เกมแข่งรถสามมิติผ่านระบบเครือข่าย

3D RACING NETWORK GAME

โดย

ปิติ ศุภนิมิตวาสนา

เลิศวุฒิ วีระธรากุล

อาจารย์ที่ปรึกษา

อาจารย์ ณฐพล พันธุวงศ์

อาจารย์ที่ปรึกษาร่วม

ผศ.ดร. ธนารัตน์ ชลิดาพงศ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรบัณฑิต

สาขาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ภาคเรียนที่ 2 ปีการศึกษา 2550

3D RACING NETWORK GAME

PITI SUPANIMITWASANA

LERTWUT WEERATARAKUL

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE PROGRAM IN INFORMATION TECHNOLOGY

FACULTY OF INFORMATION TECHNOLOGY

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

2 / 2007

ใบรับรองปริญญานิพนธ์ ประจำปีการศึกษา 2550

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง

เกมแข่งรถสามมิติผ่านระบบเครือข่าย

3D Network Racing Game

ผู้จัดทำ

1. นาย ปิติ ศุภนิมิตวาสนา รหัสประจำตัว 47070075

2. นาย เลิศวุฒิ วีระธรากุล รหัสประจำตัว 47070099

.....อาจารย์ที่ปรึกษา

(อาจารย์ ณฐพล พันธุวงศ์)

.....อาจารย์ที่ปรึกษาร่วม

(รศ.ดร. ธนารัตน์ ชลิดาพงศ์)

หัวข้อ	เกมแข่งรถผ่านระบบเครือข่าย	
นักศึกษา	นายปีติ ศุภนิมิตวาสนา	47070075
	นายเลิศวุฒิ วีระธรากุล	47070099
ปริญญา	วิทยาศาสตรบัณฑิต	
สาขาวิชา	เทคโนโลยีสารสนเทศ	
ปีการศึกษา	2550	
อาจารย์ที่ปรึกษา	อ. ณฐพล พันธุวงศ์	
	ผศ.ดร. ธนารัตน์ ชลิดาพงศ์	

บทคัดย่อ

โครงงานนี้ได้จัดทำเกมแข่งรถสามมิติที่มีการใช้เครื่องมือทางฟิสิกส์มาประกอบเพื่อให้เกิดความสมจริงให้มากที่สุดทั้งยังสามารถทำให้เล่นผ่านระบบเครือข่ายได้ซึ่งพัฒนาโดยใช้ภาษาซีพลัสพลัสและเครื่องมือโอเกอร์ในการพัฒนาเกมซึ่งสามารถทำให้เกมสามารถจัดการทรัพยากรได้ง่ายมีการประยุกต์ใช้เอ็กซ์เอ็มแอลไฟล์ในการกำหนดค่าของตัวละครและสนามเพื่อให้ง่ายต่อการแก้ไขและสามารถสื่อความหมายได้ดีขึ้น อีกทั้งทางด้านระบบเครือข่ายโดยใช้รูปแบบของลูกข่าย-แม่ข่ายและได้มีการใช้เครื่องมือแรคเน็ตมาจัดการซึ่งจะทำให้การส่งข้อมูลทำได้รวดเร็วขึ้นมีเฮดเดอร์น้อยลงเนื่องจากใช้โปรโตคอลยูดีพีในการส่งและมีการปรับปรุงเพื่อลดเฮดเดอร์ลงอีก ทำให้การเล่นเกมมีความต่อเนื่องมากขึ้น

Title	3D Racing Network Game	
Student	Mr. Piti Supanimitwassana	47070075
	Mr. Lertwut Weeratarakul	47070099
Degree	Bachelor of Science	
Programme	Information Technology	
Academic Year	2007	
Project Advisor	Mr. Natapon Pantuwong	
	Asst. Prof. Dr. Thanarat Chalidabhongse	

ABSTRACT

This project is 3D Racing Game which use physic engine to provide the most reality physic in car and support to play on network system. This project is developing with C++ programming and OGRE Game Engine. The OGRE Game Engine provide to easy to manage resource. This project has using XML to apply the configuration of character and circuit that can provide easy to change and the more meaning. The Network System use Client-Server model which manage by Raknet engine that utilize the packet which make the less packet header that can make the game faster.

กิตติกรรมประกาศ

โครงงานนี้ไม่อาจสำเร็จได้ด้วยดีหากขาดความกรุณาจากอาจารย์ที่ปรึกษา อาจารย์ณฐพล พันธุวงศ์ และผศ.ดร. ธนารัตน์ ชลิดาพงศ์ ที่ได้สละเวลาให้ความช่วยเหลือ แนะนำในการปรับปรุงแก้ไขปัญหาต่างๆ มาโดยตลอด

ขอขอบคุณ อาจารย์ทุกท่านที่ได้ให้ความรู้ที่มีประโยชน์ ช่วยให้สามารถแก้ไขปัญหาต่างๆ ให้สำเร็จลุล่วงไปได้ และเป็นตัวอย่างที่ดีในการศึกษาเล่าเรียนและการทำงาน

ขอขอบคุณ นางสาวธัญสินี ต่อวงศ์ไพชยนต์ที่ถึงแม้จะไม่สามารถร่วมทำโครงงานได้ตลอดทั้งโครงงาน แต่สำหรับแรงขับเคลื่อน ความรู้ และกำลังใจสำหรับเพื่อนได้ทำให้โครงงานนี้สำเร็จลุล่วงตามที่ได้หวังไว้

ขอขอบคุณ คณะเทคโนโลยีสารสนเทศสถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังที่เอื้ออำนวยสภาพแวดล้อมต่างๆ ในการทำโครงงาน

สุดท้ายนี้ขอขอบคุณเพื่อนคณะเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังทุกคนที่ได้ให้การช่วยเหลือและกำลังใจเพื่อให้โครงงานชิ้นนี้ได้สำเร็จโดยสมบูรณ์

ปิติ ศุภนิมิตวาสนา
เลิศวุฒิ วีระธรากุล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญตาราง (ต่อ)

ตารางที่ หน้า

สารบัญรูป

สารบัญรูป (ต่อ)

รูปที่ หน้า

สารบัญรูป (ต่อ)

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ปัจจุบันเกมเชื่อมตรงเป็นเกมประเภทใหม่ที่สามารถมาครองตลาดเกมไม่เชื่อมตรงได้เป็นอย่างมาก เนื่องจากเป็นเกมที่ผู้เล่นสามารถเล่นพร้อมกันกับผู้เล่นคนอื่นได้ผ่านระบบเครือข่ายที่มีการพัฒนาขึ้นอย่างรวดเร็ว นอกจากนี้ยังเป็นช่องทางในการสื่อสารช่องทางหนึ่งที่ได้รับความนิยมในปัจจุบัน การเติบโตของธุรกิจเกมออนไลน์นี้เองจึงเป็นสาเหตุให้มีการพัฒนาเทคโนโลยีในการสร้างแอพพลิเคชันแบบสถาปัตยกรรมรับ-ให้บริการ และคอมพิวเตอร์กราฟฟิคขึ้นมาอย่างมากมายในปัจจุบัน

โครงงานนี้เป็นโครงงานที่ได้ทำการพัฒนาเกมแบบสามมิติแบบเชื่อมตรงที่ให้ผู้เล่นสามารถเข้าเล่นเกมพร้อมกันได้หลายคนผ่านระบบเครือข่าย ซึ่งจะได้ศึกษาและพัฒนาเกมสามมิติโดยมีหัวข้อที่จะศึกษาดังนี้

1 โครงสร้างพื้นฐานของกลไกเกม
2 การจำลองภาพสามมิติ
3 เสียงสามมิติ
4 สถาปัตยกรรมรับ-ให้บริการ
5 การออกแบบระบบเชิงวัตถุ ที่สามารถตอบสนองการทำงานแบบทันที

โดยในการศึกษานี้จะได้ทำการศึกษาถึงเทคนิคต่าง ๆ ในแต่ละหัวข้อและเลือกใช้เทคนิคที่เหมาะสมกับเกมที่ได้พัฒนาขึ้น

1.2 วัตถุประสงค์ของโครงงาน

1 เพื่อศึกษาหลักการในการแสดงผมภาพ3มิติ และเทคนิคต่าง ๆ ที่สร้างภาพเหล่านี้ให้เสมือนจริง
2 เพื่อศึกษาหลักการสร้างการเคลื่อนไหวในระบบ 3 มิติ
3 เพื่อประยุกต์ใช้การออกแบบระบบเชิงวัตถุในระบบแบบทันที
4 เพื่อศึกษาและสามารถพัฒนาระบบเพื่อรองรับการทำงานแบบรับ-ให้บริการ

1.3 ขอบเขตของโครงงาน

โครงงานนี้เป็นการศึกษาการสร้างและพัฒนาเกมสามมิติที่สามารถเล่นผ่านระบบเครือข่ายได้ โดยผู้เล่นสามารถเข้าเล่นได้พร้อมๆกันหลายคน และได้ใช้ระบบการคำนวณทางฟิสิกส์ เพื่อให้เกมมีความสมจริงมากขึ้น

บทที่ 2
ทฤษฎีและเทคโนโลยีที่เกี่ยวข้อง

2.1 คอมพิวเตอร์กราฟฟิค

คอมพิวเตอร์กราฟิคเป็นศาสตร์ที่ช่วยให้เกิดยุคใหม่ของหลายๆ สิ่งไม่เว้นแม้แต่เกม ซึ่งสามารถนำไปรวมกับศาสตร์อื่นเพื่อให้เกิดสิ่งใหม่ขึ้นมาได้อย่างมากมาย

สิ่งแรกก็คือการแสดงผลของข้อมูลสถิติต่างๆ สามารถใช้แสดงให้เห็นเป็นความสัมพันธ์โดยใช้ภาพในการสื่อถึงข้อมูล ทางการแพทย์ก็มีการใช้เพื่อจำลองการทดลองต่างๆ ให้ดูเสมือนจริงและสามารถนำไปใช้จริงได้ ทางด้านเกมก็เป็นอีกศาสตร์หนึ่งที่คอมพิวเตอร์กราฟิค ได้ทำให้เกิดการเปลี่ยนแปลงอย่างมาก และยังเป็นส่วนสำคัญอีกด้วย หากเกมไม่มีภาพแสดงให้ดู ถ้าในสมัยนี้คงเรียกได้ว่าไม่ใช่เกม จากการแสดงผลของภาพซึ่งสามารถแบ่งออกได้เป็นสองอย่างคือ เกมสองมิติ ซึ่งก็คือเกมที่เราสามารถมองตัวละครได้แค่ด้านเดียว ไม่สามารถหมุนได้ ซึ่งใช้ระบบเลเยอร์ในการจัดการ ในการทำให้เคลื่อนไหว ทำโดยการเปลี่ยนภาพไปเรื่อยๆ ซึ่งเป็นวิธีเดียวกับการทำภาพเคลื่อนไหวในอดีต อย่างที่สองก็คือเกมสามมิติ ซึ่งก็คือเกมที่สามารถหมุนดูได้รอบตัวละคร ซึ่งจะช่วยเพิ่มความสมจริงในเกมได้ และอาจมีการใช้แสง เงา ซึ่งเป็นการประมวลผลชั้นสูงมาด้วย จะทำให้เราเหมือนได้อยู่ในโลกของเกมจริงๆ สามารถมองเห็นในมุมที่ตัวละครเห็น ซึ่งใช้การมองแบบสัดส่วนจริง (perspective) คือถ้าวัตถุที่มองอยู่ไกลก็จะเห็นเล็กลง แต่ถ้าวัตถุอยู่ใกล้ก็จะมองใหญ่ขึ้น การเคลื่อนไหวของตัวละครแต่ละตัวก็จะต้องมีความสมจริง ซึ่งได้ใช้ศาสตร์ตามแนวคิดของสรีระของคนโดยการสร้างกระดูก ส่วนยึดต่างๆ ให้เหมือนคนมากที่สุด เพื่อจะได้สามารถทำให้เคลื่อนไหวได้เหมือนการเคลื่อนไหวได้เหมือนคนเช่น การเดิน การพูด การปลิวของเส้นผม อย่างสุดท้ายที่ตัวละครในเกมจะขาดไม่ได้ก็คือเสื้อผ้าซึ่งอาจจะต้องทำเป็นสามมิติ แต่ก็ได้มีเทคนิคให้สามารถใช้ภาพสองมิติเพื่อไปสวมให้กับตัวละครในสามมิติได้ ซึ่งเราสามารถเลือกคุณสมบัติของวัตถุได้ เช่น แก้วก็ต้องมีความวาว ผ้าไม่สะท้อนแสง การเปลี่ยนแปลงยุคต่อไปของเกมก็จะเป็นการเล่นกันผ่านระบบเน็ตเวิร์ค ซึ่งได้ใช้ระบบเน็ตเวิร์คที่เติบโตขึ้นทุกวันทำให้เกมมีความสามารถเพิ่มขึ้น แต่เกมสามมิติยังมีข้อจำกัดในเรื่องการส่งข้อมูลที่มีจำนวนมาก ซึ่งทำให้ต้องใช้ระบบเน็ตเวิร์คที่มีความมั่นคงและต้องใช้ขนาดของโมเดลให้เล็กที่สุดเท่าที่จะเป็นไปได้

คอมพิวเตอร์กราฟิคช่วยให้สิ่งต่างๆ ให้สามารถมองเห็นได้ง่ายขึ้น เข้าถึงได้ง่ายขึ้น และมีแนวโน้มที่จะพัฒนาต่อไปเรื่อยๆ ซึ่งเมื่อศาสตร์นี้มีการพัฒนาก็จะทำให้สิ่งต่างๆ มีการพัฒนาตามไปด้วย จึงกล่าวได้ว่า คอมพิวเตอร์กราฟิคเป็นสิ่งที่ช่วยเกื้อหนุนศาสตร์ต่างๆ ให้มีประสิทธิภาพมากขึ้น

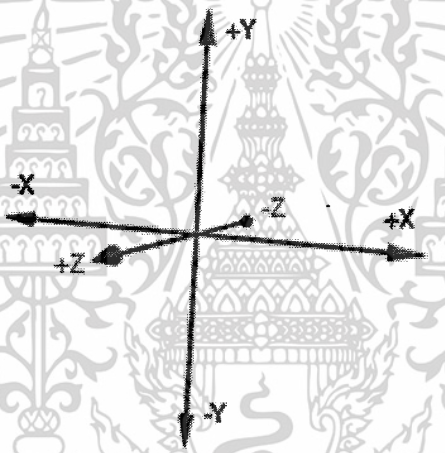
2.2 ความรู้ทั่วไปเกี่ยวกับคอมพิวเตอร์กราฟฟิค

แกนในระบบ 3 มิติ

ระบบแกนแบบคาร์ติเซียน (Cartesian) ได้นิยามเส้นตั้งฉากสามเส้นประกอบด้วยแกนสามแกนที่จะแทนด้วยตัวอักษร x y และ z แต่ละเส้นจะมีความยาวไปถึงอินฟินิติ แกนแต่ละเส้นจะพบกันที่จุดออริจิน (Origin) หรือแทนด้วนจุด <0, 0, 0> แกน x และแกน y จะทำมุมตั้งฉากกันขนาด 90 องศา

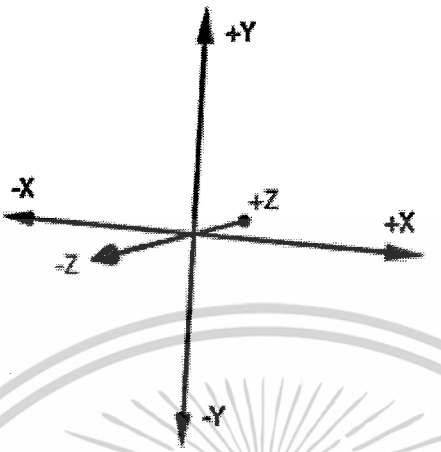
ระบบแกนสามมิติในคอมพิวเตอร์กราฟฟิกส่วนใหญ่จะแบ่งออกเป็น 2 ประเภท

1 ระบบแกนมือซ้าย ในระบบนี้แกน x และแกน y จะทำมุมตั้งฉากกันขนาด 90 องศา แกน z จะมีค่าเป็นบวกเมื่อมีทิศทางพุ่งออกนอกจอและเป็นลบในทิศทางตรงกันข้าม ดังรูป



รูปที่ 2.1 ระบบแกนแบบมือซ้าย

2 ระบบแกนมือขวา ในระบบนี้แกน x และแกน y จะทำมุมตั้งฉากกันขนาด 90 องศา แต่แกน z จะมีค่าเป็นลบเมื่อมีทิศทางพุ่งออกนอกจอและเป็นบวกในทิศทางตรงกันข้าม ดังรูป



รูปที่ 2.2 ระบบแกนแบบมือขวา

จุด

จุดในแกนสามมิติ หรือ ที่เรียกว่าเวคเตอร์ เป็นสิ่งที่สำคัญที่สุดในการคำนวณกราฟฟิคสามมิติ สามารถนำมาแทนจุดต่างๆบนวัตถุ การวางตำแหน่งของวัตถุ มุมที่วัตถุหันหน้า หรืออาจเป็นความเร็วและความเร่งของวัตถุนั้นๆ

วัตถุสามมิตินั้นประกอบด้วยความกว้าง ความยาว และความลึก ซึ่งทั้งหมดได้ถูกแทนด้วยตัวประกอบ 3 ตัว (Shorthand component) คือ x แทน ความกว้าง, y แทน ความยาว, z แทน ความลึก จุดและเวคเตอร์เมื่อถูกใช้ในการคำนวณจะมีความหมายถึงปริมาณของเวคเตอร์ ส่วนตัวเลข (regular numbers) จะมีความหมายถึงปริมาณสเกลล่า การเขียนสามารถเขียนเป็นตัวประกอบสามจำนวนเรียกกัน ดังเช่น <x, y, z> หรืออาจเขียนอยู่ในรูปเมทริกซ์หนึ่งแถว หรือเมทริกซ์หนึ่งหลัก เช่น V = [Vx Vy Vz]

ขนาด

จุดแต่ละจุด สามารถบ่งบอกถึงระยะทางระหว่างจุดนั้นและจุดออริจิน (Origin) ได้ ซึ่งระยะทางนี้ถูกเรียกว่า ขนาดของเวคเตอร์ ซึ่งถ้าหากคำนวณในระบบสองมิติแล้วจะใช้ทฤษฎีบทพีทากอรัส (Pythagorean Theorem) ดังสมการ

$$\|V\| = \sqrt{x^2 + y^2} \qquad (2.1)$$

แต่สำหรับการคำนวณในระบบสามมิติแล้วสามารถคำนวณได้โดยการเพิ่มตัวประกอบ z ลงในสมการพีทากอรัส ดังสมการ

$$\|V\| = \sqrt{x^2 + y^2 + z^2} \tag{2.2}$$

ในทางกลับกันสังเกตได้ว่า สมการพีทากอรัสในระบบสองมิตินั้น คือ สมการพีทากอรัสในระบบสามมิติที่มีตัวประกอบ z เท่ากับศูนย์

เวคเตอร์หนึ่งหน่วย

เวคเตอร์หนึ่งหน่วย คือ เวคเตอร์ใดที่มีจุดปลายข้างหนึ่งอยู่บนจุดออริจิน ส่วนอีกข้างหนึ่งสัมผัสกับทรงกลมหนึ่งหน่วย(Unit sphere) โดยเวคเตอร์หนึ่งหน่วยสามารถหาได้จาก การนำเวคเตอร์หารด้วยขนาดของเวคเตอร์ของมันเอง ดังสมการ

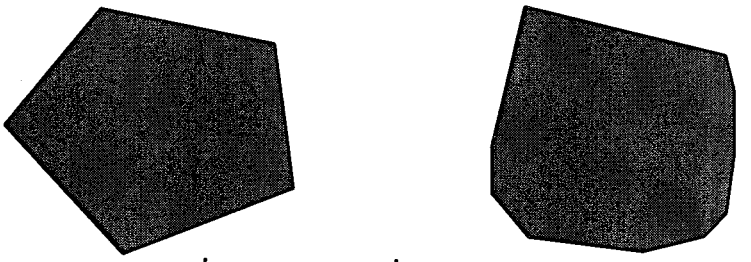
$$n = \frac{m}{\|m\|} \tag{2.3}$$

รูปหลายเหลี่ยม

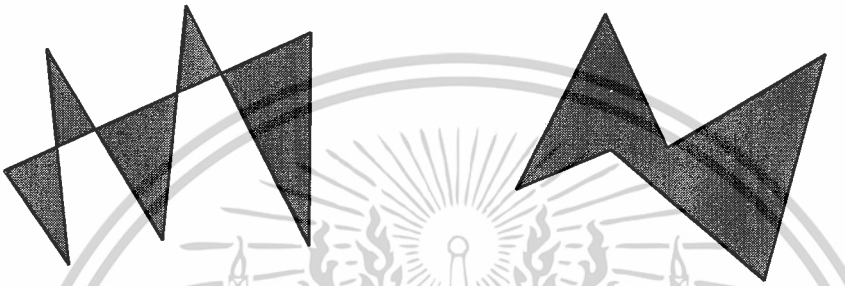
รูปทรงหลายเหลี่ยม(Polygons) เป็นสิ่งที่มีประโยชน์มากในการสร้างภาพในคอมพิวเตอร์กราฟฟิค ซึ่งในปัจจุบันคอมพิวเตอร์ทั่วไปจะมี hardware ที่ช่วยเร่งความเร็วในการประมวลผลการวาดรูปทรงหลายเหลี่ยมให้อยู่แล้ว

รูปทรงหลายเหลี่ยมในที่นี้จะพูดถึงรูปหลายเหลี่ยมที่มีคุณสมบัติคอนเว็ค (Convex) เท่านั้น คุณสมบัติที่จะทำให้เป็นคอนเว็คได้นั้นขอบของรูปทรงสี่เหลี่ยมจะต้องไม่มีมุมภายในที่เป็นมุมแหลม ซึ่งรูปทรงหลายเหลี่ยมที่มีคุณสมบัติคอนเว็คจะง่ายในการ rasterization, clip, cull และอื่นๆ

ส่วนรูปทรงหลายเหลี่ยมที่ไม่มีคุณสมบัติคอนเว็คจะเรียกว่ารูปทรงหลายเหลี่ยมที่มีคุณสมบัติคอนเคป ซึ่งการเขียนโปรแกรมเพื่อที่จะจัดการนั้นยากกว่ากรณีที่เป็นคอนเว็คมาก ดังนั้นหากจำเป็นจริงๆที่จะต้องใช้รูปทรงหลายเหลี่ยมที่มีคุณสมบัติคอนเคปก็จะต้องทำการแยกรูปทรงหลายเหลี่ยมที่มีคุณสมบัติคอนเคปออกเป็นหลายๆรูปทรงหลายเหลี่ยมที่มีคุณสมบัติคอนเว็ค



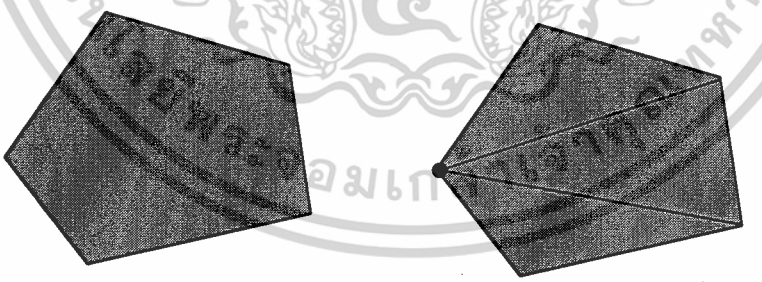
รูปที่ 2.3 รูปหลายเหลี่ยมแบบคอนเวค



รูปที่ 2.4 รูปหลายเหลี่ยมแบบคอนเคฟ

สามเหลี่ยม

รูปทรงสามเหลี่ยมในปัจจุบันนั้นเป็นรูปทรงพื้นฐานที่สุดในการวาดรูปทรงที่มีความซับซ้อน ซึ่งจริงๆแล้วการวาดโพลีกอนนั้นส่วนอุปกรณ์จะทำการวาดรูปสามเหลี่ยมที่มีจุดหนึ่งจุดที่ใช้ร่วมกัน (triangle fan)



รูปที่ 2.5 รูปหลายเหลี่ยม (รูปซ้าย) และการเปลี่ยนเป็นสามเหลี่ยม

เมื่อเราทำการสร้างแต่ละเส้นของรูปทรงสามเหลี่ยมลำดับของแต่ละจุดนั้นก็เป็นสิ่งที่สำคัญ เมื่อทำการมองในตำแหน่งของกล้องแล้วถ้าลำดับของแต่ละจุดในสามเหลี่ยมเรียงไปในแนวตามเข็มนาฬิกาจะทำให้สามารถมองเห็นได้แต่หากเรียงไปในแนวทวนเข็มนาฬิกาจะทำให้ไม่สามารถมองเห็นได้ (โดยการคำนวณหาระนาบ(Plane) ของสามเหลี่ยม และทำการดอทโพรดักด้วยเวคเตอร์จากตำแหน่งของกล้องไปสู่ตำแหน่งของเพลนนั้น)

การเปลี่ยนแปลงตำแหน่งของวัตถุบนระนาบ 2 มิติ (2D Object transformation)

การเคลื่อนย้ายวัตถุ (Object translation)

การย้ายตำแหน่งคือการทำให้วัตถุเปลี่ยนตำแหน่งจากที่หนึ่งไปสู่ที่ที่หนึ่ง ซึ่งทำได้โดยการบวกค่าระยะทาง tx, ty ให้กับตำแหน่งเดิมที่เราอยู่ (x,y) เพื่อไปยังตำแหน่งใหม่ (x',y') จะได้

$$x' = x + tx \tag{2.4}$$

และ

$$y' = y + ty \tag{2.5}$$

ในการย้ายตำแหน่งเราเรียกระยะทางที่เราใช้บวกเพิ่ม (tx,ty) ว่า เวกเตอร์พา (translation vector) หรือ เวกเตอร์เลื่อน (shift vector)

จากสมการที่ได้มาเราสามารถเขียนในรูปของเมตริกได้ดังนี้

ให้ T เป็น เมตริกของการย้ายตำแหน่ง

P เป็น เมตริกของตำแหน่งเดิม

P' เป็น เมตริกของตำแหน่งหลังจากย้ายตำแหน่งแล้ว

$$P' = \begin{bmatrix} x' \\ y' \end{bmatrix} \quad P = \begin{bmatrix} x \\ y \end{bmatrix} \quad T = \begin{bmatrix} tx \\ ty \end{bmatrix} \tag{2.6}$$

จะได้ว่า P' = P + T ซึ่ง P' คือ ค่าของ x,y ที่เกิดขึ้นหลังจากการบวกค่าตำแหน่งเดิมเข้ากับค่าที่เพิ่มเพื่อการเคลื่อนย้ายตำแหน่ง

การหมุนวัตถุ (Rotation)

การหมุนในแกนสองมิติคือการเปลี่ยนตำแหน่งบนเส้นทางเป็นวงในแกน xy ซึ่งจะต้องระบุขนาดมุมที่จะหมุนและจุดหมุน (rotation point or pivot point) ที่จะให้วัตถุหมุนรอบจุดหมุนนั้น ถ้าค่ามุมเป็นบวกจะทำให้วัตถุหมุนในทิศทวนเข็มนาฬิกา หากค่ามุมเป็นลบวัตถุก็จะหมุนในทิศตามเข็มนาฬิกา และยังมีการหมุนอีกอย่างก็คือการหมุนแกน ซึ่งคือการหมุนโดยใช้แกนเป็นจุดหมุน

สมการการหมุนของจุดที่ตำแหน่ง P เมื่อจุดหมุนคือจุด (0,0) และ r คือระยะทางจากจุดที่จะทำการหมุน กับจุด (0,0) และมุม Φ คือมุมเดิมที่วัตถุอยู่โดยคิดเริ่มจากแนวนอน และ θ คือขนานมุมที่ให้วัตุหมุนหมุน จะได้สมการดังนี้

$$\begin{aligned} x' &= r\cos(\Phi+\theta) = r\cos\Phi\cos\theta - r\sin\Phi\sin\theta \\ y' &= r\sin(\Phi+\theta) = r\cos\Phi\sin\theta + r\sin\Phi\cos\theta \end{aligned} \tag{2.7}$$

ตำแหน่งของ x และ y ในแนวเชิงขั้วสามารถเขียนได้เป็น

$$x = r\cos\Phi, \quad y = r\sin\Phi \tag{2.8}$$

นำค่าไปแทนในสมการด้านบน จะได้สมการของการหมุนของจุดที่ตำแหน่ง (x,y) เป็นมุม θ โดยมีจุด (0,0) เป็นจุดหมุน

$$\begin{aligned} x' &= x\cos\theta - y\sin\theta \\ y' &= x\sin\theta + y\cos\theta \end{aligned} \tag{2.9}$$

ด้วยสมการข้างบนเราสามารถเขียนสมการของการหมุนเป็นแบบเวกเตอร์ได้ดังนี้

$$P' = R \cdot P \tag{2.10}$$

โดยที่ R คือ เมตริกซ์การหมุน

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.11}$$

จากสมการการหมุนรอบจุด (0,0) มาสร้างความสัมพันธ์ได้สมการที่ใช้ในการหมุนรอบจุดใดๆ(xr,yr) ได้ดังนี้

$$\begin{aligned} x' &= x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta \\ y' &= y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta \end{aligned} \tag{2.12}$$

ในการหมุนรอบจุดหมุนใดๆ ทำได้อีกวิธีคือ การ translate ให้จุดหมุนที่ต้องการมาที่จุด (0,0) จากนั้นก็ทำการหมุน และก็ translate กลับมาที่จุดเดิม

การเปลี่ยนแปลงขนาดของวัตถุ (Object Scaling)

การปรับเปลี่ยนขนาดสามารถทำได้โดยการคูณตำแหน่งเดิม (x,y) ด้วยค่าขยาย sx และ sy เพื่อได้ตำแหน่งของจุดหลังขยาย (x',y')

$$x' = x \cdot s_x \qquad y = y \cdot s_y \tag{2.13}$$

โดยที่ sx คือค่าขยายในแนวแกน x และ sy คือค่าขยายในแนวแกน y จากสมการสามารถเขียนในรูปเมตริกได้ดังนี้

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.14}$$

หรือ

$$P' = S \cdot P \tag{2.15}$$

การปรับเปลี่ยนขนาดถ้าเวกเตอร์ในการขยายเป็นค่าบวกคือการขยายปกติ แต่ถ้าเป็นค่าลบวัตถุก็จะพลิกกลับข้างจะได้วัตถุเหมือนส่องกระจก เช่นถ้า sx เป็น -1 วัตถุก็ถูกพลิกกลับในแนวแกน x

การแสดงตำแหน่งโดยใช้เมทริกซ์และระบบแกนโฮโมจินัส (Matrix Representations and Homogeneous Coordinates)

คือการทำให้สามารถใช้ พิกัดเดียว ในการแสดงการ ย้ายตำแหน่ง การหมุน และการปรับเปลี่ยนขนาดได้ในการย้ายตำแหน่งจะได้

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.16}$$

สามารถเขียนในรูปย่อได้เป็น

$$P' = T(tx,ty) \cdot P \tag{2.17}$$

ในการหมุนรอบจุด (0,0) จะได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.18}$$

สามารถเขียนเป็นรูปย่อได้เป็น

$$P' = R(\theta) \cdot P \tag{2.19}$$

ในการเปลี่ยนขนาดจะได้เมตริกดังนี้

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.20}$$

เขียนในรูปย่อได้เป็น

$$P' = S(s_x,s_y) \cdot P \tag{2.21}$$

การสะท้อนวัตถุ (Object Reflection)

คือการสร้างภาพเหมือนเราส่องกระจกของวัตถุที่ต้องการ ในสองมิติเราจะสร้างแกนของการสะท้อน คือการหมุน 180 องศารอบแกนที่ต้องการที่จะทำให้เกิดภาพสะท้อน ใช้เทคนิคเหมือนการเปลี่ยนขนาดภาพ แต่คูณด้วย -1 แทน เช่นต้องการ การสะท้อนในแนวแกน x ซึ่งสามารถใช้ $\begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ มาคูณเข้ากับวัตถุที่ต้องการได้ ในแกน y ก็ใช้ $\begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ แทน

การบิดวัตถุ (Object Shear)

คือการทำให้วัตถุบิดเบือนไปในทางการเลื่อนขนานกับอีกด้านของวัตถุ เมตริกที่ใช้ในการทำเล็มในแนวแกน x คือ $\begin{bmatrix} 1 & sh_x & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ แนวแกน y คือ $\begin{bmatrix} 1 & 0 & 0 \\ sh_y & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

การเปลี่ยนแปลงวัตถุในระบบ 3 มิติ (Three-Dimensional Geometric Transformations)

ในการทำการเปลี่ยนแปลงทางเรขาคณิตในสามมิติจะคล้ายกับในสองมิติแต่มีแกน z เพิ่มขึ้นมาซึ่งจะบอกถึงว่าวัตถุนั้นอยู่ตรงไปข้างหน้ามากแค่ไหน

การเคลื่อนย้ายวัตถุ (Object translation)

ในการย้ายที่ในแกนสามมิติจะใช้เมตริกของการย้ายที่เป็นดังนี้

$$T=\begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \text{ และ } P=\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, P'=\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} \quad (2.22)$$

สามารถเขียนให้อยู่ในรูปของเวกเตอร์ดังนี้

$$P' = T \cdot P \quad (2.23)$$

การหมุนวัตถุ (Object rotation)

ในการหมุนในแกนสามมิติของออปเจ็คเราต้องเลือกแกนที่จะทำการหมุนและองศาในการหมุน ซึ่งต่างจากในสองมิติที่การเปลี่ยนแปลงทั้งหมดอยู่ในแกน xy แต่ในการหมุนในสามมิติสามารถทำได้ในทุกๆ เส้นในมิติ วิธีการหมุนที่ง่ายที่สุดคือการหมุนตามแกน ซึ่งเราสามารถในการหมุนรอบแกน บวกกับการย้ายที่ในการทำแทนการหมุนในแนวต่างๆ ได้

การหมุนในแนวตามแกน(Coordinate-Axis Rotations)

การหมุนรอบแกน z มีสมการดังนี้

$$x' = x\cos\theta - y\sin\theta$$
$$y' = x\sin\theta + y\cos\theta \quad (2.24)$$

$$z' = z$$

โดยที่ θ แทนขนาดของมุมในการหมุน ในรูปแบบของพิกัดมาตรฐาน สมการการหมุนในแกนสามมิติสามารถเขียนได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.25}$$

สามารถเขียนให้อยู่ในรูปย่อได้ดังนี้

$$P' = R_z(\theta) \cdot P \tag{2.26}$$

สมการในการหมุนสำหรับอีกสองแกนที่เหลือสามารถสร้างขึ้นจากการสลับลำดับเป็นวงกลมของตัวแปร x,y และ z

$$x \rightarrow y \rightarrow z \rightarrow x$$

เพราะฉะนั้นในการหมุนรอบแกน x จะได้สมการดังนี้

$$\begin{aligned} y' &= y\cos\theta - z\sin\theta \\ z' &= y\sin\theta + z\cos\theta \\ x' &= x \end{aligned} \tag{2.27}$$

ในรูปของพิกัดมาตรฐานเขียนได้เป็นดังนี้

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.28}$$

หรือ

$$P' = R_x(\theta) \cdot P \tag{2.29}$$

สมการของการหมุนรอบแกน y ได้ดังนี้

$$z' = z\cos\theta - x\sin\theta$$
$$x' = z\sin\theta + x\cos\theta \qquad (2.30)$$
$$y' = y$$

