สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

เกมวางแผนการรบระบบ Interactive

STRATEGY WAR GAME PROGRAMMING INTERACTIVE

สมภพ ตั้งจิตต์วัฒนา
เสริมศักดิ์ มหิทธิวาณิชชา
อติชัย ยงใจยุทธ

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

เลขหมู่.....
เลขทะเบียน 36134
วัน, เดือน, ปี 11 ก.ค. 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

STRATEGY WAR GAME PROGRAMMING INTERACTIVE

SOMPHOP TANGJITWATTANNA
SURMSAK MAHITTIWANITCHA
ATICHAI YONGJAIYUHT

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIRMENT FOR THE DEGREE OF BACHELOR OF SCIENCE
DEPARTMENT OF MATHTEMATICS AND COMPUTER SCIENCES
FACULTY OF SCIENCE
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 1999

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปัญหาพิเศษ	เกมวางแผนการรบระบบ INTERACTIVE STRATEGY WAR GAME PROGRAMMING INTERACTIVE	
ชื่อนักศึกษา	นายสมภพ ตั้งจิตต์วัฒนา	39054669
	นายเสริมศักดิ์ มหิทธิวาณิชชา	39054680
	นายอติชัย ยงใจยุทธ	39054682
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
อาจารย์ที่ปรึกษา	อาจารย์ธีรวัฒน์ ประกอบผล	

ภาควิชาคณิตศาสตร์ประยุกต์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2542

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	อาจารย์สิริลักษณ์ อนันต์สถิตย์สิน	
กรรมการ	อาจารย์นันทิกา เบญจเทพานันท์	
กรรมการและอาจารย์ที่ปรึกษา	อาจารย์ธีรวัฒน์ ประกอบผล	

(อาจารย์ไพรบูลย์ พันธรักษ์พงษ์)

หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

ลิขสิทธิ์ของภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หัวข้อปัญหาพิเศษ	เกมวางแผนการรบระบบ Interactive	
ชื่อนักศึกษา	นายสมภพ ตั้งจิตต์วัฒนา	39054669
	นายเสริมศักดิ์ มหิทธิวาณิชชา	39054680
	นายอติชัย ยงใจยุทธ	39054682
ปริญญา	วิทยาศาสตรบัณฑิต	
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์	
สาขาวิชา	วิทยาการคอมพิวเตอร์	
ปีการศึกษา	2542	
อาจารย์ที่ปรึกษา	อาจารย์ธีรวัฒน์ ประกอบผล	

บทคัดย่อ

ปัจจุบันในวงการอุตสาหกรรมการผลิตโปรแกรมเกมคอมพิวเตอร์ ได้มีการปรับตัวอย่างรวดเร็ว และมีผู้ผลิตมากมายในตลาด เพราะผู้ผลิตต่างๆได้นำเทคโนโลยีชั้นสูงเข้ามาช่วยในการพัฒนาคุณลักษณะและเทคนิคในเกม ดังนั้นทำให้คณะศึกษาได้เลือกศึกษาโครงการเกี่ยวกับการสร้างโปรแกรมเกมสำหรับคอมพิวเตอร์ ซึ่งในมุมมองของทีมงาน โปรแกรมเกมจะไม่ได้วัดกันด้วยแง่ธุรกิจ หรือยอดขาย หรือความนิยมของผู้เล่นเพียงอย่างเดียว แต่จะต้องวัดที่ความท้าทาย, ความแปลกใหม่ และการตอบสนองระหว่างเกมกับผู้เล่นที่เสมือนจริง ซึ่งคณะผู้ศึกษาได้ทำการศึกษาวิจัยเทคนิคต่าง ๆ จากโปรแกรมอื่น ๆ ในท้องตลาด

โครงงานโปรแกรมเกมนี้ จะนำเสนอความท้าทาย, ความแปลกใหม่, และการตอบสนองระหว่างคอมพิวเตอร์และผู้เล่นซึ่งจะแตกต่างจากเกมทั่วไป พร้อมกันนี้คณะผู้ศึกษาจะได้นำเสนอกระบวนการพัฒนาที่เป็นลักษณะแนวความคิดเฉพาะแบบ โดยเกมจะจำลองสถานการณ์คล้ายในสนามรบจริง ผู้เล่นจะได้รับจุดมุ่งหมายหรือเป้าหมายในการปฏิบัติการ แต่ผู้เล่นจะต้องเผชิญการต่อสู้ในรูปแบบต่าง ๆ ซึ่งคอมพิวเตอร์จะสุ่มเลือกรูปแบบการรบต่าง ๆ ในต่างสถานการณ์ ซึ่งผู้เล่นจะเป็นการฝึกฝนการวางแผนกลยุทธ์ต่าง ๆ ตลอดเวลา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Special Project Title	Strategy War Game Programming Interactive	
Students	Mr. Somphop Tangjitwattana	39054669
	Mr. Surmsak Mahittiwanitcha	39054680
	Mr. Atichai Yongjaiyuth	39054682
Degree	Bachelor's Degree of Science	
Department	Mathematics and Computer Sciences, Faculty of Science	
Programme	Computer Sciences	
Academic Year	1999	
Special Project Advisor	Lecturer Teerawat Prakobphon	

ABSTRACT

The game software industry had fast development and more competitors in the market. Because they adapt high technology for develop feature and technique in game. With this reason, it had more challenge for us for choose project about build game software. With our focus, The Game software did not performance by business or sales volume or popular game only. But we will performance with the new challenge, new style, and interactive of game that we have research technique from other products in market.

Our game software project will be create new challenge, new style, and interactive between game and player with difference from others. We will present with the special development system in our unique concept. Game will be simulate reality from the real combat. Player will got objective or target , they must flight with computer with their own technique in difference events. Computer will be random tack-tic for flight with player. With this reason, Player will be practice their strategy every times.

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กิตติกรรมประกาศ

กราบขอบพระคุณบุพการี ผู้อุปการะ ผู้ปกครอง แม่เลี้ยง พ่อเลี้ยง ที่เลี้ยงดู อุ้มชู จนพวกเราได้พบกับวันแห่งความสำเร็จเช่นวันนี้

ขอขอบพระคุณอาจารย์ธีรวัฒน์ ประกอบผล ซึ่งเป็นอาจารย์ที่ปรึกษาปัญหาพิเศษที่มอบความเป็นกันเอง และความเอาใจใส่ ทำให้พวกเรามีกำลังใจและไม่ท้อแท้

ขอขอบพระคุณอาจารย์ทุกท่านที่มอบความเมตตา และให้คำปรึกษาด้วยดีตลอดมา

ขอขอบพระคุณพี่วีรศักดิ์ พงษ์ธัญญวิชัย ที่คอยเป็นแรงบันดาลใจ ช่วยเหลือและให้กำลังใจกับพวกเราเสมอมา

ขอขอบคุณคุณ Andre LaMothe ผู้เขียนหนังสือ Game for Dummies ผู้ซึ่งช่วยแก้ปัญหาต่างๆ ด้วยดีตลอดมา

ขอขอบคุณคุณ Jim Adams ผู้เขียนเอกสารบน GameDev.net ผู้ซึ่งตอบเมลทุกฉบับด้วยความปรารถนาดี

ขอบคุณเพื่อนๆทุกคน ที่มีความตั้งใจจะเล่นเกมนี้ให้ได้ ทำให้พวกเรามีแรงใจอีกทางหนึ่ง

คณะผู้จัดทำ

มีนาคม 2543

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ

หน้า

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญตาราง

ตารางที่ **หน้า**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ

ภาพที่ **หน้า**

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

เนื่องจากในปัจจุบันอุตสาหกรรมเกมได้มีการพัฒนาและเติบโตอย่างรวดเร็ว จากการคาดการณ์ในอนาคต อุตสาหกรรมเกมจะเป็นอุตสาหกรรมที่มีการแข่งขันทางด้านการตลาดที่มีมูลค่าสูง เพราะฉะนั้น การศึกษาวิธี และหลักในการสร้างระบบเกมจึงเป็นสิ่งที่น่าสนใจ โดยที่ผู้ศึกษาจะได้รับความรู้เกี่ยวกับวิธีการสร้างระบบเกม วิธีการติดต่อกับ ฮาร์ดแวร์ อินพุท เอาท์พุท รวมถึงวิธีการใช้ เครื่องมือต่างๆ ในการสร้างรูปภาพ ภาพเคลื่อนไหว อีกทั้งยังสามารถเขียนโปรแกรมเกม เพื่อตอบโต้กับผู้เล่นได้ในหลายๆ รูปแบบอีกด้วย โดยหวังว่า จะเป็นประโยชน์ต่อการนำไปศึกษาต่อ เพื่อเป็นแนวทางเบื้องต้นสำหรับผู้ที่สนใจการพัฒนาเกม และยังเป็นแรงผลักดันให้เกิดอุตสาหกรรมการผลิตเกมในประเทศไทย

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1.2.1 เพื่อศึกษาระบบและวิธีการสร้างเกม

1.2.2 เพื่อเป็นแนวทางผลักดันให้เกิดอุตสาหกรรมการผลิตเกมในประเทศไทย

1.2.3 เพื่อศึกษาการเขียนโปรแกรมบน Windows ด้วยเทคโนโลยีการแสดงภาพและประมวลผลภาพโดยใช้โปรแกรม Microsoft DirectX

1.3 สมมติฐานของการศึกษา

1.3.1 เกมสามารถโต้ตอบกับผู้เล่นได้หลายรูปแบบ โดยผู้เล่นเองไม่สามารถคลาดการณ์ได้ล่วงหน้า

1.3.2 เป็นประโยชน์ต่อผู้ที่นำไปศึกษาเป็นแนวทางต่อไป

1.3.3 สามารถเขียนโปรแกรม แสดงภาพเคลื่อนไหว และโต้ตอบกับผู้เล่นได้

1.4 ขอบเขตการศึกษา

1.4.1 การพัฒนาโปรแกรมทำได้โดยโปรแกรม Microsoft Visual C++ 6.00

1.4.2 ใช้ Microsoft DirectX SDK เป็น Library สำคัญในการพัฒนา

1.5 ขั้นตอนการศึกษา

1.5.1 ศึกษาเกี่ยวกับระบบการสร้างเกม

เป็นขั้นตอนการศึกษา วิธี และหลักการที่ใช้ในการสร้างเกม รวมทั้งเทคนิคต่างๆ ที่จะนำมาเขียนและพัฒนาระบบเกม

1.5.2 ศึกษาโปรแกรมและเครื่องมือ ต่าง ๆ

เป็นการศึกษาโปรแกรมและเครื่องมือ ที่จะนำมาใช้และพัฒนาในการเขียนโปรแกรมโดย โปรแกรมและเครื่องมือที่จะนำมาใช้ในการพัฒนาจะเน้น Visual C ++ , DirectX SDK และโปรแกรมทางด้านกราฟฟิก เพื่อนำมาใช้ในการสร้างภาพ โดยจะเน้นที่โปรแกรม 3D Studio Max , Photo shop, Poser 3D เป็นต้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.5.3 เก็บรวบรวมเอกสารและข้อมูลต่าง ๆ

เป็นการนำเอาเอกสารและข้อมูลที่เกี่ยวข้องมารวบรวม และใช้ประกอบการทำงาน โดยส่วนมากจะเป็นการรวบรวมจาก หนังสือ ตำรา และ Web site ที่เกี่ยวข้อง รวมถึงข้อมูลที่ได้จากผู้มีความรู้ในแต่ละด้าน

1.5.4 วิเคราะห์และออกแบบระบบเกม

วิเคราะห์ และออกแบบระบบงาน โดยแบ่งออกเป็นส่วนๆ เช่น ส่วนรับข้อมูล ส่วนประมวลผล ส่วนแสดงผล และออกแบบระบบ เพื่อให้ทำงานได้เกิดประสิทธิภาพมากที่สุด

1.5.5 พัฒนาโปรแกรม

เป็นขั้นตอนการเขียนโปรแกรมตามขั้นตอนที่ได้ออกแบบไว้ในขั้นตอนการวิเคราะและออกแบบระบบ

1.5.6 การทดสอบโปรแกรมและปรับปรุงโปรแกรม

เป็นขั้นตอนการทดสอบโปรแกรม และบอกถึงความสามารถทั้งหมดที่เป็นไปได้ของโปรแกรม

1.6 ข้อตกลงเบื้องต้น

1.6.1 โปรแกรมรัน เฉพาะระบบปฏิบัติการ Windows ตั้งแต่ เวอร์ชั่น 95 ขึ้นไป

1.6.2 ในระบบปฏิบัติการต้องมี Microsoft DirectX soft ware ตั้งแต่เวอร์ชั่น 5 ขึ้นไป

1.6.3 ต้องการคอมพิวเตอร์ที่มีโปรเสสเซอร์ ตั้งแต่ ระดับ pentium 2 ความเร็ว 266 MHz ขึ้นไป

1.6.4 มีหน่วยความจำอย่างน้อย 32 Mbyte

1.6.5 ต้องมี ฮาร์ดแวร์อินพุท ประเภท Mouse และ Keyboard

1.7 ข้อจำกัดของการศึกษา

1.7.1 ระยะเวลาที่ทำการศึกษา

1.7.2 ความรู้และความเข้าใจที่ได้จากการศึกษา

1.7.3 ทฤษฎี ที่กำหนดขึ้นมาแล้วในการศึกษาปัญหา

1.7.4 เทคโนโลยีการศึกษาปัญหาพิเศษ

1.8 คำจำกัดความที่ใช้ในการศึกษา

1.8.1 Driver ตัวขับโปรแกรมควบคุมอุปกรณ์ หรือโปรแกรมอื่นๆ

1.8.2 SDK มาจากคำว่า Software Development Kit หมายถึง ชุดซอฟท์แวร์ที่ใช้ในการพัฒนาซอฟท์แวร์

1.8.3 GDI มาจากคำว่า Graphic Device Interface หมายถึง ส่วนการติดต่อกับอุปกรณ์การแสดงผลทางด้านกราฟฟิก

1.8.4 Compiler หมายถึง ตัวแปลโปรแกรม

1.8.5 Bitmap คือภาพที่ประกอบจากจุดเล็กๆ จำนวนมาก

1.8.6 GUID มาจากคำว่า Globally Unique Identifiers เป็นข้อกำหนดชนิดของอุปกรณ์ต่างๆ โดยระบุเป็น หมายเลขเพื่อให้ทราบว่า เป็นอุปกรณ์ประเภทหรือชนิดใด

1.8.7 Sprite หมายถึง ภาพนิ่งหลายภาพที่มีลักษณะต่อเนื่องกันมีประโยชน์ในการทำภาพเคลื่อนไหว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 2
หลักการและทฤษฎีที่เกี่ยวข้อง

2.1 โปรแกรม Microsoft Direct X

2.1.1 ความเป็นมา

ในสมัยก่อน การเขียนโปรแกรมให้ติดต่อกับอุปกรณ์ ต่างๆ เป็นเรื่องยุ่งยาก โปรแกรมเมอร์ จำเป็นที่จะต้องทำการศึกษาถึงคำสั่งต่างๆ ของ Hard ware แต่ละรุ่น แต่ละยี่ห้อ ซึ่งมีจำนวนมาก และเป็นเรื่องปริมาณที่ทำให้เสียเวลา และเมื่อ Hardware ตัวเก่าล้าสมัยและ Hardware ตัวใหม่ถูกผลิตออกมา โปรแกรมเมอร์ก็จำเป็นต้องทำการศึกษาถึงวิธี การใช้คำสั่งต่างๆ ของ Hard ware ตัวใหม่ ซึ่งอาจจะแตกต่างจากตัวเก่าโดยสิ้นเชิง หรือมีการเพิ่มเทคโนโลยีใหม่ๆ เข้ามาใน Hardware ซึ่งเป็นเรื่องที่ทำความลำบากแก่ผู้พัฒนาเป็นอย่างมาก

ใน Windows (ตั้งแต่ version 3.1 เป็นต้นมา) มีการใช้ Driver ของ Hardware เพื่อที่จะเข้าถึงอุปกรณ์ เป็นเหตุทำให้นักโปรแกรมเมอร์พบกับความลำบากในการเขียนโปรแกรมเพื่อติดต่อกับ Hardware ในการพัฒนาโปรแกรมโปรแกรมเมอร์คาดหวังว่าใน Windowsรุ่นใหม่ๆ จะมีชุดคำสั่ง ที่จะช่วยให้การเขียนโปรแกรม ติดต่อกับอุปกรณ์ สามารถทำผ่าน Interface ตัวหนึ่งเพียงตัวเดียวได้ ซึ่ง Interface ตัวนี้จะทำการ เรียกใช้ Driver ต่างๆ ของอุปกรณ์นั้นๆ เอง ซึ่งจะทำให้การเขียนโปรแกรมเป็นไปอย่างสะดวก เพราะเพียงทำการศึกษาถึงคำสั่งการเรียกใช้ Interface ก็เพียงพอที่จะติดต่อกับอุปกรณ์ต่างๆรุ่นใดๆ ยี่ห้อใดๆ ได้เลยทันที

บริษัท Microsoft Corporation เล็งเห็นถึงความต้องการของโปรแกรมเมอร์ ตรงจุดนี้จึงได้ทำการผลิตชุดพัฒนา SDK สำหรับ Microsoft Windows ขึ้นมา โดยมีจุดประสงค์ ที่จะช่วยให้การเข้าถึง Hardware ต่างๆ เป็นเรื่องง่ายยิ่งขึ้นอีก ซึ่งไม่จำเป็นต้องไปทำการศึกษาการทำงานของ Hardware ตัวนั้นๆ เหมือนสมัยก่อน โดยระยะแรกมีวัตถุประสงค์ เพื่อใช้พัฒนา เกมคอมพิวเตอร์โดยเฉพาะ จึงได้ตั้งชื่อว่า " Game kit " ต่อมา ได้เพิ่มความสามารถที่ช่วยทางด้านอื่นๆ ด้วยเช่น multimedia ดังนั้นจึงได้พัฒนาและถูกเรียกใหม่ว่า Microsoft DirectX และชุดพัฒนาจะมี SDK ต่อท้าย โดยปัจจุบันได้ออกมาถึง รุ่น ที่ 7.0 แล้ว

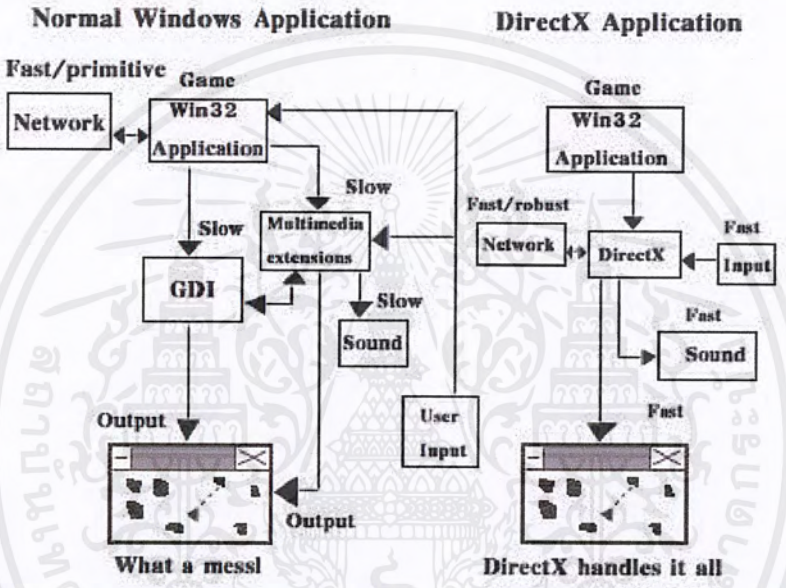
2.1.2 หลักการของ Direct X

สิ่งที่ Microsoft ได้คำนึงถึง เป็นสิ่งแรกของ Direct X ก็คือ จะต้องเป็นชั้นระหว่าง Application กับ Hardware ที่ต้องมีความสัมพันธ์กันมากเสมือนเป็นชั้นเดียวกัน เพราะต้องการให้ Application สามารถทำงานกับ Hardware ได้อย่างรวดเร็ว ความจริงแล้วก่อนที่ Direct X จะถูกพัฒนาขึ้นมา ได้มีชุดคำสั่งของ Windows ซึ่งใช้ติดต่อ กับการ์ดแสดงผลอยู่ก่อนแล้ว ซึ่งเรียกว่า GDI ซึ่งมีลักษณะของการทำงานคล้าย Direct X ซึ่งก็คือสามารถติดต่อกับอุปกรณ์ Hardware ต่างๆ ได้โดยตรงโดยไม่ต้องทำการเรียกใช้ Driver ของการ์ดจอเอง แต่ความสามารถจะช้ากว่า และความสามารถบางอย่างยังไม่สามารถทดแทนได้ เช่น หากโปรแกรมเมอร์ ทำการเรียกคำสั่ง ที่ติดต่อกับ Hardware แสดงผลซึ่งมีผลทำให้การทำงานเร็ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มากขึ้น เพราะทำงานบน Hardware หากว่าเครื่องของผู้ใช้ Application นั้นไม่มี Hardware ดังกล่าว ถ้าเป็น GDI จะไม่สามารถทำงานหรือ ใช้งาน Application นั้นได้เลย แต่หาก โปรแกรมเมอร์เรียกใช้ Hardware ผ่านทาง Direct X เมื่อ Direct X ทำการตรวจสอบหา Hardware ดังกล่าวและหาไม่เจอ ก็จะทำการจำลอง Hardware ตัวนั้นเหมือนกับว่ามี Hardware ตัวนั้นอยู่และใช้ Algorithm ที่กำหนดไว้ทำการประมวลผล และทำการแสดงผลแทน Hardware ตัวนั้น ซึ่งผู้ใช้สามารถเรียกใช้ Software แทน Hardware และยังสามารถใช้งานได้เหมือนเดิม

Direct X คือชุดของ Dynamic Link Library (DLL) ซึ่งสามารถทำให้โปรแกรมเมอร์เข้าถึง Hardware ได้อย่างรวดเร็ว โดยไม่ผ่านทาง Hardware ตามหลักการของ Windows ดังรูปภาพที่ 2.1



รูปที่ 2.1 Direct X สนับสนุน เกมที่รันบน Windows ด้วยประสิทธิภาพที่สูง

Direct X ยังเป็นมาตราฐานของ Software และ Hardware ในปัจจุบัน โดยบริษัท Software ส่วนมากที่มีอุปกรณ์ ที่เกี่ยวข้องกับ Direct X จะเขียนโปรแกรมให้เรียกใช้ Direct X และรวมไปถึงบริษัทที่ผลิต Hardware ก็ยังสนับสนุน โดย Direct X จะทำการเรียกใช้คำสั่งของ Direct X เพราะฉะนั้นเราสามารถที่จะมั่นใจได้ว่า Direct X จะเป็นที่แพร่หลายในอนาคต

Microsoft ได้ทำให้ Direct X มีคุณสมบัติพิเศษอย่างมากเช่น เกมที่เขียนโดยใช้ Direct X 1.0 จะสามารถ run บนคอมพิวเตอร์ ที่ติดตั้ง Direct X 3.0 หรือ 6.0 และยิ่งไปกว่านั้น Microsoft รู้ว่าเทคโนโลยี Direct X จะต้องหลุดออกจากหลักการ หากว่าถูกเขียนโดยไม่มีการมองการณ์ไกลและการวางแผน ดังนั้นจึงจำเป็นที่จะต้องใช้การใช้การโปรแกรมลักษณะของ Object Oriented ซึ่งสามารถปรับปรุงให้ดีขึ้นได้ และสามารถทำงานได้กับหลายๆ ภาษา และเป็นลักษณะของ Black box เทคนิคเช่นนี้ถูกเรียกว่า Component Object Model (COM)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3 ส่วนประกอบของ Direct X

2.1.3.1 HAL (The Hardware Abstraction Layer)

HAL คือ Software ระดับล่างสุดใน Direct X ซึ่งบรรจุ Driver ของ Hardware จากผู้ผลิต Hardware รายนั้นๆ เพื่อที่จะทำการควบคุม Hardware ได้โดยตรง Software ในชั้นนี้ให้ประสิทธิภาพสูงสุด เพราะเป็นชั้นที่ทำการติดต่อกับ Hardware โดยตรง

2.1.3.2 HEL (The Hardware Emulation Layer)

HEL จะวางอยู่ชั้นก่อนหน้า HAL โดยปกติ Direct X ถูกออกแบบมาให้ใช้ Hardware ให้ได้เต็มประสิทธิภาพและด้วยความเร็วสูง หากว่ามี Hardware ชนิดนั้นๆ อยู่ แต่ Direct X ก็ยังสามารถทำงานได้อยู่หากว่า Hardware ชนิดนั้นไม่มีอยู่ ตัวอย่างเช่น หากเราต้องการสร้างภาพๆ หนึ่งขึ้นมาซึ่งจำเป็นต้องเรียกใช้ Function บน Hardware ชนิดหนึ่ง โดยทำการเรียกผ่าน Direct X ซึ่งถ้ามี Hardware ตัวนั้นอยู่ก็จะทำการเรียกใช้ Function บน Hardware ชนิดนั้นได้โดยตรง แต่หากว่า Hardware ชนิดนั้นไม่มีอยู่ในเครื่อง HEL จะเข้ามาทำงานในส่วนตรงนี้ โดยการจำลองตัวเอง เป็น Hardware ตัวนั้นๆ และทำงานเคียงคู่กับ HAL ซึ่งแน่นอนว่าทำให้การทำงานช้าลง เพราะเป็นการจำลอง Hardware แต่ก็ยังทำให้ Application นั้นทำงานได้ตามปกติได้

2.1.3.3 Direct Draw

อาจจะเรียกได้ว่า ส่วนนี้เป็นส่วนที่สำคัญที่สุดของ Direct X เพราะ Direct Draw สามารถที่จะทำการเรียกใช้งาน Video card ด้วยความเร็วสูง โดยที่ Direct Draw รู้จักการ set mode ของจอภาพทุกๆ mode ไม่ว่าจะเป็น high-resolution หรือจะเป็น true color modes แล้วยังสนับสนุนเรื่องการจัดการสี การกำหนดขอบเขตจอภาพ และการทำ Animation

2.1.3.4 Direct 3D

ส่วนประกอบของ Direct X ชนิดนี้เป็นส่วนประกอบส่วนเดียวที่สามารถติดต่อกับอุปกรณ์แสดงผล 3 มิติได้ ทำให้สามารถเขียนโปรแกรมประเภท 3 มิติได้ เช่น เกมขับเครื่องบิน Simulation ต่างๆ

2.1.3.5 Direct Sound

ก่อนที่ยังไม่มี Direct X การเขียนโปรแกรมติดต่อกับการ์ดเสียงเป็นเรื่องที่ยากมาก ซึ่งการเขียนโปรแกรมที่อาศัยการ์ดเสียง และสร้างเสียงที่มีคุณภาพในสมัยก่อนมักจะว่าจ้างให้บริษัทที่ทำงานด้านนี้โดยเฉพาะเป็นผู้ออกแบบและทำการ Coding เช่นบริษัทเกม อาจเป็นบริษัทหนึ่ง แล้วจ้างบริษัทที่ทำระบบเสียงให้ทำการเพิ่มเสียงเพลงเข้าไปในตัวเกม เป็นต้น แต่ปัจจุบันเมื่อ Direct Sound ถือกำเนิดขึ้นสถานการณ์เช่นนั้นจึงหมดลงตามยุค เพราะ Direct Sound ลดขั้นตอนการเข้าถึง Sound card และทำงานได้กับ Sound card ทุกชนิด

2.1.3.6 Direct Sound 3D

Direct Sound 3D อยู่บนพื้นฐานของ Direct Sound และทำงานได้กับระบบเสียง 3 มิติ ซึ่งจำเป็นต้องมีการคำนวณทางคณิตศาสตร์ Direct 3D จะทำการคำนวณ และพิกัดตำแหน่งของเสียงในระบบ 3 มิติ ซึ่งจะทำให้เสียงที่ออกมาฟังสมจริงมากขึ้น

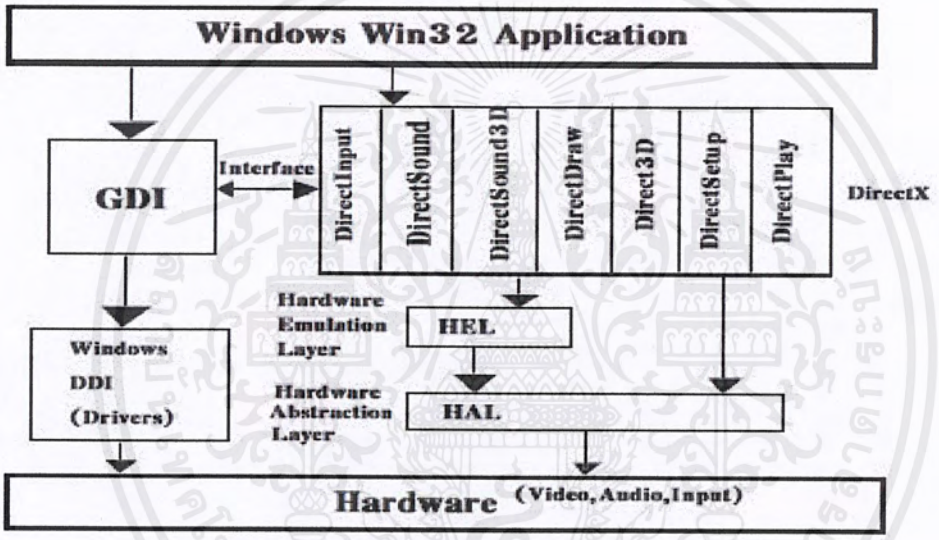
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.1.3.7 Direct Input

เป็นความสามารถใหม่ที่เพิ่งจะถูกเพิ่มเข้ามาใน Direct X 3.0 ซึ่งจะทำให้การรับข้อมูลทำได้หลากหลายขึ้นและทำได้โดยง่าย โดยรับจาก Keyboard , Mouse และ Joystick ซึ่งใน เวอร์ชั่น 5.0 ยังสนับสนุน Joystick แบบ Force Feed Back (สั่นสะเทือนได้) ในสมัยก่อน Joystick มีปุ่มเพียงเล็กน้อย แต่ในปัจจุบันมีมากมายหลายปุ่ม และมีหลากหลายยี่ห้อ ทำให้การติดต่อกับ Joystick ในแต่ละปุ่มต้องทำการศึกษาถึง Joystick ชนิดนั้นๆ อย่างลึกซึ้ง

