(โครงร่างวิทยานิพนธ์)

การสร้างรูปแบบพฤติกรรมสำหรับศัตรูในเกมแอ็คชั่นสองมิติแบบเน้นตัวละครโดยอัตโนมัติด้วยเทคนิคเหมืองข้อมูล

โครงร่างวิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต  
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์  
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย  
ปีการศึกษา 2557  
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

นายพิชิต พร้อมสุทธิพงศ์

A Thesis Proposal 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 2014  
Copyright of Chulalongkorn University

Mr. Pichit Promsutipong

(THESIS PROPOSAL)

Automatic enemy behavior pattern generation for 2D avatar-based action game using data mining technique

|  |  |
| --- | --- |
| หัวข้อวิทยานิพนธ์ | การสร้างรูปแบบพฤติกรรมสำหรับศัตรูในเกมแอ็คชั่นสองมิติแบบเน้นตัวละครโดยอัตโนมัติด้วยเทคนิคเหมืองข้อมูล |
| โดย | นายพิชิต พร้อมสุทธิพงศ์ |
| สาขาวิชา | วิศวกรรมคอมพิวเตอร์ |
| อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก | ผู้ช่วยศาสตราจารย์ ดร.วิษณุ โคตรจรัส |
|  |  |
| คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับโครงร่างวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต | |
|  | |
| คณบดีคณะวิศวกรรมศาสตร์  (ศาสตราจารย์ ดร.บัณฑิต เอื้ออาภรณ์) | |
| คณะกรรมการสอบวิทยานิพนธ์ | |
| ประธานกรรมการ  (ผู้ช่วยศาสตราจารย์ ดร.นัทที นิภานันท์) | |
| อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก  (ผู้ช่วยศาสตราจารย์ ดร.วิษณุ โคตรจรัส) | |
| กรรมการ  (อาจารย์ ดร.ณัฐพงศ์ ชินธเนศ) | |
| กรรมการภายนอกมหาวิทยาลัย  (ผู้ช่วยศาสตราจารย์ ดร.จรัสศรี รุ่งรัตนาอุบล) | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | ภาควิชา | วิศวกรรมคอมพิวเตอร์ | | สาขาวิชา | วิศวกรรมคอมพิวเตอร์ | | ปีการศึกษา | 2557 | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | ลายมือชื่อนิสิต |  | | | |  |  | | --- | --- | | ลายมือชื่อ อ.ที่ปรึกษาหลัก |  | | | |

พิชิต พร้อมสุทธิพงศ์ : การสร้างรูปแบบพฤติกรรมสำหรับศัตรูในเกมแอ็คชั่นสองมิติแบบเน้นตัวละครโดยอัตโนมัติด้วยเทคนิคเหมืองข้อมูล (Automatic enemy behavior pattern generation for 2D avatar-based action game using data mining technique) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร.วิษณุ โคตรจรัส, 4 หน้า.

No data found

บทคัดย่อภาษาไทย

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | |  |  | | --- | --- | | Department: | Computer Engineering | | Field of Study: | Computer Engineering | | Academic Year: | 2014 | | |  |  |  | | --- | --- | --- | | |  |  | | --- | --- | | Student's Signature |  | | | |  |  | | --- | --- | | Advisor's Signature |  | | | |

# # 5670308821 : MAJOR COMPUTER ENGINEERING

KEYWORDS:

PICHIT PROMSUTIPONG: Automatic enemy behavior pattern generation for 2D avatar-based action game using data mining technique. ADVISOR: ASST. PROF. VISHNU KOTRAJARAS, 4 pp.

No data found

บทคัดย่อภาษาอังกฤษ

**กิตติกรรมประกาศ**

กิตติกรรมประกาศ

**สารบัญ**

หน้า

บทคัดย่อภาษาไทย ง

บทคัดย่อภาษาอังกฤษ จ

กิตติกรรมประกาศ ฉ

สารบัญ ช

รายการอ้างอิง 2

ประวัติผู้เขียนวิทยานิพนธ์ 4

1. ที่มาและความสำคัญของปัญหา

อุตสาหกรรมเกมเป็นหนึ่งในอุตสาหกรรมสื่อที่มีอัตราการเติบโตสูงเป็นอันดับต้นของสื่อทั้งหมด รายงานการคาดการณ์เกี่ยวกับอุตสาหกรรมสื่อของ PwC [[1](#_ENREF_1)] แสดงให้เห็นว่าอัตราการเติบโตของยอดขายของธุรกิจเกมสูงเป็นอันดับสาม ข้อมูลดังกล่าวนี้สะท้อนให้เห็นถึงความสำคัญในการผลักดันให้เกิดวิธีการหรือแนวคิดใหม่ที่ช่วยให้อุตสาหกรรมเกมสามารถพัฒนาต่อไปได้

ในการผลิตเกมนั้นจำเป็นต้องอาศัยเงินทุนและทรัพยากรบุคลากรจำนวนมากเพื่อที่จะผลิตเกมที่มีคุณภาพที่เป็นที่ยอมรับได้ในตลาดภายใต้กรอบเวลาที่กำหนด ประเด็นดังกล่าวนี้อาจไม่เป็นปัญหาสำหรับบริษัทเกมที่มีขนาดใหญ่ แต่ในยุคปัจจุบันนี้ ผู้พัฒนามีช่องทางการจัดจำหน่ายเกมที่เข้าถึงได้ง่ายขึ้นกว่าเดิม ทำให้เกิดผู้พัฒนารายย่อยขึ้นเป็นจำนวนมากและเกิดการแข่งขันกันภายในตลาดเกมอย่างรุนแรง เนื่องจากผู้พัฒนารายย่อยเหล่านี้มีทรัพยากรบุคคลและเงินทุนจำกัด เมื่อรวมเข้ากับสภาพการแข่งขันของตลาดเกมในยุคปัจจุบันที่ทำให้ผู้พัฒนาจำเป็นต้องผลักดันผลงานของตัวเองออกมาอย่างรวดเร็ว เทคโนโลยีหรือวิธีการที่จะช่วยลดทอนทรัพยากร เงินทุน และเวลาในการผลิตเกมจึงเป็นตัวช่วยที่สำคัญต่อกลุ่มผู้พัฒนารายย่อยเหล่านี้

การที่เกมหนึ่ง ๆ จะออกสู่ตลาดได้ จำเป็นต้องผ่านกระบวนการมากมายตั้งแต่ขั้นตอนออกแบบและพัฒนาโปรแกรม ไปจนถึงการทำตลาดและเผยแพร่ ขั้นตอนการออกแบบเกมและสร้างเนื้อหา (content) นับเป็นหนึ่งในขั้นตอนที่สำคัญที่สุดในการผลิตเกม นักออกแบบเกมจำเป็นต้องใช้ความคิดสร้างสรรค์อย่างมากในการสร้างเนื้อหาของเกม ซึ่งในบางครั้ง กว่าจินตนาการของนักออกแบบจะเข้าถึงคำตอบของการออกแบบที่เหมาะสมต้องใช้เวลานาน แนวคิดของการสร้างเนื้อหาแบบอัตโนมัติ (Procedural content generation -- PCG) จึงเกิดขึ้นเพื่อช่วยลดปัญหาดังกล่าวนี้ [[2](#_ENREF_2)]

เกมแอ็คชั่นเป็นเกมที่ผู้เล่นจะต้องอาศัยปฏิกิริยาตอบสนองและความแม่นยำในการโต้ตอบกับสภาพแวดล้อมของตัวเกมภายใต้กฎที่เกมกำหนดไว้ และเมื่อเจาะจงลงไปเป็นเกมแอ็คชั่นแบบเน้นตัวละคร ผู้เล่นจะต้องตอบสนองกับสภาพแวดล้อมของตัวเกมผ่านการควบคุมตัวละครอวตาร และหนึ่งในสภาพแวดล้อมที่สำคัญของตัวเกมประเภทนี้คือศัตรู

ลักษณะของศัตรูในเกมประเภทนี้แบ่งเป็นศัตรูแบบฉลาดและศัตรูที่มีรูปแบบการเคลื่อนไหวตายตัว ศัตรูแบบฉลาดจะถูกควบคุมโดยปัญญาประดิษฐ์ที่มีความซับซ้อน มีความสามารถในการตัดสินใจเลือกใช้พฤติกรรมตามแต่สถานการณ์ปัจจุบัน เกมที่ใช้ศัตรูประเภทนี้จะเน้นไปที่การต่อสู้ระหว่างตัวละครจำนวนน้อย ในทางตรงกันข้าม ศัตรูที่มีการเคลื่อนไหวตายตัวจะถูกใช้ในเกมที่เน้นไปที่การต่อสู้ระหว่างอวตารผู้เล่นและศัตรูจำนวนมาก ผู้เล่นจะต้องทำความเข้าใจรูปแบบพฤติกรรมของศัตรูที่นักออกแบบกำหนดเอาไว้และเอาชนะให้ได้ ศัตรูจำพวกนี้จะต้องถูกออกแบบมาอย่างดีเพื่อให้พฤติกรรมมีความท้าทายพอที่ผู้เล่นจะยอมรับได้ กระบวนการออกแบบนี้นอกจากจะต้องใช้เวลาแล้ว อาจจะมีรูปแบบพฤติกรรมบางอย่างที่นักออกแบบอาจจะคาดไม่ถึงได้ การนำเอา PCG เข้ามาช่วยในการออกแบบพฤติกรรมของศัตรูจึงช่วยลดภาระของนักออกแบบลงได้

งานวิทยานิพนธ์นี้จึงมีวัตถุประสงค์เพื่อค้นหาวิธีการสำหรับสร้างพฤติกรรมของศัตรูที่มีรูปแบบพฤติกรรมตายตัวแบบอัตโนมัติสำหรับเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร โดยมีเงื่อนไขว่า พฤติกรรมของศัตรูที่ถูกสร้างขึ้นมาจะต้องทำให้ศัตรูนั้นสามารถนำไปใช้ในเกมได้และเป็นที่ยอมรับของผู้เล่น รวมไปถึงสามารถทำความเข้าใจได้โดยมนุษย์เพื่อที่นักออกแบบจะสามารถนำผลลัพธ์ไปปรับปรุงต่อได้ในกรณีที่ต้องการ

งานวิจัยที่เกี่ยวข้องกับพฤติกรรมศัตรูเท่าที่มีอยู่ในปัจจุบันจะเน้นไปที่ศัตรูแบบฉลาด ผู้วิจัยยังไม่พบวิจัยที่เกี่ยวข้องกับศัตรูที่มีรูปแบบพฤติกรรมตายตัว งานวิจัยนี้จึงต้องเริ่มต้นจากการกำหนดนิยามของศัตรูและพฤติกรรมในบริบทของเกมแอ็คชั่นสองมิติแบบเน้นตัวละครโดยสร้างแบบจำลองภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละครในรูปของภาษาอธิบายภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร แล้วจึงพัฒนาขั้นตอนวิธีสร้างรูปแบบพฤติกรรมจากแบบจำลองที่ทำให้รูปแบบพฤติกรรมดังกล่าวยอมรับได้โดยผู้เล่น ผลลัพธ์ที่ได้จากขั้นตอนวิธีจะอยู่ในรูปของภาษาอธิบายภาคีฯเพื่อให้มนุษย์สามารถทำความเข้าใจได้ อนึ่งการสร้างพฤติกรรมของศัตรูสำหรับงานนี้จะไม่นำลักษณะของพื้นที่เข้ามาพิจารณาร่วม

1. ทฤษฎีที่เกี่ยวข้อง
   1. เกมแอ็คชั่นแบบเน้นตัวละคร (Avatar-based action game)

Ernest Adams [[3](#_ENREF_3)] ให้นิยามของเกมประเภทแอ็คชั่นไว้ว่าเป็นเกมที่ผู้เล่นจะต้องใช้ความสามารถทางกายภาพในการเล่น ผู้เล่นต้องมีความรวดเร็วในการตัดสินใจโต้ตอบกับตัวเกมรวมไปถึงต้องมีความแม่นยำในการโต้ตอบ นอกจากนี้ยังต้องอาศัยการประสานงานระหว่างตาและมือ (Hand-eye coordination) ในการเล่น เช่น ผู้เล่นต้องสังเกตสภาพแวดล้อมในเกมตลอดเวลาพร้อมโต้ตอบกับสภาพแวดล้อมโดยการป้อนข้อมูลใส่ตัวเกมผ่านส่วนต่อประสานต่าง ๆ เช่น แผงแป้นอักขระ ก้านควบคุม เป็นต้น หากผู้เล่นไม่สามารถโต้ตอบกับสภาพแวดล้อมของเกมได้ถูกต้อง ทันเวลาหรือแม่นยำเพียงพอ อาจจะเกิดการลงโทษ (penalty) บางอย่าง

เกมแอ็คชั่นแบ่งเป็นประเภทย่อยได้หลายประเภท การแบ่งประเภทย่อยยังไม่มีกฎเกณฑ์ที่ตายตัว อาจขึ้นอยู่กับมุมมองของผู้เล่นหรือกลุ่มสังคมผู้เล่นได้ อีกทั้งหลาย ๆ เกมในท้องตลาดก็มีการควบรวมเอารูปแบบการเล่นและกลไกของเกม (Game mechanic) หลายประเภทรวมไว้ในเกมเดียวกันได้ ทำให้เกิดความแตกต่างในการแบ่งประเภทของเกมออกไปอีก สำหรับในแวดวงการศึกษาด้านเกม ตำราที่เกี่ยวข้องกับการออกแบบเกมจะมีการกล่าวถึงประเภทย่อยของเกมแอ็คชั่นเช่นเดียวกัน ได้แก่

* เกมยิงปืน (Shooter) ซึ่งบางตำราอาจจะมีการแยกเกมยิงปืนประเภทมุมมองบุคคลที่หนึ่ง (First person shooter -- FPS) หรือเกมยานยิง (Shoot 'em up) ออกมาเป็นประเภทย่อยต่างหากดังเช่นในหนังสือของ Scott Roogers [[4](#_ENREF_4)]
* เกมแพลทฟอร์ม (Platformer)
* เกมต่อสู้ (Fighting)
* เกมแอ็คชั่นผจญภัย (Action-Adventure)
* เกมดนตรี (Rhythm)
* เกมปริศนาแบบเร็ว (Fast Puzzle)

เกมแอ็คชั่นแบบเน้นตัวละครไม่ใช่ประเภทย่อยใด ๆ ของเกมแอ็คชั่น แต่เป็นลักษณะของเกมที่ผู้เล่นจะต้องควบคุมตัวละครอวตาร (Avatar) และตอบโต้กับสภาพแวดล้อมของเกมผ่านตัวละครอวตารดังกล่าวภายใต้กลไกที่กำหนดเพื่อนำพาตัวละครอวตารไปสู่จุดมุ่งหมาย (Goal) ของเกมให้ได้ หลายประเภทย่อยของเกมแอ็คชั่นมีลักษณะของเกมแอ็คชั่นแบบเน้นตัวละคร เช่น เกมยิงปืน, เกมแพลทฟอร์ม, เกมต่อสู้ และเกมแอ็คชั่นผจญภัย

Mark Davies [[5](#_ENREF_5)] แจกแจงรายละเอียดของกลไกที่สำคัญสำหรับเกมลักษณะนี้ไว้ ซึ่งสามารถสรุปได้เป็น 3 กลไกใหญ่ ได้แก่

* กลไกการเคลื่อนไหวของตัวละครอวตาร: ระบุว่าตัวละครอวตารสามารถทำสิ่งใดได้บ้างภายในโลกของเกม
* กลไกการควบคุมกล้องติดตามตัวละคร: ระบุวิธีการแสดงภาพของโลกของเกมต่อผู้เล่น
* กลไกการต่อสู้: ระบุวิธีการต่อสู้ระหว่างตัวละครภายในเกม

กลไกการต่อสู้ประกอบด้วยกลไกค่าคุณสมบัติของตัวละคร, วิธีการต่อสู้ระหว่างตัวละคร และความเกี่ยวพันระหว่างการต่อสู้และค่าพลัง ตัวละครที่มีบทบาทสำคัญในกลไกคือศัตรู ซึ่งมีหน้าที่ในการขัดขวางไม่ให้ตัวละครอวตารสามารถเดินหน้าไปต่อได้ ผู้เล่นจะต้องหลบหลีกหรือปราบศัตรูเพื่อเดินหน้าต่อไป ศัตรูจึงนับเป็นความท้าทายหลักของเกมแอ็คชั่นแบบเน้นตัวละครได้

พฤติกรรมของศัตรูในเกมแอ็คชั่นแบบเน้นตัวละครจะแตกต่างกันไปตามแต่ประเภทย่อยของเกมนั้น ในเกมต่อสู้หรือเกมยิงปืนที่เน้นการต่อสู้ระหว่างตัวละครจำนวนน้อย ตัวละครหนึ่ง ๆ จะมีความสามารถและการกระทำหลายรูปแบบ ความท้าทายของการต่อสู้กับศัตรูในบริบทนี้จะเกิดจากความสามารถของตัวศัตรูซึ่งถูกควบคุมโดยปัญญาประดิษฐ์ที่ถูกปรับแต่งให้มีความฉลาดเพียงพอที่จะรับมือผู้เล่นได้ในระดับที่กำหนด สามารถเลือกการกระทำได้ถูกต้องและเหมาะสมตามสถานการณ์ พฤติกรรมของศัตรูในบริบทเกมนี้จึงไม่มีรูปแบบที่แน่นอนหรือมีรูปแบบที่แน่นอนแต่มีความซับซ้อนมากจนไม่สามารถคาดเดาได้โดยง่าย

ในอีกประเภทย่อยหนึ่ง เช่น เกมแพลทฟอร์ม, เกมแอ็คชั่นผจญภัย หรือเกมยานยิง ศัตรูที่ผู้เล่นพบจะมีรูปแบบพฤติกรรมที่ตายตัว ความท้าทายของศัตรูในเกมประเภทนี้มาจากการที่ผู้เล่นต้องต่อสู้กับศัตรูที่มีพลังเหนือกว่ามากหรือต่อสู้กับศัตรูหลายตัวพร้อมกัน ทำให้ผู้เล่นต้องเรียนรู้รูปแบบและความสามารถที่ตายตัวของศัตรูเหล่านั้นทั้งหมดให้ได้อย่างรวดเร็วเพื่อที่จะสามารถตัดสินใจว่าในสถานการณ์หนึ่ง ๆ จะทำอย่างไร ควรจะต่อสู้หรือหลีกเลี่ยงการต่อสู้ดังกล่าว หากจะต่อสู้จะต้องใช้วิธีใดจึงจะปราบศัตรูลงได้ หากจะหลีกเลี่ยง ทำอย่างไรจึงจะเสียพลังชีวิตน้อยที่สุด

* 1. การทำเหมืองข้อมูล (Data mining)

การทำเหมืองข้อมูล หมายถึง “การสกัดรูปแบบข้อมูล (Data pattern) ที่สนใจออกมาจากข้อมูลปริมาณมาก” [[6](#_ENREF_6)] โดยทั่วไปจะมีขั้นตอนที่สำคัญได้แก่ การประมวลข้อมูลก่อน (Data preprocessing) การทำเหมืองข้อมูล (Data mining) การวัดผลรูปแบบผลลัพธ์ (Pattern evaluation) และการนำเสนอความรู้ (Knowledge presentation)

การประมวลข้อมูลก่อนเป็นการนำเอาข้อมูลที่อยู่ภายในฐานข้อมูลมาประมวลผลเพื่อให้พร้อมสำหรับการทำเหมืองข้อมูล ซึ่งประกอบด้วยการทำความสะอาดข้อมูล (Data cleaning) เพื่อลดทอนสัญญาณรบกวน (Noise), การเลือกข้อมูลที่เกี่ยวข้อง และการแปลงข้อมูล (Data transformation) เพื่อให้เหมาะสมกับการทำเหมืองข้อมูล

ขั้นตอนการทำเหมืองข้อมูลเป็นการนำเข้าข้อมูลที่ผ่านการประมวลข้อมูลก่อนแล้วมาสกัดหารูปแบบข้อมูลที่สนใจ ขั้นตอนวิธีที่ใช้สำหรับการสกัดข้อมูลมีมากมายหลายแบบ แต่ละขั้นตอนวิธีจะถูกออกแบบเพื่อสกัดข้อมูลจากฐานข้อมูลที่มีแบบ (Format) เฉพาะต่าง ๆ เช่น ฐานข้อมูลลำดับ (Sequence database), ฐานข้อมูลกราฟ (Graph database), ฐานข้อมูลข้อความ (Text database) เป็นต้น รูปแบบที่ได้จากการทำเหมืองข้อมูลจะถูกวัดความน่าสนใจ (Interestingness)

งานวิทยานิพนธ์นี้ทำเหมืองข้อมูลบนชุดข้อมูล (Dataset) รูปแบบพฤติกรรมของศัตรู โดยมีการแปลงข้อมูลให้อยู่ในรูปของลำดับและกราฟเพื่อสกัดข้อมูลที่น่าสนใจจากหลายมุมมอง ในหัวข้อนี้จึงนำเสนอนิยามของการทำเหมืองข้อมูลสำหรับข้อมูลลำดับและกราฟ

* + 1. การทำเหมืองข้อมูลสำหรับข้อมูลลำดับ (Sequential pattern mining)

การทำเหมืองข้อมูลสำหรับข้อมูลลำดับ คือ การสกัดลำดับย่อยที่พบบ่อย (Frequent subsequence) จากฐานข้อมูลลำดับ โดยมีนิยามดั้งเดิมที่ให้ไว้โดย Agrawal และ Srikant [[7](#_ENREF_7)] ดังต่อไปนี้

“กำหนดให้ฐานข้อมูลลำดับประกอบด้วยข้อมูลลำดับ โดยแต่ละลำดับประกอบด้วยรายการของทรานแซ็คชั่น (Transaction) เรียงตามเวลา และแต่ละทรานแซ็คชั่นคือไอเท็มเซ็ต (Itemset) การทำเหมืองข้อมูลสำหรับข้อมูลลำดับ คือ การค้นหาลำดับย่อยทั้งหมดที่มีค่าซัพพอร์ท (Support) เกินค่าซัพพอร์ทขั้นต่ำ (Minimum support – min\_sup) โดยค่าซัพพอร์ทของลำดับย่อย คือ จำนวนลำดับในฐานข้อมูลลำดับที่บรรจุ (Contain) ลำดับย่อยที่กำหนด”

และมีคำศัพท์และนิยามที่สำคัญดังนี้

* เซ็ตของไอเท็ม (Items) ; คือสัญพจน์ (Literal), 
* ไอเท็มเซ็ต (Itemset) เป็นเซ็ตของไอเท็มที่ไม่เป็นเซ็ตว่าง เขียนด้วย ;  ในกรณีที่ไอเท็มเซ็ตมีเพียงไอเท็มเดียว สามารถละเครื่องหมายวงเล็บได้ เช่น เขียนด้วย แทน
* ลำดับ (Sequence) เขียนด้วย ; คือไอเท็มเซ็ต และ มาก่อน เมื่อ 
* ลำดับ  เรียกว่า ลำดับความยาว  (-length sequence)
* เป็นลำดับย่อย (Subsequence) ของ  เมื่อมีจำนวนเต็ม;ที่ทำให้ โดยสามารถเขียนด้วยสัญลักษณ์ 

ข้อมูลขาเข้าในการทำเหมืองข้อมูลสำหรับข้อมูลลำดับ คือ ฐานข้อมูลลำดับ  ซึ่งเป็นเซ็ตของ 2-tuple  เมื่อ คือ ID ของข้อมูลที่ไม่ซ้ำกันสำหรับทุกข้อมูลในฐานข้อมูลและคือลำดับ

ข้อมูลขาออกในการทำเหมืองข้อมูลบนฐานข้อมูล  ที่กำหนดค่าจำนวนเต็มบวก คือ รายการของลำดับย่อยที่พบบ่อย โดยลำดับย่อย  มีนิยามว่า เมื่อ

ขั้นตอนวิธีที่ใช้ในการทำเหมืองข้อมูลสำหรับข้อมูลลำดับมีหลากหลายวิธีการ สำหรับงานวิทยานิพนธ์นี้เลือกขั้นตอนวิธี PrefixSpan [[8](#_ENREF_8)] มาปรับปรุงให้เข้ากับลักษณะของงาน โดยรายละเอียดของขั้นตอนวิธี PrefixSpan จะกล่าวถึงในหัวข้อที่ 3.5 และวิธีการปรับปรุงจะกล่าวถึงในหัวข้อที่ 4.2.3

* + 1. การทำเหมืองข้อมูลสำหรับข้อมูลกราฟ (Graph pattern mining)

การทำเหมืองข้อมูลสำหรับข้อมูลกราฟ คือ การสกัดกราฟย่อยที่พบบ่อย (Frequent subgraph) จากฐานข้อมูลกราฟ โดยมีคำศัพท์และนิยามที่สำคัญดังนี้

* กราฟ คือเซ็ตของจุดยอดและเส้นเชื่อมที่เชื่อมระหว่างคู่จุดยอด แสดงได้ด้วย 2-tuple  โดยแต่ละสมาชิกมีนิยามดังนี้
  + เซ็ตจุดยอด (Vertex) ; คือสัญพจน์, 
  + เซ็ตเส้นเชื่อม (Edge) สมาชิกของเซ็ตถือเป็นคู่แบบไม่มีลำดับ
  + ในกรณีที่เป็นกราฟระบุทิศทาง (Directed graph) สมาชิกของเซ็ตเส้นเชื่อมจะเป็นคู่แบบมีลำดับ จุดยอดตัวแรกของคู่จะเป็นหัว และจุดยอดตัวหลังเป็นหาง เส้นเชื่อมนั้นเชื่อมจากหัวไปยังหาง
* ในกรณีที่เป็นกราฟระบุชื่อ (Labeled graph) จะสามารถแสดงด้วย 4-tuple  โดยสมาชิกที่เพิ่มขึ้นมามีนิยามดังนี้
  + เซ็ตชื่อ (Label set) ; คือสัญพจน์, 
  + ฟังก์ชั่นชื่อ (Label mapping function)  สำหรับกำหนดชื่อให้กับจุดยอดและเส้นเชื่อม
* ฟังก์ชั่นสมสัณฐานกราฟ (Graph isomorphism) สำหรับกราฟระบุชื่อในกรณีของการทำเหมืองข้อมูลสำหรับข้อมูลกราฟ [[9](#_ENREF_9)] เป็นฟังก์ชั่นหนึ่งต่อหนึ่งทั่วถึง (Bijective function)  ซึ่ง
  + 
  + 
  + โดย แทนเซ็ตจุดยอดของกราฟ และ แทนเซ็ตเส้นเชื่อมของกราฟ 

หากมีฟังก์ชั่นดังกล่าวสำหรับกราฟระบุชื่อ และ แสดงว่ากราฟทั้งสองนั้นสมสัณฐานกัน รูปที่ 2.1 แสดงตัวอย่างกราฟที่สมสัณฐานกัน

* กราฟ บรรจุกราฟ  ก็ต่อเมื่อมีกราฟย่อย (Subgraph) ซึ่งและสมสัณฐานกัน

ข้อมูลขาเข้าในการทำเหมืองข้อมูลสำหรับข้อมูลกราฟ คือ ฐานข้อมูลกราฟ ซึ่งเป็นเซ็ตของกราฟ หรืออาจอยู่ในรูปของ 2-tuple  โดยคือของกราฟที่ไม่ซ้ำกันสำหรับทุกข้อมูลในฐานข้อมูลและคือกราฟ

ข้อมูลขาออกในการทำเหมืองข้อมูลบนฐานข้อมูลกราฟ ที่กำหนดค่าจำนวนเต็มบวก คือ รายการของกราฟย่อยที่พบบ่อย โดยกราฟย่อย ที่พบบ่อยมีนิยามว่า เมื่อ และ มีค่าเป็น  เมื่อ บรรจุ  และเป็น เมื่อ ไม่บรรจุ 

ปัจจุบันมีงานวิจัยจำนวนหลากหลายงานที่นำเสนอขั้นตอนวิธีสำหรับการทำเหมืองข้อมูลบนข้อมูลกราฟ ในงานวิทยานิพนธ์นี้เลือกใช้ gSpan [[9](#_ENREF_9)] สำหรับการทำเหมืองข้อมูล เนื่องจาก gSpan รองรับกราฟแบบมีทิศทางและกราฟระบุชื่อซึ่งเป็นแบบข้อมูลที่แสดงถึงรูปแบบพฤติกรรมบางประเภทที่งานวิทยานิพนธ์นี้ให้ความสนใจ gSpan ที่งานวิทยานิพนธ์นี้ใช้ถูกรวมอยู่ในชุดเครื่องมือทำเหมืองข้อมูลสำหรับข้อมูลกราฟ ParSeMiS [[10](#_ENREF_10)]

|  |  |
| --- | --- |
|  | |
|  |  |
| รูปที่ 2.1  และ  สมสัณฐานกันด้วยฟังก์ชั่น | |

* 1. ปัญหาการตอบสนองเงื่อนไขบังคับ (Constraint satisfaction problem -- CSP)

CSP [[11](#_ENREF_11)] เป็นปัญหาที่เกี่ยวข้องกับการกำหนดค่าให้กับตัวแปรที่สนใจภายใต้เงื่อนไขบังคับที่ผูกมัดระหว่างตัวแปรตั้งแต่ 1 ตัวขึ้นไป ซึ่งมีนิยามที่สำคัญต่อไปนี้

*  คือ เซ็ตของตัวแปรที่สนใจ
*  คือ เซ็ตของโดเมนของตัวแปรที่สนใจ ซึ่งสมาชิก  เป็นเซ็ตที่ระบุค่าที่สามารถกำหนดให้ตัวแปร  ได้ โดยเซ็ตดังกล่าวมีลักษณะใดลักษณะหนึ่งได้ต่อไปนี้
  + โดเมนจำกัดไม่ต่อเนื่อง (Discrete, finite domain) เช่น 
  + โดเมนไม่จำกัดไม่ต่อเนื่อง (Discrete, infinite domain) เช่น 
  + โดเมนต่อเนื่อง (Continuous domain) เช่น 
*  คือ เซ็ตของเงื่อนไขบังคับ ซึ่งสมาชิก  อยู่ในรูป 2-tuple โดย
  +  เป็น n-tuple ของตัวแปรที่เกี่ยวข้องกับเงื่อนไขจำกัด 
  +  เป็นความสัมพันธ์ที่ระบุว่าตัวแปรภายใน  เป็นค่าใดได้บ้าง โดยอาจอยู่ในรูปของ l-tuple ที่แสดงทุกค่าที่เป็นไปได้ตามเงื่อนไขบังคับ หรือความสัมพันธ์ที่สามารถ
    - ใช้ตรวจสอบ l-tuple ว่าตรงตามเงื่อนไขบังคับหรือไม่
    - แจงค่าตัวแปรที่เป็นไปได้

คำตอบของ CSP ได้จากการเลือกค่าจากเซ็ต  ให้กับตัวแปร  ทุกตัวที่เป็นสมาชิกของ โดยในหนึ่งปัญหาสามารถมีได้หลายคำตอบหรือไม่มีคำตอบก็ได้

มีหลากหลายวิธีการที่ช่วยให้การค้นหาคำตอบของ CSP ออกมาจากปริภูมิสถานะ (State space) ได้เร็วขึ้นนอกเหนือไปจากการค้นหาโดยทั่วไป ในที่นี้ขอยกตัวอย่างวิธีการส่วนหนึ่งจากหนังสือของ Stuart Russell และ Peter Norvig [[11](#_ENREF_11)] เช่น การแผ่เงื่อนไขบังคับ (Constraint propagation) เพื่อลดขนาดของ  หรือการทำแบ็คแทรคกิ้ง (Backtracking) โดยใช้ฮิวริสติก (Heuristic) ในการเลือกลำดับของการค้นหาเพื่อเพิ่มโอกาสในการพบคำตอบที่ไม่ตรงตามเงื่อนไขบังคับรวดเร็วขึ้น ส่งผลให้ต้นไม้ค้นหาถูกตัดออก (Prune) มากขึ้น เป็นต้น ทว่า CSP โดยทั่วไปแล้วเป็นปัญหาที่จัดอยู่ในกลุ่ม NP-Complete [[12](#_ENREF_12)] ทำให้ไม่มีขั้นตอนวิธีที่ดีที่สุดที่ใช้ในการแก้ปัญหานี้ งานวิทยานิพนธ์นี้จึงพิจารณาเลือกตัวแก้ไขปัญหาบังคับ (Constraint solver) โดยพิจารณาจาก API ว่ารองรับการสร้างแบบจำลองปัญหาในบริบทของงานวิทยานิพนธ์นี้ได้หรือไม่ ซึ่งเครื่องมือที่งานวิทยานิพนธ์นี้เลือกคือ JaCoP [[13](#_ENREF_13)]