ในการหมุนรอบแกน y แสดงด้วยเมตริกได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (2.31)$$

หรือในรูปของเวกเตอร์

$$P' = R_y(\theta) \cdot P \qquad (2.32)$$

ในการหมุน ณ แกนใดๆ เราสามารถทำได้โดยการย้ายแนวแกนที่จะหมุนให้มาผ่านจุด (0,0) จากนั้นให้ทำการหมุนแกน x y หรือ z เพื่อให้แกนที่เราต้องการจะหมุนมาทับกับแกนเหล่านั้น จากนั้นก็ทำการหมุนตามที่ต้องการ ขั้นสุดท้ายให้เราทำการแปลงย้อนกลับไปสู่จุดเดิม

การเปลี่ยนขนาดวัตถุ (Object scaling)

สมการของการเปลี่ยนแปลงขนาดของตำแหน่ง P = (x,y,z) เทียบกับจุด (0,0) สามารถเขียนได้ดังนี้

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \qquad (2.33)$$

หรือ

$$P' = S \cdot P \tag{2.34}$$

โดยที่ตัวแปรของการเปลี่ยนแปลงขนาด sx,sy และ sz ต้องใส่เป็นค่าบวก เพราะค่าของการเปลี่ยนแปลงขนาดสามารถแสดงได้ดังนี้

$$x' = x \cdot sx, \qquad y' = y \cdot sy, \qquad z' = z \cdot sz \tag{2.35}$$

ในการทำการเปลี่ยนแปลงทางเรขาคณิตของสองมิติ และ สามมิติจะใช้การคูณเข้าข้างหน้าของพิกัดของออปเจ็คเดิม หากเราต้องการที่จะทำการเปลี่ยนขนาด (s1) ย้ายที่ (t1) และ เปลี่ยนขนาดอีกครั้ง (s2) กับออปเจ็ค P

จะเขียนได้ดังนี้

$$P' = s2 \cdot t1 \cdot s1 \cdot P \tag{2.36}$$

เมตริกไม่สามารถสลับที่กันได้หากมีการเปลี่ยนตำแหน่งของการคูณผลลัพธ์ก็จะต่างกันไป โดยที่การเปลี่ยนแปลงที่อยู่ด้านหลังจะถูกทำก่อนซึ่งก็คือ s1 ต่อมาเป็น t1 และ s2 ตามลำดับ

มุมมอง (Viewing)

มุมมองในระบบ 2 มิติ (Two-Dimension Viewing)

ในตอนนี้เราจะพิจารณาถึงกลไกในการแสดงภาพบนหน้าจอ โดยปกติแล้วเราสามารถระบุได้ว่าจะให้ส่วนไหนของภาพบ้างที่จะแสดงบนหน้าจอและภาพส่วนนั้นจะไว้ส่วนไหนของอุปกรณ์แสดงผล ในระบบของระบบพิกัด xy จะหมายถึงพิกัดทั้งหมด (world-coordinate) ซึ่งเราสามารถใช้ในการสร้างรูปได้ ในรูปสองมิติ มุมมองที่ถูกเลือกโดยการระบุส่วนย่อยของพิกัดทั้งหมด ผู้ใช้สามารถเลือกส่วนหนึ่งเพื่อการแสดงหรือเลือกหลายส่วนเพื่อใช้แสดงต่อกันเป็นภาพเคลื่อนไหวได้ ส่วนของภาพในบริเวณที่เลือกจะถูกปรับให้เข้ากับพิกัดของอุปกรณ์ที่ใช้แสดงผล ซึ่งก็ต้องใช้การย้ายที่ การหมุน และการเปลี่ยนขนาด มาใช้เพื่อให้ภาพที่อยู่ในพิกัดทั้งหมดสามารถแสดงในพิกัดของอุปกรณ์แสดงผลได้ อีกทั้งลบในส่วนของพิกัดทั้งหมดที่อยู่นอกเหนือส่วนที่ใช้แสดงผล

ขั้นตอนในการแสดงผล (The Viewing Pipeline)

บริเวณของพิกัดทั้งหมดที่ถูกเลือกเพื่อใช้แสดงผลเรียกว่าวินโดว์(window) และบริเวณของส่วนแสดงภาพของอุปกรณ์ที่วินโดว์ปรับมาเรียกว่า วิวพอร์ต(viewport) วินโดว์จะบอกว่าอะไรที่จะแสดงให้เห็น ส่วนวิวพอร์ตจะเป็นตัวบอกว่าจะแสดงที่ไหน ส่วนใหญ่วินโดว์และวิวพอร์ตจะเป็นสี่เหลี่ยม ซึ่งจะสามารถใช้ในการปรับสัดส่วนได้โดยตรงทันที แต่ก็มีบางวินโดว์หรือวิวพอร์ตที่เป็นรูปอื่น เช่น โพลีกอนและวงกลม ที่ถูกใช้ในบางโปรแกรมซึ่งจะใช้เวลาในการประมวลผลนาน ปกติแล้วการปรับพิกัดทั้งหมดให้เป็นพิกัดของอุปกรณ์จะหมายถึงการปรับมุมมอง(viewing transformation) บางครั้งการปรับมุมมองในสองมิติจากวินโดว์มาเป็นวิวพอร์ตในทันทีอาจทำให้เกิดข้อผิดพลาดขึ้นได้

เราจะพูดถึงในกรณีที่วินโดว์และวิวพอร์ตเป็นรูปสี่เหลี่ยมมาตรฐาน ในขั้นแรกเราก็ทำการสร้างภาพในพิกัดทั้งหมดจากนั้นก็ทำการกำหนดระบบพิกัดการมองสองมิติขึ้นในพิกัดทั้งหมดเพื่อระบุวินโดว์ต่อไปก็สามารถปรับจากพิกัดการมองที่เราได้กำหนดไปเป็นระบบพิกัดปกติ(Normalize View Port)ซึ่งมีขนาดหนึ่งคูณหนึ่งหน่วย ขั้นสุดท้ายก็ทำการตัดภาพที่อยู่นอกขอบเขตของวิวพอร์ตและนำข้อมูลในวิวพอร์ตมาเปลี่ยนเป็นภาพในพิกัดของอุปกรณ์

การเปลี่ยนจากพิกัดวินโดว์ไปเป็นวิวพอร์ต (Window-to-Viewport Coordinate Transformation)

ในการปรับจากวินโดว์ไปเป็นวิวพอร์ตอันดับแรกเราต้องหาค่าในการปรับขนาดก่อนจาก

$$s_x = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$$
$$s_y = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}} \qquad (2.36)$$

เมื่อเราปรับขนาดของวินโดว์ให้เท่ากับวิวพอร์ตโดยใช้ค่าการขยายที่ได้คำนวณมาแล้ว จากนั้นทำการย้ายตำแหน่งของวินโดว์ไปที่ตำแหน่งของวิวพอร์ต

การตัด(Clipping)

วิธีในการดูว่าภาพส่วนไหนอยู่ภายในหรือภายนอกกรอบที่เรากำหนดให้แสดงเรียกว่าการตัด ส่วนบริเวณที่วัตถุถูกตัดออกมาเรียกว่า คลิพวินโดว์ ซึ่งวิธีในการตัดนี้ได้ถูกนำมาใช้กับวิวพอร์ตเพื่อลดการประมวลผลที่ไม่จำเป็นซึ่งก็คือวัตถุที่อยู่นอกวิวพอร์ตจะไม่ถูกนำมาแสดงผล

หลักการในการตัดออกคือหาจุดตัดของรูปกับคลิพวินโดว์จากนั้นก็ทำการตัดส่วนที่อยู่นอกคลิพวินโดว์ออกไป

การแสดงผลของภาพสามมิติ (Three-Dimension Viewing)

ในสองมิติการแสดงผลภาพหมายถึงการเปลี่ยนตำแหน่งจากพิกัดทั้งหมดไปยังพิกัดของอุปกรณ์ และการตัดออกก็เป็นการดูว่าภาพส่วนไหนอยู่นอกกรอบก็ตัดออก ซึ่งในสามมิติมุมมองในการมองมีได้หลายแบบ เช่นการมองจากด้านหน้า ด้านบน หรือด้านข้าง หรือเราสามารถมองจากด้านในของวัตถุเช่นอยู่ในตึก ตอนนี้การตัดออกในสามมิติก็จะขึ้นอยู่กับมุมมองการมองว่าเราเลือกแบบไหน

ขั้นตอนในการแสดงภาพ (Viewing Pipeline)

ในการแสดงมุมมองของภาพสามมิติเปรียบเสมือนการถ่ายภาพ อันดับแรกต้องกำหนดตำแหน่งของกล้องในพื้นที่ จากนั้นก็เลือกมุมของกล้อง เพื่อทำการจับภาพเมื่อกดชัทเตอร์ ภาพจะถูกจับให้เท่ากับขนาดของหน้าจอของกล้อง และแสงจากพื้นผิวที่มองเห็นถูกจับลงบนฟิล์ม

ขั้นตอนในการแสดงผลภาพสามมิติ เมื่อเราทำการจับภาพ ตำแหน่งของพิกัดทั้งหมด (world-coordinate) จะถูกเปลี่ยนเป็นบริเวณที่สามารถมองเห็นได้ (viewing-coordinate) จากนั้นก็แล้วแต่กระบวนการในการแสดงภาพ (projection operation) เพื่อทำการเปลี่ยนบริเวณที่สามารถมองเห็นได้ให้เป็นไปตามรูปแบบของการแสดงภาพที่กำหนดไว้และแสดงบนหน้าจอ

Projection

การแสดงภาพจากสามมิติบนมุมมองแนวราบสองมิติ ซึ่งมีกระบวนการพื้นฐานสองแบบ คือ การถ่ายภาพแบบขนาน(parallel projection) คือการแสดงภาพโดยการลากเส้นขนานระหว่างวัตถุกับพื้นผิวที่ใช้แสดงซึ่งจะทำให้ขนาดของวัตถุไม่มีการเปลี่ยนแปลงไม่ว่าจะอยู่ห่างจากกล้องเท่าไหร่ก็ตาม ตัวอย่างของกระบวนการแบบนี้คือ การมองภาพด้านหน้า(front-view) ด้านข้าง(side-view)และด้านบน(top-view) กระบวนการที่สองคือ การถ่ายภาพแบบเหมือนจริง(perspective projection) คือตำแหน่งของวัตถุถูกเปลี่ยนแปลงไปยังพื้นผิวที่ทำการแสดงภาพถูกบีบไปยังจุดที่เรียกว่า จุดอ้างอิงการถ่ายภาพ (projection reference point หรือ center of projection) ซึ่งจะได้ภาพที่เสมือนการมองเห็นของคนคือวัตถุที่อยู่ใกล้กว่าก็จะมองเห็นใหญ่กว่าวัตถุที่อยู่ไกลออกไป แม้เวลาสร้างจะทำให้มีขนาดเท่ากัน

การตัด(clipping)

ในสามมิติเราจะพิจารณาในการตัดวัตถุที่ไม่ถูกแสดงผลออกไปเพื่อประหยัดเวลาในการประมวลผลโดยสามารถใช้กระบวนการของในสองมิติโดยดูที่ขนาดของส่วนแสดงผลว่าวัตถุไหนอยู่นอกส่วนการแสดงผลก็จะถูกตัดออกและมีอีกอย่างที่เพิ่มในการตัดออกก็คือพื้นผิวของวัตถุที่อยู่ในพื้นที่แสดงผลแต่ไม่ถูกแสดงผล เช่นด้านหลังของวัตถุที่มุมมองของกล้องมองไม่เห็น ก็จะถูกตัดออก

2.3 OGRE game engine

บทนำ

ในเขียนโปรแกรมคอมพิวเตอร์กราฟฟิคแบบเรียลไทม์ในปัจจุบันต้องใช้การความสามารถของนักพัฒนาสูงมาก เพียงการเข้าใจในพื้นฐานคอมพิวเตอร์กราฟฟิคก็ใช้เวลาจำนวนมากแล้ว และยังต้องระมัดระวังในการเลือกใช้เทคนิคต่างๆ ไม่ว่าจะเป็นเทคนิค การควบคุมการสะท้อน(shader) แสงเงา(shadow) จุดกำเนิดแสง(light) ลวดลายพื้นผิววัตถุ(texture) และอื่นๆ ซึ่งให้ผลลัพธ์ที่แตกต่างกัน และใช้ทรัพยากรต่างกัน ซึ่งมีโครงการหนึ่งที่มีชื่อว่า OGRE ที่มีจุดมุ่งหมายในการแก้ไขปัญหาเหล่านี้โดยมีเป้าหมายสำหรับผู้พัฒนาที่ไม่ต้องการเสียเวลากับเรื่องเหล่านั้น

รายละเอียดพื้นฐาน

OGRE อ่านว่า โอ-เกอะ ซึ่งมีคำเต็มว่า Object-Oriented Graphics Rendering Engine ถูกพัฒนาและแจกจ่ายภายใต้สิทธิบัตรซอฟต์แวร์แบบเปิดเผยชุดคำสั่งโดยใช้ LGPL (Lesser GNU Public License) ซึ่งหมายความว่าผู้พัฒนาสามารถนำ OGRE ไปพัฒนาและแจกจ่ายต่อได้โดยที่ไม่มีค่าสิทธิบัตร ไม่ว่าจะพัฒนาเพื่อโครงการศึกษาวิจัย หรือพัฒนาเพื่อใช้ในเชิงพาณิชย์

รุ่นของ OGRE ที่ได้นำมาใช้ในโครงงานนี้ได้ใช้สิทธิบัตรแบบ LGPL เพียงอย่างเดียวซึ่งมีเนื้อหาสาระที่สำคัญดังนี้

- คุณสามารถแจกจ่ายไบนารี่ไฟล์ของโครงการที่ใช้สิทธิบัตร LGPL เพียงอย่างเดียวได้ตราบเท่าที่ ชุดคำสั่งที่นำมาสร้างไบนารี่ไฟล์ไม่ได้ถูกเปลี่ยนแปลงไปจากต้นฉบับ และชุดคำสั่งต้นฉบับสามารถหาได้
- สิ่งที่แตกต่างกับสิทธิบัตรแบบ GPL(GNU Public License) คือ สิทธิบัตรแบบ GPL นั้นเมื่อนำซอฟต์แวร์ GPL ไปใช้ในทางใดทางหนึ่ง ซอฟต์แวร์ที่ใช้นั้นจะต้องเปิดเผยชุดคำสั่ง แต่ LGPL ไม่มีการบังคับ (เพราะมีคำว่า Lesser นำหน้า GPL)
- คุณไม่สามารถทำการเชื่อมโยงโครงสร้างซอฟต์แวร์แบบคงที่(statically link) ของคุณเข้ากับซอฟต์แวร์แบบ LGPL ยกเว้นซอฟต์แวร์ของคุณจะต้องใช้สิทธิบัตรแบบ LGPL

ลักษณะสำคัญ

ลักษณะสำคัญ หรือความสามารถของ OGRE มีดังนี้

- รองรับการทำงานบนชุดคำสั่ง OpenGL และ Direct3D
- รองรับการทำงานบนฐานงาน(Platform) Windows Linux และ Mac OS X
- เป็นซอฟต์แวร์เชิงวัตถุ ง่ายต่อการพัฒนา แก้ไข เปลี่ยนแปลง เพิ่มเติมความสามารถ
- ควบคุมการจัดการสถานะของการเรนเดอร์(render) และการเลือกวัตถุในการแสดงผลเพื่อเพิ่มความเร็วโดยใช้แนวคิดแบบลำดับชั้น (hierarchical culling)

- รองรับระบบสคริป(script) สำหรับปรับแต่งการจัดการทรัพยากรที่จะนำมาใช้ในการแสดงผล
- รองรับภาษา shading แบบแอสเซมบลี(assembled) และแบบระดับสูง(high-level)ทั้งหมด เช่น Cg HLSL GLSL โดยนำไปใช้ในเทคนิค Programmable GPU ซึ่งออกแบบเพื่อลดการประมวลผลทางฝั่ง CPU และให้ GPU ควบคุมการประมวลผลบางอย่าง
- รองรับรูปแบบของแฟ้มภาพหลายชนิด เช่น PNG TGA DDS TIF GIF JPG 1D และเทคนิคการแสดงผล เช่น มิติของลวดลาย (volumetric texture) เทคนิคการวางลวดลายแบบกล่อง (cube maps)
- รองรับการเรนเดอร์ลงบนลวดลาย (texture) ก่อนนำไปแสดงบนวัตถุเพื่อลดการประมวลผลแบบเรียลไทม์(real time)
- รองรับเทคนิค LOD (level of detail) หรือ การลดการประมวลผลเมื่อวัตถุอยู่ห่างออกไปโดยการลดลายละเอียดของจุดลง
- รองรับเทคนิค mipmapping หรือ การแก้ไขลักษณะของลวดลายที่ผิดเพี้ยนจากการเปลี่ยนขนาดภาพเนื่องจากนำไปแสดงผลบนระบบแกนสามมิติ
- เพิ่มความเร็วให้โครงเส้นของวัตถุที่มีการเก็บข้อมูลรูปแบบไบนารี เพื่อใช้ในเทคนิค LOD
- มีส่วนต่อขยายสำหรับซอฟต์แวร์สร้างวัตถุสามมิติเพื่อให้สามารถนำมาใช้งานได้ใน OGRE
- รองรับการติดต่อโดยตรงกับ เวอร์เท็กซ์(vertex) อินเดกซ์บัฟเฟอร์(index buffer) การสร้างเวอร์เท็กซ์(vertex) และ บัฟเฟอร์แมพพิ้ง(buffer mapping) ที่ถูกซ่อนไว้ได้โดยตรง
- รองรับการเคลื่อนไหวของวัตถุแบบ โครงกระดูก(skeletal) และ โพส(pose)
- รองรับส่วนต่อขยายสำหรับการแสดงผลฉาก(scene) ซึ่งซอฟต์แวร์แต่ละชนิดอาจใช้เทคนิคการจัดการฉาก(scene) ที่แตกต่างกัน
- รองรับเทคนิคการทำเงา(shadowing) หลายชนิด เช่น stencil texture additive และ modulative ด้วยการช่วยเหลือของฮาร์ดแวร์
- รอบรับเทคนิคการจำลองท้องฟ้าแบบ skybox skyplane และ skydome
- รองรับการจัดการการประมวลผลวัตถุโปร่งใส

โครงสร้าง และการออกแบบ

หลักการ

OGRE เป็นชุดคำสั่งสำหรับแสดงผลกราฟฟิค ถูกพัฒนาด้วยภาษาเชิงวัตถุ C++ ซึ่งเตรียมการติดต่อการประมวลข้อมูลที่จำเป็นต้องทำทุกครั้งให้ เช่น การแสดงผลวัตถุทั้งฉาก โดยทั่วไปเมื่อใช้ OpenGL หรือ DirectX นั้นจะทำงานที่เป็นลำดับเช่น จัดเตรียมสถานะการแสดงผล ส่งข้อมูลโครงสร้างรูปทรงพื้นฐานเรขาคณิต ส่งสัญญาณให้ GPU แสดงผล ทำตามลำดับจนกระทั่งแสดงผลครบทั้งหน้าจอ และจะทำซ้ำทุกๆรอบการแสดงผล

ด้วยระบบเชิงวัตถุ ผู้พัฒนาไม่จำเป็นต้องกระทำกับโครงสร้างพื้นฐาน แต่จะกระทำกับทุกๆสิ่ง ที่แสดงผลได้เป็นสิ่งที่เหมือนกัน ในที่นี้จะเรียกว่า ซีนโหนด(SceneNode) เช่น ซีนโหนดที่สามารถเคลื่อนไหว ซีนโหนดที่มีสถานะคงที่ ซีนโหนดของแสง ซีนโหนดตำแหน่งกล้อง และอื่นๆ ประกอบเข้ากันเป็นฉาก

การควบคุมการเคลื่อนที่ของซีนโหนดนั้นก็จะเคลื่อนที่ในหน่วยของเวทเตอร์ การหมุนใช้หน่วยของ องศา หรือ เรเดียน โดยทำให้กับทุกๆจุดในซีนโหนด ไม่จำเป็นต้องกำหนดเมทริก เพื่อนที่จะหมุน หรือเคลื่อนที่ หรือแก้ไขกับทุกๆ ลำดับของจุด หรือ ทุกๆสามเหลี่ยม ด้วยตนเอง

การแสดงผลสามารถเลือกทำงานบนชุดคำสั่ง OpenGL หรือ DirectX ได้ส่งค่าให้ OGRE เพื่อที่จะเลือกส่วนต่อขยายที่เป็น OpenGL หรือ DirectX ซึ่งระบบการแสดงผลได้ปกปิดการทำงานที่ซับซ้อนโดยมีตัวประสานเพื่อที่จะเรียกการทำงานของส่วนต่อขยายที่แตกต่างกัน รวมไปถึงสามารถปรับเปลี่ยนส่วนขยายส่วนอื่นๆ ได้อีก เช่น การจัดการของฉาก การจัดการของซีนโหนด และอื่นๆ

ระบบย่อย

Root

ออบเจค Root เป็นส่วนหลักของระบบทั้งหมดที่จะเริ่มต้นและสิ้นสุดการทำงานของ OGRE สิ่งที่ Root จะทำหลังจากสร้าง Root คือสร้างออบเจค Ogre manager ทั้งหมด จากนั้น ออบเจค Root จะทำหน้าที่เป็นคลาส façade หรือส่วนที่ติดต่อกับระบบย่อยอื่นๆ

ออบเจค Root ได้มีการเตรียมส่วนติดต่อผู้ใช้ที่สามารถปรับแต่งระบบให้ตรงความต้องการใช้งานเช่น ความละเอียดหน้าจอ จำนวนสีที่ใช้ ทำงานเต็มจอหรือไม่

ถ้าต้องการที่จะให้ OGRE ทำงานในสถานะแสดงผลแบบต่อเนื่อง (Continuous Rendering mode) หรือทำการแสดงผลให้ถี่ที่สุด (ส่วนใหญ่ใช้กับโปรแกรมประยุกต์สื่อผสม) จะต้องเรียก

คำสั่ง startRendering จากนั้น OGRE จะเริ่มเข้าสู่การแสดงผลแบบต่อเนื่อง และจะหยุดลงเมื่อหน้าต่างทั้งหมดถูกปิด

ระบบการแสดงผล (RenderSystem)

RenderSystem เป็น abstract class ที่กำหนดส่วนติดต่อกับ API 3 มิติ ซึ่งจะรับผิดชอบการส่งรายละเอียดการแสดงผล และการปรับแต่งลักษณะการแสดงผลต่างๆ sub class ที่ทำการระบุลักษณะเฉพาะของ API การแสดงผล เช่น D3DRenderSystem สำหรับ Direct3D และ GLRenderSystem สำหรับ OpenGL ออบเจค RenderSystem จะสามารถเข้าถึงได้จากคำสั่ง Root::getRenderSystem() แต่ปกติแล้วเราไม่จำเป็นติดต่อโดยตรง แต่จะผ่าน SceneManager แทน ยกเว้นต้องการเข้าถึงลักษณะพิเศษ

OGRE Managers

คลาส Manager ใน OGRE เป็นคลาสที่ทำหน้าที่จัดการวงจรชีวิตและการเข้าถึง ออบเจคชนิดต่าง คลาส Manager ใน OGER มีดังนี้

LogManager ทำการส่งข้อความผ่านทางกระแสข้อมูลออกสู่แฟ้มข้อมูล

ControllerManager ใช้ในการควบคุมการเคลื่อนไหว ลวดลาย หรือ ส่วนประกอบของตัวละคร

DynLibManager ทำหน้าที่อ่าน DLL บนระบบปฏิบัติการ Windows และอ่าน share object บนระบบปฏิบัติการ Linux เข้าหน่วยความจำ เพื่อสำหรับเรียกใช้ส่วนต่อขยายในการเพิ่มความสามารถของ OGRE

PlatformManager เตรียมชุดคำสั่งสำหรับเข้าถึงรายละเอียดต่างๆของ ระบบปฏิบัติการ และฮาร์ดแวร์

CompositorManager เตรียมการเข้าถึงและการจัดการ ระบบ Compositor

ArchiveManager เตรียมระบบจัดการทรัพยากร เพื่อที่สามารถจัดการแฟ้มข้อมูลประเภท zip

ParticleSystemManager จัดการรายละเอียดและประมวลผล particle systems emitters และ affecters

MaterialManager ทำหน้าที่เก็บรักษา material ทั้งหมดในโปรแกรมที่อยู่ในหน่วยความจำ เพื่อสามารถนำกลับมาเรียกใช้ใหม่

SkeletonManager ทำหน้าที่เก็บรักษา skeleton ทั้งหมดในโปรแกรมที่อยู่ในหน่วยความจำ เพื่อสามารถนำกลับมาเรียกใช้ใหม่

MeshManager ทำหน้าที่เก็บรักษา mesh ทั้งหมดในโปรแกรมที่อยู่ในหน่วยความจำ เพื่อสามารถนำกลับมาเรียกใช้ใหม่

HighLevelGpuProgramManager ทำหน้าที่อ่าน แปลงข้อมูล และเก็บรักษา high-level GPU shader และ vertex ทั้งหมดที่อยู่ในหน่วยความจำ เพื่อสามารถนำกลับมาเรียกใช้ใหม่

GpuProgramManager ทำหน้าที่อ่าน แปลงข้อมูล และเก็บรักษา assembler GPU shader ทั้งหมดที่อยู่ในหน่วยความจำ เพื่อสามารถนำกลับมาเรียกใช้ใหม่

ExternalTextureSourceManager ทำหน้าที่จัดการ ทรัพยากรที่อยู่ภายนอกโปรแกรม เช่น การเล่นไฟล์วิดีโอ

FontManager ทำหน้าที่จัดการ และเก็บรักษาชุดของอักษรในการแสดงผลบนหน้าจอ

ResourceGroupManager ทำหน้าที่เป็นตัวกลางในการติดต่อ Manager อื่นๆ เพื่อทำการอ่านทรัพยากรจากแฟ้มข้อมูลจากชื่อกลุ่ม

OverlayManager ทำหน้าที่สร้างพื้นผิว 2 มิติ ส่วนใหญ่ใช้สำหรับติดต่อกับผู้ใช้ซอฟต์แวร์

HardwareBufferManager จัดการและเข้าถึง หน่วยความจำร่วมของฮาร์ดแวร์

TextureManager จัดการ อ่านแฟ้มข้อมูลบันทึกเข้าหน่วยความจำ และเข้าถึง ลวดลาย (texture) ทั้งหมด

การจัดการทรัพยากร(ResourceGroupManager)

ทรัพยากรในที่นี้หมายถึง ทุกสิ่งที่ใช้ในการแสดงผล และทรัพยากรทั้งหมดนี้จะถูกจัดการด้วย ออบเจค ResourceGroupManager เมื่อเริ่มต้นทำงาน ResourceGroupManager จะทำการค้นหาจากตำแหน่งแฟ้มข้อมูลที่ถูกกำหนดไว้ในออบเจค Root แต่จะไม่ทำการอ่านเข้าหน่วยความจำทันที แต่ละแฟ้มจะอ่านเข้าหน่วยความจำเมื่อมีการเรียกใช้งาน

ประเภทของทรัพยากรที่ OGRE สามารถที่จะอ่านได้

- Mesh เก็บข้อมูลโครงสร้างของวัตถุ OGRE จะอ่านได้เฉพาะ แฟ้มข้อมูลประเภทไบนารี และเก็บเพียงหนึ่ง mesh โดยอาจมีข้อมูลการเคลื่อนไหวเก็บในแฟ้มข้อมูลนี้ได้ ถ้ามีแฟ้มข้อมูล mesh เป็นประเภท xml สามารถแปลงจากแฟ้มข้อมูล mesh ประเภท xml เป็นไบนารี ได้จากคำสั่ง OgreXMLConverter แฟ้มข้อมูล mesh มีนามสกุลของแฟ้มคือ .mesh
- Skeleton เก็บข้อมูลโครงสร้างของลำดับชั้นโครงกระดูก และ ตำแหน่งของโครงกระดูกแต่ละชิ้น ณ เวลาต่างๆ ส่วนใหญ่จะใช้งานควบคู่กับแฟ้มข้อมูลประเภท mesh มีนามสกุลของแฟ้มคือ .skeleton
- Material เก็บข้อมูลสถานะการแสดงผลของวัตถุ ใช้เมื่อประมวลผลรูปทรงเรขาคณิตในวัตถุ ส่วนใหญ่จะใช้งานควบคู่กับแฟ้มข้อมูลประเภท mesh มีนามสกุลของแฟ้มคือ .meterial

- GPU Program เก็บข้อมูลชุดคำสั่งของ GPU ซึ่งมี 2 ลักษณะ คือ ชุดคำสั่งระดับสูง เช่น HLSL GLSL และ Cg จะมีนามสกุลของแฟ้มคือ .program และชุดคำสั่งระดับต่ำ แอสเซมบลี จะมีนามสกุลของแฟ้มคือ .asm โดยอาจใช้คู่กับแฟ้มข้อมูลประเภท material
- Texture เก็บข้อมูลลวดลาย 2 มิติ สามารถมีรูปแบบของแฟ้มข้อมูลได้ตามที่ OGRE สามารถรับได้
- Compositor ใช้เหมือนกับแฟ้มข้อมูลประเภท material แต่มีนามสกุลของแฟ้มคือ .compositor
- Font เก็บข้อมูลของชุดตัวอักษร ใช้กับการแสดงสายอักษรออกทางหน้าจอ มีนามสกุลของแฟ้มคือ .fontdef

แต่ละชนิดของทรัพยากรจะมีออบเจค ในการจัดการเป็นของตัวเอง (ResourceManager) ยกตัวอย่างเช่น MaterialManager เป็นออบเจคจัดการแฟ้มข้อมูลประเภท material FontManager เป็นออบเจคจัดการแฟ้มข้อมูลประเภท font และอื่นๆ

ResourceGroupManager มีหน้าที่รับผิดชอบการค้นหาทรัพยากร อ่านทรัพยากรเข้าหน่วยความจำ และล้างทรัพยากรออกจากหน่วยความจำเมื่อไม่ใช้งาน โดยการอ้างอิงผ่านชื่อกลุ่มของทรัพยากร เช่น กลุ่ม “building mesh” หรือกลุ่ม “simple font” เป็นต้น

Mesh

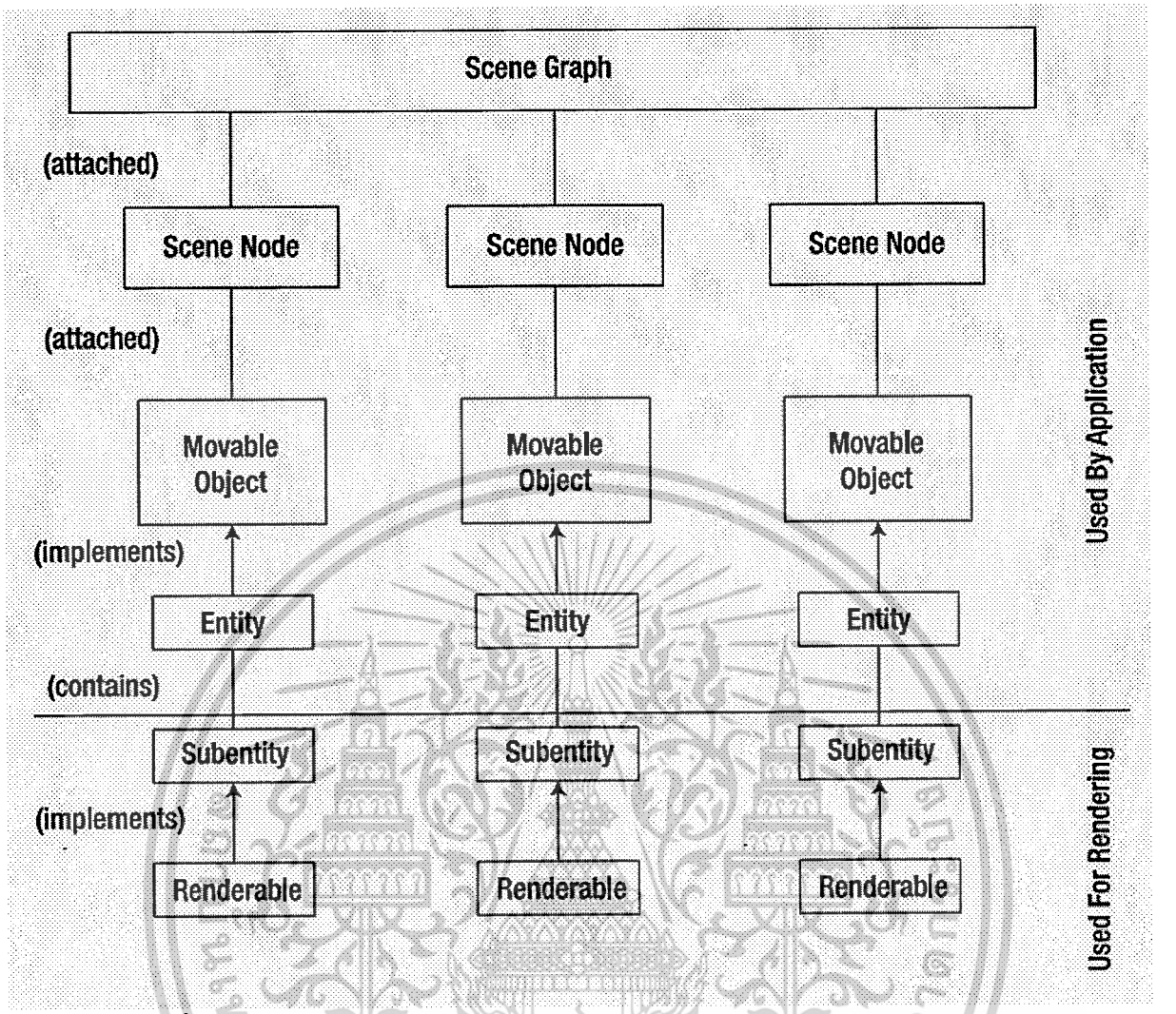
Mesh เป็นออบเจคที่เก็บ แบบจำลอง หรือกลุ่มของรูปทรงเรขาคณิตที่ใช้ในการแสดงผล เช่น ตัวละคร สิ่งของ แต่จะไม่ใช้ในการเก็บพื้นผิวของภูมิประเทศ

Mesh คือ ทรัพยากรใน OGRE ชนิดหนึ่งซึ่งมี MeshManager เป็นตัวจัดการทรัพยากร ซึ่งปกติแล้วจะอ่านจากแฟ้มข้อมูลนามสกุล .mesh โดย .mesh จะถูกสร้างโดยใช้เครื่องมือขึ้นแบบจำลอง อีกทีหนึ่ง แต่การสร้างก็สามารถสร้างได้เองจากการใช้คำสั่ง MeshManager::createManual โดยจะต้องกำหนดรูปทรงเรขาคณิตด้วยตัวเอง

การจัดการฉาก (SceneManager)

การจัดการฉากเป็น ส่วนที่ขาดไม่ได้ในทุกซอฟต์แวร์การแสดงผล โดยทั่วไปแล้วจะเก็บวัตถุต่างๆภายในฉากโดยใช้อัลกอลิทึม scene graph ซึ่งทำให้มีความสามารถค้นหา ประมวลผล และตรวจสอบการสัมผัสระหว่างวัตถุได้รวดเร็ว มีประสิทธิภาพสูงสุด หน้าที่ของ SceneManager มีดังนี้

