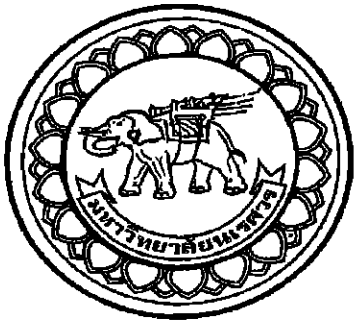


เกม Attaxx

Attaxx game

นายจิรวัฒน์ เธียรพิเชษฐพงศ์ รหัส 46360020

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2549

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน เกม Attaxx

ผู้ดำเนินโครงงาน นายจิรวัฒน์ เธียรพิเชษฐพงศ์ รหัส 46360020

อาจารย์ที่ปรึกษา ดร.สุรเดช จิตประไพกุลศาล

สาขาวิชา วิศวกรรมคอมพิวเตอร์

ภาควิชา วิศวกรรมไฟฟ้าและคอมพิวเตอร์

ปีการศึกษา 2560

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ

(ดร.สุรเดช จิตประไพกุลศาล)

.....กรรมการ

(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ

(ดร.ไพศาล มุณีสว่าง)

หัวข้อโครงงาน	เกม Attaxx
ผู้ดำเนินโครงงาน	นายจิรวัฒน์ เธียรพิเชษฐพงศ์ รหัส 46360020
อาจารย์ที่ปรึกษา	ดร.สุรเดช จิตประไพกุลศาล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2550

บทคัดย่อ

โครงงานนี้ได้พัฒนาเกม Attaxx และโปรแกรมประเภทปัญญาประดิษฐ์ (Artificial Intelligence) ที่ใช้กับเกมในการจำลองผู้เล่นฝ่ายตรงข้าม โดยจะพัฒนาด้วยโปรแกรมมาโครมีเดียแฟลช (Macromedia Flash) ที่กำลังเป็นที่นิยมอย่างแพร่หลายในปัจจุบัน ซึ่งจะแบ่งการพัฒนาเป็นสองส่วน คือ ส่วนของตัวเกมที่ตอบสนองกับผู้เล่นด้วยภาพเคลื่อนไหว (Animation) และส่วนของ ปัญญาประดิษฐ์ หรือ AI (Artificial Intelligence) ที่สามารถคำนวณหาทางเลือกในการเล่นที่ดีที่สุดเพื่อชัยชนะ

Project title	Attaxx game
Name	Mr.Jirawat Tienpichetpong ID. 46360020
Project advisor	Dr.Suradet Jitprapaikulsarn
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic year	2007

Abstract

This project develops a game name "Attaxx" and a program for simulates an opponent for the game. These game and program are develop using Macromedia Flash, an animation create program which is popular for now. So this project can be divided into 2 parts the first is game development and the second is an Artificial Intelligence (AI) program for uses as opponent player in the game. My AI program can calculate and makes decision for the best way to win in a few seconds and my game can interactive with user with the animation.

กิตติกรรมประกาศ

โครงงานวิศวกรรมคอมพิวเตอร์สำเร็จได้ด้วยดี ก็เนื่องจากความอนุเคราะห์จากท่านอาจารย์ที่ปรึกษา ดร.สุรเดช จิตประไพกุลศาล ที่กรุณาให้คำปรึกษา แนะนำวิธีการในการทำงาน ตลอดถึงการตรวจสอบการทำงานพร้อมทั้งเสนอแนวทางการแก้ไขตลอดระยะเวลาการทำโครงงาน สุดท้ายต้องขอขอบพระคุณอาจารย์ทุกท่านและเพื่อนๆ ทุกคนที่ยังไม่ได้เอ่ยนามที่คอยให้คำแนะนำในด้านต่างๆ และสนับสนุนในการทำโครงงานครั้งนี้

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

สารบัญรูป(ต่อ)

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

การเพิ่มระดับของ IQ นั้นไม่ได้จำกัดเพียงการเรียนรู้จากตำราหรือจากอาจารย์เท่านั้น การเพิ่มระดับของสติปัญญานั้นยังสามารถทำได้อีกหลายวิธี ซึ่งวิธีหนึ่งที่ได้ผลดีเป็นอย่างยิ่งและเป็นที่นิยมก็คือการเล่นเกม ซึ่งเกมในปัจจุบันนี้ก็มีมากมายที่ต้องอาศัยความคิดทางตรรกะในการเล่น ทั้งยังมีความสนุกเพลิดเพลิน จึงนับเป็นวิธีที่ดีในการเพิ่มพูนระดับสติปัญญา ในที่นี้ก็ได้เลือกเกมที่จะนำมาศึกษาคือเกม Attaxx

Attaxx เป็นเกมแบบกระดาน (Board game) อาศัยผู้เล่นจำนวน 2 คน ผลัดกันเล่น โดยรูปแบบการเล่นก็คือ ผู้เล่นแต่ละคนจะมีตัวเดินสีของตนเอง เริ่มต้นบนกระดานลักษณะเป็นตารางสี่เหลี่ยมจัตุรัสขนาด 7x7 ช่อง และมีตัวเดินเริ่มให้ฝ่ายละ 2 ตัว วางในตำแหน่งมุมทั้ง 4 ของกระดาน โดยวางสลับสีกัน ดังรูป 1.1

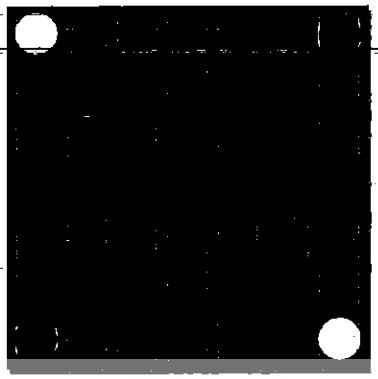
ในแต่ละตาผู้เล่นแต่ละคนมีสิทธิ์ในการเคลื่อนที่ตัวเดินของตนได้ 1 ครั้ง ซึ่งการเคลื่อนที่นั้นสามารถทำได้สองแบบ คือ Double และ Jump

- การเคลื่อนที่แบบ Double ก็คือการเคลื่อนตัวเดินไปในช่องที่ติดกันโดยการเพิ่มจำนวนตัวเดินอีก 1 ตัวในช่องที่เลือกแทน
- การเคลื่อนที่แบบ Jump ก็คือการเคลื่อนที่ตัวเดินไปในช่องที่ห่างจากตัวไป 2 ช่อง โดยการย้ายตัวเองไปในช่องที่เลือกแล้วลบตัวที่ช่องเดิมออก

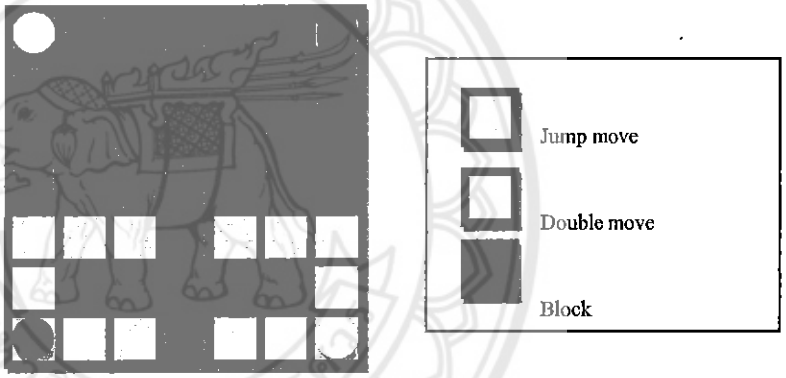
เมื่อตัวเดินของฝ่ายตนเองเคลื่อนที่ไปตกบริเวณช่องที่ติดกับตัวเดินของฝ่ายตรงข้ามก็จะเกิดการยึดครองตัวเดินของฝ่ายตรงข้ามในบริเวณที่ติดกับตัวเดินที่เราเดินทั้ง 8 ทิศ โดยการยึดครองนั้น เป็นการเปลี่ยนสีของตัวเดินฝ่ายตรงข้ามให้เป็นตัวเดินสีของเรา

เกมจะจบลงก็ต่อเมื่อบนกระดานไม่มีที่ว่างให้เดิน หรือเมื่อมีฝ่ายใดฝ่ายหนึ่งที่ไม่สามารถทำการเดินหมากได้ โดยฝ่ายที่มีตัวเดินในสีของตนมากกว่าก็จะเป็นฝ่ายที่ชนะ ซึ่งการที่ผู้เล่นจะชนะได้ก็ต้องอาศัยหลักการคิดวิเคราะห์ล่วงหน้าเพื่อหาจุดที่จะเดินแล้วได้เปรียบมากที่สุด

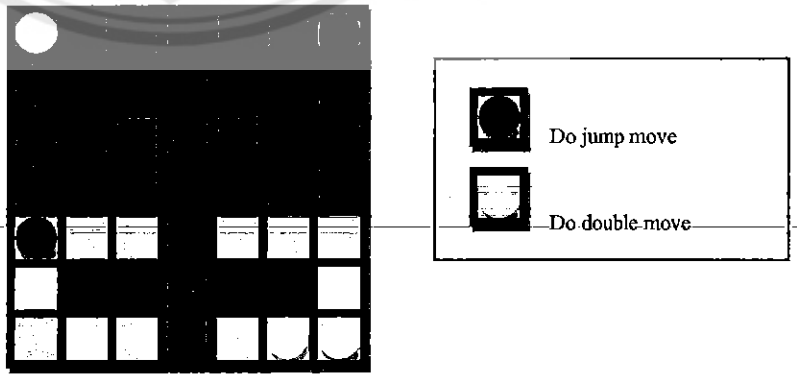
โครงงานนี้ได้สร้างเกม Attaxx โดยจะแสดงผลเป็น Graphic เพื่อให้มีความสวยงามและง่ายต่อการเล่นและนอกจากจะทำในส่วนของตัวเกมแล้วยังมีส่วนของ AI (Artificial Intelligence) หรือ Computer player ที่สามารถที่จะเล่นโต้ตอบกับผู้เล่นได้อย่างชาญฉลาดอีกด้วย



รูปที่ 1.1 แสดงตารางตอนเริ่มต้นเกม



รูปที่ 1.2 แสดงการเดินทั้งสองแบบ



รูปที่ 1.3 แสดงผลการเดินทั้งสองแบบ

1.2 วัตถุประสงค์โครงงาน

1. สามารถพัฒนาเกมเกม Attaxx ได้
2. สามารถพัฒนา AI (Computer player) ของเกม Attaxx ได้
3. ได้ศึกษาวิธีการสร้างเกมด้วยโปรแกรม Macromedia Flash ซึ่งกำลังเป็นที่นิยมในปัจจุบัน

1.3 ขอบข่ายการทำงาน

1. ศึกษากฎกติกาของเกม Attaxx
2. ศึกษาวิธีการเล่นเกม Attaxx
3. ศึกษาการเขียนโปรแกรม
4. สร้างและพัฒนาเกม Attaxx
5. สร้างและพัฒนา AI ของเกม Attaxx

1.4 ขั้นตอนดำเนินงาน

1. ศึกษาเกี่ยวกับทฤษฎีและหลักการต่างๆดังนี้
   - รูปแบบและกฎกติกาของการเล่นเกม Attaxx
   - วิธีการเล่นเกม Attaxx
   - การเขียนโปรแกรม
2. ออกแบบและพัฒนาโปรแกรม
3. ทดสอบโปรแกรม
4. ปรับปรุงและแก้ไขโปรแกรม
5. วิเคราะห์การทดสอบพร้อมทั้งสรุปผล
6. จัดทำเป็นรูปเล่ม

1.5 แผนการดำเนินงาน

กิจกรรม	ปี 2549					ปี 2550			
	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.
1. รูปแบบและกฎกติกาของการเล่นเกม Attaxx, วิธีการเล่นเกม Attaxx และการเขียนโปรแกรม	←	→							
2. ออกแบบและพัฒนาโปรแกรม	←						→		
3. ทดสอบโปรแกรม					←			→	
4. ปรับปรุงและแก้ไขโปรแกรม						←		→	
5. วิเคราะห์การทดสอบพร้อมทั้งสรุปผล							←	→	
6. จัดทำรูปเล่มโครงงาน								←	→

