เกม 3 มิติบนคอมพิวเตอร์แบบพกพา

3D GAME ON POCKET PC

นายจักษุเทพ ตี่กุล

นายจิรัชย์ ปริศวงศ์

นายจีระเดช กำลังเกื้อ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

เกม 3 มิติบนคอมพิวเตอร์แบบพกพา

3D GAME ON POCKET PC

โดย

นายจักษุเทพ ดีกุล

นายจิรัชย์ ปริศวงศ์

นายจีระเดช กำฉังเกื้อ

อาจารย์ที่ปรึกษา

ผศ. เกียรติกูล เจียรนัยธนะกิจ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

พ.ศ.2548

ปริญญานิพนธ์ปีการศึกษา 2548

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง เกม 3 มิติบนคอมพิวเตอร์แบบพกพา

3D GAME ON POCKET PC

ผู้จัดทำ

1. นายจักษุเทพ ตี้กุล รหัสนักศึกษา 45010106
2. นายจิรัชย์ ปริศวงศ์ รหัสนักศึกษา 45010126
3. นายจีระเดช กำลังเกื้อ รหัสนักศึกษา 45010133

อาจารย์ที่ปรึกษา

(ผศ. เกียรติกูล เจียรนัยธนะกิจ)

เกม 3 มิติบนคอมพิวเตอร์แบบพกพา

นายจักษุเทพ ตี้กุล 45010106
นายจิรัชย์ ปริศวงศ์ 45010126
นายจีระเดช กำลังเกื้อ 45010133
ผศ. เกียรติกูล เจียรนัยธนะกิจ อาจารย์ที่ปรึกษา
ปีการศึกษา 2548

บทคัดย่อ

ปัจจุบันวงการเกมคอมพิวเตอร์ได้มีการพัฒนาขึ้นมาก เพื่อตอบสนองกับความต้องการของผู้บริโภค ซึ่งขยายตัวมากขึ้นอันเนื่องมาจากราคาอุปกรณ์ที่ต่ำลงจนบุคคลทั่วไปสามารถมีกำลังซื้อได้ เกมคอมพิวเตอร์นั้นจึงเป็นทางเลือกหนึ่งในการพักผ่อนโดยใช้เครื่องคอมพิวเตอร์ ซึ่งปัจจุบันมีผู้พัฒนากันอย่างแพร่หลาย แต่ในส่วนของเกมสามมิติบนอุปกรณ์พกพาเช่นคอมพิวเตอร์พกพานั้นมีการพัฒนากันในวงแคบ วิทยานิพนธ์นี้เป็นการศึกษาวิธีการพัฒนาเกมสามมิติบนคอมพิวเตอร์พกพาให้มีประสิทธิภาพใกล้เคียงกับคอมพิวเตอร์ส่วนตัว ทั้งด้านกราฟิก การเล่นติดต่อผ่านเครือข่าย นอกจากนี้วิทยานิพนธ์ฉบับนี้ได้นำความรู้ด้านปัญญาประดิษฐ์มาประยุกต์ใช้ภายในเกมสามมิตินี้ด้วย

3D GAME ON POCKET PC

Mr. Jaksuthep Teekul	45010106
Mr. Jiruch Pritsawong	45010126
Mr. Jeeradech Kumlangkua	45010133
Mr. Kietikul Jiaranaitanakij	Advisor

Academic Year 2005

ABSTRACT

This thesis proposes to develop 3D Game on pocket PC. The main focuses are base on Software Develop, Applied A.I., 3D Graphic and Network Communication. Developing software is considered as well as Functioning 3D model. Pocket PC are chosen because their weight are lighter, their performance are increasing, and their cost are cheaper than the past.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้อย่างดี ด้วยคำแนะนำ และคำปรึกษาจาก ผศ. เกียรติกูล เจียรนัยธนะกิจ ซึ่งเป็นอาจารย์ผู้ควบคุมวิทยานิพนธ์ ข้าพเจ้ารู้สึกทราบซึ้งในความอนุเคราะห์จากท่านอาจารย์ และขอขอบพระคุณเป็นอย่างสูง

ขอขอบพระคุณอาจารย์เอนก กอธนสาร หรือพี่บอย ที่ได้ให้คำปรึกษาที่ดีต่างๆ มากมาย เปรียบได้กับเป็นพี่ชายที่คอยช่วยเหลือน้องๆ ตลอดมา

ขอกราบพระคุณคณาจารย์ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุก ๆ ท่านที่ได้ประสิทธิ์ประสาทวิชาให้กับข้าพเจ้า

ขอขอบคุณเพื่อนๆ พี่ๆ และน้องๆ ที่ห้องโปรเจคผู้น่ารักมากทุกคนที่ช่วยเป็นกำลังใจ และขอบคุณห้อง Hardware (ECC-811) ห้องโปรเจคที่ให้สถานที่สำหรับวิจัยและจัดทำวิทยานิพนธ์นี้

ขอขอบคุณเพื่อนๆ พี่ๆ น้องๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ สถาบันเทคโนโลยี พระจอมเกล้าเจ้าคุณทหารลาดกระบัง ทุกคนที่ให้คำแนะนำต่างๆ และคอยให้กำลังใจเสมอมา

ขอขอบคุณคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ที่ปลูกฝังความเป็นวิศวะลาดกระบังให้สู้งานหนัก มีใจสู้ไม่ถอย และเรียนรู้คุณสมบัติของวิศวกรที่ดี

สุดท้ายนี้ข้าพเจ้าขอกราบขอบพระคุณ บิดา มารดา และครอบครัวของข้าพเจ้าที่เป็นกำลังใจและให้การสนับสนุนในทุกเรื่องๆ ทำให้ข้าพเจ้าสามารถทำวิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี

คุณค่าและประโยชน์อันพึงมาจากวิทยานิพนธ์ฉบับนี้ ข้าพเจ้าขอมอบแด่ผู้มีพระคุณทุกท่าน

จักษุเทพ ตี้กุล
จิรัชย์ ปริศวงศ์
จีระเดช กำลังเกื้อ

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญรูป

สารบัญรูป (ต่อ)

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันตลาดของเกมบนพ็อกเก็ตพีซีเริ่มเติบโตขึ้น ซึ่งในอดีตเกมส่วนใหญ่บนพ็อกเก็ตพีซีจะเป็นเกมสองมิติเป็นส่วนใหญ่ เนื่องจากความสามารถอันจำกัดของพ็อกเก็ตพีซีในเรื่องการประมวลผลที่ยังไม่มีความเร็วมากนัก แต่ในปัจจุบันเมื่อมีการพัฒนาส่วนประมวลผลให้พัฒนาขึ้น ทำให้เกมสามมิติเริ่มมีวางขายกันในท้องตลาดมากขึ้น อย่างไรก็ตามเกมสามมิติที่มีการพัฒนาในปัจจุบันส่วนใหญ่มาจากชาวต่างชาติ เกมสามมิติจากฝีมือคนไทยยังมีไม่มากนัก

ปริญญานิพนธ์ฉบับนี้มาจากความตั้งใจพัฒนาเกมสามมิติบนพ็อกเก็ตพีซีด้วยฝีมือของนักศึกษาไทย โดยนำความรู้ด้านวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้ในชิ้นงานด้วย อาทิเช่น ความรู้ด้านคอมพิวเตอร์กราฟิกมาช่วยในการพัฒนาด้านกราฟิกภายในเกม, Genetic Algorithm ในส่วนของปัญญาประดิษฐ์มาใช้ในการพัฒนาวิวัฒนาการของตัวละครภายในเกม และนำความรู้ด้านเครือข่ายคอมพิวเตอร์มาช่วยให้โครงงานนี้มีการติดต่อระหว่างผู้เล่นด้วยกัน หรือระหว่างผู้เล่นกับเครื่องเซิร์ฟเวอร์ของระบบได้

1.2 วัตถุประสงค์ของโครงงาน

ปริญญานิพนธ์ฉบับนี้มุ่งหวังเพื่อศึกษาและพัฒนาโปรแกรมประยุกต์บนอุปกรณ์พ็อกเก็ตพีซีที่มีการนำความรู้ด้านวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้โดยนำเสนอในรูปแบบเกมสามมิติที่มีการเชื่อมต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์ ด้วยฝีมือของนักศึกษาไทย

ความรู้ด้านวิศวกรรมคอมพิวเตอร์ที่นำมาใช้ในโครงงาน ได้แก่ ความรู้ด้านการออกแบบซอฟต์แวร์เชิงวัตถุ(Object Oriented Software Engineering), ความรู้ด้านคอมพิวเตอร์เครือข่าย, ความรู้ด้านโปรแกรมมิ่ง, ความรู้ด้านปัญญาประดิษฐ์

1.3 ขอบเขตของโครงงาน

ในปริญญายานิพนธ์ฉบับนี้ได้นำเสนอวิธีการออกแบบและพัฒนาโปรแกรมประยุกต์บนอุปกรณ์คอมพิวเตอร์แบบพกพา(Pocket PC) ที่มีการนำความรู้ด้านวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้ โดยนำเสนอในรูปแบบของเกมสามมิติ

เกมสามมิติในโครงงานนี้อยู่ในระดับของเกมทดลองเล่น(Demo Version) โดยการทำงานของระบบครอบคลุมฟังก์ชันการทำงานพื้นฐานทั้งหมด แต่ยังขาดรายละเอียดปลีกย่อยต่างๆ เช่น ในส่วนของโมเดลตัวละครภายในเกม, กราฟิกในฉากต่อสู้ที่ยังไม่เร้าใจ

1.4 วิธีการดำเนินการ

1. ศึกษาเกี่ยวกับเกมเอนจิ้นท์สำหรับพัฒนากราฟิกสามมิติบนอุปกรณ์คอมพิวเตอร์แบบพกพา(Pocket PC)
2. ศึกษาการสร้างโมเดล 3 มิติ และการสร้างการเคลื่อนให้กับตัวละคร 3 มิติ
3. ศึกษาเกี่ยวกับเครื่องมือ(IDE Tool) สำหรับพัฒนาโปรแกรมประยุกต์บนอุปกรณ์คอมพิวเตอร์แบบพกพา(Pocket PC)
4. ศึกษาทฤษฎีการออกแบบเกม
5. วิเคราะห์ และออกแบบระบบ
6. จัดเตรียมวัตถุดิบที่จะนำมาใช้ในการพัฒนา อาทิเช่น โมเดล 3 มิติ, ฉาก 3 มิติ
7. ศึกษาความรู้ด้านปัญญาประดิษฐ์ในเรื่องที่จะนำมาประยุกต์ใช้ในเกมสามมิติ
8. ศึกษาการเชื่อมต่อผ่านเครือข่ายระหว่างเครื่องคอมพิวเตอร์แบบพกพากับเครื่องคอมพิวเตอร์แบบพกพา และการเชื่อมต่อระหว่างเครื่องคอมพิวเตอร์แบบพกพากับเครื่องคอมพิวเตอร์
9. พัฒนาโปรแกรมประยุกต์ในรูปแบบเกมสามมิติบนอุปกรณ์คอมพิวเตอร์แบบพกพา
10. ทดสอบระบบและปรับปรุงแก้ไขส่วนผิดพลาด
11. สรุปผลโครงงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้รับความรู้ความเข้าใจเกี่ยวกับกระบวนการการพัฒนาซอฟต์แวร์อย่างเป็นระบบ
2. ได้รับความรู้ความเข้าใจเกี่ยวกับการออกแบบเกม
3. ได้รับความรู้ความเข้าใจเกี่ยวกับเกมเอนจิ้นท์บนอุปกรณ์คอมพิวเตอร์แบบพกพา
4. ได้รับความรู้ความเข้าใจเกี่ยวกับการสร้างโมเดล 3 มิติ
5. ได้รับความรู้ความเข้าใจเกี่ยวกับการสร้างการเคลื่อนไหวให้กับตัวละคร 3 มิติ
6. ได้รับความรู้ความเข้าใจเกี่ยวกับการพัฒนาโปรแกรมประยุกต์และการทำงานบนอุปกรณ์คอมพิวเตอร์แบบพกพา
7. ได้รับความรู้ความเข้าใจเกี่ยวกับการเชื่อมต่อผ่านเครือข่ายคอมพิวเตอร์ของเครื่องคอมพิวเตอร์และเครื่องคอมพิวเตอร์แบบพกพา

8. โปรแกรมประยุกต์บนอุปกรณ์คอมพิวเตอร์แบบพกพาที่มีการนำความรู้ด้านวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้และมีการเชื่อมต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์ในรูปแบบของเกมสามมิติ

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาออกเป็น 5 บทด้วยกันคือ

บทที่ 1 กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 กล่าวถึงทฤษฎีพื้นฐานที่ใช้ในโครงงาน ซึ่งประกอบด้วยทฤษฎีการสร้างโมเดล 3 มิติ(Computer Graphic), ทฤษฎีปัญญาประดิษฐ์ในเรื่อง Genetic Algorithm, ทฤษฎีการเชื่อมต่อบนโปรโตคอล TCP/IP, ทฤษฎีเครือข่ายไร้สาย Wi-Fi และทฤษฎีในเรื่องของ Winsock

บทที่ 3 กล่าวถึงการออกแบบโครงงานนี้ ซึ่งประกอบด้วยการออกแบบรายละเอียด กฎกติกาของเกม, การออกแบบอัลกอริทึมที่ใช้ในการต่อสู้, การออกแบบเกม, การออกแบบสถาปัตยกรรมของระบบ และนำความรู้ที่ได้จากทฤษฎีในบทที่ 2 มาออกแบบประยุกต์แล้วนำไปใช้ในการพัฒนาโครงงานนี้

บทที่ 4 กล่าวถึงชิ้นงานของโครงงานนี้ ซึ่งประกอบด้วยรายละเอียดของการแสดงผลของเกม, ฟังก์ชันต่างๆ ภายในเกม, ความสามารถของระบบ (การเจริญเติบโตของตัวละคร, การผสมพันธุ์ของตัวละคร, การเชื่อมต่อระหว่างอุปกรณ์) โดยนำเสนอด้วยภาพที่ได้จากการจับภาพมาจากหน้าจอเครื่องคอมพิวเตอร์แบบพกพาที่เล่นเกมนี้จริงๆ

บทที่ 5 กล่าวถึงการทดลองและผลการทดลอง ประกอบไปด้วยการทดลองเล่นเกมในฟังก์ชันต่างๆ ในส่วนของการเล่นบนเครื่องคอมพิวเตอร์แบบพกพาคนเดียว, การทดลองเล่นเกมในส่วนของการติดต่อระหว่างผู้เล่นด้วยกัน และการทดลองเล่นเกมติดต่อกับเครื่องเซิร์ฟเวอร์

บทที่ 6 เป็นบทวิจารณ์และสรุป ซึ่งกล่าวถึงบทสรุปของโครงงาน วิจารณ์สิ่งที่ได้รับจากโครงงาน ปัญหาอุปสรรคพร้อมแนวทางแก้ไข และข้อเสนอแนะสำหรับเป็นแนวทางในการพัฒนาต่อ

บทที่ 2
ทฤษฎีพื้นฐานที่ใช้ในโครงงาน

2.1 ทฤษฎีทางด้านกราฟิกสามมิติ

3D computer Graphic, มีพื้นฐานบน Vector Graphics แทนที่คอมพิวเตอร์จะเก็บข่าวสารเกี่ยวกับจุด, เส้นและเส้นโค้งบนระนาบ 2 มิติ ปัจจุบันคอมพิวเตอร์จะเก็บตำแหน่งของจุด, เส้นและพื้นผิวหน้าของวัตถุ สำหรับสร้างรูปโพลีกอนในระบบ 3 มิติ แทน ดังนั้น เอนจิ้น 3D ส่วนใหญ่จะทำหน้าที่พื้นฐานคือเก็บจุด (3 จุดสำหรับแต่ละตำแหน่ง) เส้นซึ่งเชื่อมต่อจุดเหล่านั้นด้วยกัน, และใบหน้าระหว่างเส้น, ครั้นแล้วผลที่ตามกันมาของใบหน้าเพื่อสร้าง วัตถุ 3D ทุกวันนี้ การแสดงผลไม่เพียงแต่สร้างวัตถุในหลายๆรูปร่างยังให้ผลในการสร้างเงา สร้างพื้นผิวและการ Rasterization ที่จะกล่าวถึงภายหลัง

2.1.1 Shading

กระบวนการทำ shading (ที่เกี่ยวกับวิชา 3D computer graphics) จะกล่าวถึงการจำลองวัตถุให้เหมือนจริง(ส่วนหนึ่งเกี่ยวข้องกับความเร็วในการจำลองวัตถุด้วย) วิธีทำให้ผิวหน้าของวัตถุดูเหมือนจริง แหล่งกำเนิดแสงเสมือน ค่าที่คำนวณได้นั้นไม่เพียงขึ้นอยู่กับความละเอียดของข้อมูลหรือหรืออุปกรณ์ราคาแพงแต่เพียงอย่างเดียวยังขึ้นกับอยู่เทคนิคต่างๆต่อไปนี้

- Flat shading:
- Gouraud shading:
- Texture mapping: เป็นวิธีที่ทำให้เห็นรายละเอียดของพื้นผิวโดยที่ไม่เป็นภาระให้คอมพิวเตอร์มากนัก
- Phong shading: คิดค้นโดยนาย Bui Tuong Phong, เป็นวิธีการทำพื้นผิวให้เรียบอีกวิธีหนึ่ง
- Bump mapping: คิดค้นโดย Jim Blinn, สำหรับการสร้างพื้นผิวที่มีลักษณะขรุขระ หรือมีรอยยับได้
- Ray Tracing: เป็นวิธีที่พัฒนามาจากเอกลักษณ์ของแสงที่ทำให้เกิดการสะท้อนและหักเห
- Radiosity: เป็นวิธีสร้างแสงให้เหมือนจริง โดยคิดทั้งฉาก

2.1.2 Texturing

พื้นผิวของวัตถุประกอบด้วยข้อมูลที่แสดงถึงสี และข้อมูลนั้นอาจต้องแสดงถึงสีภาพเขียน น้ำมัน หรือภาพสองมิติอื่นๆ วิธีการเก็บและแสดงนั้นคือการทำ Texturing Textures ทำให้เราสามารถเลือกได้ว่าวัตถุสุดท้ายจะออกมามีรูปร่างหน้าตาอย่างไรหลังจากการทำ shading ขึ้นอยู่กับเทคนิคที่เลือกใช้และความเปลี่ยนแปลงของรูประหว่างการทำ shading

Flat shading: เป็นเทคนิคว่าด้วยเรื่องการใส่ผิววัตถุโดยอ้างอิง normal vector ในการแสดงผลทั้งตำแหน่งและความเข้มแสงของแหล่งกำเนิดแสง ด้วยการคำนวณตำแหน่งของแหล่งกำเนิดแสงโดยทั่วไปแล้วจะใช้ในงานที่ต้องการความรวดเร็วในการประมวลผล หรือ การทำเทคนิคอื่นนั้นเปลืองการคำนวณมากเกินไป ข้อเสียของการใช้วิธีนี้คือจะให้รูปเป็น Low-polygon ที่ดูแบ่งเป็นชั้นๆ แต่

FLAT SHADING PHONG SHADING

บางครั้งก็เป็นที่ต้องการในการสร้างวัตถุที่มีลักษณะเป็นลูกบาศก์

รูปที่ 2.1 ตัวอย่างการทำ Flat shading กับวิธีอื่น

Gouraud shading: คิดค้นโดยนาย Henri Gouraud ในปี ค.ศ. 1971, เป็นวิธีที่คิดขึ้นเพื่อเพิ่มความเร็วและความนุ่มของผิววัตถุ วิธีการนี้จะลดความเป็น Low-polygon โดยไม่ต้องเพิ่มการประมวลผลแต่ละ pixel มากนัก

วิธีการนี้จะคำนวณ Normal vector ที่แต่ละจุดของ Polygon ด้วยการหาค่าเฉลี่ยจาก Normal Vector ของพื้นผิวที่อยู่รอบตัวจุดนั้นแสงและความเข้มแสงก็ใช้เหมือน Flat shading เพียงแต่เพิ่มวิธีการ interpolated หรือการเกลี่ยสีที่เส้นขอบให้เข้ากันกับพื้นผิวข้างเคียง

วิธี Gouraud shading นี้ใช้กระบวนการคิดน้อยกว่า Phong shading แต่ไม่สามาแสดงแสงที่ต้องการทั้งหมดได้ ตัวอย่างที่เห็นได้ชัดเจนคือจุดสว่างสีขาวที่สะท้อนต้นกำเนิดแสงจะไม่สามารถแสดงได้ละเอียดพอขึ้นอยู่กับ normal vector ที่จุดนั้นว่าชี้ไปที่แหล่งกำเนิดแสงหรือไม่ ถ้าไม่จุดสว่างนั้นก็จะไม่แสดงผลให้เห็นแล้วทำให้ดูเหมือนว่าจุดกำเนิดแสงมีการเปลี่ยนที่ ผลกระทบนี้จะส่งผลให้เห็นเมื่อแหล่งกำเนิดแสงมีการเคลื่อนที่ รูปที่สะท้อนจากวัตถุจะมีลักษณะการเคลื่อนไหวที่ไม่ราบเรียบ

Texture mapping เป็นวิธีที่ใช้เพิ่มความเหมือนจริงเข้าไปในวัตถุด้วยวิธีการใส่ภพเข้าไปในรูปปกติที่มีรายละเอียดน้อยให้ดูเหมือนมีความละเอียดถึงขั้นระดับ pixel ตามรูปที่ใส่เข้าไปนั้น วิธีการนี้ช่วยให้กรคำนวณลดลงอย่างมาก เช่นการทำส่วนหัวของมนุษย์ก็จะให้ส่วนวัตถุที่มีความราบเรียบพอเพียงที่ระบบจะรับได้ในบางครั้งอาจเป็นแค่วัตถุทรงกลม หลังจากนั้นก็ใสภาพหน้าตาเข้าไปทำให้ไม่ต้องสร้างจมูก หรือตาอย่างละเอียด

ด้วยการพึ่งอุปกรณ์ฮาร์ดแวร์เช่น การ์ดจอที่มีความสามารถสูง นั้นทำให้วิธีนี้มีความจำเป็นลดน้อยลง อย่างไรก็ตามวิธีการนี้ยังคงใช้กันอย่างแพร่หลาย

รูปที่ 2.2 การใส่รูปลงไปในวัตถุ

Phong shading: หรือ reflection model เป็นโมเดลที่มีความเหมือนจริงมากที่สุดของระบบ 3 มิติ ที่รวมความลักษณะของพื้นผิวต่างๆ ความขรุขระหรือความสะท้อนแสง และบริเวณโดยรอบของวัตถุ ไว้ในแต่ล่ะจุดของพื้นผิววัตถุ โดยมีข้อตกลงกันว่า ต้นกำเนิดแสงทุกชนิดต้องเป็นจุด การคำนวณนี้เฉพาะกับพื้นผิวเท่านั้น สีที่สะท้อนมาต้องเป็นสีเดียวกับต้นกำเนิดแสง สำหรับค่าความขรุขระหรือความสะท้อนใช้ได้กับวัตถุชิ้นนั้นเท่านั้น แต่ค่าสภาพแวดล้อมใช้ได้กับทั้งพื้นที่

Diffuse ค่าความขรุขระ,

$$Id = Ii\ kd\sin\theta \quad \text{เมื่อ } 0 \leq \theta \leq 2\pi \tag{2.1}$$

เมื่อ Ii คือค่าความสว่างของแหล่งกำเนิดแสงที่เป็นจุด

θ เป็นมุมระหว่างพื้นผิวและต้นกำเนิดแสง

kd คือสัมประสิทธิ์การสะท้อน

สูตรการคำนวณที่มีต้นกำเนิดแสงหลายตัว

$$I_d = k_d \sum_n I_{i,n}(L_n \cdot N) \tag{2.2}$$

เมื่อ L และ N เป็นเวกเตอร์หนึ่งหน่วย

L_n เป็นทิศทางของแสงที่แสงที่ต้นกำเนิดแสงที่ n ที่ฉายมายังพื้นผิว

Specular ค่าความสะท้อนแสง

$$I_s = I_i k_s \cos^n \Omega = I_i k_s (R \cdot V)^n \tag{2.3}$$

เมื่อ n แทนค่าความสะท้อนแสงของพื้นผิว ถ้ามีค่าเป็นอนันต์จะแทนกระจก

Ω เป็นมุมระหว่างกระจกกับผู้สังเกต

V เป็นทิศทางของมุมมองปกติ

Ambient สภาพแวดล้อม

$I_g = I_a k_a$ เป็นค่าทั่วไปที่กำหนดขึ้น ส่วนใหญ่แล้วเป็นค่าคงที่

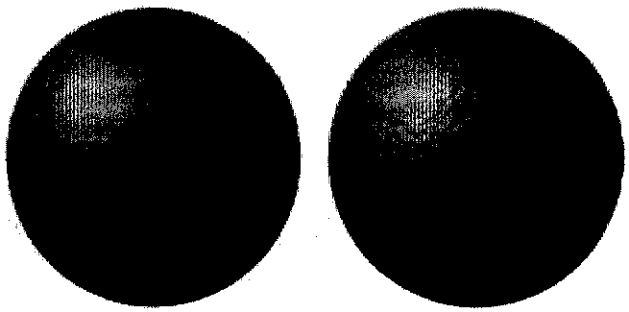
เมื่อรวม 3 สมการด้วยกัน

$$I = I_a k_a + I_i(k_d(L \cdot N) + k_s(R \cdot V)^n) \tag{2.4}$$

ในที่นี้ยังไม่ได้คิดความจางลงของแสงเมื่อผ่านระยะทางแต่สามารถเพิ่มเติมได้ภายหลัง

รูปที่ 2.3 ภาพอธิบายสมการของ Phong

Bump mapping เป็นเทคนิคที่ทำให้ normal vector ของแต่ละจุดมีความยุ่งเหยิง ก่อนที่จะทำการคำนวณเทคนิคอื่นต่อไป เป็นผลให้พื้นผิวมีรายละเอียดเพิ่มขึ้นกลายเป็นผิวขรุขระเหมือนวัตถุจริงๆ

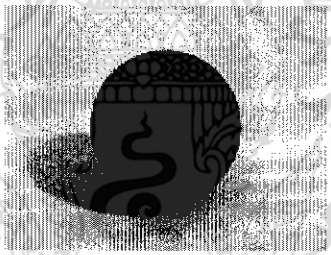


รูปที่ 2.4 ภาพหลังจากการทำ Bump mapping

Ray Tracing เป็นวิธีที่ได้จากการศึกษาเส้นทางเดินของแสงที่กระทำต่อวัตถุชนิดต่างๆ โดยเริ่มจากจุดที่ตามองไปยังต้นกำเนิดแสง

ข้อด้อยของวิธีนี้คือความเร็วในการคำนวณ ขณะที่วิธีอื่นๆนั้นสามารถแบ่งข้อมูลของแต่ละจุดกันได้แต่วิธี Ray tracing จะคิดที่แต่ละจุดแยกกัน อย่างไรก็ดีข้อด้อยนี้ก็อาจนับเป็นจุดเด่นได้ถ้าต้องการแต่งคุณภาพของรูปเช่น เพิ่มเส้นทางเดินของแสงให้มากขึ้นเพื่อแก้ปัญหา anti-aliasing

Radiosity เป็นเทคนิคที่จะคิดระบบแสงโดยรวมทั้งระบบ จากตำแหน่งที่แหล่งกำเนิดแสงอยู่ไปยังจุดต่างๆของระบบ จะทำให้ได้ค่าของแสงที่ตำแหน่งนั้นๆ แล้วเกิดเป็นเงา



รูปที่ 2.5 ภาพที่เกิดจากการทำ Radiosity

3D projection

3D projection คือการเปลี่ยนมุมมองของวัตถุในรูป 3 มิติ ให้เป็น 2 มิติ โดยใช้การคำนวณทางคณิตศาสตร์ โดย ข้อมูลจะถูกเก็บเป็นจุดโดยแทนค่าเป็น X, Y, Z โดยนับจากจุดศูนย์กลางจากจุดเชื่อมกันเป็นสามเหลี่ยม และการอ้างอิงการหมุนด้วยมุมทำให้อ้างอิงตำแหน่งจริงๆได้

ผู้สังเกต อาจเรียกว่า กล้อง ก็ได้ กล้องมีพิกัด X, Y, Z 2 ชุด ประกอบด้วยชุดบอกตำแหน่งของกล้อง และ ชุดบอกทิศทางที่กล้องมองอยู่ ข้อมูลเหล่านี้จะเก็บอยู่ในรูปของ floating point แต่บางโปรแกรมก็ยังคงมีการเปลี่ยนเป็น integer เพื่อให้การคำนวณเร็วขึ้น

Mathematical tools

เครื่องมือที่ใช้จะเป็นการใช้เมทริกซ์ขนาด 4×4 และฟังก์ชั่นตรีโกณ แต่ละขั้นตอนจะอาศัยการคูณกันของเมทริกซ์ โดยมีข้อกำหนดว่า เมทริกซ์จะต้องคูณกันตามลำดับการทำ โดยที่เมทริกซ์ที่ทำก่อนจะอยู่หลังสุด

ขั้นที่ 1: World transform

ขั้นแรกนี้เป็นการเปลี่ยนตำแหน่งของจุดหรือเปลี่ยนมุมของจุดรอบแกนต่างๆ สามารถทำได้ด้วย เมทริกซ์ 4 เมทริกซ์ต่อไปนี้

$$\begin{bmatrix} 1 & 0 & 0 & x \\ 0 & 1 & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — การเคลื่อนย้ายวัตถุ

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — การหมุนวัตถุรอบแกน X

$$\begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — การหมุนวัตถุรอบแกน Y

$$\begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — การหมุนวัตถุรอบแกน Z

หลังจากที่ได้เมทริกซ์ทั้ง4 แล้วนำมาคูณกันจะได้เมทริกซ์ของ world transform ซึ่งไม่ว่าจุดใดๆ คูณกับ เมทริกซ์นี้แล้วจะได้เหมือนกับการเลื่อนตำแหน่งเปลี่ยนมุม ตามที่เราอ้างอิงในระบบ

ลำดับของการคูณเมทริกซ์นั้นสำคัญ การเปลี่ยนลำดับนั้นจะทำให้ผลที่ได้ย่อมเปลี่ยนไป ถ้าต้องการหมุนวัตถุต้องทำให้แน่ใจว่า วัตถุอยู่ที่จุดศูนย์กลางแล้วจึงหมุนแล้วค่อยเลื่อนตำแหน่งไปตำแหน่งที่ต้องการเพราะว่าเมทริกซ์การหมุนดังกล่าวเป็นการหมุนรอบแกนที่จุดศูนย์กลาง หากวัตถุไม่อยู่ที่ศูนย์กลางก็จะกลายเป็นการเหวี่ยงวัตถุโดยที่วัตถุไม่หมุนรอบตัวเองและเปลี่ยนตำแหน่งไป และการหมุนวัตถุใดต้องเป็นไปตาม ข้อกำหนดนี้เสมอ

อีกเมทริกซ์หนึ่งที่ช่วยในการทำ Transform คือเมทริกซ์ย่อขยาย กำหนดให้ SX, SY, SZ เป็นขนาดในการย่อขยายในแนวแกนต่างๆ จะสามารถเขียนได้ดังนี้

$$\begin{bmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — เมทริกซ์ย่อขยาย