2.1.3.8 Direct Play

เป็นส่วนประกอบที่ใช้ติดต่อกับอุปกรณ์ทางด้าน Network การอ้าง Port ต่างๆ หรือการสร้างเกมที่สามารถเล่นผ่านเครือข่ายได้



รูปที่ 2.2 DirectX Component

2.2 การเขียนโปรแกรมบน Windows

การเขียนโปรแกรม Windows อาจจะเป็นเรื่องที่ง่านหรือยาก ขึ้นอยู่กับขนาดของโปรแกรมที่ต้องการทำ การเขียน Word processor อาจต้องการความรู้ทางด้านการเขียนโปรแกรมที่มากพอสมควร แต่สำหรับการเขียนเกมโดยใช้ Direct X ต้องการเพียงเล็กน้อยเท่านั้น ในส่วนนี้จะอธิบายถึงการเขียนโปรแกรมบน Windows และพื้นฐานที่ควรจะรู้สำหรับการเขียนโปรแกรมบน Windows เพื่อนำไปใช้ในการเขียนเกม

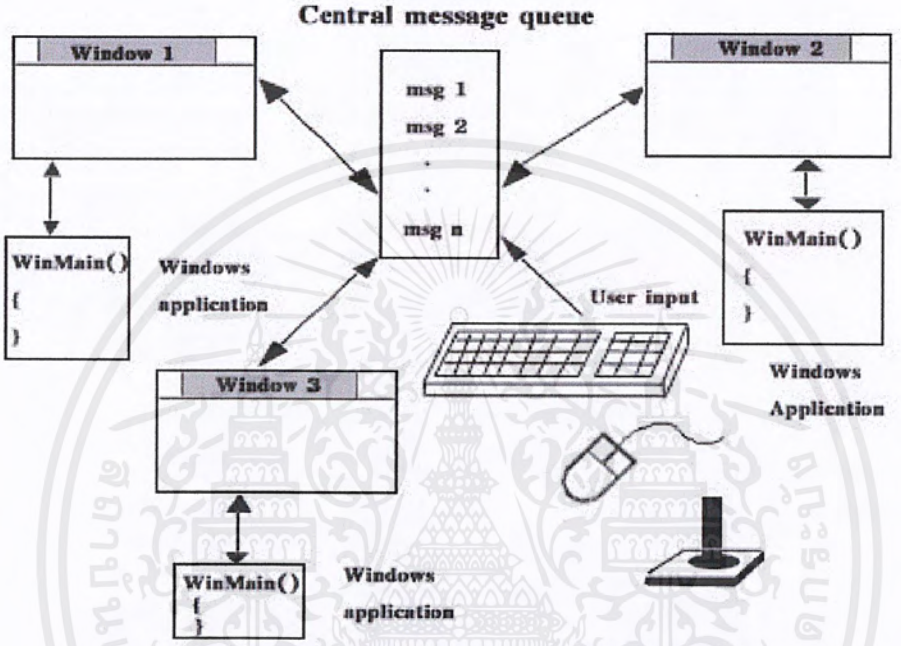
Windows เป็นระบบปฏิบัติการแบบ event-based โดย Message เป็นเสมือนสัญญาณที่จะบอกให้ Windows ต้องทำอะไร ทำอย่างไร หรือเกิดอะไร ตัวอย่าง

Windows event เมื่อผู้ใช้ต้องการเปลี่ยนขนาดของ Window Application Windows ก็จะทำการส่ง Message ไปยัง Application ตัวนั้น ว่าจะเปลี่ยนขนาดอย่างไร

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Keyboard event เมื่อผู้ใช้กด แป้นพิมพ์บน keyboard message ของตัวแป้นพิมพ์ที่กด ก็จะถูกส่งไปยัง application เพื่อกระทำการประมวลผลข้อมูลนั้น และแสดงผลข้อมูลนั้นออกมา เช่น แสดงเมนูย่อย จากเมนูบาร์หลัก

Drawing event ถ้า window ตัวใหม่ถูกแสดงออกมาทับตัวเก่าที่แสดงอยู่ หน้าจอจะต้องทำการวาดใหม่อีกครั้ง โดย Repaint message จะกระทำการในส่วนนี้



รูปที่ 2.3 ระบบการส่ง message ใน Windows

เพราะฉะนั้นอาจจะกล่าวได้ว่า ทุกสิ่งทุกอย่างที่เกิดขึ้นใน Windows ก็คือ message นั้นเอง ดังรูปที่ 2.3 ต่อไปจะกล่าวถึงส่วนประกอบโดยทั่วไป ในการสร้าง windows application ที่ต้องมี ได้แก่

2.2.1 Win32 Libraries

เมื่อเราสร้าง Windows application ขึ้นมา Compiler จะเป็นตัวจัดการเกี่ยวกับ ส่วนของ Libraries ให้เองทั้งหมด ยกเว้นเสียแต่ Libraries ของ Application ที่เราต้องการไม่ได้ถูก ตั้งค่าไว้ตั้งแต่เริ่มต้น เช่น เมื่อต้องการสร้าง Application เกี่ยวกับ Multimedia ซึ่งต้องการ Winmm.Lib Library แต่ใน Directory ของ LIB ในตัวของ Compiler ไม่มี หรือไม่ได้ถูกกำหนดไว้ ผู้ใช้ก็ต้องเพิ่ม หรือกำหนดเข้าไปเอง

เมื่อสร้าง Application ที่ใช้ Direct X ผู้ใช้ก็จะต้องเพิ่ม Libraries ของ direct X เข้าไปใน Application นั้นด้วย

2.2.2 The Include files

โปรแกรมเมอร์ต้อง include files ที่จำเป็นสำหรับ application นั้นไว้ในตัวโปรแกรมด้วยเสมอ และโดยทั่วไป ใน Windows application จะต้องเริ่มต้นด้วย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
# define win32_LEAN_AND_MEAN
#include <windows.h>
#include<windowsx.h>
```

และอาจจะเพิ่ม include files ที่จำเป็นสำหรับ application เข้าไปอีกได้ เช่น Stdio.h หรือ Math.h เป็นต้น

2.2.3 WinMain

เป็น function หลักที่ Windows application ทุก application จำเป็นต้องกำหนดไว้ คล้ายๆ กับ function main() ในภาษา C และ ภาษา C++ การประกาศ Prototype ของ WinMain( ) จะเป็นดังนี้

```
int WINAPI WinMain( HINSTANCE hinstance,
                    HINSTANCE hprevinstance,
                    LPSTR lpcmline,
                    Int ncmdshow );
```

2.2.4 Windows class

โปรแกรมเมอร์ที่ต้องการสร้าง Windows application จำเป็นต้อง กำหนด Windows class ขึ้นมาด้วย ซึ่ง windows class จะอธิบายถึงคุณลักษณะโดยทั่วไปของ windows ทั้งหมด โดยหลังจากโปรแกรมเมอร์ทำการสร้าง windows class ขึ้นมาแล้วโปรแกรมเมอร์ ก็จะสามารถสร้าง window ได้ตามwindows class ที่กำหนดขึ้นมาได้ตามต้องการ และต่อไปนี้คือโครงสร้างของ Windows class

```
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
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2.5 Registering the Windows class

หลังจากที่ทำการ สร้าง windows class ขึ้นมาแล้ว ขั้นต่อไปจะทำการ register class การ register จะทำให้ Windows รู้ว่า มี windows class ที่สร้างขึ้นมา และ สามารถให้โปรแกรมเมอร์สร้าง window ที่มีลักษณะตาม windows class ได้โดยชื่อ ASCII ของ class เพียงอย่างเดียว การ register class กระทำได้โดย

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

2.2.6 การสร้าง Window

การจะสร้าง window ขึ้นมาต้องทำการเรียก function CreateWindow ( ) ที่มีค่า parameters 11 ค่า การประกาศ prototype จะทำได้ดังนี้

```
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 ของวินโดวส์
    int         y,          //ตำแหน่งแนวแกน y ของ วินโดวส์
    int         nWidth,             // ความกว้างของวินโดวส์
    int         nHeight,            // ความสูงของวินโดวส์
    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)
    );
```

2.2.7 การแสดง Window

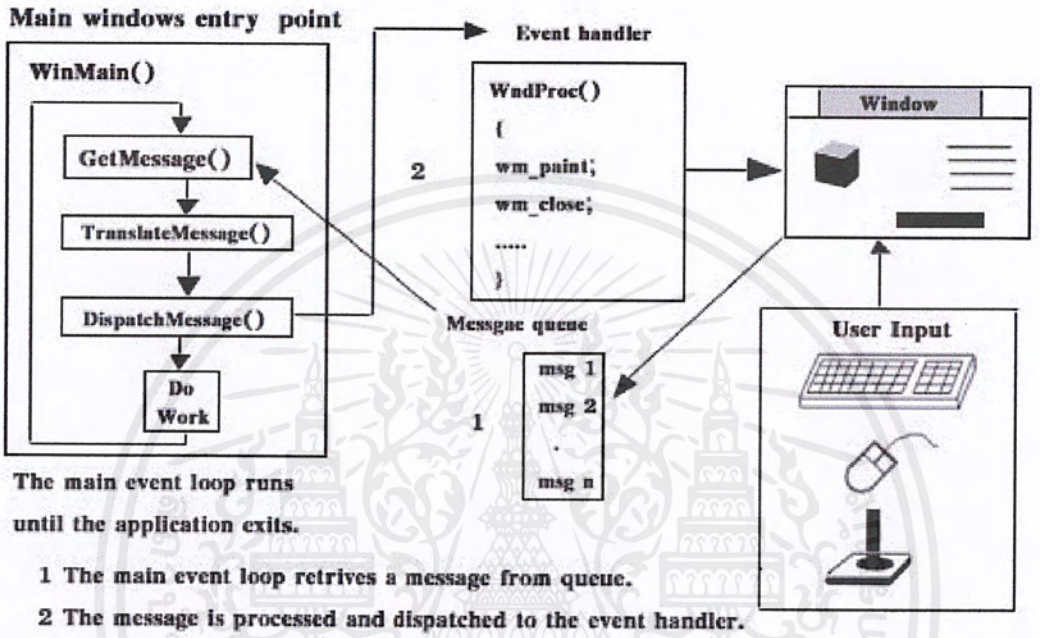
หลังจากที่ทำการ create window เสร็จแล้ว และต้องการ แสดงผลออกมา (ในกรณีที่ ตั้งค่า parameter ของ dwStyle ไม่ใช่ WS_VISIBLE ต้องทำการเรียก function ShowWindow() เสมอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

2.2.8 Event Loop

main event loop คือ loop ที่คอยรับ message record ที่เก็บไว้ จาก message queue แล้วทำการแปลง แล้วส่งต่อไปยัง event handler ดังรูปที่ 2.4



รูปที่ 2.4 The main event loop

โดยมี Function ที่ทำหน้าที่ต่างกัน อยู่ 3 function ใน event loop นี้ได้แก่

2.2.8.1GetMessage

เป็น function ที่ทำหน้าที่รับ message จาก message queue ถ้า กรณี message queue ว่าง function ก็จะทำการรอจนกว่าจะมี message เข้ามา

```
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 TranslateMessage

เป็น function ที่ใช้ แปลง message event ให้อยู่ในรูปพื้นฐาน ของ event เพื่อจะนำไปใช้ในส่วนของ event handler

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

2.2.8.3 DispatchMessage

ทำการส่ง message หลังจาก ทำการ Translate Message ไปยัง event handler เพื่อทำการประมวลผลต่อไป

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

2.2.9 Event handler

event handler หรือ WinProc คือ function ที่ต้องเขียนเพื่อ handle ทุก message ที่ application นั้นสนใจ Windows อาจจะส่ง messageมามากมาย หลายอย่าง แต่อาจมีเพียง 1 message ที่เราสนใจจะ handle การประกาศต้นแบบ ทำได้ดังนี้

```
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 Event Driven Programming

ในการเขียนโปรแกรมบน Windows อย่างแรกที่ควรจะเข้าใจคือ ลักษณะของ Event Driven Programming ลักษณะของ event driven คือ โปรแกรมจะทำงานตาม Event ที่เกิดขึ้นโดย ระบบปฏิบัติการ จะเป็นตัวคอยบอก Program ว่ามี Event อะไรเกิดขึ้นบ้าง หาก Program ได้รับ event ที่ Program สนใจก็จะทำงานตามที่โปรแกรมเมอร์ได้กำหนดไว้

สำหรับ event ในที่นี้ได้รวมถึงทุกอย่างที่เกิดขึ้น ทั้งการได้รับ Input เช่นการกดปุ่ม การปล่อยปุ่ม การ click mouse เป็นต้น และ event เกี่ยวกับ Windows เช่น การที่ window ได้รับ focus การสั่งปิด window หรือ แม้แต่การ redraw window ดังที่ได้กล่าวมาแล้ว ดังนั้น Event driven programming ก็คือการเขียนโปรแกรมที่จะทำงานโดยการ check message ที่ได้รับไปเรื่อยๆ แล้วทำงานตามที่กำหนดเมื่อมี message ที่ Program สนใจ

หากเปรียบเทียบกับการเขียนโปรแกรมโดยทั่วไป การทำงานจะทำงานแบบเรียงลำดับตามผู้เขียนโปรแกรมได้เขียนไว้ แต่สำหรับ event driven programming นั้นคือ ลักษณะที่ผู้ใช้เป็นผู้ควบคุมโปรแกรมมากกว่าที่โปรแกรมจะควบคุมให้ผู้ใช้ทำตาม ยกตัวอย่างเช่น ในการเขียนโปรแกรมบน Dos ถ้าโปรแกรมจะรับ Input 1 key โปรแกรมเมอร์อาจจะใช้ getc() แต่ถ้าเป็น event driven programming เมื่อมีการกดปุ่ม ระบบปฏิบัติการจะส่ง message บอก program ว่าได้มีการกดแป้นพิมพ์ขึ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.4 วิธีการแสดงภาพ

2.4.1 ความรู้ความเข้าใจเบื้องต้นของการแสดงภาพ

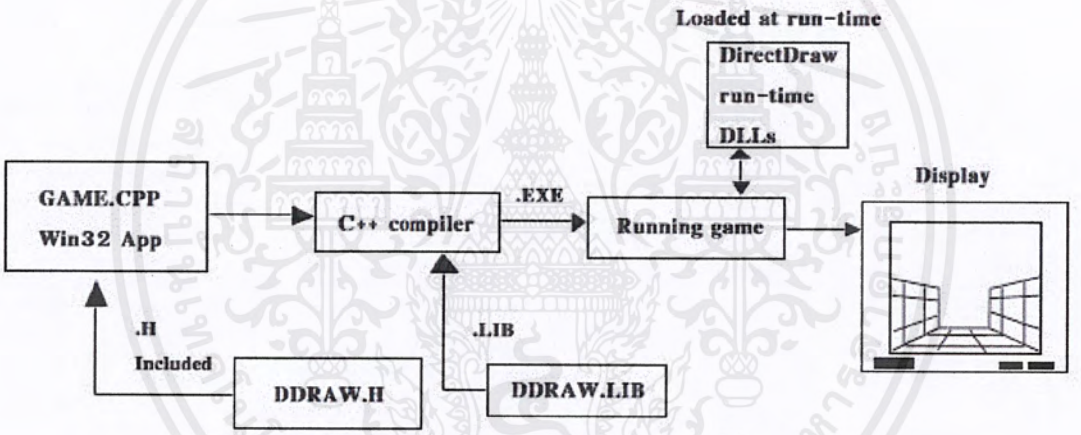
DirectDraw เป็น component ส่วนหนึ่งใน Microsoft DirectX ซึ่งเป็น component ส่วนที่สำคัญที่สุดใน DirectX เพราะว่า DirectDraw มีหน้าที่สำคัญในส่วนที่ใช้วาดหรือติดต่อโดยตรงกับ Hardware ส่วนแสดงภาพ ซึ่งหากขาดส่วน DirectDraw ไป โปรแกรมจะไม่สามารถแสดงภาพได้เลย โดยต่อไปนี้จะกล่าวถึงส่วนที่สำคัญ

DirectDraw คือ ไลบรารี่ตัวหนึ่งนั่นเอง หรือจะอาจจะเรียกได้ว่าเป็น Include file ในภาษาซี ขึ้นอยู่กับเวอร์ชั่นของ DirectX SDK (Software Delveloper Kit) ซึ่งจะถูกสร้างขึ้นมาให้รองรับภาษาต่างๆ โดยหากเป็นของภาษาซี ก็จะเป็นดังนี้

DDRAW.H และ DDRAW.LIB

เวลาเรียกใช้ก็ทำการ Include เข้าไปดังนี้

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



รูปที่ 2.5 เครือข่ายของ Direct Draw

2.4.2 การสร้าง DirectDraw Object

Window และ DirectDraw สามารถทำงานเข้ากันได้เป็นอย่างดี แต่การที่จะนำมาทำงานร่วมกัน จำเป็นต้องมีวิธีการที่มีรูปแบบเฉพาะ โดยการที่เราจะสามารถนำ DirectDraw มาใช้ได้ เราต้องสร้าง DirectDraw Object ขึ้นมาเพื่อเป็นส่วนที่ใช้ติดต่อสื่อสารกับ Hardware แต่ก่อนที่เราจะสามารถสร้าง DirectDraw Object ได้นั้น เราต้องทำการสร้าง Window application ขึ้นมาเป็น working window เสียก่อน เมื่อมี window ที่ใช้ทำงานแล้ว เราจึงจะสามารถสร้าง DirectDraw Object ได้ โดยใช้ฟังก์ชั่น

```
HRESULT DirectDrawCreate(GUID FAR *lpGUID,LPDIRECTDRAW FAR
                         *lplpDD, IUnknown FAR *pUnkOuter);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชั่นนี้จะหมายถึง DirectDrawCreate จะสร้าง DirectDraw Object ไว้ที่ตำแหน่งแอดเดรส lpIpDD และจะได้การแสดงผลแบบ lpGUID (ถ้าใส่ค่าเป็น NULL จะได้การแสดงผลแบบ Default) ฟังก์ชั่นนี้จะทำงานสำเร็จหรือไม่นั้น จะแจ้งผ่านค่า HRESULT มาโดย HRESULT เป็นชนิดข้อมูลชนิดหนึ่งของ DirectX ซึ่งใช้สำหรับตรวจสอบการทำงานของฟังก์ชั่น ซึ่งฟังก์ชั่นส่วนใหญ่ของ DirectX จะทำการคืนค่านี้กลับมา เพื่อตรวจสอบ ค่าที่บ่งบอกถึงความสำเร็จของฟังก์ชั่นก็คือ

DD_OK

หากฟังก์ชั่นคืนค่าอื่นๆที่ไม่ใช่ค่านี้ หมายถึงการทำงานของฟังก์ชั่นมีความผิดพลาด ซึ่งจะแจ้งข่าวสารข้อผิดพลาดมาทาง HRESULT เช่นกัน (แทนที่จะเป็น DD_OK)

ตัวอย่าง

```
LPDIRECTDRAW lpdd;          // ทำการสร้างตัวแปรที่ใช้เก็บค่าแอดเดรสของ
                                    DirectDraw Object
//ทำการสร้าง Object และ ตรวจสอบข้อผิดพลาด
if (DirectDrawCreate(NULL,&lpdd,NULL)!=DD_OK)
{
      //ทำส่วนนี้เมื่อพบข้อผิดพลาด}
}
```

เมื่อได้ทำการสร้าง Object ของ DirectDraw เรียบร้อยแล้ว วิธีการเรียกใช้ฟังก์ชั่น มีวิธีดังนี้

lpdd-> ฟังก์ชั่น( )

เมื่อเราทำงานกับฟังก์ชั่นนั้นๆเสร็จเรียบร้อยแล้ว เราต้องทำการลบ Object นี้ออกจากหน่วยความจำ ได้โดย

lpdd->Release( )

2.4.3 การกำหนดค่า Cooperation level ของ DirectDraw

การที่จะเขียนโปรแกรมเพื่อทำงานบนวินโดวส์ เราจำเป็นจะต้องทำการขอแบ่งทรัพยากรต่างๆจากสิ่งแวดล้อมบนวินโดวส์ ซึ่ง DirectDraw ก็ได้ให้ฟังก์ชั่นเพื่อใช้ในการกำหนดค่าความสำคัญและกำหนดระบบที่จะใช้ของโปรแกรมของเรา ดังนี้

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

HWND hWnd — คือ handle ของวินโดวส์หรือโปรแกรมที่กำลังทำงานด้วย โดยปกติแล้วเราเพียงแต่ทำการกำหนดค่าตัวแปรนี้แล้ว compiler จะทำการสร้างให้เองโดยอัตโนมัติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DWORD dwFlags คือ ค่าที่จะกำหนดให้กับฟังก์ชั่นโดยมีดังนี้

ค่า Flag	ความหมาย
DDSCL_ALLOWMODEX	อนุญาตให้ใช้ ModeX ได้ (320x200 ,320x240) โดยค่านี้จะต้องใช้คู่กับ DDSCL_EXCLUSIVE และ DDSCL_FULLSCREEN
DDSCL_ALLOWREBOOT	อนุญาตให้ใช้ Ctrl+Alt+Delete ในขณะที่อยู่ใน mode Exclusive (fullscreen)
DDSCL_EXCLUSIVE	ทำการกำหนดให้โปรแกรมเป็นแบบ Exclusive (ให้โปรแกรมมีลำดับความสำคัญสูงสุด) โดยจะต้องทำงานคู่กับ DDSCL_FULLSCREEN
DDSCL_FULLSCREEN	ค่านี้จะบอกให้วินโดวส์อนุญาตให้เฉพาะ DirectDraw เท่านั้นที่เป็นผู้เขียนข้อมูลลงบนหน่วยความจำแสดงผลทั้งหมด
DDSCL_NORMAL	บ่งบอกให้วินโดวส์ทราบว่าโปรแกรมนี้จะเป็นเหมือน Application ทั่วๆไป (เช่น Microsoft Word หรือ ACDSee) โดยหากเรากำหนดค่านี้แล้ว เราจะไม่สามารถกำหนดค่า DDSCL_ALLOWMODEX, DDSCL_EXCLUSIVE หรือ DDSCL_FULLSCREEN ได้
DDSCL_NOWINDOWCHANGES	บอกว่าโปรแกรมนี้ไม่สามารถที่จะ minimize หรือ restore โปรแกรมอื่นๆในขณะที่โปรแกรมนี้ทำงานอยู่ได้

ตารางที่ 2.1 **ค่า** Cooperative level **ของ** DirectDraw

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง

```
lpdd -> SetCooperativeLevel(hwnd,DDSCL_NORMAL)
```

หมายถึงกำหนดให้โปรแกรมเราเป็นแบบ Application ทั่วๆไป

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

หมายถึงโปรแกรมเราสามารถใช้ ModeX ได้ และจะเป็นแบบ FullScreen ซึ่งแสดงผลเต็มหน้าจอ และมีความสำคัญสูงสุด และยังสามารถออกจากโปรแกรมได้โดยใช้ปุ่ม Ctrl+Alt+Delete

2.4.4 **การกำหนดโหมดการแสดงผล**

ความสามารถในการเปลี่ยนโหมดการแสดงผลคือความสามารถที่สำคัญที่สุดของ DirectDraw ฟังก์ชั่นที่ใช้เปลี่ยนโหมดการแสดงผลคือ

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

โดยส่วนที่จำเป็นต้องกำหนดค่า มีสามส่วนด้วยกัน

1. dwWidth กำหนดความกว้างของการแสดงผลโดยมีหน่วยเป็นพิกเซล
2. dwHeight กำหนดความยาวของการแสดงผลโดยมีหน่วยเป็นพิกเซล
3. dwBPP กำหนดจำนวนข้อมูลสีต่อหนึ่งพิกเซล (8,16,32 bit)

นอกนั้นให้กำหนดค่าเป็น 0 ทั้งหมด เพราะ compiler จะทำการตรวจสอบค่าที่เหมาะสมให้เอง ทั้งหมดที่กล่าวมานี้เป็นส่วนที่ใช้สำหรับเริ่มการใช้งาน DirectDraw โดยสรุปได้ดังนี้

การเริ่มการใช้งาน DirectDraw มีสามขั้นตอน ดังนี้

- ทำการสร้าง DirectDraw Object
- ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากรโดย SetCoopeeativeLevel
- ทำการกำหนดขนาดการแสดงผล โดยใช้ SetDisplayMode

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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 **การประยุกต์ใช้** DirectDraw

DirectDraw สามารถนำมาใช้ได้ในหลายๆด้าน แต่ด้านที่ถือว่านิยมที่สุดและนำ DirectDraw มาใช้มากที่สุดก็คือด้านการสร้างเกมส์คอมพิวเตอร์ ในปัจจุบัน เกมส์ส่วนมากนำ DirectDraw มาใช้ ในการสร้าง Animation ซึ่ง DirectDraw สามารถเข้าถึง Hardware ด้านการแสดงผลได้โดยตรง ดังนั้นการทำงานต่างๆจึงรวดเร็วมาก และผู้ใช้ไม่จำเป็นต้องเข้าไปยุ่งเกี่ยวกับคำสั่งในระดับ Hardware เลย

2.4.6 **การวาดภาพด้วย** DirectDraw

วิธีที่ใช้ในการวาดภาพ ก็คือการนำภาพที่เราได้วาดไว้แล้วจากโปรแกรมวาดภาพต่างๆ เช่น Adobe Ilustrator หรือจะ Adobe Photoshop หรือเราอาจนำภาพมาจากโปรแกรมสร้างภาพสามมิติเช่น 3Dstudio ก็ได้ แต่ภาพจะต้องเป็นประเภท BMP โดยเราจะนำภาพที่เราได้สร้างไว้แล้วเก็บไว้ในหน่วยความจำ หรือที่เรียกว่า Surface แล้วทำการถ่ายเทข้อมูลไปยังหน่วยความจำที่การ์ดแสดงผล หลังจากนั้น จึงนำแสดงขึ้นจอภาพ โดยรายละเอียดของหลักการจะกล่าวในหัวข้อถัดไป

DirectDraw นิยามหน่วยความจำที่ไว้ใช้แสดงผลด้วยคำว่า Surface โดยเราสามารถนำภาพต่างๆ เก็บไว้ ณ ที่นี้ได้ ไม่ว่าจะเป็นภาพสองมิติ หรือภาพสามมิติก็ตาม ตามปกติเราต้องมีอย่างน้อย 1 Surface คือ Primary Surface ซึ่งเป็น Surface ที่เรามองเห็นขณะนั้นๆบนจอภาพ โดยเราสามารถมี Surface ได้มากแค่ไหนก็ได้โดยขึ้นอยู่กับหน่วยความจำที่มีอยู่

2.4.7 **การสร้าง** Surface

หลักการสร้าง Surface มีดังต่อไปนี้

2.4.7.1 Surface ที่สร้างนั้นมีขนาดเท่าใดก็ได้ ขึ้นอยู่กับภาพที่เราจะนำมาใส่ แต่ Surface ที่เป็น Primary Surface จะต้องมีขนาดเท่ากับขนาดของความละเอียดของจอภาพขณะนั้น เช่นหากเรากำหนดให้โปรแกรมเรามีความละเอียดระดับ 800x600 เราก็ต้องสร้างให้ Primary Surface มีขนาด 800x600 ด้วยเช่นกัน

2.4.7.2 การสร้าง Surface เราสามารถเลือกว่าจะใช้หน่วยความจำที่การ์ดแสดงผล (VRAM) หรือจะใช้หน่วยความจำหลัก (System Memory) ซึ่งถ้าเราสามารถใช้ VRAM ได้จะทำให้การแสดงผลมีความเร็วสูงขึ้น แต่ VRAM มีขนาดที่ไม่ใหญ่นัก ดังนั้นหากเราไม่สามารถใช้ VRAM ได้อีก เราสามารถไปใช้ System Memory แทนได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สำนักหอสมุดกลาง พระจอมเกล้าลาดกระบัง

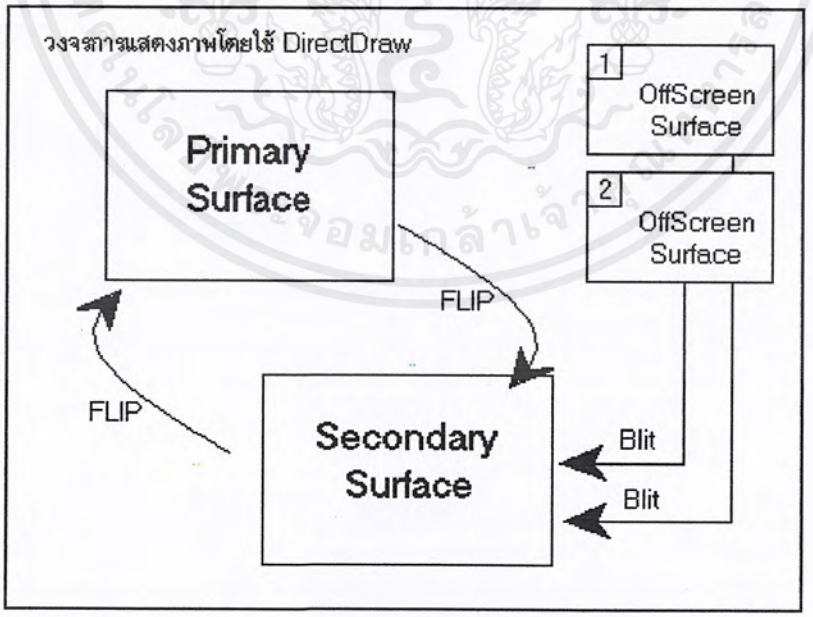
2.4.7.3 เมื่อเราสร้าง Surface ขึ้นมา ระดับความลึกของสีของ Surface นี้จำเป็นต้องเข้ากันได้กับระดับสีของ Primary Surface แต่ตามปกติเมื่อเราสร้าง Surface ขึ้นมาด้วย DirectDraw Object ตัวเดียวกัน ระดับสีของแต่ละ Surface จะเป็นระดับเดียวกันหมดอยู่แล้ว แต่หากเรามี DirectDraw Object สองตัว การที่จะมีระดับสีของแต่ละ Surface ไม่เท่ากันก็เป็นไปได้ และเราก็ไม่สามารถทำการถ่ายข้อมูลระหว่าง Surface ที่มีระดับสีต่างกันได้

โดยทั่วไปเกมส์ที่สร้างขึ้นจะใช้ Surface ดังนี้

1. Primary Surface หรือ FrontBuffer ใช้แสดงภาพที่เป็นหน้าจอปัจจุบัน โดยมีโครงสร้างและขนาดเท่ากับความละเอียดของหน้าจอ

2. Secondary Surface ใช้สำหรับวาดภาพหน้าจอต่อไปขณะที่ Primary Surface กำลังแสดงผลหน้าจอปัจจุบัน หรืออาจเรียกว่า BackBuffer โดย Surface นี้มีโครงสร้างทุกอย่างเหมือน Primary Surface เพื่อที่จะทำงานสลับกับ Primary Surface

3. OffScreen Surface เป็น Surface ที่เราใช้สำหรับเก็บภาพต่างๆที่ใช้ในโปรแกรม ไม่ว่าจะเป็นภาพสำหรับทำภาพเคลื่อนไหว (Animation) หรือภาพนิ่งก็ตาม เพื่อที่จะเพิ่มประสิทธิภาพเมื่อต้องการจะแสดงผลจะได้ไม่ต้องไปอ่านภาพจากไฟล์รูปภาพ ซึ่งหากมีภาพหลายภาพ จะทำให้เสียเวลาในการเข้าถึงข้อมูลในฮาร์ดดิสก์ แต่เราจะเก็บไว้ในหน่วยความจำแทน ทำให้เข้าถึงได้เร็วกว่า ส่วนมากแล้ว เกมจะมีจำนวน Surface นี้มากที่สุด เพื่อใช้เก็บรูปภาพต่างๆ ลักษณะของการแสดงภาพจะเป็นวัฏจักรดังนี้



รูปที่2.6 การแสดงภาพโดย DirectDraw

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

36134

เริ่มจากเรานำภาพที่เราต้องการใช้งานเก็บไว้ใน OffScreenSurface โดยอาจมีได้หลาย Offscreen Surface ต่อไปจึงนำภาพที่เก็บไว้ใน Offscreen ทำการ "Blit" (Bit Block Transfer) ซึ่งก็คือการถ่ายข้อมูลจากใน Offscreen ไปยัง Secondary Surface ณ ตำแหน่งที่อยากให้แสดงผลโดยอ้างอิงจากจอภาพ เมื่อทำการจัดวางภาพต่างๆอย่างดีแล้ว ก็พร้อมที่จะแสดงผลต่อผู้ใช้ ซึ่งเราจะใช้วิธี "FLIP" หรือการสลับหน้าจอกับ PrimarySurface โดยจะทำการสลับเฉพาะตำแหน่งในหน่วยความจำ จากนั้น จึงทำการวาดภาพซึ่งจะใช้แสดงผลต่อไปใน Secondary Surface ในตำแหน่งที่เพิ่งสลับมา และจะเป็นเช่นนี้ไปจนกว่าจะจบโปรแกรม

การสร้าง Surface จะต้องทำการใส่ข้อมูลลงในโครงสร้างข้อมูลแบบ DDSURFACEDESC แล้วจึงทำการเรียกฟังก์ชั่น CreateSurface( ) เพื่อทำการสร้าง Surface นั้นๆ โดยฟังก์ชั่น CreateSurface ( ) มีการรับค่าต่างๆดังต่อไปนี้

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

1. lpDDSurfaceDesc เป็นข้อมูลประเภทพอยเตอร์ชี้ไปยังโครงสร้างข้อมูล DDSURFACEDESC ซึ่งมีการใส่ข้อมูลลงไปเรียบร้อยแล้ว
2. lplpDDSurface ถ้าฟังก์ชั่นทำการสร้าง Surface ได้สำเร็จ จะคืนค่านี้กลับมาเป็นพอยเตอร์ ซึ่งชี้ไปยัง Surface ที่สร้างขึ้น
3. pUnkOuter เป็นพารามิเตอร์ขั้นสูง โดยปกติจะกำหนดให้เป็น NULL

ดังเช่นฟังก์ชั่นอื่นๆ หากฟังก์ชั่นนี้ทำงานสำเร็จจะคืนค่า DD_OK กลับมา แต่ก่อนที่เราจะทำการใช้ฟังก์ชั่น CreateSurface( ) ได้ เราจำเป็นต้องทำการใส่ค่าลงไปยังโครงสร้างข้อมูลที่ชื่อ DDSURFACEDESC (Direct Draw Surface Description) ก่อน โดยมีรายละเอียดของโครงสร้างดังนี้

