การพัฒนาโกสเอไอและเทสเบดสำหรับเกมการต่อสู้โดยใช้อีมูเลเตอร์

นาย วรพจน์ ธัญภัทรกุล

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2550

GHOST AI AND TESTBED DEVELOPMENT FOR FIGHTING GAME USING EMULATOR

Mr. Worapoj Thanpattarakul

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

Department of Computer Engineering
Faculty of Engineering

Academic Year 2007

หัวข้อวิทยานิพนธ์	การพัฒนาโกสเอไอและเทสเบดสำหรับเกมการต่อสู้โดยใช้อีมูเลเตอร์
โดย	นาย วรพจน์ ธัญภัทรกุล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.วิษณุ โคตรจรัส

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... รักษาการแทนคณบดี คณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ยิ่งยศ เขมะโยธิน)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... อาจารย์ที่ปรึกษา

(อาจารย์ ดร.วิษณุ โคตรจรัส)

..... กรรมการ

(อาจารย์ ดร.เศรษฐา ปานงาม)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุรพงษ์ เลิศสิทธิชัย)

วรพจน์ ธัญภัทรกุล : การพัฒนาโกสเอไอและเทสเบดสำหรับเกมการต่อสู้โดยใช้อีมูเลเตอร์. (Ghost AI and Testbed Development for Fighting Game Using Emulator) อ. ที่ปรึกษา: อ.ดร.วิษณุ โคตรจรัส 105 หน้า.

โกสเอไอคือฟังก์ชันสำหรับเกมการต่อสู้ที่ใช้ควบคุมตัวละครคอมพิวเตอร์เพื่อเลียนแบบรูปแบบการเล่นของผู้เล่น ใช้เพื่อให้ผู้เล่นใดๆ เสมือนที่จะสามารถเล่นแข่งกับผู้เล่นเจ้าของโกสเอไอนั้นได้โดยไม่ต้องเจอตัวกัน ปัจจุบันยังไม่มีงานวิจัยที่เปิดเผยเทคนิคของโกสเอไอ เป้าหมายของงานวิทยานิพนธ์นี้จึงเป็นการค้นคว้าเทคนิคและวิธีการในการสร้างโกสเอไอ นอกจากนี้ในปัจจุบันยังไม่มีสภาพแวดล้อมสำหรับทดสอบปัญญาประดิษฐ์หรือที่เรียกว่าเทสเบดสำหรับเกมการต่อสู้ ดังนั้นงานวิทยานิพนธ์นี้จึงได้ทำการพัฒนาเทสเบดสำหรับทดสอบโกสเอไอบนเกมการต่อสู้ขึ้นมาด้วย งานวิทยานิพนธ์นี้นำอีมูเลเตอร์ของเครื่องเกมบอยแอดวานซ์มาดัดแปลงเป็นเทสเบดโดยการเพิ่มเติมส่วนเกมสเตทออบเซิร์ฟเวอร์ที่ใช้สำหรับอ่านค่าข้อมูลสถานการณ์ของเกมและส่วนเอไอโมดูลสำหรับใส่เอไอที่จะใช้ทดสอบลงไปแทนที่การควบคุมของผู้เล่น ทั้งนี้เพื่อให้สามารถทดลองกับเกมการต่อสู้ที่มีจำหน่ายในท้องตลาดจริง โดยเลือกใช้เกมสตรีทไฟเตอร์ซีโร่สามเป็นกรณีศึกษา ผู้เขียนใช้หลักการของเคสเบสรีซันนิงในการพัฒนาวิธีการค้นหา จัดกลุ่ม และจัดเก็บการตัดสินใจกระทำการต่างๆ ของผู้เล่นในแต่ละสถานการณ์และสร้างเป็นเคสขึ้นมา เพื่อใช้อ้างอิงและลอกแบบยามที่ตัวละครโกสเอไอเจอสถานการณ์เดียวกัน จากผลการทดลอง โกสเอไอที่ได้มีรูปแบบการเล่นที่บ่งบอกได้ว่ามีการออกท่าและตัดสินใจตอบสนองต่อเหตุการณ์ต่างๆ ตามแบบเดียวกับผู้เล่นต้นแบบ นอกจากนี้รูปแบบวิธีการสร้างโกสเอไอที่นำเสนอนั้น เป็นรูปแบบที่ไม่เฉพาะเจาะจงกับตัวเกม ดังนั้นจึงสามารถนำวิธีการที่นำเสนอโดยวิทยานิพนธ์นี้ไปใช้สร้างโกสเอไอสำหรับเกมการต่อสู้อื่นๆ ได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....วรพจน์ ธัญภัทรกุล.....
สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา 2550

4870698421 : MAJOR Computer Engineering
KEY WORD: TESTBED/ GHOST AI/ FIGHTING GAME/ EMULATOR

WORAPOJ THANPUTTARAKUL: GHOST AI AND TESTBED DEVELOPMENT FOR FIGHTNG GAME USING EMULATOR. THESIS ADVISOR: DR. VISHNU KOTRAJARAS, Ph.D., 105 pp.

A Ghost AI is a function in fighting game that controls a computer AI to imitate a player's playing style. When a player plays with a ghost AI of another player, he should feel like playing against the ghost AI's owner. The ghost AI owner does not have to be present at all. There is no research revealing techniques for ghost AIs creation. The aim of this thesis is to find out how to develop a ghost AI system for fighting games. In addition, there exists no testbed suitable for testing AI in fighting games. Therefore the testbed for testing ghost AIs is also implemented. A GameboyAdvance emulator is modified to be the new testbed in order to allow tests to be conducted on real commercial games. By implementing Game state observer and AI module into the emulator, the situation of the game can be known and researcher's AI module can control the game's character instead of the player. Street fighter zero 3 is used as the case study. The thesis use concept of case base reasoning and proposes methodologies for collecting, grouping and sorting a player's decisions for each situation. This information is looked up by a ghost AI when similar situation in the game arise in order to provide correct imitations. Our experimental result shows that each ghost AI created makes decisions and performs actions similar to its human counterpart in the same situation. The proposed ghost AI creation methodologies are not specific to certain games, therefore they can also be used in other fighting games.

Department Computer Engineering Student's signature.....วรพจ ธนัทพงศ์.....
Field of study Computer Engineering Advisor's signature.....
Academic year 2007

กิตติกรรมประกาศ

ขอขอบพระคุณ บิดา มารดา และครอบครัว ที่เป็นกำลังใจสำคัญ และให้ความช่วยเหลือในทุกๆ ด้าน จนผู้วิจัยสามารถทำวิทยานิพนธ์ฉบับนี้เสร็จ

ขอขอบพระคุณอาจารย์ที่ปรึกษา อ.ดร.วิษณุ โคตรจรัส ซึ่งเป็นผู้ให้ข้อคิด แนวทาง และคำปรึกษา ตลอดจนเป็นผู้ตรวจทานแก้ไข จนทำให้วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วง รวมถึงให้โอกาสและสิ่งที่ดีแก่ผู้วิจัยเสมอมา ผู้เขียนมีความประทับใจมากเมื่อครั้งที่ อาจารย์ที่ปรึกษาสามารถให้คำตอบในปัญหาที่ตัวละคร “เก็น” ในเกมสตรีทไฟเตอร์ซีโร่สาม มีหมายเลขประจำตัวละครถึงสองหมายเลขในตัวละครเดียวกันได้ เนื่องจากเป็นตัวละครที่สามารถเปลี่ยนรูปแบบท่าทางได้สองแบบเสมือนหนึ่งเป็นสองตัวละครในร่างเดียว หากไม่เป็นเพราะคำแนะนำของอาจารย์ที่ปรึกษาแล้วผู้เขียนคงไม่สามารถหาคำตอบของปัญหานี้ได้

ขอขอบพระคุณคณะกรรมการสอบวิทยานิพนธ์ ผศ.ดร. วิวัฒน์ วัฒนาวุฒิ อ.ดร. เศรษฐา ปานงาม และ ผศ.ดร. สุรพงษ์ เลิศสิทธิชัย ที่ได้กรุณาให้คำแนะนำในการแก้ไขวิทยานิพนธ์ให้มีคุณภาพยิ่งขึ้น วิทยานิพนธ์ฉบับนี้ไม่อาจจะสำเร็จได้หากไม่ได้รับความร่วมมือจากทุกท่าน และขอขอบคุณสมาชิกห้องปฏิบัติการซีพีเกมรวมถึงเพื่อนๆ ทุกคนผู้ที่ให้คำแนะนำเพิ่มเติมและช่วยเหลือในการทดลองเสมอมา

ขอขอบคุณนักพัฒนาเกมทั้งหลายที่สร้างสรรค์เกมอันสนุกสนาน และเป็นแรงบันดาลใจให้แก่การทำวิทยานิพนธ์เกี่ยวกับเกมเล่มนี้

และท้ายสุดนี้ขอขอบคุณกลุ่มผู้พัฒนาวิชวลบอยแอดวานซ์และวิชวลบอยแอดวานซ์ลิงค์อีมูเลตอร์ รวมถึงนักพัฒนาโปรแกรมและนักเขียนบทความทุกคนบนโลกอินเตอร์เน็ทที่แบ่งบันความรู้และประสบการณ์ต่างๆ แก่ผู้อื่นอย่างเหนื่อยโดยไม่หวังสิ่งตอบแทน

สารบัญ

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญรูปภาพ

หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันอุตสาหกรรมเกมคอมพิวเตอร์เป็นหนึ่งในอุตสาหกรรมที่มีเงินหมุนเวียนมากที่สุด ดังนั้นการวิจัยและพัฒนาเกมคอมพิวเตอร์จึงเป็นเรื่องน่าสนใจ

นักพัฒนาเกมที่มีประสบการณ์ไม่มากมักประสบปัญหาในการพัฒนาเกมอยู่เสมอ ซึ่งปัญหาเหล่านั้นมักมีผู้ที่สามารถหาวิธีแก้ไขได้แล้ว ทว่าไม่สามารถเปิดเผยได้ด้วยเหตุผลทางธุรกิจ โกสเอไอคือระบบปัญญาประดิษฐ์สำหรับเกมการต่อสู้ที่ได้มีการใช้ในท้องตลาดปัจจุบันแล้วแต่วิธีการทำยังคงไม่เป็นที่เปิดเผย ระบบจะทำการศึกษารูปแบบและวิธีการสู้ของผู้เล่นไว้ จากนั้นก็จะสร้างโกสเอไอเพื่อใช้ควบคุมตัวละครคอมพิวเตอร์เลียนแบบรูปแบบการเล่นของผู้เล่นออกมา เพื่อนำไปใช้สู้กับผู้เล่นคนอื่นได้

ประโยชน์ของโกสเอไอคือการดึงดูดผู้เล่นให้สนใจเล่นเกมนั้นได้มากขึ้น ทำให้ผู้เล่นใดๆ เสมือนที่จะสามารถเล่นแข่งกับผู้เล่นเจ้าของโกสเอไอนั้นได้เสมอ ไม่ว่าจะเจอตัวกันหรือไม่ก็ตาม จะมีบ่อยครั้งที่ผู้เล่นเกมการต่อสู้รู้สึกเบื่อเนื่องจากไม่มีคนเล่นด้วย และการเล่นกับคอมพิวเตอร์ก็มีรูปแบบการสู้ที่ซ้ำกัน อีกทั้งยังไม่มีความตื่นเต้นหรือแรงจูงใจในการเล่น แต่โกสเอไอทำให้ผู้เล่นมีคู่ต่อสู้ที่หลากหลายนอกเหนือจากเอไอเดิมที่ผู้สร้างเกมสร้างเอาไว้ ผู้เล่นใดๆ สามารถที่จะเสมือนเล่นกับคนที่มีฝีมือสูงได้โดยที่ไม่ต้องเจอตัวกันหรือไม่ต้องนัดเวลามาเล่นออนไลน์ นอกจากนี้ในปัจจุบันเทคโนโลยีระบบเครือข่ายนั้นมีการแพร่หลายอย่างกว้างขวาง การส่งแลกเปลี่ยนหรือแจกข้อมูลโกสเอไอของกันและกันจึงเป็นเรื่องที่ทำได้ง่าย เหตุนี้จึงเป็นการสนับสนุนให้ระบบนี้มีประโยชน์มากขึ้นด้วย ระบบโกสเอไอนี้มีตัวอย่างการใช้งานในปัจจุบันคือเกมเทคเคน 5 ดาร์คริเซอรัคชัน (Tekken5: Dark Resurrection) [1] ซึ่งออกสู่ตลาดตั้งแต่ช่วงปลายปี ค.ศ. 2005 จนถึงปัจจุบัน

เป้าหมายหนึ่งของวิทยานิพนธ์ฉบับนี้จะเป็นการค้นคว้าเทคนิคและวิธีการในการพัฒนาระบบโกสเอไอ เพื่อให้เกิดความรู้ที่สามารถเผยแพร่ใช้กับเกมการต่อสู้ที่มีขายตามท้องตลาดได้จริง

ทว่าในปัจจุบันการวิจัยด้านปัญญาประดิษฐ์สำหรับเกมมักจะประสบปัญหาในการหาเกมหรือสภาพแวดล้อมมาทดสอบกับระบบปัญญาประดิษฐ์ที่ได้คิดค้นขึ้น นักวิจัยต้องดัดแปลงเกมเพิ่มเติมโดยใช้เครื่องมือของเกมที่มีอยู่แล้ว [2] หรือบางครั้งนักวิจัยต้องเขียนเกมขึ้นใหม่เองทั้งหมด [3, 4] หรือไม่ก็ต้องเป็นทีมวิจัยของบริษัทเกม [5] งานวิจัยนี้ก็ประสบกับปัญหานี้เช่นกัน คือมีความต้องการสภาพแวดล้อมของเกมการต่อสู้ที่สามารถส่งโกสเอไอที่สร้างเข้าไปเล่นดูผลได้

เนื่องจากไม่มีเกมการต่อสู้เกมใดที่มาพร้อมกับเครื่องมือที่จะเปิดโอกาสให้ทดสอบระบบปัญญาประดิษฐ์ได้ อีกทั้งการจะสร้างเกมการต่อสู้ที่มีคุณภาพสูงเทียบเท่าเกมที่เป็นที่นิยมในตลาดก็เป็นเรื่องที่ทำได้ยาก และหากทำการทดสอบโกสเอไอบนเกมจำลองที่ค่อนข้างแตกต่างกับเกมการต่อสู้ชั้นนำ ก็อาจจะเกิดความไม่น่าเชื่อถือในผลงานได้ ดังนั้นจึงต้องหาวิธีการใหม่ในการเตรียมสภาพแวดล้อมสำหรับทดสอบโกสเอไอ โดยที่ตั้งเป้าหมายในทางที่เป็นประโยชน์ต่อส่วนรวมด้วย ว่าสภาพแวดล้อมนั้นควรจะใช้ทดสอบระบบปัญญาประดิษฐ์อื่นๆ สำหรับเกมการต่อสู้ได้ด้วย

ปัญหาทั้งสองส่วนนั้นเกี่ยวเนื่องกัน การที่จะสามารถเผยแพร่วิธีการพัฒนาโกสเอไอได้ก็ต้องศึกษาและวิจัยวิธีการก่อน ซึ่งก็ต้องการสภาพแวดล้อมเกมที่เหมาะสมสำหรับการทดลองก่อนเช่นกัน ดังนั้นวิทยานิพนธ์ฉบับนี้จะนำเสนอวิธีการสร้างสภาพแวดล้อมสำหรับทดสอบเกมการต่อสู้ที่มีคุณภาพสูงเทียบได้กับเกมที่เป็นที่นิยมในท้องตลาด ซึ่งจะเรียกสภาพแวดล้อมที่ใช้ทดสอบนี้ว่าเทสเบด (Testbed) และนำเสนอเทคนิควิธีการในการพัฒนาระบบโกสเอไอสำหรับเกมการต่อสู้

1.2 วัตถุประสงค์ของการวิจัย

เพื่อหาแนวทางในการสร้างโกสเอไอสำหรับเกมการต่อสู้และเพื่อพัฒนาเทสเบดสำหรับใช้ในการสร้างโกสเอไอ

1.3 ขอบเขตการวิจัย

1 กรณีศึกษาคือเกมสตรีทไฟท์เตอร์ซีโร่สามอัพเปอร์ (Street Fighter Zero3 Upper) [6]

2 ขอบเขตการพัฒนาเทสเบด เป็นดังนี้

2.1 พัฒนาเทสเบดชื่อ ไอเทม (AI-TEM: Artificial Intelligent Testbed in Emulator) ให้สามารถดึงค่าของข้อมูลเหล่านี้ออกมาจาก วิชวลบอยแอดวานซ์ (VisualboyAdvance) [7] อีมูเลเตอร์ของเครื่องเกมบอยแอดวานซ์ [8] ที่รันเกมสตรีทไฟท์เตอร์ซีโร่สามอัพเปอร์อยู่ ออกมาเก็บไว้ในตัวแปรภายในโปรแกรม อีมูเลเตอร์ และ บันทึกเป็นไฟล์ได้

- ตำแหน่งบนจอภาพของตัวละครและกระสุนของทั้งสองฝ่าย
- สภาพตัวละครว่ากำลังอยู่ในท่าทางไหนของตัวละครทั้งสองฝ่าย
- สถานะของแถบพลังชีวิต แถบความสามารถพิเศษของตัวละครทั้งสองฝ่าย
- คำสั่งการกดปุ่มของผู้เล่นฝ่ายผู้เล่นที่1

2.2 พัฒนาเทสเบดให้มีฟังก์ชันที่สามารถเข้าถึงตัวแปรที่เก็บข้อมูลในข้อ 2.1 ได้ และให้ค่าที่ส่งออกมาของฟังก์ชันนั้นสามารถแทนที่ค่าคำสั่งควบคุมคอนโทรลเลอร์ของวิชวลบอยแอดวานซ์อีมูเลเตอร์ได้

3 ขอบเขตการทดลองสร้างโกสเอไอเป็นดังนี้

3.1 เก็บข้อมูลการเล่นของผู้เล่น (ข้อมูลในข้อ 2.1) ที่เล่นเกม สตรีทไฟท์เตอร์ซีโร่

สามอัพเปอร์โดยผ่านเทสเบดไอเทม

3.2 เก็บข้อมูลการเล่นของผู้เล่นโดยให้ทั้งผู้เล่นและฝ่ายตรงข้ามใช้ตัวละคร ริว ในโหมดการต่อสู้ แซท-ไอเอสเอ็ม (Z-ISM) โดยฝ่ายผู้เล่นเป็นฝ่ายผู้เล่นที่ 1

3.3 เก็บข้อมูลการเล่นของผู้เล่นจากโหมดฝึกซ้อม (Training Mode) และมีการตั้งค่าดังนี้

- การตั้งค่าคำสั่งปุ่มกด (Button Config) หัวข้อคำสั่ง (Command) ตั้งค่าไว้ที่อาร์เขต (Arcade) หรือ ธรรมดา (Normal) หรือ ยาว (Long)
- การตั้งค่าเกม (Game Option) หัวข้อความเสียหาย (Damage) ตั้งไว้ที่ 2
- การตั้งค่าเกม หัวข้อระดับความเร็ว (Speed) ตั้งไว้ที่เทอร์โบ2 (Turbo2)
- การตั้งค่าตัวละครคอมพิวเตอร์ (CPU Character) หัวข้อ การป้องกันการตก (Safe Fall) ตั้งค่าไว้ที่ ปิด (Off)

3.4 สร้างโกสเอไอจากข้อมูลการเล่นของผู้เล่นที่เก็บมาตามข้อ 3.1 ถึง 3.3

4 วิธีการสรุปผลการทดลอง

รันโกสเอไอบนไอเทมและระหว่างนั้นให้เจ้าของโกสเอไอคอยกดคอนโทรลเลอร์เล่นไปด้วยพร้อมๆ กัน แต่ตัวละครในเกมจะทำตามที่โกสเอไอสั่งเท่านั้น จากนั้นเปรียบเทียบการกดปุ่มที่เกิดขึ้นในเวลาเดียวกันของโกสเอไอกับตัวเจ้าของโกสเอไอ

1.4 ขั้นตอนและวิธีดำเนินงานวิจัย

1. พัฒนาไอเทมให้รองรับความต้องการของงานวิจัย
2. เก็บข้อมูลตัวอย่างของตัวเองกับคอมพิวเตอร์เพื่อใช้ในการทดสอบการสร้างระบบ
3. เตรียมข้อมูลอนิเมชั่นและท่ากดของตัวละคร ริว
4. ออกแบบวิธีการวัดผลและเงื่อนไขการวัดผล
5. พัฒนาฟังก์ชันในการเก็บข้อมูลการต่อสู้จาก ไอเทม
6. พัฒนาฟังก์ชันในการสแกนข้อมูลการต่อสู้
7. พัฒนาฟังก์ชันในการอ่าน บันทึกและเพิ่มข้อมูลโกสเอไอ
8. พัฒนาฟังก์ชันในการสร้างโกสเอไอจากข้อมูลส่วนย่อยของโกสเอไอที่ได้จากการสแกน
9. ดูผลของโกสเอไอและปรับปรุงฟังก์ชันในข้อ 5 ถึง 8 หากผลที่ได้ไม่น่าพอใจ
10. ทดสอบสร้างโกสเอไอกับกลุ่มผู้ทดลองเพิ่ม
11. ดูผลของโกสเอไอและปรับปรุงฟังก์ชันในข้อ 5 ถึง 8 หากผลที่ได้ไม่น่าพอใจ
12. สรุปผลการวิจัย

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 เกมการต่อสู้: (Fighting Game) คือเกมที่โดยส่วนมากจะให้ผู้เล่นเลือกตัวละครมา 1 ตัว ต่อสู้กับฝ่ายตรงข้ามอีก 1 ตัว แบบ 1 ต่อ 1 ฝ่ายตรงข้ามอาจจะเป็นผู้เล่นอีกคนหรือควบคุมโดยตัวเกมเองก็ได้ ลักษณะเด่นในการควบคุมคือผู้เล่นมักจะต้องสั่งกดท่าต่างๆ ซึ่งซับซ้อนกว่าการควบคุมธรรมดา เพื่อให้ตัวละครแสดงท่าทางพิเศษเพื่อใช้ในการโจมตีหรือป้องกันตัว และมักจะให้สู้กันจนพลังชีวิตของฝ่ายตรงข้ามหมดจะถือว่าชนะ 1 ยก และการจะชนะ 1 นัดต้องเอาชนะให้ได้เกินครึ่งของจำนวนยกทั้งหมดเช่น 2 ใน 3 ยก เป็นต้น เกมการต่อสู้อาจจะเป็นเกมที่วาดตัวละครด้วยวิธีวาดภาพสองมิติ หรือด้วยวิธีแสดงโมเดลสามมิติก็ได้ ตัวอย่างของเกมแนวนี้เช่นเกมชุด สตรีทไฟท์เตอร์ (Street Fighter) ดังแสดงในรูปที่ 1 เกมเดอะคิงออฟไฟท์เตอร์ (The King of Fighter) หรือเกมเทคเคน เป็นต้น

รูปที่ 1 ตัวอย่างภาพเกมการต่อสู้ สตรีทไฟเตอร์ภาคสี่

ปัจจุบันเกมการต่อสู้ที่วาดภาพแบบสองมิติมีการผลิตเกมใหม่ออกมาน้อยมากเมื่อเทียบกับเกมการต่อสู้ที่วาดภาพด้วยวิธีสามมิติ นั่นเป็นเพราะว่ากระบวนการผลิตเกมที่วาดแบบสองมิตินั้นยากลำบากกว่า เนื่องจากจะต้องวาดท่าทางของตัวละครแต่ละท่าใหม่ทุกๆ จังหวะการขยับตัว ดังนั้นศิลปินผู้สร้างตัวละครจะต้องวาดภาพจำนวนมาก ส่วนเกมที่วาดแบบสามมิตินั้นจะใช้วิธีการกำหนดท่าหลักของการทำท่าแต่ละท่าและใช้วิธี ประมาณท่าทางการขยับตัวที่อยู่ระหว่างท่าหลักที่ได้กำหนดไว้ ศิลปินผู้สร้างตัวละครไม่จำเป็นต้องวาดภาพจำนวนมากเพื่อให้เกิดอนิเมชั่นของตัวละคร แม้ว่าจะมีอัตราการผลิตออกสู่ตลาดน้อยลงแต่ยังคงมีเกมที่วาดแบบสองมิติหลายเกมที่ผ่านการปรับสมดุลและแก้ไขจุดบกพร่องมายาวนานมาก เช่น เกมสตรีทไฟท์เตอร์สอง (ปี ค.ศ. 1991) เกมสตรีทไฟท์เตอร์ซีโร่สาม (ปี ค.ศ. 1998) เกมสตรีทไฟท์เตอร์สามเทิร์ดสไตรค์ (ปี ค.ศ. 1999) หรือ เกมเดอะคิงออฟไฟท์เตอร์เก้าห้า (ปี ค.ศ. 1995) เกมเหล่านี้แม้มีอายุนาน 10-15 ปี แต่ยังคงได้รับความนิยมสูง มีการทำลง

เครื่องเกมสมัยใหม่อยู่เสมอ ตัวอย่างเช่นเกมสตรีทไฟท์เตอร์สองยังคงมีการขายผ่านบริการเอ็กซ์บ็อกไลฟ์ (xbox live) อยู่และมีการปรับปรุงภาพให้สวยงามขึ้นในปี ค.ศ. 2007 เกมสตรีทไฟท์เตอร์ซีโร่สามที่ย้ายมาลงเครื่องเกมมือถือเพลย์สเตชั่น พอร์ตเทเบิล (playstation portable-PSP) ในปี ค.ศ. 2006 หรือเกมเดอะคิงออฟไฟท์เตอร์ที่ออกภาคใหม่สม่ำเสมอทุกปี นอกจากนั้นเกมการต่อสู้ที่วาดแบบสองมิติเกมใหม่ๆ ก็ยังคงมีออกมาเช่นกัน เช่น เกมซันโกคุบาซาร่าครอส (ปี ค.ศ. 2007) และเกมฤทธิ์หมัดดาวเหนือ (ปี ค.ศ. 2006) นอกจากนั้นในการแข่งขันเกมการต่อสู้ระดับโลกในปัจจุบันเช่น อาคาเดียทัวร์นาเมนต์ หรือ อีโวลูชันทัวร์นาเมนต์ ก็ยังคงมีการแข่งเกมเหล่านั้นอยู่ด้วย ส่วนเกมการต่อสู้ที่วาดภาพแบบสามมิตินั้นมีเกมใหม่ทยอยออกมาอย่างสม่ำเสมอ แต่เกมชุดที่เป็นที่นิยมนั้นส่วนมากจะมีประวัติยาวนานมาตั้งแต่สมัยบุกเบิกเกมสามมิติในช่วงปีค.ศ. 1994-1996 ได้แก่เกมชุด เทคเคน (Tekken) เวอชวลไฟท์เตอร์ (Virtua Fighter) โซลเอจ (Soul Edge) และเดธออร์อไลว์ (Dead or Alive) เป็นต้น เกมที่วาดแบบสามมิติเหล่านี้ส่วนมากจะมีรูปแบบการเล่นแตกต่างจากเกมที่วาดแบบสองมิติในบางส่วน เช่นเกมเหล่านี้ตัวละครไม่สามารถกระโดดได้สูงเกินความเป็นจริงเหมือนกับที่เกมแบบสองมิติทำกันได้ทุกเกม ทำให้รูปแบบหลักของการบุกเปลี่ยนไป เกมเหล่านี้นิยมเน้นการเล่นโดยการโจมตีต่อเนื่องมากกว่าการชิงจังหวะ เกมการต่อสู้แบบสองมิติบางเกมนั้นมีการสร้างเป็นแบบสามมิติออกมาด้วย เช่นสตรีทไฟท์เตอร์ และ เดอะคิงออฟไฟท์เตอร์ เกมพวกนี้จะแตกต่างออกไปจากเกมการต่อสู้แบบสามมิติดั้งเดิม เพราะว่าจะยังคงรูปแบบการเล่นแบบสองมิติเอาไว้เหมือนเดิมเช่น สามารถกระโดดได้สูงเกินจริง ในทางกลับกันเกมการต่อสู้ที่วาดภาพแบบสามมิติที่มีการเปลี่ยนรูปแบบไปทำแบบสองมิตินั้นแทบจะไม่มีเลย เกมที่พบว่าเคยทำเช่นนั้นคือ เทคเคนแอดวานซ์ ที่ทำลงเครื่องเกมมือถือเกมบอยแอดวานซ์ ซึ่งไม่มีความสามารถในการประมวลผลแบบสามมิติเลยต้องทำแบบสองมิติแทน แต่ก็ไม่ได้ประสบความสำเร็จในด้านยอดขาย

โดยสรุปแล้วเกมการต่อสู้เป็นเกมที่มีมายาวนาน ได้รับความนิยมมาตลอด และมีคนสนใจเล่นอยู่ในปัจจุบัน

2.1.2 โกสเอไอ: ในที่นี้หมายถึงฟังก์ชันที่ควบคุมตัวละครในเกมการต่อสู้ โดยโกสเอไอของผู้เล่นแต่ละคนจะพยายามลอกเลียนพฤติกรรมการเล่นของผู้เล่นคนนั้น

แนวคิดเกี่ยวกับโกสเอไอพบหลักฐานการใช้งานครั้งแรกในเกมเวอชวลไฟท์เตอร์สี่ [9] (ค.ศ. 2002) ในเกมนั้นมีโหมดพิเศษที่ให้ผู้เล่นสามารถฝึกสอนตัวละครคอมพิวเตอร์ได้ ว่าในสถานการณ์แต่ละอย่างนั้นผู้เล่นจะตอบสนองอย่างไร เมื่อทำการฝึกแล้วผู้เล่นสามารถบันทึกโกสเอไอลงบนเมมโมรีการ์ดแล้วนำไปใช้เล่นกับผู้เล่นคนอื่นได้ แต่ว่าผลตอบรับต่อระบบนี้ในเวลานั้นยังไม่ค่อยดีนัก เนื่องจากวิธีการสอนโกสเอไอยังไม่สะดวกผู้เล่นต้องกำหนดสถานการณ์ขึ้นมาเองและกำหนดวิธีตอบสนองซึ่งเป็นเรื่องที่ลำบาก อีกทั้งวิธีการเผยแพร่โกสเอไอไปใช้เล่นกับผู้อื่นยังไม่สะดวกอีกด้วย เพราะว่าจะต้องทำการคัดลอกไฟล์โกสเอไอผ่านทางเมมโมรีการ์ดของเครื่องเกมคอนโซลโดยตรง เนื่องจากไม่มีอุปกรณ์อื่นใดในสมัยนั้นที่สามารถอ่านข้อมูลในเมมโมรีการ์ดของเครื่องเกมคอนโซลได้ ทำให้การแลกเปลี่ยน

โกสเอไอของผู้เล่นจำกัดอยู่ในวงแคบ และไม่เป็นที่แพร่หลาย ทำให้ผู้เล่นไม่นิยมที่จะสร้างโกสเอไอของตัวเอง

ในปี ค.ศ. 2005 เกมเทคเคน 5 ดาร์ครีเซอร์รัคชันฉบับตู้เกม ได้นำแนวคิดโกสเอไอมาใช้และได้รับการตอบรับอย่างดี เกมนี้ใช้กลยุทธ์หลายอย่างในการดึงดูดคนเล่นได้ดีรวมถึงระบบโกสเอไอก็เป็นหนึ่งในกลยุทธ์เหล่านั้นด้วย เริ่มจากตู้เกมนี้จะมีช่องสำหรับเสียบการ์ดประจำตัวผู้เล่นเอาไว้ (ราคาการ์ดเปล่าที่จำหน่ายในประเทศไทยโดยบริษัทกาแลคซี่อีทีซี ซึ่งเป็นตัวแทนนำเข้าเกมนี้อย่างเป็นทางการคือ 250-200 บาทต่อใบ) เมื่อผู้เล่นใช้การ์ดเป็นครั้งแรกเขาจะต้องเลือกตัวละครมา 1 ตัวเพื่อบันทึกลงบนการ์ดนั้น และไม่สามารถเปลี่ยนแปลงตัวละครได้อีกตราบใดที่ใช้การ์ดใบนั้น ผู้เล่นที่เสียบการ์ดไว้ขณะเล่นเกมจะถูกสร้างโกสเอไอของผู้เล่นคนนั้นขึ้นมาโดยอัตโนมัติ นั่นคือผู้เล่นสามารถเล่นเกมไปตามปรกติไม่ต้องเข้าสู่โหมดการสอน เมื่อผู้เล่นเล่นเสร็จโกสเอไอจะถูกบันทึกเอาไว้ในตู้เกมตู้นั้นโดยอัตโนมัติ และหากว่าผู้เล่นคนนั้นกลับมาเล่นที่ตู้เดิมอีกโดยเสียบการ์ดใบเดิมไว้เป็นการแสดงตนโกสเอไอตัวเดิมของผู้เล่นก็จะถูกเพิ่มเติมรูปแบบการเล่นไปตามการเล่นครั้งใหม่ของผู้เล่นคนนั้นด้วย ประเด็นต่อมาคือโกสเอไอที่ถูกสร้างขึ้นโดยผู้เล่นใดๆ จะถูกนำไปใช้สู้กับผู้เล่นคนอื่นๆ ที่มาเล่นเกมที่ตู้นั้นแบบสุ่ม ทำให้คนที่มาเล่นเกมนี้ได้เล่นกับคู่ต่อสู้ที่มีการเล่นหลากหลายแบบที่เกิดจากโกสเอไอของคนเล่นคนอื่นจำนวนมาก ไม่เหมือนสมัยก่อนที่หากเล่นคนเดียวก็ต้องสู้กับคอมพิวเตอร์มาตรฐาน นอกจากนี้เกมยังสนับสนุนให้ผู้คนกลับมาเล่นโดยใช้การ์ดใบเดิมเพื่อทำให้โกสเอไอของตนเก่งขึ้น โดยการจดคะแนนเก็บเอาไว้ว่าโกสเอไอตัวไหนสามารถเอาชนะผู้เล่นคนอื่นๆ ได้มาก เมื่อเจ้าของโกสเอไอนำการ์ดกลับมาเสียบอีกครั้ง คะแนนของโกสเอไอของเขาก็จะถูกแสดงให้เห็น และตัวเจ้าของสามารถนำคะแนนเหล่านั้นไปปรับปรุงรูปลักษณ์ของตัวละครของเขา เช่นชุด เครื่องประดับหรือพลังออร่าที่แผ่ออกมานอกกายให้ดูเหนือระดับน่าเกรงขามกว่าตัวละครเดียวกันของผู้เล่นคนอื่นได้อีกด้วย ซึ่งสิ่งนี้เป็นตัวบ่งบอกฐานะความเก่งของทั้งตัวเจ้าของและโกสเอไอของผู้เล่นนั้น

เมื่อเกมนี้ย้ายไปลงเครื่องเกมมือถือหรือคอนโซลอย่าง เพลย์สเตชั่นพอร์ตเทเบิล หรือเพลย์สเตชั่นสาม ผู้พัฒนาเกมได้ทำการรวบรวมเอาโกสเอไอจากตู้เกมต่างๆ มาบรรจุไว้เป็นคู่ต่อสู้มาตรฐานเวลาที่ผู้เล่นเล่นคนเดียวอีกด้วย เป็นการประหยัดแรงงานในการสร้างเอไอให้คอมพิวเตอร์และเชิญชวนให้ผู้เล่นหน้าใหม่สร้างโกสเอไอของตัวเองไปด้วยในตัว ในระบบเพลย์สเตชั่นพอร์ตเทเบิลนั้น ผู้เล่นสามารถเลือกปิดเปิดระบบการสร้างโกสเอไอได้ตามต้องการ สามารถบันทึกหรือเริ่มการสร้างโกสเอไอใหม่ได้เองตามใจชอบ และยังสามารถอัพโหลดโกสเอไอของตนขึ้นไปบนเซิร์ฟเวอร์ที่ทางผู้พัฒนาเกมได้เตรียมไว้ให้ ทำให้ผู้เล่นอื่นๆ สามารถเชื่อมต่อไปยังเซิร์ฟเวอร์และดาวโหลดโกสเอไอของผู้เล่นคนอื่นจำนวนมากมาเล่นด้วยได้ นอกจากนี้สำหรับผู้ที่ไม่สะดวกในการใช้เครื่องเกมเชื่อมต่ออินเทอร์เน็ต ก็ยังมีกลุ่มผู้เล่นที่รวมตัวกันแจกจ่ายโกสเอไอผ่านทางเว็บไซด์ธรรมดาอีกด้วย [10] ผู้คนสามารถดาวโหลดไฟล์โกสเอไอจากทางเว็บบราวเซอร์และนำไฟล์ไปเก็บลงบนเม็มโมรี่การ์ดของเครื่องเล่นเกมตนและสู้กับโกสเอไอของคนอื่นได้ เม็มโมรี่การ์ดในปัจจุบันใช้แบบมาตรฐานที่ทั้งเครื่องเกมคอนโซลหรือเครื่อง

คอมพิวเตอร์ใดๆ สามารถอ่านข้อมูลได้ ทำให้การโยกย้ายโอนถ่ายโกสเอไอสะดวกขึ้น และเทคโนโลยีอินเตอร์เนทในปัจจุบันก็กว้างขวางและสะดวกขึ้นด้วยเช่นเดียวกัน

2.1.3 เทสเบด: หมายถึงโปรแกรมเครื่องมือ รวมถึงสภาพแวดล้อมต่างๆ ที่ใช้ในการทดสอบระบบ เทสเบดที่ดีควรจะสามารถทดสอบปัญหาที่สนใจได้อย่างสมจริงที่สุด ใช้งานได้ไม่ยากนักและมีความสามารถต่างๆ ที่สนับสนุนการทดลองให้ดำเนินไปโดยสะดวก ปัจจุบันเทสเบดสำหรับการทดสอบระบบปัญญาประดิษฐ์สำหรับเกมจะสร้างมาเพื่อใช้ทดสอบเกมยิงจากมุมมองบุคคลที่หนึ่งหรือเกมวางแผนรบแบบเรียลไทม์เป็นส่วนมาก งานวิจัยทางปัญญาประดิษฐ์ที่ทำกับเกมแนวอื่นจึงมีจำนวนน้อยมาก

ประเภทของเทสเบดสำหรับงานวิจัยเกี่ยวกับเกมสามารถแบ่งออกเป็นสามประเภทดังนี้

- ประเภทที่หนึ่งคือพัฒนาเกมขึ้นมาใหม่เลยเพื่อใช้ในการทดสอบ วิธีนี้มีข้อดีคือผู้วิจัยจะสามารถเข้าถึงและควบคุมข้อมูลทุกอย่างในเทสเบดได้ แต่ข้อเสียคือเป็นวิธีที่ทำได้ยากหากว่าเกมที่ต้องการจะใช้ทดสอบเป็นเกมที่ซับซ้อนและต้องการการปรับสมดุลอย่างมาก อย่างเช่นเกมการต่อสู้นั้นเป็นเรื่องยากมากหากจะพัฒนาขึ้นมาเองโดยกลุ่มผู้วิจัย เพราะเกมการต่อสู้ที่ได้รับความนิยมในปัจจุบันล้วนได้รับการปรับสมดุลมาเป็นเวลานานกว่าสิบปีจึงได้เกมที่มีคุณภาพดี

- ประเภทที่สองคือเทสเบดที่ดัดแปลงมาจากเกมที่มีการเปิดเผยซอร์สโค้ด ทำให้ผู้วิจัยสามารถดัดแปลงตัวเกมให้เป็นเทสเบดได้ ข้อดีของวิธีนี้คือผู้วิจัยไม่ต้องทำการพัฒนาเกมเอง สามารถเลือกหาเกมที่มีคุณภาพน่าพอใจแล้วนำมาดัดแปลงได้ แต่ข้อเสียคือในปัจจุบันไม่มีเกมให้เลือกมากนัก มีเกมที่เปิดเผยซอร์สโค้ดและคุณภาพดีอยู่ไม่มาก นอกจากนี้ผู้วิจัยยังต้องมีภาระในการทำความเข้าใจระบบการทำงานของเกมที่จะนำมาดัดแปลงอีก ตัวอย่างของเกมที่นิยมถูกนำมาดัดแปลงเป็นเทสเบดได้แก่ เกม สตราร์เทกัส (Stratagus) [11] และ สปริง (Spring) [12] ซึ่งเป็นเกมประเภทวางแผนรบแบบเรียลไทม์

- ประเภทที่สามคือเทสเบดที่ดัดแปลงจากเกมที่มีวางขายจริงโดยจะต้องขอความร่วมมือจากผู้พัฒนาเกม วิธีนี้เป็นวิธีที่น่าจะดีที่สุดในอุดมคติ เนื่องจากสามารถทดสอบกับเกมที่มีคุณภาพที่มีวางขายในท้องตลาดได้จริง และยังมีเอกสารจากผู้ผลิตอธิบายการทำงานของเกม ช่วยให้ผู้วิจัยสามารถเข้าใจและดัดแปลงเป็นเทสเบดได้โดยสะดวกด้วย แต่ว่าวิธีดังกล่าวมีข้อเสียที่สำคัญ คือเป็นวิธีที่มีความเป็นไปได้น้อยมาก ผู้พัฒนาเกมเพื่อการค้าส่วนมากจะไม่ยอมเปิดเผยซอร์สโค้ดของตัวเกมกับผู้อื่นเพื่อนำไปดัดแปลงได้ ตัวอย่างของผู้วิจัยที่ใช้วิธีนี้ได้แก่กลุ่มวิจัยที่เป็นส่วนหนึ่งของบริษัทเกมเท่านั้น [5]

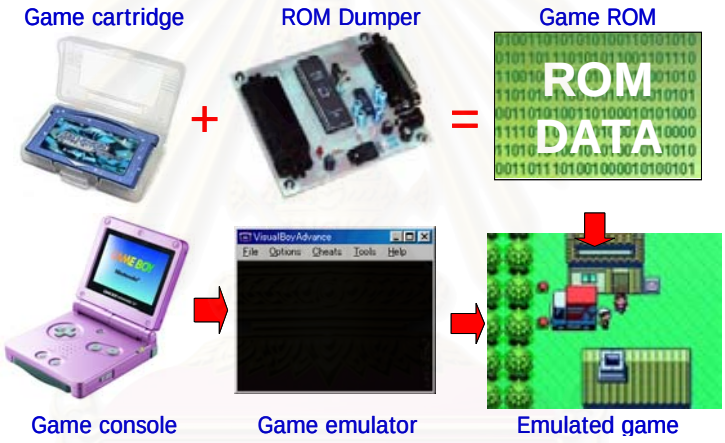
ดังนั้นจึงยังไม่มีวิธีหรือเทสเบดใดที่เหมาะสมกับการนำมาใช้กับเกมการต่อสู้เลย

2.1.4 เอไอ, ปัญญาประดิษฐ์: (AI, Artificial Intelligence) ในที่นี้หมายถึงฟังก์ชันใดๆ ที่ควบคุมการกระทำของตัวละครในเกม โดยไม่ขึ้นกับวิธีการทำงานหรือวิธีการสร้าง ซึ่งคำนี้มี

ความหมายแตกต่างไปในวงการวิชาการ แต่ความหมายตามวงการเกม (ผู้เล่นและผู้พัฒนาเกม) เป็นดังที่กล่าว

2.1.5 อีมูเลเตอร์และรอม: ในที่นี้อีมูเลเตอร์คือโปรแกรมที่จำลองการทำงานของเครื่องเล่นเกม เช่น เกมบอยแอดวานซ์ หรือ เพลย์สเตชั่น ในปัจจุบันมีอีมูเลเตอร์ของเครื่องเล่นเกมอยู่หลายประเภท เช่น วิชวลบอยแอดวานซ์ และ วิชวลบอยแอดวานซ์ลิงค์ [13] เป็นอีมูเลเตอร์สำหรับจำลองเครื่องเกมบอยแอดวานซ์ อีพีเอสเอ็กซ์อี (ePSXe) เป็นอีมูเลเตอร์สำหรับใช้จำลองเครื่องเพลย์สเตชั่น และ มาเมะ (MAME - Multiple Arcade Machine Emulator) เป็นอีมูเลเตอร์สำหรับจำลองตู้เกมเป็นต้น

อีมูเลเตอร์นั้น ใช้ข้อมูลของเกมจากรอม (ROM - Read Only Memory) ซึ่งเป็นข้อมูลเกมที่ถ่ายออกมาจากตลับเกมหรือแผ่นเกมเรียบร้อยแล้ว เกมหนึ่งเกมนั้นใช้รอมหนึ่งตัว ซึ่งผู้เล่นเกมสามารถเปลี่ยนเกมที่จะจำลองเล่นบนเครื่องได้ด้วยการโหลดรอมใหม่เท่านั้น การทำงานร่วมกันของอีมูเลเตอร์และรอมแสดงในรูปที่ 2



รูปที่ 2 อีมูเลเตอร์และรอม

2.1.6 สตรีทไฟท์เตอร์ซีโร่สามอัพเปอร์: คือเกมสตรีทไฟท์เตอร์สามฉบับที่พอร์ทลงเครื่องเกมมือถือ เกมบอยแอดวานซ์ ออกจำหน่ายในปี ค.ศ. 2002 แม้การย้ายมาลงเครื่องเกมที่มีความสามารถต่ำจะทำให้ต้องมีการตัดทอนอะไรหลายอย่างออกไป เช่นลดคุณภาพของภาพและเสียงลง แต่ตัวเกมในฉบับเครื่องพกพานี้ยังคงสนุก มีจังหวะและวิธีในการเล่นใกล้เคียงต้นฉบับมาก การต่อท่าต่างๆ แทบทั้งหมดสามารถทำได้เช่นเดียวกับต้นฉบับ และแม้ว่าวิธีการบังคับจะเปลี่ยนเป็นแบบ 4 ปุ่มจากเดิม 6 ปุ่ม แต่ว่าด้วยการตั้งค่าปุ่มกดแบบอัตโนมัติก็ช่วยให้การเล่นยังคงสะดวกได้เช่นเดิม

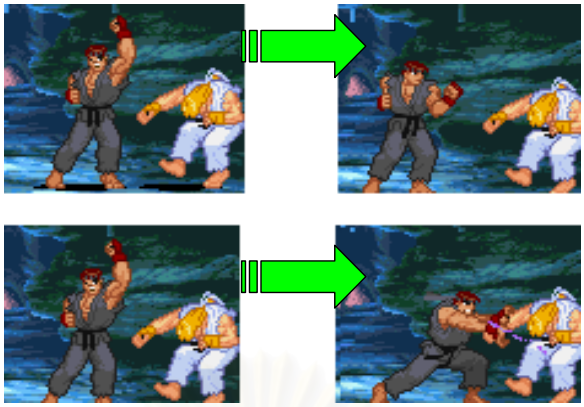
2.1.7 วิธีการเล่นเกมการต่อสู้: เกมการต่อสู้แม้ว่าจะเป็นเกมที่เริ่มเล่นได้ง่าย สามารถเล่นเป็นได้ในทันที แต่ว่าการที่จะเล่นได้เก่งนั้นเป็นเรื่องที่ต้องอาศัยเวลาและการฝึกซ้อมอย่างมาก การบังคับพื้นฐานคือการเดินหน้า ถอยหลัง ก้มและกระโดด ตามการกดปุ่มทิศทางทั้งสี่ การโจมตีพื้นฐานทำได้โดยการต่อยสามระดับ เบา กลางและหนัก การเตะสามมีระดับเช่นกัน การกดถอยหลังในระยะ

และจังหวะที่เหมาะสมจะเป็นการยกการ์ดป้องกันการโจมตีของคู่ต่อสู้สามารถทำได้ทั้งในท่ายืนและท่านั่ง ซึ่งจะต้องเลือกทำให้ถูกว่าศัตรูจู่โจมระดับบนหรือล่าง โดยธรรมชาติของเกมการต่อสู้แล้วผู้เล่นจะต้องใช้ทั้งการตอบสนองและการคาดเดาสถานการณ์ล่วงหน้าประกอบกัน ผู้เล่นไม่สามารถที่จะใช้แต่การตอบสนองเล่นเกมประเภทนี้ได้เพราะว่าสถานการณ์บางอย่างจะเร็วเกินกว่าจะตอบสนอง แต่จะอาศัยเดาอย่างเดียวก็ไม่ได้เพราะการเล่นจะออกมาไม่ได้ประสิทธิภาพ

นอกจากท่าพื้นฐานแล้วผู้เล่นจะต้องทำความเข้าใจวิธีการออกท่าพิเศษที่จะต้องใส่คำสั่งที่ซับซ้อนขึ้น แต่ก็จะให้ผลที่ดีขึ้นเช่นกัน ตัวอย่างเช่นเกมสตรีทไฟท์เตอร์ซีโร่สามตัวละครชื่อริว มีท่าที่ใช้ยิงพลังฮาโดเคน (ท่าปล่อยกระสุนพลังออกจากฝ่ามือ) สามารถโจมตีคู่ต่อสู้ได้จากระยะไกล วิธีการกดคือกด ลง ลงพร้อมเดินหน้า และเดินหน้า ตามด้วยปุ่มต่อย ภายในเวลาที่กำหนด (ประมาณ 0.5 วินาที) ก็จะสามารถออกท่าพิเศษได้ ท่าพิเศษเหล่านี้มีบางท่าในบางเกมที่มีวิธีการกดที่ยากมากเพื่อท้าทายความสามารถผู้เล่น

