การสร้างปัญญาประดิษฐ์คู่ต่อสู้ที่สูสีกับผู้เล่นในเกมวางแผนแบบผลัดกันเล่นในท้องตลาด

นาย กิตติศักดิ์ โพธิศาสตร์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2553

GENERATING AN EVENLY MATCHED OPPONENT AI IN COMMERCIAL TURN-BASED STRATEGY GAMES

Mr. Kittisak Potisartra

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University

Academic Year 2010

หัวข้อวิทยานิพนธ์	การสร้างปัญญาประดิษฐ์คู่ต่อสู้ที่สูสีกับผู้เล่นในเกมวางแผนแบบผลัดกันเล่นในท้องตลาด
โดย	นายกิตติศักดิ์ โพธิศาสตร์
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร.วิษณุ โคตรจรัส

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.เศรษฐา ปานงาม)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(ผู้ช่วยศาสตราจารย์ ดร.วิษณุ โคตรจรัส)

..... กรรมการ

(อาจารย์ ดร.นัทที นิภานันท์)

..... กรรมการภายนอกมหาวิทยาลัย

(ผู้ช่วยศาสตราจารย์ ดร.สุรพงษ์ เลิศสิทธิชัย)

กิตติศักดิ์ โพธิศาสตร์ : การสร้างปัญญาประดิษฐ์คู่ต่อสู้ที่สูสีกับผู้เล่นในเกมวางแผนแบบผลัดกันเล่นในท้องตลาด. (Generating an Evenly Matched Opponent AI in Commercial Turn-based Strategy Games) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ.ดร.วิษณุ โคตรจรัส, 82 หน้า.

การวิจัยเกี่ยวกับด้านเกมวางแผนการรบแบบผลัดกันเล่น ในปัจจุบันส่วนใหญ่มุ่งเน้นไปที่การทำให้ปัญญาประดิษฐ์เล่นได้เก่งขึ้นจนสามารถที่จะเอาชนะผู้เล่นได้ แต่การสร้างปัญญาประดิษฐ์ให้เก่งกว่าผู้เล่นอาจทำให้เกิดปัญหาเมื่อนำเกมออกจำหน่ายเนื่องจากผู้เล่นอาจเกิดความเบื่อหน่ายเมื่อเล่นแล้วไม่สามารถเอาชนะได้ ซึ่งอาจทำให้ผู้เล่นเลิกเล่นเกมไปได้ ดังนั้นการทำให้ปัญญาประดิษฐ์สามารถปรับตัวให้ต่อสู้ได้อย่างทัดเทียมกับความสามารถของผู้เล่นจึงมีความสำคัญมากสำหรับเกมที่มีการจำหน่ายจริง

งานวิจัยนี้ทำการนำเสนออีวาลูเอชัน ฟังก์ชัน เพื่อคำนวณคะแนนของการกระทำของแต่ละตัวละครในแต่ละตาเดิน นอกจากนี้ยังนำเสนอวิธีการ 3 วิธีในการนำคะแนนของตาเดินในอดีตและอนาคตมาใช้ประเมินความสามารถของผู้เล่นเพื่อทำให้ปัญญาประดิษฐ์สามารถเล่นได้ในระดับเดียวกัน โดยที่ไม่มีการแสดงการตัดสินใจที่ไม่สมเหตุสมผล หรือการเปลี่ยนแปลงระดับความยากขึ้นในทันทีทันใด เกมที่ใช้ในงานวิจัยนี้เป็นเกมที่อ้างอิงมาจากเกมตระกูล Final Fantasy Tactics โดยจากการทดลองได้ผลสรุปว่า ปัญญาประดิษฐ์ทั้ง 3 วิธีที่ได้นำเสนอมานั้น สามารถทำการเล่นกับฝ่ายตรงข้ามได้อย่างค่อนข้างสูสี แต่จะมีข้อดีและข้อเสียต่างกันไปโดยรูปแบบที่สามารถเล่นได้สูสีโดยเกิดปัญหาน้อยที่สุด คือ รูปแบบที่ใช้ข้อมูลตาเดินของผู้เล่นที่เคยเกิดขึ้นในอดีตและตาเดินที่คาดคะเนว่าจะเกิดในอนาคตระยะสั้น ตัวปัญญาประดิษฐ์สามารถนำไปใช้ได้กับเกมจริง โดยผู้ใช้งานสามารถปรับเปลี่ยนส่วนของสมการและระดับค่าความสูสีเพื่อรองรับสถานะของเกมที่แตกต่างออกไปได้

ภาควิชา ..วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....

สาขาวิชา ..วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา ..2553.....

5070219021 : MAJOR Computer Engineering

KEYWORDS : TURN-BASED STRATEGY GAME / ARTIFICIAL INTELLIGENCE / EVALUATION FUNCTIONS

KITTISAK POTISARTRA : GENERATING AN EVENLY MATCHED OPPONENT AI IN COMMERCIAL TURN-BASED STRATEGY GAMES. THESIS ADVISOR : ASST PROF. VISHNU KOTRAJARAS, Ph.D., 82 pp.

Recent researches in turn-based strategy (TBS) games concentrate on developing artificial intelligence that beats players. Such artificial opponents in games, however, may cause problems when the games go on sale. Players may get bored from not being able to win and quit playing. Therefore, keeping players engage in games according to their skill level is very important for commercial games.

This thesis introduces evaluation functions for calculating the score from each unit's action. Furthermore, three algorithms for calculating a human opponent's skill level and adjusting the artificial player's skill level to match its human opponent's skill level are introduced. These algorithms make various uses of scores from past and future turns. Unreasonable moves or sudden changes in difficulty level are prevented by the algorithms. A Final Fantasy Tactics-like game is used in our experiment. From the experiment, the three algorithms from this thesis are capable of playing as well as their opponents. Each algorithm has different limitations. The algorithm with the least number of problems takes scores from the past and estimates possible actions for the near future. The proposed algorithm can be used in commercial games by adjusting equations and skill level gaps to support different game states.

Department : Computer Engineering Student's Signature .....

Field of Study : Computer Engineering Advisor's Signature .....

Academic Year : 2010

กิตติกรรมประกาศ

ขอขอบพระคุณบิดา มารดา และครอบครัว ที่เป็นกำลังใจสำคัญ และคอยให้ความช่วยเหลือในทุกๆ ด้าน จนผู้เขียนสามารถทำวิทยานิพนธ์ฉบับนี้ได้สำเร็จ

ขอขอบพระคุณอาจารย์ที่ปรึกษา ผศ.ดร.วิษณุ โคตรจรัส ซึ่งเป็นผู้ให้ข้อคิด แนวทาง คำปรึกษา ตลอดจนช่วยตรวจทาน และแก้ไข จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง รวมถึงให้โอกาสและสิ่งที่ดีแก่ผู้วิจัยเสมอมา

ขอขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ ผศ.ดร.เศรษฐา ปานงาม ดร.นัทที นิภานันท์ และผศ.ดร.สุรพงษ์ เลิศสิทธิชัย ที่ได้กรุณาให้คำแนะนำในการแก้ไขวิทยานิพนธ์ให้มีคุณภาพยิ่งขึ้น วิทยานิพนธ์ฉบับนี้ไม่อาจจะสำเร็จได้หากไม่ได้รับความร่วมมือจากทุกท่าน และขอขอบคุณสมาชิกกลุ่มวิจัยเกม รวมถึงเพื่อนๆ ทุกคนผู้ที่ให้คำแนะนำเพิ่มเติม และช่วยเหลือในการทดลองเสมอมา

ขอขอบคุณทุกท่านที่ช่วยตอบแบบสอบถามต่างๆ ซึ่งเป็นประโยชน์อย่างมากกับวิทยานิพนธ์นี้ และทำให้วิทยานิพนธ์นี้สำเร็จลุล่วงได้ด้วยดี

และท้ายที่สุดนี้ขอขอบคุณนักพัฒนาเกมทั้งหลายที่สร้างสรรค์เกมต่างๆ และเป็นแรงบันดาลใจให้แก่การทำวิทยานิพนธ์เกี่ยวกับเกมเล่มนี้

สารบัญ

สารบัญตาราง

สารบัญภาพ

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เกมคอมพิวเตอร์เป็นศาสตร์ที่มีการพัฒนาในหลากหลายด้าน เช่น คอมพิวเตอร์กราฟิก ระบบติดต่อผู้ใช้ เสียงประกอบ และปัญญาประดิษฐ์ เป็นต้น สำหรับในส่วนของปัญญาประดิษฐ์นั้น นักวิจัยมักเน้นในการสร้างปัญญาประดิษฐ์ให้มีความฉลาดสามารถเอาชนะผู้เล่นที่เป็นมนุษย์ได้อย่างแน่นอน ตัวอย่างเช่น งานวิจัยทางด้านหมากรุก หมากล้อม และงานวิจัยที่ใช้งานการเรียนรู้แบบรีนฟอร์สเม้นท์ เป็นต้น ซึ่งงานวิจัยบางชิ้นนั้นใช้วิธีการทำให้ปัญญาประดิษฐ์เรียนรู้จากนิสัยการเล่นของผู้เล่นแล้วทำการแก้ทางให้สามารถเอาชนะผู้เล่นคนนั้นๆ ได้

สำหรับเกมลักษณะที่ต้องมีการวางแผนการรบนั้น ได้มีการวิจัยเพื่อพัฒนาปัญญาประดิษฐ์ให้เก่งขึ้นในหลากหลายรูปแบบ ทว่าการที่ปัญญาประดิษฐ์ถูกออกแบบให้เก่งขึ้นเรื่อยๆ นั้น อาจกลายมาเป็นปัญหาสำคัญในการจำหน่ายเกมได้ เนื่องจากผู้เล่นไม่สามารถเอาชนะได้ จึงอาจก่อให้เกิดความเบื่อหน่าย และทำให้ผู้เล่นเลิกเล่นไปในเวลาอันไม่นานนัก หรือผู้เล่นอาจทำการโกงเกมด้วยวิธีต่างๆ เพื่อให้ได้รับชัยชนะ เช่น ทำให้ตัวละครของผู้เล่นมีค่าความสามารถต่างๆ หรือค่าสถานะของตัวละครของฝ่ายตนเองสูงกว่าฝ่ายปัญญาประดิษฐ์ขึ้นไปมากโดยจะทำให้สามารถเอาชนะฝ่ายศัตรูได้อย่างแน่นอน หรือทำให้ตัวละครไม่มีวันตาย ซึ่งการโกงเหล่านี้จะทำให้ตัวเกมขาดความท้าทายไปทำให้ผู้เล่นเบื่อและเลิกเล่นไปในเวลาอันรวดเร็วเช่นเดียวกัน ในทางกลับกัน การเขียนโค้ดเพื่อบังคับตัวละครในเกมโดยตรงนั้น แม้จะสามารถทดสอบเพื่อให้ได้ความยากที่คนส่วนใหญ่ยอมรับได้ แต่พฤติกรรมของฝ่ายปัญญาประดิษฐ์ของเกมจะมีค่าคงที่ ผู้เล่นซึ่งพัฒนาตนเองจากประสบการณ์การเล่นจะเล่นได้เก่งกว่าในที่สุด และอาจจะก่อให้เกิดความเบื่อหน่ายจนเลิกเล่นได้เมื่อสามารถคาดการณ์การเล่นของอีกฝ่ายซึ่งจะมีลักษณะคล้ายเดิมตลอดได้ ซึ่งการที่ผู้เล่นเลิกเล่นเกมจะส่งผลกระทบต่อยอดขายของเกมต่อไป ดังนั้นเกมบางเกมจึงได้สร้างโหมดสำหรับเล่นมากกว่าหนึ่งคนเพื่อยืดอายุของเกมที่อาจหมดเร็วเนื่องจากปัญญาประดิษฐ์ที่มีความยาก หรือง่ายจนเกินไป

แม้ว่ามีทางออกในการแก้ปัญหาความยาก หรือง่ายเกินไปจากการหาผู้เล่นอื่นมาเล่นด้วย แต่การเล่นสู้กับปัญญาประดิษฐ์นั้น ยังมีความจำเป็นสำหรับเกมลักษณะวางแผนการรบเป็นอย่างมาก เนื่องจากเนื้อเรื่องในเกมโดยทั่วไปนั้นไม่สามารถนำเสนอผ่านการเล่นกับผู้เล่นคนอื่นได้

ปัญญาประดิษฐ์สำหรับเกมที่วางขายตามท้องตลาดจริงนั้น จึงไม่ควรมุ่งเน้นการสร้างพฤติกรรมที่ฉลาดที่สุด หรือพฤติกรรมที่คงที่ แต่ควรมุ่งเน้นไปในการสร้างพฤติกรรมที่เปลี่ยนแปลงไปได้ตามความสามารถของผู้เล่น เพื่อให้เกิดความท้าทายที่เหมาะสม เป็นการดึงดูดผู้เล่นให้เล่นเกมได้นานที่สุดและในขณะเดียวกันก็ดึงดูดให้ผู้เล่นใหม่ได้มากขึ้น วิทยานิพนธ์ฉบับนี้นำเสนอการพัฒนาปัญญาประดิษฐ์ของเกมวางแผนการรบในลักษณะที่จะสามารถทำการต่อสู้ได้อย่างมีความทัดเทียมกับผู้เล่น และให้ผู้เล่นมีความรู้สึกว่าปัญญาประดิษฐ์นั้นสามารถเล่นได้อย่างสูสีกับผู้เล่นในระหว่างเล่น และท้ายที่สุดเมื่อทำการเล่นจนจบเกมแล้วเกิดการแพ้หรือชนะขึ้น จะมีจำนวนทรัพยากรในส่วนของฝ่ายที่ชนะเหลืออยู่เล็กน้อย (ซึ่งจำนวนทรัพยากรที่เหลือนั้นขึ้นอยู่กับเกม) โดยในวิทยานิพนธ์นี้ใช้ความเห็นจากผู้เล่นเป็นตัวกำหนดค่าของทรัพยากรที่เหลือนี้ เกมวางแผนการรบที่นำมาทดสอบนั้นเป็นเกมวางแผนแบบผลัดกันเล่น ซึ่งในกรณีที่ผู้เล่นยังเล่นไม่เก่งนักปัญญาประดิษฐ์จะไม่ฉลาดจนเกินไป แต่หากผู้เล่นเล่นไปเรื่อยๆ แล้วมีความเก่งกาจ หรือชำนาญมากขึ้นแล้ว ปัญญาประดิษฐ์จะปรับตนเองให้มีความเก่งขึ้น และสามารถต่อสู้ได้อย่างทัดเทียมกับตัวผู้เล่นต่อไป

1.2 วัตถุประสงค์ของการวิจัย

เพื่อหาแนวทางในการสร้างปัญญาประดิษฐ์สำหรับเกมวางแผนการรบแบบผลัดกันเล่นที่จะทำให้สามารถเล่นได้สูสีกับผู้เล่นมากขึ้น

1.3 ขอบเขตของการวิจัย

1. ทำการวิจัยเกมในรูปแบบของเกมวางแผนการรบแบบผลัดกันเล่นที่ได้ทำการเขียนขึ้น
2. เกมที่เขียนขึ้นมีลักษณะเป็นการควบคุมตัวละครเข้าต่อสู้กันระหว่างผู้เล่นจำนวนสองฝ่าย
3. ผู้เล่นทั้งสองฝ่ายมีจำนวนตัวละครเท่ากัน และอาชีพของแต่ละตัวเหมือนกัน
4. ลักษณะช่องในการเดินจะเป็นช่องสี่เหลี่ยม และสามารถเดินได้ในทิศทางซ้าย ขวา ขึ้น และลง โดยไม่สามารถเดินในแนวเฉียงได้
5. ตัวละครในเกมมีการโจมตีได้หลายรูปแบบขึ้นอยู่กับความสามารถของตัวละครนั้นๆ เช่น ตัวละครที่เป็นจอมเวทมนตร์ดำ จะสามารถโจมตีได้ทั้งแบบการโจมตีธรรมดา

หรือสามารถโจมตีโดยการใช้เวทมนตร์ได้ หากมีค่าพลังเวทมนตร์ (magic point: mp) เพียงพอ

6. แผนที่ของเกมได้ทำการอ้างอิงมาจากเกม Final Fantasy Tactics A2: Grimoire of the rift [16]

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ตัวอย่างปัญญาประดิษฐ์ที่จะสามารถควบคุมการต่อสู้ในเกมวางแผนการรบแบบผลัดกันเล่นให้สามารถต่อสู้ได้มีความสูสีกับผู้เล่นมากขึ้น
2. ได้ตัวอย่างอีวาลูเอชัน ฟังก์ชันที่สามารถนำมาใช้กับเกมประเภทวางแผนการรบแบบผลัดกันเล่นที่มีแนวการเล่นในลักษณะคล้ายกับเกมที่วางขายตามท้องตลาด
3. เกิดองค์ความรู้ใหม่ในสาขาปัญญาประดิษฐ์

1.5 ขั้นตอนการดำเนินงาน

1. ศึกษาทฤษฎีพื้นฐาน และงานวิจัยที่เกี่ยวข้อง
2. ออกแบบและพัฒนาเกมในลักษณะเกมวางแผนการรบแบบผลัดกันเล่นที่สามารถรองรับปัญญาประดิษฐ์ที่จะออกแบบได้
3. ออกแบบอีวาลูเอชัน ฟังก์ชัน และการคำนวณค่าคะแนนความสูสี
4. นำอีวาลูเอชัน ฟังก์ชัน และการคำนวณความสูสีกับผู้เล่นนั้นไปใช้กับปัญญาประดิษฐ์ที่สร้างขึ้น
5. ทดสอบการทำงานของปัญญาประดิษฐ์
6. วิเคราะห์ผลการทดลอง
7. สรุปผลการวิจัยและเรียบเรียงวิทยานิพนธ์

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เกมวางแผนการรบแบบผลัดกันเล่น (Turn-based Strategy Games, TBS)

เกมวางแผนการรบแบบผลัดกันเล่นเป็นเกมในประเภทวางแผนการรบ (Strategy games) ชนิดหนึ่ง [1] ซึ่งรูปแบบของการเล่นของเกมวางแผนการรบนั้นจะมีลักษณะคือ มีความท้าทายให้กับผู้เล่นโดยการใส่เงื่อนไขที่จะทำให้ได้รับชัยชนะ ซึ่งมักจะเป็นการวางแผนการเล่นเพื่อให้ตัวละครกระทำสิ่งต่างๆ กับฝ่ายตรงข้ามหนึ่งฝ่าย หรืออาจมากกว่านั้น แล้วได้ผลลัพธ์ออกมาตามที่ต้องการ ตัวอย่างเช่น ให้ทำการยึดฐานของฝ่ายตรงข้าม หรือให้คอยคุ้มกันสิ่งก่อสร้างบางอย่างของฝ่ายเรา เป็นต้น ซึ่งเกมวางแผนการรบโดยส่วนใหญ่จะใช้กลวิธีต่างๆ ในการลดจำนวนกองทัพของศัตรูเป็นเป้าหมายหลักเพื่อให้ได้รับชัยชนะ ดังนั้นเกมวางแผนการรบจึงเป็นเกมที่มีลักษณะเหมือนการทำสงครามเสมือนนั่นเอง ตัวอย่างของเกมประเภทนี้ เช่น เกมหมากรุก เกม Age of Empires เกม Command & Conquer และเกม Advance Wars เป็นต้น รูปที่ 1 แสดงตัวอย่างเกมประเภทวางแผนการรบ

รูปที่ 1 แสดงตัวอย่างเกมวางแผนการรบ เช่น เกม Command & Conquer: Red Alert 3 [7]

เกมวางแผนการรบจะแบ่งออกเป็นสองประเภทย่อยๆ ได้แก่ เกมวางแผนการรบแบบผลัดกันเล่น และเกมวางแผนการรบแบบทันกาล (Real-Time Strategy games, RTS) ซึ่งลักษณะของการเล่นแบบวางแผนโดยแท้จริงแล้วนั้นจะคล้ายกับเกมวางแผนการรบแบบผลัดกันเล่นมากกว่าเกมวางแผนการรบแบบทันกาล เพราะเกมวางแผนการรบแบบผลัดกันเล่นนั้น ผู้เล่นจะสามารถใช้เวลาในการคิดหาตาเดินที่ดีที่สุดอย่างรอบคอบด้วยเวลาเท่าใดก็ได้ และโดยเฉพาะกับในการเล่นแบบเล่นคนเดียว (single player) ซึ่งคอมพิวเตอร์จะสามารถรอได้ไม่ว่าผู้เล่นจะต้องการใช้เวลานานเท่าใด

เกมวางแผนการรบแบบผลัดกันเล่นส่วนใหญ่จะเป็นเกมที่ผู้เล่นจะต้องทำการควบคุมกองทัพของฝ่ายตนเองเข้าต่อสู้กับกองทัพของฝ่ายศัตรู โดยลักษณะการเล่นจะผลัดกันเดินคนละตา มีการเดินลักษณะเป็นช่องคล้ายกับเกมหมากรุก ดังตัวอย่างใน รูปที่ 2 และรูปที่ 3 โดยตัวละครแต่ละตัวจะมีค่าสถานะ (Status) ของตัวเอง เช่น ค่าการโจมตี การป้องกัน ความเร็ว ระยะทางที่สามารถเดินได้ และค่าอื่นๆ ซึ่งค่าเหล่านี้จะมีลักษณะแตกต่างกันไปในแต่ละตัวละคร นอกจากนี้ความสามารถในการกระทำ (Action) ต่างๆ ของตัวละครแต่ละตัวก็อาจมีค่าที่ต่างกันได้ ซึ่งในบางเกมอาจมีการแบ่งประเภทของตัวละคร (ในคำศัพท์ของการเล่นเกม ประเภทของตัวละครโดยส่วนใหญ่แล้วจะถูกเรียกว่า อาชีพ) ออกเป็นหลายๆ อาชีพ ซึ่งหากตัวละครมีอาชีพเดียวกันก็จะมีความสามารถคล้ายกัน แต่ถ้าต่างอาชีพกัน ความสามารถก็จะต่างกันออกไป ตัวอย่างของเกมวางแผนการรบแบบผลัดกันเล่น เช่น เกมตระกูล Final Fantasy Tactics เกม Advance Wars และเกมตระกูล Fire Emblem เป็นต้น

รูปที่ 2 แสดงตัวอย่างเกม Final Fantasy Tactics Advance [17] ซึ่งเป็นเกมในลักษณะ Turn-Based Strategy games

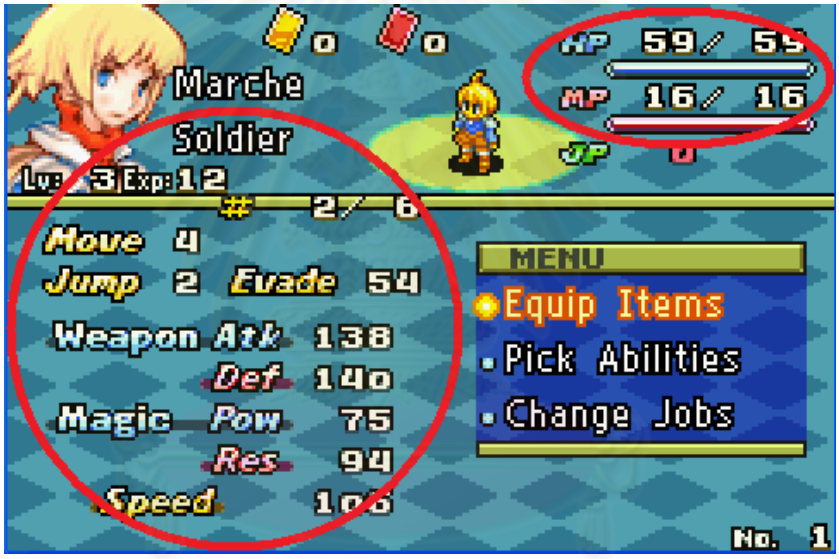
รูปที่ 3 แสดงตัวอย่างเกม Fire Emblem: The Sacred Stones [11] ซึ่งเป็นเกมในลักษณะ Turn-Based Strategy games

สำหรับตัวอย่างความสามารถของตัวละครในแต่ละอาชีพที่แตกต่างกันนั้น ยกตัวอย่างเช่นจากเกม Final Fantasy Tactics: The War of the Lions [18] ความสามารถของตัวละครที่มีอาชีพอัศวิน (knight) และตัวละครที่มีอาชีพธนู (archer) ก็จะต่างกัน โดยปกติแล้วอาชีพอัศวินนั้นจะมีระยะการเดินได้เท่ากับ 3 ช่อง และระยะที่โจมตีได้จะมีระยะ 1 ช่องจากตัวละครเนื่องจากเป็นการโจมตีในระยะประชิด ซึ่งระยะโจมตีนั้นแสดงโดยแถบที่อยู่ทับแผนที่บนพื้น ดังรูปที่ 4 ซึ่งตัวละครที่ทำการโจมตีคือตัวละครที่มีอักษร AT อยู่ด้านบน แต่หากเป็นอาชีพนักธนูซึ่งเป็นอาชีพที่โจมตีจากระยะไกลแล้ว แม้ระยะการเดินได้จะเท่ากับ 3 ช่องเช่นกัน แต่ระยะที่โจมตีได้จะต้องห่างจากตัวละครเองไป 3 ช่องจนถึง 5 ช่อง ดังรูปที่ 5 แต่ความรุนแรงของการโจมตีก็จะมีค่าน้อยกว่าการโจมตีของอัศวินเช่นกัน

รูปที่ 4 แสดงขอบเขตการโจมตีของตัวละครอาชีพอัศวินของเกม Final Fantasy Tactics: The War of the Lions [18]

รูปที่ 5 แสดงขอบเขตการโจมตีของตัวละครอาชีพนักธนูของเกม Final Fantasy Tactics: The War of the Lions [18]

เกมวางแผนการรบในปัจจุบันใช้การปรับความสูสีด้วยการปรับค่าสถานะของตัวละคร ดังเช่นในวงกลมสีแดงของรูปที่ 6 แต่ไม่ได้ทำการปรับในส่วนของการใช้กลยุทธ์ในการทำให้เกิดความสูสี ตัวอย่างเช่น หากตัวละครฝ่ายผู้เล่นมีค่าระดับของตัวละคร (หรือค่า Lv) เท่ากับ 3 ดังในรูปที่ 6 ซึ่งหากเกมต้องการจะให้ตัวละครฝ่ายศัตรูทำการต่อสู้ได้สูสีกับฝ่ายของผู้เล่น ก็จะทำการสร้างตัวละครที่มีระดับ 3 เท่าๆ กัน หรืออาจมากกว่าหรือน้อยกว่าเล็กน้อย เช่น ระดับ 2 หรือ ระดับ 4 เป็นต้น ซึ่งจะทำให้ค่าสถานะต่างๆ ของตัวละครทั้งสองฝ่ายโดยรวมค่อนข้างมีความใกล้เคียงกัน หากตัวละครมีระดับอื่นๆ เช่น ระดับ 35 เมื่อเกมต้องการให้มีความสูสีเกิดขึ้นก็จะทำการสร้างตัวละครในระดับใกล้ๆ กันเข้ามาทำการต่อสู้กับทางฝ่ายผู้เล่นนั่นเอง

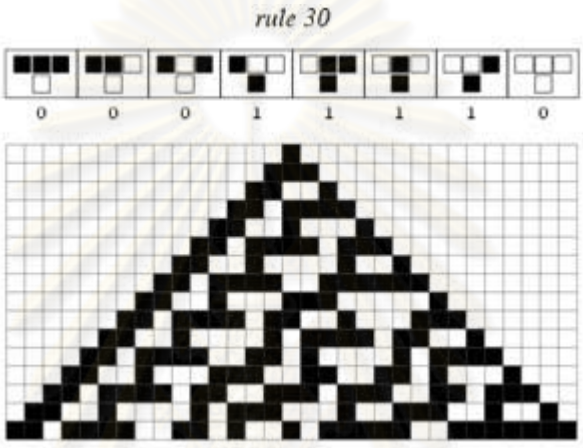


รูปที่ 6 แสดงค่าสถานะของตัวละครในเกม Final antasy Tactics Advance [17]

2.1.2 เซลลูลาร์ ออโตมาตา (Cellular Automata)

