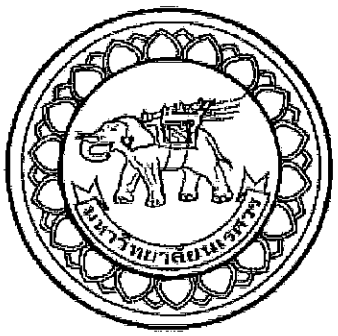


การพัฒนาเกมสามมิติโดยใช้โอเพ็นจีแอล (OpenGL)

3D GAME DEVELOPMENT BY OPENGL

นายธงชัย แก้วเพียร รหัส 45360161

นายรัฐพล ชัยนนถี รหัส 45360377

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ปีการศึกษา 2548

ใบรับรองโครงงานวิศวกรรม

หัวข้อโครงงาน การพัฒนาเกมสามมิติโดยใช้โอเพ็นจีแอล (OpenGL)

ผู้ดำเนินโครงงาน นายธงชัย แก้วเพียร รหัส 45360161

นายรัฐพล ชัยนนถี รหัส 45360377

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	การพัฒนาเกมสามมิติโดยใช้โอเพ็นจีแอล (OpenGL)		
ผู้ดำเนินโครงงาน	นายธงชัย	แก้วเพียร	รหัส 45360161
	นายรัฐพล	ชัยนนถี	รหัส 45360377
อาจารย์ที่ปรึกษา	ดร.พนมขวัญ	ริยะมงคล	
ภาควิชา	วิศวกรรมไฟฟ้าและคอมพิวเตอร์		
สาขาวิชา	วิศวกรรมคอมพิวเตอร์		
ปีการศึกษา	2548		

บทคัดย่อ

เกมคอมพิวเตอร์นั้นได้รับความนิยมมาเป็นเวลานานแล้ว มีหลายชนิดและหลายรูปแบบ ปัจจุบันเกมคอมพิวเตอร์ได้มีการพัฒนามากขึ้นที่เป็นแบบ 3D GAME หรือ เกม 3 มิติ ซึ่งกำลังเป็นที่นิยมอย่างมาก เพราะให้ภาพที่ดูสวยงามและสมจริง ในโครงงานวิจัยนี้ได้ใช้ OpenGL ในการสร้างและพัฒนาเกมสามมิติ โดยเกมสามมิติที่พัฒนานี้ส่วนของภาพ ตัวละครและพื้นหลังต่างๆ จะใช้โปรแกรม Microsoft Visual C++ ช่วยในการพัฒนา โดยเกมที่สร้างขึ้นจะมีลักษณะเป็นเกมเขาวงกตที่มีการใช้ คีย์บอร์ดในการบังคับให้เดินไปตามทิศทางต่างๆ

Project Title	3D Game development by OpenGL		
Name	Mr.Thongchai	Kaewpean	ID 45360161
	Mr.Ruttapon	Chainontee	ID 45360377
Project Advisor	Dr.Panomkhawn Riyamongkol		
Major	Computer Engineering		
Department	Electrical and Computer Engineering		
Academic Year	2005		

.....

ABSTRACT

Computer games have been popular for a long time. There are many kinds and many forms of computer games. Nowadays, computer games have been developed more and more and become 3D Game. It is very popular because of its beautiful real pictures. This project used OpenGL to build and develop the 3D Game. The pictures, characters, and settings in this 3D Game used Microsoft visual C++. The result of this project is a maze, a kind of 3D games, which uses a keyboard to control.

กิตติกรรมประกาศ

โครงงานวิศวกรรม การพัฒนาเกมสามมิติโดยใช้ OpenGL จะไม่สามารถสำเร็จได้ถ้าไม่ได้รับการสนับสนุนและความช่วยเหลือจากบุคคลจากหลายๆ ฝ่ายซึ่งประกอบไปด้วย ดร. พนมขวัญ ริยะมงคล อ. พงศ์พันธ์ กิจสนาโยธิน และ ผศ. ประทีป ตรีรณโอภาส ที่คอยให้ความเอาใจใส่และให้คำปรึกษาแนะนำจนสามารถนำความรู้ที่ได้มาทำการประยุกต์ใช้กับโครงงานนี้และทำให้โครงงานนี้สำเร็จลุล่วงไปได้ด้วยดีซึ่งต้องขอขอบพระคุณเป็นอย่างยิ่ง

สุดท้ายนี้ต้องขอขอบพระคุณ บิดา มารดา ที่ให้ความรักความอบอุ่นและกำลังใจเสมอมา และขอขอบคุณบุคคลทุกท่านที่มีส่วนช่วยเหลือที่ไม่กล่าวถึงมา ณ ที่นี้

คณะผู้จัดทำ
นายธงชัย แก้วเพียร
นายรัฐพล ชัยนนถี

สารบัญ

หน้า

สารบัญ(ต่อ)

หน้า

สารบัญตาราง

สารบัญรูป

รูปที่ หน้า

สารบัญรูป(ต่อ)

รูปที่ หน้า

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของโครงงาน

ปัจจุบันนักพัฒนาเกมคอมพิวเตอร์ส่วนใหญ่ จะมุ่งเน้นในการใช้ DirectX เป็นเครื่องมือที่ช่วยในการพัฒนาเกมซึ่ง OpenGL ก็เป็นอีกเครื่องมือที่สามารถนำมาใช้พัฒนาเกมคอมพิวเตอร์ในรูปแบบของภาพกราฟิกทั้ง 2 มิติและ 3 มิติให้มีคุณสมบัติเทียบเท่ากับ DirectX ได้และยังมีข้อได้เปรียบกว่า DirectX อีกอย่างคือ OpenGL สามารถทำงานบนระบบปฏิบัติการอื่นๆ ได้แต่ DirectX สามารถทำงานได้บนระบบปฏิบัติการ Window เท่านั้น

โครงงานนี้จึงได้นำ OpenGL มาใช้ในการพัฒนาโปรแกรมเกมแบบ 3 มิติเพื่อที่จะเผยแพร่ความสามารถของ OpenGL ให้บุคคลที่สนใจได้ศึกษาต่อไป

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อศึกษาหลักการทำงานของ OpenGL ที่ใช้ในการพัฒนาเกมประเภท 3 มิติ

1.2.2 เพื่อพัฒนาเกมคอมพิวเตอร์ที่มีการแสดงผลในรูปแบบ 3 มิติ

1.3 ขอบข่ายของโครงงาน

1.3.1 สามารถควบคุมภาพวัตถุในเกมโดยใช้คีย์บอร์ดให้เคลื่อนที่ไปข้างหน้า ไปข้างหลัง เลี้ยวซ้าย และเลี้ยวขวาได้

1.3.2 การแสดงผลในเกมจะเป็นภาพแบบ 3 มิติ

1.3.3 ลักษณะของเกมเป็นแบบมีผู้เล่นคนเดียวที่ไม่ซับซ้อน

1.4 ขั้นตอนการดำเนินงาน

ตารางที่ 1.1 ขั้นตอนการดำเนินโครงการ

กิจกรรม	เดือน-ปี										
	ธ.ค.	ม.ค.	ก.พ.	มี.ค.	เม.ย.	พ.ค.	มิ.ย.	ก.ค.	ส.ค.	ก.ย.	ต.ค.
1. รวบรวมข้อมูล	←	—	—	—	—	→					
2.จัดหาเครื่องมือ	←	—	→								
3. ศึกษาทฤษฎีที่ใช้		←	—	—	—	—	—	—	→		
4. ออกแบบเกม	←	—	→								
5. เขียนโปรแกรม		←	—	—	—	—	—	—	→		
6.ทดสอบและแก้ไข			←	—	—	—	—	—	—	→	
7. สรุปผล										←	→
8. จัดทำคู่มือ											↔

1.5 ผลที่คาดว่าจะได้รับ

1.5.1 ได้ประสบการณ์การพัฒนาเกมโดยใช้คุณลักษณะของ OpenGL

1.5.2 เพิ่มความชำนาญในการเขียนโปรแกรม Microsoft Visual C++

1.5.3 มีความรู้และความเข้าใจเกี่ยวกับงานคอมพิวเตอร์กราฟิก

1.6 งบประมาณที่ใช้

1.6.1 ค่าจัดซื้อหนังสือ	1,200	บาท
1.6.2 ค่าพิมพ์เอกสารและถ่ายเอกสาร	600	บาท
1.6.3 ค่าวัสดุคอมพิวเตอร์	200	บาท
รวม	2,000	บาท

บทที่ 2
หลักการและทฤษฎี

2.1 OpenGL คืออะไร

ในสมัยแรกๆ การเขียนโปรแกรมให้แสดงภาพกราฟิกบนจอภาพถ้าต้องการจะเขียนภาพเส้นตรง วงกลม วงรี การระบายสีโพลีกอน (รูปหลายเหลี่ยม) การย้ายวัตถุ หมุนวัตถุ ย่อหรือขยายวัตถุและการสะท้อนวัตถุจะต้องเขียนโปรแกรมติดต่อกับฮาร์ดแวร์โดยตรง ดังนั้นจึงได้มีการสร้างกราฟิกไลบรารี่เพื่อเป็นคำสั่งในการเขียนโปรแกรมด้านกราฟิก เช่น การวาดเส้นตรง การวาดวงกลมและการวาดวงรี ต่อมาบริษัท Silicon Graphics ได้สร้าง OpenGL (Open Graphics Library) เป็นระบบเปิดและไมโครซอฟต์ก็ได้สร้าง DirectX ซึ่งเป็นกราฟิกไลบรารี่เช่นเดียวกัน แต่มีมาตรฐานที่ต่างกันปัจจุบันการ์ดแสดงผลแบบสามมิติรุ่นใหม่ๆ ได้นำชุดคำสั่งของ OpenGL ไปเก็บไว้ในตัวการ์ดแสดงผลเพื่อให้การทำงานมีความรวดเร็วขึ้น

ดังนั้น OpenGL คือ ซอฟแวร์ที่ใช้ในการเชื่อมต่อกับฮาร์ดแวร์ที่แสดงผลทางด้านกราฟิก ซึ่งประกอบด้วยคำสั่งประมาณ 250 คำสั่งซึ่งตัว OpenGL นี้จะมีการวางโครงสร้างเป็น Hardware-independent interface และสามารถใช้ได้กับระบบปฏิบัติการหลายๆ แบบและ OpenGL จะใช้คำสั่งวาดภาพค่อนข้างพื้นฐาน เช่นคำสั่งในการวาดจุด เส้นและรูปเหลี่ยมต่างๆ

2.2 ภาษาที่สามารถใช้กับ OpenGL

- C/C++ (VC++, Borland C++, C++ Builder, C compiler on UNIX)
- Delphi
- Visual Basic
- Java
- Perl
- Python
- Fortran
- Ada

2.3 หลักการทำงานของ OpenGL

OpenGL เป็น Graphic API (Application Programming Interface) ที่เป็นทั้งแบบ 2 มิติและแบบ 3 มิติโดยการทำงานจะแยกเป็นสองส่วนอันดับแรก คือ ทางฮาร์ดแวร์ที่ต้องรองรับการทำงาน

ของ OpenGL และฮาร์ดแวร์จะทำงานได้ต้องทำงานผ่าน OpenGL ICD (Installable Client Driver) ซึ่งจะมีมากับไดร์เวอร์ของการ์ดแสดงผลทุกตัวซึ่งบางตัวก็จะเป็นแค่ MCD (Mini Client Driver) อีกส่วนก็คือ WGL (OpenGL Extension for Microsoft Window) หรือ Environment ของ Window (Win32) ที่จะเป็นตัวกลางระหว่าง OpenGL และฮาร์ดแวร์เพื่อให้ระบบสามารถใช้ OpenGL ได้อย่างถูกต้อง ดังนั้นOpenGL คือ API แบบ 2 มิติและ 3 มิติที่สามารถย้ายโอนระบบได้อย่างอิสระซึ่งจะหมายความว่าระบบ API ตัวนี้จะไม่อิงกับระบบปฏิบัติการ Windows, Mac หรือระบบใด ๆ

ภายใต้ระบบปฏิบัติการหนึ่งๆ จะมีสิ่งที่เรียกว่า System Bindings ที่ทำหน้าที่เป็นตัวประสานระหว่าง OpenGL กับระบบปฏิบัติการโดยตัวประสานตัวนี้จะยึดกับ library ที่เรียกว่า GLX (OpenGL Extension for the X Window System) ซึ่งพัฒนาโดย SGI แต่ถ้าอยู่ภายใต้รูปแบบของระบบปฏิบัติการ Windows (Win32) ตัวประสานนี้จะเรียกว่า WGL (วิกเกิ้ล) ซึ่งพัฒนาโดยไมโครซอฟต์จะเห็นได้ว่า OpenGL นั้นจะสามารถทำงานบนระบบปฏิบัติการต่างๆ ได้โดยอิสระแต่ตัวประสาน (GLX, WLX) นั้นจะต้องมีการแก้ไขไปตามแต่ระบบที่ใช้ เช่น Mac, Linux หรือ PC ซึ่งในแต่ละระบบปฏิบัติการนั้นจะมี Driver interface (ไดร์เวอร์ที่ทำให้ OpenGL สื่อสารกับระบบปฏิบัติการได้) ที่ต่างกันออกไปโดยถ้าอยู่ในรูปแบบของ Win32 นั้นจะมี Driver interface อยู่ 2 แบบคือ

1. ICD (Installable Client Driver)
2. MCD (Mini Client Driver)

ICD หมายถึงไดร์เวอร์ของ OpenGL ที่สมบูรณ์แบบที่สามารถรองรับการทำงานของ OpenGL ได้ทั้งหมดและสามารถทำงานร่วมกับส่วนอื่นๆ ของไดร์เวอร์นั้นได้เมื่อไดร์เวอร์ OpenGL ถูกเรียกว่า ICD ซึ่งหมายความว่าไดร์เวอร์ตัวนี้จะเชื่อมโยงกับการทำงานของ ICD interface ของไมโครซอฟต์ซึ่งจะทำให้โปรแกรมใดๆ ก็ตามที่เชื่อมโยงกับ OPENGL32.LIB และใช้ WGL และ OPENGL32.DLL เป็นตัวประสานก็จะสามารถทำงานร่วมกับไดร์เวอร์ ICD นี้ได้โปรแกรมดังกล่าวได้แก่