การเลือกใช้ท่าต่างๆ ให้เหมาะสมตามสถานการณ์เป็นหลักการสำคัญในการเล่นเกมการต่อสู้ทุกเกม ท่าแต่ละท่าจะมีคุณสมบัติต่างกันไป ตัวอย่างเช่นการต่อยเตะพื้นฐานซึ่งมีในทุกเกม มักจะเป็นสิ่งที่ผู้เล่นหน้าใหม่มองข้ามไม่ชอบใช้ เนื่องจากผู้เล่นหน้าใหม่มักสนใจท่าพิเศษที่ดูหวือหวา แต่แท้ที่จริงท่าพื้นฐานเหล่านั้นมีความสำคัญมากทีเดียว ท่าหมัดเบาสามารถใช้หลอกล่อและทำลายจังหวะโจมตีต่อเนื่องของศัตรูได้ เพราะว่าท่าเบานั้นมีการออกท่าที่เร็วมาก จนไม่สามารถยกการ์ดทันด้วยการใช้การตอบสนองต่อภาพที่เห็นได้ ทำให้การโจมตีเบานั้นนิยมใช้เป็นการเปิดจังหวะการโจมตีต่อเนื่องอีกด้วย ท่าหมัดกลางนั้นแม้จะออกท่าช้ากว่าหมัดเบาบ้างแต่ก็ทำให้ศัตรูชะงักได้นานกว่าทำให้มีโอกาสโจมตีต่อเนื่องได้มากกว่าหมัดเบา ส่วนหมัดหนักนั้นใช้เวลาในการออกท่าและการเก็บหมัดนานมากมักไม่สามารถที่จะเปิดโอกาสให้ใช้ท่าต่อเนื่องได้ แต่ว่าความแรงของหมัดนั้นก็เทียบเท่ากับหมัดเบาได้ 3-4 หมัด การทำความเข้าใจเรื่องเวลาในการออกท่า เวลาในการเก็บท่าและเวลาที่ศัตรูชะงักจากการถูกโจมตีเป็นสิ่งสำคัญมากในการเตรียมการโจมตีต่อเนื่องหรือที่เรียกกันว่าคอมโบ

การทำคอมโบนั้นเป็นเทคนิคการเล่นในระดับสูง หลักการคือเมื่อตัวละครในเกมต่อสู้ถูกโจมตีจะต้องอยู่ในท่าชะงัก บาดเจ็บเป็นเวลาหนึ่ง ประมาณ 0.1-0.5 วินาทีแล้วแต่ท่าที่ถูกโจมตีและจุดที่ถูกโจมตี ระหว่างที่อยู่ในท่าบาดเจ็บนั้นตัวละครจะทำอะไรไม่ได้เลยเป็นเป้านิ่งให้ถูกโจมตีต่อ ฝ่ายผู้ที่โจมตีจะต้องอาศัยเวลาเศษเสี้ยววินาทีนั้นออกท่าโจมตีครั้งต่อไปให้ได้เพื่อทำให้เกิดความเสียหายมากที่สุดและให้เกิดผลของการชะงักต่อไปเปิดโอกาสให้โจมตีต่อไปได้อีก แต่โดยทั่วไปแล้วฝ่ายถูกโจมตีมักจะหายจากท่าถูกโจมตีก่อนที่ผู้โจมตีจะเก็บท่าโจมตีเสร็จ ทำให้การโจมตีต่อเนื่องเป็นเรื่องที่เกิดขึ้นได้ยากมาก แต่ก็มีท่าโจมตีบางท่าที่สามารถทำต่อเนื่องกันได้โดยที่ฝ่ายที่บาดเจ็บยังไม่หายจากท่าชะงัก ตัวอย่างดังแสดงในรูปที่ 3 ด้านบนคือกรณีปรกติ ส่วนด้านล่างคือการเกิดคอมโบ จะสังเกตได้ว่าท่าเก็บหมัดจะถูกยกเลิกไปและเกิดท่าใหม่ต่อเนื่องทันทีในขณะที่ศัตรูยังยาดเจ็บอยู่



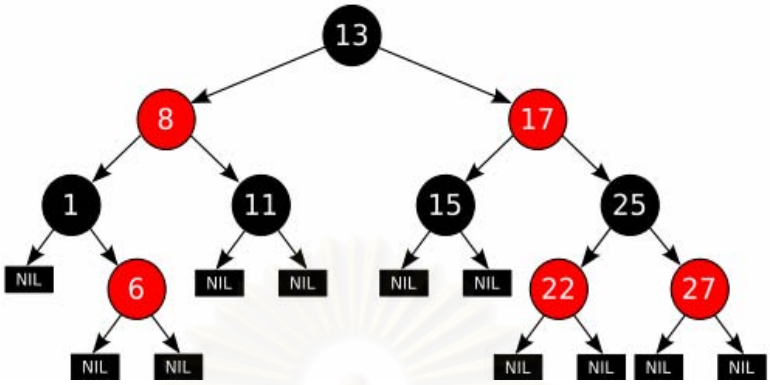
รูปที่ 3 แสดงการทำคอมโบ

ระบบการทำคอมโบนั้นมีรายงานอย่างเป็นทางการว่าเกิดจากข้อผิดพลาดในระบบเกม [14] ซึ่งพบครั้งแรกในเกมการต่อสู้ยุคบุกเบิกคือสตรีทไฟท์เตอร์สอง กรณีตัวอย่างที่พบบ่อยคือตัวละครริวสามารถทำการยิงกระสุนฮาโดเคนต่อทันทีหลังจากที่หมัดต่อยหนักเพิ่งจะปะทะตัวศัตรูได้ ทั้งที่ตามความเป็นจริงแล้วจะต้องรอให้หมัดหนักนั้นเก็บกลับมาก่อนแล้วตัวละครจึงจะทำท่าทางอื่นต่อได้ การที่เกิดการกระทำต่อเนื่องนี้ได้เป็นเพราะว่าระบบการตรวจสัญญาณคอนโทรลเลอร์เพื่อออกท่าพิเศษแบบยอมอนุโลมของทางผู้ผลิตเกมนั้นยอมให้มีคำสั่งต่อยแทรกเข้ามาในระหว่างการกดท่ายิงพลังได้ โดยที่ยอมรับให้ทั้งท่าต่อยและท่ายิงพลังแสดงผลได้ทั้งคู่ ปรากฏว่าข้อผิดพลาดดังกล่าวกลายเป็นสิ่งดีที่ผู้เล่นชอบ เนื่องจากการกดคอมโบให้สำเร็จจะต้องอาศัยฝีมือดังนั้นคอมโบจึงกลายเป็นที่นิยมเพราะเป็นการท้าทายผู้เล่น ผู้ผลิตเกมการต่อสู้ส่วนใหญ่จึงให้ความสนใจกับการทำคอมโบและพัฒนามาเป็นระบบหลักของการเล่นเกมการต่อสู้จนถึงทุกวันนี้ และเป็นหนึ่งในระบบของเกมที่ใช้เป็นกรณีศึกษาของวิทยาพนธ์นี้ด้วย

2.1.8 ต้นไม้ค้นแบบทวิภาค (Binary Search Tree): คือโครงสร้างข้อมูลแบบต้นไม้ชนิดหนึ่งซึ่งทุกโหนดภายในต้นจะมีลูกได้ไม่เกินสอง โดยที่ทุกโหนดในต้นไม้ย่อยฝั่งซ้ายของโหนดใดๆ จะต้องมีค่าน้อยกว่าโหนดนั้นเสมอ และทุกโหนดในต้นไม้ย่อยฝั่งขวาของโหนดใดๆ ก็จะต้องมีค่ามากกว่าโหนดนั้นเสมอเช่นกัน โดยทั่วไปในกรณีถัวเฉลี่ยความลึกของต้นไม้ค้นแบบทวิภาคที่มีจำนวนข้อมูล n ตัวจะมีความลึก log n ทำให้ในกรณีถัวเฉลี่ยนั้นเวลาที่ใช้ในการแทรกและค้นข้อมูลในต้นไม้นี้จะใช้เวลา O(log n)

ต้นไม้ค้นแบบทวิภาคมีรูปแบบพิเศษคือต้นไม้แดงดำ ซึ่งเป็นต้นไม้ค้นแบบทวิภาคที่มีการให้สีของโหนดเป็นสีแดงและดำเพิ่มเข้ามา โดยที่โหนดรากจะเป็นสีดำเสมอและถ้าโหนดใดเป็นสีแดงลูกของโหนดนั้นจะต้องเป็นสีดำเสมอ ผลพิเศษของต้นไม้แดงดำคือจะเป็นต้นไม้ค้นแบบทวิภาคที่มีความสูงไม่เกิน 2log(n+1) เสมอ ทำให้สามารถรับประกันเวลาในการค้นข้อมูลในเวลา O(log n) ได้ ซึ่งต้นไม้แดงดำนี้เป็นโครงสร้างข้อมูลที่ถูกนำไปใช้ในการสร้างไลบรารีชนิด แม็ป (Map) ที่เป็นมาตรฐานทั้งใน

ภาษาซีและจาวา ซึ่งนิยมนำไปใช้ในการเขียนโปรแกรมที่ต้องการเก็บข้อมูลที่ต้องมีการค้นข้อมูลอย่างมาก รูปที่ 4 แสดงตัวอย่างต้นไม้แดงดำ



รูปที่ 4 ตัวอย่างต้นไม้แดงดำ

2.2 การทบทวนวรรณกรรม และงานวิจัยที่เกี่ยวข้อง

Graepel [5] (Learning to Fight.) ได้นำเทคนิคการเรียนรู้แบบบรีอินฟอร์สเมนท์มาใช้ในการปรับพฤติกรรมของศัตรูภายในเกมให้เก่งขึ้นได้จากการต่อสู้กับผู้เล่นในเกมการต่อสู้ โดยทดลองกับเกมที่วางขายในตลาดจริง แต่ว่างานวิจัยชิ้นนี้เป็นงานจากบริษัทผู้ผลิตเกมเอง ซึ่งมีซอร์สโค้ดของเกมทั้งหมด ซึ่งตามปกติแล้ว บริษัทเกมมักไม่ยอมเปิดเผยซอร์สโค้ดต่อผู้อื่น วิธีการนี้จึงใช้ไม่ได้สำหรับนักวิจัยส่วนใหญ่ นอกจากนี้ Graepel ยังไม่ได้ทดลองเพื่อศึกษาโกสเอไออีกด้วย

Sander [15] (Gathering and utilizing domain knowledge in commercial computer games.) ได้ทำการสร้างแบบจำลองเกมการเดินหลบพื้นที่กับดักอย่างง่ายขึ้นมาเพื่อทดสอบแนวคิดในการนำเคสเบสรีซันนิง มาใช้กับการตัดสินใจของตัวละครในเกม คือการที่เปรียบเทียบหาวิธีการตัดสินใจที่จะให้ผลดีที่สุดในสถานการณ์ปัจจุบัน จากการค้นหาข้อมูลการตัดสินใจสถานการณ์ในอดีตที่ใกล้เคียงที่สุด วิธีการแบบนี้อาจจะนำมาใช้ในการสร้างโกสเอไอได้ โดยการเก็บข้อมูลการเล่นของผู้เล่นไว้เป็นฐานข้อมูลและเลือกการกระทำจากฐานข้อมูลเวลาที่โกสเอไอทำงานจริง อย่างไรก็ตามงานวิจัยนี้ทดลองในสภาพแวดล้อมของโปรแกรมจำลองการหาเส้นทาง ไม่ได้ทดลองกับเกมการต่อสู้

Pedro Demasi [4] (Online Co-Evolution for Action Games.) ได้ทำการพัฒนาปัญญาประดิษฐ์สำหรับเกมแอคชั่น เพื่อทดลองปรับระดับความยากของเกมตามความสามารถของผู้เล่น แต่ทว่ากฎของเกมที่เขาออกแบบเพื่อใช้ในการทดสอบนั้นมีช่องโหว่ ทำให้ผู้เล่นมีวิธีที่จะเล่นได้ดีโดยที่ศัตรูในเกมไม่มีโอกาสได้ปรับตัวตามจุดประสงค์ของการวิจัย งานวิจัยนี้แสดงให้เป็นถึงผลจากคุณภาพของเกมที่นำมาใช้ทดสอบในการวิจัย

Spronck [2,16] (On-line Adaptation of Game Opponent AI with Dynamic Scripting., Enhancing the Performance of Dynamic Scripting in Computer Games.) คณะของเขาได้ทดลองปัญญาประดิษฐ์แบบไดนามิคสคริปต์ โดยในการหาเกมมาใช้ทดสอบระบบปัญญาประดิษฐ์นั้น

เริ่มแรกเขาได้สร้างเกมขึ้นมาเอง โดยให้มีฉากฉากหนึ่งเลียนแบบเกม เบาเดอร์เกท (Baldur's Gate) แล้วทดลองนำปัญญาประดิษฐ์ที่คิดค้นขึ้นใหม่ไปใช้งาน แม้ผลจะเป็นที่น่าพอใจ แต่เนื่องจากเป็นเกมที่สร้างขึ้นมาเอง ทำให้ผู้วิจัยไม่สามารถแน่ใจได้ว่าปัญญาประดิษฐ์ที่คิดค้นขึ้นจะสามารถนำไปใช้กับเกมได้จริง ดังนั้นจึงมีการใช้เครื่องมือของเกมที่มีอยู่แล้ว นั่นคือเกมเนเวอร์วินเธอร์ไนท์ (NeverWinter Nights) สร้างฉากขึ้นมาทดสอบใหม่ ซึ่งผลจากการทดลองเป็นการยืนยันได้ระดับหนึ่งว่างานของพวกเขาสามารถนำไปใช้กับเกมที่มีอยู่ตามท้องตลาดได้ แต่ว่าการทดลองนั้นจำต้องจำกัดอยู่กับเกมที่มีเครื่องมือให้อยู่แล้วเท่านั้น ซึ่งเกมที่มีเครื่องมือให้นั้น เป็นเกมแนววางแผนรบแบบเรียลไทม์ หรือ เกมยิงจากมุมมองบุคคลที่หนึ่งเสียส่วนมาก เกมการต่อสู้หรือเกมประเภทอื่นจะไม่สามารถนำมาทดสอบได้เลย แนวคิดในการวิจัยปัญญาประดิษฐ์โดยใช้เกมที่มีคุณภาพ เป็นที่นิยมและวางขายจริงในท้องตลาดในงานวิจัยเป็นแรงบัลดาลใจให้กับวิทยานิพนธ์นี้

Ponsen [17] (Automatically Acquiring Adaptive Real-Time Strategy Game Opponents Using Evolutionary Learning) ได้ใช้เกมโอเพนซอร์สที่เลียนแบบเกมที่มีวางขายจริงแต่ไม่มีกาให้เครื่องมือสำหรับดัดแปลงมาด้วย การใช้เกมโอเพนซอร์สนั้นมีข้อเสียคือ ตัวเกมเลียนแบบ อาจไม่มีคุณภาพเทียบเท่าของจริงและได้ศึกษาเกมประเภทวางแผนรบแบบเรียลไทม์เท่านั้น

Bailey [18] (An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games.) ได้ทำการดัดแปลงสร้างเอนจิ้นของเกมเพื่อการทดลองการเรียนรู้ข้อมูลต่างๆ จากผู้เล่นได้ในระบบปัญญาประดิษฐ์ แต่เอนจิ้นที่สร้างนั้นเป็นเพียงตัวพื้นฐานที่สามารถรองรับการเก็บสถิติและปรับพารามิเตอร์ภายในเกมได้จำกัด ยังเป็นเทสเบดที่ไม่สมบูรณ์ นอกจากนี้ยังไม่ได้ถูกสร้างสำหรับทดสอบเกมการต่อสู้อีกด้วย ถ้าหากนักวิจัยจะนำงานวิจัยนี้ไปใช้เพื่อทดสอบเกมการต่อสู้ก็จะต้องเสียเวลาในการสร้างเกมขึ้นมาใหม่

งานวิจัยด้านระบบปัญญาประดิษฐ์สำหรับเกมนั้นส่วนใหญ่จะทำกับเกมที่มีเครื่องมือมาให้หรือใช้กับเครื่องมือของเกมเลียนแบบซึ่งไม่ใช่เกมการต่อสู้ ยังไม่มีงานวิจัยใดเกี่ยวข้องกับการพัฒนาโกสเอไอเพื่อเกมการต่อสู้เลย

บทที่ 3

ไอเทม เทสเบดสำหรับทดสอบเอไอในเกม

ไอเทมคือเทสเบดสำหรับใช้ทดสอบเอไอในเกม มีความสามารถเด่นที่ใช้ทดสอบกับเกมคอนโซลได้โดยที่ไม่ต้องใช้ซอร์สโค้ดหรือรีซอร์สของเกมนั้น บทนี้จะกล่าวถึงรายละเอียดของเทสเบดนี้ตั้งแต่แนวคิดที่มา วิธีการสร้าง หลักการทำงาน และการใช้งาน

3.1 ที่มาในการสร้างไอเทมเทสเบด

งานวิจัยเกี่ยวกับเกมส่วนมากในปัจจุบันจะทำกับเกมบนเครื่องพีซี ซึ่งเน้นเกมประเภทเกมยิงจากมุมมองบุคคลที่หนึ่ง หรือเกมประเภทวางแผนรบแบบเรียลไทม์เป็นหลัก มีงานวิจัยบางชิ้นที่ทำกับเกมประเภทอื่นเช่นพัสเซิลหรือแอคชั่น [3, 4] แต่ว่าผู้วิจัยจำเป็นต้องพัฒนาเกมสำหรับทดสอบขึ้นมาใหม่ การพัฒนาเกมที่มีช่องโหว่ในการเล่นขึ้นมานั้นส่งผลถึงผลการทดลองเป็นอย่างมาก [4] จากการศึกษาพบว่ามีงาน [5] เท่านั้นที่ทดสอบกับเกมการต่อสู้ โดยทดสอบกับเกมที่มีวางขายจริงของเครื่องคอนโซล แต่ว่าวิธีการทดลองของงานนั้นมีการใช้ข้อมูลเกมจริงที่ไม่เปิดเผยทั่วไปนอกจากใช้ในห้องทดลองของบริษัทผู้ผลิตเกมเท่านั้น ไม่สามารถนำวิธีการมาพัฒนาต่อได้ สถานการณ์เช่นนี้เป็นเหตุหนึ่งที่ทำให้ไม่มีการพัฒนาความรู้สำหรับเกมประเภทการต่อสู้ งานวิจัยนี้จึงต้องการสร้างเทสเบดสำหรับทดสอบเกมการต่อสู้ขึ้นมาเพื่อใช้ทดสอบโกสเอไอที่สนใจจะศึกษา

เทสเบดสำหรับเกมที่ดีควรใช้ทดสอบเอไอได้หลายประเภท สามารถทดสอบได้กับเกมที่มีคุณภาพ เกมที่แสดงความแตกต่างของผู้เล่นที่ชำนาญกับมือใหม่ได้ชัดเจน และหากเป็นเกมที่เป็นที่นิยมก็จะเพิ่มความน่าเชื่อถือสำหรับผลการทดลองด้วย

สิ่งที่เทสเบดต้องการนั้นแท้จริงก็คือดวงตาที่สามารถมองจอทีวีที่แสดงภาพเกมที่เล่นอยู่ และแปลความหมายของภาพนั้นให้อยู่ในสภาพที่เอไอเข้าใจได้ จากนั้นก็ให้เอไอทำการประมวลผลและตัดสินใจกระทำการใดๆ โดยส่งคำสั่งมาบังคับแขนกลเพื่อกดปุ่มคอนโทรลเลอร์ควบคุมเกม แขนกลที่ใช้กดปุ่มคอนโทรลเลอร์สามารถถูกแทนที่ได้ด้วยการส่งสัญญาณเข้าเครื่องเกมโดยตรงได้ แต่ว่าดวงตาที่ทำการมองและเข้าใจสถานการณ์ในเกมได้นั้นไม่อาจสร้างขึ้นได้ง่าย ด้วยเทคโนโลยีในปัจจุบันผู้เขียนจึงคิดหาวิธีการจำลองการสังเกตุสถานะของเกมดังกล่าวด้วยการใช้อีมูเลเตอร์ของเครื่องเล่นเกม

อีมูเลเตอร์ของเครื่องเล่นเกมนั้นเมื่อใช้คู่กับรอมของเกมจะทำให้สามารถจำลองการเล่นเกมได้ สามารถส่งคำสั่งควบคุมคอนโทรลเลอร์ได้โดยตรงโดยไม่ต้องอาศัยแขนกล ส่วนการทำความเข้าใจ

สถานการณ์ในเกมนั้นก็จะทำได้โดยอาศัยหลักการดูค่าในหน่วยความจำของอีมูเลเตอร์นั้น จากแนวคิดนี้การทดลองสร้างเทสเบดจากอีมูเลเตอร์จึงได้เริ่มขึ้น

3.2 วิชวลบอยแอดวานซ์ลิงค์อีมูเลตอร์

วิชวลบอยแอดวานซ์ลิงค์ เป็นโอเพนซอร์สอีมูเลเตอร์ของเครื่องเกมบอยแอดวานซ์ ถูกเลือกมาเป็นฐานในการดัดแปลงเป็นเทสเบด เกมบอยแอดวานซ์เป็นเครื่องเกมมือถือของบริษัทนินเทนโด ซึ่งเริ่มวางจำหน่ายในปี ค.ศ. 2001 เป็นเครื่องเกมที่มียอดจำหน่ายสูงมากกว่าเจ็ดสิบล้านเครื่องทั่วโลกมีเกมรองรับออกมามากกว่าสองพันเกม เครื่องใช้ซีพียูอาร์ไอเอสซีความเร็ว16 เมกกะเฮิตร์ แบบ 32 บิต หน่วยความจำ 32 กิโลไบต์ และหน่วยความจำวีดีโอ 256 กิโลไบต์ แม้ว่าจะเป็นเครื่องที่มีความสามารถระดับธรรมดาและปัจจุบันไม่มีเกมใหม่ออกให้กับเครื่องนี้แล้วก็ตาม แต่เกมที่ออกมานั้นมีหลายเกมที่มีความสนุกและเล่นได้ทุกยุคทุกสมัย

วิชวลบอยแอดวานซ์ลิงค์เป็นอีมูเลเตอร์ที่พัฒนาต่อมาจากวิชวลบอยแอดวานซ์ธรรมดาโดยเพิ่มความสามารถในการเล่นเชื่อมกันหลายอินสแตนท์ของอีมูเลเตอร์ได้ด้วย เนื่องจากเกมบอยแอดวานซ์เป็นเครื่องเกมมือถือที่เล่นได้คนเดียวและผู้พัฒนาวิชวลบอยแอดวานซ์ธรรมดาไม่ได้พัฒนาความสามารถในการเล่นเชื่อมกัน อย่างไรก็ตามระบบพื้นฐานของวิชวลบอยแอดวานซ์ลิงค์นั้นอิงมาจากวิชวลบอยแอดวานซ์ธรรมดาทั้งหมด เพื่อความสะดวกจากนี้ไปจะขอเรียกวิชวลบอยแอดวานซ์ลิงค์ว่าวีบีเอ

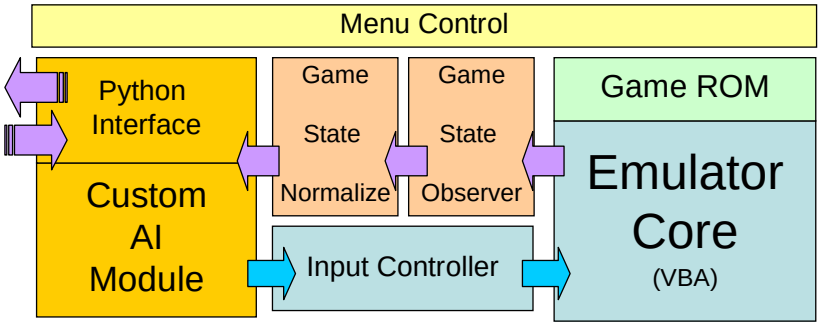
แม้ว่าจะมีอีมูเลเตอร์ของเครื่องเกมอื่นอีกมากมายแต่ว่าวีบีเอมีคุณสมบัติเหมาะสมที่สุดเพราะว่ามีเครื่องมือที่เอื้อประโยชน์ต่อการดัดแปลงเป็นเทสเบด มีความสะดวกในการแก้ไขอีมูเลเตอร์และมีเกมที่รองรับความต้องการของงานวิจัยนี้ ซึ่งก็คือเกมสตรีทไฟท์เตอร์ซีโร่สามอัพเปอร์

3.3 หลักการทำงานของไอเทม

การทำงานของไอเทมแยกเป็นส่วนต่างๆ ดังรูปที่ 5 ซึ่งรูปนี้เป็นเฟรมเวิร์ก ที่ออกแบบมาโดยเฉพาะสำหรับงานวิจัยนี้และพัฒนาส่วนต่างๆ เองทุกส่วนยกเว้นส่วนอีมูเลเตอร์ซึ่งเป็นโอเพนซอร์สอีมูเลเตอร์และส่วน อินพุตคอนโทรลเลอร์ ซึ่งเป็นส่วนหนึ่งของอีมูเลเตอร์อยู่แล้ว รายละเอียดการทำงานของส่วนต่างๆ เป็นดังนี้

1 อีมูเลเตอร์ (Emulator Core): ส่วนพื้นฐานหลักของเครื่องมือจะเป็นตัวอีมูเลเตอร์วีบีเอทำหน้าที่ในการรันตัวเกมใดๆ ก็ได้ที่มีตัวรอมถูกต้อง

2 เมนูควบคุม (Menu Control): ส่วนนี้เป็นส่วนที่ติดต่อกับผู้ใช้งานเครื่องมือ โดยผู้ใช้จะสามารถสั่งเปิดปิดการทำงานของเอไอหรือเลือกระหว่างเอไอแบบต่างๆ ได้ นอกจากนี้ยังออกแบบให้ผู้ใช้สามารถเรียกเครื่องมือย่อยต่างๆ ที่จำเป็นได้จากส่วนนี้



รูปที่ 5 ระบบสำหรับสร้างปัญญาประดิษฐ์เพื่อทดลองบนอีมูเลเตอร์

3 เกมสเตทออบเซิร์ฟเวอร์ (Game State Observer): ประเด็นหลักของการนำอีมูเลเตอร์มาใช้เป็นเทสเบดคือต้องสามารถรู้ได้ว่าข้อมูลสถาณะของเกมที่จำเป็นต่อการทดสอบเอไอในแต่ละรอบการทำงานมีค่าเป็นเท่าใดบ้าง ตัวอย่างเช่นเกม สตรีทไฟท์เตอร์ซีโร่สาม ที่ใช้เป็นกรณีศึกษานั้น สถานะของเกมที่สนใจประกอบด้วย ตำแหน่งของตัวละคร ตำแหน่งของกระสุนของตัวละคร พลังชีวิต และค่าอนิเมชั่นของตัวละคร หน้าที่ของเกมสเตทออบเซิร์ฟเวอร์คือการคัดลอกค่าต่างๆ จากแอดเดรสที่ผู้ใช้สนใจโดยคัดลอกในทุกๆ รอบการทำงานแล้วทำการส่งข้อมูลต่อไปยังส่วนของปัญญาประดิษฐ์ต่อไป

การหาค่าแอดเดรสของข้อมูลที่สนใจนั้นแม้จะเป็นงานที่ไม่ง่ายนัก แต่เสียเวลาน้อยมากเมื่อเทียบกับการสร้างเกมที่มีคุณภาพสูง วิธีการหาค่าแอดเดรสจะกล่าวในส่วนวิธีการใช้งานเทสเบด

4 เกมสเตทนอมัลไลเซอร์ (Game State Normalizer): ส่วนนี้จะอยู่คั่นระหว่าง เกมสเตทออบเซิร์ฟเวอร์กับส่วนเอไอ ใช้เพื่อการเปลี่ยนข้อมูลให้ง่ายต่อการใช้งานของส่วนเอไอเท่านั้น เช่น ค่าตัวแปรหนึ่งมีค่าตั้งแต่ 45,987 ถึง 70,158 โดยแต่ละค่านั้นเพิ่มทีละ 12 อีกด้วย ไม่ได้เพิ่มทีละ 1 ตามปรกติ ซึ่งจะมีทั้งหมด 2,014 ค่า ระบบส่วนนี้จะทำหน้าที่เปลี่ยนค่าจาก 45,987 ถึง 70,158 ให้เป็น 0 ถึง 2014 เพื่อให้สะดวกต่อการใช้งานและความเข้าใจ เกมสเตทนอมัลไลเซอร์เป็นส่วนที่ไม่จำเป็นจะต้องมีก็ได้แต่ในทางปฏิบัตินั้นช่วยให้การใช้งานสะดวกขึ้นมาก

5 เอไอโมดูล (AI Module): ตรงนี้คือส่วนที่ผู้ใช้งานจะเขียนปัญญาประดิษฐ์จริงลงไป ซึ่งในทุกรอบของการรันอีมูเลเตอร์นั้นตัวอีมูเลเตอร์จะเรียกใช้ส่วนปัญญาประดิษฐ์นี้ โดยหลักการส่วนนี้จะทำงานโดยอ่านสถานะของเกมที่ส่งมาจากเกมสเตทนอมัลไลเซอร์ แล้วตัดสินใจบังคับตัวละครโดยจะส่งการบังคับเป็นคำสั่งในการกดคอนโทรลเลอร์ของเกม ส่วนรูปแบบของเอไอที่จะเขียนลงไปในส่วนนี้ก็คือซอร์สโค้ดที่เป็นส่วนหนึ่งของวีบีเอซึ่งก็คือภาษาซีหรือซีพลัสพลัสนั่นเอง

6 อินพุตคอนโทรลเลอร์ (Input Controller): แต่เดิมวีบีเอจะรับสัญญาณคอนโทรลเลอร์มาจากคีย์บอร์ดหรือคอนโทรลเลอร์ที่ผู้ใช้กำหนดเพื่อเอาไปประมวลผลในเกม อินพุตคอนโทรลเลอร์จะทำการส่งสัญญาณคอนโทรลเลอร์จากเอไอโมดูลเข้าไปแทนสัญญาณเดิม ทำให้ส่วนเอไอสามารถควบคุมเกมได้แทน

7 ไพธอนสคริปต์ อินเทอร์เฟส (Python Script Interface): ไพธอน [19] เป็นภาษาเชิงวัตถุที่ไม่ต้องการการคอมไพล์ สามารถรันได้บนระบบปฏิบัติการหลายๆ ตัว เป็นที่นิยมใช้ในการเขียนสคริปต์ ซึ่งข้อดีของการใช้ไพธอนก็คือสามารถเปลี่ยนสคริปต์โดยไม่ต้องคอมไพล์โปรแกรมหรือรันตัวเกมใหม่ เทสเบดถูกออกแบบให้มีส่วนที่สามารถเชื่อมต่อกับไพธอนสคริปต์ได้เพื่อความสะดวกของผู้ใช้เพราะว่าสามารถแก้ไขสคริปต์และดูผลได้อย่างรวดเร็วอีกทั้งผู้ใช้ยังไม่ต้องจัดการเรื่องการดูแลซอร์สโค้ดของอีมูเลเตอร์อีกด้วย

การที่จะทำให้ไพธอนสคริปต์สามารถเชื่อมต่อกับเทสเบดได้นั้นจะต้องทำการเตรียมอินเทอร์เฟสฟังก์ชันทางฝั่งเทสเบดเพื่อให้สคริปต์สามารถเข้าถึงข้อมูลสถานะของเกมที่รันอยู่ในตัวเทสเบดได้และเพื่อที่จะให้สคริปต์สามารถส่งคำสั่งกดคอนโทรลเลอร์มาควบคุมเกมได้ด้วย ตัวอย่างการทำงานประสานระหว่างไพธอนและเทสเบดแสดงดังนี้

ทางฝั่งเทสเบดเตรียมอินเทอร์เฟสฟังก์ชันเหล่านี้ไว้ให้สคริปต์ใช้

```
int GetCharacterData (int charID, int dataID)
int PressButton (int* buttonSignal)
```

- ฟังก์ชัน `GetCharacterData` ใช้ในการขอข้อมูลสถานะของเกม รับพารามิเตอร์สองตัว ตัวแรกเป็นหมายเลขของตัวละครฝ่ายผู้เล่นที่หนึ่งหรือสอง ตัวที่สองเป็นหมายเลขที่บอกว่าต้องการรู้ข้อมูลใดของตัวละครนั้น เช่น ตำแหน่งบนแกนราบ-ดิ่ง ของตัวละคร ค่าพลังชีวิต หรือท่าทางของตัวละครขณะนั้น และรีเทิร์นเออเรอร์โค้ดออกมา

- ฟังก์ชัน `PressButton` ใช้เพื่อสั่งกดปุ่มควบคุมตัวละคร รับพารามิเตอร์หนึ่งตัวคือพอยเตอร์ชี้ไปยังอาร์เรย์ที่เก็บสัญญาณของปุ่มที่จะสั่งกดและรีเทิร์นเออเรอร์โค้ดออกมาเช่นกัน

รายละเอียดวิธีการประกาศฟังก์ชันในฝั่งเทสเบดเพื่อให้ไพธอนสคริปต์เรียกใช้งานได้จะกล่าวในภาคผนวก ส่วนตัวอย่างไพธอนสคริปต์ที่ใช้กับเทสเบดเป็นดังนี้

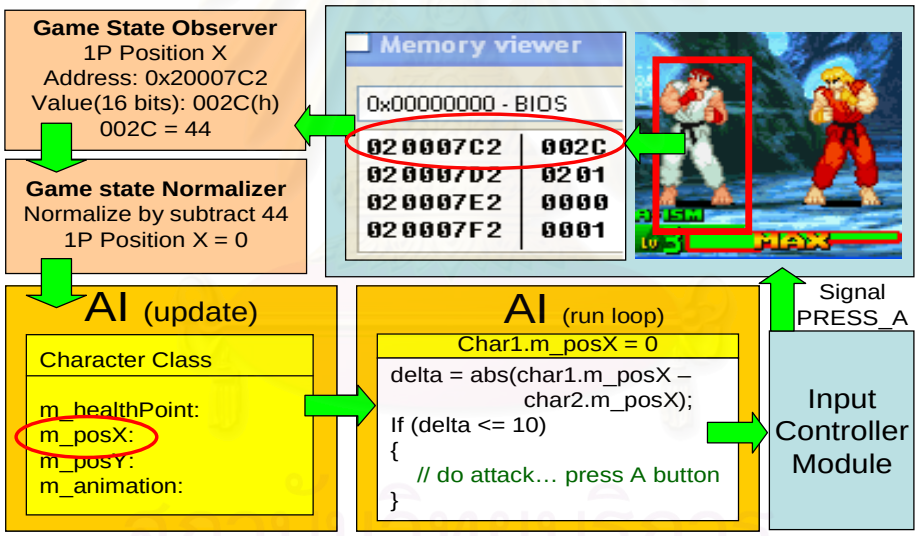
```
import mySFZ3Lib
def Main_AI_Run_Loop():
if (mySFZ3Lib.GetCharacterData(SFZ3_DATA_POS_X, ENEMY) < 10)
{
      array = ((PRESS_DOWN), (PRESS_DOWN| PRESS_FORWARD),
            (PRESS_FORWARD), (PRESS_B))
      mySFZ3Lib.PressButton(array)
}
```

`import mySFZ3Lib` คือการสั่งอิมพอร์ตไลบรารี่จากฝั่งเทสเบดเข้ามาใช้ ซึ่งก็คือสองฟังก์ชันข้างต้นนั่นเอง จากนั้นในลูปการทำงานก็จะขอข้อมูลตำแหน่งบนแกนระนาบของตัวละครฝ่ายตรงข้ามมาตรวจหากว่าน้อยกว่าค่าที่กำหนดก็สั่งให้ออกท่าพิเศษโดยการกด ลง ลงเฉียงหน้า เดินหน้าตามด้วยปุ่มต่อยเพื่อยิงกระสุนพลัง ด้วยการทำงานในลักษณะนี้การพัฒนาเอไอด้วยภาษาสคริปต์จึงทำได้ หรือหากจะใช้ภาษาสคริปต์อื่นเช่น รัว (Lua) หรือรูบี้ (Ruby) ก็ได้เช่นกัน

ตัวอย่างการทำงานจริงของเทสเบดแสดงในรูปที่ 6 เมื่อเกมรอมถูกรันโดยอีมูเลเตอร์ ตัวเกมจะถูกจำลองและเริ่มเล่นได้ สามารถตัดเข้าที่ฉากต่อสู้ของเกมนี้โดยการโหลดสเตทที่เตรียมไว้ และสั่งผ่านเมนูคอนโทรลให้เริ่มการทำงานของเอไอที่สร้างขึ้นเองได้

ในทุกรอบการทำงานนั้น เกมสเตทออบเซิร์ฟเวอร์จะคัดลอกค่าที่สนใจที่ได้กำหนดไว้ล่วงหน้าแล้วจากแอดเดรสที่กำหนด ซึ่งรูปที่ 6 นี้แสดงการคัดลอกค่าพิกัดแกนราบของตัวละครซ้ายมือที่เก็บโดยแอดเดรส 0x20007C2 ซึ่งมีค่าในระบบเลขฐาน16 เป็น 2C และส่งค่าไปให้ส่วนเกมสเตทนอมัลไลเซอร์ทำการปรับค่าให้เรียบร้อย ค่านี้เป็นค่าพิกัดแกนราบซึ่งควรจะเริ่มที่พิกัด 0 แต่ว่าจากภาพดังกล่าวแม้ว่าจะอยู่ที่ริมซ้ายสุดแล้วแต่ค่าก็ยังเป็น 2C จึงนอมัลไลซ์โดยการลบออก 2C เพื่อให้ค่าริมซ้ายสุดเป็น 0

จากนั้นจึงส่งข้อมูลนี้ไปให้ส่วนเอไอทำการประมวลผล ในที่นี้ส่วนเอไอจะตรวจดูว่าหากระยะห่างของตัวละครทั้งสองในแกนราบน้อยกว่าค่าที่กำหนดก็จะสั่งให้ตัวละครโจมตี แล้วสั่งกดปุ่มเพื่อโจมตีโดยส่งสัญญาณการกดปุ่มไปให้อีมูเลเตอร์ ด้วยระบบการทำงานนี้ไอเทมจึงสามารถเข้าใจสถานะปัจจุบันภายในเกมและใช้ทดสอบเอไอได้



รูปที่ 6 แผนภาพแสดงตัวอย่างการทำงานของไอเทมกับเกมสตรีทไฟท์เตอร์ซีโร่สาม

3.4 สรุปขั้นตอนการเตรียมการและการใช้ไอเทม

การใช้ไอเทมเทสเบดนั้นผู้ใช้ควรจะต้องเตรียมการดังต่อไปนี้

1 กำหนดรายการข้อมูลที่ต้องการรู้จากเกมเพื่อที่จะใช้ในการรันของเอไอ ข้อมูลสถานะของเกมสถานะใดบ้างที่เอไอจำเป็นต้องใช้ในการทำงาน

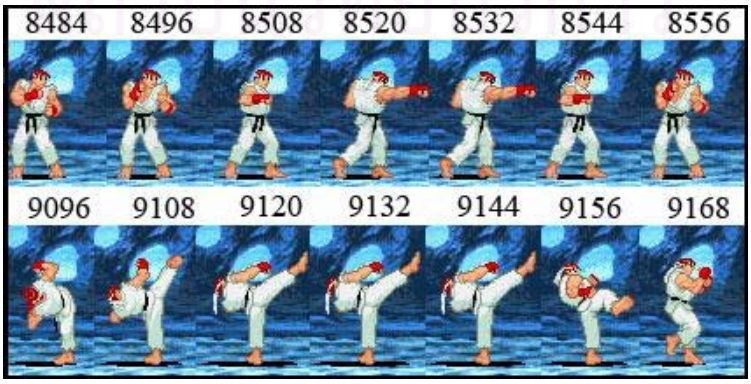
2 ทำการหาแอดเดรสที่เก็บข้อมูลเหล่านั้นบนอีมูเลเตอร์ วิธีการหาพื้นฐานคือใช้เครื่องมือเมมโมรีเซิร์จ โดยการสั่งให้ค้นหาค่าที่มีการเปลี่ยนแปลงตามที่ข้อมูลในเกมที่ต้องการรู้เปลี่ยนแปลงไป ตัวอย่างเช่นหากต้องการหาว่าแอดเดรสใดเก็บค่าพลังชีวิตของตัวละคร ก็เล่นเกมให้พลังชีวิตลดลงหรือ

เพิ่มขึ้นแล้วทำการสั่งค้นหาค่าทั้งหมดว่าค่าไหนที่ลดลงหรือเพิ่มขึ้นอย่างสอดคล้องกับค่าพลังชีวิตนั้น โดยที่การค้นหาจะต้องทำการกำหนดขนาดของข้อมูลว่าจะค้นแบบ 8 บิต 16 บิต หรือ 32 บิตให้เหมาะสมจึงจะสามารถหาเจอ ถ้าหากหาแอดเดรสไม่สำเร็จผู้ใช้เทสเบดควรพิจารณาว่าหากขาดข้อมูลนั้นไปเอไอจะยังคงทำงานได้หรือไม่และตัดสินใจที่จะใช้หรือไม่ใช้ตัวเทสเบดนี้ต่อไป อนึ่งข้อมูลแอดเดรสของเกมสตรีทไฟท์เตอร์ซีโร่สามนั้นแสดงในตารางที่ 1

3 หลังจากการหาแอดเดรสของข้อมูลที่ต้องการแล้ว ให้พิจารณาถึงช่วงและความหมายของค่าในแต่ละแอดเดรสที่ต้องการด้วย ข้อมูลในเกมบางอย่างอาจจะต้องอิงกับรูปภาพที่แสดงออกมาจึงต้องเข้าใจให้ได้ด้วยว่ารูปภาพนั้นกับค่าตัวเลขในแอดเดรสสัมพันธ์กันอย่างไร ค่าแต่ละค่ามีความหมายอย่างไร ตัวอย่างเช่นในเกมสตรีทไฟท์เตอร์ซีโร่สามค่าแอดเดรสที่แสดงข้อมูลท่าทางของตัวละครนั้นจำเป็นจะต้องรู้ด้วยว่าแต่ละท่าอ้างอิงกับค่าใดในแอดเดรสนั้นดังแสดงตัวอย่างในรูปที่ 7 ซึ่งจะเห็นได้ว่าหากค่าในแอดเดรส 0x20007D0 เป็น 8508 นั้นจะแปลว่าตัวละครกำลังทำท่าชกอยู่ ในกรณีเช่นนี้ก็จะต้องเก็บข้อมูลเพื่อให้รู้ได้ว่าทุกๆ ค่าที่ปรากฏในแอดเดรสนี้มีความหมายว่าอะไรบ้าง

ตารางที่ 1 รายการข้อมูลและแอดเดรสที่สนใจของเกมสตรีทไฟท์เตอร์ซีโร่สาม

Game State Data	Address (P1)	Address (P2)	Data size
Char ID	0x20008EA	0x20044FA	8 bits
Hit Point	0x2000818	0x2004428	16 bits
ISM Bar	0x200090E	0x200451E	16 bits
Pos X	0x20007C2	0x20043D2	16 bits
Pos Y	0x20007c6	0x20043D6	16 bits
Animation	0x20007D0	0x20043E0	32 bits
Animation ref value	0x2000828	0x2004438	32 bits
Bullet Pos X	0x200C0DA	0x20021AA	16 bits
Bullet Pos Y	0x200C0DE	0x20021AE	16 bits
Bullet End?	0x200c1c4	0x2002294	16 bits
Damage	0x2000AAA	0x20046BA	16 bits



รูปที่ 7 ตัวอย่างท่าทางและค่าในแอดเดรสของตัวละครริวในเกมสตรีทไฟท์เตอร์ซีโร่สาม

3.5 เครื่องมือที่ช่วยในการใช้งานไอเทม

วีบีเอมีเครื่องมือที่สนับสนุนแนวคิดในการสร้างเทสเบดได้แก่

- ชีทส์เชิร์จ: เครื่องมือในการค้นหาแอดเดรสของค่าต่างๆ ในเกม เดิมเครื่องมือนี้ใช้เพื่อค้นหาค่าจำพวกพลังชีวิตของตัวละครเพื่อที่จะบังคับค่าเอาไว้ให้เล่นเกมได้สะดวกขึ้น แต่ว่าหลักการทำงานของเครื่องมือนี้สามารถใช้ค้นหาค่าอื่นๆ ที่บ่งบอกถึงสภาพปัจจุบันของเกมได้ด้วย รูปที่ 8 แสดงภาพขณะเครื่องมือนี้ถูกใช้งาน

รูปที่ 8 เครื่องมือชีทส์เชิร์จ

- มูวี่ส์เรคคอร์ดส: เครื่องมือนี้ใช้ในการบันทึกการเล่นเกม สามารถบันทึกได้สองรูปแบบ แบบแรกคือในรูปเอวีไอไฟล์ (AVI file) ซึ่งสามารถเปิดดูได้กับโปรแกรมเล่นหนังทั่วไป แต่ว่าขนาดของไฟล์จะใหญ่และใช้พลังของเครื่องในการทำงานค่อนข้างมากเวลาบันทึก อีกรูปแบบหนึ่งคือแบบวีเอ็มวีไฟล์ (VMV file) ซึ่งคือการบันทึกสถานะของเกมในตอนที่เริ่มต้นบันทึกการเล่นเอาไว้ แล้วบันทึกสัญญาณคอนโทรลเลอร์ที่ใช้ควบคุมเกมต่อจากสถานะนั้นเอาไว้คู่กัน วิธีนี้ทำให้บันทึกการเล่นได้โดยใช้พลังในการประมวลผลน้อยมาก แต่ว่าสามารถเปิดดูได้ด้วยอีมูเลเตอร์เท่านั้นและต้องอาศัยรอมของเกมในการเปิดดูด้วย กล่าวคือวิธีนี้เป็นการจำลองการเล่นเกมอีกรอบจากสัญญาณคอนโทรลเลอร์ที่บันทึกไว้นั่นเอง ความสามารถในการบันทึกการเล่นได้นั้นจะเป็นประโยชน์ต่อการวิจัยในการดูหรือวิเคราะห์ผลย้อนหลัง
- เมมโมรีวิวเวอร์: เครื่องมือที่ใช้ในการดูค่าของแอดเดรสต่างๆ ที่กำหนด สามารถดูค่าได้เป็นช่วงกว้างพร้อมๆ กันได้ เลือกดูค่าในแอดเดรสโดยแบ่งตามขนาดข้อมูลได้ทั้งแบบ 8, 16 และ 32 บิต อีกทั้งยังสามารถเขียนค่าลงไปในแต่ละแอดเดรสได้โดยตรงด้วย รูปที่ 9 แสดงภาพตัวอย่างการใช้งานของเครื่องมือนี้

รูปที่ 9 เครื่องมือเมมโมรีวิวเวอร์

- เซฟ/โหลด เกมสเตท: นี่คือเครื่องมือในการบันทึกสถานะในการเล่นเกมซึ่งสามารถใช้งานได้ทุกเมื่อและสามารถโหลดสถานะกลับมาได้ตลอดเวลา ทำให้สามารถทดลองเอไอในสถานการณ์เดิมได้อย่างสมบูรณ์แบบ

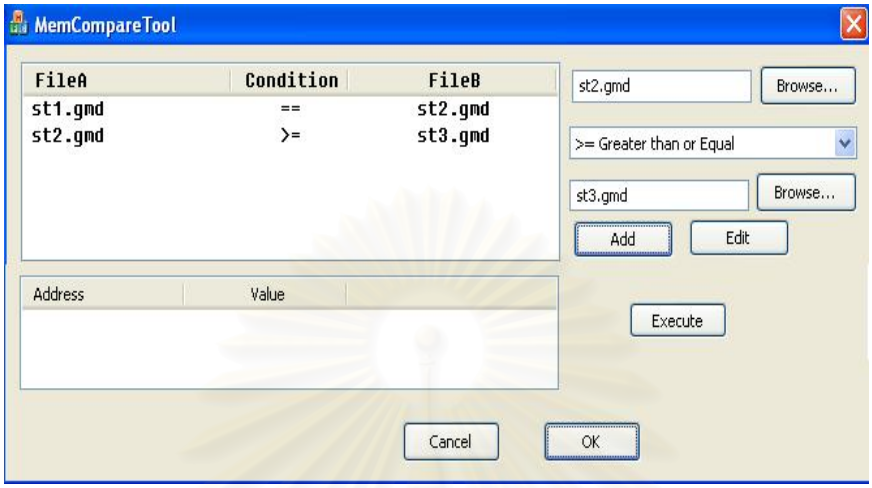
งานวิจัยนี้ได้มีการพัฒนาเครื่องมือเพิ่มเติมเพื่อช่วยสนับสนุนการทำงานของไอเทม ได้แก่

- โปรแกรมช่วยในการหาแอดเดรส: เครื่องมือนี้จะช่วยให้การหาค่าแอดเดรสของข้อมูลที่ต้องการสามารถทำได้สะดวกขึ้น โดยการให้ผู้ใช้สั่งบันทึกข้อมูลสถานะของข้อมูลทั้งหมดในเมมโมรีของอีมูเลเตอร์ขณะรันเกมออกมาเป็นช่วงๆ หลายสถานการณ์ แล้วกำหนดเงื่อนไขในการหาค่าแอดเดรสที่มีการเปลี่ยนแปลงค่าตามที่กำหนดไว้ หลักการทำงานจะเป็นแบบเดียวกับเครื่องมือซีทส์เซิร์จ เพียงแต่ว่าการกำหนดเงื่อนไขการหานั้นทำครั้งเดียวรวมกันหลายสถานการณ์ได้ และให้ผลการค้นได้ทุกรูปแบบของขนาดข้อมูลในครั้งเดียว ลดความสับสนของผู้ใช้ลง ตัวอย่างการทำงานเช่น เมื่อผู้ใช้จะหาแอดเดรสที่เก็บข้อมูลพลังชีวิตของตัวละคร เขาจะต้องบันทึกข้อมูลเมมโมรีทั้งหมดในเกมในหลายสถานการณ์ลงบนหลายไฟล์โดยใช้เครื่องมือนี้ จากนั้นจึงกำหนดเงื่อนไขการค้นหาให้เหมาะสม ตัวอย่างเช่นผู้ใช้ได้บันทึกข้อมูลในสถานการณ์มาดังนี้

สถานการณ์ที่1: พลังชีวิตของตัวละครอยู่ที่ 100 จุด
สถานการณ์ที่2: พลังชีวิตของตัวละครอยู่ที่ 50 จุด
สถานการณ์ที่3: พลังชีวิตของตัวละครอยู่ที่ 70 จุด