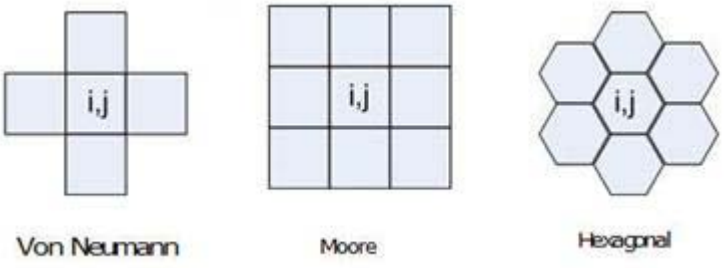
เซลลูลาร์ ออโตมาตา [8] เป็นโมเดลทางคณิตศาสตร์แบบไม่ต่อเนื่อง (Discrete Model) ซึ่งจะมีลักษณะเป็นกลุ่มของเซลล์ที่มาเรียงต่อกันอย่างเป็นระเบียบ โดยมีขนาดได้ไม่จำกัด แต่มิติจะมีขนาดที่จำกัด และเซลล์แต่ละเซลล์จะมีสถานะ (State) ได้จำกัด และขึ้นอยู่กับสถานะของเซลล์รอบข้างในเวลาก่อนหน้านั้น 1 หน่วยเวลา ซึ่งเวลาจะเป็นแบบไม่ต่อเนื่อง โดยกฎในการเปลี่ยนสถานะของเซลล์จะเหมือนกันทุกเซลล์ในเซลลูลาร์ออโตมาตาเดียวกัน และอาจจะเป็นได้ทั้งฟังก์ชันที่ให้ผลแน่นอน (Deterministic Functions) หรือ ฟังก์ชันของความน่าจะเป็น

(Probabilistic Functions) เซลลูลาร์ ออโตมาตาที่พบเห็นได้โดยทั่วไปมักจะเป็นชนิดที่เซลล์เรียงต่อกันเป็นตารางสี่เหลี่ยม เช่น เกมชีวิตของคอนเวย์ (Conway's Game of Life) ซึ่งถูกประดิษฐ์ขึ้นโดย John Horton Conway ในปี ค.ศ.1970 โดยเป็นเซลลูลาร์ ออโตมาตาสองมิติ ที่มีสถานะสองสถานะ รูปที่ 7 แสดงเซลลูลาร์ ออโตมาตาที่มีเพียงหนึ่งมิติ โดยแต่ละเซลล์จะมีคุณสมบัติเปลี่ยนไปตามกฎที่นิยามไว้ด้านบนของรูป



รูปที่ 7 เซลลูลาร์ ออโตมาตาหนึ่งมิติในสิบห้าหน่วยเวลา (เวลาอยู่ในแกนแนวตั้ง) โดยใช้กฎ "rule 30" ซึ่งกำหนดไว้ดังตารางส่วนบนของภาพ และมีสถานะเริ่มต้นดังตารางส่วนล่างในแถวบนสุด [8]

สำหรับเซลลูลาร์ ออโตมาตาแบบ 2 มิตินั้น เราสามารถทำการกำหนดเซลลูลาร์ออโตมาตาในรูปทรงหลายเหลี่ยมได้ เช่น สี่เหลี่ยมที่มี 8 เซลล์รอบข้าง หรือหกเหลี่ยมที่มี 6 เซลล์รอบข้างดังรูปที่ 8 เป็นต้น



รูปที่ 8 แสดงรูปแบบของเซลลูลาร์ออโตมาตาแบบ 2 มิติพร้อมเซลล์รอบข้าง [9]

ซึ่งเราสามารถนำเซลลูลาร์ออโตมาตานี้มาช่วยในเรื่องของการคำนวณลักษณะการคิดของตัวละครได้ เปรียบเสมือนการใช้กฎเพื่อเปลี่ยนสถานะเซลล์ เพียงแต่เซลล์ที่จะเปลี่ยนสถานะนั้นจะต้องมีตัวละครหรือสิ่งที่เกี่ยวข้องกับตัวละครอยู่ภายในก่อน (สถานะของเซลล์ในที่นี้คือ การมีตัวละครอยู่ภายในหรือไม่นั่นเอง) ส่วนเซลล์อื่นๆ ก็สามารถเปลี่ยนสถานะได้ถ้าได้รับผลกระทบจากเซลล์ที่มีตัวละคร ตัวอย่างเช่น การใช้กฎสำหรับตัวละครประเภทโจมตีระยะไกล ซึ่งหากพอตัวละครประเภทโจมตีระยะใกล้ของฝ่ายศัตรูอยู่ในเซลล์ด้านหน้าจะทำการเดินถอยออกไปที่เซลล์ที่ไกลที่สุดเพื่อหลีกเลี่ยงการโจมตีของศัตรู แต่เซลล์นั้นจะยังอยู่ในระยะที่ตนเองสามารถโจมตีศัตรูได้ (ส่งผลให้สถานะของเซลล์ที่อยู่เดิมเปลี่ยนเป็นไม่มีตัวละคร ส่วนสถานะของเซลล์ที่อยู่ไกลสุดเปลี่ยนเป็นมีตัวละคร) แล้วจึงทำการโจมตีใส่ศัตรู หรือการใช้กฎสำหรับตัวละครประเภทโจมตีระยะใกล้ ซึ่งหากพบตัวละครฝ่ายศัตรูอยู่ในเซลล์ด้านหน้าจะทำการเดินไปให้ใกล้เซลล์นั้นที่สุดแล้วทำการโจมตีศัตรูอยู่ในเซลล์นั้น เป็นต้น

2.1.3 อีวาลูเอชัน ฟังก์ชัน (Evaluation function)

อีวาลูเอชัน ฟังก์ชัน [10] หรือที่รู้จักกันในนามของฮิวริสติก อีวาลูเอชัน ฟังก์ชัน (Heuristic evaluation function) หรือสเตติก อีวาลูเอชัน ฟังก์ชัน (Static evaluation function) เป็นฟังก์ชันที่ใช้เพื่อประมาณค่าความดีของตำแหน่ง minimax ของอัลกอริทึมในโปรแกรมการเล่นเกม (game-playing programs) ซึ่งวิธีการสร้างฟังก์ชันเหล่านี้นั้นอาจจะเน้นในด้านของความเร็วหรือด้านประสิทธิภาพก็ได้ ขึ้นอยู่กับผลลัพธ์ที่ผู้สร้างนั้นต้องการ (ด้วยเหตุนี้จึงเรียกได้ว่าเป็นฮิวริสติก) และฟังก์ชันนั้นจะมองเพียงแค่ในตำแหน่งปัจจุบันไม่ได้กระจายออกไปสำรวจตาเดินอื่นๆ ที่อาจมีความเป็นไปได้ (ด้วยเหตุนั้นจึงเรียกว่าเป็นสเตติก)

แนวทางในการสร้างอีวาลูเอชัน ฟังก์ชันที่นิยมอย่างหนึ่งคือ การถ่วงน้ำหนักในการรวมค่าของแต่ละองค์ประกอบ (Factor) ที่มีผลกระทบกับค่าในแต่ละตำแหน่ง ยกตัวอย่างเช่น อีวาลูเอชันฟังก์ชันของเกมหมากรุกซึ่งอาจจะมีการใช้ดังรูปของสมการ (2.1)

$$C_1 \times MATERIAL + C_2 \times MOBILITY + C_3 \times KING\ SAFETY + \cdots \quad (2.1)$$

โดย C_1, C_2, …, C_n คือ ค่าน้ำหนักขององค์ประกอบต่างๆ ที่ใช้คำนวณ

สำหรับผู้เล่นเกมหมากรุกในระดับเริ่มต้นทั่วไปโดยส่วนใหญ่เมื่อทำการเล่นจะมีลักษณะการคิดคล้ายกับโปรแกรมหมากรุกทั่วไป คือ จะคิดคะแนนเฉพาะกับค่าของตัวหมากที่มีอยู่ในกระดาน ตัวอย่างเช่น เขาจะให้คะแนนกับตัวหมากแต่ละตัว และทำการรวมคะแนนของตัวหมากทั้งหมดในกระดาน เป็นต้น โดยทั่วไปแล้วอีวาลูเอชัน ฟังก์ชันในขั้นที่สูงขึ้นไปของคอมพิวเตอร์จะคำนวณค่าของตัวหมากได้อย่างซับซ้อนกว่าที่มนุษย์คำนวณอย่างคร่าวๆ ทั่วไปในขณะที่ใช้เวลาในการคำนวณเท่ากัน ซึ่งจากข้อได้เปรียบของคอมพิวเตอร์คือ สามารถทำการคำนวณได้อย่างรวดเร็วนั้น ทำให้สามารถที่จะสำรวจคะแนนจากสิ่งแวดล้อมต่างๆ ได้มากกว่า ซึ่งจากผลนี้ทำให้โปรแกรมของหมากรุกทั่วไปเน้นไปที่การเลือกตาเดินจากคะแนนที่คำนวณได้และเห็นว่าดีมากที่สุด มากกว่าทางฝั่งมนุษย์ที่เน้นแผนการรบ ซึ่งในตานั้นอาจไม่ใช่คะแนนที่ดีที่สุดแต่อาจส่งผลต่อเกมต่อๆ ไปได้ เป็นต้น

2.1.4 ปัญญาประดิษฐ์ (Artificial Intelligence, AI)

ปัญญาประดิษฐ์ [20] โดยทั่วไปจะหมายถึง ความฉลาดเทียมที่สร้างขึ้นให้กับสิ่งที่ไม่มีชีวิต ซึ่งหากเป็นในทางด้านคอมพิวเตอร์นั้น จะมีความหมายถึงการสร้างระบบคอมพิวเตอร์ที่ชาญฉลาด และสามารถที่จะนำมาทำงานแทน หรือช่วยมนุษย์ทำงานที่ต้องใช้ความฉลาดนั้นๆ

สำหรับในทางด้านของเกม ปัญญาประดิษฐ์จะหมายถึงวิธีการที่ใช้ในคอมพิวเตอร์ หรือวีดีโอเกม ที่จะสร้างความฉลาดให้กับตัวละครที่ไม่มีผู้บังคับ (non-player characters: NPCs) ในพฤติกรรมต่างๆ ไม่ว่าจะเป็นการกระทำต่างๆ เช่นการพูดคุย หรือการต่อสู้ ซึ่งไม่ว่าจะเป็นวิธีการทำงาน หรือวิธีการสร้างใดๆ ก็ตามจะถือว่าอยู่ในขอบเขตของปัญญาประดิษฐ์ทั้งสิ้น อย่างไรก็ตามความหมายของคำว่า ปัญญาประดิษฐ์สำหรับด้านเกมนั้นโดยส่วนใหญ่จะหมายถึง กลุ่มของอัลกอริทึมที่เพิ่มเข้ามาในการควบคุมเกมให้เป็นไปในแบบหรือแนวทางที่ได้กำหนดไว้ [5]

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยที่เกี่ยวข้องแบ่งออกเป็นสองแขนง คือ งานวิจัยที่พัฒนาปัญญาประดิษฐ์ให้เอาชนะผู้เล่น กับงานวิจัยที่พัฒนาปัญญาประดิษฐ์ให้มีความสามารถใกล้เคียงกับผู้เล่น

2.2.1 งานวิจัยที่พัฒนาปัญญาประดิษฐ์ให้เอาชนะผู้เล่น

สำหรับโปรแกรมที่พัฒนาทางมาสำหรับใช้ในการเล่นหมากรุกนั้น ส่วนใหญ่ได้มีการนำ อีวาลูเอชัน ฟังก์ชันมาช่วยในการคิด และทำนายการเดินล่วงหน้าของผู้เล่นทั้งสองฝ่าย แล้วจึงทำการหาตำแหน่งที่ดีที่สุดในการเลือกเดินตัวหมากไปยังตำแหน่งนั้นๆ ซึ่งตัวอย่างของอีวาลูเอชันฟังก์ชันของโปรแกรมหมากรุก เช่น The greenbiatt chess program [6] นั้นได้มีการใช้อีวาลูเอชันฟังก์ชันบนกระดานในลักษณะดังสมการที่ (2.2)

$$S = B + R + P + K + C \tag{2.2}$$

โดยที่ ค่า B คือ เทอมของ material balance

ค่า R คือ เทอมของ piece ratio change

ค่า P คือ เทอมของ pawn structure

ค่า K คือ เทอมของ king safety และ

ค่า C คือ เทอมของ center control

สำหรับค่าในเทอมของ material balance ที่ใช้ในอีวาลูเอชัน ฟังก์ชันนี้มีค่าดังแสดงในตารางที่ 1 โดยค่า B นั้นจะเป็นผลรวมของค่าของตัวหมากทั้งหมดของฝ่ายขาวที่เหลืออยู่บนกระดานลบกับผลรวมของค่าของตัวหมากทั้งหมดของฝ่ายดำที่เหลืออยู่บนกระดาน

Piece	Value	Value Relative to Pawn
Pawn	128	1
Knight	416	3.25
Bishop	448	3.50
Rook	640	5
Queen	1248	9.75
King	1536	12

ตารางที่ 1 แสดงค่า material balance term ที่ใช้ในอีวาลูเอชัน ฟังก์ชันของโปรแกรม the greenbiatt chess program

สำหรับค่าในเทอมของ piece ratio change จะเป็นค่าที่เอาไว้สำหรับใช้คำนวณอัตราการได้เสียของตัวหมากที่จะทำการแลกเปลี่ยนโดยการกินตัวหมากของฝ่ายตรงข้ามแล้วเสียหมากตัวนั้นไป ตัวอย่างเช่น หากกรณีที่ฝ่ายเรานำ bishop ไปกิน knight ของฝ่ายตรงข้าม แต่หลังจากนั้นจะถูกฝ่ายตรงข้ามกิน bishop ของฝ่ายเราไป ในกรณีนี้จะให้ค่าเป็นด้านบวกสำหรับเทอมของ piece ratio change ซึ่งค่า R คิดได้จากสมการ (2.3)

$$R = \{N/(T-1)\} \times \frac{1}{8} \times M \qquad (2.3)$$

โดยที่ ค่า N คือ อัตราส่วนของค่าของตัวหมากฝ่ายเราเทียบกับตัวหมากฝ่ายตรงข้ามในการแลกเปลี่ยนที่คำนวณได้ ณ ขณะนั้น

ค่า T คือ อัตราส่วนของค่าของตัวหมากฝ่ายเราเทียบกับตัวหมากฝ่ายตรงข้ามในการแลกเปลี่ยนที่ดีที่สุดที่คำนวณได้ เช่น จากตัวอย่างนี้อาจไม่ใช่การที่นำ bishop มากิน knight แต่อาจเป็นตัวหมากตัวอื่นที่แลกเปลี่ยนแล้วถือว่าคุ้มค่าที่สุด และ

ค่า M คือ ค่าของตัวหมากทั้งหมดของหนึ่งฝ่ายในขณะที่ทำการเริ่มเกม

สำหรับการประเมินค่าอัตราส่วนนี้จะใช้ค่าดังเช่นในตารางที่ 1 แต่ค่าของ king นั้นจะมีค่าเป็น 1 แทนที่จะเป็นค่า 1536

ส่วนเทอมของ pawn structure นั้นค่าคะแนนจะมีค่าในทางบวกเมื่อมีลักษณะเป็นไปตามเงื่อนไขที่ขึ้นอยู่กับ 4 เทอมย่อยๆ คือ

- pawn ของฝ่ายเรามีจำนวนเป็นสามเท่าของ pawn ของฝ่ายตรงข้าม (หรือสองเท่าในกรณีที่ pawn นั้นอยู่ห่างออกจากกันเท่านั้น)
- pawn ของฝ่ายตรงข้ามเหลืออยู่ในลักษณะแยกห่างออกจากกัน หรือก็คือไม่มี pawn ตัวอื่นๆ ที่เป็นฝ่ายเดียวกันมาอยู่ในช่องข้างเคียงเลย
- pawn ของฝ่ายของเราสามารถผ่านฝ่ายตรงข้ามมาได้แล้ว และไม่มี pawn ตัวใดของฝ่ายตรงข้ามมาขวางได้อีก หรือก็คือไม่มี pawn ของฝ่ายตรงข้ามอยู่ใน file (เป็นศัพท์ในทางหมากรุก หมายถึงแถวในแนวตั้งแต่ละแถวในกระดานหมากรุก) เดียวกันกับ pawn ตัวนั้น หรืออยู่ file ที่ติดกันเลย
- pawn ของฝ่ายตรงข้ามโดนตัวหมากของฝ่ายเราดักทางไปต่อทำให้ไม่สามารถเดินหน้าต่อไปได้

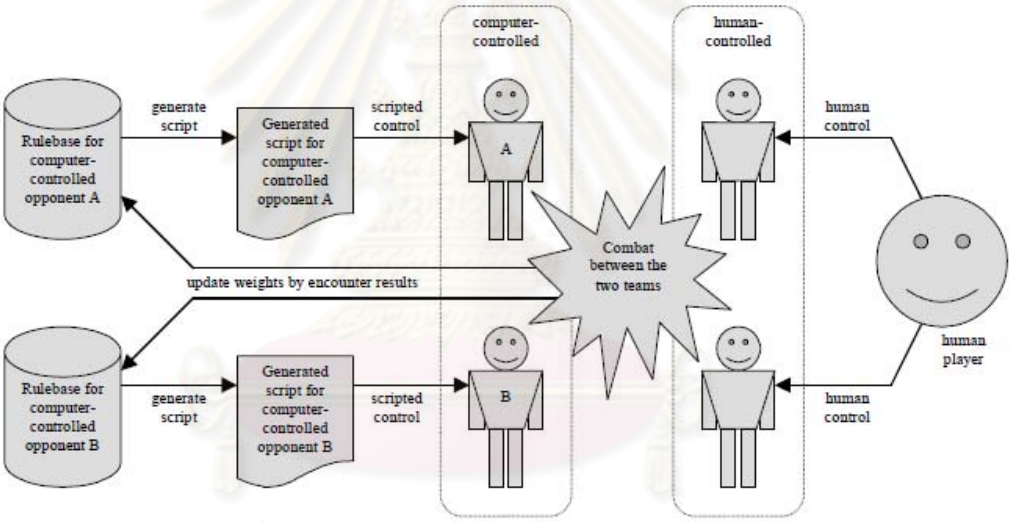
ในส่วนของเทอมของ king safety นั้นจะคิดเฉพาะในกรณีที่มีตัวหมาก queen ของทั้งสองฝ่ายอยู่บนกระดานเท่านั้น ซึ่งค่า K นั้นคือค่า 8 คูณกับ rank (เป็นศัพท์ในทางหมากรุก หมายถึงแถวในแนวนอนในแต่ละแถวของกระดานหมากรุก) ที่ king ฝ่ายตรงข้ามอยู่ ลบกับค่า 8 คูณกับ rank ที่ king ฝ่ายเราอยู่

สำหรับเทอมของ center control นั้น ค่า C นั้นจะมีค่าเป็น +1 หากมี pawn ของฝ่ายเราอย่างน้อย 1 ตัวอยู่ในอาณาเขตของ 4 ช่องกลางของกระดาน โดยที่ไม่มี pawn ของฝ่ายตรงข้ามอยู่เลย และในทางกลับกันจะมีค่าเป็น -1 เมื่อมี pawn ของฝ่ายตรงข้ามอย่างน้อย 1 ตัวอยู่ในอาณาเขตของ 4 ช่องกลางของกระดานโดยที่ไม่มี pawn ของฝ่ายเราอยู่เลย สำหรับในกรณีนอกเหนือจากนี้จะถือว่ามีค่าเป็น 0

ซึ่งจากตัวอย่างของอีวาลูเอชัน ฟังก์ชันของ The greenbiatt chess program นี้เป็นตัวอย่างหนึ่งของการคิดอีวาลูเอชัน ฟังก์ชันในโปรแกรมหมากรุก ซึ่งสามารถนำไปประยุกต์ใช้กับงานวิทยานิพนธ์นี้ได้ในส่วนหนึ่ง แต่เนื่องจากในงานวิทยานิพนธ์นี้ไม่ได้จำเพาะเจาะจงกับเกมหมากรุก แต่ใช้ในส่วนของเกมวางแผนการรบที่ขายตามท้องตลาดทั่วไปส่วนใหญ่ จึงต้องมีการ

ประยุกต์ และทำการปรับเปลี่ยนค่า และลักษณะการคิดค่าต่างๆ ของอีวาลูเอชัน ฟังก์ชันเพื่อให้มีความสอดคล้องกับเงื่อนไขการเล่นของเกมในลักษณะวางแผนการรบด้วย

Pieter Spronck และคณะ [14] [15] ได้ทำการทดลองปัญญาประดิษฐ์แบบไดนามิคสคริปต์ (Dynamic Script) ซึ่งไดนามิคสคริปต์นั้น อาศัยผู้เชี่ยวชาญ (หรืออาจใช้อัลกอริทึมการวิวัฒนาการ) นิยามพฤติกรรมที่ดีสำหรับแต่ละสถานการณ์ในเกมไว้ล่วงหน้า น้ำหนักของพฤติกรรมต่างๆ ที่นิยามไว้จะถูกเปลี่ยนไปเรื่อยๆ ในระหว่างเล่นเกม ตามค่าคะแนนที่ปัญญาประดิษฐ์ทำได้ในแต่ละสถานการณ์ เพื่อให้ปัญญาประดิษฐ์เรียนรู้เลือกการกระทำที่ทำให้ตนได้เปรียบมากที่สุดจากแต่ละสถานการณ์



รูปที่ 9 แสดงรูปแบบการทำงานของไดนามิคสคริปต์ [14] [15]

จากรูปที่ 9 จะเห็นได้ว่าฝ่ายที่ปัญญาประดิษฐ์ควบคุม (computer-controlled) จะใช้ฐานข้อมูลของกฎเป็นตัวทำการสร้างสคริปต์ขึ้นเพื่อนำมาใช้ควบคุมตัวละครให้ทำการต่อสู้ หลังจากนั้นเมื่อการต่อสู้แต่ละรอบสิ้นสุดจะทำการปรับค่าน้ำหนักของพฤติกรรมต่างๆ แล้วใช้ฐานข้อมูลของกฎสร้างสคริปต์ขึ้นมาใหม่เรื่อยๆ ซึ่งเขาได้นำหลักการนี้ไปลองทดสอบกับเกม NeverWinter Nights ดังรูปที่ 10

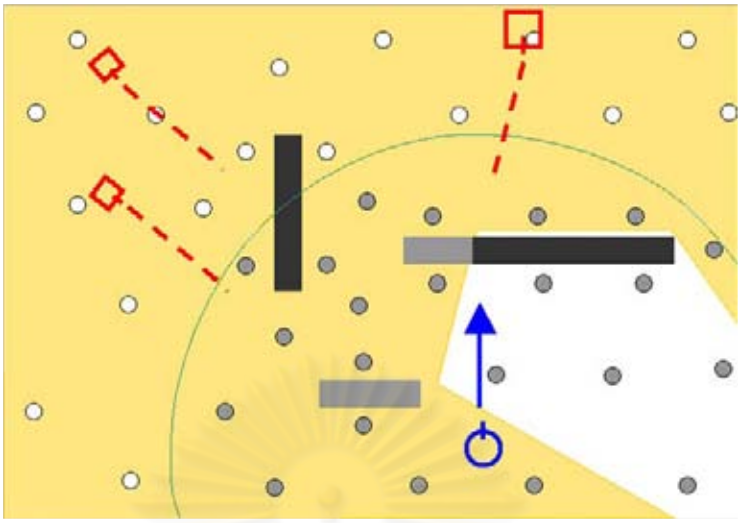
รูปที่ 10 แสดงภาพตัวอย่างจากเกม NeverWinter Nights [14] [15]

ไดนามิคสคริปต์ เป็นงานปรับปรุงค่าเพื่อถ่วงน้ำหนักพฤติกรรมของปัญญาประดิษฐ์แบบอัตโนมัติในระหว่างการเล่นที่ได้รับการกล่าวถึงจากหลายงานวิจัย โดยระบบจะทำการเรียนรู้ว่าระหว่างพฤติกรรมที่น่าจะใช้ได้ที่ผู้ออกแบบได้เลือกไว้จำนวนหนึ่ง พฤติกรรมใดใช้ได้ผลดีที่สุดในระหว่างการเล่นจริง สำหรับงานวิทยานิพนธ์นี้ มีการปรับปรุงพฤติกรรมของฝ่ายปัญญาประดิษฐ์ระหว่างการเล่นเช่นกัน แต่ความได้ผลของพฤติกรรมนั้นคำนวณได้จากอีวาลูเอชัน ฟังก์ชันโดยตรง ดังนั้นจึงไม่ต้องถ่วงน้ำหนักพฤติกรรมที่ใช้ได้ผลดี การปรับปรุงพฤติกรรมของปัญญาประดิษฐ์จะเน้นไปที่การเลือกพฤติกรรมของปัญญาประดิษฐ์ที่มีผลทำให้ปัญญาประดิษฐ์สู้ได้สูสีกับผู้เล่น

Ramco Straatman และคณะ [19] ได้สร้างปัญญาประดิษฐ์ที่มีพฤติกรรมกลยุทธการต่อสู้แบบไดนามิค (dynamic procedural combat tactics' AI) สำหรับเกม Killzone (แสดงในรูปที่ 11) ซึ่งเป็นเกมยิงในมุมมองบุคคลที่หนึ่ง ซึ่งได้มีการคำนวณตำแหน่งในการตัดสินใจเคลื่อนที่โดยใช้การใช้อีวาลูเอชัน ฟังก์ชันคล้ายกับเกมกระดาน ดังเช่น เกมหมากรุก แต่เนื่องจากไม่สามารถทำนายการกระทำของศัตรูล่วงหน้าได้ จึงใช้แนวการยิงของปืนมาเป็นตัวช่วยในการคำนวณแทน สำหรับการเคลื่อนที่ของตัวละครไปยังจุดหมายก็มีการใช้เวย์พ้อยต์ (waypoint) เป็น

แนวทางในการเดิน โดยจะทำการให้คะแนนทุกๆ จุด โดยการใช้ระยะความใกล้หรือไกลกับจุดนั้นๆ แนวการยิงของปืน และระยะในช่วงของการมองเห็นประกอบกัน แล้วทำการเลือกทางเดินไปยังจุดที่มีคะแนนดีที่สุด (แสดงในรูปที่ 12 โดยวงกลมสีน้ำเงินคือตัวละครของเรา สี่เหลี่ยมสีแดงคือตัวละครฝ่ายศัตรู โดยมีแนวการยิงเป็นเส้นประด้านหน้า รัศมีการมองเห็นของตัวละครคือเส้นวงกลมสีเขียว และจุดต่างๆ คือเวย์พ้อยต์ (waypoint) ในแผนที่ โดยจุดสีเทาคือ เวย์พ้อยต์ที่อยู่ในระยะมองเห็น และจุดสีขาวคือเวย์พ้อยต์ที่อยู่นอกระยะการมองเห็นของเรา) ซึ่งการคำนวณนี้จะทำการคำนวณอยู่ตลอดในขณะที่เคลื่อนที่ไปยังจุดต่างๆ ในแผนที่

รูปที่ 11 แสดงภาพตัวอย่างของเกม Killzone [19]



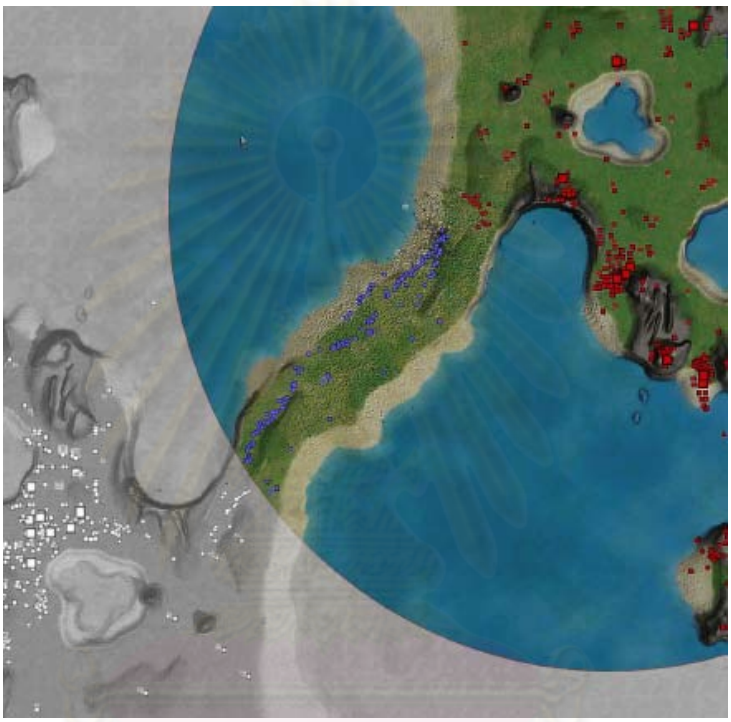
รูปที่ 12 แสดงภาพในมุมมองของการคำนวณทิศทางการเคลื่อนที่ของเกม Killzone [19]

