Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования "Восточно-Сибирский государственный университет технологий и управления"

Факультет компьютерных наук и технологий

Кафедра «Программная инженерия и искусственный интеллект»

**Курсовая работа**

по дисциплине "Геометрия и топология"

на тему "Разработка приложения с использованием OpenGL для построения динамического изображения трехмерной модели объекта "Батискаф""

Выполнил: студент г. Б720 Дондупова Л. Ч.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель: ст. преподаватель Цагадаева Е. Н.

Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_ Дата зашиты \_\_\_\_\_\_\_\_\_\_\_

Улан-Удэ

2022

**АННОТАЦИЯ**

Приложение написано на языке С# в интерфейсе программирования Windows Forms с использованием библиотеки OpenGL, Tao Framework и GLUT. Программа создана в среде Visual Studio 2019. В рамках работы были выполнены все представляемые задачи. Имеется возможность производить различные настройки, такие как: настройка и выбор перспективы. Модель анимирована, ею можно управлять перетаскиванием ползунками. Программа имеет интуитивно понятный интерфейс, который схож с другими Windows – приложениям, что указывает на массовость программного продукта.

# **ОГЛАВЛЕНИЕ**

АННОТАЦИЯ………………………………………………………………………….………..2

ВВЕДЕНИЕ…………………………………………………………………………….……..…5

1. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OPENGL……………………………...…………………………………………………....….…6

1.1 Основные возможности.………………...……..………….………………….…………….6

1.2 Работа с матрицами …………….……………………………….…………………….……7

2. Библиотека Tao Framework …………………….………………………..……………….…9

2.1 Библиотека GLUT …………..………………………………….………………….……....10

2.2 Библиотека DevIL …….………………………………………..………………….………11

3. РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ПОСТОРОЕНИЯ ДИНАМИЧЕСКОГО ИЗОБРАЖЕНИЯ ТРЕХМЕРНОЙ МОДЕЛИ ОБЪЕКТА "БАТИСКАФ"..………..………12

3.1 Визуализации трехмерной модели с помощью Autodesk 3ds Max…………………………………………………………………...………………….……...13

3.2 Разработка окна программы…………...……………………….…………………..……..15

3.3 Структура и функционал проекта…….………………………………………............….16

4. ЗАКЛЮЧЕНИЕ………………………………………………………….……………….…17

СПИСОК ЛИТЕРАТУРЫ……………………………………………………………....….....18

ПРИЛОЖЕНИЕ А. Исходный код…………..……………………………..………………...39

ПРИЛОЖЕНИЕ Б…………………………......……………………………..………………..40

**ВВЕДЕНИЕ**

От архитектуры или естественных наук до машиностроения, моды или медицинской промышленности - все отрасли промышленности сейчас видят весь потенциал трёхмерной графики и максимально используют его. Трёхмерная графика на самом деле может быть полезна самыми разными способами.

Это незаменимый инструмент для анимации, как для фильмов, так и для видеоигр. Программное обеспечение для трёхмерной графики позволяет создавать все, от персонажей до ландшафта, со световыми эффектами, текстурами и многим другим.

Для более технических применений моделирование позволит тестировать технические детали без необходимости их изготовления.

Раньше, когда трёхмерная графика существовала только на суперкомпьютерах, не существовало единого стандарта в области графики. Разработка программного продукта, способного работать на большом количестве графического оборудования, была сопряжена с большими временными и финансовыми затратами. Было необходимо отдельно создавать модули для каждого типа графических адаптеров, что порой приводило к размножению одинакового программного кода. Это сильно тормозило развитие и распространение компьютерной графики.

С приходом мощных процессоров и графических ускорителей трёхмерная графика стала реальностью для персональных компьютеров. Но в тоже время производители программного обеспечения снова столкнулись с серьёзной проблемой – это отсутствие каких-либо стандартов, которые позволяли писать программы, независимые от оборудования и операционной системы. Одним из первых таких стандартов, существующий и по сей день является OpenGL.

В начале 1990-х годов SGI была лидером на рынке рабочих станций для 3D-графики благодаря их высокопроизводительному оборудованию и простому в использовании программному обеспечению. Сделав смелый шаг, SGI очистила все функциональные возможности, которые не имели отношения к компьютерной графике, и выпустила его для широкой публики в 1992 году как OpenGL (Open Graphics Library), кроссплатформенный стандартизированный API для компьютерной графики в реальном времени.

На самом фундаментальном уровне OpenGL - это программный интерфейс, который позволяет программисту взаимодействовать с графическим оборудованием. Программы, написанные с помощью OpenGL, можно переносить практически на любые платформы и также с помощью него программист освобождается от написания программ для конкретного оборудования.

Он включает в себя около 150 различных команд, с помощью которых программист может определять различные объекты и производить рендеринг. Говоря более простым языком, вы определяете объекты, задаёте их местоположение в трёхмерном пространстве, определяете другие параметры (поворот, масштаб, ...), задаёте свойства объектов (цвет, текстура, материал, ...), положение наблюдателя, а библиотека OpenGL позаботится о том чтобы отобразить всё это на экране. Поэтому можно сказать, что библиотека OpenGL является только воспроизводящей (Rendering), и занимается только отображением 3D обьектов, она не работает с устройствами ввода (клавиатуры, мыши). Также она не поддерживает менеджер окон. OpenGL имеет хорошо продуманную внутреннюю структуру и довольно простой процедурный интерфейс. Несмотря на это с помощью OpenGL можно создавать сложные и мощные программные комплексы, затрачивая при этом минимальное время по сравнению с другими графическими библиотеками.

## **1. ПРОГРАММИРОВАНИЕ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OPENGL**

**1.1 Основные возможности**

Возможности OpenGL описаны через функции его библиотеки. Все функции встроены в библиотеки, которые можно разделить на пять категорий:

1. Ядро OpenGL (GL): состоит из сотен команд, которые моделируют объект с помощью набора геометрических примитивов, таких как точка, линия и многоугольник.
2. Библиотека утилит OpenGL (GLU): построена поверх ядра OpenGL для предоставления важных утилит (таких как настройка вида камеры и проекции) и дополнительных моделей построения (таких как qradric surfaces и polygon tessellation).
3. OpenGL Utilities Toolkit (GLUT): OpenGL разработан таким образом, чтобы быть независимым от оконной системы или операционной системы. GLUT необходим для взаимодействия с операционной системой (например, для создания окна, обработки вводов клавиш и мыши); он также предоставляет больше моделей построения (таких как сфера и тор).
4. Библиотека OpenGL Extension Wrangler Library (GLEW): "GLEW - это кроссплатформенная библиотека загрузки расширений C/C++ с открытым исходным кодом. GLEW предоставляет эффективные механизмы выполнения для определения того, какие расширения OpenGL поддерживаются на целевой платформе."
5. Другие.

**1.2 Работа с матрицами**

Для задания различных преобразований объектов сцены в OpenGL используются операции над матрицами, при этом различают три типа матриц: модельно-видовая, матрица проекций и матрица текстуры. Все они имеют размер 4x4.

Видовая матрица определяет отдельные преобразования точек и полигонов в пространстве обрабатываются базовыми матрицами преобразований, такими как перевод, масштабирование и поворот. Матрица проекций определяет, как будут проецироваться трехмерные объекты на плоскость экрана (в оконные координаты), а матрица текстуры определяет наложение текстуры на объект.

Эти составленные матрицы в конечном счете перемещают исходные данные модели в специальное координатное пространство, называемое пространством клипов.

Умножение координат на матрицы происходит в момент вызова соответствующей команды OpenGL, определяющей координату (как правило, это команда glVertex\*.

Для того чтобы выбрать, какую матрицу надо изменить, используется команда со значением параметра "mode" равным GL\_MODELVIEW, GL\_PROJECTION, или GL\_TEXTURE включает режим работы с модельно-видовой матрицей, матрицей проекций, или матрицей текстуры соответственно. Для вызова команд, задающих матрицы того или иного типа, необходимо сначала установить соответствующий режим.

**2. Библиотека Tao Framework**

Tao Framework – это свободно-распространяемая библиотека с открытым исходным кодом, предназначенная для быстрой и удобной разработки кросс-платформенного мультимедийного программного обеспечения в среде .NET Framework и Mono. Первоначально он был разработан программистом C # OpenGL Рэнди Риджем, и с момента его создания многие разработчики внесли свой вклад в проект. Последней версией Tao является версия 2.1, выпущенная 1 мая 2008 года. На сегодняшний день Tao Framework - это лучший путь для использования библиотеки OpenGL при разработке в среде .NET на языке C#.