1. โปรแกรมด้านงานออกแบบ (Modeling / Creative Content Creation)
2. CAD / CAM (COMPUTER-AIDED DESIGN / CAMERA ANALOGY)
3. Developer Toolkits & Libraries, Game Engineś, 3D APIs
4. Games
5. VRML Authoring & Viewing
6. Utilities; Screen Savers, Benchmarks
7. Simulations & Visualizations
8. Scientific, Data Analysis & Geographic Mapping

เนื่องจากว่า WGL ถูกออกแบบมานานมากแล้วตัวประสาน WGL จึงเข้าใจการทำงานแบบการ์ดแสดงผลตัวเดียวเท่านั้นดังนั้น WGL จึงไม่ได้ถูกออกแบบมาให้รองรับการ์ดเร่งความเร็ว (การ์ดแสดงผล) แบบ 3 มิติอย่างเช่นการ์ดแสดงผลรุ่น Voodoo, Voodoo 2 ซึ่งหมายความว่า 3Dfx จะไม่สามารถเขียน ICD สำหรับ Voodoo, Voodoo 2 ได้

ไดร์เวอร์ของ OpenGL สำหรับ Voodoo และ Voodoo 2 นั้นเราเรียกว่า Stand Alone Driver ซึ่งคือไดร์เวอร์ที่สามารถรองรับการทำงานของ OpenGL แต่ไม่รองรับกลไกการเชื่อมต่อของไดร์-เวอร์ ICD ของไมโครซอฟต์ที่เหลือนอกจากนี้ยังมีไดร์เวอร์อีกตัวคือ Mini Driver คือไดร์เวอร์ที่รองรับการทำงานบางส่วนของ OpenGL (คล้ายๆ กับ Stand Alone driver) ซึ่งถูกออกแบบมาให้ใช้เฉพาะชุดคำสั่งที่ OpenGL เรียกใช้ อย่างในเกม GL Quake และ Quake2 ข้อแตกต่างระหว่าง Stand Alone กับ Mini driver คือ Stand Alone จะเป็น OpenGL ที่สมบูรณ์แบบซึ่งสามารถใช้งานได้กับเกมที่ใช้ OpenGL ส่วน Mini Driver จะมีแค่บางส่วนของ OpenGL ซึ่งสามารถใช้งานได้กับเฉพาะกับเกมบางเกมที่เรียกใช้ชุดคำสั่งที่มีใน Mini Driver เท่านั้น

สรุป

ไดร์เวอร์ ICD เป็นไดร์เวอร์ของ OpenGL ที่ครบถ้วนและสมบูรณ์และยังสามารถรองรับกลไกการเชื่อมต่อ ICD ของ Microsoft ส่วน Stand Alone driver นั้นรองรับ OpenGL ครบถ้วนแต่ไม่ได้ใช้กลไกการเชื่อมต่อของไมโครซอฟต์ในการติดต่อกับระบบปฏิบัติการและท้ายสุด MCD driver เป็นไดร์เวอร์ที่รองรับการทำงานของ OpenGL แค่บางส่วนและสำหรับการ์ดแสดงผล 3 มิติรุ่นเก่าๆ ที่ไม่มี OpenGL ICD แต่รองรับการทำงานของ DirectX6 ก็สามารถใช้คุณสมบัติของ OpenGL ได้โดยผ่านทางโปรแกรม SciTech GL Direct หรือ Alt Software Mesa DirectX 6 Driver ซึ่งโปรแกรมทั้งสองตัวนี้จะทำให้โปรแกรม 3 มิติที่ใช้ OpenGL ทำงานผ่านไดร์เวอร์ Direct3D ของไมโครซอฟต์ได้

2.4 พื้นฐานการเขียน OpenGL

2.4.1 Hello World

เป็นการเขียนโปรแกรมพื้นฐานเพื่อให้แสดงหน้าต่างโดยมีรูปแบบคำสั่งดังนี้

```
#include < windows.h >
#include < stdio.h >
#include < GL/glut.h > //เป็นHeader Files ของ OpenGL
void display(void)
{
  glColor3f(0.0, 0.0, 0.0); //กำหนดค่าของสี (ให้เป็นสีดำ)
  glClear( GL_COLOR_BUFFER_BIT); //เคลียร์ color buffer
```

```
  glFlush(); //แสดงผลลัพธ์ของคำสั่ง
}
int main(int argc, char **argv)
{
  glutInit(&argc, argv); //ฟังก์ชันที่กำหนดค่าเริ่มต้นให้กับ GLUT library
  glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
  // กำหนดค่าเริ่มต้นให้กับหน้าต่าง
  glutCreateWindow ("Hello World"); //สร้างหน้าต่างที่มีชื่อว่า Hello World
  glutDisplayFunc(display); //กำหนด the display callback
  glutMainLoop(); //เริ่มต้นรับคำสั่ง
  return 0;
}
```

ผลการรันโปรแกรม

หลังจากการรันโปรแกรมจะได้ภาพของหน้าต่างออกมาทางหน้าจอและที่ title bar จะมีชื่อ Hello World แสดงอยู่ดังรูปที่ 2.1

รูปที่ 2.1 ผลการรันคำสั่งวาดหน้าต่างที่ชื่อ Hello World

คำอธิบายฟังก์ชัน

void glutInit(int *argcp, char **argv);

glutInit เป็นฟังก์ชันที่ทำหน้าที่กำหนดค่าเริ่มต้นต่างๆ ให้กับ GLUT library ซึ่งต้องเรียกเป็นฟังก์ชันแรกก่อนเรียกฟังก์ชันอื่นๆ ใน GLUT library โดยพารามิเตอร์ที่ใส่ไป 2 ตัวก็เป็น

พารามิเตอร์ของ main โปรแกรม

void glutInitDisplayMode(unsigned int mode);

glutInitDisplayMode เป็นการกำหนดค่าเริ่มต้นให้กับตัวหน้าต่างที่เราสร้างขึ้นว่าจะมีการแสดงผลอย่างไรโดยค่าของ mode ถูกกำหนดไว้ก่อนแล้วสามารถใช้เครื่องหมาย OR bitwise มาเชื่อม mode ต่างๆ ได้

int glutCreateWindow(char *name);

glutCreateWindow เป็นการสร้างหน้าต่างขึ้นมาโดย name จะเป็นชื่อของหน้าต่างและแสดงบน title bar ของหน้าต่าง

void glutDisplayFunc(void (*func)(void));

glutDisplayFunc เป็นการกำหนด the display callback ว่าเป็นฟังก์ชันไหน โดยพารามิเตอร์ที่ใส่เป็นชื่อของฟังก์ชัน กล่าวคือเมื่อหน้าต่างมีการถูกวาดใหม่ให้เรียกฟังก์ชันอะไร

void glutMainLoop(void);

glutMainLoop เป็นการเริ่มต้นรับเหตุการณ์ต่างๆ ของผู้ใช้ที่จะกระทำต่อหน้าต่าง

void glColor3f(GLfloat red, GLfloat green, GLfloat blue);

glColor3f เป็นการกำหนดค่าของของสีโดยเกิดจากแม่สี 3 สี คือ สีแดง สีเขียวและสีน้ำเงิน

void glClear(GLbitfield mask);

glClear เป็นการ initialize ตัว buffer ซึ่งตัวสีพื้นโดยถูกกำหนดจากคำสั่งพวกกำหนดสี เช่น glColor3f สำหรับ mask จะบอกว่า clear ที่ buffer ใดบ้างซึ่งเป็นค่าที่เป็นค่าคงที่ที่กำหนดไว้แล้ว โดยถ้าใส่มากกว่า 1 ตัวโดยใช้ OR bitwise เป็นตัวเชื่อม

void glFlush(void);

glFlush เป็นการบังคับให้แสดงผลลัพธ์ของคำสั่งต่างๆ ก่อนที่เรียก glFlush

2.4.2 POINTS

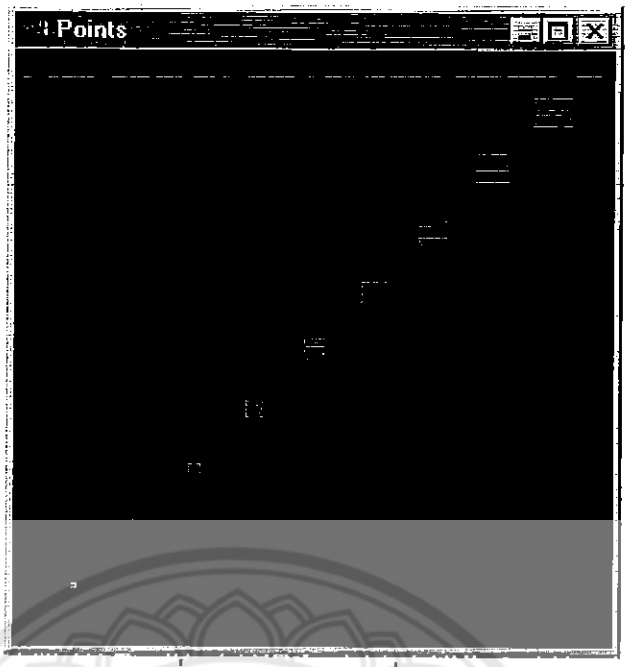
เป็นการเขียนโปรแกรมเพื่อที่จะแสดงให้เห็นถึงการวาดจุดของ OpenGL ให้มีขนาดที่ต่างกันโดยมีคำสั่งดังนี้

```
#include < windows.h >
#include < stdio.h >
#include < GL/glut.h > //เป็นHeader Files ของ OpenGL
void display(void)
{
int i,j; //ประกาศตัวแปร i,j เป็นตัวแปรชนิด integer
glColor3f(0.0, 0.0, 0.0); //กำหนดค่าของสี
```

```
glClear( GL_COLOR_BUFFER_BIT); //เคลียร์ color buffer
glColor3f(1.0, 0.0, 0.0); //กำหนดค่าของสี
 j=1;
for(i=-4;i<5;i++)//วนลูปรับค่า i ตั้งแต่ -4 จนถึง i<5 ทีละหนึ่งค่า
{
 j+=2;
glPointSize(j); //ฟังก์ชันที่กำหนดค่าขนาดของจุด
glBegin(GL_POINTS); //ฟังก์ชันบอกการเริ่มต้นของ list ของจุด
 glVertex3f(i*0.2f,i*0.2f,i*0.2f); //กำหนดตำแหน่งของจุด
glEnd(); //ฟังก์ชันที่บอกการสิ้นสุดของ list ของจุด
}
glFlush(); //แสดงผลลัพธ์ของคำสั่ง
}
int main(int argc, char **argv)
{
glutInit(&argc, argv); //ฟังก์ชันที่กำหนดค่าเริ่มต้นให้กับ GLUT library
glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
// กำหนดค่าเริ่มต้นให้กับหน้าต่าง
glutCreateWindow ("Hello World"); //สร้างหน้าต่างที่มีชื่อว่า Hello World
glutDisplayFunc(display); //กำหนด the display callback
glutMainLoop(); //เริ่มต้นรับคำสั่ง
}
```

ผลการรันโปรแกรม

หลังจากการรันโปรแกรมจะได้ภาพของหน้าต่างที่แสดงภาพการวาดจุดโดยใช้ OpenGL ซึ่งจะมีลักษณะเป็นจุดแบบสี่เหลี่ยมที่มีการไล่ระดับขนาดจากเล็กไปใหญ่ดังแสดงในรูปที่ 2.2



รูปที่ 2.2 ผลการรันคำสั่งวาดจุด

คำอธิบายฟังก์ชัน

void glPointSize(GLfloat size);

glPointSize เป็นฟังก์ชันที่ทำหน้าที่กำหนดค่าขนาดของจุดโดยต้องมีค่ามากกว่า 0.0 และถ้าไม่มีการกำหนดใหม่ก็จะใช้ค่า 1.0

void glVertex3f(GLfloat x, GLfloat y, GLfloat z);

glVertex เป็นฟังก์ชันที่บอก vertex หรือจุด

void glBegin(GLenum mode);

glBegin เป็นฟังก์ชันที่บอกการเริ่มต้นของ list ของจุดสำหรับ mode นั้นเป็นค่าที่กำหนดไว้ก่อนใน OpenGL

void glEnd(void);

glEnd เป็นฟังก์ชันที่บอกการสิ้นสุดของ list ของจุด

2.4.3 LINE

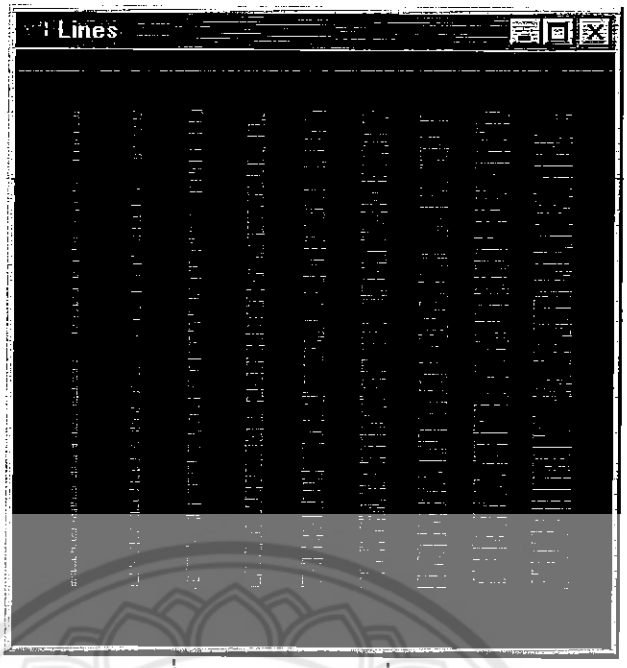
เป็นการเขียนโปรแกรมเพื่อที่จะแสดงให้เห็นถึงการวาดเส้นของ OpenGL ให้มีขนาดต่างๆ กันโดยมีคำสั่งดังนี้

```
#include < windows.h >
#include < stdio.h >
#include < GL/glut.h > //เป็นHeader Files ของ OpenGL
```

```
void display(void)
{
int i,j;
glColor3f(0.0, 0.0, 0.0); //กำหนดค่าของสี
glClear( GL_COLOR_BUFFER_BIT); //เคลียร์ color buffer
glColor3f(1.0, 0.0, 0.0); //กำหนดค่าของสี
j=1;
for(i=-4;i<5;i++) //วนลูปรับค่า i ตั้งแต่ -4 จนถึง i<5 ทีละหนึ่งค่า
{
j+=2;
glLineWidth(j); //กำหนดขนาดความกว้างของเส้น
glBegin(GL_LINES); //ฟังก์ชันบอกการเริ่มต้นของ list ของจุด
glVertex3f(i*0.2f, -0.8f, 0); //กำหนดตำแหน่งของจุด
glVertex3f(i*0.2f, 0.8f, 0); //กำหนดตำแหน่งของจุด
 glEnd(); //ฟังก์ชันที่บอกการสิ้นสุดของ list ของจุด
}
glFlush(); //แสดงผลลัพธ์ของคำสั่ง
}
int main(int argc, char **argv)
{
glutInit(&argc, argv); //ฟังก์ชันที่กำหนดค่าเริ่มต้นให้กับ GLUT library
glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
// กำหนดค่าเริ่มต้นให้กับหน้าต่าง
glutCreateWindow ("Hello World"); //ฟังก์ชันที่ สร้างหน้าต่าง
glutDisplayFunc(display); //ฟังก์ชันที่กำหนด the display callback
glutMainLoop(); //เริ่มต้นรับคำสั่ง
}
```