สำหรับระยะทางที่เป็นรัศมีในการคำนวณทิศทางการเดินไปยังเวย์พ้อยต์ต่างๆ ในการทดลองนี้ จะใช้การคำนวณในระยะระหว่าง 5 เมตร ถึง 10 เมตรของระยะทางของเกมจากตัวละครออกไป ซึ่งจากผลการวิจัยนี้พบว่า ถ้าหากระยะรัศมีที่ใช้คำนวณนี้น้อยเกินไป ก็จะทำให้เราต้องสู้กับศัตรูในตำแหน่งที่เสียเปรียบ ซึ่งถ้ายิ่งระยะรัศมีนี้ยิ่งมากก็จะทำให้สามารถเคลื่อนที่ไปยังจุดหมายได้อย่างได้เปรียบฝ่ายตรงข้ามมากขึ้น

ซึ่งแนวคิดในงานวิจัยนี้มีส่วนคล้ายกับงานวิทยานิพนธ์ในส่วนที่มีการนำอีวาลูเอชันฟังก์ชันมาช่วยในการคำนวณ โดยการนำระยะห่างระหว่างศัตรูมาใช้ร่วมในการคิดด้วย แต่ในงานวิจัยนี้จะคิดระยะแค่ในส่วนรัศมีสั้นๆ ออกจากตัวละครเท่าที่ได้กำหนดไว้ไม่ได้ครอบคลุมทั้งหมดจากทั้งแผนที่ ต่างจากวิทยานิพนธ์นี้ที่จะคำนวณระยะทางจากการดูแผนที่ทั้งหมดในเกม เนื่องจากเป็นเกมคนละประเภทกัน และงานวิทยานิพนธ์นี้ยังไม่นำอีวาลูเอชัน ฟังก์ชันไปใช้โดยตรงด้วย เนื่องจากต้องปรับความสูสีของเกมเสียก่อน

Sender Bakkes และคณะ [2] ได้ทำการสร้างอีวาลูเอชัน ฟังก์ชันจากข้อมูลที่บันทึกไว้จาก Spring (แสดงในรูปที่ 13) ซึ่งเป็นเกมวางแผนการรบแบบเวลาจริงเกมหนึ่ง โดยทำการบันทึกสภาพการเล่น ณ เวลาต่างๆ ซึ่งสิ่งที่บันทึกไว้มีอยู่ทั้งหมด 5 อย่างด้วยกัน คือ จำนวนของกลุ่มตัวละครแต่ละประเภท จำนวนของกลุ่มตัวละครศัตรูที่อยู่ใกล้ตัวผู้บังคับบัญชาฝ่ายเราในรัศมี 2000 จำนวนของกลุ่มศัตรูที่อยู่ใกล้ตัวผู้บังคับบัญชาฝ่ายเราในรัศมี 1000 จำนวนของกลุ่มตัวละครศัตรูที่อยู่ใกล้ตัวผู้บังคับบัญชาฝ่ายเราในรัศมี 500 และร้อยละของพื้นที่ที่ฝ่ายเราเห็นเมื่อเทียบกับพื้นที่

ของทั้งแผนที่ของเกม ซึ่งข้อมูลที่ได้บันทึกไว้เหล่านี้ได้นำมาใช้ในการสร้างอีวาลูเอชัน ฟังก์ชัน โดยนำมาคิดรวมกับค่าน้ำหนักของกลุ่มตัวละครแต่ละประเภท และค่าน้ำหนักที่คิดจากระยะทางจากตัวผู้บังคับบัญชาของฝ่ายตรงข้าม และนอกจากนั้นจะมีการปรับค่าน้ำหนักของข้อมูลทั้ง 5 อย่างที่ได้บันทึกไว้โดยการให้ทำการเรียนรู้ด้วยวิธี Temporal Difference (TD-Learning)



รูปที่ 13 แสดงภาพตัวอย่างเกม Spring [2]

สำหรับในการวัดผลว่าอีวาลูเอชัน ฟังก์ชันนั้นใช้บอกค่าสถานะของเกมได้ดีหรือไม่นั้น ทำการวัดจากค่าหลายๆ อย่าง เช่น ร้อยละของจำนวนเกมที่มีอีวาลูเอชัน ฟังก์ชันสามารถทำนายผลของเกมได้ถูกต้อง ซึ่งทำโดยการใช้ฟังก์ชันคำนวณตอนใกล้จบเกม เป็นต้น

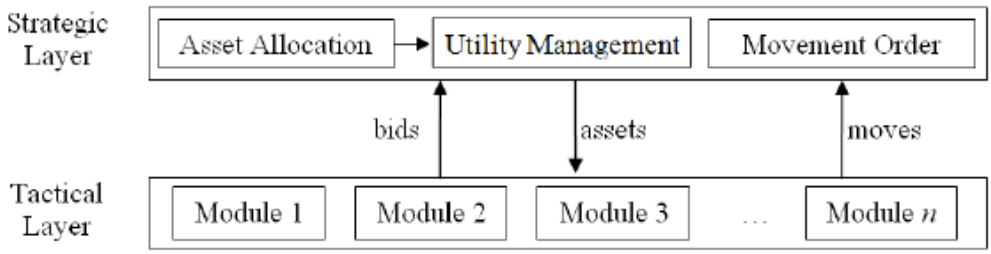
ซึ่งในส่วนของการใช้อีวาลูเอชัน ฟังก์ชันจากข้อมูลที่บันทึกไว้ถึงสภาพต่างๆ ในขณะนั้นของเกมโดยอ้างอิงจากทั้งปัจจัยของทางกลุ่มตัวละครฝ่ายเรา และกลุ่มตัวละครฝ่ายศัตรูนั้น ได้นำไปดัดแปลงประยุกต์ใช้กับอีวาลูเอชัน ฟังก์ชันของวิทยานิพนธ์นี้

Maurice Bergsma และคณะ [3] [4] ได้ทำการนำเสนอสถาปัตยกรรมทางด้านปัญญาประดิษฐ์ที่ได้ให้ชื่อว่า ADAPTA (Allocation and Decomposition Architecture for

Performing Tactical AI) ซึ่ง Bergsma และคณะได้ทำการทดลองสถาปัตยกรรมนี้กับเกมวางแผนแบบผลัดกันเล่นที่สร้างขึ้นเองชื่อว่า Simple wars (แสดงในรูปที่ 14) โดยยึดรูปแบบและลักษณะการเล่นตามแบบของเกม Advance wars

รูปที่ 14 แสดงภาพเกม simple wars [3] [4]

สำหรับลักษณะของสถาปัตยกรรมนี้จะมีการแยกงานในการคำนวณย่อยๆ ต่างๆ ออกเป็น 2 ส่วนคือ ส่วนของยุทธวิธี (Strategy Layer) และส่วนของกลวิธี (Tactic Layer) ซึ่งในส่วนของยุทธวิธีนั้นจะมีงานอยู่ 3 อย่างคือ งานกำหนดค่าของทรัพยากรต่างๆ (Asset Allocation) งานจัดการกับทรัพยากรต่างๆ (Utility Management) และงานออกคำสั่งในการกระทำ (Movement Order) สำหรับส่วนของกลวิธีนั้นจะแบ่งออกเป็นส่วนย่อยๆ หลายส่วน ซึ่งจะมีหน้าที่หลักๆ อยู่ 3 อย่างคือ Bid Generation ทำหน้าที่สร้างค่าความสำคัญของสิ่งต่างๆ แล้วจึงส่งค่ากลับไปให้กับส่วนของยุทธวิธี Utility Calculation ทำหน้าที่คำนวณค่าทรัพยากรต่างๆ ที่ได้มาจากส่วนของยุทธวิธี และ Action Generation ทำหน้าที่สร้างการกระทำเพื่อให้สามารถเลือกการกระทำที่ดีที่สุดที่จะส่งผลให้สามารถเอาชนะอีกฝ่ายหนึ่งได้ ดังแสดงในรูปที่ 15



รูปที่ 15 แสดงลักษณะของ ADAPTA Game AI architecture [3] [4]

หลังจากนั้น Bergsma และคณะ ได้ทำการทดลองโดยการเน้นไปที่การสร้างส่วนย่อยของส่วนกลวิธีส่วนหนึ่งซึ่งจะทำการจัดการให้เรียนรู้คู่ต่อสู้โดยการใช้การคำนวณจากการปรับค่าน้ำหนักตัวละครประเภทต่างๆ ประกอบกับระยะห่างระหว่างตัวละครเป็นหลักในการคำนวณ ทำการทดลองกับปัญญาประดิษฐ์ที่ได้สร้างขึ้นไว้อยู่แล้ว 4 ประเภทคือ

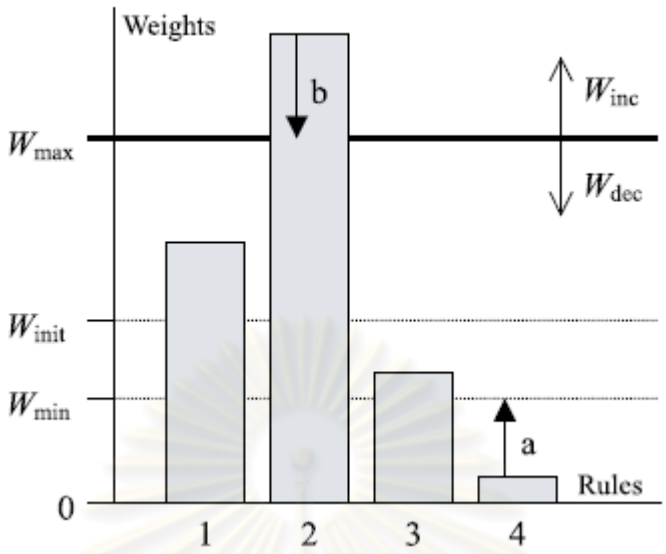
- Rush เป็นประเภทเน้นการโจมตี ซึ่งจะพยายามมุ่งเข้าหา และโจมตีศัตรูทุกประเภทที่อยู่ใกล้ที่สุดโดยไม่คำนึงถึงสิ่งใด
- Defence เป็นประเภทเน้นการตั้งรับ ซึ่งจะคอยป้องกันตัวเองอยู่ที่ฐาน และทำการโจมตีในเฉพาะกรณีที่จำเป็นเท่านั้น
- Base Offence เป็นประเภทเน้นการโจมตีฐาน ซึ่งจะพยายามเข้าหาฐานที่อยู่ใกล้ที่สุด และโจมตีตัวละครที่ขวางทางอยู่ในระยะการเดิน
- Unit Offence เป็นประเภทเน้นการโจมตีตัวละครซึ่งจะพยายามเข้าโจมตีตัวละครที่อยู่ใกล้ที่สุดก่อน

พบว่าการพัฒนาปัญญาประดิษฐ์ดังกล่าวนี้ด้วยสถาปัตยกรรม ADAPTA นั้น สามารถทำการสร้างกลวิธีที่จะเอาชนะศัตรูได้ ซึ่งงานวิทยานิพนธ์นี้มีการใช้อีวาลูเอชัน ฟังก์ชันมาทำในส่วนของการคำนวณค่าเช่นเดียวกัน แต่จะไม่ได้นำในเรื่องของการแยกส่วนระหว่างส่วนยุทธวิธี และส่วนกลวิธีดังเช่นในงานวิจัยนี้มาใช้ นอกจากนี้วิธีการทดลองโดยการนำปัญญาประดิษฐ์ในรูปแบบกลวิธีต่างๆ ที่แตกต่างกันมาใช้ในการทดลองสู้กับปัญญาประดิษฐ์ที่สร้างขึ้น ก็เป็นแนวทางในการสร้างปัญญาประดิษฐ์ที่ใช้กลยุทธ์การเล่นในแบบต่างๆ เพื่อทดลองกับปัญญาประดิษฐ์ที่พัฒนาขึ้นในวิทยานิพนธ์นี้

2.2.2 งานวิจัยที่พัฒนาปัญญาประดิษฐ์ให้มีความสามารถใกล้เคียงผู้เล่น

Pieter Spronck และคณะ [12] [13] ได้ทำการนำเสนอปัญญาประดิษฐ์สำหรับเกม Computer RolePlaying Games (CRPGs) ที่จะทำการเล่นให้มีความสูสีกับผู้เล่นโดยการใช้ไดนามิคสคริปต์ ประกอบกับการจัดมาตราส่วนความยาก (Difficulty Scaling) ซึ่งในส่วนของการจัดมาตราส่วนความยากนี้เขาได้เสนอวิธีการคิดอยู่ 3 ลักษณะคือ

- High-Fitness Penalising เป็นการปรับค่าน้ำหนักโดยการเพิ่ม หรือลดค่าฟิตเนส (fitness) ในอัตราส่วนที่คงที่อัตราหนึ่ง โดยการเพิ่มหรือลดนั้นจะคิดตามการแพ้หรือชนะหลังจากการต่อสู้แต่ละครั้ง
- Weight Clipping เป็นการปรับค่าน้ำหนักโดยตรง โดยจะทำการกำหนดค่าน้ำหนักเริ่มต้น สูงสุด และต่ำสุดไว้เป็นขอบเขตของการเพิ่มหรือลดนั้นๆ แล้วหลังจากการต่อสู้แต่ละครั้ง จะทำการเพิ่มหรือลดค่าน้ำหนักตามเป็นจำนวนหนึ่งตามค่าที่กำหนดไว้ โดยการเพิ่มหรือลดค่าน้ำหนักจะไม่สามารถเกินขอบเขตที่กำหนดนั้นไว้ได้ (แสดงในรูปที่ 16)
- Top-culling จะมีลักษณะเป็นการปรับค่าน้ำหนักคล้ายกับวิธี Weight Clipping แต่จะยินยอมค่าน้ำหนักที่ปรับเพิ่มขึ้นนั้นสามารถเกินค่าน้ำหนักสูงสุดที่กำหนดไว้ได้ แต่หากค่าน้ำหนักนั้นเกินค่าสูงสุดที่กำหนดไว้แล้ว จะไม่ทำการเลือกข้อมูลส่วนนั้นมาใช้จนกว่าจะถูกปรับลงให้อยู่ในช่วงของค่าสูงสุดที่ได้กำหนดไว้ (แสดงในรูปที่ 16)



รูปที่ 16 แสดงลักษณะของการคิด weight clipping และ top-culling [12] [13]

สำหรับส่วนของการทดลอง เขาได้ทำการทดลองโดยการใช้ปัญญาประดิษฐ์ที่ใช้เฉพาะไดนามิคสคริปต์ และไดนามิคสคริปต์ที่มีการจัดมาตราส่วนความยากในแต่ละรูปแบบเพื่อปรับค่าน้ำหนักในการกระทำของตัวละครในรูปแบบการต่อสู้ต่างๆ ทดสอบกับปัญญาประดิษฐ์มาตรฐานที่เน้นกลยุทธการต่อสู้ในรูปแบบต่างๆ เช่น รูปแบบที่เน้นการโจมตีฝ่ายตรงข้าม รูปแบบที่เน้นการป้องกัน รูปแบบที่เน้นการใช้คำสาป เป็นต้น ซึ่งจากผลการทดลองนั้น การใช้ไดนามิคสคริปต์ร่วมกับการจัดมาตราส่วนความยากแบบ Top-culling นั้นให้ผลออกมาสูสีกับฝ่ายตรงข้ามมากที่สุดโดยจำนวนร้อยละของการแพ้ชนะเฉลี่ยแล้วดีที่สุดเมื่อเทียบกับวิธีอื่นๆ

ซึ่งสำหรับงานวิทยานิพนธ์นี้ได้มีความต้องการให้ปัญญาประดิษฐ์สามารถทำการเล่นให้สูสีกับฝ่ายตรงข้ามเช่นเดียวกัน แต่จะมีการคิดอีวาลูเอชัน ฟังก์ชันในทุกช่วงของการต่อสู้ แล้วนำอีวาลูเอชัน ฟังก์ชัน มาใช้เพื่อการปรับสภาพความสูสีโดยไม่ต้องปรับน้ำหนักของสคริปต์ เพื่อให้ได้สภาพของความสูสีในระหว่างการเล่นที่ชัดเจนมากขึ้น

บทที่ 3

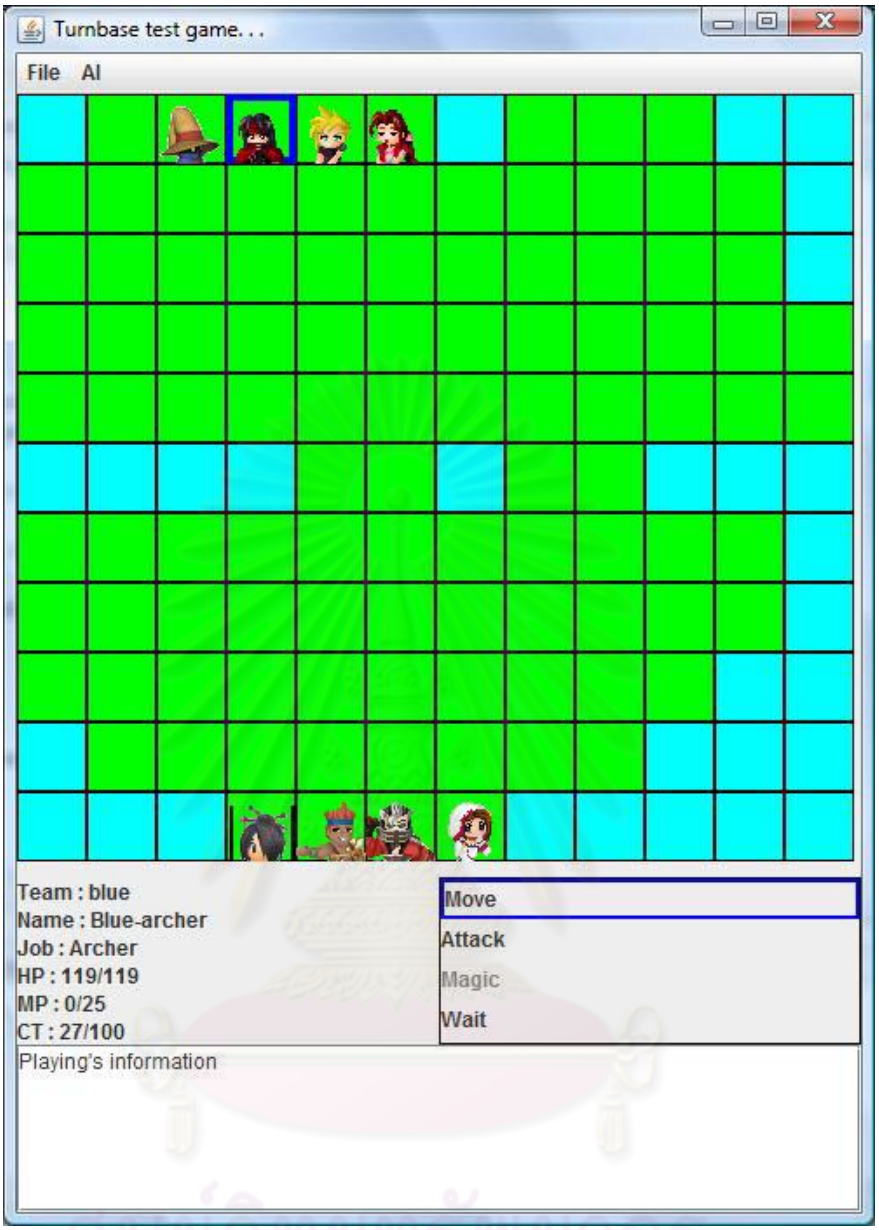
เกมที่ใช้ในการทำการวิจัย

3.1 ลักษณะของเกมที่ใช้ในการวิจัย

ลักษณะของเกมที่ใช้ในการวิจัยนี้ เป็นลักษณะของการวางแผนการรบแบบผลัดกันเล่น (Turn-Based Strategy) โดยกฎกติกาส่วนใหญ่ของเกมประเภทนี้คือ วางแผนอย่างไรให้สามารถเอาชนะอีกฝ่ายหนึ่งได้ โดยเงื่อนไขของการได้รับชัยชนะของแต่ละเกมมีลักษณะแตกต่างกันออกไป

สำหรับเกมที่ได้ใช้ในการวิจัยนี้ พัฒนาขึ้นโดยการใช้กฎ กติกา และเงื่อนไขการเล่นในการเอาชนะฝ่ายตรงข้าม โดยอ้างอิงจากเกมตระกูล Final Fantasy Tactics เป็นหลัก ซึ่งเกมที่ใช้ในการทดลองนั้นจะมีลักษณะดังรูปที่ 17

สำหรับการเอาชนะฝ่ายตรงข้าม เกิดจากการที่ทำเงื่อนไขที่กำหนดให้สำเร็จ โดยเกมที่ใช้ในการทดลองนี้ได้ใช้เงื่อนไขที่พบมากที่สุดสำหรับเกมประเภทนี้ คือ การทำให้ตัวละครฝ่ายตรงข้ามทุกตัวหมดสภาพการต่อสู้ หรือตายนั่นเอง



รูปที่ 17 แสดงลักษณะของเกมที่ใช้ในการทดลอง

3.2 กฎกติกาต่าง ๆ ในเกม

รายละเอียด และกฎกติกาต่างๆ รวมถึงวิธีการคิดค่าต่างๆ ภายในเกมที่ใช้ในการทดลองนี้มีดังต่อไปนี้

1. แผนที่สำหรับใช้ในการรบแต่ละครั้ง จะมีลักษณะเป็นพื้นที่ขนาดใหญ่ และแบ่งออกเป็นพื้นที่ย่อยๆ เป็นตารางสำหรับใช้ในการเคลื่อนที่ตัวละคร ซึ่งจะคล้ายกับเกมหมากรุก แต่ไม่มีการจำกัดขนาดของแผนที่ และแต่ละจุดของแผนที่จะแบ่งเป็น 2 ส่วน คือ ส่วนที่ตัวละครสามารถเคลื่อนที่ไป ณ ตำแหน่งนั้นได้ และไม่สามารถเคลื่อนที่ไป ณ ตำแหน่งนั้นได้

นอกจากนี้ ยังไม่มีการจำกัดจุดสำหรับการวางตัวละครของแต่ละฝ่ายว่าต้องเป็นตำแหน่งใดในบนแผนที่ แต่จะมีจุดวางตัวละครที่เปลี่ยนไปได้แล้วแต่ตัวเกมจะเป็นตัวกำหนด ซึ่งส่วนใหญ่จะเป็นคนละด้านของแผนที่เป็นหลัก แต่อาจมีบางครั้งที่ต้องกระจายแยกเป็นสองทีม หรือกระจายกันไปตามส่วนต่างๆ ของแผนที่แล้วแต่เป้าหมายของเกมว่าต้องการให้กระทำสิ่งใด เช่น การแบ่งกองกำลังเป็นสองด้านเพื่อบุกโจมตีในสองจุดพร้อมกัน สำหรับเกมที่ใช้ในการทดลองนี้ ได้ทำการกำหนดตำแหน่งเริ่มต้นของแต่ละฝ่ายเป็นตำแหน่งคงที่ โดยอ้างอิงจากเกม Final Fantasy Tactics A2: Grimoire of the Rift

2. ค่าสถานะของตัวละครที่ใช้ในเกมนี้ มีสำหรับให้ผู้เล่นสามารถคำนวณค่าความสามารถของตัวละครในการกระทำสิ่งต่างๆ เช่น ระยะทางสำหรับเคลื่อนที่ ความรุนแรงในการโจมตี หรือความสามารถในการป้องกัน เป็นต้น ซึ่งสามารถจำแนกออกได้ดังนี้

2.1 Name คือ ชื่อของตัวละคร

2.2 Team ระบุว่าตัวละครแต่ละตัวนั้นอยู่ฝ่ายใด

2.3 Job ระบุว่าตัวละครนั้นมีอาชีพใด ซึ่งมีผลทำให้ตัวละครมีความสามารถแตกต่างกัน

2.4 Hp เป็นค่าพลังชีวิตของตัวละคร หากพลังชีวิตหมดลง มีผลทำให้ตัวละครไม่สามารถกระทำการใดๆ ได้ (หรือจะเรียกได้ว่าตัวละครนั้นตายนั่นเอง)

2.5 Mp เป็นค่าพลังเวทมนต์ของตัวละคร เพื่อให้สามารถใช้เวทมนต์ได้ หากมีค่านี้ไม่เพียงพอ ก็จะมีผลทำให้ไม่สามารถใช้เวทมนต์นั้นได้

2.6 Move (Mov) เป็นค่าที่ใช้สำหรับกำหนดระยะการเคลื่อนที่ของตัวละคร

2.7 Attack (Atk) เป็นค่าที่แสดงถึงพลังโจมตีด้านกายภาพของตัวละคร หากมีค่านี้มาก ก็จะมีผลทำให้สร้างความเสียหายด้านกายภาพได้มากขึ้น

2.8 Defense (Def) เป็นค่าที่แสดงถึงพลังป้องกันจากการได้รับความเสียหายด้านกายภาพ หากมีค่านี้มาก จะยิ่งทำให้สามารถทนต่อการโจมตีด้านกายภาพได้มากขึ้น และมีความเสียหายทางกายภาพลดลง

2.9 Magic (Mag) เป็นค่าที่แสดงถึงพลังโจมตีด้านเวทมนต์ของตัวละคร หากมีค่านี้มาก ก็จะมีผลทำให้สร้างความเสียหายด้านเวทมนต์ให้กับเป้าหมายได้มากขึ้น ส่วนตัวละครที่มีเวทมนต์ฟื้นพลังชีวิต ก็จะสามารถฟื้นพลังชีวิตให้ตัวละครทีมเดียวกันได้มากขึ้นเช่นกัน

2.10 Magic Resistance (Res) เป็นค่าที่แสดงถึงความต้านทานความเสียหายจากด้านเวทมนต์ ซึ่งยิ่งมีค่านี้มาก ก็จะยิ่งทำให้สามารถทนต่อการโจมตีด้านเวทมนต์ได้มากขึ้น ทำให้ได้รับความเสียหายด้านเวทมนต์น้อยลง

2.11 Evade (Eva) เป็นค่าที่แสดงถึงการหลบหลีก ซึ่งมีผลทำให้หลบหลีกการโจมตีด้านกายภาพของศัตรู ยิ่งหากมีค่านี้มากจะทำให้มีโอกาสหลบการโจมตีด้านกายภาพของศัตรูได้มากขึ้น

2.12 Speed (Spd) เป็นค่าความเร็วของตัวละคร ซึ่งมีผลทำให้ตัวละครสามารถเข้าสู่รอบเคลื่อนที่ของตัวเองได้เร็วขึ้น

2.13 Character's turn (Ct) เป็นค่าที่ใช้ทดเพื่อคำนวณว่าตัวละครนั้นจะสามารถถึงตาเดินได้เมื่อไร

3. อาชีพในเกม จะเป็นตัวแบ่งลักษณะรูปแบบของการโจมตี และส่วนประกอบต่างๆ ของตัวละคร แบ่งออกเป็นอาชีพโดยพื้นฐานโดยทั่วไปของเกมในลักษณะนี้ 4 อาชีพ คือ

3.1 Warrior เป็นอาชีพที่เน้นด้านการโจมตีทางกายภาพ โดยจะสามารถทำการโจมตีด้านกายภาพได้รุนแรงที่สุดหากเทียบกับอาชีพอื่นๆ และมีพลังป้องกันด้านกายภาพมากสุดเช่นกัน แต่ข้อเสียคือ สามารถโจมตีได้เพียงระยะประชิดตัว หรือระยะ 1 ช่องที่ติดอยู่กับตัวละครเท่านั้น