```
typedef struct _DDSURFACEDESC
{
DWORD dwSize;
DWORD dwFlags;
DWORD dwHeight;
DWORD dwWidth;
Union
{
LONG lPitch;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
DWORD  dwLinearSize;
};
DWORD dwBackBufferCount;

union
{
DWORD dwMipMapCount;
DWORD dwZBufferBitDepth;
DWORD dwRefreshRate;
};
DWORD          dwAlphaBitDepth;
DWORD          dwReserved;
LPVOID lpSurface;
DDCOLORKEY   ddckCKDestOverlay;
DDCOLORKEY   ddckCKDestblt;
DDCOLORKEY   ddckCKSrcOverlay;
DDCOLORKEY   ddckCKSrcBlt;
DDPIXELFORMAT    ddpfPixelFormat;
DDSCAPS          ddsCaps;
} DDSURFACEDESC;
```

จะเห็นได้ว่าเป็นโครงสร้างข้อมูลที่ใหญ่ แต่เราไม่จำเป็นต้องกำหนดทุกค่า มีเพียงบางค่าเท่านั้นที่จะถูกใส่ค่าลงไป ดังนี้

- dwSize คือค่าขนาดของโครงสร้าง DDSURFACEDESC
- dwFlags คือส่วนที่ใช้กำหนดให้ CreateSurface ทราบว่าจะสร้าง Surface แบบใด ซึ่งมีหลายค่าดังนี้

DDSD_ALL คือการกำหนดให้สามารถเรียกใช้การทำงานได้ทุกๆส่วนของ Surface

หากเรากำหนด Flag ค่าไหน การทำงานในส่วนนั้นจะสามารถเรียกใช้งานได้ แต่ถ้าเราเรียก DDSD_ALL เพียงตัวเดียว ทุกตัวจะสามารถทำงานได้ (แต่เราใช้งานจริงๆเพียงบางตัวเท่านั้น) เราจะเพียงบอกให้ทราบว่ามี Flag อะไรบ้าง แต่เราจะอธิบายเพียงบาง Flag ที่ใช้งาน

DDSD_ALPHABITDEPTH , DDSD_BACKBUFFERCOUNT , DDSD_CAPS,
DDSD_CKDESTBLT , DDSD_CKDESTOVERLAY , DDSD_CKSRCBLT,
DDSD_ CKSRCOVERLAY, DDSD_HEIGHT , DDSD_LINEARSIZE ,
DDSD_LPSURFACE , DDSD_MIPMAPCOUNT , DDSD_PITCH,

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DDSD_PIXELFORMAT, DDSD_REFRESHRATE, DDSD_WIDTH,

DDSD_ZBUFFERBITDEPTH

-dwBackBufferCount คือส่วนที่ใช้กำหนดว่าเราจะมี BackBuffer กี่อัน (เราสามารถมี Backbuffer ได้มากกว่า 1 Backbuffer แต่จะทำให้เปลืองหน่วยความจำไปอีก โดยทั่วไปมีเพียงหนึ่ง Surface ก็เพียงพอแล้ว)

-ddsCaps คือส่วนที่ใช้กำหนดคุณสมบัติของ Surface โดยมีโครงสร้างดังนี้

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

-dwcaps เป็นส่วนที่ใช้เก็บค่าคุณสมบัติของ Surface ที่เราต้องการให้มี โดยส่วนใหญ่จะใช้ในระดับสูง แต่เราจะยกขึ้นมาเพียงเท่าที่จำเป็นต้องใช้

ค่า Flags	ความหมาย
DDSCAPS_BACKBUFFER	Surface ถูกกำหนดให้เป็น BackBuffer
DDSCAPS_COMPLEX	บอก DirectDrawSurface ว่า เราจะใช้หลาย Surface มากกว่า Primary Surface อันเดียว
DDSCAPS_FLIP	บอกว่า Surface นี้สามารถ Filp ได้
DDSCAPS_FRONTBUFFER	บอกว่า Surface นี้คือ frontbuffer
DDSCAPS_MODEX	กำหนดให้ Surface เป็น ModeX หรือ 320x200 หรือ 320x240
DDSCAPS_OFFSCREENPLAIN	กำหนดให้ Surface เป็น Offscreen ซึ่งไม่ใช่ Frontbuffer, Backbuffer หรือ Surface ประเภทอื่นๆ โดยปกติจะใช้สำหรับเก็บรูปภาพที่ใช้แสดงภาพเคลื่อนไหวต่างๆ
DDSCAPS_SYSTEMMEMORY	กำหนดให้ Surface นี้จองหน่วยความจำในหน่วยความจำหลัก

ตารางที่ 2.2 Surface Capabilities Flags

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง ตั้งแต่เริ่มจนทำการสร้าง PrimarySurface และ SecondarySurface

```
LPDIRECTDRAW        lpdd;           // ประกาศตัวแปรพ๊อยเตอร์ของ DirectDraw
DDSURFACEDESC       ddsd;           // ประกาศตัวแปรที่ใช้เก็บค่าของโครงสร้างข้อมูล
                                    //DDSURFACEDESC
LPDIRECTDRAWSURFACE lpddsprimary;   //เมื่อ Surface ถูกสร้างขึ้นจะทำการเก็บ
                                    //address ไว้ที่ตัวแปรนี้
//ทำการสร้าง DirectDraw Object
DirectDrawCreate(NULL,&lpdd,NULL);
//ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากร
lpdd->SetCooperatiiveLevel(hwnd, DDSCL_ALLOWREBOOT |
                                 DDSCL_ALLOWMODEX |
                                 DDSCL_FULLSCREEN | DDSCL_EXCLUSIVE);
//ทำการกำหนดขนาดการแสดงผล
lpdd->SetDisplayMode(SCREEN_WIDTH,SCREEN_HEIGHT,SCREEN_BPP);
//สร้าง Primary Surface
//เริ่มต้นโดยทำการกำหนด size
ddsd.dwSize = sizeof(ddsd);
//หลังจากนั้นจึงกำหนด Flag
ddsd.dwFlags = DDSD_CAPS | DDSD_BACKBUFFERCOUNT);
//เมื่อ DDSD_CAPS สามารถเรียกใช้งานได้หมายถึงสามารถกำหนด 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);
```

2.4.8 การนำภาพขึ้นแสดง

เมื่อเราทำการวาดภาพลงบน BackBuffer เรียบร้อยแล้ว จึงทำการ Flip สลับกับ FrontBuffer หรือ PrimarySurface โดยใช้คำสั่ง Flip()

```
HRESULT Flip(LPDIRECTDRAWSURFACE lpDDSurfaceOverride, DWORD dwFlags);
lpDDSurfaceOverride     โดยปกติจะกำหนดให้เป็น NULL
dwFlags                 โดยปกติจะกำหนดให้เป็น DDFLIP_WAIT เพราะเนื่องจากบางครั้ง
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

SecondarySurface ยังอยู่ในสถานะที่กำลังวาดอยู่ และยังไม่สามารถ Flip ได้ โดยค่านี้จะทำให้ฟังก์ชั่นทำการลอง Flip ไปจนกว่าจะ Flip ได้

วิธีใช้เป็นดังนี้

lpddsprimary->Flip(NULL,DDFLIP_WAIT);

2.4.9 การวาดภาพลงบน Surface และการทำ Transparency

แต่ก่อนอื่นที่เราจะทำการ Flip เราจะต้องทำการวาดลงบน Surface ให้ได้ก่อน โดยเราสามารถใช้ฟังก์ชั่น Blt ในการคัดลอกข้อมูล โดยเราสามารถคัดลอกข้อมูลจาก Surface ใดไป Surface ใดก็ได้ โดยมีโครงสร้างของฟังก์ชั่นดังนี้

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

lpDestRect คือ พ๊อยเตอร์ที่ชี้ไปยังตัวแปรประเภท RECT ของ Surface จุดหมาย ซึ่งตัวแปรประเภท RECT นี้มีโครงสร้างดังนี้

```
typedef struct _RECT
{
LONG  top;      //ตำแหน่ง พิกเซล ที่อยู่บนสุดของรูปภาพที่ต้องการวางหรือตำแหน่งที่ต้องการวาง
LONG  left;     //ตำแหน่ง พิกเซล ซ้ายสุด
LONG  right;    //ตำแหน่ง พิกเซล ขวาสุด
LONG  bottom;   //ตำแหน่ง พิกเซล ล่างสุด
}RECT
```

จากโครงสร้างนี้แสดงให้เห็นว่า lpDestRect ก็คือตำแหน่งที่เราจะวางรูปภาพลงไปนั่นเอง แต่หากค่านี้กำหนดเป็น NULL จะหมายถึงใช้ Surface ปลายทางทั้ง Surface

lpDDSrcSurf คือ Surface ที่เป็นรูปภาพต้นทาง ซึ่งอาจเป็น OffScreen ก็ได้

lpSrcRect คือตำแหน่งของรูปภาพที่ต้นทาง หากกำหนดให้เป็น NULL จะหมายถึงใช้รูปทั้งรูป

dwFlags เป็นส่วนที่ใช้ควบคุมการวาดภาพ

lpDDBltFx คือ พ๊อยเตอร์ที่ใช้ชี้ไปที่โครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนดความสามารถพิเศษต่างๆของฟังก์ชั่น (เช่นการหมุนภาพ หรือระบายสี)

dwFlags มีค่าดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ค่า Flag	ความหมาย
DDBLT_COLORFILL	ใช้ร่วมกับ dwFillColor ในโครงสร้างข้อมูล DDBLTFX โดยใส่เป็นสีรูปแบบ RGB (Red,Green,Blue) ซึ่งจะทำการระบายลงบน RECT ที่ surface ปลายทาง
DDBLT_DDFX	ใช้ร่วมกับ dwDDFX ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนดลักษณะพิเศษบางอย่างในการแสดงผล
DDBLT_DDROPS	ใช้ร่วมกับ dwDDROPS ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX โดยใช้กับการทำงานชั้นสูง
DDBLT_KEYDEST	เมื่อกำหนดแล้วจะสามารถกำหนด color key ที่ surface ปลายทางได้
DDBLT_KEYSRC	เมื่อกำหนดแล้วจะสามารถกำหนด color key ที่ surface ต้นทางได้
DDBLT_ROP	ใช้ร่วมกับ dwROP โดยกำหนดในโครงสร้างข้อมูล DDBLTFX โดยทำงานตรงกันข้ามกับ DDBLT_DDROPS
DDBLT_ROTATIONANGLE	ใช้ร่วมกับ dwRotationAngle ในโครงสร้างข้อมูล DDBLTFX โดยใช้หมุนภาพที่จะ blit ไปใส่ที่ surface (มีหน่วยเป็น 1/100 องศา)
DDBLT_WAIT	สั่งใช้ทำการ blit จนกว่าจะ blit สำเร็จ เมื่อใช้คำสั่งนี้ฟังก์ชั่น blit จะไม่ส่งสารข้อผิดพลาดกลับมาถึงแม้จะยังทำการ blit ไม่ได้ก็ตาม

ตารางที่ 2.3 ค่า Control flags ของ Blt( )

ในโครงสร้างข้อมูล DDBLTFX มีข้อมูลจำนวนมากต้องกำหนด แต่ข้อมูลส่วนใหญ่จะใช้สำหรับการทำงานขั้นสูง เช่น Z-buffer (ใช้กับภาพสามมิติ) ดังนั้นเราจะกล่าวถึงเพียงเท่าที่จำเป็นเท่านั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
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;
};
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        DWORD           dwAlphaSrcConstBitDepth;

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

ทั้งหมดนี้คือโครงสร้างข้อมูล DDBLTFX แต่เราจะกำหนดใช้เพียงส่วนน้อยเท่านั้น

dwSize ขนาดของโครงสร้างข้อมูล DDBLTFX นี้ โดยกำหนดเป็นไบท์

dwFillColor สีที่ต้องการระบายลง surface

ddckDestColorkey เป็น color key ของทาง surface ปลายทาง

ddckSrcColorkey เป็น color key ของทาง surface ต้นทาง

dwRotationAngle กำหนดเป็นองศาเพื่อจะใช้หมุนภาพ

การใช้ฟิลด์ที่เกี่ยวข้องกับ color key จำเป็นจะต้องทราบความหมายของ color key เสียก่อนโดย color key ก็คือ สีที่เรากำหนดให้เป็นสีโปร่งใส (Transparency) โดยจะใช้มากในเกมส์คอมพิวเตอร์ทั่วๆ ไป โดยปกติ รูปภาพที่สร้างขึ้นจะมีขอบสี่เหลี่ยม ซึ่งเป็นส่วนเกินเมื่อนำภาพไปสร้างเกมส์ เราจำเป็นต้องตัดส่วนเกินนี้ออก โดยการกำหนดสีที่เราจะใช้เป็น color key แล้วเราก็ใช้สีเดียวกันนี้ระบายลงไปที่ขอบหรือส่วนเกินของภาพ เมื่อทำการ Blit แล้วคอมไพเลอร์จะทำการจับคู่สีที่เป็น color key แล้วจะไม่แสดงสีนั้นลงบนจอภาพ ดังรูป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 2.7 **แสดงความแตกต่างระหว่าง การ** set color key **กับไม่** set

จากรูปที่2.7 รูปเดิมจะเป็นรูปด้านซ้าย คือจะมีขอบเกินมาด้วย แต่เมื่อกำหนดให้ colorkey เป็นสีดำแล้ว เมื่อเราทำการ blit ไปบน surface เราจะได้ภาพดังภาพด้านขวา คือจะไม่มีขอบเกินออกมา ทำให้ภาพดูสมจริงมากขึ้น

ตัวอย่าง การ blit โดยใช้ color key เป็นสีดำ

```
DDBLTFX        ddbltfx;          //ประกาศโครงสร้างข้อมูล ddbltfx
RECT   blit_area,src_area;       //ประกาศ RECT ที่จะวางภาพปลายทาง และภาพต้นทาง
memset(&ddbltfx,0,sizeof(DDBLTFX));
ddbltfx.dwSize=sizeof(DDBLTFX);
DDCOLORKEY  col_key;                    //กำหนดตัวแปรแบบ DDCOLORKEY
col_key.dwColorSpaceLowValue = 0;       // ทำการ set ค่าสีจากสีหนึ่ง (ต้องมีค่าสีต่ำกว่า)
col_key.dwColorSpaceHighValue = 0;      //ไปยังอีกสีหนึ่ง ทำให้เราสามารถมี transparent
                                        //  ได้หลายสี
lpddsback->SetColorKey(DDCKEY_SRCBLT,&col_key);    //ทำการ setcolorkey โดย
// เป็น Source Colorkey ซึ่งก็คือสีที่ไม่ต้องการ ณ ภาพต้นทาง โดยต้องทำการกำหนดที่ BackBuffer
// เพื่อบอกให้ BackBuffer ทราบว่าจะไม่สนใจสีนี้ และไม่ทำการ  blit ลงไป แต่หากเป็น Destination
// Colorkey 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;
```

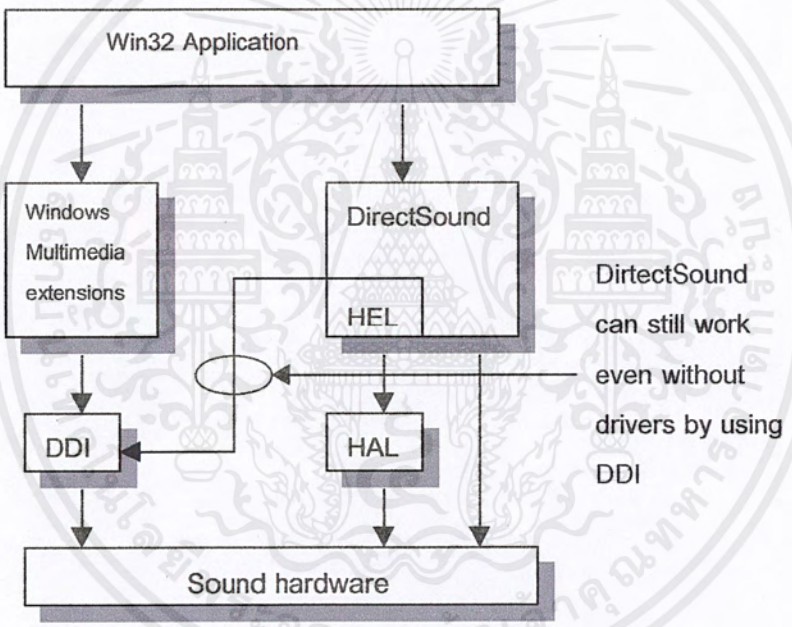
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
lpddsprimary->Blt(&blit_area,lpddsback,,&src_area,DDBLT_KEYSRC | DDBLT_WAIT,
                  NULL);
```

การใช้DirectDrawเราสามารถนำไปใช้ได้หลากหลายงานไม่เฉพาะเจาะจงว่าต้องเป็นเกมส์คอมพิวเตอร์ อยู่ที่ความสามารถในการประยุกต์ใช้ของผู้พัฒนา โดยข้อได้เปรียบต่างๆทำให้ DirectDraw เป็น library ที่น่าสนใจเป็นอย่างยิ่ง

2.5 วิธีการแสดงเสียง

ในส่วนนี้จะกล่าวถึง Direct Sound ที่ใช้ในการสร้างเสียง และมีประโยชน์อย่างมากในกรณีที่ ไม่มี Driver ของ การ์ดเสียง ตัวนั้น โดย Direct Sound จะติดต่อ กับ การ์ดเสียง ได้โดยตรง ไม่ต้องอาศัย Driver ของ การ์ดเสียง ดังรูป



รูปที่ 2.8 การทำงานของ DirectSound

ส่วนประกอบพื้นฐานของ Direct Sound มีอยู่ สามส่วนได้แก่

- ขณะ run-time .DLL จะถูกโหลดขณะเรียกใช้ Direct Sound
- ขณะ compile-time Dsound.Lib จะทำการสร้าง Direct sound application
- ขณะ compile-time Dsound.h จะทำการสร้าง Direct sound application

  ดังนั้นจะต้องทำการเพิ่ม file เหล่านี้เข้าไปใน application ที่ต้องการสร้างด้วย

  ขั้นตอนการสร้าง Direct sound Object

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.1 การสร้าง Direct Sound Object

การที่จะสร้าง Direct sound object จะต้องทำการเรียก Function DirectSoundCreate() ซึ่งกำหนด prototype ดังนี้

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

โดยจะต้องประกาศ pointer เพื่อใช้ ชี้ Direct Sound Object ดังนี้

```
LPDIRECTSOUND   lpds;
```

เมื่อทำการสร้าง Object ของ Direct Sound ขึ้นมาแล้ว และเมื่อเรียกใช้เสร็จก็ต้องทำการคืนค่า Object นั้นด้วยคำสั่ง

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

2.5.2.การตั้งค่าของ cooperation level

จะคล้ายกับการ set cooperation ของ Direct Draw โดยการเรียก Function SetCooperativeLevel( ) โดยมี prototype ดังนี้

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

ค่าของ cooperative level	ความหมาย
DSSCL_NORMAL	ตั้ง เป็นแบบปกติ โดยมีขนาด primary buffer สำหรับ ความถี่เสียง 22 kHz ,Stereo แบบ 8 bit เป็นที่นิยมที่สุด
DSSCL_PRIORITY	อนุญาติให้เปลี่ยนขนาดของ primary buffer ได้ตามต้องการ และสามารถเปลี่ยนรูปแบบเสียงเป็นแบบ 16 bit ได้
DSSCL_EXCLUSIV	เหมือนกับ priority แต่จะได้ยินเสียงเพียงเสียงเดียว
DSSCL_WRITEPRIMARY	เป็นระดับสูงสุด จะควบคุมทุกอย่าง ของ primary buffer เหมาะสำหรับการที่จะสร้างเสียงขึ้นมาเอง

ตารางที่ 2.4 ค่าของ cooperation level DirectSound

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.5.3 Primary และ secondary buffers

Primary buffer จะทำการสร้างเสียงขึ้นมาบนการ์ดเสียง โดยที่ตัวโปรแกรมไม่ต้องไปสนใจ Direct Sound จะเป็นตัวจัดการแทนให้ และไม่จำเป็นต้องสร้าง primary buffer โดยเพียงแค่ set ค่า cooperative level ต่ำสุด Direct sound ก็จะจัดการให้

ส่วนที่จะต้องสนใจจริงๆ ก็คือ Secondary buffer ซึ่งส่วนนี้ จะทำการ เล่นเสียงที่โปรแกรมต้องการ จะมีขนาดเท่าไรก็ได้ ขึ้นอยู่กับหน่วยความจำที่มีอยู่ อย่างไรก็ตาม SRAM ที่อยู่บนการ์ดเสียง ก็เพียงพอที่จะเก็บเสียงได้มากตามที่ต้องการแล้ว

2.5.4 การสร้าง Secondary Buffer

การจะสร้าง secondary sound buffer ต้องทำการเรียก Function CreateSoundBuffer() โดยกำหนด prototype ไว้ดังนี้

```
HRESULT        CreateSoundBuffer(
        LPCDSBUFFERDESC  lpcDSBufferDesc,  // ตัวชี้ไปยัง DSBUFFERDESC
        LPLPDIRECTSOUNDBUFFER  lplpDSBuff,  //ตัวชี้ไปยัง  sound buffer
                IUnknown FAR *pUnkOuter );    // ปกติใช้ NULL
```

ก่อนที่จะสร้าง secondary sound buffer จะต้องกำหนดโครงสร้างของ DirectSoundbuffer เสียก่อน โดยมีรูปแบบดังนี้

```
typedef struct {
DWORD          dwSize;   // ขนาดของโครงสร้างนี้
DWORD   dwFlags;  // ค่า control flags
DWORD   dwBufferBytes;  // ขนาดของ sound buffer ในรูป byte
DWORD   dwReserved;  // ไม่ใช้
LPWAVEFORMATEX   lpwfxFormat;  // รูปแบบ wave format
} DSBUFFERDESC   *LPDSBUFFERDESC;
```

ค่า ของ Control flags	ความหมาย
DSBCAPS_CTRLALL	BUFFER ต้องมีการควบคุมขนาด
DSBCAPS_CTRLDEFAULT	เป็นการตั้งค่า default
DSBCAPS_CTRLFREQUENCY	ควบคุมความถี่ของเสียง
DSBCAPS_CTRLPAN	ควบคุม ความ balance ของเสียง
DSBCAPS_CTRLVOLUME	ควบคุมความดังของเสียง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DSBCAPS_STATIC	บอกว่า buffer จะใช้ สำหรับ ข้อมูลเสียงแบบ static
DSBCAPS_LOCHARDWARE	ใช้ Hardware ทำการสร้างเสียงขึ้นมาและใช้ หน่วยความจำแทน buffer
DSBCAPS_LOCSOFTWARE	บังคับให้ buffer ใช้หน่วยความจำของ software เป็นตัวเก็บ
DSBCAPS_PRIMARYBUFFER	บอกให้รู้ว่า buffer นี้เป็นแบบ primary buffer ตั้งค่านี้ก็ต่อเมื่อต้องการสร้าง primary buffer ด้วยตัวเอง

ตารางที่ 2.5 ค่า control flags ของโครงสร้าง DirectSound Buffer

โดยส่วนใหญ่ จะกำหนดค่า Control flag เป็น DSBCAPS_CTRLDEFAULT | DSBCAPS_STATIC | DSBCAPS_LOCSOFTWARE

ต่อไปเป็นการ กำหนดรายละเอียดของ เสียง โดยสร้าง รูปแบบ ของ Waveformatex ขึ้นมาดังนี้

```
typedef struc {
    WORD wFormatTag ; // ปกติใช้ WAVE_FORMAT_PCM
    WORD nChannels ; // จำนวนของช่องสัญญาณ จะใช้ 1 สำหรับเสียงแบบ mono และ 2 สำหรับ stereo
    DWORD  nSamplesPerSec; // samples per second
    DWORD  nAvgBytesPerSec ; // average data rate
    WORD  nBlockAlign ;          // nchannels * bytespersample
    WORD  wBitsPerSample ; // bits per sample
    WORD  cbSize ; // ตั้งเท่ากับ 0
      } WAVEFORMATEX;
```

2.5.5 การเขียนข้อมูลลงบน Secondary Buffer

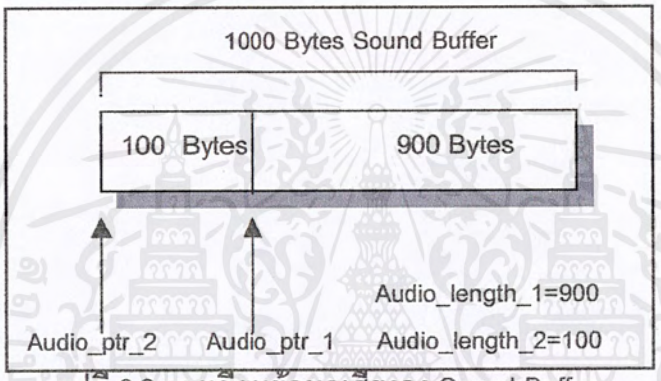
ใน Direct Draw Surface จะต้องทำการ lock surface memory ก่อน แล้วจึงเขียน ใน Direct Sound ก็เช่นเดียวกัน แต่ต่างกันที่ Direct sound มี 2 พอยเตอร์ ที่ใช้ในการเขียน โดยพอยเตอร์แรก จะใช้เขียนข้อมูล ในส่วนแรก และข้อมูลที่เหลือ สำหรับพอยเตอร์ที่ สอง โดยมีการกำหนด prototype Lock( ) ไว้ดังนี้