ผลการรันโปรแกรม

หลังจากการรันโปรแกรมจะได้ภาพของหน้าต่างที่แสดงภาพการวาดเส้นโดยใช้ OpenGL ซึ่งจะมีลักษณะเป็นเส้นแบบสี่เหลี่ยมผืนผ้าที่มีขนาดเล็กไปใหญ่ดังแสดงในรูปที่ 2.3



รูปที่ 2.3 ผลการรันคำสั่งวาดเส้น

คำอธิบายฟังก์ชัน

void glLineWidth(GLfloat size);

glLineWidth เป็นฟังก์ชันที่ทำหน้าที่กำหนดขนาดความกว้างของเส้น โดยต้องมีค่ามากกว่า 0.0 และถ้าไม่มีการกำหนดใหม่ก็จะใช้ค่า 1.0

2.4.4 POLYGON

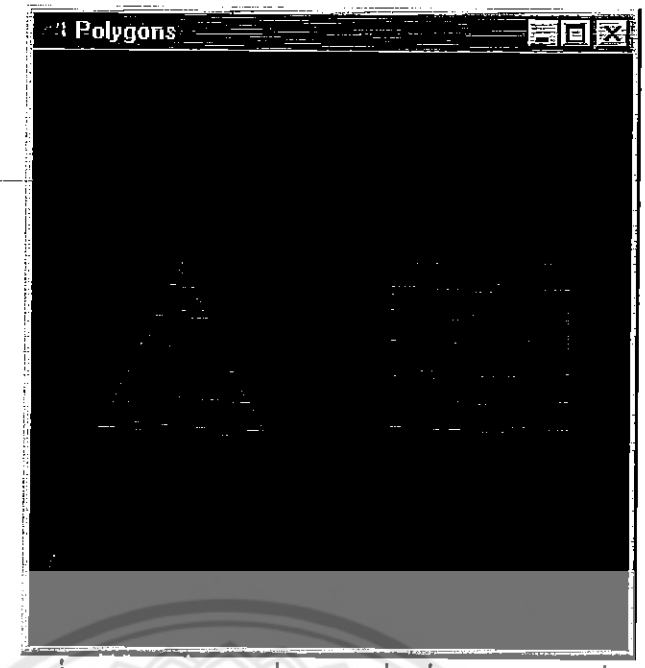
เป็นการเขียนโปรแกรมเพื่อที่จะแสดงให้เห็นถึงการวาดรูปเหลี่ยมชนิดต่างๆ ของ OpenGL โดยมีคำสั่งดังนี้

```
#include < windows.h >
#include < stdio.h >
#include < GL/glut.h > //เป็นHeader Files ของ OpenGL
void display(void)
{
glColor3f(0.0, 0.0, 0.0); //กำหนดค่าของสี
glClear( GL_COLOR_BUFFER_BIT); //เคลียร์ color buffer
glColor3f(0.0, 0.0, 1.0); //กำหนดค่าของสี
glTranslatef(0.5f,0.0f,0.0f); //กำหนดตำแหน่งของรูปสี่เหลี่ยม
glBegin(GL_QUADS); //ฟังก์ชันบอกการเริ่มต้นของ list ของจุด
```

```
glVertex3f(-0.3f, -0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสี่เหลี่ยม
glVertex3f(0.3f, -0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสี่เหลี่ยม
glVertex3f(0.3f, 0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสี่เหลี่ยม
glVertex3f(-0.3f, 0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสี่เหลี่ยม
glEnd(); //ฟังก์ชันที่บอกการสิ้นสุดของ list ของจุด
glTranslatef(-1.0f,0.0f,0.0f); //กำหนดตำแหน่งของรูปสามเหลี่ยม
glBegin(GL_TRIANGLES); //ฟังก์ชันบอกการเริ่มต้นของ list ของจุด
glVertex3f(-0.3f, -0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสามเหลี่ยม
glVertex3f(0.3f, -0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสามเหลี่ยม
glVertex3f(0.0f, 0.3f, 0.0f); //กำหนดตำแหน่งที่เป็นมุมของรูปสามเหลี่ยม
glEnd(); //ฟังก์ชันที่บอกการสิ้นสุดของ list ของจุด
glFlush(); //แสดงผลลัพธ์ของคำสั่ง
}
int main(int argc, char **argv)
{
glutInit(&argc, argv); //กำหนดค่าเริ่มต้น ให้กับ GLUT library
glutInitDisplayMode ( GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
// กำหนดค่าเริ่มต้นให้กับหน้าต่าง
glutCreateWindow ("Hello World"); //สร้างหน้าต่าง
glutDisplayFunc(display); //กำหนด the display callback
glutMainLoop(); //เริ่มต้นรับคำสั่ง
}
```

ผลการรันโปรแกรม

หลังจากการรันโปรแกรมจะได้ภาพของหน้าต่างที่แสดงภาพที่เป็นรูปสามเหลี่ยมและสี่เหลี่ยมซึ่งจะมีลักษณะดังแสดงในรูปที่ 2.4



รูปที่ 2.4 ผลการรันคำสั่งวาดรูปสี่เหลี่ยมและสามเหลี่ยม

2.5 การควบคุมวัตถุโดยใช้คีย์บอร์ด

การควบคุมโดยใช้คีย์บอร์ดจะต้องทำการประกาศตัวแปรสำหรับคีย์บอร์ดดังนี้

```
glutKeyboardFunc(void(*func)(unsigned char key, int x, int y))
```

ตัวอย่างฟังก์ชันที่ใช้ในการควบคุมโดยใช้คีย์บอร์ด

```
void keyboard(unsigned char key,int x,int y) //ประกาศตัวแปรสำหรับคีย์บอร์ด
{
switch (key) //ฟังก์ชันการเลือกรับค่าตัวแปรสำหรับคีย์บอร์ด
{
case 'A':
glutIdleFunc(spinDisplay); //ฟังก์ชันในการหมุนภาพ
break;
case 'B':
glutIdleFunc(NULL); //หยุดการทำงาน
break;
default:
break;
}
```

}

จากตัวอย่างปุ่ม A และปุ่ม B จะเป็นอินพุตคำสั่งไปสั่งงานโปรแกรมโดยที่เมื่อเรากดปุ่ม A ภาพวัตถุจะหมุนและถ้ากดปุ่ม B วัตถุจะหยุดหมุนและเราสามารถเพิ่มปุ่มคำสั่งเพิ่มเติมเพื่อเป็นอินพุตให้วัตถุเลื่อนซ้าย เลื่อนขวาหรือเคลื่อนที่ได้ตามที่เราต้องการ

ตัวอย่างโปรแกรมการควบคุมวัตถุให้หมุนโดยใช้คีย์บอร์ด

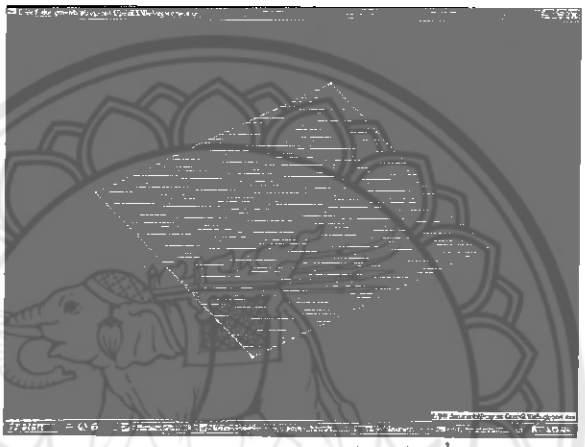
```
#include<stdlib.h>
#include<glut.h> //เป็นHeader Files ของ OpenGL
static GLfloat spin =0.0; //กำหนดให้ค่าตัวแปร spin =0.0
void init(void)
{
glClearColor(0.0,0.0,0.0,0.0);
glShadeModel(GL_FLAT); //ฟังก์ชันในการกำหนดการให้แสงและเงา
}
void display(void)
{
glClear(GL_COLOR_BUFFER_BIT);
glPushMatrix();
glRotatef(spin,0.0,0.0,1.0); //กำหนดลักษณะการหมุนของภาพ
glColor3f(1.0,0.0,1.0); //กำหนดสี
glRectf(-25.0,-25.0,25.0,25.0); //กำหนดมุมมองของภาพ
glPopMatrix();
glutSwapBuffers();
}
void spinDisplay(void)
{
spin = spin + 2.0;
if(spin > 360.0)
spin = spin - 360;
glutPostRedisplay();
}
void reshape(int w,int h)
```

```
{
glViewport(0,0,(GLsizei) w,(GLsizei)h); //ฟังก์ชันในการกำหนดมุมมอง
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-50.0,50.0,-50.0,50.0,-1.0,1.0); //ฟังก์ชันในการกำหนดตำแหน่งของมุมมอง
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}
void keyboard(unsigned char key,int x,int y) //ประกาศตัวแปรสำหรับคีย์บอร์ด
{
switch (key) //ฟังก์ชันการเลือกรับค่าตัวแปรสำหรับคีย์บอร์ด
{
case 'A':
glutIdleFunc(spinDisplay); //ฟังก์ชันในการหมุนภาพ
break;
case 'B':
glutIdleFunc(NULL); //หยุดการทำงาน
break;
default:
break;
}
}
int main(int argc,char** argv)
{
glutInit(&argc,argv); // ฟังก์ชันที่กำหนดค่าเริ่มต้น ให้กับ GLUT library
glutInitDisplayMode(GLUT_DOUBLE|GLUT_RGB); //กำหนดค่าเริ่มต้นให้กับหน้าต่าง
glutInitWindowSize(250,250); //กำหนดขนาดของหน้าต่างแสดงผลการรัน
glutInitWindowPosition(100,100); //กำหนดตำแหน่งของหน้าต่างแสดงผลการรัน
glutCreateWindow(argv[0]); //สร้างหน้าต่าง
init();
glutDisplayFunc(display); //กำหนด the display callback
glutReshapeFunc(reshape);
```

```
glutKeyboardFunc(keyboard);
glutMainLoop(); //เริ่มต้นรับคำสั่ง
return 0;
}
```

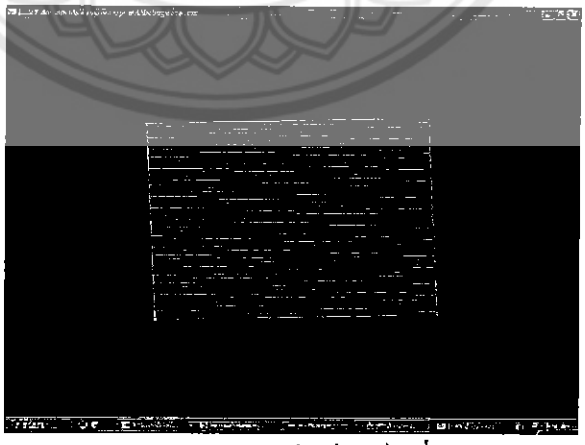
ผลการรันโปรแกรม

หลังจากการรันโปรแกรมจะมีหน้าต่างปรากฏขึ้นมาเมื่อกดปุ่ม A ภาพจะมีการหมุนดังแสดงตามรูปที่ 2.5



รูปที่ 2.5 ผลการรันโปรแกรมควบคุมโดยใช้คีย์บอร์ดเพื่อให้วัตถุหมุนเมื่อกดปุ่ม A

และเมื่อเรากดปุ่ม B วัตถุจะหยุดหมุนทันทีดังแสดงในรูปที่ 2.6



รูปที่ 2.6 ผลการรันโปรแกรมควบคุมโดยใช้ คีย์บอร์ดเพื่อให้วัตถุหยุดหมุนเมื่อกดปุ่ม B

2.6 การวาดรูปเรขาคณิตขั้นต้น

ในการวาดภาพของ OpenGL นั้นจะใช้คำสั่งพื้นฐานในการวาดซึ่งคำสั่งที่ใช้มีดังนี้

GL-POINTS คือ การวาดจุด

GL_LINES คือ การวาดเส้นระหว่างจุดสองจุด

GL_LINE_STRIP คือ การวาดเส้นระหว่างจุดต่อกันเป็นเส้นยาว

GL_LINE_LOOP คล้ายกับ GL_LINE_STRIP แต่จะวาดเส้นระหว่างจุดจนครบเป็นลูป

GL_TRIANGLES คือ การวาดจุดสามจุดให้เป็นรูปสามเหลี่ยม

GL_TRIANGLE_STRIP คือ การการต่อรูปสามเหลี่ยมให้เป็นแผ่น

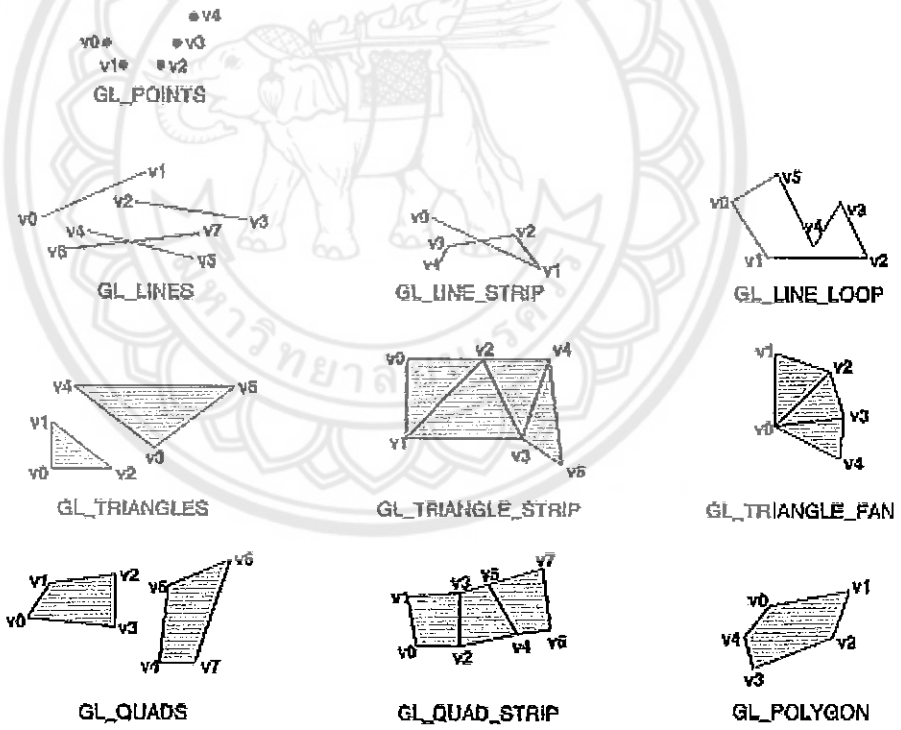
GL_TRIANGLE_FAN คล้ายกับ GL_TRIANGLE_STRIP แต่จะเป็นการการต่อรูปสามเหลี่ยมให้มีลักษณะเหมือนพัด

