวนเทิร์นในการ์ดเวทมนตร์ที่ใช้ คอมพิวเตอร์เมื่อค่านี้มากกว่า 0 จะทำให้ไม่สามารถใช้การ์ดเวทมนตร์ใดๆ ได้ เมื่อมีการจบเทิร์นแต่ละครั้งค่า smag2 จะลดลง 1 จนกว่าจะเป็น 0
- Catt1 เป็นตัวเลข ใช้เก็บค่าของการ์ดที่เป็นผู้เริ่มต้นการโจมตี ไม่ว่าจะมาจากผู้ใช้สั่งโจมตี หรือคอมพิวเตอร์สั่งโจมตีก็ตาม
- Catt2 เป็นตัวเลข ใช้เก็บค่าของการ์ดที่เป็นผู้ถูกเลือกโจมตี ไม่ว่าจะเป็นผู้ใช้ถูกโจมตี หรือคอมพิวเตอร์ถูกโจมตีก็ตาม
- Cview เป็นตัวเลข ใช้เก็บหมายเลขเฉพาะของการ์ดที่จะถูกส่งออกไปแสดงผลรูปเต็มๆ ของการ์ด
- Csel เป็นตัวเลข ใช้เก็บการ์ดที่มีการคลิกเมาส์เลือกไว้ครั้งสุดท้าย ใช้เพื่อเมื่อมีการเลือกการทำงานต่อไป อาจจะเป็นสั่งขยายไพ่ สั่งลงไพ่ หรืออื่นๆ จะได้ทราบว่าเป็นการ์ดใบใด

ตัวเกมจะแบ่งออกเป็นฟอร์มต่างๆ หลายๆฟอร์มประกอบกัน ได้แก่

- FormAnime ทำหน้าที่ในการแสดงผลการต่อสู้ที่เกิดขึ้นระหว่างการ์ดแต่ละคู่ ในการเล่นเกม เมื่อมีการโจมตีกัน จะมีการส่งการทำงานมายังฟอร์มนี้เสมอ พร้อมทั้งค่าการ์ดที่โจมตี และการ์ดที่ถูกโจมตี และผู้ชนะเข้ามา ฟอร์มนี้ก็จะแสดงการโจมตี และแสดงให้เห็นว่าการ์ดใดชนะ หรือแพ้
- FormAnime2 ทำหน้าที่ในการแสดงการ์ดเวทมนตร์ที่ถูกเรียกใช้ เพื่อให้สามารถทราบได้ว่ามีใครใช้การ์ดเวทมนตร์ใบใดๆ บ้างในเกม ทั้งฝ่ายผู้ใช้ และฝ่ายคอมพิวเตอร์
- FormBuyC ทำหน้าที่ในการแสดงการทำงานในส่วนของร้านขายการ์ดที่เป็นในกรณีผู้ใช้ต้องการซื้อการ์ดจากร้านขายการ์ด จะมีการแสดงภาพของการ์ดทั้งหมดที่มีในร้าน พร้อมทั้งแสดงราคาค่าการ์ด เมื่อมีการเลือกซื้อการ์ดก็จะจัดการดึงข้อมูลมาจากหน้าหลักมาจัดการทั้งจำนวนการ์ด และจำนวนเงินของผู้ใช้
- FormGame ทำหน้าที่ในการดำเนินเกม เมื่อผู้ใช้เลือกจะเข้าไปเล่นเกมก็จะมีการดึงเอาข้อมูลผู้ใช้มาเนื่องจากจะมีการเก็บไว้ที่ FormMenu ทั้งหมด และทำการเริ่มต้นเกม จัดการเกี่ยวกับการเล่น และหน้าจอทั้งหมด รวมทั้งควบคุมการแพ้ ชนะของเกม เมื่อจบการทำงานในส่วนนี้แล้วก็จะมีปิดการทำงานในฟอร์มนี้ลง และส่งค่าแพ้ หรือชนะกลับไปให้กับ FormMenu เพื่อให้จัดการกับสถานะ รวมทั้งการ์ดต่อไป
- FormLoadG ทำหน้าที่ในการแสดงผลการโหลดข้อมูลของผู้ใช้ที่ได้มีการบันทึกไว้ก่อนหน้านี้แล้วขึ้นมาใช้งานต่อ ซึ่งหน้าจอนี้จะมีการรับเอาชื่อไฟล์ และรหัสผ่านไปตรวจสอบว่าตรงกันกับข้อมูลในไฟล์นั้นๆ หรือไม่ ถ้าตรงกันข้อมูลก็จะมีการโหลดขึ้นมา เมื่อโหลดเสร็จก็จะมีการส่งค่าทั้งหมดไปให้กับ FormMenu เพื่อให้เปลี่ยนข้อมูลของผู้ใช้ทั้งหมดเสีย
- FormMenu ทำหน้าที่ในการแสดงตัวเลือก หรือเมนูเพื่อให้ผู้ใช้ทำการเลือกการทำงาน ซึ่งในส่วนต่างๆ ของเกมทั้งหมดจะมีการเชื่อมต่อเข้าไปจากหน้าหลักซึ่งก็คือ FormMenu นี้ทุกหน้า ในหน้านี้จะมีทั้งเมนูหลักของโปรแกรมรวมไปถึง เมนูที่เป็นเมนูย่อยบางเมนูก็อยู่ในหน้านี้ด้วย เนื่องจากเป็นการส่งผ่านหน้าที่ไปหน้าอื่นๆ

เท่านั้น อีกทั้งลักษณะ รวมทั้งภาพของหน้าจอก็ไม่เปลี่ยนไปจากหน้าหลักมากนัก จึงไม่มีความจำเป็นต้องมีการสร้างฟอร์มใหม่ขึ้นมา ในหน้านี้จะมีการเก็บตัวแปร AllC ซึ่งก็คือการ์ดทั้งหมดของผู้ใช้ รวมทั้งจำนวนเงินและค่าเนื้อเรื่องของผู้ใช้ จะมีการเก็บอยู่ในหน้านี้ หน้าอื่นๆ ที่จะใช้ต้องมีการมาขอดึงค่าเอาไปจากฟอร์มนี้ทั้งหมด

