การพัฒนาเกมเรียลไทม์สแตรเทจี
RTS GAME DEVELOPMENT

นายเชิดศักดิ์ รุทธนานุรักษ์

นายธีระพงษ์ สุขสุวรรณ

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

การพัฒนาเกมเรียลไทม์สแตรเทจี
RTS GAME DEVELOPMENT

โดย

นายเชิดศักดิ์ รุทธนานุรักษ์

นายธีรพงษ์ สุขสุวรรณ

อาจารย์ที่ปรึกษา

อ . อภิเนตร อูนากูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตร์บัณฑิต

ภาควิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2543

ปริญญานิพนธ์ปีการศึกษา 2543

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาเกมเรียลไทม์สแตรเทจี

RTS GAME DEVELOPMENT

ผู้จัดทำ

1. นายเชิดศักดิ์ รุทธนานุรักษ์ รหัสประจำตัว 40010189
2. นายธีรพงษ์ สุขสุวรรณ รหัสประจำตัว 40010348

_____ อาจารย์ที่ปรึกษา

(อ. อภิเนตร อูนากูล)

การพัฒนาเกมเรียลไทม์สแตรเทจี

นายเชิดศักดิ์ รุทธนานุรักษ์
นายธีระพงษ์ สุขสุวรรณ
อาจารย์อภิเนตร อูนากูล อาจารย์ที่ปรึกษา
ปีการศึกษา 2543

บทคัดย่อ

เกมเรียลไทม์สแตรเทจีเป็นเกมที่ผู้เล่นดำเนินกลยุทธโดยทำการออกคำสั่งหรือควบคุมตัวละครฝ่ายตนให้ทำงานต่าง ๆ เช่น เก็บทรัพยากรนำมาเป็นปัจจัยในการผลิต การซ่อมแซม และการวิจัยพัฒนาเพื่อให้บรรลุถึงเป้าหมายที่ถูกกำหนดไว้ในเกม โดยเหตุการณ์ทั้งหมดจะเสมือนดำเนินไปพร้อมๆกัน ผู้ร่วมเล่นสามารถเป็นได้ทั้งผู้เล่นที่เป็นมนุษย์และผู้เล่นที่อัตโนมัติที่ถูกจำลองโดยคอมพิวเตอร์ โครงงานนี้เป็นการพัฒนาเกมเรียลไทม์สแตรเทจีในรูปแบบของเฟรมเวิร์ค/เอนจินที่รองรับการทำงานของเกมในหลากหลายรูปแบบ และเพื่อทดสอบว่าเฟรมเวิร์ค/เอนจินที่ได้สามารถใช้งานได้จริงจึงสมควรที่จะต้องนำเฟรมเวิร์ค/เอนจินที่ได้นี้ไปทดลองสร้างเกมขนาดเล็กที่ใช้งานได้ในระดับหนึ่ง

RTS Game Development

Cherdsak Ruttananurak

Teerapong Suksuwan

Apinetr Unakul Advisor

ABSTRACT

Real-Time Strategy(RTS) game is the game that player conduct his strategy by dictate or control his units to work, such as gathering resource for production, repair and research, in order to archive the goal of playing. All event is, virtually, progress simultaneously. Player can be either human or autonomous player, player simulated by computer. Project Objective is to develop RTS game in form of framework/engine that can support many style of game. To test if framework/engine being developped is really useable, we plan to develop a small RTS game with medium functionality using this framework/engine.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความช่วยเหลือและความร่วมมือ จากท่านผู้มีพระคุณทั้งหลายต่อไปนี้

- ขอขอบพระคุณ คณาจารย์ที่ได้ประศาสตร์ความรู้ให้ โดยเฉพาะอย่างยิ่ง อาจารย์อภิเนตร อูนากูล อาจารย์ที่ปรึกษาโครงงาน ที่ได้ให้ความรู้และคำชี้แนะในการค้นคว้าและให้คำปรึกษาในด้านต่าง ๆ เป็นอย่างมาก
- ขอขอบพระคุณ คุณพ่อ คุณแม่ ที่ให้การศึกษาและเลี้ยงดูให้เป็นสมาชิกที่ใฝ่ดีของสังคม
- ขอบคุณเจ้าของแหล่งความรู้ทั้งหลายจากทั่วทุกมุมโลก ที่กรุณาเอื้อเฟื้อข้อมูลจนโครงงานนี้สามารถดำเนินลุล่วงมาได้
- สุดท้ายขอขอบคุณเพื่อน ๆ ที่ช่วยบอกแหล่งในการค้นหาข้อมูลและแลกเปลี่ยนความรู้กันมา ณ. ที่นี้ด้วย

สารบัญ

สารบัญภาพ

บทที่1
บทนำ

1.1. ความหมายและที่มา

ในหัวข้อนี้จะเป็นการนำไปสู่โครงงานนี้โดยการอธิบายความหมายของโครงงานโดยจะเริ่มอธิบายเป็นส่วน ๆ จนได้ความหมายครบถ้วน แล้วต่อจากนั้นจะเป็นการกล่าวถึงจุดประสงค์และเรื่องเกี่ยวกับขอบเขตเป้าหมายของโครงงานนี้

1.1.1 เกม

เกมคอมพิวเตอร์คือ โปรแกรมคอมพิวเตอร์ที่มีจุดมุ่งหมายให้ผู้ใช้งานเกิดความเพลิดเพลินจากการใช้งานที่มีการโต้ตอบระหว่างผู้เล่นและโปรแกรมคอมพิวเตอร์หรือผู้เล่นคนอื่นอย่างต่อเนื่องเพื่อให้บรรลุเป้าหมายของเกมที่กำหนดขึ้นมา ความเพลิดเพลินจากเกมคอมพิวเตอร์นั้นมีหลายรูปแบบตามแต่ความคิดสร้างสรรค์และจินตนาการของมนุษย์

ในความหมายที่แคบลงมา เกมคอมพิวเตอร์เป็นโปรแกรมประเภทจำลองแบบ (Simulation Program) แต่สิ่งที่ทำการจำลองขึ้นมานั้นอาจจะจำลองแบบมาจากความเป็นจริงที่มีอยู่ในโลกหรือไม่ก็ได้ขึ้นอยู่กับจินตนาการและความมุ่งหมายและรูปแบบของเกมคอมพิวเตอร์ในการตอบสนองความต้องการของมนุษย์

1.1.2 RTS เกม

RTS Game ย่อมาจาก (Real-Time Strategy Game) ซึ่งถ้าแปลความหมายตามตัวจะเป็นดังนี้

Real-Time -> เวลาแบบเป็นจริง

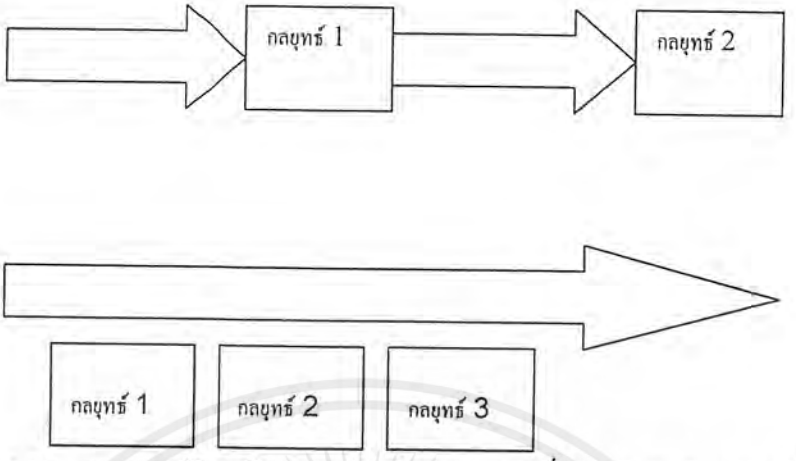
Strategy -> กลยุทธ์

Game -> *ความหมายดังได้กล่าวมาแล้วข้างต้น

เมื่อผสมความหมายเข้าด้วยกันน่าจะมีความหมายว่า "เกมที่ดำเนินกลยุทธ์ในเวลาจริง "
การแปลความหมายข้างต้นนั้นได้ความหมายอยู่บ้าง แต่ยังห่างไกลจากรูปแบบของเกมที่มีอยู่จริงในท้องตลาดเป็นอย่างมาก ดังนั้นในกรณีนี้ควรหาที่มาของชื่อมาจากความเป็นมาทางประวัติศาสตร์

ในช่วงแรกของวงการเกมในขณะนั้นจะมีเกมประเภท เทินเบสซึ่งมีลักษณะที่ผู้เล่นสองฝ่ายทำการวางกลยุทธ์ในการใช้ทรัพยากร ใช้ตัวละครฝ่ายของตน เข้าต่อสู้กับศัตรู โดยโอกาสในการดำเนินกลยุทธ์นั้นเป็นไปในลักษณะที่ผลัดกันเล่นคนละตา หากเปรียบเทียบแล้ว การดำเนินการจำลองสภาพระบบของเกมจะเป็นไปในรูปแบบที่ไม่ต่อเนื่อง ต้องมีการหยุดรอจังหวะให้ผู้เล่นคนอื่นที่อาจเป็นโปรแกรมคอมพิวเตอร์หรือมนุษย์ได้ผลัดเข้ามาเล่น เกมในรูปแบบนี้ประสพความสำเร็จในท้องตลาดพอสมควร

ต่อมาสมรรถนะของเครื่องได้ถูกให้เร็วขึ้นในระดับหนึ่ง ซึ่งเป็นผลให้มีการพัฒนาเกม ดูนทู ออกมาซึ่งนับได้ว่าเป็นเกมแรกที่ทำให้เกิด คำว่า RTS เกม โดยรูปแบบของการเล่นจะเป็นไปในรูปแบบที่ต่อเนื่อง การตัดสินใจเล่นเสมือนการตัดสินใจตามเหตุการณ์ในรูปแบบที่เสมือนจริง



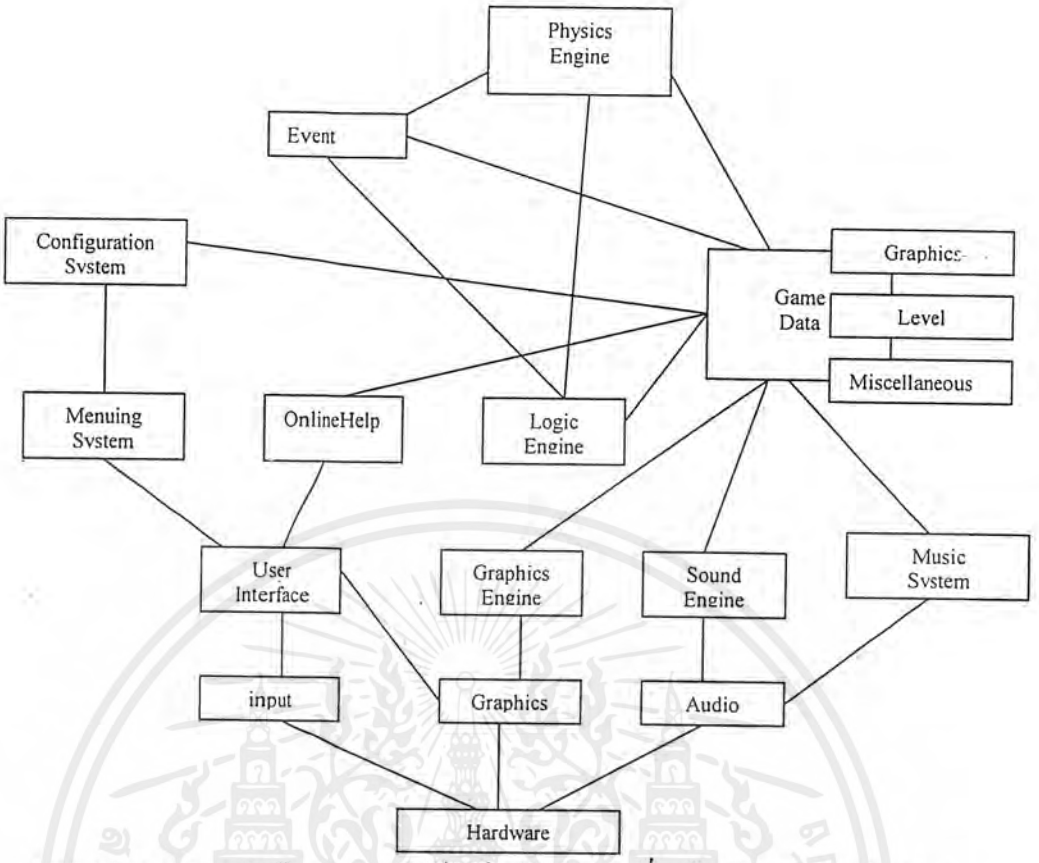
ภาพ 1.1.2 ก แสดงการเปรียบเทียบการเล่น เทินเบส เกม (ครึ่งบน) และ RTS เกม (ครึ่งล่าง)

จากประวัติข้างต้น RTS เกม จึงเป็นชื่อที่ถูกตั้งในเชิงเปรียบเทียบกับเกมในลักษณะที่คล้าย ๆ กัน แต่เดิมแต่เป็นชื่อที่เน้นให้เกิดความหมายเน้นความแตกต่างของรูปแบบในการทำการเล่นจากเกมประเภทเทินเบส อย่างไรก็ตามรูปแบบของ RTS เกมได้มีการพัฒนาไปหลากหลายรูปแบบจนบางครั้งอาจทำให้เกิดการสับสนระหว่าง RTS เกม และ ซิมมูเลชัน เกม แต่อย่างไรก็ตามในตลาดเกมคอมพิวเตอร์นั้นแยกแยะ RTS เกม และ ซิมมูเลชัน เกม ออกจากกันโดย RTS เกม จะมีการควบคุมตัวละครที่ใกล้ชิดกว่า และมีลักษณะ กึ่ง แอคชั่นเกม ส่วนซิมมูเลชันเกมนั้นไม่ได้ทำการควบคุมโดยตรงแต่จะเป็นการกำหนดตั้งค่าปรับแต่งต่าง ๆ ในการควบคุมตัวละครตอบสนองต่อสถานการณ์ต่าง ๆ

ดังนั้น โดยสรุป RTS เกม ที่นิยามโดยวงการเกมคอมพิวเตอร์คือ เกมที่ผู้เล่นสั่งการตัวละครฝ่ายตรงในการจัดหาทรัพยากร สร้างเสริมบำรุงเพิ่มตัวละครของฝ่ายตนเอง เพื่อนำมาใช้งาน ต่อสู้ศัตรู เพื่อให้บรรลุเป้าหมายต่าง ๆ ตามที่กำหนดไว้ โดยการดำเนินกลยุทธ์จะเป็นไปในแบบเสมือนจริงโต้ตอบอย่างต่อเนื่อง

1.1.3 สถาปัตยกรรมเกม (Game Architecture)

เกมคอมพิวเตอร์ก็เป็นซอฟต์แวร์แบบหนึ่งซึ่งโดยธรรมชาติแล้วซอฟต์แวร์ทุกชนิดจะมีสถาปัตยกรรมพื้นฐานที่มีลักษณะร่วมกันกับซอฟต์แวร์ในแบบเดียวกัน ซึ่งในส่วนหลักจะมีความแตกต่างกันไม่มากนัก ดังนั้นหากเราวางแผนจะสร้างซอฟต์แวร์ ประเภทใด ๆ ควรจะศึกษาถึงสถาปัตยกรรมของซอฟต์แวร์ประเภทนั้นๆ เสียก่อน โดยทั่วไปแล้วสถาปัตยกรรมพื้นฐานของแอพพลิเคชั่นประเภทเกมคอมพิวเตอร์ประกอบไปส่วนระบบย่อยดังนี้



ภาพ 1.1.3 ก แสดงสถาปัตยกรรมซอฟต์แวร์เกมและความเกี่ยวข้องกันของส่วนประกอบ

- ระบบย่อยที่เป็นส่วนหลัก
  - ยูสเซอร์อินเทอเฟส (User Interface)
    ทำหน้าที่ติดต่อระหว่างผู้ใช้กับระบบ รับการควบคุมจากผู้ใช้และ แสดงผลและให้วิธีการควบคุมที่สะดวกสบายแก่ผู้ใช้ เพื่อนำไปส่งต่อให้กับส่วนควบคุมของระบบเกม ระบบยูสเซอร์อินเทอเฟสที่เป็นระบบหลักของเกมทุกเกมที่เห็นได้ในปัจจุบันคือระบบ กราฟิกยูสเซอร์อินเทอร์เฟสในรูปแบบวินโดว์
  - เดต้าเอนจิน (Data Engine)
    ทำหน้าที่ในการจัดการข้อมูลเช่นการ โหลด เซฟ เกม การ โหลด รูปภาพ เสียง ฉาก เข้ามาในระบบควบคุมระบบร่วมใช้ ออบเจ็กต์
  - ระบบจัดการเหตุการณ์พื้นฐานแบบสองทาง (Bidirectional Event Handler)
    รับการร้องขอจากตัวละครในเกม เพื่อไปประมวลผลเหตุการณ์และส่งกลับผลลัพธ์ของเหตุการณ์ที่ร้องขอมา
  - ระบบจัดการพลวัตเพื่อบังคับกฎทางกายภาพต่างๆ (Dynamic Physical System)
    บังคับกฎของระบบทางกายภาพของโลกในเกมเช่นการตรวจสอบการชน การมองเห็น มิติความสูง โมเดลทางกายภาพของโลกในเกม

- ลอจิกเอนจิน (Logic Engine) กฎของเกมที่เป็นหัวใจของเกมเป็นส่วนที่ทำให้เกิดความแตกต่างไปในแต่ละเกม เช่นการบังคับเรื่องราวเนื้อหา จัดการลำดับเรื่องราว ตรวจสอบการแพ้ชนะ การเปลี่ยนและหักเหของเรื่องราว
- กราฟิกเอนจิน (Graphic Engine)
  ส่วนที่ทำหน้าที่แสดงโลกในเกมออกมาเป็นรูปภาพ ในมิติต่างๆเป็นส่วนสำคัญที่ทำให้เกมมีเอกลักษณ์และดึงดูดให้ผู้เล่นสนใจและมีความสนุกสนานในการเล่น
- เอนจินเสียง (Sound Engine)
  รองรับการส่งเสียงผ่านอุปกรณ์เสียง อาจจะเป็นแบบโดยตรงหรือมีโมเดลเสียงแบบพิเศษต่าง ๆ เช่นระบบเสียงทิศทางที่เข้ากับเหตุการณ์ในเกม
- ระบบชั้นพื้นฐานแอปสแตรกฮาร์ดแวร์ (HAL)
  ให้อินเทอร์เฟสร่วมสำหรับติดต่อกับฮาร์ดแวร์ที่มีหลากหลายในท้องตลาด ให้ข้อมูลเกี่ยวกับสถานะและความสามารถของฮาร์ดแวร์ประเภทนั้น ๆ ให้อินเทอร์เฟสมาตรฐานสำหรับการใช้ความสามารถพิเศษสำหรับฮาร์ดแวร์นั้น

- ระบบย่อยที่เป็นส่วนรองได้แก่
  - ระบบตั้งค่าเกม (Game Configuration System)
    ให้การเข้าถึงค่าที่ถูกตั้งสำหรับผู้ใช้และองค์ประกอบที่ต้องการใช้ค่านั้นในระบบ ทำหน้าที่เปลี่ยนแปลงค่าที่ถูกตั้งใหม่ให้เป็นผลในระบบ
  - ระบบจัดการเมนู (Menuing System)
    สร้าง และกำหนดผลที่เกิดขึ้นจากการเลือกในเมนูของยูเซอร์อินเทอเฟส
  - ระบบให้การช่วยเหลือและแสดงคำสั่งแบบออนไลน์ (Online Instruction And Help System)
    แสดงคำสั่งและข้อความช่วยเหลือจัดการเกี่ยวกับข้อความตัวหนังสือและแสดงตัวอย่างการเล่นแก่ผู้เล่น
  - ระบบดนตรี (Music System)
    เล่นดนตรีจากโน้ตดนตรีที่เก็บไว้เพื่อให้สร้างความรู้สึกร่วมในเหตุการณ์แก่ผู้เล่น

1.1.4 Game Engine

โปรแกรมประเภทเกมคอมพิวเตอร์ในมุมมองของผู้พัฒนา สามารถแยกส่วนออกเป็นสามส่วนใหญ่ ๆ นั่นคือ

1. สื่อต่าง ๆ เช่น ภาพกราฟิก เสียง โน้ตดนตรี ตัวหนังสือ บทกำกับการดำเนินเรื่อง เรื่องราว เนื้อหา ภาพยนตร์คั่นเกม
2. กฎ กติกา ของเกม
3. กลไกการทำงานที่ทำหน้าที่ขับเคลื่อนสิ่งต่าง ๆ จากข้อ 1 และ 2 ให้โลดแล่นเป็นเกมตามที่ได้ออกแบบไว้ในขั้นตอนการออกแบบเกม

ส่วนที่ 3 ที่ได้นำเสนอนั่นก็คือเกมเอนจินนั่นเอง คำว่าเอนจินเป็นศัพท์ที่เรียกกันในเชิงมุมมองของผู้ใช้งาน นั่นคือเรามองมันในแง่ขณะที่มันทำงาน แต่ศัพท์ในวงการวิศวกรรมซอฟต์แวร์ที่น่าจะใช้แทนคำว่า เอนจินได้น่าจะเป็นคำว่า เฟรมเวิร์ก ซึ่งเป็นการมองในแง่การนำไปพัฒนาต่อ ดังนั้นจึงยกความหมายของคำว่าเฟรมเวิร์ก เพื่อนำมาขยายความคำว่าเอนจินเพราะโครงงานนี้น่าจะใช้มุมมองในแง่ของผู้พัฒนาเอนจิน

เฟรมเวิร์ก คือ สถาปัตยกรรมทั่วไปของ แอพพลิเคชั่นภายในโดเมนหนึ่งซึ่งให้เทมเพลตสำหรับการขยายสำหรับการนำไปประยุกต์ใช้

เฟรมเวิร์ก จะชี้นำสถาปัตยกรรมของแอพพลิเคชั่น เช่น โครงสร้างโดยรวม การแบ่งคลาสและออบเจ็กต์ กำหนดหน้าที่หลักของคลาสและอ๊อบเจ็กต์เหล่านั่น รวมทั้ง การประสานงาน และ โมเดลของการควบคุมทำงาน เฟรมเวิร์ก ทำให้ผู้ใช้ เจาะจงไปที่งานในโดเมนของตน โดยปิดบังรายละเอียดของการอิมพลีเมนต์เอาไว้ให้ผู้ใช้เกิดความสะดวกสบายในการนำไปใช้ในโดเมนของตน

1.1.5 RTS เกมเอนจิน

RTS เกมเอนจิน ก็เป็นเกมเอนจินแบบหนึ่งซึ่งมีลักษณะร่วมบางส่วนเหมือนกับเกมเอนจินทั่วไป ซึ่งในหัวข้อที่ผ่านมาได้อธิบายสถาปัตยกรรมทั่วไปของเกมไปแล้ว ในหัวข้อนี้เพื่อให้เน้นให้เห็นถึงลักษณะเด่นที่ทำให้ RTS เกมต่างจากเกมประเภทอื่น

1. เหตุการณ์ ในระบบมีขนาดที่เล็กย่อยมาก และต้องการการจัดการสุ่มเหตุการณ์ก่อนการประมวลผลเพื่อเพิ่มความยุติธรรมของเกมมีสูง
2. ความสัมพันธ์ระหว่างตัวละครในเกมและผู้เล่นจะเป็นไปในลักษณะสั่งการณ์ให้ทำงาน ไม่ได้ควบคุมตัวละครโดยตรง ดังนั้นตัวละครจึงมีต้องมีการพฤติกรรมที่ในรูปแบบที่อัตโนมัติ และมีความฉลาดในการแก้ปัญหาเล็กน้อยที่ในระดับที่ไม่เกี่ยวกับกลยุทธ์ของผู้เล่นได้ด้วยตัวเอง
3. ความสัมพันธ์ระหว่างผู้เล่นและตัวละครในเกมเป็นแบบ 1 : N นั่นคือผู้เล่นสามารถสั่งการณ์ตัวละครเป็นกลุ่มได้
4. เหตุการณ์ทั้งหมดดำเนินไปอย่างต่อเนื่อง เสมือนเกิดขึ้นแบบพร้อมกัน
5. ผู้เล่นที่เป็นฝ่ายศัตรูดำเนินกลยุทธ์ที่เกิดจากปัญญาประดิษฐ์เพื่อให้บรรลุจุดประสงค์เลียนแบบการเล่นของมนุษย์ที่เป็นผู้เล่น

1.2 วัตถุสงค์ของโครงงาน

1. ศึกษาเกี่ยวกับสถาปัตยกรรมซอฟต์แวร์ประเภท RTS เกม
2. ออกแบบและสร้าง RTS เกมเอนจินเพื่อนำความรู้ที่ได้ไปประยุกต์ในการสร้างเฟรมเวิร์กซอฟต์แวร์ในรูปแบบที่ใกล้เคียงกันได้
3. สรุปผลและหาความเป็นไปได้ในการพัฒนาต่อไปจนสามารถถึงขั้นที่จะใช้เชิงพาณิชย์ได้

1.3 ขอบเขตของโครงงาน

1. ศึกษาความรู้พื้นฐานที่เกี่ยวกับการพัฒนา RTS เกมเอนจิน
2. ออกแบบและสร้าง RTS เกมเอนจิน
3. ทดลองนำ RTS เกมเอนจิน ไปสร้างเกมตัวอย่างขนาดเล็ก และสรุปผลการทำงาน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้ความรู้เกี่ยวกับ สถาปัตยกรรมซอฟต์แวร์เกม
2. ได้ประยุกต์ใช้ความรู้ที่ได้จากการศึกษาที่ผ่านมาใน
3. เป็นแนวทางให้ผู้อื่นสามารถนำไปศึกษาต่อได้

1.5 วิธีการดำเนินงาน

1. ศึกษาและรวบรวมข้อมูลเกี่ยวกับเกมและสถาปัตยกรรมเกมแบบ RTS จากแหล่งต่าง ๆ
2. ออกแบบและวางแผนการสร้างและสร้างซอฟต์แวร์และดำเนินการทำแบบแนวอิเทอร์เรทีพ
3. ทดสอบและสรุปผลการทำงาน

บทที่ 2

หลักการในการออกแบบ RTS เกม

การออกแบบเกม เป็นกระบวนการที่ใช้จินตนาการและความสร้างสรรค์ของมนุษย์เพื่อสร้างการละเล่นที่สามารถให้ความบันเทิงสนุกสนานแก่มนุษย์ได้ การละเล่นต่าง ๆ จะถูกกำหนดกติกาเงื่อนไขบังคับให้อยู่ในกรอบของโลกจำลองภายในเกม มีการกำหนดบทบาทหน้าที่ของผู้เล่น เป้าหมายของการเล่น