1.6 ผลที่คาดว่าจะได้รับ

1. เข้าใจรูปแบบและวิธีการเล่นของเกม Attaxx
2. สามารถสร้างและพัฒนาเกม Attaxx
3. สามารถนำความรู้ในเรื่องของ AI มาพัฒนา Computer player ของเกม Attaxx
4. สามารถนำความรู้ใน โปรแกรม Macromedia Flash ไปประยุกต์ใช้ในการทำงานได้

1.7 งบประมาณ

1.	ค่าวัสดุสำนักงาน	เป็นเงิน	500	บาท
2.	ค่าถ่ายเอกสาร	เป็นเงิน	500	บาท
		รวมเป็นเงินทั้งสิ้น	1,000	บาท (หนึ่งพันบาทถ้วน)

บทที่ 2
หลักการและทฤษฎี

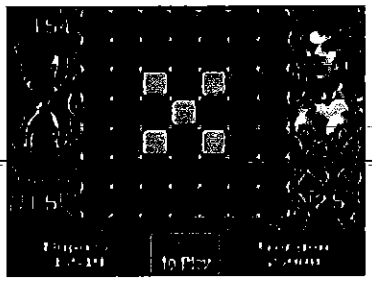
ในบทนี้จะกล่าวถึงประวัติความเป็นมาและกติกาการเล่นของเกม Attaxx นี้รวมถึงวิธีการที่ใช้ในการออกแบบและพัฒนาตัวเกม Attaxx รวมไปถึงการออกแบบและพัฒนา AI (Artificial Intelligent) หรือ Computer player ของเกมนี้ด้วย

2.1 ประวัติความเป็นมาของเกม Attaxx

Attaxx เป็นเกมที่เริ่มกำลังเป็นที่นิยมในกลุ่มของผู้ที่ชื่นชอบเกมแบบกระดาน หรือ Board game ซึ่งมีลักษณะการเล่นคล้ายคลึงกับเกม Reversi การเล่นเกมนี้ต้องอาศัยทักษะในการคิดวางแผน และคำนวณ เพื่อให้ได้เปรียบหรืออาจทำให้จบเกมได้โดยไม่ต้องเล่นจนจบกระดาน ซึ่งกลยุทธ์ ก็ขึ้นอยู่กับรูปแบบของกระดานด้วย เพราะในกระดานตอนเริ่มจะมีช่องที่เป็นสีดำทึบไม่สามารถเดินตัวหมากในช่องนี้ได้หรือเรียกว่า Block วางอยู่จำนวนหนึ่งเพื่อให้มีรูปแบบการวางแผนที่มากขึ้น เกม Attaxx นี้ปรากฏขึ้นครั้งแรกในปี ค.ศ. 1990 (พ.ศ. 2533) โดยบริษัท The Leland Corporation บนเครื่องเกม Arcade หรือเกมตู้ แต่ที่จริงผู้ที่สร้างเกมนี้ขึ้นคือ Dave Crummack และ Craig Galley ในปี ค.ศ. 1988 (พ.ศ. 2531) ตั้งชื่อว่า Infection ต่อมาก็ได้มีการดัดแปลงรูปแบบของกระดานให้แตกต่างกันออกไปมากมาย ซึ่งชื่อที่เรียกก็แตกต่างออกไปด้วยเช่นกัน

2.2 วิธีการเล่นเกม

เกมนี้จะอาศัยผู้เล่น 2 คน ผู้เล่นแต่ละคนจะผลัดกันเดินตัวหมากของตัวเองที่มีมาในตอนเริ่มเกมคนละ 2 ตัว ซึ่งการเดินจะมีด้วยกัน 2 แบบ คือ Double และ Jump อธิบายให้เข้าใจง่ายๆก็คือ การเดินแบบ Double จะเป็นการเพิ่มจำนวนตัวหมากของตนเองอีกหนึ่งตัวในตำแหน่งที่ติดกับตัวหมากเดิมในทิศทางใดก็ได้ใน 8 ทิศรอบตัว ส่วนการเดินแบบ Jump ก็คือการย้ายตัวเดินจากตำแหน่งเดิมไปยังตำแหน่งที่ห่างออกไป 2 ช่องไม่นับรวมช่องที่ตัวหมากที่จะเดินวางอยู่ โดยสามารถเดินได้ 16 ทิศทาง ในเกมจะมีการกินตัวหมากของฝ่ายตรงข้ามรอบโดยจะเปลี่ยนสีตัวหมากของฝ่ายตรงข้ามให้เป็นสีของเราใน 8 ทิศที่ติดกับตัวหมากที่ผู้เล่นเดินไป เกมจะจบลงก็ต่อเมื่อผู้เล่นคนใดคนหนึ่ง หรือผู้เล่นทั้งสองคนไม่สามารถเดินต่อไปได้ ผู้ที่มีตัวหมากในสีของตนเองมากที่สุดจะเป็นผู้ชนะ



รูปที่ 2.1 แสดงเกม Attaxx ในปี ค.ศ. 1990

รูปภาพจาก http://www.gaissa.com/images/Ataxx.bmp

2.3 หลักการทำงานของเกม

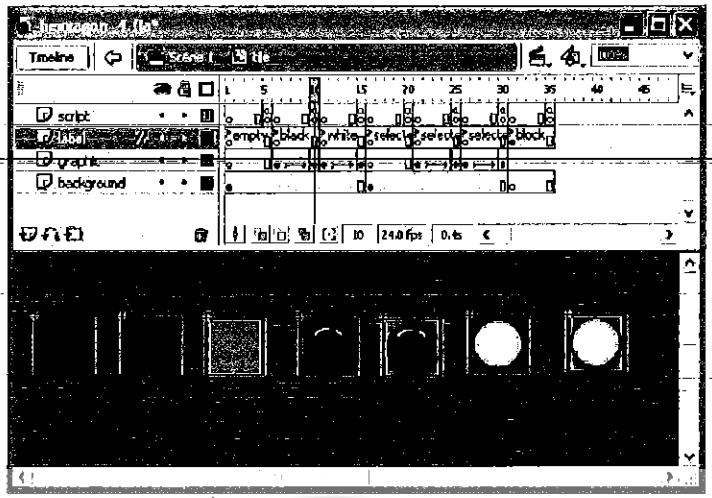
เกมนี้ถูกเขียนขึ้นด้วยโปรแกรม Macromedia Flash ซึ่งมีการทำงานคล้ายกับแบบ Object-Oriented ที่บอกว่าคล้ายก็เพราะส่วนหนึ่งจะเป็นการเขียนโปรแกรมโดยแยกเป็นวัตถุๆ ซึ่งแต่ละวัตถุก็จะมีหน้าที่ของมันเอง เป็นการเขียนโปรแกรมแบบ Object-oriented อีกส่วนหนึ่งจะเป็นการจัดการกับวัตถุและเหตุการณ์ที่สามารถเกิดกับวัตถุนั้นๆได้ นั่นคือมีการตอบสนองต่อการรับข้อมูลเข้ามาจากภายนอก หรือจากผู้ใช้นั่นเอง

เกมจะถูกสร้างขึ้นทีละ frame บน timeline หลัก ด้วยการประกอบขึ้นมาจาก Movie clip, graphic และ button หลายชิ้นงาน ซึ่งงานในการออกแบบเกมนี้ส่วนหนึ่งก็คือการออกแบบรูปลักษณ์ของเกมว่าต้องการเกมที่มีหน้าตาอย่างไร และสอง ก็คือการวางแผนว่าในแต่ละ frame ของเกมเรานั้นจะมีเหตุการณ์อะไรเกิดขึ้นบ้าง เพื่อจะได้ออกแบบและเขียนโปรแกรมเข้ามาจัดการกับเหตุการณ์ทุกๆ เหตุการณ์ หรือก็คือการวางสตอรี่บอร์ด (Storyboard) นั่นเอง

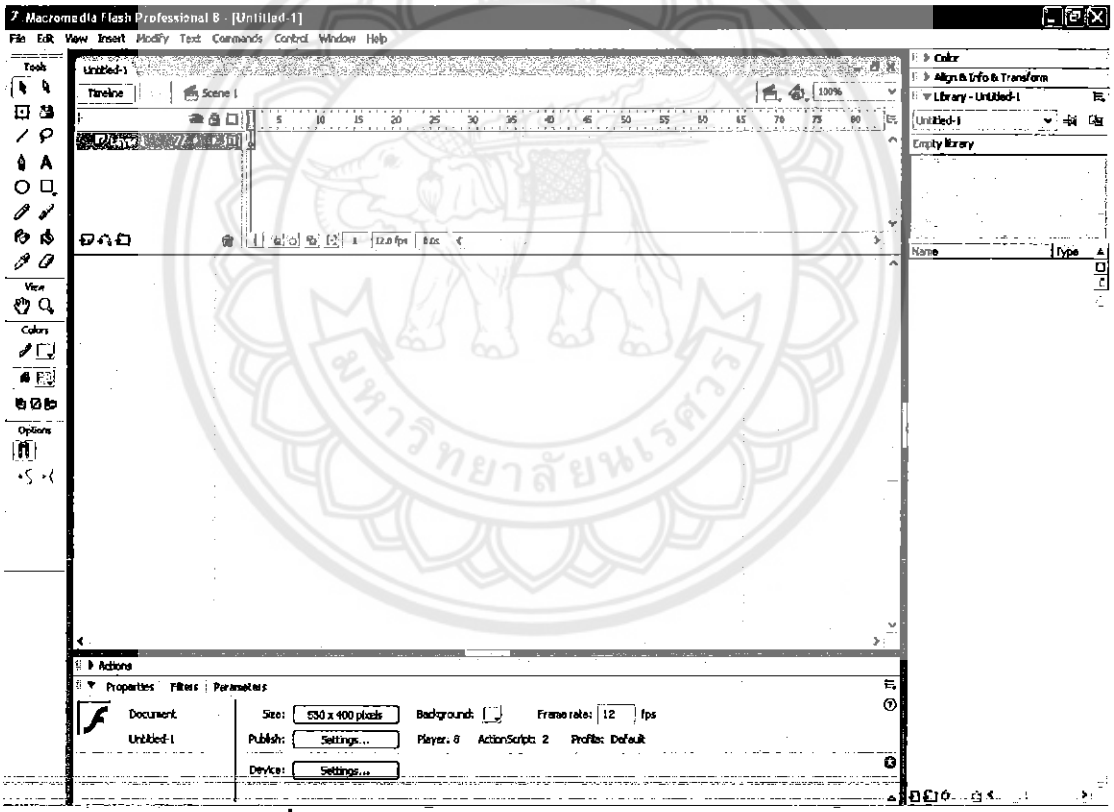
Movie clip เป็นอีกส่วนหนึ่งที่สำคัญในการสร้างเกมนี้ เพราะใน Movie clip แต่ละอันก็จะมี timeline ของตัวมันเองซึ่งทำงานแยกกันกับ timeline หลัก เมื่อเรานำ Movie clip ไปวางไว้บน timeline หลัก แล้วมีการเล่น (Play) บน timeline หลักจนถึง frame ที่เราวาง Movie clip ไว้ Movie clip นั้นก็จะเล่นเองโดยอัตโนมัติพร้อมๆกันกับ timeline หลัก

Event คือสิ่งที่เกิดขึ้นระหว่างที่ Movie ทำงานซึ่งมีผลทำให้คำสั่งใดๆที่ถูกเขียนไว้ ทำงาน และอาจทำให้เกิดการเปลี่ยนแปลงบางอย่างต่อการทำงานของ Movie เช่น เมื่อปุ่มถูกกด หรือเมื่อนำเมาส์เลื่อนไปอยู่บนปุ่ม ปุ่มจะเปลี่ยนสี เป็นต้น เหล่านี้เรียกว่า Action

ในส่วนของตัวเกมจะใช้ Movie clip ที่มีลักษณะเป็นช่อง 1 ช่องของกระดาน นำมาเรียงต่อกันให้เกิดเป็นกระดานขนาด 7x7 ซึ่งใน Movie clip ตัวนี้จะมีการตอบสนองต่อการคลิกเมาส์บน Movie clip และเมื่อมีการเลื่อนเมาส์ไปบนหรือเลื่อนออกจาก Movie clip โดยจะเป็นแบบนี้เหมือนกันทุก Movie clip บนกระดาน แล้วเขียนเงื่อนไขที่จะให้ทำงานลงไป