ในส่วนของการให้คอมพิวเตอร์คิดเพื่อเล่นเกมเองนั้น แบ่งออกเป็น 2 ส่วน ได้แก่

1. ในส่วนของการลงการ์ด
2. ในส่วนของการโจมตี

สำหรับในส่วนแรกนั้นเดิมทีคิดที่จะใช้วิธีการของอัลกอริทึมแบบมินิแมกซ์ (Minimax) ควบคู่กับฟังก์ชั่นการประเมินค่าของสถานการณ์ (Evaluation Function) และใช้วิธีการค้นหาลงไปตามทรี แต่จากการทดลองใช้วิธีดังกล่าวแล้วจะมีลักษณะดังนี้

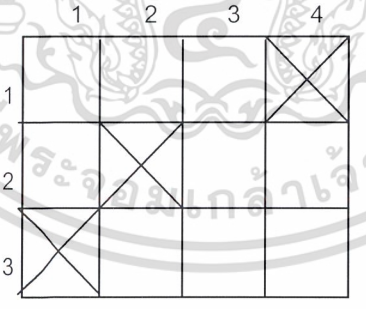
ในเทิร์นของเราเลือกลงการ์ดได้ตามจำนวนการ์ดสิ่งมีชีวิตที่มีอยู่ในมือ ในที่นี้สมมติให้มีอยู่ 2 ใบ อีก 3 ใบเป็นการ์ดเวทมนตร์ซึ่งเราสั่งให้ใช้ทั้งหมดอยู่แล้ว ดังนั้นจะสามารถลงได้เท่ากับ 2 วิธี

ในระดับถัดลงมาเป็นความน่าจะเป็นของการที่เราจะจั่วการ์ดใบใดๆ ขึ้นมาได้ เนื่องจากในตัวอย่างด้านบนเราได้ใช้การ์ดไป 4 ใบ ดังนั้นต้องจั่วขึ้นมา 4 ใบ สมมติว่าเป็นเทิร์นที่ 5 ค่าจะเป็นการเลือก 4 ใบจาก 40 จะมีค่าเท่ากับ 91390 วิธี

ในระดับถัดมาเป็นระดับของฝ่ายตรงข้าม ซึ่งเมื่อรวมเอาเหตุการณ์ที่เกิดขึ้นทั้งหมด จัดกลุ่มเข้าไว้ด้วยกันจะได้กลุ่มใหญ่ๆ ออกมา 17 กลุ่มเป็นวิธีการลงการ์ดของฝ่ายตรงข้าม ดังนั้นเมื่อรวมเอาค่าที่เกิดขึ้นทั้งหมดมารวมเป็น 1 ระดับ จะได้ว่าใน 1 ระดับจะต้องมีโหนดเกิดขึ้นเท่ากับ 2 x 91390 x 17 หรือเท่ากับ 3107260 โหนด ซึ่งถ้าต้องการใช้การค้นหาลงไปมากกว่า 1 ระดับ จะต้องคูณค่าเข้าไปเกือบเท่าตัว

ซึ่งเมื่อเปรียบเทียบวิธีการนี้ กับการใช้วิธีง่ายๆ โดยการวางรูปแบบการลงการ์ดให้กับคอมพิวเตอร์ไว้เลย ก็ไม่ทำให้ฉลาดน้อยลงเท่าไรนัก ซึ่งเมื่อเปรียบเทียบผลที่ได้แล้ว จึงเลือกวิธีการวางรูปแบบของการลงการ์ดไว้ เนื่องจากใช้เวลาน้อยกว่าการค้นหามาก ซึ่งถือว่ามีประสิทธิภาพสูงกว่า

ส่วนในการจัดรูปแบบของการโจมตีนั้น จะใช้วิธีการหาทุกๆ รูปแบบการโจมตีที่สามารถเกิดขึ้นได้ ซึ่งเมื่อมองง่ายๆ ปัญหาดังกล่าวนี้ก็จะกลายเป็นปัญหาแบบการจับคู่กัน แล้วหาค่าที่มากที่สุดจากการจับคู่ แต่เมื่อมองให้ลึกๆ แล้วปัญหาดังกล่าวจะเป็นปัญหาของ 8 ควีน แบบตรวจสอบแค่ 4 ทิศ หรือจะกลายเป็นตัวรุก (Rook) นั่นเอง ดังนี้