В состав библиотеки на данный момент входят все современные средства, которые могут понадобиться в ходе разработки мультимедиа программного обеспечения.

Библиотека Tao Framework является мощным свободно распространяемым инструментом для решения любых мультимедийных задач преимущественно кросс-платформенного характера. Работая с данной библиотекой, разработчик или группа разработчиков могут использовать базу алгоритмов и реализованных за многие годы методов, что в десятки, а то и сотни раз сокращает время разработки программных продуктов.

Преимущества компонентов, выбранных нами для изучения программирования компьютерной графики, становятся очевидными: язык C# является на данный момент одним из самых современных и удобных для быстрой разработки языков программирования. Кроме того то, что он вобрал все лучшее от существующих языков программирования и учел все их недостатки, делает его самым перспективным в плане изучения.

В данном случае основными библиотеками:

* Tao.OpenGL.dll - отвечает за реализация библиотеки OpenGL.
* Tao.FreeGlut.dll - отвечает за реализацию функций библиотеки и используется для инициализации рендера, а также для различных других целей.
* Tao.Platform.Windows.dll - отвечает за поддержку элементов непосредственно для визуализации на платформе Windows.

**2.1 Библиотека GLUT**

GLUT - это OpenGL Utility Toolkit, независимый от оконной системы инструментарий для написания программ на OpenGL. Он реализует простой интерфейс программирования оконных приложений (API) для OpenGL. GLUT значительно облегчает изучение программирования на OpenGL b предоставляет портативный API, так что вы можете написать единую программу OpenGL, которая работает как на компьютерах Win32, так и на рабочих станциях X11.

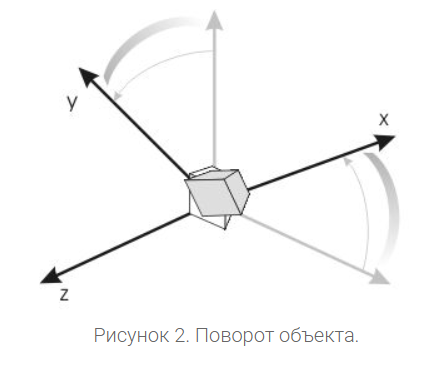
GLUT предназначен для создания программ OpenGL малого и среднего размера. Данная библиотека не является полнофункциональным инструментарием, поэтому большим приложениям, требующим сложных пользовательских интерфейсов, лучше использовать собственные наборы инструментов оконной системы, такие как Motif.

Библиотека поддерживает следующие функциональные возможности:

* Несколько окон для рендеринга OpenGL.
* Обработка событий, управляемых обратным вызовом.
* Процедура "простоя" и таймеры.
* Служебные процедуры для создания различных объектов из цельного и проволочного каркаса.
* Поддержка растровых и штриховых шрифтов.
* Различные функции управления окнами.

Библиотека GLUT была разработана Марком Килгардом, когда он работал в Silicon Graphics Inc.

Все функции GLUT начинаются с префикса glut (например, glutPostRedisplay отмечает текущее окно, как требующее перерисовки). В OpenGL API существуют команды для модельных преобразований: glTranslate\*(), glRotate\*(), glScale\*(). С помощью данных выполняются трансформации объекта или координатной системы. Данные команды являются эквивалентом соответствующих матриц с вызовом функции glMultMatrix\*() с нужной матрицей в качестве аргумента. Но в данном случае использование команд glTranslate(рис.1), glRotate(рис.2) и glScale может выполниться быстрее. GlScale\*() – является единственной из трех команд модельных преобразований, которая изменяет размер объекта.





**2.2 Библиотека DevIL**

Одной из задач, с которой постоянно приходится сталкиваться при программировании графики, является загрузка и сохранение текстур (изображений).

Существуют многие десятки различных форматов, в которых могут быть заданы текстуры и желательно уметь поддерживать чтение из большинства таких форматов и запись в несколько основных.

Библиотека [DevIL](http://openil.sourceforge.net/) (изначально она называлась OpenIL) как раз и предназначена для этого. DevIL является простой и кроссплатформенной библиотекой для чтения и записи изображений (текстур) из/в большое число различных графических форматов.

При этом сама библиотека организована по образу и подобию OpenGL - она представляет из себя набор функций и констант, разбитых на три модуля - IL, ILU и ILUT.

Собственно, за чтение и запись текстур отвечает именно библиотека IL, библиотека ILU предоставляет ряд возможностей по обработке изображений, а библиотека ILUT служит для упрощения работы с IL для сторонних библиотек (таких как OpenGL).

Библиотека DevIL предоставляет сразу несколько способов загрузки текстур в OpenGL и сохранения скриншотов. В простейшем случае для этого достаточно вызова всего одной функции из ILUT.

**3. РАЗРАБОТКА ПРИЛОЖЕНИЯ ДЛЯ ПОСТОРОЕНИЯ ДИНАМИЧЕСКОГО ИЗОБРАЖЕНИЯ ТРЕХМЕРНОЙ МОДЕЛИ ОБЪЕКТА "БАТИСКАФ"**

**3.1 Визуализации трехмерной модели с помощью Autodesk 3ds Max.**

Для рисования различных геометрических фигур используется библиотека GLUT и для прорисовки уже трехмерных фигур понадобилось бы совмещение довольно большого числа примитивов. Для упрощения реализации модели «батискаф» используется другая библиотека.

Библиотека [DevIL](http://openil.sourceforge.net/) является простой и кроссплатформенной библиотекой для чтения и записи изображений (текстур) из/в большое число различных графических форматов. С помощью него можно загружать уже готовую трехмерную модель, которую заранее смоделировали в сторонней программе, предназначенная для создания сложных графических сцен - 3D Studio Max. Экспортируя трехмерный объект из пакета трехмерного моделирования, передается файл, который хранит геометрические данные, текстурные координаты и т. д.

Загрузку трехмерной модели производится в формате ASE. В данный формат можно легко сохранить трехмерную модель, используя пакет трехмерного моделирования Autodesk 3ds Max. Метод хранения геометрии в сторонних файлах очень удобен т. к. имея класс загрузки трехмерных объектов, мы можем на ходу менять геометрию сцены, меняя файл с описанием трехмерного объекта.

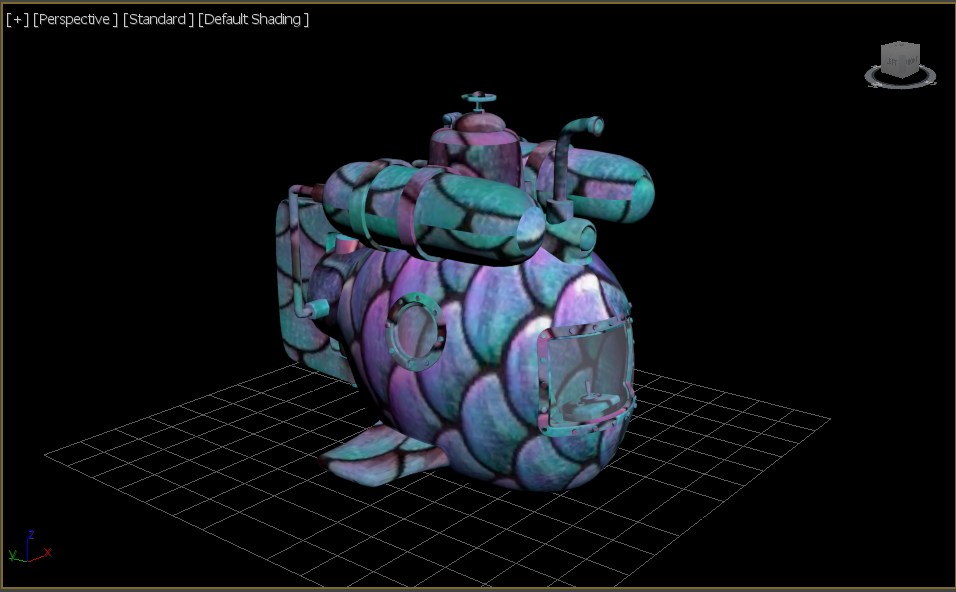


Рисунок 3 – Батискаф в Autodesk 3ds Max

**3.2 Разработка окна программы**

Для данной программы разработан интерфейс, позволяющий (рис. 4):

* Загрузка готовой 3D модели c расширением .ase;
* Вращение камеры по трем осям (x,y,z);
* Перемещение модели по трем осям (x,y,z);
* Масштабирование(zoom).

