การพัฒนาเกมสามมิติด้วยไดเรคเอกซ์

3D GAME DEVELOPMENT BY DIRECTX

นายยุทธพงษ์ หาญพยอม รหัส 45360344

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
ปีการศึกษา 2548

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน	การพัฒนาเกมสามมิติด้วยไดเรคเอกซ์
ผู้ดำเนินโครงงาน	นายยุทธพงษ์ หาญพยอม รหัส 45360344
อาจารย์ที่ปรึกษา	ดร.พนมขวัญ ริยะมงคล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2548

.....

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร อนุมัติให้โครงงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะกรรมการสอบโครงงานวิศวกรรม

.....ประธานกรรมการ
(ดร.พนมขวัญ ริยะมงคล)

.....กรรมการ
(ดร.สุรเชษฐ์ กานต์ประชา)

.....กรรมการ
(ดร.สมยศ เกียรติวนิชวิไล)

หัวข้อโครงงาน	การพัฒนาเกมสามมิติด้วยไดเรคเอกซ์
ผู้ดำเนินโครงงาน	นายยุทธพงษ์ หาญพยอม รหัส 45360344
อาจารย์ที่ปรึกษา	ดร. พนมขวัญ ริยะมงคล
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์
ปีการศึกษา	2548

.....

บทคัดย่อ

โครงงานนี้เป็นการศึกษาการพัฒนาโปรแกรมเกมสามมิติซึ่งทำงานบนระบบปฏิบัติการวินโดว์ โดยใช้โปรแกรม Microsoft Visual C++ และ DirectX เป็นเครื่องมือในการพัฒนา ซึ่งจะช่วยในการจัดการเกี่ยวกับ การแสดงผลภาพ 2 มิติ และ 3 มิติ ระบบเสียงประกอบการเล่นเกม และระบบการรับอินพุต จากผู้เล่น เช่น คีย์บอร์ด เมาส์ และจอยสติก เป็นต้น นอกจากนี้ยังมีการใช้โปรแกรม 3DS MAX7 Adobe Photoshop 7 และ Cool Edit เพื่อช่วยในการออกแบบตัวละคร การสร้างวัตถุสามมิติต่างๆ การตกแต่งภาพ การสร้างเสียงและตกแต่งเสียงเพื่อใช้สำหรับเกม เกมที่พัฒนาขึ้นเป็นเกมสามมิติที่ผู้เล่นจะต้องทำภารกิจตามที่กำหนดไว้ให้สำเร็จก่อนหมดเวลา ถ้าหากผู้เล่นทำภารกิจไม่สำเร็จจะเป็นผู้แพ้ แต่ถ้าผู้เล่นทำภารกิจสำเร็จก็จะเป็นผู้ชนะ

Project Title	3D Game Development By DirectX
Name	Mr. Yutthaphong Hanphayom ID. 45360344
Project Advisor	Dr. Panomkhawn Riyamongkol
Major	Computer Engineering
Department	Electronic and Computer Enigineering
Academic Year	2005

.....

ABSTRACT

This project is studying and developing a 3D game for a window operating system. Microsoft Visual C++ and DirectX are used to develop this game. They have functions to manage displaying system of images in 2 dimension and 3 dimension, sound system during playing game and input system from keyboard, mouse, and joystick. Moreover, 3DS MAX7 program, Adobe Photoshop 7 program and Cool Edit program are used to design character model, create 3D objects, design texture, create sound, and edit sound for the game. The developed game is a 3D game, which a player has to fulfill a mission before time out.

กิตติกรรมประกาศ

ขอขอบพระคุณคุณพ่อและคุณแม่ อาจารย์ทุกท่าน เพื่อนๆ ทุกคนที่คอยให้กำลังใจและคอยให้คำปรึกษาต่างๆ จนทำให้โครงงานนี้เสร็จสมบูรณ์ได้ โดยเฉพาะ ดร.พนมขวัญ ริยะมงคล และ อ.พงศ์พันธ์ กิจสนาโยธิน ที่คอยให้คำปรึกษาและคำแนะนำต่างๆในการทำโครงงานนี้จนทำให้โครงงานนี้เสร็จสมบูรณ์ได้

ยุทธพงษ์ หาญพยอม

สารบัญ

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ในปัจจุบันนี้คอมพิวเตอร์ได้เข้ามามีบทบาทในชีวิตประจำวันมากขึ้น ซึ่งก็ได้มีการเอาคอมพิวเตอร์มาช่วยงานในด้านต่างๆ มากมาย เช่น งานด้านการโทรคมนาคม งานด้านมัลติมีเดีย จึงทำให้เกิดซอฟต์แวร์ต่างๆมากมายเข้ามาช่วยในการทำงาน

ซอฟต์แวร์ด้านมัลติมีเดียซึ่งได้รับความนิยมมากไม่น้อยกว่าซอฟต์แวร์ด้านอื่นๆ เช่น เกม มีทั้งเกม 2 มิติ และ 3มิติ ซึ่งประเทศไทยของเราขาดนักพัฒนาซอฟต์แวร์เกมเป็นอย่างมาก มีแต่นักเล่นเกมเป็นส่วนใหญ่ ผู้จัดทำได้มองเห็นความสำคัญของปัญหาตรงนี้ จึงได้ศึกษาการเขียนซอฟต์แวร์เกม 3 มิติ เพื่อเป็นแนวทางสำหรับผู้ที่คิดจะศึกษาการพัฒนาซอฟต์แวร์เกม เครื่องมือที่ใช้ก็คือ DirectX ซึ่งกำลังได้รับความนิยมเป็นอย่างมากในปัจจุบัน

1.2 วัตถุประสงค์ของโครงงาน

1. เพื่อเป็นการเพิ่มพูนทักษะของผู้พัฒนาโปรแกรมในการเขียนโปรแกรม
2. เพื่อให้มีความรู้ความเข้าใจในการพัฒนาเกม 3 มิติ ด้วย DirectX SDK และ Microsoft Visual C++
3. เพื่อให้มีความรู้ในด้านการสร้างภาพ 3 มิติ
4. เพื่อให้มีความรู้ในด้านคอมพิวเตอร์กราฟิก

1.3 ขอบข่ายของโครงงาน

1. พัฒนาโปรแกรมให้แสดงโมเดลในรูปแบบ 3 มิติได้
2. พัฒนาโปรแกรมให้สามารถเคลื่อนที่ไปในโมเดล 3 มิติได้

1.4 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินโครงการ

กิจกรรม	เดือน - ปี					
	พ.ย. 47	ธ.ค. 47	ม.ค. 48	ก.พ. 48	มี.ค. 48	เม.ย. 48
1. รวบรวมข้อมูล	←	→				
2. จัดหาเครื่องมือ		←	→			
3. ศึกษาทฤษฎีที่ใช้ในโครงงาน			←	→		
4. ออกแบบเกม			←	→		
5. เขียนโปรแกรม			←	→		
6. ทดสอบและแก้ไข					←	→
7. สรุปผล					←	→
8. จัดทำคู่มือ						← →

1.5 ผลที่คาดว่าจะได้รับ

1. มีทักษะในการพัฒนาโปรแกรมมากขึ้น
2. มีความรู้ในด้านการเขียนโปรแกรม ด้วย DirectX SDK และ Microsoft visual C++ มากขึ้น
3. มีความรู้ความเข้าใจเกี่ยวกับการสร้างภาพ 3 มิติ มากขึ้น
4. มีความรู้ความเข้าใจเกี่ยวกับคอมพิวเตอร์กราฟิกมากขึ้น
5. ได้ซอฟต์แวร์เกม 3 มิติ ที่ใช้งานบน PC ได้

1.6 งบประมาณที่ใช้

1. ค่าจัดซื้อหนังสือ	1,500	บาท
2. ค่าพิมพ์เอกสารและถ่ายเอกสาร	500	บาท
3. ค่าวัสดุคอมพิวเตอร์	500	บาท
รวม	2,500	บาท

บทที่ 2
หลักการและทฤษฎี

ในบทที่ 2 นี้จะเป็นการกล่าวถึงทฤษฎีต่างๆ ที่ได้นำมาใช้ในการพัฒนาโปรแกรม เกม 3มิติ ซึ่งได้แก่ทฤษฎีการใช้งาน DirectX โดยเฉพาะ Direct3D ทฤษฎีการใช้งาน Microsoft Visual C++ ในการสร้าง หน้าต่างวินโดว์

ขั้นตอนการศึกษาทฤษฎี

1. ศึกษาการสร้างหน้าต่างวินโดว์ ด้วย Microsoft Visual C++ version 6.0
2. ศึกษาโครงสร้างและทฤษฎีการใช้ DirectX

2.1 การเขียนโปรแกรมด้วย Microsoft Visual C++

Visual C++ สร้างขึ้นโดยบริษัทไมโครซอฟต์ (Microsoft) เพื่อใช้ในการพัฒนาโปรแกรมบนวินโดว์ ซึ่งมีรูปแบบการพัฒนาเป็นแบบ Visual คือ ออกแบบลักษณะหน้าตาของโปรแกรมได้ง่าย มี MFC (Microsoft Foundation Class) ซึ่งเป็น Class Library ให้ใช้งานซึ่งจะทำให้การพัฒนาโปรแกรมได้ง่ายยิ่งขึ้น

ฟังก์ชันหลักๆ ที่ใช้ในการสร้างหน้าต่างวินโดว์มีดังนี้

- WinMain : จะต้อง include <windows.h>
- WindowProc :จะต้อง include <windows.h>

2.1.1 ฟังก์ชัน WinMain ฟังก์ชันนี้ เป็นฟังก์ชันหลักของโปรแกรมทั้งหมด เมื่อโปรแกรมทำงานฟังก์ชันนี้จะถูกเรียกเป็นอันดับแรก

รูปแบบฟังก์ชัน WinMain ()

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPreinstance, LPSTR lPcmdline, int nCmdshow)

พารามิเตอร์

- hInstance : Type คือ HINSTANCE เป็นตัวที่ชี้ยังโปรแกรมหรือหมายเลขของโปรแกรมใดซึ่งโปรแกรมจะสามารถทำงานได้พร้อมกันหลายๆ โปรแกรมโดยใช้ โค้ดเดียวกัน ดังนั้นจึงต้องมีตัวที่อ้างถึงโปรแกรม

- hPreinstance : เหมือนกับ hInstance แต่เป็นตัวที่ชี้ไปยังโปรแกรมที่ทำงานก่อนหน้า ถ้าหากโปรแกรมก่อนหน้าไม่ได้ทำงานด้วยโค้ดเดียว จะคืนค่า 0

- lPcmdline :Type คือLPSTR เป็นตัวชี้ที่ชี้ไปยังข้อความที่ได้จากการใช้ Command line

- nCmdshow :Type คือ Integer เป็นตัวบอกลักษณะหน้าต่าง เช่น minimize, maximize

ขั้นตอนที่เกิดขึ้นในฟังก์ชัน WinMain

1. การสร้างตัวแปรโครงสร้างของคลาส WNDCLASS (เป็นการกำหนดลักษณะของหน้าต่าง)
2. ทำการ Register Class เข้าไปในระบบ ด้วยฟังก์ชัน RegisterClass()
3. ทำการสร้างหน้าต่างด้วยฟังก์ชัน CreateWindow() มีพารามิเตอร์ 12 ตัว
4. จัดการเกี่ยวกับระบบ Message (ตรวจจับ Mesage)

2.1.2 ฟังก์ชัน WinProc ฟังก์ชัน WinProc จะทำหน้าที่ในการตอบสนองต่อ Message ที่ถูกส่งมาจากฟังก์ชัน WinMain หากต้องการให้โปรแกรมทำอะไรเมื่อมีการส่ง Message เกิดขึ้น ก็จะทำในฟังก์ชันนี้ ฟังก์ชันWinMain และฟังก์ชัน WinProcจะมีส่วนเกี่ยวข้องกันก็ต่อเมื่อฟังก์ชัน WinMain มีการตรวจสอบเจอ Message

รูปแบบฟังก์ชัน WinProc()

LRESULT FAR PASCAL WinProc (HWND hwnd, UINT msg,WPARAM wParam, LPARAM lParam)

พารามิเตอร์

- hwnd : Type คือ HWND เป็นตัวที่อ้างถึงหน้าต่างโปรแกรมที่เราสร้างขึ้น
- msg : Type คือ UINT (unsigned integer) จะเก็บหมายเลข Message ที่จะนำไปใช้ในฟังก์ชันนี้
- wParam : Type คือ WPARAM (word parameter) จะเก็บค่าเหตุการณ์ที่มากับ Message
- lParam : Type คือ LPARAM (long parameter) จะเก็บเหตุการณ์ที่มากับ Messageเช่นกัน

ซึ่งจะสอดคล้องเหตุการณ์ที่ wParam การตรวจสอบเมสเสจของฟังก์ชัน WinProc มีดังนี้

```
LRESULT FAR PASCAL WinProc(HWND hwnd, UINT msg,WPARAM wParam,
LPARAM lParam)
{
    switch(msg)
        {
            case  WM_DESTROY :
            {
```

```
                    //คืนหน่วยความจำ
                    PostQuitMessage(0);
                }break;
                default : //ไม่ตรง case ไหน DefWinProc จะจัดการค่า defaultให้
                return DefWindowProc ( hwnd,  msg,  wParam, lParam);
            }
        }
```

ใน WinProc จะทำการตรวจสอบเมสเสจ (msg) มีรับมา ว่าตรงกับ case ไหนก็จะทำตาม case นั้น ซึ่งเราสามารถสั่งให้โปรแกรมทำอะไรก็ได้ในแต่ละ case ตามเหตุการณ์ที่เราต้องการ เช่น ถ้าเมสเสจที่ส่งตรงกับ WM_DESTROY โปรแกรมก็จะทำการคืนหน่วยความแล้วฟังก์ชัน PostQuitMessage ก็จะทำการกำหนดค่า exit code เป็น 0 ซึ่งจะทำให้ฟังก์ชัน GetMessage คืนค่า False กลับมาทำให้จบ loop การทำงานแล้วก็ออกจากโปรแกรม ถ้าหากไม่เข้า case ไหนเลย โปรแกรมก็จะเรียกใช้ฟังก์ชัน DefWinProc ซึ่งฟังก์ชันนี้ก็จะจัดการค่า default ของ Windows ให้เอง

2.2 การใช้งาน DirectX SDK

2.2.1 DirectX เป็นชุดคำสั่งที่ใช้ในการเขียนเกมที่มีประสิทธิภาพสูงทั้ง เกม 2 มิติ และ 3 มิติ พัฒนาโดยบริษัท Microsoft ซึ่งเป็นชุดคำสั่งที่อำนวยความสะดวกในการเขียนเกม ทำให้ผู้พัฒนาโปรแกรมเรียกใช้ความสามารถด้านฮาร์ดแวร์ของระบบได้อย่างเต็มประสิทธิภาพไม่ว่าจะเป็นการแสดงผลภาพ เสียง และการควบคุมเกมด้วยอุปกรณ์ต่างๆ เช่น mouse keyboard หรือ joystick เป็นต้น อีกทั้งยังมีชุดคำสั่งที่สนับสนุนการเล่นเกมแบบหลายผู้เล่น (Multiplayer Game) บนเครือข่ายเดียวกัน และบนอินเตอร์เน็ทอีกด้วย

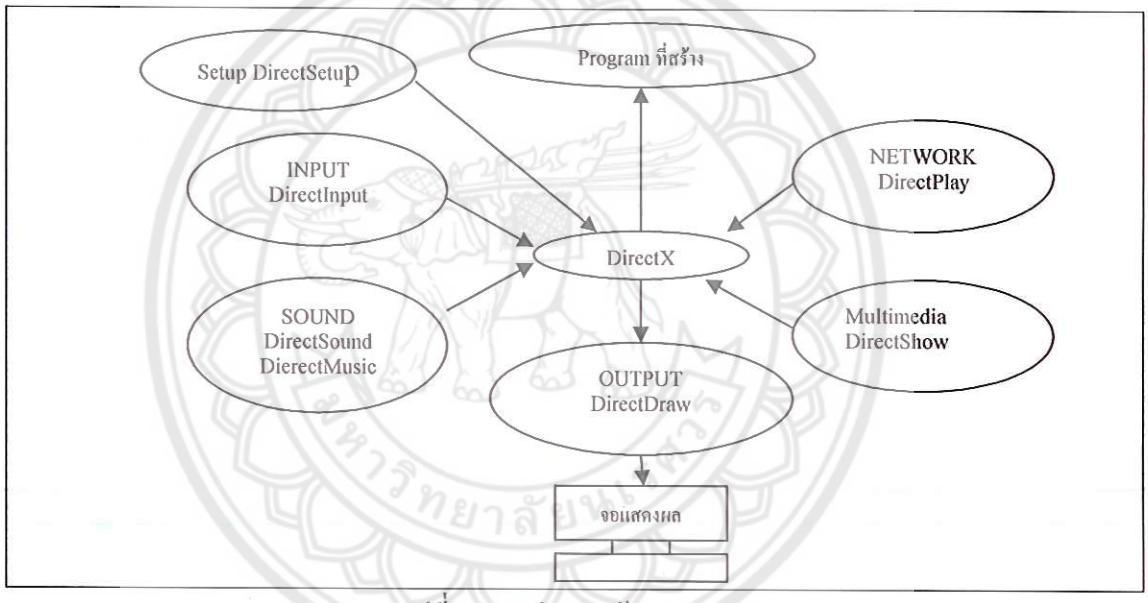
DirectX แบ่งเป็น 2 ส่วนดังนี้

1. DirectX Runtime Library เป็นส่วนหนึ่งของ DirectX ที่ทำให้สามารถเล่นเกมที่เขียนขึ้นโดยDirectX SDK ได้ ซึ่งจะถูกติดตั้งลงไปในระบบปฏิบัติการของ Windows ให้โดยอัตโนมัติ ปัจจุบันเป็น version 9.0C

2. DirectX SDK (Software Development Kit) เป็นชุดคำสั่งที่สร้างขึ้นมาเพื่อใช้ในการพัฒนาโปรแกรม ซึ่งถูกออกแบบมาให้ใช้ร่วมกับตัวแปลภาษาได้หลายภาษา เช่น C หรือ C++ Delphi และ Visual Basic เป็นต้น ซึ่งสามารถศึกษาการพัฒนาเกมด้วย DirectX SDK ได้ด้วย document ที่มีมาพร้อมโปรแกรมโครงการนี้ใช้ DirectX SDK version 8

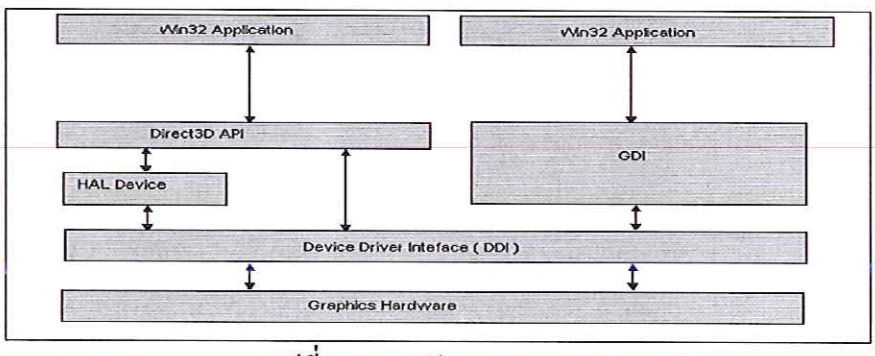
DirectX SDK version 8 ได้รวมเอาส่วนที่คล้ายๆกันเข้าไว้ด้วยกันดังนี้

1. DirectX Graphic เป็นชุดคำสั่งที่ใช้ในการสร้างภาพ ซึ่งประกอด้วย
   - DirectDraw ใช้ในการสร้างภาพ 2 มิติ
   - Direct3D ใช้ในการสร้างเกม 3 มิติ
2. DirectX Audio เป็นชุดคำสั่งที่ใช้ในการเล่นเสียงประกอบเกม ซึ่งประกอบด้วย
   - DirectSoundใช้ในการเล่นเสียง ไฟล์ *.wav
   - DirectMusic ใช้ในการเล่นเพลงประกอบเกม ไฟล์ *.mid , *.wav
3. DirectInput เป็นชุดคำสั่งที่ใช้ในการรับค่าจาก mouse keyboard หรือ joystick
4. DirectPlay เป็นชุดคำสั่งที่ใช้ในการเล่นเกมผ่านระบบเครือข่าย
5. DirectShow เป็นชุดคำสั่งที่ใช้ในการแสดงมัลติมีเดีย เช่น ไฟล์ Mpeg,AVI เป็นต้น
6. DirectSetup เป็นชุดคำสั่งที่ใช้ในการติดตั้งเกม



รูปที่ 2.1 ผังโครงสร้าง DirectX

2.2.2 DirectGraphic (Direct3D) เป็นเครื่องมืออีกตัวหนึ่งที่ DirectX ได้จัดเตรียมไว้ให้ใช้ในการพัฒนาเกมซึ่งจะช่วยในเรื่องของการพัฒนาเกม 3 มิติ



รูปที่ 2.2 สถาปัตยกรรมของ Direct3D [8]

จากรูปเป็นการแสดงความสัมพันธ์ของ Direct3D API, GDI, HAL, และ Hardware ซึ่ง Direct3Dจะมีการทำงานคล้ายกับ GDI ทั้งคู่จะมีการเข้าถึง Graphic Hardware โดยผ่านทาง Device Driver Interface นอกจากนี้ Direct3D ยังสามาราติดต่อกับ Graphic Hardware โดยผ่านทาง HAL ที่ออกแบบโดยผู้ผลิตได้อีก HAL ใน Direct3D8 จะมีการประมวลผลจุดอยู่ด้วยกัน 3 แบบ คือ