```
HRESULT  Lock(
    DWORD dwWriteCursor,
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

ตัวอย่าง เช่น ต้องการสร้าง Sound buffer ขนาด 1000 bytes เมื่อทำการ lock buffer เพื่อจะทำการเขียน จะต้องใช้ สองพอยเตอร์ในการอ้างอิง โดยขนาดของพอยเตอร์ตัวแรก อาจจะมีขนาด 900 bytes ส่วน ที่เหลือ 100 bytes จะเป็นของพอยเตอร์ตัวที่สอง ดังรูปภาพที่ 2.9



รูปที่ 2.9 การเขียนข้อมูลเสียงลง Sound Buffer

หลังจากทำการ สร้าง Sound Buffer และ โหลด sound ทั้งหมด ขั้นต่อไปก็จะเป็นการ นำ sound ที่พร้อมนำมาแสดง โดย Direct X มี Function มากมายในการแสดงเสียงที่จำเป็นได้แก่

2.5.6 การแสดงเสียง

ใช้ ฟังก์ชั่น Play() ในการแสดงเสียงจาก sound buffer โดย มีตัวต้นแบบดังนี้

```
HRESULT Play(
DWORD dwReserved1, DWORD dwReserved2,    // กำหนดเป็น 0 ทั้งสองค่า
DWORD dwFlags);   // ควบคุม flags ในการเล่น ถ้าต้องการเล่นวนรอบ ตั้งค่าเป็น
                  //DSBPLAY_LOOPING แต่ถ้าต้องการเล่นเพียงครั้งเดียว ตั้งค่า เป็น 0
```

2.5.7 การหยุดเสียง

ถ้าต้องการหยุดเสียงที่กำลัง เล่นอยู่ ใช้ ฟังก์ชั่น Stop ( ) โดยกำหนดตัวต้นแบบดังนี้

```
HRESULT Stop( ) ;
```

2.5.8 การตั้งระดับความดังของเสียง

ถ้าต้องการปรับความดังเสียง จะเรียกใช้ ฟังก์ชั่น SetVolume( ) มีการกำหนด ต้นแบบดังนี้

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6 วิธีการรับข้อมูล จาก Mouse และ Keyboard

DirectInput เป็นไลบรารีที่ช่วยให้โปรแกรมเมอร์ทำการติดต่อและเรียกใช้อุปกรณ์ในการรับอินพุทชนิดต่างๆได้อย่างมีประสิทธิภาพ direct input ครอบคลุมอุปกรณ์ในการรับอินพุทมาตรฐานทุกชนิด อันประกอบไปด้วย

- คีย์บอร์ดมาตรฐาน
- เมาส์ ทั้ง 2 และ 3 ปุ่ม
- จอยสติ้ก จอยโยก ทั้งที่เป็นแบบอนาลอค และ ดิจิตอล
- อุปกรณ์ควบคุมการขับเคลื่อน ใช้สำหรับจำลองการขับรถทั้งที่เป็นแบบอนาลอค และดิจิตอล
- อุปกรณ์ที่ตอบสนองแรงสั่นสะเทือน จอยสติ้กหรืออุปกรณ์อื่นๆที่สามารถคำนวณเพื่อทำให้เกิดแรงสั่นโต้ตอบกับผู้ใช้
- virtual headgear

2.6.1 การใช้ DirectInput

direct input ประกอบไปด้วยรันไทม์ไลบรารี และ 2 คอมไพล์ไฟล์ DINPUT.LIB DINPUT.H สำหรับโปรแกรมหรือโปรเจคที่ต้องการใช้ความสามารถของ direct input จะต้องเพิ่มไฟล์ตามที่ได้กล่าวมาข้างต้นไว้ในโปรเจคนั้นๆด้วย

ขั้นตอนในการใช้ direct input

- สร้างออปเจค direct input หลักด้วย DirectInputCreate( ).
- สร้างออบเจคของอุปกรณ์อินพุทด้วย CreateDevice( ).
- กำหนดลักษณะในการใช้งานของแต่ละอุปกรณ์ด้วย SetCooperativeLevel( ).
- กำหนดรูปแบบข้อมูลของแต่ละอุปกรณ์ด้วย SetDataFormat( ). และทำการกำหนดคุณสมบัติพิเศษของอุปกรณ์ต่างๆด้วย SetProperty( ).
- ทำการจองอุปกรณ์ต่างๆด้วย Acquire( ).
- รับข้อมูลจากแต่ละอุปกรณ์ที่ได้ทำการจองไว้แล้วด้วย GetDeviceState( ).

การอ่านข้อมูลของแต่ละอุปกรณ์จะถูกอ่านเป็นเรคคอร์ด แต่ละเรคคอร์ดจะถูกแปลงเป็นข้อมูล เพื่อให้โปรแกรมเมอร์นำข้อมูลต่างๆเหล่านี้ไปใช้ แต่ละอุปกรณ์ก็จะมีรูปแบบของเรคคอร์ดที่แตกต่างกัน ถึงแม้ว่าจะมีรูปแบบที่แตกต่างกัน แต่เรคคอร์ดเหล่านี้ก็มีลักษณะใกล้เคียงกันมาก ทำให้เกิดการสร้างข้อมูลอินพุทที่เป็นรูปแบบหลักๆขึ้น

โปรแกรมเมอร์สามารถร้องขอข้อมูลอินพุทจาก DirectInput ได้ 2 วิธี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1. **แบบทันทีทันใด**(immediate input) ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของอุปกรณ์อินพุทขณะนั้น

2.**แบบมีการพักข้อมูล**(buffered input) ข้อมูลที่ถูกเก็บเป็นดาต้าเบสของเหตุการณ์ที่เกิดขึ้นตั้งแต่การร้องขออินพุทครั้งสุดท้าย

ในการใช้ DirectInput โปรแกรมเมอร์ไม่ต้องกังวลเกี่ยวกับ แมสเสท หรือการตอบสนองกับผู้ใช้ว่าจะเป็นไปอย่างเชื่องช้า เพราะ DirectInput จะทำงานโดยตรงกับไดร์เวอร์ของอุปกรณ์อินพุทนั้นๆ

2.6.2 การติดต่อกับคีย์บอร์ดผ่านทาง DirectInput

2.6.2.1 DirectInput Object จะสร้างออปเจคหลักเพื่อใช้ในการจัดการ ติดต่อกับอุปกรณ์อินพุทชนิดอื่น โดยออปเจคหลักนี้ถูกอ้างถึงด้วย

```
LPDIRECTINPUT        lpdi;
```

ทำการสร้างออปเจคหลักด้วยฟังก์ชั่น DirectInputCreate( ) โดยมีรูปแบบการของฟังก์ชั่นดังนี้

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

ฟังก์ชั่นจะคืนค่า DI_OK(DirectInput Okey) ถ้าการเรียกใช้ฟังก์ชั่นนี้ผ่านและจะคืนค่าเป็นค่าผิดพลาดอื่นๆถ้าฟังก์ชั่นนี้ไม่ผ่าน สำหรับค่าพารามิเตอร์ต่างๆของ DirectInputCreate มีดังนี้

hinst	ใช้อ้างถึงแอพพลิเคชั่นที่ต้องการใช้งาน DirectDraw ให้แน่ใจว่าค่า hinst นี้ถูกส่งค่ามาจากค่า hinstance ใน WinMain( )
dwversion	กำหนดเวอร์ชั่นของ DirectInput ว่าจะใช้เวอร์ชั่นใด มีไว้เพื่อให้โปรแกรมเมอร์ที่ต้องการใช้ DirectInput เวอร์ชั่นเก่า แต่โดยทั่วไปจะถูกกำหนดค่าเป็น DIRECTINPUT_VERSION ซึ่งหมายความว่า DirectInput ที่จะใช้เป็นเวอร์ชั่นล่าสุด
lplpDirectInput	เป็นพอยเตอร์ที่อ้างอิงไปยังออปเจคหลักของ DirectInput
punkOuter	ถูกกำหนดเป็น NULL เสมอ

ตัวอย่าง การสร้างออปเจคหลักของ DirectInput

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อไม่ต้องการใช้งาน ออปเจคหลักของ DirectInput ก็ทำการเคลียร์ค่าด้วยฟังก์ชั่น Release( ) เช่น

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

2.6.2.2 **ทำการสร้างออปเจคของคีย์บอร์ด** ด้วยฟังก์ชั่น CreateDevice( ) ฟังก์ชั่นนี้จะใช้เมื่อต้องการสร้างออปเจคของอุปกรณ์อินพุทใดๆ รูปแบบการใช้งานฟังก์ชั่น CreateDevice( ) มีดังนี้

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

ค่าพารามิเตอร์ต่างๆของ DirectInput มีดังนี้

rguid คือค่า GUID(Globally Unique Identifier) เป็นหมายเลขของอุปกรณ์อินพุทที่ต้องการสร้าง โดยปกติจะใช้ฟังก์ชั่น EnumDevice( ) เพื่อหาค่า GUID ของอุปกรณ์อินพุทนั้นๆ แต่ใน DirectInput มีการกำหนดค่า GUID ของคีย์บอร์ดและเมาส์ไว้ให้เรียบร้อยแล้ว โดยจะมีค่าเป็น

GUID_SysKeyboard สำหรับคีย์บอร์ดมาตรฐาน

GUID_SysMouse สำหรับเมาส์มาตรฐาน

อย่างไรก็ตามการใช้ค่า GUID ที่ถูกประกาศไว้แล้วนี้จำเป็นต้องเพิ่ม #define INITGUID ไว้ส่วนบนของทุกไฟล์ C/C++ ที่มีการประกาศ OBJBASE.H เพื่อคอมไพเลอร์จะรวมข้อมูลเกี่ยวกับCOM และค่า GUID ที่ได้ถูกประกาศไว้แล้ว

lplpDiDev เป็นตำแหน่งที่พอยเตอร์ชี้เพื่ออ้างอิงถึงอุปกรณ์อินพุทชนิดนั้นๆ

punkOuter ถูกกำหนดให้เท่ากับ NULL เสมอ

ตัวอย่าง การสร้างออปเจคของคีย์บอร์ด

```
#define INITGUID
#include <OBJBASE.H>
#include "DINPUT.H"
//ทำการสร้างออปเจคหลักตามที่ได้กล่าวมาในขั้นก่อนหน้านี้หลังจากสร้างออปเจคหลักแล้วจึงทำ
//การสร้างออปเจคของคีย์บอร์ด
LPDRECTINPUTDEVICE      lpdikey;
lpdi->CreateDevice(GUID_SysKeyboard, &lpdikey, NULL);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.2.3 กำหนดลักษณะการใช้งานของคีย์บอร์ด

แต่ละอุปกรณ์ก็จะมีลักษณะการใช้งานเป็นของตัวเอง SetCooperativeLevel( ) มีรูปแบบการเรียกใช้ดังนี้

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

ค่าพารามิเตอร์ต่างๆของ SetcooperativeLevel( ) มีดังนี้

hwnd ใช้อ้างถึงวินโดว์ของแอพพลิเคชั่นนั้นๆ

dwFlag เป็นค่าที่กำหนดลักษณะของอุปกรณ์อินพุทนั้นๆมีทั้งหมด 4 ค่า

ค่าของ dwFlag	ความหมาย
DISCL_BACKGROUND	กำหนดให้แอพพลิเคชั่นสามารถใช้อุปกรณ์อินพุทนั้นๆได้แม้ว่าแอพพลิเคชั่นจะอยู่ในโหมด background
DISCL_FOREGROUND	แอพพลิเคชั่นไม่สามารถเรียกใช้อุปกรณ์อินพุทนั้นๆได้ในโหมด background เพราะอุปกรณ์อินพุทจะถูกปล่อยทันทีเมื่อแอพพลิเคชั่นเปลี่ยนเป็นโหมด background
DISCL_EXCLUSIVE	หลังจากทำการจองอุปกรณ์อินพุทนั้นๆแล้ว จะมีได้เพียงแอพพลิเคชั่นเดียวที่ร้องขอ DISCL_EXCLUSIVE ส่วนแอพพลิเคชั่นอื่นจะต้องร้องขอการใช้งานอุปกรณ์อินพุทแบบ DISCL_NONEXCLUSIVE ได้
DISCL_NONEXCLUSIVE	แอพพลิเคชั่นสามารถใช้งานอุปกรณ์อินพุทร่วมกันได้

ตารางที่ 2.6 ค่า dwFlag ลักษณะการใช้งาน ของ Keyboard

สำหรับเกมที่ใช้โหมดแบบ full screen จะมีการกำหนดลักษณะของอุปกรณ์อินพุทเป็นแบบ DISCL_BACKGROUND และ DISCL_NONEXCLUSIVE

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง การใช้ฟังก์ชั่น SetCooperativeLevel( )

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

2.6.2.4 กำหนดรูปแบบข้อมูลของคีย์บอร์ด

เพื่อกำหนดว่าข้อมูลที่จะรับเข้ามาจากคีย์บอร์มีรูปแบบอย่างไรด้วยฟังก์ชั่น SetDataFormat( ) โดยมีรูปแบบการเรียกใช้ดังนี้

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

SetDataFormat มีพารามิเตอร์เพียงตัวเดียวคือ lpdf พารามิเตอร์นี้จะใช้โครงสร้างของ DIDATAFORMAT ตามที่ได้แสดงด้านล่างนี้

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

แต่ DirectInput ได้กำหนดรูปแบบของข้อมูลสำหรับอุปกรณ์อินพุทมาตรฐานไว้เรียบร้อยแล้วนั่นคือ

c_dfDIKeyboard กำหนดรูปแบบข้อมูลสำหรับคีย์บอร์ดมาตรฐาน

c_dfDIMouse กำหนดรูปแบบข้อมูลสำหรับเมาส์มาตรฐาน

c_dfDIJoystick กำหนดรูปแบบข้อมูลสำหรับจอยสติ้กมาตรฐาน

ดังนั้นโปรแกรมเมอร์สามารถใช้รูปแบบข้อมูลที่ถูกกำหนดขึ้นไว้แล้วถ้าแอพพลิเคชั่นนั้นใช้อุปกรณ์อินพุทมาตรฐาน

ตัวอย่าง การใช้ฟังก์ชั่น SetDataFormat

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.6.2.5 ทำการจองคีย์บอร์ดด้วยฟังก์ชั่น Acquire( )มีรูปแบบการเรียกใช้ดังนี้

```
HRESULT Acquire( );
```

ตัวอย่าง การใช้ฟังก์ชั่น Acquire( );

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

2.6.2.6 ทำการอ่านข้อมูลจากคีย์บอร์ด

สำหรับลักษณะข้อมูลของคีย์บอร์ดที่ถูกกำหนดโดย DirectInput มีลักษณะเป็น state data นั่นหมายความว่าเมื่ออ่านข้อมูลขึ้นมาจากคีย์บอร์ด ข้อมูลที่ได้จะเป็นสถานะปัจจุบันของคีย์บอร์ด สำหรับการอ่านข้อมูลจากคีย์บอร์ดจะใช้ฟังก์ชั่น GetDeviceState( ) ฟังก์ชั่นนี้จะอ่านค่าสถานะปัจจุบันของคีย์บอร์ดขึ้นมา หรือสิ่งที่คีย์บอร์ดได้ทำครั้งสุดท้ายในอินพุทลูปโดยมีรูปแบบการเรียกใช้ฟังก์ชั่นดังนี้

```
HRESULT GetDeviceState (
        DWORD       cbData, //ขนาดของเรคคอร์ดข้อมูล
        LPVOID      lpvData); // พอย์เตอร์ชี้ไปยังตำแหน่งที่เก็บข้อมูล
```

ตัวอย่าง การใช้ ฟังก์ชั่น GetDeviceState

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

DirectInput จะมีค่าคงที่ของแต่ละคีย์บนคีย์บอร์ดเพื่อให้โปรแกรมเมอร์สามารถตรวจสอบสถานะของคีย์บนคีย์บอร์ดได้ว่าเป็นอย่างไร โดยค่าคงที่นี้จะขึ้นต้นด้วย DIK_ แล้วตามด้วยค่าคีย์ ตามตารางด้านล่างนี้

ค่า	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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DIK_UP	ปุ่มลูกศรชี้ขึ้น
DIK_DOWN	ปุ่มลูกศรชี้ลง
DIK_LEFT	ปุ่มลูกศรชี้ไปทางซ้าย
DIK_RIGHT	ปุ่มลูกศรชี้ไปทางขวา
DIK_PRIOR	ปุ่ม PageUp
DIK_NEXT	ปุ่ม PageDown

ตารางที่ 2.7 ค่าคงที่ของแต่ละคีย์บนคีย์บอร์ด

หลังจากทำการอ่านค่าสถานะของคีย์บอร์ดแล้ว จะสามารถตรวจสอบได้ว่าคีย์ที่สนใจมีสถานะเป็นอย่างไรด้วยการตรวจสอบค่าบิตของคีย์นั้นๆถ้าคีย์ถูกกด ค่าบิต 0x80 จะมีค่าเป็น 1 แต่ถ้าคีย์นั้นๆไม่ได้ถูกกด หรือคีย์นั้นๆถูกปล่อยไปแล้วค่าบิต 0x80 จะมีค่าเป็น 0

ตัวอย่าง การตรวจสอบค่าของคีย์

```
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++;
```

จากตัวอย่างเป็นการตรวจสอบว่าคีย์ลูกศรชี้ลง หรือ ลูกศรชี้ไปด้านขวาถูกกดหรือไม่

2.6.3 การรับข้อมูลจากเมาส์

ขั้นตอนต่างๆในการใช้ DirectInput เพื่อติดต่อกับเมาส์นั้นจะคล้ายกับขั้นตอนในการใช้ DirectInput เพื่อติดต่อกับคีย์บอร์ดจะต่างกันเพียง การอ่านค่าตำแหน่งของเมาส์เท่านั้น ขั้นตอนในการสร้างออปเจคเพื่อใช้ในการติดต่อและอ่านค่าจากเมาส์มีดังนี้

1. สร้างออปเจคเมาส์
2. ทำการกำหนดลักษณะการใช้งานของเมาส์
3. กำหนดรูปแบบข้อมูลที่ได้จากการอ่านค่าสถานะของเมาส์
4. ทำการจองเมาส์

2.6.3.1 สร้างออปเจคเมาส์

สำหรับการสร้างออปเจคเมาส์นี้ จะใช้ฟังก์ชั่น CreateDevice( ) พารามิเตอร์ GUID ที่จะใช้เป็นหมายเลขของเมาส์จะใช้ค่า GUID ที่ถูกประกาศไว้แล้วโดย DirectInput นั่นคือใช้ค่า GUID_SysMouse

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง การเรียกใช้ฟังก์ชั่น CreateDevice( )

```
LPDIRECTINPUTDEVICE          lpdimouse;
//ทำการสร้างเมาส์ออปเจค
lpdi->CreateDevice(GUID_SysMouse, &lpdimouse, NULL);
```

2.6.3.2 ทำการกำหนดลักษณะการใช้งานของเมาส์

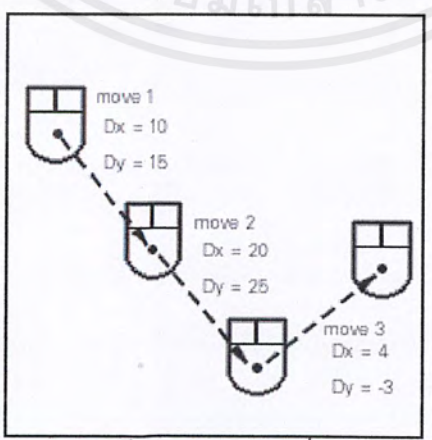
หลังจากการ CreateDevice ( ) ได้สำเร็จ ขั้นต่อไปก็จะทำการกำหนดลักษณะการใช้งานของอุปกรณ์อินพุทกับแอพพลิเคชั่นหนึ่งๆ อุปกรณ์อินพุทในที่นี้คือเมาส์ จะทำการกำหนดลักษณะการใช้งานได้ด้วยฟังก์ชั่น SetCooperativeLevel( ) สำหรับพารามิเตอร์ที่ใช้ในฟังก์ชั่นนี้จะเหมือนกับพารามิเตอร์ที่ใช้ในการกำหนดลักษณะการใช้งานให้กับคีย์บอร์ด โดยมีค่าแฟลก 4 ค่า นั่นคือ

- DISCL_BACKGROUND
- DISCL_FOREGROUND
- DISCL_EXCLUSIVE
- DISCL_NONEXCLUSIVE

ค่าแฟลกทั้ง 4 ค่านี้มีความหมายเหมือนกับค่าแฟลกที่ใช้กำหนดลักษณะการใช้งานของคีย์บอร์ด สำหรับเกมที่ใช้ในโหมด Full Screen จะกำหนดค่าแฟลกเป็น DISCL_BACKGROUND และ DISCL_NONEXCLUSIVE เพื่อหลีกเลี่ยงข้อผิดพลาดจากการจองอุปกรณ์อินพุทชนิดเดียวกัน

2.6.3.3 ทำการกำหนดรูปแบบของข้อมูลที่ได้รับ

ขั้นตอนนี้เหมือนกับขั้นตอนในการกำหนดรูปแบบของข้อมูลให้กับคีย์บอร์ด จะมีการใช้รูปแบบของข้อมูลที่ได้ถูกประกาศไว้แล้ว คือ c_dfDIMouse แต่รูปแบบของข้อมูลที่ได้จากการใช้ c_dfDIMouse จะเป็นแบบสัมพันธ์(relative) นั่นหมายความว่าทุกๆครั้งที่มีการร้องขอข้อมูลจากเมาส์ ค่าตำแหน่งของเมาส์จะอยู่ในรูปผลต่าง (เปลี่ยนจากตำแหน่งที่เมาส์ชี้อยู่เท่าไหร่)



รูปที่ 2.10 แสดงการเก็บค่าตำแหน่งเมื่อทำการเลื่อนเมาส์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะใช้ฟังก์ชั่น SetDataFormat( ) เพื่อทำการกำหนดรูปแบบของข้อมูลที่ได้จากการอ่านค่าสถานะของเมาส์

```
lpdimouse->SetDataFormat(&c_dfDIMouse);
```

2.6.3.4 ทำการจองเมาส์

DirecInput จะทำการจองเมาส์ไว้ให้แอพพลิเคชั่นนั้นๆด้วยฟังก์ชั่น Acquire( )

```
lpdimouse->Acquire( );
```

2.6.3.5 การอ่านค่าจากเมาส์

จะสามารถอ่านค่าจากเมาส์ได้ด้วยฟังก์ชั่น GetDeviceState( ) แต่ข้อมูลที่ส่งคืนกลับมาจะต่างจากข้อมูลที่ถูกส่งคืนจากคีย์บอร์ด ข้อมูลที่ได้จากการอ่านค่าเมาส์จะถูกส่งคืนมาในรูปของ DIMOUSESTATE

```
typedef struct {
        LONG   lx;      //ค่าแกน x ของเมาส์
        LONG   ly;      //ค่าแกน y ของเมาส์
        LONG   lz;      //ค่าแกน z ของเมาส์ หมายถึงปุ่มที่วงล้อ
        BYTE   rgbButtons[4]   //ปุ่มของเมาส์ บิตสูงจะหมายถึงปุ่มนั้นถูกกด
} DIMOUSESTATE,      *LPDIMOUSESTATE
```

จากโครงสร้างข้อมูลแบบนี้ ค่าตำแหน่งของแกน x y และค่าของวงล้อจะถูกส่งมายังพารามิเตอร์ lx ly และ lz และค่าสถานะของปุ่มจะถูกส่งมายัง rgbButtons[4] เป็นอาร์เรย์ขนาด 4 ไบต์ สำหรับวิธีการอ่านค่าสถานะจากเมาส์จะใช้คำสั่ง GetDeviceState( )

```
DIMOUSESTATE                mouse_state;
lpdimouse->GetDeviceState(sizeof(DIMOUSESTATE),(LPVOID)&mouse_state);
```

เมื่อได้ข้อมูลมาแล้วการตรวจสอบว่าข้อมูลที่ได้มานั้นมีค่าเช่นไรสามารถตรวจสอบได้ดังนี้

```
// เป็นการตรวจสอบว่ามีการกดเมาส์ปุ่มซ้าย ถ้าพบว่าปุ่มถูกกดก็ให้ทำงานตามหลังวงเล็บปีกกา
if (mouse_state.rgbButtons[0])
{
}
// เป็นการตรวจสอบว่ามีการกดเมาส์ปุ่มขวา ถ้าพบว่าปุ่มถูกกดก็ให้ทำงานตามหลังวงเล็บปีกกา
if (mouse_state.rgbButtons[1])
{
}
```

เมื่อต้องการยกเลิกการใช้เมาส์จะต้องทำการคืนเมาส์ให้สู่ระบบ ด้วยฟังก์ชั่น Unacquire( ) และทำการลบออปเจคของเมาส์ทิ้งไปด้วยฟังก์ชั่น Release( )

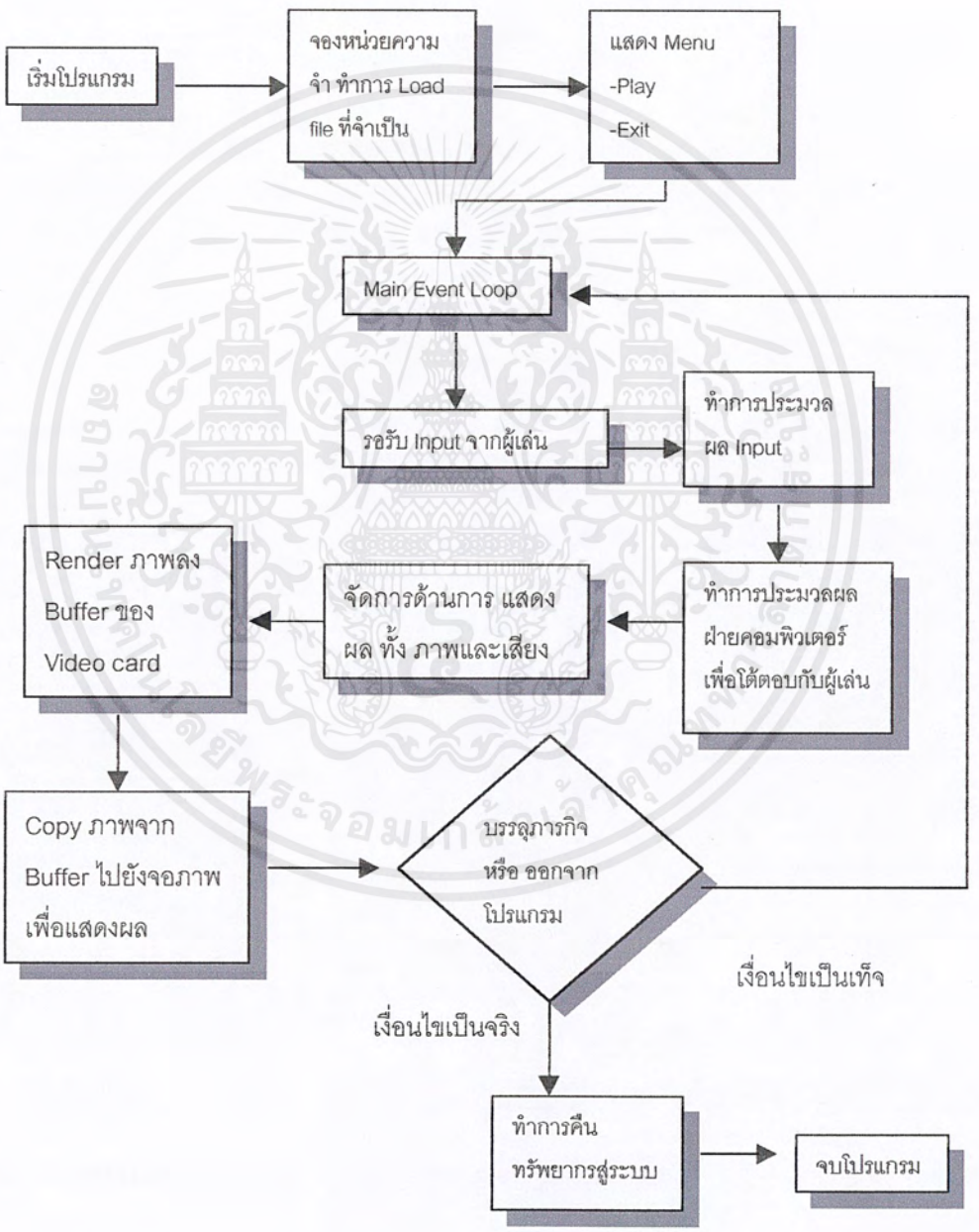
```
lpdimouse->Unacquire( );
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

lpdimouse->Release( );

2.7 วงจรเกม

หลักการโดยทั่วไปจะทำการสร้างวงจรของเกม(Game Loop)ขึ้นมา เพื่อรอรับ Input จากผู้เล่น ประมวลผล แล้วทำการแสดงผล และกลับไปรอรับ Input ประมวลผล และแสดงผล เป็นเช่นนี้เรื่อยๆ จนกว่าผู้เล่นจะสามารถทำให้เงื่อนไขการวน Loop เป็นจริง เป็นต้นว่าในปัญหาพิเศษนี้ จะต้องบรรลุเป้าหมาย หรืออาจกดปุ่ม exit ดังรูปที่ 2.11



รูปที่ 2.11 วงจรเกม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.8 โครงร่างการเขียนเกม

หลังจากที่ได้ศึกษาในส่วนของ Window Programming ไปบ้างแล้ว ในส่วนนี้ จะกล่าวถึงโครงร่างการเขียนเกม (Game console) ที่ถูกออกแบบมาเพื่อใช้ในการเขียน เกมโดยตรง โดย game console จะทำงานภายใน Winmain ดังรูปที่ 2.12