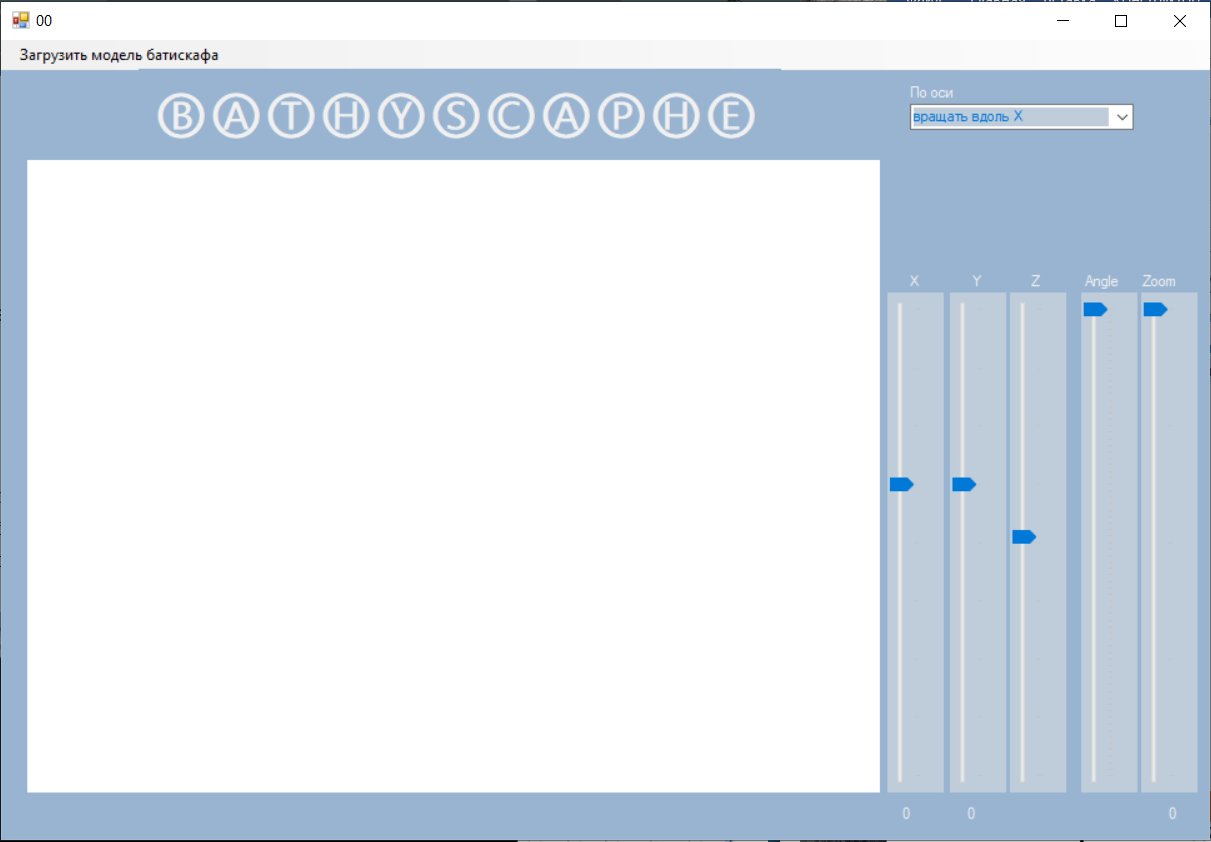


Рисунок 4 – Интерфейс окна программы

Любое windows-положение основано на событиях и их обработке, другими словами, поведение программы управляется событиями. Данный проект тоже является windows приложением, и, следовательно, обработка событий занимает важное место. К основным событиям, играющим важную, для корректной работы программы, роль относятся следующие:

* private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e) – выбор оси(x,y,z)
* private void загрузитьМодельToolStripMenuItem\_Click(object sender, EventArgs e) – загрузка модели формата .ase.
* private void trackBar1\_Scroll(object sender, EventArgs e) – для перемещения по оси X.
* private void trackBar2\_Scroll(object sender, EventArgs e) – для перемещения по оси Y.
* private void trackBar3\_Scroll(object sender, EventArgs e) – для перемещения по оси Z.
* private void trackBar4\_Scroll(object sender, EventArgs e) – для изменения угла просмотра Angle.

**3.3 Структура и функционал проекта**

Основным элементом в процессе программы является выгрузка и чтение файла 3D-модели и сцены так как файл содержит информацию о координатах модели, текстуры и всех ее вершин.

Алгоритм считывания информации из файла:

1. Построчное чтение файла (вся необходимая информация для любой управляющей конструкции будет находиться в рамках одной строки, поэтому построчное чтение нам подходит)

2. Отсечение первого слова в строке.

3. Проверка первого символа в полученном первом слове – если оно равно «\*», то мы имеем управляющее слово.

4. На основе управляющего слова мы проверяем, что это за управляющее слово, и если оно относится к тем, которые необходимы нам (описывают геометрию, текстурные координаты), то мы обрабатываем данную строку, отрезая первые слова и затем переводя их из строкового представления к числовому. Для наглядности приведем таблицу наиболее важных методов - таблица 1.

|  |  |  |
| --- | --- | --- |
| № | Метод | Назначение |
| 1 | public void createTextureVertexMem(int a) | Массивы для текстурных координат |
| 2 | public void LoadTextureForModel(string FileName) | Загрузка текстуры |
| 3 | private static uint MakeGlTexture(int Format, IntPtr pixels, int w, int h) | Создание текстуры в памяти OpenGL |
| 4 | public int LoadModel(string FileName) | Загрузка модели |
| 5 | private void CreateList() | Функция отрисовки |
| 6 | private string GetFirstWord(string word, int from) | Чтение загружаемого файла |
| 7 | public void DrawModel() | Функция отрисовки 3D модели |
| 8 | public void createTextureVertexMem(int a) | Массивы для текстурных координат |
| 9 | private void memcompl() | Память для геометрии |
| 10 | public int RotateModel(int os, float target, float step) | Вращение 3D-модели |

Таблица 1 – Основные методы и функции

|  |  |
| --- | --- |
| Название | Назначение |
| Limb | отвечает за логические единицы 3D объектов в загружаемой сцене |
| TexturesForObjects | класс для работы с текстурами |
| anModelLoader | класс, для загрузки 3D модели |

Таблица 2 – Спецификация классов

**ЗАКЛЮЧЕНИЕ**

В ходе разработки данного приложения были получены практические навыки по разработке программ для операционных систем семейства Windows с применением технологий трехмерной графики с использованием библиотеки OpenGL, GLUT, DevIL.

Таким образом, можно выделить следующие решенные в рамках данной курсовой работы задачи:

* изучение принципов работы OpenGL в оконной среде Windows;
* получение практических навыков использования средств OpenGL;
* изучение принципов работы Autodesk 3ds Max;
* получение навыков программирования интерактивных трехмерных приложений.

**СПИСОК ЛИТЕРАТУРЫ**

* 1. OpenGL Tutorial An Introduction on OpenGL with 3D Graphics <https://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg_introduction.html>
  2. OpenGL Utility Toolkit  
     http://esate.ru/uroki/OpenGL/uroki-OpenGL-c-sharp/biblioteka-glut-opengl/
  3. Тихомиров Ю. Программирование трехмерной графики. СПб,
  4. Performance OpenGL: Platform Independent Techniques.
  5. SIGGRAPH 2001 course.
  6. OpenGL performance optimization, Siggraph'97 course.
  7. Visual Introduction in OpenGL, SIGGRAPH'98.
  8. The OpenGL graphics system: a specification (version 1.1).
  9. Программирование GLUT: окна и анимация. Miguel Angel Sepulveda, LinuxFocus.
  10. The OpenGL Utility Toolkit (GLUT) Programming Interface, API version 3, specification.
  11. Работа с матрицами <https://studbooks.net/2224667/informatika/rabota_matritsami>
  12. OpenGL Utility Toolkit  
      http://esate.ru/uroki/OpenGL/uroki-OpenGL-c-sharp/biblioteka-glut-opengl/

**ПРИЛОЖЕНИЕ А. Исходный код**

/////////////////////////////////////////////////////////Form1.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 Tao.DevIl;

using Tao.FreeGlut;

using Tao.OpenGl;

using Tao.Platform.Windows;

namespace Model

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

double a = 0, b = 0, c = -5, d = 0, zoom = 1;

int os\_x = 1, os\_y = 0, os\_z = 0;

bool Wire = false;

anModelLoader Model = null;