เขาสามารถกำหนดเงื่อนไขการค้นหาโดยให้เครื่องมือหาแอดเดรสดังนี้ หาข้อมูลแบบขนาดบิตที่กำหนด โดยกำหนดให้ข้อมูลที่ต้องการหานั้นในไฟล์ที่ 1 มีค่ามากกว่าในไฟล์ที่ 2 ในไฟล์ที่ 2 มีค่าน้อยกว่าในไฟล์ที่ 3 และในไฟล์ที่ 3 มีค่าน้อยกว่าในไฟล์ที่ 1 ด้วยหลักการนี้หากบันทึกข้อมูลในสถานการณ์ที่เหมาะสมและกำหนดเงื่อนไขได้ดี การหาแอดเดรสจะสามารถทำได้ดวกขึ้น ผลลัพธ์การค้นหาจะ

บันทึกในไฟล์หลายไฟล์โดยอัตโนมัติตามรูปแบบของข้อมูล เช่นการค้นด้วยเงื่อนไขเดียวกันแต่ข้อมูลแบบ 8 บิตไซน์ จะถูกเก็บผลในไฟล์หนึ่ง ส่วนข้อมูลแบบ 8 บิตอันไซน์ จะถูกเก็บในอีกไฟล์หนึ่งเป็นต้น รูปที่ 10 แสดงภาพขณะใช้เครื่องมือนี้



รูปที่ 10 โปรแกรมช่วยในการหาแอดเดรส

- เครื่องมือในการเขียนข้อมูลบนเมมโมรีและจับภาพ: เครื่องมือนี้สามารถสั่งเขียนข้อมูลลงไปในแอดเดรสที่กำหนดและบันทึกภาพหน้าจอของอีมูเลอเตอร์ขณะนั้นได้ ใช้ในการเก็บภาพเพื่อดูว่าสภาพเกมเป็นอย่างไรเมื่อค่าในแอดเดรสนั้นเป็นค่าที่สั่งเขียน ตัวอย่างการใช้งานของเครื่องมือนี้คือใช้ในการเก็บข้อมูลท่าทางของตัวละครในเกมสตรีทไฟท์เตอร์ซีโร่สาม ได้ผลออกมาเป็นรูปที่ 7 ข้างต้น
- เครื่องมือในการรันเอไอโดยอัตโนมัติ: เครื่องมือนี้จะสั่งให้มีการโหลดสเตทและสั่งรันเอไอโดยอัตโนมัติ ทำให้ผู้ใช้สามารถทดสอบเอไอชนิดที่ต้องการรันเป็นเวลานาน ได้โดยไม่ต้องควบคุมด้วยตนเอง ผู้ใช้สามารถกำหนดเงื่อนไขของข้อมูลในแอดเดรสเอาไว้เป็นตัวกำหนดว่าจะเริ่มโหลดสเตทเพื่อทดลองส่วนต่อไปเมื่อใด ตัวอย่างเช่นในเกมสตรีทไฟท์เตอร์ซีโร่สาม หากต้องการสั่งให้ตัวละครต่อสู้กันหลายคู่หลายยกต่อเนื่องกัน ผู้ใช้สามารถทำได้โดยกำหนดเงื่อนไขในแอดเดรสที่เก็บค่าพลังชีวิตของตัวละครไว้เมื่อค่านั้นเป็น 0 แสดงว่าจบการต่อสู้ในนัดนั้นแล้ว ให้ทำการโหลดสเตทใหม่เพื่อให้คู่ต่อไปลงมาสู้ได้ โดยที่ผู้ใช้ยังสามารถปรับรายละเอียดของเงื่อนไขและสเตทที่โหลดได้ด้วย

เทสเบดไอเทมซึ่งพัฒนาขึ้นมาสำหรับใช้ในงานวิจัยนี้ทำให้สามารถทดลองรันเกมการต่อสู้ได้เหมือนจริง ทำให้สามารถจัดการต่อสู้ระหว่างผู้เล่นสองคนได้ หรือจัดการต่อสู้ระหว่างเอไอเดิมของเกมสองตัวก็ได้ ผู้ใช้เทสเบดอาจจะเล่นเกมเองแล้วให้ระบบการเรียนรู้ทำการสังเกตการเล่นของเขา หรือปล่อยให้เอไอเดิมของเกมเล่นเกมไปแล้วให้ระบบการเรียนรู้ทำการสังเกตการเล่นของเอไอเดิมก็ได้ จึงสามารถนำข้อมูลที่บันทึกได้ไปประมวลผลเพื่อสร้างโกสเอไอต่อไปได้

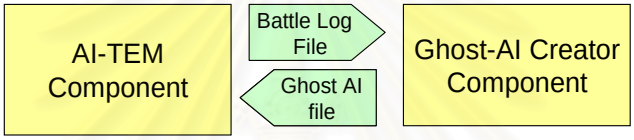
บทที่ 4

การสร้างและใช้งานโกสเอไอ

ในบทนี้จะกล่าวถึงวิธีการพัฒนาและหลักการทำงานของโกสเอไอ

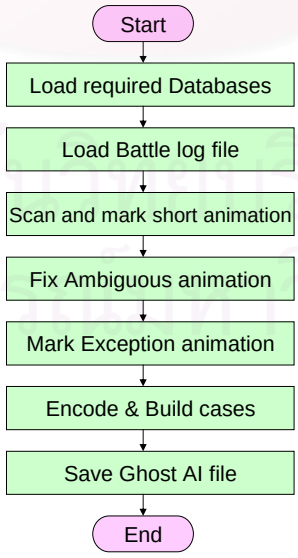
4.1 ภาพรวมของการสร้างและใช้งานโกสเอไอ

ข้อมูลที่จะนำมาใช้ในการสร้างโกสเอไอคือบันทึกการต่อสู้ของผู้เล่นที่จะได้มาจากตัวเทสเบด บันทึกการต่อสู้ของผู้เล่นจะถูกสร้างเป็นโกสเอไอด้วยโปรแกรมสำหรับสร้างโกสเอไอ (Ghost AI Creator) และหลังจากสร้างโกสเอไอเสร็จแล้วจะได้ผลลัพธ์ออกมาเป็นโกสเอไอไฟล์ ซึ่งจะนำกลับไปรันทดสอบผลในเทสเบดต่อไป การทำงานระหว่างเทสเบดและโปรแกรมสร้างโกสเอไอนี้แสดงในรูปที่ 11



รูปที่ 11 แสดงความสัมพันธ์ระหว่างเทสเบดและโปรแกรมสำหรับสร้างโกสเอไอ

หลักการสร้างและการทำงานของโกสเอไอในงานวิจัยนี้คือการเก็บข้อมูลการเล่นของผู้เล่นในสถานการณ์ที่ตัวละครของผู้เล่นเปลี่ยนกลุ่มท่ามาสร้างเป็นเคส เคสนั้นหมายถึงคู่ของสถานการณ์และการกระทำตอบสนองต่อสถานการณ์นั้น โดยที่โกสเอไอจะเลือกกระทำการตามที่ผู้เล่นเลือกทำเมื่อเจอสถานการณ์เดียวกัน ขั้นตอนการสร้างโกสเอไอของโปรแกรมสำหรับสร้างโกสเอไอแสดงในรูปที่ 12 และมีรายละเอียดดังต่อไปนี้



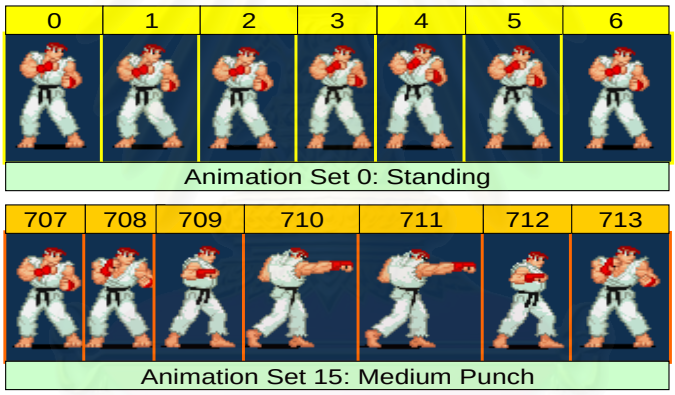
รูปที่ 12 แสดงขั้นตอนการสร้างโกสเอไอของโปรแกรมสำหรับสร้างโกสเอไอ

4.2 บันทึกข้อมูลการเล่นของผู้เล่น

เริ่มต้นให้เทสเบดบันทึกข้อมูลการเล่นของผู้เล่นลงในไฟล์หรือหน่วยความจำที่จะขอเรียกว่า บันทึกการต่อสู้ของผู้เล่น (Battle log file) ข้อมูลที่จะบันทึกนั้นคือข้อมูลที่จำเป็นในการนิยามหรือระบุความแตกต่างของสถานการณ์ในเกม ตัวอย่างเช่นหากจะระบุความแตกต่างของสถานการณ์ในเกมด้วยข้อมูลท่าทางและตำแหน่งของตัวละครทั้งสอง ก็ให้บันทึกมาเฉพาะข้อมูลดังกล่าวของทุก เฟรมไม่จำเป็นต้องเก็บข้อมูลอื่น รายการข้อมูลที่ควรต้องบันทึกโดยพื้นฐานของเกมการต่อสู้ทั่วไปนั้นได้แก่ ตำแหน่ง ท่าทางและแถบพลังของตัวละครทั้งสอง แต่อาจจะปรับเปลี่ยนได้ตามสภาพระบบเกมการต่อสู้แต่ละเกม ขั้นตอนจากนี้ไปจะเป็นงานที่ทำในโปรแกรมสำหรับสร้างโกสเอไอ

4.3 **เตรียมข้อมูลการจัดกลุ่มเฟรมของอนิเมชั่น** (Load required database)

สถานการณ์ที่จะถูกนำมาสร้างเป็นเคส คือสถานการณ์เมื่อตัวละครเปลี่ยนกลุ่มท่าทาง ดังนั้นโปรแกรมสำหรับสร้างโกสเอไอจึงมีความจำเป็นต้องการรู้ว่าท่าอนิเมชั่นแต่ละเฟรมของตัวละครนั้นถูกจัดอยู่ในกลุ่มท่าใด จึงจะสามารถรู้ได้ว่าการเปลี่ยนท่าที่ทำให้การเปลี่ยนกลุ่มเกิดเมื่อไร ตัวอย่างการจัดกลุ่มท่าสำหรับตัวละครริวในเกมสตรีทไฟท์เตอร์ซีโร่สามแสดงในรูปที่ 13



รูปที่ 13 ตัวอย่างการจัดกลุ่มของแต่ละอนิเมชั่นเฟรม

ในรูปที่ 13 อนิเมชั่นเฟรมหมายเลข 0 ถึง 6 ถูกจัดให้เป็นกลุ่มเดียวกันคือท่ายืน และเฟรมที่ 707 ถึง 713 ถูกจัดเป็นกลุ่มท่าต่อยกลาง ดังนั้นหากตัวละครยังอยู่ในท่าเฟรมที่ 0 ถึง 6 จะถือว่าอยู่ในกลุ่มท่าเดียวกันหมด ท่าเฟรมที่ 0 และ 707 นั้นหากมองด้วยตาเปล่าแล้วจะเห็นเหมือนกันไม่สามารถแยกแยะได้ แต่การจัดกลุ่มทำให้แม้ว่าเฟรมนั้นปรากฏมาเพียงเฟรมเดียวก็ยังสามารถรู้ได้ว่าท่าที่กำลังจะออกคือท่าใด อนึ่งข้อมูลการจัดกลุ่มอนิมชั่นในลักษณะนี้ควรจะมีอยู่แล้วสำหรับกลุ่มผู้ที่พัฒนาเกมการต่อสู้ ส่วนสำหรับนักวิจัยนั้นก็ไม่เป็นการยากที่จะแจกแจงข้อมูลเหล่านี้ออกมา

4.4 **ตรวจสอบและปรับแต่งข้อมูลในบันทึกการต่อสู้**

ข้อมูลบันทึกการต่อสู้ที่เก็บมาได้อาจยังไม่อยู่ในสภาพที่พร้อมใช้ดึงเอาเคสออกมา จึงต้องทำการปรับสภาพข้อมูลก่อน สาเหตุที่ยังไม่พร้อมอาจเนื่องจากมีอนิเมชั่นที่เกิดอย่างไม่ตั้งใจหรือว่ากำกวมแฝงอยู่

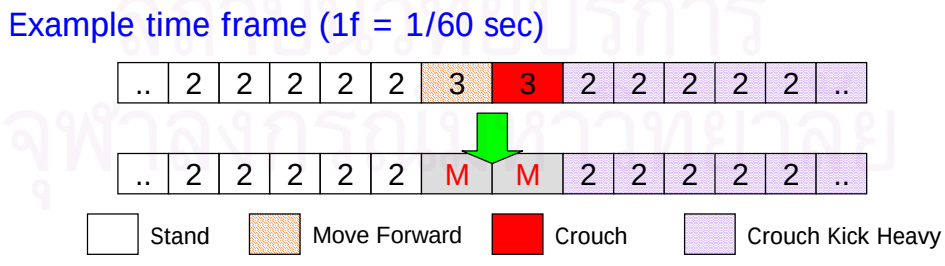
ซึ่งสมควรตัดอนิเมชั่นเหล่านั้นออก จึงต้องทำการปรับแก้ข้อมูลก่อน โดยที่การปรับแก้ข้อมูลเหล่านี้ไม่ได้เกี่ยวข้องกับการทำให้โกสเอไอที่ได้มีรูปแบบการเล่นดีกว่าตัวเจ้าของแต่อย่างใด แต่เป็นการทำเพื่อเตรียมข้อมูลให้อยู่ในสภาพที่สะดวกต่อการนำไปสร้างเคสในขั้นตอนต่อไปเท่านั้น โดยขั้นตอนย่อยของการปรับแต่งข้อมูลมีดังต่อไปนี้

4.4.1 ค้นหาอนิเมชั่นที่เกิดสั้นเกินไป (Scan and Mark Short Animation)

อนิเมชั่นที่เกิดสั้นเกินไปหมายถึงท่าที่เกิดขึ้นและจบลงเร็วกว่าระยะเวลาที่ควรจะเป็น ซึ่งหมายถึงเกิดขึ้นโดยที่ไม่ได้ตั้งใจหรือเกิดโดยปริยาย ตัวอย่างเช่นในการเปลี่ยนท่าจากท่ายืนเป็นท่านั่งเตะนั้น ผู้เล่นมักจะกดได้ท่าเดินหน้าและท่านั่งธรรมดาแทรกขึ้นมาเสี้ยววินาทีจนบางครั้งแม้แต่ผู้เล่นเองก็ไม่อาจสังเกตุเห็น ท่าเดินหน้าและท่านั่งธรรมดานั้นไม่สมควรถูกจัดการเพิ่มลงในเคส เพราะเจตนาของผู้เล่นคือจากสถานการณ์นั้นในท่ายืนผู้เล่นต้องการทำท่าก้มเตะ ไม่ได้ต้องการจะเดินหน้าหรือนั่งธรรมดา และอีกเหตุผลหนึ่งที่สมควรลบท่าเดินหน้าและท่านั่งธรรมดาออกไปก็เพื่อป้องกันการที่ท่าเหล่านี้ไปกระตุ้นให้เคสอื่นที่เก็บไว้ทำงาน ซึ่งถ้าเคสอื่นทำงาน ท่าที่ผู้เล่นตั้งใจไว้ก็อาจจะไม่ออกตอนให้โกสเอไอรัน

สาเหตุที่เกิดท่าดังกล่าวแทรกขึ้นมาเป็นเพราะการกดปุ่มที่ไม่สมบูรณ์ของผู้เล่น เช่นอาจจะมีการโดนถูกปุ่มเดินหน้าหรือว่ากดนั่งก่อนกดเตะเล็กน้อยโดยที่ไม่รู้ตัว ซึ่งจะถือว่าเป็นความผิดของผู้เล่นไม่ได้ เพราะว่าท่าแทรกนั้นสั้นมากจนเจ้าตัวไม่รู้สึกและแทบไม่ได้รบกวนการออกท่าที่ตั้งใจเลย และไม่มีผู้เล่นคนใดที่สามารถกดปุ่มได้สมบูรณ์แบบทุกครั้งจนไม่เกิดท่าแทรกเหล่านั้นขึ้นมาเลย จึงจำเป็นต้องค้นหาและทำเครื่องหมายท่าที่เกิดสั้นเหล่านั้นเอาไว้เพื่อที่จะไม่ให้ท่าเหล่านั้นถูกสร้างเป็นเคสตอนที่เกิดการเปลี่ยนกลุ่มท่า

วิธีการคือพิจารณาและนับไปทีละเฟรมของข้อมูลบันทึกการต่อสู้ ดูว่าท่าอนิเมชั่นของเฟรมนั้นอยู่ในกลุ่มไหนและเกิดมานานกี่เฟรมแล้ว หากพบว่าค่าอนิเมชั่นเกิดการเปลี่ยนกลุ่มแล้วแต่ว่ายังนับจำนวนเฟรมได้ไม่ถึงความยาวที่ควรจะเป็นก็ให้สั่งทำเครื่องหมายทุกเฟรมของกลุ่มที่เพิ่งจะนับมา ตัวอย่างแสดงในรูปที่ 14



รูปที่ 14 แผนภาพแสดงตัวอย่างการทำเครื่องหมายท่าอนิเมชั่นที่เกิดสั้น

จากภาพลายกล่องแต่ละกล่องแทนอนิเมชั่นแต่ละกลุ่ม สีขาวคือท่ายืน ลายเฉียงคือเดินหน้า สีเข้มคือท่านั่งธรรมดาและลายจุดคือท่านั่งเตะ ตัวเลขในกล่องแสดงจำนวนเวลาเป็นจำนวนเฟรมที่กล่องนั้น

เกิดท่า (ในตัวอย่าง 1 เฟรมคือเวลา 1/60 วินาที) ท่าเดินหน้าและท่านั่งเกิดและจบลงในเวลาสั้นกว่าที่กำหนดจึงต้องถูกทำเครื่องหมายไว้เป็นสีเทา เพื่อที่จะไม่ถูกพิจารณาเวลาที่สร้างเคส

ระยะเวลาที่จะใช้พิจารณาว่าสั้นหรือไม่จะต้องถูกกำหนดไว้ก่อนในฐานข้อมูล และระยะเวลาดังกล่าวสำหรับแต่ละท่าอาจจะยาว-สั้นไม่เท่ากันได้แล้วแต่พฤติกรรมของแต่ละท่า เช่นถ้ากำหนดให้ท่าเดินหน้ามีช่วงเวลาการถือว่าเป็นท่าสั้นเท่ากับ 6 เฟรม เมื่อมีท่าเดินหน้าเกิดสั้นกว่า 6 เฟรมท่านั้นจะถูกทำเครื่องหมาย แต่สำหรับท่านั่งนั้นแค่ระยะเวลาที่ใช้ย่อตัวลงนั่งก็กินเวลาไป 8 เฟรมแล้ว กล่าวคือหากเกิดท่านั่งขึ้นไม่ว่าโดยตั้งใจหรือไม่ตั้งใจก็แล้วแต่ ก็จะมีท่านั่งปรากฏขึ้นมาอย่างน้อย 8 เฟรม จึงต้องกำหนดระยะเวลาการถือว่าเป็นท่าสั้นสำหรับท่านั่งให้ยาวขึ้นเป็น 14 เฟรมเป็นต้น (เกิดจากค่าแปดที่เป็นเวลาน้อยสุดที่ท่านั่งจะเกิดได้บวกด้วยหกที่กำหนดเป็นเวลามาตรฐานในการตัดสินท่าสั้น)

4.4.2 ปรับค่าอนิเมชั่นเฟรมที่กำกวม (Fix Ambiguous Animation)

มีรูปอนิเมชั่นบางเฟรมที่ถูกใช้ร่วมกันในหลายกลุ่มอนิเมชั่น ตัวอย่างเช่นท่ากระโดดตรง กระโดดเดินหน้าและกระโดดถอยหลัง จะเริ่มด้วยเฟรมเริ่มต้นเดียวกันทั้งหมดที่เรียกว่าท่าเริ่มกระโดด สมมุติมีสถานการณ์ที่ตัวละครของผู้เล่นเปลี่ยนจากท่ายืนเป็นท่ากระโดดเดินหน้าโดยที่ในเฟรมที่ 10 ตัวละครอยู่ในกลุ่มท่ายืนเป็นเฟรมสุดท้ายและเมื่อเฟรมที่ 11 ตัวละครเริ่มอยู่ในท่าเริ่มการกระโดดซึ่งจะกลายเป็นท่ากระโดดเดินหน้าในเฟรมที่ 17 แต่เนื่องจากค่าอนิเมชั่นในเฟรมที่ 11 ซึ่งเป็นท่าเริ่มต้นของการกระโดดเดินหน้านั้นถูกใช้ร่วมกันในการกระโดดทุกแบบ ดังนั้นการที่ดูแค่เฟรม 11 เฟรมเดียวจึงไม่สามารถรู้ได้ว่าการกระโดดที่กำลังจะเกิดต่อไปนี้เป็นการกระโดดแบบใดกันแน่ จึงต้องดูข้อมูลในเฟรมถัดไปอีกสักหน่อยว่าเป็นเฟรมของการกระโดดแบบไหน แล้วจึงย้อนกลับไปกำหนดค่ากลุ่มอนิเมชั่นกระโดดแบบเดินหน้าให้กับเฟรมที่ 11 เพื่อที่จะสามารถสร้างเคสได้อย่างถูกต้องในขั้นตอนการสร้างเคสและทำให้โกสเอไอออกท่าได้ถูกจังหวะมากขึ้น

4.4.3 ตัดท่าที่ไม่ต้องการให้เกิดออก (Mark Exception Animation)

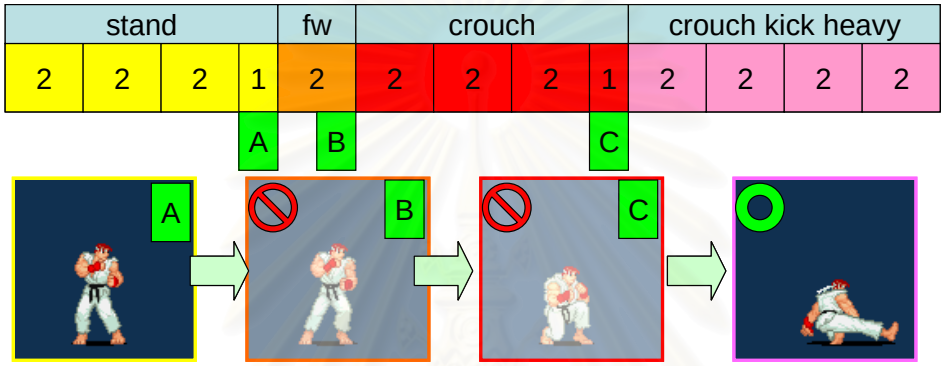
มีท่าบางท่าที่ไม่ได้เกิดจากการกระทำใดๆ เลยของผู้เล่น และเกิดนานจนไม่ถูกตัดออกด้วยขั้นตอนข้างต้น ตัวอย่างเช่นท่าบาดเจ็บทุกแบบซึ่งเกิดจากการที่โดนโจมตี ท่าเหล่านี้จะต้องถูกทำเครื่องหมายเอาไว้เพื่อที่จะไม่พิจารณาสร้างเป็นเคส เพราะไม่มีความหมายใดที่จากสถานการณ์ใดๆ จะสั่งทำท่าบาดเจ็บ ซึ่งแท้จริงแล้วสั่งทำไม่ได้ด้วย นอกจากนี้อาจจะมีท่าอื่นอีกที่ผู้สร้างระบบโกสเอไอไม่ต้องการให้โกสเอไอทำ ก็สามารถกำจัดทิ้งในขั้นตอนนี้ได้

จะเห็นได้ว่าขั้นตอน 4.4 ทั้งหมดนี้ทำเพื่อให้ข้อมูลบันทึกการต่อสู้อยู่ในสภาพที่สะดวกต่อการสร้างเคสและเพื่อให้โกสเอไอออกท่าได้ตามจังหวะเดียวกันกับผู้เล่น กลุ่มท่าที่มีการตัดออกไปเช่นท่าที่เกิดสั้นหรือท่าบาดเจ็บนั้นเป็นท่าที่ไม่ส่งผลใดๆ ต่อความเหมือนในการเล่นของโกสเอไอและตัวเจ้าของโกสเอไอนั้นเลย ส่วนกรณีท่าที่ผู้เล่นออกท่าผิดจังหวะหรือออกท่าผิดนั้นก็ยังคงอยู่ในบันทึกการต่อสู้

และจะถูกนำไปสร้างเป็นเคสเช่นเดิม เพื่อให้โกสเอไอมีการออกท่าผิดในสถานการณ์เดียวกันเช่นเดียวกับตัวเจ้าของ

4.5 **พิจารณาการสร้างเคส** (Build Case)

ขั้นตอนนี้เป็นแก่นสำคัญที่สุดของการสร้างโกสเอไอ โดยทำการพิจารณาบันทึกการต่อสู้อีกรอบหลังจากที่ได้แก้ไขข้อมูลบันทึกการต่อสู้ตามขั้นตอนในหัวข้อก่อนหน้ามาแล้ว เพื่อที่จะหาเฟรมที่เกิดการเปลี่ยนกลุ่มอนิเมชั่น และดึงเอาข้อมูลสถานการณ์ในเฟรมนั้นมาคู่กับกลุ่มอนิเมชั่นท่าที่จะเปลี่ยนไปเป็น แล้วสร้างเป็นเคสสำหรับให้โกสเอไอใช้อ้างอิงในการต่อสู้ต่อไป



รูปที่ 15 แผนภาพประกอบการแสดงตัวอย่างการสร้างเคส

จากตัวอย่างรูปที่ 15 จะเห็นว่าในเฟรมที่ 7 ตัวละครเปลี่ยนกลุ่มท่าจากท่ายืนเป็นท่าเดินหน้า เกิดการเปลี่ยนกลุ่มท่าขึ้นทำให้เกิดการสร้างเคสขึ้นในเฟรมนี้ (อันที่จริงคือเกิดการเปลี่ยนกลุ่มขึ้นในรอยต่อระหว่างเฟรมที่ 7 และ 8 แต่จะขอเรียกว่าเกิดในเฟรมที่ 7 และขอให้เข้าใจตรงกันตามนี้) สถานการณ์ในเฟรมที่ 7 จะต้องถูกนำไปสร้างเป็นเคสสำหรับโกสเอไอว่าหากเจอสถานการณ์นี้ให้ออกท่าเดินหน้า แต่ว่าท่าเดินหน้านั้นเกิดสั้นเกินไปตามตัวอย่างในหัวข้อ 4.4.1 จึงข้ามไปพิจารณากลุ่มท่าถัดไปซึ่งเป็นท่านั่งธรรมดา แต่ก็เป็นท่าที่เกิดในเวลาสั้นเกินไปอีกเช่นกัน จึงต้องข้ามไปพิจารณาท่าถัดไปอีกครั้ง คราวนี้ท่านั่งเตะไม่ได้เกิดสั้นเกินไปและไม่ได้เป็นท่าไม่พึงประสงค์ใดๆ จึงสามารถใช้ได้ สรุปคือสถานการณ์ในเฟรมที่ 7 จะถูกนำไปคู่กับท่านั่งเตะเพื่อสร้างเป็นเคสให้โกสเอไอใช้อ้างอิง

4.6 **การเข้ารหัสสถานการณ์ในเกม** (Encode)

เมื่อพิจารณาจนพบเฟรมที่เกิดการเปลี่ยนแปลงกลุ่มอนิเมชั่นแล้ว สถานการณ์ทั้งหมดของเฟรมนั้นจะต้องถูกเข้ารหัสเพื่อให้อยู่ในสภาพที่เล็กและสะดวกต่อการนำไปใช้ก่อน จึงจะถูกสร้างเป็นเคสได้ สมมุติให้สถานการณ์ในเฟรมที่ 7 ในรูปที่ 15 นั้น เป็นสถานการณ์ที่ตัวละครผู้เล่นอยู่ฝั่งซ้ายมือในท่ายืน ตัวละครศัตรูอยู่ในท่ายืนเตะเบา ระยะห่างของตัวละครทั้งสองตามแกนระนาบคือ 120 หน่วย และไม่มีกระสุนของฝ่ายใดอยู่บนจอขณะนั้นเลย หากนิยามสถานการณ์ด้วยการอธิบายเช่นนี้เวลาที่จะเปรียบเทียบสถานการณ์ใดๆ ว่าเป็นสถานการณ์เดียวกันหรือไม่ก็จะต้องเปรียบเทียบข้อมูลดังกล่าว

ทั้งหมดได้แก่ ท่าของตัวละครทั้งสอง ตำแหน่งและระยะห่างของตัวละคร และการมีอยู่รวมถึงตำแหน่งของกระสุนของตัวละครด้วย การเปรียบเทียบข้อมูลหลายอย่างเหล่านั้นแม้ว่าจะเป็นเรื่องเล็กเมื่อเทียบกับพลังประมวลผลของซีพียูสมัยนี้ก็ตาม แต่หากต้องเทียบเช่นนี้บ่อยครั้งก็เป็นการไม่สมควร และการเก็บข้อมูลแต่ละสถานการณ์จะต้องใช้เนื้อที่มากและไม่สะดวก

จึงใช้วิธีการเข้ารหัสข้อมูลสถานการณ์ทั้งหมดมาแก้ปัญหา โดยการเข้ารหัสและลดรูปสถานการณ์ทั้งหมดให้เก็บได้ในข้อมูลขนาด 32 บิต (เก็บได้ 4,294,967,296 ค่า) ซึ่งขนาดเท่ากับหนึ่งตัวแปรชนิดอินทีเจอร์ของคอมไพเลอร์ส่วนมาก การเข้ารหัสและลดรูปนั้นอย่างไรก็ย่อมบอกรายละเอียดของสถานการณ์ได้น้อยกว่าวิธีการเก็บค่าเต็มรูป ดังนั้นประเด็นสำคัญคือวิธีการเข้ารหัสจะต้องสามารถจัดกลุ่มสถานการณ์ที่ใกล้เคียงกันไว้เป็นสถานการณ์เดียวกันให้ได้ดีที่สุด เพื่อที่จะให้โกสเอไอเล่นได้เหมือนเจ้าของที่สุด หากเข้ารหัสแบบจับสถานการณ์รวมกันมากเกินไป สถานการณ์ใดๆ ก็จะถูกจับรวมเป็นสถานการณ์เดียวกันหมดผลการเล่นของโกสเอไอออกมาก็คงไม่ต่างอะไรกับการเล่นมั่ว แต่ถ้าเข้ารหัสแบบเฉพาะเจาะจงเกินไปก็จะทำให้โอกาสที่สถานการณ์ระหว่างเล่นกับที่เก็บเป็นเคสไว้เจอตรงกันน้อยลงทำให้โกสเอไอเล่นไม่ค่อยออกได้ จากการทดลองจึงได้ผลออกมาเป็นวิธีการเข้ารหัสและลดรูปสถานการณ์ดังตารางที่ 2

ตารางที่ 2 แสดงความหมายของการเข้ารหัสและลดรูปสถานการณ์ในเกมเหลือ 32 บิต

Bit no.	nBits	nValues	Meanings
1-8	8	256	Character animation set ID.
9-12	4	16	Delta position in X axis.
13-14	2	4	Delta position in Y axis.
15-18	4	16	Enemy character state.
19	1	2	Character's bullet state.
20-22	3	8	Enemy's bullet state.
23-29	7	128	Enemy damage.
30	1	2	Character side, left right.
31	1	2	Is Player at corner.
32	1	2	Is enemy at corner.

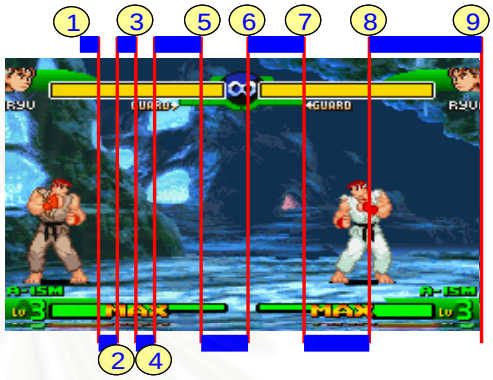
กำหนดสถานะในเกมที่ใช้นิยามหรือแบ่งแยกสถานการณ์มี 10 อย่าง ทั้ง 10 อย่างนี้คือสิ่งที่ต้องบันทึกลงในบันทึกการต่อสู้ของผู้เล่นตั้งแต่ขั้นตอน 4.2 รายละเอียดของทั้ง 10 หัวข้อและการเข้ารหัสเป็นดังนี้

4.6.1 บิตที่ 1 ถึง 8 เก็บหมายเลขกลุ่มอนิเมชั่นของตัวละครฝ่ายผู้เล่นในเฟรมนั้น หมายเลขกลุ่มอนิเมชั่นคือหมายเลขที่อ้างอิงจากข้อมูลการจัดกลุ่มอนิเมชั่น จำนวนกลุ่มอนิเมชั่นทั้งหมดของตัวละครในเกมการต่อสู้ทั่วไปแต่ละตัวอาจจะมีได้ 60 ถึง 100 กลุ่มท่าแล้วแต่ตัวละครและเกม แต่ออกแบบให้ใช้ 8 บิตที่สามารถเก็บได้ 256 ค่า สำหรับเกมในอนาคต

4.6.2 บิตที่ 9 ถึง 11 เก็บช่วงของระยะห่างของตัวละครทั้งสองตามแนวแกนระนาบ

จะไม่บันทึกระยะห่างของตัวละครทั้งสองตรงๆ ว่าห่างกันกี่หน่วย แต่จะแบ่งระยะห่างเป็นช่วง และเก็บว่าห่างกันอยู่ในช่วงไหนแทน เนื่องจากจะช่วยลดจำนวนสถานการณ์ที่แตกต่างกันไม่ให้เกิดมากเกินไป และเป็นการประหยัดพื้นที่ในการเก็บข้อมูลด้วย แนวทางการแบ่งช่วงระยะห่างแสดงในรูปที่ 16 โดยที่ตัวผู้เล่นอยู่ทางขวา ระยะห่างวัดจากตัวศัตรูทางซ้ายมือและจุดอ้างอิงตำแหน่งตัวละครคือกึ่งกลางตัวละคร

Distance (dot unit)	Range ID and Range Meanings
10-45	[1] Close attack.
46-62	[2] Throw distance.
63-75	[3] Normal attack.
76-95	[4] 25% of screen.
96-140	[5] 30% of screen.
141-190	[6] 40% of screen.
191-240	[7] 50% of screen.
241-310	[8] 62% of screen.
311-380	[9] More than 75%.



รูปที่ 16 ตัวอย่างการแบ่งช่วงระยะทางตามแกนราบ

การแบ่งช่วงของระยะนั้นไม่ควรแบ่งให้เท่ากันหมดเพราะว่ายิ่งระยะใกล้ความละเอียดของระยะจะยิ่งมีความสำคัญต่อการตัดสินใจออกท่ามากกว่าในระยะไกล จากตัวอย่างได้ทำการแบ่งระยะอย่างละเอียดในช่วงใกล้ (1 ถึง 4) และแบ่งอย่างหยาบมากเมื่อระยะไกล (8 และ 9) โดยตัวอย่างการแบ่งระยะนี้อิงระยะการออกท่าจากตัวละครริว ซึ่งในความเป็นจริงการแบ่งระยะอาจจะเปลี่ยนไปได้สำหรับตัวละครหรือเกมอื่น เพราะระยะออกท่าจะเปลี่ยนไปด้วยเช่นกัน

4.6.3 บิตที่ 13 และ 14 เก็บระยะห่างของตัวละครทั้งสองตามแนวแกนดิ่ง

ระยะห่างตามแกนดิ่งนั้นสามารถเก็บในรูปของสถานการณ์ว่าอยู่บนพื้นหรือกลางอากาศของตัวละครทั้งสองก็เพียงพอ เพราะเมื่อพิจารณาการตัดสินใจกระทำการของผู้เล่นโดยดูจากตำแหน่งตามแกนดิ่งอย่างเดียวแล้ว ผู้เล่นมักจะสนใจแต่สถานะที่ว่าใครอยู่บนพื้นใครอยู่กลางอากาศแล้วจึงออกท่าตอบสนอง ไม่ค่อยสนใจตัวเลขความสูงที่แตกต่างนัก จึงควรแบ่งระยะตามแกนดิ่งออกสี่แบบดังนี้ 1 ตัวละครทั้งสองอยู่บนพื้น 2 ผู้เล่นอยู่บนพื้นศัตรูอยู่กลางอากาศ 3 ผู้เล่นอยู่กลางอากาศศัตรูอยู่บนพื้น 4 ทั้งสองตัวละครอยู่กลางอากาศ แม้ว่าอาจดูเหมือนเป็นการแบ่งที่ไม่ละเอียด แต่เมื่อนำข้อมูลนี้ไปใช้รวมกับข้อมูลบิตอื่นๆ เช่นระยะห่างตามแกนราบและท่าทางของตัวละคร ก็จะสามารถแบ่งแยกสถานการณ์ได้เหมาะสม

4.6.4 บิตที่ 15 ถึง 18 เก็บข้อมูลสถานะของศัตรู

สถานะของศัตรูนั้นไม่จำเป็นจะต้องเก็บละเอียดถึง 8 บิตเท่ากับของผู้เล่น เพราะว่าศัตรูมีได้หลายตัวละครและโดยทั่วไปแล้วผู้เล่นจะสนใจแต่สถานะบางอย่างของศัตรูเท่านั้น นั่นคือศัตรูกำลังอยู่ในสภาพที่ถูกโจมตีได้หรือไม่ กำลังอยู่ในสภาพที่ตอบโต้ได้หรือไม่หรือกำลังทำการโจมตีอยู่หรือไม่ เมื่อแบ่งกลุ่ม

สถานะของศัตรูตามหลักนี้แล้วจะทำให้เหลือเพียง 5 สถานะเท่านั้น คือ สถานะที่ถูกโจมตีได้ สถานะที่ถูกโจมตีไม่ได้ สถานะที่กำลังมึน สถานะบาดเจ็บและสถานะที่กำลังโจมตีอยู่

4.6.5 บิตที่ 19 เก็บว่ามีกระสุนของฝ่ายผู้เล่นอยู่หรือไม่

ใช้เพียงหนึ่งบิตในการเก็บว่ามีกระสุนของผู้เล่นอยู่บนจอหรือไม่ เนื่องจากข้อมูลนี้ไม่ได้สำคัญมากนัก ผู้เล่นส่วนมากจะใช้การยิงกระสุนเพื่อรักษาระยะห่างระหว่างตัวละครหรือเพื่อวางกับดักในการออกท่าสวนเมื่อศัตรูกระโดดเข้าหา และเมื่อใช้ประกอบกับข้อมูลตำแหน่งและท่าทางของตัวละครแล้วจะสามารถแบ่งแยกสถานการณ์ได้เหมาะสม

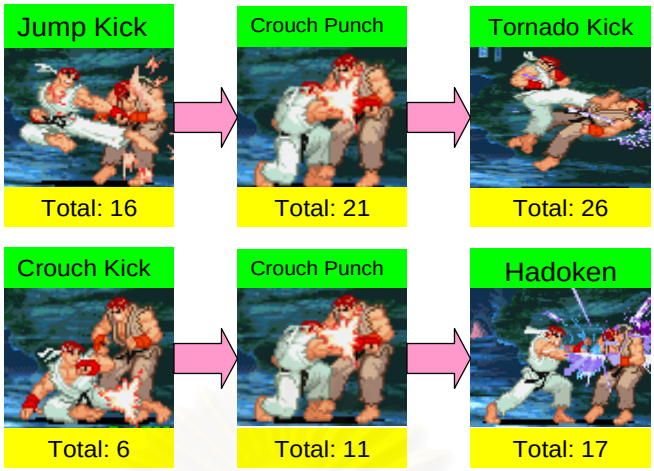
4.6.6 บิตที่ 20 ถึง 22 เก็บระยะห่างระหว่างตัวละครผู้เล่นกับกระสุนของศัตรู

กระสุนของศัตรูส่งผลต่อการตัดสินใจของผู้เล่นมากกว่ากระสุนของผู้เล่นเอง ตัวอย่างเช่นผู้เล่นอาจจะตัดสินใจที่จะยิงกระสุนสวนกลับไปหากมีเวลาและระยะห่างมากพอ หรือใช้การกระโดดบุกเข้าหาศัตรูหากว่าระยะห่างเหมาะสม หรือยกการ์ดป้องกันหากใกล้จนไม่สามารถออกท่าอื่นได้ทัน จึงออกแบบให้ใช้สามบิตเพื่อเก็บค่า โดยแบ่งช่วงระยะห่างด้วยหลักการเดียวกับระยะห่างของตัวละครตามแกนระนาบในหัวข้อ 4.6.2 อนึ่งประเด็นเรื่องกระสุนอาจจะถูกตัดทิ้งได้หากว่าตัวละครหรือเกมนั้นปล่อยกระสุนไม่ได้

4.6.7 บิตที่ 23 ถึง 29 เก็บค่าการบาดเจ็บของศัตรูในเฟรมนั้น

ข้อมูลนี้เป็นเทคนิคพิเศษที่ใช้เพื่อแบ่งแยกสถานการณ์ที่หากดูเฉพาะเวลานั้นเฟรมเดียวแล้วจะเหมือนกัน แต่ว่ามีที่มาในอดีตต่างกัน นั่นคือสถานการณ์ความต่อเนื่องของการทำคอมโบนั่นเอง สมมุติว่ามี 2 สถานการณ์ ที่มีสภาพเหมือนกันทุกอย่าง โดยที่ตัวละครผู้เล่นอยู่ในท่าก้มต่อยกลางและศัตรูอยู่ในท่าบาดเจ็บ แต่ว่าสถานการณ์แรกเกิดมาจากการทำคอมโบของท่ากระโดดเตะ ก้มต่อยกลาง ซึ่งต่อไปจะปิดคอมโบด้วยท่าเตะพายุหมุน ส่วนอีกสถานการณ์หนึ่งเกิดจากคอมโบก้มเตะกลาง ก้มต่อยกลางและจะปิดด้วยท่ายิงพลังฮาโดเคน รูปที่ 17 แสดงภาพประกอบการยกตัวอย่างนี้ จากภาพเป็นการทำคอมโบสองแบบที่ท่าจังหวะที่สองเหมือนกัน แต่สามารถบอกความแตกต่างของคอมโบได้โดยดูค่าความบาดเจ็บของศัตรูที่ไม่เท่ากัน ท่ากระโดดเตะทำความเสียหาย 16 หน่วยแต่ท่าก้มเตะทำความเสียหายเพียง 6 หน่วย

ลำดับการทำคอมโบนั้นถือเป็นเรื่องสำคัญอย่างหนึ่งที่บ่งบอกตัวตนของผู้เล่นเกมการต่อสู้ โกสเอไอจึงควรทำตามลำดับนั้นด้วย ดังนั้นจากในสถานการณ์ก้มต่อยทั้งสองดังกล่าวจึงต้องมีข้อมูลบางอย่างมาใช้บอกความแตกต่างของที่มา ซึ่งก็คือข้อมูลการบาดเจ็บของศัตรูนั่นเอง เนื่องจากท่าเริ่มต้นของคอมโบนั้นต่างกันจึงทำให้ค่าความบาดเจ็บต่อเนื่องในเวลานั้นต่างกันด้วย ทำให้สามารถใช้บ่งบอกความแตกต่างของสถานการณ์ได้ และแม้ว่าจะมีโอกาสที่ค่าบาดเจ็บจะเท่ากันได้อีกแต่ก็เป็นไปได้น้อยมาก และในทางปฏิบัติจริงแล้วสามารถใช้ค่าบาดเจ็บที่กำหนดขึ้นเองเพื่อลดโอกาสการซ้ำกันลงได้อีก



รูปที่ 17 แสดงการทำคอมโบสองแบบที่ท่าจังหวะที่สองเหมือนกัน

4.6.8 บิตที่ 30 บอกว่าตัวละครผู้เล่นอยู่ทางซ้ายหรือขวา

ฝั่งของตัวละครส่งผลกับรูปแบบการเล่นของผู้เล่นเช่นกัน ผู้เล่นมือใหม่มักจะกดท่าพิเศษติดเฉพาะเวลาที่อยู่ฝั่งซ้ายเนื่องจากนิยมซ้อมมาจากฝั่งนั้นเท่านั้น หรือแม้แต่ผู้เล่นที่ชำนาญก็มีอัตราการทำคอมโบยากๆ ลดลงได้เมื่ออยู่ในฝั่งที่ไม่ถนัด

4.6.9 บิตที่ 31 บอกว่าผู้เล่นอยู่ที่มุมขอบฉากหรือไม่

การอยู่ที่มุมเป็นความเสียเปรียบอย่างหนึ่งของเกมการต่อสู้เพราะว่าไม่มีที่ให้ถอยหนีอีกแล้ว อีกทั้งมุมยังช่วยกันตำแหน่งตัวละครไม่ให้กระเด็นหนีไปได้เมื่อถูกโจมตี ทำให้คอมโบที่รุนแรงและยาวบางชุดสามารถทำได้เมื่อต้อนศัตรูเข้ามุมแล้วเท่านั้น ดังนั้นการกระทำของผู้เล่นเมื่อเข้ามุมจึงมักต่างจากการกระทำปรกติในสถานการณ์เดียวกัน

4.6.10 บิตที่ 32 บอกว่าศัตรูอยู่ที่มุมขอบฉากหรือไม่

เช่นเดียวกัน เวลาที่ศัตรูอยู่ที่มุมก็จะเป็นโอกาสอันดีที่ผู้เล่นจะทำคอมโบได้

ตัวอย่างการเข้ารหัสโดยรวมจากรูปที่ 16 ฝั่งขวามือจะเป็นดังนี้

บิตที่1-8 เป็น 0 หมดเพราะว่าตัวละครผู้เล่นอยู่ในท่ายืนซึ่งมีหมายเลขกลุ่มอนิเมชั่นเป็น 0

บิตที่ 9-12 เป็น 0111 หรือ 7 เพราะว่าระยะห่างอยู่ในช่วงที่ 8 แต่เริ่มนับที่ 0

บิตที่ 13-14 เป็น 0 หมด เพราะว่าตัวละครทั้งสองอยู่บนพื้น

บิตที่ 15-18 เป็น 0 หมด เพราะศัตรูอยู่ในสถานะที่ถูกโจมตีได้ตามปรกติและไม่ได้ออกท่า

บิตที่ 19-22 เป็น 0 หมด เพราะไม่มีกระสุนของฝ่ายใดอยู่เลย

บิตที่ 23-29 เป็น 0 หมดเพราะว่าศัตรูไม่ได้รับบาดเจ็บ

บิตที่ 30 เป็น 1 เพราะว่าผู้เล่นอยู่ทางขวา

บิตที่ 31 เป็น 0 เพราะว่าผู้เล่นไม่ได้อยู่ที่มุม

บิตที่ 32 เป็น 1 เพราะว่าศัตรูอยู่ที่มุม

และเมื่อนำเอาค่าของทั้ง 32 บิต ทั้งหมดข้างต้นมาเรียงต่อกันแล้วก็จะได้เป็นค่าเท่ากับ 10100000000000000000011100000000 ซึ่งก็คือ 2,684,356,352 ในระบบเลขฐานสิบนั่นเอง

ทั้งหมดนั้นคือแนวทางและตัวอย่างวิธีการเข้ารหัสและลดรูปสถานการณ์อันมากมายในเกมลงเก็บในข้อมูลขนาด 32 บิต รายละเอียดของวิธีการสามารถเปลี่ยนแปลงได้เพื่อให้เหมาะสมกับรูปแบบเกมหรือแต่ละตัวละครในเกม หากการเข้ารหัสขนาด 32 บิต ไม่เพียงพอก็สามารถเปลี่ยนวิธีหรือขยายขนาดข้อมูลเข้ารหัสเพิ่มเติมได้ จะเห็นได้ว่าแนวคิดนี้ไม่ได้มีการเสนออะไรที่เฉพาะเจาะจงกับตัวละครหรือเกมเลย นอกจากเป็นการใช้เพื่อยกตัวอย่างเพิ่มความเข้าใจเท่านั้น หลักการของแนวคิดนี้ทั้งหมดเป็นสิ่งที่ใช้ได้กับเกมการต่อสู้ทั่วไปทั้งสิ้น

4.7 **การรวบรวมเคสสำหรับโกสเอไอ** (Save ghost AI file)

เมื่อได้เคสจากการเข้ารหัสสถานการณ์และจับคู่เข้ากับการกระทำที่จะทำในสถานการณ์นั้นแล้ว ให้นำเคสเหล่านั้นไปเก็บไว้ในโครงสร้างข้อมูลชนิดที่สะดวกต่อการค้นหาและการแทรก เนื่องจากระหว่างการสร้างเคสนั้น เมื่อสร้างแต่ละเคสเสร็จก็จะต้องตรวจสอบว่าสถานการณ์ในเคสนั้นเคยถูกสร้างเป็นเคสแล้วหรือยัง ถ้าเคยแล้วก็เพียงเพิ่มเติมหรืออัพเดทข้อมูลในส่วนของการกระทำในเคสนั้นเท่านั้น แต่ถ้าไม่เคยมีสถานการณ์นั้นมาก่อนก็ให้เพิ่มเคสนั้นเข้าไปในโครงสร้างข้อมูล ตัวอย่างเช่นหากเดิมมีการเก็บเคสนี้เอาไว้

```
SituationID: 0000000000
TotalRatio: 03 TotalNextAni: 02
    NextAni: Punch-Light-Close Ratio 2
    NextAni: Kick-Heavy-Close  Ratio 1
```

เคสนี้มีความหมายคือ ในสถานการณ์ที่ 0000000000 (SituationID คือหมายเลขของสถานการณ์ซึ่งเป็นค่าที่ได้จากการเข้ารหัสสถานการณ์เป็น 32 บิต) ผู้เล่นเคยเจอสถานการณ์นี้มาทั้งหมด 3 ครั้ง (TotalRatio=3) เขาเลือกกระทำการสองอย่างที่ต่างกัน (TotalNextAni=2) โดยที่เลือกต่อยเบา 2 ครั้ง และเลือกที่จะเตะหนัก 1 ครั้ง (แสดงตามค่า Ratio) ถ้าหากว่ามีการสแกนข้อมูลบันทึกการเล่นเพิ่มแล้วพบว่ามีเคสที่สถานการณ์เป็น 0 และผู้เล่นเลือกที่จะเตะหนักก็จะทำการปรับปรุงข้อมูลโดยเพิ่มจำนวนครั้งซึ่งเกิดการเตะหนักดังนี้

```
SituationID: 0000000000
TotalRatio: 04 TotalNextAni: 02
    NextAni: Punch-Light-Close Ratio 2
    NextAni: Kick-Heavy-Close  Ratio 2
```

