การพัฒนาเกมต่อสู้แบบโต้ตอบกับผู้เล่น

DEVELOPMENT OF INTERACTIVE FIGHTING GAME

นิติกร ศรีจรูญ
พิชิต ยิ้มประเสริฐ

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2547

DEVELOPMENT OF INTERACTIVE FIGHTING GAME

NITIKORN SRIJAROON

PICHIT YIMPRASERT

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LARDKRABANG

ACADAMIC YEAR 2004

หัวข้อปัญหาพิเศษ การพัฒนาเกมต่อสู้แบบโต้ตอบกับผู้เล่น
DEVELOPMENT OF INTERACTIVE FIGHTING GAME

ชื่อนักศึกษา นายนิติกร ศรีจรูญ 44050426
นายพิชิต ยิ้มประเสริฐ 44050440

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษา อ.กฤษฎา บุศรา

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตร์บัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ปีการศึกษา 2547

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ดร.กรกช ประชุมรักษ์	
กรรมการ	ผศ.ดร.ศรัณย์ อินทโกสุม	
กรรมการและอาจารย์ที่ปรึกษา	อ.กฤษฎา บุศรา	

(รองศาสตราจารย์ ดร.วีระ บุญจริง)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ การพัฒนาเกมต่อสู้แบบโต้ตอบกับผู้เล่น

ชื่อนักศึกษา นายนิติกร ศรีจรูญ 44050426
นายพิชิต ยิ้มประเสริฐ 44050440

ปริญญา วิทยาศาสตรบัณฑิต
ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์
สาขาวิชา วิทยาการคอมพิวเตอร์
ปีการศึกษา 2547
อาจารย์ที่ปรึกษา อ.กฤษฎา บุศรา

บทคัดย่อ

ปัจจุบันในวงการอุสาหกรรมเกมการผลิตโปรแกรมเกมคอมพิวเตอร์ได้มีการปรับตัวอย่างรวดเร็วและมีการแข่งขันกันสูงระหว่างบริษัทผู้ผลิต เพราะผู้ผลิตต่าง ๆ ได้นำเทคโนโลยีชั้นสูงเข้ามาช่วยในการพัฒนาคุณลักษณะและเทคนิคของเกม ดังนั้นทำให้คณะผู้ศึกษาได้เลือกศึกษาโครงงานเกี่ยวกับการสร้างโปรแกรมเกมคอมพิวเตอร์ โดยจะไม่ได้มองในแง่ของธุรกิจ หรือยอดขาย แต่จะมองในเรื่องของความท้าทาย ความแปลกใหม่รวมถึงการตอบสนองระหว่างคอมพิวเตอร์และผู้เล่น พร้อมกันนี้คณะผู้ศึกษาได้นำเสนอกระบวนการพัฒนาที่เป็นการจำลองสถานการณ์การต่อสู้ ผู้เล่นจะต้องเผชิญการต่อสู้ในรูปแบบต่าง ๆ ซึ่งคอมพิวเตอร์จะสุ่มเลือกรูปแบบการกระทำต่าง ๆ ในต่างสถานการณ์ ทำให้ผู้เล่นได้ฝึกฝนทักษะในการตอบสนองกับเหตุการณ์ต่าง ๆ ที่เกิดขึ้น

Special Project Title	DEVELOPMENT OF INTERACTIVE FIGHTING GAME
Student	Mr.Nitikorn Srijaroon 44050426
	Mr.Pichit Yimprasert 44050440
Degree	Bachelor of Science
Department	Mathematics and Computer Science, Faculty of Science
Programme	Computer Science
Academic Year	2004
Special Project Advisor	Kridsada Budsara

ABSTRACT

Today in game industry, a computer game publishing has been abruptly developed and has high potential challenges among computer game publishers since they have enhanced high technologies to approach game designs and techniques to conform game players in many ways. Thus it impacts us on making the decision of working on the computer game publishing as a special project. We do not only focus on any aspect of commercial or marketing product but we also concentrate on any of challenge, extraordinary, including being high interactive. Meanwhile, we have developed a computer game as a fighting simulation. This approach introduces the game player to encounter any kind of fighting routine at random depending on which situation is taking place. So the game player will practice on corresponding to any kind of situation.

กิตติกรรมประกาศ

ในการทำปัญหาพิเศษหัวข้อ การพัฒนาเกมต่อสู้แบบ 2 มิติ สามารถสำเร็จลุล่วงไปด้วยดี คณะผู้จัดทำต้องขอขอบพระคุณ อาจารย์กฤษฎา บุศรา อาจารย์ที่ปรึกษาปัญหาพิเศษฉบับนี้ ที่กรุณาให้คำแนะนำ และเป็นที่ปรึกษาในการแก้ปัญหาต่าง ๆ รวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

นอกจากนี้คณะผู้จัดทำต้องขอขอบพระคุณ บิดา มารดา ที่ได้ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์จนการทำปัญหาพิเศษนี้สำเร็จด้วยดี ขอขอบพระคุณอาจารย์ทุกท่านที่ให้ความรู้ทั้งในภาคทฤษฎีและภาคปฏิบัติ รวมทั้งเพื่อน ๆ ทุกคนที่มีส่วนช่วยเหลือในด้านต่าง ๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2548

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ(ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของปัญหา

เนื่องจากในปัจจุบันอุตสาหกรรมเกมได้มีการพัฒนาและเติบโตอย่างรวดเร็ว และคาดการณ์ว่าในอนาคตอุตสาหกรรมเกมจะมีการแข่งขันทางด้านการตลาดที่มีมูลค่าสูงขึ้น ด้วยเหตุนี้การศึกษาวิธีการและหลักในการสร้างระบบเกมจึงเป็นสิ่งที่น่าสนใจมาก ทำให้เราได้รับความรู้จากการสร้างระบบเกมด้วยการออกแบบคลาสตัวละครและฉากต่าง ๆ ในเกมด้วยโครงสร้างเชิงวัตถุ วิธีการติดต่อกับอุปกรณ์ฮาร์ดแวร์ที่รับข้อมูลเข้าทางคีย์บอร์ดและอุปกรณ์ฮาร์ดแวร์ที่แสดงการประมวลผลภาพของเกมออกทางหน้าจอ รวมถึงวิธีการใช้เครื่องมือช่วยต่าง ๆ ในการสร้างรูปภาพตัวละคร 3 มิติและฉาก 2 มิติ ภาพเคลื่อนไหวต่าง ๆ และวิธีการติดต่อกับระบบเสียงเพื่อแสดงเสียงประกอบเหตุการณ์ต่าง ๆ ในเกม อีกทั้งยังสามารถเขียนโปรแกรมเกมเพื่อโต้ตอบกับผู้เล่นในหลาย ๆ รูปแบบ คือ การสร้างฉากให้โต้ตอบกับผู้เล่นอีกด้วย โดยผู้วิจัยหวังว่าโครงงานนี้จะเป็นประโยชน์ต่อการนำไปศึกษาต่อในเรื่องของการพัฒนาเกมแบบ 2 มิติ ในเชิงวัตถุ เพื่อเป็นแนวทางเบื้องต้นสำหรับผู้ที่สนใจการพัฒนาเกม และยังเป็นแรงผลักดันให้เกิดอุตสาหกรรมเกมในประเทศไทยต่อไป

1.2 วัตถุประสงค์ของการทำปัญหาพิเศษ

1.2.1 เพื่อพัฒนาโปรแกรมทางคอมพิวเตอร์สำหรับการเล่นเกมต่อสู้ ที่สามารถเลือกเล่นได้ว่าต้องการเล่นกับคอมพิวเตอร์ หรือต้องการเล่นกับผู้เล่นอื่น

1.2.2 เพื่อฝึกสร้างทักษะและความสามารถในการพัฒนาโปรแกรมภาษาซีพลัสพลัสในเชิงวัตถุ

1.3 สมมุติฐานของการศึกษา

1.3.1 เกมสามารถติดต่อกับผู้เล่นได้หลายรูปแบบ โดยผู้เล่นไม่สามารถคาดการณ์ล่วงหน้าได้

1.3.2 สามารถเขียนโปรแกรมแสดงภาพเคลื่อนไหว และโต้ตอบกับผู้เล่นได้

1.3.3 เป็นประโยชน์ต่อผู้ที่นำไปศึกษาเป็นแนวทางต่อไป

1.4 ขอบเขตของปัญหา

1.4.1 เป็นเกมที่มีผู้เล่นตั้งแต่ 1 ถึง 2 คน

1.4.2 ผู้เล่นสามารถเลือกได้ว่าจะแข่งขันกับคอมพิวเตอร์ หรือแข่งขันกับผู้เล่นอื่น

1.4.3 ตัวละครที่ให้เลือกเล่นมี 2 ตัวได้แก่ นักดาบและนักเวทย์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้รับความรู้เกี่ยวกับโปรแกรมภาษาซีพลัสพลัสเชิงวัตถุ จากการพัฒนาโปรแกรม

1.5.2 ได้รับความรู้พื้นฐานของการสร้างระบบเกมโดยทั่วไป

1.5.3 ผู้เล่นได้รับประโยชน์และการฝึกทักษะต่าง ๆ ที่ได้สอดแทรกไว้ในเกม เช่น การฝึกนิ้วมือในการกดแป้นพิมพ์ ให้ตอบโต้ได้ตามภาพที่เห็น อีกทั้งยังสร้างทักษะต่าง ๆ ที่ใช้ในการแข่งขันระหว่างผู้เล่นกับคอมพิวเตอร์อีกด้วย

1.6 ขั้นตอนในการดำเนินงาน

1.6.1 ศึกษาเกี่ยวกับระบบการสร้างเกม

เป็นขั้นตอนการศึกษา วิธี และหลักการที่ใช้ในการสร้างเกม รวมทั้งเทคนิคต่าง ๆ ที่จะนำมาเขียนและพัฒนาระบบเกม

1.6.2 ศึกษาโปรแกรมและเครื่องมือต่าง ๆ

เป็นการศึกษาโปรแกรมและเครื่องมือ ที่จะนำมาใช้และพัฒนาในการเขียนโปรแกรม โดยจะเน้นการใช้โปรแกรมภาษาซีพลัสพลัส และโปรแกรมด้านกราฟฟิก เพื่อนำมาใช้ในการสร้างภาพ รวมถึงโปรแกรมในการประมวลผลด้านเสียงด้วย

1.6.3 การวิเคราะห์และออกแบบระบบ

วิเคราะห์และออกแบบระบบงาน โดยแบ่งออกเป็นส่วนๆเช่น ส่วนรับข้อมูล ส่วนประมวลผล ส่วนแสดงผล และออกแบบระบบ เพื่อให้การทำงานเกิดประสิทธิภาพมากที่สุด

1.6.4 เก็บรวบรวมเอกสารและข้อมูลต่าง ๆ

เป็นการนำเอกสารและข้อมูลที่เกี่ยวข้องมารวบรวม และใช้ประกอบการทำงาน โดยจะรวบรวมทั้งจาก หนังสือ ตำราต่าง ๆ เว็บไซต์ที่เกี่ยวข้อง รวมถึงข้อมูลที่ได้รับจากผู้มีความรู้ในแต่ละด้าน

1.6.5 พัฒนาโปรแกรม

เป็นขั้นตอนของการเขียนโปรแกรมตามขั้นตอนที่ได้ออกแบบไว้ในขั้นตอนการวิเคราะห์และออกแบบระบบ

1.6.6 การทดสอบโปแกรมและการปรับปรุงโปรแกรม

เป็นขั้นตอนของการทดสอบโปรแกรม และบอกถึงความสามารถทั้งหมดที่เป็นไปได้ของโปรแกรม

1.7 คำจำกัดความที่ใช้ในการศึกษา

1.7.1 SDK มาจากคำว่า Software Development Kit หมายถึงชุดซอฟต์แวร์ที่ใช้ในการพัฒนาซอฟต์แวร์

1.7.2 Driver คือตัวขับโปรแกรมควบคุมอุปกรณ์ หรือโปรแกรมอื่น

1.7.3 GDI มาจากคำว่า Graphic Device Interface ส่วนการติดต่อกับอุปกรณ์การแสดงผลทางด้านกราฟฟิก

1.7.4 GUID มาจากคำว่า Globally Unique Identifiers เป็นข้อกำหนดชนิดของอุปกรณ์ต่าง ๆ โดยระบุเป็นหมายเลข เพื่อให้ทราบว่า เป็นอุปกรณ์ประเภทหรือชนิดใด

1.7.5 Bitmap คือภาพที่ประกอบจากจุดเล็ก ๆ จำนวนมาก

1.7.6 Sprite คือภาพนิ่งหลาย ๆ ภาพที่มีลักษณะต่อเนื่องกัน ซึ่งมีประโยชน์ในการทำภาพเคลื่อนไหว

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 โปรแกรม Microsoft DirectX

2.1.1 ความเป็นมาเกี่ยวกับโปรแกรม Microsoft DirectX

ในการเขียนโปรแกรมติดต่อกับอุปกรณ์ฮาร์ดแวร์ต่าง ๆ โดยตรงแต่เดิมนั้นเป็นสิ่งที่ยุ่งยากและซับซ้อนมาก เนื่องจากผู้พัฒนาเกมต้องศึกษาคำสั่งต่าง ๆ ของอุปกรณ์ฮาร์ดแวร์แต่ละรุ่น แต่ละยี่ห้อ โดยคำสั่งต่าง ๆ ที่ใช้ควบคุมอุปกรณ์ฮาร์ดแวร์นั้นเขียนจากภาษาแอสเซมบลี ซึ่งเป็นภาษาระดับล่างที่มีโครงสร้างของภาษาใกล้เคียงกับภาษาเครื่อง ซึ่งเป็นภาษาที่มีความสามารถในการควบคุมอุปกรณ์ฮาร์ดแวร์ได้อย่างมีประสิทธิภาพ แต่เป็นภาษาที่ยากต่อการทำความเข้าใจของผู้พัฒนาเกม ยิ่งไปกว่านั้นเมื่ออุปกรณ์ฮาร์ดแวร์นั้น ๆ ถูกพัฒนาขึ้นโยการเพิ่มคำสั่งใหม่ ๆ เข้าไปหรือเปลี่ยนคำสั่งเดิมให้มีประสิทธิภาพมากขึ้น เป็นผลให้ผู้พัฒนาเกมจำเป็นต้องทำการศึกษาคำสั่งต่าง ๆ นั้นใหม่อีกครั้ง ทำให้ผู้พัฒนาเกมเสียเวลาและสร้างความลำบากในการพัฒนาโปรแกรมอีกด้วย

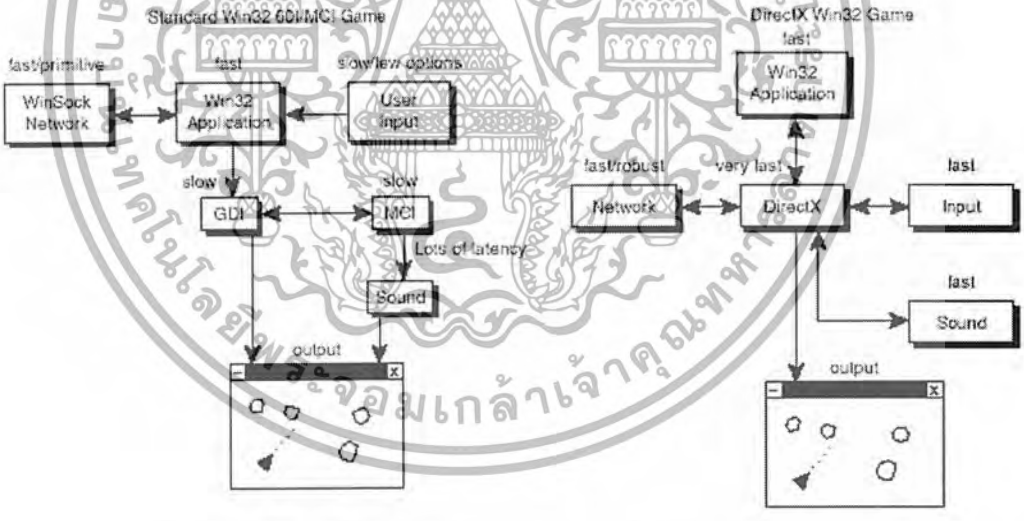
ในช่วงที่ไมโครซอฟต์สร้างระบบปฏิบัติการวินโดวส์ขึ้นมาใหม่นั้น สิ่งที่เป็นเอกลักษณ์ของวินโดวส์ คือ GUI (Graphic User Interface) โดยมี GDI (Graphic Device Interface) เป็นเครื่องมือสำหรับจัดการทางด้านภาพ ในขณะเดียวกันการพัฒนาเกมยังคงพัฒนาอยู่บนระบบปฏิบัติการดอส ดังนั้นบริษัทไมโครซอฟต์จึงเล็งเห็นถึงการพัฒนาเกมบนวินโดวส์ขึ้น ในระบบปฏิบัติการวินโดวส์ตั้งแต่เวอร์ชัน 3.1 ขึ้นไป ได้มีการใช้ไดรฟ์เวอร์ของอุปกรณ์ฮาร์ดแวร์ขึ้น โดยไดรฟเวอร์ประกอบด้วยชุดคำสั่งเพื่อที่จะควบคุมการเข้าถึงอุปกรณ์ฮาร์ดแวร์ต่าง ๆ ทำให้การเขียนโปรแกรมควบคุมการทำงานของอุปกรณ์ฮาร์ดแวร์ของผู้พัฒนาเกมง่ายและสะดวกขึ้น แต่กระนั้นผู้พัฒนาเกมยังคาดหวังว่าระบบปฏิบัติการวินโดวส์รุ่นใหม่ ๆ จะพัฒนาชุดคำสั่งที่ทำหน้าที่เป็นตัวประสานงานระหว่างอุปกรณ์ฮาร์ดแวร์ต่าง ๆ โดยทำผ่านตัวต่อประสานเพียงตัวเดียวได้ ซึ่งจะทำให้การพัฒนาโปรแกรมสะดวกขึ้น เพราะผู้พัฒนาเกมเพียงแค่ทำการศึกษาชุดคำสั่งของตัวต่อประสานตัวนั้นเพียงอย่างเดียวก็เพียงพอกับการพัฒนาโปรแกรมแล้ว

ด้วยเหตุนี้บริษัทไมโครซอฟต์ได้พัฒนาชุดคำสั่งพัฒนาซอฟท์แวร์ (Software Development Kits: SDK) ที่ทำหน้าที่เป็นตัวประสานงานระหว่างผู้พัฒนาเกมกับอุปกรณ์ฮาร์ดแวร์ในการพัฒนาซอฟท์แวร์บนวินโดวส์ เพื่อให้การเข้าถึงอุปกรณ์ฮาร์ดแวร์ทำได้ง่าย โดยตอนแรกได้เน้นเพื่อใช้พัฒนาเกมคอมพิวเตอร์โดยเฉพาะ โดยมีชื่อเรียกว่า "Game Kits" ต่อมาได้เพิ่มความสามารถทางด้านภาพกราฟิกส์, งานด้านสื่อประสม, งานด้านเน็ตเวิร์ก, งานด้านการรับข้อมูลอินพุต และงาน

ภาพกราฟิกส์ 3 มิติ ดังนั้นชุดคำสั่งนี้จึงถูกเรียกใหม่ว่า "Microsoft DirectX SDK" และปัจจุบันบริษัทไมโครซอฟต์ได้พัฒนา DirectX จนถึงรุ่น 9.0 แล้ว

2.1.2 หลักการทำงานของ DirectX

DirectX DirectX คือ ชุดคำสั่งของ Dynamic Link Library (DLL) ซึ่งทำหน้าที่เป็นตัวประสานชั้นระหว่างโปรแกรมประยุกต์กับอุปกรณ์ฮาร์ดแวร์ให้มีความสัมพันธ์กันเสมือนเป็นชั้นเดียวกัน เนื่องจากต้องการให้โปรแกรมประยุกต์สามารถทำงานกับอุปกรณ์ฮาร์ดแวร์ได้อย่างรวดเร็ว แต่เดิมบริษัทไมโครซอฟต์มีชุดคำสั่งที่ใช้ติดต่อกับการ์ดแสดงผลซึ่งเรียกว่า GDI (Graphic Device Interface) ซึ่งมีการทำงานคล้ายกับ DirectX โดยติดต่อกับอุปกรณ์ฮาร์ดแวร์ต่าง ๆ โดยตรงโดยไม่ต้องใช้ไดรฟเวอร์ของการ์ดจอเอง แต่มีความสามารถช้ากว่า และมีคุณสมบัติบางอย่างที่ไม่สามารถทดแทน DirectX ได้ เช่น ในกรณีที่โปรแกรมประยุกต์เรียกใช้อุปกรณ์ฮาร์ดแวร์แต่ DirectX ตรวจไม่พบอุปกรณ์ฮาร์ดแวร์ตัวนั้น ดังนั้น DirectX จะทำการจำลองอุปกรณ์ฮาร์ดแวร์ตัวนั้น เสมือนว่ามีอุปกรณ์ฮาร์ดแวร์ตัวนั้นอยู่ในระบบ แต่ในทางตรงกันข้างคุณสมบัตินี้ไม่มีอยู่ใน GDI



รูปที่ 2.1 การทำงานของ DirectX เปรียบเทียบกับ GDI/MCI

ยิ่งไปกว่านั้น DirectX ยังมีคุณสมบัติพิเศษโดยเกมที่เขียนด้วย DirectX เวอร์ชันระดับต่ำสามารถนำมาประมวลผลใน DirectX เวอร์ชันสูงก็ได้ อีกทั้งบริษัทไมโครซอฟต์ยังพัฒนาเทคโนโลยีของ DirectX ให้สนับสนุนการทำงานเชิงวัตถุ เพื่อให้สามารถทำงานกับภาษาอื่น ๆ ได้ และยังนำเทคโนโลยี COM (Component Object Model) มาใช้งาน ซึ่งทำให้ผู้พัฒนาเกมได้รับ

ความสะดวกสบายในการใช้งานเป็นอย่างยิ่ง ปละยังสามารถควบคุมและสร้างการทำงานได้อย่างอิสระ

2.1.3 องค์ประกอบของ DirectX

2.1.3.1 Hardware Abstraction Layer: HAL

HAL เป็นชั้นที่ใช้ติดต่อกับอุปกรณ์ฮาร์ดแวร์ระดับล่างสุด ซึ่งมีไดรฟเวอร์อุปกรณ์ฮาร์ดแวร์ของผู้ผลิตนั้น ๆ เพื่อใช้ควบคุมอุปกรณ์ฮาร์ดแวร์โดยตรง และให้ประสิทธิภาพสูงสุดเพราะว่าเป็นชั้นที่ติดต่อโดยตรง

2.1.3.2 Hardware Emulation Layer: HEL

HEL เป็นชั้นที่อยู่ถัดจากชั้น HAL เข้ามา ซึ่งเป็นชั้นที่ทำการจำลองขั้นตอนวิธีของอุปกรณ์ฮาร์ดแวร์นั้น ๆ ในกรณีที่ DirectX ตรวจไม่พบ แต่การทำงานจะช้าลงเพราะเป็นการจำลองการทำงานจริงของอุปกรณ์ฮาร์ดแวร์นั้น

2.1.3.3 Direct Draw

เป็นส่วนที่แสดงภาพกราฟิกส์ทุกชนิดออกทางหน้าจอเบื้องต้น จึงถือเป็นส่วนที่สำคัญที่สุด เพราะว่ามันทำหน้าที่ควบคุมการ์ดแสดงผล โดยที่ Direct Draw รู้จักการเซตโหมดของจอภาพทุก ๆ โหมด อีกทั้งยังสนับสนุนการจัดการสี, การกำหนดขอบเขต และการทำแอนิเมชัน

2.1.3.4 Direct Sound

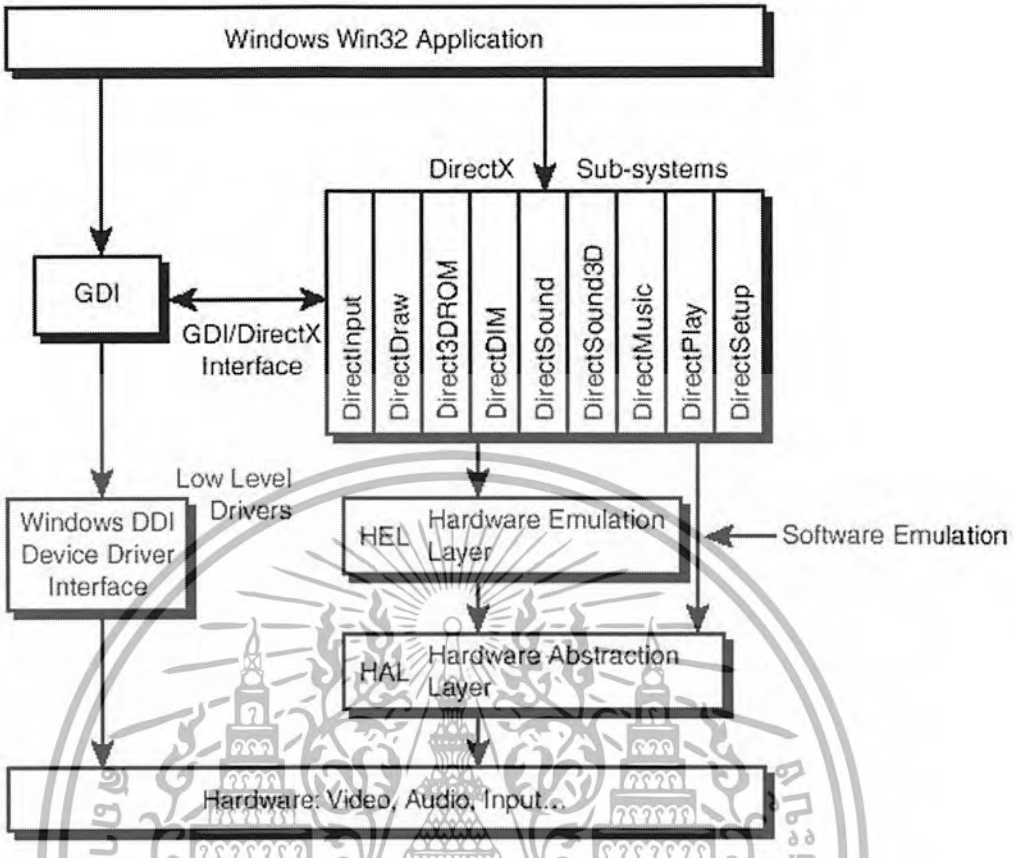
เป็นส่วนที่สนับสนุนเสียงแบบดิจิทัล แต่ไม่สนับสนุนเสียงแบบ MIDI ใช้จัดการเสียงแบบโมโนและแบบสเตริโออย่างมีประสิทธิภาพ ซึ่งประกอบด้วยหน่วยความจำ และการผสมเสียงฮาร์ดแวร์

2.1.3.5 Direct Sound 3D

เสียงแบบ 3 มิติเป็นส่วนที่มีความสามารถเหนือกว่า Direct Sound ทำให้เราสามารถกำหนดตำแหน่งของเสียงแบบ 3 มิติในอวกาศได้ ซึ่งต้องมีการคำนวณทางคณิตศาสตร์ โดยคำนวณพิกัดตำแหน่งของเสียงในระบบ 3 มิติ ทำให้เสียงมีความสมจริงมากขึ้น เสมือนวัตถุอยู่รอบตัวเรา

2.1.3.6 Direct Music

เป็นส่วนที่สนับสนุนเสียงแบบ MIDI อีกทั้งยังมีระบบ DLS (Downloadable Sound) ซึ่งเป็นระบบที่ทำให้คุณสามารถสร้างเครื่องมือที่สร้างเสียงแบบดิจิทัลที่สามารถควบคุมด้วยเทคโนโลยี MIDI ได้



รูปที่ 2.2 องค์ประกอบของ DirectX

2.1.3.7 Direct Input

ระบบนี้จะควบคุมอุปกรณ์อินพุตทุกชนิดไม่ว่าจะเป็นเมาส์, แป้นพิมพ์, จอยสติกส์, แพดเดิล, สเปซบอล ฯลฯ ยิ่งไปกว่านั้นมันยังสนับสนุนอุปกรณ์ Force Feedback ซึ่งมีเครื่องกระตุ้นไฟฟ้าและเซ็นเซอร์ของแรงสั่นสะเทือนที่ทำให้ผู้เล่นรู้สึกถึงแรงนั้น

2.1.3.8 Direct Play

เป็นส่วนที่จัดการเกี่ยวกับเครือข่าย โดยทำให้ผู้พัฒนาเกมสามารถเชื่อมต่อแบบย่อโดยใช้อินเทอร์เน็ต, โมเด็ม, การเชื่อมต่อโดยตรง ฯลฯ ซึ่งทำให้ไม่ต้องเขียนไดรฟเวอร์ หรือไม่ต้องใช้ซ็อกเก็ตเลย นอกจากนั้นมันยังสนับสนุนหลักการของช่วงเวลาซึ่งเป็นเกมที่กำลังดำเนินอยู่ และล็อบบี้ ซึ่งเป็นสถานที่ที่ผู้เล่นเกมเล่นอยู่ด้วยกัน

Direct Play สนับสนุนสถาปัตยกรรมเครือข่ายแบบหลายผู้เล่น โดยมันรับและส่ง กลุ่มข้อมูลเท่านั้น

2.1.3.9 Direct 3D Retained Mode: Direct 3D RM

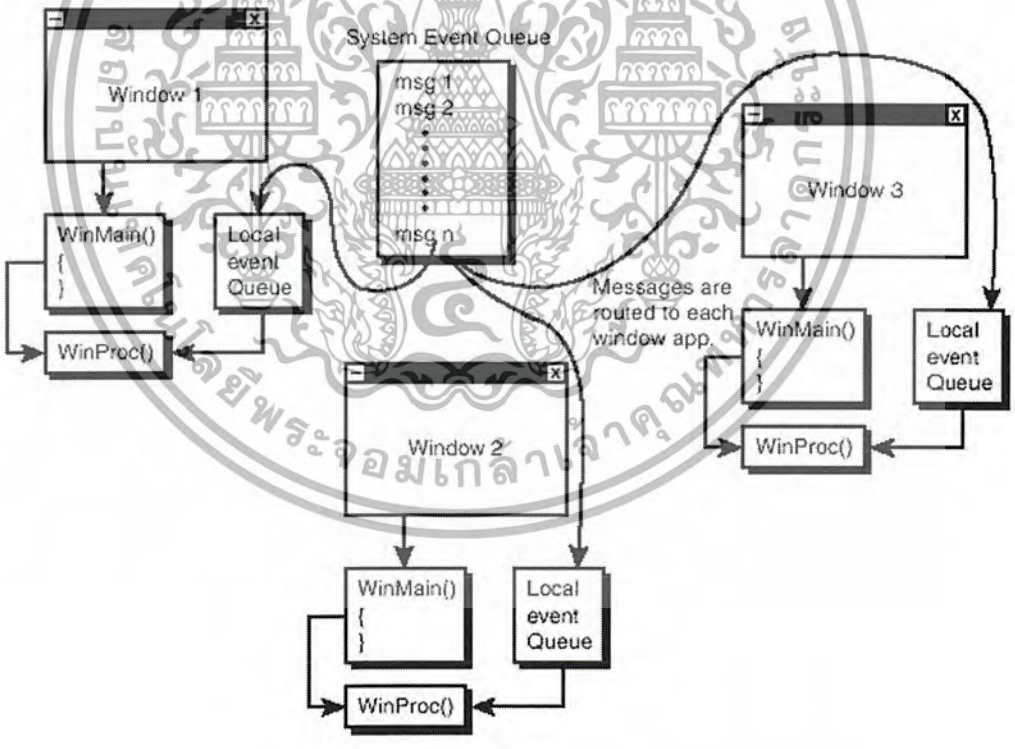
เป็นระบบ 3 มิติเชิงวัตถุและเชิงโครงร่างระดับสูง ที่ช่วยให้ผู้พัฒนาเกมสามารถเขียนโปรแกรม 3 มิติ แบบการตรวจตลอดได้

2.1.3.10 Direct 3D Immediate Mode: Direct 3D IM

เป็นระบบ 3 มิติระดับต่ำที่สนับสนุนการทำงานกับ frame war ด้วย OpenGL ผ่านฟังก์ชัน DrawPrimitive ซึ่งมีอยู่ใน DirectX 5.0 ขึ้นไป โดยทำหน้าที่ส่ง strip ภาพไปประมวลผลยัง rendering engine และเปลี่ยนสถานะด้วยฟังก์ชัน call แทนการประมวลผลบัฟเฟอร์

2.2 การเขียนโปรแกรมบนระบบปฏิบัติการวินโดวส์

วินโดวส์เป็นระบบปฏิบัติการแบบขึ้นอยู่กับเหตุการณ์โดยมีข้อความที่ส่งไปยังวินโดวส์เพื่อบอกให้รู้ว่าต้องทำอะไรและทำอย่างไร เป็นต้น ดังนั้นเราจึงควรศึกษาโครงสร้างของ WIN32 API ก่อนที่จะสร้างเกมขึ้นมาตามหัวข้อต่อไปนี้



รูปที่ 2.3 กลไกการส่งข้อความในวินโดว์

2.2.1 ไลบรารีของ Win32 API

เมื่อผู้พัฒนาเกมสร้างโปรแกรมประยุกต์ใด ๆ ขึ้นมาคอมไพเลอร์จะเชื่อมโยงกับส่วนของไฟล์ไลบรารีให้เองทั้งหมด ยกเว้นไฟล์ไลบรารีที่ไม่ได้ถูกกำหนดไว้ก่อนในตอนแรก เช่น เมื่อต้องการสร้างโปรแกรมประยุกต์ที่เกี่ยวกับการแสดงเสียงซึ่งต้องใช้ไฟล์ไลบรารี dsound.lib แต่คอมไพเลอร์ยังไม่รู้จักกับไฟล์ไลบรารีนี้ ดังนั้นผู้พัฒนาเกมต้องเพิ่มไฟล์นี้เข้าไปในไดเรกทอรีเอง