- สร้างและจัดวางตำแน่งของ ตัวละคร สิ่งของ แสง และกล้องในฉาก
- สร้างภูมิประเทศ โดยแตกต่างจาก ตัวละคร และสิ่งของ คือ ภูมิประเทศนั้นจะคงที่และไม่สามารถเปลี่ยนตำแหน่ง
- จัดการค้นหาตำแหน่งการสัมผัสระหว่างวัตถุต่างๆ หรือระหว่างวัตถุและภูมิประเทศ
- จัดการวัตถุที่สามารถมองเห็นได้ให้เก็บลง queue และวัตถุที่ถูกบังจะไม่เก็บลง queue เพื่อแสดงผล
- จัดการแสดงผล แสงและเงาในฉาก
- จัดการแสดงผล ภูมิประเทศและท้องฟ้า
- ส่งผลลัพธ์ไปยัง RenderSystem เพื่อแสดงออกทางหน้าจอ



รูปที่ 2.6 ความสัมพันธ์ของ scene graph และการจัดการของวัตถุใน OGRE

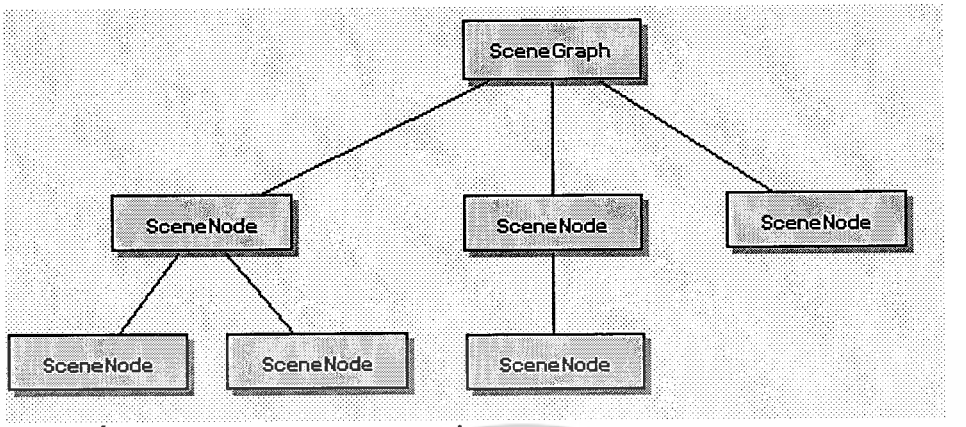
การสร้าง Scene graph สามารถสร้างจาก SceneManager โดยสามารถเลือกสร้าง Scene graph ที่มีรูปแบบการทำงานแต่ละชนิดได้ โดยการเรียกใช้งานของ Scene graph นั้น OGRE จะกระทำกับ interface Scene graph ซึ่งหมายความว่า OGRE จะไม่สนใจรูปแบบวิธีการประมวลผลของ Scene graph ซึ่งทำให้ผู้ใช้สามารถเลือกรูปแบบวิธีประมวลผลของ Scene graph หรือพัฒนา Scene graph เองได้

Renderable เก็บข้อมูลโครงสร้างของรูปทรงเรขาคณิต โดยอิมพลีเมน Subentity

Entity ทำหน้าที่เก็บออบเจคของ Subentity เมื่อ Subentity มีความสัมพันธ์กันสามารถเก็บ Subentity ได้หลายตัว เมื่อมีการหมุน หรือเคลื่อยย้าย จะการทำกับ Subentity ทั้งหมด

Scene Node จะไม่มีการเข้าถึง หรือจัดการเกี่ยวกับโครงสร้างข้อมูลที่จะนำมาแสดงผลโดยตรง ซึ่ง Scene Node จะเก็บ Entity โดยใช้ interface Moveable object

Scene Node ใน OGRE สามารถที่จะมีความสัมพันธ์ระหว่างกันเชิงลำดับชั้น (มีโหนดพ่อและ โหนดลูก) เมื่อทำการหมุน หรือเคลื่อนที่โหนดพ่อแล้ว จะมีผลไปสู่โหนดลูกด้วย



รูปที่ 2.7 แสดงผลกระทบต่อการเปลี่ยนแปลงของโหนดพ่อ ไปยังโหนดลูก

นอกจากนี้ SceneNode ยังสามารถผูกติด SceneNode ที่มีคุณสมบัติอื่นได้อีกด้วย เช่น สามารถผูกติดกับโหนดกล้องเพื่อให้ติดตามโหนดพ่อ หรือ ผูกติดกับโหนดแสงได้เช่นกัน

ประเภทของ SceneManager

OGRE มีการพัฒนาส่วนขยายเบื้องต้นของการจัดการฉากให้ 2 ชนิด คือ

- OctreeSceneManager เป็นการจัดการฉากแบบมาตรฐานสามารถใช้ได้กับซอฟต์แวร์ทุกประเภท
- TerrainSceneManager เป็นการจัดการฉากแบบพิเศษที่เพิ่มความเร็วในการสร้างภูมิประเทศจาก heightmapped

SceneQueries

SceneQueries สามารถให้ข้อมูลเกี่ยวกับการสัมผัสซึ่งกันหรือตัดกันของวัตถุแบบเบื้องต้น การสัมผัสหรือตัดกันแบบเบื้องต้น แบ่งออกเป็น 4 ประเภท คือ

- Ray queries คือการตรวจสอบเส้นตรงสัมผัสหรือตัดกันกับวัตถุ โดยกำหนดจุด 2 จุดของเส้นตรง
- Sphere queries คือการตรวจสอบทรงกลมสัมผัสหรือตัดกันกับวัตถุ โดยกำหนดจุดศูนย์กลาง และรัศมีของทรงกลม
- Bounding-box queries คือการตรวจสอบทรงสี่เหลี่ยมกับวัตถุสัมผัสหรือตัดกันกับวัตถุ โดยกำหนดจุด 2 จุดมุมทะแยงกันของสี่เหลี่ยม
- Intersection queries คือการตรวจสอบวัตถุภายในฉากที่สามารถสัมผัสหรือตัดกันได้

SceneQueries อาจใช้ในสถานการตรวจสอบระหว่างวัตถุและภูมิประเทศ (Terrain clamping) โดยการตรวจสอบทุกๆรอบการเคลื่อนไหวเพื่อให้ตรงกับทุกตำแหน่งการเคลื่อนที่

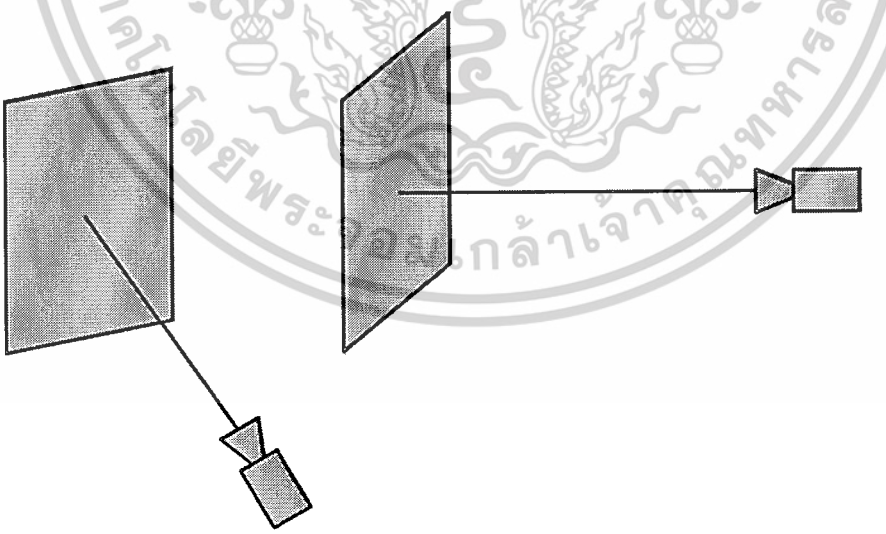
SceneQueries ทุกชนิดสามารถกรอง (maskable) ชนิดของวัตถุที่จะตรวจสอบที่ไม่ต้องการได้ เช่น อาจต้องการรายการโหนดของแสง โดยใช้วิธี Sphere queries แต่ไม่ต้องการโหนดของภูมิประเทศ สามารถตั้งค่าการกรอง ก่อนที่จะทำการสั่งตรวจสอบ

วัตถุที่สามารถเคลื่อนที่ได้ (Movable Scene Object)

วัตถุที่สามารถเคลื่อนที่ได้สามารถสร้างจาก SceneManager สามารถผูกติดกับโหนดพ่อใน SceneManager ได้มากที่สุดหนึ่งโหนด หรืออาจอยู่เพียงลำพังโดยไม่ผูกติดกับโหนดใดได้ แต่สามารถผูกติดกับโหนดลูกได้โดยไม่มีข้อจำกัด

วัตถุที่สามารถเคลื่อนที่ได้แบ่งเป็น 3 ชนิดคือ

- Resource-Base Object หรือวัตถุที่มีข้อมูลโครงสร้างเรขาคณิตภายในหรือ Mesh เมื่อถูกทำการสร้างจะไม่ถูกผูกติดกับโหนดใด แต่สามารถนำไปผูกติดกับ SceneManger จึงสามารถแสดงโครงสร้างเรขาคณิตภายในหน้าจอได้
- Quad-Base Object หรือวัตถุที่มีลักษณะเป็นแผ่นสี่เหลี่ยม ส่วนใหญ่ใช้แสดง ภาพ 2 มิติ (billboard) ส่วนติดต่อผู้ใช้ (GUI) ท้องฟ้า(sky box) ซึ่งจะหันหน้าเข้าหากล้องอยู่เสมอ โดยมีผิวที่สามารถเปลี่ยนแปลงได้จาก สคริป (script)
  - Billboard . มีลักษณะเป็นแผ่นสี่เหลี่ยมปกติ โดยจะหันหน้าเข้าหากล้องตลอดเวลา ซึ่งมีประโยชน์ในการแสดงผลตัวอักษร รูปภาพ หรือเอฟเฟคพิเศษเช่นการระเบิด หรือควัน ฝน



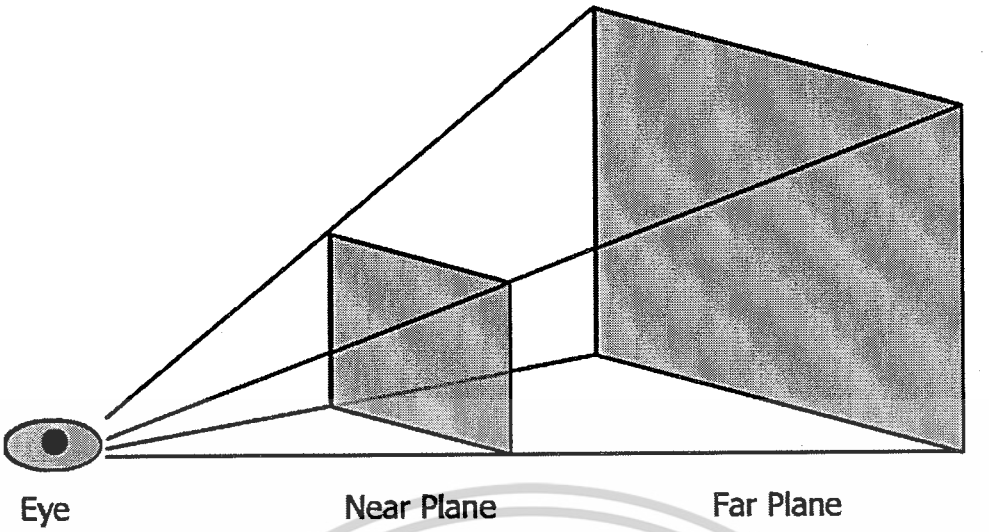
รูปที่ 2.8 แสดง Billboard ที่หันหน้าเข้าหากล้องตลอดเวลา

  - Skyplane มีลักษณะเป็นท้องฟ้า ถูกสร้างจาก plane สามารถกำหนดความกว้างยาวและระยะห่างจากกล้อง สามารถกำหนดให้โค้งงอได้ ทำให้ครอบคลุมท้องฟ้า

เอาไว้ มีคุณสมบัติใช้ร่วมกับการแสดงผลหมอกได้ดี สามารถใช้ tile texture กับพื้นผิวของท้องฟ้าได้

- Skydome ถูกสร้างจาก plane จำนวน 5 แผ่น ซึ่งคล้ายกับกล่องที่ไม่มีพื้นกล่อง แต่จะต้องใช้ภูมิประเทศเป็นพื้นแทน ซึ่งกำหนดให้โค้งงอเหมือน Skyplane ระยะห่างของผิวกล่องและกล้องจะเป็นค่าคงที่ซึ่งสามารถกำหนดได้ สามารถแสดงผลก้อนเมฆเคลื่อนไหวได้ และสามารถใช้ tile texture กับพื้นผิวท้องฟ้าได้
- Skybox ถูกสร้างจากกล่องมี 6 ด้าน ต่างจาก Skydome ที่ใช้ plane ซึ่ง Skybox ไม่สามารถทำให้โค้งงอได้ และไม่สามารถใช้ tile texture กับพื้นผิว ซึ่งทุกๆด้านของ Skybox จะใช้เทคนิค UV mapped

- Rendering Object เป็นวัตถุที่ไม่แสดงผลโดยตรงแต่จะทำหน้าที่แสดงผลวัตถุ และแสดงความลึกให้สมจริง กล้อง มีหน้าที่กำหนด มุมมองทรงกรวย (view frustum) ซึ่งสิ่งต่างๆ ในกล่องนี้จะแสดงผลออกทางหน้าจอตำแหน่งของกล้อง หรือตำแหน่งการแสดงภาพ สามารถเคลื่อนที่ไปได้ทุกตำแหน่ง สามารถหมุน และไม่จำเป็นต้องผูกติดกับ SceneNode เพื่อแสดงผล แต่เมื่อผูกติดกับวัตถุอื่นมีประโยชน์เพื่อเคลื่อนที่ตามไปเท่านั้น
  - แสง นอกจากทำให้ความลึกของภาพสมจริงแล้ว ยังจำเป็นสำหรับบางวัตถุ เช่น แสงไฟหน้ารถ แสงเทียน หรือแสงจากหลอดไฟ คุณสมบัติแสงของ OGRE จะสามารถแสดงผลได้แบบ local object illumination หรือคำนวนเพียงวัตถุต่อวัตถุ ไม่สามารถที่จะแสดงแบบ global object illumination หรือคำนวณการสะท้อนระหว่างวัตถุกับวัตถุ ได้เพราะปัจจุบันความเร็วการประมวลผลเรียลไทม์ ยังไม่สามารถทำได้ ชนิดของแสงใน OGRE ได้แบ่งออกเป็น 3 ประเภท คือ แบบจุด (point) แบบไฟฉาย (spot) แบบตามทิศทาง (directional) แสงทุกชนิดสามารถที่เคลื่อนย้ายตำแหน่ง จุดโฟกัสได้ และกำหนดสี



รูปที่ 2.9 แสดงมุมมองของ view frustum

วัตถุดงที่ (World Geometry)

วัตถุดงที่ คือ ทุกๆอย่างที่ อยู่ในฉาก เช่น ต้นไม้ สิ่งก่อสร้าง หญ้า ภูเขา ที่ไม่เคลื่อนไหว โดยเก็บอยู่ในโครงสร้างข้อมูลเรขาคณิต เมื่อสร้างวัตถุดงที่ ที่มีรายละเอียดมาก ควรแบ่งส่วนของ Mesh ออกเป็นส่วนย่อยๆ เพื่อทำการไม่ประมวลผลส่วนที่ถูกบัง หรืออยู่นอกขอบเขตการแสดงผล

2.4 wxWidgets

wxWidgets เป็นกรอบงานของภาษาซีพลัส ๆ ที่ใช้ในการทำอินเตอร์เฟสที่สามารถใช้ได้กับหลายแพลตฟอร์มได้แก่ ไมโครซอฟท์วินโดว์ ยูนิกซ์ และแมคโอเอส มีอินเทอร์เฟสให้เลือกใช้อย่างหลากหลายและสามารถนำมาเขียนโปรแกรมได้ง่าย เป็นซอฟท์แวร์เปิดจึงมีกลุ่มผู้ร่วมพัฒนามากซึ่งทำให้ wxWidgets มีความเสถียรและตรงตามความต้องการมากที่สุด

ลักษณะของ wxWidgets

- ความสามารถในการข้ามแพลทฟอร์ม

wxWidgets สามารถใช้ได้กับวินโดว์ 95/98/MP วินโดว์ NT/2K/XP ลีนุกส์/ยูนิกส์ ด้วยทูลคิต GTK+ และ MacOS จึงสามารถพัฒนาบนแพลตฟอร์มใดก็ได้และสามารถใช้ได้กับแพลตฟอร์มอื่น

- เป็นซอฟท์แวร์เปิด(Open Source)

wxWidgets เป็นซอฟท์แวร์เปิดซึ่งถูกพัฒนาขึ้นจากกลุ่มคนหลายกลุ่มที่มีความคาดหวังเดียวกันอย่างมีมาตรฐาน และยังเปิดเผยซอร์สโค้ดเพื่อให้เกิดการร่วมกันพัฒนาให้ดียิ่งขึ้น อีกทั้งยังได้ลิขสิทธ์ที่ให้ทุกคนสามารถใช้ในการประกอบธุรกิจได้อย่างถูกกฎหมาย

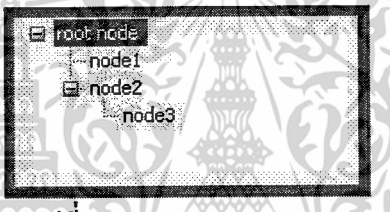
- มีเอกสารและโปรเจคตัวอย่าง

wxWidgets มีคู่มืออ้างอิงกว่า 1800 หน้า ในทุกรูปแบบเช่น HTML , Windows Help และ PDF และมีตัวอย่างกว่า 50 ตัวอย่าง เพื่อให้สามารถใช้ในการอ้างอิงเป็นรูปแบบในการสร้าง

- มีอินเทอร์เฟสให้เลือกหลายอย่างรวมทั้งของในวินโดว์ ยกตัวอย่างเช่น

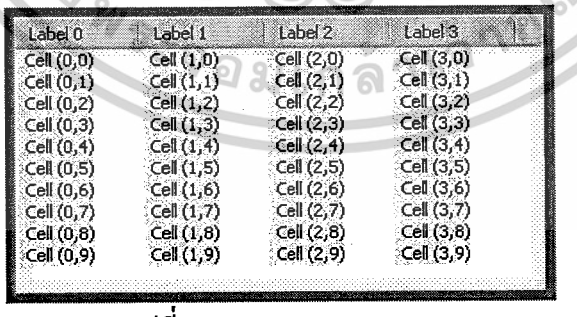
wxWidgets มีอินเทอร์เฟสเบื้องต้นเช่น ข้อความ หรือปุ่มที่มีรูปภาพ กล่องข้อความ แถบเลื่อน คอมโบบอกซ์ เช็คบอกซ์ และมีเพิ่มขึ้นมาอีกดังนี้

- wxTreeCtrl คือใช้ในการแสดงข้อความหรือข้อความที่มีรูปภาพเป็นลำดับขึ้น



รูปที่ 2.10 แสดง wxTreeCtrl

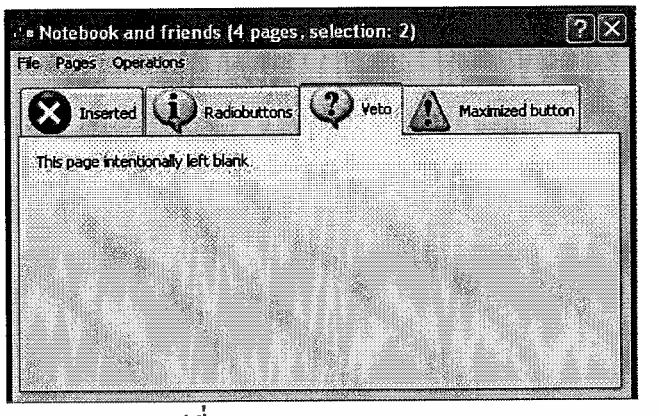
- wxListCtrl คือใช้แสดงชุดของข้อมูลสามารถเลือกแสดงเป็นไอคอนใหญ่หรือไอคอนเล็กก็ได้ จะแสดงในรูปแบบของการรายงานมีคอลัมภ์ที่สามารถปรับขนาดได้ตามต้องการ



Label 0	Label 1	Label 2	Label 3
Cell (0,0)	Cell (1,0)	Cell (2,0)	Cell (3,0)
Cell (0,1)	Cell (1,1)	Cell (2,1)	Cell (3,1)
Cell (0,2)	Cell (1,2)	Cell (2,2)	Cell (3,2)
Cell (0,3)	Cell (1,3)	Cell (2,3)	Cell (3,3)
Cell (0,4)	Cell (1,4)	Cell (2,4)	Cell (3,4)
Cell (0,5)	Cell (1,5)	Cell (2,5)	Cell (3,5)
Cell (0,6)	Cell (1,6)	Cell (2,6)	Cell (3,6)
Cell (0,7)	Cell (1,7)	Cell (2,7)	Cell (3,7)
Cell (0,8)	Cell (1,8)	Cell (2,8)	Cell (3,8)
Cell (0,9)	Cell (1,9)	Cell (2,9)	Cell (3,9)

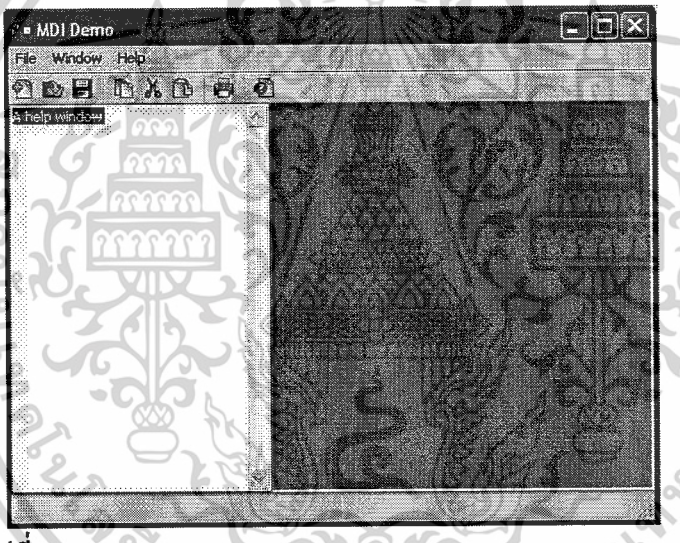
รูปที่ 2.11 แสดง wxListCtrl

- wxNotebook จะเป็นอินเทอร์เฟสที่มีลักษณะเป็นแท็บสามารถนำอินเทอร์เฟสต่างๆ มาวางในแท็บแต่ละหน้าได้



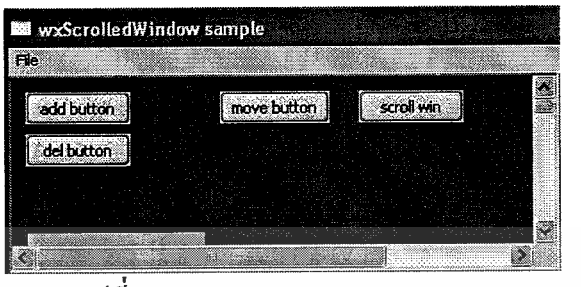
รูปที่ 2.12 แสดง wxNotebook

- wxMDIParentFrame และ wxMDIChildFrame อินเทอร์เฟสนี้จะช่วยให้สามารถแสดงเฟรมย่อยภายในเฟรมหลักได้



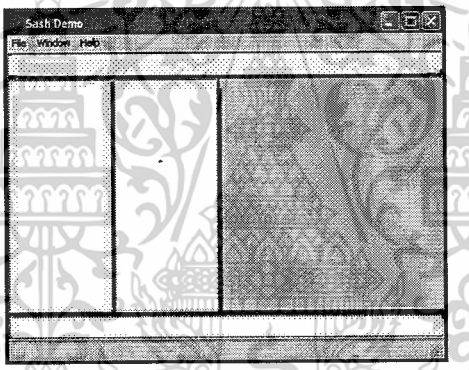
รูปที่ 2.13 แสดง wxMDIParentFrame และ wxMDIChildFrame

- wxScrolledWindow เป็นแถบเลื่อนของหน้าต่างๆ ซึ่งสามารถปรับให้เหมาะกับโปรแกรมประยุกต์ต่างๆ ได้



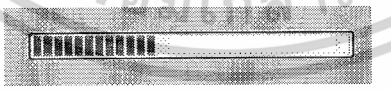
รูปที่ 2.14 แสดง wxScrolledWindow

- wxSplitterWindow และ wxSashWindow ใช้สำหรับแบ่งหน้าด้วยแถบเหมือนสายแพ



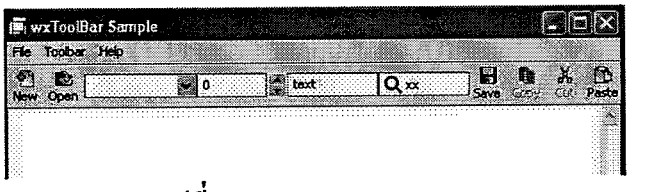
รูปที่ 2.15 แสดง wxSplitterWindow และ wxSashWindow

- wxGuage ใช้เพื่อสร้างเกตเพื่อแสดงสถานะการทำงาน



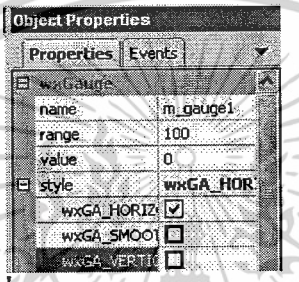
รูปที่ 2.16 แสดง wxGauge

- wxToolBar ใช้ในการสร้างทูลบาร์



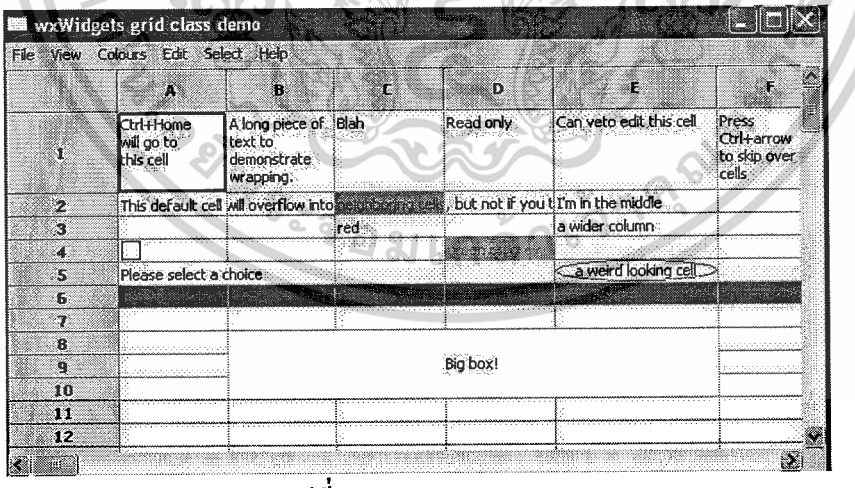
รูปที่ 2.17 แสดง wxToolBar

- wxPropertyList ใช้ในการสร้างพร็อบเพอร์ตี้สำหรับอินเทอร์เฟสต่างๆ



รูปที่ 2.18 แสดง wxPropertyList

- wxGrid ใช้ในการสร้างตารางหรือสร้างกริดเพื่อเก็บข้อมูลจะมีลักษณะเหมือนกับหน้าของโปรแกรม Excel



รูปที่ 2.19 แสดง wxGrid

- ระบบการจัดการกับเหตุการณ์ต่างๆ ที่มีประสิทธิภาพ

wxWidgets มีระบบการจัดการกับเหตุการณ์โดยการผูกเหตุการณ์ที่จะเกิดขึ้นกับอินเทอร์เฟสต่างๆ เข้ากับฟังก์ชั่นซึ่งทำให้สามารถจัดการกับเหตุการณ์ต่างๆ ได้ง่ายยิ่งขึ้น

- สะดวกสบายในการแสดงภาพ

wxWidgets รองรับไฟล์รูปภาพที่จะใช้แสดงทั้งของวินโดว์รวมทั้ง PostScript ของยูนิกซ์ด้วย

- การจัดการกับข้อผิดพลาดได้ง่าย

wxWidgets สามารถตรวจเช็คหน่วยความจำได้ว่ามีความจำรั่วรึเปล่าและจะแสดงข้อความมาเตือนเพื่อให้สามารถแก้ไขจุดที่ความจำรั่วได้

- สามารถใช้คอมไพเลอร์ได้หลายตัว

wxWidgets สามารถใช้คอมไพเลอร์ได้หลายตัวเช่น คอมไพเลอร์ซีพลัสพลัสของวินโดว์และ Cygwin หรือ Mingw32 ที่เป็นคอมไพเลอร์ที่ฟรี และมี makefiles เพื่อให้สามารถใช้ได้กับโปรเจคของ VC++ 5 และเวอร์ชั่นที่ใหม่กว่านี้ได้ ส่วนในด้านของยูนิกซ์ก็สามารถใช้คอมไพเลอร์ของยูนิกซ์ได้เลย

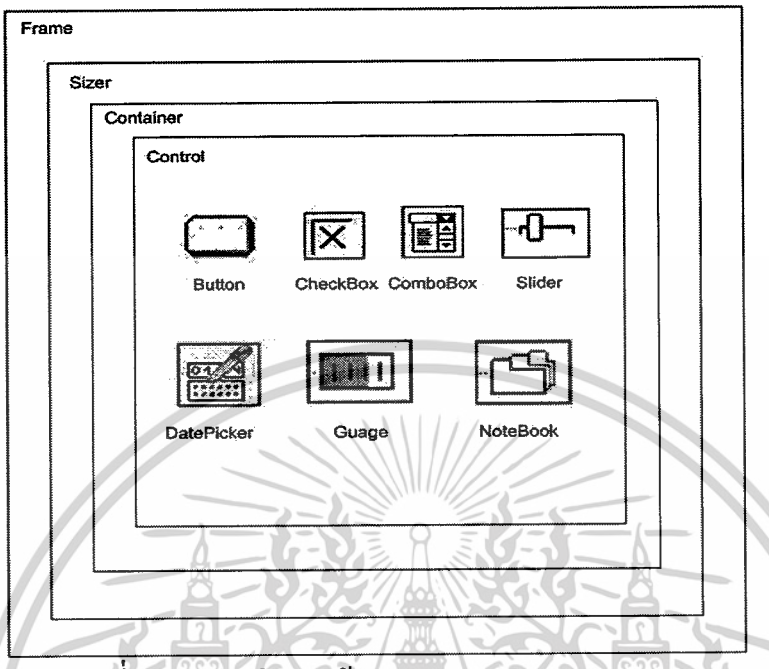
wxWidgets Architecture

ตารางแสดงสถาปัตยกรรมของ wxWidgets ซึ่งแบ่งออกเป็น 4 ชั้น คือ ชั้นแรก wxWidgets public API ซึ่งเป็น API กลางของ wxWidgets ชั้นที่สองคือ wxWidgets Port ซึ่งจะเป็นช่องทางที่จะเปลี่ยนโปรแกรมจากแพลทฟอร์มต่างๆ ให้เป็นภาษากลาง ชั้นที่สามคือ Platform API ซึ่งคือ API ของแต่ละแพลงฟอร์มที่จะใช้ ส่วนชั้นสุดท้ายคือ OS ที่สามารถรองรับได้

ตารางที่ 2.1 ตารางแสดงแพลทฟอร์มของ wxWidgets

<table>
<tr><td colspan="9">wxWidgets API</td></tr>
<tr><td colspan="9">wxWidgets Port</td></tr>
<tr><td>wxMSW</td><td>wxGTK</td><td>wxX11</td><td>wxMotif</td><td>wxMac</td><td>wxCocoa</td><td>wxOS2</td><td>wxPalmOS</td><td>wxMGL</td></tr>
<tr><td colspan="9">Platform API</td></tr>
<tr><td>Win32</td><td>GTK+</td><td>Xlib</td><td>Motif/Lesstif</td><td>Carbon</td><td>Cocoa</td><td>PM</td><td>Palm OS Protein APIs</td><td>MGL</td></tr>
<tr><td colspan="9">Operating System</td></tr>
<tr><td>Windows/Windows CE</td><td colspan="3">Unix/Linux</td><td>Mac OS 9/Mac OS X</td><td>Mac OS X</td><td>OS/2</td><td>Palm OS</td><td>Unix/DOS</td></tr>
</table>

โครงสร้างการทำงานของ wxWidgets



รูปที่ 2.20 แสดงโครงสร้างการทำงานของ wxWidgets

- Frame เป็นส่วนที่ทำหน้าที่เก็บอินเทอร์เฟสต่างๆ เอาไว้เวลาต้องการแสดงหน้าจอก็ต้องเอาส่วนประกอบต่างๆ ไว้ในเฟรมแล้วเรียกเฟรมไปแสดง
- Sizer เป็นส่วนที่ใช้ในการจัดผัง (Layout) ของส่วนประกอบในอินเทอร์เฟส
- Container เป็นส่วนที่ใช้ในการจัดคอนโทรลต่างๆ เป็นกลุ่มเพื่อง่ายต่อการจัดการ
- Control เป็นวัตถุที่จะนำมาจัดตกแต่งอินเทอร์เฟสให้ได้ตามต้องการ

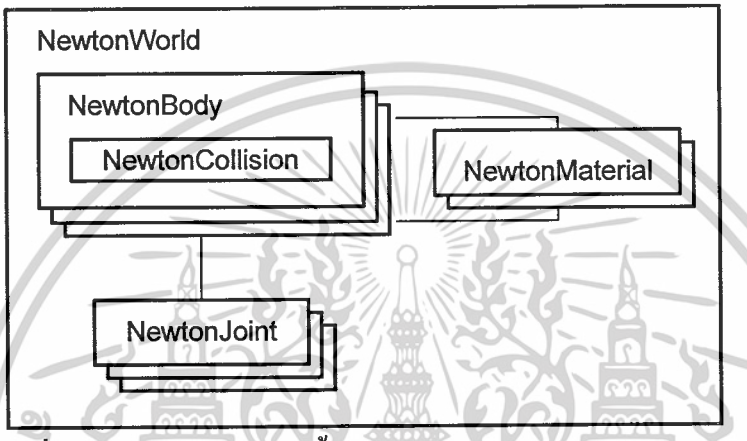
2.5 Newton Game Dynamics

Newton Game Dynamics เป็น engine สำหรับการจำลองสภาพแวดล้อมทางฟิสิกส์ ไลบรารี่ได้จัดเตรียม การจัดการเคลื่อนไหวของวัตถุ การตรวจจับการประทะกันของวัตถุ ซึ่งมีความเร็วพอที่จะประมวลผลแบบเรียลไทม์

หลักการของ Newton Game Dynamics เบื้องต้นจะให้ NewtonWorld เป็นแกนหลักของระบบ และทำการเพิ่ม NewtonBody ตามจำนวนของวัตถุ ซึ่งวัตถุนั้นไม่สามารถแบ่งแยกออกได้อีก ซึ่งวัตถุสามารถนิยามขอบเขตได้ด้วย NewtonCollision และนิยามการสัมผัสของพื้นผิวระว่างวัตถุใดๆโดย NewtonMaterial

ใน API ของ RakNet มีแม่แบบ (template) เบื้องต้นสำหรับการพัฒนา เช่น

- Vehicle container สำหรับการจำลองการเคลื่อนที่ของรถยนต์ทุกชนิด เช่น รถยนต์, รถกระบะ, รถบรรทุก และอื่นๆ
- Character controller สำหรับการเคลื่อนที่ของผู้เล่น ทำให้ผู้พัฒนาไม่จำเป็นต้องทำการคำนวณเอง
- Highfield collision สำหรับการตรวจสอบการกระทบของวัตถุ กับวัตถุแบบ Raycast function



