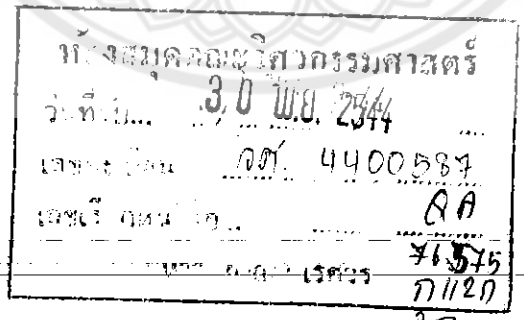
การเขียนโปรแกรมเกมบนวินโดว์สด้วยไดเรคเอกซ์

GAME PROGRAMMING WITH DIRECTX

นายกชพงศ์ เพ็ชรราช รหัส 41360215

นางสาวอรุณี นาคผสม รหัส 41360553



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2544

ใบรับรองโครงงานวิจัย

หัวข้อโครงงาน	การเขียนโปรแกรมเกมบนวินโดว์สด้วยไดเรคเอกซ์		
ผู้ดำเนินโครงงาน	นายกชพงศ์	เพ็ชรราช	รหัส 41360215
	นางสาวอรุณี	นาคผสม	รหัส 41360553
อาจารย์ที่ปรึกษา	อาจารย์วัชร์วีร์	พืชพันธ์	
สาขา	วิศวกรรมคอมพิวเตอร์		
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
ปีการศึกษา	2544		

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตร วิศวกรรมศาสตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิจัย

.....ประธานกรรมการ

(อาจารย์สิทธิโชค เชาวกุล)

.....กรรมการ

(อาจารย์ประทีป ตรีรณโอภาส)

.....กรรมการ

(อาจารย์วัชร์วีร์ พืชพันธ์)

.....กรรมการ

(อาจารย์ศิริพร เดชาศิลารักษ์)

หัวข้อโครงงาน	การเขียนโปรแกรมเกมบนวินโดว์สด้วย DirectX
ผู้ดำเนินโครงงาน	นายกชพงศ์ เพ็ชรราช รหัส 41360215
	นางสาวอรุณี นาคผสม รหัส 41360553
อาจารย์ที่ปรึกษา	อาจารย์วัชร์วีร์ พืชพันธ์
สาขา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2544

บทคัดย่อ

โครงการนี้เป็นการศึกษาพัฒนาโปรแกรมเกมซึ่งได้ทำงานบนระบบปฏิบัติการวินโดว์สโดยใช้ DirectX เป็นเครื่องมือในการช่วยพัฒนาโดย DirectX มีหน้าที่ช่วยในการจัดการ เกี่ยวกับการแสดงผลออกทางจอภาพ ทั้งกราฟิกแบบ 2 มิติ และ 3 มิติ ระบบเสียงประกอบขณะเล่นเกม การรับอินพุตต่างๆ จากผู้เล่น เช่น เมาส์ คีย์บอร์ด จอยสติ๊ก และการติดต่อผ่านเน็ตเวิร์ค เช่นการเล่นเกมแบบหลายผู้เล่น (Multiplayer Games)

โปรแกรมที่ใช้พัฒนาโครงงานนี้ คือ Microsoft Visual C++ โดยจะเป็นตัวติดต่อกับObject ต่างๆ ของ DirectX ซึ่งเป็นการเรียกใช้ความสามารถของ DirectX ผ่านทางโปรแกรม Microsoft Visual C++ นอกจากนี้ยังมีการใช้โปรแกรม Adobe Photoshop 5.5 และ Paintshop Pro ในการสร้างตัวละครสร้างฉากหลัง แต่งภาพ โปรแกรม Cool Edit ใช้ในการทำอัดเสียงตัวละคร

Project Title	Game Programming With DirectX
Name	Mr. Kochapong Petcharaj ID. 41360215
	Miss Arunee Nakpasom ID. 41360553
Project Advisor	Mr. Wachawee Phuetphan
Major	Computer Engineering
Department	Electrical and Computer Engineering
Academic Year	2001

ABSTRACT

This project is studiee and develop program game for windows operating system and used DirecX to developing. This enginc(DirectX) have function to manage about display, graphic interface, image processing both 2 and 3 dimension. Sound effect during game.Receive input from user such as mouse, keyboard, joystick and networking as multiplayer games.

The program that used to develop this project is Microsoft Visual C++ . This program is call DirectX Object to use many function of DirectX . Not only that, Another program have to used sush as Adobe Photoshop 5.5, Painshop Pro.That program used to build player object, sprite, background, edit image .Cool Edit program used to make the sound of characters.

กิตติกรรมประกาศ

ขอขอบพระคุณทุกๆ ท่านที่ทำให้โครงงานชิ้นนี้ประกอบกันจนเสร็จสิ้นไปในระดับหนึ่ง โดยเฉพาะอาจารย์วัชร์วีร์ พืชพันธ์ อาจารย์ที่ปรึกษาโครงงานชิ้นนี้ และขอบคุณเพื่อนๆ ทุกคนที่ให้กำลังใจ

ขอขอบพระคุณศูนย์เทคโนโลยีและอิเลกทรอนิกส์ สำนักงานพัฒนาวิทยาศาสตร์และเทคโนโลยีแห่งชาติ ที่ให้ทุนสนับสนุนโครงงานเกมท่องเที่ยวภาคเหนือ ในโครงการแข่งขันพัฒนาโปรแกรมคอมพิวเตอร์แห่งประเทศไทย ครั้งที่ 3

กชพงศ์ เพ็ชรราช

อรุณี นาคผสม

สารบัญ

หน้า

บทคัดย่อภาษาไทย.....ก

บทคัดย่อภาษาอังกฤษ.....ข

กิตติกรรมประกาศ.....ค

สารบัญ.....ง

สารบัญตาราง.....ฉ

สารบัญรูป.....ช

บทที่ 1 บทนำ

1.1 ที่มาและความสำคัญของโครงงาน.....1

1.2 วัตถุประสงค์ของโครงงาน.....1

1.3 ขอบข่ายของโครงงาน.....1

1.4 ขั้นตอนการดำเนินงาน.....2

1.5 ผลที่คาดว่าจะได้รับ.....2

1.6 งบประมาณที่ใช้.....2

บทที่ 2 การเขียนโปรแกรมด้วยไดเรคเอ็กซ์

2.1 การเขียนโปรแกรมบนวินโดว์สด้วย Microsoft Visual C++.....3

2.2 DirectX.....8

2.3 DirectDraw.....9

2.4 DirectSound.....21

2.5 DirectPlay.....26

2.6 DirectMusic.....29

บทที่ 3 การออกแบบและพัฒนาโปรแกรม

3.1 การออกแบบแนวเกมและการดำเนินเรื่องของเกม.....33

3.2 การจัดเตรียมทรัพยากรและข้อมูลต่างๆ เพื่อนำมาใช้ในการสร้างเกม.....33

3.3 การเขียนโปรแกรมให้เป็นไปตามเนื้อเรื่องที่ได้ออกแบบไว้.....35

สารบัญ (ต่อ)

หน้า

บทที่ 4 ผลการทดสอบโปรแกรมและวิเคราะห์ผล

4.1 จุดประสงค์ของการทดสอบโปรแกรม.....43

4.2 ขั้นตอนการทดสอบการทำงานของโปรแกรม.....43

4.3 ผลการทดสอบโปรแกรม.....43

4.4 วิเคราะห์ผล.....51

บทที่ 5 สรุปผลและข้อเสนอแนะ

5.1 สรุปผล.....52

5.2 ปัญหาในการทำงาน.....52

5.3 ข้อเสนอแนะ.....52

5.4 แนวทางในการพัฒนา.....53

เอกสารอ้างอิง.....54

ภาคผนวก ก.....55

ประวัติผู้เขียน.....61

สารบัญตาราง

ตารางที่ หน้า

1.1 ผังการปฏิบัติงาน.....2

2.1 ค่าบิตต่อการแสดงของสี.....12

2.2 ค่า dwFlags.....14

สารบัญรูป

รูปที่ หน้า

2.1 สถาปัตยกรรมของ DirectDraw 9

2.2 โครงสร้างของออบเจ็กต์ DirectSound.....21

2.3 สถาปัตยกรรมของ DirectPlay.....26

2.4 การเชื่อมต่อแบบ Peer to Peer.....26

2.5 การเชื่อมต่อแบบ Client – Server.....27

3.1 ตัวละครของผู้เล่น.....33

3.2 บางส่วนของตัวละครประกอบที่อยู่ภายในเกม.....33

3.3 ตัวอย่างของพุ่มไม้ และน้ำ.....33

3.4 แถบสถานะของผู้เล่นภายในเกม.....34

3.5 การวาดแผนที่.....35

3.6 โครงสร้างของเกม.....40

3.7 ขั้นตอนการทำงานในส่วนของคำถามภายในเกม.....42

4.1 หน้าแรกของโปรแกรม.....44

4.2 การเข้าสู่เกมในตอนเริ่มต้น.....44

4.3 สมุดบันทึก.....44

4.4 พูดคุยกับตัวละครอื่น.....45

4.5 เข้าไปยังสถานที่ท่องเที่ยวต่างๆ.....45

4.6 ตัวผู้เล่นกำลังเข้าสู่ตัวจังหวัด.....45

4.7 หน้าแรกของจังหวัด.....46

4.8 หน้าที่ 2 ของจังหวัด.....46

4.9 ตัวผู้เล่นพยายามเข้าบ้านแห่งความรู้.....46

4.10 ตัวผู้เล่นได้รับภารกิจจากตัวละคร.....47

4.11 สมุดบันทึกเมื่อได้รับภารกิจครบ 4 ภารกิจ.....47

4.12 ผู้เล่นพยายามทำภารกิจ.....47

4.13 คำถามข้อที่ 1 ถึง 4.....48

4.14 คำถามข้อที่ 5 ถึง 8.....48

สารบัญรูป (ต่อ)

รูปที่ หน้า

4.15 คำถามข้อที่ 9 ถึง 12.....48

4.16 คำถามข้อที่ 13 ถึง 16.....49

4.17 หน้าต่าง Make Connection ของฝ่าย Host.....50

4.18 หน้าต่าง Make Connection ของฝ่าย Guest.....51

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

DirectX เป็นเครื่องมือที่ช่วยพัฒนาโปรแกรมมัลติมีเดียซึ่งสามารถนำมาใช้พัฒนาเกมคอมพิวเตอร์ โดยมีหน้าที่ช่วยในการจัดการเกี่ยวกับการแสดงผลออกทางจอภาพ ทั้งกราฟิกแบบ 2 มิติ และ 3 มิติ ระบบเสียงประกอบขณะเล่นเกม การรับอินพุตต่างๆ จากผู้เล่น เช่น เมาส์ คีย์บอร์ด จอยสติ๊ก และการติดต่อผ่านเน็ตเวิร์ค เช่นการเล่นเกมแบบหลายผู้เล่น (Multiplayer Games)

ในปัจจุบันระบบปฏิบัติการวินโดว์สมีความสำคัญและเป็นที่นิยมเพราะมีความสะดวก ง่ายต่อการใช้งานอีกทั้งโปรแกรมต่างๆ ได้ถูกสร้างขึ้นมาเพื่อรองรับการใช้งานบนวินโดว์ส

เนื่องจากคณะผู้จัดทำได้เห็นความสามารถ DirectX และมีความสนใจด้านการเขียนโปรแกรมเกมจึงมีความคิดที่จะจัดทำซอร์ฟแวร์ประเภทเกมบนวินโดว์ส เพื่อศึกษาทั้งการเขียนโปรแกรมเกมและคอมพิวเตอร์กราฟิกโดยนำความสามารถของ DirectX มาใช้ในการพัฒนา

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อฝึกทักษะทางด้านการเขียนโปรแกรม

1.2.2 เพื่อให้มีความรู้เกี่ยวกับการพัฒนาเกมด้วย Microsoft Visual C++, DirectX SDK

1.2.3 เพื่อให้มีความรู้ความเข้าใจในเรื่องคอมพิวเตอร์กราฟิก

1.3 ขอบข่ายของโครงงาน

1.3.1 ศึกษาเครื่องมือและโปรแกรมที่ใช้พัฒนาโปรแกรมเกม อันได้แก่

- Microsoft DirectX7 SDK
- Microsoft Visual C++
- โปรแกรมกราฟิกต่างๆ เช่น photoshop 5.5 – 6.0 ,Painshop Pro

1.3.2 การออบแบบเกม (Designing)

- ออกแบบเนื้อเรื่องของเกม
- รูปแบบการดำเนินเรื่องของเกม
- วิธีการเล่นเกม
- เงื่อนไขการชนะ

1.3.3 การเตรียมทรัพยากรทางด้านกราฟิกและส่วนประกอบต่างๆ ของเกม

1.3.4 การเขียนโปรแกรมและทดสอบเล่นเกมเพื่อหาจุดบกพร่องเพื่อนำไปปรับปรุงแก้ไข

1.3.5 สรุปผลการทำงาน

1.3.6 จัดทำรูปเล่ม

1.4 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 ผังการปฏิบัติงาน

กิจกรรม	เดือน--ปี						
	มี.ค.44	เม.ย.44	พ.ค.44	มิ.ย.44	ก.ค.44	ส.ค.44	ก.ย.44
1.ออกแบบเกม	←	→					
2.เตรียมงาน		←	—	→			
3.เขียนโปรแกรม				←	—	—	→
4.สรุปผล						←	→
5.จัดทำรูปเล่ม						←	— →

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 โปรแกรมเกมที่ใช้ DirectX SDK เป็นเครื่องมือในการพัฒนา

1.5.2 มีทักษะในการเขียนโปรแกรมด้วย Microsoft visual C++ เพิ่มขึ้น

1.5.3 มีความรู้ในส่วนประกอบต่าง ๆ ของ DirectX

1.5.4 มีความรู้ความเข้าใจในเรื่องคอมพิวเตอร์กราฟิก

1.6 งบประมาณที่ต้องใช้ในการดำเนินงาน

ใช้เป็นค่าวัสดุอุปกรณ์ รวมทั้งสิ้นสองพันบาทถ้วน

* โดยขออนุมัติถัวเฉลี่ยทุกรายการ

บทที่ 2

การเขียนโปรแกรมเกมด้วยไดเรคเอ็กซ์

2.1 การเขียนโปรแกรมบนวินโดว์สด้วย Microsoft Visual C++[1,2]

2.1.1 ฟังก์ชัน WinMain()

```
int WINAPI WinMain (HINSTANCE hInstance , HINSTANCE hPrevInstancd,
                    PSTR szCmdLind , int iCmdShow)
```

ฟังก์ชัน WinMain() มีค่าพารามิเตอร์ดังนี้

- พารามิเตอร์ hInstance มีชนิดเป็น HINSTANCE ซึ่งข้อมูลชนิด HINSTANCE นี้ มีค่าเทียบได้กับ เบอร์ หรือ หมายเลข ของโปรแกรม เนื่องจากโปรแกรม (ไฟล์ .exe) 1 ตัว สามารถถูกรันขึ้นมาได้มากกว่า 1 ครั้ง ซึ่งวินโดวส์จะใช้ code ของโปรแกรมร่วมกัน แต่แยก data ออกจากกัน ดังนั้น จึงต้องมีเบอร์โปรแกรม เพื่อแยกความแตกต่างของโปรแกรมตัวเดียวกัน ที่รันขึ้นมาในแต่ละครั้ง

- พารามิเตอร์ hPrevInstance มีชนิดเป็น HINSTANCE เช่นกัน ซึ่งจะเป็นเบอร์ของโปรแกรมตัวเดียวกันที่ถูกรันขึ้นมาก่อนหน้าตัวนี้ ถ้าไม่โปรแกรมเดียวกันถูกรันขึ้นมาก่อน ค่านี้จะเป็น 0

- พารามิเตอร์ lpszCmdLine มีชนิดเป็น PSTR (pointer to string) เป็น พอยน์เตอร์ไปยังสตริง ที่ผู้ใช้พิมพ์ต่อจากชื่อโปรแกรมเรา (command line) เช่น “winzip *. txt” โปรแกรม winzip จะได้รับเป็น “test.zip *.txt”

- พารามิเตอร์ nCmdShow มีชนิดเป็น integer บ่งบอกถึงลักษณะเริ่มต้นของหน้าต่าง เช่น minimize (ย่อ), ปกติ, หรือ maximize (ขยายเต็มจอ)

2.1.2 WNDCLASS

class คือ โครงสร้างข้อมูลชนิดหนึ่งที่วินโดวส์ใช้ในการสร้างหน้าต่างขึ้นมา ซึ่งข้อมูลที่เก็บใน class นี้มีหลายอย่าง เช่น ไอคอนเมื่อเราย่อหน้าต่างลง สีพื้นหลังของหน้าต่าง ลักษณะของ mouse pointer เมื่อผู้ใช้ลากเมาส์ผ่านหน้าต่าง เป็นต้น การสร้าง class ทำได้โดยกำหนดลักษณะของ class ตามที่ต้องการลงใน structure WNDCLASS แล้วส่งไปให้ฟังก์ชัน RegisterClass ทำการสร้าง class ขึ้นมา เมื่อเราสร้าง class ขึ้นมาแล้ว ต่อไปเราก็สามารถใช้ class ที่สร้างขึ้นมานี้เป็น ต้นแบบ ในการสร้างหน้าต่างขึ้นมาได้ ซึ่งเขียนได้ดังนี้

```
WNDCLASS    wndclass;
wndclass.style          = CS_HREDRAW|CS_VREDRAW;
wndclass.lpfnWndProc    = WndFunc;
wndclass.cbClsExtra     = 0;
```

```
wndclass.cbWndExtra      = 0;
wndclass.hInstance       = hInstance;
wndclass.hIcon           = LoadIcon(NULL,IDI_APPLICATION);
wndclass.hCursor         = LoadCursor(NULL,IDC_ARROW);
wndclass.hbrBackground   = (HBRUSH)GetStockObjec(BLACK_BRUSH);
wndclass.lpszClassName   = szAppName;
wndclass.lpszMenuName    = NULL;
RegisterClass (&wndclass);
```

wndclass.style = CS_HREDRAW|CS_VREDRAW; จะเป็น คลาสที่จะกำหนดลักษณะพิเศษของคลาส เช่น CS_VREDRAW จะทำการ Redraws หน้าต่างทั้งหมดถ้ามีการปรับเปลี่ยนขนาดของพื้นที่ไคลเอ็นท์

wndclass.lpfnWndProc = WndFunc; เป็นชื่อของฟังก์ชันที่จะควบคุมหน้าต่างนี้ หรือเรียกว่าวินโดว์ฟังก์ชัน (window function) ทุกหน้าต่างที่ถูกสร้างจากคลาส นี้จะมีฟังก์ชัน WndFunc เป็นตัวควบคุมเหมือนกันหมด

wndclass.cbClsExtra กับ wndclass.cbWndExtra เป็นการกำหนดหน่วยความจำเพิ่มเติมให้กับคลาส และหน้าต่าง

wndclass.hIcon = LoadIcon(NULL, IDI_APPLICATION); เป็นการกำหนดไอคอนของหน้าต่างนี้ ซึ่งจะใช้ ไอคอน ที่วาดเองก็ได้ ฟังก์ชัน LoadIcon ของวินโดวส์ จะทำการเรียกไอคอนขึ้นมา

wndclass.hCursor = LoadCursor(0, IDC_ARROW); เป็นการกำหนดเคอร์เซอร์ของหน้าต่าง ซึ่งก็คล้ายกับกรณีเป็นไอคอน แต่จะใช้ ฟังก์ชัน LoadCursor แทน

wndclass.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH); จะเป็นการกำหนดสีของพื้นหลัง

wndclass.hInstance = hInstance; เบอร์ของโปรแกรม หรือแฮนเดิลของโปรแกรมที่เป็นเจ้าของคลาสนี้ ถ้าโปรแกรมถูกปิด ไม่ว่ากรณีใดๆ คลาสนี้จะหายไปด้วย

wndclass.lpszMenuName = NULL; ชื่อเมนู

wndclass.lpszClassName จะเป็นชื่อของคลาส

จากนั้นส่งไปให้ registerclass ซึ่ง registerclass จะทำการสร้างคลาสขึ้นมาตามลักษณะของ wndclass

