**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский государственный авиационный технический университет"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина: Компьютерная графика**

**Отчет по лабораторной работе №5**

**Тема: «Текстурирование объектов в OpenGL»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМИ-248 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Фаизова А.Н. |  |  |  |
| Принял | Мухтаров А.Р. |  |  |  |

**Уфа 2018**

**Цель:** разработать OpenGL-приложение, которое будет накладывать текстуру на тело вращения, в соответствии с вариантом из работы №3.

**Индивидуальное задание**

Разработайте OpenGL-приложение, которое будет накладывать текстуру на шар.

**Описание программы:**

Текстурирование

Наложение текстуры на поверхность трехмерных объектов позволяет повысить их реалистичность за счет детализации поверхности цветом. OpenGL предоставляет возможность наложения одно-, двух- и трехмерных текстур, а также кубических текстур. Если необходимо отобразить объект со сложной структурой поверхности, то без текстурирования каждый элемент необходимо рисовать вручную. Например, в кирпичной стене каждый кирпичик должен быть нарисован в виде отдельного полигона и без текстурирования такая стена может потребовать тысяч полигонов, при этом кирпичи будут выглядеть одинаковыми и недостаточно реалистичными. Если при отрисовке трехмерных сцен не использовать текстуры, то пришлось бы создавать невероятную детализацию с миллиардами полигонов. Использование текстур позволяет нанести на один большой полигон какое-либо изображение (по сути изображение натягивается на трехмерный объект, в соответствии с указанными текстурными координатами), таким образом, один полигон может представлять сразу целую часть трехмерной сцены – например небо, стены, заборы, и т.д. По сути, текстуры – это массивы данных, которые в соответствии с текстурными координатами будут наложены на трехмерный объект. Тексел (сокращение от texture element) – это индивидуальное значение элемента текстуры.

Предварительно содержание текстуры должно быть передано в библиотеку OpenGL. Для этого передается указатель на массив с информацией о текстуре. Информация о текстуре представляет собой построчное представление изображения в формате:

RGB….RGB

RGB….RGB

RGB….RGB

Где RGB – это красная, зеленая и синяя составляющие цвета каждого пиксела текстуры. Размер текстуры должен быть кратен степени двойки: 32x32, 32x64, 128x256 и т д. Текстура может быть подготовлена различным образом: может быть сформирована программно или, например, загружена из какого-либо файла.

Использование изображения в формате BMP в качестве текстуры

Одним из наиболее близких форматов к формату текстуры OpenGL является формат хранения графических изображений BMP. Отличие в хранении данных о пикселах между форматом BMP и форматом текстуры OpenGL заключается в различном порядке байт, описывающих составляющие цвета пиксела. В формате BMP данные о каждом пикселе хранятся в порядке: BGR. В формате текстуры OpenGL цветовые характеристики пикселов хранятся в противоположном порядке: RGB.

Рассмотрим процесс загрузки текстуры и наложения ее на трехмерный объект. Загрузка файла BMP в память. Формат файла BMP кроме самого изображения содержит так же общую информацию о нем, например, такую как, формат изображения (в файле формата BMP изображение может храниться в нескольких разных вариантах), ширину и высоту изображения и др. Для использования изображения в формате BMP в качестве текстуры можно использовать ниже приведенный метод

public static void LoadTexture(Bitmap bmp) { BitmapData data = bmp.LockBits( new Rectangle(0, 0, bmp.Width, bmp.Height), ImageLockMode.ReadOnly, System.Drawing.Imaging.PixelFormat.Format24bppRgb); GL.TexImage2D(TextureTarget.Texture2D, 0, PixelInternalFormat.Rgb, data.Width, data.Height, 0, OpenTK.Graphics.OpenGL.PixelFormat.Bgr, PixelType.UnsignedByte, data.Scan0); bmp.UnlockBits(data); }

В качестве параметра методу передается объект типа Bitmap, в который должно быть загружено изображение формата BMP, например, с помощью следующего фрагмента программы:

Bitmap bmpTex; bmpTex = new Bitmap("OGL64x64.bmp");

В методе LoadTexture сначала вызывается метод LockBits объекта Bitmap, который блокирует объект в системной памяти. Затем вызывается метод GL.TexImage2D, который передает изображение в библиотеку OpenGL в качестве текстуры. В конце метода LoadTexture вызывается метод UnlockBits объекта Bitmap, который разблокирует объект в памяти. Инициализация текстуры

При формировании изображения используется текущая текстура. Для определения текущей текстуры и её параметров используется команда TexImage2D:

void TexImage2D(TextureTarget target, int level, PixelInternalFormat internalformat, int width, int height, int border, PixelFormat format, PixelType type, IntPtr pixels);

Параметры команды:

• target – определяет тип текстуры, параметр должен быть равен значению TextureTarget.Texture2D.

• level – уровень детализации текстуры, значение должно быть равно 0.

• PixelInternalFormat – определяет количество цветовых компонентов в данных о текстуре.

• width – ширина текстуры в пикселах.

• height – высота текстуры в пикселах.

• border – размер "окантовки", значение должно быть равно 0.

• PixelFormat - формат передаваемых данных о текстуре.

• type – определяет тип данных, используемый для передачи каждой составляющей цвета. В формате BMP для каждой составляющей цвета используется беззнаковое байтовое значение, поэтому в C# для этого используется значение PixelType.UnsignedByte.

• pixels - указатель на блок памяти с подготовленным изображением текстуры в формате OpenGL.

Настройка параметров отображения текстуры

Для настройки параметров отображения текстуры, которые используются при изменении её положения в пространстве и масштабировании, используется команда TexParameter.

void TexParameter(TextureTarget target, TextureParameterName pname, int param); Параметры процедуры:

• target – тип текстуры для которой определяются параметры. На C# параметр должен быть равен значению Texture2D перечисления TextureTarget, в Object Pascal – значению GL\_TEXTURE\_2D.

• pname – изменяемый параметр.

• param – тип функции масштабирования изображения текстуры.

Значение pname

Функция увеличения текстуры при масштабировании TextureParameterName.TextureMagFilter

Функция уменьшения текстуры при масштабировании TextureParameterName.TextureMinFilter

Значение param

В качестве значения берется точка текстуры с ближайшими координатами TextureMinFilter.Nearest

В качестве значения берется среднеарифметическое значение на основе 4 точек рядом с координатами точки

TextureMinFilter.Linear

Для отображения текстуры, кроме загрузки текстуры и её инициализации, так же необходимо включить режим отображения текстуры с помощью команды Enable. Поэтому после загрузки и инициализации текстуры вызывается команда Enable с параметром EnableCap.Texture2D на C# для включения режима отображения текстуры. Для отключения режима отображения текстуры используется команда Disable.

При формировании сложной сцены может потребоваться, чтобы одни объекты отображались с текстурой, а другие – без неё. В этом случае необходимо перед формированием одних объектов включать режим отображения текстуры, а перед формированием других объектов отключать формирование текстуры.

Координаты текстуры

Задание координат текстуры позволяет сопоставить вершины примитивов с изображением текстуры. Вне зависимости от размера изображения текстуры в пикселах, координаты текстуры определяются библиотекой OpenGL в интервале значений [0, 1].

Для правильного изображения текстуры координата текстуры, соответствующая вершине, должна определяться с помощью команды TexCoord перед заданием координат соответствующей вершины примитива.

void TexCoord2(float s, float t); Параметры:

• s – горизонтальная координата текстуры;

• t – вертикальная координата текстуры.

Как текстурируется сложный объект? Координаты текстуры определяются индивидуально для каждого примитива. Текстура как бы разбивается на прямоугольники, и каждый прямоугольник накладывается на отдельный четырехугольный примитив, из которых формируется поверхность.

Исходный код программы

Form.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Windows.Forms;

using OpenTK;

using OpenTK.Graphics;

using OpenTK.Graphics.OpenGL;

using OpenTK.Input;

using GLTools;

namespace WindowsFormsApplication2

{

public partial class Form1 : Form

{

float AngleX = 0;

float AngleY = 0;

float AngleZ = 0;

const float AngleDl = 5;

PolygonMode mode = PolygonMode.Fill;

Bitmap bmpTex;

public Form1()

{

InitializeComponent();

}

private void glControl1\_Resize(object sender, EventArgs e)

{

SetupViewport();

AnT.Invalidate();

}

private void SetupViewport()

{

int w = AnT.Width;

int h = AnT.Height;

GL.MatrixMode(MatrixMode.Projection);

GL.LoadIdentity();

GL.Ortho(-1, 1, -1, 1, -1, 1);

GL.MatrixMode(MatrixMode.Modelview);

GL.Viewport(0, 0, w, h);

}

void Draw(double r, int nx, int ny)

{

int ix, iy;

double x, y, z;

for (iy=0; iy < ny; ++iy)

{

GL.Begin(PrimitiveType.QuadStrip);

for (ix = 0; ix <= nx; ++ix)

{

x = r \* Math.Cos(2 \* ix \* Math.PI / nx)\* Math.Sin(iy \* Math.PI / ny) ;

y = r \* Math.Sin(2 \* ix \* Math.PI / nx)\* Math.Sin(iy \* Math.PI / ny) ;

z = r \* Math.Cos(iy \* Math.PI / ny);

GL.Normal3(x, y, z);//нормаль направлена от центра

GL.TexCoord2((double)ix / (double)nx, (double)iy / (double)ny);// привязка текстуры к вершинам примитива

GL.Vertex3(x, y, z);

x = r \* Math.Cos(2 \* ix \* Math.PI / nx)\* Math.Sin((iy + 1) \* Math.PI / ny) ;

y = r \* Math.Sin(2 \* ix \* Math.PI / nx)\* Math.Sin((iy + 1) \* Math.PI / ny) ;

z = r \* Math.Cos((iy + 1) \* Math.PI / ny);

GL.Normal3(x, y, z);

GL.TexCoord2((double)ix / (double)nx, (double)(iy + 1) / (double)ny);

GL.Vertex3(x, y, z);

}

GL.End();

}

}

private void glControl1\_Paint(object sender, PaintEventArgs e)

{

// очистка буферов цвета и глубины

GL.Clear(ClearBufferMask.ColorBufferBit | ClearBufferMask.DepthBufferBit);

// поворот изображения

GL.LoadIdentity();

GL.PointSize(3.0f);

GL.Rotate(AngleX, 1.0, 0.0, 0.0);

GL.Rotate(AngleY, 0.0, 1.0, 0.0);

GL.Rotate(AngleZ, 0.0, 0.0, 1.0);

// включаем режим текстурирования

GL.Enable(EnableCap.Texture2D);

GL.Color3(1f, 1, 1);

GL.PolygonMode(MaterialFace.FrontAndBack, mode);

Draw(0.7, 20, 20);

// отключаем режим текстурирования

GL.Disable(EnableCap.Texture2D);

GL.Flush();

GL.Finish();

// обновляем элемент AnT

AnT.SwapBuffers();

}

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

{

SetupViewport();

GL.ClearColor(1f, 1f, 1f, 1f); // цвет фона

// Для отображения текстуры, кроме загрузки текстуры и её инициализации,

// так же необходимо включить режим отображения текстуры с помощью команды Enable

GL.Enable(EnableCap.DepthTest);

bmpTex = new Bitmap("map2.bmp");

// формирование изображения

GLTexture.LoadTexture(bmpTex);

// Команда с параметрами

// первый - ОПРЕДЕЛЯЕТ ТИП ТЕКСТУРЫ И ПРИНИМАЕТ ЗНАЧЕНИЕ GL\_TEXTURE\_2D

// второй - ОПРЕДЕЛЯЕТ НАЗВАНИЕ ПАРАМЕТРА ТЕКСТУРЫ, КОТОРОЕ БУДЕТ ИЗМЕНЯТЬСЯ;

// третий - УСТАНАВЛИВАЕТ НОВОЕ ЗНАЧЕНИЕ ВТОРОГО ПАРАМЕТРА

// Стандартно в OpenGL существуют два фильтра: фильтр, работающий при уменьшении текстуры, фильтр, работающий при увеличении текстуры

GL.TexParameter(TextureTarget.Texture2D, TextureParameterName.TextureMinFilter, (int)TextureMinFilter.Linear);

GL.TexParameter(TextureTarget.Texture2D, TextureParameterName.TextureMagFilter, (int)TextureMagFilter.Linear);

// GL\_TEXTURE\_MIN\_FILTER - ОПРЕДЕЛЯЕТ АЛГОРИТМ СЖАТИЯ ТЕКСТУРЫ ДЛЯ ОБЪЕКТОВ, РАЗМЕР КОТОРЫХ

// БЫЛ УМЕНЬШЕН.ПРИ ЗНАЧЕНИИ GL\_NEAREST БУДЕТ

// ИСПОЛЬЗОВАТЬСЯ ОДНА БЛИЖАЙШАЯ ТОЧКА В ТЕКСТУРЕ, А ПРИ ЗНАЧЕНИИ GL\_LINEAR

// ЧЕТЫРЕ БЛИЖАЙШИХ ТОЧЕК ТЕКСТУРЫ. ОСТАЛЬНЫЕ ЧЕТЫРЕ ОПРЕДЕЛЯЮТ УРОВНИ ДЕТАЛИЗАЦИИ.

// ПО УМОЛЧАНИЮ ЗНАЧЕНИЕ ФУНКЦИИ – GL\_LINEAR.

// GL\_TEXTURE\_MAG\_FILTER - Определяет функцию увеличения текстуры для текстурирования объектов,

// размер которых был увеличен или нет. Существует два таких алгоритма. При значении GL\_NEAREST

// будет использоваться одна ближайшая точка, а GL\_LINEAR четыре ближайших элемента текстуры

AngleX = 30;

AngleY = 30;

}

private void glControl1\_KeyDown(object sender, KeyEventArgs e)

{

switch(e.KeyCode)

{

case Keys.S:

AngleX += AngleDl;

break;

case Keys.W:

AngleX -= AngleDl;

break;

case Keys.D:

AngleY += AngleDl;

break;

case Keys.A:

AngleY -= AngleDl;

break;

case Keys.Q:

AngleZ += AngleDl;

break;

case Keys.E:

AngleZ -= AngleDl;

break;

case Keys.F1:

mode = PolygonMode.Fill;

break;

case Keys.F2:

mode = PolygonMode.Line;

break;

case Keys.F3:

mode = PolygonMode.Point;

break;

}

AnT.Invalidate();

}

private void glControl1\_Load(object sender, EventArgs e)

{

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

}

}

Texture.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

using System.Drawing.Imaging;

using OpenTK.Graphics.OpenGL;

namespace GLTools

{

class GLTexture

{

public static void LoadTexture(Bitmap bmp)

{

// метод LockBits объекта Bitmap блокирует объект в системной памяти.

BitmapData data = bmp.LockBits(new Rectangle(0, 0, bmp.Width, bmp.Height),

ImageLockMode.ReadOnly, System.Drawing.Imaging.PixelFormat.Format24bppRgb);

//glTexImage2D (Glenum target, Glint level, Glint interformat,

// Glsizei width, Glsizei height,

// Glint border, Glenum format,

// Glenum type, const Glvoid\* pixels)

// target - Определяет тип создаваемой текстуры и должен быть равен GL\_TEXTURE\_2D.

// level - Задает число уровней детализации текстуры. Для базового уровня – 0, а если уменьшенный в n раз, то n.

// interformat - Задает число цветных компонентов текстуры и может принимать значения:

// border - Задает ширину границы изображения текстуры.Принимаемые значения: 0 или 1

// format - Определяет формат данных пикселя

// type - Задает тип данных пикселей

// pixels - Определяет указатель на данные изображения для текстуры

GL.TexImage2D(TextureTarget.Texture2D, 0, PixelInternalFormat.Rgb, data.Width, data.Height, 0,

OpenTK.Graphics.OpenGL.PixelFormat.Bgr, PixelType.UnsignedByte, data.Scan0);

// метод GL.TexImage2D передает изображение в библиотеку OpenGL в качестве текстуры

bmp.UnlockBits(data);

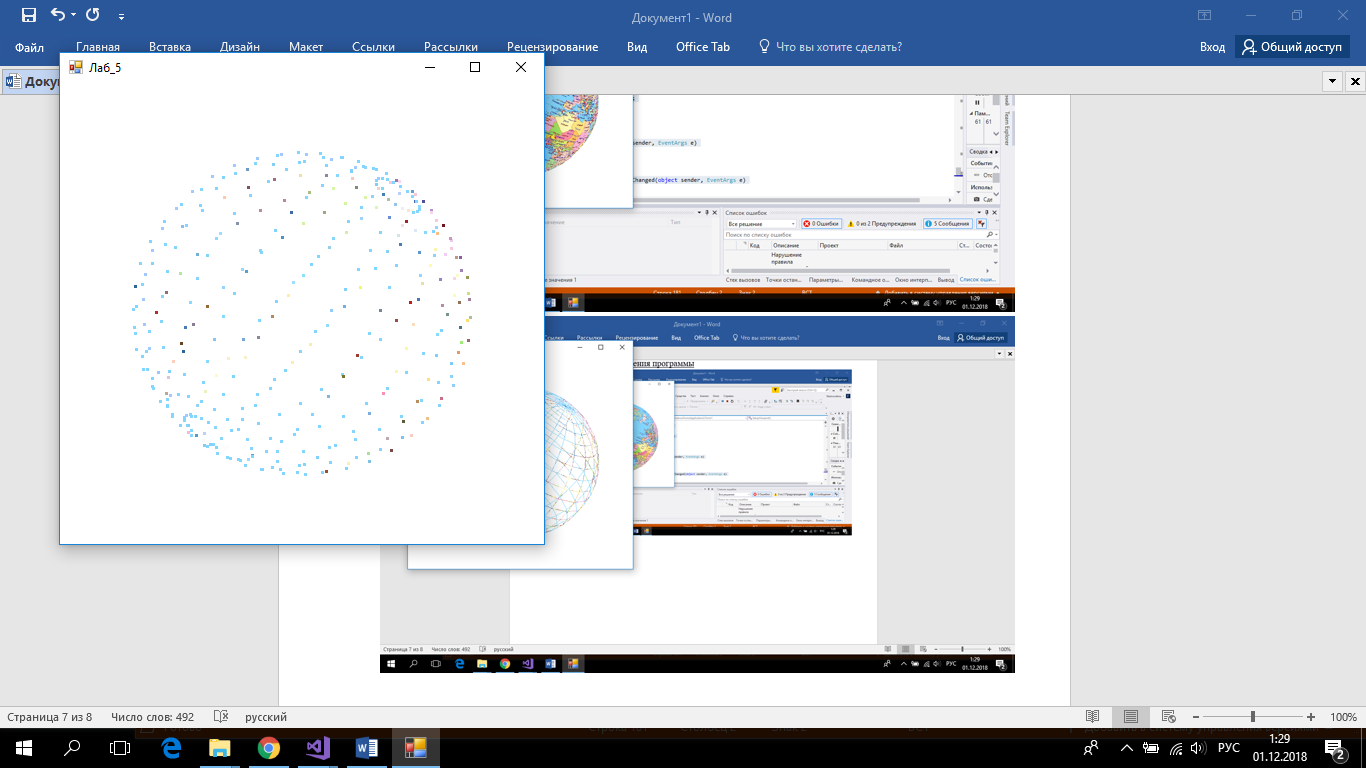
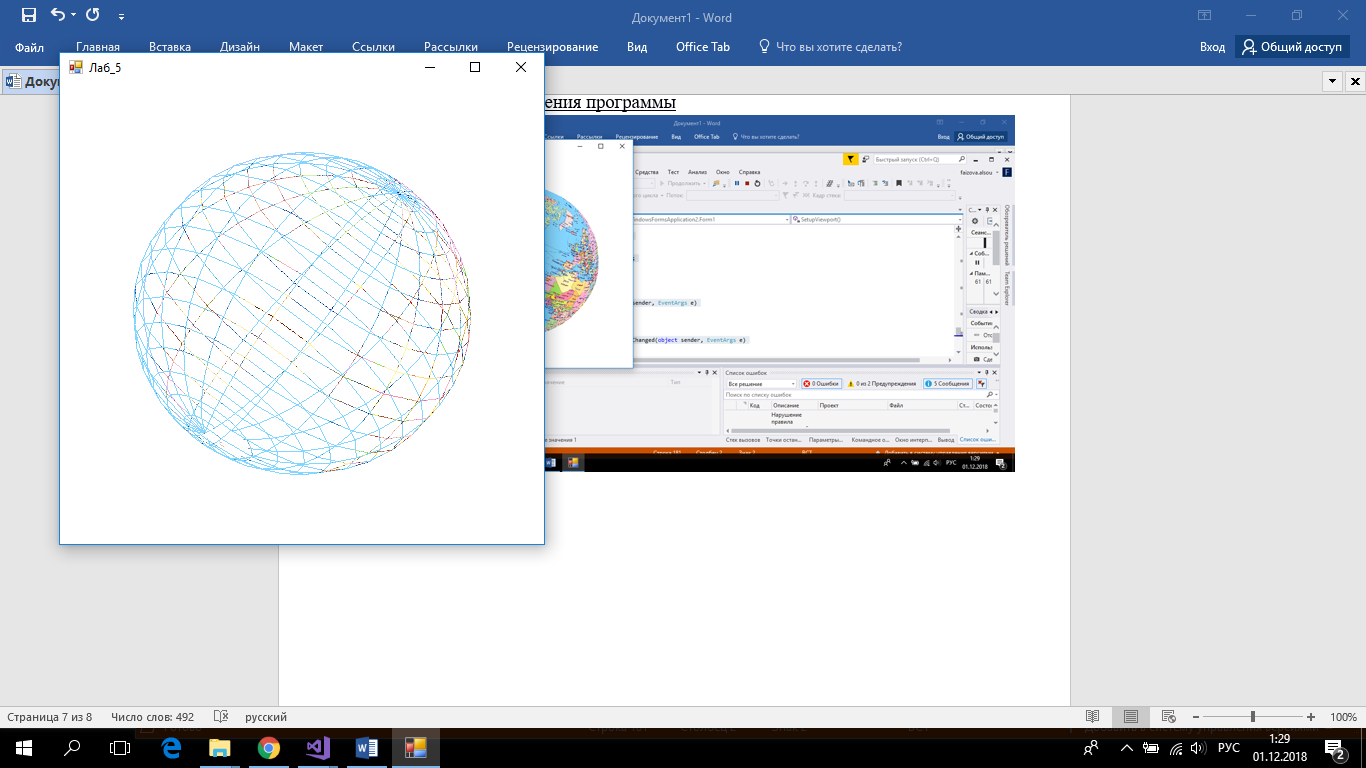
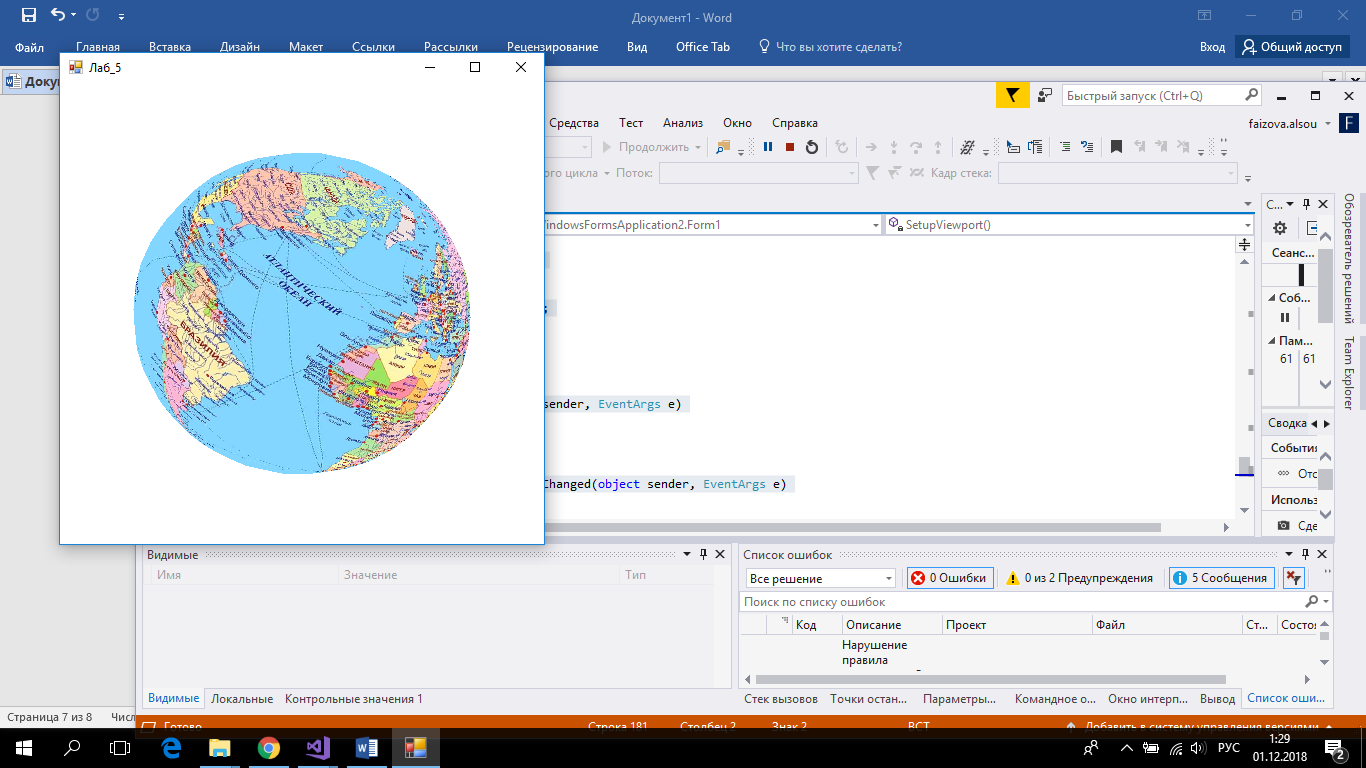
// метод UnlockBits объекта Bitmap разблокирует объект в памяти

}

}

}

Пример выполнения программы



**Вывод:**

На данной лабораторной работе мы разработали OpenGL-приложение, которое накладывает текстуру на шар.