จุดประสงค์ของการสร้าง Game Console ขึ้นมาก็เพื่อจะสร้างเกมที่รันบนระบบปฏิบัติการ วินโดวส์ และใช้ Direct X ได้ด้วยความสะดวก ดังนั้น ในปัญหาพิเศษนี้ จะกระทำบน Game Console ซึ่ง Game Console จะแบ่ง ออก เป็น 3 ฟังก์ชั่น ได้แก่ Game_Init( ) , Game_Shutdown( ) และ Game_Main( )

2.8.1 Game_Init

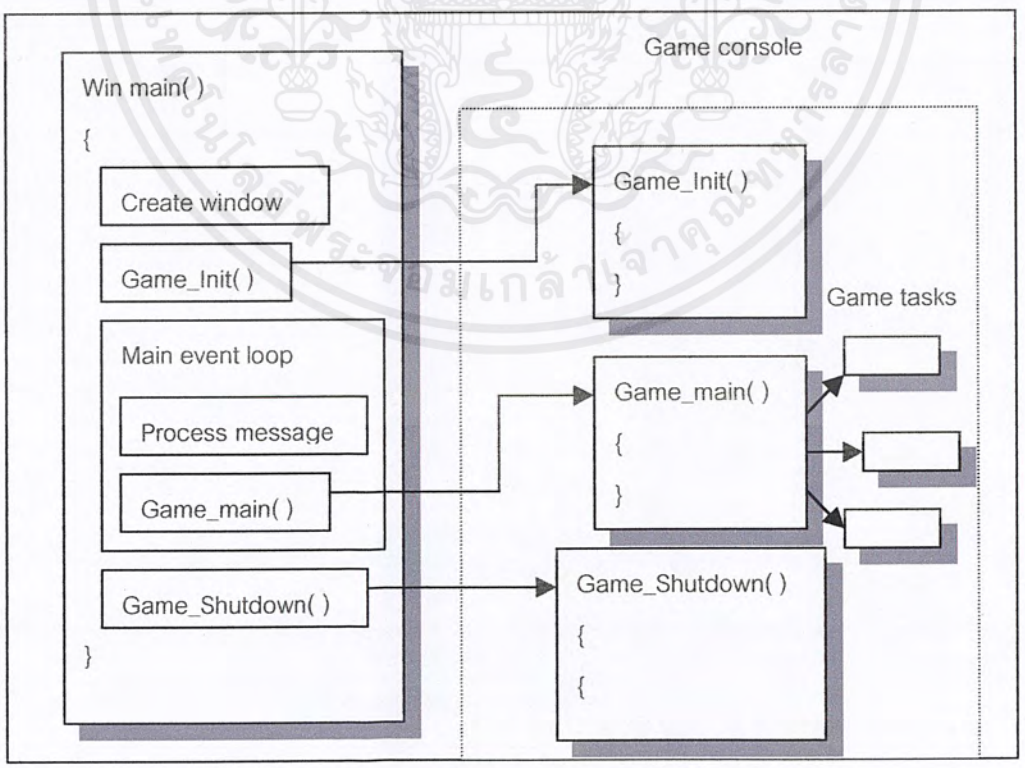
มีหน้าที่กำหนดค่าเริ่มต้นให้แก่ระบบ เช่น ปรับความละเอียดหน้าจอ เตรียม back buffer ,Front buffer ทำการตั้งค่าการ์ดเสียง เป็นต้น โดยฟังชั่นนี้จะถูกเรียกหลังจาก วินโดวส์ ได้ตั้งระบบเริ่มต้นเรียบร้อยแล้ว

2.8.2 Game_Shutdown

มีหน้าที่คืนทรัพยากรต่างๆให้แก่ระบบ เมื่อพร้อมที่จะจบการทำงานของระบบ จะถูกเรียกหลังจากระบบทำงานเสร็จ เตรียมจะปิด

2.8.3 Game_Main

จะคล้ายกับ ฟังก์ชั่น main( ) ในภาษา C ทั่วไป โดยตัวโปรแกรม จะอยู่ในส่วนนี้ และ ฟังก์ชั่น Game_main จะอยู่ใน event loop เพื่อที่จะทำการประมวลผลไปเรื่อยๆ จนกว่าโปรแกรมจะทำงานเสร็จพร้อมที่จะปิด



รูปที่ 2.12 โครงร่างการเขียนเกม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

ขั้นตอนทั้งหมดในงานวิจัยจะเกิดขึ้นไม่ได้หากขาดการออกแบบ การวางแผนที่ดีและมีเป้าหมายชัดเจน การดำเนินงานทั้งหมดจะถูกกระทำภายใต้แผนที่วางเอาไว้อย่างรัดกุมเพื่อความสำเร็จตามเป้าหมายที่วางเอาไว้ในแต่ละช่วง แผนงานทั้งหมดมีดังนี้

1. ขั้นตอนการออกแบบ ซึ่งอาจแบ่งย่อยได้อีกหลายส่วนงาน
   - การออกแบบระบบเกม
   - การออกแบบตัวละครในเกม
   - การออกแบบอินเทอร์เฟซที่ใช้ติดต่อกับผู้เล่น
   - การออกแบบวิธีการตอบโต้ของฝ่ายคอมพิวเตอร์

ในขั้นตอนการออกแบบนี้จะใช้เวลานานพอสมควร เพื่อให้ได้รูปแบบที่เป็นเอกลักษณ์และสอดคล้องกับเป้าหมายซึ่งเน้นรูปแบบที่แปลกใหม่

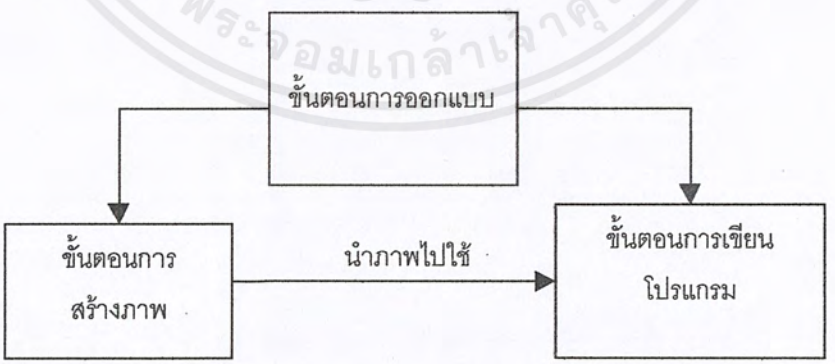
2. ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้ใช้เวลานานที่สุดในตลอดระยะเวลาทำวิจัย เพราะจำเป็นต้องศึกษารูปแบบวิธีการเขียนโปรแกรมที่เป็นลักษณะ Realtime และการเขียนโปรแกรมแบบเชิงวัตถุ เพื่อให้ได้โปรแกรมที่กระทัดรัดและทำงานได้ถูกต้อง เข้าใจง่าย

3. ขั้นตอนการสร้างตัวละคร การสร้างฉาก และการสร้างรูปภาพต่างๆที่ใช้ในเกม

ขั้นตอนนี้ใช้ระยะเวลาสั้นๆ โดยทำการสร้างตัวละคร ฉาก และรูปภาพต่างๆ และเก็บไว้เป็นแหล่งข้อมูล เพื่อใช้ในเกม

เราสามารถสรุปขั้นตอนการวางแผนเป็นแผนภาพได้ดังนี้



รูปที่ 3.1 ขั้นตอนการวางแผน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบมีรายละเอียดดังนี้

3.1.1 การออกแบบระบบเกม

การออกแบบระบบเกมเป็นส่วนที่สำคัญที่สุดในขั้นตอนการออกแบบทั้งหมด เพราะระบบเกมคือทุกสิ่งทุกอย่างตั้งแต่แนวคิดของเกม การดำเนินเรื่องของเกม จนถึงการดำเนินเกม โดยการออกแบบมีรายละเอียดต่างๆดังนี้

- การออกแบบทั่วไป
- การออกแบบคำสั่งต่างๆ และการรับข้อมูล
- การออกแบบการเลือกหน่วยทหารเพื่อรับคำสั่ง

3.1.1.1 การออกแบบทั่วไป

การออกแบบทั่วไปเป็นการออกแบบที่เป็นรูปลักษณ์ของเกม โดยมีสิ่งต่างๆดังนี้

- ตัวละครที่เห็นในเกม เป็นทหารราบทั้งหมด
- เกมมีลักษณะเป็นการวางแผนแบบเรียลไทม์ (Real time) คือสามารถสั่งการและดำเนินการได้เลยโดยไม่ต้องรอการเปลี่ยนตา
- มุมมองที่ใช้เป็นมุมมองจากด้านบนลักษณะคล้ายไอโซเมทริกซ์ (Isometrix)
- ทหารแต่ละหน่วยมีระยะที่เรียกว่าระยะการโจมตี หรือระยะการมองเห็น เมื่อทหารฝ่ายตรงข้ามบุกรุกเข้ามาในเขตนี้ ทหารหน่วยนั้นจะทำการโจมตีทันที

3.1.1.2 การออกแบบคำสั่งต่างๆและการรับข้อมูล

เป็นการออกแบบคำสั่งที่ใช้สำหรับสั่งหน่วยทหารให้ปฏิบัติ และออกแบบรูปแบบการรับข้อมูลการสั่งการทั่วๆไปสามารถใช้เมาส์ปุ่มขวาอย่างเดียวได้โดยทำการเลือกหน่วยทหารที่ต้องการ แล้วสั่งการหน่วยทหารนั้นๆด้วยเมาส์ปุ่มขวา โปรแกรมจะทำการตรวจสอบข้อมูลที่รับเข้ามาแล้วทำการเลือกการทำงานที่เหมาะสมแก่หน่วยทหารนั้นๆ ยกตัวอย่างเช่น การสั่งทหารให้เคลื่อนที่ไปยังจุด ๆ หนึ่ง เราสามารถคลิกเมาส์ปุ่มขวาไปยังจุดนั้นๆได้ ในขณะเดียวกันเมื่อเราต้องการโจมตีทหารฝ่ายตรงข้าม เราสามารถคลิกเมาส์ปุ่มขวาไปยังทหารหน่วยตรงข้ามเพื่อทำการโจมตีได้เลย การทำงานรูปแบบนี้เป็นที่นิยมในเกมวางแผนการรบทั่วๆไป โดยเรียกว่า "สมาร์ทคลิก"(Smart Click)

- ทหารทุกๆหน่วย มีคำสั่งให้ดำเนินการดังนี้
  - เคลื่อนที่ (Move)
  - โจมตี (Attack)
  - รอรับคำสั่ง (Stand by)
  - ประจำที่พร้อมรบ (Hold)
  - การซ่อนตัว (Hide)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยที่แต่ละคำสั่งมีรายละเอียดปลีกย่อยดังต่อไปนี้

การเคลื่อนที่ (Move)

เป็นคำสั่งที่ให้ทหารที่เลือกไว้ เคลื่อนที่ไปยังจุดที่กำหนด โดยไม่สนใจว่าโดนโจมตีอยู่หรือไม่สามารถสั่งการด้วยเมาส์อย่างเดียว ทำการคลิกเมาส์ปุ่มซ้ายไปยังจุดที่ต้องการ เพื่อสั่งให้ทหารเดินทางไปยังจุดที่กำหนด

การโจมตี (Attack)

เป็นคำสั่งที่ใช้สำหรับโจมตีทหารฝ่ายตรงข้ามโดยจะทำการติดตามโจมตีจนกว่าทหารฝ่ายตรงข้ามจะตาย สามารถสั่งการได้โดยคลิกเมาส์ปุ่มซ้ายที่ทหารฝ่ายตรงข้ามเพื่อทำการโจมตี ในระหว่างทางหากว่าเจอทหารฝ่ายตรงกันข้าม จะทำการโจมตีและติดตามจนกว่าทหารฝ่ายตรงข้ามจะตาย แล้วจะเดินทางต่อไปยังจุดหมายที่กำหนดไว้

รอรับคำสั่ง (Stand by)

เป็นคำสั่งที่สั่งให้ทหารหยุดรอรับคำสั่งต่อไป โดยหากมีทหารฝ่ายตรงข้ามเข้ามาในระยะที่สามารถโจมตีถึง ทหารจะทำการโจมตี และเดินตามจนกว่าจะทำลายทหารฝ่ายตรงข้ามหน่วยนั้นได้ และเมื่อคำสั่งเคลื่อนที่หรือคำสั่งโจมตีถูกกระทำจนเสร็จทหารจะเปลี่ยนสถานะเป็นสถานะนี้

ประจำตำแหน่งและป้องกันตัว (Hold)

เป็นคำสั่งที่ให้ทหารหยุดนิ่งกับที่ โดยจะทำการโจมตีต่อเมื่อทหารฝ่ายตรงข้ามเข้ามาในเขตระยะยิงโดยที่หากทหารฝ่ายตรงข้ามเดินหนี ทหารฝ่ายเราจะไม่ติดตาม

การซ่อนตัว (Hide)

เป็นคำสั่งที่ไม่ให้ทหารโจมตีถึงแม้ฝ่ายตรงข้ามจะอยู่ในระยะที่โจมตีถึง เพื่อประโยชน์ทางกลยุทธ์ แต่จะทำการโจมตีกลับถ้าหากโดนโจมตีก่อน

3.1.1.3 **การออกแบบการเลือกหน่วยทหารเพื่อรับคำสั่ง**

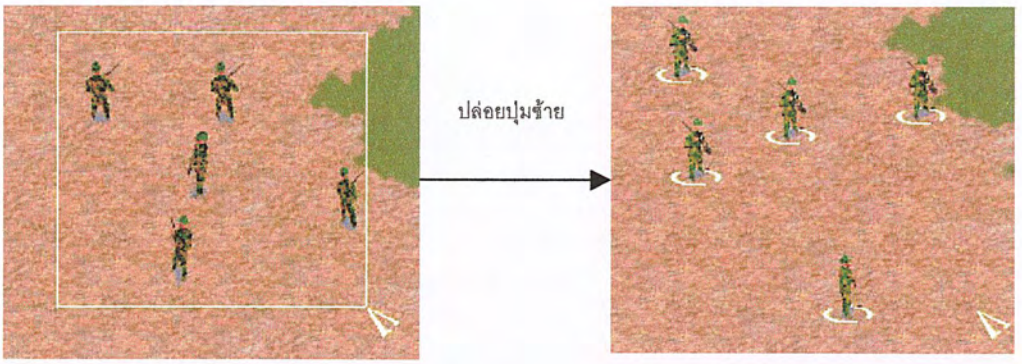
เป็นการออกแบบระบบการเลือกหน่วยทหาร โดยทั่วไปการเลือกทหารเพื่อรับคำสั่งนั้นจะทำการเลือกสั่งการทหารทีละหน่วย แต่ในเกมนี้จะสามารถเลือกได้ครั้งละหลายๆหน่วยทหาร เพื่อสั่งการให้ทำการเป็นกลุ่ม มีรายละเอียดดังนี้

- ทหารจะสามารถเลือกให้สั่งการได้โดยคลิกเมาส์ปุ่มซ้ายที่ตัวทหารที่ต้องการ

รูปที่ 3.2 การเลือกยูนิตทหาร

- สามารถเลือกทหารได้พร้อมกันทีละหลายยูนิต โดยการคลิกเมาส์ซ้ายค้างไว้แล้วลากให้เป็นกรอบสี่เหลี่ยมครอบทหารยูนิตที่ต้องการเลือก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.3 การเลือกยูนิตทหารเป็นกลุ่ม

- ทหารสามารถเลือกได้สูงสุด 12 ตัวในการลากเมาส์ล้อมกรอบ 1 ครั้ง
- สามารถจัดทัพให้ทหารได้ โดยมีได้ทัพละไม่เกิน 12 ตัว โดยกำหนดเป็นตัวเลข 0 ถึง 9 โดยทำการเลือกทหารจำนวนที่ต้องการในหนึ่งทัพ จากนั้น กดปุ่ม Ctrl ตามด้วยปุ่มตัวเลขที่ต้องการ

3.1.2 การออกแบบตัวละคร

การออกแบบตัวละครเป็นขั้นตอนที่ใช้เวลาสั้นที่สุด เนื่องจากตัวละครในเกมมีแต่ทหารราบ ดังนั้นจึงเน้นแต่การออกแบบท่าทางแต่ละท่ามากกว่า โดยที่โปรแกรมที่ใช้ในการออกแบบตัวละครนี้คือโปรแกรม Metacreation Poser เวอร์ชั่น 3.0 ท่าทางแต่ละท่าจะจะมีมุมมองหลายมุม หรือหลายทิศทาง โดยท่าทางทั้งหมดที่มีในเกมมีดังนี้

- ท่าวิ่งหรือเคลื่อนที่
- ท่ายิง
- ท่าตาย

โดยแต่ละท่าจะมีทิศทางต่างๆกันดังนี้

- ทิศตะวันออก (วิ่งไปทางขวา)
- ทิศตะวันตก (วิ่งไปทางซ้าย)
- ทิศเหนือ (วิ่งขึ้นข้างบน)
- ทิศใต้ (วิ่งลงข้างล่าง)
- ทิศตะวันออกเฉียงเหนือ (วิ่งไปทางขวาเฉียงด้านบน)
- ทิศตะวันออกเฉียงใต้ (วิ่งไปทางขวาเฉียงลงด้านล่าง)
- ทิศตะวันตกเฉียงเหนือ (วิ่งไปทางซ้ายเฉียงขึ้นด้านบน)
- ทิศตะวันตกเฉียงใต้ (วิ่งไปทางซ้ายเฉียงลงด้านล่าง)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

โดยมีรายละเอียดของแต่ละท่าทางดังนี้

3.1.2.1 ท่าวิ่งหรือเคลื่อนที่

ท่าวิ่งมีทั้งหมด 8 ทิศทาง แต่ละทิศทางจะมีทั้งหมด 16 เฟรม หรือ 16 การเคลื่อนไหว

รูปที่ 3.4 แสดงท่าวิ่งใน1มุมมอง

จากรูปเป็นการเคลื่อนไหว 1 ทิศทาง คือทิศตะวันตกเฉียงใต้ (วิ่งไปทางซ้ายเฉียงลงด้านล่าง) แต่จำเป็นที่จะต้องมีอีกหลายทิศทาง ดังรูป

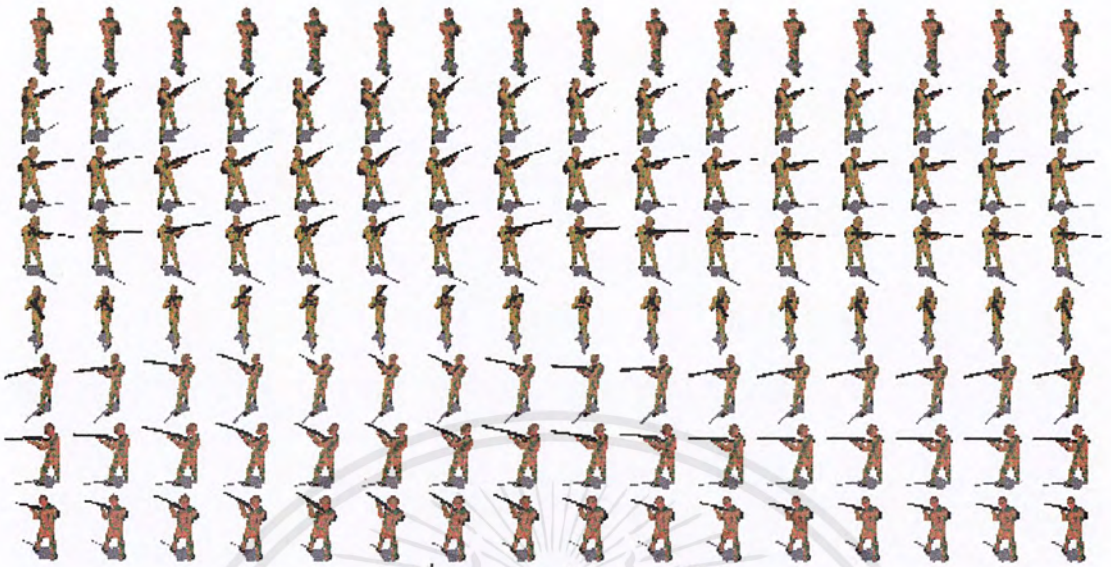
รูปที่ 3.5 แสดงทำวิ่งในหลายมุมมอง

การสร้างภาพแบบนี้เราเรียกว่าภาพสไปรท์ (Sprite) จากภาพเป็นท่าวิ่งอย่างเดียว แต่ถ้ามีท่าทางอื่นด้วยจะทำการสร้างและนำมาเพิ่มเข้าไปในท่าวิ่ง โดยการเขียนโปรแกรมจะทำการเลือกภาพมาแสดงโดยใช้จุดคู่ลำดับ X,Y ด้านบนของภาพ และ X,Y ด้านล่างของภาพ ลักษณะนี้เมื่อนำภาพมาเขียนโปรแกรมจะเป็นเทคนิคที่ใช้ในการทำภาพเคลื่อนไหว (Animation)

รูปทางซ้ายกล่าวถึงจุดคู่ลำดับ X,Y ด้านบน และจุดคู่ลำดับ X,Y ด้านล่าง โดยจะมีโครงสร้างข้อมูลชนิดหนึ่งเพื่อการนี้โดยเฉพาะ รายละเอียดจะกล่าวถึงในหัวข้อการทำภาพเคลื่อนไหว (Animation)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

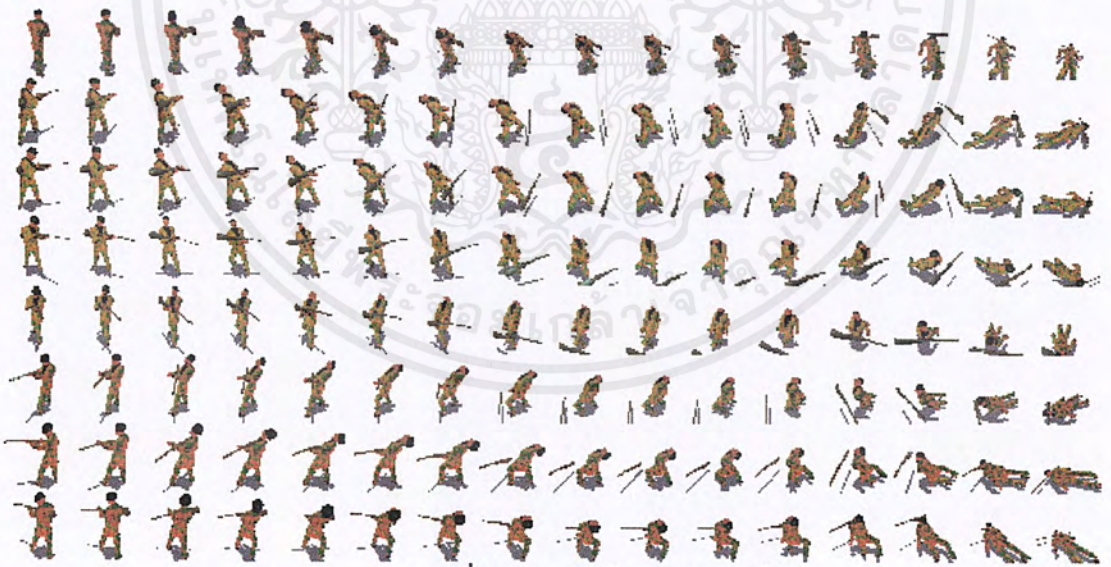
ท่ายิงมีจำนวนท่าเท่ากับท่าวิ่ง และนี่คือรูปบางส่วน



รูปที่ 3.6 แสดงท่ายิงของยูนิต

3.1.2.3ท่าตาย

ท่าตายมีลักษณะดังรูป

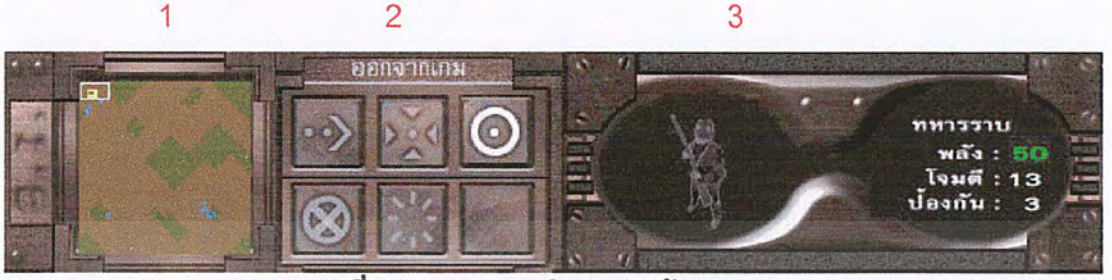


รูปที่ 3.7 แสดงท่าตายของ ยูนิต

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.1.3 การออกแบบอินเทอร์เฟซติดต่อกับผู้ใช้

ในส่วนการออกแบบอินเทอร์เฟซที่ใช้ติดต่อกับผู้ใช้ จะแสดงถึงสถานะต่างๆของเกมขณะนั้น เช่น แผนที่ขนาดเล็ก หรือปุ่มคำสั่งต่างๆ และสถานะของทหารที่เลือกขณะนั้น โดยแสดงถึงพลังของทหารคนนั้น อินเทอร์เฟซต่างๆมีรูปแบบลักษณะดังรูป



รูปที่ 3.8 แสดงส่วนติดต่อกับผู้เล่น

จากรูปแสดงถึงส่วนสำคัญสามส่วนดังนี้

1. แผนที่เล็ก
2. กลุ่มของปุ่มคำสั่ง
3. ส่วนแสดงสถานะทหาร

3.1.3.1 แผนที่เล็ก

แผนที่เล็กเป็นภาพโดยรวมของแผนที่ทั้งหมด โดยแสดงหน่วยทหารแต่ละฝ่ายแยกตามสี ยกตัวอย่าง เช่น ทหารฝ่ายเราอาจแทนด้วยจุดสีเขียว และทหารฝ่ายตรงข้ามอาจแทนด้วยจุดสีขาว (ตามรูป) เป็นต้น ในแผนที่เล็กจะมีกรอบสี่เหลี่ยมเล็กซึ่งใช้อ้างอิงให้เห็นว่าขณะนั้นเรากำลังสังเกตการณ์อยู่ที่ส่วนใดในแผนที่ทั้งหมด ในแผนที่เล็กเราสามารถคลิกเมาส์ปุ่มซ้ายไปเพื่อสังเกตการณ์ในส่วนใดก็ได้ในแผนที่ และเรายังสามารถสั่งทหารที่เลือกไว้ให้เดินทางไปยังจุดใดๆในแผนที่ได้ โดยทำการเลือกทหารแล้วคลิกเมาส์ปุ่มขวาไปยังแผนที่เล็กในจุดที่ต้องการ โปรแกรมจะทำการตรวจสอบอ้างอิงกับแผนที่จริงเพื่อให้ทหารเดินไปยังจุดที่ถูกต้อง

3.1.3.2 ส่วนแสดงสถานะของยูนิต

ส่วนนี้เป็นส่วนที่ใช้บ่งบอกถึงสภาวะทั้งหมดของยูนิตที่เลือกไว้ โดยทำการแสดงถึงพลังของยูนิต และสถานะของยูนิตว่ากำลังทำอะไรอยู่ ส่วนนี้จะทำการแสดงขึ้นมาเมื่อเลือกทหารเพียงหน่วยเดียว ถ้าเป็นการเลือกทหารหลายหน่วย จะไม่สามารถแสดงสถานะของทหารทั้งหมดได้ หรือเลือกสิ่งก่อสร้างเช่น โรงทหาร โรงสูบน้ำมัน

3.1.3.3 กลุ่มของปุ่มคำสั่ง

ในส่วนนี้เป็นส่วนที่ใช้สำหรับปุ่มคำสั่ง ซึ่งเป็นคำสั่งของทหารหน่วยที่ทำการเลือกอยู่ หรือสิ่งก่อสร้างที่เลือกอยู่ เมื่อไม่ได้อยู่ในสถานะที่เลือกทหารหรือสิ่งก่อสร้าง ส่วนนี้จะไม่แสดงปุ่มคำสั่ง ปุ่มคำสั่งที่มีเป็นคำสั่งทุกอย่างที่มี ดังนี้

- เคลื่อนที่ (Move)
- การโจมตี (Attack)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- รอรับคำสั่ง (Stand by)
- ประจำตำแหน่งและป้องกันตัว (Hold)
- การซ่อนตัว (Hide)
- ผลิตทหาร (สำหรับโรงทหาร)

3.1.4 การออกแบบการตอบโต้ของคอมพิวเตอร์

ส่วนของการออกแบบในส่วนนี้จำเป็นต้องใช้เวลาในการออกแบบนานที่สุดในขั้นตอนการออกแบบ เนื่องจากการตอบโต้ของคอมพิวเตอร์ต่อการกระทำของผู้เล่นเป็นเรื่องที่ละเอียดอ่อน และต้องใช้การระดมความคิด ในขั้นตอนการออกแบบจำเป็นจะต้องจำลองตัวเองให้เป็นผู้เล่นฝ่ายตรงข้าม เพื่อที่จะสามารถวิเคราะห์สถานการณ์และแก้ปัญหาได้ตรงจุด

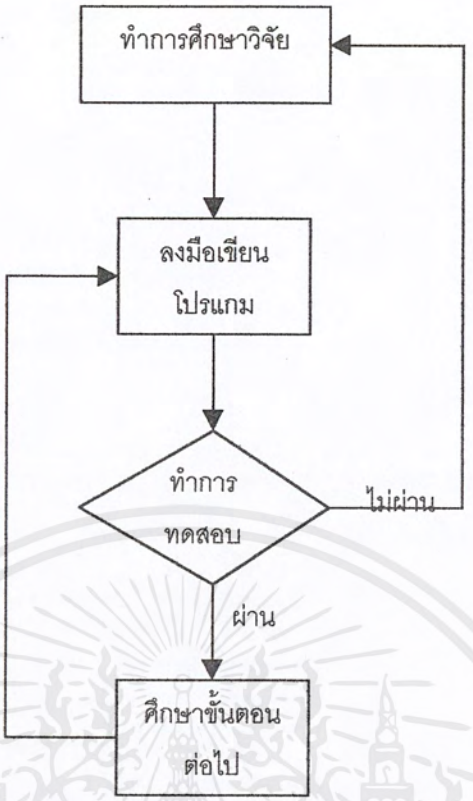
การออกแบบได้มีรูปแบบที่ชัดเจนบ้างแล้วในบางจุดในการแก้ไขปัญหาให้ฝ่ายคอมพิวเตอร์ โดยมีลักษณะดังนี้