2.1.3 การสร้างหน้าต่าง

หลังจากที่ได้กำหนดค่าพารามิเตอร์ต่างๆ ลงใน WNDCLASS และทำการลงทะเบียนคลาส จะสามารถสร้างหน้าต่างโดยอ้างอิงจากคลาสนี้ได้โดยฟังก์ชัน CreateWindow() ดังนี้

```
HWND  hWnd;
hWnd = CreateWindowEx(WS_EX_TOPMOST,  // Extended style
szAppName,
"DirectDraw ",
WS_POPUP,                          // รูปแบบของหน้าต่าง
0,                                 // ตำแหน่งเริ่มต้นของหน้าต่าง (แนวนอน)
0,                                 // ตำแหน่งเริ่มต้นของหน้าต่าง (แนวตั้ง)
GetSystemMetrics(SM_CXSCREEN),     // ความกว้างของหน้าต่าง
GetSystemMetrics(SM_CYSCREEN),     // ความสูงของหน้าต่าง
NULL,                              // Handle of parent
NULL,                              // Handle to menu
hInstance,                         // Application instance
NULL);                             // Additional data
ShowWindow(hWnd, nCmdShow);
```

เมื่อสร้างหน้าต่างด้วยฟังก์ชัน CreateWindow() แล้วจะได้ค่า hWnd เป็นแฮนเดิลหรือเบอร์ของหน้าต่างกลับมา แต่หน้าต่างที่สร้างขึ้นมานี้ยังไม่ปรากฏให้ผู้ใช้เห็นได้ทันที เพราะจะต้องใช้ฟังก์ชัน ShowWindow โดยพารามิเตอร์ตัวแรก คือ แฮนเดิลของหน้าต่าง พารามิเตอร์ที่สองคือ nCmdShow ซึ่งจะใช้ระบุลักษณะของหน้าต่างที่ต้องการแสดง หากใส่ค่า SW_MAXIMIZED เป็นการเปิดหน้าต่างแบบใหญ่เต็มจอ (Maximize) หากเราใส่ค่า SW_MINIMIZED จะเป็นการเปิดหน้าต่างแบบย่อ (Minimize)

2.1.4 การวนลูปรับเมสเสจ

การเขียนโปรแกรมด้วย DirectDraw จะต่างจากการเขียนโปรแกรมทั่วไป ในอดีตในยุคของดอสการเขียนโปรแกรมให้ทำตามคำสั่งไปเรื่อยๆ จนจบโปรแกรม ในยุคของวินโดว์สจะเขียนในลักษณะตอบสนองต่อเหตุการณ์ (event) เช่นเมื่อผู้ใช้คลิกเมาส์แล้วให้ไปทำอะไรบางอย่าง แต่การเขียนด้วย DirectDraw จะเขียนในลักษณะเดียวกับภาพยนต์โดยการวาดเป็นทีละเฟรม ซึ่งในระยะเวลาหนึ่งนาที จะต้องวาดภาพอย่างน้อย 20 เฟรม จึงจะมองเห็นเป็นภาพเคลื่อนไหว

โครงสร้างการเขียนโปรแกรมบนวินโดว์สตามปกติจะต้องคอยรับเมสเสจจากวินโดว์สแล้วตีความว่าให้ทำงานอะไร

```
while(GetMessage(&msg,NULL,0,0)
        {
```

```
        TranslateMessage(&msg);
        DispatchMessage(&msg);
    }
```

ฟังก์ชัน GetMessage จะรอรับเมสเสจ หากยังไม่ได้รับเมสเสจ โปรแกรมจะรอไปเรื่อยๆ (โดยไม่ทำอะไรต่อ) ฟังก์ชัน DispatchMessage จะส่งเมสเสจไปให้กับฟังก์ชัน WndFunc (วินโดว์ส ฟังก์ชันที่เขียนขึ้นมา ซึ่งจะกล่าวถึงในภายหลัง) เพื่อให้ทำงานที่ตอบสนองกับเมจเสจส่งมาแล้วเท่านั้นซึ่งใช้สำหรับโปรแกรม DirectDraw ที่มีการเขียนเฟรมภาพทุกระยะ ไม่ได้

โครงสร้างการวนลูปรับเมสเสจแบบใหม่ที่ใช้ใน DirectDraw ที่ต้องการเขียนเฟรมภาพทุกระยะ เป็นดังนี้

```
while(1)
{
    If(!GetMessage ( &msg,NULL,0,0 ))
        return msg.wParam;
    TranslateMessage (&msg );
    DispatchMessage (&msg );
}else{
    //เขียนภาพบนจบภาพ 1 เฟรม
    }
}
```

ฟังก์ชัน PeekMessage จะไปดูว่ามีเมสเสจส่งมาหรือยัง หากว่ายัง PeekMessage จะส่งค่าเป็นเท็จกลับคืนมา ทำให้รู้ว่าต้องตอบสนองเมสเสจหรือไม่ เมื่อว่างจากการตอบสนองเมสเสจ จึงค่อยเขียนภาพบนจอภาพ 1 เฟรม ทำให้การรีเฟรชภาพเป็นไปอย่างต่อเนื่อง

การเขียนส่วนตอบสนองเมสเสจให้ใช้เวลาน้อยที่สุดเท่าที่จะทำได้ ถ้าใช้เวลามากเกินไปจะทำให้การรีเฟรชภาพจะทำได้ไม่ต่อเนื่อง (อัตราการรีเฟรชขั้นต่ำ 20 เฟรมต่อนาที ดังนั้นระยะเวลาต่อเฟรม = 0.05 วินาทีต่อหนึ่งเฟรม) หากจำเป็นต้องทำงานบางอย่างที่ใช้เวลามากกว่านั้น ควรจะเขียนแบบ Multitasking คือจะทำงานหลายๆ งานขนานกันไป

2.1.5 ฟังก์ชันวินโดว์ส

ในการวนลูปรับเมสเสจจะมีฟังก์ชัน DiapatchMessage() ฟังก์ชันนี้จะส่งเมจเสจที่เกี่ยวข้องกับหน้าต่างไปให้กับฟังก์ชัน WndFund() ที่ได้กำหนดไว้ใน WNDCLASS ข้างต้น

```
Wndclass.lpfnWndProc = WndFunc;
```

```
LRESULT APIENTRY WndFunc(HWND hwnd,UINT message,WPARAM wParam,
LPARAM lParam)
{
    switch (message)
    {
        .....
    }
    return 0;
}
```

วินโดว์สฟังก์ชัน คือฟังก์ชันที่เขียนขึ้นมาเพื่อควบคุมการทำงานของหน้าต่างให้ทำงานตามที่ต้องการ ซึ่งโปรแกรมจะไม่เรียกใช้งานวินโดว์สฟังก์ชันนี้โดยตรง แต่จะกำหนดไว้ในคลาสว่าต้องการให้วินโดว์สฟังก์ชันอะไรเป็นตัวควบคุมหน้าต่างเรา เมื่อเกิดเหตุการณ์ (event หรือ message) ต่างๆ วินโดว์สจะเป็นผู้เรียกวินโดว์สฟังก์ชันนี้ด้วยพารามิเตอร์ 4 ตัว คือ

hWnd มีชนิดเป็น HWND ซึ่งเป็นค่าแฮนเดิลของโปรแกรม

message เป็นตัวแปรชนิด UINT (unsigned integer) เมสเสจที่ได้มาจากระบบปฏิบัติการวินโดว์ส ไม่ว่าจะมีเหตุการณ์ที่เกี่ยวข้องกับโปรแกรมหรือหน้าต่างนั้นๆ เช่น มีการกดเมาส์ กดคีย์บอร์ด หรือปิดโปรแกรม วินโดว์สจะส่งเมสเสจมาให้ตลอดเวลา เราจะต้องดูว่าเมสเสจที่ได้รับเป็นเมสเสจใดและตอบสนองต่อเมสเสจนั้นๆ อย่างเหมาะสม หากเมสเสจที่ได้รับไม่เกี่ยวข้องกับโปรแกรมาโดยตรงก็จะส่งไปให้กับ ฟังก์ชัน DefWindowProc จัดการแทน ซึ่งฟังก์ชันนี้จะจัดการตามค่าดีฟอลต์ของวินโดว์ส

wParam และ lParamWPARAM เป็นตัวแปรชนิด WPARAM (word parameter) และ LPARAM (long parameter) ตามลำดับ เป็นข้อมูลเพิ่มเติมเหตุการณ์ ขึ้นอยู่กับขณะนั้นเกิดเหตุการณ์อะไร เช่นเมื่อเกิดการคลิกเมาส์ wParam จะเก็บสถานะการกด shift คีย์บนคีย์บอร์ดขณะนั้น lParam จะเก็บค่าตำแหน่งของเมาส์ ถ้าเป็นเหตุการณ์อื่น wParam กับ lParam จะมีความหมายอื่นๆ ต่างกันไป

ในฟังก์ชันจะมีการตอบสนองต่อเหตุการณ์ อีกเหตุการณ์หนึ่งคือ WM_DESTROY เกิดขึ้น

```
case WM_DESTROY:
    // คืนหน่วยความจำ
    PostQuitMessage( 0 );
break;
```

เมื่อตัวโปรแกรมถูกปิดลงไปแล้ว WM_DESTROY จะทำการคืนหน่วยความจำ จากนั้นเรียกฟังก์ชัน PostQuitMessage พร้อมกำหนดค่า exit code ซึ่งในที่นี้ใช้เลข 0 ซึ่งฟังก์ชันนี้จะบังคับให้การ

เรียกใช้ฟังก์ชัน GetMessage ในครั้งต่อไป return ค่า FALSE ซึ่งจะทำให้เงื่อนไขใน message loop ไม่เป็นจริง และโปรแกรมจะหลุดจาก message loop ได้

หลังจากเลือกเหตุการณ์ที่สนใจแล้ว เหตุการณ์อื่นๆ ที่เราไม่สนใจจะเขียนโปรแกรม ต้องผ่านไปให้ฟังก์ชัน DefWindowProc ทำงาน ซึ่งฟังก์ชันนี้จะทำงานมาตรฐานของหน้าต่างทุกหน้าต่าง

2.2 Direct X

Microsoft DirectX SDK (Software Development Kit) เป็นเครื่องมือที่ใช้ในการพัฒนาซอฟท์แวร์ทางมัลติมีเดียของทางบริษัท ไมโครซอฟท์ รวมถึงสามารถพัฒนาเกมคอมพิวเตอร์ได้อีกด้วย

ส่วนประกอบของ DirectX

DirectX เวอร์ชั่น 7 มีส่วนประกอบสำคัญดังนี้

1. DirectDraw เป็นส่วนประกอบหลักอย่างแรกของ DirectX SDK เนื่องจากมีหน้าที่จัดการกับการแสดงผลทางจอภาพ และให้ความเร็วในการประมวลผลสูงกว่าเกมบนดอส และการใช้ GDI (Graphic Device Interface) ของวินโดว์ส เนื่องจากการส่งข้อมูลที่ทำได้โดยตรงนั่นเอง อีกทั้งยังประกอบไปด้วยฟังก์ชันต่างๆ ที่คอยสนับสนุน 2D เกมโปรแกรม

2. Direct3D มีประสิทธภาพสูงมาก ในการใช้ในการสร้างเกม 3D บนวินโดว์สเนื่องจากการได้เปรียบจากการสามารถทำงานร่วมกับการ์ดเร่งความเร็ว 3D ได้อย่างมีประสิทธิภาพเพราะมีฟังก์ชันคอยสนับสนุนอยู่ มีส่วนประกอบอยู่ด้วยกัน 2 ส่วน คือ Direct 3D retained Mode และ Direct 3D Immediate Mode API

3. DirectSound มีฟังก์ชันคอยช่วยให้ผู้เขียนโปรแกรมเกมทำงานได้ง่ายขึ้น ในการใส่เสียงเพลงประกอบและเสียงเอฟเฟ็กต์ต่างๆ เข้าไปในเกม

4. DirectPlay ทำให้การสร้างเกมระบบผู้เล่นหลายๆ คนทำได้ง่ายขึ้น และสะดวกในการใช้งานร่วมกับโมเด็ม และระบบเน็ตเวิร์กต่างๆ

5. DirectInput ทำให้การสร้างเกมเพื่อรองรับระบบการสั่งงานจากผู้เล่นไม่ว่าจะเป็น จอยสติก เมาส์ หรือ คีย์บอร์ดทำได้ง่ายและมีประสิทธิภาพเป็นอย่างยิ่ง อีกทั้งยังมีฟังก์ชั่นที่สนับสนุนการเพิ่มเติมอุปกรณ์การเล่นที่จะมีขึ้นมาใหม่อีกในอนาคต

6. DirectMusic มีลักษณะคล้ายๆ กับ DirectSound นั่นเองแต่จะมีฟังก์ชันที่คอยสนับสนุนการใช้งานไฟล์ MIDI ต่างๆ

7. DirectSetup ช่วยในการติดตั้งส่วนประกอบที่จำเป็นต้องใช้ของ DirectX ให้กับผู้ใช้อย่างอัตโนมัติ ทำให้สะดวก

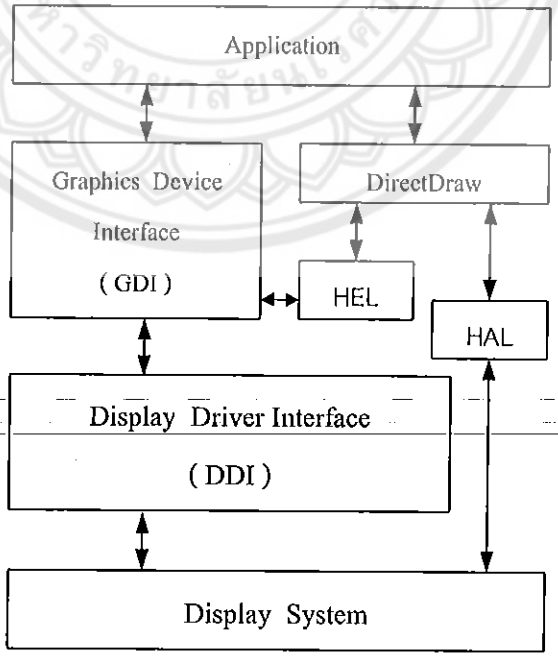
8. AutoPlay เป็นส่วนประกอบที่สำคัญของระบบวินโดว์ส ทำให้โปรแกรมเกมสามารถที่จะทำการติดตั้งลงไปในฮาร์ดดิสก์ได้อย่างอัตโนมัติ โดยโปรแกรมติดตั้งจะเริ่มทำงานเมื่อเรานำแผ่นโปรแกรมเกมใส่เข้าไปในไดร์ฟซีดีรอม

2.3 DirectDraw

DirectDraw มีหน้าที่ในการช่วยให้นักพัฒนาเกมคอมพิวเตอร์ได้ใช้งานทางด้านกราฟิกและการจัดการกับฮาร์ดแวร์ต่างๆ ได้อย่างมีประสิทธิภาพดังนี้

- ช่วยจัดการกับระบบหน่วยความจำโดยตรงไม่ต้องผ่านระบบปฏิบัติการของวินโดว์ส ดังรูปที่ 2.1
- จัดการกับระบบการแสดงผลให้อย่างอัตโนมัติโดยเราไม่จำเป็นต้องไปเขียนไดรเวอร์ให้กับฮาร์ดแวร์ต่างๆ อีกต่อไป
- สามารถใช้งานร่วมกับ GDI ของวินโดว์สได้อย่างมีประสิทธิภาพ
- มีฟังก์ชันและเมธอด (Method) ต่างๆ ที่ช่วยในการสร้างเกมคอมพิวเตอร์สองมิติ

จากประโยชน์ต่างๆ นี้ทำให้ระบบการประมวลผลต่างๆ บนวินโดว์สเร็วขึ้นกว่าระบบดอส



รูปที่ 2.1 สถาปัตยกรรมของ DirectDraw

2.3.1 โครงสร้างโดยทั่วไปของ DirectDraw

- DirectDraw object มีหน้าที่หลักในการจัดการกับระบบการแสดงผล และใช้ในการติดต่อกับส่วนอื่นๆ ในลำดับต่อไป

- DirectDrawSurface object ใช้ในการจัดการกับรูปภาพที่จะทำการแสดงที่หน้าจอ โดยจะมีฟังก์ชันและเมธอดต่างๆ คอยสนับสนุน

- DirectDrawPalette object จะช่วยในการจัดการกับพาเล็ต (Pallete) เพื่อที่จะใช้ในการแสดงโหมดสีของรูปภาพที่จะทำการแสดงที่จอภาพ

- DirectDrawClipper object ช่วยในการป้องกันไม่ให้รูปที่จะแสดงนั้น ออกจากขอบเขตที่กำหนดไว้ให้

2.3.2 DirectDraw Fundamentals

ในการเรียก DirectDraw มาใช้งาน จะมีขั้นตอนดังนี้

1. สร้าง DirectDraw Object
2. ตั้งค่าการทำงานร่วมกัน (set Cooperative Level)
3. ตั้งค่าการแสดงผล (set Display Mode)
4. สร้าง DirectDraw Surface

2.3.3 การสร้าง DirectDraw Object

มีรูปแบบของฟังก์ชันดังนี้

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

พารามิเตอร์ GUID FAR *lpGUID ฟังก์ชัน DirectDrawCreate เป็นฟังก์ชัน ที่ใช้สร้าง DirectDraw object พารามิเตอร์ตัวแรกคือ ID ของ VGA card ในกรณีนี้ เราใช้ NULL หมายถึง VGA card ตัวที่ใช้อยู่ ค่านี้สามารถเป็น ค่าคงที่

DDCREATE_EMULATIONONLY ใช้สำหรับ emulate ทุกๆ features ของฮาร์ดแวร์ เป็นส่วนที่จัดการกับฟังก์ชันต่างๆ ที่ไม่ได้รับการสนับสนุนจากฮาร์ดแวร์

DDCREATE_HARDWAREONLY ถ้าสร้าง DirectDraw Object โดยใช้ ค่าFlag นี้ นั่นหมายความว่า โปรแกรมที่สร้างขึ้นต้องได้การสนับสนุนจากฮาร์ดแวร์เท่านั้น

พารามิเตอร์ตัวที่ 2 LPDIRECTDRAW FAR * lplpDD เป็น address ของ ตัวแปร pointer ที่จะรับค่า pointer to DirectDraw object ที่ return จากฟังก์ชัน

พารามิเตอร์ตัวที่ 3 Iunknown FAR *pUnkOuter จะถูกกำหนดให้มีค่าเป็น NULL

หลังจากเรียกฟังก์ชัน นี้แล้ว ถ้าทำงานสำเร็จ ค่า return จะเป็น DD_OK

2.3.4 Cooperative levels

มีรูปแบบของฟังก์ชันดังนี้

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

ค่าพารามิเตอร์ HWND hWnd เป็นค่าแฮนเดิลของโปรแกรม

ค่าพารามิเตอร์ DWORD dwFlags เมธอดนี้อยู่ในส่วนของ IDrectDraw7 InterFace ใช้ในการกำหนดว่ารูปแบบของการแสดงภาพที่หน้าจอ จะเป็นแบบ Fullscreen หรือ Windowed Mode โดย dwFlags จะเป็นตัวกำหนดว่าจะให้เป็นรูปแบบใด