ต่อมาสมมุติว่าเจอเคสที่สถานการณ์เดียวกันอีกครั้ง แต่ผู้เล่นเลือกที่จะยิงกระสุนพลังฮาโดเคนแทน ก็จะทำการปรับปรุงข้อมูลโดยเพิ่มการกระทำของการใช้ท่าฮาโดเคนเข้าไป

```
SituationID: 0000000000
TotalRatio: 05 TotalNextAni: 03
    NextAni: Punch-Light-Close Ratio 2
    NextAni: Kick-Heavy-Close  Ratio 2
    NextAni: Hadoken           Ratio 1
```

และหากมีการสร้างเคสใหม่ขึ้นมาก็จะเพิ่มเคสลงไปอีกได้ผลเป็นดังนี้

```
SituationID: 0000000000
TotalRatio: 05 TotalNextAni: 03
    NextAni: Punch-Light-Close Ratio 2
    NextAni: Kick-Heavy-Close  Ratio 2
    NextAni: Hadoken Heavy     Ratio 1
SituationID: 2684356352
TotalRatio: 01 TotalNextAni: 01
    NextAni: Hadouken          Ratio 1
```

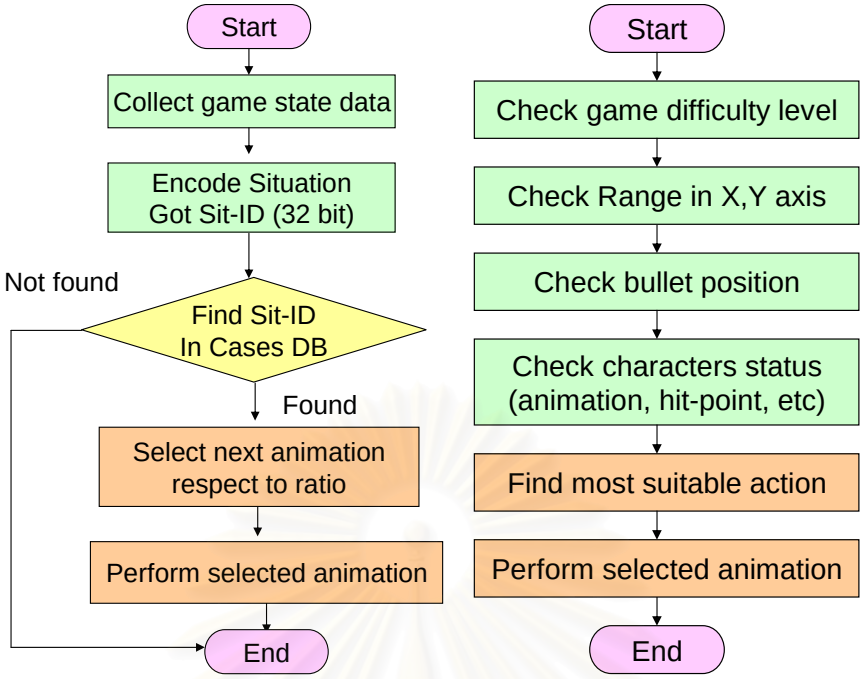
จากตัวอย่างข้างต้นจะเห็นได้ว่าการเก็บข้อมูลเคสในโครงสร้างข้อมูลที่เหมาะสมเป็นสิ่งที่ส่งผลต่อความเร็วในการทำงานมาก เพราะจะต้องมีการค้นอยู่ตลอดว่าเคสนั้นมีหรือไม่และมีการเพิ่มเคสใหม่ลงไปตลอดการสร้างโกสเอไอ ตัวอย่างโครงสร้างข้อมูลที่สามารถนำมาใช้ได้คือ ต้นไม้ค้นแบบทวิภาค (Binary Search Tree) หรือ ตารางแฮช (Hash Table) เพราะว่าสามารถทำการค้นหาและเพิ่มข้อมูลได้ในเวลารวดเร็ว

ขั้นตอนสุดท้ายของการสร้างโกสเอไอคือการบันทึกข้อมูลเคสทั้งหมดที่เก็บอยู่ในโครงสร้างข้อมูลลงในไฟล์ การบันทึกข้อมูลลงไฟล์ทำให้สามารถนำไฟล์โกสเอไอของผู้เล่นหลายไฟล์มาผสานรวมกันได้ด้วย

4.8 การใช้งานโกสเอไอ

การนำโกสเอไอในรูปแบบของเคสไปใช้ทำโดยการ เริ่มโหลดข้อมูลเคสจากไฟล์กลับเข้าไปเก็บในโครงสร้างข้อมูลที่สะดวกต่อการค้นหาเคสจากคีย์ (ซึ่งก็คือข้อมูลการเข้ารหัสสถานการณ์ 32 บิต) จากนั้นทุกรอบการทำงานของโกสเอไอจะทำงานตามแผนภูมิสายงานในรูปที่ 18 ฝั่งซ้ายเท่านั้น ซึ่งเริ่มโดยการเข้ารหัสสถานการณ์ในเกมขณะเฟรมนั้นด้วยวิธีเดียวกันกับที่ใช้สร้างเคสของโกสเอไอ และนำรหัสสถานการณ์มาค้นหาเคสจากในโครงสร้างข้อมูลว่ามีอยู่หรือไม่ ถ้าหากไม่มีก็ไม่ต้องทำอะไรรอรอบต่อไป แต่ถ้าหากเจอก็ให้เลือกการกระทำที่บันทึกเอาไว้ในเคสนั้นมาทำแบบสุ่มโดยอิงจากอัตราส่วนของท่าเหล่านั้น และท้ายสุดจึงสั่งออกท่า

รูปที่ 18 ฝั่งขวาเป็นการแสดงการทำงานของเอไอเดิมที่มีมากับเกมการต่อสู้ทั่วไป การทำงานของเอไอเดิมของเกมคือในทุกรอบการทำงานจะทำการตรวจสอบสถานการณ์ต่างๆ ในเกมขณะนั้นเช่น ระดับความยากของเกม ระยะห่างของตัวละคร ตำแหน่งกระสุน ท่าทางของตัวละคร พลังชีวิตที่เหลือ และพลังท่าไม้ตายเป็นต้น จากนั้นจะทำการตัดสินเลือกออกท่าที่เหมาะสมที่สุดโดยดูระดับความยากของเกมที่ผู้เล่นตั้งค่าเอาไว้ประกอบด้วย หากเกมเป็นแบบง่ายท่าที่เหมาะสมที่สุดในการออกอาจจะเป็นท่าที่เปิดช่องโหว่ให้ผู้เล่นโจมตีได้แทน หรือหากท่าที่เหมาะสมที่สุดมีหลายท่าเสมอกันก็จะใช้วิธีการสุ่มเข้ามาตัดสิน และท้ายสุดก็สั่งออกท่านั้น



รูปที่ 18 แผนภูมิสายงานแสดงการทำงานของโกสเอไอ(ซ้าย) และเอไอธรรมดาของเกม(ขวา)

เมื่อเปรียบเทียบการทำงานของโกสเอไอและเอไอเดิมของเกมแล้วจะพบว่า ทั้งสองทำงานด้วยหลักการเดียวกัน คือในช่วงแรกของรอบการทำงานจะรวบรวมข้อมูลสถานการณ์ในเกมก่อน ซึ่งข้อมูลหลักที่สนใจนั้นเหมือนกันคือ ตำแหน่ง ระยะห่างและท่าทางของตัวละครเป็นต้น จากนั้นจะดำเนินการหาท่าที่เหมาะสมที่สุด ตรงขั้นตอนนี้ทั้งสองเอไอจะใช้วิธีที่แตกต่างกัน เอไอเดิมของเกมจะพิจารณาจากปัจจัยระดับความยากง่ายของเกมประกอบกับสถานการณ์ในเกมแล้วออกท่าที่เหมาะสมที่สุด ส่วนโกสเอไอจะพิจารณาว่าสถานการณ์ในเกมขณะนั้นเคยเจอมาก่อนหรือไม่ แล้วจึงออกท่าตามสัดส่วนการออกท่าที่บันทึกเอาไว้ จะเห็นได้ว่าวิธีการทำงานของโกสเอไอเป็นไปในแนวทางเดียวกับเอไอเดิมของเกมและสามารถปรับเปลี่ยนการทำงานของเอไอเดิมของเกมให้เป็นโกสเอไอได้โดยง่าย

4.9 การนำวิธีการสร้างและการใช้งานโกสเอไอไปใช้กับเกมการต่อสู้อื่น

การนำวิธีการสร้างโกสเอไอในงานวิจัยนี้ไปใช้กับเกมการต่อสู้เกมอื่นนั้น ขอให้พิจารณาขั้นตอนการเตรียมข้อมูลและวิธีสร้างโกสเอไอและปรับให้เหมาะสมดังนี้

ประเด็นแรกคือฐานข้อมูลการจัดกลุ่มเฟรมของอนิเมชั่น ถ้าหากยังไม่มีก็จะต้องเตรียมไว้ เพราะวิธีการสร้างโกสเอไอจะค้นหาและดึงเอาสถานการณ์ที่มีการเปลี่ยนกลุ่มท่าของตัวละครผู้เล่นออกมาสร้างเคส ดังนั้นจะต้องมีฐานข้อมูลที่ใช้บอกกลุ่มของท่าแต่ละเฟรมของตัวละครในเกม ข้อมูลนี้เป็นสิ่งที่เกมการต่อสู้ทุกเกมจะต้องมีอยู่แล้วเพื่อใช้ในการแสดงอนิเมชั่นของตัวละครในเกม ถ้าสามารถเข้าถึงข้อมูลดังกล่าวได้ก็นำมาปรับใช้ได้เลย แต่หากไม่อาจเข้าถึงได้ก็จะต้องเตรียมสร้างขึ้นมาก่อน

ดังเช่นงานวิจัยนี้ การแบ่งกลุ่มของเฟรมออกเป็นกลุ่มของอนิเมชั่นที่เหมาะสมนั้นมีส่วนสำคัญต่อผลการสร้างเคสจึงควรแบ่งให้เหมาะสม

ประเด็นที่สองคือการบันทึกข้อมูลการเล่นของผู้เล่นในไฟล์บันทึกการต่อสู้ ให้บันทึกทุกข้อมูลที่จำเป็นต่อการแบ่งแยกสถานการณ์และสร้างเคสสำหรับแต่ละเกม ตัวอย่างเช่นหากจะทดลองกับเกมเรียลเบ้าท์สเปเชียล (Realbout Special) ซึ่งมีระบบพิกัดแกนความลึกด้วย ก็ควรต้องเก็บค่าพิกัดดังกล่าวเพิ่มลงไปด้วย

ประเด็นที่สามคือการแสกนบันทึกการต่อสู้เพื่อตัดท่าที่ไม่เป็นประโยชน์ต่อการสร้างเคส ออกสามารถเพิ่มหรือลดการแสกนได้ตามรูปแบบของเกม ตัวอย่างเช่นหากเกมนั้นมีการจัดเฟรมและกลุ่มอนิเมชั่นของการกระโดดแต่ละแบบด้วยหมายเลขเฟรมที่แตกต่างกันทั้งหมดไม่มีการใช้เฟรมเริ่มกระโดดร่วมกัน ก็ไม่จำเป็นจะต้องแสกนเพื่อระบุแก้ไขท่ากระโดดว่าเป็นการกระโดดแบบใด

ประเด็นที่สี่คือการเข้ารหัสสถานการณ์ในเกม การเข้ารหัสเป็นเรื่องสำคัญที่จะต้องมีการกำหนดให้เหมาะสมกับเกมและตัวละคร ตัวอย่างเช่นการจัดช่วงระยะห่างตามแกนราบจากรูปที่ 16 นั้นจะเห็นได้ว่ามีการอิงระยะตามระยะการออกท่าของตัวละครริว ซึ่งความจริงแล้วตัวละครแต่ละตัวจะมีระยะออกท่าแตกต่างกันไป จึงจำเป็นจะต้องมีการปรับวิธีการเข้ารหัสสถานการณ์ไปตามตัวละคร และหากบางเกมมีจำนวนข้อมูลที่สำคัญต่อการสร้างเคสจำนวนมากหรือต้องการการเข้ารหัสที่ละเอียดมากขึ้น วิธีการเก็บค่ารหัสสถานการณ์แบบใช้ 32 บิตอาจจะไม่เพียงพอ ก็สามารถขยายจำนวนบิตเพิ่มได้อีก ซึ่งอาจจะใช้ตัวแปรชนิด 64 บิตเก็บค่า หรืออาจจะใช้สตริงความยาวไม่จำกัดก็ได้

ประเด็นที่ห้าคือการเลือกใช้โครงสร้างข้อมูลที่เหมาะสมในการเก็บเคส โครงสร้างข้อมูลนั้นควรจะมีความสามารถในการแทรกและค้นข้อมูลได้เร็ว ความเร็วในการค้นและแทรกจะขึ้นอยู่กับรูปแบบของเคสและผลของการเข้ารหัสสถานการณ์ในเกม ซึ่งต้นไม้ค้นทวิภาคดังที่ใช้ในการทดลองนี้ถือเป็นตัวเลือกที่ดีตัวหนึ่ง ตารางแฮชก็เป็นอีกตัวเลือกหนึ่งที่ดีแต่ว่าจะต้องมีภาระในการกำหนดแฮชฟังก์ชันที่เหมาะสมเพิ่มเข้ามา จึงไม่ได้ใช้ในการทดลองนี้

การรันโกสเอไอเวลาทำงานจริงนั้นใช้ขั้นตอนในรูปที่ 18 ฝั่งซ้ายได้เช่นเดิม

กระบวนการสร้างและใช้งานโกสเอไอที่ได้เสนอในบทนี้เป็นแนวทางที่สามารถนำไปปรับใช้กับเกมการต่อสู้ทั้งหมด ดังตัวอย่างประกอบซึ่งแสดงด้วยเกมสตรีทไฟเตอร์ซีโร่สามที่เป็นเกมที่มีรูปแบบเกมเป็นมาตรฐานสำหรับเกมการต่อสู้ในปัจจุบัน

บทที่ 5

การพัฒนาต้นแบบเทสเบด ผลการทดลองและการวิเคราะห์ผล

บทนี้จะอธิบายวิธีการพัฒนาต้นแบบเทสเบดจากการปรับแต่งวีบีเออีมูเลเตอร์ให้เหมาะสมจนได้เป็นไอเทมเทสเบด วิธีการสร้างโกสเอไอและรันด้วยไอเทม และช่วงท้ายจะกล่าวถึงผลการทำงานและการวิเคราะห์ผล

5.1 การพัฒนาต้นแบบเทสเบด

วิธีการปรับแต่งวีบีเออีมูเลเตอร์จนได้เป็นไอเทมเทสเบดนั้นทำได้ดังต่อไปนี้

5.1.1 วีบีเอโปรเจค

วีบีเอเป็นโอเพนซอร์สอีมูเลเตอร์ โปรเจคซอร์สโค้ดของวีบีเอมีการเผยแพร่ในหลายรูปแบบ รูปแบบที่สะดวกต่อการนำมาใช้ที่สุดคือโปรเจคของไมโครซอฟท์วิชวลสตูดิโอรุ่น 7.1 แต่จะต้องทำการเพิ่มการใช้เอ็นเอเอสเอ็ม (NASM) ซึ่งเป็นคอมไพเลอร์ส่วนภาษาแอสเซมบลีเข้าไปในพาร์ทการทำคัสตอมบิลท์ด้วย เนื่องจากผู้พัฒนาวีบีเอระบุให้มีการเรียกใช้ และเนื่องจากมีความต้องการใช้ไพธอนสคริปต์จึงต้องอินคลูดไพธอนเฮดเดอร์กับไลบรารี่เพิ่มลงไปในโปรเจค และวางไพธอนดีแอลแอลไว้ในที่ที่โปรแกรมทำงานด้วย

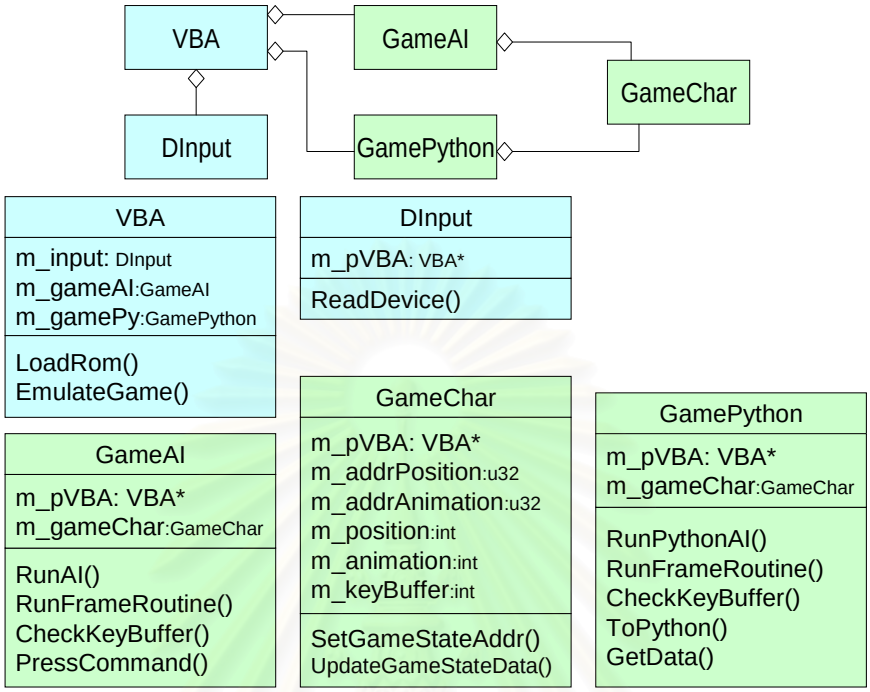
5.1.2 ค้นหาแอดเดรสของข้อมูลที่ต้องการใช้ในเกมสตรีทไฟท์เตอร์ซีโร่สาม

รายการข้อมูลที่ต้องการใช้ในการทดสอบเอไอในเกม สตรีทไฟท์เตอร์ซีโร่สาม ได้แก่ ตำแหน่ง ท่าทางของตัวละคร พลังชีวิต เกจความสามารถพิเศษของตัวละครทั้งสอง และตำแหน่งกระสุนที่ทั้งสองฝ่ายปล่อยออกมา แอดเดรสของข้อมูลทั้งหมดสามารถหาได้ด้วยวิธีการทำให้ค่าของข้อมูลนั้นเปลี่ยนขึ้นลงไปเรื่อยๆ แล้วค้นในเมมโมรีเพื่อหาค่าที่เปลี่ยนขึ้นลงอย่างสอดคล้องกันประกอบกับการกำหนดขนาดของข้อมูลที่จะค้นให้เหมาะสมด้วย ข้อมูลที่หาแอดเดรสได้ยากคือตำแหน่งของกระสุนเนื่องจากเมื่อปล่อยออกมาแล้วกระสุนจะพุ่งออกไปโดยอัตโนมัติ ไม่สามารถควบคุมตำแหน่งได้ดังใจ ทำให้การค้นหาไม่สะดวก ส่วนแอดเดรสที่บอกท่าทางของตัวละครนั้นสามารถหาเจอได้ไม่ยากเนื่องจากอยู่ติดกับแอดเดรสที่บอกตำแหน่งของตัวละครตามที่ควรจะเป็น

5.1.3 การปรับปรุงวีบีเอเป็นไอเทม

การปรับปรุงอีมูเลเตอร์วีบีเอเป็นไอเทมนั้นเริ่มโดยการวางโครงสร้างดังแสดงผ่านคลาสไดอะแกรมในรูปที่ 19 แต่เดิมวีบีเอนั้นประกอบไปด้วยคลาสจำนวนมาก แต่เนื่องจากคลาสส่วนใหญ่ไม่ได้เกี่ยวข้องกับการทำงานของไอเทมจึงไม่ขอกล่าวถึง มีเพียงคลาสสำคัญสองคลาสเท่านั้น คือคลาส VBA และ DInput ที่เกี่ยวข้องกับการพัฒนาเทสเบด ส่วนคลาส GameAI GameChar และ GamePython นั้น

เป็นคลาสที่พัฒนาเพิ่มเข้ามาเพื่อที่จะใช้ดัดแปลงอีมูเลเตอร์เป็นเทสเบด โดยที่ตัวแปรสมาชิกและเมท็อดของทุกคลาส (ทั้งคลาสเดิมและคลาสที่เพิ่มเข้าไป) เป็นแบบสาธารณะทั้งหมด



รูปที่ 19 คลาสไดอะแกรมของไอเทมและวีบีเอ

คลาส VBA เป็นคลาสหลักของอีมูเลเตอร์ที่มีเพียงหนึ่งอินแสตนซ์ ทุกคลาสที่ใช้ในการทำงานของอีมูเลเตอร์จะเป็นตัวแปรสมาชิกของคลาสนี้ทั้งหมด และทุกคลาสที่เป็นตัวแปรสมาชิกของคลาสนี้จะมีพอยเตอร์สำหรับชี้กลับมาให้รู้จักคลาส VBA นี้ได้ ประกอบกับการที่ตัวแปรสมาชิกและเมท็อดของทุกคลาสในอีมูเลเตอร์เป็นชนิดสาธารณะที่ให้คลาสอื่นเรียกใช้ได้ ผลคือทำให้คลาสใดๆ สามารถเข้าถึงและเรียกใช้ตัวแปรสมาชิกและเมท็อดของคลาสอื่นทุกคลาสได้ทั้งหมด คลาส VBA นี้คือส่วนอีมูเลเตอร์จากโครงสร้างในรูปที่ 5 นั่นเอง

คลาส DInput เป็นคลาสดั้งเดิมของอีมูเลเตอร์ มีเมท็อดสำคัญคือ `ReadDevice()` ใช้อ่านอินพุตที่ใช้ในการควบคุมการเล่นเกมเช่นจากคีย์บอร์ดหรือคอนโทรลเลอร์และนำส่งเข้าไปยังคลาส VBA เพื่อจำลองการเล่นเกม ซึ่งเมท็อดนี้ได้ถูกดัดแปลงให้ส่งสัญญาณคอนโทรลเลอร์ที่ผู้ใช้เทสเบดต้องการสั่งกดจากคลาส GameAI เข้าไปแทนสัญญาณเดิมได้ด้วย คลาสนี้คืออินพุตคอนโทรลเลอร์จากโครงสร้างในรูปที่ 5

สามคลาสที่จะกล่าวถึงต่อไปนี้เป็นอินเทอร์เฟสที่ถูกสร้างขึ้นใหม่เพื่อใช้สร้างเทสเบด ใช้เชื่อมต่อระหว่างส่วนของเอไอกับเทสเบด โดยที่คลาสเหล่านี้ถูกออกแบบมาให้ใช้ทดสอบเอไอทั่วไปสำหรับเกม สตรีทไฟท์เตอร์ซีโร่สาม ไม่เฉพาะแต่โกสเอไอเท่านั้น

คลาส GameChar: คลาสนี้เป็นคลาสที่ใช้เก็บข้อมูลของตัวละครในเกม มีตัวแปรสมาชิกที่ใช้เก็บข้อมูลของตัวละครเช่นตำแหน่งและท่าทางของตัวละคร เมท็อดสำคัญคือ `SetGameStateAddr()` ใช้เพื่อระบุว่าจะให้อ่านข้อมูลสถานการณ์เกมจากแอดเดรสไหนของอีมูเลเตอร์มาเก็บไว้ในตัวแปรสมาชิก และเมท็อด `UpdateGameStateData()` ทำหน้าที่อ่านข้อมูลจากแอดเดรสที่ระบุไว้มาเก็บในตัวแปรสมาชิก ดังนั้นสองเมท็อดนี้รวมกันจึงทำหน้าที่เป็นเกมเสตทออบเซิร์ฟเวอร์จากโครงสร้างในรูปที่ 5 นอกจากนี้ยังมีตัวแปรสมาชิกที่น่าสนใจที่ใช้ในการเก็บสัญญาณคอนโทรลเลอร์ที่ตัวละครตัวนี้จะกดด้วยได้แก่

```
int  m_keyBuffer[KEY_BUFFER_SIZE]
bool m_busy
int  m_nextKBI
int  m_KBStock
```

ตัวแปรสมาชิกกลุ่มนี้ใช้เพื่อทำการสั่งกดคอนโทรลเลอร์เพื่อควบคุมตัวละคร `m_keyBuffer` เป็นอาร์เรย์ที่เก็บสัญญาณคอนโทรลเลอร์ที่จะกดในแต่ละเฟรมจากนี้เป็นต้นไป อาร์เรย์นี้ถูกกำหนดให้มีขนาด `KEY_BUFFER_SIZE` ซึ่งเท่ากับ 1024 นั่นคือระบบจะสามารถสั่งกดคอนโทรลเลอร์ล่วงหน้าได้ถึงหนึ่งพันยี่สิบสี่เฟรม ตัวอย่างการใช้งานระบบสั่งกดคอนโทรลเลอร์เป็นดังนี้ เมื่อต้องการสั่งให้ตัวละครกดท่าพิเศษฮาโดเคนซึ่งมีวิธีการกดคือ ลง ลงเฉียงเดินหน้าและก็เดินหน้าตามด้วยปุ่มต่อย ก็สามารถกำหนดค่าเหล่านั้นให้แก่อาร์เรย์ `m_keyBuffer` ได้ดังตัวอย่างนี้

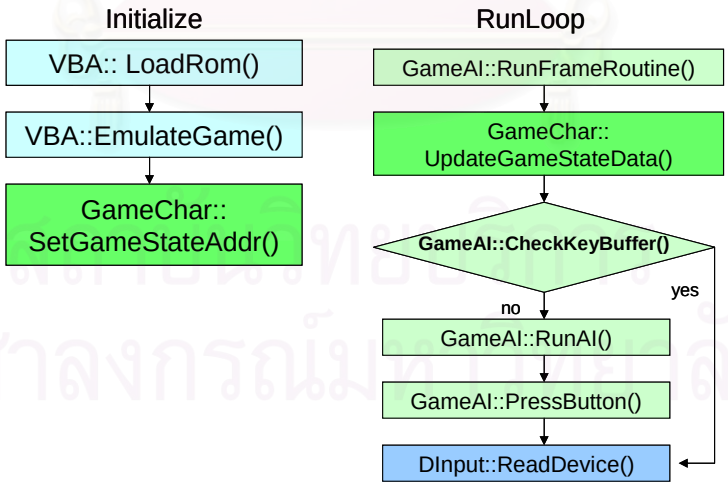
```
m_keyBuffer[0] = PRESS_DOWN;
m_keyBuffer[1] = PRESS_DOWN;
m_keyBuffer[2] = PRESS_DOWN | PressForward();
m_keyBuffer[3] = PRESS_DOWN | PressForward();
m_keyBuffer[4] = PRESS_RIGHT;
m_keyBuffer[5] = PRESS_RIGHT;
m_keyBuffer[6] = PRESS_PUNCH;
m_keyBuffer[7] = PRESS_PUNCH;

m_KBStock = 8;
m_nextKBI = 0;
m_busy = 1; // Busy
```

สังเกตได้ว่าในการสั่งงานนั้นสั่งกดสัญญาณสัญญาณละสองเฟรม เพราะการสั่งกดท่าเพียงหนึ่งเฟรมนั้นสั้นเกินไปจะทำให้มีโอกาสกดไม่ติดได้ หลังจากกำหนดสัญญาณในอาร์เรย์แล้วก็ระบุจำนวนว่ามี 8 สัญญาน (`m_KBStock = 8`) ให้เริ่มอ่านค่าที่อาร์เรย์ตำแหน่งที่ 0 (`m_nextKBI = 0`) และตั้งค่า `m_busy` เพื่อบอกว่าอาร์เรย์มีค่าอยู่พร้อมทำงานแล้ว จากนี้ไปทุกรอบการทำงานของเทสเบดหากว่าค่า `m_busy` ยังเป็น 1 อยู่ ก็จะหยิบเอาสัญญาณจาก `m_keyBuffer` ลำดับต่อไปส่งไปให้อีมูเลเตอร์แทนสัญญาณจากคีย์บอร์ดหรือคอนโทรลเลอร์ เมื่อหยิบสัญญาณไปจนหมดอาร์เรย์ ก็จะเซทค่า `m_busy` กลับเป็น 0 และจึงจะหยุดการนำสัญญาณจากอาร์เรย์ไปส่งแทน ด้วยหลักการนี้การสั่งกดคำสั่งยาวต่อเนื่องก็จะสามารถทำได้

คลาส GameAI: เป็นคลาสที่ใช้ควบคุมการทำงานของเอไอโมดูลที่ผู้ใช้เทสเบดต้องการจะทดสอบในกรณีที่จะใช้โค้ดภาษาเดียวกับที่ใช้สร้างเทสเบด (ซี/ซีพลัสพลัส) มีเมท็อดสำคัญคือ `RunAI()` ใช้สั่งรันเอไอที่ต้องการทดสอบ เมท็อดนี้จะถูกเรียกทำงานทุกรอบการทำงานของอีมูเลเตอร์ ผู้ใช้งานไอเทมจะต้องพัฒนาเอไอไม่ว่าจากซอร์สโค้ดที่ใดหรือคลาสใดก็ตาม แล้วจึงนำมาเรียกใช้ภายในเมท็อดนี้ เอไอนั้นก็จะถูกเรียกทำงานทุกรอบด้วยเช่นกัน เมท็อด `RunFrameRoutine()` ใช้สั่งให้ตัวแปรสมาชิก `m_gameChar` ซึ่งเป็นตัวแทนของตัวละครในเกมทำการเรียกเมท็อด `UpdateGameStateData()` เพื่ออัพเดทข้อมูลสถานการณ์ของเกมดังที่ได้อธิบายไปในคลาสที่แล้ว ส่วนเมท็อด `CheckKeyBuffer()` และ `PressCommand()` นั้นใช้ในการตรวจสอบและสั่งกดสัญญาณคอนโทรลเลอร์ของ `m_gameChar` และส่งให้คลาส DInput ต่อไป

คลาส GamePython: ถูกออกแบบมาให้เป็นคลาสที่ทำหน้าที่เชื่อมต่อการทำงานระหว่างเทสเบดและไพธอนสคริปต์ หลักการทำงานสำคัญของคลาสนี้คือการประกาศอินเทอร์เฟสฟังก์ชันจากทางฝั่งเทสเบดให้สคริปต์เรียกใช้ได้ และเรียกฟังก์ชันหลักของสคริปต์ให้ทำงานในทุกรอบการทำงานของอีมูเลเตอร์ โดยจะมีเมท็อด `RunPythonAI()`ให้ใช้ในลักษณะเดียวกับ `RunAI()`ของคลาส GameAI แต่ว่า `RunPythonAI()` จะไปเรียกการทำงานของไพธอนสคริปต์ที่มีชื่อไฟล์ตามที่ระบุไว้แทน ดังนั้นจึงต้องมีเมท็อดสำคัญเพิ่มขึ้นอีกสองเมท็อดคือ `ToPython()` และ `GetData()` ซึ่งทั้งสองใช้เพื่อส่งและรับข้อมูลจากไพธอนสคริปต์กับเทสเบด ด้วยหลักการทำงานเช่นนี้จะทำให้เทสเบดสามารถเรียกใช้งานเอไอที่เขียนบนไพธอนสคริปต์ไฟล์ได้ รายละเอียดเพิ่มเติมเกี่ยวกับไพธอนมีแสดงในภาคผนวก



รูปที่ 20 แสดงขั้นตอนการทำงานโดยรวมของคลาสในไอเทม

รูปที่ 20 แสดงการทำงานโดยรวมของเมท็อดของคลาสต่างๆ ที่ได้กล่าวถึง โดยแบ่งออกเป็นส่วนการเริ่มต้นการทำงานทางด้านซ้าย และส่วนลูปการรันทางด้านขวา ในส่วนเริ่มการทำงานนั้นเริ่มจากให้คลาส VBA ทำการโหลดรอมและจำลองเกม จากนั้นเรียกใช้เมท็อด `SetGameStateAddr()`

ของคลาส GameChar เพื่อตั้งค่าแอดเดรสของข้อมูลที่สนใจจะอ่าน และจึงเริ่มลูปการทำงานของเอไอ ในทุกรอบการทำงานเมท็อด `RunFrameRoutine()` ของคลาส GameAI จะถูกเรียกเป็นเมท็อดแรก โดยที่ภายในเมท็อดนี้จะทำการเรียกให้เมท็อด `UpdateGameStateData()` ของคลาส GameChar ทำงาน เพื่ออัพเดทข้อมูลสถานการณ์ในเกมจากนั้นจึงตรวจคำสั่งกดสัญญาณว่ามีเหลือในอาเรย์ `m_keyBuffer` ค้างไว้หรือไม่ด้วยเมท็อด `CheckKeyBuffer()` หากมีก็ทำการส่งค่าให้คลาส DInput นำไปใช้แล้วจบรอบการทำงาน แต่หากไม่มีจึงจะเรียกเมท็อด `RunAI()` เพื่อประมวลผลเอไอและตัดสินใจส่งคำสั่งออกท่าลงไปเก็บในอาเรย์ `m_keyBuffer` โดยใช้เมท็อด `PressButton()` แล้วจึงส่งค่าให้คลาส DInput นำไปใช้แทนสัญญาณจากจากอุปกรณ์ควบคุมเดิมต่อไป

5.1.4 ทดสอบการทำงานเบื้องต้น

หลังจากดัดแปลงอีมูเลเตอร์เป็นเทสเบดตามขั้นตอนข้างต้นแล้ว จึงได้นำเทสเบดไปใช้ทดสอบเอไอแบบพื้นฐานโดยการเขียนเงื่อนไข (if then else) เพื่อตรวจเช็คระยะห่างและท่าทางของตัวละครทั้งสองและเลือกทำการออกท่าไปตามความเหมาะสม ผลที่ได้คือเทสเบดสามารถรับรู้สถานการณ์ในเกมได้ถูกต้อง เอไอพื้นฐานสามารถนำเอาสถานการณ์ที่รับรู้นั้นมาเลือกกระทำการได้ถูกต้องและตัวละครในเกมออกท่าตามคำสั่งกดตรงกัน เบื้องต้นนี้จึงสรุปว่าเทสเบดสามารถใช้ทดสอบเกมการต่อสู้ได้เพียงพอ

5.2 การทดลองสร้างโกสเอไอ

ได้ทำการสร้างโกสเอไอตามแนวทางที่กล่าวไปในบทที่สี่ โดยเริ่มจากการบันทึกข้อมูลการเล่นของผู้เล่นทดสอบเอาไว้หนึ่งตัวอย่างก่อนเพื่อไว้ใช้ทดลอง การเล่นตัวอย่างนี้มีความยาว 2 นาที 20 วินาที เป็นการเล่นตามปรกติที่มีรูปแบบชัดเจนและไม่ได้มีเจตนาจะเน้นให้โกสเอไอลอกเลียนได้ง่ายหรือยากแต่อย่างไร

5.2.1 การสร้างฐานข้อมูลจัดกลุ่มอนิเมชั่นและอัตราการบาดเจ็บ

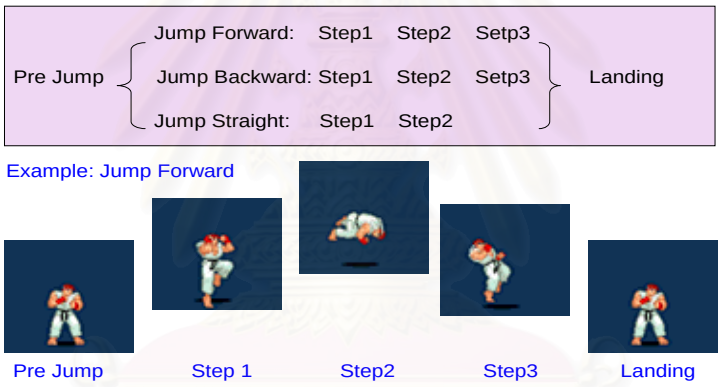
การสร้างเคสของโกสเอไอมีหลักการจากการดึงเอาสถานการณ์ที่มีการเปลี่ยนแปลงอนิเมชั่นออกมา ดังนั้นจึงต้องมีฐานข้อมูลดังกล่าวเพื่อใช้บอกว่าท่าแต่ละเฟรมอยู่ในกลุ่มอนิเมชั่นไหน ตัวอย่างการจัดกลุ่มอนิเมชั่นแสดงในรูปที่ 13 ข้างต้น และข้อมูลการจัดกลุ่มทั้งหมดแสดงในภาคผนวก ในส่วนนี้จะกล่าวถึงวิธีการจัดกลุ่ม

เบื้องต้นได้ทำการจัดกลุ่มโดยวิธีการมองจากภาพอนิเมชั่นเฟรมของตัวละครด้วยตาว่าภาพนี้คือท่าอะไร แล้วกำหนดหมายเลขกลุ่มให้ท่าเฟรมเหล่านั้น แต่มีบางเฟรมที่ดูไม่ออกว่าเป็นส่วนหนึ่งของท่าใดกันแน่ ซึ่งส่วนมากจะเป็นเฟรมที่มีลักษณะยืนเฉย เพื่อเตรียมจะทำท่าต่อไปหรือยืนเฉยหลังจากจบการทำท่าอื่น ขั้นต่อมาจะตรวจสอบผลอีกครั้งโดยเทียบผลการแบ่งด้วยตากับผลของการส่งคำสั่งกดท่าทางต่างๆ ทีละท่า กล่าวคือค่อยๆ สั่งกดท่าทีละท่าแล้วดูว่าภาพที่ปรากฏออกมานั้นคือเฟรมไหน

บ้างและเฟรมเหล่านั้นถูกจัดกลุ่มอยู่ในกลุ่มท่าที่กำหนดด้วยตาก่อนหน้าจริงหรือไม่ ผลออกมาจากวิธีการทั้งสองนี้ทำให้ได้ผลที่น่าเชื่อถือมากว่าทุกเฟรมถูกจัดกลุ่มอยู่ในกลุ่มที่เหมาะสมแล้ว

มีการจัดกลุ่มท่าที่ต้องทำด้วยวิธีพิเศษบางกรณีเช่น ท่ากระโดด ที่มีการแยกการกระโดดออกเป็นกลุ่มย่อยหลายกลุ่มตั้งแต่กลุ่มท่าเริ่มกระโดด กระโดดช่วงที่1-3 และท่าลงสู่พื้นตามจังหวะความสูงของการกระโดดดังแสดงในรูปที่ 21 เพื่อให้ท่าที่ออกหลังจากกระโดดทำได้ถูกจังหวะ เนื่องจากท่าโจมตีกลางอากาศของตัวละครจะให้ผลต่างกัน เมื่อออกท่าหลังจากการกระโดดเป็นเวลาต่างกัน เช่นหากออกท่าเตะทันทีที่เริ่มกระโดดจะส่งผลให้โจมตีศัตรูได้เร็วแต่ไม่สามารถทำคอมโบต่อเนื่องได้ แต่หากออกท่าเตะในช่วงใกล้จะลงสู่พื้นแล้วถึงจะสามารถทำคอมโบต่อได้แต่ก็เสี่ยงต่อการถูกโจมตีระหว่างลอยอยู่กลางอากาศ นอกจากนี้ยังมีการจัดกลุ่มแบบพิเศษอีกบางกรณีซึ่งจะกล่าวถึงต่อไป

ด้วยการจัดกลุ่มท่าอย่างเหมาะสมและวิธีการกดคอนโทรลเลอร์ที่เหมาะสมรวมกันทำให้สามารถสร้างโกสเอไอโดยใช้เพียงวิธีการสร้างเคสจากสถานการณ์ที่เปลี่ยนแปลงกลุ่มท่าของบันทึกการต่อสู้



รูปที่ 21 การแบ่งกลุ่มท่าการกระโดดออกเป็นหลายกลุ่ม

ฐานข้อมูลแสดงความเสียหายของการโจมตีแต่ละท่านั้น จำเป็นต้องกำหนดขึ้นมาเอง ไม่สามารถใช้ค่าการบาดเจ็บที่อ่านจากเกมโดยตรงได้ เพราะว่ามีปัจจัยในเกมหลายอย่างที่ทำให้ค่าการบาดเจ็บนั้นไม่คงที่แม้จะเกิดจากท่าเดียวกันก็ตาม เช่นจำนวนครั้งที่โดนโจมตีแบบต่อเนื่อง การกดปุ่มรัวเพื่อลดการบาดเจ็บหรือการโดนโจมตีแบบสวนเป็นต้น จึงต้องกำหนดค่าคงที่ขึ้นมาค่าหนึ่งสำหรับท่าโจมตีแต่ละท่า รายละเอียดของข้อมูลนี้แสดงในภาคผนวก

5.2.2 การบันทึกข้อมูลการต่อสู้ของผู้เล่น

ความสามารถในการบันทึกข้อมูลการต่อสู้นั้นไม่มีในโครงสร้างเดิมของไอเทม จึงต้องเพิ่มความสามารถนี้ลงไปด้วย ซึ่งทำโดยการดัดแปลงฟังก์ชันบันทึกการเล่นของอีมูเลเตอร์ แต่เปลี่ยนจากบันทึกสัญญาณคอนโทรลเลอร์เป็นบันทึกข้อมูลที่ใช้ในการนิยามและแบ่งแยกสถานการณ์แทน และตั้งชื่อสกุลของไฟล์บันทึกการต่อสู้นี้ว่า วีเอ็มวีจีไฟล์ เพิ่มตัวจีที่หมายถึงโกสเอไอเข้าไปจากแต่เดิมคือวีเอ็มวี ในเวลาที่บันทึกข้อมูลการต่อสู้ของผู้เล่นก็ได้บันทึกการเล่นนั้นเอาไว้ในรูปแบบวีเอ็มวีไฟล์ด้วยเพื่อไว้

ใช้อ้างอิงและวิเคราะห์ผลความเหมือน อนึ่งข้อมูลการบาดเจ็บของตัวละครนั้นใช้ค่าจากฐานข้อมูลที่กำหนดเองในหัวข้อ 5.2.1 ข้างต้น โดยเลือกค่ามาให้ถูกกับท่าโจมตีในแต่ละช่วงเวลา

5.2.3 การสร้างโกสเอไอ

โปรแกรมสำหรับสร้างโกสเอไอถูกสร้างขึ้นมาเพื่อทำการปรับข้อมูลจากบันทึกการต่อสู้ และดึงเอาเคสออกมาสร้างเป็นโกสเอไอดังที่เคยกล่าวไปในบทที่แล้ว ในส่วนนี้จะอธิบายรายละเอียดของการทดลองจากขั้นตอนเหล่านั้น

5.2.3.1 การหาท่าที่เกิดและจบลงในเวลาอันสั้นเกินไป

กำหนดให้ท่าอนิเมชั่นทุกกลุ่มมีช่วงเวลาที่จะใช้ตัดสินว่าเป็นท่าที่เกิดสั้นคือ 6 เฟรม ยกเว้นท่านั่งเท่านั้นที่ถูกกำหนดให้เป็น 14 เฟรมเพราะแค่ช่วงการเปลี่ยนท่าจากยืนเป็นนั่งก็ใช้เวลาไป 8 เฟรมแล้วตัวละครจึงจะเริ่มแสดงท่านั่งออกมา ถ้าหากว่าไม่กำหนดช่วงเวลาของท่าสั้นไว้เช่นนี้จะพบว่าเคสที่สร้างออกมาจะมีการสั่งให้ทำท่านั่งโดยไม่สั่งท่าอื่นที่แท้จริงแล้วต้องเกิดตามมาบ่อยมากเกินไป ตัวเลขนี้ได้มาจากการทดลองหาค่าที่เหมาะสมว่าควรตัดท่าสั้นที่กี่เฟรมจึงจะเหมาะ จากการทดลองพบว่าไม่มีค่าของจำนวนเฟรมสั้นที่ใช้ได้ผลกับทุกท่า จึงใช้วิธีกำหนดค่าสำหรับแต่ละท่าให้ไม่เท่ากัน

5.2.3.2 การแก้ไขท่าให้ถูกต้องไม่กำกวม

ในการทดลองนี้มีหลายข้อมูลจากบันทึกการต่อสู้ที่ต้องแก้ไข ดังเช่นกรณีของการแก้ไขท่าเริ่มต้นของการกระโดดแบบต่างๆ ซึ่งได้อธิบายไปแล้วในการยกตัวอย่างในหัวข้อ 4.4.2 นอกจากนั้นยังมีกรณีอื่นดังนี้

กรณีแรกคือท่าเตะพายุหมุนที่แสดงออกมานั้นไม่สามารถแยกได้ว่าเป็นท่าแบบเบา กลางหรือว่าหนัก เฉพาะท่าเตะพายุหมุนเท่านั้นที่ไม่สามารถรู้ได้ ท่าพิเศษอื่นเช่นฮาโดเคน (ท่าปล่อยพลังออกจากฝ่ามือ) หรือโชริวเคน (ท่าชกลอยตัวกลางอากาศ) นั้นสามารถรู้ได้จากค่าเฟรมที่แสดงออกมา เพราะว่าแม้จะเห็นเป็นภาพแบบเดียวกันเวลาทำท่าแบบเบา กลาง หนัก แต่หมายเลขเฟรมนั้นใช้คนละหมายเลขกัน แต่สำหรับท่าเตะพายุหมุนนั้นมีการใช้เฟรมเดียวกันสลับไปมาหลายรอบร่วมกันในท่าแบบเบา กลางและหนัก ดังนั้นจะต้องพิจารณาทั้งช่วงอนิเมชั่นของท่านี้ว่าเป็นแบบใด วิธีการคือตอนแรกนั้นจะกำหนดให้ทุกเฟรมของท่าเตะพายุหมุนถูกจัดเป็นท่าเตะพายุหมุนปริศนาก่อน จากนั้นเมื่อผ่านการตรวจเงื่อนไขดูว่ามีเฟรมใดและเกิดนานเท่าไรบ้างจึงสามารถบอกได้ว่าเป็นแบบเบา กลางหรือหนัก แล้วจึงย้อนกลับไปแก้ไขข้อมูลทุกๆ เฟรมในช่วงการทำท่านี้ว่าเป็นแบบ เบา กลางหรือหนัก

กรณีที่สองคือท่ากลับตัวกลางอากาศ การกลับตัวกลางอากาศในเกมนี้ทำได้สามแบบคือกลับตัวกลางอากาศแล้วจุดตกลงสู่พื้นเป็นจุดเดิม ใกล้กว่าเดิมและไกลกว่าเดิม นอกจากนี้ยังมีการกลับตัวที่เกิดขึ้นโดยอัตโนมัติด้วยหากว่าการบาดเจ็บกลางอากาศนั้นไม่รุนแรงมาก รวมมีการกลับตัวทั้งหมด 4 แบบ ซึ่งทุกแบบใช้เฟรมอนิเมชั่นชุดเดียวกันหมด แต่ว่าการกลับตัวแบบที่ต้องสั่งกดเองตัวละครจะมีการกระพริบสีขาวชั่วพริบตาให้เห็นว่าเป็นการสั่งกดไม่ได้เกิดอย่างอัตโนมัติ ซึ่งพบว่าการกลับตัวที่เกิด

อัตโนมัติจะเกิดทันทีที่ถูกโจมตีโดยไม่มีการเปลี่ยนเข้าสู่ท่าบาดเจ็บก่อน ดังนั้นจึงสามารถแยกแยะได้ว่าการกลับตัวที่เกิดในตอนที่อยู่ในท่าบาดเจ็บคือการกลับตัวที่ผู้เล่นสั่งกระทำเอง จึงต้องเพิ่มเป็นเคสในโกสเอไอด้วย ประเด็นต่อมาก็คือการตัดสินว่าเป็นการกลับตัวแบบจุดตกแบบไหน ซึ่งไม่สามารถแยกแยะได้ด้วยการดูท่าเฟรมเพียงอย่างเดียว จะต้องดูตำแหน่งของตัวละครในเฟรมถัดๆ ไปประกอบด้วย จนในที่สุดก็สามารถที่จะรู้ได้ว่าเป็นการกลับตัวแบบใด

กรณีที่สามคือท่าที่สามารถทำต่อเนื่องกับตัวเองได้ เนื่องจากการสร้างเคสจะเกิดเมื่อเกิดการเปลี่ยนแปลงกลุ่มท่า แต่มีท่าบางท่าที่ทำต่อเนื่องกันได้หลายครั้งด้วยตัวเองโดยไม่มีการเปลี่ยนกลุ่มท่า เช่นท่าต่อยเบา เตะเบาทั้งแบบยืนและแบบนั่งเป็นต้น ท่าเหล่านี้มีคุณสมบัติพิเศษที่ทำต่อเนื่องกับตัวเองได้ นั่นคือผู้เล่นเพียงกดต่อยเบารัวๆ ก็จะเกิดการทำคอมโบต่อยเบา 3-4 ครั้งได้โดยง่าย โดยที่ระหว่างนั้นตัวละครไม่หลุดออกจากกลุ่มท่าต่อยเบาเลย วิธีการแก้ทำได้โดยการแบ่งกลุ่มท่าต่อยเบาออกเป็นกลุ่มย่อย แบ่งเป็นช่วงออกหมัดและเก็บหมัด ดังนั้นจะทำให้เกิดการเปลี่ยนกลุ่มอนิเมชั่นได้

กรณีที่สี่คือท่าเดิน การเดินเป็นอีกหนึ่งท่าที่ทำต่อเนื่องได้ไม่มีการเปลี่ยนแปลงกลุ่มอนิเมชั่น การเดินจากขอบซ้ายมาถึงขอบขวาหรือแค่เดินแค่ 1 ก้าว จะถูกนับ 1 กลุ่มท่าเหมือนกัน หากเป็นเช่นนี้จะทำให้การเดินของโกสเอไอไม่สมจริง จึงทำการกำหนดกลุ่มอนิเมชั่นการเดินออกเป็นหลายกลุ่มหลายระยะทาง เช่น เดิน 10 หน่วย เดิน 20 หน่วย เป็นต้น ซึ่งระยะห่างการเดินท่าละ 10 หน่วยนี้มีความละเอียดและทำให้ไม่เปลืองจำนวนกลุ่มท่ามากเกินไปด้วย นั่นคือไม่ต้องกำหนดท่าเดินแบบ 10, 11, 12, ฯลฯ หน่วยซึ่งละเอียดเกินความจำเป็น จากนั้นก็ทำการแก้ไขโดยการให้ตอนแรกสุดให้การเดินทุกครั้งที่เจอเป็นการเดินแบบปริศนาก่อน แล้วจึงตรวจอีกทีว่าการเดินครั้งนั้นได้ระยะทางเท่าไรแล้วจึงใส่หมายเลขกลุ่มเข้าไปให้ใหม่