รูปที่ 2.21 แสดงโครงสร้างเบื้องต้นของ Newton Game Dynamics

2.6 RakNet

RakNet เป็น API ที่ทำงานอยู่บน Berkeley Sockets ในระบบปฏิบัติการต่างๆ หรือ Winsock ในระบบปฏิบัติการวินโดว์ ซึ่งสามารถส่งข้อมูลภายในเครื่องคอมพิวเตอร์เดียวกัน หรือส่งข้อมูลผ่านเครือข่าย หรือระบบอินเตอร์เน็ต โดยเน้นที่ความเร็วในการส่งข้อมูล ดังนั้นการส่งกลุ่มข้อมูลจึงส่งอยู่บนโปรโตคอล UDP จุดประสงค์ของ RakNet ถูกสร้างมาเพื่อรองรับเกมออนไลน์ และจัดเตรียมการทำงานต่างๆ อย่างไรก็ตามมันก็สามารถที่จะรองรับการทำงานของโปรแกรมประยุกต์ใดๆ

ข้อเสียของโปรโตคอล UDP

- กลุ่มข้อมูลของ UDP ไม่รับประกันการส่งถึงปลายทางอาจได้ข้อมู,ทั้งหมด บางส่วนหรือไม่ได้เลย
- กลุ่มข้อมูลของ UDP ไม่รับประกันการรับข้อมูลแบบลำดับ อาจเกิดปัญหาเมื่อใช้อย่างไม่รอบคอบ เช่น อาจได้รับข้อมูลให้ทำการลบ จากนั้นอาจได้รับข้อมูลให้ทำการอ่านค่าภายหลัง
- กลุ่มข้อมูลของ UDP รับประกันว่าข้อมูลจะส่งถึงปลายทางอย่างถูกต้อง แต่ไม่มีการป้องกันจากการแก้ไขเปลี่ยนแปลงจากแฮกเกอร์

การแก้ปัญหาโปรโตคอล UDP

- ส่งข้อมูลซ้ำเมื่อข้อมูลส่งไม่ถึงปลายทาง
- เรียงลำดับข้อมูลให้ก่อนผู้ใช้เรียกใช้งาน
- ป้องกันข้อมูลที่ถูกส่ง และเตือนผู้ใช้เมื่อข้อมูลเกิดการเปลี่ยนแปลง

ความสามารถที่ RakNet รองรับ

- ส่งข้อมูลเสียงโดยใช้ช่องสัญญาณต่ำ แบบเรียลไทม์
- เปิด Master Server สำหรับเป็นสารบบ สำหรับค้นหาเกมอื่นๆ บนอินเตอร์เน็ต
- Remote function call หรือสามารถเรียก function บนเครื่องคอมพิวเตอร์อื่นบนเครือข่ายได้
- คำนวณสถิติการ Ping การรับและส่งข้อมูล จำนวนกลุ่มข้อมูลสูญหาย จำนวนข้อมูลที่ส่งและรับในหน่วย Byte และ Packet และอื่นๆ
- Timestamp เพิ่มความแม่นยำให้กับเวลาในการประมวลข้อมูลที่ได้รับ

ความสำคัญของกลุ่มข้อมูล

ความสำคัญของกลุ่มข้อมูลจะมีอยู่ 3 ประเภท คือ

- ความสำคัญสูง (High Priority)
- ความสำคัญปานกลาง (Medium Priority)
- ความสำคัญต่ำ (Low Priority)

ความสำคัญของความสำคัญของกลุ่มข้อมูลคือ กลุ่มข้อมูลที่มีความสำคัญสูงจะถูกส่งก่อนหน้ากลุ่มข้อมูลที่มีความสำคัญปานกลาง และ กลุ่มข้อมูลที่มีความสำคัญปานกลางจะถูกส่งก่อนหน้ากลุ่มข้อมูลที่มีความสำคัญต่ำ

การรองรับความน่าเชื่อถือของกลุ่มข้อมูล

- ความน่าเชื่อถือต่ำ (Unreliable)

กลุ่มข้อมูลประเภทนี้จะถูกส่งผ่านโปรโตคอล UDP ซึ่งอาจจะถึงจุดหมายไม่เรียงตามลำดับหรือไม่เรียงเลย ซึ่งข้อมูลชนิดนี้จะดีสำหรับข้อมูลที่ไม่สำคัญ หรือข้อมูลที่ส่งด้วยความถี่สูงดังนั้นหากบางกลุ่มข้อมูลหล่นหาย กลุ่มข้อมูลใหม่จะชดเชยได้

  - ข้อดี กลุ่มข้อมูลชนิดนี้ไม่จำเป็นที่จะต้องรอรับการ Acknowledged จากเครือข่าย ลดขนาด header ของ UDP ประมาณ 50 ไบท์
  - ข้อเสีย กลุ่มข้อมูลจะไม่เรียงลำดับ กลุ่มข้อมูลอาจสูญหายระหว่างทาง

- ความน่าเชื่อถือต่ำ แต่ถูกส่งแบบเรียงเป็นลำดับ (Unreliable Sequenced)

กลุ่มข้อมูลประเภทนี้มีลักษณะเหมือนกับ กลุ่มข้อมูลความน่าเชื่อถือต่ำ เว้นแต่จะกรองกลุ่มข้อมูลที่เก่าออกไป

- ข้อดี ใช้ header ของ UDP น้อยเหมือนกลุ่มข้อมูลความน่าเชื่อถือต่ำ และไม่ต้องกังวลว่า ข้อมูลจะถูกเปลี่ยนแปลงซ้ำเพราะลำดับของกลุ่มข้อมูลผิดไป
- ข้อเสีย กลุ่มข้อมูลยังหล่นหายระหว่างทางได้ ถ้าเราทำการส่งข้อมูลยกเลิก แล้วไม่ถึงอาจทำให้เกิดปัญหากับโปรแกรมได้

- ความน่าเชื่อถือสูง (Reliable)

กลุ่มข้อมูลประเภทนี้จะถูกติดตามด้วย reliability layer เพื่อที่จะส่งให้ถึงปลายทาง

- ข้อดี กลุ่มข้อมูลที่ส่งถึงปลายทางอย่างแน่นอน
- ข้อเสีย จะมีการ retransmission และ acknowledge ทำให้เสียขนาดของช่องสัญญาณบางส่วน และไม่มีการลำดับกลุ่มข้อมูลให้

- ความน่าเชื่อถือสูง และถูกส่งแบบเรียงเป็นลำดับ (Reliable Ordered)

กลุ่มข้อมูลประเภทนี้จะถูกติดตามด้วย reliability layer เพื่อที่จะส่งให้ถึงปลายทาง และเรียงลำดับตามที่ถูกส่งออกจากต้นทาง

- ข้อดี กลุ่มข้อมูลถูกส่งถึงปลายทางแน่นอนและเรียงลำดับตามที่ถูกส่งออกไปด้วย ทำให้เขียนโปรแกรมง่าย เพราะไม่ต้องกังวงเกี่ยวกับลำดับ และการหล่นหายของข้อมูล
- ข้อเสีย ข้อเสีย จะมีการ retransmission และ acknowledge ทำให้เสียขนาดของช่องสัญญาณบางส่วน และถ้ามีบางกลุ่มข้อมูลส่งถึงช้าจะทำให้ข้อมูลที่ตามมาหน่วงเวลาถึงไปด้วย

- ความน่าเชื่อถือสูง และถูกส่งแบบรับเป็นลำดับ (Reliable Sequenced)

กลุ่มข้อมูลประเภทนี้จะถูกติดตามด้วย reliability layer เพื่อที่จะส่งให้ถึงปลายทาง และรับตามลำดับตามที่ถูกส่งออกจากต้นทาง ถ้ากลุ่มข้อมูลไหนมาถึงช้ากว่าข้อมูลอื่นๆ ในลำดับนั้นจะถูกตัดทิ้งออกไปทันที

- ข้อดี กลุ่มข้อมูลถูกส่งถึงปลายทางแน่นอนและเรียงลำดับตามที่ถูกส่งออกไปด้วย ทำให้เขียนโปรแกรมง่าย เพราะไม่ต้องกังวงเกี่ยวกับลำดับ และการหล่นหายของข้อมูล และไม่ต้องรอกลุ่มข้อมูลที่ยังส่งไม่ถึงอีกด้วย
- ข้อเสีย ข้อเสีย จะมีการ retransmission และ acknowledge ทำให้เสียขนาดของช่องสัญญาณบางส่วน และกลุ่มข้อมูลไหนมาถึงช้ากว่าข้อมูลอื่นๆ ในลำดับนั้นจะถูกตัดทิ้งออกไปทันที

ตัวอย่างการส่งข้อมูลด้วยการรองรับความน่าเชื่อถือของกลุ่มข้อมูลประเภทต่างๆ

เมื่อส่งข้อมูล 1, 2, 3, 4, 5, 6 จะได้ข้อมูลตามลักษณะเฉพาะของการรองรับความน่าเชื่อถือของกลุ่มข้อมูลแต่ละประเภท ดังนี้

- ความน่าเชื่อถือต่ำ (Unreliable) อาจรับข้อมูลได้เป็น [5, 1, 6]
- ความน่าเชื่อถือต่ำ แต่ถูกส่งแบบเรียงเป็นลำดับ (Unreliable Sequenced) อาจรับข้อมูลได้เป็น [5]
- ความน่าเชื่อถือสูง (Reliable) อาจรับข้อมูลได้เป็น [5, 1, 4, 6, 2, 3]
- ความน่าเชื่อถือสูง และถูกส่งแบบเรียงเป็นลำดับ (Reliable Ordered) อาจรับข้อมูลได้เป็น [1, 2, 3, 4, 5, 6]
- ความน่าเชื่อถือสูง และถูกส่งแบบรับเป็นลำดับ (Reliable Sequenced) อาจรับข้อมูลได้เป็น [5, 6]

บทที่ 3

รายละเอียดการทำงานของระบบ

3.1 ภาพรวมของระบบ

ลักษณะโดยทั่วไปของเกม

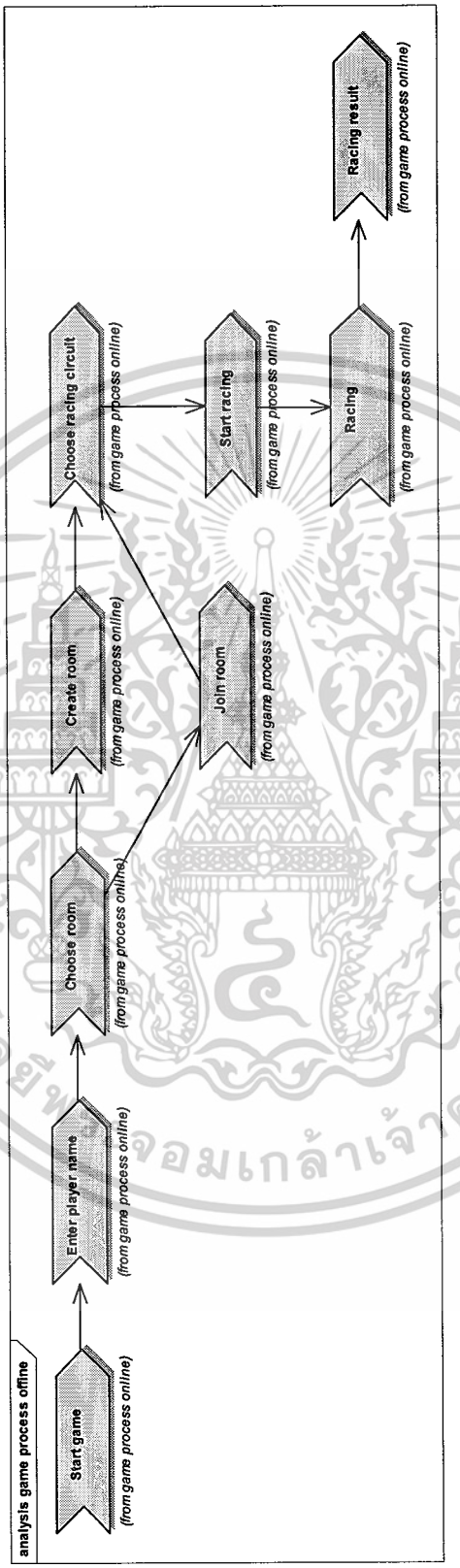
เกมที่สร้างขึ้นนั้นเป็นเกมแนวการแข่งขันความเร็ว ซึ่งใครผู้เล่นคนใดสามารถวิ่งไปถึงเส้นชัยได้ก่อนถือเป็นผู้ชนะ ซึ่งลักษณะโดยทั่วไปของเกมนั้นมีดังนี้

1 ระบบรองรับให้ผู้เล่นสามารถเลือกตัวละครเพื่อเป็นตัวแทนของตนภายในเกมได้ ซึ่งตัวละครแต่ละตัวจะมีคุณสมบัติแตกต่างกันไป

2 ผู้ใช้สามารถเลือกสนามในการแข่งขันได้ โดยแต่ละสนามจะมีเส้นทางการแข่งขัน สิ่งกีดขวางแตกต่างกันไป

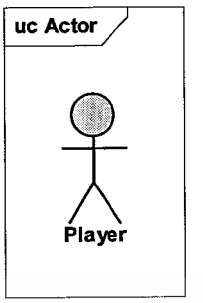
3 ภายในเกมจะมีการตั้งจุดตรวจเพื่อจะได้รู้ว่าผู้เล่นคนนั้นๆ ได้ทำการวิ่งผ่านได้ครบทั้งสนามรึเปล่า

การวิเคราะห์ระบบด้วย Use case



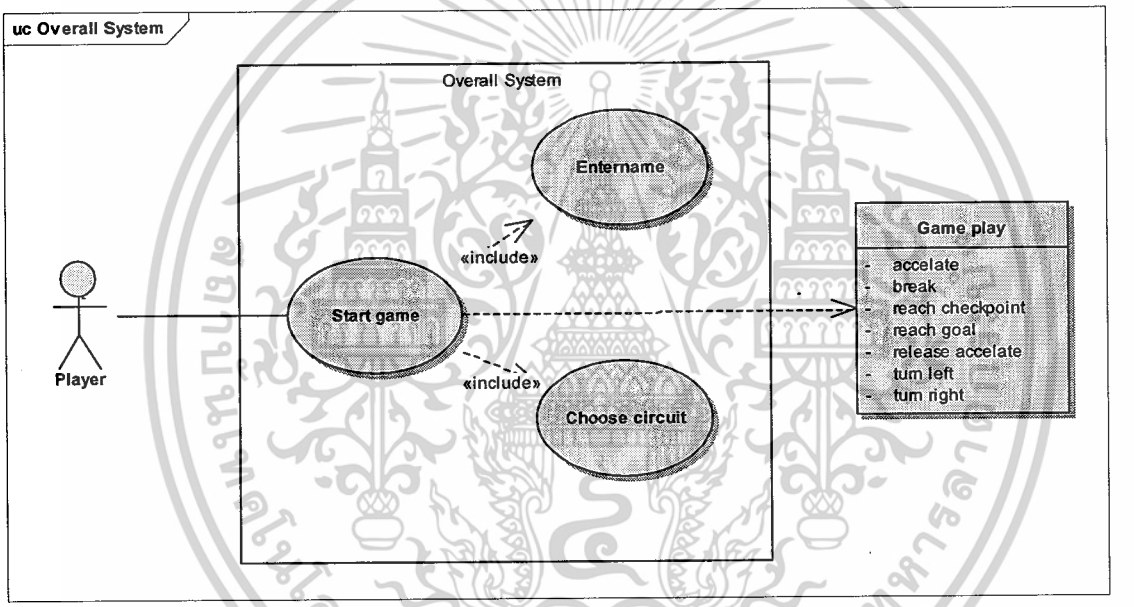
รูปที่ 3.1 Business model of online mode

1 Actor



รูปที่ 3.2 แผนภาพการใช้งานระบบในส่วนของผู้ใช้ระบบ

2 Overall of system



รูปที่ 3.3 แผนภาพการใช้งานระบบโดยทั่วไป

2.1 Start game

ตารางที่ 3.1 แสดงรายละเอียด Usecase Startgame

Context of use	เข้าเกมและกดปุ่มเริ่มเพื่อทำการเริ่มเกม
Level	ส่วนหลัก
Primary actor	Player
Stakeholder	-
Pre-condition	เมื่อผู้เล่นได้ทำการใส่ชื่อและเลือกสนาม
Minimal guarantee	-
Success guarantee	เริ่มการเล่นเกม
Trigger	กดปุ่มเริ่ม
Use case relation	ยูสเคส Choose circuit เป็นการเลือกสนามแข่ง
	ยูสเคส Entername เป็นการใส่ชื่อผู้เล่น
Flow of event	1) โหลดข้อมูลพื้นฐานของเกม เช่น ที่เก็บไฟล์ของตัวละคร ที่เก็บไฟล์ของสนามแข่ง
	2) ผู้เล่นทำการใส่ชื่อเพื่อเป็นการลงทะเบียน
	3) ผู้เล่นทำการเลือกสนามแข่ง
	4) โหลดข้อมูลของสนามแข่งที่ผู้เล่นเลือกและทำการสร้างสนามขึ้นมา
	5) เริ่มเกม
Extension	2ก) ในกรณีที่การเชื่อมต่อไม่สำเร็จระบบจะไม่ให้ผู้เล่นผ่านไปยังหน้าเลือกสนามแข่งได้

2.2 Choose circuit

ตารางที่ 3.2 แสดงรายละเอียด Usecase Choose circuit

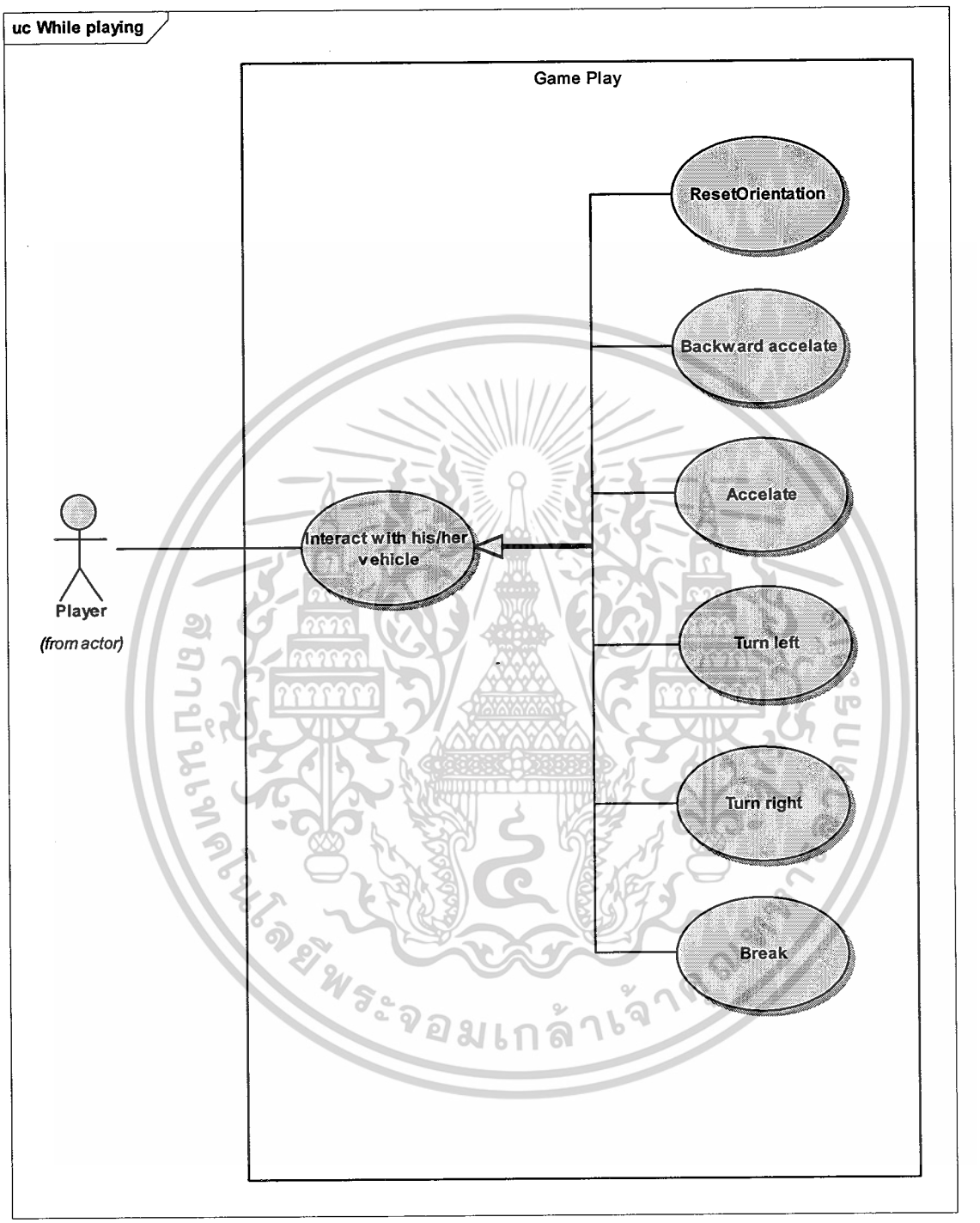
Context of use	ในขั้นตอนการเริ่มเกมผู้เล่นต้องทำการเลือกสนาม
Level	ฟังก์ชั่นย่อย
Primary actor	Player
Stakeholder	-
Pre-condition	เมื่อผู้เล่นลงทะเบียนเรียบร้อย
Minimal guarantee	-
Success guarantee	มีการเลือกสนาม
Trigger	เมื่อมีการใส่ชื่อเพื่อลงทะเบียนผู้เล่นเรียบร้อย
Use case relation	ยูสเคส Entername เพื่อลงทะเบียน
Flow of event	6) ทำการเลือกสนาม
	7) แสดงภาพของสนามที่ผู้เล่นเลือก
	8) ทำการเลือกสนาม
	9) ระบบจะทำการส่งรหัสของสนามให้แม่ข่ายรับรู้
	10) ทำการโหลดข้อมูลของสนามที่ผู้เล่นเลือก
	11) ทำการสร้างสนามตามข้อมูล ข้อมูลของสนามจะถูกเก็บเป็นไฟล์ XML
Extension	1ก) หากผู้เล่นกดปุ่มยกเลิกระบบก็จะส่งกลับไปยังหน้าก่อนหน้า

2.3 Entername

ตารางที่ 3.3 แสดงรายละเอียด Usecase Entername

Context of use	ใส่ชื่อเพื่อลงทะเบียนกับแม่ข่าย
Level	ฟังก์ชั่นย่อย
Primary actor	Player
Stakeholder	-
Pre-condition	เมื่อกดปุ่มเข้าเกม
Minimal guarantee	-
Success guarantee	ลงทะเบียนเรียบร้อย
Trigger	เมื่อกดปุ่มเข้าเกม
Use case relation	ยูสเคส start เพื่อเริ่มเกม
Flow of event	1) ทำการใส่ชื่อ
	2) กดปุ่มตกลงเพื่อละทะเบียน
Extension	1ก) หากผู้เล่นกดปุ่มยกเลิกระบบก็จะส่งกลับไปยังหน้าก่อนหน้า

3 Game play



รูปที่ 3.4 แผนภาพการใช้งานระบบในระบบการเล่นเกม

3.1 Interact with his/her vehicle

ตารางที่ 3.4 แสดงรายละเอียด Usecase Interact with his/her vehicle

Context of use	ผู้เล่นควบคุมรถเพื่อผ่านจุดตรวจต่างๆ เพื่อไปยังเส้นชัย
Level	โดยรวม
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	-
Success guarantee	-
Trigger	เมื่อผู้เล่นกดปุ่มบังคับรถ
Use case relation	ยูสเคส Start game
Flow of event	3) ผู้เล่นทำการเลือกการบังคับรถ มี 6 อย่าง คือ ก. เลี้ยวซ้าย ข. เลี้ยวขวา ค. เร่ง ง. ปล่อยคันเร่ง จ. หยุด ฉ. ทำการปรับองศาของรถให้เป็นค่าเริ่มต้น
	4) ส่งข้อมูลการบังคับให้แม่ข่าย แม่ข่ายจะทำหน้าที่คำนวณเพื่อหาตำแหน่งและทิศทางต่อไปของรถ
	5) แม่ข่ายส่งข้อมูลกลับมาให้ลูกข่าย ลูกข่ายจะนำข้อมูลมาทำการเปลี่ยนแปลงให้ตรงกับที่แม่ข่ายส่งมา
	6) แม่ข่ายจะทำการเก็บข้อมูลของการผ่านจุดตรวจ หากผู้เล่นผ่านจุดตรวจได้ครบ แล้วจึงสามารถเข้าเส้นชัยได้

3.2 Turn left

ตารางที่ 3.5 แสดงรายละเอียด Usecase Turn left

Context of use	เมื่อผู้เล่นต้องการเลี้ยวซ้าย
Level	สำคัญ
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	แรงการขับเคลื่อนไปทางซ้ายมากกว่า
Success guarantee	เมื่อรถเคลื่อนที่ไปทางซ้าย
Trigger	เมื่อผู้เล่นกดปุ่ม A หรือลูกศรซ้าย
Use case relation	ยูสเคส Interact with his/her vehicle

3.3 Turn right

ตารางที่ 3.6 แสดงรายละเอียด Usecase Turn right

Context of use	เมื่อผู้เล่นต้องการเลี้ยวขวา
Level	สำคัญ
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	แรงการขับเคลื่อนไปทางขวามากกว่า
Success guarantee	เมื่อรถเคลื่อนที่ไปทางขวา
Trigger	เมื่อผู้เล่นกดปุ่ม D หรือลูกศรขวา
Use case relation	ยูสเคส Interact with his/her vehicle

3.4 Accelerate

ตารางที่ 3.7 แสดงรายละเอียด Usecase Accelerate

Context of use	เมื่อผู้เล่นต้องการเร่งความเร็ว
Level	สำคัญ
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	แรงการทำกับล้อเพิ่มขึ้น
Success guarantee	รถเคลื่อนไปข้างหน้า
Trigger	เมื่อผู้เล่นกดปุ่ม W หรือลูกศรขึ้น
Use case relation	ยูสเคส Interact with his/her vehicle

3.5 Backward accelerate

ตารางที่ 3.8 แสดงรายละเอียด Usecase Backward accelerate

Context of use	เมื่อผู้เล่นต้องการถอยหลัง
Level	สำคัญ
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	เพิ่มแรงกระทำกับล้อเพื่อให้ถอยหลัง
Success guarantee	รถวิ่งถอยหลัง
Trigger	เมื่อผู้เล่นกดปุ่ม S หรือลูกศรลง
Use case relation	ยูสเคส Interact with his/her vehicle

3.6 Break

ตารางที่ 3.9 แสดงรายละเอียด Usecase Break

Context of use	เมื่อผู้เล่นต้องการหยุดรถ
Level	สำคัญ
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	รถลดความเร็ว
Success guarantee	รถหยุด
Trigger	เมื่อผู้เล่นกดปุ่ม spacebar
Use case relation	ยูสเคส Interact with his/her vehicle

3.7 ResetOrientation

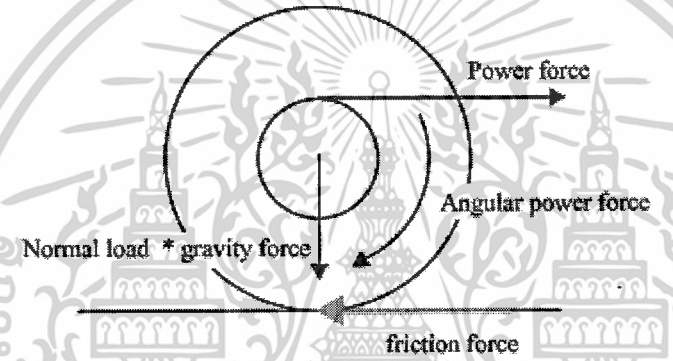
ตารางที่ 3.10 แสดงรายละเอียด Usecase ResetOrientation

Context of use	เมื่อผู้เล่นต้องการกลับรถให้ตั้งเหมือนเดิม
Level	สำคัญ
Primary actor	Player
Stakeholder	-
Pre-condition	ขณะที่เกมกำลังเล่น
Minimal guarantee	-
Success guarantee	รถพลิกกลับมาหงายเหมือนเดิม
Trigger	เมื่อผู้เล่นกดปุ่ม enter
Use case relation	ยูสเคส Interact with his/her vehicle

3.2 การควบคุมการเคลื่อนที่ของตัวละคร

หลักการทางฟิสิกส์ของยานพาหนะแบบล้อ

ในการขับเคลื่อนพหานะที่มีล้อนั้นจะเกิดจากการสร้างแรงไปในทิศทางที่ต้องการกับล้อรถ เพื่อให้ล้อนั้นหมุนไปในทิศทางตามนั้น ซึ่งแรงที่กระทำนั้นเป็นแรงในแนวเส้นตรงซึ่งแรงที่กระทำกับล้อนั้นจะก่อให้เกิดแรงเชิงมุม (Torque) ซึ่งแรงที่กระทำนั้นขึ้นอยู่กับกำลังของสิ่งที่ขับเคลื่อนพาหนะ เช่น เครื่องยนต์ของรถ แรงผลักสำหรับสเก็ตบอร์ด เป็นต้น แต่ในพาหนะจริง ๆ นั้นสิ่งที่ขับเคลื่อนล้อไม่จำเป็นต้องขับเคลื่อนล้อทุกล้อของพาหนะก็ได้ และเมื่อล้อเสียดสีกับพื้นจะเกิดแรงเสียดทานทำให้เกิดแรงในทิศทางตรงกันข้ามกับแรงขับเคลื่อนดังรูป



รูปที่ 3.5 แรงที่กระทำกับล้อโดยตรง

ขนาดของแรงเสียดทานที่กระทำกับล้อนั้นจะมีขนาดเท่ากับ $\mu * Normal\ force$ เมื่อ u คือสัมประสิทธิ์แรงเสียดทานสถิต (จะถูกกำหนดขึ้นตามลักษณะของสนาม) และ Normal force คือน้ำหนักที่กดทับล้อในทิศทางตั้งฉากกับทิศาทางการเคลื่อนที่

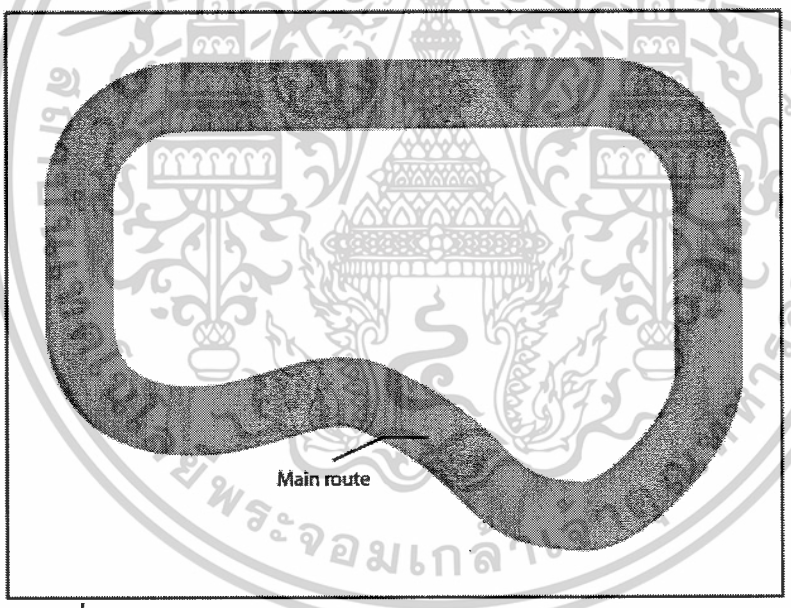
การหมุนของล้อนั้นนอกจากจะทำให้พาหนะเคลื่อนที่ไปตามแรงลัพธ์ที่กระทำแล้วอาจจะก่อให้เกิดการลื่นไถล ซึ่งความเร็วของการไถลนั้นขึ้นอยู่กับความเร็วของล้อรถและค่าสัมประสิทธิ์ความเสียดทานสถิตของยางรถและพื้นผิวที่สัมผัส แต่ในการจำลองในเกมนั้นจะตัดปัจจัยพื้นที่ผิวสัมผัสที่แตกต่างกันออกไป

3.3 สนามแข่งขัน

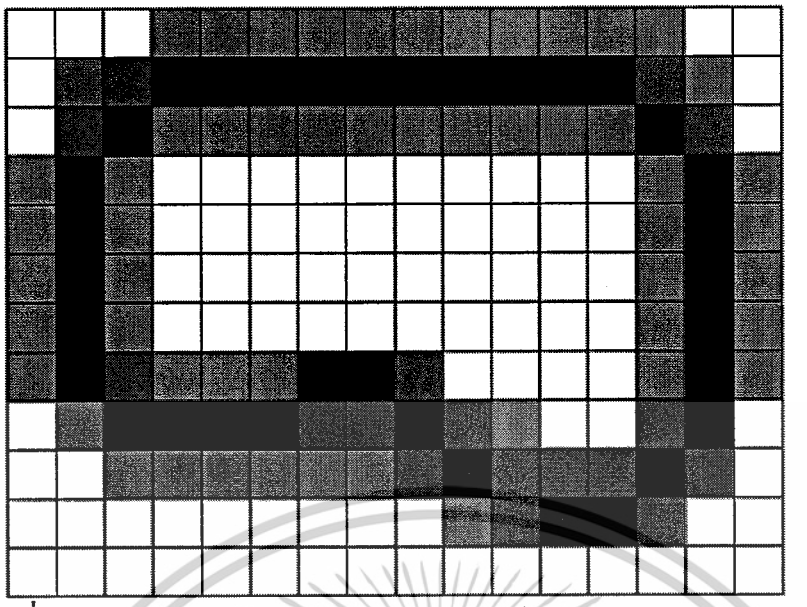
สนามในการแข่งขันนั้นจะคล้ายคลึงกับสนามแข่งรถทั่วไป คือสนามจะประกอบไปด้วยการเส้นทางในการแข่งขัน แต่ผู้เล่นก็สามารถวิ่งออกนอกสนามแข่งขันได้แต่จะทำให้ค่าความเสียดทานระหว่างล้อกับสนามเพิ่มขึ้น และเส้นทางการแข่งขันจะเป็นเส้นทางบรรจบกันเพื่อที่จะทำให้การแข่งขันนั้นสามารถวิ่งวนหลายรอบ ดังนั้นเมื่อนำหลักของสนามแข่งขันทั่วไปนำมาใช้ในเกมนั้น จะใช้เทคนิคต่าง ๆ จำลองให้เสมือนการแข่งขันจริง

การแบ่งเส้นทางการแข่งขัน

เพื่อที่จำลองการแบ่งเส้นทางหลักกับพื้นที่อื่น ๆ สนามจะถูกแบ่งออกเป็นช่องสี่เหลี่ยมขนาดเล็ก ซึ่งแต่ละช่องนั้นจะมีค่าสัมประสิทธิ์ความเสียดทานที่แตกต่างกัน โดยค่าในการแบ่งสนามนั้นจะแบ่งในแนวระนาบสองมิติตามแกน xz เท่านั้น ดังนั้นในพื้นที่มีการซ้อนทับในบนแกน y จะให้ถือว่ามีสัมประสิทธิ์ค่าแรงเสียดทางที่เท่ากัน



