การพัฒนาเกมคอมพิวเตอร์ ทิคแทคโท โดยใช้เทคนิคปัญญาประดิษฐ์

DEVELOPMENT OF TIC TAC TOE COMPUTER GAME USING AI TECHNIQUE

พฤทธ์ เชาวรัตน์
อภิชัย โคตะนันท์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2543

DEVELOPMENT OF TIC TAC TOE COMPUTER GAME USING AI TECHNIQUE

PHURT CHAOVARATANA
APICHAI KOTANUN

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHTEMATICS AND COMPUTER SCIENCES
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2000

หัวข้อปัญหาพิเศษ การพัฒนาเกมคอมพิวเตอร์ ทิคแทคโท โดยใช้เทคนิคปัญญาประดิษฐ์
DEVELOPMENT OF TIC TAC TOE COMPUTER GAME USING AI TECHNIQUE

ชื่อนักศึกษา นายพฤทธ์ เชาวรัตน์ 40056054
นายอภิชัย โคตะนันท์ 40056108

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

คณะ วิทยาศาสตร์

ปีการศึกษา 2543

อาจารย์ที่ปรึกษา ดร.นพพร โชติกกำธร
ผู้ช่วยศาสตราจารย์ธีรวัฒน์ ประกอบผล

ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2543

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผู้ช่วยศาสตราจารย์กฤษฎา ไตรสุรัตน์	
กรรมการ	อาจารย์วีระชัย ตันยะสิทธิ์	
กรรมการและอาจารย์ที่ปรึกษา	ดร.นพพร โชติกกำธร	
กรรมการและอาจารย์ที่ปรึกษา	ผู้ช่วยศาสตราจารย์ธีรวัฒน์ ประกอบผล	

(ผู้ช่วยศาสตราจารย์ไพรบูลย์ พันธรักษ์พงษ์)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ การพัฒนาเกมคอมพิวเตอร์ ทิคแทคโท โดยใช้เทคนิคปัญญาประดิษฐ์
DEVELOPMENT OF TIC TAC TOE COMPUTER GAME USING AI TECHNIQUE

ชื่อนักศึกษา นายพฤทธิ์ เชาวรัตน์ 40056054
นายอภิชัย โคตะนันท์ 40056108

ปริญญา วิทยาศาสตรบัณฑิต

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

คณะ วิทยาศาสตร์

ปีการศึกษา 2543

อาจารย์ที่ปรึกษา ดร.นพพร โชติกกำธร
ผู้ช่วยศาสตราจารย์ธีรวัฒน์ ประกอบผล

บทคัดย่อ

ปัจจุบันในวงการอุตสาหกรรมการผลิตโปรแกรมเกมคอมพิวเตอร์ ได้มีการปรับตัวเพิ่มขึ้นอย่างรวดเร็ว และมีผู้ผลิตมากมายในตลาด รายงานฉบับนี้เป็นผลจากการศึกษาโครงการสร้างโปรแกรมเกมคอมพิวเตอร์ โดยทางคณะศึกษาได้เลือกเกมทิคแทคโทซึ่งเป็นเกมประเภทที่ต้องใช้ความคิดเพื่อแก้ไขสถานการณ์มาพัฒนา โดยได้นำทฤษฎีทางปัญญาประดิษฐ์คือ ทฤษฎี Minimax และ Alpha-Beta Pruning ตลอดจนเทคนิค Heuristic Search มาประยุกต์ใช้ เพื่อให้เครื่องคอมพิวเตอร์สามารถเล่นตอบโต้กับผู้เล่นได้ โดยโปรแกรมเกมที่ศึกษาและพัฒนาขึ้น ผู้เล่นสามารถกำหนดขนาดของตารางได้ตั้งแต่ขนาด 3 x 3 ถึง 9 x 9 และเงื่อนไขในการชนะตามความเหมาะสมของแต่ละขนาดตาราง สำหรับในส่วนของการพัฒนาใช้โปรแกรม Visual Basic 6.0 เป็นเครื่องมือในการโปรแกรม ซึ่งผลของการพัฒนาทำให้เครื่องคอมพิวเตอร์สามารถเล่นตอบโต้กับผู้เล่นได้ดี มีผลการแพ้ชนะเท่าๆ กัน

Special Project Title	DEVELOPMENT OF TIC TAC TOE COMPUTER GAME USING AI TECHNIQUE		
Students	Mr.Phurt	Chaovaratana	40056054
	Mr.Apichai	Kotanun	40056108
Degree	Bachelor's Degree of Sceince		
Department	Mathematics and Computer Sciences, Faculty of Science		
Programme	Computer Sciences		
Faculty	Science		
Academic Year	2000		
Special Project Advisor	Lecturer Dr. Nopporn Chotikakamthorn		
	Assistant Professor Teerawat Prakorbphol		

ABSTRACT

Currently, software game industry is fast expanding. More competitors appear in the market. This report is a result of computer's game program development study. We have chosen Tic Tac Toe strategy game in our study. Artificial Intelligent techniques such as Minimax, Alpha-Beta Pruning and Heuristic Search have been studied and used in game development, to allow a computer to play against a user. In our program, a user can choose between 3 x 3 to 9 x 9 table sizes, and set the game's winning rule according to the size of the table. This program was developed by Visual Basic 6.0. The complete program was found to be able to play competitively with a user.

กิตติกรรมประกาศ

กราบขอบพระคุณบุพการี ผู้อุปการะ ผู้ปกครอง และท่านอาจารย์ที่ปรึกษา ที่ได้ทำให้เกิดความสำเร็จ ณ วันนี้ เกิดความภาคภูมิใจกับงานที่ได้ทำการสร้างสรรค์ขึ้นด้วยตัวของนักศึกษาเอง ด้วยคำปรึกษาชี้แนะแนวทางซึ่งก่อให้เกิดประโยชน์ แนวความคิดและเกิดความสำเร็จรุร่วง

ขอบพระคุณท่านอาจารย์ดร.นพพร โชติกกำธร และท่านอาจารย์ธีรวัฒน์ ประกอบผล ซึ่งให้คำชี้แนะ แนะนำ ให้ความรักเสมือนยิ่งนักศึกษาที่เป็นที่รักอย่างที่สุด ก่อเกิดความภาคภูมิใจแก่เรานักศึกษา ดวยได้ก่อเกิดกำลังใจเพื่อผ่านปัญหานานาที่เกิดขึ้นไม่ว่าปัญหาเพียงเล็กๆ น้อยๆ จนถึงปัญหาที่ทำให้เกิดความย่อท้ออย่างหาที่เปรียบมิได้ เรานักศึกษาจะหาลืมไม่แม้ว่าสิ่งที่ประสบปัญหานานา ใหญ่น้อยเท่าใดก็เป็นเพียงธุลีที่หามิเจอ ด้วยพลังจากท่านอาจารย์ที่ปรึกษา แรงกำลังใจ และคำชี้แนะที่หาที่ใดคงยากเท่า

ขอบพระคุณรุ่นพี่ที่ได้ให้คำแนะนำด้วยเสมือนหนึ่งดวงไฟในยามมืดสนิท เวลามองไม่เห็นทางออก

ขอบคุณเพื่อนร่วมกลุ่ม ซึ่งถ้าไม่มีกันงานชิ้นนี้คงไม่มีวันสำเร็จได้ เพื่อนร่วมกลุ่มคอยบอกจุดบกพร่องของงาน บอกขั้นตอนการทำงานเพื่อให้ทำงานได้อย่างมีคุณภาพและแม้บางครั้งจะไม่ค่อยเข้าใจในงานที่จะต้องทำแต่ก็พยายามทำงานนั้นจนสำเร็จรุร่วงและเป็นการทำงานที่ไม่มีที่ติดและคิดว่าเพื่อนร่วมกลุ่มคงภูมิใจในงานที่ได้ทำแม้ว่ามันจะยากลำบากเพียงใดแต่ก็ทำได้อย่างดี

ขอบคุณบุคคลที่ช่วยเหลือทุกๆ คนทางอินเทอร์เน็ตซึ่งเป็นชาวต่างชาติที่ไม่เคยรู้จักกันแต่ก็ได้ให้ความรู้ความคิดในการสร้างผลงานชิ้นนี้ขึ้นมา เป็นที่ปรึกษาในเชิงวิชาการอย่างดีมาก

คณะผู้จัดทำ

มีนาคม 2543

สารบัญ

หน้า

สารบัญตาราง

สารบัญภาพ

บทที่1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันได้เกิดอุตสาหกรรมเกมอย่างแพร่หลายและพัฒนาไปอย่างรวดเร็ว ซึ่งในขณะนี้ได้มีการแข่งขันกันในเชิงธุรกิจของผู้ผลิตหลายราย ทั้งทางด้านแนวความคิดจิตนาการของเกมตามค่ายต่างๆ วิธีการเล่นที่ทำความเข้าใจง่าย และมัลติมีเดียต่างๆ ที่ช่วยให้เกมตามค่ายต่างๆ มีความหน้าสนใจและผู้บริโภคจะตัดสินใจในการซื้อไม่ยาก แล้วยิ่งกับค่ายที่เคยสร้างสรรค์เกมที่มีความนิยม ผู้บริโภคจะคอยติดตามเกมจากค่ายนั้นๆ ตลอดซึ่งเหมือนกับว่าถ้าค่ายใดสร้างสรรค์เกมในครั้งแรกได้เป็นที่นิยมแล้วภาคต่อไปของเกมนั้นๆ จะถูกมองเป็นพิเศษจากผู้บริโภค ดังนั้นการศึกษาวิธี หลักการต่างๆในการสร้างเกมจึงเป็นสิ่งที่น่าสนใจยิ่ง ผู้ศึกษาจะได้รับความรู้ ความเข้าใจ แนวความคิด จิตนาการสร้างสรรค์ และวิธีการสร้างสรรค์เกมออกว่างจำหน่ายได้ โดยการศึกษาวิธีการติดต่อระหว่างตัวโปรแกรมกับตัวฮาร์ดแวร์ อินพุท เอาต์พุท การใช้เครื่องมือต่างๆ การสร้างรูปภาพเคลื่อนไหว อีกทั้งยังสามารถเขียนโปรแกรมเกมการติดต่อกับผู้ใช้ได้อย่างมีประสิทธิภาพ โดยผู้ศึกษาคาดหวังว่าจะเป็นประโยชน์ทั้งต่อตัวผู้ศึกษาเองและผู้ที่สนใจที่ต้องการศึกษาต่อ ผู้ศึกษามีความเชื่ออย่างยิ่งว่าการเขียนโปรแกรมเกมอีกไม่นานในประเทศคงจะมีค่ายผู้ผลิตต้องการบุคคลากรที่สามารถสร้างสรรค์แนวความคิดที่แวกแนวเพื่อทำการสร้างธุรกิจที่มั่งคงในประเทศอันเป็นที่รักยิ่งของผู้ศึกษา และหวังว่าการศึกษาของนักศึกษาจะช่วยเป็นพื้นฐานให้กับผู้ที่ต้องการค้นคว้าและพัฒนาการเขียนโปรแกรมเกมต่อไป

1.2 จุดมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1 เพื่อศึกษาระบบและวิธีการสร้างเกม

1.2.2 เพื่อเป็นแนวทางในการค้นคว้าสร้างสรรค์การเขียนโปรแกรมในลักษณะเกม

1.2.3 เพื่อเป็นแรงบันดาลใจและสร้างแนวความคิดให้เกิดการผลิตอุตสาหกรรมเกมในประเทศ

1.2.4 เพื่อศึกษาการเขียนโปรแกรมบนระบบปฏิบัติการวินโดวส์ ด้วยเทคโนโลยีการสร้างเกมโดยใช้แนวความคิดทางด้านปัญญาประดิษฐ์ เข้ามามีส่วนช่วยในการสร้าง ที่ทำให้เกมมีสีสันน่าเล่น กับภาษาที่ใช้ในการพัฒนาง่ายและเป็นที่นิยมในปัจจุบันคือ ภาษาโปรแกรมวิชวลเบสิก

1.2.5 สำหรับเกมนี้สร้างขึ้นเพื่อสร้างความเพลิดเพลินให้กับผู้เล่น

1.2.6 สร้างทักษะในการคิดที่จะเอาชนะคอมพิวเตอร์

1.2.7 สร้างเชาว์ปัญญาในการคิดที่จะเอาชนะเกมได้

1.2.8 สามารถนำเอาปัญญาประดิษฐ์ไปใช้ได้จริงกับเกมที่เลือกสร้างขึ้น

1.2.9 สามารถนำเอาเกมนี้เป็นแนวทางในการสร้างเกมเพื่อธุรกิจ ในอนาคตเพื่อแข่งขันกับเกมที่สร้างขึ้นในต่างประเทศ

1.3 ขอบเขตการศึกษา

1.3.1 การพัฒนาโปรแกรมทำได้โดยโปรแกรมไมโครซอฟท์วิชวลเบสิก 6.0

1.3.2 โปรแกรมเป็นโปรแกรมเกมทิคแทคโทมีผู้เล่น 1 คนสู้กับฝ่ายคอมพิวเตอร์

1.3.3 มีขนาดตารางให้เลือกตั้งแต่ 3 x 3 ถึง 9 x 9 และมีเงื่อนไขในการชนะตามความเหมาะสม

1.4 ข้อตกลงเบื้องต้น

1.4.1 ใช้ทฤษฎี Minimax with Alpha-Beta pruning สำหรับพัฒนาโปรแกรม

1.4.2 โปรแกรมสามารถรันได้บนระบบปฏิบัติการวินโดวส์ตั้งแต่เวอร์ชัน 95 ขึ้นไป

1.4.3 ต้องมีฮาร์ดแวร์อินพุทได้แก่เมาส์

1.5 ขั้นตอนการศึกษา

1.5.1 ศึกษาเกี่ยวกับระบบการสร้างเกม

เป็นขั้นตอนในการศึกษาวิธีการ และหลักการที่ใช้ในการสร้างเกม รวมทั้งเทคนิคต่างๆ ที่จะนำมาทำการสร้างสรรค์โปรแกรมเกมและพัฒนาระบบ

1.5.2 ศึกษาโปรแกรมและเครื่องมือต่างๆ

เป็นการศึกษาโปรแกรมและเครื่องมือที่จะนำไปใช้ในการพัฒนาในการเขียนโปรแกรมโดยโปรแกรมและเครื่องมือที่จะนำไปใช้ในการพัฒนาจะเน้นวิชวลเบสิก 6.0 และสร้างภาพหน้าตาของเกมให้หน้าเล่นจากโปรแกรมสร้างภาพ Photo Shop เป็นต้น

1.5.3 เก็บรวบรวมเอกสารและข้อมูลที่เกี่ยวข้องต่างๆ

เป็นการนำเอาเอกสารและข้อมูลที่เกี่ยวข้องมารวบรวม และใช้ประกอบการทำงาน โดยส่วนมากจะเป็นการรวบรวมจากหนังสือ ตำรา และเว็บไซต์ที่เกี่ยวข้อง รวมถึงข้อมูลที่ได้จากผู้มีความรู้ในแต่ละด้าน

1.5.4 วิเคราะห์และออกแบบเกม

ทำการวิเคราะห์และออกแบบเกม โดยการแบ่งออกเป็นส่วน ได้แก่ส่วนของการรับข้อมูล ส่วนประมวลผล ส่วนแสดงผล และสุดท้ายคือการทำให้ระบบเกิดประสิทธิภาพมากที่สุดในการเล่น

1.5.5 การพัฒนาโปรแกรม

เป็นขั้นตอนการเขียนโปรแกรมเกมตามขั้นตอนที่ได้ทำการค้นคว้าและศึกษามาตามขั้นตอนวิเคราะห์และออกแบบ

1.5.6 การทดสอบโปรแกรมและปรับปรุง

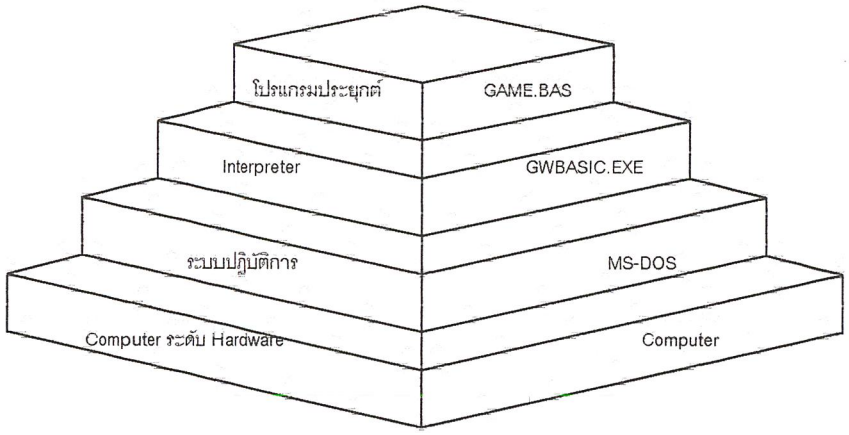
เป็นขั้นตอนการทดสอบโปรแกรมและปรับปรุงโปรแกรม

บทที่ 2
ทฤษฎีที่ใช้อ้างอิง

2.1 ทำความเข้าใจกับวิชวลเบสิก

2.1.1 การทำงานของอินเตอร์พรีเตอร์

วิชวลเบสิกเป็นภาษาโปรแกรมที่มีการทำงานแบบอินเตอร์พรีเตอร์ โปรแกรมดอสทั่วไปแต่เดิมจะมีตัวแปรภาษาเบื้องต้นที่ชื่อว่า Microsoft GWBasic ซึ่งเป็นตัวแปรภาษาหนึ่งของภาษาเบสิกให้เราได้ใช้ในการเขียนภาษาโปรแกรมอย่างง่ายอย่างเช่น Batch File เป็นต้น การเขียนโปรแกรมด้วยตัวแปรภาษาแบบอินเตอร์พรีเตอร์จะเกี่ยวข้องกับซอฟต์แวร์อยู่ 3 ระดับดังรูป 2.1



รูปที่ 2.1 แสดงระดับชั้นในการทำงานของโปรแกรมที่อาศัยอินเทอร์พรีเตอร์ในการแปรภาษา

กล่าวได้ว่าซอร์ฟแวร์ที่ตั้งอยู่บนฮาร์ดแวร์ดังรูป 2.1 ทำให้เครื่องคอมพิวเตอร์เป็นอุปกรณ์ที่มีศักยภาพในการทำงานสูง เราสามารถสังเกตได้ว่าเมื่อเราเปิดเครื่องคอมพิวเตอร์ขึ้นมาเพื่อใช้งานซอฟต์แวร์เหล่านี้จะถูกโหลดเข้าสู่คอมพิวเตอร์ตั้งแต่ข้างล่างสู่ข้างบนตามลำดับโดยขั้นแรก ระบบปฏิบัติการจะถูกโหลดเข้าสู่หน่วยความจำก่อนโดยอัตโนมัติ เมื่อต้องการเขียนโปรแกรมด้วยตัวแปรภาษา Microsoft GWBasic ก็ต้องโหลดอินเตอร์พรีเตอร์ เข้าสู่หน่วยความจำและเมื่ออินเตอร์พรีเตอร์ถูกโหลดเข้าสู่หน่วยความจำเรียบร้อยแล้ว จึงจะสามารถเริ่มเขียนโปรแกรมได้ เมื่อต้องการใช้งานโปรแกรมสามารถใช้คำสั่ง RUN ของอินเตอร์พรีเตอร์นั้นๆ เพื่อสั่งให้โปรแกรมเริ่มทำงานได้ สิ่งที่เราต้องระลึกไว้เสมอคือ ตัวอินเตอร์พรีเตอร์จะต้องถูกโหลดเข้าสู่หน่วยความจำ ก่อนที่จะเรียกใช้โปรแกรมที่เขียนด้วยตัวแปรภาษาชนิดนี้ขึ้นมาทำงานได้ เหตุผลคืออินเตอร์พรีเตอร์รับหน้าที่เป็นผู้ที่

คอยจัดการและควบคุมการติดต่อระหว่างโปรแกรมของเรากับระบบปฏิบัติการ สมมติว่าเขียนโปรแกรมขึ้นมาหนึ่งโปรแกรมใช้ชื่อว่า GAME.BAS ด้วยตัวแปรภาษา GWBasic และจะได้โค้ดโปรแกรมข้างล่างนี้เป็นตัวอย่างเพื่อประกอบคำอธิบาย

```
10 FOR J = 1 TO 10
20 PRINT J;
30 NEXT J
```

หลังจากที่พิมพ์ข้อความดังกล่าวนี้ลงไปแล้ว ใช้คำสั่ง RUN เพื่อให้โปรแกรมทำงานจากการที่อินเตอร์พรีเตอร์จะเริ่มทำงานตั้งแต่บรรทัดต่ำที่สุดเสมอ ดังนั้นบรรทัดที่ 10 จะเป็นบรรทัดแรกในตัวอย่างนี้ สิ่งที่อินเตอร์พรีเตอร์เห็นเป็นอันดับแรกก็คือ คำสั่ง FOR สำหรับคำว่า FOR ใน BASIC นี้เราเรียกว่าเป็นคีย์เวิร์ด ซึ่งหมายถึงคำที่มีความพิเศษนำมาใช้ในภาษาคอมนั้นๆ ในตัวอย่างข้างต้น FOR เป็นคีย์เวิร์ดที่ถูกใช้สำหรับการสร้าง วงรอบการทำงาน ในขั้นต้น BASIC ย่อมไม่รู้ว่า FOR เป็นคีย์เวิร์ดดังนั้นเมื่อมันเจอคำใดๆ ก็ตาม BASIC ก็จะเริ่มการค้นหาในรายชื่อคีย์เวิร์ดที่มีอยู่ ว่ามีคำว่า FOR หรือไม่ ถ้าหากว่าอินเตอร์พรีเตอร์ไม่พบคำว่า FOR จากรายชื่อคีย์เวิร์ด ก็จะฟ้องข้อผิดพลาดขึ้นมาว่า Syntax Error ซึ่งหมายความว่า โค้ดโปรแกรมที่ถูกเขียนขึ้นไม่เป็นไปตามหลักไวยากรณ์ของภาษานั้นๆ ยกตัวอย่างเช่น ถ้าพิมพ์ผิดเป็น FRO แทนที่จะเป็นคำว่า FOR ข้อผิดพลาด Syntax Error ก็จะถูกแสดงขึ้นมาเพราะอินเตอร์พรีเตอร์ไม่สามารถหาคำว่า FRO เจอในรายชื่อคีย์เวิร์ด ถ้าหากเราพิมพ์ถูกต้องอินเตอร์พรีเตอร์ก็จะค้นหาคำว่า FOR เจอในรายชื่อคีย์เวิร์ดและดำเนินการทำงานต่อไป คีย์เวิร์ดแต่ละตัวในรายชื่อคีย์เวิร์ดของตัวแปรภาษาใดๆ จะมีกฎเกณฑ์และไวยากรณ์ของตัวมันเอง ตามหลักไวยากรณ์ของคำว่า FOR ใน BASIC จะต้องตามด้วยการกำหนดค่าตัวแปรพบประโยคโปรแกรมว่า J = 1 และพบว่าทุกอย่างเป็นไปตามหลักไวยากรณ์ที่ถูกต้อง จากนั้นอินเตอร์พรีเตอร์ก็จะเริ่มตรวจสอบต่อไปว่าตัวแปรที่ชื่อ J เคยปรากฏในโปรแกรมแล้วหรือไม่ก่อนหน้านี้ การตรวจสอบที่ว่านี้ก็เกิดขึ้นโดยอินเตอร์พรีเตอร์จะไปค้นหาจากตารางสัญลักษณ์ ซึ่งหมายถึงหน่วยความจำที่ถูกอินเตอร์พรีเตอร์หรือคอมไพเลอร์ใช้สำหรับเก็บข้อมูลต่างๆ เกี่ยวกับตัวแปรและข้อมูลอื่นๆ ที่เกี่ยวข้องกับการใช้งานโปรแกรมนั้นๆ ในกรณีนี้ตารางสัญลักษณ์ยังคงว่างอยู่เพราะยังไม่เคยมีตัวแปรใดถูกใช้มาก่อนหน้านี้เลยเมื่ออินเตอร์พรีเตอร์พบว่าตัวแปรนี้เป็นตัวแปรใหม่ที่ถูกใช้เป็นครั้งแรก อินเตอร์พรีเตอร์ก็จะจัดการเก็บข้อมูลของตัวแปร (J) ลงในตารางสัญลักษณ์ของตน การที่จะทำเช่นนี้ได้ อินเตอร์พรีเตอร์ก็จะต้องหาตำแหน่งที่อยู่ในหน่วยความจำเพื่อจะเก็บข้อมูลดังกล่าวนี้เสียก่อนซึ่งเป็นหน้าที่ของอินเตอร์พรีเตอร์เองที่จะค้นหาพื้นที่หน่วยความจำที่ว่างและจัดการบันทึกข้อมูลเหล่านี้ไว้ในหน่วยความจำ เราอาจแสดงรูปแบบคร่าวๆ ของการเก็บข้อมูลตัวแปรลงในตารางสัญลักษณ์ได้ ทั้งหมดนี้จะเห็นได้ว่าอิน

เตอร์พรีเตอร์ใช้เวลาเกือบทั้งหมดไปกับการ ตรวจสอบโค้ดโปรแกรมในแต่ละบรรทัด ค้นหาและเปรียบเทียบระหว่างข้อความที่พบในโค้ดโปรแกรมกับรายชื่อคีย์เวิร์ดที่มีอยู่ และค้นหาข้อมูลเกี่ยวกับตัวแปรจากตารางสัญลักษณ์ ซึ่งสามารถบอกได้ว่าอินเตอร์พรีเตอร์ใช้เวลาไปกับ 3 ส่วนนี้มากกว่าการประมวลผลโค้ดโปรแกรมที่เราเขียนขึ้นเสียอีกผลลัพธ์ของมันก็คือ ทำให้การทำงานของโปรแกรมที่เขียนขึ้นด้วยอินเตอร์พรีเตอร์ช้ากว่าการทำงานของโปรแกรมที่เขียนขึ้นสำหรับกรณีที่ใช้คอมไพเลอร์เป็นตัวแปรภาษา ซึ่งโปรแกรมวิชวลเบสิก 5.0/6.0 ใช้ตัวแปรภาษาแบบอินเตอร์พรีเตอร์

2.2 หลักการพื้นฐานของวิชวลเบสิก

แม้ว่าวิชวลเบสิกจะเป็นภาษาแบบอินเตอร์พรีเตอร์ แต่ก็เป็นภาษาคอมพิวเตอร์ที่มีศักยภาพสูงและถูกออกแบบมาสำหรับใช้ในการพัฒนาโปรแกรมประยุกต์ ที่ทำงานบนโปรแกรมวินโดวส์ โดยเฉพาะ ทั้งรูปแบบการใช้งานก็ง่ายและไม่ซับซ้อนเท่าภาษาซีพลัสพลัสที่เป็นคอมไพเลอร์ของภาษาที่นิยมใช้ออกแบบโปรแกรมบนวินโดวส์ และดอสด้วยเช่นเดียวกัน คำว่าวิชวลหมายถึงวิธีการที่ใช้ในการสร้าง ส่วนติดต่อกับผู้ใช้แบบกราฟิก (Graphic User Interface – GUI) ส่วนคำว่าเบสิกหมายถึงภาษาคอมพิวเตอร์ (Beginner All-Purpose Symbolic Instruction Code) ซึ่งเป็นภาษาคอมพิวเตอร์ที่นักเขียนโปรแกรมในอดีตใช้กันอย่างแพร่หลายที่สุด วิชวลเบสิกเป็นวิวัฒนาการหนึ่งของภาษาเบสิกเดิมโดยมีการเพิ่มเติมประโยคโปรแกรม (Statement) ฟังก์ชัน (Function) และคีย์เวิร์ด (Keyword) ต่างๆ เข้าไปมากมาย รวมทั้งการติดต่อกับ GUI โดยตรงเช่น เมนู ปุ่มสั่งงาน เป็นต้น

ข้อดีประการหนึ่งของวิชวลเบสิกที่มีเหนือภาษาอื่นๆ สำหรับการพัฒนาโปรแกรมบนวินโดวส์ก็คือแทนที่เราจะต้องเขียนโค้ดโปรแกรมหลายๆ บรรทัดเพื่อสร้างส่วนติดต่อผู้ใช้ ซึ่งวิชวลเบสิกเรียกว่าคอนโทรล บนสภาพแวดล้อมของวินโดวส์ขึ้นมาชิ้นหนึ่งด้วยโค้ดโปรแกรมหลายสิบบรรทัดในวิชวลเบสิก ซึ่งเพียงแค่เลือกส่วนติดต่อผู้ใช้ที่ต้องการจากสิ่งที่วิชวลเบสิกเตรียมไว้ให้แล้ว ด้วยวิธีการลากและวาง (Drag and Drop) ลงบนหน้าจอ จากนั้นก็กำหนด คุณลักษณะ ของส่วนติดต่อผู้ใช้นั้นๆ ตามต้องการแล้วก็เขียนโค้ดโปรแกรมสำหรับคอนโทรลนั้นๆ เพื่อให้โปรแกรมทำงานตามที่ต้องการก็เป็นอันเรียบร้อย เนื่องจากวิชวลเบสิกถูกออกแบบมาสำหรับออกแบบโปรแกรมที่ทำงานบนวินโดวส์โดยเฉพาะ ดังนั้นหากต้องการจะเข้าใจพื้นฐานการใช้ภาษาวิชวลเบสิก จึงควรที่จะเข้าใจหลักการการทำงานเบื้องต้นของโปรแกรมวินโดวส์เสียก่อน เพราะโปรแกรมที่ทำงานบนวินโดวส์ กับโปรแกรมที่ทำงานบนดอสนั้นมีข้อแตกต่างกันอย่างมีนัยสำคัญ

ระบบปฏิบัติการวินโดวส์เป็นโปรแกรมที่ทำงานในโหมดกราฟิกอยู่ตลอดเวลา ดังนั้นตัวระบบปฏิบัติการวินโดวส์ต่างๆ ต่างก็เป็นเช่นเดียวกันคือมีลักษณะเป็นสี่เหลี่ยม มีขอบของตัวเอง และผู้ใช้สามารถเคลื่อนย้ายวินโดวส์ไปยังตำแหน่งต่างๆ ของหน้าจอได้ ตัวอย่างของวินโดวส์ก็ได้แก่ หน้าจอของโปรแกรมวินโดวส์เอ็กพลอเรอร์ ในวินโดวส์ 95/98 หน้าเอกสารของโปรแกรมไมโครซอร์ฟเวิร์ด หรือหน้าจอของโปรแกรมเกม Minesweeper ทั้งนี้คำว่า วินโดวส์ในความหมายของตัวระบบปฏิบัติการเองยังหมายถึง วินโดวส์ในลักษณะอื่นที่ไม่คุ้นเคยอีกด้วยเช่น ปุ่มสั่งงาน (Command Button) ไอคอน (Icon) ซึ่งต่างก็เป็นวินโดวส์อีกรูปแบบหนึ่งในความหมายของระบบปฏิบัติการวินโดวส์เช่นเดียวกัน ด้วยเหตุที่ระบบปฏิบัติการวินโดวส์ต้องจัดการกับวินโดวส์ที่แตกต่างกันอยู่มากมายโปรแกรมวินโดวส์จึงต้องสร้างค่าตัวเลขขึ้นมากลุ่มหนึ่งเพื่อใช้สำหรับการอ้างอิงถึงวินโดวส์แต่ละวินโดวส์โดยชุดหนึ่งคือค่าอินสแตนซ์ของโปรแกรม ซึ่งเป็นค่าเฉพาะของแต่ละโปรแกรมที่กำลังทำงานอยู่ขณะนั้น และอีกชุดหนึ่งเรียกว่า ค่าแฮนเดิลของวินโดวส์ ซึ่งใช้เป็นค่าที่อ้างถึงแต่ละวินโดวส์ของโปรแกรมโดยระบบปฏิบัติการจะตรวจสอบการทำงานของแต่ละวินโดวส์อยู่ตลอดเวลาเพื่อตรวจจับการทำงานและ เหตุการณ์ตอบสนอง ที่เกิดกับวินโดวส์นั้นๆ เหตุการณ์ตอบสนองที่ว่านี้อาจจะเกิดจากการที่ผู้ใช้คลิกเมาส์หรือกดคีย์บอร์ด หรือเกิดจากการทำงานของวินโดวส์อื่นๆ ก็ได้ ทุกครั้งที่มีเหตุการณ์ตอบสนองเกิดขึ้นก็จะมีข้อความของวินโดวส์ส่งไปยังตัวระบบปฏิบัติการ ระบบปฏิบัติการก็จะจัดการประมวลผลข้อความนั้นๆ และส่งสัญญาณให้ทุกวินโดวส์รู้และแสดงผลลัพธ์ต่างๆ โดยสัมพันธ์กับผลของการประมวลที่ได้ ตัวอย่างเช่น เมื่อเราย้ายวินโดวส์หนึ่งไปทับกับอีกวินโดวส์หนึ่งและย้ายกลับออกมาวางที่เดิมระบบปฏิบัติการวินโดวส์จะรับข้อความในการย้ายวินโดวส์นี้ และจัดการวาดวินโดวส์ที่ถูกทับขึ้นมาใหม่อีกครั้งเมื่อวินโดวส์ที่เราเคลื่อนย้ายถูกเคลื่อนย้ายออกไปจากวินโดวส์ที่ถูกทับ อย่างนี้เป็นต้น

ดังได้เห็นแล้วว่า การทำงานโปรแกรม วินโดวส์นั้นค่อนข้างสลับซับซ้อนและเกี่ยวข้องกับวินโดวส์ เหตุการณ์ตอบสนองต่างๆ และผลของการประมวลผลข้อความของวินโดวส์มากมายแต่สำหรับการเขียนโปรแกรมด้วยวิชวลเบสิก ผู้เขียนโปรแกรมไม่จำเป็นต้องเกี่ยวข้องกับการจัดการค่าแฮนเดิลต่างๆ เพราะวิชวลเบสิกจะจัดการงานระดับพื้นฐานของวินโดวส์เช่นนี้ให้เอง สิ่งที่ต้องทำคงเป็นเพียงการกำหนดเหตุการณ์ตอบสนองอื่นๆ ที่เราต้องการให้โปรแกรมของเราทำงานเมื่อเกิดเหตุการณ์ที่เรากำหนดขึ้นเช่น การคลิกเมาส์ลงบนปุ่มสั่งงาน หรือการพิมพ์ข้อความลงในช่องข้อความ เป็นต้นเพื่อบรรลุวัตถุประสงค์ของโปรแกรมเท่านั้น

2.3 การเขียนโปรแกรมแบบตอบสนองตามเหตุการณ์

ในการเขียนโปรแกรมประยุกต์ทั่วไปนั้น โปรแกรมจะเริ่มทำงานจากโค้ดโปรแกรมบรรทัดแรกไปตามลำดับขั้นตอนที่ถูกกำหนดไว้ โดยจะทำการเรียกโพรซีเยอร์ต่างๆ ขึ้นมาใช้งานตามแต่สถานการณ์ ส่วนโปรแกรมที่เป็นแบบ อีเวนท์ไดรเวอร์เช่น ตัวระบบปฏิบัติการวินโดวส์ 95/98 เองหรือโปรแกรมที่เขียนด้วยวิชวลเบสิก นั้นแตกต่างออกไป การทำงานของโค้ดโปรแกรมจะไม่ได้เริ่มต้นที่บรรทัดแรกไปตามลำดับขั้นตอนที่ถูกกำหนดมาแล้วแต่อย่างใด ทว่าโปรแกรมจะทำงานในแต่ละ โพรซีเยอร์เหตุการณ์ตอบสนองเมื่อเกิดเหตุการณ์ตอบสนองนั้นขึ้น เหตุการณ์ตอบสนองดังกล่าวอาจมีสาเหตุมาจากเหตุการณ์ที่ผู้ใช้เป็นผู้สร้างขึ้น ข้อความที่ส่งมาจากระบบเหตุการณ์ตอบสนองจากโปรแกรมอื่นๆ หรือแม้แต่จากเหตุการณ์ตอบสนองที่ตัวโปรแกรมเองเป็นผู้สร้างขึ้น ผลของเหตุการณ์ตอบสนองเหล่านี้จะเป็นตัวกำหนดว่าโค้ดโปรแกรมส่วนใดจะถูกเรียกขึ้นมาทำงานดังนั้นลำดับขั้นตอนการทำงานของโค้ดโปรแกรมแต่ละส่วนจะแตกต่างกันไปในแต่ละครั้งที่โปรแกรมถูกเรียกขึ้นมาใช้งาน

เนื่องจากการทำงานที่ไม่เป็นไปตามลำดับขั้นตอนนี้เอง ทำให้เราต้องระมัดระวังในการเรียกใช้งานแต่ละโพรซีเยอร์ให้ถูกต้องสมเหตุสมผลเช่น สมมติว่าเราต้องการให้ผู้ใช้ป้อนชื่อของตัวเองลงในช่องข้อความ แล้วคลิกปุ่มสั่งงานเพื่อป้อนข้อมูลจึงควรที่จะเขียนโปรแกรมให้ปุ่มสั่งงานนั้นไม่สามารถใช้งานได้จนกว่าชื่อของผู้ใช้จะถูกป้อนลงในช่องข้อความ หรืออาจจะเขียนโปรแกรมป้องกันเหตุผิดพลาดที่ผู้ใช้อาจจะเป็นผู้สร้างขึ้นเช่น ห้ามมิให้ใส่ตัวหนังสือลงไปในช่องข้อความที่ต้องการตัวเลขเป็นต้น

ในบางกรณีการทำงานของโค้ดโปรแกรมก็อาจจะเป็นผู้สร้างเหตุการณ์ขึ้นเสียเองก็ได้เช่น เมื่อผู้ใช้เริ่มดำเนินการเปลี่ยนแปลงข้อมูลที่แสดงอยู่ในช่องข้อความ เหตุการณ์ตอบสนองเปลี่ยนแปลงของช่องข้อความนั้นก็สามารถใช้เป็นเหตุการณ์ตอบสนองที่จะเรียกโค้ดโปรแกรมชุดหนึ่งที่เราเขียนไว้ขึ้นมาทำงานก็ได้ ดังนั้นในการออกแบบโปรแกรมภายใต้หลักการของ อีเวนท์ไดรเวอร์ จึงต้องคำนึงถึงความสมเหตุสมผลของการเรียกโค้ดโปรแกรมขึ้นมาทำงานตามแต่ละเหตุการณ์และป้องกันข้อผิดพลาดที่อาจเกิดขึ้นได้อย่างครบถ้วน

2.4 หลักการพัฒนาโปรแกรมคอมพิวเตอร์

เราสามารถกล่าวได้ว่าการพัฒนาโปรแกรมคอมพิวเตอร์ทุกชนิด ตั้งแต่โปรแกรมสำหรับงานธุรกิจอย่างเช่น Microsoft Word , Windows 95/98 หรือกระทั่งโปรแกรมทั้งหลาย ไม่ว่าจะเขียนขึ้นด้วยภาษาคอมพิวเตอร์อะไรก็ตาม จะประกอบไปด้วยขั้นตอนพื้นฐานในการสร้าง 5 ขั้นตอน ดังต่อไปนี้

1. การเริ่มต้นใช้งานโปรแกรม (Initialization)

2. ข้อมูลอินพุท (Input Data)
3. การประมวลผลข้อมูล (Processing)
4. ข้อมูลเอาต์พุท (Output Data)
5. การจบการทำงานโปรแกรม (Shut Down)