กรณีที่ห้าคือท่าที่ให้เลือกทำหรือไม่ทำเท่านั้น เช่นท่ากลับตัวกลางอากาศ ในตอนที่ตัวละครบาดเจ็บอยู่กลางอากาศ ผู้เล่นจะเลือกได้แค่ว่าจะกลับตัวกลางอากาศหรือไม่ ถ้าไม่กลับตัวตัวละครจะยังอยู่ในท่าบาดเจ็บกลางอากาศต่อไป การเลือกไม่กลับตัวทำให้กลุ่มอนิเมชั่นไม่เปลี่ยนแปลง ดังนั้นหากเลือกกลับตัวเพียงครั้งเดียวก็จะถูกบันทึกเข้าไปในเคสครั้งเดียวแต่ทำตลอดทุกครั้งเพราะว่าไม่มีการไม่ทำหรือการทำท่าอื่นมาช่วยดึงอัตราส่วน กรณีแบบนี้จะแก้ไขโดยการแบ่งการไม่ทำอะไรในระหว่างบาดเจ็บกลางอากาศเป็นสองกลุ่มเช่นเดิม ด้วยวิธีนี้การไม่กลับตัวจะถูกบันทึกลงไปเป็นกรณีพิเศษเพื่อถ่วงอัตราส่วนการกลับตัวกลางอากาศให้สมจริง

5.2.3.3 การทำเครื่องหมายท่าที่ไม่ต้องการ

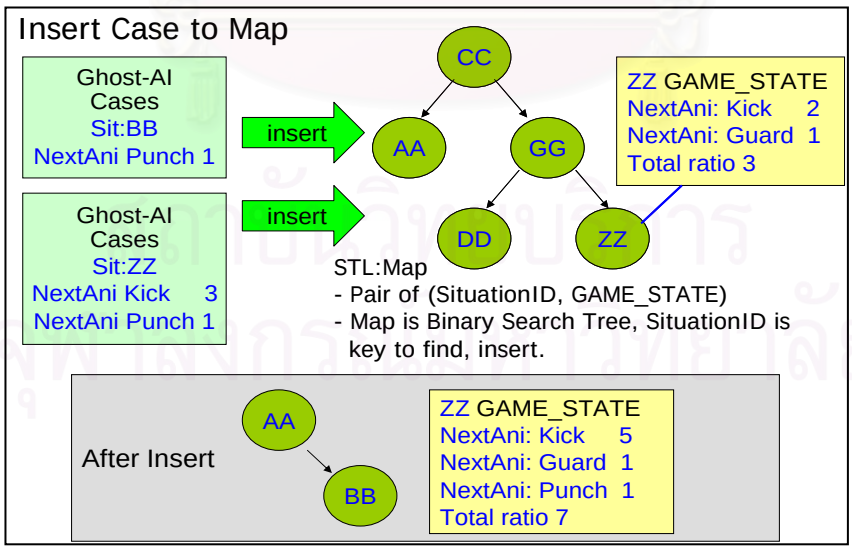
นอกจากท่าบาดเจ็บแล้วท่ายืนเฉยสมควรถูกตัดออกเช่นเดียวกัน การเปลี่ยนอนิเมชั่นจากท่าใดๆ มาเป็นท่ายืนเฉยนั้นไม่จำเป็น เนื่องจากว่าเวลาที่รันโกสเอไอ หากว่าหาเคสไม่เจอตามขั้นตอนก็จะไม่สั่งทำอะไรและก็จะได้ผลออกมาเป็นท่ายืนเฉยอยู่แล้ว

5.2.3.4 การพิจารณาบันทึกการต่อสู้ที่แก้ไขข้อมูลแล้วเพื่อดึงเอาเคสออกมา

การดึงเคสออกมานั้นทำโดยไล่ดูข้อมูลทีละเฟรมเพื่อหาว่าเฟรมไหนที่มีการเปลี่ยนแปลงกลุ่มท่าอนิเมชั่น จากนั้นจึงหยิบเอาข้อมูลสถานการณ์ในเฟรมนั้นมาเข้ารหัสตามวิธีการในหัวข้อ 4.6 หลังจากนั้นจึงดึงเอาอนิเมชั่นท่าที่เฟรมนั้นเปลี่ยนไปเป็น (อนิเมชั่นของเฟรมถัดจากเฟรมนั้น เฟรมแรกที่ไม่ถูกทำเครื่องหมายเป็นท่าที่ไม่ต้องการ) ออกมาสร้างเป็นเคสตามขั้นตอนในหัวข้อ 4.5 จากการทดลองจริงพบว่าต้องมีการแก้ไขพิเศษสำหรับบางกรณี เช่นกรณีของการกลับตัวกลางอากาศที่มีช่วงเวลาเข้ามาบังคับว่าหากไม่สั่งกดภายในเวลาที่กำหนดจะไม่ทันแล้วการกลับตัวจะไม่สำเร็จ ซึ่งหากผู้เล่นเองในตอนที่เล่นเพื่อบันทึกโกสเอไอได้สั่งกดในเสี้ยววินาทีสุดท้ายพอดี แล้วโกสเอไอหยิบเอาสถานการณ์นั้นไปเริ่มสั่งกดก็จะทำให้ออกท่าไม่ทันได้ ในกรณีเช่นนี้จึงต้องย้อนไปเอาสถานการณ์ก่อนหน้าที่จะเปลี่ยนกลุ่มท่ามาใช้

5.2.3.5 โครงสร้างข้อมูลที่ใช้ในการเก็บเคส

ได้เลือกใช้ แม็ป (map) ของ สแตนดาร์ดเทมเพลทไลบรารี่ (Standard Template Library-STL) มาใช้ในการเก็บเคส เนื่องจากมีหลักการเก็บข้อมูลในรูปของต้นไม้ค้นแบบทวิภาคชนิดแดงดำ สามารถค้นหาและแทรกข้อมูลได้ในเวลา O(log n) ในการทำการเพิ่มเคสเข้าไปนั้นต้องจัดคู่หมายเลขสถานการณ์และการกระทำเป็นแพร์ (pair) ก่อนแล้วจึงใส่เข้าไปในแม็ป โดยที่แพร์นี้ก็คือเคสนั่นเองและหมายเลขของสถานการณ์ที่ได้จากการเข้ารหัสนั้นจะใช้เป็นคีย์การใช้ค้นเอาเคส ออกมา ตัวอย่างการเพิ่มข้อมูลเข้าไปในแม็ปจะแสดงออกมาดังรูปที่ 22 ซึ่งจากภาพได้มีการเพิ่มเคสที่มีสถานการณ์เป็น BB และ ZZ เข้าไปในโครงสร้างข้อมูล กรณี ZZ นั้นมีอยู่แล้วจึงทำการปรับปรุงข้อมูลภายในเคสนั้น ส่วน BB ไม่เคยมีมาก่อนจึงเพิ่มเข้าไปเป็นกรณีใหม่



รูปที่ 22 แสดงตัวอย่างการเพิ่มเคสเข้าไปภายในแม็ป

5.2.3.6 การบันทึกข้อมูลเคสลงในโกสเอไอไฟล์

หลังจากขั้นตอนการสร้างโกสเอไอดำเนินการครบเรียบร้อยแล้ว ขั้นสุดท้ายคือการบันทึกโกสเอไอจากโครงสร้างข้อมูลลงในไฟล์โดยมีรูปแบบไฟล์ดังแสดงในรูปที่ 23 และแสดงตัวอย่างไฟล์จริงในรูปที่ 24

Ghost AI file format

NUM	Case block	Case block	…	Case block

NUM: Number of cases in this file. (int 4 bytes)
Case block: Each case base detail. (size=12+ 8*num of action in this case)

Case block

ID	NUM	T.Ratio	Action block	…	Action block

ID: Situation ID. (int 4 bytes)
NUM: Number of action in this case. (int 4 bytes)
T.Ratio: Total ratio in this case. (int 4 bytes)
Action block: Each action detail. (size = 8 bytes)

Action block

Action ID	Ratio

Action ID: Animation name. (int 4 bytes)
Ratio: Ratio of this animation. (int 4 bytes)

รูปที่ 23 ไฟล์ฟอร์แมทการบันทึกโกสเอไอ

```
Total Case:0162

ID:0000000001 CurAni:0001 P2Ani:00 Combo:000 deltaX:00 TotalRatio:0002  N:0002
    NextAni:0062  Ratio:0001
    NextAni:0144  Ratio:0001

ID:0000000002 CurAni:0002 P2Ani:00 Combo:000 deltaX:00 TotalRatio:0002  N:0001
    NextAni:0034  Ratio:0002

ID:0000000009 CurAni:0009 P2Ani:00 Combo:000 deltaX:00 TotalRatio:0001  N:0001
    NextAni:0062  Ratio:0001
```

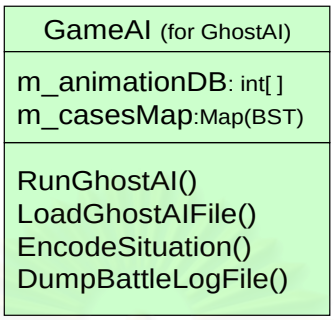
รูปที่ 24 ตัวอย่างไฟล์โกสเอไอ

5.3 การทดลองรันโกสเอไอ

5.3.1 การดัดแปลงไอเทมให้ใช้รันโกสเอไอได้

คลาส GameAI เดิมของเทสเบดได้ถูกปรับเปลี่ยนให้สามารถรันโกสเอไอได้ โดยการเพิ่มตัวแปรสมาชิกและเมท็อดใหม่ในรูปที่ 25 ลงไปในคลาส GameAI เดิม ตัวแปรสมาชิก `m_animationDB` เป็นอาเรย์ที่ใช้เป็นฐานข้อมูลเพื่อระบุว่าท่าของตัวละครเฟรมใดๆ ถูกจัดเป็นท่าอนิเมชั่นกลุ่มใดและให้ค่าความบาดเจ็บเท่าไร ตัวแปรสมาชิก `m_caseMap` คือโครงสร้างข้อมูลที่เอาไว้เก็บเคสหลังจากโหลดโกสเอไอไฟล์เข้ามาและใช้ในการค้นหาเคสเวลาที่โกสเอไอทำงาน เมท็อด `LoadGhostAI()` ใช้โหลดข้อมูลเคสจากโกสเอไอไฟล์เข้าไปเก็บไว้ในตัวแปร `m_caseMap` เมท็อด `RunGhostAI()` และ `EncodeSituation()` นั้นใช้ในการรันโกสเอไอดังที่ได้กล่าวรายละเอียดไปในหัวข้อ 4.8 ส่วน เมท็อด

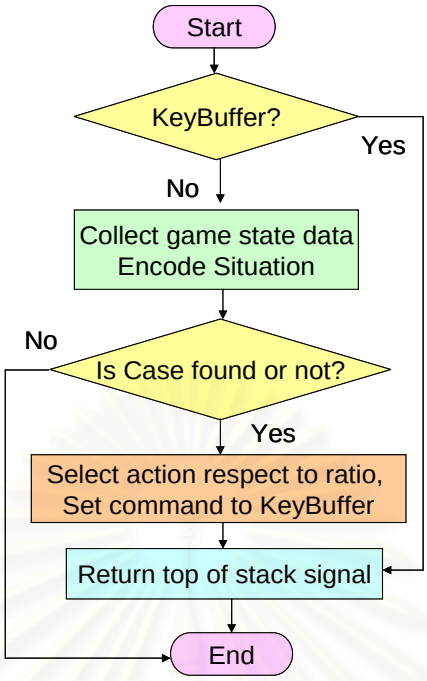
`DumpBattleLogFile()` ใช้เพื่อสร้างบันทึกการต่อสู้ที่ใช้เป็นอินพุตของโปรแกรมสำหรับสร้างโกสเอไอที่กล่าวถึงในส่วนแรกของบทนี้ เมท็อดนี้ไม่เกี่ยวข้องกับการรันโกสเอไอ แต่เพื่อความสะดวกในการใช้งานจึงจัดให้เป็นสมาชิกของคลาสนี้เช่นกัน



รูปที่ 25 คลาส GameAI แบบใหม่ที่ปรับแต่งสำหรับใช้รันโกสเอไอ

5.3.2 การรันโกสเอไอในไอเทม

วิธีการรันโกสเอไอเริ่มโดยการโหลดไฟล์โกสเอไอกลับเข้ามาเก็บในแม็ปเหมือนกับตอนที่สร้าง เนื่องจากต้องการความสามารถในการค้นหาเคสที่รวดเร็ว จากนั้นจึงโหลดฐานข้อมูลการจัดกลุ่ม อนิเมชั่น ฐานข้อมูลค่าการบาดเจ็บและฐานข้อมูลวิธีการสั่งกดท่าเข้าไปในเทสเบด จากนั้นหลักการทำงานจะเป็นดังที่กล่าวไว้ในหัวข้อ 4.8 คือในทุกรอบการทำงานโมดูลรันโกสเอไอจะทำการรวบรวมข้อมูลสถานะของเกมแล้วนำมาเข้ารหัสด้วยวิธีการเดียวกับที่ใช้ในการสร้างโกสเอไอ ได้ผลเป็นข้อมูลรูปอินทีเจอร์ 32 บิต ซึ่งจะถูกใช้เป็นคีย์ในการค้นข้อมูลในแม็ปว่าสถานการณ์นั้นเคยมีเป็นเคสเก็บไว้หรือไม่ ถ้าหากว่าไม่มีก็ไม่ต้องทำอะไรรอรอบต่อไปค่อยพิจารณาใหม่ แต่ถ้าหากเจอก็ให้เลือกการกระทำของเคสนั้นจากรายการกระทำทั้งหมดออกมาโดยวิธีการสุ่มแต่อิงการออกท่าตามสัดส่วนของท่าที่ออกตอนที่ผู้เล่นทำการต่อสู้ ตัวอย่างเช่นหากในเคสนั้นมีท่าอยู่สามท่าคือ ต่อยเบาในอัตราส่วนหนึ่ง เตะเบาในอัตราส่วนสองและต่อยกลางในอัตราส่วนสอง รวมสามท่าเคยเจอมาทั้งหมดห้าครั้ง ก็จะทำการสุ่มตัวเลขที่มีค่าระหว่าง 1 ถึง 100 มา แล้วดูว่าตกลงบนส่วนไหนเมื่อแบ่งเลข 100 ออกเป็น 5 ส่วน โดยจัดส่วนให้ท่าทั้งสามตามอัตราส่วนของมัน เช่นเลข 1 ถึง 20 เป็นของท่าต่อยเบา 21 ถึง 60 คือเตะเบาและ 61 ถึง 100 คือต่อยกลาง ดังนั้นหากค่าสุ่มได้เบอร์ 77 ก็ให้ออกท่าเตะเบาเป็นต้น เมื่อเลือกท่าได้แล้วก็จะไปสั่งกดปุ่มเพื่อออกท่าต่างๆ การสั่งกดปุ่มใช้วิธีใส่ค่าสัญญาณล่วงหน้าหลายเฟรมลงบนอาร์เรย์ `m_keyBuffer` ดังที่ได้อธิบายไปในหัวข้อ 5.1.3 และระหว่างที่อาร์เรย์นั้นมีค่าก็จะไม่ทำการตรวจสอบสถานการณ์ เข้ารหัสหรือหาเคสใดๆ แต่จะส่งค่าสัญญาณคอนโทรลเลอร์ในรอบนั้นออกไปเลย ขั้นตอนการทำงานเหล่านี้แสดงในแผนภูมิสายงานในรูปที่ 26 ในส่วนนี้ไม่พบปัญหาใดในการทดลองจริงนอกจากปัญหาการสั่งกดท่าดังจะกล่าวในส่วนต่อไป



รูปที่ 26 แผนภูมิสายงานแสดงการทำงานของโกสเอไอในทุกรอบการรัน

5.3.3 การสั่งกดท่าตอนรันของโกสเอไอ

การสั่งกดปุ่มออกท่าในการทดลอง จะต้องเรียกฟังก์ชันที่ใช้กดท่าที่ได้ออกแบบไว้อย่างเหมาะสม เนื่องจากเมื่อใช้อีมูเลเตอร์รันเกมจะสามารถส่งคำสั่งกดท่าที่ยาว ได้ในเวลารวดเร็วกว่าคนจริงมาก เช่น ท่าที่ต้องกด ลง เฉียง เดินหน้า และต่อยนั้น คนธรรมดาจะต้องใช้เวลาอย่างเร็วประมาณ 15 เฟรมในการกด แต่ว่าอีมูเลเตอร์สามารถกดเสร็จได้ใน 4 เฟรม ซึ่งเร็วกว่ามาก แต่การกดท่าที่เร็วไปนั้นมีข้อเสีย คืออาจจะทำให้ท่ากดไม่ติดได้ หรือออกผิดท่าในบางครั้งเมื่อพิจารณาสัญญาณโดยรวมกับเฟรมอื่น ตัวอย่างที่ชัดเจนคือการสั่งกดฮาโดเคนแล้วได้ท่าโชริวเคนออกมาแทน เพราะว่าทั้งสองท่ามีวิธีการกดที่ใกล้เคียงกัน และตามกลไกของเกมแล้วหากสัญญาณที่ผู้เล่นกดเข้าไปกำกวมก็จะตัดสินให้ออกท่าโชริวเคนก่อน ดังนั้นจึงต้องออกแบบวิธีกดท่าให้ได้อย่างเหมาะสม มีโอกาสติดสูงทั้งในสถานการณ์ธรรมดาและการทำคอมโบที่ต้องการสั่งกดอย่างถูกจังหวะและแม่นยำ

เนื่องจากปัญหาความเร็วของการสั่งกด จึงได้มีการออกแบบวิธีการสั่งกดท่าพิเศษเอาไว้ท่าละสองแบบ แบบแรกจะกดโดยให้แต่ละสัญญาณยาวใกล้เคียงกับที่ผู้เล่นธรรมดากดโดยเฉลี่ย เพื่อใช้ในกรณีปรกติให้เหมือนผู้เล่นจริงกด ส่วนอีกวิธีหนึ่งจะกดโดยใช้เวลาสั้นเพื่อใช้สำหรับเวลาที่ต้องการทำคอมโบหรือกดต่อเนื่องจากท่าโจมตีอื่นเพื่อต้องการให้กดเสร็จเร็ว มีเวลากดซ้ำอีกเพื่อจะได้ส่งอินพุตเข้าไปตรวจมากๆ เพื่อเพิ่มโอกาสการต่อท่าให้สำเร็จ

การปรับระดับความเร็วของเกมก็มีผลต่อความเร็วในการกดปุ่มออกท่าด้วยเช่นกัน เนื่องจากเกมจะตรวจว่าท่าที่กดนั้นติดหรือไม่โดยดูจากระยะเว้นช่วงของสัญญาณแต่ละสัญญาณว่าห่างกันเกิน

กว่าที่กำหนดหรือไม่ เช่นหากจะกดท่าฮาโดเคน (ซึ่งต้องกด ลง ลงเดินหน้า เดินหน้า ต่อย) โดยกดลงกับเฉียงห่างกันเกินกว่าเวลาที่กำหนดท่านั้นก็จะไม่ติด การปรับความเร็วของเกมให้สูงขึ้นหรือต่ำลงจะทำให้ช่วงเวลาดังกล่าวสั้นลงหรือเพิ่มขึ้นด้วย ช่วงเวลาที่กำหนดให้แต่ละสัญญาณจะต้องถูกกดนั้นไม่คงที่เป็นไปตามอัตราส่วนดังแสดงในตารางที่ 3 [14, 20] ตัวอย่างเช่นมีโอกาส 3.125% ที่เกมจะอนุญาตให้สัญญาณของการกดท่าพิเศษส่งมาห่างกันได้ถึง 15 เฟรม สมมุติว่าผู้เล่นจะกดท่าฮาโดเคนและกดสัญญาณลงกับลงเดินหน้าห่างกัน 17 เฟรม โดยที่ตอนนั้นเกมสุ่มว่าจะยอมให้สัญญาณส่งมาห่างกันได้ 17 เฟรม (โอกาส 3.125%) พอดีก็จะตรวจผ่านและยอมให้ตรวจสัญญาณการออกท่าขั้นต่อไปได้ แต่ถ้าหากตอนนั้นเกมสุ่มออกมาว่าจะยอมให้สัญญาณออกมาห่างกันได้แค่ 10 เฟรม ก็จะตรวจไม่ผ่านและกดท่าฮาโดเคนไม่ติดทันที

ความเร็วของเกมมากเท่าใดการกดท่าก็ต้องเร็วขึ้นเท่านั้นด้วย ในการทดลองนี้ได้ออกแบบวิธีการกดคอนโทรลเลอร์ให้เหมาะสมกับความเร็วเกมระดับเทอร์โบสองซึ่งเป็นความเร็วดั้งเดิมของเกม แต่ผลของโกสเอไอสามารถปรับให้ใช้เข้ากับทุกความเร็วเกม โดยการปรับความเร็วในการกดสัญญาณคอนโทรเลอร์

ในการทดลองนี้ได้ใช้โหมดการกดคำสั่งแบบสั้นซึ่งท่าพิเศษที่ต้องใช้เกจท่าไม้ตายในการออกท่าจะใช้วิธีกดต่อยและเตะพร้อมกันแทนการกดปุ่มทิศทางสองรอบ ซึ่งแม้ในโหมดนี้จะทำให้ไม่สามารถสั่งกดท่าซุปเปอร์คอมโบแบบเลือกจำนวนเกจที่จะใช้ได้ แต่เพียงพอต่อการกดคอมโบในการทดลองและทำเพื่อความสะดวกของผู้ทดสอบ

ตารางที่ 3 แสดงช่วงเวลาที่เกมยอมให้สัญญาณการกดท่าพิเศษส่งมาห่างกันนานแค่ไหน ในกรณีที่ความเร็วของเกมอยู่ที่ระดับปรกติ (Normal)

Time	Ratio		Time	Ratio
10/60	50.0%		14/60	6.25%
11/60	12.5%		15/60	3.125%
12/60	12.5%		16/60	3.125%
13/60	9.375%		17/60	3.125%

อีกประเด็นหนึ่งที่เป็นปัญหาในการสั่งกดท่าของการทดสอบโกสเอไอคือ จากกลไกเดิมของเกมตัวละครจะยังคงสถานะยืนบนพื้นอีก 5-6 เฟรม หลังจากสั่งกระโดดแบบใดๆ ไป หากว่าทำการเข้ารหัสถานการณ์ในช่วงเวลานี้ ซึ่งกลไกการเข้ารหัสสถานการณ์จะยังเข้าใจว่าตัวละครยังยืนอยู่บนพื้น (เพราะต้องรออีกสัก 5-6 เฟรมตัวละครจึงค่อยเท้าลอยจากพื้นและมีค่าพิกัดแกนตั้งมากกว่าศูนย์) แล้วได้คำตอบที่สั่งให้ทำท่าโจมตีบนพื้นและสั่งทำลงไป ท่าโจมตีบนพื้นนั้นจะไปออกผลตอนที่ตัวละครกระโดดไปแล้ว กลายเป็นท่าโจมตีกลางอากาศไป วิธีการแก้คือต้องทำการตรวจสอบก่อนและระวังไม่สั่งออกท่าโจมตีบนพื้นหลังจากที่เพิ่งสั่งกระโดดไปภายในเวลา 6 เฟรม

การรันดูผลของโกสเอไอขั้นพื้นฐานทำโดยสังเกตรูปแบบการออกท่าในจังหวะต่างๆ ของโกสเอไอและมูวี่ส์บันทึกการเล่นของผู้เล่นตัวอย่างที่ได้บันทึกเอาไว้ ผลการเล่นโดยรวมนั้นโกสเอไอมีการออกท่าตามผู้เล่นเจ้าของในสถานการณ์เดียวกันหลายครั้ง การทดลองขั้นต่อไปจึงทำโดยการวัดเทียบสัญญาณคอนโทรลเลอร์ตามที่กำหนดในข้อเสนอโครงร่างวิทยานิพนธ์

5.4 การทดลองกับผู้เล่นทดสอบและการเทียบสัญญาณคอนโทรลเลอร์

5.4.1 วิธีการทดลองกับผู้ทดสอบ

ได้ทำการทดสอบโดยการให้ผู้เล่นจำนวน 32 คน ซึ่งมาจากหลายกลุ่ม ทั้งช่วงอายุ เพศ และประสบการณ์ในการเล่นเกมการต่อสู้ มาเล่นเกมเป็นเวลาประมาณสองถึงสิบนาทีแล้วแต่ความพอใจของผู้ทดสอบ โดยใช้คอนโทรลเลอร์ของเครื่องเพลย์สเตชั่นในการเล่นทดสอบ เล่นจนผู้ทดสอบพอใจในรูปแบบการเล่นและบันทึกการเล่นเอาไว้ จากนั้นจึงสร้างโกสเอไอของผู้เล่นขึ้นมา โดยขอให้ถือว่าการเล่นเพียงครั้งเดียวของผู้เล่นที่ได้บันทึกเอาไว้นี้เป็นตัวแทนการเล่นของผู้เล่นคนนั้น และเป็นตัวแทนที่จะใช้เปรียบเทียบความเหมือนของโกสเอไอ แม้ว่าในความจริงผู้เล่นคนเดียวกันอาจจะมีรูปแบบการเล่นหลายแบบ และการเล่นแต่ละครั้งอาจจะใช้รูปแบบการเล่นไม่เหมือนกันทำให้ออกมาดูไม่เหมือนกันจนไม่สามารถระบุได้ว่าเป็นคนเดียวกันก็ตาม แต่หากปราศจากข้อกำหนดนี้แล้ว การทดลองเพื่อวัดผลโกสเอไอจะไม่สามารถทำได้ เป้าหมายของการทดลองนี้จึงทำเพื่อเปรียบเทียบความเหมือนระหว่างโกสเอไอที่เกิดจากบันทึกการเล่นของผู้เล่น กับตัวบันทึกการเล่นของผู้เล่นนั้นเป็นครั้งๆ ไป หากโกสเอไอเล่นได้เหมือนกับบันทึกการเล่นที่เป็นต้นฉบับของมัน ก็ควรจะอนุมานได้ว่าโกสเอไอเล่นได้เหมือนกับเจ้าของโกสเอไอในรูปแบบการเล่นขณะที่บันทึกการเล่นครั้งนั้น

ในการทดลองนี้ไม่สามารถนำโกสเอไอกับมูวี่ส์การเล่นของผู้เล่นมาเทียบสัญญาณกันโดยตรงได้ เนื่องจากเวลาที่ทั้งสองระบบเล่นนั้น สถานการณ์ในเกมจะแตกต่างกันออกไปตามการกระทำของตัวละครทั้งสองฝ่ายซึ่งมีปัจจัยการสุ่มอยู่ด้วย ทำให้ตัวละครตอบสนองต่างกันและสั่งกดคอนโทรลเลอร์คนละสัญญาณกัน จึงได้ทำการทดสอบผลโดยให้ผู้เล่นกดคอนโทรลเลอร์เสมือนว่าตัวเองเป็นผู้บังคับตัวละครในขณะที่โกสเอไอของตนเองหรือมูวี่ส์ของตนเองกำลังเล่นอยู่ เพื่อเทียบความเหมือนของสัญญาณระหว่าง โกส-ตนเอง และ มูวี่ส์-ตนเองและวิเคราะห์ผล

การวัดผลสัญญาณนั้นไม่ได้นำเอาสัญญาณคอนโทรลเลอร์ของโกสเอไอหรือมูวี่ส์และผู้เล่นมาเทียบกันเฟรมต่อเฟรมแบบตรงๆ แต่ได้ทำการตัดช่วงและรวบสัญญาณก่อน

5.4.2 การตัดช่วงและรวบสัญญาณ

การตัดช่วงและรวบสัญญาณ คือการตัดแบ่งสัญญาณคอนโทรลเลอร์ออกเป็นช่วงๆ และในแต่ละช่วงที่ตัดออกมานั้นให้ทำการรวบสัญญาณที่เหมือนกันที่อยู่ติดกันเข้าด้วยกัน ตัวอย่างเช่น สมมุติสัญญาณคอนโทรลเลอร์ช่วงหนึ่งที่ถูกตัดออกมาซึ่งมีความยาว 15 เฟรมเป็นดังนี้

16,16,16,32,32,32,64,64,64,64,16,16,256,256,256

สัญญาณในเฟรมที่หนึ่งถึงสามของช่วงสัญญาณนี้มีค่า 16 เหมือนกันหมดและอยู่ติดกัน จึงสามารถถูกรวบเข้าด้วยกันเหลือเป็น 16 ตัวเดียว เมื่อพิจารณาต่อไป เฟรมที่สี่ถึงหกมีค่าสัญญาณเป็น 32 ติดต่อกันจึงสามารถถูกรวบเหลือ 32 ตัวเดียวเช่นกัน ดังนั้นภายหลังจากทำการรวบสัญญาณแล้วจะได้สัญญาณออกมาเป็นดังนี้ 16, 32, 64, 16, 256

การตัดช่วงและรวบสัญญาณทำเพื่อตัดปัจจัยด้านความเร็วในการกดออกไปทำให้ความน่าเชื่อถือของผลการเทียบสัญญาณมีแนวโน้มถูกต้องมากขึ้น เนื่องจากแม้เวลาที่ผู้เล่นตัดสินใจกดคอนโทรลเลอร์เหมือนกันสองครั้งในสถานการณ์เดียวกัน ผู้เล่นไม่สามารถกดให้ตรงกันอย่างสมบูรณ์ได้ทั้งสองครั้ง มักจะมีการกดเร็วหรือช้าเหลื่อมกันบ้าง ซึ่งจะเป็นผลทำให้การนำสัญญาณมาเทียบกันโดยตรงได้ผลออกมาเป็นสัญญาณที่เหลื่อมกันไม่สามารถนำไปใช้เทียบท่าที่กดได้ ตัวอย่างเช่น สมมุติสัญญาณคอนโทรลเลอร์ของโกสเอไอและผู้เล่นแบบที่ยังไม่ได้ตัดช่วงและรวบสัญญาณเป็นดังนี้

สัญญาณของโกสเอไอแบบที่ยังไม่ได้ตัดช่วงและรวบสัญญาณ

16,16,16,32,32,32,64,64,64,64,128,128,256,256,256

สัญญาณของผู้เล่นแบบที่ยังไม่ได้ตัดช่วงและรวบสัญญาณ

16,16,16,16,16,16,32,32,32,32,64,64,128,128,128

หลังจากการรวบสัญญาณจะกลายเป็นดังนี้

สัญญาณของโกสเอไอ 16,32,64,128,256

สัญญาณของผู้เล่น 16,32,64,128,0

จะสังเกตได้ว่าหากเทียบสัญญาณกันตรงๆ แล้วจะได้ความตรงกันเพียง 3 จาก 15 สัญญาณ แม้ว่าจะกดท่าเดียวกัน แต่ถ้าเทียบสัญญาณหลังจากการตัดและรวบแล้วจะได้ตรงกัน 4 จาก 5 สัญญาณ ซึ่งถูกต้องตรงกับการกดปุ่มจริง การทดลองต่อจากนี้จึงทำโดยอิงว่าการรวบสัญญาณเข้าด้วยกันช่วยให้ผลการเทียบสัญญาณออกมาถูกต้องขึ้น เมื่อตัดแบ่งช่วงสัญญาณที่จะนำมารวบอย่างเหมาะสม

ประเด็นต่อมาที่ต้องคำนึงถึงคือควรจะตัดช่วงสัญญาณอย่างไร ในงานวิจัยนี้ได้ใช้วิธีการตัดช่วงสัญญาณสองวิธี วิธีแรกคือตัดทุกช่วงคงที่ 15 เฟรม ค่า 15 เฟรมนี้ได้มาจากการทดลองดูผลการตัดในช่วง 5-25 เฟรมและค่านี้ให้ผลออกมาตรงกับการกดจริงที่สุด จะตัดน้อยกว่านี้ก็ไม่เหมาะเพราะว่าจะเสมือนไม่ได้ทำการรวบสัญญาณเลย หรือถ้าตัดมากกว่านี้ไปก็จะไม่เหมาะเช่นกัน เพราะสัญญาณจะถูกจับรวมกันมากเกินไปทำให้ผลออกมามั่วได้ เช่นถ้าหากกำหนดให้ความกว้างของการตัดช่วงสัญญาณกว้างถึงหนึ่งร้อยเฟรม สัญญาณในเฟรมที่ 100 ของผู้เล่น อาจจะถูกรวบมาตรงกับสัญญาณในเฟรมที่ 20 ของโกสเอไอโดยบังเอิญ ซึ่งไม่ถูกเพราะว่าเกิดห่างกันเป็นเวลานานมากเกินไป โดยในการตัดวิธีนี้จะเริ่มตัดทีละ 15 สัญญาณโดยเริ่มตั้งแต่เฟรมแรกสุดไล่ไปเรื่อยๆ ช่วงที่ตัดได้คือช่วง

1-15, 16-30, 31-45, ฯลฯ เป็นต้น ไม่ได้มีการพิจารณาการเริ่มตัดที่เฟรมใดๆ แล้วไล่ไปเรื่อยๆ อย่างเช่น 3-17, 18-33, 33-47, ฯลฯ เป็นต้น แม้ว่าจะมีความเป็นไปได้ที่การเลื่อนจุดเริ่มตัดออกไปจะส่งผลต่อความตรงกันของสัญญาณ แต่ยังไม่มีแนวทางใดที่สนับสนุนว่าควรจะเลื่อนจุดเริ่มในการตัดออกไปยังเฟรมใดดี ซึ่งมีความเป็นไปได้ตั้งแต่เฟรมแรกจนถึงเฟรมสุดท้ายของสัญญาณทั้งหมด ดังนั้นจึงไม่ใช้วิธีการเลื่อนเฟรม

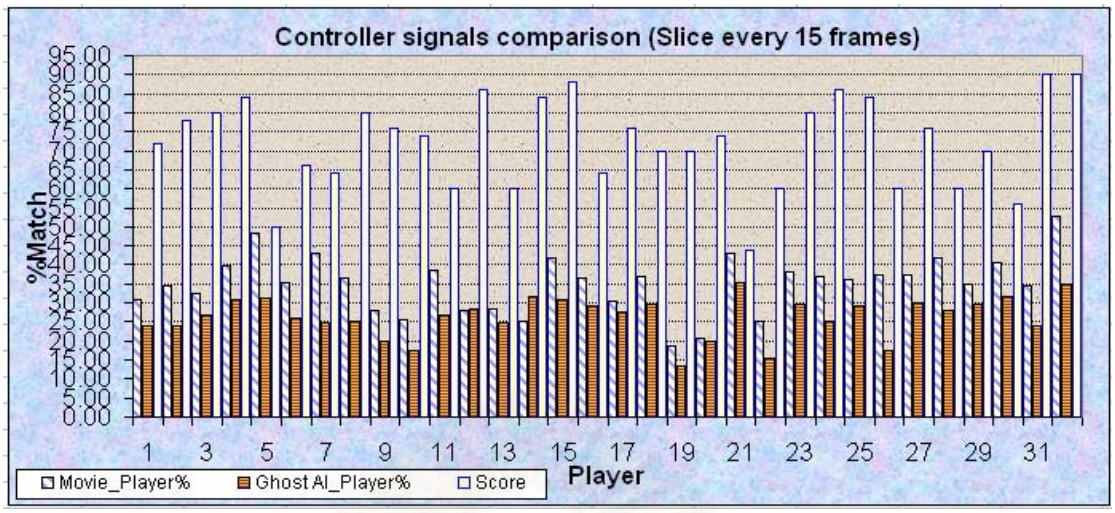
วิธีตัดสัญญาณวิธีที่สองคือตัดทุกช่วงที่สัญญาณจากโกสเอไอหรือว่ามูวี่ส์ของผู้เล่นเปลี่ยนสัญญาณ วิธีนี้จะทำให้เหลือหนึ่งสัญญาณเสมอในหนึ่งช่วงหลังจากการรวบสัญญาณแล้ว หลักการที่มาของวิธีนี้คือสัญญาณของทั้งสองฝ่ายควรจะโผล่ออกมาทับกันบ้างภายในช่วงเวลาที่สัญญาณยังอยู่จึงจะถือว่ากดตรงกันจริง

5.5 ผลการทดลอง

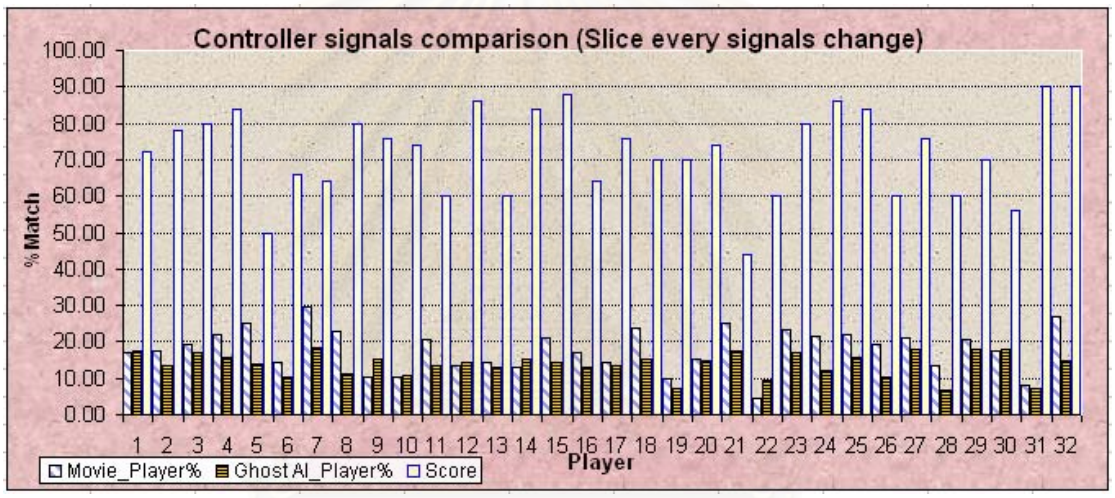
ส่วนนี้จะกล่าวถึงผลการทดลองที่ได้และผลการทดลองในอุดมคติควรจะเป็นอย่างไร

ในรูปที่ 27 เป็นการแสดงผลการเปรียบเทียบสัญญาณระหว่างมูวี่ส์ของผู้เล่นเองกับตัวผู้เล่นเอง (Movie_Player%) และ ผลการเทียบสัญญาณระหว่างโกสเอไอกับผู้เล่น (Ghost AI_Player%) ของผู้เล่นทุกคน เมื่อแบ่งช่วงสัญญาณทุก 15 เฟรม สาเหตุที่ต้องให้เทียบระหว่างมูวี่ส์ของผู้เล่นกับตัวผู้เล่นเองด้วยนั้นเพราะจะว่าถือว่ามูวี่ส์ของผู้เล่นเองคือโกสเอไอในอุดมคติที่สมบูรณ์แบบที่สุด ดังนั้นผลการเทียบสัญญาณระหว่างโกสเอไอกับผู้เล่นเองนั้นควรจะได้ใกล้เคียงกับของมูวี่ส์กับตัวผู้เล่นเอง กล่าวคือ Movie_Player% และ Ghost AI_Player% ควรจะได้ค่าใกล้เคียงกันและได้ความตรงกันสูง ส่วนค่า Score นั้นคือคะแนนความเหมือนของโกสเอไอที่ผู้เล่นเจ้าของให้คะแนนตามความรู้สึกส่วนตัว การให้คะแนนตามความรู้สึกของผู้เล่นนั้นจะให้ผู้เล่นให้คะแนนโดยบอกเป็นค่าทศนิยมหนึ่งตำแหน่ง โดยที่คะแนนเต็มคือห้าคะแนน และนำค่าคะแนนที่ผู้เล่นให้มาคูณยี่สิบเพื่อแสดงให้อยู่ในรูปคะแนนเต็มร้อยดังรูป ผลในอุดมคติของค่าคะแนนความเหมือนนี้ควรจะเป็น 100% เต็ม

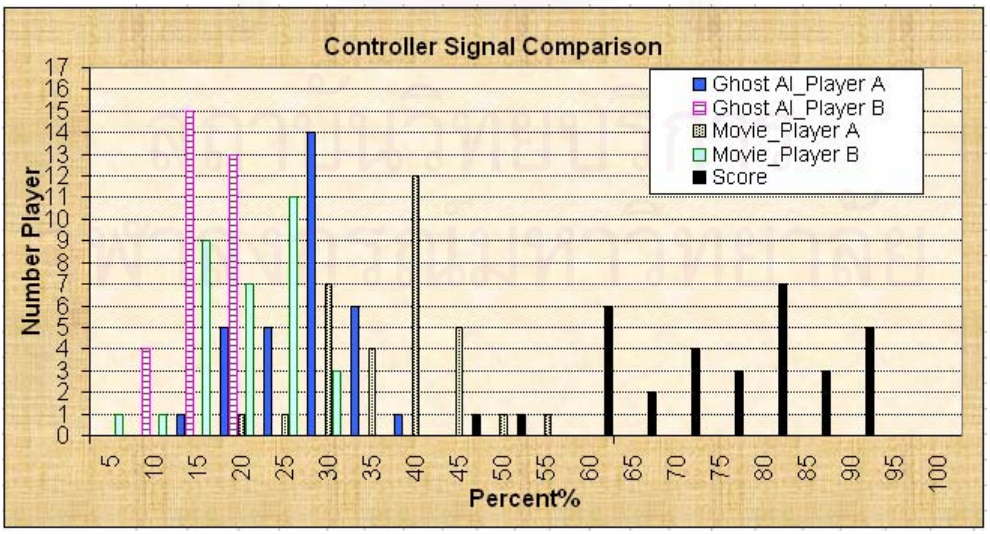
จากรูปที่ 27 จะดูเข้าใจได้ว่าผู้เล่นคนที่ 1 ทำการทดสอบเทียบกับมูวี่ส์ตัวเอง (Movie_Player%) ได้สัญญาณคอนโทรลเลอร์ตรงกัน 30.1% และเมื่อเขาเล่นกับโกสเอไอของเขา (Ghost AI_Player%) ก็ทำสัญญาณตรงกันได้ 24.2% และเขาให้คะแนนความเหมือนของโกสออกมาที่ 72% ส่วนรูปที่ 28 ก็เช่นเดียวกัน แต่แบ่งช่วงสัญญาณทุกครั้งที่สัญญาณของมูวี่ส์หรือของโกสเอไอเปลี่ยน



รูปที่ 27 แสดงผลการเปรียบเทียบสัญญาณคอนโทรลเลอร์แบบที่หนึ่ง



รูปที่ 28 แสดงผลการเปรียบเทียบสัญญาณคอนโทรลเลอร์แบบที่สอง



รูปที่ 29 แสดงจำนวนผู้เล่นและเปอร์เซ็นต์ของผลการเทียบสัญญาณแบบต่างๆ

Summary	Movie_ Player A	Ghost AI_ Player A	Movie_ Player B	Ghost AI_ Player B	Score
Min	18.81	13.6	4.37	6.54	44
Max	52.84	35.18	29.51	18.27	90
Average	34.96	26.33	17.93	13.81	72.2

รูปที่ 30 แสดงผลโดยสรุปของการทดลองเปรียบเทียบสัญญาณคอนโทรลเลอร์

รูปที่ 29 แสดงจำนวนผู้เล่นและเปอร์เซ็นต์ของผลการเทียบสัญญาณแบบต่างๆ ได้แก่ โกสเอไอกับผู้เล่นเมื่อแบ่งสัญญาณทุก 15 เฟรมและทุกช่วงที่สัญญาณเปลี่ยน (Ghost AI_Player A และ Ghost AI_Player B ตามลำดับ) มูวี่ส์ของผู้เล่นเองกับผู้เล่นเมื่อแบ่งสัญญาณทุก 15 เฟรมและทุกช่วงที่สัญญาณเปลี่ยน (Movie_Player A และ Movie_Player B ตามลำดับ) และค่าคะแนนความเหมือนตามความรู้สึกผู้เล่น (Score) เป็นการแสดงจำนวนผู้เล่นที่ทำให้เกิดผลการเทียบสัญญาณต่างๆ และคะแนนความเหมือนดังกล่าว ตัวอย่างเช่น มีผู้เล่น 14 คนที่ทำให้ผลการเทียบสัญญาณระหว่าง โกสเอไอและตัวผู้เล่นเองเมื่อแบ่งช่วงสัญญาณทุก 15 เฟรม (Ghost AI_Player A) มีค่าอยู่ในช่วง 25-30% และมีผู้เล่น 5 คนที่ให้คะแนนความเหมือนของโกสเอไอ (Score) ในช่วง 90-95% ผลอุดมคติของรูปนี้คืออัตราการตรงกันของการเทียบสัญญาณทั้งสี่แบบและคะแนนความเหมือนนั้นควรจะเป็น 100% ทั้งหมด ซึ่งแปลว่าผู้เล่นทดสอบกับมูวี่ส์ตัวเองแล้วกดตามได้ตรงกันทั้งหมดอีกทั้งโกสเอไอก็เล่นได้เหมือนคนเล่นมากจนกดปุ่มได้ตรงกับใจผู้เล่นทั้งหมด และผู้เล่นรู้สึกพอใจว่าโกสเล่นได้เหมือนตนเองอย่างที่สุด

รูปที่ 30 เป็นการแสดงค่ามากที่สุด น้อยที่สุดและค่าเฉลี่ยของผลการเทียบสัญญาณคอนโทรลเลอร์ระหว่างคู่ต่างๆ

ผลการเทียบแบบตัดสัญญาณทุก 15 เฟรม ระหว่างมูวี่ส์ของผู้เล่นกับตัวผู้เล่นเอง (Movie_Player A) มีค่าสูงสุดคือ 52.84% มีค่าต่ำสุดคือ 18.81% และค่าเฉลี่ยคือ 34.96% ระหว่างโกสเอไอกับผู้เล่น (Ghost AI_Player A) มีค่าสูงสุดคือ 35.18% ต่ำสุดคือ 13.6% เฉลี่ย 26.33%

และผลการเทียบแบบตัดสัญญาณทุกช่วงที่สัญญาณของมูวี่ส์หรือโกสเอไอเปลี่ยน ระหว่างมูวี่ส์ของผู้เล่นกับตัวผู้เล่นเอง (Movie_Player B) มีค่าสูงสุดคือ 29.51% มีค่าต่ำสุดคือ 4.37% และค่าเฉลี่ยคือ 17.93% ระหว่างโกสเอไอกับผู้เล่น (Ghost AI_Player B) มีค่าสูงสุดคือ 18.27% ต่ำสุดคือ 6.54% เฉลี่ย 13.81%

ส่วนผลคะแนนความเหมือนตามความรู้สึกของผู้เล่น (Score) ค่ามากสุดคือ 90% น้อยสุดคือ 44% และค่าเฉลี่ยอยู่ที่ 72.2%

5.6 วิเคราะห์ผลการทดลอง

โดยรวมแล้วกล่าวได้ว่าผลการวัดอัตราการเทียบสัญญาณคอนโทรลเลอร์ด้วยการแบ่งช่วงสัญญาณทั้งสองวิธีออกมาในแนวทางเดียวกัน

แม้ว่าอัตราการตรงกันของสัญญาณระหว่างผู้เล่นกับมูวี่ส์ของตนจะมากกว่าของโกสเอไอกับผู้เล่นแต่ความต่างนั้นมีค่าไม่มาก จากรูปที่ 30 ค่าเฉลี่ยของอัตราการตรงกันของสัญญาณระหว่างโกสเอไอกับผู้เล่นเมื่อใช้วิธีแบ่งช่วงสัญญาณทุก 15 เฟรมคือ 26.33% ส่วนมูวี่ส์กับผู้เล่นคือ 34.96%

การจะได้สัญญาณคอนโทรลเลอร์ออกมาตรงกัน 100% นั้นผู้เล่นจะต้องกดปุ่มได้ตรงจังหวะเหมือนกันทุกประการและตัดสินใจออกท่าเดียวกันในสถานการณ์เดียวกันทุกครั้ง แต่นั่นเป็นเรื่องที่เป็นไปไม่ได้ในความเป็นจริง เป็นไปไม่ได้ที่จะมีผู้เล่นที่ใช้เวลาในการกดปุ่มออกท่าทุกครั้งได้เวลาเดียวกันคงที่ตลอด และแม้แต่จะเป็นในสถานการณ์เดียวกันก็ตาม ก็ไม่มีผู้เล่นคนใดออกท่าเดียวกันซ้ำกันตลอด ทำให้เวลาทดลองมีโอกาสที่เจ้าตัวคนเล่นออกคนละท่ากับมูวี่ส์ที่บันทึกเอาไว้หรือท่าที่โกสเอไอเลือก แม้ว่าโกสเอไออาจจะมีท่าที่ผู้เล่นเลือกใช้เก็บไว้แล้วก็ตาม นี่คือสาเหตุสำคัญที่ทำให้ผลการเทียบสัญญาณออกมามีค่าไม่ตรงกันมาก

อีกสาเหตุหนึ่งคือวิธีการทดลองซึ่งผู้ทดสอบมากดปุ่มคอนโทรลเลอร์เพื่อบันทึกสัญญาณแต่ว่าไม่สามารถควบคุมตัวละครในเกมได้จริง ก็มีส่วนสำคัญที่ทำให้ผลของสัญญาณออกมาไม่ตรงกัน เนื่องจากเวลาที่ดูด้วยตาบนจอแล้วมากดปุ่มสัญญาณนั้นมักช้ากว่าที่มูวี่ส์หรือโกสเอไอเล่นไปแล้วเสมอ และเนื่องจากควบคุมตัวละครไม่ได้ ทำให้ต้องคอยสังเกตภาพบนจอและตัดสินใจตามว่าจะกดท่าอะไรออกต่อไป ทำให้เสียจังหวะในการกดปุ่มไปอีกเช่นกัน สาเหตุเหล่านี้รวมกันทำให้ความตรงกันของสัญญาณคอนโทรลเลอร์ลดลงไปอีก