- DDSCL_ALLOWMODEX สามารถใช้งานในรูปแบบของโหมด X ได้ (320x200 , 320x240 หรือ 320x400) มักจะใช้ร่วมกับ DDSCL_EXCLUSIVE และ DDSCL_FULLSCREEN
- DDSCL_ALLOWREBOOT สามารถทำการรีบูตได้ในขณะที่โปรแกรมกำลังทำงานอยู่ โดยการกดปุ่ม Ctrl+Alt+Delete
- DDSCL_CREATEDEVICEWINDOW ใช้ในการกำหนด Device เอง โดยจะสามารถใช้งานได้เฉพาะ Windows 98 และ Windows NT5 เท่านั้น
- DDSCL_EXCLUSIVE ใช้ในการกำหนดให้เป็นโหมด Exclusive โดยจะใช้กำหนดก็ต่อเมื่อ ต้องการให้แสดงผลแบบ Fullscreen ถ้าหากว่าไม่กำหนดก็จะไม่สามารถแสดงผลแบบ Fullscreen ได้
- DDSCL_FULLSCREEN ใช้ในการกำหนดให้แสดงผลแบบเต็มจอ
- DDSCL_NORMAL ใช้ในการกำหนดให้แสดงผลในรูปแบบของโหมดวินโดว์ส ไม่สามารถใช้ร่วมกับ DDSCL_ALLOWMODEX DDSCL_EXCLUSIVE หรือ DDSCL_FULLSCREEN
- DDSCL_NOWINDOWCHANGES ใช้ในการป้องกันไม่ให้ DirectDraw ไปทำการย่อหน้าต่าง (minimize) หรือ restore วินโดว์สนั่นเอง

2.3.5 SetDisplayMode

หลังจากทำการกำหนดการแสดงผลแบบ Fullscreen และ Windowed Mode

แล้วต่อไปก็ต้องกำหนดด้วยว่าจะต้องมีขนาดขอบเขตเท่าใด และความละเอียดของสีที่แสดงออกมีจำนวนเท่าไหร่ ซึ่งเราจำเป็นที่จะต้องอาศัยเมธอดนี้ในการกำหนด

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

- dwWidth กำหนดความกว้างของการแสดงผลในหน่วยพิกเซล pixel
- dwHeight กำหนดความสูงของการแสดงผลในหน่วยพิกเซล pixel
- dwBPP กำหนดความละเอียดของสีในการแสดงผล ในหน่วยบิตต่อ pixel
- dwRefreshRate กำหนดให้เป็น 0
- dwFlags กำหนดให้เป็น 0

ตารางที่ 2.1 การกำหนดค่าบิตต่อการแสดงจำนวนสี

BPP	Colors
1	Black&White
4	16
8	256
16	High Color
24	True Color

2.3.6 Surface

Surface เป็นส่วนประกอบที่สำคัญของ DirectDraw เนื่องจากมีหน้าที่มีความสำคัญมากในการจัดการกับรูปภาพและการประมวลผลต่างๆ โดยจะต้องทำการสร้าง COM ของ Surface ขึ้นมาเพื่อจะสามารถใช้งานเมธอดต่างๆ ในส่วนของ IdrectDrawSurface7 ได้สร้างจาก Object ของ DirectDraw อันได้แก่ ddraw นั่นเอง

ในการสร้าง Surface นั้นเราสามารถสร้าง Surface ได้หลายแบบด้วยกัน แต่ที่นิยมใช้ ก็จะสร้างไว้ 2 Surface ดังนี้ คือ

1. Primary Surface ที่แสดงให้ผู้ใช้ได้มองเห็นอยู่ตลอดเวลา ดังนั้น Primary Surface จึงถูก set ให้มีค่าการแสดงผลตามตามความกว้าง และความยาวตามที่ได้กำหนดไว้

2. Back Buffer เป็น Surface ที่เตรียมไว้สำหรับแสดงต่อไปไม่สามารถมองเห็นได้ เมื่อต้องการนำภาพมาแสดงสามารถทำได้โดยจะเลื่อนขึ้นไปอีก 1 ขั้น คือ ถ้าเป็น Surface สำรองอันดับ 1 ก็จะเลื่อนขึ้นไปเป็น Primiry Surface ในลำดับต่อไป ส่วน Surface สำรองอันดับ 2 ก็จะเลื่อนไปเป็น Surface สำรองอันดับ 1 และ Primiry Surface ก็จะเลื่อนกลับมาเป็น Surface สำรองอันดับที่ 2 ในกรณีที่เราสร้าง Surface ไว้ทั้งหมด 3 Surface ด้วยกัน และเนื่องจาก Back Buffer อยู่ติดกับ Primary Surface ดังนั้นจึงมีขนาดเดียวกับ Primary Surface

3. OffScreen Buffer จะคล้ายกับ Back Buffer มาก จะแตกต่างที่ OffScreen ไม่ต้องขึ้นกับ Primary Surface ส่วนมาก OffScreen Buffer จะใช้ทำการเก็บภาพบิตแมป

ก่อนที่จะสร้าง Surface ขึ้นมาได้นั้น จะต้องกำหนดรูปแบบของ Surface ที่จะทำการสร้างขึ้นมาเสียก่อน ว่าจะให้สามารถทำอะไรได้บ้าง รองรับรูปแบบไหนได้บ้าง หรือจะมีลักษณะอย่างไร ซึ่งจะต้องกำหนดรูปแบบเหล่านี้ให้กับตัวแปรที่ใช้แทน Structure ของ DDSURFACEDESC (DirectDraw Surface Description) โดยเราจะต้องทำการประกาศตัวแปรดังนี้

```
DDSURFACEDESC2   ddsd;
```

โครงสร้างของ DDSURFACEDESC มีดังนี้

```
typedef struct _DDSURFACEDESC2{
        DWORD    dwSize;
        DWORD    dwFlags;
        DWORD    dwHeight;
        DWORD    dwWidth;
        union
        {
                LONG   lPitch;
                DWORD  dwLinearSize;
        }DUMMYUNIONNAMEN(1);
        DWORD  dwBackBufferCount;
        union
        {
                DWORD   dwMipMapCount;
                DWORD   dwRefreshRate;
        } DUMMYUNIONNAMEN(2);
                DWORD    dwAlphaBitDepth;
                DWORD    dwPeserved;
```

```
LPVOID   lpSurface;
DDCOLORKEY   kkckCK DestOverlay;
DDCOLORKEY   ddckCKDDestBlt;
DDCOLORKEY   ddckCKSrcOverlay;
DDCOLORKEY   ddckCKSrcBlt;
DDPIXELFORMAT   ddpfPixelFormat;
DDSCAPS2        ddsCaps;
DWORD           dwTextureStage;
}DDSURFACEDESC2,FAR  *LPDDSURFACEDESC2;
```

โครงสร้างส่วนหลักที่นิยมใช้ของ DDSURFACEDESC

DWORD dwSize กำหนดจำนวนไบต์ที่ใช้บรรจุตัวแปร

DWORD dwFlags กำหนดรูปแบบมีได้มากกว่า 1 Flags

DWORD dwHeight กำหนดค่าความสูงของ ของ Surface ในหน่วยพิกเซล (pixel)

DWORD dwWidth กำหนดค่าความกว้างของ Surface ในหน่วยพิกเซล (pixel)

DWORD dwBackBufferCount กำหนดจำนวนของ Back Buffer

ตารางที่ 2.2 การกำหนดค่า dwFlags

ค่า Flags	member
DDSD_ ALPHABITDEPTH	DwAlphaBitDepth
DDSD_BACKBUFFERCOUNT	DwBackBufferCount
DDSD_CAPS	ddsCaps
DDSD_CKDESTBLT	ddckDestBlt
DDSD_CKDESTOVERLAY	ddckDestOverlay
DDSD_CKSRCBLT	ddckSrcBlt
DDSD_CKSRCOVERLAY	ddckSrcOverlay
DDSD_HEIGHT	dwHeight
DDSD_LINEARSIZE	dwLinearSize
DDSD_LPSURFACE	lpSurface
DDSD_MIPMAPCOUNT	dwMipMapCount
DDSD_PITCH	lPitch

ตารางที่ 2.2 (ต่อ)

DDSD_PIXELFORMAT	ddpfPixelFormat
DDSD_REFRESHRATE	dwRefrechRate
DDSD_TEXTURESTAGE	dwTextureStage
DDSD_WIDTH	dwWidth

ภายหลังจากที่เราทำการกำหนดรูปแบบต่างๆ ให้กับ surface เป็นอันเรียบร้อยแล้ว เราถึงจะทำการใช้ฟังก์ชันนี้ในการสร้าง surface ขึ้นมา โดยเราจะใช้ก่อนไม่ได้ เนื่องจากฟังก์ชันนี้จำเป็นที่จะต้องรู้รูปแบบของตัวแปรที่กำหนดเสียก่อน

```
HRESULT CreateSurface (
        LPDDSURFACEDESC2 lpDDSurfaceDesc,
        LPDIRECTDRAWSURFACE7 FAR* lplpDDSurface ,
        Iunknown FAR* pUnkOuter
);
```

ยกตัวอย่างการสร้าง Primary Surface 1 Back Buffer และ 1 offscreen buffer ที่มีขนาดความกว้าง 400 ความสูง 300 โดยสมมุติว่า มี IdirectDraw7 interface pointer แล้ว

```
DDSURFACEDESC2      ddsd;      // surface description structure
LPDIRECTDRAWSURFACE7   lpDDSPrimary   = NULL; // Primary Surface
//set up Primary Surface
ZeroMemory(&ddsd,sizeof(ddsd));
ddsd.dwSize = sizeof(ddsd);
ddsd.dwFlags = DDSD_CAPS|DDSD_BACKBUFFERCOUNT;  // valid flags
ddsd.ddsCaps.dwCaps = DDSCAPS_PRIMARYSURFACE |     // Primary Surface
                      DDSCAPS_COMPLEX |            // back buffer is chained
                      DDSCAPS_FLIP |               // always page flipping
                      DDSCAPS_VIDEOMEMORY;         // create in video memory
ddsd.dwBackBufferCount = 1;
// Create the primary surface.
if ( FAILED( lpDD->CreateSurface( &ddsd, &lpDDSPrimary, NULL ) ) )
{
        // error handling code
```

```
        }
    //get the attached surface
    LPDIRECTDRAWSURFACE7   lpDDsBack = NULL;
        ddsd.ddsCaps.dwCaps = DDSCAPS_BACKBUFFER;
        if (FAILED(lpDDSPrimary->GetAttachedSurface(&ddsd.ddsCaps, &lpDDSBack)))
        {
            // error handling code
        }
    LPDIRECTDRAWSURFACE7   lpDDsOffscreen  = NULL;
    // Set up offscreen surface
        ddsd.dwFlags = DDSD_CAPS | DDSD_HEIGHT |        // valid flags
                       DDSD_WIDTH| DDSD_CKSRCBLT;
      ddsd.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN;     // offsrcen buffer
      ddsd.dwHeight  =  300;          // set   height
      ddsd.dwWidth  =  400;           // set   width
    if (FAILED(lpDD->CreateSurface( &ddsd, &lpDDSOffsreen, NULL )))
    {
        // error handling code
    }
```

2.3.7 ตารางสี Pallete

ธรรมชาติของสีที่เราเห็นกันทุกวันนั้นสามารถแยกออกมาได้เป็นแม่สี 3 สี คือ สีแดง สีเขียว สีน้ำเงิน เมื่อนำสีทั้งสามมาผสมกันจะได้สีต่างๆ มากมาย คอมพิวเตอร์ใช้ระบบเดียวกันนี้ในการบอกสี โดยเก็บในรูปรหัส RGB (Red , Green , Blue) และเก็บในรูปข้อมูลขนาด 4 ไบต์ โดยไบต์แรกไม่ใช้ , ไบต์ที่สองแสดงระดับความเข้มของสีแดง (0 แทนไม่สีแดงเลย , 255 แทนสีแดงเต็มที่) , ไบต์ที่สามเป็นสีเขียว ,ไบต์ที่สี่เป็นสีน้ำเงิน เรียกว่าระบบสีแท้จริง (True Color) แต่ว่าระบบสีนี้ต้องใช้หน่วยความจำมากเพราะใช้หน่วยความจำ 4 ไบต์ ต่อหนึ่งพิกเซล จึงมีระบบอีกระบบ เรียกว่า ตารางสี (Pallete) เพื่อให้ใช้หน่วยความจำประหยัดเพียง 1 ไบต์ ต่อหนึ่งพิกเซล ทำให้มีสีได้เพียง 256 สี ต่อหนึ่งจุดสี ดังนั้นจึงต้องมีตารางสีขนาด 256 ช่องเพื่อแทนสีที่ใช้ โดยแต่ละช่องเก็บค่า RGB ไว้ ระบบนี้ใช้ได้ดีในโปรแกรมแบบดอส เพราะในขณะหนึ่งๆ มีโปรแกรมทำงานเพียงโปรแกรมเดียว แต่เมื่อใช้วินโดว์สโปรแกรมหลายตัวทำงานพร้อมกันได้ ทำให้โปรแกรมที่ทำงานพร้อมกันนั้นต้องแบ่งจำนวนสีกันใช้ โดยมีกติกาว่าโปรแกรมที่ผู้ใช้ทำงานอยู่ล่าสุด (Forehround window) จะเป็นผู้ที่ใช้ตารางสีนั้นทั้งหมด

แต่โปรแกรมอื่นๆ ต้องใช้สีที่ใกล้เคียงที่สุดแทน เช่น หากโปรแกรมนั้นต้องการใช้สีเทาอมน้ำเงิน เช่น RGB (128 , 128 , 200) แต่ไม่มีสีนั้น ต้องหาสีที่ใกล้เคียงที่สุดแทน เช่น RGB (128 , 128 , 250) และหากว่าโปรแกรมด้านล่างเปลี่ยนมาอยู่ด้านบนต้องมีการเปลี่ยนไปใช้ตารางสีของโปรแกรมด้านบนแทน และโปรแกรมอื่นๆ ต้องจัดตารางสีให้เข้ากับตารางสีใหม่ เรียกว่า การเรียลไลซ์ตารางสี (Realize Pallte)

ในโปรแกรมเกมที่เล่นในโหมด 256 สี มีตัวละครหลายตัวที่อยู่ในฉากเดียวกัน แน่นอนว่าสีที่ใช้ในตัวละครทุกตัวรวมกันจึงไม่ถึง 256 สี มีวิธีที่ช่วยให้การสร้างสีตัวละครเป็นไปได้สะดวกขึ้น โดยขณะสร้างตัวละครในไฟล์กราฟิก ให้สร้างรวมกันในไฟล์เดียวกันและใช้ตารางสีแบบ 8 บิต แล้วนำไฟล์ทั้งไฟล์นั้นไปปะบนพื้นที่เขียนภาพสำรอง (Off-screen Surface) ไว้ก่อน เมื่อใดที่ต้องการวาดภาพตัวละคร ก็ตัดภาพจากพื้นที่เขียนภาพสำรองไปปะบนพื้นที่เขียนภาพส่วนหลังอีกที

2.3.8 Clippers

DirectDraw สามารถใช้งานได้ 2 หมวด คือ Full Screen และ Windowed Mode ในส่วนของ Full Screen นั้น Font Buffer และ Back Buffer จะมีขนาดเท่ากันดังนั้นจึงไม่ต้องระวังว่า Surface ที่จะเตรียมแสดงในลำดับต่อไป จะมีการเกินขนาดของขอบเขตจอ ทำให้เราไม่มีปัญหาในการที่จะ Flip Surface นั่นเอง ในขณะที่ Windowed Mode นั้น Font Buffer เกิดจากการรวมกันของ Primary Surface ของ DirectDraw และ Surfaceของ GDI โดยแอปพลิเคชั่น ที่เราสร้างขึ้นนั้นมีขนาดไม่เต็มจอ ดังนั้นจึงเกิดการซ้อนทับกับแอพลิเคชั่นอื่นๆ อยู่ ทำให้เกิด Clip lists ขึ้นมานั่นก็หมายถึงเราจะต้องมี Clippers อยู่หลายๆ กรอบนั่นเอง แต่ในกรณีที่มีแอพลิเคชั่นอยู่ แอพลิเคชั่นเดียว DirectDraw ก็จะทำการส่งข้อมูลมาเฉพาะในส่วนของแอพลิเคชั่นเท่านั้น จะไม่มีการเลยขอบนอกขอบเขตเป็นอันขาด

2.3.9 การใช้ Gamma Control ของ DirectDraw ในการทำ Fade

Fade จะมีทั้ง Fade In และ Fade out โดยที่ Fade out จะเป็นการทำให้จอภาพการแสดงผลค่อยๆ มืดลง และ Fade In จะเป็นการทำให้จอภาพการแสดงผลค่อยๆ สว่างกลับมา

DirectDraw จะมีโครงสร้าง หรือ Structure หนึ่ง ชื่อว่า DDGAMMARAMP ซึ่งประกอบไปด้วยอะเรย์ (Array) ขนาด 256 ช่อง ของWORD จำนวน 3 ชุด โดย Array ทั้ง 3 ชุด นี้มีชื่อว่า Red,Green และ Blue โครงสร้างดังนี้

```
typedef struct DDGAMMARAMP {
        WORD red[256];
        WORD green[256];
        WORD blue[256];
```

```
} DDGAMMARAMP, FAR * LPDDGAMMARAMP;
```

Structure ตัวนี้จะเก็บค่า Gamma ของการ์ดแสดงผล

```
// กำหนดตัวแปร Gamma Control เพื่อเตรียมไว้ใช้งาน
LPDIRECTDRAWGAMMACONTROL lpDDGammaControl = NULL;

// Structure นี้จะเป็นค่า RAMP สำหรับใช้ในการเปลี่ยนแปลงค่า Gamma
DDGAMMARAMP DDGammaRamp;

// Structure นี้จะเก็บค่า Gamma เดิม เพื่อจะได้ใส่ค่ากลับได้หลังจากที่ Fade
DDGAMMARAMP DDGammaOld;
```

การตรวจสอบว่าการ์ดแสดงผลจะสนับสนุน Gamma Control สามารถทำได้โดยการทำ Query Interface กับ Primary Surface

```
hResult = lpDDSPrimary->QueryInterface(IID_IDirectDrawGammaControl,
( void **)&lpDDGammaControl);
```

ขั้นตอนต่อไปนำค่า Gamma ที่ตั้งไว้ในปัจจุบันของการ์ดจอ แล้วจัดเก็บเอาไว้ในตัวแปร DDGammaOld ที่เตรียมไว้แล้ว

```
hResult = lpDDGammaControl->GetGammaRamp(0,&DDGammaOld);
```

จากนั้นทำการเรียกฟังก์ชัน GetGammaRamp มาอีกครั้งเพื่อเก็บค่าไว้ในตัวแปร DDGammaRamp เพื่อจะได้เอาไว้เปลี่ยนแปลงค่าในการทำ Fade

```
hResult = lpDDGammaControl->GetGammaRamp(0,&DDGammaRamp);
```

1. การทำ Fade Out

ในขณะนี้เราจะมีค่า Gamma ของการ์ดแสดงผลเก็บเอาไว้ และที่โครงสร้างของ DDGAMMARAMP จะพบว่าในแต่ละช่องของ Red, Green, Blue จะมีค่าได้ตั้งแต่ 0-65535 และเพื่อให้เกิดการ Fade Out หรือการทำให้ภาพค่อยๆ มืด ดังนั้นจึงให้มีการวนลูปลดค่าของ DDGAMMARAMP ในที่นี้กำหนดให้มีการลดค่าลงครั้งละ 5 ให้กับค่าของ Red Green Blue ทุกตัวใน