- คอมพิวเตอร์จะทำการรวบรวมหน่วยทหารให้ได้จำนวนหนึ่ง(ประมาณหนึ่งทัพ หรือ 12 ตัว) เพื่อทำการเข้าโจมตีกองทหารฝ่ายผู้เล่น
- ทหารฝ่ายคอมพิวเตอร์มีคุณสมบัติทุกอย่างเหมือนทหารฝ่ายผู้เล่น
- ยูนิตของทั้งฝ่ายคอมพิวเตอร์ และฝ่ายผู้เล่นมีลักษณะเหมือนกัน
- หากทหารฝ่ายคอมพิวเตอร์โดนโจมตี คอมพิวเตอร์จะสั่งการให้ทหารที่อยู่ใกล้เคียงเข้ามาช่วยร่วมรบ
- คอมพิวเตอร์จะพยายามทำลายฐานฝ่ายผู้เล่นเพื่อชนะ
- ฝ่ายคอมพิวเตอร์จะพยายามป้องกันฝ่ายผู้เล่นมาทำลายฐาน
- ทหารฝ่ายคอมพิวเตอร์จะโจมตีฝ่ายผู้เล่นเมื่อเข้าระยะยิง
- ทหารฝ่ายคอมพิวเตอร์จะติดตามฝ่ายผู้เล่นเมื่อฝ่ายผู้เล่นทำการหนี จะติดตามจนกว่าจะทำลายฝ่ายผู้เล่นได้

3.2 ขั้นตอนการเขียนโปรแกรม

ส่วนใหญ่แล้วขั้นตอนต่างๆจะมีการทำงานเป็นเส้นตรง หมายถึงเมื่อทำเสร็จงานหนึ่งก็จะทำงานต่อไปโดยไม่มีการวนมาที่งานนั้นใหม่ แต่ขั้นตอนการเขียนโปรแกรมมีการทำงานที่ต้องวนมาศึกษาเรื่อยๆ และทำการเขียนโปรแกรม และหากยังเขียนไม่สำเร็จ ก็จะต้องทำการศึกษาเพิ่มเติมใหม่ ดังแผนภาพต่อไปนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.9 ขั้นตอนการเขียนโปรแกรม

โปรแกรมเกมจะถูกแบ่งเป็นส่วนย่อย ๆ หลายส่วนดังนี้

- ส่วนของการแสดงผล จะอยู่ในไฟล์ display.cpp และ display.h
- ส่วนของการติดต่อกับอุปกรณ์รับข้อมูลจะอยู่ในไฟล์ input.cpp และ input.h
- ส่วนของการติดต่อการ์ดเสียงและการสร้างเสียง จะอยู่ในไฟล์ sound.cpp และ sound.h
- ส่วนของ ยูนิต หรือตัวทหาร จะอยู่ในไฟล์ GI_Unit.cpp และ GI.h
- ส่วนของแผนที่ จะอยู่ในไฟล์ Map.cpp และ map.h

เหตุผลหลักของผู้ทำโครงงานแบ่งการทำงานออกเป็นหลายๆส่วน เพื่อ ต้องการจัดแบ่งหน้าที่การทำงานให้เป็นประเภทเดียวกัน ทำให้ง่ายต่อการตรวจสอบ แก้ไข หรือเพิ่มเติม

หลักการ แนวความคิดและรูปแบบของข้อมูลในแต่ละส่วนการทำงาน

3.2.1 ส่วนการแสดงผล

ในส่วนการแสดงผลนี้จะรวมฟังก์ชั่นการทำงานที่เกี่ยวกับการแสดงผล การติดต่อกับ Hardware โดยผ่านทาง DirectDraw ทำการสร้าง Primary surface ,Secondary Surface ,OffScreen Surface การทำภาพเคลื่อนไหว การแสดงเคอร์เซอร์ของเมาส์ และ การแสดงรูปยูนิต

ฟังก์ชั่นการทำงานในส่วนของการแสดงผล มีดังนี้

- SetupDirectDraw
- ClearDirectDraw

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- SetOffScreen2
- DDColorMatch
- InitSprite
-GISetColorKey
-ShowCur
-ShowUnit
-ShowMessage
-DrawRect
-DDReLoadBitmap
-DDCopyBitmap

หลักการและการทำงานของฟังก์ชั่น

3.2.1.1 SetupDirectDraw

ฟังก์ชั่นนี้จะทำการติดต่อกับ DirectDraw เพื่อกำหนดค่าต่างๆ สำหรับการแสดงผล ทำการสร้าง Primary Surface และ Secondary Surface ทำการกำหนดลักษณะของแต่ละ Surface ว่ามีข้อมูลอย่างไร ขนาด ความยาว ความกว้าง ของ surface มีค่าเท่าไร ทำการกำหนดค่าความละเอียดของจอที่ใช้แสดงผล สำหรับโปรแกรมเกมที่ได้พัฒนานี้ ใช้ความละเอียดในการแสดงผล 640 x 480 พิกเซล

3.2.1.2 ClearDirectDraw

เมื่อต้องการยกเลิกการติดต่อกับ DirectDraw จะต้องทำการลบออปเจคของ DirectDraw ทิ้งเพื่อคืนหน่วยความจำกลับสู่ระบบ

3.2.1.3 SetOffScreen2

ทำการสร้าง OffScreen Surface เพื่อนำภาพจากไฟล์มาเก็บไว้ในหน่วยความจำโดยเรียกฟังก์ชั่น DDCopyBitmap อีกทีหนึ่ง เหตุผลที่ต้องสร้าง OffScreen Surface เพราะการอ่านข้อมูลจากไฟล์ จะช้ากว่าการอ่านข้อมูลจากหน่วยความจำ ถ้าหากมีการอ่านข้อมูลจากไฟล์บ่อยครั้ง จะเป็นการหน่วงทำให้แสดงผลได้ช้า ไฟล์ข้อมูลที่ทำการอ่านขึ้นมานี้จะใช้ไฟล์ข้อมูลประเภท BMP เพราะฉะนั้นความกว้าง ความยาวของ OffScreen Surface ส่วนใหญ่จะมีขนาดเท่ากับ ความกว้าง ความยาว ของภาพ BMP นั้นๆ

3.2.1.4 InitSprite

ทำการโหลดไฟล์ภาพที่ต้องใช้ในโปรแกรม นำไปไว้ใน OffScreen Surface และกำหนดค่า Color key ของแต่ละ OffScreen Surface

3.2.1.5 GISetColorKey

ทำการกำหนดค่า Color key ให้แต่ละ Surface ค่า Color key นี้จะเป็นค่าสีในระบบ RGB เวลาแสดงผล สีที่ถูกกำหนดให้เป็น Color key จะไม่ถูกแสดงออกมายังจอภาพ ในโปรแกรมเกมที่ได้พัฒนาได้กำหนดให้ สีดำ (RGB = 0,0,0) เป็นค่า color key

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.1.6 ShowCur

ฟังก์ชั่นนี้จะทำการแสดงผลเคอร์เซอร์บนจอแสดงภาพ เคอร์เซอร์ที่แสดงจะมีอยู่ 3 แบบ

- เคอร์เซอร์ปกติ
- เคอร์เซอร์แสดงตำแหน่งเมื่อยูนิตได้รับคำสั่งเดิน
- เคอร์เซอร์แสดงตำแหน่งเมื่อยูนิตได้รับคำสั่งโจมตี

3.2.1.7 ShowUnit

ฟังก์ชั่นนี้จะทำการแสดงยูนิตบนจอแสดงภาพ ตามประเภทของยูนิต โดยยูนิตของฝ่ายผู้เล่นจะอยู่ในชุดทหารโทนสีเขียว ส่วนยูนิตฝ่ายคอมพิวเตอร์จะอยู่ในชุดทหารโทนสีเหลือง

รูปที่ 3.10 ทหารฝ่ายผู้เล่นและฝ่ายตรงข้าม

3.2.1.8 DrawRect

ฟังก์ชั่นนี้จะทำการวาดสี่เหลี่ยมมีความกว้าง ความยาว และสีตามค่าพารามิเตอร์ที่ส่งเข้ามา

3.2.1.9 ShowMessage

ฟังก์ชั่นนี้จะทำการแสดงข้อความ ขึ้นบนจอแสดงภาพ ตามตำแหน่งที่ได้รับเข้ามา

3.2.1.10 DDCopyBitmap

ฟังก์ชั่นนี้จะทำการคัดลอกข้อมูลในไฟล์ภาพ BMP ไปไว้ใน Surface

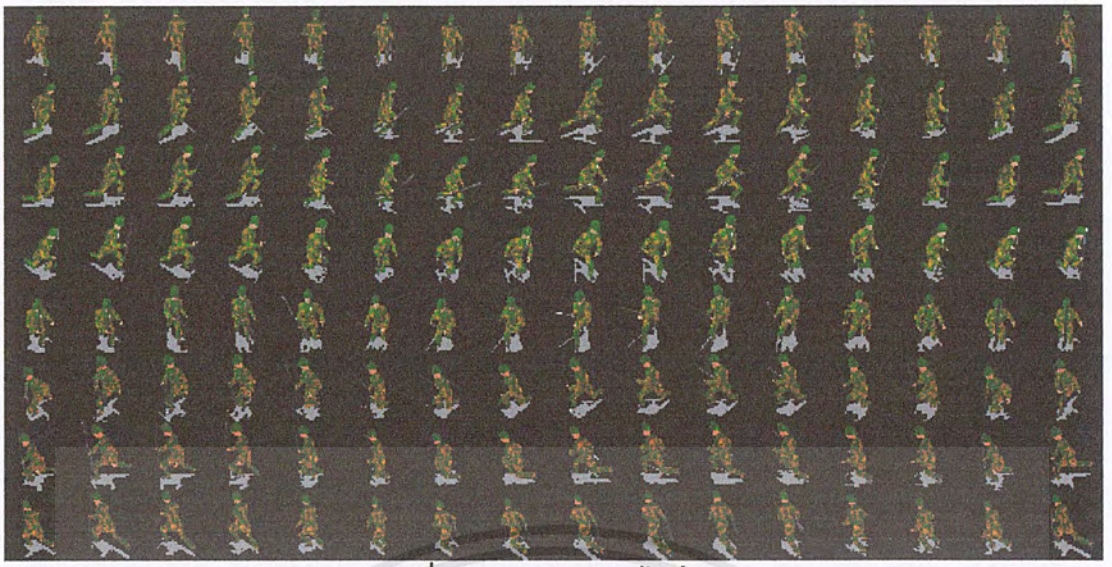
3.2.2 การทำภาพเคลื่อนไหว

ในการสร้างภาพเคลื่อนไหวทั่วไปอาศัยหลักการเหมือนฟิล์มภาพยนตร์คือการนำภาพนิ่งซึ่งเป็นภาพนิ่งที่มีการเคลื่อนไหวติดต่อกัน นำมาแสดงด้วยอัตราเร็วระดับหนึ่ง ทำให้เกิดลวงตาเป็นภาพเคลื่อนไหวขึ้นมาได้

ในส่วนนี้จะอธิบายถึงหลักการที่ใช้ในการแสดงภาพนิ่งติดต่อกันเพื่อให้เกิดเป็นภาพเคลื่อนไหว และในตอนท้ายจะแสดงถึงแผนภาพเพื่อให้เห็นภาพรวมของหลักการ

ตามที่ได้กล่าวมาข้างต้นแล้วว่า ภาพเคลื่อนไหวเกิดจากภาพนิ่งจำนวนหนึ่งซึ่งนำมาแสดงต่อกันด้วยอัตราเร็วระดับหนึ่ง ดังนั้นเราจะต้องมีวิธีนำภาพนิ่งมาแสดง วิธีการทำภาพเคลื่อนไหวที่เป็นที่แพร่หลายที่สุดคือการนำภาพนิ่งหลายๆภาพมาต่อกันซึ่งแต่ละภาพเป็นท่าทางแต่ละท่า ซึ่งแต่ละท่าจะมีระยะของพิกเซลเท่าๆกันเพื่อประโยชน์ในการคำนวณ ภาพลักษณะนี้เรียกว่าภาพ "สไปร์ท" (Sprite) และแต่ละภาพนิ่งที่ถูกแสดงเรียกว่าการแสดงภาพเคลื่อนไหวหนึ่งเฟรม

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



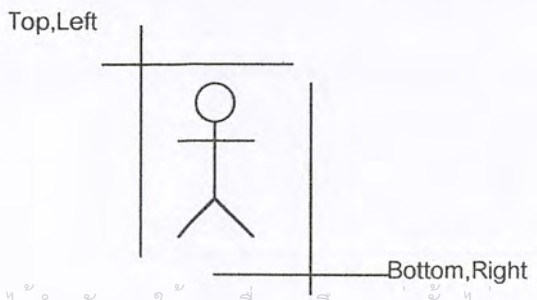
รูปที่ 3.11 แสดงภาพสไปร์ท

จากรูปคือการนำภาพนิ่งของทหารที่กำลังทำการวิ่งมาต่อกัน โดยแต่ละท่ามีระยะห่างกัน 40 พิกเซล ในการสร้างภาพเคลื่อนไหวเราจะทำการแสดงภาพนิ่งจากภาพทางซ้ายสุดไปยังภาพทางขวาสุด แล้วจะวนกลับมาแสดงภาพแรกทางซ้ายสุดแล้ววนไปทางขวาสุด เป็นเช่นนี้เรื่อยไป เราจะได้ภาพที่เป็นทหารกำลังวิ่ง หากเราต้องการให้ทหารวิ่งไปจริงๆ เราเพียงแต่ทำการเปลี่ยนตำแหน่งที่ใช้แสดงผลบนจอภาพไปยังทิศทางที่ต้องการในการแสดงภาพนิ่งจากซ้ายสุด ไปขวาสุดเราเรียกว่าทำการแสดงภาพเคลื่อนไหวครบหนึ่งรอบสไปร์ท โดยจะเห็นได้ว่าภาพทางซ้ายสุด กับภาพทางขวาสุดเป็นภาพต่อกัน หรือแทบจะเป็นภาพเดียวกัน

วิธีที่ใช้ในการอ้างอิงภาพจากภาพนิ่งคือการใช้โครงสร้างข้อมูลชนิดหนึ่ง โดยลักษณะของการอ้างอิงจะอ้างอิงโดยการใช้กรอบสี่เหลี่ยม ภาพใดก็ตามที่อยู่ในขอบเขตของกรอบสี่เหลี่ยมนี้ จะถูกนำมาแสดงขึ้นบนจอภาพ โครงสร้างข้อมูลดังกล่าวคือโครงสร้างข้อมูลชนิด RECT หรือ Rectangle นั่นเอง ประกอบด้วยข้อมูลดังนี้

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

จากโครงสร้างข้อมูลดังกล่าว สามารถนำมาแสดงเป็นรูปได้ดังนี้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวอย่าง การเขียนโปรแกรมแสดงภาพ

```
        RECT  source_rectangle, dest_rectangle;
// กำหนดขอบเขตของสี่เหลี่ยมที่ครอบรูปภาพที่จะนำมาแสดง
        source_rectangle.top = 0;
        source_rectangle.left = 0;
        source_rectangle.right = 39;
        source_rectangle.bottom=39;
// กำหนดขอบเขตของสี่เหลี่ยมที่เป็นตำแหน่งที่จะวางบนจอภาพ
        dest_rectangle.top = 50;
        dest_rectangle.left = 50;
        dest_rectangle.right = 80;
        dest_rectangle.bottom = 80;
// ทำการวาดภาพลงหน้าจอ
        draw_pic(&source_rectangle,source_surface,&dest_rectangle,dest_surface);
// source_surface คือ surface ของภาพที่ต้องการแสดง  และ dest_surface คือ surface ที่จะนำภาพไป
// แสดง
```

จากตัวอย่างเป็นการเขียนโปรแกรมเพื่อแสดงภาพหนึ่งภาพ หรือภาพนิ่งเท่านั้น แต่ในการแสดงภาพเคลื่อนไหว จำเป็นจะต้องมีการคำนวณเข้ามาเกี่ยวข้อง จากภาพทหารวิ่ง ได้กล่าวมาแล้วว่าแต่ละภาพห่างกัน 40 พิกเซล ซึ่งหากต้องการแสดงภาพเคลื่อนไหว เราเพียงแต่ทำการบวกค่าของ RECT เพิ่มเข้าไป โดยเริ่มต้นหากกำหนดให้ Source_Rect เป็นกรอบสี่เหลี่ยมสำหรับภาพที่จะนำไปแสดง เราอาจกำหนดได้ดังนี้

```
// ตรวจสอบ Source_Rect.right ว่าถึงขอบเขตขวาสุดของสไปร์ทแล้ว
while (Source_Rect.right != END_RIGHT)
{
        Source_Rect.left+=40;       // ทำการเพิ่มค่า Source_Rect.left ไป 40 พิกเซลเพื่อไปยังภาพนิ่งถัดไป
        Source_Rect.right=Source_Rect.left+40;
        Source_Rect.top = TOP_SPRITE;
        Source_Rect.bottom=BOTTOM_SPRITE;
        draw_pic(&source_rectangle,source_surface,&dest_rectangle,dest_surface);
}
```

จากโปรแกรมที่ยกตัวอย่าง จะเป็นการแสดงภาพเคลื่อนไหวโดยภาพจะเคลื่อนไหวขยับอยู่กับที่ แต่หากเราต้องการให้เคลื่อนไหวไปข้างหน้า เราเพียงแต่ทำการเพิ่มค่า destination_rect เข้าไป

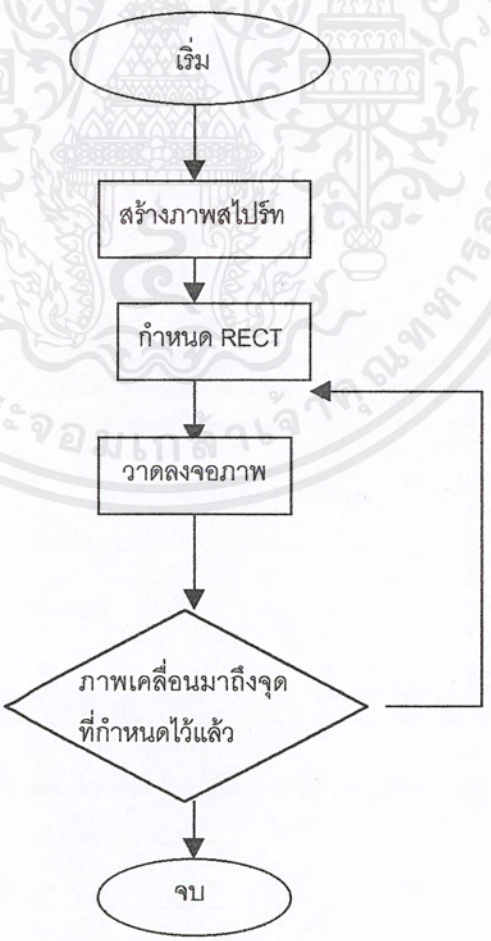
```
while (Source_Rect.right != END_RIGHT)
{
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Source_Rect.left+=40;       // ทำการเพิ่มค่า Source_Rect.left ไป 40 พิกเซลเพื่อไปยังภาพนิ่งถัดไป
Source_Rect.right+=Source_Rect.left;
Source_Rect.top = TOP_SPRITE;
Source_Rect.bottom=BOTTOM_SPRITE;

destination_rect.left += 4;       // ทำการเพิ่มค่า destination_rect.left เข้าไปครั้งละ 4 พิกเซลเพื่อ
                                  // ให้การเคลื่อนไหวดูนุ่มนวล
destination_rect.right +=destination_rect.left;
destination_rect.top = TOP_DEST;
destination_rect.bottom = BOTTOM_DEST;
draw_pic(&source_rectangle,source_surface,&dest_rectangle,dest_surface);
}
```

ในการสร้างภาพเคลื่อนไหวนั้น อาศัยการคำนวณเพียงเล็กน้อย แต่อาจสร้างความสับสนให้บ้าง แต่หากได้ลองศึกษาและทดลองจนเข้าใจแล้ว จะสามารถสร้างภาพเคลื่อนไหวได้ด้วยการโปรแกรมเพียงเล็กน้อย โดยขั้นตอนต่างๆในการสร้างภาพเคลื่อนไหวสามารถเสนอเป็นแผนภาพได้ดังนี้



รูปที่ 3.12 ขั้นตอนการทำภาพเคลื่อนไหว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3.2.3 ส่วนการติดต่ออุปกรณ์เพื่อรับอินพุท

ในส่วนนี้จะรวมฟังก์ชั่นที่ทำการติดต่อกับอุปกรณ์อินพุทโดยผ่านทาง Direct Input โปรแกรมเกมที่พัฒนาขึ้นมานี้ใช้ทั้ง เมาส์และคีย์บอร์ดในการรับอินพุท

ฟังก์ชั่นการทำงานในส่วนการติดต่ออุปกรณ์เพื่อรับอินพุต

- SetupDirectInput
- SetKeyboard
- SetMouse
- ClearDirectInput
- GetInput

หลักการและการทำงานของฟังก์ชั่น

3.2.3.1 SetupDirectInput

ฟังก์ชั่นนี้จะทำการติดต่อกับ Direct Input เพื่อเตรียมอุปกรณ์รับอินพุตให้พร้อมใช้งาน

3.2.3.2 SetKeyboard

ฟังก์ชั่นนี้จะทำการติดต่อกับคีย์บอร์ดโดยผ่าน DirectInput ทำการกำหนดลักษณะการทำงานของคีย์บอร์ดกับโปรแกรมเกมที่ได้พัฒนาขึ้น กำหนดรูปแบบของข้อมูลที่จะอ่านจากคีย์บอร์ด

3.2.3.3 SetMouse

ฟังก์ชั่นนี้จะทำการติดต่อกับเมาส์โดยผ่านทาง Direct Input ทำการกำหนดลักษณะการทำงานของเมาส์กับโปรแกรมเกมที่ได้พัฒนาขึ้น กำหนดรูปแบบของข้อมูลที่จะอ่านจากเมาส์

3.2.3.4 ClearDirectInput

เมื่อต้องการยกเลิกการติดต่อกับ DirectInput จะต้องทำการลบออปเจคของ DirectInput ทิ้งเพื่อคืนค่าหน่วยความจำกลับสู่ระบบ

3.2.3.5 GetInput

ฟังก์ชั่นจะทำการรับอินพุทจากเมาส์และคีย์บอร์ดเมื่อได้ข้อมูลอินพุท จะทำการตรวจสอบข้อมูลที่ได้รับ แล้วทำงานตามข้อมูลอินพุทที่ได้รับมา

3.2.4 ส่วนของแผนที่

ในส่วนของแผนที่นี้จะประกอบด้วย คลาส Gl_Map ภายในคลาสจะมีเมธทอด การแสดงแผนที่ การสร้างแผนที่ขนาดเล็ก การสร้างแผนที่จำลอง โครงสร้างของคลาส Gl_Map มีดังนี้

```
class Gl_Map
{
        public :
                int             x;
                int             y;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
    int                 cellx;
    int                 celly;
    int                 delx;
    int                 dely;
    MapArr          array_map;

    LPDIRECTDRAW                    pdd;
    LPDIRECTDRAWSURFACE             lpddstilemap;
    LPDIRECTDRAWSURFACE             lpddstempmap;

    LPDIRECTDRAWSURFACE             lpddsminitile;
    LPDIRECTDRAWSURFACE             lpddsminimap;

    GI_Map(LPDIRECTDRAW lpdd);
    ~GI_Map();

    void InitMap(void);
    void ShowMiniMap(int miniX, int miniY);
    LPDIRECTDRAWSURFACE     CreateMapSurface(int width, int height);
    LPDIRECTDRAWSURFACE     LoadTempMap(void);
    LPDIRECTDRAWSURFACE     Show(void);
    LPDIRECTDRAWSURFACE     InitMiniMap(void);
};
```

การทำงานของ เมธทอดในคลาส GI_Map

3.2.4.1 CreateMapSurface

ฟังก์ชั่นนี้จะทำการสร้าง Off Screen ไว้เป็นที่เก็บข้อมูลของแผนที่ชั่วคราว เพื่อช่วยให้การแสดงภาพแผนที่บนจอแสดงผลเวลาที่มีการเลื่อนแผนที่ ทำได้รวดเร็วขึ้น

3.2.4.2 LoadTempMap

ทำการตรวจสอบข้อมูลที่อยู่ในอาเรย์แผนที่ เปรียบเทียบข้อมูลนั้นว่าควรเป็นแผนที่รูปใด เมื่อตรวจสอบข้อมูลเรียบร้อย จึงนำรูปแผนที่นั้นเก็บไว้ใน Off Screen Surface ที่ได้ทำการสร้างไว้แล้วจากเมธทอด CreateMapSurface

3.2.4.3 InitMiniMap

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำการอ่านค่าจากอาร์เรย์แผนที่ เพื่อสร้างแผนที่ขนาดเล็ก

3.2.4.4 ShowMiniMap

ทำการแสดงแผนที่ขนาดเล็กออกทางจอแสดงผล

3.2.4.5 Show

ทำการแสดงภาพแผนที่จริง และแผนที่ขนาดเล็กออกทางจอแสดงผล

3.2.4.6 รายละเอียดและแนวคิดในการสร้างแผนที่

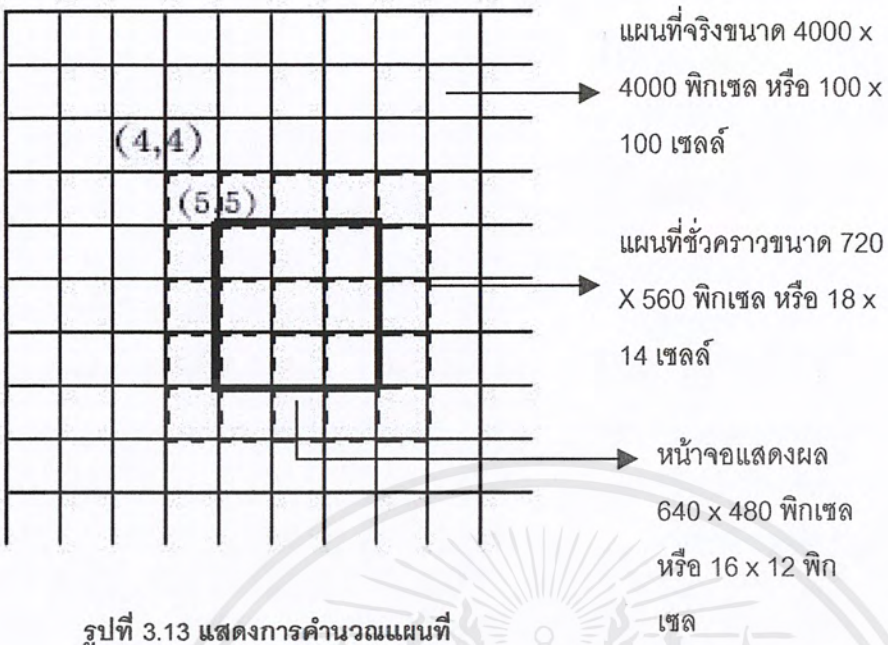
ค่าคงที่ที่จำเป็นต้องใช้และถูกประกาศอยู่ใน map.h

MAX_MAP_WIDTH	ค่าความกว้างของแผนที่ที่เป็นไปได้มากสุด หน่วยเป็น เซลล์
MAX_MAP_HEIGHT	ค่าความสูงของแผนที่ที่เป็นไปได้มากสุด หน่วยเป็นเซลล์
MAP_WIDTH	ค่าความกว้างของแผนที่ หน่วยเป็นเซลล์
MAP_HEIGHT	ค่าความสูงของแผนที่ หน่วยเป็นเซลล์
TEMP_WIDTH	ค่าความกว้างของแผนที่ชั่วคราว หน่วยเป็นเซลล์
TEMP_HEIGHT	ค่าความสูงของแผนที่ชั่วคราว หน่วยเป็นเซลล์
MAP_SCREEN_WIDTH	ค่าความกว้างของแผนที่ที่แสดงบนจอแสดงผล หน่วยเป็นพิกเซลล์
MAP_SCREEN_HEIGHT	ค่าความสูงของแผนที่ที่แสดงบนจอแสดงผล หน่วยเป็นพิกเซลล์
MAP_CELL_WIDTH	ค่าความกว้างของแผนที่ที่แสดงบนจอแสดงผล หน่วยเป็นเซลล์
MAP_CELL_HEIGHT	ค่าความสูงของแผนที่ที่แสดงบนจอแสดงผล หน่วยเป็นเซลล์
MINIMAP_POS_X	ค่าตำแหน่งในแกน X ของแผนที่ขนาดเล็ก
MINIMAP_POS_Y	ค่าตำแหน่งในแกน Y ของแผนที่ขนาดเล็ก
SCROLL_MOVE	จำนวนพิกเซลที่ใช้ในการเลื่อนหน้าจอแต่ละครั้ง
MapArr	อาร์เรย์เก็บข้อมูลของแผนที่ทั้งหมด

1 เซลล์มีขนาดเท่ากับ 40 พิกเซล

ข้อมูลแผนที่จะถูกเก็บอยู่ในไฟล์ map จะมีการอ่านค่าจากไฟล์ .map เก็บไว้ในอาร์เรย์ชื่อ MapArr หลังจากนั้นจะทำการสร้าง off screen surface เพื่อ เก็บแผนที่ชั่วคราวไว้ โดยแผนที่ชั่วคราวนี้จะมีขนาดมากกว่าขนาดของแผนที่ที่แสดงออกทางจอภาพด้านละ 1 เซลล์หรือด้านละ 40 พิกเซล นั่นคือถ้าหน้าจอมีขนาด 640 X 480 พิกเซล และ 1 เซลล์ของแผนที่มีขนาด 40 พิกเซล ดังนั้นใน 1 หน้าจอจะสามารถแสดงแผนที่ได้ 16 X 12 เซลล์ และแผนที่ชั่วคราวจะมีขนาดมากกว่าขนาดของแผนที่ที่แสดงออกทางจอภาพด้านละ 1 เซลล์ ดังนั้นแผนที่ชั่วคราวมีขนาด 18 X 14 เซลล์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.13 แสดงการคำนวณแผนที่