นอกจากนี้จังหวะเวลาและความเร็วการกดท่าของโกสเอไอกับผู้เล่นแต่ละคนนั้นไม่เหมือนกัน แม้ว่าจะสั่งออกท่าเดียวกันก็ตาม ตามปรกติแล้วถ้าให้ผู้เล่นแต่ละคนกดท่าพิเศษก็จะมีจังหวะและความเร็วในการกดต่างกัน หรือแม้แต่ผู้เล่นคนเดียวกดหลายครั้งก็ยังกดไม่ได้ความเร็วเดียวกันทุกครั้ง ส่วนโกสเอไอนั้นจังหวะและความเร็วในการกดท่าพิเศษมีจังหวะคงที่และกดด้วยความเร็วค่าหนึ่งสม่ำเสมอ ทำให้แม้ว่าในจังหวะที่เดียวกันนั้นทั้งโกสเอไอและคนเล่นตัดสินใจออกท่าเดียวกันก็ตามแต่สัญญาณก็จะไม่ออกมาตรงกันโดยสมบูรณ์ ประเด็นนี้แม้ว่ากระบวนการตัดและรวบสัญญาณจะช่วยลดความต่างลงได้บ้างแต่ก็ไม่อาจจัดการทั้งหมด จึงเป็นสาเหตุอีกอย่างที่ทำให้ผลการเทียบสัญญาณระหว่างโกสเอไอกับผู้เล่นออกมามีความตรงกันน้อยกว่าของมูวี่ส์กับผู้เล่น

ตัวผู้เล่นเองแม้จะเล่นกับมูวี่ส์ของตนเองซึ่งถือว่าเป็นโกสเอไอในอุดมคติของเขาเอง สัญญาณก็ยังออกมาตรงกันเพียง 34.96% โดยเฉลี่ยเท่านั้น แสดงว่าตัวเลข 26.33% ของโกสเอไอกับผู้เล่นนั้นไม่ถือว่าน้อยแต่อย่างใด โกสเอไอทำให้สัญญาณตรงกันได้น้อยกว่าที่ผู้เล่นทำได้เพียง 8.61% เท่านั้นโดยเฉลี่ย จากข้อมูลนี้ถือว่าค่าความแตกต่างนี้น้อยและผลออกมาน่าพอใจ นอกจากนี้มียังมีผู้เล่นบางคนที่เล่นกับโกสเอไอแล้วให้สัญญาณตรงกันมากกว่าเล่นกับมูวี่ส์ของตัวเองด้วย

จากการพิจารณาผลการเล่นของโกสด้วยความรู้สึกของผู้เล่นและให้คะแนนตามความเห็น โดยเฉลี่ยแล้วคะแนนความเหมือนของโกสเอไอที่ประเมินได้คือ 72.2% กล่าวโดยรวมแล้วผู้ทดสอบรู้สึกว่าโกสเล่นได้มีส่วนเหมือนตัวเขาเองอย่างชัดเจนแม้ว่าจะไม่เหมือนอย่างสมบูรณ์แบบก็ตาม แต่สามารถดูออกได้ว่าเป็นการเล่นที่มีความพยายามในการเลียนแบบการเล่นของผู้เล่น แต่มีจุดสังเกตอย่าหนึ่งในการเล่นของโกสเอไอส่วนใหญ่นั่นคือโกสเอไอจะเล่นแบบขยันออกท่ามากกว่าเจ้าตัวคนเล่น โกสเอไอจะคอยออกท่าต่อเนื่องเสมอไม่ค่อยมีการหยุดพัก สาเหตุเป็นเพราะว่าโกสเอไอจะไม่มีการชะลอจังหวะเพื่อหยุดคิดตัดสินใจในการออกท่าต่อไปเหมือนคนเล่น ซึ่งผู้เล่นโดยทั่วไปจะมีการหยุดคิดบ้าง แต่เมื่อโกสเอไอทำงานโปรแกรมจะคอยเทียบสถานการณ์และเลือกท่าที่จะออกต่อเนื่องกันไปได้ตลอด ซึ่งประเด็นการขยันออกท่ามากเกินไปนี้สามารถปรับแก้โดยการเพิ่มคำสั่งให้โกสเอไอมีระยะเวลารอหรือแกล้งหยุดคิดก่อนที่จะออกท่าแต่ละจังหวะได้โดยต้องพิจารณาอัตราการชะลอการออกท่าของผู้เล่นแต่ละคนแล้วบันทึกลงในโกสเอไอไฟล์ต่อไปได้

บทที่ 6

สรุปผลการวิจัยและข้อเสนอแนะ

6.1 สรุปผลการวิจัย

ปัญหาในงานวิจัยนี้ส่วนแรกคือการขาดแคลนเทสเบดสำหรับใช้ทดสอบเอไอในเกมการต่อสู้ และปัญหาที่สองคือการค้นหาวิธีการสร้างโกสเอไอสำหรับเกมการต่อสู้ ดังนั้นงานวิทยานิพนธ์นี้จึงหาแนวทางและทดลองสร้างเทสเบดและหาวิธีการสร้างโกสเอไอสำหรับเกมการต่อสู้ขึ้นมา จากแนวคิดและการทดลองในบทที่ 5 จะสรุปผลได้ดังนี้

6.1.1 สรุปผลไอเทมเทสเบด

ไอเทมเป็นแนวคิดวิธีการในการสร้างเทสเบดจากอีมูเลเตอร์ของเครื่องเล่นเกม ด้วยการใช้ความสามารถในการรันเกมเล่นได้และเทคนิคการเข้าถึงข้อมูลของเกมผ่านทางเมมโมรีของอีมูเลเตอร์ วิธีการสร้างเทสเบดนี้ถูกพิสูจน์แล้วในงานวิจัยนี้ว่าสามารถใช้การได้จริง ได้ทำการทดลองสร้างเทสเบดจากอีมูเลเตอร์ของเครื่องเกมบอยแอดวานซ์ขึ้นมาและเลือกเกม สตรีทไฟท์เตอร์ซีโร่สาม เกมการต่อสู้ที่เป็นที่นิยม ซึ่งจะใช้เป็นกรณีศึกษาในการทดลองสร้างโกสเอไอ มาใช้ในการทดสอบเทสเบด ผลคือเทสเบดสามารถเข้าถึงข้อมูลสถานการณ์ต่างๆ ในเกมได้ถูกต้อง นั่นคือทำให้เอไอโมดูลสามารถรับรู้ได้ว่าตัวละครในเกมอยู่ที่ตำแหน่งใด ทำท่าอะไรอยู่และสถานการณ์ในเกมขณะนั้นเป็นอย่างไร ครบถ้วนตามที่ระบุในขอบเขตการวิจัย เอไอทดสอบที่ใส่เข้าไปในเทสเบดสามารถเข้าใจสถานการณ์ในเกมและควบคุมตัวละครในเกมได้ถูกต้องด้วยเช่นกัน และเทสเบดนี้ยังไม่ได้จำกัดวิธีการของเอไอที่จะใช้ทดสอบด้วย ขึ้นอยู่กับการปรับแต่งของผู้ใช้ว่าจะต้องการใช้งานอย่างไร ดังนั้นแล้วจึงสรุปได้ว่าแนวคิดวิธีการสร้างเทสเบดนี้สามารถนำไปพัฒนาและใช้การได้จริง

ในแง่ของการใช้งานแม้ว่าจะต้องมีกระบวนการในการเตรียมข้อมูลต่างๆ ก่อนหลายอย่าง แต่อย่างไรก็ตามขั้นตอนเหล่านั้นก็ยังสะดวกกว่าการสร้างเกมที่ดีเยี่ยมขึ้นมาเอง และกล่าวได้ว่าเป็นทางเดียวสำหรับการทดสอบเอไอบนเกมที่มีคุณภาพระดับโลกบางประเภทด้วย

6.1.2 สรุปผลส่วนโกสเอไอ

งานวิจัยนี้เสนอวิธีการสร้างโกสเอไอโดยการพิจารณาหาสถานการณ์ในเกมที่ผู้เล่นทำการเปลี่ยนแปลงกลุ่มของท่าของตัวละครที่บังคับ แล้วจึงดึงเอาสถานการณ์นั้นออกมาจับคู่กับท่าที่สั่งสร้างเป็นเคสออกมา จากนั้นจึงนำเคสเหล่านี้ไปรวมกันไว้เป็นข้อมูลอ้างอิงให้โกสเอไอใช้เวลาสู้จริง สิ่งสำคัญอีกอย่างหนึ่งที่งานวิจัยนี้ได้เสนอไปคือวิธีการพิจารณาว่าสถานการณ์ในเกมการต่อสู้จะถูกนำมาเปรียบเทียบกันอย่างไรว่าเป็นสถานการณ์เดียวกันหรือไม่ ถ้าหากเทียบแบบละเอียดจนเกินไปโอกาสที่สถานการณ์จะตรงกันจะน้อยและโกสเอไอจะเล่นไม่ค่อยออก แต่ถ้าหยาบเกินไปจะกลายเป็น

การเล่นมั่วแทน วิธีการเทียบสถานการณ์ที่เสนอคือการกำหนดหัวข้อสิ่งที่สนใจในเกมการต่อสู้ เช่น ตำแหน่งและท่าทางของตัวละคร จากนั้นจัดกลุ่มข้อมูลในหัวข้อเหล่านั้นออกเป็นกลุ่มให้เหมาะสม สถานการณ์ใดที่ตัวละครมีตำแหน่งและท่าทางตกอยู่ในกลุ่มเดียวกันให้ถือว่าเป็นสถานการณ์เดียวกัน ด้วยการออกแบบวิธีการจัดกลุ่มอย่างเหมาะสมทำให้สามารถจำแนกสถานการณ์ได้อย่างดี

ด้วยวิธีการทำงานเช่นนี้โกสเอไอจะมีรูปแบบการเล่นคงที่ตอบสนองต่อสถานการณ์เดียวกันด้วยท่ากลุ่มเดียวกันที่มีบันทึกเอาไว้เสมอ กล่าวคือในสถานการณ์หนึ่งโกสก็จะออกแต่ท่าที่ผู้เล่นเคยกระทำโดยเลือกออกตามสัดส่วนความถี่ของการทำท่าต่างๆในสถานการณ์นั้นของผู้เล่น ไม่มีการออกท่าที่ผู้เล่นไม่เคยกระทำและไม่กระทำการออกท่าอะไรในสถานการณ์ที่ไม่เคยเจอมาก่อน ดังนี้ถือเป็นความเหมือนในการเล่นได้ในระดับหนึ่ง

ได้ทำการวัดผลโดยการให้เจ้าของโกสเอไอมากดดูโกสเอไอเล่นไปพร้อมกับกดคอนโทรลเลอร์เพื่อพยายามบังคับควบคู่ไปด้วยเพื่อเทียบสัญญาณ และใช้ความเหมือนของการเทียบสัญญาณนี้บ่งบอกผลการทดลอง วิธีการเปรียบเทียบสัญญาณคอนโทรลเลอร์นี้ในความเป็นจริงแล้วมีหลายปัจจัยที่ทำให้ผลการเทียบสัญญาณออกมาไม่ค่อยตรงกันมาก เพราะว่าคนเล่นอาจจะตัดสินใจไม่เหมือนเดิมแม้จะเจอสถานการณ์เดียวกัน ซึ่งแม้มีบันทึกไว้ในเคสแต่โกสไม่แสดงออก ก็จะถือว่าเป็นสัญญาณไม่ตรงกัน และการที่ตัวผู้เล่นไม่อาจจะควบคุมตัวละครในเกมได้ในขณะที่ทำการทดสอบกดคอนโทรลเลอร์ ทำให้จังหวะการกดไม่เหมือนจริง กดไปแล้วตัวละครไม่ทำตาม จึงต้องมีการหยุดรอดูชะลอจังหวะ และอีกทั้งวิธีการกดท่าของผู้เล่นกับวิธีการกดท่ามาตรฐานของโกสเอไอนั้นไม่เหมือนกันอย่างสมบูรณ์ ทั้งหมดจึงเป็นเหตุให้ตัวเลขผลการเทียบสัญญาณออกมาดูน้อย อย่างไรก็ตามผลการทดลองของวิธีการนี้สามารถใช้เป็นแนวทางในการวัดผลได้

ผลการทดลองแสดงออกมาว่าอัตราความตรงกันของสัญญาณคอนโทรลเลอร์ของโกสเอไอและผู้เล่นออกมาต่ำ แต่เมื่อให้ผู้เล่นกดเทียบกับมูวี่ส์ของตัวเองที่ได้บันทึกเอาไว้ เพื่อดูเป็นแนวทางว่ามูวี่ส์ของผู้เล่นเองหรืออีกนัยหนึ่งคือโกสเอไอที่สมบูรณ์แบบที่สุดของผู้เล่นคนนั้นจะให้ผลออกมาเป็นอย่างไร ผลคืออัตราตรงกันของสัญญาณมีค่าต่ำเช่นกัน ดังนั้นผลการเทียบสัญญาณระหว่างโกสเอไอกับผู้เล่นนั้นจึงไม่ใช่ตัวเลขที่บอกถึงความแตกต่างหรือความเหมือนได้จริง นอกจากนี้ เมื่อให้เจ้าของโกสเอไอประเมินโกสเอไอของตัวเองด้วยตา ผลออกมาว่าเจ้าของส่วนใหญ่มีความเห็นว่าเหมือน อีกทั้งโกสเอไอยังได้แสดงรูปแบบการเล่นและการทำคอมโบตามแบบผู้เล่นได้ซึ่งเอไอดั้งเดิมของเกมไม่สามารถทำ แสดงให้เห็นว่าทำให้เกิดเอไอใหม่ที่เลียนแบบผู้เล่นขึ้นมาได้จริง

สรุปได้ว่าวิธีการสร้างโกสเอไอนี้สามารถสร้างโกสเอไอที่เล่นได้ดี มีรูปแบบการเล่นเหมือนกับตัวเจ้าของอยู่ในระดับที่สามารถนำไปประยุกต์ใช้งานได้จริง โดยมีผลการทดลองสนับสนุนได้แก่ ผลการเทียบสัญญาณแสดงให้เห็นว่าอยู่ในเกณฑ์ที่ยอมรับได้เมื่อพิจารณาถึงที่มาของผลลัพธ์ดังที่ได้

กล่าวไปแล้ว นอกจากนี้โดยรวมแล้วผู้ทดสอบรู้สึกว่าการเล่นของโกสเอไอมีการแสดงออกถึงความเหมือนตัวเขาเองในการเล่น

ในแง่ของเวลาที่ผู้ทดสอบใช้เล่นเพื่อสร้างโกสเอไอ จากการทดลองพบว่าโกสเอไอสามารถเล่นได้ดีแม้ว่าถูกสร้างจากข้อมูลการเล่นในเวลาสั้นประมาณสองนาที ซึ่งเท่ากับเวลาการเล่นเกมการต่อสู้เพียงหนึ่งถึงสองนัดโดยเฉลี่ย จำนวนเคสที่สร้างขึ้นมาในเวลาสองนาทีจะประมาณ 180 เคสโดยเฉลี่ย และเพิ่มขึ้นถึงประมาณ 350 เคสในเวลาสิบนาที อย่างไรก็ตามค่าเหล่านี้แกว่งมาก ผู้ทดสอบบางคนสามารถทำให้เกิด 400 เคสได้ในเวลาสี่นาที ยิ่งมีจำนวนเคสมากเท่าไรอัตราการเกิดเคสใหม่ก็จะน้อยลงเรื่อยๆ จำนวนเคสดังกล่าวไม่ได้เรียกว่าน้อยเกินไป โกสเอไอที่ถูกสร้างโดยมีเพียง 180 เคสไม่ได้ยืนนิ่งเวลาสู้และไม่ได้หาเคสไม่เจอบ่อยๆ จึงไม่มีความจำเป็นต้องสั่งให้โกสเอไอทำอะไรเพิ่มในกรณีที่หาเคสไม่เจอ เพียงแค่รอจนรอบต่อไปก็พอ

ผลของเวลาที่ผู้เล่นเล่นยาวขึ้นจะไม่มีการเพิ่มจำนวนเคสขึ้นมากนักและรูปแบบการเล่นของโกสเอไอจะไม่ค่อยเปลี่ยนแปลงถ้าหากว่าผู้เล่นและศัตรูยังคงเล่นรูปแบบเดิมไม่เปลี่ยนแปลงทั้งคู่ กล่าวคือวิธีนี้สร้างโกสเอไอที่ดีได้ในเวลาอันสั้น แต่หากผู้เล่นมีการเปลี่ยนแปลงรูปแบบการเล่นไปเรื่อยๆ ในทุกๆ ช่วงเวลา เช่นสองนาทีแรกเล่นบุกแบบหนึ่ง สองนาทีต่อมาเล่นตั้งรับอีกแบบ และอีกสองนาทีก็เล่นอีกวิธีเปลี่ยนไปเรื่อยๆ เช่นนี้โกสเอไอที่ได้ออกมาจะเป็นการผสมกันของทุกแบบทำให้ไม่สามารถรักษาแบบแผนการเล่นให้แน่นอนได้ในช่วงเวลาหนึ่งๆ ซึ่งจะต่างจากผู้เล่น ไม่ได้เหมือนช่วงสองนาทีใดๆ ช่วงใดช่วงหนึ่ง แต่นั่นไม่ใช่ประเด็นสำคัญ เพราะเป็นสิ่งที่เป็นไปตามธรรมชาติของวิธีการเช่นนี้อยู่แล้ว

อนึ่งจำนวนการออกท่าของศัตรูในระหว่างที่บันทึกการเล่นนั้นไม่จำเป็นจะต้องรอให้ศัตรูออกท่าครบทั้งหมด เนื่องจากได้สร้างเคสโดยคำนึงรายละเอียดของศัตรูแบบคร่าวเท่านั้นว่าอยู่ในท่าโจมตีอยู่หรืออมตะอยู่หรือไม่ การออกท่าโจมตีทั้งหลายของศัตรูจึงถูกมองเป็นท่าเดียวกันหมด เมื่อเป็นเช่นนั้นแล้วทำให้ไม่ต้องรอให้ศัตรูออกท่าจนครบก็สามารถสร้างโกสเอไอที่เล่นได้ดีขึ้นมาได้

ในแง่ของกระบวนการการสร้างโกสเอไอ วิธีการนี้ไม่เจาะจงกับตัวละครหรือเกมใด แต่เป็นหลักการสำหรับเกมการต่อสู้โดยทั่วไป ตัวละครริวและเกม สตรีทไฟท์เตอร์ซีโร่สาม เป็นเพียงตัวแทนในการแสดงผลการทดลองที่เป็นที่รู้จักและน่าเชื่อถือเท่านั้น วิธีการนี้สามารถนำไปปรับใช้กับเกมอื่นได้โดยไม่ยากนัก ไม่มีกระบวนการขั้นใดซับซ้อนและไม่สิ้นเปลืองพลังในการคำนวณหรือหน่วยความจำ

6.2 ประโยชน์ที่ได้รับ

1. ได้ต้นแบบวิธีการในการสร้างโกสเอไอสำหรับใช้กับเกมการต่อสู้
2. ได้เครื่องมือต้นแบบที่สามารถนำไปใช้ทดสอบปัญญาประดิษฐ์สำหรับเกมการต่อสู้
3. ได้สถาปัตยกรรมสำหรับสร้างเครื่องมือทดสอบปัญญาประดิษฐ์สำหรับเกม
4. ได้ความรู้เกี่ยวกับวิธีการทำงานของเอไอในเกมการต่อสู้

5. ได้ฐานข้อมูลของท่าทางต่างๆ ของตัวละครริวจากเกมสตรีทไฟเตอร์ซีโร่สาม

6.3 ข้อเสนอแนะ

6.3.1 การปรับปรุงส่วนเทสเบด

1 สร้างโมดูลที่ใช้อำนวยความสะดวกในการเล่นสองคน เช่นสั่งเปิดสองอินแสตนท์ของเทสเบดโดยอัตโนมัติ จัดการโหลดสเตทอัตโนมัติ และโหลดสคริปต์ทั้งสองฝ่ายเองอัตโนมัติ เพื่อให้จัดการแข่งขันของสคริปต์ได้สะดวก

2 พัฒนาอินเทอร์เฟสฟังก์ชันของไพธอนสคริปต์ให้ใช้ได้สะดวกขึ้น รวมถึงทำให้ทนทานต่อการเกิดข้อผิดพลาดทุกประเภท เทสเบดควรจะต้องไม่หยุดทำงานแม้ว่าสคริปต์ที่ส่งเข้ามารันจะเขียนผิดและแจ้งสาเหตุความผิดพลาดของสคริปต์ได้ เมื่อคนใช้เขียนผิดไม่ว่าจะกรณีใดก็ตาม งานเหล่านี้จะมีประโยชน์ในการใช้สอนเอไอหรือสอนเขียนโปรแกรมได้

3 เมื่อใช้วีบีเอรันสองอินแสตนท์เล่นด้วยกัน บางครั้งพบว่าเหตุการณ์ไม่ตรงกัน ซึ่งเหตุการณ์นี้เกิดค่อนข้างยากหากเล่นเองด้วยมือมนุษย์ แต่ถ้าใช้สคริปต์ที่มีการกดปุ่มถี่เกินไปเล่น โอกาสเกิดจะสูงมาก สาเหตุยังไม่ทราบแน่ชัดแต่คาดว่าเกี่ยวกับการที่กดปุ่มเปลี่ยนปุ่มเร็วเกินไป ปัญหานี้ควรจะหาทางแก้ไข แต่หากว่าไม่สามารถแก้ปัญหานี้ได้ ก็ยังมีวิธีต่อไปคือควรจัดการให้เอไอสคริปต์ของทั้งสองฝ่ายอ่านค่าและส่งผลลัพธ์สัญญาณการกดปุ่มไปยังอินแสตนท์เดียวกันของเทสเบดให้ได้ เพื่อที่จะไม่ต้องห่วงเรื่องความไม่ตรงกันของเหตุการณ์ระหว่างสองอินแสตนท์ ส่วนอีกอินแสตนท์ที่ไม่ตรงนั้นก็ไม่ต้องไปสนใจเลย เพียงแค่เปิดขึ้นมาเพื่อให้สามารถเล่นโหมดการเล่นสองคนได้ก็เท่านั้น

4 เปลี่ยนไปใช้โอเพนซอร์สอีมูเลเตอร์ มาเมะ ซึ่งสมบูรณ์กว่าสำหรับเกมต่อสู้เพราะว่าเป็นอีมูเลเตอร์ของเครื่องเกมตู้ต้นฉบับ และรันเล่นสองคนได้ในอินแสตนท์เดียว อีกทั้งเกมที่รองรับบนเครื่องยังมีเกมที่น่าสนใจอย่างสตรีทไฟท์เตอร์สามเทิร์ดสไตรค์ที่เป็นเกมที่ดีที่สุดของซีรี่ย์สตรีทไฟท์เตอร์อีกด้วย

5 ทดลองใช้ อิมเมจโปรเซสซิ่ง กับกับหลักการของไอเทม เพื่อที่จะไม่ต้องหาแอดเดรสของข้อมูล แต่ใช้วิธีมองจอภาพแล้วประมวลผลแทน ใช้สำหรับเกมที่หาแอดเดรสยากอย่างเช่นเกมฟุตบอลที่มีตัวละครมากมายในสนามและค่าข้อมูลเปลี่ยนแปลงอยู่ตลอดเวลา ทั้งยังไม่สามารถควบคุมตัวละครทั้ง 11 ได้สะดวกอีกด้วย

6.3.2 การปรับปรุงในส่วนของเกมสตรีทไฟท์เตอร์ซีโร่สาม

สำหรับ สตรีทไฟท์เตอร์ซีโร่สาม ยังมีงานในการสร้างฐานข้อมูลแบบเดียวกับที่ได้ใช้ในการทดลองนี้ แต่สำหรับตัวละครอื่นๆ นอกจากริวอีกกว่าสามสิบตัว หากสร้างได้ครบก็จะยิ่งเสริมความสมบูรณ์ของเทสเบดขึ้นไปอีก การจัดแข่งเอไอก็จะสามารถทดลองกับตัวละครอื่นๆ เพิ่มได้ด้วย

6.3.3 การปรับปรุงในส่วนของโกสเอไอ

1 มีบางกรณีที่จำเป็นต้องสแกนหาโดยเฉพาะ ไม่สามารถดูจาก อนิเมชั่น ที่เปลี่ยนแปลงได้เช่น นิสัยที่ผู้เล่นชอบกดการ์ดค้างไว้หรือพยายามจะสวนท่าเร็วกลับระหว่างที่โดนโจมตีด้วยคอมโบหรือไม่ เพราะหากศัตรูกดพลาดเมื่อไรการการ์ดหรือชกเบาสวนจะมีผลทันที แต่หากว่าศัตรูกดไม่พลาดเลยก็จะไม่ได้เห็นความแตกต่างและไม่เห็นนิสัยการกดเช่นนี้ของผู้เล่น พฤติกรรมเช่นนี้ยังไม่สามารถดึงออกมาได้ด้วยการแสกนตามขั้นตอนปัจจุบัน จะต้องอาศัยการพิจารณาปุ่มที่ผู้เล่นกดในเฟรมต่างๆ ประกอบเพิ่มด้วย

2 ควรแยกโกสเอไอของผู้เล่นคนเดียวกันสำหรับศัตรูแต่ละตัวด้วย เนื่องจากรูปแบบการเล่นของผู้เล่นมักจะเปลี่ยนแปลงไปเมื่อต้องสู้กับคู่ต่อสู้แต่ละแบบ ศัตรูที่ถนัดการโจมตีระยะไกลจะทำให้ผู้เล่นต้องตั้งรับเป็นหลัก แต่กับศัตรูที่เชื่องช้าผู้เล่นมักจะชอบบุก หรือการที่ขนาดตัวศัตรูเปลี่ยนแปลงไปก็ทำให้การต่อท่าหรือออกท่าของผู้เล่นเปลี่ยนแปลงไปได้เช่นกัน นอกจากนี้การแยกโกสเอไอออกเป็นหลายไฟล์ยังมีประโยชน์ในการแก้ปัญหาเวลาที่ผู้เล่นเปลี่ยนรูปแบบการต่อสู้ได้ด้วย

3 สร้างระบบที่สามารถแยกแยะรูปแบบการเล่นและสถานการณ์ที่ใช้รูปแบบการเล่นต่างกันของผู้เล่นได้ เพื่อบันทึกการเล่นที่มีรูปแบบต่างกันลงในคนละไฟล์ ทั้งนี้เพื่อให้โกสเอไอเลือกเปลี่ยนรูปแบบการเล่นได้ตามแบบผู้เล่นต้นบบ

รายการอ้างอิง

[1] Namco Corporation. Tekken 5: Dark Resurrection [Online]. Available from: http://www.tekken-official.jp/tk5dr/index.html [2008, 20 Jan]

[2] Pieter Spronck, Ida Sprinkhuizen-Kuyper and Eric Postma. Online Adaptation of Game Opponent AI with Dynamic Scripting. International Journal of Intelligent Games and Simulation March/April 2004, Vol.3, No.1, ISSN: 1477-2043, pp. 45-53. University of Wolverhampton and EUROSIS.

[3] Kendall Graham and Kristian Spoerer. Scripting the Game of Lemmings with a Genetic Algorithm. Proceedings of the 2004 Congress on Evolutionary Computation, pp.117-124. 2004.

[4] Pedro Demasi and Adriano J. de O. Cruz. Online CoEvolution for Action Games. Proceeding of GAME'ON'2002, Simulation and AI in Computer Games Conference, 2002.

[5] Thore Graepel, Ralf Herbrich and Julian Gold. Learning to Fight. Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education, 2004.

[6] Capcom Corporation. Street Fighter Zero3 [Online]. Available from: http://www3.capcom.co.jp [2008, 20 Jan]

[7] VisualboyAdvance Team. VisualboyAdvance [Online]. Available from: http://vba.ngemu.com [2008, 20 Jan]

[8] Nintendo Corporation. GameboyAdvance [Online]. Available from: http://www.gameboy.com [2008, 20 Jan]

[9] Sega Corporation. Virtua Fighter 4 [Online]. Available from: http://www.virtua-fighter-4.com [2008, 20 Jan]

[10] Tekken Zaibatsu. Tekken ghost AI community [Online]. Available from: http://www.tekkenzaibatsu.com/forums/ghostlist.php [2008, 20 Jan]

[11] Stratagus Team. Stratagus Project [Online]. Available from:

http://www.stratagus.org [2008, 20 Jan]

[12] Spring Team. Spring Project [Online]. Available from:

http://spring.clan-sy.com/download.php [2008, 20 Jan]

[13] VisualboyAdvance Link Team. VisualboyAdvance Link [Online]. Available from:

http://vbalink.wz.cz/index.htm [2008, 20 Jan]

[14] Leight David. Street Fighter Alpha Anthology. Bradygames, Printed in USA, 2006.

[15] Sander Bakkes and Pieter Spronck. Gathering and Utilising Domain Knowledge in Commercial Computer Games. Proceedings of the 18th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC 2006), pp. 35-42. 2006.

[16] Pieter Spronck, Ida G. Sprinkhuizen-Kuyper and Eric O. Postma: Enhancing the Performance of Dynamic Scripting in Computer Games. International Conference on Entertainment Computing (ICEC), pp. 296-307. 2004.

[17] Marc J.V. Ponsen, Héctor Muñoz-Avila, Pieter Spronck, and David W. Aha. Automatically Acquiring Adaptive Real-Time Strategy Game Opponents Using Evolutionary Learning . Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, pp. 1535-1540. 2005.

[18] Christine Bailey and Michael Katchabaw. An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games. Proceeding of GAME'ON'2005, Simulation and AI in Computer Games Conference, 2005.

[19] Python Software Foundation. Python Language [Online]. Available from:

http://www.python.org [2008, 20 Jan]

[20] All About Street Fighter Zero3: All About series vol.21, Studio BENT STUFF, Printed in Japan, 1998.

ภาคผนวก

ภาคผนวก ก ผลงานการตีพิมพ์

1. Thunputtarakul Worapoj and Kotrajaras Vishnu. 2006. AI-TEM: Testing Artificial Intelligence in Commercial Game using Emulator. *8th CGAMES International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games.* Louisville Kentucky, USA.

2. Thunputtarakul Worapoj and Kotrajaras Vishnu. 2007. Data Analysis for Ghost AI Creation in Commercial Game. *GAME-ON The European Simulation and AI in Games Conference.* Bologna, Italy.

AI-TEM: TESTING AI IN COMMERCIAL GAME WITH EMULATOR

Worapoj Thunputtarakul and Vishnu Kotrajaras
Department of Computer Engineering
Chulalongkorn University, Bangkok, Thailand
E-mail: worapoj.t@student.chula.ac.th, vishnu@cp.eng.chula.ac.th

KEYWORDS
Testbed , Artificial Intelligence, Commercial Game.

ABSTRACT

Many artificial intelligence (AI) game researchers find that it is difficult to find a game environment that they can appropriately test their AI on. They usually have to develop parts of an existing game, using tools that come with the game. Some even have to re-write their testing game from scratch. Finding a perfect game environment that one can use to test his AI is not easy, especially if a commercial-quality game is required. Huge amount of time and effort are lost in finding such ideal testbed. This paper presents AI-TEM environment, a testing environment for testing AI by using console game emulator and their ROM data to simulate and run a commercial game. AI-TEM can be used to plug many AI onto many commercial games. Researchers interested in higher-level abstractions of game AI can test their already developed AI algorithm on a commercial game. We believe that AI-TEM adds a wider range of possibilities to AI testing.

1 INTRODUCTION

Research and developments related to computer games have always focused on graphical technology. However, players have begun to demand for more playability. Recent games have incorporated smart AI into their gameplay and became very successful because of that. Therefore, research in AI is important for the game industry. On the other hand, games provide interesting testing environments for AI researchers.

One problem faced by many researchers is to find or develop a proper game environment to use in their AI testing (Graepel et al 2004, Kendall and Spoerer 2004, Ponsen et al 2005). A game that should be used to test AI should have the following qualities: It should be a game that has many ways to play, many ways to win, and the game should be complex enough to separate expert players from novice players. It should be a commercial quality game. Because if researcher's AI can win against its initial game AI, then researchers can claim that the newly developed AI truly has enough quality and efficiency to use in commercial game (Spronck et al 2004).

There are testbeds developed for testing AI (Aha and Molineaux 2004, Bailey and Katchabaw 2005), but many of them do not come with a complete game ready to be used. Researchers will have to find a game or game engine to integrate with it. Large amount of time may be used.

This paper proposes another way to test game AI in an environment that has been well made and well designed, called AI-TEM (AI-Testbed in EMulator). AI-TEM uses an emulator of Game Boy Advance (GBA), developed to test many AI methods.

In this paper, an emulator means a game console/handheld emulator that simulates the working of game console/ handheld hardware such as GBA, PlayStation, and arcade machine on any personal computer. There are many emulators of console/handheld game hardware. VisualboyAdvance (VBA) (VisualboyAdvance 2005) and VisualboyAdvance Link (VBA Link) (VisualboyAdvance Link 2005) are GBA emulators. ePSXe is a PlayStation emulator. Even arcade machines have MAME as their emulators.

ROM (Read Only Memory) is the game data dumped from the original game cartridge or disc. Using a game ROM with its emulator, a game can be simulated and played on PC.

We used GBA emulator, VisualboyAdvance, for developing a prototype of AI-TEM. And we used Street Fighter Zero 3 (STZ3) game ROM as our test ROM, so that we could experiment and write additional tools for a real example. VBA is open-source, therefore we can modify its functions. There are many interesting games on GBA that can be used to test AI. VBA also has plenty of resources, technical documents and support tools provided for us. The VBA Link is an extended version of VBA. The VBA Link team modifies original source code to make linkage possible. In this paper we will call both VBA and VBA Link as VBA.

STZ3 is a fighting game. In this type of game, a player must select one character from many characters, and fight one by one with an opponent character. A player must decide what action he will perform in many different situations based on his character and opponent character's status. Therefore, we believe this is a good game for tuning our testbed and for AI research.

2 AI-TEM FRAMEWORK

The concept of AI-TEM is generally simple but there is some low-level work involved. Researcher's AI may need to know game state data, such as object position or character animation, but it cannot access the source code of the game. The AI can only access the source code of the emulator. So we must get the game state data from memory data the emulator is emulating. We can see only a binary (hexadecimal) value of game data that changes in every cycle of a game execution. We must therefore find out which address stores the value that we are interested in such as position, animation, etc. We will use values in those addresses as game state data for AI testing. When we know the game state, AI can be written to react in each situation, by sending a controller input, or forcing memory address value. Using this concept, we can use AI-TEM as an AI testing tool.

Finding each address that stores those game state data is difficult if done manually. Some values can be found easily, while some are rather hard to find. User should have some knowledge about programming in order to be able to identify address more successfully. Some examples of how to find the address of game data are demonstrated below.

Example 1: Finding address of character health. Starting by identifying all the values used in the game. Then the game is played and the character health is forced to decrease. The value that represents the character's health should in fact decrease too. All game values are then searched and compared with values before the health decrease. It is common to find many values decreasing. The process should be repeated, with different health, until one address is identified.

Example 2: Character's bullet position. The concept is the
same as character's health example. When a bullet moves forward, its position value should increase respective to its position. But the time period that bullet is alive is very short. Therefore, repeating the experiment as many times as one wants becomes difficult. If users must press a command every time that they want to find a value, it will not be convenient. Tool to fire character bullet many times is needed, such as movie recorder (section 3.0).

We had developed some tools to help finding the address more easily and will describe it in section 3.0 below. There are other techniques to find values that we will not discuss in this paper.

Therefore, in the beginning phase of using AI-TEM, the user must find the address of game state data that their AI module needs to know. AI-TEM is depicted in figure 1 and 2, and discussed in more detail in the sections below.

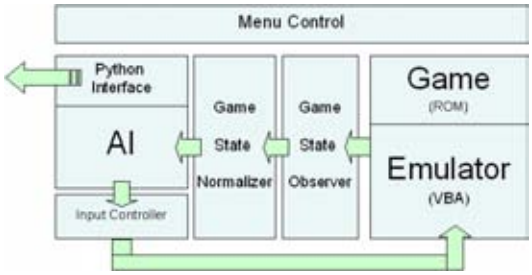


Figure 1: AI-TEM system overview.

2.1 Emulator Core (VBA)

The core of this testbed is VBA (Link) emulator. It is used to run game ROM and simulate the game. It also has many tools useful for getting game state and testing AI that will be described in section 3.1.

2.2 Menu Control

We add a menu into emulator to control the working of AI-TEM. To turn on/off AI module or switch between different AI modules. We can also activate other utilities that system may want to use.

2.3 Game State Observer

A game state can be known by observing data on the memory address of the emulator and locating which address stores the data that we want to know. (In STZ3, game states that we are interested in consist of position of a character, position of the character bullet, the character's health, and current animation of the character.) We implemented this module by modifying the memory viewer tools of VBA. The user can identify address and size of data (8, 16, 32 bits) that they are interested in. When AI-TEM is running, in every frame, Game State Observer will copy the value from those addresses to the data structure that an AI module can use.

2.4 Game State Normalizer

Before Game State Observer sends game state data to AI module's data structure, the game state data must be normalized or interpreted, depending on the game and format of data that we got from memory of emulator. Example: For STZ3, we use address 0x20007C2 (16bits) as the address that stores the character 1P position in the X axis. The range of value that we got from that address is 44 (002C) to 620 (026C), 576 units. But when using it, we should normalize x position value to 0 - 576 for user friendliness. Therefore, we must subtract 44 from the value copied from the memory of emulator. This normalization process is not necessary if researchers do not care about the format of raw data from emulator's memory. We currently provide this module as a code template for users to modify.

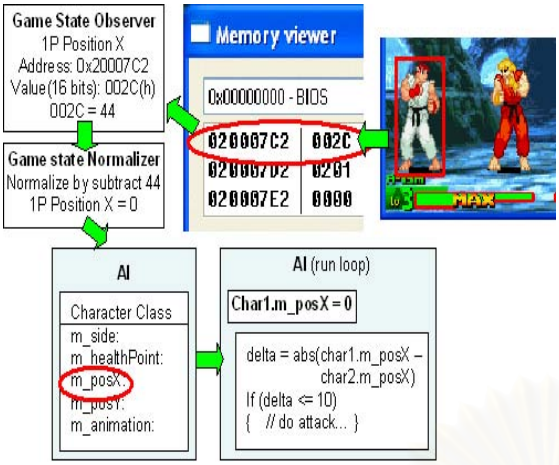


Figure 2: Work flow diagram of AI-TEM system.

2.5 AI

This module is where a user of AI-TEM will put his AI module in. In every cycle of emulation, the emulator will execute this module. This module evaluates the game data and decides what controller input it will send to the controller module. There is enough of VBA CPU power for calculating non-intensive work, such as script (Dynamic script (Spronck et al 2004), Genetic Programming result script, etc.). With extension, other AI methodologies can also be added. In our experiment with the system, we write two static scripts for testing the use of AI-TEM. The detail and result of this script will be shown below in section 4.1.

2.6 Python Script Interface

Python (Python 2006) is an interpreted, interactive, object-oriented programming language. It is also usable as an extension language for applications that need a programmable interface. Python is portable: it runs on many OS such as Windows and Linux. It is one of the most famous script languages used in many applications.

We modified the emulator to have an ability to use python
script language, providing interface functions for a script writer to obtain game state data and to control the game via any AI module. A script writer can write their python script separately without running the emulator, and can change script without recompiling AI-TEM. This will benefit users who want to test their AI with static script. If researchers can generate AI output in python script format, it can be tested conveniently without the need of rerunning the game. Example of an interface function used in the testing of STZ3 is shown below.

```
int GetCharacterPositionX (int C)
int PressButton(int button)
```

The first function will return a position in x axis of character C. The second function will send a parameter 'button' to the Input Controller module. It allows a python script to command character. Below is the example of python script that uses those interface functions.

```
import myLib
def Main_AI_Run_Loop():
if (myLib.GetCharacterPositionX(P2) <10)
{
   return myLib.PressButton(PRESS_B)
}
return 0
```

myLib is a library of interface functions that we provide from AI-TEM, it allows user of python script to use function GetCharacterPositionX and PressButton. This script results in character kicking (press B) when its opponent comes closer than a specified threshold.

2.7 Input Controller

Original VBA will capture signals from joystick or keyboard and send them as input to a game. We modified the system so that our AI module can replace input signals from normal controller with its own signals.

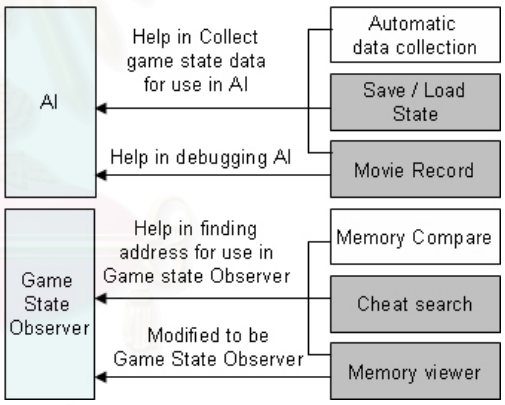


Figure 3: Usage of tools in AI-TEM system.

3 AI-TEM UTILITY TOOLS

Other than modules described in section 2.0, AI-TEM has other utility tools that can help in many tasks. Some of them are original VBA tools that we use in our testbed. Some are modified tools we made for our own use. The working of these tools is shown in figure 3 and described below.

3.1 VBA Tools

These are VBA original tools that we had used in AI-TEM.

Memory Viewer: used for displaying content of every memory data address in a variety of formats, 8, 16, 32 bits, sign, unsigned and hexadecimal. Our Game State Observer is modified from this tool.

Cheat Search: this tool is originally used for finding an address of data that we want to find, by searching all of memory and finding a value that match a condition given by user. For example, user can use it to find a value in the address that is equal, greater or less than some specific value. In AI-TEM this tool is used to help in finding address that Game State Observer will observe.

Movie Recorder: this tool can record game movie in two formats. VMV format will record only initial game state and inputs given by controller. It has a very small file but can playback only in emulator. AVI can playback in many movie player programs but its file is larger and uses a lot of CPU power. Movie recorder can help in a data collection process (section 3.2) and can help recording the testing output or debugging.

Save/Load State: When running game in emulator, the game state can be saved and reloaded to continue to play at the same point where it was saved. This ability is useful when researchers want to test decision condition of AI. They can save game state before their AI make decision and can reload it to try another decision in perfectly the same situation.

3.2 Modified Tools

Memory compare tool: As said in section 2.0, finding address of value that AI module needs to know is difficult. Therefore, we modified the original VBA tools to be Memory compare tool. This tool will help in finding an address of data, by comparing many game states data, given a condition of data that users are seeking. Example: User wants to find the address of character's health in STZ3. They will dump game states of various situations from the emulator. We define the game state of situation N as GSn.

GS1: character's health is 100%.
GS2: character's health is 50%.
GS3: character's health is 75%.

And user will set a condition of the value they want to find. In this case, a health value address will have conditions as follows: Values in our required address from state GS1 must be greater than values from state GS2. Values in our required address from state GS2 must be less than values from state GS3. And values in our required address from state GS3 must be less than values from state GS1. This tool will compare every data in those game states and find the address that matches all of conditions that user provides automatically, without any need to run the original cheat search (The cheat search tool requires user to repeatedly experiment and find any address manually.). If user provides enough game states and proper conditions, finding the address should be straightforward. We believe this tool can save a great deal of time finding those values.

Automatic Data Collector: There are some situations that we want to collect a lot of data from game state. It is difficult to collect them manually. Example: In order to imitate human reaction, AI module must know the animation of both characters in order to decide what it will do next. For example, if an opponent character is going to punch, our character must detect the opponent's movement and perform a guard. After knowing the address that stores values of character animation, we need to find out which value in that address corresponds to which animation. (for example, 0 means stand, 72 means crouch, 124 means jumping) Therefore, we need some tools to help collecting game data (character animation data).

In STZ3, we modified the original VBA function that was used for forcing values of addresses (Cheat function) to be a tool for helping us to collect animation data. By writing a value from the start animation value to the end of animation value, we forced each character to do all of its actions. We captured the character's image of each action and saved it with its animation value as its file name. We then knew the animation value for each of a character's action. Human intervention was needed to identify the meaning of each action. An animation database was then produced.

4 EXAMPLE EXPERIMENT IN STZ3

This section will discuss the result of using AI-TEM with STZ3 ROM to create a simple, static script AI. Table 1 contains the addresses of STZ3 game state data that we know by the method discussed in section 2.0. Table 2 contains some character animations from a character named RYU, which we collected after normalization, by using stand animation as a basis. (Each animation is composed of many frames, so there are many values in each animation

Table 1**:** Example address of STZ3 game state data.

Game State Data	Address	Data Size
character 1 position x axis	0x20007C2	16 bits
character 1 position y axis	0x20007C4	16 bits
character 2 position x axis	0x20043D2	16 bits
character 2 position y axis	0x20043D4	16 bits
character 1 Animation	0x20007D0	32 bits
character 2 Animation	0x20043E0	32 bits

Table 2**:** Example of character (RYU) animation.

Animation	Value
Stand	0, 12, 24, 36, 48, 60
Crouch	276, 288, 300, 312, 72, 228, 240, 252, 264

Jump	420, 432, 468, 480, 492, 504, 516, 528, 444, 456
Guard	744, 756, 768, 780, 792
Punch (light)	9192, 9204, 9216, 9228

4.1 AI Script Experiment

In order to test our AI module and python interface, we had implemented a static script to control a character in STZ3. Our test condition for our static script is the character RYU VS RYU in versus mode. This static script also allows us to test our AI in a controlled situation. We implemented a static script for character RYU. Our 1P's RYU can detect states of original game AI 2P's RYU.

Our first version of the script just randomly performs action. The result was not as bad as we originally believed. Even though it had no intelligence, it performed action continuously and was able to beat the original game AI at the easiest level. We improved our script in many aspects, using animation data that we collected. Our static AI can now sense distance between characters. We also script it not to use special moves often, since special moves leave characters defenseless. More combination attacks were also added. We obtain a better result, as expected. Our static AI can now beat the original game AI in middle level.

This experiment convinced us that AI can make use of the game state and animation of the opponent by using data in section 4. We also have a fully working python interface ready for creating future controlled situation.

5 DISCUSSION

5.1 Outline steps of Using AI-TEM

Researchers must first find a game that is suitable for testing their AI method or matches their experimental plan.

Researchers then identify the game states data that their AI module needs to know. In normal AI method, such as scripting, an AI module needs to know only current situations of the game. But in some AI method such as some type of Reinforcement Learning, it needs to know a complete set of actions that the agent can perform in every situation.

After that, they must find the address of game state data that their AI needs to know.

After the addresses are found, data must be collected from those addresses and translated into a form that the AI can understand.

Finally, AI can be implemented. This topic will be discussed in more detail in section 5.2.

5.2 Which AI method can AI-TEM be used with?

AI-TEM does not limit AI methods that it can be used with, because its concept is only using an emulator as game engine. Some AI methodology, however, requires extra functions. For example, using Genetic Algorithm requires running tests large amount of times, may be hundreds or thousands generation. Therefore, automatic result recorder is needed. High speed running mode will also be an additional welcome, for it can save time to train AI.

High speed mode is already available in VBA and many other emulators. Not all games may allow us to provide automatic running mode. This is because, if we cannot find memory data address that tells us about the beginning and the end of the game, we cannot force the situation. But in general, automatic result collection can be done. Therefore, various AI techniques can be used.

5.3 What kind of Game/AI-Subject should use AI-TEM?

If researchers are interested in first person shooter, real-time strategies or D&D-styles RPG game, there are games that come with tools. Good testing environment for such games can be built with such tools. Also, there are very few of these games on consoles. AI-TEM may not be the first choice for testing such games. If researchers are interested in simple platform action game, writing a game from scratch or finding some open source clone game is not a bad choice because all environments of the game can be fully controlled. However, developing games, from tools provided by a game, or from an open source clone cannot easily get us commercial-quality game. This is where AI-TEM can come in. AI-TEM can be used to test an AI developed on a simple, but fully controllable environment, against real commercial game. In the case of racing game it is difficult to know game state data such as opponent car position and the track situation. As a result, AI-TEM will not be appropriate. For fighting game we think that using AI-TEM is suitable, because this type of game is rather difficult to make and even more difficult to make it as good as commercial game. Therefore, we think the tradeoff in the case of fighting game is worthwhile. For sport game, we think that it is still suitable to use AI-TEM, because of the same reason as fighting game. Even though there may be many game states that an AI module needs to know, finding them may be easier than creating a high quality sport game from scratch. Some AI researchers use Robocup simulation league to be a testbed for their football AI research (Sean Luke 1997). However, Robocup simulation league rules are still not the same as real football rules.

For other types of games/AI-subject, researchers have to consider the same factors as in this section.

5.4 The Limitation of using VBA in AI-TEM

To play a multiplayer mode in VBA (Link), two or more instances of emulators have to be used. Controlling many emulators at the same time while testing is not very convenient. It will be better if

the second instance can run in the screen-off mode, in order to save CPU power.

Sometimes two connected emulators do not synchronize. This may damage the automatic module in long run. Detecting game state of both VBAs becomes necessary. We can then reload the game again if they do not synchronize.

Although there are some inconveniences, AI-TEM generally works well in our experiment. The emulator can be fixed to tackle the problem.

6 CONCLUSION AND FUTURE WORK

Our work provides an environment for testing AI on a wider range of commercial-quality games. Our experiment shows that, with appropriate game ROM, AI-TEM meets the three requirements in section 1 (test with a commercial-quality game, the game should not be too simple and there are many ways to play the game). Researchers can use AI-TEM to test their AI against the game's original AI or against a human opponent. Emulator players form huge communities, therefore many players can help with AI testing. AI developed by researchers can also be tested against AI running on script. A well designed script can help an evolutionary or learning AI improve in an appropriate direction. Although the GBA is not as powerful as next generation hardware, many games on GBA are regarded as classics and have been re-released on several new platforms. Therefore, AI-TEM is very much viable as testing environment for commercial-quality games. And the framework of AI-TEM should be adapted to more emulators in the future.