DDGAMMARAMP แล้วปรับค่า Gamma Control หลังจากการตั้งค่าแต่ละรอบ

```
// วนลูปลดค่าของ DDGAMMARAMP ทีละ 5
for (WORD gamma = 65535; gamma 0; gamma -= 5)
{
    // วนลูปตั้งค่าของ Red, Green, Blue ทั้ง 256 ช่อง
    for (int index = 0; index < 256; index++)
    {
        // ตั้งค่าของ Red
        if(gamma <  DDGammaRamp.red[index])
        {
            DDGammaRamp.red[index]  = gamma;
        }
        //  ตั้งค่าของ Green
        if(gamma <  DDGammaRamp.green[index])
        {
            DDGammaRamp.green[index]  = gamma;
        }
        //  ตั้งค่าของ Blue
        if(gamma <  DDGammaRamp.blue[index])
        {
            DDGammaRamp.blue[index]  = gamma;
        }
    }
    // ปรับค่า Gamma Control  ให้เปลี่ยนค่า Gamma
    hResult = lpDDGammaControl -> SetGammaRamp(0,&DDGammaRamp);

    // ตรวจสอบการปรับค่า Gamma Control
    if(hResult != DD_OK)
    {
        // พบปัญหาให้ออกจากการทำงานของฟังก์ชัน Fade
        return;
```

```
        }
    }
```

2. การทำ Fade In

จะเป็นการทำให้ภาพที่มืด–กลับมาสว่างอีกครั้ง–ขั้นตอนการทำจะคล้ายกับการทำ Fade out

```
// วนลูปเพิ่มค่าของ DDGAMMARAMP ทีละ 5
for (WORD gamma = 65535; gamma 0; gamma += 5)
{
    // วนลูปตั้งค่าของ Red, Green, Blue ทั้ง 256 ช่อง
    for (int index = 0; index < 256; index++)
    {
        // ตั้งค่าของ Red
        if(DDGammaRamp.red[index]< DDGammaOld.red[index])
        {
            DDGammaRamp.red[index]  = gamma;
        }
        //  ตั้งค่าของ Green
        if(DDGammaRamp.green[index]< DDGammaOld.green[index])
        {
            DDGammaRamp.green[index]  = gamma;
        }
        //  ตั้งค่าของ Blue
        if(DDGammaRamp.blue[index]< DDGammaOld.blue[index])
        {
            DDGammaRamp.blue[index]  = gamma;
        }
    }
    // ปรับค่า Gamma Control  ให้เปลี่ยนค่า Gamma
    hResult = lpDDGammaControl -> SetGammaRamp(0,&DDGammaRamp);
    // ตรวจสอบการปรับค่า Gamma Control
    if(hResult != DD_OK)
    {
```

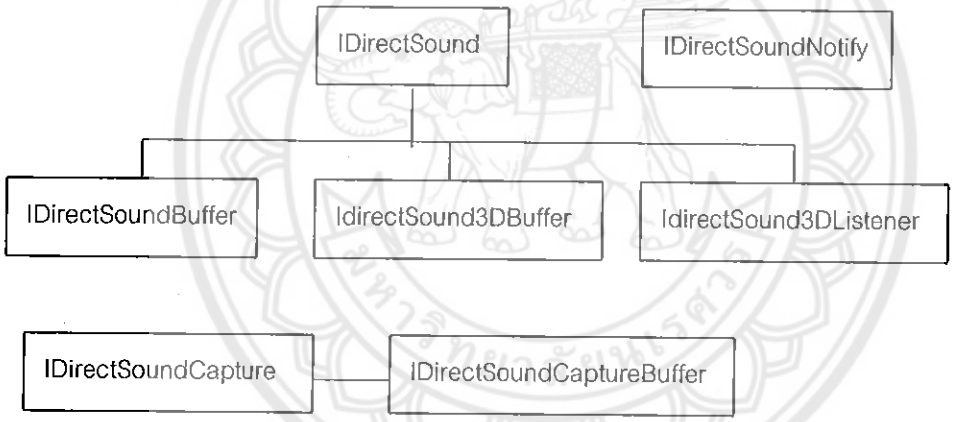
```
            // พบปัญหาให้ออกจากการทำงานของฟังก์ชัน Fade
            return;
        }
    }
```

หลังจากที่เราได้ทำการ Fade แล้วก่อนที่จะ Release Gamma control จะต้องทำการคืนค่าเดิมของ Gamma กลับมาให้เรียบร้อย โดยการตั้งค่าของ Gamma Control ให้เป็นค่าใน DDGammaOld ที่ได้เก็บไว้ตั้งแต่แรก

```
hResult = lpDDGammaControl -> SetGammaRamp(0,&DDGammaOld);
```

2.4 DirectSound

โครงสร้างของออบเจ็กต์ DirectSound



รูปที่ 2.2 โครงสร้างของออบเจ็กต์ DirectSound

IDirectSound เป็นตัวกลางที่ทำหน้าที่เหมือนหัวใจของโปรแกรม หน้าที่ของคลาสที่เกี่ยวข้องกับโปรแกรม คือ

1. การเปิดและปิดออบเจ็กต์ IdirectSound
2. การสร้างออบเจ็กต์ IDirectSoundBuffer จากออบเจ็กต์ IdirectSound

IdrectSound3DBuffer เป็นคลาสที่ใช้สำหรับเก็บข้อมูลเสียง เล่นเสียง นำเสียงหลายเสียงมารวมกันเป็นคลาสหลักที่ต้องใช้มากที่สุด

IdirectSound3DBuffer เป็นคลาสที่ใช้ร่วมกับคลาส IdirectSound3DListener สำหรับเล่นเสียง 3 มิติ โดยคลาส IdirectSound3DBuffer ใช้เก็บข้อมูลเสียงและเล่นเสียงเช่นเดียวกับ IdirectSoundBuffer แต่ได้เพิ่มส่วนที่เป็นตำแหน่งของเสียงเพื่อให้เกิดระยะซ้าย – ขวา ความลึก

IDirectSoundNotify เป็นคลาสที่ใช้ในขณะเล่นเมื่อเสียงจบหรือเมื่อถึงเวลาที่เราต้องใส่ข้อมูลเสียงเข้าไปในบัฟเฟอร์คลาสนี้ก็จะแจ้งเตือนให้เราทราบ

IdirectSoundCapture และ IdirectSoundCaptureBuffer ใช้สำหรับดักจับเสียงนำมาเก็บไว้ในไฟล์เพื่อเก็บไว้ใช้ในภายหลัง

2.4.1 การผสมเสียงด้วย IdirectSoundBuffer

IdirectSoundBuffer จะใช้ในการเก็บข้อมูลเสียงที่จะเปล่งออกมา โดย DirectSound มีบัฟเฟอร์สองแบบคือ PrimaryBuffer และ Secondary buffer

PrimaryBuffer เป็นออบเจ็กต์ที่ใช้เก็บเสียงหลักซึ่งเป็นเสียงที่ได้จากการผสมเสียงย่อยๆ เรียบร้อยแล้ว

SecondaryBuffer ใช้เก็บเสียงแต่ละเสียงที่ต้องการนำมาผสมโดย Secondary Buffer เก็บเสียงย่อยๆ ในแต่ละเสียงเช่น เสียงระเบิด เสียงรถ แยกจากกันโดยใช้ 1 บัฟเฟอร์ต่อ 1 เสียง แล้วจึงผสมเสียงรวมกันเป็นเสียงเดียวใน Primary Buffer

2.4.2 ข้อมูลเสียงที่ใช้ใน DirectSound

ข้อมูลเสียงที่ใช้ใน DirectSound จะเก็บอยู่ในรูปแบบของไฟล์เสียงซึ่งมีนามสกุล WAV ไฟล์เสียงนี้เก็บข้อมูลโดยใช้วิธีวัดความดังเสียงเป็นระยะทุกๆ เสี้ยววินาทีและเก็บสัญญาณเสียงไว้เป็นเลขดิจิตอล หากใช้หน่วยจำ 1 ไบต์ (8 บิต) จะได้ระดับเสียง 256 ระดับ หากต้องการความละเอียดสูงขึ้นก็ใช้หน่วยความจำ 2 ไบต์ ซึ่งจะได้จำนวนเสียง 65,536 ระดับ ความถี่ของเสียงจะมีผลต่อคุณภาพเสียง เช่น ถ้าต้องการคุณภาพเสียงระดับวิทยุซึ่งเป็นเสียงโมโน (ลำโพงสองข้างดังเท่ากัน) จำนวนระดับเสียง 256 ระดับ (8 บิต) ความถี่ 22,050 Hz

DirectSound และ DirectSoundBuffer บันทึกลักษณะต่างๆ ของข้อมูลเสียงเหล่านี้ไว้ในโครงสร้างชื่อว่า WAVEFORMATEX

```
Typedef struct {
    WORD    wFormatTag;
    WORD    nChannels;
    DWORD   nSamplesPerSec;
    DWORD   bAvgBytesPerSec;
    WORD    nBlockAlign;
    WORD    wBitsPerSample;
    WORD    cbSize;
} WAVEFORMATEX;
```

โดย

wFormatTag	เป็นตัวเลขที่ใช้กำหนดชนิดของเสียง ใน DirectSound ใช้ค่า WAVE_FORMAT_PMC
nChannels	หากต้องการใช้เสียงแบบโมโนให้ใช้ค่า 1 หากต้องการใช้เสียงสเตอริโอให้ใช้ค่า 2
nSamplesPerSec	ความถี่เสียงที่ใช้ (มีหน่วยเป็น Hz)
wBitsPerSample	ระดับเสียงหน่วยเป็นบิต (8 บิต = 256 ระดับ,16 บิต = 65,535 ระดับ)
bAvgBytesPerSec	จำนวนไบต์ที่ใช้เล่นเสียงต่อหนึ่งวินาที = nSamplesPerSec*nBlockAlign
nBlockAlign	จำนวนไบต์ที่ต้องใช้ต่อหนึ่งวินาที = nSamplesPerSec* nChannels/8

2.4.3 การเรียกใช้ออบเจ็กต์ DirectSound

การเรียกใช้ออบเจ็กต์ DirectSound จะเป็นการเรียกผ่านฟังก์ชัน InitDirectSound() ที่อยู่ในฟังก์ชัน WinMain() แล้วทำการโหลดไฟล์เสียงเข้าเก็บไว้ในบัฟเฟอร์

```
int WINAPI WinMain (HINSTANCE hInstance,HINSTANCE hPrevInstance,
                                PSTR szCmdLine,int nCmdShow)
{
        .............
        InitDirectSound();
        // โหลดไฟล์ *.wav มาเก็บไว้ที่บัฟเฟอร์
        LoadDirectSound(&dsbuffer1,"clear.wav");
        ..............
}
```

โดยฟังก์ชัน InitDirectSound() จะเป็นการติดต่อกับออบเจ็กต์ DirectSound โดยเป็นไปตามขั้นตอนดังนี้

เปิดออบเจ็กต์ directsound โดยใช้ฟังก์ชัน DirectsoundCrate และตั้งระดับการทำงานร่วมกัน

ฟังก์ชัน DirectSoundCreate มีค่าพารามิเตอร์ต่างๆ ดังนี้

LPCGUID lpGuid จะเป็นดีไวส์สำหรับการแสดงเสียงออกมาปกติจะกำหนดให้เป็นNULL

LPDIRECTSOUND * ppDS เป็นพอยเตอร์ที่ชี้ไปยังออบเจ็กต์ของ DirectSound

LPUNKNOWN pUnkOuter มักจะถูกกำหนดให้เป็น NULL

การเรียกใช้ออปเจ็กต์ของ DirectSound

1. การตั้งระดับการทำงานร่วมกันกับโปรแกรมอื่นๆ (Cooperative Level)

method SetCooperative มีค่าพารามิเตอร์ต่างๆ ที่นิยมใช้ดังนี้

HWND hwnd เป็นค่าแฮนเดิลของวินโดว์ส

DWORD dwLevel ระดับของรูปแบบของเสียง มีดังนี้

DSSCL_NORMAL จะทำงานร่วมกับแอพพลิเคชั่นอื่นได้ดี

DSSCL_PRIORITY จะสามารถปรับเปลี่ยนรูปแบบพื้นฐานได้

ตัวอย่างฟังก์ชัน InitDirectSound()

```
void InitDirectSound()
{
        // การเปิดออบเจ็กต์ DirectSound
        if(DS_OK != DirectSoundCreate(NULL,&dsound,NULL))
                Error("InitDirectSound DirectSoundCreate");
        // การตั้งระดับการทำงานร่วมกับโปรแกรมอื่น
        if(DS_OK != dsound->SetCooperativeLevel(hwnd,DSSCL_NORMAL))
                Error("InitDirectSound SetCooperativeLevel");
}
```

ในการโหลดไฟล์เสียงไปเก็บที่บัฟเฟอร์ จะใช้ฟังก์ชัน

```
void LoadDirectSound(LPDIRECTSOUNDBUFFER *buffer,char* filename)
{
        // ทำการเปิดไฟล์เสียง
        // หาข้อมูลไฟล์เสียง
        // หารูปแบบของข้อมูลไฟล์เสียง
        // อ่านรูปแบบของไฟล์เสียง
        // สร้างบัฟเฟอร์ของ directsound เพื่อเก็บข้อมูลของไฟล์เสียง
        // อ่านข้อมูลเสียงมาเก็บที่บัฟเฟอร์ที่ได้สร้างไว้ ขั้นตอนนี้เป็นการถ่ายข้อมูลจากไฟล์
ไปยังบัฟเฟอร์โดยอ่านมาเก็บไว้ในหน่วยความจำก่อน แล้วทำการปิดไฟล์เสียง
}
```

2. การเล่นเสียง

การเล่นเสียงจะใช้ฟังก์ชัน Play ซึ่งมีพารามิเตอร์ดังนี้

```
HRESULT Play ( DWORD dwReserve , DWORD dwPriority , DWORD dwFlags);
```

dwReserve ยังไม่ใช้ใน DirectX เวอร์ชันนี้ ให้กำหนดเป็นศูนย์

dwPriority หากเราเล่นเสียงหลายเสียงเป็นจำนวนมากในโปรแกรม โปรแกรมอาจเล่นเสียงได้ไม่ครบทุกเสียง ดังนั้นจึงมีการเลือกว่าจะเล่นเสียงไหนบ้างตามลำดับความสำคัญ (เช่น ในเกม เสียงของตัวละครตัวเอกสำคัญที่สุด เสียงตัวละครอื่นๆ รองลงมา) ตัวแปรนี้ ใช้บอกระดับความสำคัญ โดย 0 จะเป็นความสำคัญน้อยที่สุด 0xffffffff จะมีความสำคัญมากที่สุด

dwFlags ใช้บอกว่าต้องเล่นเสียงแบบวนซ้ำไปเรื่อยๆ หรือไม่ ถ้าต้องการวนซ้ำ ให้กำหนดเท่ากับ DDBPLAY_LOOPING หากต้องการเล่นเสียงครั้งเดียว ให้กำหนดเท่ากับศูนย์

```
dsbuffer1->SetCurrentPosition(0); // ให้เริ่มเล่นเสียงที่ตำแหน่งเริ่มต้น
dsbuffer1->Play(0,0,0); // เล่นแบบไม่วนซ้ำ ( รอบเดียว )
หรือถ้าต้องการให้เล่นแบบวนซ้ำ
dsbuffer1->Play(0,0,0DDBPLAY_LOOPING); // เล่นแบบไม่วนซ้ำ
```

ก่อนการจบโปรแกรมจะต้องทำการปิดออบเจ็กต์ต่างๆ ที่เคยเปิดไว้

3. การหยุดเล่นเสียง

เราสามารถหยุดการเล่นเสียงได้โดยใช้ฟังก์ชัน Stop ต่อจากนั้น หากต้องการเล่นเสียงต่อจากจุดเดิมก็สามารถใช้ฟังก์ชัน Play ต่อได้

ถ้าต้องการหยุดเสียงและเริ่มเล่นใหม่ที่จุดเริ่มต้น สามารถใช้ฟังก์ชัน SetCurrentPosition กำหนดจุดที่จะเล่นดังนี้

การหยุดพักชั่วคราวและเล่นต่อ

```
dsbuffer1 ->Stop(); //หยุดเล่นเสียง
dsbuffer1->Play(0,0,0); //เล่นต่อ
```

การหยุดเสียงและเริ่มใหม่

```
dsbuffer1 ->Stop();//หยุดเล่นเสียง
dsbuffer1->SetCurrentPosition(0); // ให้เริ่มเล่นเสียงที่ตำแหน่งเริ่มต้น
dsbuffer1->Play(0,0,0); // เล่นเสียง
```

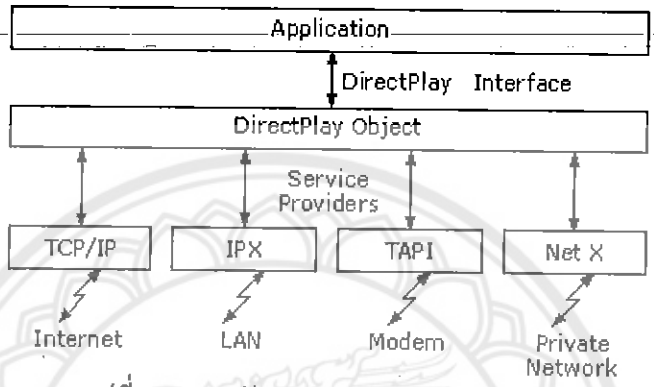
4. การปิดออบเจ็กต์ DirectSound และ DirectSound Buffer

```
//free all the buffers
dsbuffer1->Release(); dsbuffer1=NULL;
//free the interface to directsound functions
dsound->Release(); dsound=NULL;
```

2.5 DirectPlay

DirectPlay เป็นส่วนประกอบหนึ่งของ DirectX ซึ่งจะช่วยเกี่ยวกับการเขียนเกมแบบผู้เล่นหลายคน

2.5.1 DirectPlay Architecture (สถาปัตยกรรมของ DirectPlay)



รูปที่ 2.2 สถาปัตยกรรมของ DirectPlay

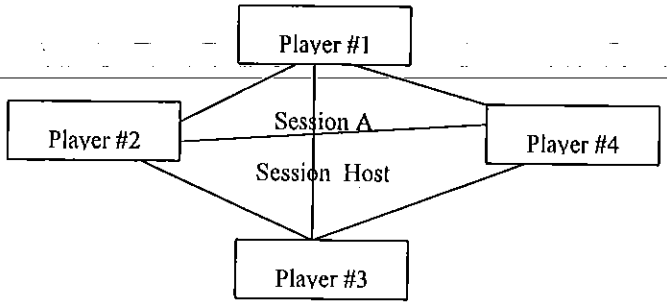
2.5.2 การเชื่อมต่อภายในเครือข่าย

ในขั้นตอนแรกของการเริ่มใช้ DirectPlay จะต้องเลือก service provider ที่จะใช้ ซึ่ง service provider หมายถึงชนิดของ protocol ที่จะใช้ในการติดต่อสื่อสาร protocol ที่นิยมใช้ได้แก่ TCP/IP, IPX โดย Application จะใช้ service provider ในการติดต่อตามจุดต่างๆ ภายในเครือข่าย

รูปแบบของการเชื่อมต่อมีดังนี้

1. **Peer – to -- Peer**

เป็นการเชื่อมต่อโดยจะเก็บข้อมูลต่างๆ ไว้ที่เครื่องคอมพิวเตอร์ ของผู้ใช้แต่ละคนโดยไม่มีคอมพิวเตอร์ส่วนกลาง

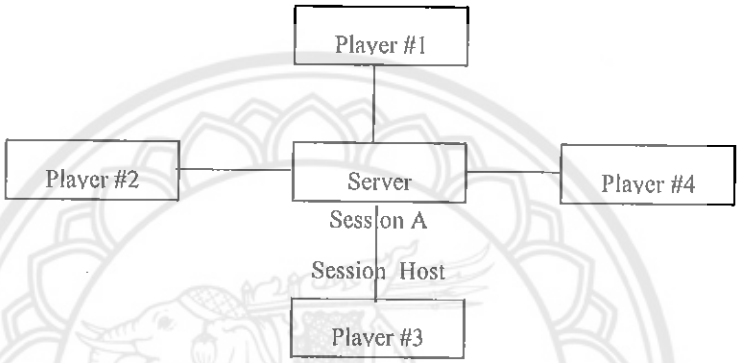


รูปที่ 2.3 การเชื่อมต่อแบบ Peer-to-Peer

ในตอนแรก จะต้องสร้าง session ขึ้นมาก่อนโดยเครื่องใดเครื่องหนึ่ง เช่น ผู้เล่นเครื่อง # 1 สร้าง session ขึ้นมาเป็น session host และจะรอรับการติดต่อจากเครื่องอื่นที่ต้องการเข้ามาร่วมเล่น

