โปรแกรมสร้างเสียงดนตรีอย่างง่าย

THE PROGRAM FOR CREATING BASIC MUSIC

ธเนศ ศรีสุรภานนท์
พนมรุ้ง สุริยะกาญจน์
พิรดา บัวพันธุ์

ปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2548

THE PROGRAM FOR CREATING BASIC MUSIC

THANETH SRISURAPANON

PANOMRUNG SURIYAKARN

PIRADA BUAPUNTH

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT

OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2005

หัวข้อปัญหาพิเศษ โปรแกรมสร้างเสียงดนตรีอย่างง่าย

THE PROGRAM FOR CREATING BASIC MUSIC

ชื่อนักศึกษา นายธเนศ ศรีสุรภานนท์ 45050482

นายพนมรุ้ง สุริยะกาญจน์ 45050495

นางสาวพิรดา บัวพันธุ์ 45050501

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

อาจารย์ที่ปรึกษา รศ.ธีรวัฒน์ ประกอบผล

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นับปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ ประจำปีการศึกษา 2548

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ. กฤษฎา บุศรา	
กรรมการ	อ. ศังกรศรัณย์ ล่องชูผล	
กรรมการและอาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล	

(รองศาสตราจารย์ ดร.วีระ บุญจริง)

หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ โปรแกรมสร้างเสียงดนตรีอย่างง่าย

ชื่อนักศึกษา นายธเนศ ศรีสุรภานนท์ 45050482

นายพนมรุ้ง สุริยะกาญจน์ 45050495

นางสาวพิรดา บัวพันธุ์ 45050501

ปริญญา วิทยาศาสตรบัณฑิต

ภาควิชา คณิตศาสตร์และวิทยาการคอมพิวเตอร์

สาขาวิชา วิทยาการคอมพิวเตอร์

ปีการศึกษา 2548

อาจารย์ที่ปรึกษา รศ. ธีรวัฒน์ ประกอบผล

บทคัดย่อ

โปรแกรมเมโลดี้ มูส เป็นโปรแกรมทางด้านมัลติมีเดีย ที่พัฒนาขึ้นเพื่อตอบสนองความต้องการของกลุ่มคนที่มีความสนใจทางด้านดนตรี แต่ยังไม่มีพื้นฐานในทฤษฎีดนตรีมากนัก โดยผู้ใช้เรียนรู้การใช้งานได้ง่าย มีการทำงานที่ไม่ยุ่งยาก ซึ่งเป็นจุดที่แตกต่างออกไปจากโปรแกรมทำเพลงที่มีอยู่ทั่วๆไปซึ่งมีความซับซ้อนมากกว่า ทำให้ผู้ใช้สร้างเพลงได้ตามที่ต้องการโดยที่ไม่จำเป็นต้องมีความรู้ด้านคอมพิวเตอร์หรือดนตรีมากนัก

โดยผู้ใช้โปรแกรมจะติดต่อกับโปรแกรมนี้ผ่านทางอินเทอร์เฟสที่มีหน้าจอหลักประกอบด้วยรายการเสียงเครื่องดนตรีต่างๆให้เลือกใช้ และมีคีย์บอร์ดซึ่งเป็นตัวแทนของเครื่องดนตรีที่ใช้สร้างเสียงดนตรี โดยสามารถทำเพลงได้โดยใช้เมาส์คลิกที่แป้นคีย์บอร์ดโดยตรงเพื่อเปล่งเสียงเครื่องดนตรีที่เลือก และสามารถบันทึกเพลงนั้นๆเก็บไว้ได้ในรูปแบบไฟล์มิดิ โดยใช้หลักการบันทึกคล้ายการบันทึกเสียงลงแถบบันทึกเสียง

ในส่วนการทำงานในตัวโปรแกรมจะทำการส่งสัญญาณเสียงที่ผู้ใช้เลือกไปยังซาวน์การ์ดและทำการส่งเสียงจากซาวน์การ์ดซึ่งเป็นเสียงในรูปแบบมิดิ ออกมาเป็นเสียงเครื่องดนตรีตามที่ผู้ใช้เลือก เมื่อผู้ใช้โปรแกรมทำเพลงของตนเองเสร็จเรียบร้อยแล้วก็สามารถเก็บเอาไฟล์นั้นๆไปใช้งานได้ หรือบันทึกเก็บไว้ เพื่อนำเพลงมาปรับปรุงใหม่ในโอกาสต่อไปได้

Special Project Title	The Program for Creating Basic Music		
Students	Mr.Thaneth	Srisurapanon	45050482
	Mr.Panomrung	Suriyakarn	45050495
	Miss.Pirada	Buapunth	45050501
Degree	Bachelor of Science		
Department	Mathematics and Computer Science, Faculty of Science		
Programme	Computer Science		
Academic Year	2005		
Special Project Advisor	Assoc.Prof.Teerawat	Prakobphol	

Abstract

Melody Muze is a multimedia program developed for people who are interested in music, but they don't have knowledge in foundation of music theory. This program is easy to learn and use because it is not complicated like other creating music programs. A user can create music which he/she wants although he/she doesn't have advanced music or computer skill. A user can use this program via graphical user interface which is composed of sounds of many instruments represented by the keyboard. A user can click at many positions of keyboard panel for emitting sounds, and user's created music can be saved in midi file. The saving process is similar to saving music to tape recorder. This program sends sound signals to sound card, and then sounds in a form of midi from sound card are transformed into sounds which a user created. A user can take his created music file to play with other programs, or he can save his created music file to modify later.

กิตติกรรมประกาศ

ปัญหาพิเศษฉบับนี้สำเร็จลุล่วงได้ คณะผู้จัดทำขอกราบขอบพระคุณ รศ. ธีรวัฒน์ ประกอบผล อาจารย์ที่ปรึกษาปัญหาพิเศษ ซึ่งได้ให้คำปรึกษา ข้อชี้แนะ และความช่วยเหลือในหลายสิ่งหลายอย่างจนกระทั่งลุล่วงไปได้ด้วยดี

ขอกราบขอบพระคุณ ผศ. กฤษฎา บุษรา ประธานกรรมการสอบปัญหาพิเศษ และ อ. ศังกร ศรัณย์ ล่องชูผล กรรมการสอบปัญหาพิเศษ ที่ให้ความกรุณาในการแก้ไขข้อบกพร่องต่าง ๆ ของปัญหาพิเศษ รวมทั้งอาจารย์ทุกท่านที่ประสิทธิ์ประสาทวิชาความรู้ให้ ซึ่งทำให้ทางคณะผู้จัดทำสามารถนำความรู้ที่ได้รับมาแก้ไขปัญหาต่าง ๆ ที่เกิดขึ้นจนปัญหาพิเศษนี้สำเร็จลุล่วงไปได้ด้วยดี

ขอขอบพระคุณเจ้าหน้าที่ห้องคอมพิวเตอร์ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ทุกท่าน ที่ให้ความช่วยเหลือด้านทรัพยากรคอมพิวเตอร์และระเบียบวินัยอันดีแก่คณะผู้จัดทำ รวมทั้งผู้มีพระคุณทุกท่านที่มิได้เอ่ยนามไว้ ณ ที่นี้ด้วย

นอกจากนี้คณะผู้จัดทำขอกราบขอบพระคุณ บิดา มารดา ที่ให้ความสนับสนุนทางด้านกำลังใจและทุนทรัพย์ จนกระทั่งการทำปัญหาพิเศษครั้งนี้สำเร็จได้ด้วยดี รวมทั้งเพื่อน ๆ และน้อง ๆ ทุกคนที่ได้ให้ความช่วยเหลือในด้านต่าง ๆ เกี่ยวกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2549

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

สารบัญภาพ

บทที่ 1
บทนำ

1.1 ที่มาของปัญหาพิเศษ

เนื่องด้วยปัจจุบันนี้โปรแกรมคอมพิวเตอร์ต่าง ๆ ได้เข้ามามีส่วนเกี่ยวข้องกับดนตรีเป็นอย่างมาก รวมถึงอุปกรณ์และฟังก์ชันการทำงานต่าง ๆ ของคอมพิวเตอร์ในด้านมัลติมีเดียมีการพัฒนาขึ้นอย่างรวดเร็ว เอื้ออำนวยต่อการนำคอมพิวเตอร์มาใช้สร้างสรรค์สื่อต่าง ๆ ทางด้านดนตรี อีกทั้งขั้นตอนการบันทึกเสียงสำหรับการสร้างเสียงดนตรีนั้น มีรายละเอียดมากมายที่ค่อนข้างซับซ้อน ด้วยเหตุผลเหล่านี้ จึงได้มีการพัฒนาโปรแกรมคอมพิวเตอร์ทางด้านดนตรีขึ้น เพื่อช่วยลดค่าใช้จ่ายและขั้นตอนต่าง ๆ ในการบันทึกเสียงทั่วไปให้น้อยลง แต่โปรแกรมสร้างเสียงดนตรีในรูปแบบซีเควนเซอร์ส่วนใหญ่นั้น มีรูปแบบการใช้งานที่เหมาะสำหรับบุคคลผู้มีความรู้ทางด้านคอมพิวเตอร์และดนตรีอยู่มาก จึงไม่สามารถตอบสนองความต้องการของผู้ใช้งานกลุ่มหนึ่งซึ่งเป็นผู้เริ่มต้นศึกษาทางด้านดนตรี และไม่มีความชำนาญในการอ่านโน้ตมากเท่าที่ควรหรือไม่มีเครื่องดนตรีที่จำเป็นต้องใช้สำหรับฝึกฝนได้ แต่เนื่องจากจำนวนของผู้ใช้งานกลุ่มนี้เพิ่มขึ้นเป็นจำนวนมาก จึงได้เกิดการพัฒนาโปรแกรมสร้างเสียงดนตรีอย่างง่ายขึ้น เพื่อเป็นอีกทางเลือกหนึ่งในการศึกษาทางด้านดนตรีโดยใช้โปรแกรมคอมพิวเตอร์ ซึ่งลักษณะของโปรแกรมนั้นมีรูปแบบการใช้งานไม่ซับซ้อน ผู้ใช้จึงไม่จำเป็นต้องมีความรู้ทางด้านทฤษฎีดนตรีสูงมากนัก เพียงแค่มีความสนใจในการพัฒนาทักษะทางด้านดนตรีเท่านั้น จึงเหมาะกับผู้ที่เริ่มต้นศึกษาทางด้านดนตรีเป็นอย่างมาก

1.2 วัตถุประสงค์ของการศึกษา

1.2.1 เพื่อพัฒนาโปรแกรมคอมพิวเตอร์ที่สามารถสร้างเสียงดนตรีได้

1.2.2 เพื่อฝึกฝนทักษะและความสามารถในการพัฒนาโปรแกรมทางด้านมัลติมีเดีย

1.2.3 เพื่อศึกษาและพัฒนาโปรแกรมมัลติมีเดียโดยใช้โปรแกรม Visual C++ 6.0

1.2.4 เพื่อศึกษาและพัฒนาโปรแกรมที่มีการติดต่อกับ DirectX 9.0

1.3 ขอบเขตของปัญหา

โปรแกรมจะจำลองเสียงของเครื่องดนตรีแต่ละชนิดไว้ เช่น คีย์บอร์ด กลอง หรือกีตาร์ เป็นต้น ผู้ใช้โปรแกรมสามารถเลือกได้ว่าจะใช้เสียงของเครื่องดนตรีชนิดไหน โดยที่เสียงเครื่องดนตรีทั้งหมดนั้นเป็นไฟล์เสียงแบบมิดิที่มีอยู่ในซาวด์การ์ดของเครื่องคอมพิวเตอร์ ผู้ใช้โปรแกรมจะติดต่อกับโปรแกรมนี้ผ่านทางอินเทอร์เฟสที่มีหน้าจอหลักเป็นคีย์บอร์ด ซึ่งง่ายต่อความเข้าใจมากกว่าเครื่องดนตรีชนิดอื่น ๆ สามารถสร้างเสียงดนตรีได้โดยใช้เมาส์คลิกที่แป้นคีย์บอร์ดโดยตรงหรืออาจจะใช้คีย์บอร์ดของคอมพิวเตอร์แทนได้ เพื่อฟังเสียงของโน้ตต่าง ๆ และฝึกจนชำนาญก่อนที่จะบันทึกเพลงนั้นเก็บไว้ โดยโปรแกรมจะบันทึกเพลงในแบบรีลไทม์ สามารถเล่นเพลงที่บันทึกไว้เพื่อฟังซ้ำได้ และเมื่อบันทึกเสียงได้ตามที่ต้องการแล้ว ก็สามารถเก็บเสียงดนตรีเหล่านั้นไว้เป็นไฟล์ .mmz ได้ ซึ่งเปิดฟังได้จากโปรแกรมนี้เช่นกัน

และผู้ใช้โปรแกรมสามารถเลือกใช้การทำงานในส่วนต่อไปของโปรแกรมได้ โดยการนำไฟล์เสียงดนตรีจากเครื่องดนตรีแต่ละชิ้นที่ได้สร้างไว้มาประกอบกันขึ้นเป็นเพลงได้อีกด้วย

1.4 ขั้นตอนในการดำเนินงาน

1.4.1 ขั้นตอนการวิเคราะห์และออกแบบโปรแกรม

1.4.1.1 **การวิเคราะห์ความต้องการ** เป็นการออกแบบรูปแบบ และระบบการทำงานของโปรแกรม

1.4.1.2 **ออกแบบหน้าจอส่วนที่ใช้ติดต่อกับผู้ใช้งาน** ออกแบบมาให้เป็นลักษณะที่ใช้งานได้ง่าย และดึงดูดความสนใจของผู้เล่น

1.4.1.3 **ออกแบบวิธีการจัดเก็บข้อมูล** โดยที่ต้องใช้พื้นที่ในการจัดเก็บน้อย เป็นระเบียบ และสามารถนำมาใช้งานได้ง่าย เพื่อเพิ่มประสิทธิภาพการทำงานให้กับโปรแกรม

1.4.1.4 **ออกแบบวิธีการเขียนโปรแกรม** เลือกใช้ฮาร์ดแวร์ , ซอฟต์แวร์ รวมถึงคอมโพเนนต์ และเครื่องมือต่าง ๆ ที่เหมาะสม เพื่อนำมาใช้ในการพัฒนาโปรแกรม

1.4.2 ขั้นตอนการลงมือปฏิบัติจริง

1.4.2.1 **ด้านโปรแกรมมิ่ง** การทำงานในส่วนนี้จะเกี่ยวข้องกับการเขียนคำสั่งต่าง ๆ เพื่อเรียกใช้งานฟังก์ชันต่างๆ ที่สร้างไว้ ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. ใช้โปรแกรม Visual C++ 6.0 ในส่วนของการเขียนและแก้ไขซอร์สโค้ดของโปรแกรม
2. ใช้ DirectX 9.0 เพื่อช่วยในการเขียนโปรแกรม

3. เขียนคำสั่งที่ทำให้เกิดการกระทำตามที่ต้องการ

1.4.2.2 ด้านซาวด์เอฟเฟกต์ การทำงานในส่วนนี้ใช้ DirectMusic ในการนำไฟล์เสียงมาใช้งานในโปรแกรม

1.4.3 ขั้นตอนการทดสอบโปรแกรม

การทำงานในส่วนนี้ จะเป็นขั้นตอนในการทดสอบโปรแกรมที่เราพัฒนาขึ้นมาว่ามีความสมบูรณ์มากน้อยเพียงใด เพื่อหาข้อผิดพลาดและประเมินประสิทธิภาพของโปรแกรม เมื่อตรวจพบข้อผิดพลาดก็สามารถทำการแก้ไขได้โดยทันที แล้วจึงจัดทำเอกสารการวิจัยสรุปผลการทดลอง

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้รับความรู้เกี่ยวกับการพัฒนาโปรแกรม โดยใช้โปรแกรม Visual C++ 6.0 ร่วมกับ DirectX 9.0

1.5.2 ได้รับความรู้เกี่ยวกับทฤษฎีพื้นฐานทางด้านดนตรี และสามารถศึกษาการทำงานของโปรแกรมทำเพลงทั่วไปได้เป็นอย่างดี

1.5.3 ได้โปรแกรมที่สามารถตอบสนองความต้องการของกลุ่มเป้าหมาย คือ ผู้ที่เริ่มต้นศึกษาทางด้านดนตรี

1.6 อุปกรณ์ที่ใช้ในการทำปัญหาพิเศษ

1.6.1 เครื่องคอมพิวเตอร์ Pentium IV 1.8 MHz จำนวน 3 เครื่อง

1.6.2 หน่วยความจำ 256 MB

1.6.3 Hard Disk 40 GB

1.6.4 Flash Memory 256 MB จำนวน 3 อัน

1.6.5 Microsoft Visual C++ 6.0

1.6.6 DirectX 9.0 (DirectX 9.0 Runtime , DirectX 9.0 SDK)

ตาราง 1.1 แสดงการวางแผนงาน

ขั้นตอนการทำงาน	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.	พ.ย.	ธ.ค.	ม.ค.	ก.พ.	มี.ค.
1) ขั้นตอนการวิเคราะห์และออกแบบโปรแกรม										
● การวิเคราะห์ความต้องการของรูปแบบในการพัฒนาโปรแกรม										
● ออกแบบหน้าจอส่วนที่ใช้ติดต่อกับผู้ใช้งาน										
● ออกแบบวิธีการจัดเก็บข้อมูลที่จำเป็นต้องใช้ภายในโปรแกรม										
● ออกแบบวิธีการเขียนโปรแกรม และเลือกใช้ฮาร์ดแวร์ , ซอฟต์แวร์ รวมถึง คอมโพเนนท์และเครื่องมือต่าง ๆ ที่นำมาใช้ในการพัฒนาโปรแกรม										
2) ขั้นตอนการลงมือปฏิบัติการจริง										
● ด้านโปรแกรมมิ่ง										
● ด้านซาวด์เอฟเฟกต์										
3) ขั้นตอนการทดสอบโปรแกรม										
● ทดสอบความสมบูรณ์ของโปรแกรม										
● หาข้อผิดพลาดของโปรแกรม										
● ประเมินประสิทธิภาพของโปรแกรม										
● ปรับปรุงแก้ไขข้อผิดพลาด										
● สรุปผลการทำงาน										

บทที่ 2
ทฤษฎีและผลงานวิจัยที่เกี่ยวข้อง

2.1 Microsoft Visual C++ 6.0

Microsoft Visual C++ 6.0 เป็นภาษาในการเขียนโปรแกรมเชิงวัตถุ (Object – Oriented Programming) แบบ GUI (Graphic User Interface) ตัวหนึ่งจากบริษัทไมโครซอฟท์ เป็นเครื่องมือสำหรับพัฒนาโปรแกรมที่มีความสามารถสูงในยุคนี้ Microsoft Visual C++ ได้รับการพัฒนาให้มีความยืดหยุ่น และมีประสิทธิภาพสูงขึ้น โดยมีที่มาจากภาษา C++ และได้สนับสนุนการพัฒนาโปรแกรมในหลาย ๆ ด้านไม่ว่าจะเป็นการสร้างโปรแกรมทั่วไป , การสร้างโปรแกรมการจัดการฐานข้อมูล , การสร้างโปรแกรมบนระบบเครือข่าย หรือการสร้างโปรแกรมที่เกี่ยวกับมัลติมีเดียนั้น ก็สามารถทำได้อย่างครบครัน