We plan to improve the implementation of the system and
its associated tools. To provide a package for AI research in the future, we plan to collect the animation data of all characters in STZ3. We also want to build a cooperative AI for sport games using our testbed. The game WORLD SOCCER Winning Eleven is a perfect candidate ROM for the task. We also have plans to use another emulator with AI-TEM, such as MAME or ePSXe, in order to access more types of games. Multiplayer games can be run on MAME and ePSXe without synchronization or performance problems because only a single instance of emulator is needed.

REFERENCES

Aha, D.W., & Molineaux, M. 2004. Integrating learning in interactive gaming simulators. Challenges of Game AI: Proceedings of the AAAI'04 Workshop (Technical Report WS-04-04). San Jose, CA: AAAI Press

Bailey, C. and M. J. Katchabaw. 2005. An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games. Proceedings of the 2005 GameOn North America Conference. Montreal, Canada.

Graepel Thore, Ralf Herbrich, Julian Gold. 2004. Learning to fight. International Conference on Computer Games: Artificial Intelligence, Design and Education.

Kendall Graham, Kristian Spoerer. 2004. Scripting the Game of Lemmings with a Genetic Algorithm. Proceedings of the 2004 Congress on Evolutionary Computation, IEEE Press, Piscataway, NJ, pp.117-124

Ponsen Marc J.V., Hector Munoz-Avila, Pieter Spronck, and David W. Aha. 2005. Automatically Acquiring Domain
Knowledge For Adaptive Game AI Using Evolutionary Learning. Proceedings The Twentieth National Conference
on Artificial Intelligence.

Sean Luke, Charles Hohn, Jonathan Farris, Gary Jackson,
James Hendler. 1997. Co-Evolving Soccer Softbot Team Coordination with Genetic Programming. First International Workshop on RoboCup, at the International Joint Conference on Artificial Intelligence.

Spronck Pieter, Ida Sprinkhuizen-Juyper, Eric Postma. 2004. Online Adaptation Of Game Opponent AI With Dynamic Scripting. International Journal of Intelligent Games and Simulation, Vol. 3, No. 1, University of Wolverhampton and EUROSIS, pp.45-53.

Python (2006). Python Language
http://www.python.org

VisualboyAdvance. (2005). GBA Emulator
http://vba.ngemu.com

VisualboyAdvance Link. (2005). GBA Emulator
http://vbalink.wz.cz/index.htm

DATA ANALYSIS FOR GHOST AI CREATION IN COMMERCIAL FIGHTING GAMES

Worapoj Thunputtarakul and Vishnu Kotrajaras
Department of Computer Engineering
Chulalongkorn University Bangkok Thailand
worapoj.t@student.chula.ac.th, vishnu@cp.eng.chula.ac.th

KEYWORDS: Ghost AI, Fighting Game, Case base

ABSTRACT

In this paper we present a simple, rapid and efficient method for creating a ghost AI, an Artificial Intelligence that can imitate playing styles of players in fighting games. The created ghost AI can perform combination actions and make a decision about any movement in a similar fashion to a player it is copying. We scan a player's battle data, and then create situation-action pair cases for its corresponding ghost AI to use in actual battles. A ghost AI can be created and run swiftly, using small amounts of memory, making it suitable for console games. Our method is general enough to be used in most 2D and 3D fighting games. We carried out our experiment on Street Fighter Zero 3, one of the most well crafted fighting games, using AI-TEM testbed engine.

1. INTRODUCTION

1.1 Ghost AI

In fighting games there have been various attempts at ghost AIs (AIs that imitate players). Virtua Fighter 4 allowed players to train computer AIs to fight like them. Such ghosts could then be assigned to fight another player. However, feedback from players was not good at the time the game was released because it was hard to train their ghosts case by case. But in recent years, a ghost AI system has been used once more, in Tekken5: Dark resurrection. This time many things have been changed. Players do not need to train their ghosts in a training mode. They just play the game normally and the system will mechanically create their ghosts. This method makes fighting games more interesting because there will be many fighting styles for computer controlled opponents. Despite the fact that the ghost AI system is being acknowledged as the definitive AI for fighting games, the method for ghost AI creation remains undisclosed. In this paper, we propose a method for ghost AI creation using data obtained from game memory. Our method can be used in most fighting games. It also requires very small amounts of memory and therefore is suitable for console games.

1.2 Street Fighter Zero3 (SFZ3)

Street Fighter Zero3 is regarded as one of the best fighting games of all times. In a fighting game, a player must select one character from many characters, and fight one by one with an opponent character (another player or computer AI). A character can perform normal action such as move, crouch, jump, guard, punch or kick. There are also special attacks, such as firing bullets or executing a powerful flying punch. These special actions can be performed when a player presses a correct sequence of commands at the right time. A player must choose to perform actions in various situations based on the status of his character and opponent character. Getting into action with SFZ3 requires only a few minutes of tutorial. Nevertheless, the game has many ways to play a single character. For that reason, we have chosen SFZ3 as our game for experimenting with the ghost AI.

1.3 Testbed Environment

For the reliability of experimental results, game researchers may want to test their AI on real commercial game environments (Graepel et al 2004). But such environments are scarcely available. Results obtained from a researcher created game may not be convincing enough to warrant an actual use of discovered techniques in genuine games. Some researchers used mod of a commercial game (Spronck et al 2004), or a clone game (Ponsen et al 2005). Some developed test games on their own (Kendall and Kristian 2004) or used a testbed (Bailey and Katchabaw 2005). But none of those methods fit our experimental goal. (Thunputtarakul and Kotrajaras 2006) proposed a system to test AI modules in real commercial games without using any source code. They implemented a testbed from VisualboyAdvance (VBA), a Nintendo GameboyAdvance emulator. The testbed was called AI-TEM. An overview of AI-TEM is presented in figure 1 and its workflow diagram is presented in figure 2. By accessing the memory pool of the emulator, AI-TEM users are able to know states of the game at any particular moment. For fighting games, a state can consist of characters' positions, current animation frames, health points, etc. Users can insert their AI modules, in the form of C/C++

code or python script, into the testbed to control the game characters by providing controller signals. Our work uses AI-TEM as its testbed.

2. OUR APPROACH FOR CREATING GHOST AI

The main concept of our ghost AI creation is case based AI construction. We extracted a player character's reaction in various situations from battle log data created while playing, then produced situation-action pairs for the ghost of that character. Our experiment was made using SFZ3 training mode with character Ryu versus Ryu. AI-TEM was modified to suit our experiment. The ghost AI creation processes are displayed in figure 3. The following subsections describe each component in the process.

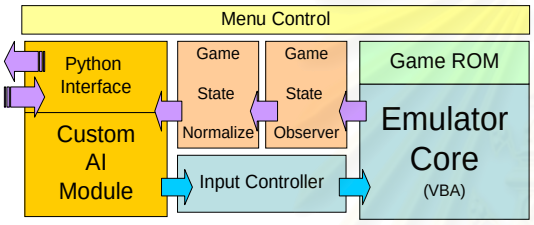


Figure 1: AI-TEM Testbed System Overview. The Light Blue Modules are VBA Original Modules.

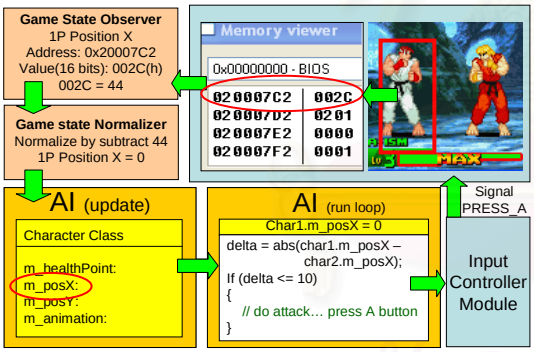


Figure 2: Workflow Diagram of AI-TEM System in SFZ3.

2.1 Obtaining Player Battle Log Data

First, while a player is playing, game states data need to be dumped from memory onto a battle log file. The data are used to identify each case in the case based AI system. The data consist of characters animation, characters positions in x and y axes, characters health points, characters bullet positions in x axes, damage that characters obtain in that frame, player character's facing direction and the corner status of characters. Recorded battle log data is in the following form:

Frame Data no: 00001
P1:Ani=002,X=120,Y=40,bullet=0,damage=0,HP=90
P2:Ani=002,X=240,Y=40,bullet=0,damage=0,HP=90
:
Frame Data no: 00720
P1:Ani=016,X=150,Y=40,bullet=0,damage=0,HP=30
P2:Ani=030,X=560,Y=40,bullet=0,damage=5,HP=20

These criteria can change depending on game or user. Creating the ghost AI while the game is running without creating the battle log file is possible if complete information about the game mechanic is known (such as short or shared animation frame, that will be described in section 2.3). For SFZ3 on AI-TEM, we did not have such information. Therefore we had to use the log file.

2.2 Animation Set Database

An animation set database is used for identifying whether a character animation frame belongs to an animation set. An example is illustrated in Figure 4. Ryu animation frame number 0 to 6 belong to animation set ID 0, which represents Ryu's standing animation, while frame number 707 to 713 belong to Ryu's medium punch action, set ID 15. Together with the battle log file, the animation sets are used to create situation-action pair cases. In our experiment, we manually defined this database. There are totally 912 frames for character Ryu. This seems daunting. However, it is relatively easy for a game company to do because any game development team usually has access to animation data.

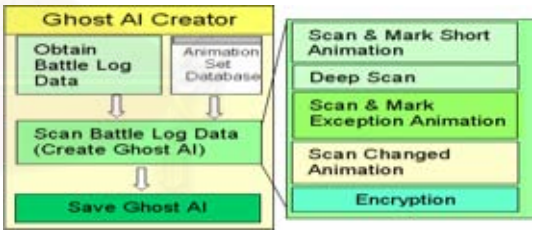


Figure 3: Ghost AI Creation Processes.

2.3 Scanning Battle Log Data

This process scans through every frame of a player's battle log data, trying to find which situation the player decided to begin his new animation set. For example, in situation *A* player1 is standing on the ground at position x=120 and player2 approaches player1 by jumping in the air at position x=150, both characters have full health bars and no bullets. Player1 decides to perform the special anti-air attack called Shoryuken punch. In short, the following situation-action pair will eventually be created:

if (Situation == A) do SHORYUKEN;

Now we look at this process in more detail. The process contains the following subtasks:

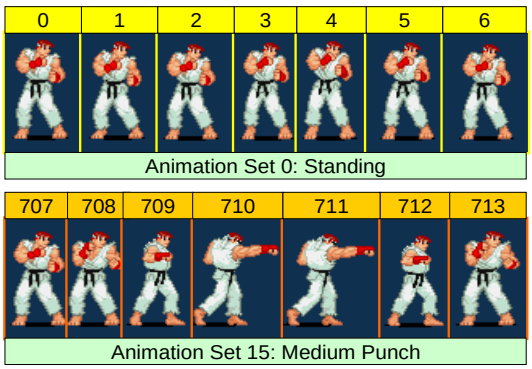


Figure 4: Example of Animation Set Database.

2.3.1 Finding Short Animation

Short animation means any animation that occurs for a very short period of time. It takes place mostly when a character is changing over from any standing animation loop to crouching animation loop. See an example animation time frame in figure 5. In figure 5, our character is standing then intends to do a crouching kick, but the crouching kick is not performed immediately. Before the crouching kick is carried out, a short period of moving forward and crouching animation is performed. This can happen due to the player not inputting the right command. For a crouching kick to be performed correctly without any prefix animation, the player needs to press down and kick at the same time on his control pad. In figure 5, the player presses down before kick and also unintentionally presses forward at the same time as down. Therefore extra animation is triggered. Nevertheless, the crouching kick is eventually performed and the prefix animation is so fast a human eye cannot see. We cannot avoid such minor mistakes made by players.

In our ghost AI model, detected animation frames tell us about a player's intention. Therefore, having the short animation taking place before the intended animation can misinform us. We must either identify a player's intention from the overall animation or get rid of the short animation before processing. In our experiment, we chose to do the latter.

All battle log data need to be scanned to find which animation set appears unusually brief, then that set is marked. Marked animation will not be considered when creating the AI. For a set of animation to be considered short, it depends on the set. In our experiment with SFZ3, short animation was no longer than 6 frames for most of the animation sets. The only exception was the crouching animation, of which short animation was no longer than 14 frames because changing from standing to crouching already took 8 frames.

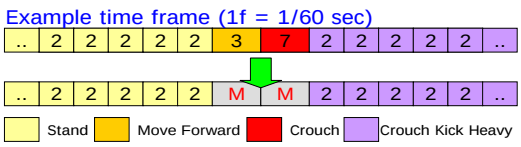


Figure 5: Short Animation Marking.

2.3.2 Deep Scanning

Some animation frames are shared between many animation sets. In such case, scanning ahead becomes necessary in order to identify the correct animation set. For example, jump straight, jump forward and jump backward begin with the same animation frames at the beginning. With the first frame obtained, we can only conclude that the character is doing an anonymous jump. With further scanning, we then know which jump the player intends to do and can go back to change from an anonymous jump to a specific jump. This step can be omitted if the controller signal can be completely analyzed. But this is not always the case.

2.3.3 Exception Animation Sets

Some animation sets should be omitted from our case base because they do not take place under players' control. Obvious examples are various damage animation sets. They occur as the results of opponent attacks. This type of animation that appears in the battle log data will be marked here.

2.3.4 Scanning Changed Animation

This step is the core of our ghost AI creation. After matching all animation frames to their corresponding animation sets and marking useless animation, it is time to scan the battle log data once more to find the situation that causes the player character to change its animation. Such situation and the changed animation set that it causes will be paired to create a situation-action case.

An example is shown in figure 6, where a player executes a crouching heavy kick. In 7^{th}-8^{th} frame, our character changes its animation set from standing to moving forward. But moving forward lasts only 2 frames so it is a short animation. It is marked useless and the next animation to consider will be crouching. However, this crouching is also a short animation and therefore marked useless (a proper crouching must last 14 frames or more). As a result, the next animation (crouching heavy kick) will be taken into account. The crouching heavy kick does not fit the useless animation category, so it is regarded as the changed animation set. Therefore the (situation at 7^{th} frame, crouching heavy kick) is added to the case based AI.

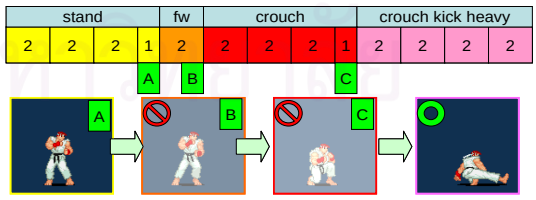


Figure 6: Scan Animation Change.

2.3.5 Situation Encryption

If the game needs to compare ten or more criteria (animation, position, bullet, etc.) to judge whether the current situation in the game is the same as any existing condition in our situation-action database, it will be a waste of processing power. Any game situation should be defined in simple form for easy

comparison and discovery. We propose a method to encrypt a fighting game state situation into a 32-bit integer (capable of holding 4,294,967,296 values). The bits can be divided into small 1-8bits sections as shown in table 1.

Table 1: Detail of Situation Encryption.

Bit no.	nBits	nValues	Meanings
1-8	8	256	Player character animation set ID. [As said in section 2.2]
9-12	4	16	Delta position in X axis. [Divide distance into 9 ranges]
13-14	2	4	Delta position in Y axis. [Divide distance into 4 ranges]
15-18	4	16	Enemy character state. [Group into 6 stages: Normal, Attacking, Blocking, Dizzy, Damaged, Invulnerable]
19	1	2	Character's bullet state. [Have or not]
20-22	3	8	Delta position in X axis between player character and enemy's bullet. [Divide distance into 8 ranges]
23-29	7	128	Damage value that enemy got in that frame (use for combination attack decision).
30	1	2	Player character side. [Left or Right]
31	1	2	Is player at corner. [Yes or No]
32	1	2	Is enemy at corner. [Yes or No]

There are ten criteria that we use for identifying the game state (ten rows in table1). Bit 1 to 8 store the animation set ID of the action that the player character performs in that frame situation. The animation set value comes from the animation set database described in section 2.2. When the player character is in any normal standing frame (frame id 0 to 6), the value in the first 8 bit will be 0. As a brief example, the situation that two characters are standing at the beginning of a battle will be encrypted as "1,792". Every criterion for this particular scene will have 0 as its value, except the delta position in the x axis, which will have its value equal to 7 due to the distance between characters at the beginning of battle (150 units). Details of this encryption can be changed to match other games or other platforms.

2.4 Creating Ghost AI File

When the scanning process discovers that animation set change takes place, the situation in the frame before that discovered frame is encrypted into 32-bit data (integer) by the process in section 2.3.5. Its corresponding case base can now be created by combining the situation ID (32-bit situation encryption result) with its response action list. An example of our case base is shown below.

```
SituationID: 0000000000
TotalRatio: 03 TotalNextAni: 02
   NextAni: Punch-Light-Close Ratio 2
   NextAni: Kick-Heavy-Close  Ratio 1
:
SituationID: 2684356352
TotalRatio: 01 TotalNextAni: 01
   NextAni: Hadouken          Ratio 1
```

Each case will have situationID for representing each game situation. TotalRatio is the number of incidents the player encounters that situation. TotalNextAni is the number of different animation sets that the player performs when facing that situation. It is followed by the list of those animation sets and the number of times the player performs each animation set. The ratio of each animation set and the total number of sets will be used in response selection while the ghost AI is actually running.

From above example cases, the player encountered situation 0 three times and decided to do a light-punch twice and a heavy kick once. These cases should be kept in a data structure that is convenient and fast to insert and find because we need to know whether the situation is a new situation that player never encounters (so we can add new data from scratch), or an old situation that updates the response action list. In our experiment we chose *map* of standard template library (STL), which is a balanced binary search tree, to store the cases. The tree was written into our ghost AI file. Using file allows for future modifications of the knowledge base.

3. USING GHOST AI

To run the ghost AI, first, the game needs to load any required database such as the animation set database. Then it needs to load the ghost AI case base into some data structure that allows quick finding and matching. A new case is never inserted while running the ghost AI.

From the data in section 2.4, the game first loads all cases into the *map*. When the situationID 0 takes place, the case that has situationID 0 in the *map* is searched. It will be found and returned. That case has a total ratio of 3 and has two next animations (light-punch with ratio 2 and heavy-kick with ratio 1). The game then randomly selects one of these actions corresponding to the ratio value and sends a command to perform that action.

When a ghost AI is running, if used with a suitable data structure such as a balanced binary search tree, searching any case is guaranteed to use *O(log n)* amount of time (when n is the number of cases). A ghost AI with one thousand cases should find a result in the tenth search. Each case based data uses approximately 40 bytes of memory. Therefore, a thousand-case ghost requires only 40KB of memory. In short, creating and running our ghost AI does not slow down the game or consume much memory at all.

4. VERIFYING METHOD AND RESULTS

The best way to evaluate a ghost AI's similarity to its creator should be: letting its creator verify with his own eyes. But sometimes, people can make incorrect judgments, forgetting even their own playing styles. Therefore we designed a measurable method for evaluating the ghost AI.

4.1 The Experiment

We appointed thirty two SFZ3 players and let them play the game for approximately 2 to 10 minutes. We recorded their game events in VMV file format

(recording the beginning game state and controller sequence) and created their ghost AI. After that, we let the player semi-play the game again two more times, while their ghost AI was playing and while their own playing movie was playing. The term semi-play means players see their ghosts or their own movies playing while pressing the controller, imagining that they are controlling their characters in that situation. We wanted to compare the controller signals of the ghosts with the players' signals. We also wanted to compare the players against their video.

Controller signals should not be compared frame-by-frame, because only 1 frame delay (1/60 second) will cause the rest of the matching process to fail.

Therefore the controller signals need to be normalized before any comparison can be done. In our approach, we normalized the signals by splitting the signals into parts. Each part contained approximately 5 to 15 signals. After that, we combined all the same signals that appear continuous into one signal (when a player presses one button normally, it takes approximately 6-8 frame, so it gives out 6-8 continuous signals). For example, if the signals are as follows:

Raw ghost signal:
16,16,16,32,32,32,64,64,64,64,128,128,256,256,256

Raw player signal:
16,16,16,16,16,16,32,32,32,32,64,64,128,128,128

After normalized they will be like these.

Normalized ghost signal: 16,32,64,128,256
Normalized player signal:16,32,64,128,0

It can be seen from the example that if we compare raw signals directly the result will be 3 of 15 signals match. The matching result is not correct because identical commands that are pressed for slightly different amount of time will be regarded as being different. However, if we compare the two signals after our normalization, the match is 4 out of 5.

We had two methods for slicing controller signals. In the first method, we sliced every 15 frames. We had tried several values and this value gave the best result. Too small values made the normalization meaningless, while too large values put more than one signals in the same frame, making the result unreliable. In the second method, we performed the slicing every time the signal of the ghost AI or the player movie changed values, based on the assumption that matching signals should occur in the same frame time period as its counterpart. With the second method, we always had one signal per slicing window. We also gave score if there were some similarity between controller signals. For example, if the ghost AI was pressing down-forward and the player was pressing forward only, we gave similarity score of 0.5 (50%) to the ghost AI.

4.2 Result

The result of our experiment is illustrated in figure 7 and table 2. *Player_Player%* is the similarity (in percentage) between each player's own movie and his actual control when re-playing the situation in the movie. *Ghost AI_Player%* compares each ghost AI with its corresponding player's re-play. *Delta*% is the difference between the two comparisons. Table 2 displays the overall statistical summary. *Delta A* and *Delta B* indicate delta percentage points between the result of [player's own movie vs. player] and [ghost AI vs. player]. *Score* is the score that the players evaluate their ghosts' similarity to their fighting styles based on their feelings.

Both signal slicing methods gave similar results. But the second method gave less matching percentage points. This is likely because the number of signals after the normalization was less than in the first method. With many long signals in play, such as idle signals, the first method scored better because it did not compress long signals into one signal. For the first method, the average similarity between ghosts and the players is 26.33%. This may seem small. But if we look at the comparison between the players and their own movies, the similarity is only 34.96%. The ghosts' performances were therefore very close to players' performances (75.31% close). Some ghosts even scored better than their corresponding players.

An average satisfactory score given by players is 72.2%, which is good. The players thought that the ghosts sometimes performed more attacks and fewer defenses than their creators. Some players could not distinguish between their ghosts and their own movies while semi-playing. (We did not tell the players which engine was really controlling the characters).

5. CONCLUSION AND FUTURE WORK

We propose a method and concept for creating ghost AI without having to know game source code. We used AI-TEM, an emulator based testbed to provide a commercial game testing environment. Our concept for ghost AI creation is general for all fighting games. Using SFZ3, which is a very well respected commercial game, with its basic systems being used in almost all fighting games, our findings are guaranteed to be applicable to other commercial games.

Our method produces good results. Ghost AIs display their creators' playing styles even when the training time is short. The two-minute average training time we used is equal to a match time in an average fighting game.

For future experiment we are interested in exploring techniques for ghost AI in team based fighting games, where characters can cooperate. Another interesting future work is developing AI that can adapt and counter an opponent's play style.

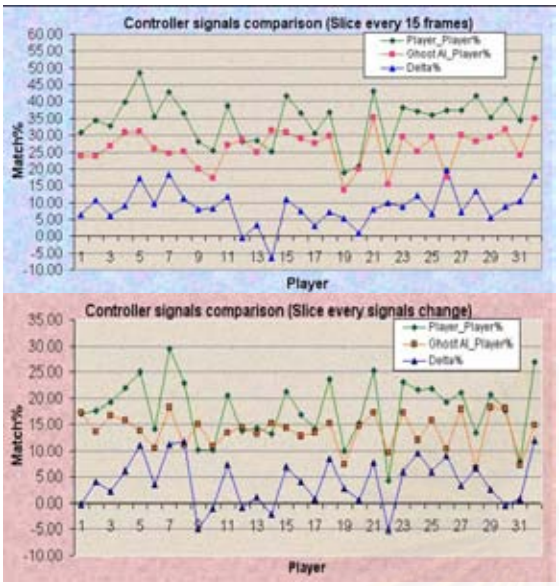


Figure 7: Players vs. Movies and Players vs. Ghosts.

Table 2: Summary Result of Experiment. A: Slice Every 15 Frames, B: Slice Every Time When Signal Change.

Summary	Player_ Player A	Ghost AI_ Player A	Delta A	Player_ Player B	Ghost AI_ Player B	Delta B	Score
Min	18.81	13.6	-6.45	4.37	6.54	-5.18	44
Max	52.84	35.18	19.82	29.51	18.27	12.03	90
Average	34.96	26.33	8.62	17.93	13.81	4.12	72.2

REFERENCES

Bailey, C. and M. J. Katchabaw. 2005. An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games. *Proceedings of the 2005 GameOn North America Conference.* Montreal, Canada.

Graepel Thore, Ralf Herbrich, Julian Gold. 2004. Learning to fight. *International Conference on Computer Games: Artificial Intelligence, Design and Education*

Kendall Graham, Kristian Spoerer. 2004. Scripting the Game of Lemmings with a Genetic Algorithm. *Proceedings of the 2004 Congress on Evolutionary Computation*, IEEE Press, Piscataway, NJ, pp. 117-124

Ponsen Marc J.V., Hector Munoz-Avila, Pieter Spronck, and David W. Aha. 2005. Automatically Acquiring Domain Knowledge For Adaptive Game AI Using Evolutionary Learning. *Proceedings The Twentieth National Conference on Artificial Intelligence.*

Spronck Pieter, Ida Sprinkhuizen-Juyper, Eric Postma. 2004. Online Adaptation Of Game Opponent AI With Dynamic Scripting. *International Journal of Intelligent Games and Simulation*, Vol. 3, No. 1, University of Wolverhampton and EUROSIS, pp. 45–53.

Thunputtarakul Worapoj and Kotrajaras Vishnu. 2006. AI-TEM: Testing Artificial Intelligence in Commercial Game using Emulator. *8th CGAMES International Conference on Computer Games: AI, Animation, Mobile, Educational & Serious Games.* Louisville Kentucky, USA.

ภาคผนวก ข ภาพ อนิเมชั่นเฟรมของริวทั้งหมด

ภาพอนิเมชั่นเฟรมในภาคผนวกนี้ได้มาจากวิธีการเก็บสองวิธีคือ

1 ใช้เอไอโมดูลของไอเทมสั่งกดท่าต่างๆ ทุกท่าโดยให้หยุดพักระหว่างท่าเป็นเวลา 20 เฟรม จากนั้นจึงเก็บหมายเลขของท่าที่ปรากฏออกมาทางแอดเดรสที่บอกท่าทางของตัวละครพร้อมทั้งสั่งจับภาพทีละเฟรมไปพร้อมกันด้วย ตัวอย่างเช่นค่าและภาพที่ปรากฏออกมาเป็นท่าแรกก็คือกลุ่มเฟรมของท่าที่สั่งกดท่าแรก กลุ่มค่าที่ปรากฏออกมาลำดับที่สองหลังจากยืนนิ่งไป 20 เฟรมแล้วก็จะเป็นกลุ่มท่าที่สองที่สั่งกด เป็นเช่นนี้ไปเรื่อยๆ วิธีการนี้จะทำให้รู้ได้ว่าค่าในแอดเดรสค่าไหนคือเฟรมท่าไหน แต่วิธีนี้มีข้อเสียคือท่าบางท่าที่ไม่ได้เกิดจากการสั่งกดจะไม่สามารถรู้ได้ เช่นท่าบาดเจ็บและท่าการ์ดซึ่งต้องเกิดจากการที่อีกฝ่ายเข้ามาเกี่ยว จึงต้องใช้วิธีที่สองเข้ามาช่วย

2 สั่งเขียนค่าลงในแอดเดรสที่เก็บค่าท่าทางของตัวละครโดยตรง วิธีนี้จะทำให้บังคับตัวละครให้ทำท่าต่างๆ ออกมาได้ครบทุกท่า แล้วจึงบันทึกค่าที่เขียนลงไปกับภาพที่ปรากฏออกมาไว้คู่กัน เช่นนี้จะทำให้รู้ได้ว่าค่าไหนหมายถึงภาพอนิเมชั่นเฟรมไหน วิธีนี้บังคับให้ออกได้ครบทุกท่าไม่ว่าท่าบาดเจ็บท่าการ์ดหรือท่าพิเศษตามเงื่อนไขต่างๆ ด้วย แต่ข้อเสียคือต้องพิจารณาเองว่าแต่ละเฟรมที่เกิดจากการสั่งเขียนค่าตามลำดับลงไปนั้นคือท่าอะไร โดยการกลับไปใช้ข้อมูลจากวิธีแรกประกอบ ด้วยทั้งสองวิธีนี้ประกอบกันเลยทำให้เก็บอนิเมชั่นได้แสดงในภาคผนวก ข

000000.png 000001.png 000002.png 000003.png 000004.png 000005.png 000006.png 000007.png 000008.png

000009.png 000010.png 000011.png 000012.png 000013.png 000014.png 000015.png 000016.png 000017.png

000018.png 000019.png 000020.png 000021.png 000022.png 000023.png 000024.png 000025.png 000026.png

000027.png 000028.png 000029.png 000030.png 000031.png 000032.png 000033.png 000034.png 000035.png

000036.png 000037.png 000038.png 000039.png 000040.png 000041.png 000042.png 000043.png 000044.png

000045.png 000046.png 000047.png 000048.png 000049.png 000050.png 000051.png 000052.png 000053.png

000054.png 000055.png 000056.png 000057.png 000058.png 000059.png 000060.png 000061.png 000062.png

000063.png 000064.png 000065.png 000066.png 000067.png 000068.png 000069.png 000070.png 000071.png

000072.png 000073.png 000074.png 000075.png 000076.png 000077.png 000078.png 000079.png 000080.png

000081.png 000082.png 000083.png 000084.png 000085.png 000086.png 000087.png 000088.png 000089.png

000090.png 000091.png 000092.png 000093.png 000094.png 000095.png 000096.png 000097.png 000098.png

000099.png 000100.png 000101.png 000102.png 000103.png 000104.png 000105.png 000106.png 000107.png

000108.png 000109.png 000110.png 000111.png 000112.png 000113.png 000114.png 000115.png 000116.png

000117.png 000118.png 000119.png 000120.png 000121.png 000122.png 000123.png 000124.png 000125.png

000126.png 000127.png 000128.png 000129.png 000130.png 000131.png 000132.png 000133.png 000134.png

000135.png 000136.png 000137.png 000138.png 000139.png 000140.png 000141.png 000142.png 000143.png

000144.png 000145.png 000146.png 000147.png 000148.png 000149.png 000150.png 000151.png 000152.png

000153.png 000154.png 000155.png 000156.png 000157.png 000158.png 000159.png 000160.png 000161.png

000162.png 000163.png 000164.png 000165.png 000166.png 000167.png 000168.png 000169.png 000170.png

000171.png 000172.png 000173.png 000174.png 000175.png 000176.png 000177.png 000178.png 000179.png

000180.png 000181.png 000182.png 000183.png 000184.png 000185.png 000186.png 000187.png 000188.png

000189.png 000190.png 000191.png 000192.png 000193.png 000194.png 000195.png 000196.png 000197.png

000198.png 000199.png 000200.png 000201.png 000202.png 000203.png 000204.png 000205.png 000206.png

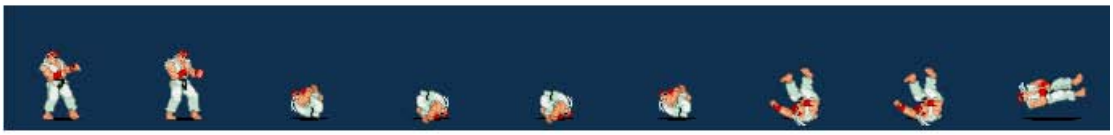
000207.png 000208.png 000209.png 000210.png 000211.png 000212.png 000213.png 000214.png 000215.png

000216.png 000217.png 000218.png 000219.png 000220.png 000221.png 000222.png 000223.png 000224.png

000225.png 000226.png 000227.png 000228.png 000229.png 000230.png 000231.png 000232.png 000233.png

000234.png 000235.png 000236.png 000237.png 000238.png 000239.png 000240.png 000241.png 000242.png

000243.png 000244.png 000245.png 000246.png 000247.png 000248.png 000249.png 000250.png 000251.png



000252.png 000253.png 000254.png 000255.png 000256.png 000257.png 000258.png 000259.png 000260.png

000261.png 000262.png 000263.png 000264.png 000265.png 000266.png 000267.png 000268.png 000269.png

000270.png 000271.png 000272.png 000273.png 000274.png 000275.png 000276.png 000277.png 000278.png

000279.png 000280.png 000281.png 000282.png 000283.png 000284.png 000285.png 000286.png 000287.png

000288.png 000289.png 000290.png 000291.png 000292.png 000293.png 000294.png 000295.png 000296.png

000297.png 000298.png 000299.png 000300.png 000301.png 000302.png 000303.png 000304.png 000305.png

000306.png 000307.png 000308.png 000309.png 000310.png 000311.png 000312.png 000313.png 000314.png

000315.png 000316.png 000317.png 000318.png 000319.png 000320.png 000321.png 000322.png 000323.png

000324.png 000325.png 000326.png 000327.png 000328.png 000329.png 000330.png 000331.png 000332.png

000333.png 000334.png 000335.png 000336.png 000337.png 000338.png 000339.png 000340.png 000341.png

000342.png 000343.png 000344.png 000345.png 000346.png 000347.png 000348.png 000349.png 000350.png

000351.png 000352.png 000353.png 000354.png 000355.png 000356.png 000357.png 000358.png 000359.png

000360.png 000361.png 000362.png 000363.png 000364.png 000365.png 000366.png 000367.png 000368.png

000369.png 000370.png 000371.png 000372.png 000373.png 000374.png 000375.png 000376.png 000377.png

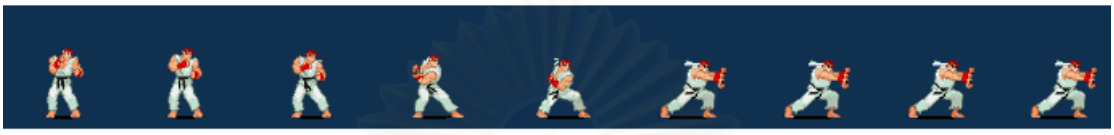
000378.png 000379.png 000380.png 000381.png 000382.png 000383.png 000384.png 000385.png 000386.png

000387.png 000388.png 000389.png 000390.png 000391.png 000392.png 000393.png 000394.png 000395.png

000396.png 000397.png 000398.png 000399.png 000400.png 000401.png 000402.png 000403.png 000404.png

000405.png 000406.png 000407.png 000408.png 000409.png 000410.png 000411.png 000412.png 000413.png

000414.png 000415.png 000416.png 000417.png 000418.png 000419.png 000420.png 000421.png 000422.png

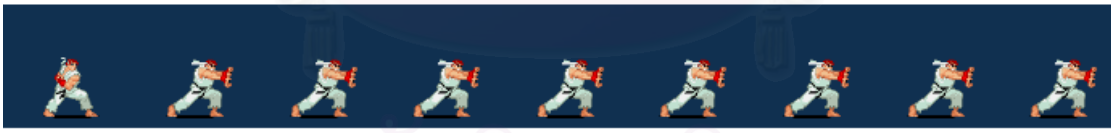


000423.png 000424.png 000425.png 000426.png 000427.png 000428.png 000429.png 000430.png 000431.png

000432.png 000433.png 000434.png 000435.png 000436.png 000437.png 000438.png 000439.png 000440.png

000441.png 000442.png 000443.png 000444.png 000445.png 000446.png 000447.png 000448.png 000449.png

000450.png 000451.png 000452.png 000453.png 000454.png 000455.png 000456.png 000457.png 000458.png



000459.png 000460.png 000461.png 000462.png 000463.png 000464.png 000465.png 000466.png 000467.png

000468.png 000469.png 000470.png 000471.png 000472.png 000473.png 000474.png 000475.png 000476.png

000477.png 000478.png 000479.png 000480.png 000481.png 000482.png 000483.png 000484.png 000485.png

000486.png 000487.png 000488.png 000489.png 000490.png 000491.png 000492.png 000493.png 000494.png

000495.png 000496.png 000497.png 000498.png 000499.png 000500.png 000501.png 000502.png 000503.png

000504.png 000505.png 000506.png 000507.png 000508.png 000509.png 000510.png 000511.png 000512.png

000513.png 000514.png 000515.png 000516.png 000517.png 000518.png 000519.png 000520.png 000521.png

000522.png 000523.png 000524.png 000525.png 000526.png 000527.png 000528.png 000529.png 000530.png

000531.png 000532.png 000533.png 000534.png 000535.png 000536.png 000537.png 000538.png 000539.png

000540.png 000541.png 000542.png 000543.png 000544.png 000545.png 000546.png 000547.png 000548.png

000549.png 000550.png 000551.png 000552.png 000553.png 000554.png 000555.png 000556.png 000557.png

000558.png 000559.png 000560.png 000561.png 000562.png 000563.png 000564.png 000565.png 000566.png

000567.png 000568.png 000569.png 000570.png 000571.png 000572.png 000573.png 000574.png 000575.png

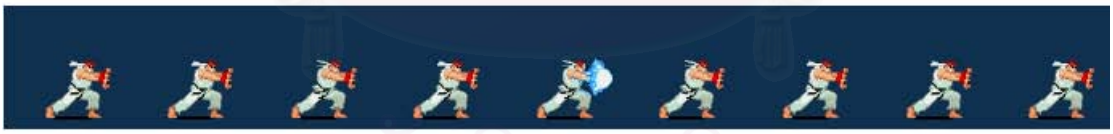
000576.png 000577.png 000578.png 000579.png 000580.png 000581.png 000582.png 000583.png 000584.png

000585.png 000586.png 000587.png 000588.png 000589.png 000590.png 000591.png 000592.png 000593.png

000594.png 000595.png 000596.png 000597.png 000598.png 000599.png 000600.png 000601.png 000602.png

000603.png 000604.png 000605.png 000606.png 000607.png 000608.png 000609.png 000610.png 000611.png

000612.png 000613.png 000614.png 000615.png 000616.png 000617.png 000618.png 000619.png 000620.png



000621.png 000622.png 000623.png 000624.png 000625.png 000626.png 000627.png 000628.png 000629.png

000630.png 000631.png 000632.png 000633.png 000634.png 000635.png 000636.png 000637.png 000638.png

000639.png 000640.png 000641.png 000642.png 000643.png 000644.png 000645.png 000646.png 000647.png

000648.png 000649.png 000650.png 000651.png 000652.png 000653.png 000654.png 000655.png 000656.png

000657.png 000658.png 000659.png 000660.png 000661.png 000662.png 000663.png 000664.png 000665.png

000666.png 000667.png 000668.png 000669.png 000670.png 000671.png 000672.png 000673.png 000674.png

000675.png 000676.png 000677.png 000678.png 000679.png 000680.png 000681.png 000682.png 000683.png

000684.png 000685.png 000686.png 000687.png 000688.png 000689.png 000690.png 000691.png 000692.png

000693.png 000694.png 000695.png 000696.png 000697.png 000698.png 000699.png 000700.png 000701.png

000702.png 000703.png 000704.png 000705.png 000706.png 000707.png 000708.png 000709.png 000710.png

000711.png 000712.png 000713.png 000714.png 000715.png 000716.png 000717.png 000718.png 000719.png

000720.png 000721.png 000722.png 000723.png 000724.png 000725.png 000726.png 000727.png 000728.png

000729.png 000730.png 000731.png 000732.png 000733.png 000734.png 000735.png 000736.png 000737.png

000738.png 000739.png 000740.png 000741.png 000742.png 000743.png 000744.png 000745.png 000746.png

000747.png 000748.png 000749.png 000750.png 000751.png 000752.png 000753.png 000754.png 000755.png

000756.png 000757.png 000758.png 000759.png 000760.png 000761.png 000762.png 000763.png 000764.png

000765.png 000766.png 000767.png 000768.png 000769.png 000770.png 000771.png 000772.png 000773.png

000774.png 000775.png 000776.png 000777.png 000778.png 000779.png 000780.png 000781.png 000782.png

000783.png 000784.png 000785.png 000786.png 000787.png 000788.png 000789.png 000790.png 000791.png

000792.png 000793.png 000794.png 000795.png 000796.png 000797.png 000798.png 000799.png 000800.png

000801.png 000802.png 000803.png 000804.png 000805.png 000806.png 000807.png 000808.png 000809.png

000810.png 000811.png 000812.png 000813.png 000814.png 000815.png 000816.png 000817.png 000818.png

000819.png 000820.png 000821.png 000822.png 000823.png 000824.png 000825.png 000826.png 000827.png

000828.png 000829.png 000830.png 000831.png 000832.png 000833.png 000834.png 000835.png 000836.png

000837.png 000838.png 000839.png 000840.png 000841.png 000842.png 000843.png 000844.png 000845.png

000846.png 000847.png 000848.png 000849.png 000850.png 000851.png 000852.png 000853.png 000854.png

000855.png 000856.png 000857.png 000858.png 000859.png 000860.png 000861.png 000862.png 000863.png

000864.png 000865.png 000866.png 000867.png 000868.png 000869.png 000870.png 000871.png 000872.png

000873.png 000874.png 000875.png 000876.png 000877.png 000878.png 000879.png 000880.png 000881.png

000882.png 000883.png 000884.png 000885.png 000886.png 000887.png 000888.png 000889.png 000890.png

000891.png 000892.png 000893.png 000894.png 000895.png 000896.png 000897.png 000898.png 000899.png

000900.png 000901.png 000902.png 000903.png 000904.png 000905.png 000906.png 000907.png 000908.png

000909.png 000910.png 000911.png 000912.png 000913.png 000914.png 000915.png 000916.png 000917.png

000918.png 000919.png 000920.png 000921.png 000922.png 000923.png 000924.png 000925.png 000926.png

ภาคผนวก ค การจัดแบ่งกลุ่มอนิเมชั่นของเฟรมต่างๆ

หมายเลขกลุ่มอนิเมชั่นมีความหมายดังนี้

```
#define  A_STAND                  0
#define  A_CROUNCH                1
#define  A_MOVE_FORWARD           2
#define  A_MOVE_BACKWARD          3

#define  A_JUMPING                4
#define  A_JUMP_STRAIGHT          5
#define  A_JUMP_FORWARD           6
#define  A_JUMP_BACKWARD          7

#define  A_GUARD                  8
#define  A_CROUNCH_GUARD          9
#define  A_AIR_GUARD             10

#define  A_DAMAGE                11
#define  A_CROUNCH_DAMAGE        12
#define  A_RECOVER_TO_STAND      13
#define  A_ELECTRIC_SHOCKED      14

#define  A_BEGIN_THROW           15
#define  A_THROW_DRAW            16
#define  A_CROUNCH_THROW_DRAW    17
#define  A_AIR_THROW_DRAW        18
#define  A_MISS_THROW            19
#define  A_THROW                 20

#define  A_ROLL                  21
#define  A_BEFORE_FIGHT          22
#define  A_VICTORY_POSE          23
#define  A_ZERO_COUNTER          24
#define  A_CHALLENGE             25

// Punch (LP, CLP, MP, CMP, HP, CHP, FMP)
#define  A_PUNCH_CLP  30
#define  A_PUNCH_LP   31
#define  A_PUNCH_CMP  32
#define  A_PUNCH_MP   33
#define  A_PUNCH_CHP  34
#define  A_PUNCH_HP   35
#define  A_PUNCH_FMP  36

// Kick (LK, MK, HK, CHK, FMK)
#define  A_KICK_LK    40
#define  A_KICK_MK    41
#define  A_KICK_CHK   42
#define  A_KICK_HK    43
#define  A_KICK_FMK   44

// Crounch Punch (LP, MP, HP)
#define  A_CROUNCH_PUNCH_LP  50
#define  A_CROUNCH_PUNCH_MP  51
#define  A_CROUNCH_PUNCH_HP  52


// Crounch Kick (LK, MK, HK)
#define  A_CROUNCH_KICK_LK   60
#define  A_CROUNCH_KICK_MK   61
```

```
#define  A_CROUNCH_KICK_HK   62

// Jump Punch (LP, MP, HP)
#define  A_JUMP_PUNCH_LP    70
#define  A_JUMP_PUNCH_MP    71
#define  A_JUMP_PUNCH_HP    72

// Jump Kick (LK, MK, HK)
#define  A_JUMP_KICK_LK     80
#define  A_JUMP_KICK_MK     81
#define  A_JUMP_KICK_HK     82

// Jump side punch (LP, MP, HP)
#define  A_JUMP_SIDE_PUNCH_LP  90
#define  A_JUMP_SIDE_PUNCH_MP  91
#define  A_JUMP_SIDE_PUNCH_HP  92

// Jump side Kick (LK, MK, HK)
#define  A_JUMP_SIDE_KICK_LK   100
#define  A_JUMP_SIDE_KICK_MK   101
#define  A_JUMP_SIDE_KICK_HK   102

#define  A_HADOKEN         110
#define  A_SHORYUKEN       111
#define  A_TORNADO_KICK    112

#define  A_SHINKU_TORNADO_KICK 120
#define  A_SHINKU_HADOKEN      121
#define  A_METSU_SHORYUKEN     122

#define  A_JUMP_FORWARD_STEP2  130
#define  A_JUMP_FORWARD_STEP3  131
#define  A_AIR_RECOVERY        132
#define  A_AUTO_ROLL           133

#define  A_HADOKEN_L           134
#define  A_HADOKEN_M_FL        135
#define  A_HADOKEN_H_FM        136
#define  A_HADOKEN_FH          137
#define  A_HADOKEN_FAKE        138

#define  A_SHINKU_HADOKEN_L    139
#define  A_SHINKU_HADOKEN_M    140
#define  A_SHINKU_HADOKEN_H    141

#define  A_SHORYUKEN_L         142
#define  A_SHORYUKEN_M         143
#define  A_SHORYUKEN_H         144

#define  A_T_KICK_L            151
#define  A_T_KICK_M            152
#define  A_T_KICK_H            153

#define  A_T_KICK_AIR_L        154
#define  A_T_KICK_AIR_M        155
#define  A_T_KICK_AIR_H        156

#define  A_SHINKU_T_KICK_L     157
#define  A_SHINKU_T_KICK_M     158
#define  A_SHINKU_T_KICK_H     159
```

```
#define  A_MOVE_FORWARD_10     161
#define  A_MOVE_FORWARD_15     162
#define  A_MOVE_FORWARD_20     163
#define  A_MOVE_FORWARD_30     164
#define  A_MOVE_FORWARD_40     165
#define  A_MOVE_FORWARD_50     166
#define  A_MOVE_FORWARD_60     167
#define  A_MOVE_FORWARD_70     168
#define  A_MOVE_FORWARD_80     169
#define  A_MOVE_BACKWARD_10    170
#define  A_MOVE_BACKWARD_15    171
#define  A_MOVE_BACKWARD_20    172
#define  A_MOVE_BACKWARD_30    173
#define  A_MOVE_BACKWARD_40    174
#define  A_MOVE_BACKWARD_50    175
#define  A_MOVE_BACKWARD_60    176
#define  A_MOVE_BACKWARD_70    177
#define  A_MOVE_BACKWARD_80    178
#define  A_AIR_RECOVERY_F      180
#define  A_AIR_RECOVERY_N      181
#define  A_AIR_RECOVERY_B      182
#define  A_JUMP_BACKWARD_STEP2 183
#define  A_JUMP_BACKWARD_STEP3 184
#define  A_JUMP_STRAIGHT_STEP2 185
#define  A_JUMP_STRAIGHT_STEP3 186
#define  A_PREJUMP             187
#define  A_STAND_FLIP          188
#define  A_CROUCH_FLIP         189
#define  A_FAKE_STATE          199
#define  A_UNKNOWN             150
```

รายละเอียดของแต่ละเฟรมว่าหมายถึงท่ากลุ่มไหนแสดงในหน้าต่อไป โดยแต่ละคอลัมน์มีความหมายดังนี้ AniNo หมายถึงเบอร์ของแต่ละเฟรม ซึ่งจะตรงกับภาพในภาคผนวก ข. เช่นเฟรมที่ 0 ก็คือรูปยืนรูปแรก AniNo*12 หมายถึงค่า AniNo คูณด้วยสิบสอง ค่านี้ไม่ใช้ประโยชน์อะไร AniSet หมายถึงภาพในเฟรมนั้นถูกจัดให้อยู่ในกลุ่มท่าอนิเมชั่นกลุ่มไหน โดยค่าในคอลัมน์นี้จะอิงกับค่าในภาคผนวก ค. ข้างต้น เช่น AniSet มีค่าเป็น 1 ก็หมายถึง A_CROUNCH หรือท่าก้มเป็นต้น (ส่วนค่า -1 หมายถึงมีค่าเดียวกันกับบรรทัดข้างบน) AniType1 หมายถึงคุณสมบัติของเฟรม ดังนี้ ค่า 0 คือเฟรมใดๆ ที่ไม่มีคุณสมบัติของ 1-5 ค่า 1 คือเฟรมที่กำลังบาดเจ็บ ค่า 2 คือเฟรมที่กำลังลุกจากล้มและเป็นอมตะ ค่า 3 คือเฟรมที่อยู่ในท่ามึน ค่า 4 ไม่ใช้ และ ค่า 5 คือเฟรมที่เป็นท่าโจมตี Animation Name คือชื่อท่า ส่วน Description และ Attack Range เป็นคำบรรยายเฟรม