เราจะกล่าวถึงแต่ละขั้นตอนในรายละเอียดดังต่อไปนี้

2.4.1 การเริ่มต้นใช้งานโปรแกรม

มีโปรแกรมอยู่หลายโปรแกรมที่ต้องการการตระเตรียมข้อมูลต่างๆ ก่อนที่โปรแกรมนั้นๆ จะสามารถทำงานได้ ขั้นตอนการกำหนดค่าและทดสอบค่าต่างๆ จะเรียกว่า ขั้นตอนเริ่มต้นใช้งานโปรแกรม (Initialization) ซึ่งหมายถึง การจัดการข้อมูลทั้งหลายที่จำเป็นในการใช้งานโปรแกรมนั้นๆ ก่อนการเริ่มใช้งานโปรแกรม ยกตัวอย่างเช่น โปรแกรมเกมบนวินโดวส์ ที่เขียนด้วย DirectX API จะต้องทำการปรับโหมดการแสดงผลของจอภาพก่อนที่จะเริ่มต้นโปรแกรมทุกครั้ง เป็นต้น

อย่างไรก็ตาม สำหรับโปรแกรมที่ไม่ซับซ้อนยุ่งยากนัก ก็อาจจะไม่มีการกำหนดค่าใดๆ เลยก่อนการใช้งานโปรแกรม ดังนั้น จึงสามารถสรุปได้ว่าทุกโปรแกรมที่จะพัฒนาขึ้นต้องมีการ Initialization ข้อมูลต่างๆ ที่จำเป็นต่อการทำงานของโปรแกรมก่อนเสมอ

2.4.2 ข้อมูลอินพุท

โปรแกรมคอมพิวเตอร์ทั้งหลายนั้น ไม่ได้เป็นอะไรไปมากกว่าโปรแกรมที่ยุ่งอยู่กับการจัดการข้อมูลและดัดแปลงข้อมูล เพราะถ้าหากไม่มีข้อมูลอินพุทป้อนเข้าไปให้โปรแกรมแล้ว โปรแกรมเกือบทุกโปรแกรมก็ไม่สามารถทำงานได้ ดังนั้นในส่วนของ ข้อมูลอินพุท (Input Data) จึงเป็นขั้นตอนที่ทำการรวบรวมข้อมูล ตรวจสอบ และคัดเลือกข้อมูลที่จะถูกใช้ในโปรแกรม

โดยทั่วไปแล้วข้อมูลที่เป็นอินพุทในการประมวลมักจะมาจากการป้อนของผู้ใช้งาน หรือการอ่านข้อมูลในไฟล์จากดิสก์ และยังมีรูปแบบของการได้มาซึ่งข้อมูลที่ใช้ในการประมวลผลอื่นๆ อีก เช่น ปากกาแสง (Light Pen) หรือการสัมผัสหน้าจอ (Touching Screen) ซึ่งต้องการอุปกรณ์พิเศษเพื่อการใช้งาน แต่ถึงกระนั้นจุดประสงค์ของการรวบรวมข้อมูล ก็คือเพื่อนำไปใช้ในการประมวลผลโปรแกรม

2.4.3 การประมวลผลข้อมูล

โปรแกรมเกือบทุกโปรแกรมถูกออกแบบให้นำข้อมูลที่ป้อนเข้าไป ไปทำการดัดแปลงอย่างใดอย่างหนึ่งจนกลายเป็นข้อมูลใหม่ ซึ่งการดัดแปลงข้อมูลจากรูปแบบหนึ่งไปเป็นข้อมูลอีกรูปแบบหนึ่งนี้เรียกว่า การประมวลผลข้อมูล (Processing) ยกตัวอย่างเช่น มีข้อมูลตัวเลขอยู่ 10 ตัวคละกัน ต้องการให้ข้อมูลทั้ง 10 ตัวนี้เรียงจากน้อยไปหามาก การประมวลผลสำหรับงานนี้ก็คือ การเรียงข้อมูล

ประสิทธิภาพของการทำงานโปรแกรมใดๆ ขึ้นอยู่กับรูปแบบการประมวลผลของโปรแกรมนั้นๆ ดังเช่นตัวอย่างที่ยกมา มีวิธีเรียงข้อมูลอยู่นับสิบแบบที่ใช้กัน แต่ละวิธีก็มีข้อดีและประสิทธิภาพในการทำงานไม่เท่ากัน ซึ่งสิ่งที่จะต้องทำความเข้าใจก็คือ วิธีการเรียงข้อมูลแต่ละแบบที่มีอยู่นั้นมีวิธีการประมวลผลอย่างไร เพื่อจะได้วิธีการที่เหมาะสมไปใช้ได้ถูกต้องและเกิดประสิทธิภาพในการทำงานสูงสุดกับโปรแกรม เพราฉะนั้นสามารถสรุปได้ว่าโปรแกรมจะทำงานได้ดีเพียงใดนั้น ขึ้นอยู่กับรูปแบบของการประมวลผลที่เขียนขึ้น ขั้นตอนนี้จึงเป็นขั้นตอนสำคัญ ที่ทำให้โปรแกรมหลายโปรแกรมที่ทำงานคล้ายกันแต่มีประสิทธิภาพแตกต่างกันอย่างเห็นได้ชัด

2.4.4 ข้อมูลเอาต์พุท

หลังจากที่ข้อมูลอินพุทถูกดัดแปลงแล้ว ข้อมูลใหม่ที่ผ่านการประมวลผลแล้วจะเรียกว่า ข้อมูลเอาต์พุท (Output Data) อย่างเช่นในตัวอย่างที่แสดง ข้อมูลที่ผ่านการจัดเรียงจากน้อยไปมากแล้วก็คือผลลัพธ์ของการจัดเรียงข้อมูล และการที่ผู้ใช้จะเห็นผลของการประมวลผลได้ก็โดยผ่านทางหน้าจอมอนิเตอร์ เครื่องพิมพ์ ข้อมูลเป็นไฟล์ในดิสก์ หรือแม้แต่ข้อมูลผ่านทางโมเด็ม ดังนั้น ไม่ว่าอุปกรณ์ในการแสดงผลลัพธ์จะเป็นอะไร ผลลัพธ์ในขั้นตอนนี้ก็มุ่งสู่จุดหมายเดียวกัน นั่นคือการแสดงข้อมูลผลลัพธ์ของการประมวลผลให้ผู้ใช้ได้รับรู้ในทางใดทางหนึ่ง

2.4.5 การจบการทำงานโปรแกรม

เมื่อใดที่ต้องการเลิกใช้งานโปรแกรม ก็เพียงแค่ออกจากโปรแกรมนั้นๆ อาจดูง่ายมากในสายตาของผู้ใช้งาน แต่ในสายตาของนักเขียนโปรแกรมแล้วเป็นขั้นตอนที่ยุ่งยากกว่านั้น การจบการทำงานโปรแกรม (Shut Down) นั้นหมายถึง ขั้นตอนที่เกิดขึ้นหลังจากการแสดงผลลัพธ์ของการประมวลผลเสร็จสิ้นแล้ว ยกตัวอย่างเช่น โปรแกรมอาจจะอ่านไฟล์จากดิสก์เพื่อเก็บข้อมูลสำหรับการประมวลผลขณะที่โปรแกรมกำลังทำงาน เมื่อมีการออกจากโปรแกรม ก็ควรที่จะจัดการปิดไฟล์นั้นๆ ด้วย อย่างเช่นโปรแกรมตัวอย่างเกมบางเกมจะมีการใช้พื้นที่ในหน่วยความจำเพื่อเก็บข้อมูลภาพที่จะแสดง และในขั้นตอนของการออกจากโปรแกรมก็จะมีการคืนหน่วยความจำเหล่านั้นกลับสู่ระบบปฏิบัติการ สำหรับนำไปใช้งานอื่นๆ ต่อไป

2.5 รายละเอียดประกอบการเขียนโปรแกรม

อย่างไรก็ตามขั้นตอน 5 ขั้นตอนที่กล่าวมาข้างต้นนั้น เป็นเค้าโครงหลักเพื่อให้มองเห็นภาพรวมในการสร้างโปรแกรมเท่านั้น ในแต่ละขั้นตอนยังต้องการ รายละเอียดประกอบการเขียนโปรแกรม (Sideways Refinement) อีกจำนวนหนึ่งก่อนที่จะเริ่มเขียนโค้ดโปรแกรมจริงๆ

รายละเอียดที่จะกล่าวถึงนี้ยังคงตั้งอยู่บน 5 ขั้นตอนดังกล่าว แต่ว่าแบ่งรายละเอียดในแต่ละขั้นตอนออกเป็นส่วนย่อยๆ เพื่อแสดงว่าแต่ละขั้นตอนนั้นโปรแกรมต้องทำอะไรบ้าง จะต้องเขียนรายละเอียดของแต่ละขั้นตอนในการสร้างโปรแกรม รวมทั้งฟังก์ชันที่ต้องการจะใช้ ดังตัวอย่างในตารางที่ 2.1 จุดประสงค์ของการสร้างรายละเอียดเช่นนี้ก็เพื่อแสดงขั้นตอนย่อยๆ ในแต่ละขั้นตอนหลักว่าจะต้องมีอะไรบ้าง เพื่อให้สามารถมองเห็นเส้นทางและลำดับขั้นตอนในการเขียนโปรแกรมได้ชัดเจนยิ่งขึ้น

ตารางที่ 2.1 แสดงตัวอย่างรายละเอียดประกอบการเขียนโปรแกรม

ขั้นตอน	งานที่ต้องทำ	ฟังก์ชันที่ใช้
การเริ่มต้นใช้งานโปรแกรม	1. เคลียร์หน้าจอ	cls()
	2. เปิดไฟล์ที่เก็บข้อมูลยอดจำหน่าย	OpenFile()
ข้อมูลอินพุท	1. ดึงข้อมูลยอดจำหน่ายเฉพาะจาก จากเดือนที่ต้องการนำมาแสดง	Get_MonthSales()
ประมวลผล	1. หาค่าต่ำสุดและสูงสุดของชุดข้อมูล	FindMinMax()
	2. กำหนดสเกลที่ใช้	DoScaling()
ข้อมูลเอาต์พุท	1. วาดแกน X และแกน Y	DrawAxis()
	2. แสดงข้อความกำกับแกน	LabelAxis()
	3. วาดแผนภูมิแท่ง	DrawBars()
	4. แสดงชื่อภาพแผนภูมิแท่ง	LabelBars()
การจบการทำงานโปรแกรม	1. ปิดไฟล์ข้อมูล	CloseFile()
	2. ออกจากโปรแกรม	End

2.6 อัลกอริทึมคืออะไร

ในการที่จะบรรลุซึ่งการกระทำใดๆ นั้นจะต้องมีลำดับความคิดอันนำไปสู่การกระทำและผลลัพธ์ของมัน อัลกอริทึม (Algorithm) หมายถึง ลำดับขั้นตอนในการปฏิบัติงานใดงานหนึ่งให้สำเร็จ อัลกอริทึมมีความสัมพันธ์โดยตรงต่อประสิทธิภาพในการทำงานของโปรแกรม ด้วยผลลัพธ์ของงานอย่างเดียวกัน หากกระทำด้วยอัลกอริทึมที่แตกต่างกัน ก็อาจจะสร้างผลลัพธ์ที่แตกต่างอย่างมีนัยสำคัญได้อย่างไม่น่าเชื่อ โดยการเขียนรายละเอียดประกอบการเขียนโปรแกรม 5 ขั้นตอนดังกล่าวเป็นตัวบังคับให้ต้องคิดถึงอัลกอริทึมในการประมวลผลข้อมูลของโปรแกรมที่ต้องการจะสร้างก่อนลงมือจริง

2.7 การเขียนโปรแกรมด้วยภาษา Visual Basic

Visual Basic เป็นภาษาคอมพิวเตอร์ที่เป็น Object-based Programming หมายถึงการเขียนโปรแกรมที่มองสิ่งต่างๆ เป็น "ออบเจ็กต์ (วัตถุ)" ความหมายของออบเจ็กต์ในทางเขียนโปรแกรมก็ไม่ต่างอะไรกับวัตถุทั่วไปที่พบในชีวิตประจำวัน มีออบเจ็กต์อยู่มากมายหลายชนิดใน Visual Basic โดยโค้ดโปรแกรมที่ต้องเขียนนั้นก็เพื่อกำหนดความสัมพันธ์ของแต่ละออบเจ็กต์ในโปรแกรมนั่นเอง

รูปแบบการออกแบบโปรแกรมของ Visual Basic เป็นแบบ ตอบสนองตามเหตุการณ์ (Event-Driven) เนื่องจากว่า Visual Basic เป็นภาษาคอมพิวเตอร์ที่ทำงานกับออบเจ็กต์ ดังนั้นรูปแบบของโค้ดโปรแกรมก็จะสัมพันธ์กับออบเจ็กต์ต่างๆ ที่เห็นบนหน้าจอเป็นหลัก กล่าวคือ ทุกออบเจ็กต์ที่ปรากฏขึ้นบนหน้าจอของโปรแกรมที่เขียนด้วย Visual Basic ต่างก็บรรจุข้อมูลเกี่ยวกับคุณลักษณะของตัวเองและโค้ดโปรแกรมจำนวนหนึ่งที่จะกำหนดการใช้งานของมัน วินโดวส์หนึ่งๆ ในโปรแกรมที่ออกแบบด้วย Visual Basic จะเรียกว่า โมดูลฟอร์ม (Form Module) คือไฟล์ที่มีนามสกุล *.FRM

ในแต่ละฟอร์มจะประกอบไปด้วย ส่วนออกแบบของฟอร์มและคอนโทรลที่เป็นหน้าตา (Presentation) ของฟอร์ม อันปรากฏให้เห็นจริงขณะโปรแกรมทำงาน และ ส่วนของโค้ดโปรแกรม (Source Code) อันประกอบด้วย ส่วนประกาศ (General Declaration) และโพรซีเยอร์ย่อยต่างๆ แบ่งเป็น 3 ประเภทคือ โพรซีเยอร์ซับรูทีน โพรซีเยอร์ฟังก์ชัน และโพรซีเยอร์เหตุการณ์ตอบสนอง ในแต่ละโพรซีเยอร์เหตุการณ์ตอบสนองแหล่านั้นจะบรรจุโค้ดโปรแกรมสำหรับทำงานเพื่อตอบสนองเหตุการณ์ใดเหตุการณ์หนึ่ง

บนฟอร์มใดๆ จะบรรจุคอนโทรลต่างๆ เอาไว้หรือไม่ก็ได้ แต่โดยทั่วไปแล้วบนฟอร์มจะบรรจุคอนโทรลไว้ด้วยกันหลายชนิด แต่ละคอนโทรลสามารถจะตอบสนองเหตุการณ์ต่างๆ ที่จะเกิดได้ตามโค้ดโปรแกรมที่ถูกเขียนไว้ในโพรซีเยอร์เหตุการณ์ตอบสนองของคอนโทรลเหล่านั้น

ในบางกรณีที่โค้ดโปรแกรมส่วนหนึ่งไม่ได้เกี่ยวข้องกับฟอร์มใดฟอร์มหนึ่งโดยเฉพาะ และต้องการให้ฟอร์มอื่นๆ สามารถเรียกใช้งานได้ด้วย ก็อาจจะเขียนโค้ดโปรแกรมแหล่านั้นไว้ในโมดูลอื่น อันได้แก่ โมดูลมาตรฐาน (Standard Module (*.BAS)) โพรซีเยอร์ใดก็ตามที่เกี่ยวข้องกับออบเจ็กต์ต่างๆ กันหลายออบเจ็กต์ ก็ไม่ควรจะเขียนโค้ดโปรแกรมให้เฉพาะเจาะจงสำหรับออบเจ็กต์ใดๆ เป็นพิเศษ เพื่อว่าจะได้นำมาบรรจุไว้ในโมดูลมาตรฐานและเรียกใช้ได้จากทุกออบเจ็กต์ แทนที่จะต้องเขียนโค้ดโปรแกรมเหล่านั้นซ้ำแล้วซ้ำอีกสำหรับแต่ละออบเจ็กต์

โมดูลคลาส (Class Module (.CLS)) เป็นโมดูลอีกชนิดหนึ่งที่มีไว้สำหรับสร้างออบเจ็กต์ที่สามารถถูกเรียกใช้งานได้จากโพรซีเยอร์ต่างๆ ในโปรแกรม ซึ่งหากเปรียบเทียบกับโมดูลมาตรฐานแล้ว โมดูลมาตรฐานจะบรรจุเพียงโค้ดโปรแกรมสำหรับทำงานเท่านั้น ในขณะที่โมดูลคลาสจะบรรจุทั้งโค้ด

โปรแกรมและข้อมูลอื่นๆที่เกี่ยวข้อง ออบเจ็กต์ดังกล่าวนี้เปรียบเสมือนคอนโทรล คือมีข้อมูลหลายชนิดรวมอยู่ด้วยกันเป็นออบเจ็กต์ แต่ไม่สามารถปรากฏให้จับต้องได้เช่นคอนโทรล

2.8 เข้าใจการทำงานของโปรแกรมแบบตอบสนองตามเหตุการณ์(Event-Driven)

ในเมื่อแต่ละฟอร์มหรือคอนโทรลสามารถตอบสนองต่อเหตุการณ์ต่างๆ ที่เกิดขึ้นในโปรแกรมได้ ก็จะใช้เหตุการณ์เหล่านั้นเป็นตัวจุดชนวน (Trigger) ในการทำงานของโปรแกรม แต่ละฟอร์มและคอนโทรลใน Visual Basic ได้ถูกกำหนดมาก่อนแล้วว่าสามารถจะตอบสนองกับเหตุการณ์แบบใดได้บ้าง โค้ดโปรแกรมที่จะเขียนก็บรรจุอยู่ในโพรซีเยอร์เหตุการณ์ตอบสนองเหล่านี้นั่นเอง

ถึงแม้ว่าฟอร์มและคอนโทรลใน Visual Basic จะถูกกำหนดมาแล้วว่ามีเหตุการณ์ตอบสนองอะไรบ้าง แต่ก็เป็นหน้าที่ของผู้เขียนโปรแกรมว่าจะเลือกใช้เหตุการณ์ตอบสนองอะไรของฟอร์มและคอนโทรลตัวไหน เพื่อบรรจุโค้ดโปรแกรมสำหรับทำงานอะไร และอย่างไร

แม้ว่าเหตุการณ์ตอบสนองของฟอร์มและคอนโทรลต่างๆ จะแตกต่างกันไปตามแต่ละชนิดของคอนโทรล แต่ก็จะมีเหตุการณ์ตอบสนองพื้นฐานจำนวนหนึ่งที่ฟอร์มและคอนโทรลทุกชนิดมีร่วมกัน สามารถสรุปขั้นตอนในการทำงานโปรแกรมที่ออกแบบด้วย Visual Basic ได้ดังนี้

1. โปรแกรมประยุกต์ถูกเรียกขึ้นมาใช้งาน เริ่มด้วยการโหลดตัวเองเข้าสู่หน่วยความจำและแสดงหน้าจอให้ผู้ใช้เห็น
2. ฟอร์มและคอนโทรลต่างๆ จะคอยรับเหตุการณ์ตอบสนองต่างๆ ที่มาจากผู้ใช้โปรแกรมผ่านทางเมาส์และคีย์บอร์ด หรือมาจากตัวระบบ เช่น เวลา หรือมาจากโค้ดโปรแกรมในตัวโปรแกรมเอง เช่น เหตุการณ์ตอบสนองการโหลดของฟอร์ม
3. หากว่ามีเหตุการณ์ตอบสนองใด ๆ เกิดขึ้น โค้ดโปรแกรมที่บรรจุอยู่ในโพรซีเยอร์เหตุการณตอบสนอง ดังกล่าวจะถูกเรียกขึ้นมาทำงาน
4. หลังจากที่ทำงานเสร็จ โปรแกรมก็จะรอคอยเหตุการณ์ตอบสนองอันต่อไป

ดังจะได้เห็นว่าสิ่งแรกที่ควรทำในการออกแบบโปรแกรมด้วย Visual Basic ก็คือ การออกแบบรูปร่างหน้าตาของ ส่วนติดต่อกับผู้ใช้ (Interface) ของโปรแกรม อันประกอบด้วยฟอร์ม และคอนโทรลทั้งหลายนั่นเอง การจัดวางคอนโทรลต่าง ๆ บนฟอร์มควรที่จะสัมพันธ์กับการทำงานของโค้ดโปรแกรมที่จะเขียนลงไปสำหรับฟอร์ม และคอนโทรลเหล่านั้นด้วยซึ่งประสิทธิภาพของโปรแกรมแต่ละโปรแกรมก็ตัดสินกันตรงความง่ายต่อการใช้งานและการทำงานของโค้ดดปรแกรม การออกแบบความสัมพันธ์ระหว่างส่วนติดต่อกับผู้ใช้กับโค้ดโปรแกรมที่ดี ยังทำให้ผู้พัฒนาโปรแกรมสามารถปรับปรุงโปรแกรมนั้นๆให้ดีขึ้นได้ง่ายกว่าเดิมอีกด้วย

โปรแกรมที่ออกแบบด้วย Visual Basic ประกอบไปด้วยโมดูลฟอร์มเป็นพื้นฐาน ส่วนโมดูลอื่นๆ นั้นจะมีหรือไม่มีก็ได้ ในแต่ละโมดูลจะประกอบไปด้วยโพรซีเยอร์ย่อยต่างๆ โดยปกติแล้วเมื่อเพิ่มโมดูลหรือคอนโทรลใหม่เข้าไปในไฟล์โปรเจ็กต์ Visual Basic จะกำหนดชื่อเริ่มต้นให้โดยอัตโนมัติ เช่น Form1 สำหรับฟอร์ม, Command1 สำหรับคอนโทรลกลุ่มสั่งงาน เป็นต้นดังนั้นเพื่อความสะดวกต่อการสื่อความหมายจึงควรเปลี่ยนค่าคุณลักษณะ Name ของฟอร์มและคอนโทรลเหล่านั้นให้สื่อความหมายมากขึ้นนักเขียนโปรแกรม Visual Basic โดยทั่วไปนิยมตั้งชื่อฟอร์มและคอนโทรลด้วยชื่อย่อของฟอร์มและคอนโทรลดังแสดงในตารางที่ 2.2 แล้วตามที่ชื่อที่สื่อความหมายถึงฟอร์มและคอนโทรลนั้นๆ

ตารางที่ 2.2 แสดงการย่อชื่อในการเขียนโปรแกรม

ฟอร์มและชนิดคอนโทรล	ชื่อย่อที่นิยมใช้	ตัวอย่าง
Form	frm	frmMain
Label	lbl	lblInfo
Textbox	txt	txtInput
Command button	cmd	cmdCancel
Checkbox	chk	chkBold
Optionbutton	opt	optRed
Combobox	cbo	cboPayment
Listbox	lst	lstSurname
Hscrollbar	hbar	hbarBrightness
Vscrollbar	vbar	vbarVolume
Timer	tmr	tmrNextTurn
DriveListbox	drv	drvViewer
DirListbox	dir	dirViewer
FileListbox	fist	fistViewer
Picturebox	pct	pctUser
Image	img	imgLogo
DataControl	dat	datBibilo
Frame	fme	fmeFont

2.9 รูปแบบการเขียนโค้ดโปรแกรมใน Visual Basic

การเขียนโค้ดโปรแกรมสำหรับ Visual Basic จะกระทำในวินโดวส์แสดงโค้ดโปรแกรม (Code Editor Windows) ซึ่งถือเป็นโปรแกรมเขียนข้อความที่มีความสามารถสูงกว่าโปรแกรมเขียนข้อความทั่วไปอยู่หลายด้าน ที่จะช่วยให้การเขียนโค้ดโปรแกรมง่ายมากขึ้น

โปรแกรมจะทำงานตามโค้ดโปรแกรมที่บรรจุอยู่ในโมดูลทั้ง 3 ชนิด ในแต่ละโมดูลก็จะถูกแบ่งเป็นโพรซีเยอร์ย่อยๆของออบเจ็กต์ต่างๆ โดยการสร้างหรือไปยังโพรซีเยอร์ของออบเจ็กต์ใดๆในแต่ละโมดูลสามารถทำได้ด้วยการคลิกเมาส์ที่ช่องรายชื่อออบเจ็กต์ (Object List Box) ของโมดูลนั้นๆและเรียกชื่อโพรซีเยอร์ที่ต้องการจากช่องรายชื่อโพรซีเยอร์ (Procedure List Box)

จะสามารถสังเกตได้ว่าเมื่อเลือกที่ส่วนบนสุดของโมดูลฟอร์มจะมีคำว่า (General) ปรากฏอยู่ในช่องรายชื่อออบเจ็กต์ และมี(Declaration)เป็นโพรซีเยอร์ General ในที่นี้ไม่ใช่ออบเจ็กต์ใดๆใน Visual Basic แต่จะเป็นส่วนที่บอกให้ได้รับทราบว่าโค้ดโปรแกรมในพื้นที่ส่วนนี้จะบรรจุโค้ดที่เกี่ยวข้องกับการประกาศตัวแปร (Variable) , ตัวแปรค่าคงที่ (Constants) และโพรซีเยอร์ในไดนามิกลิงก์ไลบรารี (DLLs) ต่างๆ ซึ่งเรียกว่า ส่วนประกาศ (General Declaration) และถ้าเขียนโค้ดโปรแกรมที่เป็นซับรูทีนหรือฟังก์ชัน โค้ดโปรแกรมดังกล่าวก็จะบรรจุอยู่ต่ำลงไปจากส่วนประกาศ และชื่อของโพรซีเยอร์เหล่านั้นก็จะถูกเพิ่มเติมเข้าไปในช่องรายชื่อโพรซีเยอร์ เมื่อเลือก General เป็นชื่อออบเจ็กต์

รูปแบบการแสดงโค้ดโปรแกรมของวินโดวส์แสดงโค้ดโปรแกรมมีได้ 2 รูปแบบคือ แบบแสดงโพรซีเยอร์เพียงโพรซีเยอร์เดียว (Single Procedure View) และแบบแสดงโพรซีเยอร์ที่มีทั้งหมดพร้อมกันโดยมีเส้นคั่นระหว่างโพรซีเยอร์ (Full Module View) เมื่อผู้เขียนโปรแกรมพิมพ์ชื่อของคอนโทรลใดๆที่ปรากฏอยู่ในโปรเจ็กต์ปัจจุบันได้ถูกต้อง รายชื่อแสดงคุณลักษณะและวิธีใช้งาน (Auto List Members) ทั้งหมดที่เป็นไปได้ของคอนโทรลนั้นๆจะปรากฏขึ้นเพื่อให้ผู้เขียนโปรแกรมสามารถใช้ได้อย่างรวดเร็ว

2.10 ระบบตัวเลขใน Visual Basic

ตัวเลขที่ใช้งานใน Visual Basic เกือบทุกชนิดเป็นเลขฐาน 10 แต่ในบางกรณีอาจจะเป็นเลขฐาน 8 หรือเลขฐาน 16 ซึ่งผู้เขียนโปรแกรมสามารถรับรู้ความแตกต่างได้ โดยเมื่อ Visual Basic แสดงเลขเป็นแบบเลขฐาน 8 ก็จะมีตัวอักษร &O ปรากฏนำหน้าเลขฐาน 8 นั้น และตัวอักษร &H จะปรากฏนำหน้าเลขฐาน 16 นั้นๆ ดังตัวอย่างต่อไปนี้ซึ่งจะแสดงตัวเลขทั้ง 3 รูปแบบที่มีค่าเท่ากันในมุมองของ Visual Basic

ตารางที่ 2.3 แสดงเลขฐานต่าง ๆ

เลขฐาน 10	เลขฐาน 8	เลขฐาน 16
9	&O11	&H9
15	&O17	&HF
16	&O20	&H10
20	&O24	&H14
255	&O377	&HFF

2.11 หลักการตั้งชื่อทั่วไปของ Visual Basic

ในการใช้งาน Visual Basic จะต้องตั้งชื่อ ฟอร์ม และคอนโทรล ค่าคงที่ตัวแปร ซับรูทีน ฟังก์ชัน โพรซีเยอร์หรืออื่น ๆ ชื่อของสิ่งต่าง ๆ เหล่านี้จะตกอยู่ภายใต้กฎเกณฑ์การตั้งชื่อเดียวกันคือ

1. จะต้องขึ้นต้นด้วยตัวอักษรเท่านั้น (ไม่สามารถขึ้นต้นด้วยตัวเลขได้)
2. จะต้องไม่ปรากฏตัวอักษรพิเศษที่ใช้ในการประกาศชนิดของข้อมูลปรากฏอยู่ในชื่อ
3. ความยาวของชื่อจะต้องไม่เกิน 255 ตัวอักษร ในขณะที่ชื่อของฟอร์ม คอนโทรล โมดูลมาตรฐาน และโมดูลคลาสจะต้องไม่เกิน 40 ตัวอักษร
4. จะต้องไม่ซ้ำกับคีย์เวิร์ดทั้งหลายใน Visual Basic

คีย์เวิร์ด (Keyword) หมายถึง คำที่มีความหมายพิเศษของ Visual Basic อย่างเช่นคำว่า If , For หรือชื่อของฟังก์ชั่นที่ Visual Basic รู้จัก เช่น Len , Abs หรือชื่อของเครื่องหมายต่างๆ เช่น Or , Mod

2.12 ตัวแปรคืออะไร

Visual Basic ใช้ ตัวแปร (Variable) สำหรับเก็บค่าของข้อมูล โดยตัวแปรจะเก็บข้อมูลชนิดใดก็ขึ้นอยู่กับผู้เขียนโปรแกรมเป็นผู้กำหนด และ อาร์เรย์ (Array) คือกลุ่มของตัวแปรที่เก็บข้อมูลชนิดเดียวกัน โดยมี เลขดัชนี (Index) ที่แตกต่างกันเป็นตัวบ่งชี้ถึงตัวแปรแต่ละตัว

ตัวแปรค่าคงที่ (Constants) เป็นตัวแปรชนิดหนึ่งที่ใช้สำหรับเก็บข้อมูล แต่ค่าของตัวแปรค่าคงที่ค่าหนึ่งจะไม่มีการเปลี่ยนแปลงใด ๆ ตลอดการทำงานของโปรแกรม

ชนิดของข้อมูล (Data Type) ใน Visual Basic เป็นตัวกำหนดให้ Visual Basic รู้ว่าจะต้องใช้พื้นที่ในหน่วยความจำ จำนวนมากเท่าใดในการเก็บค่าของตัวแปรนั้น ๆ ซึ่งข้อมูลใน Visual Basic มีอยู่ด้วยกันหลายชนิด และจะศึกษาว่าชนิดของข้อมูลแต่ละชนิดเหมาะสมสำหรับเก็บข้อมูลอะไร

เมื่อสร้างตัวแปรขึ้นมาแล้ว สามารถกำหนดค่าให้กับตัวแปรนั้นได้ด้วยการเขียนโค้ดโปรแกรมเพื่อกำหนดค่าดังนี้

ชื่อตัวแปร = ค่าของตัวแปร

a = 150

หรือทำการเปลี่ยนแปลงค่าตัวแปรนั้น ๆ ด้วยการเขียนโค้ดโปรแกรมดังนี้

ชื่อตัวแปร = รูปแบบการเปลี่ยนแปลงของตัวแปร

a = a + 100

อย่างไรก็ตาม Visual Basic จะรู้จักตัวแปรใดๆที่สร้างขึ้นได้ก็ต่อเมื่อ ประกาศ (Declaring) ตัวแปรนั้น ๆ ให้ Visual Basic รับรู้เสียก่อน หลักไวยากรณ์ในการประกาศตัวแปรใน Visual Basic คือ

Dim variable's name [As Data Type]

Dim เป็นคีย์เวิร์ดใน Visual Basic ที่จะรับรู้ว่าผู้เขียนโปรแกรมต้องการจะประกาศตัวแปร

Variable's name คือชื่อของตัวแปรที่ Visual Basic จะใช้เป็นตัวอ้างถึงตัวแปรตัวนั้น โดยการตั้งชื่อตัวแปรจะต้องเป็นไปตามกฎเกณฑ์การตั้งชื่อทั่วไป และจะต้องไม่ซ้ำกับตัวแปรตัวอื่นในขอบเขตเดียวกัน

As เป็นคีย์เวิร์ดใน Visual Basic เพื่อกำหนดชนิดของข้อมูลของตัวแปร

Data Type คือชนิดของข้อมูลที่ตัวแปรตัวนั้นจะจัดเก็บค่าของข้อมูลได้ ซึ่งอาจจะเป็นชนิดข้อมูลทั่วไปที่ Visual Basic รู้จักอยู่แล้ว เช่น Integer , Currency หรือเป็นชนิดข้อมูลที่ผู้เขียนโปรแกรมกำหนดขึ้นเองก็ได้ นอกจากนี้ชนิดของข้อมูลยังอาจจะเป็นชนิดออบเจ็กต์หรือคลาสที่ Visual Basic รู้จัก เช่น Form , Textbox เป็นต้น

การประกาศตัวแปรสามารถทำได้ใน 2 ระดับคือ ระดับโมดูล (Module-level) และระดับโพรซีเยอร์ (Procedure –level) โดยการประกาศตัวแปรระดับโมดูลสามารถทำได้ด้วยการประกาศตัวแปรในส่วนประกาศของโมดูลนั้น ๆ ส่วนการประกาศตัวแปรระดับโพรซีเยอร์สามารถทำได้ด้วยการประกาศตัวแปรไว้ภายในโพรซีเยอร์นั้น ๆ

2.13 ขอบเขตของตัวแปร

ขอบเขต (Scope) ของตัวแปรคือ ตัวกำหนดว่าโค้ดโปรแกรมส่วนใดบ้างที่สามารถรับรู้ว่ามีตัวแปรนี้อยู่และสามารถใช้งานตัวแปรนี้ได้ สมมติว่าประกาศตัวแปรในโพรซีเยอร์ใดโพรซีเยอร์หนึ่ง ก็แสดงว่า

เฉพาะโค้ดโปรแกรมในโพรซีเยอร์ดังกล่าวเท่านั้นที่สามารถรับรู้และใช้งานตัวแปรตัวนั้นได้ เรียกตัวแปรแบบนี้ว่าเป็นแบบ Local ต่อโพรซีเยอร์นั้น ๆ สามารถกำหนดขอบเขตของตัวแปรได้ในขั้นตอนการประกาศตัวแปร (Variable Declaration)

ในการประกาศตัวแปรสามารถใช้คำว่า Private และ Public เป็นตัวกำหนดขอบเขตของตัวแปร การใช้คำสองคำนี้ในระดับโพรซีเยอร์และระดับโมดูลนั้นให้ผลต่างกัน ดังตารางต่อไปนี้

ตารางที่ 2.4 แสดงระดับขอบเขตตัวแปร

ระดับขอบเขตของตัวแปร	Private	Public
ระดับโพรซีเยอร์	ตัวแปรจะเป็นที่รู้จักเฉพาะในโพรซีเยอร์นี้เท่านั้น	ไม่สามารถประกาศแบบ Public ได้ในระดับโพรซีเยอร์
ระดับโมดูล	ตัวแปรจะเป็นที่รู้จักเฉพาะในโมดูลที่ประกาศ	ตัวแปรจะเป็นที่รู้จักในทุกโมดูล

ตัวแปรที่ประกาศในระดับโพรซีเยอร์จะมีขอบเขตแบบ Private เท่านั้น ซึ่งตัวแปรที่ประกาศจะเป็นที่รู้จักเฉพาะกับโค้ดโปรแกรมที่บรรจุภายในโพรซีเยอร์ ซึ่งจะเรียกว่าเป็นตัวแปรแบบ Local

โดยปกติแล้วตัวแปรแบบ Local จะถูกสร้างขึ้นและใช้เก็บค่าเป็นการชั่วคราวเพื่อจุดประสงค์อย่างใดอย่างหนึ่ง เมื่อโพรซีเยอร์นั้นทำงานเสร็จ ตัวแปรนั้นก็จะหายไป แต่ถ้าหากต้องการให้ค่าของตัวแปรนั้นยังคงอยู่แม้ว่าโพรซีเยอร์จะสิ้นสุดการทำงาน จะใช้คำว่า Static แทนคำว่า Dim

หากประกาศตัวแปรในระดับโมดูลนั้น ก็หมายความว่า ตัวแปรที่ประกาศจะสามารถถูกโพรซีเยอร์ใดๆ ภายในโมดูลนั้นเปลี่ยนแปลงค่าตัวแปรได้ แต่โพรซีเยอร์ของโมดูลอื่นจะไม่สามารถทำได้ ในการประกาศตัวแปรระดับโมดูลเพื่อให้รู้จักเฉพาะโมดูลที่ประกาศนี้ ให้ใช้คีย์เวิร์ด Private แทนคำว่า Dim ในการประกาศตัวแปรในส่วนประกาศ ดังตัวอย่าง

```
Private intX As Integer
```

หากต้องการประกาศตัวแปรที่เป็นที่รู้จักของทุกโมดูลในโปรแกรม ก็สามารถทำได้ด้วยการประกาศตัวแปรในระดับโมดูลในส่วนประกาศโดยใช้คำว่า Public แทนคำว่า Private หรือ Dim เป็นผลให้ทุกโพรซีเยอร์ของโปรแกรมสามารถเปลี่ยนแปลงค่าตัวแปรนั้นได้ ดังตัวอย่าง

```
Public intX As Integer
```

2.14 ค่าคงที่ใน Visual Basic

ตัวแปรค่าคงที่ที่ Visual Basic รู้จัก แบ่งออกเป็น 2 ประเภท คือ

1. ตัวแปรค่าคงที่ของระบบ (Intrinsic or System-defined) ที่กำหนดโดยโปรแกรมหรือคอนโทรลใดๆ โปรแกรมประยุกต์แต่ละโปรแกรมต่างก็มีตัวแปรค่าคงที่ชุดหนึ่งของตัวเองอยู่แล้วจำนวนหนึ่งให้ใช้งานได้ทันทีโดยไม่ต้องประกาศ
2. ตัวแปรที่กำหนดเอง (User-defined) เป็นตัวแปรค่าคงที่ที่ผู้เขียนโปรแกรมบัญญัติขึ้นเองด้วยคำสั่ง Const

2.15 ชนิดของข้อมูลใน Visual Basic

ในการประกาศตัวแปรเพื่อทำให้ Visual Basic รู้จักและทำให้สามารถเรียกใช้งานตัวแปรนั้นได้ นอกจากผู้เขียนโปรแกรมจะต้องกำหนดชื่อของตัวแปรแล้ว ยังควรที่จะกำหนด ชนิดข้อมูล (Data Type) ของข้อมูลที่ตัวแปรนั้นจะถูกใช้เป็นตัวเก็บด้วย ชนิดของข้อมูลจะเป็นตัวกำหนดจำนวนบิตของหน่วยความจำของคอมพิวเตอร์ว่าจะมีจำนวนบิตเท่าไรในการเก็บค่าของตัวแปรชนิดนั้น ตัวแปรทุกตัวที่ถูกประกาศในภาษาคอมพิวเตอร์ทุกภาษาจะมีชนิดข้อมูลของมันเอง หากว่าผู้เขียนโปรแกรมไม่ได้กำหนดชนิดของข้อมูลไว้ Visual Basic จะกำหนดให้ตัวแปรนั้นมีชนิดข้อมูลเป็นแบบ Variant ซึ่งเป็นชนิดข้อมูลที่เก็บข้อมูลได้เกือบทุกชนิดข้อมูลใน Visual Basic อย่างไรก็ตามเพื่อความมีประสิทธิภาพในการทำงานของโปรแกรม ควรที่จะประกาศชนิดข้อมูลของตัวแปรให้ตรงกับข้อมูลที่ตัวแปรนั้นๆ จะต้องเก็บ