รูปที่ 2.2 แสดง Movie clip ชื่อ tile



รูปที่ 2.3 แสดงหน้าตาของโปรแกรม Macromedia Flash

2.4 หลักการทำงานของ AI (Artificial Intelligent)

2.4.1 วิธีการคำนวณคะแนนของ AI

AI จะคิดคำนวณแยกเป็น 2 กรณีตามชนิดการเดิน คือ

1. **กรณีเดินแบบ Double**

การเดินแบบนี้คือการเดินในช่องที่ติดกับตัวเดินที่จะทำการเดิน ซึ่งมีผลให้เกิดตัวเพิ่มอีกตัวหนึ่งหลังจากการเดิน ทำให้การเดินแบบนี้เป็นผลดีต่อการเล่นมากกว่าการเดินแบบ Jump จึงจำเป็นต้องให้ AI คำนวณในส่วนของความได้เปรียบเสียเปรียบนี้ด้วย ดังนั้น การเดินแบบ Double จะมีการเพิ่มค่าของคะแนนที่ได้ในแต่ละช่องโดยคำนวณจากสูตร

$$\text{Double score} = (\text{point} * 1.35) + 0.05$$

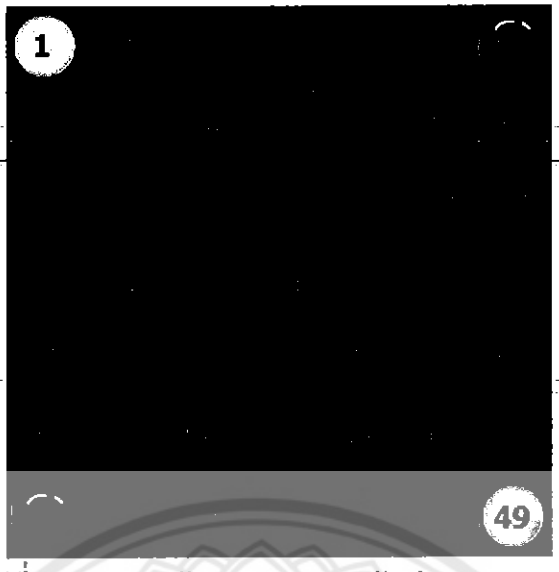
โดยค่า point คือจำนวนตัวที่กินได้หากเลือกช่องนั้น ส่วนสาเหตุที่เลือกใช้ค่าคงที่ 1.35 ก็มาจากการทดลองเล่นหลายๆครั้งและหาหนทางที่คิดว่าจะสามารถชนะเกมได้ จึงทราบว่าควรจะเลือกเดินแบบ Double ในกรณีที่ส่วนต่างของจำนวนตัวที่กินได้ของ Jump มากกว่าไม่เกิน 2 ตัว และได้ทดลองกับค่าต่างๆอีกหลายค่าที่ให้ผลต่างกันแบบเห็นได้ชัดเจน จนได้ผลสรุปว่าค่า 1.35 นี้สามารถทำให้ AI มีความฉลาดได้มากกว่าค่าอื่นๆที่ทดลอง

2. **กรณีเดินแบบ Jump**

การเดินแบบนี้นั้นเราก็จะให้คะแนนตามจำนวนที่กินได้ คือ 1 ตัวต่อ 1 แต้ม ในส่วนของการทดลองค่าต่างๆที่เป็นค่าคงที่นั้นจะอธิบายอีกครั้งในบทที่ 4

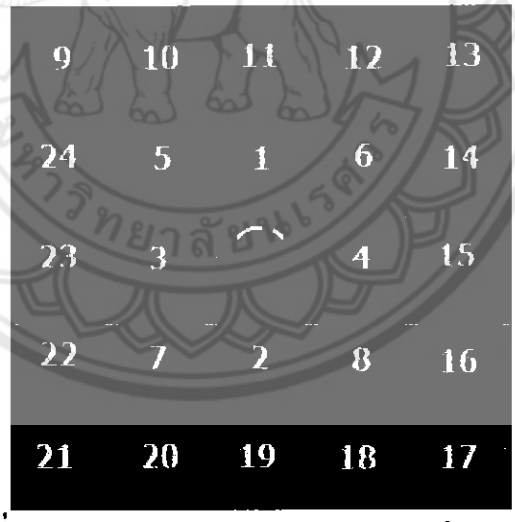
2.4.2. วิธีการตัดสินใจเลือกเดินของ AI

อันดับแรก AI จะทำการตรวจสอบทีละช่องว่าช่องใดเป็นตัวที่สามารถเลือกเดินได้หรือเป็นตัวของฝ่าย AI แล้วจึงทำการตรวจสอบว่าตัวนั้นๆสามารถเดินได้กี่ช่อง แล้วจึงคิดคะแนนว่าถ้าเดินในแต่ละช่องจะกินได้กี่คะแนน แล้วจึงตัดสินใจเลือกช่องที่เดินแล้วได้คะแนนสูงที่สุด



รูปที่ 2.4 แสดงลำดับการตรวจสอบตัวเล่นบนกระดาน

เมื่อตรวจเจอตำแหน่งที่เป็นตัวเล่นของฝ่าย AI ก็จะทำการหาตำแหน่งที่ตัวนั้นสามารถเดินได้ ตามลำดับในรูปที่ 2.5



รูปที่ 2.5 แสดงลำดับการตรวจสอบคะแนนในการเดิน

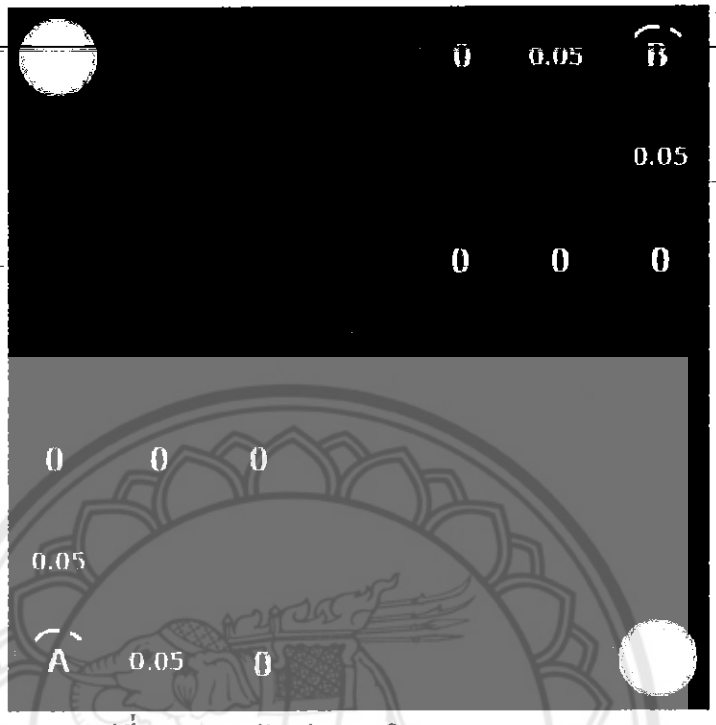
เมื่อตรวจสอบตั้งแต่ช่องที่ 1 ไปจนถึงช่องที่ 24 ว่าช่องใดสามารถเดินได้ ไม่มีตัวของฝ่ายตรงข้ามหรือฝ่ายเราวางอยู่ ก็จะทำการคำนวณหาคะแนนที่สามารถกินได้จาก ชนิดของการเดิน และจำนวนตัวที่สามารถกินได้หากเดินช่องนั้น จากนั้นเก็บค่าตำแหน่งของช่องที่เดินแล้วทำให้กินได้คะแนนมากที่สุดไว้ แล้วจึงเลื่อนไปทำแบบเดียวกันนี้กับช่องถัดไปจนครบ

เมื่อตรวจสอบและคำนวณคะแนนของตัวนี้ครบทุกช่องแล้ว ก็เปลี่ยนไปหาตัวหมากฝ่าย AI ตัวอื่นบนกระดานเพื่อหาตาเดินที่ได้คะแนนมากที่สุดต่อไป และเลือกเดินในช่องที่ทำให้กินได้มากที่สุดและได้เปรียบมากที่สุดด้วย

แต่จะมีในกรณีที่ไม่มีการกินกันเกิดขึ้น และกรณีที่กินได้เท่ากัน ที่จะต่างออกไป โดยในกรณีที่ไม่มีการกินกันเกิดขึ้น point ก็จะเป็น 0 จึงทำให้คะแนนที่คำนวณมาได้เป็น 0 ในแบบ Jump ส่วนในแบบ Double จะเป็น 0.05 เท่ากันทุกช่อง แต่ด้วยลำดับการตรวจสอบคะแนน AI จะเก็บคะแนนเฉพาะที่มากกว่าหรือเท่ากับคะแนนปัจจุบัน จึงทำให้ AI เก็บค่าของช่องสุดท้ายที่ตรวจสอบไว้ และเลือกเดินช่องนั้นถ้ามีคะแนนเท่ากัน

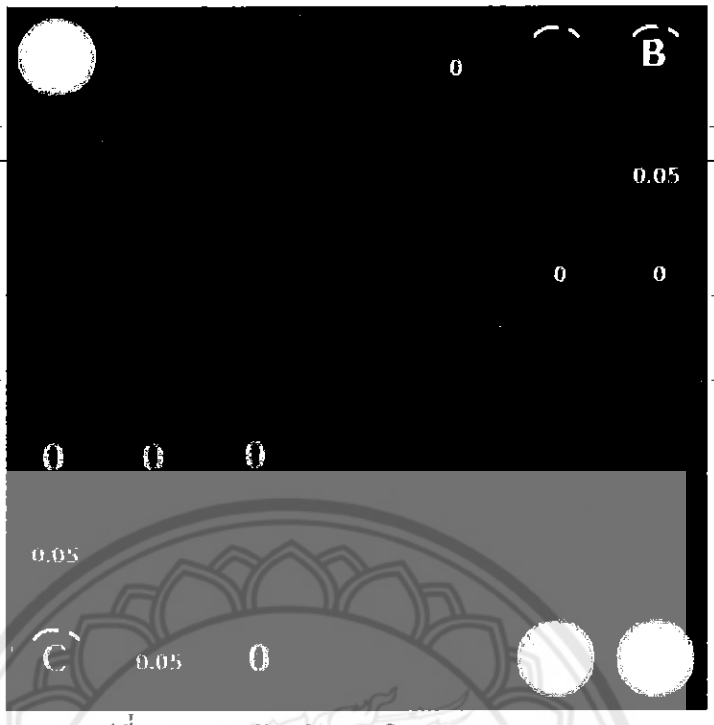
ส่วนในกรณีที่มีการกินได้เท่ากันนั้นก็จะคล้ายกับกรณีที่ไม่มีการกินกัน AI ก็จะทำการเลือกเดินในช่องที่มีคะแนนเท่ากันในลำดับการตรวจสอบที่สุดท้าย เช่น ถ้าช่องลำดับที่ 7 และ 12 มีคะแนนที่กินได้เท่ากัน AI จะเลือกเดินช่องที่ 12 เพราะการตรวจสอบของช่องที่ 12 เกิดขึ้นเป็นลำดับสุดท้ายในกลุ่มของช่องที่คะแนนเท่ากัน