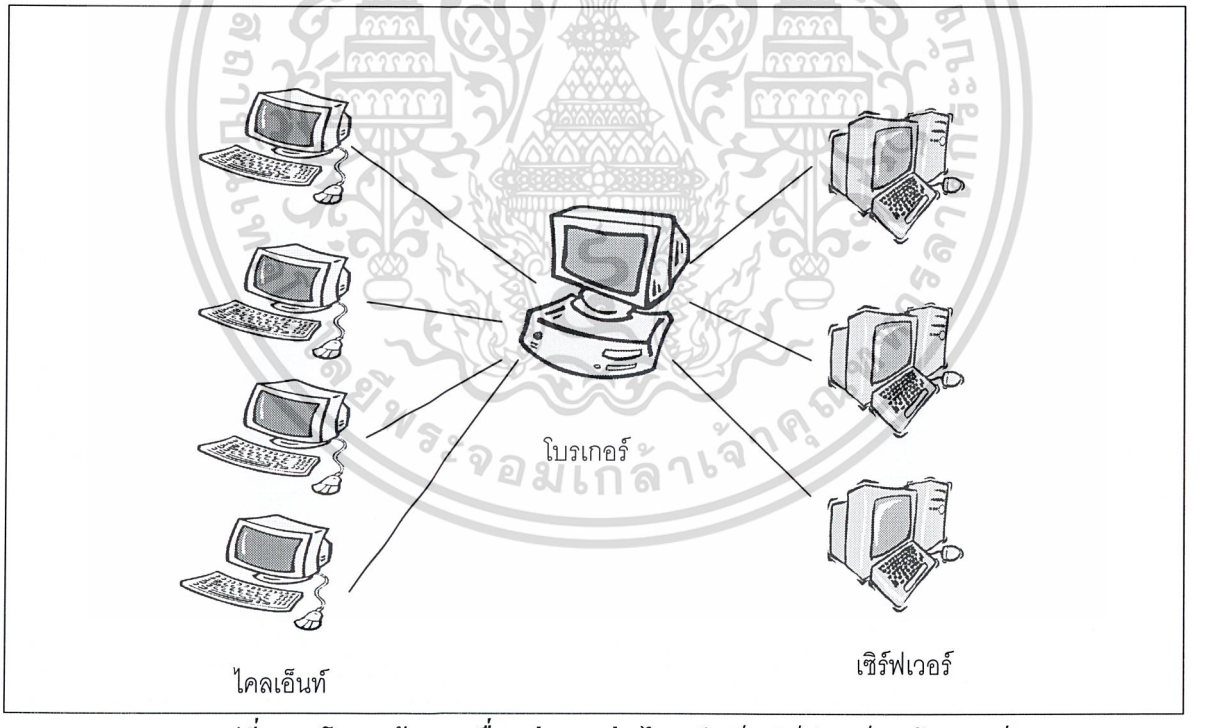
จากรูป เมื่อนำเอาการ์ดของเรามาวางตามแถว และนำการ์ดฝ่ายตรงข้ามมาวางตามคอลัมน์แล้ว จะได้ว่าเมื่อเราต้องการจับคู่โจมตีคู่ใดๆ ก็โดยการวางรุกลงไปที่ตารางช่องที่ตัดกันนั้นๆ หลังจากนี้หมายความว่าห้ามวางตัวรุกซ้ำแถว หรือ คอลัมน์เดิมอีก คือห้ามวางให้รุกกินกันได้นั่นเอง นอกจากนี้ ยังมีข้อกำหนดพิเศษเพิ่มขึ้นมาอีกคือในกรณีที่เห็นว่าวางได้ แต่ทำให้ค่าที่ออกมาติดลบ นั่นคือเมื่อโจมตีใส่ศัตรูแล้ว การ์ดเราแพ้ ก็ห้ามวางด้วย และไม่จำเป็นต้องวางครบตัว นั่นคือ หากโจมตีไม่ได้ก็ไม่ควรโจมตี ดีกว่าโจมตีแล้วเราแพ้

หลังจากนี้หากการ์ดศัตรูหมดแล้ว แต่การ์ดเรายังเหลือก็สั่งโจมตีใส่ผู้เล่นโดยตรงตามไปด้วย จะทำให้ขั้นตอนการโจมตีสมบูรณ์

5.3 การออกแบบการสื่อสารผ่านเครือข่ายโดยใช้วินซ็อค

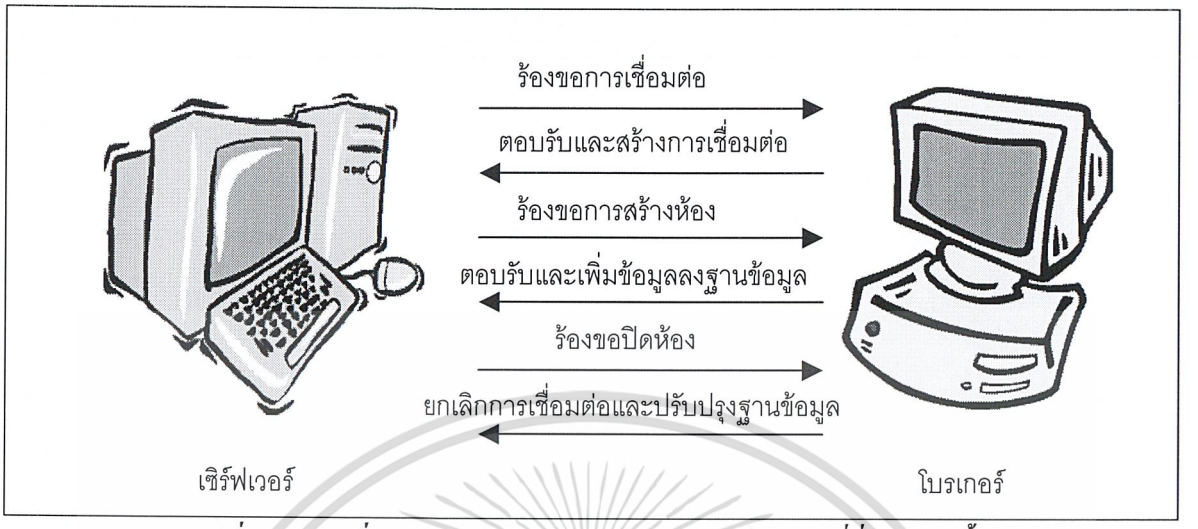
การสื่อสารผ่านเครือข่ายโดยใช้วินซ็อคสำหรับโปรแกรมนี้แบ่งออกเป็น 3 ส่วนคือ โบรเกอร์ (Broker), เซิร์ฟเวอร์และไคลเอ็นท์ โดยโครงสร้างการเชื่อมต่อระหว่างไคลเอ็นท์, เซิร์ฟเวอร์และโบรเกอร์จะมีลักษณะดังรูปที่ 6.1 โดยในแต่ละกลุ่มมีหน้าที่และการทำงานดังต่อไปนี้

- โบรเกอร์ เป็นตัวกลางระหว่างไคลเอ็นท์และเซิร์ฟเวอร์ มีหน้าที่เก็บและแจกจ่ายหมายเลขไอพีของเซิร์ฟเวอร์และชื่อห้องที่เซิร์ฟเวอร์นั้นตั้งให้แก่ไคลเอ็นท์ที่ต้องการเล่นในห้องของเซิร์ฟเวอร์นั้นๆ
- เครื่องที่ติดต่อกับโบรเกอร์เพื่อสร้างห้องเรียกว่า เซิร์ฟเวอร์ โดยที่หน้าที่หลักของเซิร์ฟเวอร์คือการกำหนดเทิร์นให้กับผู้เล่นแต่ละคนที่เข้ามาร่วมเล่นในห้องเท่านั้น และในแต่ละเซิร์ฟเวอร์จะมีการกำหนดให้มีผู้เล่นสูงสุดได้เพียง 2 คนเท่านั้น
- เครื่องที่ติดต่อกับเซิร์ฟเวอร์เพื่อเข้าร่วมเล่นเกมนั้นเรียกว่า ไคลเอ็นท์

