

การพัฒนาโปรแกรม แสดงภาพสเตอริโอสลับเชิงเวลา
ภายใต้ระบบปฏิบัติการไมโครซอฟต์วินโดวส์

The Development of a Time-Multiplexed Stereoscopic Image
Display Program under the Microsoft Windows



วัน เดือน ปี.....	09 ส.ค. 2550
เลขทะเบียน.....	01781
เลขเรียกหนังสือ.....	กบ 251ก 2543
"ห้องสมุดคณะเทคโนโลยีสารสนเทศ สจล."	

รายงานนี้เป็นส่วนหนึ่งของวิชาโครงการพัฒนาระบบงาน
หลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
ภาคเรียนที่ 2 ปีการศึกษา 2543
คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ชื่อหัวข้อ	การพัฒนาโปรแกรม แสดงภาพสเตอริโอสลับเชิงเวลา ภายใต้ ระบบปฏิบัติการไมโครซอฟต์วินโดวส์
นักศึกษา	นายณริศ อธิธาว์ชกุล
อาจารย์ที่ปรึกษา	ดร.นพพร โชติกกำธร
ระดับการศึกษา	วิทยาศาสตรมหาบัณฑิต สาขาวิชาเทคโนโลยีสารสนเทศ
แขนงวิชา	วิทยาการสารสนเทศ
ปีการศึกษา	2543

บทคัดย่อ

โครงงานฉบับนี้นำเสนอการออกแบบและการพัฒนาโปรแกรมสำหรับใช้ในการแสดงภาพสามมิติ ที่ทำงานภายใต้ระบบปฏิบัติการไมโครซอฟต์วินโดวส์ การแสดงภาพสามมิตินี้ใช้เทคนิคในการแสดงภาพคู่สลับเชิงเวลา ซึ่งแสดงภาพฉายสำหรับตาซ้าย และภาพฉายสำหรับตาขวาสลับไปมาบนจอภาพ ให้สัมพันธ์กับการเปิดและปิดอุปกรณ์บังแสงตาซ้ายและตาขวาที่ทำด้วยผลึกเหลวที่ผู้มองภาพสวมอยู่ ทำให้ผู้มองภาพรับรู้ความลึกในระดับต่าง ๆ ตามความแตกต่างของภาพฉายที่แสดงอยู่

โปรแกรมที่พัฒนาจะพัฒนาโดยผ่าน คลังโปรแกรมของ Microsoft DirectX Version 7 ซึ่งเป็นโปรแกรมที่ใช้ติดต่อกับฮาร์ดแวร์ในเครื่อง และคลังโปรแกรม WINx3D ที่เป็นโปรแกรมที่ใช้ติดต่อกับอุปกรณ์แว่น 3 มิติ โดยที่ส่วนติดต่อกับผู้ใช้งานอยู่ภายในโปรแกรมจะอยู่ในโหมด 3 มิติ

Title	The Development of a Time-Multiplexed Stereoscopic Image Display Program under the Microsoft Windows
Student	Mr.Narit Ittiarwatchakul
Advisor	Dr.Nopporn Chotikakamthron
Level of Student	Master of Science in Information Technology
Major	Information Science
Academic Year	2000

Abstract

This project presents the design and development of a 3D image display program for used under Microsoft Windows operating system. Time multiplexed stereoscopic is used for displaying a 3D image by alternating the left and right eye projection image on the screen in synchronization with the liquid crystal display shutter glasses. As a result, the viewer wearing the shutter glasses can perceive image depth from the disparity of the left and right images.

This program is using Microsoft DirectX 7 library to control hardware in computer and using WINx3D library to control the shutter glasses. Some interface of this application is in 3D mode.

กิตติกรรมประกาศ

กราบขอบพระคุณบุพการี, อาม่า, อาง และผู้อุปการะเลี้ยงดูข้าพเจ้าให้เติบโตมาจนพบกับความสำเร็จจนทุกวันนี้

ขอขอบพระคุณ อาจารย์ ดร.นพพร โชติกคำธร ที่ให้คำแนะนำ เป็นที่ปรึกษา และเอาใจใส่ ทำให้งานสำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณ พี่น้องทั้ง 4 คนของข้าพเจ้าที่ให้ความช่วยเหลือในทุก ๆ สิ่ง และดักเตือนเสมอมา

ขอขอบพระคุณ คุณชาญณรงค์ และ คุณสันติจาก นิตยสาร Com Gamer ที่ให้ความอนุเคราะห์ในการยืมแว่น 3 มิติในการพัฒนาโครงการนี้จนสำเร็จ

ขอขอบพระคุณ พี่เจียบ, พี่ก้อง, พี่ปรีชา ที่ให้ยืมหนังสือที่ใช้ในการพัฒนาโปรแกรม

ขอขอบพระคุณ พี่โอ้ หัวหน้าที่มที่ให้ยืม Hardisk ไว้สำหรับเก็บข้อมูลที่ใช้ในการพัฒนา

ขอขอบพระคุณ พี่ป้าน, พี่เบ็น, พี่จิม, พี่แก้ว, พี่อ็ค, พี่โอ ที่ช่วยให้กำลังใจต่าง ๆ ในการทำโครงการเสมอมา

ขอขอบพระคุณ เพื่อน ๆ ทุกคนที่เรียนมาด้วยกัน ช่วยเหลือกันทำให้ประสบความสำเร็จทุกคน

นายนิธิต อธิธิอาวัชกุล

นักศึกษาปริญญาโท สาขาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีสารสนเทศ

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....	I
บทคัดย่อภาษาอังกฤษ.....	II
กิตติกรรมประกาศ.....	III
สารบัญ.....	IV
สารบัญตาราง.....	VI
สารบัญภาพ.....	VII
บทที่	
1. บทนำ	
1.1 ความเป็นมาและความสำคัญของปัญหา.....	1
1.2 วัตถุประสงค์.....	1
1.3 ขอบเขตของการพัฒนาโครงการ.....	2
1.4 ขั้นตอนของการพัฒนา.....	2
1.5 องค์ประกอบและเครื่องมือที่ใช้ในการพัฒนาโครงการ.....	3
1.6 ผลที่คาดว่าจะได้รับ.....	3
2. ระบบการมองเห็นของมนุษย์	
2.1 การรับรู้ความลึกของมนุษย์.....	4
2.2 การแสดงภาพ 3 มิติ.....	11
3. การสร้างภาพ 3 มิติสเตอริโอ	
3.1 การฉายภาพ.....	14
3.2 บริเวณมุมมอง.....	16
3.3 การเหลื่อมในแนวนอน.....	18
3.4 ความสัมพันธ์ของการเหลื่อมในแนวนอนกับมุมมอง.....	19
3.5 การหาค่าที่เหมาะสมของมุมมองในแนวนอน.....	20
3.6 ปัจจัยที่มีผลกับคุณภาพของภาพ.....	20

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญ (ต่อ)

หน้า

บทที่

4.	การพัฒนาโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์ด้วยไมโครซอฟต์ DirectX	
4.1	ความเป็นมาของไมโครซอฟต์ DirectX.....	22
4.2	หลักการของ DirectX	23
4.3	ส่วนประกอบของ DirectX	24
4.4	การเขียนโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์.....	26
4.5	Event Driven Programming	31
4.6	วิธีการแสดงภาพ.....	32
5.	การพัฒนาโปรแกรมแสดงภาพ 3 มิติ ด้วย WINx3D	
5.1	บทนำ.....	49
5.2	ความต้องการของเครื่องที่ใช้ในการพัฒนาโปรแกรมด้วย WINx3D	49
5.3	สถาปัตยกรรมของ WINx3D	49
5.4	ส่วนประกอบภายในคลังโปรแกรม WINx3D	51
6.	ขั้นตอนการพัฒนาโปรแกรม	
6.1	การพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 2 มิติ	52
6.2	การพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 3 มิติ	58
7.	ผลการพัฒนาโปรแกรม	
7.1	ภาพรวมของการผลพัฒนาโปรแกรม.....	61
7.2	ผลการพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 2 มิติ	61
7.3	ผลการพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 3 มิติ	68
8.	สรุปผลการค้นคว้าและการพัฒนาโปรแกรม	
8.1	ผลการดำเนินการพัฒนาโปรแกรม	71
8.2	ข้อเสนอแนะ.....	71
	บรรณานุกรม	72
	ประวัติผู้เขียน.....	73

สารบัญตาราง

หน้า

ตารางที่

4.1 ค่า Cooperative level ของ DirectDraw	35
4.2 Surface Capabilities Flags	42
4.3 ค่า Control flags ของ Blt()	45



สารบัญภาพ

หน้า

ภาพที่

2.1	Linear Perspective.....	4
2.2	Light and Shade	5
2.3	Aerial Perspective	5
2.4	Interposition	6
2.5	Texture Gradient	6
2.6	Relative Size	7
2.7	Depth Cuing.....	7
2.8	Binocular Disparity.....	8
2.9	Motion Parallax.....	9
2.10	Zero Parallax.....	9
2.11	Positive Parallax	10
2.12	Negative Parallax.....	10
2.13	Divergent Parallax	10
2.14	View-Master stereoscope.....	13
3.1	การฉายภาพ.....	14
3.2	การฉายภาพนอกแกน.....	15
3.3	บริเวณของมุมมอง สำหรับการฉายภาพนอกแกน	17
3.4	บริเวณของมุมมอง สำหรับการฉายภาพบนแกน	17
3.5	แสดงมุมมองการมองในแนวนอน	19
4.1	แสดงการเปรียบเทียบ Application ที่ใช้ GDI กับ DirectX.....	23
4.2	ส่วนประกอบของ DirectX.....	26
4.3	ระบบการส่ง message ใน Windows	27
4.4	The main event loop.....	30
4.5	เครือข่ายของ Direct Draw	32
4.6	การแสดงผลภาพโดยใช้ Direct Draw	39

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สารบัญภาพ (ต่อ)

หน้า

ภาพที่

5.1	สถาปัตยกรรมของ WINx3D.....	50
6.1	แสดง Layout File Stereoscopic แบบ Side-by-Side.....	54
6.2	แสดง Layout File Stereoscopic แบบ Over-Under	55
6.3	แสดง Layout File Stereoscopic แบบ Interlaced	55
6.4	แสดง ขั้นตอนการสร้างปุ่มกด 3 มิติ.....	59
6.5	แสดง ปุ่มเมื่อถูกกด กับ กรอบภาพแบบนูน	60
7.1	หน้าจอหลัก.....	62
7.2	เมนูการเปิดเพิ่มข้อมูล	62
7.3	Dialog box การเปิดเพิ่มข้อมูลภาพในมุมมองของตาซ้ายและขวา.....	63
7.4	Dialog box การเปิดเพิ่มข้อมูลภาพในมุมมองของตาซ้าย.....	64
7.5	Dialog box การเปิดเพิ่มข้อมูลภาพในมุมมองของตาขวา.....	64
7.6	เพิ่มข้อมูลที่สนับสนุน	66
7.7	หน้าจอแสดง Layout ของภาพ	68
7.8	หน้าจอที่แสดงในโหมด 3 มิติ	69
7.9	Mouse Cursor ในการขยายภาพ.....	69
7.10	Mouse Cursor ในการย่อภาพ	70

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

การแสดงผลภาพด้วยเครื่องคอมพิวเตอร์ได้กลายเป็นส่วนหนึ่งของการประยุกต์การใช้งานเครื่องคอมพิวเตอร์ในทุกแขนงงาน คอมพิวเตอร์กราฟิกส์ (Computer Graphics) เป็นศาสตร์ที่ได้รับความสนใจอย่างมากทั้งในอดีต ปัจจุบัน และอนาคต โดยเฉพาะอย่างยิ่งคอมพิวเตอร์กราฟิกส์แบบสามมิติ ทั้งนี้สืบเนื่องมาจากวิวัฒนาการที่เติบโตไปอย่างคล่องจองกันทางด้านการพัฒนาซอฟต์แวร์และความสามารถของฮาร์ดแวร์ทางด้านการคำนวณและการแสดงผล ทำให้การแสดงผลภาพสามมิติกระทำได้รวดเร็ว ได้ตอบกับผู้ใช้ได้ การแสดงผลภาพสามมิติ ไม่ว่าจะเป็นภาพนิ่งหรือภาพเคลื่อนไหว เป็นที่ต้องการอย่างสูง ในการประยุกต์ใช้คอมพิวเตอร์ในการแสดงผลเพื่อการจินตทัศน์ข้อมูลแบบหลายมิติ (Multidimensional Data Visualization), การแสดงผลภาพทางการแพทย์ (Medical Imaging), การใช้คอมพิวเตอร์ช่วยออกแบบ (Computer-Aided Design), การวิจัยในเรื่องของความจริงเสมือน (Virtual Reality) หรือการแสดงผลภาพสามมิติแบบมีความลึกที่แท้จริง (True 3D) ซึ่งส่วนใหญ่จะถูกสร้างขึ้นมาจากอุปกรณ์ฮาร์ดแวร์พิเศษ หรือใช้อุปกรณ์ประกอบอื่น ๆ ช่วยให้เห็นภาพ 3 มิติเป็นต้น อุปกรณ์พิเศษดังกล่าวนี้มีราคาค่อนข้างสูงและต้องการการสนับสนุนจากคอมพิวเตอร์โดยเฉพาะ ทำให้ไม่สามารถใช้งานได้กับคอมพิวเตอร์ส่วนบุคคลทั่วไป

โครงการพัฒนาลำดับนี้ จึงขอเสนอ การนำพื้นฐานการรับรู้ความลึก การแสดงผลภาพสามมิติแบบสเตอริโอบนจอแสดงผล การใช้โปรแกรม DirectX ติดต่อกับฮาร์ดแวร์บางส่วนในเครื่องคอมพิวเตอร์ และ ชุดการพัฒนาซอฟต์แวร์ (Software Development Kit) ชื่อ WINx3D มาประยุกต์ร่วมกัน เพื่อใช้เป็นแนวทางการออกแบบและพัฒนาการแสดงผลภาพสามมิติ แบบสเตอริโอ บนพีซีคอมพิวเตอร์ ที่ใช้ระบบปฏิบัติการ ไมโครซอฟต์วินโดวส์

1.2 วัตถุประสงค์

- เพื่อศึกษาพื้นฐานของการสร้างภาพสเตอริโอ และวิธีการแสดงผลสเตอริโอบนจอแสดงผล
- พัฒนาโปรแกรมประยุกต์ เพื่อใช้ในการแสดงผลภาพสเตอริโอภายใต้ ระบบปฏิบัติการ ไมโครซอฟต์วินโดวส์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.3 ขอบเขตของการพัฒนาโครงการ

การพัฒนาโครงการนี้มีขึ้นเพื่อพัฒนาโปรแกรมแสดงภาพสเตอริโอ 3 มิติแบบสลับทั้งหน้าจอเชิงเวลา (Page Flipping) โดยมีหลักการและขอบเขตเพื่อให้ครอบคลุมส่วนต่างๆ ที่สำคัญคือ

- ทำงานได้บนเครื่องคอมพิวเตอร์พีซี ที่มีระบบปฏิบัติการไมโครซอฟต์วินโดวส์ 95 ขึ้นไป มีซอฟต์แวร์ DirectX และ WINx3D
- สามารถใช้ได้กับฮาร์ดแวร์ที่เป็นแว่นที่ใช้สำหรับดูภาพสเตอริโอหลากหลายชนิดที่สามารถใช้กับซอฟต์แวร์ WINx3D ได้
- โปรแกรมที่พัฒนาขึ้นจะมีหน้าที่ต่าง ๆ ดังนี้
 - สามารถแสดงภาพซ้ายหรือขวา ที่มีนามสกุล BMP, JPG และ PNG ได้ในโหมด 2 มิติ
 - สามารถบันทึกภาพซ้ายหรือขวา ที่มีนามสกุล BMP, JPG และ PNG และภาพที่มี Layout เป็นแบบ Stereoscopic ได้
 - สามารถแสดงภาพสเตอริโอที่มีนามสกุล BMI, BMS, JPS, JPO, JPI, JPP, PNG และ PNS ได้ในโหมด 2 มิติ
 - นำภาพซ้ายและขวาที่ได้ในโหมด 2 มิติมาแสดงในโหมด 3 มิติได้
 - ส่วนติดต่อกับผู้ใช้บางอย่างในโหมด 3 มิติ จะเป็นในลักษณะ 3 มิติ
 - ย่อ / ขยาย ภาพได้ในโหมด 2 และ 3 มิติ

1.4 ขั้นตอนของการพัฒนา

- ศึกษาทฤษฎีที่เกี่ยวข้อง
 - การแสดงผลสเตอริโอ 3 มิติแบบภาพสลับเชิงเวลา
 - การคำนวณภาพฉายแบบสเตอริโอ 3 มิติ
 - วิธีการเขียนโปรแกรมโดยใช้ Microsoft Visual C++ 6
 - วิธีการเขียนโปรแกรมโดยใช้ Microsoft DirectX 7 SDK
 - วิธีการเขียนโปรแกรมโดยใช้ WINx3D SDK
- ออกแบบโปรแกรมแสดงภาพสเตอริโอสลับเชิงเวลาที่ทำงานภายใต้ ระบบปฏิบัติการไมโครซอฟต์วินโดวส์
- พัฒนาโปรแกรม
- ทดสอบและแก้ไขปรับปรุงโปรแกรม
- สรุปผลการวิจัยและข้อเสนอแนะ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

1.5 องค์ประกอบและเครื่องมือที่ใช้ในการพัฒนาโครงการ

1.5.1 องค์ประกอบทางด้านฮาร์ดแวร์

- ชุดเครื่องคอมพิวเตอร์พีซี ที่มีจอที่สามารถเปลี่ยนอัตราการวาดภาพบนจอ (Refresh Rate) ในโหมด 3 มิติได้อย่างน้อย 120 MHz เพื่อไว้ใช้ในการแสดงผลภาพสเตอริโอ สลับเชิงเวลา
- แวนหรือชุดเครื่องมือที่ใช้สำหรับดูภาพสเตอริโอ

1.5.2 องค์ประกอบทางด้านซอฟต์แวร์ (Software)

- ระบบปฏิบัติการไมโครซอฟต์วินโดวส์รุ่น 9x (Microsoft Windows 9x) หรือระบบปฏิบัติการไมโครซอฟต์วินโดวส์รุ่น ME (Microsoft Windows Millinium Edition)
- ซอฟต์แวร์ Microsoft DirectX Runtime ตั้งแต่ Version 6 ขึ้นไป
- ซอฟต์แวร์ WINx3D ใช้สำหรับติดต่ออุปกรณ์ในการดูภาพสเตอริโอ
- เครื่องมือหรือซอร์ฟแวร์ที่ใช้ในการพัฒนาโครงการ
 - Microsoft Visual C++ Version 6
 - Microsoft DirectX Version 7 SDK
 - WINx3D Version 2.5 SDK

1.6 ผลที่คาดว่าจะได้รับ

โครงการที่พัฒนาก่อให้เกิดประโยชน์ดังนี้

- เป็นโปรแกรมที่ใช้ในการแสดงภาพสเตอริโอสลับเชิงเวลาภายใต้ระบบปฏิบัติการไมโครซอฟต์วินโดวส์
- เป็นต้นแบบในการพัฒนาโปรแกรมประยุกต์ในลักษณะเช่นนี้ต่อไป
- ทำให้การพัฒนาโปรแกรมในด้านนี้สามารถพึ่งตนเองได้ในระดับหนึ่ง และประหยัดค่าใช้จ่ายในการซื้อโปรแกรมจากต่างประเทศ

บทที่ 2

ระบบการมองเห็นของมนุษย์

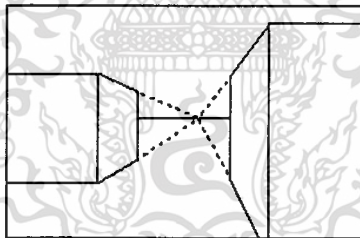
2.1 การรับรู้ความลึกของมนุษย์

ในระบบการมองเห็นของมนุษย์นั้น อาศัยการแทรกสอดทางความลึกของวัตถุในหลายระดับ เพื่อแยกตำแหน่งที่สัมพันธ์กันของวัตถุในภาพ 3 มิติ ซึ่งการแทรกสอดแบบนี้ถูกแบ่งออกเป็น 2 ประเภท คือ การแทรกสอดทางจิต (Psychological) และ การแทรกสอดทางกายภาพ (Physiological)

2.1.1 การแทรกสอดทางความลึกแบบจิตวิทยา

มีดังต่อไปนี้

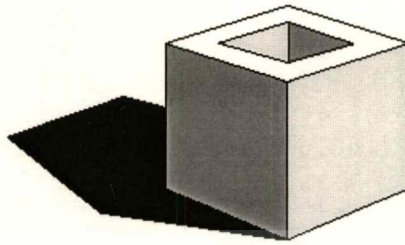
2.1.1.1 Linear Perspective



ภาพที่ 2.1 Linear Perspective

คือ คุณสมบัติของการมองเห็น ซึ่งเป็นสาเหตุให้ขนาดของภาพบนเรตินาเปลี่ยนแปลงในสัดส่วนที่สมดุล เพื่อที่จะเปลี่ยนระยะระหว่างวัตถุกับตาที่มีความสัมพันธ์กันระหว่างวัตถุด้านหน้าและด้านหลัง เช่น เมื่อเวลามองถนนที่เป็นเส้นตรงขนานกันไปจะพบว่า ถนนจะบรรจบกันที่ปลายขอบฟ้า ซึ่งจะพบเห็นได้บ่อย ๆ ในภาพถ่าย สัดส่วนที่ปรากฏต่อสายตาจะลดลงเรื่อย ๆ เมื่อระยะทางเพิ่มขึ้น

2.1.1.2 Light and Shade



ภาพที่ 2.2 Light and Shade

คือ คุณสมบัติของแสงที่สำคัญในภาพ ที่จะช่วยให้เราหารูปร่างและความลึกที่สัมพันธ์กันของวัตถุ โดยตำแหน่งและแหล่งกำเนิดแสง เมื่อเราทราบตำแหน่งที่ตั้งของแสง และเห็นเงาของวัตถุที่ตกกระทบบนวัตถุอีกชิ้นหนึ่ง เราก็จะทราบว่าวัตถุที่ให้เงาอยู่ใกล้แหล่งกำเนิดแสงมากกว่า และเวลาที่เรามองเห็นวัตถุโดยทั่วไปก็จะพบว่า วัตถุที่สว่างกว่าจะอยู่ใกล้ผู้สังเกตมากกว่า