รูปที่ 3.6 ตัวอย่างเส้นทางหลักในสนามแข่งขันในระนาบสองมิติ

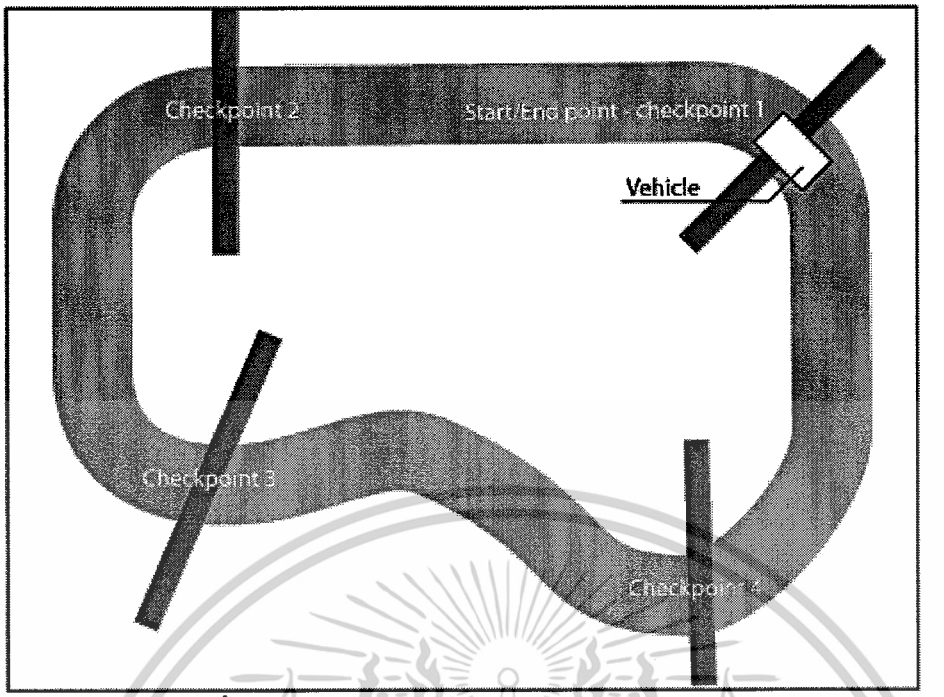


รูปที่ 3.7 ตัวอย่างในการแบ่งสนามแข่งขันจากเส้นทางการแข่งขันในรูป 1

ในรูปที่ 2 นั้นแสดงถึงการลักษณะการแบ่งสนามออกเป็นช่องสี่เหลี่ยม โดยที่สีในแต่ละช่องจะแสดงถึงค่าสัมประสิทธิ์ความเสียดทานที่แตกต่างกัน โดยให้สีเข้มแทนถึงค่าที่น้อย และสีขาวจะแทนค่ามากโดยหลักการในการแทนค่าลงในตารางนั้นจะให้เส้นทางหลักมีค่าน้อยเพื่อให้รถที่วิ่งไปตามทางไปได้เร็วกว่าการออกนอกเส้นทาง ซึ่งในแทนค่าสัมประสิทธิ์ความเสียดทานของสนามภายในโปรแกรมจะแทนด้วยเมทริกซ์ แล้วจึงทำการเชื่อมโยงแต่ละค่าในเมทริกซ์นั้นเข้ากับสนามจริง โดยระบบจะไม่ได้กำหนดอัตราส่วนระหว่างสนามต่อขนาดของเมทริกซ์อย่างตายตัว เพื่อให้ผู้สร้างข้อมูลสนามมีอิสระในการเลือกอัตราส่วนดังกล่าวอย่างเหมาะ ทำให้ระบบจะต้องตรวจสอบอัตราส่วนนี้เอง และการกำหนดค่าสัมประสิทธิ์แรงเสียดทานนั้นจะต้องเป็นค่าที่มากกว่าศูนย์ และในแต่ละสนามจะมีค่าสัมประสิทธิ์ความเสียดทานแตกต่างกันไป และสามารถเปลี่ยนแปลงได้จากไฟล์ข้อมูลสนาม และจะมีค่าแตกต่างกันไม่เกิน 4 ค่าเพื่อความเหมาะสมในการเก็บข้อมูลลงในไฟล์

การตรวจจับการวิ่งของยานพาหนะ

ในการตรวจสอบเวลาการวิ่งของยานพานะ เพื่อตัดสินหาผู้ชนะนั้นจะใช้การวางจุดตรวจสอบเป็นรูปทรงสี่เหลี่ยมภายในสนาม ทำการตรวจสอบว่าผู้เล่นนั้นได้วิ่งไปรอบ ๆ สนามจริงและได้วิ่งวนรอบสนามครบสามรอบตามที่กติกาได้ระบุไว้ โดยจุดตรวจสอบจะถูกจัดเรียงไว้เป็นลำดับ และผู้เล่นจะต้องวิ่งผ่านจุดตรวจสอบเหล่านี้เป็นลำดับเช่นกัน จะข้ามไปยังจุดตรวจสอบอื่นไม่ได้ (เช่น การวิ่งสวนทางกับเส้นทางที่กำหนด)



รูปที่ 3.8 ตัวอย่างการวางจุดตรวจสอบภายในสนาม

จากรูปนั้นในสนามนี้จะมีจุดตรวจสอบทั้งหมด 4 จุด โดยเรียงลำดับตามตัวเลขที่กำหนดในชื่อจุดตรวจสอบแต่ละอัน ซึ่งการกำหนดจากไฟล์นั้นจะกำหนดแค่ 4 จุดนี้ แต่ในการวางจุดตรวจสอบในขณะเล่นเกมจริง จะต้องมีการเรียงลำดับจุดเหล่านี้ใหม่เพื่อตรวจสอบการวิ่งของรถให้ครบ 3 รอบ จึงต้องมีการเรียงลำดับจุดตรวจสอบใหม่ดังนี้

Start point – Intermediate points – Start point – Intermediate points – Start point – Intermediate points – Start point

ตัวอย่างการเรียงลำดับจุดตรวจสอบในขณะเล่นเกมจากรูป 3 นั้น สามารถเรียงใหม่คือ 1 2 3 4 1 2 3 4 1 2 3 4 1 ดังนี้เมื่อผู้เล่นได้วิ่งผ่านจุดตรวจสอบที่ 1 ไปแล้ว จะส่งผลให้จุดตรวจสอบต่อไปจุดตรวจสอบที่ 2 จนกระทั่งได้วิ่งผ่านจุดตรวจสอบทั้งหมด

3.4 การจัดเก็บข้อมูลการปรับแต่งภายในเกม

ชนิดของข้อมูลที่ใช้ในระบบ

ข้อมูลต่าง ๆ จะถูกจัดเป็นเก็บเป็นหมวดหมู่โดยแยกตามการจัดเก็บข้อมูลทางกายภาพ คือ การจัดแยกตามระบบไฟล์ หรือจัดกลุ่มไฟล์ที่เก็บข้อมูลชนิดเดียวกันอยู่ในแฟ้มข้อมูล (directory) ซึ่งชนิดของข้อมูลที่ต้องจัดเก็บมีดังนี้

- ข้อมูลการปรับแต่งระบบพื้นฐาน (basic configuration file) คือข้อมูลในการบ่งชี้ตำแหน่งไฟล์ข้อมูลต่าง ๆ ที่จะนำมาใช้ภายในระบบ โดยในระบบหนึ่ง ๆ นั้นจะมีไฟล์ข้อมูลชนิดนี้เพียงไฟล์เดียวเท่านั้นและจะถูกจัดเก็บในตำแหน่ง “”
- ข้อมูลตัวละคร คือข้อมูลพื้นฐานของตัวละครต่าง ๆ ที่มีภายในเกม จะถูกใช้เมื่อมีการนำมาสร้างตัวละครขึ้น และการแสดงผลตัวละครต่าง ๆ ก่อนเริ่มเกม โดยที่ไฟล์ชนิดนี้สามารถมีได้หลายไฟล์ และจะถูกจัดเก็บในแฟ้มข้อมูลเดียวกันตามตำแหน่งที่ระบุไว้ในข้อมูลการปรับแต่งพื้นฐานของระบบ
- ข้อมูลสนาม คือข้อมูลของสนามต่าง ๆ ที่มีภายในเกม เพื่อใช้ในการสร้างสนามภายในเกม โดยที่ไฟล์ชนิดนี้สามารถมีได้หลายไฟล์ และจะถูกจัดเก็บในแฟ้มข้อมูลเดียวกันตามตำแหน่งที่ระบุไว้ในข้อมูลการปรับแต่งพื้นฐานของระบบ
- ข้อมูลการปรับแต่ง Ogre Engine เป็นข้อมูลที่ใช้ในการปรับแต่งการทำงานต่าง ๆ ของ ogre engine โดยลักษณะของข้อมูลนั้นจะถูกกำหนดตามที่ ogre engine กำหนดมา โดยกลุ่มข้อมูลนี้จะประกอบด้วย 3 ไฟล์ด้วยกันคือ
    - Ogre configuration file เป็นไฟล์ที่บอกลักษณะในการแสดงผลทางกราฟฟิคต่าง ๆ เช่น ความละเอียดและขนาดของหน้าจอ ลักษณะการแสดงผลหน้าต่าง เครื่องมือในการแสดงผลกราฟฟิค เป็นต้น ซึ่งผู้ใช้จะต้องสามารถที่จะเปลี่ยนแปลงข้อมูลเหล่านี้ให้เหมาะสมกับเครื่องคอมพิวเตอร์ที่ใช้งานผ่านทางระบบได้เอง
    - Plug-in file เป็นไฟล์ที่ระบุชื่อชุดคำสั่งต่าง ๆ ที่ช่วยในการทำงานของ ogre engine เพิ่มเติมจากชุดคำสั่งพื้นฐาน
    - Resource configuration file เป็นไฟล์ระบุถึงตำแหน่งของไฟล์ทรัพยากรที่เป็นข้อมูลที่นำมาใช้การแสดงผล ได้แก่ ไฟล์เก็บข้อมูล mesh ต่าง ๆ , ไฟล์ที่บรรยายลักษณะในการแสดงผลติดต่อกับผู้ใช้, ไฟล์ที่บรรยายลักษณะของ particle เป็นต้น

ลักษณะของข้อมูลตัวละคร

ข้อมูลของตัวละครนั้นเป็นเสมือนข้อมูลต้นแบบในการสร้างตัวละครแต่ละตัวภายในระบบ ซึ่งข้อมูลที่จะต้องจัดเก็บมีดังนี้

- ข้อมูลทางด้านฟิสิกส์ของพาหนะ คือข้อมูลที่จะนำไปเป็นตัวแปร (parameter) ให้แก่ระบบฟิสิกส์ในเกม เพื่อทำให้ผู้เล่นได้เลือกลักษณะในการควบคุมพาหนะที่เหมาะกับตัวเองต่อไป ซึ่งข้อมูลที่จัดเก็บมีดังต่อไปนี้
  - น้ำหนักของยานพาหนะ
  - ตำแหน่งของล้อต่าง ๆ และล้อที่ควบคุมทิศทาง
  - กำลังในการขับเคลื่อนล้อ
  - ความเร็วในการเปลี่ยนทิศทาง
  - ความต้านทานการลื่นไถลของล้อทั้งทางตรงและทางเลี้ยว
- ข้อมูลทางด้านการแสดงผลกราฟฟิค คือข้อมูลเกี่ยวกับรูปร่างของตัวละครในการแสดงผลกราฟฟิคในเกม ซึ่งข้อมูลชุดนี้ประกอบด้วย
  - ชื่อของรูปร่างตัวละคร
  - คำอธิบายรูปร่างตัวละคร
  - ชื่อไฟล์ mesh ของส่วนตัวรถของตัวละคร
  - คุณลักษณะของตัวรถ
  - ชื่อไฟล์ mesh ของล้อ ซึ่งจะมีการแบ่งข้อมูลแยกออกเป็นล้อๆ ไป จึงทำให้สามารถทำการปรับแต่งให้มีล้อมีความแตกต่างกันได้ในรถคันเดียวกัน และยังสามารถแสดงผลได้ถูกต้องตามองศาของล้อได้
  - คุณลักษณะของล้อ

ลักษณะของข้อมูลสนาม

ข้อมูลของสนามนั้นเปรียบเสมือนต้นแบบในการสร้างสนามแข่งขันให้แก่ผู้เล่น ซึ่งข้อมูลเหล่านี้ได้จัดเก็บข้อมูลดังต่อไปนี้

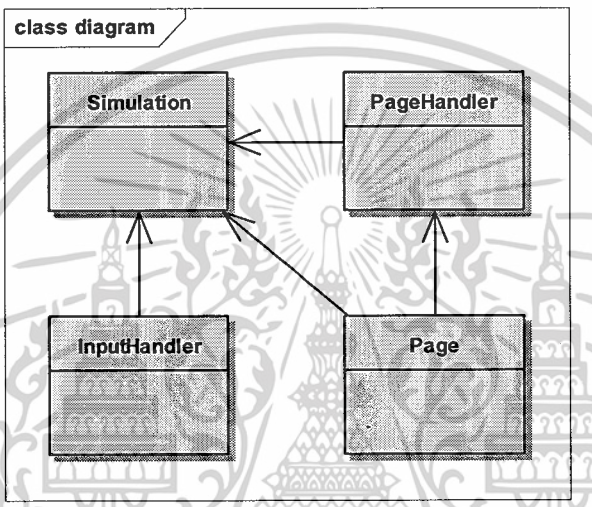
- ข้อมูลการแสดงผลกราฟฟิค คือข้อมูลเกี่ยวกับรูปร่างของสนามแข่งขันและสิ่งกีดขวางต่าง ๆ ที่วางอยู่ในสนามแข่งขัน ซึ่งประกอบด้วยข้อมูลเหล่านี้
  - ชื่อสนาม
  - คำอธิบายสนาม
  - ตำแหน่งตั้งต้นของตัวละครแต่ละตัว
  - ชื่อไฟล์ mesh ของสนามแข่งขัน
  - ลักษณะของท้องฟ้า

- ลักษณะของแสงและตำแหน่งที่ใช้
- ตำแหน่งของการวางจุดตรวจ
- สิ่งกีดขวางในการแข่งขัน
  - ชื่อไฟล์ mesh ของสิ่งกีดขวาง
  - รูปร่างและขนาดของสิ่งกีดขวาง เพื่อใช้ในการสร้างขอบเขตไม่ให้ผู้เล่นวิ่งผ่านสิ่งกีดขวางนี้
  - ตำแหน่งสิ่งกีดขวาง (สามารถมีได้หลายตำแหน่ง ถ้าสิ่งกีดขวางนั้นมีหลายชิ้น)

บทที่ 4
การออกแบบโครงสร้างระบบ

4.1 ส่วนลูกข่าย(Client)

4.1.1 ส่วนการแสดงผลติดต่อผู้ใช้



รูปที่ 4.1 Class diagram of GUI Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.1 แสดงรายละเอียดคลาส Page

Class name	Description		
Page	เป็นคลาสที่ทำหน้าที่สร้างอินเตอร์เฟสต่างๆ และจัดการผูกเหตุการณ์ของอินเตอร์เฟสนั้นๆ กับอินเตอร์เฟสให้ถูกต้อง		
Attribute name	Scope	Type	Description
mCInfo	Private	Network::ClientInfo	ข้อมูลผู้เล่น
mRInfo	Private	Network::RoomInfo	ข้อมูลห้อง
mNetwork	Private	Network::ClientNetwork	จัดการเครือข่าย
mEnterNamePage	Private	EnterNamePageHandler	PageHandler
mMainPage	Private	MainPageHandler	PageHandler

ตารางที่ 4.1 (ต่อ)

mRoomListPage	private	RoomListPageHandler	PageHandler
mSelectMapPage	private	SelectMapPageHandler	PageHandler
mCreateRoomPage	private	CreateRoomPageHandler	PageHandler
mSystem	private	CEGUI::System	ระบบ GUI
mPageLayout	private	CEGUI::Window	Layout ของ GUI
mSim	private	Simulation	จัดการสถานะ
Method name	Scope	Description	
getPageHandler()	public	ดึง PageHandler จากGUI	
loadDefaultConfig()	public	กำหนดค่าเริ่มต้นของอินเตอร์เฟส	
loadDefaultFont()	public	กำหนดค่าเริ่มต้นของฟอนต์ในอินเตอร์เฟส	
loadDefaultMouse()	public	กำหนดค่าเริ่มต้นของเม้าในอินเตอร์เฟส	
loadDefaultScheme()	public	กำหนดค่าเริ่มต้นของอินเตอร์เฟสแรก	
requestPageChange (State)	public	ร้องขอเปลี่ยนสถานะ	
setupHandler(State)	public	ผูกเหตุการณ์เข้ากับอินเตอร์เฟส	

ตารางที่ 4.2 แสดงรายละเอียดคลาส PageHandler

Class name	Description		
PageHandler	มีหน้าที่รับเหตุการณ์การของผู้เล่นจากอินเตอร์เฟส และตอบสนองต่อผู้เล่น เช่น ผู้เล่นกดปุ่ม		
Attribute name	Scope	Type	Description
mPageSimulation	private	Simulation	จัดการสถานะ
mPageSystem	private	CEGUI::System	ระบบ GUI
mPageWindow	private	CEGUI::Window	GUI Page
Method name	Scope	Description	
SubScribe()	public	ลงทะเบียนเหตุการณ์ของอินเตอร์เฟส	

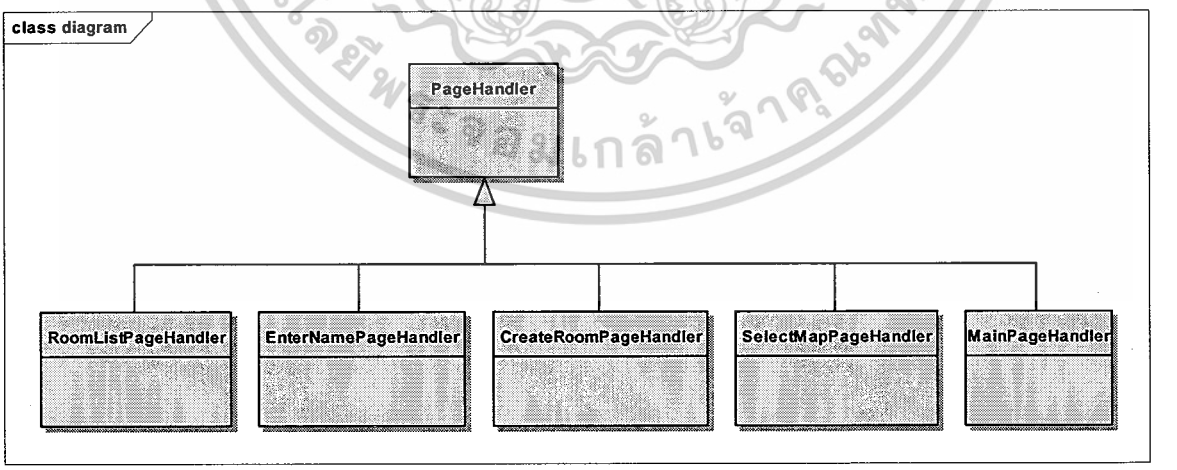
ตารางที่ 4.3 แสดงรายละเอียดคลาส InputHandler

Class name	Description		
InputHandler	มีหน้าที่รับเหตุการณ์จากผู้ใช้เช่น กดคีย์บอร์ด เลื่อนเม้า		
Attribute name	Scope	Type	Description
m_hWnd	private	unsigned long	
m_ois	private	OIS::InputManager	ระบบการรับค่า
m_pSystem	private	CEGUI::System	ระบบ GUI
m_simulation	private	Simulation	จัดการสถานะ
mKeybloard	private	OIS::Keyboard	ระบบจัดการคีย์บอร์ด
mMouse	private	OIS::Mouse	ระบบจัดการเม้าท์
mRInfo	private	Network::RoomInfo	ข้อมูลห้อง
mCInfo	private	Network::ClientInfo	ข้อมูลผู้เล่น
mNetwork	private	Network::ClientNetwork	ระบบเครือข่าย
Method name	Scope	Description	
capture()	public	จับเหตุการณ์ของอุปกรณ์	
setMode (std::string)	public	ตั้งโหมดระหว่าง GUI กับเกม	
setWindowExtend (int,int)	public	เป็นเหตุการณ์ที่เกิดขึ้นเมื่อมีการเปลี่ยนแปลงขนาดหน้าจอ	
keyPressed (OIS::KeyEvent)	public	ดักจับเหตุการณ์เมื่อกดปุ่มคีย์บอร์ด	
keyReleased (OIS::KeyEvent)	public	ดักจับเหตุการณ์เมื่อปล่อยปุ่มคีย์บอร์ด	
modeGame ()	public	ใช้ในการเปี่ยนเป็นโหมดเกม	
modeGUI ()	public	ใช้ในการเปลี่ยนเป็นโหมด GUI	
mouseMoved (OIS::MouseEvent)	public	ดักจับเหตุการณ์ที่เกิดการเลื่อนเม้าท์	
mousePressed (OIS::MouseEvent)	public	ดักจับเหตุการณ์ที่เกิดการคลิ๊กเม้าท์	
mouseReleased (OIS::MouseEvent)	public	ดักจับเหตุการณ์ที่เกิดการปล่อยปุ่มเม้าท์	

ตารางที่ 4.4 แสดงรายละเอียดคลาส Simulation

Class name	Description		
Simulation	มีหน้าที่ในการควบคุมและจัดการกับสถานะของเกมเช่น อินเตอร์เฟสต่างๆ และการเข้าเกม		
Attribute name	Scope	Type	Description
m_frame_time	protected	float	เวลาในการเปลี่ยนเฟรม
m_locked	protected	bool	สถานะที่ถูกจอง
m_state	Protected	State	สถานะ
Method name	Scope	Description	
getCurrentState()	Public	ดึงค่าสถานะปัจจุบัน	
getFrameTime()	Public	ดึงค่าเวลาในการเปลี่ยนเฟรม	
lockState()	Public	ใช้ในการคงสถานะปัจจุบันไว้	
requestStateChange (State)	Public	ใช้ร้องขอในการเปลี่ยนสถานะ	
setFrameTime (float)	Public	ใช้ตั้งค่าเวลาในการเปลี่ยนเฟรม	
unlockState ()	Public	ใช้ในการปลดล็อกให้สามารถเปลี่ยนสถานะได้	

4.1.2 ส่วนการรับเหตุการณ์ที่เกิดขึ้นจากส่วนแสดงผลติดต่อกับผู้ใช้



รูปที่ 4.2 Class diagram of PageHandler Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.5 แสดงรายละเอียดคลาส PageHandler

Class name	Description		
PageHandler	มีหน้าที่รับเหตุการณ์การของผู้เล่น และตอบสนองต่อผู้เล่น เช่น ผู้เล่นกดปุ่ม		
Attribute name	Scope	Type	Description
mPageSimulation	protected	Simulation	จัดการสถานะ
mPageSystem	protected	CEGUI::System	ระบบ GUI
mPageWindow	protected	CEGUI::Window	Layout ของ GUI
Method name	Scope	Description	
subScribe()	Public	ลงทะเบียนเหตุการณ์ของอินเตอร์เฟส	

ตารางที่ 4.6 แสดงรายละเอียดคลาส MainPageHandler

Class name	Description		
MainPagehandler ::PageHandler	มีหน้าที่รับเหตุการณ์ของผู้เล่น และตอบสนองต่อผู้เล่นในหน้าหลัก		
Attribute name	Scope	Type	Description
mNetwork	Private	Network:: ClientNetwork	ระบบเน็ตเวิร์ค
Method name	Scope	Description	
Lunch_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม start เพื่อเริ่มเกม	
Options_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม option เพื่อปรับแต่งค่า	
Quit_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม quit เพื่อออกจากเกม	
subscribe()	Public	ลงทะเบียนเหตุการณ์ของเมนเพจ	

ตารางที่ 4.7 แสดงรายละเอียดคลาส SelectMapPageHandler

Class name	Description		
SelectMapPageHandler	มีหน้าที่รับเหตุการณ์ของผู้เล่น และตอบสนองต่อผู้เล่นในหน้าเลือกแผนที่ ที่จะใช้ในเกม		
Attribute name	**Scope**	**Type**	**Description**
mNetwork	Private	Network::ClientNetwork	ระบบเครือข่าย
mCInfo	Private	Network::ClientInfo	ข้อมูลลูกข่าย
mRInfo	Private	Network::RoomInfo	ข้อมูลห้อง
mCurrentMap	Private	int	รหัสของแผนที่ปัจจุบัน
mMap	Private	CEGUI::DefaultWindow	แผนที่
mClientList	private	CEGUI::MultiColumnList	ลิสใช้เก็บข้อมูลลูกข่ายในห้อง
Method name	**Scope**	**Description**	
addListBoxItem (char,char*)	Public	ใช้ในการใส่ข้อมูลให้กับลิส	
Back_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม back เพื่อเลือกแผนที่ก่อนหน้า	
Cancel_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม cancle เพื่อเลือกออกจากห้อง	
Next_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม next เพื่อเลือกแผนที่ต่อไป	
Refresh_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม refresh เพื่อเรียกดูข้อมูลของลูกข่ายที่อยู่ในห้อง	
Start_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม start เพื่อเริ่มเกม	
subscribe()	Public	ลงทะเบียนเหตุการณ์ของหน้าเลือกแผนที่	

ตารางที่ 4.8 แสดงรายละเอียดคลาส RoomListPageHandler

Class name	Description		
RoomListPageHandler	มีหน้าที่รับเหตุการณ์ของผู้เล่น และตอบสนองต่อผู้เล่นในหน้าเลือกห้องที่จะใช้เล่น		
Attribute name	Scope	Type	Description
mNetwork	Private	Network:: ClientNetwork	ระบบเครือข่าย
mCInfo	Private	Network::ClientInfo	ข้อมูลลูกข่าย
mRInfo	Private	Network::RoomInfo	ข้อมูลห้อง
mRoomList	Private	CEGUI:: MultiColumnList	ลิสใช้เก็บข้อมูลห้อง
Method name	Scope	Description	
addListBoxItem (char*,char)	Public	ใช้ในการใส่ข้อมูลให้กับลิส	
Create_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม create เพื่อสร้างห้อง	
Join_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม join เพื่อเข้าร่วมห้อง	
Refresh_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม refresh เพื่อดูข้อมูลห้อง	
subscribe()	Public	ลงทะเบียนเหตุการณ์ของหน้าเลือกห้อง	

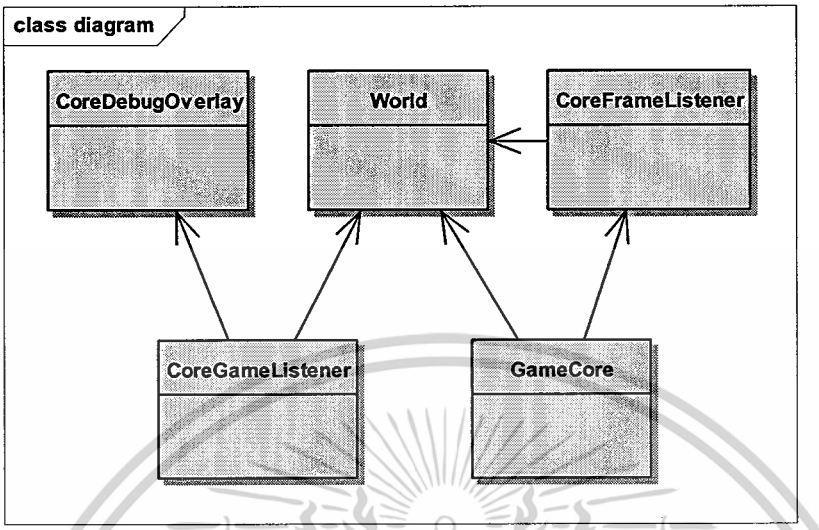
ตารางที่ 4.9 แสดงรายละเอียดคลาส EnterNamePageHandler

Class name	Description		
EnterNamePageHandler	มีหน้าที่รับเหตุการณ์ของผู้เล่น และตอบสนองต่อผู้เล่นในหน้ารับข้อมูลชื่อผู้เล่น		
Attribute name	Scope	Type	Description
mNetwork	private	Network::ClientNetwork	ระบบเครือข่าย
mCInfo	private	Network::ClientInfo	ข้อมูลลูกข่าย
mRInfo	private	Network::RoomInfo	ข้อมูลห้อง
mNameTxt	Private	CEGUI::EditBox	กล่องข้อความรับชื่อ
Method name	Scope	Description	
Cancel_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม cancle เพื่อกลับสู่หน้าหลัก	
OK_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม ok เพื่อลงทะเบียนลูกข่าย	
subscribe()	Public	ลงทะเบียนเหตุการณ์ของหน้าลงทะเบียน	

ตารางที่ 4.10 แสดงรายละเอียดคลาส CreateRoomPageHandler

Class name	Description		
CreateRoomPageHandler	มีหน้าที่รับเหตุการณ์ของผู้เล่น และตอบสนองต่อผู้เล่นในหน้าการสร้างห้อง		
Attribute name	Scope	Type	Description
mNetwork	private	Network::ClientNetwork	ระบบเครือข่าย
mCInfo	private	Network::ClientInfo	ข้อมูลลูกข่าย
mRInfo	private	Network::RoomInfo	ข้อมูลห้อง
mRoomTxt	private	CEGUI::EditBox	กล่องข้อความรับชื่อห้อง
Method name	Scope	Description	
Cancel_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม cancle เพื่อกลับสู่หน้าเลือกห้อง	
Create_OnClick (CEGUI::EventArgs)	Public	จะถูกเรียกเมื่อกดปุ่ม create เพื่อสร้างห้อง	
subscribe()	Public	ลงทะเบียนเหตุการณ์ของหน้าสร้างห้อง	

4.1.3 ส่วนเกม



รูปที่ 4.3 Class diagram of Game Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.11 แสดงรายละเอียดคลาส GameCore

Class name	Description		
GameCore	ทำหน้าที่เป็นส่วนหลักในการประสานงานระหว่างส่วนต่างๆ ในเกม		
Attribute name	Scope	Type	Description
mNetwork	private	Network::ClientNetwork	ระบบเครือข่าย
mCInfo	private	Network::ClientInfo	ข้อมูลลูกข่าย
mRInfo	private	Network::RoomInfo	ข้อมูลห้อง
mCamera	private	Ogre::Camera	กล้องหลักของเกม
mFrameListener	private	CoreFrameListener	เป็นส่วนควบคุมและเปลี่ยนแปลงสถานะต่างๆ ในเกม
mGUISystem	private	CEGUI::System	ระบบ GUI
mHandler	private	CoreInputHandler	ระบบรับอินพุตหลัก
mPage	private	Page	GUI Page

ตารางที่ 4.11 (ต่อ)

mRoot	private	Ogre::Root	เป็นต้นกำเนิดของ Ogre
mSceneMgr	private	Ogre::SceneManager	ใช้จัดการทรัพยากรใน Scene
mSim	private	Simulation	จัดการสถานะ
mWindow	private	Ogre::RenderWindow	จอการแสดงผล
mWorld	private	Game::World	โลกของเกม
mGUIRenderer	private	CEGUI:: OgreCEGUIRenderer	ช่วยให้ CEGUI สามารถใช้ผ่าน Ogre
Method name	Scope	Description	
chooseSceneManager()	public	เลือกประเภทของ Scene	
configure()	public	ปรับแต่งคุณลักษณะของการแสดงผล	
createCamera()	public	สร้างกล้องหลัก	
createFrameListener()	public	สร้าง FrameListener ที่ใช้ในเกม	
createViewPort()	public	สร้างมุมมองจากกล้อง	
destroyScene()	public	จะถูกเรียกเมื่อต้องการทำลาย Scene	
getPage()	public	ดึง Page	
go()	public	ถูกเรียกเพื่อให้เกมเริ่มทำงาน	
initializeGUI()	public	กำหนดค่าเริ่มต้นให้กับระบบ GUI	
initializeNetwork()	public	กำหนดค่าเริ่มต้นให้กับระบบเครือข่าย	
initializePage()	public	กำหนดค่าเริ่มต้นให้กับอินเตอร์เฟสแรก	
initializeWorld()	public	กำหนดค่าเริ่มต้นให้กับโลกของเกม	
loadResource()	public	ใช้สำหรับดึงค่าทรัพยากรที่กำหนดให้สามารถนำมาใช้ได้	
setup()	public	ใช้สำหรับกำหนดค่าเริ่มต้นให้กับทุกระบบ	
setupInputHandler()	public	ติดตั้งการดักจับเหตุการณ์จากอุปกรณ์	
setupResource()	public	ใช้เพื่อดึงข้อมูลของทรัพยากร	

ตารางที่ 4.12 แสดงรายละเอียดคลาส CoreFrameListener

Class name	Description		
CoreFrameListener :FrameListener :WindowEventListener	ทำหน้าที่ตรวจสอบสถานะของอินเตอร์เฟสที่เป็นอยู่และทำการเปลี่ยนสถานะตามที่ร้องขอ		
Attribute name	Scope	Type	Description
mHandler	private	CoreInputHandler	ไว้สำหรับดักจับเหตุการณ์จากอุปกรณ์
mPage	private	Page	ใช้จัดการ GUI
mSim	private	Simulation	ใช้จัดการสถานะ
mSystem	private	CEGUI::System	ระบบ GUI
mWindow	private	Ogre::RenderWindow	จอแสดงผล
mWorld	private	Game::World	โลกของเกม
Oldstate	private	State	ใช้เก็บ state เก่า
Method name	Scope	Description	
frameStarted (Ogre::FrameEvent)	public	ถูกเรียกเมื่อเริ่มเฟรม	
frameEnded (Ogre::FrameEvent)	public	ถูกเรียกเมื่อจบเฟรม	

ตารางที่ 4.13 แสดงรายละเอียดคลาส CoreGameListener

Class name	Description		
CoreGameListener	ทำหน้าที่คอยตรวจสอบสถานะของตัวละครต่างๆ ที่ถูกส่งมาจากเครื่องแม่ข่าย และทำการเปลี่ยนแปลงค่าให้ตรงกับค่าที่เครื่องแม่ข่ายส่งมา		
Attribute name	Scope	Type	Description
Debug	private	Game::CoreDebugOverlay	แสดงสถานะของการแสดงผล
mRInfo	private	Network::RoomInfo	ข้อมูลห้อง
mSceneMgr	Private	Ogre::SceneManager	ใช้ในการจัดการ Scene
mSim	Private	Simulation	จัดการสถานะ

ตารางที่ 4.13 (ต่อ)

mWindow	Private	Ogre::RenderWindow	จอแสดงผล
mWorld	Private	Game::World	ไว้รับค่าจากเกมที่มีการเปลี่ยนแปลง
Method name	Scope	Description	
frameStarted (Ogre::FrameEvent)	Public	ถูกเรียกเมื่อเริ่มเฟรม	
frameEnded (Ogre::FrameEvent)	Public	ถูกเรียกเมื่อจบเฟรม	

ตารางที่ 4.14 แสดงรายละเอียดคลาส CoreDebugOverlay

Class name	Description		
CoreDebugOverlay	ทำหน้าที่แสดงสถานะของโปรแกรมเช่น FPS หรือแม้กระทั่งความเร็วของตัวละคร		
Attribute name	Scope	Type	Description
mDebugOverlay	Private	Ogre::Overlay	ใช้สำหรับแสดงผลลัพธ์
mWindow	Private	Ogre::RenderWindow	ใช้สำหรับแสดงผลค่าต่างๆ
Method name	Scope	Description	
ShowDebugOverlay(bool)	public	ใช้ในการกำหนดว่าจะแสดงค่าสถานะรึเปล่า	

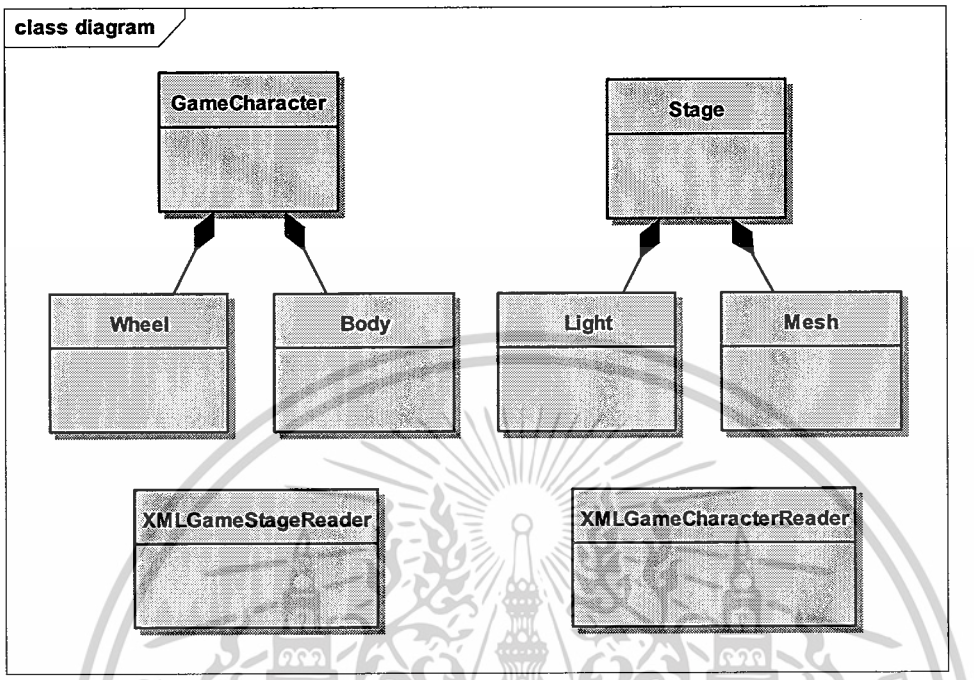
ตารางที่ 4.15 แสดงรายละเอียดคลาส World

Class name	Description		
World	ทำหน้าที่ในการสร้างโลกของเกมทั้งหมดคือ ฉาก ท้องฟ้า แสง กล้อง และตัวละคร		
Attribute name	Scope	Type	Description
mCamera	private	Ogre::Camera	กล้องหลักของระบบ
mCharacters	private	std::map <int, Model::GameCharacter>	สำหรับเก็บค่าสถานะของตัวละครทุกตัวบนสนาม
mNetwork	private	Network::ClientNetwork	ระบบเครือข่าย

ตารางที่ 4.15 (ต่อ)

mCInfo	private	Network::ClientInfo	ข้อมูลลูกข่าย
mRInfo	private	Network::RoomInfo	ข้อมูลห้อง
mPlayerCreate	private	Bool	ใช้ตรวจสอบว่าตัวละครถูกสร้างขึ้นแล้วรึยัง
mSceneMgr	private	Ogre::SceneManager	ใช้จัดการ Scene
mSim	private	Simulation	จัดการสถานะ
mStage	private	Model::Stage	ข้อมูลของสนาม
mWindow	private	Ogre::RenderWindow	จอแสดงผล
Method name	Scope	Description	
createCamera()	public	ใช้สร้างกล้องของเกม	
createCharacter (std::vector <Model::GameCharacter>)	public	ใช้สร้างตัวละครในแผนที่	
createLight()	public	ใช้สร้างแสง	
createScene()	public	ใช้สร้างฉาก	
getPosition(int)	public	ดึงค่าตัวละครจาก id	
requestStageChange(State)	public	ร้องขอเปลี่ยนสถานะ	
setPlayerPosition	public	ใช้ในการตั้งค่าตัวละคร	

4.1.4 ส่วนข้อมูลภายในเกม



รูปที่ 4.4 Class diagram of Game World Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.16 แสดงรายละเอียดคลาส Wheel

Class name	Wheel
Description	เป็นโครงสร้างทำหน้าที่เก็บข้อมูลของล้อ

ตารางที่ 4.17 แสดงรายละเอียดคลาส Body

Class name	Body
Description	เป็นโครงสร้างทำหน้าที่เก็บข้อมูลของตัวรถ

ตารางที่ 4.18 แสดงรายละเอียดคลาส GameCharacter

Class name	GameCharacter
Description	เป็นโครงสร้างทำหน้าที่เก็บข้อมูลของตัวละคร

ตารางที่ 4.19 แสดงรายละเอียดคลาส XMLGameCharacterReader

Class name	XMLGameCharacterReader
Description	ทำหน้าที่ในการอ่านไฟล์ XML ที่เป็นข้อมูลของตัวละคร

ตารางที่ 4.20 แสดงรายละเอียดคลาส Stage

Class name	Stage
Description	ทำหน้าที่เป็นตัวแทนของสนามต่างๆ ในเกม

ตารางที่ 4.21 แสดงรายละเอียดคลาส PlayerPosition

Class name	PlayerPosition
Description	เป็นโครงสร้างที่ทำหน้าที่เก็บข้อมูลของตำแหน่งกับค่าองศาการหมุนของตัวละคร

ตารางที่ 4.22 แสดงรายละเอียดคลาส Mesh

Class name	Mesh
Description	ทำหน้าที่ในการเก็บข้อมูลของโมเดล

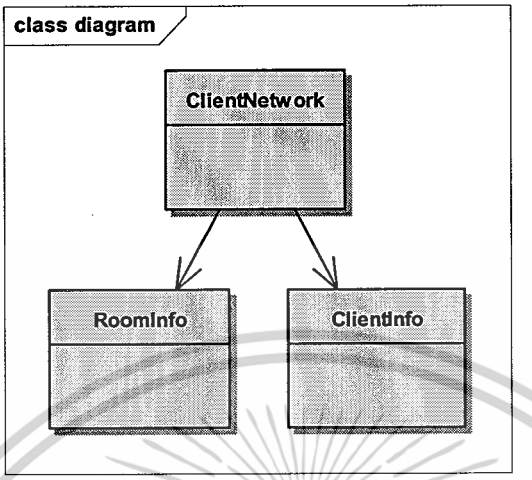
ตารางที่ 4.23 แสดงรายละเอียดคลาส Light

Class name	Light
Description	ทำหน้าที่ในการสร้างไฟให้กับเกม

ตารางที่ 4.24 แสดงรายละเอียดคลาส XMLGameStageReader

Class name	XMLGameStageReader
Description	ทำหน้าที่อ่านไฟล์ XML เพื่อใช้ในการสร้างสนาม

4.1.5 ส่วนเครือข่ายของลูกข่าย



รูปที่ 4.5 Class diagram of ClientNetwork Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.25 แสดงรายละเอียดคลาส ClientNetwork

Class name	Description		
ClientNetwork	ทำหน้าที่ในการติดต่อส่งข้อมูลและรับข้อมูลจากเครื่องแม่ข่าย		
Attribute name	Scope	Type	Description
mCInfo	private	ClientInfo	ระบบ Server
mPage	private	RoomManager	ระบบจัดการข้อมูลสนามแข่งขัน
mPeer	private	RakPeerInterface	ระบบเครือข่าย
mRInfo	private	bool	สถานะการทำงานของระบบ network มีสถานะทำงาน และหยุดทำงาน
mServer	private	SystemAddress	ที่อยู่ SystemAddress ของเครื่องแม่ข่าย
mSim	private	Simulation	จัดการสถานะ
mWorld	private	Game::World	ข้อมูลเกมทั้งหมด
mThread	private	boost::thread	เก็บตัวแปรที่แตกการทำงานออกเป็นเธรด

ตารางที่ 4.25 (ต่อ)

mRetry	private	int	จำนวนครั้งการส่งข้อมูลที่ผิดพลาด
mRunning	private	bool	สถานะการทำงานของระบบเครือข่าย

Method name	Scope	Description
GetClientManager ()	public	เรียกใช้งานระบบจัดการข้อมูลผู้เล่น
GetMainFrame ()	public	เรียกใช้งานระบบติดต่อผู้ใช้
GetRoomManager ()	public	เรียกใช้งานระบบจัดการข้อมูลสนามแข่งขัน
GetServerAddress ()	public	ดึงข้อมูลที่อยู่ของเครื่องแม่ข่าย
Init (ClientManager, RoomManager)	public	กำหนดค่าสถานะเริ่มต้น และเชื่อมการทำงานระหว่างระบบ RoomManager, ClientManager และ Server
Notify (SystemAddress, unsigned char)	public	ส่งข้อมูล message id ให้เครื่องปลายทางที่มีที่อยู่ตาม System address
Process	private	การทำงานหลักคลาสนี้ คือเป็นการรับข้อมูลจากเครื่องลูกข่าย เช่น ข้อมูลการจัดการข้อมูลผู้ใช้ การเริ่มเล่นเกม การบังคับตัวผู้
ReadString (BitStream, string)	private	เป็นการอ่านข้อมูลสายอักษรที่มีความซับซ้อนจากสายของข้อมูล
WriteString (BitStream, string)	private	เป็นการเขียนข้อมูลสายอักษรที่มีความซับซ้อนลงในสายของข้อมูล
SendReliablePacket (BitStrem, SystemAddress)	private	เป็นการส่งข้อมูลที่ต้องการความน่าเชื่อถือ มีลำดับ และต้องรับข้อมูลให้ครบ
SendUnreliablePacket (BitStream, SystemAddress)	public	เป็นการส่งข้อมูลที่ไม่ต้องการความน่าเชื่อถือ และมีลำดับโดยไม่ต้องรอรับข้อมูลให้ครบ
SetMainFrame (MainFrame)	public	ส่งค่า MainFrame ให้แก่ คลาสนี้เพื่อเรียกใช้ภายหลัง
Shutdown ()	Public	ปิดและทำลายระบบเครือข่าย

ตารางที่ 4.25 (ต่อ)

Start ()	public	เริ่มทำงานรับข้อมูล
Startup ()	public	ตั้งค่าเริ่มต้นให้ระบบเครือข่าย
Stop ()	public	หยุดการรับข้อมูลชั่วคราว

ตารางที่ 4.26 แสดงรายละเอียดคลาส ClientInfo

Class name	Description		
ClientInfo	มีหน้าที่เก็บข้อมูลต่างๆของผู้เล่นหนึ่งคน		
Attribute name	Scope	Type	Description
clientAddress	public	SystemAddress	ที่อยู่ SystemAddress ของผู้เล่น
clientId	public	unsigned char	ID ของผู้เล่น
clientName	public	std::string	ชื่อของผู้เล่น
connectTime	public	unsigned int	เวลาเมื่อผู้เล่นเข้าเชื่อมต่อครั้งแรก
curRoomId	public	unsigned char	ID ของสนามแข่งขันปัจจุบัน
Playing	public	bool	สถานะการแข่งขัน ซึ่งมีสองค่า คือ กำลังแข่งขัน และไม่ได้แข่งขัน
Method name	Scope	Description	
-			

ตารางที่ 4.27 แสดงรายละเอียดคลาส RoomInfo

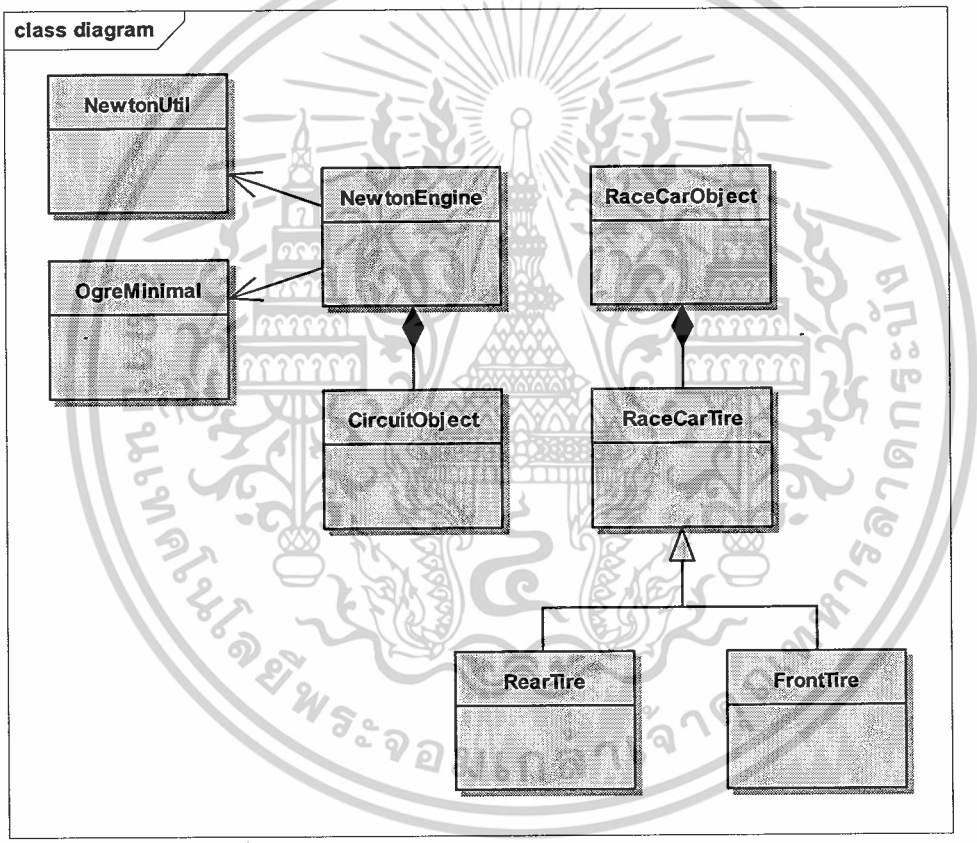
Class name	Description		
RoomInfo	มีหน้าที่เก็บข้อมูลต่างๆของสนามแข่งขันหนึ่งสนาม		
Attribute name	Scope	Type	Description
clientId	public	std::vector <unsigned char>	ID ของผู้เล่นทั้งหมดที่อยู่ในสนามแข่งขัน
mapId	public	unsigned char	ID ของรูปแบบสนามแข่งขันที่
ownerId	public	unsigned char	ID ของผู้เล่นของเจ้าของสนามแข่งขัน
Roomed	public	unsigned char	ID ของสนามแข่งขัน

ตารางที่ 4.27 (ต่อ)

roomName	Public	std::string	ชื่อของสนามแข่งขัน
Method name	Scope	Description	
GetClientCount	public	ดึงจำนวนผู้เล่นทั้งหมดในสนามแข่งขัน	

4.2 ส่วนแม่ข่าย (Server)

4.2.1 ส่วนการคำนวณเชิงฟิสิกส์



รูปที่ 4.6 Class diagram of ServerPhysic Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.28 แสดงรายละเอียดคลาส NewtonEngine

Class name	Description		
NewtonEngine	มีหน้าที่เป็นส่วนประมวลผลหลัก สร้าง ยกเลิกสภาพแวดล้อมทางฟิสิกส์ และ เปลี่ยนแปลงข้อมูลของสภาพแวดล้อม		
Attribute name	Scope	Type	Description
mLoopCount	Private	unsigned int	จำนวนรอบการทำงานของเครื่องแม่ข่ายในหนึ่งวินาที
mPlayer	Private	map <int, RaceCarObject>	ผู้เล่นทั้งหมดที่อยู่ในสนามแข่งขัน
mRoomId	Private	unsigned char	ID ของสนามแข่งขัน
mSecond	private	unsigned long	เวลาจริงในหน่วยมิลลิวินาที
mServer	private	Server	ระบบ Server
mWorld	private	NewtonWorld	แกนหลักของระบบฟิสิกส์
mThread	private	boost::thread	เก็บตัวแปรที่แตกการทำงานออกเป็นเธรด
mRunning	private	bool	สถานะการทำงานเครื่องแม่ข่าย
mTimer	private	CHiResTimer	ตัวนับเวลาแบบ Hi Resolution
Method name	Scope	Description	
FindFloor (dFloat, dFloat, NewtonWorld)	public	หาจุดที่เส้นตั้งฉากกับสนามแข่งระหว่างความสูง y1, y2	

ตารางที่ 4.29 แสดงรายละเอียดคลาส NewtonUtil

Class name	Description		
NewtonUtil	มีหน้าที่ให้บริการเพิ่มเติมให้กับการประมวลผลทางฟิสิกส์		
Attribute name	Scope	Type	Description
-			
Method name	Scope	Description	
FindFloor (dFloat, dFloat, NewtonWorld)	public	หาจุดที่เส้นตั้งฉากกับสนามแข่งระหว่างความสูง y1, y2	
RayCastPlacement (NewtonBody, dFloat, int, void, dFloat)	public	หาจุดที่เส้นตั้งฉากกับสนามแข่งระหว่างความสูง y1, y2 กับ NewtonBody	

ตารางที่ 4.30 แสดงรายละเอียดคลาส OgreMinimal

Class name	Description		
OgreMinimal	มีหน้าที่ให้บริการดึงข้อมูลจุดที่จะนำมาสร้างเป็นรูปทรงจากไฟล์เพื่อที่จะนำมาใช้ในเอ็นจิ้นฟิสิกส์		
Attribute name	Scope	Type	Description
-			
Method name	Scope	Description	
ExportMeshInformation (std::string, dVector, sizt_t, unsigned long, size_t)	public	เขียนข้อมูลเกี่ยวกับจุดของโมเดลลงในไฟล์	
ImportMeshInformation (std::string, dVector, sizt_t, unsigned long, size_t)	public	อ่านข้อมูลเกี่ยวกับจุดของโมเดลลงในไฟล์	

ตารางที่ 4.31 แสดงรายละเอียดคลาส CircuitObject

Class name	Description		
CircuitObject	มีหน้าที่สร้าง และเก็บข้อมูลจุดที่จะนำมาสร้างเป็นรูปทรงสำหรับจำลองลักษณะพื้นผิวสนามแข่งขัน		
Attribute name	Scope	Type	Description
mMatrix	private	dMatrix	เมทริกซ์เก็บตำแหน่งและแกนของรถ
mLevel	public	NewtonBody	จัดการเกี่ยวกับตัวสนามแข่งขัน เช่น ความเฉื่อย ความเร็ว แรงเสียดทาน
Method name	Scope	Description	
Destructor (NewtonBody)	public	ทำลายระบบฟิสิกส์ของสนามแข่ง	
GetRigidBody ()	public	ดึงฟิสิกส์ของสนามแข่ง	

ตารางที่ 4.32 แสดงรายละเอียดคลาส RaceCarTire

Class name	Description		
RaceCarTire	มีหน้าที่สร้าง และเก็บข้อมูลจุดที่จะนำมาสร้างเป็นรูปทรงสำหรับจำลองลักษณะล้อของรถ รวมถึงการเคลื่อนที่ของล้อในสนามแข่งขัน		
Attribute name	Scope	Type	Description
mMatrix	public	dMatrix	เมทริกซ์เก็บตำแหน่งและแกนของรถ
mTireId	public	void	ID ของล้อ
mWidth	public	dFloat	ความกว้างของล้อ
mRadius	public	dFloat	รัศมีของล้อ
Method name	Scope	Description	
Setup (RaceCarObject, char, dVector)	public	กำหนดค่าเริ่มต้นฟิสิกส์	
SetTirePhysics (NewtonJoint, void)	public	คำนวณฟิสิกส์ของล้อ	

ตารางที่ 4.33 แสดงรายละเอียดคลาส RearTire

Class name	Description		
RearTire	มีหน้าที่ประมวลผลข้อมูลของล้อหลังซึ่งสามารถที่จะทำการเพิ่มทอร์ค หรือสามารถกำหนดให้เบรกได้		
Attribute name	Scope	Type	Description
mTorque	public	dFloat	ค่าแสดงแรงบิดของล้อ
mBrakes	public	dFloat	ค่าแสดงการเบรกของล้อ
Method name	Scope	Description	
SetBrakes (dFloat)	public	กำหนดแรงบิดของล้อ	
SetTorque (dFloat)	public	กำหนดการเบรกของล้อ	
SetTirePhysics (NewtonJoint, void)	public	คำนวณฟิสิกส์ของล้อ	

ตารางที่ 4.34 แสดงรายละเอียดคลาส FrontTire

Class name	Description		
FrontTire	มีหน้าที่ประมวลผลข้อมูลของล้อหน้าซึ่งสามารถที่จะทำการหมุนล้อเพื่อให้สามารถเลี้ยวรถได้ตามต้องการ		
Attribute name	Scope	Type	Description
mSteer	public	dFloat	ทิศทางการหมุนของพวงมาลัย
Method name	Scope	Description	
SetSteer (dFloat)	public	กำหนดทิศทางการหมุนของพวงมาลัย	
SetTirePhysics (NewtonJoint, void)	public	คำนวณฟิสิกส์ของล้อ	

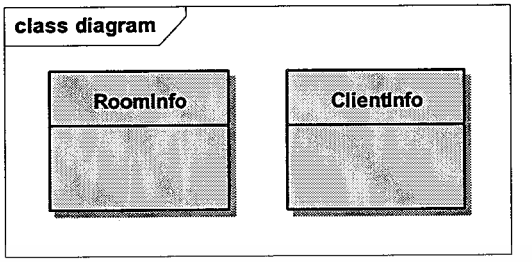
ตารางที่ 4.35 แสดงรายละเอียดคลาส RaceCarObject

Class name	Description		
RaceCarObject	มีหน้าที่สร้าง และเก็บข้อมูลจุดที่จะนำมาสร้างเป็นรูปทรงสำหรับจำลองลักษณะโครงหลักของรถ และยังเป็นส่วนหลักในการเปลี่ยนแปลงข้อมูลฟิสิกส์ต่างๆของรถในระบบฟิสิกส์นี้		
Attribute name	Scope	Type	Description
mFrontLeftTire	private	FrontTire	จัดการเกี่ยวกับล้อรถตำแหน่งหน้าซ้าย

ตารางที่ 4.35 (ต่อ)

mFrontRightTire	private	FrontTire	จัดการเกี่ยวกับล้อรถตำแหน่งหน้าขวา
mRearLeftTire	private	RearTire	จัดการเกี่ยวกับล้อรถตำแหน่งหลังซ้าย
mRearRightTire	private	RearTire	จัดการเกี่ยวกับล้อรถตำแหน่งหลังขวา
mVehicleBody	protected	NewtonBody	จัดการเกี่ยวกับตัวรถ เช่น ความเฉื่อย ความเร็ว แรงเสียดทาน
mVehicleJoint	protected	NewtonJoint	จัดการจุดเชื่อมระหว่างล้อทุกตัว และโครงรถ
mMatrix	protected	dMatrix	เมทริกซ์เก็บตำแหน่งและแกนของรถ
Method name	Scope	Description	
ApplyGravityForce (NewtonBody)	private	คำนวณแรงดึงดูดของโลกที่เกิดกับตัวรถ	
DestroyVehicle (NewtonBody)	private	ทำลายระบบฟิสิกส์ของรถ (ใช้ภายใน)	
DestroyVehicle (NewtonWorld)	public	ทำลายระบบฟิสิกส์ของรถ (ใช้ภายนอก)	
GetJoint ()	public	ดึงจุดเชื่อมต่อระหว่างล้อรถ และตัวรถ	
GetMatrix ()	dMatrix	ดึงตำแหน่งและระนาบแกน XYZ ของตัวถังรถ	
GetRigidBody ()	NewtonBody	ดึงฟิสิกส์ของตัวถังรถ	
GetSpeed ()	dFloat	ดึงความเร็วของตัวรถ	
GetTireMatrixFL ()	dMatrix	ดึงตำแหน่งและระนาบแกน XYZ ของล้อตำแหน่งหน้าซ้าย	
GetTireMatrixFR ()	dMatrix	ดึงตำแหน่งและระนาบแกน XYZ ของล้อตำแหน่งหน้าขวา	
GetTireMatrixRL ()	dMatrix	ดึงตำแหน่งและระนาบแกน XYZ ของล้อตำแหน่งหลังซ้าย	
GetTireMatrixRR ()	dMatrix	ดึงตำแหน่งและระนาบแกน XYZ ของล้อตำแหน่งหลังขวา	

4.2.3 ส่วนการจัดการข้อมูลเครือข่าย



รูปที่ 4.7 Class diagram of Network Package

รายละเอียดคลาส (Class specification)

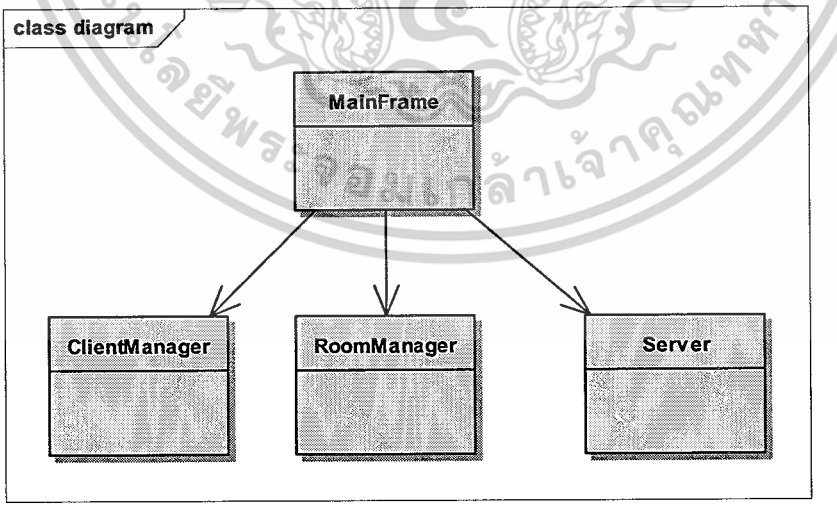
ตารางที่ 4.36 แสดงรายละเอียดคลาส ClientInfo

Class name	Description		
ClientInfo	มีหน้าที่เก็บข้อมูลต่างๆของผู้เล่นหนึ่งคน		
Attribute name	Scope	Type	Description
clientAddress	Public	SystemAddress	ที่อยู่ SystemAddress ของผู้เล่น
clientId	Public	unsigned char	ID ของผู้เล่น
clientName	public	std::string	ชื่อของผู้เล่น
connectTime	public	unsigned int	เวลาเมื่อผู้เล่นเข้าเชื่อมต่อครั้งแรก
curRoomId	public	unsigned char	ID ของสนามแข่งขันปัจจุบัน
Playing	public	bool	สถานะการแข่งขัน ซึ่งมีสองค่าคือ กำลังแข่งขัน และไม่ได้แข่งขัน
Method name	Scope	Description	
-			

ตารางที่ 4.37 แสดงรายละเอียดคลาส RoomInfo

Class name	Description		
RoomInfo	มีหน้าที่เก็บข้อมูลต่างๆของสนามแข่งขันหนึ่งสนาม ซึ่งระบบฟิสิกส์จะสามารถเข้าถึงได้ผ่านคลาสนี้เช่นกัน		
Attribute name	Scope	Type	Description
clientId	public	std::vector <unsigned char>	ID ของผู้เล่นทั้งหมดที่อยู่ในสนามแข่งขัน
mapId	public	unsigned char	ID ของรูปแบบสนามแข่งขันที่
Newton	public	NewtonEngine	ส่วนการทำงานด้านฟิสิกส์
ownerId	public	unsigned char	ID ของผู้เล่นของเจ้าของสนามแข่งขัน
roomed	public	unsigned char	ID ของสนามแข่งขัน
roomName	Public	std::string	ชื่อของสนามแข่งขัน
Method name	Scope	Description	
GetClientCount	public	ดึงจำนวนผู้เล่นทั้งหมดในสนามแข่งขัน	

4.2.4 ส่วนการประมวลผลเครื่องแม่ข่าย



รูปที่ 4.8 Class diagram of Server Package

รายละเอียดคลาส (Class specification)

ตารางที่ 4.38 แสดงรายละเอียดคลาส MainFrame

Class name	Description		
MainFrame	มีหน้าที่แสดงผลสถานะต่างๆของเครื่องแม่ข่าย สนามแข่งขันทั้งหมด และผู้เล่นทั้งหมดที่เชื่อมต่ออยู่		
Attribute name	Scope	Type	Description
mServer	Private	Server	ระบบ Server
mRoom	private	RoomManager	ระบบจัดการข้อมูลสนามแข่งขัน
mClient	private	ClientManager	ระบบจัดการข้อมูลผู้เล่น
clearLogBtn	private	wxButton	ปุ่มสำหรับลบสถานะการทำงาน
clientList	private	wxListCtrl	รายชื่อผู้เล่นที่เชื่อมต่ออยู่
clientListLbl	private	wxStaticText	ชื่อกำกับ รายชื่อผู้เล่นที่เชื่อมต่ออยู่
logLbl	private	wxStaticText	ชื่อกำกับ ช่องแสดงสถานะการทำงาน
logTxt	private	wxTextCtrl	ช่องแสดงสถานะการทำงาน
maxClientLbl	private	wxStaticText	ชื่อกำกับ ช่องแสดงสถานะการทำงาน
maxClientTxt	private	wxTextCtrl	ช่อง แสดงจำนวนผู้เล่นที่รองรับได้
refreshClientBtn	private	wxButton	ปุ่มสำหรับดึงข้อมูลผู้เล่นล่าสุด
refreshRoomBtn	private	wxButton	ปุ่มสำหรับดึงข้อมูลสนามแข่งล่าสุด
roomList	private	wxListCtrl	รายชื่อสนามที่ถูกสร้างขึ้น
roomListLbl	private	wxStaticText	ชื่อกำกับ รายชื่อสนามที่ถูกสร้างขึ้น
serverIpLbl	private	wxStaticText	ชื่อกำกับ หมายเลขไอพีเครื่องแม่ข่าย
serverIpTxt	private	wxTextCtrl	หมายเลขไอพีเครื่องแม่ข่าย
serverPortLbl	private	wxTextCtrl	ชื่อกำกับ หมายเลขพอร์ทเครื่องแม่ข่าย
serverPortTxt	private	wxTextCtrl	หมายเลขพอร์ทเครื่องแม่ข่าย
shutdownBtn	private	wxButton	ปุ่มปิด และทำลายระบบเครือข่าย
startBtn	private	wxButton	ปุ่มเริ่มการทำงานของระบบเครือข่าย
startupBtn	private	wxButton	ปุ่มตั้งค่าเริ่มต้น และเปิดการทำงานของระบบเครือข่าย
stopBtn	private	wxButton	ปุ่มหยุดการทำงานของระบบเครือข่าย

ตารางที่ 4.38 (ต่อ)

mThread	private	boost::thread	เก็บตัวแปรที่แตกการทำงานออกเป็นเธรด
Method name	**Scope**	**Description**	
AppendLog (std::string)	public	แสดงสถานะของระบบผ่านส่วนติดต่อผู้ใช้	
AppendLog (wxString)	public	แสดงสถานะของระบบผ่านส่วนติดต่อผู้ใช้	
ClearLog (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม clearLogBtn	
RefreshClient (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม refreshClientBtn	
RefreshRoom (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม refreshRoomBtn	
Shutdown (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม shutdownBtn	
Start (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม startBtn	
StartUp (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม startupBtn	
Stop (wxCommandEvent)	public	สำหรับเรียกเมื่อเกิดเหตุการณ์เมื่อผู้ใช้กดปุ่ม stopBtn	

ตารางที่ 4.39 แสดงรายละเอียดคลาส Server

Class name	Description		
Server	มีหน้าที่รับผิดชอบการรับส่งข้อมูลของเครื่องแม่ข่ายและเครื่องลูกข่ายทุกเครื่อง		
Attribute name	**Scope**	**Type**	**Description**
mServer	private	Server	ระบบ Server
mRoom	private	RoomManager	ระบบจัดการข้อมูลสนามแข่งขัน
mClient	private	ClientManager	ระบบจัดการข้อมูลผู้เล่น
mLoopCount	private	Unsigned int	จำนวนรอบการทำงานของเครื่องแม่ข่ายในหนึ่งวินาที
mMainFrame	private	MainFrame	ระบบแสดงผลติดต่อผู้ใช้
mPeer	private	RakPeerInterface	ระบบ
mRunning	private	bool	สถานะการทำงานของระบบ network มีสถานะทำงาน และหยุดทำงาน
mSecond	private	unsigned long	เวลาจริงในหน่วยมิลลิวินาที
mThread	private	boost::thread	เก็บตัวแปรที่แตกการทำงานออกเป็นเธรด
Method name	**Scope**	**Description**	
GetClientManager ()	public	เรียกใช้งานระบบจัดการข้อมูลผู้เล่น	
GetMainFrame ()	public	เรียกใช้งานระบบติดต่อผู้ใช้	
GetRoomManager ()	public	เรียกใช้งานระบบจัดการข้อมูลสนามแข่งขัน	
GetServerAddress ()	public	ดึงข้อมูลที่อยู่ของเครื่องแม่ข่าย	
Init (ClientManager, RoomManager)	public	กำหนดค่าสถานะเริ่มต้น และเชื่อมการทำงานระหว่างระบบ RoomManager, ClientManager และ Server	
Notify (SystemAddress, unsigned char)	public	ส่งข้อมูล message id ให้เครื่องปลายทางที่มีที่อยู่ตาม System address	
Process	private	การทำงานหลักคลาสนี้ คือเป็นการรับข้อมูลจากเครื่องลูกข่าย เช่น ข้อมูลการจัดการข้อมูลผู้ใช้ การเริ่มเล่นเกม การบังคับตัวผู้	

ตารางที่ 4.39 (ต่อ)

ReadString (BitStream, string)	private	เป็นการอ่านข้อมูลสายอักษรที่มีความซับซ้อนจากสายของข้อมูล
WriteString (BitStream, string)	private	เป็นการเขียนข้อมูลสายอักษรที่มีความซับซ้อนลงในสายของข้อมูล
SendReliablePacket (BitStrem, SystemAddress)	private	เป็นการส่งข้อมูลที่ต้องการความน่าเชื่อถือ มีลำดับ และต้องรับข้อมูลให้ครบ
SendUnreliablePacket (BitStream, SystemAddress)	public	เป็นการส่งข้อมูลที่ไม่ต้องการความน่าเชื่อถือ และมีลำดับโดยไม่ต้องรอรับข้อมูลให้ครบ
SetMainFrame (MainFrame)	public	ส่งค่า MainFrame ให้แก่ คลาสนี้เพื่อเรียกใช้ภายหลัง
Shutdown ()	Public	ปิดและทำลายระบบเครือข่าย
Start ()	public	เริ่มทำงานรับข้อมูล
Startup ()	public	ตั้งค่าเริ่มต้นให้ระบบเครือข่าย
Stop ()	public	หยุดการรับข้อมูลชั่วคราว

ตารางที่ 4.40 แสดงรายละเอียดคลาส RoomManager

Class name	Description		
RoomManager	มีหน้าที่จัดการข้อมูลของสนามแข่ง เช่น สร้าง-ลบสนามแข่งขัน การเพิ่ม-ลบผู้เล่นในสนามแข่งขัน		
Attribute name	Scope	Type	Description
mServer	private	Server	ระบบ Server
mRInfo	private	std::map<unsigned char, RoomInfo>	ข้อมูลของสนามแข่งซึ่งเก็บอยู่ในรูป key และ value
mClient	private	ClientManager	ระบบจัดการข้อมูลผู้เล่น
Method name	Scope	Description	
Clear ()	public	ลบข้อมูลสนามแข่งขันทั้งหมดออกจากระบบ	
Create (RoomInfo)	public	สร้างสนามแข่งขัน และเพิ่มผู้เล่นที่เป็นเจ้าของห้องเข้าไป	

ตารางที่ 4.40 (ต่อ)

Destroy (unsigned char, unsigned char)	public	ทำลายสนามแข่งขัน และนำผู้เล่นออกจากสนามทั้งหมด
Join (unsigned char, unsigned char)	Public	เพิ่มผู้เล่นเข้าไปยังสนามแข่งขันจาม ID ของสนามแข่งขัน และ ID ของผู่เล่น
Disjoin (unsigned char, unsigned char)	public	นำผู้เล่นออกจากสนามแข่งขันตาม ID ของสนามแข่งขัน และ ID ของผู้เล่น
FindByOwnerId (unsigned char)	public	ดึงข้อมูลสนามแข่งขันจาก ID ของเจ้าของสนาม
GetCount ()	public	ดึงจำนวนสนามแข่งขันทั้งหมด
GetFirstRoom ()	public	ดึงข้อมูลสนามแข่งขันแรกสุด
GetLastRoom ()	public	ดึงข้อมูลสนามแข่งขันสุดท้าย
GetRoom (unsigned char)	public	ดึกข้อมูลสนามแข่งขันจาก ID ของสนามแข่งขัน
Init (ClientManager, Server)	public	กำหนดค่าสถานะเริ่มต้น และเชื่อมการทำงานระหว่างระบบ RoomManager, ClientManager และ Server

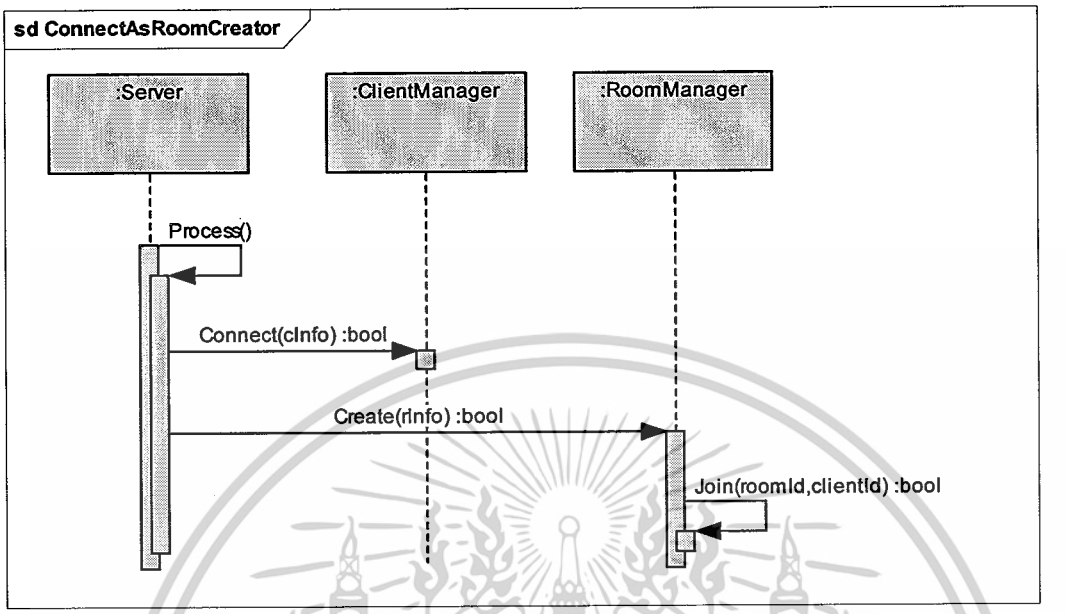
ตารางที่ 4.41 แสดงรายละเอียดคลาส ClientManager

Class name	Description		
ClientManager	มีหน้าที่จัดการข้อมูลของผู้เล่น เช่น การเริ่มการเชื่อมต่อ การตัดการเชื่อมต่อ และที่อยู่ของผู้เล่น		
Attribute name	Scope	Type	Description
mServer	private	Server	ระบบ Server
mRoom	private	RoomManager	ระบบจัดการข้อมูลสนามแข่งขัน
mCInfo	private	std::map<unsigned char, ClientInfo>	ข้อมูลผู้เล่นซึ่งเก็บอยู่ในรูป key และ value
Method name	Scope	Description	
Clear ()	public	ลบข้อมูลผู้เล่นทั้งหมดออกจากระบบ	
Connect (ClientInfo)	public	ผู้เล่นเชื่อมต่อกับระบบและเพิ่มข้อมูลเข้าระบบ	

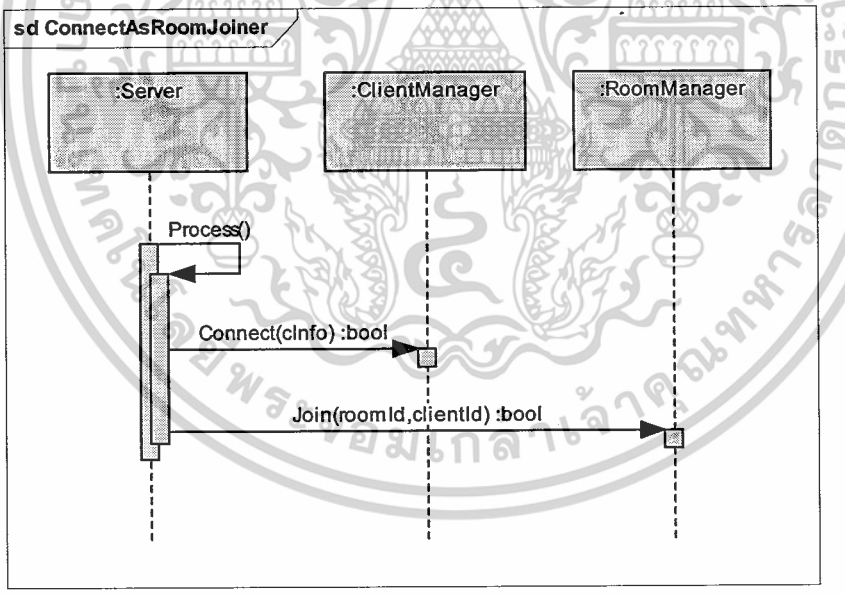
ตารางที่ 4.41 (ต่อ)

Disconnect (unsigned char)	public	ผู้เล่นตัดการเชื่อมต่อกับระบบและลบข้อมูลออกจากระบบ
FindBySystemAddress (SystemAddress)	public	ดึงข้อมูลผู้เล่นจากค่า System address
GetClient (unsigned char)	public	ดึงข้อมูลผู้เล่นจากค่า ID ของผู้เล่น
GetCount ()	public	ดึงจำนวนผู้เล่นทั้งหมด
GetFirstClient ()	public	ดึงข้อมูลผู้เล่นแรกสุด
GetLastClient ()	public	ดึงข้อมูลผู้เล่นสุดท้าย
Init (RoomManager, Server)	public	กำหนดค่าสถานะเริ่มต้น และเชื่อมการทำงานระหว่างระบบ RoomManager, ClientManager และ Server

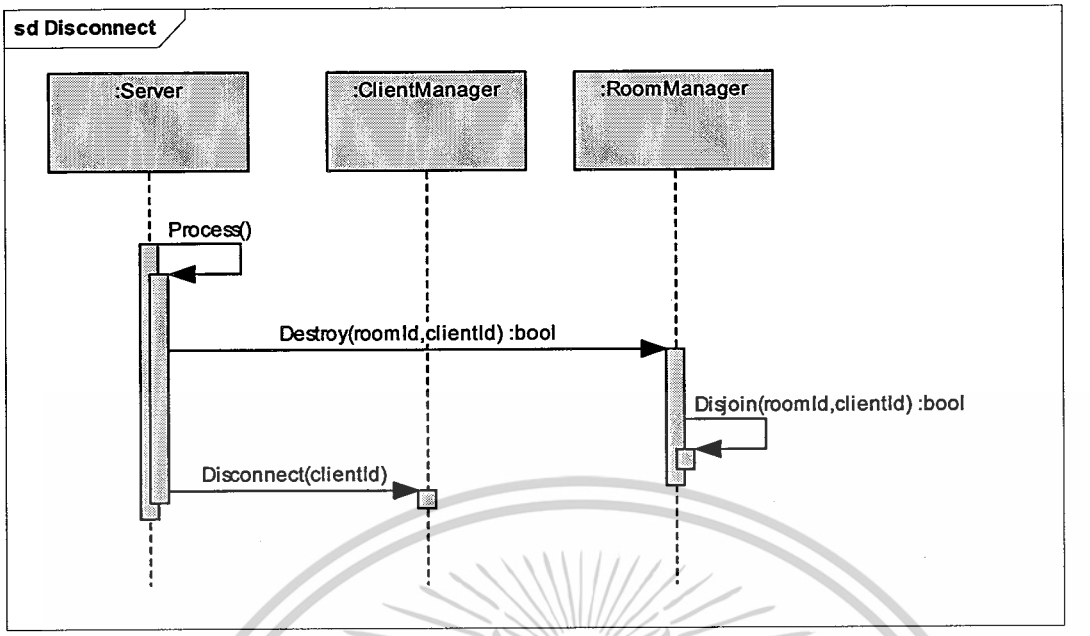
Sequence diagram



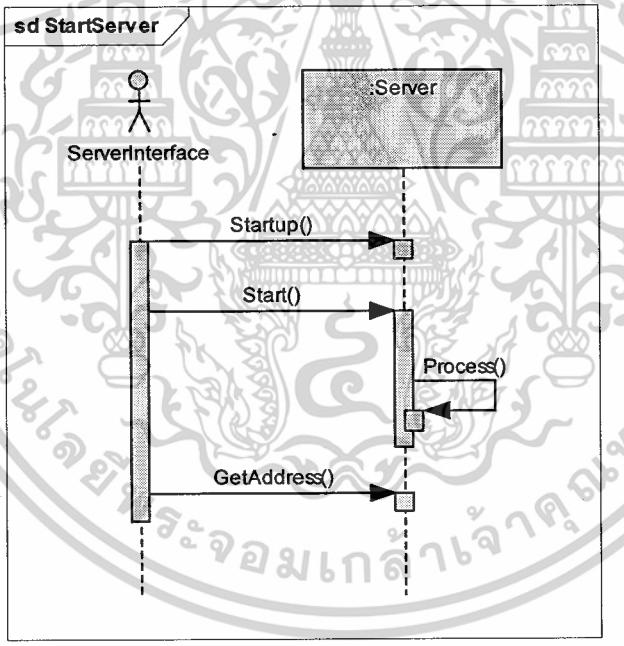
รูปที่ 4.9 sequence diagram of connecting server as room creator



รูปที่ 4.10 sequence diagram of connecting server as room joiner



รูปที่ 4.11 sequence diagram of disconnecting from server



รูปที่ 4.12 sequence diagram of starting server

4.3 รูปแบบข้อมูลที่ส่งผ่านเครือข่าย

กลุ่มของข้อมูลที่ต้องการความน่าเชื่อถือของเครือข่าย และเรียงลำดับ แต่ไม่ต้องการความเร็ว (RELIABLE, ORDER, LOW PRIORITY)

ตารางที่ 4.42 แสดงรายละเอียดกลุ่มของข้อมูลที่ต้องการความน่าเชื่อถือของเครือข่าย และเรียงลำดับ แต่ไม่ต้องการความเร็ว (RELIABLE, ORDER, LOW PRIORITY) ที่ส่งจากผู้เล่นไปยังฝั่งเครื่องให้บริการ

Message ID	Time Stamp			
ID_CLIENT_CONNECT	unsigned long	ClientInfo		
ID_CLIENT_DISCONNECT	unsigned long	clientId		
ID_ROOM_CREATE	unsigned long	RoomInfo		
ID_ROOM_LIST_REQ	unsigned long			
ID_ROOM_INFO_REQ	unsigned long	roomId		
ID_ROOM_JOIN	unsigned long	roomId	clientId	
ID_ROOM_DISJOIN	unsigned long	roomId	clientId	
ID_ROOM_DESTROY	unsigned long	roomId	clientId	
ID_GAME_START	unsigned long	roomId	mapId	
ID_GAME_END	unsigned long	ClinetCount	[ClientInfo, unsigned char rank]...	

ตารางที่ 4.43 แสดงรายละเอียดกลุ่มของข้อมูลที่ต้องการความน่าเชื่อถือของเครือข่าย และเรียงลำดับ แต่ไม่ต้องการความเร็ว (RELIABLE, ORDER, LOW PRIORITY) ที่ส่งจากเครื่องให้บริการไปยังฝั่งผู้เล่น

Message ID	Time Stamp			
ID_CLIENT_CONNECT_OK	unsigned long	clientId		
ID_CLIENT_DISCONNECT_OK	unsigned long			
ID_ROOM_CREATE_OK	unsigned long	roomId		
ID_ROOM_LIST_RES	unsigned long	unsigned char length	RoomInfo …	

ตารางที่ 4.43 (ต่อ)

ID_ROOM_INFO_RES	unsigned long	RoomInfo	unsigned char length	ClientInfo …
ID_ROOM_JOIN_OK	unsigned long			
ID_ROOM_DISJOIN_OK	unsigned long			
ID_ROOM_DESTROY_OK	unsigned long			
ID_GAME_START_OK	unsigned long	ClientCount	ClientInfo …	mapId

ตารางที่ 4.44 แสดงรายละเอียดกลุ่มของข้อมูลที่ต้องการความน่าเชื่อถือของเครือข่าย และเรียงลำดับ แต่ไม่ต้องการความเร็ว (RELIABLE, ORDER, LOW PRIORITY) ที่ส่งจากฝั่งใดฝั่งหนึ่งเพื่อรายงานข้อผิดพลาด

Message ID ที่ส่งจากทั้งสองฝั่ง	Time Stamp			
ID_ERROR	unsigned long	unsigned short length	std::string	
ID_INVALID_ID	unsigned long			

กลุ่มของข้อมูลที่ไม่ต้องการความน่าเชื่อถือของเครือข่าย ไม่ต้องเรียงลำดับ แต่ต้องการความเร็ว (UNRELIABLE, MEDIUM_PRIORITY)

ตารางที่ 4.45 กลุ่มของข้อมูลที่ไม่ต้องการความน่าเชื่อถือของเครือข่าย ไม่ต้องเรียงลำดับ แต่ต้องการความเร็ว (UNRELIABLE, MEDIUM_PRIORITY)

Message ID	Time Stamp			
ID_CLIENT_CONTROL	unsigned long	roomId	clientId	direction
ID_UPDATE_OBJECT	unsigned long	ClientCount	Character [clientId position Orientation]	

อธิบายโครงสร้างต่างๆที่จะส่งไปยังเครือข่าย

ตารางที่ 4.46 อธิบายโครงสร้างที่ส่งผ่านเครือข่ายในแต่ละกลุ่มข้อมูล

ชื่อโครงสร้าง	ชนิดข้อมูลภายใน	ชื่อข้อมูลภายใน	คำอธิบาย
ClientInfo	unsigned char	clientId	เมื่อยังไม่รู้จะเป็น 0
	string	clientName	ชื่อของผู้เชื่อมต่อ
RoomInfo	unsigned char	roomId	เมื่อยังไม่รู้จะเป็น 0
	string	roomName	ชื่อของห้องที่สร้าง
	unsigned char	ownerId	ระบุเจ้าของห้อง
string	unsigned char	length	ความยาวข้อมูล
	char*	data	ข้อมูล

บทที่ 5
การดำเนินงาน และผลลัพธ์ที่ได้

5.1 เครื่องมือที่ใช้ในการพัฒนาระบบ

- Microsoft Visual Studio 2005 Professional Edition สำหรับเขียนซอร์สโค้ด คอมไพล์ และสร้างไบนารี่ไฟล์
- Maya 8.5 สำหรับสร้างแบบจำลองโครงสร้างสามมิติ เช่น ตัวละคร ฉาก วัตถุต่างๆ
- Maya Exporters 1.2.5 for OGRE สำหรับเก็บข้อมูลโครงสร้างเรขาคณิตเป็นแฟ้มข้อมูลนามสกุล .mesh
- Subversion สำหรับจัดเก็บเอกสารและซอร์สโค้ดสำหรับผู้ใช้งานร่วมกันแบบกลุ่ม

5.2 ชุดพัฒนาซอฟต์แวร์ที่ใช้ในการพัฒนาระบบ

- OGRE 1.4.4 เป็นชุดพัฒนาซอฟต์แวร์สำหรับแสดงผลภาพสามมิติที่ถูกพัฒนาขึ้นจากภาษา C++ มีส่วนต่อขยายที่พร้อมใช้อยู่มากมาย และสามารถพัฒนาเพิ่มเติมได้ด้วยตัวเอง
  อ้างอิงจาก http://www.ogre3d.org/
- Newton Game Dynamics 1.53 for Window 32bit เป็นชุดพัฒนาซอฟต์แวร์สำหรับจำลองสภาพแวดล้อมทางกลศาสตร์ ซึ่งมีการจัดเตรียมการจัดการฉาก ตรวจสอบการปะทะ ซึ่งมีจุดเด่นคือ มีขนาดเล็ก เร็ว และง่ายต่อการใช้ ไม่จำกัดสำหรับการสร้างเกมเท่านั้น แต่ใช้กับซอฟต์แวร์เรียลไทม์อื่นๆได้อีกด้วย
  อ้างอิงจาก http://www.newtondynamics.com/
- OgreNewt 0.11 เป็นชุดพัฒนาซอฟต์แวร์สำหรับประสานการทำงานระหว่าง Newton Game Dynamics และ OGRE ซึ่งทำการห่อหุ้ม Newton Game Dynamics ให้อยู่ในกลุ่มของคลาสในระบบเชิงวัตถุโดยใช้ชนิดข้อมูลของOGRE
  อ้างอิงจาก http://www.walaber.com/
- Raknet 3.0 Beta เป็นชุดพัฒนาซอฟต์แวร์สำหรับการสร้างระบบเน็ตเวิร์คสำหรับใช้ในเกม ซึ่งมีการปรับปรุงเฮดเดอร์ของแพคเกตให้มีขนาดน้อยลงจะทำให้ประสิทธิภาพในการส่งข้อมูลมีมากขึ้น
- Boost C++ Library เป็นไลบรารีซึ่งนำมาใช้ในการสร้างเธรดเพื่อให้การทำงานของแม่ข่ายและลูกข่ายมีประสิทธิภาพมากขึ้น
- wxWidgets เป็นชุดพัฒนาซอฟต์แวร์สำหรับสร้างอินเทอร์เฟสซึ่งสามารถใช้ได้ในทุกแพลตฟอร์มพร้อมทั้งมีคอนโทรลให้เลือกอย่างหลากหลาย

5.3 การติดตั้งชุดพัฒนาซอฟต์แวร์

1 ติดตั้งซอฟต์แวร์

1.1 Microsoft Visual Studio 2005 Professional Edition

1.2 Microsoft Visual Studio 2005 Service Pack 1

1.3 OGRE 1.4.4

1.4 Newton Game Dynamics 1.5.3 for Window 32 bit

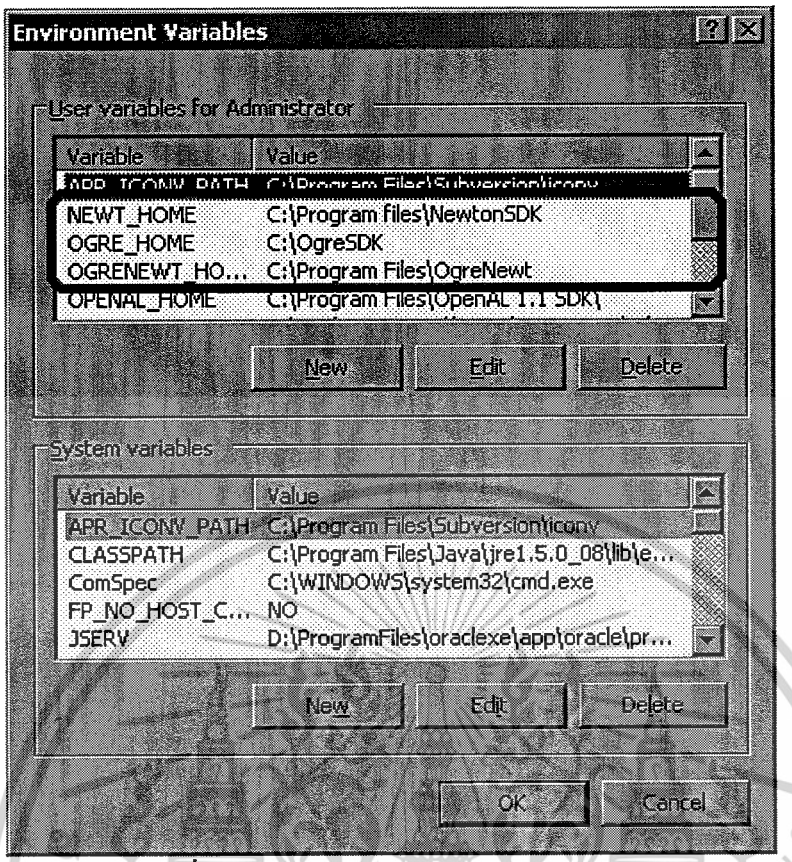
1.5 OgreNewt 0.11

2 เพิ่มค่า Environment Variables ใน System Properties โดยการเลือกแถบ Advance และกดปุ่ม Environment Variables จากนั้นกดปุ่ม New ในกลุ่มของ User variables for Administrator และใส่ค่าดังต่อไปนี้

2.1 ในช่อง Variable ใส่ OGRE_HOME ในช่อง Value ใส่ตำแหน่งของ OGRE ที่ได้ติดตั้งในเครื่อง

2.2 ในช่อง Variable ใส่ NEWT_HOME ในช่อง Value ใส่ตำแหน่งของ Newton ที่ได้ติดตั้งในเครื่อง

2.3 ในช่อง Variable ใส่ OGRENEWT_HOME ในช่อง Value ใส่ตำแหน่งของ OgreNewt ที่ได้ติดตั้งในเครื่อง



รูปที่ 5.1 ตัวอย่างค่า Environment Variables

3 เข้าไปยังโปรแกรม Microsoft Visual Studio 2005 และเปิด Project ที่ต้องการสร้างเพื่อตั้งค่าเริ่มต้นให้ Project

3.1 กดที่ Project Menu จากนั้นเลือก Properties

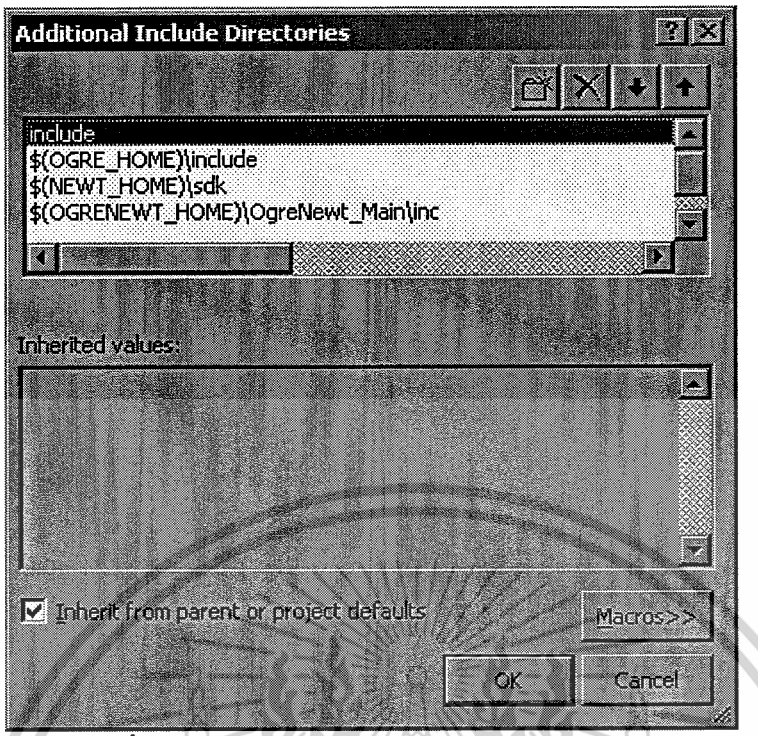
3.2 หน้าต่างซ้ายเลือกที่ Configuration Properties จากนั้นเลือกที่ C/C++ จากนั้นเลือกที่ General

3.3 หน้าต่างขวาเลือกที่ Additional Include Directories จากนั้นกดปุ่มเล็กๆ ด้านขวามือ จากนั้นเพิ่มบรรทัดต่างๆลงไปดังนี้

$(OGRE_HOME)\include

$(NEWT_HOME)\sdk

$(OGRENEWT_HOME)\OgreNewt_Main\inc



รูปที่ 5.2 ตัวอย่างค่า Additional Include Directories

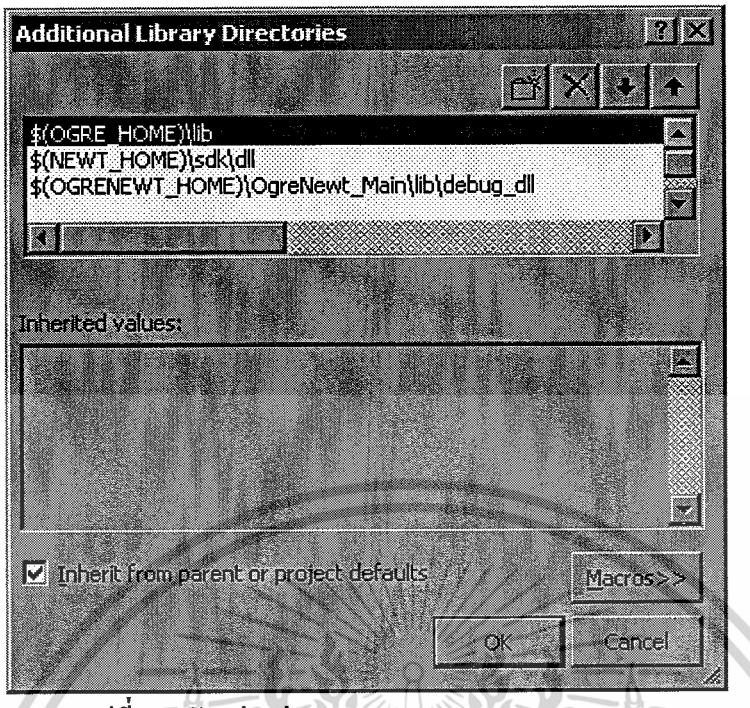
3.4 หน้าต่างซ้ายเลือกที่ Configuration Properties จากนั้นเลือกที่ Linker จากนั้นเลือกที่ General

3.5 หน้าต่างขวาเลือกที่ Additional Library Directories จากนั้นกดปุ่มเล็กๆ ด้านขวามือ จากนั้นเพิ่มบรรทัดต่างๆลงไปดังนี้

$(OGRE_HOME)\lib

$(NEWT_HOME)\sdk\dll

$(OGRENEWT_HOME)\OgreNewt_Main\lib\debug_dll



รูปที่ 5.3 ตัวอย่างค่า Additional Library Directories

3.6 หน้าต่างซ้ายเลือกที่ Configuration Properties จากนั้นเลือกที่ Linker จากนั้นเลือกที่ Input

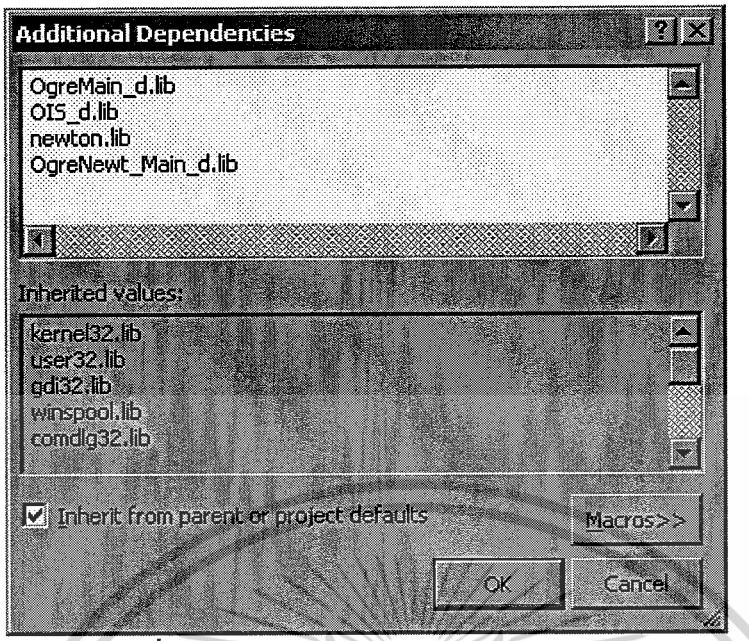
3.7 หน้าต่างขวาเลือกที่ Additional Dependencies จากนั้นกดปุ่มเล็กๆ ด้านขวามือ จากนั้นเพิ่มบรรทัดต่างๆลงไปดังนี้

OgreMain_d.lib

OIS_d.lib

newton.lib

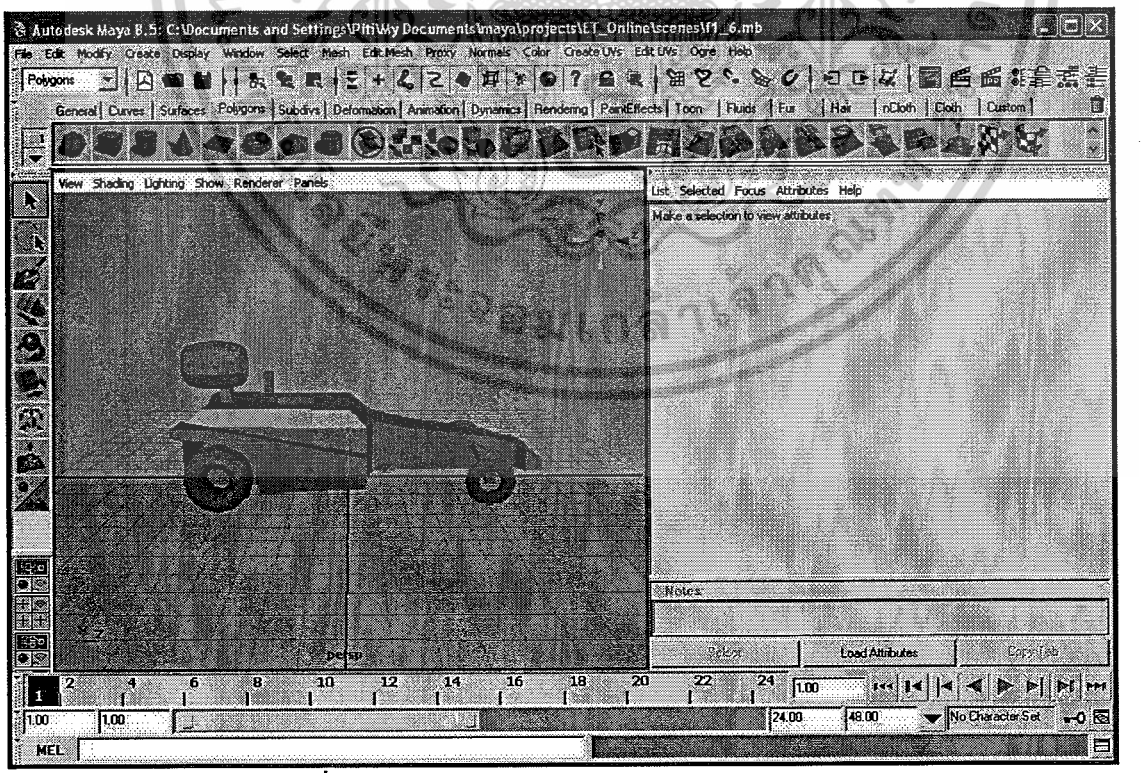
OgreNewt_Main_d.lib



รูปที่ 5.4 ตัวอย่างค่า Additional Dependencies

5.4 การสร้างตัวละครสามมิติ

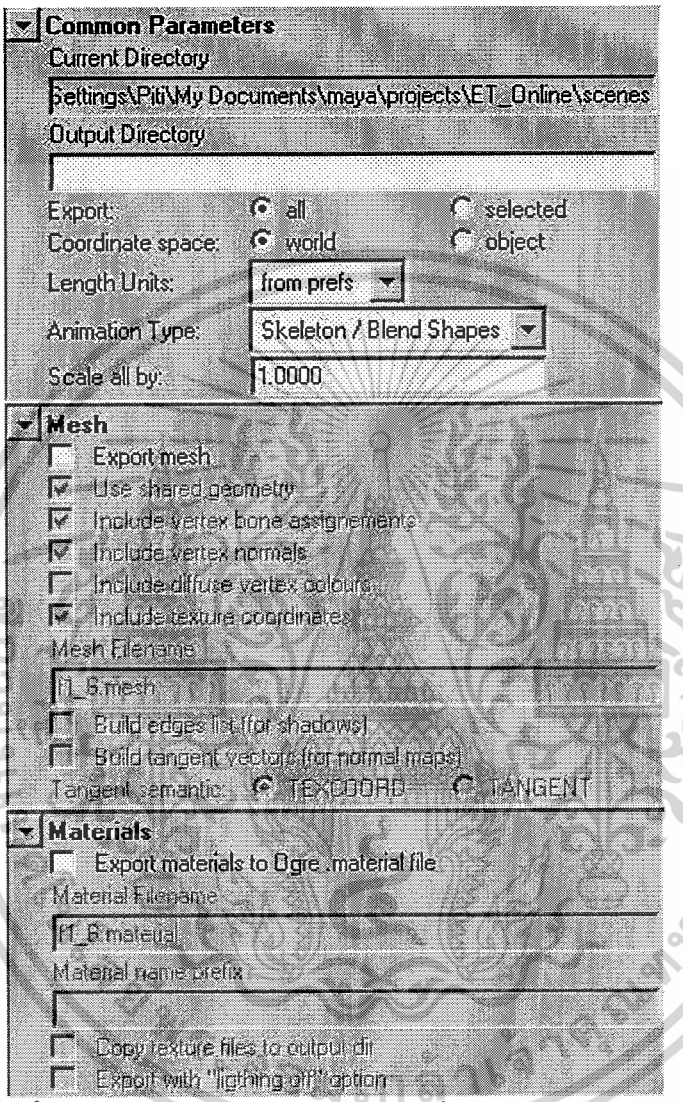
เครื่องมือการพัฒนา Maya version 8.5



รูปที่ 5.5 แสดงหน้าจอโปรแกรม Maya version 8.5

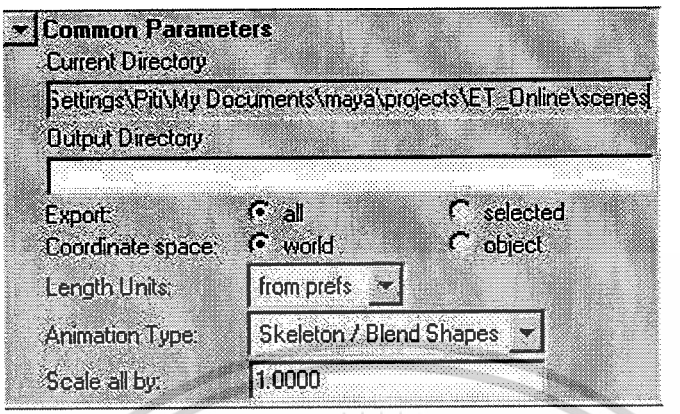
การแปลงตัวละครเพื่อใช้ใน OGRE

การนำตัวละครมาใช้งานผ่าน Ogre Engine มี module ในการ export คือ OgreExporter โดยวิธีใช้งานมีดังนี้



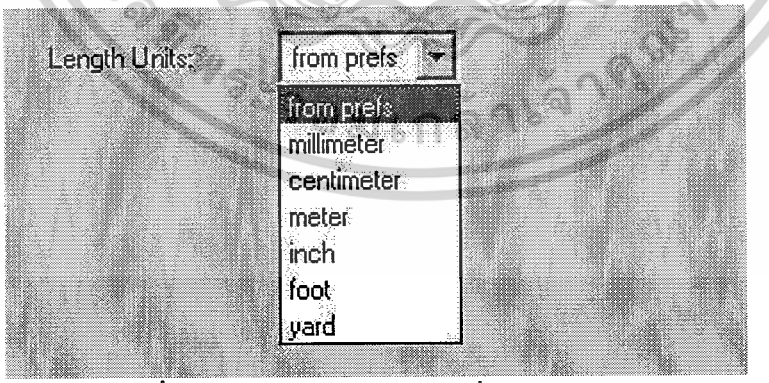
รูปที่ 5.6 แสดงหน้าต่าง Ogre Exporter ซึ่งมีการปรับแต่งต่างๆ

กลุ่มของ Common Parameter



รูปที่ 5.7 แสดงกลุ่มของ Common Parameter

- Output Directory คือโฟล์เดอร์ที่ต้องการให้ไฟล์ที่ทำการ export มาอยู่ในโฟล์เดอร์นี้
- Export จะมีให้เลือกระหว่าง all กับ selected ซึ่งก็คือจะใช้โมเดลทั้งหมดในฉากหรือเอาเฉพาะตัว ที่เลือกเท่านั้น
- Coordinate space โมเดลที่จะนำไปใช้จะใช้พิกัดของ world หรือ object ก็คือจะเลือกให้โมเดลใช้พิกัดของ world คือใช้พิกัดอ้างอิงของพิกัดของพื้นที่ทั้งหมดหรือใช้พิกัดของวัตถุ
- Length Units: จะเป็นการเลือกหน่วยของพื้นที่ของโมเดล ซึ่งถ้าเลือกหน่วยยิ่งมีขนาดใหญ่ก็จะทำให้โมเดลมีขนาดใหญ่ตามไปด้วย มีให้เลือกดังนี้

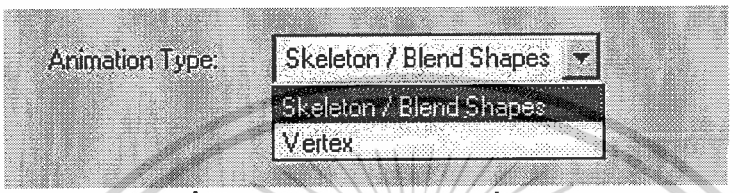


รูปที่ 5.8 แสดงหน่วยของขนาดที่ต้องการใช้ใน

- from prefs: ใช้หน่วยเดียวกับที่ใช้ในตอนสร้างโมเดล
- millimeter คือ มิลลิเมตร
- centimeter คือ เซนติเมตร

- meter คือ เมตร
- inch คือ นิ้ว
- foot คือ ฟุต
- yard คือ หลา

- Animation Type: คือชนิดของการเคลื่อนไหวว่าเป็นการเคลื่อนไหวที่มาจากรูปร่างหรือจากจุด

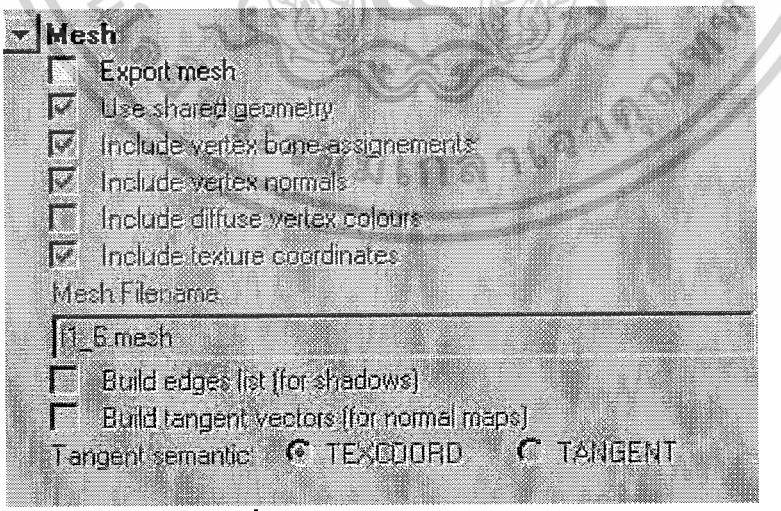


รูปที่ 5.9 แสดงชนิดของการเคลื่อนไหว

- Scale all by: คือ โมเดลที่จะเอาไปใช้ต้องการที่จะเปลี่ยนแปลงขนาดจากเดิมเท่าไหร่

รูปที่ 5.10 แสดงอัตราส่วนของแบบจำลองที่ต้องการนำไปใช้

กลุ่มของ Mesh

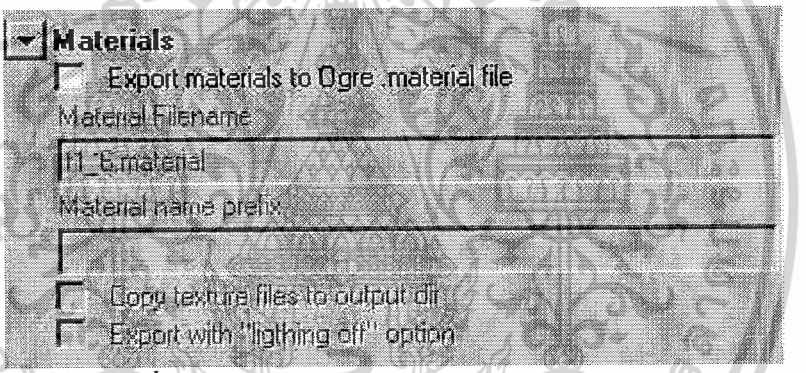


รูปที่ 5.11 แสดงกลุ่มของ Mesh

เนื่องจาก Ogre Engine ที่ได้เลือกใช้นั้นรองรับการใช้งานไฟล์โมเดลที่มีนามสกุล .mesh ซึ่งเป็นไฟล์ที่เป็นตัวโมเดลซึ่งเราจะเรียกเป็น mesh การระบุคุณลักษณะของตัว mesh ซึ่งโดยปกติจะเลือกดังนี้

- use shared geometry คือใช้การเปลี่ยนแปลงร่วมกับโมเดล
- Include vertex bone assignments คือการให้คิดพิกัดของกระดูกด้วย
- Include vertex normals คือ ให้ใช้ normal vector ตามที่กำหนดในโมเดล
- Include texture coordinates คือ ให้ทำการเก็บพิกัดของภาพที่ใช้กับโมเดล
- Mesh Filename คือ ชื่อของโมเดลที่จะนำไปใช้
- ส่วน Include diffuse vertex colours ไม่เลือกเพราะเราไม่ได้ใช้จุดในการขึ้นโมเดล จึงไม่จำเป็นต้องเอาสีของจุดไปด้วย

กลุ่มของ Material



รูปที่ 5.12 แสดงกลุ่มของ material

Material คือ สิ่งที่ใช้เพื่อกำหนดลักษณะพื้นผิวของโมเดลเพื่อใช้ในการแสดงผลและใส่ภาพเข้ากับตัวโมเดลได้ ซึ่งก็ต้องตั้งชื่อของไฟล์ .material และเลือก copy texture files to output dir ก็คือการนำไฟล์ภาพที่ใช้กับตัวโมเดลเพื่อให้เกิดลายต่างๆ คัดลอกมาไว้ที่โฟล์เดอร์ที่กำหนดไว้ด้วย

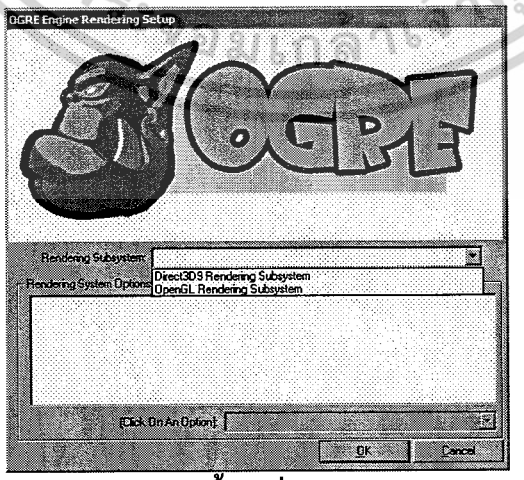
โครงสร้างของไฟล์ .material

```
material lambert1  // ใช้ material ชื่อ lambert1 ซึ่งเป็นชนิดที่ไม่มีการสะท้อนแสงใดๆ
{
        technique
        {
                pass
                {
```

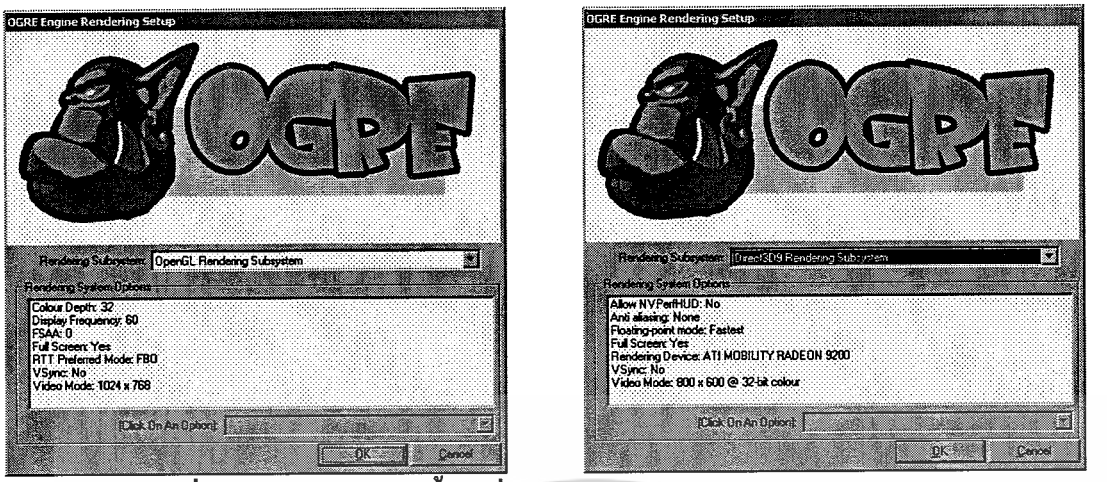
```
ambient 0 0 0 1        // สีของ Material
diffuse 1 1 1 1        // ความสว่างของสีใน material
specular 0 0 0 1
emissive 0 0 0

texture_unit
{
texture skate_body.jpg //ภาพที่ใช้ในการติดเข้ากับ Material
        tex_coord_set 0
        colour_op modulate
        scale 1 1
        scroll 0 0
        rotate 0
    }
  }
 }
}
```

สามส่วนที่ได้อธิบายไปเป็นส่วนหลักๆ ในการ export ตัวโมเดลให้สามารถนำมาใช้ในเกมส์ได้

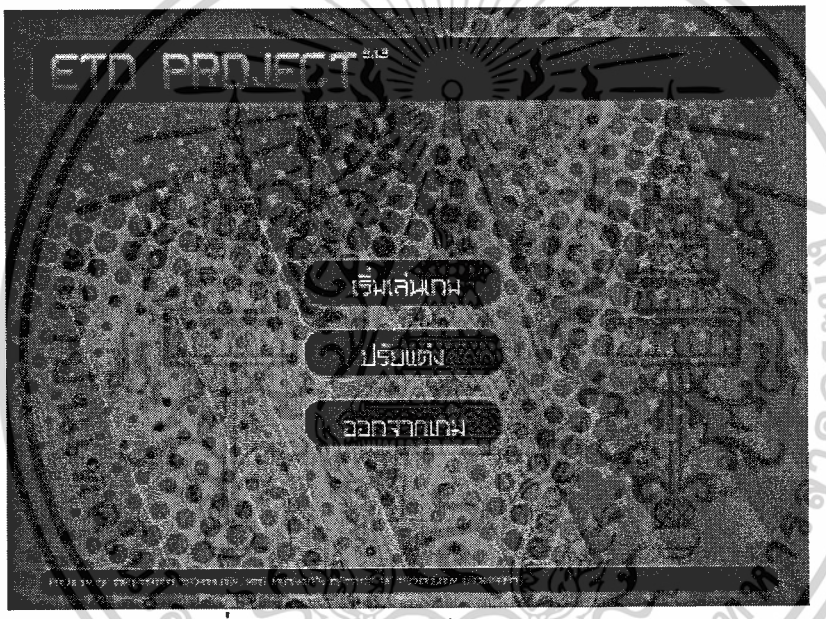
ภาพหน้าจอของระบบที่พัฒนา



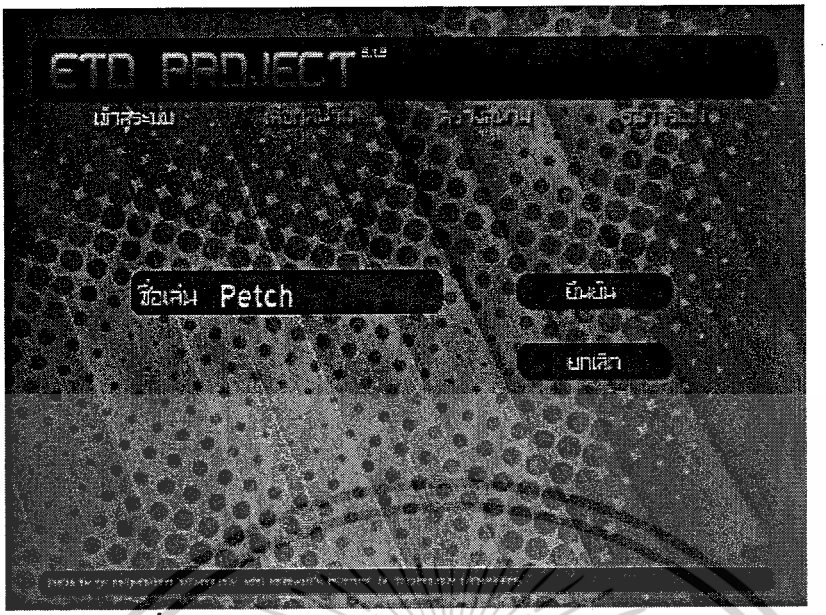
รูปที่ 5.13 ภาพแสดงหน้าต่างให้เลือกตั้งค่าเริ่มต้นแบบ OpenGL หรือแบบ Direct3D



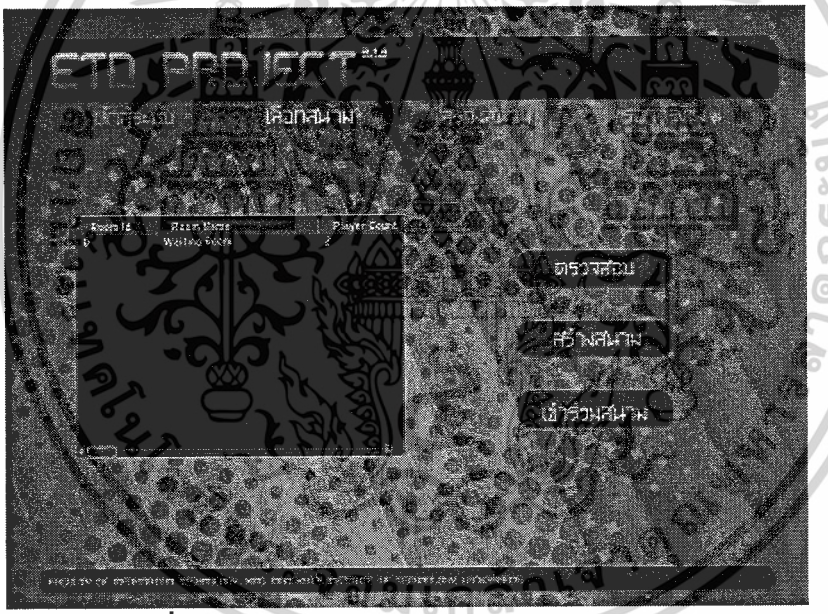
รูปที่ 5.14 ภาพแสดงการตั้งค่าเริ่มต้นแบบ OpenGL และแบบ Direct3D



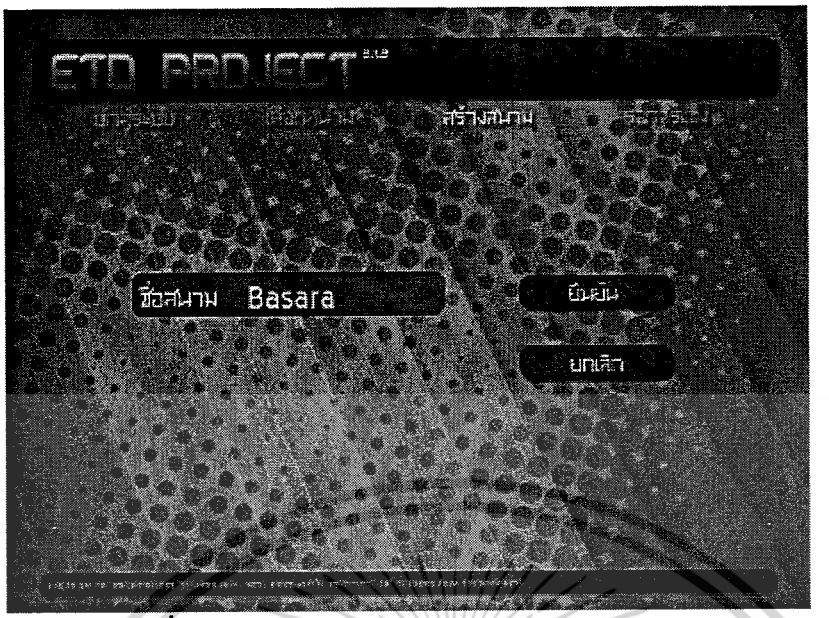
รูปที่ 5.15 ภาพแสดงหน้าจอหลักของเกม



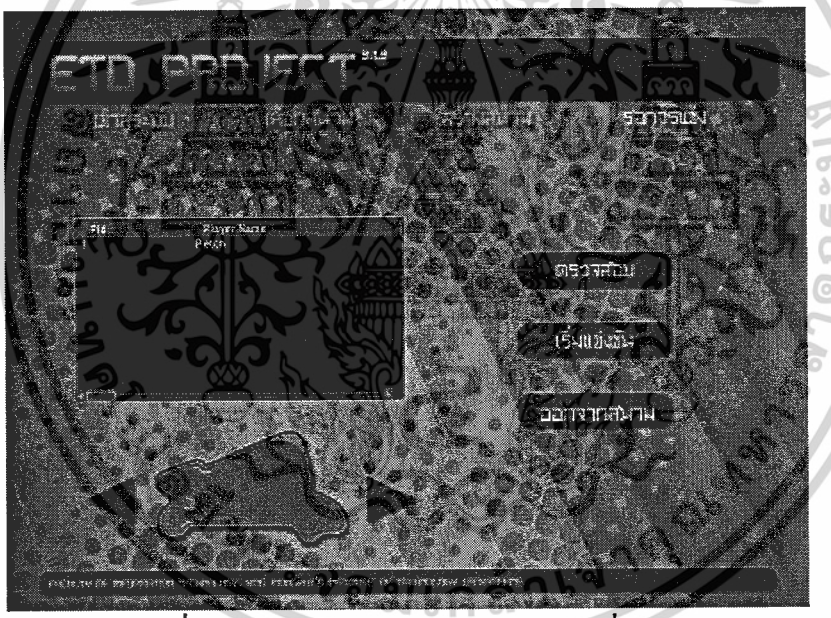
รูปที่ 5.16 ภาพแสดงหน้าจอการเข้าสู่ระบบของผู้เล่น



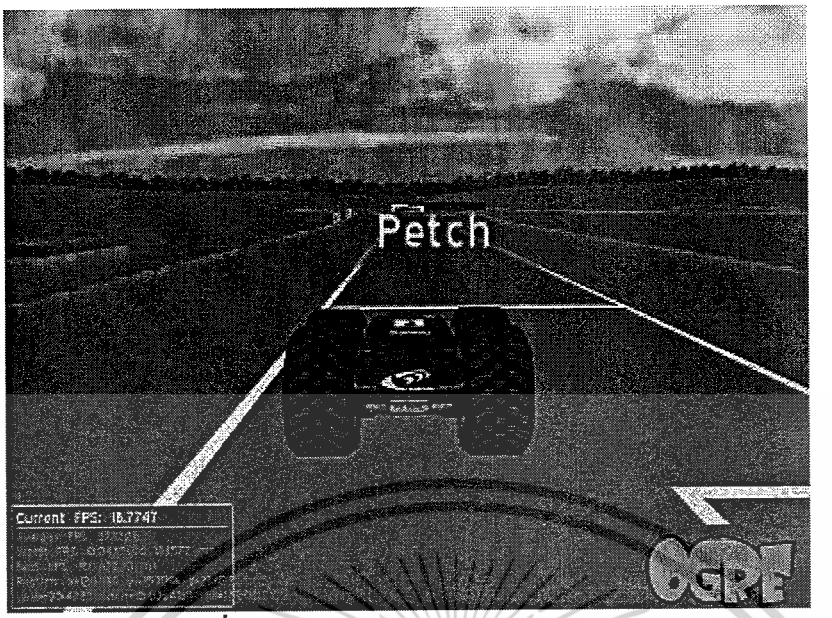
รูปที่ 5.17 ภาพแสดงหน้าจอการเลือกสนามแข่งขัน



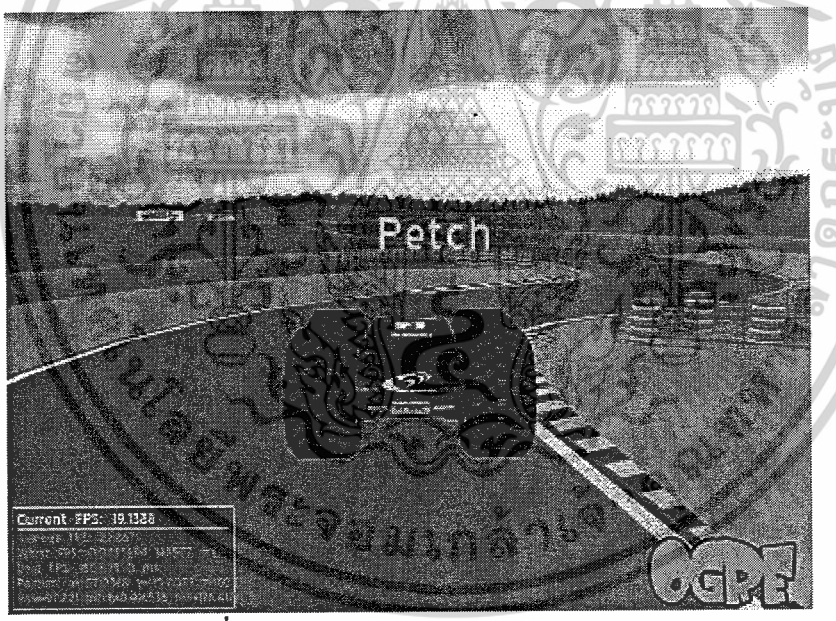
รูปที่ 5.18 ภาพแสดงหน้าจอการสร้างสนามแข่งขัน



รูปที่ 5.19 ภาพแสดงหน้าจอการรอยืนยันเริ่มเล่น



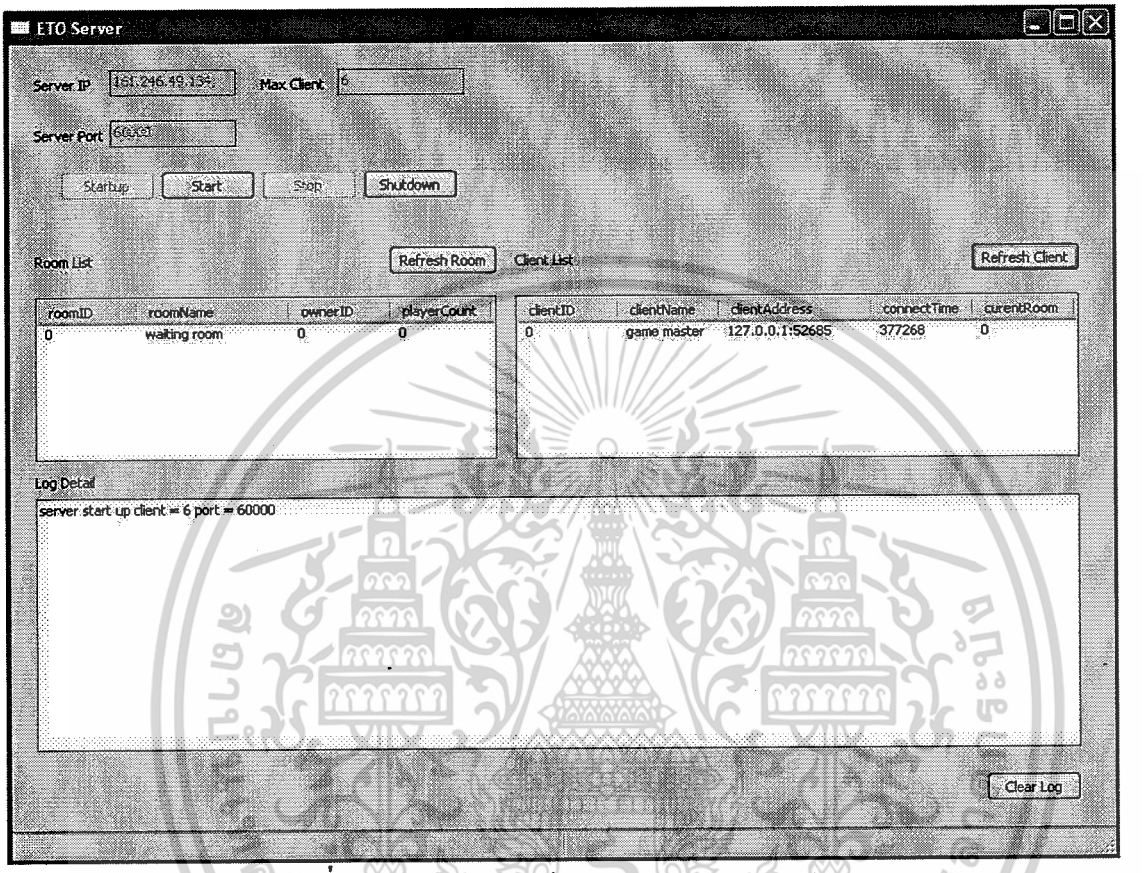
รูปที่ 5.20 ภาพแสดงหน้าจอการแข่งขัน 1



รูปที่ 5.21 ภาพแสดงหน้าจอการแข่งขัน 2

5.5 การสร้างแม่ข่ายเพื่อรองรับการติดต่อจากลูกข่าย

หน้าจอของระบบแม่ข่ายซึ่งสามารถทำการติดตามสถานะได้ผ่านทางหน้าจอ



รูปที่ 5.22 ภาพแสดงหน้าจอการทำงานของแม่ข่าย

บทที่ 6

สรุปผลการดำเนินงาน และข้อเสนอแนะ

6.1 ผลที่ได้จากการดำเนินงาน

- ศึกษา Ogre Engine
  - ทำการโหลดทรัพยากรเข้ามาใช้ในเกมได้
  - ควบคุมการเคลื่อนที่ของตัวละครได้
  - ใช้ Ogre Engine มาใช้ร่วมกับ Newton Engine ได้
- ศึกษา Maya
  - ทำการสร้างโมเดลจากโปรแกรมได้
  - ใช้วิธี texture mapping กับตัวละครได้
  - สร้างการเคลื่อนไหวโดยใช้ frame ได้
  - นำตัวละครมาแปลงเพื่อใช้ในเกมได้
- ขั้นตอนการวิเคราะห์ระบบ
  - กำหนดรายละเอียดภาพรวมการทำงานของระบบ
  - กำหนดรายละเอียดของการควบคุมการเคลื่อนที่ของตัวละคร
  - กำหนดลักษณะของสนามแข่งขัน
  - กำหนดลักษณะของเวทย์มนต์
  - กำหนดคุณสมบัติของกล่องปริศนา
  - กำหนดข้อมูลในการปรับแต่งภายในเกม
- ขั้นตอนการออกแบบระบบ
  - ออกแบบโครงสร้างของระบบในภาพรวม เพื่อให้เห็นการทำงานร่วมกันของระบบโดยภาพรวม
  - ออกแบบส่วนการทำงานหลักของระบบ เป็นส่วนที่ทำหน้าที่ประสานส่วนต่างๆ เพื่อให้สามารถทำงานร่วมกันได้
  - ออกแบบส่วนการทำงานรับข้อมูลเข้า ใช้ในการรอรับข้อมูลที่ถูกป้อนมาจากอุปกรณ์ต่างๆ เพื่อใช้ในการควบคุมเกม
  - ออกแบบส่วนการทำงานตัวละคร เพื่อใช้ในการเก็บข้อมูลของตัวละคร
  - ออกแบบส่วนการทำงานสนามแข่งขัน เพื่อใช้ในการเก็บข้อมูลเกี่ยวกับสนามทั้งหมด และควบคุมการเข้าเส้นชัยของแต่ละตัวละคร

- ขั้นตอนการพัฒนาระบบ
  - แม่ข่าย
    - พัฒนาระบบเน็ตเวิร์คให้สามารถติดต่อกับลูกข่ายได้
    - พัฒนาส่วนติดต่อกับผู้ใช้เพื่อให้สามารถดูสถานะของแม่ข่ายได้ง่ายขึ้น
    - พัฒนาส่วนการคำนวณทางฟิสิกส์เพื่อส่งผลลัพธ์ไปยังลูกข่ายได้
    - ทำการรวมส่วนประกอบต่างๆ เข้าด้วยกันได้
  - ลูกข่าย
    - พัฒนาโครงสร้างของระบบออกเป็นส่วนๆ
    - พัฒนาส่วนติดต่อกับผู้ใช้
    - พัฒนาส่วนการควบคุมเหตุการณ์ต่างๆ
    - พัฒนาส่วนการสร้างเกม
    - ทำการรวมส่วนประกอบเข้าด้วยกันได้

6.2 ปัญหาและอุปสรรค

ในช่วงแรกสมาชิกโครงงานใช้เวลาศึกษาชุดพัฒนาซอฟต์แวร์ต่างๆ โดยใช้เวลานาน เพราะขาดความคุ้นเคยกับ OGRE Newton และอื่นๆ จึงได้แลกเปลี่ยนความคิดเห็น และเสนอการทำซอฟต์แวร์ต้นแบบขึ้นมาชุดหนึ่ง ซึ่งทำให้เกิดความเปลี่ยนแปลง สมาชิกโครงงานเกิดความคุ้นเคยกับ เครื่องมือในการใช้ เทคนิคภาษาในการเขียนโปรแกรม ชุดพัฒนาซอฟต์แวร์ต่างๆ และช่วยในการออกแบบซอฟต์แวร์ในระยะต่อมา

มีการใช้เวลาทางด้านการสร้างข้อมูลโครงสร้างสามมิติของตัวละครมาก เพราะสมาชิกโครงงานมีประสบการณ์ในด้านการตกแต่งน้อย ทำให้ต้องใช้เวลาฝึกฝนการใช้เครื่องมือ ซึ่งจากการแลกเปลี่ยนความคิดเห็นได้เสนอว่าควรเน้นเป้าหมายไปที่ซอฟต์แวร์ และด้านความสวยงามให้มีการขอความช่วยเหลือจากผู้เชี่ยวชาญ

เนื่องจากเป็นโปรเจคเกม การจัดการทรัพยากรต่างๆ ในทุกส่วนต้องทำงานสอดคล้องกัน เมื่อเกิดข้อผิดพลาดจุดหนึ่ง ก็จะส่งผลให้เกิดข้อผิดพลาดขึ้นอีกหลายจุด จึงต้องมีการออกแบบที่ดี

ในการส่งข้อมูลระหว่างเครื่องลูกข่ายกับเครื่องแม่ข่าย การส่งข้อมูลต้องมีความสอดคล้องกัน หากรับช้าไปก็เกิดการกระตุก หรือถ้าเร็วไปก็จะประมวลผลไม่ทัน

การรวมส่วนประกอบต่างๆ ของทั้งลูกข่ายและแม่ข่าย เกิดความยุ่งยาก เนื่องจากต้องทำให้ข้อมูลในแต่ละส่วนมีความสัมพันธ์กัน และการส่งข้อมูลให้แต่ละส่วนต้องสอดคล้องกัน อีกทั้งการจัดการเกี่ยวกับการใช้ทรัพยากรของระบบให้น้อยที่สุด

6.3 ข้อเสนอแนะและแนวทางในการพัฒนาต่อ

ซอฟต์แวร์รุ่นถัดไปควรสามารถใส่ข้อกำหนดต่างๆ ในการเล่น อาจใส่ปัญญาประดิษฐ์อย่างง่ายเพื่อช่วยในการทดสอบตรรกะของซอฟต์แวร์ หลังจากนั้นจะเพิ่มส่วนของลูกเล่นภายในเกมเช่นการโจมตีกัน หรือผลกระทบต่อพื้นสนาม เพื่อให้เกิดความสนุกมากขึ้น

บรรณานุกรม

Gregory Junker, **"Pro OGRE 3D Programming"**, Apress, 2006

Newton Game Dynamics. **"Newton Game Dynamics Community"** [Online] Available: http://www.newtondynamics.com

Ogre 3D. **"Ogre 3D Community"** [Online.] Available: http://www.ogre3d.org

RakNet – Multiplayer game network engine. **"RakNet Community"** [Online] Available: http://www.jenkinsoftware.com

WxWidgets. **"WxWidgets Community"** [Online] Available: http://www.wxwidgets.org

ประวัติผู้เขียน

ชื่อ-นามสกุล	นายปีติ ศุภนิมิตวาสนา
วัน เดือน ปีเกิด	20 ธันวาคม 2528
ที่อยู่	75/5 หมู่บ้านรื่นฤดี ซ.รามคำแหง 180 ถ.รามคำแหง แขวงมีนบุรี เขตมีนบุรี กรุงเทพฯ 10510
โทรศัพท์	02-540-2948-9
โทรศัพท์เคลื่อนที่	086-8962073
อีเมล	PITI_NKIVY@YAHOO.COM
ประวัติการศึกษา	
2550	วิทยาศาสตรบัณฑิต คณะเทคโนโลยีสารสนเทศ สาขาเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ความชำนาญ	1) Game Programming
รางวัล	
พ.ศ. 2549	รางวัลที่สามในการแข่งขันหมวดการออกแบบซอฟต์แวร์ ในโครงการ Imagine cup 2006 โดยมี Microsoft (Thailand) เป็นผู้สนับสนุน
พ.ศ. 2549	รางวัลที่สองในการแข่งขัน Network Security Contest 2006

ชื่อ-นามสกุล	นายเลิศวุฒิ วีระธรากุล
วัน เดือน ปีเกิด	9 ธันวาคม 2529 จังหวัดกรุงเทพมหานคร
ที่อยู่	54/1005 ซอยพัฒนาการ69 ถนนพัฒนาการ แขวงประเวศ เขตประเวศ กรุงเทพมหานคร 10250
โทรศัพท์เคลื่อนที่	086-619-6601
อีเมล	LERTONDEB@HOTMAIL.COM
ประวัติการศึกษา	
2550	วิทยาศาสตรบัณฑิต คณะเทคโนโลยีสารสนเทศ สาขาเทคโนโลยีสารสนเทศ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
รางวัล	
พ.ศ. 2549	รางวัลที่สามในการแข่งขันหมวดการออกแบบซอฟต์แวร์ ในโครงการ Imagine cup 2006 โดยมี Microsoft (Thailand) เป็นผู้สนับสนุน