ในปัจจุบัน Microsoft Visual C++ ได้รับการพัฒนาจนถึงเวอร์ชันที่ 6 มีลักษณะเป็น IDE (Integrated Development Environment) คือเป็นโปรแกรมซึ่งมีไว้ใช้สำหรับเพิ่มความสะดวกในการสร้าง และแก้ไขโปรเจ็กต์ให้ง่ายขึ้นโดยจะมีเครื่องมืออำนวยความสะดวกในการพัฒนาโปรแกรมต่าง ๆ ให้เรียกใช้งานได้ใน IDE เช่น โปรแกรมที่ใช้ในการดีบัก , คอมไพเลอร์ และลิงค์เกอร์ เป็นต้น นอกจากนี้ยังมีส่วนรองรับการพัฒนาโปรแกรมบนวินโดวส์ โดยมี MFC (Microsoft Foundation Class) ซึ่งเป็นไลบรารีที่จะช่วยอำนวยความสะดวกในการพัฒนาโปรแกรมบนวินโดวส์ด้วย

นอกจากนี้ Microsoft Visual C++ ยังมีส่วนของการทำงานที่สนับสนุนทางด้านของมัลติมีเดียต่าง ๆ เกี่ยวกับรูปภาพ , การเล่นไฟล์ภาพเคลื่อนไหว รวมถึงทางด้านของเสียง (Sound) อีกทั้งยังมีส่วนของเครื่องมือที่สนับสนุนการใช้งานที่เกี่ยวข้องกับ Buffer I/O , Riff files หรือที่เรียกกันว่า Multimedia file I/O services อีกด้วย โดยมีความสามารถและเครื่องมือต่าง ๆ (Component) ให้ใช้งานมากมาย ซึ่งใน Visual C++ นั้นประกอบไปด้วยคอมโพเนนต์ต่าง ๆ ดังนี้

2.1.1 VC++ Developer Studio

ทำหน้าที่เป็นศูนย์กลางในการสร้างแอพพลิเคชัน ซึ่งจะประกอบไปด้วยองค์ประกอบย่อย ๆ ภายใน ดังนี้

- Project Manager ทำหน้าที่จัดการกับการสร้างแอพพลิเคชันในลักษณะของโปรเจ็กต์ (แอพพลิเคชันหนึ่ง ๆ จะประกอบด้วยองค์ประกอบหลาย ๆ ส่วนซึ่งเรียกรวมกันว่า โปรเจ็กต์)

- Text Editor ทำหน้าที่ในการเขียนโปรแกรม โดยจะเรียกโปรแกรมที่เขียนว่า Source code
- Resource Editor ทำหน้าที่ออกแบบหน้าตาของโปรแกรม เช่น เมนู ไอคอน และไดอะล็อกต่าง ๆ
- Wizard ต่าง ๆ ใช้เพื่อสร้างแอพพลิเคชันในลักษณะวิซาร์ดต่าง ๆ เช่น AppWizard และ ClassWizard จะช่วยให้สร้างโค้ดพื้นฐานแก่แอพพลิเคชันได้อย่างรวดเร็ว โดยเพียงแค่กำหนดคลาส C++ จัดการกับวินโดว์แมสเสจ และกระทำงานอื่น ๆ เพิ่มเติมอีก
- Compiler จะทำการคอมไพล์อย่างอัตโนมัติ มีการลิงค์ไฟล์ต่าง ๆ เข้าด้วยกัน
- Debugger ดีบักเพื่อแก้ไขความผิดพลาดของแอพพลิเคชัน โดยทำงานผ่าน Debugger
- Online Help รายละเอียดความช่วยเหลือ (กรณีที่ติดตั้ง MSDN แล้ว)

2.1.2 VC++ Runtime Libraries

คือ คอมโพเนนต์นี้ทำหน้าที่ในการเก็บฟังก์ชันมาตรฐานต่าง ๆ ของ ANSI C เช่น ฟังก์ชัน sin ซึ่งสามารถเรียกใช้ได้ภายในโปรแกรม C หรือ C++

2.1.3 VC++ MFC and Template Libraries

MFC เป็นไลบรารีคลาส C++ ที่ถูกสร้างมาเพื่อใช้สำหรับการสร้างแอพพลิเคชันเพื่อใช้งานกับวินโดวส์โดยเฉพาะ ซึ่งมีรูปแบบการติดต่อกับผู้ใช้แบบกราฟิก หรือที่เรากันเรียกว่า GUI (Graphic User Interface) ซึ่งคลาส MFC ทำให้การเขียนโปรแกรมง่ายขึ้น และยังช่วยประหยัดเวลาในการเขียนโค้ดด้วย นอกจากนี้ยังติดตั้งสามารถ ATL (Active Template Libraries) ซึ่งเป็นชุดของคลาส C++ ที่เป็นเท็มเพลตหรือต้นแบบ ที่จะช่วยอำนวยความสะดวกในการสร้าง ActiveX control รวมทั้งรูปแบบในการสร้างแอพพลิเคชันอื่น ๆ ที่เป็นวัตถุ COM (Component Object Model) ได้ด้วย

2.1.4 VC++ Build Tools

ประกอบด้วยคอมไพเลอร์ C/C++ , Linker , คอมไพเลอร์ Resource (สำหรับเตรียมการเกี่ยวกับ Resource ในโปรแกรมต่าง ๆ เช่น เมนู ไดอะล็อกต่าง ๆ และไอคอน เป็นต้น) รวมทั้งเครื่องมืออื่น ๆ ที่จำเป็นต่อการสร้างแอพพลิเคชันสำหรับวินโดวส์ (การสร้างแอพพลิเคชันแบบ Win32)

2.1.5 ActiveX

คือ ซอฟต์แวร์ย่อย ๆ (Software Component) หรือองค์ประกอบย่อย ๆ ที่เพิ่มเติมเข้าไปในแอพพลิเคชันที่เราสร้างขึ้น ซึ่ง ActiveX นั้นช่วยให้เราไม่จำเป็นต้องสร้างทุกส่วนของแอพพลิเคชันเอง เพียงแต่เลือกใช้องค์ประกอบย่อย ๆ ที่เหมาะสมกับงานเพื่อสร้างเป็นแอพพลิเคชันที่สมบูรณ์เท่านั้น

2.1.6 Data Access

คือ คอมโพเนนต์ที่รวมไดรเวอร์สำหรับฐานข้อมูลชนิดต่าง ๆ ไว้ เพื่อให้เราสามารถสร้างแอพพลิเคชันที่เชื่อมต่อกับฐานข้อมูลได้อย่างสะดวก และนอกจากไดรเวอร์แล้วยังประกอบไปด้วยคอนโทรล และเครื่องมืออื่น ๆ ที่ช่วยให้การสร้างแอพพลิเคชันกับฐานข้อมูลนั้นสามารถทำได้อย่างรวดเร็ว

2.1.7 Enterprise Tools

คือ คอมโพเนนต์ที่ประกอบด้วยคอมโพเนนต์ย่อย ๆ อีกดังต่อไปนี้

- Microsoft Visual SourceSafe 6.0 Client
- Application Performance Explorer
- Repository
- Visual Component Manager
- Self-installing .exe redistributable files
- Visual Basic Enterprise Components
- VC++ Enterprise Tools
- Microsoft Visual Modeler
- Visual Studio Analyzer

2.1.8 Graphics

คือ คอมโพเนนต์เกี่ยวกับรูปภาพในรูปแบบต่าง ๆ เช่น metafile , bitmap , cursor และ icon รวมทั้ง video clip ซึ่งคอมโพเนนต์เหล่านี้ใช้ในการเขียนโปรแกรมเกี่ยวกับกราฟิกต่าง ๆ

2.1.9 Tools

คือ คอมโพเนนต์ที่เป็นเครื่องมือเสริมการทำงานชนิดต่าง ๆ ของ Visual C++ ดังนี้

- API Text Viewer
- MS Info
- MFC Trace Utility
- Spy++
- Win32 SDK Tools
- OLE/COM Object Viewer
- ActiveX Control Test Container

2.2 DirectX 9.0

2.2.1 ประวัติความเป็นมาของ DirectX 9.0

ในช่วงที่ระบบปฏิบัติการไมโครซอฟท์วินโดวส์เริ่มต้นขึ้นมานั้น สิ่งที่เป็นเอกลักษณ์ของวินโดวส์ คือเรื่องของ GUI (Graphical User Interface) โดยมี GDI (Graphical Device Interface) เป็นเครื่องมือสำหรับจัดการทางด้านภาพ แต่เกมทั่วไปยังคงพัฒนาอยู่ภายใต้ระบบปฏิบัติการเดิมอยู่ ซึ่งก็คือระบบปฏิบัติการ "DOS" นั่นเอง

การเริ่มต้นครั้งแรกสำหรับไมโครซอฟท์ คือ "เครื่องมือ Win-G" สำหรับ Window95 และดูเหมือนว่าเครื่องมือตัวนี้จะไม่ได้รับการตอบรับที่ดีนักจากนักพัฒนาโปรแกรมทั่วโลก ส่งผลให้ไมโครซอฟท์ต้องปรับปรุงรูปแบบให้กับเครื่องมือตัวนี้ใหม่อีกครั้ง

ไมโครซอฟท์ได้ทำการจัดตั้งทีมงานขึ้นมาใหม่เพื่องานกราฟิก งานมัลติมีเดีย งานเนตเวิร์ก งานด้านการรับข้อมูล Input และ งานกราฟิก 3 มิติ (ชื่อเทคโนโลยี "Render Graphic) โดยใช้ชื่อว่า "DirectX"

DirectX 2.0 สามารถเข้ามาจัดการงานด้านเกมและมัลติมีเดียได้เป็นอย่างดี การทำงานเร็วขึ้นกว่าระบบปฏิบัติการ DOS จึงเป็นเหตุผลให้โปรแกรมเมอร์ทั่วโลกได้เริ่มหันมามองเทคโนโลยีตัวนี้ใหม่อีกครั้ง และจากการพัฒนาอย่างต่อเนื่องของไมโครซอฟท์เพื่อเพิ่มประสิทธิภาพของ DirectX ให้มีความสามารถมากขึ้น จึงทำให้เกิด DirectX 2.0 , DirectX 3.0 , DirectX 5.0 , DirectX 6.0 , DirectX 7.0 , DirectX 8.0 และ DirectX 9.0 ซึ่งเป็นเวอร์ชันใหม่ล่าสุดที่ไมโครซอฟท์ผลิตขึ้น และยังจะมีการพัฒนาเวอร์ชันใหม่ ๆ ต่อไป จากเทคโนโลยีนี้เอง เป็นผลให้เกมบนระบบปฏิบัติการ DOS ค่อย ๆ เลือนหายไปในที่สุด

ทุกวันนี้ DirectX 8.0 ได้นำเทคโนโลยี COM (Component object Model) มาใช้งาน ทำให้ได้รับความสะดวกสบายในการใช้งานเป็นอย่างยิ่ง ผู้พัฒนาเกมและโปรแกรมมัลติมีเดียสามารถควบคุมและสร้างการทำงานได้อย่างอิสระ

2.2.2 ความรู้เบื้องต้นเกี่ยวกับ DirectX

DirectX เป็น API ของไมโครซอฟท์ที่พัฒนาเพื่อใช้จัดเตรียมอินเตอร์เฟส สำหรับควบคุมฮาร์ดแวร์ มัลติมีเดียบนระบบ Microsoft Windows ได้อย่างมีประสิทธิภาพ และเป็นเครื่องมือให้โปรแกรมเมอร์ทำงานกับคำสั่ง และโครงสร้างข้อมูลในระดับใกล้ชิดฮาร์ดแวร์ โดยไม่ต้องสร้างโค้ดติดต่อในระดับล่างซึ่งวิธี ติดต่อจะแตกต่างกันไปตามประเภทของอุปกรณ์ การเขียนโค้ดที่เป็นอิสระจากอุปกรณ์ในลักษณะนี้ ช่วยให้โปรแกรมเมอร์สามารถสร้างซอฟต์แวร์เพื่อทำงานดังกล่าวได้อย่างดี แม้ผู้ใช้จะปรับเปลี่ยนอุปกรณ์ตัวใหม่ และเพิ่มการ์ดเร่งความเร็วแบบสามมิติ เสียง อุปกรณ์อินพุต และอื่น ๆ ก็ตาม

DirectX ได้รับการออกแบบให้นักพัฒนามีสภาพแวดล้อมคล้ายคลึงกับสภาพแวดล้อมที่มีประสิทธิภาพของ MS-DOS ซึ่งทำงานได้เร็วกว่าโค้ดที่ทำงานบนวินโดวส์ เนื่องจากไม่ต้องสูญเสียประสิทธิภาพจาก API สำหรับจัดการงานมัลติมีเดียของวินโดวส์รุ่นก่อน แต่อย่างไรก็ตามการสนับสนุนความสามารถในการทำงานของฮาร์ดแวร์ที่มีอยู่บนระบบ โดยใช้โค้ดที่เขียนขึ้นด้วย DirectX ก็สามารถรันได้เร็วกว่าแอพพลิเคชันของ MS-DOS

เมื่อใดก็ตามที่สร้างวัตถุ DirectX ให้กับดีไวซ์นั้น DirectX จะเข้าไปซักถามฮาร์ดแวร์ผ่าน HAl เพื่อดึงเอาข้อมูลเกี่ยวกับดีไวซ์ออกมา ซึ่งมีอยู่ในตาราง Cap Bits (Capability Bits) ข้อมูลที่มีอยู่ใน Cap Bits เป็นข้อมูลที่ใช้บอกความสามารถที่ฮาร์ดแวร์สามารถทำได้ หรือความสามารถใดที่ HEL ต้องจำลอง

ไมโครซอฟท์จัดเตรียม Cap Bits ให้รับทราบฟีเจอร์ของฮาร์ดแวร์ที่มีอยู่ใน HAL และฟีเจอร์ที่ต้องจำลองขึ้นโดยซอฟต์แวร์ HEL ดังนั้นวิธีที่ดีที่สุดก็คือเขียนแอพพลิเคชันโดยใช้ค่าระบบ หรือฟีเจอร์ต่ำสุดที่ยอมรับได้ และ Optimize โค้ดที่เขียนให้รันได้เร็วและมีประสิทธิภาพมากที่สุด อีกทางหนึ่งคือ ควรเขียนโค้ดเผื่อการทำงานของ HEL ในอนาคตไว้ด้วย การสนับสนุนการใช้ฟีเจอร์ขั้นสูงบนระบบที่มีความสามารถของฮาร์ดแวร์ มากกว่าค่าระบบที่ตั้งไว้นั้น สามารถจัดเตรียมฟีเจอร์เหล่านี้เป็นฟีเจอร์พิเศษที่จะมีให้เลือกบนเกมได้ เช่น การทำงาน Texture ขั้นสูง , การทำโพลีกอนที่มีความซับซ้อนสูง หรือแม้แต่การสร้างแสงแบบ ไดนามิก เผื่อว่าใครก็ตามที่มีฮาร์ดแวร์ที่มีความสามารถสูงจะสามารถใช้งานฟีเจอร์ดังกล่าวได้ ดังนั้นควรออกแบบให้รองรับฟีเจอร์ ต่าง ๆ ที่มีได้ทั้งหมด

2.2.3 DirectX คืออะไร

ก่อนที่จะมี DirectX เกิดขึ้นนั้น นักพัฒนาเกมคอมพิวเตอร์สำหรับดอสหรือวินโดวส์จะต้องเขียนเกม คอมพิวเตอร์ให้รู้จักกับฮาร์ดแวร์ ซึ่งมีอยู่มากมายในท้องตลาด และในกรณีที่

มีฮาร์ดแวร์ตัวใหม่เกิดขึ้น อาจเกิดปัญหาความไม่สนับสนุนกันระหว่างเกมคอมพิวเตอร์กับฮาร์ดแวร์ตัวใหม่นั้นได้ นักเล่นเกมจะต้องรอจนกว่านักพัฒนาเกมจะทำการอัพเดตเกมนั้น ๆ ให้ใช้ความสามารถของฮาร์ดแวร์ตัวใหม่ได้ เมื่อเป็นเช่นนี้ไมโครซอฟท์จึงได้ทำการพัฒนาเทคโนโลยีที่มีความสามารถในการเป็นสื่อกลางติดต่อระหว่างเกมคอมพิวเตอร์ หรือโปรแกรมมัลติมีเดียต่าง ๆ กับฮาร์ดแวร์ขึ้นมาโดยใช้ชื่อว่า "DirectX"

DirectX ตามความหมายจะหมายถึง ไลบรารีคำสั่ง (Run Time Library) ที่ช่วยทำงานด้านมัลติมีเดีย และ Graphic โดยตัว DirectX Foundation จะมีส่วนประกอบที่เรียกว่า HAL (Hardware Abstraction Layer) จะใช้ซอฟต์แวร์ในการตรวจสอบความสามารถของฮาร์ดแวร์ที่อยู่ในเครื่องคอมพิวเตอร์นั้นอย่างอัตโนมัติ แล้วนำมากำหนดพารามิเตอร์ของแอพพลิเคชันให้ตรงตามความเหมาะสมระหว่างเกมคอมพิวเตอร์ หรือโปรแกรมมัลติมีเดียนั้นกับไดร์เวอร์ของฮาร์ดแวร์ที่มีส่วนเกี่ยวข้องกับเกมหรือโปรแกรมนั้น ทำให้การประมวลผลโปรแกรมทำได้เร็วขึ้น เนื่องจากขั้นตอนต่าง ๆ จะถูกนำไปประมวลผลโดยตรง ไม่ต้องอาศัยตัวกลางอย่างเช่น GDI (Graphic Device Interface) ก่อน ทำให้นักพัฒนาโปรแกรมสามารถเขียนโปรแกรมให้สื่อสารกับ DirectX เท่านั้นก็เพียงพอ นอกจากนี้ DirectX Foundation ยังมีส่วนประกอบที่เรียกว่า HEL (Hardware Emulation Layer) ทำให้สามารถใช้โปรแกรมมัลติมีเดีย หรือโปรแกรมที่เกี่ยวข้องกับ 3D บน Hardware ที่ไม่สนับสนุนการใช้งานทางด้าน 3 มิติ โดยจะทำการจำลองความสามารถบางอย่างที่ฮาร์ดแวร์ตัวนั้นไม่มี ให้สามารถใช้งานได้กับโปรแกรมที่ต้องการ แม้จะมีข้อเสียอยู่บ้างตรงที่อาจทำให้การทำงานช้าลงบ้างก็ตาม แต่ก็คุ้มค่ากับความสามารถของ DirectX ที่มีอยู่ในปัจจุบัน