คอมพิวเตอร์เกมก็เป็นรูปแบบหนึ่งของเกมที่ใช้ระบบคอมพิวเตอร์เป็นสื่อในการถ่ายทอดจินตนาการและความสร้างสรรค์เหล่านั้นรูปแบบหนึ่งซึ่งให้ความสมจริงสมจังสูงเพราะความเร็วของคอมพิวเตอร์ในการบังคับต่าง ๆ สามารถทำได้ซับซ้อนในเวลาอันรวดเร็ว ทำให้เกิดความเสมือนจริงสูงทำให้ จึงกล่าวได้ว่าระบบคอมพิวเตอร์เป็นตัวถ่ายทอดจินตนาการและความสร้างสรรค์เหล่านั้นได้เป็นอย่างดี แต่อย่างไรก็ตามระบบคอมพิวเตอร์ก็เป็นเพียงตัวถ่ายทอดเท่านั้นไม่ได้เป็นองค์ประกอบทั้งหมดของความสำเร็จของเกมที่มุ่งหมายได้ผู้เล่นเกิดความบันเทิงสนุกสนาน จึงสรุปได้ตัวเกมเป็นหัวใจสำคัญของความสนุกสนาน ซึ่งถูกรองรับโดยระบบคอมพิวเตอร์ และความสนุกสนานมักแปรผันตรงกับเสมือนจริงซึ่งได้มาจากการทำงานที่ซับซ้อนของระบบคอมพิวเตอร์

การสร้าง เฟรมเวิร์ก/เอนจินใด ๆ ขึ้นมารองรับจึงควรศึกษาถึงการออกแบบเกมด้วยเพราะตัว เฟรมเวิร์ก/เอนจิน นั้นเป็นส่วนที่ทำหน้าที่รองรับจินตนาการที่เกิดจากการออกแบบเกมให้ทำงานได้เป็นจริง จินตนาการและความสร้างสรรค์ของมนุษย์ในการออกแบบเกมจึงนับได้ว่าเป็นความต้องการของการสร้างเฟรมเวิร์ก/เอนจิน ดังนั้นเป้าหมายของ เฟรมเวิร์ก/เอนจินสำหรับการออกแบบเกมน่าจะหมายถึงการสามารถตอบสนองต่อจินตนาการเหล่านั้นได้โดย ปิดบังการทำงานในระดับที่ไม่เกี่ยวข้องออกไปให้มากที่สุด และมีความยืดหยุ่นเปลี่ยนแปลงรองรับรูปแบบของจินตนาการได้มากที่สุดโดยมีการเปลี่ยนแปลงโครงสร้างของเอนจินให้น้อยที่สุด

เนื่องจากการที่ RTS เกมถูกนิยามให้มีความหมายตามรูปแบบที่ปรากฏให้เห็นในเกมประเภทนี้ในปัจจุบัน ดังนั้นการออกแบบเกมที่ปรากฏอยู่ในตลาดจึงน่าจะเป็นความต้องการพื้นฐานของเกมเอนจินที่ควรศึกษาก่อนเป็นอันดับแรกส่วนการรองรับอื่นๆ ที่อาจเกิดในอนาคตนั้น ในระดับนี้จึงไม่ควรคาดเดาความต้องการไปล่วงหน้ามากเกินไปนักแต่ควรออกแบบระบบให้มีความยืดหยุ่นเพียงพอตามหลักการของวิศวกรรมซอฟต์แวร์

2.1 แนวคิดเบื้องต้น

ขั้นตอนนี้เป็นขั้นตอนแรกสุดของการออกแบบเกม เป็นกระบวนการแห่งความสร้างสรรค์ที่เกิดจากการใช้การนึกฝันและจินตนาการของมนุษย์

2.1.1 การได้มาซึ่งแนวคิด

ประกอบด้วย 4 ช่วงของได้แก่

1. แรงบันดาลใจ (Inspiration)

แรงบันดาลใจของมนุษย์มักเกิดจากการรับรู้เรียนรู้แล้วเกิดความซาบซึ้งและเข้าใจในเรื่องนั้นก่อให้เกิดสิ่งที่เรียกว่าแรงบันดาลใจ ในวงการเกมแรงบันดาลใจเหล่านี้มาได้หลายทางด้วยกันเช่น

- ธรรมชาติ
- ประวัติศาสตร์
- นิยายปรัมปรา
- นิยายทางวิทยาศาสตร์
- สิ่งประดิษฐ์และผลงานทางวิทยาศาสตร์
- สภาพการดำเนินชีวิต การทำงาน
- ภาพยนตร์
- กีฬาและนันทนาการ
- เกมคอมพิวเตอร์อื่น
- ฯลฯ

เกมหนึ่ง ๆอาจจะถูกสร้างด้วยแรงบันดาลใจที่มากกว่าหนึ่งก็ได้และมักจะเป็นอย่างนั้นเสียเป็นส่วนใหญ่

2. การสังเคราะห์ (Synthesis)

หลังจากที่แรงบันดาลใจได้ก่อร่างตัวขึ้นมาแล้ว ขั้นตอนนี้จะเป็นการเติมเต็มโดยนำแรงบันดาลใจที่เกิดขึ้นมามาผสมผสานเรียบเรียงต่อเติมให้เกิดความสมบูรณ์มากขึ้น โดยใช้เหตุผล อารมณ์ รสนิยม เป็นเครื่องมือ

3. การสร้างความสอดคล้องแนวคิด (Resonance)

หลังจากช่วงการสังเคราะห์แล้ว แนวคิดจะถูกขยายออกจากการสร้างความสอดคล้องของความคิด ความสอดคล้องกันทำให้แนวคิดที่อาจมาจากหลายแรงบันดาลใจ กลมกลืนและรวมตัวเป็นสิ่งที่ยิ่งใหญ่มากขึ้นกว่าเดิม

4. คอนเวอร์เจนต์ (Convergence)

หลังจากการขยายของแนวคิดจากการสร้างความสอดคล้องแล้วช่วงนี้ความคิดจะถูกประเมินคุณค่าทดสอบ ทำให้ความคิดถูกบีบลงมาเพื่อให้เกิดความเป็นไปได้ ทั้งในแง่การยอมรับของผู้เล่น และทางเทคนิค

2.1.2 การสร้างแนวคิดเป็นรูปร่าง

การสร้างแนวคิดเป็นรูปร่างนั้นต้องใช้หลักการคิดในรูปแบบเดียวกับการแต่งละครซึ่งต้องคำนึงถึง

- สไตล์

สิ่งที่สร้างความเป็นเอกลักษณ์ของแนวคิดของคุณแม้เกมอาจจะออกมาในแง่การทำงานที่เหมือนกันแต่สไตล์ทำให้เกิดรูปแบบที่แตกต่างขึ้นมา

- บท

ลำดับของเรื่องราวของเหตุการณ์ที่เกิดขึ้นภายในเกม

- ตัวละคร

  เป็นส่วนสำคัญในเกมที่จะทำให้เกมเกิดเอกลักษณ์และความแต่งต่าง ตัวละครที่มีบุคลิกเด่นชัดเข้ากับเกมอาจถึงกับทำให้ชี้เป็นชี้ตายของความสำเร็จ ในเกมบางประเภทอาจเป็นสัญลักษณ์ตัวแทนของเกมนั้น ๆ
- การตั้งค่า

  รายละเอียดและความสัมพันธ์ของสิ่งต่าง ๆ ภายในเกม
- ตีม

  หัวใจของบทละครที่ทำให้การถ่ายทอดแรงบันดาลใจสมบูรณ์

2.1.3 การนำเสนอแนวคิด

ขั้นตอนนี้คือการนำเสนอแนวคิดต่อทีมผู้สร้าง เจ้าของทุน ฯลฯขั้นนี้จะมีการประนีประนอม ระหว่างความฝันและความเป็นจริงทั้งทางด้านเทคนิค การตอบรับของเจ้าของทุน

2.2 การออกแบบแกนหลัก (Core Design)

จุดมุ่งหมายของขั้นตอนนี้คือให้ได้มาซึ่ง เกม สเปคซิฟิเคชั่นซึ่งต้องกำหนดสิ่งต่างๆ ดังนี้ ซึ่งในขั้นนี้จะมุ่งลงไปเฉพาะของเกมแนว RTS

2.2.1. คุณลักษณะ

คุณลักษณะที่ควรพิจารณาเป็นประเด็นได้ดังนี้

- มิติของโลกจำลองปัจจุบัน
  - สองมิติ
  - สามมิติ
  - สามมิติเทียม
- การประสานงานภายในของกลุ่มตัวละคร
  - การร่วมกันต่อสู้
  - การร่วมกันทำงาน
  - การตอบโต้อัตโนมัติ
- การจัดรูปแบบแถว
  - จัดแถวการเดิน
  - การจัดแถวหลบหลีกอัตโนมัติ
- การใช้ ปัญญาประดิษฐ์ควบคุมการทำงานที่เป็นอัตโนมัติ
  - การทำงานประจำตามหน้าที่ของตัวละคร
  - การเปลี่ยนสถานะ การต่อสู้

2.2.2 การเล่น

- รูปแบบหลักของการเล่น

การเล่นเกมแนว RTS ในปัจจุบันประกอบด้วยส่วนผสมของสองแนวคิด

- เน้นหนักให้ผู้ใช้ใช้การตัดสินใจในการต่อสู้

  เปิดโอกาสให้ผู้เล่นสามารถควบคุมตัวละครให้ทำการ ต่อสู้ได้อย่างใกล้ชิด

- เน้นหนักให้ผู้ใช้ใช้การตัดสินใจในการสร้าง

  เปิดโอกาสให้ผู้เล่นควบคุมการสร้างการผลิตที่มีความซับซ้อน

การเล่นเกม RTS ในรูปแบบที่ปรากฏในช่วงหลังจะมีแนวคิดของการผสมผสานทั้งสองข้อโดยเปิดโอกาสให้ผู้เล่นเลือกได้ว่าต้องการเน้นในส่วนใดตามความชอบของตนเอง โดยงานส่วนที่ไม่สนใจควบคุมก็จะถูกควบคุมด้วยกลไกอัตโนมัติ

- สมดุลของการเล่น

ความสมดุลของการเล่นนับเป็นปัจจัยสำคัญที่ชี้ความประสพความสำเร็จของเกมความสม ดุลของเกมสามารถมองได้ในหลายระดับด้วยกันได้แก่

- ความสมดุลระหว่างความสามารถของเผ่าพันธุ์แต่ละฝ่าย

  คำนึงถึงกฎเกณฑ์ในการใช้พื้นที่และทรัพยากรว่ามีความจำกัดแค่ไหน

- ความสมดุลระหว่างความสามารถของตัวละครที่มีอยู่ในแต่ละฝ่าย

  ความสามารถของตัวละครแต่ละฝ่ายนั่นมีจุดได้เปรียบจุดแก้ไขได้หรือไม่ ความยากง่ายใน การแก้จุดเสียเปรียบเป็นอย่างไร

- กฎ

กฎของการเล่นเกมแนว RTS ได้มีการพัฒนาขึ้นมามีรูปแบบมากขึ้นพอเรียบเรียงมาได้ดังนี้

1. ชนะจากการทำให้ตัวละครฝ่ายตรงข้ามหมด
2. ชนะจากการสะสมทรัพยากรมาครอบครองได้ตามจำนวนกำหนด
3. ชนะจากการปกป้องสิ่งใดสิ่งหนึ่งจากการถูกทำลายได้ตามระยะเวลาที่กำหนด
4. ชนะจากการสร้างสิ่งใดสิ่งหนึ่งให้คงอยู่ได้ตามระยะเวลาที่กำหนด
5. ชนะจากการเดินทางไปถึงเป้าหมายที่กำหนด
6. ชนะจากความก้าวหน้าทางเทคโนโลยีที่เหนือกว่า

2.2.3 การติดต่อกับผู้ใช้

โดยทั่วไปรูปแบบการติดต่อก็จะเป็นไปในลักษณะวินโดว์ซึ่งประกอบด้วย

1. หน้าต่างสังเกตหลัก (World View)
2. หน้าต่างสังเกตย่อ (Mini Map)
3. หน้าต่างแสดงทรัพยากร (Resource View)
4. หน้าต่างแสดงข้อความช่วยเหลือ (Help View)
5. หน้าต่างแสดงคำสั่ง (Instruction View)
6. หน้าต่างแสดงรายละเอียดตัวละครที่สนใจอยู่ (Detail View)

7. หน้าต่างแสดงคำสั่งที่สั่งตัวละครที่สนใจอยู่ได้ (Command view)

2.2.4 การออกแบบด่าน

การออกแบบด่านจะมีรูปแบบทั่วไปแทบจะเหมือนกันหมดคือ เริ่มจากการเล่นที่ง่ายไม่ซับซ้อนก่อนแล้วจึงเพิ่มระดับความยากและความซับซ้อนตามความชำนาญที่เพิ่มขึ้นของผู้เล่น สถานการณ์ส่วนการออกแบบด่านสำหรับการเล่นหลายคนจะเป็นการมุ่งเน้นให้เกิดความสมดุลตั้งแต่ต้นและ เน้นให้เกิดจุดที่มีการขัดแย้งกันทางผลประโยชน์

อ้างอิง

หัวข้อ 2.1 และ 2.2 จาก [2] Andrew Rollings, Dave Moris GAME ARCHITECTURE AND DESIGN

บทที่ 3
การวิเคราะห์และออกแบบเกมด้วยแนวคิดเชิงวัตถุ

3.1. การประยุกต์กระบวนทัศน์เชิงวัตถุ

เพื่อที่จะให้การประยุกต์ใช้แนวคิดเชิงวัตถุให้ได้ผลเราควรกลับมาทบทวนถึงหลักการของแนวคิดเชิงวัตถุเสียก่อน โดยจะสนใจเฉพาะแนวคิดที่เป็นแกนกลางก่อนแล้วจึงนำความรู้เฉพาะในด้านเกมมาประยุกต์ใช้

3.1.1 หลักการของแนวคิดเชิงวัตถุออบเจ็กต์

- ออบเจ็กต์

  สิ่งที่เราสนใจและเจาะจงมองให้เป็นหน่วยเดียว

- เอนแคปซูเลชั่นเชิงวัตถุ
  - เอนแคปซูเลชั่น

    การรวมกลุ่มทางความคิดเข้าเป็นหนึ่งหน่วย แล้วตั้งชื่อที่ใช้เรียกหน่วยนั้น
  - เอนแคปซูเลชั่นเชิงวัตถุ

    กลุ่มของโอเปอเรชัน และกลุ่มของแอทริบิวต์ที่ใช้แทนสถานะ ที่ถูกรวมเข้าเป็นหนึ่งหน่วย การเข้าถึงแอททริบิวต์ต่าง ๆจะต้องเข้าถึงโดยผ่านอินเทอเฟสที่กำหนดไว้แน่ชัด
  - การซ่อนสารสนเทศ (Information hiding)

    การใช้เอ็นแคปซูเลชั่นเพื่อจำกัดการเข้าถึงจากภายนอก ต่อ สารสนเทศหรือการ ตัดสินใจในการอิมพลีเมนต์ภายในโครงสร้างที่ถูกเอนแคปซูเลต

- การคงสถานะ (Persistent)

  การคงค่าของสถานะในเวลาแห่งการคงอยู่ของออบเจ็กต์

- ไอเดนติตี้

  คุณสมบัติที่มีโดยธรรมชาติของออบเจ็กต์ที่แทนการมีอยู่

- แมสเซจ

  พาหะที่ออบเจ็กต์หนึ่งใช้ส่งผ่านความต้องการไปสู่อีกอ๊อบเจ็กต์หนึ่งเพื่อให้ออบเจ็กต์ตัวหลังดำเนิน การใด ๆผ่านทางเมธอดของตน

- คลาส

  แบบที่ใช้สร้างออบเจ็กต์

- การสืบทอด (Inheritance)

  การที่คลาสหนึ่งคลาสใด รับพฤติกรรมและคุณสมบัติมาจากอีกคลาสหนึ่งโดยปริยาย

- พอลิมอฟิซึม

  ความสามารถของออบเจ็กต์ที่ถูกจัดให้อยู่ในคลาสได้มากกว่าหนึ่ง ในต่างเวลาและโอกาส

3.1.2 การประยุกต์ใช้กระบวนทัศน์เชิงวัตถุในเกม

หลักการของแนวคิดเชิงวัตถุนั้นเริ่มแพร่หลายในวงการคอมพิวเตอร์จากด้านการทำซิมมูเลชันก่อน จึงเริ่มนิยมอย่างแพร่หลายในด้านอื่น ๆ ดังที่กล่าวมามาในบทที่หนึ่งแล้วว่าโปรแกรมเกมสามารถมองเป็นโปรแกรมซิมมูเลชันแบบหนึ่งได้ดังนั้น แนวคิดเชิงวัตถุจึงนำมาประยุกต์ในเกมได้แบบตรง ๆ เลยก็ว่าได้ ดังนั้นเราจึงเริ่มต้นจากออบเจ็กต์โมเดลที่เป็นแนวคิดใน การจำลองแบบแบบเหตุการณ์ไม่ต่อเนื่อง (Discrete Event Simulation) ก่อนแล้วจึงเปรียบเทียบโดยตรงกับลักษณะของเกม

- ออบเจ็กต์โมเดลจากแนวคิด การจำลองแบบแบบเหตุการณ์ไม่ต่อเนื่อง
  - ระบบ
    กลุ่มของเอนติตี้ที่มีปฏิสัมพันธ์กันในช่วงเวลาหนึ่งเพื่อให้บรรลุเป้าหมายร่วมกัน
  - โมเดล
    ตัวแทนนามธรรมของระบบซึ่งประกอบด้วยเอนติตี้และความสัมพันธ์ต่อกัน
  - สถานะระบบ
    ค่าที่คงอยู่ในเวลาหนึ่งๆของระบบเพื่ออธิบายสภาวะของระบบ
  - เอนติตี้
    สิ่งที่เราสนใจและแยกออกเป็นเอกเทศจากสิ่งอื่น
  - แอททริบิวต์
    คุณสมบัติของแต่ละเอนติตี้
  - ลิสต์
    กลุ่มของเอนติตี้ที่ถูกจัดรูปลำดับในลักษณะใดลักษณะหนึ่ง
  - อีเวนต์
    สิ่งที่เกิดขึ้นมาเพื่อเปลี่ยนสถานะของระบบ
  - บันทึกอีเวนต์
    การเก็บบันทึกอีเวนต์เพื่อให้สามารถจัดการกับอีเวนต์ให้เกิดผลหรือติดตามความเปลี่ยนแปลงได้ในภายหลัง
  - อีเวนต์ลิส
    ลิสต์ของบันทึกอีเวนต์เรียงลำดับตามเวลาของการเกิด
  - กิจกรรม
    ช่วงเวลาจำกัดแน่ชัดที่เรารู้เมื่อเริ่ม
  - ดีเลย์
    ช่วงเวลาที่ไม่ได้จำกัดแน่ชัด ซึ่งไม่รู้จนกว่ามันจะจบ
  - นาฬิกา
    ตัวแปรที่ใช้แทนเวลาของระบบ

เห็นได้ว่าออบเจ็กต์โมเดลตามแนวคิดการจำลองแบบแบบเหตุการณ์ไม่ต่อเนื่องนั้นมีรายละเอียดบางอย่างมากเกินความจำเป็นของเกม เพราะการซิมมูเลชันโดยทั่วไปจะเน้นเพื่อหาคำตอบใด ๆ ที่ต้องแม่นยำสูงก่อนนำไปสร้างระบบจริง แต่การซิมมูเลชันในลักษณะของเกมทำเพื่อความสนุกสนานของผู้ใช้งานเท่านั้น

- ออบเจ็กต์โมเดลของเกม
  - เกมออบเจ็กต์

    คือออบเจ็กต์ที่แทนสิ่งต่าง ๆในเกมเช่นตัวละครต่าง ๆ สิ่งก่อสร้าง พื้นที่ภูมิประเทศ วัตถุสิ่งของ ทรัพยากรต่าง ๆ

    แบ่งเป็นประเภทต่าง ๆ ได้ดังนี้

    1. ตัวละคร

       เกมออบเจ็กต์ที่ต้องการการบริการจากพลังธรรมชาติ
    2. ทรัพยากร

       เกมออบเจ็กต์ที่ใช้ในกิจกรรมต่าง ๆในเกม
    3. วัตถุประดับ

       เกมออบเจ็กต์ที่ทำหน้าที่ประดับไม่ต้องการการบริการจากพลังธรรมชาติ
  - พลังธรรมชาติ

    คือออบเจ็กต์ที่แทนพลังธรรมชาติเป็นผู้ควบคุมระบบประกอบด้วย

    1. กฎธรรมชาติ

       กฎเกณฑ์ต่าง ๆที่จะถูกบังคับใช้การควบคุมว่าด้วยการตัดสินเหตุการณ์ต่าง ๆ
    2. ผู้ควบคุม

       พลังหรืออำนาจที่ทำหน้าที่ควบคุมในความคิดของศาสนาต่าง ๆเช่น พระเจ้า ในศาสนาคริสต์ และ อี้จิง ในลัทธิเต๋า
  - อีเวนต์ร้องขอ

    คือออบเจ็กต์ที่แทนสิ่งที่เกมออบเจ็กต์ต้องการกระทำใน 1 ขั้นเวลาเพื่อให้เกิดผลกระทบต่อสถานะระบบ จะถูกส่งเข้าไปในลิสต์อีเวนต์ร้องขอเพื่อให้พลังธรรมชาติมาตัดสินบังคับกฎธรรมชาติ
  - อีเวนต์ตอบรับ

    คือผลตอบรับจากพลังธรรมชาติที่ส่งให้เกมออบเจ็กต์เพื่อให้คำตอบผลการร้องขอ
  - อีเวนต์อนาคต

    คือผลที่จะเกิดขึ้นในระบบในอนาคตที่ถูกเก็บไว้ก่อนเพื่อความสะดวกในการจัดการ
  - ลิสต์อีเวนต์ร้องขอ

    คือลิสต์ของอีเวนต์การร้องขอแทนเหตุการณ์ร้องขอในเวลาเดียวกัน
  - ลิสต์อีเวนอนาคต

    คือลิสต์ของอีเวนต์อนาคตแทนเหตการณ์ที่จะเกิดขึ้นในขั้นเวลาต่อไป

- โมเดลทางภูมิศาสตร์
  ออบเจ็กต์ที่แทนสภาพที่อยู่ทางภูมิศาสตร์ ของเกมออบเจ็กต์ในระบบ
- ตัวบรรจุเกมออบเจ็กต์
  ที่เก็บเกมออบเจ็กต์ทั้งหมดแทนความสามารถเข้าถึงได้อย่างไร้ขีดจำกัด (พลังธรรมชาติ ผู้เล่น)
- มุมมองสังเกตการ
  การนำเสนอสถานะของระบบในแง่ที่สนใจต่อออบเจ็กต์ผู้เล่น หรือออบเจ็กต์อื่น ๆ
- ขั้นเวลา
  หน่วยย่อยที่สุดของเวลาในระบบ
- ผู้เล่น
  ผู้ควบคุมการดำเนินไปของตัวละครและทรัพยากร ของตนที่ตนเป็นเจ้าของ

3.2 การประยุกต์ใช้ดีไซน์แพทเทิร์น

ดีไซน์แพทเทิร์นคือ การประสานงานที่ปรับแต่งค่าได้ (Paramiterize Collaboration) ซึ่งช่วยให้สามารถศึกษาประสบการณ์ ของนักออกแบบคนอื่นเคยเจอมาใช้ได้อีก (การนำการออกแบบมาใช้ใหม่) วิศวกรซอฟต์แวร์ได้ค้นพบว่างานแก้ปัญหาการออกแบบที่ตนเคยใช้มานั้นเกิดขึ้นซ้ำแล้วซ้ำเล่าในรูปแบบที่ต่างไปในรายละเอียดแต่โดยแกนกลางแล้วเป็นการแก้ปัญหาแบบเดียวกันและพบว่ามักเกิดปัญหาให้การสื่อสารและถ่ายทอดแนวคิดไปให้คนอื่นจึงนำประสบการ์ณแก้ปัญหาเหล่านั้นมาทำให้เป็นมาตรฐานกำหนดรายละเอียดและชื่อทำให้งานออกแบบในระยะเวลาต่อมาพัฒนาขึ้นอย่างมาก ในวงการเกมก็เช่นเดียวกันปัญหาการออกแบบซอฟต์แวร์เกมก็พบว่ามีลักษณะที่ซ้ำ ๆ กันมากมาย ส่วนใหญ่ซ้ำกับรูปแบบที่เกิดขึ้นในซอฟต์แวร์ทุกแบบในส่วนของรายละเอียดของแพทเทิร์นที่ถูกอ้างอิงมาใช้ประยุกต์ในโครงงานนี้มาจากอ้างอิง [1] ซึ่งผู้ใช้สามารถหารายละเอียดแพทเทิร์นที่อ้างถึงในส่วนเพิ่มเติม

3.2.1 ประโยคปัญหา

หัวข้อนี้เป็นการตีปัญหาให้ชัดเจนเพื่อจะที่จะสามารถหาจุดที่นำดีไซน์แพทเทิร์นไปประยุกต์ใช้ได้โลกที่ต้องการจำลอง ประกอบด้วย "สิ่งแวดล้อม" และ "ชีวิต" ซึ่งกระจายตัวอยู่อย่างเกลื่อนกลาด บ่อยครั้งที่สิ่งมีชีวิตเหล่านี้จะมีการเคลื่อนที่ไปในตำแหน่งที่มันมุ่งหมายจะไป แต่ในบางครั้งเจตจำนงของมันอาจถูกขัดขวางโดย สิ่งกดดัน สิ่งผลักเร้า หรือความต้องการบางอย่างที่จำเป็นกว่ามาแทรกจนต้องเปลี่ยนสภาวะของชีวิตของตนเองไป แต่ละชีวิตอาจมี ปฏิสัมพันธ์กับสิ่งแวดล้อม หรืออาจเป็นสิ่งมีชีวิตด้วยกันเองก็ได้

ระบบที่ต้องการต้องมีความสามารถในการดูแลรับผิดชอบในการแสดง มุมมองการสังเกตได้หนึ่งมุมมองสังเกตเป็นอย่างน้อย อย่างถูกต้องและต้องแสดงค่าสถานะต่างๆ ของแต่ละสมาชิกที่เราสนใจจะสังเกตได้อย่างถูกต้องด้วย ระบบจะเป็นผู้สร้างโอกาส ที่จะให้แต่ละสมาชิกทุกตัว สร้างเหตุการณ์ขึ้นมาได้อย่างเท่าเทียมกัน ในทุก ๆ เวลาควอนตัม และทำการตัดสินเหตุการณ์ที่เกิดขึ้นอย่างยุติธรรม นั่นคือแต่ละสมาชิกจะกระทำการอย่างอิสระและขนานกันได้

3.2.2 การวิเคราะห์เบื้องต้น

เราจะทำการพิจารณาปัญหาหลาย ๆ อย่างในการออกแบบระบบจำลองเหตุการณ์ดิสครีต

3.2.1.1 การทำคิวเหตุการณ์

เพื่อจะให้เหตุการณ์หลาย ๆ ชนิดบังเกิดขึ้นได้ เราต้องทำการจัดการคิวเหตุการณ์ในรูปแบบที่ง่าย ต่อการเข้าใจ และสามารถดำเนินการต่อได้

3.2.1.2 มุมมองแบบหลายอย่างหลายหน้าต่าง

การแสดงผลของทุกสมาชิกในมุมมอง เราต้องการมุมมองมากกว่าหนึ่งหน้าต่าง และในแต่ละมุมมองในแต่ละหน้าต่าง ต้องแสดงข้อมูลสถิติอย่างอย่างถูกต้องในทุกขณะ

3.2.1.3 ทางเลือกของ การกระทำ อยู่บนฐาน ของเงื่อนไข และเหตุการณ์

ชีวิตในระบบจำลองไม่ได้ต้องการเพียงแค่ รูปแบบการตอบสนองในรูปคู่ กระตุ้น-ตอบสนอง แต่มันจะเลือกการกระทำที่มันพอใจ ซึ่งเป็นสิ่งที่ดีที่สุดในขณะนั้น ขึ้นอยู่กับ สภาวะของมันเอง และสิ่งแวดล้อม