ด้วย โดยการติดต่อเป็นหน้าที่ของ DirectPlay เมื่อเครื่องอื่นติดต่อเข้ามาใน session ในการส่งข้อความถึงกันสามารถส่งถึงกันได้โดยตรง ไม่ต้องผ่าน session host และผู้เล่นเครื่อง # 1 ออกจากเกมที่เล่นอยู่ไป จะมีคอมพิวเตอร์เครื่องใดเครื่องหนึ่งที่จะถูกเลือกเป็น session host โดยอัตโนมัติ และเกมสามารถดำเนินต่อไปได้เมื่อไม่มี ผู้เล่นเครื่อง # 1

2. Client/Server



รูปที่ 2.4 การเชื่อมต่อแบบ Client-Server

การเชื่อมต่อชนิดนี้เป็นผลจากการเลือกใช้ Service Provider แบบ Internet Gaming server หรือ dial – up networking รูปแบบของเครือข่ายแบบ Client/Server จะต้องมีเครื่องคอมพิวเตอร์หลักอยู่ เพื่อทำหน้าที่เป็นศูนย์กลางของระบบในการจัดการให้บริการ การส่งข้อความถึงกันจะต้องผ่าน session host การเชื่อมต่อแบบนี้มีข้อดีคือ สามารถรองรับผู้เล่นจำนวนมากได้

2.5.3 รูปแบบการติดต่อภายในเครือข่ายโดยใช้ Direct play

สามารถเชื่อมต่อโดยใช้ Protocal ต่างๆ ดังนี้

1. IPX/SPX

IPX(Internetwork Packet Exchange)

ใช้ในการส่งข้อมูลระหว่างอุปกรณ์ที่อยู่ในเครือข่ายต่างกัน แต่จะไม่มีการตรวจสอบความผิดพลาดในการส่งข้อมูล

SPX(Sequeced Packet Exchange)

เป็นโปรโตคอลที่ขยายความสามารถของ IPX โดยจะมีการตรวจสอบการส่งข้อมูล

2. TCP/IP

เป็นโปรโตคอลที่ใช้ในการติดต่อสื่อสารระหว่างเครือข่ายและเป็นโปรโตคอลหลักของเครือข่ายอินเตอร์เน็ต ถ้าเครือข่ายเสียหาย TCP/IP สามารถหาเส้นทางใหม่เพื่อส่งข้อมูลไปถึงจุดหมายปลายทางได้ TCP/IP ประกอบด้วย

- TCP เป็นโปรโตคอลที่ใช้ส่งข้อมูลระหว่างอุปกรณ์สองตัวในเครือข่าย TCP/IP โดย TCP จะใช้พอร์ตเสมือน (Virtual Port) ในการเชื่อมต่อและคอยตรวจสอบการส่งข้อมูล
- IP (Internet Protocal) ทำหน้าที่จัดการเกี่ยวกับที่อยู่ของข้อมูลและส่งไปยังปลายทางที่ถูกต้องในเครือข่าย TCP/IP

2.5.2 การใช้งาน DirectPlay

ก่อนการใช้งาน DirectPlay จะต้องมีการกำหนดค่า GUID(globally unique identifiers) ซึ่ง DirectPlay จะใช้ค่า GUID ในการระบุ DirectPlay objects เพื่อใช้ในการติดต่อสื่อสารผ่านเครือข่าย ดังนี้

```
GUID CHATTER_GUID = {
0x949a21e1,
0x6179,
0x11cf,
{0x95, 0x4f, 0x00, 0xaa, 0x00, 0x6c, 0x26, 0x57}
};
```

จากนั้นจะเริ่มการสร้าง DirectPlay Object และนำมาใช้งาน โดยมีวิธีการตามขั้นตอนต่อไปนี้

1. **การสร้าง DirectPlay Object** จะใช้ฟังก์ชัน CoCreateInstance ดังต่อไปนี้

```
if (FAILED( CoCreateInstance( CLSID_DirectPlay,
                              NULL, CLSCTX_ALL, IID_IDirectPlay3A,
                              ( LPVOID* ) &lpDPTemp ) ))
    return( FALSE );
```

เมื่อมีการสร้าง DirectPlay Object จะสามารถกำหนดลักษณะของตัวอักษรได้ 2 แบบ คือ ANSI หรือ Unicode ถ้าเป็น ANSI จะใช้พารามิเตอร์ IID_IDirectPlay3A ส่วน Unicode จะใช้ IID_IDirectPlay3

2. **Enumerate connection shortcuts** จะระบุหรือการเริ่มสร้างการติดต่อโดยจะใช้ฟังก์ชัน EnumConnection ซึ่งจะมีการใช้งานดังนี้

```
if FAILED( lpDP->EnumConnections( NULL,
                                  ( LPDPENUMCONNECTIONSCALLBACK )EnumConnection,
                                  ( LPVOID )GetDlgItem( hWnd, IDC_CONNECTIONS ), 0 ) )
```

```
{
    MessageBox( hWnd, "Couldn't enumerate connections.",
                "Error", MB_OK );
    EndDialog( hWnd, FALSE );
}
```

3. การ Initialize Connection ในขั้นตอนนี้จะเป็นตัวกำหนดค่าเริ่มต้นให้ DirectPlay Object เพื่อใช้ในการติดต่อหรือสร้าง Connection โดยจะใช้ฟังก์ชัน InitializeConnection ดังนี้

```
if FAILED( lpDP->InitializeConnection( lpConnection, 0 ) )
    {
        MessageBox( hWnd,
        "Error initializing DirectPlay object.",
        "Error", MB_OK );
    }
```

4. ระบุและกำหนดการติดต่อให้ Session ที่มีอยู่ โดยจะใช้ฟังก์ชัน EnumSessions ซึ่งมีรูปแบบการใช้งานดังนี้

```
lpDP->EnumSessions( &dpDesc, 0,
    ( LPDPENUMSESSIONSCALLBACK2 )EnumSession,
    ( LPVOID ) GetDlgItem( hWnd, IDC_SESSIONS ),
    dwSessions );
```

5. สร้าง Session ใหม่ เป็นการเปิดหรือสร้าง Session ขึ้นมา โดยจะใช้ฟังก์ชัน Open ซึ่งมีรูปแบบการใช้งานดังนี้

```
if FAILED( lpDP->Open( &dpDesc, DPOPEN_JOIN ) )
    {
        MessageBox( hWnd, "Could not join session.",
                    "Error", MB_OK );
        EndDialog( hWnd, FALSE );
    }
```

จากขั้นตอนที่ผ่านมาทั้งหมด จะเป็นการสร้างการติดต่อผ่านเครือข่ายโดยเปิด Session ไว้เพื่อรอรับการ Connection จากเครื่องอื่น โดยจะใช้ DirectPlay Object เป็นตัวที่ใช้ในการเชื่อมต่อ

2.6 DirectMusic

DirectMusic มีลักษณะคล้ายกับ DirectSound แต่จะมีฟังก์ชันที่คอยสนับสนุนการใช้งาน ไฟล์ MIDI (Musical Instrument Digital Interface)

การเรียกใช้ออกเจ็กต์ของ DirectMusic

ขั้นตอนที่ 1 ก่อนที่จะทำการเรียก DirectMusic จะต้องทำการ Initialize COM ดังนี้

```
if (FAILED(CoInitialize(NULL)))
{
    //Terminate the application
}
```

ขั้นตอนที่ 2 สร้าง performance object

ออบเจ็กต์ที่สำคัญ ของ DirectMusic คือ performance object ซึ่งถูกสร้างมาจากพื้นฐานของ COM ฟังก์ชัน CoCreateInstance ตามตัวอย่างดังนี้

```
IDirectMusicPerformance* CreatePerformance(void)
{
    IDirectMusicPerformance* pPerf;

    if (FAILED(CoCreateInstance(
        CLSID_DirectMusicPerformance,
        NULL,
        CLSCTX_INPROC,
        IID_IDirectMusicPerformance2,
        (void**)&pPerf
        )))
    {
        pPerf = NULL;
    }
    return pPerf;
}
```

เมื่อ performance object ถูกสร้างขึ้นเราจะต้องทำการ Initialize โดยเรียกจากการใช้ เมธอด Init() เมธอดนี้ จะสร้างทำการสร้าง DirectMusic Object ขึ้นมา

```
if (FAILED(g_pPerf->Init(NULL, NULL, NULL)))
{
    // Failure -- performance not initialized
```

```
}
```

จากนั้นทำการ AddPort ให้กับ performance ด้วยการเรียกเมธอด AddPort โดยให้ performance เป็น NULL เพื่อจะได้เป็น Default port

```
if (FAILED(pPerf->AddPort(NULL)))
{
    // Failure -- port not initialized
}
```

ขั้นตอนที่ 3 สร้าง Loader

ในการสั่ง Load ไฟล์ midi อย่างแรกที่ต้องทำคือ การสร้าง DirectMusicLoader Object ดังตัวอย่างต่อไปนี้

```
IDirectMusicLoader* CreateLoader(void)
{
    IDirectMusicLoader* pLoader;
    if (FAILED(CoCreateInstance(
        CLSID_DirectMusicLoader,
        NULL,
        CLSCTX_INPROC,
        IID_IDirectMusicLoader,
        (void**)&pLoader)))
    {
        pLoader = NULL;
    }
    return pLoader;
}
```

ขั้นตอนที่ 4 Load ไฟล์ Midi

ในการ Load ไฟล์ Midi จะใช้ DMUS_OBJECTDESC ยังเป็น stucture ที่ทำงานคล้ายกับ Structure ของ DirectDraw หรือ buffer desc ของ DirectSound ซึ่ง stucture DMUS_OBJECTDESC นี้ จะถูกใช้โดย Loader object เพื่อใช้ระบุค่าต่างๆ ของ object ในการ Load ไฟล์

ขั้นตอนที่ 5 เล่นเพลงไฟล์ Midi ซึ่งทำได้ดังตัวอย่างต่อไปนี้

```
void PlaySegment (void)
{
    HRESULT hr;
    hr = performance->PlaySegment (segment, // segment to play
                                   0, // behavior flags (มักจะให้เป็น 0)
                                   0, // start time (มักจะให้เป็น 0)
                                   NULL); // segment state (มักจะให้เป็น NULL)
    if (FAILED(hr))
    {
        //PlaySegment failed
    }
}
```

ขั้นตอนที่ 6 Shut Down DirectMusic

ก่อนที่จะออกจากโปรแกรมจะต้องทำการ release objects ทั้งหมดที่ได้สร้างขึ้น และปิดการทำงานของ COM โดยใช้ CoUninitialize()

บทที่ 3
การออกแบบและพัฒนาโปรแกรม

3.1 การออกแบบแนวเกมและการดำเนินเรื่องของเกม

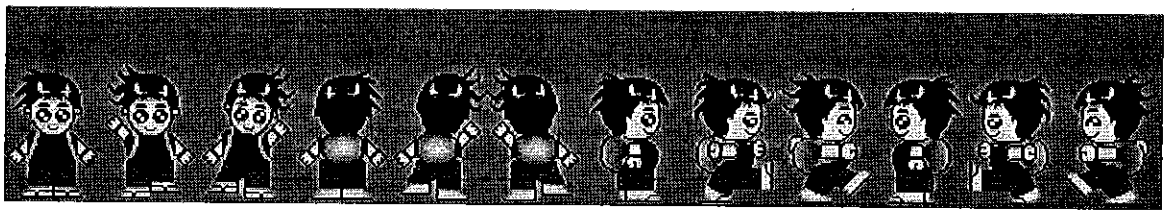
เกมท่องเที่ยวภาคเหนือที่ได้จัดทำขึ้นเป็นเกมในลักษณะ RPG Adventure มีเนื้อเรื่องของเกมคือ ผู้เล่นจะต้องออกเดินทางท่องเที่ยวไปตามจังหวัดและสถานที่ต่างๆ เพื่อเก็บข้อมูล และต้องใช้ข้อมูลและความรู้ที่ได้จากการท่องเที่ยวในการตอบคำถามในตอนสุดท้ายของเกม

เมื่อเริ่มต้นเล่นเกมผู้เล่นทุกคนจะมี สมุดบันทึกเหตุการณ์ เงิน และช่องเก็บสิ่งของต่าง ๆ โดยเงิน และ ช่องเก็บสิ่งของจะแสดงอยู่ที่ แถบสถานะ ผู้เล่นได้รับมอบหมายให้เดินทางค้นหาสมบัติและนำสมบัตินี้กลับมาช่วยประเทศชาติ แต่การจะได้สมบัตินั้น จะต้องตอบคำถามจากเจ้าของสมบัติให้ได้ทุกข้อ ซึ่งจะเป็นคำถามเกี่ยวกับสถานที่ท่องเที่ยว และจังหวัดต่างๆ ในภาคเหนือ

การเอาชนะเกมนั้นผู้เล่นจึงจำเป็นต้องท่องเที่ยวไปในสถานที่ต่างๆ เพื่อหาข้อมูล มาใช้ในการตอบคำถาม ในตอนสุดท้ายของเกม แต่การที่ผู้เล่นจะมีสิทธิในการตอบคำถามนั้น ผู้เล่นจะต้องมีคุณสมบัติเพียงพอ คือ ต้องทำภารกิจต่างๆ ที่ได้รับจากตัวละครต่างๆ ในเกม ให้เสร็จสมบูรณ์ทั้งหมดก่อน ดังนั้นผู้เล่นจึงต้องพูดคุยกับตัวละครต่างๆ ภายในเกมด้วย เพื่อรวบรวมข้อมูล และรับภารกิจโดยภารกิจที่ได้รับมาจะถูกบันทึกอยู่ในสมุดบันทึกประจำตัวผู้เล่น เมื่อผู้เล่นพูดคุย กับตัวละครต่างๆ ในเกมแล้วจะสามารถทำภารกิจที่ได้รับมอบหมาย ให้สำเร็จได้ ซึ่งจะทำให้ผู้เล่นมีสิทธิในการเข้าไปตอบคำถามได้ โดยผู้เล่นตอบคำถามได้ทั้งหมด ผู้เล่นจะได้รับสมบัติและนำกลับมาช่วยประเทศชาติได้จึงเป็นผู้ชนะ

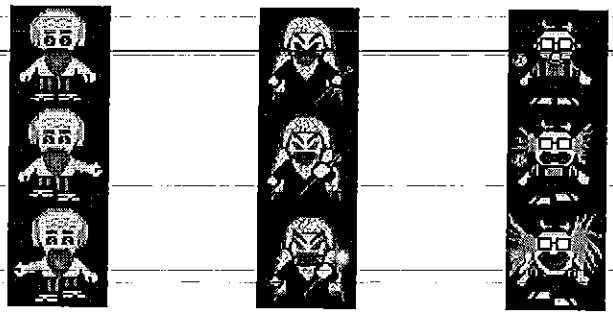
3.2 การจัดเตรียมทรัพยากรและข้อมูลต่างๆ เพื่อนำมาใช้ในการสร้างเกม

3.2.1 การออกแบบสร้างตัวละครของผู้เล่น (ตัวผู้เล่น) โดยจะเก็บเป็นภาพทั้งหมด 12 ภาพ คือ ภาพที่ตัวละครเดินไปในทิศทั้ง 4 ทิศ คือ ทิศเหนือ ทิศตะวันออก ทิศใต้ และทิศตะวันตก แบ่งเป็นทิศและ 3 เฟรม ในแต่ละเฟรมจะมีขนาด 50x100 พิกเซล ดังรูป 3.1



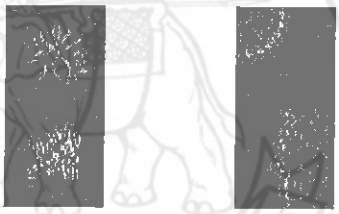
รูปที่ 3.1 ตัวละครของผู้เล่น

3.2.2 การออกแบบสร้างตัวละครต่างๆ ที่อยู่ภายในเกม โดยมีตัวละครในแผนที่ทั้งหมด 16 ตัว แต่ละตัวจะเก็บภาพทั้งหมดเป็นภาพ 3 เฟรม เพื่อให้ดูมีการเคลื่อนไหว ดังรูปที่ 3.2



รูปที่ 3.2 บางส่วนของตัวละครประกอบที่อยู่ภายในเกม

3.2.3 การออกแบบสร้างภาพส่วนประกอบต่างๆ ภายในเกม เช่น ต้นไม้ ภูเขา ดอกไม้ พุ่มไม้ น้ำ หิน สิ่งของต่างๆ เป็นต้น



รูปที่ 3.3 ตัวอย่างของภาพ พุ่มไม้และน้ำ

โดยรูปภาพทั้งหมด ยกเว้นภาพของตัวละครหลัก จะจัดเก็บเรียงเป็นแถวยาวลงมา โดยแต่ละภาพ จะมีขนาดเท่ากันทั้งหมด 50x50 พิกเซล

3.2.4 การออกแบบสร้างเมนู

โดยจะมีให้เลือกว่าจะเล่นแบบผู้เล่นคนเดียวหรือแบบผู้เล่นสองคน

3.2.5 การเตรียมข้อมูลเกี่ยวกับจังหวัดต่างๆ ทั้ง 17 จังหวัด โดยแต่ละจังหวัดในภาคเหนือประกอบด้วย

1. ตราประจำจังหวัด 17 จังหวัด โดยแต่ละตราประกอบด้วยภาพ 3 เฟรม
2. คำอธิบายตราประจำจังหวัด
3. คำขวัญประจำจังหวัด
4. อาณาเขตแต่ละจังหวัด
5. ภาพสถานที่ท่องเที่ยวที่น่าสนใจในแต่ละจังหวัด

3.2.6 เตรียมข้อมูลเกี่ยวกับสถานที่ท่องเที่ยวที่สำคัญในภาคเหนือ โดยประกอบด้วย รายละเอียดของสถานที่ท่องเที่ยวและภาพของสถานที่ท่องเที่ยวนั้น

3.2.7 เตรียมบทพูดของตัวละครทั้ง 16 ตัว โดยแต่ละตัวอาจมีต้องการแตกต่างกัน ซึ่งผู้เล่นต้องทำตามความต้องการของตัวละคร

3.2.8 ออกแบบและสร้างแถบสถานะ ซึ่งประกอบด้วย เงิน และช่องใส่สิ่งของทั้งหมด 7 ช่อง

รูปที่ 3.4 แถบสถานะของผู้เล่นภายในเกม

3.2.9 การออกแบบสร้าง Information ซึ่งแสดงแผนที่อาณาเขตแต่ละจังหวัด และตำแหน่งสถานที่ท่องเที่ยวสำคัญในแผนที่

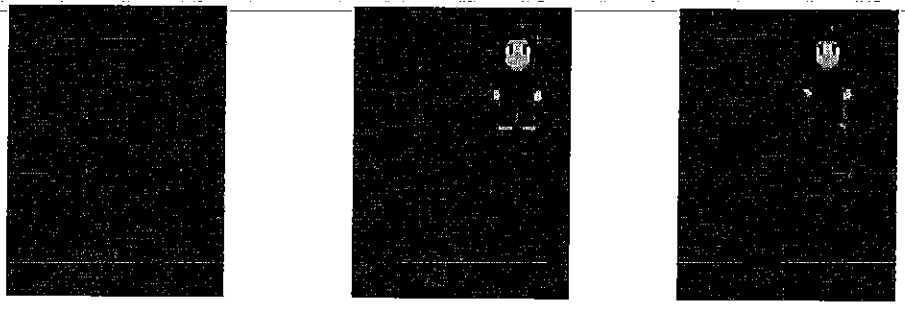
3.2.10 เตรียมคำถามซึ่งจะใช้ในตอนสุดท้ายของเกม โดยจะในแต่ละครั้งจะ สุ่มคำถามมา 16 คำถาม