2.1.1.3 Aerial Perspective

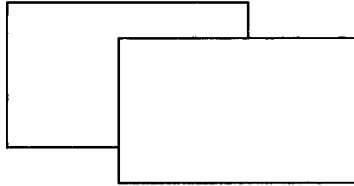


ภาพที่ 2.3 Aerial Perspective

คือ คุณสมบัติซึ่งเป็นสาเหตุให้วัตถุที่อยู่ห่างไกลออกไปเห็นได้ไม่ชัดเจนและมีสีมืดมัวลง เช่น ภาพของภูเขาเมื่อเวลามองเห็นในแนวระดับ มักจะเป็นสีฟ้าหม่น ๆ สาเหตุก็เนื่องมาจากมีละอองน้ำเล็ก ๆ และอนุภาคของฝุ่นในอากาศ อยู่ระหว่างตาและภูเขา ถ้าภูเขาอยู่ไกลออกไปเท่าไรก็ยิ่งเห็นภูเขาเป็นสีหม่นมากขึ้น สีที่เห็นก็เป็นสิ่งที่บอกความใกล้ไกล หรือสีของน้ำทะเล ถ้าเห็นเป็นสีน้ำเงินแสดงว่ามีความลึก แต่ถ้าเป็นสีดำแสดงว่าลึกมาก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

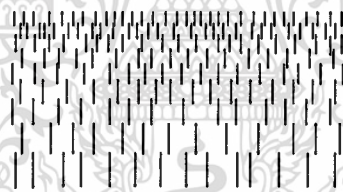
2.1.1.4 Interposition



ภาพที่ 2.4 Interposition

จะเกิดขึ้นเมื่อวัตถุนั้นถูกปิดบัง หรือทับคาบเกี่ยวกับอันอื่น ซึ่งทำให้เราสรุปได้ว่า วัตถุนั้นอยู่ใกล้กว่า โดยอาศัยขนาดของภาพที่เกิดขึ้นบนเรตินา และความรู้ที่เรามีอยู่กับขนาดของวัตถุจริง

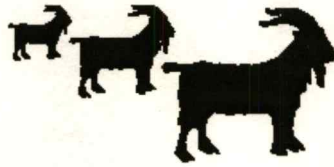
2.1.1.5 Texture Gradient



ภาพที่ 2.5 Texture Gradient

คือ ลักษณะของพื้นผิวที่ช่วยให้เราหาความลึกได้ โดยจำนวนของรายละเอียดที่สามารถมองเห็นได้บนวัตถุ กล่าวคือ เมื่อเรามองใกล้ ๆ วัตถุ ยิ่งใกล้วัตถุเท่าไรก็ยิ่งเห็นรายละเอียดพื้นผิวของวัตถุนั้นมาก ดังนั้น วัตถุที่มีลักษณะเรียบจะให้ความรู้สึกที่วัตถุอยู่ไกลออกไป ซึ่งพื้นผิวที่มีลักษณะเหมือนกันหมดก็จะให้ความรู้สึกเหมือนกัน

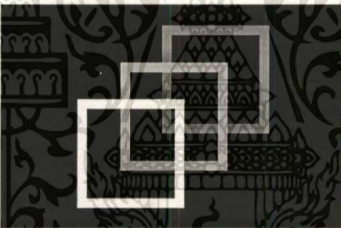
2.1.1.6 Relative Size



ภาพที่ 2.6 Relative Size

คือ การเปรียบเทียบความสัมพันธ์ของขนาดของวัตถุ ซึ่งถ้าเราทราบขนาดที่แท้จริงของวัตถุ สมองเราก็จะเปรียบเทียบขนาดที่รับรู้ได้กับขนาดของวัตถุจริง เราก็จะทราบได้ว่า วัตถุนั้นอยู่ใกล้หรือไกล

2.1.1.7 Depth Cuing



ภาพที่ 2.7 Depth Cuing

คือ ชื่อของเทคนิคที่ถูกใช้ใน Computer Graphic ซึ่งจะลดขนาดของความเข้มของวัตถุให้เป็นสัดส่วนกับระยะทางจากผู้มอง

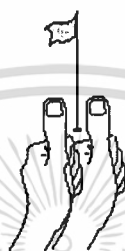
2.1.2 การแทรกสอดความลึกทางกายภาพ

มีดังต่อไปนี้

2.1.2.1 Accommodation/Convergence คือ การปรับเปลี่ยนความยาวของจุดโฟกัสของเลนส์ตาที่โฟกัสอยู่ที่ภาพ และการหมุนลูกตาเข้าสู่ข้างในให้ลู่เข้าสู่วัตถุหรือภาพที่มองด้วยตาทั้งสองข้างเป็นจุดเดียว เมื่อเรามองวัตถุทั่วไป ตาของเราต้องโฟกัสและลู่เข้าไปยังวัตถุที่มอง ทำให้เห็นวัตถุได้ชัดเจนจากนั้นสมองก็จะทำหน้าที่รวมภาพที่ได้จากตาทั้งสองข้างนั้นเป็นภาพ 3 มิติ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

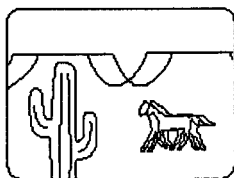
2.1.2.2 Binocular Disparity



ภาพที่ 2.8 Binocular Disparity

คือ ความแตกต่างของตาทั้งสองข้างเมื่อมีการมองภาพ 3 มิติ ถ้าเรานำภาพมาวางไว้ตรงกับตาข้างซ้ายและขวา เราเกือบจะเห็นภาพที่ซ้อนกัน 1 ภาพ โดยเป็นภาพที่มองจากทางด้านซ้ายและขวา ซึ่งนักสรีรวิทยาจะเรียกสิ่งนี้ว่า Disparity, Disparity คือ ระยะทางที่มีทิศทางในแนวนอนระหว่างจุดของรูปภาพข้างซ้ายและขวาที่ตรงกัน ของการซ้อนกันในการมองภาพ จุดที่ตรงกันของการมองภาพวัตถุของนัยน์ตาจะเบนเข้าหากัน ทำให้เกิดลักษณะที่เรียกว่า Zero Disparity ส่วน Retinal Disparity มีสาเหตุจากข้อเท็จจริงที่ว่า ในตาแต่ละข้างของเราจะมองเห็นสิ่งต่าง ๆ จากจุดที่แตกต่างกัน ซึ่งโดยทั่ว ๆ ไปตาของเราทั้งสองข้างจะห่างกันประมาณ 2 นิ้วครึ่ง หรือ 64 มิลลิเมตร Disparity จะถูก Eye-Brain ดำเนินการให้เป็นภาพเดี่ยวเหมือนกับที่เห็นในโลก ซึ่งเป็นความสามารถทางสมองของมนุษย์ในการรวมสองความแตกต่างของภาพที่คล้ายกัน ให้เป็นภาพเดี่ยว ซึ่งเรียกว่า Fusion หรือการหลอมรวมภาพ ซึ่งผลของความรู้สึกถึงความลึกนี้เรียกว่า Stereopsis หรือ ภาพ 3 มิติ

2.1.2.3 Motion Parallax

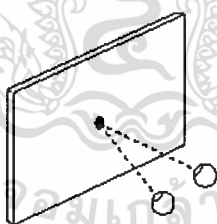


ภาพที่ 2.9 Motion Parallax

คือ การใช้ความแตกต่างในการมองภาพ โดยการเคลื่อนไหวกภาพหรือผู้ดูภาพนั้น ซึ่งวัตถุที่อยู่ใกล้กว่าจะดูเหมือนว่าเคลื่อนไหวกได้เร็วกว่า เช่น การที่เรามองเห็นเครื่องบินที่บินอยู่สูงมาก ๆ จะดูเคลื่อนที่ช้า ซึ่งแสดงว่าอยู่ไกลมาก

Parallax และ Disparity มีรูปธรรมที่คล้ายกัน Parallax วัดที่จอแสดงภาพ และ Disparity วัดที่นัยน์ตา ซึ่งทั้งสองก็คือความแตกต่างของภาพจากจุดสังเกตสองจุดทางแนวนอนนั่นเอง Parallax ถูกแบ่งออกเป็น 4 แบบด้วยกัน คือ

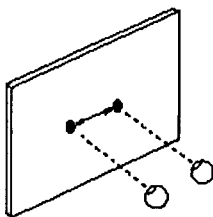
2.1.2.3.1 Zero Parallax



ภาพที่ 2.10 Zero Parallax

ตามรูปแสดงให้เห็นถึงตาของผู้สังเกตมองไปยังพื้นรูป หรือจอแสดงภาพเป็นจุดเดียว โดยที่ตาของเราจะลู่เข้าไปสู่พื้นจอแสดงภาพ

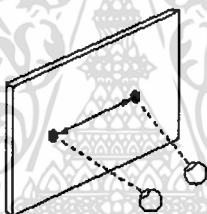
2.1.2.3.2 Positive Parallax



ภาพที่ 2.11 Positive Parallax

หรือเรียกอีกอย่างหนึ่งว่า Uncross คือการมองของตาทั้งสองไปอย่างขนานกัน ซึ่งจะเกิดขึ้นในกรณีที่เรามองวัตถุที่มีระยะห่างจากเรามาก ๆ

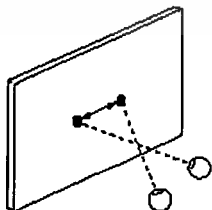
2.1.2.3.3 Negative Parallax



ภาพที่ 2.12 Negative Parallax

เกิดขึ้นเมื่อระยะห่างของภาพมากกว่าระยะห่างของตา และตาเราจะดูออก (ถ่างออก)

2.1.2.3.4 Divergent Parallax



ภาพที่ 2.13 Divergent Parallax

เกิดขึ้นเมื่อแนวของสายตาคัดกันก่อนถึงจอแสดงภาพ ดังนั้นภาพวัตถุที่เห็นจะอยู่ใกล้กว่าจอแสดงภาพ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

2.2 การแสดงภาพ 3 มิติ

ในช่วงหลายปีที่ผ่านมา เทคนิคและเทคโนโลยีทางการแสดงภาพสามมิติเกิดขึ้นใหม่มากมาย มีการพัฒนาเทคโนโลยีใหม่ ๆ ในการแสดงผล อาทิเช่น หน่วยปิดเปิดบังคับแสงทางเดียวแบบผลึกเหลว (Liquid-crystal Polarizing Shutter) ที่ใช้ในโรงภาพยนตร์ หรือในงานแสดงต่าง ๆ เป็นต้น อุปกรณ์แสดงภาพสามมิติพอแบ่งออกได้เป็น 3 ลักษณะคือ แบบโฮโลกราฟิกส์ (Holographics) แบบหลายระนาบ (Multi-planar) และ แบบภาพคู่สเตอริโอ (Stereo pair)

2.2.1 โฮโลแกรม

โฮโลแกรม (Hologram) เป็นการแสดงภาพสามมิติที่เป็นที่คุ้นเคยกันโดยทั่วไป ตัวอย่างเช่น รูปที่ปรากฏอยู่บนบัตรเครดิตทั่วไป ในการสร้างภาพโฮโลแกรม (Hologram) นั้น ผู้สร้างจะทำการบันทึกรูปแบบการแทรกซ้อนที่เกิดจากลำแสงเลเซอร์สองลำ ที่ยิงจากแหล่งเดียวกันลงบนสื่อบันทึกภาพความละเอียดสูง ลำแสงหนึ่งยิงตรงเข้าหาสื่อบันทึกภาพ ในขณะที่อีกลำแสงหนึ่งสะท้อนจากวัตถุและเกิดการแทรกซ้อนกับลำแสงอ้างอิงลำแรก รูปแบบการแทรกซ้อนของลำแสงนี้เอง ที่เอื้ออำนวยการสร้างภาพสามมิติของวัตถุนั้นในภายหลังได้ ผู้มองสามารถโยกย้ายมุมมอง เพื่อเปลี่ยนมุมมองการเห็นภาพสามมิติของวัตถุได้ อย่างไรก็ตาม เทคโนโลยีทางด้านโฮโลแกรมยังก้าวไปไม่ถึงจุดที่เอื้ออำนวยการแสดงภาพแบบทันทีได้ เนื่องจากต้องการความเร็วของการประมวลผลที่สูง

2.2.2 หลายระนาบ

การแสดงภาพแบบหลายระนาบนั้น จะใช้การแบ่งวัตถุออกเป็นร่อย ๆ หรืออาจเป็นพื้น ๆ ระนาบ จากนั้นทำการวาดจุดต่าง ๆ ของวัตถุในแต่ละระนาบ โดยทั่วไประบบการแสดงภาพแบบนี้จะใช้กระจกแปรจุดรวม (Verifocal Mirror Technique) ทำการแบ่งภาพสามมิติออกเป็นหลายพื้นระนาบ จากนั้นวาดจุดต่าง ๆ บนแต่ละระนาบด้วยหลอดภาพแล้วใช้กระจกโค้งที่สั่นด้วยความถี่ 30 Hz สะท้อนภาพจากหลอดภาพเข้าสู่ตาผู้มอง ตัวรับภาพในตามนุษย์จะจดจำภาพที่มองเห็นในช่วงเวลาต่าง ๆ (Temporal Persistent) ดังนั้นภาพที่เห็นจากกระจกที่เลื่อนได้ (อย่างรวดเร็ว) จะทำให้โสตประสาทการมองเห็นของมนุษย์รวมภาพต่าง ๆ (ในแต่ละระนาบ) เข้าด้วยกันจนเกิดการรับรู้สามมิติที่มีปริมาตร (Volume Image) ภาพสามมิติที่แสดงด้วยวิธีนี้จะเป็นภาพที่โปร่งใสและมีเพียงสี่เดียว

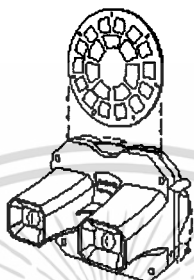
2.2.3 ภาพคู่สเตอริโอ

เทคนิคการแสดงผลภาพคู่สเตอริโอคือการแสดงผลภาพที่เหมาะสมสำหรับตาทั้งสอง นั่นคือแสดงภาพ ๑ หนึ่งสำหรับตาซ้าย และอีกภาพหนึ่งสำหรับตาขวา ถ้าภาพทั้งสองมีความแตกต่างกันของระยะวัตถุตามแนวนอน จะทำให้ผู้มองรับรู้ความลึกของวัตถุจากความแตกต่างของระยะวัตถุดังกล่าว เทคนิควิธีการแสดงผลภาพทั้งสองนี้ แบ่งออกได้เป็น การแสดงผลภาพคู่พร้อมกัน (Time-parallel) และการแสดงผลภาพคู่สลับเชิงเวลา (Time-multiplexed)

การแสดงผลภาพคู่พร้อมกันที่เป็นที่รู้จักกันมากคือ กล้องดูรูปภาพนิ่งการ์ตูนสามมิติ (View-Master stereoscope ดังภาพที่ 2.14) ผู้มองมองผ่านช่องมองสองช่อง แต่ละช่องจะมีภาพสองภาพสำหรับตาแต่ละข้าง อีกวิธีหนึ่งที่เป็นที่รู้จักคือ การแสดงผลภาพยนต์สามมิติในสมัยก่อน ซึ่งใช้วิธีที่เรียกว่า Anaglyph ที่แสดงผลภาพสำหรับตาทั้งสองบนจอภาพพร้อม ๆ กัน โดยใช้สีที่ต่างกันสำหรับภาพทั้งสอง เช่นสีแดงกับสีน้ำเงิน เป็นต้น โดยที่ผู้มองจะต้องสวมแว่นที่มีตัวกรองสีแดงและสีน้ำเงิน วิธีนี้มีปัญหาเรื่องภาพหลอน เนื่องจากตัวกรองแสงไม่สามารถกรองแสงได้หมดจริง จึงทำให้เกิดการรับภาพสำหรับตาอีกข้างหนึ่งด้วยการแสดงผลภาพยนต์สามมิติแบบ Anaglyph นี้ได้รับการปรับปรุงให้ดีขึ้น ด้วยการฉายภาพจากกล้องฉายสองกล้องผ่านแผ่นกรองบังคับทิศทางที่ต่างกันของคลื่นแสง โดยที่ทิศทางคลื่นของการฉายภาพทั้งสองจะตั้งฉากซึ่งกันและกัน แว่นกรองทิศทางที่ผู้ชมสวมจะกรองทิศทางของคลื่นที่ต่าง ๆ สำหรับตาทั้งสอง ดังนั้นแผ่นกรองจะกันภาพที่ไม่ตรงทิศออกได้

การแสดงผลภาพคู่สลับเชิงเวลา มีหลักการทำงานคือการแสดงผลภาพสำหรับตาซ้ายในขณะที่เปิดตาซ้ายแต่ปิดตาขวา สลับกับการแสดงผลภาพตาขวาในขณะที่เปิดตาขวาแต่ปิดตาซ้าย หากการสลับดังกล่าวกระทำได้ในระยะเวลาอันสั้น โสตประสาทการรับรู้จะสามารถรวมภาพทั้งสองและรับรู้ความลึกได้ การเปิด-ปิดตากระทำได้ในสองลักษณะคือ การใช้แว่นแบบ Active กับแบบ Passive สำหรับแบบ Active นั้นตัวแว่นเองจะทำการเปิดหรือบังการมองของนัยน์ตา คอยการใช้แว่นที่ทำจากผลึกเหลวที่สามารถกั้นหรือยอมให้แสงผ่านตามลักษณะสัญญาณที่ป้อนเข้าตัวแว่น การเปิดหรือปิดแว่นนี้จะกระทำไปแบบประสานจังหวะไปกับการแสดงผลภาพตาซ้ายและขวาสลับกันบนจอภาพ สำหรับแว่นแบบ Passive นั้นการเปิดปิดภาพจะกระทำที่หน้าจอโดยที่ฉากหน้าจอภาพจะสามารถควบคุมทิศทางคลื่นแสงได้ โดยให้ภาพตาซ้ายและภาพตาขวาถูกแสดงสลับกันไป โดยมีคลื่นแสงที่ถูกบังคับทิศให้ตั้งฉากซึ่งกันและกัน ผู้ชมก็เพียงสวมแว่นกรองทิศคลื่น ในลักษณะเดียวกันกับการชมภาพยนต์สามมิติ

เทคนิคการใช้ภาพคู่สเตอริโอนี้จะมีราคาย่อมเยาที่สุด วิธีนี้สามารถใช้ภาพสีเหมือนจริงที่สร้างขึ้นด้วยเทคนิคทางคอมพิวเตอร์อื่น ๆ ได้ แต่ส่วนเสียก็คือจะมีผู้มองอยู่จำนวนหนึ่งที่ไม่รับรู้ความมีมิติของภาพที่สร้างขึ้นโดยใช้เทคนิคนี้ เรียกว่า บอดภาพคู่สเตอริโอ (Stereo Blindness)



ภาพที่ 2.14 View-Master stereoscope

บทที่ 3

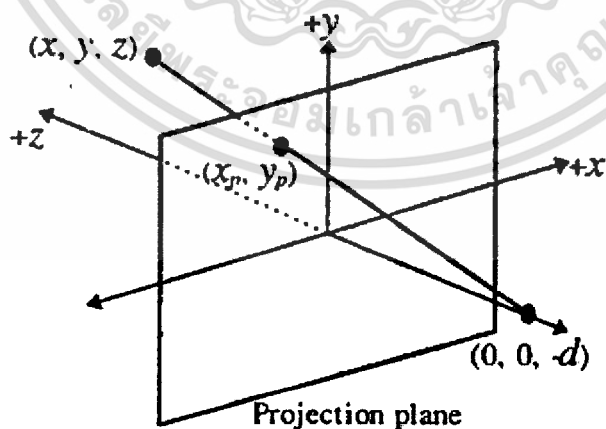
การสร้างภาพ 3 มิติสเตอริโอ

ในระบบการแสดงผลภาพสเตอริโอสามมิตินั้น จะต้องสร้างภาพสองภาพสำหรับตาแต่ละข้าง ภาพทั้งสองเกิดจากการคำนวณภาพฉาย (Projection) ของวัตถุบนระนาบการฉาย ซึ่งสามารถคำนวณได้ในสองลักษณะ คือ การฉายภาพนอกแกน (Off-axis Projection) และ การฉายบนแกน (On-axis Projection) ถึงแม้ว่าวิธีทั้งสองจะเหมือนกันในทางคณิตศาสตร์ก็ตาม แต่ในทางปฏิบัติแล้วจะได้ภาพที่แตกต่าง ดังจะได้นำเสนอต่อไป

3.1 การฉายภาพ

กำหนดให้พิกัดของจุด p คือ (x, y, z) การฉายจุด p ลงบนระนาบ $(x, y, 0)$ โดยมีจุดศูนย์กลางการฉาย (center of projection) อยู่ที่ $(0, 0, -d)$ ดังแสดงในภาพที่ 3.1 จะได้จุดฉาย (x_p, y_p) บนระนาบ ดังนี้

$$x_p = \frac{xd}{d+z} \quad y_p = \frac{yd}{d+z} \quad (1)$$



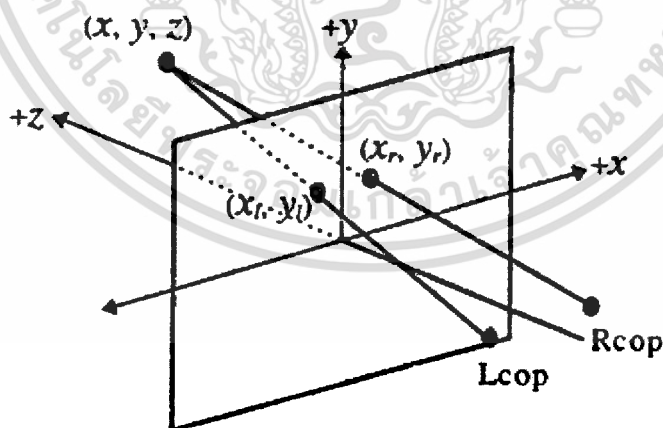
ภาพที่ 3.1 การฉายภาพ

3.1.1 การฉายภาพนอกแกน

การฉายภาพนอกแกนใช้จุดศูนย์กลางการฉายสองจุด คือจุดศูนย์กลางการฉายด้านขวา (Rcop) และจุดศูนย์กลางการฉายด้านซ้าย (Lcop) จุดสองจุดนี้อยู่บนแกน z เป็นระยะทาง $e/2$ นั่นคือพิกัดของ Rcop และ Lcop เป็น $(e/2, 0, -d)$ และ $(-e/2, 0, -d)$ ตามลำดับ โดยที่ e คือระยะห่างระหว่างจุดทั้งสอง (ระยะห่างนี้มีผลต่อการแสดงภาพแบบสเตอริโอซึ่งจะได้กล่าวไว้ในภายหลัง) เมื่อใช้ Rcop เป็นจุดศูนย์กลางการฉายจุดบนระนาบสำหรับภาพตาขวา และใช้ Lcop เป็นจุดศูนย์กลางการฉายจุดบนระนาบสำหรับภาพตาซ้าย จะได้ภาพคู่ของวัตถุบนระนาบสำหรับตาทั้งสองเพื่อการรับรู้ภาพสามมิติแบบสเตอริโอ สมมติให้ p เป็นจุดบนวัตถุ มีพิกัด (x, y, z) การฉายภาพนอกแกนของจุด p จะได้จุดสองจุดคือ จุด (x_r, y_r) และจุด (x_l, y_l) บนระนาบ ซึ่งได้จากการฉายโดยใช้จุด Rcop และ Lcop เป็นจุดศูนย์กลางการฉายตามลำดับดังนี้ (ดูภาพที่ 3.2 ประกอบ)

$$x_r = \frac{xd - \frac{ze}{2}}{d + z} \quad y_r = \frac{yd}{d + z} \quad (2)$$

$$x_l = \frac{xd + \frac{ze}{2}}{d + z} \quad y_l = \frac{yd}{d + z} \quad (3)$$



ภาพที่ 3.2 การฉายภาพนอกแกน

3.1.2 การฉายภาพบนแกน

สมการการฉายภาพนอกแกน ที่เสนอในหัวข้อ 3.1.1 สามารถเขียนได้ใหม่ ในรูปแบบดังต่อไปนี้

$$x_l = \frac{xd - \frac{ze}{2}}{d+z} = \frac{d\left(x + \frac{e}{2}\right)}{d+z} - \frac{e}{2} \quad (4)$$

$$x_r = \frac{xd + \frac{ze}{2}}{d+z} = \frac{d\left(x - \frac{e}{2}\right)}{d+z} + \frac{e}{2} \quad (5)$$