3.2.1.4 การกระทำ กระทำอย่าง อิสระต่อเงื่อนไข และเหตุการณ์

แม้จากข้อ ที่ผ่านมาบางสมาชิกจะมีพฤติกรรมบางอย่างที่ทำการกระทำอย่างอิสระต่อสภาวะในขณะนั้น และยังแตกต่างกันไปแล้วแต่รูปแบบชีวิต

3.2.1.5 การกระทำต่าง ๆ เกิดขึ้นอย่างขนานกันไปอย่างเห็นได้ชัด

การกระทำของชีวิตหนึ่งจะกระทำต่อชีวิตอื่น ดังนั้น การกระทำอยู่บนฐานของ สิ่งที่เป็นจริงจากตอนจบของเหตุการณ์สุดท้าย จนถึงเวลาที่ชีวิตนั้นได้โอกาสที่จะปฏิบัติเหตุการณ์ใหม่

3.2.2 แก้ปัญหาการออกแบบโดยประยุกต์ใช้ ดีไซน์แพทเทิร์น

3.2.2.1 การทำคิวเหตุการณ์

แกนหลักของระบบจำลองของเรา คือตัวกำเนิดเหตุการณ์ มันจะทำการสร้างกระแสของเหตุการณ์ซึ่งทำการวางเหตุการณ์ว่าจะเกิดอย่างไรและเกิดเมื่อไร แต่ละเหตุการณ์จะต้องการ การประมวลผลอย่างสลักสำคัญ ก่อนที่เหตุการณ์นั้น ๆ จะเสร็จสมบูรณ์ ดังนั้นจึงจำเป็นที่จะต้องทำคิวสำหรับเหตุการณ์ที่ถูกสร้างขึ้นเพื่อทำการประมวลผลในภายหลัง ต้องออกแบบให้คิวทำงานอย่างง่าย ๆ และไม่ขึ้นกับชนิดของเหตุการณ์ที่เรามีในคิว และเป็นเหตุการณ์ที่ธรรมดาสามัญและแยกย่อย อย่างเพียงพอที่จะทำการสร้างเหตุการณ์ใหม่ ๆ ที่จะทำการจำลองต่อไปโดยไม่ต้องทำการเปลี่ยนแปลงรหัสโปรแกรมเป็นจำนวนมาก เราสามารถแก้ปัญหานี้ด้วย คอมมานด์ แพทเทิร์น ใช้กับแต่ละรหัสระบุเป้าหมาย ดังนั้นเราสามารถนำ คอมมานด์ แพทเทิร์น ที่ห่อหุ้มคำสั่งระบุเป้าหมายมาทำการเข้าคิวเพื่อทำการดำเนินการในภายหลังได้ ดังนั้นคิวจะไม่จำเป็นต้องรู้ว่าเหตุการณ์ชนิดนั้น ๆ เป็นอย่างไร และเหตุการณ์ใหม่ ๆ ก็จะเป็นซูเปอร์ของเซตเหตุการณ์เดิม ดังนั้นคิวจะดำเนินการกับเหตุการณ์ใหม่ ๆได้คืออย่างเป็นธรรมชาติ โดยอัตโนมัติ

ตัว คอมมานด์ เองจะเป็นตัวส่งผ่านการเกิดเหตุการณ์ โดยจะหน่วงการดำเนินการเอาไว้เพื่อให้สามารถจัดการกับเหตุการณ์ได้ ตัวกำเนิดเหตุการณ์จะทำการสร้างกระแส คอมมานด์ ขึ้นมาจากสมการจำลองพฤติกรรมและเมื่อได้โอกาสจากระบบจำลองเหตุการณ์ก็จะนำ คอมมานด์ นั้นส่งผ่านเข้าไปในคิวเพื่อให้ระบบจัดการดำเนินการเหตุการณ์อีกทีหนึ่ง

3.2.2.2 มุมมองแบบหลายอย่างหลายหน้าต่าง

เพื่อให้การจำลองเป็นไปอย่างน่าสนใจ เราน่าจะสร้างให้ระบบสามารถมองดูการจำลองในขณะที่กิจกรรมต่างๆ ดำเนินไปในอาณาบริเวณของโลก และดูชีวิตทุกชีวิตในโลก โดยเราสามารถมองดูข้อมูลในรายละเอียดด้านสถานะของมันได้ ไม่ว่าจะเป็นเพียงชีวิตหนึ่ง ๆหรือดูเป็นกลุ่ม ๆก็ต้องให้ผลอย่างถูกต้องเป็นหนึ่งเดียวกันทั้งระบบแต่ละมุมมองจำเป็นที่จะต้องทำการอัพเดตเมื่อชีวิตที่เรากำลังสนใจอยู่มีการเปลี่ยนแปลง กรณีที่ต้องคำนึงเป็นพิเศษคือหากมีหลายๆชีวิตรอบๆตัวชีวิตที่เรากำลังสนใจอยู่กระทำการต่อชีวิตนั้น ทำให้ชีวิตนั้นเปลี่ยนแปลงสภาพหลายๆ อย่างในช่วงเหตุการณ์หนึ่งๆเราต้องทำการอัพเดตภาพในมุมมองของเราเมื่อเกิดการเปลี่ยนแปลงทุกอย่างกับแต่ละชีวิตแล้ว แทนที่จะทำการอัพเดตทุกๆครั้งที่ชีวิตเกิดการเปลี่ยนแปลง

สำหรับปัญหานี้เราใช้ ดีไซน์แพทเทิร์น สามอย่างประกอบกันสร้างวิธีแก้ปัญหานี้ Observer ซึ่งใช้แทนแต่ละมุมมอง ทำให้เราสามารถอินเทอร์เฟซอย่างง่าย ๆ กับ Subject ที่เรากำลังสนใจดูอยู่เพื่อทำการแจ้งให้ลิสต์ของ Observerใน Subject ทราบถึงความเปลี่ยนแปลงที่เกิดขึ้น เมื่อมีการเปลี่ยนแปลงเกิดขึ้นกับสมาชิกใด ๆในมุมมองที่เรากำลังสนใจอยู่ ดังนั้นตัวแต่ละ Observer ที่เกี่ยวข้องต้องทำการอัพเดตตัวเอง และเนื่องจากความซับซ้อนในการติดต่อระหว่างสมาชิกต่าง ๆในโลก ทำให้เราไม่ต้องการที่จะทำการอัพเดตมุมมองจนกว่าความเปลี่ยนแปลงจะเกิดขึ้นทั้งหมดเสียก่อนในเวลาควอนตัมนั้น ๆ ดีไซน์แพทเทิร์น ที่เรียกว่า Mediator สามารถถูกใช้เป็นกลวิธีในการตัดสินใจในการว่าเมื่อไรสมควรที่จะต้องทำการอัพเดต มันทำหน้าที่เป็นคนกลางในการ ให้ Subject ร้องขอการอัพเดตจากแต่ละ Observer และสุดท้ายเราต้องการ Mediator ตัวเดียาในระบบและมันต้องเป็นตัวเดียวกันอย่างทั่วถึงทั้งระบบ ดีไซน์แพทเทิร์น ที่จะนำมาช่วยคือ Singleton

3.2.2.3 ทางเลือกของ การกระทำ อยู่บนฐาน ของเงื่อนไข และเหตุการณ์

แต่ละชีวิตที่เราจำลองขึ้นควรจะทำให้มันมีจิตใจเป็นของตนเอง แต่ละชีวิตจะทำการพิจรณาว่าจะกระทำสิ่งใดโดยขึ้นกับ สภาวะส่วนตัว เป้าประสงค์ และสิ่งที่มันรับรู้จากโลกรอบ ๆ ตัวมันเราควรจะทำให้มันสามารถทำการเปลี่ยนแปลงการตัดสินใจของชีวิต มากกว่าที่จะเป็นเพียงการปฏิบัติการลำดับของโปรแกรมอย่างธรรมดา ๆ ทำให้มันสามารถกระทำการสิ่งใด ๆ โดยขึ้นอยู่กับ สภาวะ สถานะ และเงื่อนไขของตัวมันเอง ที่เป็นไปได้ในขณะนั้นๆ เราสามารถแก้ปัญหานี้โดยใช้ ดีไซน์แพทเทิร์นสองตัวประกอบกันสองชั้น ดีไซน์แพทเทิร์น ที่ใช้คือ State และ Strategy ทำให้เราสามารถทำให้เปลี่ยนแปลงเซตของหน้าที่ความสารถ โดยรักษาการมีอินเทอร์เฟซแบบเดิมไว้ โดยใน State ออบเจ็กต์นั้นประกอบด้วยฟังก์ชันโอเรียนเต็ดเพื่อพิจารณาว่าเป้าประสงค์ที่เหมาะสมที่สุดที่สุดกับสถานะนั้นๆ คืออะไร ตัวอย่างเช่นชีวิตนั้นอยู่ในสภาวะอดอยาก มันก็จะมีรูปแบบการเคลื่อนไหวในการหาอาหารที่อยู่ใกล้ตัวที่สุดแพทเทิร์นชั้นที่สองที่จะมาใช้ในการแก้ปัญหาคือ Strategy ทำให้ชีวิตสามารถตัดสินใจในภาวะที่แตกต่างเพื่อให้สามารถบรรลุเป้าประสงค์ เช่นการเดินในขณะที่บาดเจ็บก็ต้องมีความแตกต่างกับการเดินในสภาวะปกติ Strategy ที่แตกต่างกันก็เหมาะสมกับเหตุการณ์ที่แตกต่างกัน หนทางที่มันตัดสินใจเลือกทำในที่สุดขึ้นอยู่กับว่ามันรู้สึกอย่างไร และเลือกว่าจะทำอย่างไรจึงจะบรรลุเป้าหมายได้

3.2.2.4 การกระทำ กระทำอย่าง อิสระต่อเงื่อนไข และเหตุการณ์

แต่ละชีวิตจะมีจิตสำนึกของตนเองซึ่งมันจะรู้สึกว่ามันต้องกระทำให้สำเร็จไม่ว่ามันจะอยู่ในสภาวะ การณ์ เช่นไร หรือสิ่งที่อยู่รอบๆตัวกำลังดำเนินไปเช่นไร ยกตัวอย่างเช่นชีวิตที่มีนิสัยชอบความรุนแรงก็จะรู้สึกว่าต้องโจมตีสิ่งมีชีวิตทุกๆตัวที่เข้าไปใกล้มัน ก่อนที่จะทำสิ่งอื่นๆต่อไป อย่างไรก็ตามเราก็อยากจะได้รูปแบบของชีวิตที่มีรูปแบบเหมือนๆกันซึ่งไม่ได้เป็นไปเพื่อรองรับจิตสำนึกของชีวิตใดชีวิตหนึ่ง สิ่งที่เราต้องการก็คือการให้สับคลาสของรูปแบบชีวิตได้ตัดสินใจว่ากรณีที่มีการเรียกร้องให้มันเคลื่อนที่ หรือกระทำการใดๆก็ตามมันควรจะทำอะไรก่อนหน้าและหลังจากนั้น โดยไม่กระทบถึงโค้ดของ super-life โดยตรงวิธีที่จะแก้ปัญหานี้ได้ได้อย่างสมบูรณ์คือ Template Method แพทเทิร์น วิธีการตามแพทเทิร์นนี้ก็คือเมื่อเราเขียนฟังก์ชันในเบสคลาสภายในนั้นเราจะเรียกใช้ฟังก์ชันซึ่งกำหนดไว้ให้เป็นฟังก์ชันว่างๆที่ไม่ได้กระทำการอะไรเลย จากนั้นเราจะให้สับคลาสกำหนดการทำงานของฟังก์ชันว่างๆนั้นใหม่ ซึ่งเท่ากับว่าเราเพิ่มการทำงานซึ่งเป็นส่วนตัวของแต่ละชีวิตเข้าไปโดยปราศจากการเปลี่ยนแปลงข้อกำหนดของฟังก์ชันในเบสคลาส

3.2.2.5 การกระทำต่าง ๆ เกิดขึ้นอย่างขนานกันไปอย่างเห็นได้ชัด

ในแต่ละเหตุการณ์หนึ่งๆของชีวิตจะประกอบไปด้วยหลายๆขั้นตอน เราต้องการเก็บข้อมูลสถานะของชีวิตขณะแรกเข้าสู่เหตุการณ์ไว้ เนื่องจากชีวิตในแต่ละขั้นตอนจะดำเนินต่อไปโดยตัดสินใจจากสถานะก่อนเข้าสู่เหตุการณ์นั้น ไม่ใช่สถานะขณะดำเนินเหตุการณ์ ปัญหาก็คือเราไม่ต้องการให้กระทบต่อเอนแคปซูเลชัน เราไม่ต้องการจะดูข้อมูลในออบเจ็กต์ แต่เราต้องการจะเก็บข้อมูลเหล่านั้นไว้โดยไม่รู้ว่ามันคือข้อมูลอะไรเราแก้ปัญหาเหล่านี้ด้วย Memento แพทเทิร์น ด้วยวิธีนี้เราขอข้อมูลที่ออบเจ็กต์เห็นว่าจำเป็นในการกลับสู่สถานะจากออบเจ็กต์ ข้อมูลที่ออบเจ็กต์ส่งมาจะอยู่ในรูปที่ไม่มีใครรู้จักซึ่งเรามีหน้าที่เก็บไว้เท่านั้น เมื่อเหตุการณ์จบลงแล้วเราจึงคืนข้อมูลสถานะนี้ให้กับออบเจ็กต์เพื่อกลับสู่สถานะเดิม หรือเหตุการณ์อื่นเข้ามาแทรกก่อนเหตุการณ์ปัจจุบันจะจบลงเราก็จะทำการเก็บข้อมูลสถานะอีกครั้งโดยยังเก็บข้อมูลก่อนหน้าไว้ เมื่อมีเหตุการณ์หนึ่งจบลงเราจะทำการคืนสถานะที่ยังไม่ได้คืนที่เก็บไว้ล่าสุดให้แก่ออบเจ็กต์(ก็คงมองออกว่าควรใช้สแต็ก แต่จะไม่ขอเจาะจง เนื่องจากเป็นส่วนอิมพลีเมนเทชัน)

3.3 การประยุกต์ใช้ยูเอ็มแอล

ยูเอ็มแอล คือภาษาที่ใช้อธิบายสิ่งที่เราสนใจในรูปของโมเดลมุ่งหมายที่จะทำให้เกิดความเข้าใจที่ดีขึ้น ใช้ถ่ายทอดสู่ผู้อื่น และมีความเป็นรูปแบบชัดเจนเพียงพอที่จะทำการ ฟอร์เวิดและรีเวิร์ดเอนจิเนียโดยเน้นในการนำโมเดลที่ได้ไปสร้างซอฟต์แวร์

ภาษาเป็นสิ่งประดิษฐ์อันยิ่งใหญ่ของมนุษย์ มนุษย์คิดและอธิบายสิ่งต่าง ๆโดยใช้กลไกของภาษา ภาษา ยูเอ็มแอล เป็นภาษาหนึ่งของมนุษย์ที่ใช้ในการสร้างซอฟต์แวร์ ดังนั้นพื้นฐานของ ยูเอ็มแอล จึงเหมือนกับภาษาที่มนุษย์ใช้ในชีวิตประจำวัน

เนื่องจากยูเอ็มแอลมีรายละเอียดและความลึกซึ้งมากจึงไม่ขอนำรายละเอียดมาลงในที่นี้แต่จะขอนำเสนอการนำยูเอ็มแอลไปประยุกต์ใช้ในการทำโมเดลเกม ผู้ต้องการศึกษา ยูเอ็มแอล ขอแนะนำหนังสือ [7] ที่ปรากฏในหมวดอ้างอิง

3.3.1 การใช้ Use Case ไดอะแกรม เพื่อหาความต้องการของระบบ

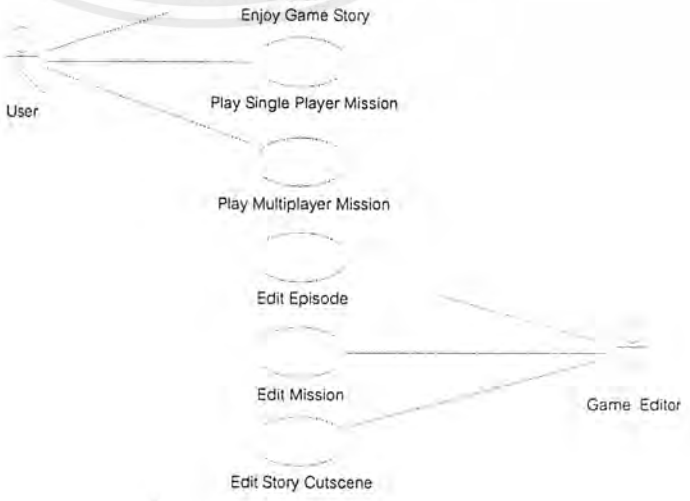
ในการพัฒนาซอฟต์แวร์ทุกชนิดสิ่งที่สำคัญเป็นอันดับต้น ๆ คือการหาความต้องการของระบบ ยูเอ็มแอล ให้ให้เครื่องมือที่มีประสิทธิภาพในการจัดการคือ Use Case ไดอะแกรม ถ้าเราต้องการนำ Use Case ไปใช้ในกระบวนการตลอดขั้นตอนแล้ว เราต้องมุ่งเน้นให้ Use Case หลัก ๆ ออกมาให้ได้เพื่อที่จะส่งต่อไปทำกระบวนการต่อ ๆ ไปได้และต้องหมั่นทบทวนแก้ไข Use Case ให้ตรงกับความต้องการที่เปลี่ยนแปลงไปด้วย

การทำโมเดลความต้องการเบื้องต้นของเกมควรเริ่มจากคุณลักษณะต่าง ๆ ของระบบหากต้องการข้ามส่วนที่เกี่ยวข้องกับการเตรียมตัวไปก่อนให้เริ่มการทำโมเดลยูสเคสจากการเล่นที่คิดว่าจะเป็นรูปแบบที่ธรรมดาที่สุดและจะถูกใช้บ่อยที่สุดเสียก่อน เช่นการเริ่มด้วย Use Case เล่นเกม โดยละการตั้งค่าอื่น ๆไว้ก่อน

ภาพ 3.3.1 ก แสดงการโมเดล Use Case โดยเน้นที่แกนกลางของเกมก่อน

โดย Use Case เล่นเกมนี้จะหมายถึงการเล่นเกมของผู้เล่นในส่วนหลักขอการทำงานของโปรแกรมจริง ๆ ซึ่งเป็นสิ่งที่จะต้องเกิดขึ้นแน่นอน

ส่วนถ้าต้องการทำ Use Case เพื่อสะท้อนความต้องการในแง่คุณลักษณะของโปรแกรมโดยรวมเราควรเริ่มต้นมีตัวอย่างดังนี้



ภาพ 3.3.1 ข แสดงการโมเดล Use Case โดยเน้นที่คุณลักษณะของโปรแกรม

เห็นได้ว่าการโมเดลในลักษณะออกมาอย่างนี้ก่อนเป็นสิ่งที่ดีเพราะเป็นการหาความต้องการโดยองค์รวมแต่แรก ๆซึ่งอาจมีผลกระทบโดยรวมต่อ Use Case เล่นเกมด้วย ทีสำคัญไปยิ่งกว่านั้นรูปแบบการเล่นนั้นอาจจะเป็นจุดชี้เป็นชี้ตายในเกมบางชนิดได้เพราะ ถ้าเราเริ่มจาก Use Case เล่นเกมอย่างไม่คำนึงถึงส่วนอื่นอาจทำให้เกิดปัญหาภายหลังได้

แต่อย่างไรก็ตามในความรู้สึกของผู้พัฒนาเกมย่อมคำนึงถึงการเล่นเกมเป็นอันดับแรกซึ่งน่าจะประนีประนอมให้ทำได้ถ้าผู้ใช้มีหลักการออกแบบส่วนหลักให้มีการเตรียมการรองรับขยับขยายส่วนอื่น ๆ อย่างเหมาะสม

หลังจากเราได้ Use Case ที่เป็นตัวตั้งต้นของเกมแล้วเราควรทำ Use Case Refinement ให้ลงไปในระดับของนามธรรมที่ต่ำลงไปเรื่อย ๆ โดยใช้ Use Case Relationship เพื่อลดความซับซ้อนของ Use Case

1. การใช้ Relationship include

ภาพ 3.3.1ค แสดงการใช้ยูสเคส รีเรชั่นชิป อินครูด

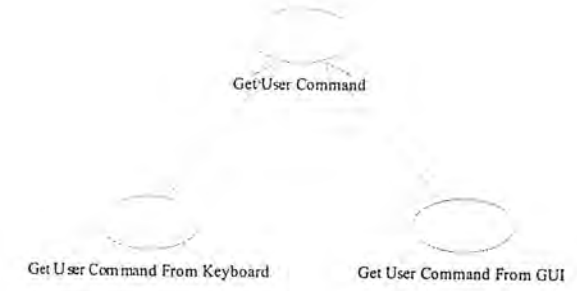
Relationship include เป็น Dependency ที่มี Stereotype แบบ include Use Case A include Use Case B หมายถึง Use Case A รวมความต้องการของ Use Case B ด้วย จากตัวอย่าง Use Case Play Episode nclude Use Case Play Game หมายความว่า การเล่นเกมแบบ Episode รวมถึงการดูภาพยนต์ด้วยอย่างแน่นอน

2. การใช้ Relationship extend

ภาพ 3.3.1 ง แสดงตัวอย่างการใช้ยูสเคสรีเรชั่นชิป เอ็กเทน

Use Case A extend from Use Case B หมายถึง การที่ Use Case B ขยายความจาก Use Case A เป็นการขยายความเฉพาะส่วน ดังตัวอย่าง การเล่นเป็นเล่นคนเดียวขยายมาจากการเล่นเกมช่วงหนึ่ง แต่แตกต่างกันในเฉพาะส่วนการเตรียมตัวผู้ร่วมเล่นเท่านั้นว่าจะเป็นอย่างไร

3. การใช้ Relationship generalize



ภาพ 3.3.1 จ แสดงตัวอย่างการใช้ ยูสเคสรีเรชั่นชิป เจเนรอลไลเซชั่น

Relationship Generalize Use Case A generalize from Use Case B หมายถึง การที่ Use Case A เป็น Use Case B ชนิดหนึ่ง ดังตัวอย่าง Use Case รับคำสั่งจากผู้ใช้ มีสองชนิดคือ การรับจากแป้นพิมพ์ และการรับผ่านกราฟิกยูสเซอร์อินเทอร์เฟส

การทำฟอร์เวิดเอนจิเนียต่อจากโมเดล Use Case ทำจากความสัมพันธ์ของ Use Case ที่ทำหน้าที่เป็นข้อกำหนดของความต้องการ ไปสู่ข้อกำหนดของการตอบสนองความต้องการ

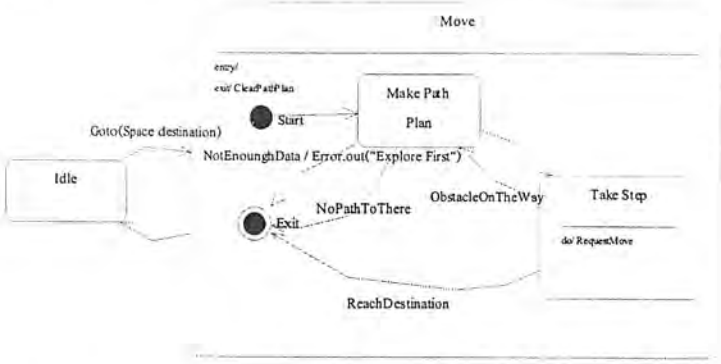
ภาพ 3.3.1 ฉ แสดงความสัมพันธ์ระหว่างยูสเคสและคอลาบอเรชัน

การตอบสนองความต้องการ Use Case เราทำโดยการให้ Collaboration Realize Use Case หมายถึงว่า Collaboration นั้นตอบสนอง Use Case นั้นจากนั้นจึงทำโมเดลต่อไปใน Collaboration ซึ่งเป็นกระบวนการที่ใช้ หาคลาสและออบเจ็กต์ที่ทำงานร่วมกันเพื่อตอบสนอง Use Case ซึ่งอาจเริ่มด้วยการ หาออบเจ็กต์จากประโยคปัญหาแล้วจึงใช้ Sequence ไดอะแกรม และ Collaboration ไดอะแกรม เข้าทดสอบและหารายละเอียด ซึ่งในกระบวนการนี้ ลักษณะของออบเจ็กต์จะชัดขึ้นและ จะพบออบเจ็กต์ต่าง ๆ ที่ไม่ได้ค้นพบแต่แรกหรืออาจ กำจัดออบเจ็กต์ที่คาดไว้แต่แรกว่าจะมีออกไปได้ ทำจนกระทั่งตอบสนองต่อ Use Case ได้สมบูรณ์ ตาม Scenario ต่างๆ ที่กำหนดไว้อย่างเพียงพอ

3.3.2 การใช้ State Chart ไดอะแกรม โมเดล State ซับซ้อนของตัวละคร

ปัญหาอย่างหนึ่งซึ่งมักพบเสมอในเกมคือการจัดการความซับซ้อนของสถานะของตัวละครซึ่งเครื่องมือทั่วไปสำหรับการโมเดลสถานะได้แก่การใช้ FSM หรือโมเดลแบบ มัว แต่อย่างไรก็ตามนักออกแบบมักพบว่า เครื่องมือเหล่านั้นขาดการจัดการด้านรายละเอียดที่พอเหมาะและขาดหลักการของการนำมาใช้ใหม่ ยูเอ็มแอล ได้นำเสนอ State Chart ไดอะแกรม ซึ่งเติมเต็มต่อความต้องการและ

สามารถนำมาเป็นเครื่องมือในกาจัดการการทำโมเดลที่มีสถานะซับซ้อนได้ดี



ภาพ 3.3.2 ก แสดงตัวอย่างการนำแผนผัง state chart ไปใช้ในการออกแบบพฤติกรรมของตัวละคร

จากภาพเป็นตัวอย่างของการนำ State Chart ไดอะแกรม มาโมเดลแทนสภาวะที่เกมออบเจ็กต์อยู่ในสภาวะเคลื่อนที่ คุณลักษณะพิเศษของ State Chart ไดอะแกรม ของ ยูเอ็มแอล ที่ช่วยให้การโมเดลสถานะของตัวละครที่ซับซ้อนง่ายมากขึ้นได้แก่

1. การมี Substate

   ทำให้มีการร่วมใช้การตอบสนองต่าง ๆ อย่างเหมาะสมซึ่งในแบบเดิม ๆ ต้องเขียนซ้ำ ๆเป็นจำนวนมากซึ่งทำให้ยุ่งยากต่อกาทำความเข้าใจ

2. การมี Advance State Transition

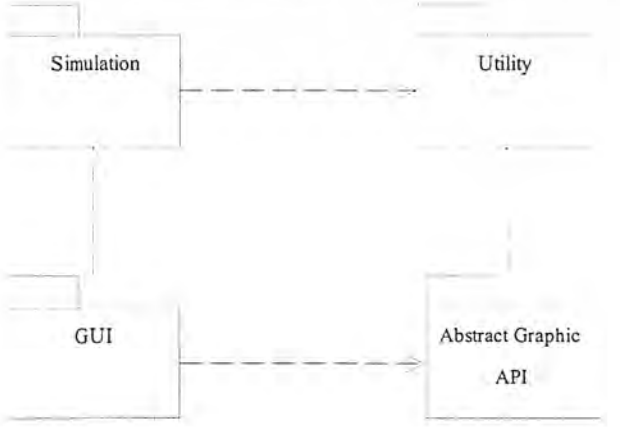
   Advance State Transition คือ Transition แบบพิเศษเช่น Enter, Exit