เมื่อทำการสร้าง Off screen ของแผนที่ชั่วคราวเรียบร้อยแล้ว จะทำการนำภาพใส่ในแผนที่ชั่วคราวนี้โดยทำการตรวจสอบกับอาร์เรย์ MapArr ว่าที่ตำแหน่ง (x , y) ของแผนที่ชั่วคราวควรนำภาพใดใส่ลงไป

ตัวอย่าง การนำภาพใส่ในแผนที่ชั่วคราว

สมมติข้อมูลในอาร์เรย์ MapArr มีค่า = { { 0,1,2 },{ 1,4,0 },{ 1,2,0 } }

0	1	2	3	4
5	6	7	8	9

รูปที่3.14 ลักษณะของข้อมูลภาพที่อยู่ในไฟล์ tilemap.bmp

ข้อมูลของ MapArr ในตำแหน่งที่ [1][1] มีค่าเป็น 4 ดังนั้นจะนำค่า 4 ไปเปรียบเทียบกับข้อมูลที่อยู่ในไฟล์ tilemap.bmp แล้วจะนำภาพที่ 4 ที่อยู่ไฟล์ tilemap นั้นมาใส่ลงในแผนที่จำลองตำแหน่งที่ (1,1) เมื่อนำภาพใส่แผนที่จำลองเรียบร้อยก็จะเริ่มอ่านค่า ในอาร์เรย์ใหม่ทำเช่นนี้จนได้ภาพเต็มแผนที่ชั่วคราว

เมื่อต้องการแสดงภาพออกทางจอแสดงภาพก็จะดึงข้อมูลที่อยู่ในแผนที่ชั่วคราวขึ้นมาแสดง

การแสดงภาพแผนที่จะเกิดขึ้นอยู่ 2 กรณี

1.เกิดจากเมาส์ถูกเลื่อนไปจนสุดตำแหน่งของขอบจอด้านใดด้านหนึ่ง

2.เกิดจากการคลิกเมาส์ปุ่มซ้ายบนแผนที่เล็ก

สำหรับกรณีแรกเมาส์ถูกเลื่อนไปจนสุดตำแหน่งของขอบจอแสดงผลด้านใดด้านหนึ่งจะมีตัวแปรที่ใช้ในการตรวจสอบการเลื่อนแผนที่คือ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

delx ถ้ามีค่าเป็นบวก แสดงว่าแผนที่ถูกเลื่อนไปด้านขวา ถ้ามีค่าเป็นลบแสดงว่าแผนที่ถูกเลื่อนไปทางซ้าย

dely ถ้ามีค่าเป็นบวก แสดงว่าแผนที่ถูกเลื่อนลงด้านล่าง ถ้ามีค่าเป็นลบแสดงว่าแผนที่ถูกเลื่อนขึ้นด้านบน

โดยที่ค่า delx และ dely จะเพิ่มหรือลดลง เกิดจากการตรวจสอบตำแหน่งของเมาส์ว่า เมาส์อยู่ที่ขอบจอด้านใด ถ้าเมาส์อยู่ที่ขอบจอด้านบน ค่า dely จะถูกลดลง ถ้าเมาส์อยู่ที่ขอบจอด้านล่างค่า dely จะเพิ่มขึ้น ถ้าเมาส์อยู่ที่ขอบจอด้านซ้ายค่า delx จะลดลง และถ้าเมาส์อยู่ที่ขอบจอด้านขวาค่า delx จะเพิ่มขึ้น โดยในโปรแกรมจะเขียนโค้ดไว้ดังนี้

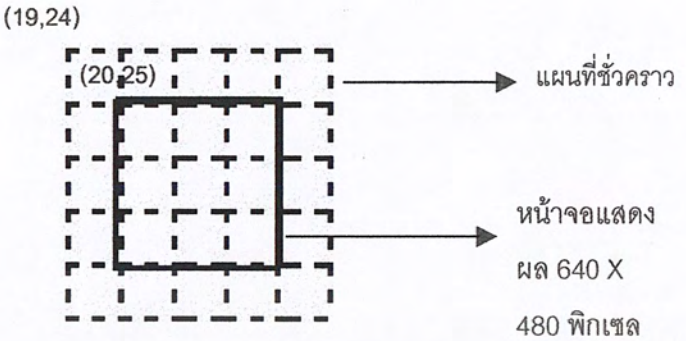
```
if (mouse_x >= 639)
        {
                mouse_x = 639;
                lpmap[0].delx = lpmap[0].delx + SCROLL_MOVE;
        }
        else
        if (mouse_x <= 0)
        {
                mouse_x = 0;
                lpmap[0].delx = lpmap[0].delx - SCROLL_MOVE;
        }

        if (mouse_y >= 479)
        {
                mouse_y= 479;
                lpmap[0].dely = lpmap[0].dely + SCROLL_MOVE;
        }
        else
        if (mouse_y <= 0)
        {
                mouse_y = 0;
                lpmap[0].dely = lpmap[0].dely - SCROLL_MOVE;
        }
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

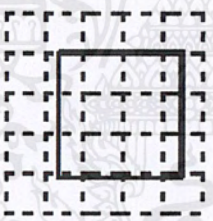
ตัวอย่าง การคำนวณเพื่อหาตำแหน่งเมื่อเกิดการเลื่อนแผนที่

สมมติตำแหน่ง Cell X และ Cell Y ที่อยู่มุมซ้ายบนของหน้าจอคือ (20,25) หรือตำแหน่งพิกเซลล์ที่ (800,1000)



รูปที่ 3.15 รูปตัวอย่างการคำนวณเพื่อหาตำแหน่งเมื่อเกิดการเลื่อนแผนที่

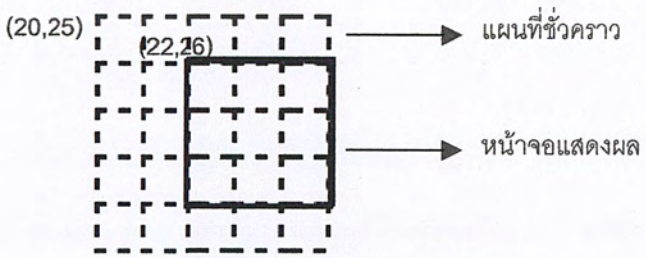
เมื่อเมาส์อยู่ที่ตำแหน่งขวาสุดของภาพค่า delx จะถูกเพิ่มขึ้นตามค่า SCROLL_MOVE สมมติค่า SCROLL_MOVE เป็น 10 ดังนั้น delx จะมีค่า delx + 10 หมายความว่าแผนที่ถูกเลื่อนไปด้านขวา 10 พิกเซล ดังนั้นการแสดงผลภาพแผนที่จะเริ่มแสดงที่ตำแหน่งพิกเซลล์ที่(810,1000)



รูปที่ 3.16 แสดงแผนที่ตำแหน่งใหม่หลังจากการเลื่อน

เมื่อค่า delx หรือ dely มีค่ามากกว่าเท่ากับ 40 หรือน้อยกว่าเท่ากับ –40 จะต้องทำการโหลดข้อมูลเข้ามายังแผนที่ชั่วคราวใหม่อีกครั้งหนึ่ง

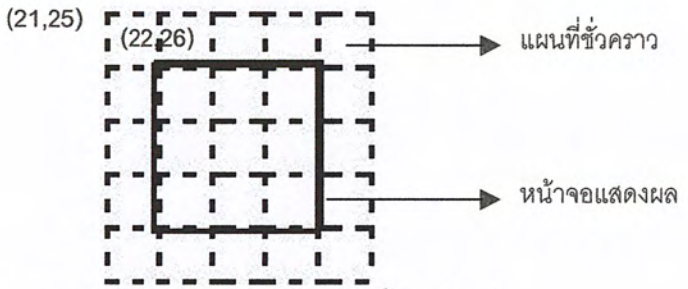
ตัวอย่างเมื่อค่า delx มีค่าเป็น4



รูปที่ 3.17 แสดงแผนที่ที่ถูกเลื่อนจนสุดแผนที่ชั่วคราว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เมื่อค่า delx = 40 ทำการโหลดข้อมูลเข้าสู่แผนที่ชั่วคราวอีกครั้ง



รูปที่ 3.18 แสดงการโหลดแผนที่ชั่วคราวขึ้นมาใหม่

ในกรณีที่2 เกิดการคลิกเมาส์ปุ่มซ้ายบนแผนที่เล็ก

อ่านค่าตำแหน่งของเมาส์บนแผนที่เล็ก โดยแผนที่เล็ก 1 เซลล์จะมีขนาดเท่ากับ 1 พิกเซลล์ เมื่อได้ค่าตำแหน่งในแนวแกน x และแนวแกน yของเมาส์แล้วจะนำค่าตำแหน่งในแกน x ลบออกด้วยค่า (MAP_CELL_WIDTH / 2) และนำค่าตำแหน่งแกน y ลบออกด้วยค่า (MAP_CELL_HEIGHT / 2)ก็จะได้ค่าตำแหน่งเซลล์ซ้ายบนที่ต้องการแสดงแผนที่ออกมา

การสร้างแผนที่ขนาดเล็ก

ทำการสร้าง off screen surface ขนาดเท่าจำนวนความกว้างและความยาวของแผนที่หน่วยเป็นเซลล์ อ่านค่าจากอาร์เรย์ MapArr ทุกค่าแล้วนำค่าที่อ่านได้จากอาร์เรย์ MapArr เปรียบเทียบกับไฟล์ minitile นำภาพที่อยู่ในไฟล์ minitile ที่ผ่านการเปรียบเทียบแล้วใส่ลงใน off screen surface

3.2.5 ส่วนยูนิต

ส่วนนี้จะประกอบด้วยคลาส GI_Unit ภายในคลาสจะประกอบด้วยเมธทอดการทำงานของยูนิต ทั้งการเดิน การต่อสู้ การรอรับคำสั่ง การแสดงภาพยูนิต สถานะของตำแหน่ง ชนิดรวมถึงรายละเอียดปลีกย่อยอื่นๆ

```
class GI_Unit
{
        public :
                int             ID;             // รหัสของยูนิตนั้นๆ
                int             hp;             // พลังชีวิต
                int             ap;             // พลังโจมตี
                int             dp;             // พลังป้องกัน
                int             reload_time;     // เวลาในการบรรจุกระสุน
                int             time_count;     // ค่าเวลา
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
int                 fire_range;             // ระยะยิง
int                 type;                   // ชนิดของยูนิต
int                 order;                  // คำสั่งที่ยูนิตนั้นๆต้องปฏิบัติ
int                 status;                 // ค่าสถานะของยูนิต
int                 pos;        // ตำแหน่งของภาพที่ใช้แสดงท่าทางของยูนิต
int                 force;                  // ฝ่าย
int                 group;                  // กลุ่มของยูนิต
int                 desX,desY;              // ตำแหน่งของเป้าหมาย
int                 solX,solY;              // ตำแหน่งของยูนิต
int                 targetID;               // รหัสของยูนิตเป้าหมาย
GI_Unit*        target;             // ออปเจคของยูนิตเป้าหมาย

int     frame;          // ค่าเฟรมที่ใช้ในการแสดงภาพเคลื่อนไหวของยูนิต
int     frame_move;     // ค่าเฟรมที่ใช้ในการแสดงภาพเคลื่อนไหวของเคอร์เซอร์
                        เมาส์
int     attack_by;      //รหัสของยูนิตที่ทำการโจมตียูนิตนี้อยู่
BOOL    attack;         // ค่าแฟลกตรวจสอบว่าทำการโจมตีอยู่หรือไม่
BOOL    move;           // ค่าแฟลกใช้ตรวจสอบว่ากำลังเคลื่อนที่อยู่หรือไม่
BOOL    selected;               // ค่าแฟลกใช้ตรวจสอบว่าถูกเลือกอยู่หรือไม่
RECT    unit_pos;       // สี่เหลี่ยมไว้กำหนดรูปของยูนิต
BOOL    moveClicked;    //ค่าแฟลกไว้ตรวจสอบว่ามีการคลิกเมาส์เพื่อสั่งให้ยูนิตเดิน
                        หรือไม่
BOOL    attackClicked;
char    buffer[256];    //ค่าแฟลกไว้ตรวจสอบว่ามีการคลิกเมาส์เพื่อสั่งให้ยูนิตโจม
                        ตีหรือไม่

GI_Unit();
~GI_Unit(void);