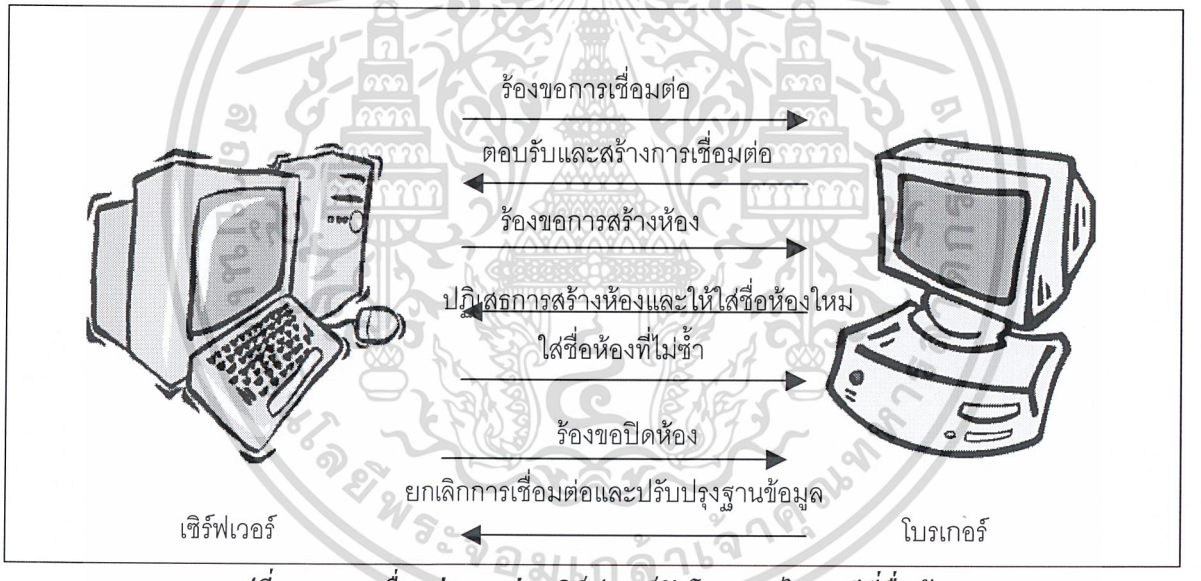


รูปที่ 5.10 โครงสร้างการเชื่อมต่อระหว่างไคลเอ็นท์, เซิร์ฟเวอร์และโบรเกอร์

5.3.1 ขั้นตอนการติดต่อระหว่างเซิร์ฟเวอร์กับโบรเกอร์



รูปที่ 5.11 การเชื่อมต่อระหว่างเซิร์ฟเวอร์กับโบรเกอร์ในกรณีที่ชื่อห้องไม่ซ้ำ



รูปที่ 5.12 การเชื่อมต่อระหว่างเซิร์ฟเวอร์กับโบรเกอร์ในกรณีที่ชื่อห้องที่เซิร์ฟเวอร์ต้องการสร้างใหม่นั้นซ้ำกับที่เปิดอยู่แล้ว

เซิร์ฟเวอร์จะร้องขอการติดต่อกับโบรเกอร์ในกรณีที่ต้องการร้องขอการสร้างห้องใหม่ โดยการติดต่อจะมีขั้นตอนที่แสดงในรูปที่ 5.10 ดังนี้

ขั้นที่ 1 เซิร์ฟเวอร์ร้องขอการเชื่อมต่อกับโบรเกอร์

ขั้นที่ 2 โบรเกอร์ตอบรับพร้อมสร้างการเชื่อมต่อ

ขั้นที่ 3 เซิร์ฟเวอร์ส่งข้อความร้องขอสร้างห้องไปยังโบรเกอร์โดยมีส่วนประกอบดังนี้

ชื่อห้อง	หมายเลขไอพี

ข้อความร้องขอสร้างห้องที่ส่งไปยังโบรเกอร์ประกบด้วย 2 ส่วนคือ

1. ชื่อห้องที่ต้องการสร้าง
2. หมายเลขไอพีของเครื่องเซิร์ฟเวอร์ที่ร้องขอสร้างห้อง

ขั้นที่ 4 โบรเกอร์ทำการตรวจสอบว่าชื่อห้องที่เซิร์ฟเวอร์ส่งมาเพื่อร้องขอการสร้างห้องนั้นซ้ำกับชื่อห้องที่มีอยู่หรือไม่หากไม่ซ้ำก็ส่งข้อความอนุญาตกลับไปยังเซิร์ฟเวอร์พร้อมกับเก็บข้อมูลการสร้างห้องเอาไว้ในฐานข้อมูล แต่ถ้าหากว่าชื่อห้องที่ต้องการสร้างใหม่นั้นซ้ำกับที่มีอยู่แล้ว ทางโบรเกอร์ก็จะส่งข้อความแจ้งความผิดพลาดกลับมา เพื่อให้เซิร์ฟเวอร์ตั้งชื่อห้องใหม่อีกจนกว่าชื่อจะไม่ซ้ำ

ขั้นที่ 5 เมื่อได้รับข้อความอนุญาตให้สร้างห้องแล้วเซิร์ฟเวอร์ก็จะรอการร้องขอการเชื่อมต่อจากไคลเอ็นท์ที่ต้องการเข้ามาเล่นในห้องที่สร้างไว้ แล้วจึงเริ่มเล่นเกมเมื่อมีไคลเอ็นท์ตามจำนวนที่ต้องการ (ที่กำหนดไว้จำนวนผู้เล่นสูงสุดคือ 2 คน)

ขั้นที่ 6 หากเซิร์ฟเวอร์ต้องการปิดห้องก็จะส่งข้อความยกเลิกการเชื่อมต่อ โดยมีโครงสร้างดังนี้