   ซึ่งสามารถใช้จุดนี้ในการทำ Initial, Clean up routine

   นอกจากนั้นยังมีเทคนิคการใช้งาน State Chart ไดอะแกรม อีกมากมายซึ่งแล้วแต่ความสร้าง สรรค์ของผู้นำไปประยุกต์ใช้

3.3.3 การนำ Package มาจัดการกับความซับซ้อนของโมเดล

เมื่อเราเริ่มทำโมเดลไปได้ซักระยะจะพบว่าโมเดลโดยรวมขาดความเป็นระเบียบเนื่องจากขาดการจัดกลุ่มที่ดี ยิ่งในการโมเดลเกมซึ่งมีรายละเอียดความซับซ้อนมากจึงขาดการจัดกลุ่มโมเดลไปเสียไม่ได้ ในที่นี้ขอแนะนำการจัดกลุ่มโมเดลโดยใช้ แพคเกตโดยจะแสดงรูปแบบการจัดกลุ่มในระดับบนสุดซึ่งนอกจากจะทำให้เป็นระเบียบแล้วยังทำให้มองออกว่าส่วนใดสามารถนำมาใช้ใหม่ได้ดีขึ้นอีกด้วย



ภาพ 3.3.3 ก แสดงตัวอย่างการใช้ Package ในการแบ่งแยกกลุ่มของโมเดลในเกม

จากแพคเกตต่าง ๆที่แยกโมเดลต่าง ๆ ออกจากกันอธิบายการแยกแยะดังนี้

- Package Simulation

  เก็บส่วนที่เกี่ยวกับการ ซิมมูเลชันเอาไว้ได้แก่เกมออบเจ็กต์ต่าง ๆ

- GUI

  เก็บกราฟฟิกยูสเซอร์อินเทอเฟสที่สามารถใช้ข้ามเกมได้ส่วนใหญ่

- Apstrac Graphic API

  เก็บโมเดลที่ทำหน้าที่เป็นสะพานระหว่าง API ที่มีในจริงกับลักษณะที่เราต้องการใช้เป็นแนวคิดจาก Bridge แพทเทิร์น

- Game Utillity

  เก็บโมเดลที่ทำหน้าที่อัตถประโยชน์ต่าง ๆ ในเกมที่เราต้องการใช้ในกรณีที่หาไม่ได้ในภาษาโปรแกรมหรือ API อื่นใด เช่น Container แบบเฉพาะของเกม อีเวนต์คิว

3.3.4 การใช้ Collaboration ไดอะแกรม เพื่อออกแบบสภาพการเปลี่ยนผ่านการควบคุม

การมองภาพองค์รวมของการทำงานเพื่อให้เห็นการเปลี่ยนผ่านการควบคุมนับเป็นกลไกสำคัญอย่างหนึ่งในการทำโมเดลโดยเฉพาะในเกมการมองการส่งผ่านการควบคุมเชิงโครงสร้างเป็นสิ่งจำเป็นเนื่องจากโครงสร้างของระบบประกอบด้วยสมาชิกที่มีเป็นจำนวนมากซึ่ง Collaboration ไดอะแกรม ใน ยูเอ็มแอล ตอบสนองต่อความต้องการนี้ได้เป็นอย่างดี ต่อไปเป็นการนำเสนอตัวอย่างการนำ Collaboration ไดอะแกรมไปประยุกต์ใช้ในการ โมเดลเกม

ภาพ 3.3.4 ก แสดงตัวอย่างการนำ Collaboration ไดอะแกรม ไปประยุกต์ใช้ในการทำโมเดลเกม

ในตัวอย่างเป็นการโมเดลการประสานงานกันของออบเจ็กต์ โดยเริ่มที่เกมออบเจ็กต์ที่เป็นพลทหารร้องขอการเดินโดยส่งเป็นอีเวนต์ออบเจ็กต์ไปที่ RequestEventQueue ในที่นี้สมมุติว่ามีเกมออบเจ็กต์เดียวใน

ระบบ หลังจากนั้น RequestEventQueue จึงทำการสลับ Event อย่างสุ่ม GodForce จึงนำ Message ที่ร้องขอมาประมวลผลตามกฎ แล้วจึงสร้าง FutureEvent ที่ตอบสนองผลซึ่งในที่สุดจะบังเกิดผลต่อระบบซึ่งในที่นี้คือการเปลี่ยนที่ของ พลทหาร

อ้างอิง

หลักการแนวคิดเชิงวัตถุจาก [8] Meilir Page Jones "Fundamental Object-Oriented Design in UML"

อ๊อบเจ็กต์โมเดลของ Discreate Event System Simulation จาก [3] Jerry Banks, John S. Carson II, Barry L.Nelson "Discrete-Event System Simulation"

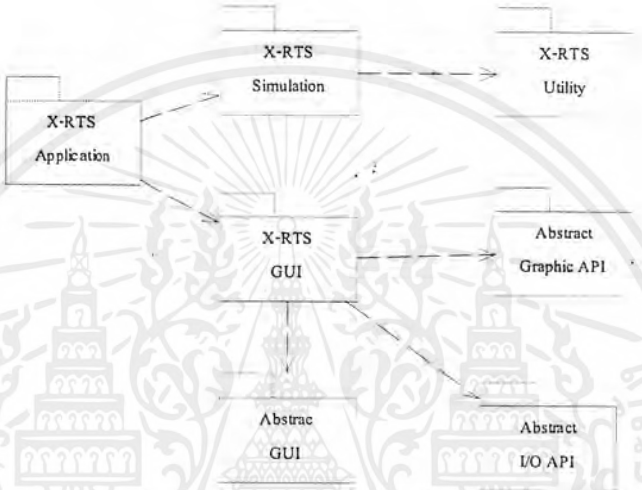
Design Pattern จาก [1] Gamma, Erich. "Design patterns elements of reusable object-oriented software /Erich Gamma ... [et al.] (Gof95)"

การประยุกต์ใช้ Design Pattern ในเกม จาก [4] David Blackledge "A Case Study: Designing a Discrete Event Simulation"

บทที่ 4
สถาปัตยกรรมของ X-RTS เกมเอนจิน

4.1 มุมมองโดยรวมเกี่ยวกับสถาปัตยกรรม X-RTS

สถาปัตยกรรมของ X-RTS เกมเอนจินนั้นมีลักษณะร่วมจากสถาปัตยกรรมทั่วไปของเกมที่ได้กล่าวมาแล้วในบทต้น ๆ ในขั้นนี้จะขอนำเสนอการจัดสถาปัตยกรรมในมุมมองโดยรวมเพื่อแยกแยะส่วนต่าง ๆออกจากกันเพื่อลดความซับซ้อนของสถาปัตยกรรม



ภาพ 4.1 ก แสดงสถาปัตยกรรมโดยรวมของ X-RTS เกม Engine

ต่อไปขออธิบายบทบาทของแต่ละแพคเกตว่ามีความหมายอย่างไรต่อสถาปัตยกรรมทั้งหมด

4.1.1 Abstract Graphic API

แพคเกตมีบทบาทเป็นชั้นพื้นฐานของการติดต่ออุปกรณ์แสดงผลเป็นการทำเอพีไอของการแสดงผลให้เป็นนามธรรมที่สามารถเชื่อมต่อกับเอพีไอในหลายระบบได้

4.1.2 Abstract I/O API

แพคเกตนี้มีบทบาทเป็นชั้นพื้นฐานของการติดต่อกับอุปกรณ์นำเข้าและส่งออกข้อมูลเช่นเม้าส์และแป้นพิมพ์

4.1.3 Abstract GUI

แพคเกตนี้ทำหน้าที่ติดต่อผู้ใช้แบบกราฟิกที่มีลักษณะไม่เจาะจงไม่ได้มีการประยุกต์ลักษณะเฉพาะด้านของเกมเข้าไว้

4.1.4 X-RTS Utility

แพคเกตนี้มีบทบาทในการช่วยเหลือการทำงานของเกมและไม่ได้เฉพาะเจาะจงกับรูปแบบของเกม

4.1.5 X-RTS Application

แพคเกตนี้มีบทบาทเป็นตัวจัดการการทำงานของเกมในแง่ของโปรแกรมเช่นการเตรียมตัวและออกจากโปรแกรมควบคุมการทำงานของโปรแกรมระดับนอกสุด

4.1.6 X-RTS Simulation

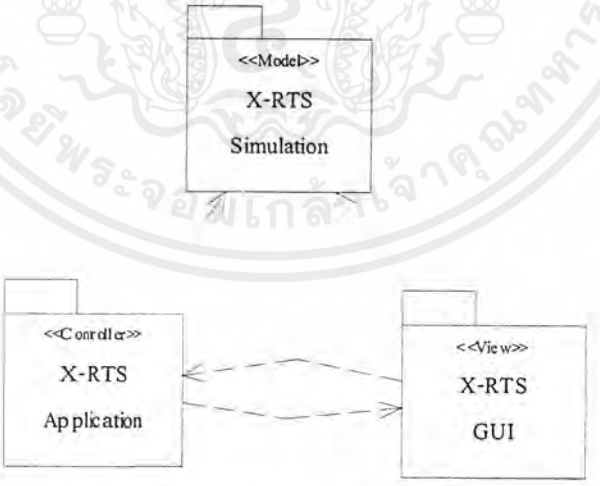
แพคเกตนี้มีบทบาทหลักในเกมคือเป็นแพคเกตที่เป็นส่วนการจำลองผลจากแนวคิดของการออกแบบเกม

4.1.7 X-RTS GUI

แพคเกตนี้มีบทบาทเป็นตัวติดต่อกับผู้ใช้ในแง่ที่เป็นลักษณะเฉพาะของเกม

หลังจากที่ได้นำเสนอบทบาทของแต่ละแพคเกตแล้ว เพื่อให้มองเห็นภาพรวมได้ชัดยิ่งขึ้น จึงขอนำเสนอความสัมพันธ์ของแพคเกตต่างๆในสถาปัตยกรรม X-RTS เกมเอนจินดังนี้

แพคเกต Abstract I/O, Abstract Graphic API, Abstract GUI เป็นส่วนพื้นฐานที่จำเป็นต่อแพคเกต X-RTS GUI ซึ่งต้องพึ่งพาการแสดงผล กลไกการจัดการวินโดว์ กลไกการติดต่อกับอุปกรณ์นำเข้า โดย X-RTS GUI จะมีลักษณะที่เฉพาะกับเกมประเภท RTS ซึ่งเป็นลักษณะที่ประยุกต์ความต้องการเฉพาะของเกมเข้ามาไว้ด้วยแล้วแพคเกตนี้ทำหน้าที่นำเสนอโลกของเกมของแพคเกต Simulation กับผู้เล่นและทำหน้าที่รับการควบคุมของผู้เล่นเข้าสู่ระบบเกมในแพคเกต Simulation ซึ่งการควบคุมจะถูกตอบสนองและประมวลผลภายในแพคเกต Simulation ส่วน X-RTS Application นั้นทำหน้าที่ควบคุมการทำงานหลักของโปรแกรมเป็นจุดเริ่มของการควบคุมการทำงานทั้งหมดของระบบและเป็นจุดสุดท้ายของการทำงานในการออกจากระบบ สถาปัตยกรรมนี้ตรงตาม MVC แพทเทิร์นเทียบได้ดังภาพ



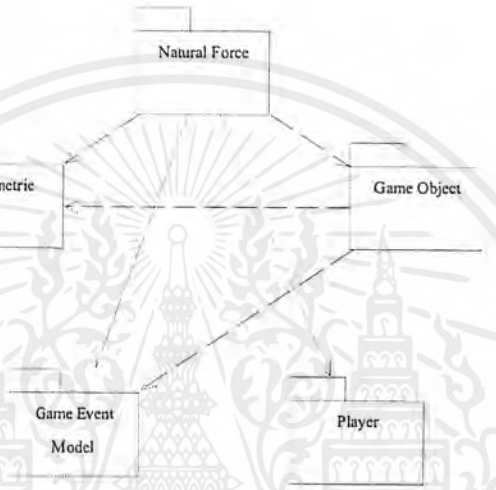
ภาพ 4.1.7ก แสดงการการประยุกต์แพทเทิร์น MVC ในสถาปัตยกรรม X-RTS เกมเอนจิน

4.2 รายละเอียดของสมาชิกในสถาปัตยกรรม X-RTS

หลังจากที่เราได้เห็นภาพรวมของระบบไปแล้วแต่อย่างไรก็ตามในแพคเกต X-RTS Simulation และ X-RTS GUI ยังมีความซับซ้อนอยู่อีกจึงของนำเสนอลงไปในรายละเอียดที่เพิ่มมากขึ้นเนื่องจากการทำงานทั้งสองเป็นหัวใจหลักของเกม

4.2.1 แพคเกตย่อยภายในแพคเกต X-RTS Simulation

ในแพคเกต X-RTS Simulation นั้นมีแพคเกตย่อยอยู่ห้าแพคเกตที่ทำหน้าที่ร่วมกันเพื่อจำลองการทำงานของแนวคิดเกมที่ได้ออกแบบไว้



ภาพ 4.2.1 ก แสดงแพคเกตย่อยของแพคเกต X-RTS Simulation

บทบาทของแพคเกตย่อยต่าง ๆ มีดังนี้

- Natural Force

  ทำหน้าที่ควบคุมกฎทางกายภาพและจินตภาพภายในเกมควบคุมการดำเนินไปของเกมทั้งหมด

- Game Object

  ทำหน้าที่แทนตัวละครต่าง ๆที่บทบาทในโลกสมมุติภายในเกมเก็บสถานะเฉพาะตัวของตัวละคร ทำหน้าที่ร้องขอเหตุการณ์ต่าง ๆไปยังพลังธรรมชาติ

- Geometric

  ทำหน้าที่แทนตำแหน่งและที่อยู่ของตัวละครค่าง ๆภายในเกมซึ่งเกี่ยวข้องกับการบังคับใช้กฎทางธรรมชาติภายในเกม และยังทำหน้าที่เป็นแบบจำลองการแสดงผลหลักที่ผู้ใช้ใช้สังเกตเป็นส่วนหลักอีกด้วย

- Game Event Model

  ทำหน้าที่แทนเหตุการณ์ที่ถูกร้องขอและเหตุการณ์ที่ปรากฏต่อโลกสมมุติภายในเกม

- Player

  มีบทบาทเป็นตัวแทนของผู้เล่นต่าง ๆ เก็บสถานะการควบคุมต่าง ๆ ทรัพยากรที่ผู้เล่นมีอยู่ นอกจากนั้นในกรณีที่เป็นผู้เล่นเทียมที่เกิดจากการจำลองของคอมพิวเตอร์ ก็รวมถึงส่วนฉลาดเทียมอีกด้วย

ความสัมพันธ์ของแต่ละสับแพคเกตภายในแพคเกต RTS – Simulation สรุปย่อ ๆได้คือส่วน Natural Force เป็นส่วนหลักที่ทำหน้าที่ควบคุมการดำเนินไปของเกมโดยอาศัยกลไกการร้องขอและตอบสนองของจาก GameObject ซึ่งการร้องขอและตอบสนองชนิดต่างๆ จะอยู่ในแพคเกต Game Event Model โดยการร้องขอและตอบสนองของ Game Object จะเกิดจากการตอบสนองของคำสั่งภายนอกจาก ผู้ใช้ไม่ว่าจะเป็นมนุษย์หรือผู้เล่นเทียมที่อยู่ในแพคเกต Player ผลลัพธ์ของการตอบสนองจะสะท้อนออกมาเป็นสถานะของระบบที่ถูกเก็บอยู่ภายในแพคเกต Geometric ซึ่งเก็บสถานะด้านพิกัดและที่อยู่ส่วนสถานะเพาะของตัวละครจะถูกเก็บไว้กับแต่ละตัวละครในแพคเกต Game Object

4.2.2 แพคเกตย่อยภายในแพคเกต X-RTS GUI

ในแพคเกต X-RTS Simulation นั้นมีแพคเกตย่อยอยู่สามแพคเกตที่ทำหน้าที่ร่วมกันเพื่อทำหน้าที่เชื่อมต่อผู้ใช้งานกับโลกสมมุติในเกม

ภาพ 4.2.2 ก แสดงแพคเกตย่อยของแพคเกต X-RTS GUI

- Game View

  ในแพคเกตนี้ประกอบด้วยมุมมองการแสดงผลต่าง ๆ ของโลกสมมุติภายในเกมนำเสนอต่อผู้ใช้ นอกจากหน้าที่ในการแสดงผลแล้วยังทำหน้าที่ตีความการควบคุมจากผู้ใช้ในระดับ GUI Action ไปสู่ความหมายที่เกี่ยวข้องกับมุมมองการแสดงผลนั้น ๆ ซึ่งส่วนใหญ่จะเป็นการส่งคำสั่งที่มีความหมายต่อโลกสมมุติภายในเกม

- Presentation Manager

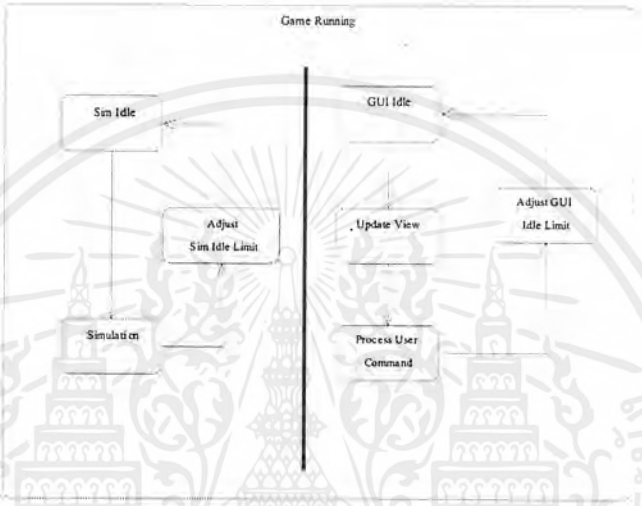
  ทำหน้าที่ควบคุมการแสดงผลตามจังหวะเวลา และการจัดการส่งคำสั่งต่างจากผู้ใช้ในระดับ GUI Action ไปสู่หน้าต่างที่ทำหน้าที่รับผิดชอบในการตีความคำสั่งนั้น ๆ

- User Command

  เป็นแพคเกตที่ประกอบด้วยคำสั่งต่าง ๆในระดับ GUI Action เช่นการกดเมาส์รูปแบบต่าง ๆ เช่นเมาส์ปุ่มขวา ปุ่มซ้าย การกดแป้นพิมพ์ปุ่มต่าง ๆ

4.3 กลไกการทำงานของ X-RTS

เกมเอนจินในระดับกลไกการควบคุม (Thred of Control) ซึ่งจำเป็นต่อการทำงานของระบบโดยรวม กลไกการควบคุมของ X-RTS เกมเอนจินประกอบด้วย 2 เธรด โดยมีจุดมุ่งหมายเพื่อให้การควบคุมและการติดต่อกับผู้ใช้เป็นไปอย่างราบรื่น และสามารถให้ผลที่เหมาะสมและคาดหมายได้บนระบบที่มีสมรรถนะต่างๆกัน โดยเป็นการประนีประนอมระหว่างการแสดงผลที่มีจำนวนการปรับปรุงภาพใหม่และการดำเนินไปของโลกสมมุติในเกมให้อยู่ในระดับที่เหมาะสมซึ่งเป็นสิ่งที่กระทบต่อความรู้สึกของผู้ใช้โดยตรง



ภาพ 4.3 ก แสดงกลไกการควบคุมของสถาปัตยกรรม X-RTS เกมเอนจิน

เธรดของ X-RTS เกมเอนจินมีอยู่ด้วยกันสองเธรดได้แก่

1. ซิมมูเลชันเธรด

   เป็นเธรดที่ทำหน้าที่ขับเคลื่อนเหตุการณ์ในโลกสมมุติให้ดำเนินไปประกอบด้วยสามสถานะย่อยได้แก่ Sim Idle เป็นสถานะว่างรอคอยการทำงานในรอบต่อไปโดยจะพิจารณาจากค่าปรับแต่งการรอคอยการซิมมูเลต เมื่อเวลาที่อยู่ในสถานะนี้ครบตามค่าปรับแต่งการรอคอยการจำลองเธรดก็เข้าสู่สถานะการทำการจำลองต่อไปสถานะนี้มีความจำเป็นเพราะทำให้ความเร็วในการทำงานของเกมไม่เร็วจนเกินไปซึ่งอาจทำให้ผู้เล่นเล่นไม่ทันได้ขั้นตอนนี้สถานะของโลกสมมุติภายในเกมจะอยู่ในภาวะมีเสถียรภาพซึ่งเป็นภาวะที่นำไปแสดงผลได้

   Simulation เป็นสถานะการดำเนินการปรับปรุงเปลี่ยนแปลงสถานะของโลกสมมุติในเกมจากขั้นเวลาปัจจุบันไปสู่ขั้นเวลาถัดไปหนึ่งขั้นเวลาขั้นนี้จะประกอบด้วยการดำเนินการคำนวณเป็นจำนวนที่ขึ้นอยู่กับจำนวนของตัวละครที่เกิดขึ้นมาภายในเกมซึ่งถ้าเมีเป็นจำนวนมากก็จะทำให้ขั้นตอนนี้ใช้เวลามากตามไปด้วย โดยตอนเริ่มเข้าสถานะนี้ตัวแปรที่ทำหน้าที่บอกสภาวะเสถียรภาพของโลกจำลองจะถูกเปลี่ยนเป็น ไร้เสถียรภาพ และจะถูกเปลี่ยนกลับเป็นมีเสถียรภาพอีกครั้งหลังจากทำการปรับปรุงสถานะทั้งหมดของโลกสมมุติในเกมแล้ว ต่อจากนั้นจะเป็นการปลุกผู้เล่นเทียมที่อาจมีในระบบขึ้นมาให้ทำงานด้วย Adjust Sim Idle Limit เป็นสถานะที่ดำเนินการปรับแต่งค่ารอคอยการจำลอง โดย

คำนวณจากเวลาที่ใช้ของสถานะ Simulation ให้สอดคล้องกับค่าที่เหมาะสมโดยค่ารอคอยการจำลองจะลดลงเมื่อสภานะ Simulation ใช้เวลาในการทำงานมากขึ้น

2. พรีเซนเทชันเธรด

เป็นเธรดที่ทำหน้าที่นำเสนอสภาวะที่มีเสถียรภาพในโลกจำลองสู่ผู้ใช้ประกอบด้วยสี่สถานะได้แก่

- GUI Idle

  เป็นสถานะรอคอยการทำงานในรอบต่อไปโดยพิจารณาจากค่าปรับแต่งรอคอยการแสดงผลค่ารอคอยการแสดงผลจะถูกจำกัดด้วยข้อจำกัดอีกอย่างคือถ้ามีการปรับปรุงภาพในจำนวนที่มากพอตามที่ตั้งไว้ในหนึ่งวินาทีสถานะการรอคอยก็จะรอตามเวลาเพราะการแสดงผลที่มากเกินไปจะทำให้สมถนะโดยรวมของระบบช้าลงไปอย่างไม่จำเป็น

- Update View

  เป็นสถานะการปรับปรุงภาพที่แสดงต่อผู้ใช้โดยสถานะนี้จะประสานจังหวะกับเธรดการจำลองโดยพิจารณาว่า โลกสมมุตินั้นอยู่ในสถานะที่มีเสถียรภาพหรือไม่ซึ่งถ้าไม่อยู่ในสถานะที่มีเสถียรภาพ ก็จะแสดงภาพเดิมไปก่อนในรอบนี้

- Process User Command

  เป็นสถานะการประมวลผลการควบคุมของผู้ใช้ไปจนเป็นคำสั่งระดับที่ใช้ในระบบโลกสมมุติของเกมขั้นตอนนี้รวมทั้งนำเอาแผนของผู้เล่นเทียมมาส่งให้กับโลกสมมุติในเกมด้วย

- Adjust GUI Idle Limit

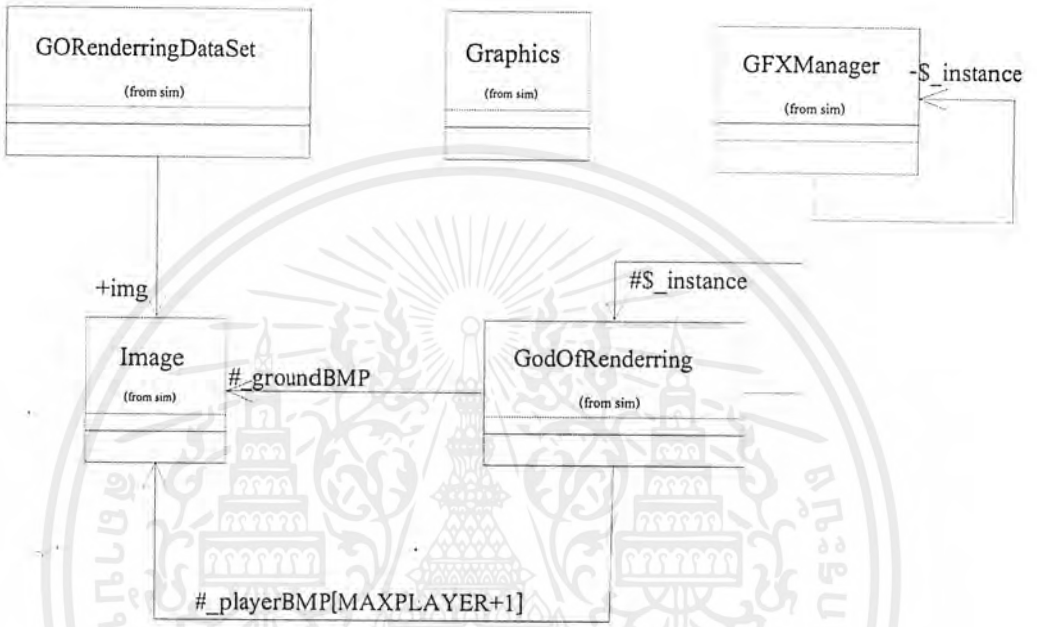
  เป็นสถานะปรับแต่งค่า รอคอยการแสดงผลโดยจะคำนวณจากเวลาที่ใช้และจำนวนครั้งที่ได้ปรับปรุงไปในหนึ่งวินาทีแล้ว

บทที่ 5

การวิเคราะห์และออกแบบเชิงวัตถุของ X-RTS เกมเอนจิน

ในบทนี้จะเป็นการนำเสนอการออกแบบเชิงวัตถุของ X-RTS เกมเอนจินโดยจะนำเสนอแบ่งตาม แพคเกต ต่าง ๆ ตามที่ได้นำเสนอไปในบทที่แล้ว