เมทริกซ์นี้เหมือนกับเมทริกซ์อื่นๆที่ทำการเปลี่ยนแปลงโดยยึดจุดศูนย์กลางของพิกัด (X=0, Y=0, Z=0) เป็นหลัก การจะย่อขยาย วัตถุใดๆนั้นต้องให้แน่ใจก่อนว่า วัตถุต้องอยู่ที่จุดศูนย์กลางก่อน จึงจะทำการย่อขยายได้แล้วจึงค่อยย้ายไปตำแหน่งที่ต้องการ

ขั้นที่ 2: Camera transform

ขั้นนี้มีความคลายกับขั้นที่ 1 เพียงแต่เมทริกซ์นี้ต้องอาศัย ค่า 6 ค่าที่แสดงตำแหน่งของผู้สังเกต(กล้อง) โดยเริ่มต้นที่จุดศูนย์กลางหันไปทางแกน Z การเลื่อนตำแหน่งทำได้โดยเมทริกซ์ต่อไปนี้โดยกำหนด Z คือค่าความลึก X คือค่าที่ชี้ไปทางซ้าย Y คือค่าที่ชี้ขึ้นข้างบน

$$\begin{bmatrix} 1 & 0 & 0 & -x \\ 0 & 1 & 0 & -y \\ 0 & 0 & 1 & -z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — อินเวอร์สเมทริกซ์ที่ใช้ย้ายตำแหน่งกล้อง

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — อินเวอร์สเมทริกซ์การหมุนรอบแกน X

$$\begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — อินเวอร์สเมทริกซ์การหมุนรอบแกน Y

$$\begin{bmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ — อินเวอร์สเมทริกซ์การหมุนรอบแกน Z

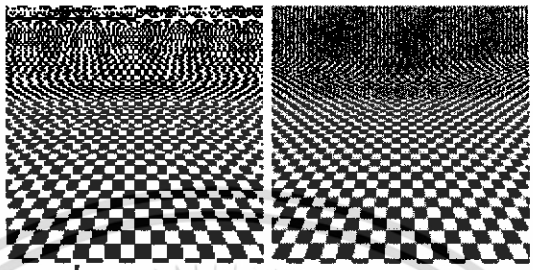
เมทริกซ์ในขั้นที่ 2 นี้สามารถคูณกันได้พิกัดที่อ้างอิงจากผู้สังเกตได้ และจากความรู้ที่ว่า

$(A\times B)^{-1}=B^{-1}\times A^{-1}$ เราจะได้

Camera transform = inverse rotation × inverse translation

Transform = camera transform × world transform

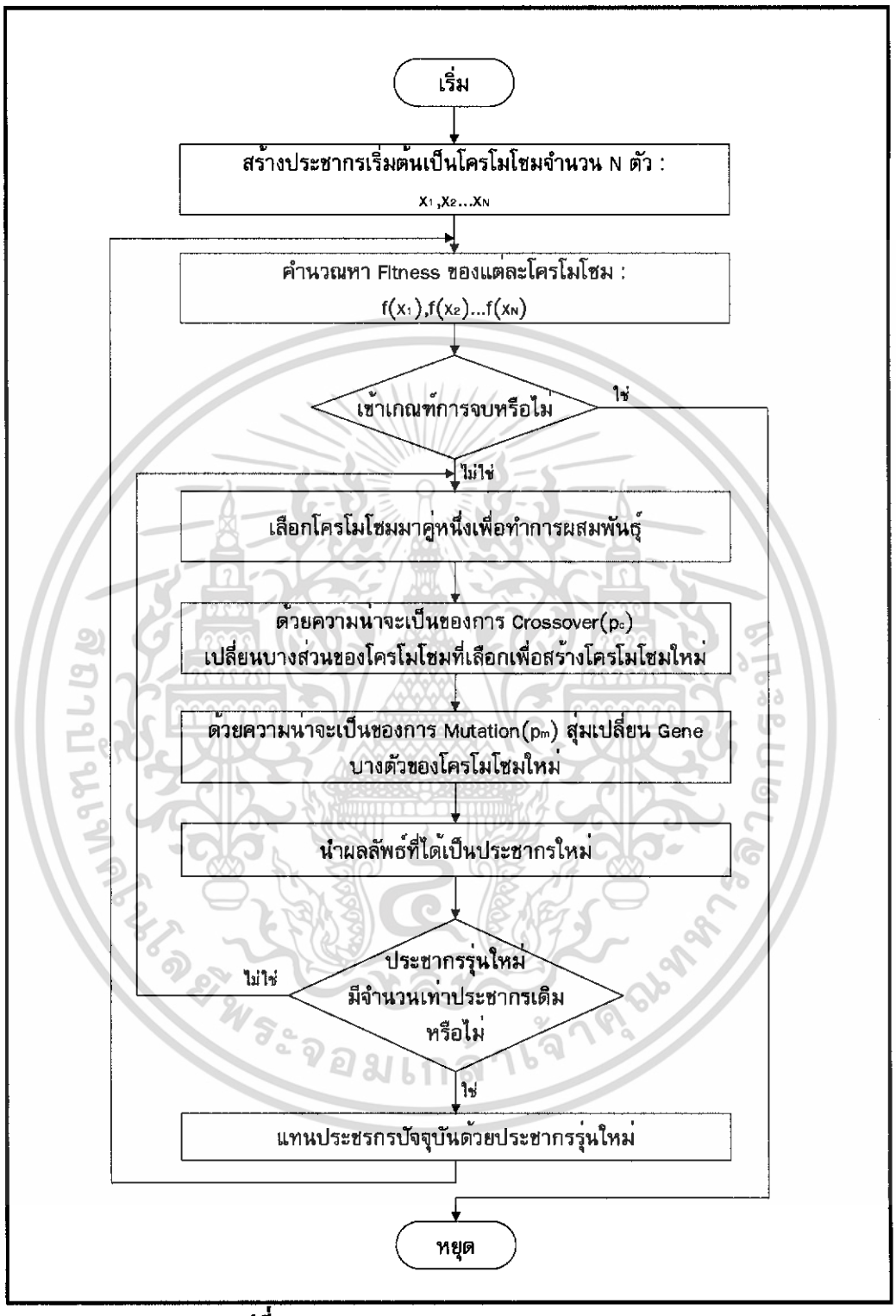
Anti-aliasing



รูปที่ 2.6 (a) aliasing (b) anti-aliasing

Aliasing คือการที่ความละเอียดของจุดไม่เพียงพอทำให้เกิดภาพเส้นตรงที่เหมือนเส้นขาดๆหายๆดังภาพ (a) อันเนื่องมาจากสาเหตุที่ภาพความละเอียดสูงเกินที่อุปกรณ์แสดงผลจะรับได้ การแก้ปัญหานี้ทำได้โดยเปลี่ยนบางจุดให้เป็นสีเทาจะได้ผลลัพธ์เป็นภาพ (b)

2.2 ทฤษฎีทางด้าน Genetic Algorithm



รูปที่ 2.7 Flow chart ของ Genetic Algorithm

กระบวนการ Genetic Algorithm

ขั้นที่ 1 : ใช้ปัญหากำหนดโครโมโซมที่ความยาวเท่ากัน กำหนดจำนวนของโครโมโซม ความน่าจะเป็นในการ Cross Over กำหนดความน่าจะเป็นในการ Mutation

Crossover probability – ในขั้นตอนกระบวนการ crossover ถ้าหากไม่มีการ crossover เกิดขึ้น รุ่นลูกจะมีลักษณะเหมือนพ่อแม่ แต่หากเกิดการ crossover ลูกหลานใหม่จะถูกประกอบขึ้นจากโครโมโซมของพ่อแม่ทำให้มีการเปลี่ยนแปลงไปจากรุ่นเดิม ถ้าค่า crossover probability เป็น 100% แล้วหมายความลูกที่ได้เกิดจากการ crossover ทุก Bit แต่ถ้าค่า crossover probability มีค่าเป็น 0% รุ่นใหม่ที่กำเนิดมาจะมีโครโมโซมเช่นเดียวกับประชากรเก่า กระบวนการ crossover มีขึ้นเพื่อให้ส่วนดีของโครโมโซมเก่ายังมีการสืบทอดอยู่และทำให้รุ่นใหม่มีประสิทธิภาพดีขึ้น ดังนั้นจึงเป็นการดีที่จะให้มีการหลงเหลือของประชากรเก่าไว้ไปยังรุ่นต่อไป

Mutation probability – ในขั้นตอนการ Mutation หากไม่มีการ mutation เกิดขึ้น ลูกหลานที่ได้จะเกิดมาหลังจากการ crossover โดยไม่มีการเปลี่ยนแปลง บางกรณีที่ GA ไม่สามารถแก้ปัญหาได้ถูกต้อง เช่น ปัญหา Hill climbing ที่อาจเจอ Local Maxima, Plateau, และ Shoulder(สันเขา)โดยมีการ mutation ขึ้น ทำให้ส่วนใดส่วนหนึ่งของโครโมโซมจะมีการเปลี่ยนแปลงทำให้ดูเหมือนมีการเปลี่ยนแบบก้าวกระโดด

ขั้นที่ 2 : นิยาม Fitness Function เพื่อจัดลำดับความสามารถในการดำรงอยู่โดยดูจากปัญหาที่ต้องการแก้

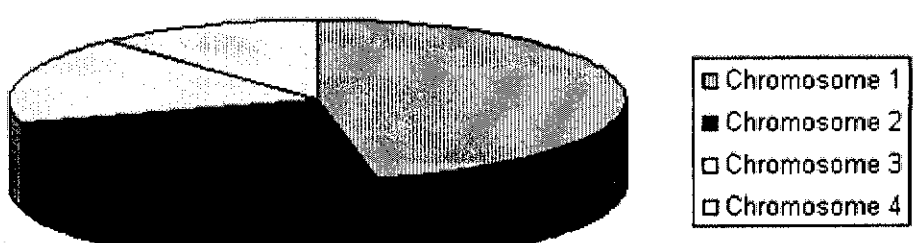
ขั้นที่ 3 : สุ่มสร้างประชากรจำนวน N ตัว X1, X2…Xn

ขั้นที่ 4 : คำนวณค่า Fitness Value f(X1), f(X2)…f(Xn)

ขั้นที่ 5 : เลือกคู่ของโครโมโซม ตามค่า Fitness Value มีวิธีดังต่อไปนี้

1. วิธี Roulette Wheel Selection

กระบวนการเลือกพ่อแม่นั้นขึ้นอยู่กับค่าความเหมาะสม (fitness) โครโมโซมที่ดีย่อมได้รับโอกาสที่จะถูกเลือกมาก หากเรานึกถึงวงล้อรูเล็ต (roulette wheel) ที่มีโครโมโซมทั้งหมดวางอยู่ ขนาดของส่วนใน roulette wheel จะขึ้นอยู่กับค่าความเหมาะสม (fitness) ของโครโมโซมนั้น หากพื้นที่มีขนาดใหญ่แสดงว่ามีค่าความเหมาะสมมากดังเห็นได้จากภาพด้านล่าง



รูปที่ 2.8 จำลองการแบ่งพื้นที่โดยใช้วิธี roulette wheel

เช่นเดียวกับเวลาเราเล่นรูเล็ต ลูกกลิ้งจะถูกใส่ลงไปในวงล้อรูเล็ตถ้าลูกกลิ้งหยุด ณ ตำแหน่งใดโครโมโซมนั้นจะถูกเลือกมาเป็นพ่อแม่ให้รุ่นต่อไป

กระบวนการนี้มีอัลกอริทึมในการทำงานดังต่อไปนี้

1. [Sum] คำนวณหาผลรวมของค่าความเหมาะสมของโครโมโซมทั้งหมด -sum S

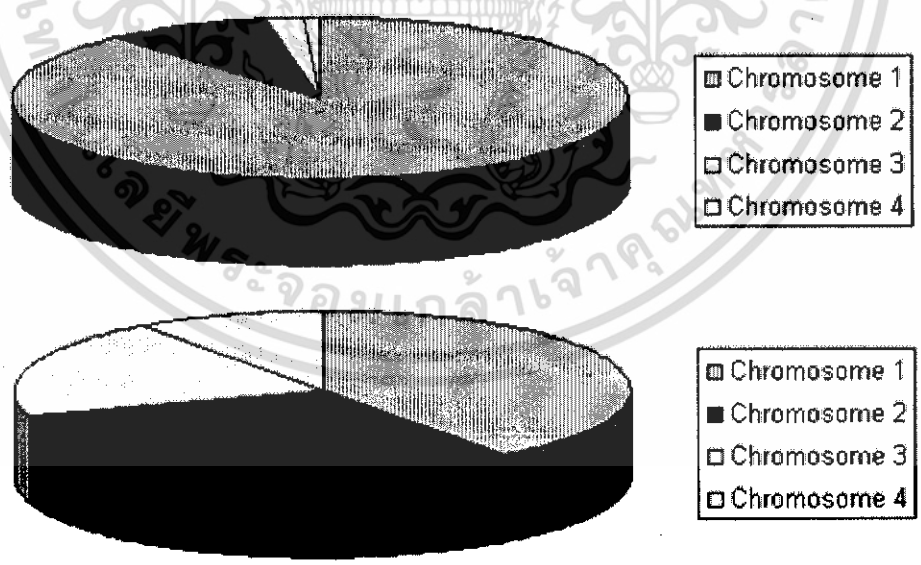
2. [Select] สุ่มตัวเลขเพื่อหาช่วงที่ต้องการในการถูกเลือก (0,S) – r

3. [Loop] เคลื่อนที่ผ่านประชากรและรวมค่าความเหมาะสมไปเรื่อย ๆ จนเมื่อผลรวม s มากว่าค่า r คืนค่าโครโมโซมที่อยู่ที่ตำแหน่งนั้นออกมา

2. วิธี rank Selection

วิธีการ roulette wheel selection นั้นจะเกิดปัญหาขึ้นถ้ามีความแตกต่างของค่าความเหมาะสมมาก ยกตัวอย่างเช่นหากโคโมโซมที่ดีที่สุดมีค่าความเหมาะสมคิดเป็น 90% ของค่าผลรวมของค่าความเหมาะสมทั้งหมด โครโมโซมอื่นนั้นจะมีโอกาสน้อยมากที่จะถูกเลือก

วิธี Rank selection จะทำการจัดอันดับโครโมโซมโดยที่โครโมโซมที่แย่ที่สุดจะมีค่าความเหมาะเป็น 1 โครโมโซมที่แย่รองลงมาจะมีค่าเป็น 2 และโครโมโซมที่ดีที่สุดจะมีค่าเป็น n (n เป็นจำนวนของโครโมโซมทั้งหมด) จากภาพด้านล่างจะเห็นความแตกต่างของส่วนของพื้นที่เมื่อใช้วิธีที่แตกต่างกัน



รูปที่ 2.9 จำลองการแบ่งพื้นที่โดยใช้วิธี roulette wheel เทียบกับวิธี Rank selection

ดังนั้นทุกโครโมโซมจะมีโอกาสในการถูกเลือกพอ ๆ กัน อย่างไรก็ตามวิธีนี้จะทำให้กระบวนในการหาคำตอบช้าลงเนื่องจากวิธีที่ดีที่สุดมีโอกาสที่จะถูกเลือกพอ ๆ กับวิธีอื่น ๆ

3. วิธี Steady-State Selection

วิธีการนี้เป็นวิธีการพิเศษในการเลือกพ่อแม่โดยมีแนวคิดที่ว่าชนิดที่ถูกเลือกของประชากรใหม่จะเป็นส่วนใหญ่ของผู้รอดชีวิตในรุ่นต่อไป โดยใช้วิธีการที่ว่าโครโมโซมที่ค่อนข้างดีจะถูกเลือกเพื่อนำไปผลิตลูกหลาน และโครโมโซมที่ไม่ค่อยดีนักจะถูกกำจัดออกและลูกหลานใหม่จะถูกนำมาแทนที่ ประชากรที่เหลือจะเป็นรุ่นใหม่ที่กำเนิดขึ้นแทน

4. วิธี Elitism

แนวคิดของ elitism นั้นคือเป็นการป้องกันการสร้างลูกหลานใหม่โดยวิธี Crossover และ mutation ซึ่งโครโมโซมที่ดีอาจมีโอกาสที่จะไม่ถูกเลือก ดังนั้น elitism จึงใช้วิธีการโดยการคัดลอกโครโมโซมที่ดีที่สุดไปยังประชากรใหม่ก่อนในขั้นตอนแรก ส่วนประชากรอื่นๆยังคงทำตามวิธีอื่นๆที่ได้กล่าวมา วิธี elitism นี้สามารถเพิ่มประสิทธิภาพได้อย่างรวดเร็วเนื่องจากสามารถป้องกันการสูญหายของโครโมโซมที่ดีที่สุดได้

ขั้นที่ 6 : สร้างโครโมโซมใหม่ด้วยการทำกระบวนการ Cross Over และ Mutation

การ Crossover

Single point crossover – จะมีตำแหน่งที่มีการ crossover เพียงตำแหน่งเดียว จากจุดเริ่มต้นของไบนารี่สตริงต้นแบบของโครโมโซมไปจนถึงตำแหน่งที่มีการ crossover จะถูกคัดลอกมาจากพ่อแม่ตัวแรก ที่เหลือมาจากอีกพ่อแม่หนึ่ง

11001011+11011**111 = 11001111**

รูปที่ 2.10 ภาพจำลองการ crossover แบบ one-point crossover

Two point crossover – เราจะเลือกจุด crossover 2 จุด จากตำแหน่งเริ่มต้นไบนารี่สตริงถึงจุดcrossover จุดแรกจะคัดลอกมาจากพ่อแม่ตัวแรก ส่วนที่อยู่ระหว่างจุด crossover แรกและสอง จะคัดลอกมาจากพ่อแม่อื่นส่วนที่เหลือจะนำมาจากพ่อแม่ตัวแรกอีกครั้ง

11001**011** + 11**011**111 = **11011111**

รูปที่ 2.11 ภาพจำลองการ crossover แบบ two-point crossover

Uniform crossover – บิตในโครโมโซมของพ่อแม่จะถูกสุ่มขึ้นเพื่อใช้ในการประกอบเป็นลูกหลานใหม่

11001**011** + **1**1011**1**01 = 11011111

รูปที่ 2.12 ภาพจำลองการ crossover แบบ uniform crossover

Arithmetic crossover - เป็นวิธีที่ใช้กระบวนการทางคณิตศาสตร์ในการสร้างลูกหลานใหม่ขึ้น

11001011 + 11011111 = 11001001 (AND)

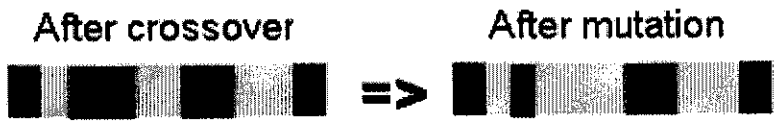
รูปที่ 2.13 ภาพจำลองการ crossover แบบ Arithmetic crossover

Permutation Single point crossover – ทำการเลือกตำแหน่ง crossover มาหนึ่งจุด ค่า permutation จะถูกคัดลอกจากพ่อแม่ตัวแรกไปถึงจุด crossover จากนั้นพ่อแม่อีกตัวหนึ่งจะทำการสแกนหาหมายเลขที่ยังไม่มีในลูกหลานและทำการเพิ่มต่อเข้าไป

(**1 2 3 4 5** 6 7 8 9) + (4 5 3 **6 8 9 7** 2 1) = (**1 2 3 4 5** 6 8 9 7)

การ Mutation

Bit inversion – เลือกบิตใดมาหนึ่งบิตแล้วทำการกลับค่าบิตนั้นใหม่



11001001 => 10001001

รูปที่ 2.14 ภาพจำลองการ mutation แบบ Bit inversion

Order changing – วิธีการนี้หมายเลข 2 หมายเลขจะถูกเลือกขึ้นมาและทำการสลับที่กัน

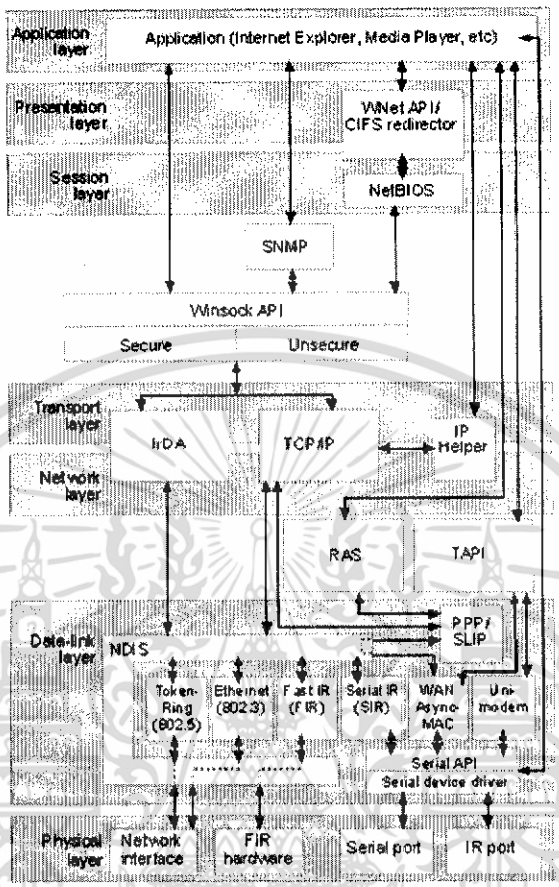
(1 **2** 3 4 5 6 **8** 9 7) => (1 **8** 3 4 5 6 **2** 9 7)

Value mutation – จะใช้การบวกหรือลบเลขจำนวนน้อยๆ หรืออาจคิดเป็นเปอร์เซ็นต์ในการเพิ่มหรือลด เข้าไปในค่าที่เราทำการเลือกไว้

(1.29 5.68 **2.86 4.11** 5.55) => (1.29 5.68 **2.73 4.22** 5.55)

ขั้นที่ 7 : นำโครโมโซมที่ได้ไปเป็นประชากรรุ่นถัดไป
ขั้นที่ 8 : ทำซ้ำขั้นตอนที่ 5จนได้จำนวนประกรใหม่เท่ากับ N
ขั้นที่ 9 : แทนที่ประชากรรุ่นเก่าด้วยประชากรรุ่นใหม่
ขั้นที่ 10 : ทำซ้ำขั้นตอนที่ 4 จนครบตามกำหนด

2.3 ทฤษฎีการเชื่อมต่อบนโปรโตคอล TCP/IP



รูปที่ 2.15 TCP/IP Model Architecture

ในที่นี้เราจะอธิบายใน 2 ชั้นสำคัญคือชั้นนำส่งข้อมูลและชั้นเครือข่าย

2.3.1 ชั้นนำส่งข้อมูล (Transport layer)

ระบบอินเตอร์เน็ตมีโปรโตคอลชั้นนำส่งข้อมูลหลัก 2 แบบคือ แบบ TCP (Transport Control Protocal) ซึ่งเป็นการนำส่งแบบต่อเนื่อง (connection-oriented) และแบบ UDP(User Datagram Protocal) ซึ่งเป็นการเชื่อมต่อแบบไม่ต่อเนื่อง (commectionless) เนื่องจาก UDP มีลักษณะคล้ายกับแพ็กเก็ต IP แต่มีการเติมส่วนหัวสั้นๆเข้าไปด้วย

2.3.1.1 UDP

ดังที่ได้กล่าวไว้ในตอนต้นว่า UDP เป็นรูปแบบการส่งดาต้าแกรมโดยที่ไม่ต้องมีการเชื่อมต่อก่อน UDP จะทำการส่งเซกเมนท์ข้อมูล (Segment) ที่ประกอบด้วยข้อมูลส่วนหัว

(header) ขนาด 8 ไบต์ ตามด้วยข้อมูล (payload) โดยที่ UDP จะใช้หมายเลขพอร์ตในการกำหนดหมายเลขที่อยู่ของเครื่องผู้ส่งข้อมูลและเครื่องผู้รับข้อมูล

ข้อมูล	ขนาด(บิต)
Source Port	16
Destination	16
Length	16
Checksum	16

ตารางที่ 2.1 แสดงรายละเอียดส่วนหัวของ UDP

หมายเลขที่อยู่ของผู้ส่ง (source port) ถูกนำไปใช้ในกรณีที่ต้องส่งแพ็กเก็ตตอบรับกลับไปยังผู้ส่งข้อมูลโดยการสำเนาหมายเลขผู้ส่งข้อมูลจากข้อมูลที่ได้รับไปยัง หมายเลขที่อยู่ผู้รับข้อมูลในแพ็กเก็ตที่จะส่งกลับไป

ในส่วนของ Length นั้นทำหน้าที่ระบุความยาวของข้อมูล(payload) ซึ่งรวมของความยาว header ขนาด 8 ไบต์ไว้ด้วย ในส่วนของ checksum มีหน้าที่ในการใช้นำมาคำนวนความถูกต้องของข้อมูลที่ได้รับ

UDP นั้นเป็นรูปแบบการส่งข้อมูลที่ไม่มีการรับประกันความถูกต้อง นั้นหมายความว่าต้นทางไม่สามารถทราบได้ว่าข้อมูลถูกส่งถึงแล้วหรือเสียหายระหว่างทาง ซึ่งดูแล้วอาจไม่เป็นประโยชน์นัก แต่ในการส่งข้อมูลที่ต้องการความรวดเร็วมากกว่าความถูกต้องนิยมที่จะใช้รูปแบบ UDP เช่นการทำประชุมผ่านเครือข่าย และการส่งข้อมูลของเกมผ่านเครือข่าย เนื่องจากเกมนั้นอาศัยลักษณะการส่งข้อมูลแบบ trigger คือบอกแต่ลักษณะที่เปลี่ยนไปเท่านั้น

2.3.1.2 TCP

โปรโตคอล TCP (Transmission Control Prtocol) ได้ถูกออกแบบให้เป็นโปรโตคอลที่มีความน่าเชื่อถือ เพื่อใช้ในการสื่อสารผ่านระบบเครือข่ายทั่วไป ที่อาจมีการผิดพลาดในการนำส่งข้อมูลอยู่เสมอ ซึ่ง TCP จะมีกระบวนการต่างๆในการตรวจสอบความผิดพลาด การควบคุมการไหลของข้อมูล เช่นการใช้เฟรมตอบรับ การทำ sliding window ด้วยสาเหตุดังกล่าวทำให้รูปแบบของ TCP มีความซับซ้อนมากกว่าของ UDP

ข้อมูล	ขนาด(บิต)
Source Port	16
Destination	16
Sequence #	32
Ack. #	32
Data Offset	4
[Reserved]	6
Control Bits	6
Window	16
Checksum	16
Urgent Pointer	16
Options	ไม่กำหนด(ขนาดเป็นจำนวนเท่าของ 8)
Padding	มีขนาดไม่แน่นอน

ตารางที่ 2.2 แสดงรายละเอียดส่วนหัวของ TCP

Source port และ Destination port ใช้ในการระบุหมายเลขพอร์ตที่อยู่ปลายทางทั้งสองข้างของสายสื่อสาร โฮสแต่ละฝั่งจะเป็นผู้กำหนดหมายเลขพอร์ต

Sequence number และ Acknowledgement number คือหมายเลขลำดับของแพ็กเก็ตและหมายเลขตอบรับ ซึ่งจะเป็นหมายเลขแพ็กเก็ตในลำดับต่อไปที่ผู้รับกำลังรอคอย ทั้ง 2 หมายเลขมีขนาด 32 บิต

TCP header length ทำหน้าที่ในการบอกขนาดของข้อมูลควบคุมสำหรับแพ็กเก็ต

บิต URG (Urgent) ใช้บอกความหมายว่าเป็นข้อมูลด่วน เมื่อถูกกำหนดให้เป็น 1 ซึ่งส่งผลให้ข้อมูลจะถูกส่งออกไปทันที

บิต ACK (Acknowledgement) จะถูกกำหนดให้เป็น 1 เพื่อบอกให้ทราบว่าหมายเลขลำดับการตอบรับ ซึ่งเป็นหมายเลขแพ็กเก็ตที่ผู้รับกำลังรอคอย

บิต PSH (Push) ทำหน้าที่ในการสั่งให้ส่งข้อมูลนั้นออกไปทันที ส่วนด้านผู้รับก็จะบังคับให้จัดการส่งข้อมูลนั้นไปประมวลผลในทันทีที่ได้รับ

บิต RST (Reset) ใช้ในกรณีที่การสื่อสารในการเชื่อมต่อนั้นเกิดความสับสนด้วยเหตุผลต่างๆ กันเช่น โฮสเกิดการทำงานล้มเหลว จึงต้องการให้ผู้รับยกเลิกการทำงานต่างๆที่ค้างอยู่แล้วเริ่มต้นกันใหม่นอกจากนี้ยังนำไปใช้ในการปฏิเสธกรชารเชื่อมต่อ หรือการปฏิเสธเซกเมนท์ที่ผิดพลาด ซึ่งหากบิตนี้ถูเซตเป็นหนึ่งแสดงว่าเกิดปัญหาในการเชื่อมต่อ

บิต SYN (Synchronous) ใช้สำหรับการเริ่มต้นติดต่อสื่อสาร โดยปกติจะกำหนดค่าให้ SYN = 1 และ ACK = 0 ถ้าการเชื่อมต่อเรียบร้อย ผู้รับจะตอบกลับมาด้วย SYN = 1 และ ACK = 1 ซึ่งเป็นค่าแทนที่ความหมายConnection request และ Connection accept นั่นเอง

การส่งข้อมูลใน TCP นั้นมีรูปแบบที่เป็น data stream มากกว่าเป็น message ซึ่งจำเป็นต้องมีการเชื่อมต่อก่อนการส่งข้อมูลซึ่งใช้วิธีที่เรียกว่า 3-way handshake ซึ่งการเชื่อมต่อนั้นจะเชื่อมต่อผ่าน port ทำให้สามารถส่งข้อมูลหลายโปรแกรมได้ในเวลาเดียวกันโดยใช้หมายเลข port ในการแยกข้อมูลของแต่ละโปรแกรม ซึ่งแต่ละหมายเลข TCP port จะมีทั้งที่กำหนดไว้เป็นมาตราฐานว่าหมายเลขใดๆ ใช้ในการทำงานอะไรและแบบที่เป็น dynamic

ถ้าเราสังเกตความแตกต่างของขนาดของ Header ของ UDP และ TCP จะเห็นว่าขนาดของ TCP ใหญ่กว่ามาก ซึ่งเป็นเหตุผลให้เกมส่วนใหญ่ไม่นิยมใช้ TCP ยกตัวอย่างเช่นตัวละครอาจกระโดดไปนานแล้ว แต่เนื่องจากมีการผิดพลาดในการส่งข้อมูลผิดพลาดซึ่งต้องผ่านกระบวนการตรวจสอบจึงต้องมีการส่งซ้ำจนกว่าจะถูกต้อง ส่งผลให้กินระยะเวลานาน ไม่ถูกต้องตามความเป็นจริง

2.3.2 ชั้นอินเตอร์เน็ต (Internet Layer)

ในชั้นนี้จะทำการระบุปลายทางและต้นทางในการส่งข้อมูลโดยอาศัย IP address ในการกำหนดเส้นทาง ซึ่งจะรวมอยู่ใน header ของชั้นนี้ และทำหน้าที่การหาเส้นทางในการส่งข้อมูลโดยใช้ routing protocol เช่น RIP OSPFโดยที่รายละเอียดของ header มีดังนี้