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

{

Glut.glutInit();

Glut.glutInitDisplayMode(Glut.GLUT\_RGB | Glut.GLUT\_DOUBLE | Glut.GLUT\_DEPTH);

Il.ilInit();

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

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

Gl.glViewport(0, 0, AnT.Width, AnT.Height);

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

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

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

Gl.glLoadIdentity();

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

Gl.glEnable(Gl.GL\_LIGHTING);

Gl.glEnable(Gl.GL\_LIGHT0);

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

Gl.glEnable(Gl.GL\_BLEND);

Gl.glEnable(Gl.GL\_LINE\_SMOOTH);

Gl.glLineWidth(1.0f);

comboBox1.SelectedIndex = 0;

// опции для загрузки файла

openFileDialog1.Filter = "ase files (\*.ase)|\*.ase|All files (\*.\*)|\*.\*";

}

private void Draw()

{

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

Gl.glLoadIdentity();

Gl.glColor3i(255, 0, 0);

Gl.glPushMatrix();

Gl.glTranslated(a, b, c);

Gl.glRotated(d, os\_x, os\_y, os\_z);

Gl.glScaled(zoom, zoom, zoom);

if(Model != null)

Model.DrawModel();

Gl.glPopMatrix();

Gl.glFlush();

AnT.Invalidate();

}

private void RenderTimer\_Tick(object sender, EventArgs e)

{

Draw();

}

private void trackBar1\_Scroll(object sender, EventArgs e)

{

a = (double)trackBar1.Value / 1000.0;

label4.Text = a.ToString();

Draw();

}

private void trackBar2\_Scroll(object sender, EventArgs e)

{

b = (double)trackBar2.Value / 1000.0;

label5.Text = b.ToString();

Draw();

}

private void trackBar3\_Scroll(object sender, EventArgs e)

{

c = (double)trackBar3.Value / 1000.0;

label6.Text = c.ToString();

Draw();

}

private void trackBar4\_Scroll(object sender, EventArgs e)

{

d = (double)trackBar4.Value;

label6.Text = d.ToString();

Draw();

}

private void trackBar5\_Scroll(object sender, EventArgs e)

{

zoom = (double)trackBar5.Value / 1000.0;

label6.Text = zoom.ToString();

Draw();

}

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

{

switch (comboBox1.SelectedIndex)

{

case 0:

{

os\_x = 1;

os\_y = 0;

os\_z = 0;

break;

}

case 1:

{

os\_x = 0;

os\_y = 1;

os\_z = 0;

break;

}

case 2:

{

os\_x = 0;

os\_y = 0;

os\_z = 1;

break;

}

}

Draw();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

Draw();

}

// загрузка модели

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

{

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

Model = new anModelLoader();

Model.LoadModel(openFileDialog1.FileName);

RenderTimer.Start();

}

}

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

{

}

private void label6\_Click(object sender, EventArgs e)

{

}

private void label10\_Click(object sender, EventArgs e)

{

}

}

}

/////////////////////////////////////////////////////////anModelLoader.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.IO;

using System.Windows.Forms;

using Tao.OpenGl;

using Tao.FreeGlut;

using Tao.Platform.Windows;

using Tao.DevIl;

namespace Model

{

// класс LIMB отвечает за логические единицы 3D объектов в загружаемой сцене

class LIMB

{

// при инициализации мы должны указать количество вершин (vertex) и полигонов (face) которые описывают геометри под-объекта

public LIMB(int a, int b)

{

if (temp[0] == 0)

temp[0] = 1;

// записываем количество вершин и полигонов

VandF[0] = a;

VandF[1] = b;

// выделяем память

memcompl();

}

public int Itog; // флаг успешности

// массивы для хранения данных (геометрии и текстурных координат)

public float[,] vert;

public int[,] face;

public float[,] t\_vert;

public int[,] t\_face;

// номер материала (текстуры) данного под-объекта

private int MaterialNom = -1;

// временное хранение информации

public int[] VandF = new int[4];

private int[] temp = new int[2];

// флаг , говорящий о том, что модель использует текстуру

private bool ModelHasTexture = false;

// функция для определения значения флага (о наличии текстуры)

public bool NeedTexture()

{

// возвращаем значение флага

return ModelHasTexture;

}

public void SetMaterialNom(int new\_nom)

{

MaterialNom = new\_nom;

if (MaterialNom > -1)

// отмечаем флаг о наличии текстуры

ModelHasTexture = true;

}

// массивы для текстурных координат

public void createTextureVertexMem(int a)

{

VandF[2] = a;

t\_vert = new float[3, VandF[2]];

}

// привязка значений текстурных координат к полигонам

public void createTextureFaceMem(int b)

{

VandF[3] = b;

t\_face = new int[3, VandF[3]];

}

// память для геометрии

private void memcompl()

{

vert = new float[3, VandF[0]];

face = new int[3, VandF[1]];

}

// номер текстуры

public int GetTextureNom()

{

return MaterialNom;

}

};

// класс для работы с текстурами

public class TexturesForObjects

{

public TexturesForObjects()

}

}

// имя текстуры

private string texture\_name = "";

// ее ID

private int imageId = 0;

// идетификатор текстуры в памяти openGL

private uint mGlTextureObject = 0;

// получение этого идентификатора

public uint GetTextureObj()

{

return mGlTextureObject;

}

// загрузка текстуры

public void LoadTextureForModel(string FileName)

{

// запоминаем имя файла

texture\_name = FileName;

// создаем изображение с индификатором imageId

Il.ilGenImages(1, out imageId);

// делаем изображение текущим

Il.ilBindImage(imageId);

string url = texture\_name;

// если загрузка удалась

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 оп 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);

}

}

// создание текстуры в памяти openGL

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

{

// индетефекатор 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 индетефекатор текстурного объекта

return texObject;

}

}

// небольшой класс для описания ориентации модели в 3D пространстве сцены

public class Model\_Prop

{

public Model\_Prop()

{

pos\_abs[0] = 0;

pos\_abs[1] = 0;

pos\_abs[2] = 0;

maximum[0] = 0;

maximum[1] = 0;

maximum[2] = 0;

minimum[0] = 0;

minimum[1] = 0;

minimum[2] = 0;

rotating\_angles[0] = 0;

rotating\_angles[1] = 0;

rotating\_angles[2] = 0;

}

public float[] pos\_abs = new float[3];

public float[] maximum = new float[3];

public float[] minimum = new float[3];

public float[] rotating\_angles = new float[3];

};

// класс, выполняющий загрузку 3D модели

public class anModelLoader

{

public anModelLoader()

{

}

// имя файла

public string FName = "";

// загружен ли (флаг)

private bool isLoad = false;

// счетчик по-объектов

private int count\_limbs;

// переменная для зранения номера текстуры

private int mat\_nom = 0;

// номер дисплейног осписка с данной моделью

private int thisList = 0;

// данная переменная будет указывать на количество прочитанных символов в строке при чтении информации из файла

private int GlobalStringFrom = 0;

// массив под-объектов

LIMB[] limbs = null;

// массви для хранения текстур

TexturesForObjects[] text\_objects = null;

// описание ориентации модели

Model\_Prop coord = new Model\_Prop();

// установка минимумов и максимумов для размещения модели

public void SetMinimum(float x, float y, float z)

{

coord.minimum[0] = x;

coord.minimum[1] = y;

coord.minimum[2] = z;

}

public void SetMaximum(float x, float y, float z)

{

coord.maximum[0] = x;

coord.maximum[1] = y;

coord.maximum[2] = z;

}

public void SetAbsCoords(float x, float y, float z)

{

coord.pos\_abs[0] = x;

coord.pos\_abs[1] = y;

coord.pos\_abs[2] = z;

}

// вращение 3D модели

public int RotateModel(int os, float target, float step)

{

if ((coord.rotating\_angles[os] - target) > 0)

{

coord.rotating\_angles[os] -= step;

if (coord.rotating\_angles[os] < target)

{

coord.rotating\_angles[os] = target;

return -1;

}

}

else

{

coord.rotating\_angles[os] += step;

if (coord.rotating\_angles[os] > target)

{

coord.rotating\_angles[os] = target;

return -1;

}

}

return 0;

}

// перемещение модели

public int MoveModel(int os, float target, float step)

{

if (step == 0)

return -1;

float real\_target = target;

if ((coord.pos\_abs[os] - real\_target) > 0)

{

if (coord.pos\_abs[os] - step >= coord.minimum[os])

{

coord.pos\_abs[os] -= step;

if (coord.pos\_abs[os] < real\_target)