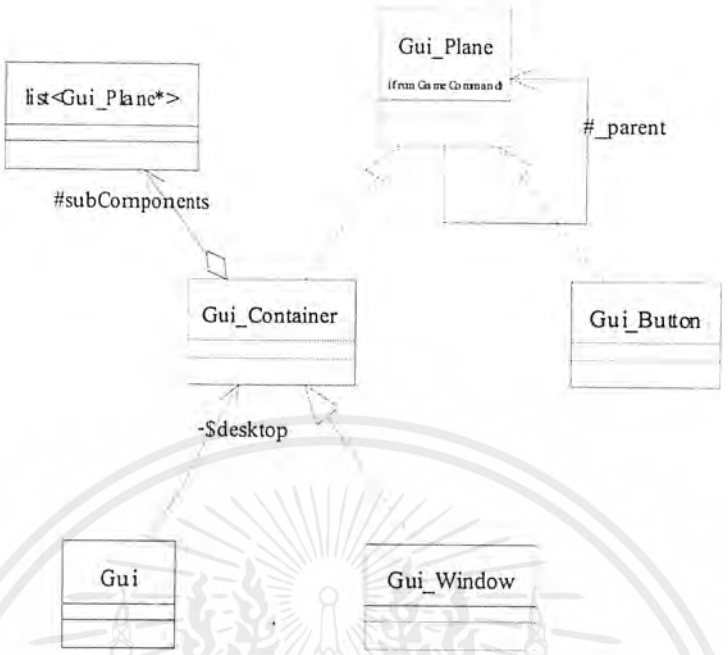
5.1 แพคเกต Abstract Graphic API



ภาพ 5.1 ก แสดงคลาสในแพคเกต Abstract Graphic API

- คลาส GFXManager
  เป็นคลาสที่ห่อหุ้มการทำงานของไดเร็กดรอว์ไว้ทำหน้าที่ในการติดต่อกับ ไดเร็กดรอว์เช่นการสร้างไดเร็กดรอว์เซอร์เฟส การแสดงไดเร็กกรอว์เซอร์เฟสออกสู่หน้าจอของผู้ใช้
- คลาส Graphics
  คือออบเจ็กต์ที่แทน กราฟฟิกซึ่งใช้เก็บภาพแบบบิตแมบจะถูกใช้โดยองค์ประกอบต่างๆที่ต้องเก็บภาพไว้เช่นคลาสต่าง ๆ ที่มีภาพในระบบกราฟฟิกยูสเซอร์อินเทอร์เฟส เป็นเหมือนพื้นที่ผืนผ้าใบที่จะถูกวาดและแสดงภาพต่างๆ บนนั้นเป็นบริเวณจะถูกนำไปแสดงบนระบบกราฟฟิกยูสเซอร์อินเทอร์เฟส
- คลาส Image
  ทำหน้าที่เป็นตัวเก็บภาพบิตแมปซึ่งสามารถทำงานได้ในแบบภาพโปร่งใสหรือไม่โปร่งใสก็ได้ ถูกใช้ในการแสดงผลภาพในส่วนต่าง ๆ เช่น ภาพตัวละครในเกมที่ปรากฏบนเกมวิว และภาพของปุ่มบนกราฟฟิกยูสเซอร์อินเทอร์เฟส ซึ่ง Image จะถูกทำซ้ำลงบนลง Graphic ในการแสดงผล

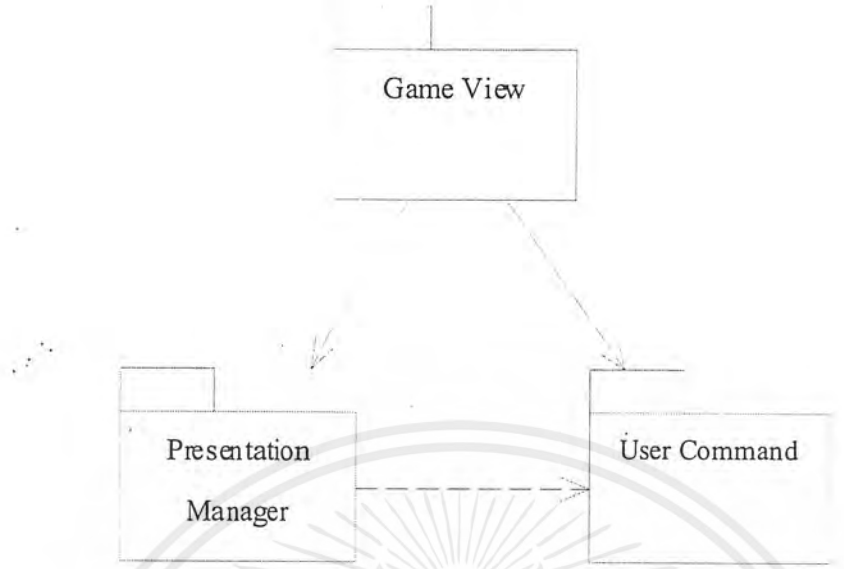
5.2 แพคเกต Abstract GUI



ภาพ 5.2 ก แสดงคลาสที่เป็นสมาชิกในแพคเกต Abstract GUI

- คลาส Gui_Plane
  เป็นคลาสฐานของสิ่งที่ปรากฏบนระบบกราฟฟิกยูสเซอร์อินเทอร์เฟสมีความหมายคือเป็นบริเวณสี่เหลี่ยมที่มีภาพและถูกแสดงบนระบบกราฟฟิกยูสเซอร์อินเทอร์เฟสที่สามารถรับและตอบสนองต่ออุปกรณ์อินพุตที่เป็นเมาส์และส่งต่อความรับผิดชอบได้ได้ มีบทบาทเป็นHandle ตามดีไซน์แพทเทิร์นแบบ Chain of Responsibility
- คลาส Gui_Button
  เป็นคลาสที่แทนปุ่มกดบนระบบกราฟฟิกยูสเซอร์อินเทอร์เฟสสืบทอดมาจากGui_Plane โดยมีการเพิ่มส่วนที่ใช้แสดงตัวอักษรที่เป็นคำอธิบายบนปุ่มไว้
- คลาส Gui_Container
  มีบทบาทเป็น Composite ในดีไซน์แพทเทิร์นแบบ Composite มีการอ้างอิงถึงคลาสตัวเอง และมีลิสต์ของตัวชี้ไปที่ Gui_Plane
- คลาส Gui_Window
  เป็น Gui_Container ที่มีขอบและกรอบแสดงชื่อในแบบวินโดว์ทั่วไปซึ่งมักไม่ได้ถูกใช้ในเกม
- คลาส Gui
  ทำหน้าที่เป็นผู้ควบคุมการแสดงผลของระบบกราฟฟิกยูสเซอร์อินเทอร์เฟสโดยดูแลและจัดการวินโดว์ที่ปรากฏในระบบกราฟฟิกยูเซอร์อินเทอร์เฟสโดยจะได้รับเทรดในการแสดงผลของตัวเอง Gui จะเก็บวินโดว์ต่าง ๆ โดยมี Gui_Container เก็บวินโดว์ต่าง ๆไว้มีชื่อเรียกว่า desktop

5.3 แพคเกต X-RTS GUI



ภาพ 5.3 ก แสดงแพคเกตต่าง ๆ ในแพคเกต X-RTS GUI

5.3.1 แพคเกต Game View

ภาพ 5.3.1 ก แสดงคลาสและความสัมพันธ์ในแพคเกต GameView และคลาสที่เกี่ยวข้องอื่น

- คลาส MapViewver

  เป็นคลาสที่สืบทอดมาจาก Gui_Container โดยเป็นมุมมองเฉพาะที่แสดงสภาพของโลกในเกมในเชิงภูมิศาสตร์ในบริเวณที่สนใจ โดยจะใช้โมเดลทางภูมิศาสตร์เป็นแบบในการแสดง นอกจากการแสดงผลแล้วคลาสนี้ยังทำหน้าที่ตีความการกดปุ่มบนเมาส์ว่ามีความหมายอย่างไรและจะเกิดผลอย่างไรโดยจะทำงานร่วมกับคลาส ActionInterpreter แล้วจะตีความการกดเป็นความหมายต่าง ๆ ในการควบคุม

- คลาส MiniMapViewer

  เป็นคลาสที่สืบทอดมาจากคลาส Gui_Plane โดยจะทำหน้าที่ในการแสดงภาพรวมในของสภาพภูมิศาสตร์ในโลกสมมุติภายในเกมโดยทั้งหมด โดยแสดงเป็นภาพเล็ก ๆ แต่ครอบคลุมพื้นที่ทั้งหมด ซึ่งสามารถใช้ในการช่วยเหลือผู้ใช้ในการหาพื้นที่บริเวณที่ต้องการซึ่งจะมีผลต่อมุมมองปัจจุบันของ MapViewer ด้วย

- คลาส Fansy_Window

  เป็นคลาสที่สืบทอดมาจากคลาส Gui_Window โดยจะมีการเก็บภาพซึ่งใช้เป็นพื้นหลังในการแสดงของวินโดว์นั้น

- คลาส Fansy_Button

  เป็นคลาสที่สืบทอดมากจากคลาส Gui_Button โดยจะมีการเก็บภาพซึ่งใช้เป็นพื้นหลังของการแสดงภาพปุ่มทำนองเดียวกันกับ Fansy_Window

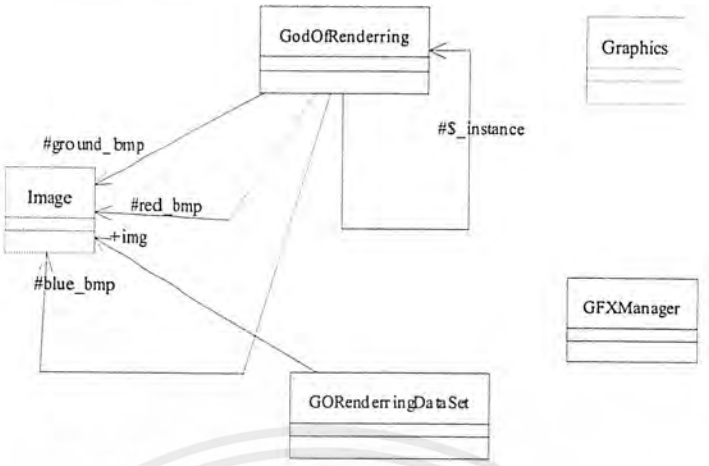
- คลาส GameCommandButton

  เป็นปุ่มที่ใช้กดในการสั่งงานตัวละครสืบทอดมากจาก Fansy_Button

- คลาส CmdButtonBar

  เป็นคลาสที่สืบทอดมาจากคลาส Fansy_Window โดยจะทำหน้าที่ในการเป็นตัวเก็บและประสานงานชุดของปุ่มกด ที่เป็นแถบเพื่อการควบคุมตัวละครโดยจะผูกพันกับตัวละครที่กำลังสนใจในปัจจุบันของผู้เล่นโดยจะแสดงคำสั่งต่าง ๆ ที่เป็นไปได้สำหรับตัวละครนั้น ๆ ทำงานกับคลาส ActionInterpreter ในการตีความหมายและส่งคำสั่งเพื่อไปควบคุมตัวละครนั้นๆให้ตอบสนองต่อคำสั่งให้ทำตามความต้องการของผู้ใช้

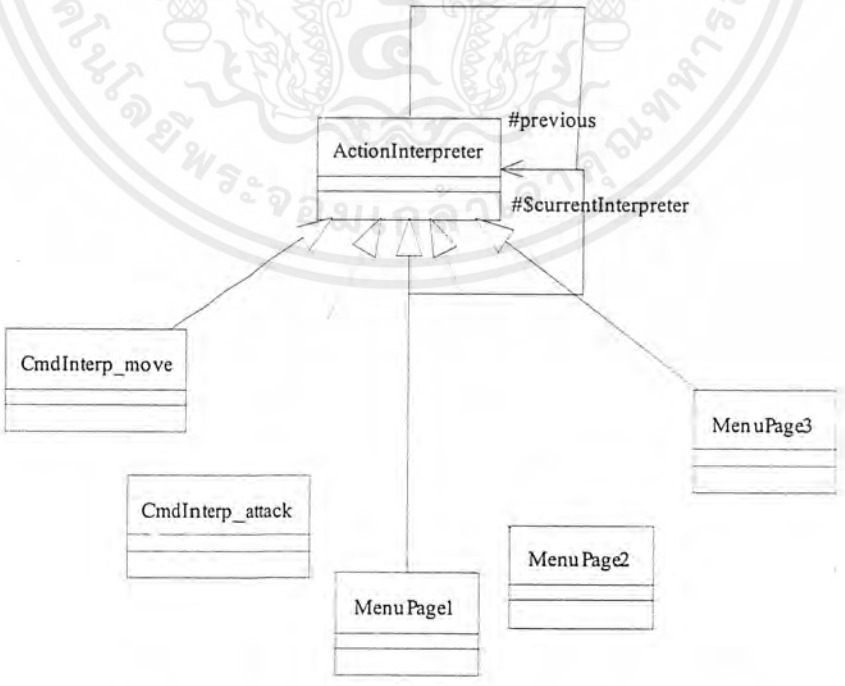
5.3.2 แพคเกต Presentation Manager



ภาพ 5.3.2 ก แสดงคลาสที่เป็นสมาชิกของแพคเกต Presentation Manager และคลาสอื่นที่เกี่ยวข้อง

- คลาส GORenderingDataSet
  ไว้เก็บคู่ของค่าสถานะและภาพของเกมออบเจ็กต์เพื่อใช้ในการแสดงผล
- คลาส GodOfRenderring
  เป็นคลาสที่ทำหน้าที่ในการรับผิดชอบการระบุว่าเกมออบเจ็กต์ใดๆในสถานะหนึ่ง จะมีภาพในการแสดงผลเป็นอย่างไรโดยถูกทำงานประสานกับ MapViewer ในการแสดงผลในคลาสนี้จะมีการเก็บคู่ของค่าที่แทนสถานะและภาพของเกมออบเจ็กต์ในสถานะนั้น ๆ เอาไว้ (GORenderingDataSet)

5.3.3 แพคเกต User Command



5.3.3 ก ภาพแสดงคลาสที่เป็นสมาชิกของแพคเกต User Command

- คลาส ActionInterpreter

  เป็นคลาสฐานนามธรรมที่ทำหน้าที่ในการแปรความหมายของคำสั่งของผู้ใช้ในการกดเม้าส์ให้มีผลต่อระบบโลกสมมุติภายในเกม

- คลาส CmdInterp_move

  เป็นคลาสที่สืบทอดมาจากคลาส ActionInterpreter ทำหน้าที่ในการตีความการกระทำของผู้เล่นในกรณีที่ผู้เล่นมีแนวโน้มจะสั่งให้ตัวละครเคลื่อนที่

- คลาส CmdInterp_attack

  เป็นคลาสที่สืบทอดมาจากคลาส ActionInterpreter ทำหน้าที่ในการตีความการกระทำของผู้เล่นในกรณีที่ผู้เล่นมีแนวโน้มจะสั่งให้ตัวละครโจมตี

- คลาส MenuPage1

  เป็นคลาสที่สืบทอดมาจากคลาส ActionInterpreter ทำหน้าที่ในการตีความการกระทำของผู้เล่นในกรณีที่โอกาสในการที่ผู้เล่นยังไม่สร้างแนวโน้มว่าจะสั่งการใดแก่ตัวละคร และกรณีที่ยังไม่มีตัวละครใดถูกเลือก

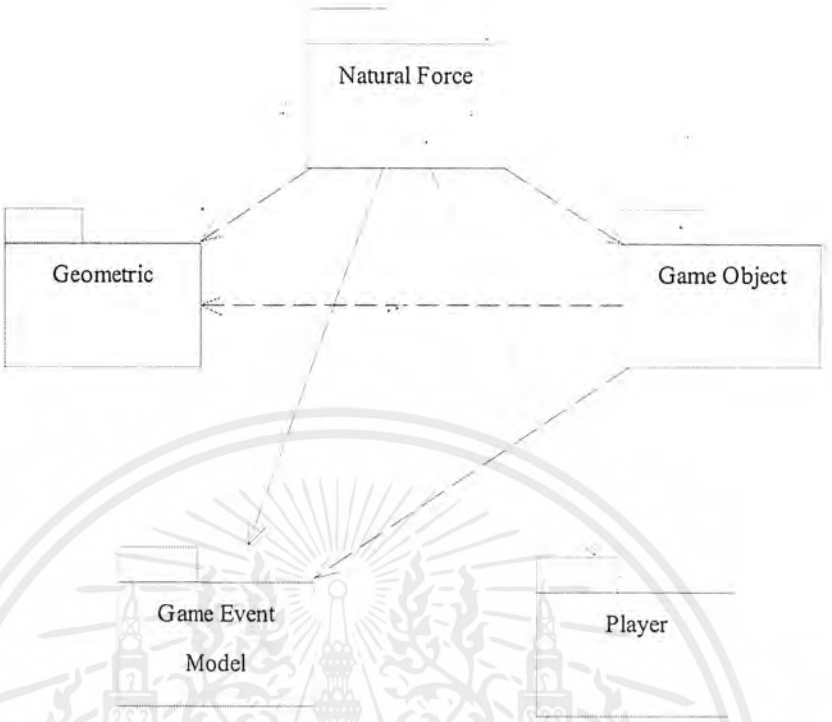
- คลาส MenuPage2

  เป็นคลาสที่สืบทอดมาจากคลาส ActionInterpreter ทำหน้าที่ในการตีความการกระทำของผู้เล่นในกรณีที่ผู้เล่นแสดงเจตจำนงที่จะสร้างสิ่งก่อสร้างขั้นพื้นฐาน

- คลาส MenuPage3

  เป็นคลาสที่สืบทอดมาจากคลาส ActionInterpreter ทำหน้าที่ในการตีความการกระทำของผู้เล่นในกรณีที่ผู้เล่นแสดงเจตจำนงที่จะสร้างสิ่งก่อสร้างขั้นสูง

5.4 แพคเกต X-RTS Simulation



ภาพ 5.4 ก แสดงแพคเกตย่อยภายในแพคเกต X-RTS Simulation

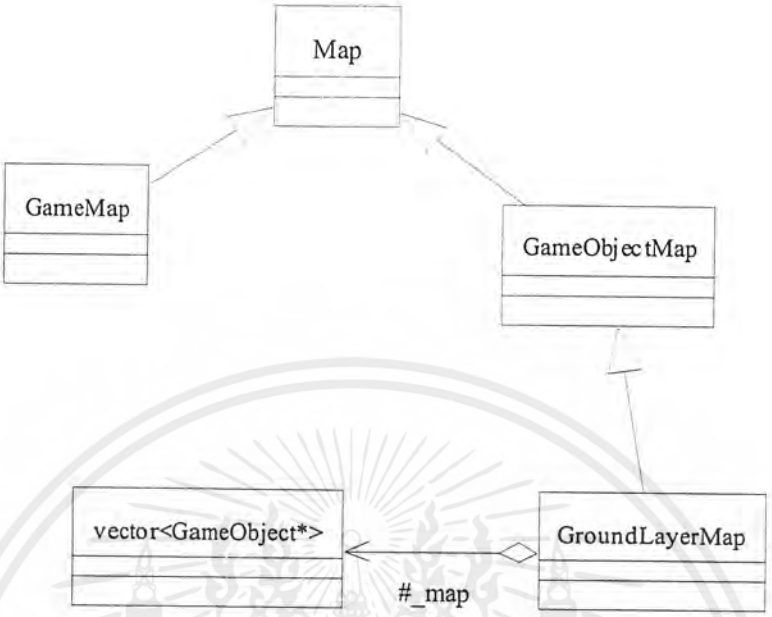
5.4.1 แพคเกต Natural Force

ภาพ 5.4.1 ก แสดงคลาสที่เป็นสมาชิกของแพคเกต Natural Force

- คลาส God

  ทำหน้าที่ในการควบคุมโลกสมมุติในเกมตั้งแต่การสร้างการควบคุมและการทำลาย โดยจะทำการสร้างเทรดเมื่อมันได้ควบคุมการดำเนินของโลกสมมุติ

5.4.2 แพคเกต Geo Model



ภาพ 5.4.2 ก แสดงคลาสที่เป็นสมาชิกของแพคเกต Geo Model

- คลาส Map

  เป็นคลาสฐานของคลาสที่ทำหน้าที่ในการเก็บแบบสภาพที่อยู่และตำแหน่งไม่ได้มีการสร้างเพื่อใช้จริงประกอบด้วยลักษณะพื้นฐานเท่านั้น

- คลาส GameMap

  เป็นคลาสที่สืบทอดมาจากคลาส Map เป็นการเก็บลักษณะของสภาพพื้นดินมีการกำหนดความสามารถในการเคลื่อนที่ผ่านได้และยังสามารถนำเข้ามาจากไฟล์

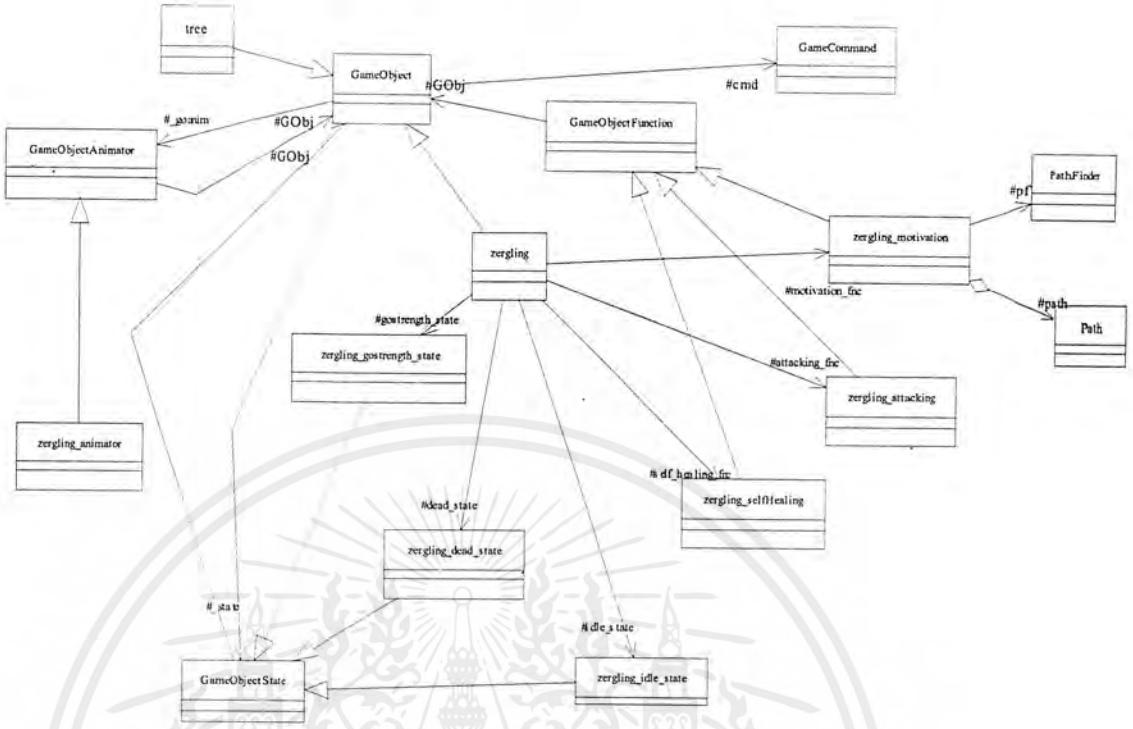
- คลาส GameObjectMap

  เป็นคลาสที่สืบทอดมาจากคลาส Map เป็นคลาสนามธรรม ที่เป็นการเก็บลักษณะของเกมออบเจ็กต์ที่อยู่บนพื้นดินสามารถใช้ตรวจสอบความสามารถในการอยู่ของออบเจ็กต์ที่ต้องการตั้งอยู่ ณ ที่นั้น ๆ ว่าทำได้หรือไม่

- คลาส GroundLayerMap

  เป็นคลาสทีสืบทอดมาจากคลาส GameObjectMap และนิยามโอเปอเรชั่นต่าง ๆ ที่นิยามไว้ใน GameObjectMap

5.4.3 แพคเกต Game Object



ภาพ 5.4.3 ก ที่เป็นสมาชิกของแพคเกต Game Object

- คลาส GameObject

  เป็นคลาสฐานนามธรรมของตัวละครทุกตัวในระบบ
- คลาส Zergling

  เป็นคลาสตัวละครตัวอย่างมีความสามารถในการฟื้นพลังชีวิตตัวเองได้ โจมตีได้ เคลื่อนที่ได้
- คลาส Tree

  เป็นคลาสตัวละครตัวอย่างไม่สามารถเคลื่อนที่ได้ เป็นตัวละครประกอบฉากอยู่ในกลุ่มธรรมชาติเท่านั้น ไม่สามารถเป็นศัตรูกับฝ่ายใดๆ
- คลาส House1

  เป็นคลาสตัวละครตัวอย่างที่อยู่ในรูปของสิ่งก่อสร้างไม่สามารถเคลื่อนที่ได้ แต่เป็นของฝ่ายใดฝ่ายหนึ่ง
- คลาส GameObjectState

  เป็นคลาสฐานนามธรรมที่แทนการตอบสนองต่อการกระตุ้นจากภายนอกในสถานะต่าง ๆว่าจะทำในลักษณะใดซึ่งใน GameObject หนึ่งจะมีเพียง GameObjectState เดียวเท่านั้นที่กำลังมีผลในการตอบสนองอยู่

- คลาส Zergling_gostrength_state

  เป็นคลาส แทนการตอบสนองต่อการกระตุ้นจากภายนอกในสภานะการเพิ่มพลังของตัวละครตัวอย่าง Zergling

- คลาส Zergling_dead_state

  เป็นคลาส แทนการตอบสนองต่อการกระตุ้นจากภายนอกในสภานะการตายของตัวละครตัวอย่าง Zergling

- คลาส Zergling_idle_state

  เป็นคลาส แทนการตอบสนองต่อการกระตุ้นจากภายนอกในสภานะการว่างของตัวละครตัวอย่าง Zergling

- คลาส GameObject Function

  เป็นคลาสฐานนามธรรมที่แทนความสามารถต่าง ๆของตัวละครโดยสามารถกำหนดให้ถูกปรับแต่งให้เปิดปิดการทำงาน หรือทำงานตามเงื่อนไขต่าง ๆได้ตามแต่สถานะต่าง ๆของตัวละคร

- คลาส Zergling_motivation

  เป็นคลาสที่แทนความสามารถในการเคลื่อนที่ของตัวละครโดยจะมี ออบเจ็กต์ PathFinder เป็นผู้หาเส้นทางการเคลื่อนที่ไปเก็บไว้ในออบเจ็กต์ Path ซึ่งเปรียบได้กับแผนในการเดินทางไปให้ถึงจุดหมาย Zergling

- คลาส Zergling_attacking

  เป็นคลาสที่แทนความสามารถในการโจมตีของตัวละคร Zergling

- คลาส Zergling_selfHealing

  เป็นคลาสที่แทนความสามารถในการฟื้นพลังตัวเองของตัวละคร Zergling

- คลาส GameObject Animator

  เป็นคลาสฐานนามธรรมที่แทนลักษณะท่าทางของตัวละครในเวลาใดเวลาหนึ่งโดยจะทำการพิจารณาจากสถานะของตัวละคร

- คลาส Zergling_animator

  เป็นคลาสที่แทนลักษณะท่าทางของตัวละคร Zergling

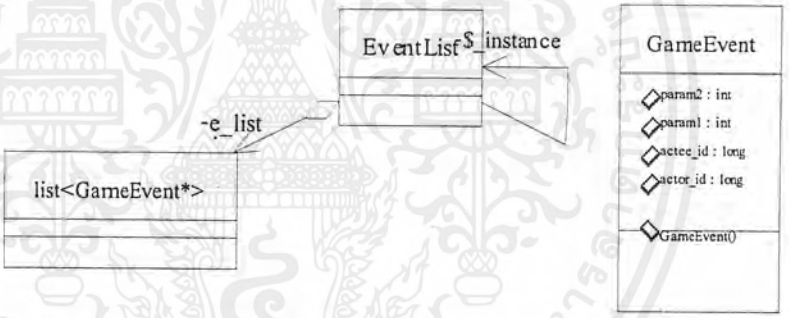
5.4.4 แพคเกต Player

ภาพ 5.4.4ก แสดงคลาสที่เป็นสมาชิกในแพคเกต Player

- คลาส Player

  เป็นคลาสที่แทนผู้เล่นมีการกำหนดความสัมพันธ์ต่าง ๆ ของผู้เล่นไว้แต่ในชั้นนี้ยังไม่ได้นิยามไว้ชัดเจน