ตัวอย่างการเดินของ AI ในกรณีที่ไม่มีการกินกันเกิดขึ้น



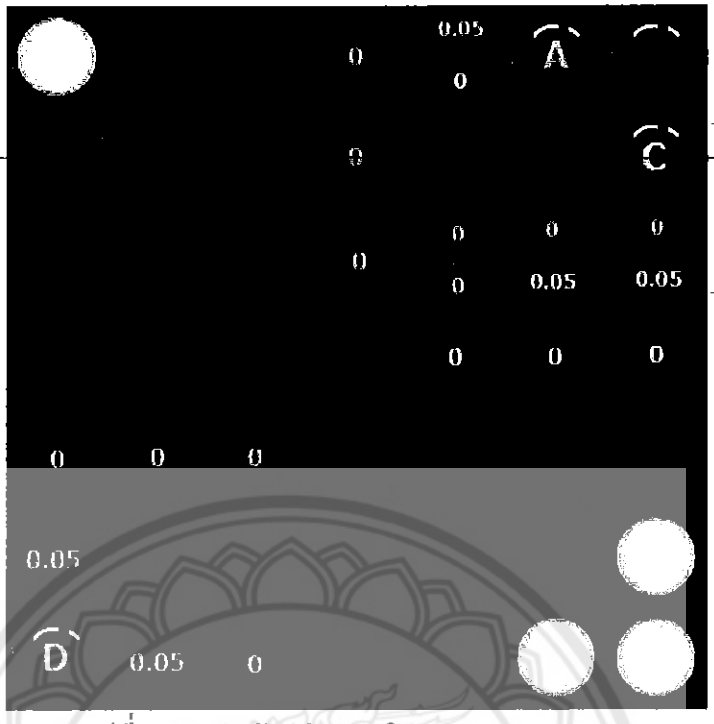
รูปที่ 2.6 แสดงตัวอย่างการคิดคะแนนของ AI (1)

จากรูปที่ 2.6 AI จะคิดคะแนนของ A ก่อน แล้วจึงคิดคะแนนของ B ดังนั้นเมื่อไม่มีการกินกันเกิดขึ้น AI จะเลือกค่า 0.05 เพื่อเดิน แต่มีค่า 0.05 ทั้งหมด 4 ค่า AI ก็จะเลือกค่าของตัวที่คิดเป็นตัวสุดท้าย นั่นคือ B ตามลำดับในรูปที่ 2.4 แต่ที่ B ยังมีค่า 0.05 อีก 2 ค่า AI ก็จะเลือกจากค่าแรกของการคำนวณตามลำดับในรูปที่ 2.5 นั่นคือ ค่า 0.05 ที่อยู่ทางซ้ายของ B นั่นเอง



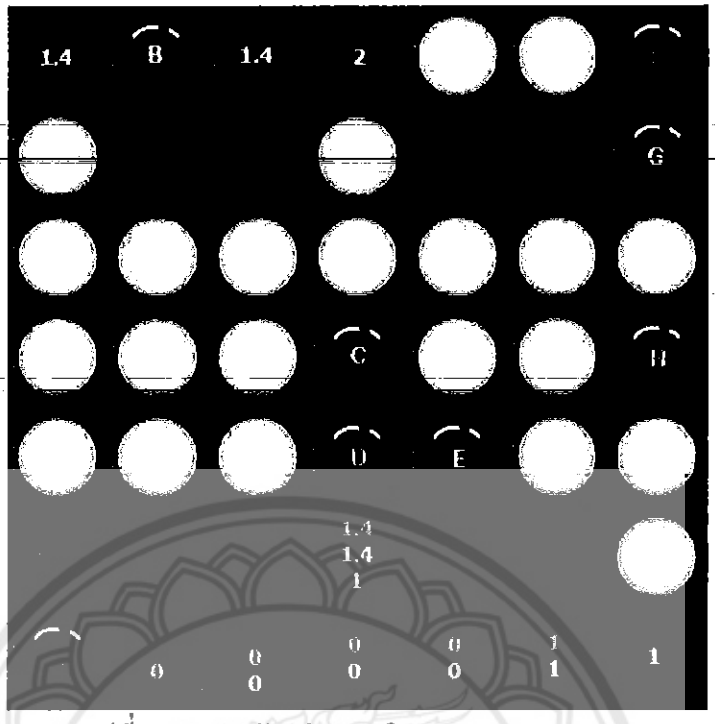
รูปที่ 2.7 แสดงตัวอย่างการคิดคะแนนของ AI (2)

จากรูปที่ 2.7 เป็นผลการเดินของรูปที่ 2.6 ในรูปนี้ AI จะมีลำดับในการคิดคะแนนคือ C -> A -> B เมื่อคิดคะแนนครบแล้วจะ ได้ว่า AI เลือกค่า 0.05 ซึ่งมีค่าซ้ำกันอีก 5 ตัว แต่ด้วยลำดับการคิดคะแนน AI จะเลือกค่าของตัวที่คิดคะแนนตัวสุดท้ายคือ B จึงเหลือค่าเดียวให้เลือกคือช่องที่อยู่ล่างของ B ดังนั้น AI จึงเลือกเดินในช่องนั้น



รูปที่ 2.8 แสดงตัวอย่างการคิดคะแนนของ AI (3)

หลังจาก AI ได้เลือกเดินแล้วจากรูปที่ 2.7 ก็จะได้ดังรูปที่ 2.8 ด้วยหลักการคิดคะแนนเหมือนที่ผ่านมาที่ทุกช่องไม่มีการกินกันเกิดขึ้น ก็จะเลือกค่า 0.05 ในช่อง ล่างซ้ายของ C เนื่องจากลำดับในการคิดคะแนน และลำดับในการเลือกของ C ตามรูปที่ 2.5



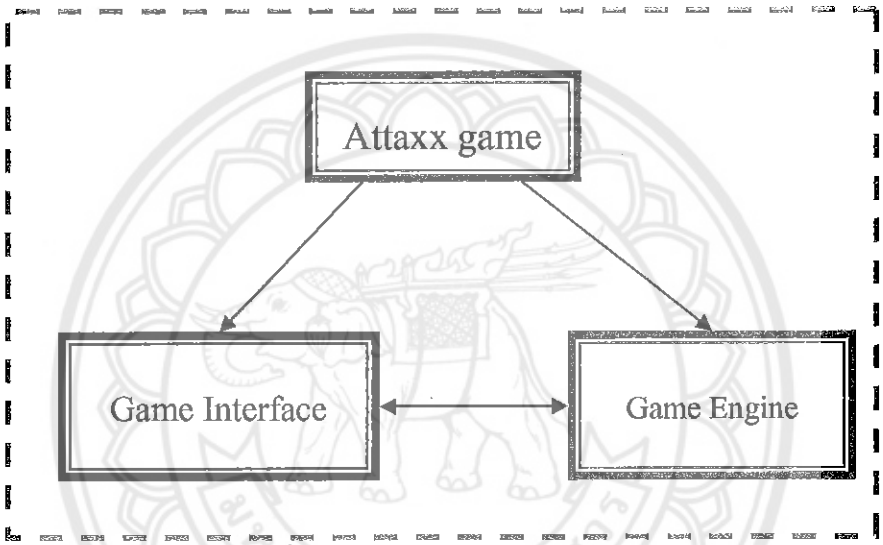
รูปที่ 2.9 แสดงตัวอย่างการคิดคะแนนของ AI (4)

จากรูปที่ 2.9 นี้ เป็นตาเดินของ AI ดังนั้น AI จะทำการคิดคะแนนโดยมีลำดับจาก A -> B -> C -> D -> E -> F -> G -> H และดังคะแนนที่แสดงไว้ในรูปด้วยสีที่ต่างกัน AI จะตัดสินใจเลือกคะแนน 2.75 ซึ่งเป็นการเดินแบบ Double ที่กินได้ 2 ตัวแทนที่จะเลือกเดินแบบ Jump จากตัว B ที่กินได้ 2 ตัวเท่ากัน แต่มีเพียง 2 คะแนน

บทที่ 3
วิธีการดำเนินการ

เนื้อหาในบทนี้จะกล่าวถึงขั้นตอนและวิธีการดำเนินการต่างๆเพื่อสร้าง Attaxx game ดังนี้คือ

3.1 Architectures of the game.



รูปที่ 3.1 แสดง System Architecture

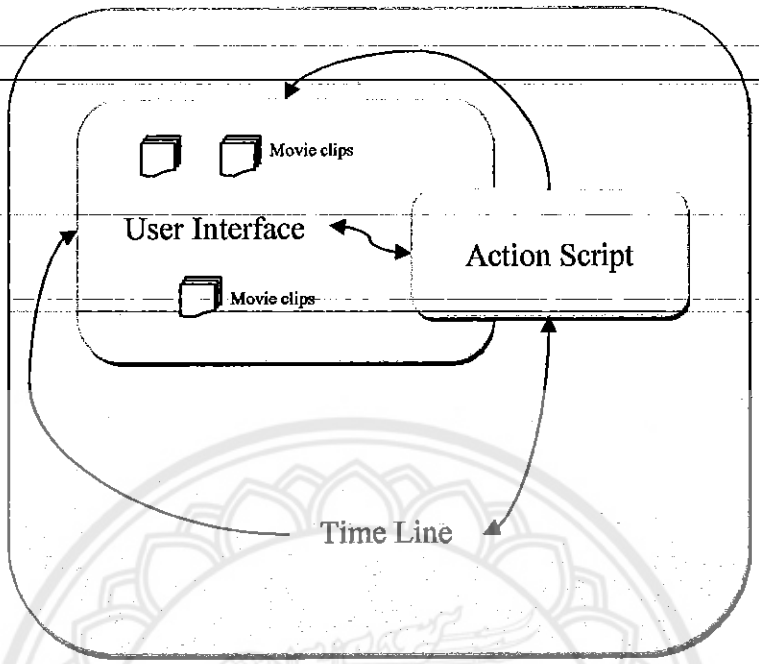
Attaxx game มีโครงสร้างหลักอยู่สองส่วน คือส่วนของ Game Interface และส่วนของ AI

Game Interface เป็นส่วนที่ทำหน้าที่แสดงผลและติดต่อระหว่างผู้เล่นกับ AI โดยในส่วนนี้ประกอบไปด้วยส่วนย่อยอีกคือ

1. User Interface (UI) เป็นส่วนที่สร้างด้วยการใช้โปรแกรมวาดขึ้นเป็นส่วนใหญ่ เพื่อติดต่อระหว่างผู้เล่นกับ AI โดยการรับค่าต่างๆจากผู้เล่น ผ่านทาง Mouse หรือ Keyboard
2. Action Script (AS) เป็นส่วนที่เป็นการเขียนโปรแกรมเพื่อกำกับการทำงานของ UI อีกที ทั้งยังเป็นการกำกับลำดับขั้นตอนการแสดงผลของโปรแกรมด้วย AS ในส่วนของ Game Interface นี้จะเป็นการเขียนโปรแกรมที่ไม่ซับซ้อนเท่าในส่วนของ AI

Game Engine เป็นส่วนที่ทำหน้าที่คิดคำนวณและเลือกคำตอบที่ดีที่สุด เพื่อตอบโต้กับผู้เล่น ในส่วนนี้กีจะเป็นการเขียน AS ที่ซับซ้อนยิ่งขึ้น และมีฟังก์ชันย่อยอีกมากมายที่ทำหน้าที่ต่างๆ ซึ่งจะอธิบายโดยละเอียดต่อไป

3.2 Game Interface



รูปที่3.2 แสดงความสัมพันธ์ระหว่างแต่ละโครงสร้างของ Game Interface

3.2.1 User Interface (UI)

UI เป็นวัตถุที่เราสร้างขึ้นในที่นี้ขอเรียกว่า Movie clip ซึ่งทำงานเมื่อถึงเวลาที่กำหนดหรือเมื่อถูกสั่งการให้ทำงานผ่าน AS โดยในหนึ่งเกมจะประกอบด้วย Movie clip จำนวนมากมาประกอบกันเป็นเกมๆหนึ่ง เช่น Movie clip ของตารางที่ใช้ประกอบกันเป็นกระดาน หรือ Movie clip ของ AI ที่ใช้เล่นตอบโต้กับผู้เล่น เป็นต้น ซึ่ง Movie clip เหล่านี้จะถูกจัดวางไว้บน Time line ที่เปรียบเสมือนพื้นที่ที่เราสามารถวางวัตถุลงไปได้ และจะมีลำดับการทำงานอยู่บน Time line ด้วย ถ้าเราต้องการให้ Movie clip ตัวใดทำงานเวลาใดก็เพียงนำ Movie clip ตัวนั้นไปวางไว้บนช่วงเวลาที่เราต้องการ บน Time line และเมื่อถึงเวลาที่กำหนด มันก็จะทำงานโดยอัตโนมัติ