2.2.2 การเพิ่มไฟล์เฮดเดอร์

ไฟล์ที่จะเพิ่มเข้าไปในโปรแกรมประยุกต์เป็นไฟล์ไลบรารีและไฟล์เฮดเดอร์ที่จำเป็นสำหรับโปรแกรมประยุกต์นั้น ๆ ซึ่งโดยทั่วไปผู้พัฒนาเกมจะกำหนดไฟล์ไลบรารีและไฟล์เฮดเดอร์พื้นฐานไว้ดังนี้

```
#define WIN32_LEAN_AND_MEAN
#include<windows.h>
#include<windowsx.h>
```

ผู้พัฒนาเกมอาจจะต้องเพิ่มไฟล์ที่จำเป็นสำหรับโปรแกรมประยุกต์นั้นเข้าไปอีก เช่น stdio.h, conio.h หรือ math.h เป็นต้น

2.2.3 ฟังก์ชัน WinMain

ฟังก์ชันหลักของวินโดว์ทำหน้าที่เหมือนกับฟังก์ชัน main ในภาษาซีพลัสพลัส ซึ่งทำหน้าที่กำหนดค่าเริ่มต้นให้กับวินโดว์ โดยการประกาศต้นแบบของฟังก์ชัน WinMain เป็นดังนี้

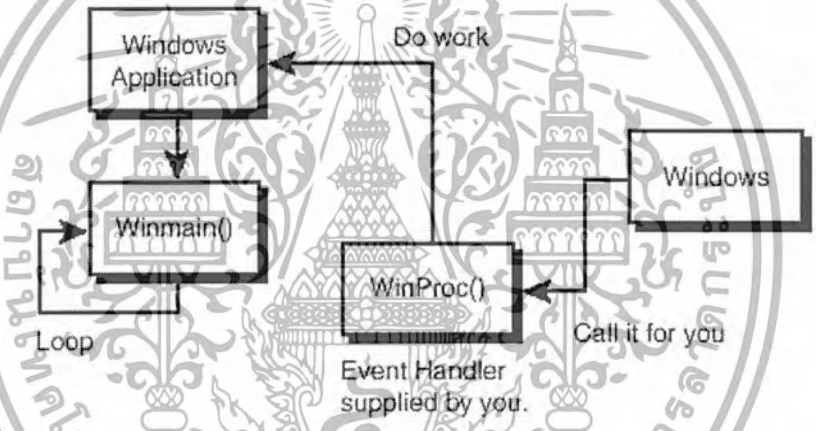
```
int WINAPI WinMain(HINSTANCE   hinstance, //handle of current of instance
                   HINSTANCE   hprevinstance, //handle of previous instance
                   LPSTR       lpcmline, //address of command line
                   int         ncmdshow ); //show state of window
```

2.2.4 คลาสของวินโดว์

คลาสของวินโดว์เป็นคลาสที่ประกอบด้วยองค์ประกอบทั่วไปของวินโดว์ ทำให้ผู้พัฒนาเกมสามารถสร้างวินโดว์ได้ตามคลาสของวินโดว์ที่กำหนดขึ้น โดยการประกาศโครงสร้างคลาสของวินโดว์เป็นดังนี้

```
typedef struct _WNDCLASS
{
    UINT        style; // style flags
    WNDPROC     lpfnWndProc; // pointer to event handler
```

```
    Int         cbClsExtra; // extra byte
    Int         cbWndExtra; //extra byte
    HANDLE      hInstance; //handle of instance
    HICON       hIcon; //handle to Icon
    HCURSOR     hCursor; //handle to cursor
    HBRUSH      hbrBackground; //handle to background brush
    LPCTSTR     lpszMenuName; //name of Menu
    LPCTSTR     lpszClassName;//name of class
} WNDCLASS;
 WNDCLASS       wndclass; // a declaration of a window class
```



รูปที่ 2.4 การทำงานของฟังก์ชัน Winmain กับตัวจัดกระทำตามเหตุการณ์

2.2.5 ฟังก์ชัน RegisterClass

เป็นการขึ้นทะเบียนคลาสของวินโดว์เพื่อเอาไว้เรียกใช้ในภายหลังเมื่อผู้พัฒนาเกมเรียกใช้ฟังก์ชัน CreateWindow หรือ CreateWindowEx ถ้าขึ้นทะเบียนสำเร็จฟังก์ชันจะคืนค่า atom มาให้ แต่ถ้าไม่สำเร็จฟังก์ชันจะคืนค่าศูนย์มาให้แทน การรีจิสเตอร์คลาสทำได้ดังนี้

```
if (RegisterClass(&wndclass) == 0)
{
    //error
} // end if
```

การประกาศโครงสร้างของ RegisterClass เป็นดังนี้

```
typedef struct tagWNDCLASS
{
        UNIT        style;
        WNDPROC     lpfnWndProc;
        Int         cbClsExtra;
        Int         cbWndExtra;
        HANDLE      hInstance;
        HICON       hIcon;
        HCURSOR     hCursor;
        HBRUSH      hbrBackground;
        LPCTSTR     lpszMenuName;
        LPCTSTR     lpszClassName;
} WNDCLASS;
```

2.2.6 ฟังก์ชัน CreateWindow

การสร้างวินโดว์อ็อบเจกต์ขึ้นมาต้องทำการเรียกฟังก์ชัน CreateWindow โดยการประกาศโครงสร้างของฟังก์ชัน CreateWindow เป็นดังนี้

```
HWND CreateWindow (
        LPCTSTR     lpClassName, //pointer to class name (string)
        LPCTSTR     lpWindowName, //pointer to window title (string)
        DWORD       dwStyle, // windows style flags
        int         x, // ตำแหน่งแนวแกน x ของวินโดว์
        int         y, //ตำแหน่งแนวแกน y ของ วินโดว์
        int         nWidth, // ความกว้างของวินโดว์
        int         nHeight, // ความสูงของวินโดว์
        HWND        hWndParent, //handle to parent (usually NULL)
        HMENU       hMenu, //handle to menu (usually NULL)
        HANDLE      hInstance, //handle to application instance
        LPVOID      lpParam //pointer to startup creation data (NULL)
);
```

ตารางที่ 2.1 ค่าตัวบ่งชี้ของพารามิเตอร์ dwStyle

ค่าตัวบ่งชี้ของ dwStyle	ความหมาย
WS_BORDER	สร้างวินโดว์ที่มีเส้นขอบบาง
WS_CAPTION	สร้างวินโดว์ที่มีไทเทิลบาร์ (รวมทั้ง WS_BORDER)
WS_DISABLE	สร้างวินโดว์ที่ไม่สามารถรับอินพุตของผู้ใช้ได้
WS_OVERLAPPED	สร้างวินโดว์ที่ซ้อนเหลื่อมกันได้
WS_POPUP	สร้างวินโดว์แบบ Pop-up
WS_VISIBLE	สร้างวินโดว์ที่สามารถมองเห็นได้

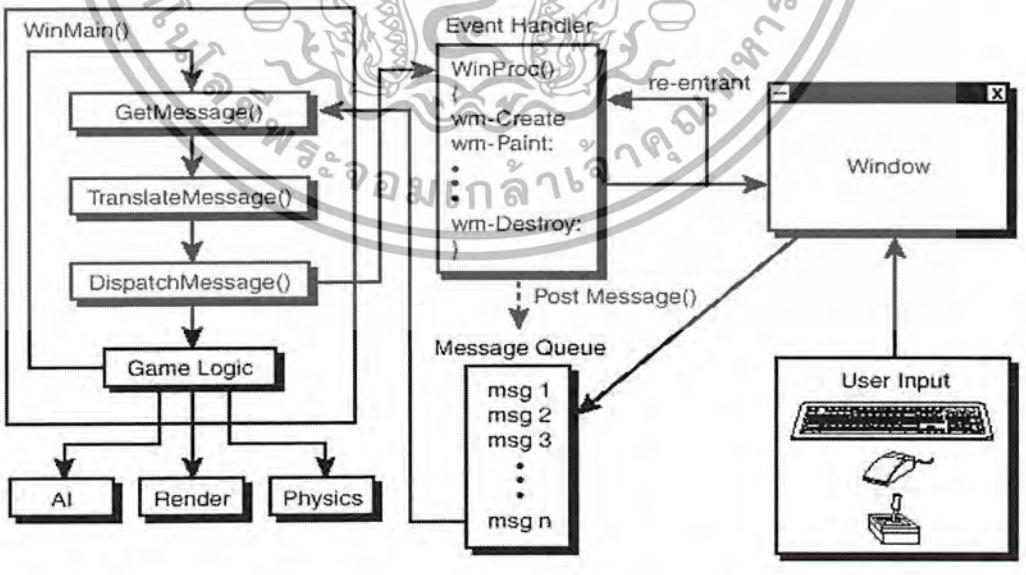
2.2.7 ฟังก์ชัน ShowWindow

การแสดงวินโดว์บนจอภาพ ในกรณีที่ตั้งค่าตัวบ่งชี้ของพารามิเตอร์ dwStyle ไม่เป็น WS_VISIBLE ผู้พัฒนาเกมต้องเรียกฟังก์ชัน ShowWindow ขึ้นมาด้วยเสมอ

```
BOOL ShowWindow(HWND    hWnd, // handle of window
                int     nCmdshow); // show state of window
```

2.2.8 ลูปของเหตุการณ์

ลูปของเหตุการณ์หลักเป็นลูปที่คอยรับข้อความจากวินโดว์แล้วนำไปเก็บไว้ในคิวของข้อความ จากนั้นส่งต่อไปยังตัวจัดกระทำตามเหตุการณ์เพื่อทำหน้าที่ตามข้อความที่ได้รับดังรูปที่ 2.5



รูปที่ 2.5 ลูปเหตุการณ์หลัก

จากรูปที่ 2.5 ฟังก์ชันใน WinMain ประกอบด้วย 3 ฟังก์ชันย่อย คือ

2.2.8.1 **ฟังก์ชัน** GetMessage

เป็นฟังก์ชันที่ทำหน้าที่รับข้อความจากคิวของข้อความที่เกี่ยวข้องกับวินโดว์อ็อปเจกต์นั้นมาทำให้อยู่ในรูปโครงสร้างเฉพาะ โดยฟังก์ชันจะคืนค่าจริงกลับมาเมื่อฟังก์ชันทำงานสำเร็จ ซึ่งถ้าข้อความว่างฟังก์ชันนี้จะรอจนกว่าจะมีข้อความเข้ามาใหม่ การประกาศรูปแบบของฟังก์ชันเป็นดังนี้

```
BOOL GetMessage(LPMSG   lpMsg, //pointer to message structure
                HWND    hWnd, //handle of window
                UINT    wMsgFilterMin, //first message
                UINT    wMsgFilterMax); //last message
```

2.2.8.2 **ฟังก์ชัน** TranslateMessage

เป็นฟังก์ชันที่แปลข้อความที่อยู่ในคิว จากประเภทแป้นเสมือนให้เป็นประเภทอักขระ โดยข้อความประเภทอักขระจะถูกทำไปไว้ในคิวของข้อความเพื่อให้ฟังก์ชัน GetMessage อ่านข้อความในครั้งต่อไป โดยฟังก์ชันจะคืนค่าจริงกลับมาเมื่อฟังก์ชันทำงานสำเร็จ การประกาศรูปแบบของฟังก์ชันเป็นดังนี้

```
BOOL TranslateMessage(CONST MSG *lpmsg); //pointer to message structure
```

2.2.8.3 **ฟังก์ชัน** DispatchMessage

เป็นฟังก์ชันที่ทำการส่งข้อความไปยังฟังก์ชัน WindowProc เพื่อทำงานตามข้อความที่ได้รับ โดยฟังก์ชันจะคืนค่าเฉพาะที่มีความหมายขึ้นกับข้อความที่ส่งไปประมวลผล และเราจะไม่สนใจค่าที่ส่งกลับ การประกาศรูปแบบของฟังก์ชันเป็นดังนี้

```
LONG DispatchMessage( CONST MSG *lpmsg); //pointer to message structure
```

2.2.9 **ตัวจัดกระทำตามเหตุการณ์**

เป็นฟังก์ชัน WindowProc ซึ่งจัดการกับทุก ๆ ข้อความของวินโดว์ที่สนใจในขณะนั้น โดยฟังก์ชันจะคืนผลลัพธ์ของการประมวลผลข้อความมาให้ และค่าที่คืนนั้นจะขึ้นอยู่กับข้อความที่ได้รับ การประกาศรูปแบบของฟังก์ชันเป็นดังนี้

```
LRESULT CALLBACK WindowProc(HWND    hwnd, //the window
                            UINT    msg , //the message itself
                            WPARAM  wParam, //more info on message
                            LPARAM  lParam ); //more info on message
```

2.3 การเขียนโปรแกรมแบบขับเคลื่อนด้วยเหตุการณ์

สิ่งที่ต่อคำนึงถึงเป็นอย่างแรกในการเขียนโปรแกรมบนระบบปฏิบัติการวินโดวส์ คือ การเขียนโปรแกรมแบบขับเคลื่อนด้วยเหตุการณ์ ซึ่งเป็นการโปรแกรมที่ทำงานตามเหตุการณ์ที่เกิดขึ้น โดยระบบปฏิบัติการจะเป็นตัวบอกโปรแกรมว่ามีเหตุการณ์อะไรเกิดขึ้นบ้าง หากโปรแกรมได้รับเหตุการณ์ที่สนใจ มันก็จะทำงานตามที่ผู้พัฒนาเกมได้เขียนโปรแกรมไว้

หากเปรียบเทียบกับการเขียนโปรแกรมโดยทั่วไปซึ่งจะทำงานแบบเรียงลำดับตามผู้พัฒนาเกมได้เขียนโปรแกรมไว้ แต่สำหรับการเขียนโปรแกรมตามเหตุการณ์จะให้ผู้เขียนโปรแกรมควบคุมโปรแกรมได้มากกว่าผู้ใช้

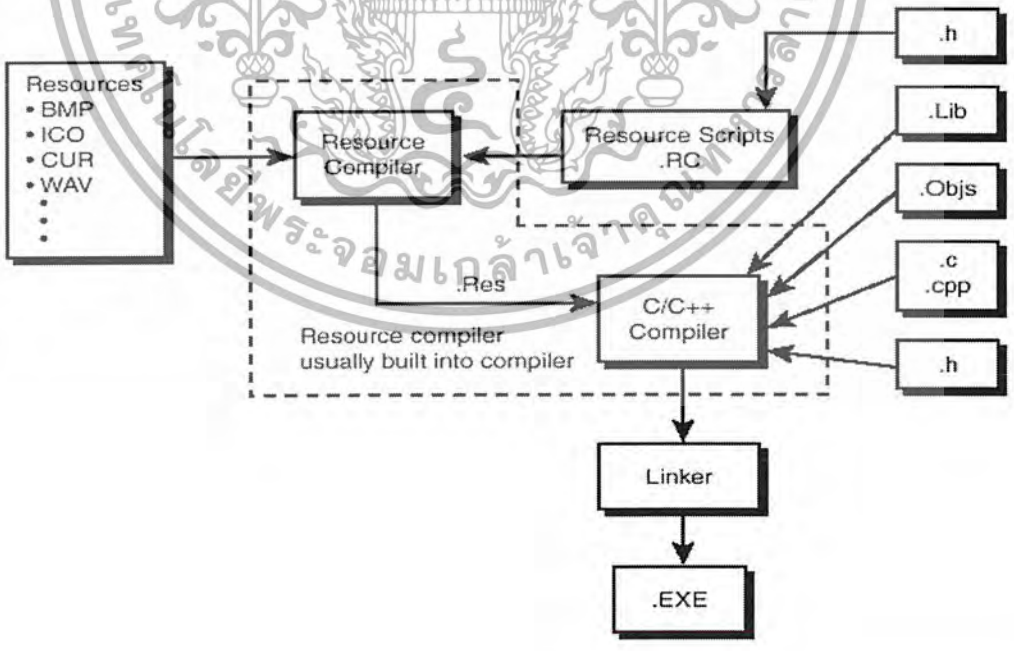
2.4 วิธีการแสดงภาพบนจอภาพ

2.4.1 ความรู้เบื้องต้นในการแสดงภาพด้วยคำสั่งของ DirectDraw

Direct Draw เป็นไลบรารีของชุดทำสั่งพัฒนาโปรแกรม DirectX ของภาษาซีพลัสพลัสซึ่งประกอบด้วยไฟล์เฮดเดอร์ ddraw.h และไฟล์ไลบรารี ddraw.lib เวลาเรียกใช้ให้ประกาศดังนี้

```
#include <ddraw.h>
```

และกำหนดไฟล์ไลบรารี ddraw.lib ให้กำหนดในไดเรกทอรีลิงค์ไลบรารี



รูปที่ 2.6 ทรัพยากรที่ใช้สร้างโปรแกรมประยุกต์ของ Win32 DirectX

2.4.2 การสร้างอ็อปเจกต์ของ DirectDraw

ผู้พัฒนาเกมจะสร้างอ็อปเจกต์ของ Direct Draw หลังจากที่สร้างอ็อปเจกต์ของวินโดว์แล้ว โดยรูปแบบของฟังก์ชันเป็นดังนี้

```
HRESULT DirectDrawCreate(GUID            FAR *lpGUID, //guid of object
                         LPDIRECTDRAW    FAR *lplpDD, //receive interface
                         IUnknown        FAR *pUnkOuter); //com stuff
```

ฟังก์ชันนี้จะทำการสร้างอ็อปเจกต์ของ DirectDraw ไว้ที่ตำแหน่งแอดเดรส lplpDD และแสดงผลแบบ lpGUID ซึ่งถ้ากำหนดค่าเป็น ค่าว่าง การแสดงผลจะเป็นแบบ โดยปริยาย ฟังก์ชันนี้จะแจ้งผลผ่านค่า HRESULT เป็นชนิดข้อมูลชนิดหนึ่งของ DirectX ซึ่งใช้สำหรับตรวจสอบการทำงานของฟังก์ชัน ไม่ว่าจะทำงานสำเร็จหรือไม่ก็ตาม ถ้าสำเร็จจะส่งค่า DD_OK หรือ DirectDraw Okay หากฟังก์ชันคืนค่าอื่น ๆ ที่ไม่ใช่ค่านี้ นั้นหมายถึงการทำงานของฟังก์ชันมีความผิดพลาดเกิดขึ้น

ตัวอย่าง

```
LPDIRECTDRAW    lpdd;  //สร้างตัวแปรที่ใช้เก็บค่าแอดเดรสอ็อปเจกต์ของ DirectDraw
//สร้างอ็อปเจกต์และตรวจสอบข้อผิดพลาด
if (DirectDrawCreate(NULL, &lpdd, NULL) != DD_OK)
{ //จัดการกับข้อผิดพลาด }
```

การเรียกฟังก์ชันของอ็อปเจกต์เป็นดังนี้

```
lpdd->ชื่อฟังก์ชัน();
```

หลังจากนั้นต้องลบอ็อปเจกต์เพื่อคืนหน่วยความจำให้กับระบบโดยใช้ฟังก์ชัน Release

```
lpdd->Release();
```

2.4.3 การกำหนดค่าการแสดงผลของวินโดว์ด้วยฟังก์ชัน SetCooperativeLevel ของ DirectDraw

การเขียนโปรแกรมเพื่อทำงานบนระบบปฏิบัติการวินโดวส์นั้น ผู้พัฒนาเกมต้องจัดสรรทรัพยากรต่าง ๆ โดยใช้ฟังก์ชันดังนี้

```
HRESULT SetCooperativeLevel (HWND hWnd, DWORD dwFlags);
```

HWND hWnd คือ แอดเดรสของวินโดว์ที่กำลังทำงานอยู่ในขณะนั้น โดยปกติแล้วผู้พัฒนาเกมจะกำหนดค่าตัวแปรนี้ให้คอมไพล์เลอร์สร้างให้โดยอัตโนมัติ

ตารางที่ 2.2 ค่าตัวบ่งชี้ Cooperative level ของ DirectDraw

ค่าตัวบ่งชี้ของ dwFlag	ความหมาย
DDSCL_ALLOWMODEX	อนุญาตให้แสดงภาพในโหมด X ได้ (320x200, 320x240) โดยค่านี้จะต้องใช้คู่กับ DDSCL_EXCLUSIVE และ DDSCL_FULLSCREEN
DDSCL_ALLOWREBOOT	อนุญาตให้ใช้ปุ่ม Ctrl+Alt+Delete ในขณะที่อยู่ในโหมด Exclusive (เต็มหน้าจอ) เพื่อ Reboot
DDSCL_EXCLUSIVE	กำหนดให้โปรแกรมมีลำดับความสำคัญสูงสุด โดยจะต้องทำงานคู่กับ DDSCL_FULLSCREEN
DDSCL_FULLSCREEN	อนุญาตให้เขียนข้อมูลลงบนหน่วยความจำแสดงผลทั้งหมด
DDSCL_NORMAL	บอกให้ระบบปฏิบัติการวินโดวส์ทราบว่าโปรแกรมนี้เป็นเหมือนโปรแกรมประยุกต์ทั่ว ๆ ไป โดยผู้พัฒนาเกมจะไม่สามารถกำหนดค่า DDSCL_ALLOWMODEX, DDSCL_EXCLUSIVE หรือ DDSCL_FULLSCREEN ได้
DDSCL_NOWINDOWCHANGES	บอกว่าโปรแกรมนี้ไม่สามารถย่อหรือขยายขนาดวินโดว์ของโปรแกรมอื่น ๆ ในขณะที่โปรแกรมนี้ทำงานอยู่ได้

ตัวอย่าง

```
lpdd->SetCooperativeLevel(hwnd, DDSCL_MODEX|
                                DDSCL_FULLSCREEN|
                                DDSCL_EXCLUSIVE|
                                DDSCL_ALLOWREBOOT);
```

โปรแกรมนี้เป็นแบบโหมด X แบบเต็มหน้าจอ อีกทั้งยังมีความลำดับสำคัญสูงสุด และยังสามารถออกจากโปรแกรมได้โดยใช้ปุ่ม Ctrl+Alt+Delete อีกด้วย

2.4.4 การกำหนดโหมดการแสดงผลของวินโดว์ด้วยฟังก์ชัน SetDisplayMode ของ DirectDraw

เป็นฟังก์ชันที่ใช้เปลี่ยนโหมดการแสดงผล ฟังก์ชันนี้จะแจ้งผลผ่านค่า HRESULT ถ้าสำเร็จจะส่งค่า DD_OK หรือ DirectDraw Okay รูปแบบของฟังก์ชันเป็นดังนี้

```
HRESULT SetDisplayMode(DWORD   dwWidth,
                       DWORD   dwHeight,
                       DWORD   dwBPP,
                       DWORD   dwRefreshRate,
                       DWORD   dwFlags);
```

การกำหนดพารามิเตอร์เป็นดังนี้

พารามิเตอร์ dwWidth กำหนดความกว้างของการแสดงผลโดยมีหน่วยเป็นพิกเซล

พารามิเตอร์ dwHeight กำหนดความยาวของการแสดงผลโดยมีหน่วยเป็นพิกเซล

พารามิเตอร์ dwBPP กำหนดจำนวนข้อมูลสีต่อหนึ่งพิกเซล (8,16,32 bit)

ส่วนพารามิเตอร์ dwRefreshRate และ dwFlags ให้กำหนดเป็น 0 ทั้งหมดเนื่องจากคอมไพล์เลอร์จะทำการตรวจสอบค่าที่เหมาะสมให้เอง

การใช้งานฟังก์ชันของ Direct Draw มีสามขั้นตอนดังนี้

1.) สร้างอ็อปเจกต์ DirectDraw

2.) กำหนดระดับความสำคัญและจัดสรรทรัพยากรโดยใช้ฟังก์ชัน SetCoopeeativeLevel

3.) ทำการกำหนดขนาดของการแสดงผลโดยใช้ฟังก์ชัน SetDisplayMode

ตัวอย่าง

```
LPDIRECTDRAW        lpdd;
if (DirectDrawCreate(NULL, &lpdd, NULL) != DD_OK)
{//ERROR//}
lpdd->SetCooperativeLevel(hwnd, DDSCL_ALLOWMODEX|
                                DDSCL_FULLSCREEN|
                                DDSCL_EXCLUSIVE|
                                DDSCL_ALLOWREBOOT);
if ((lpdd->SetDisplayMode(640, 480, 8)) != DD_OK)
{//ERROR}
```

2.4.5 การวาดภาพลงบนวินโดว์ด้วยคำสั่ง DirectDraw

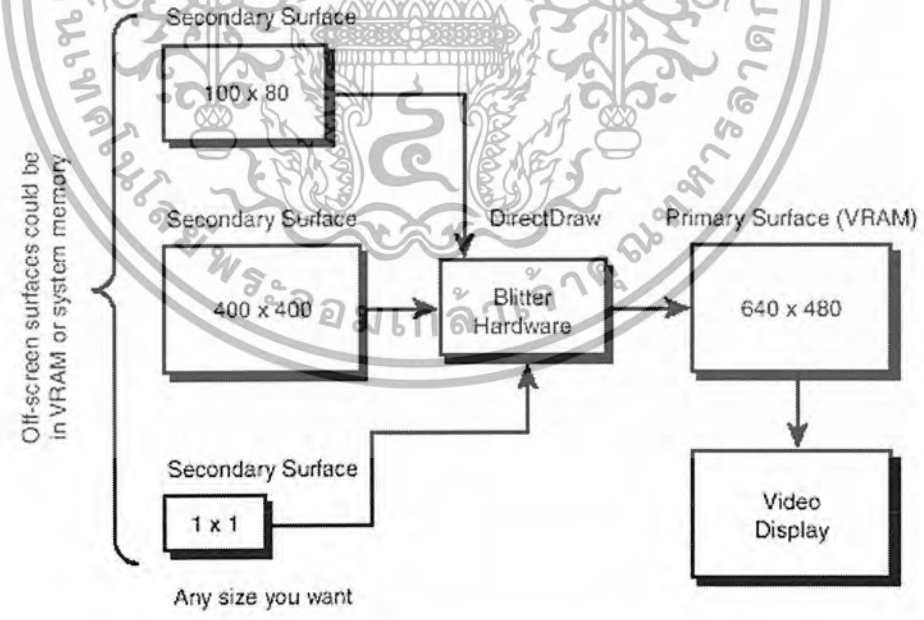
DirectDraw มี Surface เป็นหน่วยความจำที่เก็บรูปภาพที่จะแสดงผลไว้ใน Primary Surface ซึ่งเป็น Surface ที่เรามองเห็นขณะนั้นบนจอภาพ และเราสามารถกำหนด Surface ได้มาก 1 Surface แต่ต้องขึ้นอยู่กับหน่วยความจำที่มีอยู่ในระบบ

2.4.6 หลักการสร้าง Surface ของ DirectDraw

1.) Surface ที่สร้างนั้นมีขนาดเท่าใดก็ได้ขึ้นอยู่กับภาพที่เราจะนำมาแสดง แต่ Surface ที่เป็น Primary Surface ต้องมีขนาดเท่ากับขนาดของความละเอียดของจอภาพขณะนั้น

2.) การสร้าง Surface ผู้พัฒนาเกมควรจะเลือกใช้หน่วยความจำของการ์ดแสดงผล (VRAM) หรือหน่วยความจำหลัก (System Memory) ซึ่งถ้าใช้ VRAM ได้จะทำให้การแสดงผลมีความเร็วสูงขึ้น แต่ VRAM มีขนาดเล็ก ดังนั้นผู้พัฒนาเกมสามารถใช้ System Memory แทนได้

3.) เมื่อสร้าง Surface ขึ้นมา ระดับสีใน Surface ต้องเข้ากันได้กับระดับสีของ Primary Surface ตามปกติ Surface ที่ขึ้นสร้างมาจากอ็อบเจกต์ของ Direct Draw ตัวเดียวกันจะมีระดับสีเป็นระดับเดียวกันหมด แต่ถ้ามีอ็อบเจกต์ของ Direct Draw สองตัว อาจเป็นไปได้ที่ระดับสีของ Surface ไม่เท่ากัน ทำให้ไม่สามารถทำการถ่ายโอนข้อมูลระหว่าง Surface ได้



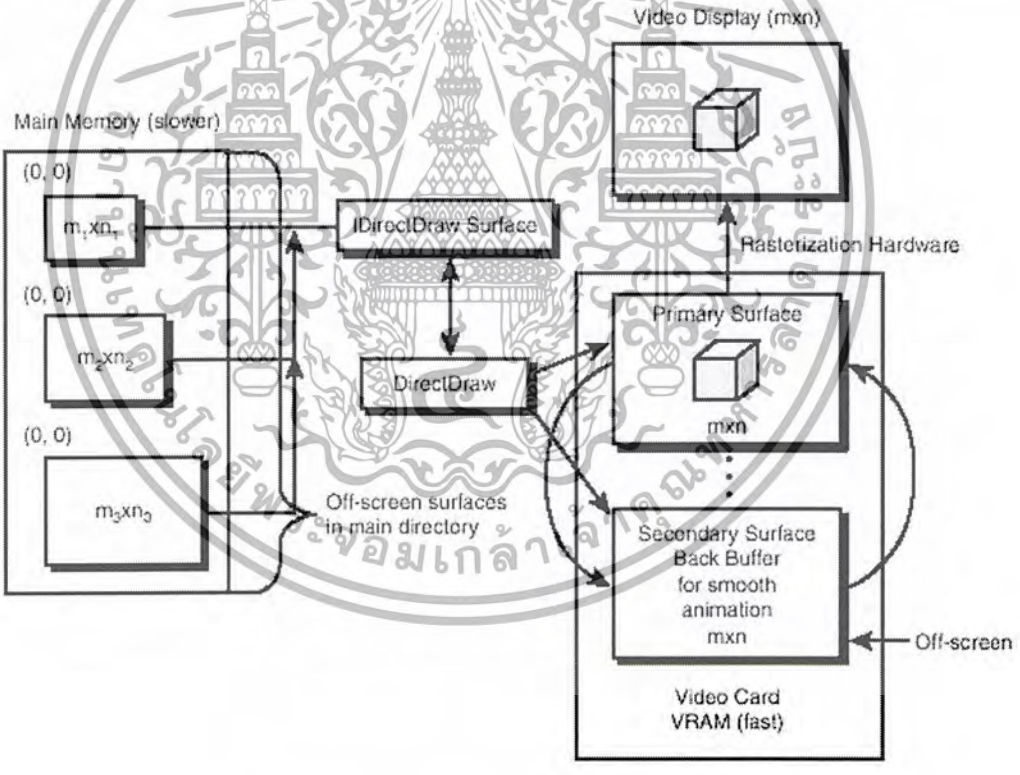
รูปที่ 2.7 การกำหนดขนาดของ Surface

โดยทั่วไปเกมที่สร้างขึ้นจะใช้ Surface ดังนี้

1.) Primary Surface หรือ FrontBuffer ทำหน้าที่แสดงภาพที่เป็นหน้าจอปัจจุบันโดยมีโครงสร้างและขนาดเท่ากับความละเอียดของหน้าจอ

2.) Secondary Surface หรือ BackBuffer ทำหน้าที่วาดภาพหน้าจอต่อไปขณะที่ Primary Surface โดย Surface นี้มีโครงสร้างทุกอย่างเหมือน Primary Surface เพื่อที่จะทำงานสลับกับ Primary Surface

3.) OffScreen Surface เป็น Surface สำหรับเก็บภาพต่าง ๆ ที่ใช้ในโปรแกรม เพื่อที่จะเพิ่มประสิทธิภาพในการแสดงผลโดยไม่ต้องไปอ่านภาพจากไฟล์รูปที่อยู่ในในฮาร์ดดิสก์ ทำให้เข้าถึงได้เร็วกว่า ส่วนมากแล้วเกมจะมีจำนวน Surface นี้มากที่สุด เพื่อใช้เก็บรูปภาพต่าง ๆ ลักษณะของการแสดงภาพจะเป็นวัฏจักรดังนี้



รูปที่ 2.8 การแสดงภาพบนหน้าจอโดย DirectDraw

ขั้นแรกนำภาพที่เราต้องการใช้งานเก็บไว้ใน OffScreen Surface โดยอาจมีได้หลาย Offscreen Surface จากนั้นทำการ Blit (Bit Block Transfer) โดยการถ่ายโอนข้อมูลจาก

Offscreen ไปยัง Secondary Surface ณ ตำแหน่งที่ต้องการแสดงผลโดยอ้างอิงจากจอภาพ จากนั้นก็แสดงผลต่อผู้ใช้ด้วยวิธี FLIP หรือการสลับหน้าจอกับ Primary Surface โดยจะทำการสลับเฉพาะตำแหน่งหน่วยความจำที่อ้างอิง จากนั้นทำการวาดภาพที่จะใช้แสดงผลต่อไปใน Secondary Surface ในตำแหน่งที่เพิ่งสลับมาและจะเป็นเช่นนี้ไปเรื่อย ๆ จนจบโปรแกรม