3.3 การเขียนโปรแกรมให้เป็นไปตามเนื้อเรื่องที่ได้ออกแบบไว้

3.3.1 การทำแผนที่โดยใช้ Tile

การวาดแผนที่โดยใช้ Tile จะเป็นการนำภาพ เล็ก ๆ ที่มีขนาดเท่ากันมาต่อกันขึ้นอยู่กับการกำหนด แต่ต้องเหมาะสมกับความละเอียดในการแสดงผลทางจอภาพ ยกตัวอย่างเช่น ถ้าแสดงผลโดยใช้ความละเอียด 640 x 480 พิกเซล ควรจะใช้ Tile ขนาด 32x32 พิกเซล แต่เนื่องจากเกมท่องเที่ยวไทย ใช้ความละเอียดในการแสดงผล 800x600 ดังนั้นจึงใช้ Tile ที่มีขนาด 50x50 พิกเซล

Layer แผนที่ที่จะใช้ทั้งหมด 2 Layer โดย Layer แรกจะเป็นพื้นหญ้า ภูเขา ทางเดิน ต้นไม้ Layer ที่สอง จะเป็นส่วนบนของต้นไม้ที่บังถนนอยู่ ในการวาดภาพ Tile จะวาดทีละ Layer โดย เริ่มจาก Layer ที่ 1 จากนั้นจึงวาดตัวละคร แล้วจึงวาด Layer ที่ 2 ดังรูปที่ 3.5



รูปที่ 3.5 ตัวอย่างการวาดแผนที่

สำหรับ Tile animation สามารถกำหนดได้ว่าจะให้ Tile ใดเป็น Tile ที่เคลื่อนไหว และจะให้ Tile ใดเป็นเฟรมถัดไป ซึ่งสามารถนำมาทำเป็นตัวประกอบที่มีการเคลื่อนไหว ลมพัดต้นไม้ลู่ไปตามลม มีคลื่นน้ำ เป็นต้น

จากนั้นทำการกำหนดขนาดของแผนที่ว่าจะให้มีขนาดเท่าไหร่ แผนที่ในเกมนี้มีขนาดความกว้าง 48 Tile ยาว 60 Tile การจัดเก็บรูปภาพของ Tile จะเก็บเป็นลักษณะแนวยาวลงมา จะมีภาพ Tile ทั้งสิ้น 256 Tile แต่ละภาพให้เป็นเลข 0 – 255

ตัวอย่างการเก็บข้อมูลแผนที่ลงในไฟล์ *.dat โดยจะเก็บเป็นหมายเลข Tile ซึ่งมีความกว้างแผนที่ 48 Tile ความยาว 60 Tile

```
struct map
{
        unsigned char Base[60][48];      // Layer ที่ 1
        unsigned char Fringe[60][48];    // Layer ที่ 2
};
// จากนั้นทำการกำหนดค่าใน Array ของแต่ละ Layer แล้วทำการเก็บข้อมูลลงใน ไฟล์ *.dat เพื่อทำการเรียกใช้ต่อไป

FILE  *fp;
struct  map  Tile;
// ทำการกำหนดค่าหมายเลขของแต่ละ Tile
// Row #0
Tile.Base[0][0] = 0;Tile.Base[0][1] = 0; Tile.Base[0][2] = 0;
Tile.Base[0][3] = 0;Tile.Base[0][4] = 0; Tile.Base[0][5] = 0;
Tile.Base[0][6] = 0;Tile.Base[0][7] = 0; Tile.Base[0][8] = 0;
        .....
        .....
Tile.Base[59][39] = 0;Tile.Base[59][40] = 0; Tile.Base[59][41] = 0;
Tile.Base[59][42] = 0;Tile.Base[59][43] = 0; Tile.Base[59][44] = 0;
Tile.Base[59][45] = 0;Tile.Base[59][46] = 0; Tile.Base[59][47] = 0;

Tile.Fringe[20][21] = 114; Tile.Fringe[23][32] = 114;
Tile.Fringe[28][29] = 114; Tile.Fringe[30][21] = 114;
```

```
// เก็บข้อมูลลงไปไฟล์โดยให้มีชื่อว่า MapTile.dat
fp = fopen ("c:\\source\\MapTile.dat","w");
fwrite((char*)&Tile,sizeof(Tile),1,fp);
if(fp == NULL)printf("ERROR");
fclose(fp);
```

ในการเรียกใช้จะเรียกจากไฟล์ที่ชื่อว่า Maptile.dat

```
#define NUM_TILES 256
typedef struct TILE_type
{
  RECT rcLocation;        // หมายเลขของ Tile
  int bWalkOK;            // ใช้กำหนดว่าให้ตัวละครสามารถเดินผ่านได้หรือไม่
  int nAnimSpeed;         // เป็นค่าที่delay เพื่อจะแสดงต่อไปของ tile ที่มีการเคลื่อนไหว
  TILE_type *lpNext;      // ถ้าเป็น Tile ที่มี animation *lpNext จะเป็นพอยด์เตอร์ที่ชี้ไปที่
                          Tile stucture เพื่อแสดง Tile ต่อไป ในการ animation
  UINT uiFrames;          // frame counter
} TILEMAP, FAR* LPTILE;

TILEMAP tileData[NUM_TILES];
LPTILE lptile[NUM_TILES];

FILE *fp;
int x;
fp = fopen ("maptile.dat","r");
if(fp == NULL)printf("ERROR");

// read maptile and close the file
fread((char *)&Tile,sizeof(Tile),1,fp);
fclose(fp);
```

// อ่านข้อมูลที่ได้จากไฟล์ maptile.dat มาเก็บไว้ที่ sturcture tile ที่ประกาศไว้ข้างต้น

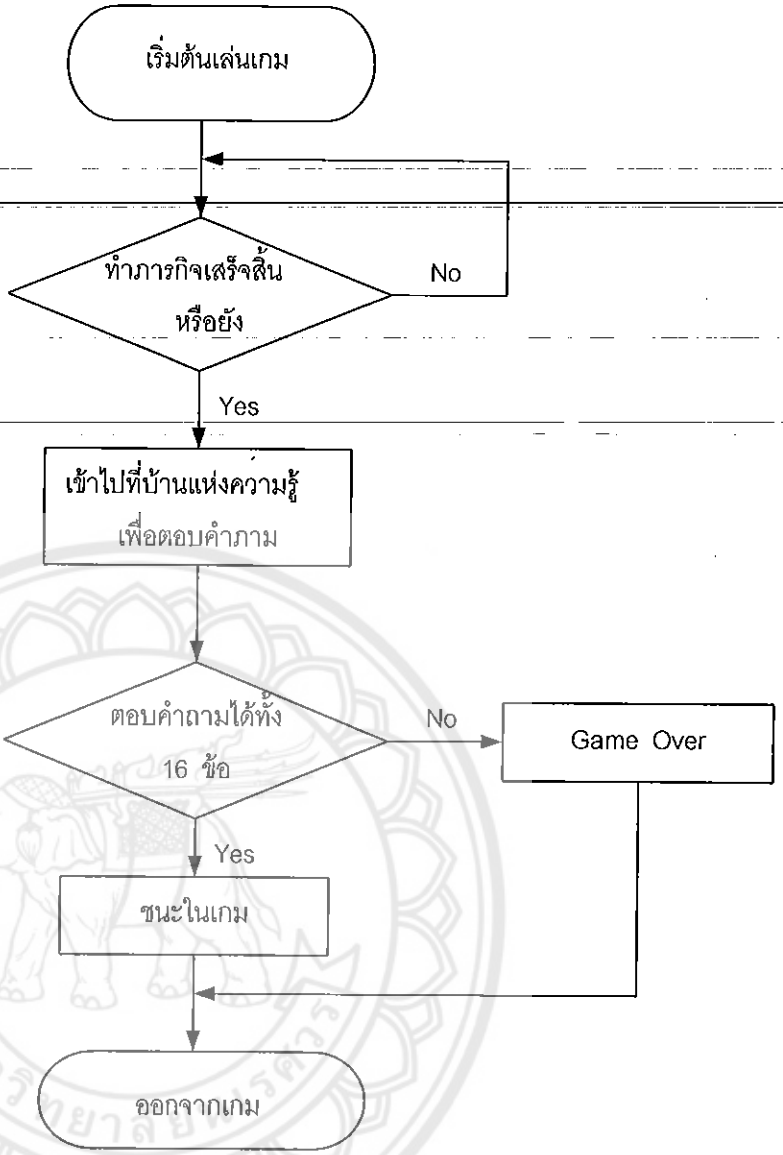
```
for (x=0; x<NUM_TILES; x++)
```

```
            lptile[x] = &tileData[x];
        for (x=0; x<NUM_TILES; x++)
        {
            // set all the RECTs on the surface
            RECT rcTemp = {0,x*50, 50,x*50+50 };
            tileData[x].rcLocation = rcTemp;
            tileData[x].uiFrames = 0;
            tileData[x].bWalkOK = FALSE; // ในตอนแรกจะกำหนดให้เป็น Tile เดินผ่านไม่
ได้
        }
        // จากนั้นให้การกำหนดว่า Tile ใดจะเป็น Tile ที่เดินผ่านได้ซึ่งสามารถกำหนดได้ดังนี้
            tileData[64].bWalkOK = TRUE;
            tileData[110].bWalkOK = TRUE;
        // สำหรับ tile animation จะมีการกำหนดว่าจะให้ Tile ใดเป็นเฟรมถัดไป พร้อมทั้งมีการ
กำหนด delay ของแต่ละเฟรม โดยถ้า delay มีค่ามาก จะทำให้เวลาในการแสดงเฟรมของนั้นนานขึ้น
        tileData[130].lpNext = &tileData[131];tileData[130].nAnimSpeed = 30;
        tileData[131].lpNext = &tileData[132];tileData[131].nAnimSpeed = 5;
        tileData[132].lpNext = &tileData[130];tileData[132].nAnimSpeed = 5;
        ในการวาดรูป Tile จะทำได้โดยการเรียกจาก Stucture Tile และการวาดจะวาด Tile Layer
ที่ 1 ก่อนจากนั้นวาดตัวละคร แล้วจึงวาด Layer ที่ 2 ถัดมา
        // วาด Layer 1
        for (x=0; x<16; x++)
            for (y=0; y<12; y++)
            {
                byTile = Tile.Base[stRow+y][stCol+x];
                lpDDSBack->BltFast(x*50,y*50,
                                   lpDDSOff[5],
                                   &lptile[byTile]->rcLocation,
                                   DDBLTFAST_WAIT);
            }
        // วาดตัวละคร
        DrawCharacter(&player.move);
```

```
//วาด Layer 2
for (x=0; x<16; x++)
        for (y=0; y<12; y++)
        {
                byTile = Tile.Fringe[stRow+y][stCol+x];
                lpDDSBack->BltFast(x*50,y*50,
                                lpDDSOff[5],
                                &lptile[byTile]->rcLocation,
                                DDBLTFAST_WAIT|
                                DDBLTFAST_SRCCOLORKEY);
        }
```

3.3.2 โครงสร้างของเกม

เมื่อเริ่มต้นเกม ผู้เล่นจะต้องเดินทางท่องเที่ยวไปยังสถานที่ต่างๆ และทำภารกิจให้เสร็จสิ้นทั้ง 4 ภารกิจ จึงจะมีสิทธิ์ไปตอบคำถาม ถ้าผู้เล่นยังทำภารกิจไม่เสร็จจะต้องกลับไปทำภารกิจให้เสร็จ เมื่อผู้เล่นทำภารกิจเสร็จแล้วจะสามารถเข้าไปตอบคำถาม โดยจะต้องตอบคำถามให้ได้ทั้งหมด 16 คำถาม โดยไม่ผิดจึงจะเป็นผู้ชนะ ถ้าผู้เล่นตอบผิดแม้เพียงข้อเดียวก็จะแพ้



รูปที่ 3.6 โครงสร้างของเกม

3.3.3 โครงสร้างของการทำงานส่วนของคำถามภายในเกม

เมื่อผู้เล่นทำภารกิจเสร็จสิ้นทั้งหมดแล้ว จะสามารถเข้าไปตอบคำถามได้ โดยคำถามจะมีทั้งหมด 16 คำถาม แบ่งออกเป็น 4 ช่วง ช่วงละ 4 คำถาม ดังนี้

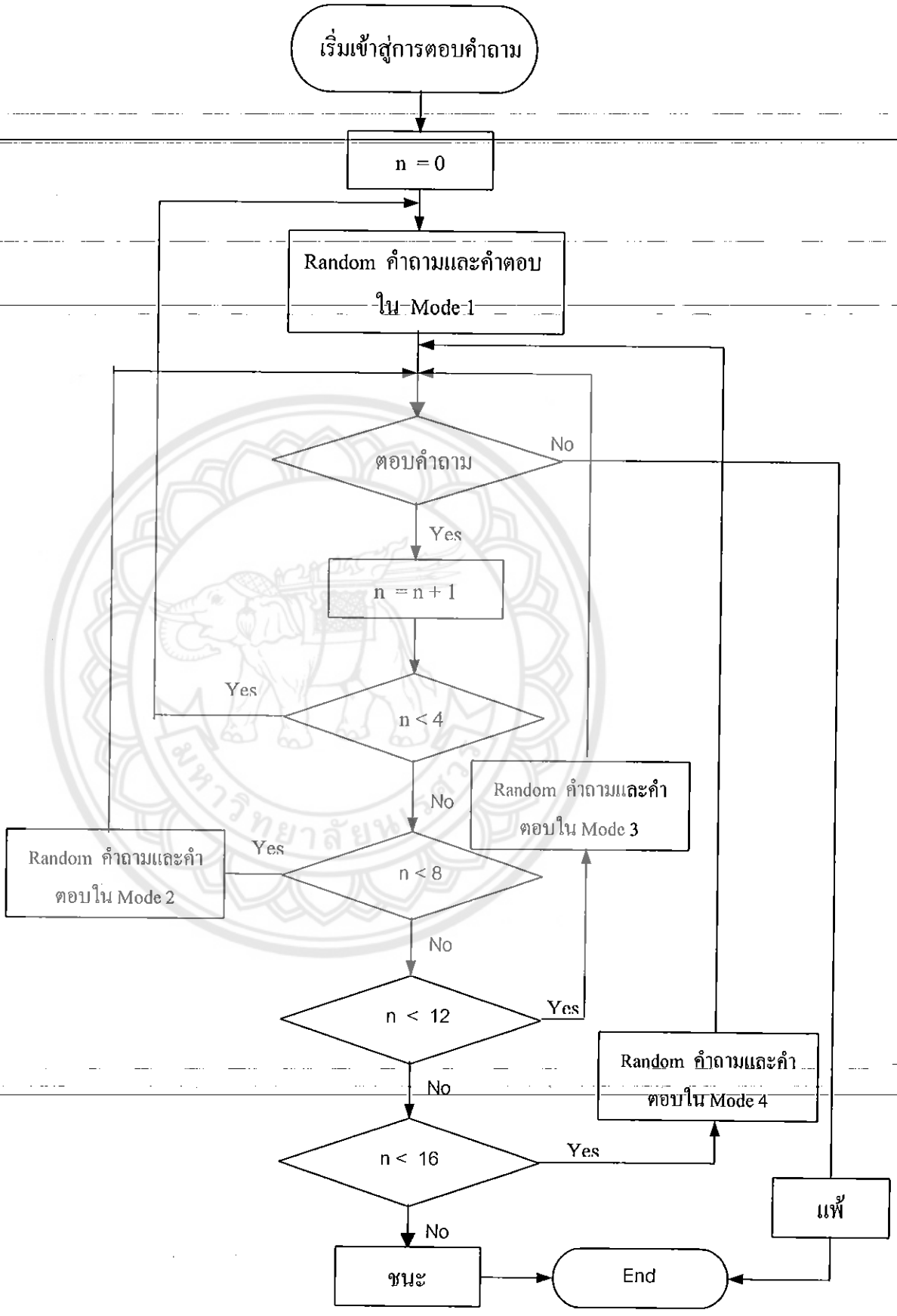
- ข้อ 1-4 เป็นคำถามเกี่ยวกับชื่อสถานที่ท่องเที่ยวต่างๆ
- ข้อ 5-8 เป็นคำถามเกี่ยวกับตราประจำจังหวัด
- ข้อ 9-12 เป็นคำถามเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัด
- ข้อ 13-16 เป็นคำถามเกี่ยวกับคำขวัญประจำจังหวัด

จากรูป 3.6 เมื่อเข้าสู่การตอบคำถาม จะมีการกำหนดค่าให้ตัวแปร n มีค่าเป็น 0 โดยตัวแปร n จะเป็นตัวกำหนดว่าเป็นข้อที่เท่าไร หลังจากกำหนดให้ n=0 แล้ว จะมีการ สุ่มคำถามและคำตอบใน

mode 1 คือเป็นคำถามเกี่ยวกับชื่อสถานที่ท่องเที่ยวต่างๆ ถ้าผู้เล่นตอบผิด จะแพ้และจบเกม แต่ถ้าตอบถูก จะมีการเพิ่มค่า n ขึ้นไปอีก 1 ค่า เป็นการเปลี่ยนเป็นข้อต่อไป(ข้อ 2) และจากนั้นจะตรวจสอบว่าเป็นคำถามในโหมดใด โดยตรวจจากค่า n ถ้า n<4 คือ อยู่ใน mode 1 จะกลับไปทำการสุ่มคำถามและคำตอบในโหมด 1 และให้ผู้เล่นตอบคำถามข้อต่อไป ถ้าผู้เล่นตอบคำถามถูก จะเพิ่มค่า n ขึ้นอีก 1 จนเมื่อค่า n>= 4 คือ n ไม่น้อยกว่า 4 แล้ว จะตรวจสอบต่อไปว่า ค่า n<8 หรือไม่ ถ้าใช่ จะทำการ random คำถามและคำตอบใน mode 2 ซึ่งเป็นคำถามเกี่ยวกับตราประจำจังหวัด

ถ้าผู้เล่นตอบถูก จะทำการเพิ่มค่า n อีก 1 ค่า และ random คำถามและคำตอบใน mode 2 ใหม่ จนค่า n=8 จะเปลี่ยนไป random คำถามและคำตอบใน mode 3 จากนั้น ถ้าผู้เล่นตอบถูกอีก จะเพิ่มค่า n ขึ้นเรื่อยๆ จนค่า n=12 จะเปลี่ยนเป็นคำถามใน mode 4 ซึ่งเป็นคำถาม mode สุดท้าย ถ้าผู้เล่นสามารถตอบคำถามถูกทั้งหมด จนค่า n=16 แล้ว ผู้เล่นจะเป็นผู้ชนะในเกม และจบเกม

การจะชนะเกมได้นั้น ผู้เล่นจะต้องตอบคำถามทั้ง 16 ข้อให้ถูกต้องทั้งหมด ถ้าตอบผิดเพียงข้อเดียวจะแพ้และจบเกม



รูปที่ 3.7 ขั้นตอนการทำงานในส่วนของคำถามภายในเกม

บทที่ 4

ผลการทดสอบโปรแกรมและวิเคราะห์ผล

4.1 จุดประสงค์ของการทดสอบโปรแกรม

1. เพื่อทดสอบโปรแกรมว่าสามารถทำงานได้ตามวัตถุประสงค์ที่ต้องการหรือไม่ และผลการทำงานของโปรแกรมมีประสิทธิภาพเพียงใด

2. เพื่อตรวจสอบหาข้อผิดพลาดของโปรแกรมที่เกิดขึ้นแล้วนำข้อผิดพลาดเหล่านั้นมาปรับปรุงแก้ไขให้ดีขึ้น

4.2 ขั้นตอนการทดสอบการทำงานของโปรแกรม

1. ติดตั้งโปรแกรมเกมท่องเที่ยวภาคเหนือลงบนเครื่องคอมพิวเตอร์

2. ติดตั้งโปรแกรม DirectX runtime เวอร์ชัน 7.0 หรือสูงกว่า ถ้าเครื่องคอมพิวเตอร์ยังไม่ได้ติดตั้ง DirectX runtime โปรแกรมจะไม่สามารถทำงานได้

3. ทดสอบโปรแกรมเพื่อดูผลการทำงานของโปรแกรมเกมท่องเที่ยวภาคเหนือว่ามีขั้นตอนการดำเนินเรื่องตามที่กำหนดไว้หรือไม่

4. ตรวจสอบผลของการทดสอบโปรแกรมว่ามีข้อผิดพลาดหรือไม่

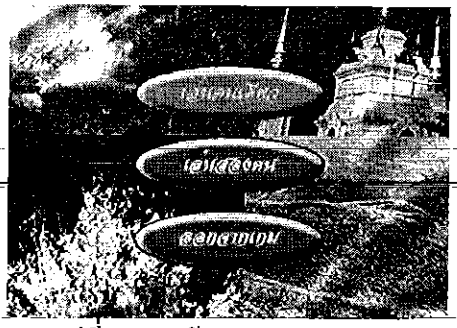
5. นำข้อผิดพลาดที่เกิดขึ้นมาปรับปรุงแก้ไข

4.3 ผลการทดสอบโปรแกรม

การรับค่า Input จากผู้เล่น ผู้เล่นจะมีการใช้ทั้ง Mouse และ Keyboard โดยใช้ดังนี้

Space Bar	-	สำหรับเข้าไปยังสถานที่ต่างๆ และใช้พูดกับตัวละครตัวอื่น
Ctrl	-	สำหรับเปิดสมุดบันทึกเพื่อดูว่าควรทำอะไรบ้าง
Mouse	-	สำหรับเลือกคำตอบในการตอบคำถามที่บ้านแห่งความรู้ และในส่วนของหน้าเมนู
Arrow Key	-	ทั้ง 4 ปุ่ม คือ ซ้าย ขวา หน้า หลัง สำหรับการเดินของผู้เล่น
ตัวเลขต่างๆ ใต้ NumLock เช่น 1,2,3,4,5,6,7		สำหรับเลือกคำตอบ

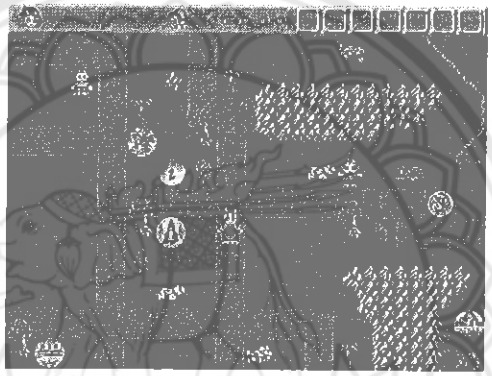
เมื่อเริ่มเรียกใช้โปรแกรมจะปรากฏหน้าแรก เป็นหน้า Menu โดยจะมีปุ่มให้เลือก 3 ปุ่ม คือ ปุ่มเล่นคนเดียว ปุ่มเล่นสองคน และ ปุ่มออกจากเกม



รูปที่ 4.1 หน้าแรกของโปรแกรม

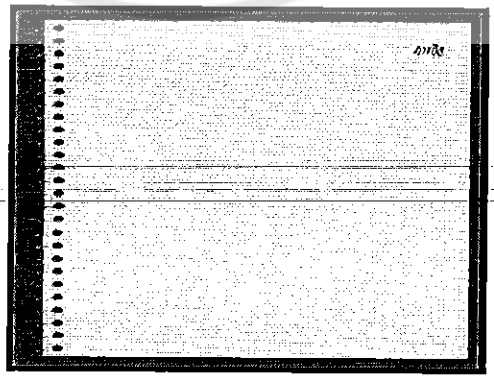
4.3.2 เมื่อเลือกเล่นเกมแบบผู้เล่นคนเดียว

เมื่อกดปุ่มเล่นคนเดียวจะเข้าสู่เกมและเริ่มเล่นเกมได้ ดังรูปที่ 4.2



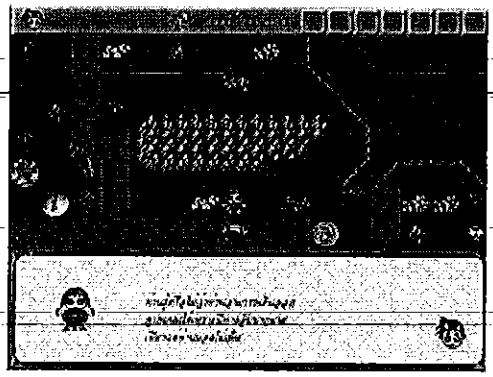
รูปที่ 4.2 การเข้าสู่เกมในตอนเริ่มต้น

เมื่อเริ่มต้นเกม ผู้เล่นจะอยู่บริเวณจังหวัดพิษณุโลก มีเงิน 5000 บาท ยังไม่มีสิ่งของ(Item) และภารกิจว่างเปล่า (กด Ctrl เพื่อดูสมุดบันทึก)

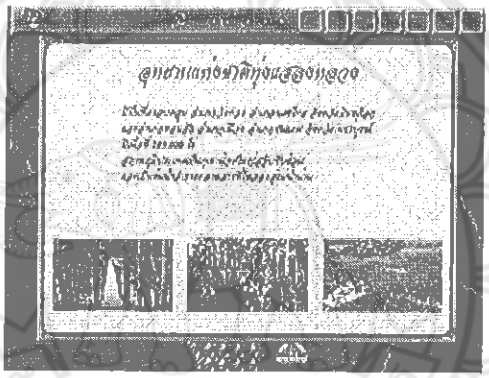


รูปที่ 4.3 สมุดบันทึก

จากนั้นทดลองเดินไปพูดคุยกับตัวละครและเข้าไปยังสถานที่ท่องเที่ยวต่างๆ

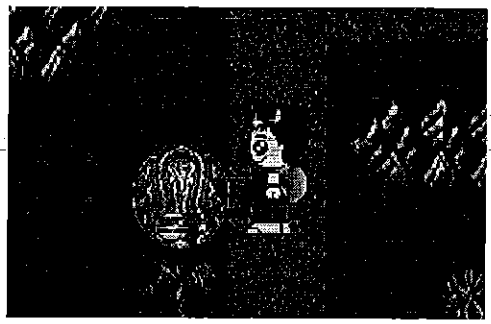


รูปที่ 4.4 พูดคุยกับตัวละครอื่น



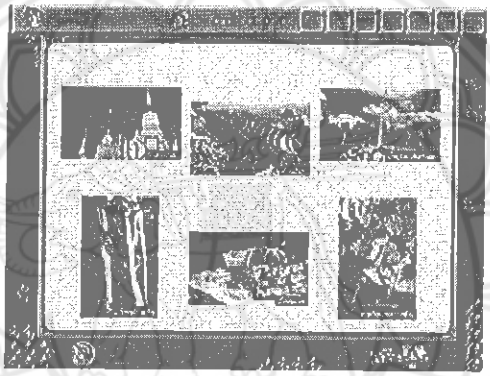
รูปที่ 4.5 เข้าไปยังสถานที่ท่องเที่ยวต่างๆ

ทดลองเข้าไปสำรวจในจังหวัดต่างๆ เพื่อหาข้อมูลเกี่ยวกับจังหวัด



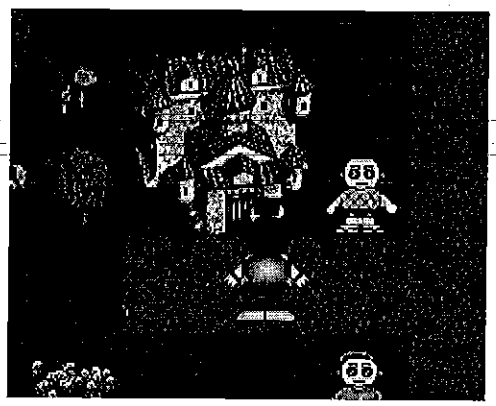
รูปที่ 4.6 ตัวผู้เล่นกำลังเดินเข้าตัวจังหวัด

รูปที่ 4.7 หน้าแรกของจังหวัด



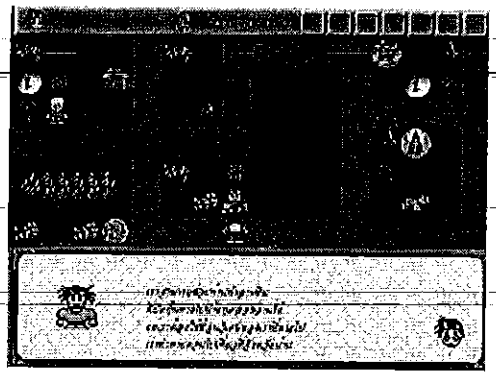
รูปที่ 4.8 หน้าที่ 2 ของจังหวัด

เมื่อสามารถเข้าพูดคุยกับตัวละครและเข้าไปยังสถานที่ต่างๆได้แล้ว จะทดลองเข้าไปที่บ้านแห่งความรู้เพื่อตอบคำถาม แต่ยังภารกิจไม่เสร็จจะไม่สามารถเข้าสู่การตอบคำถามได้



รูปที่ 4.9 ตัวผู้เล่นพยายามเข้าบ้านแห่งความรู้

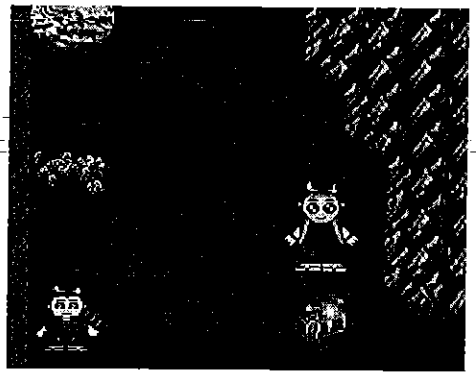
ทดลองไปพูดคุยกับตัวละครเพื่อรับภารกิจ โดยจะมีภารกิจทั้งหมด 4 ภารกิจ



รูปที่ 4.10 ตัวผู้เล่นได้รับภารกิจจากตัวละคร

รูปที่ 4.11 สมุดบันทึกเมื่อได้รับภารกิจครบ 4 ภารกิจ

ทดลองแก้ไขปัญหาเพื่อให้ภารกิจต่างๆเสร็จสิ้น

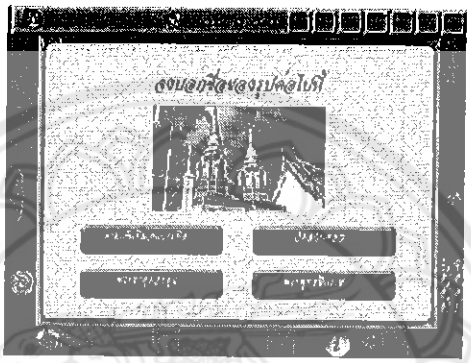


รูปที่ 4.12 ผู้เล่นพยายามทำภารกิจ

เมื่อทำภารกิจเสร็จสมบูรณ์แล้ว ทดลองเข้าไปยังบ้านแห่งความรู้เพื่อตอบคำถามโดยในการตอบคำถาม จะใช้ Mouse เลือกคำตอบที่ถูกต้อง (จะใช้ Mouse เฉพาะตอนเลือกคำตอบเท่านั้น) คำถามและคำตอบจะเปลี่ยนตลอดทุกข้อโดยการ Random ถ้าตอบถูก คำถามต่อไปจะขึ้นมาแทน แต่ถ้าตอบผิด ผู้เล่นจะแพ้(Game Over)

คำถามจะมีทั้งหมด 16 ข้อ โดยแบ่งเป็น 4 ช่วง ดังนี้

คำถามข้อ 1 ถึง 4 เป็นการเลือกชื่อให้ตรงกับรูป



รูปที่ 4.13 คำถามข้อที่ 1 ถึง 4

คำถามข้อ 5 ถึง 8 เป็นคำถามเกี่ยวกับตราจังหวัด

รูปที่ 4.14 คำถามข้อที่ 5 ถึง 8

คำถามข้อ 9 ถึง 12 ถามเกี่ยวกับสถานที่ท่องเที่ยวในจังหวัดต่างๆ

รูปที่ 4.15 คำถามข้อที่ 9 ถึง 12

คำถามข้อ 13 ถึง 16 เป็นคำถามเกี่ยวกับคำขวัญประจำจังหวัด

รูปที่ 4.16 คำถามข้อที่ 13 ถึง 16

เมื่อตอบคำถามได้จนครบทั้ง 16 ข้อแล้ว ผู้เล่นจะชนะและออกจากโปรแกรม

4.3.2 เมื่อเลือกเล่นเกมแบบผู้เล่น 2 คน

เมื่อกดปุ่มเล่น 2 คนจากเมนูจะปรากฏหน้าต่าง Make Connection ซึ่งเป็นการสร้างการติดต่อระหว่าง 2 เครื่อง โดยมีการกรอกค่าต่างๆดังนี้

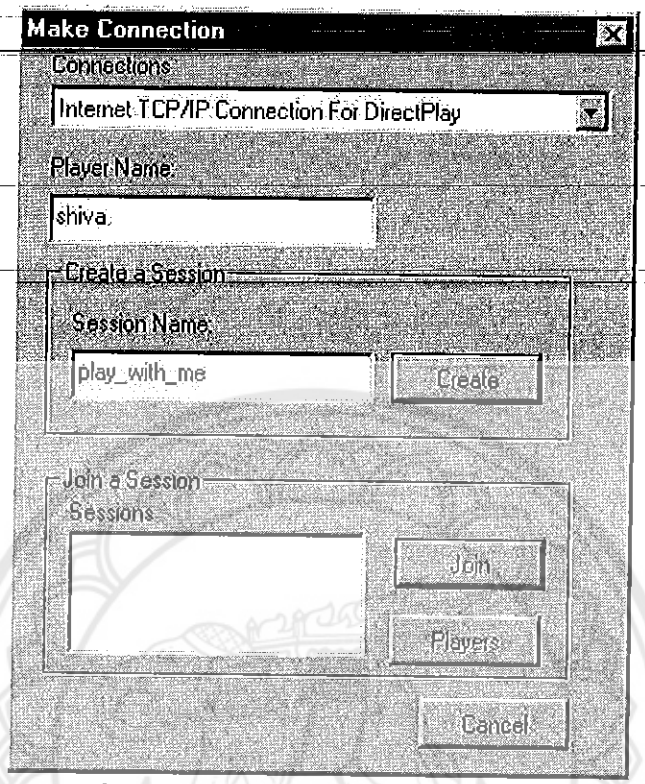
-Connections เป็นการเลือกรูปแบบการติดต่อว่าจะใช้โปรโตคอลใดในการติดต่อ

-Player Name เป็นการใส่ชื่อของผู้เล่น

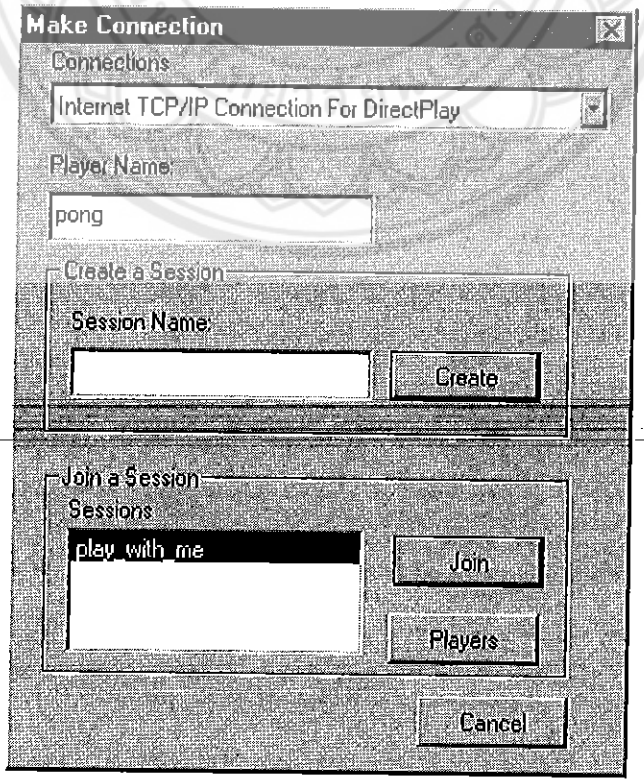
-Create a Session เป็นการตั้งชื่อ Session และทำการสร้างด้วยการกดปุ่ม Create โดยฝ่าย Host จะเป็นผู้สร้าง Session จากนั้นจะเป็นฝ่ายรอการConnect จาก Guest

-Join Session จะแสดงรายชื่อ Session ที่ถูกสร้างขึ้น เมื่อ Guest เห็น Session ที่ถูกสร้างขึ้นจะสามารถเข้ามา Join ได้โดยกดปุ่ม Join

เมื่อทั้ง 2 เครื่องสามารถติดต่อกันได้แล้วจะสามารถเข้ามาเล่นด้วยกันได้โดยจะเป็นส่วนของการตอบคำถามเกี่ยวกับสถานที่ท่องเที่ยวและจังหวัดต่างๆในภาคเหนือ



รูปที่ 4.17 หน้าต่าง Make Connection ของฝ่าย Host



รูปที่ 4.18 หน้าต่าง Make Connection ของฝ่าย Guest

4.2 วิเคราะห์ผล

จากผลการทดสอบโปรแกรมปรากฏว่า เกมได้ดำเนินเรื่องไปตามที่กำหนดไว้ คือ เริ่มจากการที่ผู้เล่นต้องออกเดินทางไปตามสถานที่ท่องเที่ยวต่างๆและพูดคุยกับตัวละคร เพื่อหาข้อมูลและรับภารกิจต่างๆมา ถ้าผู้เล่นยังปฏิบัติภารกิจไม่เสร็จทั้ง 4 ภารกิจ จะไม่สามารถเข้าไปยังบ้านแห่งความรู้เพื่อตอบคำถามได้ ผู้เล่นจึงต้องออกเดินทางเพื่อหาข้อมูลหรือสิ่งของเพื่อแก้ไขภารกิจที่ได้รับมาให้เสร็จสิ้น

เมื่อผู้เล่นปฏิบัติภารกิจเสร็จสิ้นทั้งหมดแล้วจะสามารถเข้าไปตอบคำถามที่บ้านแห่งความรู้ได้ คำถามและคำตอบจะเปลี่ยนตลอดทุกข้อโดยการสุ่ม ถ้าตอบถูก คำถามต่อไปจะขึ้นมาแทน แต่ถ้าตอบผิด ผู้เล่นจะแพ้(Game Over) และจบโปรแกรม โดยคำถามจะมีทั้งหมด 16 ข้อ และแบ่งเป็น 4 ช่วงตามที่ได้กำหนดไว้ เมื่อผู้เล่นตอบคำถามได้หมดทุกข้อแล้ว ผู้เล่นจะเป็นฝ่ายชนะและจบโปรแกรม

ในส่วนของการเล่นแบบ 2 ผู้เล่น ต้องมีการสร้างการติดต่อก่อน โดยจะมีการเลือกโปรโตคอลและสร้าง Session ขึ้น หลังจากสร้าง Session เสร็จแล้ว จะมีการรออีกเครื่องหนึ่ง Join เข้ามา จากนั้นจึงสามารถเข้ามาเล่นด้วยกันได้ โดยจะเป็นส่วนของการตอบคำถามเกี่ยวกับสถานที่ท่องเที่ยวและจังหวัดต่างๆในภาคเหนือ

บทที่ 5
สรุปผลและข้อเสนอแนะ

ในบทนี้เป็นการสรุปผลทั้งหมดของการทำโครงงานนี้ ซึ่งประกอบด้วยส่วนสรุปผล ปัญหาในการทำงาน ข้อเสนอแนะ และแนวทางในการพัฒนาต่อไป หากมีผู้ที่สนใจที่จะนำโครงงานนี้ไปปรับปรุงและพัฒนาต่อไป