GL_QUADS คือ การวาดจุดสี่จุดให้เป็นรูปสี่เหลี่ยม

GL_QUAD_STRIP คือ การการนำรูปสี่เหลี่ยมมาต่อกัน

GL_POLYGON คือ การวาดรูปจากจุดจำนวน n จุดให้เป็นรูปเหลี่ยมหลายด้าน

ซึ่งลักษณะการวาดภาพของฟังก์ชันต่างๆดังกล่าวสามารถอธิบายเป็นรูปภาพได้ดังรูปที่ 2.7



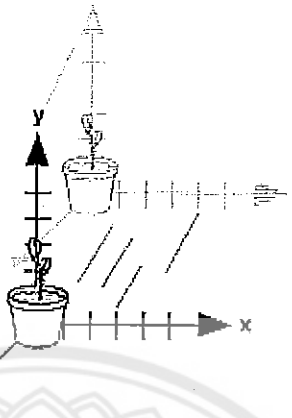
รูปที่ 2.7 ชนิดของรูปเรขาคณิต

2.7 มุมมอง

จะมีฟังก์ชัน GL_MODELVIEW เป็นตัวกำหนดการแสดงผลซึ่งแบ่งออกได้ดังนี้

2.7.1 การTranslate

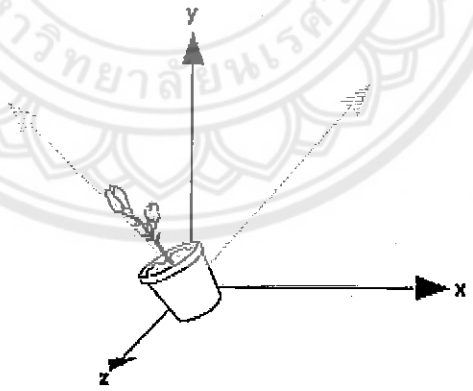
คือ การเลื่อนตำแหน่งของวัตถุไปตามแกน x, y และ z โดยมีฟังก์ชันที่ใช้ คือ void glTranslate {fd}(TYPEx, TYPE y, TYPEz); ซึ่งแสดงการ Translate ของวัตถุได้ดังรูปที่ 2.8



รูปที่ 2.8 การ Translate ของวัตถุ

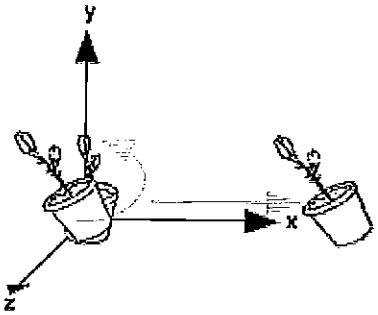
2.7.2 การ Rotate

คือ การหมุนวัตถุไปตามพิกัดที่กำหนดโดยที่วัตถุอยู่ในตำแหน่งเดิมฟังก์ชันที่ใช้ คือ void glRotate{fd} (TYPE angle, TYPE x, TYPE y, TYPE z); เช่นในตัวอย่างจะทำการหมุนวัตถุไปตามแกน z 45 องศา ซึ่งแสดงได้ดังรูปที่ 2.9

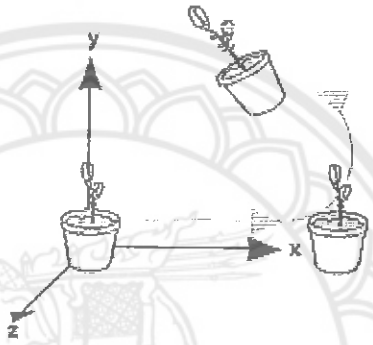


รูปที่ 2.9 ภาพแสดงการ Rotate ของวัตถุ

เราสามารถนำการ Translate และการ Rotate มาใช้ร่วมกัน เช่น ทำการหมุนภาพของวัตถุไปตามแกน z แล้วเลื่อนภาพของวัตถุไปตามแกน x (รูปที่ 2.10) หรือทำการเลื่อนวัตถุไปตามแกน x แล้วทำการหมุนภาพของวัตถุไปตามแกน z (รูปที่ 2.11)



รูปที่ 2.10 ภาพแสดงการ Rotate ก่อนการ Translate



รูปที่ 2.11 ภาพแสดงการ Translate ก่อนการ Rotate

2.8 การโปรเจ็คชันของภาพ

คือ การวางโครงสร้างของภาพให้เหมือนจริงซึ่งจะแบ่งระนาบการมองออกเป็น 6 ด้านโดยมีฟังก์ชันหลักที่ใช้ คือ void glFrustum(GLdouble left, GLdouble right, GLdouble bottom, GLdouble top, GLdouble near, GLdouble far); เมื่อ

GLdouble left คือ ฟังก์ชันในการกำหนดตำแหน่งของภาพทางด้านซ้าย

GLdouble Right คือ ฟังก์ชันในการกำหนดตำแหน่งของภาพทางด้านขวา

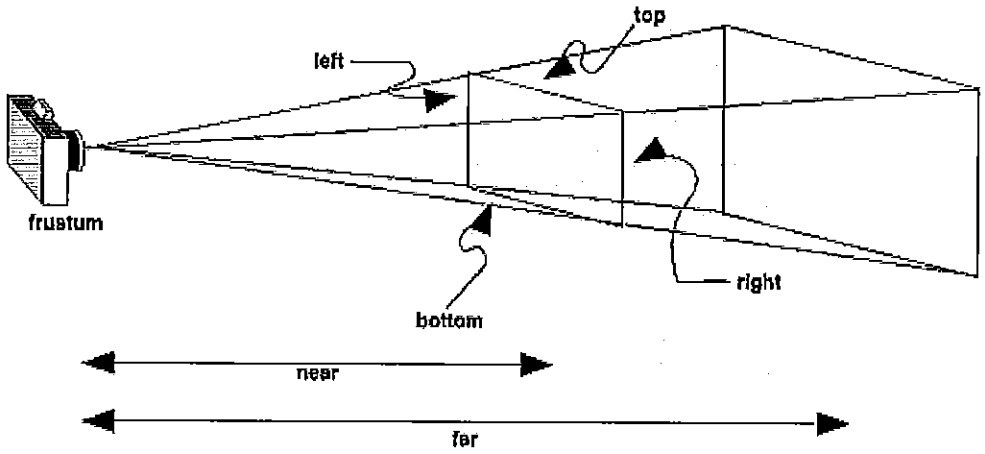
GLdouble bottom คือ ฟังก์ชันในการกำหนดตำแหน่งของภาพทางด้านบน

GLdouble top คือ ฟังก์ชันในการกำหนดตำแหน่งของภาพทางด้านล่าง

GLdouble near คือ ฟังก์ชันในการกำหนดตำแหน่งเริ่มต้นที่ภาพจะปรากฏ

GLdouble far คือ ฟังก์ชันในการกำหนดตำแหน่งสุดท้ายที่ภาพจะปรากฏ

ลักษณะการโปรเจ็คชันของภาพจะเป็นดังรูปที่ 2.12



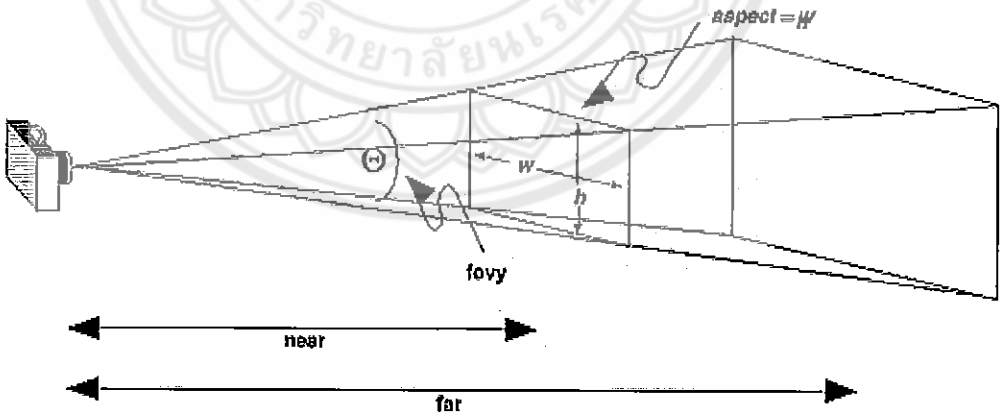
รูปที่ 2.12 ภาพการโปรเจ็คชันของภาพโดยใช้ฟังก์ชัน glFrustum

และมีอีกฟังก์ชันที่ใช้คือ void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far); โดย

GLdouble fovy คือ ฟังก์ชันในการกำหนดมุมในระนาบ x และ y

GLdouble aspect คือ ฟังก์ชันในการกำหนดอัตราส่วนระหว่างความกว้างของภาพต่อความยาวของภาพ

ซึ่งฟังก์ชัน gluPerspective นี้จะใช้เมื่อมีการใช้ภาพที่ซ้ำกันในเวลาที่ใกล้กันหรือในหลายๆ ครั้งที่ฟังก์ชัน glFrustum ไม่สามารถทำได้โดยมีลักษณะการโปรเจ็คชันของภาพจะเป็นดังรูปที่ 2.13



รูปที่ 2.13 ภาพการโปรเจ็คชันของภาพโดยใช้ฟังก์ชัน gluPerspective

2.9 การซ้อนทับบนพื้นผิว (Texture Mapping)

การซ้อนทับบนพื้นผิว (Texture Mapping) คือ การนำภาพมาซ้อนทับกับพื้นผิวของวัตถุ เพื่อให้พื้นผิวของวัตถุดูมีความเหมือนจริงมากขึ้น โดยมีฟังก์ชันที่ใช้ดังนี้

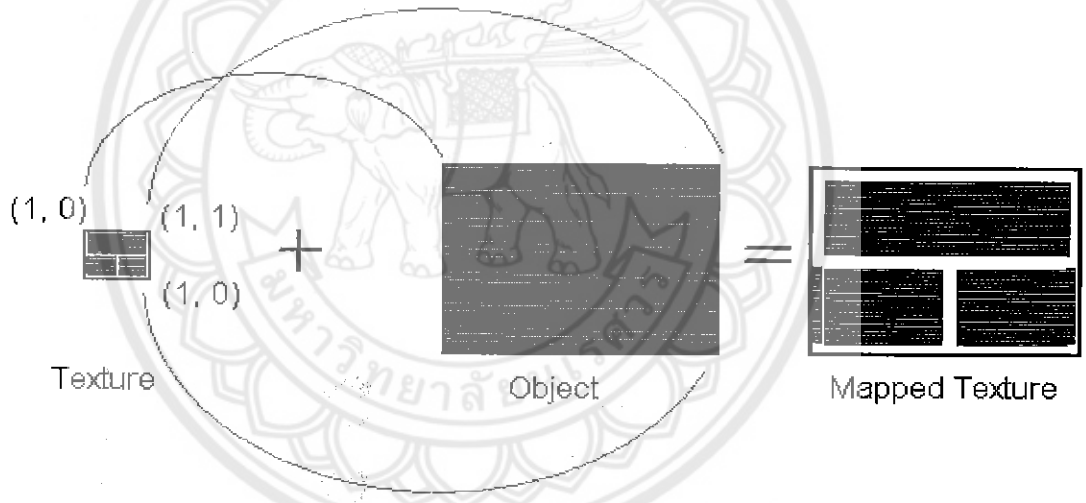
BMP_Texture(TextureArray, "Texture.bmp", 7); //โหลดไฟล์ที่มีชื่อว่า Texture.bmp เก็บไว้ในตัวแปรที่ชื่อว่า TextureArray[7]

glEnable(GL_TEXTURE_2D); //เปิดหมดการทำงานของ Texture

glBindTexture(GL_TEXTURE_2D, TextureArray[7]); //คำสั่งในการกำหนดการใช้ไฟล์ภาพที่ใช้ในการ mapping

```
glBegin(GL_QUADS);
 glTexCoord2d(0.0,1.0); glVertex3f(2.0,0.0,0.0);
 glTexCoord2d(1.0,1.0); glVertex3f(2.0,0.0,-6.0);
 glTexCoord2d(1.0,0.0); glVertex3f(2.0,2.0,-6.0);
 glTexCoord2d(0.0,0.0); glVertex3f(2.0,2.0,0.0);
 glEnd();
```