การสร้าง Surface จะต้องทำการใส่ข้อมูลลงในโครงสร้างข้อมูลแบบ DDSURFACEDESC (Direct Draw Surface Description) แล้วจึงทำการเรียกฟังก์ชัน CreateSurface เพื่อทำการสร้าง Surface นั้น ๆ ถ้าฟังก์ชันนี้ทำงานสำเร็จจะคืนค่า DD_OK กลับมา โดยรูปแบบฟังก์ชันเป็นดังนี้

```
HRESULT CreateSurface(LPDDSURFACEDESC       lpDDSurfaceDesc,
                      LPDIRECTDRAWSURFACE   FAR *lplpDDSurface,
                      lunknown              FAR *pUnkOuter);
```

พารามิเตอร์ lpDDSurfaceDesc เป็นข้อมูลประเภทตัวชี้ชี้ไปยังโครงสร้างข้อมูลแบบ DDSURFACEDESC ซึ่งมีการใส่ข้อมูลลงไปแล้ว

พารามิเตอร์ lplpDDSurface ถ้าฟังก์ชันทำการสร้าง Surface ได้สำเร็จ จะคืนค่านี้กลับมาเป็นตัวชี้ที่ชี้ไปยัง Surface ที่สร้างขึ้น

พารามิเตอร์ pUnkOuter เป็นพารามิเตอร์ขั้นสูง โดยปกติจะกำหนดให้เป็น ค่าว่าง

DDSURFACEDESC มีรูปแบบของโครงสร้างดังนี้

```
typedef struct _DDSURFACEDESC
{
    DWORD           dwSize;
    DWORD           dwFlags;
    DWORD           dwHeight;
    DWORD           dwWidth;
    Union
    {
        LONG        lPitch;
        DWORD       dwLinearSize;
    };
    DWORD dwBackBufferCount;
    union
    {
```

```
        DWORD           dwMipMapCount;
        DWORD           dwZBufferBitDepth;
        DWORD           dwRefreshRate;
    };
    DWORD           dwAlphaBitDepth;
    DWORD           dwReserved;
    LPVOIDlpSurface;
    DDCOLORKEY      ddckCKDestOverlay;
    DDCOLORKEY      ddckCKDestblt;
    DDCOLORKEY      ddckCKSrcOverlay;
    DDCOLORKEY      ddckCKSrcBlt;
    DDPIXELFORMAT   ddpfPixelFormat;
    DDSCAPS         ddsCaps;
} DDSURFACEDESC;
```

ค่าพารามิเตอร์ที่ต้องกำหนด คือ

- พารามิเตอร์ dwSize คือ ค่าขนาดของโครงสร้าง DDSURFACEDESC
- พารามิเตอร์ dwFlags คือ ส่วนที่ใช้กำหนดค่าให้กับ CreateSurface ทราบว่าจะสร้าง Surface แบบใด ซึ่งมีหลายค่าดังนี้

ค่าตัวบ่งชี้ DDSD_ALL เป็นตัวกำหนดให้ผู้พัฒนาเกมสามารถเรียกใช้การทำงานของได้ทุก ๆ ส่วนของ Surface คือ DDSD_ALPHABITDEPTH, DDSD_BACKBUFFERCOUNT, DDSD_CAPS, DDSD_CKDESTBLT, DDSD_CKDESTOVERLAY, DDSD_CKSRCBLT, DDSD_ CKSRCOVERLAY, DDSD_HEIGHT, DDSD_LINEARSIZE, DDSD_LPSURFACE, DDSD_MIPMAPCOUNT, DDSD_PITCH, DDSD_PIXELFORMAT, DDSD_REFRESHRATE, DDSD_WIDTH,DDSD_ZBUFFERBITDEPTH

- dwBackBufferCount คือ ส่วนที่ใช้กำหนดจำนวน BackBuffer โดยทั่วไปมีเพียง 1 Surface
- ddsCaps คือ ส่วนที่ใช้กำหนดคุณสมบัติต่าง ๆ ของ Surface โดยมีโครงสร้างดังนี้

```
typedef struct _DDSCAPS
{
        DWORD dwCaps;
} DDSCAPS.FAR* LPDDSCAPS;
```

- dwcaps เป็นส่วนที่ใช้เก็บค่าคุณสมบัติของ Surface โดยส่วนใหญ่จะใช้ในระดับสูง

ตารางที่ 2.3 ค่าตัวบ่งชี้ของ dwCaps

ค่าตัวบ่งชี้ของ dwCaps	ความหมาย
DDSCAPS_BACKBUFFER	Surface ถูกกำหนดให้เป็น BackBuffer
DDSCAPS_COMPLEX	บอก DirectDrawSurface ว่าเราจะใช้หลาย Surface
DDSCAPS_FLIP	บอกว่า Surface นี้สามารถ Filp ได้
DDSCAPS_FRONTBUFFER	Surface ถูกกำหนดให้เป็น FrontBuffer
DDSCAPS_MODEX	กำหนดให้ Surface เป็น ModeX หรือ 320x200 หรือ 320x240 พิกเซล
DDSCAPS_OFFSCREENPLAIN	กำหนดให้ Surface เป็น Offscreen ซึ่งไม่ใช้ Frontbuffer หรือ Backbuffer โดยปกติจะใช้สำหรับเก็บภาพเคลื่อนไหวต่าง ๆ
DDSCAPS_SYSTEMMEMORY	กำหนดให้ Surface นี้จองหน่วยความจำในหน่วยความจำหลัก

ตัวอย่างการสร้าง PrimarySurface และ SecondarySurface

```
LPDIRECTDRAW            lpdd; // ประกาศตัวแปรตัวชี้ของ DirectDraw
DDSURFACEDESC           ddsd; //ประกาศตัวแปรที่ใช้เก็บค่าของ
                        //โครงสร้างข้อมูล DDSURFACEDESC
LPDIRECTDRAWSURFACE     lpddsprimary; //เก็บ address ของ Surface
//ทำการสร้างอ็อปเจกต์ของ DirectDraw
DirectDrawCreate(NULL, &lpdd, NULL);
//กำหนดระดับความสำคัญและจัดสรรทรัพยากร
```

```
lpdd->SetCooperatiiveLevel(hwnd,
                           DDSCL_ALLOWREBOOT|
                           DDSCL_ALLOWMODEX|
                           DDSCL_FULLSCREEN|
                           DDSCL_EXCLUSIVE);
//กำหนดขนาดของการแสดงผล
lpdd->SetDisplayMode(SCREEN_WIDTH, SCREEN_HEIGHT, SCREEN_BPP);
//กำหนดขนาดของ Primary Surface
ddsd.dwSize = sizeof(ddsd);
//หลังจากนั้นจึงกำหนดตัวบ่งชี้ของ Surface
ddsd.dwFlags = DDSD_CAPS| DDSD_BACKBUFFERCOUNT);
//เมื่อ DDSD_CAPS สามารถเรียกใช้งานได้นั้นหมายถึงสามารถกำหนด Primary Surface ได้แล้ว
ddsd.dwCaps.dwCaps = DDSCAPS_PRIMARYSURFACE|
                     DDSCAPS_FLIP|
                     DDSCAPS_COMPLEX;
ddsd.dwBackBufferCount = 1; //มี BackBuffer 1 อัน
lpdd->CreateSurface(&ddsd, &lpddsprimary, NULL); //สร้าง Primary Surface
ddscaps.dwCaps = DDSCAPS_BACKBUFFER; //ทำการกำหนด BackBuffer
//กำหนดให้ lpddsback เป็น Backbuffer ซึ่งใช้ทำงานร่วมกับ lpddsprimary
lpddsprimary->GetAttachedSurface(&ddscaps, &lpddsback);
```

2.4.7 **การแสดงภาพบนหน้าจอวินโดว์**

เมื่อเราทำการวาดภาพลงบน Backbuffer เรียบร้อยแล้ว หลังจากนั้นทำการสลับภาพระหว่าง Frontbuffer กับ PrimarySurface โดยใช้ฟังก์ชัน Flip

```
HRESULT Flip(LPDIRECTDRAWSURFACE lpDDSurfaceOverride, DWORD dwFlags);
```

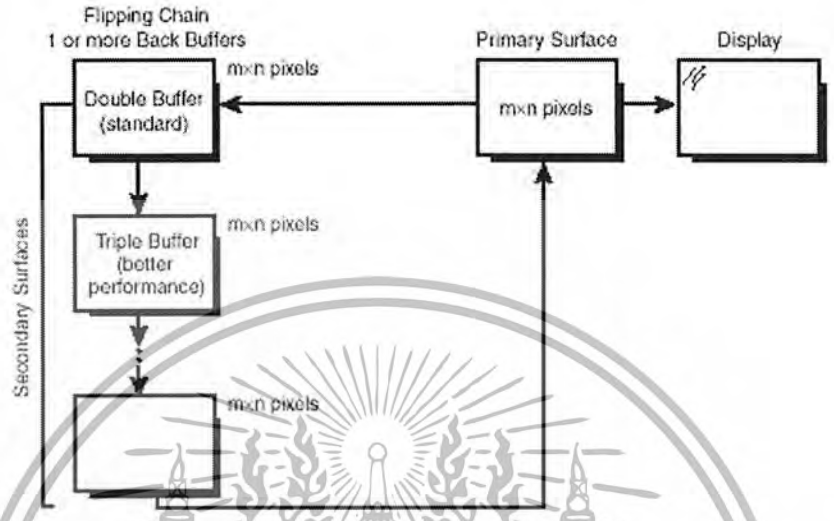
ค่าพารามิเตอร์ที่กำหนด คือ

พารามิเตอร์ lpDDSurfaceOverride โดยปกติกำหนดให้เป็นค่าว่าง

พารามิเตอร์ dwFlags โดยปกติกำหนดให้เป็น DDFLIP_WAIT เนื่องจากบางครั้ง SecondarySurface ยังอยู่ในสถานะที่กำลังวาดอยู่ และยังไม่สามารถสลับได้ โดยค่านี้จะทำให้ฟังก์ชันทำการลองสลับ Surface ไปจนกว่าจะสลับได้

วิธีใช้เป็นดังนี้

```
lpddsprimary->Flip(NULL, DDFLIP_WAIT);
```



รูปที่ 2.9 การสลับ Surface

2.4.8 การวาดภาพลงบน Surface และการปรับสีพื้นหลังให้โปร่งใส

ก่อนที่จะทำการสลับ Surface ผู้พัฒนาเกมจะต้องทำการวาดลงบน Surface ให้ได้ก่อน โดยใช้ฟังก์ชัน Blt ในการคัดลอกข้อมูล โดยเราสามารถคัดลอกข้อมูลจาก Surface หนึ่ง ๆ ไป Surface ใดก็ได้ โดยฟังก์ชันมีโครงสร้างดังนี้

```
HRESULT Blt(LPRECT               lpDestRect,
            LPDIRECTDRAWSURFACE  lpDDSrcSurf,
            DWORD                dwFlags,
            LPDDBLTFX            lpDDBltFx);
```

lpDestRect คือ ตัวชี้ที่ชี้ไปยังตัวแปรประเภท RECT ของ Surface ปลายทาง ซึ่งตัวแปรประเภท RECT นี้มีโครงสร้างดังนี้

```
typedef struct _RECT
{
    LONG  top; //ตำแหน่งพิกเซลที่อยู่บนสุด
    LONG  left; //ตำแหน่งพิกเซลซ้ายสุด
    LONG  right; //ตำแหน่งพิกเซลขวาสุด
```

LONG bottom; //ตำแหน่งพิกเซลล่างสุด

} RECT

พารามิเตอร์ lpDestRect คือ ตำแหน่งที่จะวางรูปภาพนั่นเอง แต่หากค่านี้กำหนดเป็นค่าว่าง จะหมายถึงใช้ Surface ปลายทางทั้งหมด

lpDDSrcSurf คือ Surface ที่เป็นรูปภาพต้นทาง ซึ่งอาจเป็น OffScreen ก็ได้

lpSrcRect คือ ตำแหน่งของรูปภาพที่ต้นทาง ถ้ากำหนดให้เป็นค่าว่าง หมายถึงใช้รูปทั้งรูป

dwFlags คือ ส่วนที่ใช้ควบคุมการวาดภาพ

lpDDBltFx คือ ตัวชี้ที่ใช้ชี้ไปที่โครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนดความสามารถพิเศษต่าง ๆ ของฟังก์ชัน เช่น การหมุนภาพ หรือระบายสี

ตารางที่ 2.4 ค่าตัวบ่งชี้ควบคุมของฟังก์ชัน Blt

ค่าตัวบ่งชี้ของ dwFlags	ความหมาย
DDBLT_COLORFILL	ใช้ร่วมกับ dwFillColor ในโครงสร้างข้อมูล DDBLTFX โดยใส่เป็นสีรูปแบบ RGB (Red, Green, Blue) ซึ่งจะทำการระบายลงบน RECT ที่ Surface ปลาย ทาง
DDBLT_DDFX	ใช้ร่วมกับ dwDDFX ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนด ลักษณะพิเศษบางอย่างในการแสดงผล
DDBLT_DDROPS	ใช้ร่วมกับ dwDDROPS ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX โดยใช้กับ การทำงานชั้นสูง
DDBLT_KEYDEST	เมื่อกำหนดแล้วจะสามารถกำหนด Color Key ที่ Surface ปลายทางได้
DDBLT_KEYSRC	เมื่อกำหนดแล้วจะสามารถกำหนด Color Key ที่ Surface ต้นทางได้
DDBLT_ROP	ใช้ร่วมกับ dwROP โดยกำหนดในโครงสร้างข้อมูล DDBLTFX โดยทำงานตรง กันข้ามกับ DDBLT_DDROPS
DDBLT_ROTATIONANGLE	ใช้ร่วมกับ dwRotationAngle ในโครงสร้างข้อมูล DDBLTFX โดยใช้หมุนภาพที่จะ Blit ไปใส่ที่ Surface โดยมีหน่วยเป็น 1/100 องศา
DDBLT_WAIT	สั่งทำการ Blit จนกว่าจะ Blit สำเร็จ เมื่อใช้คำสั่งนี้ฟังก์ชัน Blit จะไม่ส่งข้อผิดพลาดกลับมาถึงแม้จะยังทำการ Blit ไม่ได้ก็ตาม

ในโครงสร้างข้อมูล DDBLTFX มีข้อมูลจำนวนมากที่ต้องกำหนด แต่ข้อมูลส่วนใหญ่จะใช้สำหรับการทำงานระดับสูง เช่น Z-buffer ที่ใช้กับภาพสามมิติ ดังนั้นเราจะกล่าวถึงเพียงเท่าที่จำเป็นเท่านั้น

```
typedef struct _DDBLTFX
{
        DWORD       dwSize;
        DWORD       dwDDFX;
        DWORD       dwROP;
        DWORD       dwDDROP
        DWORD       dwRotationAngle;
        DWORD       dwZBufferOpCode;
        DWORD       dwZbufferLow;
        DWORD       dwZbufferHigh;
        DWORD       dwZBufferBaseDest;
        DWORD       dwZDestConstBitDepth;
union {
        DWORD                   dwZDestConst;
        LPDIRECTDRAWSURFACE     lpDDSZBufferDest;
};
DWORD dwZSrcConstBitDepth;
union {
        DWORD                   dwzSrcConst;
        LPDIRECTDRAWSURFACE     lpDDSZBufferSrc;
};
DWORD       dwAlphaEdgeBlendBitDepth;
DWORD       dwAlphaEdgeBlend;
DWORD       dwReserved;
DWORD       dwAlphaDestConstBitDepth;
union
{
        DWORD                   dwAlphaDestConst;
```

```
        LPDIRECTDRAWSURFACE     lpDDSAlphaDest;
};
DWORD dwAlphaSrcConstBitDepth;
union {
        DWORD                   dwAlphaSrcConst;
        LPDIRECTDRAWSURFACE     lpDDSAlphaSrc;
};
union {
        DWORD                   dwFillColor;
        DWORD                   dwFillDepth;
        DWORD                   dwFillPixel;
        LPDIRECTDRAWSURFACE     lpDDSPattern;
};
DDCOLORKEY      ddckDestColorkey;
DDCOLORKEY      ddckSrcColorkey;
} DDBLTFX,FAR*  LPDDBLTFX;
```

พารามิเตอร์ที่กำหนดค่า คือ

พารามิเตอร์ dwSize	เป็นขนาดของโครงสร้างข้อมูล DDBLTFX มีค่าเป็นไบต์
พารามิเตอร์ dwFillColor	สีที่ต้องการระบายลง Surface
พารามิเตอร์ ddckDestColorkey	เป็น Color Key ของทาง Surface ปลายทาง
พารามิเตอร์ ddckSrcColorkey	เป็น Color Key ของทาง Surface ต้นทาง
พารามิเตอร์ dwRotationAngle	กำหนดเป็นองศาเพื่อจะใช้หมุนภาพ

Color Key คือ สีที่เรากำหนดให้เป็นสีโปร่งใส ซึ่งจะใช้มากในเกมคอมพิวเตอร์ทั่ว ๆ ไป โดยปกติรูปภาพที่สร้างขึ้นจะมีขอบสี่เหลี่ยมที่เป็นส่วนเกิน เมื่อนำภาพไปสร้างเกมเราต้องตัดส่วนนี้ออกโดยการกำหนดสีที่เป็น Color Key แล้วเราก็ใช้สีเดียวกันนี้ระบายลงไปที่ขอบหรือส่วนเกินของภาพ หลังจากทำการ Blit แล้ว คอมไพเลอร์จะทำการจับคู่สีที่เป็น Color Key แล้วจะไม่แสดงสีนั้นลงบนจอภาพ ดังรูป

รูปที่ 2.10 การเปรียบเทียบระหว่างการกำหนดค่า Color Key

จากรูปที่ 2.10 รูปเดิมจะเป็นรูปด้านซ้ายมีขอบเกินมาด้วย แต่เมื่อกำหนดให้ Color Key เป็นสีดำแล้ว เมื่อเราทำการ Blit บน Surface เราจะได้ภาพด้านขวาที่ไม่มีขอบเกินออกมา ทำให้ภาพดูสมจริงมากขึ้น

ตัวอย่าง การ Blit โดยใช้ Color Key เป็นสีดำ

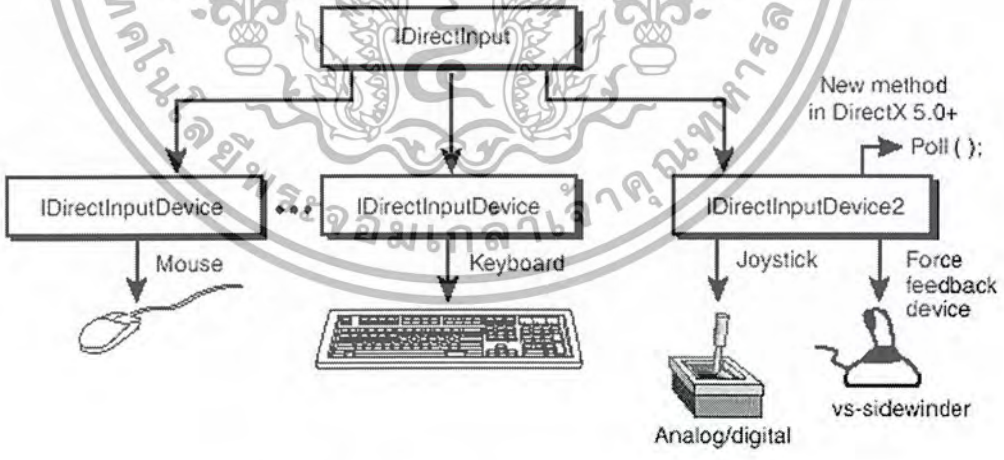
```
DDBLTFX        ddbltfx; //ข้อมูลประกาศโครงสร้าง
RECT           blit_area, src_area; //ประกาศ RECT ที่จะวางภาพปลายทางและภาพต้นทาง
memset(&ddbltfx, 0, sizeof(DDBLTFX));
ddbltfx.dwSize = sizeof(DDBLTFX);
DDCOLORKEY     col_key; //กำหนดตัวแปรแบบ DDCOLORKEY
col _key.dwColorSpaceLowValue = 0; //ทำการเซตค่าสีหนึ่งซึ่งต้องมีค่าสีต่ำกว่า
col_key.dwColorSpaceHighValue = 0; //ไปยังอีกสีหนึ่ง ทำให้เราสามารถมี
                                   transparent ได้หลายสี
lpddsback->SetColorKey(DDCKEY_SRCBLT, &col_key); //ทำการเซต Color Key โดยเป็น
                                   //Source Colorkey ซึ่งก็คือสีที่ไม่ต้องการ ณ ภาพต้นทาง
                                   //โดยต้องทำการกำหนดที่ BackBuffer
                                   //เพื่อบอกให้ BackBuffer ทราบว่าจะไม่สนใจสีนี้
                                   //และไม่ทำการ Blit ลงไป แต่หากเป็น DestinationColorkey
                                   //จะหมายถึงว่าเป็นสีที่จะไม่ให้วาดทับแทน
blit_area.top = top;
blit_area.left = left;
blit_area.bottom = bottom;
blit_area.right = right;
src_area.top=top_src;
```

```
src_area.left=left_src;
src_area.bottom=bottom_src;
src_area.right=right_src;
lpddsprimary->Blt(&blit_area, lpddsback,,&src_area, DDBLT_KEYSRC | DDBLT_WAIT,
NULL);
```

2.5 วิธีการรับข้อมูลจากแป้นพิมพ์

DirectInput เป็นไลบรารีที่ช่วยให้ผู้พัฒนาเกมทำการติดต่อและเรียกใช้อุปกรณ์ในการรับข้อมูลเข้าชนิดต่าง ๆ ได้อย่างมีประสิทธิภาพ Direct Input ครอบคลุมอุปกรณ์ในการรับข้อมูลเข้ามาตรฐานทุกชนิด อันประกอบไปด้วย

- แป้นพิมพ์มาตรฐาน
- เมาส์ทั้ง 2 และ 3 ปุ่ม
- ก้านควบคุม, จอยโยก (Paddle) ทั้งที่เป็นแบบแอนะล็อกและดิจิทัล
- อุปกรณ์ควบคุมการขับเคลื่อนใช้สำหรับจำลองการขับรถแบบแอนะล็อกและดิจิทัล
- อุปกรณ์ที่ตอบสนองแรงสั่นสะเทือน, ก้านควบคุมหรืออุปกรณ์อื่น ๆ ที่สามารถคำนวณเพื่อทำให้เกิดแรงสั่นโต้ตอบกับผู้ใช้



รูปที่ 2.11 อุปกรณ์รับข้อมูลเข้าต่าง ๆ

2.5.1 การใช้ DirectInput

Direct Input ประกอบด้วยไฟล์เฮดเดอร์ dinput.h และไฟล์ไลบรารี dinput.lib สำหรับโปรแกรมที่ต้องการติดต่อกับอุปกรณ์อินพุต

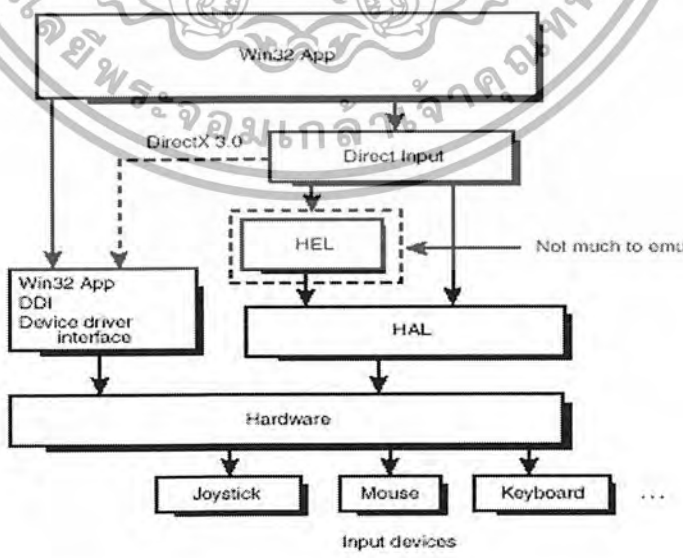
ขั้นตอนในการใช้ Direct Input

- สร้างอ็อปเจกต์ของ Direct Input หลักด้วยฟังก์ชัน DirectInputCreate
- สร้างอ็อปเจกต์ของอุปกรณ์รับข้อมูลเข้าด้วยฟังก์ชัน CreateDevice
- กำหนดลักษณะการใช้งานของแต่ละอุปกรณ์ด้วยฟังก์ชัน SetCooperativeLevel
- กำหนดรูปแบบการรับข้อมูลของแต่ละอุปกรณ์ด้วยฟังก์ชัน SetDataFormat และทำการกำหนดคุณสมบัติพิเศษของอุปกรณ์ต่าง ๆ ด้วยฟังก์ชัน SetProperty
- ทำการจองอุปกรณ์ต่าง ๆ ด้วยฟังก์ชัน Acquire
- รับข้อมูลจากแต่ละอุปกรณ์ที่ได้ทำการจองไว้แล้วด้วยฟังก์ชัน GetDeviceState

การอ่านข้อมูลของแต่ละอุปกรณ์จะถูกอ่านเป็นระเบียน โดยอุปกรณ์แต่ละตัวมีรูปแบบของระเบียนที่แตกต่างกันแต่มีลักษณะใกล้เคียงกันมาก ทำให้เกิดการสร้างข้อมูลเข้าที่เป็นรูปแบบหลัก ๆ ขึ้น โดยผู้พัฒนาเกมสามารถร้องขอข้อมูลเข้าจาก DirectInput ได้ 2 วิธี ดังนี้

1.) แบบทันทีทันใด ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของอุปกรณ์รับข้อมูลเข้าขณะนั้น

2.) แบบมีการพักข้อมูล ข้อมูลถูกเก็บเป็นฐานข้อมูลของเหตุการณ์ที่เกิดขึ้นตั้งแต่การร้องขอข้อมูลเข้าครั้งสุดท้าย



รูปที่ 2.12 ระดับโครงสร้างภายในของ DirectInput

2.5.2 การติดต่อกับแป้นพิมพ์ผ่านทาง Direct Input

2.5.2.1 การสร้างอ็อปเจกต์ของ DirectInput

การกำหนดอ็อปเจกต์ของ Direct Input มีจุดประสงค์เพื่อใช้ในการจัดการติดต่อกับอุปกรณ์รับข้อมูลเข้าชนิดอื่น

```
LPDIRECTINPUT lpdi;
```

การสร้างอ็อปเจกต์หลักของอุปกรณ์อินพุตทำได้โดยใช้ฟังก์ชัน DirectInputCreate โดยมีรูปแบบการของฟังก์ชันดังนี้

```
HRESULT DirectInputCreate(HINSTANCE      hinst,
                          DWORD          dwversion,
                          LPDIRECTINPUT  * lplpDirectInput,
                          LPUNKNOWN      punkOuter);
```

ฟังก์ชันจะคืนค่า DI_OK(DirectInput Okey) ถ้าการเรียกใช้ฟังก์ชันนี้ผ่าน มันจะคืนค่าเป็นค่าผิดพลาดอื่น ๆ ถ้าฟังก์ชันนี้ไม่ผ่าน สำหรับค่าพารามิเตอร์ต่าง ๆ ของ DirectInputCreate มีดังนี้

hinst	ใช้อ้างถึงโปรแกรมประยุกต์ที่ต้องการใช้งาน Direct Draw ให้แน่ใจว่าค่า hinst นี้ ถูกส่งค่ามาจากค่า hinstance ใน WinMain
dwversion	กำหนดเวอร์ชันของ Direct Input ว่าจะใช้เวอร์ชันใดมีไว้เพื่อให้ผู้พัฒนาเกมที่ต้องการใช้ Direct Input เวอร์ชันเก่าแต่โดยทั่วไปจะถูกกำหนดค่าเป็น DIRECTINPUT_VERSION ซึ่ง DirectInput ที่กำหนดจะใช้เป็นเวอร์ชันล่าสุด
lplpDirectInput	เป็นตัวชี้ที่อ้างอิงไปยังอ็อปเจกต์หลักของ DirectInput
punkOuter	ถูกกำหนดเป็นค่าว่างเสมอ

ตัวอย่างการสร้างอ็อปเจกต์หลักของ DirectInput

```
LPDIRECTINPUT lpdi;
DirectInputCreate(hinstance, DIRECTINPUT_VERSION, &lpdi, NULL);
```

เมื่อไม่ต้องการใช้งานอ็อปเจกต์หลักของ DirectInput ก็ทำการลบด้วยฟังก์ชัน Release เช่น

```
lpdi->Release();
```

2.5.2.2 การสร้างอ็อปเจกต์ของแป้นพิมพ์

ด้วยฟังก์ชัน CreateDevice ฟังก์ชันนี้จะใช้เมื่อต้องการสร้างอ็อปเจกต์ของอุปกรณ์รับข้อมูลเข้าใด ๆ รูปแบบการใช้งานฟังก์ชัน CreateDevice มีดังนี้

```
HRESULT CreateDevice( REFGUI                    rguid,
                      LPDIRECTINPUTDEVICE       * lplpDIDev,
                      LPUNKNOWN                 punkOuter);
```

ค่าพารามิเตอร์ต่าง ๆ ของ DirectInput มีดังนี้

rguid คือ ค่า GUID(Globally Unique Identifier) เป็นหมายเลขของอุปกรณ์รับข้อมูลเข้าที่ต้องการสร้าง โดยปกติจะใช้ฟังก์ชัน EnumDevice เพื่อหาค่า GUID ของอุปกรณ์รับข้อมูลเข้านั้น ๆ แต่ใน DirectInput มีการ กำหนดค่า GUID ของแป้นพิมพ์ไว้ให้เรียบร้อยแล้ว โดยจะมีค่าเป็น

GUID_SysKeyboard สำหรับแป้นพิมพ์มาตรฐาน

อย่างไรก็ตามการใช้ค่า GUID ที่ถูกประกาศไว้แล้วนี้จำเป็นต้องเพิ่ม #define INITGUID ไว้ส่วนบนของทุกไฟล์ซีพลัสพลัส ที่มีการประกาศ OBJBASE.H เพื่อคอมไพเลอร์จะรวมข้อมูลเกี่ยวกับCOM และค่า GUID ที่ได้ถูกประกาศไว้แล้ว

lplpDiDev เป็นตำแหน่งที่ตัวชี้ชี้เพื่ออ้างอิงถึงอุปกรณ์รับข้อมูลเข้าชนิดนั้น ๆ

punkOuter ถูกกำหนดให้เท่ากับค่าว่างเสมอ

ตัวอย่างการสร้างอ็อปเจกต์ของแป้นพิมพ์

```
#define INITGUID
#include <OBJBASE.H>
#include "DINPUT.H"
```

//สร้างอ็อปเจกต์หลักตามที่ได้กล่าวมาในขั้นก่อนหน้านี้หลังจากสร้างอ็อปเจกต์หลักแล้วจึงทำการสร้าง

```
//อ็อปเจกต์ของแป้นพิมพ์
LPDRECTINPUTDEVICE      lpdikey;
lpdi->CreateDevice(GUID_SysKeyboard, &lpdikey, NULL);
```

2.5.2.3 การกำหนดลักษณะการใช้งานของแป้นพิมพ์

แต่ละอุปกรณ์ก็จะมีลักษณะการใช้งานเป็นของตัวเอง SetCooperativeLevel มีรูปแบบการเรียกใช้ดังนี้

```
HRESULT SetCooperativeLevel(HWND hwnd,
                            DWORD dwFlag);
```

ค่าพารามิเตอร์ต่าง ๆ ของ SetcooperativeLevel มีดังนี้

hwnd ใช้อ้างถึงแอดเดรสของวินโดว์ของโปรแกรมประยุกต์นั้น ๆ

dwFlag เป็นค่าที่กำหนดลักษณะของอุปกรณ์รับข้อมูลเข้านั้น ๆ มีทั้งหมด 4 ค่า

ตารางที่ 2.5 ค่าตัวบ่งชี้ dwFlag ลักษณะการใช้งานของแป้นพิมพ์

ค่าตัวบ่งชี้ dwFlag	ความหมาย
DISCL_BACKGROUND	กำหนดให้โปรแกรมประยุกต์สามารถใช้อุปกรณ์รับข้อมูลเข้านั้น ๆ ได้แม้ว่า โปรแกรมประยุกต์จะอยู่ในโหมด Background
DISCL_FOREGROUND	โปรแกรมประยุกต์ไม่สามารถเรียกใช้อุปกรณ์รับข้อมูลเข้านั้น ๆ ได้ในโหมด Background เพราะอุปกรณ์รับข้อมูลเข้าจะถูกปล่อยทันทีเมื่อโปรแกรมประยุกต์เปลี่ยนเป็นโหมด Background
DISCL_EXCLUSIVE	หลังจากทำการจองอุปกรณ์รับข้อมูลเข้านั้น ๆ แล้ว จะมีได้เพียงโปรแกรมประยุกต์เดียวที่ ร้องขอ DISCL_EXCLUSIVE ส่วนโปรแกรมประยุกต์อื่นจะต้องร้องขอการใช้งานอุปกรณ์อินพุทแบบ DISCL_NONEXCLUSIVE ได้
DISCL_NONEXCLUSIVE	โปรแกรมประยุกต์สามารถใช้งานอุปกรณ์รับข้อมูลเข้ารวมกันได้

สำหรับเกมที่ใช้โหมดแบบ Full Screen จะมีการกำหนดลักษณะของอุปกรณ์รับข้อมูลเข้าเป็นแบบ DISCL_BACKGROUND และ DISCL_NONEXCLUSIVE

ตัวอย่างการใช้ฟังก์ชัน SetCooperativeLevel

```
lpdikey->SetCooperativeLevel(hwnd, DISCL_BACKGROUND| DISCL_NONEXCLUSIVE);
```