void Init(int x, int y,int id, int force_type);
void Order(GI_Unit* enemy, int map_x, int map_y);
int  DetectEnemy(int x, int y);
void Move(void);
void Attack(void);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
        void Hold(Gl_Unit* enemy);
        void Standby(Gl_Unit* enemy);
        void Show(int map_x, int map_y);
};
```

หลักการและการทำงานของเมธทอดในคลาส Gl_Unit

3.2.5.1 Gl_Unit :: DetectEnemy

จะทำการตรวจสอบ ที่ตำแหน่ง x , y ใดๆมียูนิตของฝ่ายตรงข้ามอยู่หรือไม่

3.2.5.2 Gl_Unit :: Order

ตรวจสอบข้อมูลที่อยู่ในตัวแปร order แล้วทำตามข้อมูลนั้น ถ้าข้อมูลที่อยู่ในตัวแปร order คือ MOVE จะทำการเรียกเมธทอด Move ถ้าข้อมูลที่อยู่ในตัวแปร order คือ ATTACKจะทำการเรียกเมธทอด Attack าข้อมูลที่อยู่ในตัวแปร order คือ STANDBY จะทำการเรียกเมธทอด Standbyแต่ถ้าข้อมูลที่อยู่ในตัวแปร order คือ HOLD จะทำการเรียกเมธทอด Hold

3.2.5.3 Gl_UNIT :: Standby

เมื่อเรียกใช้เมธทอดนี้ยูนิตจะยืนอยู่กับที่ ถ้ายูนิตฝ่ายตรงข้ามมาอยู่ในระยะยิง ยูนิตนี้จะทำการเปลี่ยนค่า order จาก STANDBY เป็น ATTACK แล้วทำการยิงตอบโต้ถ้ายูนิตฝ่ายตรงข้ามออกนอกระยะยิง ยูนิตนี้จะทำการเคลื่อนที่ตาม เมื่อยูนิตฝ่ายศัตรูเข้ามาในระยะยิงอีกครั้งก็จะทำการยิงทันที

3.2.5.4 Gl_Unit :: Hold

ลักษณะการทำงานของเมธทอดนี้จะคล้ายกับเมธทอด Standby จะมีข้อแตกต่างเล็กน้อยคือ เมื่อยูนิตฝ่ายตรงข้ามออกนอกระยะยิง ยูนิตนี้จะไม่เคลื่อนที่ตาม แต่ถ้ายูนิตฝ่ายตรงข้ามเข้ามาในระยะยิงอีกครั้ง ยูนิตนี้ก็จะทำการโจมตีทันที

3.2.5.5 Gl_Unit :: Attack

เมื่อเรียกใช้เมธทอดนี้ ยูนิตจะทำการโจมตียูนิตฝ่ายตรงข้ามจน ยูนิตฝ่ายตรงข้ามอยู่ในสถานะ DEAD ยูนิตนี้จะอยู่ในสถานะ DEAD เมื่อผู้เล่นทำการออกคำสั่งอื่นให้กับยูนิต

3.2.5.6 Gl_Unit :: Move

เมื่อเรียกใช้ ยูนิตจะทำการเคลื่อนที่ไปยังตำแหน่งที่ระบุในตัวแปร desx และ desy

3.2.5.7 Gl_Unit :: Show

เมื่อเรียกใช้เมธทอดนี้ ยูนิตที่มีสถานะอื่นที่ไม่ใช่ DEAD ทำการแสดงภาพยูนิตออกทางจอแสดงผล

3.2.6 ส่วนของการติดต่อการ์ดเสียงและการแสดงเสียง

ในส่วนนี้ประกอบด้วยฟังก์ชั่นที่ทำหน้าที่ติดต่อกับการ์ดเสียงโดยผ่านทาง Direct Sound และทำการแสดงเสียงออกทางลำโพง การโหลดไฟล์ WAV

ฟังก์ชั่นที่ใช้ในส่วนนี้มีดังนี้

-Dsound_Init

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

-Dsound_Shutdown

-Load_WAV

-Load_AllSound

-Stop_Sound

-Stop_AllSound

-Delete_Sound

-Delete_AllSound

-Status_Sound

-Play_Sound

หลักการ

3.2.6.1 Dsound_Init

ฟังก์ชั่นทำการเตรียมให้ระบบพร้อมที่จะใช้ Direct Sound โดยทำการสร้าง Direct Sound ออปเจค กำหนดลักษณะการทำงานของ DirectSound

3.2.6.2 Direct_Shutdown

เมื่อยกเลิกการใช้ DirectSound จะต้องยกเลิกออปเจคของ DirectSound เพื่อคืนหน่วยความจำกลับสู่ระบบ

3.2.6.3 Load_WAV

ทำการสร้าง Direct Sound Buffer โหลดข้อมูลเสียงจากไฟล์เข้าสู่หน่วยความจำเพื่อเตรียมที่จะแสดงเสียงนั้น ฟังก์ชั่นนี้จะรับพารามิเตอร์เป็นชื่อไฟล์ที่มีนามสกุล WAV เพื่อทำการโหลดไฟล์นั้นจากดิสก์ ถ้าสามารถโหลดไฟล์ได้สำเร็จจะคืนค่าเป็นเลขรหัสกลับมา และจะใช้เลขรหัสนี้ในการอ้างอิงเสียงที่โหลดขึ้นมา

3.2.6.4 Load_AllSound

ทำการโหลดไฟล์เสียงที่จำเป็นต้องใช้ในโปรแกรมเกมทั้งหมด เข้าสู่หน่วยความจำ

3.2.6.5 Stop_Sound

ใช้เมื่อต้องการหยุดเสียงที่กำลังเล่นอยู่ในขณะนั้น

3.2.6.6 Stop_All_Sound

ใช้เมื่อต้องการหยุดเสียงทุกเสียงที่กำลังเล่นอยู่ขณะนั้น

3.2.6.7 Delete_Sound

ทำการลบเสียงออกจากหน่วยความจำและทำการปล่อย DirectSound buffer

3.2.6.8 Delete_All_Sound

ทำการลบเสียงทุกเสียงที่มีการโหลดเข้ามาออกจากหน่วยความจำ

3.2.6.9 Play_Sound

ทำการเล่นเสียงที่ถูกโหลดเข้ามาแล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 4
ผลการทดลองและการวิเคราะห์ปัญหา

การทดลองและการวิเคราะห์ปัญหาที่กล่าวถึงในบทนี้ จะเป็นขั้นตอนการทดสอบและผลการทดสอบที่ได้ในแต่ละขั้น โดยผลการทดสอบนี้ จะถูกนำไปวิเคราะห์ถึงปัญหาและแนวทางในการพัฒนาต่อไปในอนาคต โดยทางผู้จัดทำหวังว่าจะเป็นประโยชน์แก่ผู้ที่จะนำไปศึกษาเพื่อพัฒนาต่อ และเพื่อให้เห็นถึงปัญหา และข้อดีข้อเสียของปัญหานี้โดยที่ผู้ศึกษาไม่จำเป็นต้องทำการทดสอบเพื่อหาปัญหา และนำไปพัฒนาต่ออีกครั้งหนึ่ง

คุณสมบัติของระบบที่นำมาทดสอบ

- ระบบปฏิบัติการ Microsoft Window 98
- ติดตั้ง Microsoft DirectX เวอร์ชั่น 7.0
- เครื่องคอมพิวเตอร์ที่มีซีพียูความเร็ว 300 MHz
- หน่วยความจำหลักขนาด 32 MByte

ขั้นตอนการดำเนินการทดสอบ

- ทำการติดตั้งตัวโปรแกรมและซอฟท์แวร์ที่จำเป็นต้องใช้
- ทำการรันและประมวลผลภายใต้ระบบที่กำหนด
- ตรวจสอบการใช้คำสั่งของยูนิตและการทำงานของยูนิต
- ตรวจสอบการทำงานของแผนที่โดยการใช้เมาส์
- ตรวจสอบการตอบโต้ของฝ่ายคอมพิวเตอร์ขณะทำการเล่น
- ตรวจสอบการการหา Error

การประเมินผล

- หลังจากติดตั้งตัวโปรแกรมแล้วสามารถใช้โปรแกรมได้
- การประมวลผลของโปรแกรมจะต้องทำความเร็วในระดับที่ยอมรับได้
- การตอบโต้ของคอมพิวเตอร์จะต้องเป็นไปตามที่กำหนดไว้
- การใช้คำสั่งของยูนิตเป็นไปตามรูปแบบ
- การเลื่อนแผนที่ถูกต้องตามเป้าหมาย
- จำนวน Error ที่เกิดขึ้นต้องไม่มี หรือน้อยที่สุด

ต่อไปจะกล่าวถึงรายละเอียดของขั้นตอนการทดสอบและผลการทดสอบที่ได้

4.1 ขั้นตอนการทดสอบการติดตั้งโปรแกรมและซอฟท์แวร์ที่จำเป็น

- ติดตั้ง ตัวโปรแกรมได้โดยคลิกไฟล์ “SETUP.EXE”
- ตัวโปรแกรมและซอฟท์แวร์จะถูกทำการติดตั้ง
- ผู้ใช้ทำการรันโปรแกรมได้โดยคลิกที่ไฟล์ “GI.EXE”

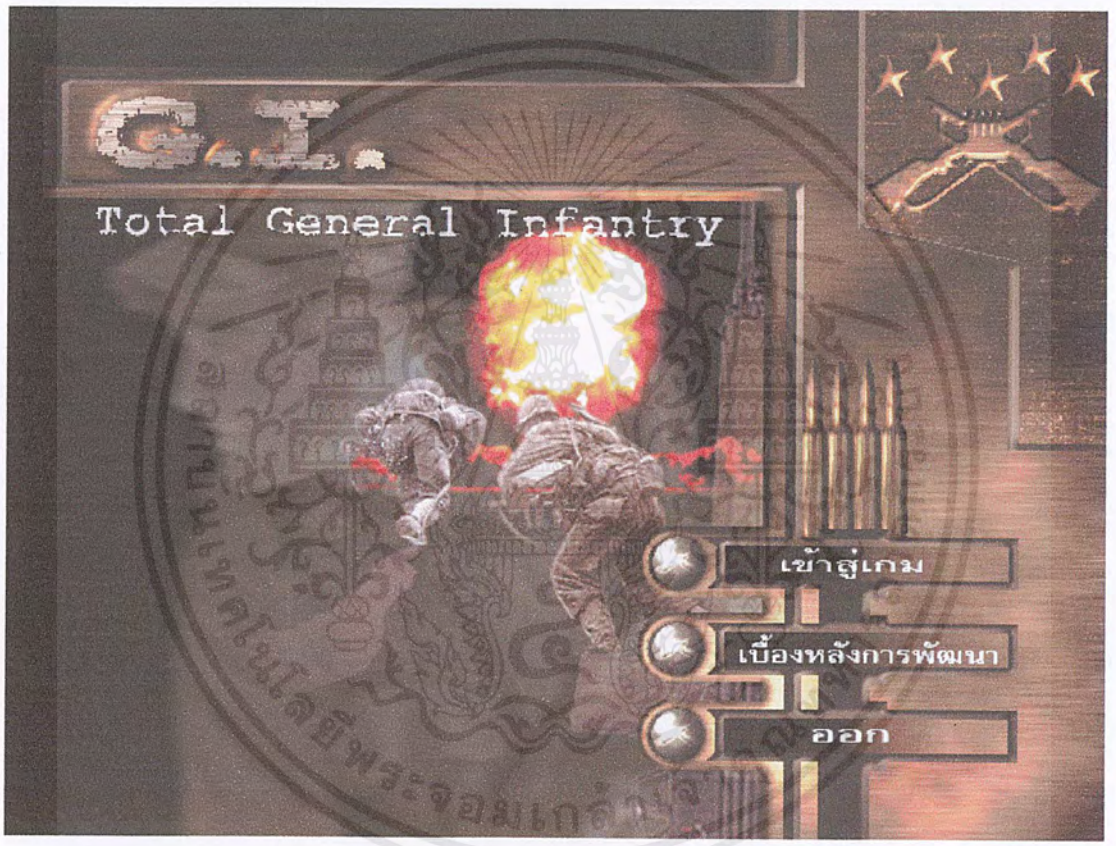
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ผลการทดสอบ

- ติดตั้งโปรแกรมและ ซอฟท์แวร์ที่จำเป็นลงบนเครื่องคอมพิวเตอร์ได้
- ผู้ใช้สามารถรันโปรแกรมได้
- เนื้อที่ของ ฮาร์ดดิสก์ถูกใช้ไปประมาณ 50 Mbyte

4.2 ขั้นตอนการทดสอบการรันและประมวลผลภายใต้ระบบที่กำหนด

รันโปรแกรมภายใต้เครื่องคอมพิวเตอร์ที่กำหนด โดยคลิกที่ไฟล์ "GI.EXE" จะได้จอภาพดังนี้



รูปที่ 4.1 ภาพหน้าจอก่อนเข้าสู่เกม

จากรูปจะเห็นได้ว่ามีทางเลือกสามทาง ดังนี้

1. เข้าสู่เกม — เริ่มเล่นเกม
2. เบื้องหลังการพัฒนา — บทความที่กล่าวถึงผู้ที่อยู่เบื้องหลังการพัฒนา
3. ออก — ออกจากเกม

ผลการทดสอบ

- สามารถรันโปรแกรมได้บนระบบที่กำหนด
- ความเร็วในการแสดงภาพหน้าจอแรก อยู่ในขั้นดี

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- สามารถเลือกรายการที่แสดงได้ตามความต้องการของผู้เล่น

4.2.1 ขั้นตอนการทดสอบจะเริ่มต้นที่การเลือกทางเลือก “เบื้องหลังการพัฒนา” จะได้หน้าจอดังนี้

รูปที่ 4.2 ภาพแสดงเบื้องหลังการพัฒนา

ผลการทดสอบ

- แสดงหน้าจอเบื้องหลังการพัฒนาได้ถูกต้อง
- สามารถย้อนกลับมาหน้ารายการหลักได้โดยการคลิกปุ่มตกลง ด้านล่างขวามือ

4.2.2 ขั้นตอนการเลือกรายการ “ออก” จากรายการหลัก

ผลการทดสอบ

- ตัวโปรแกรมปิด
- โปรแกรมคืนค่าทรัพยากรให้สู่หน่วยความจำของระบบ
- ความเร็วที่ได้จากการออกจากเกมอยู่ในขั้นดี

4.2.3 ขั้นตอนการเลือกรายการ “ เข้าสู่เกม “ จากรายการหลัก จะได้หน้าจอดังนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

รูปที่ 4.3 **ภาพหน้าจอขณะกำลังเล่น**

ผลการทดสอบ

- โปรแกรมแสดงภาพหน้าจอขณะทำการเล่นได้ดังรูปที่ 4.3
- เคลื่อนเมาส์ได้ตามขอบเขตที่กำหนด
- แผนที่สามารถแสดงได้
- สถานะของยูนิตถูกแสดง
- ส่วนควบคุมยูนิตแสดงได้ตามที่กำหนด
- ยูนิตที่กำหนดให้ปรากฏตอนเริ่มเป็นไปตามที่กำหนด

4.3 ขั้นตอนการทดสอบการใช้คำสั่งของยูนิตและการทำงานของยูนิต

การทดสอบการใช้คำสั่งของยูนิตวิธีการใช้ และผลการทำงานจะแสดงได้ดังตารางที่ 4.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดสอบ	วิธีทดสอบ	ผลการทดสอบ
ตรวจสอบการเลือกทหาร 1 ยูนิต	คลิกเมาส์ปุ่มซ้าย ที่ตัวทหาร	ปรากฏเคอร์เซอร์สีเหลืองล้อมรอบตัวทหารที่ถูกเลือก และที่จอแสดงสถานะของทหารปรากฏขึ้น
ตรวจสอบการเลือกทหารเป็นกลุ่ม	ลากเมาส์โดยกดปุ่มซ้ายค้างตีเป็นกรอบสี่เหลี่ยมล้อมรอบกลุ่มทหารที่ต้องการเลือก แล้วปล่อย	ปรากฏเคอร์เซอร์สีเหลืองล้อมรอบตัวทหารทุกตัวที่ทำการเลือกโดยมีจำนวนทหารอย่างน้อย 1 ตัว และมากที่สุดจำนวน 12 ตัว โดยจอแสดงสถานะทหารไม่ปรากฏ
ตรวจสอบการตั้งกลุ่มทหารเป็นกลุ่มๆ	ทำวิธีเดียวกับการเลือกทหารเป็นกลุ่ม หลังจากนั้นทำการตั้งเบอร์กลุ่มโดยกดปุ่ม Ctrl ตามด้วยหมายเลขกลุ่ม 0-9	กดหมายเลขระหว่าง 0-9 เคอร์เซอร์จะเปลี่ยนไปตามตัวทหารแต่ละกลุ่ม
ตรวจสอบการเลือกยูนิตประเภทสิ่งก่อสร้าง	คลิกเมาส์ปุ่มซ้ายที่สิ่งก่อสร้าง	มีเคอร์เซอร์สีเหลืองล้อมรอบสิ่งก่อสร้างที่ถูกเลือก และปรากฏสถานะของสิ่งก่อสร้าง
ตรวจสอบคำสั่งเดินของยูนิตทหาร	เลือกยูนิตทหาร คลิกเมาส์ปุ่มขวาที่จุดหมายที่ต้องการให้ทหารเดิน	ทหารเดินไปที่จุดเป้าหมายที่ทำการคลิก โดยที่ปุ่มคำสั่งจะแสดงสัญลักษณ์การเดิน เมื่อถึงเป้าหมายปุ่มคำสั่งหยุดก็จะถูกแสดงแทน
ตรวจสอบการยกเลิกการเลือกยูนิต	เลือกยูนิต คลิกเมาส์ปุ่มซ้ายที่จุดใดๆ ที่ไม่ใช้ตัวยูนิตนั้น	เคอร์เซอร์ล้อมรอบตัวยูนิตที่ถูกเลือกไว้หายไป และจอแสดงสถานะหายไป
ตรวจสอบการโจมตีของยูนิตทหาร	ทำการเลือกยูนิตที่ต้องการให้โจมตี แล้วคลิกเมาส์ปุ่มขวาที่ยูนิตฝ่ายคอมพิวเตอร์	ยูนิตที่ถูกเลือกให้โจมตีจะโจมตียูนิตที่ถูกเลือกโดยจะเดินไปในระยะโจมตีแล้วทำการโจมตี โดยปุ่มคำสั่งโจมตีจะถูกแสดง
ตรวจสอบการสร้างยูนิตทหารของโรงทหาร	ทำการเลือกสิ่งก่อสร้างโรงทหาร กดปุ่ม S ที่แป้นคีย์บอร์ด	ที่จอแสดงสถานะจะแสดงระยะเวลาในการสร้างยูนิตทหาร และยูนิตทหารปรากฏอยู่รอบๆ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตรวจสอบการสร้างสิ่งก่อสร้างของยูนิตที่มีความสามารถในการสร้าง	ทำการเลือกยูนิตที่มีความสามารถในการสร้างสิ่งก่อสร้าง คลิกเมาส์ปุ่มขวาไปที่ปุ่มคำสั่งสร้าง แล้วเลือกจุดที่ต้องการให้สิ่งก่อสร้าง สร้าง โดยคลิกเมาส์ปุ่มขวาที่จุดที่ต้องการ	สิ่งก่อสร้างปรากฏบนจุดที่ทำการเลือกในระยะเวลาที่กำหนด

ตารางที่ 4.1 วิธีการทดสอบและผลการทดสอบ

4.4 ขั้นตอนการทดสอบการทำงานของแผนที่

- เลื่อนเมาส์ไปสุดขอบจอทั้งสี่ด้าน
- เลื่อนเมาส์โดยคลิกปุ่มซ้ายค้างไว้แล้วเลื่อนไปบนแผนที่เล็ก
- เลือกยูนิตทหารแล้วสั่งให้เดินไปยังจุดบนแผนที่เล็ก

ผลการทดสอบ

- แผนที่เลื่อนไปตามทิศทางที่เมาส์ต้องการ
- ระบบแผนที่ (เช่น การอ้างอิงแผนที่อื่นๆนอกจอภาพ) สามารถทำงานได้ถูกต้อง
- แผนที่เล็กอ้างอิงตำแหน่งตามแผนที่ใหญ่ได้ถูกต้อง
- ทหารเดินไปตามจุดบนแผนที่เล็กและแสดงตำแหน่งบนแผนที่เล็ก
- ทุกยูนิตสามารถแสดงตำแหน่งบนแผนที่เล็ก

4.5 ขั้นตอนการทดสอบการตอบโต้ของคอมพิวเตอร์

- โจมตีทหารฝ่ายคอมพิวเตอร์
- ทำการบุกฐานฝ่ายคอมพิวเตอร์
- ป้องกันฐานจากการโจมตีของทหารฝ่ายคอมพิวเตอร์
- เดินผ่านระยะโจมตีของทหารฝ่ายคอมพิวเตอร์

ผลการทดสอบ

- ทหารฝ่ายคอมพิวเตอร์ทำการตอบโต้เมื่อถูกโจมตี
- ทหารฝ่ายคอมพิวเตอร์ป้องกันฐานโดนกำลังจำนวนหนึ่ง
- ทหารฝ่ายคอมพิวเตอร์ทำการโจมตีฐานฝ่ายผู้เล่น
- เมื่ออยู่ในระยะโจมตีของทหารฝ่ายคอมพิวเตอร์ ทหารฝ่ายคอมพิวเตอร์จะทำการโจมตี และถ้าทหารฝ่ายผู้เล่นหนี ทหารฝ่ายคอมพิวเตอร์จะติดตามจนกว่าจะทำลายทหารฝ่ายผู้เล่นได้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.6 ขั้นตอนการทดสอบการหาข้อผิดพลาดของโปรแกรม

ในขั้นตอนการทดสอบเพื่อหาข้อผิดพลาดจะไม่มีรูปแบบที่ชัดเจน แต่จะทำการทดสอบโดยการเล่นเกมหลายๆรอบ เพื่อหาข้อผิดพลาดและจากการทดสอบพบว่า ข้อผิดพลาดที่พบจะได้แก่

- พบข้อผิดพลาดในการเลื่อนแผนที่ คือไม่สามารถแสดงบางส่วนของยูนิตได้เมื่อยูนิตอยู่ระหว่างขอบจอ
- การตอบโต้ของคอมพิวเตอร์ ยังพบปัญหาเมื่อผู้เล่นตอบโต้ด้วย
- การแสดงท่าทางของยูนิต ยังมีข้อผิดพลาด
- การเคลื่อนที่ของยูนิตเป็นกลุ่มยังไม่ดี

4.7 ปัญหาและการวิเคราะห์

จากผลการทดสอบโปรแกรมทำให้ทราบถึงปัญหาดังนี้

4.7.1 ความล่าช้าในการประมวลผล

เมื่อยูนิตมีจำนวนมากขึ้น การสั่งการยูนิตทำได้ช้าลง และเป็นสาเหตุให้การแสดงผลแต่ละเฟรมทำได้ช้า ลักษณะที่แสดงออกมาคือ เคอร์เซอร์ของเมาส์ และการเคลื่อนไหวของทหารมีอาการขยับบ้าง นิ่งบ้าง เป็นช่วงๆ

สาเหตุของความช้าในการประมวลผลอาจเนื่องมาจากการเขียนโปรแกรมที่ไม่กระชับ และมีการวนลูปตรวจสอบค่ามากเกินไป

4.7.2 การตอบโต้ของฝ่ายคอมพิวเตอร์

ปัญหาที่สำคัญอีกปัญหาหนึ่งก็คือการตอบโต้ของฝ่ายคอมพิวเตอร์ซึ่งยังทำได้ไม่สมจริงนัก ยกตัวอย่างเช่น เมื่อทำการสั่งทหารให้เดินเข้าไปยังขอบเขตที่ฝ่ายคอมพิวเตอร์สามารถโจมตีถึง ฝ่ายคอมพิวเตอร์จะโจมตีทันที โดยไม่ดูถึงสถานการณ์ว่ากำลังได้เปรียบหรือเสียเปรียบ เมื่อสั่งให้ทหารฝ่ายเราเดินหนี ฝ่ายคอมพิวเตอร์จะทำการวิ่งไล่ตามโดยไม่สนใจด้านกลยุทธ์ใดๆ ทำให้ง่ายต่อการทำลาย

สาเหตุของปัญหาอาจมาจากการที่ยังเก็บรวบรวมข้อมูลด้านกลยุทธ์ได้น้อยจนเกินไป ทำให้ไม่สามารถตอบโต้ฝ่ายผู้เล่นได้ดีนัก และขั้นตอนการเขียนโปรแกรมที่มีความซับซ้อน

4.7.3 การเคลื่อนไหวของยูนิต

เนื่องจากปัญหาพิเศษที่ทำขึ้นนี้ ใช้ภาพเคลื่อนไหวเพียงแค่ 8 ทิศ แบ่งเป็นทิศละ 16 เฟรม ทำให้ภาพเคลื่อนไหวของยูนิตยังไม่สมจริงนัก แต่เนื่องจากทางผู้ศึกษาต้องการใช้ทรัพยากรในตัวโปรแกรมให้น้อยที่สุดจึงได้กำหนดเพียงแค่ 8 ทิศ และยังเพื่อต้องการให้การประมวลผลทำได้รวดเร็ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5
สรุปผลการวิจัย และข้อเสนอแนะ

การทำงานโครงงานนี้มีการวางแผนขั้นตอนการทำงานเป็นอย่างดี แต่ถึงแม้จะได้วางแผนแต่ละขั้นตอนไว้อย่างดี ก็ยังมีปัญหากับขั้นตอนบางขั้นตอน ซึ่งบางขั้นตอนเกิดจากการศึกษาการใช้ซอฟท์แวร์กราฟฟิกใหม่ๆ เนื่องจากต้องใช้ประสบการณ์และระยะเวลาพอสมควร ถึงจะเข้าใจและสามารถทำงานกับซอฟท์แวร์แต่ละตัวได้เป็นอย่างดี บางขั้นตอนเกิดจากความล่าช้าในการคิดระเบียบวิธีการโปรแกรมมิ่ง ซึ่งจำต้องศึกษาจากบทความภาษาอังกฤษตามโฮมเพจต่างๆ

โดยขั้นตอนต่างๆสรุปได้ดังนี้

- ขั้นตอนการออกแบบระบบเกม
- ขั้นตอนการเขียนโปรแกรม
- ขั้นตอนการสร้างตัวละครและภาพต่างๆ

5.1 ขั้นตอนการออกแบบ

ขั้นตอนการออกแบบเป็นขั้นตอนที่รวดเร็วที่สุด เนื่องมาจากผู้วิจัยมีความคิดต่างๆของแต่ละคนไว้อยู่แล้ว ซึ่งก็เกิดจากประสบการณ์ซึ่งเคยเล่นเกม และมีแนวคิดของเกมที่ตัวเองชื่นชอบ จึงนำมารวบรวมและสรุปเป็นรูปแบบของเกมในงานวิจัย

5.2 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้สามารถทำควบคู่ไปได้กับขั้นตอนการสร้างตัวละครและภาพต่างๆ โดยขั้นตอนนี้ใช้เวลานานที่สุดในงานวิจัย เนื่องจากเป็นเรื่องใหม่ซึ่งผู้วิจัยต้องทำการศึกษานานพอสมควร แล้วจึงลงมือเขียนโปรแกรม การทำงานในส่วนนี้มีซอฟท์แวร์ที่ใช้ในการพัฒนาดังนี้

- Microsoft Visual C++ เวอร์ชั่น 6.0
- Microsoft Direct X SKD เวอร์ชั่น 5.0

5.3 ขั้นตอนการสร้างตัวละครและภาพต่าง ๆ

ขั้นตอนนี้เป็นขั้นตอนที่ใช้ระยะเวลาหลังจากทำการเขียนโปรแกรมไปได้ระยะหนึ่ง เนื่องจาก การสร้างตัวละครเป็นงานที่ใช้เวลานาน และปริมาณมาก ปัญหาส่วนใหญ่อยู่ที่การทำงานกับซอฟท์แวร์กราฟฟิกต่างๆ ซึ่งต้องใช้เวลาศึกษาพอสมควร แต่หลังจากได้ศึกษาและชินกับการทำงาน ทำให้การทำงานเป็นไปอย่างราบรื่น ขั้นตอนนี้ใช้ Application ดังนี้

- MetaCreation Poser เวอร์ชั่น 3.0
- 3Dstudio Max เวอร์ชั่น 3.0
- Adobe Photoshop เวอร์ชั่น 5.5

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

MetaCreation Poser เป็นซอฟท์แวร์ที่ใช้สำหรับออกแบบสร้างตัวคนต่างๆ โดยเป็นระบบภาพสามมิติ ใช้สำหรับการออกแบบท่าเคลื่อนไหวของยูนิตต่างๆ ในมุมที่แตกต่างกันออกไป

3Dstudio Max เป็นซอฟท์แวร์ที่ใช้ออกแบบภาพสามมิติต่างๆ รวมถึงปืนสำหรับทหาร

Adobe Photoshop เป็นซอฟท์แวร์ใช้สำหรับการตกแต่งภาพทุกชนิด เรานำมาใช้ตกแต่งและตัดต่อภาพยูนิต และภาพแผนที่ในภูมิประเทศต่างๆกัน

สรุปแล้วในการทำงานวิจัยเพื่อสร้างเกมโดยตอบโต้กับคอมพิวเตอร์ การทำงานต่างๆ ปัญหาที่เกิดขึ้น ตลอดจนการได้ติดต่อกับผู้ผลิตเกมในต่างประเทศ เพื่อทำการศึกษานี้ ทำให้ผู้วิจัยทราบถึงแนวทางในการผลิตเกมในตลาดซอฟท์แวร์ไทย เกมในปัจจุบันบางเกมคนไทยเป็นผู้ผลิต แต่ไม่สามารถบอกได้ว่าเป็นคนไทย ก็เนื่องมาจากบริษัทต่างๆที่ถือลิขสิทธิ์เป็นบริษัทต่างประเทศ ดังนั้นการที่คนไทยจะก่อตั้งบริษัทซอฟท์แวร์ขึ้นก็มีความเป็นไปได้ แต่ปัญหาที่สำคัญที่สุดในตลาดซอฟท์แวร์ไทยก็คือปัญหาละเมิดลิขสิทธิ์ซอฟท์แวร์ ซึ่งเป็นสาเหตุสำคัญที่ทำให้คนไทยไม่กล้าที่จะผลิตซอฟท์แวร์คุณภาพออกมาแข่งขันกัน และปัญหาอีกอย่างในตลาดซอฟท์แวร์เกมก็คือการให้ความสนใจในวิธีการผลิตของคนไทยน้อยเกินไป ในขณะเดียวกันก็บริโภคหรือให้ความสนใจในเนื้อหาของเกมมาก ทำให้ความรู้ในการผลิตยังไม่สามารถทำได้เทียบเท่ากับชาวต่างชาติซึ่งเป็นชาติผู้ผลิต ดังนั้นโครงงานนี้น่าจะเป็นแนวทางหนึ่งที่จะแนะนำให้ผู้สนใจได้นำไปศึกษาและผลิตเกมที่สามารถนำออกมาจำหน่ายได้จริงต่อไป

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ภาคผนวก

คู่มือการเล่น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

คู่มือการเล่นเกม

ความเป็นมา

เกมนี้จับเอาเรื่องราวในประวัติศาสตร์สมัยสงครามโลกครั้งที่ 2 เป็นจุดดำเนินเรื่อง โดยจะเน้นถึงการทำการรบของทหารราบ โดยส่วนใหญ่เกมที่มีอยู่ในปัจจุบันจะอาศัยหน่วยของทหารที่แข็งแกร่งที่สุดเข้าโจมตีกัน เช่น รถถัง ดังนั้นผู้เล่นก็จะผลิตแต่รถถัง ให้ได้มากที่สุด ดังนั้นรูปแบบของเกมจึงกลายเป็นว่าผู้ที่สามารถสร้างได้มากกว่าและหน่วยทหารแข็งแกร่งกว่าเป็นผู้ชนะ ในความเป็นจริงนั้น ใครมีจำนวนมากกว่าและกองทหารแข็งแรงกว่าย่อมได้ชัยชนะ แต่ในบางครั้งความจริงนั้นก็ไม่จริงเสมอไป การรบด้วยกำลังน้อยกว่าและกองทหารอ่อนแอกว่า ในบางครั้งอาจได้รับชัยชนะหากทำการรบด้วยสติปัญญาและการควบคุมที่ดี "ผู้นำที่ดีย่อมสร้างชัยชนะในความสูญเสียที่น้อยที่สุด" หรือการรบน้อยชนะมากนั่นเอง โดยอาศัยการควบคุมที่เป็นระเบียบ การวางแผนเป็นขั้นตอน และการบริหารทรัพยากรให้เหมาะสม และนั่นก็คือที่มาของแนวคิดของเกมนี้ คือ การรู้จักใช้สติปัญญาในการแก้ปัญหา การรู้จักใช้การวางแผนและมีเหตุผลในการแก้ปัญหา การใช้กำลังเข้าประหัตประหารในเกมนี้ จะสร้างความสูญเสียมากกว่าผลดี

ประสิทธิภาพของระบบที่แนะนำ

1. คอมพิวเตอร์ระดับความเร็ว 266 MHz
2. หน่วยความจำขนาด 32 Mbyte
3. อุปกรณ์อินพุทประเภทเมาส์ และคีย์บอร์ด
4. ซอฟท์แวร์ Microsoft DirectX ตั้งแต่เวอร์ชั่น 5 ขึ้นไป

ขั้นตอนการติดตั้งตัวโปรแกรม

1. คลิกที่ไฟล์ "setup.exe"
2. เลือกติดตั้งโปรแกรมระหว่างตัวโปรแกรม และซอฟท์แวร์ DirectX ถ้ามีอยู่แล้วให้ข้ามขั้นตอนการติดตั้ง ซอฟท์แวร์ Microsoft DirectX ไป
3. ทำตามขั้นตอน
4. เริ่มเล่นด้วยการคลิกไฟล์ที่ชื่อ "GI.exe" หรือคลิกที่ start menu ไฟล์ไอคอนชื่อ GI

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตัวละครและสิ่งก่อสร้างในเกม

ทหารราบ ป.ล.ย.

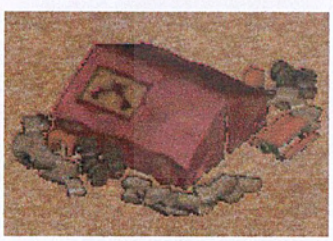
ตัวละครใช้ทหารราบ ปลย. (ปืนเล็กยาว) ในการทำการรบ ปืนที่ใช้เป็นปืนไรเฟิลแบบกึ่งอัตโนมัติ ใช้กันโดยทั่วไปในหน่วยทหารราบสมัยสงครามโลกครั้งที่สอง เป็นอาวุธประจำกายทหารของทั้งสองฝ่าย มีคุณสมบัติต่างๆดังนี้

คุณสมบัติของทหาร

- พลังชีวิตเต็ม 50
- แรงโจมตีของกระสุน 13
- การป้องกันตัว 3
- ราคา 25
- ทหารสามารถมีได้สูงสุด 30 ตัว หากตายไปก็สามารถสร้างเพิ่มได้เรื่อยๆ

สิ่งก่อสร้างในเกมมีสามชนิด โดยมีประโยชน์ต่างๆกัน ดังนี้

โรงทหาร



เป็นสิ่งก่อสร้างที่สำคัญที่สุด เพราะใช้สำหรับผลิตทหารซึ่งเป็นกุญแจสำคัญในการสู้รบ และด้วยเพราะเหตุนี้ โรงทหารจึงเป็นสิ่งก่อสร้างที่สำคัญยิ่งที่จะต้องปกป้องเป็นอันดับแรก โดยโรงทหารมีความสามารถในการสร้างสิ่งก่อสร้างอื่นๆอีกด้วย

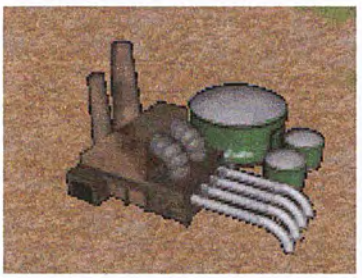
คุณสมบัติของโรงทหาร

- ใช้สร้างสิ่งต่างๆในเกม
- พลัง 850
- ความทนทาน 10
- โรงทหารมีหน้าที่โดยตรงใช้สำหรับสร้างทหาร การสร้างทหารสร้างได้โดยใช้ปุ่ม M (Marine) เมื่อกดแล้ว ส่วนแสดงสถานะจะเปลี่ยนเป็น progress bar แสดงถึงเวลาในการสร้าง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ไม่สามารถสร้างเพิ่มได้ มีได้แค่แห่งเดียว

โรงสูบน้ำมัน



เป็นสิ่งก่อสร้างซึ่งมีความสำคัญรองลงมาจากโรงทหาร โดยโรงสูบน้ำมันจะทำการสูบน้ำมันจากบาดาลขึ้นมา แล้วทำการแปลงน้ำมันให้เป็นทองคำ ซึ่งเป็นปัจจัยอีกอย่างหนึ่งใช้ในการจ้างหน่วยทหาร และซื้อวัตถุดิบต่างๆเพื่อใช้ในการก่อสร้างอาคาร ตามปกติทองคำจะได้มาจากรัฐบาลของประเทศต่างๆ แต่มีจำนวนน้อยมาก เนื่องจากอยู่ในภาวะสงคราม โดยทองคำจะถูกส่งมาเรื่อยๆขึ้นอยู่กับเวลา แต่หากเราทำการสร้างโรงสูบน้ำมันขึ้นมาจะทำให้เราสามารถหาทองคำได้เอง และมีปัจจัยมากพอที่จะใช้ผลิตทหารและสิ่งก่อสร้างอื่นๆ และทำให้ได้เปรียบฝ่ายศัตรู ด้วยเหตุนี้เองโรงสูบน้ำมันจึงเป็นจุดยุทธศาสตร์ที่สำคัญที่ควรป้องกันและวางแผนทำลายเป็นอันดับแรกก่อน เนื่องจากเป็นจุดที่สามารถชะลอการสะสมกำลังของฝ่ายศัตรูได้เป็นอย่างดี

คุณสมบัติของโรงสูบน้ำมัน

- ใช้สูบน้ำมันและแปลงเป็นทองคำ
- ราคา 200
- พลัง 600
- ความทนทาน 10
- สร้างโรงแรกจะทำให้ทองคำเพิ่มขึ้น 10 บาท
- เมื่อสร้างโรงถัดไป จะสามารถสูบน้ำมันได้ขึ้นมาได้เพียง 60% ของที่เคยสูบได้ในโรงล่าสุด ดังนั้น หากเดิมทองคำเพิ่ม 10 เมื่อสร้างโรงถัดไปจะทำให้ทองคำเพิ่ม 10+(60% ของ 10) คือ 6 ดังนั้นจะเพิ่มขึ้น 16 บาท และหากสร้างอีกโรงจะทำให้ทองคำเพิ่ม 10+6+(60% ของ 6) คือ 4 ดังนั้นจะเพิ่มขึ้น 10+6+4=20 ดังนั้นการสร้างโรงสูบน้ำมันจำนวนมากๆก็ไม่ได้ทำให้เงินเพิ่มขึ้นอย่างรวดเร็ว เนื่องจากน้ำมันมีจำนวนจำกัด
- สร้างจากโรงทหาร โดยที่โรงทหารจะมี icon รูปโรงสูบน้ำมัน เราสามารถสร้างได้โดยทำการใช้ปุ่ม O (Oil) และเลือกตำแหน่งที่จะสร้าง แล้วคลิกปุ่มซ้ายที่ตำแหน่งนั้นเพื่อทำการสร้าง ขณะสร้าง icon รูปโรงสูบน้ำมันที่โรงทหารจะเรืองแสงขึ้น

ป้อมปราการ

ป้อมปราการเป็นสิ่งก่อสร้างที่คุ้มค่าที่สุด เนื่องจากเป็นรูปแบบการตั้งรับที่เข้ากับคำสุภาษิตว่า "รบน้อย ชนะมาก" การเข้าต่อกรกับป้อมปราการเป็นสิ่งที่ไม่ใช่เรื่องง่าย การเข้าทำลายป้อมจำเป็นต้องใช้การวางแผนที่รัดกุม และให้เกิดประโยชน์มากที่สุด โดยจะต้องมีกำลังทหารจำนวนมาก เพราะป้อม ปราการมีอานุภาพรุนแรง พลังทำลายสูง และมีความแม่นยำมาก ดังนั้นไม่ใช่ เรื่องแปลกอะไรหากทหารจำนวน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

มากที่คุณส่งเข้าไปจะถูกยิงตายหมด

คุณสมบัติของป้อมปราการ

- ราคา 250
- พลัง 350
- พลังโจมตี 15
- ความทนทาน 10
- ป้อมปราการมีความสามารถพิเศษคือสามารถยิงได้เป็นระยะทางไกลกว่าทหารราบปกติ เพราะปืนที่ใช้ เป็นปืนกลกระบอกคู่ที่มีอานุภาพร้ายแรง และการยิงแต่ละครั้งยิงเป็นชุด ชุดละสามนัด จึงทำให้ทหารที่บุกรุกถูกยิงตายภายในสองถึงสามชุด
- การสร้างทำได้โดยการเลือกทหารมาสองตัว จะมี icon การสร้างป้อมปรากฏขึ้น เมื่อเราเลือกที่จะสร้าง ให้ใช้ปุ่มบนคีย์บอร์ดปุ่ม U (Bunker)แล้วคลิกปุ่มซ้ายที่ตำแหน่งที่แผนที่ทหารทั้งสองจะวิ่งมาที่จุดดังกล่าวแล้วทำการสร้างป้อม

หน้าจอการเล่น

จากรูปจะเป็นหน้าจอขณะทำการเล่น โดย จะแบ่งออกเป็น 4 ส่วน

- ส่วนการต่อสู้
- ส่วนแผนที่เล็ก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ส่วนสถานะยูนิต
- ส่วนแสดงสถานะคำสั่งการควบคุมของยูนิต

ส่วนการต่อสู้

เป็นส่วนที่ใช้แสดงการต่อสู้ การควบคุม และการใช้ยูนิตต่างๆ ยังแสดงถึงจำนวนเงิน และจำนวนยูนิตที่มีอยู่ โดยจำนวนเงินจะเพิ่มขึ้นตามเวลา และจำนวนทหารจะเพิ่มขึ้นเมื่อมีการสร้างทหารขึ้นมา โดยจำนวนทหารที่มีได้สูงสุดคือ 30 หน่วย

ส่วนแผนที่เล็ก

แสดงถึงแผนที่ทั้งหมดได้เกม สามารถแสดงตำแหน่งของยิตทุกตัวที่อยู่ในเกม ไม่ว่าจะอยู่นิ่งหรือทำการเคลื่อนไหว ส่วนนี้ยังสามารถใช้ในการชี้ตำแหน่งเพื่อสั่งยูนิตเดินไปยังตำแหน่งนั้นได้ หรือ ทำการชี้ตำแหน่งเพื่อดูแผนที่ใหญ่ได้อีกด้วย

ส่วนสถานะยูนิต

จะแสดงถึง คุณสมบัติของทหารแต่ละตัว ได้แก่ พลังชีวิต พลังโจมตี พลังป้องกัน และเวลาในการสร้าง

ส่วนแสดงสถานะคำสั่งการควบคุมของยูนิต

จะอธิบายด้วยสัญญาลักษณ์คำสั่งต่างๆ บอกว่าขณะนั้นยูนิตทำคำสั่งอะไรอยู่

การเลือกยูนิต

การเลือกยูนิตมีสองชนิด

- เลือกยูนิตเดียว (Single Selection)
- เลือกครั้งละหลายๆยูนิต (Multi Selection)

การเลือกยูนิตเดียว

ทำได้โดยการคลิกเมาส์ไปที่ตัวยูนิตที่ต้องการ หรือทำการคลิกเมาส์ค้างไว้แล้วลากกรอบสี่เหลี่ยมล้อมรอบตัวยูนิตนั้นๆก็ได้ เมื่อปล่อยเมาส์ ยูนิตที่ถูกล้อมกรอบไว้จะถูกเลือก

การเลือกยูนิตเป็นกลุ่ม

ทำได้โดยการคลิกเมาส์แล้วลากล้อมรอบตัวยูนิตทั้งหมดที่ต้องการ เมื่อปล่อยเมาส์ ยูนิตทั้งหมดที่อยู่ในกรอบสี่เหลี่ยมจะถูกเลือก

การจัดทัพ

การจัดทัพมีประโยชน์มากในการวางแผนเข้าโจมตี ซึ่งเราสามารถสั่งการให้ทหารหลายๆทัพเข้าโจมตีพร้อมๆกันทุกทิศทาง หรือวางแผนเข้าโจมตีหลอกล่อ แล้วสั่งให้ทัพอื่นๆเข้าโจมตีอีกทางหนึ่งได้ ซึ่งสามารถทำได้ดังนี้

1. ทำการเลือกทหารที่ต้องการเกณฑ์เข้าไว้ในทัพ ซึ่งควรเป็นทหารที่ใหม่ซึ่งยังแข็งแรงและพร้อมจะเข้าทำการรบ

2. เมื่อได้ทหารจำนวนที่ต้องการในหนึ่งทัพ ให้กดคีย์ Ctrl บนคีย์บอร์ดค้างไว้ แล้วเลือกลำดับของทัพ โดยเลือกกดแป้นคีย์เลข 1, 2 หรือ คีย์อื่นๆ ซึ่งมีได้ตั้งแต่คีย์ 0 ถึงคีย์ 9 ดังนั้นจึงมีได้ 9 ทัพ ในหนึ่งทัพ จำกัดไว้ทัพละ 12 นาย

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

3. การเลือกทัพเพื่อรับคำสั่ง เพียงกดแป้นคีย์เลขที่เคยกำหนดไว้ เช่น หากเรากำหนดไว้เป็นทัพเบอร์ 1 เราเพียงกดแป้นคีย์เลข 1 ทหารนายที่ถูกเลือกไว้ในทัพที่ 1 จะขานรับทราบ และเราสามารถสั่งการได้เหมือนปกติ

การสั่งการกองทัพและยูนิต

คำสั่งหลักๆที่ใช้สำหรับสั่งยูนิตให้ทำตามมีดังนี้

1. คำสั่งเคลื่อนที่ ทำได้โดยการคลิกเมาส์ปุ่มขวาไปยังตำแหน่งที่ต้องการ
2. คำสั่งโจมตี ทำได้โดยการคลิกเมาส์ปุ่มขวาไปยังยูนิตศัตรูหรือสิ่งก่อสร้างศัตรูที่ต้องการโจมตี โดยเมื่อคลิกไปแล้ว เคอร์เซอร์เมาส์จะเปลี่ยนเป็นสีแดง
3. คำสั่งหยุด คำสั่งนี้ทหารยูนิตนั้นๆจะปฏิบัติเองเมื่อทำคำสั่งอื่นๆเสร็จ โดยการหยุดนิ่งอยู่กับที่นี้ เมื่อศัตรูเข้ามาอยู่ในระยะโจมตี จะทำการโจมตี และติดตาม หากไม่ต้องการติดตามให้คลิกเมาส์ปุ่มขวาไปยังจุดอื่น

เงื่อนไขการจบเกม

ผู้ชนะคือผู้ที่สามารถทำลายสิ่งก่อสร้างของฝ่ายศัตรูได้หมดสิ้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บรรณานุกรม

LaMothe , A . 1998. **Windows Game Programming For Dummies.** IDG Books Worldwide, Inc.

Adams , J. **Introduction to Isometric Engine.** [online]. Available :

http://www.gamedev.net

Adams, J. **How to get smooth scrolling map.** [online]. Available :

http://www.gamedev.net

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้