1. งานวิจัยที่เกี่ยวข้อง
   1. งานวิจัยที่เกี่ยวกับการสร้างสิ่งมีชีวิตประดิษฐ์ (Artificial creature)

สมบัติสำคัญประการหนึ่งของการเป็นสิ่งมีชีวิตคือ ความสามารถในการตอบสนองต่อสิ่งเร้า [[14](#_ENREF_14)] จากมุมมองของผู้วิจัย การตอบสนองของสิ่งมีชีวิตอาจทำให้เกิดเป็นพฤติกรรมที่มีความหมายขึ้นได้ ในงานวิทยานิพนธ์นี้ต้องการสร้างพฤติกรรมที่มีรูปแบบตายตัวของศัตรู แม้ว่าพฤติกรรมที่มีรูปแบบตายตัวอาจไม่ได้เกิดจากการตอบสนองต่อสิ่งเร้าในลักษณะเดียวกับที่เกิดขึ้นในสิ่งมีชีวิต แต่พฤติกรรมที่อาจเกิดขึ้นจากการสร้างสิ่งมีชีวิตประดิษฐ์อาจจะมีความเกี่ยวข้องหรือสามารถนำมาประยุกต์ใช้กับงานวิทยานิพนธ์นี้ได้ ผู้วิจัยจึงทำการสำรวจงานวิจัยทางด้านนี้เพื่อความครอบคลุมของการทบทวนวรรณกรรม

Thomas Miconi และ Alastair Channon [[15](#_ENREF_15)] นำเสนอระบบสำหรับการสร้างและวิวัฒน์สิ่งมีชีวิตประดิษฐ์ สิ่งมีชีวิตในงานวิจัยนี้เกิดจากการนำบล็อกสี่เหลี่ยมหลายรูปทรงมาเชื่อมต่อกันอย่างมีลำดับชั้นตามแบบจำลองของ Karl Sim โดยมีบล็อกหนึ่งเป็นบล็อกราก บล็อกลำดับชั้นที่สองจะถูกเชื่อมเข้ากับบล็อกรากด้วยข้อต่อแบบบานพับ (Hinge joint) บล็อกลำดับชั้นถัดไปจะเชื่อมต่อกับบล็อกลำดับชั้นก่อนหน้าในลักษณะเดียวกัน บล็อกแต่ละบล็อกสามารถเคลื่อนไหวได้โดยการหมุนบล็อกลำดับชั้นรองรอบข้อต่อที่เชื่อมกับบล็อกในลำดับชั้นที่เหนือกว่า การหมุนของแต่ละบล็อกจะถูกควบคุมโดยโครงข่ายประสาทเทียมซึ่งรับค่านำเข้าจากหน่วยรับความรู้สึกที่ฝังอยู่ภายในทุกบล็อกซึ่งมีหน้าที่วัดค่าจากสภาพแวดล้อม เช่น ระยะทางจากบล็อกไปยังสิ่งมีชีวิตประดิษฐ์ตัวอื่น เมื่อนำสิ่งมีชีวิตประดิษฐ์ตามนิยามดังกล่าวนี้ไปอยู่ในโลกเสมือนสามมิติที่มีการจำลองฟิสิกส์แบบสมจริงโดยกำหนดให้ทุกบล็อกของสิ่งมีชีวิตเป็นวัตถุแข็งเกร็ง สิ่งมีชีวิตประดิษฐ์ก็จะสามารถเคลื่อนไหวได้

สิ่งมีชีวิตประดิษฐ์ที่ได้จะถูกวิวัฒน์ด้วยตัวดำเนินการเชิงพันธุกรรมเพื่อผสมรวมบล็อกจากสิ่งมีชีวิตประดิษฐ์สองตัว หรือกลายพันธุ์ค่าพารามิเตอร์ของโครงข่ายประสาทเทียมของแต่ละบล็อก สิ่งมีชีวิตประดิษฐ์ที่ได้จะถูกวัดผลด้วยการแข่งขันครองกล่อง สิ่งมีชีวิตใดที่เข้าใกล้กล่องที่กำหนดในโลกเสมือนมากกว่าจะได้คะแนนมากกว่า

งานชิ้นถัดมาของ Thomas Miconi [[16](#_ENREF_16)] นำเสนอสภาพแวดล้อมสำหรับทดลองการวิวัฒนาการของสิ่งมีชีวิตประดิษฐ์ที่อาศัยแนวคิดของการคัดเลือกโดยธรรมชาติ ซึ่งสิ่งมีชีวิตในงานนี้จะต้องต่อสู้กันเพื่อขยายเผ่าพันธุ์ของตนเอง สิ่งมีชีวิตที่สามารถฆ่าสิ่งมีชีวิตตัวอื่นได้ จะทำการเพาะพันธุ์สิ่งมีชีวิตพันธุ์เดียวกัน อีกวิธีหนึ่งของการเพิ่มจำนวนสิ่งชีวิตพันธุ์เดียวกันคือการมีอายุยืนถึงจุดที่กำหนด

งานนี้ใช้สิ่งมีชีวิตอิงแบบจำลองของ Karl Sim ซึ่งก็คือ สิ่งมีชีวิตแบบบล็อกที่เชื่อมต่อกันด้วยข้อต่อบานพับและควบคุมด้วยโครงข่ายประสาทเทียม โลกเสมือนที่สิ่งมีชีวิตอยู่จะมีการจำลองฟิสิกส์แบบสมจริงและบล็อกของสิ่งมีชีวิตจะถูกนับเป็นวัตถุแข็งเกร็ง วิวัฒนาการจะเกิดขึ้นเมื่อจะเพาะพันธุ์สิ่งมีชีวิตใหม่ โดยจะใช้วิธีการผสมรวมบล็อกหรือกลายพันธุ์เช่นเดียวกับงานก่อนหน้าของผู้วิจัย [[15](#_ENREF_15)] จุดที่ต่างออกไปคือการให้สิ่งมีชีวิตสามารถสร้างความเสียหายให้กับสิ่งมีชีวิตตัวอื่นได้โดยการเคลื่อนบล็อกส่วนหนึ่งของตนเข้าหาบล็อกของสิ่งมีชีวิตตัวอื่น ยิ่งเคลื่อนที่เร็วก็ยิ่งสร้างความเสียหายได้มาก

จากสองงานวิจัยข้างต้น จะพบว่า พฤติกรรมของสิ่งมีชีวิตในโลกเสมือนได้จากการทำงานของโครงข่ายประสาทเทียมที่ควบคุมการเคลื่อนไหวของบล็อกร่วมกับการใช้ฟิสิกส์แบบสมจริง หากนำมาใช้งานภายในเกม ข้อดีของวิธีดังกล่าวคือศัตรูที่สร้างขึ้นด้วยวิธีแบบเดียวกันนี้จะมีรูปร่าง กล่าวคือ วิธีนี้นอกจากจะสร้างพฤติกรรมได้แล้ว ยังสามารถสร้างรูปร่างของศัตรูขึ้นมาด้วย ทว่ารูปร่างที่เกิดขึ้นอาจไม่เหมาะสมกับการใช้งานในบริบทของเกมได้เนื่องจากรูปร่างของสิ่งมีชีวิตและพฤติกรรมจะต้องสอดคล้องกันภายใต้กฎฟิสิกส์ ซึ่งศัตรูในเกมจำนวนมากมีพฤติกรรมที่ไม่สามารถกระทำด้วยรูปร่างที่ปรากฏในเกมเมื่ออยู่ภายใต้กฎฟิสิกส์ เช่น ศัตรูรูปร่างทรงกลมที่สามารถลอยและเปลี่ยนทิศทางในอากาศได้ นอกจากนี้ เนื่องจากรูปร่างของสิ่งมีชีวิตจากงานวิจัยทั้งสองนี้เกี่ยวพันกับพฤติกรรมที่เป็นไปได้ของสิ่งมีชีวิตโดยตรง ทำให้ไม่สามารถแยกพฤติกรรมและรูปร่างออกจากกันได้ ผลที่ตามมาคือแนวคิดดังกล่าวนี้ไม่น่าจะนำมาใช้กับการสร้างพฤติกรรมในลักษณะเดียวกับงานวิทยานิพนธ์นี้ได้

Nedjma Djezzar และคณะ [[17](#_ENREF_17)] สร้างสิ่งมีชีวิตประดิษฐ์ด้วยการใช้ L-System ซึ่งเป็นไวยากรณ์ (Grammar) รูปแบบหนึ่งที่ใช้ในการอธิบายการเติบโตของเซลล์ร่วมกับแนวคิดเคมีของเซลล์ สิ่งมีชีวิตประดิษฐ์ในงานวิจัยนี้เป็นสิ่งมีชีวิตระดับเซลล์ที่อาศัยอยู่ในโลกเสมือนสองมิติ ภายในโลกแห่งนี้จะประกอบด้วยเซลล์และสารเคมีในธรรมชาติ พฤติกรรมของเซลล์จะถูกนิยามไว้ในระดับต่ำซึ่งเกี่ยวข้องกับสารเคมีของเซลล์และการกระทำของเซลล์ในโลกจริง เช่น การย่อยสลายตัวเองและการแบ่งตัว

วิธีการในงานวิจัยนี้น่าจะเหมาะสมกับการศึกษาเรื่องของเซลล์และสิ่งมีชีวิตในเชิงชีววิทยามากกว่า ด้วยความที่พฤติกรรมอยู่ในระดับต่ำมาก รวมทั้งไม่มีฟิสิกส์เข้ามาเกี่ยวข้อง ทำให้กิจกรรมเชิงกล เช่น การเคลื่อนที่ การผลักกันของเซลล์ไม่สามารถเกิดขึ้นได้ในโลกเสมือนของงานวิจัยนี้ งานวิจัยในลักษณะของการสร้างสิ่งมีชีวิตด้วยแนวคิดเคมีของเซลล์จึงไม่เหมาะสมที่จะนำมาใช้ร่วมกับงานวิทยานิพนธ์นี้

* 1. งานวิจัยเกี่ยวกับการสร้างเนื้อหาอัตโนมัติ (PCG)

งานวิทยานิพนธ์นี้เกี่ยวข้องกับการสร้างรูปแบบพฤติกรรมของศัตรูซึ่งนับได้ว่าเป็นเนื้อหาของเกม ในที่นี้จึงสำรวจงานวิจัยที่เกี่ยวกับการสร้างเนื้อหาของเกมอัตโนมัติ (PCG)

Mark Hendrikx และคณะ [[18](#_ENREF_18)] แบ่งประเภทเนื้อหาของเกมออกเป็น 6 ลำดับชั้น แต่ละลำดับชั้นจะมีการแบ่งชนิดย่อยของเนื้อหาลงไป เนื้อหาที่อยู่ในลำดับชั้นด้านล่างจะสามารถนำมาประกอบรวมกันเป็นเนื้อหาในลำดับชั้นบนที่ถัดขึ้นไปได้ เนื้อหาทั้ง 6 ลำดับชั้นเรียงจากชั้นล่างสุดประกอบด้วย Game bit, Game space, Game system, Game scenario, Game design และ Derived content เนื้อหาในชั้น Game bit เป็นหน่วยย่อยที่สุดของเกมที่ประกอบด้วย เท็กซ์เจอร์ เสียง พันธุ์ไม้ สิ่งปลูกสร้าง พฤติกรรม และวัตถุธรรมชาติ (ไฟ น้ำ หิน และเมฆ) ในงานวิจัยนี้ได้ให้คำจำกัดความของ Game bit พฤติกรรมไว้ว่า "เป็นวิธีการที่วัตถุโต้ตอบกับวัตถุอื่นหรือสิ่งแวดล้อม" ซึ่งตรงกับรูปแบบพฤติกรรมของศัตรูที่งานวิทยานิพนธ์นี้ให้ความสนใจ

ในงานวิจัยนี้ได้สำรวจขั้นตอนวิธีที่ใช้ในการสร้างเนื้อหาที่เกี่ยวข้องพฤติกรรมเอาไว้สองงาน ได้แก่ การใช้ไวยากรณ์แบบ L-System ร่วมกับเงื่อนไขเพิ่มเติมในการสร้างเส้นทางการเคลื่อนที่ของดอกไม้ไฟ [[19](#_ENREF_19)] และการใช้ Compositional pattern producing networks (CPPNs) ในการสร้างเส้นทางการเคลื่อนที่ของกระสุนอัตโนมัติสำหรับเกมยานยิง [[20](#_ENREF_20)]

งานวิจัยที่เกี่ยวข้องกับการสร้างพฤติกรรมทั้งสองงานดังกล่าวเน้นไปที่เส้นทางการเคลื่อนที่ของวัตถุเป็นหลัก ถึงแม้ว่าเส้นทางการเคลื่อนที่อาจจะเรียกได้ว่าเป็นส่วนสำคัญหนึ่งของรูปแบบพฤติกรรม แต่การเคลื่อนที่ของศัตรูภายในเกมที่วางขายและเป็นที่ยอมรับของผู้เล่นไม่ได้มีเพียงเส้นทางเท่านั้น แต่ยังมีเงื่อนไขของการเปลี่ยนแปลงเส้นทางการเคลื่อนที่และการกระทำเพิ่มเติมอื่น ๆ ส่งผลให้วิธีการที่เน้นไปที่เส้นทางการเคลื่อนที่อาจนำมาใช้ร่วมเป็นส่วนหนึ่งของงานได้ แต่ไม่สามารถทำหน้าที่ในการสร้างรูปแบบพฤติกรรมที่สมบูรณ์เพียงพอตามที่งานวิทยานิพนธ์นี้ต้องการได้

ณ จุดเวลาที่จัดทำโครงร่างวิทยานิพนธ์ฉบับนี้ เนื่องจากผู้วิจัยไม่พบงานวิจัยที่เกี่ยวข้องกับการสร้างรูปแบบพฤติกรรมของศัตรูที่ตายตัวโดยตรง จึงทำการสำรวจงานวิจัยที่เกี่ยวข้องกับการวิเคราะห์องค์ประกอบของเกมและความเกี่ยวพันกับศัตรู เพื่อหาคำค้นที่จะช่วยในการค้นหางานวิจัยที่เกี่ยวข้องต่อไป

Mark J. Nelson และ Michael Mateas [[21](#_ENREF_21)] ได้แบ่งเกมออกเป็นออกเป็น 4 ชั้น ได้แก่ ชั้นกลไก (Abstract game mechanic), ชั้นนำเสนอ (Concrete game representation), ชั้นธีมของเนื้อหา (Thematic content) และชั้นจับคู่การควบคุม (Control mapping) ชั้นกลไกนั้นเป็นตัวระบุว่าสถานะของเกม (Game state) จะเปลี่ยนแปลงไปอย่างไรตลอดการเล่น โดยการเปลี่ยนแปลงของสถานะอาจจะเกิดขึ้นเองอัตโนมัติจากความเป็นไปของเกม หรือเกิดจากการโต้ตอบจากผู้เล่นก็ได้

ในขณะเดียวกัน Miguel Sicart [[22](#_ENREF_22)] อธิบายกลไกของเกมด้วยแนวคิดของการเขียนโปรแกรมเชิงวัตถุโดยมองว่ากลไกของเกม "เป็นเมท็อดที่ถูกเรียกโดยภาคีเพื่อโต้ตอบกับสถานะของเกม" ซึ่งอิงกับคำจำกัดความของเมท็อดในการเขียนโปรแกรมเชิงวัตถุว่าเป็นเหมือนกับการกระทำหรือพฤติกรรมสำหรับคลาสและเป็นกลไกที่ใช้ในการเข้าถึงสถานะต่าง ๆ ของวัตถุอื่น

สำหรับกรณีของศัตรูในเกม พฤติกรรมของศัตรูจะทำให้สถานะของศัตรูหรือสถานะของโลกของเกมเปลี่ยนแปลงไป เช่น พฤติกรรมเดินทำให้สถานะตำแหน่งของศัตรูเปลี่ยนแปลง พฤติกรรมยิงกระสุนทำให้เกิดกระสุนใหม่ขึ้นในโลกของเกม ส่งผลให้สถานะของตัวโลกเปลี่ยนไป ซึ่งเทียบได้กับชั้นกลไกในงานของ Nelson และเมื่อใช้แนวคิดของ Sicart การกระทำย่อย ๆ ของศัตรูอาจเทียบได้กับการเรียกเมธอดของคลาสศัตรู หากพิจารณาประเด็นนี้ อาจกล่าวได้ว่า รูปแบบพฤติกรรมของศัตรูนับเป็นส่วนหนึ่งของกลไกของเกมเช่นกัน

ผู้วิจัยจึงอาศัยข้อสรุปจากการทบทวนสองงานวิจัยข้างต้น เลือกใช้ “กลไกของเกม” เป็นคำค้นเพื่อสำรวจวรรณกรรมด้าน PCG ที่เกี่ยวข้องกับกลไกของเกม

* 1. งานวิจัยเกี่ยวกับการสร้างกลไกของเกมอัตโนมัติ (Automatic game mechanic generation)

ก่อนจะเข้าสู่การทบทวนวรรณกรรมที่เกี่ยวข้องกับการสร้างกลไกของเกม ในที่นี้ขอกล่าวถึงคำว่า "กฎของเกม" ซึ่งเป็นคำศัพท์สำคัญอีกคำหนึ่งที่จำเป็นต้องทำให้เกิดความเข้าใจตรงกัน เนื่องจากปัจจุบันยังไม่มีการกำหนดความแตกต่างอย่างชัดเจนระหว่างคำว่ากลไกของเกมและกฎของเกมเป็นทฤษฎี ผู้วิจัยหรือบุคลากรในวงการเกมอาจนำคำศัพท์เหล่านี้ไปใช้งานภายใต้ความหมายที่แตกต่างกัน ในงานวิทยานิพนธ์นี้เลือกใช้การแยกแยะความแตกต่างของศัพท์ทั้งสองตามบทความของ Lewis Pulsipher [[23](#_ENREF_23)] ในบทความดังกล่าวนิยามถึงกลไกของเกมไว้ว่า เป็นข้อกำหนดที่ระบุถึงสิ่งที่สามารถกระทำได้เมื่อถูกต้องตามเงื่อนไข ส่วนกฎของเกมเป็นข้อกำหนดที่ระบุถึงกลไกของเกมและอื่น ๆ นั่นคือ กลไกของเกมนับเป็นส่วนหนึ่งของกฎของเกม สำหรับงานวิจัยที่ทำการสำรวจ ในงานนี้จะเลือกงานที่มีความเกี่ยวข้องกับกลไกของเกมตามนิยามข้างต้นโดยไม่คำนึงถึงคำศัพท์ที่ผู้เขียนเลือกใช้งาน

Julian Togelius และ Jurgen Schmidhuber [[24](#_ENREF_24)] นำเสนอการออกแบบเกม Grid-based อัตโนมัติโดยการวิวัฒน์กฎของเกมจากชุดกฎเริ่มต้นภายใต้ฟังก์ชั่นความเหมาะสมที่กำหนดขึ้นจากทฤษฎีของ Schmidhuber ซึ่งมีแนวคิดว่า เกมจะสนุกหากผู้เล่นสามารถเรียนรู้ที่จะเล่นได้ แต่ต้องไม่ง่ายจนเกินไปในระดับที่ไม่ต้องพึ่งการเรียนรู้หรือสามารถเอาชนะโดยการสุ่มเท่านั้น

งานวิจัยชิ้นนี้กำหนดชุดของกฎเริ่มต้นเพื่อใช้เป็นปริภูมิของกฎ (Rule space) ของเกมทั้งหมดที่เป็นไปได้ ซึ่งในกฎจะระบุถึงขนาดพื้นที่ โครงสร้างของพื้นที่ เงื่อนไขการจบเกม เงื่อนไขการชนะ รูปแบบของภาคีที่เป็นไปได้ซึ่งประกอบด้วยสีของภาคี รูปแบบการเคลื่อนที่ และผลลัพธ์จากการชนกับภาคีอื่น จากนั้นสร้างเกมเริ่มต้นโดยเลือกกฎออกมาจากปริภูมิอย่างสุ่มแล้ววิวัฒน์ด้วยการสุ่มปรับพารามิเตอร์ของกฎหนึ่งข้อ กฎที่วิวัฒน์แล้วจะถูกประเมินความยากในการเล่นโดยใช้ปัญญาประดิษฐ์ซึ่งสร้างจากโครงข่ายประสาทเทียม

ในงานนี้มีส่วนของกลไกที่เกี่ยวข้องกับพฤติกรรมของภาคี โดยรูปแบบพฤติกรรมดังกล่าวเป็นพฤติกรรมที่มีระดับ สูง มีเพียง 5 รูปแบบที่เป็นไปได้ ได้แก่ อยู่นิ่ง, เดินระยะสั้นแล้วสุ่มทิศ, เดินระยะยาวแล้วสุ่มทิศ, เดินและหันขวาเมื่อชนกำแพง, เดินและหันซ้ายเมื่อชนกำแพง ซึ่งส่งผลให้ขาดความหลากหลายของพฤติกรรมหากนำเอารูปแบบพฤติกรรมนี้มาใช้ในการสร้างรูปแบบพฤติกรรมของศัตรู

Adam M. Smith และ Michael Mateas [[25](#_ENREF_25)] นำเสนอวิธีการสำหรับสร้างกฎของเกมแบบ Grid-based โดยใช้ Answer Set Programming (ASP) ซึ่งเป็นการเขียนโปรแกรมเชิงเงื่อนไขบังคับ (Constraint programming) แบบหนึ่ง ตัวโปรแกรม จะถูกเขียนขึ้นเพื่อใช้เป็นฐานความรู้สำหรับกฎของเกมที่เป็นไปได้ทั้งหมดเรียกว่า ปริภูมิออกแบบ (Design space) ซึ่งเทียบได้กับปริภูมิของกฎในงานของ Togelius จากนั้นส่วนสร้างกฎซึ่งงานนี้เลือกใช้เครื่องมือ SMODELS จะแจงนับ (enumerate) กฎที่ถูกต้องตามเงื่อนไขบังคับที่กำหนดไว้ในโปรแกรมออกมาเพื่อใช้งานในเกมทดสอบต่อไป

ข้อมูลในฐานความรู้ของงานนี้ประกอบด้วย ขนาดและโครงสร้างของพื้นที่ เป้าหมายของเกม รูปแบบของภาคีที่เป็นไปได้ซึ่งประกอบด้วยสี รูปแบบการเคลื่อนที่ ผลลัพธ์การชนกับทั้งภาคีอื่นและพื้นที่ และจำนวนของภาคี ซึ่งโดยรวมแล้วมีความใกล้เคียงกับปริภูมิของกฎในงานของ Togelius แต่จะแตกต่างกันในรายละเอียดของค่าที่เป็นไปได้ในกฎแต่ละประเภท สำหรับในส่วนของการเคลื่อนที่ซึ่งเทียบเท่ากับพฤติกรรมของศัตรูที่งานวิทยานิพนธ์นี้สนใจ งานนี้ใช้แบบจำลองของการเคลื่อนที่ซึ่งระบุว่าภาคีมีการเคลื่อนที่แบบใดให้เลือกบ้าง แต่ไม่ได้กล่าวถึงรายละเอียดของการเคลื่อนที่ที่เลือกใช้ได้และวิธีการตัดสินใจว่าจะเคลื่อนที่อย่างไร ผลจากการใช้งานแบบจำลองการเคลื่อนที่ซึ่งมีลักษณะเป็นการเลือกการเคลื่อนที่แบบหนึ่งออกมาจากเซ็ตจำกัด ส่งผลให้ความหลากหลายของพฤติกรรมมีน้อยเช่นเดียวกับกรณีงานของ Togelius

Michael Cook, Simon Colton และ Jeremy Gow [[26](#_ENREF_26)] นำเสนอระบบสำหรับออกแบบเกมแพลทฟอร์มอัตโนมัติโดยใช้วิธีการวิวัฒน์ร่วม (Co-operative co-evolution -- CCE) ระหว่าง 3 องค์ประกอบ ได้แก่ แผนที่ (Map) โครงสร้าง (Layout) และไอเท็มเพิ่มพลัง (Powerset) งานวิจัยนี้ทดสอบระบบโดยการสุ่มสร้างองค์ประกอบขึ้นมาองค์ประกอบละ 200 หน่วยประชากร นำมาสร้างเป็นเกมทั้งหมด 600 เกม แล้วเลือกเกมที่มีคะแนนจากการประเมินด้วยฟังก์ชั่นความเหมาะสมสูงที่สุด เป็นจำนวน 10% ของประชากรทั้งหมดเพื่อเป็นพ่อแม่ของประชากรรุ่นถัดไป การผลิตประชากรรุ่นถัดไปจะได้จากการไขว้เปลี่ยนและการกลายพันธุ์ของพ่อแม่ ซึ่งตัวดำเนินการดังกล่าวจะแตกต่างกันออกไปสำหรับแต่ละองค์ประกอบเช่นกัน สำหรับการประเมินระบบของงานนี้จะใช้ผู้เล่นในการประเมินเกมที่ได้จากการสร้างจากระบบโดยการให้คะแนนในช่วง 1-5 คะแนน

ในรายละเอียดสำหรับส่วนขององค์ประกอบโครงสร้างของงานวิจัยนี้มีการกล่าวถึงภาคีที่เป็นศัตรู โดยองค์ประกอบโครงสร้างจะอยู่ในรูปของรายการของศัตรูที่จะปรากฏในเกม ศัตรูแต่ละตัวประกอบด้วย การกระทำ, การเคลื่อนที่ และตำแหน่ง

* การกระทำจะกล่าวถึงวิธีการที่ศัตรูใช้ในการโจมตีผู้เล่น ซึ่งมีทั้งหมด 3 รูปแบบ ศัตรูตัวหนึ่ง ๆ อาจมีรูปแบบการโจมตีมากกว่า 1 รูปแบบหรืออาจไม่มีรูปแบบการโจมตีก็ได้
* การเคลื่อนที่ใช้ระบุถึงวิธีที่ศัตรูจะเคลื่อนที่ไปในโลกของเกม มีทั้งหมด 3 รูปแบบเช่นกัน
* ตำแหน่งระบุตำแหน่งเริ่มต้นที่ศัตรูอยู่เมื่อเริ่มต้นเกม ซึ่งไม่เกี่ยวกับรูปแบบพฤติกรรม

จากองค์ประกอบทั้ง 3 ของศัตรูข้างต้น ส่งผลให้งานวิจัยนี้มีรูปแบบพฤติกรรมของศัตรูที่เป็นไปได้เพียง 24 รูปแบบเท่านั้นอันเป็นผลมาจากปริภูมิที่มีขนาดเล็ก

งานวิจัยทั้ง 3 งานที่ได้กล่าวไป มีจุดร่วมเดียวกัน คือ ความหลากหลายของศัตรูต่ำเนื่องจากตัวงานเน้นไปที่ภาพรวมของการออกแบบเกมอัตโนมัติมากกว่าการเจาะจงลงไปยังรายละเอียดเรื่องใดเรื่องหนึ่งของเกม ทำให้ขาดสัจพจน์ที่เหมาะสมกับการสร้างความหลากหลายให้พฤติกรรมของภาคี การปรับปรุงหรือออกแบบสัจพจน์ใหม่เพื่อสร้างปริภูมิของพฤติกรรมโดยเฉพาะสำหรับภาคีที่เป็นศัตรูจึงเป็นวิธีที่น่าจะแก้ไขข้อจำกัดนี้ได้

ในงานวิทยานิพนธ์นี้ใช้การเลือกพฤติกรรมในระดับกลางซึ่งได้จากแบบจำลองภาคีฯมาเรียงต่อกันภายใต้กฎและความยาวที่กำหนดแทนการเลือกกฎเพียงหนึ่งข้อจากหมวดหมู่ต่าง ๆ ทำให้รูปแบบพฤติกรรมทั้งหมดที่เป็นไปได้ซึ่งเกิดจากการเลือกและเรียงสับเปลี่ยนของพฤติกรรมมีความหลากหลายกว่างานวิจัยข้างต้น

Alexander Zook และ Mark O. Rield [[27](#_ENREF_27)] เสนอการสร้างกลไกของเกมสำหรับเกมแบบผลัดกันเล่น (turn-based) เน้นตัวละคร โดยใช้ ASP ในการสร้างกลไกของเกมขึ้นจากโดเมนความรู้และตัวดำเนินการบนโดเมนความรู้ โดยกลไกที่ได้จะถูกตรวจสอบว่าถูกต้องตามเงื่อนไขการออกแบบ (Design requirement) จากนั้นใช้ Planner สำหรับตรวจสอบกลไกที่ได้ว่าเป็นไปตามเงื่อนไขจำกัดที่ผู้ใช้กำหนดหรือไม่

โดเมนความรู้ซึ่งในงานนี้เรียกว่าแบบจำลองสถานะ (State model) จะอยู่ในรูปของชนิดตัวละครที่ปรากฏอยู่ในเกมได้ ซึ่งแต่ละตัวละครจะมีรายการค่าพารามิเตอร์และช่วงค่าที่เป็นไปได้ เช่น ตัวละครชนิดผู้เล่น มีพารามิเตอร์พลังชีวิตที่มีค่าตั้งแต่ 0 ถึง 3 เป็นต้น โดเมนความรู้ต้องถูกกำหนดโดยผู้ใช้งานก่อนการสร้างกฎแบบอัตโนมัติ ส่งผลให้ผู้ใช้งานสามารถควบคุมลักษณะของกลไกของเกมผลลัพธ์ที่จะสร้างออกมาได้

หัวใจของงานนี้อยู่ที่แบบจำลองการเปลี่ยนสถานะ (Transition model) ซึ่งใช้อธิบายการเปลี่ยนแปลงของค่าพารามิเตอร์ของตัวละคร ภายใต้เงื่อนไขของเวลา ค่าพารามิเตอร์ของตัวละครอื่น ๆ และเงื่อนไขของการใช้งานกลไกอื่น การเปลี่ยนแปลงค่าพารามิเตอร์เป็นตัวดำเนินการระดับต่ำซึ่งประกอบด้วยการเพิ่มหรือลดค่าของพารามิเตอร์ สำหรับเงื่อนไขจะเป็นการเปรียบเทียบระหว่างพารามิเตอร์ที่สนใจโดยใช้ตัวเปรียบเทียบ เช่น เท่ากับ, มากกว่า เป็นต้น

ASP จะทำการสร้างกลไกโดยเลือกเงื่อนไขและการเปลี่ยนแปลงค่าพารามิเตอร์ที่ทำให้กลไกผลลัพธ์สามารถทำงานได้ถูกต้องตามเงื่อนไขการออกแบบ และ Planner จะช่วยตรวจสอบผลลัพธ์จาก ASP ให้เป็นไปตามเงื่อนไขที่ผู้ใช้ที่กำหนด ซึ่งในกรณีตัวอย่างที่ปรากฏในงานวิจัยนี้ได้ทดลองใช้ “เกมสามารถเล่นจนจบ” เป็นเงื่อนไข

ในทางทฤษฎี แนวคิดของงานวิจัยนี้สามารถครอบคลุมเกมแบบเน้นตัวละครได้โดยไม่ได้จำกัดว่าต้องเป็นประเภทใด ผู้ใช้งานเพียงกำหนดโดเมนความรู้ให้เหมาะสมกับประเภทเกมที่ต้องการ ก็น่าจะสร้างกลไกของเกมได้ รวมไปถึงพฤติกรรมที่ตายตัวของศัตรูด้วย แต่ในทางปฏิบัติงานนี้ยังมีข้อจำกัดหลายอย่างหากจะนำไปใช้กับการสร้างรูปแบบพฤติกรรมของศัตรูดังนี้:

* เนื่องจากตัวดำเนินการต่าง ๆ ในงานวิจัยนี้อยู่ในระดับต่ำมาก เพื่อที่จะสร้างพฤติกรรมที่มีความหมายอาจจะต้องใช้เงื่อนไขและการเปลี่ยนแปลงพารามิเตอร์จำนวนมากมาเรียงต่อกัน การค้นหารูปแบบพฤติกรรมของศัตรูที่เหมาะสมในปริภูมิน่าจะใช้เวลานาน เนื่องจากปริภูมิจะมีขนาดใหญ่ และคำตอบที่อยู่ในปริภูมิจะอยู่อย่างกระจัดกระจาย
* ตัวงานเน้นไปที่เกมแบบผลัดกันเล่น ซึ่งพฤติกรรมของตัวละครมักจะเกิดขึ้นและจบลงเป็นตา การอธิบายพฤติกรรมในเกมที่ไม่เป็นแบบผลัดกันเล่น เช่น เกมแอ็คชั่นทั่วไปที่ศัตรูสามารถเคลื่อนไหวอยู่ตลอดเวลา โดยใช้แบบจำลองการเปลี่ยนสถานะของงานนี้อาจจะทำได้ยาก หรือต้องใช้จำนวนของตัวดำเนินการเปลี่ยนแปลงพารามิเตอร์มาก
* ยังขาดการกระทำสำคัญที่พบในเกมแอ็คชั่นทั่วไป เช่น การเพิ่มตัวละครใหม่ลงในโลกของเกม (ในกรณีที่ศัตรูสามารถยิงกระสุนได้) หรือการลบตัวละครที่มีอยู่แล้วออกไป