5.4.5 แพคเกต Game Event Model



ภาพ 5.4.5 ก แสดงคลาสที่เป็นสมาชิกของแพคเกต Game Event Model

- คลาส GameEvent

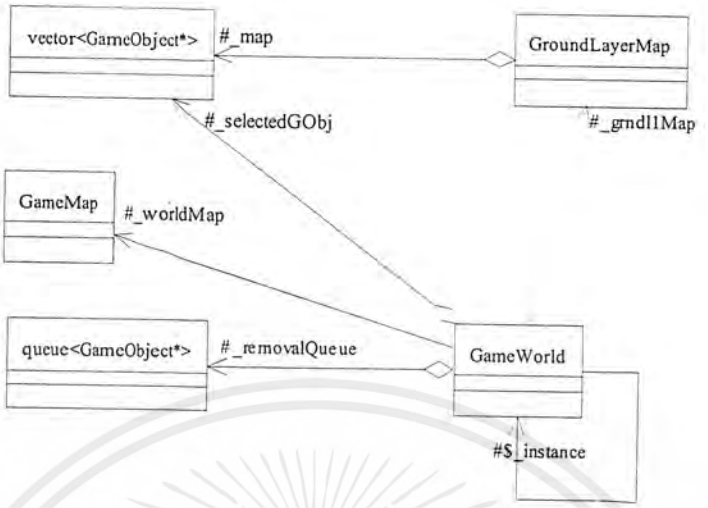
  เป็นคลาสที่เก็บคำสั่งร้องขอของตัวละครหนึ่งในการต้องการกระทำการใดใดโดยจะได้รับโอกาสในทุกรอบของขั้นเวลาเพื่อให้ God เป็นผู้ตัดสินเหตุการณ์นั้นๆว่าจะบังเกิดผลเช่นไรต่อสถานะของระบบ

- คลาส EventList

  เป็นคลาสที่เก็บคำร้องของตัวละครทั้งหมดในหนึ่งขั้นเวลาโดยจะถูกพิจารณาว่าเป็นเหตุการณ์ที่เกิดขึ้นพร้อม ๆ กันแต่จะถูกดำเนินการโดยลำดับโดยมีการประนีประนอมความความถูกต้องและยุติธรรมในระดับหนึ่ง

5.5 การทำงานในภาพรวม

5.5.1 ความสัมพันธ์ที่สำคัญของคลาสภายในเกม



ภาพ 5.5.1 ก แสดงความสัมพันธ์ของโลกในเกม

เพื่อให้สมาชิกต่าง ๆในระบบสามารถเข้าถึงคลาสที่ทำหน้าที่หลัก ๆ ได้อย่างทั่วถึงจึงกำหนดให้มีออบเจ็กต์

GameWorld ที่เป็น Singleton ทำหน้าที่เป็นตัวกลางที่เก็บคลาสหลัก ๆ เช่น GameMap, queue <GameObject*> (คอนเทนเนอร์เก็บเกมออบเจ็กต์)

หลังจากที่ได้นำเสนอคลาสต่าง ๆ ในแง่ความหมายและหน้าที่ในแง่ของมุมมองนิ่งไปแล้วเพื่อให้เราสามารถเข้าใจบทบาทหน้าที่การทำงานที่ประสานกันจึงขอนำเสนอเหตุการณ์จำลอง

5.5.2 สถานการณ์จำลอง

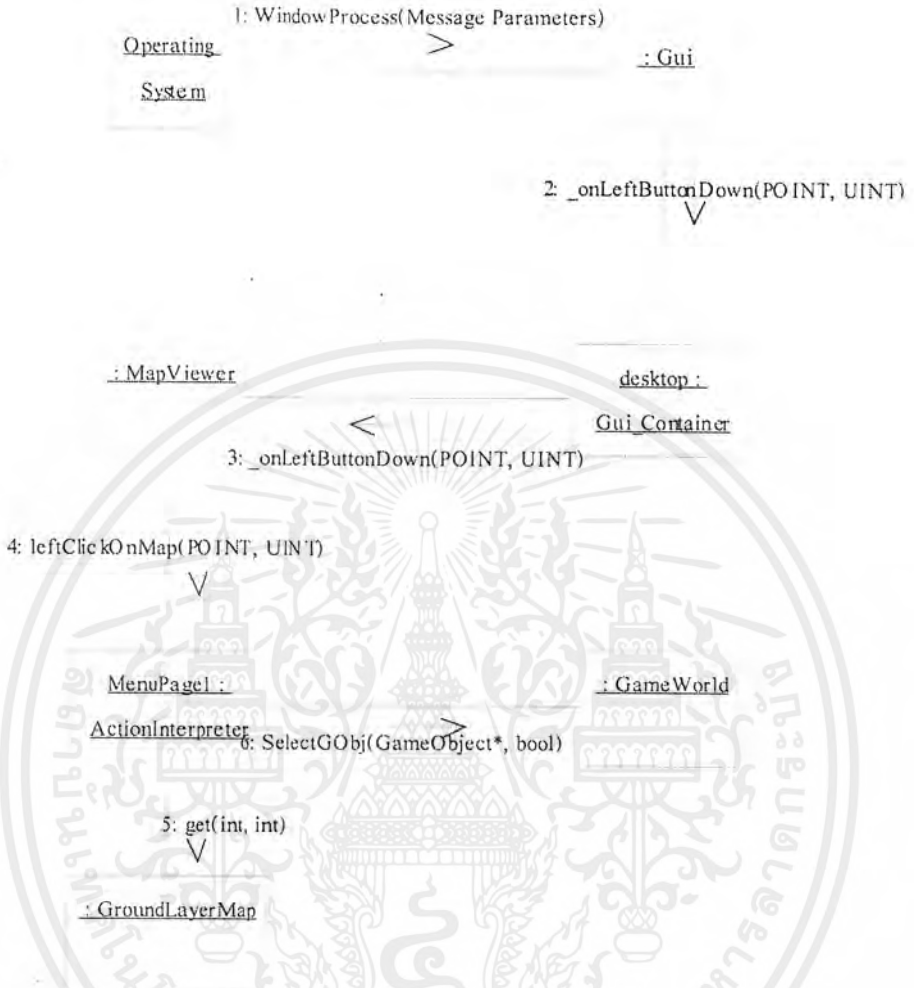
ผู้ใช้เลือกตัวละครเพื่อสั่งให้เคลื่อนที่ไปยังเป้าหมาย

ประกอบด้วยสถานะการทำงานได้ 3 ขั้นใหญ่ๆ ด้วยกันได้แก่

1. ผู้เล่นทำการคลิ๊กเม้าส์ปุ่มซ้ายเลือกตัวละครเมื่อตัวละครอยู่ในสภาพพร้อมรับคำสั่ง (มีกรอบสี่เหลี่ยม ล้อมรอบตัวละครนั้น)
2. ผู้เล่นจึงทำการสั่งให้เคลื่อนที่ไปที่เป้าหมายโดยการใช้เม้าส์ปุ่มขวาเลือกจุดเป้าหมายของการเคลื่อนที่
3. ตัวละครตอบสนองโดยการเคลื่อนที่เข้าไปที่เป้าหมายตามที่สั่ง

ในหน้าถัดไปจากนี้จะเป็นแผนผังคอลาบอเรชันและคำอธิบายประกอบแสดงการทำงานของของระบบในสถานะทั้งสามดังที่ได้กล่าวมา

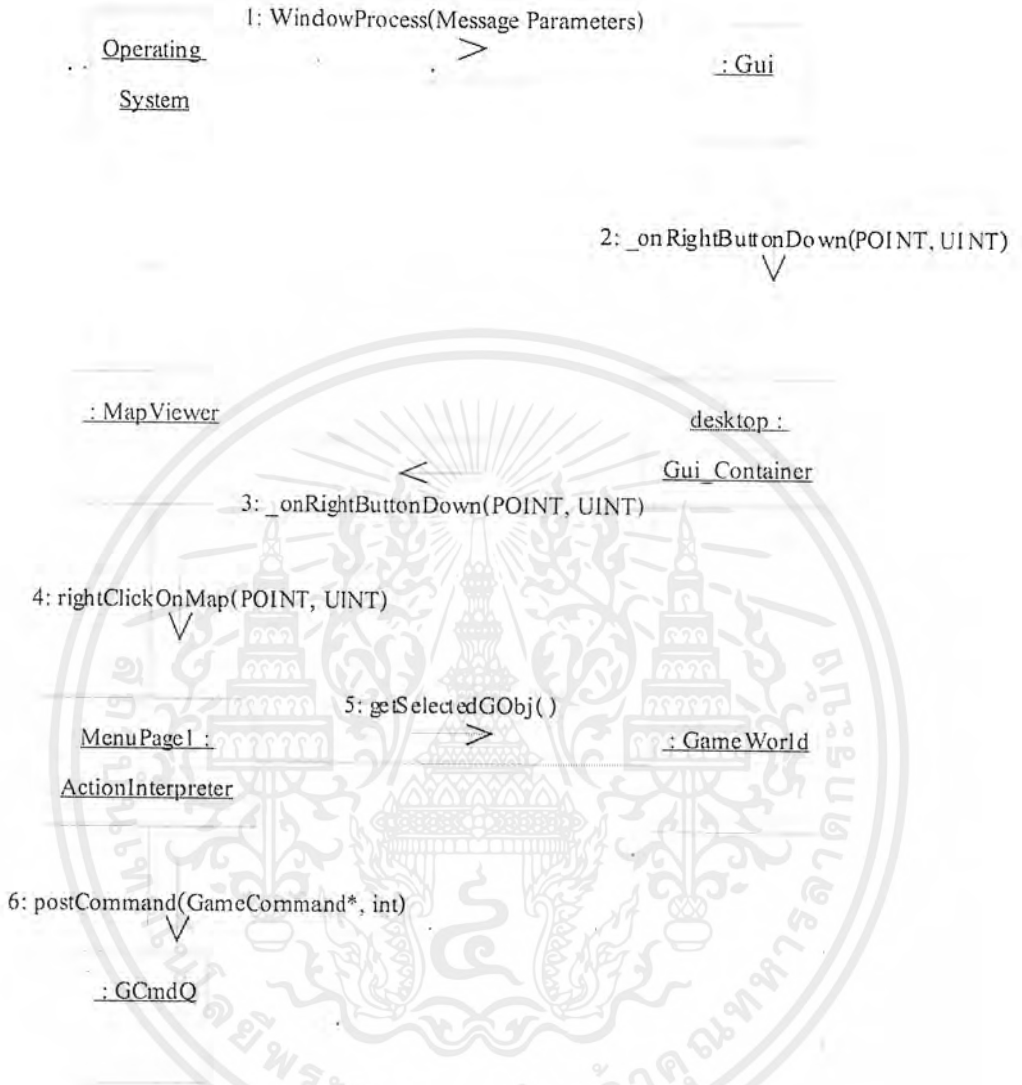
ขั้นที่ 1 การประสานงานของระบบเมื่อผู้ใช้เลือกตัวละคร



ภาพ 5.5.2 ก แสดงการประสานของระบบในการตอบสนองต่อการเลือกตัวละคร

1. เมื่อผู้เล่นคลิ๊กเมาส์ปุ่มซ้ายระบบปฏิบัติการจะส่งเมสเสจไปให้ GUI โดยเรียกฟังก์ชัน WindowProces
2. Gui ทำการตีความเมสเสจที่ได้รับ ซึ่ง GUI ตกลงที่จะส่งผ่านเมสเสจไปยัง desktop
3. เมสเสจจาก desktop ถูกส่งลงมาตามลำดับชั้นของคอนเทนเนอร์จนมาถึง MapViewer
4. MapViewer บอกกับ MenuPage1 ซึ่งเป็น ActionInterpreter ในขณะนั้นว่ามีการคลิ๊กเมาส์ซ้ายลงบน MapViewer
5. MenuPage1 ตีความว่าผู้เล่นต้องการเลือกตัวละครจึงถามจาก GroundLayerMap ว่า ณ ตำแหน่งที่คลิ๊กมีตัวละครใดอยู่
6. จากนั้นMenuPage1 จะบอกให้ GameWorld เก็บการอ้างถึงตัวละครที่ถูกเลือกไว้

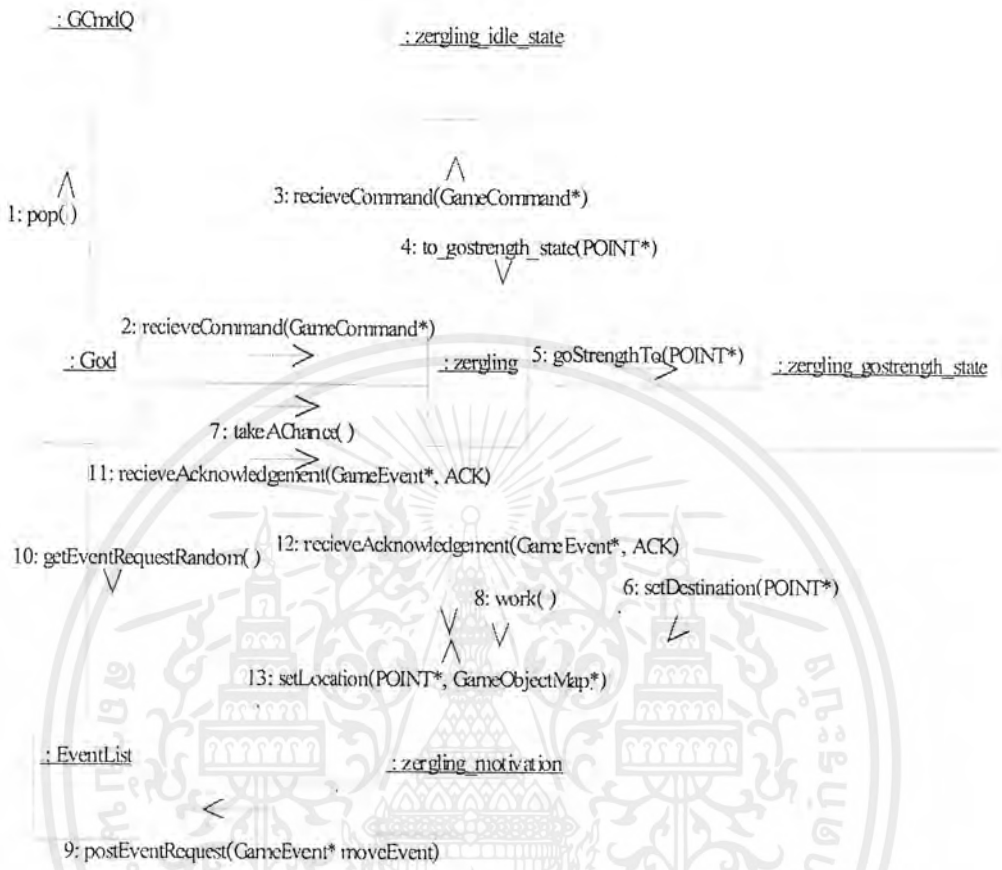
ขั้นที่ 2 การประสานงานของระบบเมื่อผู้ใช้เลือกเป้าหมายการเคลื่อนที่



ภาพ 5.5.2ข แสดงการประสานของระบบในการตอบสนองต่อการสั่งตัวละครให้เคลื่อนที่

1. เมื่อผู้เล่นคลิ๊กเมาส์ปุ่มขวาระบบปฏิบัติการจะส่งเมสเสจไปให้ GUI โดยเรียกฟังก์ชัน WindowProces
2. Gui ทำการตีความเมสเสจที่ได้รับ ซึ่ง GUI ตกลงที่จะส่งผ่านเมสเสจไปยัง desktop
3. เมสเสจจาก desktop ถูกส่งลงมาตามลำดับชั้นของคอนเทนเนอร์จนมาถึง MapViewer
4. MapViewer บอกกับ MenuPage1 ซึ่งเป็น ActionInterpreter ในขณะนั้นว่ามีการคลิ๊กเมาส์ขวาลงบน MapViewer
5. MenuPage1 ตีความว่าผู้เล่นต้องการสั่งให้ตัวละครที่ถูกเลือกอยู่เคลื่อนที่ MenuPage1 จะขอตัวอ้างอิงตัวละครที่ถูกเลือกอยู่จาก GameWorld
6. MenuPage1 สร้าง GameCommand ออบเจ็กต์ที่ระบุคำสั่ง move แล้วโพสท์ไว้ในคิวของคำสั่ง (GCmdQ)

ขั้นที่ 3 การประสานงานของระบบเพื่อตอบสนองต่อคำสั่งเคลื่อนที่



ภาพ 5.5.2 ค แสดงการประสานของระบบในการตอบสนองต่อคำสั่งเคลื่อนที่

1. God ขอ GameCommand ที่ถูกโพสท์ไว้มาจาก GCmdQ
2. God ส่ง GameCommand ไปให้ zergling โดยเรียกเมทอด recieveCommand()
3. zergling ส่ง GameCommand ให้ zergling_idle_state ซึ่งเป็นสถานะปัจจุบันทำการตอบสนองต่อคำสั่ง
4. zergling_idle_state ตัดสินว่าให้มีการเปลี่ยนสถานะไปเป็น zergling_gostrength_state
5. zergling กำหนด zergling_gostrength_state ให้เป็นสถานะปัจจุบันและกำหนดจุดเป้าหมายให้แก่ zergling_gostrength_state
6. zergling_gostrength_state กำหนดจุดเป้าหมายในการเคลื่อนที่ให้แก่ zergling_motivation
7. God ให้โอกาสแก่ zergling ในการกระทำ 1 หน่วยของงานโดยเรียกเมทอด takeAChance()
8. zergling เรียก zergling_motivation ให้กระทำหนึ่งหน่วยของงานโดยเรียกเมทอด work
9. zergling_motivation ตัดสินใจที่จะขยับไปยังตำแหน่งใกล้เคียงจึงโพสท์คำขออนุญาตเคลื่อนที่ไว้ที่ EventList
10. Godอ่านคำร้องขอต่างๆจาก EventList โดยการสุ่มลำดับ

11. God พิจารณาคำขอและส่งผลการตัดสินกลับไปยังตัวละครเจ้าของคำร้องซึ่งในกรณีนี้ก็คือ zergling
12. zergling เห็นว่าเป็นผลการตัดสินคำขอในการเคลื่อนที่จึงส่งผ่านคำตัดสินไปยัง zergling_motivation
13. zergling_motivation เมื่อได้รับอนุญาตให้เคลื่อนที่ได้จึงทำการเปลี่ยนตำแหน่งของ zergling โดยลบ zergling ออกจากตำแหน่งเดิม และ เพิ่มลงบนตำแหน่งใหม่

บทที่ 6

การอิมพลีเมนต์ X-RTS เกมเอนจิน

6.1 ออบเจ็กต์โมเดลในภาษาซีพลัสพลัส

จากโมเดลที่เราได้ออกแบบมาด้วย ยูเอ็มแอลแล้วในการนำไปอิมพลีเมนต์นั้นเราต้องทำการแมปออบเจ็กต์โมเดลยูเอ็มแอลสู่ออบเจ็กต์โมเดลในภาษาที่เราเลือกใช้คือภาษาซีพลัสพลัส ภาษาซีพลัสพลัส ถูกสร้างโดย Bjarne Straustrup ในปีคริสต์ศักราช 1980 บนพื้นฐานของภาษาซีและ ซิมมูล่า โดยมุ่งหมายให้เป็นภาษาที่รองรับหลายกระบวนทัศน์ (Multi Paradigm Programming Language) แต่แนวทางที่ถูกนำมาใช้เกือบทั้งหมดจะเป็นการนำมาใช้แบบกระบวนทัศน์เชิงวัตถุซึ่งจะได้นำเสนอดังนี้

6.1.1 คลาส

เป็นกลไกที่ใช้ในการนิยามเดต้าไทร์ปชนิดผู้ใช้นิยาม ประกอบด้วยเมมเบอร์เดต้า และ เมมเบอร์ฟังก์ชัน ซึ่งสามารถเทียบได้กับ แอททริบิว และ โอเปอเรชันในภาษายูเอ็มแอล โดยคลาสจะทำหน้าที่เป็นแม่แบบของออบเจ็กต์ต่าง ๆ ในโปรแกรม คลาสในภาษาซีพลัสพลัสนั้นมีความหมายที่แน่ชัดในช่วงเวลาคอมไพล์ไทม์ ส่วนความหมายในช่วงรันไทมท์มาตรฐานแอนไซไอโซซีพลัสพลัสนั้นยังไม่ได้นิยามไว้สมบูรณ์แบบเหมือนภาษาจาวา ซึ่งในตลาดส่วนนี้จะเป็นการนิยามโดยมาตรฐานของแต่ละผู้ผลิตคอมไพเลอร์/คลาสไลบรารีนั้นเอง เช่นเอ็มเอฟซีของบริษัทไมโครซอฟต์ คลาสของภาษาซีพลัสพลัสมีกลไกของ การซ่อนอินฟอร์เมชัน อิมพลิเมนเตชัน โดยการใช้ มอดิไฟล์เออร์ต่าง ๆเพื่อควบคุมความสามารถในการเข้าถึงสมาชิกเหล่านั้นซึ่งได้แก่ public, protected, private ซึ่งสามารถเปรียบเทียบโดยตรงกับความหมายเดียวกันกับของยูเอ็มแอล

6.1.2 ออบเจ็กต์

ออบเจ็กต์ในภาษาซีพลัสพลัสจะมีที่อยู่ในเมนเมโมรีโดยมีรูปแบบของช่วงชีวิตของออบเจ็กต์ดังนี้

- แบบออโตเมติก
  ตัวออบเจ็กต์จะถูกเก็บในแสดกซึ่งออบเจ็กต์จะถูกสร้างและทำลายโดยอัตโนมัติเมื่อเข้าและออกสโคบตามลำดับ
- แบบไดนามิก
  ตัวออบเจ็กต์จะถูกเก็บในฮีบโดยใช้กลไกการจองเมโมรีแบบไดนามิคโดยอาศัยกลไกนี้ของโฮสเอนไวรอนเมนต์ ออบเจ็กต์แบบนี้จะมีช่วงชีวิตที่เริ่มจากการสร้างที่เด่นชัดด้วยโอเปอร์เรเตอร์ new และจะถูกทำลายด้วยโอเปอร์เรเตอร์ delete ซึ่งเป็นหน้าที่ของผู้พัฒนาเองในการจัดการส่วนนี้ซึ่งถ้าไม่ได้รับการพัฒนาอย่างถูกต้องจะเกิดปัญหาเมโมรีลีก
- แบบสแตติก
  ช่วงชีวิตของออบเจ็กต์แบบนี้เกิดพร้อมกับการเริ่มของโปรแกรมในช่วงโหลดไทม์และจะจบช่วงชีวิตพร้อมกับโปรแกรมด้วยเช่นกัน

ในแอนไซน์ไอโซซีพลัสพลัสนั้นไม่ได้มีการนิยามออบเจ็กต์แบบเพอซิสแตนส์เอาไว้ซึ่งในทางปฏิบัติมักพบปัญหากับจุดนี้เป็นจำนวนมากโดยเฉพาะการพัฒนาเกมต่าง ๆซึ่งต้องเกี่ยวข้องกับการโหลดและ

เซฟเกม ซึ่งในส่วนนี้ผู้พัฒนามักต้องพัฒนากลไกนี้ขึ้นมาเองหรือพึงพากลไกอื่นที่หามาได้ทั้งเสียค่าใช้จ่ายและไม่เสียค่าใช้จ่าย ในส่วนของ X-RTS เกมเอนจินยังไม่ได้รองรับในส่วนนี้เอาไว้จึงต้องใช้กลไกในรูปแบบที่ไม่เป็นระเบียบ

- ไอเดนติตี้

  ไอเดนติตี้ของออบเจ็กต์ในภาษาซีพลัสพลัสนั้นใช้แอดเดรสของหน่วยความจำเป็นไอเดนติตี้ของออบเจ็กต์การพัฒนาระบบเพอซิสแตนส์ของผู้พัฒนาต้องคำนึงถึงสิ่งนี้ด้วยเพราะเป็นเงื่อนไขสำคัญในการพัฒนาผู้พัฒนาระบบเพอซิสแตนส์บนซีพลัสพลัสต้องมีระบบการออกไอเอนติตี้เฉพาะของตนออกมา

6.1.3 แมสเซจ

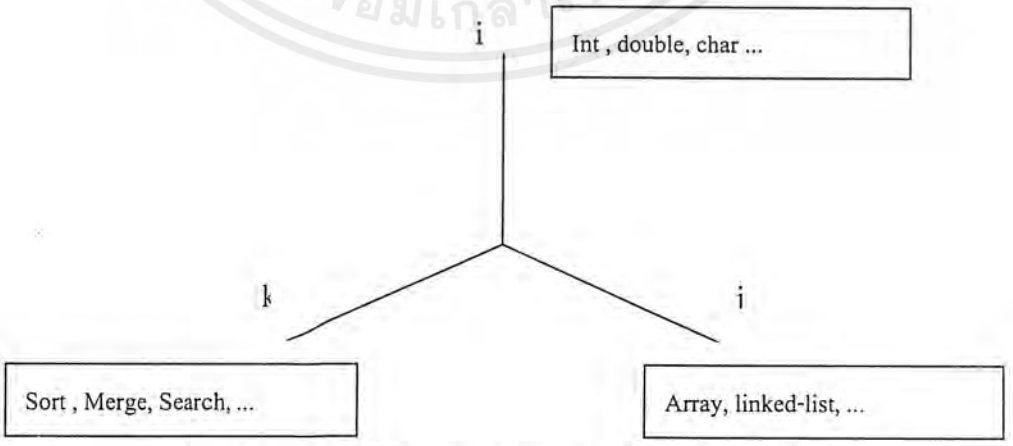
แมสเซจที่ใช้ในภาษาซีพลัส ๆ รองรับแมสเซสในแบบซิงโครนัส ซึ่งจะเกิดในการเรียกฟังก์ชันกลไกการส่งพารามิเตอร์ ส่วนแมสเซจในแบบอะซิงโครนัสนั้นเป็นหน้าที่ของผู้พัฒนาเองที่จะพัฒนาในส่วนนี้ใน X-RTS เกมเอนจินได้แก่กลไกอีเวนต์ของเกมออบเจ็กต์และอีเวนต์ในระบบกราฟิกยูเซอร์อินเทอร์เฟส

6.1.4 พอลิมอฟิซึ่ม

ในภาษาซีพลัสพลัสรองรับกลไกนี้ได้โดยการประกาศ ให้เมมเบอร์ฟังก์ชัน ใดด้วยมอดิไฟเออร์ virtual ซึ่งจะทำให้กลไกนี้เกิดขึ้นได้ โดยภาษาซีพลัสพลัสใช้กลไกที่เรียกว่า v-table ซึ่งเป็นตารางที่เก็บพอยท์เตอร์ไปยังตำแหน่งของโคดและตัวแปรที่ผูกพันไว้ซึ่งสามารถถูกพิจารณาได้แบบรันไทม์