การกำหนดชนิดข้อมูลของตัวแปรสามารถทำได้ในขั้นตอนการประกาศตัวแปรนั้นๆ โดยใช้คีย์เวิร์ด As และตามด้วยชื่อชนิดข้อมูล ดังโค้ดโปรแกรมตัวอย่างที่แสดงการประกาศตัวแปรเป็นชนิดข้อมูล Integer , Double , String และ Currency ตามลำดับ

```
Private I As Integer
Dim Expense As Double
Static MyName As String
```

```
Public PayBill As Currency
```

นอกจากนี้ยังสามารถกำหนดชนิดข้อมูลของตัวแปรที่ต้องการให้มีขอบเขตตัวแปรเดียวกันได้หลายตัวแปรภายในบรรทัดเดียวกัน โดยใช้เครื่องหมายคอมมาเป็นตัวคั่น ดังโค้ดโปรแกรมตัวอย่าง

```
Private I As Integer , Expense As Double
Private MyName As String , PayBill As Cureency
Private MyTest , Amount , J As Integer
```

2.16 ชนิดข้อมูลแบบตัวเลข (Numeric)

ชนิดข้อมูลแบบตัวเลขใน Visual Basic มีอยู่ 5 ชนิด ดังตารางที่ 2.5

ตารางที่ 2.5 แสดงชนิดข้อมูล

ชนิดข้อมูล	จำนวนบิต	ช่วงค่าที่เป็นไปได้	ตัวอักขระพิเศษในการประกาศ
Integer	16 (2 ไบต์)	-32,768 ถึง 32,767	%
Long	32 (4 ไบต์)	-2,147,438,648 ถึง 2,147,438,647	&
Single	32 (4 ไบต์)	-3.402823E38 ถึง –1.401298E-45 (สำหรับจำนวนติดลบ) 1.401298E-45 ถึง 3.402823E38 (สำหรับจำนวนบวก)	!
Double	64 (8 ไบต์)	-1.79769313486232E308 ถึง -4.94065645841247E-324 4.94065645841247E-324 ถึง 1.79769313486232E308	#
Currency	64 (8 ไบต์)	-922,337,203,685,477.5808 ถึง 922,337,203,685,477.5807	@

ทุกชนิดข้อมูลเกี่ยวกับตัวเลขดังกล่าวนี้ ต่างก็ใช้พื้นที่ในหน่วยความจำสำหรับเก็บค่าตัวแปรน้อยกว่าชนิดข้อมูลแบบ Variant (ซึ่งใช้พื้นที่หน่วยความจำ 32 ไบต์ในการจัดเก็บข้อมูล) ดังนั้นถ้าหากทราบเป็นที่แน่นอนแล้วว่าตัวแปรที่สร้างขึ้นจะเก็บข้อมูลชนิดใด ก็ควรจะกำหนดชนิดข้อมูลให้ถูกต้องในขั้นตอนการประกาศตัวแปรนั้น

เช่นเดียวกันหากว่าข้อมูลที่ต้องการจัดเก็บในค่าของตัวแปรเป็นข้อมูลประเภทที่มีเลขทศนิยม (Fractional Number) ก็ควรจะกำหนดชนิดของข้อมูลให้เป็นแบบ Single หรือ Double เพราะเป็นชนิดข้อมูลที่สามารถเก็บข้อมูลตัวเลขที่เป็นทศนิยมได้ แม้ว่าชนิดข้อมูลแบบ (Variant) จะสมารถเก็บข้อมูลตัวเลขแบบทศนิยมได้เช่นกัน แต่ก็ใช้พื้นที่หน่วยความจำมากกว่าชนิดข้อมูล Single หรือ Double

2.17 ชนิดข้อมูลแบบไบต์ (Byte)

ถ้าหากต้องการประกาศตัวแปรที่ต้องการจะนำไปเก็บข้อมูลแบบไบนารี ก็สามารถกำหนดชนิดข้อมูลของตัวแปรนั้นให้เป็นอาร์เรย์ที่มีชนิดข้อมูลแบบ Byte ได้ เช่น เก็บข้อมูลภาพบิตแมปในบางกรณีจะใช้ตัวแปรชนิด Byte ในการเก็บข้อมูลไบนารีไว้ระหว่างการเปลี่ยนแปลงรูปแบบของไฟล์ เมื่อข้อความถูกเปลี่ยนจาก ANSI ไปเป็น Unicode ข้อมูลไบนารีในตัวแปรจะหายไป Visual Basic จะเปลี่ยนโค้ดของ ANSI กับ Unicode โดยอัตโนมัติเมื่อ

1. ทำการอ่านโค้ดจากไฟล์
2. ทำการเขียนบันทึกลงในไฟล์
3. การเรียกใช้งานไดนามิกลิงก์ไลบรารี
4. การเรียกใช้วิธีใช้งานและค่าคุณลักษณะของออบเจ็กต์

เครื่องหมายใดๆ ที่สามารถใช้ได้กับ Integer ก็สามารถใช้ได้กับข้อมูลชนิด Byte ยกเว้นเครื่องหมายติดลบ เพราะว่า Byte เป็นข้อมูลที่ไม่มีการใช้เครื่องหมาย โดยจะมีค่าอยู่ระหว่าง 0 – 255 จึงไม่มีค่าติดลบ

2.18 ชนิดข้อมูลแบบข้อความ (String)

หากว่าข้อมูลที่ตัวแปรหนึ่งๆ จะต้องจัดเก็บบรรจุไว้เป็นตัวอักษรเพียงอย่างเดียว โดยไม่มีตัวเลขผสมอยู่ด้วย ก็สามารถกำหนดชนิดข้อมูลของตัวแปรนั้นๆ เป็นแบบข้อความได้ด้วยรูปแบบการประกาศตัวแปรคล้ายกับการประกาศตัวแปรชนิดข้อมูลแบบตัวเลข ดังนี้

```
Private Str As String
```

จากนั้นสามารถกำหนดข้อความใดให้กับตัวแปรดังกล่าวก็ได้ ซึ่งอาจจะเป็นข้อความทั่วไปอย่างเช่น " Welcome to Visual Basic" หรือบรรจุชื่อฟังก์ชันและอาร์กิวเมนต์ก็ได้ เช่น

```
Str = "MyPicture"
Str = Right(Str,4)
```

ปกติแล้วชนิดข้อมูลแบบข้อความ จะเปลี่ยนแปลงความยาวของข้อความไปตามค่า String ใหม่ที่ตัวแปรนั้นถูกกำหนดให้จัดเก็บ แต่ก็สามารถจำกัดความยาวของข้อความที่จะถูกตัวแปรดังกล่าวสามารถจัดเก็บได้ ด้วยการเติมเครื่องหมายดอกจัน (*) และกำหนดจำนวนตัวอักษรสูงสุดของข้อความที่จะจัดเก็บได้ไว้หลังชนิดข้อมูลแบบข้อความในขั้นตอนการประกาศตัวแปร ดังตัวอย่าง

```
Dim UserName As String * 40
```

เมื่อเป็นเช่นนี้หากว่าข้อมูลที่ถูกจัดเก็บในตัวแปรชื่อ UserName มีจำนวนตัวอักษรน้อยกว่า 40 Visual Basic จะจัดการเติมช่องว่างหลังข้อความไปจนข้อความนั้นมีจำนวนตัวอักษรเท่ากับ 40 หากว่าข้อมูลที่ถูกจัดเก็บมีจำนวนตัวอักษรมากกว่า 40 ก็จะถูก Visual Basic ตัดออกโดยอัตโนมัติ ดังนั้นเพื่อให้การแสดงค่าของตัวแปรในกรณีแรกออกมาตรงกับความเป็นจริง Visual Basic จึงมีฟังก์ชันที่ชื่อว่า

Trim(String) ไว้สำหรับตัดช่องว่างทั้งด้านซ้ายและขวาของข้อความที่เป็นอาร์กิวเมนต์ออก

Rtrim(String) ไว้สำหรับตัดช่องว่างด้านขวาของข้อความที่เป็นอาร์กิวเมนต์ออก

2.19 ชนิดข้อมูลแบบบูลลีน (Boolean)

บางครั้งอาจจะต้องสร้างตัวแปรเพื่อใช้จัดเก็บข้อมูลที่มีค่าที่เป็นไปได้สองทางคือ จริงหรือเท็จ ก็ควรจะกำหนดชนิดข้อมูลของตัวแปรนั้นเป็นแบบ Boolean เมื่อตัวแปรชนิดข้อมูลแบบ Boolean ถูกประกาศขึ้น ค่าเริ่มต้นของค่าตัวแปรชนิดนี้จะถูก Visual Basic กำหนดให้เท่ากับ False ดังโค้ดโปรแกรมตัวอย่าง

```
Dim binLight As Boolean
        If Light.Open = 1 Then
        BinLight = True
End if
```

2.20 ชนิดข้อมูลแบบออบเจ็กต์ (Object)

ตัวแปรที่ถูกประกาศให้มีชนิดข้อมูลเป็นแบบ Object จะใช้พื้นที่หน่วยความจำ 32 บิตในการเก็บค่าตัวแปร ตัวแปรที่กำหนดให้มีชนิดข้อมูลเป็นแบบ Object นี้จะต้องตามด้วยการกำหนดค่าออบเจ็กต์นั้นด้วยเสมอในที่ใดที่หนึ่งของโค้ดโปรแกรมก่อนการใช้งานของตัวแปร การกำหนดค่าสำหรับออบเจ็กต์สามารถทำได้ด้วยการใช้

คีย์เวิร์ด Set ดังนี้

```
Dim objPicture As Object
Set objPicture = LoadPicture("c::\Graphics\Balloon.bmp")
```

2.21 ชนิดข้อมูลแบบ Variant

ชนิดข้อมูลแบบ Variant นี้จะเป็นชนิดข้อมูลเริ่มต้นของข้อมูลใดๆ ใน Visual Basic ถ้าหากผู้เขียนโปรแกรมไม่ได้กำหนดชนิดข้อมูลของตัวแปรที่ประกาศไว้ ซึ่งชนิดข้อมูลแบบ Variant นี้สามารถจะจัดเก็บข้อมูลชนิดอื่นๆ ใน Visual Basic ได้เกือบทุกชนิด และ Visual Basic จะเป็นตัวจัดการให้เอง เช่น

```
Dim MyValue
MyValue = "27"
MyValue = My – 25
MyValue = "U" & MyValue
```

2.22 อาร์เรย์ (Array)

อาร์เรย์ คือ ชุดของตัวแปรที่มีชื่อเดียวกันโดยมี เลขดัชนี (Index) เป็นตัวอ้างถึงตัวแปรแต่ละตัว การใช้ตัวแปรแบบอาร์เรย์ในการเก็บข้อมูลในบางกรณีช่วยให้การเขียนโค้ดโปรแกรมทำได้กระชับขึ้นกว่าการกำหนดตัวแปรทุกตัวสำหรับเก็บข้อมูลที่มีความสัมพันธ์กัน

อาร์เรย์มีเลขดัชนีเป็นตัวอ้างถึงตัวแปรแต่ละตัว โดยแต่ละตัวแปรในอาร์เรย์เรียกว่า Element จำนวน Element ของอาร์เรย์ใด ๆ หมายถึง จำนวนตัวแปรทั้งหมดของอาร์เรย์นั้น ๆ ซึ่งเลขดัชนีที่ใช้บ่งชี้ถึงตัวแปรแต่ละ Element นั้นจะมี ขอบบน (Upper Bound) เป็นตัวกำหนดเลขดัชนีสูงสุด และ ขอบล่าง (Lower Bound) เป็นตัวกำหนดเลขดัชนีต่ำสุด

สำหรับอาร์เรย์ใดๆ Visual Basic จะจองพื้นที่หน่วยความจำ 20 ไบต์ และอีก 4 ไบต์ต่อ 1 มิติ กับหน่วยความจำอีกจำนวนหนึ่งตามแต่ชนิดข้อมูลสำหรับเก็บค่าตัวแปรคูณกับจำนวน Element ของอาร์เรย์นั้น ซึ่งการสร้างอาร์เรย์ของข้อมูลใด ๆ ถือเป็นการใช้ความจำจำนวนมาก ดังนั้นจึงไม่ควรสร้างอาร์เรย์ของตัวแปรเกินความจำเป็นที่ใช้งานจริง เพราะจะทำให้การจัดการหน่วยความจำด้อยประสิทธิภาพลง

ตัวแปรทุกตัวในอาร์เรย์จะต้องมีชนิดข้อมูลเป็นแบบเดียวกัน ซึ่งถ้าหากกำหนดให้อาร์เรย์ใดเป็นชนิดข้อมูลแบบ Integer แสดงว่าตัวแปรทุกตัวในอาร์เรย์ก็จะสามารถเก็บข้อมูลได้เฉพาะเลขจำนวนเต็มเท่านั้น

อาร์เรย์ใน Visual Basic แบ่งเป็นออกเป็น 2 แบบคือ อาร์เรย์ที่มีขนาดคงที่ กับอาร์เรย์แบบไดนามิก ที่ขนาดของอาร์เรย์สามารถเปลี่ยนแปลงได้ระหว่างโปรแกรมกำลังทำงาน

ในการประกาศกลุ่มของตัวแปรให้เป็นแบบอาร์เรย์ สามารถทำได้ 3 รูปแบบดังต่อไปนี้

1. หากต้องการสร้างอาร์เรย์แบบ Public ก็ให้ใช้คำว่า Public เป็นตัวกำกับขอบเขตของตัวแปร และประกาศไว้ในส่วนประกาศของโมดูล
2. หากต้องการสร้างอาร์เรย์ระดับโมดูลแต่เป็น Local ต่อโมดูลนั้น ก็ให้ใช้คำว่า Private เป็นตัวกำกับขอบเขตตัวแปร และประกาศไว้ในส่วนประกาศของโมดูลนั้น
3. หากต้องการสร้างอาร์เรย์ที่เป็น Local ต่อโพรซีเยอร์ ก็ให้ใช้คำว่า Private เป็นตัวกำกับขอบเขตตัวแปร และประกาศไว้ในส่วนประกาศของโพรซีเยอร์นั้น

หลักไวยากรณ์ในการประกาศกลุ่มตัวแปรเป็นแบบอาร์เรย์ก็กระทำเช่นเดียวกับการประกาศตัวแปรทั่วไป แต่ต้องกำหนดขอบบนของอาร์เรย์นั้นไว้ข้างหลังชื่อตัวแปรภายในเครื่องหมายวงเล็บ เพื่อให้ Visual Basic รับรู้ว่าเป็นการประกาศอาร์เรย์ไม่ใช่ตัวแปร โดยขนาดของอาร์เรย์ดังกล่าวจะเท่ากับขอบบนของอาร์เรย์บวกด้วย 1 ดังแสดงตัวอย่างในการประกาศกลุ่มตัวแปรแบบอาร์เรย์ ดังนี้

```
Dim Counters(14) As Integer
Dim Sums(20) As Double
```

หรืออาจจะใช้คำว่า Public หรือ Private แทนคำว่า Dim ก็ได้ อาร์เรย์ของตัวแปรที่ประกาศนี้ ตัวแปรแต่ละตัวจะมีชื่อเดียวกันคือ Counters แต่จะมีเลขดัชนีต่างกัน โดยปกติจะเริ่มต้นที่ 0 จนถึงขอบบน ในที่นี้คือ 14 เมื่อนับจำนวน Element จะได้ทั้งหมด 15 Element

อาร์เรย์ที่ปรากฏในโค้ดโปรแกรมดังตัวอย่างข้างต้นนั้นเรียกว่า อาร์เรย์แบบหนึ่งมิติ หมายความว่าสามารถเก็บเฉพาะข้อมูลที่เกี่ยวข้องกับค่าของข้อมูลได้เพียงค่าเดียว แต่ในบางครั้งอาจจะต้องการเก็บข้อมูลที่เกี่ยวข้องกับค่าที่ต้องเกี่ยวข้องกับค่าของข้อมูลสองค่าภายในเวลาเดียวกันก็สามารถสร้างอาร์เรย์แบบสองมิติ เพื่อเก็บค่าเหล่านี้ได้เช่นกัน ด้วยการใช้เครื่องหมายคอมมาคั่นระหว่างมิติ ยกตัวอย่างเช่น การประกาศอาร์เรย์แบบสองมิติขนาด 10 X 10 สามารถทำได้ดังโค้ดโปรแกรมตัวอย่าง

```
Static MatrixA(9,9) As Double
```

จำนวน Element ของอาร์เรย์หลายมิติสามารถหาได้จากการนำเอามิติแต่ละมิติของอาร์เรย์มาคูณกัน หมายความว่าอาร์เรย์แบบสองมิติในตัวอย่างก็จะมีทั้งหมด 100 Elements (10 X 10)

ในบางกรณีไม่สามารถรู้ขนาดแน่นอนของอาร์เรย์ได้ล่วงหน้า ก็สามารถประกาศอาร์เรย์นั้นให้เป็นอาร์เรย์แบบ ไดนามิก เพื่อให้อาร์เรย์ดังกล่าวสามารถเปลี่ยนขนาดของตัวมันเองขณะที่โปรแกรมทำงานได้

ในการสร้างอาร์เรย์แบบไดนามิก มีขั้นตอนแตกต่างจากการประกาศอาร์เรย์แบบปกติเล็กน้อย ตามขั้นตอนดังต่อไปนี้

1. ประกาศขอบเขตของอาร์เรย์ว่าเป็นแบบ Public หรือ Private ตามต้องการในตำแหน่งการประกาศที่ถูกต้อง อาร์เรย์สามารถประกาศให้เป็นแบบ Static ในโพรซีเยอร์หนึ่ง ๆ ได้ด้วยการใช้คำว่า Static โดยที่ภายในวงเล็บซึ่งควรจะเป็นขอบล่างและ/หรือขอบบนของเลขดัชนีอาร์เรย์ให้ปล่อยเว้นว่างเอาไว้
2. เมื่อรู้ขนาดของอาร์เรย์หรือต้องการกำหนดขนาดของอาร์เรย์ในขณะโปรแกรมทำงานก็สามารถกำหนดขนาดของอาร์เรย์ดังกล่าวได้ด้วยคำสั่ง ReDim อาร์เรย์นั้น ๆ อย่างเช่น

```
ReDim DynArray(4 to 12)
```

อย่างไรก็ตามในโพรซีเยอร์หนึ่ง ๆ จะปรากฏคำสั่ง ReDim เพื่อกำหนดขนาดของอาร์เรย์ได้เพียงแห่งเดียวเท่านั้น คำสั่ง ReDim นั้นเป็นประโยคโปรแกรมที่จะถูกเรียกใช้งานขณะโปรแกรมทำงาน โดยจะเป็นตัวบอกให้ Visual Basic รับรู้ว่าผู้เขียนโปรแกรมกำลังจะกำหนด ขอบล่างและ/หรือขอบบนของอาร์เรย์ในแต่ละมิติของอาร์เรย์แบบไดนามิกที่ประกาศไว้แล้ว

คำสั่ง ReDim นอกจากจะเปลี่ยนขนาดของอาร์เรย์ด้วยตัวเลขโดยตรงได้แล้ว ยังสามารถเปลี่ยนขนาดขออาร์เรย์ด้วยค่าของตัวแปรได้ด้วย อย่างเช่น

```
ReDim Matrix1(X,Y)
ReDim DynArray(X +1)
```

ในบางกรณีเพียงแค่ต้องการปรับขนาดของอาร์เรย์ใหม่เพื่อความมีประสิทธิภาพในการใช้งานหน่วยความจำโดยยังต้องการรักษาข้อมูลที่อยู่ในอาร์เรย์นั้นไว้ ก็สามารถทำได้ด้วยการใช้คำสั่ง ReDim พร้อมกับคีย์เวิร์ด Preserve ดังแสดงให้เห็นในโค้ดโปรแกรมตัวอย่าง ซึ่งแสดงการขยายขนาดอาร์เรย์ไป 1 Element โดยที่ไม่ทำให้ข้อมูลในอาร์เรย์เดิมสูญหายด้วยฟังก์ชัน UBound โดยที่ฟังก์ชัน UBound เป็นฟังก์ชันที่อ้างถึงขอบบนของอาร์เรย์นั้น ๆ

```
ReDim Preserve DynArray(Ubound(DynArray) + 1)
```

สำหรับอาร์เรย์แบบหลายมิติ ผู้เขียนโปรแกรมจะสามารถเปลี่ยนของบนของอาร์เรย์ได้เพียงขอบบนของมิติ สุดท้ายเท่านั้นด้วยคำสั่ง Preserve เช่นเดียวกัน ดังเช่นโค้ดโปรแกรมตัวอย่างนี้

```
ReDim Preserve Matrix(10, UBound(Matrix,2) + 1)
ReDim  Preserve Matrix(UBound(Matrix,1) + 1,10)
```

2.23 โพรซีเยอร์ต่าง ๆ ใน Visual Basic

การเขียนโค้ดโปรแกรมใน Visual Basic จะแบ่งเป็นส่วนย่อย ๆ ที่เรียกว่า "โพรซีเยอร์" โดยที่ในแต่ละโพรซีเยอร์ก็บรรจุโค้ดโปรแกรมเพื่อทำงานอย่างใดอย่างหนึ่ง ซึ่งอาจจะเกี่ยวข้องหรือไม่เกี่ยวข้องกับโพรซีเยอร์อื่นก็ได้ การเขียนโค้ดโปรแกรมที่ถูกแบ่งเป็นส่วน ๆ เช่นนี้มีคุณประโยชน์สำคัญในการทำงานของโปรแกรมอยู่ 2 ประการ คือ

1. การเขียนโปรแกรมแบบนี้จะทำให้แต่ละโพรซีเยอร์มีความหมายในตัวเอง ทำให้ง่ายต่อการทำความเข้าใจ และแก้ไขข้อผิดพลาดที่เกิดขึ้น
2. โพรซีเยอร์ที่ถูกสร้างขึ้นสำหรับใช้ในโปรแกรมหนึ่ง สามารถนำไปใช้ในโปรแกรมอื่น ๆ ได้เลยโดยไม่ต้องทำการแก้ไขหรือแก้ไขเพิ่มเติมเพียงเล็กน้อย

ใน Visual Basic มีรูปแบบโพรซีเยอร์อยู่ด้วยกัน 3 ชนิด คือ

1. ซับรูทีน (Subroutine) คือโพรซีเยอร์ที่ไม่มีการคืนค่า (Return Value) ใด ๆ
2. ฟังก์ชัน (Function) คือโพรซีเยอร์ที่มีกี่คืนค่าอย่างใดอย่างหนึ่งกลับมา
3. Property Procedure ซึ่งสามารถคืนค่า และกำหนดค่าตัวแปร (Assign Values) หรือกำหนดค่าอ้างอิง (Set References) ให้กับออบเจ็กต์ใน Visual Basic

2.24 โพรซีเยอร์ทั่วไป (General Procedure)

โพรซีเยอร์ทั่วไป หมายถึง โพรซีเยอร์ที่บรรจุประโยคโปรแกรมที่ทำงานอย่างใดอย่างหนึ่ง โพรซีเยอร์ชนิดนี้จะเริ่มทำงานด้วยตัวของมันเอง หรือถูกเรียกไปใช้งานโดยโพรซีเยอร์อื่นก็ได้ ในขณะที่โพรซีเยอร์เหตุการณ์ตอบสนองจะไม่ถูกเรียกขึ้นมาทำงานจนกว่าจะเกิดเหตุการณ์ตอบสนองที่เป็นตัวเรียกโพรซีเยอร์นั้น ๆ ขึ้นมาทำงาน

จุดประสงค์หลักในการสร้างโพรซีเยอร์ทั่วไปขึ้นมาทำงานก็คือเพื่อทำงาน จะเกิดขึ้นบ่อยครั้งเมื่อโพรซีเยอร์เหตุการณ์ตอบสนองต่าง ๆ ถูกเรียกขึ้นมาทำงาน ดังนั้นแทนที่จะเขียนโค้ดโปรแกรมเพื่อทำงานเหล่านั้นในทุก ๆ โพรซีเยอร์เหตุการณ์ตอบสนอง ก็สร้างโพรซีเยอร์ทั่วไปแล้วเขียนประโยคโปรแกรมในโพรซีเยอร์เหตุการณ์ตอบสนอง เพื่อเรียกโพรซีเยอร์ทั่วไปนี้ขึ้นมาทำงานแทน เป็นการหลีกเลี่ยงการเขียนโค้ดซ้ำซ้อน และทำให้ง่ายต่อการปรับปรุงโค้ดโปรแกรม โพรซีเยอร์ทั่วไปมีอยู่ 2 ชนิดคือ โพรซีเยอร์ซับรูทีน และโพรซีเยอร์ฟังก์ชัน

2.25 โพรซีเยอร์ซับรูทีน (Subroutine)

โพรซีเยอร์ซับรูทีน คือกลุ่มของโค้ดโปรแกรมที่จะถูกเรียกขึ้นมาทำงานเมื่อโพรซีเยอร์ถูกเรียกใช้งาน เมื่อทำงานตามประโยคโปรแกรมที่บรรจุในโพรซีเยอร์เรียบร้อยแล้วก็สิ้นสุดการทำงาน ไม่มีการคืนค่าใด ๆ กลับมา

หลักไวยากรณ์ในการสร้างโพรซีเยอร์แบบซับรูทีน เป็นดังนี้

```
[Private|Public][Static]Sub procedurename (arguments)
        statemens
End Sub
```

Private/Public เป็นตัวกำกับขอบเขตของโพรซีเยอร์ซับรูทีน

Static เป็นตัวกำกับว่าตัวแปรทุกตัวในโพรซีเยอร์เป็นแบบ Static หรือไม่

Sub เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกชนิดของโพรซีเยอร์ว่าเป็นซับรูทีน

procedurename เป็นชื่อของโพรซีเยอร์

(arguments) เป็นรายชื่ออาร์กิวเมนต์ของโพรซีเยอร์นั้น

statements เป็นประโยคโปรแกรมซึ่งจะถูกเรียกขึ้นมาทำงานเมื่อโพรซีเยอร์เริ่มทำงาน

End Sub เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกว่าเป็นจุดสิ้นสุดการทำงานของโพรซีเยอร์ซับรูทีน

2.26 โพรซีเยอร์ฟังก์ชัน (Function)

ฟังก์ชัน คือ กลุ่มของโค้ดโปรแกรมที่จะถูกเรียกขึ้นมาทำงานเมื่อโพรซีเยอร์ถูกเรียกใช้งาน เมื่อทำงานตามประโยคโปรแกรมที่บรรจุอยู่ในโพรซีเยอร์เรียบร้อยแล้วก็จะสิ้นสุดการทำงาน และก็จะมีการคืนค่าใดค่าหนึ่งกลับมา โดยขึ้นอยู่กับการกำหนดในฟังก์ชันนั้นๆ ว่าจะคืนค่าใด

Visual Basic มีฟังก์ชันพื้นฐานอยู่มากมาย อย่างเช่น Abs สำหรับการหาค่า Absolute ของตัวเลข เป็นต้น นอกจากจะเรียกใช้ฟังก์ชันพื้นฐานต่าง ๆ เหล่านี้แล้ว ผู้เขียนโปรแกรมสามารถสร้างฟังก์ชันขึ้นเพื่อใช้ในโปรแกรมด้วยตัวเองได้

หลักไวยากรณ์ในการสร้างฟังก์ชัน เป็นดังนี้

```
[Private|Public][Static]Function procedurename (arguments) [As type]
        statements
End Function
```

Private/Public เป็นตัวกำกับขอบเขตของโพรซีเยอร์ฟังก์ชัน

Static เป็นตัวกำกับว่าตัวแปรทุกตัวในโพรซีเยอร์เป็นแบบ Static หรือไม่

Function เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกชนิดของโพรซีเยอร์ว่าเป็นฟังก์ชัน

procedurename เป็นชื่อของโพรซีเยอร์

(arguments) เป็นรายชื่ออาร์กิวเมนต์ของโพรซีเยอร์ฟังก์ชันนั้น ๆ

[As type] กำหนดชนิดข้อมูลของค่าที่โพรซีเยอร์ฟังก์ชันนั้นคืนกลับมา

statements เป็นประโยคโปรแกรมซึ่งจะถูกเรียกขึ้นมาทำงานเมื่อโพรซีเยอร์เริ่มทำงาน

End Function เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกว่าเป็นจุดสิ้นสุดการทำงานของโพรซีเยอร์ฟังก์ชัน

โพรซีเยอร์ซับรูทีนกับโพรซีเยอร์ฟังก์ชันมีความเหมือนกันอยู่หลายประการคือ เป็นส่วนบรรจุโค้ดสำหรับทำงานอย่างใดอย่างหนึ่ง สามารถรับค่าอาร์กิวเมนต์เพื่อทำงานได้ อย่างไรก็ตามการเรียกใช้งานโพรซีเยอร์ฟังก์ชันมีข้อแตกต่างกับการเรียกใช้งานโพรซีเยอร์ซับรูทีนอยู่ 3 ประการคือ

1. รูปแบบการเรียกโพรซีเยอร์ซับรูทีนสามารถเรียกชื่อโพรซีเยอร์ได้โดยตรงและตามด้วยอาร์กิวเมนต์ ในขณะที่การเรียกใช้งานโพรซีเยอร์ฟังก์ชันมักจะอยู่ทางด้านขวาของเครื่องหมายกำหนดค่า (=) โดยที่ทางด้านซ้ายของเครื่องหมายกำหนดค่าคือ ค่าที่โพรซีเยอร์ฟังก์ชันนั้นคืนกลับมา โดยปกติแล้วมักจะสร้างตัวแปรขึ้นรับค่าคืนกลับ หรือนำค่าคืนกลับนั้นไปใช้ทันที
2. โพรซีเยอร์ฟังก์ชันมีชนิดข้อมูลเป็นของตัวเองเหมือนกับตัวแปรทั่วไป เพื่อเป็นตัวกำหนดค่าข้อมูลที่เป็นค่าคืนกลับ หากว่าผู้เขียนไม่ได้กำหนดชนิดข้อมูลไว้ Visual Basic จะกำหนดชนิดข้อมูลให้เป็นแบบ Variant โดยปริยาย
3. ในการกำหนดค่าวนกลับ สามารถทำได้ด้วยการกำหนดค่าคืนกลับให้ชื่อโพรซีเยอร์ฟังก์ชันในโพรซีเยอร์ฟังก์ชันนั้น ๆ เอง

2.27 โพรซีเยอร์เหตุการณ์ตอบสนอง

เมื่อเกิดเหตุการณ์ตอบสนอง (Event) ขึ้นกับออบเจ็กต์ใด ๆใน Visual Basic และออบเจ็กต์นั้น ๆ สามารถรับรู้เหตุการณ์ตอบสนองดังกล่าวได้ ประโยคโปรแกรมที่บรรจุอยู่ในโพรซีเยอร์ของเหตุการณ์ตอบสนองของออบเจ็กต์นั้น ๆ จะถูกเรียกขึ้นมาทำงานทันที เนื่องจากว่าโพรซีเยอร์ชนิดนี้จะต้องเกี่ยวข้องกับฟอร์ม และ คอนโทรลใน Visual Basic จึงกล่าวได้ว่าโพรซีเยอร์ชนิดนี้จะตั้งอยู่ในส่วนโค้ดโปรแกรมของฟอร์ม และ คอนโทรลต่าง ๆ

หลักไวยากรณ์ของการสร้างโพรซีเยอร์เหตุการณ์ตอบสนอง เป็นดังนี้

1. ชื่อโพรซีเยอร์ของเหตุการณ์ตอบสนองของคอนโทรลใด ๆ จะเริ่มต้นด้วยชื่อของคอนโทรล (Object Name) ตามด้วยเครื่องหมายขีดล่าง และชื่อของเหตุการณ์ตอบสนอง (Event Name)

2. ชื่อของโพรซีเยอร์เหตุการณ์ตอบสนองของฟอร์มจะเริ่มต้นด้วยคำว่า “Form” ตามด้วยเครื่องหมายขีดล่างและชื่อของเหตุการณ์ตอบสนอง

หลักไวยากรณ์ของโพรซีเยอร์เหตุการณ์ตอบสนองของคอนโทรล เป็นดังนี้

```
Private Sub controlname_eventname (arguments)
        statements
End Sub
```

หลักไวยากรณ์ของโพรซีเยอร์เหตุการณ์ตอบสนองของฟอร์ม เป็นดังนี้

```
Private Sub Form_eventname (arguments)
        statements
End Sub
```

Private เป็นคำกำกับขอบเขตของโพรซีเยอร์เหตุการณ์ตอบสนองซึ่งจะเป็นแบบ Private เท่านั้น เพราะว่ามีเพียง

โมดูลเจ้าของโพรซีเยอร์เท่านั้นที่จะสามารถเรียกใช้งานโพรซีเยอร์นี้ได้

Sub เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกชนิดของโพรซีเยอร์ว่าเป็นเป็นซับรูทีน

controlname เป็นชื่อของคอนโทรล

Form เป็นชื่อบังคับสำหรับโมดูลฟอร์ม

ขีดล่าง (_) เป็นหลักไวยากรณ์ของ Visual Basic ในการสร้างโพรซีเยอร์ชนิดนี้

eventname เป็นชื่อของเหตุการณ์ตอบสนองต่อคอนโทรลที่คอนโทรลนั้น ๆ ยอมรับได้

(arguments) เป็นรายชื่ออาร์กิวเมนต์ของโพรซีเยอร์นั้น

statements เป็นประโยคโปรแกรมซึ่งจะถูกเรียกขึ้นมาทำงานเมื่อโพรซีเยอร์เริ่มทำงาน

End Sub เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกว่าเป็นจุดสิ้นสุดการทำงานของโพรซีเยอร์ซับรูทีน

2.28 การเรียกใช้งานโพรซีเยอร์

ในการเรียกใช้โพรซีเยอร์มาใช้งานนั้นแตกต่างกันไปตามแต่ละชนิดของโพรซีเยอร์ โดยแบ่งเป็นการเรียกใช้งานโพรซีเยอร์ซับรูทีน การเรียกใช้งานโพรซีเยอร์ฟังก์ชัน และการเรียกใช้งานโพรซีเยอร์ที่อยู่ในโมดูลอื่น ๆ

2.29 การเรียกใช้งานโพรซีเยอร์ซับรูทีน

ในการเรียกใช้โพรซีเยอร์ซับรูทีนมีความแตกต่างกับการเรียกใช้โพรซีเยอร์ฟังก์ชันตรงที่ โพรซีเยอร์ซับรูทีนไม่สามารถถูกเรียกในระหว่างประโยคโปรแกรมได้เหมือนโพรซีเยอร์ฟังก์ชัน เช่นการเรียกโพรซีเยอร์ฟังก์ชันดังตัวอย่างนี้

```
Label1.Caption = Abs(-35)
```

ดังจะเห็นได้ว่าโพรซีเยอร์ฟังก์ชัน Abs() สามารถถูกเรียกใช้งานในระหว่างประโยคโปรแกรมได้ สำหรับการเรียกใช้งานโพรซีเยอร์ซับรูทีนสามารถทำได้ 2 รูปแบบ ก็คือใช้คำสั่ง Call ในการเรียกโพรซีเยอร์ หรือการเรียกชื่อโพรซีเยอร์โดยตรงตามด้วยอาร์กิวเมนต์ของโพรซีเยอร์นั้น ตัวอย่างโค้ดโปรแกรมด้านล่างจะแสดงการเรียกโพรซีเยอร์ซับรูทีนที่ชื่อว่า MyProc โดยมีอาร์กิวเมนต์ 2 ตัว ใช้ชื่อว่า FirstArgument และ SecondArgument ตามลำดับ 2 รูปแบบที่ให้ผลเหมือนกันทุกประการ

```
Call MyProc (FirstArgument, SecondArgument)
MyProc FirstArgument, SecondArgument
```

2.30 การเรียกใช้งานโพรซีเยอร์ฟังก์ชัน

ในการเรียกใช้งานโพรซีเยอร์ฟังก์ชันที่ผู้เขียนโปรแกรมเป็นผู้สร้างขึ้นเอง สามารถกระทำได้เช่นเดียวกับการเรียกฟังก์ชันพื้นฐานของ Visual Basic โดยการระบุชื่อโพรซีเยอร์ฟังก์ชัน และค่าอาร์กิวเมนต์สำหรับฟังก์ชันนั้น ๆ ในประโยคโปรแกรม ตัวอย่างโค้ดโปรแกรมข้างล่างแสดงการเรียกใช้โพรซีเยอร์ฟังก์ชันที่ชื่อว่า Profit โดยส่งค่าอาร์กิวเมนต์ 2 ตัว คือ Revenue และ Cost เพื่อเก็บเป็นค่าตัวแปรชื่อ ThisYear จากนั้นนำไปบวกกับค่าของตัวแปร LastYear และแสดงเป็นข้อความบนคอนโทรลช่องแสดงข้อความชื่อ Text1

```
Dim ThisYear As Integer
Dim LastYear As Integer
LastYear = 50
ThisYear = Profit(100 , 60)
Text1.Text = LastYear + ThisYear
```

2.31 การเรียกใช้งานโพรซีเยอร์ต่างโมดูล

ในการเรียกใช้งานโพรซีเยอร์ที่อยู่ในโมดูลฟอร์มจากโมดูลอื่นสามารถทำได้โดยระบุชื่อโมดูลฟอร์มเจ้าของโพรซีเยอร์โดยสมมติชื่อฟอร์มของโมดูลฟอร์มว่า Form1 และสมมติชื่อฟังก์ชันว่า MyArea

```
Call Form1.MyArea(arguments)
```

ในการเรียกใช้งานโพรซีเยอร์ฟังก์ชันที่อยู่ในโมดูลมาตรฐาน หากว่าโพรซีเยอร์นั้นไม่ซ้ำกับโพรซีเยอร์ใด ๆ ในโมดูลตัวเอง และโมดูลอื่น ๆ ในโปรแกรมเลย ผู้เขียนโปรแกรมสามารถเรียกใช้งานโพรซีเยอร์เหล่านั้นได้โดยตรง โดยไม่จำเป็นต้องระบุโมดูลเจ้าของแต่อย่างใด แม้ว่าโพรซีเยอร์ที่เรียกใช้โพรซีเยอร์ฟังก์ชันดังกล่าวจะอยู่ต่าง โมดูลกันก็ตาม

แต่ถ้าหากโพรซีเยอร์ฟังก์ชันที่ต้องการเรียกใช้งานมีชื่อซ้ำกัน 2 โพรซีเยอร์ (สมมติว่าชื่อ MyFunction) ปรากฏอยู่ในโมดูลมาตรฐาน 2 โมดูล หากต้องการจะเรียกใช้โพรซีเยอร์ฟังก์ชันนี้จากโมดูลอื่น จะต้องระบุชื่อโมดูลมาตรฐานเจ้าของโพรซีเยอร์ที่เรียกใช้งานด้วย อย่างเช่น

```
Call Module1.MyFunction(arguments)
Call Module2.MyFunction(arguments)
```

ถ้าหากเรียกโพรซีเยอร์ MyFunction ใน Module1 จากโพรซีเยอร์ที่อยู่ใน Module1 เอง หากว่าไม่ได้กำหนดชื่อโมดูลเจ้าของ Visual Basic จะถือว่าเรียกใช้โพรซีเยอร์ฟังก์ชัน MyFunction ที่อยู่ใน Module1 เอง เช่น

```
MyFunction arguments
```