ซึ่งมีลักษณะการทำงานดังรูปที่ 2.14



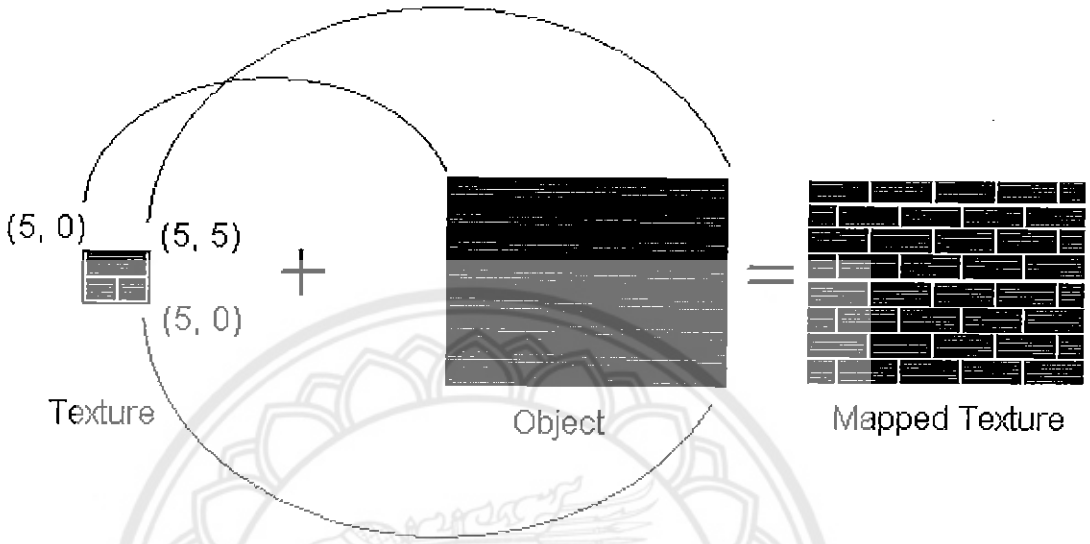
รูปที่ 2.14 Texture mapping

และเพื่อทำให้ภาพที่เรานำมาวางบนวัตถุมีความเหมือนจริงมากขึ้นเราก็สามารถทำการทับซ้อนภาพหลายๆ รอบ อย่างเช่นในตัวอย่างนี้จะเป็นการทำให้ภาพสี่เหลี่ยมดูเหมือนกับภาพของผนังอิฐโดยจะต้องมีการกำหนดค่าของพิกัดให้มีค่ามากกว่า 1.0 เช่นเท่ากับ 5.0 ดังนี้

```
glBegin(GL_QUADS);
glTexCoord2d(0.0, 0.0);  glVertex3f(2.0,0.0,0.0);
glTexCoord2d(5.0, 0.0);  glVertex3d(2.0,0.0,-6.0);
glTexCoord2d(5.0, 5.0);  glVertex3d(2.0,2.0,-6.0);
```

```
    glTexCoord2d(0.0, 5.0); glVertex3d(2.0,2.0,0.0);
  glEnd();
```

ซึ่งมีลักษณะการทำงานดังรูปที่ 2.15



รูปที่ 2.15 Texture mapping ที่มีการใช้พิกัดที่มากกว่า 1.0

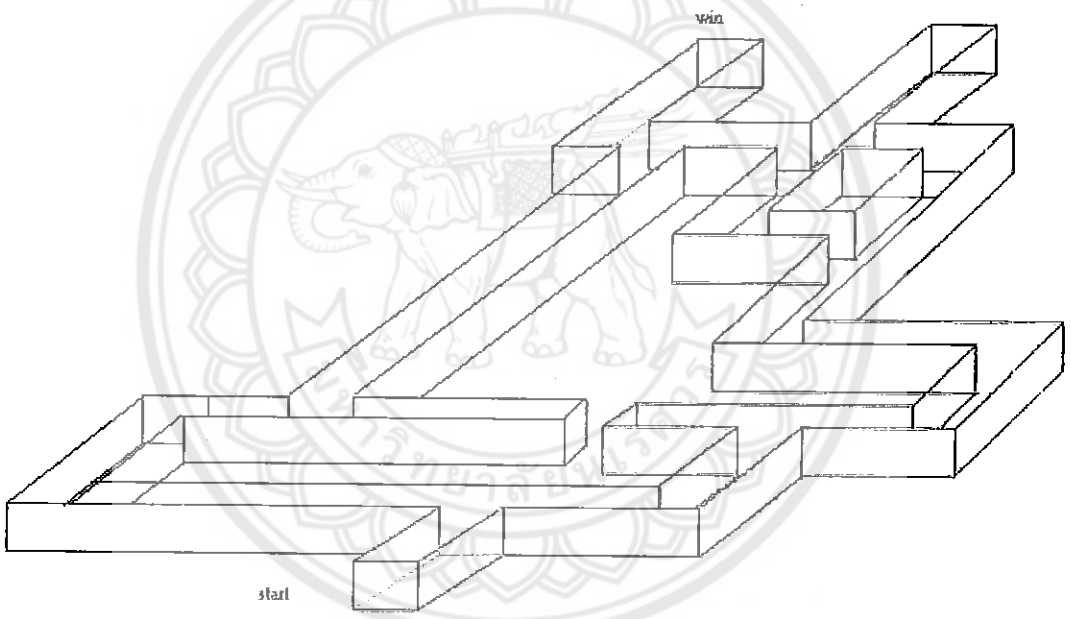
เมื่อมาถึงตอนนี้เราก็สามารถนำความรู้ขั้นต้นดังที่ได้กล่าวมานำไปใช้ในการพัฒนาเกมให้มีลักษณะตามที่เราได้กำหนดไว้แล้วซึ่งสามารถทำการศึกษาได้ในบทต่อไป

บทที่ 3
ขั้นตอนการดำเนินงาน

เมื่อมีการศึกษาทฤษฎีและวิธีการที่จำเป็นต้องใช้ในการพัฒนาโครงงานนี้แล้วขั้นตอนต่อไปก็จะเป็นส่วนของการออกแบบและพัฒนาโปรแกรมโดยมีลำดับขั้นตอนต่างๆ ดังนี้

3.1 การออกแบบภาพรวมของโครงงาน

โครงงานนี้จะเป็นการพัฒนาเกมที่มีลักษณะเป็นเกมเขาวงกตที่ไม่ซับซ้อนซึ่งการออกแบบแผนที่ของเกมนั้นก็จะแสดงได้ดังรูปที่ 3.1

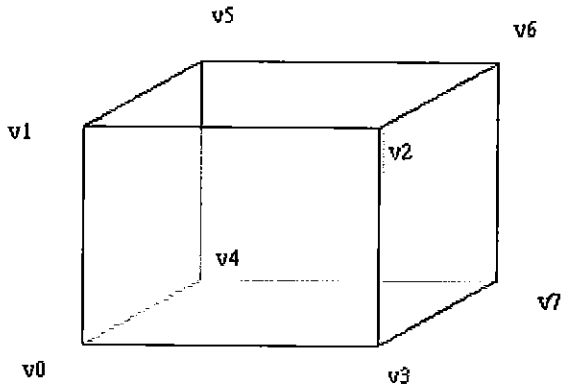


รูปที่ 3.1 ภาพแผนที่ของเกมโดยรวม

จากรูปที่ 3.1 เป็นภาพแผนที่โดยรวมของโครงงานนี้ ซึ่งจะทำโดยใช้ลักษณะของรูปสี่เหลี่ยมลูกบาศก์ต่อกันให้เป็นทางเดิน

3.2 การออกแบบรูปสี่เหลี่ยมลูกบาศก์

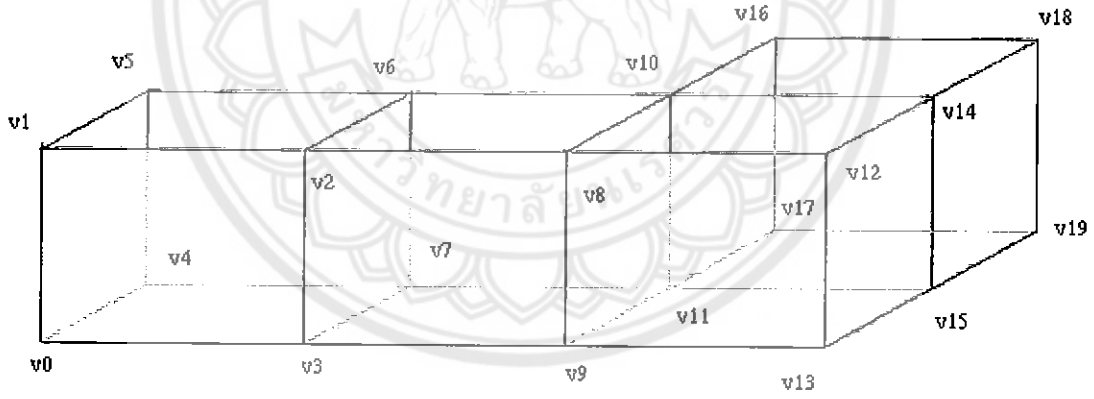
ในการทำรูปสี่เหลี่ยมลูกบาศก์นั้นจะใช้ฟังก์ชัน GL_QUADS ที่เป็นฟังก์ชันที่ใช้ในการวาดรูปสี่เหลี่ยมตามจุดทั้งสี่จุดแล้วนำมาต่อกันให้เป็นรูปสี่เหลี่ยมลูกบาศก์ดังแสดงได้ตามรูปที่ 3.2



รูปที่ 3.2 ภาพสี่เหลี่ยมลูกบาศก์ที่ทำการเชื่อมจุดโดยใช้ฟังก์ชัน GL_QUADS

3.3 การนำรูปสี่เหลี่ยมลูกบาศก์มาต่อกันเป็นทางเดิน

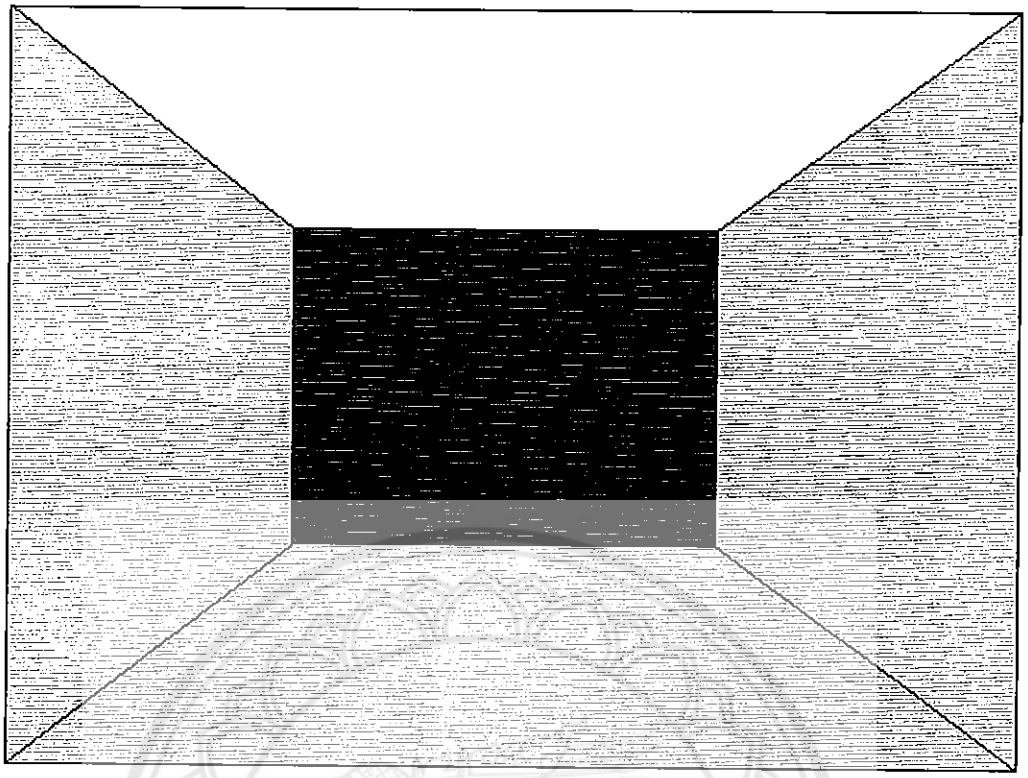
เมื่อเราได้รูปสี่เหลี่ยมลูกบาศก์ก็นำรูปสี่เหลี่ยมลูกบาศก์เหล่านั้นมาทำการเชื่อมต่อกันให้เป็นรูปทางเดิน ดังรูปที่ 3.3



รูปที่ 3.3 ภาพทางเดินที่นำสี่เหลี่ยมลูกบาศก์มาเชื่อมต่อกันโดยใช้ฟังก์ชัน GL_QUADS

3.4 การกำหนดมุมกล้องให้มาอยู่ตรงกลางของเส้นทาง

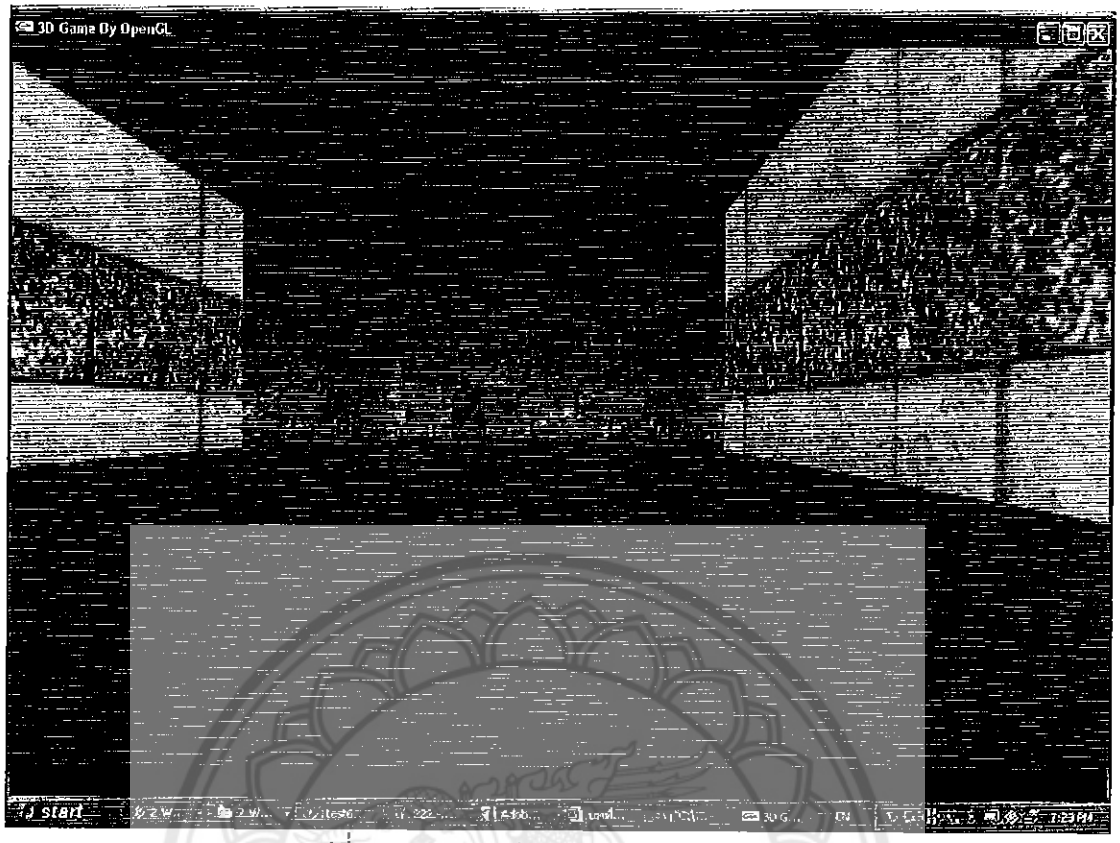
เมื่อทำการสร้างแผนที่โดยรวมเสร็จแล้ว ต่อไปเราก็ทำการกำหนดให้ตัวละคร (มุมมอง) ให้มีการเดินอยู่ภายในกล่อง (ทางเดิน) ที่เราได้สร้างไว้โดยการกำหนดมุมกล้องให้มาอยู่ตรงกลางของเส้นทางนั้นจะใช้ฟังก์ชัน void gluPerspective(GLdouble fovy, GLdouble aspect, GLdouble near, GLdouble far); ซึ่งจะมีลักษณะดังรูปที่ 3.4