Tobias Mahlmann [[28](#_ENREF_28)] นำเสนอวิธีการสร้างเกมวางแผนการรบอัตโนมัติโดยนำเสนอแบบจำลองสำหรับอธิบายเกมวางแผนการรบแบบผลัดกันเล่นและสร้างเกมวางแผนการรบโดยการสุ่มและวิวัฒน์ยูนิต (Unit) ที่อธิบายด้วยแบบจำลองดังกล่าว

แบบจำลองของ Mahlmann อยู่ในรูปของภาษาอธิบายเกมวางแผนการรบ (Strategy game description language -- SGDL) ซึ่งอธิบายเกมวางแผนการรบโดยใช้ 4 องค์ประกอบ ได้แก่

* ยูนิต: เป็นตัวละครหน่วยหนึ่งบนสนามรบ ยูนิตมีได้หลายประเภท ยูนิตแต่ละประเภทจะมีค่าคุณสมบัติและความสามารถประจำตัว ซึ่งอยู่ในรูปของการกระทำที่สามารถทำได้
* สถานะของผู้เล่น: ประกอบด้วยค่าสถานะที่สำคัญสำหรับผู้เล่นแต่ละฝ่ายในเกมวางแผนการรบ เช่น เงิน
* สถานะของแผนที่: ประกอบด้วยค่าสถานะที่สำคัญของแผนที่ภายในเกม เช่น ยูนิตที่อยู่บนสนาม อุปสรรคบนแผนที่
* เงื่อนไขการจบเกม

SGDL เป็นภาษาแบบ Strongly-typed และนำเสนอในรูปของต้นไม้ ปมภายในต้นไม้มีหลายชนิด แตกต่างกันไปตามข้อมูลที่อยู่ภายในปม ในการทบทวนวรรณกรรมนี้จะกล่าวถึงเฉพาะต้นไม้ย่อยในส่วนที่ใช้อธิบายยูนิตซึ่งเทียบเท่ากับศัตรูในงานวิทยานิพนธ์นี้

ต้นไม้ย่อยที่ใช้อธิบายยูนิตประกอบด้วยรายการค่าคุณสมบัติและรายการการกระทำ ผู้วิจัยระบุว่ารายการค่าคุณสมบัติเป็นรายการของคู่ระหว่างชื่อคุณสมบัติและค่าตัวเลขโดยไม่มีการแสดงลักษณะของต้นไม้ให้เห็นชัดเจน ส่วนรายการการกระทำประกอบด้วยต้นไม้ย่อยการกระทำ (Action sub-tree) หลายต้น แต่ละต้นไม้ย่อยจะมีรากเป็นปมชนิดการกระทำ (Action node) ซึ่งปมชนิดนี้จะมีต้นไม้ย่อยเป็นลูกได้เพียง 2 ชนิดได้แก่ ต้นไม้ย่อยเงื่อนไข (Condition sub-tree) และต้นไม้ย่อยผลลัพธ์ (Consequence sub-tree) ซึ่งเป็นการกระทำที่จะเกิดขึ้นได้เมื่อเงื่อนไขเป็นจริงเท่านั้น ในกรณีที่ปมการกระทำมีต้นไม้ย่อยผลลัพธ์มากกว่าหนึ่งต้น เส้นเชื่อมจากปมการกระทำไปยังต้นไม้ย่อยแต่ละต้นจะต้องมีป้ายกำกับลำดับการทำงานเพื่อกำหนดว่าจะให้ต้นไม้ย่อยผลลัพธ์ใดเริ่มทำงานก่อน

ต้นไม้ย่อยเงื่อนไขจะต้องมีรากเป็นปมชนิดตัวเปรียบเทียบ (Comparator node) เท่านั้น ซึ่งจะเปรียบเทียบระหว่างข้อมูลจากต้นไม้ย่อยลูกซ้ายและต้นไม้ย่อยลูกขวาของปม ภายในนั้น มีปมได้อีกหลายชนิด เช่น ปมค่าคงที่ (Constant node) ปมคุณสมบัติ (Property node) สำหรับเข้าถึงค่าคุณสมบัติของยูนิต และปมอ้างอิงวัตถุ (Object reference node) สำหรับเลือกยูนิตหรือวัตถุอื่น ๆ ในเกม ปมตัวดำเนินการ (Operator node) สำหรับคำนวณค่าทางคณิตศาสตร์หรือค่าตรรกะ เป็นต้น ปมต่าง ๆ ที่กล่าวถึงสามารถนำมาสร้างเป็นต้นไม้ที่คืนค่าข้อมูลสำหรับใช้ในการเปรียบเทียบ หากกล่าวโดยสรุป ต้นไม้ย่อยเงื่อนไขเทียบได้กับต้นไม้นิพจน์ที่คืนค่าจริงหรือเท็จ ตัวอย่างต้นไม้เงื่อนไขเป็นไปดังรูปที่ 3.1

|  |  |
| --- | --- |
|  |  |
| รูปที่ 3.1 ตัวอย่างต้นไม้เงื่อนไข | รูปที่ 3.2 ตัวอย่างต้นไม้ย่อยผลลัพธ์ที่มีรากเป็นปมชนิดการกระทำ |
|  |  |

ต้นไม้ย่อยผลลัพธ์มีรากเป็นปมชนิดตัวดำเนินการกำหนดค่าหรือปมชนิดการกระทำ ในกรณีที่เป็นปมชนิดตัวดำเนินการกำหนดค่า ปมดังกล่าวจะมีต้นไม้ย่อยลูกได้เพียง 2 ต้นเท่านั้นดังรูปที่ 3.2 โดยตัวดำเนินการนี้จะทำการประเมินค่าของต้นไม้ย่อยขวาและกำหนดค่าให้กับคุณสมบัติในต้นไม้ย่อยซ้าย

ในส่วนของการสร้างยูนิตแบบอัตโนมัติจะแบ่งออกเป็น 2 ตอน ได้แก่ การกำหนดค่าคุณสมบัติ และการสร้างพฤติกรรม การกำหนดค่าคุณสมบัติเป็นการสุ่มกำหนดค่าให้กับคุณสมบัติต่าง ๆ ของยูนิตแล้วใช้ขั้นตอนวิธีเชิงพันธุกรรมเพื่อวิวัฒน์ให้ค่าเหล่านั้นมีความเหมาะสม ความเหมาะสมวัดด้วยการเล่นเกมอัตโนมัติโดยใช้ปัญญาประดิษฐ์แบบ Monte-Carlo ในการควบคุมยูนิตที่ถูกสร้างขึ้นสำหรับผู้เล่นทุกฝ่าย การสร้างพฤติกรรมจะได้จากการวิวัฒน์การกระทำของยูนิตที่มีอยู่แล้วด้วยขั้นตอนวิธีเชิงพันธุกรรม ตัวดำเนินการกลายพันธุ์สำหรับการกระทำจะสุ่มลบปมและ/หรือต่อเติมต้นไม้ย่อยที่เป็นส่วนหนึ่งของต้นไม้ย่อยการกระทำ ตัวดำเนินการไขว้เปลี่ยนจะสุ่มเลือกปมจากสองต้นไม้ย่อยการกระทำแล้วสับเปลี่ยนต้นไม้ย่อยโดยมีปมที่เลือกเป็นราก

SGDL อธิบายพฤติกรรมด้วยระดับที่ต่ำกว่างานของ Togelius [[24](#_ENREF_24)], Smith [[25](#_ENREF_25)] และ Cook [[26](#_ENREF_26)] ซึ่งมีข้อจำกัดในส่วนของความหลากหลายของพฤติกรรมที่เป็นไปได้ ปมตัวดำเนินการอยู่ในระดับต่ำเทียบเท่ากับงานของ Zook [[27](#_ENREF_27)] แต่เนื่องจากการกำหนดประเภทของเกมไว้ ทำให้ค่าคุณสมบัติซึ่งเทียบได้กับแบบจำลองสถานะในงานของ Zook มีชุดกฎที่รองรับโดยเอ็นจิ้นแล้ว เช่น คุณสมบัติตำแหน่ง x และ y สามารถใช้อธิบายตำแหน่งในโลกโดยไม่ต้องมีการสร้างกฎขึ้นในภายหลัง การมีชุดกฎที่รองรับล่วงหน้าจึงส่งผลให้ขนาดของปริภูมิลดลงอย่างมาก

งานวิทยานิพนธ์นี้ใช้แนวคิดเดียวกับงานของ Mahlmann ในการสร้างรูปแบบพฤติกรรมของศัตรู นั่นคือ สร้างแบบจำลองภาคีฯแล้วใช้แบบจำลองเป็นฐานความรู้ในการสร้างรูปแบบพฤติกรรม ทว่าเนื่องจากแบบจำลองในงานของ Mahlmann ออกแบบไว้สำหรับเกมวางแผนการรบแบบผลัดกันเล่น ซึ่งแตกต่างจากงานวิทยานิพนธ์นี้ที่เป็นเกมแอ็คชั่นและลักษณะการเล่นแบบทันกาล (Real-time) ศัตรูในเกมแสดงรูปแบบพฤติกรรมตลอดเวลาโดยไม่มีการรอรับข้อมูลจากผู้เล่น งานวิทยานิพนธ์นี้จึงนำเสนอแบบจำลองภาคีฯซึ่งได้จากการวิเคราะห์เกมแอ็คชั่นที่มีอยู่

นอกจากนี้ ยูนิตที่สร้างขึ้นในงานของ Mahlmann มีจุดประสงค์เพื่อให้เกมมีความสมดุล ในขณะที่งานวิทยานิพนธ์นี้ต้องการสร้างรูปแบบพฤติกรรมของศัตรูที่ยอมรับได้โดยผู้เล่นซึ่งเป็นข้อมูลที่ขึ้นกับความชอบของบุคคล การหาฟังก์ชั่นความเหมาะสมสำหรับวัดค่าความพึงพอใจที่มีต่อรูปแบบของศัตรูโดยไม่มีข้อมูลสนับสนุนแรกเริ่ม จำเป็นจะต้องทำการทดลองกับผู้เล่นเป็นจำนวนมากคนละหลายครั้งเพื่อให้ได้ฟังก์ชั่นที่แม่นยำเพียงพอในการประเมินซึ่งระยะเวลาดังกล่าวน่าจะไม่เหมาะสมต่อกรอบเวลาของงานวิทยานพนธ์ งานวิทยานิพนธ์นี้จึงเลือกใช้การทำเหมืองข้อมูลเข้ามาช่วยสกัดรูปแบบพฤติกรรมที่น่าสนใจและข้อมูลสนับสนุนอื่น ๆ จากชุดข้อมูล ชุดข้อมูลที่งานวิทยานิพนธ์นี้ใช้ประกอบด้วยศัตรูในเกมที่ได้รับความนิยมและเป็นที่ยอมรับของผู้เล่น ซึ่งถูกอธิบายในรูปของแบบจำลองภาคีฯ

* 1. งานวิจัยเกี่ยวกับการออกแบบเกม

งานวิทยานิพนธ์นี้จำเป็นต้องสร้างแบบจำลองภาคีฯและการสร้างแบบจำลองจำเป็นต้องเข้าใจถึงองค์ประกอบต่าง ๆ ที่รวมกันเป็นสิ่งที่สนใจ ในที่นี้จึงสำรวจงานวิจัยทางด้านการออกแบบเกมซึ่งจะแสดงวิธีการวิเคราะห์เกมสำหรับศึกษาองค์ประกอบภายในเกม จุดประสงค์การสำรวจงานวิจัยกลุ่มนี้เป็นไปเพื่อศึกษาถึงวิธีที่ใช้วิเคราะห์เกมอย่างเป็นระบบที่มีการใช้งานอยู่แล้วในงานวิจัยอื่น และนำมาประยุกต์ใช้กับการวิเคราะห์ศัตรูของเกมแอ็คชั่นสองมิติแบบเน้นตัวละครในงานวิทยานิพนธ์นี้

Daniel Boutros [[29](#_ENREF_29)] วิเคราะห์เกมใน 4 มุมมองได้แก่ ภาพ, วิธีการควบคุม, รางวัล และความท้าทาย โดยใช้การเล่นเกมที่ต้องการวิเคราะห์เป็นเวลา 10 นาที สาเหตุที่เลือกใช้ระยะเวลาสั้นเช่นนี้มาจากแนวคิดทางจิตวิทยาว่าด้วยความประทับใจแรกพบจะเกิดจากการพบเจอภายใน 5 วินาทีแรก แต่เนื่องจากระยะเวลา 5 วินาทีนั้นสั้นเกินไปจนไม่สามารถสัมผัสสิ่งที่เกมมีอย่างทั่วถึงเพียงพอ ผู้วิเคราะห์จึงเพิ่มระยะเวลาเป็น 10 นาที

Kenneth Hullett และ Jim Whitehead [[30](#_ENREF_30)] วิเคราะห์แบบแผนการออกแบบ (Design pattern) เกมยิงปืนประเภทมุมมองบุคคลที่หนึ่งโดยการเล่นเกมแล้วบันทึกสิ่งที่พบของเกมโดยระบุว่าสิ่งที่พบดังกล่าวส่งผลต่อการกระทำของผู้เล่นอย่างไร หากสิ่งที่พบส่งผลให้การกระทำของผู้เล่นเป็นไปในทางเดียวกันจะนับเป็นแบบแผนการออกแบบเดียวกันและตั้งชื่อให้

Mark Nelson [[31](#_ENREF_31)] และ Gillian Smith และคณะ [[32](#_ENREF_32)] ต่างวิเคราะห์เกมด้วยวิธีการแยกย่อยองค์ประกอบ โดยแบ่งเกมที่สนใจออกเป็นองค์ประกอบย่อยลงไปจนกระทั่งองค์ประกอบดังกล่าวส่งผลต่อวิธีการของผู้เล่นเพียงแบบใดแบบหนึ่งเท่านั้น

วิธีการวิเคราะห์เกมของ Hullett, Nelson และ Smith และคณะเป็นการหาองค์ประกอบที่ส่งผลต่อตัวผู้เล่นเพียงอย่างใดอย่างหนึ่งทั้งสิ้น เพียงแต่ Hullett และ Whitehead วิเคราะห์แบบรวมกลุ่มซึ่งอาศัยการเก็บรวบรวมข้อมูลจากการเล่น ในขณะที่ Nelson และ Smith และคณะใช้การวิเคราะห์แบบแยกย่อยโดยดูจากภาพรวมแล้วแบ่งองค์ประกอบย่อยลงไป งานวิทยานิพนธ์นี้อาศัยความคุ้นเคยต่อเกมประเภทแอ็คชั่นแบบเน้นตัวละครของผู้วิจัยร่วมกับการวิเคราะห์แยกย่อยในส่วนที่ไม่ใช่ศัตรูแต่เป็นองค์ประกอบที่เกี่ยวข้อง จากนั้นตั้งชื่อให้กับองค์ประกอบต่าง ๆ

* 1. งานวิจัยเกี่ยวกับการทำเหมืองข้อมูลสำหรับข้อมูลแบบลำดับ

ปัจจุบันมีขั้นตอนวิธีสำหรับการทำเหมืองข้อมูลสำหรับข้อมูลลำดับหลายวิธี ไลบรารี่ SPMF [[33](#_ENREF_33)] รวบรวม 7 ขั้นตอนวิธีให้ใช้งาน โดยมีงานวิจัยของ Manika Verma และ Devarshi Mehta [[34](#_ENREF_34)] ทดลองเปรียบเทียบประสิทธิภาพการทำงานของขั้นตอนวิธี GSP, SPADE และ PrefixSpan ภายในไลบรารี่ SPMF และพบว่า PrefixSpan เป็นขั้นตอนวิธีที่มีประสิทธิภาพสูงที่สุดทั้งในแง่ของระยะเวลาที่ใช้และปริมาณหน่วยความจำที่ต้องการ งานวิทยานิพนธ์จึงเลือกนำ PrefixSpan มาใช้สำหรับการทำเหมืองข้อมูล

PrefixSpan [[8](#_ENREF_8)] เป็นขั้นตอนวิธีที่นำเสนอโดย Jian Pei และคณะ ขั้นตอนวิธีนี้หาลำดับย่อยที่พบบ่อยด้วยวิธีต่อเติมรูปแบบ (Prefix growth) โดยเริ่มต้นจากการนับค่าซัพพอร์ทของลำดับความยาว 1 แล้วแบ่งปริภูมิค้นหา (Divide search space) โดยการสร้างฐานข้อมูลภาพฉาย (Projected database) ของฐานข้อมูลต้นสำหรับทุกลำดับความยาว 1 ที่พบบ่อย (มีค่าซัพพอร์ทเกิน min\_sup ที่กำหนด) ในขั้นตอนสุดท้ายจะเป็นการวนซ้ำเพื่อต่อเติมลำดับย่อยที่ได้จากการวนซ้ำรอบก่อนหน้าด้วยลำดับย่อยที่พบบ่อยใหม่ในการวนซ้ำรอบปัจจุบัน แล้วสร้างฐานข้อมูลภาพฉายใหม่ของฐานข้อมูลภาพฉายปัจจุบันสำหรับลำดับย่อยใหม่จนกระทั่งไม่สามารถสร้างฐานข้อมูลภาพฉายได้อีก ทุกลำดับย่อยที่พบบ่อยที่ได้จากการวนซ้ำทุกรอบเป็นคำตอบของการทำเหมืองข้อมูลนี้ รหัสเทียมของขั้นตอนวิธีเป็นไปดังรูปที่ 3.3

|  |
| --- |
| การทำเหมืองข้อมูลบนฐานข้อมูลลำดับ  ซึ่งประกอบด้วยลำดับ และมีเซ็ตของไอเท็ม กำหนดค่าซัพพอร์ทขั้นต่ำ ด้วยวิธี PrefixSpan เป็นดังนี้  Initialize ,  For ( in ) : if()  For(in ) do:  Create -projected database of  If( is not empty) Recursive()  Return  as result  เมื่อฟังก์ชั่นวนซ้ำ Recursive(Sequence , Projected database ) มีรหัสเทียมดังนี้  Initialize  For(in ) do: if()  For(in ) do:    Create -projected database of  If( is not empty) Recursive()  และมีนิยามที่สำคัญต่อไปนี้   * เป็นตัวเติมหน้า (Prefix) ของลำดับ กำหนดให้ ก็ต่อเมื่อ    + ซึ่งไอเท็มทั้งหมดของ  จะต้องมาก่อน  ตามลำดับอักษร เช่น  และ  เป็นต้น * เป็นตัวเติมหลัง (Suffix) ของลำดับ เมื่อพิจารณาลำดับ กำหนดให้ และ    + หาก  ไม่เป็นไอเท็มเซ็ตว่าง ให้ใช้สัญลักษณ์ เป็นไอเท็มแรกของไอเท็มเซ็ต เช่น * ฐานข้อมูลภาพฉายสำหรับลำดับ ของ (-projected database of ) เมื่อ  เป็นฐานข้อมูลลำดับ ซึ่งประกอบด้วยลำดับ ประกอบด้วยลำดับย่อย ;  ซึ่ง   + เป็นตัวเติมหลังของ  เมื่อพิจารณาลำดับ   + เป็นลำดับย่อยของ  ซึ่งเลือกลำดับย่อยโดยเริ่มจาก ที่ปรากฏครั้งแรกใน จนจบลำดับ   + ตัวอย่าง เช่น , จะได้ เป็นต้น |

รูปที่ 3.3 แสดงขั้นตอนวิธี PrefixSpan

งานวิทยานิพนธ์นี้ใช้ PrefixSpan ในการหารูปแบบพฤติกรรมที่น่าสนใจจากชุดข้อมูลรูปแบบพฤติกรรมที่ถูกแปลงให้อยู่ในรูปของลำดับแล้ว นอกจากนี้ ชุดข้อมูลรูปแบบพฤติกรรมจะถูกแปลงให้อยู่ในรูปลำดับเชื่อม (Joined sequence) ซึ่งงานวิทยานิพนธ์นี้จะปรับปรุงขั้นตอนวิธี PrefixSpan ให้เหมาะสมกับลักษณะของข้อมูลดังกล่าว รายละเอียดทั้งหมดจะกล่าวถึงในหัวข้อ 4.2.3

นอกจาก PrefixSpan แล้ว ในที่นี้ได้ทำการสำรวจงานวิจัยที่เกี่ยวข้องกับการทำเหมืองข้อมูลเพื่อหารูปแบบของการเรียกใช้ฟังก์ชั่น (Function call pattern) เนื่องจากการทำเหมืองข้อมูลบนชุดข้อมูลรูปแบบพฤติกรรมที่แปลงเป็นลำดับมีจุดประสงค์เพื่อหารูปแบบของการกระทำซึ่งอยู่ในรูปของฟังก์ชั่น

Takashi Ishio และคณะ [[35](#_ENREF_35)] ทำเหมืองข้อมูลเพื่อหารูปแบบของการเรียกใช้ฟังก์ชั่นสำหรับไลบรารี่ที่เขียนขึ้นด้วยภาษาจาวา (Java) โดยแปลงซอร์สโค้ด (Source code) ให้อยู่รูปของลำดับแล้วใช้ขั้นตอนวิธี PrefixSpan ซึ่งปรับปรุงแล้วในการทำเหมืองข้อมูล

ลำดับซอร์สโค้ดของงานวิจัยนี้ประกอบด้วยสัญพจน์สองชนิด ได้แก่ เอเลเมนท์เมธอด (Method call element) และเอเลเมนท์ควบคุม (Control element) โดยวิธีการสร้างลำดับจะได้จากการพิจารณาลำดับการทำงานของซอร์สโค้ด เมื่อพบการเรียกใช้เมธอด จะเพิ่มเอเลเมนท์เมธอดเข้าไปในลำดับ เมื่อพบคำสงวนสำหรับโครงสร้างควบคุม เช่น if หรือ for จะเพิ่มเอเลเมนท์ควบคุมเข้าไปในลำดับ (ในที่นี้ คือ IF และ FOR) โดยมีการใช้งานตัวปิดบล็อก เช่น ENDIF, END-LOOP ร่วมด้วย เอเลเมนท์ควบคุมแต่ละตัวภายในลำดับจะรู้ว่าตัวปิดบล็อกที่คู่กับตนเองคืออะไรโดยการตรวจสอบต้นไม้วากยสัมพันธ์นามธรรม (Abstract syntax tree -- AST) และเมื่อขั้นตอนวิธี PrefixSpan พบรูปแบบที่ประกอบด้วยเอเลเมนท์ควบคุม ก็จะดึงตัวปิดบล็อกที่คู่กันมาไว้ในรูปแบบด้วย

ข้อจำกัดของงานวิจัยนี้ คือ ในกรณีที่มีเอเลเมนท์เมธอดอยู่ภายใต้โครงสร้างควบคุม 2 ชั้น และโครงสร้างควบคุมในชั้นที่ 2 ไม่ปรากฏบ่อยเพียงพอ จะส่งผลให้รูปแบบที่ได้จากการทำเหมืองข้อมูลขาดข้อมูลของโครงสร้างชั้นที่ 2 และไม่มีทางรู้ได้ว่าเอเลเมนท์เมธอดเป็นเมธอดที่อยู่ในโครงสร้างข้อมูล 2 ชั้น ซึ่งการสูญเสียข้อมูลเช่นนี้จะส่งผลต่อรูปแบบพฤติกรรมหากนำวิธีเดียวกันนี้มาใช้ งานวิทยานิพนธ์นี้จึงใช้วิธีการติดป้ายข้อมูลโครงสร้างควบคุม (ซึ่งในกรณีของงานวิทยานิพนธ์นี้ มีเพียงโครงสร้างเงื่อนไข) ให้กับการกระทำ ทำให้สามารถเก็บข้อมูลไว้ได้ครบถ้วนกว่า

1. แนวคิดและวิธีการดำเนินงาน

จุดประสงค์ของงานวิทยานิพนธ์นี้คือพัฒนาวิธีการสร้างรูปแบบพฤติกรรมที่ตายตัวของศัตรูแบบอัตโนมัติสำหรับเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร จากการค้นคว้างานวิจัยที่เกี่ยวข้อง พบว่าไม่มีงานวิจัยที่เกี่ยวข้องกับการสร้างรูปแบบพฤติกรรมที่ตายตัวของศัตรูในเกมประเภทนี้โดยตรง งานวิทยานิพนธ์นี้จึงต้องเริ่มต้นจากการกำหนดนิยามของศัตรูและพฤติกรรมที่เป็นไปได้โดยใช้การวิเคราะห์เพื่อศึกษาองค์ประกอบของเกม แล้วจึงสร้างแบบจำลองของศัตรู (Enemy model) ซึ่งจะถูกปรับปรุงไปเป็นแบบจำลองภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร (2D avatar-based action game agent model) เพื่อความครอบคลุมของพฤติกรรม ต่อมาจึงใช้การทำเหมืองข้อมูลเพื่อสกัดรูปแบบพฤติกรรมของศัตรูที่เป็นที่ยอมรับของผู้เล่นรวมถึงข้อมูลสนับสนุนที่จำเป็นจากชุดข้อมูลซึ่งได้จากศัตรูที่ปรากฏในเกมที่ได้รับความนิยม รูปแบบพฤติกรรมของศัตรูและข้อมูลสนับสนุนที่ได้จากการทำเหมืองข้อมูลจะถูกแปลงให้อยู่ในรูปของปัญหาการตอบสนองเงื่อนไขบังคับ และใช้ตัวแก้ปัญหาเงื่อนไขบังคับในการสร้างศัตรูขึ้นมาโดยการแก้ปัญหาดังกล่าว ศัตรูผลลัพธ์ซึ่งอยู่ในรูปของภาษาอธิบายภาคีฯที่ได้จะถูกวัดผลโดยใช้ผู้เล่นและแบบสอบถาม ภาพรวมของขั้นตอนวิธีในการสร้างรูปแบบพฤติกรรมของศัตรูอัตโนมัติเป็นไปดังรูปที่ 4.1



รูปที่ 4.1 ภาพรวมของขั้นตอนวิธี

* 1. แบบจำลองภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร

งานวิทยานิพนธ์นี้สร้างแบบจำลองภาคีฯขึ้นจากการวิเคราะห์ศัตรูและองค์ประกอบอื่น ๆ ที่เกี่ยวข้องที่ปรากฏภายในเกมแอ็คชั่นสองมิติแบบเน้นตัวละครจำนวน 5 เกม ได้แก่ Metroid, Super C, Megaman, Megaman 4 และ Shovel Knight ซึ่งแต่ละเกมเป็นเกมที่ได้รับความนิยมในยุคสมัยที่เกมนั้นได้รับการผลิตออกมา นอกจากนี้ ช่วงเวลาที่แต่ละเกมถูกผลิตออกมานั้นมีตั้งแต่ปี ค.ศ. 1968 จนถึง ค.ศ. 2014 จึงน่าจะทำให้รูปแบบพฤติกรรมที่ได้จากการวิเคราะห์สามารถครอบคลุมแนวคิดของการออกแบบในต่างช่วงเวลาได้

แบบจำลองภาคีฯจะอยู่ในรูปของภาษาอธิบายภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร ซึ่งจุดแข็งของการใช้ภาษาอธิบายภาคีฯเป็นแบบจำลองนั้นทำให้มนุษย์สามารถทำความเข้าใจแบบจำลองได้ง่าย รวมไปถึงสามารถสร้างตัวแปลภาษาขึ้นมาเพื่อแปลแบบจำลองและแสดงผลลัพธ์ในเกมได้ ในหัวข้อนี้จึงได้อธิบายขั้นตอนการสร้างแบบจำลองตั้งแต่แบบจำลองศัตรูไปจนถึงการปรับปรุงเป็นแบบจำลองภาคีฯ รวมถึงรายละเอียดของภาษาอธิบายภาคีฯ และวิธีการทำงานของตัวแปลภาษาเพื่อแสดงให้เห็นความสอดคล้องระหว่างแบบจำลองและสิ่งที่จะเกิดขึ้นภายในเกม

* + 1. การวิเคราะห์ศัตรู

ก่อนเริ่มต้นวิเคราะห์ศัตรู ในที่นี้จำเป็นต้องให้นิยามของ “ศัตรู” ในบริบทของงานวิทยานิพนธ์นี้เพื่อแยกศัตรูออกจากองค์ประกอบอื่นของเกมเสียก่อน ในงานวิจัยเกี่ยวกับเกมแพลทฟอร์มของ Smith [[32](#_ENREF_32)] ได้กำหนดให้ศัตรูเป็นส่วนหนึ่งของสิ่งกีดขวาง (Obstacle) โดยนิยามของสิ่งกีดขวางภายในงานวิจัยดังกล่าว คือ “องค์ประกอบที่สามารถขัดขวางการเคลื่อนไหวของตัวละครอวตารหรือสามารถสร้างความเสียหายให้กับตัวละครอวตารได้” หนังสือของ Adams [[3](#_ENREF_3)] นิยามศัตรูในเกมยุคเก่าไว้ว่า “ศัตรูเป็นภาคีที่สามารถโจมตีตัวละครอวตารและเคลื่อนที่ด้วยรูปแบบที่ตายตัวที่ผู้เล่นสามารถเรียนรู้เพื่อหลบหลีกได้” เมื่อพิจารณานิยามของทั้งสองงานร่วมกัน งานวิทยานิพนธ์นี้จึงนิยามศัตรูว่า

“ศัตรู คือ ภาคีที่มีรูปแบบพฤติกรรมและสร้างความเสียหายให้ตัวละครอวตารได้”

นิยามส่วนที่ระบุว่าศัตรูจะต้องสร้างความเสียหายได้ มีจุดประสงค์เพื่อคัดกรองสิ่งกีดขวางอื่น ๆ ตามนิยามของ Smith เช่น กำแพง ซึ่งมีผลในการขัดขวางการเคลื่อนที่ของผู้เล่นเท่านั้น ในทางกลับกัน นิยามที่ระบุว่าศัตรูคือภาคีที่มีรูปแบบพฤติกรรมจะทำให้ครอบคลุมศัตรูที่ไม่เคลื่อนที่ เช่น ศัตรูรูปแบบป้อมปืน ซึ่งจะไม่ถูกนับเป็นศัตรูหากยึดตามนิยามของ Adams

การวิเคราะห์ศัตรูในเกมแอ็คชั่นทั้ง 5 เกมที่ถูกเลือกจะเริ่มต้นจากการค้นหาองค์ประกอบที่เป็นศัตรูตามนิยามที่กำหนด จากนั้นพิจารณาองค์ประกอบที่มีส่วนเกี่ยวข้องกับศัตรูและแยกย่อยองค์ประกอบดังกล่าวให้เหลือเพียงองค์ประกอบที่เกี่ยวข้องกับศัตรูโดยตรง รูปที่ 4.2 แสดงองค์ประกอบที่เป็นรูปธรรมส่วนหนึ่งของเกม Megaman 4

|  |
| --- |
| แถบสถานะ (Status bar)  พื้น  กระสุน  กำแพง  ศัตรู  อุปสรรคอันตราย (Hazard)  ตัวละครอวตาร  ไอเท็ม |
| รูปที่ 4.2 องค์ประกอบส่วนหนึ่งของเกม Megaman 4 |

* + 1. องค์ประกอบที่เกี่ยวข้องกับศัตรู

พฤติกรรมของศัตรูย่อมส่งผลหรือได้รับผลกระทบจากองค์ประกอบที่เกี่ยวข้อง เช่น กำแพงอาจจะขัดขวางการเคลื่อนที่ของศัตรูและส่งผลให้ศัตรูเปลี่ยนทิศทางการเคลื่อนที่ เป็นต้น การทำความเข้าใจถึงองค์ประกอบเหล่านี้จึงมีบทบาทสำคัญในการสร้างแบบจำลองศัตรู รวมไปถึงการสร้างเกมทดสอบสำหรับใช้วัดผลศัตรูซึ่งจำเป็นต้องควบคุมองค์ประกอบที่เกี่ยวข้องให้คงที่ตลอดทุกการทดสอบ

จากนิยาม “ศัตรู คือ ภาคีที่มีรูปแบบพฤติกรรมและสร้างความเสียหายให้ตัวละครอวตารได้” จะพบคำสำคัญได้แก่ รูปแบบพฤติกรรม, ความเสียหาย และตัวละครอวตาร ซึ่งทำให้ค้นพบองค์ประกอบที่เกี่ยวข้องได้แก่ โครงสร้างพื้นที่, กลไกการต่อสู้ และตัวละครอวตาร