ประโยคโปรแกรมข้างต้นนี้หากปรากฏใน Module1 ก็จะเรียกใช้โพรซีเยอร์ฟังก์ชัน MyFunction ใน Module1 แต่หากปรากฏใน Module2 ก็จะไปเรียกใช้โพรซีเยอร์ฟังก์ชัน MyFunction ใน Module2

2.32 การส่งค่าอาร์กิวเมนต์ให้โพรซีเยอร์

โดยทั่วไปแล้วโพรซีเยอร์ทั้งแบบซับรูทีนและฟังก์ชันมักจะต้องอาศัยข้อมูลสำหรับทำงานโดยรับรู้ข้อมูลดังกล่าวผ่านค่าของตัวแปรจำนวนหนึ่ง ซึ่งตัวแปรที่ทำหน้าที่ส่งข้อมูลให้กับโพรซีเยอร์เหล่านั้นเรียกว่า อาร์กิวเมนต์ (Argument)

ในฐานะที่อาร์กิวเมนต์เป็นตัวแปรชนิดหนึ่ง จึงมีชนิดข้อมูลเป็นตัวกำหนดชนิดข้อมูลของอาร์กิวเมนต์ตัวนั้น โค้ดโปรแกรมต่อไปนี้คือ ตัวอย่างของโพรซีเยอร์ฟังก์ชันชื่อ RectArea ซึ่งมีอาร์กิวเมนต์ 2 ตัวชื่อ width และ height ที่มีชนิดข้อมูลเป็นแบบ Integer

```
Function RectArea (width As Integer , height As Integer) As Integer
```

```
End Function
```

การส่งค่าของอาร์กิวเมนต์ให้กับโพรซีเยอร์สามารถทำได้ 2 รูปแบบคือ By Value (ส่งค่าของตัวแปร) กับ By Reference (ส่งค่าตำแหน่งที่อยู่ในหน่วยความจำของตัวแปร) ให้โพรซีเยอร์รับรู้

ในการส่งค่าของอาร์กิวเมนต์แบบ By Value นั้น Visual Basic จะทำการจัดหาพื้นที่ว่างในหน่วยความจำขึ้นใหม่เพื่อเก็บ ค่าสำเนา (Copy) ของอาร์กิวเมนต์นั้นไว้ แล้วจึงส่งค่านั้นไปยังโพรซีเยอร์ หากว่าโพรซีเยอร์ดังกล่าวทำการเปลี่ยนแปลงค่าของอาร์กิวเมนต์ที่ได้รับ การเปลี่ยนแปลงนั้นก็จะเกิดขึ้นกับค่าสำเนาที่สร้างขึ้นใหม่เท่านั้น โดยที่ค่าของตัวอาร์กิวเมนต์จริงยังคงเท่าเดิม เพราะตำแหน่งที่อยู่ในหน่วยความจำของอาร์กิวเมนต์กับอาร์กิวเมนต์สำเนาเป็นคนละตำแหน่งกัน

ในการสร้างโพรซีเยอร์ ผู้เขียนโปรแกรมสามารถกำหนดได้ว่าต้องการให้อาร์กิวเมนต์ใดมีการส่งค่าแบบใด ดังโค้ดโปรแกรมตัวอย่างแสดงการสร้างโพรซีเยอร์ซับรูทีนที่มีการส่งค่าอาร์กิวเมนต์แบบ By Value

```
Sub PlotPixel(ByVal X As Single, ByVal Y As Single)


End Sub
```

สำหรับการส่งค่าอาร์กิวเมนต์แบบ By Reference หมายความว่า ค่าอาร์กิวเมนต์ที่ส่งให้โพรซีเยอร์นั้นคือตำแหน่งที่อยู่ในหน่วยความจำของอาร์กิวเมนต์ ฉะนั้นหากว่าโพรซีเยอร์ดังกล่าวมีการเปลี่ยนแปลงค่าอาร์กิวเมนต์ จะทำให้อาร์กิวเมนต์ถูกเปลี่ยนแปลงค่าไปอย่างถาวร หากว่าผู้เขียนโปรแกรมไม่ได้กำหนดว่าการส่งค่าอาร์กิวเมนต์ให้แก่โพรซีเยอร์เป็นแบบใด Visual Basic จะยึดเอาว่าเป็นการส่งค่าอาร์กิวเมนต์แบบ By Reference เป็นค่าเริ่มต้น ตัวอย่างการกำหนดการส่งค่าอาร์กิวเมนต์ให้โพรซีเยอร์แบบ By Reference แสดงไว้ในโค้ดโปรแกรมด้านล่างนี้

```
Sub Drawline(ByRef X Asa Single, ByRef Y As Single)


End Sub
```

นอกจากการส่งค่าอาร์กิวเมนต์เป็นค่าตัวเลขเดี่ยว ๆ แล้ว ยังสามารถส่งค่าอาร์กิวเมน์แบบเป็นประโยคโปรแกรมได้ด้วย โดย Visual Basic จะแปลงชนิดข้อมูลให้ตรงกับอาร์กิวเมนต์ที่โพรซีเยอร์นั้นต้องการ โค้ดโปรแกรมต่อไปนี้แสดงการส่งค่าตัวแปรที่ประกาศไว้เป็นแบบ Long ไปยังโพรซีเยอร์ที่ต้องการชนิดของอาร์กิวเมนต์แบบ String

```
Sub MyTest()
```

```
    Dim X As Long
    X = 50 Mod 15
    MyTest2(X)

End Sub

Sub MyTest2(Str As String)
    Print Str
End Sub
```

จะเห็นได้ว่าค่าของตัวแปร X จะถูกแปลงชนิดข้อมูลจาก Long ให้เป็นแบบ String เพื่อให้โพรซีเยอร์ชื่อ MyTest2 นำไปแสดงด้วยคำสั่ง Print

ในบางโพรซีเยอร์สามารถกำหนดอาร์กิวเมนต์ให้เป็นแบบ Optional ได้ ซึ่งหมายถึงว่า ในการเรียกใช้โพรซีเยอร์นั้นๆ ผู้เขียนโปรแกรมอาจจะส่งค่าอาร์กิวเมนต์หรือไม่ส่งค่าอาร์กิวเมนต์ดังกล่าวในการเรียกครั้งนั้นก็ได้ การกำหนดสามารถทำได้โดยใช้คีย์เวิร์ด Optional วางไว้ตรงหน้าชื่อของอาร์กิวเมนต์ในโพรซีเยอร์ ดังแสดงในโค้ดโปรแกรมตัวอย่าง

```
Sub DrawCircle(X As Single, Y Single, Optional Color  As Integer)

End Sub
```

แสดงว่าในการเรียกใช้ซับรูทีน DrawCircle จะต้องส่งค่าอาร์กิวเมนต์ 2 ตัวเป็นอย่างต่ำ ก็คือค่า X และ Y โดยที่ผู้เรียกอาจจะส่งค่า Color หรือไม่ส่งไปด้วยก็ได้ อย่างเช่น

```
DrawCircle 10,20,vbRed          'ส่งค่าคงที่ vbRed สำหรับค่าอาร์กิวเมนต์ Color
DrawCircle 20,35                'ไม่ส่งค่าอาร์กิวเมนต์ Color
```

2.33 โครงสร้างประโยคโปรแกรมใน Visual Basic

2.33.1 โครงสร้างการตัดสินใจ

ใน Visual Basic มีรูปแบบการเขียนโค้ดโปรแกรมเกี่ยวกับโครงสร้างการตัดสินใจอยู่ 3 รูปแบบ คือ

โครงสร้างการตัดสินใจแบบ If...Then

โครงสร้างการตัดสินใจแบบ If...Then...Else

โครงสร้างการตัดสินใจแบบ Select Case

โครงสร้างการตัดสินใจแบบ If...Then

โครงสร้างการตัดสินใจแบบ If...Then จะทำการตรวจสอบเงื่อนไขที่อยู่หลังคำว่า If ว่าเป็นจริงหรือไม่ เมื่อเงื่อนไขเป็นจริง ประโยคโปรแกรมใดๆ ที่อยู่หลังคำว่า Then จะถูกเรียกขึ้นมาทำงาน การเขียนโครงสร้างการตัดสินใจชนิดนี้สามารถทำได้ 2 รูปแบบ คือ แบบบรรทัดเดี่ยว (Single-line) และแบบหลายบรรทัด (Multiple-line)

หลักไวยากรณ์ของการใช้งานโครงสร้างการตัดสินใจแบบ If...Then แบบบรรทัดเดี่ยว และแบบหลายบรรทัดเป็นดังตัวอย่างข้างล่างนี้ ตามลำดับ

```
If condition Then statement          'Single-line

If condition Then                    'Multiple-line
    Statementblock
End If
```

If เป็นคีย์เวิร์ดในภาษา Visual Basic เพื่อบอกให้ Visual Basic รับรู้ว่าผู้เขียนโปรแกรมจะใช้ประโยค If...Then

condition คือ เงื่อนไขที่โปรแกรมจะต้องทดสอบเพื่อกำหนดว่าจะทำงานหรือไม่

statement คือ ประโยคโปรแกรมที่จะถูกเรียกขึ้นมาทำงานหากว่าผลของการตรวจสอบเงื่อนไขเป็นจริง

statementblock คือ กลุ่มของประโยคโปรแกรมที่อยู่ระหว่างคำว่า Then กับ End If ในประโยค If...Then แบบหลายบรรทัด จะถูกเรียกขึ้นมาทำงานเมื่อเงื่อนไขเป็นจริง

End If เป็นคีย์เวิร์ดในภาษา Visual Basic กำหนดจุดสิ้นสุดของประโยค If...Then แบบหลายบรรทัด ในประโยค If...Then ประโยคโปรแกรมหลังคำว่า Then จะถูกเรียกขึ้นมาทำงานก็ต่อเมื่อภาวะของเงื่อนไขเป็นจริงเท่านั้น อย่างไรก็ตาม ภาวะที่เงื่อนไขเป็นจริงไม่ได้หมายความว่าค่าตรรกะของเงื่อนไขนั้นๆ จะต้องเป็น True ยกตัวอย่างเช่น ประโยคเงื่อนไขต่อไปนี้จะนำไปสู่การเรียกใช้งานประโยคโปรแกรมหลังคำว่า Then แม้ว่ารูปประโยคจะมีค่าเป็น False

```
If Label1.Visible = False Then Label1.Visible = True
```

2.33.1 .1 โครงสร้างการตัดสินใจแบบ If...Then...Else

ในบางครั้งอาจจะต้องการให้มีการทดสอบเงื่อนไขที่เป็นไปได้หลายเงื่อนไข อันนำไปสู่ผลลัพธ์ที่แตกต่างกัน อย่างเช่น การกดปุ่มเมาส์ที่มีทั้งปุ่มซ้าย ขวา และตรงกลาง ดังนั้นผู้เขียนโปรแกรมสามารถใช้ประโยค If...Then...Else ได้ดังมีรูปแบบไวยากรณ์ดังนี้

```
If condition1 Then
        statementblock1
[ElseIf condition2 Then
        statementblock2
[Else
        statementblockn
End If
```

ประโยค If...Then...Else มีรูปแบบการใช้งานคล้ายประโยค If...Then แบบหลายบรรทัด แต่อาจจะมีคีย์เวิร์ดใน Visual Basic เพิ่มขึ้นหนึ่งหรือสองคำคือ ElseIf และ Else ตามแต่ที่ผู้เขียนโปรแกรมกำหนด โดยขั้นตอนในการทำงานของประโยคโปรแกรมชนิดนี้จะเริ่มต้นที่การตรวจสอบเงื่อนไข condition1 ว่าเป็นจริงหรือไม่ หากว่าเป็นจริง statementblock1 จะถูกเรียกขึ้นมาทำงาน เมื่อเสร็จเรียบร้อยแล้วก็เป็นการสิ้นสุดการทำงานของประโยค If...Then...Else ทันที หากว่าเงื่อนไข condition1 ไม่เป็นจริง Visual Basic ก็จะข้าม statementblock1 มาทดสอบเงื่อนไข condition2 ที่อยู่หลังคีย์เวิร์ด ElseIf หากว่า condition2 เป็นจริง กลุ่มประโยคโปรแกรมใน statementblock2 จะถูกเรียกขึ้นมาทำงาน เมื่อเสร็จสิ้นแล้วก็จะจบการทำงานประโยค If ทันที

แต่ถ้าทั้งเงื่อนไข condition1 และ condition2 ไม่เป็นจริงทั้งคู่ กลุ่มประโยคที่อยู่หลังคำว่า Else และ End If จะถูกเรียกขึ้นมาทำงานและจบประโยค If...Then...Else เมื่อทำงานเสร็จ

2.33.1 .2 โครงสร้างการตัดสินใจแบบ Select Case

โครงสร้างการตัดสินใจแบบ Select Case เหมาะสำหรับใช้ทดสอบเงื่อนไขก่อนการทำงาน โดยที่เงื่อนไขที่ต้องการทดสอบนั้นมีความเป็นไปได้จำนวนมาก การใช้งานโครงสร้างแบบ Select Case มีหลักดังนี้

```
Select Case testexpression
        Case expressionlist1
                statementblock1
        Case expressionlist2
                statementblock2
        Case Else
                statementblockn
End Select
```

Select Case เป็นคีย์เวิร์ดใน Visual Basic เพื่อระบุจุดเริ่มต้นการใช้ประโยค Select Case
testexpression เป็นเงื่อนไขที่ต้องการจะทดสอบค่าความจริง
Case เป็นคีย์เวิร์ดใน Visual Basic เพื่อกำหนดค่าของเงื่อนไขที่จะนำไปทดสอบก่อนการทำงาน
expressionlist เป็นค่าของเงื่อนไขที่จะนำค่าของ testexpression มาเปรียบเทียบ
statementblock เป็นกลุ่มประโยคโปรแกรมที่จะถูกเรียกขึ้นทำงานหากว่า testexpression มีค่าเท่ากับค่า expressionlist ของแต่ละ Case
Case Else เป็นคีย์เวิร์ดใน Visual Basic เพื่อกำหนดค่าของเงื่อนไขอื่นๆ นอกเหนือจากที่กำหนดแล้วในแต่ละ Case
End Select เป็นคีย์เวิร์ดใน Visual Basic เพื่อแสดงจุดสิ้นสุดของประโยค Select Case

ลำดับการทำงานของโครงสร้างการตัดสินใจแบบ Select Case คือค่าของ testexpression จะถูกนำมาเก็บไว้ก่อนเป็นประการแรก จากนั้นก็เริ่มทดสอบกับค่าของ expressionlist ที่อยู่หลังคำว่า Case ในแต่ละ Case เพื่อเปรียบเทียบว่ามีค่าเท่ากับค่าของ expressionlist ใน Case ใด และจะทำการเรียก statementblock ใน Case นั้นๆ ขึ้นมาทำงานจนครบและจบการทำงานประโยค Select Case ถ้าหากว่าไม่มีค่าของ expressionlist ใดเลยที่เท่ากับค่าของ testexpression ประโยคโปรแกรมหลังคำว่า Case Else ก็จะถูกเรียกขึ้นมาทำงานและจบการทำงานของประโยค Select Case

2.33.2 วงรอบการทำงานซ้ำ

โครงสร้างประโยคโปรแกรมอีกประการหนึ่งที่ทำให้เครื่องคอมพิวเตอร์เป็นอุปกรณ์ที่มีศักยภาพในการทำงานสูงก็คือ สามารถทำงานซ้ำๆ กันได้อย่างแม่นยำ เที่ยงตรง โปรแกรมสามารถทำเช่นนั้นได้ด้วยการที่ผู้เขียนโปรแกรมกำหนด วงรอบการทำงานซ้ำ (Loop) ของโปรแกรมไว้อย่างเหมาะสม และถูกกับสถานการณ์

วงรอบการทำงานซ้ำที่มีประสิทธิภาพจะประกอบไปด้วยปัจจัย 3 อย่างครบถ้วนดังนี้

การกำหนดการเริ่มต้นของวงรอบการทำงานซ้ำ (Loop Initialization) หมายถึง การกำหนดค่าตัวแปรเริ่มต้นที่จะนำไปทดสอบในวงรอบการทำงานซ้ำนั้น

การตรวจสอบเงื่อนไขในการวนวงรอบการทำงานซ้ำ (Loop Testing) หมายถึง การตรวจสอบค่าของตัวแปรเพื่อกำหนดว่าจะเกิดการวนวงรอบการทำงานซ้ำอีกครั้งหรือไม่

ตัวควบคุมวงรอบการทำงานซ้ำ (Loop Control) หมายถึง ประโยคโปรแกรมที่จะเปลี่ยนแปลงค่าตัวแปรที่จะนำไปตรวจสอบก่อนเริ่มการวนวงรอบการทำงานซ้ำ

ผู้เขียนโปรแกรมสามารถสร้างวงรอบการทำงานซ้ำได้หลายวิธี วิธีการแรกก็คือ การใช้คำสั่ง GoTo ซึ่งเป็นคำสั่งให้โปรแกรมกระโดดการทำงานไปยังตำแหน่งที่หมายและเริ่มทำงานตามโค้ดโปรแกรม จากจุดนั้น หลักไวยากรณ์การใช้คำสั่ง GoTo ใน Visual Basic เป็นดังนี้

```
GoTo label
        statement1

label :
        statement2
```

GoTo เป็นคีย์เวิร์ดใน Visual Basic เพื่อบอกถึงความต้องการใช้คำสั่ง GoTo

label หมายถึง ชื่อที่เป็นไปตามกฎการตั้งชื่อใน Visual Basic ของตำแหน่งหนึ่งๆ ในโพรซีเยอร์นั้น

label : หมายถึง ตำแหน่งใดๆ ในโพรซีเยอร์ ซึ่งมีชื่อเป็นตัวอ้างอิงตำแหน่ง

statements หมายถึง ประโยคโปรแกรมของ Visual Basic ทั่วไป

การทำงานของคำสั่ง GoTo นั้นเป็นการทำงานซ้ำรูปแบบหนึ่ง เนื่องจากทุกครั้งที่ Visual Basic เห็นคำสั่ง GoTo ก็จะกระโดดข้ามประโยคโปรแกรมใดๆ ที่อยู่หลังชื่อ label (ในตัวอย่างหมายถึง statement1) ไปยังตำแหน่งที่อยู่ของ label ดังกล่าวในโพรซีเยอร์นั้นๆ ทันที และเริ่มทำงานต่อไปด้วยการเรียกโค้ดที่อยู่ต่อจากชื่อ label นั้นขึ้นมาทำงาน (ในที่นี้คือ statement2)

2.33.2.1 การสร้างวงรอบการทำงานซ้ำด้วย Do...Loop

คำสั่ง Do...Loop เป็นประโยคคำสั่งวนรอบการทำงานซ้ำโดยเรียกโค้ดโปรแกรมที่อยู่ระหว่างคำว่า Do กับ Loop ขึ้นมาทำงานซ้ำแล้วซ้ำเล่า โดยมีคำว่า While และ Until เป็นตัวกำหนดเงื่อนไขในการวนวงรอบการทำงานซ้ำ มีหลักไวยากรณ์ในการใช้งานดังนี้

```
Do While condition
        statements
Loop
```

Do เป็นคีย์เวิร์ดใน Visual Basic บอกจุดเริ่มต้นของประโยค Do...Loop

While เป็นคีย์เวิร์ดใน Visual Basic กำหนดรูปแบบการตรวจสอบเงื่อนไข

condition เป็นเงื่อนไขที่จะได้รับการตรวจสอบก่อนการวนรอบการทำงานซ้ำ

statements เป็นประโยคโปรแกรมที่จะถูกเรียกขึ้นมาทำงานซ้ำๆ ตราบเท่าที่ภาวะเงื่อนไขของ condition เป็นจริง

Loop เป็นคีย์เวิร์ดใน Visual Basic บอกว่าเป็นจุดสิ้นสุดของประโยค Do...Loop

ลำดับขั้นตอนของวงรอบการทำงานซ้ำประโยค Do...Loop โดยใช้ While เป็นตัวกำหนดรูปแบบการตรวจสอบเงื่อนไข เริ่มต้นที่การตรวจสอบภาวะของเงื่อนไข (Condition) ของประโยค Do...Loop และประโยคโปรแกรมหลังคำว่า Do While จะถูกเรียกขึ้นมาทำงานก็ตราบเท่าที่ภาวะของเงื่อนไขเป็นจริง

นอกจากคำว่า While ในการกำหนดรูปแบบการตรวจสอบเงื่อนไขแล้ว ยังสามารถใช้คำว่า Until แทนคำว่า While เพื่อกำหนดรูปแบบการตรวจสอบเงื่อนไขใหม่ โดยมีหลักไวยากรณ์ดังนี้

```
Do Until condition
        statements
Loop
```

ประโยค Do...Loop ข้างต้นจะทำงานซ้ำไปเรื่อยๆ จนกว่าภาวะเงื่อนไขของ condition จะเป็นจริง จึงหยุดการทำงาน โดยที่การตรวจสอบเงื่อนไขจะเกิดขึ้นก่อนการวนรอบการทำงานซ้ำ อย่างเช่น หากต้องการให้พรินท์ค่าของตัวแปรตั้งแต่ 1 ถึง 10 บนหน้าจอ โดยใช้คำว่า Until สามารถทำได้ดังนี้

```
Dim A As Integer
Do Until A > 9
        A = A + 1
        Print A
Loop
```

ตราบเท่าที่ค่าตัวแปร A ยังไม่มากกว่า 9 วงรอบการทำงานซ้ำนี้ก็ยังทำงานอยู่ ดังนั้นในวงรอบการทำงานซ้ำรอบที่ 10 โปรแกรมจะหยุดทำงานเนื่องจากค่าตัวแปร A มีค่าเท่ากับ 10 และทำให้เงื่อนไข A > 9 เป็นจริงและวงรอบการทำงานซ้ำหยุดการทำงาน

2.33.2.2 การสร้างวงรอบในการทำงานซ้ำด้วยคำสั่ง For...Next

คำสั่ง For...Next มีหลักไวยากรณ์ดังนี้

```
For counter = start To end [Step increment]
        statements
Next [counter]
```

For เป็นคีย์เวิร์ดใน Visual Basic บอกจุดเริ่มต้นของประโยค For...Next

counter เป็นตัวแปรที่จะถูกนำไปทดสอบว่าจะวนรอบการทำงานซ้ำอีกหรือไม่

start เป็นค่าของตัวแปรที่กำหนดค่าเริ่มต้นของตัวแปร counter

To ในที่นี้เป็นคีย์เวิร์ดใน Visual Basic เพื่อจะกำหนดค่า end

end เป็นค่าของตัวแปรที่กำหนดค่าสุดท้ายของตัวแปร counter

Step เป็นคีย์เวิร์ดใน Visual Basic เพื่อกำหนดค่า increment

increment เป็นตัวกำหนดค่าที่จะเปลี่ยนแปลงคาสตัวแปร counter เมื่อวงรอบการทำงานซ้ำแต่ละรอบเสร็จสิ้น สามารถเป็นไปได้ทั้งจำนวนบวก และจำนวนติดลบ ถ้าหากผู้เขียนโปรแกรมไม่ได้กำหนดค่าด้วยคำสั่ง Step ไว้ ค่าเริ่มต้นจะเท่ากับ 1

statements คือกลุ่มประโยคโปรแกรมที่จะถูกเรียกขึ้นมาทำงานในแต่ละวงรอบการทำงานซ้ำ

Next เป็นคีย์เวิร์ดใน Visual Basic เพื่อระบุให้เริ่มทำงานวงรอบการทำงานซ้ำรอบใหม่

[counter] เป็นชื่อตัวแปรที่จะถูกเปลี่ยนแปลงค่าเท่ากับค่า increment ก่อนเริ่มวงรอบการทำงานซ้ำรอบใหม่ หากว่าผู้เขียนโปรแกรมไม่ได้ระบุชื่อตัวแปร counter ไว้ Visual Basic จะยึดเอาชื่อตัวแปรหลังคำว่า For ของประโยค For...Next นั้นๆ เป็นหลัก

ลำดับขั้นตอนในการทำงานของประโยค For...Next เป็นดังนี้

1. เริ่มต้นด้วยการตั้งค่าตัวแปร counter ให้เท่ากับค่า start
2. นำค่าตัวแปร counter ในปัจจุบันไปเปรียบเทียบกับค่า end
3. ทำงานตามประโยคโปรแกรมที่อยู่ระหว่างบรรทัดของคำว่า For กับคำว่า Next
4. ทำการเปลี่ยนค่าตัวแปร counter ไปเท่ากับค่าของ increment
5. เริ่มวงรอบการทำงานซ้ำตั้งแต่ข้อ 2 ถึง 4 อีกรอบหนึ่ง

วงรอบการทำงานซ้ำจะเป็นเช่นนี้เรื่อยไปจนกระทั่งค่าของตัวแปร counter จะมากกว่า ค่า end ในกรณีที่ค่า increment เป็นบวกจึงหยุดทำงานซ้ำ ส่วนในกรณีที่ค่า increment เป็นค่าติดลบ วงรอบการทำงานซ้ำจะหยุดก็ต่อเมื่อค่าตัวแปร counter มีค่าน้อยกว่าค่า end

2.33.2.3 โครงสร้างโปรแกรมแบบซ้อนกัน (Nested)

โครงสร้างการตัดสินใจแบบที่มีเงื่อนไขที่จะต้องตรวจสอบซับซ้อนหรือหลายรูปแบบ ประโยคโครงสร้างการตัดสินใจแบบ If...Then หรือ Select Case เพียงประโยคเดียวอาจจะไม่มีประสิทธิภาพที่เพียงพอ หรือในการสร้างวงรอบการทำงานซ้ำสำหรับงานที่ซับซ้อน โครงสร้างแบบ Do...Loop หรือ For...Next เพียงประโยคเดียวอาจจะไม่สามารถทำงานซ้ำได้อย่างที่ผู้เขียนโปรแกรมต้องการ ในกรณีเช่นนี้ Visual Basic ยินยอมให้ผู้เขียนโปรแกรมสามารถเขียนประโยคเหล่านี้ให้ซ้อนกันได้อย่างไม่จำกัด ยกตัวอย่างเช่น ต้องการกำหนดค่าของอาร์เรย์แบบ 2 มิติชุดหนึ่งให้เท่ากับ 20 ทุกตัว สามารถเขียนโค้ดโปรแกรมได้ดังนี้

```
Dim MyArray(10,10)
Dim I As Integer , J As Integer
```

```
For i = 1 To 10
        For J =1 To 10
                MyArray(I,J) = 20
        Next J
Next i
```

โปรแกรมจะเริ่มต้นด้วยการกำหนดค่าตัวแปร MyArray(1,1) ไปจนถึง MyArray(1,10) ให้เท่ากับ 20 ก่อน จากนั้นก็จะเริ่มกำหนดค่าตัวแปร MyArray(2,1) ไปจนถึง MyArray(2,10) และทำซ้ำไปเรื่อยๆ จนครบถึง 100 element

2.33.3 การใช้คำสั่ง Exit

ในโครงสร้างประโยคโปรแกรมบางโครงสร้าง ผู้เขียนโปรแกรมอาจจะต้องการให้โค้ดโปรแกรมส่วนนั้นหยุดทำงานทันที แม้ว่าจะยังทำงานได้ไม่ครบถ้วนตามที่ควรจะเป็นก็ตาม สามารถทำเช่นนั้นได้ด้วยการใช้คำสั่ง Exit ซึ่งสามารถใช้ได้กับโครงสร้างวงรอบการทำงานซ้ำทั้งประโยค Do...Loop และประโยค For...Next หรือใช้กับโพรซีเยอร์ซับรูทีนและโพรซีเยอร์ฟังก์ชัน โดยมีหลักไวยากรณ์ดังนี้

Exit Do	สำหรับโครงสร้างวงรอบการทำงานซ้ำแบบ Do...Loop
Exit For	สำหรับโครงสร้างวงรอบการทำงานซ้ำแบบ For...Next
Exit Sub	สำหรับโพรซีเยอร์ซับรูทีน
Exit Function	สำหรับโพรซีเยอร์ฟังก์ชัน

การใช้คำสั่ง Exit ในวงรอบการทำงานซ้ำมีประโยชน์ในกรณีที่ต้องการให้วงรอบการทำงานนั้นหยุดทำงานทันทีเมื่อมีเงื่อนไขใดเงื่อนไขหนึ่งเกิดขึ้น แม้ว่าจะยังวนรอบการทำงานซ้ำไม่ครบก็ตาม โดยสามารถวางคำสั่ง Exit Do หรือ Exit For ไว้ตรงตำแหน่งใดก็ได้ภายในบรรทัดระหว่างคำว่า Do...Loop หรือ For...Next ตามแต่ชนิดของประโยคที่เลือกใช้ เพื่อให้การใช้คำสั่ง Exit Do และ Exit For เป็นไปอย่างมีประสิทธิภาพ มีสิ่งที่ผู้เขียนโปรแกรมจะต้องจดจำเกี่ยวกับธรรมชาติของโครงสร้างวงรอบการทำงานซ้ำ ดังนี้

1. ปกติแล้วเมื่อวงรอบการทำงานซ้ำทำงานครบถ้วนด้วยตัวของมันเอง ค่าของตัวแปร counter จะมีค่าเท่ากับค่า end บวกกับค่า increment ในตอนสิ้นสุดการทำงานของประโยค
2. หากใช้คำสั่ง Exit ในวงรอบการทำงานซ้ำก่อนที่วงรอบการทำงานซ้ำจะทำงานครบถ้วนด้วยตัวมันเอง ค่าของตัวแปร counter จะมีค่าเท่ากับค่าปัจจุบันตอนที่คำสั่ง Exit เป็นผล

การใช้คำสั่ง Exit Sub ในโพรซีเยอร์ซับรูทีนและ Exit Function ในโพรซีเยอร์ฟังก์ชันก็ให้ผลเช่นเดียวกับการใช้คำสั่ง Exit Do หรือ Exit For โดยโปรแกรมจะหยุดการทำงานในโพรซีเยอร์นั้นและออกจากโพรซีเยอร์นั้นทันที

การใช้งานคำสั่ง Exit Sub หรือ Exit Function สามารถเกิดขึ้นได้หลายตำแหน่งในโพรซีเยอร์และปรากฏอยู่ในตำแหน่งใดก็ได้ภายในโพรซีเยอร์นั้นๆ โดยปกติแล้วมักจะใช้คำสั่ง Exit ในโพรซีเยอร์เพื่อหยุดการทำงานที่ไม่ควรจะเกิดขึ้นหรืออื่นๆ ตามแต่ผู้เขียนโปรแกรมจะกำหนด ยกตัวอย่างเช่น ไม่ต้องการให้การคลิกเมาส์ที่ปุ่มสั่งงานเกิดขึ้นในขณะที่ค่าตัวแปร blnGameStart ยังเท่ากับ False สามารถเขียนโค้ดโปรแกรมได้ดังนี้

```
if blnGameStart = False Then Exit Sub
Label1.Caption = "Hello World"
```

ตราบเท่าที่ตัวแปร blnGameStart ยังมีค่าเท่ากับ False การคลิกเมาส์บนปุ่มสั่งงานจะถูกเพิกเฉย (ประโยค Label1.Caption = "Hello World" จะไม่ถูกเรียกขึ้นมาทำงาน) เพราะว่าคำสั่ง Exit Sub จะทำให้โปรแกรมออกจากโพรซีเยอร์นี้ทันที

2.34 แนวความคิดของออบเจ็กต์โอเรียนเท็ดโปรแกรมมิ่ง (Object Oriented Programming) หรือ (OOP)

โอโอพี (OOP) หรือออบเจ็กต์โอเรียนเท็ดโปรแกรมมิ่ง (Object Oriented Programming) เป็นแนวความคิดในการเขียนโปรแกรมแบบหนึ่ง ที่ผู้รู้หลายๆ ท่านได้สรุปหรือได้ให้คำนิยามไว้ว่า เป็นการเขียนโปรแกรมเชิงวัตถุ บางท่านก็สรุปว่าเป็นการเขียนโปรแกรมแบบอ้างอิงวัตถุบ้าง ซึ่งก็เป็นการให้ความหมายที่ตรงมาก

ถ้าเราไม่มองในแง่มุมของการเขียนโปรแกรมเพียงอย่างเดียว ให้เรามองไปในภาพรวมมองไปในสิ่งรอบๆ ตัวเราเราสามารถบอกได้ว่า แนวคิดของโอโอพี ก็คือ "ธรรมชาติของวัตถุ" หมายความว่า จะมองสิ่งต่างๆ แต่ละสิ่งเป็นวัตถุชิ้นหนึ่ง มันจะมีสีแดงหรือเขียว ยาวหรือสั้น มันก็คือวัตถุชิ้นหนึ่งเหมือนกัน ซึ่งวัตถุแต่ละสิ่งนั้น ย่อมมีคุณสมบัติที่ต่างกัน แต่อาจจะมีบางอย่าง ที่เหมือนกันบ้าง และเราก็สามารถกำหนดประเภทหรือคลาสให้กับวัตถุเหล่านั้นได้ เช่นวัตถุสีแดงก็มารวมอยู่ในกลุ่มเดียวกัน หรือวัตถุที่มีขนาดยาวก็มารวมอยู่ในกลุ่มเดียวกันเป็นต้น

นอกจากนี้ยังคิดต่อไปอีกว่า "วัตถุแต่ละอย่างนั้น ต่างก็จะมีลักษณะและวิธีการใช้งานเป็นของตัวมันเอง" ประโยคนี้ มีความหมายว่า วัตถุแต่ละชนิดหรือแต่ละชิ้น ต่างก็มีรูปร่างลักษณะและการใช้งาน (การกระทำ) ที่แตกต่างกันออกไป เราจะเรียกคุณลักษณะของวัตถุว่า แอตทริบิวต์ (Attribute) และเรา

จะเรียกวิธีการใช้งานว่า เมธอด (Method) ยกตัวอย่างเช่น "ดินสอเป็นวัตถุที่มีลักษณะยาวเรียว, ภายในเป็นไส้ถ่านใช้สำหรับเขียน การใช้งานดินสอทำได้ โดยใช้มือจับและเขียนลงบนวัสดุรองรับ"

จากประโยคข้างต้นนี้ เราสามารถจับใจความได้ว่า คุณลักษณะของวัตถุ (Attribute) ก็คือ "ยาวเรียว, ภายในเป็นไส้ถ่าน" ส่วนการใช้งาน (Method) ก็คือ "ใช้มือจับและเขียนลงบนวัสดุรองรับ"

จากการยกตัวอย่างในข้างต้น คราวนี้เราสามารถสรุปได้แล้วว่า ถ้าเกิดวัตถุใดมีลักษณะยาวเรียว, มีไส้เป็นถ่าน เมื่อจะใช้งานมัน เราจะต้องใช้มือจับและเขียนลงบนวัสดุรองรับ เราก็สามารถบอกได้เลยว่าสิ่งนั้นก็คือ "ดินสอ" นั่นเอง

เราจะเห็นได้ว่า แนวความคิดของโอโอพี นั้นจะมีลักษณะที่คล้ายกับธรรมชาติของสิ่งของสิ่งหนึ่งซึ่งเราสามารถแบ่งแยกสิ่งต่างๆ ออกเป็นประเภทๆ ได้ ถ้าเราได้นำเอาแนวคิดของโอโอพีมาใช้ในการเขียนโปรแกรมและการจัดการข้อมูล เราจะพบว่าโปรแกรมหรือฟังก์ชั่น จะมีความเป็นอิสระแก่กันอย่างเห็นได้ชัด อธิบายง่ายๆ ก็คือ โปรแกรมหรือฟังก์ชั่นแต่ละตัวถึงแม้จะมาจากที่เดียวกันแต่มันสามารถทำงานในคนละหน้าที่ เก็บข้อมูลคนละค่าได้โดยจะไม่มายุ่งเกี่ยวกันแต่อย่างใด

2.35 ทฤษฎีทางปัญญาประดิษฐ์

2.35.1 การค้นหาแบบฮิวริสติก (Heuristic Search)

การค้นหาคำตอบ หรือการค้นหาข้อมูลในทางคอมพิวเตอร์มักจะกระทำบนโครงสร้างข้อมูลแบบต้นไม้ (Tree) และกราฟ (Graph) ทั้งนี้เพราะโครงสร้างข้อมูลในลักษณะนี้สามารถทำให้การค้นหาทำได้สะดวก และสามารถพลิกแพลงการค้นหาได้ง่าย ในความเป็นจริงแล้ว การค้นหาข้อมูลบางครั้งสามารถกระทำบนโครงสร้าง ข้อมูลชนิดอื่นก็ได้ เช่น อาเรย์ แสตก และคิว แต่การจัดข้อมูลในโครงสร้างเช่นนี้มีข้อจำกัดในการค้นหาข้อมูลมาก การค้นหาทำได้แบบเรียงลำดับ (Sequencial Search) เท่านั้น ซึ่งใช้ได้กับข้อมูลที่มีขนาดเล็ก ดังนั้นในการค้นหาข้อมูลที่มีขนาดใหญ่ ก่อนการค้นหา หรือระหว่างการค้นหา ข้อมูลที่จะถูกค้นจะต้องถูกจัดให้อยู่ในรูปแบบของต้นไม้ หรือกราฟเท่านั้น ซึ่งการค้นหาข้อมูลบนโครงสร้างต้นไม้และกราฟก็คือ การค้นหาแบบฮิวริสติก(Heuristic Search)นั่นเอง

ทางด้านปัญญาประดิษฐ์ การค้นหาคำตอบโดยอาศัยวิธีการแบบฮิวริสติก (Heuristic Search) มีความแตกต่างจากการค้นหาข้อมูลแบบธรรมดาและแบบฮิวริสติกนั้นอยู่ที่การค้นหาข้อมูลธรรมดา ผู้ที่ทำการค้นข้อมูลจะต้องตรวจสอบข้อมูลทีละตัวทุกตัวจนครบ แต่ฮิวริสติกจะไม่ลงไปดูข้อมูลทุกตัว วิธีการนี้จะเลือกได้คำตอบที่เหมาะสมให้กับการค้นหา ซึ่งมีข้อดีคือ สามารถทำการค้นหาคำตอบจากข้อมูลที่มีขนาดใหญ่มากๆ ได้ แต่มีข้อเสียคือคำตอบที่ได้เป็นเพียงคำตอบที่ดีเท่านั้น ไม่แน่ว่าจะดีที่สุด

แต่เนื่องจากว่าปัญหาในบางลักษณะนั้นใหญ่มาก และเป็นไปไม่ได้ที่จะทำการค้นหาด้วยวิธีธรรมดา กระบวนการของฮิวริสติกจึงเป็นสิ่งที่จำเป็น

ในเรื่องของฮิวริสติกนั้น นอกจากจะมีการค้นหาแบบฮิวริสติกแล้ว ยังมีอีกสิ่งหนึ่งที่สำคัญคือ ฮิวริสติกฟังก์ชัน (Heuristic Function) ซึ่งหมายถึงฟังก์ชันที่ทำหน้าที่ในการวัดขนาดของความเป็นไปได้ในการแก้ปัญหาซึ่งจะแสดงด้วยตัวเลข