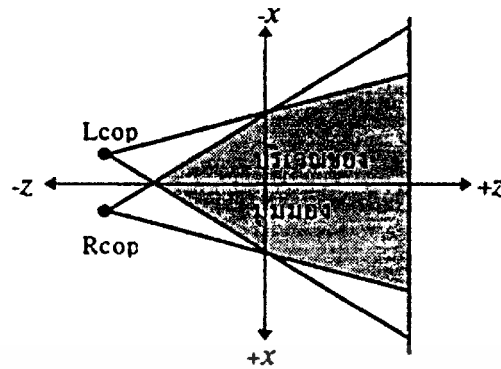
ซึ่งสามารถตีความการฉายภาพบนระนาบได้ในอีกหนึ่งลักษณะดังนี้

ขั้นตอนการฉายภาพบนแกน	ภาพตาขวา	ภาพตาซ้าย
1. เริ่มด้วยการย้ายจุด (x, y, z) ตามแนวนอนไป $e/2$ (หรือ $-e/2$)	$x - \frac{e}{2}$	$x + \frac{e}{2}$
2. จากนั้นฉายจุดที่ย้ายตำแหน่งแล้วลงบนระนาบ โดยใช้ (0, 0, -d) เป็นจุดศูนย์กลางการฉาย (ซึ่งอยู่บนแกนความลึก)	$\frac{d\left(x - \frac{e}{2}\right)}{d+z}$	$\frac{d\left(x + \frac{e}{2}\right)}{d+z}$
3. แล้วจึงย้ายจุดฉายบนระนาบกลับตามแนวนอนไป $-e/2$ (หรือ $e/2$)	$\frac{d\left(x - \frac{e}{2}\right)}{d+z} + \frac{e}{2}$	$\frac{d\left(x + \frac{e}{2}\right)}{d+z} - \frac{e}{2}$

3.2 บริเวณมุมมอง

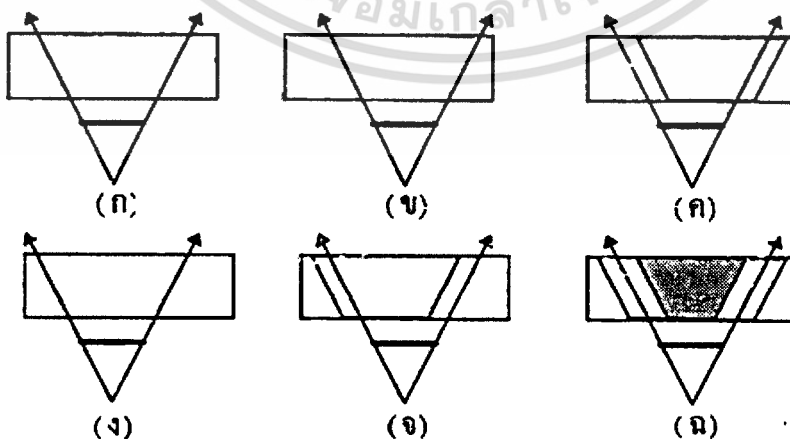
การฉายภาพแบบนอกแกน และแบบบนแกน มีรูปลักษณะทางคณิตศาสตร์ที่เหมือนกัน แต่จะต่างกันในเรื่องของบริเวณที่ถูกฉายบนระนาบของตาทั้งสองซึ่งเรียกว่า บริเวณของมุมมอง (Field of view) ในกรณีของการฉายภาพนอกแกนนั้น บริเวณของมุมมองหาได้จากบริเวณร่วมของการฉายภาพจากจุดศูนย์กลางทางซ้ายและทางขวา ที่ฉายลงบนพื้นที่ร่วมกันบนระนาบ ดังแสดงในภาพที่

3.3



ภาพที่ 3.3 บริเวณของมุมมอง สำหรับการฉายภาพนอกแกน

สำหรับการฉายภาพบนแกนนั้น ให้พิจารณาจากภาพที่ 3.4 กล้องสี่เหลี่ยมผืนผ้าแทนขอบเขตของวัตถุที่สนใจที่ถูกฉายลงบนระนาบการฉาย ในกรณีของภาพฉายสำหรับตาขวานั้นหาได้จาก การย้ายตำแหน่งของทุก ๆ วัตถุในขอบเขตที่สนใจไปทางซ้าย เป็นระยะทาง $e/2$ (ภาพที่ 3.4 ข) จากนั้นฉายภาพลงบนระนาบแล้วจึงเลื่อนภาพฉายบนระนาบทั้งภาพกลับไปทางขวาเป็นระยะทาง $e/2$ (ภาพที่ 3.4 ค) ให้สังเกตว่าการย้ายภาพกลับนี้เอง ที่ทำให้เกิดการสูญเสียข้อมูลที่ทำให้การฉายแล้วทางด้านซ้ายของจอภาพ นั่นคือเกิดช่องว่างทางด้านซ้าย ในทำนองเดียวกันการสร้างภาพฉายสำหรับตาซ้ายนั้น หาได้จากการย้ายตำแหน่งของทุก ๆ วัตถุในขอบเขตที่สนใจไปทางขวาฉายภาพลงบนระนาบ (ภาพที่ 3.4 ง) แล้วย้ายภาพที่ได้กลับไปทางซ้าย (ภาพที่ 3.4 จ) เมื่อรวมภาพฉายทั้งสองจะได้บริเวณของมุมมองร่วมของตาซ้ายและขวาในภาพที่ 2.4 ฉ บริเวณของมุมมองที่ได้นี้จะครอบคลุมบริเวณได้น้อยกว่าการฉายภาพแบบนอกแกน ซึ่งโดยทั่วไปอาจแตกต่างกันมากถึง 40 %



ภาพที่ 3.4 บริเวณของมุมมอง สำหรับการฉายภาพบนแกน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แม้ว่าการฉายภาพแบบบนแกนจะครอบคลุมบริเวณได้น้อยกว่า การฉายภาพแบบนอกแกน นั้น วิธีการฉายภาพแบบบนแกนง่ายต่อการดำเนินการ โดยเฉพาะอย่างยิ่งกับส่วนแสดงผลกราฟิกส์ ระดับเครื่องเวิร์คสเตชันที่สามารถทำการฉายภาพ การย้ายจุด และการย้ายภาพได้ด้วยฮาร์ดแวร์ ดังนั้นจะสามารถฉายภาพคู่สเตอริโอได้รวดเร็วกว่า

3.3 การเหลื่อมในแนวนอน

ในการเลือกระยะระหว่าง L_{cop} และ R_{cop} หรือ ค่า e ที่อธิบายในส่วนที่แล้วเป็นสิ่งสำคัญ ในการที่ผู้มองภาพจะรวมภาพที่ได้รับบนตาซ้ายและตาขวามาเป็นภาพเดียว การเลือกค่า e ขึ้นอยู่กับปัจจัยหลายประการเช่น สีเด่นของวัตถุ, ตำแหน่งของวัตถุบนจอภาพ, วิธีเหลื่อมภาพที่นำมาใช้แบบลบ (สองภาพไขว้กัน) หรือแบบบวก (สองภาพไม่ไขว้กัน), ประสบการณ์และระบบสายตาของผู้สังเกต, ระยะของตาจนถึงจอภาพ, ขนาดของจอภาพ และ ระยะจากวัตถุไปยังระนาบการฉาย

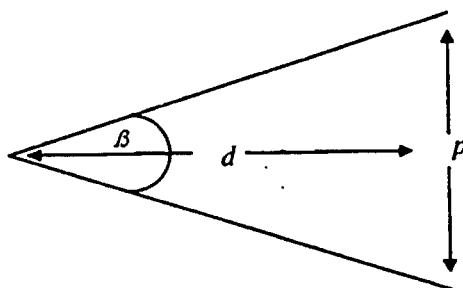
ขนาดของการเหลื่อมในแนวนอน ที่ปรากฏบนจอภาพเนื่องมาจากค่า e ใด ๆ ขึ้นกับระยะ ตั้งฉากจากระนาบการฉายของวัตถุในฉาก เราสามารถคำนวณจากจุดใด ๆ จากค่าแตกต่างของ x , ($x_r - x_l$) จากสมการ (4) และ (5) จะได้ ขนาดของการเหลื่อมในแนวนอน p คือ

$$\begin{aligned} p &= x_r - x_l \\ &= \left[\frac{d\left(x - \frac{e}{2}\right)}{d+z} + \frac{e}{2} \right] - \left[\frac{d\left(x + \frac{e}{2}\right)}{d+z} - \frac{e}{2} \right] \\ &= e \left(1 - \frac{d}{d+z} \right) \end{aligned} \quad (6)$$

โดยทั่วไป ค่าของ e จะสัมพันธ์กับปริมาณของการเหลื่อมในแนวนอน โดย

$$e = \frac{p}{1 - \frac{d}{d+z}} = \frac{p(i+1)}{i} \quad (7)$$

โดยที่ $i.d$ เป็นค่าความลึกที่มากที่สุดของบริเวณที่มีวัตถุในฉาก



ภาพที่ 3.5 แสดงมุมมองการมองในแนวนอน

3.4 ความสัมพันธ์ของการเลี้ยวในแนวนอนกับมุมการมอง

จอภาพที่ใช้สำหรับการแสดงภาพสเตอริโอ มีความหลากหลายตั้งแต่จอภาพธรรมดาเล็ก ๆ ไปจนถึงจอภาพขนาดใหญ่ ดังนั้น เราจึงเลือกที่จะแสดงค่า e ในรูปของมุมการมองในแนวนอน (Horizontal Visual Angle : HVA) สำหรับการเลี้ยวในแนวนอนขนาด p หน่วย บนจอภาพ HVA คือ มุม ดังที่แสดงไว้ในภาพที่ 3.5

$$p = 2d \cdot \tan\left(\frac{\beta}{2}\right) \quad (8)$$

จากสมการที่ (8) แทนค่า p ในรูปของ HVA ลงในสมการที่ (7) เราสามารถคำนวณค่า e โดยใช้ค่า HVA ที่มากที่สุดที่ยอมได้ ดังนี้

$$e = 2d \cdot \tan\left(\frac{\beta}{2}\right) \cdot \frac{i+1}{i} \quad (9)$$

โดยที่

- d = ระยะตั้งฉาก ระหว่าง Cop จากระนาบการฉาย
- e = ระยะระหว่าง Lcop และ Rcop
- β = มุมการมองในแนวนอน (HVA) ที่มากที่สุด
- i = ความลึกของฉากในหน่วยของ d (เป็นจำนวนเท่าของ d)

3.5 การหาค่าที่เหมาะสมของมุมการมองในแนวนอน

ในการเลือกค่าสำหรับ HVA เป็นส่วนที่สำคัญในการสร้างภาพสเตอริโอ หาก HVA มีค่ามากเกินไป การรวมภาพที่สร้างขึ้นของสมองจะยากในบางส่วนหรือทั้งหมดของภาพ หากค่า HVA น้อยเกินไป ผลที่เกิดจากความเป็นสเตอริโอก็จะหายไป และดูเป็นธรรมชาติมากหากจะกำหนดค่า HVA ที่มากที่สุดตามระยะจากตาไปยังจอภาพและระยะห่างระหว่างตา ยกตัวอย่างเช่น หากกำหนดให้ระยะระหว่างตาเป็น 6.35 เซนติเมตร และ อยู่ห่างจากจอภาพ 61 เซนติเมตร วัตถุที่อยู่ไกลเป็นอนันต์จะทำมุมระหว่างกันบนระนาบการฉายประมาณ 6 องศา และ HVA ของวัตถุที่อยู่หลังจากระนาบการฉาย 61 เซนติเมตร จะทำมุมประมาณ 3 องศา โดยทั่วไปแล้ว หากกำหนด HVA แล้วทำให้เป็นการยากที่จะดูภาพให้เป็นสามมิติก็ควรลดมุม HVA ลงมา

3.6 ปัจจัยที่มีผลกับคุณภาพของภาพ

3.6.1 ภาพหลอนและการรบกวนข้ามช่อง (Interocular Cross Talk and Ghosting)

ในระบบภาพสลับเชิงเวลา (Time-Multiplexed) ที่สมบูรณ์ เมื่อภาพสำหรับตาขวา กำลังแสดงบนจอภาพ ภาพสำหรับตาซ้ายก็จะถูกปิดไปอย่างหมดจด หรือในทางกลับกัน เมื่อภาพสำหรับตาซ้ายกำลังแสดงบนจอ ภาพสำหรับตาขวาก็จะถูกปิดไปอย่างหมดจดเช่นกัน แต่ในระบบจริง ๆ แล้ว การรบกวนข้ามช่องระหว่างภาพจะเกิดขึ้น ปัจจัยที่มีผลต่อปริมาณการรบกวนข้ามช่องได้แก่ อัตราการส่งผ่านแสงในสภาวะเปิดต่อสภาวะปิด (Dynamic Range) ของชุดเตอร์, การคงอยู่ของฟอสฟอรัสบนจอภาพ (Phosphor Persistence) และตำแหน่งทางแนวตั้งของภาพ ปัจจัยที่มีผลต่อการได้รับหรือเห็นภาพหลอน (Ghosting) ระหว่างภาพของตาซ้ายและขวา ซึ่งเป็นผลมาจากการรบกวนข้ามช่องได้แก่ ความสว่างของภาพ, ความเข้ม, ความซับซ้อนของพื้นผิว และ การเหลื่อมในแนวนอน

การรบกวนข้ามช่องเป็นผลมาจากการรวมกันของผลที่เกิดจากการรั่วของชุดเตอร์ ในขณะที่ยังปิดอยู่และฟอสฟอรัสที่ยังไม่จางหายไปของภาพตรงข้ามกับที่แสดงอยู่ในขณะที่ชุดเตอร์กำลังเปิดอยู่ ปริมาณการรั่วของชุดเตอร์ที่อยู่ในสถานะปิด จะถูกวัดในรูปของอัตราส่วนของการส่งผ่านของแสงในสภาวะเปิดต่อสภาวะปิด สำหรับการวัดปริมาณฟอสฟอรัสที่ตกค้างมาจากภาพตรงข้ามจะวัดโดยใช้ระยะเวลาที่ความสว่างของฟอสฟอรัสลดลงเหลือ 10 เปอร์เซ็นต์ จากความสว่างเริ่มต้น ซึ่งมักขึ้นอยู่กับสีของวัตถุ เช่น ฟอสฟอรัสหมายเลข P22 ที่ใช้กันมากในจอสีทั่วไป ฟอสฟอรัสสีแดงและน้ำเงินจะลดความสว่างลงเร็วกว่าฟอสฟอรัสสีเขียว

ผลกระทบเรื่องการคงอยู่ของฟอสฟอรัสยังขึ้นกับตำแหน่งของวัตถุบนจอภาพแบบราสเตอร์สแกน (Raster Scan) ฟอสฟอรัสที่อยู่ด้านบนของจอภาพจะมีเวลาในการสลายตัวมากกว่าบริเวณที่อยู่ด้านล่าง เมื่อซีทเตอร์เริ่มเปลี่ยนสถานะ ดังนั้นภาพที่อยู่ส่วนบน ๆ จะได้รับผลกระทบน้อยกว่าบริเวณที่อยู่ด้านล่าง

ปรากฏการณ์ภาพหลอน ซึ่งเป็นผลมาจากการรบกวนข้ามช่องระหว่างตาซ้ายและขวาสามารถลดลงได้โดยหลายปัจจัยเช่น ความสว่างของภาพ, ความเข้ม, ความซับซ้อนของพื้นผิว และ ปริมาณการเคลื่อนไหวในแนวนอน ปริมาณผลกระทบที่ได้รับจากปัจจัยเหล่านี้ ไม่สามารถวัดเป็นปริมาณที่วัดได้ง่าย ๆ แต่สามารถใช้หลักการง่าย ๆ ในการเพิ่มคุณภาพของภาพได้โดยที่ ภาพหลอนจะแปรผกผันโดยตรงกับความสว่างของภาพ, ความเข้ม และ ปริมาณการเคลื่อนไหวในแนวนอน และ แปรผกผันกับความซับซ้อนของพื้นผิวหรือรายละเอียดของภาพ

3.6.2 อัตราการกวาดภาพของจอแสดงผล

อัตราการกวาดภาพของจอแสดงผลจะมีผลต่อการกระพริบของภาพที่แสดงในจอส่วนใหญ่ที่อัตราการกวาดภาพเป็น 60 ภาพต่อวินาที จะทำให้แต่ละตาได้รับภาพในอัตรา 30 ภาพต่อวินาที การสลับภาพมักจะทำโดยใช้ ฮาร์ดแวร์ และมีสองบัฟเฟอร์ (Buffer) การทำระบบแบบตอบโต้มักทำได้ยากเนื่องจากการเตรียมภาพสำหรับแต่ละตานั้น ต้องทำให้ได้ภายใน 1/60 วินาที ในจะภาพที่มีอัตราการกวาดภาพเป็น 120 ภาพต่อวินาที มักจะทำการลดความละเอียดของภาพการแสดงผลและเพิ่มบัฟเฟอร์อีกสองบัฟเฟอร์สำหรับการเตรียมภาพที่ใช้แสดงผล การทำระบบแบบโต้ตอบทันทีสำหรับระบบเช่นนี้จะเป็นไปไม่ได้ในทางปฏิบัติมากกว่าแบบแรก

บทที่ 4

การพัฒนาโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์ด้วย ไมโครซอฟต์ DirectX

4.1 ความเป็นมาของไมโครซอฟต์ DirectX

ในสมัยก่อนการเขียนโปรแกรมให้ติดต่อกับอุปกรณ์ต่าง ๆ เป็นเรื่องยุ่งยาก ผู้พัฒนาจำเป็นต้องทำการศึกษาถึงคำสั่งต่าง ๆ ของฮาร์ดแวร์ แต่ละรุ่น แต่ละยี่ห้อ ซึ่งมีจำนวนมาก และเป็นเรื่องที่ทำให้เสียเวลา และเมื่อฮาร์ดแวร์ตัวเก่าล้าสมัยและฮาร์ดแวร์ตัวใหม่ถูกผลิตออกมา โปรแกรมเมอร์ก็จำเป็นต้องทำการศึกษาถึงวิธีการใช้คำสั่งต่าง ๆ ของฮาร์ดแวร์ตัวใหม่ ซึ่งอาจจะแตกต่างจากตัวเก่าโดยสิ้นเชิง หรือมีการเพิ่มเทคโนโลยีใหม่ ๆ เข้ามาในฮาร์ดแวร์ซึ่งเป็นเรื่องที่ทำให้ความลำบากแก่ผู้พัฒนาเป็นอย่างมาก

ในระบบปฏิบัติการวินโดวส์ ตั้งแต่รุ่น 3.1 เป็นต้นมา มีการใช้ไดรเวอร์ (Driver) ของฮาร์ดแวร์เพื่อที่จะเข้าถึงอุปกรณ์ เป็นเหตุให้นักพัฒนาโปรแกรมพบกับความลำบากในการเขียนโปรแกรมเพื่อติดต่อกับฮาร์ดแวร์ ซึ่งผู้พัฒนาโปรแกรมคาดหวังว่าในระบบปฏิบัติการวินโดวส์รุ่นใหม่ ๆ จะมีชุดคำสั่งที่จะช่วยให้การเขียนโปรแกรมติดต่อกับอุปกรณ์สามารถทำผ่านอินเตอร์เฟส (Interface) ตัวหนึ่งเพียงตัวเดียวได้ ซึ่งอินเตอร์เฟส (Interface) ตัวนี้จะทำการเรียกใช้ ไดรเวอร์ (Driver) ต่าง ๆ ของอุปกรณ์นั้น ๆ เอง ซึ่งจะทำให้การเขียนโปรแกรมเป็นไปอย่างสะดวก เพราะเพียงทำการศึกษาถึงคำสั่งการเรียกใช้อินเตอร์เฟส (Interface) ก็เพียงพอที่จะติดต่อกับอุปกรณ์ต่าง ๆ รุ่นใด ๆ ยี่ห้อใด ๆ ได้เลยทันที

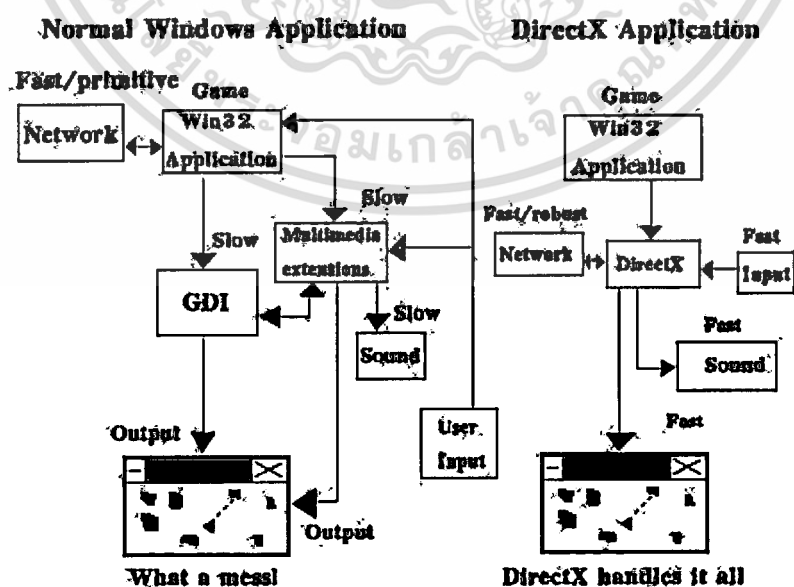
บริษัท Microsoft Corporation เล็งเห็นถึงความต้องการของโปรแกรมเมอร์ ตรงจุดนี้จึงได้ทำการผลิตชุดพัฒนา SDK สำหรับระบบปฏิบัติการวินโดวส์ขึ้นมาโดยมีจุดประสงค์ ที่จะช่วยให้การเข้าถึงฮาร์ดแวร์ต่าง ๆ เป็นเรื่องง่ายยิ่งขึ้นอีก ซึ่งไม่จำเป็นต้องไปทำการศึกษางานของฮาร์ดแวร์ตัวนั้น ๆ เหมือนสมัยก่อน โดยระยะแรกมีวัตถุประสงค์ เพื่อใช้พัฒนาเกมคอมพิวเตอร์โดยเฉพาะ จึงได้ตั้งชื่อว่า “Game kit” ต่อมา ได้เพิ่มความสามารถที่ช่วยทางด้านอื่น ๆ ด้วยเช่น มัลติมีเดีย (Multimedia) ดังนั้นจึงได้พัฒนาและถูกเรียกใหม่ว่า Microsoft DirectX และชุดพัฒนาจะมี SDK ต่อท้าย โดยปัจจุบันได้ออกมาถึง รุ่น ที่ 8.0 แล้ว