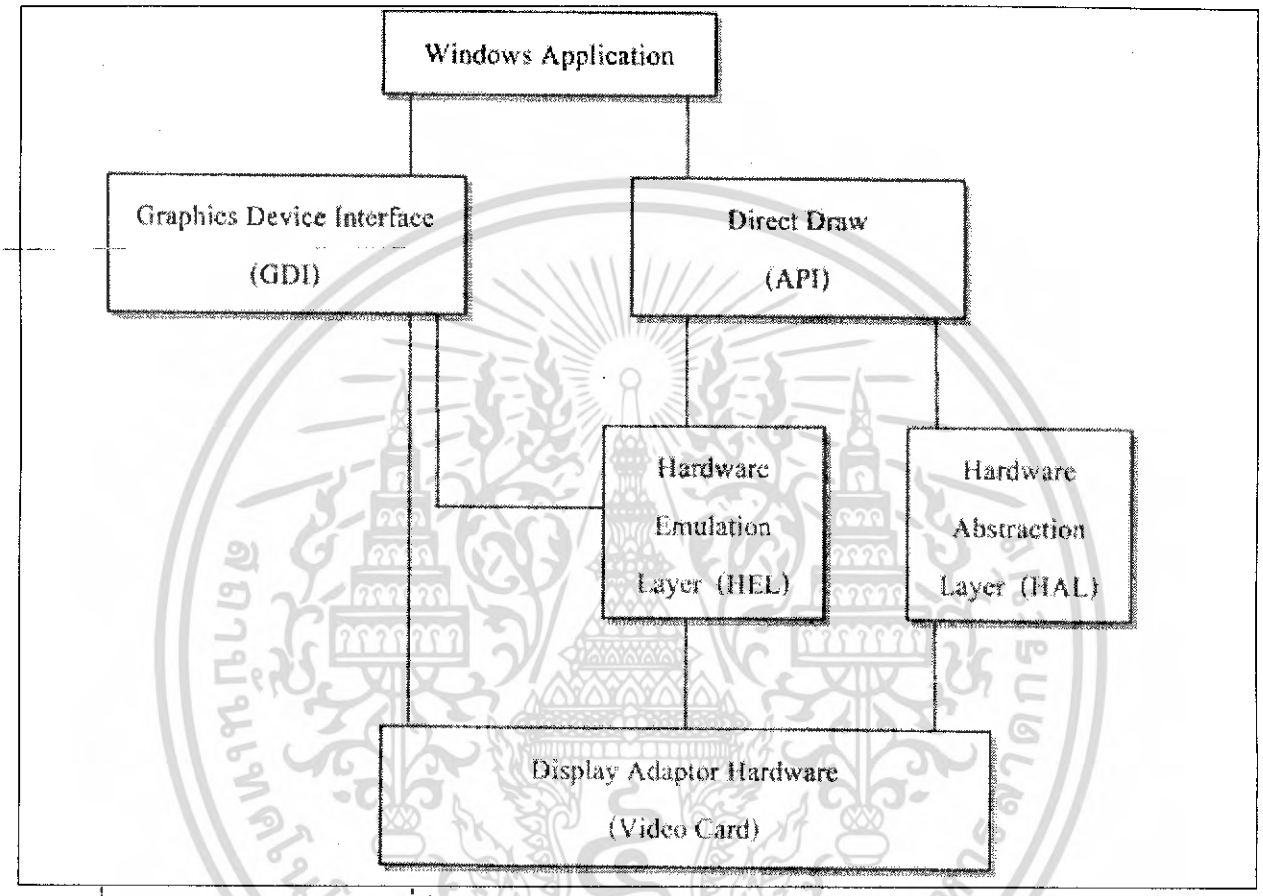
2.2.4 API ของ DirectX

DirectX ประกอบไปด้วย API มากมายที่ได้รับการออกแบบเพื่อใช้พัฒนาเกม และการจำลองแบบแบบสามมิติ (3D Simulation) แต่ส่วนใหญ่จะยังไม่ได้พัฒนาให้เป็นแบบสามมิติ โดย DirectX มีไลบรารีที่เก็บฟังก์ชันที่ใช้ในการเรนเดอร์แบบสองมิติและสามมิติ , สร้างเสียงแบบปกติ และแบบสามมิติ , ดนตรี , ติดต่อแป้นพิมพ์ , จอยสติ๊ก และอุปกรณ์อินพุตชนิดต่าง ๆ รวมทั้งฮาร์ดแวร์ที่มีความสามารถในการสร้างปฏิกิริยาสะท้อนกลับ (จอยสติ๊กแบบสั่น) และการเล่นเกมผ่านเครือข่าย สามารถใช้ไลบรารีของคำสั่งต่าง ๆ ที่รวมเข้ามาเพื่อสร้างเกม และทำซิมูเลชันที่งดงามได้

ชุด API ที่มีอยู่ใน DirectX คือ DirectDraw , Direct3D , DirectMusic , DirectSound , DirectPlay , DirectInput และ DirectSetup

2.2.4.1 DirectDraw

DirectDraw เป็นชุด API สำหรับใช้จัดการอุปกรณ์แสดงผล , ควบคุมข้อมูลบิตแมปหน่วยความจำที่ ออกนอกพื้นที่สกรีน และสร้างการติดต่อที่รวดเร็วให้กับฟีเจอร์ของฮาร์ดแวร์ เช่น Blitting และ Page Flipping ซึ่งเป็นฟีเจอร์พื้นฐานที่ Direct3D สามารถทำได้



ภาพที่ 2.1 แสดงสถาปัตยกรรมทั่วไปของ DirectDraw

- Cooperative level และ Display mode

ก่อนที่จะคำสั่งต่าง ๆ ใน DirectX ให้แสดงผลภาพได้นั้น เราจะต้องเซต Cooperative level และ Display mode ก่อน โดยการเซต Cooperative level ก็คือการบอก DirectDraw ว่าเราจะทำงานกับ DirectDraw ในลักษณะใด เช่น Window mode หรือ Full screen mode สามารถ Interrupt ด้วยการกดปุ่ม Ctrl-Alt-Del ได้หรือไม่ และสามารถที่จะเปลี่ยนขนาดของ Window ได้หรือไม่ (ใน Window mode) เป็นต้น หลังจากเซต Cooperative level แล้วจะต้องเซต Display mode เป็นลำดับถัดมา เพื่อระบุ Screen resolution , Color depths และ Refresh rate ที่ต้องการโดยจะต้องระบุความกว้าง ความยาวของ Resolution เป็นหน่วย Pixel และจะใช้สีแบบใด 6 , 8 , 24 หรือ 32 บิต หรือจะใช้

ตารางสี (palette) ทั้งนี้ Video hardware ที่ใช้จะต้องสนับสนุนกับ Mode ที่เลือกด้วย มิฉะนั้นจะเกิด Error display mode

- Surface

Surface เป็นพื้นที่บน Memory ที่ใช้สำหรับเก็บข้อมูลภาพ Bitmap เพื่อเตรียมสำหรับการนำมาแสดงผลบนจอภาพต่อไป โดยพื้นที่นี้อาจอยู่บน Video memory หรือ System memory ก็ได้ขึ้นอยู่กับค่า Flags ที่ส่งให้ตอนสร้าง โดย Surface แบ่งออกเป็น 2 ชนิด คือ Primary surface และ Off-screen surface

Primary surface นั้นเป็น Surface หลักที่จะแสดงผลออกสู่จอภาพ โดยจะต้องมีขนาดเท่ากับ Display mode ที่เลือกไว้ ภาพใด ๆ ก็ตามที่ต้องการแสดงออกทางจอภาพต้องนำมาวาดไว้บน Primary surface เสมอ แต่การนำภาพมาวาดลงบน Primary surface โดยตรงจะทำให้ภาพที่ได้กระพริบไม่ราบรื่น จึงต้องใช้การทำ Flipping มาช่วย (จะกล่าวรายละเอียดในหัวข้อถัดไป)

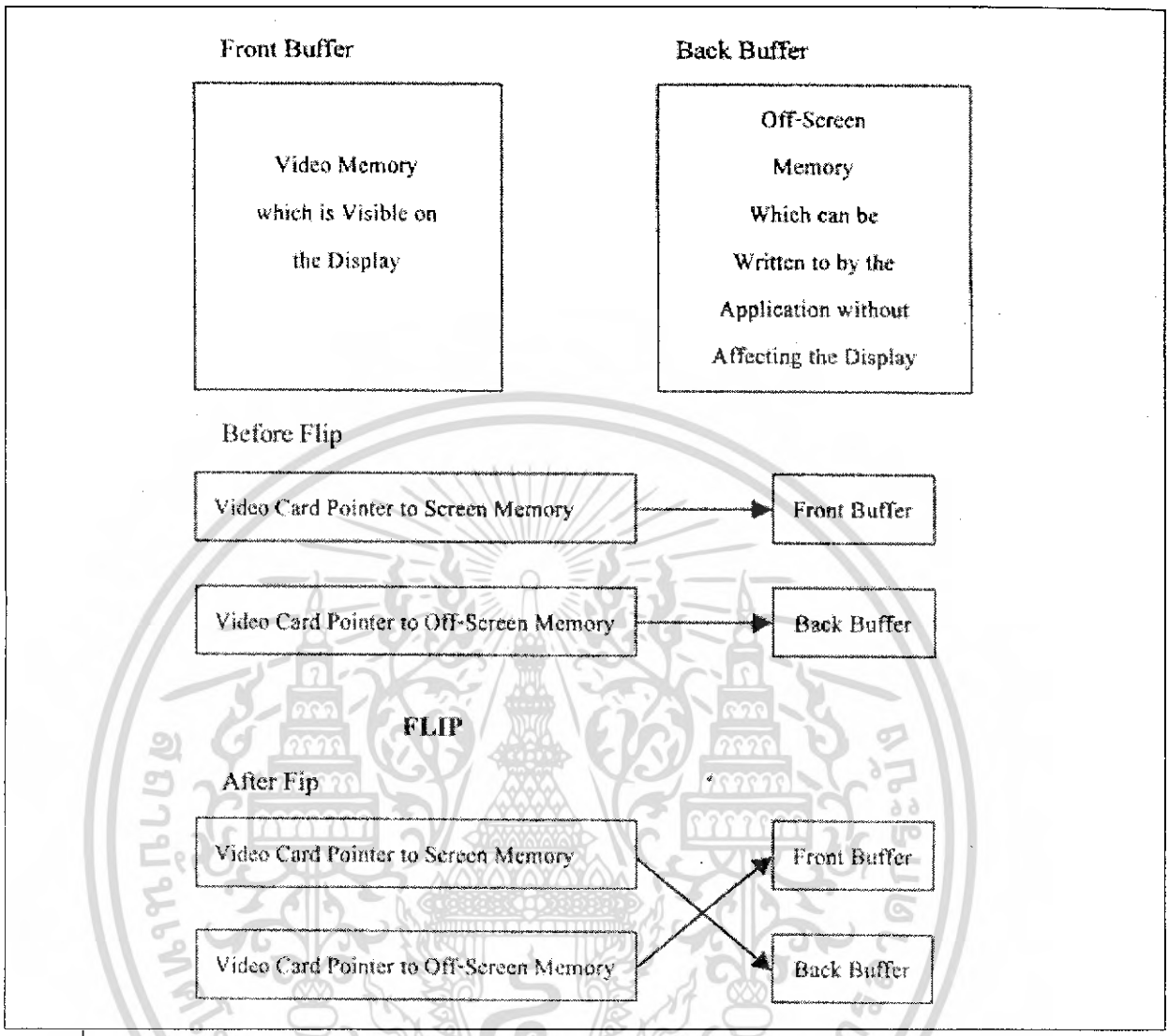
Off-screen surface คือ Surface ที่ใช้สำหรับเก็บข้อมูลภาพต่าง ๆ เพื่อ Blitting ลงใน Buffer หรือ Surface อื่นเพื่อเตรียมแสดงผลต่อไป

- Blitting และ Flipping

Blitting คือ การนำภาพต่าง ๆ จาก Source ไปวาดบนบริเวณ Destination ที่กำหนด โดยอาจมีการย่อ-ขยายภาพให้พอดีกับขนาด Destination ที่กำหนด และสามารถทำการ Transparent สีบางสีได้ด้วย การทำ Blitting สามารถทำได้หลายครั้ง โดยภาพที่ทำ Blitting จะเรียงซ้อนทับกันเป็นชั้น ๆ

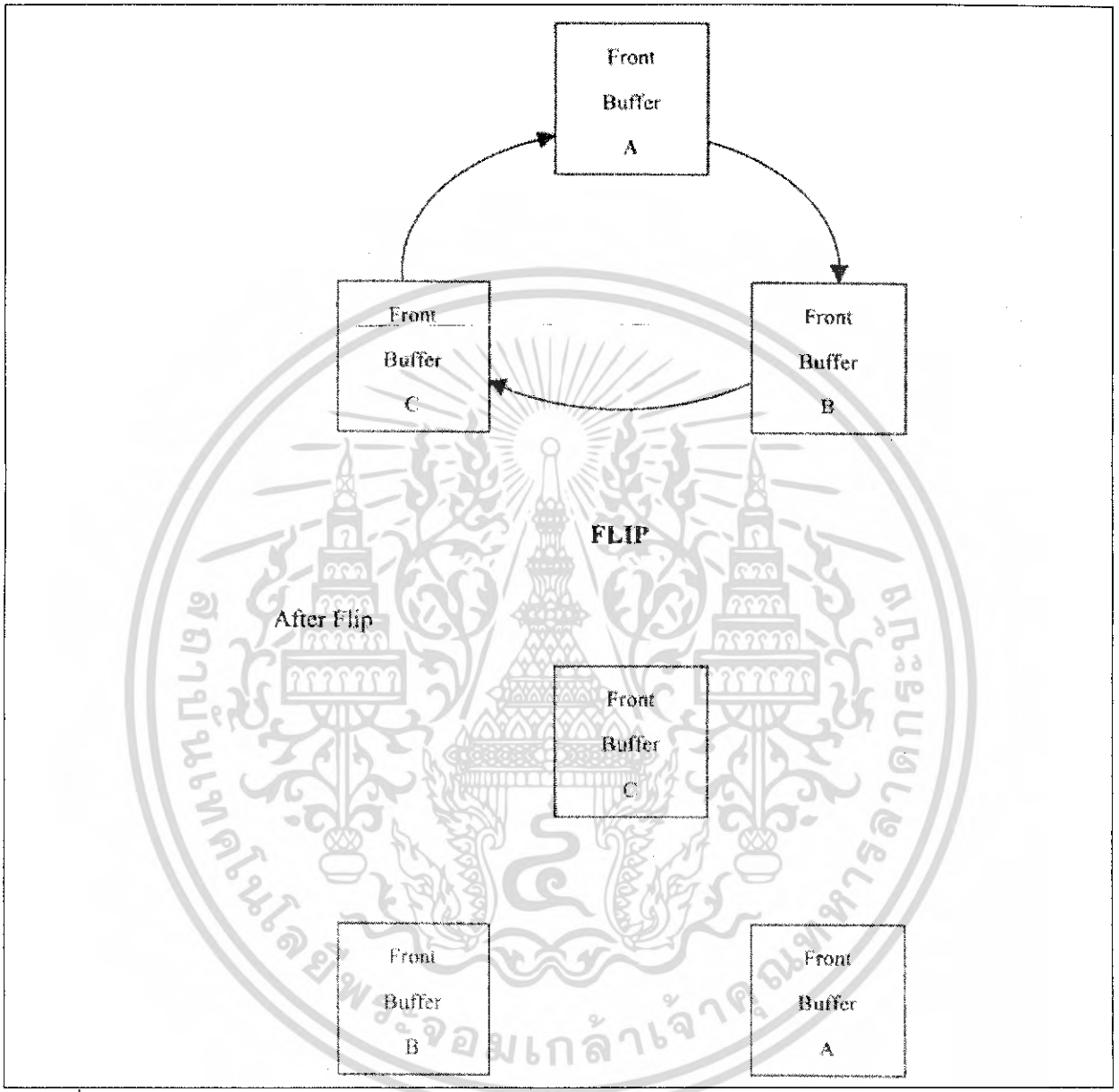
Flipping เป็นการสลับที่กันของ Buffer โดยก่อนการทำ Flipping จะต้องมีการสร้าง Buffer ขึ้นมาอย่างน้อย 2 Buffer คือ Front buffer กับ Back buffer และอาจมี Third buffer เพิ่มขึ้นมาก็ได้ โดย Front buffer คือ Buffer ที่แสดงภาพออกที่หน้าจอ หรือ Primary surface นั่นเอง ส่วน Back buffer กับ Third buffer เป็นที่พักของภาพที่จะรอแสดงผลบนหน้าจอ Buffer ทั้ง 3 ชนิดนี้จะมีขนาดเท่ากับ Display mode ที่เลือกไว้

การวาดหรือ Blitting ลงบน Front buffer โดยตรงจะให้ภาพกระพริบ ควรเตรียมภาพที่จะแสดงผลไว้ให้พร้อมบน Back buffer แล้วทำการ Flipping โดย DirectDraw จะสลับภาพระหว่าง Front buffer กับ Back buffer เพื่อนำภาพบน Back buffer มาแสดงบนจอภาพ โดยการสลับในที่นี้ไม่ได้หมายถึง การสลับที่ข้อมูลภาพแต่เป็นการสลับเฉพาะ Pointer ที่ชี้พื้นที่ที่เก็บข้อมูลอยู่เท่านั้น ซึ่งทำได้อย่างรวดเร็ว และไม่ทำให้เกิดการกระพริบ หรือการกระตุกของภาพ



ภาพที่ 2.2 แสดงการทำ Flipping ระหว่าง 2 buffer

ถ้ามี Third buffer การสลับภาพจะทำเป็นวงจร ดังภาพ

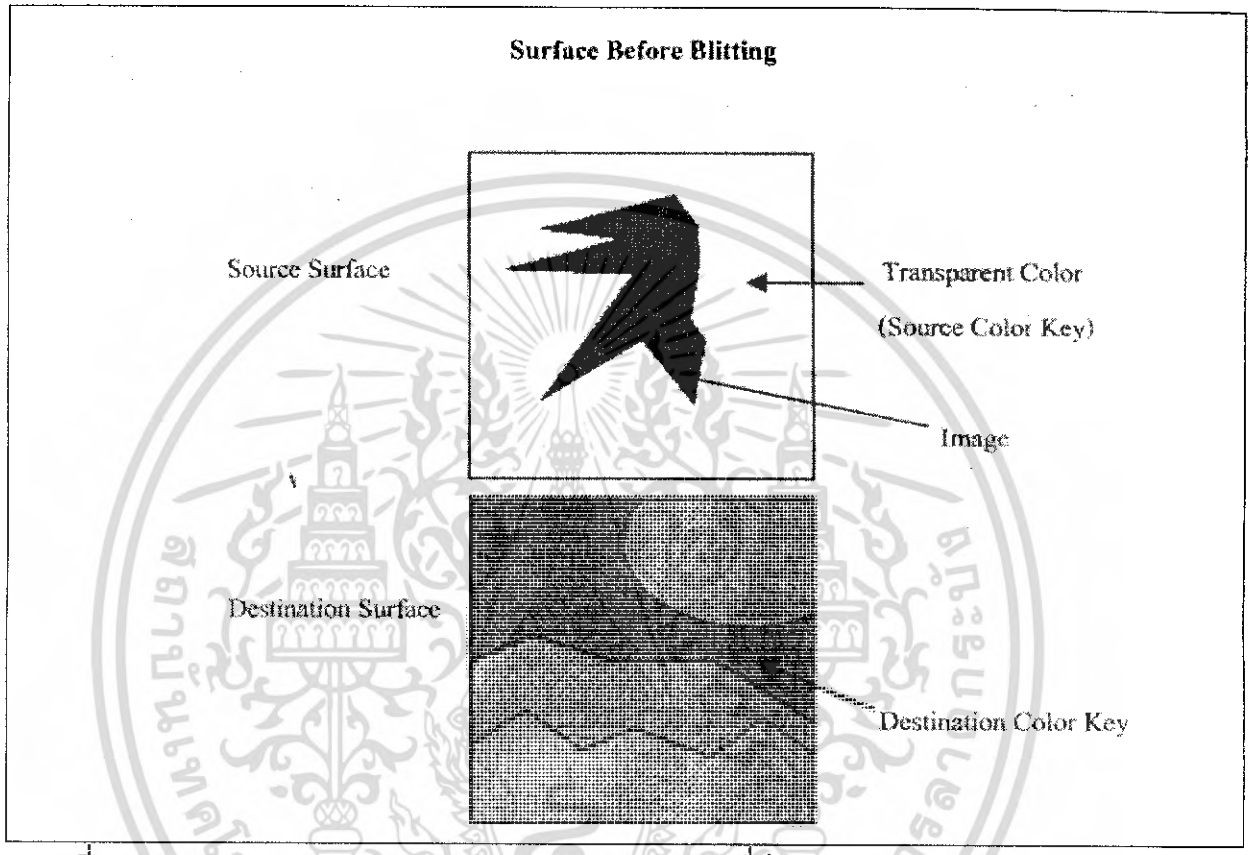


ภาพที่ 2.3 แสดงการทำ Flipping ระหว่าง 3 buffer

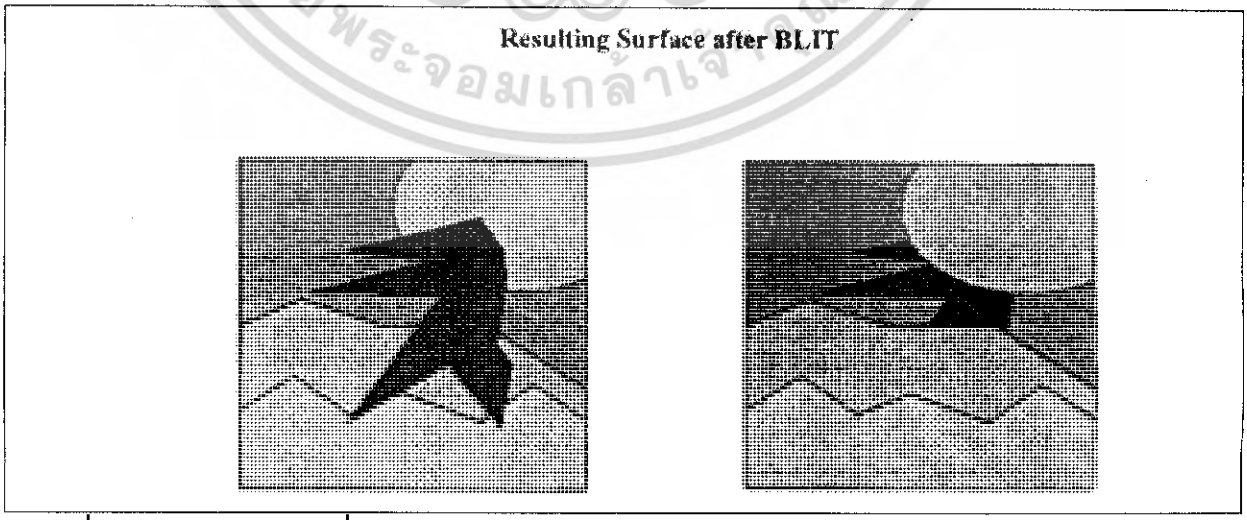
- Transparent Blitting และ Color Keys

DirectDraw สามารถทำ Transparent Blitting ได้โดยอาศัย Color Keys Surface ต่าง ๆ โดยสามารถเซต Color Keys เป็นสีใดสีหนึ่ง หรือช่วงของค่าสีก็ได้ Color Keys ที่ถูกเซตไว้จะแสดงผลตอน Blitting โดยถ้า Source มีการเซต Color Keys ไว้ ค่าสีตาม Color Keys จะไม่ถูก Copy ไปบน Destination ทำให้บริเวณสีนั้นโปร่งแสงไป ถ้า Destination มีการทำ Color Keys ไว้ ค่าสีตาม

Color Keys เท่านั้นที่จะถูกทับโดยภาพที่ Copy มาจาก Source การทำ Color Keys กับ Source มักจะใช้กับตัว Sprite ที่มีการเคลื่อนที่ไปบนฉากหลัง ส่วนการทำ Color Keys กับ Destination จะใช้กับกรณีที่ต้องการปิดบังบางส่วนของตัว Sprites



ภาพที่ 2.4 แสดง Source Surface และ Destination Surface ที่มีการกำหนด Color Keys ก่อนการ Blitting



ภาพที่ 2.5 (ซ้าย) Surface ที่ได้หลังการ Blitting โดยใช้ Source Color Key อย่างเดียว
(ขวา) Surface ที่ได้หลังการ Blitting โดยใช้ Source Color Key และ Destination Color Key

- Palettes

Palettes คือ ตารางสีที่ใช้ในระบบ Index Color ซึ่งสามารถมีจำนวนสีได้ตั้งแต่ 2 , 4 , 16 และ 256 สี ระบบ Index Color นั้นจะมี Array สำหรับเก็บค่าสี RGB โดยแต่ละค่าอาจจะมีความละเอียดที่ 16 หรือ 24 บิต แล้วในข้อมูลภาพจะเก็บเป็นค่า Index จากตาราง Palettes แทนการเก็บค่าสีจริง ซึ่งจะใช้พื้นที่ 8 บิต ต่อ 1 จุด DirectDraw สามารถสร้าง Palettes เพื่อเชื่อมต่อกับแต่ละ Surface

- Clippers

Clippers เป็น Object ที่ช่วยในการตัดภาพส่วนที่เกินออกจากพื้นที่ที่กำหนดไว้ รวมถึงตัดภาพที่นำมา Blitting บนพื้นที่ที่มี Clippers ด้วย พื้นที่ที่จะทำ Clippers นี้จะเป็นรูปสี่เหลี่ยมใน Window mode Clippers จะช่วยในการจำกัดขอบเขตของการวาดภาพไม่ให้เกินออกไปจาก Window ที่กำหนด ส่วนใน Full screen Clippers จะช่วยในการแบ่งหน้าจอออกเป็นส่วนๆ ตามที่ต้องการ

2.2.4.2 Direct3D

Direct3D คือ API ที่สามารถใช้เขียนโปรแกรมกราฟิกสามมิติ และติดต่อใช้งานฮาร์ดแวร์เร่งความเร็วสามมิติ การ์ดแสดงผลส่วนใหญ่ที่มีขายในท้องตลาดสนับสนุนความสามารถในการเร่งการแสดงผลสามมิติ และเกมสามมิติส่วนใหญ่ที่มีอยู่ในปัจจุบันก็มักจะรันบนวินโดว์ซึ่งใช้ Direct3D

Direct3D ในยุคแรกมี API อยู่สองโหมดคือ โหมด Immediate (IM) และโหมด Relation (RM) ในโหมด IM (การเรนเดอร์วัตถุทำได้โดยฉับพลันได้ตามความต้องการของโปรแกรมเมอร์) เป็นโหมดที่ใช้งานยากแต่มีความยืดหยุ่นสูง เป็น API ในระดับล่างสำหรับใช้เขียนเกมที่ทำงานได้เร็ว และมีประสิทธิภาพเท่าที่จะเป็นไปได้บนระบบ ในโหมด RM (API จะเก็บซีนลงในฐานข้อมูล แล้วนำมาเรนเดอร์ทั้งหมดในคราวเดียวกัน) เป็นโหมดที่สร้างขึ้นมาเป็นเลเยอร์ที่อยู่บนสุดของโหมด IM โดยโหมดนี้จะจัดเตรียมบริการต่าง ๆ เช่น การจัดการ Texture, การโหลด Object File, การจัดลำดับเฟรม และการทำวัตถุเคลื่อนไหว การศึกษา และใช้งานโหมด RM นั้นง่ายกว่าเมื่อเทียบกับโหมด IM แต่ในโหมด IM เป็นโหมดที่มีประสิทธิภาพ และความยืดหยุ่นสูงกว่า ดังนั้นการพัฒนาการทำงานในโหมด RM จึงได้หยุดลงใน DirectX เวอร์ชัน 6 และมุ่งพัฒนาการทำงานในโหมด IM ให้มีความสามารถ และใช้งานง่ายในเวอร์ชันต่อมา ด้วยเหตุนี้การทำงานในโหมด RM จึงไม่สนับสนุนเทคโนโลยีใหม่ เช่น

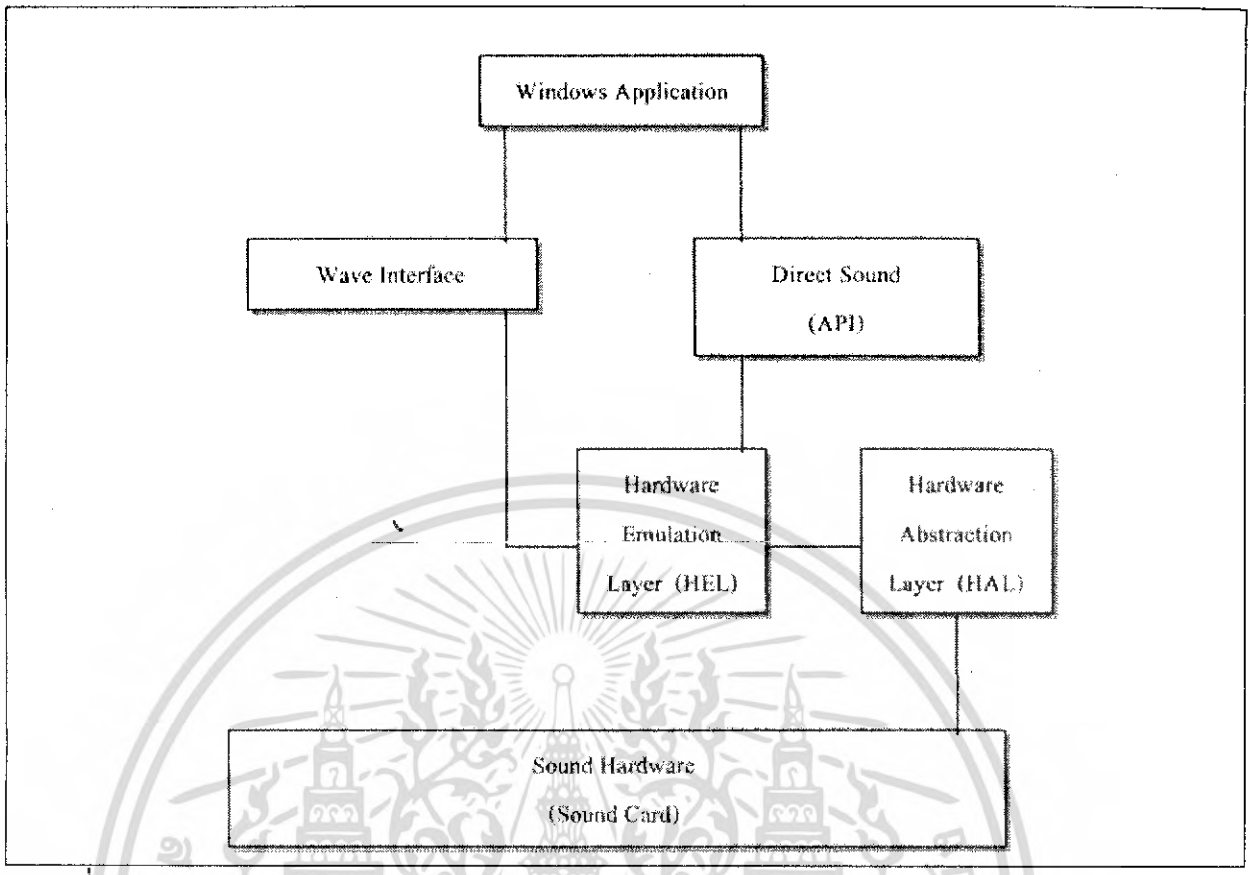
Multitexturing Bump Mapping Hardware Transformation และ Lighting ดังนั้น ความสามารถทั้งหมดของโปรแกรมสามมิติ ควรเขียนขึ้นมาด้วยการใช้ โหมด IM

2.2.4.3 DirectMusic

เป็นชุดของ API ที่ทำงานร่วมกับข้อมูลประเภท Message-based Musical Data ซึ่งเป็นข้อมูลที่แปลงมาจาก Wave Sample ด้วยซินธีไซเซอร์ (Synthesizer) ทั้งแบบฮาร์ดแวร์ หรือซอฟต์แวร์ โดยปกติถ้าเป็นแบบซอฟต์แวร์จะใช้โปรแกรม Microsoft Software Synthesizer เพื่อสร้าง Wave Sample ให้กับ DirectSound การใช้เสียงดนตรีที่ได้จากซินธีไซเซอร์จะเป็นไปตามมาตรฐาน DLS (Downloadable Sound) นอกจากนี้ DirectMusic ยังเป็นกลไกที่ใช้สร้างเพลงตามที่กำหนด

2.2.4.4 DirectSound

API ชุดนี้เป็นเครื่องมือที่ใช้จัดการเสียงแบบเสียงสเตริโอ และแบบสามมิติอย่างมีประสิทธิภาพ ประกอบด้วยความสามารถในการจัดการหน่วยความจำ และการผสมเสียงฮาร์ดแวร์ DirectSound ได้รับการออกแบบมาเพื่อดึงความสามารถของฮาร์ดแวร์บนระบบ การรวมเสียงสามมิติเข้าไปในเกม หรือการซิมูเลชัน ทำให้แอพพลิเคชันให้เสียงได้สมจริงสมจัง เช่น การได้ยินเสียงทางซ้าย ทางขวา ด้านบน หรือเสียงที่ มารอบ ๆ ตัว



ภาพที่ 2.6 แสดงสถาปัตยกรรมของ DirectSound

2.2.4.5 DirectPlay

DirectPlay ทำให้เกมสามารถเล่นได้หลายคน และทำให้สามารถเชื่อมต่อแบบ Transport-independent รวมทั้งยังให้บริการMessaging Service ด้วย ซึ่งเกมที่มีการพัฒนาออกมากันอยู่ในทุกวันนี้ล้วนสนับสนุนความสามารถดังกล่าว

2.2.4.6 DirectInput

เป็น API ของ DirectX ที่สนับสนุนอินพุตความเร็วต่ำ (Low-latency Input) ซึ่งเป็นอุปกรณ์อินพุตส่วนใหญ่ที่มีในปัจจุบัน ใน DirectX7 เพิ่มการสนับสนุนอุปกรณ์อินพุตทุกชนิดที่มีความสามารถ แสดงปฏิกิริยาสะท้อนกลับ เช่น จอยสติ๊ก หรือพวงมาลัยแบบสั่นได้ อุปกรณ์อินพุตแบบนี้สามารถจำลองสถานการณ์บางส่วนให้สมจริง เช่น จำลองสถานการณ์เมื่อขับรถชนบนถนน อุปกรณ์อินพุต (พวงมาลัย) จะส่งแรงสั่นสะเทือนให้รู้สึกตามความรุนแรง และความเสียหายของรถ, แรงถีบของปืนเมื่อยิง, แรงลมปะทะเมื่อบิน และคลื่นที่กระทบเรือ เป็นต้น

2.2.4.7 DirectSetup

API ชุดนี้ของ DirectX มีหน้าที่ติดตั้งคอมโพเนนต์ DirectX Runtime ให้กับระบบโดยอัตโนมัติถ้าหากระบบดังกล่าวยังไม่ได้รับการติดตั้ง แม้ในปัจจุบันมักจะมีการติดตั้งคอมโพเนนต์มากับระบบปฏิบัติการแล้วก็ตาม แต่สำหรับ DirectX แล้ว ไมโครซอฟท์อนุญาตให้ใช้โดยไม่ต้องเสียค่าใช้จ่าย ดังนั้นจึงไม่ต้องกังวลเรื่องของลิขสิทธิ์ โดยสามารถติดตั้งกี่ครั้งก็ได้ ไมโครซอฟท์เตรียมคอมโพเนนต์ DirectX Runtime ไว้ในไดเร็กทอรี DFX\redist ใน DirectX7 SDK

2.3 ไฟล์เสียงประเภทมิดิ

2.3.1 ประวัติความเป็นมาของมิดิ

ไฟล์ที่เป็นรูปแบบเวฟเป็นไฟล์ที่ทำให้การแสดงผลของเสียงเครื่องดนตรีของจริงมีความถูกต้องแม่นยำมาก แต่มันเป็นไฟล์ที่ค่อนข้างใหญ่ สำหรับดนตรีง่าย ๆ เราอาจจะพึงพอใจในรูปแบบการสังเคราะห์เสียงในการฟังแบบเอฟเอ็มซึ่งสัญญาณนั้นควรที่จะง่ายต่อการสร้างโดยการ์ดเสียง การ์ดเสียงจะถูกเพิ่มเข้าไปบนบอร์ดของเครื่องคอมพิวเตอร์ส่วนตัว ซึ่งสามารถที่จะจัดการควบคุม และผลิตเสียงที่ผ่านออกมาทางลำโพงที่ถูกเชื่อมต่อกับบอร์ด และสัญญาณเสียงนั้นควรที่สามารถทำให้มีการบันทึกข้อมูลเข้าประเภทเสียงทางไมโครโฟนที่เชื่อมต่อการเครื่องคอมส่วนตัวนั้น ๆ และยังสามารถควบคุมจัดการเสียงที่ถูกเก็บเอาไว้บนดิสก์ ถ้าเราพึงพอใจกับเสียงมากมายที่มีอยู่ในการ์ดเสียงอยู่แล้ว พวกเราก็ควรที่จะใช้ภาษาต้นฉบับแบบง่าย และการติดตั้งฮาร์ดแวร์ที่เรียกว่ามิดิ

มิดิกำเนิดขึ้นเมื่อประมาณปี1980 ซึ่งเป็นชื่อย่อมาจาก Musical Instrument Digital Interface มีองค์ประกอบมาจากการนำภาษามาใช้โดยอุตสาหกรรมดนตรีไฟฟ้าที่ทำให้สามารถใช้กับคอมพิวเตอร์ ตัวสังเคราะห์ คีย์บอร์ด และเครื่องมือดนตรีอื่น ๆ ที่สามารถใช้ร่วมกับเครื่องดนตรีอื่นที่สามารถใช้ด้วยกันได้ โดยตัวสังเคราะห์ จะทำการผลิตดนตรีสังเคราะห์ขึ้นมาและจะถูกนำไปรวมไว้บนการ์ดเสียง มาตรฐานมิดิจะได้รับการสนับสนุนจากตัวสังเคราะห์เป็นอย่างดี ดังนั้นเสียงจะถูกสร้างขึ้นได้ด้วยใครที่สามารถเล่นและควบคุมผู้อื่นและเสียงที่เหมาะสมอย่างใกล้ชิด คอมพิวเตอร์ ต้องมีส่วนติดต่อกับMIDIที่พิเศษ แต่มันจะถูกรวบรวมไว้ในการ์ดเสียง และในการ์ดเสียงจะต้องมีตัวแปลงทั้ง DAและAD

รูปแบบภาษามิดิจะเข้ารหัสอีเวนท์ที่อยู่บนการผลิตเสียงที่แน่นอน ถึงแม้ว่ามิดิไฟล์ทั่ว ๆ ไปจะเล็กมาก ตัวอย่างเช่น จะมีส่วนของค่าของระดับเสียงของโน๊ตตัวเดียว ระยะเวลาของเสียง และความดังของเสียง

2.3.2 ความรู้เบื้องต้นเกี่ยวกับมิดิ

ดนตรีจะถูกจัดการทำให้อยู่ในรูปของแทร็คโดยการเรียงลำดับ โดยในแต่ละแทร็คสามารถที่จะเปิดหรือปิดการบันทึกหรือการเล่นซ้ำ หลายๆครั้งที่เครื่องดนตรีเฉพาะทาง มีความเกี่ยวข้องกันกับช่องสัญญาณMIDI ช่องสัญญาณMIDIเคยเป็นตัวแบ่งmessage มีทั้งหมด16ช่องสัญญาณ มีหมายเลขตั้งแต่0-15 ช่องสัญญาณจะประกอบไปด้วย 4บิตสุดท้าย(บิตที่สำคัญน้อยที่สุด)ของmessage แต่ละช่องสัญญาณจะเกี่ยวข้องกับเครื่องดนตรีเฉพาะของแต่ละช่องสัญญาณนั้นๆ ตย.เช่น ช่องสัญญาณ1 เป็นเปียโนหรือช่องสัญญาณ10 เป็นกลอง นอกจากนี้ยังสามารถสลับเครื่องดนตรีระหว่างเทอมด้วยถ้าต้องการ และยังสามารถเชื่อมโยงความสัมพันธ์ระหว่างเครื่องดนตรีในช่องสัญญาณต่างๆด้วย เราเรียกmessageที่มี4บิตแรกรวมอยู่ด้วยว่าsystem common message ซึ่งสามารถจองพื้นที่ที่อยู่ในmessageได้ ในทางเดียวกันนั้น channel message (ที่รวมหมายเลขช่องสัญญาณไว้ด้วย) หลายๆชนิดของmessageที่ถูกส่ง เช่น messageทั่วไปสำหรับการแสดงผลของการปรับเปลี่ยนเสียง หรือการจับเวลา ทั้งหมดจะถูกเรียกว่า system message มีความเป็นไปได้ที่จะส่งmessageพิเศษถึงช่องสัญญาณของเครื่องดนตรีที่อนุญาตให้ส่งตัวโน๊ตจำนวนมากโดยปราศจากการระบุช่องสัญญาณ

วิธีที่เสียงเครื่องดนตรีสังเคราะห์จะตอบสนองต่อMIDI message นั้น ส่วนใหญ่จะเป็นการไม่ตอบสนองต่อmessage ของการเล่นเสียงที่ไม่ใช่ช่องสัญญาณสำหรับตัวมันเอง ถ้ามีหลายๆmessageที่เป็นของมันนั้น จะเป็นการเล่นโน๊ตหลายเสียงในเวลาเดียวกันบนเปียโน จากนั้นการตอบสนองเครื่องดนตรีจะให้เสียงหลายเสียงที่สามารถเล่นได้มากกว่าโน๊ตตัวเดียวจำนวนครั้งเดียว

มันเป็นการง่ายที่จะสับสนระหว่างvoice กับtimbre ซึ่งtimbreเช่นการเล่นเปียโนนั้นจะเล่นสวนทางกับไวโอลิน มันเป็นคุณภาพของเสียง เครื่องมือ(หรือการ์ดเสียง)ที่เป็นแบบหลายtimbreจะทำให้สามารถที่จะเล่นเสียงที่แตกต่างกันได้มากมายในเวลาเดียวกันเช่น เปียโน เบส กลอง เป็นต้น

ในทางกลับกัน ในส่วนของvoiceนั้น สำหรับนักดนตรีจะมีความหมายไม่แตกต่างจากคำว่าtimbre แต่ในMIDIนั้นไม่ความหมายที่แตกต่างกันกับtimbreและpitchที่เป็นรูปแบบหนึ่งของเสียงสามารถสร้างขึ้นได้ในเวลาเดียวกัน ตัวสังเคราะห์สามารถที่จะมีหลายๆเสียงได้ แต่ละvoiceจะทำงานเป็นอิสระต่อกันแต่ทำในเวลาเดียวกัน เพื่อที่จะผลิตเสียงที่แตกต่างกับtimbreและpitch

ในส่วนของ polyphony นั้นจะหมายถึงจำนวนของเสียงที่สามารถสร้างขึ้นได้ในเวลา

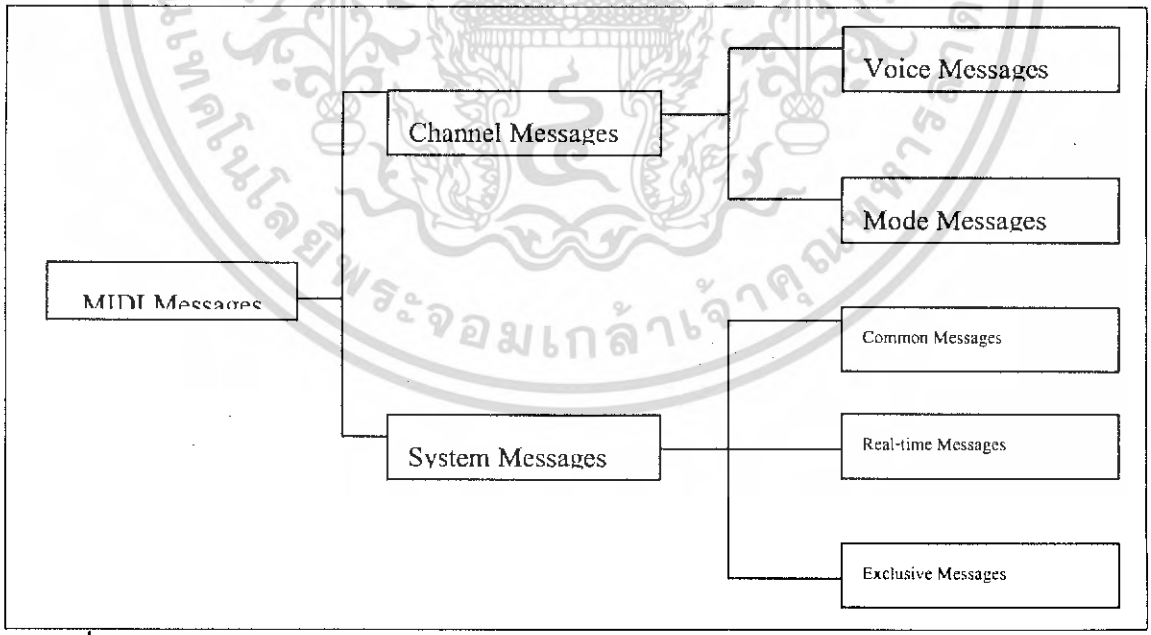
เดียวกัน ดังนั้นรูปแบบของเสียงอาจจะสามารถสร้างได้เป็น64ตัวโน๊ตในเวลาเดียวกันและ16 เสียงเครื่องดนตรี สำหรับเครื่องดนตรีส่วนใหญ่แล้วนั้น typical message อาจจะเป็นNote On,ประกอบด้วยช่องสัญญาณใด,pitch ใดและความเร็วเท่าไร สำหรับเครื่องดนตรีประเภทเคาะนั้น ข้อมูลpitchนั้นจะหมายถึงชนิดของกลอง

ในส่วนของNote On message นั้นส่วนประกอบของสถานะไบต์จะเป็นช่องสัญญาณใด pitchใด ตามด้วยไบต์ข้อมูลอีก2ไบต์ และทั้งหมดจะตามมาด้วย Note Off message ซึ่งจะมี pitch(ของโน๊ตที่จะปิดเสียง)และโดยทั่วไปvelocity จะเซ็ตเป็นศูนย์ โดยข้อมูลในไบต์สถานะในMIDIจะอยู่ระหว่าง128และ255 และในไบต์ข้อมูลอยู่ระหว่าง0ถึง127 แท้จริงแล้วไบต์ของMIDIคือ8บิต และรวมกับบิต0เริ่มต้นและสิ้นสุด ทำให้มีทั้งหมด10บิตใน1ไบต์

การตัดต่อMIDIนั้นจะอนุญาตให้สามารถทำงานด้วยความจำหรือความเข้าใจในมาตรฐานของเพลงตามที่ต้องการ ไฟล์MIDIสามารถที่จะเก็บไว้ในตารางข้อมูลWAVE ข้อดีของตารางข้อมูลWAVEคือจะมีการเก็บเสียงของเครื่องดนตรีอย่างถูกต้องแม่นยำ

2.3.3 โครงสร้างของมิดิ

ไฟล์MIDIสามารถแบ่งออกได้เป็น2ประเภทคือ channel messages และ system messages โดยในแต่ละชนิดของmessageประกอบด้วยmessageต่าง ๆ ดังภาพที่



ภาพที่ 2.7 แสดงสถาปัตยกรรมของมิดิเมสเสจ

2.3.3.1 channel messages

จะเป็นที่อยู่ของส่วนหนึ่งของmidi channel ซึ่งเป็นส่วนที่ใช้ติดต่อกับส่วนของ

เสียงดนตรี โดยจะมีทั้งหมด3ไบต์ ตัวแรกเป็นไบต์สถานะ และมันต้องเซ็ตบิตที่สำคัญที่สุดให้เป็น1 จากนั้น4บิตต่อมาจะบอกchannelจาก16channelที่เป็นไปได้ของmessageนี้ และ3บิตที่เหลือเป็นตัวholding กับmessage สำหรับดาต้าไบต์จะเซ็ตบิตที่สำคัญที่สุดเป็น0

2.3.3.1.1 voice messages

midi messages ทั่วไปส่วนใหญ่จะเป็น voice messagesเกือบทั้งหมด channel messageชนิดนี้ จะเป็นส่วนควบคุมเสียงทางด้าน การส่งข้อมูลที่แน่นอนที่เป็นตัวระบุช่องสัญญาณเพื่อจะเล่น หรือว่าปิด และจะถอดรหัสkey pressure ด้วย voice messageจะเป็นส่วนกำหนดตัวควบคุมผลกระทบของด้วย เช่น sustain vibrato tremolo

ส่วนของ note onและnote off massage นั้นvelocityจะเป็นตัวบอกว่าkeyจะถูกเล่นเร็วเท่าไหร่ รูปแบบของการตอบสนองตัวสังเคราะห์ ต่อความเร็ว(velocity)ที่มีมากขึ้นโดยการทำให้ เสียงดังขึ้น และกังวานใสขึ้น ในส่วนNote onนั้นจะเป็นส่วนที่ทำให้เกิดตัวโน๊ตขึ้น และตัวสังเคราะห์ก็จะพยายามสร้างเสียงโน๊ตตัวนั้น ให้มีความคล้ายกับเครื่องดนตรีขึ้นในขณะที่กำลังเล่นด้วย ในpressure message จะทำให้เลือกเสียงของตัวโน๊ตได้ในขณะที่โน๊ตกำลังเล่นอยู่

channel pressure message จะถูกบังคับให้เข้าจังหวะ(measure)สำหรับkey บนchannelที่เฉพาะเจาะจง(เครื่องดนตรี) และจะมีผลกระทบที่เฉพาะเจาะจง บนทุกโน๊ตที่กำลังเล่นอยู่บนchannel

ในpressure message อื่นเช่น polyphonic key pressure (หรือเรียกว่าkey pressure)นั้นจะเป็นตัวบอกว่า ระดับเสียงเท่าไหร่ที่keyที่ถูกเล่นด้วยกันนั้นจะมีความแตกต่างกันในแต่ละตัวโน๊ตที่อยู่ในคอร์ด pressureจะถูกเรียกว่าafftertouch ด้วย ตัวสร้างcontrol change จะเซ็ตตัวแปรcontroller(fader vibrato อื่นๆ) ในแต่ละตัวตัวประดิษฐ์อาจจะทำให้ใช้หมายเลขcontrollerที่แตกต่างสำหรับ งานที่ต่างกัน ทั้งๆที่controllerหมายเลข 1 จะมีความคล้ายกับการทำให้เสียงสูงต่ำก็ตาม

ตัวอย่าง เช่นnote on message อยู่หลังตาม2ไบต์ โดยไบต์แรกเป็นชื่อตัวโน๊ต และอีกตัวเป็นตัวบอกความเร็ว ดังนั้นเพื่อที่จะเล่นโน๊ตหมายเลข80ด้วยความเร็วสูงสุดบนchannel 13นั้น เครื่องมือmidi จะส่งตามหลังด้วย3ไบต์ของเลขฐานหกคือ&H9C &H50 &H7F จากนั้นโน๊ตจะถูกนับเช่น middle C มีหมายเลขคือ60

เพื่อที่จะเล่น2ตัวโน๊ตในขณะเดียวกัน อันดับแรกเราจะต้องส่งprogram change message สำหรับแต่ละ2channel การrecallคือวิธีการที่Program

change จะเดินทางไปเพื่อจะโหลดpatchที่เฉพาะเจาะจงสำหรับchannelนั้นๆ เพียงเท่านี้เราก็จะได้ 2 เสียงแตกต่างกันที่ติดมาด้วยกันบน2channel จากนั้นnote on message 2ตัวที่ส่งมาจะไปเปิดchannelของทั้งคู่ หรือมิฉะนั้นเราจะส่งแต่ละตัวnote on message ไปที่ในแต่ละchannelที่จำเพาะกันทีละตัวด้วยเสียงที่แตกต่างกัน ก่อนที่จะส่งnote off messageสำหรับโน็ตแรก หลังจากนั้นเราก็จะเล่นโน็ต2ตัว ในเวลาเดียวกันบนเครื่องมือเดียวกันโดยไม่มีผลซึ่งกันและกัน

polyphonic pressureจะเป็นตัวบอกว่ามีตัวโน็ตที่เล่นในเวลาเดียวกันบนเครื่องดนตรีหลายชิ้นนั้นกี่ตัว channel pressureจะบอกว่ามีตัวโน็ตเดี่ยวเท่าไหร่บนเครื่องดนตรีชิ้นเดียว

2.3.3.1.2 channel mode messages

channel mode message เป็นตัวที่ใช้ตัดสินใจว่าเครื่องดนตรีจะตอบสนองกับmidi voice messageอย่างไร ผลตอบสนองที่ได้จะเป็นแค่บอกchannelที่ถูกต้องเท่านั้น ไม่ใช่ตอบสนองทั้งหมด

ตารางที่ 2.1 แสดง status byte ของchannel messages

		Channel Voice Messages:
1000nnnn	2	Note Off event
1001nnnn	2	Note On event (velocity=0: Note Off)
1010nnnn	2	Polyphonic key pressure/after touch
1011nnnn	2	Control change
1100nnnn	1	Program change
1101nnnn	1	Channel pressure/after touch
1110nnnn	2	Pitch bend change

		Channel Mode Messages:
1011nnnn	2	Selects Channel Mode

2.3.3.2 System Messages

System Messagesจะไม่มีหมายเลขช่องสัญญาณ นั่นจะมีความหายต่อคำสั่งที่ไม่ระบุช่องสัญญาณ เช่นการจับเวลาสัญญาณสำหรับการสร้างความสอดคล้องการวางตำแหน่งข้อมูลในการตัดต่อMIDI และการติดตั้งข้อมูลสำหรับอุปกรณ์ปลายทางอย่างละเอียด ในsystem Messageจะเริ่มต้นด้วย"&HF." System Messages จะถูกแบ่งเป็น3ประเภท ได้แก่

(1) common messages

เป็น messages ซึ่งเกี่ยวข้องกับการจับเวลาหรือการวางตำแหน่ง โดยตำแหน่งของเพลงจะถูกวัดจากจังหวะ

(2) real-time messages

เป็น messages ซึ่งเกี่ยวข้องกับการเข้าจังหวะให้สอดคล้องกัน

(3) exclusive messages

ซึ่งจะรวมเอาสิ่งที่ผลิตขึ้นที่สามารถเพิ่มเติมมาตรฐานของMIDI และเป็นmessagesที่ถูกสมมุติให้เป็นส่วนสิ้นสุดโดยไบต์สิ้นสุดคือ"&HF7,"

ตารางที่ 2.2 แสดง status byte ของ system messages

Status Byte	Data Byte 1	Data Byte 2	Description
system commom			
11110010	0lllllll	0hhhhhhh	Song Position Pointer (l=least significant bit, h=most significant bit)
11110011	0sssssss		Song Select (s=song number)
11110110	none		Tune Request
11110111	none		EOX (end of system exclusive mssage)
system real time			
11111000	none		Timing Clock
11111010	none		Start (song)
11111011	none		Stop
11111110	none		Active Sensing
11111111	none		System Reset

บทที่ 3
ขั้นตอนการดำเนินงานวิจัย

ลำดับขั้นตอนในการดำเนินการวิจัยทั้งหมดแบ่งออกได้เป็น 3 ส่วนใหญ่ ได้แก่ 1. ส่วนของการวิเคราะห์และออกแบบโปรแกรม 2. ส่วนของการลงมือปฏิบัติการจริง 3. ส่วนของการทดสอบโปรแกรม ซึ่งแต่ละส่วนล้วนมีความสำคัญกับการพัฒนาโปรแกรมเมโลดี้ มูสเป็นอย่างมาก และแต่ละขั้นตอนประกอบด้วยขั้นตอนย่อย ๆ ดังนี้

1. ขั้นตอนการวิเคราะห์และออกแบบโปรแกรม

- การวิเคราะห์ความต้องการว่าจะสร้างโปรแกรมให้เป็นไปในรูปแบบใด เนื้อหาของโปรแกรมทั้งหมดควรมีลักษณะเป็นอย่างไร ออกแบบแนวทางในการใช้งาน และสิ่งที่ผู้ใช้โปรแกรมจะได้รับจากการใช้โปรแกรมนี้
- ออกแบบหน้าจอส่วนที่ใช้ติดต่อกับผู้ใช้โปรแกรม ซึ่งต้องเป็นรูปแบบที่ใช้งานง่าย เข้าใจได้ง่าย และสามารถดึงดูดความสนใจจากผู้ใช้โปรแกรมได้เป็นอย่างดี
- ออกแบบวิธีการจัดเก็บข้อมูลที่จำเป็นต้องใช้ภายในโปรแกรม โดยที่ต้องใช้พื้นที่ในการจัดเก็บน้อย จัดเก็บได้อย่างเป็นระเบียบ และสามารถนำข้อมูลออกมาใช้งานได้ง่าย เพื่อช่วยเพิ่มประสิทธิภาพการทำงานให้กับโปรแกรม
- ออกแบบวิธีการเขียนโปรแกรม และเลือกใช้ฮาร์ดแวร์ , ซอฟต์แวร์ รวมถึงคอมโพเนนต์และเครื่องมือต่าง ๆ อย่างเหมาะสม เพื่อนำมาใช้ในการสร้างโปรแกรม

2. ขั้นตอนการลงมือปฏิบัติการจริง

- ด้านการสร้างหน้าจออินเทอร์เฟส เพื่อใช้ติดต่อกับผู้ใช้โปรแกรม
- ด้านเสียงประกอบต่าง ๆ ภายในโปรแกรม การทำงานส่วนนี้ใน DirectMusic ในการนำไฟล์เสียงมาใช้งานในโปรแกรม
- ด้านโปรแกรมมิ่ง การทำงานในส่วนนี้จะเกี่ยวข้องกับการเขียนคำสั่งต่าง ๆ เพื่อเรียกใช้งานฟังก์ชันต่าง ๆ ที่สร้างไว้ ซึ่งมีขั้นตอนการทำงาน ดังนี้

    1. ใช้โปรแกรม Visual C++ 6.0 ในส่วนของการเขียนและแก้ไขซอร์สโค้ดของโปรแกรม
    2. ใช้ DirectX 9.0 เพื่อช่วยในการเขียนโปรแกรม
    3. เขียนคำสั่งที่ทำให้เกิดการกระทำตามที่ต้องการ

3. ขั้นตอนการทดสอบโปรแกรม

การทำงานในส่วนนี้ จะเป็นขั้นตอนในการทดสอบโปรแกรมที่พัฒนาขึ้นมาว่ามีความสมบูรณ์มากน้อยเพียงใด เพื่อหาข้อผิดพลาดและประเมินประสิทธิภาพของโปรแกรม เมื่อตรวจพบข้อผิดพลาดก็สามารถทำการแก้ไขได้โดยทันที แล้วจึงจัดทำเอกสารการวิจัยสรุปผลการทดลอง

3.1 ขั้นตอนการวิเคราะห์และออกแบบโปรแกรม

3.1.1 การออกแบบรูปแบบของโปรแกรม และลักษณะการใช้งาน

โปรแกรมเมโลดี้ มูส เป็นโปรแกรมที่ส่งเสริมทักษะทางด้านดนตรีและพัฒนาความคิดสร้างสรรค์ โดยมีรูปแบบของโปรแกรมคือ โปรแกรมจะจำลองเสียงของเครื่องดนตรีแต่ละชนิดไว้ แล้วให้ผู้ใช้งานเลือกว่าจะใช้เสียงของเครื่องดนตรีชนิดไหน เหมือนเป็นการจำลองเครื่องดนตรีมาไว้ในโปรแกรม เพื่อให้ผู้ใช้งานเลือกใช้เครื่องดนตรีได้เอง โดยที่ไม่มีการทำงานกับทฤษฎีทางดนตรีแต่อย่างใด

เมื่อผู้ใช้งานทำเพลงของตนเองเสร็จเรียบร้อยแล้ว ก็สามารถเก็บเอาไฟล์นั้น ๆ ไปเพื่อใช้งานได้ หรืออาจจะบันทึกเก็บไว้ เพื่อนำเพลงมาปรับปรุงใหม่ในโอกาสต่อไปก็เป็นได้

โปรแกรมเมโลดี้ มูส มีรูปแบบการใช้ที่ง่ายจึงสามารถตอบสนองความต้องการของผู้ใช้งานได้มากกว่าโปรแกรมการทำเพลงที่มีอยู่ทั่ว ๆ ไปในปัจจุบันนี้ เช่น Cake Walk Pro Audio หรือ Adobe Audition เป็นต้น ที่ผู้ใช้งานจำเป็นจะต้องมีพื้นฐานทางทฤษฎีดนตรีอยู่มากพอสมควร ประกอบกับต้องมีความเข้าใจในกระบวนการทำงานของระบบคอมพิวเตอร์มัลติมีเดียด้วย จึงไม่เหมาะกับการใช้งานสำหรับผู้ที่เริ่มต้นศึกษาในด้านดนตรี

3.1.2 การออกแบบหน้าจอส่วนที่ใช้ติดต่อกับผู้ใช้โปรแกรม

รูปแบบของหน้าจอที่ใช้ติดต่อกับผู้ใช้โปรแกรมนั้น มีการจำลองเครื่องดนตรีมาให้ คือ คีย์บอร์ด ซึ่งสามารถทำความเข้าใจในโน้ตต่าง ๆ ได้ง่ายกว่าเครื่องดนตรีชนิดอื่น โดยที่ผู้ใช้โปรแกรมสามารถใช้เมาส์คลิกเลือกที่แป้นของคีย์บอร์ดได้เลย ว่าจะเลือกเล่นที่คีย์ไหนบ้าง แล้วโปรแกรมจะเปล่งเสียงของคีย์นั้น ๆ ออกมาให้ได้ยิน หน้าจอจะถูกออกแบบให้สามารถทำความเข้าใจได้ง่าย และสื่อสารด้วยคำสั่งในการเลือกใช้การทำงานที่ตรงไปตรงมา โดยที่กลุ่มผู้ใช้งานเป้าหมายของโปรแกรมเมโลดี้ มูส คือ กลุ่มของเด็กและผู้เริ่มต้นศึกษาเรื่องดนตรี เมื่อหน้าจอในการติดต่อนั้นสามารถทำความเข้าใจได้ไม่ยาก ก็จะเป็นองค์ประกอบหนึ่งที่ช่วยให้กลุ่มเป้าหมายสนใจจะศึกษาต่อในด้านดนตรีและตัดสินใจเลือกใช้โปรแกรมนี้ด้วย

หน้าจอหลักที่ใช้ติดต่อกับผู้ใช้โปรแกรม ประกอบด้วย แถบเมนูเพื่อเข้าสู่โปรแกรม แป้นคีย์บอร์ดเพื่อสร้างเสียงดนตรี และส่วนของการปรับแต่งค่าต่าง ๆ ของโปรแกรม

ตารางที่ 3.1 แสดงหน้าที่ของอินเทอร์เฟสต่าง ๆ ในโปรแกรมเมโลดี้ มูส

Interface	ตัวเลือก	หน้าที่
Menu Bar		
File	New	เปิดหน้าจอการทำงานใหม่
	Open	เปิดงานที่บันทึกเก็บไว้
	Save	บันทึกงานลงในไฟล์เดิม
	Save As	บันทึกงานลงในไฟล์ใหม่
Note Range		
Note Range	-	เลือก Note Range ที่ต้องการใช้
Output Port		
Output Port	-	เลือก Output Port ที่ต้องการใช้
Instrument Selection		
Instrument Selection	-	เลือกเสียงเครื่องดนตรีที่ต้องการใช้
Keyboard		
Keyboard	-	เลือกใช้คีย์เสียงที่ต้องการ
Process Bar		
Process Bar	Record	บันทึกเพลงตามที่ผู้ใช้ต้องการ
	Stop	หยุดการบันทึกหรือเล่นเพลง
	Play	เล่นเพลง

3.1.3 การออกแบบเสียงและซาวน์เอฟเฟกต์ประกอบ

ไฟล์เสียงที่ใช้ภายในโปรแกรม คือ ไฟล์ MIDI ซึ่งเป็นเสียงของเครื่องดนตรีแต่ละประเภท โดยจะใช้การทำงานของ DirectMusic มาช่วย เพื่อนำไฟล์เสียงของเครื่องดนตรีมาใช้งาน

3.1.4 การออกแบบการจัดเก็บข้อมูล

ข้อมูลที่จำเป็นที่ใช้ในโปรแกรมนี้ที่จะต้องจัดเก็บ จะประกอบไปด้วยพารามิเตอร์เริ่มต้นต่าง ๆ ของโปรแกรม ไฟล์เสียงของเครื่องดนตรีแต่ละชนิดที่จะต้องนำมาใช้ในการทำเพลง

และไฟล์เพลงที่ผู้ใช้งานแต่งขึ้นมาเสร็จเรียบร้อยแล้ว สำหรับไฟล์ทั้งหมดนั้นต้องการให้เข้าถึงได้อย่างรวดเร็ว จึงได้เลือกวิธีการเก็บข้อมูลแบบไฟล์มาใช้ในการเก็บข้อมูล

3.1.5 การกำหนดฮาร์ดแวร์, ซอร์ฟแวร์ และเครื่องมือที่จะใช้

การเขียนโปรแกรมเมโลดี้ มูส ได้เลือกใช้คอมไพเลอร์ภาษา Visual C++ 6.0 มาใช้ร่วมกับ DirectX 9.0 ซึ่ง DirectX จะเป็นผู้ติดต่อกับฮาร์ดแวร์ต่าง ๆ โดยตรง โดยเฉพาะอุปกรณ์แสดงผลทั้งภาพ และเสียง จุดเด่นของ DirectX นั้นคือ สามารถทำงานติดต่อกับฮาร์ดแวร์ได้โดยไม่ต้องผ่านระบบปฏิบัติการ ทำให้มีประสิทธิภาพในการทำงานสูงขึ้น สามารถทำงานได้เร็วขึ้น ส่วนซอฟต์แวร์ที่ใช้ประกอบการสร้างฉากและภาพ ได้แก่ Adobe Photoshop CS และ Adobe Illustrator CS

3.2 ขั้นตอนการลงมือปฏิบัติจริง

3.2.1 สร้างหน้าจออินเทอร์เฟสเพื่อใช้ติดต่อกับผู้ใช้โปรแกรม

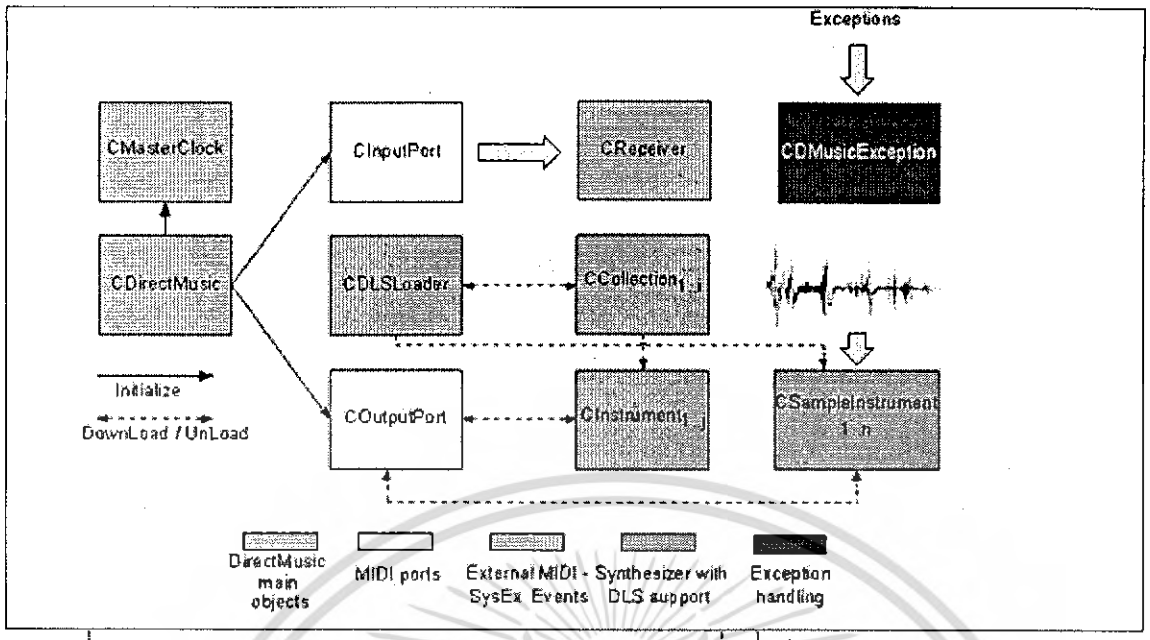
หน้าจออินเทอร์เฟสที่ใช้ในโปรแกรมเมโลดี้ มูสถูกสร้างขึ้นโดยใช้เทคโนโลยีเอ็มเอฟซี (MFC : Microsoft Foundation Class) ในการสร้างหน้าจออินเทอร์เฟส ซึ่งมีลักษณะดังภาพที่ 3.1

ภาพที่ 3.1 แสดงลักษณะหน้าจออินเทอร์เฟสเพื่อใช้ติดต่อกับผู้ใช้โปรแกรม

3.2.2 ด้านเสียงประกอบต่าง ๆ ภายในโปรแกรม

การทำงานส่วนนี้นำไดเรคมิวสิคและไดเรคมิดิคลาสไลบรารีมาใช้ในการนำไฟล์เสียงมาใช้งานในโปรแกรม ซึ่งมีขั้นตอนและหลักการต่าง ๆ ดังนี้

(1) ระบบการจัดการของไดเรคมิดิมีการทำงานหลักเกี่ยวกับคลาส 10 คลาสที่เกี่ยวข้องกัน ซึ่งเป็นตัวกำหนดความหมายให้กับออบเจ็กต์ที่แตกต่างกัน เพื่อให้แอพพลิเคชั่นที่สร้างขึ้นนั้นเข้าใจการทำงานกับไฟล์เสียงมิดิได้ ซึ่งมีลักษณะการใช้งานออบเจ็กต์ของแอพพลิเคชั่นที่ใช้ไดเรคมิดิคลาสไลบรารี ดังภาพที่ 3.1



ภาพที่ 3.2 แสดงลักษณะการใช้งานออบเจ็กต์ของแอพพลิเคชั่นที่ใช้ไดเรคมิดิคลาสไลบรารี

จากภาพที่ 3.1 มีออบเจ็กต์หลักเป็นของคลาสซีไดเรคมิวสิค (CDirectMusic) ออบเจ็กต์นี้รับผิดชอบการกำหนดค่าเริ่มต้นให้กับออบเจ็กต์ของมิดิพอร์ตที่แบ่งออกเป็น 2 ประเภท คือ 1. อินพุทพอร์ต (Input Port) และ 2. เอาท์พุทพอร์ต (Output Port) นอกจากนี้ยังมีออบเจ็กต์เพิ่มเติม คือ ซีมาสเตอร์คล็อก (CMasterClock) ซึ่งเป็นตัวระบุและเลือกใช้เวลาของฮาร์ดแวร์มาเป็นเวลาพื้นฐานของโปรแกรมนั่นเอง

มีออบเจ็กต์อีก 3 ออบเจ็กต์ที่เกี่ยวข้องกับคลาสซีเอาท์พุทพอร์ต (COutputPort) ทั้งโดยทางตรงและทางอ้อม คือ 1. คลาสซีดีแอลเอสโหลดเดอร์ (CDLSLoader) ที่รับผิดชอบในการโหลดแฟ้มข้อมูลดีแอลเอสมา เพื่อเก็บเข้าในออบเจ็กต์ของคลาสซีคอลเล็กชั่น (CCollection) ซึ่งออบเจ็กต์นี้เป็นตัวแทนของเครื่องดนตรีที่อยู่ในรูปแบบของดีแอลเอสหนึ่งจุดศูนย์และสองจุดศูนย์ (DLS 1.0/2.0) และยอมให้เครื่องดนตรีเหล่านี้ถูกย้ายไปเก็บอยู่ในออบเจ็กต์ที่เหมาะสมกว่าได้ คือ ออบเจ็กต์ของคลาสซีอินสตรูเมนท์ (CInstrument) ทั้งหมดนี้คือหน้าที่ความรับผิดชอบในการสร้างอินสแตนท์ของเครื่องดนตรี เพื่อการจัดการที่ดียิ่งขึ้น

เพิ่มเติมจากคลาสซีอินสตรูเมนท์นั้น จะมีออบเจ็กต์ลักษณะเดียวกันนี้ที่มีอยู่ในไดเรคมิดิไลบรารี ซึ่งมีหน้าที่เก็บข้อมูลที่อยู่ในรูปแบบของไฟล์เวฟ (.wav) และรับผิดชอบถึงการนำไปใช้ด้วย ออบเจ็กต์นี้คือ ซีแซมเปิลอินสตรูเมนท์ (CSampleInstrument) ซึ่งจะมีฟังก์ชันช่วยจัดการไฟล์ให้เรียบร้อยก่อนที่จะถูกส่งไปที่เอาท์พุทพอร์ต

สุดท้ายคือ คลาสซีมิวสิคเอ็กซ์เซพชั่น ที่จัดการเอ็กซ์เซพชั่นทั้งหมดที่ถูกสร้างขึ้นมา อีกทั้งยังบอกรายละเอียดเกี่ยวกับปัญหาที่ก่อให้เกิดข้อผิดพลาดเหล่านั้นด้วย

(2) ในการนำมิดิเมสเสจมาใช้งานนั้นมีคลาส CInputPort รับผิดชอบการจัดการกับมิดิเมสเสจที่ถูกส่งเข้ามา โดยมีเธรดเป็นตัวเรียกใช้ฟังก์ชัน 2 ฟังก์ชันที่ถูกโอเวอร์โหลดไว้ในคลาสซีรีซีฟเวอร์ (CReceiver) ซึ่งการเรียกใช้นั้นจะขึ้นอยู่กับประเภทของมิดิเมสเสจที่ถูกส่งมา ขั้นตอนการโอเวอร์ไรด์ฟังก์ชันนั้นต้องมีคลาสที่สืบทอดมาจากคลาสซีรีซีฟเวอร์ ซึ่งคำสั่งเป็นไปตามโค้ดดังต่อไปนี้

```
// Derived class from CReceiver

class CDMReceiver:public CReceiver
{
public:
    // Overriden functions
    void RecvMidiMsg(REFERENCE_TIME rt,DWORD dwChannel,DWORD
        dwBytesRead,BYTE *lpBuffer);
    void RecvMidiMsg(REFERENCE_TIME rt,DWORD dwChannel,DWORD dwMsg);
};
```

ทำการเขียนโปรแกรมให้ฟังก์ชันทั้ง 2 ฟังก์ชัน ดังโค้ดต่อไปนี้

```
// Overriden function for SysEx data capture
void CDMReceiver::RecvMidiMsg(REFERENCE_TIME lprt,DWORD dwChannel,
    DWORD dwBytesRead,BYTE *lpBuffer)
{
    DWORD dwBytecount;

    // Print the received buffer
    for (dwBytecount = 0;dwBytecount < dwBytesRead;dwBytecount++)
    {
        cout.width(2);
        cout.precision(2);
```

```
            cout.fill('0');
            cout << hex << static_cast<int>(lpBuffer[dwBytecount]) << " ";
            if ((dwBytecount % 20) == 0) cout << endl;
            if (lpBuffer[dwBytecount] == END_SYS_EX)
                cout << "\nSystem memory dumped" << endl;
        }
    }

    // Overriden function for structured MIDI data capture
    void CDMReceiver::RecvMidiMsg(REFERENCE_TIME lprt,DWORD dwChannel,
        DWORD dwMsg)
    {
        unsigned char Command,Channel,Note,Velocity;

        // Extract MIDI parameters from a MIDI message
        CInputPort::DecodeMidiMsg(dwMsg,&Command,&Channel,&Note,&Velocity);

        if (Command == NOTE_ON) //Channel #0 Note-On
        {
            cout << "Received on channel " << static_cast<int>(Channel) <<
                    " Note " << static_cast<int>(Note)
                << " with velocity " << static_cast<int>(Velocity) << endl;
        }
    }
```

ฟังก์ชันแรกจะจัดการเก็บข้อมูลไว้ในรูปแบบของเลขฐานสิบหกและคอยตรวจดูว่าข้อมูลจบแล้วหรือยัง ส่วนฟังก์ชันที่สองนั้นจะรับผิดชอบการจัดการข้อมูลที่เป็นมิดิเมสเสจธรรมดา เช่น note-on หรือ program-change ในรูปแบบของดับเบิลเวิร์ด และหากต้องการกระจายเมสเสจที่มีอยู่ออกเป็นส่วน ๆ สามารถทำได้โดยการเรียกใช้ฟังก์ชัน CInputPort :: DecodeMidiMsg

(3) ในขั้นตอนการกำหนดค่าเริ่มต้นให้กับออบเจ็กต์นั้น มีการประกาศออบเจ็กต์หลักที่จะถูกใช้ในแอพพลิเคชั่น แสดงได้ดังโค้ดต่อไปนี้

```
int main(int argc, char* argv[])
{
    CDirectMusic CDMusic;
    CInputPort  CInPort;
    CDMReceiver  Receiver;
    COutputPort  COutPort;
    CDLSLoader  CLoader;
    CCollection  CCollectionA,CCollectionB;
    CInstrument  CInstrument1,CInstrument2;
    CSampleInstrument CSample1,CSample2;

// Continues
```

ขั้นตอนการเรียกใช้เมท็อดเพื่อเริ่มต้นการใช้งานระบบมิดิทั้งหมด แสดงได้ดังโค้ดต่อไปนี้

```
    // Initialize DirectMusic
try
{
    CDMusic.Initialize();
    // Initialize ports given the DirectMusic manager object
    COutPort.Initialize(CDMusic);
    CInPort.Initialize(CDMusic);

// Continues
```

ขั้นตอนการเริ่มต้นใช้งานอินพุทพอร์ทและเอาท์พุทพอร์ท แสดงได้ดังโค้ดต่อไปนี้

```
INFOPORT PortInfo;
```

```
DWORD dwPortCount = 0;

// Software Synthesizer selection
do
    COutPort.GetPortInfo(++dwPortCount,&PortInfo);
while (!(PortInfo.dwFlags & DMUS_PC_SOFTWARESYNTH));

// Output port activation given the port information
COutPort.SetPortParams(0,0,1,SET_REVERB | SET_CHORUS,44100);
COutPort.ActivatePort(&PortInfo);
cout << "Selected output port: " << PortInfo.szPortDescription << endl;

// Input port activation, select the first one (by default)
CInPort.GetPortInfo(1,&PortInfo);
CInPort.ActivatePort(&PortInfo,SYSTEM_EXCLUSIVE_MEM);
cout << "Selected input port: " << PortInfo.szPortDescription << endl;

// Sets up the receiver object
CInPort.SetReceiver(Receiver);

getch();

// Continues
```

(4) ขั้นตอนการเริ่มต้นการใช้มิดิเมสเสจนั้นจะเรียกใช้ฟังก์ชัน CInPort :: ActivatePort แสดงได้ดังโค้ดต่อไปนี้

```
// Activates input MIDI message handling
CInPort.ActivateNotification();
// Redirects messages from source global channel 0 to destination
// global channel 0 over channel group 0 (channels 1-16)
```

```
CInPort.SetThru(0,0,0,COutPort);

// Continues
```

(5) ขั้นตอนการจบแอพพลิเคชั่น แสดงดังโค้ดต่อไปนี้

```
// Breaks the redirection
CInPort.BreakThru(0,0,0);
// Ends the notification
CInPort.TerminateNotification();

// Unloads the collections from the loader
CLoader.UnloadCollection(CCollectionA);
CLoader.UnloadCollection(CCollectionB);

// Unloads the instruments from the port
COutPort.UnloadInstrument(CInstrument1);
COutPort.UnloadInstrument(CInstrument2);

// Unloads the sample instruments
COutPort.UnloadInstrument(CSample1);
COutPort.UnloadInstrument(CSample2);

// Frees allocated memory
COutPort.DeallocateMemory(CSample1);
COutPort.DeallocateMemory(CSample2);

// Disposes the memory
delete [] pRawData;

// Exit
```

```
}
catch (CDMusicException& DMEx)
{
    cout << DMEx.GetErrorDescription() << endl;
}

return 0;
}
```

3.2.3 **การเขียนโปรแกรมรายละเอียดของฟังก์ชันการทำงานในส่วนต่าง ๆ ของโปรแกรม**

3.2.3.1 **ขั้นตอนของการบันทึกเสียงดนตรี** เมื่อกดปุ่มบันทึกเสียงแล้ว มีขั้นตอนการเขียนโปรแกรม ดังนี้

```
void CMidiStationDlg::OnRecord()
{
    m_MemProgress.PostMessage(PBM_SETRANGE32,0,
        rec.GetTotalMsgMemory());
    rec.ClearArray();
    rec.ResetIndex();
    m_Rec.EnableWindow(FALSE);
    m_Play.EnableWindow(FALSE);
    m_Stop.EnableWindow(TRUE);
    State = RECORDING;
    SetFocus();
}
```

3.2.3.2 **ขั้นตอนของการเล่นเพลงที่ได้บันทึกไว้** เมื่อกดปุ่มเล่นเพลงแล้ว มีขั้นตอนการเขียนโปรแกรม ดังนี้

```
void CMidiStationDlg::OnPlayBack()
{
```

```
if (rec.GetTotalRecorded()==0)
{
    AfxMessageBox(_T("Sorry, nothing to play"),
        MB_ICONINFORMATION);
    return;
}
m_Rec.EnableWindow(FALSE);
m_Play.EnableWindow(FALSE);
m_Menu->EnableMenuItem(IDR_NEW,MF_GRAYED);
m_MemProgress.PostMessage(PBM_SETRANGE32,0,
    rec.GetTotalRecorded()-1);
m_bExitPlayBack = FALSE;
State = PLAYING;
AfxBeginThread(ProcessMidiRecord,this);
// Begins the playback thread
}
```

3.2.3.3 **ขั้นตอนของการบันทึกไฟล์เพลงที่ได้สร้างไว้** เมื่อสร้างเพลงได้ตามที่ต้องการแล้ว สามารถบันทึกไฟล์เก็บไว้เพื่อเรียกใช้งานต่อไปได้ ซึ่งมีขั้นตอนการเขียนโปรแกรม ดังนี้

```
int CMidiStationDlg::SaveProcedure(BOOL bOnSave)
{
    _MMZHeader Header;
    INFOPORT InfoPort;

    CFileDialog FileDlg(FALSE,_T("mmz"),_T("*.mmz"),0,_T("Melody Muze
        (*.mmz)"));

    if (rec.GetTotalRecorded()==0)
    {
        AfxMessageBox(_T("Sorry, nothing to save"));
```

```
        return 0; // Failure
    }

    // Converts strings to the UNICODE counterpart

    #ifdef _UNICODE
        wcscpy(Header.Author,(LPCTSTR)m_strAuthor);
        wcscpy(Header.Title,(LPCTSTR)m_strTitle);
    #else
        mbstowcs(Header.Author,(LPCTSTR)m_strAuthor,
            m_strAuthor.GetLength() + 1);
        mbstowcs(Header.Title,(LPCTSTR)m_strTitle,
            m_strTitle.GetLength() + 1);
    #endif

    // Stores the selected instrument

    Header.dwPatch = m_InstList.GetCurSel() + 1;

    // Stores the desired output port

    COutPort.GetPortInfo(m_nOutPortSel + 1,&InfoPort);
    Header.dwFlags = InfoPort.dwFlags;

    // Stores the number of recorded items

    Header.Items = rec.GetTotalRecorded();

    if (bOnSave == FALSE)
    {
        if (FileDlg.DoModal() != IDCANCEL)
```

```
            CFileHandler::WriteMMZFile((LPCTSTR)(m_FileName =
                FileDlg.GetFileName()),&Header,rec.GetArrayPointer());
        }
        else CFileHandler::WriteMMZFile((LPCTSTR)m_FileName,&Header,
            rec.GetArrayPointer());

        return 1; // Success
    }
```

3.2.3.4 **ขั้นตอนของการเปิดไฟล์เพลงที่บันทึกเก็บไว้** เมื่อต้องการเปิดไฟล์เพลงที่บันทึกเก็บไว้ มีขั้นตอนของการเขียนโปรแกรม ดังนี้

```
    void CMidiStationDlg::OnOpen()
    {
        TCHAR strDest[60];
        _MMZHeader Header;

        CFileDialog FileDlg(TRUE,_T("mmz"),_T("*.mmz"),0
            ,_T("Melody Muze (*.mmz)"));

        // Opens a file dialog
        if (FileDlg.DoModal() != IDCANCEL)
        {
            // The dialog worked correctly
            rec.ClearArray(); // We don't need the old data

            // Reads the header and extracts the number of items

            if (!CFileHandler::ReadMMZFileHeader((LPCTSTR)(m_FileName =
                FileDlg.GetFileName()),&Header)) return;
```

```
if (strcmp(Header.Id,"MMZ"))
{
    AfxMessageBox(_T("Invalid MidiStation file!"),MB_ICONSTOP);
    return;
}

// Allocates space for the new array

rec.AllocateSpace(Header.Items);
rec.SetTotalRecorded(Header.Items);

// Reads the MIDI data and stores them into the array

If (!CFileHandler::ReadMMZFileData
    ((LPCTSTR)FileDlg.GetFileName(),rec.GetArrayPointer(),
    Header.Items*2*sizeof(WORD))) return;

// Establishes the selected output port

int dwNumPorts =static_cast<int>(COutPort.GetNumPorts());

for (int i=0;i<dwNumPorts;i++)
{
    if (m_OutPortList.GetItemData(i) == Header.dwFlags)
    {
        m_OutPortList.SetCurSel(i);
        OnSelchangeOutputPorts();
        break;
    }
}
```

```
// Establishes the selected instrument
m_InstList.SetCurSel(Header.dwPatch - 1);
OnSelchangeInstruments();

// Gets the author string
_tcscpy(strDest,_T("Author: "));

#ifdef _UNICODE
    wcscat(strDest,Header.Author);
#else
    TCHAR strConv[50]; // Converts author string to MBCS
    wcstombs(strConv,Header.Author,50);
    _tcscat(strDest,strConv);
#endif

_tcscat(strDest,_T("\0"));

SetDlgItemText(IDC_STATIC_AUTHOR,strDest);

// Gets the title string
_tcscpy(strDest,_T("Title: "));

#ifdef _UNICODE
    wcscat(strDest,Header.Title);
#else
    wcstombs(strConv,Header.Title,50); // Converts it to MBCS
    _tcscat(strDest,strConv);
#endif

_tcscat(strDest,_T("\0"));
```

```
            SetDlgItemText(IDC_STATIC_TITLE,strDest);

            // Sets the CRecord object to the start
            rec.ResetIndex();

            m_MemProgress.PostMessage(PBM_SETRANGE32,0,
                rec.GetTotalRecorded()-1);

            // State chart control

            switch(State)
            {
                case STOPPED:
                    m_Play.EnableWindow();
                    m_Stop.EnableWindow();
                    break;
                case RECORDING:
                    m_Play.EnableWindow();
                    m_Rec.EnableWindow();
                    State = STOPPED;
            }
            m_Menu->EnableMenuItem(IDR_SAVE,MF_ENABLED);
        }
    }
```

3.2.3.5 **ขั้นตอนของการเล่นไฟล์เพลงที่เปิดขึ้นมา** เมื่อเปิดไฟล์เพลงที่ต้องการขึ้นมาแล้ว สามารถเล่นไฟล์เพลงนั้น ๆ พร้อมกัน เพื่อให้เกิดเป็นเสียงเครื่องดนตรีหลายชิ้นได้ ซึ่งมีขั้นตอนการเขียนโปรแกรม ดังนี้

```
UINT CMidiStationDlg::ProcessMidiRecord(LPVOID lpParam)
{
```

```
DWORD dwMsg; // Millisecond counter
BYTE Note,Velocity,Channel,Command;
DWORD dwIndex,dwTime,dwTotalRecorded;

CSingleLock csl(&CS);

// Executes the thread while there are messages in the record object
    array
// and the user doesn't want to exit

csl.Lock();

rec.ResetIndex();
dwIndex = rec.GetIndex();
dwTotalRecorded = rec.GetTotalRecorded();

csl.Unlock();

while ((dwIndex < dwTotalRecorded) && (!pMSDlg->m_bAbandon) &&
    (!pMSDlg->m_bExitPlayBack))
{
    csl.Lock();                         // Protects the critical section

    dwMsg = rec.GetStoredMessage();
    try
    {
        COutPort.SendMidiMsg(dwMsg,0);
        //Sends the stored message to the port
    } catch(CDMusicException CDMusicEx)
    {
        OutputDebugString(CDMusicEx.GetErrorDescription());
```

```
}

CInputPort::DecodeMidiMsg(dwMsg,&Command,&Channel,
    &Note,&Velocity);

if (Command == NOTE_ON)
    {pMSDlg->m_Piano.DrawPianoKeyFromNote(Note);
    // Sends the note to the drawing function
    pMSDlg->m_MessageList.AddItemListCtrl
    (Note,Velocity,LIST_ITEM_BLUE);
    // Adds the messages to the list control
} else pMSDlg->m_Piano.ReleasePianoKeyFromNote(Note);

pMSDlg->m_MemProgress.PostMessage
    (PBM_SETPOS,rec.GetIndex(),0);

dwTime = rec.GetTime(rec.GetIndex() + 1);

csl.Unlock();

// Waits for the elapsed time between messages
if (dwIndex < dwTotalRecorded - 1 )
    WaitForSingleObject(pMSDlg->m_hEvent,dwTime);

csl.Lock();

rec.StepIndex();
dwIndex = rec.GetIndex();
dwTotalRecorded = rec.GetTotalRecorded();

csl.Unlock();
```

```
        }
        csl.Lock();
        pMSDlg->PostMessage(WM_END_PLAYBACK,0,0);
        csl.Unlock();
        return 0;
    }
```

3.3 ขั้นตอนการทดสอบโปรแกรม

เป็นขั้นตอนสุดท้ายของการพัฒนาโปรแกรม โดยขั้นตอนนี้เป็นการตรวจสอบความผิดพลาดและประสิทธิภาพโดยรวมของโปรแกรมทั้งหมด ว่าสามารถโต้ตอบและแสดงผลได้ตามที่ออกแบบไว้หรือไม่ ถ้าตรวจสอบพบความผิดพลาดจะได้นำข้อผิดพลาดนั้นมาปรับปรุงแก้ไขให้มีความถูกต้องมากที่สุด

บทที่ 4
ผลการทดสอบและการวิเคราะห์ปัญหา

การทดสอบและการวิเคราะห์ปัญหาที่จะกล่าวถึงในบทนี้ เป็นขั้นตอนของการทดสอบเพื่อบอกถึงรายละเอียดของผลที่ได้จากการทดสอบในแต่ละส่วนของโปรแกรมทั้งหมด โดยผลการทดสอบที่ได้นี้จะถูกนำไปวิเคราะห์ถึงปัญหาและแนวทางในการแก้ไขปัญหาเพื่อการพัฒนาต่อไปในอนาคต โดยทางผู้จัดทำหวังเป็นอย่างยิ่งว่าผลการทดสอบและแนวทางในการแก้ปัญหาที่ได้จากการทดสอบในครั้งนี้ จะเป็นประโยชน์ต่อผู้ที่สนใจ ในการนำไปศึกษาและค้นคว้าเพิ่มเติม รวมถึงสามารถมองเห็นปัญหาและข้อดี – ข้อเสียของปัญหานั้น ๆ และใช้เป็นแนวทางในการพัฒนาต่อไปได้

คุณสมบัติของระบบที่นำมาทดสอบ

1. ระบบปฏิบัติการ Microsoft Windows XP Professional
2. BIOS : ASUS – 42302e31
3. หน่วยประมวลผลกลาง Intel(R) Pentium(R) 4 CPU 1.8 GHz
4. หน่วยความจำหลักขนาด 256 MB
5. VGA Card ของ S3 Graphics ProsavageDDR
6. ระบบเสียง Sound Card ของ MPU-401 Compatible MIDI Device
7. หน่วยความจำสำรองขนาด 20 GB โดยมีพื้นที่เหลือว่าง 7.88 GB

ขั้นตอนการดำเนินการทดสอบ

1. ทำการติดตั้งตัวโปรแกรมและคอมโพเนนต์ที่จำเป็นต้องใช้
2. ทำการประมวลผลภายใต้ระบบที่กำหนด
3. ทดสอบความสมบูรณ์ของโปรแกรม
4. ประเมินประสิทธิภาพของโปรแกรม

จุดประสงค์ของการดำเนินการทดสอบ

1. หลังจากการติดตั้งตัวโปรแกรมแล้ว สามารถใช้งานโปรแกรมได้
2. การประมวลผลคำสั่งในหน้าจอต่าง ๆ ต้องเป็นไปตามที่กำหนด
3. ข้อผิดพลาดที่เกิดขึ้นจะต้องไม่มี หรือมีน้อยที่สุด

4.1 ขั้นตอนการดำเนินการทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็นต้องใช้

ตารางที่ 4.1 ขั้นตอนการดำเนินการทดสอบการติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็นต้องใช้

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
การติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็น	ทำการรันโปรแกรมโดยดับเบิลคลิกที่ไฟล์ "Setup.exe" ในแผ่นซีดี	1. สามารถติดตั้งโปรแกรมและคอมโพเนนต์ที่จำเป็นได้อย่างสมบูรณ์ 2. ผู้ใช้สามารถรันโปรแกรมได้

4.2 ขั้นตอนการดำเนินการทดสอบการประมวลผลภายใต้ระบบที่กำหนด

ตารางที่ 4.2 ขั้นตอนการดำเนินการทดสอบการประมวลผลภายใต้ระบบที่กำหนด

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
รันโปรแกรมภายใต้ระบบเครื่องคอมพิวเตอร์ที่กำหนด	ทำการรันโปรแกรมโดยดับเบิลคลิกที่ไฟล์ "MelodyMuze.exe" ในโฟลเดอร์ปลายทางที่ได้ทำการติดตั้ง	โปรแกรมสามารถทำงานได้โดยแสดงหน้าจอการทำงานขึ้นมา
ทดสอบสถานะของหน้าจอ	คลิกที่ปุ่มต่าง ๆ เพื่อเข้าสู่การทำงานในส่วนต่าง ๆ ของโปรแกรม	การเปลี่ยนสถานะของหน้าจอถูกต้องตามที่ได้ออกแบบไว้

4.3 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของโปรแกรม

4.3.1 การใช้เมนูบาร์

ตารางที่ 4.3 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของโปรแกรมในส่วนของการใช้เมนูบาร์

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
ทดสอบการเปิดไฟล์งานใหม่ (New)	1. คลิกเลือก File ที่เมนูบาร์ 2. คลิกเลือก New	โปรแกรมแสดงหน้าจอการทำงานใหม่ขึ้นมา
ทดสอบการเปิดไฟล์งานที่เคยทำไว้ (Open)	1. คลิกเลือก File ที่เมนูบาร์ 2. คลิกเลือก Open 3. เลือกเปิดไฟล์งานตามที่ต้องการ	โปรแกรมเปิดไฟล์งานที่เคยทำไว้ขึ้นมาได้อย่างถูกต้อง
ทดสอบการบันทึกไฟล์งานที่กำลังทำอยู่ (Save)	1. คลิกเลือก File ที่เมนูบาร์ 2. คลิกเลือก Save 3. ตั้งชื่อไฟล์งานที่ต้องการบันทึก แล้วจึงบันทึก	โปรแกรมแสดงหน้าจอการบันทึกงาน เพื่อให้กรอกชื่อของไฟล์และไดเรคเทอรีที่จะบันทึกแล้วจึงบันทึกไฟล์เก็บไว้
ทดสอบการบันทึกงานที่กำลังทำอยู่ โดยไม่บันทึกทับในไฟล์เดิม (Save As)	1. คลิกเลือก File ที่เมนูบาร์ 2. คลิกเลือก Save As 3. ตั้งชื่อไฟล์งานที่ต้องการบันทึก แล้วจึงบันทึก	โปรแกรมแสดงหน้าจอการบันทึกงาน เพื่อให้กรอกชื่อของไฟล์และไดเรคเทอรีที่จะบันทึกแล้วจึงบันทึกไฟล์เก็บไว้

4.3.2 การกำหนดโน้ตเรนจ์

ตารางที่ 4.4 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของโปรแกรมในส่วนของการกำหนดโน้ตเรนจ์

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
ทดสอบการกำหนดโน้ตเรนจ์	คลิกเลือกโน้ตเรนจ์ให้ครบทุกตัวเลือก เพื่อดูว่าสามารถเลือกได้หรือไม่	สามารถเลือกได้ทุกตัวเลือก

4.3.3 การกำหนดอินสตรูเมนท์ ซีเลคชัน

ตารางที่ 4.5 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของโปรแกรมในส่วนของการกำหนดอินสตรูเมนท์ ซีเลคชัน

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
ทดสอบการกำหนดอินสตรูเมนท์ ซีเลคชัน	คลิกเลือกอินสตรูเมนท์ให้ครบทุกตัวเลือก เพื่อดูว่าสามารถเลือกได้หรือไม่	สามารถเลือกได้ทุกตัวเลือก

4.3.4 การกดแป้นคีย์บอร์ด

ตารางที่ 4.6 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของโปรแกรมในส่วนของการกดแป้นคีย์บอร์ด

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
ทดสอบการกดแป้นคีย์บอร์ด	คลิกที่แป้นคีย์บอร์ด เพื่อทดลองว่ามีเสียงดนตรีดังออกมาหรือไม่	มีเสียงดนตรีดังออกมาตามคีย์ที่คลิกเลือก

4.3.5 การใช้โพรเซสบาร์

ตารางที่ 4.7 ขั้นตอนการดำเนินการทดสอบความสมบูรณ์ของโปรแกรมในส่วนของการใช้โพรเซสบาร์

การทดสอบ	ขั้นตอนการทดสอบ	ผลการทดสอบ
ทดสอบการใช้โพรเซสบาร์	1. กดปุ่มบันทึกเสียง 2. คลิกที่แป้นคีย์บอร์ด เพื่อบันทึกเสียงตามที่ต้องการ 3. เมื่อได้เพลงตามที่ต้องการแล้ว จึงกดปุ่มหยุดการบันทึก 4. กดปุ่มเล่นเพลงที่ได้บันทึกไว้	สามารถบันทึกเพลง และเปิดเพลงนั้นขึ้นมาเล่นใหม่ได้

4.5 ปัญหาและแนวทางการแก้ไข

1. เฮดเดอร์ของไฟล์ .mmz ประกอบด้วย ข้อมูลของเครื่องดนตรี และเอาท์พุตพอร์ท ที่เลือกทำให้ไฟล์ .mmz มีขนาดเล็กกว่าไฟล์ midi มาตรฐาน เพราะไม่ต้องเก็บข้อมูลของเครื่องดนตรีในทุกๆโน้ตที่มี แต่นำมาเก็บที่เฮดเดอร์เพียงที่เดียว ขณะทำการพัฒนาโปรแกรม เมื่อทดสอบการเล่นเสียงดนตรีขณะมีเครื่องดนตรีเพียงชิ้นเดียว สามารถเล่นได้ไม่พบปัญหาใดๆ

2. เมื่อพัฒนาให้สามารถเล่นหลายเครื่องดนตรี โดยใช้วิธีการเขียนเธรด เพื่อเล่นเสียงเครื่องดนตรีแต่ละชิ้นไปพร้อมกัน ผลปรากฏว่าสามารถเล่นไปอย่างพร้อมกันได้ แต่เสียงของเครื่องดนตรีที่แตกต่างกันนั้นถูกเปล่งเสียงออกมาเป็นเสียงเครื่องดนตรีชิ้นเดียวกัน เนื่องจากการเลือกเสียงเครื่องดนตรีนั้น มาจากการอ่านค่าเสียงเครื่องดนตรีที่เลือกไว้ ซึ่งเก็บไว้ในเฮดเดอร์ของไฟล์ ดังนั้นเมื่อเปิดไฟล์เครื่องดนตรีขึ้นมาเสียงเครื่องดนตรีในเฮดเดอร์ก็จะไปแทนเสียงเดิมที่เลือกไว้จากไฟล์ก่อนหน้านั้น ทำให้เมื่อเล่นเสียงเครื่องดนตรีพร้อมกันทุกชิ้นก็จะเปล่งเสียงเครื่องดนตรีจากไฟล์เครื่องดนตรีสุดท้ายที่ถูกเปิดขึ้นมา

3. การแก้ปัญหาดังกล่าวโดยเขียนโปรแกรมให้สร้างอ็อปเจคของเอาท์พุทพอร์ตขึ้นมาใหม่สำหรับเครื่องดนตรีทุกชิ้น จากเดิมที่ทุกเครื่องดนตรีจะทำงานบนสร้างอ็อปเจคของเอาท์พุทพอร์ตตัวเดียวกัน ผลปรากฏว่าสามารถเล่นไฟล์เครื่องดนตรีแยกกันบนอ็อปเจคของเอาท์พุทพอร์ตแต่ละตัวได้ แต่เสียงเครื่องดนตรีจะออกมาแค่ไฟล์เดียวเท่านั้นเพราะเครื่องจะยอมให้แอ็กติเวทอ็อปเจคของเอาท์พุทพอร์ตได้เพียงครั้งละ 1 อ็อปเจคเท่านั้น ดังนั้นจึงไม่สามารถใช้วิธีนี้แก้ไขปัญหาดังกล่าวได้

4. สำหรับแนวทางในการพัฒนาโปรแกรมลักษณะนี้ต่อไป คือต้องออกแบบการจัดเก็บไฟล์ในลักษณะที่ไม่มีการแยกเอาข้อมูลของเครื่องดนตรีของโน้ตแต่ละโน้ตออกมาไว้ที่เฮดเดอร์ ถึงแม้ว่าจะเป็นข้อมูลที่เหมือนกันในทุกๆโน้ต ผลที่คาดว่าจะได้คือ โปรแกรมจะเปล่งเสียงเครื่องดนตรีได้ครบทุกเสียงพร้อมกันตามที่ต้องการได้ เนื่องจากมีรายละเอียดของเสียงเครื่องดนตรีกำกับไปในทุกๆตัวโน้ต แต่จะต้องแลกกับขนาดไฟล์ที่จะใหญ่ขึ้นตามด้วย

5. การจัดเก็บไฟล์ในรูปแบบ midi มาตรฐาน ก็เป็นหนึ่งในรูปแบบการเก็บไฟล์ดังที่กล่าวอาจจะเป็นทางเลือกที่ดีสำหรับผู้ที่จะพัฒนาโปรแกรมลักษณะนี้ เนื่องจากไฟล์ที่ได้ออกมาจะสามารถนำไปเปิดกับโปรแกรมเล่นเสียงดนตรีหรือเพลง ทั่วไปได้ และสามารถแก้ไขปัญหาที่ปรากฏดังที่กล่าวมาด้านบนได้

6.ในการออกแบบรูปแบบของการจัดเก็บไฟล์ในโปรแกรมลักษณะเล่นหลายเสียงพร้อมกันนั้น เมื่อใช้สัญญาณแบบ MIDI จะพบว่าถึงแม้ byte ที่ใช้ระบุเสียงเครื่องดนตรีจะเหมือนกันในทุกโน้ต แต่ไม่ควรจัดเก็บในรูปแบบนำมาจัดเก็บรวมกันที่ header เพราะจะเกิดปัญหาที่แก้ไขได้ยาก ดังเช่นปัญหาพิเศษนี้

7.เมื่อพิจารณาจากปัญหาพิเศษนี้แล้วพบว่า ข้อดีของการจัดเก็บไฟล์ในรูปแบบดังกล่าวจะมีประโยชน์มากขึ้น หากนำในรูปแบบไฟล์นั้นไปใช้กับอุปกรณ์เครื่องดนตรีอิเลคโทนประเภทต่างๆ เช่น คีย์บอร์ด เป็นต้น เนื่องจากไฟล์ที่เก็บไว้นั้นจะมีขนาดเล็กกว่าไฟล์ MIDI ซึ่งอุปกรณ์จำพวกอิเลคโทนนั้นไม่มีการใช้หลักการของการเปล่งเสียงหลายเครื่องดนตรีในเวลาเดียวกันในอุปกรณ์ชิ้นเดียว

4.4 ประเมินประสิทธิภาพของโปรแกรม

จากการทดสอบดังที่ผ่านมาทั้งหมดทำให้เราสามารถประเมินประสิทธิภาพโดยรวมของโปรแกรมได้ว่า โปรแกรมนี้สามารถทำงานได้ตามที่ออกแบบไว้ได้สมบูรณ์พอสมควร โดยมีข้อผิดพลาดเล็กน้อยที่เกิดขึ้น จากการทดสอบด้วยระบบที่กำหนดไว้ข้างต้นนั้น อาจจะสามารถเปลี่ยนแปลง เพิ่มหรือลดประสิทธิภาพได้ ถ้าหากมีการเปลี่ยนระบบที่ใช้ในการประมวลผลของโปรแกรม และเนื่องจากโปรแกรมนี้มีการติดต่อกับเสียงค่อนข้างมาก ทำให้ระบบที่จะนำโปรแกรมนี้ไปประมวลผลนั้น ต้องมีประสิทธิภาพสูงพอสมควร

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

5.1.1 การศึกษาและเก็บรวบรวมข้อมูล

การศึกษาและเก็บรวบรวมข้อมูลที่จำเป็นต้องใช้ในการพัฒนาโปรแกรมเริ่มต้นด้วย การเก็บรวบรวมข้อมูลในขั้นต้นเพื่อพิจารณาความเป็นไปได้ในการพัฒนาโปรแกรมในเครื่องคอมพิวเตอร์ เมื่อคิดวิเคราะห์และสามารถกำหนดขอบเขตของโปรแกรมนี้ได้แล้ว จึงเข้าสู่ขั้นตอนการออกแบบรูปแบบการทำงานของโปรแกรมตามขอบเขตที่ได้กำหนดไว้ในข้างต้น ซึ่งในการออกแบบนั้นมีความจำเป็นที่จะต้องศึกษาขั้นตอนต่าง ๆ ในการพัฒนาโปรแกรม ขั้นตอนในการออกแบบหน้าจอติดต่อกับผู้ใช้โปรแกรม และการใช้งานเครื่องมือต่าง ๆ ที่จำเป็นประกอบกันไปด้วย เพื่อให้ผู้ออกแบบสามารถมองเห็นภาพรวมของโปรแกรมได้ดียิ่งขึ้น จากนั้นจึงรวบรวมข้อมูลทั้งหมดเพื่อนำไปสร้างขึ้นเป็นโครงสร้างของโปรแกรม และด้วยข้อมูลที่ได้เก็บรวบรวมมาตั้งแต่ต้นนั้นจะช่วยให้การพัฒนาโปรแกรมเป็นไปอย่างมีระบบ สามารถเลือกใช้เครื่องมือต่าง ๆ ได้อย่างเหมาะสม และสนับสนุนให้การพัฒนาโปรแกรมเป็นไปอย่างมีประสิทธิภาพ

5.1.2 การวิเคราะห์และออกแบบโปรแกรม

โปรแกรมเมโลดี้ มูสมีจุดประสงค์หลักเพื่อตอบสนองความต้องการของผู้ที่มีความสนใจทางด้านดนตรี แต่ยังไม่มีทักษะทางทฤษฎีดนตรีมากนัก คนกลุ่มนี้จึงไม่สามารถใช้งานโปรแกรมทำเพลงในลักษณะคล้ายกันนี้ที่มีอยู่ทั่วไปได้ เพราะฉะนั้นรูปแบบของโปรแกรมจึงควรมีหน้าจอติดต่อกับผู้ใช้โปรแกรมที่เข้าใจง่าย ไม่ยุ่งยากซับซ้อนแต่สามารถดึงดูดผู้ใช้โปรแกรมได้ มีสัญลักษณ์ที่มองแล้วสามารถทำความเข้าใจได้ไม่ยาก และโปรแกรมมีฟังก์ชันการทำงานที่ไม่ยุ่งยากมากจนเกินไป ในขั้นตอนของการออกแบบโปรแกรมนี้จะนำเอาข้อมูลต่าง ๆ ที่ได้จากการศึกษาค้นคว้าข้างต้นมาใช้เพื่อประกอบการออกแบบด้วย เพื่อให้รูปแบบของโปรแกรมมีโครงสร้างที่เป็นระบบ เมื่อออกแบบโปรแกรมเสร็จเรียบร้อยแล้วจึงมอบหมายงานส่วนต่าง ๆ ให้กับสมาชิก เพื่อเข้าสู่ขั้นตอนการพัฒนาโปรแกรมต่อไป

5.1.3 การพัฒนาโปรแกรม

ขั้นตอนนี้เป็นขั้นตอนที่สำคัญและใช้ระยะเวลานานที่สุดในการพัฒนาโปรแกรม เนื่องจากขั้นตอนนี้เป็น การนำความรู้ที่เรารวบรวมมาได้ทั้งหมด มาประกอบเข้ากับโครงสร้างและรูปแบบการใช้งานของโปรแกรมที่ได้ออกแบบไว้ เมื่อได้ลงมือพัฒนาโปรแกรมจริงก็ต้องมีการค้นหาข้อมูลเพิ่มเติมนอกเหนือจากที่ได้ค้นคว้าไว้ในตอนแรก เพื่อให้สามารถพัฒนาโปรแกรมให้มีการทำงานเป็นไปตามอัลกอริธึมที่ออกแบบไว้ เมื่อพบกับปัญหาต่าง ๆ เช่น โปรแกรมทำงานไม่ตรงตามความต้องการ ก็ต้องคิดวิเคราะห์หาสาเหตุของปัญหาเหล่านั้น แล้วหาทางแก้ไขปัญหานั้น ๆ ให้ผ่านพ้นไปให้ได้ และต้องทำการทดสอบการทำงานของโปรแกรมที่พัฒนาอยู่เสมอ ๆ ด้วยเหตุผลเหล่านี้จึงส่งผลให้ต้องใช้เวลาในขั้นตอนนี้มากพอสมควร ซึ่งขั้นตอนในการทำงานส่วนนี้มีซอร์ฟแวร์และเครื่องมือที่ใช้ในการพัฒนาโปรแกรมดังนี้ Microsoft Visual C++ 6.0 , DirectX 9.0 Runtime และ DirectX 9.0 SDK

5.2 ข้อจำกัดของโปรแกรม

โปรแกรมเมโลดี้ มูส เป็นโปรแกรมที่พัฒนาขึ้นบนพื้นฐานโครงสร้างของ Microsoft Windows โดยที่โปรแกรมนี้มีการทำงานร่วมกับเสียงค่อนข้างมาก ส่งผลให้โปรแกรมนี้ต้องทำงานอยู่บนเครื่องคอมพิวเตอร์ที่มีประสิทธิภาพสูงพอสมควร โดยรายละเอียดของข้อจำกัดของโปรแกรมมีดังต่อไปนี้

1. ระบบของเครื่องคอมพิวเตอร์ที่ใช้รันโปรแกรมนี้ต้องมีประสิทธิภาพสูง เนื่องจากโปรแกรมจำเป็นต้องประมวลผลเกี่ยวกับเสียงค่อนข้างมาก
2. เครื่องคอมพิวเตอร์ที่ใช้ควรเป็นเครื่อง Personal Computer หรือเป็นเครื่อง Laptop Computer ที่ใช้พลังงานจากไฟฟ้า เนื่องจากระบบปฏิบัติของเครื่อง Laptop Computer ที่ใช้พลังงานจากแบตเตอรี่นั้นจะลดประสิทธิภาพการประมวลผลด้านกราฟิกและเสียงลงเพื่อช่วยให้ประหยัดพลังงานของแบตเตอรี่
3. โปรแกรมเมโลดี้ มูส ต้องการ DirectX เวอร์ชัน 8.0 ขึ้นไป
4. การบันทึกไฟล์เพลงที่สร้างขึ้นด้วยโปรแกรมเมโลดี้ มูสนั้นจะต้องบันทึกในแบบรีล ไทม์ (Real Time) เท่านั้น คือ บันทึกตามลักษณะการกดคีย์บนคีย์บอร์ดของผู้ใช้โปรแกรมเลย ว่าเว้นช่วงระหว่างคีย์มากน้อยแค่ไหน
5. ห้ามเคลื่อนย้ายโฟลเดอร์ต่าง ๆ ออกจากตำแหน่งที่ติดตั้งโปรแกรม เนื่องจากในระหว่างการใช้งานโปรแกรมต้องเรียกใช้ข้อมูลที่อยู่ในโฟลเดอร์นั้น ๆ โดยหากโปรแกรมไม่สามารถหาข้อมูลที่มีความจำเป็นต้องใช้ในการทำงานได้นั้นก็จะก่อให้เกิดความผิดพลาดของโปรแกรมได้

5.3 ข้อเสนอแนะ

เนื่องจากโปรแกรมเมโลดี้ มูสเป็นรูปแบบของภาษาซีพลัสพลัส (C++) โดยใช้โปรแกรมวิชวลซีพลัสพลัส (Visual C++ 6.0) มาประกอบกับการทำงานของไดเรคเอ็กซ์ (DirectX 9.0) ในส่วนของไดเรคมิวสิค (DirectMusic) ซึ่งซอร์ฟแวร์ต่าง ๆ ที่นำมาใช้นี้ล้วนแต่มีไลบรารีที่ช่วยสนับสนุนการพัฒนาโปรแกรมให้ง่ายขึ้น แต่เนื่องจากไม่ได้เป็นไลบรารีพื้นฐานที่คนทั่วไปรู้จักกันและมีฟังก์ชันการทำงานอยู่มากมายหลายประเภท จึงต้องค้นคว้าหาข้อมูลและศึกษาอย่างละเอียดก่อนนำมาใช้จริง เพื่อค้นหาไลบรารีและฟังก์ชันการทำงานที่เหมาะสมกับการพัฒนาโปรแกรมนี้มากที่สุด

บรรณานุกรม

นิรุธ อำนวยศิลป์. 2542. **คู่มือการเขียนโปรแกรม Microsoft Visual C++ Version 6.0 ฉบับเพื่อการใช้งานจริง.** กรุงเทพฯ : บริษัท ซัคเซส มีเดีย จำกัด.

ยุทธนา ลีลาศวัฒนกุล. 2547. **เริ่มต้นการเขียนโปรแกรมด้วยภาษา C++.** กรุงเทพฯ : บริษัท ดวงกมลสมัย จำกัด.

ยุทธนา ลีลาศวัฒนกุล. 2546. **คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ .NET ฉบับสมบูรณ์.** นนทบุรี : อินโฟเพรส.

ยุทธนา ลีลาศวัฒนกุล. 2544. **คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ 6.0 ฉบับโปรแกรมเมอร์.** กรุงเทพฯ : อินโฟเพรส.

มานพ พรเพียรวิชานนท์. 2544. **ก้าวสู่ Game Developer มือโปรกับ DirectX.** กรุงเทพฯ : บริษัท วิตตี้ กรุ๊ปป จำกัด.

เดชฤทธิ์ พลเยี่ยม. 2547. **คู่มือคอมพิวเตอร์ดนตรี "ฉบับผู้เริ่มต้น".** กรุงเทพฯ : อินโนเวชั่น มีเดีย พริ้นติ้ง