- ประมวลผลจุดด้วยโปรแกรม
- ประมวลผลจุดด้วยอุปกรอิเลคทรอนิกส์
- ประมวลผลจุดด้วยโปรแกรมและอุปกรอิเลคทรอนิกส์รวมกัน

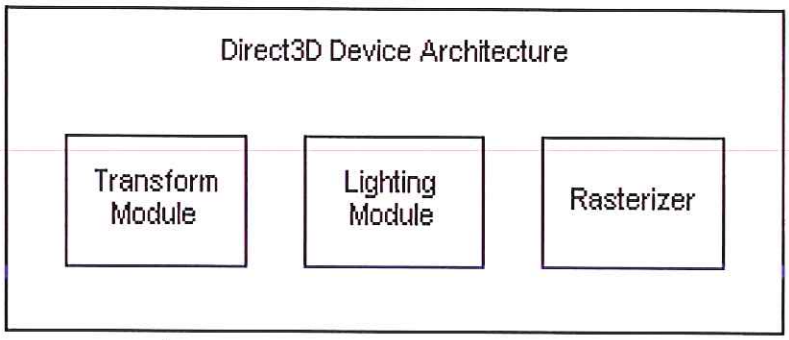
การทำงานเกี่ยวกับ Direct3D

1. **Direct3D Object** เป็นสิ่งที่ใช้ในการอ้างถึง Direct3D หรือเป็นวัตถุหลักของ Direct3D ที่ใช้ในการเข้าถึงฟังก์ชันต่างๆ ใน Direct3D ซึ่งเป็นสิ่งแรกที่จะต้องทำการสร้างและเป็นสิ่งสุดท้ายที่จะต้องทำลายเมื่อมีการเริ่มเขียน Direct3D ก็จะต้องมีตัวชี้ที่ชี้ไปยัง IDirect3D8 ซึ่งจะทำหน้าที่ในการติดต่อกับฟังก์ชันต่าง ๆ ใน Direct3D

```
//ประกาศตัวแปรวัตถุของ Direct3D
LPDIRECT3D8 g_pD3D = NULL;
//ทำการสร้าง Direct3D ถ้าเท่ากับ NULL แสดงว่าสร้างไม่สำเร็จ
if (NULL == (g_pD3D = Direct3DCreate8(D3D_SDK_VERSION))) return E_FAIL;
```

LPDIRECT3D8 เป็นชนิดตัวแปรแบบโครงสร้างของ IDirect3D8 ซึ่งจะมี g_pD3D เป็นวัตถุที่ใช้ในการเข้าถึงฟังก์ชันต่างๆใน Direct3D

2. **Direct3D Device** เป็นส่วนที่ใช้ในการเรนเดอร์ของ Direct3D ซึ่งจะมีเรื่องของการแปรสภาพ แสง และ ราสเตอร์ไร เพื่อสร้างพื้นผิว Direct3D Device ประกอบไปด้วย Transformation Module Light Module และก็ Rasterizing Module ซึ่งแสดงได้ดังรูปที่ 2.3



รูปที่ 2.3 สถาปัตยกรรมของDirect3D Device [8]

การสร้าง Direct3D Device

```
//สร้างวัตถุของ Direct3D Device
LPDIRECT3DDEVICE8 d3dDevice = NULL;
//สร้างตัวที่ใช้ในการอ้างถึงการกำหนดค่าต่างๆใน Direct3D Device
D3DPRESENT_PARAMETERS d3dpp;
//จองหน่วยความจำ
ZeroMemory( &d3dpp, sizeof(d3dpp) );
// กำหนดการสร้างเป็นแบบหน้าต่าง
d3dpp.Windowed   = TRUE;
//ทำการสลับ back buffer กับ font buffer แบบ VSYNC
d3dpp.SwapEffect = D3DSWAPEFFECT_COPY_VSYNC;
//ทำการเริ่มสร้าง Direct3D Device พร้อมทั้งเช็คว่าสร้างสำเร็จหรือไม่
if(FAILED(g_pD3D->CreateDevice(D3DADAPTER_DEFAULT,
D3DDEVTYPE_HAL, hWnd, D3DCREATE_SOFTWARE_VERTEXPROCESSING,
&d3dpp, &d3dDevice  ) ) )
 return E_FAIL;
```

การสร้าง Direct3D Device และสร้างตัวแปรโครงสร้าง D3DPRESENT_PARAMETERS (d3dpp) เพื่อใช้ในการกำหนดค่าต่างๆ จากตัวอย่างเป็นการกำหนดคุณสมบัติของหน้าต่าง และคำสั่งสุดท้ายเป็นการเริ่มสร้าง Direct3D Device พร้อมทั้งเช็คว่าสร้างสำเร็จหรือไม่

3. การเรนเดอร์ ก่อนจะทำการเรนเดอร์ทุกครั้งก็จะต้องทำการ Clean Surface เสียก่อนเพื่อทำการลบพื้นผิวที่ทำการวาดไปก่อนหน้าแล้วค่อยทำการวาดใหม่ และเพื่อเป็นการตั้งค่า Dept Buffer และ Stencil Buffer ใหม่โดยใช้เมธอด Clear ของ IDirect3DDevice8 หลังจากนั้นก็จะใช้เมธอด BeginScene เพื่อเช็คข้อมูลในโครงสร้างที่เราได้ทำการกำหนดค่าไว้แล้ว และเช็คความถูกต้องและความสมบูรณ์ของข้อมูลแล้วจึงทำการเรียกเมธอดต่างๆที่ใช้ในการเรนเดอร์ ซึ่งส่วนนี้จะเป็นการเรียกใช้ทรัพยากรที่ได้ทำการจัดเตรียมไว้แล้ว เมื่อเรนเดอร์เสร็จก็ทำการเรียกเมธอด EndScene เพื่อทำการเช็คความถูกต้องอีกครั้งหลังจากที่ทำการเรนเดอร์เสร็จ

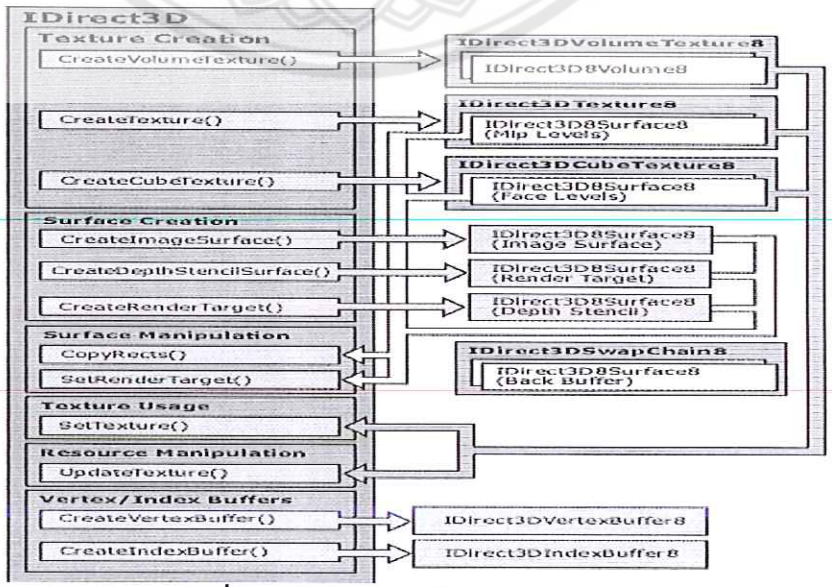
```
HRESULT hr;
//เช็คว่าเมธอด BeginScene ทำงานสำเร็จหรือไม่
if ( SUCCEEDED(pDevice->BeginScene()))
{
```

```
// ทำการเรนเดอร์ตรงส่วนนี้
// ปิดการเรนเดอร์
hr = pDevice->EndScene();
//เช็คว่าเมธอด BeginScene ทำงานสำเร็จหรือไม่
if(FAILED(hr))
 return hr;
}
```

จากตัวอย่างโปรแกมข้างบนถ้าเกิดผิดพลาดขึ้นจะทำอย่างไร

- ถ้า BeginScene เกิดข้อผิดพลาด ก็จะต้องไม่มีการเรียกใช้งานเมธอด EndScecne
- ถ้าเกิดข้อผิดพลาดขึ้นระหว่างการเรนเดอร์ จะต้องเรียกเมธอด EndScene เพื่อจบการทำงาน
- ถ้า EndScene เกิดข้อผิดพลาด ขึ้นอยู่กับเหตุผลว่าจะทำอย่างไร หากไม่ต้องการเรียกมันซ้ำอีกรอบ

4. **Direct3D Resource** เป็น texture และ buffer ที่ใช้ในการเรนเดอร์โปรแกรมของเราจะต้องมีการ สร้าง โหลด คัดลอก ทรัพยากร การใช้ทรัพยากรทั้งหมดสามารถเรียกใช้เมธอดผ่านทาง IDirect3DIndexBuffer8 และ IDirect3DVertexBuffer8 ซึ่งสืบทอดมาจากIDirect3DResource8 ซึ่งเป็นชนิดของทรัพยากรและ texture สามารถเรียกใช้เมธอดผ่านทาง IDirect3DCubeTexture8 IDirect3DTexture8 และ IDirect3DVolumeTexture8 ซึ่งสืบทอบทอดจาก IDirect3DBaseTexture8 ซึ่งเป็นชนิดของ texture



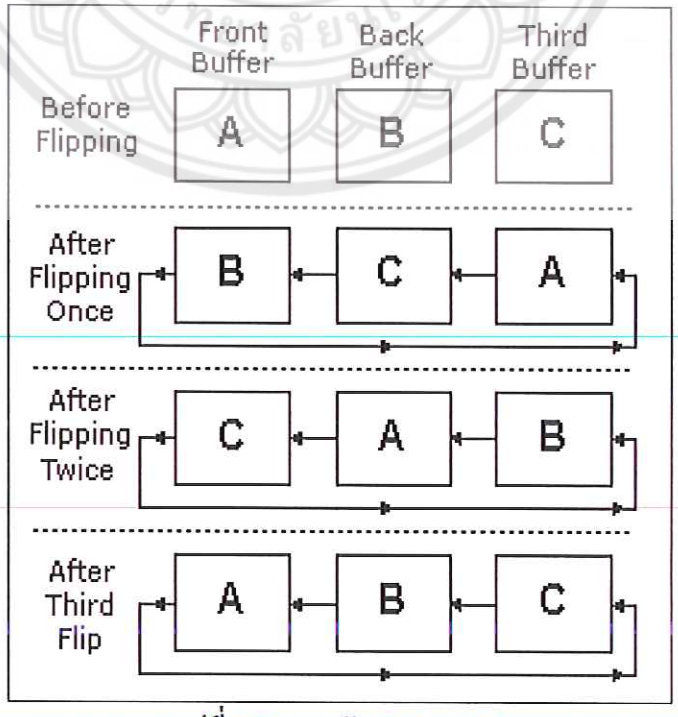
รูปที่ 2.4 ความสัมพันธ์ของทรัพยากร [8]

จากรูปลูกศรชี้ไปทางขวาเป็นการบอกว่ามีการเรียกใช้ชนิดเมธอดใน resource ส่วนลูกศรชี้ไปทางซ้ายเป็นการบอกว่า ชนิดของ resource สามารถที่จะส่งผ่าน argument ไปให้เมธอดได้

5. Surface เป็นตัวแทนของพื้นที่ที่สร้างออกมาซึ่งจะถูกเก็บในหน่วยความจำแสดงผลซึ่งได้จากการ์ดแสดงผล เราสามารถสร้างวัตถุของ Surface โดยใช้ IDirect3DSurface8 ซึ่งมีเมธอดดังนี้ที่เกี่ยวข้องกับวัตถุของ Surface

- IDirect3DCubeTexTure8::GetCubeMapSurface
- IDirect3DDevice8::CreateDepthStencilSurface
- IDirect3DDevice8::CreateImageSureface
- IDirect3DDevice8::CreateRenderTarget
- IDirect3DDevice8::GetBackBuffer
- IDirect3DDevice8::GetDepthStencilSurface
- IDirect3DDevice8::GetFontBuffer
- IDirect3DDevice8::GetRendert Target
- IDirect3DSwapChain8::GetBackBuffer
- IDirect3DTexture8::GetSurfaceLevel

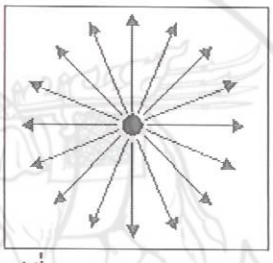
6. การสลับ Surface ในการจะทำให้เกิดการเคลื่อนไหวของภาพนั้นจะต้องมีการสลับ Surface เป็นลำดับไปเรื่อยๆเพื่อทำให้ภาพนิ่งหลายๆภาพที่สลับกันไปมานั้นมีการเคลื่อนไหวเกิดขึ้น



รูปที่ 2.5 การสลับ Surface [8]

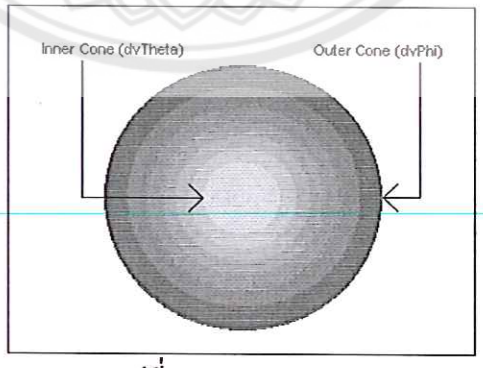
จากรูปจะมี Front Buffer เป็นตัวเก็บ Surface (A) ที่ต้องการแสดง ส่วน Back Buffer จะเป็นตัวเก็บ Surface (B) ที่รอการแสดงต่อจาก Surface (A) ที่อยู่ใน Front Buffer ส่วน Third Buffer ก็จะเป็นตัวเก็บ Surface (C) ที่รอการแสดงต่อจาก Surface(B) เมื่อจะแสดง Surface (B) ก็จะต้องทำการเลื่อนSurface (B) ยัง Front Buffer เพื่อทำการแสดง ส่วน Surface (A) ก็จะเลื่อนไปอยู่ที่ Third Buffer เพื่อรอการแสดงต่อไปส่วน Surface (C) ก็จะเลื่อนไปที่ Back Buffer ซึ่งการทำงานนี้ก็จะทำการสลับกันไปเรื่อยๆ จึงทำให้มองดูเหมือนภาพมีการเคลื่อนไหว

7. Lights ความสามารถของ Direct3D สามารถที่จะจำลองแสงขึ้นมาใช้งานได้ จึงทำให้เกมที่จะพัฒนามีความสมจริงมากขึ้น ใน Direct3D จะมีแสงที่ใช้อยู่ 3 ชนิดคือ Point Lights Spotlights และ Directional Lights เราสามารถสร้างวัตถุของแสงโดยใช้การสร้างผ่านทาง D3DLIGHT8 ซึ่งเป็นโครงสร้างของแสงใน Direct3D เช่น D3DLIGHT8 d3dLight; ซึ่ง d3dLight จะเป็นวัตถุของแสงที่เราจะใช้ในการกำหนดคุณสมบัติต่างๆ ของแสง



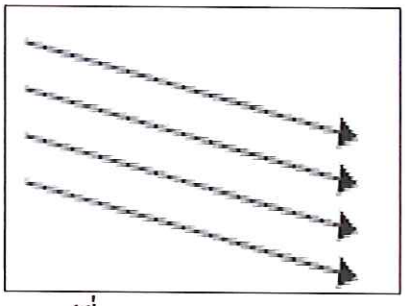
รูปที่ 2.6 Point Lights [8]

จากรูปที่ 2.6 เป็นการแสดงลักษณะของแสงแบบ Point Lights ซึ่งแสงจะกระจายตัวออกไปเป็นสีของแสงรอบๆจุดที่เป็นจุดกำเนิดแสง



รูปที่ 2.7 Spotlights [8]

จากรูปที่ 2.7 เป็นการแสดงลักษณะของแสงแบบ Spotlights ซึ่งมีลักษณะคล้ายกรวยซึ่งตรงกลางของกรวยจะมีความสว่างของแสงมากที่สุดและจะค่อยๆลดความสว่างลงไปเรื่อยๆจนถึงขอบของกรวย



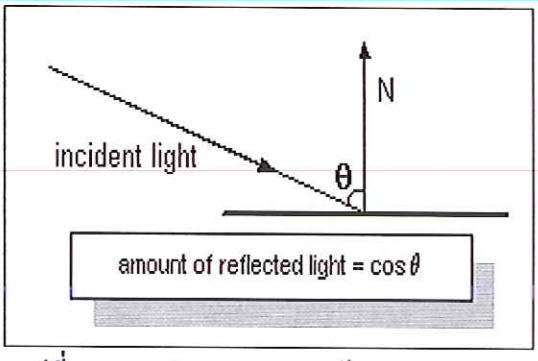
รูปที่ 2.8 Directional Lights

จากรูปที่ 2.8 เป็นการแสดงลักษณะของแสงแบบ Directional Lights ซึ่งจะมีลักษณะเป็นแสงขนาน มีสีเดียวและทิศทางเดียว ไม่มีตำแหน่งซึ่งเป็นแสงที่ส่องมาจากที่ไกลๆ

8. Materials เป็นตัวที่อธิบายถึงการสะท้อนของแสงและแสงที่เปล่งออกมาที่เกิดขึ้นกับวัตถุเราสามารถกำหนดคุณสมบัติต่างของแสงใน Direct3D โดยใช้การอ้างผ่านโครงสร้างของ D3DMATERIAL8 ซึ่งมีโครงสร้างดังนี้

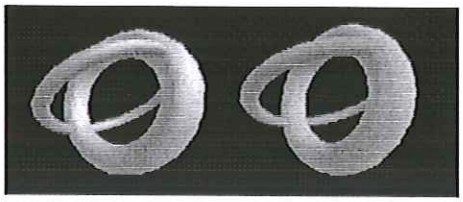
```
typedef struct _D3DMATERIAL8 {
        D3DCOLORVALUE   Diffuse;
        D3DCOLORVALUE   Ambient;
        D3DCOLORVALUE   Specular;
        D3DCOLORVALUE   Emissive;
        float           Power;
                        } D3DMATERIAL8;
```

Diffuse และ Ambient (ค่าที่กำหนดอยู่ในช่วง 0 -1) Diffuse เป็นการกระจายตัวของแสง ส่วนAmbient เป็นการสะท้อนของแสงออกไปรอบๆวัตถุ สามารถคำนวณการสะท้อนของแสงได้ตามรูป



รูปที่ 2.9 การคำนวณการสะท้อนของแสง [8]

Specular และ Power (ค่าที่กำหนดอยู่ในช่วง 0 -1) เป็นการสร้างแสงที่สว่างมากๆบนวัตถุ เพื่อให้เกิดความมันวาวบนวัตถุส่วน Power เป็นการกำหนดรูปร่างของ Specular เช่น รูปร่างยาว กลม วงรี เป็นต้น



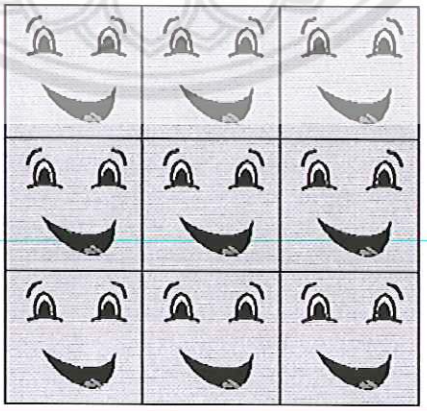
รูปที่ 2.10 การใช้งาน Specular และ Power [8]

จากรูปที่ 2.10 รูปทางซ้ายเป็นใช้คุณสมบัติของ Specular และ Power ส่วนรูปทางขวาจะไม่ใช้คุณสมบัติของ Specular และ Power

Emissive (ค่าที่กำหนดอยู่ในช่วง 0 -1) เป็นการกำหนดความสว่างของวัตถุเอง

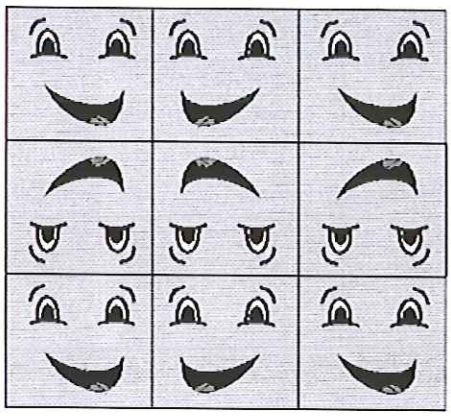
9. Textures ก่อนที่คอมพิวเตอร์จะทำการแสดงภาพ 3 มิติ ออกมานั้นจะต้องมีการทำให้ภาพมีความสมจริงโดยกำหนดค่าต่างๆมากมาย ซึ่ง Texture ก็จะต้องถูกกำหนดด้วย Texture ก็คือรูปต่างที่ต้องการแสดงออกมาให้เห็นบนตัววัตถุ เช่น รอยขีดข่วน รอยเปื้อนสี เป็นต้น ดังจะเห็นในเกมหลายๆเกมเช่น ป้ายรถประจำทาง พื้นหินอ่อน กล่อง กำแพง ก้อนหิน เป็นต้นในการนำ Texture ไปใส่ลงบนวัตถุมีอยู่ด้วยกัน 4 แบบดังนี้

- แบบ Warp เป็นการนำ Texture แบบเดียวกันมาต่อๆกันจนเต็มพื้นผิวของวัตถุซึ่งวัตถุจะมีขนาดใหญ่กว่า Texture



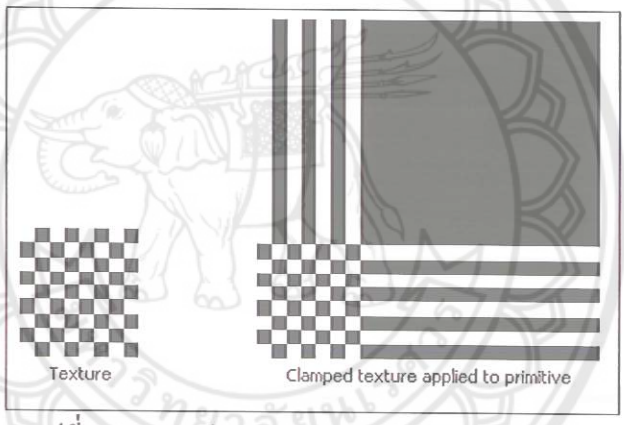
รูปที่ 2.11 ตัวอย่างการทำ Texture แบบ Warp [8]

- แบบ Mirror เป็นการนำ Texture แบบเดียวกันมาต่อๆกันจนเต็มพื้นผิวของวัตถุแต่จะทำการกลับ Texture ที่อยู่ติดๆกัน ซึ่งวัตถุจะมีขนาดใหญ่กว่า Texture



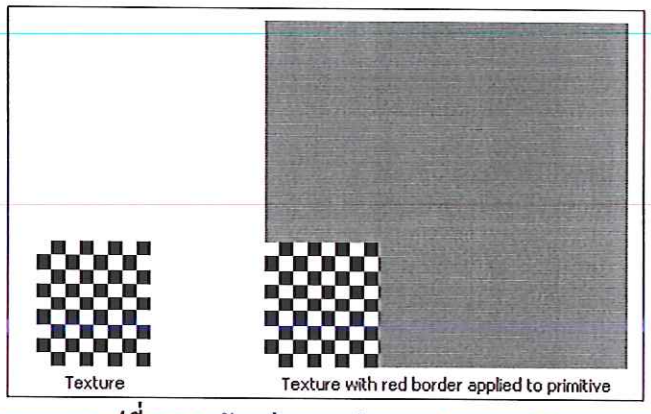
รูปที่ 2.12 ตัวอย่างการทำ Texture แบบ Mirror [8]

- แบบ Clamp จะเป็นการสร้างภาพแบบเดียวกับจุดพิกเซลที่อยู่รอบๆขอบของ Texture ออกไปจนเต็มพื้นผิวของวัตถุ ซึ่งวัตถุจะมีขนาดใหญ่กว่า Texture



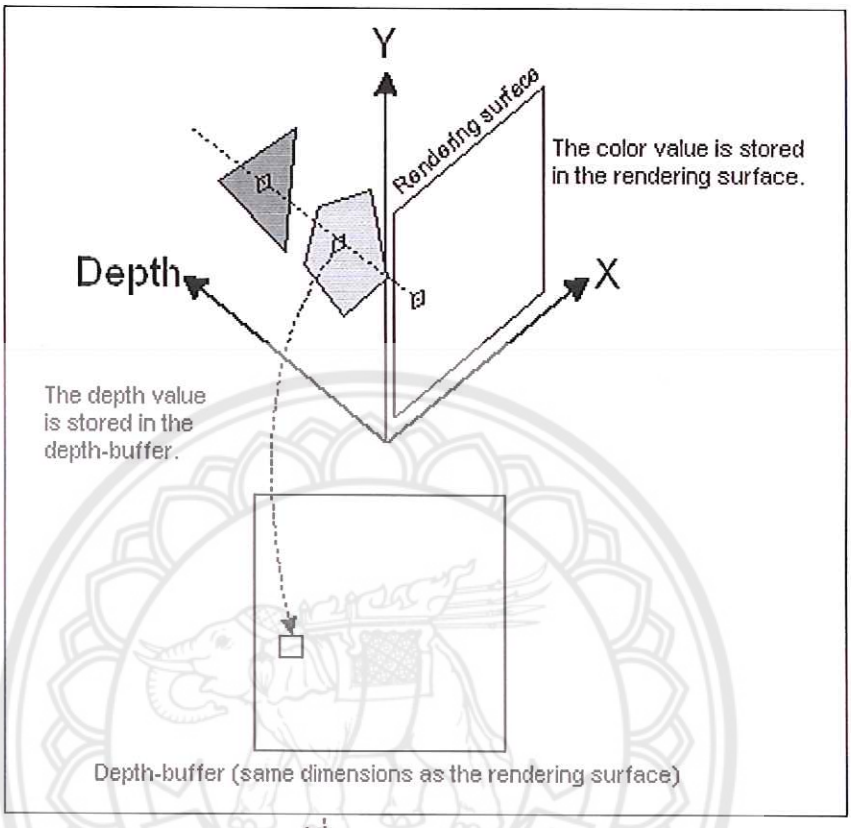
รูปที่ 2.13 ตัวอย่างการทำ Texture แบบ Clamp [8]

- แบบ Border จะเป็นการสร้างสีที่ต้องการรอบๆ Texture แทนการนำ Texture มาแสดงซ้ำหลายๆรอบ



รูปที่ 2.14 ตัวอย่างการทำแบบ Border [8]

10. Dept Buffer คือ z-buffer เป็นส่วนที่ใช้ในการเก็บค่าความลึกของวัตถุ เมื่อ Direct3D จะทำการเรนเดอร์ก็จะไปอ่านค่าความลึกของวัตถุจาก z-buffer แสดงได้ดังรูปที่ 2.15



รูปที่ 2.15 Dept Buffer [8]

จากรูปเป็นการแสดงให้เห็นถึงการใช้งาน Dept Buffer ถ้าหากค่าความลึกมีค่าเท่ากับโพลิกอนรูปห้าเหลี่ยม เมื่อทำการเรนเดอร์เสร็จ Rendering Surface ก็จะแสดงสี่เหลี่ยมสีเหลืองออกมา แต่ถ้าหากค่าความลึกมีค่าเท่ากับโพลิกอนรูปสามเหลี่ยม เมื่อทำการเรนเดอร์เสร็จ Rendering Surface ก็จะแสดงสี่เหลี่ยมสีม่วงออกมา

2.2.3 DirectAudio เกมจำเป็นจะต้องมีการใส่เสียง effect และเพลงประกอบเกมด้วย เพราะเสียงเป็นสิ่งสำคัญที่จะทำให้เกมที่พัฒนามีความน่าสนใจมากขึ้นทำให้เร้าใจผู้เล่นซึ่งผู้เล่นก็จะเพลิดเพลินไปกับการเล่นเกมซึ่ง DirectAudio มีเครื่องมือที่ช่วยให้สามารถใส่เสียงลงไปในเกมเรียบร้อยแล้ว

DirectAudio แบ่งออกเป็น 2 ประเภทดังนี้

1. DirectSound เป็นชุดคำสั่งหนึ่งของ DirectX ที่ช่วยในการจัดการเรื่องเกี่ยวกับเสียง effect โดย DirectSound จะทำการโหลดไฟล์เสียง (*.wav) มาเก็บไว้ที่ Buffer ก่อนแล้วจึงนำไฟล์เสียงที่เก็บไว้ใน Buffer นี้ออกมาทำการเล่นเสียงตามที่ต้องการ

ขั้นตอนที่เกิดขึ้นในการเล่นเสียง

1. ทำการสร้าง Object ของ DirectSound โดยใช้ฟังก์ชัน DirectSoundCreate8 ตั้งระดับการทำงานร่วมกันกับโปรแกรมอื่นๆ โดยใช้ฟังก์ชัน SetCooperativeLevel
2. ทำการโหลดไฟล์เสียงมาเก็บไว้ที่ Buffer
3. ทำการเล่นเสียงด้วยฟังก์ชัน Play
4. การหยุดเล่นเสียงด้วยฟังก์ชัน Stop
5. ทำการปิด Object ที่สร้างขึ้นทั้งหมด

1. สร้าง Object ของ DirectSound ใช้ชุดคำสั่ง DirectSoundCreate8 ในการสร้าง Object ของ DirectSound

รูปแบบฟังก์ชัน DirectSoundCreate8 ()

```
HRESULT WINAPI DirectSoundCreate8 (
                                    LPCGUID lpcGuidDevice,
                                    LPDIRECTSOUND8 * ppDS8,
                                    LPUNKNOWN pUnkOuter
                                  );
```

พารามิเตอร์

- lpcGiudDevice: เป็นdeviceสำหรับแสดงเสียงออกมา ซึ่งส่วนนี้จะถูกกำหนดให้เป็น NULL (default)
- ppDS8 : เป็นพอยเตอร์ที่ชี้ไปยัง DirectSound หรือ Object ที่สร้างขึ้นนั่นเอง
- pUnkOuter : มักจะถูกกำหนดให้เป็น NULL

* ถ้าทำงานสำเร็จจะ return ค่า DS_OK

2. ตั้งระดับการทำงานร่วมกัน (Cooperation Level) จะต้องมีการตั้งระดับการทำงานร่วมกันกับโปรแกรมอื่นๆ เพื่อให้โปรแกรมทำงานสัมพันธ์กัน

รูปแบบฟังก์ชัน SetCooperativeLevel ()

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

พารามิเตอร์

- Hwnd : เป็นแฮนเดิลของหน้าต่าง

- dwLevel : รูปแบบการแสดงเสียงเช่น DSSCL_NORMAL จะทำงานกับแอพพลิเคชั่นอื่นได้ดี

* ถ้าทำงานสำเร็จจะ return ค่า DS_OK

3. โหลดไฟล์เสียงมาเก็บที่ Buffer มีขั้นตอนดังนี้

- ทำการเปิดไฟล์เสียง

- หาข้อมูลของไฟล์เสียง

- หารูปแบบของข้อมูลไฟล์เสียง

- อ่านรูปแบบของไฟล์เสียง

- สร้าง Buffer ของ DirectSound เพื่อเก็บข้อมูลของไฟล์เสียง

- อ่านข้อมูลเสียงที่ได้มาเก็บไว้ใน Buffer ที่เราได้สร้างเอาไว้

4. ทำการเล่นเสียงด้วยฟังก์ชัน Play

รูปแบบฟังก์ชัน Play ()

```
HRESULT Play   (
                    DWORD dwReserved1,
                    DWORD dwPriority,
                    DWORD dwFlags
                ) ;
```

พารามิเตอร์

- dwReservedl : กำหนดให้เป็น 0

- dwPriority : เป็นค่าความสำคัญ(Priority)ของเสียงที่จะเล่น มีค่าตั้งแต่ 0x00000000 – 0xffffffff

- dwFlage : รูปแบบการเล่น เช่น DSBPLAY_LOOPING เป็นการเล่นวนไปเรื่อยๆไม่รู้จบ

5. การหยุดเล่นเสียงด้วยฟังก์ชัน Stop

รูปแบบฟังก์ชัน Stop ()

```
HRESULT Stop();
```

ฟังก์ชัน Stop จะไม่มีพารามิเตอร์ ตัวอย่างการใช้ เช่น soundPointer -> Stop ();

6. ทำการปิด Object ที่สร้างขึ้น เมื่อมีการสร้าง Object ขึ้นก็จะต้องทำการปิด Object ที่สร้างขึ้น ทั้งหมดของ DirectSound และ DirectSound Buffer

รูปแบบฟังก์ชัน Release ()

```
ULONG Release(void);
```

ไม่มีพารามิเตอร์ตัวอย่างการใช้งานเช่น

```
Object->Release(); object = NULL;
ppDS8->Release(); ppDS8 = NULL;
```

2. DirectMusic มีลักษณะการใช้งานคล้ายๆกับ DirectSound เพียงแต่จะเป็นการเล่นไฟล์ประเภท MIDI ในเวอร์ชัน 8 นี้ DirectMusic สามารถที่จะเล่นไฟล์ Wave ได้เหมือนกันแต่ก็ไม่ดีเท่ากับ DirectSound ซึ่ง DirectSound สามารถทำงานที่ซับซ้อนได้ดีกว่า

ขั้นตอนที่เกิดขึ้นของ DirectMusic

1. ทำการ Initialize Com ก่อน
2. Load ไฟล์ที่จะทำการเล่น
3. เล่นไฟล์ที่ได้โหลดมา
4. การหยุดเล่นและทำการลบ object ที่ได้สร้างขึ้น

1. ทำการ Initialize Com ทุกครั้งที่จะมีการเรียกใช้ DirectMusic จะต้องทำการ Initialize ใน WinMain สร้างตัวแปร Pointer ก่อน

รูปแบบฟังก์ชัน InitAudio ()

```
HRESULT InitAudio ( IDirectMusic** ppDirectMusic,
                    IDirectSound** ppDirectSound,
                    HWND hWnd,
                    DWORD dwDefaultPathType,
                    DWORD dwPChannelCount,
                    DWORD dwFlags,
                    DMUS_AUDIOPARAMS *pParams
                   );
```

พารามิเตอร์

- ppDirectMusic : ตัวชี้ที่ชี้ไปยัง DirectMusic object
- ppDirectSound : เป็น Address ที่อ้างถึง IDirectSound ถ้าเป็น NULL จะทำให้ ฟังก์ชันนี้เล่น ไฟล์ Wave ได้
- hWnd : เป็นแฮนเดิลของหน้าต่างสำหรับสร้าง DirectSound
- dwDefaulPathType : รูปแบบของเสียง กำหนดเป็น 0 ถ้าไม่ต้องการ Default Path Type
- dwPChannelCount : เป็นจำนวน Performance Channel จะใช้ได้ถ้า dwDefaultPathType ไม่เป็น 0
- dwFlage : เป็นรูปแบบ Synthesizer เราจะไม่สนใจถ้า lParams ไม่เป็น NULL
- lParams : กำหนดให้เป็น NULL (Default)

2. Load ไฟล์ที่จะทำการเล่น

รูปแบบฟังก์ชัน LoadObjectFromFile ()

```
HRESULT LoadObjectFromFile (
                    REFGUID rguidClassID,
                    REFIID iidInterfaceID,
                    WCHAR *pwzFilePath,
                    void ** ppObject
                    );
```

พารามิเตอร์

- rguidClassID : ชื่อ Class ใน IDirectMusicLoader8 สำหรับเลือก Type ของ Object
- iidInterfaceID : เป็นชื่อที่ต้องการเชื่อมต่อจะขึ้นต้นด้วย " IID_ "
- wstrFilePath : ชื่อไฟล์ที่ต้องการอ้างถึงพร้อมที่อยู่ของไฟล์
- ppObject : เป็นตัวแปรที่ใช้ในการอ้างถึงเมื่อต้องการติดต่อ

3. เล่นไฟล์ที่ได้โหลดมา ก่อนที่จะเล่นไฟล์จะต้องทำการโหลดไปยัง Synthesizer ก่อนดังนี้

```
g_pSegment->Download( g_pPerformance );
```

ต่อไปก็ทำการเล่นเสียงดังนี้

```
g_pPerformance->PlaySegmentEx (
                    g_pSegment,  // Segment ที่จะทำการเล่น
                    NULL,      // ยังไม่มีใช้งานใน version 8 กำหนดเป็น NULL
                    NULL,      // For transitions.
                    0,         // Flags.
                    0,         // Start time;
                    NULL,      // Pointer that receives segment state.
                    NULL,      // กำหนดเป็น NULL
                    NULL       // กำหนดเป็น NULL (Default Path)
                              );
```

รูปแบบฟังก์ชัน PlaySegmentEx ()

```
HRESULT PlaySegmentEx (
                    IUnknown* pSource,
                    WCHAR *pwzSegmentName,
                    IUnknown* pTransition,
                    DWORD dwFlags,
                    __int64 i64StartTime,
                    IDirectMusicSegmentState** ppSegmentState,
                    IUnknown* pFrom,
                    IUnknown* pAudioPath
                         );
```

พารามิเตอร์

- pSource : Segment ที่จะทำการเล่น
- pwzSegmentName : ยังไม่มีใช้งานใน version 8 กำหนดเป็น NULL
- pTransition : กำหนด Transition กำหนดเป็น NULL
- dwFlags : ค่า Flage ใน DMUS_SEGF_FLAGS เป็นการควบคุมเวลาและ action อื่นๆ บน Segment
- i64StartTime : เวลาที่จะทำการเริ่มเล่น กำหนดเป็น 0
- ppSegmentState : Pointer ที่ใช้อ้างถึง Segment State

- pFrom : กำหนดเป็น NULL
- pAudioPath : กำหนดเป็น NULL (Default Path)

4. การหยุดเล่นและทำการลบ Object ที่ได้สร้างขึ้น ก่อนที่จะออกจากโปรแกรมจะต้องทำการหยุดเล่นเสียงและทำการคืนหน่วยความจำโดยทำการลบ object ที่ได้สร้างขึ้นดังนี้

รูปแบบฟังก์ชัน Stop ()

```
HRESULT  Stop (
                IDirectMusicSegment* pSegment,
                IDirectMusicSegmentState* pSegmentState,
                MUSIC_TIME mtTime,
                DWORD dwFlags
            );
```

พารามิเตอร์

- pSegment : หยุดการเล่น Segment กำหนดให้เป็น NULL หยุดเล่นทั้งหมดที่ mtTime
- pSegmentState : หยุดการเล่น Segment State กำหนดเป็น NULL หยุดเล่นทั้งหมด
- mtTime : เวลาที่ทำการหยุดเล่น
- dwFlag : ค่า Flags รูปแบบการหยุดเล่น 0 คือ ทำการหยุดเล่นโดยทันที

หลังจากนั้นก็ทำการคืนหน่วยวามจำ

```
// ทำการปิด Performane
g_pPerformance->CloseDown();

// ลบ object ที่สร้างขึ้น
g_pLoader->Release();
g_pPerformance->Release();
g_pSegment->Release();

// ทำการปิด COM
CoUninitialize();
```

2.2.4 DirectInput ในการรับค่าจากผู้เล่นเกมเพื่อควบคุมเกมที่สร้างขึ้นนั้นสามารถรับได้ 2 ทาง คือ WM_KEYDOWN และ DirectInput ซึ่งข้อแตกของทั้ง 2 วิธีก็คือ ในการกดปุ่มค้างไว้นั้น WM_KEYDOWN จะต้องรอเลวลาประมาณ 0.5 – 1 วินาที จึงจะเริ่มทำงานต่อ แต่ DirectInput ไม่ต้องรอสามารถทำงานต่อได้เลยซึ่งจะทำให้เกมสามารถควบคุมได้เร็วและสมจริงมากขึ้น

ขั้นตอนที่เกิดขึ้นในการรับค่าจากDirectInput

1. ประกาศตัวแปร DirectInput
2. ทำการสร้าง DirectInput
3. ตั้งค่าการทำงานร่วมกัน
4. ทำการอ่านค่าจากการกดปุ่ม
5. ลบ DirectInput และคืนหน่วยความจำ

1. ประกาศตัวแปร **DirectInput**

```
LPDIRECTINPUT8     input;
LPDIRECTINPUTDEVICE8     keyboard;
```

จากตัวอย่างข้างบน

* Input เป็น object ของ DirectInput
* keyboard เป็น object ของ DirectInput Device

2. ทำการสร้าง **DirectInput** ขั้นตอนนี้จะเป็นการสร้าง Directinput และ DirectInput Device

```
//สร้างDirectInput
DirectInput8Create (hInstance, DIRECTINPUT_VERSION,
                    IID_IDirectInput8, (void**)&input, NULL);
//สร้าง Device
input->CreateDevice (GUID_SysKeyboard, &keyboard, NULL);
```

รูปแบบฟังก์ชัน DirectInput8Create()

```
HRESULT WINAPI DirectInput8Create(
```

```
                    HINSTANCE hinst ,
                    DWORD dwVersion ,
                    REFIID riidltf ,
                    LPVOID* ppvOut ,
                    LPUNKNOWN punkOuter
                );
```

พารามเตอร์

- hinst : ตัวแปรที่อ้างถึงโปแกรมที่เรากำลังทำงานอยู่
- dwVersion : เป็น version ของ DirectX ใช้ DIRECTINPUT_VERSION (ค่าคงที่)
- riidltf : ค่ารหัสที่ต้องการใช้ ถ้าใช้ IID_IDirectInput8 หมายถึงใช้รหัสแบบ ANSI หรือ UNICODE
- ppvOut : Address ที่อ้างถึง object ของ DirectInput
- punkOuter : กำหนดให้เป็น NULL

รูปแบบฟังก์ชัน CreateDevice ()

```
HRESULT CreateDevice (
                    REFGUID rguid,
                    LPDIRECTINPUTDEVICE *lplpDirectInputDevice,
                    LPUNKNOWN pUnkOuter
                    );
```

พารามิเตอร์

- Rguid : เป็นการกำหนดค่าคงที่ GIUD ของ Input Device ที่เราสร้างถ้าเป็น Keyboard ใช้ GUID_SysKeboardถ้าเป็น Mouse ใช้ GUID_SysMouse
- lplpDeviceInputDevice : Address ที่อ้างถึง object ของ DirectInuputDevice
- pUnkOuter : กำหนดให้เป็น NULL

เมื่อทำการสร้าง Device แล้วต่อไปเราก็ต้องทำการกำหนดคุณสมบัติให้กับ Device ที่เราสร้างขึ้นดังนี้

```
keyboard->SetDataFormat (&c_dfdDIKeyboard);
```

ฟังก์ชัน SetDataFormat ใช้กำหนดรูปแบบของข้อมูลที่รับมาจาก Device โดยใช้ตัวแปรโกบอลที่ได้ประกาศไว้ใน Direct Input ในที่นี้เราใช้ c_dfDIKeyboard หมายถึง เรารับข้อมูลมาจาก Keyboard นอกจากนี้เรายังใช้ค่าอื่นๆได้อีก เช่น

c_dfDIMouse รับค่าจาก Mouse

c_dfDIMouse2 รับค่าจาก Mouse2

c_dfDIJoystick รับค่าจาก Joystick

c_dfDIJoystick2 รับค่าจาก Joystick2

3. ตั้งค่าการทำงานร่วมกัน ขั้นตอนนี้จะต้องตั้งค่าการทำงานร่วมกันของอุปกรณ์ก่อนเช่น

```
keyboard->SetCooperativeLevel(hwnd, DISL_FOREGROUND |
                                    DISL_NONEXCLUSIVE);
```

พารามิเตอร์ตัวแรก (hwnd) เป็นแฮนเดิลวินโดว์ส่วนตัวที่สองเป็น Flag ที่อธิบายถึง Cooperative Level

หลังจากที่กำหนดคุณสมบัติของ Device แล้วเราจะต้องเรียกใช้ฟังก์ชัน Acquire เพื่อเป็นการบอกให้เริ่มต้นรับข้อมูล เช่น

```
//เริ่มรับข้อมูลจาก Keyboard
keyboard->Acquire();
```

4. อ่านค่าจากการกดปุ่ม เราจะทำกาเรียกใช้ฟังก์ชัน GetDeviceState เพื่อทำการรับค่ามาตรวจสอบ

```
//ฟังก์ชันตรวจสอบค่าจาก Keyboard
void WINAPI ProcessKBInput()
{
//ประกาศค่าคงที่ KEYDOWN เพื่อตรวจสอบปุ่มที่กดว่าใช่ 0x80 (ฐาน 16) หรือไม่
#define KEYDOWN(name, key) (name[key] & 0x80)
 // ประกาศตัวแปร Array คือ buffer
 char    buffer[256];
//ทำการเก็บค่าไว้ใน buffer เพื่อจะนำไปตรวจสอบ
```

```
keyboard->GetDeviceState(sizeof(buffer),(LPVOID)&buffer);
// เช็คปุ่มลูกศรไปทางขวา
if (KEYDOWN(buffer, DIK_RIGHT));
// เช็คปุ่มลูกศรไปทางซ้าย
else if(KEYDOWN(buffer, DIK_LEFT));
// เช็คปุ่มลูกศรขึ้น
if (KEYDOWN(buffer, DIK_UP)) ;
// เช็คปุ่มลูกศรลง
else if (KEYDOWN(buffer, DIK_DOWN));
}
```

5. การคืนหน่วยความจำ สุดท้ายคือ การลบ Direct Input และคืนหน่วยความจำ

```
// หยุดการรับข้อมูล
keyboard->Unacquire();
//ลบ object ที่สร้างขึ้น
keyboard->Release();
keyboard = NULL;
input->Release();
input = NULL;
```

บทที่ 3
วิธีการดำเนินงานโครงงานวิศวกรรม

โครงงานนี้แบ่งขั้นตอนการพัฒนาเกมสามมิติออกเป็น 3 ส่วนคือ

1. ขั้นตอนการออกแบบโมเดลที่จะใช้ในเกม
2. ขั้นตอนการออกแบบเงื่อนไขต่างๆในการเล่นเกม
3. ขั้นตอนการเขียนโปรแกรมเพื่อให้เกมเป็นไปตามเงื่อนไขที่ได้ออกแบบไว้

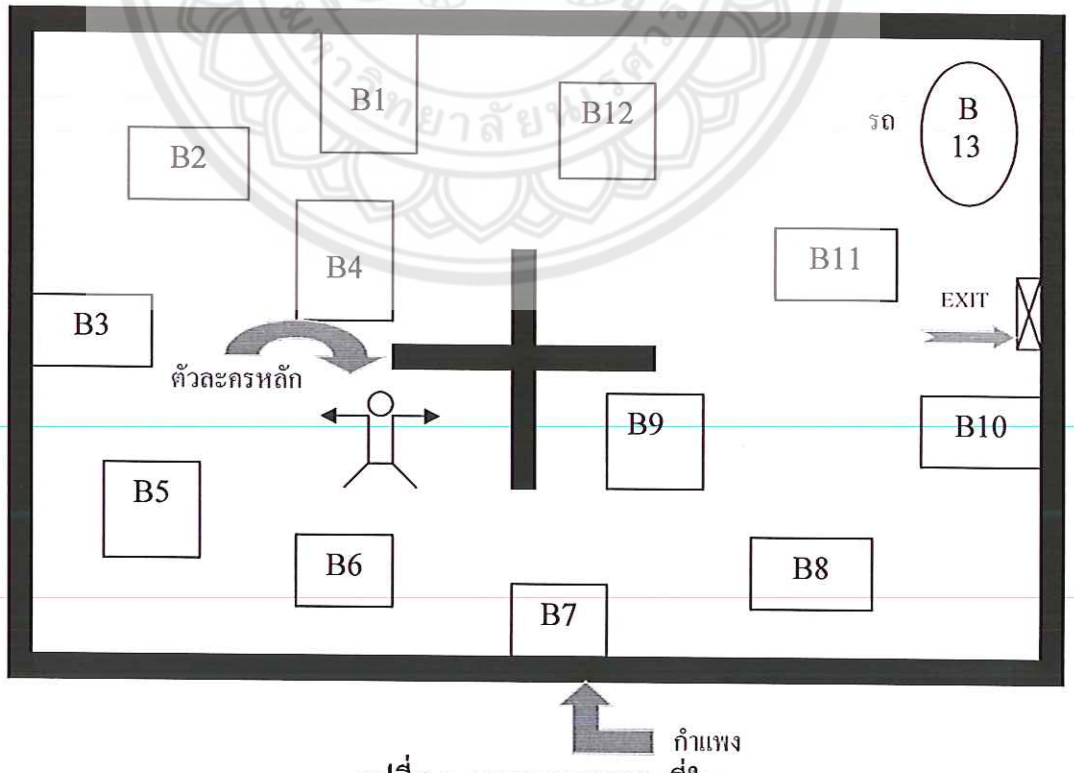
3.1 การออกแบบโมเดล

โมเดลที่ใช้ในเกมนี้จะแบ่งเป็น 3ส่วนดังนี้

1. โมเดลที่ใช้เป็นแผนที่ในเกม
2. โมเดลที่ใช้เป็นตัวละครหลักในเกม
3. โมเดลที่ใช้เป็นสิ่งของต่างๆในเกม

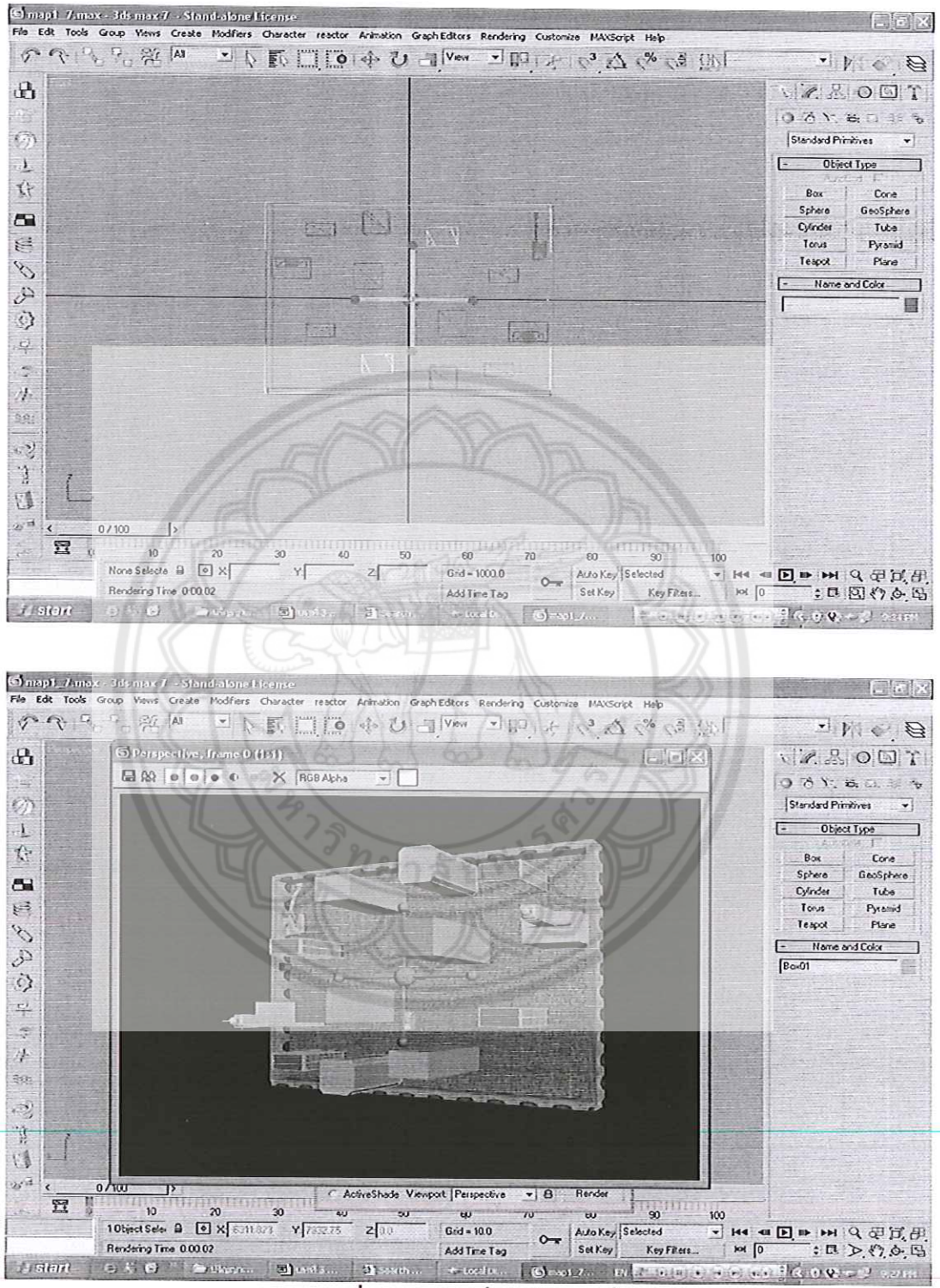
ซึ่งโมเดลเหล่านี้สร้างโดยใช้โปรแกรมสร้างโมเดลสามมิติ คือ โปรแกรม 3ds max 7

1. โมเดลที่ใช้เป็นแผนที่ในเกม จะเป็นโมเดลที่มีขนาดใหญ่ที่สุด ซึ่งทำหน้าที่จำลองเป็นโลกเสมือนในเกมจะประกอบไปด้วยตึก (B1-B12) กำแพง ประตูทางออก รถ (B13) ตามรูปที่ 3.1



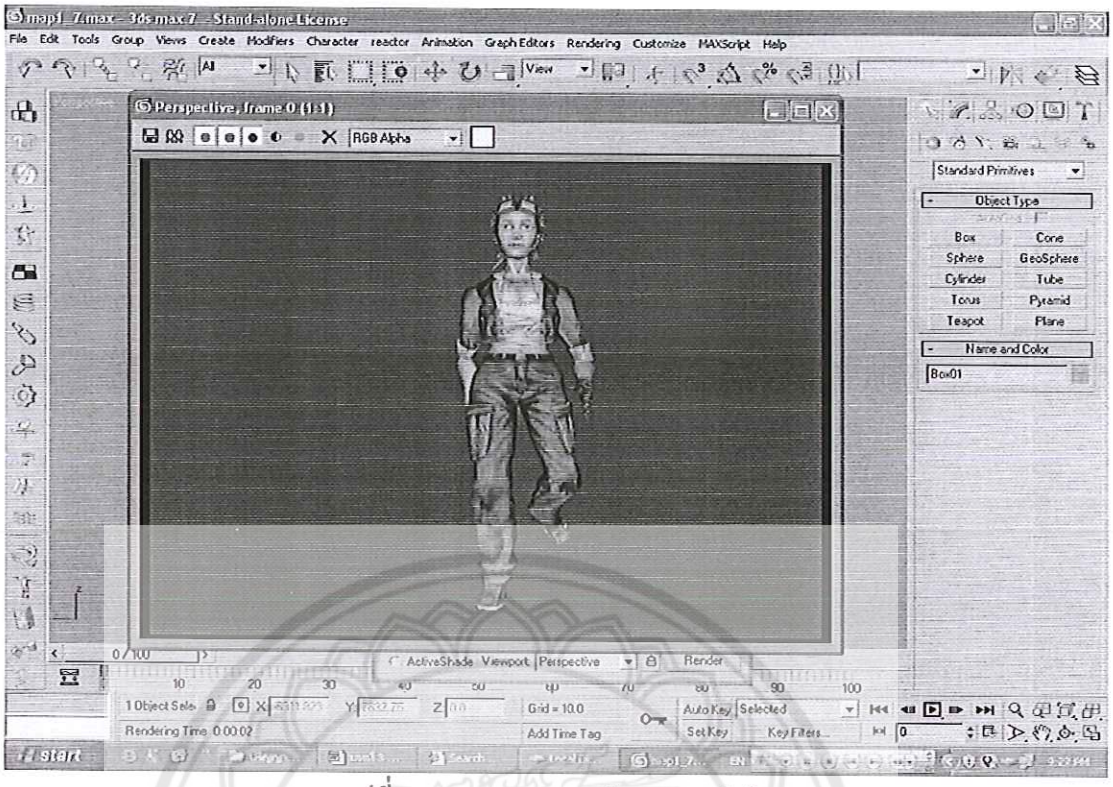
รูปที่ 3.1 การออกแบบแผนที่ในเกม

เมื่อออกแบบเสร็จก็เป็นขั้นตอนการสร้างโมเดลด้วยโปรแกรม 3ds MAX7 ดังรูปที่ 3.2



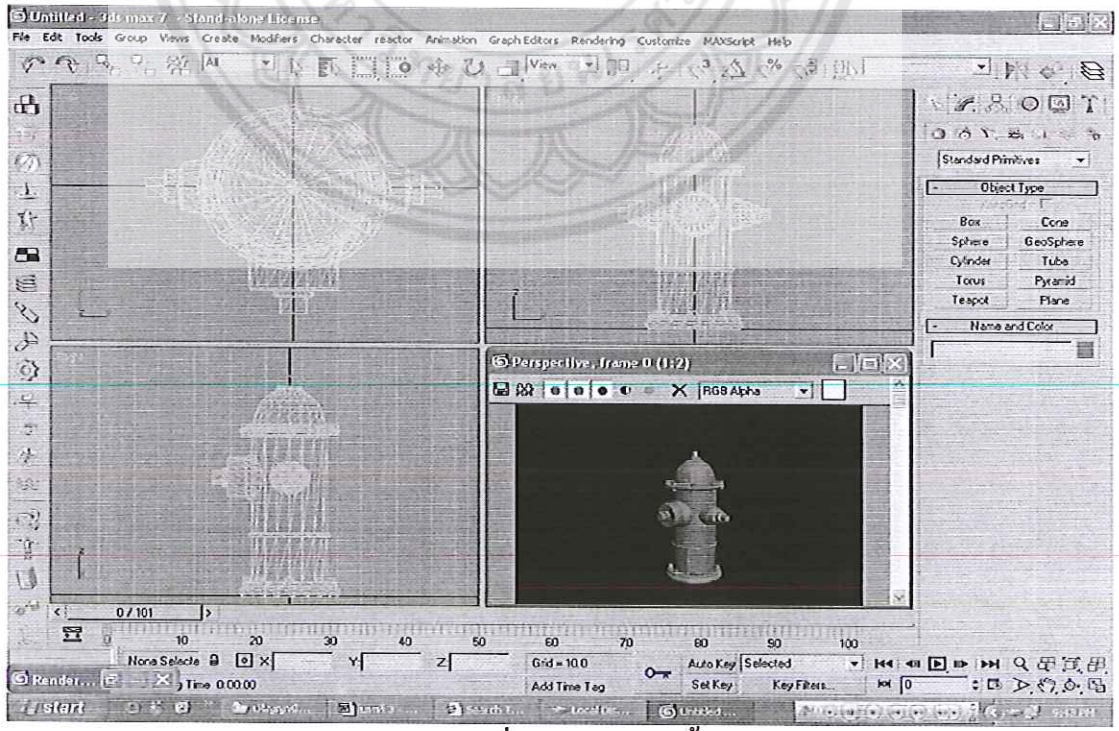
รูปที่ 3.2 โมเดลที่ใช้เป็นแผนที่ในเกม

2. โมเดลที่ใช้เป็นตัวละครหลักในเกม โมเดลนี้จะใช้เป็นตัวละครหลักในเกมที่มีการเคลื่อนไหวต่างๆ ตามท่าทางของมนุษย์เพื่อให้เกิดความสมจริงของเกมดัง รูปที่ 3.3

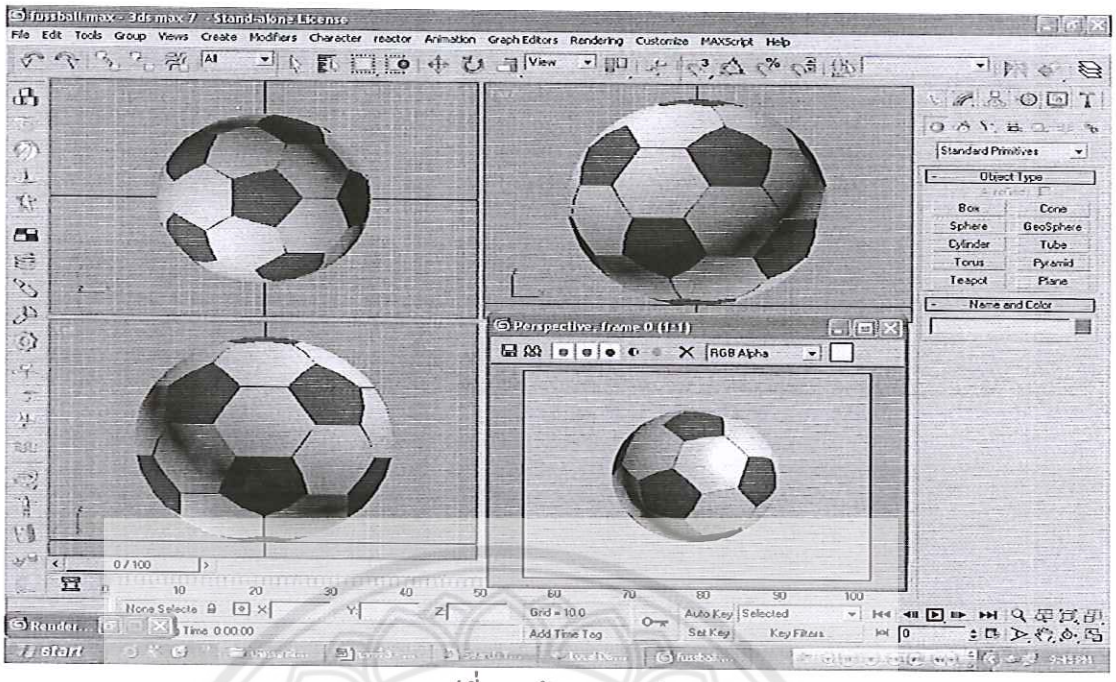


รูปที่ 3.3 การออกแบบตัวละครหลักในเกม

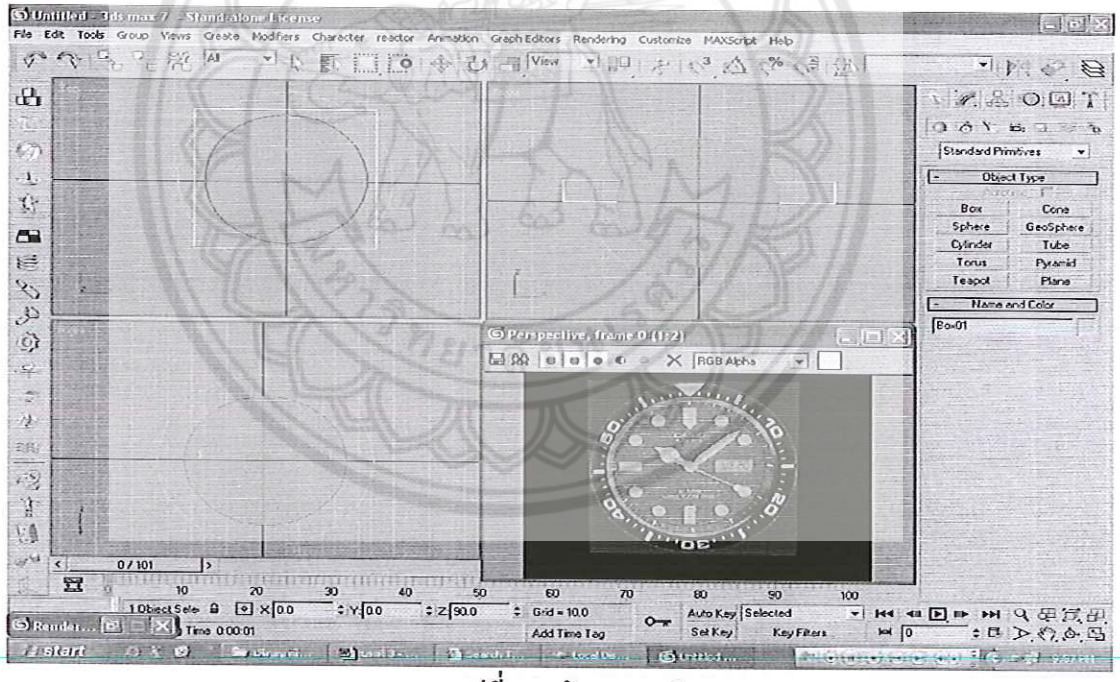
3. โมเดลที่ใช้เป็นสิ่งของต่างๆในเกม โมเดลนี้เราจะใช้เป็นสิ่งของต่างๆที่มีผลต่อเงื่อนไขของเกม ซึ่งมีอยู่ด้วยกัน 3 ชนิดคือท่อน้ำ ดังรูปที่ 3.4 ลูกฟุตบอล ดังรูปที่ 3.5และนาฬิกาดังรูปที่ 3.6