4.2 หลักการของ DirectX

สิ่งที่ไมโครซอฟต์ได้คำนึงถึงเป็นสิ่งแรกในการพัฒนา DirectX ก็คือจะต้องเป็นชั้นระหว่าง Application กับ Hardware ที่ต้องมีความสัมพันธ์กันมากเหมือนเป็นชั้นเดียวกัน เพราะต้องการให้ Application สามารถทำงานกับ Hardware ได้อย่างรวดเร็ว ความจริงแล้วก่อนที่ DirectX จะถูกพัฒนาขึ้นมา ได้มีชุดคำสั่งของ Windows ซึ่งใช้ติดต่อกับการ์ดแสดงผลอยู่ก่อนแล้ว เรียกว่า GDI (Graphic Device Interface) ซึ่งมีลักษณะของการทำงานคล้าย DirectX ก็คือสามารถติดต่อกับอุปกรณ์ Hardware ต่าง ๆ ได้โดยตรงโดยไม่ต้องทำการเรียกใช้ Driver ของการ์ดแสดงผลเอง แต่ความสามารถจะช้ากว่า และความสามารถบางอย่างยังไม่สามารถทดแทนได้ เช่น การที่โปรแกรมเมอร์ ทำการเรียกคำสั่ง ที่ติดต่อกับ Hardware แสดงผลซึ่งมีผลทำให้สามารถทำงานเร็วมาก เพราะทำงานบน Hardware และหากว่าเครื่องของผู้ใช้ Application นั้น ไม่มี Hardware ดังกล่าว ถ้าเป็น GDI จะไม่สามารถทำงานหรือ ใช้งาน Application นั้นได้เลย แต่หาก โปรแกรมเมอร์เรียกใช้ Hardware ผ่านทาง DirectX เมื่อ DirectX ทำการตรวจสอบหา Hardware ดังกล่าวและหาไม่พบ ก็จะทำการจำลอง Hardware ตัวนั้นเหมือนกับว่ามี Hardware ตัวนั้นอยู่และใช้ Algorithm ที่กำหนดไว้ทำการประมวลผล และทำการแสดงผลแทน Hardware ตัวนั้น ซึ่งผู้ใช้สามารถเรียกใช้ Software แทน Hardware และยังสามารถใช้งาน ได้เหมือนเดิม

DirectX คือชุดของ Dynamic Link Library (DLL) ซึ่งสามารถทำให้โปรแกรมเมอร์เข้าถึง Hardware ได้อย่างรวดเร็ว โดยไม่ผ่านทาง Hardware ตามหลักการของ Windows ดังแสดงในภาพที่

4.1



ภาพที่ 4.1 แสดงการเปรียบเทียบ Application ที่ใช้ GDI กับ DirectX

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

DirectX ยังเป็นมาตรฐานของ Software และ Hardware ในปัจจุบัน โดยบริษัท Software ส่วนมากที่มีอุปกรณ์ ที่เกี่ยวข้องกับ DirectX จะเขียนโปรแกรมให้เรียกใช้ DirectX และรวมไปถึง บริษัทที่ผลิต Hardware ก็ยังสนับสนุน โดย DirectX จะทำการเรียกใช้คำสั่งของ DirectX เพราะฉะนั้นเราสามารถที่จะมั่นใจได้ว่า DirectX จะเป็นที่แพร่หลายในอนาคต

Microsoft ได้ทำให้ DirectX มีคุณสมบัติพิเศษอย่างหนึ่งก็คือ เกมที่เขียนโดยใช้ DirectX 1.0 จะสามารถทำงานได้บนคอมพิวเตอร์ ที่ติดตั้ง DirectX 3.0 หรือ 6.0 และยิ่งไปกว่านั้น Microsoft รู้ว่าเทคโนโลยี DirectX จะต้องหลุดออกจากหลักการ หากว่าถูกเขียนโดยไม่มีการมองการณ์ไกล และการวางแผน ดังนั้นจึงจำเป็นที่จะต้องใช้การโปรแกรมลักษณะของ Object Oriented ซึ่งสามารถปรับปรุงให้ดีขึ้นได้ และสามารถทำงานได้กับหลาย ๆ ภาษา และเป็นลักษณะของ Black Box เทคนิคเช่นนี้ถูกเรียกว่า Component Object Model (COM)

4.3 ส่วนประกอบของ DirectX

DirectX ประกอบด้วยส่วนต่าง ๆ ดังนี้

4.3.1 HAL (The Hardware Abstraction Layer)

HAL คือ Software ระดับล่างสุดใน DirectX ซึ่งบรรจุ Driver ของ Hardware จากผู้ผลิต Hardware รายนั้น ๆ เพื่อที่จะทำการควบคุม Hardware ได้โดยตรง Software ในชั้นนี้ ให้ประสิทธิภาพสูงสุด เพราะเป็นชั้นที่ทำการติดต่อกับ Hardware โดยตรง

4.3.2 HEL (The Hardware Emulation Layer)

HEL จะวางอยู่ชั้นก่อนหน้า HAL โดยปกติ DirectX ถูกออกแบบมาให้ใช้ Hardware ให้ได้เต็มประสิทธิภาพและด้วยความเร็วสูง หากว่ามี Hardware ชนิดนั้น ๆ อยู่ แต่ DirectX ก็ยังสามารถทำงานได้อยู่หากว่า Hardware ชนิดนั้นไม่มีอยู่ ตัวอย่างเช่น หากเราต้องการสร้างภาพ ๆ หนึ่งขึ้นมาซึ่งจำเป็นต้องเรียกใช้ฟังก์ชันบน Hardware ชนิดหนึ่ง โดยทำการเรียกผ่าน DirectX ซึ่งถ้ามี Hardware ตัวนั้นอยู่ก็จะทำการเรียกใช้ฟังก์ชันบน Hardware ชนิดนั้นได้โดยตรง แต่หากว่า Hardware ชนิดนั้นไม่มีอยู่ในเครื่อง HEL จะเข้ามาทำงานในส่วนตรงนี้ โดยการจำลองตัวเอง เป็น Hardware ตัวนั้นๆ และทำงานเคียงคู่กับ HAL ซึ่งแน่นอนว่าทำให้การทำงานช้าลง เพราะเป็นการจำลอง Hardware แต่ก็ยังทำให้ Application นั้นทำงานได้ตามปกติได้

4.3.3 Direct Draw

อาจจะเรียกได้ว่า ส่วนนี้เป็นส่วนที่สำคัญที่สุดของ DirectX เพราะ Direct Draw สามารถที่จะทำการเรียกใช้งานการ์ดแสดงผลด้วยความเร็วสูง โดยที่ Direct Draw รู้จักการ set mode ของจอภาพทุกๆ mode ไม่ว่าจะเป็น high-resolution หรือจะเป็น true color modes แล้วยังสนับสนุนเรื่องการจัดการสี การกำหนดขอบเขตจอภาพ และการทำ Animation

4.3.4 Direct 3D

ส่วนประกอบของ DirectX ชนิดนี้เป็นส่วนประกอบส่วนเดียวที่สามารถติดต่อกับ อุปกรณ์แสดงผล 3 มิติได้ ทำให้สามารถเขียนโปรแกรมประเภท 3 มิติได้ เช่น เกมขับเครื่องบิน และ Simulation ต่าง ๆ

4.3.5 Direct Sound

ก่อนที่จะมี DirectX การเขียนโปรแกรมติดต่อกับการ์ดเสียงเป็นเรื่องที่ยากมาก ซึ่งการเขียนโปรแกรมที่อาศัยการ์ดเสียง และสร้างเสียงที่มีคุณภาพในสมัยก่อนมักจะว่าจ้างให้บริษัทที่ทำงานด้านนี้ โดยเฉพาะเป็นผู้ออกแบบและทำการ Coding เช่นบริษัทเกม อาจเป็นบริษัทหนึ่ง แล้วจ้างบริษัทที่ทำระบบเสียงให้ทำการเพิ่มเสียงเพลงเข้าไปในตัวเกม เป็นต้น แต่ปัจจุบันเมื่อ Direct Sound ถือกำเนิดขึ้นสถานการณ์เช่นนั้นจึงหมดลงตามยุค เพราะ Direct Sound ลดขั้นตอนการเข้าถึงการ์ดเสียง และทำงานได้กับการ์ดเสียงทุกชนิด

4.3.6 Direct Sound 3D

Direct Sound 3D อยู่บนพื้นฐานของ Direct Sound และทำงานได้กับระบบเสียง 3 มิติ ซึ่งจำเป็นต้องมีการคำนวณทางคณิตศาสตร์ Direct Sound 3D จะทำการคำนวณ และ พิกัดตำแหน่งของเสียงในระบบ 3 มิติ ซึ่งจะทำให้เสียงที่ออกมาฟังสมจริงมากขึ้น

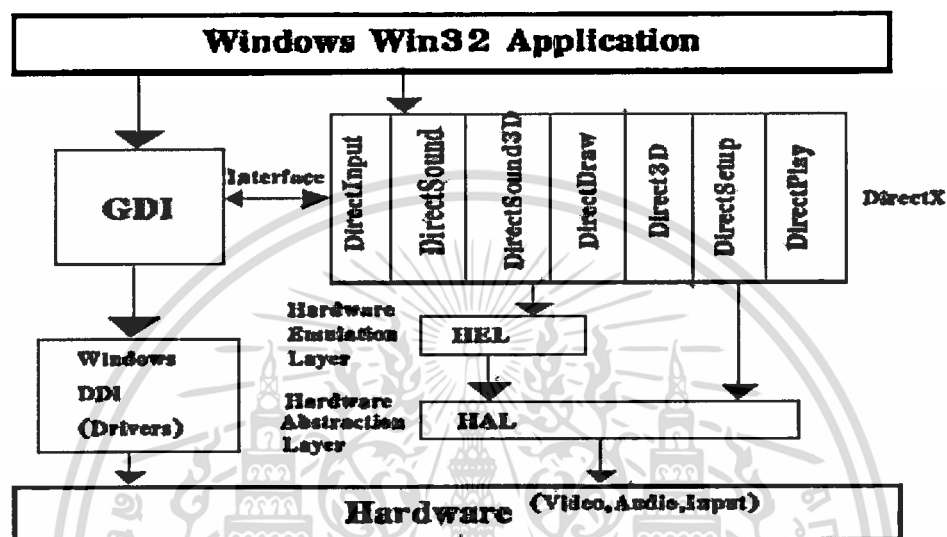
4.3.7 Direct Input

เป็นความสามารถใหม่ที่เพิ่งจะถูกเพิ่มเข้ามาใน DirectX 3.0 ซึ่งจะทำให้การรับข้อมูลทำได้หลากหลายขึ้นและทำได้โดยง่าย โดยรับจาก Keyboard, Mouse และ Joystick ซึ่งใน เวอร์ชัน 5.0 ยังสนับสนุน Joystick แบบ Force Feed Back (สั่นสะเทือนได้) ในสมัยก่อน Joystick มีปุ่มเพียงเล็กน้อย แต่ในปัจจุบันมีมากมายหลายปุ่ม และมีหลากหลายชื่อ ทำให้การติดต่อกับ Joystick ในแต่ละปุ่มต้องทำการศึกษาถึง Joystick ชนิดนั้น ๆ อย่างลึกซึ้ง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.3.8 Direct Play

เป็นส่วนประกอบที่ใช้ติดต่อกับอุปกรณ์ทางด้าน Network การอ้าง Port ต่าง ๆ หรือการสร้างเกมที่สามารถเล่นผ่านเครือข่ายได้



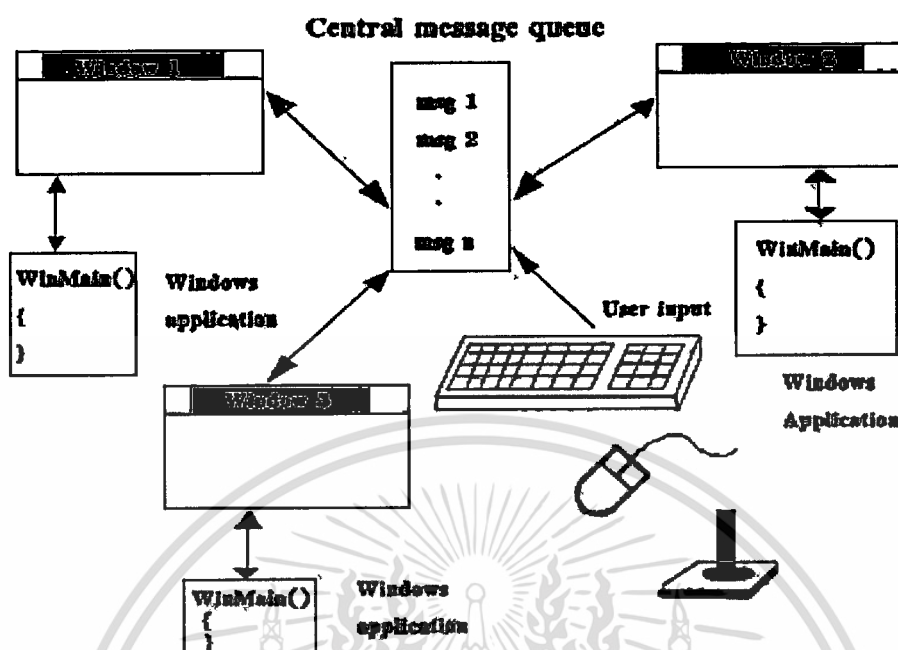
ภาพที่ 4.2 ส่วนประกอบของ DirectX

4.4 การเขียนโปรแกรม ภายใต้ระบบปฏิบัติการวินโดวส์

การเขียนโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์ อาจจะเป็นเรื่องที่ยากหรือยาก ขึ้นอยู่กับขนาดของโปรแกรมที่ต้องการทำ การเขียน Word processor อาจต้องการความรู้ทางด้านการเขียนโปรแกรมที่มากพอสมควร แต่สำหรับการเขียนเกมโดยใช้ DirectX ต้องการเพียงเล็กน้อยเท่านั้น ในส่วนนี้จะอธิบายถึงการเขียนโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์ และพื้นฐานที่ควรรู้สำหรับการเขียนโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์ เพื่อนำไปใช้ในการพัฒนาต่อไป

ระบบปฏิบัติการวินโดวส์เป็นระบบปฏิบัติการแบบ Event-based โดย Message เป็นเสมือนสัญญาณที่จะบอกให้ระบบปฏิบัติการต้องทำอะไร ทำอย่างไร หรือเกิดอะไร ตัวอย่างเช่น เมื่อผู้ใช้งานต้องการเปลี่ยนขนาดของ Window ตัว Application Windows ก็จะมีการส่ง Message ไปยัง Application ตัวนั้น ว่าจะเปลี่ยนขนาดอย่างไร หรือเมื่อผู้ใช้งานกดแป้นพิมพ์บน Keyboard Message ของตัวแป้นพิมพ์ที่กด ก็จะถูกส่งไปยัง Application เพื่อกระทำการประมวลผลข้อมูลนั้น และแสดงผลข้อมูลนั้นออกมา เช่น แสดงเมนูย่อย จากเมนูบาร์หลัก หรือไม่ว่าจะเป็นการที่ window ตัวใหม่ถูกแสดงออกมาทับตัวเก่าที่แสดงอยู่ หน้าจอจะต้องทำการวาดใหม่อีกครั้ง โดย Repaint message จะกระทำการในส่วนนี้

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 4.3 ระบบการส่ง message ใน Windows

เพราะฉะนั้นอาจจะกล่าวได้ว่า ทุกสิ่งทุกอย่างที่เกิดขึ้นใน Windows ก็คือ message นั่นเอง ดังภาพที่ 4.3 ต่อไปจะกล่าวถึงส่วนประกอบโดยทั่วไป ในการสร้าง windows application ที่ต้องมีได้แก่

4.4.1 Win32 Libraries

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

เมื่อสร้าง Application ที่ใช้ DirectX ผู้ใช้ก็ต้องเพิ่ม Libraries ของ DirectX เข้าไปใน Application นั้นด้วย

4.4.2 The Include files

โปรแกรมเมอร์ต้อง include files ที่จำเป็นสำหรับ application นั้นไว้ในตัวโปรแกรมด้วยเสมอ และโดยทั่วไป ใน Windows application จะต้องเริ่มต้นด้วย

```
#include <windows.h>
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ตัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และอาจจะเพิ่ม include files ที่จำเป็นสำหรับ application เข้าไปอีกได้ เช่น stdio.h ddraw.h หรือ Header File ที่เกี่ยวข้องที่ใช้ใน Application เช่น Win3D Header Library เป็นต้น

4.4.3 WinMain

เป็นฟังก์ชันหลักที่ Windows application ทุก application จำเป็นต้องกำหนดไว้คล้าย ๆ กับฟังก์ชัน main() ในภาษา C และ ภาษา C++ การประกาศ Prototype ของ WinMain() จะเป็นดังนี้

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

4.4.4 Windows class

โปรแกรมเมอร์ที่ต้องการสร้าง Windows application จำเป็นต้องกำหนด Windows class ขึ้นมาด้วย ซึ่ง windows class จะอธิบายถึงคุณลักษณะโดยทั่วไปของ windows ทั้งหมด โดยหลังจากโปรแกรมเมอร์ทำการสร้าง windows class ขึ้นมาแล้วโปรแกรมเมอร์ ก็จะสามารถสร้าง window ได้ตาม windows 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    lpstrMenuName; // name of Menu
                        LPCTSTR    lpstrClassName; // name of class
                        } WNDCLASS;
```

4.4.5 Registering the Windows class

หลังจากที่ทำการสร้าง windows class ขึ้นมาแล้ว ขั้นตอนต่อไปจะทำการ register class การ register จะทำให้ Windows รู้ว่า มี windows class ที่สร้างขึ้นมา และสามารถให้โปรแกรมเมอร์สร้าง window ที่มีลักษณะตาม windows class ได้โดยชื่อ ASCII ของ class เพียงอย่างเดียว การ register class กระทำได้โดย

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

4.4.6 การสร้าง Window

การจะสร้าง window ขึ้นมาต้องทำการเรียกฟังก์ชัน 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,              // ตำแหน่งแนวนอน x ของวินโดวส์
    int         y,              //ตำแหน่งแนวนอน 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)
);
```

4.4.7 การแสดง Window

หลังจากที่ทำการ create window เสร็จแล้ว และต้องการแสดงผลออกมา (ในกรณีที่ตั้งค่า parameter ของ dwStyle ไม่ใช่ WS_VISIBLE ต้องทำการเรียกฟังก์ชัน ShowWindow() เสมอ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

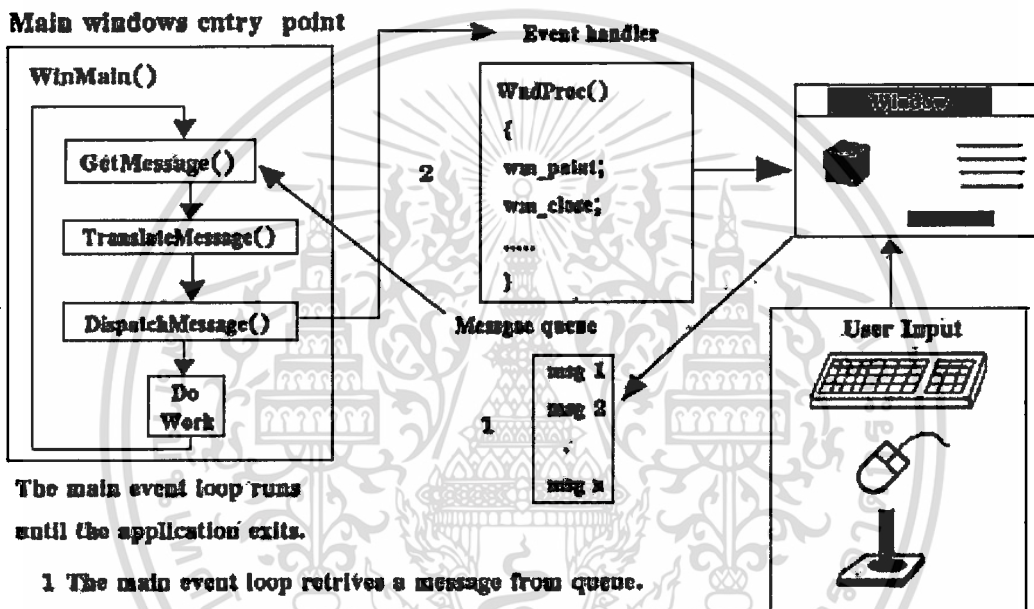
```

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

```

4.4.8 Event Loop

main event loop คือ loop ที่คอยรับ message record ที่เก็บไว้จาก message queue แล้วทำการแปลง แล้วส่งต่อไปยัง event handler ดังภาพที่ 4.4



The main event loop runs until the application exits.

- 1 The main event loop retrieves a message from queue.
- 2 The message is processed and dispatched to the event handler.

ภาพที่ 4.4 The main event loop

โดยมีฟังก์ชัน ที่ทำหน้าที่ต่างกัน อยู่ 3 ฟังก์ชัน ใน event loop นี้ได้แก่

4.4.8.1 GetMessage

เป็นฟังก์ชัน ที่ทำหน้าที่รับ message จาก message queue ถ้ากรณี message queue ว่าง ฟังก์ชันก็จะทำการรอนจนกว่าจะมี message เข้ามา

```

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

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาค้นคว้า ไม่อนุญาตให้เผยแพร่โดยไม่ได้รับอนุญาต
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.4.8.2 TranslateMessage

เป็นฟังก์ชันที่ใช้แปลง message event ให้อยู่ในรูปแบบพื้นฐาน ของ event เพื่อจะนำไปใช้ในส่วนของ event handler

//pointer to message structure

BOOL TranslateMessage(CONST MSG *lpmsg);

4.4.8.3 DispatchMessage

ทำการส่ง message หลังจากทำการ Translate Message ไปยัง event handler เพื่อทำการประมวลผลต่อไป

//pointer to message structure

LONG DispatchMessage(CONST MSG *lpmsg);

4.4.9 Event handler

event handler หรือ WinProc คือ ฟังก์ชันที่ต้องเขียนเพื่อ handle ทุก message ที่ application นั้นสนใจ Windows อาจจะส่ง message มากมายหลายอย่าง 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

4.5 Event Driven Programming

ในการเขียนโปรแกรมภายใต้ ระบบปฏิบัติการวินโดวส์ อย่างแรกที่เราจะเข้าใจคือ 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 ในที่นี่ได้รวมถึงทุกอย่างที่เกิดขึ้น 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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

programming ก็คือการเขียนโปรแกรมที่จะทำงานโดยการ check message ที่ได้รับไปเรื่อย ๆ แล้วทำงานตามที่กำหนด เมื่อมี message ที่ Program สนใจ

หากเปรียบเทียบกับ การเขียนโปรแกรมโดยทั่วไป การทำงานจะทำงานแบบเรียงลำดับตาม ผู้เขียนโปรแกรมได้เขียนไว้ แต่สำหรับ event driven programming นั้นคือ ลักษณะที่ผู้ใช้เป็นผู้ควบคุมโปรแกรมมากกว่าที่โปรแกรมจะควบคุมให้ผู้ใช้ทำตาม ยกตัวอย่างเช่น ในการเขียนโปรแกรมภายใต้ ระบบปฏิบัติการ Dos ถ้าโปรแกรมจะรับ Input 1 key โปรแกรมเมอร์อาจจะใช้ getch() แต่ถ้าเป็น event driven programming เมื่อมีการกดปุ่ม ระบบปฏิบัติการจะส่ง message บอก program ว่าได้มีการกดแป้นพิมพ์ขึ้น

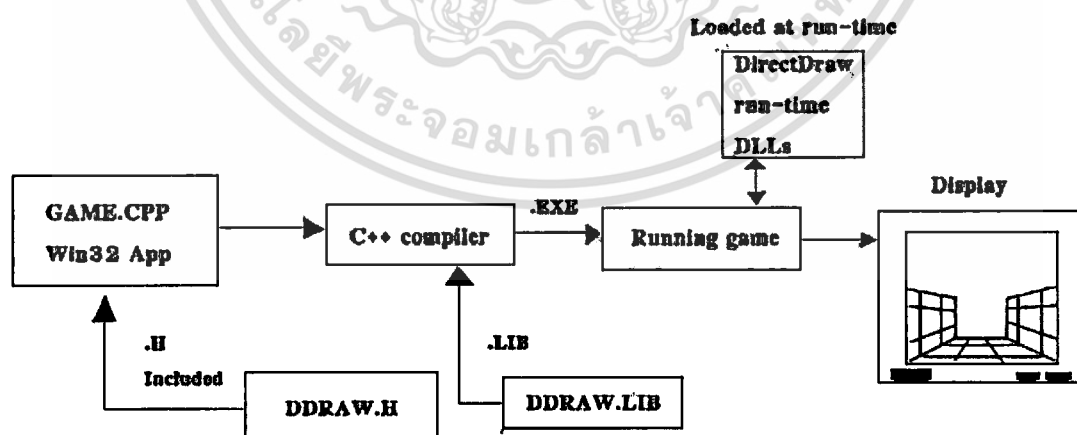
4.6 วิธีการแสดงภาพ

4.6.1 ความรู้ความเข้าใจเบื้องต้นของการแสดงภาพ

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

DirectDraw คือ ไลบรารีตัวหนึ่งซึ่งจะอยู่ใน File DDRAW.LIB และเวลาที่จะเรียกใช้ในภาษาซี ต้อง Include File DDRAW.H เข้าไปในโปรแกรมด้วย ดังนี้

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



ภาพที่ 4.5 เครือข่ายของ Direct Draw

4.6.2 การสร้าง DirectDraw Object

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

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

ฟังก์ชันนี้จะหมายถึง DirectDrawCreate จะสร้าง DirectDraw Object ไว้ที่ตำแหน่งแอดเดรส lplpDD และจะได้รับการแสดงผลแบบ lpGUID (ถ้าใส่ค่าเป็น NULL จะได้รับการแสดงผลแบบ Default) ฟังก์ชันนี้จะทำงานสำเร็จหรือไม่นั้น จะแจ้งผ่านค่า HRESULT มาโดย HRESULT เป็นชนิดข้อมูลชนิดหนึ่งของ DirectX ซึ่งใช้สำหรับตรวจสอบการทำงานของฟังก์ชัน ซึ่งฟังก์ชันส่วนใหญ่ของ DirectX จะทำการคืนค่านี้กลับมาเพื่อตรวจสอบ ค่าที่บ่งบอกถึงความสำเร็จของฟังก์ชันก็คือ

DD_OK

หากฟังก์ชันคืนค่าอื่น ๆ ที่ไม่ใช่ค่านี้ หมายถึง การทำงานของฟังก์ชันมีความผิดพลาด ซึ่งจะแจ้งข่าวสารข้อผิดพลาดมาทาง HRESULT เช่นกัน (แทนที่จะเป็น DD_OK)

ตัวอย่าง

```
// ทำการสร้างตัวแปรที่ใช้เก็บค่าแอดเดรสของ DirectDraw Object
LPDIRECTDRAW lpdd;

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

เมื่อได้ทำการสร้าง Object ของ DirectDraw เรียบร้อยแล้ว วิธีการเรียกใช้ฟังก์ชันมีวิธีดังนี้

lpdd->Function()

เมื่อเราทำงานกับฟังก์ชันนั้น ๆ เสร็จเรียบร้อยแล้ว เราต้องทำการลบ Object นี้ออกจากหน่วยความจำ ได้โดย

lpdd->Release()

4.6.3 การกำหนดค่า Cooperation level ของ DirectDraw

การที่จะเขียนโปรแกรมเพื่อทำงานภายใต้ ระบบปฏิบัติการวินโดวส์ เราจำเป็นจะต้องทำการขอแบ่งทรัพยากรต่าง ๆ จากสิ่งแวดล้อมบนวินโดวส์ ซึ่ง DirectDraw ก็ได้ให้ฟังก์ชันเพื่อใช้ในการกำหนดค่าความสำคัญและกำหนดระบบที่จะใช้ของโปรแกรมของเรา ดังนี้

HRESULT SetCooperativeLevel (HWND hWnd, DWORD dwFlags);

HWND hWnd คือ handle ของวินโดวส์หรือโปรแกรมที่กำลังทำงานด้วย โดยปกติ แล้วเราเพียงแค่ทำการกำหนดค่าตัวแปรนี้แล้ว compiler จะทำการสร้างให้เองโดยอัตโนมัติ

DWORD dwFlags คือ ค่าที่จะกำหนดให้กับฟังก์ชัน โดยมีดังนี้

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	บ่งบอกให้วินโดวส์ทราบว่าโปรแกรมนี้อาจเป็นเหมือน Application ทั่วไป (เช่น Microsoft Word หรือ ACDSee) โดยหากเรากำหนดค่านี้แล้ว เราจะไม่สามารถกำหนดค่า DDSCl_ALLOWMODEX, DDSCl_EXCLUSIVE หรือ DDSCl_FULLSCREEN ได้
DDSCl_NOWINDOWCHANGES	บอกว่าโปรแกรมนี้ไม่สามารถที่จะ minimize หรือ restore โปรแกรมอื่นๆ ในขณะที่โปรแกรมนี้ทำงานอยู่ได้

ตารางที่ 4.1 ค่า Cooperative level ของ DirectDraw

ตัวอย่าง

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

หมายถึง กำหนดให้โปรแกรมเราเป็นแบบ Application ทั่ว ๆ ไป

```
lpdd->SetCooperativeLevel(hwnd,DDSCl_ALLOWMODEX
```

```
| DDSCl_FULLSCREEN
```

```
| DDSCl_EXCLUSIVE
```

```
| DDSCl_ALLOWREBOOT);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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

4.6.4 การกำหนดโหมดการแสดงผล

ความสามารถในการเปลี่ยนโหมดการแสดงผลคือความสามารถที่สำคัญที่สุดของ DirectDraw ฟังก์ชันที่ใช้เปลี่ยนโหมดการแสดงผลคือ

```
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
- ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากร โดย SetCooperativeLevel
- ทำการกำหนดขนาดการแสดงผล โดยใช้ SetDisplayMode

ตัวอย่าง

```
LPDIRECTDRAW    lpdd;

if (DirectDrawCreate(NULL,&lpdd,NULL)!=DD_OK)

{ //ERROR //}

lpdd->SetCooperativeLevel(hwnd,DDSCL_ALLOWMODEX |

    DDSCL_FULLSCREEN | DDSCL_EXCLUSIVE |

    DDSCL_ALLOWREBOOT);

if ((lpdd->SetDisplayMode(640,480,8))!=DD_OK)

{ //ERROR }
```

4.6.5 การประยุกต์ใช้ DirectDraw

DirectDraw สามารถนำมาใช้ได้หลายๆด้าน แต่ด้านที่ถือว่านิยมที่สุดและนำ DirectDraw มาใช้มากที่สุดก็คือด้านการสร้างเกมส์คอมพิวเตอร์ ในปัจจุบัน เกมส์ส่วนมาก นำ DirectDraw มาใช้ ในการสร้าง Animation ซึ่ง DirectDraw สามารถเข้าถึง Hardware ด้านการแสดงผลได้โดยตรง ดังนั้นการทำงานต่างๆจึงรวดเร็วมาก และผู้ใช้ไม่จำเป็นต้องเข้าไปยุ่งเกี่ยวกับคำสั่งในระดับ Hardware เลย

4.6.6 การวาดภาพด้วย DirectDraw

วิธีที่ใช้ในการวาดภาพ ก็คือการนำภาพที่เราได้วาดไว้แล้วจากโปรแกรมวาดภาพ ต่างๆ เช่น Adobe Illustrator หรือจะ Adobe Photoshop หรือเราอาจนำภาพมาจาก โปรแกรมสร้างภาพสามมิติเช่น 3D Studio ก็ได้ แต่ภาพจะต้องเป็นประเภท BMP โดยเราจะนำภาพที่เราได้สร้างไว้แล้วเก็บไว้ในหน่วยความจำ หรือที่เรียกว่า Surface แล้วทำการ ถ่ายเทข้อมูลไปยังหน่วยความจำที่การ์ดแสดงผล หลังจากนั้น จึงนำแสดงขึ้นจอภาพ โดย รายละเอียดของหลักการจะกล่าวในหัวข้อถัดไป

DirectDraw นิยามหน่วยความจำที่ไว้ใช้แสดงผลด้วยคำว่า Surface โดยเราสามารถนำภาพต่างๆเก็บไว้ ณ ที่นี้ได้ ไม่ว่าจะเป็นภาพสองมิติ หรือภาพสามมิติก็ตาม ตาม ปกติเราต้องมีอย่างน้อย 1 Surface คือ Primary Surface ซึ่งเป็น Surface ที่เรามองเห็นขณะ นั้น ๆ บนจอภาพ โดยเราสามารถมี Surface ได้มากแค่ไหนก็ได้โดยขึ้นอยู่กับหน่วยความ จำที่มีอยู่

4.6.7 การสร้าง Surface

หลักการสร้าง Surface มีดังต่อไปนี้

4.6.7.1 Surface ที่สร้างนั้นมีขนาดเท่าใดก็ได้ ขึ้นอยู่กับภาพที่เราจะนำมาใส่ แต่ Surface ที่เป็น Primary Surface จะต้องมีความเท่ากันกับขนาดของความละเอียดของจอภาพขณะนั้น เช่นหากเรากำหนดให้โปรแกรมเรามีความละเอียดระดับ 800x600 เราก็ต้องสร้างให้ Primary Surface มีขนาด 800x600 ด้วยเช่นกัน

4.6.7.2 การสร้าง Surface เราสามารถเลือกที่จะใช้หน่วยความจำที่การ์ดแสดงผล (VRAM) หรือจะใช้หน่วยความจำหลัก (System Memory) ซึ่งถ้าเราสามารถ VRAM ได้จะทำให้การแสดงผลมีความเร็วสูงขึ้น แต่ VRAM มีขนาดที่ไม่ใหญ่นัก ดังนั้นหากเราไม่สามารถใช้ VRAM ได้อีก เราสามารถไปใช้ System Memory แทนได้

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

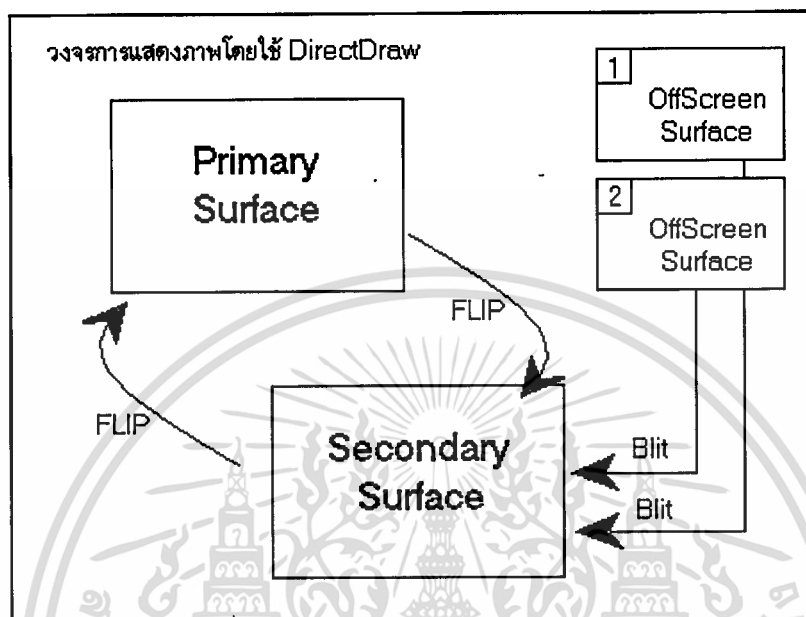
โดยทั่วไปโปรแกรมทางมัลติมีเดียส่วนมากที่สร้างขึ้นจะใช้ Surface ดังนี้

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 ส่วนมาก

แล้ว เกมจะมีจำนวน Surface นี้นมากที่สุด เพื่อใช้เก็บรูปภาพต่าง ๆ ลักษณะของการแสดงภาพจะเป็นวัฏจักรดังภาพที่ 4.6



ภาพที่ 4.6 การแสดงภาพ โดย DirectDraw

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

การสร้าง Surface จะต้องทำการใส่ข้อมูลลงในโครงสร้างข้อมูลแบบ DDSURFACEDESC แล้วจึงทำการเรียกฟังก์ชัน CreateSurface() เพื่อทำการสร้าง Surface นั้น ๆ โดยฟังก์ชัน CreateSurface() มีการรับค่าต่าง ๆ ดังต่อไปนี้

```
HRESULT CreateSurface(LPDDSURFACEDESC lpDDSurfaceDesc,
                      LPDIRECTDRAW_SURFACE FAR *lpDDSurface,
                      IUnknown FAR *pUnkOuter);
```

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

ดังเช่นฟังก์ชันอื่น ๆ หากฟังก์ชันนี้ทำงานสำเร็จจะคืนค่า DD_OK กลับมา แต่ก่อนที่เราจะทำการใช้ฟังก์ชัน CreateSurface() ได้ เราจำเป็นต้องทำการใส่ค่าลงไปยังโครงสร้างข้อมูลที่มีชื่อ DDSURFACEDESC (Direct Draw Surface Description) ก่อน โดยมีรายละเอียดของโครงสร้างดังนี้

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

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        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 คือการกำหนดให้สามารถเรียกใช้การทำงานได้ทุก ๆ ส่วนของ Surface

หากเรากำหนด Flag ค่าไหน การทำงานในส่วนนั้นจะสามารถเรียกใช้งานได้ แต่ถ้าเราเรียก DDSD_ALL เพียงตัวเดียว ทุกตัวจะสามารถทำงานได้ (แต่เราใช้งานจริง ๆ เพียงบางตัวเท่านั้น) เราจะเพียงบอกให้ทราบว่ามี Flag อะไรบ้าง แต่เราจะอธิบายเพียงบาง Flag ที่ใช้งาน

```

DDSD_ALPHABITDEPTH, DDSD_BACKBUFFERCOUNT, DDSD_CAPS,
DDSD_CKDESTBLT, DDSD_CKDESTOVERLAY, DDSD_CKSRCLBLT,
DDSD_CKSRCOVERLAY, DDSD_HEIGHT, DDSD_LINEARSIZE,
DDSD_LPSURFACE, DDSD_MIPMAPCOUNT, DDSD_PITCH,
DDSD_PIXELFORMAT, DDSD_REFRESHRATE, DDSD_WIDTH,
DDSD_ZBUFFERBITDEPTH

```

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

- ddsCaps คือส่วนที่ใช้กำหนดคุณสมบัติของ Surface โดยมีโครงสร้างดังนี้

```
typedef struct _DDSCAPS
```

```
{
```

```
    DWORD dwCaps;
```

```
} DDSCAPS.FAR * LPDDSCAPS;
```

- dwcaps คือส่วนที่ใช้เก็บค่าคุณสมบัติของ 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 Flip 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 4.2 Surface Capabilities Flags

ตัวอย่าง ตั้งแต่เริ่มจนทำการสร้าง PrimarySurface และ SecondarySurface

```

LPDIRECTDRAW      lpdd; // ประกาศตัวแปรพอยเตอร์ของ DirectDraw
DDSURFACEDESC     ddsd; // ประกาศตัวแปรที่ใช้เก็บค่าของโครงสร้าง
                        // ข้อมูล DDSURFACEDESC
// เมื่อ Surface ถูกสร้างขึ้นจะทำการเก็บ address ไว้ที่ตัวแปรนี้
LPDIRECTDRAW_SURFACE lpddsprimary;
// ทำการสร้าง DirectDraw Object
DirectDrawCreate(NULL,&lpdd,NULL);
// ทำการกำหนดระดับความสำคัญและการแบ่งใช้ทรัพยากร
lpdd->SetCooperativeLevel(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);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4.6.8 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 SecondarySurface ยังอยู่ในสถานะที่กำลังวาดอยู่ และยังไม่สามารถ Flip ได้ โดยค่านี้นี้จะทำให้ฟังก์ชันทำการลอง Flip ไปจนกว่าจะ Flip ได้

วิธีใช้เป็นดังนี้

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

4.6.9 การวาดภาพลงบน Surface

แต่ก่อนอื่นที่เราจะทำการ Flip เราจะต้องทำการวาดลงบน Surface ให้ได้ก่อน โดยเราสามารถใช้ฟังก์ชัน Blt ในการคัดลอกข้อมูล โดยเราสามารถคัดลอกข้อมูลจาก Surface ใดไป Surface ใดก็ได้ โดยมีโครงสร้างของฟังก์ชันดังนี้

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

- lpDestRect คือ พ้อยเตอร์ที่ชี้ไปยังตัวแปรประเภท RECT ของ Surface จุดหมาย ซึ่งตัวแปรประเภท RECT นี้มีโครงสร้างดังนี้

```
typedef struct _RECT
```

```
{
```

```
    LONG top;    //ตำแหน่ง พิกเซล ที่อยู่บนสุดของรูปภาพ
```

```
    LONG left;   //ตำแหน่ง พิกเซล ที่อยู่ซ้ายสุดของรูปภาพ
```

```
    LONG right;  //ตำแหน่ง พิกเซล ที่อยู่ขวาสุดของรูปภาพ
```

```
    LONG bottom; //ตำแหน่ง พิกเซล ที่อยู่ล่างสุดของรูปภาพ
```

```
}RECT
```

เอกสารนี้เป็นเอกสารที่เผยแพร่ไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จากโครงสร้างนี้แสดงให้เห็นว่า lpDestRect ก็คือตำแหน่งที่เราจะวางรูปภาพลงไปนั่นเอง แต่หากค่านี้กำหนดเป็น NULL จะหมายถึงใช้ Surface ปลายทางทั้ง Surface

- lpDDSrcSurf คือ Surface ที่เป็นรูปภาพต้นทาง ซึ่งอาจเป็น OffScreen ก็ได้
- lpSrcRect คือ ตำแหน่งของรูปภาพที่ต้นทาง ถ้าเป็น NULL ใช้ทั้งรูป
- dwFlags คือ เป็นส่วนที่ใช้ควบคุมการวาดภาพ ซึ่งมีค่าดังตารางที่ 4.3

ค่า Flag	ความหมาย
DDBLT_COLORFILL	ใช้ร่วมกับ dwFillColor ในโครงสร้างข้อมูล DDBLTFX โดยใส่เป็นสีรูปแบบ RGB (Red, Green, Blue) ซึ่งจะทำให้การระบายลงบน RECT ที่ surface ปลายทาง
DDBLT_DDFX	ใช้ร่วมกับ dwDDFX ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนดลักษณะพิเศษบางอย่างในการแสดงผล
DDBLT_DDROPS	ใช้ร่วมกับ dwDDROPS ซึ่งกำหนดในโครงสร้างข้อมูล DDBLTFX โดยใช้กับการทำงานขั้นสูง
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	สั่งให้ทำการ blit จนกว่าจะ blit สำเร็จ เมื่อใช้คำสั่งนี้ฟังก์ชัน blit จะไม่ส่งสารข้อผิดพลาดกลับมาถึงแม้จะยังทำการ blit ไม่ได้ก็ตาม

ตารางที่ 4.3 ค่า Control flags ของ Blt()

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

lpDDBltFx คือ ฟังก์ชันที่ใช้ชี้ไปที่โครงสร้างข้อมูล DDBLTFX ซึ่งใช้กำหนดความสามารถพิเศษต่าง ๆ ของฟังก์ชัน (เช่นการหมุนภาพ หรือระบายสี) ซึ่งในโครงสร้างข้อมูล DDBLTFX มีข้อมูลจำนวนมากต้องกำหนด แต่ข้อมูลส่วนใหญ่จะใช้สำหรับการทำงานขั้นสูง เช่น Z-buffer (ใช้กับภาพสามมิติ) ดังนั้นเราจะกล่าวถึงเพียงเท่าที่จำเป็นเท่านั้น

```
typedef struct _DDBLTFX
{
    DWORD    dwSize;
    DWORD    dwDDFX;
    DWORD    dwROP;
    DWORD    dwDDROP;
    DWORD    dwRotationAngle;
    DWORD    dwZBufferOpCode;
    DWORD    dwZbufferLow;
    DWORD    dwZbufferHigh;
    DWORD    dwZBufferBaseDest;
    DWORD    dwZDestConstBitDepth;
    union
    {
        DWORD    dwZDestConst;
        LPDIRECTDRAW SURFACE    lpDDSZBufferDest;
    };
    DWORD    dwZSrcConstBitDepth;
    union
    {
        DWORD    dwzSrcConst;
        LPDIRECTDRAW SURFACE    lpDDSZBufferSrc;
    };
    DWORD    dwAlphaEdgeBlendBitDepth;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

DWORD        dwAlphaEdgeBlend;
DWORD        dwReserved;
DWORD        dwAlphaDestConstBitDepth;
union
{
    DWORD        dwAlphaDestConst;
    LPDIRECTDRAW SURFACE    lpDDSAAlphaDest;
};
DWORD        dwAlphaSrcConstBitDepth;
union
{
    DWORD        dwAlphaSrcConst;
    LPDIRECTDRAW SURFACE    lpDDSAAlphaSrc;
};
union
{
    DWORD        dwFillColor;
    DWORD        dwFillDepth;
    DWORD        dwFillPixel;
    LPDIRECTDRAW SURFACE    lpDDSPattern;
};
DDCOLORKEY    ddckDestColorkey;
DDCOLORKEY    ddckSrcColorkey;
} DDBLTFX.FAR*    LPDDBLTFX;

```

ทั้งหมดนี้คือโครงสร้างข้อมูล DDBLTFX แต่เราจะกำหนดใช้เพียงส่วนน้อยเท่านั้นดังต่อไปนี้

dwSize ขนาดของโครงสร้างข้อมูล DDBLTFX นี้ โดยกำหนดเป็นไบต์
dwFillColor สีที่ต้องการระบายลง surface

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ddckDestColorkey	เป็น color key ของทาง surface ปลายทาง
ddckSrcColorkey	เป็น color key ของทาง surface ต้นทาง
dwRotationAngle	กำหนดเป็นองศาเพื่อจะใช้หมุนภาพ



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

บทที่ 5

การพัฒนาโปรแกรมแสดงภาพ 3 มิติ ด้วย WINx3D

5.1 บทนำ

ในการพัฒนาโปรแกรมแสดงภาพ 3 มิติ แบบ Stereoscopic นั้นต้องมีการเข้าถึงอุปกรณ์ในการดูภาพชนิดนี้ซึ่งก็คือ แว่นตาที่ต้องคอยปิดเปิดภาพซ้ายและขวาให้สัมพันธ์กับการแสดงภาพที่อยู่บนจอคอมพิวเตอร์ โปรแกรมต่าง ๆ ที่สร้างขึ้นมามีทั้ง Software ที่ใช้ช่วยในการติดต่อกับแว่นซึ่งก็อาจมาทั้งกับ Hardware ที่ซื้อเป็นราย ๆ ไป แต่ในปัจจุบันได้มี Software ชนิดหนึ่งที่สามารถติดต่อกับอุปกรณ์แว่นนี้ได้หลายชนิดหลายยี่ห้อ ในการแสดงภาพ 3 มิติแบบนี้ Software ตัวนี้คือ WINx3D ซึ่งเป็นคลังโปรแกรมตัวหนึ่งที่ได้เตรียมเครื่องมือต่าง ๆ ที่จำเป็นในการพัฒนาโปรแกรมในการแสดงภาพ 3 มิติ แบบ Stereoscopic บนพื้นฐานของ DirectX ใน DirectX Application ซึ่งในเวอร์ชันปัจจุบัน คือ เวอร์ชัน 3.0 นั้นสนับสนุนในการพัฒนาภายใต้ระบบปฏิบัติการไมโครซอฟต์วินโดวส์ 95, 98, NT, 2000 และ WindowsME

5.2 ความต้องการของเครื่องที่ใช้ในการพัฒนาโปรแกรมด้วย WINx3D

1. DirectX SDK ตั้งแต่เวอร์ชัน 1 เป็นต้นไป
2. WINx3D ในเวอร์ชัน 3.0 นี้จะสนับสนุน Surface ที่ใช้ในการแสดงภาพ 3 มิติเป็นแบบ full-screen page-flipping ในโหมด exclusive mode เท่านั้น
3. ต้องมีหน่วยความจำที่เพียงพอในการแสดงภาพโดยต้องมีอย่างน้อย 2 Surface ในการแสดงภาพ หนึ่งในนั้นจะเป็น Primary Surface ที่ใช้สำหรับบรรจุภาพซ้ายหรือขวา และ Back Surface ที่ใช้ในการบรรจุภาพที่เหลือ นำมาเพื่อสลับกันเพื่อแสดงในจอคอมพิวเตอร์

5.3 สถาปัตยกรรมของ WINx3D

สถาปัตยกรรมของ WINx3D ประกอบไปด้วย ส่วนหลัก ๆ 3 Layer คือ

- Application Layer
- WINx3D Abstraction Layer
- Hardware Layer

ดังภาพที่ 5.1

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Application	Application and Renderer		
	WINx3D Renderer-Specific Layer		
WINx3D Abstraction Layer	Win3D Stereoscopic Abstraction Module		
	Scheduler		
	Flipper	Toggler	Clocker
Hardware	Video Chip	VR Gear	System Clock

ภาพที่ 5.1 สถาปัตยกรรมของ WINx3D

1. Application Layer เป็นส่วนที่ให้ผู้พัฒนาต่าง ๆ ได้เตรียมภาพ Stereoscopic เพื่อที่จะส่งไปให้ WINx3D แสดงผลต่อไป ซึ่งประกอบด้วยส่วนประกอบ 2 ส่วน
 - 1.1 Application and Renderer เป็นส่วนที่ไว้สำหรับให้ Application ที่ทำหน้าที่ Render ภาพเตรียมภาพก่อนที่จะส่งไปให้ WINx3D แสดงผลต่อไป โดยใน Layer นี้จะเข้าใจ Stereoscopic Images ในแบบที่ Application ต่าง ๆ เตรียมให้ เช่น ถ้า Application เป็น DirectDraw Stereoscopic Images ในที่นี้ก็จะหมายถึง Surface แต่ถ้า Application เป็น OpenGL หรืออย่างอื่น ๆ ก่อนที่จะส่งไปให้ Layer ถัดไปใน Layer นี้มีหน้าที่ต้องแปลงข้อมูล Image เหล่านี้ให้อยู่ใน Format ที่ใช้ได้ ใน Abstraction Layer ก่อน ซึ่ง WINx3D ขณะนี้จะสนับสนุน DirectDraw อย่างเดียว
 - 1.2 WINx3D Renderer-Specific Layer เป็นส่วนที่สามารถที่จะเข้าไปแก้ไขภาพที่ส่งมาจากข้อ 1.1 เพื่อที่จะนำไปใช้เป็น Control Code ในการ Toggler อุปกรณ์ VR ที่ใช้ในการปิดเปิดตาซ้ายหรือขวา
2. WINx3D Abstraction Layer เป็น Layer ถ่างสุดของ Software ที่ใช้ในการควบคุมอุปกรณ์ทาง Hardware ประกอบด้วยส่วนประกอบต่าง ๆ ดังนี้
 - 2.1 WINx3D Stereoscopic Abstraction Module เป็นส่วนที่สามารถรู้ start-addresses บน Video Card อยู่ที่ไหนโดยจะติดต่อกันระหว่าง Renderer-Specific กับ Scheduler
 - 2.2 Scheduler รับผิดชอบในการเข้าจังหวะ (Synchronize) ระหว่างภาพที่อยู่บน Video Card กับ VR Gear โดยจะสื่อสารกับ Flipper โดยการ Set ค่า start address ของภาพ ส่วนกับ Toggler จะเป็นการบอกว่าตาข้างไหนควรจะเปิดหรือปิด และ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กับ Clocker ก็จะเป็นการบอกว่าเมื่อไหร่ถึงจะทำการ Synchronize

2.3 Flipper เป็น Driver ที่ใช้ติดต่อกับ Video Chip ซึ่งมันจะเข้าใจเฉพาะ start address ซึ่ง Driver ต่าง ๆ ก็อาจจะใช้ได้กับ Video Chip ต่าง ๆ กันไป โดยใน WINx3D ก็ จะ Support Video Chip เกือบทุกชนิดในการแสดงภาพแบบ Page Flip

2.4 Toggler เป็น Driver ในการควบคุมอุปกรณ์ VR Gear มันจะเข้าใจเฉพาะตาซ้าย หรือตาขวา และการให้เห็นหรือไม่เห็น ก็คือการให้ปิดเปิดตาซ้ายหรือขวานั้นเอง ซึ่ง VR Gear ก็มีมากมายหลายชนิด และก็มี Driver เฉพาะของแต่ละชนิด ซึ่งใน การใช้งาน WINx3D ก็ต้องเลือกอุปกรณ์ VR Gear ให้ถูกต้องเพื่อให้ได้ประสิทธิภาพ ในการดูภาพ Stereoscopic Image ที่ดีที่สุด

2.5 System Clock เป็นตัวที่ติดต่อกับเวลาของเครื่อง

3. Hardware Layer เป็นชั้นของ Hardware จริง ๆ ซึ่งประกอบด้วยส่วนประกอบ 3 ส่วน

3.1 Video Chip ซึ่งก็แล้วแต่ว่าจะเป็นของค่ายใด เช่น ค่าย nVidia หรือ S3

3.2 VR Gear เป็นอุปกรณ์ที่ใช้ในการดูภาพ Stereoscopic

3.3 System Clock นาฬิกาของเครื่อง

5.4 ส่วนประกอบภายในคลังโปรแกรม WINx3D

ประกอบด้วย 4 ส่วนหลัก ๆ แยกตามหน้าที่ของแต่ละส่วนดังนี้

1. WINx3D API เป็นส่วนที่สำคัญที่สุดเพราะเป็นกลุ่มฟังก์ชันที่ใช้ในการติดต่อกับ อุปกรณ์ VR ต่าง ๆ และส่วนหนึ่งจะเป็นกลุ่มฟังก์ชันที่ใช้ในการ Interface กับ DirectDraw เพื่อสร้าง W3D Object ที่เป็นพื้นที่ใช้ในการเก็บภาพในการแสดงผล
2. SETUP API เป็นกลุ่มของฟังก์ชันที่ใช้ในการติดตั้งโปรแกรมเริ่มต้นเพื่อนำไปใช้งาน ในเครื่องอื่น ๆ เช่น ฟังก์ชันในการ Copy File, ฟังก์ชันในการ Setup ค่าเริ่มต้นของการ ใช้งาน, ฟังก์ชันในการติดต่อกับอุปกรณ์และ Setup ว่าควรใช้อุปกรณ์ใด เป็นต้น
3. INTERLACE CONTROL เป็นกลุ่มของฟังก์ชันที่ใช้ในการแสดงภาพแบบ Interlace Mode ที่ใช้ได้กับเฉพาะบาง Video Chip เท่านั้น
4. HELPER FUNCTIONS เป็นกลุ่มของฟังก์ชันที่ใช้ช่วยในการทำประโยชน์อื่น ๆ ที่ สามารถเรียกใช้ใน Win32 ได้แต่ถ้าเรียกผ่านฟังก์ชันนี้ก็จะง่ายขึ้น เช่น ฟังก์ชันในการ อ่าน, เขียน และลบ Registry ของวินโดวส์ หรือ ในการ Save และ Restore ค่า Screen Setting ก่อนที่จะเข้าไปในโหมด 3 มิติ

บทที่ 6

ขั้นตอนการพัฒนาโปรแกรม

6.1 การพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 2 มิติ

ประกอบไปด้วยหน้าที่ดังต่อไปนี้

- การเปิดภาพจากแฟ้มข้อมูลนามสกุล BMP, JPG และ PNG โดยอาจเป็นภาพเดี่ยวที่เป็นภาพจากตาซ้ายหรือภาพจากตาขวา หรือเป็นภาพคู่สเตอริโอที่มี Layout ชนิดต่าง ๆ นำมาแสดงบนกรอบภาพซ้ายและขวา
- การบันทึกภาพลงแฟ้มข้อมูลนามสกุล BMP, JPG และ PNG ได้โดยสามารถบันทึกเป็นภาพเดี่ยวที่เป็นภาพจากตาซ้ายหรือภาพจากตาขวา หรือบันทึกเป็นภาพคู่สเตอริโอที่มี Layout ชนิดต่าง ๆ ได้
- การดึงข้อมูลของความละเอียดและจำนวนสีที่จะใช้ในหน้าจอในแต่ละเครื่องมาแสดงและให้เลือกเพื่อที่จะนำไปแสดงผลในโหมด 3 มิติ
- การที่สามารถสลับภาพซ้ายและภาพขวาในการกรณีที่เลือกภาพซ้ายและขวาสลับกัน

ในการเปิดและบันทึกภาพนามสกุล BMP, JPG และ PNG นั้นได้นำ Software ของบริษัท Colosseum Builders มาใช้เป็นฟังก์ชัน ในการเรียกใช้เปิดและบันทึกภาพนามสกุลต่าง ๆ ข้างต้น โดยจะใช้ฟังก์ชัน ReadImage ในการเปิดภาพดังนี้

```
extern ImageType ReadImage (const std::string &,  
    BitmapImage &image,  
    PROGRESSFUNCTION function,  
    void *data) ;
```

ซึ่งฟังก์ชันนี้จะหมายถึงให้ส่ง Path ที่เก็บแฟ้มที่ต้องการเปิดไปและจะนำข้อมูลที่อ่านได้จากแฟ้มนั้นไว้ที่ตำแหน่งแอดเดรส image ซึ่งเป็น class ของ Colosseum Builders ชื่อ BitmapImage ใช้เก็บข้อมูลของภาพที่เปิดได้นั้น โดยข้อมูลของภาพที่เก็บจะเป็น Stream Pixel ของภาพในรูปแบบ BGRA ดังนี้

```

struct Pixel
{
    // The BGRA order is assumed in the BMP encoder
    UBYTE1 blue ;
    UBYTE1 green ;
    UBYTE1 red ;
    UBYTE1 alpha ;
};

```

ส่วน Class BitmapImage จะเก็บว่าภาพที่เปิดได้มีความกว้างและสูงเท่าไรโดยสามารถดึงค่าได้โดยเรียกใช้ฟังก์ชัน getWidth เพื่อดึงข้อมูลความกว้างของภาพ และฟังก์ชัน getHeight เพื่อดึงข้อมูลความสูงของภาพ ฟังก์ชัน ReadImage นี้จะทำงานสำเร็จหรือไม่นั้น จะแจ้งผ่านค่า ImageType โดย ImageType เป็นชนิดข้อมูลชนิดหนึ่งของ Colosseum Builders ซึ่งจะบอกถึงประเภทของแฟ้มภาพที่เปิดได้ โดยจะมีค่าต่าง ๆ ดังนี้

```

enum ImageType
{
    UnknownImage=0,
    BmpImage = 2,
    JpegImage = 3,
    PngImage = 7, };

```

และการบันทึกภาพลงแฟ้มข้อมูลภาพนามสกุล BMP, JPG และ PNG นั้นจะใช้ฟังก์ชันดังนี้

```
void writeImageFile (const std::string &, const BitmapImage &)
```

ซึ่งจะต้องมีการส่งที่อยู่ของแฟ้มข้อมูลที่ต้องการบันทึก และ Object ของ class BitmapImage ซึ่งมีการ Set ค่าความกว้างและความสูงของภาพลงใน Object ของ Class นั้นแล้ว ก่อนที่จะมีการเรียกใช้ฟังก์ชัน writeImageFile

ในการเปิดภาพเพื่อแสดงผลเป็น 3 มิติแบบ Stereoscopic Image ต้องมีข้อมูลภาพจากตาซ้ายและขวา ซึ่งอาจจะดึงข้อมูลภาพตาซ้ายหรือขวาจากแฟ้มข้อมูลคนละแฟ้ม แต่ถ้าข้อมูลภาพตาซ้ายและขวายู่ในแฟ้มเดียวกันซึ่งมี Layout ที่เก็บในแฟ้มนั้นเป็นแบบ Stereoscopic ในโปรแกรมที่พัฒนาขึ้นมานี้ก็จะสนับสนุนในการดึงข้อมูลจากแฟ้มประเภทนี้โดยเราจะกล่าวถึงก่อนว่ารายละเอียดของแฟ้มข้อมูลประเภท Stereoscopic Image นี้เป็นอย่างไรดังนี้

6.1.1 แฟ้มข้อมูลประเภท Stereoscopic Image

แฟ้มข้อมูลประเภท Stereoscopic Image เป็นการจัดเก็บข้อมูลภาพถ่ายและขวาลงในแฟ้มข้อมูลเดียวกัน เพื่อประโยชน์ในการเปิดข้อมูลภาพ Stereoscopic Image เพราะจัดเก็บไว้ในแฟ้มเดียว ทำให้ง่ายในการเปิดข้อมูลโดยไม่ต้องเปิด 2 ครั้งจากแฟ้มข้อมูลภาพถ่ายและขวา โดยมี Layout ที่จัดเก็บลงในแฟ้มข้อมูลประเภทนี้ดังนี้

1. Side-by-side

เป็นการจัดเก็บข้อมูลลงแฟ้มโดยให้ข้อมูลภาพถ่ายและขวาอยู่กันคนละด้านทางแนวนอนซ้ายหรือขวา ดังรูปที่ 6.1



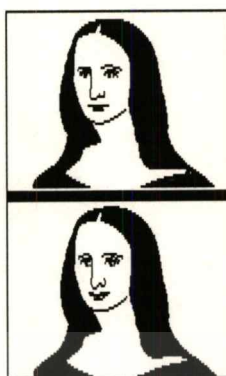
ภาพที่ 6.1 แสดง Layout File Stereoscopic แบบ Side-by-Side

ซึ่งสามารถแบ่งได้เป็นดังนี้

- Parallel (Left Right) หมายถึงภาพถ่ายจะอยู่ด้านซ้ายและภาพขวาจะอยู่ด้านขวา
- Cross-Eyed (Right Left) หมายถึงภาพถ่ายและขวาจะอยู่สลับกันคนละด้าน โดยภาพถ่ายจะอยู่ด้านขวาและภาพขวาจะอยู่ด้านซ้าย

2. Over-Under

เป็นการจัดเก็บข้อมูลลงแฟ้มโดยให้ข้อมูลภาพถ่ายและขวาอยู่กันคนละด้านทางแนวตั้งบนหรือล่าง ดังรูปที่ 6.2



ภาพที่ 6.2 แสดง Layout File Stereoscopic แบบ Over-Under

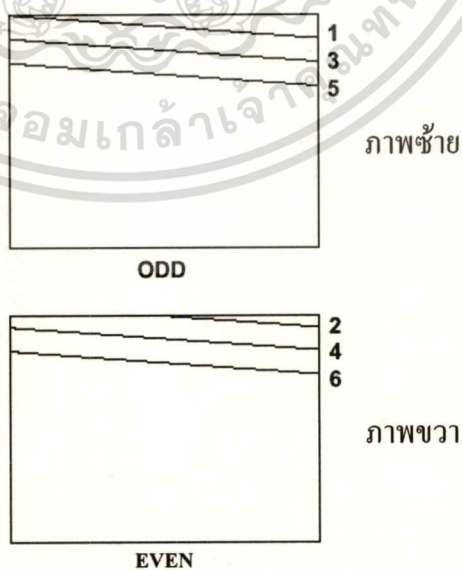
ซึ่งสามารถแบ่งได้เป็นดังนี้

- Left Over Right หมายถึงภาพซ้ายจะอยู่บนและภาพขวาจะอยู่ด้านล่าง
- Right Over Left หมายถึงภาพขวาจะอยู่บนและภาพซ้ายจะอยู่ด้านล่าง

3. Interlaced

เป็นการจัดเก็บข้อมูลลงแฟ้มโดยให้ข้อมูลภาพซ้ายและขวายู่กันคนละเส้นทางแนวนอนสลับกันไปโดยจะให้ภาพขวายู่เส้นคู่และภาพซ้ายอยู่เส้นคี่ ดังรูปที่

6.3



ภาพที่ 6.3 แสดง Layout File Stereoscopic แบบ Interlaced

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

6.1.2 การพัฒนาโปรแกรมในการดึงข้อมูลจากแฟ้มข้อมูลประเภท Stereoscopic Image

เมื่อเราสามารถดึงข้อมูลจากแฟ้มข้อมูลภาพชนิดต่าง ๆ ได้โดยผ่านฟังก์ชัน ReadImage ของ Colosseum Builders แล้วและได้ข้อมูลมาอยู่ในรูปโครงสร้าง Pixel ซึ่งต่อ 1 Pixel จะเก็บ 32 บิต ถ้าเป็นแฟ้มข้อมูลที่เก็บภาพซ้ายหรือภาพขวาภาพเดียวใน 1 แฟ้มข้อมูลเราก็สามารถนำมาแสดงผลลงบนหน้าจอหรือเฟรมต่าง ๆ ได้เลย แต่ถ้าแฟ้มข้อมูลนั้นเก็บข้อมูลภาพ 2 ภาพในแฟ้มเดียวกันต้องมีการแยกข้อมูลภาพทั้งสองออกจากข้อมูลภาพที่เราอ่านมาโดยสามารถแยกได้ตาม Layout File Stereoscopic ดังนี้

1. Side-by-side

เนื่องจากการจัดเก็บข้อมูลลงแฟ้มโดยให้ข้อมูลภาพซ้ายและขวายู่ติดกันทางแนวนอนทำให้ความกว้างของภาพในแฟ้มข้อมูลเท่ากับ 2 เท่าของภาพ 1 ภาพในการแยกภาพเราก็จะทำการอ่านภาพทีละเส้นโดยถ้าต้องการภาพทางด้านซ้ายเราก็จะอ่านทีละเส้นไปจนครึ่งหนึ่งของความกว้างของภาพในแฟ้มข้อมูล และทำการอ่านจนหมดความสูงของภาพ ก็จะได้ภาพทางด้านซ้ายออกมา เช่นเดียวกันถ้าต้องการข้อมูลภาพทางด้านขวาก็จะเริ่มอ่านทีละเส้นจากกึ่งกลางภาพไปเท่ากับครึ่งหนึ่งของความกว้างของภาพในแฟ้มข้อมูล ก็จะได้ภาพทางด้านขวาออกมา

2. Over-Under

เนื่องจากการจัดเก็บข้อมูลลงแฟ้มโดยให้ข้อมูลภาพซ้ายและขวายู่ติดกันด้านทางแนวตั้ง ทำให้ความสูงของภาพในแฟ้มข้อมูลเท่ากับ 2 เท่าของภาพ 1 ภาพในการแยกภาพเราก็จะทำการอ่านภาพไปเท่ากับความกว้างของภาพในแฟ้มข้อมูลคูณด้วยครึ่งหนึ่งของความสูงของภาพในแฟ้มข้อมูล ซึ่งถ้าต้องการภาพบนก็จะทำการอ่านจากต้นของแฟ้มข้อมูล แต่ถ้าต้องการภาพล่างก็จะเริ่มต้นทำการอ่านจากครึ่งหนึ่งของภาพไปจนสิ้นสุดภาพ

3. Interlaced

เนื่องจากการจัดเก็บข้อมูลลงแฟ้มโดยให้ข้อมูลภาพซ้ายและขวายู่กันคนละเส้นทางแนวนอนสลับกันไปโดยจะให้ภาพขวายู่เส้นคู่และภาพซ้ายอยู่เส้นคี่ เมื่อเราแยกภาพซ้ายและขวาออกมาก็จะทำให้ความสูงของภาพลดลงครึ่งหนึ่งทำให้เราต้องทำการคัดลอกข้อมูลนั้นลงเส้นถัดไปเส้นต่อเส้นของการอ่านทีละเส้น

6.1.3 การพัฒนาโปรแกรมในการบันทึกข้อมูลลงเพิ่มข้อมูลประเภท Stereoscopic Image

จากการที่เราได้อ่านข้อมูลภาพได้เป็นข้อมูลภาพซ้ายและขวานั้น ถ้าเราต้องการที่จะบันทึกข้อมูลภาพซ้ายและขวานั้นลงเพิ่มข้อมูลประเภท Stereoscopic Image ซึ่งเป็นเพิ่มข้อมูลที่รวมภาพซ้ายและขวาเข้าด้วยกันใน Layout ต่าง ๆ ของเพิ่มข้อมูล ซึ่งจะมีวิธีต่าง ๆ ตาม Layout File ดังนี้

1. Side-by-side

จะเป็นการรวมภาพซ้ายและขวาตามแนวนอน ซึ่งเราต้องรู้ว่าจะให้ภาพใดอยู่ทางด้านใด และทำการต่อภาพนั้นทางแนวนอนโดยต้องดึงข้อมูลจากแต่ละภาพที่ละเส้นต่อกันจะหมดความสูงของภาพทั้งสอง

2. Over-Under

จะเป็นการรวมภาพซ้ายและขวาตามแนวตั้ง ซึ่งเราต้องรู้ว่าจะให้ภาพใดอยู่บนหรือล่าง และทำการต่อภาพนั้นทางแนวตั้ง โดยจะทำการอ่านข้อมูลภาพ 1 ภาพมาต่อด้วยอีกหนึ่งภาพนั่นเอง

3. Interlaced

จะเป็นการรวมภาพซ้ายและขวาตามเส้นแนวนอนที่ละเส้นสลับกันไป โดยให้เส้นคู่เป็นภาพขวาและเส้นคี่เป็นภาพซ้าย ซึ่งก็ต้องอ่านข้อมูลภาพขวาและซ้ายที่ละเส้นนำมารวมกันเป็นข้อมูลของเพิ่มภาพ Stereoscopic เพียงภาพเดียว

6.1.4 การพัฒนาโปรแกรมในการดึงข้อมูลของความละเอียดและจำนวนสีของหน้าจอ

ในการดึงข้อมูลของความละเอียดและจำนวนสีที่จะใช้ในหน้าจอในแต่ละเครื่องมาแสดงและให้เลือกเพื่อที่จะนำไปแสดงผลในโหมด 3 มิตินั้นจะใช้ฟังก์ชัน EnumDisplayModes ของ DirectX เป็นตัวดึงข้อมูลทั้งหมด

```
HRESULT EnumDisplayModes(
    DWORD                dwFlags,
    LPDDSURFACEDESC2    lpDDSurfaceDesc2,
    LPVOID                lpContext,
    LPDDENUMMODESCALLBACK2 lpEnumModeCallback
);
```

โดยในที่นี้ต้องการดึงข้อมูล Refresh Rate ของหน้าจอเพราะฉะนั้นค่าของ dwFlags จะเท่ากับ DDEDM_REFRESHRATES, lpDDSurfaceDesc2 เท่ากับ Null เพื่อดึงค่าที่สามารถมีได้ทั้งหมด, lpContext คือแอดเดรสตัวแปรที่ใช้เก็บค่าความละเอียดและจำนวนสีทั้งหมด และ lpEnumModeCallback คือแอดเดรสของฟังก์ชันที่จะเข้าทุกครั้งที่เมื่อเกิดการ Enumeration อาจใช้สำหรับการ Filter ความละเอียดที่ไม่ต้องการก็ได้

6.2 การพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 3 มิติ

ประกอบไปด้วยหน้าที่ดังต่อไปนี้

- การนำภาพที่เปิดจากแฟ้มข้อมูลในโหมด 2 มิติมาแสดงผลในโหมด 3 มิติได้
- การสร้าง Interface บางอย่างกับผู้ใช้ให้รู้สึกว่ามีทางเลือก เช่น ปุ่มกด หรือกรอบของรูปภาพ
- สามารถที่จะสลับเปลี่ยนให้ภาพที่แสดงในกรอบ นำไปแสดงในเต็มหน้าจอ (Full Screen) ได้
- ย่อหรือขยายภาพได้

หลังจากที่ได้รับข้อมูลภาพซ้ายและขวาที่เปิดจากแฟ้มข้อมูลใน โหมด 2 มิติแล้ว เราจะเก็บข้อมูลนั้นไว้ในตัวแปรโครงสร้างที่สร้างขึ้นโดยมีรูปแบบดังนี้

```
typedef struct
{
    char          szFileName[_MAX_PATH]; // Path ที่เก็บไฟล์
    BITMAPINFOHEADER bmiHeader; // ข้อมูลรายละเอียดของภาพ
    BYTE          * pbImage; // ส่วนข้อมูลภาพ }

```

โดยขั้นตอนต่อไปจะเป็นการสร้าง Object ของ WINx3D และติดต่อกับอุปกรณ์ เพื่อที่จะส่งภาพและแสดงในโหมด 3 มิติได้โดยมีขั้นตอนที่ต้องทำดังต่อไปนี้

1. สร้าง Direct Draw Object โดยใช้ฟังก์ชันของ DirectX ดังนี้

```
hRetval = DirectDrawCreate(NULL, &lpDD, NULL);
```

2. สร้าง Object WINx3D โดยใช้ฟังก์ชันของ WINx3D ดังนี้

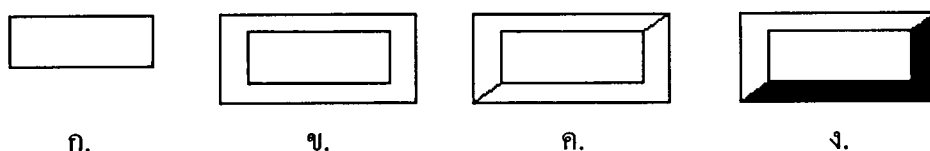
```
HRESULT W3D_DDCreate(LPW3D_DDOBJECT *lpW3D_DD)
```

ซึ่ง lpW3D_DD คือแอดเดรสของ Object WINx3D ที่ถูก Define ไว้ใน w3d_dd.h ซึ่งต้อง Include ไว้ใน File ที่เรียกฟังก์ชันนี้ เมื่อสร้างสำเร็จจะคืนค่ากลับมาเท่ากับ W3D_OK ซึ่งหมายความว่าสร้างสำเร็จ

3. กำหนดความสำคัญของโปรแกรมที่ขอจากวินโดวส์ โดยใช้ฟังก์ชันของ DirectX ดังนี้
`lpDD->lpVtbl->SetCooperativeLevel(lpDD, hWnd, DDSCL_FULLSCREEN | DDSCL_EXCLUSIVE);`
4. กำหนดโหมดการแสดงผล โดยดึงมาจากขณะที่เลือกในโหมด 2 มิติ และใช้ฟังก์ชันของ DirectX ดังนี้
`hRetval = lpDD->lpVtbl->SetDisplayMode(lpDD, gdwWidth, gdwHeight, gdwBpp);`
5. สร้าง Surface โดยใช้ฟังก์ชันของ DirectX ดังนี้
`HRESULT CreateSurface(LPDDSURFACEDESC lpDDSurfaceDesc,
LPDIRECTDRAW_SURFACE FAR *lpDDSsurface,
IUnknown FAR *pUnkOuter);`
6. Initialize WINx3D โดยใช้ฟังก์ชันดังนี้
`hRetval = lpW3D_DD->lpVtbl->Initialize(lpW3D_DD,
gdwIndex, lpDD, lpSurfaces[0], lpSurfaces[1]);`
7. Render ภาพซ้ายและขวาโดยนำใส่ Surface ของ DirectDraw เพื่อเตรียมส่ง WINx3D ต่อไป
8. Queue ภาพซ้ายและขวา โดยใช้ฟังก์ชันของ WINx3D ดังนี้
`lpW3D_DD->lpVtbl->Queue(lpW3D_DD, gdwIndex, lpSurfaces[dwLeftSurface],
FALSE);`
9. Display ภาพ 3 มิติ โดยใช้ฟังก์ชันของ WINx3D ดังนี้
`hRetval = lpW3D_DD->lpVtbl->Display(lpW3D_DD, gdwIndex, TRUE);`

6.2.1 การพัฒนาโปรแกรมในส่วนของการสร้าง Interface 3 มิติ

ส่วนของโปรแกรมสำหรับการสร้าง Interface 3 มิติ ในโครงงานฉบับนี้จะเป็นการสร้างปุ่มกด 3 มิติ และ กรอบ 3 มิติ โดยมีขั้นตอนดังนี้



ภาพที่ 6.4 แสดง ขั้นตอนการสร้างปุ่มกด 3 มิติ

การสร้างปุ่มกดจะเป็นการนำกรอบสี่เหลี่ยม 2 อันมาซ้อนกันและระบายสีแสงและเงาโดยต้องกำหนดแหล่งกำเนิดแสงเพื่อสร้างแสงและเงา โดยในที่นี้จะกำหนดให้แหล่งกำเนิดแสงเกิดจากมุมบนซ้ายของจอ เพราะฉะนั้นก็จะระบายสีของแสงที่ตกกระทบเป็นสีขาว โดยจะระบายที่พื้นที่ระหว่างกรอบในกับกรอบนอกของพื้นที่ส่วนบนกับพื้นที่ส่วนทางด้านซ้ายเป็นสีขาว และระบายเงาของปุ่มทางพื้นที่ส่วนล่างกับพื้นที่ทางด้านขวาเป็นสีดำ จากภาพที่ 6.4 (ก.) เป็นการกำหนดกรอบสำหรับพื้นที่ของปุ่มที่ใช้กด ภาพที่ 6.4 (ข.) เป็นการสร้างกรอบนอกมาซ้อนทับกรอบใน เมื่อทำการกำหนดจุดต่าง ๆ ของกรอบเสร็จ ก็จะทำให้นำจุดของกรอบเหล่านี้ไปทำการ Map จากระนาบ XYZ เป็นระนาบ XY โดยใช้สูตรการฉายภาพในบทที่ 3.1.1 ภาพที่ 6.4 (ค.) และ 6.4 (ง.) เป็นการกำหนดพื้นที่ที่จะระบายสีและการระบายสีขาวและดำเพื่อให้แสงและเงาเพื่อให้ปุ่มมองดูเป็น 3 มิติ

ส่วนปุ่มเมื่อเกิดการกดแล้วก็จะเป็นการระบายสีสลับกัน โดยจะระบายสีขาวที่พื้นที่ส่วนขวาและพื้นที่ส่วนล่างของกรอบปุ่ม และระบายสีดำที่พื้นที่ส่วนบนและพื้นที่ส่วนซ้ายของปุ่มเพื่อให้ดูเหมือนปุ่มเกิดการยุบตัวลงไปดังภาพที่ 6.5 (ก.) และสำหรับการสร้างกรอบบน 3 มิติ ก็เกิดจากการสร้างปุ่มขึ้นมาซ้อนข้างในปุ่มบนดังภาพที่ 6.5 (ข.)



ภาพที่ 6.5 แสดง ปุ่มเมื่อถูกกด กับ กรอบภาพแบบบน

บทที่ 7

ผลการพัฒนาโปรแกรม

7.1 ภาพรวมของการพัฒนาโปรแกรม

โปรแกรมที่พัฒนาจะแบ่งออกเป็น 2 ส่วนใหญ่ คือ ส่วนที่แสดงในโหมด 2 มิติ กับส่วนที่ต้องแสดงในโหมด 3 มิติ

1. ส่วนที่แสดงในโหมด 2 มิติ จะประกอบด้วย
 - การเปิดภาพจากแฟ้มข้อมูลชนิดต่าง ๆ แล้วนำมาแสดงให้เห็นบนกรอบภาพ 2 กรอบ
 - การดึงข้อมูลของความละเอียดของหน้าจอในแต่ละเครื่องมาแสดงและให้เลือกความละเอียดของหน้าจอที่จะนำไปแสดงผลในโหมด 3 มิติ
 - การที่สามารถสลับภาพซ้ายและขวาในการเลือกภาพซ้ายและขวาที่ผิดได้
2. ส่วนที่แสดงในโหมด 3 มิติ จะประกอบด้วย
 - การนำภาพที่เปิดจากแฟ้มข้อมูลมาแสดง
 - การสร้าง Interface บางอย่างกับผู้ใช้ให้รู้สึกว่ามีความลึก เช่น ปุ่มกด หรือกรอบของรูปภาพ
 - สามารถที่จะเปลี่ยนให้ภาพสามารถแสดงในโหมดเต็มหน้าจอได้
 - ย่อ / ขยาย ภาพได้

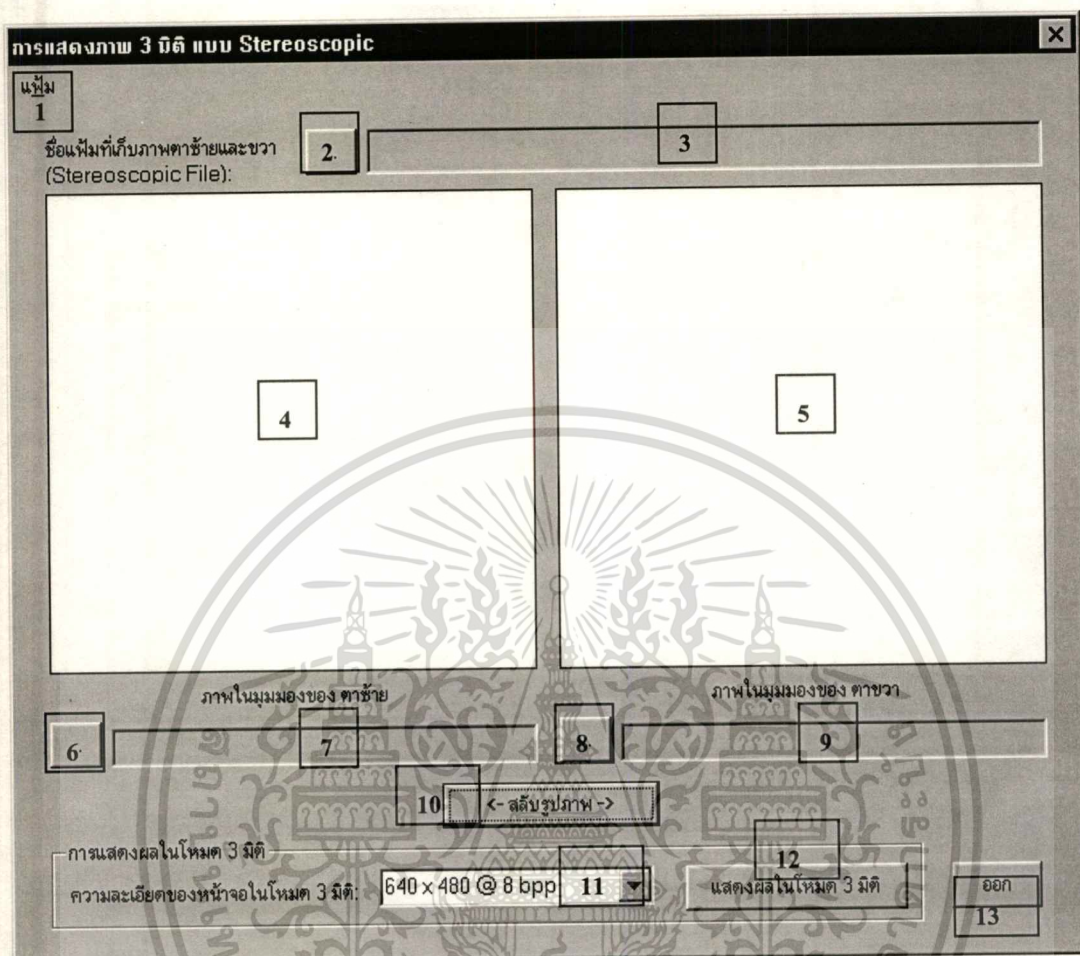
7.2 ผลการพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 2 มิติ

โดยมีส่วนประกอบดังต่อไปนี้

7.2.1 หน้าจอหลัก

ซึ่งเป็นหน้าแรกที่จะเปิดขึ้นมาเมื่อมีการเรียกใช้โปรแกรม ซึ่งประกอบไปด้วย เมนูแฟ้มที่ใช้ในการเปิดภาพที่ใช้ในการแสดงผลลงบน Frame สีขาวทั้งสอง โดยมี Label ติดอยู่ข้างล่างว่าเป็น Frame ของภาพใด, ปุ่มที่ใช้ในการสลับภาพ และ Combo Box ที่ใช้เลือกความละเอียดของจอภาพที่จะใช้ในการแสดงผลในโหมด 3 มิติ โดยมีหน้าตา ดังภาพที่ 7.1

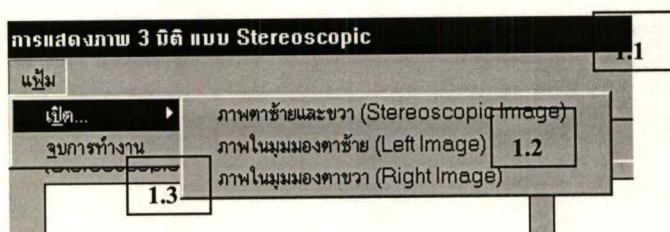
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 7.1 หน้าจอหลัก

จากภาพที่ 7.1 จะมีหมายเลขกำกับส่วนประกอบต่างๆ ที่จะอธิบายดังนี้

หมายเลข 1 : เมื่อกดก็จะแสดงเมนูลูก 2 อย่างคือ เปิด กับ จบการทำงาน ซึ่งเมนูเปิด ใช้สำหรับในการเปิดแฟ้มข้อมูลภาพต่าง ๆ เมื่อกดเมนูเปิด จะแสดงดังภาพที่ 7.2



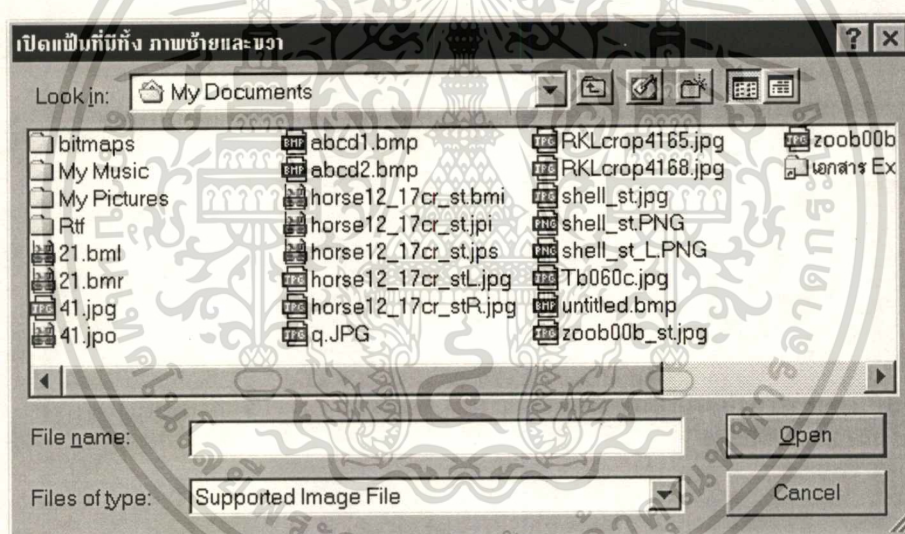
ภาพที่ 7.2 เมนูการเปิดแฟ้มข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ซึ่งเมนูลูกที่แสดงขึ้นมาจะมีการให้เปิดเพิ่มข้อมูล 3 ประเภทคือ

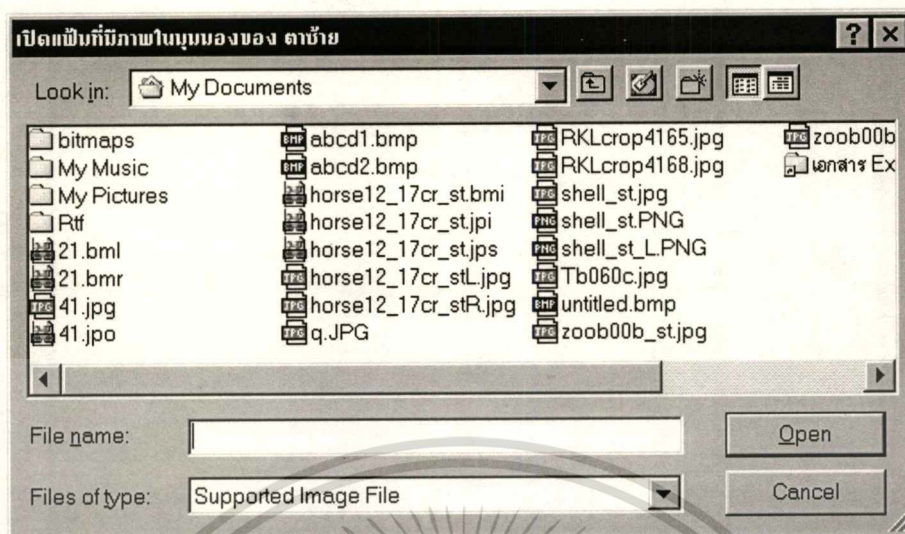
- 1.1 ภาพตาซ้ายและขวา (Stereoscopic Image) ซึ่งเป็นประเภทเพิ่มข้อมูลที่ข้อมูลภาพในมุมมองของตาซ้ายและขวาวจะอยู่เพิ่มข้อมูลเดียวกัน
- 1.2 ภาพในมุมมองตาซ้าย (Left Image) หมายถึง เพิ่มข้อมูลที่เก็บภาพที่อยู่มุมมองของตาซ้าย
- 1.3 ภาพในมุมมองตาขวา (Right Image) หมายถึง เพิ่มข้อมูลที่เก็บภาพที่อยู่มุมมองของตาขวา

และเมื่อเลือกเมนูลูกหมายเลข 1.1 ก็จะแสดง Dialog box ให้เลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาซ้ายและขวาดังภาพที่ 7.3