3.2.2 Action Script (AS)

AS เป็นภาษาในการเขียนโปรแกรมของ Flash ซึ่งมีลักษณะคล้ายกับภาษา C++ และ Java แตกต่างกันตรงที่ AS จะเขียนไว้ที่ Movie clip หรือ Time line และจะทำงานเมื่อ Movie clip ทำงานหรือหากเขียนไว้บน Time line ก็จะทำงานเมื่อถึงเวลาที่กำหนดไว้ ซึ่ง AS ที่เขียนไว้ในส่วนของ Game Interface นี้ส่วนใหญ่จะเขียนด้วยคำสั่งง่ายๆ เพื่อทำการควบคุมลำดับการทำงานของ Movie clip และ Time line เช่น สั่งให้ข้ามไปเล่นที่เวลาใดบน Time line หรือสั่งให้ Movie clip ตัวใดทำงาน เป็นต้น

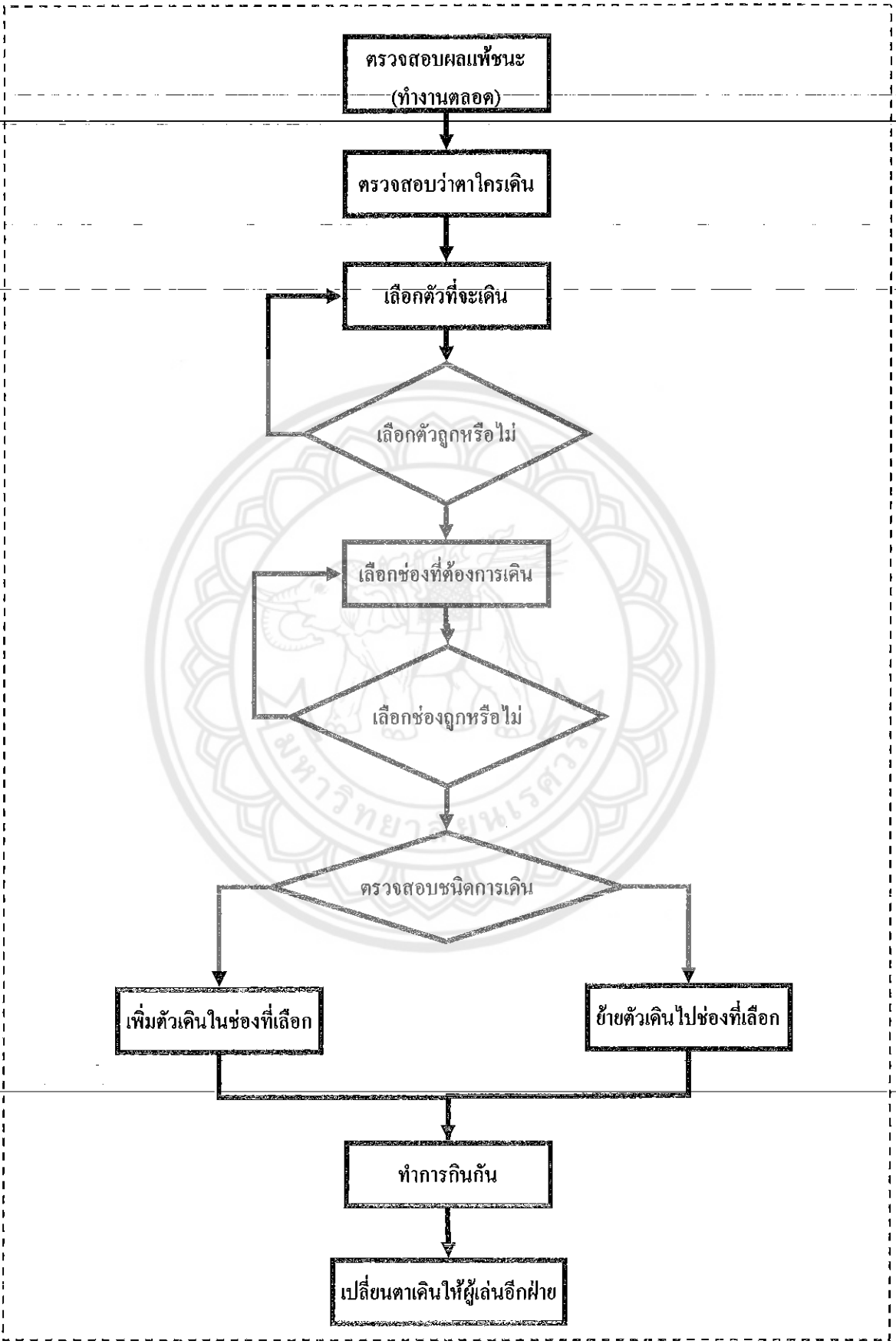
3.2.3 หลักการทำงานของส่วน Game Interface

การทำงานโต้ตอบระหว่างโปรแกรมกับผู้เล่นสามารถเขียนอธิบายตามหลักตรรกะได้ดังนี้

0. เริ่มต้นโปรแกรมจะตรวจสอบผลแพ้ชนะ (หน้าที่นี้จะทำงานตลอดเวลาที่เล่นเกม)
1. ผู้เล่นเลือกตัวเดิน
2. โปรแกรมตรวจสอบว่าเลือกถูกต้องหรือไม่
    a. ถ้าถูก โปรแกรมตรวจสอบช่องที่ตัวนั้นสามารถเดินไปได้
    b. ถ้าไม่ถูก ไม่เกิดอะไรขึ้น และให้ผู้เล่นเลือกตัวอื่น
3. ผู้เล่นเลือกช่องที่ต้องการจะเดินไป
4. โปรแกรมตรวจสอบว่าช่องที่เลือกสามารถเดินไปได้หรือไม่
    a. ถ้าได้ โปรแกรมตรวจสอบชนิดของการเดิน
    b. ถ้าไม่ได้ ไม่เกิดอะไรขึ้น ให้ผู้เล่นเลือกช่องที่ต้องการเดินใหม่
5. หลังจากตรวจสอบว่าเป็นการเดินชนิดใด
    a. ถ้าเป็นแบบ Double เพิ่มตัวเดินของผู้เล่นในช่องที่เลือก
    b. ถ้าเป็นแบบ Jump ลบตัวเดินของผู้เล่นในช่องที่เดินมาและเพิ่มในช่องที่เลือก
6. โปรแกรมตรวจสอบผลการกินกันบริเวณรอบตัวที่เดิน และดำเนินการกินกัน
7. เปลี่ยนตาเดินไปให้อีกฝ่าย

เมื่อเปลี่ยนตาเดินไปให้อีกฝ่าย ก็จะเริ่มกระบวนการทำซ้ำเดิมตั้งแต่ข้อ 1 ถึง ข้อ 7 อีกครั้ง แล้วในระหว่างที่ผ่านทุกขั้นตอนนั้น โปรแกรมที่ทำหน้าที่คอยตรวจสอบผลการแพ้ชนะก็จะทำการตรวจสอบอยู่ตลอดเวลา หากเมื่อใดเกิดผลการแพ้ชนะขึ้น เช่น เมื่อฝ่ายใดฝ่ายหนึ่งไม่สามารถเดินต่อไปได้ หรือทั้งสองฝ่ายไม่มีตาเดินเหลืออยู่ ก็จะหยุดเกมลงและประกาศตัวผู้ชนะทันที

จากที่ได้อธิบายมาเป็นเชิงตรรกะ เราสามารถนำความคิดนี้มาเขียนเป็น Flowchart เพื่อให้เข้าใจได้ง่ายขึ้น ดังต่อไปนี้



รูปที่ 3.3 Flowchart แสดงหลักการทำงานของเกม

3.3 Game Engine

ในส่วนนี้จะเป็นการเขียนโปรแกรม AS ที่ซับซ้อน เพื่อควบคุม คำนวณ และตอบโต้กับผู้เล่นผ่านในส่วนของ Game Interface ในส่วนของ AI เองประกอบไปด้วย function มากมายที่ทำหน้าที่แต่ละอย่างตามที่ได้โปรแกรมไว้ และจะทำงานตอบสนองต่อ Event ซึ่งจะรับเข้ามาจากผู้เล่นผ่านทาง mouse หรือ keyboard

3.3.1 Event

คือเหตุการณ์ที่สามารถเกิดขึ้นได้กับโปรแกรมผ่านการกระทำของผู้เล่น Event มีด้วยกันหลายอย่าง ดังนี้

1. onPress ตอบสนองต่อการคลิกเมาส์
2. onRelease ตอบสนองต่อการคลิกและปล่อยเมาส์
3. onRollOver ตอบสนองต่อการเลื่อกเมาส์ไปอยู่บนวัตถุ
4. onRollOut ตอบสนองต่อการเลื่อนเมาส์ออกจากวัตถุ
5. onKeyPress ตอบสนองต่อการกดปุ่มบนคีย์บอร์ด

3.3.2 Function

คือ AS ที่เขียนโดยอาศัยเงื่อนไขของ Event เพื่อกำหนดการทำงานต่างๆ function ต่างๆที่ใช้มีดังนี้

1. function setTile ทำหน้าที่ในการพลิกตัวหมากในการกินกัน
2. function result ทำหน้าที่ตรวจสอบผลการแพ้ชนะซึ่งจะทำงานตลอดเวลาที่เล่นเกม
3. function scoreAI ทำหน้าที่ตรวจสอบคะแนนที่กินได้ในแต่ละช่องที่สามารถเดินได้เพื่อนำไปเปรียบเทียบระหว่างแต่ละช่องว่าช่องใดสมควรเดินมากที่สุด
4. function checkTile ทำหน้าที่ตรวจสอบว่าช่องใดบ้างที่สามารถเดินได้
5. function runAI ทำหน้าที่เป็นตัวหลักในการเรียกใช้ function อื่นเป็นตัวจัดลำดับว่าจะเรียกใช้ function ใดทำงานในลำดับใด

จากนั้นเมื่อโปรแกรมคำนวณและตัดสินใจได้แล้วว่าตัวเลือกใดเหมาะสมที่สุดในการเดินก็จะทำการเลือกเดินในช่องนั้นและทำการกินกันจากนั้นก็เปลี่ยนตาเดินไปให้อีกฝ่าย ทั้งนี้จะต้องตรวจสอบแล้วว่ายังไม่มีผลแพ้ชนะเกิดขึ้น