2.5.2.4 การกำหนดรูปแบบการรับข้อมูลของแป้นพิมพ์

เพื่อกำหนดว่าข้อมูลที่จะรับเข้ามาจากแป้นพิมพ์มีรูปแบบอย่างไร โดยใช้ฟังก์ชัน SetDataFormat โดยมีรูปแบบการเรียกใช้ดังนี้

```
HRESULT SetDataFormat(LPCDIDATAFORMAT lpdf);
```

ฟังก์ชัน SetDataFormat มีพารามิเตอร์เพียงตัวเดียว คือ lpdf พารามิเตอร์นี้จะใช้โครงสร้างของ DIDATAFORMAT ตามที่ได้แสดงด้านล่างนี้

```
Typedef struct
{
    DWORD       dwSize; // ขนาดของโครงสร้างรูปแบบนี้หน่วยเป็นไบต์
    DWORD       dwObjSize; //ขนาดของ DIOBJECTDATAFORMAT เป็นไบต์
    DWORD       dwFlags; //กำหนดการรายงานผลว่าเป็นรูปแบบ Reletive
                         หรือว่า Absolute
    DWOR        dwDataSize; //ขนาดของ packet ข้อมูล
    DWORD       dsNumObjs; //จำนวนอ็อปเจกต์กำหนดให้เป็นขนาดของแถวลำดับ
    LPDIOBJECTDATAFORMAT rgodf; //ตัวชี้ชี้ไปยังอาณ์เรย์อ็อปเจกต์
} DIDATAFORMAT, *LPDIDATAFORMAT;
```

แต่ DirectInput ได้กำหนดรูปแบบของข้อมูลสำหรับอุปกรณ์รับข้อมูลเข้ามาตรฐานไว้เรียบร้อยแล้วนั่นคือ

c_dfDIKeyboard	กำหนดรูปแบบข้อมูลสำหรับแป้นพิมพ์มาตรฐาน
c_dfDIMouse	กำหนดรูปแบบข้อมูลสำหรับเมาส์มาตรฐาน
c_dfDIJoystick	กำหนดรูปแบบข้อมูลสำหรับก้านควบคุมมาตรฐาน

ดังนั้นผู้พัฒนาเกมสามารถใช้รูปแบบข้อมูลที่ถูกกำหนดขึ้นไว้แล้ว ถ้าโปรแกรมประยุกต์นั้นใช้อุปกรณ์รับข้อมูลเข้ามาตรฐาน

ตัวอย่างการใช้ฟังก์ชัน SetDataFormat

```
lpdikey->SetDataFormat(&c_dfDIKeyboard);
```

2.5.2.5 การจองแป้นพิมพ์ด้วยฟังก์ชัน Acquire

มีรูปแบบการเรียกใช้ดังนี้

```
HRESULT Acquire();
```

ตัวอย่างการใช้ฟังก์ชัน Acquire

```
lpdikey->Acquire();
```

2.5.2.6 การอ่านข้อมูลจากแป้นพิมพ์

สำหรับลักษณะข้อมูลของแป้นพิมพ์ที่ถูกกำหนดโดย DirectInput มีลักษณะเป็น State Data นั่นหมายความว่าเมื่ออ่านข้อมูลขึ้นมาจากแป้นพิมพ์ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของแป้นพิมพ์ สำหรับการอ่านข้อมูลจากแป้นพิมพ์จะใช้ฟังก์ชัน GetDeviceState ฟังก์ชันนี้จะอ่านค่าสถานะปัจจุบันของแป้นพิมพ์ขึ้นมา หรือสิ่งที่แป้นพิมพ์ได้ทำครั้งสุดท้ายในลูปของข้อมูลเข้าโดยมีรูปแบบการเรียกใช้ฟังก์ชันดังนี้

```
HRESULT GetDeviceState (DWORD cbData, //ขนาดของระเบียนข้อมูล
                        LPVOID  lpvData); // ตัวชี้ชี้ไปยังตำแหน่งที่เก็บข้อมูล
```

ตัวอย่างการใช้ฟังก์ชัน GetDeviceState

```
UCHAR keystate[256];
lpdikey ->GetDeviceState(256, keystate)
```

DirectInput จะมีค่าคงที่ของแต่ละแป้นบนแป้นพิมพ์เพื่อให้ผู้พัฒนาเกมสามารถตรวจสอบสถานะของแป้นบนแป้นพิมพ์ได้ว่าเป็นอย่างไร โดยค่าคงที่นี้จะขึ้นต้นด้วย DIK_ แล้วตามด้วยค่าแป้น ตามตารางด้านล่างนี้

ตารางที่ 2.6 ค่าแป้นบนแป้นพิมพ์

ค่าตัวบ่งชี้	ความหมาย
DIK_ESCAPE	ปุ่ม ESC
DIK_0 – DIK_9	หมายเลข 0 ถึง 9
DIK_A – DIK_Z	A ถึง Z
DIK_RETURN	ปุ่ม Enter
DIK_LCONTROL	ปุ่ม Ctrl ทางซ้ายมือ
DIK_RCONTROL	ปุ่ม Ctrl ทางขวามือ
DIK_SPACE	ปุ่ม Spacebar
DIK_F1 – DIK_F12	ปุ่ม F1 ถึง F12
DIK_UP	ปุ่ม ลูกศรชี้ขึ้น
DIK_UP	ปุ่ม ลูกศรชี้ขึ้น

ตารางที่ 2.6 (ต่อ) ค่าแป้นบนแป้นพิมพ์

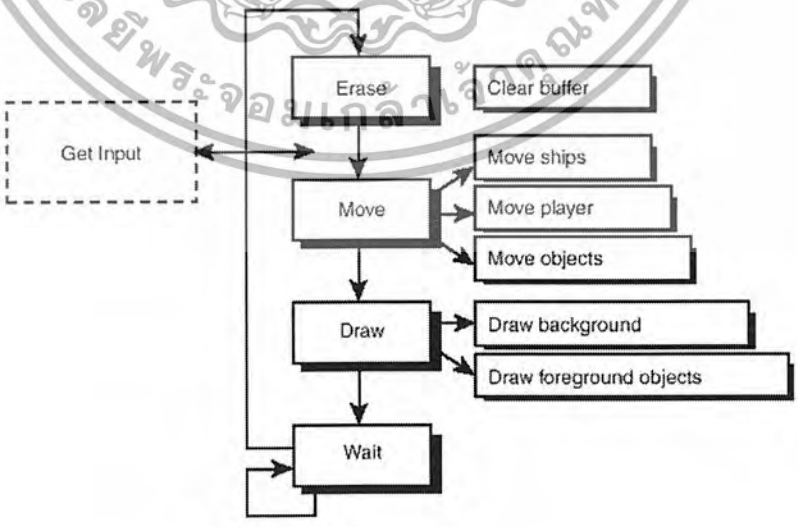
ค่าตัวบ่งชี้	ความหมาย
DIK_DOWN	ปุ่ม ลูกศรชี้ลง
DIK_LEFT	ปุ่ม ลูกศรชี้ไปทางซ้าย
DIK_RIGHT	ปุ่ม ลูกศรชี้ไปทางขวา
DIK_PRIOR	ปุ่ม PageUp
DIK_NEXT	ปุ่ม PageDown

หลังจากทำการอ่านค่าสถานะของแป้นพิมพ์แล้ว จะสามารถตรวจสอบได้ว่าแป้นที่สนใจมีสถานะเป็นอย่างไรด้วยการตรวจสอบค่าบิตของแป้นนั้น ๆ ถ้าแป้นถูกกด ค่าบิต 0X80 จะมีค่าเป็น 1แต่ถ้าแป้นนั้น ๆ ไม่ได้ถูกกด หรือแป้นนั้น ถูกปล่อยไปแล้วค่าบิต 0X80 จะมีค่าเป็น 0

ตัวอย่างการตรวจสอบค่าของแป้น

```
int ship_x = 100, ship_y = 100;
UCHAR keystate[256];
lpdikey->GetDeviceState(256, keystate)
if (keystate[DIK_RIGHT] & 0x80) ship_x++;
if (keystate[DIK_DOWN] & 0x80) ship_y++;
```

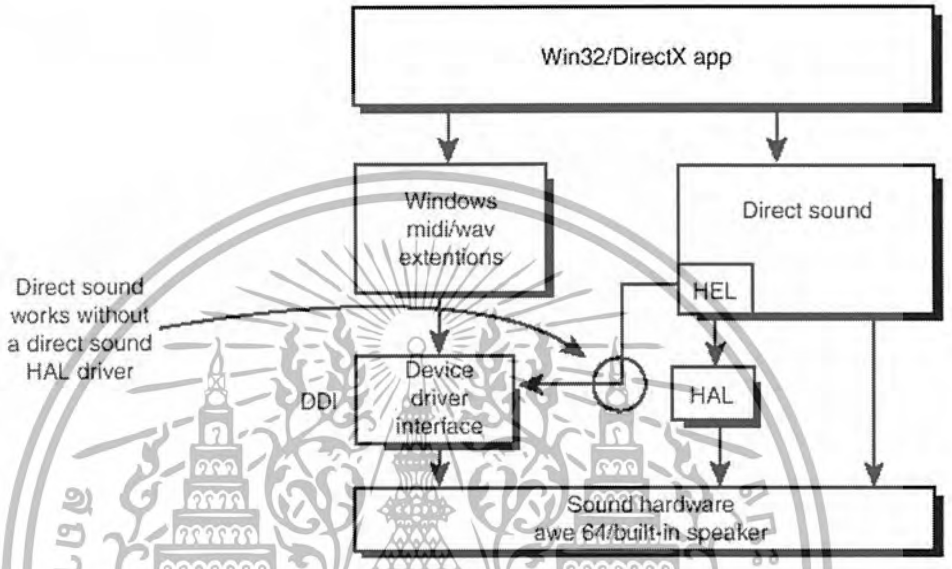
จากตัวอย่างเป็นการตรวจสอบว่าแป้นลูกศรชี้ลง หรือลูกศรชี้ไปด้านขวาถูกกดหรือไม่



รูปที่ 2.13 ลูปของข้อมูลเข้า

2.6 วิธีการแสดงเสียง

ในส่วนนี้จะกล่าวถึง Direct Sound ที่ใช้ในการสร้างเสียงและมีประโยชน์อย่างมากในกรณีที่ไม่มีไดรฟเวอร์ของการ์ดเสียงตัวนั้น โดย Direct Sound จะติดต่อกับการ์ดเสียงได้โดยตรงไม่ต้องอาศัยไดรฟเวอร์ของการ์ดเสียง ดังรูป



รูปที่ 2.14 การทำงานของ DirectSound

ส่วนประกอบพื้นฐานของ Direct Sound มีอยู่ 3 ส่วนได้แก่• ขณะ run-time .DLL จะถูกโหลดขณะเรียกใช้ Direct Sound• ขณะ compile-time Dsound.Lib จะทำการสร้างโปรแกรมประยุกต์ของ Direct sound • ขณะ compile-time Dsound.h จะทำการสร้างโปรแกรมประยุกต์ของ Direct sound

ดังนั้นจะต้องทำการเพิ่ม file เหล่านี้เข้าไปในโปรแกรมประยุกต์ที่ต้องการสร้างด้วย

ขั้นตอนการสร้างอ็อปเจกต์ของ Direct sound

2.6.1 การสร้างอ็อปเจกต์ของ Direct Sound

การที่จะสร้างอ็อปเจกต์ Direct sound จะต้องทำการเรียก Function DirectSoundCreate() ซึ่งกำหนด ต้นแบบ ดังนี้

```
HRESULT DirectSoundCreate(LPGUID          lpGuid, // GUID ของ การ์ดเสียง
                          LPDIRECTSOUND  *lpDS, //interface pointer to obj
                          Iunknow    FAR  *pUnkOuter ) //ปกติ ค่าว่าง
```

โดยจะต้องประกาศตัวชี้เพื่อใช้ชี้อ็อปเจกต์ของ Direct Sound ดังนี้

```
LPDIRECTSOUND lpds;
```

เมื่อทำการสร้างอ็อปเจกต์ของ Direct Sound ขึ้นมาแล้ว และเมื่อเรียกใช้เสร็จก็ต้องทำการคืนค่าอ็อปเจกต์
นั้นด้วยคำสั่ง

```
lpds -> Release();
```

2.6.2. การกำหนดค่า Cooperation Level ของ Direct Sound

จะคล้ายกับการกำหนดค่า cooperation ของ Direct Draw โดยการเรียกฟังก์ชัน SetCooperativeLevel โดยมีต้นแบบดังนี้

```
HRESULT SetcooperativeLevel (HWND    hwnd, // window handle
                             DWORD   dwLevel); // ตั้งค่า Cooperation
```

ตารางที่ 2.7 ค่าตัวบ่งชี้ของ Cooperation Level ของ DirectSound

ค่าตัวบ่งชี้ dwLevel	ความหมาย
DSSCL_NORMAL	ตั้งเป็นแบบปกติโดยมีขนาด Primary Buffer สำหรับความถี่เสียง 22 kHz, Stereo แบบ 8 bit เป็นที่นิยมที่สุด
DSSCL_PRIORITY	อนุญาติให้เปลี่ยนขนาดของ Primary Buffer ได้ตามต้องการและสามารถเปลี่ยนรูปแบบเสียงเป็นแบบ 16 bit ได้
DSSCL_EXCLUSIV	เหมือนกับ Priority แต่จะได้ยินเสียงเพียงเสียงเดียว
DSSCL_WRITEPRIMARY	เป็นระดับสูงสุดจะควบคุมทุกอย่างของ Primary Buffer เหมาะสำหรับการที่จะสร้างเสียงขึ้นมาเอง

2.6.3 บัฟเฟอร์หลักและรองของ Direct Sound การสร้างเสียงขึ้นมาบนการ์ดเสียง โดยที่ตัวโปรแกรมไม่ต้องไปสนใจ Direct Sound จะเป็นตัวจัดการแทนให้ และไม่จำเป็นต้องสร้างบัฟเฟอร์หลัก โดยเพียงแค่ set ค่า cooperative level ต่ำสุด Directt sound ก็จะจัดการให้ส่วนที่จะต้องสนใจจริงๆ ก็คือ บัฟเฟอร์รองซึ่งส่วนนี้จะทำการเล่นเสียงที่โปรแกรมต้องการจะมีขนาดเท่าไรก็ได้ ขึ้นอยู่กับหน่วยความจำที่มีอยู่ อย่างไรก็ตาม SRAM ที่อยู่บนการ์ดเสียงก็เพียงพอที่จะเก็บเสียงได้มากตามที่ต้องการแล้ว

2.6.4 การสร้างบัฟเฟอร์รองของ Direct Sound

การจะสร้างบัฟเฟอร์รองของเสียงต้องทำการเรียก Function CreateSoundBuffer โดยกำหนดต้นแบบไว้ดังนี้

```
HRESULT CreateSoundBuffer(LPCDSBUFFERDESC lpcDSBufferDesc,
                                        //ตัวชี้ไปยัง DSBUFFERDESC
                          LPLPDIRECTSOUNDBUFFER lplpDSBuff,
                                        //ตัวชี้ไปยังบัฟเฟอร์ของเสียง
                          IUnknown FAR *pUnkOuter ); //ปกติใช้ค่าว่าง
```

ก่อนที่จะสร้างบัฟเฟอร์รองของเสียงจะต้องกำหนดโครงสร้างของ DirectSoundbuffer เสียก่อนโดยมีรูปแบบดังนี้

```
typedef struct {
    DWORD           dwSize; // ขนาดของโครงสร้างนี้
    DWORD           dwFlags; // ค่าตัวบ่งชี้ควบคุม
    DWORD           dwBufferBytes; // ขนาดของบัฟเฟอร์ของเสียงในรูป ไบต์
    DWORD           dwReserved; // ไม่ใช้
    LPWAVEFORMATEX  lpwfxFormat; // รูปแบบรูปแบบของเสียง
} DSBUFFERDESC      *LPDSBUFFERDESC;
```

ตารางที่ 2.8 ค่าตัวบ่งชี้ควบคุมของ DirectSound

ค่าตัวบ่งชี้ของ dwFlags	ความหมาย
DSBCAPS_CTRLALL BUFFER	ต้องมีการควบคุมขนาด
DSBCAPS_CTRLDEFAULT	เป็นการตั้งค่าโดยปริยาย
DSBCAPS_CTRLFREQUENCY	ควบคุมความถี่ของเสียง
DSBCAPS_CTRLPAN	ควบคุมความ balance ของเสียง
DSBCAPS_CTRLVOLUME	ควบคุมความดังของเสียง
DSBCAPS_STATIC	บอกว่าบัฟเฟอร์จะใช้สำหรับข้อมูลเสียงแบบ static
DSBCAPS_LOCHARDWARE	ใช้ Hardware ทำการสร้างเสียงขึ้นมาและใช้ หน่วยความจำแทนบัฟเฟอร์
DSBCAPS_LOCSOFTWARE	บังคับให้บัฟเฟอร์ใช้หน่วยความจำของซอฟต์แวร์เป็นตัวเก็บ
DSBCAPS_PRIMARYBUFFER	บอกให้รู้ว่าเป็นแบบบัฟเฟอร์หลักที่สามารถสร้างด้วยตัวเอง

โดยส่วนใหญ่ จะกำหนดค่า Control flag เป็น DSBCAPS_CTRLDEFAULT| DSBCAPS_STATIC| DSBCAPS_LOCSOFTWARE

ต่อไปเป็นการ กำหนดรายละเอียดของเสียงโดยสร้างรูปแบบของ Waveformatex ขึ้นมาดังนี้

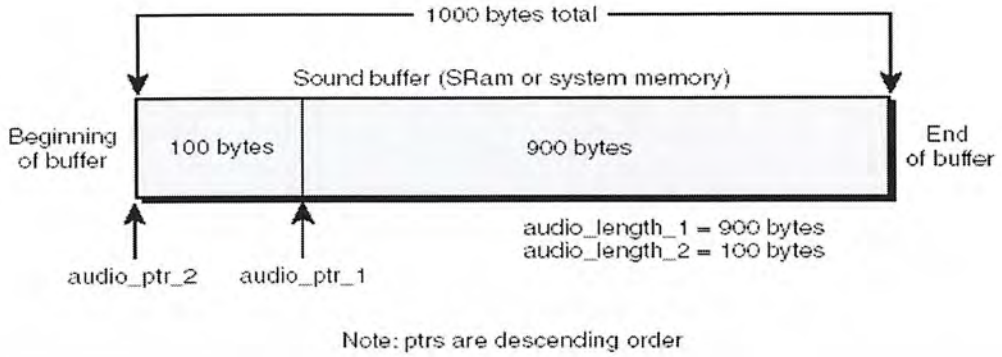
```
typedef struct {
  WORD   wFormatTag ; //ปกติใช้ WAVE_FORMAT_PCM
  WORD   nChannels ; //จำนวนช่องสัญญาณจะใช้ 1 สำหรับเสียงแบบโมโน
                          และ 2 สำหรับสเตอริโอ
  DWORD   nSamplesPerSec; //samples per second
  DWORD   nAvgBytesPerSec ; //average data rate
  WORD   nBlockAlign ; //nchannels * bytespersample
  WORD   wBitsPerSample ; //bits per sample
  WORD   cbSize ; // ตั้งเท่ากับ 0
} WAVEFORMATEX;
```

2.6.5 การเขียนข้อมูลลงบน Secondary Buffer ของ Direct Sound

ใน Direct Draw Surface จะต้องทำการยึดหน่วยความจำของ surface ก่อน แล้วจึงเขียนใน Direct Sound ก็เช่นเดียวกัน แตกต่างกันที่ Direct sound มี 2 ตัวชี้ ที่ใช้ในการเขียน โดยตัวชี้แรกจะใช้เขียนข้อมูลในส่วนแรกแบะข้อมูลที่เหลือสำหรับตัวชี้ที่สอง โดยมีการกำหนด ต้นแบบ Lock ไว้ดังนี้

```
HRESULT Lock(DWORD          dwWriteCursor,
             DWORD          dwWriteBytes, // ขนาดของ Buffer ที่ต้องการ
lock
             LPVOID         lplpvAudioPtr1,
             LPDWORD        lpdwAudioBytes1,
             LPVOID         lplpvAudioPtr2,
             LPDWORD        lpdwAudioBytes2,
             DWORD          dwFlags);
```

ตัวอย่าง เช่น ต้องการสร้างบัฟเฟอร์ของเสียงขนาด 1000 ไบต์ เมื่อทำการยึดบัฟเฟอร์เพื่อจะทำการเขียน จะต้องใช้ สองตัวชี้ในการอ้างอิง โดยขนาดของตัวชี้ตัวแรก อาจจะมีขนาด 900 ไบต์ ส่วน ที่เหลือ 100 ไบต์จะเป็นของตัวชี้ตัวที่สอง ดังรูปภาพที่ 2.17



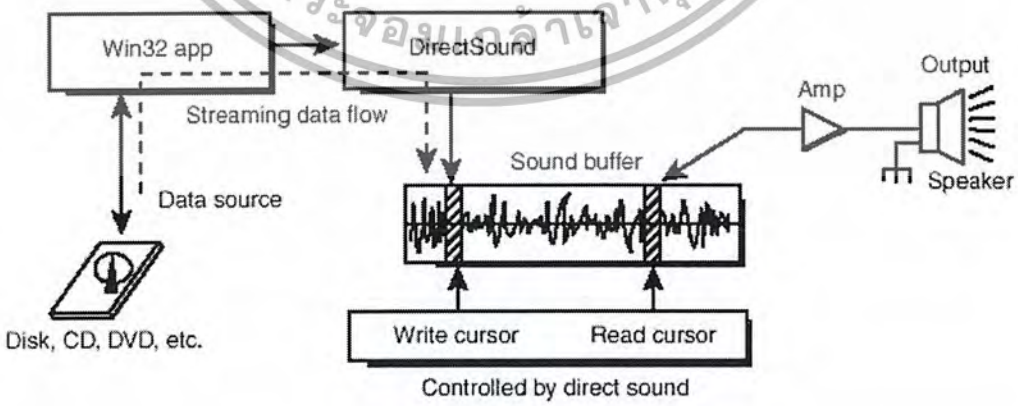
รูปที่ 2.15 การเขียนข้อมูลเสียงลงบัฟเฟอร์เสียง

หลังจากทำการ สร้างบัฟเฟอร์ของเสียงและ โหลดเสียงทั้งหมด ขั้นต่อไปก็จะเป็นการ นำเสียงที่พร้อมนำมาแสดง โดย DirectX มีฟังก์ชันมากมายในการแสดงเสียงที่จำเป็นได้แก่

2.6.6 การแสดงเสียงของ Direct Sound

ใช้ฟังก์ชัน Play ในการแสดงเสียงจากบัฟเฟอร์ของเสียงโดย มีตัวต้นแบบดังนี้

```
HRESULT Play(DWORD     dwReserved1,
             DWORD     dwReserved2,  // กำหนดเป็น 0 ทั้งสองค่า
             DWORD     dwFlags);     // ควบคุมตัวบ่งชี้ในการเล่น
                                     ถ้าต้องการเล่นวนรอบ
                                     //ตั้งค่าเป็น DSBPLAY_LOOPING ถ้าต้องการ
                                     เล่นเพียง 1 ครั้งให้ตั้งค่า 0
```



รูปที่ 2.16 การแสดงข้อมูลเสียง

2.6.7 การหยุดเสียงของ Direct Sound

ถ้าต้องการหยุดเสียงที่กำลัง เล่นอยู่ ใช้ฟังก์ชัน Stop โดยกำหนดตัวต้นแบบดังนี้

```
HRESULT Stop();
```

2.6.8 การตั้งระดับความดังของเสียงของ Direct Sound

ถ้าต้องการปรับความดังของเสียงจะเรียกใช้ ฟังก์ชัน SetVolume มีการกำหนดต้นแบบดังนี้

```
HRESULT SetVolume(LONG VOLUME);
```

2.7 หลักการสร้างเกม

2.7.1 ทฤษฎีเกม

ทฤษฎีเกม เป็นเครื่องมือชนิดหนึ่งที่อธิบายและกล่าวถึงปัญหาทางสังคม เกมทั่วๆไปมักจะสะท้อนหรือบอกให้ทราบถึงลักษณะที่เกิดขึ้นในสถานการณ์จริง โดยเฉพาะอย่างยิ่งในสถานการณ์การแข่งขันหรือการร่วมมือกัน เกมเหล่านี้จะแนะนำกลยุทธ์ที่ใช้ในการแก้ไขสถานการณ์ที่เกิดขึ้น ซึ่งเราอาจสามารถเข้าใจกลยุทธ์ของผู้เล่นเกมในแต่ละเกม เราอาจสามารถทำนายได้ว่าประชาชน กลุ่มการเมือง หรือรัฐต่างๆจะวางตัวอย่างไรในสถาน การณ์ที่กำหนดให้

โดยทั่วไปขณะที่คนเรามุ่งจะชนะเกมการแข่งขันเขาจะพยายาม "ชนะ" หรือบรรลุผลประโยชน์หรือเป้าหมายในการแข่งขัน อย่างไรก็ตามทั้งในการเล่นเกมและในความเป็นจริง จำเป็นต้องปฏิบัติตามกฎกติกาที่วางไว้เพื่อให้บรรลุสิ่งเหล่านี้ ในบางเกมก็คล้ายกับสถานการณ์จริง คือผู้ชนะครอบครองหมด โดยธรรมชาติเกมเหล่านี้มีลักษณะการแข่งขันสูง คือ มีผู้ชนะเพียงคนเดียว เช่น เกมหมากรุกเป็นตัวอย่างของเกมเหล่านี้ เกมอื่นๆอาจต้องการความร่วมมือเพื่อชัยชนะ ตัวอย่าง เช่น เกมวีดีโอที่ออกมาใหม่จำนวนมากต้องการกลยุทธ์ความร่วมมือจากผู้เล่นหลายคนเพื่อให้ผู้เล่นคนใดคนหนึ่งได้รับชัยชนะ ในโลกแห่งความจริงแม้ในช่วง เวลาการต่อสู้ ส่วนมากคู่ต่อสู้จะมีผลประโยชน์ร่วมกัน และต้องร่วมมือกันในระดับหนึ่ง

2.7.1.1 ความหมายของทฤษฎีเกม

ทฤษฎีเกมให้เครื่องมือที่ช่วยวิเคราะห์ตรวจสอบกลยุทธ์ที่เกิดจากความสัมพันธ์ของ ผู้เล่นเกมสองฝ่ายหรือมากกว่าสองฝ่าย โดยการใช้สถานการณ์จำลองทางคณิตศาสตร์แบบง่ายๆ ในการศึกษาความเกี่ยวข้องทางสังคมที่ยุ่งยากซับซ้อนทฤษฎีเกมจะอธิบายให้ทราบถึงศักยภาพ และความเสี่ยงที่ควบคู่มากับพฤติกรรมที่ต้องร่วมมือกันในระหว่างคู่แข่งขันที่ไม่ไว้วางใจกันและกัน แม้ว่าเราจะไม่คุ้นเคยเหมือนเกมที่เล่นบนแผ่นกระดานแบบอื่นๆหรือวีดีโอเกม แต่บทเรียนที่ได้รับจาก

ทฤษฎีเกมมีลักษณะเป็นนามธรรมหรือมีสมมติฐานมากกว่าสามารถนำไปใช้ในการแก้ไขสถานการณ์ทางสังคมได้อย่างกว้างขวางมากกว่า

เกมที่เล่นจำลองเหตุการณ์ในชีวิตจริง มีองค์ประกอบ 5 อย่าง ได้แก่

1.) ผู้เล่นหรือผู้ตัดสินใจ
2.) กลยุทธ์ของผู้เล่นแต่ละคน
3.) กฎกติกาที่ใช้ในการควบคุมการตัดสินใจและแสดงพฤติกรรมของผู้เล่นแต่ละคน
4.) ผลคะแนน คะแนนที่ผู้เล่นแต่ละคนได้รับ เนื่องจากตัดสินใจเลือกทางเดิน
5.) การมีส่วนได้ส่วนเสียที่สะสมขึ้นเรื่อยๆของผู้เล่นแต่ละคน เนื่องมาจากคะแนนสะสม

เกมนี้ตั้งสมมติฐานไว้ว่า ผู้เล่นแต่ละคนจะใช้กลยุทธ์ที่ช่วยให้เขาประสบความสำเร็จได้คะแนนมากที่สุดในทุกๆสถานการณ์ ในชีวิตจริงที่เต็มไปด้วยสถานการณ์ที่เกิดขึ้นโดยตั้งใจหรือไม่ตั้งใจก็ดี ผู้คนต่างแสวงหาผลประโยชน์โดยให้คนอื่นเป็นฝ่ายเสียเปรียบ นี่เป็นการกระทำที่นำไปสู่ความขัดแย้งหรือแข่งขันกัน มีการใช้เกมอธิบายความสัมพันธ์เหล่านี้ โดยการกำหนดผลประโยชน์ของผู้เล่นทั้งสองฝ่ายให้ขัดแย้งกัน เมื่อผู้เล่นคนหนึ่งเสียผลประโยชน์มาก ผู้เล่นอีกคนหนึ่งก็เสียผลประโยชน์น้อย เพื่อจะบรรลุผลประโยชน์ร่วมกัน ผู้เล่นหลายคนจำเป็นต้องใช้กลยุทธ์ร่วมมือกัน เพราะว่าถ้าหากผู้เล่นแต่ละคนต่างเลี่ยงทุ่มเทให้กับการได้และเสียมากที่สุด คะแนนที่ได้รับจะไม่มาก อย่างไรก็ตาม เกมเหล่านี้จะเน้นให้เห็นถึงความยากลำบากของการได้รับความร่วมมือจากคู่แข่งขันที่ไม่น่าไว้วางใจ เพราะว่าผู้เล่นแต่ละคนต่างก็อยากจะได้ผลประโยชน์มากที่สุด การร่วมมือกันต้องการให้ผู้เล่นทั้งสองฝ่ายประนีประนอมกัน ห้ามใจตนเองมิให้มุ่งหวังจะได้รับประโยชน์สูงสุด และในการประนีประนอมกันผู้เล่นแต่ละคนต้องเสี่ยงกับการสูญเสีย ถ้าหากคู่ต่อสู้ตัดสินใจอยากได้ผลประโยชน์สูงสุด ผู้เล่นมีแนวโน้มชอบเสียน้อยมากกว่าที่จะสูญเสียจนหมด

2.7.1.2 **ประโยชน์ของการมีทฤษฎีเกม**

สถานการณ์จำลองเหล่านี้จะทำให้ผู้เล่นสามารถหยั่งรู้ในกลยุทธ์ที่เป็นผู้เล่นเกมคนอื่นใช้เป็นทางเลือกและคะแนนที่พวกเขาจะได้รับจากสถานการณ์นั้น ๆ การหยั่งรู้นี้ ผู้ตัดสินใจจะสามารถประเมินผลที่จะเกิดขึ้นจากการกระทำของเขาได้ดี และสามารถตัดสินใจเพื่อบรรลุเป้าหมายที่ต้องการโดยหลีกเลี่ยงความขัดแย้ง

ยกตัวอย่างทฤษฎีการยับยั้งฝ่ายตรงกันข้าม ทำให้เกิดยุทธศาสตร์การป้องกันตัวของสหรัฐอเมริกา ตั้งแต่สิ้นสงครามโลกครั้งที่ 2 มีสมมติฐานว่าการตอบโต้อย่างรุนแรง แบบตาต่อตาฟันต่อฟันสามารถป้องกันยับยั้งพฤติกรรมก้าวร้าวของผู้บุกรุกได้ ถ้าหากปัจเจกชนเชื่อว่าพฤติกรรมก้าวร้าวรุนแรงอาจก่อให้เกิดการตอบโต้อย่างรุนแรงจากผู้อื่น ปัจเจกชนผู้นั้นย่อมจะไม่ประพฤติก้าวร้าวรุนแรงต่อผู้อื่น การข่มขู่ว่าจะตอบโต้โจมตีไม่สามารถลดความรุนแรงที่จะเกิดขึ้น

โดยตรง แต่การรับรู้ถึงผลลัพธ์ที่จะเกิดขึ้นเนื่องจากความก้าวร้าวจะช่วยลดความรุนแรงในสถานการณ์ที่อาจขึ้นเกิดจากการโต้ตอบโจมตีกันลงมา หากปัจเจกชนทั้ง 2 ฝ่ายยอมรับว่าผลประโยชน์สูงสุดที่จะได้รับเกิดจากการหลีกเลี่ยงไม่ใช้ความรุนแรงโต้ตอบกัน ไม่มีการอาฆาตมาดร้ายกันและกัน นี่คือแนวทางหลักเบื้องหลังสงครามเย็นระหว่างสหรัฐอเมริกากับรัสเซีย

แนวความคิดเกี่ยวกับผลประโยชน์รวมที่เกิดจากการยับยั้งฝ่ายตรงกันข้ามทำให้เกิดมาตรการควบคุมอาวุธและการร่วมมือกัน การเน้นความสนใจเรื่องยุทธศาสตร์ทางเลือกและผลตอบแทนที่จะได้รับ ทฤษฎีเกมอธิบายให้เราเห็นภาพวาความสัมพันธ์ที่มีศักยภาพในการทำลายล้าง เราสามารถควบคุมจัดการและเปลี่ยนแปลงให้เกิดผลประโยชน์ร่วมกันได้ รวมทั้งหลีกเลี่ยงการแข่งขันการสร้างอาวุธ และการทำสงครามนิวเคลียร์

