**Министерство образования и науки Российской Федерации**

федеральное государственное автономное образовательное учреждение

высшего образования

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**ТОМСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Инженерная школа информационных технологий и робототехники

Отделение Информационных технологий

Профиль Геоинформационные системы

**ОТЧЕТ  
по лабораторной работе №6**

Применение визуальных эффектов в OpenGL

по дисциплине Компьютерная графика

Выполнил студент группы 8И5Б \_\_\_\_\_\_\_\_\_\_ Смирнов П.О.

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Отчет принят:

Принял: \_\_\_\_\_\_\_\_\_\_\_ Шумихин В.С.

\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2018 г.

Томск 2018 г.

# Цель

Изучить основы работы с библиотекой OpenGL.

# Задание варианта

# Создать трехмерную сцену, используя примитивы библиотеки GLU.

# Нанести текстуру на один из объектов.

# Использовать эффекты освещения, тумана, прозрачности.

# Организовать вращение по таймеру или перемещение (перенос, поворот, масштабирование) объектов по нажатию горячих клавиш.

# Ход работы

Для выполнения данной лабораторной работы необходимо установить и подключить библиотеки TAO. После установки Tao.Framework решается задача инициализации OpenGL в C#.

После создания проекта, к нему подключаются библиотеки Tao.OpenGL.dll, Tao.FreeGlut.dll, Tao.Platform.Windows.dll, Tao.Devil.dll.

# Результаты

На рисунке 1 приведен пример работы программы с тектурированным объектом, над которым выполнены геометрические преобразования.

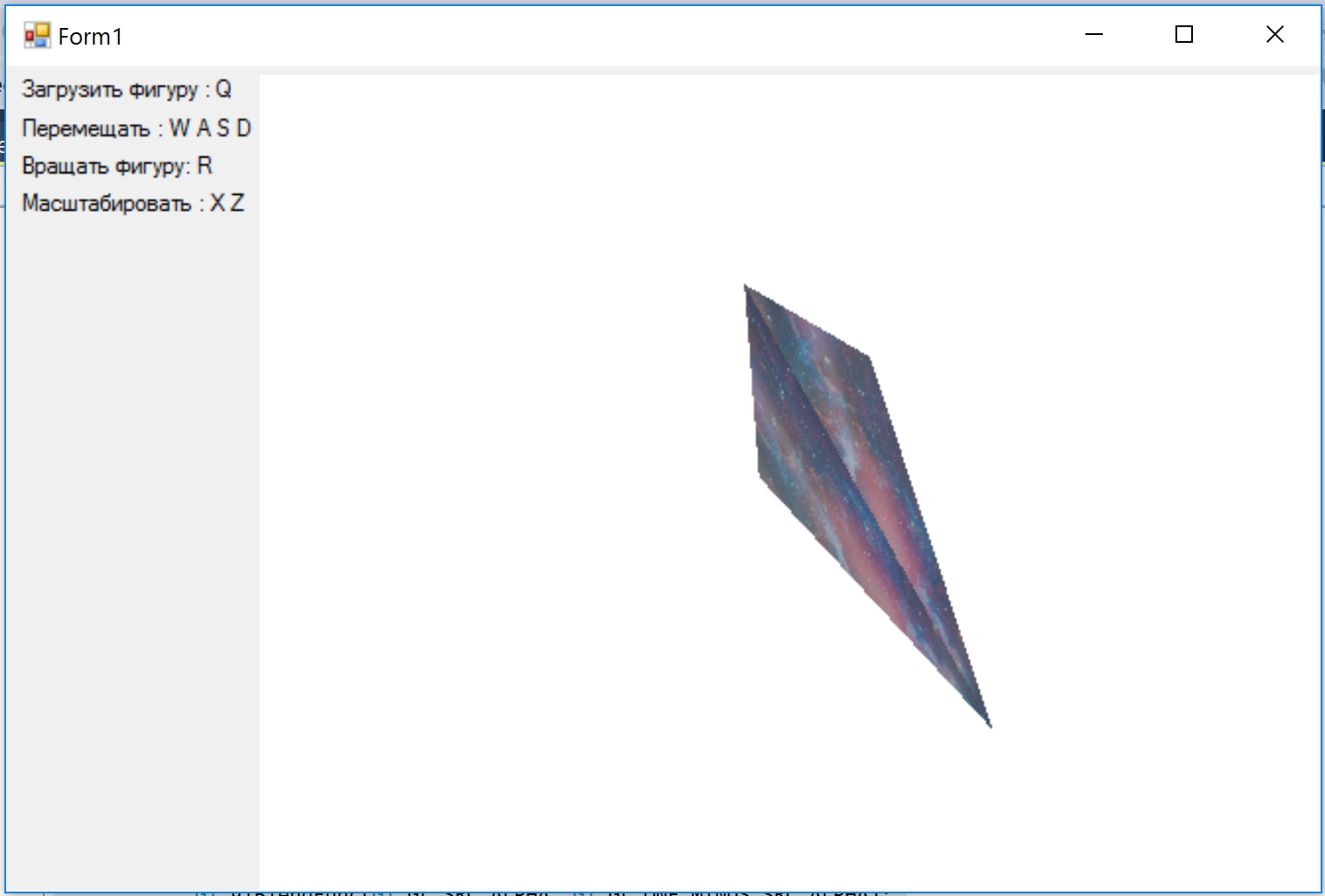


Рисунок 1 – Текстурированная фигура после геометрических преобразований

**Вывод**

В ходе проделанной лабораторной работы были изучены основные возможности библиотеки OpenGL при создании трехмерных сцен.

В программе реализовано отображение трехмерной сцены – плоскости, вращающейся по таймеру. Пользователь имеет возможность выполнять масштабирование и перенос по осям X и Y, изменять скорость вращения фигуры. Для придания сцене реализма добавлены источник освещения, туман и прозрачность материала фигуры.

**Код программы**

namespace Gr6

{

public partial class Form1 : Form

{

double positionX = 0, positionY = 0, angle = 0, scale = 1;

public int imageId;

public uint mGlTextureObject;

public Form1()

{

InitializeComponent();

this.KeyPreview = true;

glControl.InitializeContexts();

mGlTextureObject = 0;

imageId = 0;

positionX = 0;

positionY = 0;

Gl.glClearColor(255, 255, 255, 1);

DrawMethod();

}

private void Form1\_Load(object sender, EventArgs e)

{

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH); // инициализация режима экрана

Il.ilInit(); // инициализация библиотеки openIL

Il.ilEnable(Il.IL\_ORIGIN\_SET);

Gl.glClearColor(255, 255, 255, 1); // установка цвета очистки экрана (RGBA)

Gl.glViewport(0, 0, glControl.Width, glControl.Height); // установка порта вывода

Gl.glMatrixMode(Gl.GL\_PROJECTION); // активация проекционной матрицы

Gl.glLoadIdentity(); // очистка матрицы

Glu.gluPerspective(45, (float)glControl.Width / (float)glControl.Height, 0.1, 200);

Gl.glMatrixMode(Gl.GL\_MODELVIEW); // установка объектно-видовой матрицы

Gl.glLoadIdentity();

Gl.glEnable(Gl.GL\_DEPTH\_TEST); // начальные настройки OpenGL

Lightning();

//FogEffect();

}

public void DrawMethod()

{

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

Gl.glLoadIdentity();

Gl.glClearColor(255, 255, 255, 1);

// включаем режим текстурирования

Gl.glEnable(Gl.GL\_TEXTURE\_2D);

// включаем режим текстурирования, указывая идентификатор mGlTextureObject

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, mGlTextureObject);

Gl.glPushMatrix();

Gl.glTranslated(positionX, positionY, -7);

Gl.glScaled(scale, scale, scale);

Gl.glRotated(angle, 1, 0, 1);

InitOurFigure();

Gl.glPopMatrix();

// отключаем режим текстурирования

Gl.glDisable(Gl.GL\_TEXTURE\_2D);

glControl.Invalidate();

}

private void TextureMethod()

{

// создаем изображение с идентификатором imageId

Il.ilGenImages(1, out imageId);

// делаем изображение текущим

Il.ilBindImage(imageId);

// адрес изображения

string url = @"C:\Users\Pavel\Documents\Visual Studio 2015\Projects\Gr6\Gr6\bin\Debug\space.jpg";

if (Il.ilLoadImage(url))

{

// если загрузка прошла успешно

// сохраняем размеры изображения

int width = Il.ilGetInteger(Il.IL\_IMAGE\_WIDTH);

int height = Il.ilGetInteger(Il.IL\_IMAGE\_HEIGHT);

// определяем число бит на пиксель

int bitspp = Il.ilGetInteger(Il.IL\_IMAGE\_BITS\_PER\_PIXEL);

switch (bitspp) // в зависимости от полученного результата

{

// создаем текстуру, используя режим GL\_RGB или GL\_RGBA

case 24:

mGlTextureObject = MakeGlTexture(Gl.GL\_RGB, Il.ilGetData(), width, height);

break;

case 32:

mGlTextureObject = MakeGlTexture(Gl.GL\_RGBA, Il.ilGetData(), width, height);

break;

}

// очищаем память

Il.ilDeleteImages(1, ref imageId);

DrawMethod();

}

}

private static uint MakeGlTexture(int Format, IntPtr pixels, int w, int h)

{

// идентификатор текстурного объекта

uint texObject;

// генерируем текстурный объект

Gl.glGenTextures(1, out texObject);

// устанавливаем режим упаковки пикселей

Gl.glPixelStorei(Gl.GL\_UNPACK\_ALIGNMENT, 1);

// создаем привязку к только что созданной текстуре

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, texObject);

// устанавливаем режим фильтрации и повторения текстуры

Gl.glTexParameteri(Gl.GL\_TEXTURE\_2D, Gl.GL\_TEXTURE\_WRAP\_S, Gl.GL\_REPEAT);

Gl.glTexParameteri(Gl.GL\_TEXTURE\_2D, Gl.GL\_TEXTURE\_WRAP\_T, Gl.GL\_REPEAT);

Gl.glTexParameteri(Gl.GL\_TEXTURE\_2D, Gl.GL\_TEXTURE\_MAG\_FILTER, Gl.GL\_LINEAR);

Gl.glTexParameteri(Gl.GL\_TEXTURE\_2D, Gl.GL\_TEXTURE\_MIN\_FILTER, Gl.GL\_LINEAR);

Gl.glTexEnvf(Gl.GL\_TEXTURE\_ENV, Gl.GL\_TEXTURE\_ENV\_MODE, Gl.GL\_REPLACE);

// создаем RGB или RGBA текстуру

switch (Format)

{

case Gl.GL\_RGB:

Gl.glTexImage2D(Gl.GL\_TEXTURE\_2D, 0, Gl.GL\_RGB, w, h, 0, Gl.GL\_RGB, Gl.GL\_UNSIGNED\_BYTE, pixels);

break;

case Gl.GL\_RGBA:

Gl.glTexImage2D(Gl.GL\_TEXTURE\_2D, 0, Gl.GL\_RGBA, w, h, 0, Gl.GL\_RGBA, Gl.GL\_UNSIGNED\_BYTE, pixels);

break;

}

// возвращаем идентификатор текстурного объекта

return texObject;

}

public void Lightning()

{

Gl.glDisable(Gl.GL\_ALPHA\_TEST);

Gl.glDisable(Gl.GL\_BLEND);

Gl.glDisable(Gl.GL\_FOG);

Gl.glEnable(Gl.GL\_LIGHTING);

Gl.glEnable(Gl.GL\_LIGHT0);

var ambient = new float[] { 1.0f, 1.0f, 1.0f, 1.0f };

Gl.glEnable(Gl.GL\_LIGHTING);

Gl.glLightModelfv(Gl.GL\_LIGHT\_MODEL\_AMBIENT, ambient);

Gl.glEnable(Gl.GL\_COLOR\_MATERIAL);

Gl.glColorMaterial(Gl.GL\_FRONT, Gl.GL\_AMBIENT);

}

public void FogEffect()

{

Gl.glDisable(Gl.GL\_ALPHA\_TEST);

Gl.glDisable(Gl.GL\_BLEND);

Gl.glEnable(Gl.GL\_ALPHA\_TEST);

Gl.glEnable(Gl.GL\_FOG);

float[] fog\_color = { 0.5f, 0.5f, 0.5f, 1f };

Gl.glFogfv(Gl.GL\_FOG\_COLOR, fog\_color);

Gl.glFogi(Gl.GL\_FOG\_MODE, Gl.GL\_LINEAR);

Gl.glFogf(Gl.GL\_FOG\_START, 3.0f);

Gl.glFogf(Gl.GL\_FOG\_END, 10.0f);

}

public void TransparencyEffect()

{

Gl.glDisable(Gl.GL\_FOG);

Gl.glDisable(Gl.GL\_ALPHA\_TEST);

Gl.glDisable(Gl.GL\_BLEND);

Gl.glEnable(Gl.GL\_ALPHA\_TEST);

Gl.glEnable(Gl.GL\_BLEND);

Gl.glBlendFunc(Gl.GL\_SRC\_ALPHA, Gl.GL\_ONE\_MINUS\_SRC\_ALPHA);

//Gl.glBlendFunc(Gl.GL\_SRC\_ALPHA, Gl.GL\_ONE\_MINUS\_SRC\_ALPHA);

Gl.glColor4d(0, 0, 0, 0.3);

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyData == Keys.Q)

{

TransparencyEffect();

}

if (e.KeyData == Keys.E)

{

FogEffect();

}

if (e.KeyData == Keys.V)

{

Lightning();

}

if (e.KeyData == Keys.D)

{

positionX += 1;

}

if (e.KeyData == Keys.A)

{

positionX -= 1;

}

if (e.KeyData == Keys.W)

{

positionY += 1;

}

if (e.KeyData == Keys.S)

{

positionY -= 1;

}

if (e.KeyData == Keys.R)

{

angle += 2;

}

if (e.KeyData == Keys.Z)

{

scale += 0.1;

}

if (e.KeyData == Keys.X)

{

scale -= 0.1;

}

DrawMethod();

TextureMethod();

}

public void InitOurFigure()

{

Gl.glBegin(Gl.GL\_TRIANGLES);

float x = -1, y = -2, z = 2;

float len = 1.5f;

float[] a = { x, y, z };

float[] b = { x, y + len, z };

float[] c = { x + len, y + len, z };

float[] d = { 0, 0, -1 };

Gl.glNormal3f(0.0f, 0.0f, 1.0f);

Gl.glVertex3d(a[0], a[1], a[2]);

Gl.glTexCoord2f(0.0f, 0.0f);

Gl.glVertex3d(b[0], b[1], b[2]);

Gl.glTexCoord2f(1.0f, 0.0f);

Gl.glVertex3d(c[0], c[1], c[2]);

Gl.glTexCoord2f(0.0f, 1.0f);

Gl.glColor4d(0.0, 0.0, 0.0, 0.1);

Gl.glNormal3f(0.0f, 0.0f, 1.0f);

Gl.glVertex3d(a[0], a[1], a[2]);

Gl.glTexCoord2f(0.0f, 0.0f);

Gl.glVertex3d(b[0], b[1], b[2]);

Gl.glTexCoord2f(1.0f, 0.0f);

Gl.glVertex3d(d[0], d[1], d[2]);

Gl.glTexCoord2f(0.0f, 1.0f);

Gl.glNormal3f(0.0f, 0.0f, 1.0f);

Gl.glVertex3d(c[0], c[1], c[2]);

Gl.glTexCoord2f(0.0f, 0.0f);

Gl.glVertex3d(b[0], b[1], b[2]);

Gl.glTexCoord2f(1.0f, 0.0f);

Gl.glVertex3d(d[0], d[1], d[2]);

Gl.glTexCoord2f(0.0f, 1.0f);

Gl.glNormal3f(0.0f, 0.0f, 1.0f);

Gl.glVertex3d(c[0], c[1], c[2]);

Gl.glTexCoord2f(0.0f, 0.0f);

Gl.glVertex3d(a[0], a[1], a[2]);

Gl.glTexCoord2f(1.0f, 0.0f);

Gl.glVertex3d(d[0], d[1], d[2]);

Gl.glTexCoord2f(0.0f, 1.0f);

Gl.glEnd();

}

}

}