6.2 STL ทูลคิด

STL (Standard Template Library) คือโปรแกรมมิ่งไลบรารีภาษาซีพลัสพลัส ซึ่งถูกออกแบบโดย Alexander Stepanov และ Meng Lee ที่ห้องวิจัยในบริษัท ฮิวเลต แพคการด์ ในเมืองพาโล อัลโต รัฐแคลิฟอเนีย สหรัฐอเมริกา STL ถูกออกแบบมาให้ โปรแกรมเมอร์ภาษาซีพลัสพลัสสามารถทำการโปรแกรมมิงเชิงเจเนริกได้ โดยใช้พื้นฐานของเทมเพลตซึ่งเป็นความสามารถในภาษาซีพลัสพลัสแนวคิดของ หลักของ STL มาจากการแบ่งแยกมิติของ ซอฟต์แวร์ออกเป็นสามมิติซึ่งได้แก่



ภาพ 6.2 แสดงมิติของซอฟต์แวร์แนวคิดพื้นฐานของ STL

มิติ i เป็นชนิดข้อมูล เป็นส่วนที่แทนข้อมูลและรายละเอียดที่เราสนใจ

มิติ j เป็นโครงสร้างข้อมูล เป็นส่วนที่แทนรูปแบบและระเบียบวิธีการเชื่อมโยงการเข้าถึงข้อมูล

มิติ k เป็นขั้นตอนวิธีการทำงาน เป็นส่วนที่แทนการขั้นตอนและดำเนินการต่าง ๆเพื่อให้ได้ผลลัพธ์ตามที่ต้องการ

จากแนวคิดนี้ในการโปรแกรมที่ไม่ได้มีการแยกมิติเหล่านี้ออกจากกันการโปรแกรมหนึ่งใด ๆที่เกี่ยวข้องกับทั้งสามมิติของซอฟต์แวร์จะเป็นการรวมกันของทั้งสามมิติผูกติดกัน ซึ่งหมายความว่าถ้าเวอร์ชันทั้งหมดของซอฟต์แวร์ที่เราโปรแกรมจะมีถึง i*j*k เวอร์ชันแต่ถ้าเราแยกทั้งสามมิติออกจากกันและใช้การผูกติดในเวลาที่ต้องการเราต้องการการโปรแกรมเพียง i+j+k เวอร์ชันเท่านั้น

จากแนวคิดข้างต้น STL ทำให้การพัฒนาซอฟต์แวร์เป็นไปได้อย่างรวดเร็วยิ่งขึ้นทำให้การดีบักโปรแกรมเป็นไปง่ายดายขึ้น และเพิ่มการโอนผ่านโคดข้ามระบบ ทำให้ผู้พัฒนาซอฟต์แวร์ใช้ความสามารถของตนเจาะจงไปในงานของตนได้มากขึ้นและสนใจรายละเอียดอื่น ๆ ลดลง

STL ประกอบด้วยส่วนประกอบหลักห้าส่วนด้วยกันซึ่งได้แก่

- Algorithm

  กระบวนการเชิงการคำนวณที่สามารถใช้ได้กับคอนเทนเนอร์ต่างชนิดกันได้

- Containner

  ออบเจ็กต์ที่ทำหน้าที่บรรจุและจัดกาออบเจ็กต์อื่น

- Iterator

  วิธีการเข้าถึงแต่ละสมาชิกในคอนเทนเนอร์ที่เป็นนามธรรมดังนั้นจึงสามารถใช้เข้าถึงคอนเทนเนอร์ได้หลายรูปแบบ

- Functional Object

  คลาสที่นิยามโอเปอเรเตอร์ เรียกฟังก์ชัน (operator())

- Adaptor

  แปลงอินเทอร์เฟสของการติดต่อส่วนประกอบต่างๆให้เป็นอินเทอร์เฟลในรูปแบบที่สนใจเนื่องจาก

STL ประกอบด้วยสมาชิกเป็นจำนวนนับร้อยการอธิบายลงรายละเอียดจะเป็นการออกนอกประเด็นจนเกินไปดังนั้นจึงขอนำเสนอเฉพาะสมาชิกที่เลือกนำมาใช้ในการอิมพลีเมนต์ X-RTS เกมเอนจินเท่านั้น

- คอนเทนเนอร์
  - Vector

    เป็นซีเควนเชียลคอนเทนเนอร์ซึ่งคล้ายคลึงกับ อาเรย์ในภาษาซีแต่ขนาดของมันสามารถขยายได้อัตโนมัติ ใน X-RTS เกมเอนจินใช้ในการทำการเก็บลิสต์ของเกมออบเจ็กต์ที่ถูกเลือกเอาไว้

  - List

    เป็นซีเควนเชียลคอนเทนเนอร์ภายในเป็นลิงค์ลิสแบบสองทางการเข้าถึง โหนดใดใดจะใช้การเข้าถึงได้แบบลำดับในเกมเอนชินใช้ในการเก็บลิสต์ของเหตุการณ์ในเกม

- Map

  เป็นแอสโซสิเอทีพคอนเทนเนอร์ซึ่งจัดการคู่ของ กุญแจและค่า โดยคู่เหล่านั้นจะถูกเรียงลำดับด้วยกุญแจ

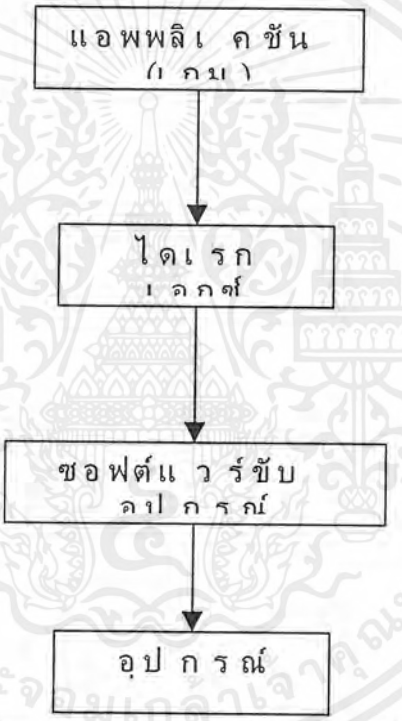
- อัลกอริทึม

  - Random_shuffle

    เป็นอัลกอริทึมที่ทำหน้าที่สลับที่แบบสุ่มใช้ในการสลับเกมอีเวนต์ในเกมอีเวนต์คิวในเพื่อให้เหตุการณ์กระจายตัวอย่างสุ่มและมีความยุติธรรม

6.3 DirectDraw

ไดเรกดรอว์เป็นส่วนหนึ่งของไดเรกเอ็กซ์ซึ่งเป็นเอพีไอที่กำหนดโดยบริษัทไมโครซอฟต์โดยมุ่งหวังให้เป็นแพลตฟอร์มของการพัฒนาโปรแกรมที่ต้องใช้กราฟิกบนระบบ win32 โดยแนวคิดหลักของของไดเร็กเอ็กซ์แสดงดังรูป



ภาพ 6.3 ก แสดงแนวคิดของการสร้างไดเร็กเอ็กซ์เอพีไอ

ในส่วนของไดเร็กเอกซ์เอพีไอนี้จะถูกนิยามในรูปแบบที่แอพพลิเคชันต่างๆ ต้องการโดยจะทำการจับคู่ส่วน เอพีไอนี้กับซอฟต์แวร์ขับอุปกรณ์ที่มีรูปแบบต่างๆกันไป ทำให้ผู้พัฒนาสามารถพัฒนาโปรแกรมโดยไม่ต้องมีภาระในการเขียนรองรับอุปกรณ์ที่มีหลากหลายทำให้การพัฒนาเป็นไปได้อย่างง่ายขึ้นนอกจากนั้นไดเร็กเอ็กซ์เอพีไอยังให้โอกาสในการเข้าถึงอุปกรณ์ในระดับล่างพอที่ผู้พัฒนาจะปรับแต่งเลือกใช้ความสามารถต่างๆ ได้ดังต้องการ และมีความรวดเร็ว ซึ่งแต่เดิมกลไก GDI นั้นรองรับตรงส่วนนี้ได้ไม่ดีเนื่องจากถูกออกแบบมาเพื่อการแสดงผลโปรแกรมทางธุรกิจ

ใน X-RTS เกมเอนจินนี้ใช้งานส่วนไดเร็กดรอว์ของไดเร็กเอ็กซ์ซึ่งถูกใช้หลัก ๆดังนี้

- DirectDraw

  ในการทำงานของ ไดเร็กดรอว์ในเอนจินนี้โคดส่วนที่ทำการกับไดเร็กดรอว์โดยตรงจะอยู่ในส่วนของ GFXManager ออบเจ็กต์ซึ่งทำหน้าที่เป็นกราฟิกเอนจิน

  - การสร้างไดเร็กซ์ดรอว์เซอร์เฟส

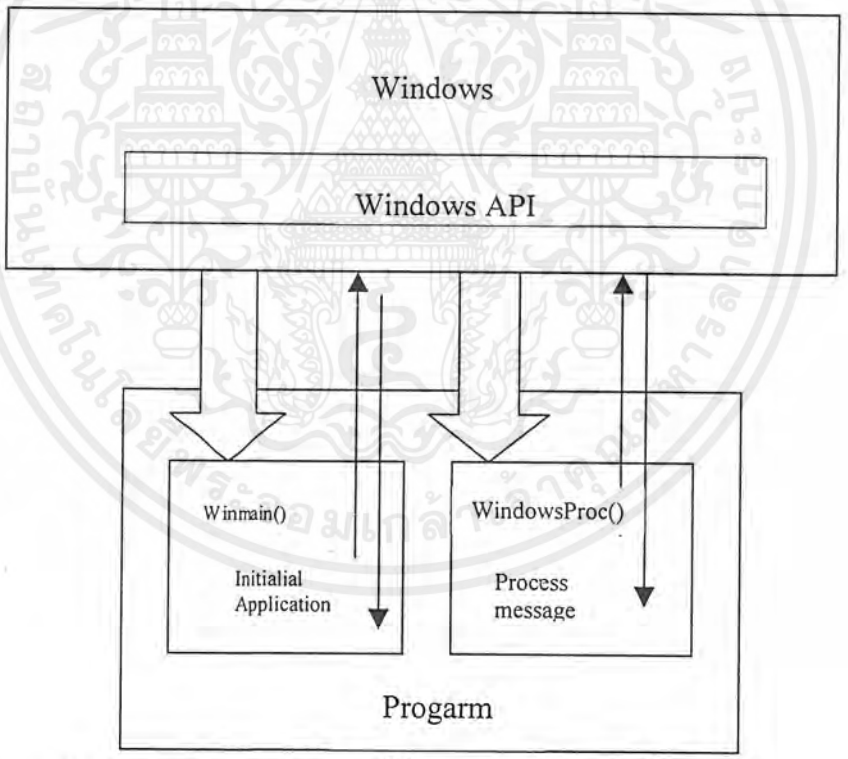
    ไดเร็กซ์ดรอว์เซอร์เฟสคือพื้นที่ในเมโมรีที่ทำหน้าที่เก็บข้อมูลจุดภาพสำหรับการแสดงผลซึ่งจะถูกสร้างเมื่อ GFXManager ออบเจ็กต์ทำการเรียก Init()

  - การวาดภาพลงในไดเร็กซ์ดรอว์เซอร์เฟส

    ภาพจะถูกวาดโดยผ่าน Graphic ออบเจ็กต์ซึ่งทำหน้าที่เป็นเสมือนผืนภาพของหน้าจอโดยจะทำผ่าน เมมเบอร์ฟังก์ชัน GraphicBlt และ GdrawImage

6.4 สภาพแวดล้อม win32

สภาพแวดล้อม win32 เป็นสภาพแวดล้อมสำหรับการติดต่อกับระบบปฏิบัติการของไมโครซอฟต์ในสถาปัตยกรรมคอมพิวเตอร์แบบ 32 บิต ซึ่งมีกลไกการทำงานแบบวินโดว์และการจัดการแมสเซจระหว่างวินโดว์สามารถแสดงคร่าว ๆ ได้ดังนี้

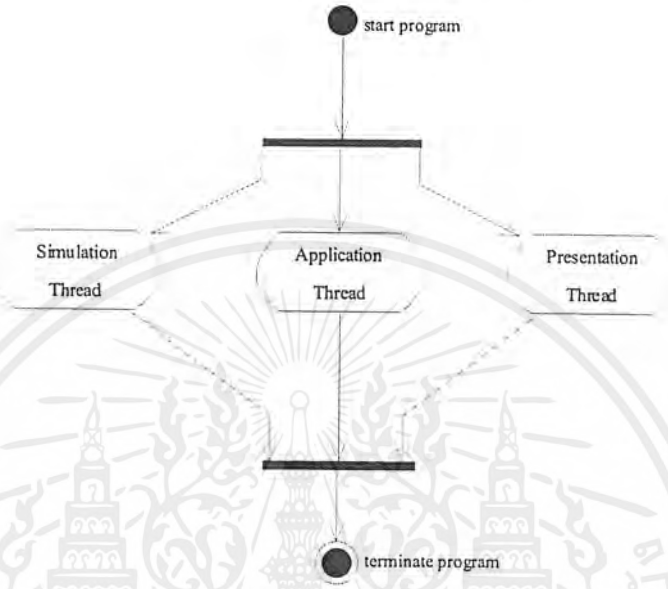


ภาพ 6.4 ก แสดงการทำงานของโปรแกรมบนสภาวะแวดล้อม win32

โปรแกรมจะเริ่มการทำงานที่ ฟังก์ชัน Winmain() โดยในส่วนนี้จะเป็นส่วนที่ทำหน้าที่ในการตั้งค่ากำหนดค่าต่างๆ ของแอพพลิเคชัน โดยในส่วนมีจะเป็น เมสเซจลูปทำหน้าที่ในการจัดการส่งเมสเซจต่างๆ ไปให้ WindowsProc() เมื่อเกิดเหตุการณ์ใดๆขึ้น windows จะส่งแมสเซจมาที่วินโดว์โปรแกรมเพื่อให้โปรแกรมของเราตอบสนองต่อเหตุการณ์นั้นตามที่เรากำหนดไว้แต่อย่างไรก็ตามพบว่าสภาพแวดล้อมแบบที่นำ

เสนอนี้ไม่เหมาะสมและไม่เพียงพอกับการทำงานแบบเกมจึงต้องมีการนำเธรดเข้ามาช่วยเพื่อให้รูปแบบของการทำงานสอดคล้องกับการทำงานของเอนจินมากขึ้น

ใน X-RTS เกมเอนจินจะถูกออกแบบมาให้ทำงานบนสภาวะแวดล้อมแบบนี้โดยทำการเชื่อมโยงการทำงานของโปรแกรมบนสภาวะแวดล้อมกับ การทำงานของเอนจินดังต่อไปนี้



ภาพ 6.4 ขแสดงเธรดควบคุมของ X-RTS เกมเอนจิน

เมื่อโปรแกรมเริ่มทำงานจะเป็นการทำการตั้งค่าต่างๆ ที่ระบบ Win32 ต้องการเช่นการสร้างแอพพลิเคชันวินโดว์ หลังจากนั้นเมื่อเรียก GUI ให้เริ่มทำงาน GUI ซึ่งเป็นซิงเกิลตันจะทำการสร้างและตั้งค่าต่างๆ ตามที่กำหนดเอาไว้ และจะสร้างเธรดสำหรับการทำงานของ GUI โดยทำหน้าที่ในการแสดงผลและรับข้อมูลนำเข้าจากผู้ใช้ หลังจากนั้นโปรแกรมก็จะเรียก GameWorld ขึ้นมาทำงานโดยจะทำการสร้างและตั้งค่าต่าง ๆตามที่กำหนดเอาไว้และจะสร้างเธรดสำหรับการจำลองขึ้นมา

ซิมมูเลชันเธรดและพรีเซนเทชันเธรดจะทำงานประสานงานกันตามรูปแบบที่อธิบายไว้ในส่วนสถาปัตยกรรมส่วน แอพพลิเคชันเธรดนั่นจะว่างและรอเหตุการณ์วินโดว์ต่าง ๆ เข้ามาถ้าไม่มีเหตุการณ์ใด ๆ เธรดนี้จะว่าง เมื่อโปรแกรมถูกผู้ใช้สั่งให้จบ ซิมมูเลชันเธรดจะจบโปรแกรมและพรีเซนเทชันเธรดและแอพพลิเคชันเธรดตามลำดับ

อ้างอิง

6.2 STL ทูลคิด จาก [9] Johannes Weidi "The Standard Template Library Tutorial"

บทที่ 7

วิธีการสร้าง RTS เกม โดยใช้ X-RTS เกมเอนจิน

7.1 ขั้นตอนที่ 1 การสร้างตัวละคร

1 กำหนดความสามารถของตัวละคร ยกตัวอย่าง เช่น

- เคลื่อนที่ได้
- โจมตีได้
- ฟื้นสภาพได้เอง
- ฯลฯ

2 กำหนดคุณสมบัติทั่วไปและคุณสมบัติที่จำเป็นสำหรับความสามารถที่ได้กำหนดไว้ในข้อ 1 ดังต่อไปนี้

- ขนาด
- พลังชีวิต (maximum life points)
- ความแข็งแกร่งในการต้านทานการโจมตี (strength)
- ความรุนแรงในการโจมตี, พิสัยโจมตี
- พิสัยการมองเห็น
- ความเร็วในการเคลื่อนที่
- ฯลฯ

3 กำหนดกริยาที่เป็นไปได้ทั้งหมดของตัวละคร เป็นกริยาที่เห็นความแบ่งแยกได้อย่างชัดเจน กริยาที่มีการใช้อยู่เป็นปกติจะถูกระบุด้วยพรีโพรเซสเซอร์ #define ไว้ในไฟล์ GameObject.h ได้แก่

- ACT_STAND
- ACT_MOVE
- ACT_ATTACK
- ฯลฯ

สำหรับกริยาที่ยังไม่ได้มีการกำหนดไว้ ให้กำหนดเพิ่มเติมลงไปในไฟล์ GameObject.h

4 แบ่งการทำงานของตัวละครออกเป็นส่วนๆ

สร้างอ็อบเจ็คท์ขึ้นมารับผิดชอบการทำงานแต่ละส่วน โดยสืบทอดอินเทอร์เฟซมาจากคลาส GameObjectFunction (กำหนดในไฟล์ "GameObject.h") ดังตัวอย่างเช่น สร้าง XXX_motivation มารับผิดชอบส่วนการเคลื่อนที่ของตัวละครชนิด XXX และ สร้าง XXX_selfhealer มารับผิดชอบส่วนการฟื้นฟูสภาพตนเองสำหรับแต่ละ GameObjectFunction ที่สร้างขึ้นมาใหม่นี้ควรกำหนดเมทอดเพิ่ม

เติมขึ้นมาเพื่อให้สามารถถูกควบคุมและสั่งงานได้จากภายนอก ซึ่งส่วนที่จะเข้ามาสั่งงาน GameObjectFunction เกือบร้อยเปอร์เซ็นต์ก็คือ GameObjectState

GameObjectFunction ถูกประกาศไว้ดังนี้คือ

```
class GameObjectFunction{
public:
        GameObjectFunction(GameObject *GObj);
        void setEnabled(bool b_value);
        bool isEnabled();
        virtual void recieveAcknowledgement(GameEvent *e, ACK ack);
        virtual void work();
protected:
        bool b_isEnabled;
        GameObject *GObj;
};
```

XXX_motivation ต่อไปนี้คือ ตัวอย่างของการสร้าง GameObjectFunction

```
class zergling_motivation:public GameObjectFunction{
public:
        zergling_motivation(GameObject *GObj);
        virtual void recieveAcknowledgement(GameEvent *e, ACK ack);
        virtual void work();
        void stop();
        void setDestination(POINT *dest);
protected:
        void constructPath();
        POINT destination;
        Path path;
        PathFinder *pf;
        int counter;
        int state;
```

```
        bool isNewDest;
};
```

จากโค้ดข้างต้น เมทอดที่ถูกกำหนดเพิ่มเติมเพื่อใช้ควบคุมการทำงานคือ setDestination() ที่ใช้ในการกำหนดเป้าหมาย และ stop() ที่ใช้ในการสั่งหยุดเคลื่อนที่

5 ออกแบบลักษณะนิสัยของตัวละคร และกำหนดสถานะหลักๆที่เป็นไปได้ของตัวละคร เช่น

- สถานะที่จะเคลื่อนที่และทำลายศัตรูที่ขวางทาง กำหนดเป็น XXX_move_and_attack_state
- สถานะที่จะเคลื่อนที่ไปยังจุดหมาย กำหนดเป็น XXX_move_state ถึงแม้ตัวละครจะถูกทำร้ายก็จะไม่หยุดต่อสู้จนกว่าจะถึงจุดหมาย
- สถานะอยู่รักษาพื้นที่ กำหนดเป็น XXX_hold_position_state ตัวละครจะไม่เคลื่อนที่เด็ดขาด แต่จะยังสามารถโจมตีศัตรูที่เข้ามาในพิสัยการโจมตี

ฯลฯ

ทุกๆสถานะจะต้องสืบทอดมาจากคลาส GameObjectState (กำหนดในไฟล์ "GameObject.h") ซึ่งมีส่วนประกาศดังนี้

```
class GameObjectState{
public:
        GameObjectState(GameObject *GObj);
        virtual void recieveCommand(GameCommand *cmd){};
        virtual void recieveEvent(GameEvent *e){};
        virtual void recieveAcknowledgement(GameEvent *e, ACK ack){};
        virtual void takeAChance(){};
protected:
        GameObject *GObj;
};
```

ต่อไปนี้คือคำอธิบายเมทอดที่สำคัญ

ทุกๆ 1 หน่วยเวลา สถานะปัจจุบัน(อินสแตนท์ของ GameObjectState)จะได้รับโอกาสในการตัดสินใจว่าจะดำเนินการใดต่อไปโดยผ่านทางการเรียกเมทอด ***takeAChance()*** GameObjectState จะตัดสินใจบนพื้นฐานของข้อมูลสิ่งแวดล้อมและสถานะภายใน

God จะส่งคำสั่งมาให้ GameObject และผ่านมายังสถานะปัจจุบันทางเมทอด ***recieveCommand()*** คำสั่งนี้อาจจะถูกละเลยได้ถ้าสถานะปัจจุบันไม่เอื้ออำนวย

God จะบอกกับ GameObject ว่าเกิดเหตุการณ์ใดที่เกี่ยวข้องกับมันบ้าง และเหตุการณ์นั้นจะถูกส่งผ่านมายังสถานะปัจจุบันผ่านทาง ***recieveEvent()*** สถานะปัจจุบันมีหน้าที่ในการตัดสินใจและกระทำการใดๆ เพื่อตอบต่อสนองเหตุการณ์นั้นๆอย่างเหมาะสม

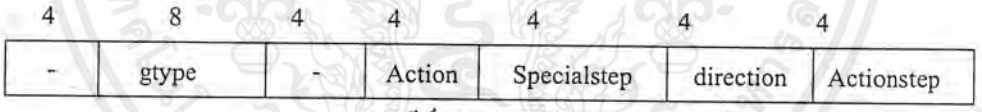
ในการกระทำบางอย่างจะต้องมีการขออนุญาตจาก God เสียก่อน จากนั้นผลการตัดสินว่าอนุญาติให้กระทำหรือไม่จะถูกส่งกลับมาโดยเรียกเมทอด recieveAcknowledgement()

ขั้นตอนที่ 2 การเตรียมรูปภาพในการใช้วาดตัวละคร

คุณสมบัติของตัวละครที่ใช้ในการกำหนดรูปภาพที่ใช้ในการวาดตัวละครได้แก่ gtype, groupid, action, direction, actionstep และ specialstep เราต้องหาว่าในช่วงชีวิตของตัวละครนี้ สมมติเป็น XXX จะสามารถเกิดกรณีใดได้บ้างของข้อมูลข้างต้น (combination) จากนั้นสร้างภาพที่จะใช้แสดงกริยาของตัวละครในแต่ละสถานะ

ตัวละครชนิดหนึ่งๆอาจอยู่ฝ่ายใดก็ได้ในเกมซึ่งจะทำให้มีสีสันต่างกันไป เราจะต้องสร้างภาพให้ครบทุกสีที่ต้องการ ซึ่งใน X-RTS เกมนี้กำหนดไว้สูงสุด 8 ฝ่ายดังนั้นเราต้องสร้างภาพที่แสดงกริยาเดียวกันทั้งหมด 8 สี เมื่อได้ภาพสำหรับทุกกริยาแล้ว นำภาพสีเดียวกันมารวมอยู่ในไฟล์เดียวกัน จะทำให้ได้ไฟล์รูปภาพทั้งหมด 8 ภาพ

ในการเตรียมพร้อมการแสดงเหตุการณ์ในเกมผ่านทาง MapViewer เราจะต้องสั่งให้ GodOfRenderring ทำการโหลดรูปภาพไปเก็บไว้โดยเรียกเมทอด loadPlayerBMP(ชื่อไฟล์) จากนั้นจะต้องกำหนด bitmap mapping ที่ใช้สำหรับแต่ละกริยา คีย์ที่ใช้ในการแมปจะสร้างจากข้อมูลสถานะของตัวละครดังนี้



คีย์ขนาด 32 บิต

ภาพที่ 7.1 ก แสดงคีย์ขนาด 32 บิตที่ใช้ในการแสดง Game View บน MapViewer

ข้อมูลที่ต้องกำหนดเพื่อใช้ในการวาดภาพแต่ละกริยาคือ

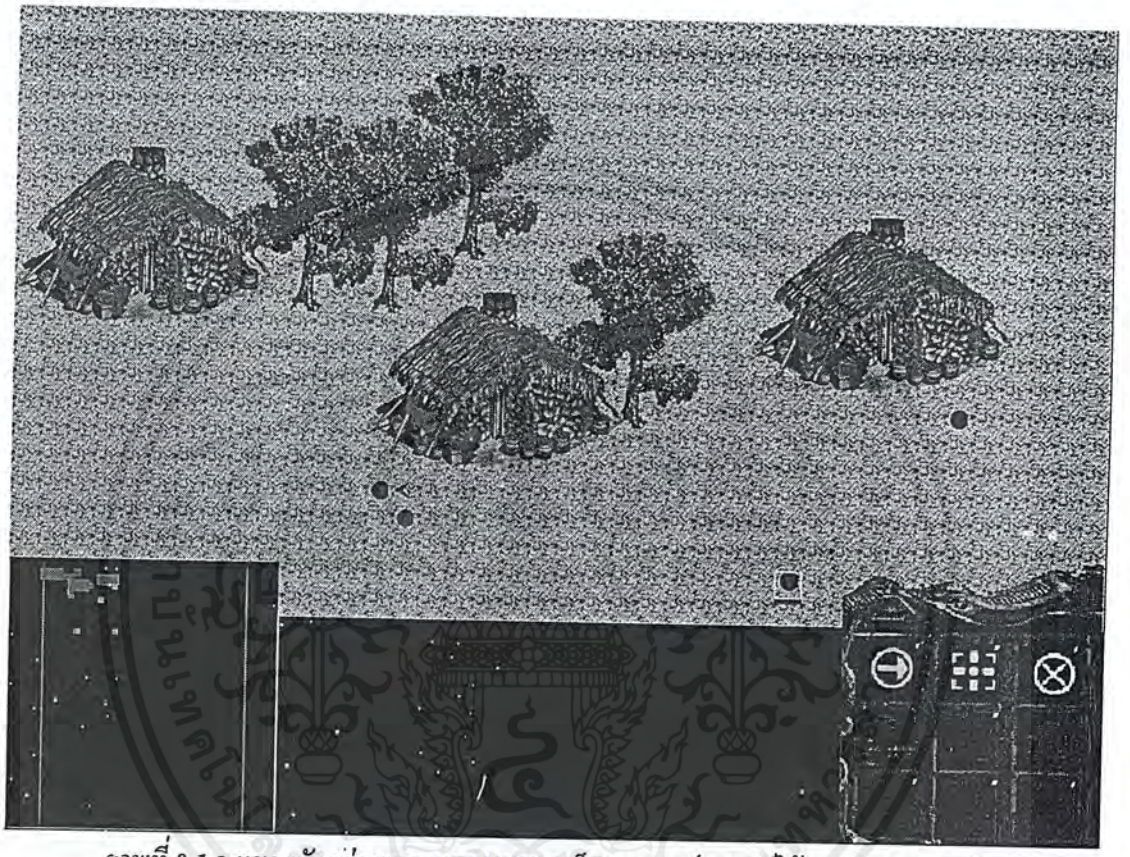
- พื้นที่สี่เหลี่ยมบนภาพใหญ่ที่จะใช้ (RECT มี left, top, right และ bottom)
- ตำแหน่งที่จะวาดภาพอ้างอิงกับตำแหน่งของตัวละคร