รูปที่ 3.4 โมเดลท่อน้ำ



รูปที่ 3.5 โมเดลลูกฟุตบอล

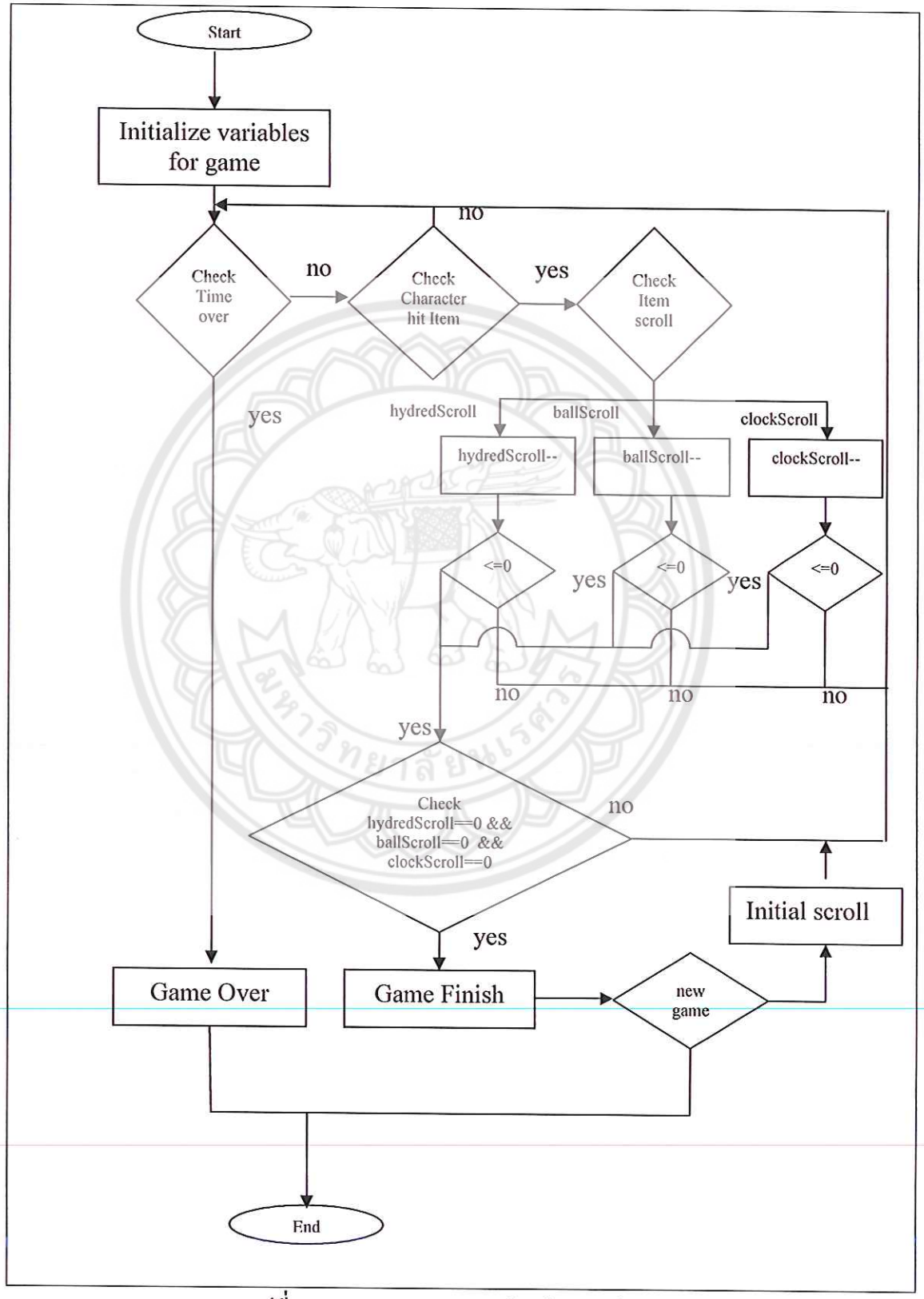


รูปที่ 3.6 โมเดลนาฬิกา

3.2 ขั้นตอนการออกแบบเงื่อนไขต่างๆในการเล่นเกม

ขั้นตอนนี้จะเป็นขั้นตอนที่สำคัญเพราะความสนุกสนานของเกมจะอยู่ตรงที่เงื่อนไขต่างๆ ในการเล่นเกมซึ่งเกมสามมิตินี้ได้มีการออกแบบรูปแบบหรือเงื่อนไขการเล่นดังนี้ เกมสามมิตินี้เป็นลักษณะเกมที่ค้นหาทางออก เมื่อผู้เล่นต้องการจะชนะเกมนี้จะต้องหาทางออกให้เจอ และจะออกได้ก็ต่อเมื่อผู้เล่นมีการเก็บสิ่งของต่างๆในเกมให้ครบตามที่กำหนด และเก็บให้ทันเวลาที่กำหนดด้วย

หากผู้เล่นหาทางออกจากเกมนี้ไม่ทันเวลาจะถือว่าผู้เล่นเป็นฝ่ายแพ้ สามารถอธิบายด้วย Flow Chart แสดงกติกาในการเล่นเกม ดังรูปที่ 3.7



รูปที่ 3.7 Flow Chart แสดงกติกาในการเล่นเกม

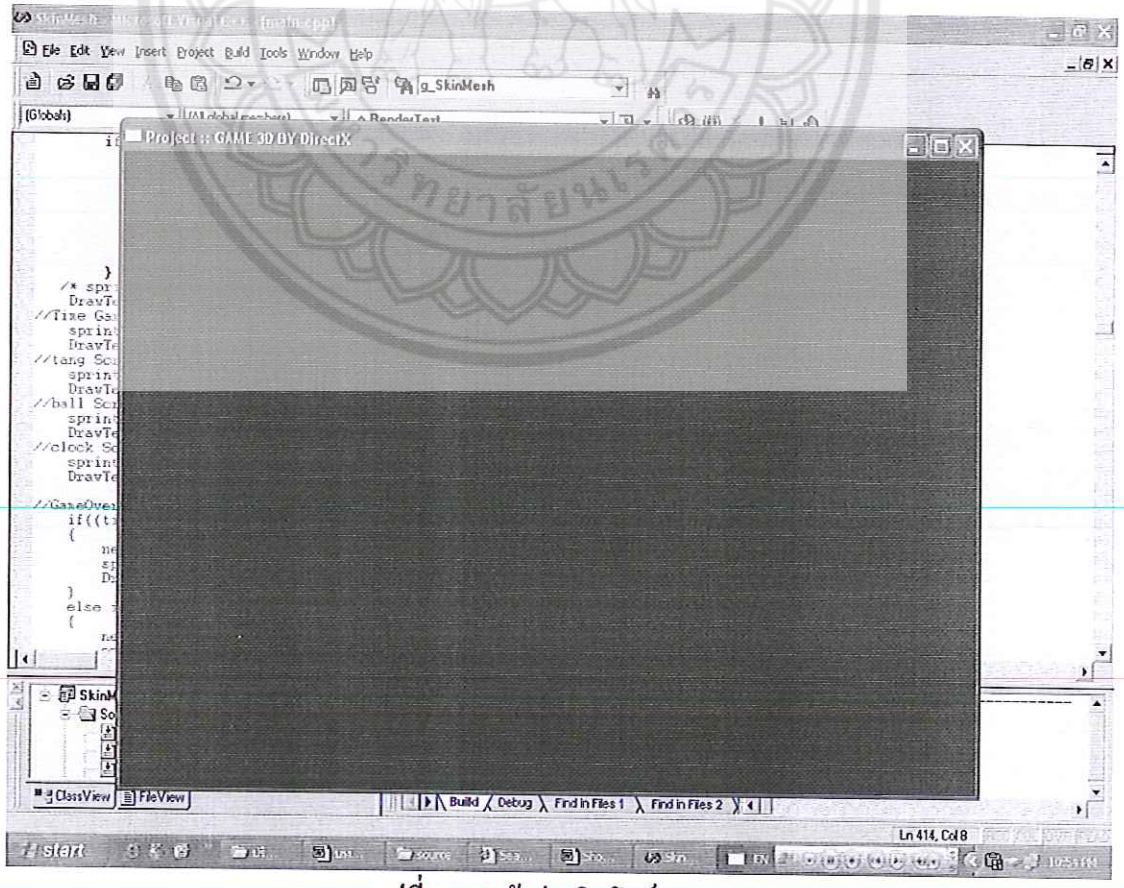
3.3 ขั้นตอนการเขียนโปรแกรม

ขั้นตอนนี้เป็นส่วนที่สำคัญที่สุดในการสร้างเกมเพราะต้องมีการเขียนโปรแกรมเพื่อให้เกมเป็นไปตามเงื่อนไขที่ได้กำหนดไว้ และจะต้องมีการจัดการเกี่ยวเรื่องของ โมเดล เสียง การควบคุมเกม ความผิดพลาดที่อาจเกิดขึ้นระหว่างการทำงานของโปรแกรม และอื่นๆอีกมาก

ในส่วนการเขียนโปรแกรมจะแบ่งขั้นตอนการพัฒนาเป็นดังนี้

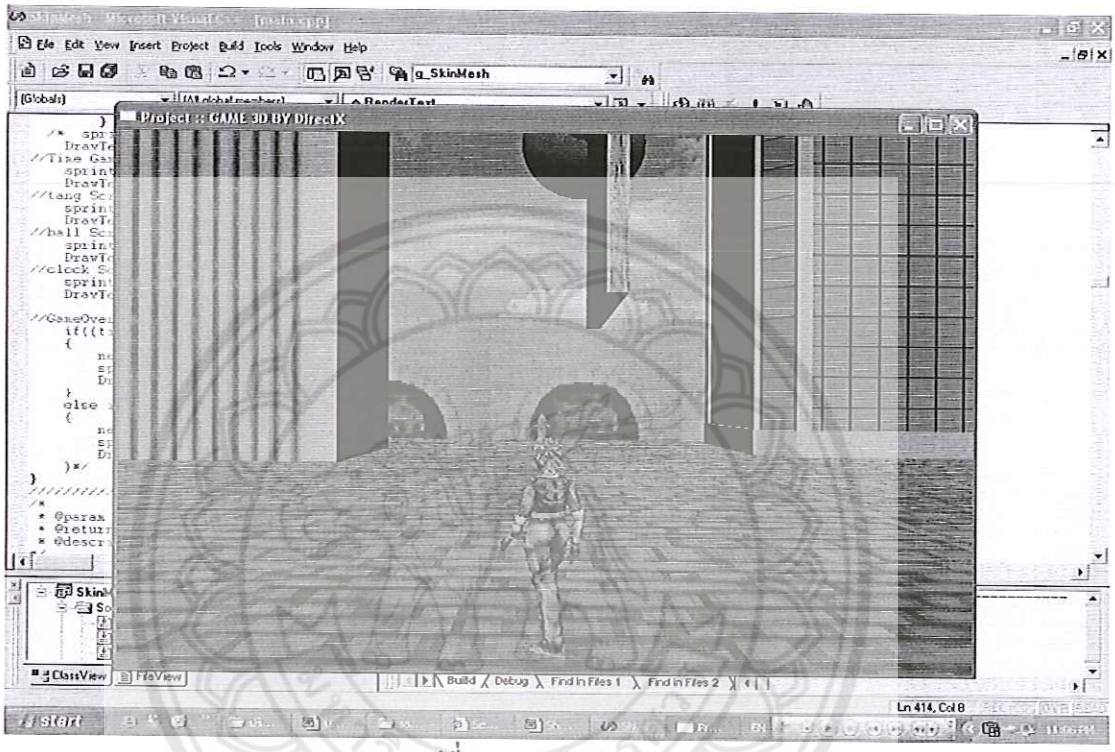
1. การเขียนโปรแกรมเพื่อสร้างหน้าต่างวินโดว์ในการเล่นเกม
2. การเขียนโปรแกรมเพื่อโหลดโมเดลเข้ามาใช้ในเกม
3. การเขียนโปรแกรมเพื่อควบคุมการเคลื่อนที่และควบคุมการมองเห็นในเกม
4. การเขียนโปรแกรมเพื่อตรวจสอบการชนกันของวัตถุในเกม
5. การเขียนโปรแกรมเพื่อแสดงข้อความรูปแบบสองมิติในเกมสามมิติ
6. การเขียนโปรแกรมเพื่อโหลดเสียงเข้ามาใช้ประกอบในการเล่นเกม

1. การเขียนโปรแกรมเพื่อสร้างหน้าต่างวินโดว์ในการเล่นเกม ในส่วนนี้จะเป็นส่วนที่ใช้ในการสร้างหน้าต่างหลักของเกม ซึ่งมีลักษณะเป็นแบบหน้าต่างวินโดว์ขนาดของหน้าต่างวินโดว์คือ 800 x 600 ไม่ใช่แบบแสดงผลเต็มจอ ดังรูปที่ 3.8



รูปที่ 3.8 หน้าต่างวินโดว์ของเกม

2. การเขียนโปรแกรมเพื่อโหลดโมเดลเข้ามาใช้ในเกม ในส่วนนี้จะเป็นการเขียนโปแกรมเพื่อโหลดโมเดลเข้ามาใช้ในเกม ซึ่งจะเป็นโมเดลสามมิติ มีอยู่ด้วยกัน 2 ลักษณะคือ โมเดลที่ไม่มีการเคลื่อนไหว และโมเดลที่มีการเคลื่อนไหว ในเกมนี้จะมีการใช้โมเดลทั้ง 2 แบบนี้ในการพัฒนาเกม โมเดลที่จะทำการโหลดมีอยู่ด้วย 5 ชนิดคือ โมเดลตัวละครหลักใช้ชื่อว่า tiny โมเดลแผนที่ใช้ชื่อว่า maps โมเดลสิ่งของที่ใช้ในการเล่นเกม มีชื่อว่า hydred, football2 และ clock3 ดังรูป 3.9



รูปที่ 3.9 การโหลดโมเดล

3. การเขียนโปรแกรมเพื่อควบคุมการเคลื่อนที่และควบคุมการมองเห็น**ในเกม** ในขั้นตอนนี้จะกำหนดให้มีการควบคุมการเคลื่อนที่ในเกมด้วยคีย์บอร์ดเท่านั้น โดยจะกำหนดให้มีการใช้งานได้เพียง 5 ปุ่มกด เท่านั้น คือ

1. ปุ่มลูกศรชี้ไปทางขวา ควบคุมการหมุนของตัวละครหลักและมุมมองของการมองเห็นไปทางขวามือ

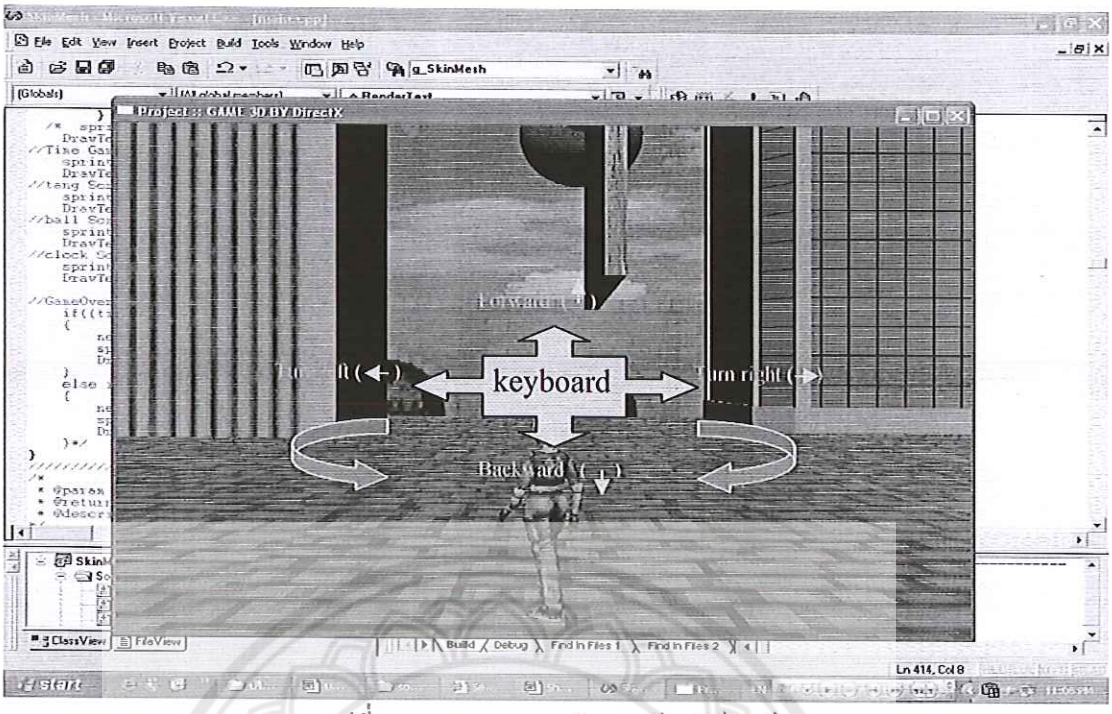
2. ปุ่มลูกศรชี้ไปทางซ้าย ควบคุมการหมุนของตัวละครหลักและมุมมองของการมองเห็นไปทางซ้ายมือ

3. ปุ่มลูกศรชี้ขึ้น ควบคุมการเดินของตัวละครหลักและมุมมองของการมองเห็นไปข้างหน้า

4. ปุ่มลูกศรชี้ลง ควบคุมการเดินของตัวละครหลักและมุมมองของการมองเห็นไปข้างหลัง

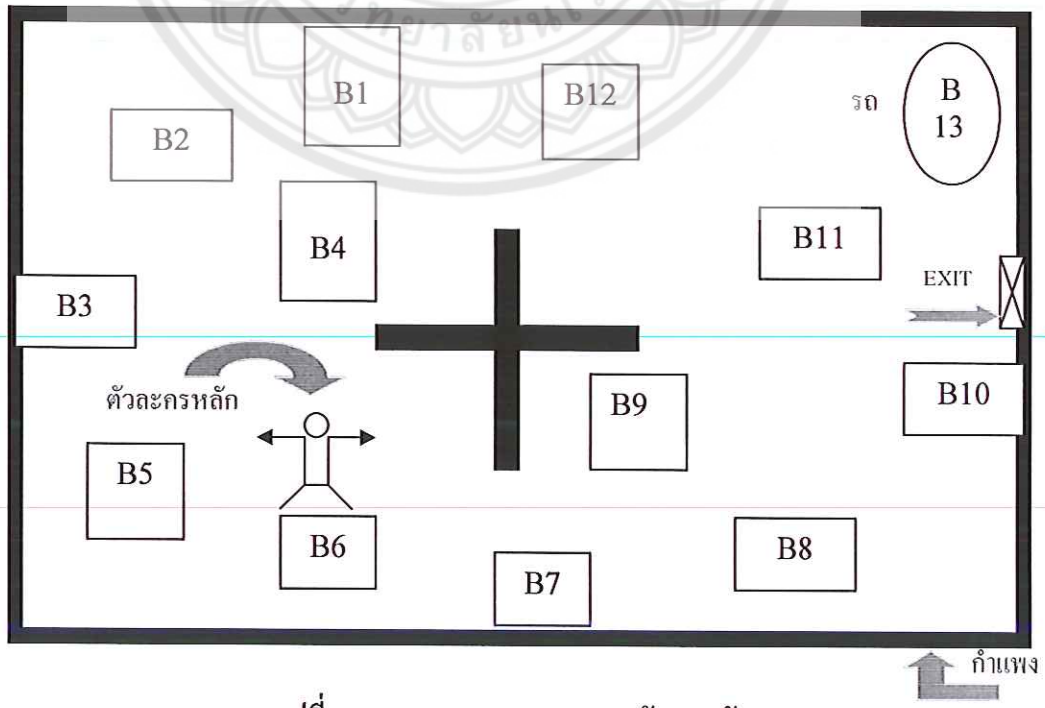
5. ปุ่ม F11 เป็นปุ่มเริ่มเล่นเกมใหม่ขณะที่เกมกำลังดำเนินการเล่นอยู่

6. ปุ่ม F12 เป็นปุ่มสำหรับเริ่มการเล่นเกมใหม่

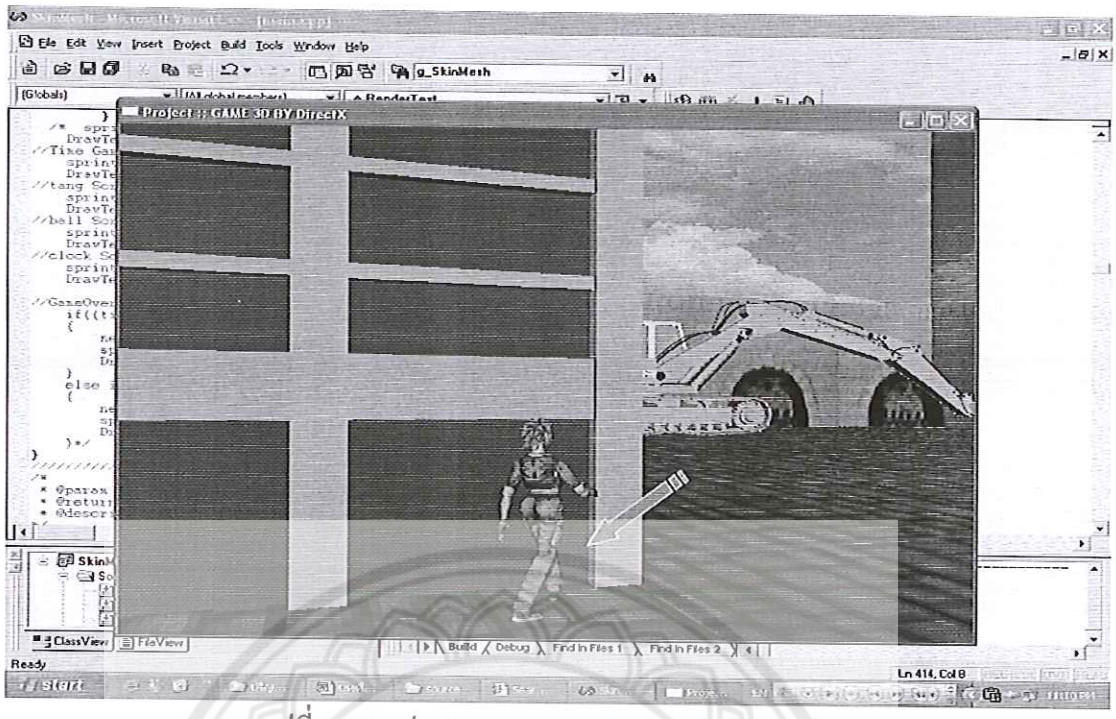


รูปที่ 3.10 การควบคุมโมเดลด้วยคีย์บอร์ด

4. การเขียนโปรแกรมเพื่อตรวจสอบการชนกันของวัตถุในเกม ในขั้นนี้จะเป็นส่วนของการตรวจสอบการชนกันของวัตถุ เมื่อตัวละครหลักมีการเคลื่อนที่ไปในแผนที่ซึ่งมีสิ่งกีดขวางต่างๆ มากมาย เพื่อให้เกิดความสมจริงของเกมตัวละครจะต้องไม่สามารถเดินทะลุผ่านสิ่งกีดขวางต่างๆ หรือกำแพงได้ แสดงได้ดังรูปที่ 3.11 และรูปที่ 3.12

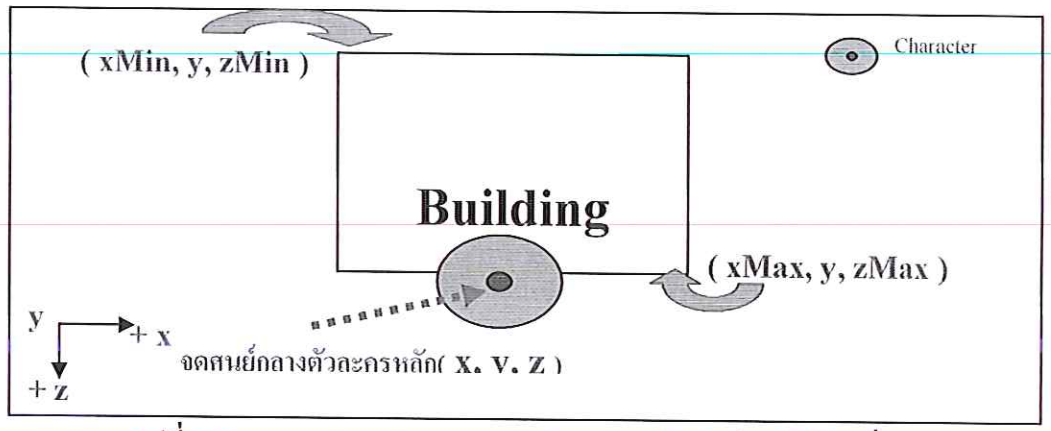


รูปที่ 3.11 การตรวจสอบการชนกันของวัตถุ



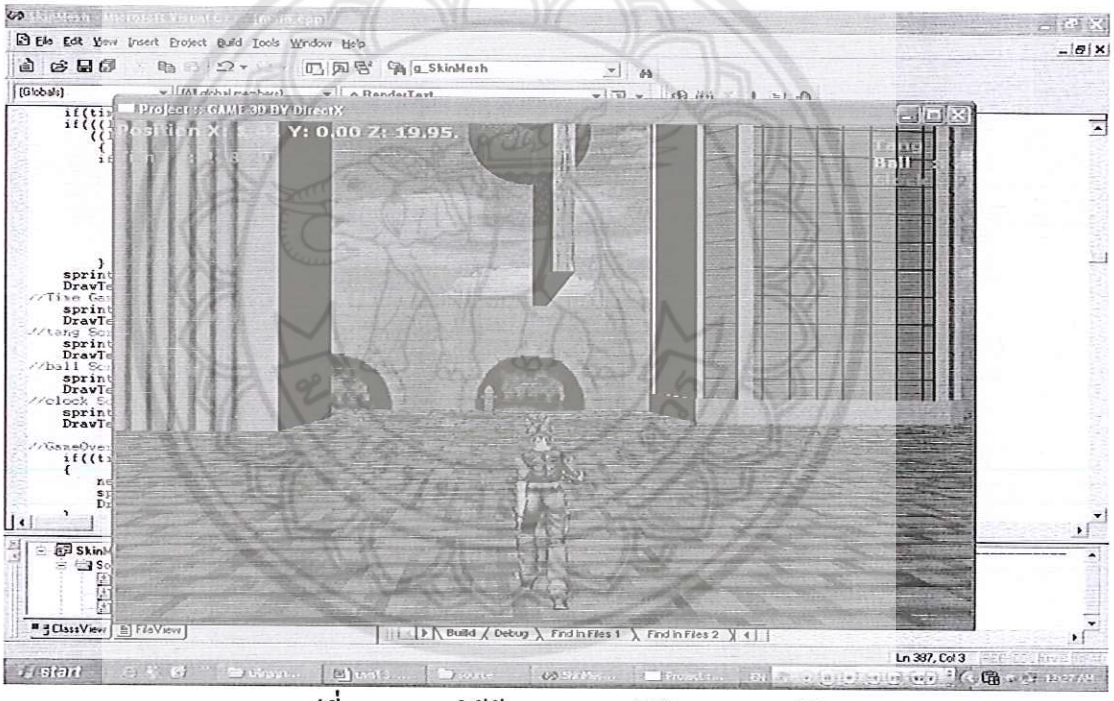
รูปที่ 3.12 รูปแสดงการตรวจสอบการชนกันของวัตถุ

ในรูปที่ 3.13 จะเป็นการแสดงการตรวจการชนของตัวละครหลักกับตึกหรือวัตถุอื่นๆ ซึ่งต่อไปนี้จะเรียกว่า Building โดยจะมีจุดศูนย์กลางตัวละครหลัก คือจุด (x, y, z) เป็นจุดที่ใช้ในการตรวจสอบการชนกับตึกหรือวัตถุต่างๆ จุดศูนย์กลางตัวละครหลัก จะต้องไม่สามารถเข้าไปในเขตของ Building ได้ ซึ่ง Building จะมีพิกัดเพื่อกำหนดขอบเขตของ Building คือ พิกัดซ้ายบน (xMin, y, zMin) และพิกัดขวาล่าง (xMax, y, zMax) จะมี xMin, xMax เป็นตำแหน่งของพิกัด x ส่วน y เป็นตำแหน่งของพิกัด y ซึ่งจะให้เป็นค่าคงที่เพราะตัวละหลักจะไม่มีการเคลื่อนที่ในแนวแกน y และ zMin, zMax เป็นตำแหน่งของพิกัด z ตัวละครหลักจะไม่สามารถเข้าไปในเขต Building ได้เมื่อจุดศูนย์กลางตัวละครหลัก มีค่ามากกว่าพิกัด xMin และ zMin และมีค่าน้อยกว่า xMax และ zMax ดังรูปที่3.11



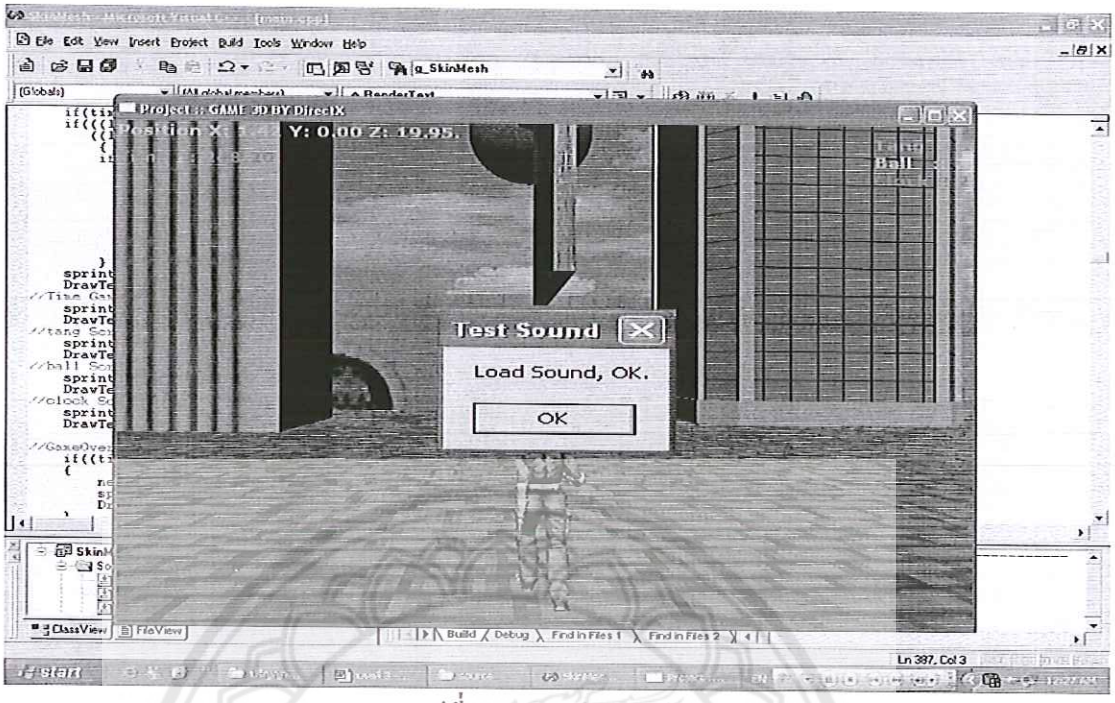
รูปที่ 3.13 การตรวจสอบการชนของตัวละครหลักกับตึกหรือวัตถุอื่นๆ

5. การเขียนโปรแกรมแสดงข้อความรูปแบบสองมิติในเกมสามมิติ ในขั้นตอนนี้จะเป็นส่วนในการเขียนโปรแกรมเพื่อสร้างตัวอักษรขึ้นเพื่อใช้ในเกม ตัวอักษรเหล่านี้มีความสำคัญเช่นกัน ซึ่งจะช่วยเพิ่มสีสันให้กับเกม ทำให้เกมน่าเล่นมากขึ้น และยังเป็นส่วนที่ช่วยให้ผู้เล่นเกมสามารถสื่อสารกับเกมได้รู้เรื่อง จากรูปที่ 3.14 จะเห็นว่าในเกมจะมีการเพิ่มการแสดงผลข้อความขึ้น ซึ่งประกอบไปด้วย ข้อความที่แสดงตำแหน่งของตัวละครหลัก ซึ่งปรากฏอยู่บนมุมซ้ายบนสุด ข้อความแสดงเวลาที่เหลือในการเล่นเกม อยู่มุมซ้ายบนลองลงมาจากข้อความที่แสดงตำแหน่งของตัวละครหลัก ข้อความแสดงจำนวนสิ่งของที่ยังไม่ได้เก็บ คือ ท่อน้ำ(Tang) ลูกฟุตบอล (Ball) นาฬิกา (Clock) ซึ่งอยู่บนมุมขวาบน และยังสามารถแสดงข้อความอื่นๆได้อีก อย่างเช่นเมื่อมีการเล่นเกมแบบหลายผู้เล่นก็จะมีการสื่อสารกันระหว่างผู้เล่นก็จะใช้ข้อความแบบสองมิติในการสื่อสารกัน



รูปที่ 3.14 การใช้ข้อความสองมิติในเกมสามมิติ

6. การเขียนโปรแกรมโหลดเสียงเข้ามาใช้ประกอบในการเล่นเกม ในขั้นตอนนี้เป็นส่วนในการโหลดเสียงเข้ามาใช้ในเกม เสียงก็มีความสำคัญเช่นกัน เสียงจะช่วยเพิ่มความน่าเล่นของเกมได้มากขึ้นเสียงที่ใช้ในเกมจะใช้ทั้งเสียงที่เป็นนามสกุล *.wav และเสียงที่เป็นนามสกุล *.mid ซึ่งเสียงที่ใช้ในเกมนี้มีชื่อว่า 002.mid จะเป็นเสียงเพลงประกอบระหว่างการเล่นเกมและเสียงที่เป็นนามสกุล *.wav จะใช้เล่นเมื่อมีการเก็บสิ่งของ จากรูปที่ 3.15 เป็นการทดสอบการโหลดเสียง หากการทดสอบผ่านจะมีบล็อกแสดงข้อความบอกว่าโหลดเสียงผ่าน ถ้าไม่ผ่านก็จะมีบล็อกแสดงข้อความบอกว่าไม่สามารถโหลดเสียงได้ ดังรูปที่ 3.15



รูปที่ 3.15 การโหลดเสียง

บทที่ 4
ผลการทดสอบโปรแกรมและวิเคราะห์ผล

4.1 จุดประสงค์ของการทดสอบโปรแกรม

1. เพื่อทำการทดสอบโปรแกรมที่ได้สร้างขึ้นเป็นไปตามเงื่อนไขหรือตามที่ได้ออกแบบไว้หรือไม่

2. เพื่อทดสอบประสิทธิภาพการทำงานของโปรแกรม

3. เพื่อทดสอบหาข้อผิดพลาดที่เกิดขึ้นกับโปรแกรมเพื่อจะได้นำข้อผิดพลาดที่เกิดขึ้นมาวิเคราะห์และหาทางแก้ไขให้โปรแกรมมีความผิดพลาดเกิดขึ้นน้อยที่สุด

4.2 ขั้นตอนการทดสอบการทำงานของโปรแกรม

1. ติดตั้งโปรแกรม Visual C++ เวอร์ชัน 6.0

2. ติดตั้งโปรแกรม DirectX SDK เวอร์ชัน 8.0 และ DirectX Runtime เวอร์ชัน 8.0 ขึ้นไป

3. ตั้งค่าการทำงานเพื่อให้โปรแกรม Visual C++ กับ โปรแกรม DirectX SDK สามารถทำงานร่วมกันได้

4. เปิดโปรแกรมเกมเพื่อทดสอบการทำงานของเกมว่าเป็นไปตามเงื่อนไขที่กำหนดไว้หรือไม่

5. ตรวจสอบหาข้อผิดพลาดที่เกิดขึ้นในโปรแกรม

6. นำข้อผิดพลาดที่เกิดขึ้นมาวิเคราะห์หาสาเหตุและทำการปรับปรุงแก้ไขโปรแกรม

4.3 ผลการทดสอบโปรแกรม

โปรแกรมเกมนี้ได้ทำการกำหนดให้มีการควบคุมเกมผ่านทางคีย์บอร์ดเท่านั้น เมาส์จะไม่สามารถใช้ในเกมได้ โดยจะมีการกำหนดปุ่มการทำงานดังนี้

Arrow Right - ใช้สำหรับควบคุมการหมุนตัวของตัวละครและควบการหมุนมุมมองของตัวละครไปทางขวามือ

Arrow Left - ใช้สำหรับควบคุมการหมุนตัวของตัวละครและควบการหมุนมุมมองของตัวละครไปทางซ้ายมือ

Arrow Up - ใช้สำหรับควบคุมการเคลื่อนที่ของตัวละครไปข้างหน้า

Arrow Down - ใช้สำหรับควบคุมการเคลื่อนที่ของตัวละครเพื่อถอยหลัง

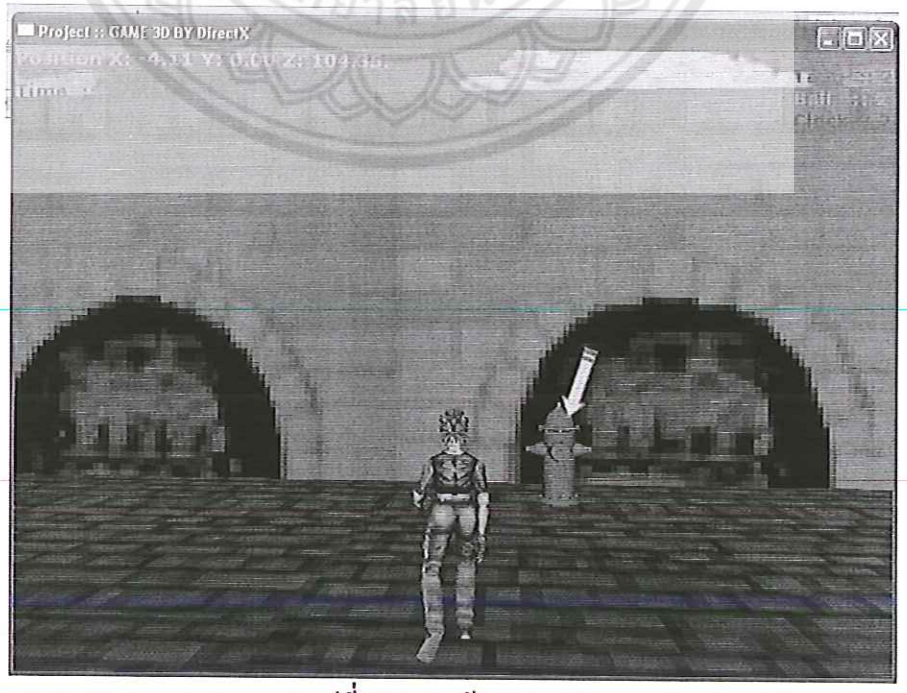
F11 - ทำการเริ่มเล่นเกมใหม่ขณะที่เกมกำลังดำเนินการเล่นอยู่

F12 - ทำการเริ่มเล่นเกมใหม่เมื่อจบเกมแล้ว

เมื่อเริ่มเปิดโปรแกรมก็จะมีเมนูของเกมให้เลือกคือเข้าเล่นเกมและออกจากเกม ดังรูปที่ 4.1

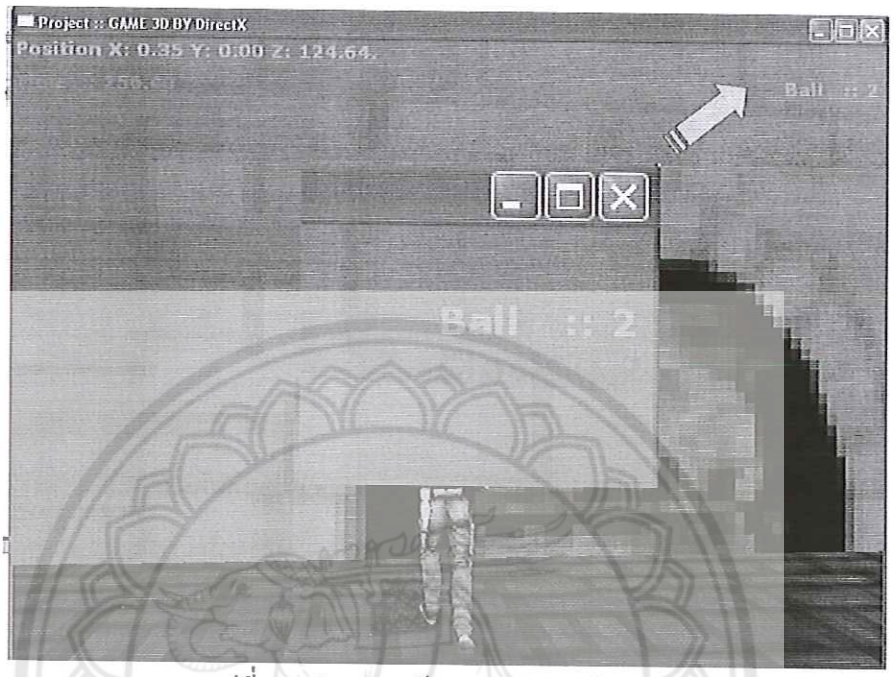
รูปที่ 4.1 เมนูเกมเมื่อเริ่มเปิดโปรแกรม

เมื่อทำการเลือกเมนูเข้าเล่นเกม เกมก็จะเริ่มทำงานและเริ่มจับเวลาและสุ่มตำแหน่งการเกิดของสิ่งของ ผู้เล่นจะต้องหาทางออกให้ทันก่อนเวลาหมด และจะออกได้ก็ต่อเมื่อทำเงื่อนไขให้ครบก่อน คือจะต้องเก็บ Tang, Ball และ Clock ให้ครบอย่างละ 2 อย่างเสียก่อนจะเก็บของชิ้นไหนก่อนก็ได้ จากรูปที่ 4.2 เป็นตัวอย่างการค้นหา Tang



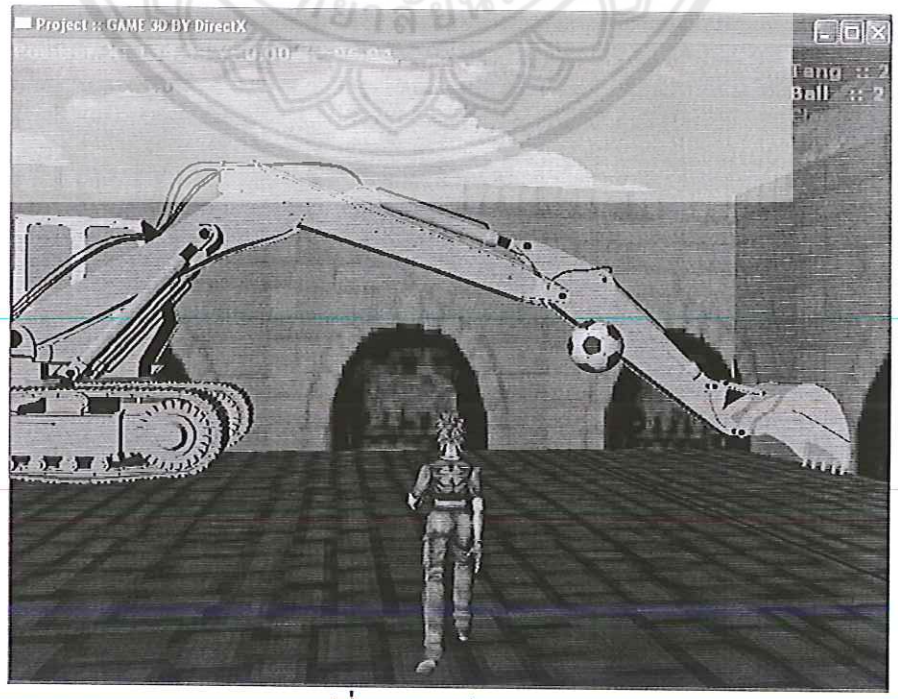
รูปที่ 4.2 การค้นหา Tang

เมื่อค้นหา Tang พบแล้วก็เดินเข้าไปเก็บเพื่อให้ครบทั้ง 2 อัน จะเห็นได้จากจำนวนแต้มที่โชว์อยู่บนขวามือของหน้าต่างเกมแต้มของ Tang ก็จะลดลงและแต้มที่ปรากฏอยู่คือจำนวนสิ่งของที่เรายังไม่ได้เก็บ ดังรูปที่ 4.3



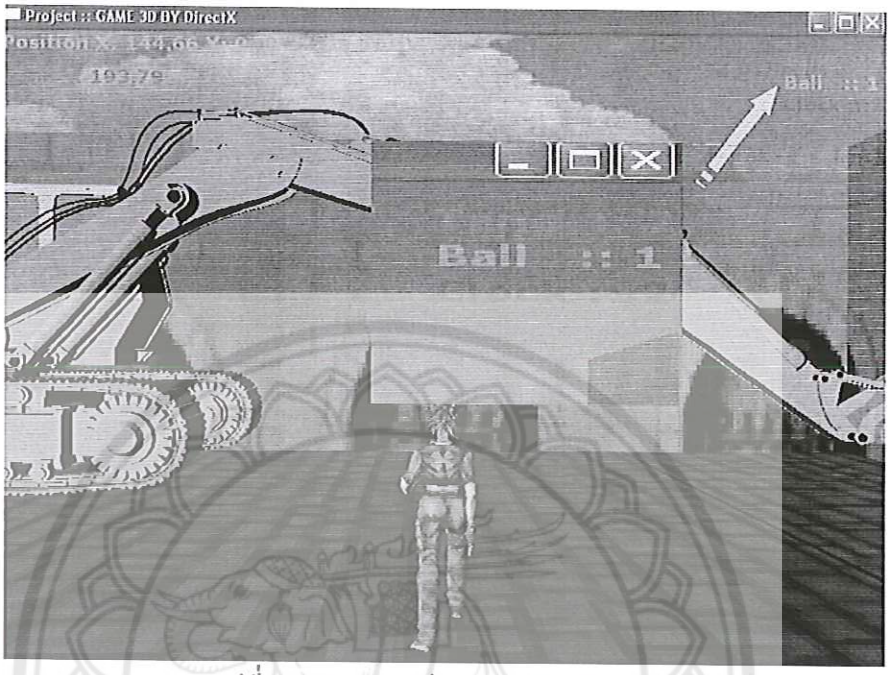
รูปที่ 4.3 จำนวนแต้มของ Tang หลังจากทำการเก็บ

การค้นหาสิ่งของที่ชื่อว่า Ball แสดงได้ดังรูปที่ 4.4



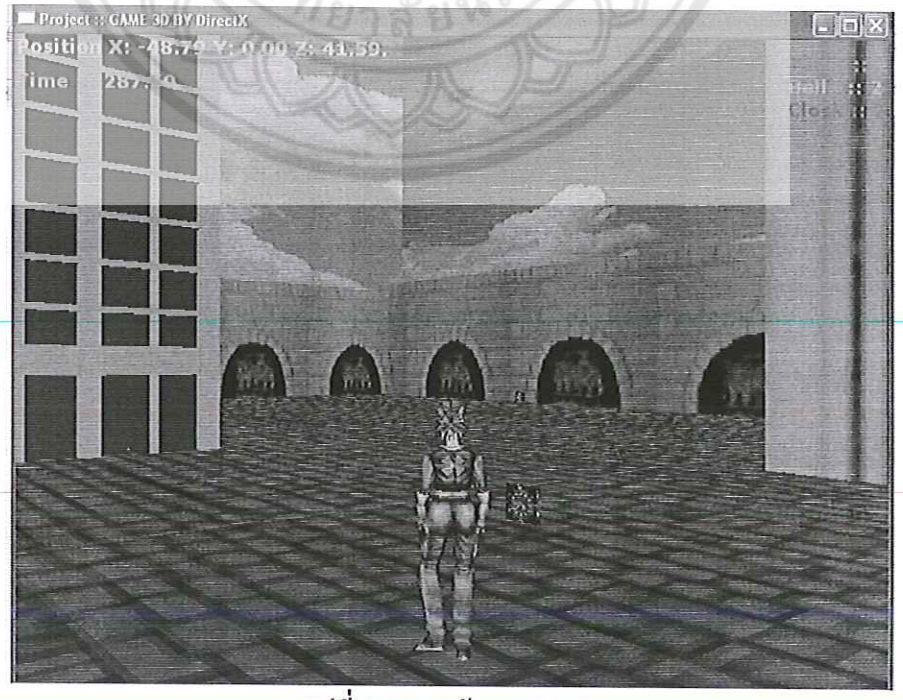
รูปที่ 4.4 การค้นหา Ball

เมื่อค้นหา Ball พบแล้วก็เดินเข้าไปเก็บเพื่อให้ครบทั้ง 2 อัน จะเห็นได้จากจำนวนแต้มที่โชว์อยู่บนขวามือของหน้าต่างเกมแต้มของ Ball ก็จะลดลงและแต้มที่ปรากฏอยู่คือจำนวนสิ่งของที่เรายังไม่ได้เก็บ ดังรูปที่ 4.5



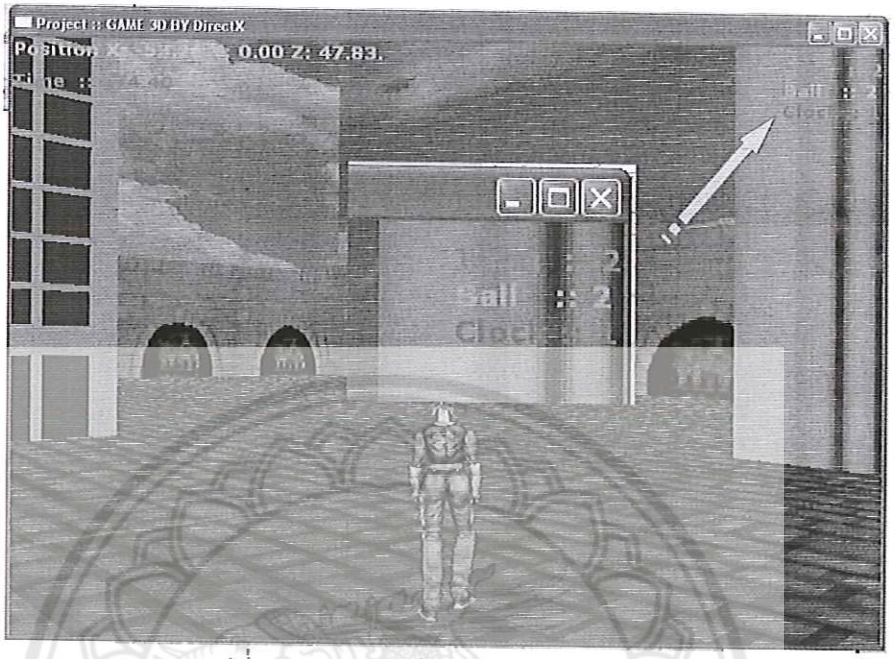
รูปที่ 4.5 จำนวนแต้มของ Ball หลังจากทำการเก็บ

การค้นหาสิ่งของที่ชื่อว่า Clock แสดงได้ดังรูปที่ 4.6



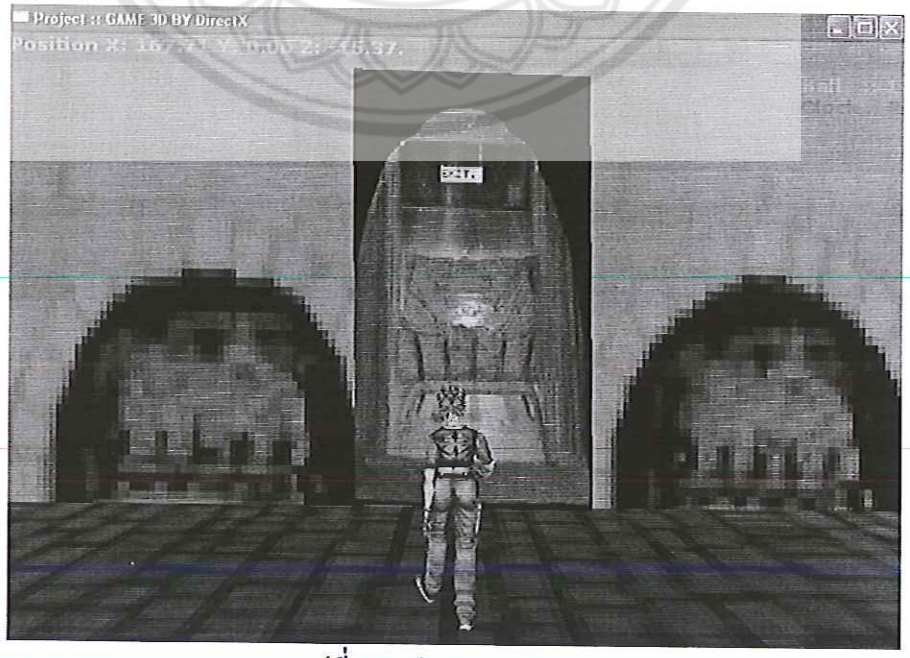
รูปที่ 4.6 การค้นหา Clock

เมื่อค้นหา Clock พบแล้วก็เดินเข้าไปเก็บเพื่อให้ครบทั้ง 2 อัน จะเห็นได้จากจำนวนแต้มที่โชว์อยู่บนขวามือของหน้าต่างเกม แต้มที่ปรากฏอยู่คือจำนวนสิ่งของที่เรายังไม่ได้เก็บ ดังรูปที่ 4.7

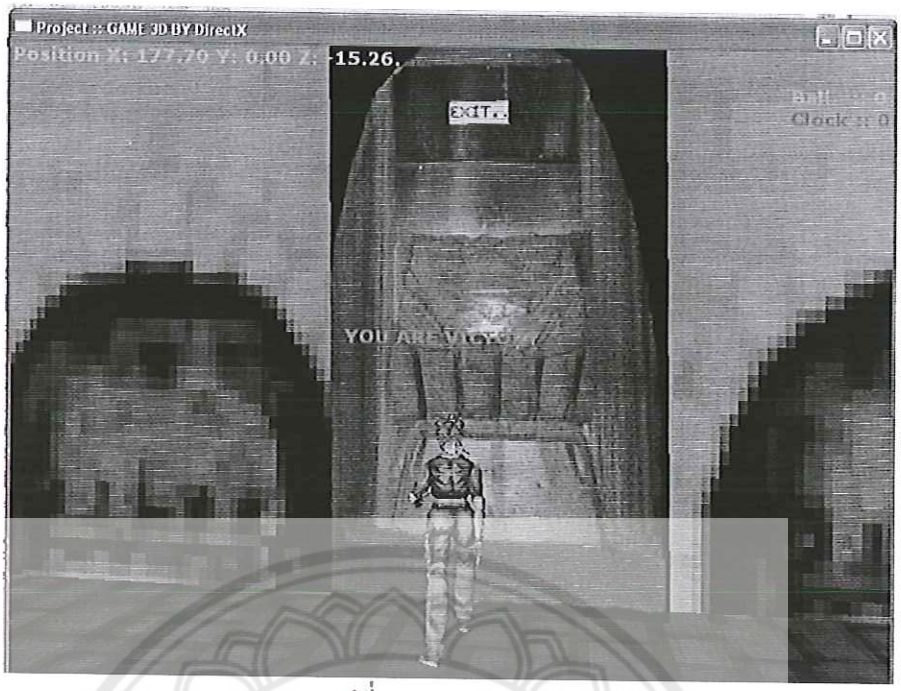


รูปที่ 4.7 จำนวนแต้มของ Clock หลังจากทำการเก็บ

เมื่อทำการเก็บสิ่งของครบตามที่กำหนดแล้วและเวลายังไม่หมดก็เดินไปยังที่ประตูทางออก ดังรูปที่ 4.8 เพื่อจบเกมและจะถือว่าผู้เล่นเป็นฝ่ายชนะเกม จากรูปที่ 4.9 เป็นการแสดงผลของการชนะเกม

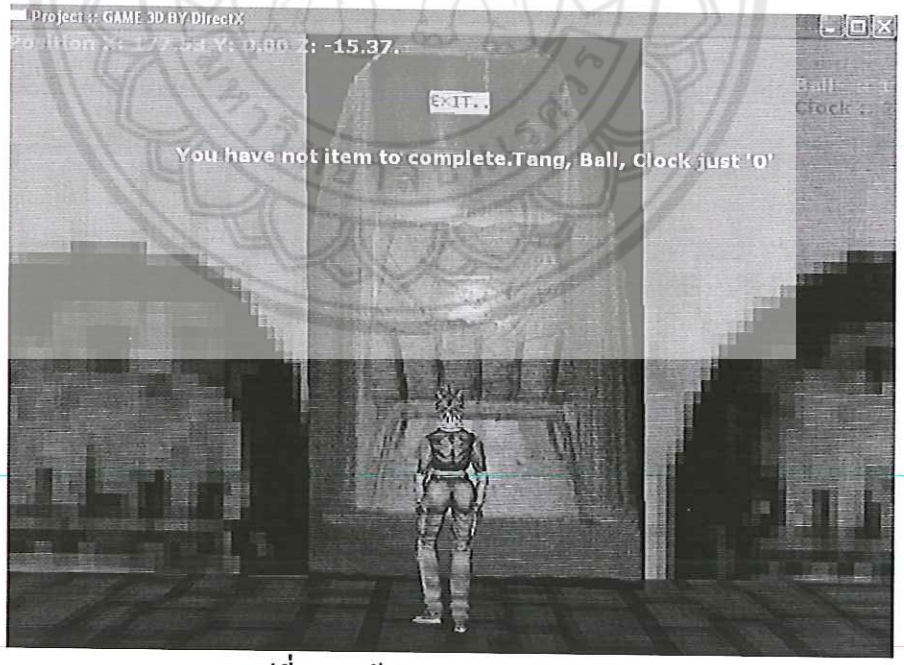


รูปที่ 4.8 ประตูทางออกเกม



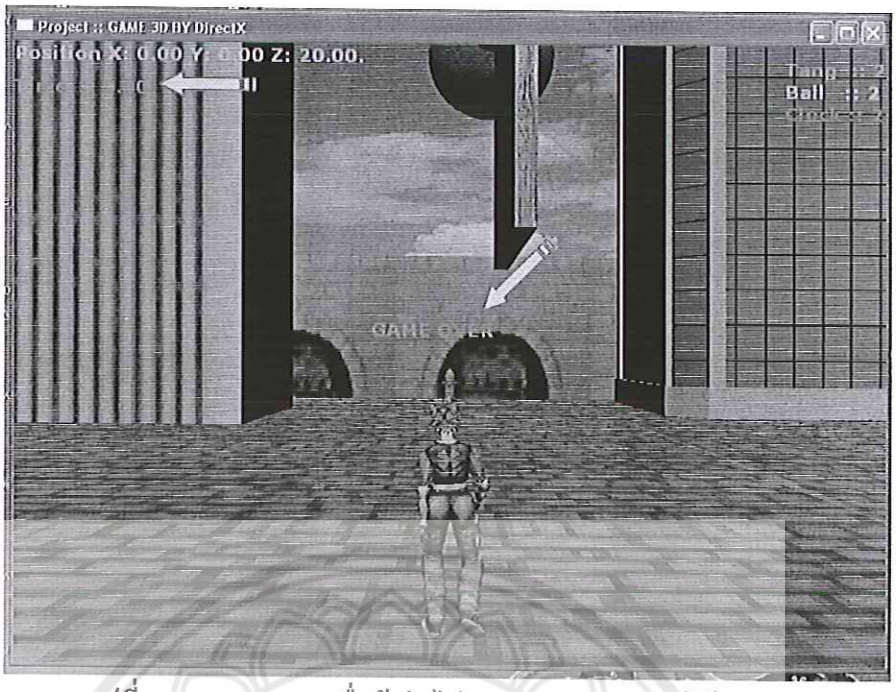
รูปที่ 4.9 การชนะเกม

เมื่อผู้เล่นเจอทางออกและพยายามจะเข้าประตู ขณะที่ผู้เล่นยังไม่สามมารถเก็บของได้ครบตามเงื่อนไขที่กำหนด จะมีข้อความบอกว่า "You have not item to complete." ดังรูปที่ 4.10



รูปที่ 4.10 ผู้เล่นพยายามเล่นผิดกติกา

เมื่อผู้เล่นไม่สามารถเล่นเกมให้จบทันเวลาได้เกมก็จะหยุดลง และจะมีข้อความบอกว่า "Game Over" เพื่อบอกว่าจบเกม ดังรูปที่ 4.11



รูปที่ 4.11 การจบเกมเมื่อผู้เล่นไม่สามารถหาทางออกได้ทันเวลา

4.4 วิเคราะห์ผล

ผลการทดสอบโปรแกรมปรากฏว่าเกมที่ได้พัฒนาได้ทำงานเป็นไปตามเงื่อนไขที่ได้กำหนดไว้ คือ เมื่อผู้เล่นเลือกเมนูเข้าเล่นเกมก็จะเริ่มทำการจับเวลาทันที ผู้เล่นจะต้องค้นหาสิ่งของสามสิ่งให้ครบตามที่กำหนดไว้คือ Tang 2 อัน Ball 2 ลูก และClock 2 เรือน เมื่อหาครบแล้วจะต้องหาทางออกก่อนเวลาจะหมด

เมื่อผู้เล่นหาทางออกได้ทันเวลาและทำตามเงื่อนไขที่กำหนดไว้ผู้เล่นก็จะเป็นฝ่ายชนะเกม หากผู้เล่นหาทางออกไม่ทันเวลาก็จะถือว่าผู้เล่นเป็นฝ่ายแพ้

บทที่ 5
สรุปผลและข้อเสนอแนะ

ในบทนี้จะเป็นการสรุปผลของโครงการนี้ ซึ่งจะกล่าวถึงการสรุปผลของโครงงาน ปัญหาในการทำงาน ข้อเสนอแนะและแนวทางในการพัฒนา เพื่อเป็นประโยชน์สำหรับผู้สนใจจะพัฒนาโครงงานนี้ต่อไป

5.1 สรุปผล

1. โปรแกรมที่พัฒนาเป็นเกมสามมิติ ซึ่งพัฒนาโดยโปรแกรม Microsoft Visual C++ เวอร์ชัน 6.0 และ DirectX SDKเวอร์ชัน 8.0 ซึ่งเป็นส่วนที่ช่วยในการแสดงผลภาพทั้งสองมิติและสามมิติ ในเรื่องของระบบเสียงประกอบการเล่นเกม การควบคุมตัวละครในเกมและการสร้างโมเดลด้วยโปรแกรมสร้างโมเดลสามมิติคือ 3ds max 7 การสร้างภาพและตกแต่งภาพด้วยโปรแกรม Photoshop 7.0 การแต่งเสียงด้วย Cool Edit ProV1.2a

2. โปรแกรมที่พัฒนาเป็นแบบเล่นคนเดียว เป้าหมายของเกมคือชัยชนะที่ต้องแข่งขันกับเวลาและก็เงื่อนไขต่างๆในเกม

3. โปแกรมที่พัฒนาเป็นโปรแกรมแบบสามมิติซึ่งจะทำให้สามารถมองเห็นวัตถุในเกมได้ทุกมุม และสามารถเคลื่อนที่ไปยังจุดต่างๆในโลกเสมือนของเกมได้

5.2 ปัญหาในการทำงาน

1. ผู้พัฒนามีความรู้ในการใช้โปแกรม Visual C++ เวอร์ชัน 6.0 ไม่เพียงพอ จำเป็นต้องเสียเวลาในการศึกษาการใช้งาน โปแกรม Visual C++ เวอร์ชัน 6.0 พอสมควร

2. ผู้พัฒนามีความรู้ในการใช้ DirectX SDK เวอร์ชัน 8.0 ไม่เพียงพอ จำเป็นต้องเสียเวลาในการศึกษาการใช้งาน DirectX SDK เวอร์ชัน 8.0 พอสมควร

3. การพัฒนาเกมต้องใช้โมเดลสามมิติซึ่งเป็นส่วนที่ใช้เวลานานพอสมควรเนื่องจากต้องมีการออกแบบโมเดล และยังต้องใช้เวลาในการศึกษาการใช้งานโปรแกรม 3ds max 7 เพื่อใช้ในการสร้างโมเดล

5.3 ข้อเสนอแนะ

1. ก่อนการดำเนินงานในแต่ละขั้นตอนควรจะทำการศึกษาข้อมูลให้เข้าใจมากที่สุด เพื่อให้เกิดข้อผิดพลาดในการทำงานน้อยที่สุด

2. ควรจะดำเนินงานในแต่ละขั้นตอนให้เสร็จก่อนกำหนดจะดีที่สุด เพราะบางขั้นตอนอาจต้องใช้เวลาในการพัฒนามากกว่าเวลาที่กำหนดไว้

3. ควรจะมีการทดสอบการทำงานของโปแกรมแต่ละส่วนเพื่อหาข้อผิดพลาดที่เกิดขึ้น และทำการแก้ไขให้เสร็จเสียก่อน แล้วค่อยนำแต่ละส่วนมาประกอบกันเป็นโปรแกรมที่มีขนาดใหญ่ขึ้น เพื่อจะได้หาข้อผิดพลาดและแก้ไขโปรแกรมได้ง่ายขึ้น

5.4 แนวทางในการพัฒนา

1. เพิ่มแผนที่ในการเล่นเกมให้มากขึ้นเพื่อเพิ่มความสนุกสนานของเกม

2. เพิ่มการควบคุมจากผู้เล่นเกมโดยให้ผู้เล่นสามารถใช้เมาส์ควบคุมการเล่นร่วมกับคีย์บอร์ดหรือจอยสติกได้

3. เพิ่มสิ่งของต่างๆในเกมให้มากขึ้นเพื่อเพิ่มความสนุกสนานของเกม ทำให้เกมมีความน่าตื่นเต้นมากขึ้น

4. เพิ่มเงื่อนไขในเกมให้มากขึ้นเพื่อให้เกมมีความสนุกมากขึ้น

5. เพิ่มให้มีสัตว์ประหลาดหรืออุปสรรคต่างในเกมเพื่อเพิ่มความสนุกสนานของเกม

6. เพิ่มการเล่นหลายผู้เล่นโดยผ่านระบบเครือข่ายเพื่อเพิ่มความสนุกสนานของเกม

เอกสารอ้างอิง

[1] Sarmad Kh. Abdulla." Implementing Skin Meshes with DirectX 8. " [Online]. Available: http://www.gamedev.net/reference/programming/features/skinmesh/default.asp

[2] ANDY PINK.COM. " andypike_com tutorials directx 8. " [Online]. Available: http://www.andypike.com/tutorials/directx8

[3] Drunken Hyena. " NeHe-Style Tutorials. " [Online]. Available: http://www.drunkenhyena.com/cgi-bin/directx.pl

[4] Denis " Mr. Snow " Kozhukhov. "Mr.Snow DirectX8 Column. " [Online]. Available: http://www.xdev.ru/dxgp/rgd_articles_e.asp?s=columns&art=mrsdx8_0000:

[5] Vaughan Young. " **PROGRAMMING A MULTIPLAYER FPS IN DIRECTX.** " United State of America. CHARLES RIVER MEDIA, INC. 2005.

[6] พูนศักดิ์ ธนพันธ์พานิช. " **3D studio MAX5** " สำนักพิมพ์ เอส.พี.ซี.บุ๊คส์. กรุงเทพฯ. 2545.

[7] อนุศาสน์ สุวรรณเพรหม และ พีรพล อริยรัตนา. " **3D Studio MAX RELEASE 3** " หจก.วัฒนกันต์. กรุงเทพฯ. 2542.

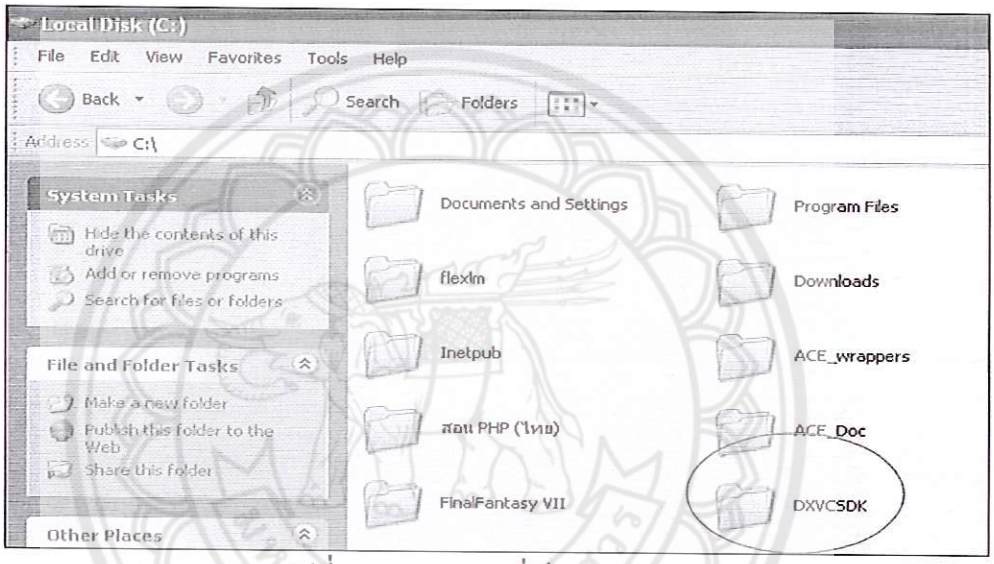
[8] Microsoft Corporation. "DirectX 8.0 for C/C++" [Online]. Available: http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/dx81_c/directx_cpp/Intro/DX8WhatsNew.asp

ภาคผนวก ก

การติดตั้งโปรแกรม Microsoft Visual C++ 6.0 และ DirectX 8.0

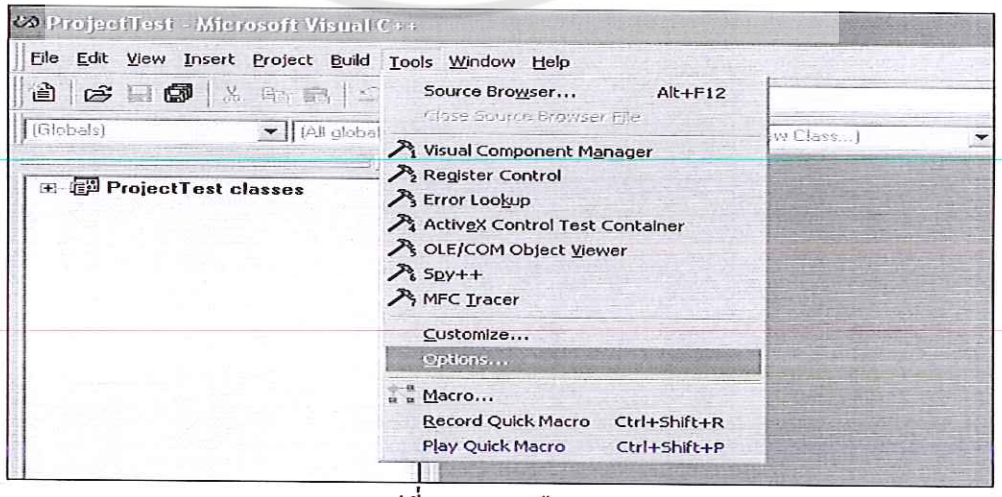
ก่อนจะใช้งานโปรแกรม Microsoft Visual C++ 6.0 และ DirectX 8.0 จะต้องตั้งค่าการทำงานให้ Microsoft Visual C++ 6.0 ใช้งานร่วมกับ DirectX 8.0 ก่อนดังนี้

1. ติดตั้งโปรแกรม Microsoft Visual C++ 6.0
2. ติดตั้งโปรแกรม DirectX 8.0 ดังรูปที่ ก-1



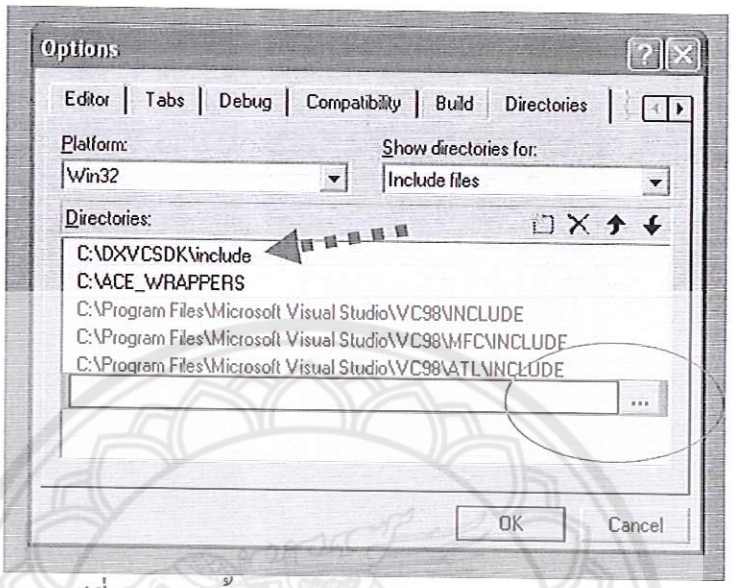
รูปที่ ก-1 Directory ที่เก็บ DirectX 8.0

3. เปิดโปรแกรม Microsoft Visual C++ แล้วเลือกเมนู Tools =>Options ดังรูปที่ ก-2



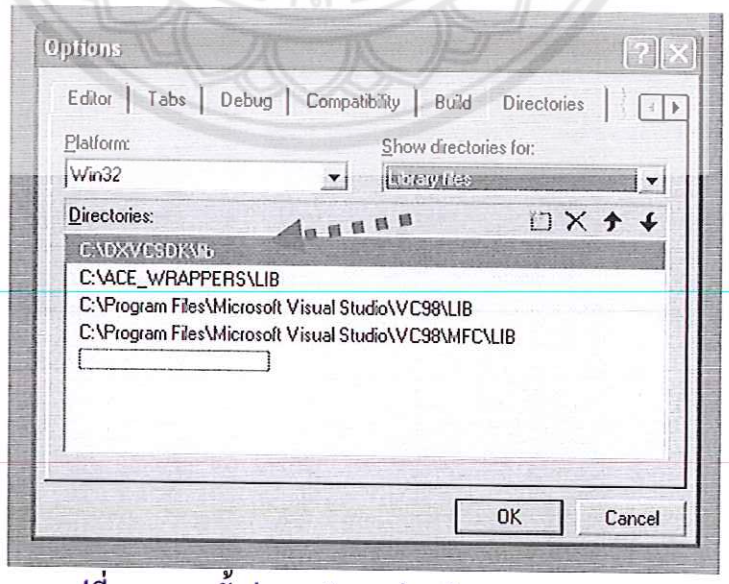
รูปที่ ก-2 การเลือกเมนู

4. เลือกเมนู Directories ในเมนู Show directories for: เลือก Include files จากนั้นเลือกจุดที่แสดงในรูปที่ ก-3 เลือกโฟลเดอร์ที่เก็บไฟล์ include ของ DirectXไว้ แล้วเลื่อนไปไว้บนสุดดังรูปที่ ก-3



รูปที่ ก-3 การตั้งค่าการทำงานร่วมกันของ Include files

5. สุดท้ายทำเหมือนข้อ 4 ในเมนู Show directories for: เลือก Library files จากนั้นเลือกจุดที่แสดงในรูปที่ ก-4 เลือกโฟลเดอร์ที่เก็บไฟล์ include ของ DirectXไว้ แล้วเลื่อนไปไว้บนสุดดังรูปที่ ก-4



รูปที่ ก-4 การตั้งค่าการทำงานร่วมกันของ Library files

ภาคผนวก ข

การใช้งานฟังก์ชันที่สำคัญ ในโครงงาน

1. ฟังก์ชัน CreateWindow ()

```
CreateWindow(

DWORD dxExStyle,          //รูปแบบหน้าต่าง
LPCTSTR lpszClassName,    //ชื่อ class ที่ต้องการ Register
LPCTSTR lpszWindowName,   //ข้อความที่ต้องการให้แสดงบน Title Bar
DWORD dwStyle,    //กำหนดลักษณะหน้าต่าง เช่นWS_OVERLAPPEDWINDOW
Int x,                    //แสดง windows จากตำแหน่งขอบจอในแนวแกน x
Int y,                    //แสดง windows จากตำแหน่งขอบจอในแนวแกน y
Int nWidth,               //ความกว้างของ windows
Int nHeight,              //ความสูงของ windows
HWND hWndParent,     //ชี้ไปที่หน้าต่างหลัก ถ้าไม่ใช่หน้าต่างหลักกำหนดเป็น
NULL
HMENU hMenu,              //ชี้ไปที่เมนูรีซอส ถ้าไม่มีกำหนดให้เป็น NULL
HINSTANCE hinstance,      //อ้างถึงโปรแกรมที่กำลังทำงานอยู่
LPVOID lpParam            //Additional Menu
                );
```

2. การใช้งานฟังก์ชัน Winmain ()

```
//function WinMain
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
int WINAPI WinMain( HINSTANCE hInstance,HINSTANCE hPreinstance,
                    LPSTR lPcmdline,int nCmdshow)
{
```

```
//1. ส่วนนี้เป็นเป็น class WNDCLASS ใช้กำหนดลักษณะ Windows
//=============================================================
    WNDCLASS wndClass;          //ประกาศตัวแปรโครงสร้าง wndClass
    wndClass.cbSize = sizeof(WNDCLASS); //cbSize จะเก็บขนาดของ WNDCLASS
    wndClass.style  = CS_VREDRAW|CS_HREDRAW://กำหนดรูปแบบclass
    wndClass.lpfnWndProc  = WinProc; //เก็บชื่อฟังก์ชันที่ทำการตอบสนอง Message
    wndClass.cbClsExtra     = 0; //เป็นการกำหนดหน่วยความจำเพิ่มเติมให้กับ class
    wndClass.cbWndExtra   = 0;//เป็นการกำหนดหน่วยความจำเพิ่มเติมให้กับ windows
    wndClass.hInstance      = hInstance; //hInstance:เก็บหมายเลขโปรแกรมที่กำลังทำงาน
                                         //อยู่
    wndClass.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
                                         //กำหนดสีพื้นหลัง: สีขาว
    wndClass.lpszClassName = "TestWin"; //เก็บชื่อ class ของโปรแกรม
    wndClass.lpszMenuName          = NULL; //เก็บชื่อเมนู
    wndClass.hIcon  = LoadIcon(hInstance,IDI_APPLICATION);
    wndClass.hCursor = LoadCursor(hInstance,IDC_ARROW);
    wndClass.hIconSm = LoadIcon(hInstance,IDI_APPLICATION);

    //3 บรรทัดข้างบนนี้ เป็นการกำหนด Icon (สี่เหลี่ยม) และ cursor (รูปลูกศร) ของ windows
    // hIcon และ hIconSm จะเหมือนกัน ตรง IDI_APPLICATION สามารถเปลี่ยนได้ดังนี้
                                         //      IDI_WINLOGO
                                         //      IDI_HAND
                                         //      IDI_QUESTION
                                         //      IDI_EXCLAMATION
                                         //      IDI_ASTERISK
//=============================================================
//2.ทำการ Regoster Class
//*************************************************************
    RegisterClass(&wndClass); //RegisterClass จะทำการสร้าง class ตามแบบ wndClass

//*************************************************************
```

```
//3. สร้าง Windows
//-------------------------------------------------------------------
        HWND hwnd;
        MSG msg;
        hwnd = CreateWindow(WS_EX_TOPMOST, "TestWin", "Window",
WS_POPUP,0,0,GetSystemMetrics(SM_CXSCREEN),
                        GetSystemMetrics(SM_CYSCREEN),
                        NULL, NULL, hInstance, NULL);
        ShowWindow(hwnd,nCmdshow); //ทำการแสดง windows ที่สร้างตามแบบ
                                   //nCmdshow
        UpdateWindow(hwnd);        //สั่งให้หน้าต่างเริ่มทำงาน
//-------------------------------------------------------------------
4.จัดการเกี่ยวกับระบบ message
while(1)
        {
                //PeekMessage จะตรวจสอบว่ามี Message ส่งเข้าหรือไม่
                if (PeekMessage(&msg,NULL,0,0,PM_NOREMOVE))
                {
                        //GetMessage จะทำการขอเอา Message มาใช้
                        if (!GetMessage(&msg,NULL,0,0))
                                return msg.wParam;
                        TranslateMessage(&msg); //ทำการแปล Message
                        DispatchMessage(&msg); //ส่ง Message ออกไปให้ WindowProc
                } else
                {
                        //ถ้าไม่มี Message จะให้โปรแกรมทำอะไร เช่น วาดภาพตัวละคร
                }
        }
        return (msg.wParam); //คืนค่า wParam หากต้องการตรวจสอบค่าครั้งล่าสุด
}
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

3. การใช้งานฟังก์ชัน WinProc ()

```
LRESULT FAR PASCAL WinProc(HWND hwnd, UINT msg,WPARAM wParam,
LPARAM lParam)
{
        switch(msg)
                {
                        case  WM_DESTROY :
                        {
                                //คืนหน่วยความจำ
                                PostQuitMessage(0);
                        }break;

                        default : //ถ้าไม่ตรง case ใดๆ DefWinProc จะจัดการค่า default ให้
                        return DefWindowProc ( hwnd,  msg,  wParam, lParam);
                }
}
```

5. การ Initialize COM

```
IDirectMusicLoader8*      g_pLoader      = NULL;
IDirectMusicPerformance8* g_pPerformance  = NULL;
IDirectMusicSegment8*     g_pSegment     = NULL;

ทำการ Initialize Com

int WINAPI WinMain( HINSTANCE hInst, HINSTANCE hPrevInst,
   LPSTR pCmdLine, INT nCmdShow )
{
  // ทำการ Initialize Com
   CoInitialize(NULL);
```

```
// สร้าง Loader Object
CoCreateInstance(CLSID_DirectMusicLoader, NULL,
        CLSCTX_INPROC, IID_IDirectMusicLoader8,
        (void**)&g_pLoader);

// สร้าง Performance Object
CoCreateInstance(CLSID_DirectMusicPerformance, NULL,
        CLSCTX_INPROC, IID_IDirectMusicPerformance8,
        (void**)&g_pPerformance );

//ต่อไปก็ทำการ Initialize Performance และ Setup the Synthesizer

g_pPerformance->InitAudio(
                NULL,           // กำหนดให้เป็น NULL
                NULL,           // กำหนดให้เป็น NULL
                NULL,           // แฮนเดิล วินโดว์

                // Default audiopath type.เป็นรูปแบบทั่วไปเสียงที่ได้จะเป็น
                //Stereo และ Reverb
                DMUS_APATH_SHARED_STEREOPLUSREVERB,
                64,             // จำนวน Performance Channel
                DMUS_AUDIOF_ALL,    // รูปแบบ Synthesizer
                NULL        // Audio parameters กำหนดเป็น NULL(default)
               );
```

6. การโหลดไฟล์เสียง

```
if (FAILED(g_pLoader->LoadObjectFromFile(
                CLSID_DirectMusicSegment, // ชื่อ Class ใน IDirectMusicLoader8
                IID_IDirectMusicSegment8,  // ID ที่ต้องการเชื่อมต่อ
               wstrFileName,          // ชื่อไฟล์ที่จะโหลด
```

```
                    (LPVOID*) &g_pSegment  //ตัวแปร Pointer ที่จะใช้อ้างถึงเมื่อต้องการ
                                           //ติดต่อ
)))
{

 //Error

 return 0;
}
```

7. การหยุดเล่นโดยใช้ฟังก์ชัน Stop ()

```
g_pPerformance->Stop(
                        NULL,   // หยุดการเล่น segment ทั้งหมดที่ mtTime
                        NULL,   // หยุดการเล่น segment state
                        0,      // เวลาที่หยุดเล่น
                        0       // ค่า Flags ทำการหยุดเล่นโดยทันที
                    );
```

ประวัติผู้เขียนโครงงาน

ชื่อ นายยุทธพงษ์ หาญพยอม

วันเดือนปีเกิด 26 มกราคม 2526

ที่อยู่ 180 ม.7 ต.เพชรละคร อ.หนองไผ่ จ.เพชรบูรณ์

โทรศัพท์ 04-0487554

ประวัติการศึกษา

- จบระดับประถมศึกษาจากโรงเรียนบ้านท่าเสา
- จบระดับมัธยมศึกษาตอนต้นจากโรงเรียนเพชรละครวิทยา
- จบระดับมัธยมศึกษาตอนปลายจากโรงเรียนเพชรละครวิทยา
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นที่ 4
  สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์
  มหาวิทยาลัยนเรศวร

E-mail : fanstar@hotmail.com