วิธีการดังกล่าวจะกระทำได้โดยการพิจารณาถึงวิธีการ (Aspects) ต่างๆ ที่ใช้ในการแก้ปัญหา ณ สถานะหนึ่งว่าจะสามารถแก้ปัญหาได้ตามที่ต้องการหรือไม่ โดยกำหนดเป็นน้ำหนักที่ให้กับการแก้ปัญหาของแต่ละวิธี น้ำหนักเหล่านี้จะถูกแสดงด้วยตัวเลขที่กำกับไว้กับโหนดต่างๆ ในกระบวนการค้นหา และค่าเหล่านี้จะเป็นตัวที่ใช้ในการประมาณความเป็นไปได้ว่าเส้นทางที่ผ่านโหนดนั้นจะมีความเป็นไปได้ในการนำไปสู่หนทางการแก้ปัญหาได้มากน้อยขนาดไหน

จุดประสงค์ที่แท้จริงของฮิวริสติกฟังก์ชันก็คือ การกำกับทิศทางของกระบวนการค้นหา เพื่อให้อยู่ในทิศทางที่ได้ประโยชน์สูงสุด โดยการบอกว่าควรเลือกเดินเส้นทางไหนก่อน ในกรณีที่มีเส้นทางมากกว่าหนึ่งเส้นทางที่ต้องเลือก

กระบวนการค้นหาแบบฮิวริสติก โดยปกติแล้วจะต้องอาศัยฮิวริสติกฟังก์ชัน ทำการแก้ปัญหาหนึ่งๆ ซึ่งจะดีหรือไม่นั้น ก็ขึ้นอยู่กับฮิวริสติกฟังก์ชัน ดังนั้นการค้นหาแบบนี้จึงไม่มีอะไรเป็นหลักประกันว่าจะได้สิ่งที่ต้องการออกมา ด้วยเหตุนี้เอง จึงเรียกการค้นหาแบบฮิวริสติกนี้ว่า Weak Methods หรือจะกล่าวอีกนัยหนึ่งคือ Weak Methods เป็นกระบวนการควบคุมโดยทั่วไป (General-Purpose Control Strategies)

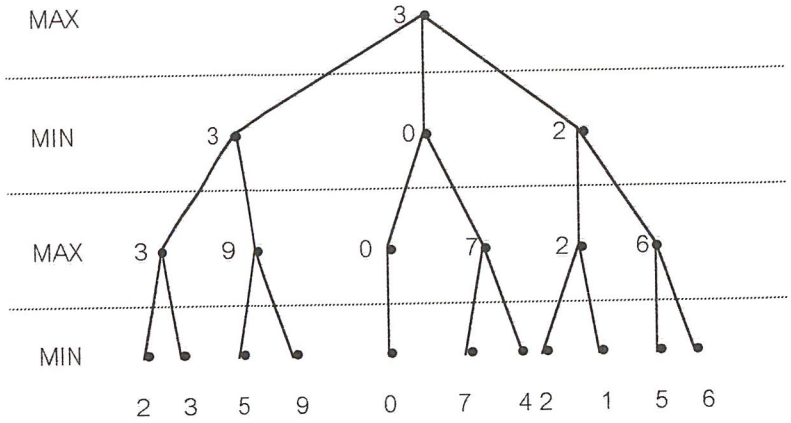
2.35.2 Minimax Procedure

ทฤษฎี Minimax ถูกสร้างขึ้นมาสำหรับเกมที่มีผู้เล่น 2 ฝ่าย เช่น หมากรุก, หมากฮอส หรือ เกมทิคแทคโท เป็นต้น ซึ่งเกมที่มีผู้เล่น 2 ฝ่ายจะสลับกันเดินในแต่ละครั้ง ซึ่งในทฤษฎี Minimax นี้จะกำหนดให้

Max แทนฝ่ายที่ต้องการชนะหรือเลือกทางที่จะได้เปรียบที่สุด (Maximize the adventages)

Min แทนฝ่ายตรงข้ามที่พยายามเลือกทางเดินที่ดีที่สุดของตนซึ่งจะทำให้แย่ที่สุดสำหรับฝ่าย Max (Minimize Max 's adventages)

ในทฤษฎี Minimax นี้ ฝ่าย Max จะทำการเลือกคะแนนที่มากที่สุดในขณะนั้นที่จะสามารถเลือกได้ (ซึ่งจะเป็นทางเดินที่ดีที่สุดสำหรับ Max) ส่วนฝ่าย Min จะทำการเลือกคะแนนที่น้อยที่สุดในขณะนั้นที่จะสามารถเลือกได้ (ซึ่งจะเป็นทางเดินที่ดีที่สุดสำหรับ Min)



รูปที่ 2.2 แสดง Minimax

จากทฤษฎี Minimax สมมติว่าสร้าง Search Tree ออกมาได้เป็นดังนี้ จาก Leaf Node (จากรูปที่ 2.2 Level ล่างสุด) เราจะใช้ Heuristic ฟังก์ชันทำการกำหนดค่าให้กับแต่ละ Leaf Node แล้วทำการเลือกจากด้านล่างไปยังด้านบน ยกตัวอย่างเช่น Node 2 กับ Node 3 จะเลือก Node 3 เนื่องจากเป็นตาที่ฝ่าย Max เดิน และที่ Level ถัดมาจะได้ Node3 กับ Node 9 จะเลือก Node 3 เนื่องจากเป็นตาที่ฝ่าย Min เดิน และที่ Level ถัดมาจะได้ Node 3 , Node 0 และ Node 2 จะเลือก Node 3 เนื่องจากเป็นตา Max เดิน เป็นต้น

ทฤษฎี Minimax นั้นจะมีประสิทธิภาพหรือไม่ขึ้นอยู่กับการกำหนดค่าความลึกในการค้นหา (Depth Search) ถ้ากำหนดให้มีค่ามาก จะมีความเที่ยงตรงสูง และถ้ากำหนดให้ถึง Node สุดท้ายจะเป็น Complete Search แต่ในความเป็นจริงแล้วจะไม่สามารถทำได้ เนื่องจากจะมี Problem Space มาก และอีกปัจจัยหนึ่งที่จะทำให้ทฤษฎี Minimax มีประสิทธิภาพก็คือ Heuristic ฟังก์ชัน ซึ่งจะต้องใช้ให้เหมาะสมกับโปรแกรมเกมที่สร้างขึ้น และจะต้องสะท้อนให้เห็นภาพของผู้เล่นทั้ง 2 ฝ่าย ซึ่งเรื่องของฮิวริสติกนั้นได้กล่าวมาแล้วในหัวข้อข้างต้น

เนื่องจากว่าทฤษฎี Minimax นั้นจะต้องทำการตรวจสอบ Leaf Node ทุกตัว ซึ่งถ้าเป็นโปรแกรมที่มี Search Tree ที่ใหญ่มากๆ จะทำให้ต้องใช้เวลาในการค้นหามาก จึงมีการคิดค้นทฤษฎี Alpha-Beta ขึ้นมาเพื่อปรับปรุงประสิทธิภาพของทฤษฎี Minimax ซึ่งจะกล่าวในหัวข้อถัดไป

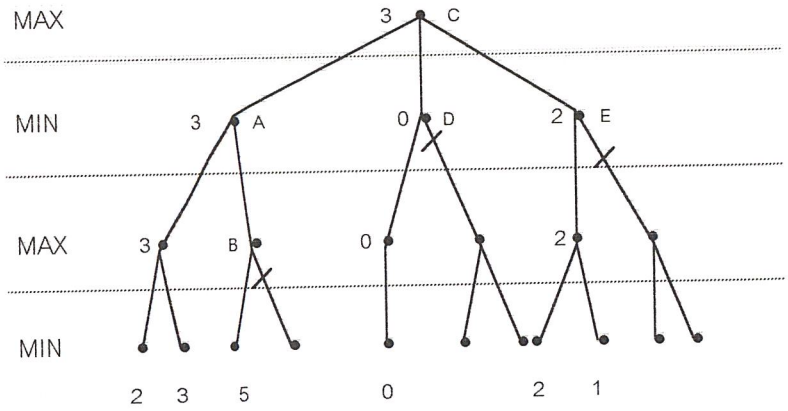
2.35.3 The Alpha-Beta Procedure

ทฤษฎี Alpha-Beta ถูกคิดขึ้นมาเพื่อปรับปรุงประสิทธิภาพของทฤษฎี Minimax ให้มีประสิทธิภาพมากขึ้น โดยที่ทฤษฎี Minimax นั้นจะทำการตรวจสอบหาทางเลือกที่ดีที่สุดจากทุก ๆ ความเป็นไปได้ที่

มีอยู่ในขณะนั้น ซึ่งจะทำให้ใช้เวลาในการตรวจสอบมาก จึงได้คิดทฤษฎี Alpha-Beta ขึ้นมาเพื่อเพิ่มประสิทธิภาพของ Minimax โดยการตัดทางที่ไม่จำเป็นที่จะต้องตรวจสอบออกไป ทำให้ใช้เวลาในการตรวจสอบน้อยลง โดยที่ทฤษฎี Alpha-Beta มีแนวคิดคือ ใช้วิธี Depth-first Search เพื่อทำการหาค่าสมมติของ Alpha และ Beta ซึ่ง Alpha เป็นค่าของ Max Nodes ซึ่งจะไม่ลดลง (Minimum of Maximum) และ Beta เป็นค่าของ Min Nodes ซึ่งจะไม่เพิ่มขึ้น (Maximum 0f Minimum)

กฎที่ใช้ในการตรวจสอบ Tree มี 2 ข้อคือ

1. ถ้า Min Node มีค่า Beta ที่น้อยกว่าหรือเท่ากับค่า Alpha ของ Ancestor Max Node ให้หยุดการตรวจสอบในสายนั้น
2. ถ้า Max Node มีค่า Alpha ที่มากกว่าหรือเท่ากับ Beta ของ Ancestor Min Node ให้หยุดการตวจสอบในสายนั้น



รูปที่ 2.3 แสดงทฤษฎี Alpha-Beta

สมมติว่าสร้าง Search Tree และ ให้ค่าฮิวริสติกฟังก์ชันได้ดังนี้ จากทฤษฎี Alpha-Beta จะมีขั้นตอนการทำงานคือ จากด้านซ้ายสุด และ Level ล่างสุด เลือกค่า Max ระหว่าง Node 2 กับ Node 3 ได้ Node 3 ใน Level ถัดไป ยังไม่ทราบค่า B จึงมี Node 3 เป็นตัวการันตีว่า Node ที่มีค่าน้อยกว่า 3 จะไม่ทำการเลือก และเมื่อมาค้นหาที่ B มันการันตีว่ามันจะไม่ต่ำกว่า 5 จึงทำการตัดสายนี้ทิ้ง (ไม่ทำการสำรวจต่อ) ต่อมาดูที่เส้นกลาง จะทำการดึง Node 0 ขึ้นไป ที่ Level Min มันการันตีว่ามันจะเลือกค่าที่มากกว่า 0 (ไม่เกิน 0) เมื่อเทียบกับ Level บนสุด ซึ่งเป็น Level Max มันจะการันตีว่าต้องไม่ต่ำกว่า 3 ดังนั้นจึงสามารถตัดเส้นทางนี้ออกไปได้ ต่อไปจะดูที่เส้นขวามือ

จะทำการเลือก Node 2 มาที่ Level Min ซึ่งจะการันตีว่า ค่าที่ได้จะต้องไม่เกิน 2 คือ น้อยกว่าหรือเท่ากับ 2 เมื่อเทียบกับ Node 3 ที่ Level บนสุด จึงสามารถตัดเส้นทาง E ออกได้

จากรูปที่ 2.3 จะสามารถสรุปได้ดังนี้

ที่จุด A ได้ $\beta = 3$ จะได้ว่า ค่าที่จุด A จะไม่มากกว่า 3

ที่จุด B จะถูกตัดออกไปเนื่องจาก $5 > 3$ (β Pruning)

ที่จุด C ได้ $\alpha = 3$ จะได้ว่า ค่าที่จุด C จะไม่น้อยกว่า 3

ที่จุด D จะถูกตัดออกไปเนื่องจาก $0 < 3$ (α Pruning)

ที่จุด E จะถูกตัดออกไปเนื่องจาก $2 < 3$ (α Pruning)

จะได้ค่าที่จุด C คือ 3

โดยทั่วไปแล้ว ทฤษฎี Alpha-Beta จะทำงานได้มีประสิทธิภาพกว่าทฤษฎี Minimax เพราะใช้ Resource เท่ากันแต่ทำการค้นหาได้ดีกว่า แต่กรณีที่ Node มีการเรียงกันวิธีการนี้จะไม่ช่วยอะไร เพราะจะต้องทำการค้นหาทั้งหมด

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

ขั้นตอนทั้งหมดในงานวิจัยจะเกิดขึ้นไม่ได้หากขาดการออกแบบ การวางแผนงานที่ดีและมีเป้าหมายชัดเจน การดำเนินงานทั้งหมดจะถูกกระทำภายใต้แผนที่วางเอาไว้อย่างรัดกุมเพื่อความสำเร็จตามเป้าหมายที่วางเอาไว้ในแต่ละช่วง แผนงานทั้งหมดมีดังนี้

1. ขั้นตอนการออกแบบ ซึ่งอาจแบ่งย่อยได้อีกหลายส่วนงาน
   - การออกแบบระบบเกม
   - การออกแบบอินเทอร์เฟสที่ใช้ติดต่อกับผู้เล่น
   - การออกแบบวิธีการตอบโต้ของฝ่ายคอมพิวเตอร์

ในขั้นตอนการออกแบบนี้จะใช้เวลานานพอสมควร เพื่อให้ได้เกมที่ตรงตามเป้าหมายที่ต้องการ โดยจะทำการเริ่มการออกแบบดังนี้

ตารางที่ 3.1 แสดงลำดับขั้นการดำเนินงาน

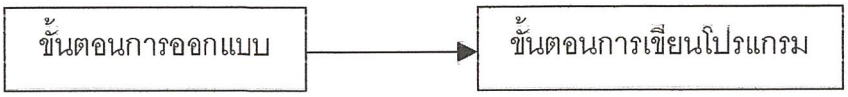
ขั้นตอนในการสร้างโปรแกรมเกม	งานที่ต้องทำ
การเริ่มต้นใช้งานโปรแกรม ข้อมูล Input	เริ่มเข้าโปรแกรมเข้าสู่เมนูหน้าแรก - รับข้อมูลจากผู้เล่น จากเมนูหน้าแรก เลือกตารางขนาดเท่าไรพร้อมเงื่อนไขการชนะ(เพื่อเป็นข้อมูลให้ปัญญาประดิษฐ์และการสร้างเกมพร้อมทั้งบอกกติกาการชนะให้กับโปรแกรม) และการเลือกการเริ่มเกมว่าฝ่ายใดเป็นผู้เริ่มลงเดินก่อน - รับข้อมูลจากผู้เล่นในการลงเดิน ณ ตำแหน่งที่คลิกว่าลงเดินได้หรือไม่ถ้าได้ถึงจะเขียนรูปตัวเดินและเปลี่ยนตา แต่ถ้าไม่แสดงเสียง - รับข้อมูลขึ้นเกมใหม่เมื่อกดปุ่ม “เริ่มเล่นใหม่” - รับข้อมูลการเปลี่ยนเกมที่ปุ่ม “เปลี่ยนเกม” - รับข้อมูลออกจากเกมที่ปุ่ม “ออกจากเกม” - รับข้อมูลการเปิดเสียงที่ปุ่ม “เปิดเสียง” - รับข้อมูลการปิดเสียงที่ปุ่ม “ปิดเสียง”

	- รับข้อมูลให้แสดงส่วนช่วยเหลือที่ปุ่ม “คู่มือการใช้งาน”
ปัญหา	- การสร้างปัญญาประดิษฐ์ให้สามารถเดินตอบโต้กับผู้เล่นได้ - การสร้างปัญญาประดิษฐ์ให้คอมพิวเตอร์มีทางชนะโดยให้เลือกทางเดินที่ดีที่สุด - ตรวจสอบการชนะได้ในเงื่อนไขที่ทำการเลือก - ตรวจการลงตราเดินของทั้ง 2 ฝ่าย - การสลับตาเดิน
ข้อมูล Output	- แสดงเมนูหลักได้ - แสดงเสียงเพลงเริ่มต้นในเมนูแรก - แสดงเสียงเมื่อไม่สามารถลงตาเดินได้ - แสดงเสียงและข้อความออกในเวลาที่ชนะว่าใครชนะและแสดงข้อความเสมอเมื่อเสมอ - สามารถทำงานในการกดปุ่ม "เริ่มเกมใหม่“, “เปลี่ยนเกม” และ "ออกจากโปรแกรม” - วาดภาพตารางตามที่ผู้เล่นได้เลือกออกมาตามขนาดที่เลือก - วาดภาพตัวเดินได้ 2 แบบที่เป็นฝ่ายผู้เล่นกันคอมพิวเตอร์ - วาดภาพตัวเดินลงในตำแหน่งที่คลิกเลือก - แสดงจอเริ่มต้นจอเมนูเลือกเล่น - แสดงจอเกมสำหรับเล่นเกม - แสดงจอขอบคุณท่านอาจารย์ - แสดงส่วนช่วยเหลือ
จบโปรแกรม	- รับข้อมูลเพื่อจบโปรแกรมจากปุ่ม “GoodLuck see you again!”

2. ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้ใช้เวลานานที่สุดในตลอดระยะเวลาทำวิจัย เพราะจำเป็นต้องศึกษารูปแบบวิธีการเขียนโปรแกรม เพื่อให้ได้โปรแกรมที่กระทัดรัด สามารถเข้าใจได้ง่าย และทำงานได้ถูกต้อง

เราสามารถสรุปขั้นตอนการวางแผนเป็นแผนภาพได้ดังนี้



รูปที่ 3.1 ขั้นตอนการวางแผน

3.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบมีรายละเอียดดังนี้

3.1.1 การออกแบบระบบเกม

การออกแบบระบบเกมเป็นส่วนที่สำคัญที่สุดในขั้นตอนการออกแบบทั้งหมด เพราะระบบเกมคือทุกสิ่งทุกอย่างตั้งแต่แนวคิดของเกม จนถึงการดำเนินไปของเกม

3.1.2 การออกแบบอินเทอร์เฟสที่ใช้ติดต่อกับผู้เล่น

ในส่วนการออกแบบอินเทอร์เฟสที่ใช้ติดต่อกับผู้เล่น เริ่มต้นจากการสร้าง Form ต่างๆ ด้วยภาษาวิชวลเบสิก แบ่งเป็น Form เมนูสำหรับเลือกเกม, Form ช่วยสำหรับแสดงส่วนของวิธีการเล่น, Form เกมสำหรับเล่นเกม และForm ขอบคุณเป็น Form ก่อนออกจากเกมไป ซึ่งจะประกอบด้วยปุ่มต่างๆ ในแต่ละ Form ดังนี้

Form เมนูเริ่มต้น โดยประกอบด้วยปุ่ม

- แฟ้มเกม ซึ่งมีปุ่มย่อยคือ ตารางเกมขนาดต่างๆ และ มีปุ่มย่อยในแต่ละปุ่มอีกคือ ปุ่มเงื่อนไขการชนะ ซึ่งปุ่มเหล่านี้ใช้สำหรับเลือกเกมที่จะเล่น
- ช่วย ซึ่งมีปุ่มย่อยคือ ปุ่มคู่มือใช้งาน สำหรับเรียกแสดงคู่มือใช้งาน
- ผู้เล่นเริ่มทำการเดินก่อน สำหรับเริ่มเล่นเกมโดยผู้เล่นทำการเริ่มเดินก่อน
- คอมพิวเตอร์เริ่มทำการเดินก่อน สำหรับเริ่มเล่นเกมโดยคอมพิวเตอร์ทำการเริ่มเดินก่อน

Form เกม โดยประกอบด้วยปุ่ม

- เริ่มเกมใหม่ สำหรับเริ่มเกมใหม่อีกครั้ง
- เปลี่ยนเกม สำหรับเปลี่ยนเกมโดยกลับไปเลือกเกมใหม่ที่ Form เมนูเริ่มต้น
- ออกจากเกม สำหรับออกจากเกมโดยไปที่ Form ขอบคุณ เพื่อจบโปรแกรม
- เปิดเสียง สำหรับเปิดเสียง
- ปิดเสียง สำหรับปิดเสียง

Form ช่วยเหลือ สำหรับไว้แสดงข้อมูลคู่มือการใช้

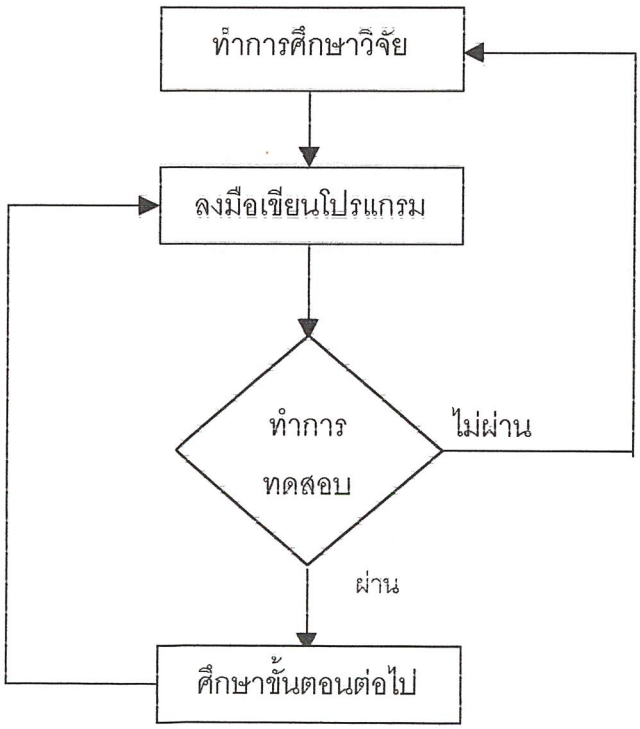
Form ขอบคุณ สำหรับไว้แสดงคำขอบคุณ ประกอบด้วยปุ่มจบโปรแกรม เพื่อจบโปรแกรม

3.1.3 การออกแบบการตอบโต้ของคอมพิวเตอร์

ส่วนของการออกแบบการตอบโต้ของคอมพิวเตอร์นั้นต้องใช้เวลาในการออกแบบนานที่สุดในขั้นตอนการออกแบบ เนื่องจากการตอบโต้ของคอมพิวเตอร์ต่อการกระทำของผู้เล่นเป็นเรื่องที่ละเอียดอ่อน และต้องใช้การระดมความคิด ในขั้นตอนการออกแบบจำเป็นจะต้องจำลองตัวเองให้เป็นผู้เล่นฝ่ายตรงข้าม เพื่อที่จะสามารถวิเคราะห์สถานการณ์และแก้ปัญหาได้ตรงจุด โดยใช้ทฤษฎีทางปัญญาประดิษฐ์เข้ามาทำการช่วยเหลือเพื่อให้คอมพิวเตอร์สามารถทำการตอบโต้ได้อย่างสมเหตุสมผล

3.2 ขั้นตอนการเขียนโปรแกรม

ส่วนใหญ่แล้วขั้นตอนต่างๆ จะมีการทำงานเป็นเส้นตรง หมายถึงเมื่อทำเสร็จงานหนึ่งก็จะทำงานต่อไปโดยไม่มีการวนมาที่งานนั้นใหม่ แต่ขั้นตอนการเขียนโปรแกรมนั้นมีการทำงานที่ต้องวนมาศึกษาเรื่อยๆ และทำการเขียนโปรแกรม และหากยังเขียนไม่สำเร็จ ก็ต้องทำการศึกษาเพิ่มเติมใหม่ ดังแผนภาพต่อไปนี้



รูปที่ 3.2 ขั้นตอนการเขียนโปรแกรม

โปรแกรมเกมจะถูกแบ่งเป็นส่วนย่อยๆ หลายส่วนดังนี้

- ส่วนของการแสดงผล
- ส่วนของการรับข้อมูลอินพุท
- ส่วนของการแสดงเสียง
- ส่วนของการตอบโต้ของคอมพิวเตอร์

เหตุผลหลักของผู้ทำโครงงานที่แบ่งการทำงานออกเป็นหลายๆ ส่วน เพื่อต้องการจัดแบ่งหน้าที่การทำงานให้เป็นประเภทเดียวกัน ทำให้ง่ายต่อการตรวจสอบ แก้ไข หรือเพิ่มเติม

หลักการ แนวความคิดและรูปแบบของข้อมูลในแต่ละส่วนการทำงาน

3.2.1 ส่วนการแสดงผล

ได้แก่ส่วนของ Form ต่างๆ ที่ได้กล่าวมาแล้ว รวมไปถึง Box แสดงข้อความการเสมอ และการชนะ

3.2.2 ส่วนของการรับข้อมูลอินพุท

โปรแกรมที่พัฒนาขึ้นมานี้จะใช้เมาส์ในการรับอินพุท โดยการคลิกที่ปุ่มคำสั่งต่างๆ ในแต่ละ Form เพื่อทำงานแสดงผลที่ต้องการตามที่ได้กล่าวไว้แล้วตั้งแต่ตอนต้น

3.2.3 ส่วนของการแสดงเสียง

โปรแกรมจะทำการแสดงเสียงแบ่งเป็น เสียงเพลงในเกม และเสียงเตือน ซึ่งทำให้เกมมีความหน้าสนใจและมีความตื่นเต้นมากขึ้น

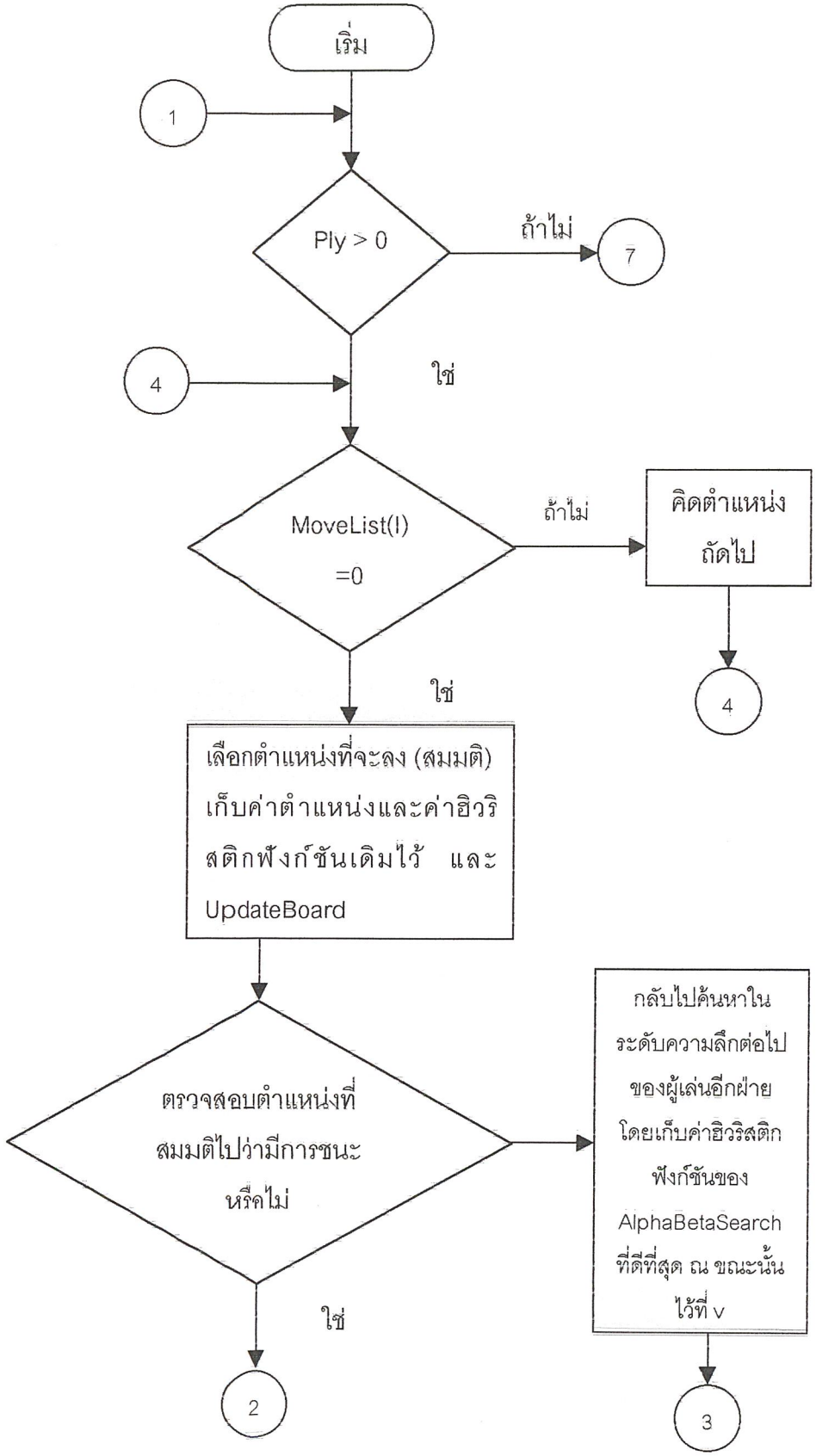
3.2.4 ส่วนของการตอบโต้ของคอมพิวเตอร์

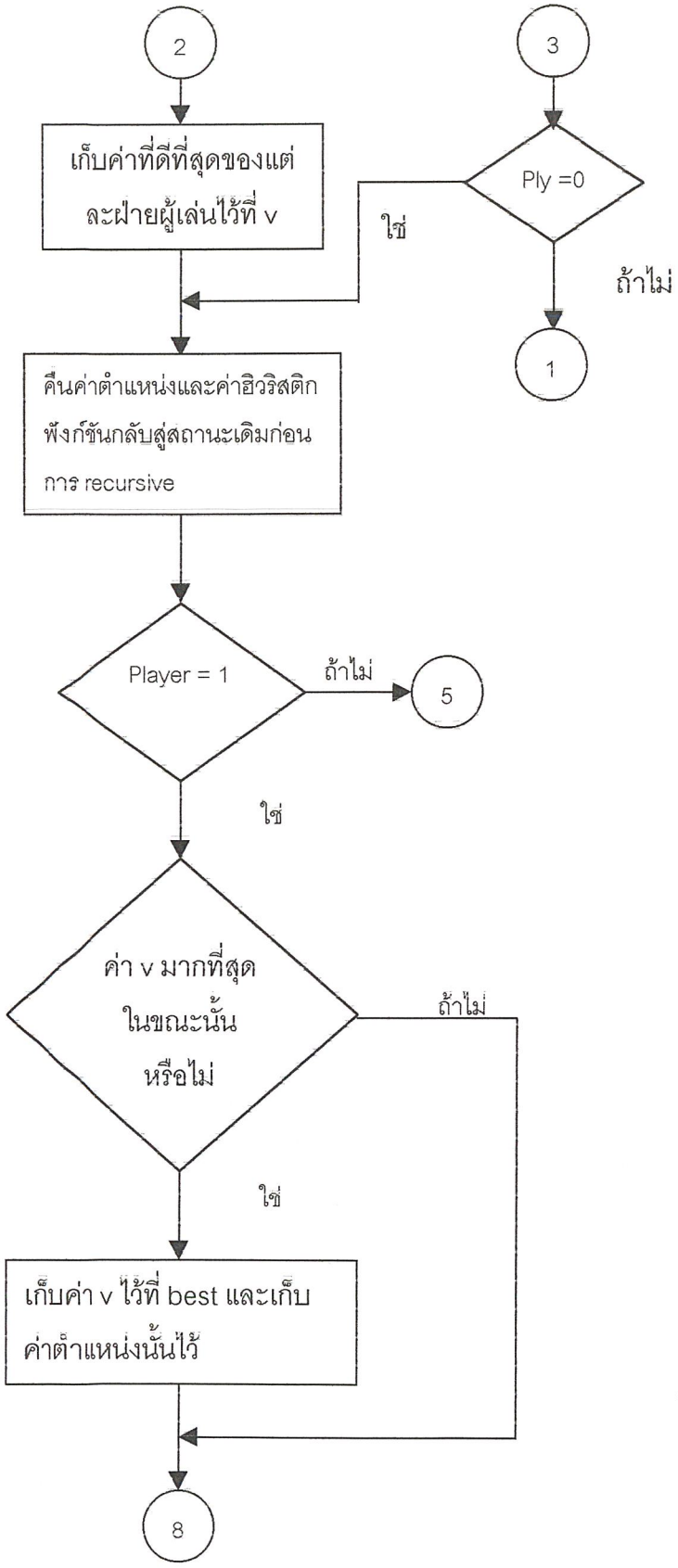
ในส่วนของการตอบโต้ของคอมพิวเตอร์นี้จะใช้ฟังก์ชัน Minimax โดยมี AlphaBeta Pruning ในการเขียนโปรแกรม ซึ่งมีอัลกอริทึมดังนี้

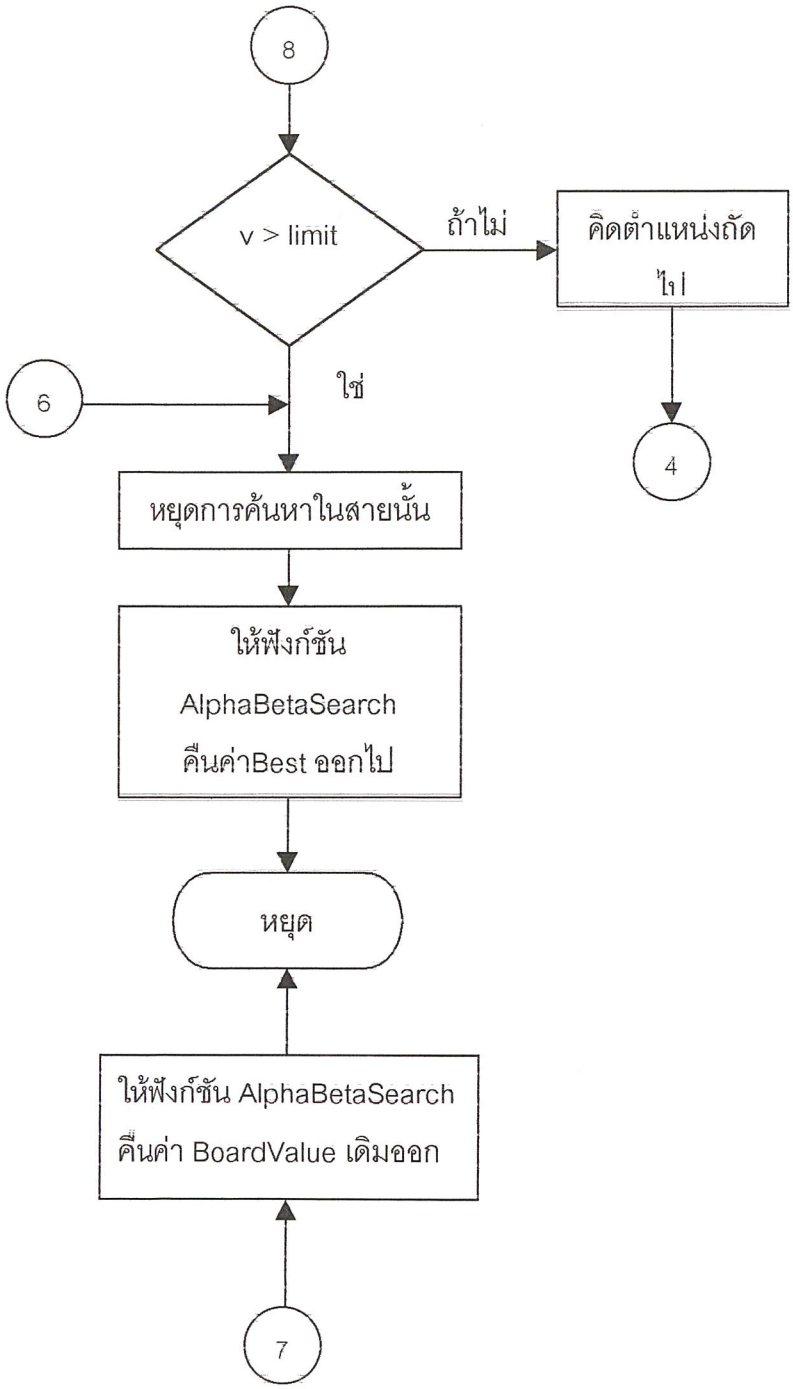
AlphaBetaSearch Algorithm

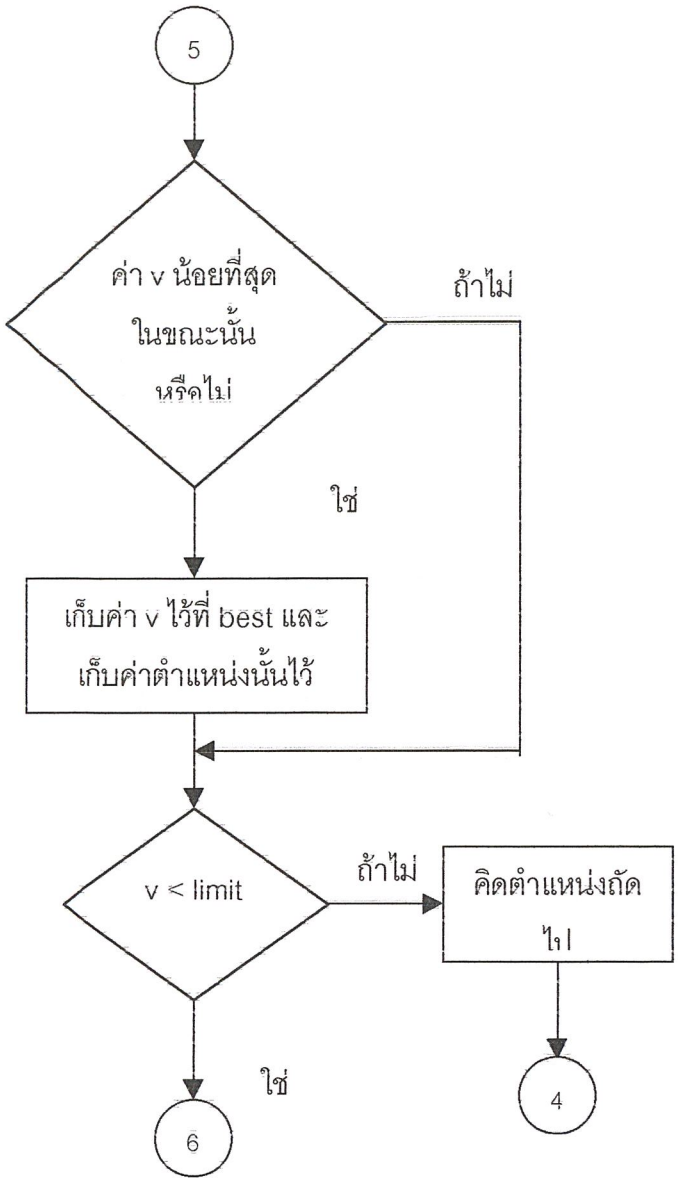
1. ทำการเลือกตำแหน่งที่สมมติว่าจะลงจาก MoveList(i)
2. การสมมติตัวผู้เล่นลงไปที่ตำแหน่งที่เลือก
3. ทำการหาค่าฮิวริสติกฟังก์ชัน ณ ตำแหน่งที่สมมติลงไป
4. ทำการเก็บค่าฮิวริสติกฟังก์ชันและตำแหน่งที่สมมติไป ถ้าค่าฮิวริสติกฟังก์ชันที่คำนวณได้ดีกว่าค่าฮิวริสติกเดิม
5. ถ้าระดับความลึกในการค้นหาไม่เท่ากับ 0 ให้ทำการสลับผู้เล่น และลดค่าระดับความลึกในการค้นหาลง 1 แล้วกลับไปทำข้อ 1 จนกว่าจะสมมติตัวที่จะลงครบทุกตำแหน่ง
6. กลับไปทำไปข้อ 1 จนกว่าจะสมมติตัวที่จะลงครบทุกตำแหน่ง
7. ฟังก์ชันจะทำการคืนค่าฮิวริสติกฟังก์ชัน และค่าตำแหน่งที่ดีที่สุดที่คำนวณได้
8. จบการทำงาน

