МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №4**

**з дисципліни «Комп’ютерна графіка»**

**на тему:** **«Тривимірна графіка з використанням бібліотеки OpenGL»**

Виконав: студент гр. ПЗ2011

Кулик С. В.

Прийняла: Шаповал І. В.

.

Дніпро, 2022

**Тема:** Тривимірна графіка з використанням бібліотеки OpenGL.

**Мета:**Вивчення механізмів OpenGL для формування тривимірних зображень.

**Завдання**

Створити тривимірне зображення, що повторює плоске зображення з лаб. роботи №2. Виконати афінні перетворення отриманого зображення використовуючи засоби бібліотеки OpenGL, реалізувати поворот і масштабування усієї отриманої фігури. Включити та налаштувати освітленість об'єктів.

**Опис використаних в програмі графічних засобів бібліотеки OpenGL.**

Gl.Scale – Функція, що проводить операції масштабування.

GL.Translate – Функція, що проводить операції переміщення.

GL.Rotate – Функція, що проводить операції повороту.

GL.Clear – Функція, для очищення буферу.

GL.ClearColor – Функція, для очищення та встановлення кольору фону.

GL.Enable – Функція, для підключення різних налаштувань.

GL.LoadIdentity – Функція, для завантаження даних.

GL.LineWidth – Функція, для встановлення ширини ліній.

GL.PolygonMode – Функція, для встановлення режиму відображення полігонів.

GL.BlendFunc – Функція, для встановлення графічних режимів.

GL.Disable – Функція, для відключення налаштувань.

GL.Vertex – Функція, для створення точки за координатами.

GL.Begin – Функція, для вказування початку встановлення точок фігури

GL.End – Функція, для вказування завершення встановлення точок фігури.

GL.Light – Функція, для включення світла (та налаштування його).

GL.LightingModel – Функція, для налаштування режиму світла.

GL.MatrixMode – Функція, для встановлення режиму відображення матриці.

GL.Perspective – Функція, для встановлення перспективи.

GL.LookAt – Функція, для встановлення позиції камери.

GL.Color – Функція, для встановлення кольору полігонів (всіх, що об’явлені нижче функції).

**Текст програми**

***MainWindow.xaml.cs***

using SharpGL;

using System.Windows;

namespace CG4

{

public partial class MainWindow : Window

{

private float[] LightingParams = { 0.0f, 0.0f, 0.0f, 1.0f };

private Figure Figure;

private OpenGL gl;

public MainWindow()

{

InitializeComponent();

gl = OpenGLControl.OpenGL;

Figure = new Figure(gl);

}

private void Transform()

{

gl.Scale(Figure.Size[0], Figure.Size[1], Figure.Size[2]); // Масштабування

gl.Translate(Figure.Position[0], -Figure.Position[1], Figure.Position[2]); // Переміщення

gl.Rotate(Figure.Angle[0], Figure.Angle[1], Figure.Angle[2]); // Поворот

}

private void OpenGLControl\_OpenGLDraw(object sender, SharpGL.WPF.OpenGLRoutedEventArgs args)

{

Init();

EnableLighting();

DrawGrid();

CheckValue();

Transform();

Figure.Draw();

}

private void Init()

{

gl.Clear(OpenGL.GL\_COLOR\_BUFFER\_BIT | OpenGL.GL\_DEPTH\_BUFFER\_BIT); // Очищення буферу

gl.ClearColor(0.6f, 0.6f, 0.6f, 1.0f); // Встановлення кольору фону

gl.Enable(OpenGL.GL\_COLOR\_MATERIAL); // Підключення матеріалів

gl.Enable(OpenGL.GL\_DEPTH\_TEST); // Підключення глубини різкості

gl.LoadIdentity();

}

private void CheckValue()

{

LightingParams[0] = (float)LightPower.Value;

LightingParams[1] = (float)LightPower.Value;

LightingParams[2] = (float)LightPower.Value;

}

private void DrawGrid()

{

gl.LineWidth(1);

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_LINE); // Малюування фігури лініями

gl.Enable(OpenGL.GL\_BLEND); // Підключення BlendMode (для прозорості)

gl.BlendFunc(OpenGL.GL\_SRC\_ALPHA, OpenGL.GL\_ONE\_MINUS\_SRC\_ALPHA); // Підключення прозорості

// Малювання сітки

for (int i = -20; i < 20; i++)

{

for (int j = -20; j < 20; j++)

{

gl.Begin(OpenGL.GL\_QUAD\_STRIP);

gl.Color(0.8f, 0.8f, 0.8f, 0.4f);

gl.Vertex(0.0f + j, 0 + i, 0.0f);

gl.Vertex(0.0f + j, -1.0f + i, 0.0f);

gl.Vertex(-1.0f + j, 0.0f + i, 0.0f);

gl.Vertex(-1.0f + j, -1.0f + i, 0.0f);

gl.End();

}

}

gl.Enable(OpenGL.GL\_LINE\_STIPPLE); // Підключення ліній

gl.Disable(OpenGL.GL\_BLEND); // Відключення BlendMode

gl.PolygonMode(OpenGL.GL\_FRONT\_AND\_BACK, OpenGL.GL\_FILL); // Малювання заповненної фігури

}

private void EnableLighting()

{

gl.Enable(OpenGL.GL\_LIGHTING); // Підключення глобального освітлення

gl.Enable(OpenGL.GL\_LIGHT0); // Підключення світла (перший тип, Ambient)

gl.Enable(OpenGL.GL\_LIGHTING); // Підключення глобального освітлення

gl.Light(0, OpenGL.GL\_AMBIENT, 10); // Підключення ліхтарика

gl.LightModel(OpenGL.GL\_LIGHT\_MODEL\_TWO\_SIDE, OpenGL.GL\_TRUE); // Режим змішування

gl.Light(OpenGL.GL\_LIGHT0, OpenGL.GL\_POSITION, LightingParams); // Включення світла за режимом змішування

}

private void OpenGLControl\_Resized(object sender, SharpGL.WPF.OpenGLRoutedEventArgs args)

{

gl.MatrixMode(OpenGL.GL\_PROJECTION);

gl.LoadIdentity();

// Трансформування перспективи. Параметри:

// Віддаленність від фігури

// Співвідношення сторін

// Відстань до найближчої площини відсікання (не чіпати, завжди позитивне)

// Відстань до дальної площини відсікання (не чіпати, завжди позитивне)

gl.Perspective(80.0f, (double)Width / (double)Height, 0.01, 100.0);

// Позиція та точка погляду камери

gl.LookAt(-7, 7, -3, 0, 0, 0, 0, 0, -5);

gl.MatrixMode(OpenGL.GL\_MODELVIEW);

}

private void RotatePlus\_Click(object sender, RoutedEventArgs e)

{

if(XCoordBox.IsChecked == true)

Figure.Angle[0] += 3.0f;

if (YCoordBox.IsChecked == true)

Figure.Angle[1] += 3.0f;

if (ZCoordBox.IsChecked == true)

Figure.Angle[2] += 3.0f;

}

private void RotateMinus\_Click(object sender, RoutedEventArgs e)

{

if (XCoordBox.IsChecked == true)

Figure.Angle[0] -= 3.0f;

if (YCoordBox.IsChecked == true)

Figure.Angle[1] -= 3.0f;

if (ZCoordBox.IsChecked == true)

Figure.Angle[2] -= 3.0f;

}

private void ScalePlus\_Click(object sender, RoutedEventArgs e)

{

if (XCoordBox.IsChecked == true)

Figure.Size[0] -= 0.5f;

if (YCoordBox.IsChecked == true)

Figure.Size[1] -= 0.5f;

if (ZCoordBox.IsChecked == true)

Figure.Size[2] -= 0.5f;

}

private void ScaleMinus\_Click(object sender, RoutedEventArgs e)

{

if (XCoordBox.IsChecked == true)

Figure.Size[0] += 0.5f;

if (YCoordBox.IsChecked == true)

Figure.Size[1] += 0.5f;

if (ZCoordBox.IsChecked == true)

Figure.Size[2] += 0.5f;

}

private void MovePlus\_Click(object sender, RoutedEventArgs e)

{

if (XCoordBox.IsChecked == true)

Figure.Position[0] += 0.5f;

if (YCoordBox.IsChecked == true)

Figure.Position[1] += 0.5f;

if (ZCoordBox.IsChecked == true)

Figure.Position[2] += 0.5f;

}

private void MoveMinis\_Click(object sender, RoutedEventArgs e)

{

if (XCoordBox.IsChecked == true)

Figure.Position[0] -= 0.5f;

if (YCoordBox.IsChecked == true)

Figure.Position[1] -= 0.5f;

if (ZCoordBox.IsChecked == true)

Figure.Position[2] -= 0.5f;

}

}

}

***Figure.cs***

using SharpGL;

namespace CG4

{

public class Figure

{

public float[] Angle = { 0.0f, 0.0f, 0.0f };

public float[] Size = { 1.0f, 1.0f, 1.0f };

public float[] Position = { 0.0f, 0.0f, 0.0f };