* + - 1. โครงสร้างพื้นที่

รูปแบบพฤติกรรมกล่าวถึงความสามารถในการกระทำการบางอย่างในโลกของเกม หากยกตัวอย่างเกม Megaman 4 จะมีศัตรูบางชนิดสามารถเดินไปบนพื้น หรือในเกม Super C ซึ่งกระสุนของศัตรูบางชนิดไม่สามารถเคลื่อนที่ผ่านกำแพงได้ นอกจากนี้ ในทุกเกมที่ทำการวิเคราะห์ ตัวละครอวตารจะถูกกำแพงหรือพื้นกำหนดบริเวณที่สามารถเข้าถึงได้เช่นกัน ลักษณะดังกล่าวนี้แสดงให้เห็นถึงองค์ประกอบที่มีหน้าที่ระบุถึงขอบเขตพื้นที่ที่สามารถผ่านได้หรือผ่านไม่ได้ ในที่นี้จึงกำหนดให้องค์ประกอบที่ทำหน้าที่เช่นนี้เป็น “โครงสร้างพื้นที่” (Level layout)

โครงสร้างพื้นที่ในเกมที่นำมาวิเคราะห์แบ่งออกได้เป็น 2 ประเภท ได้แก่ พื้นที่ที่ผ่านได้ และพื้นที่ที่ผ่านไม่ได้ ในบางเกม เช่น Super C ศัตรูบางชนิดสามารถเคลื่อนที่ผ่านบริเวณที่ตัวละครอวตารผ่านไม่ได้ ในขณะที่ศัตรูอื่นอีกหลายชนิดไม่สามารถทำเช่นเดียวกันนี้ นั่นแสดงให้เห็นว่า ศัตรูแต่ละชนิดมีรายการที่ระบุว่าตนสามารถเคลื่อนผ่านพื้นที่บริเวณประเภทใดได้บ้าง ในงานวิทยานิพนธ์นี้จะกำหนดให้เกมทดสอบมีโครงสร้างพื้นที่ 2 ประเภทดังข้างต้น และเพิ่มคุณสมบัติให้กับศัตรูเพื่อแยกแยะบริเวณที่ผ่านได้หรือไม่ได้ โดยไม่ใช้ข้อมูลในรูปแบบรายการ ค่าคุณสมบัติที่เกี่ยวข้องกับโครงสร้างพื้นที่จะกล่าวถึงในหัวข้อ 4.1.3.1 อีกครั้งหนึ่ง

* + - 1. กลไกการต่อสู้

ความเสียหายในเกมแอ็คชั่นแบบเน้นตัวละคร คือ การสูญเสียค่าพลังชีวิต หากพลังชีวิตหมดจะทำให้ตัวละครอวตารหรือศัตรูนั้นตาย ในกรณีของตัวละครอวตาร แต่ละเกมจะมีวิธีจัดการกับตัวละครอวตารที่ตายแตกต่างกัน เช่น ในเกม Megaman ผู้เล่นจะถูกนำไปกลับไปยังจุดที่กำหนด หรือในเกม Metroid เกมจะจบลงทันทีเมื่อตาย ในกรณีของศัตรู ศัตรูจะถูกนำออกจากโลกของเกมไป ความเสียหายต่อตัวละครอวตารและศัตรูอาจเกิดจากอุปสรรคอันตรายหรือการต่อสู้ระหว่างภาคีก็ได้ แต่เนื่องจากงานวิทยานิพนธ์นี้ไม่ได้นำลักษณะของพื้นที่นอกจากคุณสมบัติการผ่านได้ของพื้นที่มาคิด ดังนั้นกรณีของอุปสรรคอันตรายจึงถูกตัดออกจากเรื่องการคิดความเสียหาย

การต่อสู้ระหว่างตัวละครอวตารและศัตรูในเกมตัวอย่าง หากมองจากมุมมองของผู้เล่น จะพบลักษณะการโจมตี 2 ประเภท ได้แก่ การโจมตีระยะใกล้ (Melee attack) และการโจมตีระยะไกล (Ranged attack) โดยตัดสินจากระยะหวังผลของการโจมตี เช่น ในเกม Shovel Knight ตัวละครอวตารจะโจมตีโดยใช้พลั่วแทงออกไปด้านหน้า ในมุมมองของผู้เล่นที่เห็นภาพเช่นนี้จะมองว่าเป็นการโจมตีระยะใกล้ ในทางตรงข้าม ตัวละครในเกมอื่นที่วิเคราะห์ซึ่งโจมตีด้วยการยิงกระสุนจะถูกมองว่าเป็นการโจมตีระยะไกลแทน การโจมตีสองประเภทในมุมมองของผู้เล่นนี้สามารถทำให้เหลือรูปแบบเดียวโดยการอธิบายด้วยตัวตรวจจับการชน (Collider) แทน เมื่อตัวตรวจจับการชนของการโจมตีกระทบกับตัวตรวจจับการชนของเป้าหมาย จะถือว่าเป้าหมายถูกโจมตีและเสียพลังชีวิต ทั้งหมดที่กล่าวมาล้วนเป็นสิ่งที่เรียกว่ากลไกการต่อสู้ รายละเอียดของกลไกจะกล่าวถึงในหัวข้อ 4.1.3 อีกครั้งหนึ่ง

* + - 1. ตัวละครอวตาร

ตัวละครอวตารเป็นภาคีที่ถูกควบคุมโดยผู้เล่น ในเกมแอ็คชั่นแบบเน้นตัวละคร พฤติกรรมที่สามารถกระทำได้ของตัวละครผู้เล่นจะถูกกำหนดไว้แล้ว สำหรับเกมตัวอย่างที่วิเคราะห์ พฤติกรรมที่สามารถกระทำได้ของตัวละครอวตารจะใกล้เคียงกันทั้งหมด ผู้เล่นสามารถควบคุมให้ตัวละครเดินไปทางซ้ายและขวาได้ สามารถสั่งให้ตัวละครกระโดดและโจมตีได้ นอกเหนือจากนี้ แต่ละเกมจะมีชุดพฤติกรรมเฉพาะตัวที่แตกต่างกันออกไปบ้าง เช่น ใน Super C จะมีบางฉากที่ตัวละครอวตารไม่สามารถกระโดดได้ แต่ผู้เล่นสามารถควบคุมตัวละครให้เดินไปในทิศซ้าย-ขวา-บน-ล่างแทน หรือในเกม Megaman 4 ตัวละครจะสามารถเคลื่อนที่แบบไถลได้

ความเกี่ยวพันระหว่างศัตรูและตัวละครอวตารจะอยู่ใน 2 ลักษณะ ได้แก่ ความเป็นพวกพ้องเมื่อเกิดการต่อสู้ระหว่างภาคี และพฤติกรรมของศัตรูที่ส่งผลหรือได้รับผลกระทบจากการเคลื่อนไหวของตัวละครอวตาร ในกรณีแรกจะกล่าวถึงว่าฝ่ายใดสามารถทำการโจมตีฝ่ายใดได้ ทำให้แบบจำลองภาคีฯต้องมีคุณสมบัติสำหรับการแบ่งฝ่าย ผลของคุณสมบัติดังกล่าวจะกล่าวถึงในหัวข้อ 4.1.3 สำหรับกรณีของผลกระทบต่อพฤติกรรมของศัตรูจะทำให้เกิดการกระทำและข้อมูลแบบจำเพาะบางชนิด

* + 1. องค์ประกอบของศัตรู

ในการวิเคราะห์ศัตรู ผู้วิจัยแบ่งศัตรูออกเป็นองค์ประกอบย่อย โดยเริ่มต้นจากค่าคุณสมบัติ (Property) ที่จำเป็นต่อตัวศัตรูซึ่งเกิดจากองค์ประกอบอื่นๆที่เกี่ยวข้อง เช่น ค่าคุณสมบัติสำหรับการแยกแยะความสามารถในการผ่านบริเวณเกิดจากโครงสร้างพื้นที่ เป็นต้น จากนั้นจึงวิเคราะห์ว่าศัตรูมีการเปลี่ยนแปลงค่าคุณสมบัติต่าง ๆ ไปอย่างไร และรวบรวมวิธีการเปลี่ยนแปลงค่าคุณสมบัติเหล่านั้นเป็นการกระทำ (Action) โดยการกระทำของศัตรูจะถูกการจัดลำดับว่าให้ทำสิ่งใดก่อนสิ่งใดหลัง รวมถึงมีเงื่อนไขของการกระทำบางอย่าง และการเปลี่ยนชุดการกระทำ ทำให้เกิดเป็นองค์ประกอบที่ว่าด้วยลำดับพฤติกรรม (Behavior sequence), การทดสอบเงื่อนไข (Condition testing), การสอบถามข้อมูล (Information query) และเรื่องของสถานะ (State) และลำดับพฤติกรรมคู่ขนาน (Parallel sequence)

* + - 1. ค่าคุณสมบัติ (Property)

รูปแบบพฤติกรรมของศัตรูที่พบจากการสังเกตจะสามารถเปลี่ยนแปลงค่าคุณสมบัติบางอย่าง ค่าคุณสมบัติดังกล่าวอาจจะเป็นค่าคุณสมบัติที่ผู้เล่นสามารถรับรู้ได้อย่างชัดเจน เช่น ตำแหน่งของศัตรูในโลก หรือคุณสมบัติที่ไม่ชัดเจน เช่น ตัวนับจำนวนการกระทำ ในศัตรูบางตัวที่มีรูปแบบพฤติกรรมเปลี่ยนแปลงได้หลังจากกระทำพฤติกรรมบางอย่างซ้ำกันจนครบตามจำนวนที่กำหนด คำถามสำคัญสำหรับประเด็นนี้คือ "มีสิ่งใดที่เป็นค่าคุณสมบัติของศัตรูบ้าง?"

ในที่นี้จะทำการวิเคราะห์ค่าคุณสมบัติที่จำเป็นต้องมีโดยพิจารณาจากความสัมพันธ์ระหว่างศัตรูและองค์ประกอบที่เกี่ยวข้องทั้ง 3 ได้แก่ โครงสร้างพื้นที่ , กลไกการต่อสู้ และตัวละครอวตาร

ดังที่ได้กล่าวไปแล้วในหัวข้อ 4.1.2.1 ว่า โครงสร้างพื้นที่มีผลในการจำกัดบริเวณที่ตัวละครอวตารและศัตรูสามารถเคลื่อนผ่านไปได้ โดยพื้นที่จะถูกแบ่งเป็น 2 ประเภทได้แก่ พื้นที่ที่ผ่านได้ และพื้นที่ที่ผ่านไม่ได้ แต่เนื่องจากในเกมที่วิเคราะห์มีศัตรูหลายตัวที่สามารถเคลื่อนที่ผ่านพื้นที่ที่ผ่านไม่ได้ เช่น ศัตรู Alien Egg ใน Super C, Taketento ใน Megaman 4 และ Swarm ใน Metroid เป็นต้น ในที่นี้จึงกำหนดให้มีค่าคุณสมบัติ “ผ่านได้” (Through) ที่ทำให้ศัตรูสามารถผ่านได้ทุกพื้นที่หากมีคุณสมบัตินี้ นอกเหนือจากนี้ เนื่องจากเกมนี้เป็นเกมแอ็คชั่นสองมิติ การตรวจสอบว่าศัตรูที่กำหนดอยู่ในบริเวณใดจึงต้องมีข้อมูลตำแหน่งและขนาดของศัตรู รวมไปถึงการบอกขนาดและตำแหน่งของพื้นที่ประเภทต่างๆ งานวิทยานิพนธ์นี้รวมคุณสมบัติตำแหน่งและขนาดเข้าด้วยกันและเรียกว่า “ตัวตรวจจับการชน” (Collider) เป็นอีกค่าคุณสมบัติของศัตรู ซึ่งตัวละครอวตารและโครงสร้างพื้นที่เองต่างก็ใช้คุณสมบัตินี้ในการกำหนดขอบเขตเช่นเดียวกัน

เกมแอ็คชั่นถูกแบ่งเป็นประเภทย่อยหลายประเภทและหนึ่งในประเภทย่อยคือเกมแพลทฟอร์ม ซึ่งเกมตัวอย่างเองก็เป็นเกมแพลทฟอร์มเป็นส่วนใหญ่ จุดที่เป็นกุญแจสำคัญของเกมแพลทฟอร์มที่ทำให้เกิดลักษณะการเล่นแบบกระโดดไปตามพื้นต่างระดับ คือ แรงดึงดูด ซึ่งคอยดึงให้ตัวละครอวตารหรือศัตรูที่กระโดดกลับลงมาบนพื้น ทว่าในเกมแพลทฟอร์มมักจะมีศัตรูที่สามารถลอยได้โดยไม่สนใจแรงดึงดูด เช่น ใน Megaman 4 ศัตรู Battonton สามารถบินเข้าหาตัวละครอวตาร หรือศัตรู M-422A ลอยค้างอยู่กลางอากาศได้ เป็นต้น นั่นแสดงให้เห็นความจำเป็นของค่าคุณสมบัติ “ผลของแรงดึงดูด” (Gravity effect) สำหรับใช้ระบุว่าศัตรูที่กำหนดรับผลจากแรงดึงดูดมากเพียงใด

กลไกการต่อสู้เป็นอีกหนึ่งองค์ประกอบที่สำคัญซึ่งกล่าวถึงวิธีการต่อสู้ระหว่างตัวละครอวตารและศัตรูว่าเกิดขึ้นอย่างไรและดำเนินไปอย่างไร ในหัวข้อ 4.1.2.2 ได้กล่าวถึงการโจมตี 2 ประเภทในมุมมองของผู้เล่น ได้แก่ การโจมตีระยะใกล้ และการโจมตีระยะไกล การโจมตีระยะไกลมักจะอยู่ในรูปแบบของการปล่อยกระสุนออกไป ในขณะที่การโจมตีระยะใกล้มักจะเกิดขึ้นในรูปแบบของการเปลี่ยนภาพท่าทางของตัวละคร เช่น ศัตรู Specter Knight จากเกม Shovel Knight จะเปลี่ยนจากภาพอยู่นิ่งไปเป็นภาพท่าฟันเมื่อใช้เคียวโจมตี ทว่ากลไกเบื้องหลังของท่าโจมตีระยะใกล้สามารถใช้หลักการเดียวกับการโจมตีระยะไกลได้ โดยใช้การปล่อยกระสุนที่มองไม่เห็นออกไปเมื่อทำการโจมตีให้รับกับภาพท่าทางของศัตรูนั้น ผู้เล่นก็จะไม่รู้สึกว่าการโจมตีนั้นเป็นการโจมตีระยะไกล ด้วยแนวคิดของการใช้กระสุนสำหรับการโจมตีทุกประเภท กระสุนจะส่งผลในการต่อสู้เมื่อกระสุนนั้นเข้าถึงเป้าหมาย การตรวจสอบว่ากระสุนเข้าถึงเป้าหมายหรือไม่ จำเป็นต้องมีการตรวจสอบขนาดและตำแหน่งของกระสุนกับขนาดและตำแหน่งของเป้าหมาย ในที่นี้จึงต้องมีการนิยามสิ่งที่เป็นภาคีกระสุนซึ่งมีค่าคุณสมบัติตัวตรวจจับการชนเช่นเดียวกัน งานวิทยานิพนธ์นี้จะนิยามภาคีกระสุนเป็นภาคีแบบเดียวกับศัตรูในหัวข้อ 4.1.4.2

เมื่อกระสุนเข้าถึงเป้าหมายแล้วย่อมหมายถึงเกิดการโจมตีขึ้น ประเด็นสำคัญต่อมาจึงเกี่ยวข้องกับความเสียหายที่เกิดขึ้นจากการต่อสู้ ความเสียหายที่ได้กล่าวถึงไปแล้วในหัวข้อ 4.1.2.2 เป็นค่าความเสียหายที่เกิดขึ้นต่อพลังชีวิตของตัวละครอวตารหรือศัตรู ศัตรูจึงต้องมีค่าคุณสมบัติ “พลังชีวิต” (Life) นอกจากนี้ ในเกมที่วิเคราะห์ ศัตรูและกระสุนโดยส่วนใหญ่จะสามารถสร้างความเสียหายให้กับตัวละครอวตารด้วยการชนได้ ทำให้ทั้งสองจำเป็นต้องมีค่าคุณสมบัติ “พลังโจมตี” (Attack) ทว่าฝ่ายผู้เล่นเองก็สามารถโจมตีศัตรูด้วยกระสุน แต่ไม่ใช่ศัตรูทุกประเภทจะได้รับความเสียหายจากการโจมตีด้วยเช่นกัน ความสัมพันธ์ในส่วนของ “ใคร” สามารถโจมตีใส่ “ใคร” ได้ทำให้เกิดค่าคุณสมบัติสองค่าที่สำคัญได้แก่ “ฝ่ายโจมตี” (Attacker) และ “ฝ่ายตั้งรับ” (Defender) โดยเมื่อฝ่ายโจมตีชนกับฝ่ายตั้งรับ จะสร้างความเสียหายต่อพลังชีวิตของฝ่ายตั้งรับเท่ากับพลังโจมตีของตนเอง

การปะทะกันของฝ่ายโจมตีและฝ่ายตั้งรับอาจไม่ได้ทำให้เกิดความเสียหายเสมอไป เช่น ศัตรู Metall ใน Megaman หรือศัตรูที่ถือโล่ใน Shovel Knight ไม่รับความเสียหายจากการโจมตีของผู้เล่น ทำให้จำเป็นต้องมีค่าคุณสมบัติ “คงกระพัน” (Invulnerable) สำหรับศัตรูเพื่อระบุว่า ศัตรูที่มีคุณสมบัตินี้จะไม่รับความเสียหายถึงแม้ตนซึ่งเป็นฝ่ายตั้งรับจะถูกโจมตีด้วยฝ่ายโจมตี

การใช้งานคุณสมบัติฝ่ายโจมตีและฝ่ายตั้งรับยังไม่สามารถครอบคลุมในบางกรณี เนื่องจากศัตรูอาจจะโจมตีใส่ศัตรูอื่นได้หากศัตรูเป้าหมายดังกล่าวเป็นฝ่ายตั้งรับ ในที่นี้จึงต้องเพิ่มค่าคุณสมบัติ “กลุ่ม” (Group) เพื่อระบุว่าเป็นศัตรูหรือกระสุนเป็นของฝ่ายใด กฎที่เกี่ยวข้องกับกลุ่มคือฝ่ายตั้งรับไม่รับการโจมตีจากฝ่ายโจมตีที่อยู่กลุ่มเดียวกัน

ค่าคุณสมบัติที่วิเคราะห์มาทั้งหมดข้างต้น จะแสดงสรุปอีกครั้งหนึ่งในหัวข้อ 4.1.4.1

* + - 1. การกระทำ (Action)

การกระทำของศัตรูมีหน้าที่เปลี่ยนแปลงค่าคุณสมบัติบางอย่างของตนเองอย่างรวมไปถึงเปลี่ยนแปลงสถานะของโลกของเกมหรือสิ่งอื่นที่อยู่ในโลก การกระทำของศัตรูที่เห็นจากมุมมองของผู้เล่นอาจส่งผลต่อค่าคุณสมบัติหรือสถานะหลายอย่างหรือได้รับผลจากค่าสถานะต่าง ๆ ในคราวเดียวกัน เพื่อให้การวิเคราะห์เป็นไปอย่างมีระบบ การกระทำของศัตรูที่ผู้วิเคราะห์มองเห็นในฐานะของผู้เล่นจะถูกแยกย่อยเพื่อค้นหาค่าคุณสมบัติหรือสถานะต่าง ๆ ที่การกระทำส่งผลกระทบ แล้วจดบันทึกไว้ในรูปของ 2-tuple ซึ่งประกอบด้วยการกระทำในมุมมองของผู้เล่น และรายการของค่าคุณสมบัติ/ค่าสถานะที่ได้รับผลจากการกระทำ ตารางที่ 4.1 แสดงตัวอย่าง 2-tuple ของการกระทำ

|  |  |
| --- | --- |
| ตารางที่ 4.1 ตัวอย่าง 2-tuple ของการกระทำและผลกระทบ | |
| การกระทำที่สังเกตได้ | ค่าคุณสมบัติที่ได้รับ/ส่งผลกระทบ |
| เดินไปทางซ้าย (หรือขวา) | ค่าแกน x ของตัวตรวจจับการชน |
| กระโดด | ค่าแกน x และค่าแกน y ของตัวตรวจจับการชน, ค่าแรงดึงดูดของโลกของเกม |
| ยิงกระสุน | รายการวัตถุที่อยู่ในโลกของเกม |

หากอธิบายกระทำของศัตรูด้วยค่าคุณสมบัติที่ได้รับผลกระทบ จะทำให้การกระทำอยู่ในระดับต่ำมากเทียบได้กับตัวดำเนินการในงานของ Zook [[27](#_ENREF_27)] ส่งผลให้ปริภูมิค้นหามีขนาดใหญ่มาก ไม่ช่วยในการแก้ข้อจำกัดต่าง ๆ ที่พบในงานของ Zook ในที่นี้จึงมีความจำเป็นในการรวมผลกระทบต่อหลายคุณสมบัติเป็นหนึ่งการกระทำที่มีความหมายมากขึ้นโดยอิงจากเกมตัวอย่าง เช่น การกระทำกระโดดที่สังเกตได้ ไม่ควรแยกเป็นการเปลี่ยนแปลงของค่าแกน x และค่าแกน y ของตัวตรวจจับการชน แต่ควรคงไว้เป็นพฤติกรรมกระโดดซึ่งได้รับผลจากแรงดึงดูด

นอกจากนี้ การกระทำบางอย่างที่พบในเกมตัวอย่างจะมีความคล้ายคลึงกัน เช่น การเดินในทิศทางต่าง ๆ ด้วยความเร็วที่แตกต่างกัน หากนิยามการกระทำเหล่านี้แยกออกจากกันจะทำให้เกิดความซ้ำซ้อนกันได้ ตัวอย่างของความซ้ำซ้อนได้แก่ การเดินไปทางซ้ายและเดินไปทางขวา ซึ่งเป็นการเปลี่ยนแปลงค่าคุณสมบัติเดียวกัน แต่เครื่องหมายแตกต่างกัน หรือแม้กระทั่งการเดินไปในทิศอื่นเองก็เป็นการเปลี่ยนแปลงที่คุณสมบัติตำแหน่งของตัวตรวจจับการชนเหมือนกัน งานวิทยานิพนธ์นี้จึงรวมเอาการกระทำที่มีการเปลี่ยนแปลงค่าคุณสมบัติเดียวกันและมีความหมายของการกระทำไปในทางเดียวกันไว้ในรูปของฟังก์ชั่นและพารามิเตอร์เพื่อลดความซ้ำซ้อน เช่น การเดินในทิศทางต่าง ๆ ตามตัวอย่างข้างต้น ถูกรวมเป็นฟังก์ชั่น RunStraight( Running direction, Running speed) ซึ่งสามารถรับทิศทางและความเร็วในการเดินได้

* + - 1. การทดสอบเงื่อนไข (Condition testing) และการสอบถามข้อมูล (Information query)

การกระทำที่พบในเกมที่นำมาวิเคราะห์จำนวนหนึ่งไม่ได้เกิดขึ้นทันทีทันใด แต่เกิดขึ้นเมื่อสถานะของโลกของเกมหรือค่าคุณสมบัติบางอย่างของภาคีถูกต้องตามเงื่อนไขบางอย่าง เช่น ศัตรู Flyer ใน Metroid จะบินขึ้นในแนวดิ่งจนกระทั่งอยู่ในแนวระดับเดียวกับตัวละครอวตาร จึงเปลี่ยนทิศการเคลื่อนที่เป็นแนวราบแทน ตัวอย่างนี้แสดงให้เห็นถึงการทดสอบเงื่อนไขของคุณสมบัติตำแหน่งในแกน y ระหว่างศัตรูและตัวละครอวตาร ในบางครั้ง เงื่อนไขสำหรับการกระทำอาจจะซับซ้อนกว่าการทดสอบระหว่างค่าคุณสมบัติเพียงค่าเดียว เช่น ศัตรู Bright man ใน Megaman 4 จะโจมตีด้วยท่าพิเศษเมื่อค่าพลังชีวิตของศัตรูมีค่าตรงตามที่กำหนดและตัวศัตรูยืนอยู่บนพื้น

ในที่นี้วิเคราะห์รูปแบบของเงื่อนไขที่พบและข้อมูลที่ใช้เป็นเงื่อนไขในเกมที่กำหนด พบว่าโดยส่วนใหญ่ ลักษณะของเงื่อนไขจะเป็นดังต่อไปนี้

* การเปรียบเทียบระหว่างข้อมูล 2 ข้อมูล ซึ่งเป็นข้อมูลเดียวกันที่ได้จากภาคีที่แตกต่างกัน
* การเปรียบเทียบระหว่างข้อมูลและค่าคงที่
* การตรวจสอบว่าภาคีที่กำหนดอยู่ในสถานะที่สนใจหรือไม่
* อื่น ๆ ที่ไม่เข้าพวก เช่น การตรวจสอบว่าผู้เล่นกดปุ่มที่กำหนดอยู่หรือไม่ เป็นต้น

จากรูปแบบของเงื่อนไข งานวิทยานิพนธ์นี้จึงกำหนดให้มีวิธีการสำหรับการสร้างเงื่อนไขเพื่อทดสอบและกำหนดพฤติกรรมตามแต่เงื่อนไขให้กับศัตรูโดยใช้ตรรกะแบบถ้า-เงื่อนไข-แล้ว ซึ่งเทียบได้กับโครงสร้างควบคุม If-Then ในภาษาโปรแกรมมิ่งทั่วไป ด้วยลักษณะของตรรกะดังกล่าว เงื่อนไขในที่นี้จึงต้องเป็นนิพจน์แบบบูล (Boolean expression) ส่วนเงื่อนไขสามารถสร้างขึ้นจากการนำข้อมูลมาเชื่อมต่อกันด้วยตัวดำเนินการต่าง ๆ เมื่ออ้างอิงรูปแบบของเงื่อนไขข้างต้น แสดงให้เห็นความจำเป็นของตัวดำเนินการเปรียบเทียบ (Comparator) ในกรณีที่เงื่อนไขมีความซับซ้อนขึ้น ต้องมีการเชื่อมระหว่างเงื่อนไขย่อย ๆ หลายเงื่อนไข ในที่นี้จึงต้องมีตัวดำเนินการเชิงตรรกะ (Logical operator)

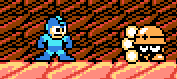
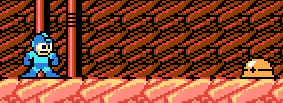
สิ่งสำคัญในการสร้างเงื่อนไขคือข้อมูล ข้อมูลในที่นี้อาจจะเป็นค่าคุณสมบัติของภาคีหรือสถานะของโลกของเกมก็ได้ รวมถึงข้อมูลพิเศษ เช่น การกดปุ่มของผู้เล่น เป็นต้น การเก็บข้อมูลของข้อมูลที่ถูกใช้เป็นเงื่อนไขกระทำในลักษณะเดียวกับการกระทำ นั่นคือ จดบันทึกลักษณะของข้อมูลที่ถูกใช้งานแล้วรวมข้อมูลที่มีลักษณะคล้ายคลึงกันให้อยู่ในรูปของฟังก์ชั่น เช่น ข้อมูลสถานะกำแพงอยู่ด้านหน้าหรือไม่และภาคีอยู่บนพื้นหรือไม่ จะถูกรวมเป็นฟังก์ชั่นที่มีพารามิเตอร์ SurfaceInDirection(Direction) ซึ่งคืนค่าตรรกะที่บอกว่ามีโครงสร้างพื้นที่ที่ผ่านไม่ได้อยู่ในทิศที่กำหนดหรือไม่ ฟังก์ชั่นข้อมูลเหล่านี้จะแตกต่างจากฟังก์ชั่นการกระทำที่ความสามารถในการให้ข้อมูล การสอบถามข้อมูลเพื่อนำไปใช้งานในเงื่อนไขหรือใช้งานเป็นพารามิเตอร์ของฟังก์ชั่นอื่น ๆ จึงได้จากการเรียกใช้งานฟังก์ชั่นข้อมูล

* + - 1. ลำดับพฤติกรรม (Behavior sequence)

ในเกมที่นำมาวิเคราะห์รวมถึงเกมแอ็คชั่นโดยทั่วไป ศัตรูไม่ได้มีการกระทำเพียงแบบเดียว ผู้เล่นสามารถรับรู้ได้จากความเปลี่ยนแปลงของการกระทำที่เปลี่ยนไปตามเวลา มีการเปลี่ยนแปลงจากการกระทำหนึ่งไปยังอีกแบบหนึ่ง เช่น ศัตรู Pakatto24 ซึ่งเป็นศัตรูแบบป้อมปืนใน Megaman 4 จะมีการกระทำเป็นลำดับต่อเนื่อง คือ เปิดเกราะ, ยิง, หยุดรอ, ซ่อนกลับเข้าเกราะ, หยุดรอ อีกทั้งในศัตรูบางตัวอาจจะมีเงื่อนไขของการกระทำเพื่อเลือกรูปแบบการกระทำได้ด้วย นั่นแสดงให้เห็นว่าพฤติกรรมของศัตรูมีลักษณะเป็นลำดับของการกระทำรวมถึงอาจมีการทดสอบเงื่อนไขและฟังก์ชั่นสอบถามข้อมูลเพื่อสร้างทางเลือกของการกระทำที่เป็นไปได้ ในที่นี้เรียกลำดับดังกล่าวว่า ลำดับพฤติกรรม

* + - 1. สถานะของพฤติกรรม (Behavior state) และการเปลี่ยนสถานะ (State transition)

ศัตรูส่วนหนึ่งในเกมที่วิเคราะห์มีลำดับพฤติกรรมหลายแบบขึ้นอยู่กับเงื่อนไข เช่น ศัตรู Metall ใน Megaman 4 จะซ่อนอยู่ในหมวกเหล็กและคอยหันเข้าหาตัวละครอวตาร จนกระทั่งตัวละครอวตารเข้าใกล้ศัตรูระดับหนึ่งจึงออกมาจากหมวกเหล็ก, ยิงกระสุนโจมตี, วิ่งเข้าใส่ แล้วจึงกลับไปซ่อนในหมวกเหล็กเพื่อรอตัวละครอวตารกลับเข้ามาในระยะอีกครั้ง ในตัวอย่างนี้ แสดงให้เห็นถึงลำดับพฤติกรรม 2 ลำดับ ได้แก่ ลำดับพฤติกรรมที่ประกอบด้วยการกระทำหันหน้าเข้าหาผู้เล่นเพียงอย่างเดียว และลำดับพฤติกรรมที่ประกอบด้วยการโจมตีและการเคลื่อนที่ โดยลำดับพฤติกรรมแรกมีระยะห่างจากผู้เล่นเป็นเงื่อนไขในการเปลี่ยนไปใช้งานลำดับพฤติกรรมที่สอง ในขณะที่ลำดับพฤติกรรมที่สองมีการจบลำดับเป็นเงื่อนไขในการเปลี่ยนกลับไปยังลำดับพฤติกรรมแรก การเปลี่ยนแปลงเช่นนี้มีความคล้ายคลึงกับหลักการของเครื่องสถานะจำกัด โดยมองว่าแต่ละลำดับพฤติกรรมของศัตรูเป็นสถานะหนึ่ง และเงื่อนไขการเปลี่ยนแปลงลำดับพฤติกรรมเป็นเงื่อนไขการเปลี่ยนสถานะ รูปที่ 4.3 แสดงตัวอย่างการเปลี่ยนลำดับพฤติกรรมด้วยหลักการของเครื่องสถานะจำกัด



การเปลี่ยนสถานะ 2 🡪 1:

จบลำดับพฤติกรรม

การเปลี่ยนสถานะ 1 🡪 2: เข้าใกล้

สถานะที่ 1:

ลำดับพฤติกรรมหันเข้าหาผู้เล่น

สถานะที่ 2:

ลำดับพฤติกรรมโจมตีแล้ววิ่ง

รูปที่ 4.3 การเปลี่ยนลำดับพฤติกรรมแสดงด้วยหลักการของเครื่องสถานะจำกัด

จากความคล้ายคลึงของหลักการข้างต้น สถานะและการเปลี่ยนสถานะจึงเป็นอีกองค์ประกอบหนึ่งของศัตรูที่ใช้ระบุความเปลี่ยนแปลงของลำดับพฤติกรรมในกรณีที่ศัตรูมีลำดับพฤติกรรมหลายลำดับ โดยศัตรูจะต้องมีการระบุตัวเลขสถานะให้กับทุกลำดับพฤติกรรม และสำหรับทุกลำดับพฤติกรรมจะต้องมีเซ็ตอธิบายการเปลี่ยนสถานะซึ่งเป็น เซ็ตของ 2-tuple  เมื่อ  คือเงื่อนไขการเปลี่ยนสถานะ และ  คือตัวเลขสถานะปลายทาง