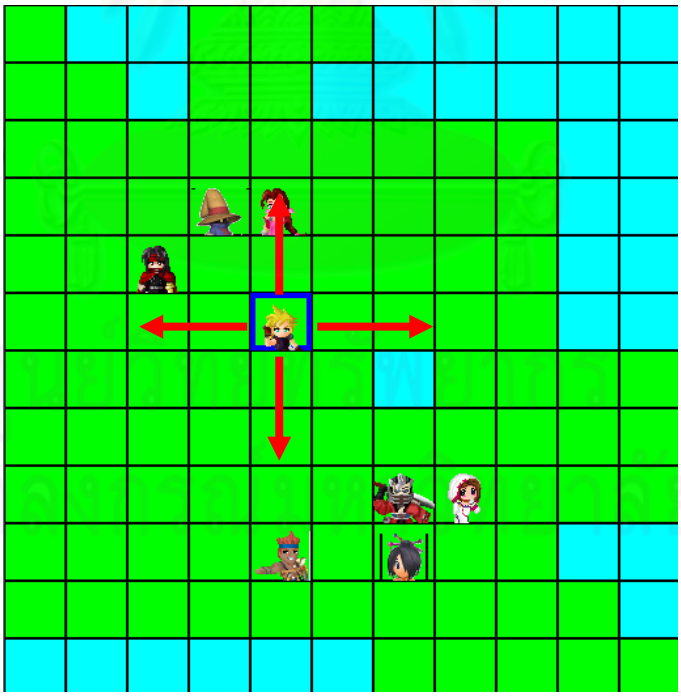
3.2 Archer เป็นอาชีพที่เน้นการโจมตีด้านกายภาพ คล้ายกับ Warrior แต่จะโจมตีได้เบากว่า และมีพลังป้องกันที่น้อยกว่า แต่ข้อดีคือ สามารถโจมตีในระยะไกลได้

3.3 Black Mage เป็นอาชีพที่เน้นการโจมตีด้านเวทมนต์เป็นหลัก โดยการใช้ค่า Mp เพื่อแลกกับการโจมตีศัตรูโดยใช้เวทมนต์ สามารถโจมตีได้ในระยะไกล และมีความ

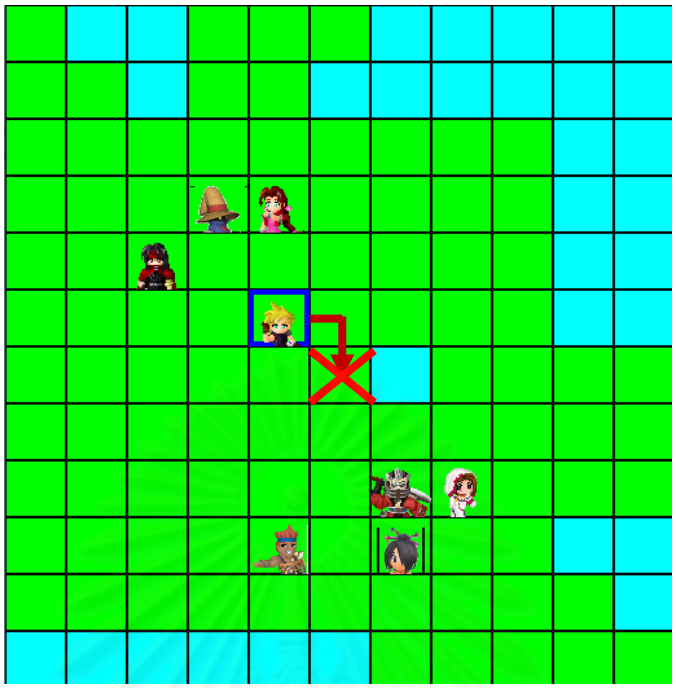
รุนแรง แต่การโจมตีด้านกายภาพจะเบา และจำกัดแค่ในระยะประชิดเท่านั้น รวมถึงยังมีพลังป้องกันที่น้อยอีกด้วย

3.4 White Mage เป็นอาชีพที่สามารถช่วยฟื้นพลังชีวิต หรือค่า Hp ให้กับตัวละคร แต่มีพลังโจมตีและป้องกันด้านกายภาพค่อนข้างต่ำ

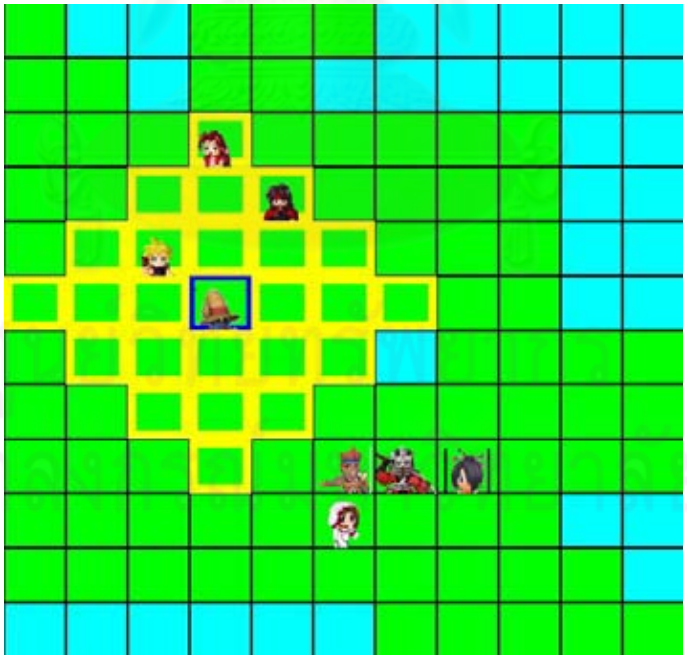
4. ลักษณะการเคลื่อนที่ของตัวละคร ทุกตัวละครจะมีลักษณะการเคลื่อนที่เช่นเดียวกัน คือ สามารถเคลื่อนไปได้ในทิศทางตามแนวตั้ง หรือแนวนอน เพื่อไปตามจุดต่างๆ บนแผนที่ตามความต้องการดังในรูปที่ 18 แต่จะไม่สามารถเดินในแนวทแยงได้ ดังนั้นหากต้องการเดินไปในช่องที่เป็นช่องทแยงมุมดังเช่นในรูปที่ 19 เราจะต้องเสียค่าเคลื่อนที่จำนวน 2 หน่วย เพื่อไปยังตำแหน่งดังกล่าว ซึ่งระยะทางที่ตัวละครสามารถเดินได้นั้น จะขึ้นอยู่กับค่า Move ตามความสามารถของตัวละคร ตัวอย่างเช่น จากรูปที่ 20 ตัวละครที่อยู่ในกรอบสีน้ำเงินมีค่า Move เท่ากับ 3 ดังนั้นระยะทางที่ตัวละครสามารถเคลื่อนที่ได้คือ ระยะไม่เกิน 3 ช่องนับจากตำแหน่งนั้น (ช่องที่มีกรอบสีเหลืองในรูป)



รูปที่ 18 แสดงลักษณะการเดินของตัวละครในเกมที่ทำการทดลอง



รูปที่ 19 แสดงลักษณะการเดินของตัวละครหากต้องการเดินไปยังช่องในแนวทแยงมุม



รูปที่ 20 แสดงขอบเขตการเดินของตัวละครที่อยู่ในกรอบสีน้ำเงิน

5. การกระทำต่างๆ ที่ตัวละครสามารถทำได้ในเกม จะมีลักษณะเป็นคำสั่งให้เลือกโดยแบ่งได้ออกเป็น

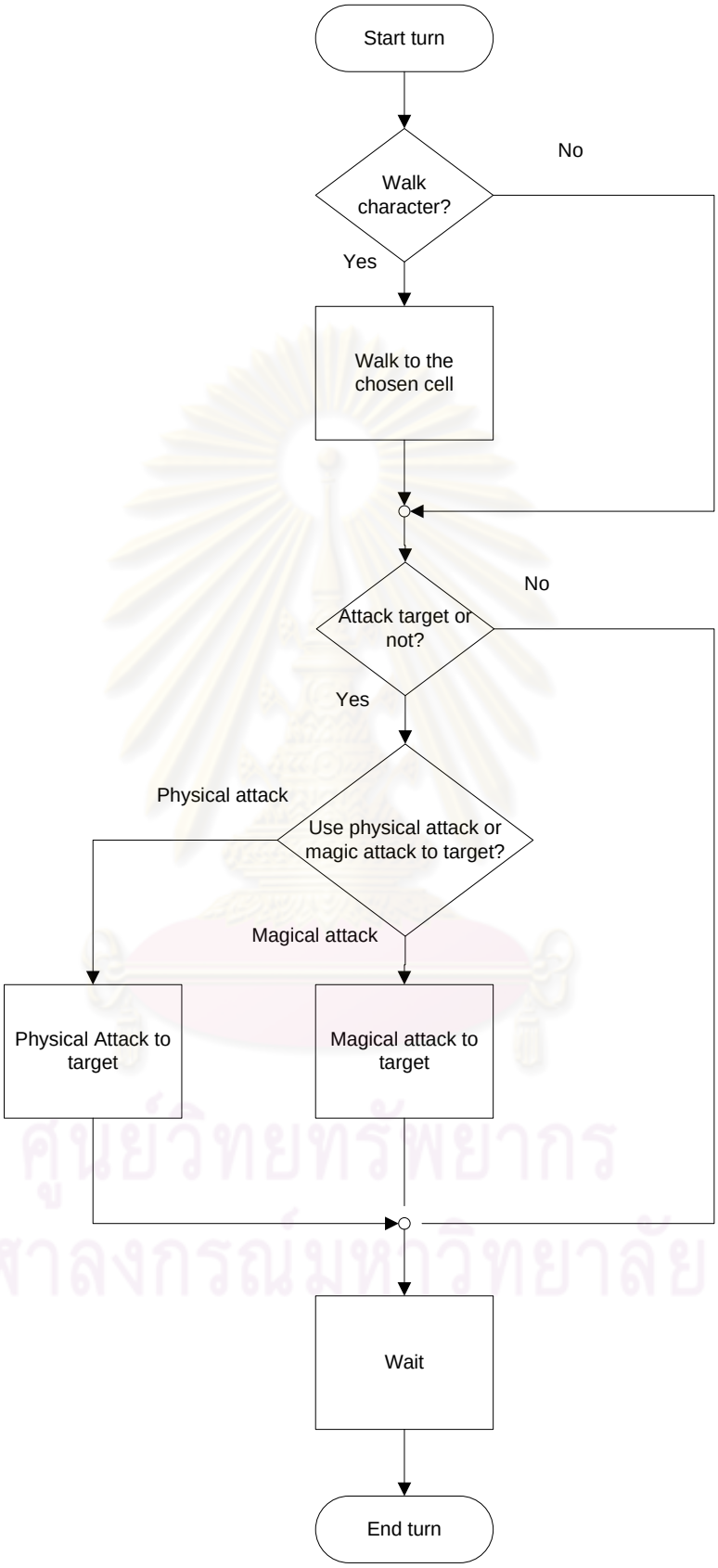
5.1 Move จะเป็นคำสั่งเพื่อให้ตัวละครสามารถเดินไปยังจุดต่างๆ บนแผนที่ได้

5.2 Attack จะเป็นคำสั่งเพื่อให้ตัวละครสามารถทำการโจมตีด้านกายภาพไปที่ตัวละครอื่นได้ โดยระยะที่สามารถโจมตีได้นั้นก็จะขึ้นอยู่ค่าระยะโจมตีของตัวละคร

5.3 Magic จะเป็นคำสั่งเพื่อให้ตัวละครสามารถทำการโจมตีด้านเวทมนต์ใส่ตัวละครอื่นได้ โดยคำสั่งนี้จะสามารถใช้ได้ก็ต่อเมื่อค่า Mp ของตัวละครมีพอให้สามารถใช้การโจมตีทางเวทมนต์ด้วย

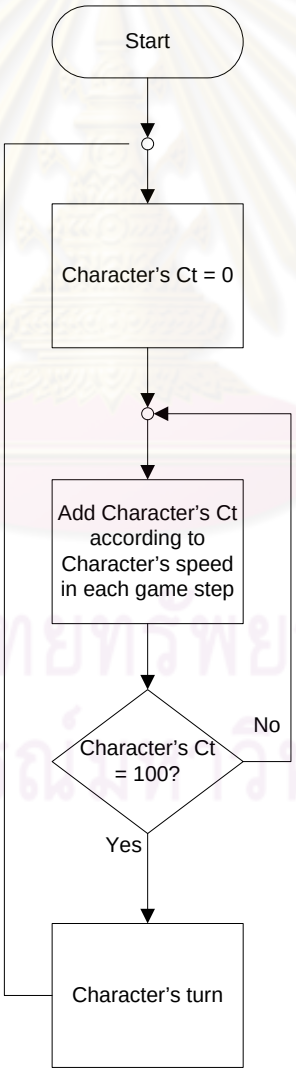
5.4 Wait จะเป็นคำสั่งเพื่อให้ตัวละครหยุดการกระทำ หรือจบการเดินในตานั้น

6. สำหรับลำดับในการกระทำต่างๆ นั้น จะมีลักษณะคือ เมื่อถึงตาเดินของตัวละคร ตัวละครดังกล่าวจะสามารถเลือกได้ว่าจะทำการเดิน โจมตีตัวละครอื่น หรือไม่กระทำสิ่งใดๆ เลยได้ โดยที่ตัวละครจะสามารถทำการเดินก่อนแล้วจึงทำการโจมตีได้ แต่หากเลือกทำการโจมตีแล้วจะไม่สามารถเดินไปที่ใดได้อีก ดังเช่นแสดงในรูปที่ 21 โดยการที่จะโจมตีศัตรูนั้น เลือกได้เพียงอย่างเดียวว่าจะโจมตีทางกายภาพ หรือโจมตีทางเวทมนต์ โดยที่ไม่สามารถกระทำทั้งสองอย่างพร้อมกันในเวลาเดียวกันได้



รูปที่ 21 แสดงรูปวิธีการเล่นในหนึ่งตาเดินตัวละคร

7. การคำนวณตาเดินของตัวละคร จะใช้ค่า speed ของตัวละครเป็นหลักในการคิด โดยแต่ละตัวละครจะมีค่า Character's Turn (Ct) อยู่ เมื่อเริ่มเกมค่า Ct ของทุกตัวละครจะมีค่าเป็น 0 (บางกรณีในเกมจริงบางสนามที่มีเหตุการณ์ที่ตัวละครฝ่ายตรงข้ามจะต้องได้เดินก่อน ฝ่ายตรงข้ามจะได้ค่า Ct ที่สูงกว่าตั้งแต่แรกทำให้มีโอกาสได้เดินก่อน ตามเหตุการณ์ที่ตัวเกมกำหนดไว้) จากนั้นจึงบวกทบค่า Ct ของแต่ละตัวละครเข้าไปในทุกๆ ตาเดินของตัวเกม โดยเมื่อตัวละครใดมีค่า Ct เต็ม (โดยทั่วไปคือค่า 100) ก็จะเท่ากับว่าถึงตาเดินของตัวละครนั้น และสามารถเลือกเดินหรือโจมตีตัวละครอื่นได้ เมื่อตัวละครจบตาเดินของตนเองค่า Ct ของตัวละครก็จะกลับไปเป็นค่า 0 และเริ่มเก็บสะสมใหม่ในแต่ละตาเดินของเกมดังรูปที่ 22



รูปที่ 22 แสดงการคำนวณตาเดินของตัวละคร

8. การคำนวณค่าความเสียหายต่างๆ ที่เกิดขึ้นภายในเกมนั้น แต่ละเกมจะมีค่าเหล่านี้แตกต่างกันไป แล้วแต่เกมนั้นๆจะกำหนด ซึ่งการคำนวณค่าความเสียหายต่างๆ ในเกมที่ได้ใช้ในการทดลองนี้ เกิดจากสมการต่างๆ ดังจะเสนอต่อไปนี้ ซึ่งค่าต่างๆนี้ได้มาจากการทดลองแล้วได้ค่าจากการสังเกตที่ใกล้เคียงกับค่าที่ได้เกิดขึ้นในเกม Final Fantasy Tactics A2: Grimoire of the Rift

สำหรับการโจมตีด้านกายภาพ ค่าความเสียหายที่สามารถโจมตีใส่ศัตรูได้ สามารถคำนวณได้จากสมการดังนี้

$$Damage = Attack - (0.5 \times Target's\ Defense) \quad (3.1)$$

โดยที่ Damage คือ ค่าความเสียหายที่เกิดขึ้น

Attack คือ ค่าพลังโจมตีด้านกายภาพของตัวละคร

Target's Defense คือ ค่าพลังป้องกันด้านกายภาพของตัวละครฝ่ายที่ถูกโจมตี

โดยส่วนของการโจมตีนี้ตัวละครของฝ่ายตรงข้ามที่ถูกโจมตีมีโอกาสที่จะสามารถทำการหลบการโจมตีได้ โดยการคิดค่าร้อยละที่จะหลบได้นั้น อ้างอิงจากส่วนของค่าสถานะ Evade (Eva) ของตัวละครที่ถูกโจมตี ซึ่งตัวละครที่จะทำการโจมตีนั้น จะทำการสุ่มค่ามาค่าหนึ่งในช่วงค่า 0 ถึง 100 หากค่านั้นมีค่า มากกว่าหรือเท่ากับ ค่าร้อยละที่จะหลบได้ ก็จะสามารถทำการโจมตีโดนฝ่ายที่ถูกโจมตีนั่นเอง ซึ่งค่าร้อยละที่จะหลบได้นี้ คิดได้จากสมการดังนี้

$$\%Evade = \frac{Target's\ Evade}{255} \times 100 \quad (3.2)$$

โดยที่ %Evade คือ ค่าร้อยละที่จะหลบได้ของตัวละครฝ่ายที่ถูกโจมตี

Target's evade คือ ค่าการหลบหลีก (Eva) ของตัวละครที่จะถูกโจมตี

สำหรับการโจมตีด้านเวทมนต์ ค่าความเสียหายที่สามารถโจมตีใส่ศัตรูได้ สามารถคำนวณได้จากสมการดังนี้

$$Damage = (2 \times Magic) - (0.5 \times Target's\ Magic\ Resistance) \quad (3.3)$$

โดยที่ Damage คือ ค่าความเสียหายที่เกิดขึ้น

Magic คือ ค่าพลังโจมตีด้านเวทมนต์ของตัวละคร

Target's Magic Resistance คือ ค่าพลังป้องกันด้านเวทมนต์ของตัวละครฝ่ายที่ถูกโจมตี

บทที่ 4

การพัฒนาปัญญาประดิษฐ์ให้สามารถเล่นได้อย่างสูสีกับผู้เล่น

4.1 แนวคิดของการสร้างปัญญาประดิษฐ์

ในการพัฒนาเพื่อสร้างปัญญาประดิษฐ์ให้สามารถเล่นได้อย่างสูสีกับผู้เล่นนั้น ได้ทำการพัฒนาโดยอาศัยข้อคิดที่ว่า ปัญญาประดิษฐ์นั้นจะต้องไม่เก่งเกินไป หรือด้อยเกินไปสำหรับผู้เล่น ไม่ว่าจะเป็นผู้เล่นที่เริ่มเล่นใหม่ หรือเคยเล่นจนมีความชำนาญกับเกมในลักษณะนี้แล้ว รวมทั้งไม่ว่าผู้เล่นจะเล่นด้วยกลยุทธ์การเล่นแบบใด ก็ควรจะสามารถทำการเล่นได้อย่างสูสีในที่สุด ซึ่งในที่นี้จะไม่ใช่การสู้ในรูปแบบที่ตั้งใจเล่นให้แย่เกินไป หรือดีเกินไป แต่พฤติกรรมโดยรวม หรือลักษณะการเล่นโดยรวมนั้น ควรเป็นไปอย่างเป็นธรรมชาติ ยกตัวอย่างเช่น หากผู้เล่นนั้นเป็นลักษณะเพิ่งเริ่มเล่น ก็ไม่ควรมีการกระทำในรูปแบบที่ว่า ตอนแรกสู้อย่างมีประสิทธิภาพมาก จนผู้เล่นเหลือตัวละครเพียงตัวเดียว แล้วจึงอยู่เฉยๆ ให้ผู้เล่นกระทำตอบโต้ จนผลสุดท้ายเหลือตัวละครหลังจากจบเกมเล็กน้อยให้ดูเหมือนลักษณะสู้ได้อย่างสูสี หรือสู้จนเกือบจะชนะแล้ว แต่เมื่อฝ่ายตนเองยังเหลืออยู่มาก ก็ทำการโจมตีฝ่ายตัวเองให้ตายไปจนเหลือน้อยเทียบกับฝ่ายตรงข้าม เป็นต้น ดังนั้นจึงไม่สามารถใช้การหาตาล่วงหน้าแบบต้นไม้มินิแม็กซ์ (minimax tree) ได้ เพราะต้นไม้มินิแม็กซ์นั้นจะเน้นการทำงานไปที่การรักษาผลสุดท้ายของการเล่น ซึ่งอาจก่อให้เกิดลักษณะที่เป็นการเล่นที่ทำให้ผู้เล่นไม่รู้สึกว่าสามารถเล่นได้อย่างสูสีในระหว่างการเล่นนั่นเอง

ด้วยเหตุนี้ จึงทำให้เกิดแนวทางในการพัฒนาโดยอาศัยคะแนนในระหว่างการเล่น เป็นตัวช่วยในการคาดคะเนความสามารถของแต่ละฝ่าย โดยที่ปัญญาประดิษฐ์ที่สร้างให้สามารถเล่นได้อย่างสูสีกับผู้เล่นนี้ จะอาศัยคะแนนจากการเล่นของฝ่ายตรงข้าม มาคิดร่วมกับคะแนนที่ตนเองจะสามารถทำได้ และพยายามทำการกระทำให้มีคะแนนใกล้เคียงกับผู้เล่นอีกฝ่ายในช่วงการเล่นระยะสั้น เพื่อให้การต่อสู้ออกมาค่อนข้างมีความสูสีกับอีกฝ่ายอย่างสม่ำเสมอตลอดเกม

4.2 การคำนวณเพื่อคิดคะแนนของผู้เล่น

ในการพัฒนาปัญญาประดิษฐ์ให้เล่นได้อย่างสูสีกับผู้เล่นนี้ ได้มีการพัฒนาโดยการสร้างอีวาลูเอชัน ฟังก์ชันหนึ่งขึ้นมาเพื่อเป็นตัววัดสถานการณ์ต่างๆ ในระหว่างการเล่น โดยอีวาลูเอชันฟังก์ชันนี้จะทำหน้าที่คิดคะแนนการเล่นในแต่ละรอบการเดินของตัวผู้เล่นออกมาเพื่อวัดว่า ในการเดินรอบนั้น ผู้เล่นสามารถทำคะแนนได้ดีเท่าใด โดยอีวาลูชัน ฟังก์ชันที่ใช้ นั้นคือ

$$s(x,y,action,c) = \sum_{c_{enemy}}^{enemy} \beta\left(\delta(c_{enemy},x,y), ds_{(action,c_{enemy})}, en_{c_{enemy}}\right) + \sum_{c_{ally}}^{ally} \gamma(\delta(c_{ally},x,y), ds_{(action,c_{ally})}, en_{c_{ally}}) \quad (4.1)$$

โดยที่ s(x,y,action,c) คือ ค่าคะแนนของแต่ละช่องในแผนที่ ตามแนวแกน x และ y โดยการกระทำ action กับตัวละครเป้าหมาย c

enemy คือ เซตของตัวละครที่อยู่ฝ่ายศัตรูทั้งหมด

c_{enemy} คือ ตัวละครที่เป็นฝ่ายศัตรู ซึ่งจะเป็นสมาชิกที่อยู่ในเซตของ enemy

ally คือ เซตของตัวละครที่อยู่ฝ่ายเดียวกันทั้งหมด

c_{ally} คือ ตัวละครที่เป็นฝ่ายเดียวกัน ซึ่งจะเป็นสมาชิกที่อยู่ในเซตของ ally

$\beta(d_c, ds_{(action,c)}, en_c)$ คือ ฟังก์ชันในการคำนวณค่าคะแนนจากการกระทำกับตัวละครฝ่ายศัตรู โดยใช้ค่าคะแนนระยะทาง (d_c) จากตัวละครฝ่ายศัตรู c คะแนนความเสียหาย ($ds_{(action,c)}$) เมื่อกระทำความเสียหายโดยการกระทำ action ให้กับตัวละครเป้าหมาย c และสภาวะแวดล้อม (en) อื่นๆ ที่เกี่ยวข้องกับตัวละคร c นั้น

$\gamma(d_c, ds_{(action,c)}, en_c)$ คือ ฟังก์ชันในการคำนวณค่าคะแนนจากการกระทำกับตัวละครฝ่ายเดียวกัน โดยใช้ค่าคะแนนระยะทาง (d_c) จากตัวละครฝ่ายเดียวกัน c คะแนนความเสียหาย ($ds_{(action,c)}$) เมื่อกระทำความเสียหายโดยการกระทำ action ให้กับตัวละครเป้าหมาย c และสภาวะแวดล้อม (en) ที่เกี่ยวข้องกับตัวละคร c นั้น

$\delta(c,x,y)$ คือ ฟังก์ชันในการคำนวณค่าคะแนนระยะทางจากตัวละคร (c) มายังช่อง (x,y) ในแผนที่

โดยค่าคะแนน s(x,y,action,c) นี้จะสามารถเกิดขึ้นได้หลายค่าในช่อง (x,y) ใดๆ โดยขึ้นอยู่กับว่า สถานการณ์ที่กำลังคิดคะแนนนั้นๆ ได้กระทำ action ใด ใส่ตัวละครเป้าหมาย c ตัวใด ซึ่งตัวละครเป้าหมาย c นี้อาจจะเป็นตัวละครตัวใดตัวหนึ่ง ที่อยู่ในฝ่ายศัตรู (c_{enemy}) หรือ ฝ่ายเดียวกัน (c_{ally}) ก็ได้ ตัวอย่างเช่น หากเดินไปยังช่อง (x_1,y_1) แล้วเลือกกระทำการโจมตีด้านกายภาพใส่ตัวละคร c_1 ก็จะทำให้ได้ค่าคะแนนค่าหนึ่งออกมาที่ช่อง (x_1,y_1) นั้น แต่หากเดินไปยังช่อง (x_1,y_1) แล้วกระทำการโจมตีด้านเวทมนต์ใส่ตัวละคร c_1 ก็จะได้อีกค่าคะแนนหนึ่งออกมา หรือในกรณีทำการโจมตีด้านกายภาพใส่ตัวละคร c_2 เมื่อเดินไปยังช่อง (x_1,y_1) ก็จะทำให้เกิดค่าคะแนนอีกค่าหนึ่งออกมาเช่นเดียวกัน หรือในกรณีที่เมื่อทำการเดินไปยังช่อง (x_1,y_1) แล้วไม่เลือกที่จะกระทำการโจมตีตัวละครใด ก็จะทำให้ได้อีกค่าคะแนนหนึ่งออกมาสำหรับช่อง (x_1,y_1) เช่นกัน

สำหรับสมการ $\beta(d_c, ds_{(action,c)}, en_c)$ นั้นจะมีลักษณะการคิด คือ

$$\beta\left(d_c, ds_{(action,c)}, en_c\right) = d_c + ds_{(action,c)} + en_c \tag{4.2}$$

โดยที่ d_c คือ ค่าคะแนนระยะทางจากตัวละครฝ่ายศัตรู c

$ds_{(action,c)}$ คือ ค่าคะแนนความเสียหาย ds เมื่อกระทำ action กับตัวละครฝ่ายศัตรู c

en_c คือ ค่าคะแนนสภาวะแวดล้อมอื่นๆ ที่เกี่ยวข้องกับตัวละคร c นั้น

โดยในการคำนวณหาค่าจากสมการ $\beta(d_c, ds_{(action,c)}, en_c)$ ในแต่ละครั้งจะเป็นการคำนวณเมื่อคิดกับตัวละครฝ่ายศัตรูหนึ่งตัว ดังนั้นในสมการการคำนวณคะแนนผู้เล่น s(x,y,action,c) นั้นจึงเป็นการคิดค่าผลรวมจากคะแนนของฝ่ายศัตรูทั้งหมดโดยใช้ผลรวมของค่าจาก $\sum_{c_{enemy}}^{enemy} \beta\left(\delta\left(c_{enemy}, x, y\right), ds_{(action,c_{enemy})}, en_{enemy}\right)$ โดยหากตัวละครเป้าหมาย c นั้นไม่ใช่ตัวละครที่กำลังพิจารณาอยู่ในตัวละครฝ่ายศัตรู (c_{enemy}) แล้ว ก็จะทำให้ค่า ds นั้นมีค่าเป็น 0 นั่นเอง

ส่วนสมการ $\gamma(d_c, ds_{(action,c)}, en_c)$ นั้นจะมีลักษณะการคิด คือ

$$\gamma(d_c, ds_{(action,c)}, en_c) = d_c + ds_{(action,c)} + en_c \quad (4.3)$$