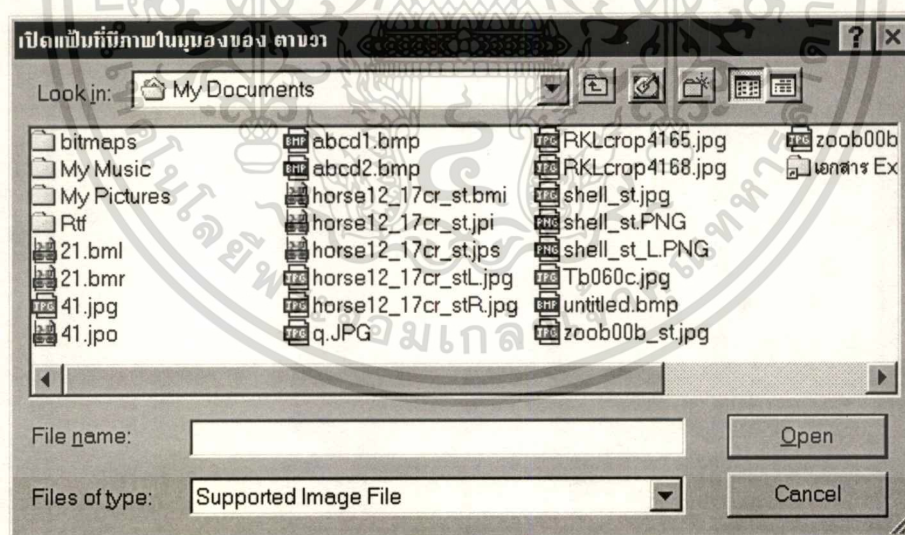
ภาพที่ 7.3 Dialog box การเปิดเพิ่มข้อมูลภาพในมุมมองของตาซ้ายและขวา

และเมื่อเลือกเมนูลูกหมายเลข 1.2 ก็จะแสดง Dialog box ให้เลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาซ้ายดังภาพที่ 7.4



ภาพที่ 7.4 Dialog box การเปิดแฟ้มข้อมูลภาพในมุมมองของตาซ้าย

และเมื่อเลือกเมนูลูกหมายเลข 1.3 ก็จะแสดง Dialog box ให้เลือกแฟ้มข้อมูลที่มีข้อมูลภาพในมุมมองของตาขวาดังภาพที่ 7.5



ภาพที่ 7.5 Dialog box การเปิดแฟ้มข้อมูลภาพในมุมมองของตาขวา

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายเลข 2 : เป็นปุ่มที่เมื่อกดก็จะแสดง Dialog box ให้เลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาซ้ายและขวาดังภาพที่ 7.3

หมายเลข 3 : เป็นกล่องที่ใช้แสดงข้อความ ชื่อเพิ่มข้อมูลที่เปิดเมื่อเลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาซ้ายและขวาโดยเลือกผ่านทาง Dialog box ของภาพที่ 7.3

หมายเลข 4 : เป็นพื้นที่ในการแสดงภาพในมุมมองของตาซ้าย เมื่อเลือกเพิ่มข้อมูลที่มีภาพในมุมมองของตาซ้ายแล้ว

หมายเลข 5 : เป็นพื้นที่ในการแสดงภาพในมุมมองของตาขวา เมื่อเลือกเพิ่มข้อมูลที่มีภาพในมุมมองของตาขวาแล้ว

หมายเลข 6 : เป็นปุ่มที่เมื่อกดก็จะแสดง Dialog box ให้เลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาซ้ายดังภาพที่ 7.4

หมายเลข 7 : เป็นกล่องที่ใช้แสดงข้อความ ชื่อเพิ่มข้อมูลที่เปิดเมื่อเลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาซ้ายโดยเลือกผ่านทาง Dialog box ของภาพที่ 7.4

หมายเลข 8 : เป็นปุ่มที่เมื่อกดก็จะแสดง Dialog box ให้เลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาขวาดังภาพที่ 7.5

หมายเลข 9 : เป็นกล่องที่ใช้แสดงข้อความ ชื่อเพิ่มข้อมูลที่เปิดเมื่อเลือกเพิ่มข้อมูลที่มีข้อมูลภาพในมุมมองของตาขวาโดยเลือกผ่านทาง Dialog box ของภาพที่ 7.5

หมายเลข 10 : เป็นปุ่มที่เมื่อกด จะทำการสลับภาพที่อยู่ทางด้านซ้ายและขวา โดยให้ภาพที่อยู่ทางซ้ายไปอยู่ทางขวา และภาพที่อยู่ทางขวาไปอยู่ทางซ้ายแทน

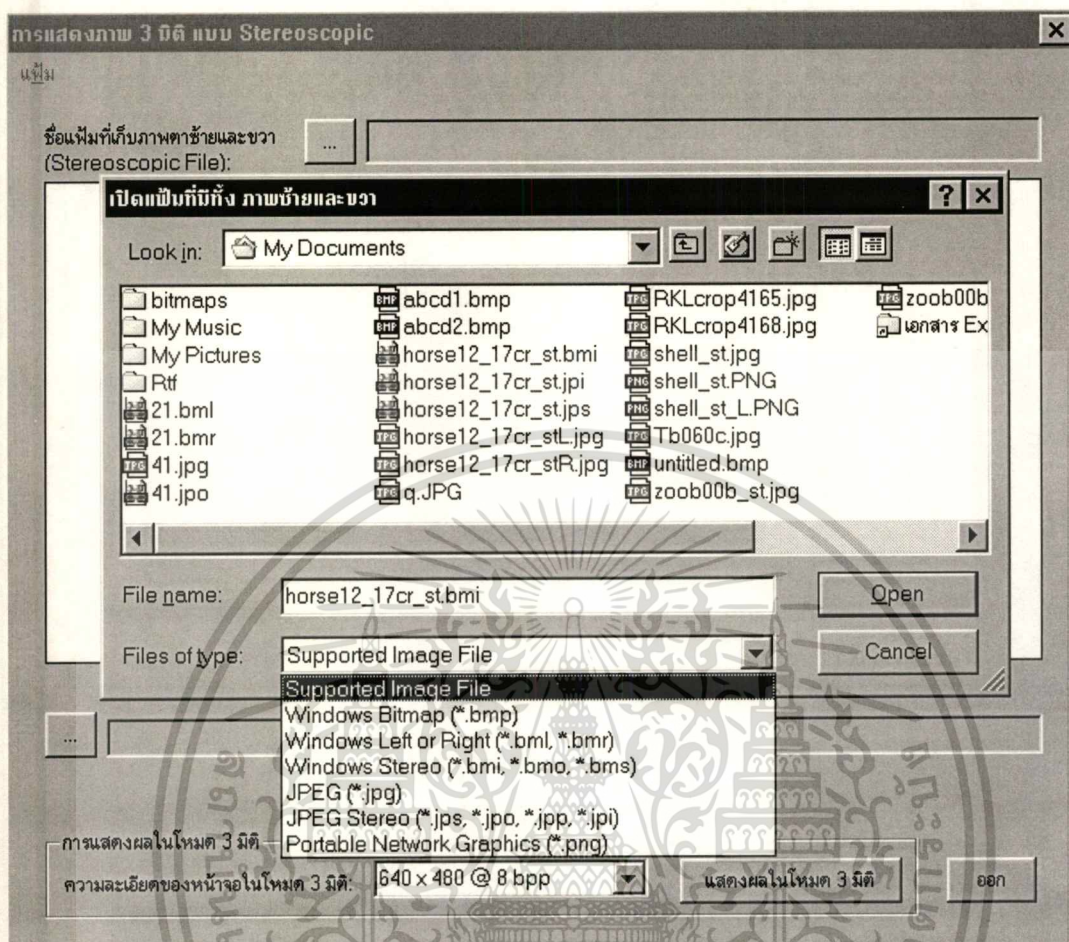
หมายเลข 11 : เป็น Drop Down ที่ใช้เลือกความละเอียดของหน้าจอที่จะแสดงในโหมด 3 มิติ สเตอริโอ โดย List ที่อยู่ใน Drop Down จะเป็นความละเอียดที่สามารถใช้ได้ในการแสดงผลโหมด 2 มิติ

หมายเลข 12 : เป็นปุ่มที่เมื่อกด จะทำนำภาพที่เลือกไปแสดงในโหมด 3 มิติ

หมายเลข 13 : เป็นปุ่มที่เมื่อกด จะทำการออกจากโปรแกรม

ประเภทของเพิ่มข้อมูลภาพที่สนับสนุน

ประเภทเพิ่มข้อมูลที่สนับสนุนจะแบ่งเป็น เพิ่มข้อมูลใน Format ธรรมดา กับ เพิ่มข้อมูลใน Format Stereo ซึ่งแสดงได้ดังภาพที่ 7.6



ภาพที่ 7.6 เพิ่มข้อมูลที่สนับสนุน

จากภาพที่ 7.5 สามารถแบ่งเพิ่มข้อมูลที่สนับสนุนได้เป็น

- เพิ่มข้อมูลประเภท Bitmap (BMP)
- เพิ่มข้อมูลประเภท Jpeg (JPG)
- เพิ่มข้อมูลประเภท Portable Network Graphic (PNG)

และยังสามารถแบ่งเพิ่มข้อมูลในแต่ละประเภทตาม Layout ของภาพ Stereoscopic ได้ดังนี้

1. Side-by-Side, Parallel (Left Right)

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพ 2 ภาพมาต่อกันทางแนวนอน โดยภาพทั้งสองจะอยู่คนละข้าง ซึ่งใน Layout นี้ภาพซ้ายจะอยู่ทางซ้าย และภาพทางขวาก็จะอยู่ทางขวา

2. Side-by-Side, Cross-Eyed (Right Left)

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพ 2 ภาพมาต่อกันทางแนวนอน โดยภาพทั้งสองจะอยู่คนละข้าง ซึ่งใน Layout นี้ภาพขวาจะอยู่ทางซ้าย และภาพทางซ้ายก็จะอยู่ทางขวา

3. Left over Right

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพ 2 ภาพมาต่อกันทางแนวนอน ซึ่งใน Layout นี้ภาพซ้ายจะอยู่บนภาพขวา

4. Right over Left

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพ 2 ภาพมาต่อกันทางแนวนอน ซึ่งใน Layout นี้ภาพขวาจะอยู่บนภาพซ้าย

5. Interlaced

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพ 2 ภาพที่ภาพในแต่ 1 แถวของข้อมูลภาพซ้ายและขวาจะสลับกัน โดยอาจจะขวาก่อนแถวแรก หรือซ้ายก่อนแถวแรกก็ได้ แต่ในแฟ้มข้อมูลที่นามสกุลลงท้ายด้วยตัวอักษร 'i' จะให้ภาพขวาก่อนเป็นแถวแรก

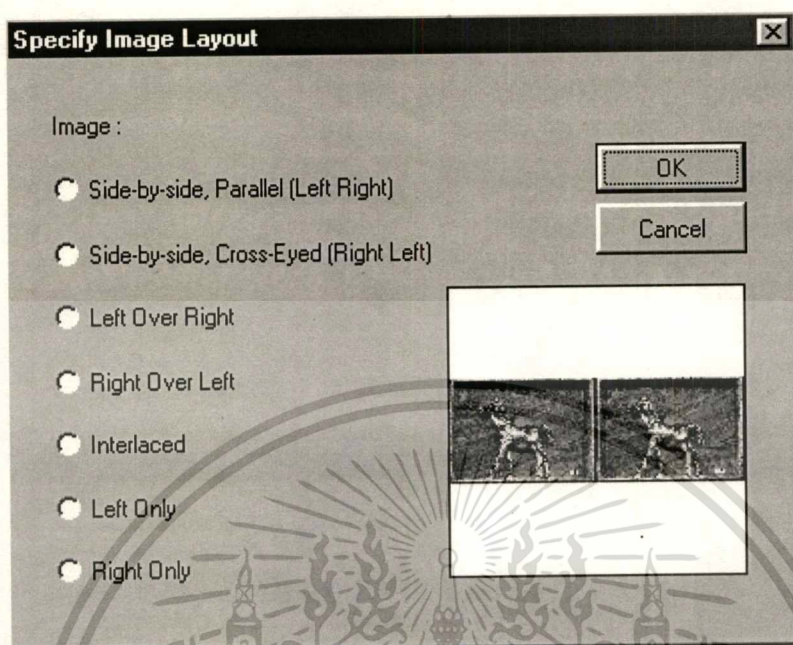
6. Left Only

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพซ้ายเพียงอย่างเดียวเท่านั้น

7. Right Only

หมายถึง ข้อมูลภายในแฟ้มข้อมูล 1 แฟ้มนั้นจะประกอบด้วยข้อมูลภาพขวาเพียงอย่างเดียวเท่านั้น

โดยเมื่อเลือกที่จะเปิดแฟ้มข้อมูล ภาพตาซ้ายและขวา (Stereoscopic Image) จะมี Dialog box ขึ้นมาให้เลือก Layout ของข้อมูลในแฟ้มดังภาพที่ 7.7

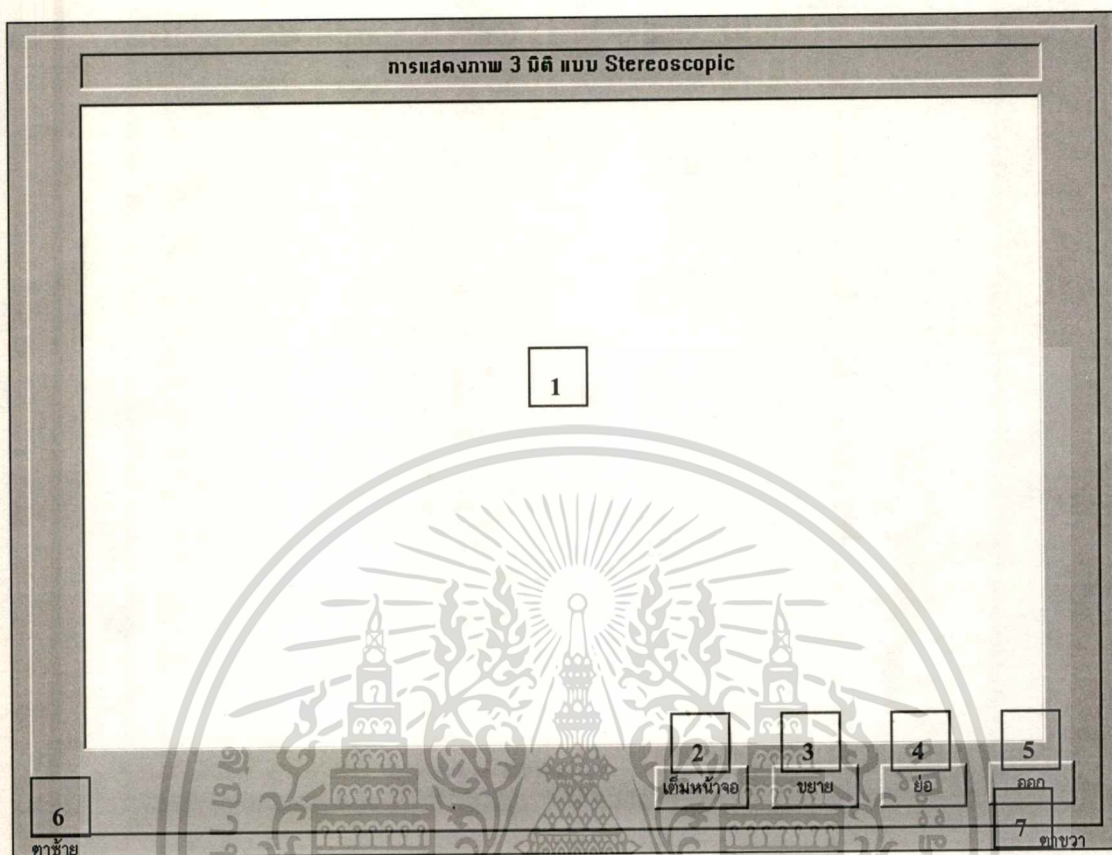


ภาพที่ 7.7 หน้าจอแสดง Layout ของภาพ

ซึ่งก่อนที่จะเปิดหน้านี้ให้เลือก Layout ของภาพ โปรแกรมจะทำการตรวจสอบว่าเพิ่มข้อมูลทีเลือกมาในตอนแรก สามารถเปิดได้เลยหรือไม่โดยดูจากนามสกุลของแฟ้ม โดยนามสกุลที่โปรแกรมสามารถตรวจสอบได้ คือ นามสกุลที่ลงท้ายด้วยตัวอักษร 's' กับ 'i' โดยถ้าเป็นตัวอักษร 's' โปรแกรมก็จะตรวจสอบว่าเพิ่มข้อมูลภาพ Layout แบบ Side-by-Side, Cross-Eyed (Right Left) และให้เปิดขึ้นมาเลย ส่วนถ้านามสกุลลงท้ายเป็นตัวอักษร 's' โปรแกรมก็จะตรวจสอบว่าเพิ่มข้อมูลภาพ Layout แบบ Interlaced และให้เปิดขึ้นมาเลย

7.3 ผลการพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 3 มิติ

การพัฒนาโปรแกรมในส่วนที่แสดงในโหมด 3 มิติจะทำการสร้างส่วนติดต่อผู้ใช้บางอย่างให้เป็น 3 มิติ คือมีความลึก โดยใช้ทฤษฎีการฉายภาพจากจุดบนพิกัด (x, y, z) ลงบนระนาบ (x, y) หรือ (x, y, 0) ที่ได้กล่าวไว้ในบทที่ 3 มาใช้ในการสร้างปุ่มกด และ กรอบของรูป โดยมีหน้าจอที่จำลองมาดังภาพที่ 7.8



ภาพที่ 7.8 หน้าจอที่แสดงในโหมด 3 มิติ

จากภาพที่ 7.8 สามารถแสดงส่วนประกอบต่าง ๆ ในหน้าจอได้ดังนี้

หมายเลข 1 : เป็นพื้นที่ในการแสดงภาพที่เปิดจากเพิ่มข้อมูลในโหมด 2 มิติ

หมายเลข 2 : เป็นปุ่มที่เมื่อกดแล้วจะทำให้ภาพที่แสดงอยู่ในกรอบหมายเลข 1 จะขยายเต็มหน้าจอ

หมายเลข 3 : เป็นปุ่มที่เมื่อกดแล้ว Mouse Cursor จะเปลี่ยนไปดังภาพที่ 7.9 เมื่ออยู่ในพื้นที่กรอบหมายเลข 1 และเมื่อใช้เมาส์ปุ่มซ้ายกดในพื้นที่หมายเลข 1 ก็จะขยายภาพให้ใหญ่ขึ้น



ภาพที่ 7.9 Mouse Cursor ในการขยายภาพ

หมายเลข 4 : เป็นปุ่มที่เมื่อกดแล้ว Mouse Cursor จะเปลี่ยนไปดังภาพที่ 7.10 เมื่ออยู่ในพื้นที่

ที่กรอบหมายเลข 1 และเมื่อใช้เมาส์ปุ่มซ้ายกดในพื้นที่หมายเลข 1 ก็จะย่อภาพให้เล็กลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



ภาพที่ 7.10 Mouse Cursor ในการถ่ายภาพ

หมายเลข 5 : เป็นปุ่มที่เมื่อกดแล้ว จะออกจากการแสดงในโหมด 3 มิติกลับไปอยู่ที่หน้าจอ ก่อนที่จะเข้ามา

หมายเลข 6 : เป็นข้อความที่แสดงให้เห็นเมื่อมองด้วยตาข้างซ้ายผ่านแว่น ซึ่งเมื่อปิดตาขวา และมองก็จะทำให้ไม่เห็นข้อความที่อยู่ ที่ หมายเลข 7

หมายเลข 7 : เป็นข้อความที่แสดงให้เห็นเมื่อมองด้วยตาข้างขวาผ่านแว่น ซึ่งเมื่อปิดตาซ้าย ก็เช่นเดียวกันก็จะทำให้ไม่เห็นข้อความที่อยู่ ที่ หมายเลข 6



บทที่ 8

สรุปผลการค้นคว้าและการพัฒนาโปรแกรม

8.1 ผลการดำเนินการพัฒนาโปรแกรม

การทำโครงการพัฒนาระบบงาน เรื่อง การพัฒนาโปรแกรมแสดงภาพสลับเชิงเวลา ภายใต้ระบบปฏิบัติการไมโครซอฟต์วินโดวส์ ได้เริ่มต้นจาก การศึกษาวิธีการที่มนุษย์มองเห็นภาพในระบบสเตอริโอ การฉายภาพเพอร์สเปกทีฟแบบสเตอริโอ การพัฒนาโปรแกรมภายใต้ ระบบปฏิบัติการไมโครซอฟต์วินโดวส์ด้วย ไมโครซอฟต์ DirectX และ WINx3D Library และการนำทั้งหมดมาประยุกต์โดยใช้ภาษา C และ C++ ในการพัฒนานบนเครื่องมือที่เรียกว่า Visual C++ version 6.0 ซึ่งรายละเอียดของโปรแกรมที่พัฒนามีดังต่อไปนี้

- สามารถแสดงภาพซ้ายหรือขวา ที่มีนามสกุล BMP, JPG และ PNG ได้ในโหมด 2 มิติ
- สามารถแสดงภาพสเตอริโอที่มีนามสกุล BMI, BMS, JPS, JPO, JPI, JPP, PNG และ PNS ได้ในโหมด 2 มิติ
- นำภาพซ้ายและขวาที่ได้ในโหมด 2 มิติมาแสดงในโหมด 3 มิติได้
- ส่วนติดต่อกับผู้ใช้บางอย่างในโหมด 3 มิติ จะเป็นในลักษณะ 3 มิติ
- ย่อ / ขยาย ภาพได้ในโหมด 3 มิติ

8.2 ข้อเสนอแนะ

ในการพัฒนาโปรแกรมในครั้งนี้ได้พัฒนาในส่วนที่ใช้สำหรับแสดงภาพซึ่งจะดึงข้อมูลมาจากแฟ้มข้อมูล ซึ่งถ้าสามารถดึงข้อมูลจากกล้องหรืออุปกรณ์ถ่ายภาพต่าง ๆ ได้ ก็จะสามารถนำไปประยุกต์ใช้ในการสอนการผ่าตัดของทางการแพทย์ได้ ซึ่งในการ ย่อ หรือ ขยายก็จะได้ภาพที่ดูชัดเจนกว่านี้ เพราะเป็นการย่อ / ขยาย ผ่านอุปกรณ์ถ่ายภาพ และในส่วนของการเปิดแฟ้มข้อมูลภาพชนิดต่าง ๆ ยังไม่ครบถ้วนทั้งหมดของประเภทแฟ้มข้อมูลภาพ เช่น นามสกุล gif และภาพที่สามารถเคลื่อนไหวได้ เช่น พิกเจอร์ไอต่าง ๆ

บรรณานุกรม

- LaMothe, A . 1998. **Windows Game Programming For Dummies**. IDG Books Worldwide, Inc.
- StereoGraphics Corporation. 1997. StereoGraphics Developers' Handbook. [Online]. Available :
<http://www.stereographics.com>
- the Win3D company. 2000. **Developing Stereoscopic DirectDraw Applications with WINx3D**.
 [Online]. Available : <http://www.win3d.com>



ประวัติผู้เขียน

ชื่อผู้เขียน	นาย นริศ อธิธำวาชกุล
วันเดือนปีเกิด	4 กุมภาพันธ์ 2519
สถานที่เกิด	ชลบุรี
วุฒิการศึกษาระดับปริญญาตรี	วท.บ.(สถิติประยุกต์)
สถานที่สำเร็จการศึกษา	คณะวิทยาศาสตร์ประยุกต์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
ปีที่สำเร็จการศึกษาในระดับปริญญาตรี	2539