บทที่ 8

การประเมินผลและบทสรุป

8.1 สรุปผลที่ได้จากการนำ X-RTS เกมเอนจินไปประยุกต์ใช้

จากการทดลองการนำ X-RTS เกมเอนจิน ไปสร้าง เกมทดสอบขนาดเล็กซึ่งสามารถดูรายละเอียดการทดลองได้ใน คู่มือโปรแกรมที่ควบคู่กับโครงงานนี้โดยจะไม่กล่าวในที่นี้เพราะจะเป็นการซ้ำซ้อนกัน ซึ่งได้ผลลัพธ์จากการทดลองดังนี้



ภาพที่ 8.1 ก แสดงตัวอย่างเกมทดสอบขนาดเล็กจากการประยุกต์ใช้ X-RTS เกมเอนจิน

จากภาพสามารถอธิบายคร่าว ๆ ได้ดังนี้

เมื่อเริ่มโปรแกรมจะเห็นหน้าต่าง หลายหน้าต่างที่ทำหน้าที่ต่าง ๆ กันไป หน้าต่างใหญ่ที่สุดเป็นมุมมองสังเกตการของโลกสมมุติ หน้าต่างล่างขวา แสดงมุมมองแผนที่ขนาดเล็กเพื่อการเปลี่ยนมุมมองสมมุติในเกมและการแสดงเหตุการณ์ในโลกสมมุติอย่างย่อ มุมมองล่างขวาเป็นปุ่มที่สามารถกดเพื่อทำการควบคุมตัวละครปัจจุบันในขณะนั้น ผู้เล่นสามารถควบคุมตัวละครฝ่ายใดก็ได้เนื่องจากในที่นี้ยังไม่มีผู้เล่นที่เป็น คอมพิวเตอร์ โดยเลือกตัวละครที่จะควบคุม (ก้อนวงกลม) ด้วยการคลิ้กเมาส์ซ้ายไปที่ตัวละครนั้นๆ ในมุมมองสังเกตุการณ์ เมื่อเลือกตัวละครสำเร็จจะเห็นตัวละครที่พร้อมรับคำสั่งมีกรอบสี่เหลี่ยมขึ้นมาแสดงแถบพลังในขณะนั้น ผู้เล่นสามารถเลือกตัวละครเพื่อควบคุมทีละ มากกว่าหนึ่งตัวได้โดยการ กดแป้น Shift ค้างไว้แล้วใช้เมาส์ซ้ายเพื่อเพิ่มเติมตัวละครที่จะควบคุม ผู้ใช้สามารถควบคุมตัวละครให้ทำงานได้โดยการกดปุ่มคำสั่งที่มุมมองคำสั่งที่หน้าต่างด้านขวาล่าง หรือคำสั่งที่เป็นค่า ดีฟอลต์คือการ คลิ้กปุ่มขวาไปที่เป้าหมายใด ๆ โดยระบบควบคุมจะทำการตีความคำสั่งตามเป้าหมายให้เอง เช่นเมื่อเราเลือกตัวคร

ให้พร้อมรับคำสั่งแล้ว เรากดเม้าส์ขวาไปที่ สิ่งก่อสร้างที่เป็นบ้านของศัตรู ตัวละครก็จะเดินไปยังเป้าหมายและทำลายสิ่งก่อสร้างนั้น ๆ หรือ ถ้าเปลี่ยนเป้าหมายเป็นตัวละครศัตรู ตัวละครที่เราควบคุมอยู่ก็จะเดินไปหาเป้าหมายเพื่อทำการต่อสู้ตามคำสั่ง หากเป้าหมายเป็นบริเวณว่างเปล่า จะหมายถึงการสั่งให้ตัวละครที่พร้อมรับคำสั่งเคลื่อนที่ไปยังจุดนั้น ในกรณีที่เราต้องการสั่งตัวละครให้เคลื่อนที่ออกนอกบริเวณมุมมองปัจจุบัน ก็สามารถ สั่งให้เคลื่อนที่ออกนอกบริเวณด้วยการกดปุ่มเม้าส์ขวาไปที่ มุมมองแผนที่ขนาดเล็ก ที่หน้าต่างมุมล่างซ้ายของจอ ผู้เล่นสามารถเปลี่ยนมุมมองเพื่อติดตามดูสถานการณ์ได้โดยการคลิ้กเมาส์ซ้ายที่ บริเวณที่ต้องการสังเกตที่จุด ใด ๆ บนมุมมองแผนที่เล็ก เมื่อศัตรูหรือสิ่งก่อสร้างถูกโจมตีจนพลังหมดก็จะหายไปจากโลกสมมุติในเกม พฤติกรรมบางอย่างจะสามารถเกิดขึ้นได้เองโดยอัตโนมัติเช่นเมื่อตัวละครเขาไปใกล้จนถึงระยะมองเห็นของศัตรู ก็จะเข้ามาเพื่อทำการต่อสู้ด้วยเอง เกมนี้ไม่มีการเล่นจนจบเกม ถูกสร้างมาเพื่อใช้ในการทดสอบเท่านั้น

8.2 การประเมินผล

8.2.1 วัดเชิงปริมาณ

ในการประเมินผลงานงานทางซอฟต์แวร์ในด้านปริมาณนั้นสามารถทำได้หลายรูปแบบแต่ที่จะนำมาใช้ในการประเมินผลครั้งนี้ได้แก่ จำนวนบรรทัดของโคดโปรแกรมที่ได้เขียนลงไป และในระดับที่เป็นกลุ่มเป็นก้อนขึ้นได้แก่ จำนวน ฟังก์ชัน คลาส หรือ โมดูล และ สำหรับบางส่วนที่มีความซับซ้อน มักจะถูกพิจารณาแยกเป็นพิเศษเช่น ส่วนที่เกี่ยวข้องกับปัญญาประดิษฐ์ การจำลองพฤติกรรม วิธีที่เลือกนำมานั้นเป็นวิธีที่ไม่ซับซ้อนนักแต่ก็พอที่จะใช้ชี้วัดได้ในระดับหนึ่งแล้ว

- *ระดับบรรทัดของโคดโปรแกรมทั้งหมด (Line of Code)*
  - ส่วนประกาศ (*.h) 1449 บรรทัด
  - ส่วนอิมพลีเมนเตชั่น (*.cpp) 4366 บรรทัด
- *ระดับหน่วยตามกระบวนทัศน์ที่ใช้*
  - จำนวน คลาส 56 คลาส
  - จำนวน เมมเบอร์ฟังก์ชั่น 406 เมมเบอร์ฟังก์ชั่น
  - จำนวน เมมเบอร์เดต้า 230 เมมเบอร์เดต้า
- *ส่วนที่มีความซับซ้อนเป็นพิเศษ*
  - ทั้งสิ้น 21 ฟังก์ชัน

8.2.2 วัดเชิงคุณภาพ

ในการวัดคุณภาพซอฟต์แวร์ประเภทเฟรมเวิร์กในเวลาสั้นๆค่อนข้องจะหาผลที่ชัดเจนได้ยากเนื่องจากคุณค่าของซอฟต์แวร์ประเภทนี้คือการรองรับความต้องการได้หลากรูปแบบมากที่สุดโดยคงไว้ซึ่งส่วนหลัก ซึ่งเห็นได้ว่าความต้องการที่จะนำมาทดสอบนั้นควรได้มาจากสถานการ์ณการใช้งานจริงแต่อย่างไรก็ตาม เราสามารถใช้หลักการของการเปรียบเทียบกับสิ่งที่มีอยู่แล้วมาใช้ได้ โดยเฉพาะในแง่ของความรู้สึกในการใช้งานซึ่งในส่วนนี้อาศัยประสบการณ์ส่วนตัวของผู้พัฒนาเองและจากการทดลองประกอบกัน

- *ความสมบูรณ์ของเฟรมเวิร์ก*

  อยู่ในระดับค่อนข้างต่ำเมื่อเทียบกับเฟรมเวิร์กของเกมในท้องตลาดยังขาดในส่วนของการรองรับพื้นฐานด้านต่าง ๆ เช่น การจัดการข้อมูลเกม การจัดการโหลดและการเซฟเกม และเกม ออบเจ็กต์ โครงสร้างในส่วนการติดต่อผ่านเครือข่าย การทำงานเกี่ยวกับเสียง ปัญญาประดิษฐ์ในการเล่นโต้ตอบกับมนุษย์

- *การรองรับแนวคิดระดับแอพพลิเคชัน*

  อยู่ในระดับปานกลางเมื่อเทียบกับเฟรมเวิร์กของเกมที่มีอยู่ในท้องตลาดแม้จะมีส่วนหลัก ๆที่ได้รองรับแนวคิดไปบ้างซึ่งได้ การประมวลผลเหตุการณ์ ส่วนการแสดงผลและการติดต่อกับผู้ใช้ ส่วนการจำลองทางภูมิศาสตร์ แต่ก็ยังขาดส่วนที่เป็นรายละเอียดสำคัญ เช่นส่วน ทริกเกอร์เหตุการณ์ รูปแบบการจำลองพฤติกรรมยังไม่สะดวกสบายต่อการนำไปใช้งานซับซ้อน

- *ความสามารถในการขยายได้*

  อยู่ในเกณฑ์ที่ค่อนข้างดีสามารถพัฒนาต่อไปได้โดยไม่ขัดกับส่วนที่ทำไว้ก่อนแล้วโดยสังเกตจากการเพิ่มเข้าไปในแต่ละครั้งไม่ต้องไปแก้ในส่วนพื้นฐาน และสามารถมองเห็นช่องทางและความเป็นไปได้ในการต่อเติมส่วนต่าง ๆเข้าไปใหม่โดยไม่ขัดกับหลักการเดิมที่ได้วางไว้แล้ว

- *สมรรถนะและความเร็วในการทำงาน*

  อยู่ในระดับปานกลางถึงค่อนข้างต่ำเมื่อเทียบกับเฟรมเวิร์กของเกมที่มีอยู่ในท้องตลาด และไม่อยู่ในระดับที่เลวร้ายจนเกินไปนักและสามารถปรับแต่งให้มีความเร็วที่สูงขึ้นได้โดยไม่กระทบต่อแนวคิดหลัก

- *ความสามารถนำกลับมาใช้ใหม่*

  ในข้อนี้ทำการวัดได้ยากแต่ก็คาดว่าอยู่ในระดับที่ปานกลางเพราะมีระดับของการแบ่งคลาสที่ไม่เล็กย่อยหรือใหญ่โตจนเกินไปโดยการเปรียบเทียบกับส่วนประกอบที่คล้าย ๆกันในเฟรมเวิร์กที่มีอยู่ในท้องตลาด

อย่างไรก็ตามการวัดเชิงคุณภาพเชิงเปรียบเทียบกับเฟรมเวิร์กที่มีในท้องตลาดนั้นเป็นการวัดในแง่ของตัวงาน ซึ่งเป็นมิติหนึ่งในการวัดแต่ไม่ได้บ่งชี้ว่าโครงงานนี้ด้อยคุณค่าเพราะหากเปรียบเทียบโครงงานนี้กับโครงงานแบบเดียวกันในระดับมืออาชีพ ในแง่กับงบประมาณ แรงงานที่ลงไป ความชำนาญ และความได้เปรียบจาก ด้านอื่น ๆ ของ งานระดับมืออาชีพแล้วโครงงานนี้นับว่าอยู่ในระดับที่ใช้ได้เลยทีเดียว โดยเฉพาะการสร้างเฟรมเวิร์กจริงๆนั้นต้องทำหลาย ๆ รอบจนอยู่ในสถานะที่อยู่ตัว ซึ่งเป็นลักษณะเฉพาะของการพัฒนาซอฟต์แวร์ประเภทเฟรมเวิร์กดังนั้นในการพัฒนาต่อยอดจากโครงงานนี้อาจให้ผลที่ดีใกล้เคียงเทียบเท่า หรือเหนือกว่า เฟรมเวิร์กที่มีในท้องตลาดก็มีโอกาสเป็นไปเป็นได้

8.3 การปรับปรุงแก้ไขและการพัฒนาต่อ

- *ส่วนที่ต้องปรับปรุง*

  1. ส่วนการจำลองพฤติกรรมยังยุ่งยากจนเกินไปสำหรับพฤติกรรมที่ซับซ้อน
  2. สมรรถนะยังต่ำจนเกินไป ควรมีการจัดจังหวะการทำงานให้ดีขึ้น
  3. ส่วนการจำลองภูมิศาสตร์ยังไม่สมจริงเพียงพอต่อความต้องการ ควรมีการจัดระบบใหม่

- *ส่วนที่ควรมีการพัฒนาต่อ*

  1. ระบบทริกเกอร์เหตุการณ์
  2. การจัดการข้อมูลเกม การโหลด การเซฟเกม
  3. การติดต่อผ่านเครือข่าย
  4. ส่วนปัญญาประดิษฐ์ในการเล่นโต้ตอบกับมนุษย์

8.4 สรุปผล

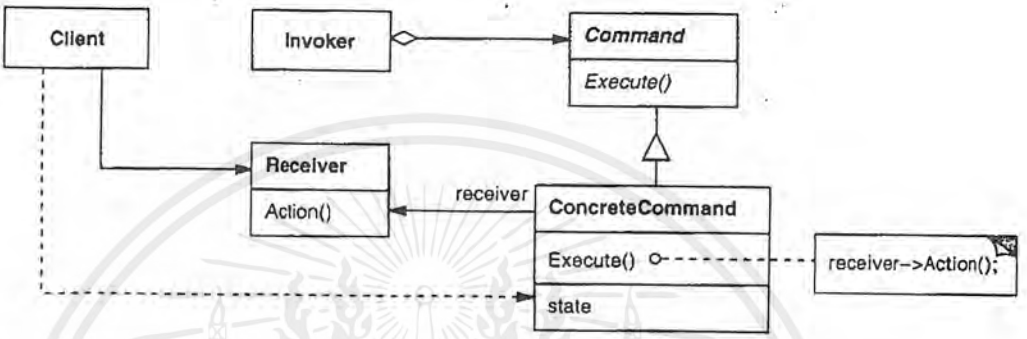
โครงงานนี้ได้บรรลุเป้าหมายตามที่กำหนดไว้แล้วและอยู่ในระดับที่น่าพอใจตามบริบทของโครงงานในระดับนี้ แต่ก็ยังมีบางจุดที่ต้องต่อเติมแก้ไขซึ่งเป็นเรื่องธรรมดา ของการพัฒนาเฟรมเวิร์ก ชิ้นงานที่ออกมานั้นสามารถนำไปใช้งานจริงได้แต่ไม่ได้อยู่ในระดับเดียวกับที่สามารถทำเชิงพาณิชย์ได้ ซึ่งก็เหมาะสมกับเงื่อนไขต่าง ๆของการพัฒนาในระดับมือสมัครเล่น ซึ่งแตกต่างไปจากเงื่อนไขของการพัฒนาระดับมืออาชีพที่คร่ำหวอดอยู่ในอุตสาหกรรมซอฟต์แวร์ประเภทเดียวกัน แนวทางการศึกษาของโครงงานนั้นมาในทิศทางที่คาดว่าถูกต้องและคาดว่าสามารถต่อยอดให้เกิดผลที่ดีขึ้นได้

ภาคผนวก ก

Design Pattern Catalog

1 Command

คือการทำการเคลือบคำร้องขอ หนึ่งให้เป็นออบเจ็กต์ ทำให้เราสามารถทำการพารามิเตอร์ ไคลเอนต์ ด้วยคำร้องขอ ที่แตกต่างกันได้ เช่นสามารถทำคิว ทำล็อก และยังสนับสนุนการทำอันดูอีกด้วย ตัวอย่างของการนำรูปแบบ Command มาประยุกต์ใช้ที่เห็นอยู่เสมอคือใช้ในทูลคิดปัญหาในการออก



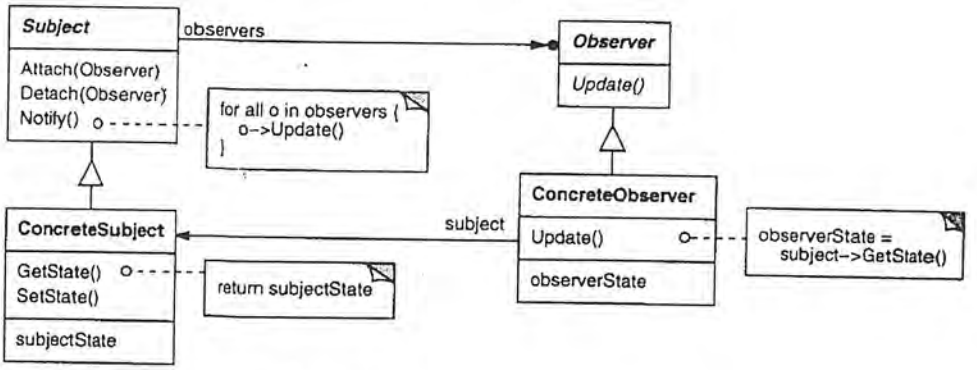
ภาพ 1a แสดงโครงสร้างของ Command แพทเทิร์น

แบบทูลคิต คือผู้ออกแบบไม่สามารถรู้ได้เลยว่า ผู้รับหรือผู้ส่งรีเควสต์จะเป็นออบเจ็กต์ที่มีลักษณะอย่างไร เมื่อเรานำ ดีไซน์แพทเทิร์น รูปแบบ Command มาใช้ทำให้ทูลคิต สร้างรีเควสต์ของแอปพลิเคชั่นที่ไม่กำหนดแน่นอนได้ โดยการเปลี่ยนรีเควสต์ของมันให้ กลายเป็นออบเจ็กต์โดยออบเจ็กต์นี้สามารถถูกส่งผ่านไปยัง ออบเจ็กต์อื่น ๆ ได้

2 Observer

คือการกำหนด ความสัมพันธ์ที่ขึ้นต่อกันแบบ 1:N ระหว่าง ออบเจ็กต์ ดังนั้นเมื่อออบเจ็กต์หนึ่งมีการเปลี่ยนแปลงสถานะ ออบเจ็กต์อื่น ๆ ที่ขึ้นต่อกันจะรับรู้และทำการอัพเดตอย่างอัตโนมัติ

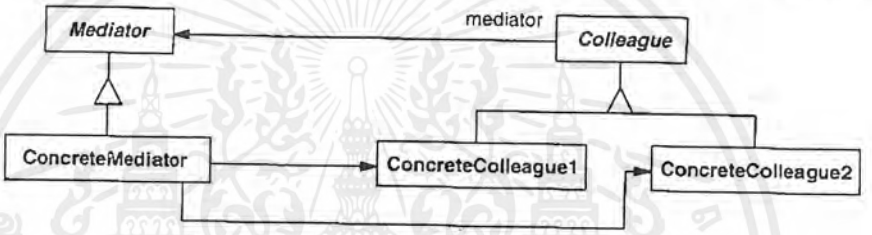
ตัวอย่างการนำไปประยุกต์ใช้ เช่นการทำระบบกราฟฟิกยูสเซอร์อินเทอเฟซ ของระบบบัญชี ตารางบัญชีกับแต่ละหน้าต่างที่แสดงกราฟต่าง ๆ ของ ตารางบัญชีนั้น ๆ ไม่รู้จักกันแต่เมื่อมีการทำการเปลี่ยนแปลงในตารางบัญชีนั้น ข้อมูลกราฟต้องแสดงข้อมูลของบัญชีนั้นที่เปลี่ยนแปลงอย่างถูกต้อง เราสามารถนำ ดีไซน์แพทเทิร์น แบบ Observer มาใช้โดยกำหนดความสัมพันธ์ของ ออบเจ็กต์ให้เป็น Subject และ Observer โดย Subject จะมี Observer จำนวนหนึ่งซึ่งขึ้นอยู่ต่อกัน เมื่อมีการเปลี่ยนแปลงใน Subject แต่ละ Observer ใด ๆจะถูกแจ้งให้ทราบถึงความเปลี่ยนแปลงนั้นและจะตอบสนองด้วนการทำการถาม Subject ถึง สถานะปัจจุบันเพื่อนำไปทำให้สถานะตรงกัน



ภาพที่ 2a แสดงโครงสร้างของ Observer แพทเทิร์น

3 Mediator

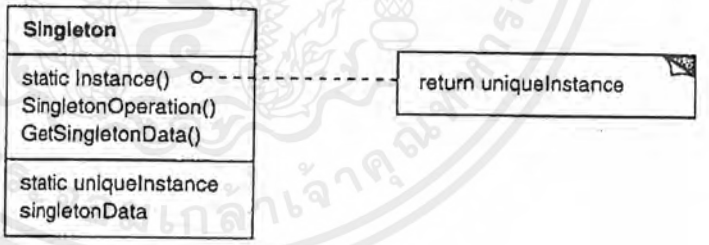
ทำการกำหนดออบเจ็กต์ซึ่งทำการเอนแคปซูเลตเซตของการปฏิสัมพันธ์กันของออบเจ็กต์ Mediator สนับสนุนการแยกคู่ออบเจ็กต์ โดยการรักษาออบเจ็กต์จากการเกี่ยวข้องถึงออบเจ็กต์อื่น ๆ อย่างชัดแจ้ง



ภาพที่ 3a แสดงโครงสร้างของ Mediator Patter

4 Singleton

ทำการรับรองว่าแต่ละคลาสจะมีอินสแตนซ์เพียงตัวเดียว และยังทำให้สามารถเข้าถึงได้อย่าง

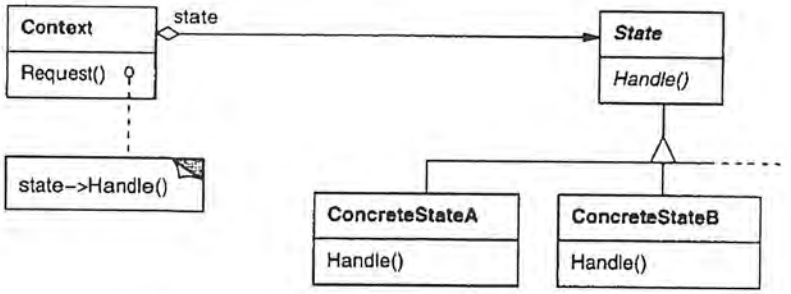


โกลบอล

ภาพที่ 4a แสดงโครงสร้างของ Singleton แพทเทิร์น

5 State

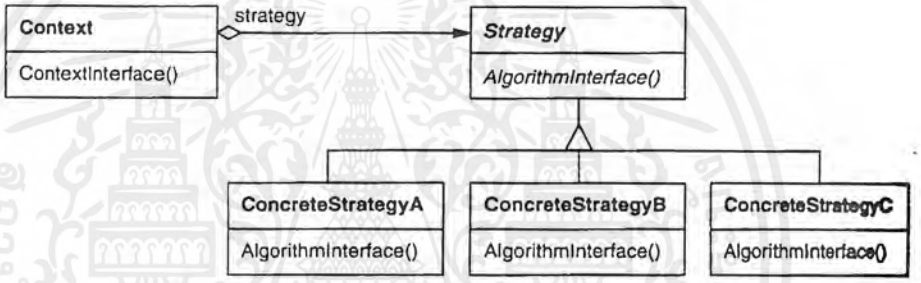
ทำให้ออบเจ็กต์หนึ่งเปลี่ยนแปลงพฤติกรรมเมื่อสถานะภายในเปลี่ยน ออบเจ็กต์จะแสดงตัวเปลี่ยนคลาสของมัน



ภาพที่ 5a ก แสดงโครงสร้างของ State แพทเทิร์น

6 Strategy

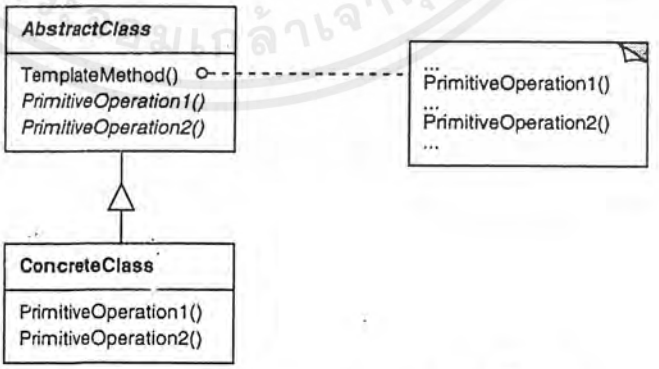
ทำการกำหนดชนิดของอัลกอริทึม และเอนแคปซูเลตแต่ละชนิดไว้ และทำให้มันสามารถถูกสับเปลี่ยนใช้ได้ Strategy ยังทำให้อัลกอริทึมเปลี่ยนแปลงอย่างอิสระจากไคล์เอนที่ใช้มัน



ภาพที่ 6a แสดงโครงสร้างของ Strategy แพทเทิร์น

7 Template Method

กำหนดโครงร่างของ อัลกอริทึมหนึ่งในโอเปอร์เรชันหนึ่ง ทำการรอให้ สับคลาสเป็นผู้กำหนดมันเอง Template Method ทำให้สับคลาสสามารถกำหนดบางขั้นของ อัลกอริทึมหนึ่งๆได้โดยไม่ต้องเปลี่ยนแปลงโครงสร้างของอัลกอริทึม



ภาพที่ 7a แสดงโครงสร้างของ Template Method แพทเทิร์น

อ้างอิง

จาก Design Pattern Catalog [1] Gamma Erich. (GoF95)

บรรณานุกรม

[1] Gamma, Erich. "Design patterns elements of reusable object-oriented software /Erich Gamma ... [et al.] (Gof95)" Reading, MA : Addison-Wesley, c1995.

[2] Andrew Rollings , Dave Morris "GAME ARCHITECTURE AND DESIGN" CORIOLIS, c2000.

[3] Jerry Banks, John S. Carson II, Barry L.Nelson "Discrete-Event System Simulation" Pentice-Hall Inc, c1996

[4] David Blackledge "A Case Study: Designing a Discrete Event Simulation" http://www.unm.cs.edu/~david/unm/

[5] Dattatri, Kayshav. "C++ : effective object-oriented software construction : concepts, principles, industrial strategies, and practices / Kayshav Dattatri." Upper Saddle River, NJ : Prentice Hall PTR, c1997.

[6] Bret Timmins. "DIRECT DRAW PROGRAMMING" NY : M&T Books, c1996

[7] James Rumbaugh, Ivar Jarcobson, Grady Booch "The Unified Modeling Language Reference Manual" Addison Wesley Longman , Inc c1999

[8] Meilir Page Jones "Fundamental Object-Oriented Design in UML" Addison Wesley Longman , Inc c2000

[9] Johannes Weidi "The Standard Template Library Tutorial" Information System Institute , Distributed Systems Department , Technical University Vienna c1996