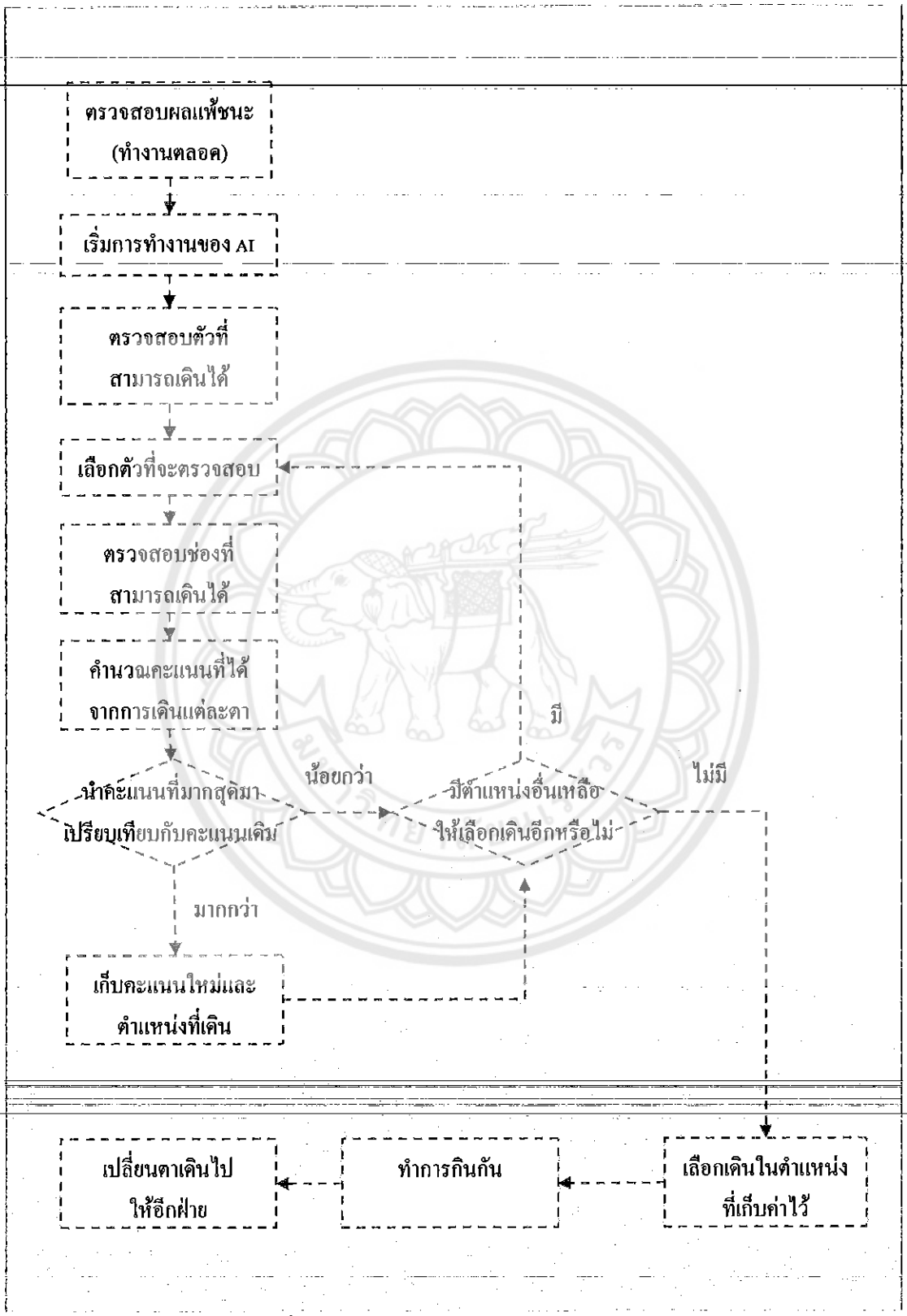
3.3.3 Algorithm of AI

AI คือชุดคำสั่งที่เขียนขึ้นให้มีการคำนวณเพื่อตอบสนองต่อผู้เล่น การทำงานของ AI สามารถเขียนเป็นตรรกะได้คือ

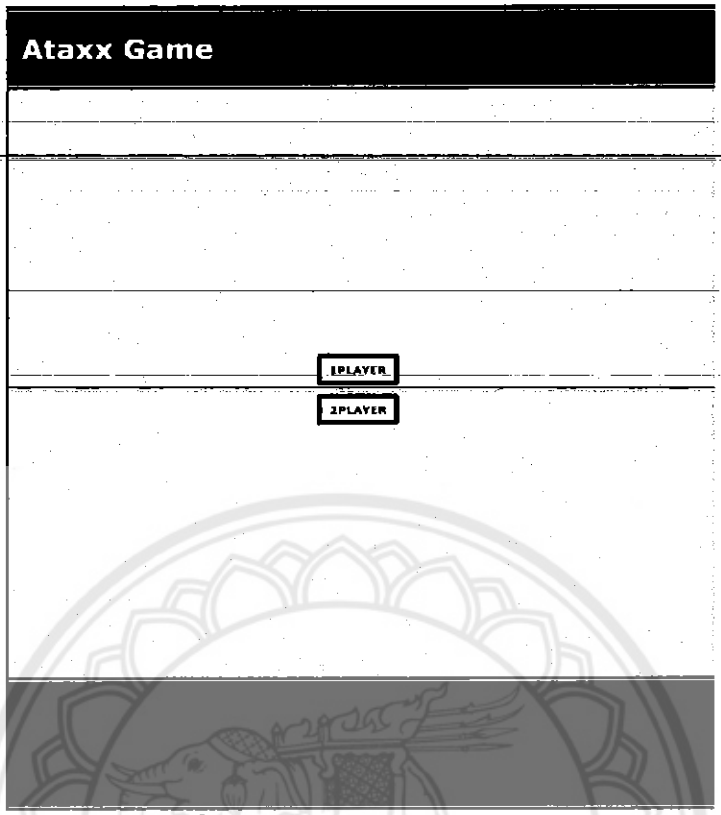
0. ตรวจสอบผลแพ้ชนะ (ทำงานตลอด)
1. เริ่มการทำงานของ AI
2. ตรวจสอบช่องที่มีตัวเดินของ AI ว่ามีช่องใดบ้าง
3. เลือกตัวที่จะตรวจสอบ
4. ตรวจสอบตาเดินที่ตัวนี้สามารถเดินได้
5. คำนวณคะแนนที่ได้จากการเดินในแต่ละช่องที่สามารถเดินได้
6. บันทึกคะแนนที่มากที่สุดที่ตัวนี้สามารถเดินได้ พร้อมตำแหน่งที่เดิน
7. เปรียบเทียบคะแนนที่ได้กับคะแนนที่สูงที่สุดที่เก็บไว้ จากตัวเดินทั้งกระดาน
8. คะแนนใหม่ที่ได้มากกว่าคะแนนเก่าหรือไม่
    a. ถ้ามากกว่า บันทึกคะแนนใหม่แทนที่คะแนนเดิม และบันทึกตำแหน่งที่เดิน
    b. ถ้าน้อยกว่า ทิ้งคะแนนใหม่ และตำแหน่งเดินใหม่
9. มีตัวอื่นให้ตรวจสอบอีกหรือไม่
10. ถ้ามี ทำซ้ำตั้งแต่ข้อ 3
11. ถ้าไม่มี เลือกเดินในตัวที่สามารถทำคะแนนได้มากที่สุด
12. โปรแกรมทำการกินกัน
13. เปลี่ยนตาเดินไปให้ผู้เล่น

ทั้งหมดนี้เป็นการทำงานเฉพาะของ AI ซึ่งอยู่ภายใต้เงื่อนไขการทำงานของโปรแกรมหลัก รวมถึงโปรแกรมที่ตรวจสอบการจบเกมด้วย ฉะนั้นเมื่อมีการกินกันแล้วเกิดผลแพ้ชนะขึ้น ก็จะจบเกมเช่นกัน

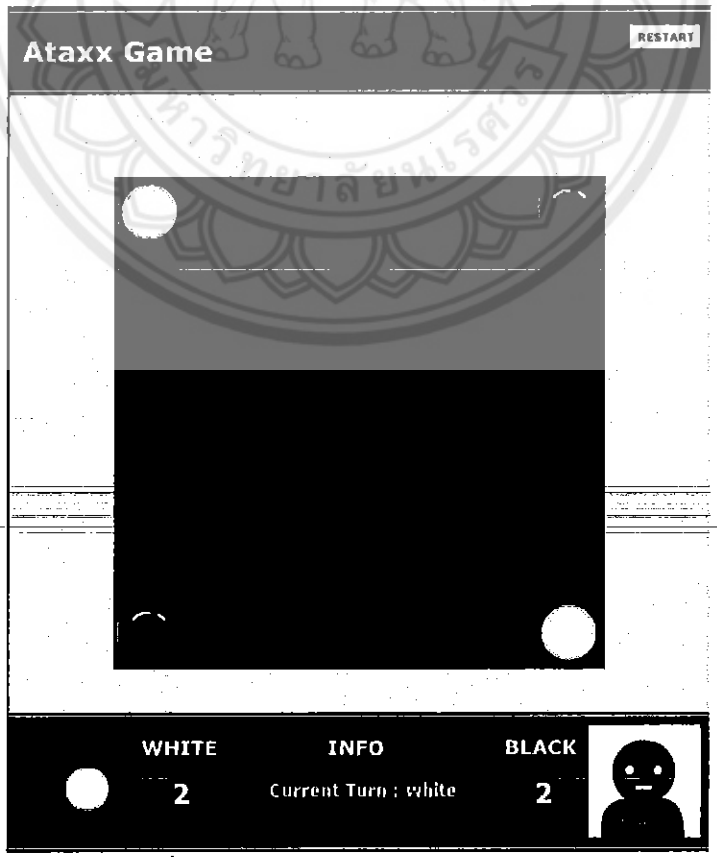
จากลักษณะขั้นตอนการทำงานที่ได้อธิบายข้างต้น นำมาเขียนอธิบายในรูปแบบของ Flowchart ได้ดังต่อไปนี้



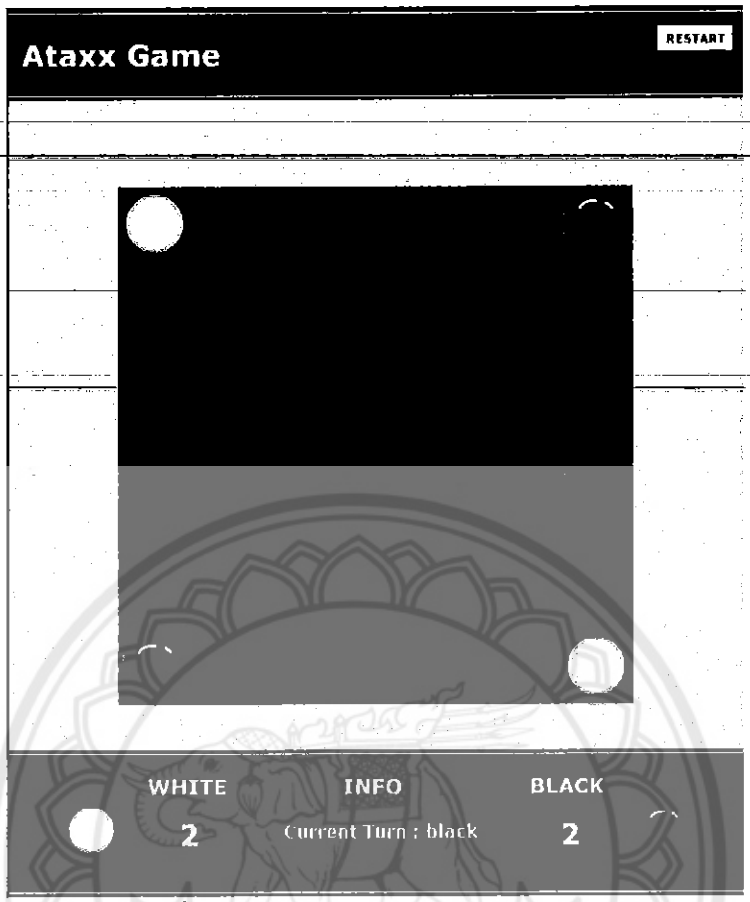
รูปที่ 3.4 Flowchart แสดงหลักการทำงานของ AI



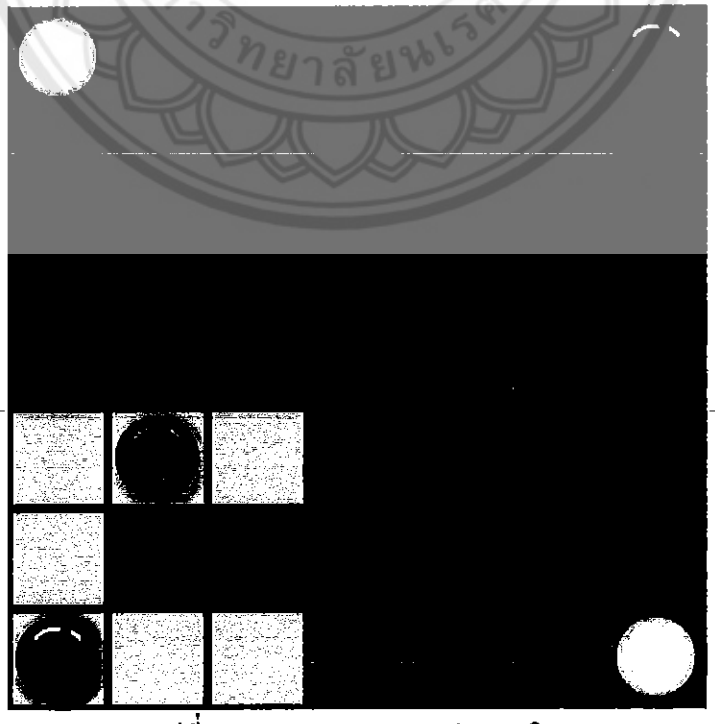
รูปที่ 3.5 หน้าแรกของเกมที่เสร็จแล้ว



รูปที่ 3.6 หน้าตาของเกมในโหมด 1Player



รูปที่ 3.7 หน้าตาของเกมในโหมด 2Players



รูปที่ 3.8 ภาพของเกมขณะทำการเดิน

บทที่ 4

ผลการทดลอง

4.1 การเลือกค่าที่เหมาะสม

จากที่ได้ทำการสร้างเกมพร้อมทั้ง AI ไป จึงจำเป็นต้องมีการทำสอบการทำงานของ AI และระดับความฉลาดของ AI โดยให้ผู้เล่นอาสาสมัครทดลองเล่นเกม ซึ่งตัวเกมนั้นจะแตกต่างกันไปตามค่าคงที่ที่ต้องการทดสอบ มีทั้งหมด 6 ค่า คือ 1.5, 1.35, 1.26, 1.21, 1.17 และ 1.15 สาเหตุที่เลือกใช้ค่าคงที่เหล่านี้ในการทดสอบก็เพราะให้ผลแตกต่างในคะแนนที่ให้อย่างชัดเจน ทำให้มีผลต่อการตัดสินใจเลือกตาเดินของ AI ซึ่งค่าเหล่านี้จะถูกนำไปคูณเข้ากับคะแนนการเดินของแบบ Double เพื่อเพิ่มคะแนนความได้เปรียบในการเดินให้ ทำให้เกิดความสมดุลในการตัดสินใจของ AI มากขึ้น

4.1.1. Weight 1.5

ความแตกต่างของการตัดสินใจของ AI เมื่อเปลี่ยนค่าคงที่ต่างๆให้กับค่า weight ที่นำไปคูณกับคะแนนการเดินแบบ Double นั้นจะแสดงให้เห็นตามตาราง

แสดงการเลือกชนิดของการเดิน

Weight 1.5		จำนวนตัวที่กินได้ด้วยการเดินแบบ Jump								
		0	1	2	3	4	5	6	7	8
จำนวนตัวที่กินได้จากการเดินแบบ Double	0	D	J	J	J	J	J	J	J	J
	1	D	D	J	J	J	J	J	J	J
	2	D	D	D	D	J	J	J	J	J
	3	D	D	D	D	D	J	J	J	J
	4	D	D	D	D	D	D	D	J	J
	5	D	D	D	D	D	D	D	D	J
	6	D	D	D	D	D	D	D	D	D
	7	D	D	D	D	D	D	D	D	D
	8	D	D	D	D	D	D	D	D	D

D คือการเลือกเดินแบบ Double
J คือการเลือกเดินแบบ Jump

ตารางที่ 4.1 แสดงการเลือกเดินที่ Weight 1.5

จากตารางที่ 4.1 จะเทียบถึงจำนวนตัวที่กินได้ของแบบ Double และแบบ Jump ซึ่งด้านบนจะเป็นจำนวนตัวที่กินได้ของแบบ Jump ส่วนด้านซ้ายจะเป็นจำนวนตัวที่กินได้ของแบบ Double

สัญลักษณ์ตัว D หมายถึงการเลือกเดินแบบ Double ตัว J หมายถึงการเลือกเดินแบบ Jump อ่านเทียบกัน เช่น ที่การเดินแบบ Double กินได้ 5 ตัว ต่อแบบ Jump กินได้ 6 ตัว AI จะเลือกเดินแบบ Double

จากตารางที่ 4.1 จะเห็นว่าที่จำนวนการกินเท่ากันของ 2 ประเภทการเดิน AI จะเลือกเดินแบบ Double และทุกๆการกินที่เพิ่มขึ้น 2 ตัว จะทำให้การเลือกเดินแบบ Double เพิ่มขึ้น เช่น เลือกกินแบบ Double 2 ตัวมากกว่าแบบ Jump 3 ตัว หรือ เลือกกินแบบ Double 4 ตัว มากกว่ากินแบบ Jump 6 ตัว เป็นต้น

4.1.2. Weight 1.35

แสดงการเลือกชนิดของการเดิน

Weight 1.35		จำนวนตัวที่กินได้ด้วยการเดินแบบ Jump								
		0	1	2	3	4	5	6	7	8
จำนวนตัวที่กินได้จากการเดินแบบ Double	0	D	J	J	J	J	J	J	J	J
	1	D	D	J	J	J	J	J	J	J
	2	D	D	D	J	J	J	J	J	J
	3	D	D	D	D	D	J	J	J	J
	4	D	D	D	D	D	D	J	J	J
	5	D	D	D	D	D	D	D	J	J
	6	D	D	D	D	D	D	D	D	D
	7	D	D	D	D	D	D	D	D	D
	8	D	D	D	D	D	D	D	D	D

D คือการเลือกเดินแบบ Double
J คือการเลือกเดินแบบ Jump

ตารางที่ 4.2 แสดงการเลือกเดินที่ Weight 1.35

จากตารางที่ 4.2 เช่นเดียวกับ ตารางที่ 4.1 เพียงแต่เปลี่ยน Weight เป็น 1.35 ทำให้ ทุกๆการกินที่เพิ่มขึ้น 3 ตัว การเลือกเดินแบบ Double จะมากขึ้น และค่านี้เป็นค่าที่ถูกเลือกใช้ว่าเหมาะสมกับเกมนี้ เพราะว่าการกินแบบ Jump ที่กินได้ 2-3 ตัวนั้นจะเกิดขึ้นในช่วงต้นของเกมจึงไม่ส่งผลต่อการ ได้เปรียบหรือเสียเปรียบมากนักหากเลือกเดินแบบนี้ แต่หากใช้ค่า 1.5 แทน จะทำให้เลือกเดินแบบ Double มากกว่าทั้งในช่วงต้นและท้ายเกมทำให้ลดโอกาสที่จะจบเกมในช่วงต้นเกมไป หรือถ้าหากเลือกใช้ค่าอื่นๆที่ทำให้เลือกเดินแบบ Double น้อยกว่านี้ก็จะทำให้เกิดข้อเสียเปรียบในช่วงท้ายเกมที่การเดินแบบ Jump เป็นการเดินที่เสียเปรียบค่อนข้างมากด้วย

4.1.3. Weight 1.26

แสดงการเลือกชนิดของการเดิน

Weight 1.26		จำนวนตัวที่กินได้ด้วยการเดินแบบ Jump								
		0	1	2	3	4	5	6	7	8
จำนวนตัวที่กินได้จากการเดินแบบ Double	0	D	J	J	J	J	J	J	J	J
	1	D	D	J	J	J	J	J	J	J
	2	D	D	D	J	J	J	J	J	J
	3	D	D	D	D	J	J	J	J	J
	4	D	D	D	D	D	D	J	J	J
	5	D	D	D	D	D	D	D	J	J
	6	D	D	D	D	D	D	D	D	J
	7	D	D	D	D	D	D	D	D	D
	8	D	D	D	D	D	D	D	D	D

D คือการเลือกเดินแบบ Double
J คือการเลือกเดินแบบ Jump

ตารางที่ 4.3 แสดงการเลือกเดินที่ Weight 1.26

จากตารางที่ 4.3 จะเห็นได้ว่าการเปลี่ยนค่า weight เป็น 1.26 นั้น จะทำให้ทุกๆการกินที่เพิ่มขึ้น 4 ตัวจะทำให้โอกาสในการเดินแบบ Double ถูกเลือกใช้มากกว่าแบบ Jump สำหรับค่านี้จะทำให้ได้เปรียบในการเดินเกมช่วงหลังๆ เพราะโอกาสเลือกเดินแบบ Double มีน้อยอยู่ ซึ่งการเดินแบบ Double จะได้เปรียบมากกว่าในช่วงแรก

4.1.4. Weight 1.21

แสดงการเลือกชนิดของการเดิน

Weight 1.21		จำนวนตัวที่กินได้ด้วยการเดินแบบ Jump								
		0	1	2	3	4	5	6	7	8
จำนวนตัวที่กินได้จากการเดินแบบ Double	0	D	J	J	J	J	J	J	J	J
	1	D	D	J	J	J	J	J	J	J
	2	D	D	D	J	J	J	J	J	J
	3	D	D	D	D	J	J	J	J	J
	4	D	D	D	D	D	J	J	J	J
	5	D	D	D	D	D	D	D	J	J
	6	D	D	D	D	D	D	D	D	J
	7	D	D	D	D	D	D	D	D	D
	8	D	D	D	D	D	D	D	D	D

D คือการเลือกเดินแบบ Double
J คือการเลือกเดินแบบ Jump

ตารางที่ 4.4 แสดงการเลือกเดินที่ Weight 1.21

จากตารางที่ 4.4 จะเห็นได้ว่า ทุกๆการกินที่เพิ่มขึ้น 5 ตัวถึงจะทำให้โอกาสในการเดินแบบ Double เพิ่มขึ้น ก็เช่นเดียวกันกับค่า 1.26 คือ จะได้เปรียบในช่วงหลังมากกว่า แต่ช่วงแรกเสียเปรียบจนอาจจะทำให้แพ้ก่อนที่จะจบเกมก็ได้

4.1.5. Weight 1.17

แสดงการเลือกชนิดของการเดิน

Weight 1.17		จำนวนตัวที่กินได้ด้วยการเดินแบบ Jump								
		0	1	2	3	4	5	6	7	8
จำนวนตัวที่กินได้จากการเดินแบบ Double	0	D	J	J	J	J	J	J	J	J
	1	D	D	J	J	J	J	J	J	J
	2	D	D	D	J	J	J	J	J	J
	3	D	D	D	D	J	J	J	J	J
	4	D	D	D	D	D	J	J	J	J
	5	D	D	D	D	D	D	J	J	J
	6	D	D	D	D	D	D	D	D	J
	7	D	D	D	D	D	D	D	D	D
	8	D	D	D	D	D	D	D	D	D

D คือการเลือกเดินแบบ Double
J คือการเลือกเดินแบบ Jump

ตารางที่ 4.5 แสดงการเลือกเดินที่ Weight 1.17

จากตารางที่ 4.5 จะเห็นว่า ทุกๆการกินที่เพิ่มขึ้น 6 ตัวถึงจะทำให้โอกาสการเดินแบบ Double มากขึ้น ทำให้เป็นการเดินที่เสียเปรียบมากขึ้น เพราะจะทำให้ AI เลือกเดินแบบ Jump เยอะมาก จึงทำให้เสียเปรียบทั้งช่วงต้นและกลางเกมอย่างเห็นได้ชัด

4.1.6. Weight 1.15

แสดงการเลือกชนิดของการเดิน

Weight 1.15		จำนวนตัวที่กินได้ด้วยการเดินแบบ Jump								
		0	1	2	3	4	5	6	7	8
จำนวนตัวที่กินได้จากการเดินแบบ Double	0	D	J	J	J	J	J	J	J	J
	1	D	D	J	J	J	J	J	J	J
	2	D	D	D	J	J	J	J	J	J
	3	D	D	D	D	J	J	J	J	J
	4	D	D	D	D	D	J	J	J	J
	5	D	D	D	D	D	D	J	J	J
	6	D	D	D	D	D	D	D	J	J
	7	D	D	D	D	D	D	D	D	D
	8	D	D	D	D	D	D	D	D	D

D คือการเลือกเดินแบบ Double
J คือการเลือกเดินแบบ Jump

ตารางที่ 4.6 แสดงการเลือกเดินที่ Weight 1.15

จากตารางที่ 4.6 จะเห็นว่า ทุกๆการกินที่เพิ่มขึ้น 7 ตัว โอกาสที่จะเดินแบบ Double จึงจะเพิ่มขึ้น ก็ยิ่งทำให้เสียเปรียบมากกว่าค่าที่ผ่านๆมาอีก และเป็นอย่างนี้ต่อไปเรื่อยถ้าหากเราลดค่าลงไปอีก

สรุปได้ว่า จากตารางที่ผ่านมาทั้ง 6 ตาราง แสดงให้เห็นถึงความแตกต่างและข้อได้เปรียบเสียเปรียบในการเลือกเดิน ที่มี Weight ต่างกันไป ส่วนค่าคงที่ 0.05 ที่ทำการบวกเพิ่มเข้าไปนั้นเพียงเพื่อให้ AI เลือกตัดสินใจเดินแบบ Double มากกว่า Jump ที่มีจำนวนตัวที่กินได้เท่ากัน หรือมีจำนวนตัวที่กินได้เป็น 0 แต่ที่ใช้ค่านี้ก็เพราะจะไปสอดคล้องกับค่า Weight ที่เลือกใช้ ทั้งยังไม่ส่งผลกระทบมากต่อการคิดคะแนนด้วย ดังที่ได้วิเคราะห์มาจึงเห็นว่า ค่าที่ทำให้ได้เปรียบในการเดินมากที่สุดก็คือค่า 1.35 ซึ่งทำให้ได้เปรียบทั้งช่วงต้น และกลางเกม ส่วนในช่วงท้ายเกมนั้น หลายๆค่าก็ทำได้ดีไม่แตกต่างกัน