{

coord.pos\_abs[os] = real\_target;

return -1;

}

return 0;

}

else

{

coord.pos\_abs[os] = coord.minimum[os];

return -1;

}

}

if ((coord.pos\_abs[os] - real\_target) < 0)

{

if (coord.pos\_abs[os] + step <= coord.maximum[os])

{

coord.pos\_abs[os] += step;

if (coord.pos\_abs[os] > real\_target)

{

coord.pos\_abs[os] = real\_target;

return -1;

}

return 0;

}

else

{

coord.pos\_abs[os] = coord.maximum[os];

return -1;

}

}

if ((coord.pos\_abs[os] - real\_target) == 0)

return -1;

return 0;

}

// загрузка модели

public int LoadModel(string FileName)

{

// модель может содержать до 256 под-объектов

limbs = new LIMB[256];

// счетчик скинут

int limb\_ = -1;

// имся файла

FName = FileName;

// начинаем чтение файла

StreamReader sw = File.OpenText(FileName);

// временные буферы

string a\_buff = "";

string b\_buff = "";

string c\_buff = "";

// счетчики вершин и полигонов

int ver = 0, fac = 0;

// если строка успешно прочитана

while ((a\_buff = sw.ReadLine()) != null)

{

// получаем первое слово

b\_buff = GetFirstWord(a\_buff, 0);

if (b\_buff[0] == '\*') // определеям, является ли первый символ звездочкой

{

switch (b\_buff) // если да, то проверяем какое управляющее слово содержится в первом прочитаном слове

{

case "\*MATERIAL\_COUNT": // счетчик материалов

{

// получаем первое слово от символа указанного в GlobalStringFrom

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

int mat = System.Convert.ToInt32(c\_buff);

// создаем объект для текстуры в памяти

text\_objects = new TexturesForObjects[mat];

continue;

}

case "\*MATERIAL\_REF": // номер текстуры

{

// записываем для текущего под-объекта номер текстуры

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

int mat\_ref = System.Convert.ToInt32(c\_buff);

// устанавливаем номер материала, соответствующий данной модели.

limbs[limb\_].SetMaterialNom(mat\_ref);

continue;

}

case "\*MATERIAL": // указание на материал

{

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

mat\_nom = System.Convert.ToInt32(c\_buff);

continue;

}

case "\*GEOMOBJECT": // начинается описание геметрии под-объекта

{

limb\_++; // записываем в счетчик под-объектов

continue;

}

case "\*MESH\_NUMVERTEX": // количесвто вершин в под-объекте

{

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

ver = System.Convert.ToInt32(c\_buff);

continue;

}

case "\*BITMAP": // имя текстуры

{

c\_buff = ""; // обнуляем временный буффер

for (int ax = GlobalStringFrom + 2; ax < a\_buff.Length - 1; ax++)

c\_buff += a\_buff[ax]; // считываем имя текстуры

text\_objects[mat\_nom] = new TexturesForObjects(); // новый объект для текстуры

text\_objects[mat\_nom].LoadTextureForModel(c\_buff); // загружаем текстуру

continue;

}

case "\*MESH\_NUMTVERTEX": // количество текстурных координат, данное слово говорит о наличии текстурных координат - следовательно мы должны выделить память для них

{

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

if (limbs[limb\_] != null)

{

limbs[limb\_].createTextureVertexMem(System.Convert.ToInt32(c\_buff));

}

continue;

}

case "\*MESH\_NUMTVFACES": // память для текстурных координат (faces)

{

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

if (limbs[limb\_] != null)

{

// выделяем память для текстурныйх координат

limbs[limb\_].createTextureFaceMem(System.Convert.ToInt32(c\_buff));

}

continue;

}

case "\*MESH\_NUMFACES": // количество полиговов в под-объекте

{

c\_buff = GetFirstWord(a\_buff, GlobalStringFrom);

fac = System.Convert.ToInt32(c\_buff);

// если было объвляющее слово \*GEOMOBJECT (гарантия выполнения условия limb\_ > -1) и были указаны количство вершин

if (limb\_ > -1 && ver > -1 && fac > -1)

{

// создаем новый под-объект в памяти

limbs[limb\_] = new LIMB(ver, fac);

}

else

{

// иначе завершаем неудачей

return -1;

}

continue;

}

case "\*MESH\_VERTEX": // информация о вершине

{

// под-объект создан в памяти

if (limb\_ == -1)

return -2;

if (limbs[limb\_] == null)

return -3;

string a1 = "", a2 = "", a3 = "", a4 = "";

// полчучаем информацию о кооринатах и номере вершины

// (получаем все слова в строке)

a1 = GetFirstWord(a\_buff, GlobalStringFrom);

a2 = GetFirstWord(a\_buff, GlobalStringFrom);

a3 = GetFirstWord(a\_buff, GlobalStringFrom);

a4 = GetFirstWord(a\_buff, GlobalStringFrom);

// преобразовываем в целое цисло

int NomVertex = System.Convert.ToInt32(a1);

// заменяем точки в представлении числа с плавающей точкой, на запятые, чтобы правильно выполнилась функция

// преобразования строки в дробное число

a2 = a2.Replace('.', ',');

a3 = a3.Replace('.', ',');

a4 = a4.Replace('.', ',');

// записываем информацию о вершине

limbs[limb\_].vert[0, NomVertex] = (float)System.Convert.ToDouble(a2); // x

limbs[limb\_].vert[1, NomVertex] = (float)System.Convert.ToDouble(a3); // y

limbs[limb\_].vert[2, NomVertex] = (float)System.Convert.ToDouble(a4); // z

continue;

}

case "\*MESH\_FACE": // информация о полигоне

{

// под-объект создан в памяти

if (limb\_ == -1)

return -2;

if (limbs[limb\_] == null)

return -3;

// временные перменные

string a1 = "", a2 = "", a3 = "", a4 = "", a5 = "", a6 = "", a7 = "";

// получаем все слова в строке

a1 = GetFirstWord(a\_buff, GlobalStringFrom);

a2 = GetFirstWord(a\_buff, GlobalStringFrom);

a3 = GetFirstWord(a\_buff, GlobalStringFrom);

a4 = GetFirstWord(a\_buff, GlobalStringFrom);

a5 = GetFirstWord(a\_buff, GlobalStringFrom);

a6 = GetFirstWord(a\_buff, GlobalStringFrom);

a7 = GetFirstWord(a\_buff, GlobalStringFrom);

// получаем нмоер полигона из первого слова в строке, заменив последний символ ":" после номера на флаг окончания строки.

int NomFace = System.Convert.ToInt32(a1.Replace(':', '\0'));

// записываем номера вершин, которые нас интересуют

limbs[limb\_].face[0, NomFace] = System.Convert.ToInt32(a3);

limbs[limb\_].face[1, NomFace] = System.Convert.ToInt32(a5);

limbs[limb\_].face[2, NomFace] = System.Convert.ToInt32(a7);

continue;

}

// текстурые координаты

case "\*MESH\_TVERT":

{

// под-объект создан в памяти

if (limb\_ == -1)

return -2;

if (limbs[limb\_] == null)

return -3;

// временные перменные

string a1 = "", a2 = "", a3 = "", a4 = "";

// получаем все слова в строке

a1 = GetFirstWord(a\_buff, GlobalStringFrom);

a2 = GetFirstWord(a\_buff, GlobalStringFrom);

a3 = GetFirstWord(a\_buff, GlobalStringFrom);

a4 = GetFirstWord(a\_buff, GlobalStringFrom);

// преобразуем первое слово в номер вершины

int NomVertex = System.Convert.ToInt32(a1);

// заменяем точки в представлении числа с плавающей точкой, на запятые, чтобы правильно выполнилась функция

// преобразования строки в дробное число

a2 = a2.Replace('.', ',');

a3 = a3.Replace('.', ',');

a4 = a4.Replace('.', ',');

// записываем значение вершины

limbs[limb\_].t\_vert[0, NomVertex] = (float)System.Convert.ToDouble(a2); // x

limbs[limb\_].t\_vert[1, NomVertex] = (float)System.Convert.ToDouble(a3); // y

limbs[limb\_].t\_vert[2, NomVertex] = (float)System.Convert.ToDouble(a4); // z

continue;

}

// привязка текстурных координат к полигонам

case "\*MESH\_TFACE":

{

// под-объект создан в памяти

if (limb\_ == -1)

return -2;

if (limbs[limb\_] == null)

return -3;

// временные перменные

string a1 = "", a2 = "", a3 = "", a4 = "";