รูปที่ 3.4 ภาพตัวอย่างการกำหนดมุมกล้อง

3.5 การ Mapping Texture

เพื่อให้พื้นผิวของวัตถุมีความสมจริงมากยิ่งขึ้นเราจึงมีการนำการ mapping texture มาช่วยในการนำภาพพื้นผิวที่ต้องการมาติดทับไว้บนพื้นผิวของวัตถุโดยนำไฟล์ภาพที่มีนามสกุล .bmp มาวางไว้ที่ folder ที่เก็บไฟล์ .cpp ไว้แล้วทำการโหลดภาพโดย include ไฟล์ BMP.cpp เข้าไปไว้ใน source file เมื่อทำการรันโปรแกรมจะได้ภาพตามรูปที่ 3.5



รูปที่ 3.5 ภาพตัวอย่างการทำ texture mapping

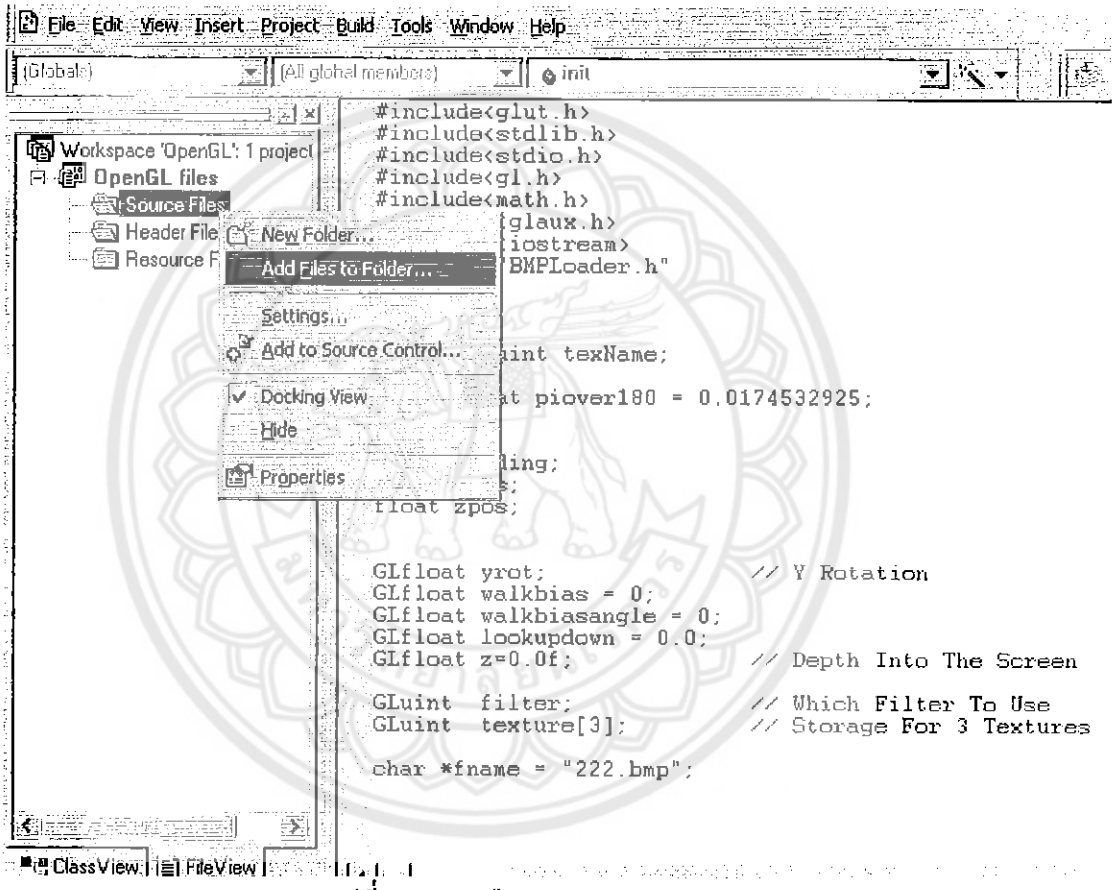
เมื่อมาถึงตอนนี้เราก็จะได้องค์ประกอบของเกมที่มีความสมบูรณ์พร้อมที่จะใช้ในการประยุกต์และใช้ในการพัฒนาเกมได้ ซึ่งสามารถศึกษาได้ในบทต่อไป

บทที่ 4

การทดสอบ

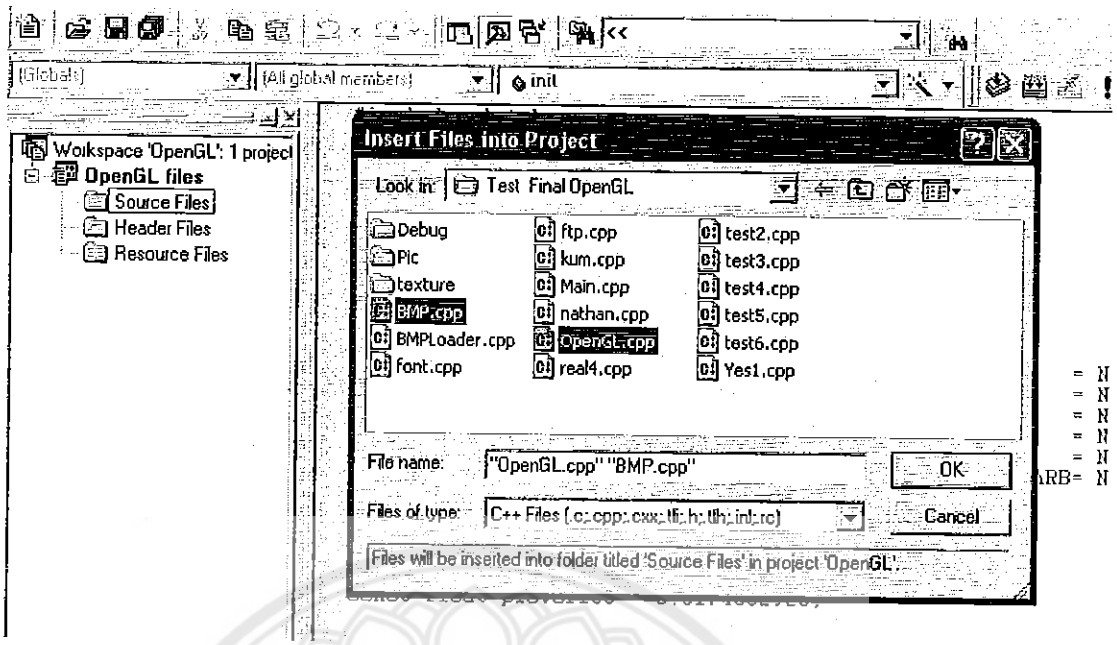
4.1 วิธีการทดสอบ

ทำการ สร้าง Project โดยเลือก File/New แล้วดูที่ FileView ให้คลิ้กขวาที่ folder Source File แล้วเลือก Add Files to folder ดังรูปที่ 4.1



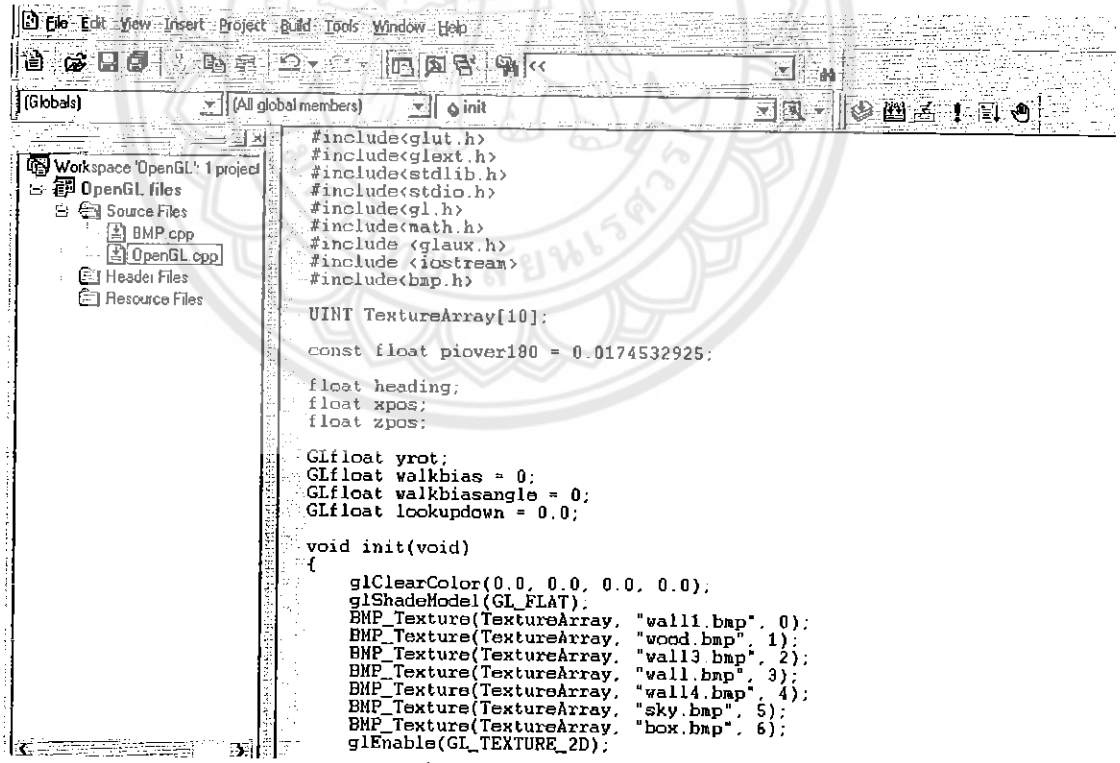
รูปที่ 4.1 การเลือก Source Files to folder

ทำการเลือกไฟล์ BMP.cpp (ไฟล์ที่ใช้ในการโหลดภาพที่มีนามสกุล bmp) และ OpenGL.cpp (ไฟล์หลักของโปรแกรม) แล้วกด OK ดังรูปที่ 4.2



รูปที่ 4.2 การเลือกไฟล์ .cpp

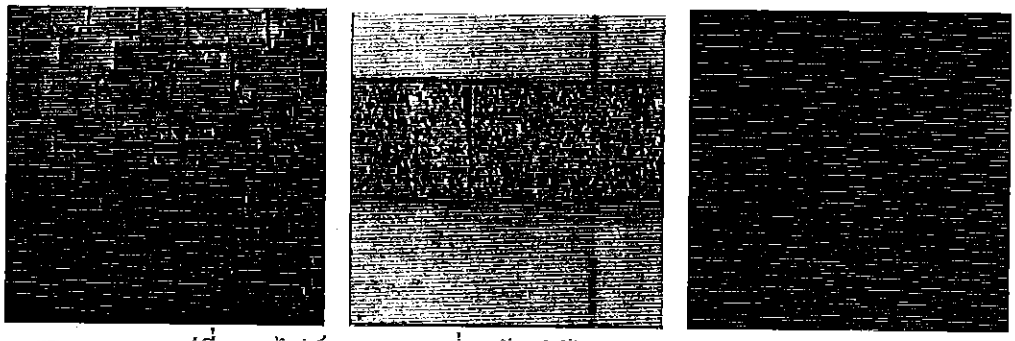
จะได้ไฟล์หลักและไฟล์ที่ใช้ในการโหลดภาพขึ้นมาดังรูปที่ 4.3



รูปที่ 4.3 ภาพแสดงไฟล์ที่เลือก

จากนั้นทำการ save และ compile

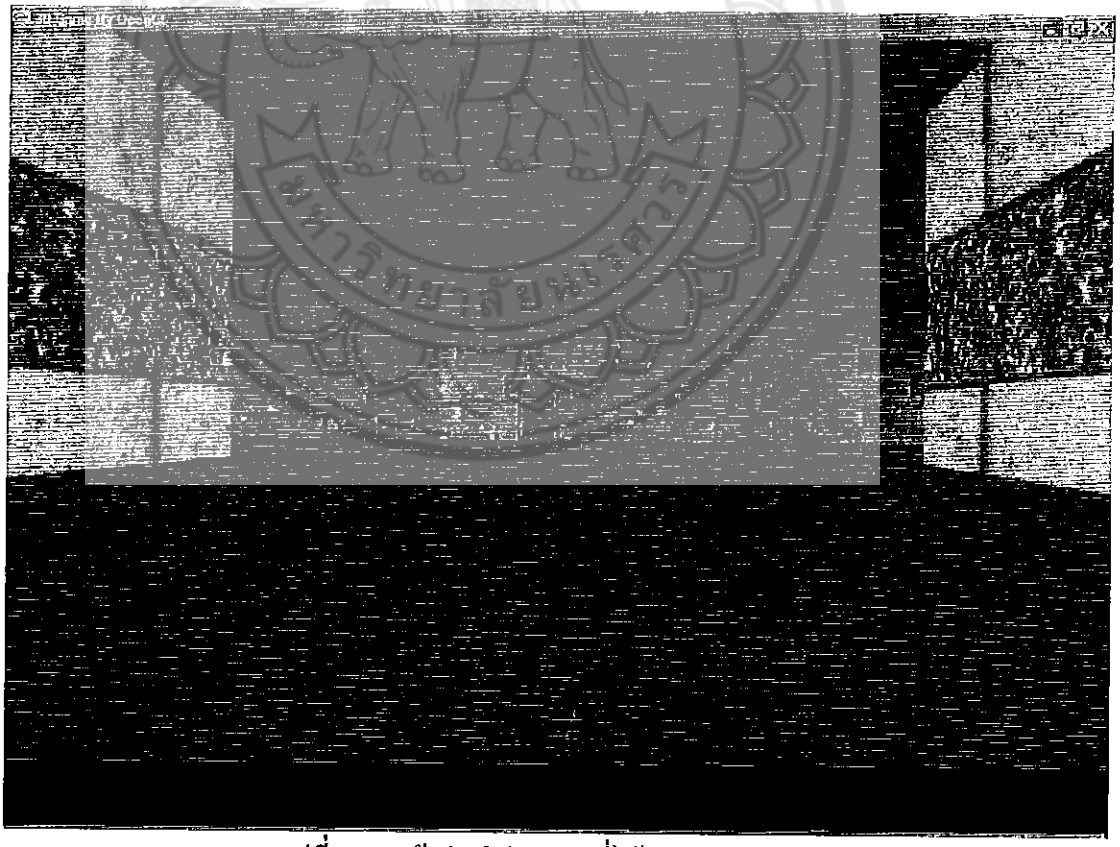
<u>หมายเหตุ</u> โดยก่อนการ compile ต้องนำไฟล์ภาพ .bmp ที่ต้องการใช้ในการ mapping ไปใส่ในโฟลเดอร์ที่เก็บไฟล์หลักของโปรแกรมก่อนเพื่อใช้ในการ mapping