โดยที่ d_c คือ ค่าคะแนนระยะทางจากตัวละครฝ่ายเดียวกัน c

$ds_{(action,c)}$ คือ ค่าคะแนนความเสียหาย ds เมื่อกระทำ action กับตัวละครฝ่ายเดียวกัน c

en_c คือ ค่าคะแนนสภาวะแวดล้อมอื่นๆ ที่เกี่ยวข้องกับตัวละคร c นั้น

โดยการคำนวณหาค่าจากสมการ $\gamma(d_c,ds_{(action,c)},en_c)$ นี้จะมีการคำนวณในลักษณะคล้ายกับการคำนวณจากสมการ $\beta(d_c,ds_{(action,c)},en_c)$ แต่จะมีข้อแตกต่างตรงที่จะพิจารณาที่ฝ่ายเดียวกันเป็นหลักนั่นเอง

ซึ่งสมการที่ใช้ในการหาคะแนน s(x,y,action,c) นี้จะช่วยให้สามารถวัดค่าคะแนนโดยอาศัยปัจจัยต่างๆ ในเกมเป็นส่วนประกอบ และยังสามารถนำค่าคะแนนที่ได้นี้ไปช่วยในการพิจารณาในสิ่งต่างๆ อื่นๆ ได้อีก ตัวอย่างเช่น เราจะสามารถคาดคะเนได้ว่า การเดินที่ผ่านมานั้นทำการเดินได้ดีแค่ไหน กล่าวคือ มีค่าคะแนนเป็นเท่าใด นอกจากนั้นยังจะสามารถทำนายได้ด้วยว่า หากเดินไปยังช่องทางใด แล้วกระทำการใดๆ นั้น จะก่อให้เกิดผลดีหรือเสียมากน้อยแค่ไหนจากการทำนายผลด้วยคะแนนของการกระทำที่จะกระทำต่อไป

โดยรายละเอียดในการหาค่าต่างๆ สำหรับสมการนั้นจะทำการหาจากการคำนวณตามฟังก์ชันต่างๆ ดังนี้

ฟังก์ชันคำนวณค่าคะแนนระยะห่างจากตัวละคร c มายังตำแหน่ง (x,y) ได้แก่

$$\delta(c,x,y) = W_d \times W_c \times \sqrt{((x-x_0)^2 + (y-y_0)^2)} \quad (4.4)$$

โดยที่ W_d คือ ค่าน้ำหนัก (Weight) ของคะแนนระยะทาง

W_c คือ ค่าน้ำหนัก (Weight) ของตัวละคร c

x คือ ตำแหน่งตามแนวแกน x ที่ต้องการคิดคะแนน

x_0 คือ ตำแหน่งตามแนวแกน x ของตัวละคร c

y คือ ตำแหน่งตามแนวแกน y ที่ต้องการคิดคะแนน

y_0 คือ ตำแหน่งตามแนวแกน y ของตัวละคร c

ซึ่งค่า W_c นั้นจะแตกต่างกันไปตามอาชีพของตัวละครแต่ละอาชีพ(มีค่าเป็นบวกเสมอ) สำหรับค่า Wd (ซึ่งเป็นลบเสมอ) จะมีเกณฑ์การให้คะแนนที่ต่างกันเล็กน้อยโดยที่หากตัวละคร c ที่เป็นเป้าหมายเป็นทีมเดียวกับตัวละครที่กำลังคำนวณอยู่ในขณะนั้น จะมีผลทำให้ค่าน้ำหนักมีค่ามากกว่าของตัวละคร c ที่เป็นฝ่ายตรงข้ามอยู่เล็กน้อย เนื่องจากถือว่าการอยู่แบบค่อนข้างเกาะกลุ่มกันจะทำให้สามารถช่วยเหลือกันได้ดีมากขึ้น ทำให้ควรมีคะแนนมากกว่าเมื่อเป็นฝ่ายตรงข้ามเล็กน้อย นอกจากนี้กรณีของการเข้าหาฝ่ายตรงข้ามนั้นจะมีเงื่อนไขการให้คะแนนแบ่งได้อีกสองกรณี นั่นคือ หากค่า Hp ของตัวละครที่กำลังจะทำการเดินในตานี้มีค่ามากกว่าครึ่งหนึ่งของ Hp ทั้งหมดของตัวละคร หรือเมื่อเดินไปในตำแหน่ง (x,y) นั้นแล้วสามารถกระทำการโจมตีฝ่ายตรงข้ามให้ตายได้ คะแนนค่าน้ำหนักนี้จะได้รับค่าสูงขึ้นเพื่อกระตุ้นให้ตัวละครเข้าไปกำจัดตัวละครที่ใกล้ตายนั้น แต่หากค่า Hp ของตัวละครที่กำลังจะทำการเดินในตานี้มีค่าน้อยกว่าครึ่งหนึ่งของค่า Hp ทั้งหมดของตัวละครนั้น ค่าน้ำหนักที่ให้จะได้รับค่าลดลงเพื่อไม่ให้ตัวละครเข้าไปเสี่ยงต่อสู้กับศัตรูในขณะที่พลังชีวิตเหลือน้อย โดยรายละเอียดของค่าน้ำหนักที่ใช้ในงานวิทยานิพนธ์นี้อยู่ในส่วนของภาคผนวก ข

สำหรับคะแนนความเสียหายนั้น จะสามารถเกิดขึ้นได้เมื่อตัวละครที่กำลังคิดคะแนนนั้นสามารถกระทำการโจมตีตัวละครอื่นได้ โดยมีการคิดคะแนนดังนี้

$$ds_{(action,c)} = W_{ds} \times W_c \times W_{Dead} \times Damage \quad (4.5)$$

โดยที่ $ds_{(action,c)}$ คือ ค่าคะแนนความเสียหายเมื่อกระทำ action กับตัวละครเป้าหมาย c

W_{ds} คือ ค่าน้ำหนัก (Weight) ของคะแนนความเสียหาย

W_c คือ ค่าน้ำหนัก (Weight) ของตัวละครที่ถูกโจมตี

W_{Dead} คือ ค่าน้ำหนัก (Weight) หากการโจมตีทำให้ตัวละครที่ถูกโจมตีนั้นตาย ซึ่งหากตัวละครที่ถูกโจมตีไม่ได้ตายจากการโดนโจมตีนั้น ค่านี้จะมีค่าเป็น 1

Damage คือ ค่าความเสียหายที่สามารถกระทำได้

ซึ่งค่า W_{ds} นั้นมีค่าเปลี่ยน โดยขึ้นกับสถานการณ์ หากมีการกระทำความเสียหายกับตัวละครที่เป็นฝ่ายตรงกันข้าม ค่าน้ำหนักนี้จะเป็นตัวที่ทำให้คะแนนความเสียหายเพิ่มขึ้น แต่หากเลือกที่จะทำการโจมตีตัวละครฝ่ายเดียวกันจะทำให้มีค่าคะแนนความเสียหายติดลบ เพราะถือว่าเป็นการกระทำที่ไม่ดีเป็นอย่างมาก ส่วน W_c ก็จะมีค่าแตกต่างกันไปตามอาชีพของตัวละคร c นั่นเอง โดยรายละเอียดของค่าน้ำหนักที่ใช้ในงานวิทยานิพนธ์นี้อยู่ในส่วนของภาคผนวก ข

ส่วนในด้านของคะแนนของสภาวะแวดล้อมอื่นๆ (en) นั้น จะสามารถหาได้จากสมการดังต่อไปนี้

$$en_c = Score_{HP_c} + Score_{Heal_c} + Score_{Status\ effect_c} + \cdots \quad (4.6)$$

ซึ่งคะแนนในส่วนของสภาวะแวดล้อมอื่นๆ นี้ อาจแตกต่างกันไปแล้วแต่เกม แต่ละอย่างที่นำมาคิดคะแนนในส่วนนี้ไม่จำเป็นต้องเป็นพารามิเตอร์หลักที่ใช้คิดคะแนนตามปกติ ซึ่งในกรณีที่จะใช้ในวิทยานิพนธ์นี้ ได้นำในส่วนของ $Score_{HP\ c}$ นั่นคือ คะแนนของพลังชีวิต (Hp) ที่เหลืออยู่ของตัวละครในทีมมาช่วยในการคิด โดยคิดจากค่าร้อยละของ Hp ที่เหลือคูณกับค่าน้ำหนักของคะแนน Hp นี้ ดังสมการ (4.7)

$$Score_{HP_c} = W_{HP} \times \frac{HP_c}{HP_{Max\ c}} \times 100 \quad (4.7)$$

โดยที่ W_{HP} คือ ค่าน้ำหนักของคะแนนพลังชีวิต (Hp)

HP_c คือ ค่าพลังชีวิต (Hp) ที่จะเหลือของตัวละคร c อยู่เมื่อทำการเดินนี้จบ

$HP_{Max\ c}$ คือ ค่าพลังชีวิต (Hp) สูงสุดของตัวละคร c

ซึ่งค่า W_{HP} นี้หากคิดในกรณีของฝ่ายเดียวกันจะมีค่าเป็นบวก ซึ่งจะส่งผลให้เมื่อพลังชีวิตของทีมน้อย คะแนนก็จะมีค่าน้อยตาม และถ้าพลังชีวิตของทีมมีมากก็จะทำให้คะแนนมากตามเช่นกัน แต่หากเป็นในกรณีของการคิดคะแนนฝ่ายตรงข้าม ค่า W_{HP} นี้จะมีค่าเป็นลบ ซึ่งจะส่งผลให้คะแนนน้อยลงหากพลังชีวิตของฝ่ายตรงข้ามมาก และในทางกลับกันหากทำให้ฝ่ายตรงข้ามพลังชีวิตน้อยลงก็จะได้คะแนนมากขึ้นตามไปด้วย โดยรายละเอียดของค่าน้ำหนักที่ใช้ในงานวิทยานิพนธ์นี้อยู่ในส่วนของภาคผนวก ข

ส่วนอีกค่าที่จะใช้ในวิทยานิพนธ์นี้คือ $Score_{Heal\ c}$ ซึ่งจะมีค่าในกรณีที่ตัวละครที่มีอาชีพ White mage ได้ทำการฟื้นพลัง (หรือก็คือการเติม Hp) ให้กับเป้าหมายดังสมการ (4.8)

$$Score_{Heal_c} = W_{Heal} \times W_c \times Heal\ Point \quad (4.8)$$

โดยที่ W_{Heal} คือ ค่าน้ำหนักของคะแนนของการฟื้นพลัง

W_c คือ ค่าน้ำหนักของตัวละครที่ได้รับการฟื้นพลัง

Heal Point คือ ค่าพลังที่ฟื้นคืนขึ้นมาได้จากการฟื้นพลังนี้

ซึ่งค่า $Score_{Heal\ c}$ นี้จะเกิดขึ้นก็ต่อเมื่อมีการฟื้นพลังเกิดขึ้น หากไม่มีการกระทำที่เป็นการฟื้นพลังแล้ว จะกำหนดให้ค่า $Score_{Heal\ c}$ นี้มีค่าเท่ากับ 0 สำหรับค่า W_{Heal} นั้นจะมีลักษณะคล้ายกับ W_{ds} ของการคำนวณคะแนนค่าความเสียหาย แต่จะมีลักษณะการคิดตรงกันข้าม นั่นคือ หากทำการฟื้นพลังให้กับฝ่ายเดียวกันค่า W_{Heal}จะมีค่าเป็นบวก ทำให้ค่าคะแนนการฟื้นพลังนี้มากขึ้น แต่หากทำการฟื้นพลังให้กับฝ่ายตรงข้าม ค่า W_{Heal} จะมีค่าเป็นลบ ทำให้ค่าคะแนนการฟื้นพลังมีค่าเป็นลบไปด้วย ส่งผลให้คะแนนรวมมีค่าลดลง ส่วนค่า W_c นั้นก็จะแตกต่างกันไปตามอาชีพของตัวละคร c โดยรายละเอียดของค่าน้ำหนักที่ใช้ในงานวิทยานิพนธ์นี้อยู่ในส่วนของภาคผนวก ข

4.3 การปรับเพื่อให้เกิดความสูสี

ในการปรับเพื่อให้เกิดความสูสีกับผู้เล่นนั้น ผู้เขียนได้ทำการพัฒนาปัญญาประดิษฐ์ขึ้นโดยมีแนวทางในการเล่นโดยอ้างอิงการเล่นจากการใช้คะแนนของฝ่ายตรงข้ามเข้าช่วย ผนวกกับการคิดคะแนนของฝ่ายเราที่จะทำการเดิน เพื่อให้สามารถเล่นได้อย่างสูสีกับฝ่ายตรงข้าม โดยจะทำการคำนวณค่าจากสมการ (4.7) ก่อนทำการเดินในตาการเดินนั้น

$$e(x,y) = s(x,y) - \phi(\mu) \quad (4.9)$$

โดยที่ e(x,y) คือ ค่าความสูสีเมื่อตัวละครกระทำการใดๆ อยู่ในช่อง (x,y)

s(x,y) คือ ค่าคะแนนของปัญญาประดิษฐ์ที่สามารถทำได้เมื่อไปกระทำการใดๆ อยู่ที่ช่อง (x,y) นั้น

$\phi(\mu)$ คือ ฟังก์ชันของการหาค่าคะแนนของฝ่ายตรงข้ามที่ได้เกิดขึ้นมาก่อน

ซึ่งค่า $\phi(\mu)$ จะมีค่าที่มาจากการนำคะแนนของฝ่ายตรงข้ามซึ่งได้เคยเกิดขึ้นมาก่อนหน้าที่จะมีการเดินในตานี้ โดยค่านี้จะสามารถหาค่าได้หลายวิธี ซึ่งจะกล่าวต่อไปในหัวข้อ 4.3.1 ถึง 4.3.3

จากสมการจะเห็นได้ว่าหากค่า e(x,y) มีค่าเป็นบวก แสดงว่าฝ่ายเราทำคะแนนได้ดีกว่าฝ่ายตรงข้าม และในทางกลับกันหากได้ค่าออกมาเป็นลบ ก็จะแสดงว่าฝ่ายตรงข้ามทำคะแนนได้ดีกว่าฝ่ายเรา และยิ่งค่า e(x,y) เข้าใกล้ค่า 0 เท่าไหร่ก็จะแสดงว่าสามารถเล่นได้สูสี กับผู้เล่นมากขึ้นเท่านั้น

จากรูปที่ 23 จะเห็นได้ว่าเราสามารถนำค่าความสูสีที่คำนวณได้นั้น มาจัดระดับให้เกิดความแตกต่างในการเล่นต่างๆ ได้ด้วย นอกจากจะเล่นเพื่อให้สูสีเท่านั้น โดยในที่นี้ค่า Level ในระดับ 0 คือ ปัญญาประดิษฐ์สามารถเล่นได้อย่างสูสีกับผู้เล่น และยิ่งเมื่อ Level สูงขึ้น ค่าความสูสีจากสมการ ก็ยิ่งมีค่ามากขึ้น แสดงว่าปัญญาประดิษฐ์จะเก่งมากขึ้น ส่วนหากค่าความสูสีลดลง ปัญญาประดิษฐ์ก็จะมีความเก่งน้อยลงตาม โดยในที่นี้ค่า L0, L1, L2 นั้นจะเป็นระยะที่กำหนดขึ้นเพื่อเป็นค่าแบ่งระหว่างค่าความสูสีแต่ละช่วง ว่าจะให้ค่าความสูสีมากเท่าใดถึงจะเป็นระดับถัดไป ซึ่งจากแนวคิดนี้ เมื่อนำไปประยุกต์เข้ากับตัวเกมที่ทำขึ้นจริง สามารถปรับให้ด่านช่วงต้นของเกมมีระดับการเล่นที่ต่ำกว่าตัวผู้เล่นเล็กน้อย หลังจากนั้นค่อยเพิ่ม Level ขึ้นทำให้เริ่มมาเล่นได้อย่างสูสีกับผู้เล่น และด่านหลังๆ จึงให้ Level มากขึ้นเพื่อให้เกิดความท้าทายแก่ผู้เล่นได้ด้วย โดยที่ใช้ความสามารถของผู้เล่นเป็นหลัก

Smarter

Level	Range of e(x,y)
3	L2 < e(x,y)
2	L1 < e(x,y) < L2
1	L0 < e(x,y) < L1
0	-L0 < e(x,y) < L0
-1	-L1 < e(x,y) < -L0
-2	-L2 < e(x,y) < -L1
-3	e(x,y) < -L2

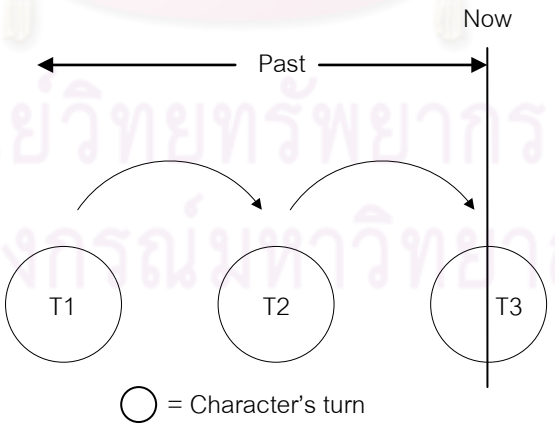
รูปที่ 23 แสดงการจัดระยะของระดับความสูสี

สำหรับวิธีในการเลือกค่าจากในช่วงความสูสีต่างๆ มาใช้นั้น หากต้องการให้เล่นกับผู้เล่นได้ในระดับใด ก็สามารถทำการสุ่มค่าที่อยู่ในช่วงระดับความสูสีนั้นมาใช้

สำหรับแนวคิดในการนำค่าคะแนนความสูสีนี้ไปประยุกต์ใช้นั้น ได้นำขั้นตอนการหาความสูสีนี้ไปใช้ทดลองโดยที่พิจารณาใน 3 รูปแบบ คือ

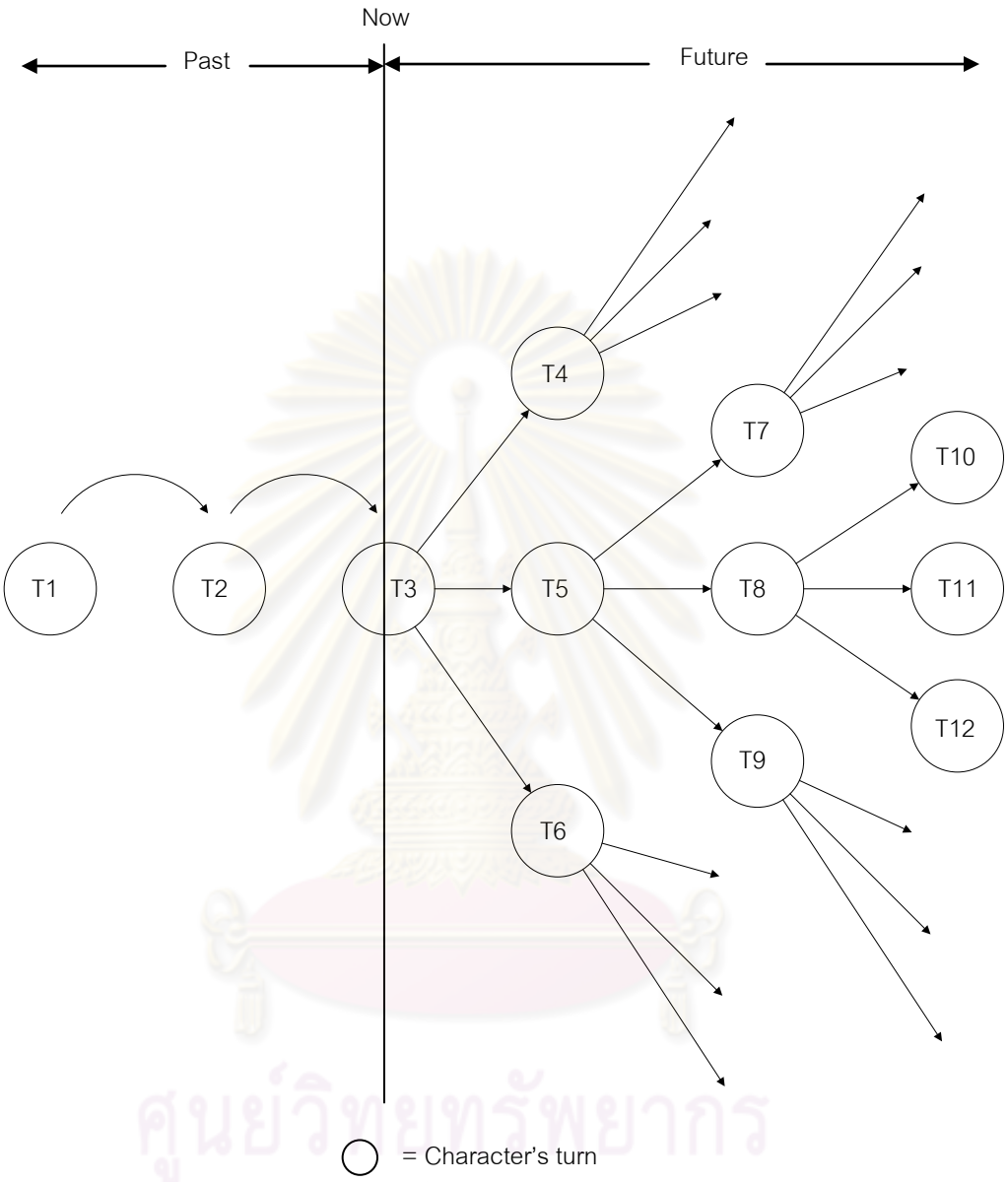
4.3.1 การคิดค่าความสูสีโดยอาศัยคะแนนจากฝ่ายตรงข้ามในตาก่อนหน้าหนึ่งตา ซึ่งแนวคิดนี้จะคล้ายกับการเลียนแบบฝ่ายของผู้เล่น โดยที่จะอ้างอิงจากสิ่งที่ฝ่ายของผู้เล่นเคยทำมาก่อนหน้านั้น ตัวอย่างเช่น หากฝ่ายของผู้เล่นทำการเดินเข้ามาหา เมื่อคิดคะแนนค่าความสูสีแล้วค่าความสูสีที่อยู่ในช่วงใกล้เคียงส่วนใหญ่จะอยู่ในกรณีของการเดินในลักษณะคล้ายกับการเดินเข้ามาหาเช่นเดียวกับฝ่ายของผู้เล่น และเมื่อฝ่ายของผู้เล่นเลือกกระทำในกรณีอื่นอย่างเช่นทำการโจมตีตัวละคร เมื่อถึงตาเดินของตัวละครฝ่ายปัญญาประดิษฐ์ก็จะมีแนวโน้มส่วนใหญ่ให้ตัวละครทำการโจมตีฝ่ายของผู้เล่นกลับไปเช่นกัน

4.3.2 การคิดค่าความสูสีโดยอาศัยคะแนนจากฝ่ายตรงข้ามในตาก่อนหน้าเป็นจำนวนหนึ่ง ซึ่งแนวคิดนี้จะแตกต่างจากแนวคิดแรกตรงที่ จะทำการรวบรวมข้อมูลของฝ่ายตรงข้ามมามากกว่าเก่า และใช้ข้อมูลโดยรวมของฝ่ายของผู้เล่นในอดีตเป็นตัวช่วยในการประมวลค่าความสูสี ดังเช่นรูปที่ 24 ตาเดินในปัจจุบันที่เกิดขึ้นก็คือตาเดิน T3 ซึ่งในรูปแบบของแนวคิดนี้ จะนำคะแนนของตาเดินก่อนหน้าที่ผ่านมาได้แก่ T1 และ T2 มาดูค่าเฉลี่ยที่เกิดขึ้น แล้วจึงนำมาหาค่าความสูสีเพื่อพิจารณาคะแนนที่ควรจะทำได้ในตาเดิน T3 โดยพยายามให้ ค่าเฉลี่ยของคะแนนปัญญาประดิษฐ์รวมตั้งแต่ T1 ถึง T3 เท่ากับค่าเฉลี่ยของคะแนนของผู้เล่น (ถ้าไม่มีการเดินของผู้เล่นเลย จะพยายามให้ค่าเฉลี่ยของคะแนนฝ่ายปัญญาประดิษฐ์เท่ากับค่าคะแนนล่าสุดของผู้เล่นที่ได้เก็บไว้) ซึ่งหากเป็นแนวคิดแรกจะพิจารณาคะแนนค่าความสูสีจากเพียงแค่ระหว่าง T2 กับ T3 เท่านั้น

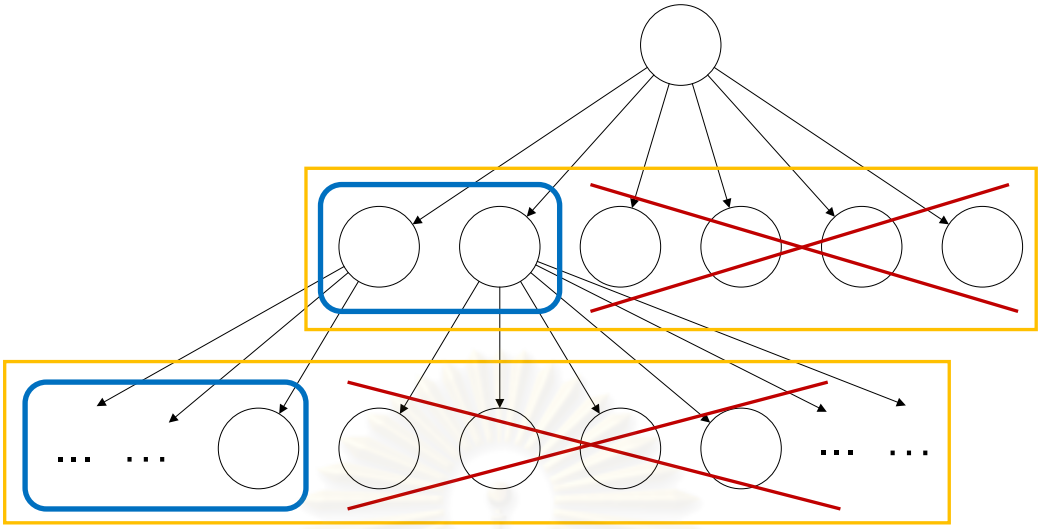


รูปที่ 24 แสดงลักษณะของการมองตาเดินในอดีต

4.3.3 การคิดค่าความสูสีโดยอาศัยคะแนนจากทั้งตาเดินที่เคยเกิดขึ้นก่อนหน้า และตาเดินที่มีความเป็นไปได้ในอนาคต ซึ่งแนวคิดนี้จะทำการพิจารณาดังรูปที่ 25 ซึ่งตาเดินที่เคยเกิดขึ้นในอดีตคือ T1 และ T2 ส่วน T3 จะเป็นตาเดินในปัจจุบันที่กำลังจะเลือกเดิน ซึ่งจะก่อให้เกิดตาเดินอื่นๆ ที่อาจจะเป็นไปได้ในอนาคต เช่น T4, T5, T6, ..., T12 เป็นต้น ซึ่งจากการที่แนวการคิดนี้จะต้องทำการพิจารณาลักษณะของตาเดินที่อาจจะเกิดขึ้นได้ในอนาคต ดังนั้นการคำนวณเพื่อหาตาที่จะสามารถเกิดขึ้นได้ในอนาคตนี้ จะต้องใช้เวลาในการคำนวณ ซึ่งหากพิจารณาในทุกกรณีที่มีโอกาสเกิดขึ้นได้นั้น จะทำให้ต้องใช้เวลาในการคำนวณค่อนข้างช้ามาก ซึ่งจะไม่สามารถยอมรับได้หากนำไปประยุกต์เพื่อใช้จริง ดังนั้น การมองล่วงหน้านั้นจะไม่สามารถทำการมองล่วงหน้าไปได้ไกลมากนัก และเพื่อเป็นการเพิ่มประสิทธิภาพ โดยการพยายามมองออกไปให้ไกลที่สุด โดยที่ใช้เวลาให้น้อยลง ผู้เขียนจึงได้นำเสนอแนวคิดในการพิจารณาคัดกรอง เพื่อทำการตัดตาเดินบางตาออก เพื่อให้สามารถมองลึกลงไปได้ในเวลาที่เร็วขึ้นกว่าการเลือกที่จะพิจารณาหมดทุกกรณี ซึ่งจากการที่เกมในลักษณะนี้นั้น ตาเดินในอนาคตที่จะเกิดขึ้นได้ไม่ได้เรียงลำดับเป็นการตายตัวว่าตาเดินของแต่ละฝ่ายจะเป็นการผลัดกันเดินแบบตาต่อตา และด้วยแนวคิดที่พยายามให้สามารถเล่นได้อย่างสูสีกับผู้เล่นในระหว่างการเล่น จึงไม่สามารถใช้การหาตาล่วงหน้าแบบต้นไม้มินิแม็กซ์ได้แต่จะใช้แนวคิดในลักษณะคล้ายกัน โดยจะทำการพิจารณาตาที่จะกำลังจะเกิดขึ้นในอนาคตอันใกล้ และหาว่าตานั้นสามารถยอมรับได้หรือไม่ ถ้าจะให้ค่าระดับความสูสีเป็นไปตามที่ต้องการแล้วจึงเลือกเอาตาเดินที่ยอมรับได้เป็นจำนวนประมาณ 1 ใน 3 ของตาเดินที่ยอมรับได้ทั้งหมดไปพิจารณาต่อทีละระดับชั้นดังรูปที่ 26 โดยการคิดเพื่อพิจารณาถึงตาเดินที่ค่าความสูสีเป็นที่ยอมรับได้ แบ่งออกได้เป็น 2 ลักษณะ คือ



รูปที่ 25 แสดงลักษณะการมองตาเดินทั้งในอดีต และอนาคต



รูปที่ 26 แสดงลักษณะตัวอย่างการเลือกข้อมูลที่สูสีกับผู้เล่น

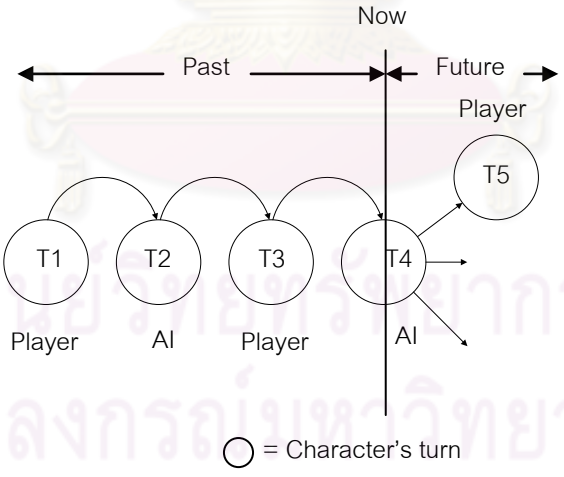
1. ตาเดินในอนาคตของตัวละครฝ่ายของผู้เล่น จะทำการพิจารณาโดยการอ้างอิงจากอดีตของคะแนนที่ฝ่ายของผู้เล่นเคยทำไว้ได้ในอดีตเอง ซึ่งในลักษณะการพิจารณา จะแบ่งพิจารณาออกได้เป็น 3 กรณี คือ

กรณีแรก เมื่อค่าคะแนนของผู้เล่นในอดีตมีแนวโน้มว่าทำคะแนนได้สูงขึ้น เช่นจากรูปที่ 27 ตาเดินในปัจจุบันของปัญญาประดิษฐ์คือตาเดิน T4 ส่วนตาเดินในอนาคตของตัวละครฝ่ายของผู้เล่นที่กำลังจะพิจารณาในลักษณะนี้คือตาเดิน T5 และตาเดิน T1 และ T3 เป็นตาเดินในอดีตของฝ่ายของผู้เล่น กรณีที่ค่าคะแนนของผู้เล่นมีแนวโน้มว่าจะทำคะแนนได้สูงขึ้นนี้เกิดในกรณีที่คะแนนของ T3 มากกว่าคะแนนของ T1 ตั้งแต่หนึ่งระดับความสูสีขึ้นไป วิธีพิจารณาเลือกตาเดินของฝ่ายของผู้เล่นที่จะเป็นที่ยอมรับได้ในกรณีนี้จะเลือกวิธีการเดินที่จะทำให้ตาเดิน T5 นี้มีคะแนนสูงหนึ่งระดับขึ้นไปจากตาเดิน T1 เป็นหลัก เพราะใช้หลักการพิจารณาว่า ถ้าคะแนนของฝ่ายผู้เล่นในอดีตมีแนวโน้มสูงขึ้น มีโอกาสที่ฝ่ายของผู้เล่นจะเล่นได้ดีขึ้นนั่นเอง

กรณีที่สอง เมื่อค่าคะแนนของผู้เล่นในอดีตมีแนวโน้มค่อนข้างเท่าเดิม เช่นจากรูปที่ 27 หากคะแนนของ T1 และ T3 มีค่าใกล้เคียงกัน (อยู่ในระดับค่าความสูสีระดับเดียวกัน) จะถือว่าผู้เล่นมีแนวโน้มของคะแนนในอดีตค่อนข้างเท่าเดิมเช่นในกรณีที่สองนี้ วิธีพิจารณาเลือกตาเดินของฝ่ายผู้เล่นในกรณีนี้จะเลือกวิธีเดินโดยการเลือกวิธีการเดินที่จะทำให้ตาเดินของ T5 นี้ มีคะแนนเฉลี่ยใกล้เคียงกับคะแนนเฉลี่ยของ T1 และ T3 นั่นเอง เพราะใช้หลักพิจารณาว่า ถ้าคะแนนของ

ฝ่ายผู้เล่นในอดีตมีแนวโน้มค่อนข้างเท่าเดิม ผู้เล่นจะมีโอกาสที่จะทำคะแนนได้เท่าๆ กับของเดิมเป็นส่วนใหญ่

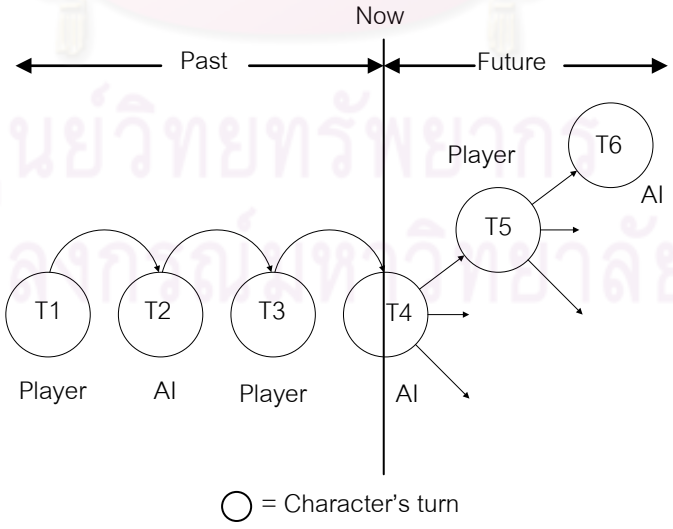
กรณีที่สาม เมื่อค่าคะแนนของผู้เล่นในอดีตมีแนวโน้มว่าทำคะแนนได้ลดลง เช่นจากรูปที่ 27 หากคะแนนของ T1 มากกว่าคะแนนของ T3 ตั้งแต่หนึ่งระดับความสูสีขึ้นไป วิธีพิจารณาเลือกตาเดินของฝ่ายของผู้เล่นที่ยอมรับได้ในกรณีนี้จะเลือกวิธีการเดินที่จะทำให้ตาเดินของ T5 นี้ มีคะแนนเฉลี่ยใกล้เคียงกับคะแนนเฉลี่ยของ T1 และ T3 เพราะใช้หลักในการพิจารณาว่า การที่แนวโน้มของคะแนนที่ผู้เล่นทำได้ลดลงนี้ไม่อาจแน่ใจได้ว่า ผู้เล่นทำคะแนนได้แย่ลงจริง หรืออาจเป็นกลยุทธ์การเดินของผู้เล่นที่อาจเดินให้คะแนนน้อยลง เพื่อจะสามารถทำคะแนนที่มากขึ้นได้ในตาเดินถัดไปก็เป็นได้ ซึ่งหากปัญญาประดิษฐ์เลือกทำการกระทำที่มีคะแนนลดลง ฝ่ายปัญญาประดิษฐ์อาจถูกหลอก และไม่สามารถตามเกมทันได้ แต่ถ้าเลือกทำการกระทำที่มีคะแนนดีเพื่อกันการหลอกไว้ก่อน แล้วฝ่ายของผู้เล่นอาจไม่ได้หลอก แต่เล่นไม่ดีจริงๆก็เป็นได้ ดังนั้นการเลือกการกระทำที่มีคะแนนดีเลยนี้ อาจทำให้ผู้เล่นกลายเป็นฝ่ายเสียเปรียบและตามเกมไม่ทันในทันที ดังนั้นสิ่งที่ดีที่สุดที่ทำได้คือ เลือกคะแนนที่อยู่ในช่วงของค่าเฉลี่ยที่ฝ่ายตรงข้ามสามารถทำได้ในอดีตแทน เพื่อไม่ให้เกิดกรณีที่ฝ่ายใดฝ่ายหนึ่งไม่สามารถแก้เกมฝ่ายตรงข้ามได้



รูปที่ 27 แสดงตัวอย่างเมื่อพิจารณาตาเดินในอนาคตของตัวละครฝ่ายของผู้เล่น

ซึ่งทั้งนี้สาเหตุที่ต้องตั้งสมมติฐานแบ่งออกเป็น 3 กรณีเช่นนี้ เนื่องมาจากผลการทดลองที่เกิดจากการที่ได้ใช้ค่าเฉลี่ยจากผู้เล่นคิดในทุกกรณีนั้น ปรากฏว่าส่วนใหญ่แล้วกรณีที่ผู้เล่นสามารถทำคะแนนได้ดีขึ้นนั้น ผู้เล่นสามารถเล่นเก่งขึ้นจริงเป็นส่วนใหญ่ และมีผลทำให้ฝ่ายปัญญาประดิษฐ์ไม่สามารถปรับตัวให้เก่งขึ้นตามได้ทัน แต่จะยิ่งปรับตัวให้เก่งขึ้นได้ช้ากว่าแนวคิด 4.3.2 เสียอีก เนื่องจากคะแนนในส่วนอนาคตที่ทำนายไว้ว่าจะทำได้ไม่ดียิ่งถ่วงน้ำหนักให้ค่าของคะแนนที่ควรจะคิดได้จริงตกลง จึงทำให้ความสามารถในการเล่นได้สูสีกับผู้เล่นตกลง

2. ตาเดินในอนาคตที่สามารถยอมรับได้ของตัวละครฝ่ายปัญญาประดิษฐ์ จะทำการพิจารณาโดยการดูว่า คะแนนที่ฝ่ายของผู้เล่นสามารถทำได้ในอดีตนั้น เฉลี่ยแล้วเทียบกับค่าที่ฝ่ายปัญญาประดิษฐ์สามารถทำได้โดยที่รวมจากทั้งคะแนนที่เคยทำได้ในอดีตกับคะแนนที่กำลังจะเกิดขึ้นนี้มีความสูสีกันเท่าใด ดังรูปที่ 28 หาก T4 คือตาเดินในปัจจุบัน และตาเดินในอนาคตของฝ่ายปัญญาประดิษฐ์ที่จะพิจารณาคือ T6 และ T1 กับ T3 เป็นตาเดินของฝ่ายผู้เล่นที่เคยเกิดขึ้นในอดีต T2 คือตาเดินของฝ่ายปัญญาประดิษฐ์ที่เคยเกิดขึ้นในอดีต เช่นกัน ส่วน T5 คือตาเดินในอนาคตของผู้เล่นที่คาดว่าน่าจะเกิดขึ้น (ผ่านการพิจารณาว่ายอมรับได้แล้ว) การพิจารณา T6 นี้จะทำโดยการดูว่าค่าเฉลี่ยจาก T1 T3 และ T5 นั้น เปรียบเทียบกับคะแนนเฉลี่ยจาก T2 T4 รวมถึง T6 นี้ว่ามีความสูสีในระดับเดียวกันหรือไม่ T6 จะเป็นที่ยอมรับได้ก็ต่อเมื่อทำให้เกิดความสูสีในระดับเดียวกัน



รูปที่ 28 แสดงตัวอย่างเมื่อพิจารณาตาเดินในอนาคตของตัวละครฝ่ายปัญญาประดิษฐ์

โดยหลังจากที่หาแนวทางที่สามารถเกิดขึ้นได้ต่างๆ ขึ้นแล้ว จึงนำมาดูค่าความสูงสีจากคะแนนนั้นเพื่อเลือกทางเดินที่จะทำให้สามารถเล่นได้อย่างสูงสีกับฝ่ายตรงข้ามมากที่สุด

บทที่ 5

วิธีการทดลอง และผลการทดลอง

5.1 วิธีการทดลอง

สำหรับการทดลอง ได้ทำการทดลองโดยนำเกมวางแผนการรบแบบผลัดกันเล่นที่ได้พัฒนาขึ้นนี้ มาทดสอบระหว่างปัญญาประดิษฐ์สองฝ่าย โดยฝ่ายหนึ่งจะเป็นปัญญาประดิษฐ์ที่พัฒนาให้สามารถเล่นแบบสูสีกับผู้เล่น ส่วนอีกฝ่ายหนึ่งเป็นปัญญาประดิษฐ์ที่มีการโปรแกรมให้ใช้กลยุทธ์การเล่นในแบบต่างๆ กัน โดยกลยุทธ์ที่ใช้มีดังนี้

Rush มีกลยุทธ์การเล่นในแบบที่ตัวละครจะพยายามทำการเดินหน้าเข้าโจมตีศัตรูตัวที่อยู่ใกล้ที่สุดเสมอเท่าที่สามารถทำได้

Unit Offence มีกลยุทธ์การเล่นในแบบที่จะมุ่งโจมตีตัวละครตัวใดตัวหนึ่งที่ได้ตัดสินใจไว้ก่อนจนกว่าตัวละครนั้นจะตาย แล้วจึงโจมตีตัวละครตัวต่อไปแบบนี้ไปเรื่อยๆ กล่าวคือ จะช่วยกันโจมตีตัวละครให้ตายทีละตัวไม่แยกกันโจมตีนั่นเอง

Defense มีกลยุทธ์การเล่นในแบบที่จะคอยตั้งรับเท่านั้น คือจะไม่พยายามเข้าไปโจมตีฝ่ายศัตรูนัก แต่จะตอบโต้ก็ต่อเมื่อโดนศัตรูเข้ามาบุกเท่านั้น

Balance มีกลยุทธ์การเล่นแบบค่อนข้างสมดุล ไม่เน้นไปที่การโจมตีมากเกินไป หรือการป้องกันมากเกินไป

สำหรับในส่วนของการทดลองนี้ ได้ทดลองปัญญาประดิษฐ์แบบปรับตัวได้ (ซึ่งต่อไปนี้จะขอเรียกว่า ฝ่ายปัญญาประดิษฐ์) ในการพิจารณาจากคะแนนของฝ่ายตรงข้ามใน 3 ลักษณะเข้าต่อสู้กับปัญญาประดิษฐ์ที่ใช้กลยุทธ์ที่กล่าวมาแล้วทั้ง 4 กลยุทธ์ (ซึ่งต่อไปนี้จะขอเรียกว่า ฝ่ายของผู้เล่น) ซึ่งลักษณะในการพิจารณาคะแนนของฝ่ายตรงข้ามของฝ่ายปัญญาประดิษฐ์นั้น แบ่งออกได้เป็น 3 ลักษณะ ดังแนวคิดที่ได้กล่าวมาก่อนหน้านี้ซึ่งได้แก่

5.1.1 ปัญญาประดิษฐ์แบบปรับตัวได้แบบสนใจมองคะแนนของฝ่ายของผู้เล่นในตาก่อนหน้าหนึ่งตาเท่านั้น โดยในลักษณะนี้จะทำการมองคะแนนของฝ่ายของผู้เล่นว่า ตาก่อนหน้านั้นมีการทำคะแนนได้ดีเพียงใด และพยายามทำคะแนนให้ได้สูสีกับฝ่ายของผู้เล่นในตาก่อนหน้านั้น โดยการอาศัยคะแนนในตาการเดินก่อนหน้าเพียงตาเดียวนั้น ปัญญาประดิษฐ์ควรจะสามารถ

ตอบสนองสิ่งที่คู่ต่อสู้กระทำได้อย่างรวดเร็ว แต่หากฝ่ายตรงข้ามมีการเล่นที่ใช้กลยุทธ์ที่ซับซ้อนขึ้น อาจทำให้ไม่สามารถตอบโต้กลยุทธ์นั้นได้เพราะมัวแต่ไปตอบสนองกับเหตุการณ์ที่พึ่งจะเกิดขึ้น

5.1.2 ปัญญาประดิษฐ์แบบปรับตัวได้แบบสนใจมองคะแนนของตาก่อนหน้าที่เคยเกิดขึ้นมาแล้วประมาณ 3 ตา ว่าฝ่ายของผู้เล่นนั้นสามารถทำคะแนนในการเล่นได้เป็นอย่างไร แล้วใช้คะแนนนั้นมาประยุกต์เพื่อหาค่าความสูสี และนำมาตัดสินใจการกระทำในตาปัจจุบัน ซึ่งการใช้แนวคิดนี้ควรจะสามารถปรับปรุงฝ่ายปัญญาประดิษฐ์ให้สามารถตอบสนองต่อการเล่นที่มีกลยุทธ์ซับซ้อนได้ดีกว่าแบบแรก แต่อาจทำให้การตอบสนองฝ่ายตรงข้ามนั้นทำได้ช้าลง

5.1.3 ปัญญาประดิษฐ์แบบปรับตัวได้แบบสนใจมองคะแนนของทั้งตาก่อนหน้า และตาที่จะเดินต่อไปในอนาคต โดยจะอาศัยคะแนนทั้งในส่วนตาที่เคยเกิดขึ้นมาแล้วจำนวน 3 ตาก่อนหน้า และกรณีที่อาจจะเกิดขึ้นได้ในอนาคตอีกจำนวน 3 ตา เพื่อดูว่าควรจะทำการเดินอย่างไรให้เกิดความสูสีกับฝ่ายของผู้เล่น โดยแนวคิดนี้จะเป็นการปรับปรุงแนวคิดที่สองให้สามารถตอบสนองต่อฝ่ายตรงข้ามได้ดีขึ้น โดยใช้แนวโน้มในการตัดสินใจ และการคาดคะเนการกระทำของฝ่ายตรงข้ามล่วงหน้า ดังนั้นจึงควรจะสามารถเล่นได้อย่างสูสีกับผู้เล่นได้ดีโดยมีข้อเสียน้อยที่สุดจากทั้ง 3 แนวคิดนี้

สำหรับแผนที่ที่ใช้ในการทดลองนั้นได้จำลองมาจากแผนที่จริงในเกม Final Fantasy Tactics A2: Grimoire of the Rift โดยในการทำการทดลองนี้จะให้ฝ่ายปัญญาประดิษฐ์ทั้ง 3 ลักษณะเข้าทำการแข่งกับฝ่ายของผู้เล่นที่ใช้กลยุทธ์ข้างต้นเป็น เป็นจำนวนกลยุทธ์ละ 100 เกม (รวมแล้วคือ ฝ่ายปัญญาประดิษฐ์หนึ่งแนวคิดจะทำการต่อสู้กับคู่ต่อสู้รวมทุกกลยุทธ์เป็นจำนวน 400 เกม) จากนั้นเก็บผลดูว่า ฝ่ายปัญญาประดิษฐ์นั้นสามารถเล่นได้สูสีกับฝ่ายตรงข้ามมากน้อยเพียงใด โดยที่เกมในการทดลองแต่ละเกม จะทำการเล่นโดยใช้แผนที่เดิมทุกครั้ง และมีการกำหนดจุดเริ่มต้นของตัวละครแต่ละฝั่งให้เป็นจุดเดิมทุกครั้ง รวมถึงตัวละครทั้งสองฝั่งจะมีจำนวนเท่ากัน และมีค่าสถานะต่างๆ เท่ากันทั้งฝ่าย (ซึ่งข้อมูลต่างๆ เหล่านี้ สามารถดูได้ในส่วนของภาคผนวก ข) และทำการรันบนคอมพิวเตอร์ notebook ยี่ห้อ lenovo Y550 ซึ่งมีหน่วยประมวลผล Intel® Core™2 Duo CPU P8800 @ 2.66 GHz 2.67 GHz และมีหน่วยความจำ 4.00 GB บนระบบปฏิบัติการ Windows Vista™ Business (32-bit) Service Pack 2

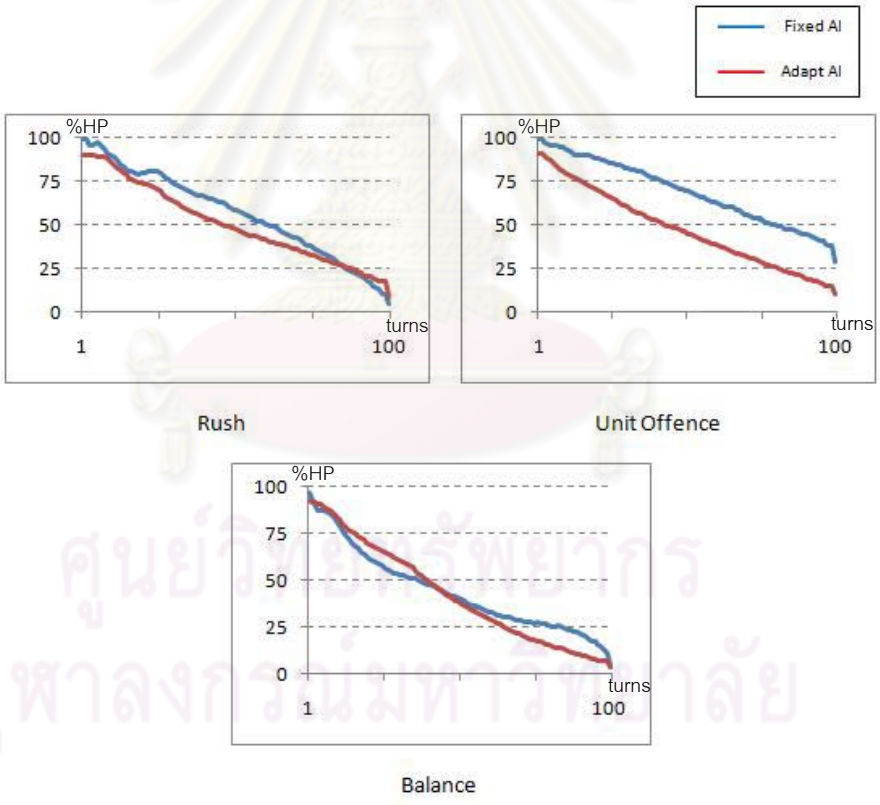
5.2 ผลการทดลอง และวิเคราะห์ผลการทดลอง

จากการทดลองนั้น ผลการทดลองที่ได้ซึ่งจะแสดงให้เห็นต่อไปนี้ คือค่าของค่าพลังชีวิต (Hp) ที่เหลือ ของฝ่ายที่ทำการสู้ชนะ โดยฝ่ายปัญญาประดิษฐ์นี้ ได้ทำการสู้กับฝ่ายของผู้เล่นทั้ง 4 ประเภทให้พยายามออกมาได้อย่างสูสีที่สุด ซึ่งจากผลการสอบถามจากผู้เล่นจำนวนหนึ่ง ได้ข้อมูลมาว่า การที่จะเล่นให้ได้สูสีในกรณีของการทดลองนี้นั้น ควรจะมีค่าร้อยละของ Hp ของฝ่ายที่ชนะเหลืออยู่ในอัตราเฉลี่ยไม่เกิน 21.00 โดยที่มีค่าเบี่ยงเบนมาตรฐานเท่ากับ 16.21

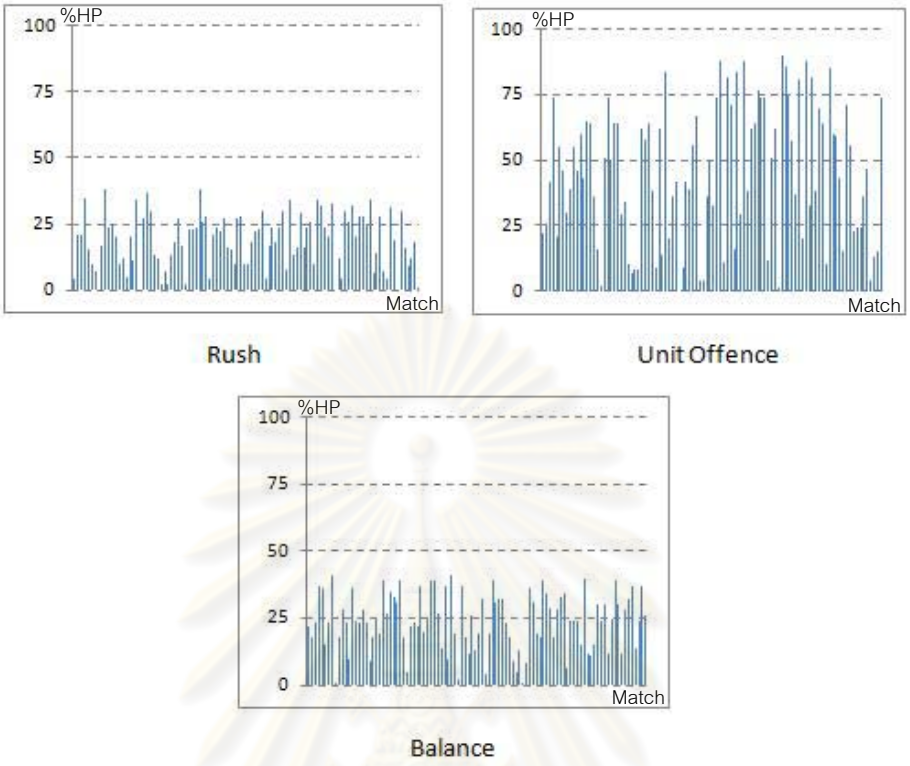
ส่วนผลการทดลองของฝ่ายปัญญาประดิษฐ์ทั้ง 3 ลักษณะนั้นได้ผลการทดลองออกมาดังต่อไปนี้

5.2.1 ปัญญาประดิษฐ์ที่มองคะแนนฝ่ายตรงข้ามในกรณีตาเดินก่อนที่จะพิจารณาเท่านั้น ซึ่งผลการทดลองที่ได้เป็นดังรูปที่ 29 และรูปที่ 30 ซึ่งรูปที่ 29 นั้นจะแสดงค่า HP เฉลี่ยที่เหลืออยู่ของแต่ละทีมในการเล่น 1 เกม โดยจากรูปแนวแกน y จะเป็นค่าร้อยละของ HP ที่เหลืออยู่ของทั้งสองทีม ส่วนแนวแกน x จะเป็นจำนวนตาเดิน ซึ่งได้ผ่านการปรับให้เป็นค่ามาตรฐานเท่ากับ 100 ตาเดิน เนื่องจากแต่ละเกมมีจำนวนตาเดินไม่เท่ากัน ทั้งนี้เพื่อสะดวกในการเทียบอัตราส่วนการลดลงของ HP รวมของทีมจากการกระทำของแต่ละฝ่ายนั่นเอง ซึ่งกราฟนี้จะแสดงค่าเฉลี่ยในการเล่นแต่ละตาเดินของเกมการเล่นทั้ง 100 เกมว่าในแต่ละช่วงเวลาของเกมนั้น HP โดยรวมของทีมทั้งสองมีแนวโน้มที่จะลดลงสูสีกันเพียงใด ส่วนรูปที่ 30 จะแสดงผลสุดท้ายของร้อยละของ HP ที่เหลือของทีมที่ชนะในแต่ละเกม โดยจากรูปที่ 30 นั้น แนวแกน y ของกราฟ คือค่าร้อยละของ Hp ที่เหลืออยู่ของฝ่ายที่ชนะในเกมนั้นๆ ส่วนแนวแกน x คือ เกมที่ทำการต่อสู้กันแต่ละเกม ซึ่งจะเห็นได้ว่าการแข่งกับฝ่ายของผู้เล่นในกลยุทธ์ Rush และ Balance นั้นฝ่ายปัญญาประดิษฐ์นี้สามารถจะเล่นได้ค่อนข้างสูสีกับฝ่ายของผู้เล่น แต่สำหรับในกรณีของ Unit Offence นั้น ฝ่ายของผู้เล่นจะมี Hp เหลือในแต่ละการต่อสู้ค่อนข้างมาก ทั้งนี้เพราะ เนื่องจากกลยุทธ์ของ Unit Offence นั้น จะมีการเน้นโจมตีไปที่ตัวละครตัวใดตัวหนึ่งให้ตายเสียก่อน แล้วจึงจะมาโจมตีตัวอื่นต่อ ดังนั้นจะทำให้เกิดกรณีเช่น ตัวละครเป้าหมายนั้นไปอยู่ในตำแหน่งด้านหลังของตัวละครอื่น ทำให้ตัวละครของฝ่ายของผู้เล่นที่ใช้กลยุทธ์ Unit Offence นี้ไม่สามารถโจมตีถึงได้ ทำให้มีการเดินโดยที่ไม่ยอมโจมตีทั้งที่สามารถโจมตีตัวละครของฝ่ายปัญญาประดิษฐ์ตัวอื่นที่ยืนขวางอยู่ได้ แต่กลับยอมเดินอ้อมไปด้านหลังเพื่อโจมตีตัวละครเป้าหมายแทน และเนื่องจากฝ่ายปัญญาประดิษฐ์นี้ ใช้เพียงคะแนนจากตาเดินล่วงหน้าเพียงตาเดียวเท่านั้น ทำให้ฝ่ายปัญญาประดิษฐ์นี้ ทำตัวเหมือนกับการเลียนแบบฝ่ายตรงข้ามในตาก่อนหน้า ดังนั้นจึงไม่โจมตีตัวละครใดๆ ตามที่ฝ่ายของผู้เล่นกระทำ แต่เมื่อฝ่ายของผู้เล่นสามารถโจมตีถึงตัวเป้าหมายได้แล้ว พอทำการโจมตีเป้าหมาย ตาเดินถัดมา

ฝ่ายปัญญาประดิษฐ์จึงทำการโจมตีใส่เป้าหมายตาม แต่เนื่องจากการโจมตีของปัญญาประดิษฐ์แบบปรับตัวได้นี้ ไม่ได้มีการจำกัดว่า ต้องโจมตีเป้าหมายตัวเดียว ดังเช่นฝ่ายของผู้เล่น เพียงแต่เป็นการพยายามทำให้คะแนนที่จะเกิดขึ้นสูสีกับคะแนนที่ฝ่ายตรงข้ามทำได้ ทำให้เมื่อตัวละครของฝ่ายปัญญาประดิษฐ์นี้ตายไปจากการรุมของฝ่ายของผู้เล่น จะเกิดข้อเสียเปรียบด้านตัวละครขึ้นทันที (ฝ่ายปัญญาประดิษฐ์ไม่ได้บังคับโจมตีตัวละครตัวใดตัวหนึ่ง จึงมักไม่สามารถฆ่าตัวละครฝ่ายของผู้เล่นได้เร็วเท่าที่ทางฝ่ายของผู้เล่นฆ่าได้) และทำให้ฝ่ายปัญญาประดิษฐ์โดนฝ่ายตรงข้ามโจมตีจนเสียหายหนักได้ สำหรับการแข่งกับฝ่ายของผู้เล่นที่ใช้กลยุทธ์แบบ Defense นั้น เนื่องจากพฤติกรรมที่คล้ายกับการเลียนแบบของฝ่ายปัญญาประดิษฐ์ทำให้ต่างฝ่ายต่างไม่ยอมเข้าหากันเป็นส่วนใหญ่ จึงทำให้เกมส่วนใหญ่นั้น ไม่จบลงเพราะไม่มีการเริ่มกระทำของทั้งสองฝ่ายนั่นเอง



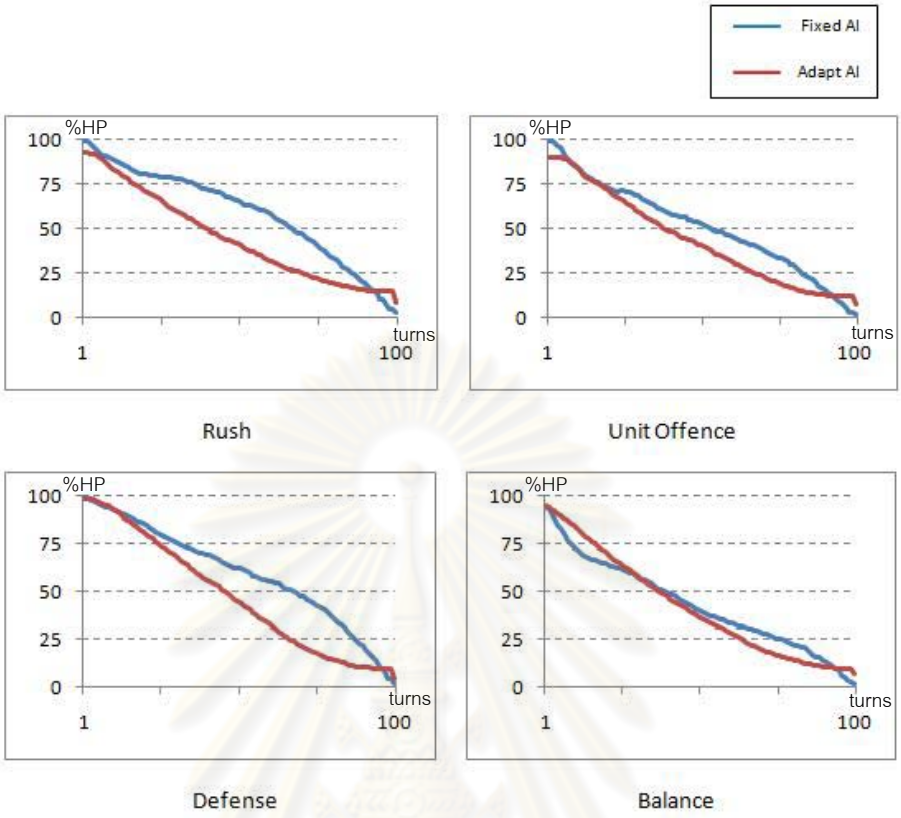
รูปที่ 29 แสดงค่า HP เฉลี่ยที่เหลือของตัวละครในแต่ละตาเดินของปัญญาประดิษฐ์แบบมองตาล่วงหน้า 1 ตาเท่านั้น



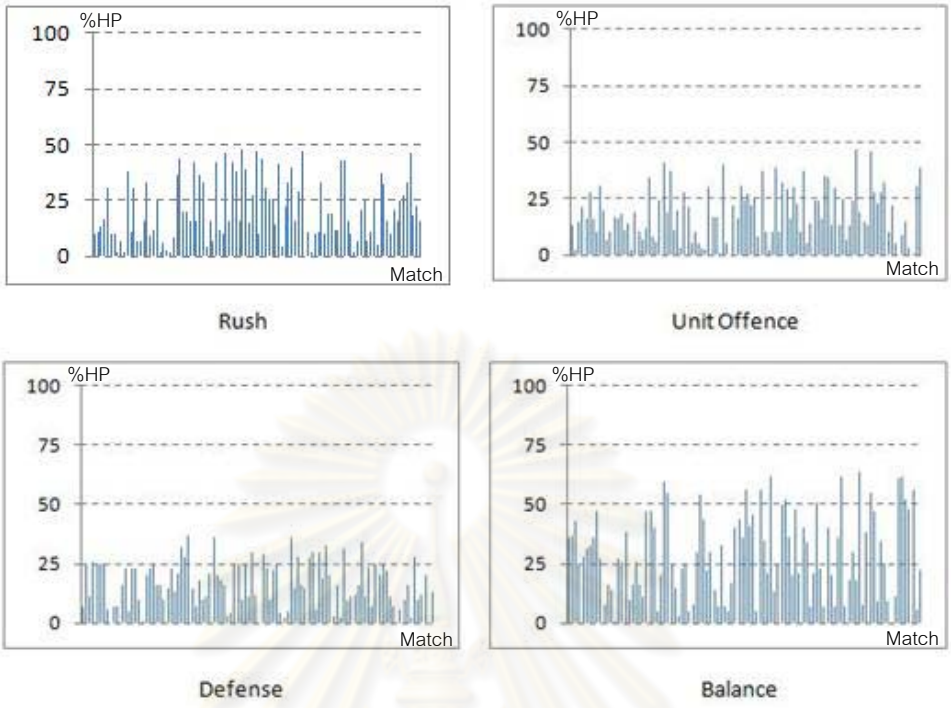
รูปที่ 30 แสดงผลการทดลองของปัญญาประดิษฐ์แบบมองตาล่วงหน้า 1 ตาเท่านั้น

5.2.2 ปัญญาประดิษฐ์แบบปรับตัวได้แบบสนใจมองคะแนนของตาก่อนหน้าที่เคยเกิดขึ้นมาแล้วประมาณ 3 ตา ผลการทดลองนั้นเป็นดังรูปที่ 31 และรูปที่ 32 ซึ่งรูปที่ 31 นั้นจะแสดงค่า HP ที่เหลืออยู่ของแต่ละทีมในการเล่น 1 เกม โดยจากรูปแนวแกน y จะเป็นค่าร้อยละของ HP ที่เหลืออยู่ของทั้งสองทีม ส่วนแนวแกน x จะเป็นจำนวนตาเดิน ซึ่งได้ผ่านการปรับให้เป็นค่ามาตรฐานเท่ากับ 100 ตาเดิน เนื่องจากแต่ละเกมมีจำนวนตาเดินไม่เท่ากัน ทั้งนี้เพื่อสะดวกในการเทียบอัตราส่วนการลดลงของ HP รวมของทีมจากการกระทำของแต่ละฝ่ายนั่นเอง ซึ่งกราฟนี้จะแสดงค่าเฉลี่ยในการเล่นแต่ละตาเดินของเกมการเล่นทั้ง 100 เกมว่าในแต่ละช่วงเวลาของเกมนั้น HP โดยรวมของทีมทั้งสองมีแนวโน้มที่จะลดลงสูสีกันเพียงใด ส่วนรูปที่ 32 จะแสดงผลสุดท้ายของร้อยละของ HP ที่เหลือของทีมที่ชนะในแต่ละเกม โดยจากรูปที่ 32 นั้น แนวแกน y ของกราฟ คือ ค่าร้อยละของ Hp ที่เหลืออยู่ของฝ่ายที่ชนะในเกมนั้นๆ ส่วนแนวแกน x คือ เกมที่ทำการต่อสู้กันแต่ละเกม ซึ่งจะเห็นได้ว่าการต่อสู้กับฝ่ายของผู้เล่นที่ใช้กลยุทธ์การต่อสู้แบบ Rush นั้นผลที่ได้จะค่อนข้างแย่ลงกว่าแบบแรกดังเช่นแสดงในตารางที่ 2 แต่ค่าเฉลี่ยของค่า Hp ที่เหลือนั้นก็ยังอยู่

ในช่วงที่ถือว่าสูสีกับฝ่ายตรงข้ามอยู่ ซึ่งสาเหตุที่ผลออกมาแย่ลงนี้ เนื่องมาจากเมื่อทำการเฉลี่ยคะแนนจากฝ่ายของผู้เล่นที่เคยเกิดขึ้นในอดีตประมาณ 3 ตานั้น ทำให้ฝ่ายปัญญาประดิษฐ์นี้ จะเริ่มทำการโจมตีฝ่ายของผู้เล่นช้ากว่าในแบบแรก ยกตัวอย่างเช่นในกรณีที่ฝ่ายของผู้เล่นเดินเข้ามาหาเพื่อจะโจมตี แต่ยังไม่สามารถโจมตีได้เป็นจำนวนประมาณ 3 หรือ 4 ตาเดิน ทำให้คะแนนนั้นอยู่ในช่วงหนึ่ง ซึ่งพอเมื่อฝ่ายของผู้เล่นเริ่มโจมตีแล้ว ซึ่งจะถือว่าสามารถทำคะแนนได้ดีขึ้น แต่ผลจากการมองอดีตที่เคยเกิดขึ้นมา 3 ตาก่อนหน้านี้แล้วนำคะแนนมาเฉลี่ยกัน ทำให้คะแนนเฉลี่ยที่ออกมานั้น จะยังไม่ถึงช่วงคะแนนที่ดีเท่าที่ฝ่ายปัญญาประดิษฐ์นี้จะทำการโจมตีออกมา จึงทำให้ฝ่ายของผู้เล่นค่อนข้างจะได้เปรียบเหมือนกับสามารถโจมตีแบบกินเปล่าได้ก่อนประมาณ 2 หรือ 3 ตาเดิน ซึ่งนั่นเป็นผลทำให้ฝ่ายปัญญาประดิษฐ์ ไม่สามารถรุกกลับได้ทันจึงทำให้โดยเฉลี่ยแล้วผลออกมาไม่ดีเท่ากับในกรณีแรก ส่วนในกรณีกลยุทธ์แบบ Balance นั้นผลเฉลี่ยที่ได้ออกมาจะค่อนข้างใกล้เคียงกับในกรณีแรก สำหรับกรณีของ Unit Offence นั้นผลที่ได้ออกมาดีขึ้น ทั้งนี้เพราะจากการเฉลี่ยค่าที่เกิดขึ้นจากอดีตนี้ เมื่อฝ่ายของผู้เล่นเดินเพื่อมาโจมตีตัวละครของอีกฝ่ายช่วงที่โจมตีนั้น ก็จะโจมตีค่อนข้างต่อเนื่องจนตัวละครนั้นตายไป แล้วจึงพยายามเดินเพื่อไปหาเป้าหมายต่อไป ซึ่งฝ่ายปัญญาประดิษฐ์ในกรณีนี้ยังคงไม่ได้หยุดการโจมตีกระทันหันเมื่อฝ่ายตรงข้ามโจมตีตัวละครหนึ่งตายไปและพยายามเปลี่ยนเป้าหมายไปยังตัวละครต่อไป และไม่ได้โจมตีตัวละครที่ขวางหน้าอย่างเช่นกรณีแรก แต่จะสังเกตจากค่าเฉลี่ยที่ฝ่ายของผู้เล่นเคยทำไว้ได้ดีในตาก่อนหน้า ยังคงค่อนข้างอยู่ในช่วงสูงอยู่ จึงยังคงพยายามทำคะแนนให้ได้ค่อนข้างสูงกลับไป เลยทำให้สามารถสู้ได้อย่างมีความสูสีมากขึ้น สำหรับในส่วนของกลยุทธ์แบบ Defense นั้นเนื่องจากเป็นการใช้คะแนนเฉลี่ยไม่ได้ยึดติดกับค่าก่อนหน้าเท่านั้นดังเช่นกรณีแรก จึงจะมีช่วงของคะแนนที่ทำให้ฝ่ายปัญญาประดิษฐ์มีการขยับเพื่อเข้าหาฝ่ายของผู้เล่นไปบ้างทีละน้อย ซึ่งก็สามารถเล่นได้ค่อนข้างสูสีกับฝ่ายของผู้เล่น แต่เกมก็จะค่อนข้างนานเล็กน้อยเพราะไม่ได้เป็นการก้าวเข้าหากันอย่างรวดเร็ว เพราะฝ่ายปัญญาประดิษฐ์จะพยายามกระทำให้เกิดความสูสีกับฝ่ายของผู้เล่นขึ้นจึงพยายามรักษาระยะห่างไว้และไม่ค่อยบุกเข้าไปมากนักเช่นเดียวกันกับกลยุทธ์ของฝ่ายของผู้เล่น



รูปที่ 31 แสดงค่า HP เฉลี่ยที่เหลือของตัวละครในแต่ละตาเดินของปัญญาประดิษฐ์แบบมองตาที่เคยเกิดขึ้นก่อนหน้าประมาณ 3 ตา



รูปที่ 32 แสดงผลการทดลองของปัญญาประดิษฐ์แบบมองตาที่เคยเกิดขึ้นก่อนหน้าประมาณ 3 ตา

5.2.3 ปัญญาประดิษฐ์ที่มองทั้งคะแนนในตาเดินที่เคยเกิดขึ้นก่อนหน้า และตาเดินที่จะเกิดขึ้นภายหลังประมาณ 3 ตา ผลการทดลองนั้นเป็นดังรูปที่ 33 และรูปที่ 34 ซึ่งรูปที่ 33 นั้นจะแสดงค่า HP เฉลี่ยที่เหลืออยู่ของแต่ละทีมในการเล่น 1 เกม โดยจากรูปแนวแกน y จะเป็นค่าร้อยละของ HP ที่เหลืออยู่ของทั้งสองทีม ส่วนแนวแกน x จะเป็นจำนวนตาเดิน ซึ่งได้ผ่านการปรับให้เป็นค่ามาตรฐานเท่ากับ 100 ตาเดิน เนื่องจากแต่ละเกมมีจำนวนตาเดินไม่เท่ากัน ทั้งนี้เพื่อสะดวกในการเทียบอัตราส่วนการลดลงของ HP รวมของทีมจากการกระทำของแต่ละฝ่ายนั่นเอง ซึ่งกราฟนี้จะแสดงค่าเฉลี่ยในการเล่นแต่ละตาเดินของเกมการเล่นทั้ง 100 เกมว่าในแต่ละช่วงเวลาของเกมนั้น HP โดยรวมของทีมทั้งสองมีแนวโน้มที่จะลดลงสูสีกันเพียงใด ส่วนรูปที่ 34 จะแสดงผลสุดท้ายของร้อยละของ HP ที่เหลือของทีมที่ชนะในแต่ละเกมโดยจากรูปที่ 34 นั้นแนวแกน y ของกราฟ คือ ค่าร้อยละของ Hp ที่เหลืออยู่ของฝ่ายที่ชนะในเกมนั้นๆ ส่วนแนวแกน x คือ เกมที่ทำการต่อสู้กันแต่ละเกม จะเห็นได้ว่าผลการทดลองจะอยู่ในแนวโน้มที่ค่อนข้างดีในการต่อสู้กับฝ่ายของผู้เล่นทั้ง 4 รูปแบบ โดยสำหรับการต่อสู้กับฝ่ายของผู้เล่นในกลยุทธ์การต่อสู้แบบ Rush นั้นจะได้ผลที่มีลักษณะคล้ายกับแบบที่สองแต่ผลค่อนข้างที่จะดีขึ้นเนื่องจาก การทำนายอนาคตที่เกิดขึ้นนั้นยังดูผลจากเหตุการณ์ที่เคยเกิดขึ้นจากในอดีตเป็นหลัก แต่จะสามารถตอบสนองได้ไวขึ้นจากแบบที่

สองเพราะการดูแนวโน้มที่มีความเป็นไปได้ว่าศัตรูจะเก่งขึ้น และทำคะแนนได้ดีขึ้นนั้น จะทำให้ฝ่ายปัญญาประดิษฐ์ทำนายอนาคตได้ว่าฝ่ายของผู้เล่นจะสามารถทำคะแนนได้ดีขึ้น และพยายามจะทำคะแนนให้ดีขึ้นตามผลการทำนายนั่นเอง สำหรับในส่วนของการต่อสู้กับฝ่ายของผู้เล่นที่ใช้กลยุทธ์แบบ Unit Offence และ Defense นั้น ผลที่ออกมาจะมีลักษณะค่อนข้างคล้ายเคียงกับในแบบที่สอง และสำหรับการต่อสู้กับปัญญาประดิษฐ์แบบ Balance นั้นมีแนวโน้มที่จะตอบสนองได้ดีขึ้นกว่าแบบที่สองเล็กน้อย และหากดูจากค่าเฉลี่ยดังตารางที่ 2 จะเห็นได้ว่าฝ่ายปัญญาประดิษฐ์ที่ใช้แนวคิดนี้ จะสามารถต่อสู้กับฝ่ายของผู้เล่นได้อยู่ในเกณฑ์ที่สูสีตามผลจากแบบสอบถามที่เคยได้กล่าวไว้ข้างต้น

รูปที่ 33 แสดงค่า HP เฉลี่ยที่เหลือของตัวละครในแต่ละตาเดินของปัญญาประดิษฐ์แบบมองตาเดินทั้งจากอดีตและอนาคตเป็นจำนวน 3 ตา

รูปที่ 34 แสดงผลการทดลองของปัญญาประดิษฐ์แบบมองตาเดินทั้งจากอดีตและอนาคตเป็นจำนวน 3 ตา

กลยุทธ์ของคู่ต่อสู้	ปัญญาประดิษฐ์แบบมองอดีตเพียงตาเดียวเท่านั้น		ปัญญาประดิษฐ์แบบมองอดีตที่เคยเกิดก่อนหน้า 3 ตา		ปัญญาประดิษฐ์แบบที่มองทั้งอดีตและอนาคต	
	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตรฐาน	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตรฐาน	ค่าเฉลี่ย	ค่าเบี่ยงเบนมาตรฐาน
Rush	18.75	10.28	20.68	13.59	20.14	12.73
Unit Offence	45.07	25.88	18.25	11.55	17.06	8.88
Defense	-	-	16.61	9.64	16.83	8.70
Balance	23.85	10.51	29.29	17.55	19.74	10.90

ตารางที่ 2 แสดงค่าเฉลี่ยของพลังชีวิตที่เหลือ และค่าเบี่ยงเบนมาตรฐานเมื่อใช้ปัญญาประดิษฐ์แบบปรับตัวได้ทั้ง 3 ลักษณะกับคู่ต่อสู้ที่ใช้กลยุทธ์ทั้ง 4 รูปแบบ

บทที่ 6

สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย และข้อเสนอแนะ

วัตถุประสงค์ของงานวิทยานิพนธ์ฉบับนี้ คือเพื่อหาแนวทางในการสร้างปัญญาประดิษฐ์ที่สามารถเล่นเกมวางแผนการรบแบบผลัดกันเล่นได้อย่างสูสีกับผู้เล่นมากขึ้น ซึ่งจากแนวคิดและการทดลอง สามารถสรุปผลการวิจัยได้ดังนี้คือ

ปัญญาประดิษฐ์ที่ใช้แนวคิดทั้ง 3 ประเภทในการพิจารณาคะแนนของผู้เล่นนั้นจะมีลักษณะจุดเด่น และจุดด้อยบางอย่างที่แตกต่างกัน โดยปัญญาประดิษฐ์ที่ใช้แนวคิดแรกนั่นคือการตัดสินใจจากเหตุการณ์ที่เกิดขึ้นล่วงหน้าเพียงตาเดียว จะมีการกระทำที่คล้ายกับการเลียนแบบผู้เล่นในตาเดินก่อนหน้านั้น จึงจะทำให้สามารถตอบสนองกับกลยุทธ์การต่อสู้แบบ Rush ที่ทำการเดินหน้าเข้าโจมตีอย่างเดียวได้ค่อนข้างดี เพราะว่าไม่ค่อยมีเทคนิคการต่อสู้ที่ซับซ้อนเท่าใดนัก แต่เมื่อเจอกับกลยุทธ์ที่มีการวางแผนที่ดีขึ้นและเฉพาะเจาะจงอย่างเช่นกลยุทธ์การต่อสู้แบบ Unit Offense ก็จะทำให้ไม่สามารถตอบสนองได้ดี เพราะเมื่อกลยุทธ์แบบ Unit Offence เน้นการโจมตีอย่างเฉพาะเจาะจงไปที่การกำจัดตัวละครตัวใดตัวหนึ่งได้แล้ว จะทำให้เกิดข้อได้เปรียบในเรื่องตัวละครขึ้น รวมทั้งข้อได้เปรียบในเรื่องตัวละครนี้ทำให้เกิดข้อได้เปรียบทางจำนวนตาเดินที่มีมากกว่าอีกด้วย หรือในกรณีกลยุทธ์การต่อสู้แบบ Defense ก็จะทำให้เกมเกิดการชะงัก เพราะไม่สามารถทำการต่อสู้ให้จบเกมได้ด้วยนั่นเอง สำหรับปัญญาประดิษฐ์ในแนวคิดที่สอง ซึ่งก็คือการมองลงไปยังอดีตที่เคยเกิดขึ้น แล้วนำคะแนนมาช่วยในการพิจารณานั้น ก็จะสามารถนำมาประยุกต์ใช้กับกลยุทธ์แบบต่างๆ ได้ดีขึ้น แต่จะมีข้อเสียเล็กน้อยในเรื่องของการต่อสู้กับกลยุทธ์ในแบบ Rush ที่จะทำให้ตอบสนองได้ช้ากว่าในแบบแรก เพราะคิดจากค่าคะแนนเฉลี่ยในอดีต ทำให้ผลของคะแนนก่อนที่ผู้เล่นจะเริ่มโจมตีมาเฉลี่ยกับคะแนนเมื่อทำการเริ่มโจมตีในช่วงแรกๆได้คะแนนต่ำ ปัญญาประดิษฐ์ในแนวคิดนี้จึงยังตอบสนองต่อการโจมตีมาช่วงแรกๆ ไม่ทันนั่นเอง สำหรับปัญญาประดิษฐ์ในแนวคิดที่สาม นั่นคือการมองตาเดินทั้งในอดีตและอนาคตนั้น เป็นเหมือนการปรับปรุงแนวคิดที่สองให้มีการทำนายในส่วนของอนาคตเพิ่มเติมขึ้น เพื่อช่วยในการดูคะแนน และเลือกกระทำในสิ่งที่น่าจะทำให้การต่อสู้นั้นมีความสูสีมากขึ้น ซึ่งในส่วนของการมองอนาคตนี้จะเป็นเพียงการทำนายโดยอ้างอิงจากข้อมูลที่เคยเกิดขึ้นของผู้เล่นในอดีต โดยอาศัยแนวโน้มที่ควรจะเป็น ซึ่งการอาศัยแนวโน้มในอดีตมาทำนายอนาคตของผู้เล่นนี้จะช่วยแก้ไขข้อบกพร่องที่ทำให้ปัญญาประดิษฐ์ของแนวคิดที่สองตอบสนองช้าไป ดังตัวอย่างของการต่อสู้กับผู้เล่นที่ใช้กลยุทธ์แบบ Rush เป็นต้น และทำให้สามารถเล่นได้อย่างสูสีกับผู้เล่นมากขึ้นนั่นเอง

ในการนำเอาแนวคิดและวิธีการของวิทยานิพนธ์นี้ไปใช้กับเกมวางแผนการรบแบบผลัดกันเล่นในท้องตลาดจริงนั้น สามารถนำไปใช้ได้เลย แต่จะต้องมีการปรับค่าน้ำหนักที่อยู่ในสมการส่วนต่างๆ ให้มีความสอดคล้องกับสถานะของตัวละคร และวิธีการคิดค่าความเสียหายต่างๆ ของเกมนั้นๆ นอกจากนี้เกมแต่ละเกมที่มีวางขายจริงอาจมีรายละเอียดปลีกย่อยอื่นๆที่มีผลกระทบต่อการเล่นเพิ่มเข้ามา ยกตัวอย่างเช่น การโจมตีเพื่อทำให้เกิดค่าสถานะผิดปกติกับตัวละครที่ฝ่ายตรงข้าม เช่น ทำให้ติดสถานะพิษ ซึ่งทำให้ค่า Hp จะลดลงทุกครั้งเมื่อถึงช่วงเวลาที่ตัวละครนั้นจะต้องเดิน หรือทำให้ติดสถานะเป็นใบ้ ซึ่งทำให้ตัวละครไม่สามารถกระทำการใดๆ ที่เป็นส่วนของการใช้เวทมนต์ได้ หรือทำให้ติดสภาวะเชื่องช้า ซึ่งทำให้ค่า Speed ของตัวละครลดลงไปชั่วขณะหนึ่ง ทำให้ตัวละครถึงตาเดินได้ช้าลง นอกจากนี้อาจมีการเพิ่มสถานะบางอย่างให้กับตัวละครฝ่ายเดียวกัน เช่น ทำให้พลังป้องกันเพิ่มขึ้นในชั่วขณะ ทำให้พลังโจมตีเพิ่มขึ้นชั่วขณะ หรือทำให้ค่า Speed ของตัวละครเพิ่มขึ้นชั่วขณะ เป็นต้น ซึ่งข้อมูลรายละเอียดปลีกย่อยต่างๆ เช่นนี้อาจมีผลทำให้เกิดกลยุทธ์รูปแบบต่างๆ มากมายในการต่อสู้ ซึ่งอาจมีผลทำให้การเล่นกับปัญญาประดิษฐ์ตามที่ได้ทดลองในวิทยานิพนธ์เล่มนี้แย่ลงในบางกลยุทธ์ ในการปรับวิธีการของวิทยานิพนธ์นี้ให้สามารถใช้ได้กับรายละเอียดปลีกย่อยต่างๆเหล่านี้ ผู้พัฒนาเกมสามารถนำข้อมูลจากรายละเอียดต่างๆนี้มาคิดได้ในส่วนของการคิดค่าคะแนนของสภาวะแวดล้อมอื่นๆ (en) ในฟังก์ชันที่เป็นส่วนของการคิดคะแนน การที่ไม่มีการคิดในส่วนของรายละเอียดปลีกย่อยต่างๆ เหล่านี้ลงในวิทยานิพนธ์นี้แต่แรก เนื่องจากว่าเกมวางแผนการรบแบบผลัดกันเล่นที่มีอยู่ในท้องตลาดจริงนั้นมีอยู่มากมาย และแต่ละเกมก็มีรายละเอียดปลีกย่อยต่างๆ เหล่านี้ต่างกันออกไปอีกด้วย

วิทยานิพนธ์นี้ วัดผลการทดลองความสูสีจากสถานะเริ่มต้นที่ตัวละครทั้งสองฝ่ายมีจำนวนเท่ากันและค่าสถานะต่างๆเท่ากัน สำหรับในกรณีที่ตัวละครของทั้งสองฝ่ายมีจำนวนไม่เท่ากันนั้นจะต้องมีการปรับปรุงในส่วนของการเลือกคะแนนเพื่อให้เกิดความสูสีมากขึ้น โดยอาจทำการเลือกระดับความสูสีให้มีความสูงขึ้นหากฝ่ายที่ใช้ปัญญาประดิษฐ์แบบต้องการให้เล่นได้อย่างสูสีนี้ มีจำนวนตัวละครน้อยกว่าอีกฝ่าย ในทางกลับกันหากฝ่ายปัญญาประดิษฐ์มีตัวละครมากกว่าอีกฝ่ายอาจมีการปรับให้เลือกค่าระดับความสูสีให้อยู่ในช่วงระดับที่น้อยลง เป็นต้น และหากในกรณีที่ค่าสถานะของแต่ละฝ่ายไม่เท่ากัน ก็สามารถปรับเปลี่ยนค่าน้ำหนักของแต่ละตัวละคร (W_c) ให้เหมาะสม เพื่อให้สามารถเล่นได้อย่างสูสีขึ้นนั่นเอง

รายการอ้างอิง

[1] Adams, E. and Rollings, A.. Fundamentals of Game Design. Printed in Harrisonburg : Pearson Education, Inc., 2007.