มีแผนภาพการทำงานดังนี้




2
3
เก็บค่าที่ดีที่สุดของแต่ละฝ่ายผู้เล่นไว้ที่ v
Ply =0
ใช่
ถ้าไม่
1
คืนค่าตำแหน่งและค่าฮิวริสติกฟังก์ชันกลับสู่สถานะเดิมก่อนการ recursive
Player = 1
ถ้าไม่
5
ใช่
ค่า v มากที่สุดในขณะนั้นหรือไม่
ถ้าไม่
ใช่
เก็บค่า v ไว้ที่ best และเก็บค่าตำแหน่งนั้นไว้
8


8
v > limit
ถ้าไม่
คิดต้ำแหน่งถัด
ไป
6
ใช่
4
หยุดการค้นหาในสายนั้น
ให้ฟังก์ชัน
AlphaBetaSearch
คืนค่าBest ออกไป
หยุด
ให้ฟังก์ชัน AlphaBetaSearch
คืนค่า BoardValue เดิมออก
7



รูปที่ 3.3 ขั้นตอนการทำงานของ AlphaBetaSearch

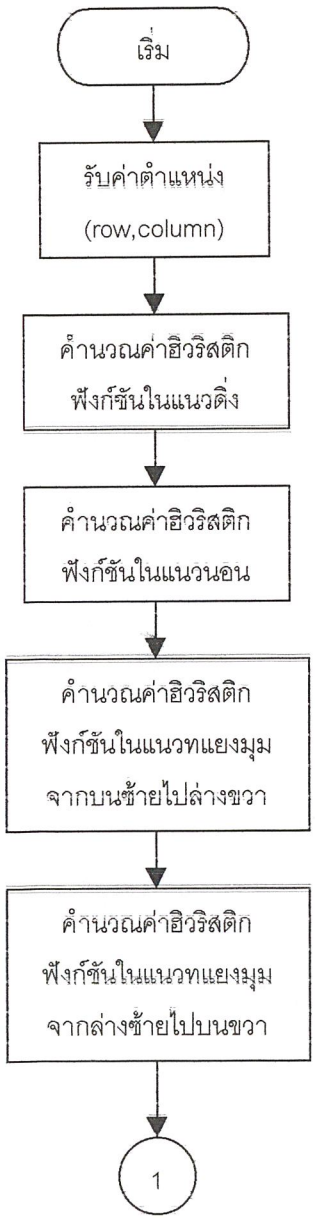
โดยฟังก์ชัน AlphaBetaSearch นี้จะทำงานร่วมกับฟังก์ชัน UpdateBoard ซึ่งทำหน้าที่คำนวณค่าฮิวริสติกฟังก์ชัน โดยฟังก์ชัน UpdateBoard มีอัลกอริทึมดังนี้

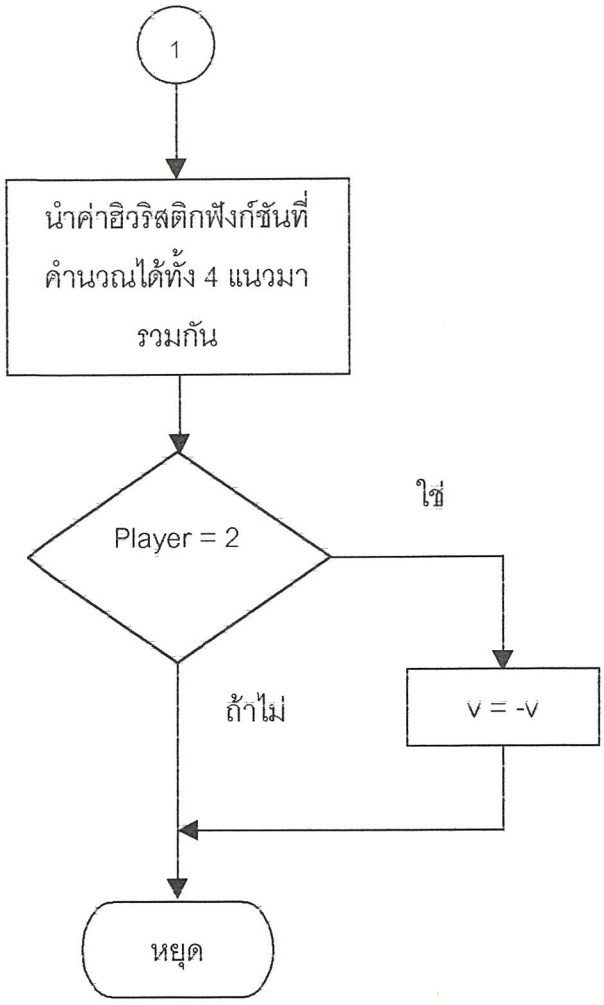
1. รับค่าตำแหน่ง(row, cloumn)
2. คำนวณค่าฮิวริสติกฟังก์ชันในแนวดิ่ง
3. คำนวณค่าฮิวริสติกฟังก์ชันในแนวนอน
4. คำนวณค่าฮิวริสติกฟังก์ชันในแนวทแยงมุมจากบนซ้ายไปล่างขวา
5. คำนวณค่าฮิวริสติกฟังก์ชันในแนวทแยงมุมจากล่างซ้ายไปบนขวา

6. นำค่าฮิวริสติกฟังก์ชันที่คำนวณได้จากข้อ 2 ถึงข้อ 5 มารวมกันแล้วเก็บไว้ที่ v
7. ถ้า Player = 2 ให้ v = -v.
8. จบการทำงาน

ฮิวริสติกฟังก์ชันที่ใช้ คือ f(n) = g(n) - h(n) โดย g(n) คือ ค่าฮิวริสติกฟังก์ชันของฝ่ายผู้เล่น และ h(n) คือ ค่าฮิวริสติกฟังก์ชันของฝ่ายคอมพิวเตอร์ หลักการของฮิวริสติกฟังก์ชันที่ใช้ คือ นับหนทางที่จะชนะในแนวที่คิด ณ ขณะนั้น เช่นถ้าในแนวที่คิดอยู่นั้นถ้ามีตัวของผู้เล่นอยู่ 1 ตัวจะนับได้ 1 และถ้ามีตัวของผู้เล่นอยู่ 2 ตัว จะนับได้ 11 และถ้ามีตัวผู้เล่นอยู่ 3 ตัวในแนวนั้นจะนับได้ 111 เป็นต้น

ฟังก์ชัน UpdateBoard มีแผนภาพการทำงานดังนี้

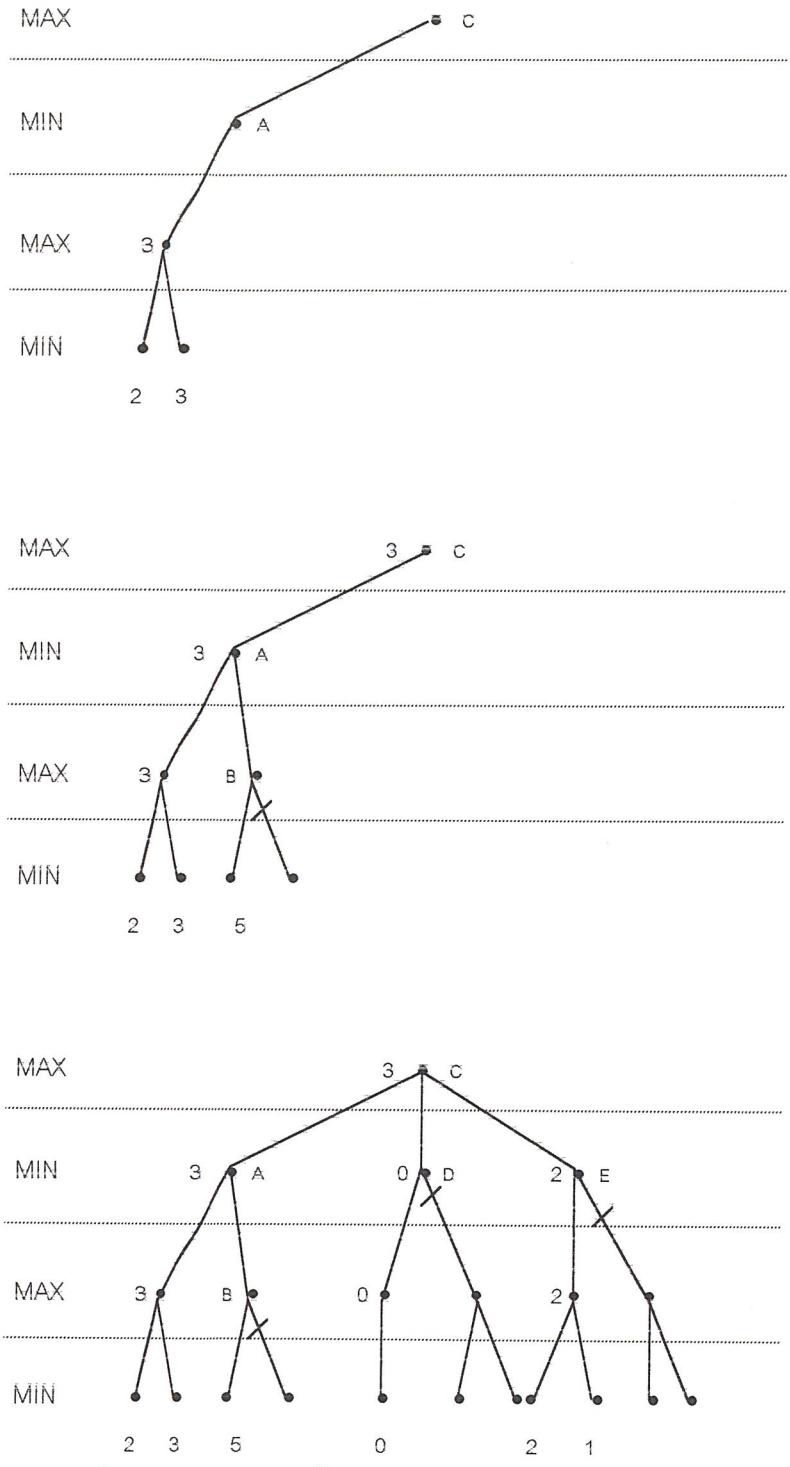




รูปที่ 3.4 ขั้นตอนการทำงานของ UpdateBoard

จากทฤษฎี Minimax โดยมี AlphaBeta Pruning นั้นเป็นทฤษฎีที่สร้างขึ้นมาสำหรับเกมที่มีผู้เล่น 2 ฝ่าย ซึ่งในโปรแกรมเกมนี้จะกำหนดให้ผู้เล่นเป็นฝ่าย Maximize และคอมพิวเตอร์เป็นฝ่าย Minimize พิจาณาถึงผู้เล่น 2 ฝ่าย ในเกมผู้เล่นทั้ง 2 จะผลัดกันเล่น โดยจะเรียกการให้คะแนนมากสุดและน้อยสุดของฟังก์ชัน โดยสมมติ ให้ Max เริ่มเดินก่อน จะมีการใช้ Node แบบต้นไม้แสดงตำแหน่งปัจจุบันและการเคลื่อนที่ เมื่อมีการเปลี่ยนตาเดินจะแสดงการเลือกเดินของอีกฝ่ายที่ต้องเดินจะเรียกว่า Node Max หรือ Min ขึ้นอยู่กับว่าใครเป็นฝ่ายเดิน การออกจากการแข่งขันเมื่อ Max ชนะ (มีคะแนนเป็นบวกอนันต์) หรือ Min ชนะ (มีคะแนนเป็นลบอนันต์) หรือตามคะแนนที่กำหนดไว้ The Ply of Node คือจำนวนการเดินที่ต้องการใน Node นั้น The Ply of the tree คือจำนวนความลึกมากที่สุดของ Node นั้นๆ ที่จะทำการหาคำตอบ

กลยุทธ์สำหรับการเล่น Minimax สำหรับ Max (Min) ทำการเลือกการเดินที่ทำให้ได้คะแนนสูงสุด (ต่ำสุด) คะแนนจะถูกคำนวณตั้งแต่เริ่ม จากล่างสุดของ Tree ขึ้นไป ซึ่งสามารถแสดงได้ดังนี้



รูปที่ 3.5 แสดงลำดับขั้นการคิดของปัญญาประดิษฐ์

จากรูปที่ 3.5 มีขั้นตอนการทำงาน ดังนี้

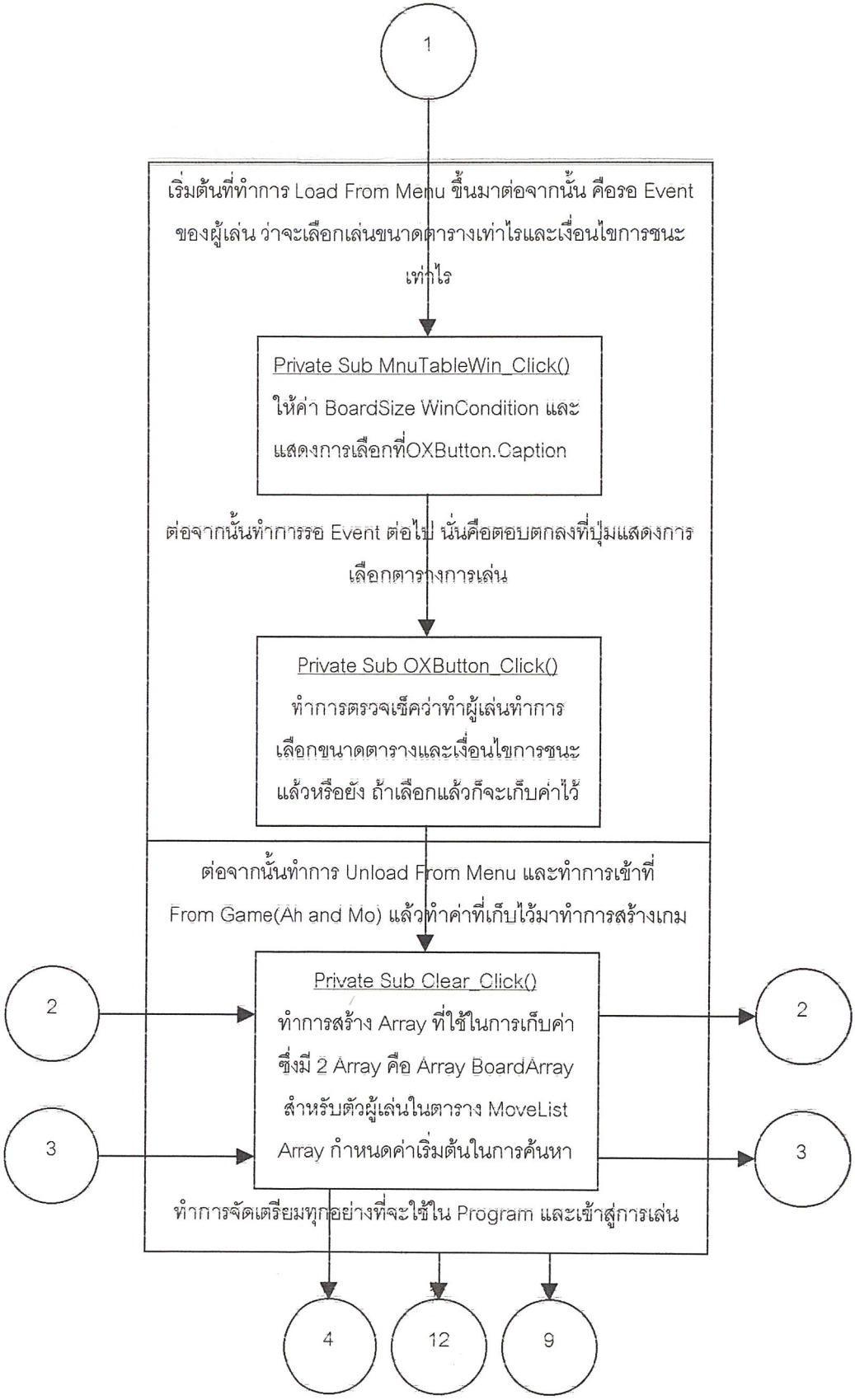
1. ทำการค้นหามาที่ Leaf Node แล้วทำการเลือกค่าที่ดีที่สุดในขั้นตอนนั้น (แล้วแต่ว่าเป็นขั้น Min หรือ Max)
2. เมื่อได้ค่าแล้วให้ทำการ Promote ค่านั้นมาที่จุด A

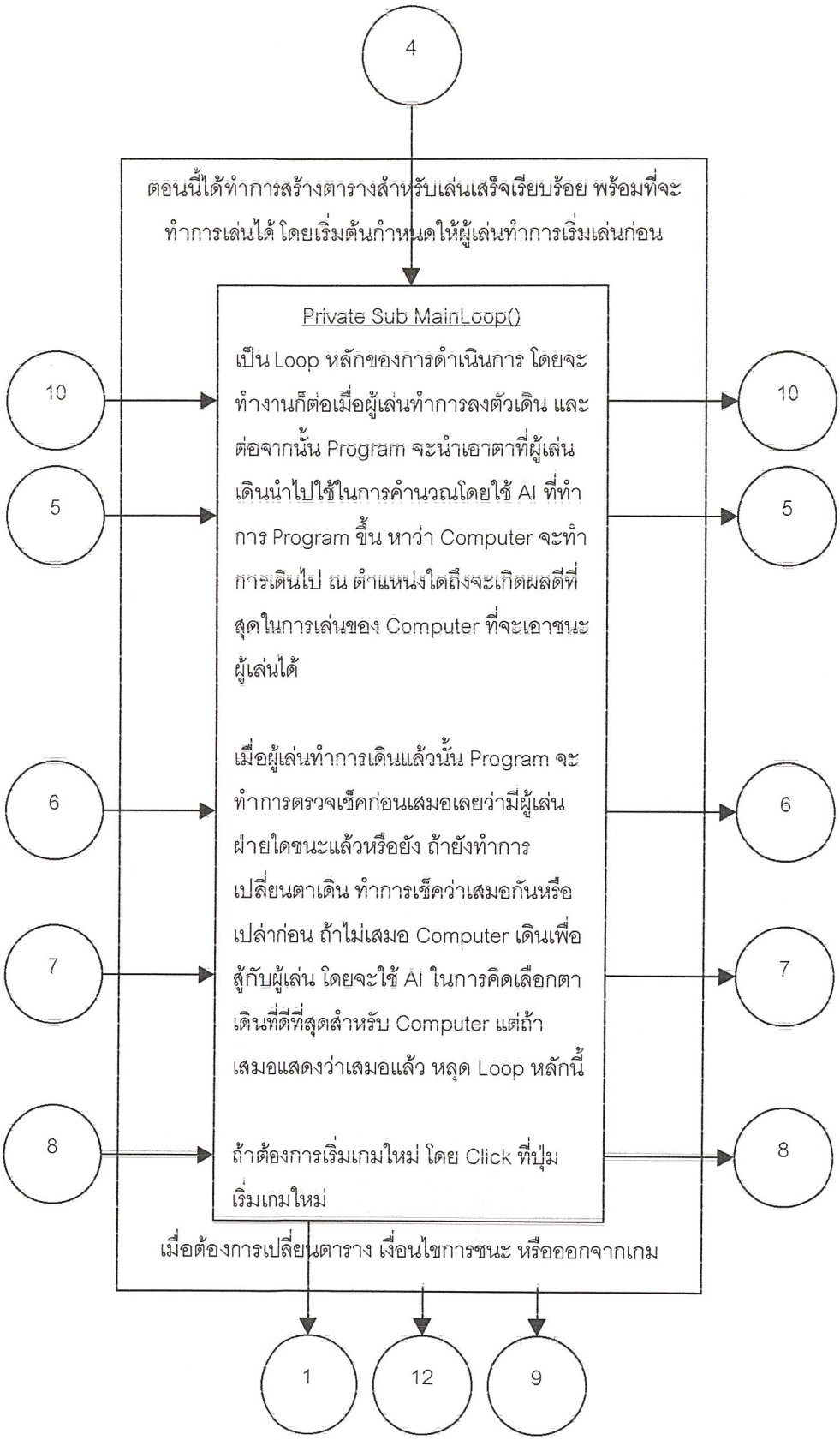
3. ทำการค้นหาไปที่ Leaf Node ในสายต่อไป แล้วทำการดึงค่ามาที่จุด B
4. ทำการเปรียบเทียบค่าระหว่างจุด A และจุด B ถ้าค่าที่จุด A มีค่าน้อยกว่าที่จุด B ให้ทำการหยุดค้นหาในสายนั้น
5. เมื่อได้ค่าที่จุด A แล้วให้ทำการ Promote ไปที่จุด C
6. ทำการค้นหาไปที่ Leaf Node ของสายต่อไป แล้วทำการดึงค่ามาที่จุด D
7. ทำการเปรียบเทียบค่าระหว่างจุด C และจุด D ถ้าค่าที่จุด C มีค่ามากกว่าที่จุด D ให้ทำการหยุดค้นหาในสายนั้น
8. ทำการค้นหามาที่ Leaf Node ของสายต่อไปแล้วทำการเลือกค่าที่ดีที่สุดในขั้นตอนนั้น (แล้วแต่ว่าเป็นขั้น Min หรือ Max)
9. เมื่อได้ค่าแล้วให้ทำการ Promote ค่านั้นมาที่จุด E
10. ทำการเปรียบเทียบค่าระหว่างจุด C และจุด E ถ้าค่าที่จุด C มีค่ามากกว่าที่จุด E ให้ทำการหยุดค้นหาในสายนั้น
11. จะได้ค่าที่จุด C คือ 3 และจะได้เส้นทางที่ควรเลือกคือควรจะเลือกที่จุด A

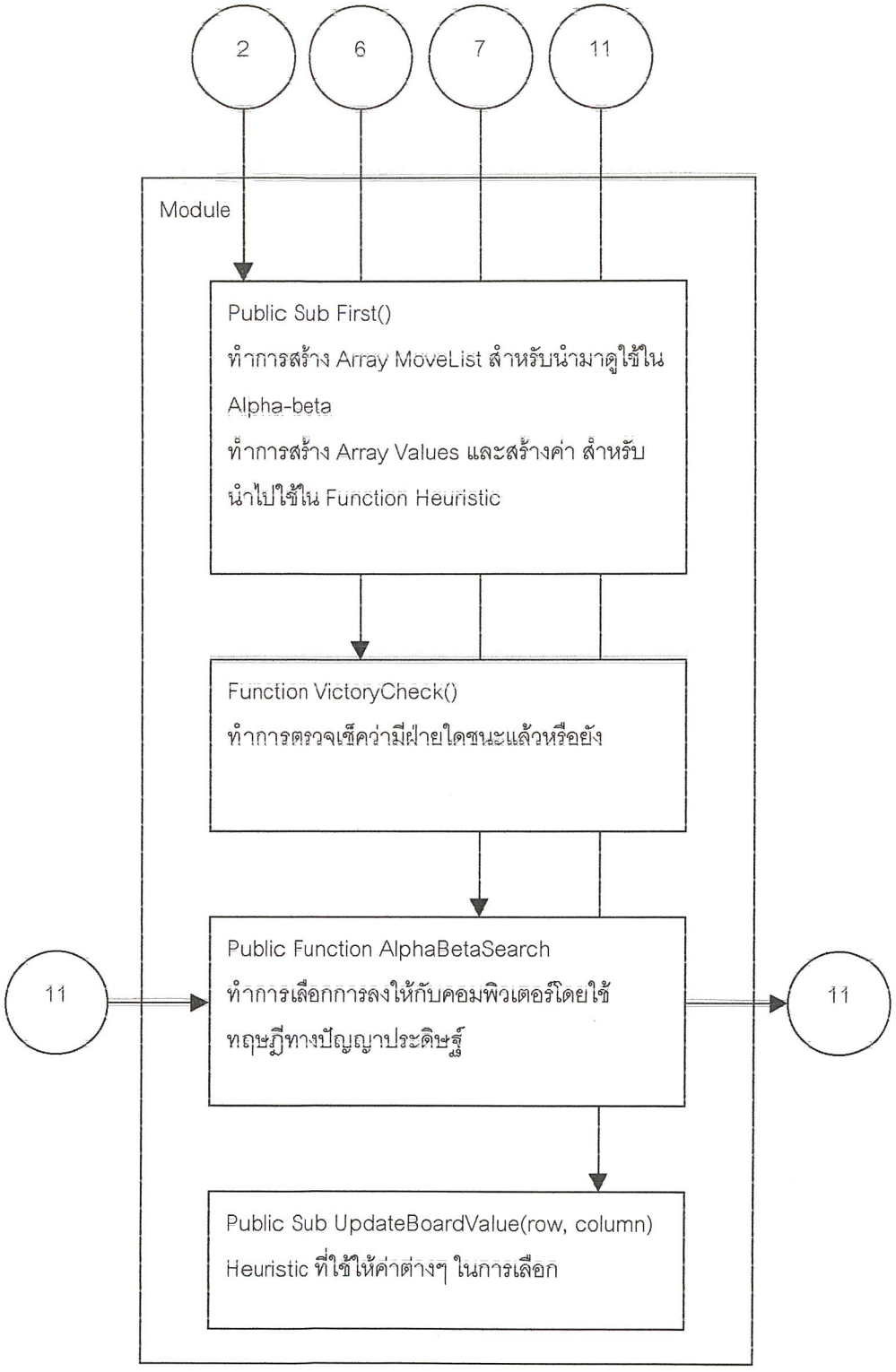
3.2.5 Flow Chart การทำงานทั้งหมดของโปรแกรม

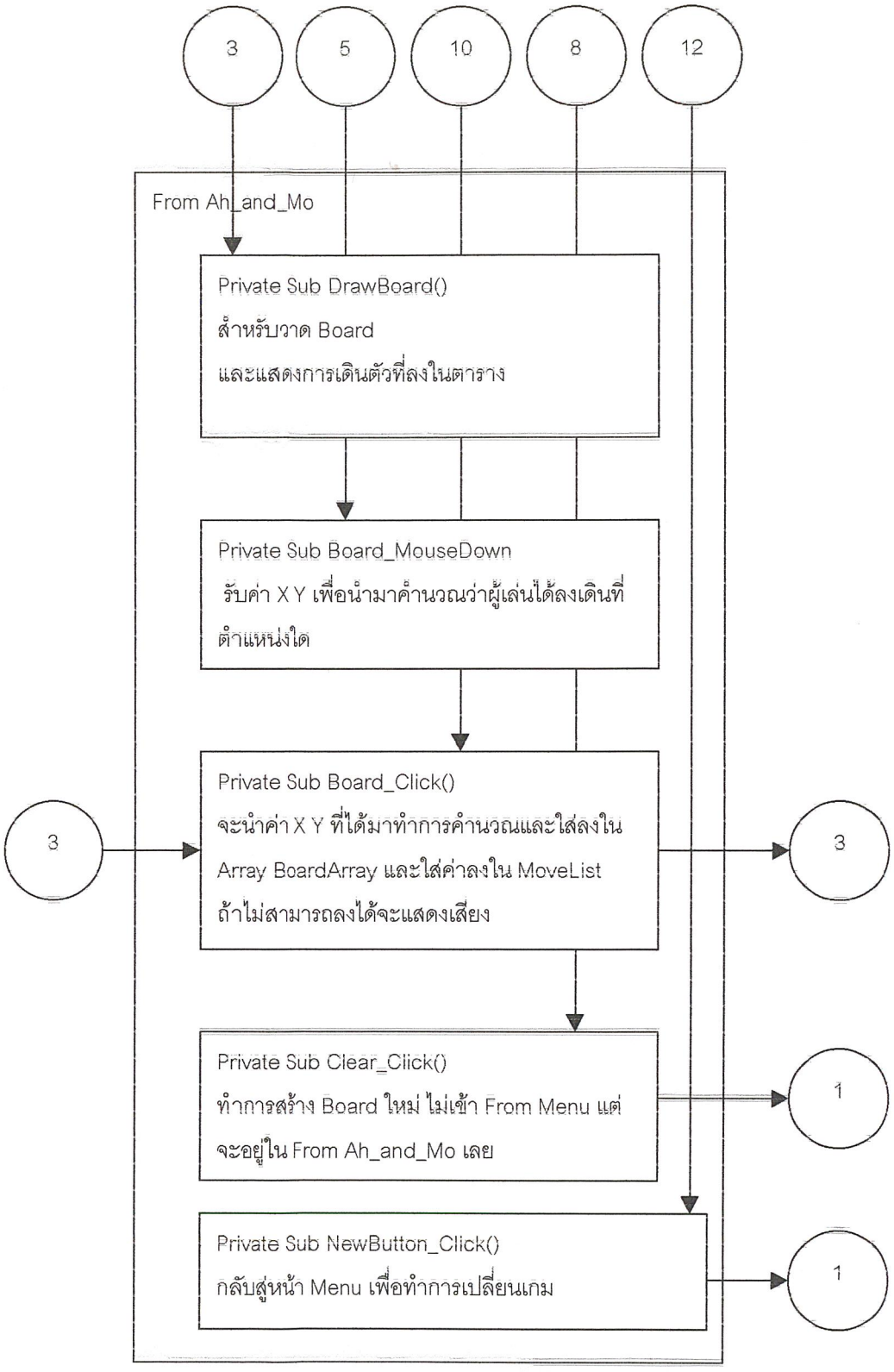
ซึ่งจะอธิบายการทำงานของโค้ดโปรแกรมที่เขียน และบอกถึงเหตุการณ์การต่างๆ ที่ต่อเนื่องและการดำเนินงานของโปรแกรมในขณะที่ผู้เล่นทำการเล่นเกม

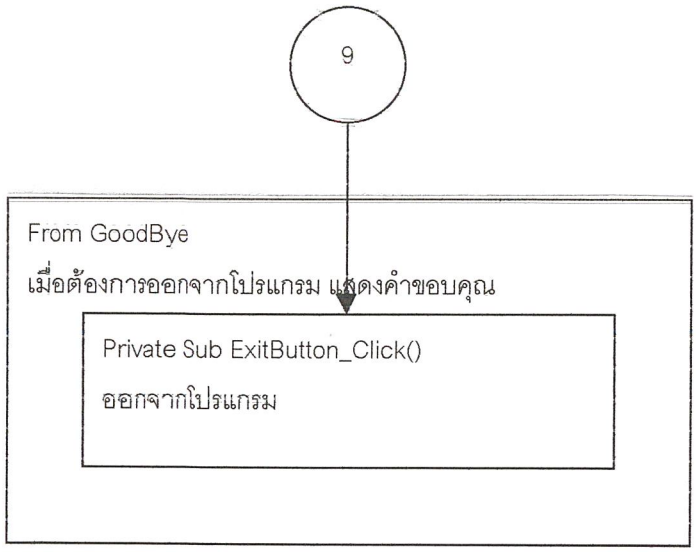
Flow Chart อธิบายการทำงานทั้งหมดของ Program




4
ตอนนี้ได้ทำการสร้างตารางสำหรับเล่นเสร็จเรียบร้อย พร้อมที่จะทำการเล่นได้ โดยเริ่มต้นกำหนดให้ผู้เล่นทำการเริ่มเล่นก่อน
Private Sub MainLoop()
เป็น Loop หลักของการดำเนินการ โดยจะทำงานก็ต่อเมื่อผู้เล่นทำการลงตัวเดิน และต่อจากนั้น Program จะนำเอาตาที่ผู้เล่นเดินนำไปใช้ในการคำนวณโดยใช้ AI ที่ทำการ Program ขึ้น หาว่า Computer จะทำการเดินไป ณ ตำแหน่งใดถึงจะเกิดผลดีที่สุดในการเล่นของ Computer ที่จะเอาชนะผู้เล่นได้
เมื่อผู้เล่นทำการเดินแล้วนั้น Program จะทำการตรวจเช็คก่อนเสมอเลยว่ามีผู้เล่นฝ่ายใดชนะแล้วหรือยัง ถ้ายังทำการเปลี่ยนตาเดิน ทำการเช็คว่าเสมอกันหรือเปล่าก่อน ถ้าไม่เสมอ Computer เดินเพื่อสู้กับผู้เล่น โดยจะใช้ AI ในการคิดเลือกตาเดินที่ดีที่สุดสำหรับ Computer แต่ถ้าเสมอแสดงว่าเสมอแล้ว หลุด Loop หลักนี้
ถ้าต้องการเริ่มเกมใหม่ โดย Click ที่ปุ่มเริ่มเกมใหม่
เมื่อต้องการเปลี่ยนตาราง เงื่อนไขการชนะ หรือออกจากเกม
10
5
6
7
8
10
5
6
7
8
1
12
9


2
6
7
11
Module
Public Sub First()
ทำการสร้าง Array MoveList สำหรับนำมาดูใช้ใน Alpha-beta
ทำการสร้าง Array Values และสร้างค่า สำหรับนำไปใช้ใน Function Heuristic
Function VictoryCheck()
ทำการตรวจเช็คว่ามีฝ่ายใดชนะแล้วหรือยัง
11
Public Function AlphaBetaSearch
ทำการเลือกการลงให้กับคอมพิวเตอร์โดยใช้ทฤษฎีทางปัญญาประดิษฐ์
11
Public Sub UpdateBoardValue(row, column)
Heuristic ที่ใช้ให้ค่าต่างๆ ในการเลือก


3
5
10
8
12
From Ah_and_Mo
Private Sub DrawBoard()
สำหรับวาด Board
และแสดงการเดินตัวที่ลงในตาราง
Private Sub Board_MouseDown
รับค่า X Y เพื่อนำมาคำนวณว่าผู้เล่นได้ลงเดินที่
ตำแหน่งใด
3
Private Sub Board_Click()
จะนำค่า X Y ที่ได้มาทำการคำนวณและใส่ลงใน
Array BoardArray และใส่ค่าลงใน MoveList
ถ้าไม่สามารถลงได้จะแสดงเสียง
3
Private Sub Clear_Click()
ทำการสร้าง Board ใหม่ ไม่เข้า From Menu แต่
จะอยู่ใน From Ah_and_Mo เลย
1
Private Sub NewButton_Click()
กลับสู่หน้า Menu เพื่อทำการเปลี่ยนเกม
1



รูปที่ 3.6 Flow Chart อธิบายการทำงานทั้งหมดของ Program

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

การทดลองและการวิเคราะห์ปัญหาที่กล่าวถึงในบทนี้ จะเป็นขั้นตอนการทดสอบและผลการทดสอบที่ได้ในแต่ละขั้น โดยผลการทดสอบนี้ จะถูกนำไปวิเคราะห์ถึงปัญหาและแนวทางในการพัฒนาต่อไปในอนาคต โดยทางผู้จัดทำหวังว่าจะเป็นประโยชน์แก่ผู้ที่จะนำไปศึกษาเพื่อพัฒนาต่อและเพื่อให้เห็นถึงปัญหา และข้อดีข้อเสียของปัญหานี้โดยที่ผู้ศึกษาไม่จำเป็นต้องทำการทดสอบเพื่อหาปัญหา และนำไปพัฒนาต่ออีกครั้งหนึ่ง

คุณสมบัติของระบบที่นำมาทดสอบ

- ระบบปฏิบัติการ Microsoft Windows 98
- เครื่องคอมพิวเตอร์ที่มีซีพียูความเร็ว 300 MHz
- หน่วยความจำหลักขนาด 32 MByte

ขั้นตอนการดำเนินการทดสอบ

- ทำการติดตั้งตัวโปรแกรมและซอฟท์แวร์ที่จำเป็นต้องใช้
- ทำการรันและประมวลผลภายใต้ระบบที่กำหนด
- ตรวจสอบการใช้คำสั่งและการทำงาน
- ตรวจสอบการตอบโต้ของฝ่ายคอมพิวเตอร์ขณะทำการเล่น
- ตรวจสอบการหา Error

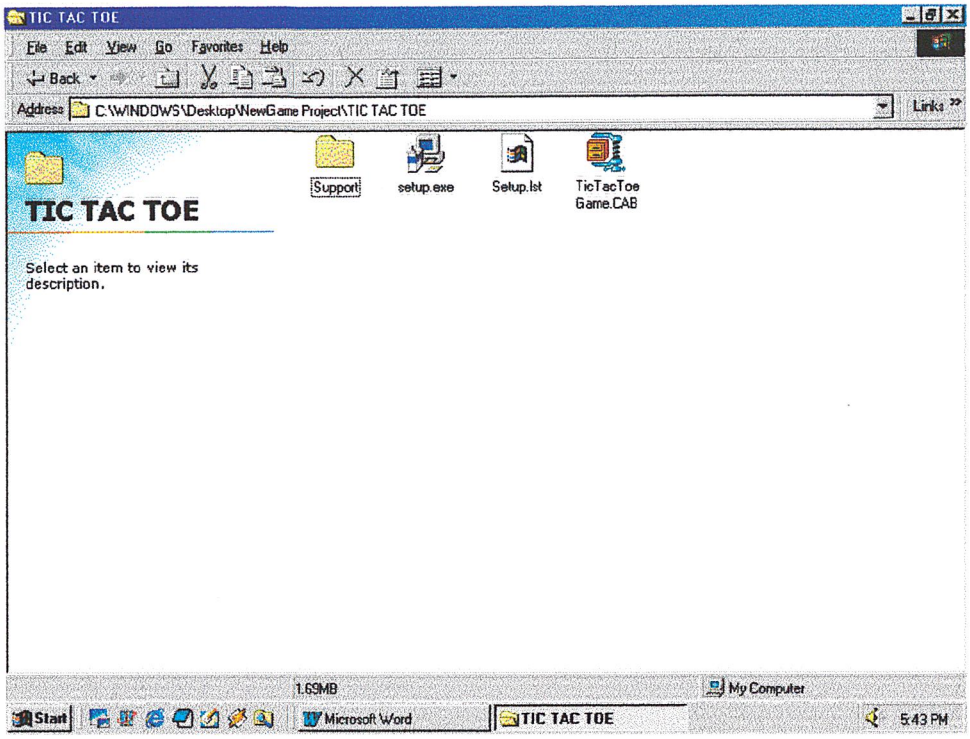
การประเมินผล

- หลังจากติดตั้งตัวโปรแกรมแล้วสามารถใช้โปรแกรมได้
- การตอบโต้ของคอมพิวเตอร์จะต้องเป็นไปตามที่กำหนดไว้
- การใช้คำสั่งและการทำงานต้องเป็นไปตามที่กำหนดไว้
- จำนวน Error ที่เกิดขึ้นจะต้องไม่มี หรือน้อยที่สุด

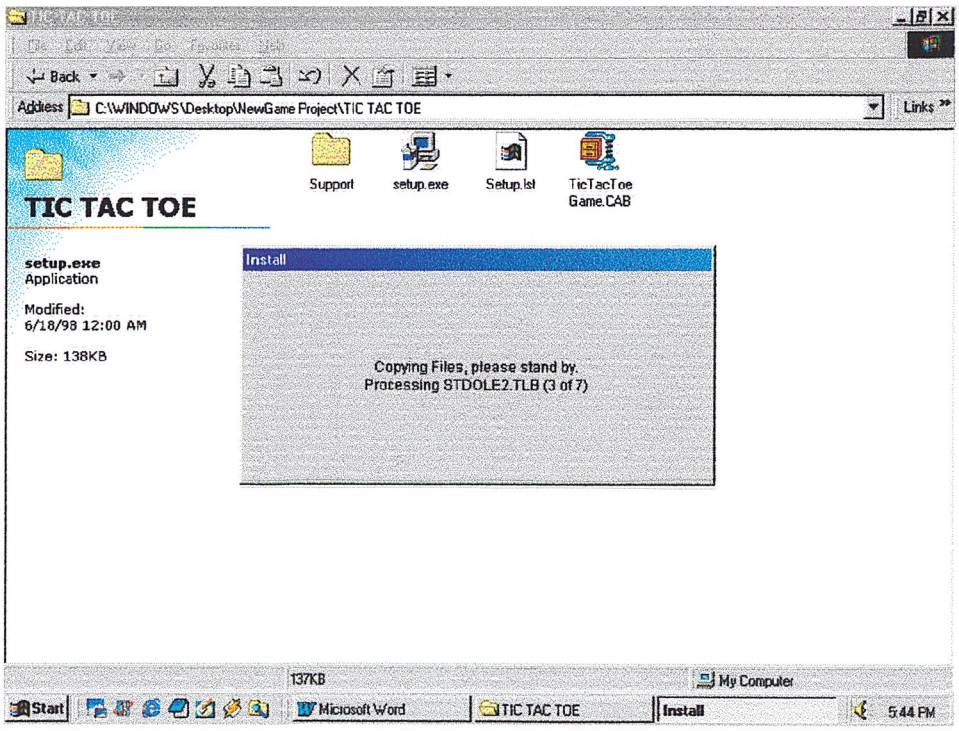
ต่อไปจะกล่าวถึงรายละเอียดของขั้นตอนการทดสอบและผลการทดสอบที่ได้

4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและซอฟท์แวร์ที่จำเป็น

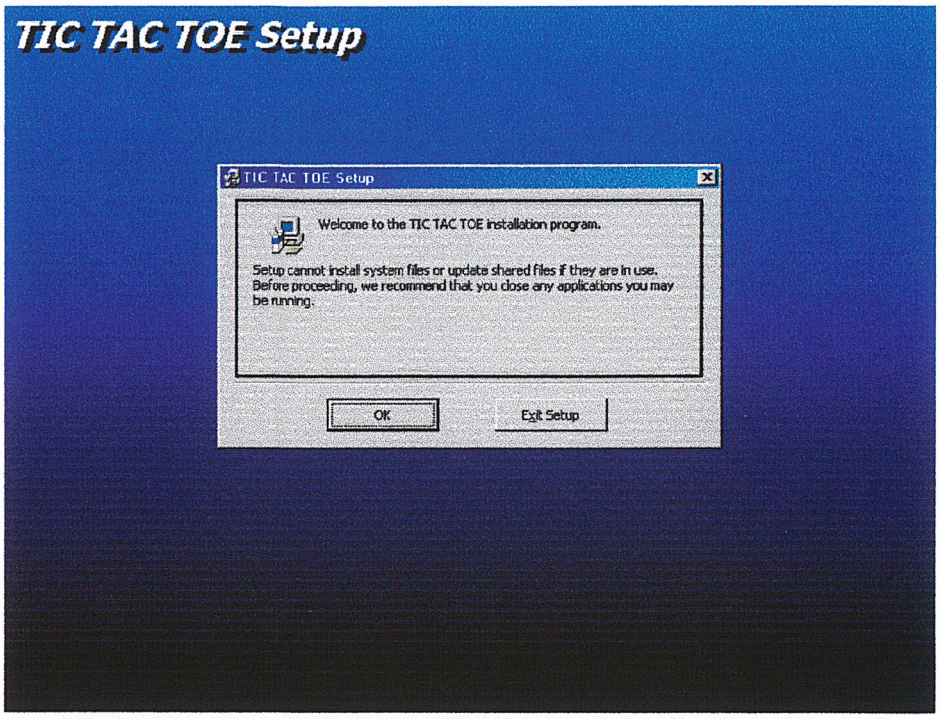
- ทำการติดตั้งโปรแกรมและซอฟท์แวร์
- ผู้ใช้ทำการรันโปรแกรมได้โดยคลิกที่ไฟล์ “setup.exe” ใน Folder “TIC TAC TOE” โดยต่อจากนี้ไปจะอธิบายด้วยรูปภาพ



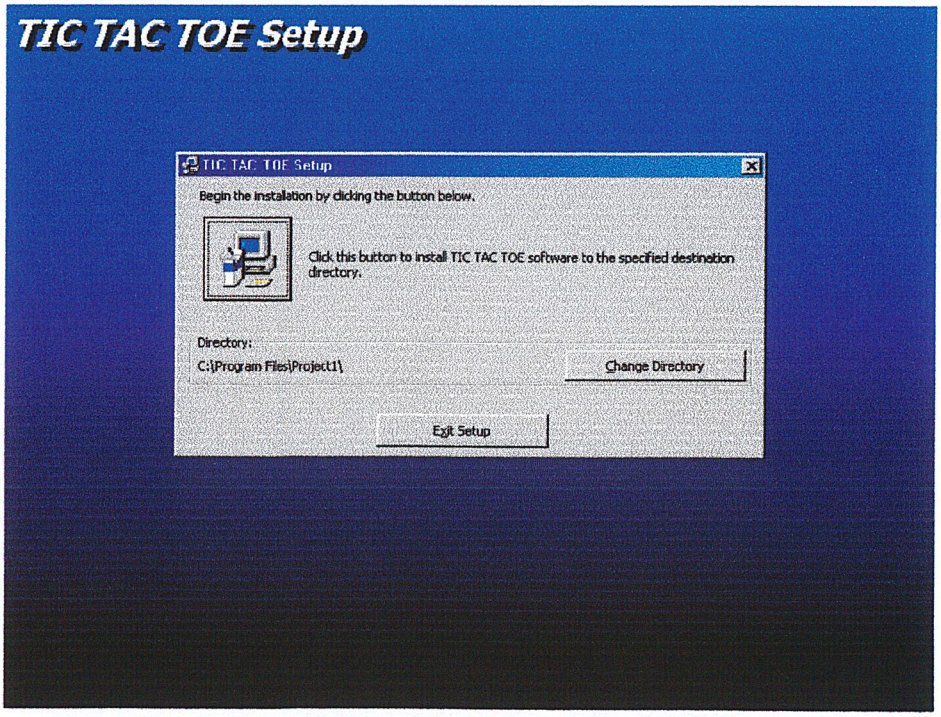
รูปที่ 4.1 แสดงการเข้าที่ Folder “TIC TAC TOE”



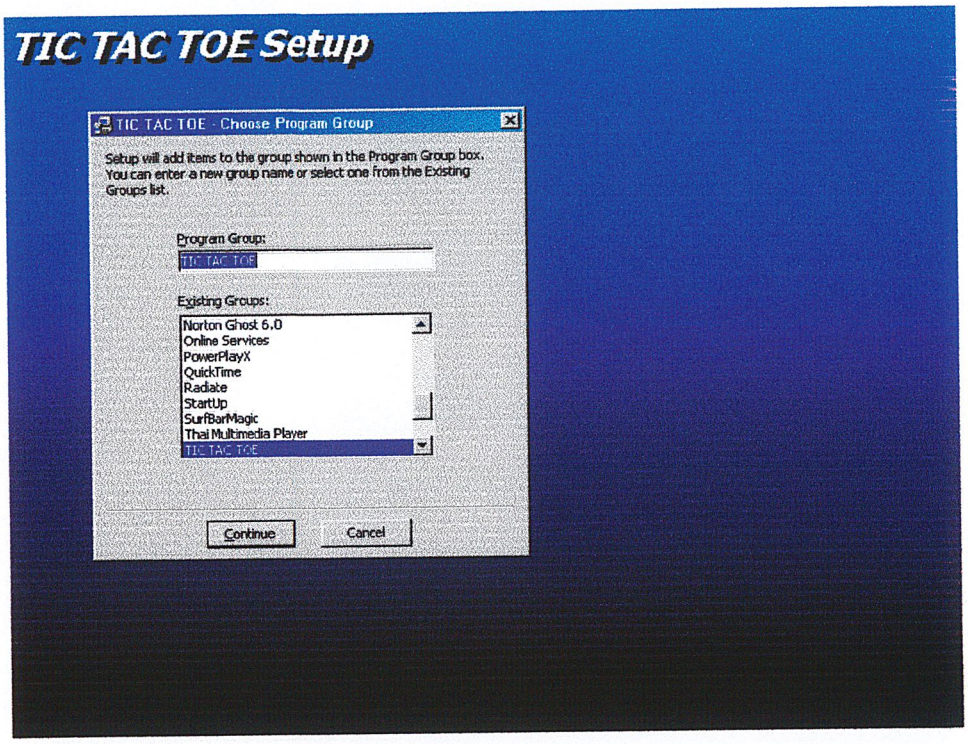
รูปที่ 4.2 แสดงผลของการคลิกที่ไอคอน “setup.exe”



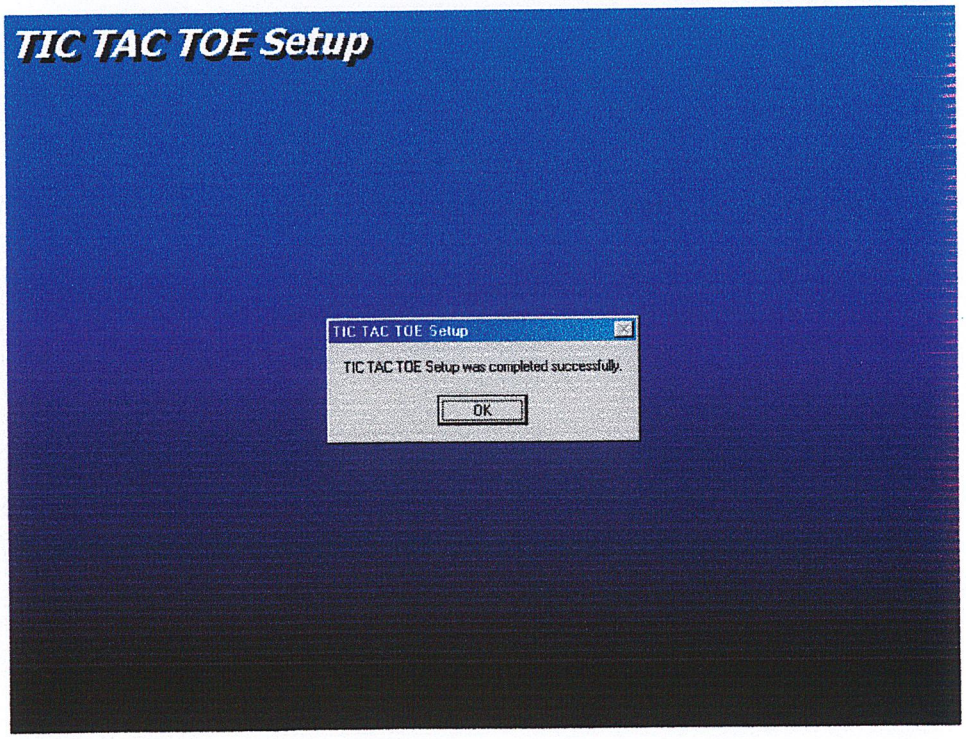
รูปที่ 4.3 แสดงการเข้าสู่การ Set up โปรแกรมเกม



รูปที่ 4.4 เมื่อทำการคลิกที่ปุ่ม “OK” จะทำการเริ่ม Set up



รูปที่ 4.5 แสดงหน้าจอหลังจากคลิกที่ปุ่มรูปคอมพิวเตอร์

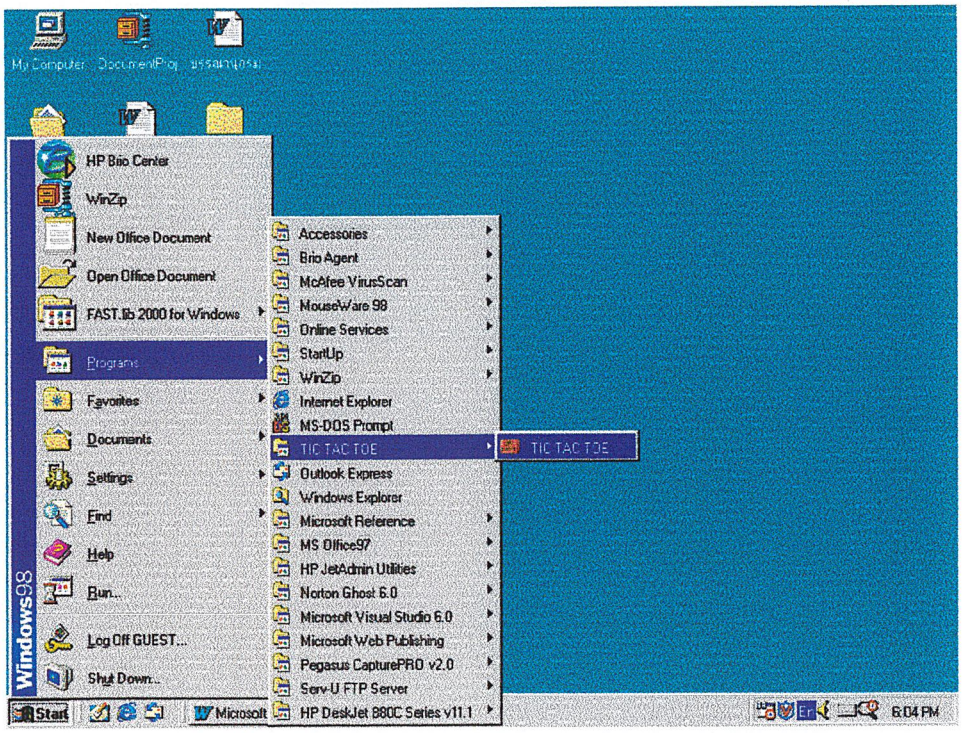


รูปที่ 4.6 แสดงหน้าจอหลังจากคลิกที่ปุ่ม “Continue”

- เมื่อสามารถลงโปรแกรมได้อย่างสมบูรณ์แล้วจะแสดงข้อความขึ้นมาว่า "TIC TAC TOE Setup was completed successfully" แล้วให้ทำการคลิกที่ปุ่ม "OK"

ผลการทดสอบ

- ติดตั้งโปรแกรมและซอฟท์แวร์ที่จำเป็นลงบนเครื่องคอมพิวเตอร์ได้ จะมีไอคอนดังนี้



รูปที่ 4.7 แสดงไอคอนเมื่อทำการติดตั้งเสร็จ

- ผู้ใช้สามารถรันโปรแกรมได้โดยการคลิกที่ไอคอนซึ่งจะแสดงหน้าจอของเกมขึ้น

4.2 ขั้นตอนการทดสอบการรันและประมวลผลภายใต้ระบบที่กำหนด

รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนดโดยคลิกที่ไฟล์ "TAC TAC TOE.exe" จะได้จอภาพดังนี้

- จะทำการทดลองการรันโปรแกรมและทำการตรวจเช็คการใช้งานของปุ่มคำสั่งต่างๆ ในเกมว่าเมื่อทำการกดที่ปุ่มใดๆ แล้ว โปรแกรมสามารถรันผลได้ตามต้องการและแสดงผลการทำงานดังรูป เริ่มต้นจากหน้าจอเมนูของเกมผู้เล่นจะต้องทำการเลือกขนาดของตาราง เงื่อนไขในการชนะที่เมนูบาร์ และเลือกที่จะเป็นฝ่ายเริ่มก่อน หรือให้คอมพิวเตอร์เป็นฝ่ายเริ่มก่อน ซึ่งมีดังนี้

1. แฟ้มเกม ซึ่งมีเมนูย่อยดังนี้

1.1 ขนาดตาราง 3 X 3 ซึ่งมีเมนูย่อย เงื่อนไขการชนะ 3

1.2 ขนาดตาราง 4 X 4 ซึ่งมีเมนูย่อย เงื่อนไขการชนะ 4

1.3 ขนาดตาราง 5 X 5 ซึ่งมีเมนูย่อย เงื่อนไขการชนะ 4 และเงื่อนไขการชนะ 5

1.4 ขนาดตาราง 6 X 6 ซึ่งมีเมนูย่อย เงื่อนไขการชนะ 4 และเงื่อนไขการชนะ 5

1.5 ขนาดตาราง 7 X 7 ซึ่งมีเมนูย่อย เงื่อนไขการชนะ 5 และเงื่อนไขการชนะ 6

1.6 ขนาดตาราง 8 X 8 ซึ่งมีเมนูย่อย เงื่อนไขการชนะ 5, เงื่อนไขการชนะ 6 และเงื่อนไขการชนะ 7

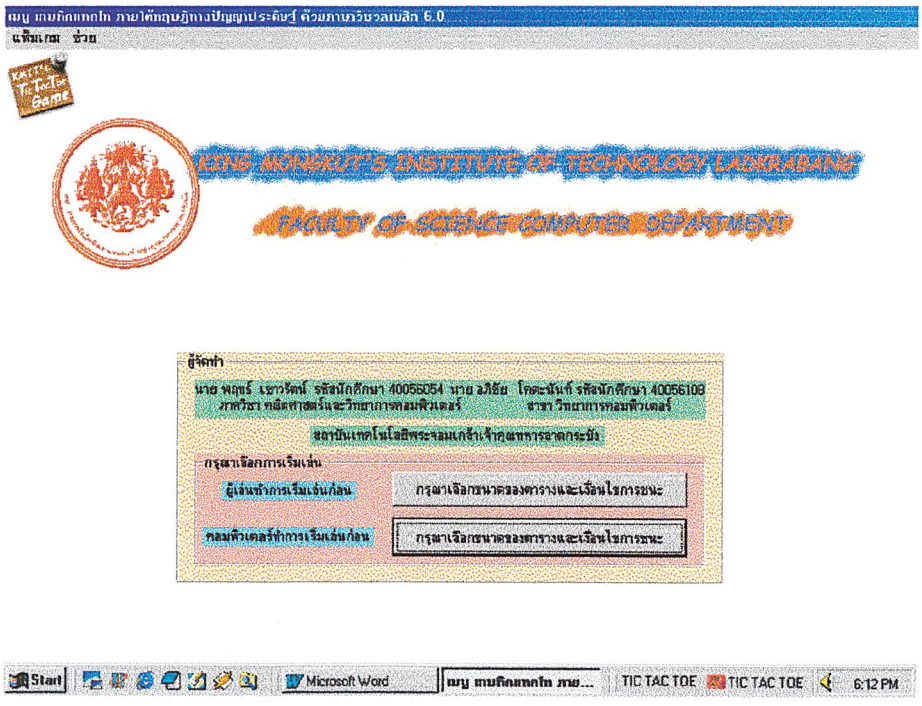
1.7 ขนาดตาราง 9 X 9 ซึ่งมีเมนูย่อยคือ เงื่อนไขการชนะ 5, เงื่อนไขการชนะ 6, เงื่อนไขการชนะ 7 และเงื่อนไขการชนะ 8

1.8 ออกจากข้อมูล

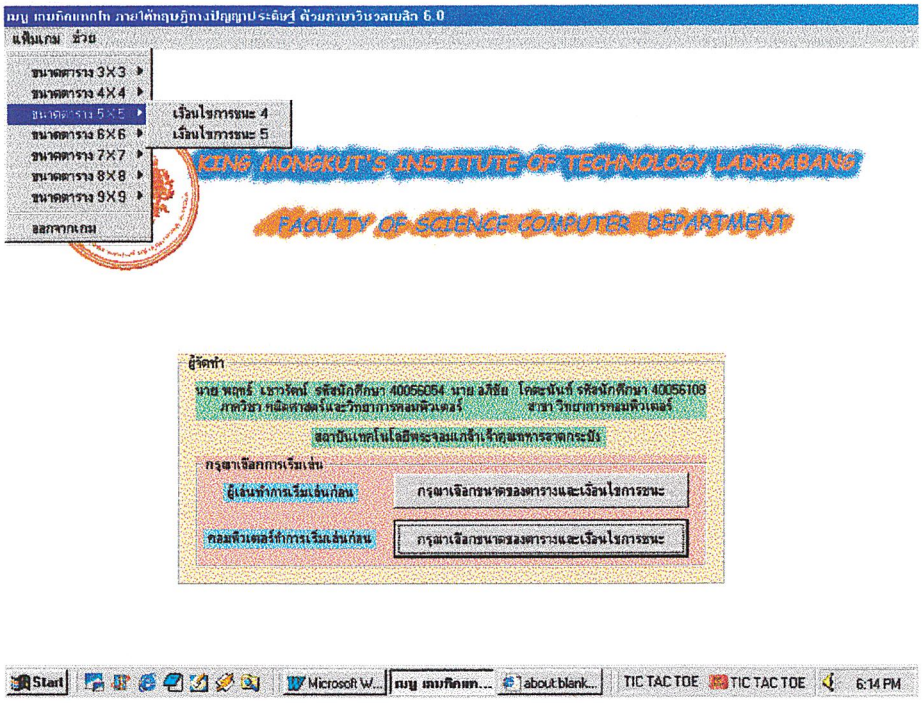
2. ช่วย ซึ่งมีเมนูย่อย คู่มือใช้งาน

3. ปุ่มผู้เล่นเริ่มเดินก่อน

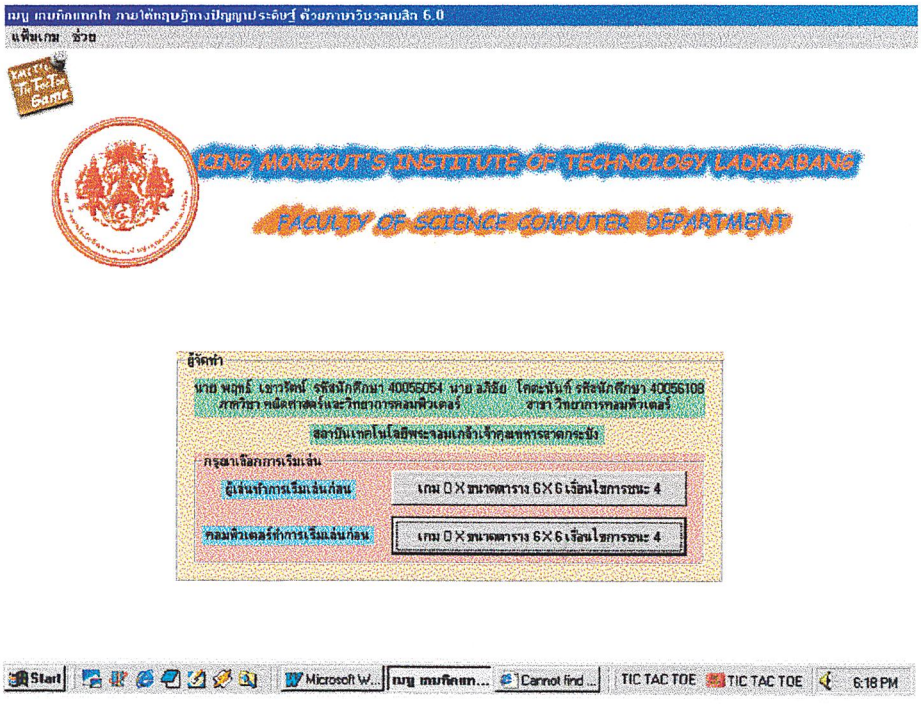
4. ปุ่มคอมพิวเตอร์เริ่มเดินก่อน



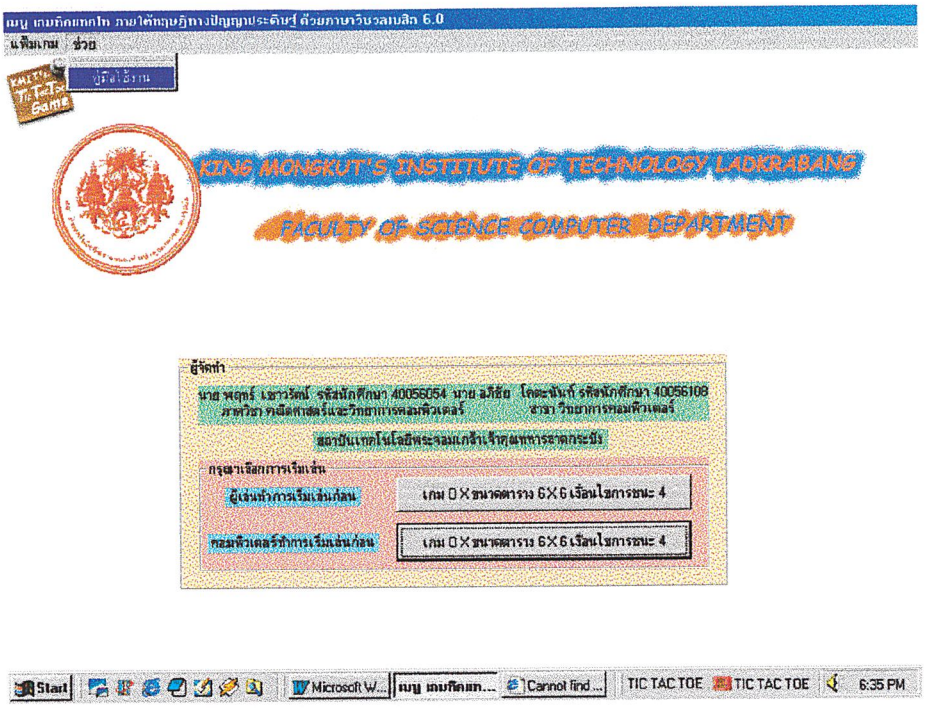
รูปที่ 4.8 แสดงรายการเกมเมนูเริ่มต้นของเกม



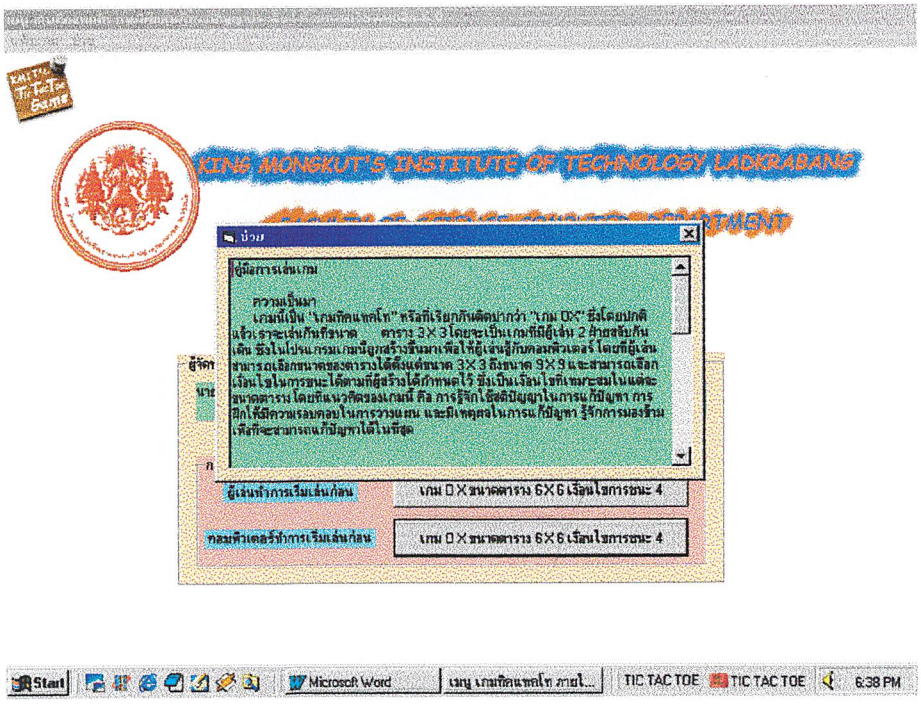
รูปที่ 4.9 แสดงรายการเกมเมนูเริ่มต้นของเกมเมื่อคลิกที่ “แฟ้มเกม”



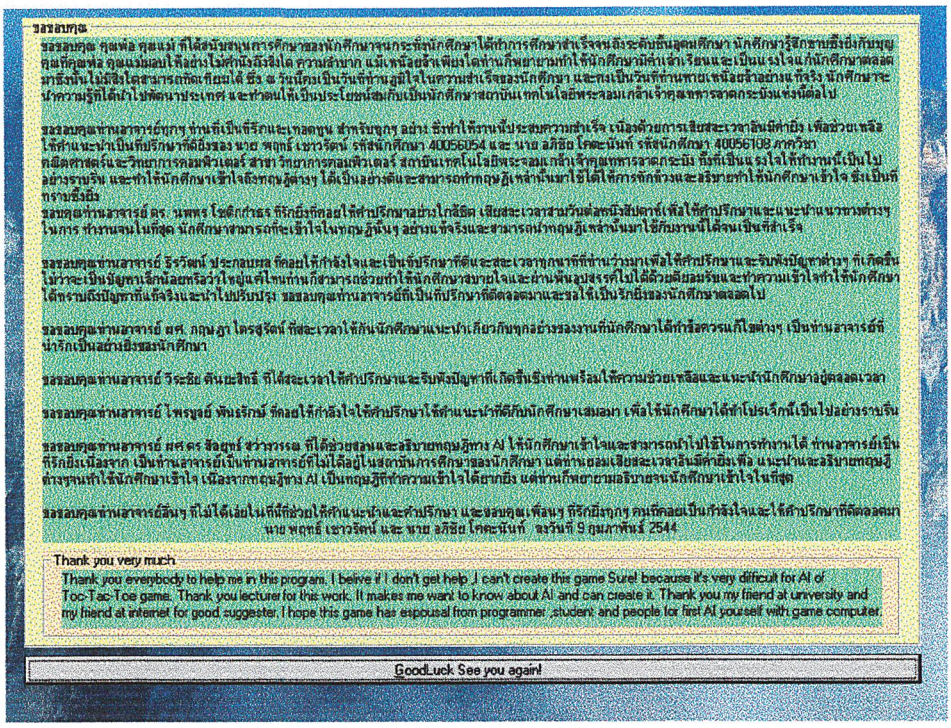
รูปที่ 4.10 แสดงข้อความการเลือกเกมลงในปุ่ม และเมื่อคลิกที่ปุ่มจะทำการสร้างเกมต่อไป



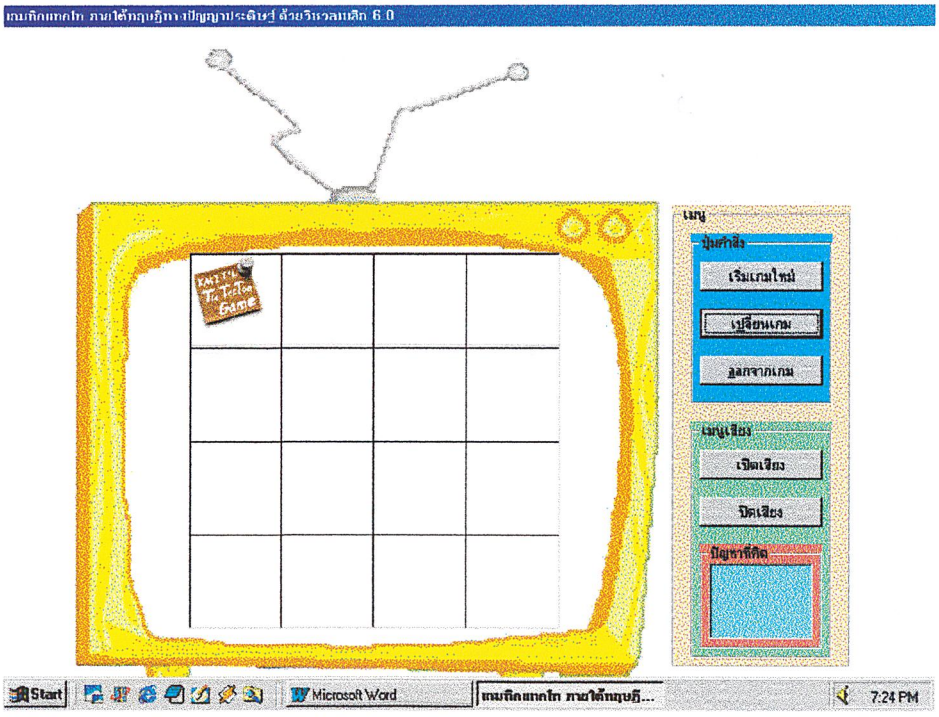
รูปที่ 4.11 เมื่อต้องการความช่วยเหลือในการเกมคลิกที่ปุ่ม “ช่วย” และเลือกที่คู่มือการใช้งาน



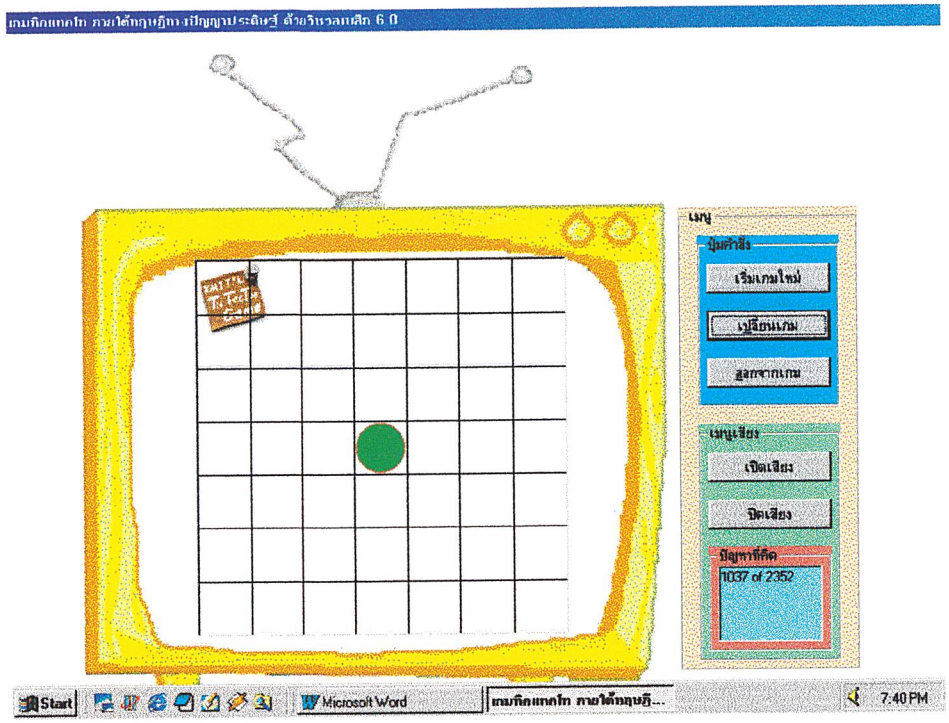
รูปที่ 4.12 หลังจากคลิกที่ปุ่ม “คู่มือ” จะแสดงคู่มือการเล่น



รูปที่ 4.13 เมื่อกดที่ปุ่ม “ออกจากเกม” ในแฟ้มเมนูจะแสดงคำขอบคุณ



รูปที่ 4.14 แสดงเกมที่เลือกเพื่อเริ่มเล่นเกม

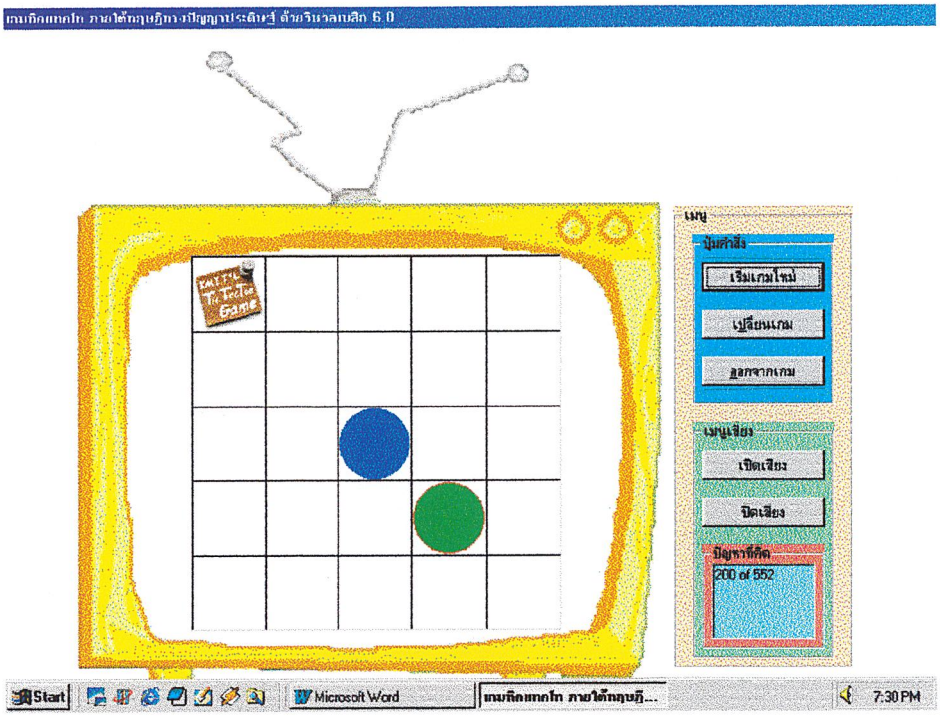


รูปที่ 4.15 แสดงการเริ่มเดินของคอมพิวเตอร์เมื่อคอมพิวเตอร์เริ่มเดินก่อน

ผลการทดสอบที่จอเมนู

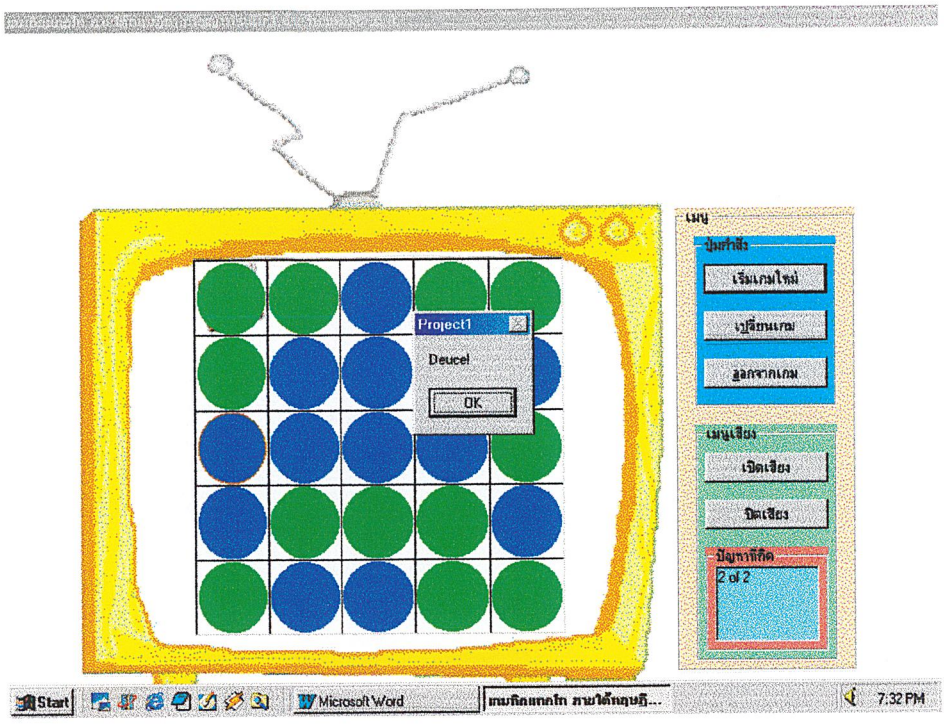
- ทำการทดลองเลือกเกมทุกเกมในเมนูแฟ้มเกมแต่ไม่พบสิ่งใดผิดพลาด สามารถทำงานตามคำสั่งได้อย่างถูกต้องและสมบูรณ์โดยคลิกเลือกแล้วแสดงสิ่งที่เลือกที่ปุ่มเริ่มเล่นเกม
- ทำการทดลองเลือกปุ่ม “ออกจากเกม” จากเมนูแฟ้มเกมปรากฏว่าสามารถใช้ได้งานได้ตามคำสั่งอย่างถูกต้องไม่พบข้อผิดพลาด โดยการออกจากเกมและทำการแสดงหน้าจอขอคุณ
- ทำการทดลองเลือกปุ่ม “ช่วย” จากเมนูแฟ้มเกมและเลือกที่ปุ่ม “คู่มือการใช้งาน” ปรากฏว่าสามารถแสดงหน้าจอความช่วยเหลือได้อย่างถูกต้อง
- ทำการทดลองเลือกปุ่มเริ่มเกมของผู้เล่นหรือคอมพิวเตอร์ ซึ่งจะแสดงหน้าจอการเริ่มเล่นเกมได้อย่างถูกต้องโดยแสดงหน้าจอเกมขึ้นมาเพื่อทำการเล่นต่อไป
- ทุกปุ่มคำสั่งทำงานได้ตามต้องการทุกประการดังแสดงผลภาพที่ได้กล่าวมา ซึ่งทำการทดลองซ้ำๆ กันหลายๆ ครั้งแต่ไม่พบปัญหาแต่อย่างใดและสุดท้ายของหน้านี้คือการตรวจการแสดงเสียงมัลติมีเดียได้ตามต้องการคือสามารถแสดงเสียงในเกมได้เป็นอย่างดี
- จากนั้นทำการทดลองในหน้าเกม โดยการทดลองเล่นเกมและใช้ปุ่มคำสั่งต่างๆ ซึ่งประกอบด้วยปุ่มคำสั่งต่างๆ ดังนี้ และทำการแสดงการตรวจสอบดังภาพ

