МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №4**

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

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Нежуміра О. І.

Дніпро, 2020

**Лабораторна робота №4**

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

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

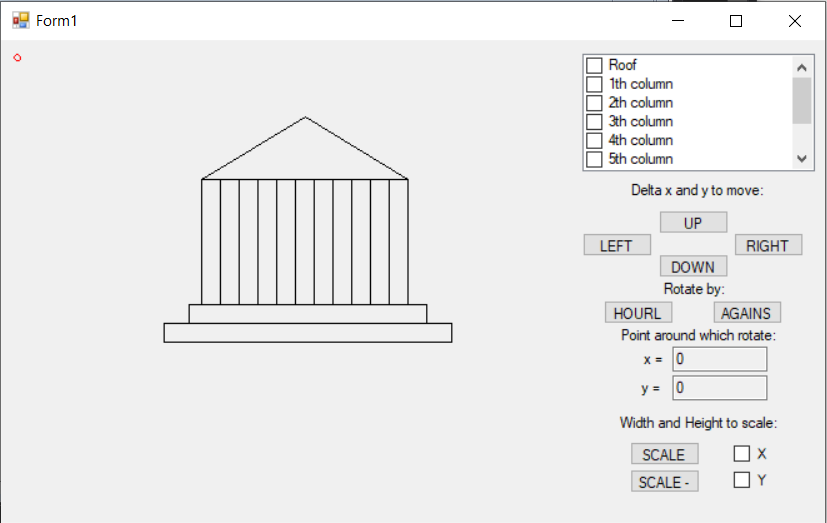
**Постановка задачі**

**Загальне завдання**

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

**Індивідуальне завдання**

Відтворити зображення із лабораторної роботи №2 зображене на рисунку 1.



**Опис використаних графічних засобів бібліотеки OpenGL**

Для малювання графічних фігур ми застосовували glTranslatef для переміщення центру з трьома параметрами(x, y, z), glClearColor(1, 1, 1, 1) для білого кольору фону, glColor3f(01 1, 1) для зміни кольору поверхні, фігури яку ми будуємо, gluCylinder для побудови циліндру, з параметрами радіусів (верхнього, нижнього), висоти, glBegin(GL\_QUADS) оператори дужок та спосіб з'єднання, glVertex3f(0, 0, 0) для завдання вершини з параметрами координат. SwapBuffers(GetDC(Panel1->Handle)) для зміни буферу із зображенням, та інші для налаштування джерела світла та налаштування компонентів.

**Текст програми**

**Unit.h**

//---------------------------------------------------------------------------

#ifndef Unit1H

#define Unit1H

//---------------------------------------------------------------------------

#include <System.Classes.hpp>

#include <Vcl.Controls.hpp>

#include <Vcl.StdCtrls.hpp>

#include <Vcl.Forms.hpp>

#include <Vcl.ComCtrls.hpp>

#include <Vcl.ExtCtrls.hpp>

#include <gl/gl.h>

#include <gl/glu.h>

//---------------------------------------------------------------------------

class TForm1 : public TForm

{

\_\_published: // IDE-managed Components

TPanel \*Panel1;

TEdit \*Edit1;

TUpDown \*UpDown1;

TButton \*Button1;

TEdit \*Edit2;

TUpDown \*UpDown2;

TCheckBox \*CheckBox4;

TCheckBox \*CheckBox5;

TCheckBox \*CheckBox6;

TEdit \*Edit3;

TEdit \*Edit4;

TStaticText \*textX;

TStaticText \*StaticText1;

TStaticText \*StaticText2;

TUpDown \*UpDown3;

TUpDown \*UpDown4;

TEdit \*Edit5;

TUpDown \*UpDown5;

TEdit \*Edit6;

TEdit \*Edit7;

TStaticText \*StaticText3;

TStaticText \*StaticText4;

TStaticText \*StaticText5;

TUpDown \*UpDown6;

TUpDown \*UpDown7;

TStaticText \*StaticText6;

TStaticText \*StaticText7;

TStaticText \*StaticText8;

void \_\_fastcall FormResize(TObject \*Sender);

void \_\_fastcall FormCreate(TObject \*Sender);

void \_\_fastcall Button1Click(TObject \*Sender);

void \_\_fastcall Edit1Change(TObject \*Sender);

void \_\_fastcall Edit2Change(TObject \*Sender);

private: // User declarations

void PaintIt();

public: // User declarations

\_\_fastcall TForm1(TComponent\* Owner);

};

//---------------------------------------------------------------------------

extern PACKAGE TForm1 \*Form1;