รูปที่ 4.4 ไฟล์ภาพ .bmp ที่จะต้องใช้ในการ mapping

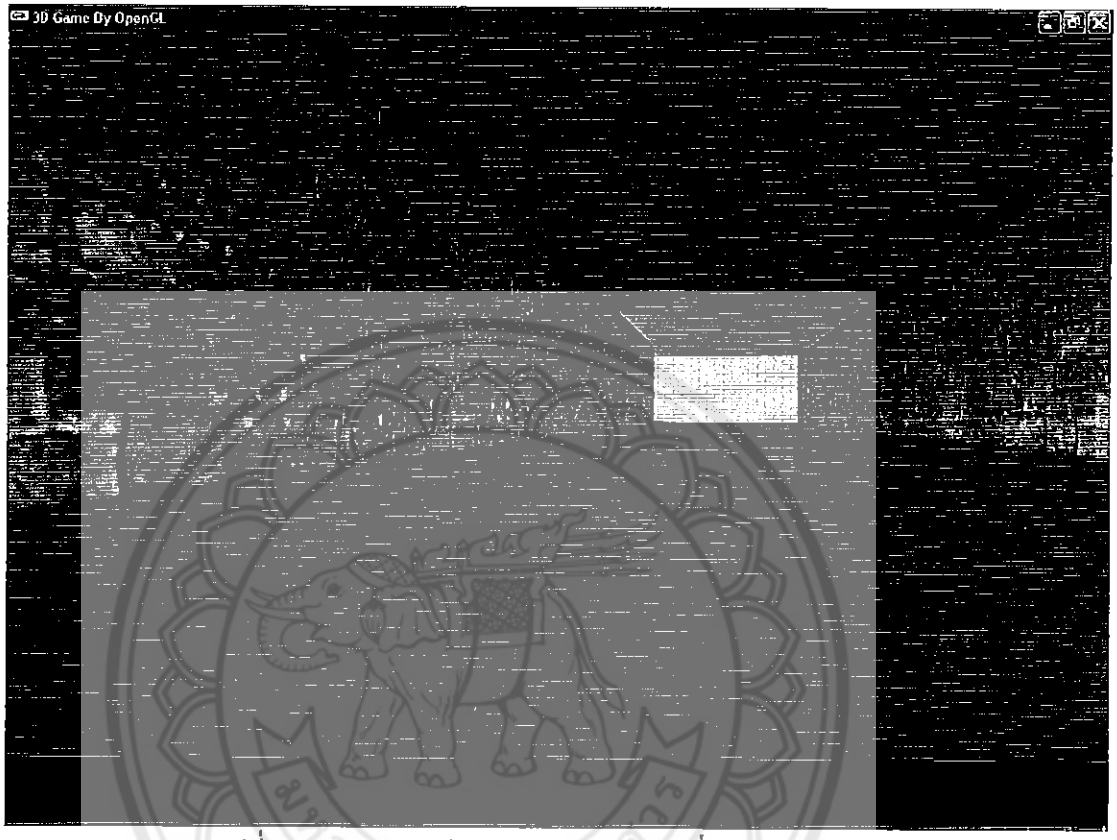
4.2 ผลการทดสอบ

เมื่อทำการ compile และรันโปรแกรมจะได้หน้าต่างขึ้นมาที่มีลักษณะเป็นภาพเส้นทางเริ่มต้นของเกมดังรูปที่ 4.5



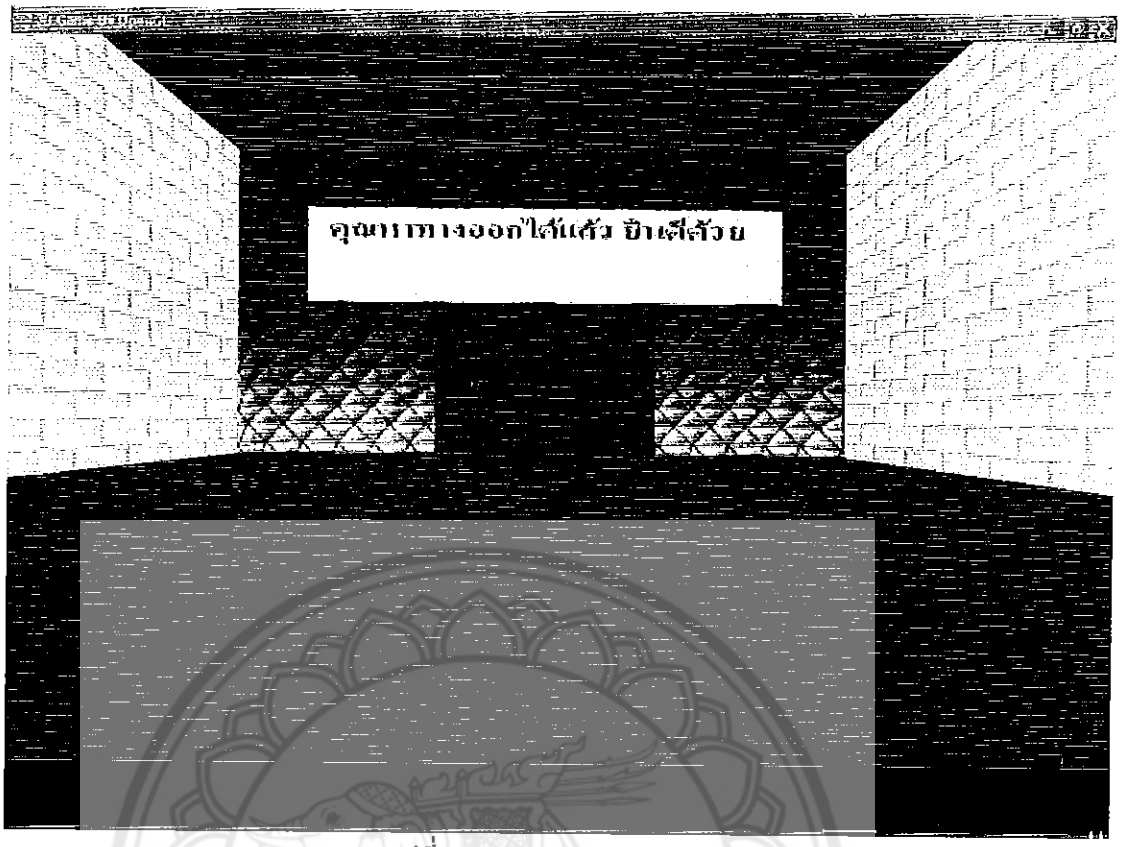
รูปที่ 4.5 หน้าต่างโปรแกรมที่ได้หลังจากการรันโปรแกรม

เมื่อปรากฏหน้าต่างขึ้นมาเราสามารถกดปุ่มต่างๆ (ปุ่ม w เพื่อเดินหน้าปุ่ม s เพื่อถอยหลัง ปุ่ม a เพื่อเลี้ยวซ้ายปุ่ม d เพื่อเลี้ยวขวา) เพื่อที่จะเดินไปตามเส้นทางที่กำหนดเพื่อหาทางออกของเกม ซึ่งแสดงได้ดังรูปที่ 4.6



รูปที่ 4.6 ภาพของเส้นทางภายในของเกมเมื่อมีการเดินเข้าไป

เมื่อเดินมาถึงจุดจุดหนึ่งซึ่งจะเป็นจุดเส้นชัยก็จะสังเกตเห็นที่ผนังจะมีข้อความว่า "คุณหาทางออกได้แล้ว ยินดีด้วย" สามารถทำการเดินชนผนังเพื่อที่จะออกจากเกม ก็จะถือว่าเป็นการสิ้นสุดของเกมนี้ ดังแสดงตามรูปที่ 4.7



รูปที่ 4.7 ภาพทางออกของเกม

บทที่ 5
บทวิจารณ์และสรุป

5.1 บทวิจารณ์

โครงงานนี้เมื่อเริ่มการพัฒนาและทดสอบการใช้งานแล้วได้พบปัญหาต่างๆ ดังนี้

5.1.1 เนื่องจากการพัฒนา OpenGL นี้ต้องการพัฒนาทำบน Microsoft Visual C++ ทำให้ผู้พัฒนาต้องใช้เวลาในการเรียนรู้การใช้งาน Microsoft Visual C++ เพิ่มเติม

5.2 แนวทางแก้ไขและการพัฒนาต่อ

5.2.1 เนื่องจากโครงงานนี้เป็นการพัฒนา OpenGL ในเบื้องต้นการเขียนโปรแกรมจึงใช้ฟังก์ชันที่ค่อนข้างพื้นฐานในการพัฒนาทำให้ตัวโปรแกรมมีความยาว จึงทำให้ยากต่อการแก้ไขและการหาข้อผิดพลาด ดังนั้นจึงควรมีการศึกษาเพิ่มเติมเกี่ยวกับเรื่องเมตริกและการเก็บข้อมูลที่เป็นแบบอาร์เรย์ เพื่อให้การเก็บข้อมูลต่างๆ ของโปรแกรมมีความง่ายและสามารถแก้ไขได้รวดเร็วยิ่งขึ้น

5.2.2 สามารถพัฒนาต่อได้โดยทำการเพิ่มเติมส่วนเชื่อมต่อกับฐานข้อมูล (Data base) ซึ่งจะทำให้เกมมีรูปแบบที่หลากหลายมากขึ้น

5.2.3 เพิ่มอุปกรณ์ที่ใช้ควบคุมเกม เช่น เมาส์และจอยสติ๊ก

5.3 สรุป

โครงงานการพัฒนาเกมสามมิติโดยใช้ OpenGL นี้เป็นโครงงานที่นำ OpenGL มาใช้ในการพัฒนาโปรแกรมเกมแบบ 3 มิติ ซึ่งมีผลการทำงานเป็นไปตามวัตถุประสงค์ที่ตั้งไว้ ดังนี้

5.2.1 จากการศึกษาหลักการและทฤษฎีการทำงานของ OpenGL ที่ใช้ในการพัฒนาเกมประเภท 3 มิติแล้วทำให้สามารถนำความรู้ที่ได้ไปประยุกต์ใช้และพัฒนาเกมได้เป็นอย่างดี

5.2.2 สามารถทำการพัฒนาเกมคอมพิวเตอร์ที่มีการแสดงผลในรูปแบบ 3 มิติและมีการควบคุมโดยใช้คีย์บอร์ดได้

เอกสารอ้างอิง

[1] Mason Woo. Jackie Neider. Tom Davis. Dave Shreiner. **OpenGL Programming Guide**. 3rd Ed.,United State of America. An Imprint of Addision Wesley Longman, Inc. 2000.

[2] Andy McGovern. "A Texture Mapping Technique Using OpenGL." [Online]. Available: http://www.codeguru.com/Cpp/G-M/opengl/texturemapping/print.php/c5589/. April 2, 2003.

[3] OpenGL.org. "The OpenGL Programming Guide - The Redbook." [Online]. Available: http://www.opengl.org/documentation/red_book_1.0/. 2004.

[4] Jeff Molofee. "lesson=01." [Online]. Available: http://nehe.gamedev.net/data/lessons/lesson.asp?lesson=01. 2005.

ภาคผนวก ก.

การ set up OpenGL บน MS Visual C++ 6.0

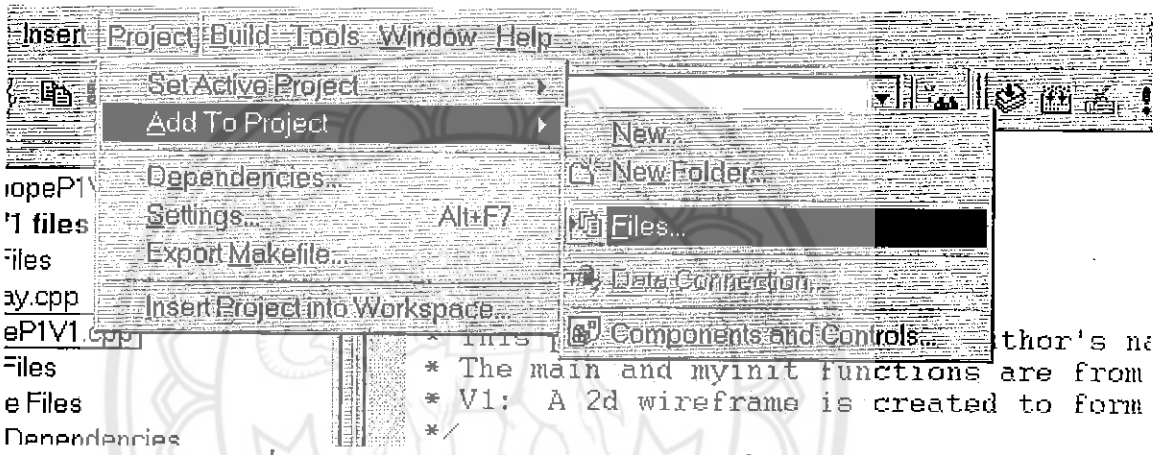
Microsoft Visual C++ 6.0 เป็นเครื่องมือที่นิยมใช้ในการสร้างแอพพิเคชันที่เป็นภาษา C++ สำหรับการใช้ Microsoft Visual C++ 6.0 กับ OpenGL มีดังนี้ สำหรับไฟล์ glut_vc.zip เมื่อแตกไฟล์ออกมาสำหรับ header file (ไฟล์ glut.h) ก็ใส่เข้าไปใน include\GL ซึ่งจะมีไฟล์ gl.h และ glu.h อยู่ก่อนแล้ว copy ไฟล์ glut32.lib ใส่ในไดเรกทอรี่ lib และต้อง copy ไฟล์ชื่อ glut32.dll ลงในไดเรกทอรี่ที่เก็บไฟล์ที่ compile แล้ว หรือใส่ไว้ใน windows\system หลังจากนั้นในการสร้าง project ต้องทำดังนี้

1. เลือกสร้างโปรเจ็คแบบ Win32 Application
2. เลือกเมนู project จากเมนูหลัก
3. เลือก Link tab จาก dialog box
4. เลือก Output จาก Category combo box
5. ใส่ mainCRTStartup ในช่อง Entry-point symbol
6. เลือก General จาก Category combo box
7. ใส่ opengl32.lib glut32.lib glu32.lib ในช่อง Object/library modules
8. นำตัวอย่างโปรแกรมมาใส่แล้วสามารถ compile และ run ได้

ภาคผนวก ข.