* + - 1. ลำดับพฤติกรรมคู่ขนาน (Parallel sequence)

นอกจากศัตรูจะมีได้หลายลำดับพฤติกรรมแล้ว ในบางครั้งศัตรูอาจกระทำหลายลำดับพฤติกรรมในเวลาเดียวกันได้ เช่น ศัตรู Sniper ใน Super C ซึ่งสามารถยิงปืนโจมตีผู้เล่นทุกช่วงเวลาที่กำหนด ลำดับพฤติกรรมในที่นี้ คือ ยิง, หยุดรอ แต่ในขณะเดียวกัน ศัตรูตัวดังกล่าวนี้ยังหันปืนเข้าหาผู้เล่นอีกด้วย แสดงให้เห็นถึงลำดับพฤติกรรมหันเข้าหาผู้เล่นที่กระทำไปพร้อมกับลำดับพฤติกรรมยิงแล้วหยุดรอ นั่นหมายถึงในหนึ่งสถานะจะไม่ได้มีเพียงลำดับพฤติกรรมเดียวที่กระทำอยู่ แต่อาจมีหลายลำดับพฤติกรรมได้ ด้วยแนวคิดดังกล่าวนี้ จึงต้องมีการปรับปรุงองค์ประกอบสถานะของศัตรู โดยให้แต่ละสถานะประกอบด้วยตัวเลขสถานะและเซ็ตของลำดับพฤติกรรม

* + 1. รายละเอียดแบบจำลองภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร

หัวข้อนี้จะแสดงรายละเอียดของแบบจำลองศัตรูซึ่งสร้างขึ้นจากองค์ประกอบของศัตรูที่อธิบายไว้ในหัวข้อก่อนหน้า แบบจำลองดังกล่าวนี้จะถูกปรับปรุงเพื่อลดความซ้ำซ้อนลงรวมถึงเพื่อให้ครอบคลุมกระสุน แบบจำลองที่ผ่านการปรับปรุงแล้วจะถูกเรียกว่าแบบจำลองภาคีฯเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร ซึ่งจะถูกนำเสนอในรูปของภาษาอธิบายภาคีฯที่ตัวแปลภาษาสามารถทำความเข้าใจได้

* + - 1. แบบจำลองศัตรู

ศัตรูประกอบด้วยองค์ประกอบต่าง ๆ ดังที่ได้กล่าวถึงในหัวข้อ 4.1.3 แบบจำลองศัตรูจะถูกสร้างขึ้นด้วยองค์ประกอบเหล่านี้ ดังนั้นแบบจำลองศัตรูจึงประกอบด้วย

* ค่าคุณสมบัติ โดยบางคุณสมบัตินั้นเป็นคุณสมบัติที่ศัตรูจะต้องมีโดยค่าจะแตกต่างกันไปในศัตรูแต่ละตัว ค่าคุณสมบัติเหล่านี้จะขาดไปไม่ได้ มิเช่นนั้นศัตรูจะไม่สามารถอยู่ในโลกของเกมได้ สำหรับบางคุณสมบัติจะเป็นคุณสมบัติทางเลือก อาจไม่มีก็ได้ นอกเหนือจากค่าคุณสมบัติที่ผ่านการวิเคราะห์แล้ว เนื่องจากการเคลื่อนที่ของศัตรูในเกมที่วิเคราะห์มีลักษณะเป็นแบบไปข้างหน้า เพื่อเป็นการลดทอนจำนวนพารามิเตอร์ของฟังก์ชั่นการกระทำบางส่วน ในที่นี้จึงเพิ่มค่าคุณสมบัติ “ทิศทาง” (Direction) ให้กับแบบจำลองด้วย ค่าคุณสมบัติจะเป็นไปดังตาราง 4.2

|  |  |
| --- | --- |
| ตาราง 4.2 ค่าคุณสมบัติสำหรับแบบจำลองศัตรู | |
| ค่าคุณสมบัติจำเป็น | ค่าคุณสมบัติทางเลือก |
| ตัวตรวจจับการชน (ซึ่งประกอบด้วยขนาดและตำแหน่ง), ผลของแรงดึงดูด, พลังชีวิต, พลังโจมตี, กลุ่ม, ทิศทาง | ผ่านได้, คงกระพัน, ฝ่ายโจมตี, ฝ่ายตั้งรับ |

* เซ็ตของสถานะ โดยแต่ละสถานะจะประกอบด้วยข้อมูล 3 ชนิด ได้แก่
  + หมายเลขสถานะ ซึ่งจะต้องไม่ซ้ำกันสำหรับทุกสถานะของศัตรูนั้น ๆ โดยศัตรูจะต้องมีสถานะเริ่มต้นเสมอ ในที่นี้ให้สถานะหมายเลข 0 เป็นสถานะเริ่มต้นของศัตรู
  + เซ็ตของลำดับพฤติกรรมที่แสดงถึงลำดับพฤติกรรมทั้งหมดที่กระทำคู่ขนานกันในสถานะหนึ่ง
  + เซ็ตของการเปลี่ยนสถานะซึ่งเป็นเซ็ตของ 2-tuple ที่ประกอบด้วยเงื่อนไขและสถานะปลายทาง
* ลำดับพฤติกรรมซึ่งเป็นส่วนหนึ่งของสถานะ จะประกอบด้วยฟังก์ชั่นการกระทำ ฟังก์ชั่นข้อมูล และโครงสร้างทดสอบเงื่อนไข
  + โครงสร้างทดสอบเงื่อนไขเป็นข้อมูลในรูปแบบถ้า-เงื่อนไข-แล้ว แต่เพื่อสร้างทางเลือกเพิ่มเติมในกรณีที่เงื่อนไขไม่เป็นจริง จึงต้องปรับเปลี่ยนโครงสร้างเป็น ถ้า-เงื่อนไข-แล้ว-ถ้าไม่ใช่-แล้ว ซึ่งเทียบได้กับโครงสร้างแบบ If-Then-Else ในภาษาโปรแกรมมิ่ง
  + เงื่อนไข เป็นนิพจน์แบบบูลซึ่งประกอบขึ้นจากนิพจน์แบบบูล ตัวดำเนินการเปรียบเทียบ และตัวดำเนินการตรรกะ
    - 1. การปรับปรุงแบบจำลอง

การปรับปรุงแบบจำลองจะแบ่งออกเป็น 2 ส่วน ได้แก่ การละเซ็ตของการเปลี่ยนสถานะ และการรวมกระสุนเข้ากับแบบจำลอง

เซ็ตของการเปลี่ยนสถานะที่กล่าวถึงในหัวข้อ 4.1.3.5 สามารถละได้โดยเปลี่ยนไปใช้ฟังก์ชั่นการกระทำพิเศษ Goto( Target state ) เพื่อลดความซับซ้อนของแบบจำลองและทำให้ควบคุมการเปลี่ยนสถานะได้ดียิ่งขึ้น

ฟังก์ชั่น Goto เป็นฟังก์ชั่นที่สั่งให้ศัตรูเปลี่ยนสถานะไปเป็นสถานะที่กำหนดและเริ่มกระทำลำดับพฤติกรรมทั้งหมดในสถานะปลายทาง สาเหตุที่สามารถใช้ฟังก์ชั่น Goto แทนเซ็ตของการเปลี่ยนสถานะได้ เนื่องมาจากวิธีการทำงานของเซ็ตของการเปลี่ยนสถานะ ซึ่งจะถูกตรวจสอบตลอดเวลาไปพร้อมกับการทำลำดับพฤติกรรมของสถานะปัจจุบัน เมื่อพบ 2-tuple ที่เงื่อนไขเป็นจริงจึงทำการเปลี่ยนไปยังสถานะปลายทางที่กำหนดใน 2-tuple นั้น ลักษณะการทำงานดังกล่าวสามารถเปลี่ยนรูปแบบไปเป็นลำดับพฤติกรรมคู่ขนานอีกหนึ่งลำดับที่ประกอบด้วยเงื่อนไขและฟังก์ชั่นการกระทำ Goto ที่มีพารามิเตอร์เป็นสถานะปลายทาง สำหรับกรณีเงื่อนไขเป็นจริง นอกจากนี้การใช้งานในรูปฟังก์ชั่นการกระทำ Goto จะทำให้ควบคุมการเปลี่ยนสถานะได้ดียิ่งขึ้น โดยสามารถแทรกฟังก์ชั่นดังกล่าวระหว่างการกระทำอื่น ๆ ทำให้สามารถยืนยันได้ว่าสถานะจะถูกเปลี่ยนโดยไม่ทำการกระทำถัดไปได้ ซึ่งในกรณีที่ใช้งานเซ็ตของการเปลี่ยนสถานะจะไม่สามารถควบคุมพฤติกรรมเช่นนี้ได้

การรวมกระสุนเข้ากับแบบจำลองศัตรูจะทำได้ก็ต่อเมื่อสามารถแสดงให้เห็นว่าสามารถอธิบายกระสุนได้ครบถ้วนด้วยแบบจำลองศัตรู กระสุนอยู่ในโลกของเกมและสามารถชนกับภาคีอื่น ๆ เพื่อสร้างความเสียหายได้ จึงต้องมีตัวตรวจจับการชนซึ่งได้กล่าวถึงแล้วในหัวข้อ 4.1.3.1 และเมื่อกล่าวถึงความเสียหาย ย่อมเกี่ยวข้องกับกลไกการต่อสู้ซึ่งมีค่าคุณสมบัติที่สำคัญได้แก่ ฝ่ายโจมตี, พลังโจมตี และกลุ่ม นอกจากนี้ กระสุนบางชนิดของศัตรูจะสามารถทำลายได้โดยใช้การโจมตีของตัวละครอวตาร แสดงให้เห็นว่า กระสุนสามารถมีค่าคุณสมบัติฝ่ายป้องกันและพลังชีวิตได้ด้วย

อีกกลไกหนึ่งที่น่าสนใจสำหรับกระสุนคือการสลายตัว กระสุนในเกมที่วิเคราะห์บางชนิดจะสลายตัวเมื่อกระทบถูกเป้าหมายแล้ว ในขณะที่บางชนิดจะไม่สลายตัวไปหรือสลายตัวไปหลังจากกระทบเป้าหมายหลายครั้ง ในที่นี้จึงนำค่าคุณสมบัติพลังชีวิตและคงกระพันมาใช้จัดการกับกลไกเช่นนี้ โดยกำหนดว่า หากกระสุนกระทบกับเป้าหมายแล้ว จะเสียพลังชีวิตลง 1 หน่วยในกรณีที่ไม่มีค่าคุณสมบัติคงกระพัน และเมื่อพลังชีวิตหมดกระสุนจะหายไป เนื่องจากกลไกการสลายตัวของกระสุนซึ่งกล่าวถึงการสลายตัวหลังทำการโจมตีเป็นจำนวนครั้งที่กำหนดเป็นกลไกพิเศษที่ไม่พบในศัตรู จึงต้องเพิ่มค่าคุณสมบัติพิเศษ “ภาคีกระสุน” (Projectile) ให้กับแบบจำลองศัตรูหากต้องการรวมภาคีกระสุนเข้ากับแบบจำลอง และให้กลไกการสลายตัวเกิดขึ้นกับภาคีที่มีค่าคุณสมบัติกระสุนเท่านั้น

สำหรับค่าคุณสมบัติอื่น ๆ ได้แก่ ผลแรงดึงดูด, ทิศทาง และผ่านได้ กระสุนจำเป็นต้องมีค่าคุณสมบัติผลแรงดึงดูดเนื่องจากกระสุนที่พบในเกมที่วิเคราะห์มีทั้งกระสุนที่ลอยได้และกระสุนที่ตกลงตามแรงโน้มถ่วง สำหรับค่าคุณสมบัติผ่านได้ เนื่องจากกระสุนส่วนใหญ่สามารถทะลุผ่านโครงสร้างพื้นที่ที่ผ่านไม่ได้ ยกเว้นแต่กระสุนในเกม Super C ที่กระสุนทุกชนิดไม่สามารถทะลุผ่านโครงสร้างพื้นที่ที่ผ่านไม่ได้เช่นเดียวกับศัตรู ซึ่งน่าจะเป็นความตั้งใจของการออกแบบเกมมากกว่าการทำให้เกิดความแตกต่างของพฤติกรรม ในที่นี้จึงถือว่ากระสุนมีคุณสมบัติผ่านได้อยู่เสมอ ส่วนค่าคุณสมบัติทิศทางนั้นเป็นค่าคุณสมบัติที่เกี่ยวข้องกับการลดทอนความซับซ้อนของพฤติกรรม หากพฤติกรรมของกระสุนอธิบายได้ด้วยแบบจำลองศัตรู กระสุนก็ต้องมีค่าคุณสมบัติทิศทางเช่นกัน ซึ่งกระสุนส่วนใหญ่มีพฤติกรรมที่เรียบง่าย ไม่ได้มีลำดับพฤติกรรมที่ซับซ้อนเช่นเดียวกับศัตรู ส่งผลให้แบบจำลองศัตรูที่มีความซับซ้อนมากกว่าสามารถครอบคลุมองค์ประกอบทางด้านพฤติกรรมของกระสุนได้

ด้วยเหตุผลทั้งหมดที่กล่าวมา งานวิทยานิพนธ์นี้จึงรวมกระสุนซึ่งนับได้ว่าเป็นภาคีอีกประเภทหนึ่งเข้ากับแบบจำลองของศัตรู และเรียกแบบจำลองซึ่งมีค่าคุณสมบัติภาคีกระสุนเพิ่มเติมขึ้นมาว่า “แบบจำลองภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร”

* + - 1. ภาษาอธิบายภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร

เพื่อให้แบบจำลองภาคีฯมีรูปแบบที่ชัดเจน มนุษย์สามารถเข้าใจได้ง่าย และสามารถแปลด้วยตัวแปลภาษาได้ งานวิทยานิพนธ์นี้จึงทำให้แบบจำลองภาคีฯอยู่ในรูปของภาษาอธิบายภาคีเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร

ภาษาอธิบายภาคีฯเป็นภาษาที่มีชนิดข้อมูล ชนิดข้อมูลที่นำมาใช้งานจะต้องตรงกับชนิดข้อมูลที่ฟังก์ชั่นหรือโครงสร้างเงื่อนไขต้องการ โครงสร้างของภาษาอธิบายภาคีฯจะเป็นรูปแบบบล็อก กล่าวคือ องค์ประกอบต่าง ๆ ของภาษาจะถูกห่อหุ้มด้วยสัญลักษณ์บล็อก ซึ่งในที่นี้ใช้เครื่องหมายวงเล็บปีกกาเปิด “{“ สำหรับจุดเริ่มต้นบล็อก และวงเล็บปีกกาปิด “}” สำหรับจุดสิ้นสุดบล็อก สิ่งที่อยู่ระหว่างบล็อกใด ๆ เรียกว่า เนื้อหาของบล็อก (Block content) ภาพรวมของภาษาอธิบายภาคีฯแสดงได้ด้วยไวยากรณ์ดังรูปที่ 4.4

|  |
| --- |
| Agent 🡪 IDENTIFIER { Initializer StateGroup Destructor }  Initializer 🡪 .init { Sequence } | ^  Destructor 🡪 .des { Sequence } | ^  StateGroup 🡪 StateGroup StateGroup | State  State 🡪 IDENTIFIER { Parallel }  Parallel 🡪 Parallel Parallel | SequenceBlock | ^  SequenceBlock 🡪 IDENTIFIER { Sequence }  Sequence 🡪 Sequence Sequence | CondStruct | ACTION ;  CondStruct 🡪 IfBlock | IfBlock ElseBlock  IfBlock 🡪 if( BOOLEANEXP ) { Sequence }  ElseBlock 🡪 else IfBlock | else { Sequence } |
| รูปที่ 4.4 ไวยากรณ์ของภาษาอธิบายภาคีฯ |

รายละเอียดของภาคีทั้งหมดจะถูกห่อหุ้มอยู่ในบล็อกที่เรียกว่า “บล็อกภาคี” (Agent block) ซึ่งประกอบด้วยตัวระบุ (Identifier) ตามสัญลักษณ์บล็อก ภายในบล็อกนี้จะประกอบด้วย “บล็อกเริ่มต้น” (Initializer block), “บล็อกทำลาย” (Destructor block) และ “กลุ่มบล็อกสถานะ” (State group)

บล็อกเริ่มต้นเป็นบล็อกที่มีตัวระบุบังคับชื่อ “.init“ บล็อกภาคีจะมีบล็อกเริ่มต้นหรือไม่ก็ได้ เนื้อหาของบล็อกเป็นลำดับพฤติกรรมที่ภาคีจะกระทำเมื่อภาคีเข้าสู่โลกของเกม โดยปกติลำดับพฤติกรรมภายในบล็อกนี้จะเป็นการกำหนดค่าคุณสมบัติต่าง ๆ ให้กับภาคี

บล็อกทำลายเป็นบล็อกที่มีตัวระบุบังคับชื่อ “.des” บล็อกภาคีจะมีบล็อกทำลายหรือไม่ก็ได้ เนื้อหาของบล็อกเป็นลำดับพฤติกรรมที่ภาคีจะกระทำเมื่อถูกนำออกจากโลกไปแล้ว โดยปกติจะเป็นการกระทำที่ไม่เปลี่ยนแปลงค่าคุณสมบัติของตัวเอง แต่เปลี่ยนแปลงค่าคุณสมบัติของภาคีอื่น ๆ หรือสถานะของโลกของเกม

กลุ่มบล็อกสถานะประกอบด้วย “บล็อกสถานะ” (State block) จำนวนอย่างน้อย 1 บล็อก กลุ่มบล็อกสถานะใช้อธิบายพฤติกรรมหลักของภาคี หลังจากที่ภาคีกระทำลำดับพฤติกรรมในบล็อกเริ่มต้นเสร็จสิ้น ภาคีจะเข้าสู่สถานะแรกสุดที่ปรากฏในกลุ่มสถานะ และกระทำตามพฤติกรรมที่ระบุไว้ในบล็อกสถานะนั้น

บล็อกสถานะเป็นบล็อกที่มีตัวระบุชื่อใด ๆ ที่ไม่ซ้ำกับบล็อกสถานะอื่น ๆ ภายในบล็อกภาคีเดียวกัน เนื้อหาของบล็อกสถานะประกอบด้วย “บล็อกลำดับ” (Sequence block) จำนวนกี่บล็อกก็ได้ แต่ละบล็อกลำดับเป็นตัวแทนของ 1 ลำดับพฤติกรรม ทุกบล็อกลำดับภายในบล็อกสถานะจะอธิบายลำดับพฤติกรรมที่คู่ขนานกัน บล็อกลำดับภายในบล็อกสถานะเดียวกันจะต้องมีตัวระบุชื่อไม่ซ้ำกัน โดยเนื้อหาของแต่ละบล็อกลำดับคือลำดับพฤติกรรม

ค่าคุณสมบัติจะไม่มีการระบุไว้ในภาษาอธิบายภาคีฯ แต่ตัวแปลภาษาจะกำหนดค่าคุณสมบัติเริ่มต้นให้กับภาคี ซึ่งสามารถเปลี่ยนแปลงได้ตลอดเวลา การตั้งค่าคุณสมบัติเริ่มต้นเฉพาะสำหรับภาคีจะกระทำในบล็อกเริ่มต้นดังที่ได้กล่าวไว้แล้ว สำหรับรายละเอียดปลีกย่อยอื่น ๆ ของภาษาอธิบายภาคีฯเป็นดังต่อไปนี้

สัญลักษณ์สิ้นสุด (Terminal symbol) ที่ปรากฏในไวยากรณ์

* IDENTIFIER เป็นตัวระบุ ใช้เป็นชื่อของบล็อกต่าง ๆ ซึ่งมีกฎการเลือกชื่อดังที่ได้กล่าวไว้แล้ว โดยการเขียนตัวระบุจะต้องขึ้นต้นด้วยเครื่องหมายมหัพภาค “.” ตามด้วยพยัญชนะภาษาอังกฤษ ตัวเลข หรือเส้นใต้อักขระ “\_”
* ACTION เป็นฟังก์ชั่นการกระทำที่ไม่มีการคืนค่าซึ่งได้จากองค์ประกอบการกระทำของศัตรู พารามิเตอร์ของฟังก์ชั่นที่ได้จากการวิเคราะห์ในหัวข้อ 4.1.3.2 จะถูกกำหนดชนิดข้อมูลให้และใช้เป็นฟังก์ชั่นการกระทำ ค่าพารามิเตอร์ของฟังก์ชั่นอาจเป็นสัญพจน์ที่มีชนิดข้อมูลตรงกันหรือฟังก์ชั่นข้อมูลที่คืนค่าเป็นชนิดข้อมูลที่กำหนด โดยฟังก์ชั่นข้อมูลในที่นี้ได้จากการวิเคราะห์ในหัวข้อ 4.1.3.3 โดยมีการกำหนดชนิดข้อมูลให้กับพารามิเตอร์และค่าที่คืนกลับของฟังก์ชั่นดังกล่าว
* BOOLEANEXP เป็นนิพจน์แบบบูลซึ่งได้จากการนำนิพจน์แบบบูลหรือสัญพจน์แบบบูลมาเชื่อมกันด้วยตัวดำเนินการตรรกะ หรือได้จากการเปรียบเทียบระหว่างข้อมูลชนิดต่าง ๆ ด้วยตัวเปรียบเทียบ หรือได้จากการเรียกใช้ฟังก์ชั่นข้อมูลที่คืนค่าเป็นค่าแบบบูล

ชนิดข้อมูล

ตัวแปลภาษาจะใช้ชนิดข้อมูลในการตรวจสอบความเข้ากันระหว่างฟังก์ชั่นและอากิวเมนท์ที่ส่งเข้ามา โดยชนิดข้อมูลบางชนิดจะมีสัญพจน์ของตนเองดังตัวอย่างในตารางที่ 4.3 ในขณะที่ชนิดข้อมูลอีกหลายชนิดจะไม่มีสัญพจน์แต่ได้จากการเรียกใช้ฟังก์ชั่นที่คืนค่าที่มีชนิดข้อมูลดังกล่าว เช่น ชนิดข้อมูลตัวอ้างอิงภาคีไม่สามารถเขียนด้วยสัญพจน์ได้ แต่สามารถใช้งานได้จากฟังก์ชั่น เช่น DynamicFilter ซึ่งเป็นฟังก์ชั่นที่คืนค่าเป็นชนิดข้อมูลตัวอ้างอิงภาคีเป็นต้น

|  |  |  |
| --- | --- | --- |
| ตารางที่ 4.3 ตัวอย่างสัญพจน์ | | |
| ชนิดข้อมูล | สัญพจน์ | ความหมาย |
| Boolean | true, false | ค่าตรระกะจริง, เท็จ ตามลำดับ |
| Decimal | 1245, 13.02 | ตัวเลขจำนวนจริง |
| Direction | “North” | ทิศทางที่ชี้ไปทิศเหนือ หรือเวกเตอร์ 2 มิติ (0,1) |
| “230” | ทิศทางที่ได้จากการหมุนเวกเตอร์ 2 มิติ (1,0) 230 องศารอบจุด (0,0) ทวนเข็มนาฬิกา |
| Position | “c(120,150)” | ตำแหน่งในระบบพิกัดคาร์ทีเซียนที่มีตำแหน่ง x = 120 และ y = 150 |
| “p(50,180)” | ตำแหน่งในระบบพิกัดเชิงขั้วที่มีรัศมีเป็น 50 พิกเซล และมุม 180 องศา |

ตัวเปรียบเทียบ

ตัวเปรียบเทียบใช้สำหรับสร้างนิพจน์แบบบูลจากข้อมูล 2 ตัวโดยใช้การเปรียบเทียบ สามารถใช้งานได้โดยการเขียนตัวเปรียบเทียบไว้ระหว่างข้อมูล โดยข้อมูลดังกล่าวต้องเป็นนิพจน์ที่นำมาเปรียบเทียบได้ ซึ่งในที่นี้ข้อมูลที่เป็นตัวเลขจะสามารถนำมาเปรียบเทียบกันด้วยตัวเปรียบเทียบทุกแบบ ได้แก่ “==”, “!=”, “>”, “<”, “>=”, “<=” ในขณะที่ข้อมูลชนิดอื่น ๆ สามารถเปรียบเทียบความเท่ากันด้วยตัวเปรียบเทียบ “==” และ “!=” เท่านั้น

ตัวดำเนินการ

ตัวดำเนินการของภาษาอธิบายภาคีฯแบ่งเป็น ตัวดำเนินการเชิงตรรกะ และตัวดำเนินการคำนวณ ตัวดำเนินการเชิงตรรกะใช้สำหรับดำเนินการกับนิพจน์แบบบูล โดยแบ่งเป็นตัวดำเนินการทวิภาคสำหรับเชื่อม 2 นิพจน์แบบบูลเข้าด้วยกัน ประกอบด้วยตัวดำเนินการ “&&” และ “||” และตัวดำเนินการเอกภาค “!” สำหรับเปลี่ยนผลลัพธ์ของนิพจน์เป็นค่าตรงข้าม ส่วนตัวดำเนินการคำนวณประกอบด้วย “+”, “-“, “\*”, “/” และ “%”

โครงสร้างเงื่อนไข

ในที่นี้ใช้โครงสร้างแบบ If-Else ดังรูปที่ 4.5 เหมือนในภาษาโปรแกรมมิ่งทั่วไป โดยเงื่อนไขต้องเป็นนิพจน์แบบบูลเท่านั้น และเนื้อหาภายในของบล็อก If และบล็อก Else เป็นลำดับอธิบายภาคี โดยบล็อก Else อาจละได้ในกรณีที่ไม่ต้องการลำดับพฤติกรรมสำหรับกรณีที่เงื่อนไขไม่เป็นจริง

|  |  |  |
| --- | --- | --- |
|  | if(Random(DecimalSet(1,2,1)) == 1){  Goto(.feather);  }else{ Goto(.tomahawk);  } |  |
| รูปที่ 4.5 ตัวอย่างโครงสร้างเงื่อนไข | | |

ซิงเกิ้ลคิวรี (Single query)

เขียนด้วยสัญลักษณ์ “$” สำหรับใช้ต่อท้ายฟังก์ชั่นข้อมูลที่ใช้งานภายในฟังก์ชั่นการกระทำระยะยาว (Spanned action) เพื่อให้ข้อมูลถูกดึงมาครั้งเดียวเมื่อเริ่มการกระทำ รายละเอียดของซิงเกิ้ลคิวรีจะกล่าวถึงในหัวข้อถัดไป

ตัวอย่างของภาคีที่อธิบายด้วยภาษาอธิบายภาคีฯเป็นไปดังรูปที่ 4.6 ซึ่งในบล็อกเริ่มต้นของภาคีศัตรู Watton จาก Megaman 4 จะเห็นตัวอย่างของการกำหนดค่าคุณสมบัติต่าง ๆ ด้วยฟังก์ชั่น Set โดยค่าคุณสมบัติ “texture” เป็นค่าคุณสมบัติเพิ่มเติมที่ทำการกำหนดรูปให้กับภาคีซึ่งไม่เกี่ยวข้องกับพฤติกรรมภายในขอบเขตของงานวิทยานิพนธ์นี้ นอกจากนี้ยังมีฟังก์ชั่นที่น่าสนใจ เช่น Spawn ซึ่งใช้สำหรับเพิ่มภาคีใหม่ลงในโลกของเกมด้วย

|  |
| --- |
| .Watton{  .init{  Set("texture", DynamicFilter("this"), 4);  Set("position", DynamicFilter("this"), "c(400,200)");  Set("direction", DynamicFilter("this"), TurnToPlayer(DirectionSet("H")));  Set("collider", DynamicFilter("this"), "48,48");  Set("gravityeff", DynamicFilter("this"), 0);  Set("phasing", DynamicFilter("this"), true);  Set("hp", DynamicFilter("this"), 10);  Set("attacker", DynamicFilter("this"), true);  Set("atk", DynamicFilter("this"), 10);  Set("defender", DynamicFilter("this"), true);  }  .state0{  .seq0{  RunStraight(Get("direction", DynamicFilter("this")), 2, TimePass() >= 150);  FlipDirection("H");  }  .seq1{  Wait(TimePass() >= 90);  Spawn(.BulletPod, Get("position", DynamicFilter("this")));  }  }  }  .BulletPod{  .init{  Set("texture", DynamicFilter("this"), 3);  Set("collider", DynamicFilter("this"), "24,24");  Set("gravityeff", DynamicFilter("this"), 1);  Set("projectile", DynamicFilter("this"), true);  Set("invul", DynamicFilter("this"), true);  }  .state0{  .seq0{  Jump(Get("position", DynamicFilter("this")), 32, 5, false);  }  .seq1{  if(Abs(DistanceToPlayer("Y")) <= 30){  Despawn();  }  }  }  .des{  Spawn(.Bullet\_Watton, Get("position", DynamicFilter("this")), "0");  Spawn(.Bullet\_Watton, Get("position", DynamicFilter("this")), "-45");  Spawn(.Bullet\_Watton, Get("position", DynamicFilter("this")), "-90");  Spawn(.Bullet\_Watton, Get("position", DynamicFilter("this")), "-135");  Spawn(.Bullet\_Watton, Get("position", DynamicFilter("this")), "-180");  }  }  .Bullet\_Watton{  .init{  Set("texture", DynamicFilter("this"), 6);  Set("collider", DynamicFilter("this"), "24,24");  Set("projectile", DynamicFilter("this"), true);  Set("attacker", DynamicFilter("this"), true);  Set("atk", DynamicFilter("this"), 1);  }  .state0{  .seq0{  RunStraight(Get("direction", DynamicFilter("this")), 3, false);  }  }  } |
| รูปที่ 4.6 ตัวอย่างภาคี Watton |

* + - 1. ตัวแปลภาษา

ตัวแปลภาษาทำหน้าที่แปลภาษาอธิบายภาคีฯให้เป็นพฤติกรรมของภาคีที่ปรากฏในโลกของเกม โดยก่อนที่ภาษาอธิบายภาคีฯจะถูกแปลได้ด้วยตัวแปลภาษา จะต้องถูกแปลงให้อยู่ในรูปของรหัสกลาง (Intermediate code) ด้วยคอมไพเลอร์ก่อน รหัสกลางเป็นรายการของคำสั่งเครื่อง (Instruction) ซึ่งออกแบบมาให้ทำงานได้บนเครื่องกองซ้อน (Stack machine) ซึ่งตัวแปลภาษาที่ทำขึ้นในงานวิทยานิพนธ์นี้ทำงานแบบเครื่องกองซ้อน

ตัวแปลภาษาจะมีตัวนับโปรแกรม (Program counter -- PC) สำหรับชี้ไปยังคำสั่งเครื่องปัจจุบันของรายการคำสั่งเครื่องที่ถูกแปลอยู่ ข้อมูลที่จะถูกดำเนินการจะถูกจัดเก็บบนกองซ้อนเพื่อรอคำสั่งที่จะนำข้อมูลดังกล่าวออกไปดำเนินการต่อไป เมื่อคำสั่งปัจจุบันถูกแปล ตำแหน่งของ PC จะถูกคำนวณใหม่เพื่อให้ชี้ไปยังคำสั่งถัดไป

ในการแปลภาษาอธิบายภาคีฯเป็นพฤติกรรมสำหรับภาคี 1 ตัว ตัวแปลภาษาจะแปลเฉพาะสถานะปัจจุบันของภาคีนั้น ๆ โดยใช้ตัวแปลภาษาจำนวนเท่ากับจำนวนของบล็อกลำดับในสถานะนั้น ตัวแปลภาษาทั้งหมดจะแปลบล็อกลำดับโดยให้ผู้เล่นรู้สึกได้ว่าลำดับพฤติกรรมของแต่ละบล็อกลำดับทำงานพร้อมกัน ซึ่งในที่นี้ไม่ได้ใช้วิธีคำนวณแบบขนาน (Parallel computing) แต่จัดลำดับการแปลสำหรับทุกบล็อกลำดับและแปลบล็อกลำดับครั้งละบล็อกจนครบทั้งหมดให้เสร็จสิ้นภายใน “เฟรม[[1]](#footnote-1)” เดียวของเกม ซึ่งจะทำให้ผู้เล่นรู้สึกว่าลำดับพฤติกรรมทั้งหมดในสถานะทำงานพร้อมกันได้