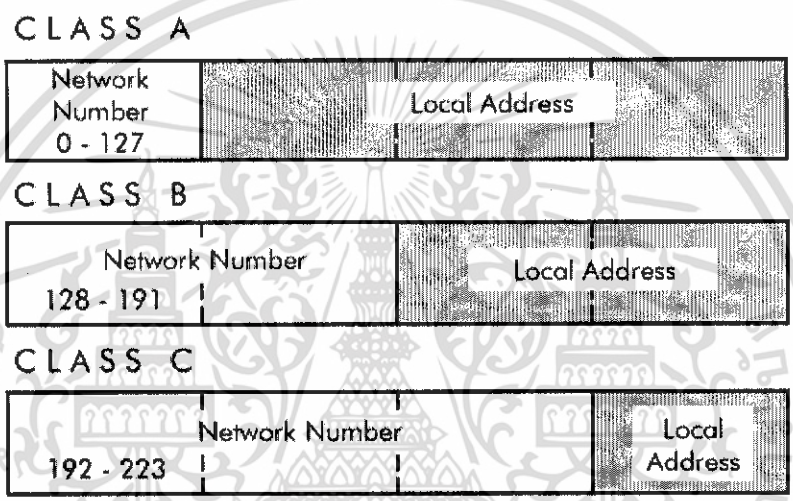
ข้อมูล	ขนาด(บิต)
Version Number	4
Header Length	4
Type of Service	8
Packet Length	16
Packet ID	16
Fragmentation	16
Time to Live	8
Protocol	8
Header Checksum	16
Source Address	32
Destination Address	32

ตารางที่ 2.3 แสดงรายละเอียดส่วนหัวของ IP

ซึ่งการส่งข้อมูลในชั้นนี้จะไม่มีความน่าเชื่อถือในการส่งข้อมูลว่าข้อมูลได้เดินทางไปถึงปลายทางหรือไม่ ซึ่งส่วนนี้จะถูกควบคุมโดยการทำงานในชั้นนำส่งข้อมูลในชั้นนี้ทำหน้าที่ในกำหนดที่อยู่ที่จะนำส่งและค้นหาเส้นทางในการส่งข้อมูลเท่านั้น

2.3.2.1 การกำหนดที่อยู่โดยใช้ IP Address

IP Address เป็นมาตรฐานในการกำหนดที่อยู่เสมือนให้แก่โฮส ซึ่งใช้ในการกำหนดที่อยู่ต้นทางและปลายทาง เพื่อให้ Routing protocol ทำการหาเส้นทางการส่งข้อมูลจากต้นทางไปยังปลายทาง โดยที่ IP Address แบ่งเป็นคลาสดังนี้



รูปที่ 2.16 แสดงการหมายเลข IP address ในแต่ละคลาส

Class	ขนาดของ หมายเลขเครือข่าย	หมายเลขแรก	จำนวนของหมายเลขทั้งหมด
A	1	0-127	16,277,216
B	2	128-191	65,536
C	3	192-223	256

ตารางที่ 2.4 แสดงจำนวนของหมายเลข IP address ในแต่ละคลาส

Class D อยู่ในหมายเลขระหว่าง 224 และ 239 ใช้ในการส่งแบบ multicast ซึ่งเป็นการส่งแบบเฉพาะไปยังกลุ่มเครือข่ายกลุ่มใดกลุ่มหนึ่ง

Class E ใช้ในการทดลองมีหมายเลขอยู่ระหว่าง 240 – 250

2.3.2.2 Routing Protocol

เป็นกระบวนการในการหาเส้นทางเพื่อส่งข้อมูลผ่านเครือข่ายไปยังปลายทาง ซึ่งอาศัยข้อมูลของ IP address เป็นตัวตัดสินใจว่าใช้เส้นทางใดในการขนส่ง ซึ่งส่วนใหญ่ protocol เหล่านี้จะทำงานบน router routing protocol แบ่งเป็น 2 ประเภทใหญ่คือ

1. Interior routing protocol ถูกออกแบบขึ้นสำหรับเครือข่ายที่มีผู้ดูแลเพียงคนเดียวหรือภายในAutonomous system โดยที่จุดประสงค์หลักคือหาเส้นทางที่ดีที่สุดในเครือข่าย ตัวอย่างวิธีการเช่น RIP IGRP OSPF เป็นต้น

2. Exterior routing protocol ออกแบบมาสำหรับค้นหาเส้นทางที่ดีที่สุดระหว่างเครือข่ายที่แตกต่างกัน หรือคนละ Autonomous system ซึ่งมักใช้ในการเชื่อมต่อระหว่าง ISP และระหว่าง บริษัท ตัวอย่างวิธีการเช่น BGP

ซึ่งหากจำแนกรูปแบบ Routing Protocol ตามกระบวนการหาเส้นทางแล้วสามารถแบ่งได้เป็น 2 ประเภทคือ

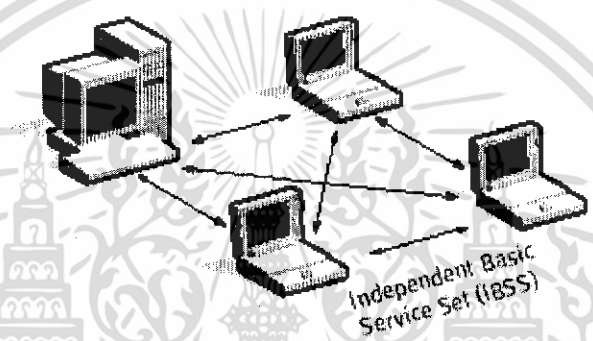
1. Distance Vector Routing Protocol ใช้จำนวนของ Hop มาเป็นปัจจัยในการเลือกเส้นทางที่ดีที่สุด โดยไม่สนใจปัจจัยอื่นๆ เช่น ความเร็ว ความน่าเชื่อถือ ตัวอย่างเช่น RIP BGP มีข้อดีตรงที่มีการใช้งาน bandwidth ในการสร้าง routing table น้อยทำให้ไม่สิ้นเปลืองทรัพยากร แต่สามารถใช้กับเครือข่ายขนาดเล็กเท่านั้น เนื่องจากมีการกำหนดค่าสูงสุดของ Hop ไว้ เพื่อป้องกันการนับสู่ infinity

2. Link – State Routing Protocol ใช้การจำลองรูปแบบของเครือข่ายไว้เป็นของตัวเองซึ่งสร้าง database ของตัวเองไว้ แล้วจึงใช้กระบวนการ SPF ในการหาเส้นทางที่ดีที่สุด มีข้อดีตรงที่เมื่อระบบล้มเหลวจะสามารถกลับมาทำงานตามปรกติได้อย่างรวดเร็ว แต่มีข้อเสียตรงที่สิ้นเปลืองทรัพยากรเนื่องจากมีการประมวลผลสูง และใช้การแลกเปลี่ยนข้อมูลขนาดใหญ่จึงเปลือง bandwidth

2.4 เครือข่ายไร้สาย Wi-Fi

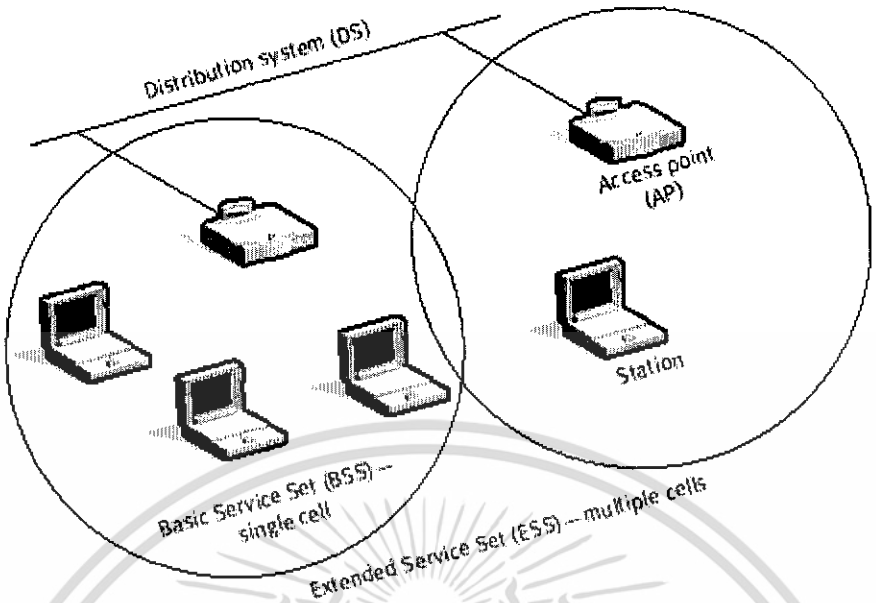
2.4.1 สถาปัตยกรรม

เมื่อมีอุปกรณ์ไร้สายมาเชื่อมต่อกัน จะเกิดการสร้างกลุ่มการให้บริการ(Basic Service Set (BSS)) ซึ่ง BSS จะสร้างขึ้นได้ต้องมีอย่างน้อย 2 สถานี BSS ที่ไม่ได้เชื่อมต่ออยู่กับสถานีฐาน เราจะเรียกว่ากลุ่มการให้บริการอิสระ(Independent Basic Set Service(IDSS)) หรือที่รู้จักกันในเครือข่าย AD-Hoc ซึ่งเป็นเครือข่ายที่เป็นการติดต่อแบบ peer – peer ซึ่งใช้ในเครือข่ายที่สร้างขึ้นเองในการใช้งานชั่วคราว



รูปที่ 2.17 แสดงการติดต่อแบบไร้สายภายใน BSS(Ad-Hoc)

เมื่อ BSS มีการติดต่อกันระหว่างเครือข่ายซึ่งเราจะเรียกว่า Infrastructure ซึ่งมีการติดต่อหลานรูปแบบ เช่น การเชื่อมต่อโดยใช้ Distribution System หรือ DS DS จำทหน้าที่ในการเชื่อมต่อเครือข่ายให้มีขนาดใหญ่ขึ้น การเข้าสู่ DS จำเป็นต้องมี Access point(AS) ซึ่งทำหน้าที่เป็นสถานี ข้อมูลจะเคลื่อนย้ายระหว่าง BSS โดยอาศัย DS และ access point



รูปที่ 2.18 แสดงการติดต่อแบบไร้สายระหว่าง BSS(Infrastructure)

การสร้างเครือข่ายที่มีขนาดใหญ่และซับซ้อนทำให้มีส่วนหนึ่งเพิ่มขึ้นซึ่งเราเรียกว่ากลุ่มการให้บริการส่วนต่อขยาย(Extended Service Set(ESS)) ซึ่งจะทำให้การเขาสู่เครือข่ายนั้นเป็นอิสระจาก Logical Link Control layer (LLC)

การเชื่อมต่อระหว่าง 802.11 กับเครือข่ายที่ใช้สายนั้นจำเป็นต้องมีส่วนที่เรียกว่า portal ซึ่งทำหน้าที่ในการแปลงรูปแบบของข้อมูลที่ผ่านให้เหมาะสมตามมาตรฐานระหว่างการส่งแบบใช้สายและไร้สาย

การให้บริการของ DS แบ่งออกได้เป็น 2 ส่วนดังนี้

1. Station Services (SS)
2. Distribution System Services (DSS) โดย DSS ให้บริการ 5 สิ่งคือ
    1. Association
    2. Reassociation
    3. Disassociation
    4. Distribution
    5. Integration

การให้บริการในสามข้อแรกนั้นใช้กับสถานีที่มีการเคลื่อนที่ ถ้าสถานีมีการเคลื่อนที่อยู่ใน BSS ของตนเองหรือไม่เคลื่อนที่ เราจะถือว่าสถานีนั้นไม่มีการเปลี่ยนแปลงตำแหน่ง ถ้ามีการเคลื่อนย้านระหว่าง BSS ใน ESS เดียวกัน เราถือว่ามีการ BSS-transition แต่ถ้ามีการเคลื่อนย้ายระหว่าง BSS ไปยัง ESS อื่น สถานีต้องทำการติดต่อกับ access point ของ BSS ที่ใหม่

Association ทำหน้าที่ในการสนับสนุนการทำงานของสถานีที่ไม่มีการเคลื่อนย้าย หากสถานีมีการเคลื่อนย้าย ส่วนของ Reassociation จะทำหน้าที่แทน ซึ่งจะอนุญาติให้สถานีทำการสลับการ Association จาก AP หนึ่งไปยัง AP หนึ่ง Disassociation จะทำหน้าที่เมื่อ สิ้นสุดการติดต่อกับ AP

ในส่วนของ Distribution และ Integration นั้น Distribution ทำหน้าที่ในการรับข้อมูลจากผู้ส่งส่งต่อไปยังผู้รับ หากผู้ส่งส่งข้าม BSS จะทำการส่งข้อมูลออกจาก AP ต้นทางผ่าน DS ไปยัง AP ปลายทาง หากอยู่ใน BSS เดียวกัน AP ที่ใช้จะใช้แค่ AP ต้นทาง

แต่ละสถานียังมีการให้บริการอีก 4 แบบคือ

1. Authentication
2. Deauthentication
3. Privacy
4. MAC Service Data Unit (MSDU) Delivery

ในระบบเครือข่ายไร้สาย ขอบเขตของการสื่อสารจะไม่แน่นอนเหมือนระบบวางสาย ดังนั้นในการควบคุมการเข้าถึงเครือข่าย ซึ่งสถานีจะต้องทำการระบุตัวตนก่อนเข้าถึง นั้นคือขั้นตอนการ Authentication ซึ่งมีการยืนยันตัวตนแล้วจึงสามารถทำการ Association ได้ การทำ Authentication มีอยู่ 2 ประเภทในมาตราฐาน 802.11

1. Open System Authentication ใครก็ตามที่ทำการยืนยันตัวจะมรสิทธิ์ในการเข้าถึงเครือข่าย

2. Shared Key Authentication ผู้ที่มีคียเท่านั้นจึงจะได้รับอนุญาติ ซึ่ง Shared Key จะใช้อัลกอริทึม Wired Equivalent Privacy (WEP) privacy ในการค้นส่งไปยังสถานีอื่น เพื่อให้ข้อมูลเป็นความลับ

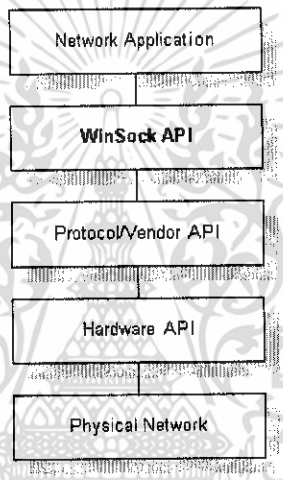
ขั้นตอนDeauthentication ใช้เมื่อแต่ละสถานีต้องการสิ้นสุดการ authentication ซึ่งจะทำให้เกิดการ disassociated

ซึ่ง 802.11 จะใช้การเข้ารหัส เพื่อป้องกันการดักฟังโดยใช้ WEP ในการเข้ารหัสในส่วนของข้อมูล

2.5 Winsock

2.5.1 สถาปัตยกรรมและการทำงาน

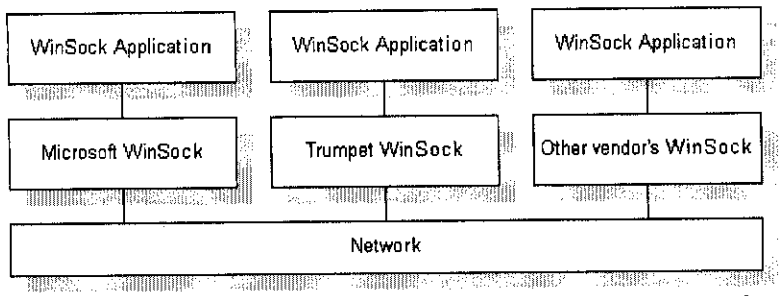
Winsock เป็น network application programming interface (API) สำหรับ Microsoft Windows ซึ่งถูกออกแบบมาให้เรียกใช้ datastructure และ function โดยใช้ dynamic link library (DLL). Winsock ทำหน้าที่เป็นตัวแปลงการทำงาน หากในโปรแกรมของเรามีการเรียกฟังก์ชั่นในการติดต่อ winsock จะแปลงฟังก์ชั่นที่ร้องขอเป็นไปตาม การทำงานของ protocol ที่ร้องขอมา และทำงานในส่วนอื่นที่จำเป็น นั้นคือทำหน้าที่อยู่ระหว่าง application และ การทำงานของ network



รูปที่ 2.19 แสดงการรูปแบบการทำงานของ Winsock

ก่อนหน้าที่จะมีการคิดค้น winsock แต่ละผู้ผลิตจะสร้าง interface library เป็นของตนเอง ทำใหรูปแบบการทำงานของ network application มีหลากหลายมาก ผีพัฒนาจำเป็นต้องเรียนรู้ในการเขียนโปรแกรมของแต่ละ interface ทั้งยังคงต้องเขียนโปรแกรมเพื่อรองรับ protocol ที่ต่าวกันด้วย ซึ่งทำให้พัฒนาได้ยาก ซึ่ง winsock ได้เข้ามาแก้ปัญหาเหล่านี้โดยทำข้อตกลงกับผู้ผลิตส่วนใหญ่ให้ใช้ wimsock เป็นมาตราฐาน ทำให้ผีพัฒนาทำการพัฒนาได้ง่ายขึ้นเพราะใช้การเขียนโปรแกรมเพียงรูปแบบเดียวแต่สามารถใช้ได้แพร่หลาย

การทำงานของ Winsock API ไม่ใช้เป็นเพียงมาตราฐานในการเขียนซอร์สโค้ดเท่านั้น แต่ยังเป็นมาตราฐานในระดับเลขฐานสองอีกด้วย ทำให้โปรแกรม winsock ที่ผ่านการคอมไพล์มาแล้ว สามารถทำงานได้โดยไม่ต้องคอมไพล์ใหม่บนเครื่องที่มีการรองรับ winsock อยู่แล้ว Winsock สามารถทำงานได้บน interface ของผู้ผลิตใดก็ได้ที่รองรับ



รูปที่ 2.20 แสดงการทำงานของ winsock กับส่วนติดต่อของผู้ผลิตอื่นๆ

Winsock นั้นมีการพัฒนามาจาก Berkeley sockets API ซึ่งใน winsock ยังคงมีฟังก์ชั่นส่วนใหญ่ของ Berkeley sockets API โดยที่แนวคิดของ Berkeley sockets API ในการติดต่อนั้นแทนที่เราจะทำการสลับสายในการติดต่อกับหลายปลายทาง หรือใช้วงจรอิเล็กทรอนิกส์มาช่วยสลับเช่นเดียวกับการทำงานของโทรศัพท์สมัยก่อน เราสร้าง socket ขึ้นมาเพื่อทำหน้าที่เป็นเสมือนช่องทางในการติดต่อแทนเพื่อลดความยุ่งยาก

2.5.2 รุ่นของ Winsock

2.5.2.1 รุ่น 1.1

เป็นรุ่นมาตรฐานรุ่นแรกที่ใช้ในการเป็นมาตรฐานในการผลิต ซึ่งมีจุดเด่นดังนี้

- เป็นมาตรฐานในการใช้งาน ซึ่งมีความเหมาะสมกับโปรโตคอล TCP/IP แทนที่จะโปรโตคอลของแต่ละผู้ผลิต
- รองรับในระดับเลขฐานสอง โดยใช้ dynamic link library (DLL) ในการให้บริการแก่ผู้ผลิตที่แตกต่างกัน
- มีการจัดเรียงลำดับของข้อมูลตามชนิดของ CPU
- มี WSA ซึ่งประกอบด้วยฟังก์ชั่น ชนิดของข้อมูล และโครงสร้างซึ่งทำให้ง่ายในการพัฒนา windows application

2.5.2.2 รุ่น 2

มีความสามารถเพิ่มขึ้นจากรุ่นก่อนมาก ซึ่งที่สำคัญก็คือรองรับโปรโตคอลอื่นๆนอกจาก TCP/IP และมีกระบวนการในการทำให้ application เป็นอิสระจากโปรโตคอล ซึ่งข้อดีที่เพิ่มขึ้นมีดังนี้

- มีการรองรับได้หลายโปรโตคอล โดยอนุญาตให้ application สามารถ ติดต่อกับ socket interface อื่นในเวลาพร้อมๆ ซึ่งขึ้นอยู่กับโปรโตคอลที่ติดตั้งไว้ไม่เฉพาะแต่ TCP/IP เท่านั้น
- มีการใช้ Asynchronous I/O และ event objects โดยที่ Winsock2 จะใช้ overlapped I/O model ใน win32 ซึ่ง Asynchronous I/O จะอนุญาตให้ application สามารถทำงานกับ

สิ่งอื่นขณะที่รอกระบวนการ I/O ทำงานเสร็จสิ้น และ Event objects ใช้ในการวิเคราะห์เมื่อการทำงานเสร็จสิ้นถ้า application ไม่มีหน้าต่างกราฟิก

- คุณภาพในการให้บริการ เช่นการให้บริการด้าน real-time multimedia ซึ่ง Winsock2 จะสามารถทำให้ application มีการตกลงถึงความต้องการของการบริการเช่น bandwidth และ latency ในการเริ่มเชื่อมต่อ

Winsock 2 นั้นออกแบบมาตามรูปแบบของ Windows Open Services Architecture (WOSA) ซึ่งได้ระบุชนิดของการติดต่อสำหรับการเชื่อมต่อของ front-end applications และ back-end services ทำให้เกิดเป็นรูปแบบการทำงานของ Application Program Interface (API) และ Service Provider Interface (SPI) API จะระบุฟังก์ชั่นและโครงสร้างข้อมูลแก่ application ที่ต้องการเข้าถึงการให้บริการ ส่วน SPI จะทำการเชื่อมต่อโปรโตคอลของผู้ผลิตเพื่อให้สามารถใช้การให้บริการเครือข่ายได้

Winsock 2 DLL นั้นไม่สนับสนุนกับโปรโตคอลของผู้ผลิต ซึ่ง Winsock.DLL และ WSOCK32.DLL ก็เช่นกัน แต่ว่าแต่ละโปรโตคอลของผู้ผลิตจะมีไลบารี่ที่สามารถเชื่อมต่อเข้ากับ Winsock2 DLL ได้ ซึ่งจำนวนของการให้บริการของไลบารี่ในการเชื่อมต่อไปยัง Winsock สามารถแปลเปลี่ยนได้ตาม runtime ทำให้ Winsock 2 สามารถทำการติดต่อกับหลายโปรโตคอลในเวลาเดียวกัน ซึ่งเป็นการพัฒนาขึ้นจาก Winsock 1.1 ซึ่งใช้งาน DLL ไฟล์เดียว

บทที่ 3
การออกแบบระบบ

ในหัวข้อนี้กล่าวถึงการออกแบบชิ้นงานวิจัย หรือการออกแบบส่วนต่างๆ ของโครงงานนี้

3.1 บทนำ

โครงงานชิ้นนี้นำหลักการของออกแบบเชิงวัตถุ หลักการออกแบบเกม ความรู้ด้านปัญญาประดิษฐ์ และความรู้ด้านการติดต่อสื่อสารมาประกอบการออกแบบเกม

3.2 การออกแบบรายละเอียดของเกม

3.2.1 เนื้อเรื่องภายในเกม

ในอดีตดวงดาวดวงหนึ่งมีสัตว์เทพเจ้าอยู่ 4 ชนิด ได้แก่ พยัคฆ์, มังกร, นกไฟ และเต่ายักษ์ โดยสัตว์เทพทั้ง 4 นี้เป็นผู้ดูแลดวงดาวดวงนั้นมาช้านาน เมื่อเวลาผ่านพ้นมาจนถึงจุดเปลี่ยนแปลงครั้งใหญ่ของดวงดาวแห่งนี้เป็นวันที่สรรพสิ่งต่างๆ บนดวงดาวจะถูกธรรมชาติทำลายจนหมดสิ้น เพื่อเป็นการก่อกำเนิดสิ่งต่างๆ ขึ้นใหม่ จากเหตุการณ์ครั้งนั้นทำให้สัตว์เทพเจ้าทั้ง 4 ชนิดได้หายไปเหลือไว้เพียงแต่ “สายเลือด” ที่ตกทอดไปยังสิ่งมีชีวิตที่เกิดขึ้นใหม่ ดังนั้นสิ่งมีชีวิต หรือมอนสเตอร์ที่เกิดขึ้นใหม่ภายในดวงดาวแห่งนี้ประกอบไปด้วยสายเลือดของสัตว์เทพเจ้าทั้ง 4 ผสมผสานอยู่ในกายมากน้อยแตกต่างกันไป

ผู้เล่นรับบทเป็นมนุษย์โลกที่เดินทางไปยังดวงดาวแห่งนั้นและเก็บมอนสเตอร์จากดาวดวงนั้นกลับมาเลี้ยงที่โลกมนุษย์ เป็นสัตว์เลี้ยง และเป็นเพื่อนกับผู้เล่นตลอดเวลา

ผู้เล่นสามารถเลี้ยงดูมอนสเตอร์ของตนแล้วนำไปผสมพันธุ์กับมอนสเตอร์ของผู้เล่นคนอื่นเพื่อให้ได้มอนสเตอร์ชนิดใหม่ที่มีคุณสมบัติที่ดีเยี่ยมยิ่งขึ้นกลับมาเลี้ยง หรือสามารถนำมอนสเตอร์ของตนไปต่อสู้ประลองฝีมือกับมอนสเตอร์ของผู้เล่นคนอื่นเพื่อวัดความสามารถในการเลี้ยงดูก็ได้

การต่อสู้ภายในเกมผู้เล่นสามารถต่อสู้ตัวคนเดียว หรือต่อสู้กับผู้เล่นด้วยกันได้ด้วยการเชื่อมต่อแบบ wi-fi

เส้นทางสู่การเป็นนักเลี้ยงมอนสเตอร์มือหนึ่ง กำลังเปิดรอผู้เล่นอยู่...

3.2.2 วัตถุประสงค์ของเกม

วัตถุประสงค์ของเกม คือ การเลี้ยงดูมอนสเตอร์ของตนเองให้มีความแข็งแกร่งเหนือมอนสเตอร์อีกมากมายหลากหลายชนิดที่มีอยู่ในเกม และมีความแข็งแกร่งเหนือกว่ามอนสเตอร์ของผู้เล่นอื่นๆ ซึ่งผู้เล่นต้องแสวงหาพ่อพันธุ์-แม่พันธุ์ที่แข็งแกร่งจึงจะก่อกำเนิดมอนสเตอร์ที่แข็งแกร่งที่สุดได้

3.2.3 มุมมองภายในเกม

มุมมองภายในเกมเป็นมุมมองด้านหน้า บนหน้าจออุปกรณ์พ็อกเก็ตพีซี ขนาด 640 x 480 พิกเซล ด้วยการแสดงผลแบบ 3 มิติ

3.2.4 กฎ กติกาภายในเกม

1. รายละเอียดของมอนสเตอร์

มอนสเตอร์ 1 ตัวมีรายละเอียด ดังนี้

พารามิเตอร์	รายละเอียด	หมายเหตุ
Hp	ค่าพลังชีวิต	มีค่าไม่เกิน HPmax
Hpmax	ค่าพลังชีวิตสูงสุด	-
Sp	ค่าพลังวิญญาณ	มีค่าไม่เกิน Hpmax
Spmax	ค่าพลังวิญญาณสูงสุด	ค่าพลังวิญญาณสูงสุด
Age	อายุ	-
Exp	ค่าประสบการณ์	มีผลต่อการเลื่อนระดับ Level
Level	ระดับความเก่ง	มีค่าได้ตั้งแต่ 1 ถึง 99
Energy	ค่าพลังงาน	มีผลต่อการทำการฝึกฝน มีค่าได้ตั้งแต่ 0 ถึง 99
Status	ค่าสถานะ	มีผลต่อการเจริญเติบโต มีสถานะ {ตาย, ใกล้ตาย, ป่วย, ปกติ, มีความสุข, แข็งแรงดีมาก}
Name	ชื่อ	-
Full	ระดับความอิ่ม	มีผลต่อค่าสถานะ

		มีค่าได้ตั้งแต่ 0 ถึง 100
Thirsty	ระดับความกระหาย	มีผลต่อค่าสถานะ มีค่าได้ตั้งแต่ 0 ถึง 100
Str	ค่าความแข็งแกร่งของกาย	มีผลต่อค่าพลังโจมตีระยะใกล้ มีค่าได้ตั้งแต่ 1 ถึง 255
Qi	ค่าความแข็งแกร่งของจิต	มีผลต่อค่าพลังโจมตีระยะไกล มีค่าได้ตั้งแต่ 1 ถึง 255
Def	ค่าความทนทานของร่างกาย	ค่าพลังป้องกันการถูกโจมตี มีค่าได้ตั้งแต่ 1 ถึง 255
Agi	ค่าความคล่องตัว	มีผลต่อการหลบหลีกเมื่อถูกโจมตี มีค่าได้ตั้งแต่ 1 ถึง 255
Dex	ค่าความแม่นยำ	ส่งผลให้สามารถโจมตีได้แรงมากขึ้น มีค่าได้ตั้งแต่ 1 ถึง 255
Fit	ค่าความพร้อมของร่างกาย	ส่งผลต่อการโจมตีด้วยค่าพลังโจมตีสูงสุด มีค่าได้ตั้งแต่ 1 ถึง 255
Vit	ระดับพลังชีวิต	ส่งผลต่อค่า Hpmax มีค่าได้ตั้งแต่ 1 ถึง 255
Breath	ระดับพลังปราณ	ส่งผลต่อค่า Spmax มีค่าได้ตั้งแต่ 1 ถึง 255
Int	ค่าความฉลาด	ส่งผลต่อการเลือกการกระทำ (action) ที่ดีที่สุดในขณะต่อสู้ มีค่าได้ตั้งแต่ 1 ถึง 255
Hierarchy_level	ระดับชั้นสายพันธุ์	-
Body	รหัสร่างกาย	บอกว่ามีร่างกายแบบไหน
Type	ชนิดหลักของสายเลือด	มี 4 ชนิด ได้แก่ พยัคฆ์, มังกร, นกไฟ, เต่ายักษ์
Team	ประเภทของมอน	บอกว่าเป็นมอนสเตอร์ของผู้เล่น, ของผู้เล่นคนอื่น หรือของ

		computer
expRequired	จำนวนค่าประสบการณ์ที่ต้องใช้เพื่อเลื่อนระดับความเก่ง (Level) ในระดับถัดไป	เมื่อมีค่าประสบการณ์ (Exp) ถึงค่านี้ ตัวละครจะมีระดับ Level เพิ่มขึ้นอีก 1 ระดับ
Space	ขอบเขตการเรียนรู้จากการฝึกฝน (จะเพิ่มขึ้นเมื่อระดับ Level เพิ่ม)	ค่าพารามิเตอร์ที่ใช้ในการต่อสู้ (Str, Qi, Def, Agi, Dex, Fit, Vit, Breath, Int) มีค่ารวมกันได้ไม่เกินค่า Space

ตารางที่ 3.1 แสดงค่าพารามิเตอร์ประจำตัวมอนสเตอร์

การกระทำ	รายละเอียด	หมายเหตุ
Move_Forward	เคลื่อนที่ไปข้างหน้า	-
Move_Backward	เคลื่อนที่ไปข้างหลัง	-
Jump	เคลื่อนที่หลบการโจมตี	นำค่า Agi มาคำนวณโอกาสที่จะหลบได้
Punch	ต่อย / เตะ	การโจมตีใกล้ ค่าความรุนแรงขึ้นกับค่า Str
Power	ปล่อยพลัง	การโจมตีไกล ค่าความรุนแรงขึ้นกับค่า Qi
Charge	ชาร์จพลัง	เพิ่มค่า Sp ครั้งละ 5 ค่า