1. ตารางสำหรับลงตาเดิน
2. ช่องแสดงข้อมูลในการหาคำตอบ
3. ปุ่ม "เริ่มเกมใหม่"
4. ปุ่ม "เปลี่ยนเกม"
5. ปุ่ม "ออกจากเกม"
6. ปุ่ม "เปิดเสียง"
7. ปุ่ม "ปิดเสียง"

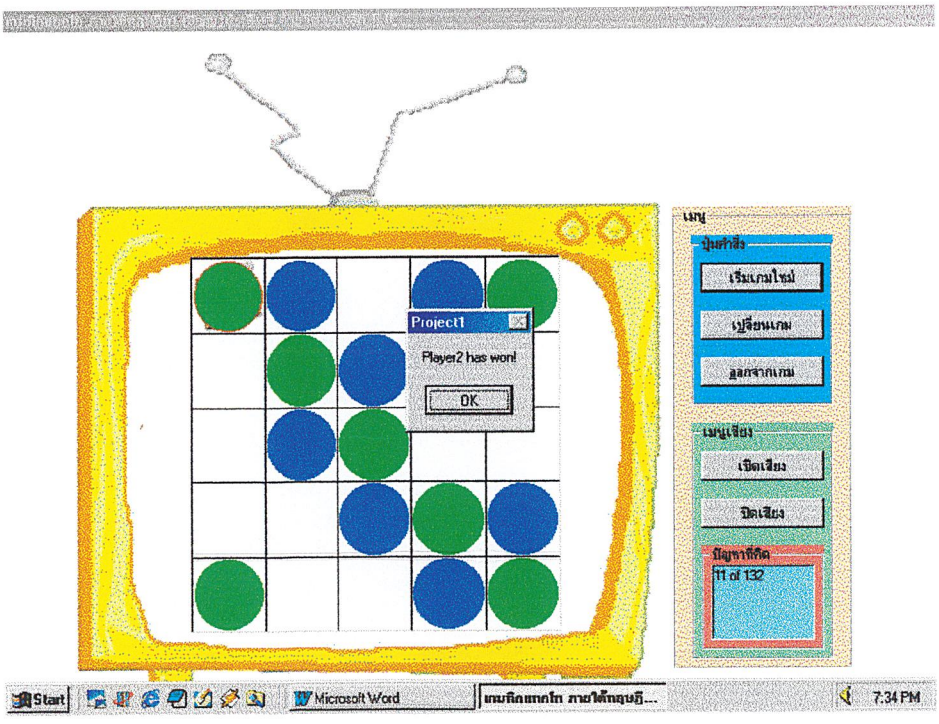


รูปที่ 4.16 แสดงการลงเดินเมื่อผู้เล่นเริ่มเดินก่อนและคอมพิวเตอร์ลงตามมา

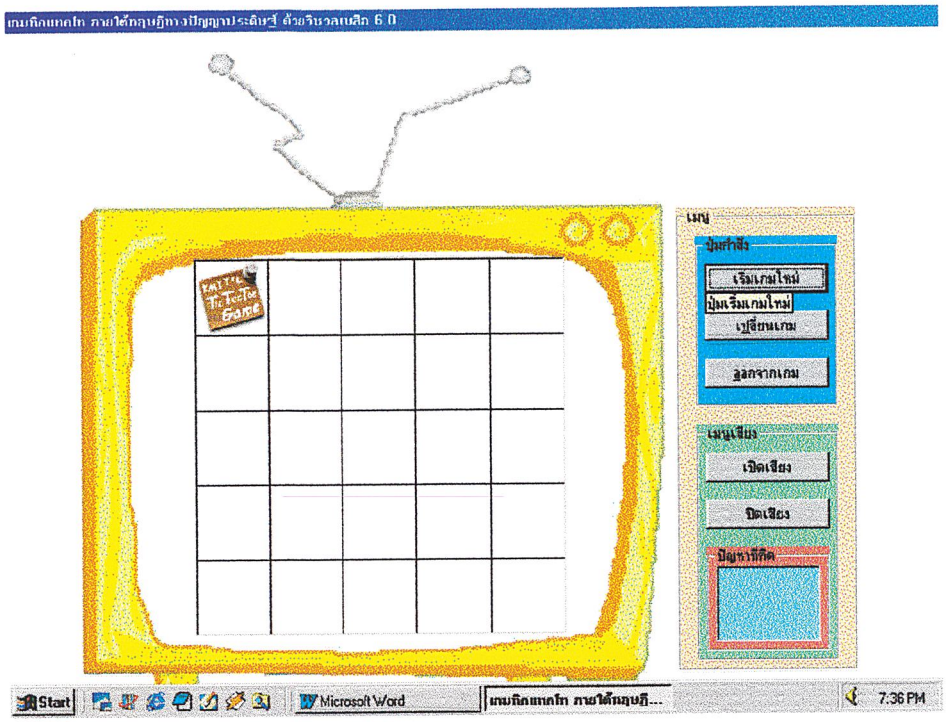
รูปที่ 4.17 แสดงช่องปัญหาที่คิดเพียง 32 จากทั้งหมด 56 ในตารางขนาด 3 X 3



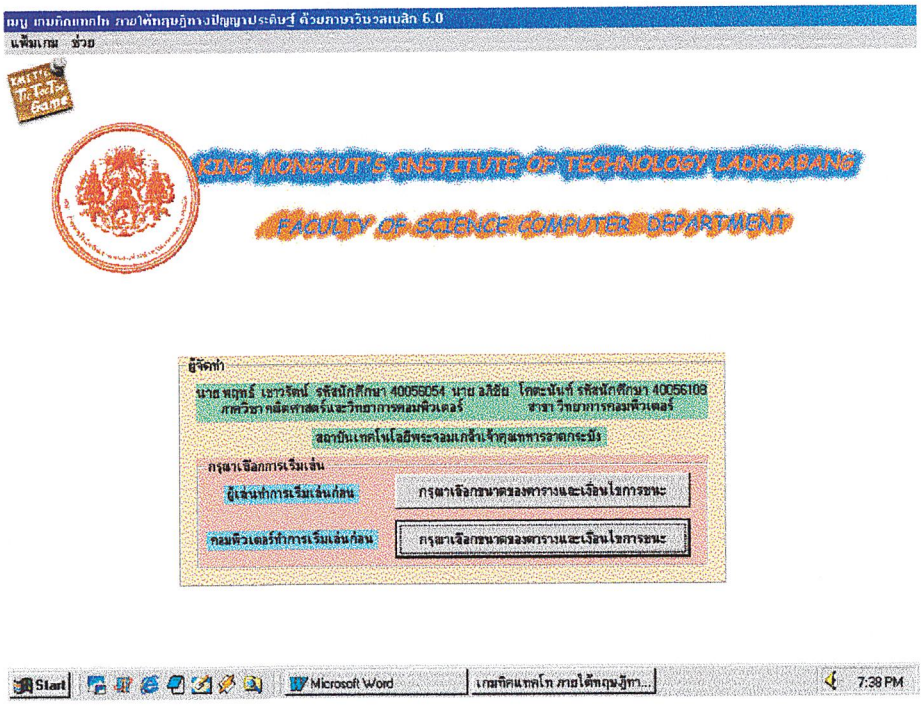
รูปที่ 4.18 แสดงการเสมอ



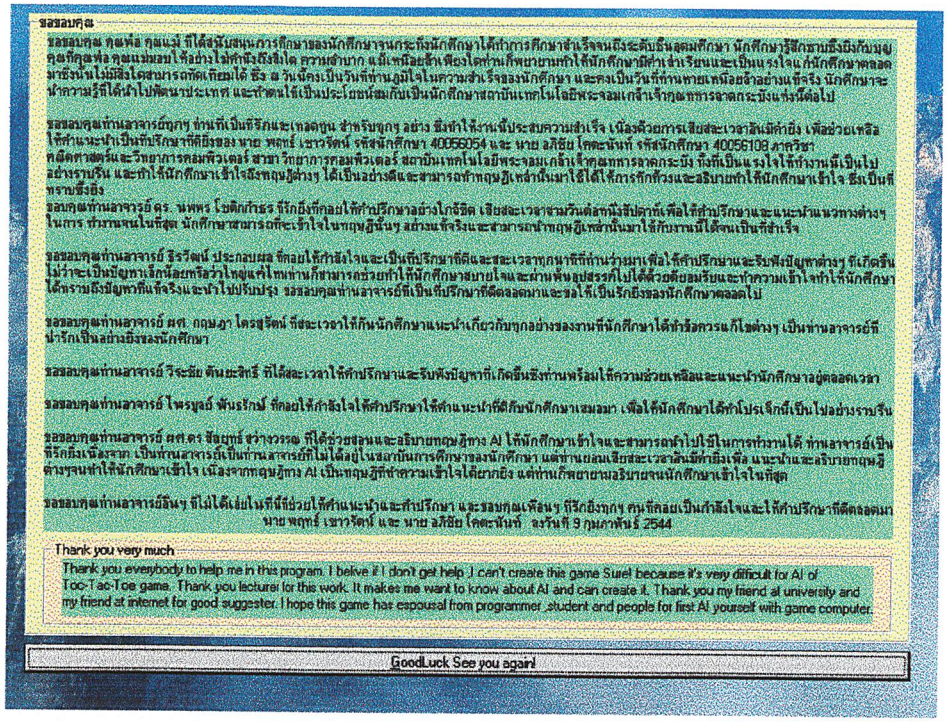
รูปที่ 4.19 แสดงการชนะ



รูปที่ 4.20 แสดงการคลิกที่ปุ่ม “เริ่มเกมใหม่” ซึ่งจะแสดงตารางใหม่



รูปที่ 4.21 แสดงการคลิกที่ปุ่ม “เปลี่ยนเกม” ซึ่งจะกลับสู่หน้าเมนูเริ่มต้น



รูปที่ 4.22 แสดงการคลิกที่ปุ่ม “ออกจากเกม” ซึ่งจะเข้าสู่หน้าคำขอบคุณ

ผลการทดสอบที่หน้าจอเกมและหน้าจอขอบคุณ

- ทำการทดสอบคลิกลงเดินในตารางเดิน ไม่พบข้อผิดพลาด สามารถลงเดินได้ทั้งฝ่ายผู้เล่นและฝ่ายคอมพิวเตอร์
- ทำการทดสอบช่องแสดงปัญหาที่คิด สามารถแสดงปัญหาที่คิดได้จากปัญหาทั้งหมด ไม่พบข้อผิดพลาด
- ทำการทดสอบผลการแสดงผลเสมอ ไม่พบข้อผิดพลาด
- ทำการทดสอบผลการชนะ ไม่พบข้อผิดพลาด
- ทำการทดสอบเลือกปุ่ม “เริ่มเกมใหม่” จะทำการแสดงหน้าจอเกมเดิมใหม่อีกครั้งและทำการเริ่มต้นเกมที่ได้เคยเลือกไว้ใหม่ โดยไม่พบข้อผิดพลาด
- ทำการทดสอบเลือกปุ่ม “เปลี่ยนเกม” จะทำการแสดงหน้าจอเมนูขึ้นมาใหม่เพื่อให้ผู้เล่นทำการเลือกเกมใหม่ได้ โดยไม่พบข้อผิดพลาด
- ทำการทดสอบเลือกปุ่ม “ออกจากเกม” ปรากฏว่าสามารถใช้ได้งานได้ตามคำสั่งอย่างถูกต้อง ไม่พบข้อผิดพลาด โดยการออกจากเกมและทำการแสดงหน้าจอขอคุณ โดยไม่พบข้อผิดพลาด

- ทำการทดสอบเลือกปุ่ม “เปิดเสียง” จะทำการปิดเสียงเดิมก่อนและทำการเปิดเพลงใหม่ ซึ่งไม่มีข้อผิดพลาด
- ทำการทดสอบเลือกปุ่ม “ปิดเสียง” จะทำการปิดเสียงที่มีเปิดอยู่ ซึ่งไม่มีข้อผิดพลาด
- และสุดท้ายทำการทดสอบเลือกปุ่มจบโปรแกรม ปุ่ม "GoodLuck see you again!" จะทำการออกจากโปรแกรมเกม

4.3 ขั้นตอนการทดสอบการตอบโต้ของคอมพิวเตอร์

- ทำการเลือกเดิน ณ ตำแหน่งใดๆ และคอมพิวเตอร์สามารถลงตาเดินโต้ตอบได้

ผลการทดสอบ

- ฝ่ายคอมพิวเตอร์ทำการเดินตามค่าแสดงความเหมาะสมของเหตุการณ์กำหนดขึ้นโดยเลือกจากฮิลริสติกฟังก์ชั่น
- ฝ่ายคอมพิวเตอร์จะเลือกตำแหน่งที่จะนำไปสู่ชัยชนะของฝ่ายคอมพิวเตอร์

4.4 ขั้นตอนการทดสอบการหาข้อผิดพลาดของโปรแกรม

ในขั้นตอนการทดสอบเพื่อหาข้อผิดพลาดจะไม่มีรูปแบบที่ชัดเจน แต่จะทำการทดสอบโดยการเล่นเกมหลายๆ รอบ เพื่อหาข้อผิดพลาดและจากการทดสอบ ซึ่งไม่พบข้อผิดพลาด

4.5 ปัญหาและการวิเคราะห์

จากผลการทดสอบโปรแกรมทำให้ทราบถึงปัญหา ดังนี้

- การตอบโต้ของฝ่ายคอมพิวเตอร์

การตอบโต้ของฝ่ายคอมพิวเตอร์เป็นปัญหาที่สำคัญที่เกิดขึ้น เนื่องจากฮิวริสติกฟังก์ชันอาจจะไม่มีประสิทธิภาพมากถึงกับทำให้ฝ่ายคอมพิวเตอร์ชนะตลอดได้ อาจมีการเสมอ

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

การทำงานโครงงานนี้มีการวางแผนขั้นตอนการทำงานเป็นอย่างดี แต่ถึงแม้จะได้วางแผนแต่ละขั้นตอนไว้เป็นอย่างดี ก็ยังมีปัญหากับขั้นตอนบางขั้นตอน ซึ่งบางขั้นตอนเกิดจากความล่าช้าในการคิดระเบียบวิธีการโปรแกรมมิ่ง ซึ่งจำต้องศึกษาจากบทความภาษาอังกฤษตามหนังสือและตามโฮมเพจต่างๆ เนื่องจากต้องใช้ประสบการณ์และระยะเวลาพอสมควร ถึงจะเข้าใจและสามารถนำความรู้ต่างๆที่ได้จากหนังสือและโฮมเพจต่างๆ ได้เป็นอย่างดี

โดยขั้นตอนต่างๆสรุปได้ดังนี้

- ขั้นตอนการออกแบบระบบเกม
- ขั้นตอนการออกแบบอินเทอร์เฟสที่ใช้ติดต่อกับผู้เล่น
- ขั้นตอนการเขียนโปรแกรม

5.1 ขั้นตอนการออกแบบระบบเกม

ขั้นตอนการออกแบบระบบเกมนั้นจะต้องใช้เวลานานพอสมควร เนื่องจากจะต้องทำการออกแบบระบบเกมทั้งหมด ซึ่งระบบเกมที่ต้องออกแบบนั้นจะต้องสามารถใช้งานได้ง่าย และจะต้องออกแบบการตอบโต้ของคอมพิวเตอร์ด้วยซึ่งจะต้องครอบคลุมทุกๆ ขนาดตารางที่สร้างขึ้น จึงจะสามารถทำให้การตอบโต้ของคอมพิวเตอร์มีประสิทธิภาพ แล้วจึงนำมารวบรวมและสรุปเป็นรูปแบบของเกมในงานวิจัย

5.2 ขั้นตอนการออกแบบอินเทอร์เฟสที่ใช้ติดต่อกับผู้เล่น

ขั้นตอนนี้สามารถทำควบคู่ไปได้กับขั้นตอนการเขียนโปรแกรม ซึ่งในขั้นตอนนี้จะทำการออกแบบอินเทอร์เฟสต่างๆ ที่ใช้ติดต่อกับผู้เล่น โดยที่จะสร้างอินเทอร์เฟสขึ้นมาโดยการใช้ฟังก์ชันในโปรแกรม Microsoft Visual Basic 6.0 สร้างขึ้นมา

5.3 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้จะใช้เวลานานที่สุดในงานวิจัย เนื่องจากเป็นเรื่องใหม่ซึ่งผู้วิจัยต้องทำการศึกษานานพอสมควร จึงลงมือเขียนโปรแกรม ซึ่งระหว่างการเขียนโปรแกรมนั้น ผู้วิจัยจะต้องทำการทดสอบโปรแกรมที่เขียนขึ้นมาด้วย จึงทำให้ใช้เวลานานพอสมควร ซึ่งขั้นตอนการทำงานในส่วนนี้มีซอฟท์แวร์ที่ใช้พัฒนาคือ

- Microsoft Visual Basic 6.0

สรุปแล้วในการทำงานวิจัยเพื่อสร้างเกมโดยตอบโต้กับคอมพิวเตอร์ การทำงานต่างๆ ปัญหาที่เกิดขึ้น ตลอดจนการได้ติดต่อกับผู้ผลิตเกมในต่างประเทศ เพื่อทำการศึกษานี้ ทำให้ผู้วิจัยทราบถึงแนวทางในการผลิตเกมในตลาดซอฟท์แวร์ไทย เกมในปัจจุบัน บางเกมถูกผลิตโดยคนไทย แต่ไม่สามารถบอกได้ว่าเป็นคนไทย ก็เนื่องมาจากบริษัทต่างๆที่ถือลิขสิทธิ์เป็นบริษัทต่างประเทศ ดังนั้นการที่คนไทยจะก่อตั้งบริษัทซอฟท์แวร์ขึ้นก็มีความเป็นไปได้ แต่ปัญหาที่สำคัญที่สุดในตลาดซอฟท์แวร์ไทยก็คือปัญหาละเมิดลิขสิทธิ์ซอฟท์แวร์ ซึ่งเป็นสาเหตุสำคัญที่ทำให้คนไทยม่กล้าที่จะผลิตซอฟท์แวร์คุณภาพออกมาแข่งขันกัน และปัญหาอีกอย่างในตลาดซอฟท์แวร์เกมก็คือ การให้ความสนใจในวิธีการผลิตของคนไทยน้อยเกินไป ในบางครั้งใช้วิธีในการผลิตที่ลำบากพอสมควร แต่พอนำออกมาจำหน่ายแล้วไม่คุ้มค่ากับการลงทุน ในขณะเดียวกันก็บริโภคหรือให้ความสนใจในเนื้อหาของเกมมาก ทำให้ความรู้ในการผลิตยังไม่สามารถทำได้เทียบเท่ากับชาวต่างชาติซึ่งเป็นชาติผู้ผลิต ดังนั้นโครงงานนี้น่าจะเป็นแนวทางหนึ่งที่จะแนะนำให้ผู้สนใจได้นำไปศึกษา และผลิตเกมที่สามารถนำออกมาจำหน่ายได้จริงต่อไป

ภาคผนวก

คู่มือการเล่น

คู่มือการเล่นเกม

ความเป็นมา

เกมนี้เป็น "เกมทิคแทคโท" หรือที่เรียกกันติดปากว่า "เกม OX" ซึ่งโดยปกติแล้วเราจะเล่นกันที่ขนาด ตาราง 3 X 3 โดยจะเป็นเกมที่มีผู้เล่น 2 ฝ่ายสลับกันเดิน ซึ่งในโปรแกรมเกมนี้ถูกสร้างขึ้นมาเพื่อให้ผู้เล่นสู้กับคอมพิวเตอร์ โดยที่ผู้เล่นสามารถเลือกขนาดของตารางได้ตั้งแต่ขนาด 3 X 3 ถึงขนาด 9 X 9 และสามารถเลือกเงื่อนไขในการชนะได้ตามที่ผู้สร้างได้กำหนดไว้ ซึ่งเป็นเงื่อนไขที่เหมาะสมในแต่ละขนาดตาราง โดยที่แนวคิดของเกมนี้ คือ การรู้จักใช้สติปัญญาในการแก้ปัญหา การฝึกให้มีความรอบคอบในการวางแผน และมีเหตุผลในการแก้ปัญหา รู้จักการมองข้ามเพื่อที่จะสามารถแก้ปัญหาได้ในที่สุด

ความต้องการของระบบที่ใช้

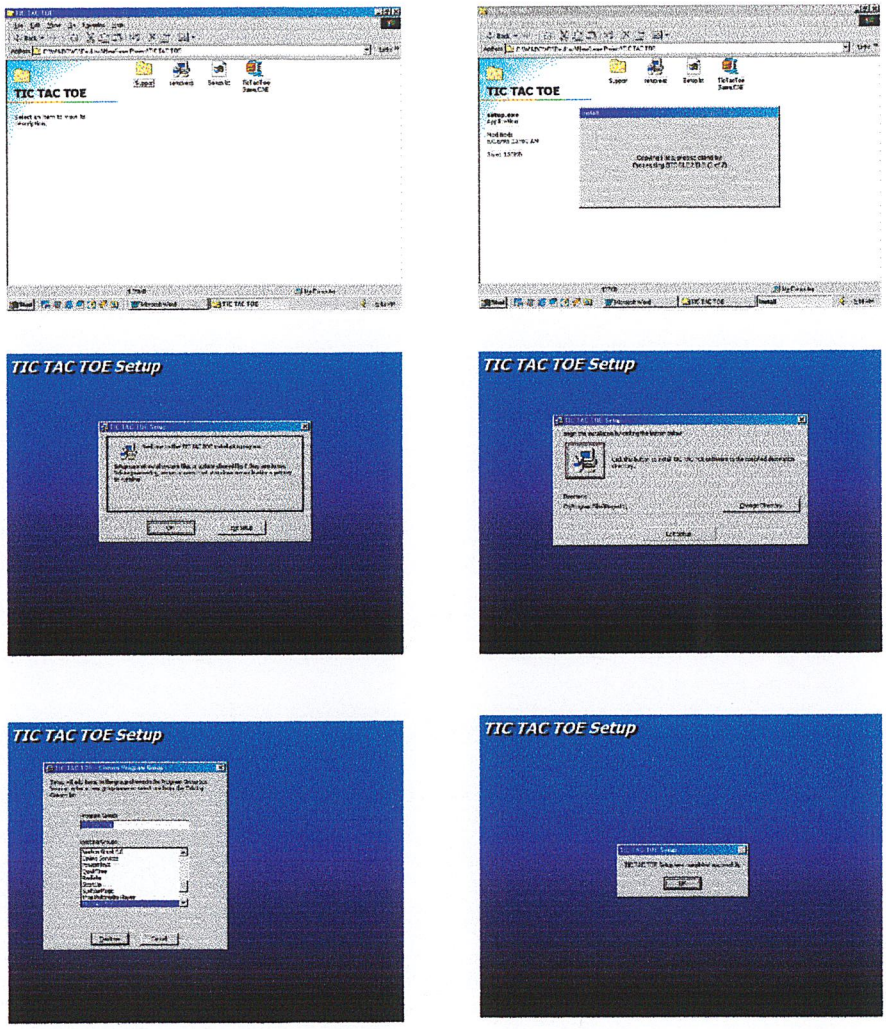
1. คอมพิวเตอร์ติดตั้งระบบปฏิบัติการ Windows 95 หรือมากกว่านั้น
2. เครื่องคอมพิวเตอร์รุ่น Pentium 100 ขึ้นไป
3. หน่วยความจำไม่ควรต่ำกว่า 16 MB เป็นอย่างน้อย
4. Keyboard, Mouse และ จอ VGA หรือสูงกว่า
5. เนื้อที่ว่างในฮาร์ดดิสก์ประมาณ 100 MB ขึ้นไป
6. อุปกรณ์อินพุทเมาส์

ขั้นตอนการใช้โปรแกรม

1. คลิกที่ไอคอน "Setup.exe" ทำการลงโปรแกรม
2. เมื่อต้องการเล่นทำการเลือกคลิกที่ไอคอน "TIC TAC TOE.exe" โปรแกรมจะทำการ Run ขึ้นมา
3. คลิกเลือกเกมที่ต้องการที่เมนูบาร์ หรือถ้าต้องการทราบวิธีเล่นคลิกที่เมนู "ช่วย" และคลิกที่ "คู่มือการใช้งาน"
4. ทำการคลิกเลือกว่าจะให้ผู้เล่นเดินก่อน หรือให้คอมพิวเตอร์เดินก่อน ต่อจากนั้นเกมจะทำการแสดงเกมที่ต้องการเล่นออกมาในหน้าจอเกมโดยฝ่ายผู้เล่นเป็นสีน้ำเงิน ฝ่ายคอมพิวเตอร์เป็นสีเขียว
5. ทำการเล่นโดยคลิกที่ตาราง
6. ถ้าต้องการเริ่มเกมเดิมใหม่อีกครั้งคลิกที่ ปุ่ม "เริ่มเกมใหม่"
7. ถ้าต้องการเปลี่ยนเกมใหม่อีกครั้งคลิกที่ ปุ่ม "เปลี่ยนเกม"

8. ถ้าต้องการออกจากเกมคลิกที่ ปุ่ม “ออกจากเกม” จะทำการแสดงหน้าจอขอบคุณและถ้าคลิกที่ ปุ่ม “GoodLuck see you again!” ก็จะจบโปรแกรม
9. ถ้าต้องการเปิดเสียงเพลงในการเล่นคลิกที่ ปุ่ม “เปิดเสียง”
10. ถ้าต้องการปิดเสียงเพลงในการเล่นคลิกที่ ปุ่ม “ปิดเสียง”

ต่อจากนี้จะทำการแสดงการลงโปรแกรมโดยไล่จากซ้ายไปขวาลงไปดังภาพ

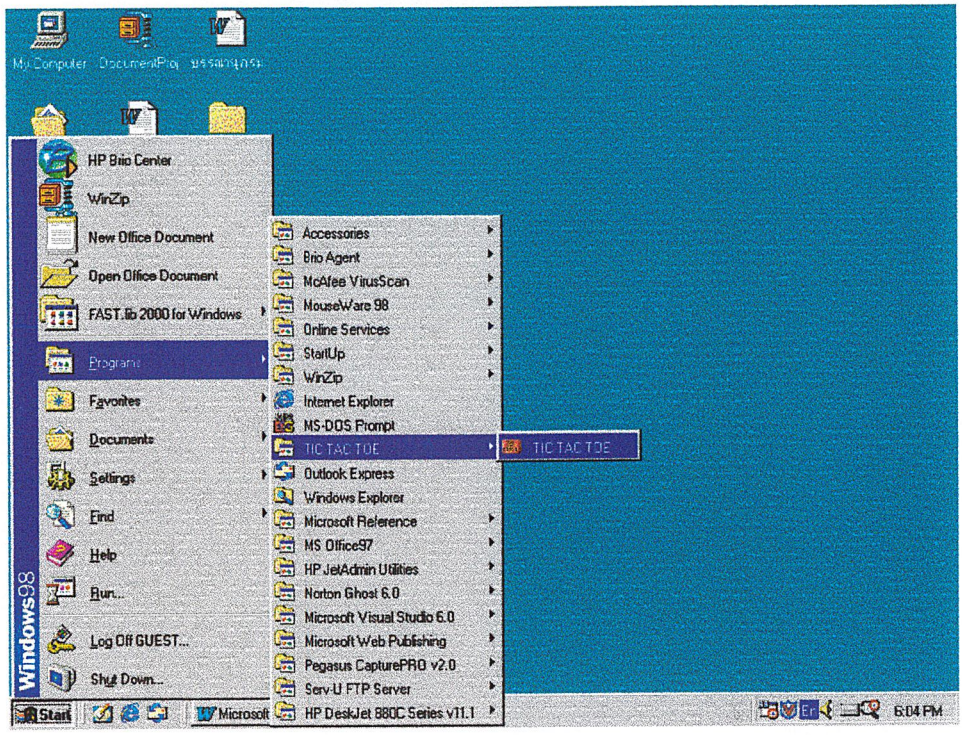


รูปที่ 1 ภาคผนวก

ในการลงโปรแกรมให้คลิกที่ไอคอน “Setup.exe” ใน Folder TIC TAC TOE

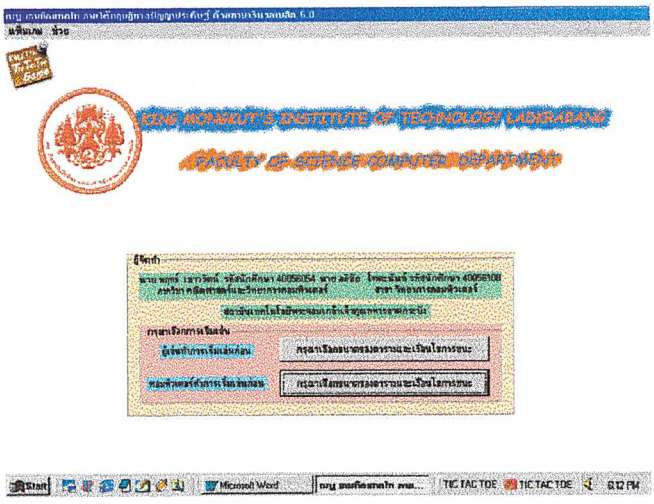
1. ทำการที่ ปุ่ม “OK” เพื่อทำการลงโปรแกรม
2. กำหนดที่เก็บของโปรแกรมแล้วคลิกที่รูปคอมพิวเตอร์
3. จากนั้นคลิกที่ ปุ่ม “Continue”
4. จากนั้นคลิกที่ ปุ่ม “OK”

เมื่อทำการลงโปรแกรมเสร็จเรียบร้อยแล้วจะแสดงหน้าจอดังนี้

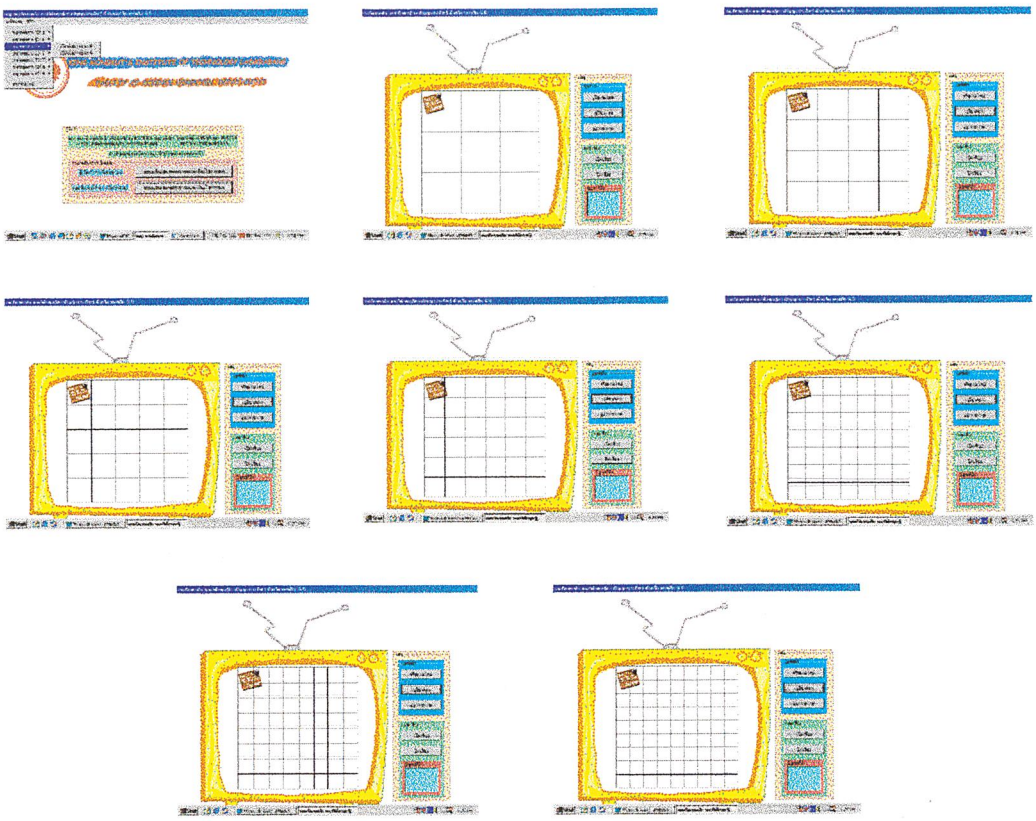


รูปที่ 2 ภาคผนวก

เมื่อทำการ Setup เรียบร้อยแล้วจะมีหน้าจอดังรูปที่ 2 ภาคผนวก เมื่อคลิกที่ไอคอน TIC TAC TOE ในรูปที่ 2 แล้วจะทำการ Run โปรแกรมขึ้น โดยต่อไปนี้จะแสดงหน้าจอแรกของโปรแกรมเกม ดังรูปที่ 3 ภาคผนวก และรูปของเกมทั้งหมด



รูปที่ 3 ภาคผนวก

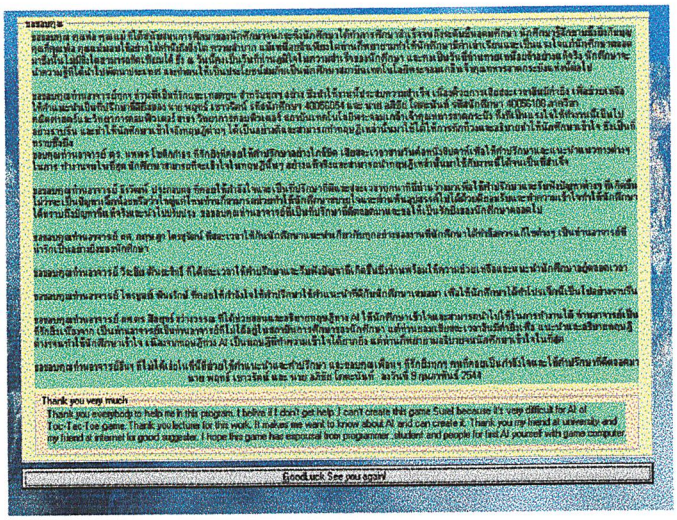


รูปที่ 4 ภาคผนวก แสดงเมนูแฟ้มข้อมูลและตารางเกมทั้งหมด

ต่อไปจะแสดงภาพการเลือกที่เมนูช่วย เพื่อแสดงคู่มือการใช้

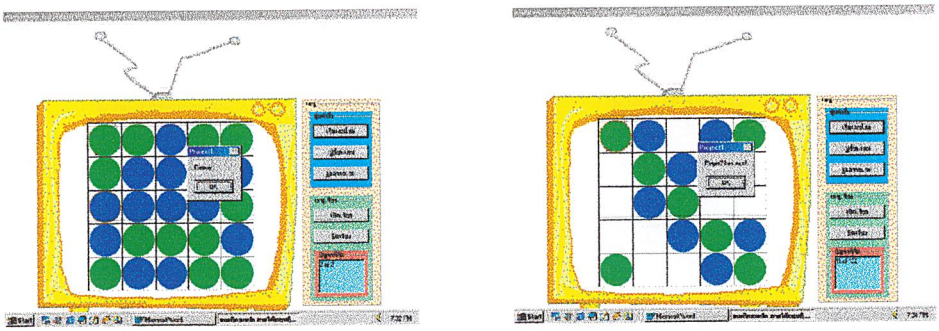
รูปที่ 5 ภาคผนวก

และการออกจากโปรแกรมโดยเมื่อคลิกที่ ปุ่ม “ออกจากโปรแกรม” แล้วจะแสดงดังรูปที่ 6 ภาคผนวก



รูปที่ 6 ภาคผนวก เมื่อทำการคลิกที่ ปุ่ม “ GoodLuck see you again!” จะทำการออกจากโปรแกรม

เมื่อมีการเสมอหรือชนะจะแสดงดังรูปที่ 7 ภาคผนวก โดยเรียงจากภาพซ้ายไปขวา



รูปที่ 7 ภาคผนวก

บรรณานุกรม

พุฒิพงษ์ นาคะปัท. **การเขียนเกมบนวินโดวส์ด้วย Visual Basic ครอบคลุม version 5.0-6.0.** กรุงเทพฯ : บริษัท ซีเอ็ดยูเคชัน จำกัด (มหาชน), 2542.

นิรุธ อำนวยศิลป์. **Visual C++ version 6.0 ฉบับเพื่อการใช้งานจริง.** พิมพ์ครั้งที่ 3. กรุงเทพฯ : บริษัท ซัคเซส มีเดีย จำกัด, 2539.

Elaine Rich and Kevin Knight. **Artificial Intelligence** : McGraw-Hill.INC, Singapore,1983.

Elaine Rich. **Artificial Intelligence** : McGraw-Hill.INC, Singapore,1984.

Nils, J. Nilsson. **Artificial Intelligence** : A New Synthesis, Morgan Kaufmann Publishers. INC, California America,1998.

Winston, Patrick Henry. **Artificial Intelligence.** 3rd ed : A New Synthesis, Morgan Kaufmann Publishers. INC, California America,1992.

George F. Luger and William A. Stubblefield. **Artificial Intelligence** : A New Synthesis, The Benjamin/ Cummigs Publishing Company. INC, California America,1992.

Giorgio P. Ingargiola. "**MiniMax with Alpha-Beta.**" [Online]. Availble : http://yoda.cis.temple .edu:8080/UGAIWWW/lectures/alpha-beta.html. 1998.

Giorgio P. Ingargiola. "**MiniMax with Alpha-Beta.**" [Online]. Availble : http://www.cis.temple.edu/~ingargio/cis587/readings/alpha-beta.html. 1998.

Giorgio P. Ingargiola. "**MiniMax with Alpha-Beta.**" [Online]. Availble : http://www.cis.temple.edu/~ingargio/cis587/readings/alpha-beta-example.html. 1998.

Nick. "**Minimax Algorithm with Alpha-Beta Pruning.**" [Online]. Availble : http://www.cee.hw.ac.uk/~nick/oddsan/minimax.html. 1998.