แต่ละบล็อกลำดับใช้ตัวแปลภาษา 1 ตัวเนื่องจากจำเป็นต้องมี PC สำหรับจดจำคำสั่งเครื่องปัจจุบันของบล็อกลำดับแยกกัน หากตัวแปลภาษาทำงานคำสั่งเครื่องสุดท้ายของบล็อกลำดับเสร็จ PC จะถูกตั้งค่ากลับไปยังจุดเริ่มต้นของบล็อกลำดับ (ตั้งค่า PC เป็น 0) ตัวแปลภาษาจะแปลคำสั่งเครื่องของบล็อกลำดับไปเรื่อย ๆ จนกว่าสถานะของตัวแปลภาษาจะตรงตามเงื่อนไขพัก ตัวแปลภาษาตัวปัจจุบันที่ทำงานอยู่จึงพักชั่วคราวจนถึงเฟรมถัดไปและให้ตัวแปลภาษาสำหรับลำดับพฤติกรรมถัดไปทำงานต่อ

เงื่อนไขพักสำหรับตัวแปลภาษามี 2 เงื่อนไข เงื่อนไขใดเงื่อนไขหนึ่งต้องเป็นจริง ตัวแปลภาษาตัวปัจจุบันจึงหยุดพัก เงื่อนไขที่หนึ่งคือ PC ถูกตั้งค่ากลับไปยังจุดเริ่มต้นของบล็อกลำดับ เงื่อนไขที่สองคือ ตัวแปลภาษาแปลคำสั่งเครื่องปัจจุบันที่เป็นการกระทำระยะยาว (Spanned action) เสร็จ

การกระทำระยะยาวเป็นฟังก์ชั่นการกระทำที่ถูกระบุว่าเป็นการกระทำที่ไม่สามารถทำให้เสร็จสิ้นในเวลาอันสั้นได้ โดยมีจุดที่แตกต่างจากฟังก์ชั่นการกระทำปกติ คือ มีพารามิเตอร์ endCondition ที่รับนิพจน์แบบบูลเพื่อใช้ระบุว่าการกระทำเสร็จสิ้นแล้วหรือไม่ หากการกระทำยังไม่เสร็จสิ้น PC จะไม่เคลื่อนไปยังคำสั่งเครื่องถัดไป นอกจากนี้ ทุกครั้งที่ตัวแปลภาษายังคง PC อยู่ที่การกระทำระยะยาว พารามิเตอร์ทั้งหมดของฟังก์ชั่นจะถูกแปลใหม่เพื่อให้ข้อมูลที่ได้มีความทันสมัย

ทว่าความทันสมัยของข้อมูลสำหรับฟังก์ชั่นการกระทำระยะยาวอาจไม่ได้จำเป็นเสมอไป ในบางครั้งฟังก์ชั่นการกระทำอาจต้องการข้อมูลเพียงครั้งเดียวเมื่อการกระทำเริ่มต้นและใช้ข้อมูลดังกล่าวจนกว่าจะจบการกระทำโดยไม่มีการขอข้อมูลใหม่ เช่น ศัตรู Skull man ใน Megaman 4 จะวิ่งไปยังตำแหน่งที่ผู้เล่นอยู่เมื่อการกระทำเริ่มต้น เป็นต้น ภาษาอธิบายภาคีฯจึงมีสัญลักษณ์ซิงเกิ้ลคิวรีสำหรับต่อท้ายฟังก์ชั่นข้อมูลที่ต้องการให้ตัวแปลภาษาแปลและดึงข้อมูลมาเฉพาะในขณะที่การกระทำเริ่มต้น และใช้ข้อมูลดังกล่าวนี้จนกว่าการกระทำระยะยาวที่มีการใช้งานฟังก์ชั่นข้อมูลนั้นจะจบลง

* 1. การทำเหมืองข้อมูลบนชุดข้อมูลรูปแบบพฤติกรรม

จุดประสงค์ของงานวิทยานิพนธ์นี้คือการสร้างรูปแบบพฤติกรรมของศัตรูที่ใช้งานในเกมได้และเป็นที่ยอมรับของผู้เล่น คำถามสำคัญในการสร้างศัตรูที่เป็นที่ยอมรับของผู้เล่น คือ รูปแบบพฤติกรรมลักษณะใดทำให้ศัตรูเป็นที่ยอมรับของผู้เล่น งานวิทยานิพนธ์นี้หาคำตอบของคำถามดังกล่าวโดยใช้การทำเหมืองข้อมูลวิธีต่าง ๆ บนชุดข้อมูลรูปแบบพฤติกรรมของศัตรูที่ผู้เล่นยอมรับได้เพื่อหาความสัมพันธ์ที่พบบ่อย ความสัมพันธ์ดังกล่าวจะถูกใช้ในการสร้างรูปแบบพฤติกรรมอัตโนมัติต่อไป

* + 1. ชุดข้อมูลรูปแบบพฤติกรรม

ชุดข้อมูลรูปแบบพฤติกรรมที่งานวิทยานิพนธ์นี้ใช้ได้จากเกมแอ็คชั่นสองมิติเน้นตัวละครที่งานวิทยานิพนธ์นี้นำมาวิเคราะห์ซึ่งเป็นเกมที่ได้รับความนิยม และเนื่องจากศัตรูเป็นองค์ประกอบสำคัญหลักของเกมลักษณะนี้ ในที่นี้จึงถือว่าความนิยมของเกมส่วนหนึ่งมาจากการออกแบบศัตรูที่ดีที่ผู้เล่นยอมรับได้ด้วย ดังนั้นชุดข้อมูลนี้จึงนับได้ว่าเป็นชุดข้อมูลของรูปแบบพฤติกรรมที่ผู้เล่นยอมรับได้

ในงานวิทยานิพนธ์นี้ ผู้วิจัยจะทำซ้ำรูปแบบพฤติกรรมของศัตรูส่วนหนึ่งจากเกมทั้ง 5 ที่นำมาวิเคราะห์ ได้แก่ Metroid, Super C, Megaman, Megaman 4 และ Shovel Knight โดยใช้ภาษาอธิบายภาคีฯในการอธิบายรูปแบบพฤติกรรมเพื่อใช้เป็นชุดข้อมูลรูปแบบพฤติกรรม

ถึงแม้ว่าศัตรูหนึ่ง ๆ จะสามารถอธิบายด้วยภาษาอธิบายภาคีได้หลายวิธี แต่ผู้ที่มีประสบการณ์ในการเล่นเกมแอ็คชั่นสองมิติแบบเน้นตัวละครจะสามารถตีความพฤติกรรมออกมาเป็นภาษาอธิบายภาคีฯในลักษณะที่เหมือนกัน สามารถแบ่งจำนวนสถานะของศัตรูได้เหมือนกัน รู้ว่าแต่ละสถานะควรจะเกิดพฤติกรรมแบบใดบ้าง จุดที่แตกต่างกันจะเป็นลำดับปรากฏของลำดับพฤติกรรมคู่ขนานภายในสถานะหนึ่ง ๆ ซึ่งไม่มีผลต่อการทำเหมืองข้อมูล เนื่องจากข้อมูลความสัมพันธ์แบบคู่ขนานไม่มีลำดับมาเกี่ยวข้อง ดังนั้น การที่ผู้วิจัยซึ่งมีประสบการณ์เล่นเกมแอ็คชั่นสองมิติแบบเน้นตัวละครเป็นผู้ทำซ้ำรูปแบบพฤติกรรม จึงไม่ได้ทำให้ชุดข้อมูลเกิดความเอนเอียงเป็นพิเศษมากไปกว่าการให้ผู้เล่นอื่นที่มีประสบการณ์เป็นผู้ทำซ้ำ

* + 1. ความสัมพันธ์ของพฤติกรรม

ในเกมแอ็คชั่นแบบเน้นตัวละคร ผู้เล่นจะต้องเรียนรู้และโต้ตอบกับศัตรูที่มีรูปแบบพฤติกรรมที่ตายตัว วิธีที่ผู้เล่นโต้ตอบกับศัตรูจะขึ้นอยู่กับรูปแบบพฤติกรรมของศัตรู หากศัตรูมีรูปแบบพฤติกรรมที่คล้ายคลึงกัน ผู้เล่นจะใช้แนวคิดในการโต้ตอบกับศัตรู เช่น ศัตรู Metall ใน Megaman, ศัตรู Pakatto24 ใน Megaman 4 และศัตรูป้อมปืนผนังใน Super C ต่างก็มีรูปแบบพฤติกรรมที่มีลักษณะเป็นการหลบเข้าเกราะกำบังแล้วออกมาจากเกราะเพื่อโจมตีผู้เล่นสลับกันไปมา ผู้เล่นจะใช้วิธีการหลอกล่อให้ศัตรูออกมาจากเกราะกำบังแล้วโจมตีสวนกลับเพื่อจัดการกับศัตรูทั้งสามชนิดนี้

การที่ศัตรูในเกมต่าง ๆ ซึ่งเป็นที่นิยมมีรูปแบบพฤติกรรมที่คล้ายคลึงแสดงให้เห็นว่า มีแบบแผนบางอย่างในการออกแบบรูปแบบพฤติกรรมให้ออกศัตรูมาเป็นที่ยอมรับได้โดยผู้เล่น โดยที่แบบแผนดังกล่าวไม่ได้มีเพียงหนึ่งเดียวเนื่องจากศัตรูแต่ละชนิดในเกมที่นำมาวิเคราะห์ต่างก็มีรูปแบบพฤติกรรมที่จำเป็นต้องโต้ตอบด้วยวิธีที่แตกต่างกันออกไป งานวิทยานิพนธ์นี้จะใช้วิธีการทำเหมืองข้อมูลบนชุดข้อมูลรูปแบบพฤติกรรมเพื่อค้นหาแบบแผนทั้งหลายดังกล่าวบนสมมติฐานที่ว่า “แบบแผนที่พบบ่อย” น่าจะเป็นแนวทางที่เหมาะสมสำหรับใช้ออกแบบรูปแบบพฤติกรรมของศัตรู

เนื่องจากการทำเหมืองข้อมูลจำเป็นต้องรู้ว่าจุดประสงค์ของการทำเหมืองข้อมูลคืออะไร มีข้อมูลขาเข้าและขาออกเป็นอย่างไร จึงจะสามารถเลือกขั้นตอนวิธีสำหรับการทำเหมืองข้อมูลได้ ในที่นี้มีจุดประสงค์เพื่อค้นหาแบบแผนสำหรับออกแบบรูปแบบพฤติกรรม คำถามสำคัญจึงเป็น “แบบแผนซึ่งเป็นข้อมูลขาออกที่ต้องการค้นหาจากชุดข้อมูลมีลักษณะเป็นอย่างไร” เมื่อกำหนดให้ข้อมูลขาเข้าเป็นชุดข้อมูลรูปแบบพฤติกรรมซึ่งอยู่ในรูปของภาษาอธิบายภาคีฯ

ภาษาอธิบายภาคีฯในงานวิทยานิพนธ์นี้ใช้สำหรับอธิบายรูปแบบพฤติกรรมโดยใช้การเรียงต่อฟังก์ชั่นการกระทำและเงื่อนไขออกมาเป็นลำดับพฤติกรรม ลำดับพฤติกรรมหลายลำดับจะถูกรวมเป็นพฤติกรรมคู่ขนานของสถานะหนึ่ง ๆ สถานะหลายสถานะร่วมกับการตั้งค่าคุณสมบัติในบล็อกเริ่มต้นจะถูกรวมเป็นภาคีศัตรูหนึ่งชนิด ศัตรูบางชนิดอาจมีความสามารถในการสร้างภาคีชนิดอื่น ๆ ด้วยฟังก์ชั่นการกระทำ Spawn เพื่อให้ภาคีที่สร้างใหม่แสดงผลการกระทำไปพร้อมกับตนได้ หากพิจารณาลักษณะของภาษาอธิบายภาคีฯและพฤติกรรมผลลัพธ์ที่เกิดจากลักษณะของภาษา จะพบว่าความแตกต่างในเชิงความหมายของพฤติกรรมหรือพฤติกรรมผลลัพธ์ที่เกิดขึ้นเกิดจากการเลือกใช้ฟังก์ชั่นการกระทำที่แตกต่างกันและ/หรือความสัมพันธ์ระหว่างองค์ประกอบภายในภาษาอธิบายภาคีฯดังนี้

* ลำดับของฟังก์ชั่นการกระทำและโครงสร้างเงื่อนไข: สามารถยกตัวอย่างความแตกต่างที่เกิดจากลำดับ เช่น มีฟังก์ชั่นการกระทำวิ่ง, กระโดด และยิง หากนำมาเรียงเป็นวิ่ง, กระโดด และยิง ศัตรูจะยิงหลังจากกระโดด ในขณะที่หากเรียงการกระทำเป็นกระโดด, วิ่ง, ยิง ซึ่งศัตรูจะยิงหลังจากวิ่ง เป็นต้น ความแตกต่างเช่นนี้ทำให้ผู้เล่นต้องโต้ตอบด้วยการรอจังหวะของพฤติกรรมที่ต่างกัน
* ฟังก์ชั่นการกระทำที่คู่ขนานกัน: ส่งผลให้เกิดความแตกต่างได้ในกรณีที่ฟังก์ชั่นการกระทำที่เกิดคู่ขนานกันมีความแตกต่างกัน เช่น ฟังก์ชั่นการกระทำกระโดดคู่ขนานกับยิง และฟังก์ชั่นการกระทำวิ่งคู่ขนานกับยิง ย่อมส่งผลให้วิถีกระสุนที่ได้แตกต่างกันถึงแม้จะยิงกระสุนแบบเดียวกันเนื่องจากจุดยิงแตกต่างกัน
* ฟังก์ชั่นข้อมูลที่เลือกใช้: แม้ฟังก์ชั่นการกระทำหรือฟังก์ชั่นข้อมูลจะเป็นฟังก์ชั่นเดียวกัน หากเลือกใช้ข้อมูลที่แตกต่างกันย่อมมีความหมายแตกต่างกันและอาจทำให้เกิดผลลัพธ์ที่แตกต่างกัน เช่น ในการกระทำระยะยาว RunStraight ที่สั่งให้ภาคีวิ่งตรงไปข้างหน้าที่มีเงื่อนไขเสร็จสิ้นการกระทำ (พารามิเตอร์ endCondition) เป็นฟังก์ชั่นข้อมูล TimePass() > 40 เพื่อจำกัดระยะเวลาที่วิ่งได้ ย่อมได้ผลลัพธ์ต่างไปจากการใช้ฟังก์ชั่นข้อมูล SurfaceInFront() เพื่อสั่งให้หยุดวิ่งเมื่อเจอกับโครงสร้างพื้นที่ที่ผ่านไม่ได้เท่านั้น

แบบแผนของการออกแบบพฤติกรรมที่ได้ยกตัวอย่างไปก่อนหน้านี้เกิดจากการจัดเรียงองค์ประกอบภายในภาษาอธิบายภาคีฯอย่างเป็นระบบ (ในกรณีของตัวอย่างศัตรู Metall และศัตรู Pakatto24 เกิดจากการจัดเรียงการกระทำภายในลำดับพฤติกรรมแบบเดียวกัน) การมีแบบแผนได้หลายแบบหมายความว่ามีวิธีในการจัดเรียงองค์ประกอบให้เกิดเป็นรูปแบบพฤติกรรมที่แตกต่างกันได้หลายวิธี และวิธีการจัดเรียงองค์ประกอบของภาษาอธิบายภาคีฯให้เกิดความแตกต่างของรูปแบบพฤติกรรมสามารถทำได้โดยการเลือกใช้ฟังก์ชั่นการกระทำที่แตกต่างกันร่วมกับความสัมพันธ์ที่ทำให้เกิดพฤติกรรมที่แตกต่างทั้ง 3 รูปแบบข้างต้น

ดังนั้น แบบแผนของรูปแบบพฤติกรรมที่ต้องการจากการทำเหมืองข้อมูลในที่นี้จะอยู่ในรูปความสัมพันธ์ของฟังก์ชั่นการกระทำทั้ง 3 รูปแบบอันได้แก่ ลำดับของฟังก์ชั่นการกระทำ, ฟังก์ชั่นการกระทำที่คู่ขนานกัน และการเลือกใช้ฟังก์ชั่นข้อมูล

* + 1. การทำเหมืองข้อมูลสำหรับลำดับของฟังก์ชั่นการกระทำ

การทำเหมืองข้อมูลสำหรับลำดับฟังก์ชั่นการกระทำมีจุดประสงค์เพื่อค้นหาว่า การจัดเรียงฟังก์ชั่นการกระทำที่พบได้บ่อยในชุดข้อมูลของรูปแบบพฤติกรรมเป็นอย่างไร ซึ่งข้อมูลที่จะถูกนำมาใช้เป็นข้อมูลขาเข้า คือ ลำดับพฤติกรรมทั้งหมดที่พบในชุดข้อมูล โดยข้อมูลจะถูกแบ่งออกเป็น 3 ชุด ได้แก่ ข้อมูลลำดับพฤติกรรมเฉพาะบล็อกเริ่มต้น, ข้อมูลลำดับพฤติกรรมเฉพาะบล็อกทำลาย และข้อมูลลำดับพฤติกรรมที่ได้จากกลุ่มสถานะ

เนื่องจากความสัมพันธ์ผลลัพธ์อยู่ในรูปของลำดับ งานวิทยานิพนธ์นี้จึงใช้ขั้นตอนวิธี PrefixSpan ซึ่งเป็นขั้นตอนวิธีสำหรับการหารูปแบบจากฐานข้อมูลลำดับ ดังนั้นจึงต้องมีการแปลงลำดับพฤติกรรมให้อยู่ในรูปของลำดับตามนิยามของปัญหาทางด้านการทำเหมืองข้อมูลสำหรับข้อมูลแบบลำดับ

วิธีที่ง่ายที่สุดในการแปลงลำดับพฤติกรรมไปเป็นข้อมูลลำดับคือการไล่รหัสจากบรรทัดแรกสุดของลำดับจนถึงบรรทัดสุดท้ายของลำดับ หากพบฟังก์ชั่นการกระทำให้แปลงฟังก์ชั่นการกระทำเป็นไอเท็มเซ็ตที่ประกอบด้วยไอเท็มที่อยู่ในรูปของชื่อฟังก์ชั่นนั้นแล้วเพิ่มลงในข้อมูลลำดับ โดยวิธีการนี้มีข้อเสีย 2 ข้อได้แก่

1. ไม่สามารถแยกความแตกต่างระหว่างฟังก์ชั่นที่อยู่/ไม่ได้อยู่ภายใต้โครงสร้างเงื่อนไข รูปที่ 4.7 แสดงให้เห็นลำดับซึ่งแปลงมาจากฟังก์ชั่นเดียวกันที่อยู่และไม่อยู่ใต้โครงสร้างเงื่อนไข
2. การใช้ชื่อฟังก์ชั่นเป็นไอเท็มทำให้ใช้เวลาตรวจสอบความเท่ากันของไอเท็มใช้นานกว่าการเลือกใช้ตัวเลขเป็นไอเท็ม

|  |
| --- |
| .seq1{ if(Abs(DistanceToPlayer("Y")) <= 30){  Despawn();  }  }  ลำดับ: <Despawn>  แปลง  ลำดับ: <Despawn>  แปลง  .seq1{  Despawn();  } |
| รูปที่ 4.7 การแปลงลำดับพฤติกรรมโดยใช้เฉพาะชื่อฟังก์ชั่น |

การแก้ข้อเสียที่ได้กล่าวถึงไปทำได้โดยการใช้ตัวเลขเป็นไอเท็มโดยตัวเลขดังกล่าวจะต้องเก็บชื่อฟังก์ชั่นและข้อมูลโครงสร้างเงื่อนไขที่ครอบฟังก์ชั่นอยู่ได้ ในที่นี้ใช้ตัวเลขจำนวนเต็มขนาด 32 บิทโดยมีวิธีการบีบอัดข้อมูลเป็นไปดังตารางที่ 4.4

|  |  |  |  |
| --- | --- | --- | --- |
| ตารางที่ 4.4 การบีบอัดข้อมูลฟังก์ชั่นการกระทำ | | | |
| บิทที่ | ความหมาย | คำอธิบาย | |
| 0-5 | การกระทำ | ใช้สำหรับเก็บชื่อการกระทำ ปัจจุบันมีฟังก์ชั่นการกระทำทั้งหมด 31 แบบ ฟังก์ชั่นการกระทำแต่ละแบบจะถูกกำหนดตัวเลขแทนชื่อโดยเริ่มต้นจาก 0 | |
| 6-7 | ชนิดของโครงสร้างเงื่อนไข 1 | โครงสร้างเงื่อนไขที่ครอบฟังก์ชั่นการกระทำอยู่ หากมีโครงสร้างเงื่อนไขสองชั้น จะเป็นรายละเอียดของโครงสร้างเงื่อนไขชั้นนอก โดยแต่ละค่าแทนความหมายดังนี้ | |
| 00: ไม่มีโครงสร้างเงื่อนไขนี้ | 01: ฟังก์ชั่นการกระทำถูกครอบด้วยโครงสร้างเงื่อนไข If |
| 10: ฟังก์ชั่นการกระทำอยู่ในบล็อก If ของโครงสร้างเงื่อนไขแบบ If-Else | 11: ฟังก์ชั่นการกระทำอยู่ในบล็อก Else ของโครงสร้างเงื่อนไขแบบ If-Else |
| 8-12 | นิพจน์ 1 ของโครงสร้างเงื่อนไข 1 | ฟังก์ชั่นข้อมูลที่ใช้สร้างนิพจน์รากของเงื่อนไข โดยในที่นี้เลือกเฉพาะฟังก์ชั่นข้อมูลที่คืนค่าเป็นข้อมูลชนิดบูลหรือตัวเลขซึ่งเป็นข้อมูลเพียง 2 ชนิดที่ใช้เป็นนิพจน์รากของเงื่อนไขได้ ฟังก์ชั่นข้อมูลที่คืนชนิดข้อมูลที่กำหนดมีเพียง 28 ฟังก์ชั่น แต่ละฟังก์ชั่นจะถูกกำหนดตัวเลขแทนชื่อโดยเริ่มต้นจาก 0 หากไม่มีฟังก์ชั่นนี้ให้ใช้ค่า 11111 | |
| 13-17 | นิพจน์ 2 ของโครงสร้างเงื่อนไข 1 | ฟังก์ชั่นข้อมูลที่ 2 หากไม่มีให้ใช้ค่า 11111 | |
| 18 | ความสัมพันธ์ของนิพจน์ 1 และ 2 ของเงื่อนไข 1 | 0: เมื่อนิพจน์ 1 และนิพจน์ 2 เชื่อมกันด้วยตัวดำเนินการตรรกะหรือตัวเปรียบเทียบ (โดยตัดสัญพจน์ออกจากนิพจน์ทั้งหมด)  1: เมื่อนิพจน์ 2 เป็นอากิวเมนท์แรกที่เป็นฟังก์ชั่นข้อมูลของนิพจน์ 1  หากไม่มีนิพจน์ 1 ให้ใช้ค่า 0 | |
| 19-31 | เงื่อนไข 2 | ความหมายและความยาวบิทจะเหมือนกับเงื่อนไข 1 | |

โดยใช้วิธีการบีบอัดข้อมูลดังกล่าว ลำดับพฤติกรรมจึงสามารถแปลงให้อยู่ในรูปของข้อมูลลำดับได้โดยแต่ละไอเท็มเซ็ตในลำดับจะมีไอเท็มเพียงตัวเดียวเหมือนกับกรณีอย่างง่ายที่ได้ยกตัวอย่างไปแล้ว แต่เก็บข้อมูลได้ครบถ้วนกว่าและเป็นข้อมูลแบบตัวเลขซึ่งเปรียบเทียบความเท่ากันได้รวดเร็วกว่าข้อมูลแบบข้อความ รูปที่ 4.8 แสดงตัวอย่างการแปลงลำดับพฤติกรรมเป็นข้อมูลลำดับ ข้อมูลลำดับที่ได้สามารถนำเข้าสู่ขั้นตอนวิธี PrefixSpan ต่อไป

ผลลัพธ์จากขั้นตอนวิธี PrefixSpan จะได้ออกมาเป็นข้อมูลลำดับซึ่งสามารถทำการแปลงค่ากลับไปเป็นฟังก์ชั่นการกระทำและเงื่อนไขที่ “ไม่สมบูรณ์” การนำเอาผลลัพธ์ดังกล่าวนี้ไปใช้งานต่อจะต้องมีการเติมเต็มส่วนที่ขาดหายไปซึ่งเป็นหน้าที่ของส่วนสร้างรูปแบบพฤติกรรมซึ่งจะกล่าวถึงในหัวข้อ 4.3

|  |
| --- |
| Wait 16762878  **00000 00 11111 11111 0 00 11111 11111 0**  Jump (If-block) 103933950  **00001 10 00110 00111 1 00 11111 11111 0**  Jump (Else-block) 120711166  **00001 11 00110 00111 1 00 11111 11111 0**  Goto (If-block) 171042814  **00010 10 00110 00111 1 00 11111 11111 0**  Goto (Else-block) 187820030  **00010 11 00110 00111 1 00 11111 11111 0**  Wait(TimePass() >= 60);  if(Random(DecimalSet(1,2,1)) == 1){  Jump(Anchor("c(32,0)"), 180, 5, SurfaceInDir("south"));  }else{  Jump(Get("position", DynamicFilter("player")),300, 5, SurfaceInDir("south"));  }  if(Random(DecimalSet(1,2,1)) == 1){  Goto(.flame);  }else{  Goto(.bullet);  }  บีบอัด  ตัวเลขแทนชื่อสำหรับฟังก์ชั่นการกระทำ  0: Wait 1: Jump 2: Goto  ตัวเลขแทนชื่อสำหรับฟังก์ชั่นข้อมูล  6: Random 7: DecimalSet  ข้อมูลลำดับ <16762878 103933950 120711166 171042814 187820030> |
| รูปที่ 4.8 ตัวอย่างการบีบอัดลำดับพฤติกรรมเป็นข้อมูลลำดับ |

นอกเหนือจากการทำเหมืองข้อมูลโดยใช้ลำดับพฤติกรรมเป็นข้อมูลนำเข้าแล้ว ในที่นี้ยังมีข้อมูลลำดับอีกลักษณะหนึ่งซึ่งเป็นลำดับพฤติกรรมที่เกิดจากฟังก์ชั่นการกระทำที่ใช้เปลี่ยนสถานะของภาคี การเปลี่ยนสถานะของภาคีทำให้พฤติกรรมผลลัพธ์ที่เกิดขึ้นเป็นเสมือนลำดับพฤติกรรมที่ประกอบด้วยลำดับพฤติกรรมที่กระทำก่อนเปลี่ยนสถานะเชื่อมต่อกับลำดับพฤติกรรมของสถานะปลายทาง ความต่อเนื่องของพฤติกรรมข้ามสถานะน่าจะมีแบบแผนเกี่ยวกับการเปลี่ยนแปลงจากชุดพฤติกรรมหนึ่งไปเป็นอีกชุดหนึ่ง ในที่นี้จึงทำเหมืองข้อมูลเพื่อค้นหาความสัมพันธ์ในลักษณะนี้เพิ่มเติมจากลำดับพฤติกรรม โดยเรียกลำดับพฤติกรรมที่เกิดจากการเชื่อมระหว่างลำดับพฤติกรรมก่อนและหลังเปลี่ยนสถานะว่า ลำดับพฤติกรรมเชื่อม (Joined behavior sequence) โดยลำดับพฤติกรรมก่อนเปลี่ยนสถานะจะเรียกว่าลำดับพฤติกรรมต้นทาง และลำดับพฤติกรรมหลังเปลี่ยนสถานะเรียกว่าลำดับพฤติกรรมปลายทาง

ลำดับพฤติกรรมเชื่อมสามารถสร้างขึ้นจากชุดข้อมูลรูปแบบพฤติกรรมโดยการเลือกฟังก์ชั่นการกระทำและโครงสร้างเงื่อนไขทั้งหมดที่มีโอกาสถูกแปลตั้งแต่เริ่มต้นลำดับจนถึงฟังก์ชั่นการกระทำที่เปลี่ยนสถานะได้ (ฟังก์ชั่น Goto และ Destroy) เป็นลำดับพฤติกรรมต้นทาง จากนั้นจึงเชื่อมต่อกับลำดับพฤติกรรมปลายทาง ซึ่งในที่นี้คือ ลำดับพฤติกรรมของสถานะปลายทาง หากสถานะปลายทางมีมากกว่า 1 ลำดับพฤติกรรม จะต้องมีการเชื่อมกับลำดับพฤติกรรมทั้งหมด ทำให้เกิดเป็นลำดับพฤติกรรมเชื่อมจำนวนเท่ากับจำนวนลำดับพฤติกรรมของสถานะปลายทาง

ในกรณีที่ลำดับพฤติกรรมต้นทางมีฟังก์ชั่นเปลี่ยนสถานะมากกว่า 1 ฟังก์ชั่น จะต้องสร้างลำดับพฤติกรรมต้นทางสำหรับทุกฟังก์ชั่น และเชื่อมทุกลำดับพฤติกรรมต้นทางกับทุกลำดับพฤติกรรมปลายทาง รูปที่ 4.9 แสดงตัวอย่างการสร้างลำดับพฤติกรรมเชื่อม โดยแสดงในรูปรหัสเทียม

|  |
| --- |
| ลำดับพฤติกรรมต้นทาง  .seq0{  A  B  If(CA){  }else{  D  }  If(CB){  Destroy  }  }  ลำดับพฤติกรรมเชื่อม  .des{  C  D  }  .seq0{  A  B  If(CA){  C  Goto(.state1)  }else{  D  }  If(CB){  Destroy  }  E  }  .state1{  .seq0{  A  B  C  }  .seq1{  D  E  }  }  .seq0{  A  B  If(CA){  C  Goto(.state1)  }else{  }  }  สร้าง  ลำดับพฤติกรรมเชื่อม  (.seq0 + .state1.seq0){  A  B  If(CA){  C  Goto(.state1)  }else{  }  A  B  C  }  (.seq0 + .state1.seq1){  A  B  If(CA){  C  Goto(.state1)  }else{  }  D  E  }  (.seq0 + .des){  A  B  If(CA){  }else{  D  }  If(CB){  Destroy  }  C  D  }  เชื่อม  เชื่อม |
| รูปที่ 4.9 ตัวอย่างการสร้างลำดับพฤติกรรมเชื่อม |

เมื่อได้ลำดับพฤติกรรมเชื่อมแล้ว จึงทำการแปลงลำดับพฤติกรรมเชื่อมให้อยู่ในรูปของลำดับเชื่อม (Joined sequence) โดยงานวิทยานิพนธ์นี้กำหนดนิยามลำดับเชื่อมว่า ลำดับเชื่อม เมื่อ  และ  เป็นลำดับ โดย  เป็นลำดับต้นทาง และ เป็นลำดับปลายทาง

ลำดับพฤติกรรมเชื่อมสามารถแปลงเป็นข้อมูลลำดับเชื่อมได้โดยแปลงลำดับพฤติกรรมต้นทางเป็นข้อมูลลำดับด้วยวิธีการบีบอัดข้อมูลเช่นเดียวกับการแปลงลำดับพฤติกรรมเป็นข้อมูลลำดับ จากนั้นตัดไอเท็มเซ็ตสุดท้ายซึ่งเป็นฟังก์ชั่นเปลี่ยนสถานะที่ผ่านการบีบอัดออก และกำหนดให้ข้อมูลลำดับที่ได้นี้เป็นลำดับต้นทางของข้อมูลลำดับเชื่อม จากนั้นใช้วิธีบีบอัดเพื่อแปลงลำดับพฤติกรรมปลายทางเป็นข้อมูลลำดับและใช้เป็นลำดับปลายทางของข้อมูลลำดับเชื่อม นอกจากนี้ลำดับเชื่อมที่ได้จะถูกเสริมด้วยข้อมูลตัวเลข 1 ตัวเพื่อบอกจำนวนสถานะทั้งหมดของภาคีนี้ ดังนั้นลักษณะของลำดับเชื่อมขาเข้าสำหรับทำเหมืองข้อมูลจึงเป็น  โดย  เป็นข้อมูลเสริม รูปที่ 4.10 แสดงตัวอย่างการแปลงลำดับพฤติกรรมเชื่อมเป็นข้อมูลลำดับเชื่อม โดยในตัวอย่างจะใช้เขียนชื่อฟังก์ชั่นลงไปแทนการบีบอัดเป็นข้อมูลตัวเลขเพื่อความเรียบง่าย และสมมติว่าภาคีเจ้าของลำดับมีจำนวนสถานะทั้งหมดเท่ากับ 4 สถานะ

|  |
| --- |
| ลำดับพฤติกรรมเชื่อม  (.seq0 + .des){  A  B  If(CA){  }else{  D  }  If(CB){  Destroy  }  C  D  }  ลำดับพฤติกรรมเชื่อม:  <A B D:(IfElse-CA-Else) | C D>(4)  ลำดับพฤติกรรมปลายทาง: <C D>  เชื่อม  ลำดับพฤติกรรมต้นทาง:  <A B D:(IfElse-CA-Else) ~~Destroy:(If-CB)~~> |
| รูปที่ 4.10 ตัวอย่างการแปลงลำดับพฤติกรรมเชื่อมเป็นข้อมูลลำดับเชื่อม |