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
0	0	0	0	Stand		
1	12	-1	-1			
2	24	-1	-1			
3	36	-1	-1			
4	48	-1	-1			
5	60	-1	-1			
6	72	1	-1	Crounch		
7	84	2	-1	Move forward		
8	96	-1	-1			
9	108	-1	-1			
10	120	-1	-1			
11	132	-1	-1			
12	144	-1	-1			
13	156	3	-1	Move backward		
14	168	-1	-1			
15	180	-1	-1			
16	192	-1	-1			
17	204	-1	-1			
18	216	-1	-1			
19	228	1	-1	Crounch	กำลังจะลุกยืน	
20	240	-1	-1		Mid-crounch	
21	252	-1	-1		Stand after crounch	
22	264	-1	-1		กำลังก้ม	
23	276	-1	-1		Stand before crounch	
24	288	-1	-1		Mid-crounch	
25	300	-1	-1	Crounch		
26	312	-1	-1	Crounch		
27	324	188	-1	Stand Flip	กลับตัวพลิกข้างตอนยืน	
28	336	-1	-1			
29	348	-1	-1			
30	360	-1	-1			
31	372	189	-1	Crounch Flip	กลับตัวพลิกข้างตอนนั่ง	
32	384	-1	-1			
33	396	-1	-1			
34	408	-1	-1			
35	420	187	-1	Stand	ยืน ก่อนเตรียมโดด	
36	432	-1	-1			
37	444	0	-1		ยืน หลังโดด	
38	456	-1	-1			
39	468	5	5	Jump straight	โดดกำลังขึ้น	
40	480	-1	-1		กำลังขึ้น	
41	492	-1	-1		กำลังขึ้น	
42	504	185	-1		โดดจนถึงสุดแล้ว	
43	516	-1	-1		กำลังตก	
44	528	-1	-1		กำลังตก	
45	540	6	-1	Jump forward	กำลังจะม้วนหน้า	
46	552	-1	-1			
47	564	-1	-1		กำลังม้วนหน้า	
48	576	130	-1			
49	588	-1	-1			
50	600	-1	-1			
51	612	131	-1			
52	624	-1	-1		กำลังตก หลังจากม้วน	
53	636	-1	-1			
54	648	7	-1	Jump backward	กำลังจะม้วนหลัง	

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
55	660	-1	-1			
56	672	183	-1			
57	684	-1	-1			
58	696	-1	-1			
59	708	184	-1			
60	720	-1	-1		กำลังตก หลังจากม้วน	
61	732	-1	-1			
62	744	8	0	Guard	แขนยกมาเริ่มการ์ดแล้ว	
63	756	-1	-1		การ์ดปรกติ	
64	768	-1	-1			
65	780	-1	-1			
66	792	-1	-1		แขนเริ่มคลายการ์ด	
67	804	9	-1	Crounch Guard	แขนยกมาเริ่มการ์ดแล้ว	
68	816	-1	-1		การ์ดปรกติ	
69	828	-1	-1			
70	840	-1	-1		แขนเริ่มคลายการ์ด	
71	852	-1	-1			
72	864	10	-1	Air Guard	แขนยกมาเริ่มการ์ดแล้ว	
73	876	-1	-1		การ์ดปรกติ	
74	888	-1	-1			
75	900	-1	-1		แขนเริ่มคลายการ์ด	
76	912	5	-1	Jump Staright	กำลังตก	
77	924	21	-1	Roll		
78	936	-1	-1			
79	948	-1	-1			
80	960	-1	-1			
81	972	-1	-1			
82	984	-1	-1			
83	996	-1	-1			
84	1008	-1	-1			
85	1020	-1	-1			
86	1032	-1	-1			
87	1044	-1	-1			
88	1056	-1	-1			
89	1068	22	-1	Before Fight	ท่าตอนเริ่มต่อสู้	
90	1080	-1	-1			
91	1092	-1	-1			
92	1104	-1	-1			
93	1116	-1	-1			
94	1128	-1	-1			
95	1140	-1	-1			
96	1152	-1	-1			
97	1164	-1	-1	a		
98	1176	-1	-1			
99	1188	-1	-1			
100	1200	-1	-1			
101	1212	-1	-1			
102	1224	-1	-1			
103	1236	-1	-1			
104	1248	-1	-1			
105	1260	-1	-1			
106	1272	-1	-1			
107	1284	-1	-1	?	ยืน	
108	1296	-1	-1			
109	1308	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
110	1320	-1	-1			
111	1332	23	-1	Victory Pose	ท่าชูมือตอนชนะ	
112	1344	-1	-1			
113	1356	-1	-1			
114	1368	-1	-1			
115	1380	-1	-1	Victory Pose	ท่ากอดอกตอนชนะ	
116	1392	-1	-1			
117	1404	-1	-1			
118	1416	-1	-1			
119	1428	-1	-1			
120	1440	-1	-1			
121	1452	-1	-1			
122	1464	-1	-1			
123	1476	-1	-1			
124	1488	-1	-1			
125	1500	-1	-1			
126	1512	-1	-1			
127	1524	-1	-1	a		
128	1536	-1	-1			
129	1548	-1	-1			
130	1560	-1	-1			
131	1572	-1	-1			
132	1584	-1	-1			
133	1596	-1	-1			
134	1608	-1	-1			
135	1620	-1	-1			
136	1632	-1	-1	a		
137	1644	-1	-1			
138	1656	-1	-1			
139	1668	-1	-1			
140	1680	-1	-1			
141	1692	-1	-1			
142	1704	-1	-1			
143	1716	-1	-1			
144	1728	-1	-1			
145	1740	-1	-1			
146	1752	-1	-1	Victory Pose	หันหลังให้กล้อง	
147	1764	-1	-1			
148	1776	-1	-1			
149	1788	-1	-1			
150	1800	-1	-1			
151	1812	-1	-1			
152	1824	11	-1	Damage		
153	1836	150	-1	?		
154	1848	-1	-1			
155	1860	-1	-1			
156	1872	-1	-1			
157	1884	-1	-1			
158	1896	22	-1	Before Fight	ท่าก่อนสู้	
159	1908	-1	-1			
160	1920	-1	-1			
161	1932	-1	-1			
162	1944	-1	-1			
163	1956	-1	-1	?		
164	1968	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
165	1980	-1	-1			
166	1992	-1	-1			
167	2004	-1	-1			
168	2016	-1	-1	?		
169	2028	-1	-1			
170	2040	-1	-1			
171	2052	-1	-1			
172	2064	-1	-1			
173	2076	11	0	Damage	ท่าโดนอัดตอนยืน 1	
174	2088	-1	-1			
175	2100	-1	-1			
176	2112	-1	-1			
177	2124	-1	-1		ท่าโดนอัดตอนยืน 2	
178	2136	-1	-1			
179	2148	-1	-1			
180	2160	-1	-1			
181	2172	-1	1		โดนอัดกำลังจะล้ม	
182	2184	-1	-1	a	โดนอัดล้ม	
183	2196	-1	-1			
184	2208	-1	-1		ลอยอยู่ระหว่างโดนอัด	
185	2220	-1	-1			
186	2232	-1	-1			
187	2244	-1	-1			
188	2256	-1	-1			
189	2268	-1	-1		นอนกองอยู่กับพื้น	
190	2280	-1	-1			
191	2292	-1	-1		ลอยอยู่ระหว่างโดนอัด	
192	2304	-1	-1			
193	2316	-1	-1			
194	2328	-1	-1			
195	2340	-1	-1		หัวหิ่มพื้นกำลังตก	
196	2352	-1	-1		ลอยอยู่ระหว่างโดนอัด	
197	2364	-1	-1			
198	2376	-1	-1	a		
199	2388	-1	-1		หลังกระแทกพื้น	
200	2400	-1	-1		ก้นกระแทกพื้น	
201	2412	-1	0			
202	2424	-1	-1		ท่าโดนอัดตอนยืน	
203	2436	-1	-1			
204	2448	-1	-1			
205	2460	-1	-1			
206	2472	-1	-1			
207	2484	-1	-1			
208	2496	-1	-1			
209	2508	-1	-1			
210	2520	-1	-1	a		
211	2532	-1	-1			
212	2544	-1	-1			
213	2556	-1	-1			
214	2568	-1	-1			
215	2580	-1	-1			
216	2592	-1	-1			
217	2604	-1	-1			
218	2616	-1	-1			
219	2628	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
220	2640	12	-1	Crounch Damage	ท่าโดนอัดตอนล้ม	
221	2652	-1	-1			
222	2664	-1	-1			
223	2676	-1	-1			
224	2688	-1	-1			
225	2700	-1	-1			
226	2712	-1	-1			
227	2724	-1	-1			
228	2736	-1	1	a	ล้มไปกองกับพื้น	
229	2748	-1	-1			
230	2760	-1	-1			
231	2772	-1	-1			
232	2784	-1	-1			
233	2796	-1	-1		โดนอัด ลอยอยู่	
234	2808	-1	-1			
235	2820	-1	-1		หลังกระแทกพื้น	
236	2832	132	0		Air Recovery begin	
237	2844	-1	-1	Jumping	Air recovery	
238	2856	-1	-1			
239	2868	-1	-1			
240	2880	-1	-1	Jumping		
241	2892	-1	-1			
242	2904	11	2	Damage	นอนกองอยู่กับพื้น	
243	2916	13	-1	Recover to stand	กำลังจะลุกจากท่าล้ม	
244	2928	-1	-1		ใช้มือดีดตัวขึ้น	
245	2940	-1	-1			
246	2952	-1	-1			
247	2964	-1	-1		กลับตัวลุกขึ้น	
248	2976	-1	-1			
249	2988	-1	-1		ยืนได้แล้ว	
250	3000	11	3	Dizzy	มึน	
251	3012	-1	-1			
252	3024	-1	-1			
253	3036	-1	-1			
254	3048	133	0	Auto Roll	กลิ้งเองไม่ต้องสั่ง	
255	3060	-1	-1			
256	3072	-1	-1			
257	3084	-1	-1			
258	3096	11	1	Damage	หลังกระแทกพื้น	
259	3108	-1	-1		โดนอัด ลอยอยู่	
260	3120	-1	-1			
261	3132	-1	-1			
262	3144	-1	0		โดนอัดตอนยืน	
263	3156	-1	-1			
264	3168	-1	1		โดนอัดลอย	
265	3180	-1	-1			
266	3192	14	-1	Electrick Shock	โดนท่าช็อตไฟฟ้า	
267	3204	-1	-1			
268	3216	11	-1	Damage	โดนอัดลอย	
269	3228	-1	-1			
270	3240	-1	-1			
271	3252	-1	-1			
272	3264	-1	-1			
273	3276	-1	-1			
274	3288	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
275	3300	16	0	Throw Draw	ท่าตอนที่หุ่มพร้อมกัน	
276	3312	-1	-1			
277	3324	-1	-1			
278	3336	17	-1	Crounch Throw Draw	แบบล้ม ?มีด้วยหรอเนี่ย	
279	3348	-1	-1			
280	3360	-1	-1			
281	3372	18	-1	Air Throw Draw		
282	3384	-1	-1			
283	3396	-1	-1			
284	3408	11	-1	Damage	ก้นกระแทกพื้น	
285	3420	-1	-1			
286	3432	-1	-1		โดนอัดอยู่กลางอากาศ	
287	3444	-1	-1			
288	3456	-1	-1			
289	3468	-1	-1			
290	3480	-1	-1			
291	3492	-1	-1			
292	3504	-1	-1			
293	3516	-1	-1			
294	3528	-1	-1			
295	3540	-1	-1			
296	3552	-1	-1			
297	3564	-1	-1			
298	3576	20	-1	Throw	ท่าหุ่มข้ามหลัง	
299	3588	-1	-1		Shoulder Toss: HP	
300	3600	-1	-1		Shoulder Toss: HP	
301	3612	-1	-1		Shoulder Toss: HP	
302	3624	-1	-1		Shoulder Toss: HP	
303	3636	-1	-1		Shoulder Toss: HP	
304	3648	-1	-1		Shoulder Toss: HP	
305	3660	-1	-1	a	Leg Roll: HK	
306	3672	-1	-1		Leg Roll: HK	
307	3684	-1	-1		Leg Roll: HK	
308	3696	-1	-1		Leg Roll: HK	
309	3708	-1	-1		Leg Roll: HK	
310	3720	-1	-1		Leg Roll: HK	
311	3732	-1	-1		Leg Roll: HK	
312	3744	-1	-1		Leg Roll: HK	
313	3756	-1	-1		Leg Roll: HK	
314	3768	15	-1	Begin Throw	กำลังเริ่มจับหุ่ม	
315	3780	-1	-1			
316	3792	-1	-1			
317	3804	-1	-1			
318	3816	-1	-1			
319	3828	142	-1	Shoryuken L	เบา	
320	3840	-1	-1			
321	3852	-1	-1			
322	3864	-1	-1			
323	3876	-1	-1			
324	3888	-1	-1			
325	3900	-1	-1			
326	3912	-1	-1			
327	3924	-1	-1			
328	3936	-1	-1			
329	3948	143	-1	Shoryuken M	กลาง	

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
330	3960	-1	-1			
331	3972	-1	-1			
332	3984	-1	-1			
333	3996	-1	-1			
334	4008	-1	-1			
335	4020	-1	-1			
336	4032	-1	-1			
337	4044	144	-1	Shoryuken H	หนัก	
338	4056	-1	-1			
339	4068	-1	-1			
340	4080	-1	-1			
341	4092	-1	-1			
342	4104	-1	-1			
343	4116	-1	-1			
344	4128	-1	-1			
345	4140	112	-1	T.Kick Default		
346	4152	-1	-1			
347	4164	-1	-1			
348	4176	-1	-1			
349	4188	-1	-1			
350	4200	-1	-1			
351	4212	-1	-1	a		
352	4224	-1	-1			
353	4236	-1	-1			
354	4248	-1	-1			
355	4260	-1	-1	T.Kick L		
356	4272	-1	-1			
357	4284	-1	-1			
358	4296	-1	-1	a		
359	4308	-1	-1			
360	4320	-1	-1	T.Kick M		
361	4332	-1	-1			
362	4344	-1	-1			
363	4356	-1	-1			
364	4368	-1	-1			
365	4380	-1	-1	T.Kick H		
366	4392	-1	-1			
367	4404	-1	-1			
368	4416	-1	-1			
369	4428	-1	-1			
370	4440	-1	-1	a		
371	4452	-1	-1			
372	4464	-1	-1			
373	4476	-1	-1			
374	4488	-1	-1			
375	4500	-1	-1			
376	4512	154	-1	T.Kick Air L	กลางอากาศ	
377	4524	-1	-1			
378	4536	-1	-1			
379	4548	-1	-1			
380	4560	-1	-1			
381	4572	-1	-1			
382	4584	-1	-1	a		
383	4596	-1	-1			
384	4608	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
385	4620	-1	-1			
386	4632	-1	-1			
387	4644	-1	-1			
388	4656	-1	-1	T.Kick Air M	เริ่มเก็บขาลงแล้ว	
389	4668	-1	-1	T.Kick Air H		
390	4680	-1	-1			
391	4692	-1	-1			
392	4704	-1	-1			
393	4716	134	-1	Hadoken L	เบา	
394	4728	-1	-1		เตรียมปล่อย	
395	4740	-1	-1			
396	4752	-1	-1		ปล่อย	
397	4764	-1	-1			
398	4776	-1	-1			
399	4788	-1	-1			
400	4800	-1	-1			
401	4812	-1	-1	a		
402	4824	-1	-1			
403	4836	-1	-1			
404	4848	-1	-1			
405	4860	-1	-1			
406	4872	-1	-1			
407	4884	-1	-1			
408	4896	-1	-1			
409	4908	135	-1	Hadoken M / FL	กลาง / ไฟเบา	
410	4920	-1	-1		เตรียมปล่อย	
411	4932	-1	-1			
412	4944	-1	-1		ปล่อย	
413	4956	-1	-1			
414	4968	-1	-1			
415	4980	-1	-1			
416	4992	-1	-1	a		
417	5004	-1	-1			
418	5016	-1	-1			
419	5028	-1	-1			
420	5040	-1	-1			
421	5052	-1	-1			
422	5064	-1	-1			
423	5076	-1	-1			
424	5088	-1	-1			
425	5100	136	-1	Hadoken H / FM	หนัก / ไฟกลาง	
426	5112	-1	-1		เตรียมปล่อย	
427	5124	-1	-1			
428	5136	-1	-1		ปล่อย	
429	5148	-1	-1			
430	5160	-1	-1			
431	5172	-1	-1	a		
432	5184	-1	-1			
433	5196	-1	-1			
434	5208	-1	-1			
435	5220	-1	-1			
436	5232	-1	-1			
437	5244	-1	-1			
438	5256	-1	-1			
439	5268	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
440	5280	-1	-1			
441	5292	137	-1	Hadoken Fire H	ยืนไฟหนัก	
442	5304	-1	-1		เตรียมปล่อย	
443	5316	-1	-1			
444	5328	-1	-1		ปล่อย	
445	5340	-1	-1			
446	5352	-1	-1			
447	5364	-1	-1			
448	5376	-1	-1			
449	5388	-1	-1	a		
450	5400	-1	-1			
451	5412	-1	-1			
452	5424	-1	-1			
453	5436	-1	-1			
454	5448	-1	-1			
455	5460	-1	-1			
456	5472	-1	-1			
457	5484	110	-1	Hadoken unknown	ยืน	
458	5496	-1	-1		เตรียมปล่อย	
459	5508	-1	-1			
460	5520	-1	-1		ปล่อย	
461	5532	-1	-1			
462	5544	-1	-1			
463	5556	-1	-1	a		
464	5568	-1	-1			
465	5580	-1	-1			
466	5592	-1	-1			
467	5604	-1	-1			
468	5616	-1	-1			
469	5628	-1	-1			
470	5640	-1	-1			
471	5652	-1	-1			
472	5664	-1	-1			
473	5676	-1	-1	Hadoken unknown	ยืน	
474	5688	-1	-1		เตรียมปล่อย	
475	5700	-1	-1			
476	5712	-1	-1		ปล่อย	
477	5724	-1	-1			
478	5736	-1	-1			
479	5748	-1	-1			
480	5760	-1	-1			
481	5772	-1	-1	a		
482	5784	-1	-1			
483	5796	-1	-1			
484	5808	-1	-1			
485	5820	-1	-1			
486	5832	-1	-1			
487	5844	-1	-1			
488	5856	-1	-1		ยืน	
489	5868	-1	-1			
490	5880	24	-1	Zero Counter?	น่าจะใช่นะ	
491	5892	-1	-1			
492	5904	-1	-1			
493	5916	-1	-1		สวนด้วยโชริวเคน	
494	5928	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
495	5940	-1	-1	a		
496	5952	-1	-1			
497	5964	-1	-1			
498	5976	-1	-1			
499	5988	-1	-1			
500	6000	-1	-1			
501	6012	-1	-1			
502	6024	62	-1	Crounch Kick	ล้มเตะ กวาดขา	
503	6036	-1	-1			
504	6048	-1	-1			
505	6060	-1	-1			
506	6072	-1	-1			
507	6084	-1	-1			
508	6096	19	-1	Miss Throw?		
509	6108	-1	-1			
510	6120	-1	-1			
511	6132	-1	-1			
512	6144	120	-1	Shinku T.Kick Default		
513	6156	-1	-1			
514	6168	-1	-1			
515	6180	-1	-1			
516	6192	-1	-1			
517	6204	-1	-1			
518	6216	-1	-1			
519	6228	-1	-1			
520	6240	-1	-1			
521	6252	-1	-1			
522	6264	-1	-1	Shinku T.Kick H		
523	6276	-1	-1			
524	6288	-1	-1			
525	6300	-1	-1			
526	6312	-1	-1			
527	6324	-1	-1			
528	6336	-1	-1	a		
529	6348	-1	-1			
530	6360	-1	-1			
531	6372	-1	-1	Shinku T.Kick L		
532	6384	-1	-1			
533	6396	-1	-1			
534	6408	-1	-1			
535	6420	-1	-1			
536	6432	-1	-1			
537	6444	-1	-1			
538	6456	-1	-1			
539	6468	-1	-1			
540	6480	-1	-1	Shinku T.Kick M		
541	6492	-1	-1			
542	6504	-1	-1	a		
543	6516	-1	-1			
544	6528	-1	-1			
545	6540	-1	-1			
546	6552	-1	-1			
547	6564	-1	-1			
548	6576	-1	-1			
549	6588	-1	-1	Shinku T.Kick L		

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
550	6600	-1	-1			
551	6612	-1	-1			
552	6624	-1	-1			
553	6636	-1	-1			
554	6648	-1	-1			
555	6660	-1	-1	Shinku T.Kick M		
556	6672	-1	-1	a		
557	6684	-1	-1			
558	6696	-1	-1			
559	6708	-1	-1			
560	6720	-1	-1			
561	6732	-1	-1	Shinku T.Kick H		
562	6744	-1	-1			
563	6756	-1	-1			
564	6768	-1	-1			
565	6780	-1	-1			
566	6792	-1	-1			
567	6804	22	-1	Before Fight	ท่าก่อนต่อสู้ ขยับข้อมือ	
568	6816	-1	-1			
569	6828	25	-1	Challenge	ยั่วยุ	
570	6840	-1	-1			
571	6852	-1	-1			
572	6864	-1	-1			
573	6876	-1	-1			
574	6888	-1	-1			
575	6900	-1	-1			
576	6912	-1	-1			
577	6924	-1	-1			
578	6936	-1	-1			
579	6948	22	-1	Before Fight	ท่ายกผ้าคาดหัว	
580	6960	-1	-1			
581	6972	-1	-1			
582	6984	-1	-1			
583	6996	-1	-1			
584	7008	-1	-1	a		
585	7020	-1	-1			
586	7032	-1	-1			
587	7044	-1	-1			
588	7056	-1	-1			
589	7068	-1	-1			
590	7080	-1	-1			
591	7092	-1	-1			
592	7104	138	-1	Hadoken Fake	fake	
593	7116	-1	-1		เตรียมยิง	
594	7128	-1	-1			
595	7140	-1	-1		ยิง	
596	7152	-1	-1			
597	7164	-1	-1	a		
598	7176	-1	-1			
599	7188	-1	-1			
600	7200	-1	-1			
601	7212	-1	-1			
602	7224	139	-1	Shinku Hadoken L	ยืน	
603	7236	-1	-1		เตรียมยิง	
604	7248	-1	-1			

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
605	7260	-1	-1		ยิง	
606	7272	-1	-1			
607	7284	-1	-1			
608	7296	-1	-1	a		
609	7308	-1	-1			
610	7320	-1	-1			
611	7332	-1	-1			
612	7344	-1	-1			
613	7356	-1	-1			
614	7368	-1	-1			
615	7380	-1	-1			
616	7392	-1	-1			
617	7404	-1	-1			
618	7416	140	-1	Shinku Hadoken M	ยืน	
619	7428	-1	-1		เตรียมยิง	
620	7440	-1	-1			
621	7452	-1	-1		ยิง	
622	7464	-1	-1			
623	7476	-1	-1			
624	7488	-1	-1			
625	7500	-1	-1	a		
626	7512	-1	-1			
627	7524	-1	-1			
628	7536	-1	-1			
629	7548	-1	-1			
630	7560	-1	-1			
631	7572	-1	-1		ยืน	
632	7584	-1	-1			
633	7596	-1	-1			
634	7608	141	-1	Shinku Hadoken H		
635	7620	-1	-1		เตรียมยิง	
636	7632	-1	-1			
637	7644	-1	-1		ยิง	
638	7656	-1	-1			
639	7668	-1	-1			
640	7680	-1	-1			
641	7692	-1	-1			
642	7704	-1	-1	a		
643	7716	-1	-1			
644	7728	-1	-1			
645	7740	-1	-1			
646	7752	-1	-1			
647	7764	-1	-1			
648	7776	-1	-1		ยืน	
649	7788	-1	-1			
650	7800	122	-1	Metsu Shoryuken	เริ่มชักศอก	
651	7812	-1	-1			
652	7824	-1	-1			
653	7836	-1	-1			
654	7848	-1	-1		ศอก	
655	7860	-1	-1			
656	7872	-1	-1			
657	7884	-1	-1			
658	7896	-1	-1			
659	7908	-1	-1		เริ่มโชริวเคน	

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
660	7920	-1	-1			
661	7932	-1	-1			
662	7944	-1	-1			
663	7956	-1	-1	a		
664	7968	-1	-1			
665	7980	-1	-1			
666	7992	-1	-1			
667	8004	-1	-1			
668	8016	-1	-1		ตกถังพื้น	
669	8028	-1	-1		เริ่ม โซริวเคนคนั้งที่สอง	
670	8040	-1	-1			
671	8052	-1	-1			
672	8064	-1	-1			
673	8076	-1	-1			
674	8088	-1	-1			
675	8100	-1	-1			
676	8112	-1	-1			
677	8124	-1	-1	a		
678	8136	-1	-1			
679	8148	-1	-1			
680	8160	-1	-1			
681	8172	-1	-1			
682	8184	-1	-1			
683	8196	-1	-1			
684	8208	-1	-1			
685	8220	-1	-1			
686	8232	-1	-1		จบ metsu shryuken	
687	8244	0	-1	Stand	?	
688	8256	-1	-1			
689	8268	-1	-1			
690	8280	-1	-1			
691	8292	-1	-1			
692	8304	-1	-1			
693	8316	30	-1	Punch	Elbow Jab: LP	[Close Attacks]
694	8328	-1	-1		Elbow Jab: LP	[Close Attacks]
695	8340	-1	-1		Elbow Jab: LP	[Close Attacks]
696	8352	-1	-1		Elbow Jab: LP	[Close Attacks]
697	8364	31	-1	Punch	Jab: LP	
698	8376	-1	-1		Jab: LP	
699	8388	-1	-1		Jab: LP	
700	8400	-1	-1		Jab: LP	
701	8412	32	-1	Punch	Body Blow: MP	[Close Attacks]
702	8424	-1	-1		Body Blow: MP	[Close Attacks]
703	8436	-1	-1		Body Blow: MP	[Close Attacks]
704	8448	-1	-1		Body Blow: MP	[Close Attacks]
705	8460	-1	-1		Body Blow: MP	[Close Attacks]
706	8472	-1	-1		Body Blow: MP	[Close Attacks]
707	8484	-1	-1		Body Blow: MP	[Close Attacks]
708	8496	33	-1	Punch	Straight: MP	
709	8508	-1	-1		Straight: MP	
710	8520	-1	-1		Straight: MP	
711	8532	-1	-1		Straight: MP	
712	8544	-1	-1		Straight: MP	
713	8556	-1	-1		Straight: MP	
714	8568	-1	-1		Straight: MP	

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
715	8580	34	-1	Punch	Uppercut: HP	[Close Attacks]
716	8592	-1	-1		Uppercut: HP	[Close Attacks]
717	8604	-1	-1		Uppercut: HP	[Close Attacks]
718	8616	-1	-1		Uppercut: HP	[Close Attacks]
719	8628	-1	-1		Uppercut: HP	[Close Attacks]
720	8640	-1	-1		Uppercut: HP	[Close Attacks]
721	8652	-1	-1		Uppercut: HP	[Close Attacks]
722	8664	-1	-1		Uppercut: HP	[Close Attacks]
723	8676	35	-1	Punch	Strong Jab: HP	
724	8688	-1	-1		Strong Jab: HP	
725	8700	-1	-1		Strong Jab: HP	
726	8712	-1	-1		Strong Jab: HP	
727	8724	-1	-1		Strong Jab: HP	
728	8736	-1	-1		Strong Jab: HP	
729	8748	-1	-1		Strong Jab: HP	
730	8760	40	-1	Kick	Low Kick: LK	
731	8772	-1	-1		Low Kick: LK	
732	8784	-1	-1		Low Kick: LK	
733	8796	-1	-1		Low Kick: LK	
734	8808	-1	-1		Low Kick: LK	
735	8820	-1	-1		Low Kick: LK	
736	8832	-1	-1		Low Kick: LK	
737	8844	41	-1	Kick	High Kick: MK	
738	8856	-1	-1		High Kick: MK	
739	8868	-1	-1		High Kick: MK	
740	8880	-1	-1		High Kick: MK	
741	8892	-1	-1		High Kick: MK	
742	8904	-1	-1		High Kick: MK	
743	8916	-1	-1		High Kick: MK	
744	8928	42	-1	Kick	Axe Kick: HK	[Close Attacks]
745	8940	-1	-1		Axe Kick: HK	[Close Attacks]
746	8952	-1	-1		Axe Kick: HK	[Close Attacks]
747	8964	-1	-1		Axe Kick: HK	[Close Attacks]
748	8976	-1	-1		Axe Kick: HK	[Close Attacks]
749	8988	-1	-1		Axe Kick: HK	[Close Attacks]
750	9000	-1	-1		Axe Kick: HK	[Close Attacks]
751	9012	-1	-1	a	Axe Kick: HK	[Close Attacks]
752	9024	-1	-1		Axe Kick: HK	[Close Attacks]
753	9036	-1	-1		Axe Kick: HK	[Close Attacks]
754	9048	-1	-1		Axe Kick: HK	[Close Attacks]
755	9060	-1	-1		Axe Kick: HK	[Close Attacks]
756	9072	-1	-1		Axe Kick: HK	[Close Attacks]
757	9084	-1	-1		Axe Kick: HK	[Close Attacks]
758	9096	43	-1	Kick	Roundhouse Kick: HK	
759	9108	-1	-1		Roundhouse Kick: HK	
760	9120	-1	-1		Roundhouse Kick: HK	
761	9132	-1	-1		Roundhouse Kick: HK	
762	9144	-1	-1		Roundhouse Kick: HK	
763	9156	-1	-1		Roundhouse Kick: HK	
764	9168	-1	-1		Roundhouse Kick: HK	
765	9180	-1	-1		Roundhouse Kick: HK	
766	9192	50	-1	Crounch Punch	Jab: LP	
767	9204	-1	-1		Jab: LP	
768	9216	-1	-1		Jab: LP	
769	9228	-1	-1		Jab: LP	

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
770	9240	51	-1	Crounch Punch	Straight: MP	
771	9252	-1	-1		Straight: MP	
772	9264	-1	-1		Straight: MP	
773	9276	-1	-1		Straight: MP	
774	9288	-1	-1		Straight: MP	
775	9300	-1	-1		Straight: MP	
776	9312	52	-1	Crounch Punch	Uppercut: HP	
777	9324	-1	-1		Uppercut: HP	
778	9336	-1	-1		Uppercut: HP	
779	9348	-1	-1		Uppercut: HP	
780	9360	-1	-1		Uppercut: HP	
781	9372	-1	-1		Uppercut: HP	
782	9384	-1	-1		Uppercut: HP	
783	9396	-1	-1		Uppercut: HP	
784	9408	60	-1	Crounch Kick	Short Kick: LK	
785	9420	-1	-1		Short Kick: LK	
786	9432	-1	-1		Short Kick: LK	
787	9444	-1	-1		Short Kick: LK	
788	9456	61	-1	Crounch Kick	Long Kick: MK	
789	9468	-1	-1		Long Kick: MK	
790	9480	-1	-1		Long Kick: MK	
791	9492	-1	-1		Long Kick: MK	
792	9504	-1	-1		Long Kick: MK	
793	9516	-1	-1		Long Kick: MK	
794	9528	-1	-1		Long Kick: MK	
795	9540	62	-1	Crounch Kick	Sweep Kick: HK	
796	9552	-1	-1		Sweep Kick: HK	
797	9564	-1	-1		Sweep Kick: HK	
798	9576	-1	-1		Sweep Kick: HK	
799	9588	-1	-1		Sweep Kick: HK	
800	9600	-1	-1		Sweep Kick: HK	
801	9612	70	5	Jump Punch	Downward Jab: LP	
802	9624	-1	-1		Downward Jab: LP	
803	9636	-1	-1		Downward Jab: LP	
804	9648	-1	-1		Downward Jab: LP	
805	9660	71	-1	Jump Punch	Downward Punch: MP	
806	9672	-1	-1		Downward Punch: MP	
807	9684	-1	-1		Downward Punch: MP	
808	9696	-1	-1		Downward Punch: MP	
809	9708	72	-1	Jump Punch	Downward Punch: HP	
810	9720	-1	-1		Downward Punch: HP	
811	9732	-1	-1		Downward Punch: HP	
812	9744	-1	-1		Downward Punch: HP	
813	9756	80	-1	Jump Kick	Punt Kick (up): LK	
814	9768	-1	-1		Punt Kick (up): LK	
815	9780	-1	-1		Punt Kick (up): LK	
816	9792	-1	-1		Punt Kick (up): LK	
817	9804	-1	-1		Punt Kick (up): LK	
818	9816	-1	-1		Punt Kick (up): LK	
819	9828	81	-1	Jump Kick	High Kick (up): MK	
820	9840	-1	-1		High Kick (up): MK	
821	9852	-1	-1		High Kick (up): MK	
822	9864	-1	-1		High Kick (up): MK	
823	9876	-1	-1		High Kick (up): MK	
824	9888	-1	-1		High Kick (up): MK	
825	9900	82	-1	Jump Kick	Roundhouse (up): HK	
826	9912	-1	-1		Roundhouse (up): HK	
827	9924	-1	-1		Roundhouse (up): HK	
828	9936	-1	-1		Roundhouse (up): HK	
829	9948	-1	-1		Roundhouse (up): HK	
830	9960	-1	-1		Roundhouse (up): HK	
831	9972	-1	-1		Roundhouse (up): HK	
832	9984	90	-1	Jump side punch	Uppercut (side): LP	
833	9996	-1	-1		Uppercut (side): LP	
834	10008	-1	-1		Uppercut (side): LP	
835	10020	-1	-1		Uppercut (side): LP	
836	10032	91	-1	Jump side Uppercut	Uppercut (side): MP	
837	10044	-1	-1		Uppercut (side): MP	
838	10056	-1	-1		Uppercut (side): MP	
839	10068	-1	-1		Uppercut (side): MP	
840	10080	-1	-1		Uppercut (side): MP	
841	10092	-1	-1		Uppercut (side): MP	
842	10104	-1	-1		Uppercut (side): MP	
843	10116	-1	-1		Uppercut (side): MP	
844	10128	-1	-1		Uppercut (side): MP	
845	10140	92	-1	Jump side Punch	Uppercut (side): HP	
846	10152	-1	-1		Uppercut (side): HP	
847	10164	-1	-1		Uppercut (side): HP	
848	10176	-1	-1		Uppercut (side): HP	
849	10188	100	-1	Jump side Kick	Knee Drop (side): LK	
850	10200	-1	-1		Knee Drop (side): LK	
851	10212	-1	-1		Knee Drop (side): LK	
852	10224	-1	-1		Knee Drop (side): LK	
853	10236	101	-1	Jump side Kick	Thrust Kick (side): MK	
854	10248	-1	-1		Thrust Kick (side): MK	
855	10260	-1	-1		Thrust Kick (side): MK	
856	10272	-1	-1		Thrust Kick (side): MK	
857	10284	-1	-1		Thrust Kick (side): MK	
858	10296	102	-1	Jump side Kick	Thrust Kick (side): HK	
859	10308	-1	-1		Thrust Kick (side): HK	
860	10320	-1	-1		Thrust Kick (side): HK	
861	10332	-1	-1		Thrust Kick (side): HK	
862	10344	-1	-1		Thrust Kick (side): HK	
863	10356	-1	-1		Thrust Kick (side): HK	
864	10368	36	0	Punch	Sakotsu Wari: F+MP [Misc. Techniques]	
865	10380	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
866	10392	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
867	10404	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
868	10416	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
869	10428	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
870	10440	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
871	10452	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
872	10464	-1	-1	a	Sakotsu Wari: F+MP [Misc. Techniques]	
873	10476	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
874	10488	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
875	10500	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
876	10512	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
877	10524	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
878	10536	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	
879	10548	-1	-1		Sakotsu Wari: F+MP [Misc. Techniques]	

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
880	10560	44	-1	Kick	Senpukyaku: F+MK [Misc. Techniques]	
881	10572	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
882	10584	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
883	10596	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
884	10608	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
885	10620	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
886	10632	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
887	10644	-1	-1		Senpukyaku: F+MK [Misc. Techniques]	
888	10656	-1	-1			
889	10668	-1	-1			
890	10680	150	-1	Seichu Nidan Tsuki	ท่าศอก X-Only	
891	10692	-1	-1		ท่าศอก X-Only	
892	10704	-1	-1		ท่าศอก X-Only	
893	10716	-1	-1		ท่าศอก X-Only	
894	10728	-1	-1		ท่าศอก X-Only	
895	10740	-1	-1		ท่าศอก X-Only	
896	10752	-1	-1		ท่าศอก X-Only	
897	10764	-1	-1		ท่าศอก X-Only	
898	10776	-1	-1		ท่าศอก X-Only	
899	10788	-1	-1	Punch		
900	10800	-1	-1			
901	10812	-1	-1			
902	10824	-1	-1			
903	10836	-1	-1			
904	10848	-1	-1			
905	10860	-1	-1	Punch		
906	10872	-1	-1			
907	10884	-1	-1			
908	10896	-1	-1			
909	10908	-1	-1			
910	10920	-1	-1			
911	10932	-1	-1			
912	10944	-1	-1			
913	10956	-1	-1	?	คล้าย ๆ ท่าจับหุ่ม	
914	10968	-1	-1			
915	10980	151	-1	T.Kick Series Correction		
916	10992	152	-1			
917	11004	153	-1			
918	11016	154	-1			
919	11028	155	-1			
920	11040	156	-1			
921	11052	157	-1			
922	11064	158	-1			
923	11076	159	-1			
924	11088	160	-1			
925	11100	161	-1	Walk FW 10		
926	11112	162	-1	Walk FW 15		
927	11124	163	-1	Walk FW 20		
928	11136	164	-1	Walk FW 30		
929	11148	165	-1	Walk FW 40		
930	11160	166	-1	Walk FW 50		
931	11172	167	-1	Walk FW 60		
932	11184	168	-1	Walk FW 70		
933	11196	169	-1	Walk FW 80		
934	11208	170	-1	Walk BW 10		

AniNo	AniNo*12	AniSet	Anitype1	Animation Name	Description	Attack Range
935	11220	171	-1	Walk BW 15		
936	11232	172	-1	Walk BW 20		
937	11244	173	-1	Walk BW 30		
938	11256	174	-1	Walk BW 40		
939	11268	175	-1	Walk BW 50		
940	11280	176	-1	Walk BW 60		
941	11292	177	-1	Walk BW 70		
942	11304	178	-1	Walk BW 80		
943	11316	179	-1			
944	11328	180	-1	Aerial Recover F	Front	
945	11340	181	-1	Aerial Recover N	Neutral	
946	11352	182	-1	Aerial Recover B	Back	
947	11364	199	-1	Fake State		

ภาคผนวก ง ค่าความบาดเจ็บที่เกิดจากท่าต่างๆ ที่กำหนดขึ้นมาเอง

```
m_myDefineDamage[A_PUNCH_CLP]=6;
m_myDefineDamage[A_PUNCH_LP]=6;
m_myDefineDamage[A_PUNCH_CMP]=14;
m_myDefineDamage[A_PUNCH_MP]=14;
m_myDefineDamage[A_PUNCH_CHP]=16;
m_myDefineDamage[A_PUNCH_HP]=16;
m_myDefineDamage[A_PUNCH_FMP]=8;

// Kick (LK, MK, HK, CHK, FMK)
m_myDefineDamage[A_KICK_LK]=7;
m_myDefineDamage[A_KICK_MK]=13;
m_myDefineDamage[A_KICK_CHK]=10;
m_myDefineDamage[A_KICK_HK]=18;
m_myDefineDamage[A_KICK_FMK]=13;

// Crounch Punch (LP, MP, HP)
m_myDefineDamage[A_CROUNCH_PUNCH_LP]=5;
m_myDefineDamage[A_CROUNCH_PUNCH_MP]=12;
m_myDefineDamage[A_CROUNCH_PUNCH_HP]=16;

// Crounch Kick (LK, MK, HK)
m_myDefineDamage[A_CROUNCH_KICK_LK]=4;
m_myDefineDamage[A_CROUNCH_KICK_MK]=11;
m_myDefineDamage[A_CROUNCH_KICK_HK]=16;

// Jump Punch (LP, MP, HP)
m_myDefineDamage[A_JUMP_PUNCH_LP]=8;
m_myDefineDamage[A_JUMP_PUNCH_MP]=12;
m_myDefineDamage[A_JUMP_PUNCH_HP]=16;

// Jump Kick (LK, MK, HK)
m_myDefineDamage[A_JUMP_KICK_LK]=7;
m_myDefineDamage[A_JUMP_KICK_MK]=12;
m_myDefineDamage[A_JUMP_KICK_HK]=16;

// Jump side punch (LP, MP, HP)
 m_myDefineDamage[A_JUMP_SIDE_PUNCH_LP]=8;
 m_myDefineDamage[A_JUMP_SIDE_PUNCH_MP]=8;
 m_myDefineDamage[A_JUMP_SIDE_PUNCH_HP]=16;

// Jump side Kick (LK, MK, HK)
m_myDefineDamage[A_JUMP_SIDE_KICK_LK]=7;
m_myDefineDamage[A_JUMP_SIDE_KICK_MK]=11;
m_myDefineDamage[A_JUMP_SIDE_KICK_HK]=15;

m_myDefineDamage[A_HADOKEN]=12;
m_myDefineDamage[A_SHORYUKEN]=19;
m_myDefineDamage[A_TORNADO_KICK]=5;

m_myDefineDamage[A_SHINKU_TORNADO_KICK]=5;
m_myDefineDamage[A_SHINKU_HADOKEN]=5;
m_myDefineDamage[A_METSU_SHORYUKEN]=5;

m_myDefineDamage[A_HADOKEN_L]=12;
m_myDefineDamage[A_HADOKEN_M_FL]=12;
m_myDefineDamage[A_HADOKEN_H_FM]=12;
m_myDefineDamage[A_HADOKEN_FH]=12;
m_myDefineDamage[A_HADOKEN_FAKE]=0;
```

```
m_myDefineDamage[A_SHINKU_HADOKEN_L]=5;
m_myDefineDamage[A_SHINKU_HADOKEN_M]=5;
m_myDefineDamage[A_SHINKU_HADOKEN_H]=5;

m_myDefineDamage[A_SHORYUKEN_L]=17;
m_myDefineDamage[A_SHORYUKEN_M]=18;
m_myDefineDamage[A_SHORYUKEN_H]=19;

m_myDefineDamage[A_T_KICK_L]=5;
m_myDefineDamage[A_T_KICK_M]=5;
m_myDefineDamage[A_T_KICK_H]=5;

m_myDefineDamage[A_T_KICK_AIR_L]=5;
m_myDefineDamage[A_T_KICK_AIR_M]=5;
m_myDefineDamage[A_T_KICK_AIR_H]=5;

m_myDefineDamage[A_SHINKU_T_KICK_L]=5;
m_myDefineDamage[A_SHINKU_T_KICK_M]=5;
m_myDefineDamage[A_SHINKU_T_KICK_H]=5;
```

ภาคผนวก จ เกี่ยวกับไพธอน

อินเทอร์เฟสฟังก์ชันที่สำคัญจากฝั่งเทสเบดที่จะให้สคริปต์ไพธอนเรียกใช้ได้นั้นถูกออกแบบไว้ 2 อันดังนี้

```
CPGetData(PyObject *args)
CPPressCustomCombo(PyObject *args)
```

สังเกตได้ว่าทุกฟังก์ชันจะรับพารามิเตอร์ชนิดเดียวกันหมดคือ `PyObject` ซึ่งเป็นไปตามข้อกำหนดของวิธีการใช้ไพธอน และก็เป็นรูปแบบมาตรฐานที่ภาษาสคริปต์นิยมใช้กัน คือจะห่อหุ้มพารามิเตอร์ทุกชนิดเป็นอ็อบเจกต์ชนิดเดียวกันหมด แล้วจึงค่อยไปแปลงกลับเอาว่าต้องการเปลี่ยนกลับให้อ็อบเจกต์นั้นมีค่าแบบตัวแปรชนิดใด หน้าที่การทำงานของทั้งสองฟังก์ชันเป็นดังนี้ ตัวแรกใช้เพื่อขอข้อมูลต่างๆ จากฝั่งเทสเบดเพื่อฝั่งสคริปต์จะได้รู้สถานการณ์ในเกมและตัดสินใจถูก ส่วนฟังก์ชันที่สองใช้เพื่อส่งคำสั่งกดปุ่มกลับไปควบคุมตัวละครในเกม วิธีการส่งพารามิเตอร์เวลาที่ทางฝั่งสคริปต์จะส่งอะไรไปก็จะส่งตามรูปแบบปรกติเลยโดยไม่ต้องสั่งห่อหุ้มเอง ตัวอย่างเช่นสมมุติฝั่งสคริปต์เรียกใช้แบบนี้

```
stz3.CPGetData(GET_POSX, STZ3_P1)
```

ซึ่งมีความหมายว่าจะขอข้อมูลตำแหน่งบนแกนระนาบ (`GET_POSX`) ของตัวละครฝ่ายผู้เล่นที่1 (`STZ3_P1`) เมื่อฟังก์ชัน `CPGetData` ทางฝั่งเทสเบดรับพารามิเตอร์เข้ามาก็จะทำการแปลงโดยใช้คำสั่ง `PyArg_ParseTuple(PyObject*, char*, ...)` ตัวอย่างในกรณีนี้เป็นดังนี้

```
PyObject* CPGetData(PyObject *args)
{
    int commandID; int objID;
    if(PyArg_ParseTuple(args, "ii", &commandID, &objID))
    {
      // Now commandID holds value GET_POSX
      // and objID holds value STZ_P1
    }
}
```

พารามิเตอร์ตัวแรกของ `PyArg_ParseTuple` คือ `PyObject*` ซึ่งคืออาร์เรย์ของอ็อบเจกต์ที่จะแปลงค่านั่นเอง พารามิเตอร์ตัวต่อไปเป็น `char*` ที่ใช้บอกรูปแบบในการแปลง ในที่นี้คือ “ii” ซึ่งหมายความว่าให้แปลงค่าอากิวเมนต์ตัวแรกและตัวที่สองเป็น integer ทั้งคู่ พารามิเตอร์ตัวที่เหลือต่อจากนี้คือพอยเตอร์ของตัวแปรที่จะมารับค่าหลังจากการแปลง ซึ่งในที่นี้ก็คือพอยเตอร์ของตัวแปรชื่อ `commandID` ที่จะมาใช้รับค่า `GET_POSX` และตัวแปร `objID` ที่จะมารับค่า `STZ3_P1` นั่นเอง เมื่อได้ค่ามาถูกต้องแล้วทางเทสเบดก็จะจัดหาค่าที่ต้องการดังกล่าวและเตรียมส่งกลับไปยังฝั่งไพธอน

ส่วนต่อไปนี้จะกว่างถึงวิธีการประกาศฟังก์ชันในฝั่งเทสเบดเพื่อให้สคริปต์เรียกใช้ได้จะต้องทำการดังนี้

1 เรียกใช้ฟังก์ชัน `Py_Initialize()` ซึ่งเป็นการประกาศเริ่มใช้ไพธอน ฟังก์ชันนี้จะมีให้ใช้ได้เมื่อทำการอินคลูดไพธอนไลบรารี่และแอดเฮดเดอร์เข้าไปไว้ในโปรเจคแล้ว

2 ประกาศสร้าง `object PyMethodDef` เพื่อเอาไว้เก็บอินเทอร์เฟสฟังก์ชันฝั่งเทสเบดให้ฝั่งสคริปตร์เรียกใช้ได้ วิธีการเซทค่าทำดังตัวอย่างนี้

```
PyMethodDef stz3Method;
m_stz3Method.ml_name = "CPPressPackageCommand";
m_stz3Method.ml_meth  = CPPressPackageCommand;
m_stz3Method.ml_flags = METH_VARARGS;
m_stz3Method.ml_doc   = "Call from C.";
```

`ml_name` เอาไว้ตั้งชื่อฟังก์ชันที่จะให้ฝั่งไพธอนรู้จักและเรียกใช้ได้ `ml_meth` เอาไว้ใส่ชื่อ `static` ฟังก์ชันทางฝั่งเทสเบดที่จะห้ไพธอนเรียกใช้ ในที่นี้ตั้งให้เป็นชื่อเดียวกันเพื่อความสมเหตุสมผล แต่ในความเป็นจริงชื่อฟังก์ชันในฝั่งเทสเบดอาจจะชื่อ ก แต่ประกาศให้ไพธอนรู้จักในนาม ข ก็ได้ ส่วนค่า `ml_flag` ให้ใส่เป็นค่าดังกล่าวเพื่อให้เป็นฟังก์ชันที่สามารถส่งพารามิเตอร์กันได้และสุดท้าย `ml_doc` เป็นตัวแปรที่เอาไว้ใส่คอมเมนต์หรือว่าวิธีการใช้

3 ทำการโหลดไฟล์สคริปต์ไพธอนที่จะรันโดยการเรียกใช้ฟังก์ชันทั้งสองนี้ตามลำดับ

```
Py_Object* m_pModule = PyImport_Import(“MyPyScript”);
Py_Object* m_pDict = PyModule_GetDict(m_pModule);
```

อันแรกคือนำเข้าไพธอนสริปต์รับพารามิเตอร์คือชื่อสคริปต์ที่จะรัน โปรดระวังว่าให้ส่งแต่ชื่อไฟล์เท่านั้นไม่ต้องส่งนามสกุลตามมาด้วย ผลที่รีเทิร์นมาจากฟังก์ชันแรกนี้ให้นำไปใช้เป็นพารามิเตอร์ของฟังก์ชันที่สอง เมื่อเรียกครบสองอันแล้วก็เสร็จสิ้นการเตรียมการ

4 ทำการเรียกใช้ฟังก์ชันจากในไพธอนสคริปต์ ทางฝั่งเทสเบดสามารถเรียกใช้ฟังก์ชันจากสคริปต์ได้โดยใช้ฟังก์ชันนี้

```
Py_Object* m_pFunc = PyDict_GetItemString(m_pDict, "Main");
Py_Object* m_pValue = PyObject_CallObject(m_pFunc, pArgs2);
```

อันแรกคือการส่ง `m_pDict` จากขั้นตอนข้างบนพร้อมกับชื่อฟังก์ชันในไฟล์สคริปต์ที่ต้องการรันเข้าไป ผลจะได้พอยเตอร์ของฟังก์ชันนั้นกลับออกมา แล้วจึงนำเอาพอยเตอร์นั้นไปเรียกใช้ ซึ่งหากต้องการส่งพารามิเตอร์อะไรเข้าไปก็ส่งเข้าไปทาง `Py_Object* pArgs2` ที่เป็น `Py_Object` ซึ่งมีวิธีการประกอบทำนองเดียวกับ `PyArg_ParseTuple` ที่ได้กล่าวไปแล้วข้างต้น

ประวัติผู้เขียนวิทยานิพนธ์

ผู้เขียนเกิดที่จังหวัดกรุงเทพมหานครอำเภอคลองสาน สำเร็จการศึกษาระดับปริญญาวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากภาควิชา วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัยในปีการศึกษา 2544 หลังจากทำงานแล้วได้เข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2548