5.1 สรุปผล

1. โปรแกรมที่พัฒนาเป็นโปรแกรมเกมท่องเที่ยวภาคเหนือ โดยลักษณะของเกมเป็นแบบ RPG และ Adventure ซึ่งพัฒนาโดยใช้โปรแกรม Visual C++ 6.0 และได้นำความสามารถของ DirectX มาใช้ในการจัดการเกี่ยวกับการแสดงผลออกทางจอภาพ ระบบเสียงประกอบขณะเล่นเกม การเล่นแบบหลายผู้เล่น การตกแต่งภาพใช้ Adobe Photoshop 5.5 การอัดและแต่งเสียงใช้โปรแกรม Cool Edit ProV1.2a

2. โปรแกรมที่พัฒนาแบ่งออกเป็น 2 โปรแกรม คือ แบบเล่นคนเดียว และแบบเล่น 2 คนโดยแบบเล่น 2 คน จะเป็นส่วนของการตอบคำถาม ในการใช้งานโปรแกรมเกมท่องเที่ยวภาคเหนือ ต้องมีการติดตั้ง DirectX runtime ด้วย จึงจะสามารถใช้งานได้

5.2 ปัญหาในการทำงาน

1. การพัฒนาโปรแกรมจะใช้โปรแกรม Visual C ++ 6.0 ในการเรียกใช้ความสามารถของ DirectX ซึ่งโปรแกรม Visual C ++ 6.0 เป็นโปรแกรมที่ผู้จัดทำไม่คุ้นเคยในการใช้งานมาก่อน ทำให้ต้องใช้เวลาในการศึกษาการใช้งานค่อนข้างมาก มีผลทำให้การพัฒนาโปรแกรมเป็นไปอย่างล่าช้า

2. การเตรียมทรัพยากรในการนำมาใช้ในการสร้างเกมใช้เวลาค่อนข้างมาก เพราะส่วนใหญ่จะเป็นภาพกราฟิก ซึ่งผู้จัดทำต้องวาดขึ้นมาเอง เช่น ตัวละคร ตัวผู้เล่น เป็นต้น

3. การออกแบบและสร้างแผนที่ใช้เวลาค่อนข้างมากเนื่องจากแผนที่ค่อนข้างใหญ่ และมีการอ้างอิงจากแผนที่จริง

5.3 ข้อเสนอแนะ

1. ควรดำเนินงานแต่ละขั้นตอนให้เสร็จก่อนเวลากำหนด เพราะบางขั้นตอนอาจต้องใช้เวลามากกว่าที่เราได้กำหนดไว้ซึ่งจะทำให้เกิดความล่าช้าในการทำโครงงาน

2. ก่อนการดำเนินงานในแต่ละขั้นตอนควรมีการศึกษาข้อมูลให้ถ่องแท้เสียก่อนเพื่อไม่ให้เกิดข้อผิดพลาดขึ้นในการทำงาน

5.4 แนวทางในการพัฒนา

1. เพิ่มแผนที่ในส่วนของภาคอื่นๆที่เหลือในประเทศไทยเนื่องจากผู้จัดทำได้พัฒนาในเฉพาะส่วนของภาคเหนือ ซึ่งการเพิ่มแผนที่ภาคอื่นๆจะทำให้เกมมีความหลากหลายและมีความสนุกสนานมากขึ้น

2. เพิ่มตัวละคร สิ่งของ หรือเพิ่มภารกิจต่างๆ จะทำให้เกมสนุกยิ่งขึ้น

3. เพิ่มข้อมูลเกี่ยวกับสถานที่ท่องเที่ยวหรือข้อมูลเกี่ยวกับจังหวัดต่างๆ จะทำให้ผู้เล่นได้รับความรู้เกี่ยวกับสถานที่ท่องเที่ยวและข้อมูลเกี่ยวกับจังหวัดต่างๆมากขึ้น

4. ปรับปรุงเปลี่ยนแปลงเนื้อเรื่องหรือใช้โปรแกรมท่องเที่ยวภาคเหนือเป็นแนวทางในการพัฒนาเป็นเกมอื่นๆต่อไป

5. พัฒนาในส่วนของเกมแบบหลายผู้เล่นให้เล่นในรูปแบบอื่นๆนอกจากการตอบคำถาม

เอกสารอ้างอิง

[1] Bradley Bargen. and Peter Donnelly., **Inside DirectX**, Redmond: Microsoft Press, 1998.

[2] Julio Sanchez. and Maria P. Canton., **DirectX 3d Graphics Programming Bible**, IDG Books Worldwide, Inc., 2000.

ภาคผนวก ก

ปุ่มควบคุมภายในเกม

การรับค่า Input จากผู้เล่น ผู้เล่นจะมีการใช้ทั้ง Mouse และ Keyboard โดยใช้ดังนี้

Space Bar - สำหรับเข้าไปยังสถานที่ต่างๆ และใช้พูดกับตัวละครตัวอื่น

Ctrl - สำหรับเปิดสมุดบันทึกเพื่อดูว่าควรทำอะไรบ้าง

ตัวเลขต่างๆ ใต้ NumLock เช่น 1,2,3,4,5,6,7 สำหรับเลือกคำตอบ

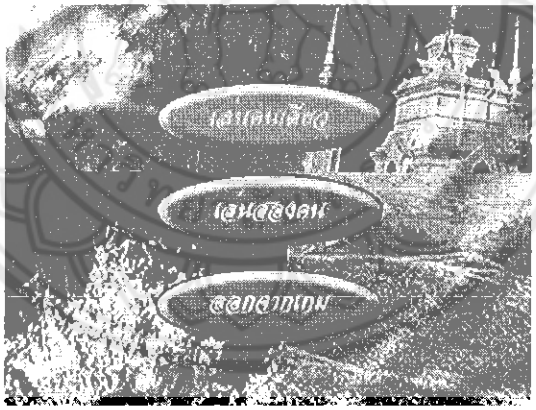
Mouse - สำหรับเลือกคำตอบในการตอบคำถามที่บ้านแห่งความรู้ และในส่วนของหน้าเมนู

Arrow Key - ทั้ง 4 ปุ่ม คือ ซ้าย ขวา หน้า หลัง สำหรับการเดินของผู้เล่น

เรียกโปรแกรม Thai Touring Game จาก Start Up Menu หรือจากใน Folder ที่ได้ทำการ ติดตั้งไว้จะได้หน้าแรกของ Menu ดังรูป

วิธีการเล่นเกม

ในส่วนของเมนูเกมดังรูปที่ ก.1



รูปที่ ก.1 ส่วนของเมนูเกม

ความหมายของปุ่ม

เมื่อต้องการเข้าสู่เกมแบบผู้เล่นคนเดียว

เมื่อต้องการเข้าสู่เกมแบบผู้เล่นสองคน

เมื่อต้องการออกจากเกม

เกมแบบผู้เล่นคนเดียว

เมื่อผู้เล่นเลือกเล่นเกมแบบผู้เล่นคนเดียว หน้าถัดมาจะเป็นดังรูปที่ ก.2

รูปที่ ก.2 เมนูเริ่มเล่น

ความหมายของปุ่ม

เมื่อต้องการเข้าสู่เกม

เมื่อต้องการกลับไปที่เมนูหลัก

เมื่อผู้เล่นเริ่มเล่นเกม ในตอนเริ่มต้นผู้เล่นจะอยู่บริเวณจังหวัดพิษณุโลก มีเงิน 5000 บาท ยังไม่มีสิ่งของ(Item) สิ่งที่ผู้เล่นต้องทำให้สำเร็จคือ การทำภารกิจต่างๆ ที่ได้รับมอบหมายจากตัวละครที่อยู่ในแผนที่ เมื่อภารกิจเสร็จแล้วผู้เล่นจะต้องไปตอบคำถามที่บ้านแห่งความรู้ซึ่งเป็นจุดสุดท้ายของเกม

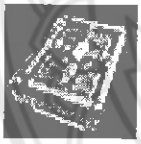
รูปต่างๆบนแผนที่มีความหมายดังนี้

เป็นตราประจำจังหวัดต่าง ๆ แทนตัวเมืองของจังหวัดนั้น

ตัว i (information) คือ จุดแสดงแผนที่ บอกตำแหน่งของจังหวัดและสถานที่ท่องเที่ยวต่างๆซึ่งจะช่วยทำให้ผู้เล่นไม่หลงทาง

ตัวละครต่างต่างๆที่อยู่ตามตำแหน่งต่างๆในแผนที่ มีทั้งหมด 16 ตัว โดยแต่ละตัวจะให้ข้อมูลหรือสิ่งของต่างๆ กับผู้เล่นแต่ผู้เล่นจะต้องแก้ปัญหาให้ตัวละครนั้นก่อน ตัวละครบางตัวอาจไม่มีความสำคัญ แต่บางตัวเป็นตัวสำคัญในเกม ซึ่งตัวละครบางตัวอาจเกี่ยวโยงกัน

เป็นสัญลักษณ์ของสถานที่ท่องเที่ยวต่างๆ ซึ่งผู้เล่นสามารถเข้าไปดูได้โดยกด Space Bar ภายในจะเป็นข้อมูลของสถานที่ท่องเที่ยวนั้นๆ



รูป Item ต่างๆ ซึ่งเราสามารถเก็บได้ โดยจะแสดงรูป Item ที่เก็บได้ ในช่องใส่ของ ซึ่งจะอยู่บน Status Bar

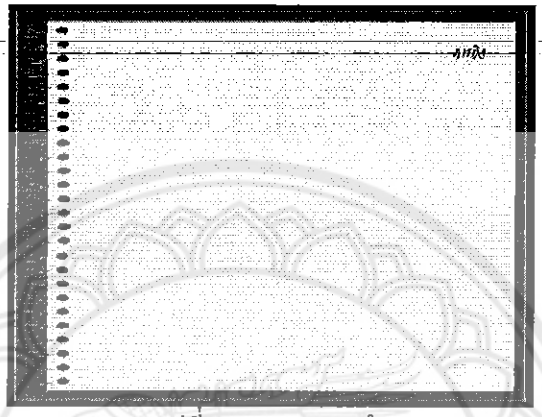
รูปบ้านแห่งความรู้ ซึ่งเป็นจุดสุดท้ายที่ผู้เล่นจะต้องเข้าไปและเป็นจุดตัดสินการแพ้ชนะของเกม แต่การจะเข้าไปในบ้านแห่งความรู้ได้ ผู้เล่นจะต้องทำภารกิจต่างๆให้เสร็จสิ้นก่อน

แถบแสดงสถานะของผู้เล่น (Status Bar)

ประกอบด้วยจำนวนเงินที่มีอยู่ของผู้เล่น และช่องใส่สิ่งของ(Item)

สมุดบันทึก

ผู้เล่นจะมีสมุดบันทึกประจำตัวอยู่ ซึ่งสมุดบันทึกนี้จะบอกภารกิจที่ผู้เล่นจะต้องทำโดยในตอนแรก สมุดบันทึกจะว่างเปล่า ผู้เล่นจะได้ข้อความบอกว่าควรทำอะไรหรือภารกิจ เมื่อผู้เล่นพูดคุยกับตัวละครที่ต้องการให้ผู้เล่นช่วยเหลือ ผู้เล่นสามารถเปิดสมุดบันทึกดูได้โดยกด Ctrl



รูปที่ ก.3 สมุดบันทึก

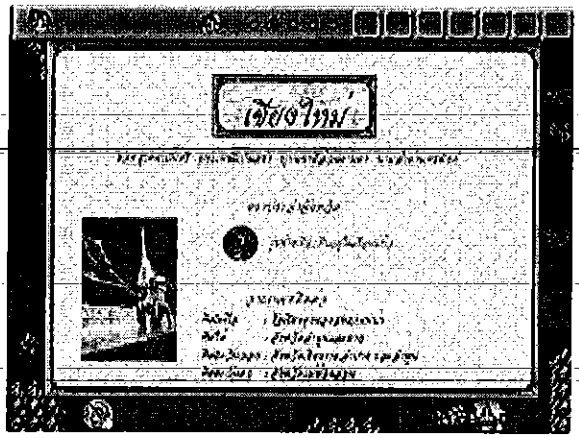
การพูดคุยกับตัวละครอื่นภายในแผนที่

ผู้เล่นสามารถพูดคุยกับตัวละครอื่นในแผนที่ได้โดยหันหน้าเข้าหาตัวละครแล้วกด Space Bar จะปรากฏคำพูดของคนนั้น ดังรูป ถ้าต้องการออกให้กด Space Bar อีกครั้ง

รูปที่ ก.4 ผู้เล่นพูดคุยกับตัวละคร

การเข้าไปในตัวจังหวัด

ข้อมูลของแต่ละจังหวัดจะมีทั้งหมด 2หน้า หน้าแรกได้แก่ คำขวัญประจำจังหวัด ตราประจำจังหวัด อาณาเขตติดต่อ หน้าที่ 2 คือ สถานที่ท่องเที่ยวที่น่าสนใจ



รูปที่ ก.5 ข้อมูลของจังหวัด

การเข้าไปตอบคำถามในบ้านแห่งความรู้

ผู้เล่นจะเข้าไปตอบได้ก็ต่อเมื่อ ได้ทำภารกิจเสร็จสมบูรณ์แล้ว การตอบคำถามจะใช้ mouse ให้ผู้เล่นเลือกคำตอบที่ถูกต้อง

คำถามจะมีทั้งหมด 16 ข้อ โดยแบ่งเป็น 4 ช่วง ดังนี้

คำถามที่ 1 ถึง 4 เป็นการถามว่าภาพที่ให้มาเป็นภาพชื่ออะไร

คำถามที่ 5 ถึง 8 เป็นคำถามเกี่ยวกับตราประจำจังหวัด

คำถามที่ 9 ถึง 12 เป็นการถามว่าภาพที่ให้มาอยู่ในจังหวัดใด

คำถามที่ 13 ถึง 16 เป็นคำถามเกี่ยวกับคำขวัญประจำจังหวัด

เมื่อตอบคำถามถึงข้อ 16 แล้ว จะถือว่าภารกิจเสร็จสิ้นสมบูรณ์ ผู้เล่นจะชนะเกม

เกมแบบผู้เล่น 2 คน

เมื่อผู้เล่นเลือกเล่นเกมแบบผู้เล่น 2 คนจะปรากฏหน้าต่าง Make Connection ซึ่งเป็นการสร้างการติดต่อระหว่าง 2 เครื่อง โดยมีการกรอกค่าต่างๆดังนี้

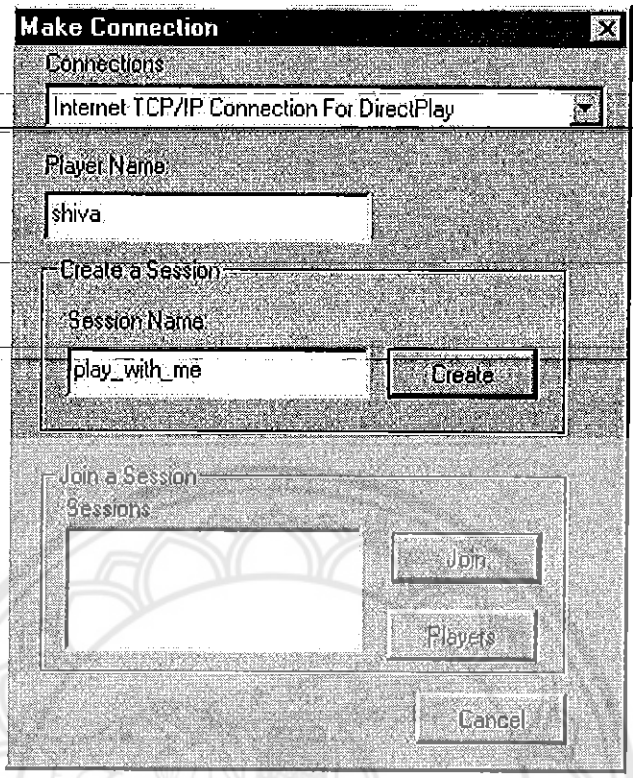
-Connections เป็นการเลือกรูปแบบการติดต่อว่าจะใช้โปรโตคอลใดในการติดต่อ

-Player Name เป็นการใส่ชื่อของผู้เล่น

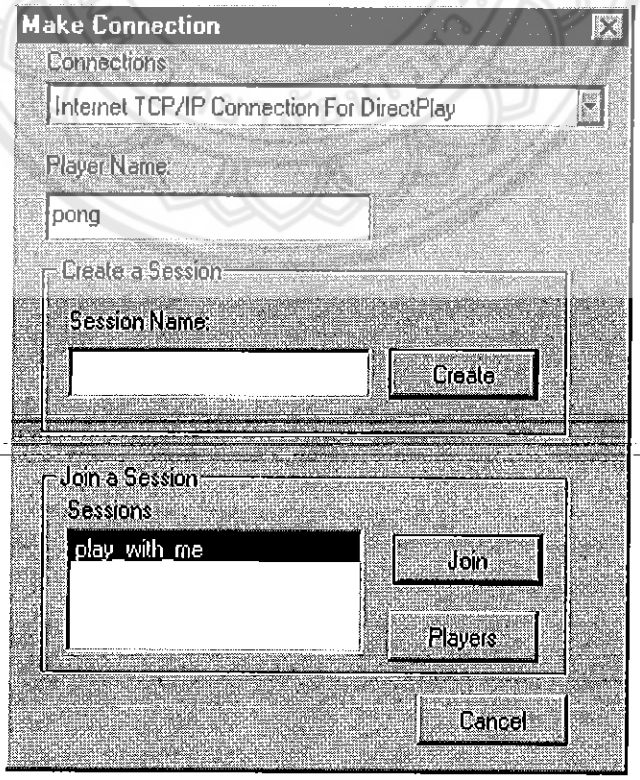
-Create a Session เป็นการตั้งชื่อ Session และทำการสร้างด้วยการกดปุ่ม Create โดยฝ่าย Host จะเป็นผู้สร้าง Session จากนั้นจะเป็นฝ่ายรอการConnect จาก Guest

-Join Session จะแสดงรายชื่อ Session ที่ถูกสร้างขึ้น เมื่อ Guest เห็น Session ที่ถูกสร้างขึ้นจะสามารถเข้ามา Join ได้โดยกดปุ่ม Join

เมื่อทั้ง 2 เครื่องสามารถติดต่อกันได้แล้วจะสามารถเข้ามาเล่นด้วยกันได้โดยจะเป็นส่วนของการตอบคำถามเกี่ยวกับสถานที่ท่องเที่ยวและจังหวัดต่างๆในภาคเหนือ



รูปที่ ก.6 หน้าต่าง Make Connection ของฝ่าย Host



รูปที่ ก.7 หน้าต่าง Make Connection ของฝ่าย Guest

ประวัติผู้เขียน

นายกชพงศ์ เพ็ชรราช

วันเดือนปีเกิด	1 กันยายน 2521
ที่อยู่	27/98 สถาบันราชภัฏอุตรดิตถ์ ต.ท่าอิฐ อ.เมือง จ.อุตรดิตถ์ 53000
โทรศัพท์	01-5339850
E-mail	kochapong27@hotmail.com
ประวัติการศึกษา	
ประถม	โรงเรียนเทศบาลท่าอิฐ
มัธยมต้น	โรงเรียนอุตรดิตถ์
มัธยมปลาย	โรงเรียนอุตรดิตถ์

นางสาวอรุณี นาคผสม

วันเดือนปีเกิด	7 พฤษภาคม 2523
ที่อยู่	46 หมู่ 3 ต.หัวรอ อ.เมือง จ.พิษณุโลก 65000
โทรศัพท์	055-214181
E-mail	ar_pong99@hotmail.com
ประวัติการศึกษา	
ประถม	โรงเรียนอนุบาลพิษณุโลก
มัธยมต้น	โรงเรียนเฉลิมขวัญสตรี
มัธยมปลาย	โรงเรียนเฉลิมขวัญสตรี