การทำเหมืองข้อมูลสำหรับข้อมูลลำดับเชื่อมจะทำการหาลำดับเชื่อมย่อย  ที่พบบ่อยจากฐานข้อมูลลำดับเชื่อม  โดยนิยามต่าง ๆ เหมือนกับการทำเหมืองข้อมูลบนข้อมูลลำดับ ยกเว้นลำดับเชื่อมย่อย (Sub-joined sequence) ซึ่งมีนิยามว่า ลำดับเชื่อม  เป็นลำดับเชื่อมย่อยของ  เมื่อ  และ  และ 

สำหรับขั้นตอนวิธีทำเหมืองข้อมูลจะนำเอา PrefixSpan มาแก้ไขโดยปรับปรุงขั้นตอนบันทึกลำดับความยาว 1 ที่พบบ่อย โดยบันทึกลำดับเชื่อม  ที่พบบ่อยแทนเพื่อเป็นการยืนยันว่าลำดับต้นทางและลำดับปลายทางของลำดับเชื่อมผลลัพธ์ไม่เป็นลำดับว่างและลำดับเชื่อมมีข้อมูลเสริม จากนั้นในขั้นตอนแบ่งปริภูมิค้นหา ให้สร้างฐานข้อมูลภาพฉายของฐานข้อมูลเริ่มต้นสำหรับทุกลำดับเชื่อมที่พบบ่อย โดยแบ่งเป็นฐานข้อมูลภาพฉายสำหรับลำดับต้นทางและฐานข้อมูลภาพฉายสำหรับลำดับปลายทางแยกออกจากกัน แล้วใช้ขั้นตอนวิธี PrefixSpan ปกติสำหรับหาลำดับย่อยที่พบบ่อยจากฐานข้อมูลภาพฉายทั้งสอง จากนั้นทำการเชื่อมผลลัพธ์โดยใช้ผลลัพธ์จากฐานข้อมูลภาพฉายสำหรับลำดับต้นทางเป็นลำดับต้นทางของลำดับเชื่อมและผลลัพธ์จากฐานข้อมูลภาพฉายสำหรับลำดับปลายทางเป็นลำดับปลายทางของลำดับเชื่อมภายใต้เงื่อนไขว่า ลำดับเชื่อมย่อยที่ได้จะต้องเป็นลำดับเชื่อมย่อยที่พบบ่อยในฐานข้อมูลเริ่มต้น ขั้นตอนวิธีทำเหมืองข้อมูลสำหรับข้อมูลลำดับเชื่อมแสดงด้วยรหัสเทียมได้ดังรูปที่ 4.11 โดยในที่นี้เรียกขั้นตอนวิธีนี้ว่า JS-PrefixSpan

|  |
| --- |
| การทำเหมืองข้อมูลบนฐานข้อมูลลำดับเชื่อม  ซึ่งประกอบด้วยลำดับเชื่อม และมีเซ็ตของไอเท็ม และเซ็ตของตัวเลขเสริมที่เป็นไปได้  กำหนดค่าซัพพอร์ทขั้นต่ำ  เป็นดังนี้  Initialize ,  For ( in ) do:  For ( in ) do:  For ( in ) do:  if()    For(in ) do:  Create -projected database of  PrefixSpan on  with itemset and  Create -projected database of  PrefixSpan on  with itemset and  For( in ) do:  For( in ) do:  If()    Return  as result  เมื่อ  และ  และลำดับย่อยผลลัพธ์จาก PrefixSpan มีรายการ  ซึ่งเก็บ  ของลำดับที่บรรจุลำดับย่อยดังกล่าว |
| รูปที่ 4.11 แสดงขั้นตอน JS-PrefixSpan |

ผลลัพธ์ที่ได้จากขั้นตอนวิธี JS-PrefixSpan จะอยู่ในรูปลำดับเชื่อมที่มีข้อมูลเสริมซึ่งสามารถแปลงกลับเป็นลำดับพฤติกรรมต้นทางและลำดับพฤติกรรมปลายทางพร้อมจำนวนลำดับพฤติกรรมในสถานะปลายทาง โดยลำดับพฤติกรรมที่แปลงกลับมาจะไม่สมบูรณ์เช่นเดียวกับลำดับพฤติกรรมที่ได้จากขั้นตอนวิธี PrefixSpan ซึ่งต้องเติมเต็มในขั้นตอนสร้างรูปแบบพฤติกรรมอัตโนมัติเช่นเดียวกัน

* + 1. การทำเหมืองข้อมูลสำหรับฟังก์ชั่นการกระทำคู่ขนาน

การทำเหมืองข้อมูลสำหรับฟังก์ชั่นการกระทำคู่ขนานมีจุดประสงค์เพื่อค้นหาแบบแผนของฟังก์ชั่นที่ทำงานขนานกันที่พบได้บ่อย ฟังก์ชั่นที่ทำงานขนานกันจะไม่สนใจลำดับการทำงานสำหรับฟังก์ชั่นที่มาจากลำดับพฤติกรรมเดียวกัน สิ่งที่ต้องให้ความสนใจมีเพียงฟังก์ชั่นใดทำงานคู่ขนานไปกับฟังก์ชั่นใดบ้าง ในสถานะหนึ่งสามารถมีลำดับพฤติกรรมที่ทำงานคู่ขนานกันได้หลายลำดับ อีกทั้งแต่ละลำดับพฤติกรรมจะมีฟังก์ชั่นอยู่ได้เป็นจำนวนมาก ทำให้จำนวนพฤติกรรมที่ขนานกันเพิ่มเป็นทวีคูณ ยกตัวอย่างภาคีที่ประกอบด้วยฟังก์ชั่นจำนวนน้อย เช่น สถานะของภาคีหนึ่งมี 3 ลำดับพฤติกรรมได้แก่ AB, DE และ F จะมีพฤติกรรมที่คู่ขนานกันได้ทั้งหมดถึง 24 แบบ (A/D, A/E, B/D, B/E, A/F, B/F, D/F, E/F, A/D/F, A/E/F, B/D/F, B/E/F, AB/D, AB/E, AB/F, A/DE, B/DE, DE/F, AB/D/F, AB/E/F, A/DE/F, B/DE/F, AB/DE, AB/DE/F) ดังนั้น การแจงนับพฤติกรรมคู่ขนานเพื่อสร้างฐานข้อมูลลำดับแล้วใช้ PrefixSpan ในการหาย่อมไม่มีประสิทธิภาพ และยังเกิดข้อผิดพลาดขึ้นได้จากการแจงนับข้อมูลที่ซ้ำซ้อน เช่น ลำดับพฤติกรรม AB, BA จะแจงนับได้เป็น A/B, A/A, B/B, B/A, AB/B, AB/A, A/BA, B/BA, AB/BA มีตัวอย่างข้อมูลคู่ขนานที่ซ้ำกัน เช่น A/B และ B/A, AB/B และB/BA เป็นต้น

เพื่อป้องกันปัญหาการแจงนับซ้ำซ้อนรวมถึงปัญหาประสิทธิภาพจากการแจงนับ งานวิทยานิพนธ์นี้จึงทำการแปลงลำดับพฤติกรรมที่ขนานกันให้อยู่ในรูปของกราฟระบุชื่อแบบมีทิศทางไม่มีวัฏจักร การใช้กราฟมีข้อดีคือไม่ต้องมีการแจงนับพฤติกรรมคู่ขนาน แม้ว่าจะมีข้อเสียที่ต้องทดสอบความสมสัณฐานเพื่อนับค่าซัพพอร์ทซึ่งเป็นปัญหาที่มีความซับซ้อนระดับ NP-Complete แต่ก็มีงานวิจัยทางด้านการทำเหมืองข้อมูลสำหรับข้อมูลแบบกราฟเพื่อลดทอนจำนวนครั้งการทดสอบความสมสัณฐานลง เช่น ขั้นตอนวิธี gSpan ที่งานวิทยานิพนธ์นี้เลือกใช้

งานวิทยานิพนธ์นี้แปลงลำดับพฤติกรรมเป็นข้อมูลกราฟซึ่งแต่ละจุดยอดจะมีป้ายชื่อเป็นตัวเลขซึ่งได้จากการบีบอัดฟังก์ชั่นการกระทำ โดยมีจุดยอดที่มีป้ายชื่อพิเศษสำหรับใช้ในการแบ่งฟังก์ชั่นการกระทำที่มาจากลำดับพฤติกรรมที่แตกต่างกัน วิธีการแปลงลำดับพฤติกรรมเป็นกราฟทำได้โดยการสร้างจุดยอดรากซึ่งมีป้ายชื่อพิเศษเป็น 134217727 (00000 11 11111 11111 1 11 11111 11111 1) ซึ่งไม่มีการกระทำที่บีบอัดแล้วได้ข้อมูลนี้ จากนั้นจึงสร้างจุดยอดเพิ่มเติมเท่ากับจำนวนลำดับพฤติกรรมที่ขนานกันอยู่เพื่อแบ่งฟังก์ชั่นการกระทำจากต่างลำดับพฤติกรรมออกจากกัน แต่ละจุดยอดเพิ่มเติมจะเป็นตัวแทนของแต่ละลำดับพฤติกรรมโดยมีป้ายชื่อ 134217727 เหมือนกันทั้งหมด ทุกจุดยอดเพิ่มเติมจะมีเส้นเชื่อมจากจุดยอดรากมายังจุดยอดดังกล่าว ท้ายที่สุดสำหรับทุกจุดยอดเพิ่มเติม จะทำการสร้างจุดยอดสำหรับแทนฟังก์ชั่นจำนวนเท่ากับจำนวนฟังก์ชั่นการกระทำในลำดับพฤติกรรมที่คู่กันโดยแต่ละจุดยอดจะมีป้ายชื่อเป็นตัวเลขแทนฟังก์ชั่นการกระทำ แล้วสร้างเส้นเชื่อมจากจุดยอดเพิ่มเติมมายังจุดยอดแทนฟังก์ชั่นนี้ สุดท้ายจึงกำหนดป้ายชื่อให้กับทุกเส้นเชื่อมเป็น 0 รูปที่ 4.12 แสดงตัวอย่างการแปลงลำดับพฤติกรรมที่ขนานกันเป็นข้อมูลกราฟ โดยลำดับพฤติกรรมจะแสดงในรูปข้อมูลลำดับสมมติ

|  |
| --- |
| ลำดับ 0: <0 1 2 3>  ลำดับ 1: <0 2 4>  ลำดับ 2: <5 3 3>  แปลง |
| รูปที่ 4.12 ตัวอย่างการแปลงลำดับพฤติกรรมที่ขนานกันเป็นกราฟ |

ข้อมูลพฤติกรรมคู่ขนานในรูปแบบกราฟจะสามารถใช้เป็นข้อมูลขาเข้าของขั้นตอนวิธี gSpan ซึ่งจะให้ผลลัพธ์ออกมาเป็นกราฟย่อยที่พบบ่อย กราฟย่อยที่ได้ที่มีจุดยอดเพิ่มเติมเพียงจุดยอดเดียวจะถูกคัดออกเนื่องจากไม่ได้แสดงความสัมพันธ์ของพฤติกรรมคู่ขนานแต่อย่างใด กราฟย่อยที่เหลือจะถูกแปลงกลับเป็นฟังก์ชั่นการกระทำที่ขนานกันเพื่อนำไปใช้ในขั้นตอนสร้างรูปแบบพฤติกรรมต่อไป

นอกจากพฤติกรรมที่คู่ขนานกันแล้ว หากพิจารณาฟังก์ชั่นการกระทำ Spawn ซึ่งสามารถสร้างภาคีเพิ่มลงในโลกของเกมได้และแสดงพฤติกรรมไปพร้อมกับภาคีเจ้าของ พฤติกรรมลักษณะนี้อาจมองได้ว่าเป็นแบบแผนหนึ่งที่แฝงอยู่ในความสัมพันธ์ของพฤติกรรมคู่ขนานแบบข้ามภาคี ในที่นี้จึงมีการทำเหมืองข้อมูลเพื่อค้นหาความสัมพันธ์ดังกล่าวนี้ด้วยโดยเรียกข้อมูลลักษณะนี้ว่าพฤติกรรมคู่ขนานข้ามภาคี ซึ่งประกอบด้วยลำดับพฤติกรรมทั้งหมดของภาคีเจ้าของเรียกว่า พฤติกรรมคู่ขนานเจ้าของ และลำดับพฤติกรรมของภาคีที่ถูกสร้างเรียกว่า พฤติกรรมคู่ขนานลูก

ข้อมูลพฤติกรรมคู่ขนานข้ามภาคีซึ่งเป็นข้อมูลขาเข้าในที่นี้สร้างขึ้นโดยเลือกสถานะของภาคีที่มีลำดับพฤติกรรมซึ่งมีฟังก์ชั่นการกระทำ Spawn และใช้ลำดับพฤติกรรมทั้งหมดในสถานะดังกล่าวเป็นพฤติกรรมคู่ขนานเจ้าของ จากนั้นเลือกสถานะเริ่มต้นของภาคีที่ถูกสร้างด้วยคำสั่ง Spawn และใช้ลำดับพฤติกรรมทั้งหมดในสถานะดังกล่าวเป็นพฤติกรรมคู่ขนานลูก

ในกรณีที่พฤติกรรมคู่ขนานเจ้าของมีการใช้งานฟังก์ชั่น Spawn เพื่อสร้างภาคีมากกว่า 1 ชนิด จะต้องสร้างพฤติกรรมคู่ขนานเจ้าของสำหรับภาคีทุกชนิดเพื่อจับคู่พฤติกรรมคู่ขนานลูกทั้งหมด รูปที่ 4.13 แสดงตัวอย่างการสร้างพฤติกรรมคู่ขนานข้ามภาคี โดยแสดงในรูปของรหัสเทียม

|  |
| --- |
| พฤติกรรมคู่ขนานเจ้าของ  .seq1{  D  E  }  .seq0{  A  B  Spawn(**.agent1**)  C  Spawn(**.agent2**)  }  **.agent2**{  .state0{  .seq0{  E  A  }  }  }  **.agent1**{  .state0{  .seq0{  D  C  }  .seq1{  A  B  }  }  }  .state3{  .seq0{  A  B  Spawn(.agent1)  C  Spawn(.agent2)  }  .seq1{  D  E  }  }  สร้าง  พฤติกรรมคู่ขนานลูก  .seq0{  D  C  }  .seq1{  A  B  }  สร้าง  พฤติกรรมคู่ขนานลูก  .seq0{  E  A  }  สร้าง  พฤติกรรมคู่ขนานข้ามภาคี  .seq0{  A  B  Spawn(**.agent1**)  C  Spawn(**.agent2**)  }  .seq1{  D  E  }  (.agent2).seq0{  E  A  }  รวม  พฤติกรรมคู่ขนานข้ามภาคี  .seq0{  A  B  Spawn(**.agent1**)  C  Spawn(**.agent2**)  }  .seq1{  D  E  }  (.agent1).seq0{  D  C  }  (.agent1).seq1{  A  B  }  รวม |
| รูปที่ 4.13 ตัวอย่างการสร้างพฤติกรรมคู่ขนานข้ามภาคี |

เมื่อได้พฤติกรรมคู่ขนานข้ามภาคีแล้ว จึงแปลงให้อยู่ในรูปของกราฟโดยใช้วิธีเดียวกับการแปลงลำดับพฤติกรรมคู่ขนานเป็นกราฟ เพียงแต่ต้องเปลี่ยนวิธีการตั้งชื่อเส้นเชื่อมเพื่อแยกความแตกต่างระหว่างลำดับพฤติกรรมที่มาจากพฤติกรรมคู่ขนานเจ้าของและลำดับพฤติกรรมที่มาจากพฤติกรรมคู่ขนานลูกออกจากกัน โดยตั้งชื่อเส้นเชื่อมที่เชื่อมจากจุดยอดรากไปยังจุดยอดเพิ่มเติมที่คู่กับพฤติกรรมคู่ขนานลูกเป็น 1 และตั้งชื่อเส้นเชื่อมที่เชื่อมจากจุดยอดเพิ่มเติมที่คู่กับพฤติกรรมคู่ขนานลูกไปยังจุดยอดแทนฟังก์ชั่นเป็น 1 เช่นกัน จากนั้นเพิ่มข้อมูลเสริมสำหรับบอกจำนวนลำดับพฤติกรรมของพฤติกรรมคู่ขนานลูกโดยสร้างจุดยอดที่มีป้ายชื่อเป็นจำนวนลำดับพฤติกรรมดังกล่าว จากนั้นจึงสร้างเส้นเชื่อมจากจุดยอดรากมายังจุดยอดข้อมูลเสริมนี้โดยกำหนดป้ายชื่อเส้นเชื่อมเป็น 2 รูปที่ 4.14 แสดงตัวอย่างกราฟผลลัพธ์จากการแปลงพฤติกรรมคู่ขนานข้ามภาคีสำหรับกรณี agent2 ในรูปที่ 4.13 อนึ่งป้ายชื่อสำหรับจุดยอดแทนฟังก์ชั่นจะแสดงโดยใช้ชื่อฟังก์ชั่นที่ปรากฏในรหัสเทียมโดยตรง

|  |
| --- |
|  |
| รูปที่ 4.14 ตัวอย่างกราฟผลลัพธ์จากการแปลงพฤติกรรมคู่ขนานข้ามภาคี |

gSpan จะถูกใช้ทำเหมืองข้อมูลบนชุดข้อมูลรูปแบบพฤติกรรมซึ่งถูกแปลงให้อยู่ในรูปพฤติกรรมคู่ขนานข้ามภาคีแล้ว กราฟย่อยที่พบบ่อยที่ได้จากขั้นตอนวิธีจะถูกคัด โดยกราฟย่อยที่มีเส้นเชื่อมติดป้ายชื่อทั้ง 0 และ 1 ซึ่งแสดงถึงพฤติกรรมคู่ขนานข้ามภาคีเท่านั้นจะถูกนำไปใช้งานต่อในการสร้างรูปแบบพฤติกรรมอัตโนมัติ

* + 1. การทำเหมืองข้อมูลสำหรับการเลือกใช้ฟังก์ชั่นข้อมูล

ในการอธิบายภาคีด้วยภาษาอธิบายภาคีฯจะมีการใช้งานฟังก์ชั่นการกระทำเพื่ออธิบายการกระทำและฟังก์ชั่นข้อมูลเพื่อใช้เป็นเงื่อนไขในโครงสร้างเงื่อนไข ซึ่งฟังก์ชั่นเหล่านี้มีพารามิเตอร์อีกจำนวนหนึ่งซึ่งอาจเติมสัญพจน์หรือฟังก์ชั่นข้อมูลที่มีชนิดข้อมูลตรงกันก็ได้ การเติมฟังก์ชั่นข้อมูลที่แตกต่างกันจะส่งผลต่อพฤติกรรมผลลัพธ์ของภาคีต่างกันออกไป การทำเหมืองข้อมูลสำหรับการเลือกใช้ฟังก์ชั่นข้อมูลจึงมีจุดประสงค์เพื่อค้นหาแบบแผนของฟังก์ชั่นข้อมูลที่ถูกใช้เป็นอากิวเมนท์สำหรับพารามิเตอร์ต่าง ๆ ของฟังก์ชั่นการกระทำและฟังก์ชั่นข้อมูลหลัก ด้วยจุดประสงค์ดังกล่าว ผลลัพธ์สำหรับการทำเหมืองข้อมูลในกรณีจึงต้องทำให้รู้ว่าแต่ละพารามิเตอร์ของฟังก์ชั่นหลักมีการใช้งานฟังก์ชั่นข้อมูลอะไร

ในที่นี้เลือกแปลงข้อมูลขาเข้าให้อยู่ในรูปของกราฟระบุชื่อแบบมีทิศทางไม่มีวัฏจักร ข้อมูลกราฟสามารถเก็บข้อมูลทั้งชื่อฟังก์ชั่นหลัก, ฟังก์ชั่นข้อมูลที่เลือกใช้ และตำแหน่งพารามิเตอร์ในฟังก์ชั่นหลักของแต่ละฟังก์ชั่นข้อมูล รวมถึงสามารถแสดงการเรียกใช้ฟังก์ชั่นซ้อนฟังก์ชั่นได้โดยไม่มีข้อมูลสูญหายอีกด้วย ทำให้กราฟย่อยจากการทำเหมืองข้อมูลเก็บข้อมูลที่ต้องการได้อย่างครบถ้วน

การแปลงฟังก์ชั่นหลักเป็นกราฟทำได้โดยสร้างจุดยอดรากที่มีป้ายชื่อเป็นตัวเลขแทนชื่อฟังก์ชั่นหลัก โดยตัวเลขแทนชื่อฟังก์ชั่นนี้จะแตกต่างจากที่ใช้ในการบีบอัดฟังก์ชั่นการกระทำในหัวข้อ 4.2.3 ในที่นี้ฟังก์ชั่นการกระทำและฟังก์ชั่นข้อมูลจะใช้ลำดับตัวเลขเดียวกัน ทำให้ฟังก์ชั่นทั้งสองประเภทไม่มีตัวเลขแทนชื่อซ้ำกัน โดยเริ่มต้นเลข 0 จากฟังก์ชั่นการกระทำแรก เมื่อสร้างจุดยอดรากแล้วจึงสร้างจุดยอดแทนอากิวเมนท์เท่ากับจำนวนฟังก์ชั่นข้อมูลที่ใช้เป็นอากิวเมนท์ของฟังก์ชั่นหลัก และกำหนดป้ายชื่อให้แต่ละจุดยอดแทนอากิวเมนท์ด้วยตัวเลขแทนชื่อฟังก์ชั่นข้อมูล จากนั้นจึงสร้างเส้นเชื่อมจากจุดยอดรากไปยังทุกจุดยอดแทนอากิวเมนท์โดยกำหนดป้ายชื่อให้กับเส้นเชื่อมด้วยตัวเลขแทนตำแหน่งพารามิเตอร์ของอากิวเมนท์นั้น ๆ กำหนดให้พารามิเตอร์ตำแหน่งแรกสุดแทนด้วยเลข 0 รูปที่ 4.15 แสดงตัวอย่างกราฟแทนความสัมพันธ์แบบการเลือกใช้ฟังก์ชั่น โดยภาษาอธิบายภาคีฯที่ปรากฏในตัวอย่างแสดงด้วยรหัสเทียม

|  |
| --- |
| A( B(), C(), 3.5, E() )  สมมติตัวเลขแทนฟังก์ชั่น  A: 10 B: 1 C: 2 E: 8  แปลง |
| รูปที่ 4.15 การแปลงฟังก์ชั่นหลัก A เป็นกราฟ |

สำหรับกรณีที่มีฟังก์ชั่นซ้อนฟังก์ชั่นให้ใช้วิธีแปลงในลักษณะเดียวกัน โดยให้ฟังก์ชั่นข้อมูลที่มีฟังก์ชั่นข้อมูลอื่นซ้อนอยู่ภายในเป็นฟังก์ชั่นหลักแทน แล้วสร้างจุดยอดแทนอากิวเมนท์พร้อมเส้นเชื่อมเพิ่มตามจำนวนฟังก์ชั่นข้อมูลซ้อนที่ปรากฏ รูปที่ 4.16 แสดงตัวอย่างการแปลงฟังก์ชั่นหลักโดยมีการใช้ฟังก์ชั่นซ้อนฟังก์ชั่น

|  |
| --- |
| แปลง  A( B( G() ), C( 5, D( 8, F() ) ), E() )  สมมติตัวเลขแทนฟังก์ชั่น  A: 10 B: 1 C: 2 D: 7 E: 8 F: 9 G: 15 |
| รูปที่ 4.16 การแปลงฟังก์ชั่นหลัก A ที่มีฟังก์ชั่นซ้อนฟังก์ชั่น |

ฟังก์ชั่นการกระทำและเงื่อนไขทั้งหมดที่พบในชุดข้อมูลลำดับพฤติกรรมจะถูกแปลงเป็นกราฟด้วยวิธีข้างต้นแล้วใช้ขั้นตอนวิธี gSpan เพื่อค้นหากราฟย่อยที่พบบ่อย กราฟย่อยที่ได้จะสามารถแปลงกลับเป็นฟังก์ชั่นพร้อมฟังก์ชั่นข้อมูลเพื่อนำไปใช้สร้างรูปแบบพฤติกรรมต่อไป พารามิเตอร์ที่ถูกเว้นว่างไว้จะถูกเติมเต็มในช่วงสร้างรูปแบบพฤติกรรม

* + 1. ข้อมูลสนับสนุน

ในหัวข้อนี้จะกล่าวถึงข้อมูลย่อยอื่น ๆ ที่จำเป็นในการสร้างรูปแบบพฤติกรรมของศัตรู โดยเรียกข้อมูลย่อยเหล่านี้ว่าข้อมูลสนับสนุน

* + - 1. ช่วงค่าของจำนวนความสัมพันธ์

ในขั้นตอนสร้างรูปแบบพฤติกรรมของศัตรูจะมีการนำเอาแบบแผนความสัมพันธ์รูปแบบต่าง ๆ ได้แก่ ลำดับฟังก์ชั่นการกระทำ, ฟังก์ชั่นการกระทำคู่ขนาน และการเลือกใช้ฟังก์ชั่นข้อมูล ที่ได้จากการทำเหมืองข้อมูลมาใช้ หากนำความสัมพันธ์ทั้งหมดที่พบมาใช้น่าจะทำให้ศัตรูมีแบบแผนทุกอย่าง ส่งผลให้ขาดเอกลักษณ์ อีกทั้งยังเป็นแนวคิดที่ไม่สมเหตุสมผล ในที่นี้จึงใช้วิธีเลือกความสัมพันธ์บางส่วนแทน คำถามที่ตามมา คือ จะเลือกความสัมพันธ์แต่ละชนิดอย่างไร เป็นจำนวนเท่าไร

สำหรับวิธีการเลือกความสัมพันธ์แต่ละรูปแบบ ในที่นี้จะใช้วิธีการสุ่มเลือกแบบถ่วงน้ำหนัก กล่าวคือ แบบแผนความสัมพันธ์แต่ละตัวที่ได้จากการทำเหมืองข้อมูลมีโอกาสถูกเลือกไม่เท่ากัน แต่ความน่าจะเป็นจะขึ้นกับความถี่ที่ความสัมพันธ์นั้นปรากฏในชุดข้อมูล ส่งผลให้แบบแผนความสัมพันธ์ที่ปรากฏบ่อยมีโอกาสถูกเลือกมากกว่า ซึ่งสามารถยกตัวอย่างวิธีคำนวณได้ดังนี้

|  |
| --- |
| กำหนดชุดข้อมูลขาเข้าสำหรับการทำเหมืองข้อมูลเพื่อหาความสัมพันธ์แบบลำดับฟังก์ชั่นการกระทำกรณีไม่เป็นลำดับพฤติกรรมเชื่อม  ซึ่งประกอบด้วยลำดับ  และรายการลำดับย่อยตัวอย่างที่พบบ่อยจากการทำเหมืองข้อมูล  ซึ่งแต่ละลำดับย่อยมีรายการ สำหรับเก็บลำดับที่บรรจุลำดับย่อยนั้น โอกาสที่  จะถูกเลือกคำนวณได้จาก  เมื่อ |

ในส่วนของจำนวนความสัมพันธ์แต่ละรูปแบบ ในที่นี้จะใช้วิธีสุ่มเลือกแบบถ่วงน้ำหนักเช่นเดียวกันว่าแต่ละรูปแบบความสัมพันธ์จะมีการนำเอาแบบแผนมาใช้เป็นจำนวนกี่แบบแผน แต่ละรูปแบบความสัมพันธ์จะมีการคำนวณความน่าจะเป็นของจำนวนแบบแผนดังนี้

|  |
| --- |
| กำหนดชุดข้อมูลรูปแบบพฤติกรรมของภาคี  และชุดข้อมูลรูปแบบพฤติกรรมที่ประกอบด้วยภาคีศัตรูเท่านั้น  ซึ่ง  และ  โอกาสที่รูปแบบความสัมพันธ์  ที่กำหนดจะถูกใช้เป็นจำนวน แบบแผน คำนวณได้จาก    เมื่อ มีค่าเท่ากับจำนวนแบบแผนที่เป็นความสัมพันธ์รูปแบบ ที่ถูกใช้ในภาคีศัตรู |

* + - 1. ช่วงค่าของพารามิเตอร์

ในขั้นตอนการทำเหมืองข้อมูลที่ผ่านมา ข้อมูลที่เป็นสัญพจน์จะถูกละออกจากชุดข้อมูลขาเข้าทั้งหมด ในขั้นตอนนี้จึงเป็นการหาช่วงค่าของพารามิเตอร์ที่ควรจะเป็นเพื่อให้ขั้นตอนสร้างรูปแบบพฤติกรรมใช้ในการเติมเต็มส่วนที่ขาดไปหลังจากใช้แบบแผนความสัมพันธ์ที่เลือกมาในการสร้างรูปแบบพฤติกรรมแล้ว

แต่ละฟังก์ชั่น (ทั้งการกระทำและข้อมูล) จะรับพารามิเตอร์ที่แตกต่างกันไปทั้งจำนวนของพารามิเตอร์และชนิดข้อมูล โดยแต่ละพารามิเตอร์จะส่งผลต่อตัวฟังก์ชั่นแตกต่างกันตามอากิวเมนท์ที่ส่งเข้ามา ข้อมูลช่วงค่าของพารามิเตอร์จึงถูกแยกตามแต่ละฟังก์ชั่นด้วย นั่นคือ ลักษณะของข้อมูลช่วงค่าของพารามิเตอร์จะอยู่ในรูปของชื่อฟังก์ชั่นและรายการของช่วงค่าพร้อมตำแหน่งพารามิเตอร์ที่ช่วงค่านั้นถูกใช้เป็นอากิวเมนท์ เช่น ฟังก์ชั่นสมมติ A มีพารามิเตอร์ที่ 0 เป็นข้อมูลชนิดตัวเลขซึ่งพบว่ามีค่าในช่วง [-1,50] เป็นต้น

การหาข้อมูลช่วงค่าดังกล่าวนี้สามารถทำได้โดยสำรวจทุกฟังก์ชั่นที่พบในชุดข้อมูลรูปแบบพฤติกรรม และบันทึกค่าทั้งหมดที่พบสำหรับทุกพารามิเตอร์ที่สนใจของทุกฟังก์ชั่น จากนั้นจึงสรุปข้อมูลช่วงค่าออกมา การบันทึกค่าทั้งหมดไว้จะช่วยให้สามารถคำนวณค่าทางสถิติอื่น ๆ เพิ่มเติมได้หากต้องการนำไปใช้งาน

สำหรับพารามิเตอร์ที่สนใจของแต่ละฟังก์ชั่น งานวิทยานิพนธ์นี้จะสนใจเฉพาะพารามิเตอร์ชนิดตัวเลขเท่านั้น

* 1. การสร้างรูปแบบพฤติกรรมของศัตรูอัตโนมัติ

ภาษาอธิบายภาคีฯได้กำหนดโครงสร้างของภาคีไว้ ซึ่งประกอบด้วยค่าคุณสมบัติ สถานะที่มีได้หลายสถานะโดยแต่ละสถานะมีลำดับพฤติกรรมได้หลายลำดับที่ทำงานขนานกัน รวมถึงการฟังก์ชั่นกระทำและฟังก์ชั่นข้อมูลซึ่งแต่ละฟังก์ชั่นรับพารามิเตอร์ที่มีชนิดข้อมูลต่างกันออกไป โดยในที่นี้ไม่ได้แสดงรายละเอียดของแต่ละฟังก์ชั่นไว้

ในขั้นตอนการทำเหมืองข้อมูลบนชุดข้อมูลพฤติกรรม จะได้ข้อมูลดังต่อไปนี้

* ความสัมพันธ์รูปแบบลำดับฟังก์ชั่นกระทำ
* ความสัมพันธ์รูปแบบลำดับเชื่อมของฟังก์ชั่นกระทำ เพื่อแสดงความสัมพันธ์เชิงลำดับแบบข้ามสถานะ
* ความสัมพันธ์รูปแบบฟังก์ชั่นคู่ขนาน
* ความสัมพันธ์รูปแบบฟังก์ชั่นคู่ขนานข้ามภาคี
* ความสัมพันธ์รูปแบบการเรียกใช้ฟังก์ชั่นข้อมูล
* ข้อมูลสนับสนุน 1: ช่วงค่าของจำนวนความสัมพันธ์แต่ละรูปแบบที่ปรากฏในภาคี
* ข้อมูลสนับสนุน 2: ช่วงค่าพารามิเตอร์สำหรับทุกฟังก์ชั่น

ข้อมูลจากขั้นตอนการทำเหมืองข้อมูลและโครงสร้างและรายละเอียดต่าง ๆ ของภาษาอธิบายภาคีฯจะนำมาใช้ร่วมกันเพื่อสร้างรูปแบบพฤติกรรมของศัตรู โดยเริ่มจากการเลือกความสัมพันธ์ต่าง ๆ สำหรับศัตรูที่จะถูกสร้างมาจำนวนหนึ่งโดยอิงจำนวนจากข้อมูลสนับสนุน 1 ความสัมพันธ์ที่ถูกเลือกมาทั้งหมดจะเรียกว่า เซ็ตความสัมพันธ์ อนึ่งการเลือกความสัมพันธ์นี้จะไม่เลือกความสัมพันธ์รูปแบบการเรียกใช้ฟังก์ชั่นข้อมูล จากนั้น สมาชิกในเซ็ตความสัมพันธ์จะถูกนำออกจากเซ็ตครั้งละ 1 ตัว และเติมความสัมพันธ์นั้นลงในโครงภาคีซึ่งเป็นภาคีที่อธิบายด้วยภาษาภาคีที่ยังไม่เสร็จสมบูรณ์ หากโครงภาคีที่มีอยู่ไม่สามารถรองรับความสัมพันธ์นั้นได้ ให้ทำการต่อเติมโครงจนกระทั่งสามารถใส่ความสัมพันธ์ลงไปได้ กระบวนการนี้จะทำต่อไปจนกระทั่งเซ็ตความสัมพันธ์กลับเป็นเซ็ตว่างอีกครั้ง จากนั้นจึงเติมเต็มพารามิเตอร์ที่ขาดหายไปโดยเลือกใช้ค่าอิงกับข้อมูลสนับสนุน 2 จนได้ภาคีที่เสร็จสมบูรณ์

รายละเอียดของคำศัพท์ต่าง ๆ ที่ปรากฏข้างต้นรวมถึงขั้นตอนในการสร้างรูปแบบพฤติกรรมเป็นไปดังนี้

* + 1. โครงภาคี

โครงภาคี คือ รหัสภาษาอธิบายภาคีฯสำหรับอธิบายภาคีหนึ่งที่ยังไม่สมบูรณ์ ฟังก์ชั่นยังมีอากิวเมนท์ไม่ครบถ้วน ความไม่สมบูรณ์ของโครงภาคีทั้งหมดจำเป็นต้องถูกเติมเต็มเพื่อให้ภาคีสมบูรณ์ ช่องว่างที่จะต้องถูกเติมเพื่อให้ภาคีสมบูรณ์เหล่านี้เรียกว่า สล็อตจำเป็น

นอกเหนือจากสล็อตจำเป็น โครงภาคียังมีช่องว่างอื่น ๆ ที่อนุญาตให้เติมรหัสลงไปได้อีก เช่น ช่องว่างก่อนฟังก์ชั่นแรกสุดของลำดับพฤติกรรมหรือช่องว่างระหว่างฟังก์ชั่นการกระทำ ซึ่งแทรกฟังก์ชั่นการกระทำใหม่ลงไปได้ หรือแม้กระทั่งภายในสถานะหนึ่ง ก็มีช่องว่างให้เพิ่มบล็อกลำดับใหม่ เป็นต้น ช่องว่างเหล่านี้ไม่ใช่ช่องว่างที่จำเป็นต้องเติมให้เต็ม แต่สามารถเพิ่มรหัสลงไปได้ ในที่นี้เรียกช่องว่างเหล่านี้ว่า สล็อตทางเลือก

นอกจากนี้ โครงภาคีในที่นี้ไม่จำเป็นจะต้องมีบล็อกภาคีเพียงบล็อกเดียวเสมอไป ภาคีศัตรูหนึ่งตัวสามารถมีภาคีที่เกี่ยวข้องอื่น ๆ ได้อีก เช่น ศัตรูที่สามารถยิงกระสุนได้ จะมีภาคีกระสุนมาเกี่ยวข้อง เป็นต้น ทำให้มีสล็อตทางเลือกสำหรับเพิ่มภาคีใหม่ด้วย รูปที่ 4.17 แสดงตัวอย่างโครงภาคีของศัตรู พร้อมสล็อตจำเป็นและสล็อตทางเลือก

|  |
| --- |
| .Sample{  .init{  **[OPT - Action]**  Set("texture", DynamicFilter("this"), 4);  **[OPT - Action]**  }  .state0{  .seq0{  **[OPT - Action]**  RunStraight(Get("direction", DynamicFilter("this")), **[CRIT]**, **[CRIT]**);  **[OPT - Action]**  FlipDirection(**[CRIT]**);  **[OPT - Action]**  Spawn(.Bullet)  **[OPT - Action]**  }  **[OPT – Sequence block]**  }  **[OPT – State block]**  }  .Bullet{  .init{  **[OPT - Action]**  Set("texture", DynamicFilter("this"), 3);  **[OPT - Action]**  Set("projectile", DynamicFilter("this"), true);  **[OPT - Action]**  }  .state0{  .seq0{  **[OPT - Action]**  Jump(Get("position", DynamicFilter("this")), **[CRIT]**, **[CRIT]**, false);  **[OPT - Action]**  }  **[OPT – Sequence block]**  }  **[OPT – State block]**  }  **[OPT – Agent block]** |
| รูปที่ 4.17 ตัวอย่างโครงภาคีและสล็อต โดย [CRIT] คือสล็อตจำเป็น และ [OPT] คือสล็อตทางเลือก |

* + 1. การเติมความสัมพันธ์

ในการสร้างรูปแบบพฤติกรรมอัตโนมัติ ความสัมพันธ์ที่ถูกเลือกออกมาจากเซ็ตความสัมพันธ์จะต้องถูกเติมลงในสล็อตที่มีอยู่ปัจจุบันของโครงภาคี วิธีการเลือกสล็อตที่จะถูกเติมความสัมพันธ์ทำได้โดยการแปลงปัญหาดังกล่าวนี้เป็นปัญหาตอบสนองเงื่อนไขจำกัด

ในขั้นแรกจะต้องทำการพิจารณาว่าความสัมพันธ์ดังกล่าวนี้สามารถเติมลงในโครงได้หรือไม่โดยการคำนวณจำนวนบล็อกต่าง ๆ ที่ความสัมพันธ์นี้ต้องการ หากไม่สามารถเติมได้จะต้องมีการต่อเติมโครงโดยการเพิ่มจำนวนบล็อกของโครงภาคีจนกระทั่งได้จำนวนบล็อกที่เพียงพอต่อการเติมความสัมพันธ์ วิธีการคำนวณจำนวนบล็อกสำหรับความสัมพันธ์แต่ละรูปแบบเป็นไปดังตารางที่ 4.5

|  |  |
| --- | --- |
| ตารางที่ 4.5 วิธีคำนวณจำนวนบล็อกที่จำเป็น | |
| รูปแบบความสัมพันธ์ | วิธีคำนวณ |
| ลำดับฟังก์ชั่นกระทำ | เติมลงในโครงได้เสมอ |
| ลำดับฟังก์ชั่นกระทำ (ข้ามสถานะ) | จำนวนบล็อกสถานะ = ข้อมูลเสริม |
| ฟังก์ชั่นคู่ขนาน | จำนวนบล็อกลำดับของสถานะ >= จำนวนจุดยอดเพิ่มเติม |
| ฟังก์ชั่นคู่ขนานข้ามภาคี | จำนวนบล็อกภาคี >= 2  จำนวนบล็อกลำดับของสถานะหนึ่งของภาคีเจ้าของ >= จำนวนจุดยอดเพิ่มเติมที่มีป้ายชื่อเส้นเชื่อมเป็น 0  ต้องมีภาคีอื่นที่สถานะเริ่มต้นมีบล็อกลำดับ = ข้อมูลเสริม หากไม่มีข้อมูลเสริม ภาคีอื่นนั้นต้องมีบล็อกลำดับเป็นจำนวน >= จำนวนจุดยอดเพิ่มเติมที่มีป้ายชื่อเส้นเชื่อมเป็น 1 |

เมื่อทำการเติมบล็อกจนครบถ้วนตามที่จำเป็นสำหรับความสัมพันธ์แล้ว จึงแปลงปัญหาการเติมความสัมพันธ์นี้เป็นปัญหาตอบสนองเงื่อนไขจำกัด โดยวิธีการแตกต่างกันไปตามรูปแบบความสัมพันธ์

* ลำดับฟังก์ชั่นการกระทำ
  + สุ่มเลือกภาคี แล้วสุ่มเลือกบล็อกลำดับหนึ่งของภาคี
  + เซ็ตตัวแปร ประกอบด้วยตัวแปร  โดย จำนวนฟังก์ชั่น
  + โดเมนสำหรับแต่ละตัวแปร คือ เซ็ตของตัวเลขแทนสล็อตทางเลือกของบล็อกลำดับที่สุ่มมา โดยตัวเลขแทนสล็อตเริ่มจากเลข 0 ซึ่งแทนสล็อตแรกสุด
  + เงื่อนไขจำกัด คือค่าของตัวแปรจะต้องเป็นดังนี้ 
* ลำดับฟังก์ชั่นการกระทำ (ข้ามสถานะ)
  + สุ่มเลือกภาคีซึ่งมีจำนวนบล็อกสถานะเท่ากับข้อมูลเสริม แล้วสุ่มเลือก 2 บล็อกสถานะ  และ  ของภาคีโดย 
  + เซ็ตตัวแปร ประกอบด้วยตัวแปร  และ  โดย จำนวนฟังก์ชั่นของลำดับพฤติกรรมต้นทาง และ จำนวนฟังก์ชั่นของลำดับพฤติกรรมปลายทาง
  + โดนเมนสำหรับตัวแปร  คือเซ็ตของตัวเลขแทนสล็อตทางเลือกของบล็อกลำดับที่สุ่มมาจากบล็อกสถานะ  และโดเมนสำหรับตัวแปร  คือเซ็ตของตัวเลขแทนสล็อตทางเลือกของบล็อกลำดับที่สุ่มมาจากบล็อกสถานะ 
  + เงื่อนไขจำกัด คือค่าของตัวแปรจะต้องเป็นดังนี้  และ 
* ฟังก์ชั่นคู่ขนาน
  + สุ่มเลือกภาคี แล้วสุ่มเลือกบล็อกสถานะของภาคีที่มีจำนวนบล็อกลำดับมากกว่าหรือเท่ากับจำนวนจุดยอดเพิ่มเติม แล้วสุ่มเลือกบล็อกลำดับมา  บล็อก () เมื่อ จำนวนจุดยอดเพิ่มเติม
  + เซ็ตตัวแปร เป็นเซ็ตซึ่งเท่ากับ  โดยแต่ละเซ็ต  จะมีจำนวนสมาชิกเท่ากับจำนวนจุดยอดแทนฟังก์ชั่นของจุดยอดเพิ่มเติม 
  + โดเมนสำหรับตัวแปรในแต่ละเซ็ต  คือเซ็ตของตัวเลขแทนสล็อตทางเลือกของบล็อกลำดับ 
  + ไม่มีเงื่อนไขจำกัด
* ฟังก์ชั่นคู่ขนานข้ามภาคี
  + สุ่มเลือกภาคี  และ  โดย  ซึ่ง
    - มีอย่างน้อย 1 สถานะของภาคี  ที่มีจำนวนบล็อกลำดับมากกว่าหรือเท่ากับ  เมื่อ จำนวนจุดยอดเพิ่มเติมที่มีป้ายชื่อเส้นเชื่อมเป็น 0
    - สถานะเริ่มต้นของภาคี  มีจำนวนบล็อกลำดับเท่ากับข้อมูลเสริม หากไม่มีข้อมูลเสริม จำนวนบล็อกลำดับจะต้องมีค่ามากกว่าหรือเท่ากับจำนวนจุดยอดเพิ่มเติมที่มีป้ายชื่อเส้นเชื่อมเป็น 1 (กำหนดให้ จำนวนบล็อกดังกล่าวนี้)
  + สุ่มเลือกสถานะของภาคี  ที่มีจำนวนบล็อกลำดับมากกว่าหรือเท่ากับ  แล้วสุ่มเลือกบล็อกลำดับมา บล็อก ()
  + สุ่มเลือกบล็อกลำดับจากสถานะเริ่มต้นของภาคี  มา  บล็อก ()
  + เซ็ตตัวแปร เป็นเซ็ตซึ่งเท่ากับ  โดยแต่ละเซ็ต  จะมีจำนวนสมาชิกเท่ากับจำนวนจุดยอดแทนฟังก์ชั่นของจุดยอดเพิ่มเติม  และแต่ละเซ็ต  จะมีจำนวนสมาชิกเท่ากับจำนวนจุดยอดแทนฟังก์ชั่นของจุดยอดเพิ่มเติม 
  + โดเมนสำหรับตัวแปรในแต่ละเซ็ต  คือเซ็ตของตัวเลขแทนสล็อตทางเลือกของบล็อกลำดับ  และโดเมนสำหรับตัวแปรในแต่ละเซ็ต  คือเซ็ตของตัวเลขแทนสล็อตทางเลือกของบล็อกลำดับ 
  + ไม่มีเงื่อนไขจำกัด

เมื่อกำหนดตัวแปร โดเมน และเงื่อนไขจำกัดได้แล้ว จะสามารถสร้างแบบจำลองของปัญหาการตอบสนองเงื่อนไขจำกัดและแก้ปัญหาด้วย JaCoP ได้ ซึ่งผลลัพธ์จากการแก้ปัญหาจะทำให้ทราบว่าแต่ละตัวแปรในเซ็ตตัวแปรมีค่าเป็นเท่าใด จากนั้นจึงนำเอาฟังก์ชั่นที่คู่กับตัวแปรไปแทรกลงในสล็อตที่กำหนดโดยค่าของตัวแปร เมื่อแทรกฟังก์ชั่นทั้งหมดของความสัมพันธ์ครบถ้วนแล้ว จะนับได้ว่าความสัมพันธ์ถูกแทรกลงในโครงภาคีเรียบร้อยแล้ว

หลังจากความสัมพันธ์ทั้งหมดในเซ็ตความสัมพันธ์ถูกแทรกลงในโครงภาคีทั้งหมดแล้ว ทุกฟังก์ชั่นที่ยังมีสล็อตอยู่จะถูกเติมโดยใช้ความสัมพันธ์รูปแบบการเรียกใช้ฟังก์ชั่นข้อมูลเพื่อเลือกฟังก์ชั่นข้อมูลมาใช้เป็นอากิวเมนท์ ความสัมพันธ์รูปแบบนี้จะระบุว่าสำหรับแต่ละพารามิเตอร์ของแต่ละฟังก์ชั่น มีการใช้ฟังก์ชั่นข้อมูลใดเป็นอากิวเมนท์

เมื่อเติมสล็อตด้วยฟังก์ชั่นข้อมูลแล้ว สุดท้ายจะมีสล็อตจำเป็นที่ยังว่างอยู่ งานวิทยานิพนธ์นี้อาศัยการเลือกค่าจากข้อมูลสนับสนุน 2 เพื่อเติมเต็มสล็อตจำเป็น ข้อมูลสนับสนุน 2 จะระบุว่าสำหรับแต่ละพารามิเตอร์ของแต่ละฟังก์ชั่น ช่วงค่าที่เป็นไปได้อิงชุดข้อมูลรูปพฤติกรรมเป็นเท่าใด

* 1. การทดลองและวัดผล

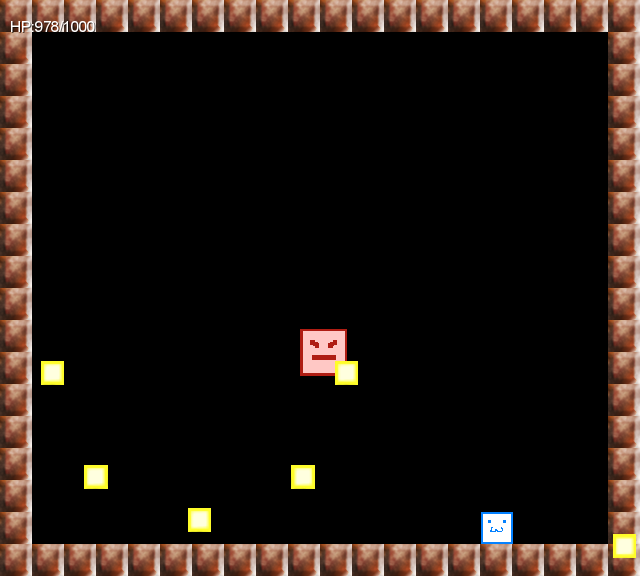
การวัดผลรูปแบบพฤติกรรมจะทำโดยใช้เกมทดสอบ โดยให้ผู้เล่นทดลองต่อสู้กับศัตรูในเกมทดสอบและตอบแบบสอบถามที่เกี่ยวข้องกับการยอมรับได้ของรูปแบบพฤติกรรม

* + 1. เกมทดสอบ

เกมทดสอบที่สร้างขึ้นจะมีการใช้งานตัวแปลภาษาสำหรับภาษาอธิบายภาคีฯ กลไกของตัวเกมจะถูกกำหนดให้เป็นไปตามองค์ประกอบที่เกี่ยวข้องกับศัตรู

* โครงสร้างพื้นที่: เกมทดสอบใช้ห้องสี่เหลี่ยมซึ่งมีกำแพงสองด้าน พื้น และเพดานเป็นบริเวณที่ผ่านไม่ได้ โดยใช้ตัวตรวจจับการชนเพื่อตรวจสอบว่าภาคีที่กำหนดสามารถเคลื่อนผ่านบริเวณที่กำหนดได้หรือไม่ โดยมีค่าคุณสมบัติ “ผ่านได้” เข้ามาเกี่ยวข้อง นอกจากนี้ โครงสร้างพื้นที่จะมีค่าแรงดึงดูดของพื้นที่คอยดึงให้ภาคีตกลงมาตามแรงโน้มถ่วงโดยขึ้นกับค่าคุณสมบัติ “ผลของแรงดึงดูด” ของภาคี
* กลไกการต่อสู้: เกมทดสอบใช้ตัวตรวจจับการชนร่วมกับค่าคุณสมบัติ “กลุ่ม”, “ฝ่ายโจมตี” และ “ฝ่ายตั้งรับ” สำหรับตรวจจับการโจมตีที่เกิดขึ้น และสร้างความเสียหายโดยมีค่าคุณสมบัติ “พลังชีวิต”, “พลังโจมตี” เข้ามาเกี่ยวข้อง รวมถึงมีกลไกสลายตัวสำหรับภาคีกระสุน กล่าวคือ เกมทดสอบกำหนดกลไกการต่อสู้ตามที่ได้กล่าวถึงในส่วนของบทบาทของค่าคุณสมบัติของศัตรูในหัวข้อ 4.1.3.1
* ตัวละครอวตาร: ตัวละครอวตารในเกมทดสอบสามารถเคลื่อนที่ในโลกของเกมโดยเดินไปทางซ้ายหรือขวาเมื่อผู้เล่นกดปุ่มซ้ายหรือขวาตามลำดับ เมื่อผู้เล่นกดปุ่ม Z จะเป็นการสั่งให้ตัวละครกระโดดซึ่งจะได้รับผลจากแรงดึงดูดของโครงสร้างพื้นที่เช่นเดียวกับภาคีอื่น ๆ ผู้เล่นสามารถสั่งให้ตัวละครอวตารโจมตีโดยใช้กระสุนโดยการกดปุ่ม X ได้

เกมทดสอบสำหรับงานวิทยานิพนธ์นี้สร้างขึ้นโดยใช้ libgdx ซึ่งเป็นข่ายงานสำหรับพัฒนาเกมข้ามแพลทฟอร์มด้วยภาษาจาวา ตัวอย่างเกมทดสอบเป็นไปดังรูปที่ 4.18



รูปที่ 4.18 เกมทดสอบ

* + 1. การทดลองและแบบสอบถามที่คาดว่าจะใช้

ในการทดลองจะให้ผู้ทดสอบทำการต่อสู้กับศัตรูที่ได้จากการสร้างรูปแบบพฤติกรรมอัตโนมัติและตอบแบบสอบถามเพื่อวัดว่าพฤติกรรมของศัตรูยอมรับได้หรือไม่ โดยพฤติกรรมของศัตรูที่ยอมรับได้นั้น จะต้องเป็นไปในลักษณะดังนี้

* ศัตรูพยายามโจมตีผู้เล่น และ
* ผู้เล่นจะต้องสามารถโต้ตอบหรือหลีกเลี่ยงพฤติกรรมของศัตรูได้

คำถามที่เกี่ยวข้องกับการยอมรับได้จึงประกอบด้วย 2 คำถามข้างต้น ซึ่งคำตอบอยู่ในรูปใช่หรือไม่ใช่ รูปแบบพฤติกรรมของศัตรูที่ได้รับคำตอบใช่จากทั้ง 2 คำถามจึงนับว่าเป็นรูปแบบพฤติกรรมที่ยอมรับได้

* + 1. การวัดผล

ขั้นตอนวิธีที่นำเสนอในงานวิทยานิพนธ์นี้เป็นขั้นตอนวิธีที่ไม่มีการใช้อีวาลูเอชั่นฟังก์ชั่น (Evaluation function) ในการคัดศัตรูผลลัพธ์ที่ไม่สามารถยอมรับได้ออกไป นั่นหมายความว่า หากขั้นตอนวิธีนี้สมบูรณ์แบบ ศัตรูผลลัพธ์ที่ได้จากขั้นตอนวิธีจะต้องยอมรับได้เสมอ ในที่นี้จะทำการวัดผลโดยการสร้างศัตรูจำนวนหนึ่ง และให้ผู้เล่นทำการต่อสู้กับศัตรูแล้วตอบแบบสอบถาม ผู้ทำวิทยานิพนธ์มีความคาดหวังว่า แม้ขั้นตอนวิธียังไม่สามารถหวังผลได้สมบูรณ์แบบ ก็ควรจะให้ผลที่ยอมรับได้เป็นส่วนใหญ่ นั่นคือ รูปแบบพฤติกรรมของศัตรูเป็นที่ยอมรับได้เกินกว่าครึ่งหนึ่งของศัตรูทั้งหมดที่สร้างขึ้นมา

นอกจากนี้ งานวิทยานิพนธ์นี้จะทำการทดสอบความถูกต้องของขั้นตอนวิธีโดยเทียบกับศัตรูที่สร้างขึ้นด้วยขั้นตอนวิธีแบบสุ่มโดยไม่พึ่งข้อมูลจากการทำเหมืองข้อมูล ซึ่งเป็นการสุ่มสร้างรูปแบบพฤติกรรมโดยไม่ใช้ข้อมูลความสัมพันธ์ที่ได้จากการทำเหมืองข้อมูล โดยที่รูปแบบพฤติกรรมที่สร้างจะต้องทำงานได้ กล่าวคือ ทุกสล็อตจำเป็นจะต้องถูกเติมเต็ม การทดสอบความถูกต้องขั้นตอนวิธีแบบสุ่มจะทำโดยสร้างศัตรูจำนวนเท่ากับจำนวนของศัตรูที่สร้างขึ้นด้วยขั้นตอนวิธีที่นำเสนอ จากนั้นให้ผู้เล่นกลุ่มเดียวกันทำการต่อสู้กับศัตรูที่สร้างขึ้นโดยวิธีแบบสุ่ม โดยผู้เล่นแต่ละคนจะต้องต่อสู้กับศัตรูจำนวนเท่ากับจำนวนศัตรูที่ได้ต่อสู้ด้วยในช่วงทดสอบขั้นตอนวิธีที่นำเสนอ สุดท้ายจึงทำการคำนวณร้อยละของศัตรูที่ยอมรับได้สำหรับกรณีขั้นตอนวิธีแบบสุ่มเพื่อใช้เทียบกับผลลัพธ์ของขั้นตอนวิธีที่นำเสนอ

1. วัตถุประสงค์

เพื่อหาแนวทางการสร้างรูปแบบพฤติกรรมสำหรับศัตรูในเกมแอ็คชั่นสองมิติเน้นตัวละครโดยอัตโนมัติที่ผู้เล่นยอมรับได้

1. ขอบเขตการดำเนินงาน
   1. การสร้างรูปแบบพฤติกรรมจะไม่นำลักษณะของพื้นที่เข้ามาพิจารณา
   2. ชุดข้อมูลรูปแบบพฤติกรรมทำซ้ำรูปแบบพฤติกรรมของศัตรูบางส่วนจากเกม Metroid, Super C, Megaman, Megaman 4 และ Shovel Knight
   3. เกมทดสอบที่ใช้จะเป็นเกมแพลทฟอร์ม
   4. รูปร่างของศัตรูที่ปรากฏในเกมทดสอบจะถูกกำหนดโดยมนุษย์และไม่เกี่ยวข้องกับการสร้างรูปแบบพฤติกรรม
2. ขั้นตอนการดำเนินงาน
   1. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง
   2. วิเคราะห์เกมเพื่อจัดทำแบบจำลองภาคีฯ
   3. พัฒนาตัวแปลภาษาอธิบายภาคีฯและเกมทดสอบ
   4. จัดเตรียมและปรับปรุงขั้นตอนวิธีสำหรับการทำเหมืองข้อมูล
   5. ทดสอบความถูกต้องขั้นตอนวิธีทำเหมืองข้อมูล
   6. สร้างชุดข้อมูลรูปแบบพฤติกรรม
   7. พัฒนาขั้นตอนวิธีสำหรับแปลงความสัมพันธ์รูปแบบต่าง ๆ และข้อมูลสนับสนุนเป็นปัญหาการตอบสนองเงื่อนไขบังคับ
   8. ทดลองสร้างรูปแบบพฤติกรรมด้วยขั้นตอนวิธีที่นำเสนอ และเก็บข้อมูลการทดสอบกับผู้เล่น
   9. วิเคราะห์ผลการทดลอง
   10. สรุปผลการวิจัยและจัดทำวิทยานิพนธ์
3. ประโยชน์ที่คาดว่าจะได้รับ
   1. ได้แบบจำลองภาคีฯสำหรับอธิบายภาคีในเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร
   2. ค้นพบแบบแผนสำหรับออกแบบรูปแบบพฤติกรรมศัตรูสำหรับเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร
   3. ได้ขั้นตอนวิธีสำหรับสร้างรูปแบบพฤติกรรมศัตรูอัตโนมัติสำหรับเกมแอ็คชั่นสองมิติแบบเน้นตัวละคร

**รายการอ้างอิง**

**ภาคผนวก**

**ประวัติผู้เขียนวิทยานิพนธ์**

ประวัติผู้เขียนวิทยานิพนธ์

[1] PwC. (2014). *Global entertainment and media outlook 2014-2018*.

[2] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *Computational Intelligence and AI in Games, IEEE Transactions on,* vol. 3, pp. 172-186, 2011.

[3] E. Adams, *Fundamentals of Game Design*, 2nd ed. vol. 13: Pearson Education Inc, 2010.

[4] S. Rogers, *LEVEL UP! The Guide to Great Video Game Design* vol. 1: John Wiley & Sons, Ltd, 2010.

[5] M. Davies, *Designing Character-based Console Games*, 1st ed. vol. 7-10, 14: Thompson Learning Inc, 2007.

[6] J. Han, M. Kamber, and J. Pei, *Data Mining: Concepts and Techniques*, 2nd ed. vol. 1, 8, 9: Morgan Kaufmann, 2005.

[7] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, 1995, pp. 3-14.

[8] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen*, et al.*, "Mining sequential patterns by pattern-growth: The prefixspan approach," *Knowledge and Data Engineering, IEEE Transactions on,* vol. 16, pp. 1424-1440, 2004.

[9] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, 2002, pp. 721-724.

[10] (2011, 11/6/2015). *ParSeMiS*. Available: https://www2.informatik.uni-erlangen.de/EN/research/zold/ParSeMiS/index.html

[11] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. vol. 6: Pearson Education Inc, 2009.

[12] A. A. Bulatov, "Complexity of conservative constraint satisfaction problems," *ACM Transactions on Computational Logic (TOCL),* vol. 12, p. 24, 2011.

[13] (2015, 11/6/2015). *Java constraint solver (JaCoP)*. Available: http://www.jacop.eu/

[14] C. P. McKay, "What is life-and how do we search for it in other worlds?," *PLoS Biology,* vol. 2, pp. 1260-1262, 2004.

[15] T. Miconi and A. Channon, "An improved system for artificial creatures evolution," in *Procs 10th Intl Conf on Simulation and Synthesis of Living Systems (ALIFE X)*, 2006.

[16] T. Miconi, "Evosphere: evolutionary dynamics in a population of fighting virtual creatures," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 3066-3073.

[17] N. Djezzar, N. Djedi, S. Cussat-Blanc, H. Luga, and Y. Duthen, "L-systems and artificial chemistry to develop digital organisms," in *Artificial Life (ALIFE), 2011 IEEE Symposium on*, 2011, pp. 225-232.

[18] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM),* vol. 9, p. 1, 2013.

[19] J. L. Hidalgo, E. Camahort, F. Abad, and M. J. Vicent, "Procedural graphics model and behavior generation," in *Computational Science–ICCS 2008*, ed: Springer, 2008, pp. 106-115.

[20] E. J. Hastings, R. K. Guha, and K. O. Stanley, "Automatic content generation in the galactic arms race video game," *Computational Intelligence and AI in Games, IEEE Transactions on,* vol. 1, pp. 245-263, 2009.

[21] M. J. Nelson and M. Mateas, "Towards automated game design," in *AI\* IA 2007: Artificial Intelligence and Human-Oriented Computing*, ed: Springer, 2007, pp. 626-637.

[22] M. Sicart, "Defining game mechanics," *Game Studies,* vol. 8, pp. 1-14, 2008.

[23] L. Pulsipher. (2012, 11/6/2015). *Game descriptions, rules, and mechanics: what are the differences and similarities?* Available: http://www.gamasutra.com/blogs/LewisPulsipher/20120109/90875/Game\_descriptions\_rules\_and\_mechanics\_what\_are\_the\_differences\_and\_similarities.php

[24] J. Togelius and J. Schmidhuber, "An experiment in automatic game design," in *Computational Intelligence and Games, 2008. CIG'08. IEEE Symposium On*, 2008, pp. 111-118.

[25] A. M. Smith and M. Mateas, "Variations forever: Flexibly generating rulesets from a sculptable design space of mini-games," in *Computational Intelligence and Games (CIG), 2010 IEEE Symposium on*, 2010, pp. 273-280.

[26] M. Cook, S. Colton, and J. Gow, "Initial results from co-operative co-evolution for automated platformer design," in *Applications of Evolutionary Computation*, ed: Springer, 2012, pp. 194-203.

[27] A. Zook and M. O. Riedl, "Generating and adapting game mechanics," in *Proceedings of the 2014 Foundations of Digital Games Workshop on Procedural Content Generation in Games, Ft. Lauderdale, Florida*, 2014.

[28] T. Mahlmann, "Modelling and generating strategy games mechanics," Ph. D. dissertation, ITU Copenhagen, 2012.

[29] D. Boutros. (2006, 11/6/2015). *A Detailed Cross-Examination of Yesterday and Today's Best-Selling Platform Games*. Available: http://www.gamasutra.com/view/feature/1851/a\_detailed\_cros-sexamination\_of\_.php

[30] K. Hullett and J. Whitehead, "Design patterns in FPS levels," in *proceedings of the Fifth International Conference on the Foundations of Digital Games*, 2010, pp. 78-85.

[31] M. Nelson. (2007, 11/6/2015). *Breaking Down Breakout: System And Level Design For Breakout-style Games*. Available: http://www.gamasutra.com/view/feature/1630/breaking\_down\_breakout\_system\_and\_.php

[32] G. Smith, J. Whitehead, M. Mateas, M. Treanor, J. March, and M. Cha, "Launchpad: A rhythm-based level generator for 2-d platformers," *Computational Intelligence and AI in Games, IEEE Transactions on,* vol. 3, pp. 1-16, 2011.

[33] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C.-W. Wu, and V. S. Tseng, "SPMF: a Java open-source pattern mining library," *The Journal of Machine Learning Research,* vol. 15, pp. 3389-3393, 2014.

[34] M. Verma and D. Mehta, "Sequential Pattern Mining: A Comparison between GSP, SPADE and Prefix SPAN," in *International Journal of Engineering Development and Research*, 2014.

[35] T. Ishio, T. Miyake, and K. Inoue, "Mining coding patterns to detect crosscutting concerns in java programs," in *Reverse Engineering, 2008. WCRE'08. 15th Working Conference on*, 2008, pp. 123-132.

1. เฟรม (Frame) คือ หน่วยนับเวลาประมวลผลเกม โดยการทำงาน 1 รอบรับนับเป็น 1 เฟรม การทำงาน 1 รอบนั้นประกอบด้วยการประมวลผลตรรกะและการสั่งวาด [↑](#footnote-ref-1)