//---------------------------------------------------------------------------

#endif

**Unit.cpp**

// ---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

// ---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

// ---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner) : TForm(Owner) {

}

// ---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormResize(TObject \*Sender) {

GLfloat nRange = 300.0; // размер области в мировых координатах

// область вывода для окна в физических координатах

glViewport(0, 0, Panel1->Width, Panel1->Height);

// настройка матрицы проецирования

glMatrixMode(GL\_PROJECTION);

// сброс матрицы

glLoadIdentity();

// установка ортогональной проекции по размерам окна

if (Panel1->Width <= Panel1->Height)

glOrtho(-nRange, nRange, -nRange\*Panel1->Height / Panel1->Width,

nRange\*Panel1->Height / Panel1->Width, -nRange \* 2, nRange \* 2);

else

glOrtho(-nRange\*Panel1->Width / Panel1->Height,

nRange\*Panel1->Width / Panel1->Height, -nRange, nRange, -2 \* nRange,

nRange \* 2);

}

// ---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormCreate(TObject \*Sender) {

// отключение ошибок с вещественными числами

\_control87(MCW\_EM, MCW\_EM);

// получение контекста окна компонента Panel1

HDC hdc = GetDC(Panel1->Handle);

// установка формата пикселей

PIXELFORMATDESCRIPTOR pfd = {

sizeof(PIXELFORMATDESCRIPTOR), 1,

PFD\_DRAW\_TO\_WINDOW | PFD\_SUPPORT\_OPENGL | PFD\_DOUBLEBUFFER,

PFD\_TYPE\_RGBA, 24, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 32, 0, 0,

PFD\_MAIN\_PLANE, 0, 0, 0, 0};

int PixelFormat = ChoosePixelFormat(hdc, &pfd);

SetPixelFormat(hdc, PixelFormat, &pfd);

// создание контектса отображения на компоненте Panel1

HGLRC hrc = wglCreateContext(hdc);

wglMakeCurrent(hdc, hrc);

}

// ---------------------------------------------------------------------------

void TForm1::PaintIt() {

// настройка используемых возможностей OpenGL

glEnable(GL\_COLOR\_MATERIAL); // установка вывода материала объектов

glEnable(GL\_DEPTH\_TEST); // включить расчеты буфера глубины

glEnable(GL\_LIGHTING); // использовать освещение

glEnable(GL\_LIGHT1); // включить источник света 1

glShadeModel(GL\_SMOOTH); // разрешить плавное цветовое сглаживание

// инициализация матрицы модели

glMatrixMode(GL\_MODELVIEW);

glLoadIdentity();

// Настройка источника света

GLfloat light1\_diffuse[] = {4, 7, 2};

GLfloat light1\_position[] = {100.0, 100.0, 100.0, 1.0};

glLightfv(GL\_LIGHT1, GL\_DIFFUSE, light1\_diffuse);

glLightfv(GL\_LIGHT1, GL\_POSITION, light1\_position);

// Установка параметров преобразования объектов

// поворот объектов на угол заданный в Edit1, вокруг вектора (1,1,1)

glRotatef(Edit1->Text.ToDouble(), CheckBox4->Checked, CheckBox5->Checked, CheckBox6->Checked);

// установка масштаба по значению в Edit2 по всем осям

glScalef(Edit2->Text.ToDouble()/2, Edit3->Text.ToDouble()/2,

Edit4->Text.ToDouble()/2);

//перемещение по всем осям

glTranslatef(Edit5->Text.ToDouble(), Edit6->Text.ToDouble(),

Edit7->Text.ToDouble());

// перемещение по оси Х

glTranslatef(-100, 0, 0);

// Создание объектов изображения

// очитска белым цветом

glClearColor(1, 1, 1, 1);

glClear(GL\_DEPTH\_BUFFER\_BIT | GL\_COLOR\_BUFFER\_BIT);

GLUquadricObj \*quadObj = gluNewQuadric();

// создание сферы красного цвета

// glColor3f(1, 0, 0);

// gluSphere(quadObj, 50, 50, 10);

// перемещение начала координат

glTranslatef(-100, 0, 0);

// создание конуса

glColor3f(0, 1, 0);

gluCylinder(quadObj, 50, 50, 100, 100, 100);

glTranslatef(200, 0, 0);

gluCylinder(quadObj, 50, 50, 100, 100, 100);

glTranslatef(200, 0, 0);

gluCylinder(quadObj, 50, 50, 100, 100, 100);

glTranslatef(200, 0, 0);

gluCylinder(quadObj, 50, 50, 100, 100, 100);

glTranslatef(-700, -50, -100);

// создание параллелепипеда по граням

glBegin(GL\_QUADS);

glColor3f(1, 0, 0);

// по четыре вершины на грань

glVertex3f(0, 0, 0);

glVertex3f(0, 100, 0);

glVertex3f(800, 100, 0);

glVertex3f(800, 0, 0);

glColor3f(1, 0, 0);

glVertex3f(800, 0, 100);

glVertex3f(800, 100, 100);

glVertex3f(0, 100, 100);

glVertex3f(0, 0, 100);

glColor3f(1, 0, 1);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 100);

glVertex3f(0, 100, 100);

glVertex3f(0, 100, 0);

glVertex3f(800, 0, 0);

glVertex3f(800, 0, 100);

glVertex3f(800, 100, 100);

glVertex3f(800, 100, 0);

glColor3f(1, 0, 0);

glVertex3f(0, 0, 0);

glVertex3f(0, 0, 50);

glVertex3f(800, 0, 100);

glVertex3f(800, 0, 0);

glVertex3f(0, 100, 0);

glVertex3f(0, 100, 100);

glVertex3f(800, 100, 100);

glVertex3f(800, 100, 0);

glEnd();

// -------------------

glTranslatef(0, 0, 200);

glBegin(GL\_QUADS);

glColor3f(0, 0, 1);

glVertex3f(0, 0, 0);

glVertex3f(800, 0, 0);

glVertex3f(800, 100, 0);

glVertex3f(0, 100, 0);

glColor3f(0, 0, 1);

glVertex3f(0, 0, 0);

glVertex3f(400, 0, 100);

glVertex3f(400, 100, 100);

glVertex3f(0, 100, 0);

glColor3f(0, 0, 1);

glVertex3f(800, 0, 0);

glVertex3f(800, 100, 0);

glVertex3f(400, 100, 100);

glVertex3f(400, 0, 100);

glEnd();

// ---------------------

glBegin(GL\_TRIANGLES);

glColor3f(0, 0, 1);

glVertex3f(0, 0, 0);

glVertex3f(400, 0, 100);

glVertex3f(800, 0, 0);

glColor3f(0, 0, 1);

glVertex3f(0, 100, 0);

glVertex3f(800, 100, 0);

glVertex3f(400, 100, 100);

glEnd();

// смена буфера с изображением

SwapBuffers(GetDC(Panel1->Handle));

}

// ---------------------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender) {

PaintIt();

}

// ---------------------------------------------------------------------------

void \_\_fastcall TForm1::Edit1Change(TObject \*Sender) {

PaintIt();

}

// ---------------------------------------------------------------------------

void \_\_fastcall TForm1::Edit2Change(TObject \*Sender)

{

PaintIt();

}

//---------------------------------------------------------------------------

**Демонстрація роботи**

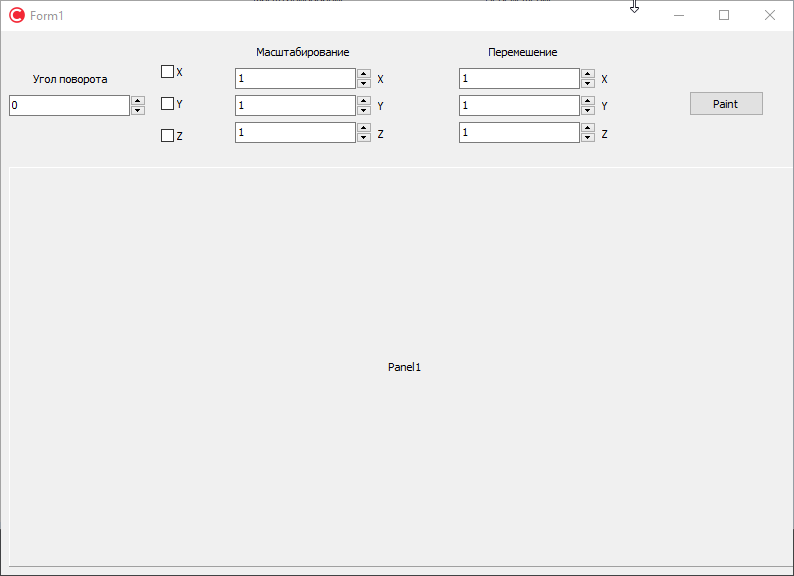


Рис. 2 - запуск програми

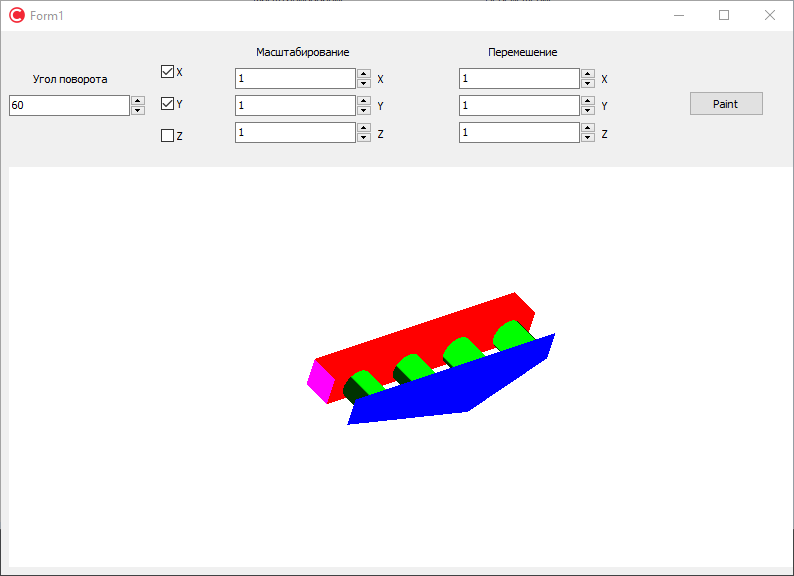
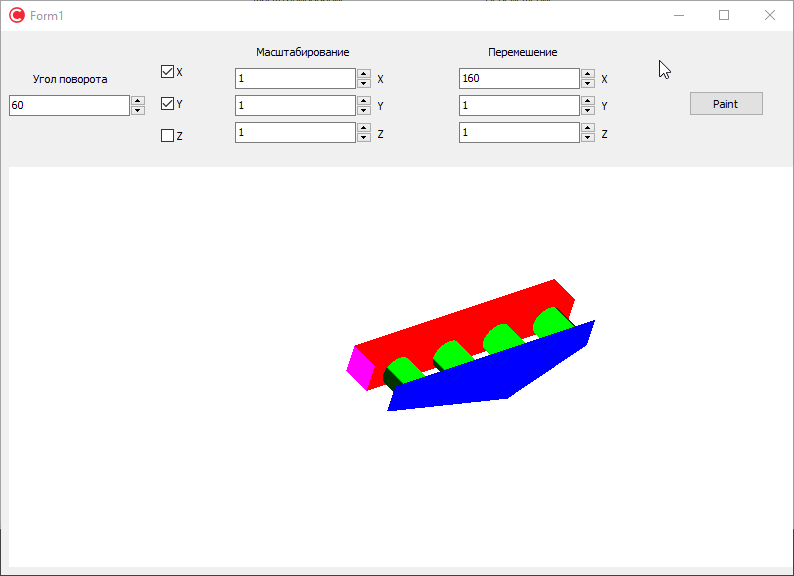


Рис. 3 – обертання по осям



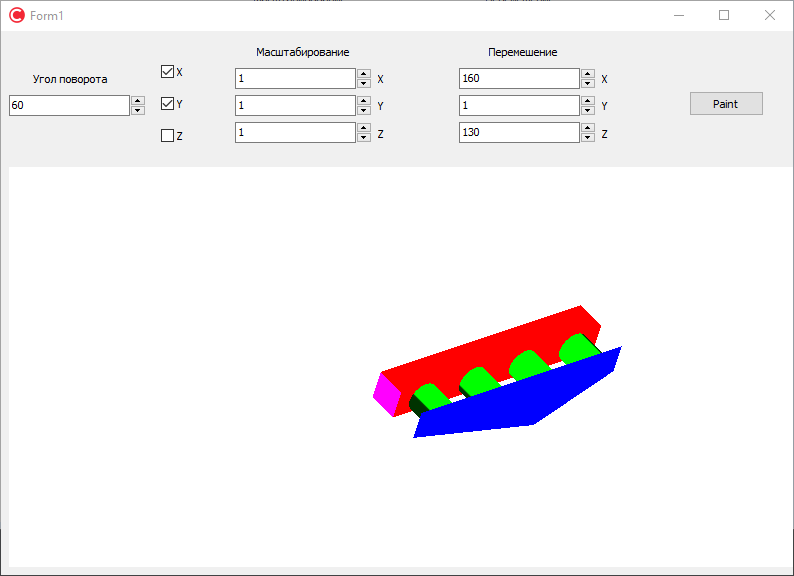
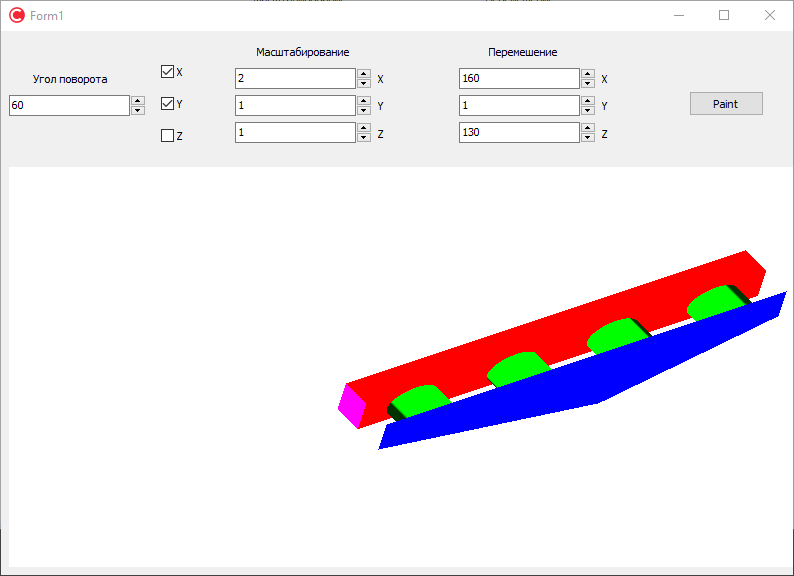
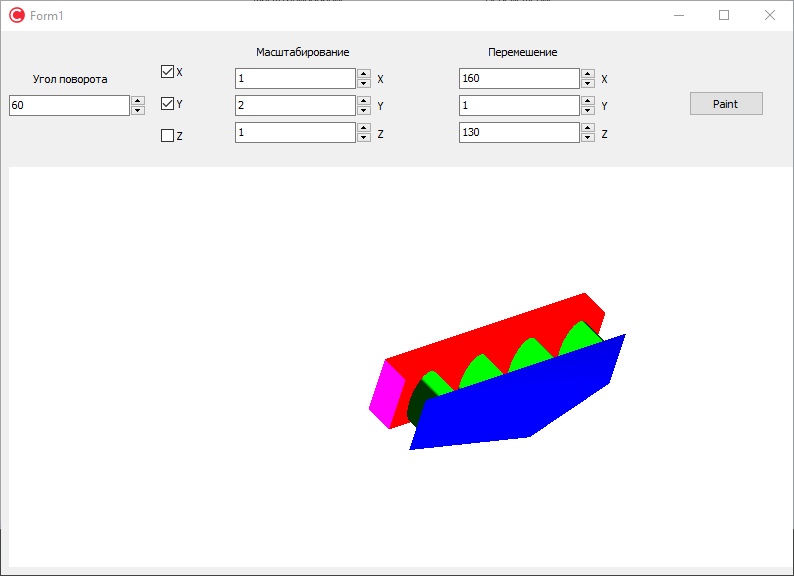


Рис. 4-5 - переміщення по осям





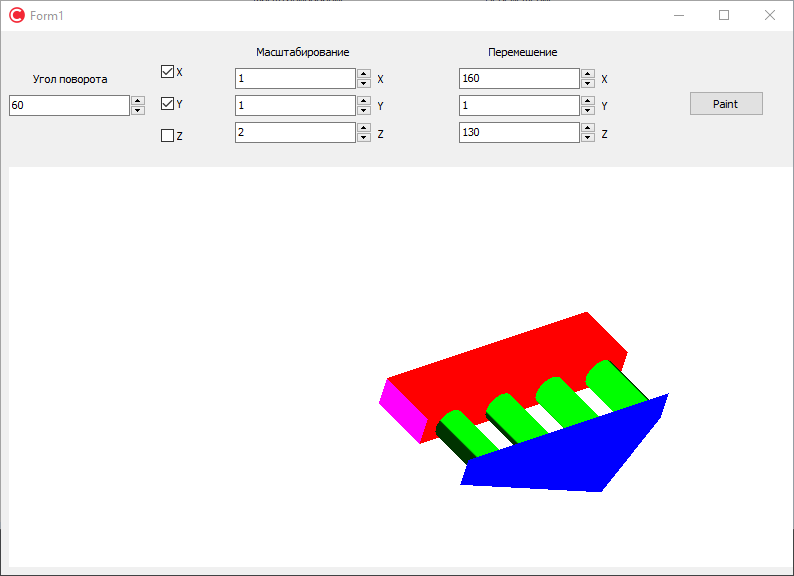


Рис. 5-6-7 – масштабування по осям

**Висновок**: В ході лабораторної роботи ми перетворювали наш 2D малюнок у 3D, для реалізації ми ознайомились з бібліотекою OpenGL, щоб виконати поставлену задачу. Данна бібліотека значно полегшує роботу завдяки своїм вбудованим функціями і тд.