[2] Bakkes, S., Spronck, P. and Herik, V. J.. Phase-dependent Evaluation in RTS games. Proceedings of the 19th Belgian-Dutch Conference on Artificial Intelligence (eds. Mohammad Mehdi Dastani and Edwin de Jong) 2007 : pp. 3-10.

[3] Bergsma, M. and Spronck, P.. Adaptive Intelligence for Turn-based Strategy Game. Proceedings of the twentieth Belgian-Dutch Artificial Intelligence Conference (eds. Anton Nijholt, Maja Pantic, Mannes Poel, and Hendri Hondrop) 2008 : pp. 17-24.

[4] Bergsma, M. and Spronck, P.. Adaptive Spatial Reasoning for Turn-based Strategy Games. Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference (eds. Michael Mateas and Chris Darken) 2008 : pp. 161-166.

[5] Bourg, D. and Seemann, G.. AI for Game Developers. First Edition. Printed in the United States of America : O'Reilly Media, Inc., 2004

[6] David, L.. Computer Chess Compendium. Printed in United States of America : Ishi Press International, 2009.

[7] Electronic Arts. Red Alert Universe > Command & Conquer Red Alert 3 [Online]. Available from: http://portal.commandandconquer.com/portal/site/redalert/redalert3. [2009, 16 Oct].

[8] Goles, E. and Martinez, S.. Cellular Automata and Complex Systems. First Edition. Springer, 1998

[9] Ilachinski, A.. Artificial War: Multiagent-Based Simulation of Combat. Printed in Singapore : World Scientific Publishing Co. Pte. Ltd., 2004.

[10] Laramée, F. D.. GameDev.net – Chess Programming PartVI: Evaluation Functions [Onlline]. Available from: http://www.gamedev.net/reference/articles/article1208.asp. [2009, 16 Oct].

[11] Nintendo. Fire Emblem: The Sacred Stones for Game boy Advance [Online]. Available from: http://www.fire-emblem.com/sacredstones/launch/index.html. [2009, 16 Oct].

[12] Spronck, P. and Herik, V. J.. A Tutoring System for Commercial Games. ICEC 2005 (eds. Fumio Kishino, Yoshifumi Kitamura, Hirokazu Kato, and Noriko Nagata), Lecture Notes in Computer Science 3711 2005 : pp. 389-400.

[13] Spronck, P., Sprinkhuizen-Kuyper, I. and Postma, E.. Difficulty Scaling of GameAI. GAME-ON 2004: 5th Internatinal Conference on Intelligent Games and Simulation (eds. Abdennour El Rhalibi and Danny Van Welden) 2004 : pp. 33-37.

[14] Spronck, P., Sprinkhuizen-Kuyper, I. and Postma, E.. Enhancing the Performance of Dynamic Scripting in Computer Games. Entertainment Computing – ICEC 2004 (ed. Matthias Rauterberg), Lecture Notes in Computer Science 3166. 2004 : pp. 296-307.

[15] Spronck, P., SprinkHuizen-Kuyper, I. and Postma, E.. Online Adaptation of Game Opponent AI with Dynamic Scripting. International Journal of Intelligent Games and Simulation (eds. N.E. Gough and Q.H. Mehdi) 2004, Vol.3 : pp. 45-53.

[16] SQUARE ENIX. FINAL FANTASY TACTICS A2: Grimoire of the Rift | SQUARE ENIX [Onine]. Available from: http://na.square-enix.com/ffta2. [2009, 16 Oct].

[17] SQUARE ENIX. Final Fantasy Tactics Advance [Online]. Available from: http://www.square-enix-usa.com/game/FFT-A. [2009, 16 Oct].

[18] SQUARE ENIX. FINAL FANTASY TACTICS: THE WAR OF THE LIONS | SQUARE ENIX [Online]. Available from: http://na.square-enix.com/fftactics. [2009, 16 Oct].

[19] Straatman, R., Sterren, W. and Beij, A.. Killzone's AI: dynamic procedural combat tactics. Game Developers Conference (GDC) 2005.

[20] บุญเสริม กิจศิริกุล. ปัญญาประดิษฐ์ เอกสารคำสอนวิชา 2110654. 2548 : pp. 1-5.

ภาคผนวก

ภาคผนวก ก

รายละเอียดอื่น ๆ เกี่ยวกับเกมที่ใช้ในการทดสอบ

1. ข้อมูลความสามารถ และค่าสถานะของแต่ละตัวละคร

ค่าสถานะของตัวละครที่ใช้ในงานวิจัยนี้ อ้างอิงมาจากช่วงหนึ่งของค่าสถานะของตัวละครในเกม Final Fantasy Tactics A2: Grimoire of the Rift ซึ่งในกระบวนการทดลองนั้น แต่ละฝ่ายนั้นจะมีตัวละครจำนวนเท่ากัน คือฝ่ายละ 4 ตัวละคร โดยแบ่งออกเป็นสายอาชีพทั้งหมด 4 อาชีพ ได้แก่ อาชีพ Warrior อาชีพ Archer อาชีพ White Mage และ อาชีพ Black Mage ซึ่งแต่ละอาชีพจะมีค่าสถานะดังตารางที่ 3

Job	Warrior	Archer	Black Mage	White Mage
HP	134	119	106	110
MP	22	25	68	50
Move	4	4	3	3
Attack	66	52	43	48
Defense	57	50	45	47
Magic	24	30	35	31
Magic Resistance	31	51	87	79
Evade	12	20	0	0
Speed	65	70	60	62
Attack Range	1	4	1	1
Magic Range	0	0	4	4
MP Use	0	0	8	8

ตารางที่ 3 แสดงรายละเอียดของค่าสถานะต่างๆ ของตัวละครในแต่ละอาชีพ

2. แผนที่ที่ใช้ในการทำการทดลอง

แผนที่ที่ได้ใช้ในการทดลองนี้จะนำมาใช้โดยอ้างอิงจากแผนที่ของฉากหนึ่งในเกม Final Fantasy Tactics A2: Grimoire of the Rift ด้วยเช่นกัน โดยจะแสดงดังตารางที่ 4 โดยจุด x ในรูปคือตำแหน่งที่ไม่สามารถเดินผ่านได้ ส่วนจุด o คือจุดเริ่มต้นของตัวละครทั้งสองฝ่าย สำหรับพื้นที่ที่เหลือคือ ตำแหน่งที่สามารถเดินตัวละครไปตามตำแหน่งนั้นๆ ได้

	0	1	2	3	4	5	6	7	8	9	10	11
0	X		O	O	O	O	X				X	X
1												X
2												X
3												
4												
5	X	X	X	X			X			X	X	X
6												X
7												X
8											X	X
9	X									X	X	X
10	X	X	X	O	O	O	O	X	X	X	X	X

ตารางที่ 4 แสดงแผนที่ที่ใช้ในการทดลอง

โดยตำแหน่งที่วางตัวละครอาชีพต่างๆ ได้แก่

ตำแหน่ง	อาชีพตัวละคร	ทีม
(0,2)	Black Mage	Blue
(0,3)	Archer	Blue
(0,4)	Warrior	Blue
(0,5)	White Mage	Blue
(10,3)	Black Mage	Red
(10,4)	Archer	Red
(10,5)	Warrior	Red
(10,6)	White Mage	Red

ตารางที่ 5 แสดงตำแหน่งในการวางตัวละครเริ่มต้นในแผนที่

3. ค่าพารามิเตอร์ต่างๆ ที่ใช้ในการทดลองในเกมที่ใช้ในการทดสอบ

ค่าน้ำหนักของตัวละครในอาชีพต่างๆ ที่ใช้ในเกม (W_c) ได้แก่

ค่าน้ำหนักของตัวละครอาชีพ Warrior หรือ $W_{c\_Warrior}$ = 1.3

ค่าน้ำหนักของตัวละครอาชีพ Archer หรือ W_{c_Archer} = 1.4

ค่าน้ำหนักของตัวละครอาชีพ Black Mage หรือ $W_{c\_Black\ Mage}$ = 1.5

ค่าน้ำหนักของตัวละครอาชีพ White Mage หรือ $W_{c\_White\ Mage}$ = 1.3

ค่าน้ำหนักของการคิดคะแนนระยะทางที่ใช้ในเกม (W_d) ซึ่งจะแบ่งออกเป็น

กรณีที่ใช้เมื่อคิดคะแนนตัวละครฝ่ายเดียวกันจะได้ว่า W_d = -1.2

กรณีที่ใช้เมื่อคิดคะแนนตัวละครฝ่ายตรงกันข้ามจะได้ว่า W_d = -1.4

ค่าน้ำหนักในส่วนของการเข้าเงื่อนไขเกี่ยวกับการพิจารณาค่าคะแนนระยะห่างของตัวละคร (W_{d_HP}) ซึ่งจะเป็นตัวเปลี่ยนแปลงค่าให้กับคะแนนในส่วนที่เป็นกรณีเข้าเงื่อนไขที่กำหนด ซึ่งเงื่อนไขนั้นคือ หาก Hp ของตัวละครที่กำลังพิจารณานี้มีค่ามากกว่าหรือเท่ากับครึ่งหนึ่งของ Hp

ทั้งหมดของตัวละคร หรือตัวละครนั้นสามารถจะโจมตีฝ่ายตรงข้ามให้ตายได้หากเคลื่อนที่ไปยังตำแหน่งนั้น จะนำค่า W_{d_HP} นี้มาคูณเพิ่มให้กับค่าคะแนนของฝ่ายเดียวกัน (เพื่อให้ค่าคะแนนเมื่อเข้าใกล้ฝ่ายตรงข้ามมีค่าเหนือกว่าขึ้นอีกเล็กน้อย เพราะเป็นช่วงที่ยังสามารถบุกโจมตีได้อย่างมีประสิทธิภาพอยู่) แต่ในกรณีนอกเหนือจากนั้นให้นำค่า W_{d_HP} นี้มาคูณเพิ่มให้กับค่าคะแนนของฝ่ายตรงข้าม (เพื่อให้ค่าคะแนนเมื่อเข้าใกล้พวกเดียวกันมีค่าเหนือกว่าขึ้นมาเล็กน้อย) ดังตัวอย่างดังต่อไปนี้

```
if ((thisCharacter.getHp() >= (thisCharacter.getMaxHp()/2)) ||
(thisCharacter.attackDmg(targetCharacter) >= targetCharater.getHp()))
{
      scoreAlly = scoreAlly * W_D_HP;
} else {
      scoreEnemy = scoreEnemy * W_D_HP;
}
```

ซึ่งค่า W_{d_HP} นี้มีค่า = 1.2

ค่าน้ำหนักของค่าคะแนนความเสียหาย (W_{ds}) จะแบ่งออกเป็นสองประเภทคือ ความเสียหายด้านกายภาพ (W_{ds_phy}) และ ความเสียหายด้านเวทมนต์ (W_{ds_mag})

ค่าน้ำหนักของความเสียหายด้านกายภาพ (W_{ds_phy}) = 10.3

ค่าน้ำหนักของความเสียหายด้านเวทมนต์ (W_{ds_mag}) = 6.0

ค่าน้ำหนักหากตัวละครที่ถูกโจมตีนั้นตาย (W_{Dead}) = 3.0

ค่าน้ำหนักของคะแนนพลังชีวิต (W_{HP}) = 0.2

ค่าน้ำหนักของคะแนนการฟื้นพลัง (W_{Heal}) = 7.3

ค่าช่วงคะแนนของความสูสี (Range of e(x,y)) = 250

ภาคผนวก ข

แบบสอบถามและผลการสอบถามที่ใช้ในการทดลอง

แบบสอบถามเกี่ยวกับความคิดเห็นของเกมแนว Turn-based strategy (TBS) เรื่องเกี่ยวกับ จำนวนค่าทรัพยากรคงเหลือที่เห็นว่าสูสี และขอบเขตการมองล่วงหน้า

1. คุณคิดว่าจำนวนทรัพยากรที่เหลืออยู่ของฝ่ายชนะ (ค่าพลังชีวิต หรือ HP) ควรจะมีจำนวนเหลือเป็นปริมาณประมาณกี่เปอร์เซ็นต์จาก ที่มีตอนแรกทั้งหมด (คิดเป็นค่าพลังชีวิตโดยรวม) จึงจะถือว่าเล่นได้อย่างสูสีกัน

(ตัวอย่างเช่น มีตัวละครทั้งหมด 5 ตัว แต่ละตัวมี HP เต็มอยู่ที่ 100 ดังนั้น HP รวมของทั้งกลุ่มจึงมีค่าเป็น 500 เมื่อทำการเล่นจนจบฝ่ายเราเหลือตัวละคร 2 ตัว โดยมี HP เหลือตัวละ 50 รวมเป็น 100 ดังนั้น HP ที่เหลือจะมีปริมาณประมาณ 20% โดยขณะทำการเล่นเกมนั้นเราจะไม่สามารถทำการสร้างตัวละครเพิ่มได้)

_____

_____

_____

2. ในการวางแผนการเล่นนี้ คุณได้ทำการวางแผนการเล่นล่วงหน้าโดยประมาณเป็นจำนวนกี่ตาล่วงหน้า (ในที่นี้ให้ตอบเป็นปริมาณของตาที่มองล่วงหน้าที่ใช้ส่วนใหญ่โดย ทั่วไปอย่างคร่าวๆ อย่างเช่น 1 ตา, 2 ตา, 3 ตา หรือ 4 ตา เป็นต้น)

_____

_____

_____

ผลของแบบสอบถามเรื่องเกี่ยวกับ จำนวนค่าทรัพยากรคงเหลือที่เห็นว่าสูสี และขอบเขตการมองล่วงหน้า มีดังนี้

%HP left	Turn to predict
22.5	3
10	1
40	1
10.5	10
35	3
10	1
10	2
15	4.5
10	2
25	3
10	2
50	2
10	1
10	2
10	2.5
45	4
45	4
10	1.5

ตารางที่ 6 แสดงผลของแบบสอบถามเกี่ยวกับเรื่องร้อยละของ HP ที่เหลือและจำนวนตาที่ดูล่วงหน้า

ซึ่งค่าเฉลี่ยของร้อยละของพลังชีวิตที่เหลือที่เห็นว่าสูสีมีค่าเท่ากับ 21.00 โดยมีค่าเบี่ยงเบนมาตรฐานเท่ากับ 16.21 สำหรับค่าเฉลี่ยของจำนวนตาที่มองล่วงหน้านั้นมีค่าเท่ากับ 2.75 โดยมีค่าเบี่ยงเบนมาตรฐานเท่ากับ 1.10

แบบสอบถามเกี่ยวกับความคิดเห็นของเกมแนว Turn-based strategy (TBS) เรื่องเกี่ยวกับ การคิดคะแนนในด้านต่างๆ ของตัวละคร

สำหรับลักษณะพื้นฐานโดยทั่วไปของเกมจะมีตัวละครทั้งหมด 4 ประเภท โดยแบ่งเป็น 4 อาชีพพื้นฐาน ได้แก่

- warrior เน้นด้าน hp และพลังโจมตีด้าน physical attack ไม่มีการโจมตีด้านเวทมนต์ พลังป้องกันค่อนข้างสูง แต่พลังป้องกันเวทมนต์น้อย

- white mage มี hp และพลังโจมตีด้าน physical attack ต่ำ พลังป้องกันต่ำ พลังป้องกันเวทมนต์ปานกลาง และสามารถเติม hp ให้กับฝ่ายเดียวกันได้

- black mage มี hp และพลังโจมตีด้าน physical attack และพลังป้องกันค่อนข้างน้อย แต่มีพลังโจมตีด้านเวทมนต์ และพลังป้องกันเวทมนต์สูง

- archer มี hp และพลังโจมตีด้าน physical attack พลังป้องกัน มีพลังโจมตีด้านเวทมนต์ และพลังป้องกันเวทมนต์ในระดับปานกลาง แต่สามารถโจมตีได้ในระยะค่อนข้างไกล

ซึ่งในเกมนี้แต่ละฝ่ายจะมีตัวละครอยู่ทั้ง 4 ประเภท โดยจะมีประเภทละ 1 ตัว

สำหรับลักษณะการเดินของตัวละครนั้นจะมีลักษณะคล้ายกับเกม Final Fantasy Tactics สำหรับลักษณะการเทิร์นของตัวละครนั้นจะมีลักษณะเดียวกับเกม Final Fantasy Tactics เช่นกัน โดยที่ตัวละครที่มี speed สูงจะได้เทิร์นก่อน และเทิร์นทีละตัวไม่ใช่การเทิร์นแบบทีละฝ่าย

สำหรับคำถามความคิดเห็นมีดังนี้

1. ปัจจัยที่คุณใช้ดูจริงๆ ณ ขณะที่ทำการเล่นนั้นมีด้านในบ้าง (ตัวอย่างเช่น hp mp พลังโจมตี พลังโจมตีเวทมนต์ ความเร็ว ฯลฯ)

_____
_____
_____
_____

2. ปัจจัยเกี่ยวกับตัวละครฝ่ายเดียวกันที่คุณใช้ดูจริงๆ ณ ขณะที่ทำการเล่นนั้นมีด้านในบ้าง (ตัวอย่างเช่น hp mp พลังโจมตี พลังโจมตีเวทมนต์ ความเร็ว ฯลฯ)

_____
_____
_____
_____

3. ปัจจัยเกี่ยวกับตัวละครฝ่ายศัตรูที่คุณใช้ดูจริงๆ ณ ขณะที่ทำการเล่นนั้นมีด้านในบ้าง (ตัวอย่างเช่น hp mp พลังโจมตี พลังโจมตีเวทมนต์ ความเร็ว ฯลฯ)

_____
_____
_____
_____

4. จากปัจจัยที่ใช้ในการคิดในข้อที่ 1 นั้น**<u>โดยส่วนใหญ่</u>**แล้วมี ลำดับการให้ความสำคัญกับแต่ละค่าที่ดูอย่างไรบ้าง โปรดระบุเป็นคะแนน ซึ่งถือว่าคะแนนเต็ม 10 (ตัวอย่างเช่นให้คะแนนความสำคัญของ hp 8 คะแนน พลังโจมตี 7 คะแนน เป็นต้น โดยที่อาจเป็นการให้คะแนนโดยที่เป็นการทำนายของตาล่วงหน้าด้วย) โดยบรรยายให้ละเอียด (แสดงขั้นตอนการคิดก็ได้)

ซึ่งลักษณะการคิดค่าของการทำ damage ให้กับฝ่ายตรงข้ามจะมีลักษณะ ดังนี้

ค่าความเสียหายที่เกิดจากการโจมตีด้าน physical = พลังโจมตีทาง physical - (0.5 * พลังป้องกันทาง physical)

ค่าความเสียหายที่เกิดจากการโจมตีด้าน magic = (2 * พลังโจมตีทาง magic) - (0.5 * พลังป้องกันทาง magic)

ค่าที่สามารถเติมพลังได้จากเวทมนต์ (Heal) = 1.5 * พลังโจมตีทาง magic

ผลของแบบสอบถามเกี่ยวกับการคิดคะแนนในด้านต่างๆ ของตัวละคร มีดังตารางที่ 7

	Hp	Mp	Move	Evade	Attack Range	Magic Range	Attack Damage	Magic Damage	Defend	Resistance	Speed
	10	5.5	6	7	4	8	7.5	9	9	9	10
	9	7	6	1	8	8	9	8	9	9	9
	7	1	1	7	1	8	8	1	8	5	9
	7	1	1	1	1	8	10	1	5	1	10
	4	9	10	1	8	7	9	10	5	6	7
	4	9	1	0	6		1	10	5	6	7
	4	6	2	5			1	7	6	8	10
	6	6					7	10	7	5	8
	10	9					10	10	7	6	8
	8	6					7.5	6	6	9	6
	8	10					9	9	3	2	
	8	7					6	7			
	10	6					10	7			
	9						5				
mean	7.43	6.35	4.17	2.71	4.71	7.8	7.14	7.31	6.36	6	8.4

ตารางที่ 7 แสดงผลของแบบสอบถามเกี่ยวกับการคิดคะแนนในด้านต่างๆ ของตัวละคร

ภาคผนวก ค

อภิธานศัพท์สำหรับตัวแปรในสมการที่ใช้ในระบบ

%Evade = ค่าร้อยละที่จะหลบได้ของตัวละครฝ่ายที่ถูกโจมตี

$\beta(d_c, ds_{(action,c)}, en_c)$ = ฟังก์ชันในการคำนวณค่าคะแนนจากการกระทำกับตัวละครฝ่ายศัตรู โดยใช้ค่าคะแนนระยะทาง (d_c) จากตัวละครฝ่ายศัตรู c คะแนนความเสียหาย ($ds_{(action,c)}$) เมื่อกระทำความเสียหายโดยการกระทำ action ให้กับตัวละครเป้าหมาย c และสภาวะแวดล้อม (en) อื่นๆ ที่เกี่ยวข้องกับตัวละคร c นั้น

$\gamma(d_c, ds_{(action,c)}, en_c)$ = ฟังก์ชันในการคำนวณค่าคะแนนจากการกระทำกับตัวละครฝ่ายเดียวกัน โดยใช้ค่าคะแนนระยะทาง (d_c) จากตัวละครฝ่ายเดียวกัน c คะแนนความเสียหาย ($ds_{(action,c)}$) เมื่อกระทำความเสียหายโดยการกระทำ action ให้กับตัวละครเป้าหมาย c และสภาวะแวดล้อม (en) ที่เกี่ยวข้องกับตัวละคร c นั้น

$\delta(c,x,y)$ = ฟังก์ชันในการคำนวณค่าคะแนนระยะทางจากตัวละคร (c) มายังช่อง (x,y) ในแผนที่

$\phi(\mu)$ = ฟังก์ชันของการหาค่าคะแนนของฝ่ายตรงข้ามที่เกิดขึ้นมาก่อนถึงตาเดินที่กำลังพิจารณา

ally = เซตของตัวละครที่อยู่ฝ่ายเดียวกันทั้งหมด

Attack = ค่าที่ใช้แสดงถึงพลังโจมตีด้านกายภาพของตัวละคร

c_{ally} = ตัวละครที่เป็นฝ่ายเดียวกัน ซึ่งจะเป็นสมาชิกที่อยู่ในเซตของ ally

c_{enemy} = ตัวละครที่เป็นฝ่ายศัตรู ซึ่งจะเป็นสมาชิกที่อยู่ในเซตของ enemy

Ct = ค่าที่ใช้ทดเพื่อคำนวณตาเดินของตัวละคร

d_c = ค่าคะแนนระยะทางจากตัวละครฝ่ายศัตรู c

$ds_{(action,c)}$ = ค่าคะแนนความเสียหาย ds เมื่อกระทำ action กับตัวละคร c

Damage = ค่าความเสียหายที่เกิดขึ้น

Def = ค่าที่ใช้แสดงถึงพลังป้องกันด้านกายภาพของตัวละคร

e(x,y) = ค่าความสูงสีเมื่อตัวละครกระทำการใดๆ อยู่ในช่อง (x,y)

en_c = ค่าคะแนนสภาวะแวดล้อมอื่นๆ ที่เกี่ยวข้องกับตัวละคร c นั้น

enemy = เซตของตัวละครที่อยู่ฝ่ายศัตรูทั้งหมด

Eva = ค่าที่ใช้แสดงถึงโอกาสที่จะสามารถหลบหลีกได้ของตัวละคร

Heal Point = ค่าพลังที่ฟื้นคืนขึ้นมาได้จากการฟื้นพลังนี้

Hp = พลังชีวิตของตัวละคร

HP_c = ค่าพลังชีวิตที่จะเหลือของตัวละคร c อยู่เมื่อทำการเดินนี้จบ

$HP_{Max\ c}$ = ค่าพลังชีวิตสูงสุดของตัวละคร c

Job = อาชีพของตัวละคร

Magic = ค่าที่ใช้แสดงถึงพลังโจมตีด้านเวทมนต์ของตัวละคร

Mov = ค่าที่ใช้กำหนดระยะการเคลื่อนที่ของตัวละคร

Mp = พลังเวทมนต์ของตัวละคร

Name = ชื่อของตัวละคร

Res = ค่าที่ใช้แสดงถึงพลังป้องกันด้านเวทมนต์ของตัวละคร

s(x,y,action,c) = ค่าคะแนนของช่องในแผนที่ตามแนวแกน x และ y โดยการกระทำ action กับตัวละครเป้าหมาย c

s(x,y) = ค่าคะแนนของปัญญาประดิษฐ์ที่สามารถทำได้เมื่อไปกระทำการใดๆ อยู่ที่ช่อง (x,y) นั้น

$Score_{HP_c}$ = ค่าคะแนนของพลังชีวิตที่เหลืออยู่ของตัวละคร

$Score_{Heal_c}$ = ค่าคะแนนการฟื้นพลังเมื่อกระทำการฟื้นพลังให้กับตัวละคร c

$Score_{Status\ effect_c}$ = ค่าคะแนนเมื่อกระทำการต่างๆ ให้มีการเปลี่ยนแปลงค่าทางสถานะของตัวละคร c

Spd = ค่าที่ใช้แสดงความเร็วของตัวละคร

Target's Defense = ค่าพลังป้องกันด้านกายภาพของตัวละครที่ถูกโจมตี

Target's evade = ค่า Eva ของตัวละครที่จะถูกโจมตี

Target's Magic Resistance = ค่าพลังป้องกันด้านเวทมนต์ของตัวละครที่ถูกโจมตี

Team = ฝ่ายของตัวละคร

W_c = ค่าน้ำหนักของตัวละคร c

W_d = ค่าน้ำหนักของคะแนนระยะทาง

W_{Dead} = ค่าน้ำหนักหากการโจมตีทำให้ตัวละครที่ถูกโจมตีนั้น

W_{ds} = ค่าน้ำหนัก (Weight) ของคะแนนความเสียหาย

W_{Heal} = ค่าน้ำหนักของคะแนนของการฟื้นพลัง

W_{HP} = ค่าน้ำหนักของคะแนนพลังชีวิต (Hp)

x = ตำแหน่งตามแนวแกน x

x_0 = ตำแหน่งตามแนวแกน x ของตัวละคร c

y = ตำแหน่งตามแนวแกน y

y_0 = ตำแหน่งตามแนวแกน y ของตัวละคร c

ประวัติผู้เขียนวิทยานิพนธ์

ผู้เขียนเกิดที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยธรรมศาสตร์ในปีการศึกษา 2549 หลังจากนั้นได้เข้าทำการศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2550 และมีผลงานตีพิมพ์ 2 ผลงาน ได้แก่

- Potisartra, K. and Kotrajaras, V. Towards an Evenly Match Opponent AI in Turn-based Strategy Games. Computer Games, Multimedia and Allied Technology 09 2009 : pp. 61-66.
- Potisartra, K. and Kotrajaras, V. An Evenly Matched Opponent AI in Turn-based Strategy Games. Proceeding of 2010 3rd IEEE International Conference on Computer Science and Information Technology 2010 : pp. 42-45.