// получаем все слова в строке

a1 = GetFirstWord(a\_buff, GlobalStringFrom);

a2 = GetFirstWord(a\_buff, GlobalStringFrom);

a3 = GetFirstWord(a\_buff, GlobalStringFrom);

a4 = GetFirstWord(a\_buff, GlobalStringFrom);

// преобразуем первое слово в номер полигона

int NomFace = System.Convert.ToInt32(a1);

// записываем номера вершин, которые описывают полигон

limbs[limb\_].t\_face[0, NomFace] = System.Convert.ToInt32(a2);

limbs[limb\_].t\_face[1, NomFace] = System.Convert.ToInt32(a3);

limbs[limb\_].t\_face[2, NomFace] = System.Convert.ToInt32(a4);

continue;

}}

}}

// пересохраняем количесвто полигонов

count\_limbs = limb\_;

// получаем ID для создаваемого дисплейного списка

int nom\_l = Gl.glGenLists(1);

thisList = nom\_l;

// генерируем новый дисплейный список

Gl.glNewList(nom\_l, Gl.GL\_COMPILE);

// отрисовываем геометрию

CreateList();

// завершаем дисплейный список

Gl.glEndList();

// загрузка завершена

isLoad = true;

return 0;

}

// функция отрисовки

private void CreateList()

{

// сохраняем тек матрицу

Gl.glPushMatrix();

// проходим циклом по всем под-объектам

for (int l = 0; l <= count\_limbs; l++)

{

// если текстура необходима

if (limbs[l].NeedTexture())

if (text\_objects[limbs[l].GetTextureNom()] != null) // текстурный объект существует

{

Gl.glEnable(Gl.GL\_TEXTURE\_2D); // включаем режим текстурирования

// ID текстуры в памяти

uint nn = text\_objects[limbs[l].GetTextureNom()].GetTextureObj();

// активируем (привязываем) эту текстуру

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

}

Gl.glEnable(Gl.GL\_NORMALIZE);

// начинаем отрисовку полигонов

Gl.glBegin(Gl.GL\_TRIANGLES);

// по всем полигонам

for (int i = 0; i < limbs[l].VandF[1]; i++)

{

// временные переменные, чтобы код был более понятен

float x1, x2, x3, y1, y2, y3, z1, z2, z3 = 0;

// вытакскиваем координаты треугольника (полигона)

x1 = limbs[l].vert[0, limbs[l].face[0, i]];

x2 = limbs[l].vert[0, limbs[l].face[1, i]];

x3 = limbs[l].vert[0, limbs[l].face[2, i]];

y1 = limbs[l].vert[1, limbs[l].face[0, i]];

y2 = limbs[l].vert[1, limbs[l].face[1, i]];

y3 = limbs[l].vert[1, limbs[l].face[2, i]];

z1 = limbs[l].vert[2, limbs[l].face[0, i]];

z2 = limbs[l].vert[2, limbs[l].face[1, i]];

z3 = limbs[l].vert[2, limbs[l].face[2, i]];

// рассчитываем номраль

float n1 = (y2 - y1) \* (z3 - z1) - (y3 - y1) \* (z2 - z1);

float n2 = (z2 - z1) \* (x3 - x1) - (z3 - z1) \* (x2 - x1);

float n3 = (x2 - x1) \* (y3 - y1) - (x3 - x1) \* (y2 - y1);

// устанавливаем номраль

Gl.glNormal3f(n1, n2, n3);

// если установлена текстура

if (limbs[l].NeedTexture() && (limbs[l].t\_vert != null) && (limbs[l].t\_face != null))

{

// устанавливаем текстурные координаты для каждой вершины, ну и сами вершины

Gl.glTexCoord2f(limbs[l].t\_vert[0, limbs[l].t\_face[0, i]], limbs[l].t\_vert[1, limbs[l].t\_face[0, i]]);

Gl.glVertex3f(x1, y1, z1);

Gl.glTexCoord2f(limbs[l].t\_vert[0, limbs[l].t\_face[1, i]], limbs[l].t\_vert[1, limbs[l].t\_face[1, i]]);

Gl.glVertex3f(x2, y2, z2);

Gl.glTexCoord2f(limbs[l].t\_vert[0, limbs[l].t\_face[2, i]], limbs[l].t\_vert[1, limbs[l].t\_face[2, i]]);

Gl.glVertex3f(x3, y3, z3);

}

else // иначе - отрисовка только вершин

{

Gl.glVertex3f(x1, y1, z1);

Gl.glVertex3f(x2, y2, z2);

Gl.glVertex3f(x3, y3, z3);

}

}

// завершаем отрисовку

Gl.glEnd();

Gl.glDisable(Gl.GL\_NORMALIZE);

// открлючаем текстурирование

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

}

// возвращаем сохраненную ранее матрицу

Gl.glPopMatrix();

}

// функиц я получения первого слова строки

private string GetFirstWord(string word, int from)

{

// from указывает на позицию, начиная с которой будет выполнятся чтение файла

char a = word[from]; // первый символ

string res\_buff = ""; // временный буффер

int L = word.Length; // длина слова

if (word[from] == ' ' || word[from] == '\t') // если первый символ, с которого предстоит искать слово является пробелом или знаком табуляции

{

// необходимо вычисслить наличие секции проблеов или знаков табуляции и откинуть их

int ax = 0;

// проходим до конца слова

for (ax = from; ax < L; ax++)

{

a = word[ax];

if (a != ' ' && a != '\t') // если встречаем символ пробела или табуляции

break; // выходим из цикла.

// таким образом мы откидываем все последовательности пробелов или знаков табуляции, с которых могла начинатся переданная строка

}

if (ax == L) // если вся представленная строка является набором пробелов или знаков табуляции - возвращаем res\_buff

return res\_buff;

else

from = ax; // иначе сохраняем значение ax

}

int bx = 0;

// теперь, когда пробелы и табуляция откинуты мы непосредственно вычисляем слово

for (bx = from; bx < L; bx++)

{

// если встретили знак пробела или табуляции - завершаем чтение слова

if (word[bx] == ' ' || word[bx] == '\t')

break;

// записываем символ в бременный буффер, постепенно получая таким образом слово

res\_buff += word[bx];

}

// если дошли до конца строки

if (bx == L)

bx--; // убераем посл значение

GlobalStringFrom = bx; // позиция в данной строке, для чтения следующего слова в данной строке

return res\_buff; // возвращаем слово

}

// функция отрисовки 3D модели

public void DrawModel()

{

// если модель не загружена - возврат из функции

if (!isLoad)

return;

// сохраняем матрицу

Gl.glPushMatrix();

// масштабирование по умолчанию

Gl.glScalef(0.05f, 0.05f, 0.05f);

// вызов дисплейного списка

//CreateList();

Gl.glCallList(thisList);

// возврат матрицы

Gl.glPopMatrix();

}}

}

/////////////////////////////////////////////////////////Form1.Designer.cs

namespace Model

{

partial class Form1

{

/// <summary>

/// Требуется переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">истинно, если управляемый ресурс должен быть удален; иначе ложно.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм Windows

/// <summary>

/// Обязательный метод для поддержки конструктора - не изменяйте

/// содержимое данного метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

this.components = new System.ComponentModel.Container();

this.AnT = new Tao.Platform.Windows.SimpleOpenGlControl();

this.RenderTimer = new System.Windows.Forms.Timer(this.components);

this.trackBar1 = new System.Windows.Forms.TrackBar();

this.trackBar2 = new System.Windows.Forms.TrackBar();

this.trackBar3 = new System.Windows.Forms.TrackBar();

this.label1 = new System.Windows.Forms.Label();

this.label2 = new System.Windows.Forms.Label();

this.label3 = new System.Windows.Forms.Label();

this.label4 = new System.Windows.Forms.Label();

this.label5 = new System.Windows.Forms.Label();

this.label6 = new System.Windows.Forms.Label();

this.trackBar4 = new System.Windows.Forms.TrackBar();

this.trackBar5 = new System.Windows.Forms.TrackBar();

this.label7 = new System.Windows.Forms.Label();

this.label8 = new System.Windows.Forms.Label();

this.label9 = new System.Windows.Forms.Label();

this.comboBox1 = new System.Windows.Forms.ComboBox();

this.menuStrip1 = new System.Windows.Forms.MenuStrip();

this.загрузитьМодельToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.выбратьФайлДляЗагрузкиToolStripMenuItem = new System.Windows.Forms.ToolStripMenuItem();

this.openFileDialog1 = new System.Windows.Forms.OpenFileDialog();

this.label10 = new System.Windows.Forms.Label();

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar2)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar3)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar4)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar5)).BeginInit();

this.menuStrip1.SuspendLayout();

this.SuspendLayout();

//

// AnT

//

this.AnT.AccumBits = ((byte)(0));

this.AnT.AutoCheckErrors = false;

this.AnT.AutoFinish = false;

this.AnT.AutoMakeCurrent = true;

this.AnT.AutoSwapBuffers = true;

this.AnT.BackColor = System.Drawing.Color.Black;

this.AnT.ColorBits = ((byte)(32));

this.AnT.DepthBits = ((byte)(16));

this.AnT.Location = new System.Drawing.Point(28, 118);

this.AnT.Margin = new System.Windows.Forms.Padding(4);

this.AnT.Name = "AnT";

this.AnT.Size = new System.Drawing.Size(909, 623);

this.AnT.StencilBits = ((byte)(0));

this.AnT.TabIndex = 0;

this.AnT.Load += new System.EventHandler(this.AnT\_Load);

//

// RenderTimer

//

this.RenderTimer.Interval = 25;

this.RenderTimer.Tick += new System.EventHandler(this.RenderTimer\_Tick);

//

// trackBar1

//

this.trackBar1.BackColor = System.Drawing.SystemColors.InactiveCaption;

this.trackBar1.Location = new System.Drawing.Point(945, 249);

this.trackBar1.Margin = new System.Windows.Forms.Padding(4);

this.trackBar1.Maximum = 4000;

this.trackBar1.Minimum = -4000;

this.trackBar1.Name = "trackBar1";

this.trackBar1.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar1.Size = new System.Drawing.Size(56, 492);

this.trackBar1.TabIndex = 2;

this.trackBar1.TickFrequency = 1000;

this.trackBar1.Value = 1000;

this.trackBar1.Scroll += new System.EventHandler(this.trackBar1\_Scroll);

//

// trackBar2

//

this.trackBar2.BackColor = System.Drawing.SystemColors.InactiveCaption;

this.trackBar2.Location = new System.Drawing.Point(1012, 249);

this.trackBar2.Margin = new System.Windows.Forms.Padding(4);

this.trackBar2.Maximum = 4000;

this.trackBar2.Minimum = -4000;

this.trackBar2.Name = "trackBar2";

this.trackBar2.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar2.Size = new System.Drawing.Size(56, 492);

this.trackBar2.TabIndex = 3;

this.trackBar2.TickFrequency = 1000;

this.trackBar2.Value = 1000;

this.trackBar2.Scroll += new System.EventHandler(this.trackBar2\_Scroll);

//

// trackBar3

//

this.trackBar3.BackColor = System.Drawing.SystemColors.InactiveCaption;

this.trackBar3.Location = new System.Drawing.Point(1076, 249);

this.trackBar3.Margin = new System.Windows.Forms.Padding(4);

this.trackBar3.Maximum = 4000;

this.trackBar3.Minimum = -4000;

this.trackBar3.Name = "trackBar3";

this.trackBar3.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar3.Size = new System.Drawing.Size(56, 492);

this.trackBar3.TabIndex = 4;

this.trackBar3.TickFrequency = 1000;

this.trackBar3.Value = 100;

this.trackBar3.Scroll += new System.EventHandler(this.trackBar3\_Scroll);

//

// label1

//

this.label1.AutoSize = true;

this.label1.Location = new System.Drawing.Point(965, 229);

this.label1.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label1.Name = "label1";

this.label1.Size = new System.Drawing.Size(17, 17);

this.label1.TabIndex = 5;

this.label1.Text = "X";

//

// label2

//

this.label2.AutoSize = true;

this.label2.Location = new System.Drawing.Point(1032, 229);

this.label2.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label2.Name = "label2";

this.label2.Size = new System.Drawing.Size(17, 17);

this.label2.TabIndex = 6;

this.label2.Text = "Y";

//

// label3

//

this.label3.AutoSize = true;

this.label3.Location = new System.Drawing.Point(1095, 229);

this.label3.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label3.Name = "label3";

this.label3.Size = new System.Drawing.Size(17, 17);

this.label3.TabIndex = 7;

this.label3.Text = "Z";

//

// label4

//

this.label4.AutoSize = true;

this.label4.Location = new System.Drawing.Point(957, 753);

this.label4.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label4.Name = "label4";

this.label4.Size = new System.Drawing.Size(16, 17);

this.label4.TabIndex = 8;

this.label4.Text = "0";

//

// label5

//

this.label5.AutoSize = true;

this.label5.Location = new System.Drawing.Point(1027, 753);

this.label5.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label5.Name = "label5";

this.label5.Size = new System.Drawing.Size(16, 17);

this.label5.TabIndex = 9;

this.label5.Text = "0";

//

// label6

//

this.label6.AutoSize = true;

this.label6.Location = new System.Drawing.Point(1241, 753);

this.label6.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label6.Name = "label6";

this.label6.Size = new System.Drawing.Size(16, 17);

this.label6.TabIndex = 10;

this.label6.Text = "0";

this.label6.Click += new System.EventHandler(this.label6\_Click);

//

// trackBar4

//

this.trackBar4.BackColor = System.Drawing.SystemColors.InactiveCaption;

this.trackBar4.Location = new System.Drawing.Point(1152, 249);

this.trackBar4.Margin = new System.Windows.Forms.Padding(4);

this.trackBar4.Maximum = 360;

this.trackBar4.Minimum = -360;

this.trackBar4.Name = "trackBar4";

this.trackBar4.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar4.Size = new System.Drawing.Size(56, 492);

this.trackBar4.TabIndex = 11;

this.trackBar4.TickFrequency = 10;

this.trackBar4.Value = 360;

this.trackBar4.Scroll += new System.EventHandler(this.trackBar4\_Scroll);

//

// trackBar5

//

this.trackBar5.BackColor = System.Drawing.SystemColors.InactiveCaption;

this.trackBar5.Location = new System.Drawing.Point(1216, 249);

this.trackBar5.Margin = new System.Windows.Forms.Padding(4);

this.trackBar5.Maximum = 500;

this.trackBar5.Minimum = 100;

this.trackBar5.Name = "trackBar5";

this.trackBar5.Orientation = System.Windows.Forms.Orientation.Vertical;

this.trackBar5.Size = new System.Drawing.Size(56, 492);

this.trackBar5.TabIndex = 12;

this.trackBar5.TickFrequency = 100;

this.trackBar5.Value = 500;

this.trackBar5.Scroll += new System.EventHandler(this.trackBar5\_Scroll);

//

// label7

//

this.label7.AutoSize = true;

this.label7.Location = new System.Drawing.Point(1152, 229);

this.label7.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label7.Name = "label7";

this.label7.Size = new System.Drawing.Size(44, 17);

this.label7.TabIndex = 13;

this.label7.Text = "Angle";

//

// label8

//

this.label8.AutoSize = true;

this.label8.Location = new System.Drawing.Point(1213, 229);

this.label8.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label8.Name = "label8";

this.label8.Size = new System.Drawing.Size(44, 17);

this.label8.TabIndex = 14;

this.label8.Text = "Zoom";

//

// label9

//

this.label9.AutoSize = true;

this.label9.Location = new System.Drawing.Point(965, 43);

this.label9.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label9.Name = "label9";

this.label9.Size = new System.Drawing.Size(53, 17);

this.label9.TabIndex = 15;

this.label9.Text = "По оси";

//

// comboBox1

//

this.comboBox1.BackColor = System.Drawing.SystemColors.InactiveCaption;

this.comboBox1.ForeColor = System.Drawing.SystemColors.MenuHighlight;

this.comboBox1.FormattingEnabled = true;

this.comboBox1.Items.AddRange(new object[] {

"вращать вдоль X",

"вращать вдоль Y",

"вращать вдоль Z"});

this.comboBox1.Location = new System.Drawing.Point(969, 63);

this.comboBox1.Margin = new System.Windows.Forms.Padding(4);

this.comboBox1.Name = "comboBox1";

this.comboBox1.Size = new System.Drawing.Size(237, 24);

this.comboBox1.TabIndex = 16;

this.comboBox1.SelectedIndexChanged += new System.EventHandler(this.comboBox1\_SelectedIndexChanged);

//

// menuStrip1

//

this.menuStrip1.Items.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.загрузитьМодельToolStripMenuItem});

this.menuStrip1.Location = new System.Drawing.Point(0, 0);

this.menuStrip1.Name = "menuStrip1";

this.menuStrip1.Padding = new System.Windows.Forms.Padding(8, 2, 0, 2);

this.menuStrip1.Size = new System.Drawing.Size(1289, 28);

this.menuStrip1.TabIndex = 20;

this.menuStrip1.Text = "menuStrip1";

//

// загрузитьМодельToolStripMenuItem

//

this.загрузитьМодельToolStripMenuItem.DropDownItems.AddRange(new System.Windows.Forms.ToolStripItem[] {

this.выбратьФайлДляЗагрузкиToolStripMenuItem});

this.загрузитьМодельToolStripMenuItem.Name = "загрузитьМодельToolStripMenuItem";

this.загрузитьМодельToolStripMenuItem.Size = new System.Drawing.Size(221, 24);

this.загрузитьМодельToolStripMenuItem.Text = "Загрузить модель батискафа";

//

// выбратьФайлДляЗагрузкиToolStripMenuItem

//

this.выбратьФайлДляЗагрузкиToolStripMenuItem.Name = "выбратьФайлДляЗагрузкиToolStripMenuItem";

this.выбратьФайлДляЗагрузкиToolStripMenuItem.Size = new System.Drawing.Size(269, 24);

this.выбратьФайлДляЗагрузкиToolStripMenuItem.Text = "Выбрать файл для загрузки";

this.выбратьФайлДляЗагрузкиToolStripMenuItem.Click += new System.EventHandler(this.выбратьФайлДляЗагрузкиToolStripMenuItem\_Click);

//

// openFileDialog1

//

this.openFileDialog1.FileName = "openFileDialog1";

//

// label10

//

this.label10.AutoSize = true;

this.label10.Font = new System.Drawing.Font("Microsoft Sans Serif", 45F, System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point, ((byte)(204)));

this.label10.ForeColor = System.Drawing.SystemColors.Control;

this.label10.Location = new System.Drawing.Point(146, 28);

this.label10.Margin = new System.Windows.Forms.Padding(4, 0, 4, 0);

this.label10.Name = "label10";

this.label10.Size = new System.Drawing.Size(631, 85);

this.label10.TabIndex = 21;

this.label10.Text = "ⒷⒶⓉⒽⓎⓈⒸⒶⓅⒽⒺ";

this.label10.Click += new System.EventHandler(this.label10\_Click);

//

// Form1

//

this.AutoScaleDimensions = new System.Drawing.SizeF(8F, 16F);

this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;

this.BackColor = System.Drawing.SystemColors.ActiveCaption;

this.ClientSize = new System.Drawing.Size(1289, 788);

this.Controls.Add(this.label10);

this.Controls.Add(this.comboBox1);

this.Controls.Add(this.label9);

this.Controls.Add(this.label8);

this.Controls.Add(this.label7);

this.Controls.Add(this.trackBar5);

this.Controls.Add(this.trackBar4);

this.Controls.Add(this.label6);

this.Controls.Add(this.label5);

this.Controls.Add(this.label4);

this.Controls.Add(this.label3);

this.Controls.Add(this.label2);

this.Controls.Add(this.label1);

this.Controls.Add(this.trackBar3);

this.Controls.Add(this.trackBar2);

this.Controls.Add(this.trackBar1);

this.Controls.Add(this.AnT);

this.Controls.Add(this.menuStrip1);

this.ForeColor = System.Drawing.SystemColors.Control;

this.MainMenuStrip = this.menuStrip1;

this.Margin = new System.Windows.Forms.Padding(4);

this.Name = "Form1";

this.Text = "00";

this.Load += new System.EventHandler(this.Form1\_Load);

((System.ComponentModel.ISupportInitialize)(this.trackBar1)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar3)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar4)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.trackBar5)).EndInit();

this.menuStrip1.ResumeLayout(false);

this.menuStrip1.PerformLayout();

this.ResumeLayout(false);

this.PerformLayout();

}

#endregion

private Tao.Platform.Windows.SimpleOpenGlControl AnT;

private System.Windows.Forms.Timer RenderTimer;

private System.Windows.Forms.TrackBar trackBar1;

private System.Windows.Forms.TrackBar trackBar2;

private System.Windows.Forms.TrackBar trackBar3;

private System.Windows.Forms.Label label1;

private System.Windows.Forms.Label label2;

private System.Windows.Forms.Label label3;

private System.Windows.Forms.Label label4;

private System.Windows.Forms.Label label5;

private System.Windows.Forms.Label label6;

private System.Windows.Forms.TrackBar trackBar4;

private System.Windows.Forms.TrackBar trackBar5;

private System.Windows.Forms.Label label7;

private System.Windows.Forms.Label label8;

private System.Windows.Forms.Label label9;

private System.Windows.Forms.ComboBox comboBox1;

private System.Windows.Forms.MenuStrip menuStrip1;

private System.Windows.Forms.ToolStripMenuItem загрузитьМодельToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem выбратьФайлДляЗагрузкиToolStripMenuItem;

private System.Windows.Forms.OpenFileDialog openFileDialog1;

private System.Windows.Forms.Label label10;

}

}

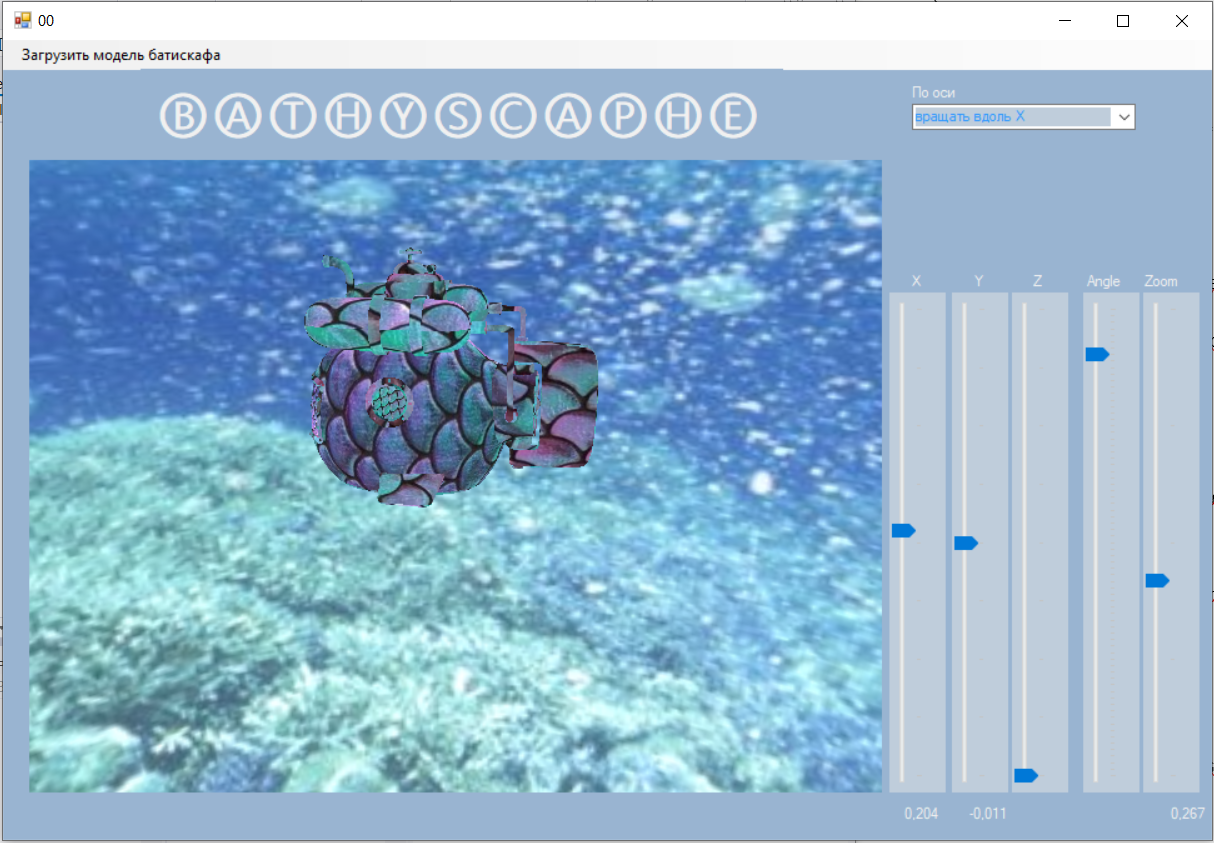
**ПРИЛОЖЕНИЕ Б.**

Рисунок 5 – Отображение батискафа в OpenGl





Рисунок 6 – Прототипы модели