4.2. ทดสอบการนำไปใช้จริง

เพื่อเป็นการทดสอบว่าค่า Weight ที่ได้เลือกใช้นั้นทำให้ AI มีความฉลาดมากกว่าค่าอื่นๆ จึงได้ทำการทดลองโดยเขียนเกมขึ้นมาเพิ่มโดยเปลี่ยนค่า Weight ของ AI ให้ต่างกันทั้งหมด 6 ค่า คือ 1.5, 1.35, 1.26, 1.21, 1.17 และ 1.15 โดยค่าคงที่ 0.05 ที่นำมาบวกเพิ่มนั้นจะใช้เหมือนกันทั้งหมด แล้วนำเกมที่ได้ไปให้อาสาสมัครทดลองเล่น ได้ผลดังนี้

ผู้เล่น	ครั้งที่	Weight 1.5		Weight 1.35		Weight 1.26		Weight 1.21		Weight 1.17		Weight 1.15	
		ดำ	ขาว	ดำ	ขาว	ดำ	ขาว	ดำ	ขาว	ดำ	ขาว	ดำ	ขาว
A	1	27	22	26	23	24	25	25	24	26	23	29	20
	2	25	24	28	21	26	23	22	27	22	27	24	25
	3	30	19	25	24	25	24	23	26	17	32	22	27
B	1	28	21	30	3	23	26	30	19	25	24	28	21
	2	25	24	26	0	26	23	18	31	26	23	22	27
	3	26	23	27	22	29	20	17	32	20	29	23	26
C	1	30	19	19	0	20	29	27	22	20	29	22	27
	2	26	23	32	17	17	32	19	30	23	26	28	21
	3	27	22	28	21	30	19	25	24	21	28	19	30
D	1	29	20	33	16	22	27	26	23	23	26	29	20
	2	27	22	22	2	28	21	25	24	20	29	26	23
	3	28	21	41	8	19	30	24	25	21	28	21	28
E	1	26	23	40	9	18	31	26	23	22	27	24	25
	2	23	26	38	11	30	19	22	27	24	25	20	29
	3	25	24	29	20	23	26	19	30	20	29	12	37

ตารางที่ 4.7 ผลการทดสอบโดยผู้เล่นอาสาสมัคร

จากตาราง 4.7 ผลการทดสอบจะเห็นว่าค่า Weight ที่มีผลให้ AI ชนะมากที่สุดคือค่า 1.5 และ 1.35 แต่ต่างกันที่ค่า weight 1.35 นั้น จะมีผลการชนะที่ฝ่ายผู้เล่นไม่มีช่องให้เดินก่อนจบเกม และแบบที่ผู้เล่นไม่เหลือตัวเดินเลย ทั้งนี้เพราะการเลือกรูปแบบการเดินที่เหมาะสมต่อช่วงเริ่มเกมและท้ายเกม ทำให้ AI ที่ใช้ค่า weight 1.35 ดีกว่าค่า 1.5 ที่ให้ผลชนะใกล้เคียงกัน

บทที่ 5
สรุปผล

เกม Attaxx นี้เป็นเกมที่ได้รับการพัฒนามาจากเกม Reversi และมีการพัฒนารูปแบบของกระดานออกมามากมายหลายแบบทั้งยังมีชื่อเรียกแตกต่างกันออกไปตามรูปแบบ ซึ่งในโครงงานนี้ก็เป็นอีกรูปแบบหนึ่งของเกมนี้ ที่นำมาประยุกต์ใหม่ด้วยโปรแกรมที่กำลังเป็นที่นิยมในปัจจุบันอย่าง Macromedia Flash ซึ่งนำเสนอในรูปแบบของ Animation เพื่อความสวยงามและเพลิดเพลินของผู้เล่น ทั้งนี้โครงการนี้ยังมีการพัฒนาโปรแกรม AI เพื่อเป็นผู้เล่นจำลองที่สามารถโต้ตอบกับผู้เล่นได้อย่างมีเหตุผล และได้ทำการทดลองโดยให้อาสาสมัครจำนวน 10 คน ทดลองเล่นเกมนี้แล้วบันทึกผลที่ได้เมื่อจบเกม

5.1 สรุปผลการทดลอง

จะเห็นได้ว่าเกมนี้ถูกสร้างขึ้นจากการประยุกต์ของหลายๆเกมที่คนทั่วไปรู้จักและคุ้นเคยเป็นอย่างดีแล้ว เช่น Othello หรือ Reversi ทำให้ผู้เล่นสามารถเข้าใจเกมนี้ได้ไม่ยากนัก และอีกทั้ง AI ของเกมที่ไม่ค่อยซับซ้อนเท่าไร จึงไม่เป็นการยากเกินไปสำหรับผู้ที่เพิ่งเคยเล่น

จากที่ได้ทำการทดสอบในส่วน AI ของเกม ก็ทำให้เห็นว่า AI ที่มีการพัฒนาให้สามารถคำนวณทางเลือกโดยมีผลของความได้เปรียบเสียเปรียบในวิธีการเดินทั้งสองแบบของเกมนี้ คือแบบ jump และแบบ double จะให้ผลเป็นที่น่าพอใจมากกว่าในแง่ของผู้จัดทำ เพราะจะทำให้ AI สามารถตัดสินใจได้ไม่ขัดกับความรู้สึก และเสมือนเป็นผู้เล่นมากกว่า

5.2 ปัญหาและอุปสรรค

เนื่องจากโปรแกรมที่เลือกนำมาพัฒนาเกมนี้ เป็นโปรแกรมที่ผู้จัดทำเองก็ยังไม่เคยมีโอกาสได้ใช้ จึงต้องศึกษาใหม่ทั้งหมด ทำให้ไม่สามารถใช้งานโปรแกรม Macromedia Flash นี้ได้อย่างเต็มประสิทธิภาพ และยังต้องออกแบบเกมเป็นแบบ Animation ด้วย ทำให้ต้องใช้เวลาพอสมควร อีกทั้งยังมีปัญหาในเรื่องของเวลาที่มีในการทำโครงงานนี้ไม่มากนัก เนื่องจากเหตุผลบางประการ และในส่วนของรูปแบบการเดินของเกมที่มี 2 แบบก็เป็นผลให้การออกแบบตัวเกมและ AI ยากขึ้นด้วย

5.3 ข้อเสนอแนะ

ในส่วนของการออกแบบตัวเกมนั้น เป็นเรื่องที่ต้องอาศัยพื้นฐานทางศิลปะพอสมควรและเป็นส่วนที่ใช้เวลามาก อีกทั้งยังเป็นโปรแกรมที่ยังไม่เคยใช้ทำให้ต้องเรียนรู้ใหม่ทั้งหมด แต่ก็ได้คำแนะนำจากพี่ๆในบริษัทที่ไปฝึกงาน ถึงโปรแกรมอีกโปรแกรมหนึ่งคือ Flash decompiler เป็นโปรแกรมที่สามารถจัดเก็บงาน flash จากเว็บไซต์ต่างๆ และนำมาแยกองค์ประกอบเป็นส่วนๆได้ด้วย จึงทำให้สามารถที่จะศึกษาจากงานที่มีคนทำไว้แล้วได้ ทำให้การเรียนรู้เป็นไปได้เร็วขึ้น และอย่างที่ได้กล่าวไว้ในข้างต้นว่า เกมนี้สามารถพลิกแพลงได้หลากหลายรูปแบบตามลักษณะของกระดาน จึงทำให้เกมนี้สามารถพัฒนาได้อีกหลากหลายไปด้วย

ภาคผนวก

ก. โปรแกรม Macromedia Flash และ Action script

โปรแกรม Macromedia Flash เป็นโปรแกรมที่ได้รับความนิยมเป็นอย่างมากในการสร้างงานเกี่ยวกับ flash ทั้งงานการ์ตูน เกม เว็บไซต์หรือสื่อต่างๆ เนื่องจากงานที่เป็น flash นั้นมีขนาดงานที่เล็กและมีการนำเสนอที่น่าสนใจ เข้าใจง่ายด้วยภาพเคลื่อนไหวหรือ Animation จึงทำให้ปัจจุบันมีผู้ให้ความสนใจและพัฒนางานด้านนี้ออกมาเป็นที่แพร่หลาย

หลักการทำงานของโปรแกรมนี้จะทำงานอยู่บน timeline โดยใน timeline นี้จะมี frame อยู่หลายๆ frame ในแต่ละ frame เราสามารถสร้างภาพที่ต้องการให้เคลื่อนไหวได้ โดยการจะทำให้ภาพเคลื่อนไหวได้นั้นอาศัยหลักการคล้ายกับการทำภาพยนตร์ นั่นคือภาพในแต่ละ frame ที่มีความแตกต่างกันเล็กน้อย เมื่อนำมาฉายต่อเนื่องกันทีละ frame ด้วยความไว 12 frame ต่อวินาที ก็จะทำให้เราเห็นเป็นการเคลื่อนไหว แต่ในโปรแกรมนี้เรายังสามารถเขียนชุดของคำสั่งเข้าไปควบคุมภาพแต่ละภาพได้ ชุดคำสั่งนี้เรียกว่า Action script ซึ่ง Action script นี้เราสามารถเขียนลงไปในภาพแต่ละภาพได้เลย หรือจะเขียนไว้ที่ใดก็ได้แล้วใช้การอ้างอิงจากชื่อและตำแหน่งของภาพแต่ละภาพ

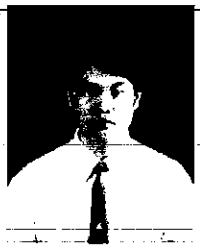
ในการเขียน Action script นี้จะสามารถกำหนดการจัดการกับเหตุการณ์ (Event handler) ได้ เช่น เมื่อมีการใช้เมาส์คลิกที่ภาพจะให้เกิดอะไรขึ้น หรือเมื่อมีการกดแป้นพิมพ์ให้ทำอะไร เป็นต้น ซึ่งการจัดการและสั่งการกับภาพหรือ Movie clip อื่นนั้นสามารถทำได้โดยอ้างอิงจากตำแหน่งที่อยู่โดยที่ timeline หลักนั้นจะมีชื่อเรียกว่า _root หากมีภาพที่อยู่ใน timeline หลักที่ชื่อ sample1 ตำแหน่งของภาพนั้นก็จะเป็น _root.sample1 จะเห็นว่ามีการใช้จุด (.) เป็นตัวคั่นระหว่างแต่ละตำแหน่ง และยังใช้คั่นระหว่างคำสั่งอีกด้วย เช่น _root.sample1.play() จะเป็นการสั่งให้ภาพที่ชื่อ simple1 ที่อยู่บน timeline หลักเล่นหรือทำงานนั่นเอง

บรรณานุกรม

[1] Derek Franklin and Jobe Makar. "Macromedia Flash MX Actionscripting advance training from the source." California, New York : Macromedia press.

[2] ไม่ปรากฏชื่อผู้แต่ง. "Attaxx!!" [Online]. Available : http://www.pressibus.org/ataxx/indexgb.html

ประวัติผู้เขียนโครงงาน



ชื่อ นายจิรวัฒน์ เธียรพิเชษฐพงศ์

ภูมิลำเนา 10/1 ถนนพระยาจักรี ตำบลในเมือง
อำเภอเมือง จังหวัดพิษณุโลก 65000

ประวัติการศึกษา

- จบมัธยมศึกษาจากโรงเรียนพิษณุโลกพิทยาคม จังหวัดพิษณุโลก
- ปัจจุบันศึกษาอยู่ระดับปริญญาตรีชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร จังหวัดพิษณุโลก

E-mail destiny2jt@hotmail.com