OpenGL gl;

public Figure(OpenGL gl)

{

this.gl = gl;

}

public void Draw()

{

//стены

gl.Begin(OpenGL.GL\_QUADS);

gl.Color(1.0f, 0.5f, 0.084f);

//bottom

gl.Vertex(-4.0f, -4.0f, 0.0f);

gl.Vertex(-4.0f, 4.0f, 0.0f);

gl.Vertex(4.0f, 4.0f, 0.0f);

gl.Vertex(4.0f, -4.0f, 0.0f);

//top

gl.Vertex(-4.0f, -4.0f, -4.0f);

gl.Vertex(-4.0f, 4.0f, -4.0f);

gl.Vertex(4.0f, 4.0f, -4.0f);

gl.Vertex(4.0f, -4.0f, -4.0f);

//front

gl.Vertex(-4.0f, -4.0f, 0.0f);

gl.Vertex(-4.0f, -4.0f, -4.0f);

gl.Vertex(4.0f, -4.0f, -4.0f);

gl.Vertex(4.0f, -4.0f, 0.0f);

//back

gl.Vertex(4.0f, 4.0f, 0.0f);////

gl.Vertex(4.0f, 4.0f, -4.0f);

gl.Vertex(-4.0f, 4.0f, -4.0f);

gl.Vertex(-4.0f, 4.0f, 0.0f);

//left

gl.Vertex(-4.0f, -4.0f, 0.0f);

gl.Vertex(-4.0f, -4.0f, -4.0f);

gl.Vertex(-4.0f, 4.0f, -4.0f);

gl.Vertex(-4.0f, 4.0f, 0.0f);

//right

gl.Vertex(4.0f, -4.0f, 0.0f);

gl.Vertex(4.0f, -4.0f, -4.0f);

gl.Vertex(4.0f, 4.0f, -4.0f);

gl.Vertex(4.0f, 4.0f, 0.0f);

gl.End();

//крыша

gl.Begin(OpenGL.GL\_TRIANGLES);

gl.Color(0.38f, 0.38f, 0.38f);

gl.Vertex(-4.0f, -4.0f, -4.0f);

gl.Vertex(0.0f, 0.0f, -7.0f);

gl.Vertex(4.0f, -4.0f, -4.0f);

gl.Vertex(4.0f, 4.0f, -4.0f);

gl.Vertex(0.0f, 0.0f, -7.0f);

gl.Vertex(-4.0f, 4.0f, -4.0f);

gl.Vertex(4.0f, -4.0f, -4.0f);

gl.Vertex(0.0f, 0.0f, -7.0f);

gl.Vertex(4.0f, 4.0f, -4.0f);

gl.Vertex(-4.0f, 4.0f, -4.0f);

gl.Vertex(0.0f, 0.0f, -7.0f);

gl.Vertex(-4.0f, -4.0f, -4.0f);

gl.End();

//дверь

gl.Begin(OpenGL.GL\_QUADS);

gl.Color(0.6f, 0.34f, 0.28f);

gl.Vertex(1.0f, 4.01f, 0.0f);

gl.Vertex(1.0f, 4.01f, -3.0f);

gl.Vertex(-1.0f, 4.01f, -3.0f);

gl.Vertex(-1.0f, 4.01f, 0.0f);

gl.End();

gl.Begin(OpenGL.GL\_QUADS);

gl.Color(0.37f, 0.6f, 0.85f);

//вікна

gl.Vertex( -2f, 4.01f, -1.3f);

gl.Vertex( -2f, 4.01f, -2.7f);

gl.Vertex(-3f, 4.01f, -2.7f);

gl.Vertex(-3.0f, 4.01f, -1.3f);

gl.Vertex( 2f, 4.01f, -1.3f);

gl.Vertex( 2f, 4.01f, -2.7f);

gl.Vertex( 3f, 4.01f, -2.7f);

gl.Vertex( 3.0f, 4.01f, -1.3f);

gl.End();

//лінії

gl.LineWidth(3);

gl.Begin(OpenGL.GL\_LINES);

gl.Color(0.0f, 0.0f, 0.0f);

gl.Vertex(2.5f, 4.02f, -1.3f);

gl.Vertex(2.5f, 4.02f, -2.7f);

gl.Vertex(2f, 4.02f, -2f);

gl.Vertex(3f, 4.02f, -2f);

gl.Vertex(-2.5f, 4.02f, -1.3f);

gl.Vertex(-2.5f, 4.02f, -2.7f);

gl.Vertex(-2f, 4.02f, -2f);

gl.Vertex(-3f, 4.02f, -2f);

gl.Vertex(0.35f, 4.02f, -0f);

gl.Vertex(0.35f, 4.02f, -3f);

gl.Vertex(-0.35f, 4.02f, -0f);

gl.Vertex(-0.35f, 4.02f, -3f);

gl.End();

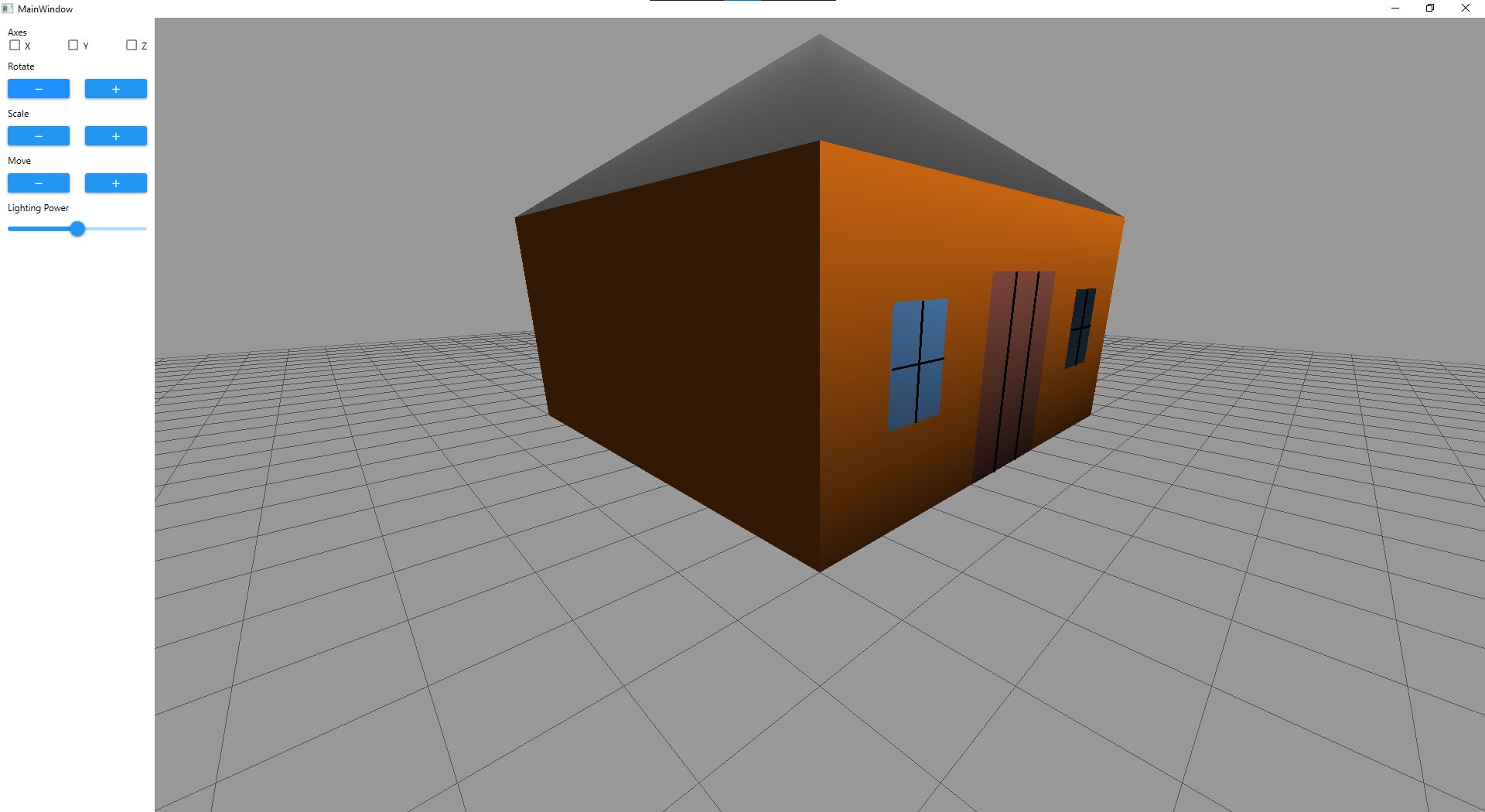
gl.Flush();

}

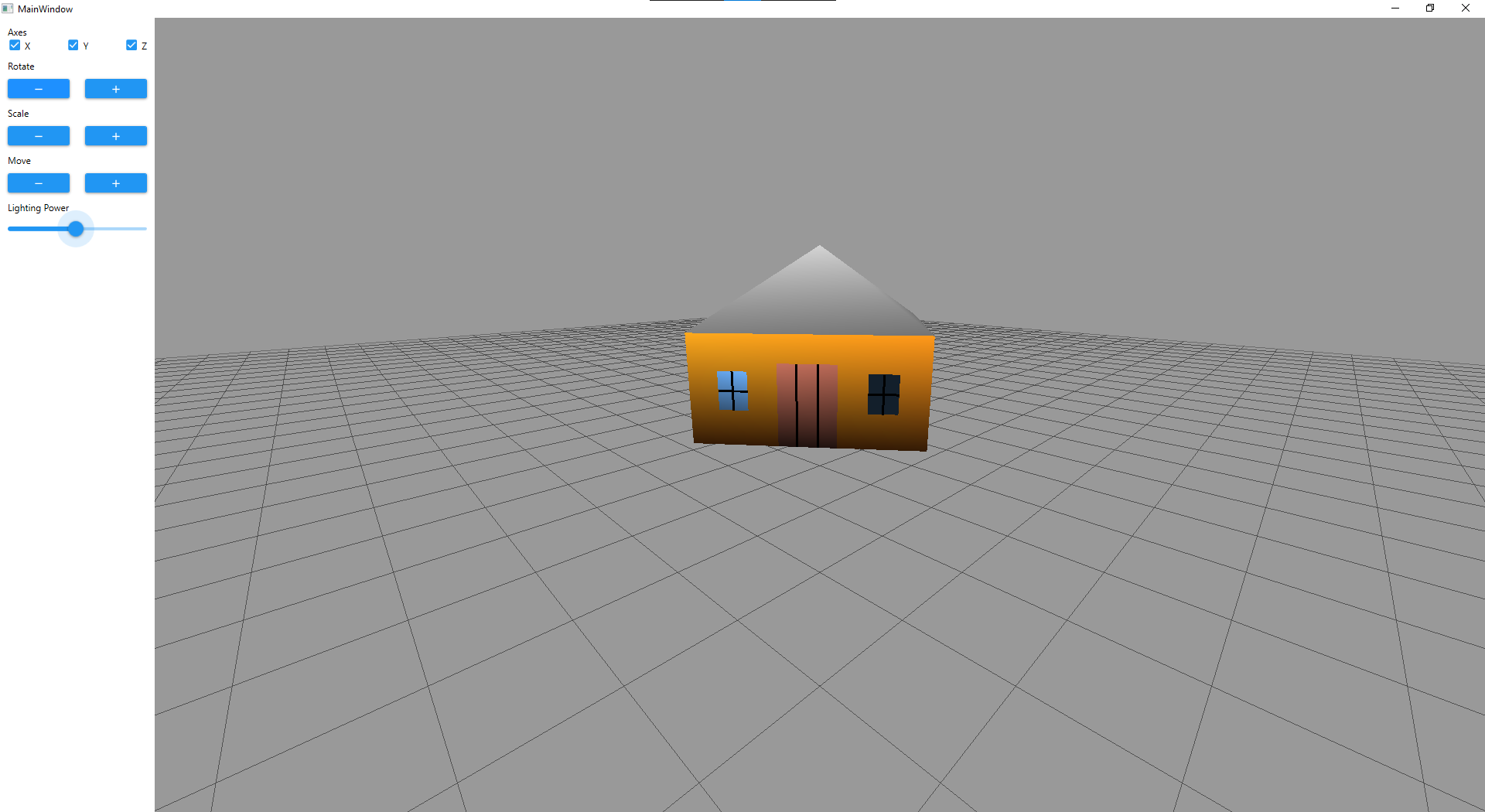
}

}

**Демнстрація роботи програми**

****

*Рис. 1 – Початковий вигляд фігури*

****

*Рис. 2 – Вигляд фігури після масштабквання та повороту*

****

*Рис. 3 – Вигляд фігури після переміщення*

**Аналіз результатів**

Програма правильно виконує дії над фігурою, такі як поворот, переміщення та масштабування.

**Висновки**

Під час виконання лабораторної роботи, познайомився з графічною бібліотекою OpenGL, котра також була перероблена під мову програмування C# і була названа SharpGL. По способу написання коду бібліотека дуже схожа на версію для C++, тому документація майже однакова.

У моїй програмі були реалізовані афінні перетворення над 3D фігурою, такі як масштабування, поворот і переміщення, проте це лише мала частина від того, що можна робита за допомогою бібліотеки OpenGL.