ตารางที่ 3.2 แสดงการกระทำประจำตัวของมอนสเตอร์

2. ชนิดของมอนสเตอร์

ตัวละครมอนสเตอร์ภายในเกมสืบสายเลือดมาจากสัตว์เทพในอดีต 4 ชนิด ที่มีลักษณะเด่นแตกต่างกัน ดังนี้

1) พยัคฆ์ เน้นการโจมตีระยะใกล้รุนแรง (เน้น Str)
2) มังกร เน้นการโจมตีระยะไกลรุนแรง (เน้น Qi)
3) นกไฟ เน้นความรวดเร็วในการหลบหลีก (เน้น Agi)
4) เต่ายักษ์ เน้นความแข็งแกร่งป้องกันการโจมตี (เน้น Def)

มอนสเตอร์ภายในเกมมีสายเลือดทั้ง 4 ชนิดนี้ปะปนอยู่ในร่างกาย สำหรับตัวละครมอนสเตอร์ของผู้เล่นเป็นมอนสเตอร์ชนิดไหนนั้น ขึ้นอยู่กับค่าพารามิเตอร์หลักของมอนสเตอร์นั้นๆ กล่าวคือหากมอนสเตอร์มีค่าพารามิเตอร์ค่าใดใน 4 ค่าที่กล่าวถึงด้านบนมากกว่า จะถือว่ามอนสเตอร์ตัวนั้นมีสายเลือดของสัตว์เทพเจ้าที่เน้นค่าพารามิเตอร์นั้นมากที่สุด

3. การเจริญเติบโต

ตัวละครมอนสเตอร์ภายในเกมสามารถมีการเจริญเติบโต หรือมีวิวัฒนาการได้ 2 ทาง ได้แก่

1) การฝึกฝนเลี้ยงดูจากผู้เล่น
2) การผสมพันธุ์ระหว่างมอนสเตอร์จากผู้เล่นตั้งแต่ 2 คนขึ้นไป

ในกรณีที่เจริญเติบโตจากการเลี้ยงดูฝึกฝนจากผู้เล่นนั้น ระบบจะทำการตรวจสอบสถานะของมอนสเตอร์อยู่ตลอดเวลาว่ามอนสเตอร์มีการเจริญเติบโตเพียงพอหรือยัง หากการเจริญเติบโตถึงระดับหนึ่งแล้วมอนสเตอร์จะได้รับการเปลี่ยนแปลงร่างกายไปเป็นร่างกายตามชนิดสายเลือดหลักของมอนสเตอร์ในขณะนั้น

การตรวจสอบการเปลี่ยนร่างของมอนสเตอร์ทำการเปลี่ยนแปลงทุกครั้งที่ค่า Level ของมอนสเตอร์มีค่าที่ลงท้ายด้วยเลข 5 หรือเลข 0 เช่น ที่ค่า Level 5, 10, 15, 20, 25 เป็นต้น โดยการเปลี่ยนร่างในครั้งถัดไปสามารถเปลี่ยนเป็นร่างมอนสเตอร์ที่อยู่คนละสายกับร่างในปัจจุบันได้ (ตามชนิดของมอนสเตอร์ในระดับ Level นั้นจริงๆ)

4. เมนูภายในเกม

4.1 ดูสถานะของมอนสเตอร์

แสดงค่าพารามิเตอร์และสถานะของมอนสเตอร์ในสภาวะปัจจุบัน

4.2 ฝึกฝน

เมนูการฝึกฝนมอนสเตอร์ ซึ่งประกอบไปด้วยชนิดของการฝึกฝน ดังนี้

เมนูการฝึกฝน	**สิ่งที่ได้รับ**	**หมายเหตุ**
ยกดัมเบล	เพิ่มค่า Str และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Str
งอข้อน	เพิ่มค่า Qi และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Qi
วิ่ง	เพิ่มค่า Agi และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับ

		การเพิ่มค่า Agi
ลากท่อนซุง	เพิ่มค่า Def และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Def
ปาเป้า	เพิ่มค่า Dex และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Dex
กินอาหารชีวจิต	เพิ่มค่า Vit, Breath, Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Vit, Breath
กินวิตามิน	เพิ่มค่า Fit และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Fit
อ่านหนังสือ	เพิ่มค่า Int และค่า Exp	สุ่มค่าตั้งแต่ 1 ถึง 3 สำหรับการเพิ่มค่า Int
ฝึกฝนต่อสู้	ได้รับค่า Exp	ได้รับค่า Exp สูงกว่าการฝึกฝนอื่นๆ จะได้ในกรณีที่เป็นผู้ชนะ

ตารางที่ 3.3 แสดงการฝึกฝนภายในเมนูฝึกฝน

4.3 ให้อาหาร

เมนูอาหาร มีดังนี้

1) ไก่ย่าง - เพิ่มระดับความอิ่ม +5
2) ข้าวผัดปู - เพิ่มระดับความอิ่ม +10
3) สเต็ก - เพิ่มระดับความอิ่ม +15

เมนูเครื่องดื่ม มีดังนี้

1) น้ำเปล่า - เพิ่มระดับความไม่กระหาย +5
2) ชาเขียว - เพิ่มระดับความไม่กระหาย +10
3) ไวน์ - เพิ่มระดับความไม่กระหาย +15

4.4 พักผ่อน

เมนูนี้ทำการให้มอนสเตอร์พักผ่อน โดยตลอดระยะเวลาที่มอนสเตอร์อยู่ในโหมดพักผ่อนนี้ ค่าพารามิเตอร์เหล่านี้จะค่อยๆ เพิ่มขึ้นทีละน้อย

- ค่าพลังชีวิต (Hp)
- ค่าพลังวิญญาณ (Sp)

- ค่าพลังงาน (Energy)

4.5 ต่อสู้ประลองฝีมือกับผู้เล่นคนอื่น

เมนูนี้ทำการเข้าสู่การต่อสู้ระหว่างผู้เล่นกับผู้เล่นคนอื่น ซึ่งรองรับการต่อสู้ระหว่างผู้เล่นได้ครั้งละ 2 คนต่อ 1 การต่อสู้เท่านั้น โดยจะต้องระบุหมายเลข IP Address ของผู้เล่นฝ่ายตรงข้าม พร้อมทั้งเลือกว่าจะให้ใครเป็นเครื่องที่ทำการต่อสู้ จากนั้นเครื่องที่ไม่ได้เป็นเครื่องสำหรับทำการต่อสู้จะส่งข้อมูลมอนสเตอร์ของตัวเองไปให้แก่เครื่องที่ทำหน้าที่จัดการต่อสู้

จากนั้นระบบจะตรวจสอบข้อมูลของมอนสเตอร์ของผู้เล่นทั้งสองคนแล้วทำการต่อสู้ เมื่อรู้ผลแพ้ชนะแล้วระบบจะเพิ่มค่าประสบการณ์ให้แก่มอนสเตอร์ที่เป็นผู้ชนะ แล้วส่งข้อมูลมอนสเตอร์กลับคืนแก่ผู้เล่นทั้งสอง และปิดการเชื่อมต่อ

4.6 การผสมพันธุ์มอนสเตอร์

เมนูนี้ทำการผสมพันธุ์มอนสเตอร์โดยรับข้อมูลพ่อพันธุ์ – แม่พันธุ์จากผู้เล่นเข้ามาในระบบ แล้วทำการผสมพันธุ์จนได้ลูกมอนสเตอร์ที่มีวิวัฒนาการสูงขึ้น แล้วส่งข้อมูลลูกมอนสเตอร์ที่ได้ในรูปแบบของไข่ (Egg) กลับคืนไปให้แก่ผู้เล่น ซึ่งมีข้อกำหนดว่าไข่ (Egg) ที่ได้สามารถส่งค่ากลับคืนไปให้แก่ผู้เล่นได้แค่ 1 คนเท่านั้น

ระบบที่จัดการการผสมพันธุ์นี้อยู่บนเครื่องเซิร์ฟเวอร์ ซึ่งผู้เล่นทุกคนที่ต้องการผสมพันธุ์มอนสเตอร์ในแต่ละครั้งต้องทำการเชื่อมต่อเข้าสู่เครื่องเซิร์ฟเวอร์นี้ โดยมีข้อจำกัดว่าระบบรองรับการเชื่อมต่อจากผู้เล่นได้ครั้งละ 5 คน / เครื่อง เท่านั้นต่อการผสมพันธุ์มอนสเตอร์แต่ละครั้ง

4.7 ห้องฟักไข่

ส่วนของการเก็บข้อมูลไข่มอนสเตอร์ หากมีข้อมูลไข่มอนสเตอร์จัดเก็บอยู่ผู้เล่นสามารถเลือกคำสั่ง 'ฟักไข่' ได้ภายในเมนูนี้

หากผู้เล่นทำการ 'ฟักไข่' มอนสเตอร์ตัวใหม่ที่ออกมาจากไข่ (Egg) จะทำการทับข้อมูลมอนสเตอร์ตัวที่ผู้เล่นมีอยู่ก่อนหน้าที่จะทำการฟักไข่ และไม่สามารถนำข้อมูลมอนสเตอร์ตัวที่ถูกลบไปแล้วกลับคืนมาได้

5. การต่อสู้ของมอนสเตอร์

การต่อสู้ภายในเกมตัวมอนสเตอร์จะทำการคิด และดำเนินการกระทำต่างๆ เอง โดยผู้เล่นไม่สามารถบังคับมอนสเตอร์ได้

ผลการต่อสู้จะสิ้นสุดลงเมื่อมอนสเตอร์ฝ่ายใดฝ่ายหนึ่งมีค่าพลังชีวิต (Hp) เท่ากับ 0 ก่อน หรือเมื่อเวลาในฉากต่อสู้ลดลงจนหมด ในกรณีที่เวลาหมด มอนสเตอร์ตัวที่มีพลังชีวิต (Hp) เหลือมากกว่าจะถือว่าเป็นผู้ชนะในการต่อสู้นั้น

หลักการคิดของมอนสเตอร์ในการต่อสู้อธิบายในหัวข้อที่ '3.4 การออกแบบอัลกอริทึมที่ใช้ในการต่อสู้'

6. การผสมพันธุ์ของมอนสเตอร์

การผสมพันธุ์ของมอนสเตอร์มีเพื่อสร้างมอนสเตอร์รุ่นใหม่ที่มีวิวัฒนาการ และความก้าวหน้ามากกว่ารุ่นก่อน โดยการออกแบบกล่าวถึงในหัวข้อที่ '3.4 การออกแบบการผสมพันธุ์มอนสเตอร์'

7. การสร้างตัวละครมอนสเตอร์ศัตรู

เมื่อผู้เล่นเลือกเมนู 'ฝึกฝนต่อสู้' ภายในเมนู 'ฝึกฝน' มอนสเตอร์ของผู้เล่นจะเข้าสู่การต่อสู้กับมอนสเตอร์ศัตรูที่ระบบเป็นผู้สร้างขึ้น

ระบบทำการสร้างมอนสเตอร์ศัตรูที่มีระดับความเก่ง (Level) ระดับเดียวกับมอนสเตอร์ของผู้เล่น และทำการสุ่มชนิดและค่าพารามิเตอร์ต่างๆ ของมอนสเตอร์ศัตรู ภายในขอบเขตของค่า Space ในระดับ Level นั้นๆ

3.3 การออกแบบรูปแบบของไฟล์ที่ใช้เก็บข้อมูลตัวละคร

ข้อมูลของมอนสเตอร์ที่เป็นตัวละครของผู้เล่น มีการจัดเก็บข้อมูลในรูปแบบของไฟล์ รูปแบบของข้อมูลที่เก็บในไฟล์ มีรูปแบบดังนี้

name # hp # hpmax # sp # spmax # age # level # energy # status # full # thirsty #
str # qi # def # agi # dex # fit # vit # breath # Int # team # space # body # exp # exprequired !

name	=	ข้อมูลชื่อของตัวละคร
hp	=	ข้อมูลพลังชีวิตปัจจุบันของตัวละคร
hpmax	=	ข้อมูลพลังชีวิตสูงสุดของตัวละคร
sp	=	ข้อมูลพลังวิญญาณของตัวละคร
spmax	=	ข้อมูลพลังวิญญาณสูงสุดของตัวละคร
age	=	ข้อมูลอายุของตัวละคร
level	=	ข้อมูลค่า Level ของตัวละคร
energy	=	ข้อมูลค่าพลังงาน Energy ของตัวละคร
status	=	ข้อมูลค่าสถานะของตัวละคร
full	=	ข้อมูลค่า Full ของตัวละคร

thirsty = ข้อมูลค่า Thirsty ของตัวละคร

str, qi, def, agi, dex, fit, vit, breath, Int = ค่าพารามิเตอร์ของตัวละคร

team = ข้อมูลบอกว่ามอนสเตอร์เป็นมอนสเตอร์ของฝ่ายไหน

space = ข้อมูลค่า Space ขอบเขตการฝึกฝนของตัวละคร

body = ข้อมูลบอกชนิดร่างกายของตัวละคร

exp = ข้อมูลค่าประสบการณ์ (Exp) ของตัวละคร

exprequired = ข้อมูลค่าประสบการณ์ที่ต้องใช้เพื่อเลื่อนระดับ Level ถัดไปของตัวละคร

เครื่องหมาย # ใช้สำหรับแยกแยะชุดของข้อมูลแต่ละค่า

เครื่องหมาย ! หมายถึงการบ่งบอกว่าจบข้อมูลของตัวละครที่ทำการบันทึก

3.4 การผสมพันธุ์มอนสเตอร์

การผสมพันธุ์มอนสเตอร์จะต้องมีการเชื่อมต่อระหว่างเครื่องพ็อกเก็ตพีซีของผู้เล่น กับเครื่องเซิร์ฟเวอร์ โดยการจัดการการผสมพันธุ์มอนสเตอร์นั้นโปรแกรมแอพพลิเคชั่นที่เครื่องเซิร์ฟเวอร์จะเป็นผู้ดูแลและจัดการทั้งหมด

รูปแบบของข้อมูลในไฟล์ที่โปรแกรมแอพพลิเคชั่นในเครื่องเซิร์ฟเวอร์รับและส่งคืนแก่เครื่องพ็อกเก็ตพีซีของผู้เล่นอธิบายอยู่ในหัวข้อย่อยที่ 2. "โปรโตคอลในการติดต่อและรูปแบบข้อมูลที่ใช้" ในหัวข้อที่ 3.8.2 "สถาปัตยกรรมชั้นพื้นฐาน (Low Level Architecture)"

กระบวนการ Genetic Algorithm

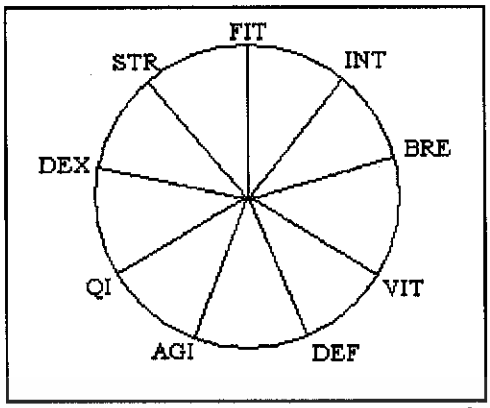
1. สร้างประชากรเริ่มต้น
   - กำหนดจำนวนประชากร 5 ตัว
   - กำหนดรูปแบบ chromosome ดังนี้

     $Gene \in \{0-255\}$

     Chromosome ประกอบด้วย Gene 9 ตัว

2. นิยาม Fitness Function

นำค่าความสามารถของตัวละครแต่ละค่ามาเรียงในวงกลมโดยค่าที่ผลถึงกันอยู่ติดกัน กำหนดค่า Fitness Function จากพื้นที่ภายในของค่าความสามารถทั้งหมดของตัวละคร



รูปที่ 3.1 แสดงการออกแบบตำแหน่งพารามิเตอร์ที่สัมพันธ์กัน

สูตร : $(FIT+INT)^2+ (INT+BRE)^2+ (BRE+VIT)^2+ (VIT+DEF)^2+ (DEF+ AGI)^2+ (AGI+ QI)^2+ (QI+ DEX)^2+ (DEX+ STR)^2+ (STR+ FIT)^2$

3. คำนวณ Fitness Value ของแต่ละ Chromosome
4. Selection พ่อพันธุ์ แม่พันธุ์ (Rank Selection) ขั้นตอนการเลือกจะขึ้นกับอัตราส่วนของ Fitness Value โดยที่โครโมโซมที่มีค่า Fitness Value มาก จะมีโอกาสได้รับเลือกมาก
5. ทำ GA operation
   - Crossover

     Crossover rate เป็น 0.1 คือมี Crossover Point เพียงจุดเดียว

     Crossover point เลือกจากการสุ่ม
   - Mutation

     เลือก Mutation rate = 0.1
6. นำประชากรที่ได้ใหม่ไปแทนที่ประชากรชุดเดิมโดยคัดเลือกโครโมโซมที่มีค่า Fitness Value น้อยที่สุดออก
7. ทำซ้ำ ที่ step 3 ทำซ้ำประมาณ 10 รอบ

เมื่อสิ้นสุดกระบวนการจะได้ข้อมูลของมอนสเตอร์ 'ลูก' ซึ่งเป็นผลลัพธ์จากกระบวนการผสมพันธุ์นั่นเอง แล้วเครื่องเซิร์ฟเวอร์จะทำการส่งข้อมูลนี้กลับไปให้ผู้เล่นเก็บข้อมูลของมอนสเตอร์ลูกไว้ในเมนู 'ห้องเก็บไข่'

3.5 การออกแบบอัลกอริทึมที่ใช้ในการต่อสู้

3.5.1 การต่อสู้ระหว่างผู้เล่นกับคอมพิวเตอร์

การต่อสู้ระหว่างผู้เล่นกับคอมพิวเตอร์สามารถทำได้โดยเลือกเมนูต่อสู้ภายในเกมบนเครื่องของผู้เล่นได้เอง

รูปที่ 3.2 แสดงการเข้าสู่การต่อสู้ระหว่างผู้เล่นกับคอมพิวเตอร์

3.5.2 การต่อสู้ระหว่างผู้เล่นกับผู้เล่นคนอื่น

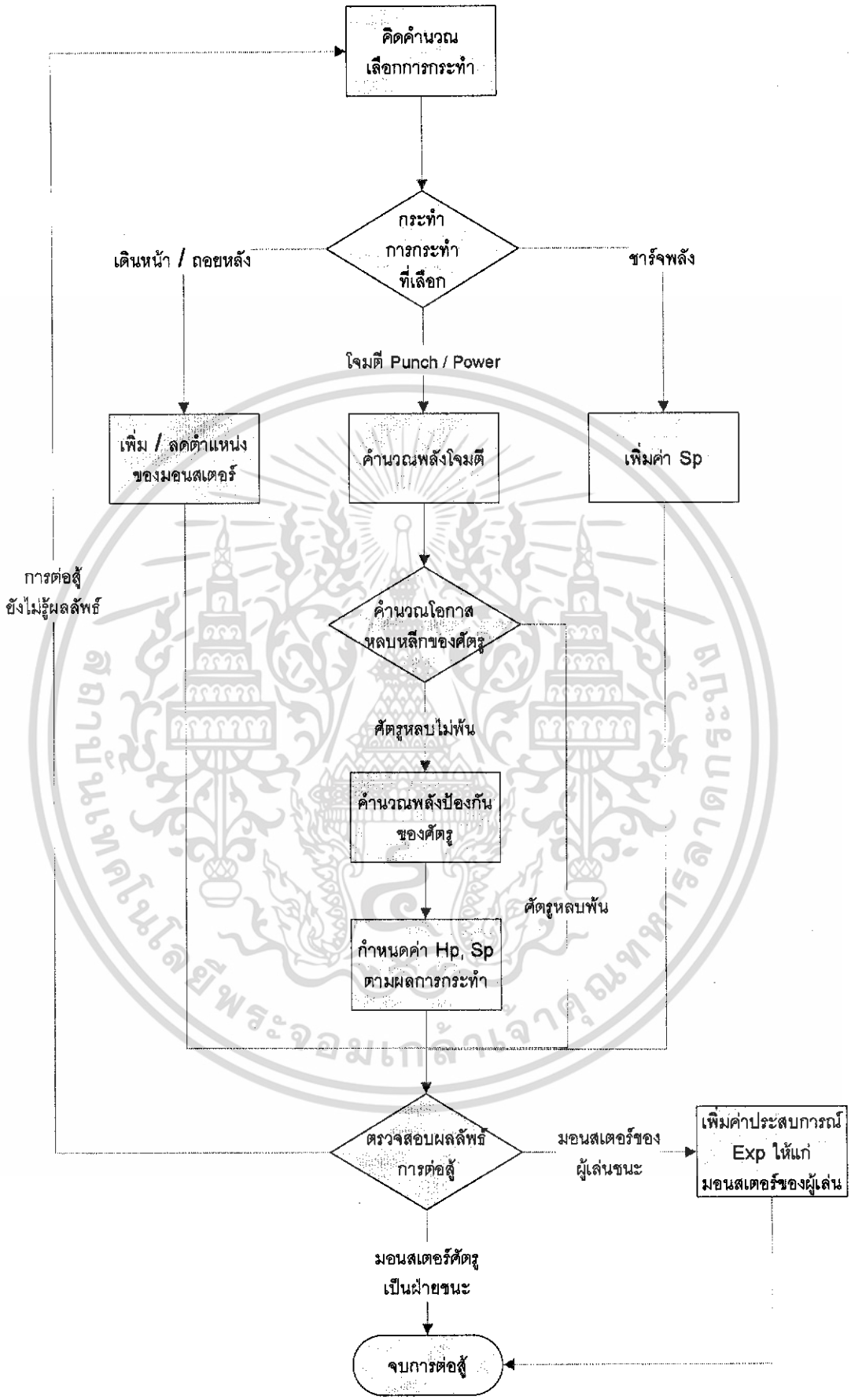
การต่อสู้ระหว่างผู้เล่นกับผู้เล่นคนอื่นต้องทำการเชื่อมต่อระหว่างเครื่องของผู้เล่นทั้งสองก่อน

การดำเนินการต่อสู้กระทำบนเครื่องของผู้เล่นเครื่องใดเครื่องหนึ่งเท่านั้น (ผู้เล่นที่ไม่ได้เป็นผู้จัดการต่อสู้สามารถชมการต่อสู้ได้บนเครื่องของผู้เล่นอีกคนหนึ่ง) เมื่อการต่อสู้จบลงระบบจึงจะส่งข้อมูลของมอนสเตอร์หลังจากต่อสู้กลับคืนไปยังเครื่องของผู้เล่นอีกฝ่ายหนึ่ง

รูปที่ 3.3 แสดงการเข้าสู่การต่อสู้ระหว่างผู้เล่นกับผู้เล่นคนอื่น

3.5.3 การทำงานของระบบต่อสู้

หลักการการต่อสู้ระหว่างมอนสเตอร์ (ทั้งแบบระหว่างผู้เล่นกับคอมพิวเตอร์ และแบบระหว่างผู้เล่นด้วยกันเอง) แสดงดังแผนภาพต่อไปนี้



รูปที่ 3.4 แสดงระบบการจัดการการต่อสู้

3.5.3.1 การคำนวณค่าพลังโจมตี

ปัจจัยที่มีผลต่อค่าพลังโจมตีมีดังนี้

- ค่า Str - ในการโจมตีระยะใกล้
- ค่า Qi - ในการโจมตีระยะไกล
- ค่า Dex - ส่งผลต่อขอบเขตของพลังทำลาย
- ค่า Fit - ส่งผลต่อโอกาสโจมตีด้วยพลังทำลายสูงสุด

หลักการจัดการคำนวณค่าพลังทำลาย แสดงด้วยแผนภาพ ดังนี้

รูปที่ 3.5 แสดงการหาค่าพลังทำลายจากค่าพลังโจมตี

อธิบายการทำงานได้ดังนี้

1. ตรวจสอบการกระทำ (action) ที่จะกระทำ
    a. การโจมตีใกล้ (action = Punch) คำนวณค่าพลังทำลายสูงสุด (maximum_damage) ดังนี้

        damage = Str x 3

    b. การโจมตีไกล (action = Power) คำนวณค่าพลังทำลายสูงสุด (maximum_damage) ดังนี้

        damage = Qi x 2

    ในขั้นตอนนี้จะได้ค่าพลังทำลายสูงสุด(maximum_damage)

2. คำนวณโอกาสได้ค่าพลังทำลายสูงสุดในการโจมตีครั้งนั้น
    a. สุ่มค่าตัวเลข (random_number) ระหว่างค่า 0 ถึง 255
    b. นำค่าพารามิเตอร์ Fit ของมอนสเตอร์มาเปรียบเทียบกับค่าตัวเลขที่ได้จากการสุ่ม (random_number) หากค่าพารามิเตอร์ Fit มีค่ามากกว่าจะถือว่าได้โอกาสโจมตีด้วยพลังทำลายสูงสุด แต่หากค่าพารามิเตอร์ Fit มีค่าน้อยกว่า ระบบจะไปคำนวณค่าพลังทำลายที่จะได้ต่อไป

จากหลักการนี้จะเห็นว่ายิ่งมีค่า Fit มากจะยิ่งมีโอกาสโจมตีด้วยพลังทำลายสูงสุดมากไปด้วย

3. คำนวณค่าพลังทำลายที่ได้จากการโจมตีครั้งนั้น
    a. ตรวจสอบโอกาสโจมตีด้วยพลังทำลายสูงสุด หากมีโอกาสจะโจมตีด้วยพลังทำลายสูงสุด
    b. ในกรณีที่ไม่มีโอกาสโจมตีด้วยพลังทำลายสูงสุด คำนวณค่าพลังทำลายต่ำสุด โดยนำค่าพารามิเตอร์ Dex มาใช้ แต่มีข้อจำกัดว่า ค่าพลังทำลายต่ำสุด (minimum_damage) มีค่าได้ไม่เกินครึ่งหนึ่งของค่าพลังทำลายสูงสุด (maximum_damage) ดังนี้

        ค่าพลังทำลายต่ำสุด (minimum_damage) = Dex x 2

        โดยที่ minimum_damage <= maximum_damage / 2

    c. สุ่มค่าพลังทำลาย (damage) ระหว่างค่าพลังทำลายต่ำสุด (minimum_damage) จนถึง ค่าพลังทำลายสูงสุด (maximum_damage)
    d. กำหนดให้ค่าพลังทำลายในการโจมตีครั้งนั้นเท่ากับค่าพลังทำลายที่สุ่มได้ (ค่า damage)

จากหลักการนี้จะเห็นว่ายิ่งมีค่า Dex มากจะยิ่งมีขอบเขตพลังทำลายยิ่งมากตาม

3.5.3.2 การคำนวณอัตราการหลบหลีก

หลักการรับการโจมตีจากมอนสเตอร์ศัตรูแสดงในแผนภาพต่อไปนี้

รูปที่ 3.6 แสดงการหาโอกาสหลบหลีกและการหาค่าพลังป้องกัน

การคำนวณหาอัตราการหลบหลีกมีขั้นตอนดังต่อไปนี้

1. สุ่มค่าตัวเลข (random_number) ตั้งแต่เลข 0 ถึง 255
2. เปรียบเทียบค่าตัวเลขที่ได้จากการสุ่ม กับค่าพารามิเตอร์ Agi ของตัวละคร

a. หากค่า Agi มากกว่าหรือเท่ากับ ถือว่าหลบหลีกได้ จะกำหนดค่าพลังทำลายจากการโจมตีของศัตรูให้เท่ากับ 0
b. หากค่า Agi น้อยกว่า ถือว่าหลบหลีกไม่ได้ ระบบจะทำการคำนวณค่าพลังป้องกันเป็นลำดับถัดไป

3.5.3.3 การคำนวณค่าพลังป้องกัน

จากรูปที่ 3.6 การคำนวณค่าพลังป้องกันมีหลักการดังต่อไปนี้

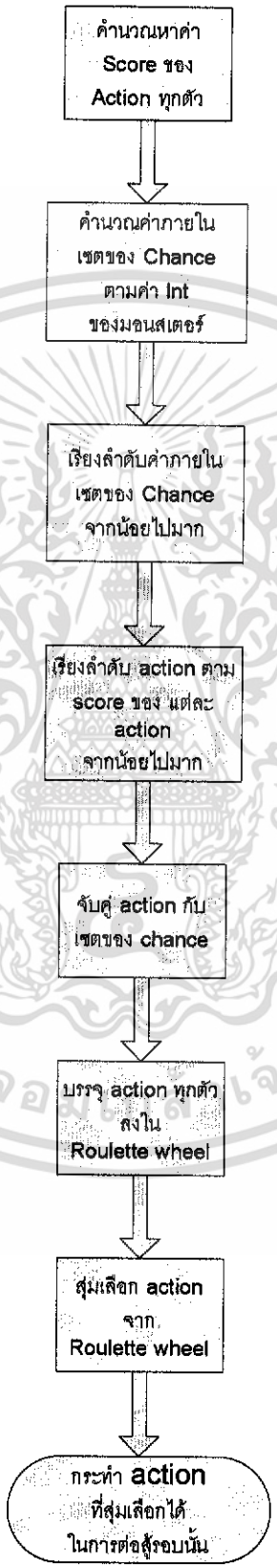
1. คำนวณค่าพลังป้องกันสูงสุดจากค่าพารามิเตอร์ Def ดังนี้

   ค่าพลังป้องกันสูงสุด (maximum_def) = (ค่าพารามิเตอร์ Def) x 3

2. สุ่มตัวเลขหาค่าพลังป้องกัน (Defense_value) จากค่าระหว่าง 0 ถึง พลังป้องกันสูงสุด (maximum_def)
3. กำหนดค่าพลังป้องกันของตัวมอนสเตอร์ต่อการถูกโจมตีครั้งนั้นเท่ากับค่าตัวเลขที่สุ่มได้

3.5.3.4 การเลือกการกระทำในการต่อสู้

หลักการคิดคำนวณของมอนสเตอร์ในการต่อสู้ แสดงในแผนภาพต่อไปนี้



รูปที่ 3.7 แสดงการคำนวณเลือก action ที่กระทำในการต่อสู้

รายละเอียดของการออกแบบมีดังต่อไปนี้

1. ในฉากต่อสู้มอนสเตอร์มีเซ็ตของการกระทำ (action) ดังนี้

Action (A) = {MoveForward, MoveBackward, Punch, Power, Charge}

โดยรายละเอียดได้กล่าวถึงในตารางที่ 3.2 แล้ว

ในฉากต่อสู้มีสิ่งแวดล้อม (Environment) ดังนี้

Environment (E) = {HPe, HPme, SPme, Time, De, Str, Qi, Agi, Def}

HPe = ค่าพลังชีวิตในขณะปัจจุบันของมอนสเตอร์ศัตรู

HPme = ค่าพลังชีวิตในขณะปัจจุบันของมอนสเตอร์

SPme = ค่าพลังวิญญาณในขณะปัจจุบันของมอนสเตอร์

Time = ค่าเวลาที่เหลืออยู่ในฉากต่อสู้

De = ระยะห่างระหว่างตัวมอนสเตอร์ กับตัวมอนสเตอร์ศัตรู แบ่งเป็น 3 ระยะ ดังนี้

- Near = ระยะใกล้ สามารถโจมตีระยะใกล้ถึง
- Medium = ระยะกลาง
- Far = ระยะไกลมาก

Str, Qi, Agi, Def = ค่าพารามิเตอร์ที่ใช้ในการต่อสู้ของมอนสเตอร์ (มีอธิบายรายละเอียดของแต่ละตัวในตารางที่ 3.1)

ในฉากต่อสู้ตัวละครมอนสเตอร์มีเป้าหมาย (Goal) ดังนี้

Goal (G) = {HPe, HPme, SPme, AtkOpp, DefOpp}

HPe = ลดค่าพลังชีวิต (HP) ของศัตรู

HPme = ปกป้องรักษาพลังชีวิต (HP) ของตัวเอง

SPme = ปกป้องรักษาพลังวิญญาณ (SP) ของตัวเอง

AtkOpp = สร้างโอกาสที่จะได้โจมตีคู่ต่อสู้

DefOpp = หลีกเลี่ยงสถานการณ์ที่จะถูกโจมตี

การหาค่าคะแนน (Score) ของการกระทำ action ใดๆ ในสภาวการณ์ขณะนั้นกระทำได้โดยนำฟังก์ชันที่ช่วยคำนวณผลการกระทำ action นั้นๆ มาใช้

Action \ Goal	HPe	HPme	SPme	AtkOpp	DefOpp	SUM
MoveForward	Func1(a1)	Func2(a1)	Func3(a1)	Func4(a1)	Func5(a1)	Sum(a1)
MoveBackward	Func1(a2)	Func2(a2)	Func3(a2)	Func4(a2)	Func5(a2)	Sum(a2)
Punch	Func1(a3)	Func2(a3)	Func3(a3)	Func4(a3)	Func5(a3)	Sum(a3)
Power	Func1(a4)	Func2(a4)	Func3(a4)	Func4(a4)	Func5(a4)	Sum(a4)
ChargeSP	Func1(a5)	Func2(a5)	Func3(a5)	Func4(a5)	Func5(a5)	Sum(a5)

ตารางที่ 3.4 แสดงการหาค่า Score ของแต่ละ Action

จากตารางที่ 3.4 ฟังก์ชัน Func(action) ทำการนำ action ประเภทนั้นมาคิดคำนวณกับสิ่งแวดล้อม (Environment) ในปัจจุบันว่าการกระทำ action นั้นๆ สนับสนุนให้ Goal มีความสำเร็จมากขึ้นเพียงใด

Func1(Action) ฟังก์ชันสำหรับส่งค่า Score ที่ได้จากการกระทำ action ใดๆ แล้วมีผลต่อ Goal::HPe มากน้อยแค่ไหน กลับมา

Func2(Action) ฟังก์ชันสำหรับส่งค่า Score ที่ได้จากการกระทำ action ใดๆ แล้วมีผลต่อ Goal::HPme มากน้อยแค่ไหน กลับมา

Func3(Action) ฟังก์ชันสำหรับส่งค่า Score ที่ได้จากการกระทำ action ใดๆ แล้วมีผลต่อ Goal::SPme มากน้อยแค่ไหน กลับมา

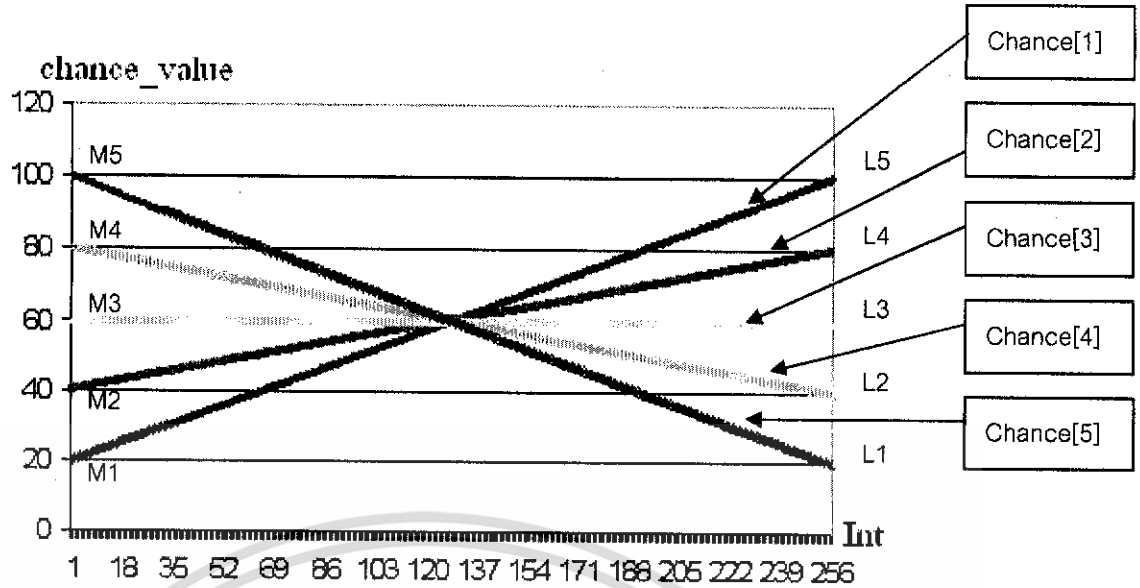
Func4(Action) ฟังก์ชันสำหรับส่งค่า Score ที่ได้จากการกระทำ action ใดๆ แล้วมีผลต่อ Goal::AtkOpp มากน้อยแค่ไหน กลับมา

Func5(Action) ฟังก์ชันสำหรับส่งค่า Score ที่ได้จากการกระทำ action ใดๆ แล้วมีผลต่อ Goal::DefOpp มากน้อยแค่ไหน กลับมา

เมื่อได้ค่า score ของ action ใดๆ จากการคำนวณในทุก Goal แล้วนำมาหาค่ารวมเฉพาะ action นั้นๆ action ใดที่มีผลรวมของ score มากที่สุดถือว่าเป็น action ที่ดีที่สุดในขณะนั้น

2. เซ็ตของ Chance คือ เซ็ตที่เก็บค่าโอกาสที่จะกระทำ action ที่ดีมาก – น้อย ต่างกันไป โดยขึ้นกับค่าพารามิเตอร์ความฉลาด (Int) ของตัวมอนสเตอร์ ในที่นี้มีประกอบด้วย

{Chance[1], Chance[2], Chance[3], Chance[4], Chance[5]}



รูปที่ 3.8 แสดงค่าเซ็ตของค่า chance

จากรูปที่ 3.8 ค่า Chance[1] เก็บค่า Chance ที่มีประสิทธิภาพสูงสุด ซึ่งจะมีค่าเพิ่มขึ้นเมื่อค่าพารามิเตอร์ Int เพิ่มขึ้น โดยที่ประสิทธิภาพของค่า Chance สามารถเรียงลำดับได้ดังนี้

Chance[1] > Chance[2] > Chance[3] > Chance[4] > Chance[5]

โดยวัดตามค่าที่เพิ่มขึ้นเมื่อค่าพารามิเตอร์ Int ของมอนสเตอร์เพิ่มมากขึ้น ในระบบเกมนี้กำหนดให้ Chance[1] มีหน้าที่เก็บ action ที่ดีที่สุด และ Chance[2] มีหน้าที่เก็บ action ที่ดีรองลงมาเป็นอันดับสอง และไล่ลงมาเรื่อยๆ จนถึง Chance[5] มีหน้าที่เก็บ action ที่แย่ที่สุด

ในสภาวการณ์ตอนนั้น Chance[] ใดมีค่ามากที่สุดจะขึ้นกับค่า Int ตามสมการหาความชันต่อไปนี้

Chance[1] = (((L5 - M1) / 255) * Int) + M1

Chance[2] = (((L4 - M2) / 255) * Int) + M2

Chance[3] = (((L3 - M3) / 255) * Int) + M3

Chance[4] = (((L2 - M4) / 255) * Int) + M4

Chance[5] = (((L1 - M5) / 255) * Int) + M5

3. เมื่อได้ชุดข้อมูล Score รวมของแต่ละ Action และค่า Chance[] แล้ว ระบบทำการจับคู่ระหว่าง Action กับ Chance[] ที่เหมาะสม ดังนี้

Chance[1] จับคู่กับ Action ที่มีผลรวม Score มากเป็นอันดับ 1

Chance[2] จับคู่กับ Action ที่มีผลรวม Score มากเป็นอันดับ 2

Chance[3] จับคู่กับ Action ที่มีผลรวม Score มากเป็นอันดับ 3

Chance[4] จับคู่กับ Action ที่มีผลรวม Score มากเป็นอันดับ 4

Chance[5] จับคู่กับ Action ที่มีผลรวม Score มากเป็นอันดับ 5 หรือน้อยที่สุด

4. นำค่าที่เก็บอยู่ใน Chance[] ไปใส่ใน Roulette Wheel ขนาด 15 ช่อง ดังนี้

- ช่อง 0 ถึง 4 จำนวน 5 ช่อง บรรจุ Action ภายใน Chance[] ที่มีค่า Chance มากที่สุด
- ช่อง 5 ถึง 8 จำนวน 4 ช่อง บรรจุ Action ภายใน Chance[] ที่มีค่า Chance มากเป็นลำดับที่ 2
- ช่อง 9 ถึง 11 จำนวน 3 ช่อง บรรจุ Action ภายใน Chance[] ที่มีค่า Chance มากเป็นลำดับที่ 3
- ช่อง 12 ถึง 13 จำนวน 2 ช่อง บรรจุ Action ภายใน Chance[] ที่มีค่า Chance มากเป็นลำดับที่ 4
- ช่อง 14 จำนวน ช่อง บรรจุ Action ภายใน Chance[] ที่มีค่า Chance น้อยที่สุด

5. สุ่มเลือก Action ที่อยู่ภายใน Roulette Wheel เป็น Action ที่กระทำในการคิดตัดสินใจเลือก Action ที่จะใช้ในการต่อสู้รอบนั้นๆ

6. สรุป : หากมอนสเตอร์มีค่าพารามิเตอร์ความฉลาด (ค่า Int) สูง จะส่งผลให้ค่า Chance[1] ยิ่งมีค่ามาก ซึ่งส่งผลให้ Action ที่ดีที่สุด (มี Score สูงที่สุด) ถูกบรรจุใน Roulette Wheel จำนวน 5 ช่อง (มีอยู่ในที่ที่สำหรับสุ่มเลือกเป็นจำนวนมาก) ส่งผลให้โอกาสที่มอนสเตอร์จะเลือก Action ที่ดีที่สุดมากขึ้นไปด้วย

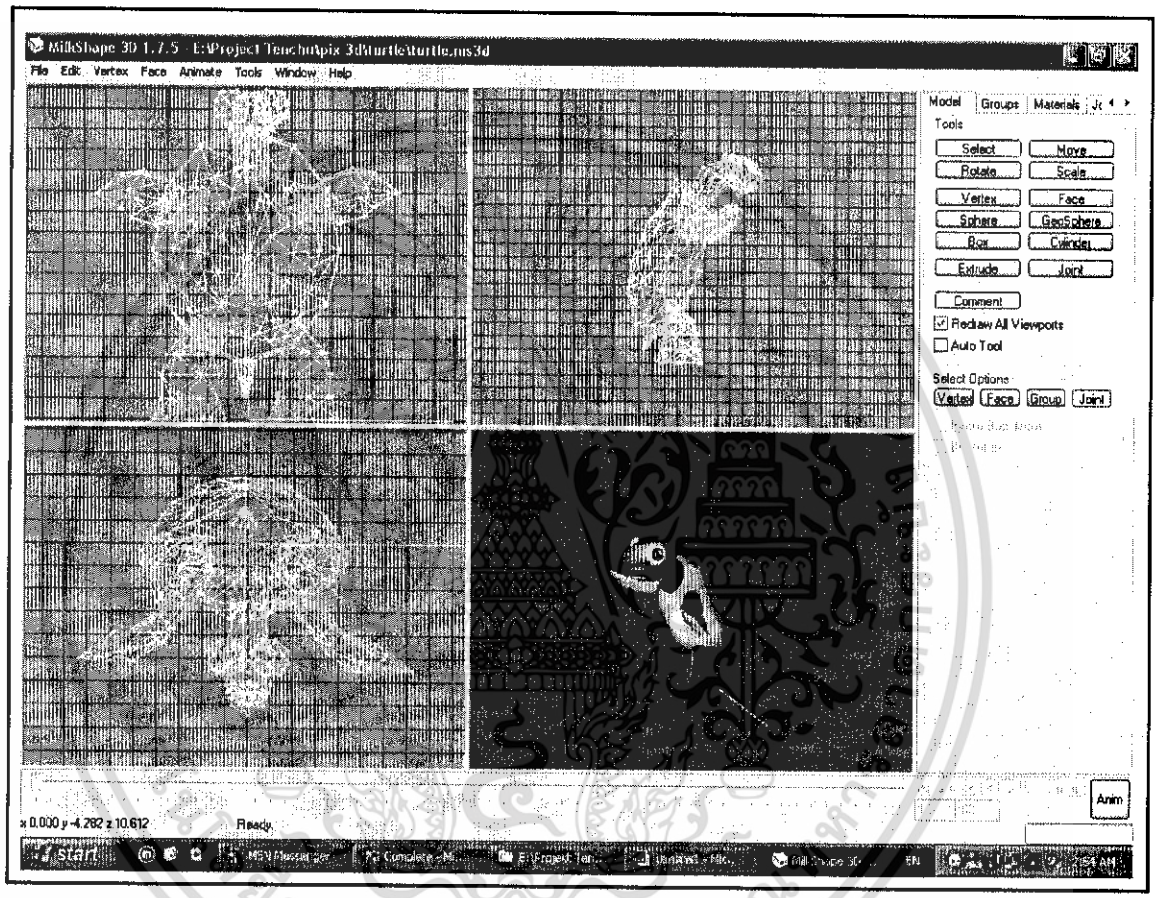
ในขณะที่ ค่า Int มาก จะส่งผลให้ ค่า Chance[5] มีค่าต่ำ ซึ่งส่งผลให้ Action ที่แย่ที่สุด (มีค่าScore น้อยที่สุด) มีโอกาสถูกเลือกน้อยมาก

3.6 การออกแบบกราฟิกสามมิติ

3.6.1 การออกแบบส่วนโมเดล 3 มิติ

การสร้างโมเดล 3 มิติภายในเกมแบ่งออกเป็น 4 ส่วนดังนี้

1. ส่วนในการสร้างโมเดล 3 มิติใช้โปรแกรม Milkshape 3D ในการพัฒนา



รูปที่ 3.9 โปรแกรม Milkshape3D

สร้างรูปโมเดลโดยแบ่งโมเดลออกเป็นกลุ่มย่อยหลายๆกลุ่ม จากนั้นสร้างโครงตามแนวทางดังต่อไปนี้

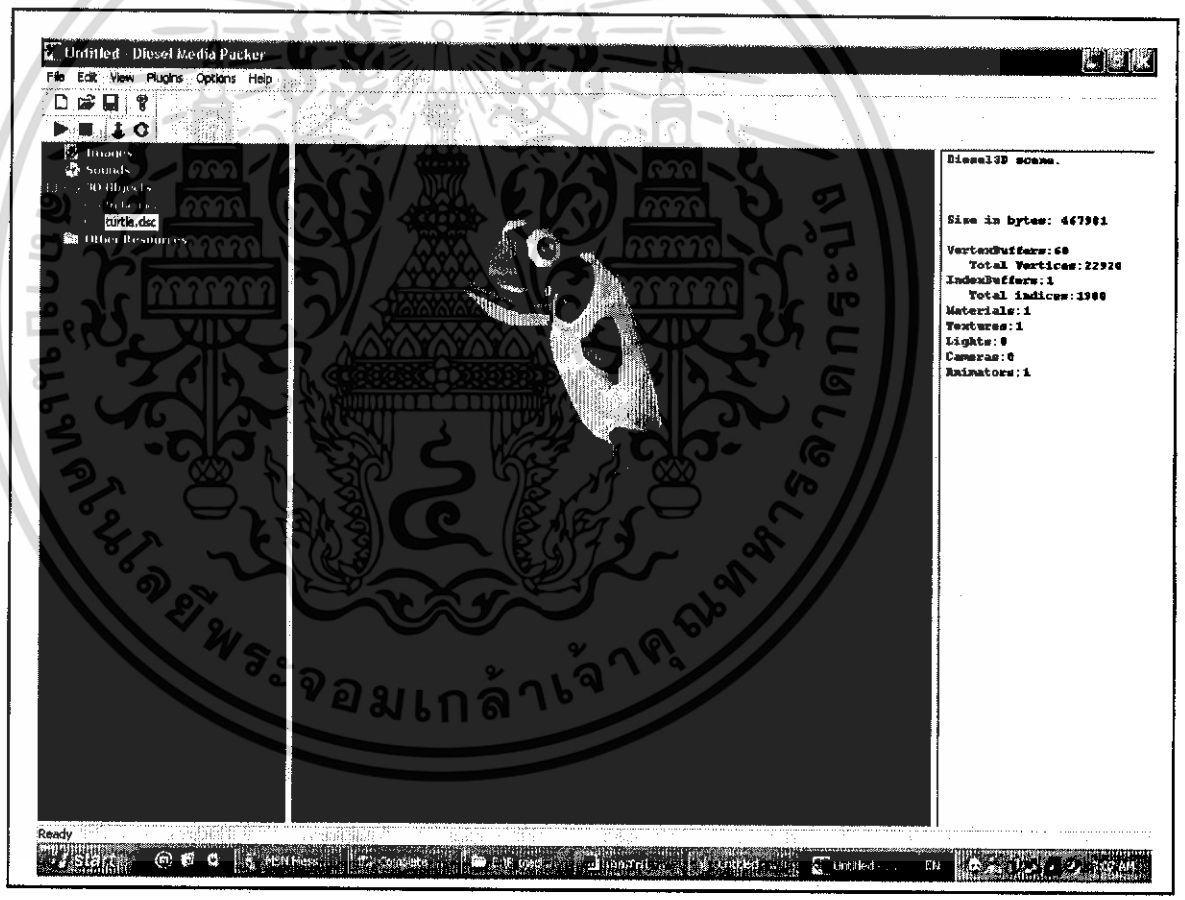
- กำหนดจุด 3 จุดเพื่อใช้เป็นพื้นผิวในหนึ่งกลุ่มย่อย แล้วทำการสร้างพื้นผิวจนครบตัวโมเดล
- ใช้ฟังก์ชั่นพื้นฐานสร้างเป็น ทรงกลม ทรงสี่เหลี่ยม ทรงกระบอก แล้วทำการย้าย ย่อ ขยาย จุด จนได้โครงที่ต้องการ
- หาตัวละครที่เป็น Open Source มาทำการแก้ไข อาจใช้ทั้งหมด หรือตัดมาเพียงบางส่วนแล้วมาประกอบกับส่วนอื่นๆ

2. ส่วนในการแต่งพื้นผิว จะอยู่ในแท็ป Material หลังจาก assign โมเดลให้กับ Texture แล้ว สามารถกำหนดตำแหน่งอย่างละเอียดได้ด้วย Texture Coordinate Editor

3. ส่วนในการทำภาพเคลื่อนไหว จำเป็นต้องสร้าง ข้อต่อ (Joint) ให้กับโมเดลก่อนพร้อมทั้ง Assign โมเดลแต่ละส่วนให้กับข้อต่อนั้นๆ เมื่อเข้าสู่ Animation Mode ให้ทำการขยับ และหมุน ข้อต่อ ตามแต่ละการกระทำนั้นๆ

4. การแปลงไฟล์โมเดล 3 มิติให้อยู่ในรูปแบบที่นำไปใช้งานในเกมได้

ให้ทำการ Export ไฟล์เป็นสกุล .MD2 แล้วนำไฟล์ดังกล่าวไปปรับแก้ในโปรแกรม 'DieselMediaPacker' เมื่อได้โมเดลที่ต้องการแล้วให้ Export Itemไปเป็นไฟล์สกุล .DSC (Diesel3D Scene Converter)



รูปที่ 3.10 โปรแกรม Diesel Media Packer

3.6.2 วิธีการนำโมเดล 3 มิติไปใช้งานบนคอมพิวเตอร์แบบพกพา

- สร้างหน้าต่างของโปรแกรมขึ้นมาโดยใช้ คลาส IDieselApplication โดยกำหนดหลายละเอียดต่างๆให้หน้าต่างและเริ่มต้นทำงานด้วย
- สร้างหน้าต่างควบคุมการแสดงภาพโดยใช้คลาส CDieselD3DDevice และกำหนดตัวแปรในการแสดงผลเช่นกำหนดค่า z-buffer
- ทำการสร้าง path ไปยัง file ของ animation แล้วทำการโหลดข้อมูลไว้ในคลาส CDiesel3DScene
- เปลี่ยนแปลงการแสดงผลของ animation ไปตาม frametime ของ animation ทำให้ภาพกราฟิกแสดงไปเรื่อยๆ
- ทำการลบหน้าจอเดิมออกจาก buffer แล้วทำการเขียนภาพใหม่ลงไปในหน้าจอ

3.7 การออกแบบเกม

3.7.1 Use case diagram

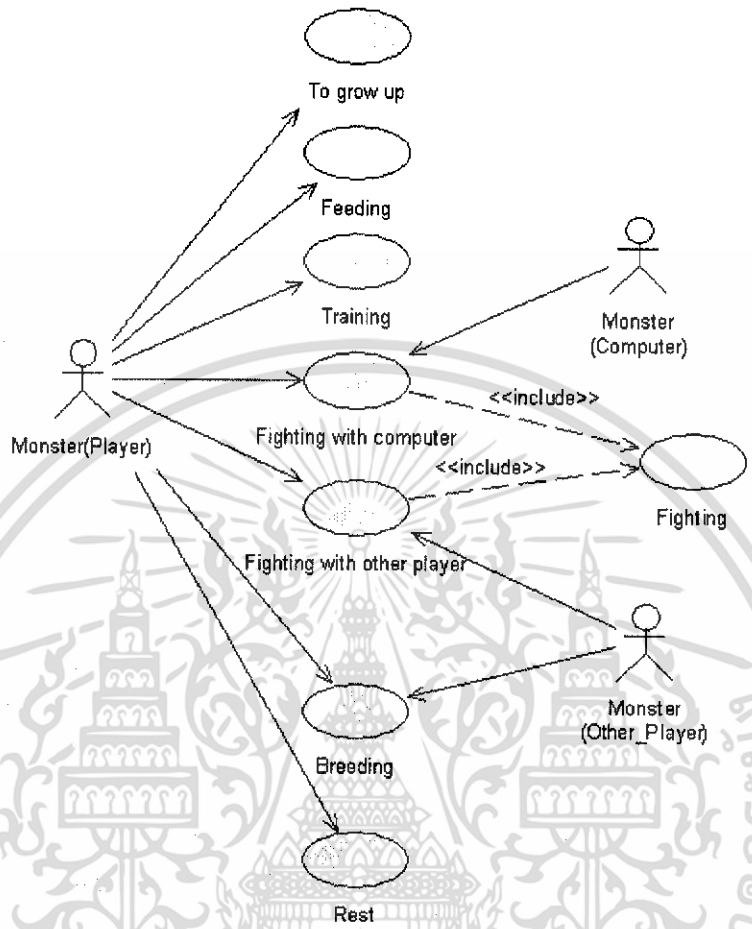
หัวข้อนี้แสดง Use case diagram ในมุมมอง 2 ระดับ คือ

1) มุมมองระดับผู้เล่น แสดงสิ่งที่ผู้เล่นสามารถกระทำกับระบบได้
2) มุมมองระดับภายในเกม แสดงสิ่งที่ตัวละครมอนสเตอร์ภายในเกมสามารถกระทำกับระบบได้

3.7.1.1 Use case diagram ในมุมมองผู้เล่น

รูปที่ 3.11 แสดง Use case diagram ของระบบ

3.7.1.2 Use case diagram ในมุมมองระดับภายในเกม



รูปที่ 3.12 แสดง Use case diagram ของเกม

3.7.1.3 Use Case Description

หัวข้อนี้อธิบายรายละเอียดของแต่ละ Use case จาก Use case diagram

Use Case :

Play Game

Short Description :

การเล่นเกม

Actor :

Player, Other_Player

Pre – Conditions :

1. ผู้เล่นมีอุปกรณ์พ็อกเก็ตพีซี ที่มีระบบสื่อสารไร้สาย
2. ผู้เล่นมีแอพพลิเคชั่นเกมเลี้ยงมอนสเตอร์ 3 มิติ

Post – Conditions :

1. ผู้เล่นสามารถเล่นเกมเลี้ยงมอนสเตอร์ 3 มิตินี้ได้
2. ผู้เล่นสามารถเล่นเกมเลี้ยงมอนสเตอร์ 3 มิตินี้ร่วมกับผู้เล่นคนอื่นได้

Main Flow :

1. ผู้เล่นลงโปรแกรมแอพพลิเคชั่นเลี้ยงมอนสเตอร์ 3 มิติ
2. ผู้เล่นเล่นเกม โดยเลือกเมนูต่างๆ ได้ตามใจ
3. ในกรณีที่ต้องการเล่นร่วมกับผู้เล่นคนอื่นต้องทำการเปิดระบบติดต่อไร้สาย

Alternate Flow(s) :

-

Exception Flow(s) :

-

Use Case :

To grow up

Short Description :

การเจริญเติบโตของตัวละครมอนสเตอร์

Actor :

Monster(Player)

Pre – Conditions :

ตัวละครมีค่าประสบการณ์และระดับความเก่งถึงระดับหนึ่ง

Post – Conditions :

ตัวละครเปลี่ยนร่างตามสายชนิดของตัวละครที่ได้รับการฝึกฝนมา

Main Flow :

1. ผู้เล่นฝึกฝนเพิ่มค่าประสบการณ์จนระดับความเก่ง (Level) เพิ่มขึ้น
2. ระบบตรวจสอบสถานะและค่าพารามิเตอร์ของตัวละคร
3. ระบบเปลี่ยนร่างตัวละครตามชนิดของตัวละคร

Alternate Flow(s) :

-

Exception Flow(s) :

-

Use Case :

Feeding

Short Description :

การกินอาหารของมอนสเตอร์

Actor :

Monster(Player)

Pre – Conditions :

-

Post – Conditions :

ระดับค่า Full หรือระดับค่า Thirsty ของตัวละครเพิ่มขึ้น

Main Flow :

1. ผู้เล่นเลือกเมนูให้อาหารจากหน้าจอหลัก
2. ผู้เล่นเลือกชนิดของอาหารที่ต้องการให้มอนสเตอร์กิน
3. ในกรณีที่ผู้เล่นเลือกชนิดเป็นอาหารระดับค่า Full ของตัวละครเพิ่มขึ้น
4. ในกรณีที่ผู้เล่นเลือกชนิดเป็นเครื่องดื่มระดับค่า Thirsty ของตัวละครเพิ่มขึ้น

Alternate Flow(s) :

-

Exception Flow(s) :

-

Use Case :

Training

Short Description :

การฝึกฝนของตัวละคร

Actor :

Monster(Player)

Pre – Conditions :

ตัวละครมีค่าพลังงาน (Energy) เพียงพอกับเมนูฝึกฝนนั้นๆ

Post – Conditions :

1. ค่าประสบการณ์ (Exp) ของตัวละครเพิ่มขึ้น

2. ค่าพารามิเตอร์ความเก่งของตัวละครเพิ่มขึ้นตามเมนูการฝึกฝนที่ผู้เล่นเลือก
3. ค่าพลังงาน (Energy) ของตัวละครลดลงตามเมนูการฝึกฝนที่ผู้เล่นเลือก

Main Flow :

1. ผู้เล่นเลือกเมนูฝึกฝนจากหน้าจอหลัก
2. ผู้เล่นเลือกชนิดประเภทการฝึกฝน
3. ระบบตรวจสอบสถานะค่าพลังงาน (Energy) ของตัวละคร [E1]
4. ระบบเพิ่มค่าพารามิเตอร์ตามประเภทการฝึกที่ผู้เล่นเลือก [E2]
5. ระบบเพิ่มค่าประสบการณ์ให้แก่ตัวละครของผู้เล่น [A1]
6. ระบบลดค่าพลังงาน (Energy) ของตัวละคร

Alternate Flow(s) :

[A1] ค่าประสบการณ์ของตัวละครสามารถเพิ่มได้จากการต่อสู้ด้วย

Exception Flow(s) :

[E1] ตัวละครมีค่าพลังงาน (Energy) ไม่เพียงพอต่อการฝึกฝน

[E2] ตัวละครมีค่าพารามิเตอร์ Space เต็มแล้ว ค่าพารามิเตอร์จึงไม่สามารถเพิ่มได้

Use Case :

Fighting with computer

Short Description :

การต่อสู้ระหว่างตัวละครมอนสเตอร์ของผู้เล่นกับตัวละครมอนสเตอร์ศัตรูจากระบบ

Actor :

Monster(Player), Monster(Computer)

Pre – Conditions :

ตัวละครของผู้เล่นมีค่าพลังชีวิต (Hp) มากกว่า 0

Post – Conditions :

ตัวละครที่เป็นผู้ชนะการต่อสู้ได้รับค่าประสบการณ์หลังจบการต่อสู้

Main Flow :

1. ผู้เล่นเลือกเมนู 'ฝึกฝนต่อสู้' ซึ่งอยู่ภายในเมนู 'ฝึกฝน' จากหน้าจอหลัก
2. ระบบตรวจสอบสถานะของตัวละคร [E1]
3. ระบบสร้างตัวละครศัตรูให้ตรงตามระดับของตัวละครของผู้เล่น
4. ระบบเข้าสู่การต่อสู้
5. ระบบตรวจสอบผลการต่อสู้

6. ระบบเพิ่มค่าประสบการณ์ให้แก่ตัวละครของผู้เล่นในกรณีที่ต่อสู้ชนะ [A1]

Alternate Flow(s) :

[A1] ค่าประสบการณ์เพิ่มได้อีกทางโดยการฝึกฝนตัวละคร

Exception Flow(s) :

[E1] ตัวละครมีพลังชีวิต (Hp) เท่ากับ 0

Use Case :

Fighting with other player

Short Description :

การต่อสู้ระหว่างตัวละครมอนสเตอร์ของผู้เล่นกับตัวละครมอนสเตอร์ของผู้เล่นคนอื่น

Actor :

Monster(Player), Monster(Player)

Pre – Conditions :

ตัวละครของผู้เล่นทั้งคู่มีค่าพลังชีวิต (Hp) มากกว่า 0

Post – Conditions :

ตัวละครที่เป็นผู้ชนะการต่อสู้ได้รับค่าประสบการณ์หลังจบการต่อสู้

Main Flow :

1. ผู้เล่นทั้งสองเปิดการเชื่อมต่อไร้สายที่อุปกรณ์พ็อกเก็ตพีซี
2. ผู้เล่นทั้งสองเลือกเมนู 'ต่อสู้' จากหน้าจอหลัก
3. ผู้เล่นทั้งสองเลือกการเชื่อมต่อระหว่างผู้เล่น
4. ระบบตรวจสอบสถานะของตัวละคร [E1]
5. ระบบเข้าสู่การต่อสู้
6. ระบบตรวจสอบผลการต่อสู้
7. ระบบเพิ่มค่าประสบการณ์ให้แก่ตัวละครของผู้เล่นที่ต่อสู้ชนะ [A1]
8. ระบบปิดการเชื่อมต่อระหว่างผู้เล่น

Alternate Flow(s) :

[A1] ค่าประสบการณ์เพิ่มได้อีกทางโดยการฝึกฝนตัวละคร

Exception Flow(s) :

[E1] ตัวละครของผู้เล่นคนใดคนหนึ่งมีพลังชีวิต (Hp) เท่ากับ 0

Use Case :

Breeding

Short Description :

การผสมพันธุ์ระหว่างมอนสเตอร์ของผู้เล่น ซึ่งระบบรองรับการเชื่อมต่อจากผู้เล่นได้ไม่เกิน 5 เครื่อง และไข่มอนสเตอร์ที่ได้จากการผสมพันธุ์จะถูกส่งไปให้กับผู้เล่นที่ทำการเชื่อมต่อกับเครื่องเซิร์ฟเวอร์เป็นคนแรก

Actor :

Monster(Player), Monster(Other_Player)

Pre – Conditions :

-

Post – Conditions :

ไข่ของมอนสเตอร์ที่ได้จากการผสมพันธุ์

Main Flow :

1. ผู้เล่นทุกคนเลือกเมนู 'ผสมพันธุ์' จากหน้าจอหลัก
2. ระบบทำการเชื่อมต่อระหว่างเครื่องพ็อกเก็ตพีซีของผู้เล่นทุกคนกับเครื่องเซิร์ฟเวอร์ซึ่งเป็นเครื่องคอมพิวเตอร์
3. ระบบบันทึกข้อมูลมอนสเตอร์ของผู้เล่นทุกคน
4. ระบบทำการผสมพันธุ์มอนสเตอร์ทุกตัว จนได้ไข่มอนสเตอร์ที่บรรจุลูกมอนสเตอร์ที่ได้จากการผสมพันธุ์
5. ระบบเลือกผู้เล่นที่เชื่อมต่อกับเครื่องเซิร์ฟเวอร์เป็นคนแรกให้เป็นผู้ได้รับไข่มอนสเตอร์ โดยข้อมูลไข่มอนสเตอร์ใหม่นี้จะทับข้อมูลไข่มอนสเตอร์ที่เครื่องผู้เล่นคนนั้นมีอยู่ก่อนหน้า
6. ระบบส่งข้อมูลไข่มอนสเตอร์ไปให้แก่ผู้เล่นที่เลือกให้เป็นผู้ได้รับไข่
7. ระบบปิดการเชื่อมต่อระหว่างเครื่องเซิร์ฟเวอร์กับเครื่องของผู้เล่นทุกคน

Alternate Flow(s) :

-

Exception Flow(s) :

-

Use Case :

Rest

Short Description :

มอนสเตอร์ทำการพักผ่อน

Actor :

Monster(Player)

Pre – Conditions :

-

Post – Conditions :

ค่าพลังชีวิต (Hp) , ค่าพลังวิญญาณ (Sp) และค่าพลังงาน (Energy) ของมอนสเตอร์ค่อยๆ ฟื้นคืนตลอดเวลาที่ทำการพักผ่อน

Main Flow :

1. ผู้เล่นเลือกเมนู 'พักผ่อน' จากหน้าจอหลัก

2. ระบบเพิ่มค่าพารามิเตอร์ต่างๆ ของตัวละคร ได้แก่ ค่าพลังชีวิต (Hp) , ค่าพลังวิญญาณ (Sp) และค่าพลังงาน (Energy)

3. ระบบออกจากโหมดพักผ่อนเมื่อผู้เล่นกดปุ่มใดๆ บนหน้าจอ

Alternate Flow(s) :

-

Exception Flow(s) :

-

3.7.2 Class diagram

หัวข้อนี้แสดง Class diagram ของระบบ ซึ่งประกอบไปด้วย Class diagram จาก 2 ระดับ ดังนี้

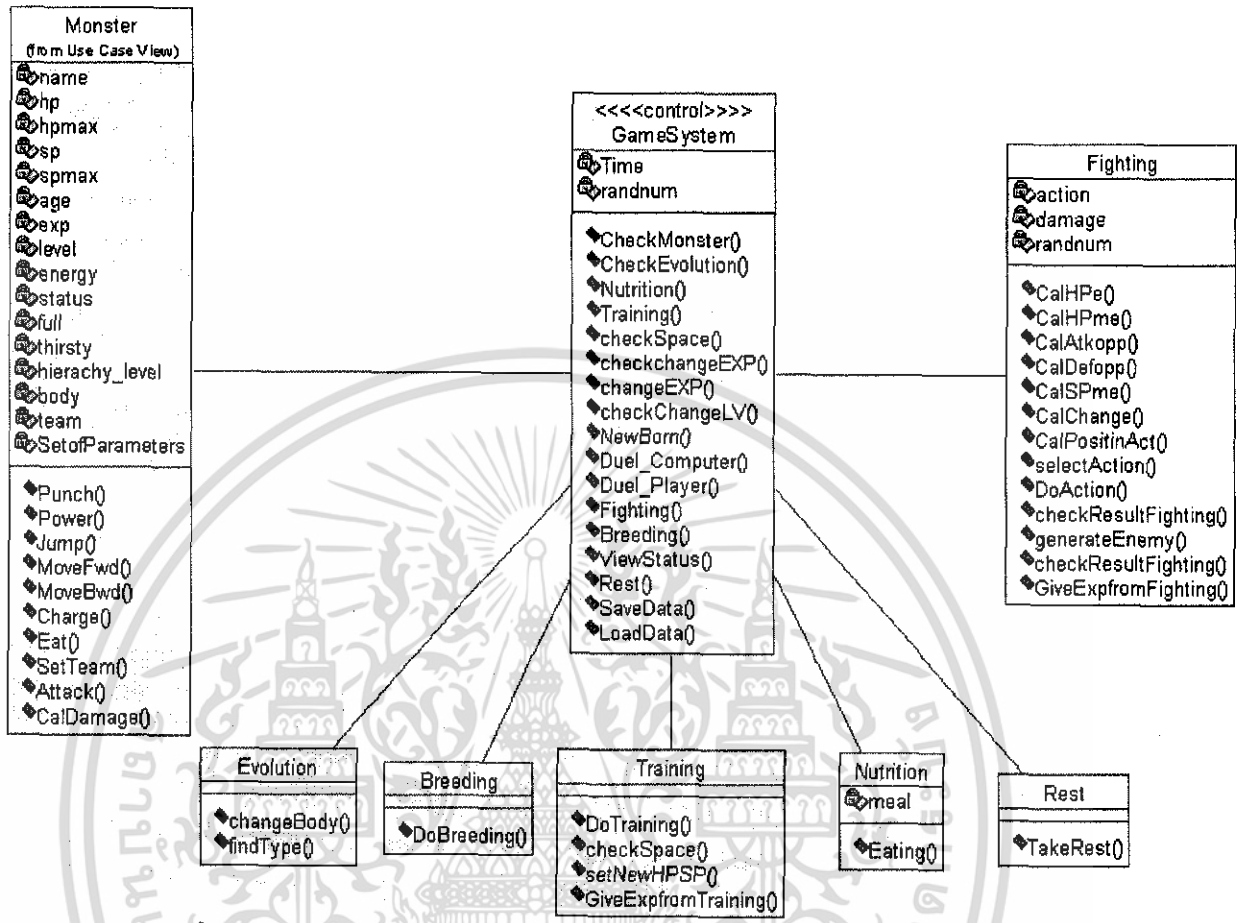
1) Class diagram ในมุมมองภายในเกม ระดับการวิเคราะห์ (Analysis)
2) Class diagram ในมุมมองภายในเกม ระดับการวิเคราะห์ (Design)

Class diagram ในระดับการวิเคราะห์ระบบ (Analysis) เป็นไดอะแกรมที่ได้จากการวิเคราะห์ระบบ ใช้เพื่อจำลองความเข้าใจในระบบ และใช้เป็นแหล่งข้อมูล (Resource) ประกอบการออกแบบ (Design) ระบบในระดับถัดไป

Class diagram ในระดับการออกแบบระบบ (Design) เป็นไดอะไรแกรมที่ได้จากการออกแบบระบบที่นำไปใช้จริง โดยนำข้อมูลจาก Class diagram ที่ได้จากการวิเคราะห์ระบบ

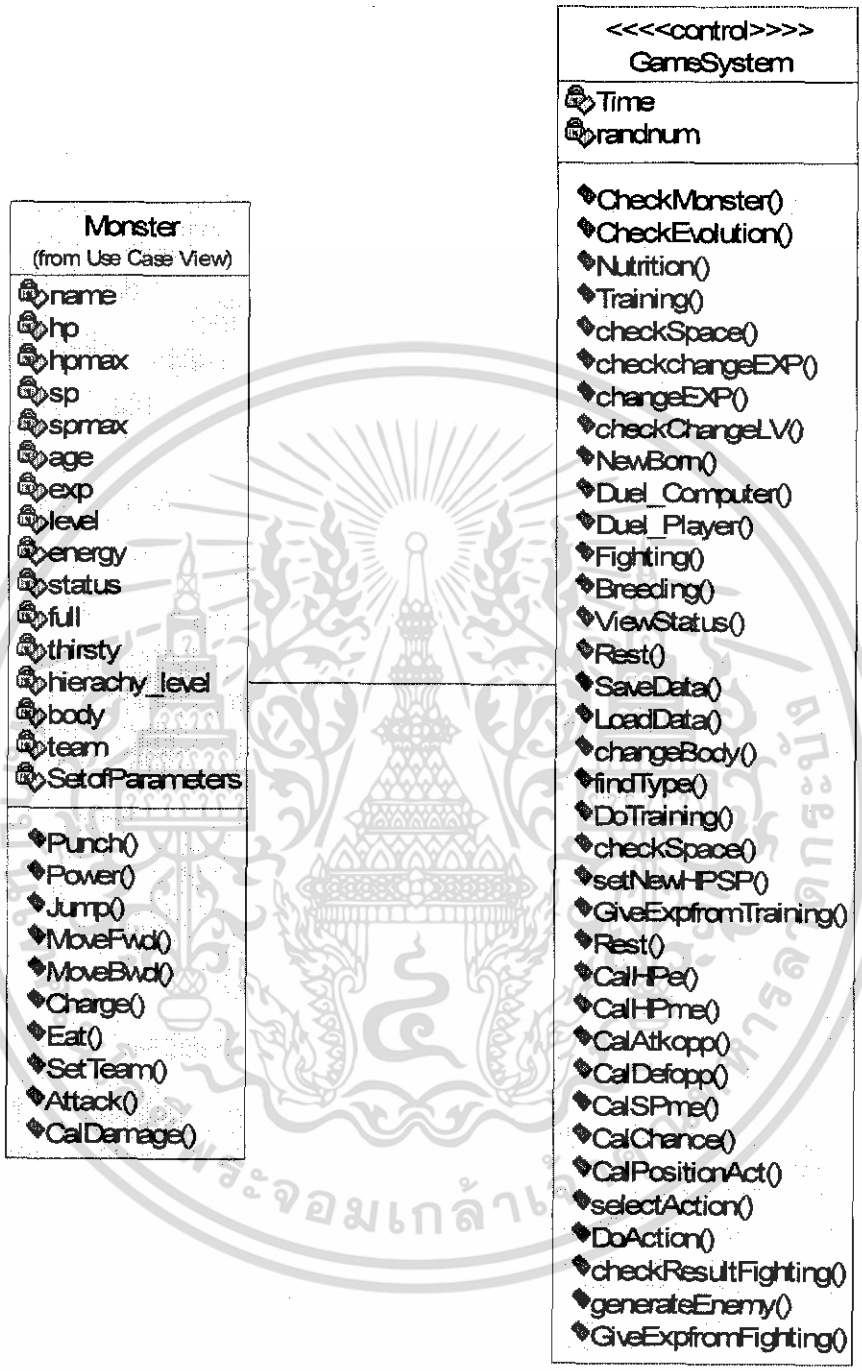
ไดอะแกรมในระดับการออกมีความแตกต่างจากไดอะแกรมในระดับการวิเคราะห์ เนื่องจากความเหมาะสมในการนำไปใช้พัฒนาระบบจริง แต่ยังอ้างอิงข้อมูลที่ได้จากขั้นตอนวิเคราะห์ระบบ

1. Class diagram ในมุมมองภายในเกม ระดับการวิเคราะห์ (Analysis)



รูปที่ 3.13 แสดงคลาสไดอะแกรมของเกมในระดับการวิเคราะห์ระบบ

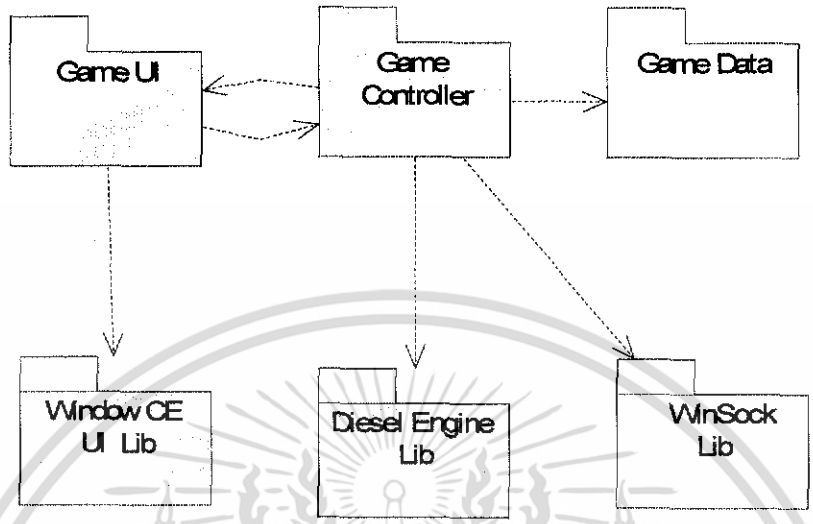
2. Class diagram ในมุมมองภายในเกม ระดับการออกแบบ (Design)



รูปที่ 3.14 แสดงคลาสไดอะแกรมของเกมในระดับการออกแบบระบบ

3.8 การออกแบบสถาปัตยกรรมภายในเกม

สถาปัตยกรรมของระบบในระดับของระบบเกม แสดงในภาพต่อไปนี้



รูปที่ 3.15 Package diagram แสดงสถาปัตยกรรมของระบบภายในเกม

ระบบของเกมมีการติดต่อระหว่างส่วนประกอบต่างๆ ดังนี้

1. Game Controller ทำหน้าที่ควบคุมดูแลจัดการการทำงานภายในระบบทั้งหมด เป็นเสมือนตัวกลางที่เชื่อมการทำงานระหว่างส่วนต่างๆ ภายในระบบ
2. Game Data ทำหน้าที่ในการจัดการส่วนของข้อมูลที่ใช้ภายในเกมทั้งหมด
3. Game UI ทำหน้าที่จัดการการติดต่อระหว่างผู้เล่นกับระบบตัวเกม
4. Window CE UI Lib ไลบรารี่ของระบบปฏิบัติการบนอุปกรณ์พ็อกเก็ตพีซี ทำหน้าที่จัดการการกระทำของผู้เล่นที่กระทำกับอุปกรณ์พ็อกเก็ตพีซี แล้วเชื่อมข้อมูลให้กับ Game UI ในแอพพลิเคชั่น
5. Winsock Lib ทำหน้าที่จัดการการเชื่อมต่อเครือข่าย และส่งข้อมูลผ่านเครือข่าย
6. Diesel Engine Lib ทำหน้าที่จัดการประมวลผลการแสดงผลภาพสามมิติและระบบเสียงที่ใช้ในแอพพลิเคชั่น

การทำงานของระบบภายในเกมมีลักษณะการทำงาน ดังต่อไปนี้

1. เมื่อผู้เล่นเปิดเกมขึ้นมา ระบบเกมจะทำการโหลดรูปภาพและข้อมูลที่ต้องใช้ในเกม เช่น ข้อมูลโมเดลสามมิติ เข้าไปเก็บในหน่วยความจำ โดยระบบเกมใช้เกมเอนจิ้นท์ 'Diesel Engine' ซึ่งมีหน้าที่ในการควบคุมดูแลจัดการข้อมูลและการแสดงผลสามมิติ เมื่อจัดการข้อมูลเบื้องต้นเสร็จแล้วจึงทำการแสดงผลไปยังหน้าจอ

2. เมื่อเริ่มเกมแล้ว ส่วน Game UI ทำหน้าที่รอรับการกระทำต่างๆ จากผู้เล่นผ่านทางหน้าจอติดต่อผู้ใช้ (User Interface)

3. เมื่อผู้เล่นกระทำการใดๆ ต่อหน้าต่างของเกม Input จากผู้เล่นจะผ่านอุปกรณ์แล้วส่วนของ Window CE UI Lib จะทำการประมวลผลส่งผลที่ได้ไปให้ส่วน Game UI แล้วส่งค่าที่ใช้ภายในแอพพลิชั่นไปให้ Game Controller เพื่อทำการประมวลผลแล้วจัดการส่งค่าไปยังส่วนการทำงานภายในระบบที่เกี่ยวข้องกับเมนู หรือคำสั่งที่ผู้เล่นได้กระทำในลำดับต่อไป

4. หากการทำงานส่วนใดเกี่ยวข้องกับข้อมูลที่ใช้ภายในแอพพลิเคชั่น ส่วน Game Controller จะทำการส่งการทำงานไปให้ส่วน Game Data ทำการจัดการข้อมูลต่อ

5. ในกรณีที่ผู้เล่นเลือกเมนูการทำงานที่ต้องมีการติดต่อกับผู้เล่นคนอื่น หรือเครื่องเซิร์ฟเวอร์ระบบในส่วน Game Controller จะทำการส่งการทำงานไปให้ส่วน Winsock Lib ทำการติดต่อผ่านเครือข่ายไร้สาย

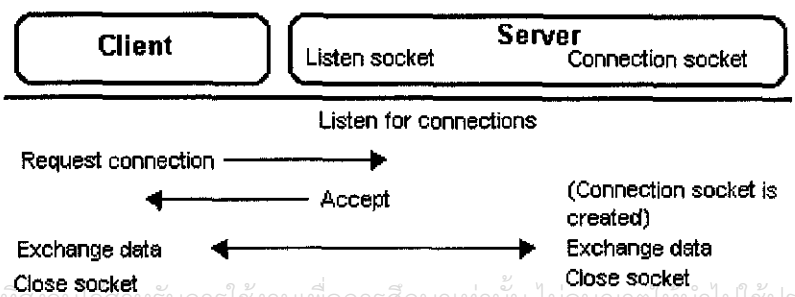
3.9 การออกแบบการเชื่อมต่อผ่านเครือข่ายโดยใช้ Winsock

3.9.1 สถาปัตยกรรมชั้นสูง(High Level Architecture)

รูปแบบในการออกแบบสถาปัตยกรรมชั้นสูงนั้นขึ้นอยู่กับว่าเกมที่เราออกแบบมานั่นต้องการติดต่อกันในรูปแบบใดซึ่งรูปแบบทั่วไปที่นิยมใช้กันมีดังนี้

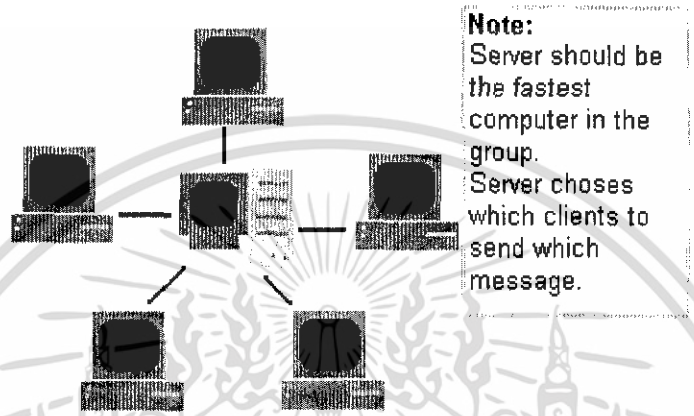
1. Client-server

ในรูปแบบนี้ผู้เล่นทุกคนหรือ client จะเชื่อมต่อไปยังเครื่องส่วนกลาง หรือ server ซึ่ง server จะทำหน้าที่ตัดสินใจข้อมูลที่สำคัญ ควบคุมสถานะ และทำการกระจายข้อมูลไปยัง client ซึ่งทำให้ระบบมีความคงทนเนื่องจากระบบถูกประมวลอยู่ในส่วนกลาง แต่ในบางระบบserver อาจไม่ทำการประมวลผลข้อมูลเลยก็ได้ขึ้นอยู่กับการออกแบบของผู้พัฒนา จากการที่เราใช้ server เป็นส่วนกลางทำให้ server ต้องทำงานร่วมกับ client ทุกเครื่องที่ต่อเข้ามา ทำให้มีปัญหาที่ตามในด้านของระบบเครือข่ายเช่น Bottleneck แต่ข้อดีก็คือ server จะทำหน้าที่ประมวลผลแทน client ซึ่ง client มีหน้าที่เพียงแสดงผลเท่านั้น



รูปที่ 3.16 แสดงการจำลองรูปแบบการติดต่อระหว่าง Client กับ Server

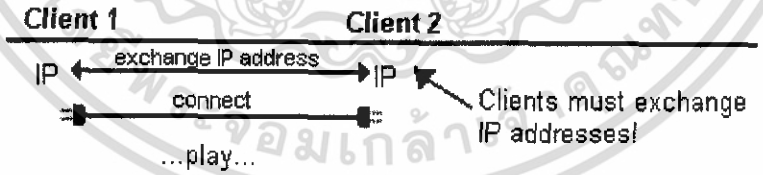
จากรูปจะเริ่มต้นโดยที่ client สร้าง socket ขึ้นมา 1 socket เพื่อใช้ในการเชื่อมต่อ ส่วน Server ใช้ 2 socket socket ทำหน้าที่ในการรับฟังเพื่อรอรับการเชื่อมต่อ เมื่อได้รับแล้วจึงทำการสร้าง socket เพิ่ม คือ connection socket เพื่อป้องกันให้สามารถทำการเชื่อมต่อกับเครื่องอื่นๆได้อีก



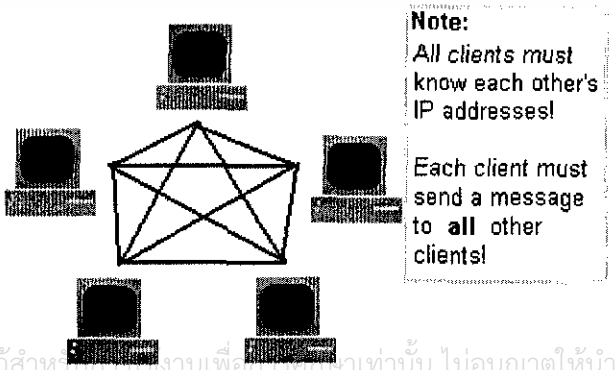
รูปที่ 3.17 แสดงการจำลองการเชื่อมต่อระหว่าง Client กับ Server

2. Client-Client

เป็นรูปแบบที่ทุกผู้เล่นต้องการประมวลผลและส่งข้อมูลในการตัดสินใจไปยังทุกเครื่องที่เล่นอยู่ ผลให้มีการสิ้นเปลื้องทรัพยากรทั้งในการประมวลผลและ Bandwidth ที่ใช้ ซึ่งเมื่อต้องการการเชื่อมต่อจำเป็นที่ต้องรู้ IP address ของอีกฝั่ง ซึ่งไม่สะดวกในการใช้งาน



รูปที่ 3.18 แสดงการจำลองรูปแบบการติดต่อระหว่าง Client กับ Client



รูปที่ 3.19 แสดงการจำลองการเชื่อมต่อระหว่าง Client กับ Client

3. Hybrid

เป็นวิธีที่ช่วยกันทำงานทั้งในส่วนของ Client และ server โดยให้ client สามารถทำการตัดสินใจในข้อมูลบางอย่างได้เช่นการเคลื่อนที่ แล้วจึงส่งข้อมูลอัพเดตไปยัง server ซึ่งเกมส่วนใหญ่จะใช้รูปแบบนี้ แต่ข้อเสียก็คือผู้เล่นสามารถโกงได้

3.9.2 สถาปัตยกรรมขั้นพื้นฐาน (Low Level Architecture)

1. TCP/IP

ในระดับนี้ส่วนใหญ่เกมในปัจจุบันมักมีตัวเลือก 2 ทางคือ TCP หรือUDP TCP หรือ Transmission control protocol เป็น Connection orient ซึ่งมีความน่าเชื่อถือในการส่งข้อมูล ส่วน UDP เป็นโปรโตคอลแบบ Connectionless กล่าวคือสามารถส่งได้รวดเร็ว แต่ข้อมูลขาดความน่าเชื่อถือเนื่องจากอาจเกิดการสูญหาย หรือชนกันของข้อมูลสูงกว่าโปรโตคอล TCP

2. โปรโตคอลในการติดต่อและรูปแบบข้อมูลที่ใช้

โดยรวมแล้วระบบของเราออกแบบให้ใช้รูปแบบ TCP ข้อมูลที่ส่งนั้นเป็นค่าข้อมูลประจำตัวมอนสเตอร์ของผู้เล่น ตัวเกมส์กับ เซิฟเวอร์ มีการตกลงให้รับ- ส่ง ข้อมูลในรูปแบบเดียวกันเป็นดังนี้

ข้อมูลที่เซิร์ฟเวอร์รับ : STR#QI#AGI#DEF#VIT#DEX#FIT#BRE#INT#name
ข้อมูลที่เซิร์ฟเวอร์ส่งกลับ : name#STR#QI#AGI#DEF#VIT#DEX#FIT#BRE#INT#Lv#!

รูปแบบข้อมูลดังกล่าวเป็นรูปแบบที่ผู้พัฒนาได้กำหนดขึ้นเอง โดยตัวข้อมูลในไฟล์เป็นเซ็ตของตัวอักษร (String) สำหรับข้อมูลภายในนั้นมีความหมาย คือ (ตัวอักษร) แทนค่าพารามิเตอร์นั้นๆ ของตัวมอนสเตอร์, เครื่องหมาย '#' หมายถึงหมดขอบเขตของค่าพารามิเตอร์นั้นๆ แล้ว และเครื่องหมาย '!' หมายถึง จบกระแสข้อมูลที่ผู้ส่งต้องการส่งแล้ว

ตัวอย่างโปรแกรมการทำงานอาจเป็นดังนี้

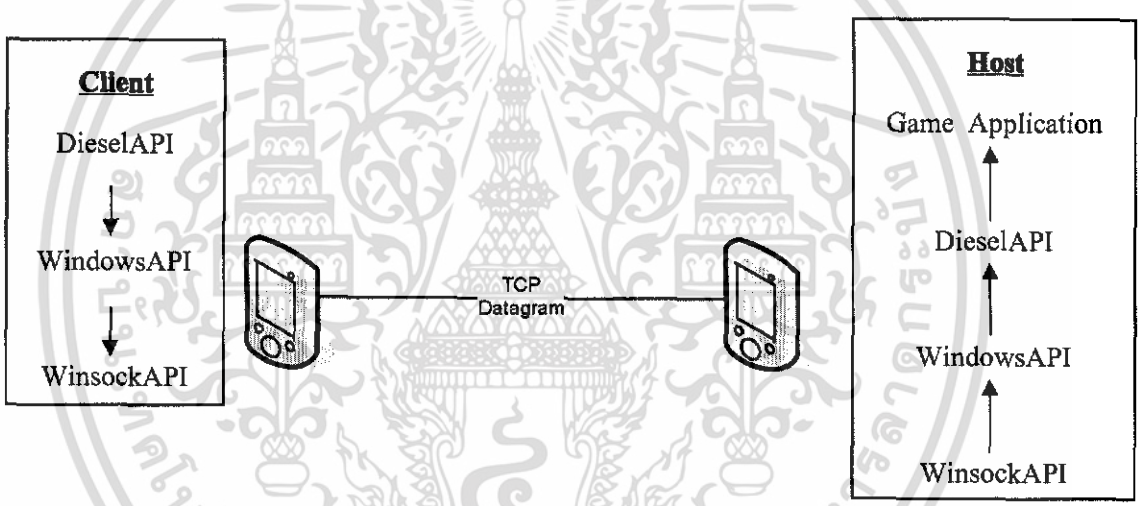
- การเชื่อมต่อระหว่างเครื่องผู้เล่น กับผู้เล่นคนอื่น
- เก็บข้อมูลตัวละครมอนสเตอร์ของผู้เล่นเอง
- รับข้อมูลมอนสเตอร์ของผู้เล่นคนอื่น
- ทำการสร้างโมเดลของมอนสเตอร์ของผู้เล่นคนอื่นตามข้อมูลที่ได้รับ
- ทำการ 'ต่อสู้ระหว่างมอนสเตอร์' (ไม่ใช้การเชื่อมต่อกับเครือข่าย)

- วนทำงานในระบบการต่อสู้จนกว่าจะรู้ผลแพ้ - ชนะ
- อัพเดตสถานะของผู้เล่นตามผลการต่อสู้
- ส่งข้อมูลมอนสเตอร์คืนแก่ผู้เล่นที่เป็นเจ้าของ

3.9.3 สถาปัตยกรรมการเชื่อมต่อโดยรวม

1. การเชื่อมต่อระหว่าง client – client

การเชื่อมต่อระหว่าง 2 เครื่องนั้นเครื่องหนึ่งทำหน้าที่เป็น Host ทำหน้าที่ในการเปิดการติดต่อ และอีกเครื่องหนึ่งเป็น client เข้ามาทำการเชื่อมต่อ เมื่อมีการเริ่มต้นการติดต่อโดย host จะทำการเปิด socket โดยเริ่มจากรับคำสั่งจากหน้าจอผ่าน DieselAPI ส่งผ่าน WinAPI ไปยัง WinsockAPI เพื่อเปิดการติดต่อ socket สำหรับเป็นเส้นทางในการส่ง TCP Datagram ไปยังปลายทาง โดยที่ในส่วนของ client มีการทำงานเช่นเดียวกัน