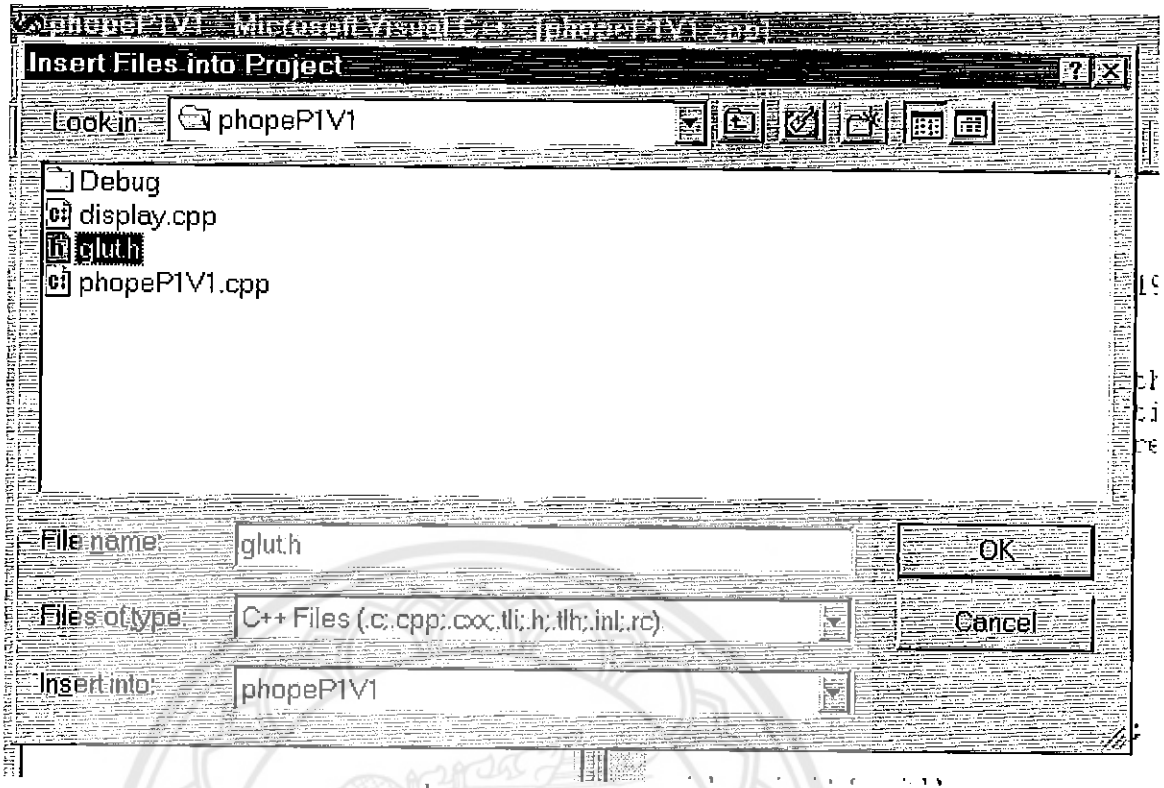
COMPILE ERRORS

ในกรณีที่เกิดการ Error หลังการ Compile แล้วมีข้อความขึ้นมาว่า "fatal error C1083: Cannot open include file: 'gl/glut.h': No such file or directory" แสดงว่าโปรแกรมไม่สามารถหาที่อยู่ของ header files ที่เราเรียกใช้ได้ ดังนั้นเราต้องกำหนดที่อยู่ของ header files โดยการ Copy file ที่มีชื่อว่า glut.h มาไว้ยัง directory ที่ไฟล์ .cpp ใช้ compile อยู่โดยเลือกที่เมนู Project จากเมนูบาร์แล้วเลือก Add- to Project แล้ว Files. ดังรูปที่ ข.- 1



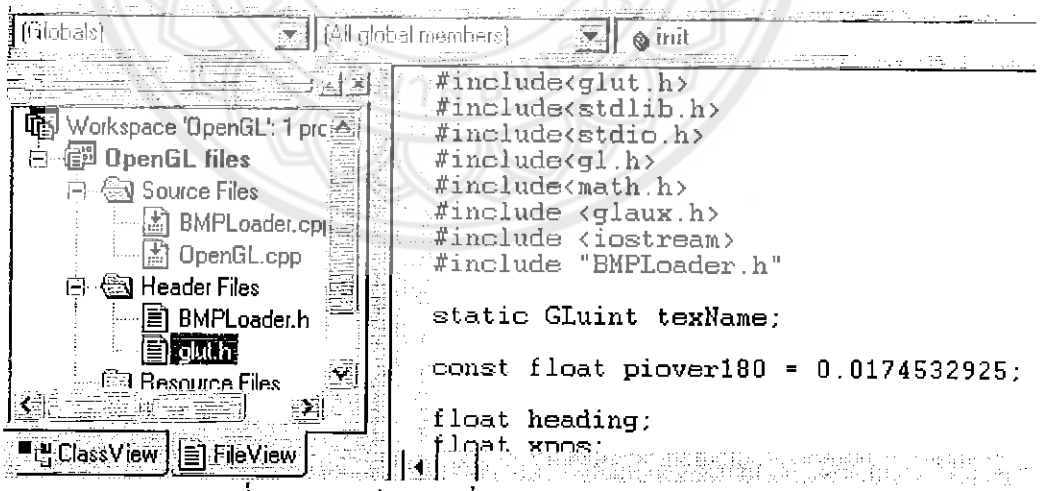
รูปที่ ข.- 1 การเลือกเมนู Project จากเมนูบาร์เพื่อ Add Files

จากนั้นจะมีหน้าต่าง Insert Files into Project ขึ้นมาถ้ามีไฟล์ glut.h อยู่แล้วก็ทำการคลิ๊กเลือกแล้วกด OK ดังรูปที่ ข.- 2



รูปที่ ข.- 2 การเลือกไฟล์ glut.h

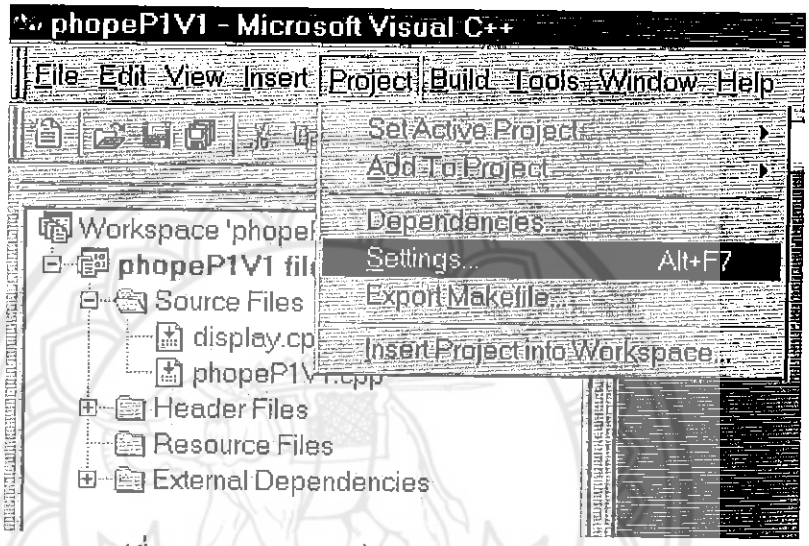
แล้วในหน้าต่าง "File View" ก็จะปรากฏไฟล์ glut.h ขึ้นมาแล้วก็ทำการ compile ได้ทันที ดัง-รูปที่ ข.- 3



รูปที่ ข.- 3 ไฟล์ glut.h ที่ปรากฏอยู่ในหน้าต่าง File View

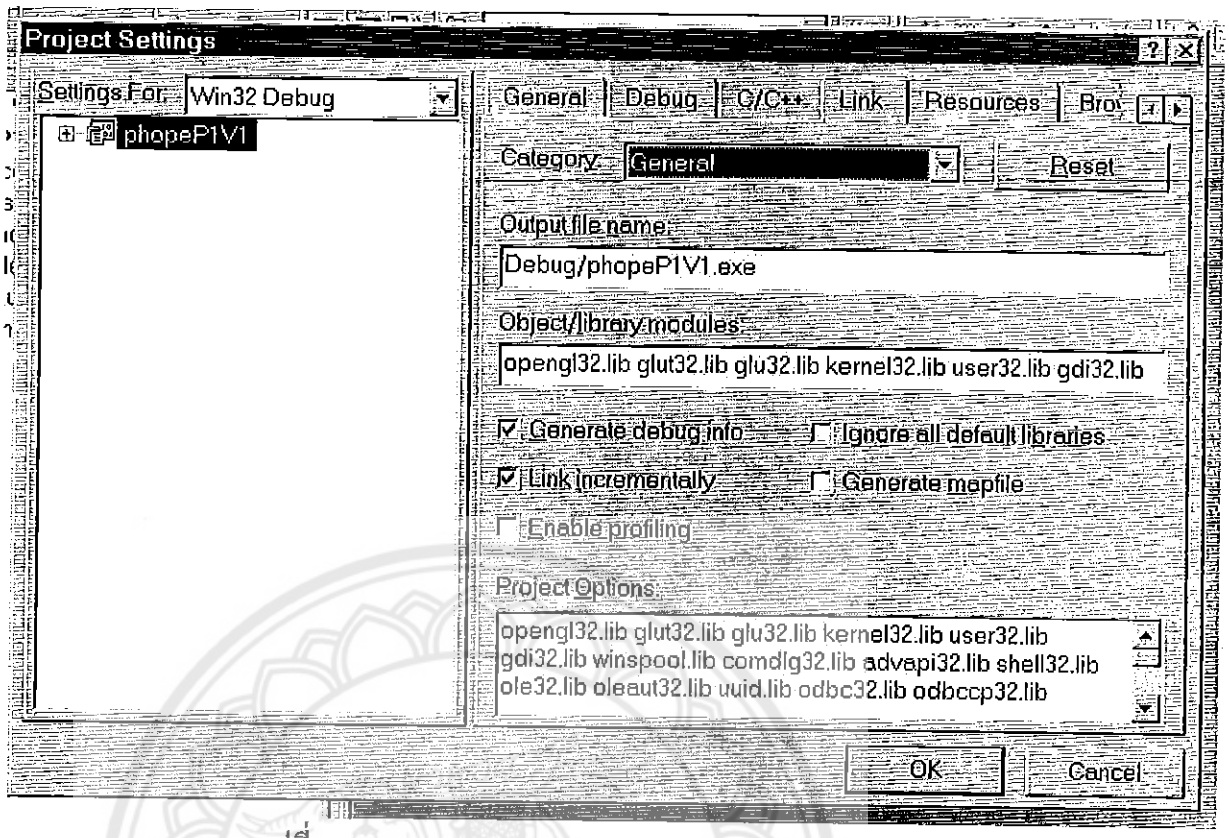
LINK ERROR

ในกรณีที่เกิดการ Error แล้วมีข้อความขึ้นมาว่า "error LNK2001: unresolved external symbol __imp __glFlush@0" แสดงว่ายังไม่ได้ตั้งค่าไลบรารี หรือกำหนดที่อยู่ของไฟล์ .lib ที่แน่นอน หรือไม่มีไฟล์ .lib ใน directory C:\Program Files\Microsoft Visual Studio\VC98\Lib โดยทำการแก้ไขจากเมนู-บาร์ของ visual studio โดยเลือกที่เมนูบาร์ Project แล้วเลือก Settings ดังรูปที่ ข.- 4



รูปที่ ข.- 4 ภาพเมนูบาร์การเลือก Project และ Settings

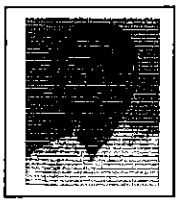
จะปรากฏ dialog box Project Settings ขึ้นมาดังรูปที่ ข.- 5



รูปที่ ข.- 5 ภาพ dialog box ของเมนู Project Settings

กดเลือกที่ปุ่ม link แล้วดูที่ช่อง Object/library modules ว่ามีคำว่า opengl32.lib glut32.lib glu32.lib หรือไม่ถ้าไม่มีก็ให้ใส่ลงไปแล้วกด ok ทำการ compile/link/execute โปรแกรมได้ทันที

ประวัติผู้เขียนโครงงาน



ชื่อ นายธงชัย แก้วเพียร

วันเดือนปีเกิด 3 มิถุนายน 2526

ที่อยู่ 280 ม.9 ต.บ้านเกาะ อ.เมือง จ.อุตรดิตถ์

โทรศัพท์ -

ประวัติการศึกษา

- สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนน้ำ-พองศึกษา
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร

ชื่อ นายรัฐพล ชัยนนถี

วันเดือนปีเกิด 3 พฤษภาคม 2526

ที่อยู่ 219/5 ม.8 ต.มะต้อง อ.พรหมพิราม จ.พิษณุโลก

โทรศัพท์ -

ประวัติการศึกษา

- สำเร็จการศึกษาระดับมัธยมศึกษาตอนปลายจากโรงเรียนพิษณุโลกพิทยาคม
- ปัจจุบันกำลังศึกษาในระดับปริญญาตรีชั้นปีที่ 4 สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร