**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 №5**

Основы работы с библиотекой 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

* Инициализировать библиотеку OpenGL.
* С помощью двумерных примитивов нарисовать осмысленную двумерную сцену.
* Использовать текстуры для полигональных объектов.
* Организовать 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.

Для отображения на форме двумерной сцены используется компонент библиотеки Tao.Platform.Windows.dll – компонент SimpleOpenGLControl.

Для работы с функциями библиотеки OpenGL используется класс Gl, находящийся в пространстве имен Tao.OpenGL.

Для работы с функциями библиотеки Glut используется класс Glut.

В обработчике загрузки формы сначала проходит инициализация библиотеки Glut. Далее функция glutInitDisplayMode вызывается для настройки RGB режима визуализации, режима двойной буферизации окна. Для последующего текстурирования объекта необходимо инициализировать библиотеку openIL.

Установка цвета очистки окна осуществляется с помощью функции glClearColor.

Порт представляет собой камеру, ограничивающую область видимости пространства, в которой находится наблюдаемый объект. С помощью glViewport устанавливается необходимое значение порта вывода, т.е. задается область компонента SimpleOpenGLControl, визуализирующая сцену.

С помощью функции glMatrixMode происходит настройка проекции, а именно задается матричный режим, т.е. определяется матрица над которой будут производится дальнейшие операции. GL\_PROJECTION – матрица проекций.

С помощью функции glLoadIdentity осуществляется очищение матрицы (т.е. замена матрицы единичной матрицей).

Далее устанавливается тип текущей проекции с помощью функции Glu.gluOrtho2D.

**Результат**

На рисунке 1 показан результат работы программы после загрузки фигуры.

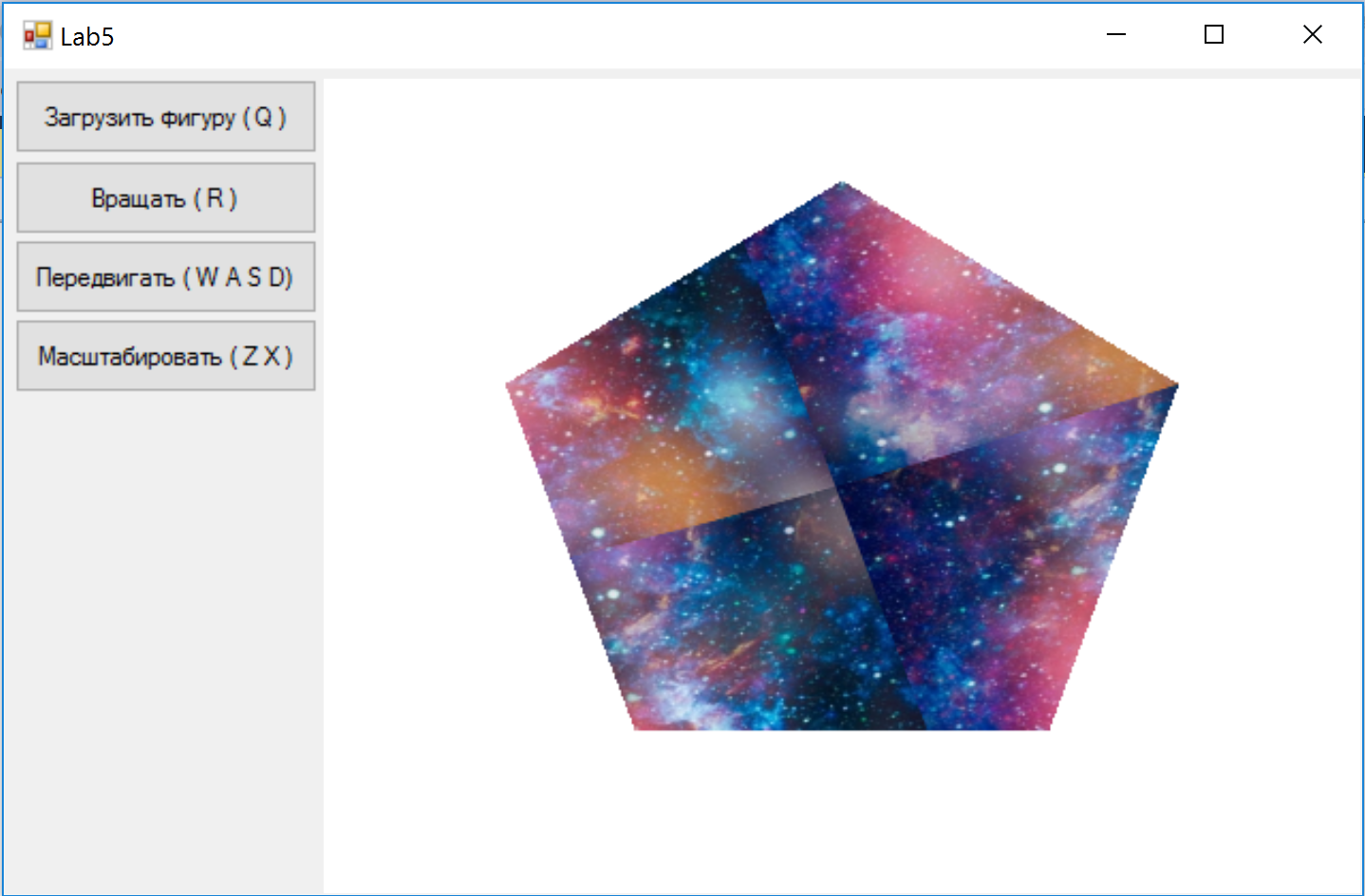


Рисунок 1 – Работа программы

На рисунке 2 показан результат работы программы после проделанных некоторых геометрических преобразований на фигурой.

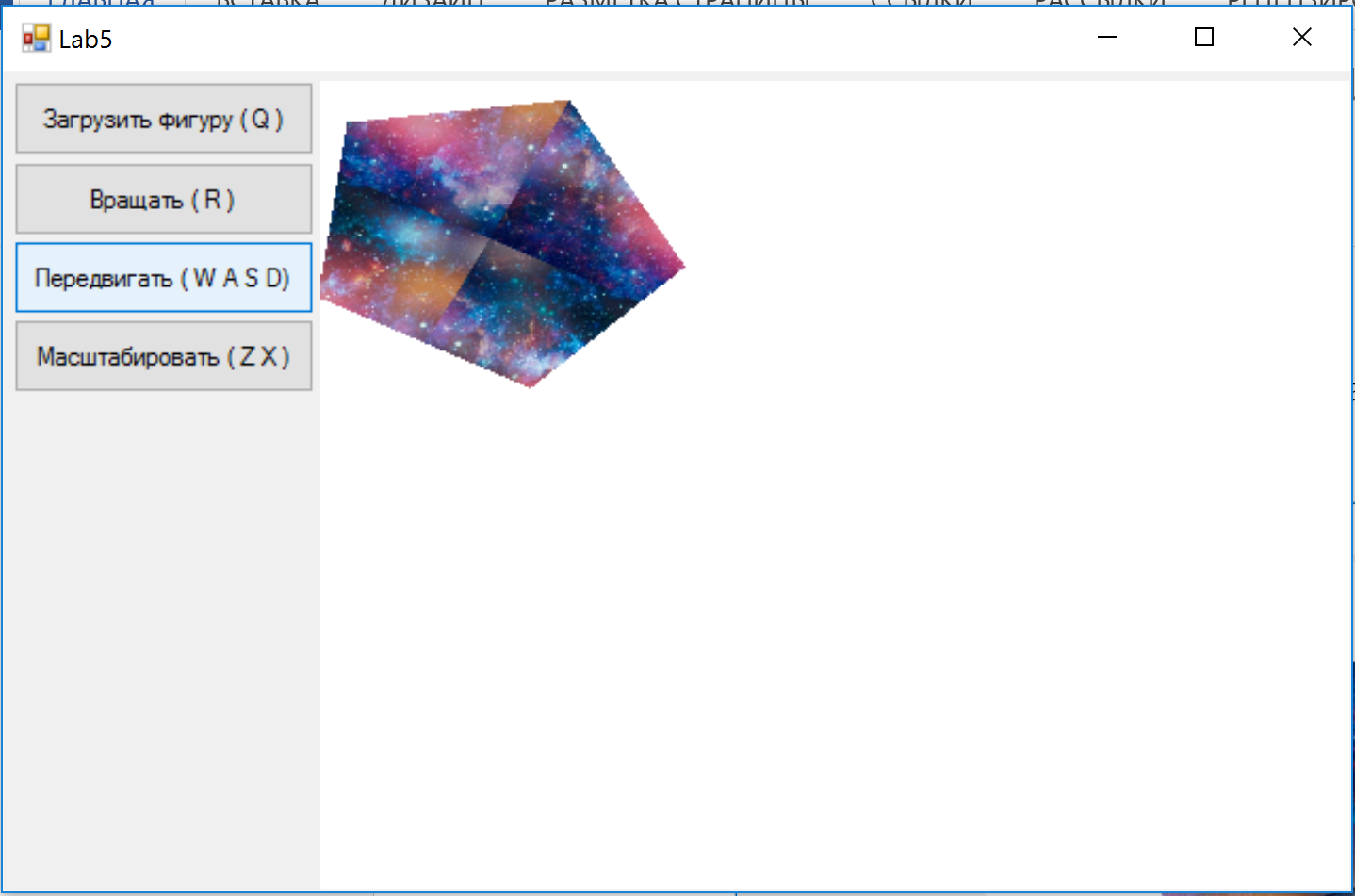


Рисунок 2 – Работа программы после выполнения преобразований

**Вывод**

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

Библиотека значительно упрощает процесс выполнения преобразований (масштабирование, поворот и смещение). Для выполнения этих преобразований достаточно указать всего одну команду.

Процесс текстурирования объектов в OpeGL не может быть выполнен одной командой, поэтому появляется необходимость реализовывать методы загрузки изображения и преобразования его в текстуру.

**Код программы**

public partial class Form1 : Form

{

double positionX, positionY;

public int imageId;

public uint mGlTextureObject;

public Form1()

{

InitializeComponent();

this.KeyPreview = true;

positionX = 0;

positionY = 0;

imageId = 0;

mGlTextureObject = 0;

glControl.InitializeContexts();

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.gluOrtho2D(0, 0, glControl.Width, glControl.Height);

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

Gl.glLoadIdentity();

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

Gl.glEnable(Gl.GL\_LIGHTING);

Gl.glEnable(Gl.GL\_LIGHT0);

}

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\Gr5\Gr5\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;

}

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);

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;

}

return texObject;

}

public void DrawMethod()

{

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

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

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

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

Gl.glBegin(Gl.GL\_POLYGON);

Gl.glVertex2d(0, 0.75);

Gl.glTexCoord2f(0, (float)0.75);

Gl.glVertex2d(0.65, 0.25);

Gl.glTexCoord2f((float)0.65, (float)0.25);

Gl.glVertex2d(0.4, -0.6);

Gl.glTexCoord2f((float)0.4, (float)-0.6);

Gl.glVertex2d(-0.4, -0.6);

Gl.glTexCoord2f((float)-0.4, (float)-0.6);

Gl.glVertex2d(-0.65, 0.25);

Gl.glTexCoord2f((float)-0.65, (float)0.25);

Gl.glEnd();

Gl.glFlush();

Gl.glDisable(Gl.GL\_TEXTURE\_2D); // отключаем режим текстурирования

glControl.Invalidate();

}

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

{

if(e.KeyData == Keys.D)

{

positionX += 0.1;

Gl.glTranslated(positionX, 0, 0);

positionX -= 0.1;

}

if (e.KeyData == Keys.A)

{

positionX += 0.1;

Gl.glTranslated(-positionX, 0, 0);

positionX -= 0.1;

}

if (e.KeyData == Keys.W)

{

positionY += 0.1;

Gl.glTranslated(0, positionY, 0);

positionY -= 0.1;

}

if (e.KeyData == Keys.S)

{

positionY += 0.1;

Gl.glTranslated(0, -positionY, 0);

positionY -= 0.1;

}

if (e.KeyData == Keys.R)

{

Gl.glRotated((double)15, 0, 0, 1);

}

if (e.KeyData == Keys.Z)

{

Gl.glScaled(1.1, 1.1, 1);

}

if (e.KeyData == Keys.X)

{

Gl.glScaled(0.9, 0.9, 1);

}

DrawMethod();

TextureMethod();

}

private void button1\_Click(object sender, EventArgs e)

{

positionX += 0.1;

Gl.glTranslated(positionX, 0, 0);

positionX -= 0.1;

DrawMethod();

}

}