2.4.2 โครงสร้างของเกม

ปัญหาสำคัญในการออกแบบโครงสร้างเกมคือการจินตนาการตอนจบของเกมและวิธีที่จะทำให้ระบบทำงานได้ โครงสร้าง I/O จะทำให้ทราบถึงข้อจำกัดที่มีในโครงสร้างเกม ผู้ออกแบบมีอิสรภาพในการออกแบบโครงสร้างภายใน เพราะผู้เล่นจะติดต่อตรงส่วนนั้นโดยตรงไม่ได้หรือไม่จำเป็นต้องเข้าใจการทำงานภายในระบบ ผู้ออกแบบเกมต้องระบุคีย์เอลิเมนต์ (key element) จำนวนหนึ่งจากหัวข้อสิ่งแวดล้อมต่างๆ และเริ่มสร้างเกมขึ้นรอบๆ คีย์นั้น คีย์นี้เป็นสัญลักษณ์ของคำสั่งที่ระบุอยู่ภายใน โดยคีย์เอลิเมนต์จะต้องยอมให้ผู้เล่นทำสิ่งที่ต้องการทำได้เพื่อให้จินตนาการไปกับเกม เป็นต้นว่าในเกมต่อสู้ การยิงถือเป็นคีย์เอลิเมนต์ ถ้าการยิงของผู้เล่นถูกจำกัด ผู้เล่นก็จะไม่สามารถจินตนาการไปตามเกมได้ ซึ่งเกมส่วนมากจะใช้คีย์เอลิเมนต์หลายคีย์

ความผิดพลาดทั่วๆ ไปของผู้ออกแบบคือการใส่รายละเอียดต่างๆ ในโครงสร้างเกมมากเกินไปทำให้กลายเป็นเกมที่ประณีตมากเกินไปหรือผู้เล่นสามารถใช้วิธีโกงในการดำเนินเกมได้ จึงควรปรับเปลี่ยนโครงสร้าง I/O ให้เหมาะสม

2.7.3 โครงสร้างโปรแกรม

โครงสร้างโปรแกรมเสมือนเป็นวิธีที่จะเปลี่ยนโครงสร้าง I/O และโครงสร้างเกมให้ออกมาเป็นเกมจริงๆ เอลิเมนต์หนึ่งที่สำคัญที่สุดของโครงสร้างโปรแกรมคือแผนที่หน่วยความจำ ต้องมีการจัดสรรพื้นที่ของหน่วยความจำให้ดีเพราะอาจมีการใช้หน่วยความจำมากเกินไป ทำให้เหลือหน่วยความจำไม่เพียงพอสำหรับงานสำคัญๆ

2.7.4 การประเมินการออกแบบ

การประเมินการออกแบบคือการประเมินจุดบกพร่องทั้งหมดของการออกแบบที่จะทำให้เกมมีข้อเสีย ต้องมีการตรวจดูความมั่นคงของโครงสร้างเกม เนื่องจากเกมเป็นกระบวนการไดนามิค (dynamic process) ดังนั้นอาจมีเหตุการณ์ใดๆ ที่ทำให้เกมหลุดพ้นจากขอบเขตที่ควบคุมไว้ ตัว

อย่างเช่น หากในเกมมีเงินให้ผู้เล่น เราควรจะเขียนคำสั่งควบคุมเอาไว้ด้วย และควรทดสอบหาวิธีลัดที่ผู้เล่นอาจใช้เพื่อให้ดำเนินไปถึงตอนจบเร็วๆ ได้ เพราะผู้เล่นจะไม่ได้รับประสบการณ์ที่ต้องการให้ผู้เล่นได้รับ

2.7.5 สรุปการออกแบบก่อนทำการเขียนโปรแกรม

กำหนดโครงสร้าง I/O และโครงสร้างภายในเกม แนวของเอกสารอ้างอิงนี้ควรจะเน้นด้านประสบการณ์ที่ผู้เล่นจะได้รับมากกว่าการเทคนิคที่ใช้ เปรียบเทียบเอกสารส่วนแรกนี้กับหมายเหตุโครงสร้างโปรแกรมเบื้องต้น ปรับเอกสารโครงสร้างโปรแกรมถ้าจำเป็น

2.7.6 การเขียนโปรแกรม

การเขียนโปรแกรมต้องการรายละเอียดมากกว่าสิ่งอื่นๆ บางกรณีที่เกมสร้างไม่สำเร็จเป็นเพราะโปรแกรมเมอร์ไม่มีทักษะในการเขียนเกมหรือไม่มีความพยายาม

2.7.7 การทดสอบการเล่นเกม

ในทางทฤษฎีแล้ว การทดสอบระบบเกมคือกระบวนการที่จะให้ข้อมูลเพื่อใช้ปรับปรุงการออกแบบเกม ส่วนทางปฏิบัติ การทดสอบระบบเกมแสดงให้เห็นแนวการออกแบบพื้นฐานและปัญหาในการแก้ไขโปรแกรม บางครั้งการทดสอบระบบเกมแยกให้เห็นส่วนประกอบต่างๆ หรือการกระทำที่มีน้อยหรือมากเกินไปซึ่งไม่ได้สังเกตก่อนหน้านี้

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

การดำเนินงานทั้งหมดจะเกิดขึ้นไม่ได้หากขาดการวางแผนรวมถึงการออกแบบที่ดีและมีเป้าหมายที่ชัดเจน การดำเนินงานทั้งหมดจะถูกกระทำภายใต้แผนที่รัดกุมเพื่อความสำเร็จตามเป้าหมายที่ได้วางเอาไว้ แผนงานทั้งหมดมีดังนี้

1.) ขั้นตอนการออกแบบ แบ่งออกเป็นหลายส่วนได้แก่

- การออกแบบระบบเกม
- การออกแบบคลาสต่าง ๆ ในระบบเกม
- การออกแบบวิธีการโต้ตอบของฝ่ายคอมพิวเตอร์
- การออกแบบการติดต่อกับระบบของผู้เล่น

2.) ขั้นตอนการสร้างตัวละคร ฉากและรูปภาพต่าง ๆ ที่ใช้ในเกม

เป็นขั้นตอนของการสร้างตัวละคร การสร้างฉากรวมถึงการสร้างรูปภาพต่าง ๆ ที่ใช้ภายในเกม โดยเก็บไว้เป็นแหล่งข้อมูลเพื่อใช้ภายในเกม

3.) ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้เป็นขั้นตอนที่ใช้เวลานานที่สุด เพราะจำเป็นต้องศึกษารูปแบบ วิธีการเขียนโปรแกรมเชิงวัตถุและเป็นลักษณะเรียลไทม์ เพื่อให้ได้โปรแกรมที่กระทัดรัด ทำงานได้ถูกต้องและเข้าใจง่าย

3.1 ขั้นตอนการออกแบบเกม

ขั้นตอนการออกแบบมีรายละเอียดดังนี้

3.1.1 การออกแบบระบบเกม

การออกแบบในส่วนนี้มีความสำคัญที่สุด เพราะระบบเกมคือทุกสิ่งทุกอย่างตั้งแต่แนวคิดของเกม การดำเนินเรื่อง จนถึงการดำเนินเกม โดยการออกแบบในส่วนนี้มีรายละเอียดดังนี้

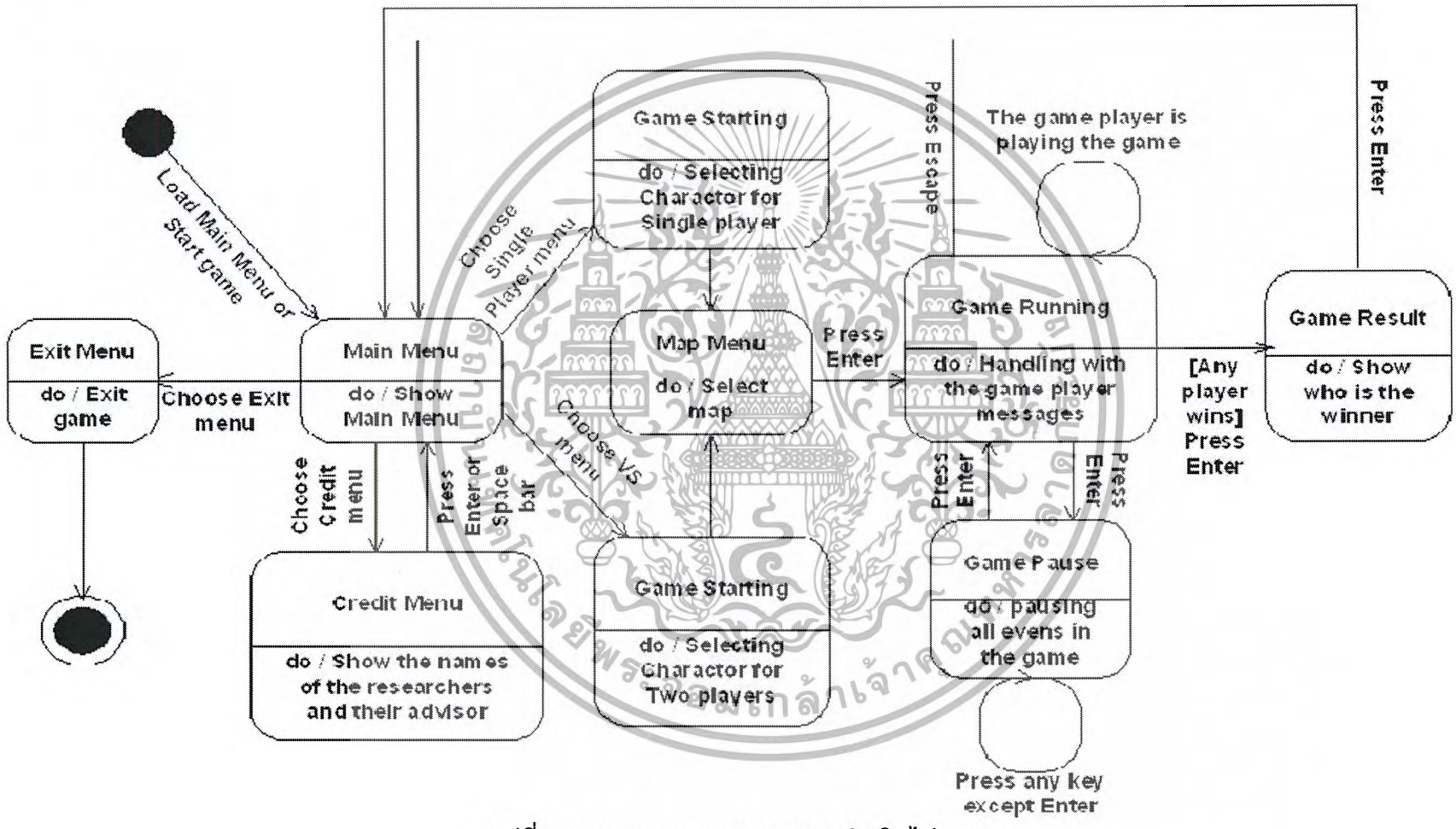
3.1.1.1 การออกแบบทั่วไป

1.) การออกแบบทั่วไปเป็นการออกแบบที่เป็นรูปลักษณ์ของเกม โดยมีรายละเอียดดังนี้ เกมมีลักษณะเป็นการต่อสู้แบบเรียลไทม์ (Real Time) คือ สามารถสั่งการและดำเนินการได้โดยไม่ต้องรอการเปลี่ยนตาหรือการเปลี่ยนเทิร์น

2.) มุมมองที่ใช้เป็นมุมมองจากด้านข้าง

3.) ตัวละครแต่ละตัวจะมีระยะโจมตี พลังโจมตี พลังป้องกัน พลังชีวิตเป็นของตัวเอง

4.) การดำเนินไปของเกมจะเริ่มจากการโหลดหน้าจอเมนูหลัก ถ้าผู้เล่นเลือก Single เกมจะกำหนดค่าเริ่มต้นต่าง ๆ และเข้าสู่การต่อสู้ ถ้าผู้เล่นเลือก VS เกมจะกำหนดค่าเริ่มต้นต่าง ๆ และเข้าสู่การต่อสู้ ขณะที่เข้าสู่การต่อสู้เกมจะดำเนินไปจนกว่าฝ่ายใดฝ่ายหนึ่งชนะ หรือผู้เล่นกด Escape ในกรณีที่ผู้เล่นต้องการทราบว่าผู้จัดทำเป็นใครก็เลือก Credit เกมก็จะแสดงชื่อผู้จัดทำและอาจารย์ที่ปรึกษา การดำเนินไปของเกมสามารถแสดงได้ดังรูป



รูปที่ 3.1 แผนภาพสภานะของการดำเนินไปของเกม

3.1.1.2 การออกแบบคำสั่งและการรับคำสั่งของตัวละคร

เป็นการออกแบบคำสั่งที่ใช้สำหรับตัวละครให้ปฏิบัติ และออกแบบรูปแบบการรับคำสั่ง โดยจะรับคำสั่งการต่าง ๆ ด้วยคีย์บอร์ด

ตัวละครทุกตัว มีคำสั่งให้ดำเนินการดังนี้

1.) MoveRight

เป็นคำสั่งให้ตัวละครเคลื่อนที่ไปทางขวา โดยถ้าโดนโจมตีในช่วงเวลานี้จะทำให้ตัวละครหยุดเคลื่อนที่ทันที สั่งการด้วยคีย์บอร์ดอย่างเดียว ทำการกดคีย์ D (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ลูกศรขวา (สำหรับผู้เล่นคนที่2) เพื่อสั่งให้ตัวละครเคลื่อนที่ไปทางขวา

2.) MoveLeft

เป็นคำสั่งให้ตัวละครเคลื่อนที่ไปทางซ้าย โดยถ้าโดนโจมตีในช่วงเวลานี้จะทำให้ตัวละครหยุดเคลื่อนที่ทันที
สั่งการด้วยคีย์บอร์ดอย่างเดียว ทำการกดคีย์ A (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ลูกศรซ้าย (สำหรับผู้เล่นคนที่2) เพื่อสั่งให้ตัวละครเคลื่อนที่ไปทางซ้าย

3.) MoveDown

เป็นการสั่งให้ตัวละครก้มลง โดยจะช่วยหลบหลีกการโจมตีของฝ่ายตรงข้าม สั่งการด้วยคีย์บอร์ดอย่างเดียว ทำการกดคีย์ S (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ลูกศรลง (สำหรับผู้เล่นคนที่2)เพื่อสั่งให้ตัวละครทำการก้ม

4.) MoveUp

เป็นการสั่งให้ตัวละครกระโดด โดยจะช่วยหลบหลีกการโจมตีของฝ่ายตรงข้าม สั่งการด้วยคีย์บอร์ดอย่างเดียว ทำการกดคีย์ W (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ลูกศรขึ้น (สำหรับผู้เล่นคนที่2) เพื่อสั่งให้ตัวละครทำการกระโดด

5.) Attack

เป็นการสั่งให้ตัวละครทำการโจมตี โดยจะเป็นการทำความเสียหายให้ฝ่ายตรงข้าม ซึ่งจะแบ่งออกเป็น 2 แบบคือ การโจมตีระยะใกล้ (สำหรับนักดาบ) และการโจมตีระยะไกล (สำหรับนักเวทย์) สั่งการด้วยคีย์บอร์ดอย่างเดียว ทำการกดคีย์ I (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ 2 (สำหรับผู้เล่นคนที่2) เพื่อสั่งให้ตัวละครทำการโจมตี

6.) Critical Attack

เป็นการสั่งให้ตัวละครทำการโจมตีด้วยท่าไม้ตาย โดยจะเป็นการทำความเสียหายให้ฝ่ายตรงข้ามอย่างหนัก ซึ่งการโจมตีด้วยท่าไม้ตายจะถูกจำกัดจำนวนครั้งไว้ด้วย สั่งการด้วยคีย์บอร์ดอย่าง

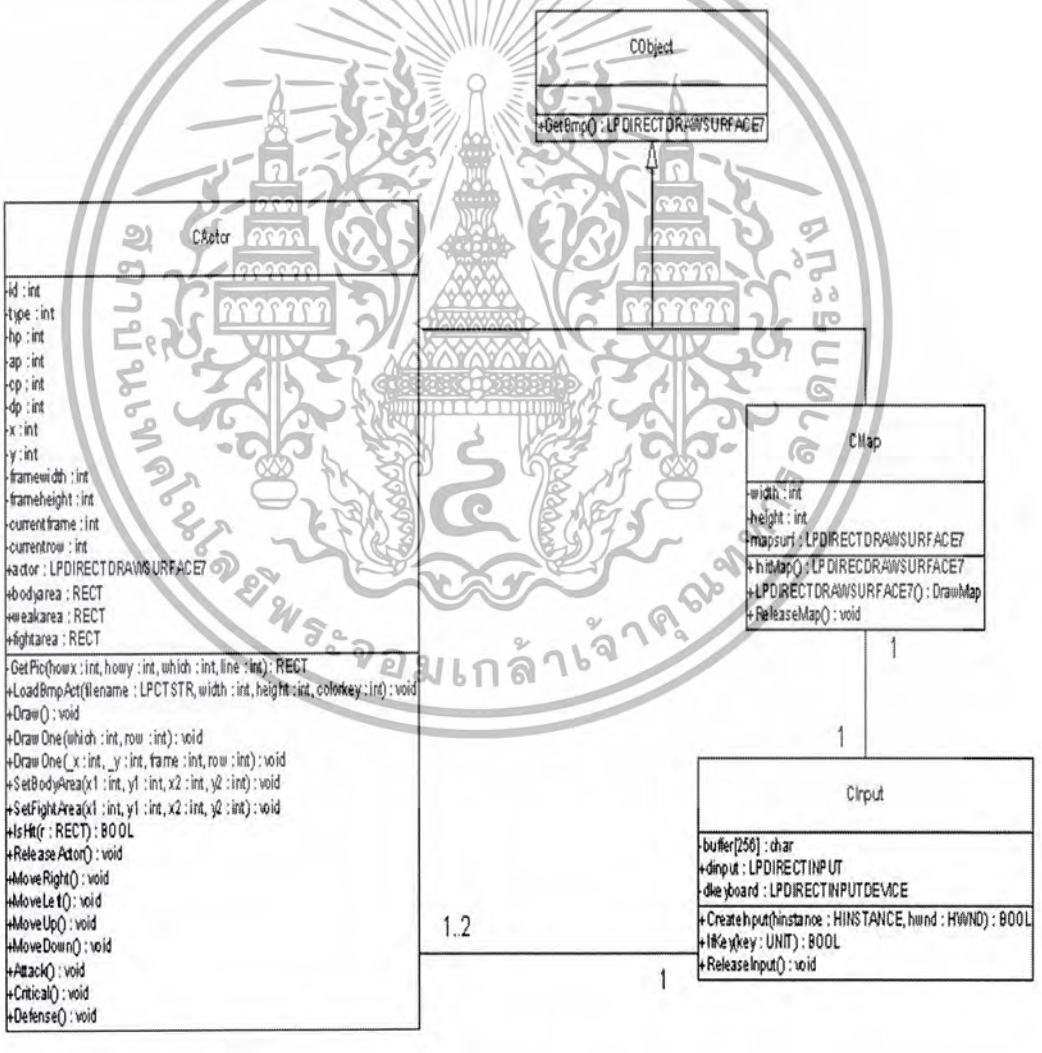
เดียว ทำการกดคีย์ U (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ 1 (สำหรับผู้เล่นคนที่2) เพื่อสั่งให้ตัวละครทำการโจมตีด้วยท่าไม้ตาย

7.) Defense

เป็นการสั่งให้ตัวละครทำการป้องกัน โดยการป้องกันจะช่วยลดค่าความเสียหายจากการโจมตีของฝ่ายตรงข้าม สั่งการด้วยคีย์บอร์ดอย่างเดียว ทำการกดคีย์ O (สำหรับผู้เล่นคนที่1) หรือทำการกดคีย์ 3 (สำหรับผู้เล่นคนที่2) เพื่อสั่งให้ตัวละครทำการป้องกัน

3.1.2 การออกแบบคลาสต่างๆ ในระบบเกม

ในส่วนการออกแบบคลาสต่าง ๆ นี้ จะแสดงคลาสหลัก ๆ ที่มีในระบบเกม ซึ่งมีดังนี้



รูปที่ 3.2 แผนภาพความสัมพันธ์ของคลาสต่าง ๆ

คลาสไดอะแกรมนี้แสดงให้เห็นถึงความสัมพันธ์คลาสหลักในเกม ซึ่งประกอบไปด้วยคลาสดังต่อไปนี้

1.) คลาส Cobject เป็นคลาสต้นแบบของคลาส CActor และ Cmap ซึ่งมีหน้าที่แสดงภาพบนหน้าจอ โดยมีรายละเอียดดังนี้

เมท็อด	รายละเอียด
LPDIRECTDRAWSURFACE7 GetBmp()	แสดงภาพบนหน้าจอ

2.) คลาส Cactor เป็นคลาสตัวละคร ซึ่งจะเป็นตัวแทนของผู้เล่น โดยมีรายละเอียดดังนี้

แอตทริบิวต์	รายละเอียด
int id	รหัสของแต่ละตัวละคร
int type	ชนิดของตัวละคร
int team	ฝ่ายของผู้เล่น
int hp	พลังชีวิตของตัวละคร
int ap	พลังโจมตีของตัวละคร
int cp	พลังโจมตีด้วยท่าไม้ตายของตัวละคร
int dp	พลังป้องกันของตัวละคร
int x	ตำแหน่งบนแกนนอนของตัวละคร
int y	ตำแหน่งบนแกนตั้งของตัวละคร
int framewidth	ความกว้างของเฟรม
int frameheight	ความสูงของเฟรม
int currentframe	เฟรมปัจจุบันของสไปรท์
int currentrow	แถวปัจจุบันของสไปรท์
LPDIRECTDRAWSURFACE7 actor	Surface ที่ใช้แสดงตัวละคร
RECT bodyarea	ระยะของตัวละครที่อยู่ในเฟรม
RECT weakarea	ระยะจุดอ่อนของตัวละคร
RECT fightarea	ระยะทำแต้มของตัวละคร

เมท็อด	รายละเอียด
RECT GetPic(int howx, int howy, int which)	อ่านรูปภาพแต่ละเฟรม
void LoadBmpActor(LPCTSTR filename, int width, int height, int colorkey)	โหลดภาพลงบน Surface
void Draw()	วาดภาพเพียงภาพเดียว
void DrawOne(int which, int row)	วาดภาพเพียงภาพเดียว โดยระบุว่าเป็นภาพใด แถวใด
void DrawOne(int _x, int _y, int frame, int row)	วาดภาพเพียงภาพเดียว โดยกำหนดตำแหน่งให้กับภาพได้
void SetBodyArea(int x1, int y1, int x2, int y2)	กำหนดระยะของตัวละครที่อยู่ในเฟรม
void SetWeakArea(int x1, int y1, int x2, int y2)	กำหนดระยะจุดอ่อนของตัวละคร
Void SetFightArea(int x1, int y1, int x2, int y2)	กำหนดระยะทำแต้มของตัวละคร
BOOL IsHit(RECT r)	ตรวจสอบการทับกันของเฟรมของตัวละคร
Void ReleaseActor()	คืนหน่วยความจำสู่ระบบ
void MoveRight()	เคลื่อนที่ไปทางขวาของหน้าจอ
void MoveLeft()	เคลื่อนที่ไปทางซ้ายของหน้าจอ
void MoveUp()	เคลื่อนที่ไปข้างบนของหน้าจอ
void MoveDown()	เคลื่อนที่ไปข้างล่างของหน้าจอ
void Attack()	โจมตีคู่ต่อสู้
void Critical()	โจมตีคู่ต่อสู้ด้วยท่าไม้ตาย
void Defence()	ป้องกันการโจมตี

3.) คลาส Cmap

แอตทริบิวต์	รายละเอียด
Int width	ความกว้างของฉาก
Int height	ความสูงของฉาก
LPDIRECTDRAWSURFACE7 mapsurf	Surface ที่ใช้แสดงภาพของฉาก
เมท็อด	**รายละเอียด**
void InitMap()	กำหนดค่าเริ่มต้นของการสร้างฉาก
LPDIRECTDRAWSURFACE7 DrawMap()	แสดงภาพของฉากลงบนหน้าจอ
Void ReleaseMap()	คืนหน่วยความจำสู่ระบบ

4.) คลาส CInput

แอตทริบิวต์	รายละเอียด
char buffer[255]	กำหนดอาร์เรย์ขนาด 256 ไบต์ เพื่อเก็บข้อมูลเข้า
LPDIRECTINPUT dinput	กำหนด directinput เพื่อเก็บที่อยู่ของอ็อบเจ็กต์คีย์บอร์ด
LPDIRECTINPUTDEVICE dkeyboard	กำหนดอ็อบเจกต์ของคีย์บอร์ด
เมท็อด	**รายละเอียด**
BOOL CreateInput(HINSTANCE hinstance, HWND hwnd)	สร้างอ็อบเจกต์คีย์บอร์ด
BOOL IfKey(UNIT key)	ตรวจสอบว่าผู้เล่นกดคีย์อะไร
void ReleaseInput()	คืนหน่วยความจำสู่ระบบ

3.1.3 การออกแบบวิธีการโต้ตอบของฝ่ายคอมพิวเตอร์

ในส่วนนี้ต้องออกแบบการโต้ตอบของคอมพิวเตอร์ต่อการกระทำของผู้เล่นซึ่งเป็นเรื่องละเอียดอ่อน ต้องใช้การระดมความคิด โดยจะต้องจำลองตัวเองเป็นผู้เล่นฝ่ายตรงข้าม เพื่อที่จะสามารถวิเคราะห์สถานการณ์และแก้ปัญหาได้ตรงจุด

การออกแบบได้มีรูปแบบที่ชัดเจนบ้างแล้วในการแก้ไขปัญหาบางจุดให้ฝ่ายคอมพิวเตอร์ โดยมีลักษณะดังนี้

1.) คอมพิวเตอร์จะพยายามโจมตีผู้เล่นฝ่ายตรงข้าม

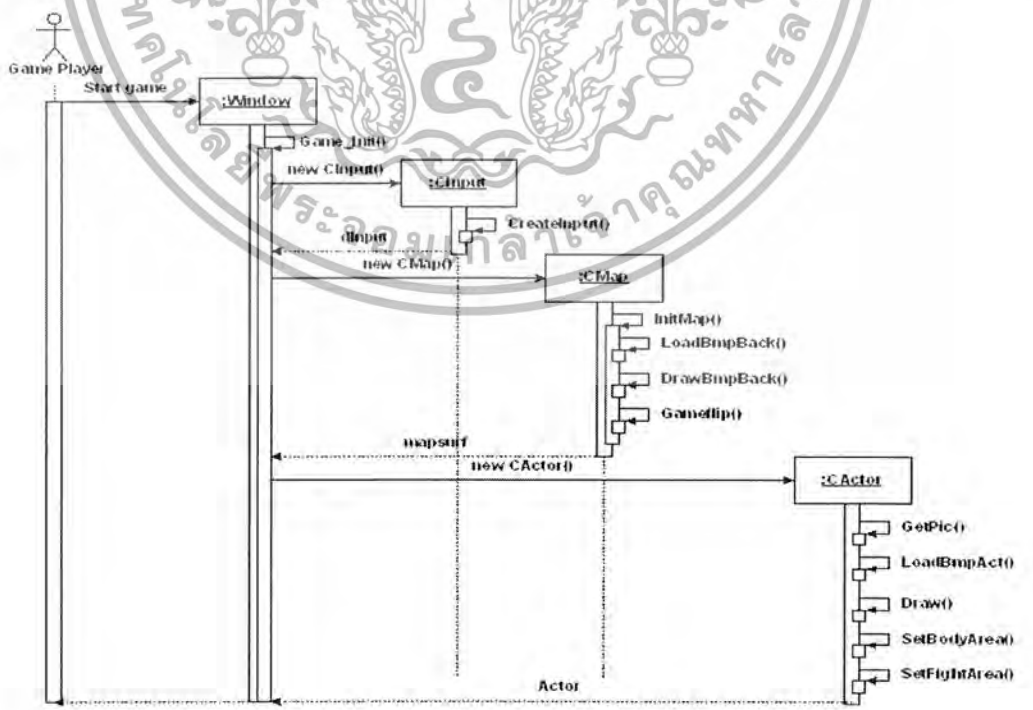
2.) เมื่อผู้เล่นทำการโจมตี คอมพิวเตอร์จะทำการป้องกัน และอาจจะก้มหรือกระโดดเพื่อหลบหลีกการโจมตีของผู้เล่นฝ่ายตรงข้าม

3.) เมื่อคอมพิวเตอร์ไม่สามารถโจมตีถึงตัวผู้เล่นได้ คอมพิวเตอร์จะทำการเคลื่อนที่เข้าไปหาผู้เล่นจนกว่าจะสามารถโจมตีผู้เล่นได้

4.) คอมพิวเตอร์จะหาทางโจมตีผู้เล่นด้วยท่าไม้ตายเมื่อมีโอกาส

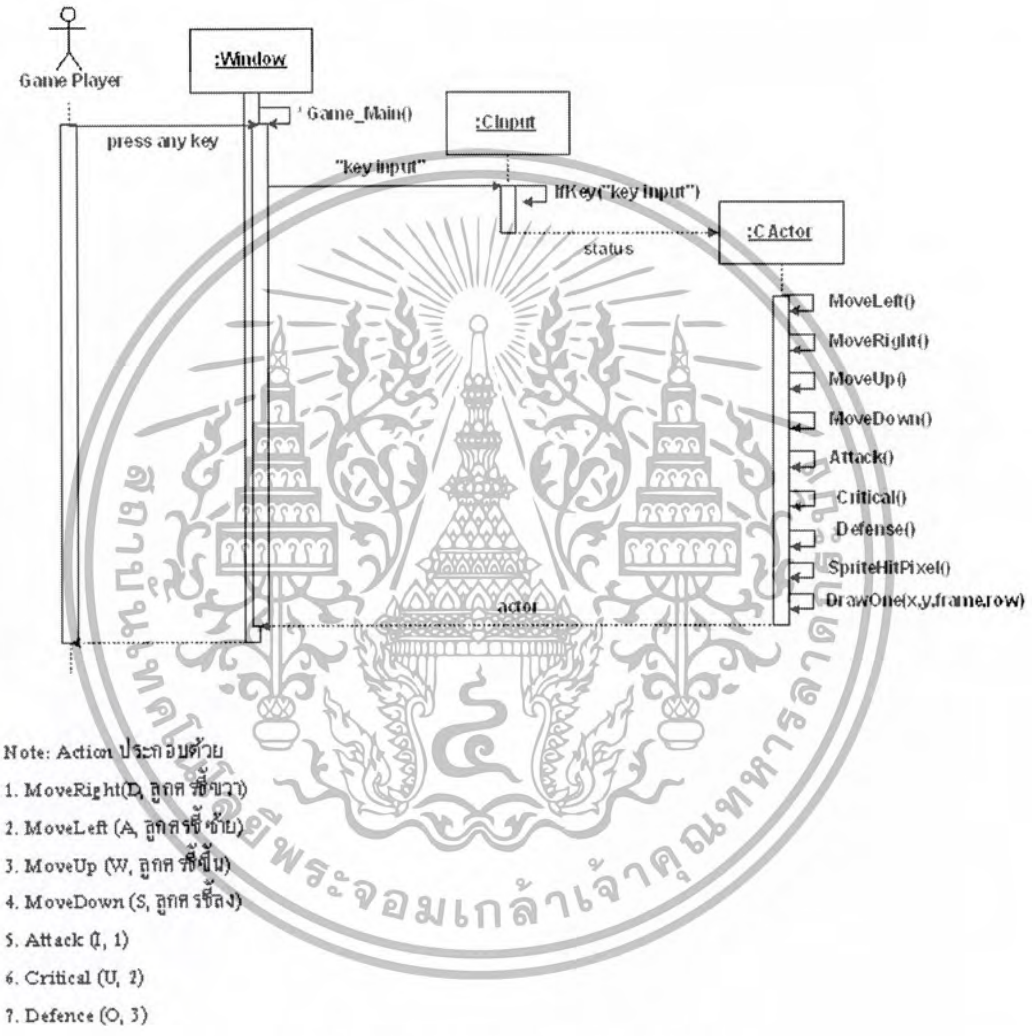
3.1.4 การออกแบบการติดต่อกับเกมโดยผู้เล่น

การออกแบบการติดต่อกับระบบของผู้เล่นเกม แสดงให้ผู้เขียนเกมทราบลำดับการทำงานของโปรแกรมเป็นอย่างไร โดยดูได้จากซีเควนซ์ไดอะแกรมดังต่อไปนี้



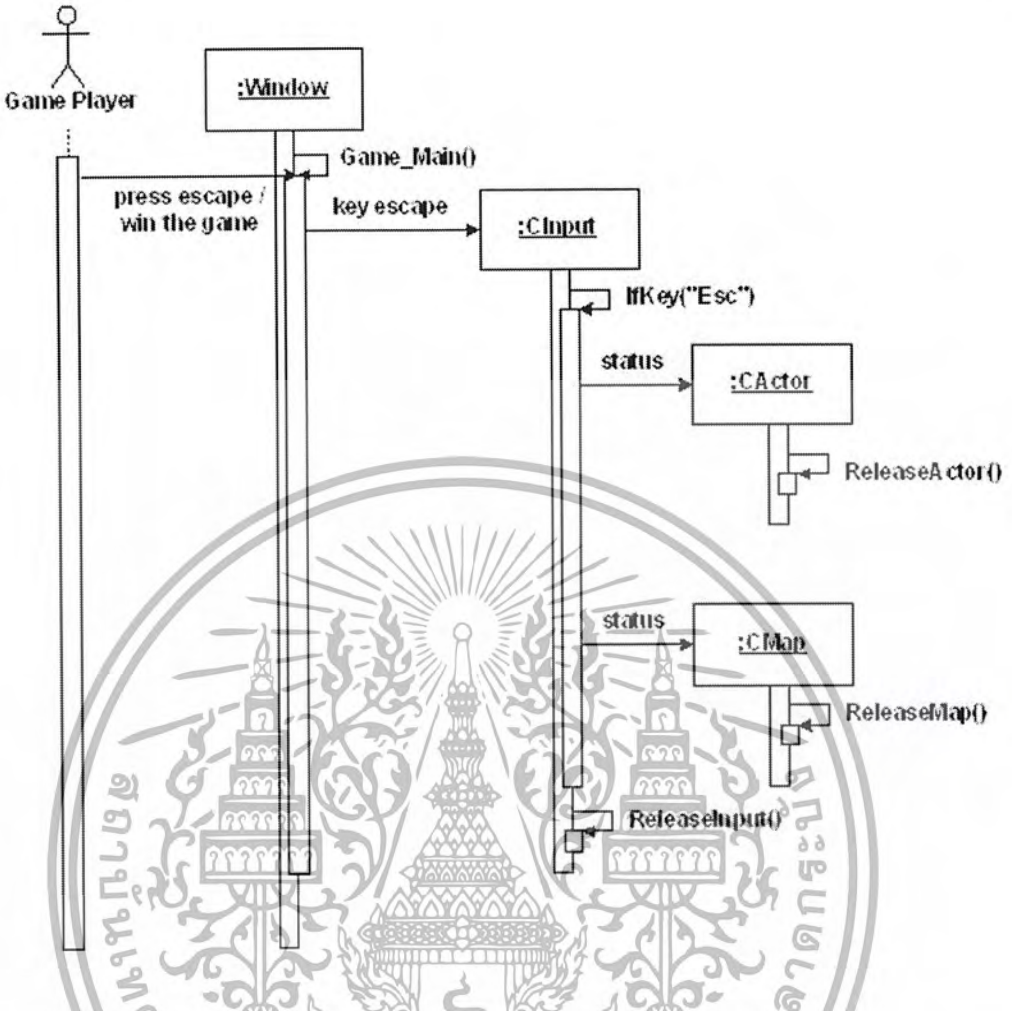
รูปที่ 3.3 ซีเควนซ์ไดอะแกรมของการเริ่มเกม

จากรูปที่ 3.3 แสดงซีเควนซ์ไดอะแกรมของการเริ่มเกม เริ่มจากผู้เล่นเปิดโปรแกรมขึ้นมา โปรแกรมจะสร้าง window ขึ้นมา 1 อันเพื่อเอาไว้รันเกม จากนั้น window จะโหลดเกมขึ้นมาแล้ว ทำการกำหนดค่าเริ่มต้นต่าง ๆ ให้กับเกม ได้แก่ การสร้างคีย์บอร์ด, การสร้างฉากและตัวละครขึ้นมาแสดงบน window



รูปที่ 3.4 ซีเควนซ์ไดอะแกรมของเกมเมน

จากรูปที่ 3.4 แสดงซีเควนซ์ไดอะแกรมของเกมเมน โดยเกมเมนจะรอรับข้อมูลอินพุตของผู้เล่น จากนั้นจะทำการตรวจสอบคีย์ที่ผู้เล่นกดว่าจะเกิดเหตุการณ์ใด จากนั้นก็จะแสดงผลออกมาทางหน้าจอ



รูปที่ 3.5 ซีเควนซ์ไดอะแกรมของการออกจากเกม

รูปที่ 3.5 แสดงการออกจากเกม โดยผู้เล่นกดปุ่ม Escape แล้วเกมจะตรวจสอบดูว่าคีย์ที่ผู้เล่นกดเป็น Escape หรือไม่ ซึ่งในกรณีนี้ใช่ ดังนั้นเกมก็จะทำการคืนหน่วยความจำในการสร้างอ็อบเจกต์ของ CMap, CActor และ Cinput หลังจากนั้นเกมก็จะกลับสู่หน้าจอเมนูหลัก

3.2 ขั้นตอนการสร้างตัวละคร, ฉากและรูปภาพต่าง ๆ ที่ใช้ในเกม

ในขั้นตอนนี้จะเป็นการออกแบบท่าทางของตัวละครในอิริยาบทต่าง ๆ ของตัวละคร โดยโปรแกรมที่ใช้ในการออกแบบตัวละครนี้คือ POSER 5 โดยท่าทางทั้งหมดของตัวละครในเกมมีดังนี้

1.) การเคลื่อนที่ไปทางขวา

2.) การเคลื่อนที่ไปทางซ้าย

3.) การก้ม

4.) การกระโดด

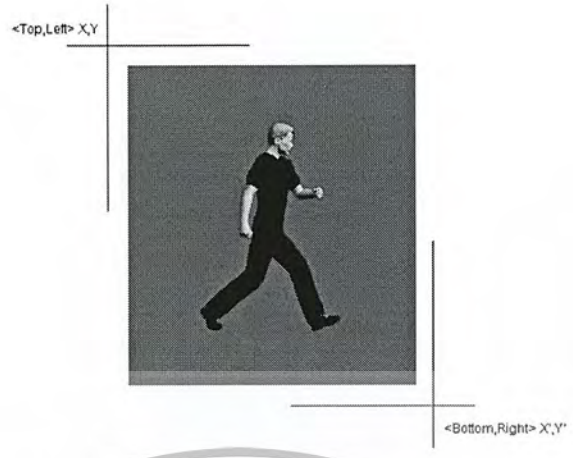
5.) การโจมตี

6.) การโจมตีด้วยท่าไม้ตาย

7.) การป้องกัน

โดยภาพที่ใช้จะเรียกว่าภาพสไปร์ท ซึ่งท่าทางแต่ละท่าจะมีประมาณ 16 เฟรมในหนึ่งบรรทัด จากซ้ายไปขวา โดยการเขียนโปรแกรมจะทำการเลือกภาพมาแสดงโดยใช้จุดคู่ลำดับ X, Y ที่มุมบนซ้ายของภาพ และจุดคู่ลำดับ X', Y' ที่มุมล่างขวาของภาพ เป็นตัวกำหนดภาพที่เลือก ซึ่งลักษณะนี้เมื่อนำมาเขียนโปรแกรม จะเป็นเทคนิคที่ใช้ในการทำภาพเคลื่อนไหว (Animation) ซึ่งอาศัยหลักการเหมือนฟิล์มภาพยนต์ คือ การนำเอาภาพนิ่งที่มีการเคลื่อนไหวติดต่อกันมาแสดงด้วยอัตราเร็วระดับหนึ่ง ทำให้เกิดการลวงตาเสมือนว่าภาพนั้นเคลื่อนไหวได้

รูปที่ 3.6 ตัวอย่างภาพสไปร์ทของตัวละคร



รูปที่ 3.7 การใช้จุดคู่ลำดับในการเลือกภาพจากสไปร์ทมาแสดงผล

รูปที่ 3.7 แสดงการใช้จุดคู่ลำดับในการเลือกภาพจาก Sprite มาแสดงผล โดยภาพที่ได้ออกมานี้จะรับเป็นรูปสี่เหลี่ยมซึ่งเป็นโครงสร้างข้อมูลชนิด RECT หรือ Rectangle นั่นเองซึ่งประกอบด้วยข้อมูลดังนี้

```
typedef struct _RECT
{
    long    top;    // เป็นค่าตำแหน่งพิกเซลบนสุดของสี่เหลี่ยม
    long    left;   // เป็นค่าตำแหน่งพิกเซลซ้ายสุดของสี่เหลี่ยม
    long    right;  // เป็นค่าตำแหน่งพิกเซลขวาสุดของสี่เหลี่ยม
    long    bottom;// เป็นค่าตำแหน่งพิกเซลล่างสุดของสี่เหลี่ยม
}RECT
```

ส่วนภาพฉากและรูปภาพต่าง ๆ ที่ใช้ภายในเกมก็จะใช้ภาพชนิด Bitmap โดยจะใช้ DirectDraw ช่วยในการดึงภาพเข้าไปแสดงผลในหน้าจอ

3.3 ขั้นตอนการเขียนโปรแกรมเกม

ส่วนใหญ่แล้วขั้นตอนต่างๆจะมีการทำงานเป็นเส้นตรง กล่าวคือเมื่อทำงานอย่างหนึ่งเสร็จก็จะทำงานอย่างอื่นต่อโดยไม่วนมาที่งานนั้นใหม่ แต่ขั้นตอนของการเขียนโปรแกรมนั้นมีการทำงานที่ต้องวนมาศึกษาและทำการเขียนโปรแกรม และหากไม่ผ่านการทดสอบแล้ว ก็ต้องทำการศึกษาเพิ่มเติมใหม่ การเขียนโปรแกรมจะสิ้นสุดได้ โปรแกรมในทุกส่วนจะต้องถูกเขียนและผ่านการทดสอบเรียบร้อยแล้ว ดังแผนภาพต่อไปนี้

รูปที่ 3.8 โฟลชาร์ตของขั้นตอนการเขียนโปรแกรม

โปรแกรมจะถูกแบ่งออกเป็นส่วนย่อยหลายส่วนดังนี้

- ส่วนของการแสดงผล
- ส่วนของการติดต่อกับอุปกรณ์รับข้อมูล
- ส่วนของการติดต่อกับการ์ดเสียงและการสร้างเสียง
- ส่วนของตัวละคร
- ส่วนของฉากต่าง ๆ

เหตุผลที่แบ่งโปรแกรมออกเป็นส่วนย่อย ก็เพื่อต้องการจัดการแบ่งหน้าที่การทำงานที่เป็นประเภทเดียวกัน ไว้ด้วยกัน ทำให้ง่ายต่อการตรวจสอบหรือแก้ไขข้อผิดพลาดที่อาจจะเกิดขึ้นได้ และยังง่ายต่อการเพิ่มเติมส่วนต่าง ๆ ได้ทีหลังอีกด้วย

3.3.1 ส่วนของการแสดงผลบนหน้าจอ

ในส่วนการแสดงผลนี้จะรวมฟังก์ชันการทำงานที่เกี่ยวกับการแสดงผล การติดต่อกับฮาร์ดแวร์ โดยผ่านทาง DirectDraw เพื่อสร้าง Surface ต่าง ๆ รวมถึงการทำภาพเคลื่อนไหวของตัวละคร

3.3.2 ส่วนของการติดต่อกับอุปกรณ์รับข้อมูลเข้า

ในส่วนการติดต่ออุปกรณ์รับข้อมูลจะรวมฟังก์ชันที่ทำการติดต่อกับอุปกรณ์อินพุตโดยผ่านทาง DirectInput โปรแกรมเกมที่พัฒนาขึ้นมานี้ใช้คีย์บอร์ดในการรับอินพุต

3.3.3 ส่วนของการติดต่อกับการ์ดเสียงและการสร้างเสียง

ในส่วนการติดต่อการ์ดเสียงและการสร้างเสียงนี้จะฟังก์ชันการทำงานที่ติดต่อกับการ์ดเสียง โดยผ่านทาง DirectSound และทำการแสดงเสียงออกทางลำโพงด้วยการโหลดไฟล์ WAV

3.3.4 ส่วนของตัวละคร

ส่วนของตัวละครนี้จะประกอบไปด้วยคลาส CActor ซึ่งมีฟังก์ชันการทำงานของตัวละคร ทั้งการเคลื่อนที่ไปทางขวา การเคลื่อนที่ไปทางซ้าย การก้ม การกระโดด การป้องกัน การโจมตี การโจมตีด้วยท่าไม้ตาย รวมถึงรายละเอียดปลีกย่อยอื่นๆ โครงสร้างของคลาส CActor มีดังนี้

```
class CActor
{
        private :
                int     ID ;
                int     Type ;
                int     Team ;
                int     HP ;
                int     AP ;
                int     CP ;
```

```
    int     DP ;
    int     framewidth;
    int     frame height;
    int     currentframe;
    int     currentrow;
    int     x;
    int     y;
    LPDIRECTDRAWSURFACE7   actor;
    RECT    bodyarea;
    RECT    weakarea;
    RECT    fightarea;
public :
    CActor( );
    ~CActor(void);
    RECT    GetPic(int howx, int howy, int which);
    void    LoadBmpAct(LPCSTR filename, int width, int height, int colorkey);
    void    Draw();
    void    DrawOne(int which, int row);
    void    DrawOne(int _x, int _y, int frame, int row);
    void    SetBodyArea(int x1, int y1, int x2, int y2);
    void    SetWeakArea(int x1, int y1, int x2, int y2);
    void    SetFightArea(int x1, int y1, int x2, int y2);
    BOOL    IsHit(RECT r);
    void    ReleaseActor();
```

```
        void    MoveRight(int step);
        void    MoveLeft(int step);
        void    MoveUp();
        void    MoveDown();
        void    Attack();
        void    Critical();
        void    Defense();
};
```

การทำงานของฟังก์ชันในคลาส CActor มีดังนี้

1.) CActor :: RECT GetPic(int howx, int howy, int which)

เมื่อเรียกใช้ฟังก์ชันนี้ จะเป็นอ่านภาพที่ละเฟรม

2.) CActor :: void LoadBmpAct(LPCSTR filename, int width, int height, int colorkey)

เมื่อเรียกใช้ฟังก์ชันนี้ จะเป็นการโหลดไฟล์ภาพตัวละคร filename ประเภท Bitmap ขนาดความกว้างและความสูงที่กำหนด มาเก็บไว้ใน Surface actor พร้อมทั้งใช้ฟังก์ชัน SetColorKey() กำหนดสีพื้นหลังของสไปรท์ให้โปร่งใสด้วย

3.) CActor :: void Draw()

ฟังก์ชันทำหน้าที่วาดภาพใด ๆ หนึ่งภาพ คือฟังก์ชันจะวาดภาพในตำแหน่ง x, y ในเฟรมและแถวปังจจุบันของสไปรท์

4.) CActor :: void DrawOne(int which, int row)

ฟังก์ชันทำหน้าที่วาดภาพใด ๆ หนึ่งภาพโดยระบุลำดับที่ของเฟรมในแถวนั้น

5.) CActor :: void DrawOne(int _x, int _y, int frame, int row)

ฟังก์ชันทำหน้าที่วาดภาพใด ๆ หนึ่งภาพโดยระบุตำแหน่งที่จะแสดงภาพตัวละครบนจอภาพและระบุลำดับที่ของเฟรมในแถวนั้น

6.) CActor :: void SetBodyArea(int x1, int y1, int x2, int y2)

ฟังก์ชันทำหน้าที่กำหนดขอนาดของ RECT ที่เป็น Body Area โดยระบุตำแหน่ง top = x1, left = y1, bottom = x2 และ right = y2

7.) CActor :: void SetWeakArea(int x1, int y1, int x2, int y2)

ฟังก์ชันทำหน้าที่กำหนดขอนาดของ RECT ที่เป็น Weak Area โดยระบุตำแหน่ง top = x1, left = y1, bottom = x2 และ right = y2

8.) CActor :: void SetFightArea(int x1, int y1, int x2, int y2)

ฟังก์ชันทำหน้าที่กำหนดขอนาดของ RECT ที่เป็น Fight Area โดยระบุตำแหน่ง top = x1, left = y1, bottom = x2 และ right = y2

9.) CActor :: BOOL IsHit(RECT r)

ฟังก์ชันทำหน้าที่ตรวจสอบว่า Fight Area ชนกับ RECT r อื่นหรือไม่ ฟังก์ชันคืนค่าจริงถ้าเกิดการชนกัน

10.) CActor :: void ReleaseActor()

ฟังก์ชันทำการคืนหน่วยความจำให้แก่ระบบ

11.) CActor :: void MoveRight()

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะเคลื่อนที่ไปทางขวา

12.) CActor :: void MoveLeft()

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะเคลื่อนที่ไปทางซ้าย

13.) CActor :: void MoveUp(void)

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะกระโดดขึ้นสูงจากพื้น สามารถเลี่ยงค่าความเสียหายจากการโจมตีของฝ่ายตรงข้ามได้

14.) CActor :: void MoveDown(void)

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะทำการก้มต่ำลง สามารถเลี่ยงค่าความเสียหายจาการโจมตีของฝ่ายตรงข้ามได้

15.) CActor :: void Attack(void)

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะทำการโจมตีผู้เล่นฝ่ายตรงข้าม ให้ได้รับความเสียหาย โดยลดค่าพลังชีวิตของผู้ถูกโจมตี

16.) CActor :: void CritAttack(void)

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะทำการโจมตีผู้เล่นฝ่ายตรงข้ามด้วยท่าไม้ตาย ให้ได้รับความเสียหายอย่างหนัก โดยลดค่าพลังชีวิตของผู้ถูกโจมตี

17.) CActor :: void Defense(void)

เมื่อเรียกใช้ฟังก์ชันนี้ ตัวละครจะทำการป้องกันการโจมตีของผู้เล่นฝ่ายตรงข้าม เพื่อให้ได้รับค่าความเสียหายน้อยลง

3.3.5 **ส่วนของฉากต่าง ๆ**

ส่วนของฉากนี้จะประกอบด้วยคลาส CMap ภายในคลาสจะมีฟังก์ชันการแสดงแผนที่สนามประลอง โครงสร้างของคลาส CMap มีดังนี้

```
class CMap
{
        private :
                int     width ;
                int     height ;
                LPDIRECTDRAWSURFACE7   mapsurf;
        public:
                CMap() ;
                ~CMap( ) ;
                LPDIRECTDRAWSURFACE7   InitMap(void) ;
                LPDIRECTDRAWSURFACE7   DrawMap(void) ;
                Void    ReleaseMap();
};
```

การทำงานของฟังก์ชันในคลาส CMap มีดังนี้

1.) CMap :: InitMap(void)

ฟังก์ชันนี้จะทำการสร้าง Off Screen Surface ไว้เป็นที่เก็บฉาก เพื่อช่วยแสดงฉากบนจอภาพ แล้วนำรูปฉากนั้นมาเก็บไว้ใน Off Screen Surface ที่ได้สร้างไว้แล้ว

2.) Cmap :: DrawMap(void)

ฟังก์ชันนี้จะแสดงฉากออกทางจอแสดงผล

3.) Cmap :: ReleaseMap(void)

ฟังก์ชันทำหน้าที่คืนหน่วยความจำให้แก่ระบบ

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

การทดลองและการวิเคราะห์ปัญหาที่กล่าวถึงในบทนี้ จะเป็นขั้นตอนการทดสอบและแสดงผลการทดสอบในแต่ละขั้น โดยผลการทดสอบนี้จะถูกนำไปวิเคราะห์ถึงสาเหตุของการเกิดปัญหาและแนวทางในการพัฒนาต่อไปในอนาคต โดยทางผู้จัดทำหวังว่าจะเป็นประโยชน์แก่ผู้ที่จะนำไปศึกษาเพื่อพัฒนาต่อ และเพื่อแสดงให้เห็นถึงปัญหา และข้อดีข้อเสียของปัญหานี้ โดยที่ผู้ศึกษาไม่จำเป็นต้องทำการทดสอบเพื่อหาปัญหา แต่ยังสามารถนำโครงงานนี้ไปพัฒนาต่อได้อีกด้วย

คุณสมบัติของระบบที่นำมาทดสอบ

- ระบบปฏิบัติการ Microsoft Windows XP
- ติดตั้งชุดคำสั่ง Microsoft DirectX SDK เวอร์ชัน 8.0
- เครื่องคอมพิวเตอร์ที่มีซีพียูความเร็ว 1.8 GHz
- หน่วยความจำหลักขนาด 256 Mbytes

ขั้นตอนการดำเนินการทดสอบ

- ติดตั้งโปรแกรมและซอฟต์แวร์ที่จำเป็นต้องใช้การประมวลผล
- รันและประมวลผลภายใต้ระบบที่กำหนด
- ตรวจสอบการใช้คำสั่งของตัวละครและการทำงานของตัวละคร
- ตรวจสอบการตอบโต้ของฝ่ายคอมพิวเตอร์ขณะเล่นเกม
- ตรวจสอบการการหาความผิดพลาดของเกม

การประเมินผล

- หลังจากติดตั้งโปรแกรมแล้ว ผู้เล่นสามารถใช้โปรแกรมได้ทันที
- การประมวลผลของโปรแกรมจะต้องทำภายใต้ความเร็วในระดับที่ยอมรับได้
- การตอบโต้ของคอมพิวเตอร์จะต้องเป็นไปตามที่นักเขียนโปรแกรมกำหนดไว้
- การใช้คำสั่งของตัวละครเป็นไปตามรูปแบบที่นักเขียนโปรแกรมกำหนดไว้
- จำนวนความผิดพลาดของเกมที่เกิดขึ้นต้องไม่มีเลย หรือมีน้อยที่สุด

ต่อไปจะกล่าวถึงรายละเอียดของขั้นตอนการทดสอบและผลการทดสอบที่ได้

4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและซอฟต์แวร์ที่จำเป็น

- ติดตั้งโปรแกรมโดยดับเบิ้ลคลิกที่ไฟล์ “SETUP.EXE”
- โปรแกรมและซอฟต์แวร์จะถูกติดตั้งลงบนเครื่องคอมพิวเตอร์ของผู้เล่น
- ผู้ใช้รันโปรแกรมได้โดยดับเบิ้ลคลิกที่ไฟล์ “Fighting2.EXE”

ผลการทดสอบ

- ติดตั้งโปรแกรมและซอฟต์แวร์ที่จำเป็นลงบนเครื่องคอมพิวเตอร์ของผู้เล่นได้
- ผู้เล่นสามารถรันโปรแกรมได้

4.2 ขั้นตอนการทดสอบการรันและประมวลผลเกมภายใต้ระบบที่กำหนด

รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนด โดยดับเบิ้ลคลิกที่ไฟล์ “Fighting2.EXE” จอภาพปรากฏดังนี้

รูปที่ 4.1 ภาพหน้าจอเมนูหลักของเกม

จากรูปจะเห็นได้ว่ามีสี่ทางเลือกดังนี้

1. Single เริ่มเล่นเกมแบบสู้กับคอมพิวเตอร์
2. Versus เริ่มเล่นเกมแบบสู้กันระหว่างผู้เล่น
3. Credit กล่าวถึงผู้ที่อยู่เบื้องหลังการพัฒนาเกม
4. Exit ออกจากเกม

ผลการทดสอบ

- สามารถรันโปรแกรมได้บนระบบที่กำหนด
- ความเร็วในการแสดงภาพหน้าจอแรกอยู่ในขั้นดี
- สามารถเลือกรายการที่แสดงได้ตามความต้องการของผู้เล่น

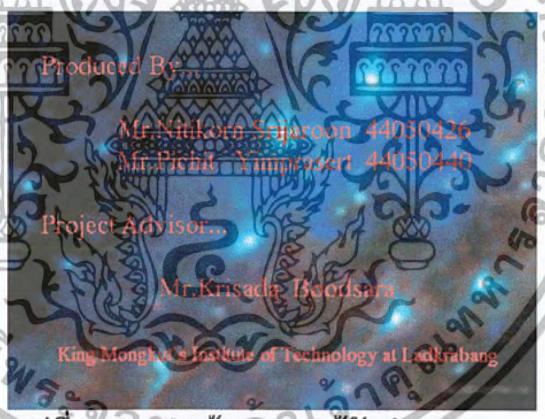
4.2.1 การทดสอบทางเลือก Credit

ขั้นตอนนี้เป็นการทดสอบซึ่งเริ่มต้นจากการเลือกทางเลือก “Credit” จะได้หน้าจอดังนี้

รูปที่ 4.2 ภาพหน้าจอการเลือกเมนู Credit

ผลการทดสอบ

- แสดงหน้าจอเบื้องหลังการพัฒนาได้ถูกต้องดังรูปที่ 4.3



รูปที่ 4.3 ภาพหน้าจอคณะผู้จัดทำ

- สามารถย้อนกลับมาหน้ารายการหลักได้โดยการกดปุ่ม Enter

4.2.2 การทดสอบทางเลือก Exit

ขั้นตอนนี้เป็นการทดสอบซึ่งเริ่มต้นจากการเลือกทางเลือก “Exit” จะได้หน้าจอดังนี้

รูปที่ 4.4 ภาพหน้าจอเมนู Exit

ผลการทดสอบ

- ตัวโปรแกรมปิด
- โปรแกรมคืนค่าทรัพยากรให้สู่หน่วยความจำของระบบ
- ความเร็วที่ได้จากการออกจากเกมอยู่ในขั้นดี

4.2.3 การทดสอบทางเลือก Single

ขั้นตอนนี้เป็นการทดสอบซึ่งเริ่มต้นจากการเลือกทางเลือก “Single” จะได้หน้าจอดังนี้

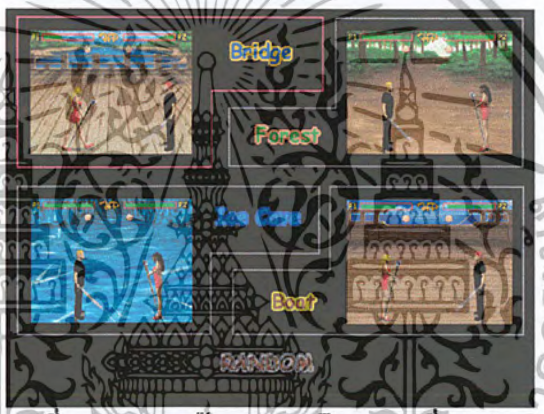
รูปที่ 4.5 ภาพหน้าจอการเลือกเมนู Single

ผลการทดสอบ

- โปรแกรมแสดงภาพหน้าจอเลือกตัวละครที่จะเล่นได้ดังรูปที่ 4.6

รูปที่ 4.6 ภาพหน้าจอการเลือกตัวละครที่จะเล่น
ในกรณีที่เลือก Single

- เมื่อเลือกตัวละครที่ต้องการแล้วหน้าจอจะเข้าสู่การเลือกฉากดังรูปที่ 4.7



รูปที่ 4.7 ภาพหน้าจอการเลือกฉากที่จะเล่น

- เมื่อเลือกฉากที่ต้องการแล้วหน้าจอจะเข้าสู่การแสดงตัวละครดังรูปที่ 4.8

รูปที่ 4.8 ภาพหน้าจอแสดงตัวละครก่อนเข้าสู่เกม

- เมื่อกด Enter ก็จะเข้าสู่เกม

4.2.4 การทดสอบทางเลือก VS

ขั้นตอนนี้เป็นการทดสอบซึ่งเริ่มต้นจากการเลือกทางเลือก “VS” จะได้หน้าจอดังนี้

รูปที่ 4.9 ภาพหน้าจอการเลือกเมนู VS

ผลการทดสอบ

- โปรแกรมแสดงภาพหน้าจอเลือกตัวละครที่จะเล่นได้ดังรูปที่ 4.10

รูปที่ 4.10 ภาพหน้าจอการเลือกตัวละครที่จะเล่นในกรณีที่เลือก VS

- สามารถเลือกตัวละครที่ 1 ได้ตามความต้องการด้วยการกด Enter

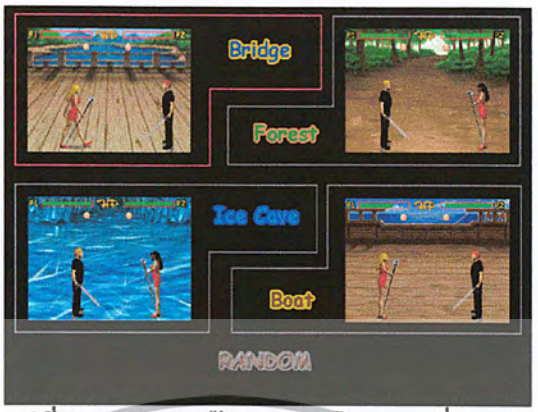
รูปที่ 4.11 ภาพหน้าจอตัวละครที่ 1 ถูกเลือกแล้ว

รูปที่ 4.12 ภาพหน้าจอตัวละครที่ 2 ถูกเลือกแล้ว

- ในกรณีที่ต้องการยกเลิกตัวละครที่เลือกไปแล้วให้กด Space bar

รูปที่ 4.13 ภาพหน้าจอการยกเลิกตัวละคร

- เมื่อเลือกตัวละครที่ต้องการแล้วหน้าจอจะเข้าสู่การเลือกฉาก



รูปที่ 4.14 ภาพหน้าจอการเลือกฉากที่จะเล่น

- เมื่อเลือกฉากที่ต้องการแล้วหน้าจอจะเข้าสู่การแสดงตัวละครดังรูปที่ 4.15

รูปที่ 4.15 ภาพหน้าจอแสดงตัวละครก่อนเข้าสู่เกม

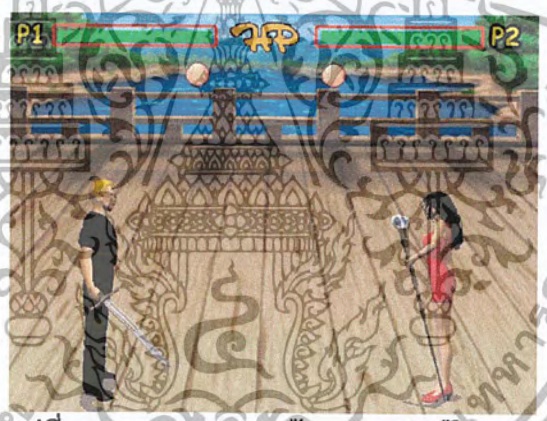
- เมื่อกด Enter ก็จะเข้าสู่เกม

4.3 ขั้นตอนการทดสอบฉาก

4.3.1 การเลือกฉาก Bridge

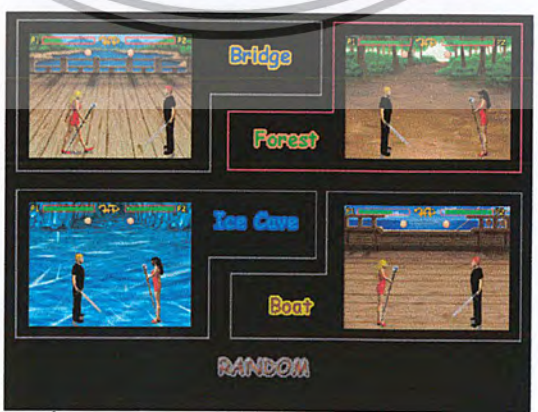
รูปที่ 4.16 ภาพหน้าจอการเลือกฉาก Bridge

ผลการทดสอบดังรูปที่ 4.17



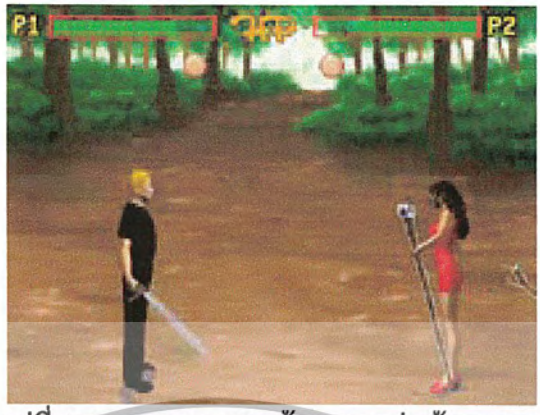
รูปที่ 4.17 แสดงภาพหน้าจอการต่อสู้ในฉาก Bridge

4.3.2 การเลือกฉาก Forest



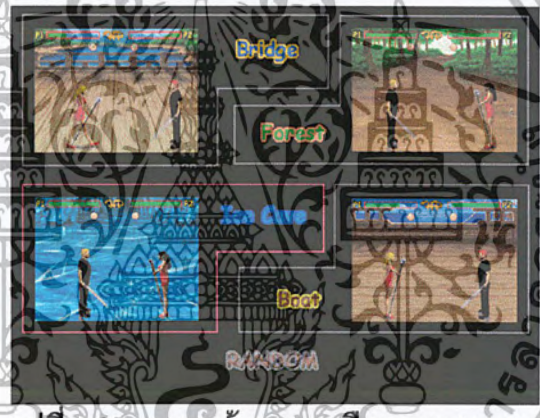
รูปที่ 4.18 ภาพหน้าจอการเลือกฉาก Forest

ผลการทดสอบดังรูปที่ 4.19



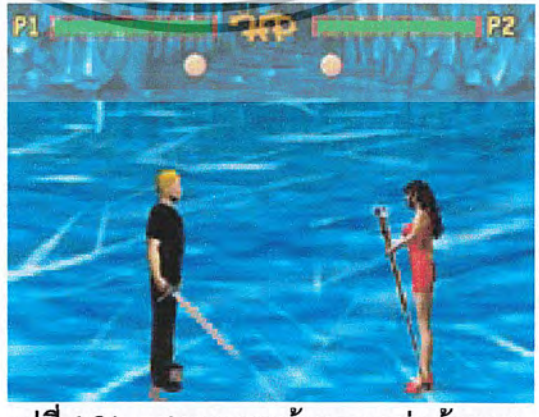
รูปที่ 4.19 แสดงภาพหน้าจอการต่อสู้ฉาก Forest

4.3.3 การเลือกฉาก Ice cave



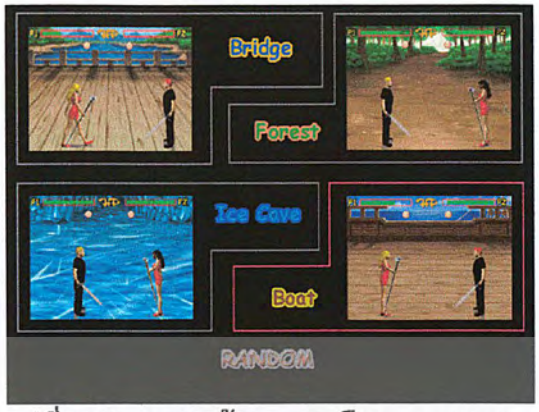
รูปที่ 4.20 ภาพหน้าจอการเลือกฉาก Ice Cave

ผลการทดสอบดังรูปที่ 4.21



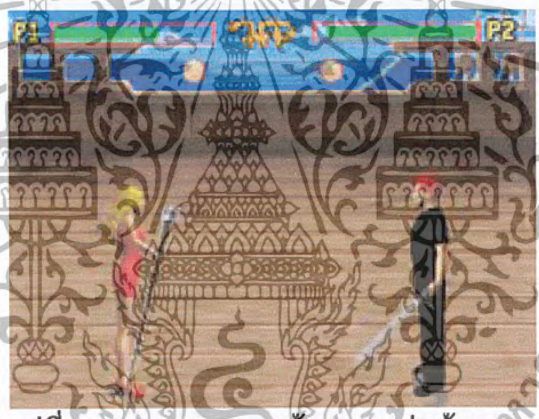
รูปที่ 4.21 แสดงภาพหน้าจอการต่อสู้ฉาก Ice Cave

4.3.4 การเลือกฉาก Boat



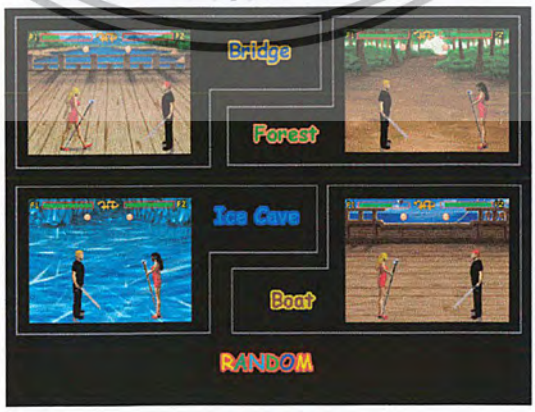
รูปที่ 4.22 ภาพหน้าจอการเลือกฉาก Boat

ผลทดสอบดังรูปที่ 4.23



รูปที่ 4.23 แสดงภาพหน้าจอการต่อสู้ฉาก Boat

4.3.5 การเลือกฉากแบบ Random



รูปที่ 4.24 ภาพหน้าจอการเลือกฉากแบบ Random

ผลการทดสอบจะเป็นการสุ่มฉากขึ้นมา 1 ฉาก จากฉากทั้งหมด 4 ฉากที่ได้แสดงไปแล้ว

4.4 ขั้นตอนการทดสอบการใช้คำสั่งของตัวละครและการทำงานของตัวละคร

การทดสอบจะแบ่งเป็น 2 ส่วนคือ

- การทดสอบด้วยผู้เล่น 1 คน
- การทดสอบด้วยผู้เล่น 2 คน

4.4.1 การทดสอบเกมด้วยผู้เล่น 1 คน

การทดสอบการใช้คำสั่งของตัวละคร วิธีการที่ใช้ทดสอบ และผลการทดสอบแสดงได้ดังตารางที่ 4.1

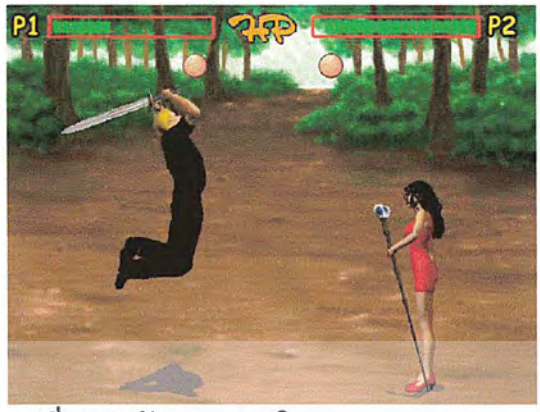
ตารางที่ 4.1 แสดงวิธีการทดสอบและผลการทดสอบของผู้เล่น 1 คน

การทดสอบ	วิธีทดสอบ	ผลการทดสอบ
ตรวจสอบการเคลื่อนที่ของตัวละคร	กดปุ่ม A W S D	ตัวละครจะเคลื่อนที่ไปทางซ้าย(A) กระโดด(W) ก้ม(S) และเคลื่อนที่ไปทางขวา(D) ตามลำดับ
ตรวจสอบการโจมตี	กดปุ่ม I	ตัวละครจะทำการโจมตี
ตรวจสอบการโจมตีด้วยท่าไม้ตาย	กดปุ่ม U	ตัวละครจะทำการโจมตีด้วยท่าไม้ตาย
ตรวจสอบการป้องกัน	กดปุ่ม O	ตัวละครจะทำการป้องกัน

ผลการทดสอบการกดปุ่ม D เป็นดังรูปที่ 4.25

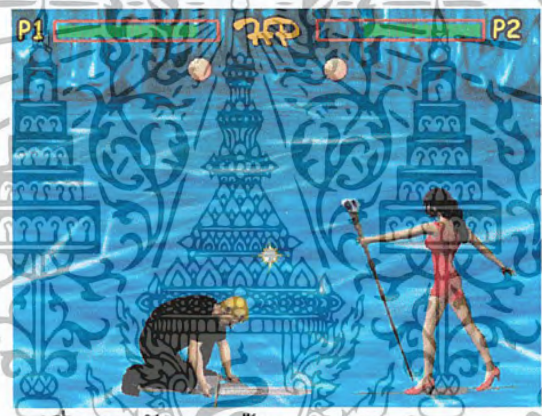
รูปที่ 4.25 ตัวละครเดินไปข้างหน้า

ผลการทดสอบการกดปุ่ม W เป็นดังรูปที่ 4.26



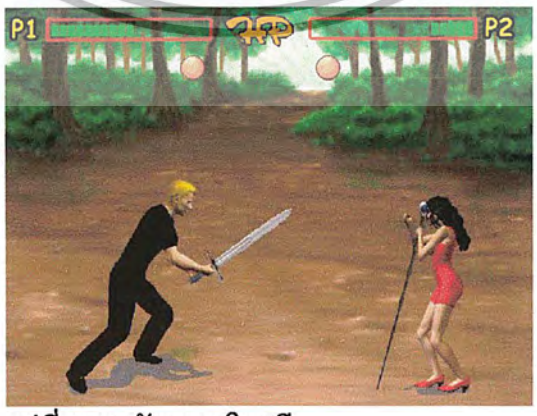
รูปที่ 4.26 ตัวละครกระโดด

ผลการทดสอบการกดปุ่ม S เป็นดังรูปที่ 4.27



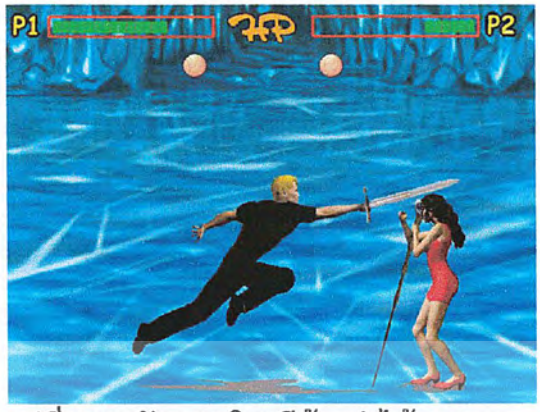
รูปที่ 4.27 ตัวละครก้ม

ผลการทดสอบการกดปุ่ม I เป็นดังรูปที่ 4.28



รูปที่ 4.28 ตัวละครโจมตี

ผลการทดสอบการกดปุ่ม U เป็นดังรูปที่ 4.29



รูปที่ 4.29 ตัวละครโจมตีด้วยท่าไม้ตาย

ผลการทดสอบการกดปุ่ม O เป็นดังรูปที่ 4.30

รูปที่ 4.30 ตัวละครป้องกัน

4.4.2 การทดสอบเกมด้วยผู้เล่น 2 คน

การทดสอบจะแบ่งเป็น 2 ส่วนคือ

- การทดสอบของผู้เล่นคนที่ 1
- การทดสอบของผู้เล่นคนที่ 2

การทดสอบการใช้คำสั่งของตัวละคร วิธีการที่ใช้ทดสอบ และผลการทดสอบของผู้เล่นคนที่ 1 แสดงได้ดังตารางที่ 4.2

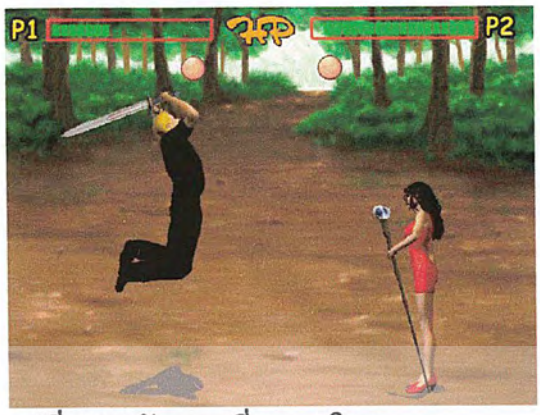
ตารางที่ 4.2 แสดงวิธีการทดสอบและผลการทดสอบของผู้เล่นคนที่ 1

การทดสอบ	วิธีทดสอบ	ผลการทดสอบ
ตรวจสอบการเคลื่อนที่ของตัวละคร	กดปุ่ม A W S D	ตัวละครจะเคลื่อนที่ไปทางซ้าย(A) กระโดด(W) ก้ม(S) และเคลื่อนที่ไปทางขวา(D) ตามลำดับ
ตรวจสอบการโจมตี	กดปุ่ม I	ตัวละครจะทำการโจมตี
ตรวจสอบการโจมตีด้วยท่าไม้ตาย	กดปุ่ม U	ตัวละครจะทำการโจมตีด้วยท่าไม้ตาย
ตรวจสอบการป้องกัน	กดปุ่ม O	ตัวละครจะทำการป้องกัน

ผลการทดสอบการกดปุ่ม D เป็นดังรูปที่ 4.31

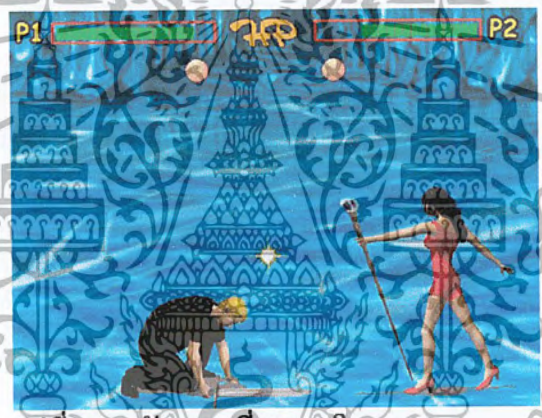
รูปที่ 4.31 ตัวละครที่ 1 เดินไปด้านขวา

ผลการทดสอบการกดปุ่ม W เป็นดังรูปที่ 4.32



รูปที่ 4.32 ตัวละครที่ 1 กระโดด

ผลการทดสอบการกดปุ่ม S เป็นดังรูปที่ 4.33

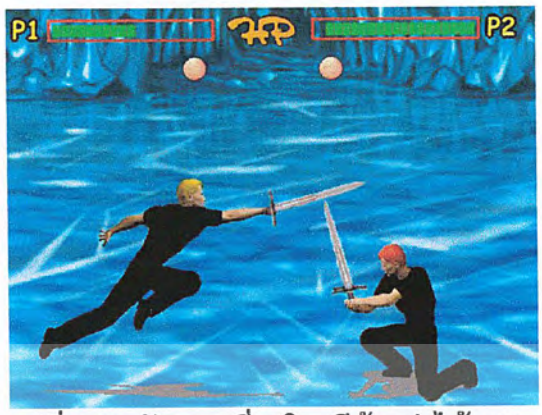


รูปที่ 4.33 ตัวละครที่ 1 กระโดด

ผลการทดสอบการกดปุ่ม I เป็นดังรูปที่ 4.34

รูปที่ 4.34 ตัวละครที่ 1 โจมตี

ผลการทดสอบการกดปุ่ม U เป็นดังรูปที่ 4.35



รูปที่ 4.35 ตัวละครที่ 1 โจมตีด้วยท่าไม้ตาย

ผลการทดสอบการกดปุ่ม O เป็นดังรูปที่ 4.36

รูปที่ 4.36 ตัวละครที่ 1 ป้องกัน

การทดสอบการใช้คำสั่งของตัวละคร วิธีการที่ใช้ทดสอบ และผลการทดสอบของผู้เล่นคนที่ 2 แสดงได้ดังตารางที่ 4.3

ตารางที่ 4.3 แสดงวิธีการทดสอบและผลการทดสอบของผู้เล่นคนที่ 2

การทดสอบ	วิธีทดสอบ	ผลการทดสอบ
ตรวจสอบการเคลื่อนที่ของตัวละคร	กดปุ่มลูกศรซ้าย ขึ้น ลง ขวา	ตัวละครจะเคลื่อนที่ไปทางซ้าย (ลูกศรซ้าย) กระโดด(ลูกศรขึ้น) ก้ม(ลูกศรลง) และเคลื่อนที่ไปทางขวา(ลูกศรขวา) ตามลำดับ
ตรวจสอบการโจมตี	กดปุ่ม 2	ตัวละครจะทำการโจมตี
ตรวจสอบการโจมตีด้วยท่าไม้ตาย	กดปุ่ม 1	ตัวละครจะทำการโจมตีด้วยท่าไม้ตาย
ตรวจสอบการป้องกัน	กดปุ่ม 3	ตัวละครจะทำการป้องกัน

ผลการทดสอบการกดปุ่ม ลูกศรซ้าย เป็นดังรูปที่ 4.37

รูปที่ 4.37 ตัวละครที่ 2 เดินไปข้างหน้า

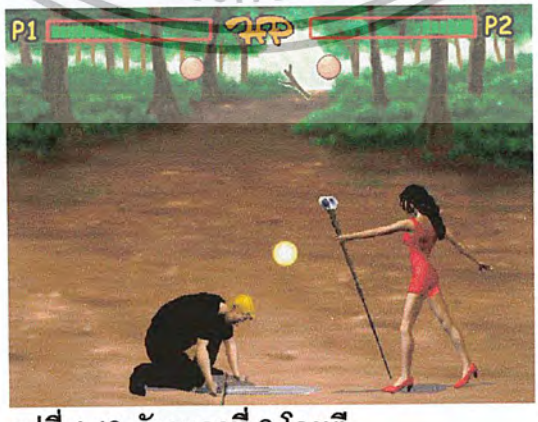
ผลการทดสอบการกดปุ่ม ลูกศรขึ้น เป็นดังรูปที่ 4.38

รูปที่ 4.38 ตัวละครที่ 2 กระโดด

ผลการทดสอบการกดปุ่ม ลูกศรลง เป็นดังรูปที่ 4.39

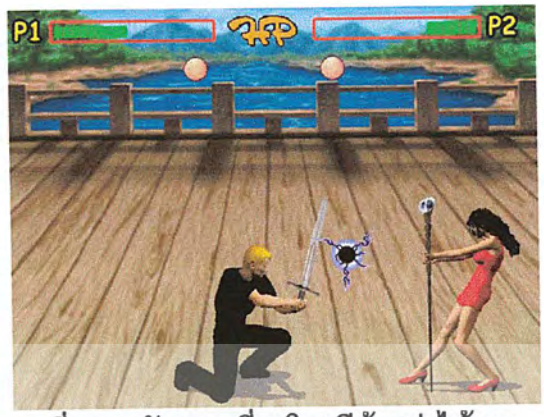
รูปที่ 4.39 ตัวละครที่ 2 ก้ม

ผลการทดสอบการกดปุ่ม 2 เป็นดังรูปที่ 4.40



รูปที่ 4.40 ตัวละครที่ 2 โจมตี

ผลการทดสอบการกดปุ่ม 1 เป็นดังรูปที่ 4.41



รูปที่ 4.41 ตัวละครที่ 2 โจมตีด้วยท่าไม้ตาย

ผลการทดสอบการกดปุ่ม 3 เป็นดังรูปที่ 4.42

รูปที่ 4.42 ตัวละครที่ 2 ป้องกัน

4.5 ขั้นตอนการทดสอบการตอบโต้ของคอมพิวเตอร์

- โจมตีตัวละครฝ่ายคอมพิวเตอร์
- โจมตีตัวละครฝ่ายคอมพิวเตอร์ด้วยท่าไม้ตาย

ผลการทดสอบ

- ตัวละครฝ่ายคอมพิวเตอร์ทำการตอบโต้ ป้องกัน กระโดดหรือก้มเพื่อหลบหลีกการโจมตีเมื่อถูกโจมตี

ผลการทดสอบคอมพิวเตอร์ป้องกันจากการถูกโจมตี เป็นดังรูปที่ 4.43ก และ 4.43ข

รูปที่ 4.43ก คอมพิวเตอร์ป้องกันจากการถูกโจมตี

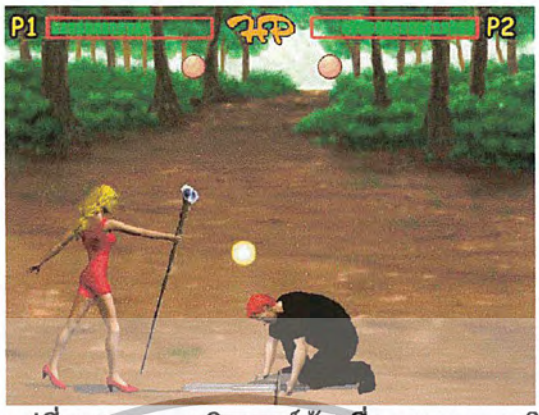
รูปที่ 4.43ข คอมพิวเตอร์ป้องกันจากการถูกโจมตี

ผลการทดสอบคอมพิวเตอร์ป้องกันจากการถูกโจมตีด้วยท่าไม้ตาย เป็นดังรูปที่ 4.44ก และ 4.44ข

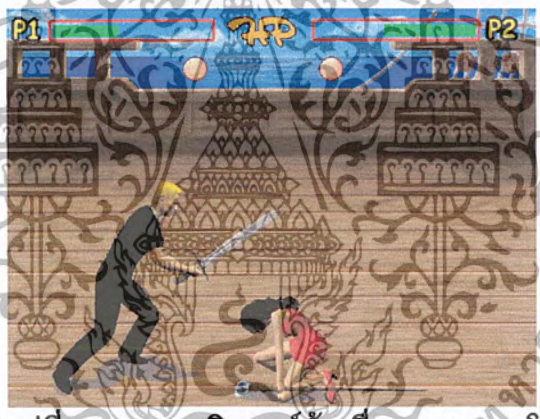
รูปที่ 4.44ก คอมพิวเตอร์ป้องกันจากการถูกโจมตีด้วยท่าไม้ตาย

รูปที่ 4.44ข คอมพิวเตอร์ป้องกันจากการถูกโจมตีด้วยท่าไม้ตาย

ผลการทดสอบคอมพิวเตอร์ก้มเพื่อหลบการถูกโจมตี เป็นดังรูปที่ 4.45ก และ 4.45ข

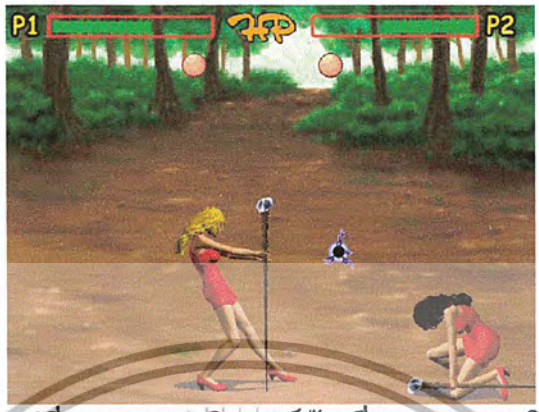


รูปที่ 4.45ก คอมพิวเตอร์ก้มเพื่อหลบการถูกโจมตี



รูปที่ 4.45ข คอมพิวเตอร์ก้มเพื่อหลบการถูกโจมตี

ผลการทดสอบคอมพิวเตอร์ก้มเพื่อหลบการถูกโจมตีด้วยท่าไม้ตาย เป็นดังรูปที่ 4.46ก และ 4.46ข

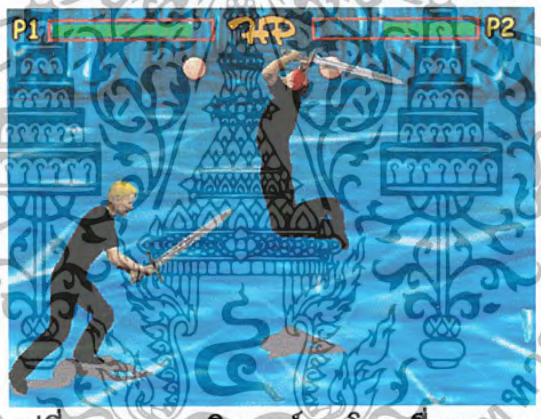


รูปที่ 4.46ก คอมพิวเตอร์ก้มเพื่อหลบการถูกโจมตีด้วยท่าไม้ตาย

รูปที่ 4.46ข คอมพิวเตอร์ก้มเพื่อหลบการถูกโจมตีด้วยท่าไม้ตาย

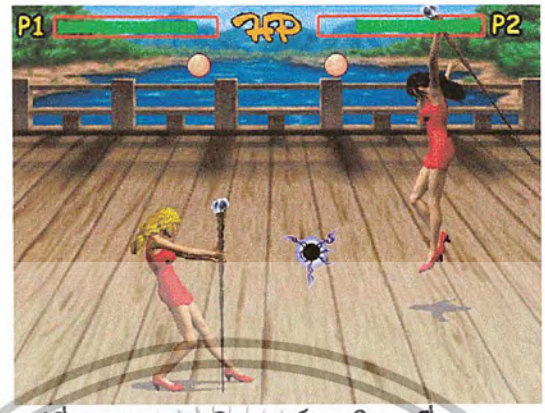
ผลการทดสอบคอมพิวเตอร์กระโดดเพื่อหลบการถูกโจมตี เป็นดังรูปที่ 4.47ก และ 4.47ข

รูปที่ 4.47ก คอมพิวเตอร์กระโดดเพื่อหลบการถูกโจมตี



รูปที่ 4.47ข คอมพิวเตอร์กระโดดเพื่อหลบการถูกโจมตี

ผลการทดสอบคอมพิวเตอร์กระโดดเพื่อหลบการถูกโจมตีด้วยท่าไม้ตาย เป็นดังรูปที่ 4.48ก และ 4.48ข



รูปที่ 4.48ก คอมพิวเตอร์กระโดดเพื่อหลบการถูกโจมตีด้วยท่าไม้ตาย

รูปที่ 4.48ข คอมพิวเตอร์กระโดดเพื่อหลบการถูกโจมตีด้วยท่าไม้ตาย

4.6 ขั้นตอนการทดสอบการหาข้อผิดพลาดของโปรแกรมเกม

ในขั้นตอนการทดสอบเพื่อหาข้อผิดพลาดจะไม่มีรูปแบบที่ชัดเจน แต่จะทำการทดสอบโดยการเล่นเกมหลาย ๆ รอบ เพื่อหาข้อผิดพลาดและจากการทดสอบพบว่า ข้อผิดพลาดที่พบจะได้แก่

- การตอบโต้ของคอมพิวเตอร์ ยังพบปัญหาเมื่อผู้เล่นทำการตอบโต้ตอบด้วย
- การแสดงท่าทางของตัวละคร ยังมีข้อผิดพลาด

4.7 ปัญหาและการวิเคราะห์

จากผลการทดสอบโปรแกรมทำให้ทราบถึงปัญหาดังนี้

4.7.1 ความล่าช้าในการประมวลผล

สาเหตุของความช้าในการประมวลผลอาจเนื่องมาจากการเขียนโปรแกรมที่ไม่กระชับ และมีการวนลูปตรวจสอบค่ามากเกินไป

4.7.2 การตอบโต้ของฝ่ายคอมพิวเตอร์

ปัญหาที่สำคัญอีกปัญหาหนึ่งก็คือ การตอบโต้ของฝ่ายคอมพิวเตอร์ซึ่งยังทำได้ไม่สมจริงนัก เพราะมีการสุ่มการกระทำ ทำให้ง่ายต่อการถูกโจมตีและแพ้ให้กับผู้เล่นในที่สุด

สาเหตุของปัญหาอาจมาจากการที่ยังเก็บรวบรวมข้อมูลด้านกลยุทธ์ได้น้อยจนเกินไป ทำให้ไม่สามารถตอบโต้ฝ่ายผู้เล่นได้ดีนัก และขั้นตอนการเขียนโปรแกรมที่มีความซับซ้อน

4.7.3 การเคลื่อนไหวของตัวละคร

เนื่องจากปัญหาพิเศษที่ทำขึ้นนี้ ใช้ภาพเคลื่อนไหวเพียงแค่ 16 เฟรม ทำให้ภาพเคลื่อนไหวของตัวละครยังไม่สมจริงนัก แต่เนื่องจากทางผู้ศึกษาต้องการใช้ทรัพยากรในตัวโปรแกรมให้น้อยที่สุดจึงได้กำหนดเพียงแค่ 16 เฟรมและยังเพื่อต้องการให้การประมวลผลทำได้รวดเร็ว

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

การทำงานโครงงานนี้มีการวางแผนขั้นตอนการทำงานเป็นอย่างดี แต่ถึงแม้จะได้วางแผนแต่ละขั้นตอนไว้อย่างดี ก็ยังมีปัญหากับขั้นตอนบางขั้นตอน ซึ่งบางขั้นตอนเกิดจากการศึกษาการใช้ซอฟต์แวร์กราฟฟิกใหม่ ๆ เนื่องจากต้องใช้ประสบการณ์และระยะเวลาพอสมควร ถึงจะเข้าใจและสามารถทำงานกับซอฟต์แวร์แต่ละตัวได้เป็นอย่างดี บางขั้นตอนเกิดจากความล่าช้าในการคิดระเบียบวิธีการเขียนโปรแกรม ซึ่งต้องศึกษาจากบทความภาษาอังกฤษตามโฮมเพจต่าง ๆ

โดยขั้นตอนต่าง ๆ สรุปได้ดังนี้

- ขั้นตอนการออกแบบระบบเกม
- ขั้นตอนการเขียนโปรแกรมเกม
- ขั้นตอนการสร้างตัวละครและภาพต่าง ๆ

5.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบเป็นขั้นตอนที่รวดเร็วที่สุด เนื่องมาจากผู้วิจัยมีความคิดต่าง ๆ ของแต่ละคนไว้อยู่แล้ว ซึ่งเกิดจากประสบการณ์การเคยเล่นเกมต่าง ๆ มาก่อน และมีแนวคิดของเกมที่ตัวเองชื่นชอบไว้ จึงนำมารวบรวมและสรุปเป็นรูปแบบของเกมในงานวิจัย

5.2 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้สามารถทำควบคู่ไปได้กับขั้นตอนการสร้างตัวละครและภาพต่าง ๆ โดยขั้นตอนนี้ใช้เวลานานที่สุดในงานวิจัย เนื่องจากเป็นเรื่องใหม่ ซึ่งผู้วิจัยต้องทำการศึกษานานพอสมควร แล้วจึงลงมือเขียนโปรแกรม การทำงานในส่วนนี้มีซอฟท์แวร์ที่ใช้ในการพัฒนาดังนี้

1.) Microsoft Visual C++ เวอร์ชั่น 6.0

2.) Microsoft DirectX SDK เวอร์ชั่น 8.0

5.3 ขั้นตอนการสร้างตัวละครและภาพต่าง ๆ

ขั้นตอนนี้เป็นขั้นตอนที่ใช้ระยะเวลาหลังจากทำการเขียนโปรแกรมไปได้ระยะหนึ่ง เนื่องจากการสร้างตัวละครเป็นงานที่ใช้เวลานาน และปริมาณมาก ปัญหาส่วนใหญ่อยู่ที่การใช้งานซอฟท์แวร์

กราฟฟิกต่างๆ ซึ่งต้องใช้เวลาศึกษาพอสมควร แต่หลังจากได้ศึกษาและทำความเคยชินกับการทำงานแล้ว ทำให้การทำงานเป็นไปอย่างราบรื่น ขั้นตอนนี้ใช้โปรแกรมประยุกต์ดังนี้

1.) Poser เวอร์ชัน 5.0

2.) Adobe Photoshop เวอร์ชัน 7.0

Poser เป็นซอฟท์แวร์ที่ใช้ออกแบบภาพสามมิติต่าง ๆ ที่ใช้ในเกมทั้งหมด โดย export ออกมาเป็นภาพชนิด Bitmap

Adobe Photoshop เป็นซอฟท์แวร์ใช้สำหรับการตกแต่งภาพทุกชนิด เรานำมาใช้ตกแต่งและตัดต่อภาพตัวละคร และภาพฉากที่แตกต่างกัน

สรุปแล้วในการทำงานวิจัยเพื่อสร้างเกมโดยตอบโต้กับคอมพิวเตอร์, การทำงานต่าง ๆ ,ปัญหาที่เกิดขึ้น ตลอดจนการได้ติดต่อกับผู้ผลิตเกมในต่างประเทศเพื่อทำการศึกษานี้ ทำให้ผู้วิจัยทราบถึงแนวทางในการผลิตเกมในตลาดซอฟต์แวร์ไทย เกมในปัจจุบันนั้นบางเกมคนไทยเป็นผู้ผลิต แต่ไม่สามารถบอกได้ว่าเป็นคนไทย ก็เนื่องมาจากบริษัทต่าง ๆ ถือลิขสิทธิ์เป็นบริษัทต่างประเทศ ดังนั้นการที่คนไทยจะก่อตั้งบริษัทซอฟต์แวร์ขึ้นก็มีความเป็นไปได้ แต่ปัญหาที่สำคัญที่สุดในตลาดซอฟต์แวร์ไทยก็คือปัญหาการละเมิดลิขสิทธิ์ซอฟท์แวร์ ซึ่งเป็นสาเหตุสำคัญที่ทำให้คนไทยไม่กล้าที่จะผลิตซอฟท์แวร์คุณภาพออกมาแข่งขันกัน และปัญหาอีกอย่างในตลาดซอฟท์แวร์เกมก็คือ การให้ความสนใจในวิธีการผลิตซอฟต์แวร์ของคนไทยมีน้อยเกินไป ในขณะเดียวกันก็คือ การบริโภคหรือการให้ความสนใจในเนื้อหาของเกมนั้นมีมาก ทำให้ความรู้ในการผลิตยังไม่สามารถทำได้เทียบเท่ากับชาวต่างชาติซึ่งเป็นชาติผู้ผลิต ดังนั้นโครงงานนี้น่าจะเป็นแนวทางหนึ่งที่จะแนะนำให้ผู้ที่สนใจได้นำไปศึกษาและผลิตเกมที่สามารถนำออกมาจำหน่ายได้จริงต่อไป

บรรณานุกรม

นิรุธ อำนวนศิลป์. 2545. **เขียนเกมอย่างมืออาชีพด้วย Visual C++ และ DirectX.** กรุงเทพฯ : อินโฟเพรส.

มานพ พรเพียรวิชานนท์. 2544. **ก้าวสู่ game developer มือโปร กับ DirectX.** กรุงเทพฯ : วิตตี้กรุ๊ป.

ยุทธนา สีลาศวัฒนกุล. 2547. **เริ่มต้นการเขียนโปรแกรมด้วยภาษา C++.** พิมพ์ครั้งที่ 2. กรุงเทพฯ : หจก.ไทยเจริญการพิมพ์.

ฉัตรชัย บุษบงค์. 2543. **POSER 4 สร้างสรรค์งาน 3 มิติให้เหมือนจริง.** กรุงเทพฯ : ซีเอ็ดยูเคชั่น.

วงศ์ประชา จันทร์สมวงศ์ และมานิตา เจริญปรุ. 2545. **คัมภีร์ Photoshop 7 & ImageReady 7.** กรุงเทพฯ : โปรวิชั่น.

Crawford, Chris. et. Al. 1982. "The Art of Computer Game Design." [Online]. Available : http://www.vancouver.wsu.edu/fac/peabody/game-book/Coverpage.html.

Rouse, Richard. 2000. **Game design: theory & practice.** U.S.A. : Wordware Publishing.

Lamothe, Andre. 1999. **Tricks of the Windows Game Programming Gurus Fundamentals of 2D and 3D Game Programming .** U.S.A. : SAMS.

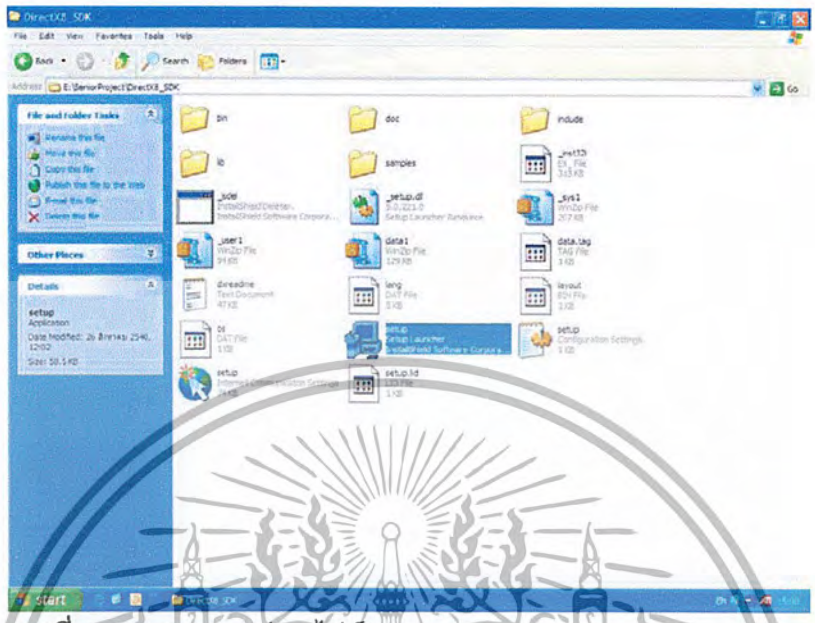
Curious Labs and e-frontier. 2002. "Curious Labs Poser 5 Reference Manual." [Compact Disc]. U.S.A. : Curious Labs and e-frontier.

ภาคผนวก ก.

การติดตั้ง DirectX SDK Version 8.0
การปรับแต่ง Visual C++ ให้ทำงานร่วมกับ DirectX SDK
เพื่อให้ Visual C++ สามารถเรียกชุดคำสั่ง DirectX ได้

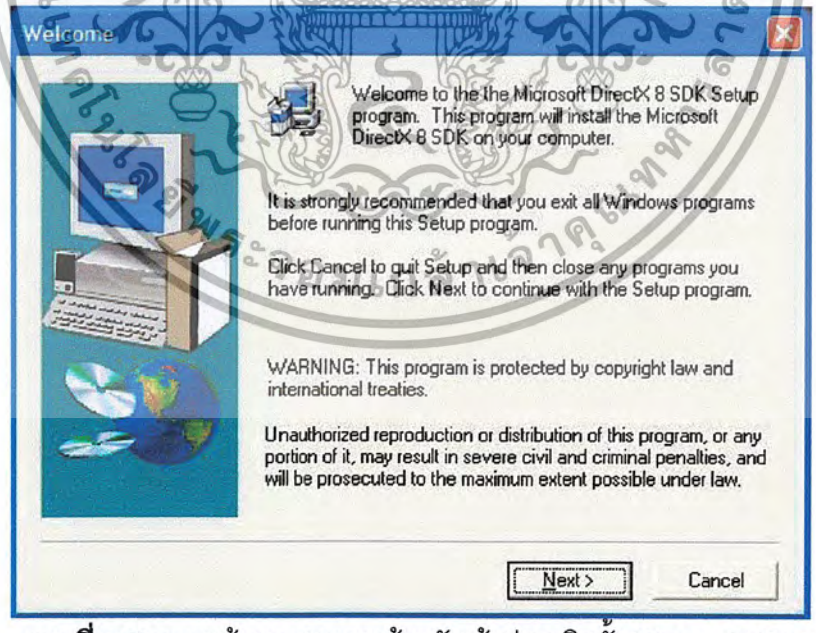
การติดตั้ง DirectX SDK Version 8.0

1.) ทำการติดตั้ง Microsoft DirectX 8 SDK โดยการดับเบิ้ลคลิกที่ไอคอน setup.exe



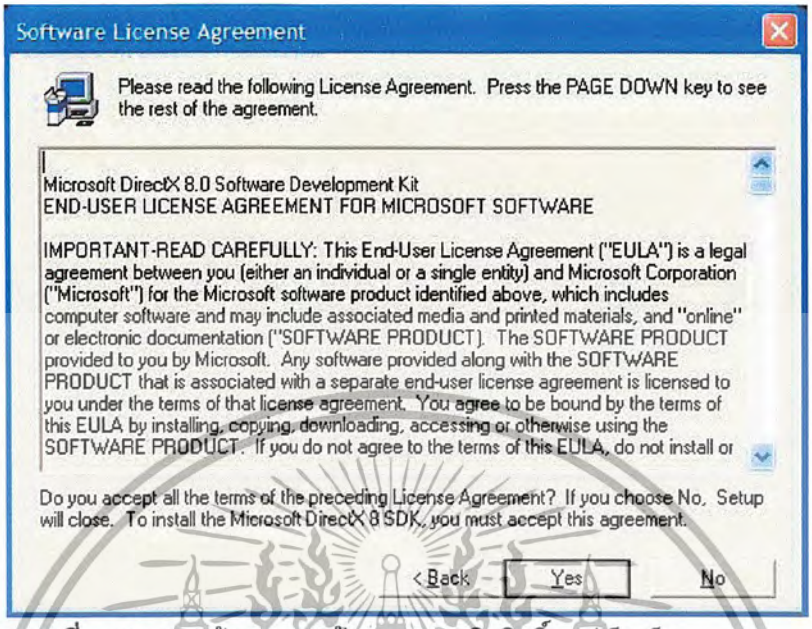
ภาพที่ ก-1 ภาพแสดงรูปของไฟล์ setup.exe

2.) ภาพหน้าจอแสดงการต้อนรับเข้าสู่การติดตั้ง Microsoft DirectX 8 SDK ให้กด Next



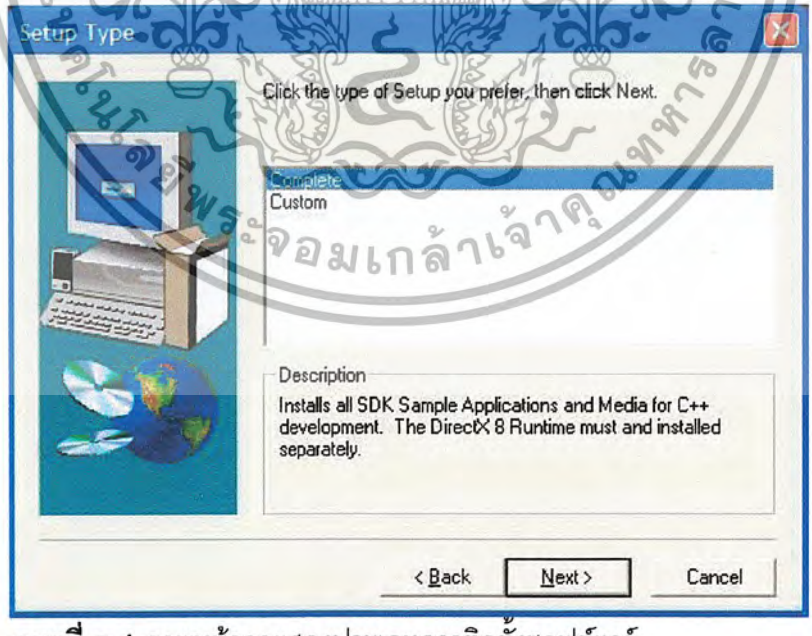
ภาพที่ ก-2 ภาพหน้าจอแสดงการต้อนรับเข้าสู่การติดตั้ง Microsoft DirectX 8 SDK

3.) ภาพหน้าจอแสดงข้อตกลงทางลิขสิทธิ์ซอฟต์แวร์ ให้กด Yes เพื่อยืนยันว่าเห็นด้วยกับข้อตกลง



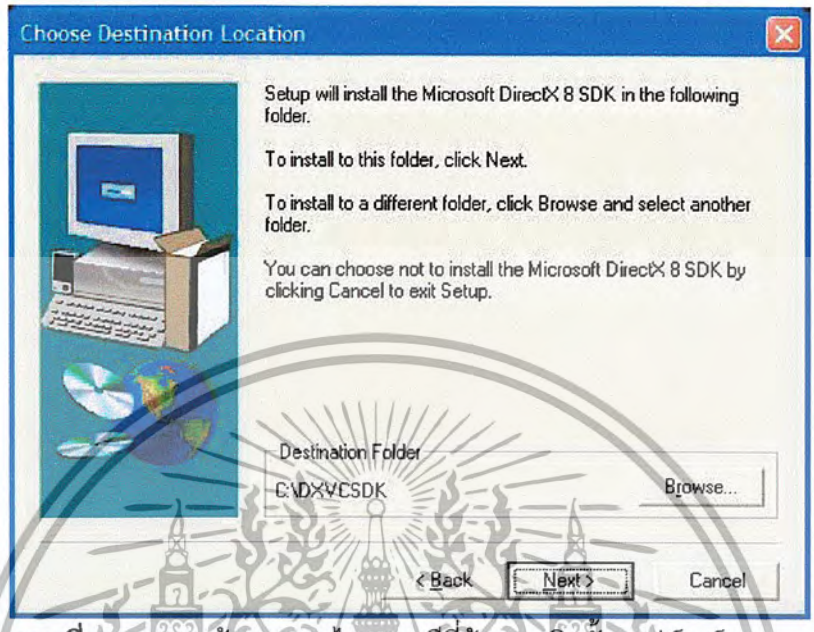
ภาพที่ ก-3 ภาพหน้าจอแสดงข้อตกลงทางลิขสิทธิ์ซอฟต์แวร์

4.) ภาพหน้าจอแสดงประเภทการติดตั้งซอฟต์แวร์ ให้เลือก Complete เพื่อติดตั้งซอฟต์แวร์แบบสมบูรณ์



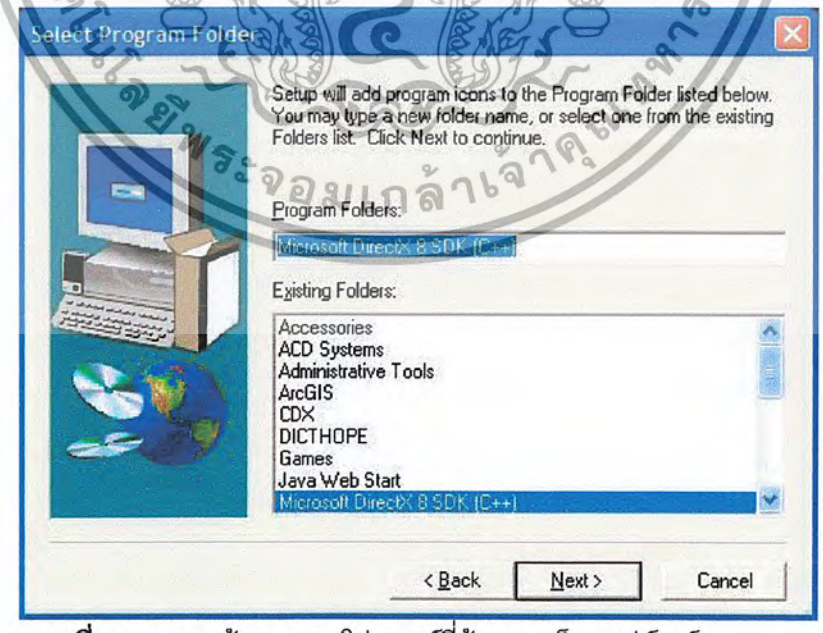
ภาพที่ ก-4 ภาพหน้าจอแสดงประเภทการติดตั้งซอฟต์แวร์

5.) ภาพหน้าจอแสดงไดเรกทอรีที่ต้องการติดตั้งซอฟต์แวร์ ถ้าต้องการเปลี่ยนไดเรกทอรีให้กด Browse จากนั้นกด Next



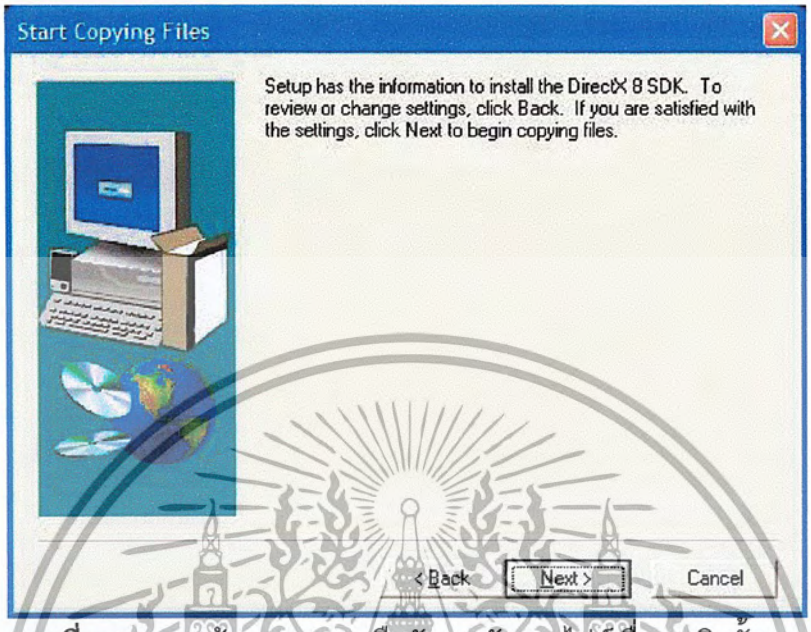
ภาพที่ ก-5 ภาพหน้าจอแสดงไดเรกทอรีที่ต้องการติดตั้งซอฟต์แวร์

6.) ภาพหน้าจอแสดงโฟลเดอร์ที่ต้องการเก็บซอฟต์แวร์ สามารถตั้งชื่อโฟลเดอร์ตามต้องการได้ในช่อง Program Folders หรือเลือกโฟลเดอร์ที่มีอยู่แล้วใน Existing Folders จากนั้นกด Next



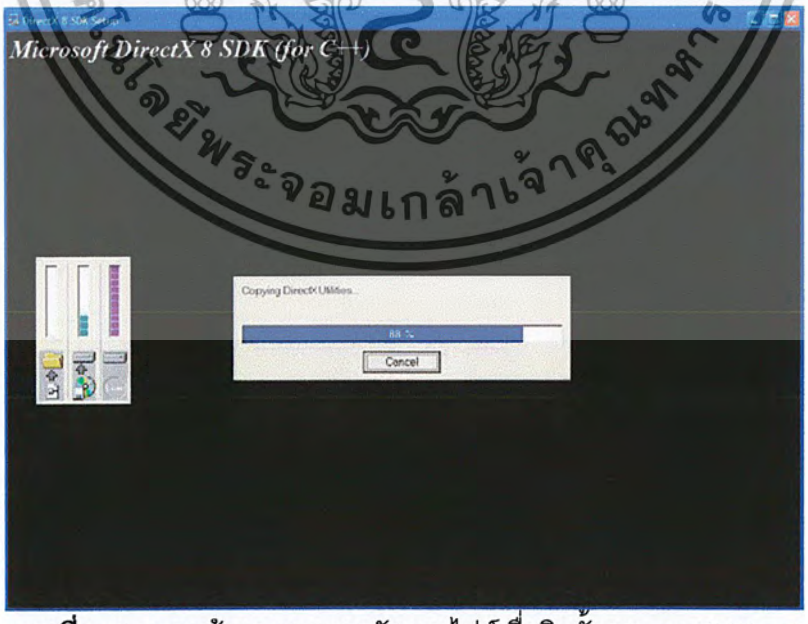
ภาพที่ ก-6 ภาพหน้าจอแสดงโฟลเดอร์ที่ต้องการเก็บซอฟต์แวร์

7.) ภาพหน้าจอแสดงการยืนยันการคัดลอกไฟล์เพื่อการติดตั้ง Microsoft DirectX 8 SDK ให้กด Next เพื่อยืนยันการตกลง



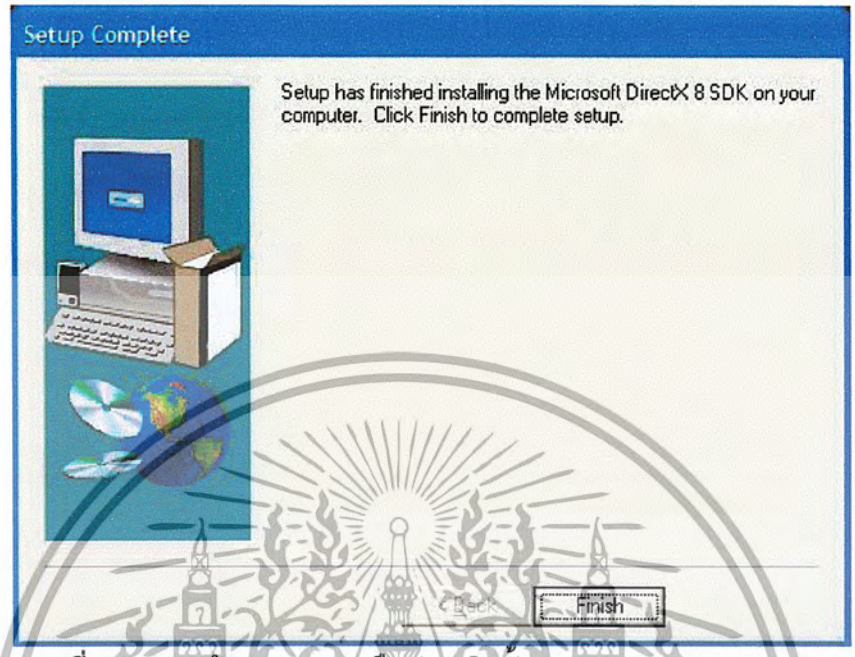
ภาพที่ ก-7 ภาพหน้าจอแสดงการยืนยันการคัดลอกไฟล์เพื่อการติดตั้ง Microsoft DirectX 8 SDK

8.) ภาพหน้าจอแสดงการคัดลอกไฟล์เพื่อติดตั้ง Microsoft DirectX 8 SDK



ภาพที่ ก-8 ภาพหน้าจอแสดงการคัดลอกไฟล์เพื่อติดตั้ง Microsoft DirectX 8 SDK

9.) ภาพหน้าจอแสดงการยืนยันว่าติดตั้ง Microsoft DirectX 8 SDK เสร็จแล้ว ให้กด Finish เพื่อยืนยันการเสร็จสิ้น



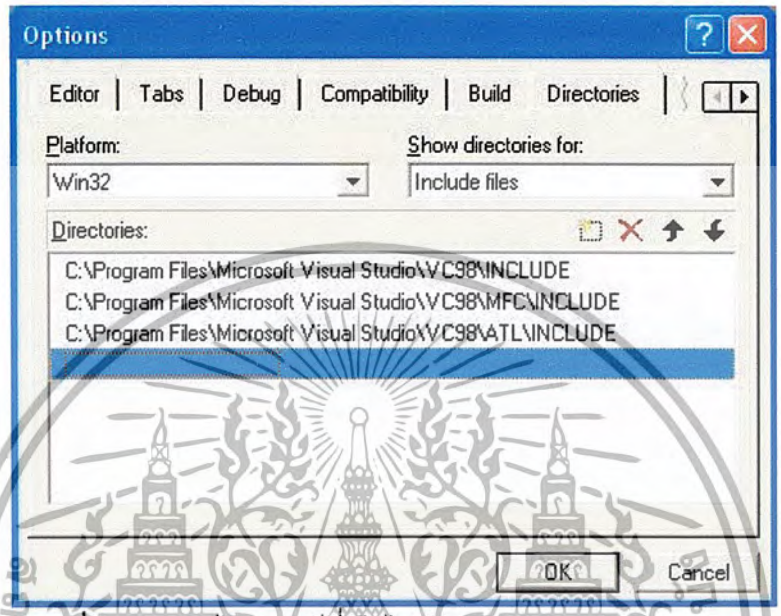
ภาพที่ ก9 ภาพหน้าจอแสดงการยืนยันว่าติดตั้ง Microsoft DirectX 8 SDK เสร็จสิ้น

การปรับแต่ง Visual C++ ให้ทำงานร่วมกับ DirectX SDK

1.) เปิดโปรแกรม Visual C++ ขึ้นมาก่อนจากนั้นเลือกเมนู Tools > Options > Directories

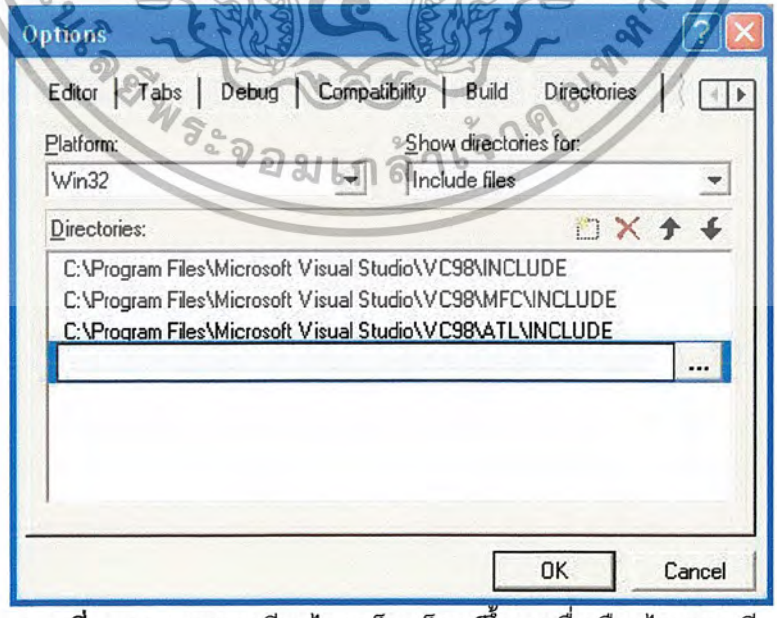
2.) เลือกตัวเลือกในช่อง Show directories for ไปที่ Include files

3.) ดับเบิ้ลคลิกบรรทัดว่าง ๆ ที่ต่อจากตัวเลือกสุดท้ายในรายการ



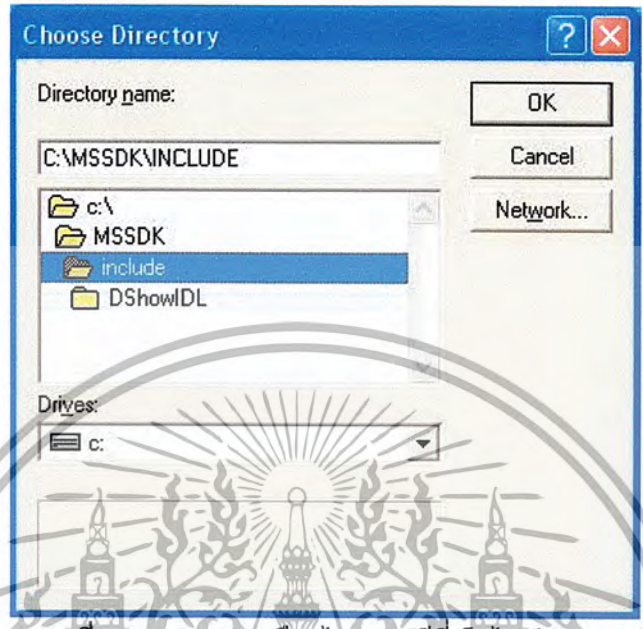
ภาพที่ ก-10 ภาพไดเรกทอรีที่เก็บไฟล์ประเภท Include

4.) คลิกปุ่ม เพื่อเรียก dialog box สำหรับเลือก directories ขึ้นมา



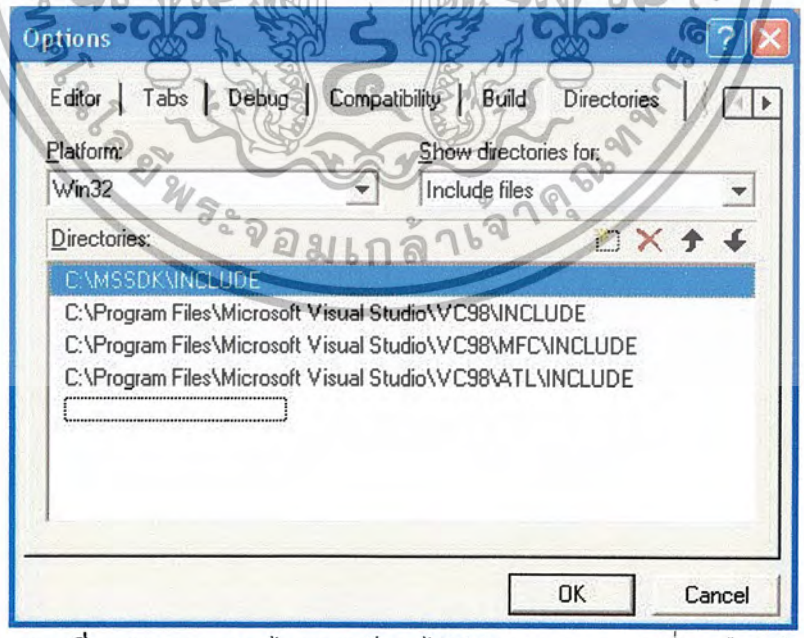
ภาพที่ ก-11 ภาพการเรียกไดอะล็อกบ็อกซ์ขึ้นมาเพื่อเลือกไดเรกทอรี

5.) ให้เลือกไดเรกทอรีที่ได้ลง Microsoft DirectX 8 SDK ไว้ แล้วเลือกที่ include จากนั้นกดปุ่ม OK



ภาพที่ ก-12 ภาพการเลือกไดเรกทอรีที่เก็บไฟล์ประเภท Include

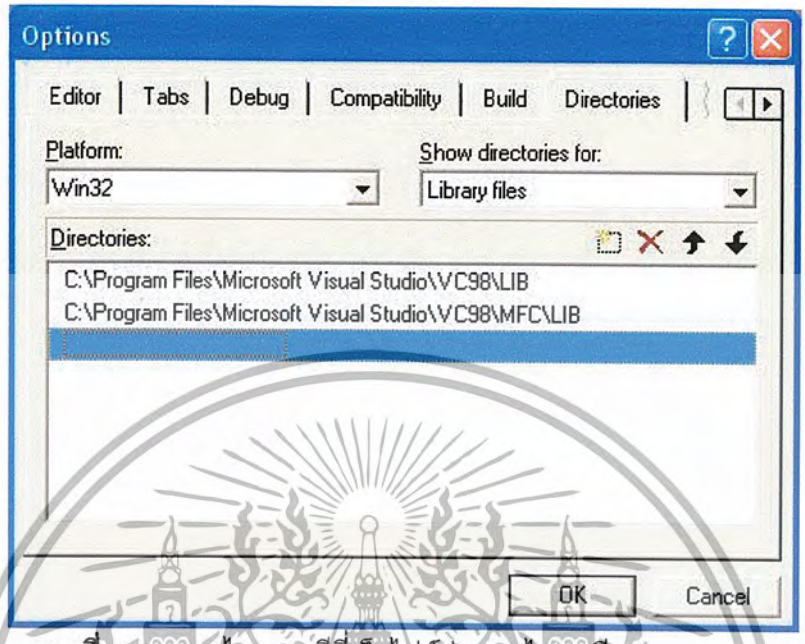
6.) เลื่อนรายการล่าสุดนี้ให้ขึ้นไปเป็นรายการแรกสุดของรายการทั้งหมด



ภาพที่ ก-13 ภาพแสดงไดเรกทอรีของไฟล์ประเภท Include ที่ถูกเลือกเข้ามาแล้ว

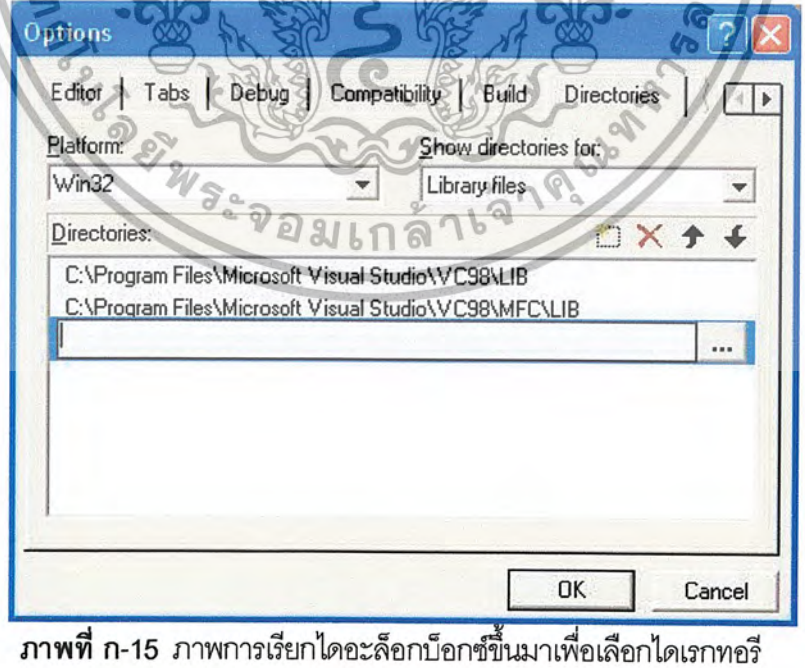
7.) เลือกตัวเลือกในช่อง Show directories for ไปที่ Library files

8.) ดับเบิ้ลคลิกบรรทัดว่าง ๆ ที่ต่อจากตัวเลือกสุดท้ายในรายการ



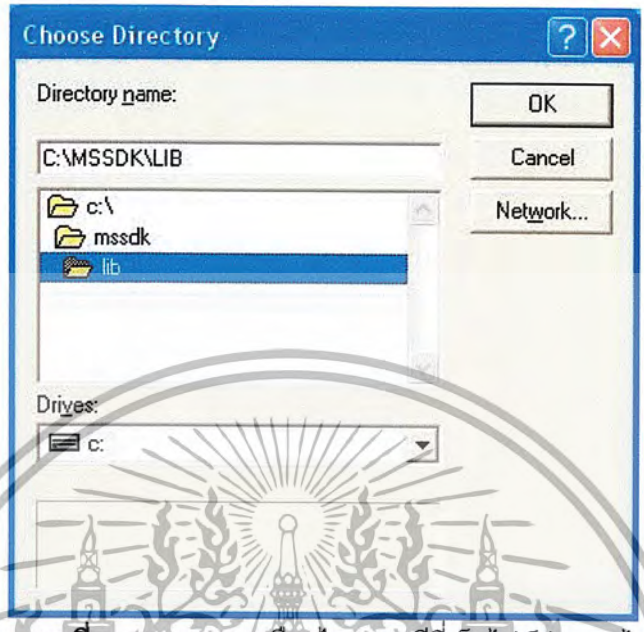
ภาพที่ ก-14 ภาพไดเรกทอรีที่เก็บไฟล์ประเภทไลบรารี

9.) คลิกปุ่ม เพื่อเรียก dialog box สำหรับเลือก directories ขึ้นมา



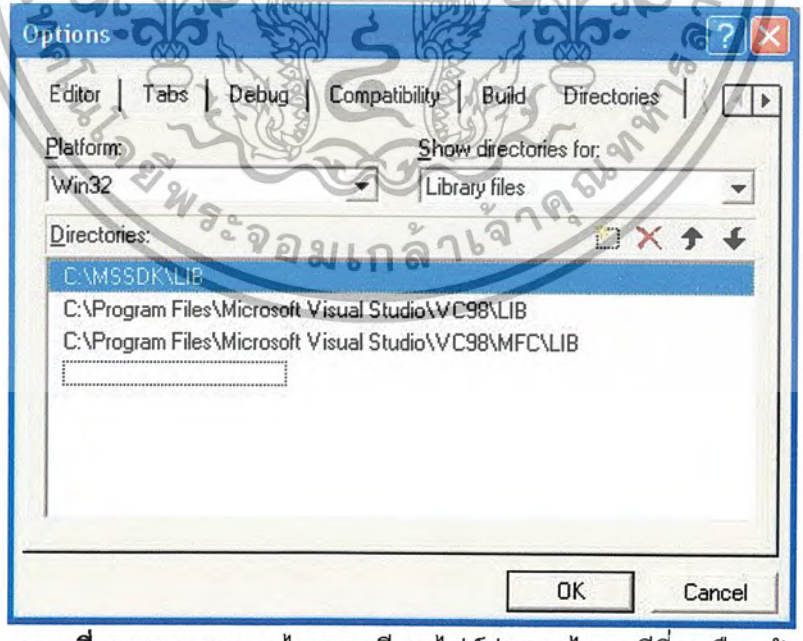
ภาพที่ ก-15 ภาพการเรียกไดอะล็อกบ็อกซ์ขึ้นมาเพื่อเลือกไดเรกทอรี

10.) ให้เลือก directories ที่ได้ลง Microsoft DirectX 8 SDK ไว้ แล้วเลือกที่ lib จากนั้นกดปุ่ม OK



ภาพที่ ก-16 ภาพการเลือกไดเรกทอรีที่เก็บไฟล์ประเภทไลบรารี

11.) เลื่อนรายการล่าสุดนี้ให้ขึ้นไปเป็นรายการแรกสุดของรายการทั้งหมด



ภาพที่ ก-17 ภาพแสดงไดเรกทอรีของไฟล์ประเภทไลบรารีที่ถูกเลือกเข้ามาแล้ว

12.)จากนั้นคลิกปุ่ม OK เพื่อปิด dialog box ของหน้าจอ ptions

ภาคผนวก ข.

ตารางการวางแผนงาน

ตารางที่ ข-1 ตารางการวางแผนงาน

ขั้นตอนการทำงาน	มิ.ย.	ก.ค	ส.ค	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ	มี.ค.
1. ศึกษาหัวข้อปัญหาพิเศษ										
2. วางขอบเขตของปัญหาพิเศษและเลือกโปรแกรมประยุกต์ที่ใช้										
3. กำหนดรายละเอียดใหญ่ของระบบเกม เช่น ส่วนของภาพตัวละครและภาพฉาก ส่วนของ game engine										
4. สร้างภาพตัวละครและฉากด้วย Poser 5 และ PhotoShop 7										
5. ศึกษาการเขียนเกมเชิงวัตถุพร้อมทั้งสร้าง game engine										
6. พัฒนาเกมในแต่ละส่วน										
7. ทดสอบการใช้งานโปรแกรมประยุกต์และแก้ไขปรับปรุงโปรแกรมประยุกต์ พร้อมสรุปผลและทำเอกสาร										