รูปที่ 3.20 แสดงการจำลองการเชื่อมต่อระหว่าง Client กับ Client

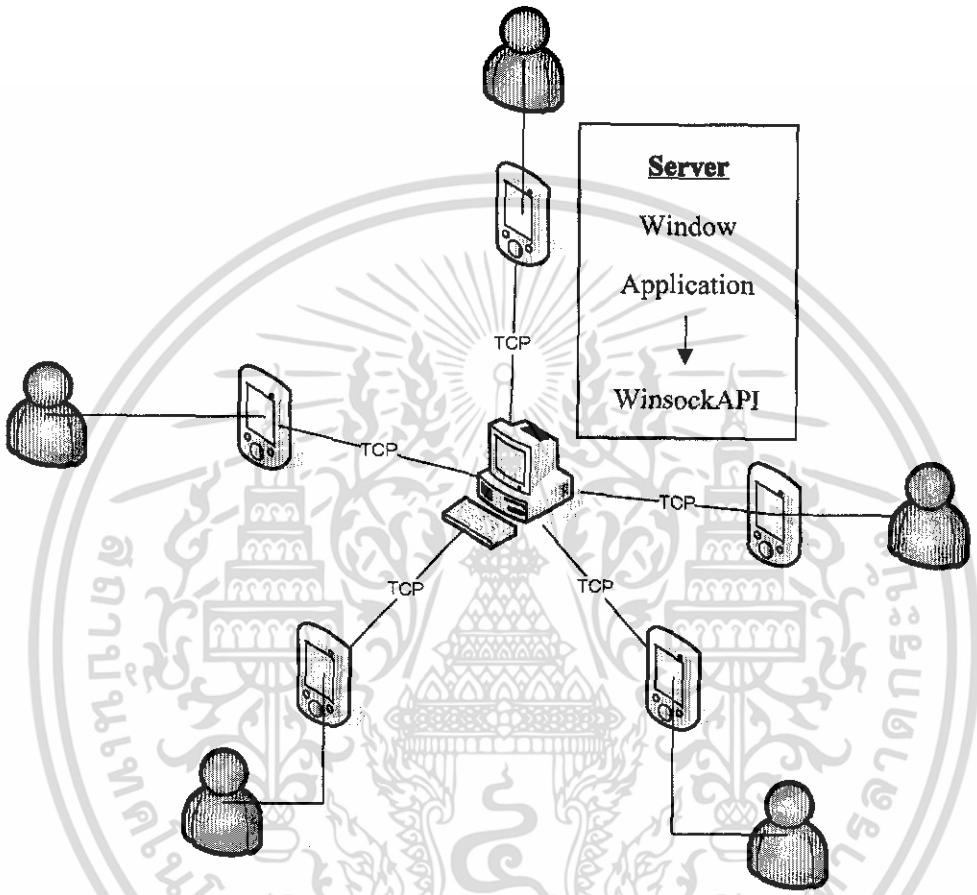
2. การเชื่อมต่อระหว่าง client – server

การเชื่อมต่อระหว่าง Client – server เครื่อง server ทำหน้าที่ในการเปิดการติดต่อให้เครื่อง client เข้ามาทำการเชื่อมต่อ เมื่อมีการเริ่มต้นการติดต่อโดย server จะทำการเปิด socket และรอรับการติดต่อจาก client โดยการติดต่อนั้นต้องสถาปนาการเชื่อมต่อให้เสร็จสิ้นเสียก่อน เพราะเป็นการติดต่อแบบ Connection Oriented

เครื่อง Client รับคำสั่งของผู้เล่นจากหน้าจอเกมผ่าน DieselAPI ส่งผ่าน WinAPI ไปยัง WinsockAPI เพื่อเปิดการติดต่อ socket สำหรับเป็นเส้นทางในการส่ง TCP Datagram ซึ่งบรรจุข้อมูลตัวละครของผู้เล่นส่งไปยังเครื่อง server ปลายทาง

เครื่อง Server ทำการรับข้อมูลจากเครื่อง client แต่ละเครื่องผ่านทาง socket ซึ่งจัดการโดย Winsock API จากนั้นส่งข้อมูลที่รับมาไปยัง Window Application เพื่อทำการประมวลผล ซึ่งการ

อ้างอิงและบันทึกข้อมูลทั้งหมดจะกระทำกับฐานข้อมูลของระบบที่ Database Server และในทุกครั้งที่มีการรับข้อมูลเข้ามาใหม่จะทำการอัพเดตข้อมูลนั้นๆ ไปยังเครื่อง client ที่ทำการเชื่อมต่ออยู่ทั้งหมด



รูปที่ 3.21 แสดงการจำลองการเชื่อมต่อระหว่าง Client กับ Server

บทที่ 4
ชิ้นงานของโครงงาน

รายละเอียดการทำงานของฟังก์ชันต่างๆ อธิบายไว้ในหัวข้อย่อยที่ '3.2.4 กฎ กติกาภายในเกม' ภายในบทที่ 3 "การออกแบบรายละเอียดของเกม"แล้ว โดยหัวข้อนี้ทำการแสดงชิ้นงานที่ได้จากการนำสิ่งที่ได้ออกแบบไว้มาพัฒนา

เมื่อพัฒนาชิ้นงานแล้วมีการแสดงผลดังต่อไปนี้

4.1 การแสดงผล

การแสดงผลภายในเกมเป็นมุมมองจากด้านหน้า ดังแสดงในรูปต่อไปนี้

รูปที่ 4.1 แสดงหน้าจอปกติภายในเกม

ผู้เล่นสามารถเลือกเมนูต่างๆ และชมผลลัพธ์การกระทำได้ในมุมมองด้านหน้า

4.2 ฟังก์ชันการทำงานภายในเกมและความสามารถของระบบ

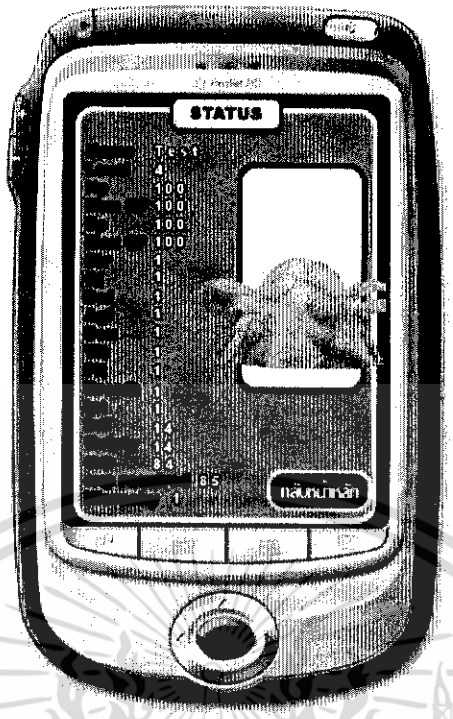
4.2.1 การตอบสนองต่อเมนูการทำงานภายในเกม

รูปที่ 4.2 แสดงเมนูฟังก์ชันการทำงานภายในเกม

ฟังก์ชันการทำงานของระบบหลัก มีดังนี้

1. เมนูดูสถานะและค่าพารามิเตอร์ของมอนสเตอร์

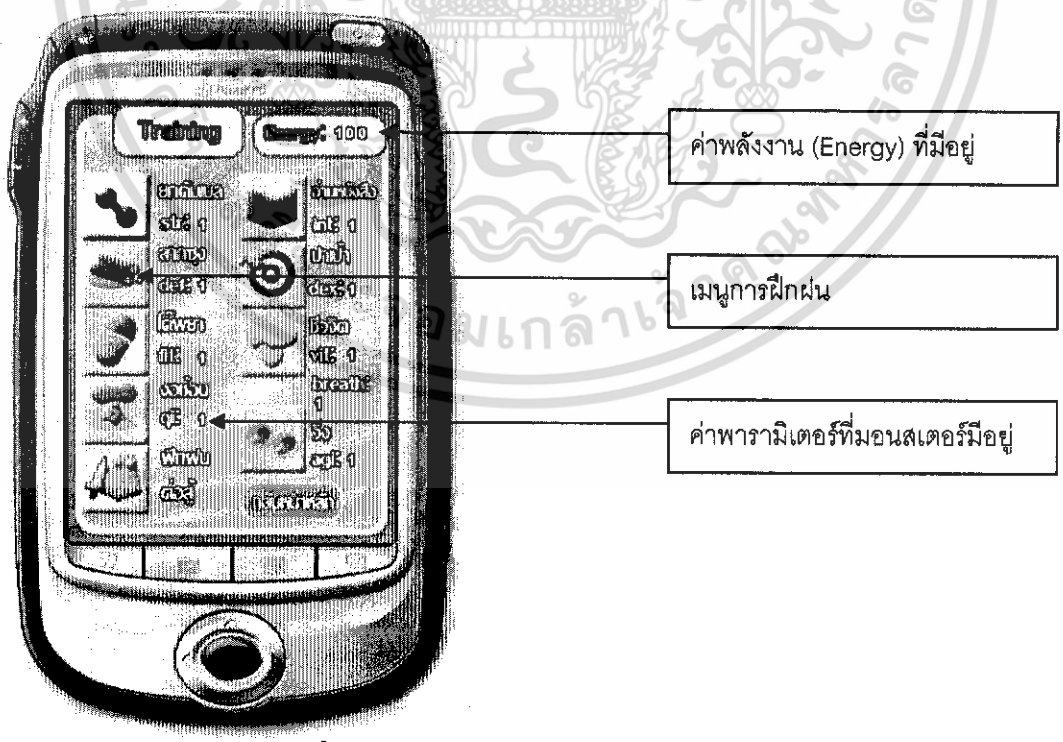
เมื่อเลือกเมนูนี้จะแสดงหน้าต่างบอกค่าพารามิเตอร์ปัจจุบันของมอนสเตอร์ ดังรูปต่อไปนี้



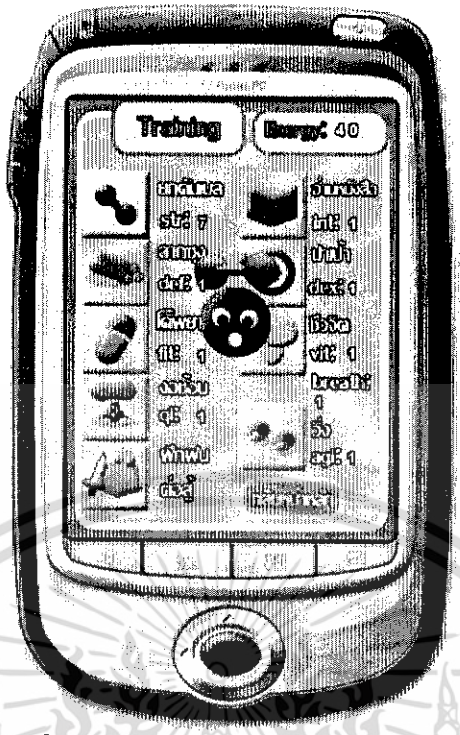
รูปที่ 4.3 แสดงหน้าต่างบอกค่าพารามิเตอร์ของตัวละคร

2. เมนูฝึกฝน

เมื่อเลือกเมนูนี้จะแสดงหน้าต่างให้เลือกประเภทของการฝึกฝน ดังนี้

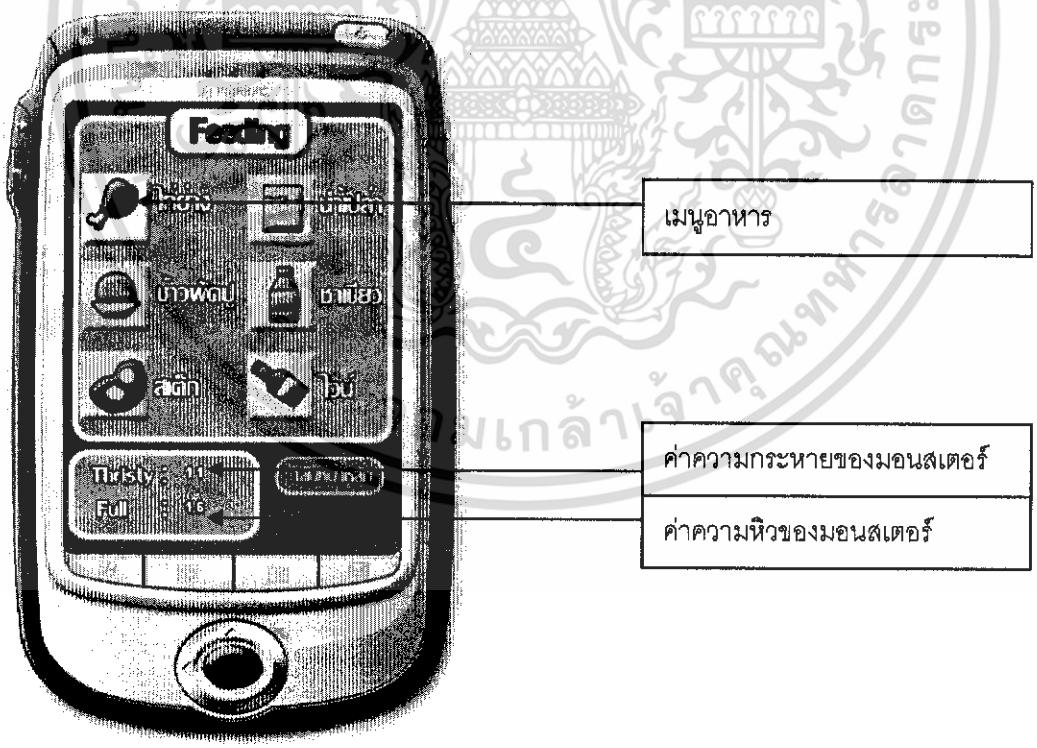


รูปที่ 4.4 แสดงหน้าต่างให้เลือกเมนูการฝึกฝน

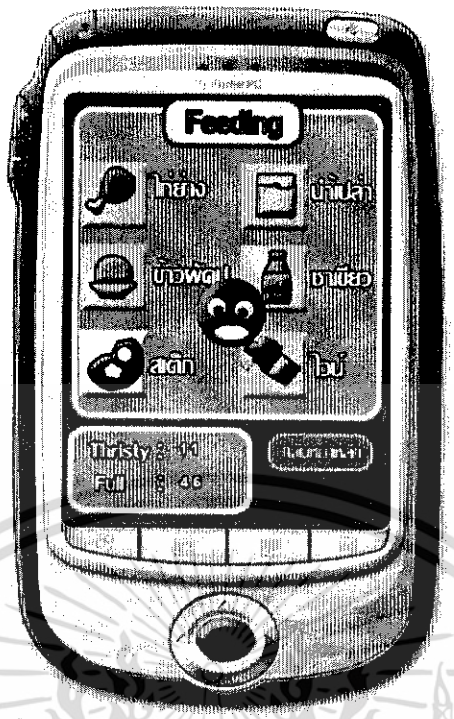


รูปที่ 4.5 แสดงการฝึกฝนของมอนสเตอร์

3. เมนูให้อาหารมอนสเตอร์



รูปที่ 4.6 แสดงหน้าต่างให้เลือกเมนูให้อาหาร



รูปที่ 4.7 แสดงการกินอาหารของมอนสเตอร์

4. เมนูพักผ่อน

เมื่อเลือกเมนูพักผ่อนจากหน้าต่างหลักของเกมแล้ว มอนสเตอร์จะทำการพักผ่อน

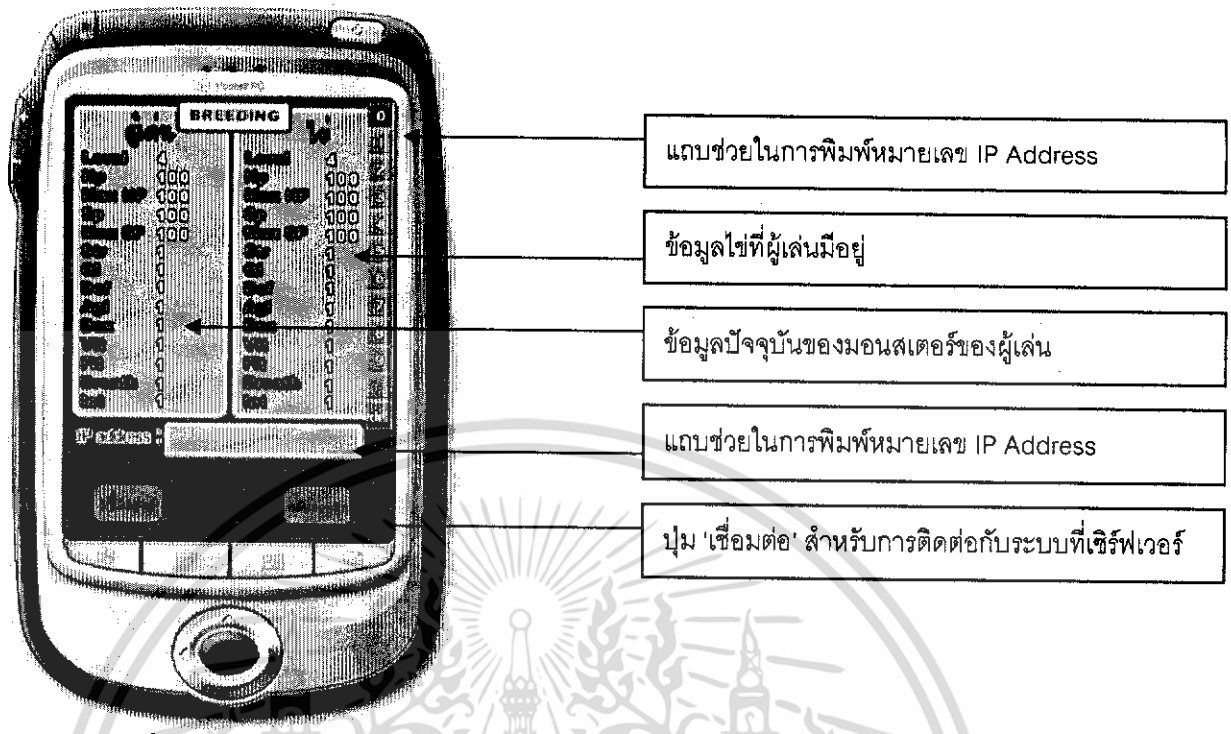
รูปที่ 4.8 แสดงมอนสเตอร์ขณะพักผ่อน

5. เมนูต่อสู้

เมื่อเลือกเมนูต่อสู้นี้ จะเข้าสู่การประลองฝีมือของมอนสเตอร์ของผู้เล่น 2 คน

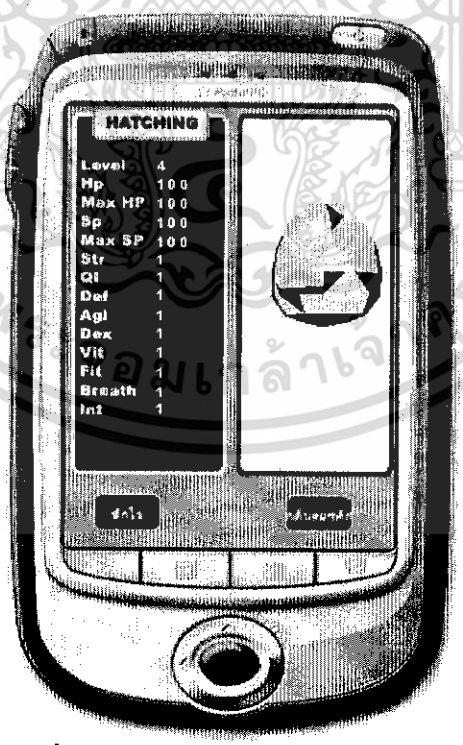
รูปที่ 4.9 แสดงหน้าต่างการเชื่อมต่อระหว่างผู้เล่นเพื่อทำการต่อสู้กัน

6. เมนูผสมพันธุ์มอนสเตอร์

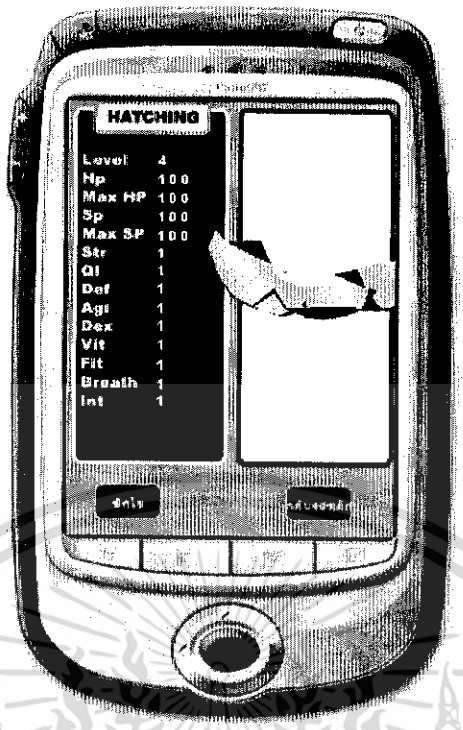


รูปที่ 4.10 แสดงหน้าต่างการเชื่อมต่อระหว่างผู้เล่นเพื่อผสมพันธุ์มอนสเตอร์

7. เมนูห้องเก็บไข่



รูปที่ 4.11 แสดงห้องเก็บไข่มอนสเตอร์



รูปที่ 4.12 แสดงการฟักไข่

8. เมนู Option

รูปที่ 4.13 แสดงหน้าต่าง Option

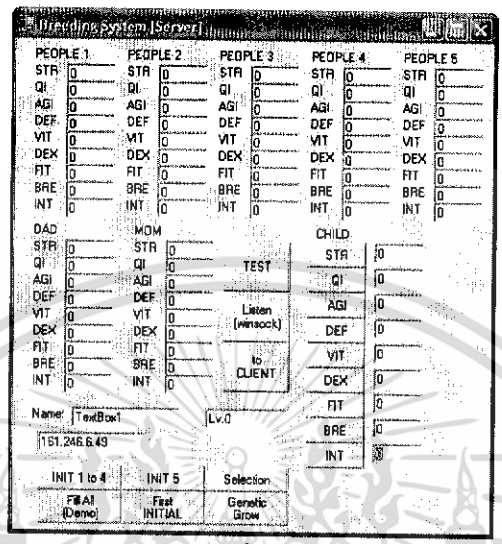
4.2.2 การเจริญเติบโตของมอนสเตอร์

รูปที่ 4.14 แสดงมอนสเตอร์ร่างแรก

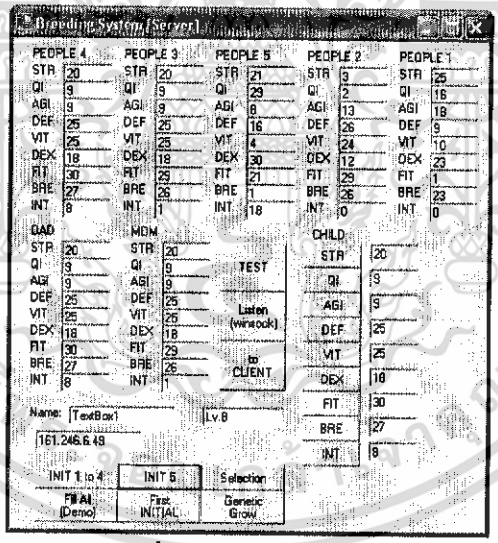
รูปที่ 4.15 แสดงมอนสเตอร์ที่ทำการฝึกฝนจนเปลี่ยนร่าง

4.2.3 การผสมพันธุ์มอนสเตอร์

หัวข้อนี้แสดงเฉพาะแอพพลิเคชั่นของด้านเซิร์ฟเวอร์ ในส่วนของไคลเอนท์ได้แสดงไว้แล้วในรูปที่ 4.10

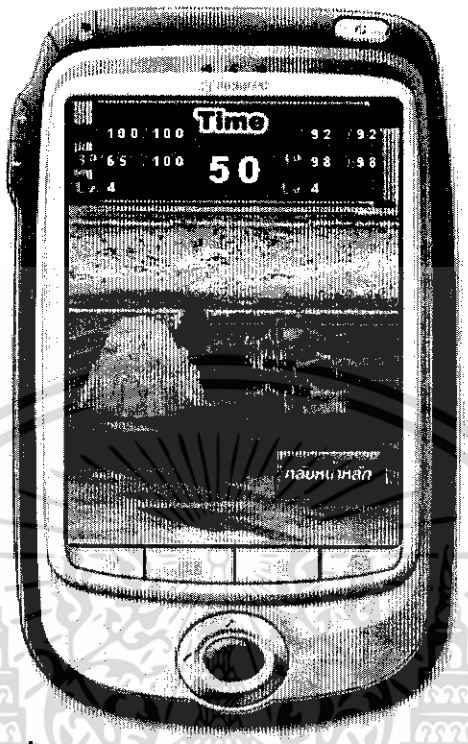


รูปที่ 4.16 แสดงแอพพลิเคชั่นที่เซิร์ฟเวอร์ขณะเริ่มทำงาน



รูปที่ 4.17 แสดงแอพพลิเคชั่นบนเซิร์ฟเวอร์หลังจากที่รับข้อมูลจากผู้เล่น

4.2.4 การต่อสู้ระหว่างมอนสเตอร์



รูปที่ 4.18 แสดงการต่อสู้ระหว่างมอนสเตอร์

บทที่ 5

การทดลองและผลการทดลอง

เนื้อหาในบทนี้กล่าวถึงการวางแผนการทดสอบระบบ และทดสอบระบบตามแผนที่วางไว้บนอุปกรณ์พ็อกเก็ตพีซีพร้อมทั้งแสดงผลการทดสอบระบบด้วยภาพที่ได้จากการจับภาพจากหน้าจอในระบบที่ทำงานบนอุปกรณ์จริง

5.1 ทดสอบระบบในส่วนของการเล่นคนเดียว (stand alone)

แผนการทดสอบระบบมีดังนี้

5.1.1 ทดสอบฟังก์ชันการกินอาหารของมอนสเตอร์

สิ่งที่คาดหวัง :

1) กินอาหารแล้วค่า Full เพิ่มขึ้น
2) ดื่มเครื่องดื่มแล้วค่า Thirsty เพิ่มขึ้น

การทดสอบ :

1) ดื่มเครื่องดื่มชาเขียว
2) กินอาหารสเต็ก 2 ครั้ง

รูปที่ 5.1 แสดงค่า Thirsty, Full ก่อนกินอาหาร(ซ้าย) และหลังกินอาหาร(ขวา)

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้

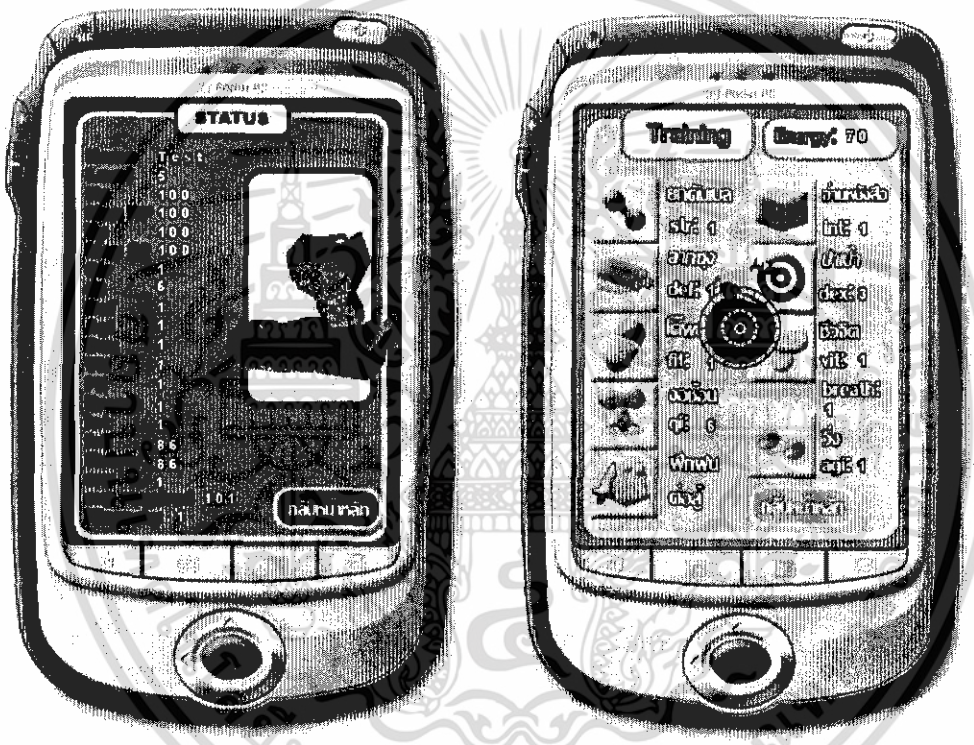
5.1.2 ทดสอบการฝึกฝนมอนสเตอร์

สิ่งที่คาดหวัง :

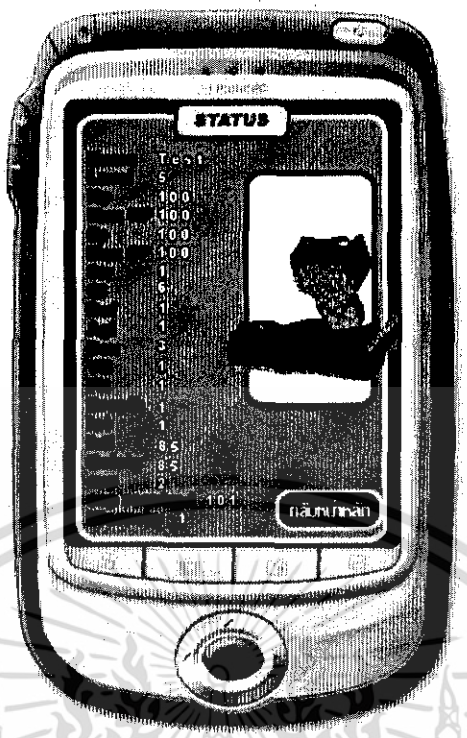
หลังการฝึกฝนค่าพารามิเตอร์ Str เพิ่มขึ้น

การทดสอบ :

ผู้เล่นเลือกเมนู 'ปาเป้า' ภายในเมนู 'ฝึกฝน'



รูปที่ 5.2 แสดงค่า พารามิเตอร์ก่อนทำการฝึก(ซ้าย), การทำการฝึกฝนปาเป้า(ขวา)



รูปที่ 5.3 แสดงค่าพารามิเตอร์หลังทำการฝึก

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้ ค่า Dex เพิ่มขึ้น

5.1.3 ทดสอบการเจริญเติบโต เปลี่ยนร่างของมอนสเตอร์

สิ่งที่คาดหวัง :

หลังจากฝึกฝนมอนสเตอร์โดยเน้นค่าพารามิเตอร์ Qi จนมอนสเตอร์มีค่า Level เท่ากับ 5 มอนสเตอร์ต้องเปลี่ยนร่างเป็น 'มังกร'

การทดสอบ :

ฝึกฝนมอนสเตอร์จนค่าระดับ Level เท่ากับ 5 และเน้นฝึกค่า Qi เป็นหลัก

รูปที่ 5.4 แสดงมอนสเตอร์ก่อนเปลี่ยนร่าง(ซ้าย) มอนสเตอร์ทำการฝึกฝนเพิ่มค่า Qi(ขวา)

รูปที่ 5.5 แสดงมอนสเตอร์หลังทำการฝึกฝนเสร็จแล้ว

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้ มอนสเตอร์เปลี่ยนร่างเป็น 'มังกร'

5.1.4 ทดสอบการต่อสู้กับศัตรูที่เป็นคอมพิวเตอร์

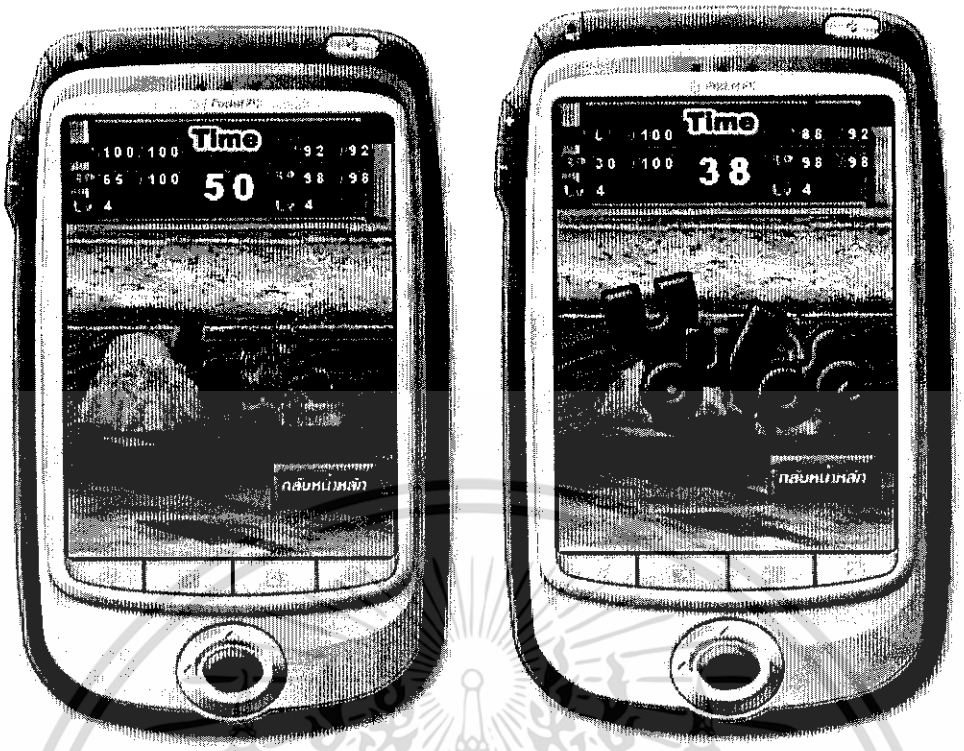
สิ่งที่คาดหวัง :