"Disconnect"	หมายเลขไอพี

ข้อความร้องขอปิดห้องที่ส่งไปยังโบรเกอร์ประกบด้วย 2 ส่วนคือ

1. ข้อความ "Disconnect"
2. หมายเลขไอพีของเครื่องเซิร์ฟเวอร์ที่ร้องขอปิดห้อง

ขั้นที่ 7 เมื่อโบรเกอร์ได้รับข้อความปิดห้องจากเซิร์ฟเวอร์ก็จะทำการลบชื่อห้องและหมายเลขไอพีของเซิร์ฟเวอร์ที่สร้างห้องนั้น แล้วส่งข้อความตอบรับกลับไปยังเซิร์ฟเวอร์

5.3.2 ขั้นตอนการติดต่อระหว่างไคลเอ็นท์กับโบรเกอร์

เมื่อไคลเอ็นท์ต้องการร่วมเล่นเกมกับห้องที่เซิร์ฟเวอร์ต่างๆเปิดไว้ จะทำการเชื่อมต่อกับโบรเกอร์และเซิร์ฟเวอร์ตามขั้นตอนดังแสดงในรูปที่ 5.11 ดังนี้

ขั้นที่ 1 ไคลเอ็นท์ร้องขอการเชื่อมต่อกับโบรเกอร์

ขั้นที่ 2 โบรเกอร์ตอบรับพร้อมสร้างการเชื่อมต่อ

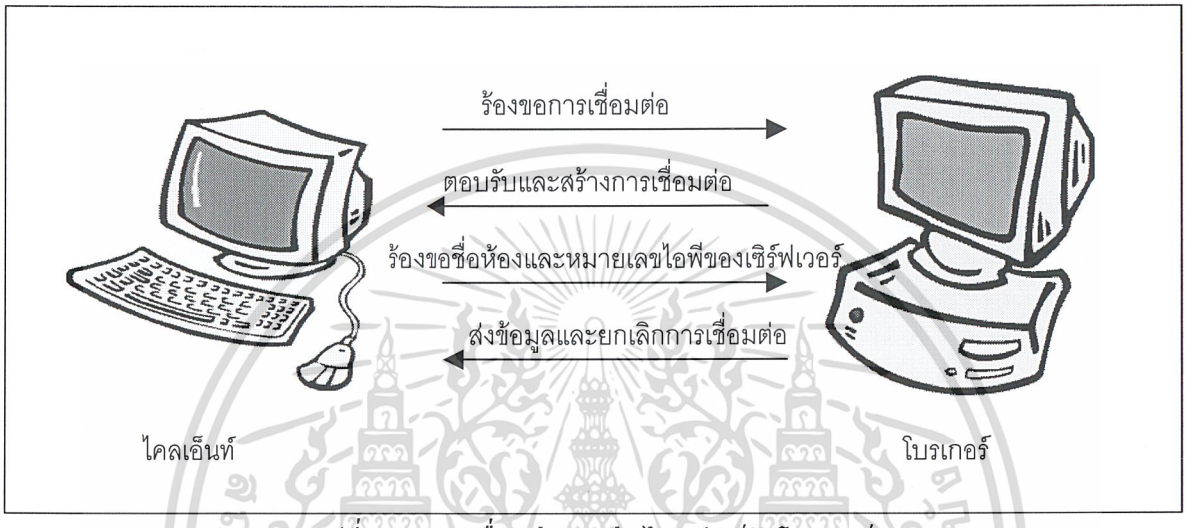
ขั้นที่ 3 ไคลเอ็นท์ส่งข้อมูลการร้องขอหมายเลขไอพีและชื่อห้องมีเปิดอยู่มีโครงสร้างดังนี้

"Request"	หมายเลขไอพี

ข้อความร้องขอหมายเลขไอพีของเซิร์ฟเวอร์และชื่อห้องที่เปิดอยู่ประกอบด้วย 2 ส่วนคือ

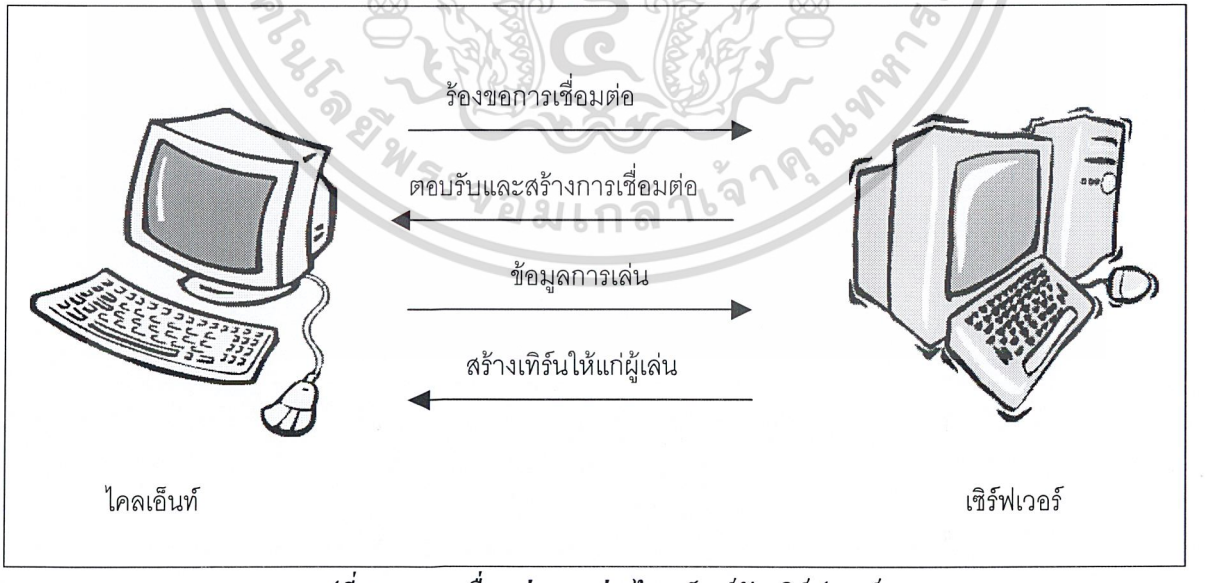
1. ข้อความ “Request”
2. หมายเลขไอพีของเครื่องไคลเอ็นท์ที่ร้องขอ

ขั้นที่ 4 เมื่อโบรเกอร์ได้รับข้อความร้องขอหมายเลขไอพีและชื่อห้องที่เปิดอยู่ จะทำการส่งข้อมูลหมายเลขไอพีและชื่อห้องที่เก็บไว้ในฐานข้อมูลส่งไปยังไคลเอ็นท์ แล้วยกเลิกการเชื่อมต่อ



รูปที่ 5.13 การเชื่อมต่อระหว่างไคลเอ็นท์กับโบรเกอร์

5.3.3 ขั้นตอนการติดต่อระหว่างไคลเอ็นท์กับเซิร์ฟเวอร์

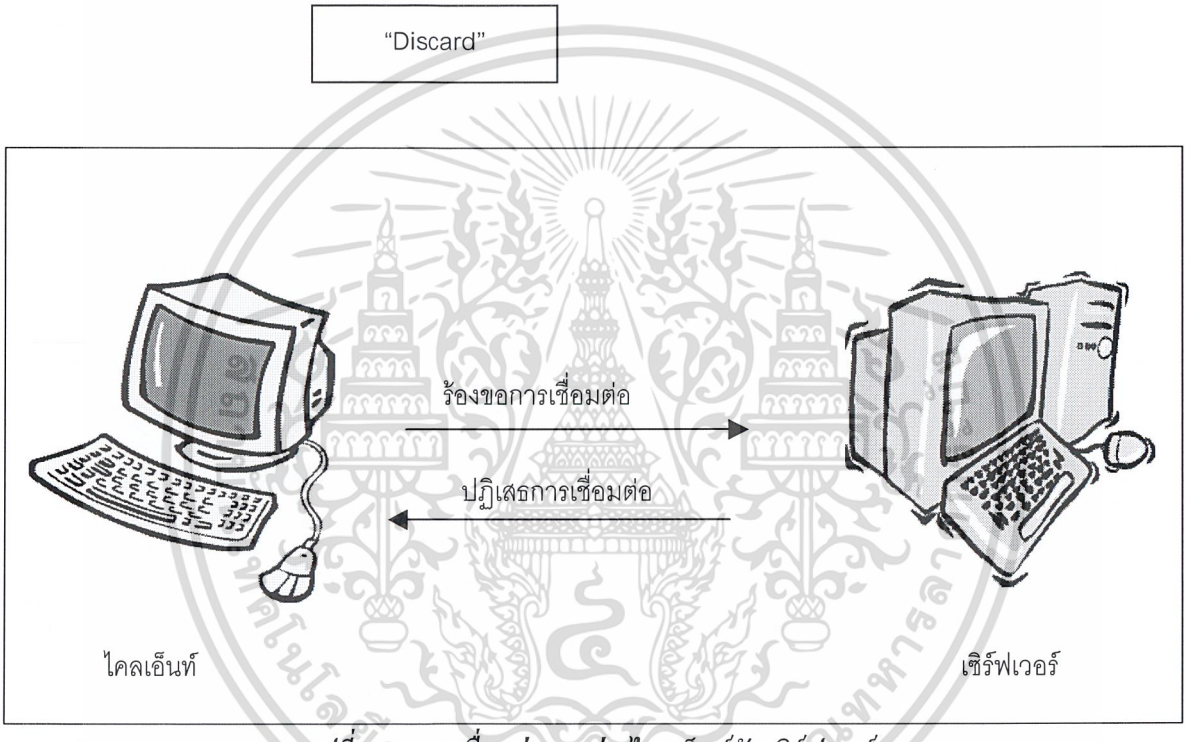


รูปที่ 5.14 การเชื่อมต่อระหว่างไคลเอ็นท์กับเซิร์ฟเวอร์
ในกรณีที่จำนวนผู้เล่นไม่ครบตามที่กำหนด (2 คน)

เมื่อไคลเอ็นท์ได้ทำการร้องขอหมายเลขไอพีและชื่อห้องที่เปิดอยู่มาจากโบรเกอร์แล้ว จะทำการร้องขอการเชื่อมต่อไปยังเครื่องเซิร์ฟเวอร์ที่ต้องการ โดยมีขั้นตอนดังแสดงในรูปที่ 5.12 ดังนี้

ขั้นที่ 1 ไคลเอ็นท์ส่งข้อความร้องขอการเชื่อมต่อไปยังเซิร์ฟเวอร์ โดยข้อความดังกล่าวมีโครงสร้างเช่นเดียวกับโครงสร้างข้อความการร้องขอการเชื่อมต่อที่ไคลเอ็นท์ส่งไปยังโบรเกอร์

ขั้นที่ 2 เซิร์ฟเวอร์จะตรวจสอบว่าจำนวนไคลเอ็นท์ที่เชื่อมต่ออยู่ในขณะนั้นอยู่ในจำนวนที่จำกัดหรือไม่ หากครบแล้วก็จะส่งข้อความปฏิเสธไปยังไคลเอ็นท์นั้น โดยข้อความปฏิเสธมีโครงสร้างดังต่อไปนี้



รูปที่ 5.15 การเชื่อมต่อระหว่างไคลเอ็นท์กับเซิร์ฟเวอร์ ในกรณีที่จำนวนผู้เล่นเกินตามที่กำหนดไว้ (มากกว่า 2 คน)

ข้อความปฏิเสธประกอบด้วยข้อความ "Discard" เพียงอย่างเดียว แล้วจึงยกเลิกการเชื่อมต่อ แต่ถ้าหากจำนวนไคลเอ็นท์ที่ติดอยู่ยังไม่ครบตามที่กำหนดเซิร์ฟเวอร์ก็จะส่งข้อความตอบรับไปยังไคลเอ็นท์ โดยข้อความตอบรับมีโครงสร้างดังต่อไปนี้

"Accept"

โดยข้อความตอบรับที่ไคลเอ็นท์ได้รับนั้นจะประกอบด้วยข้อความ "Accept" เพียงอย่างเดียว จากนั้นเซิร์ฟเวอร์ก็จะสร้างการเชื่อมต่อกับไคลเอ็นท์ตัวนั้น เพื่อเล่นเกม

ขั้นที่ 3 ในระหว่างการเล่นเกมนั้น ทางไคลเอ็นท์และเซิร์ฟเวอร์จะส่งข้อมูลการลงการ์ดไปยังฝ่ายตรงข้าม โดยข้อมูลที่ส่งไปนั้นจะเป็นรายละเอียดของการ์ดใบนั้นๆและจะได้รับข้อมูลของการ์ดทั้งหมดในสนามกลับมา โดยในส่วนนี้เซิร์ฟเวอร์จะมีหน้าที่เพียงจัดเทิร์นการเล่นให้กับผู้เล่นแต่ละคนเท่านั้น

บทที่ 6

แนวทางการพัฒนา

6.1 แนวทางการพัฒนา

การพัฒนาโปรแกรมเกมการ์ดบนเน็ตเวิร์ก นี้ใช้ไมโครซอฟต์ วิชวลเบสิกเป็นเครื่องมือในการเขียนโปรแกรม เนื่องจากใช้งานได้ง่าย มีประสิทธิภาพในระดับที่น่าพอใจ ส่วนในเรื่องของความเร็วในการประมวลผลแล้ว แม้จะช้ากว่าการใช้ภาษาอื่นๆ บางภาษา เช่น วิชวลซี หรือเดลไฟ แต่เนื่องจากเกมเป็นเกมที่ไม่เน้นภาพที่มีการเคลื่อนไหวตลอดเวลา เกมประเภทเกมการ์ดจะเป็นเกมที่เล่นเป็นเทิร์นๆ ดังนั้น ความช้า ความเร็วของการประมวลผลจึงไม่ส่งผลต่อตัวเกมมากนัก

การพัฒนาจะมีการเขียนแบ่งเป็นฟอร์มๆ ไป เนื่องจากวิชวลเบสิกนั้นใช้การเขียนแบบแบ่งเป็นฟอร์ม แต่ต้องมีการแตกฟอร์มออกมาเป็นหลายๆ ฟอร์ม ทั้งนี้เนื่องจากการเก็บโปรแกรมทั้งหมดให้ทำงานบนฟอร์มๆ เดียวนั้นจะเขียนโปรแกรมได้ยาก อีกทั้งยังยากต่อการควบคุมอีกด้วย

ฟอร์มเริ่มต้นที่ควรทำก่อนก็คือ ฟอร์มที่ใช้เล่นเกมจริงๆ คือ FormGame นั่นเอง ซึ่งเป็นการรวมเอาในส่วนของส่วนติดต่อกับผู้ใช้ การคิดคำนวณการเล่นโดยคอมพิวเตอร์ และส่วนที่ทำหน้าที่ดำเนินเกม

ฟอร์มอื่นๆ นอกจากนี้ถือเป็นฟอร์มที่ทำหน้าที่ช่วยให้โปรแกรมสมบูรณ์มากขึ้น

สำหรับความฉลาดของคอมพิวเตอร์นั้น ในส่วนของการลงไพ่จะใช้วิธีการแบบค้นหาลงไปในทรีของข้อมูล ผสมกับการใช้แพทเทิร์น สำหรับการลงไพ่ลงไปในสนามจะเป็นดังนี้

- หากมีการ์ดเวทมนตร์อยู่บนมือ จะต้องใช้เสมอในเทิร์นนั้นๆ
- การ์ดเวทมนตร์ประเภททำลายจะถูกใช้ก่อนที่จะลงไพ่ ส่วนการ์ดเวทมนตร์ประเภทอื่นๆ จะใช้หลังจากลงไพ่ไปแล้ว
- การลงการ์ดสิ่งมีชีวิต ขึ้นแรกจะนำเอาค่าบูชายัญของการ์ดระดับล่างทั้งสนามมารวมกันเสียก่อน แล้วนำลงไปหาในการ์ดบนมือเรา ว่ามีตัวไหนที่มีค่าเรียกน้อยกว่า ค่ารวมนี้บ้าง แล้วเลือกเอาตัวที่มากที่สุดลงไปในสนาม
- ในกรณีที่การ์ดที่จะลงไปในสนามนั้นเป็นการ์ดระดับต่ำ ไม่ต้องใช้ค่าเรียก ก็จะลงไปเลย แต่ในกรณีที่เป็นการ์ดที่ต้องใช้ค่าเรียกก็จะมีการเลือกเอาการ์ดในสนามที่มีค่าบูชายัญมากที่สุดไปบูชายัญก่อน แล้วค่อยเอาตัวที่มีค่าบูชายัญต่ำลงมาไปบูชายัญต่อ จนกว่าจะได้ค่าบูชายัญมากกว่าค่าเรียกที่ต้องการ แล้วจึงลงการ์ดใบดังกล่าวลงไป
- หลังจากลงการ์ดสิ่งมีชีวิตไปแล้วก็จะเลือกการ์ดเวทมนตร์บนมือทั้งหมดออกมาใช้
- สำหรับการโจมตีใส่ฝ่ายตรงข้ามจะใช้วิธีการคิดดังนี้
- จะนำไพ่ของคอมพิวเตอร์ และไพ่ของผู้ใช้มาทำการหาวิธีที่ดีที่สุดมาเลือกวิธีโจมตีเพื่อจะก่อให้เกิดเหตุการณ์ที่ดีที่สุดเท่าที่เป็นไปได้

- ในส่วนของการเล่นหลายคนใช้วิธีการเปิดห้อง โดยเขียนตัวหนึ่งเป็นโบรเกอร์ (Broker) ไว้เมื่อเครื่องใดต้องการเปิดห้องก็จะติดต่อไปยังโบรเกอร์ โบรเกอร์จะทำการเก็บค่าไว้ว่าเป็นเซิร์ฟเวอร์เปิดห้องมา
- ส่วนเมื่อจะมีผู้ใช้คนอื่นต้องการเข้าร่วมในห้องที่เปิดไว้แล้วก็จะมีการส่งไปขอข้อมูลจากโบรเกอร์ โบรเกอร์จะส่งข้อมูลของเซิร์ฟเวอร์ รวมทั้งห้องที่มีการเปิดอยู่ไว้กลับคืนไปให้กับผู้ใช้ที่ขอมา ทั้งหมด
- หลังจากนี้การติดต่อกันระหว่างไคลเอ็นต์ และเซิร์ฟเวอร์จะไม่มีการผ่านไปยังโบรเกอร์อีกแล้ว จะเป็นการส่งข้อมูลโดยตรงระหว่างไคลเอ็นต์ และเซิร์ฟเวอร์เลย เนื่องจากทั้ง 2 ฝั่งจะมีหมายเลขตำแหน่งของอีกฝั่งแล้ว
- เมื่อเซิร์ฟเวอร์ต้องการปิดห้องก็จะมีการส่งข้อมูลไปให้กับโบรเกอร์เพื่อบอกโบรเกอร์ว่าห้องปิดแล้ว โบรเกอร์ก็จะลบข้อมูลของเครื่องนั้นทิ้งไป
- การส่งผ่านข้อมูลระหว่างกันทั้งหมด ไม่ว่าจะเป็นจากเซิร์ฟเวอร์ หรือไคลเอ็นต์ กับโบรเกอร์ หรือระหว่างไคลเอ็นต์กับเซิร์ฟเวอร์จะใช้การส่งแบบเป็นตัวกลุ่มอักษรไปทั้งหมด

6.2 ปัญหา และอุปสรรคในการดำเนินโครงงาน

1. เนื่องจากครั้งแรกที่กำหนดว่าจะใช้โปรแกรมวิชวลซีกับไดเร็กเอ็กซ์ เป็นเครื่องมือนั้นทำได้ยาก และผลงานที่ได้ก็ไม่แตกต่างจากการใช้วิชวลเบสิกมากนักดังนั้นจึงมีการเปลี่ยนมาใช้วิชวลเบสิก
2. การเขียนโปรแกรมให้ความฉลาดกับคอมพิวเตอร์ ด้วยวิธีการค้นหาแบบแตกทรีนั้น ทำได้ยาก
3. การเขียนโปรแกรมเพื่อส่งผ่านเครือข่ายนั้นมีปัจจัยภายนอกอื่นๆ มามีผลกระทบมากมาย ซึ่งบางปัจจัย ไม่สามารถคาดเดาได้เลย
4. การเขียนโปรแกรมด้วยวิชวลเบสิกนั้นเขียนโปรแกรมได้ง่ายจริง แต่ควบคุมผลที่เกิดขึ้นได้ค่อนข้างยากกว่าเครื่องมืออื่นๆ มาก

บทที่ 7
บทสรุปและแนวทางการพัฒนาต่อ

7.1 บทสรุป

โครงงานนี้เป็นโครงงานประเภทเกม ถึงแม้จะดูว่าไม่ค่อยจะมีประโยชน์ต่อบุคคลทั่วไปเท่าไรนัก แต่โครงงานประเภทเกมนี้ก็ถือเป็นก้าวแรกที่ช่วยให้เกิดมีการพัฒนาการเขียนโปรแกรมประเภทเกมขึ้นในประเทศไทย เนื่องจากปัจจุบันนี้เกมส่วนมากที่ผู้คนเล่นนั้น เป็นเกมที่เขียนโดยบริษัท หรือผู้พัฒนาจากต่างประเทศทั้งสิ้น ซึ่งในปีหนึ่งๆ นั้น ประเทศไทยได้มีการนำเข้าโปรแกรมประเภทเกมเป็นกลุ่มหลักๆ ของโปรแกรมทุกประเภท ซึ่งต่อไปนั้นถ้าในประเทศมีการเขียนเกมกันมากขึ้นนั้นจะทำให้ระดับคุณภาพของเกมที่เขียนโดยคนไทยสูงขึ้น จนอาจจะสามารถเทียบกับต่างประเทศได้ ซึ่งเมื่อถึงจุดนั้นแล้ว โปรแกรมเกมอาจจะเป็นหนึ่งในสินค้าส่งออก ซึ่งเป็นการทำรายได้ให้แก่ประเทศด้วย

โดยเฉพาะเกมในปัจจุบันได้มีการเข้ามาสู่ในยุคของเกมแบบหลายผู้เล่น ซึ่งเกมประเภทนี้ปัจจุบันยังไม่ได้รับความสนใจจากนักเขียนโปรแกรมไทยเท่าที่ควร ทั้งนี้เนื่องจากการเขียนเกมผ่านเครือข่ายนั้นมีความยุ่งยากซับซ้อนค่อนข้างมาก ดังนั้นโครงงานนี้จึงเป็นเสมือนการเริ่มต้นของการเขียนเกมประเภทเกมการ์ดบนเน็ตเวิร์ก

7.2 แนวทางการพัฒนาต่อ

1. เพิ่มจำนวนการ์ด เพื่อให้มีการ์ดหลายหลากชนิดมากขึ้น
2. เพิ่มประเภท และชนิดของการ์ดเวทมนตร์ เพื่อให้เกมมีทางเลือกมากขึ้น
3. ปรับปรุงในเรื่องของความฉลาดของคอมพิวเตอร์ให้ใช้วิธีที่ทำงานได้ดี และเร็วมากยิ่งขึ้น
4. ปรับปรุงให้เล่นผ่านอินเตอร์เน็ตได้

บรรณานุกรม

หนังสืออ้างอิง

1. Adrian Perez, with Don Royer, *"Advanced 3D Programming Using MS DirectX 7.0"*, Wordware Publishing,inc.
2. Mickey Kawick,*"Real-Time Strategy Game Programming Using MS DirectX 6.0"*, Wordware Publishing,inc.
3. Andre Lamothe,*"Tricks of Windows Game Pregramming Gurus "*,SAMS
4. Michael J. Norton, *"Spells of Fury"*, White Group Press
5. Peter J.Kovach,ชัยดำรงค์ อุทิรัมย์ แปลและเรียบเรียง,*"Direct 3D พลังพัฒนาแห่งเกมสามมิติ"*, สำนักพิมพ์ สามย่าน.com
6. นิรุท อำนวยศิลป์,*"คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับเพื่อการใช้งานจริง"*, พิมพ์ครั้งที่ 4, บริษัท ซัคเซส มีเดีย จำกัด

เว็บไซต์อ้างอิง

1. http::://www.ThaiDev.com
2. http::://www.ThaiGameDevX.com
3. http::://www.CProgramming.com
4. http::://www.ScorpionCity.com
5. http::://www.Microsoft.com/DirectX/