1) มอนสเตอร์ศัตรูมีระดับเดียวกับมอนสเตอร์ของผู้เล่น
2) ค่าพารามิเตอร์ของมอนสเตอร์ของผู้เล่นเปลี่ยนแปลงตามผลการต่อสู้

การทดสอบ :

1) ผู้เล่นเลือกเมนู 'ฝึกฝนต่อสู้' ที่อยู่ภายในเมนู 'ฝึกฝน'
2) มอนสเตอร์ของผู้เล่นทำการต่อสู้กับมอนสเตอร์ที่ระบบสร้างขึ้น
3) ตรวจสอบระดับ Level ของมอนสเตอร์ศัตรู
4) ตรวจสอบค่าพารามิเตอร์ต่างๆ ของมอนสเตอร์หลังการต่อสู้

รูปที่ 5.6 แสดงมอนสเตอร์ของผู้เล่นก่อนเข้าสู่การต่อสู้



รูปที่ 5.7 แสดงการต่อสู้ (ซ้าย) และผลการต่อสู้ (ขวา)

รูปที่ 5.8 แสดงมอนสเตอร์ของผู้เล่นหลังจบการต่อสู้

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้

5.1.5 ทดสอบฟังก์ชันพักผ่อน

สิ่งที่คาดหวัง :

ค่า Hp, Sp และค่า Energy ของมอนสเตอร์เพิ่มขึ้นในระหว่างที่อยู่ในช่วงพักผ่อน

การทดสอบ :

1) ผู้เล่นเลือกเมนู 'ฝึกฝนต่อสู้' ภายในเมนู 'ฝึกฝน' จากหน้าจอหลัก
2) ตรวจสอบค่าพลังชีวิต (Hp), ค่าพลังวิญญาณ (Sp), ค่าพลังงาน (Energy) หลังจากทำการต่อสู้
3) ผู้เล่นเลือกเมนู 'พักผ่อน'
4) ตรวจสอบค่าพลังชีวิต (Hp), ค่าพลังวิญญาณ (Sp), ค่าพลังงาน (Energy)

รูปที่ 5.9 แสดงมอนสเตอร์ขณะทำการพักผ่อน

รูปที่ 5.10 แสดงมอนสเตอร์ก่อนพักผ่อน (ซ้าย) และหลังพักผ่อน (ขวา)

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้ การพักผ่อนช่วยเพิ่มค่าพลังแก่มอนสเตอร์

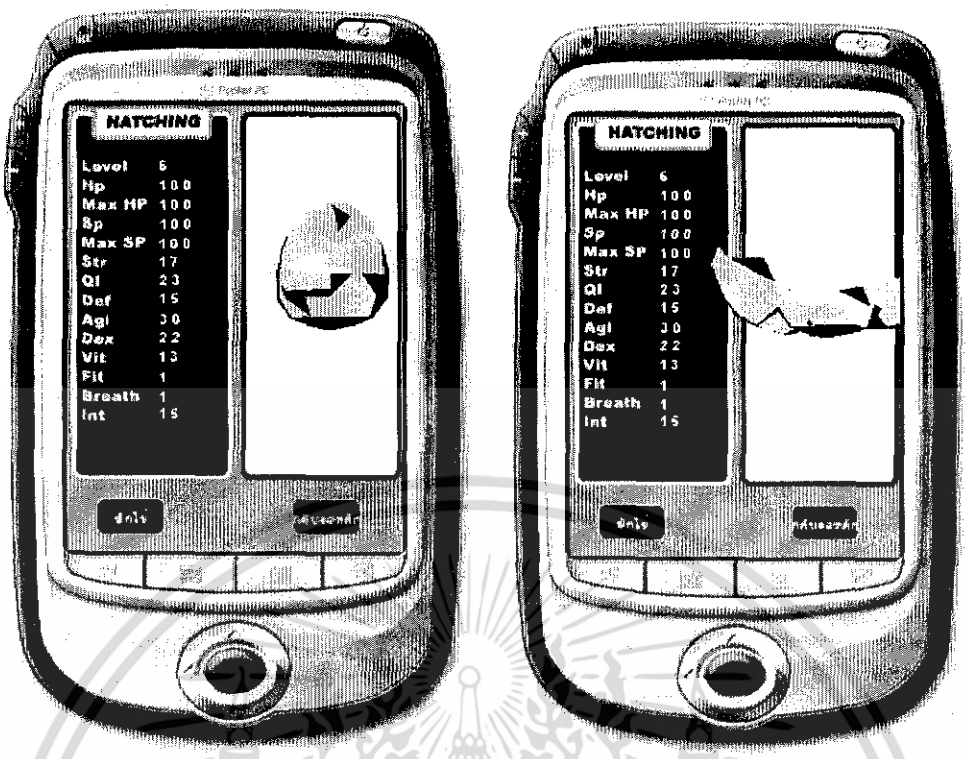
5.1.6 ทดสอบการฟักไข่มอนสเตอร์

สิ่งที่คาดหวัง :

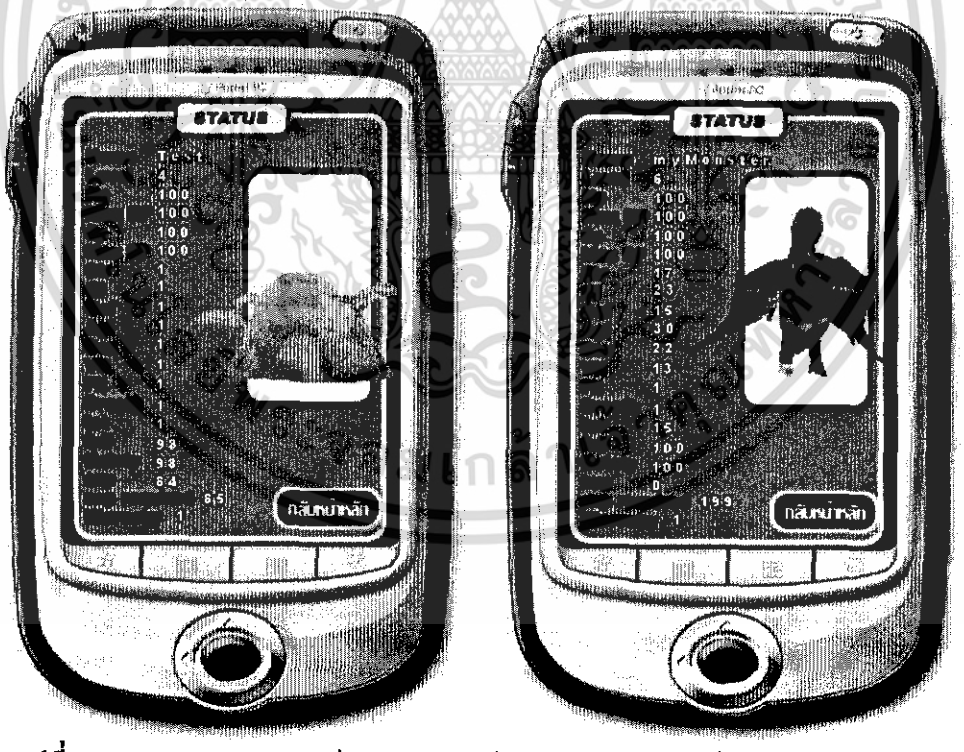
เมื่อทำการฟักไข่มอนสเตอร์แล้ว มอนสเตอร์ที่อยู่ในไข่จะถูกนำมาแทนที่มอนสเตอร์ที่เลี้ยงอยู่ในปัจจุบัน

การทดสอบ :

1) ผู้เล่นเลือกเมนู 'ห้องไข่' แล้วเข้าไปทำการเลือกคำสั่ง 'ฟักไข่' แล้วกลับสู่หน้าจอหลัก
2) ตรวจสอบมอนสเตอร์ปัจจุบันว่าตรงกับมอนสเตอร์ที่ออกมาจากไข่หรือไม่



รูปที่ 5.11 แสดงห้องฟักไข่พร้อมข้อมูลมอนสเตอร์ที่อยู่ภายในไข่ (ซ้าย) และหลังจากผู้เล่นเลือก 'ฟักไข่'



รูปที่ 5.12 แสดงมอนสเตอร์ก่อนฟักไข่ (ซ้าย) และมอนสเตอร์หลังจากฟักไข่ (ขวา)

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้ มอนสเตอร์ภายในไข่มอนสเตอร์มาแทนที่มอนสเตอร์ของผู้เล่นตัวปัจจุบันหลังจากที่ผู้เล่นทำการฟักไข่แล้ว

5.1.7 ทดสอบฟังก์ชัน Save และ Load

สิ่งที่คาดหวัง :

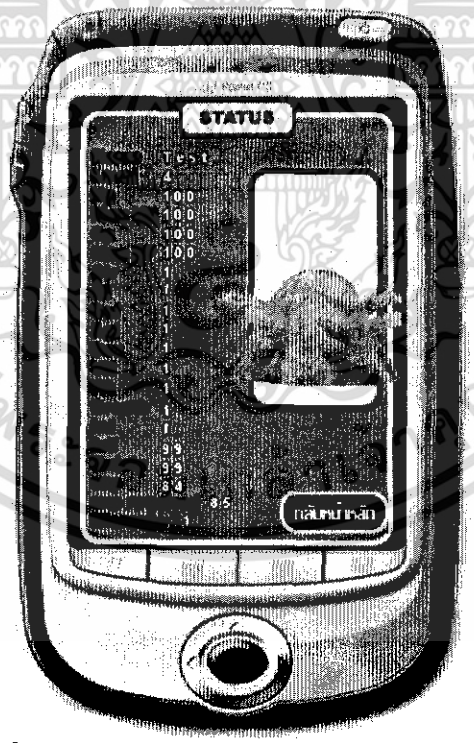
เมื่อผู้เล่นทำการ Save เกม ข้อมูลของตัวละครต้องถูกบันทึก เมื่อผู้เล่นทำการ Load เกม ข้อมูลมอนสเตอร์ที่ถูกบันทึกไว้ต้องถูกนำมาแทนที่ข้อมูลมอนสเตอร์ปัจจุบัน

การทดสอบ :

1) ผู้เล่นทำการเลือกคำสั่ง 'Save' ภายในเมนู 'Option'
2) ผู้เล่นทำการฝึกฝนเปลี่ยนแปลงค่าพารามิเตอร์ของมอนสเตอร์
3) ผู้เล่นทำการเลือกคำสั่ง 'Load' ภายในเมนู 'Option'
4) ตรวจสอบข้อมูลมอนสเตอร์ว่ากลับมาเท่ากับข้อมูลตอนที่ทำการ Save หรือไม่

รูปที่ 5.13 *แสดงมอนสเตอร์ก่อนทำการ Save (ซ้าย) และผู้เล่นเลือกคำสั่ง Save เรียบร้อย (ขวา)*

รูปที่ 5.14 แสดงมอนสเตอร์ที่ทำการเปลี่ยนแปลงค่าหลังจากที่ Save (ซ้าย) และผู้เล่นเลือกคำสั่ง Load เรียบร้อย (ขวา)



รูปที่ 5.16แสดงมอนสเตอร์หลังจากทำการโหลด

ผลที่ได้ :

เป็นไปตามที่คาดหวังไว้ มอนสเตอร์ที่ได้จากการโหลดมีค่าพารามิเตอร์ทุกอย่างเท่ากับมอนสเตอร์ที่ ผู้เล่นทำการ Save

5.2 ทดสอบระบบในส่วนของการเล่นกับผู้เล่นคนอื่น

แผนการทดสอบระบบมีดังนี้

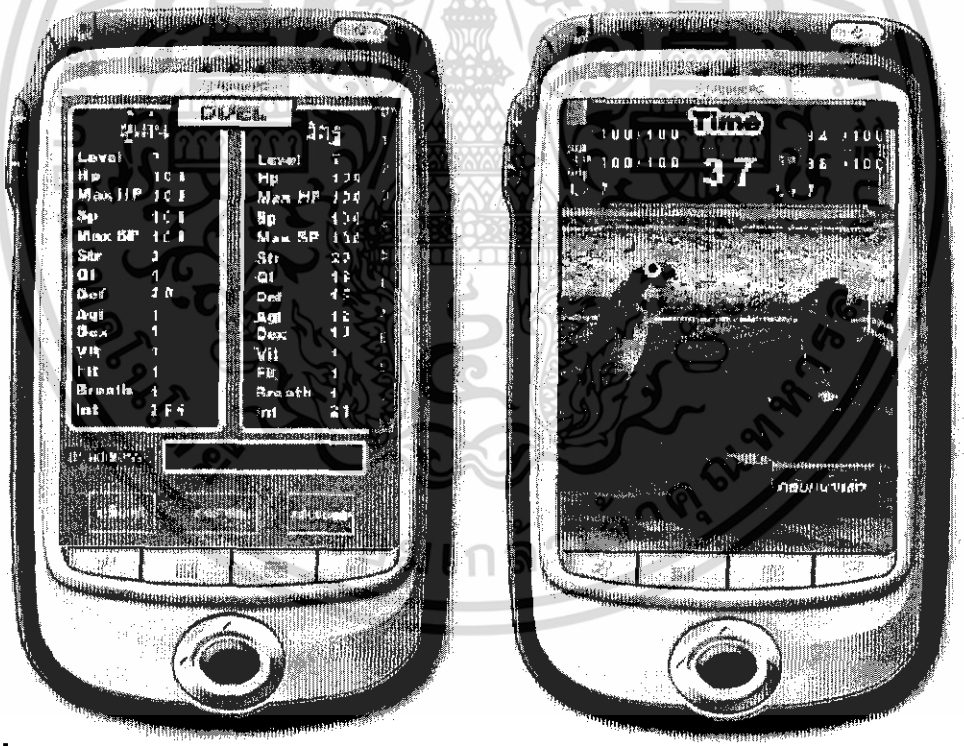
ทดสอบฟังก์ชันการต่อสู้ระหว่างผู้เล่น

สิ่งที่คาดหวัง :

1) สามารถเชื่อมต่อระหว่างแอพพลิเคชั่นของผู้เล่นทั้งสองได้
2) เมื่อรู้ผลการต่อสู้ ข้อมูลมอนสเตอร์ของผู้เล่นที่เป็น Client ต้องถูกส่งคืนตามข้อมูลหลังการต่อสู้จริง

การทดสอบ :

1) ผู้เล่นทั้งสองคนเปิดระบบเชื่อมต่อไร้สาย
2) ผู้เล่นทั้งสองคนเลือกเมนู 'ต่อสู้' จากหน้าจอหลักของเกม
3) ผู้เล่นคนหนึ่งเลือกเป็น Server, ผู้เล่นอีกคนเลือกเป็น Client (พิมพ์หมายเลข IP Address ของเครื่องที่เป็นเซิร์ฟเวอร์)
4) ตรวจสอบข้อมูลมอนสเตอร์ของผู้เล่นทั้งสองหลังจบการต่อสู้



รูปที่ 5.17 แสดงหน้าจอติดต่อระหว่างผู้เล่น (ซ้าย) และการต่อสู้ระหว่างตัวละครของผู้เล่นทั้งสอง (ขวา)

ผลที่ได้ :

เป็นไปตามที่คาดไว้ ผู้เล่นทั้งสองคนสามารถเชื่อมต่อกันได้ และมอนสเตอร์ของผู้เล่นทั้งสองมีค่าพารามิเตอร์ทั้งหมดตรงตามข้อมูลหลังจากทำการต่อสู้เสร็จสิ้นแล้ว

5.3 ทดสอบระบบในส่วนของการติดต่อกับเครื่องเซิร์ฟเวอร์

แผนการทดสอบระบบมีดังนี้

ทดสอบการเชื่อมต่อระหว่างไคลเอนท์กับเซิร์ฟเวอร์จากฟังก์ชันผสมพันธุ์มอนสเตอร์

สิ่งที่คาดหวัง :

1) สามารถเชื่อมต่อระหว่างแอพพลิเคชั่นของผู้เล่น 5 คนกับเครื่องเซิร์ฟเวอร์ได้
2) ระบบสามารถส่งข้อมูล 'ไข่มอนสเตอร์' ที่ได้จากการผสมพันธุ์มอนสเตอร์กลับไปยังอุปกรณ์พีอกเก็ตพีซีของผู้เล่นที่เชื่อมต่อกับระบบในเซิร์ฟเวอร์ได้เป็นคนแรก

การทดสอบ :

1) ผู้เล่น 5 คนทำการเชื่อมระบบเข้าสู่เครื่องเซิร์ฟเวอร์ จากเมนูผสมพันธุ์ในหน้าจอหลักในเกม
2) ระบบทำการผสมพันธุ์มอนสเตอร์
3) ตรวจสอบว่าระบบคืนค่าของไข่มอนสเตอร์ที่ได้จากการผสมพันธุ์ให้แก่ผู้เล่นที่ทำการเชื่อมต่อกับเครื่องเซิร์ฟเวอร์เป็นคนแรกหรือไม่

ผลที่ได้ :

เป็นไปตามที่คาดไว้ เครื่องพีอกเก็ตพีซีที่เป็นเครื่องไคลเอนท์สามารถเชื่อมต่อกับเครื่องเซิร์ฟเวอร์ได้ และเมื่อเครื่องเซิร์ฟเวอร์ทำการผสมพันธุ์มอนสเตอร์เสร็จก็ได้ทำการส่งคืนค่าของไข่มอนสเตอร์ กลับไปให้ผู้เล่นที่ทำการเชื่อมต่อกับเครื่องเซิร์ฟเวอร์เป็นคนแรก

บทที่ 6
บทวิจารณ์และสรุป

6.1 บทสรุป

ปัจจุบันตลาดเกมในบ้านเรายังเป็นการนำเข้าจากต่างประเทศเป็นส่วนใหญ่ โดยเฉพาะกลุ่มเกมสามมิติทั้งบนเครื่องคอมพิวเตอร์ หรืออุปกรณ์พกพาต่างๆ โดยเฉพาะอุปกรณ์พ็อกเก็ตพีซี อาจเรียกได้ว่าไม่มีคนไทยพัฒนาเกมสามมิติบนอุปกรณ์ชนิดนี้เลยก็ว่าได้ ทำให้พวกเราต้องเสียเงินซื้อซอฟต์แวร์จากต่างประเทศกันเป็นจำนวนมาก

โครงงานนี้เป็นงานวิจัยและพัฒนาเกมสามมิติบนอุปกรณ์พ็อกเก็ตพีซี เป็นเกมฝึกฝนเลี้ยงดูมอนสเตอร์ที่มีการติดต่อระหว่างผู้เล่นกับผู้เล่นคนอื่น หรือระหว่างผู้เล่นกับเครื่องเซิร์ฟเวอร์ได้ โดยมีการนำความรู้เชิงวิศวกรรมคอมพิวเตอร์มาประยุกต์ใช้ในโครงงาน ความรู้ที่นำมาประยุกต์ใช้ได้แก่ ความรู้ด้านคอมพิวเตอร์กราฟิก (Computer Graphics), ปัญญาประดิษฐ์ (เรื่อง Genetic Algorithm), คอมพิวเตอร์เครือข่าย นอกจากนี้ยังนำความรู้ที่มีมาประยุกต์สร้างอัลกอริทึมสำหรับให้มอนสเตอร์สามารถคิดและตัดสินใจในฉากต่อสู้โดยสามารถต่อสู้ได้ด้วยตัวเอง ไม่ต้องให้ผู้เล่นบังคับ

โครงงานที่พัฒนาโดยนักศึกษาไทยชิ้นนี้หวังเป็นอย่างยิ่งว่าจะเป็นการจุดประกายให้เห็นว่าคนไทยสามารถพัฒนาซอฟต์แวร์ได้ไม่แพ้ชาวต่างชาติ ที่ยังขาดอยู่ส่วนใหญ่ก็เป็นเรื่องเทคนิคที่นำมาประยุกต์ใช้ในชิ้นงานที่ต้องอาศัยประสบการณ์และความตั้งใจที่จะเรียนรู้สิ่งใหม่ๆ เทคโนโลยีใหม่ๆ

6.2 วิจารณ์สิ่งที่ได้จากโครงงาน

เนื่องจากเครื่องพ็อกเก็ตพีซีมีทรัพยากรจำกัด ถือว่าน้อยกว่าเครื่องคอมพิวเตอร์ปกติมาก ทั้งในเรื่องของหน่วยความจำ, การแสดงผล หรือการติดต่อกับผู้ใช้งาน ในการดำเนินการพัฒนาจึงต้องออกแบบระบบให้บริหารทรัพยากรที่มีอยู่อย่างจำกัดให้คุ้มค่าที่สุด ออกแบบอัลกอริทึมให้ประหยัดเวลาในการประมวลผล นอกจากนี้ทางผู้พัฒนาไม่มีใครมีประสบการณ์การพัฒนากราฟิกสามมิติ หรือการพัฒนาโปรแกรมประยุกต์บนอุปกรณ์พ็อกเก็ตพีซีมาก่อน จึงต้องอาศัยเวลามาเริ่มศึกษาใหม่ตั้งแต่ต้น ดังนั้นชิ้นงานที่ได้จากโครงงานจึงเป็นเวอร์ชันทดลองเล่น (Demo) ที่มีระบบการทำงานหลักครบถ้วนแต่ยังขาดรายละเอียดปลีกย่อย อาทิเช่น ตัวละครสามมิติอีกหลายสิบตัว, ระบบเสียง เป็นต้น เพราะการจะดำเนินการพัฒนาให้ได้เกมสามมิติสมบูรณ์แบบนั้นต้องอาศัยระยะเวลาและทรัพยากรเป็นจำนวนมาก

6.3 ปัญหาอุปสรรคและแนวทางในการแก้ไข

1. เครื่องมือที่ใช้ในการพัฒนาล้าสมัย (Embedded Visual C++ 3.0) และโปรแกรมจำลองเครื่องพ็อกเก็ตพีซี (Emulator) ทำงานได้ช้า และทำงานได้สมบูรณ์ตามโปรแกรมที่นำเข้าไปรัน (Run) เช่น มีปัญหาในการแสดงภาพโมเดลสามมิติเต็มตัว, มีปัญหาในเรื่องการเชื่อมต่อเครือข่าย การแก้ไขทำได้โดยเปลี่ยนเครื่องมือที่ใช้พัฒนาให้มีประสิทธิภาพสูงขึ้น เช่น ใช้ Microsoft .Net Framework หรือขอทุนซื้อเครื่องพ็อกเก็ตพีซีมาเป็นเครื่องใช้ประกอบการพัฒนาโครงงาน
2. การสร้างภาพ 3 มิติของวัตถุต่างๆ ที่จะนำมาแสดงเป็นสิ่งแวดล้อมของโปรแกรมเป็นงานที่ต้องใช้เวลาในการทำอย่างมาก จึงไม่สามารถออกแบบวัตถุให้มีรายละเอียดมาก ๆ ได้ อีกทั้งถ้านำวัตถุที่มีรายละเอียดมากๆ มาใช้ จะทำให้การแสดงภาพ 3 มิติช้าลงไปมาก ซึ่งแนวทางในการแก้ไขปัญหานี้ คือ ออกแบบวัตถุให้มีรายละเอียดน้อยที่สุด หรือถ้าเป็นวัตถุที่มีรายละเอียดมาก ก็จะใช้วิธีการหาจากอินเตอร์เนตแล้วนำมาประยุกต์ใช้ในโปรแกรม
3. ไลบรารี่ (Library) ภายในเกมเอนจิ้นท์ที่ใช้ (Diesel Engine) เป็น Open source และยังไม่สมบูรณ์ในบางคำสั่งการทำงาน การแก้ไขทำได้โดยเปลี่ยนไปใช้เกมเอนจิ้นท์ที่มีมาตรฐานแน่นอน และมีความน่าเชื่อถือของผู้พัฒนาสูง เช่น OpenGL ES (ในขณะที่พัฒนาโครงงานนี้ เกมเอนจิ้นท์ OpenGL ES นี้ยังมีข้อมูลน้อย หาข้อมูลเชิงเทคนิค หรือตัวโปรแกรมได้ยาก และเกมเอนจิ้นท์อีกหลายตัวก็เป็นการค้าต้องสมัครและชำระเงินซื้อ)
4. อัลกอริทึมสำหรับให้มอนสเตอร์คิดคำนวณการกระทำในการต่อสู้ที่ทางผู้พัฒนาคิดประยุกต์ขึ้นมาใช้งานยังไม่สมบูรณ์ ยังมีการเลือกการกระทำที่ไม่เหมาะสมกับสถานการณ์อยู่บ้าง เนื่องจากมีเวลาในการคิด และพัฒนาจำกัด วิธีแก้คือนำความรู้ด้านปัญญาประดิษฐ์มาประยุกต์ใช้ เช่น ความรู้ในเรื่อง Reinforcement Learning, Q-Learning, Markov Decision Process

6.4 แนวทางการพัฒนาต่อ

1. พัฒนาในส่วนของกราฟิกสามมิติให้มีความสวยงามมากขึ้น และเพิ่มขนาดของเวิลด์ที่ใช้ในแอพพลิเคชันให้ใหญ่ขึ้น ทั้งนี้จะต้องคำนึงถึงความสามารถของเครื่องด้วยว่า มีความสามารถในการแสดงภาพ 3 มิติได้ดีเพียงใด
2. เพิ่มรายละเอียดของเกมที่ยังขาดไปให้ครบ เช่น ระบบเสียง, จำนวนตัวละครมอนสเตอร์ภายในเกม, จำนวนฉาก, ตกแต่งฉากให้ดูสมจริง และสวยงามมากขึ้น
3. ปรับปรุงฉากต่อสู้ให้มีกราฟิกเร้าใจมากยิ่งขึ้น
4. พัฒนาอัลกอริทึมสำหรับให้มอนสเตอร์คิดคำนวณการต่อสู้ให้ดีขึ้น ให้มอนสเตอร์มีความฉลาดขึ้น โดยนำความรู้ด้านปัญญาประดิษฐ์มาช่วย เช่น เรื่อง Reinforcement Learning, Q-Learning, Markov Decision Process
5. เปลี่ยนแปลงระบบเจริญเติบโตของตัวละครมอนสเตอร์ ในโครงงานนี้ตรวจสอบการเจริญเติบโตของมอนสเตอร์แบบ Rule Based ควรเปลี่ยนแปลงเป็นการ Learning จากสิ่งแวดล้อม ณ ปัจจุบันของตัวละคร อาจใช้ความรู้จากเรื่อง Neural Network มาประยุกต์ใช้ หรือทำเป็น Hybrid Intelligent Systems (จะใช้ระบบใดบ้างแล้วแต่การออกแบบและความเหมาะสม)
6. พัฒนาระบบผสมพันธุ์มอนสเตอร์ที่เครื่องเซิร์ฟเวอร์ให้สามารถรองรับมอนสเตอร์พ่อพันธุ์ – แม่พันธุ์ ได้มากกว่านี้ (ระบบที่พัฒนารองรับได้พร้อมกันไม่เกิน 5 เครื่อง)

บรรณานุกรม

ตัวอย่างเอกสารอ้างอิงที่เป็นหนังสือ

[1] Andrew Rolling and Dave Morris 2004 Game Architecture and Design , Indiana , New Rider Publishing

[2] Krell, Bruce E. Pocket PC developer's guide ,New York McGraw-Hill/Osborne , 2002

[3] นายทรงศักดิ์ ลิ้มสิริสันติกุล Pocket PC utility รวมสุดยอดโปรแกรมสารพัดประโยชน์บนเครื่องพ็อกเก็ตพีซี ,กรุงเทพ ซีเอ็ดยูเคชั่น, 2545

[4] นายเฉลิมพล โสมภีร์ และ นาย ภาณุรักษ์ รับผล 3ds max plug-ins, จัดพิมพ์โดยสำนักพิมพ์อินโฟเพรส พิมพ์ครั้งที่ 1,2545

ภาคผนวก

ภาคผนวก ก.

การติดตั้ง Embedded Visual C++ และ Pocket PC Emulator

ขั้นตอนการติดตั้ง

ขั้นตอนการติดตั้ง Embedded Visual C++ เพื่อใช้ในการเขียนโปรแกรมบน Pocket PC และตัว Emulator ของ Pocket PC มีขั้นตอนดังต่อไปนี้

1. จะต้องทำการดาวน์โหลดโปรแกรมทั้งหมดจาก http://www.microsoft.com ซึ่งประกอบด้วย
   - Embedded Visual C++ 4.0 เพื่อใช้ในกรเขียนโปรแกรมบน Pocket PC
   - Embedded Visual C++ 4.0 service pack 4 ไว้ใช้อัพเดท Embedded Visual C++ 4.0
   - Microsoft ActiveSync 3.7.1 เพื่อใช้ในการติดต่อกับ Pocket PC
   - Microsoft Pocket PC 2003 SDK ซึ่งเป็น Emulator ของ Pocket PC
2. ต่อไปจะเป็นการเริ่มต้นการติดตั้งโปรแกรมต่าง ๆ ที่ดาว์นโหลดมา โดยขั้นแรกให้ Uninstall ตัวโปรแกรม Embedded Visual C++ 3.0 และ Emulator เวอร์ชั่นเก่าออกก่อน
3. โปรแกรมแรกที่เราจะต้องทำการติดตั้งคือ โปรแกรม Microsoft ActiveSync 3.7.1 ซึ่งถ้าเรามี Visual studio.NET 2003 อยู่แล้ว เราต้องทำการ install ใหม่ หรือ repair โดยจะเป็นโปรแกรมที่ใช้เชื่อมต่อ PC กับ Pocket PC จริงๆ และสามารถจำลองการเชื่อมกับตัว Emulator ของ Pocket PC โดยต้อง install ก่อน Pocket PC 2003 SDK เสมอ
4. ต่อไปทำการติดตั้งโปรแกรมที่ใช้ เขียนโปรแกรมบน Pocket PC ซึ่งก็คือ Embedded Visual C++ 4.0 และตัวอัพเดท service pack 4 โดยต้องติดตั้งก่อน Pocket PC 2003 SDK เสมอ

   Note: ถ้าจะทำการติดตั้งบน Window Server 2003 อาจทำให้มี Dialog box แจ้งว่า Emulator driver ไม่สามารถมองเห็นได้เราควรทำการยอมรับว่ามองไม่เห็น และดำเนินขั้นตอนต่อไป

ทำการติดตั้ง Microsoft Pocket PC 2003 SDK เป็นขั้นตอนสุดท้าย