**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное бюджетное

образовательное учреждение высшего образования

**«Владимирский государственный университет имени**

**Александра Григорьевича и Николая Григорьевича Столетовых»**

Кафедра информационных систем и программной инженерии

***КУРСОВОЙ ПРОЕКТ***

*по дисциплине: «Программирование компьютерной графики»*

Разработка графического приложения

на C# с использованием OpenGL

Предметная область:

«Питомник»

*Выполнил:*

ст. гр. ПРИ-117

Емельянов Д.В.

*Принял:*

Жигалов И. Е.

Владимир, 2021

**Аннотация**

Курсовой проект выполнен по дисциплине «Программирование компьютерной графики». В данном курсовом проекте было разработано графическое приложения по предметной области «Питомник», которое формирует заданную двухмерную и трехмерную графическую сцену с интерактивным управлением. При разработке были использованы возможности языка программирования C# и его фреймворка OpenGL по созданию графических приложений, формированию простых графических объектов из примитивов, применению геометрических преобразований, использованию цветов и текстур, анимации сцены и управлению сценой.

В ходе выполнения данного курсового проекта были выполнены следующие задачи:

1. Анализ предметной области;
2. Формирование технического задания;
3. Разработка интерфейса приложения;
4. Разработка 3D моделей животных;
5. Реализация приложения с использованием заданных технологий.

Пояснительная записка содержит 34 листа. Включает в себя: 10 рисунков, 2 приложения, 1 таблица. Количество использованных источников – 7.

The course project was carried out in the discipline "Computer Graphics Programming". In this project, a graphic application was developed for the "Nursery" subject area, which forms a given two-dimensional and three-dimensional graphic scene with interactive control. During the development, the capabilities of the C # programming language and its OpenGL framework were used to create graphical applications, the formation of simple graphical objects from primitives, geometric transformations, the use of colors and textures, scene animation and scene control.

In the course of this course project, the following tasks were completed:

1. Domain analysis;
2. Formation of technical specifications;
3. Application interface development;
4. Development of 3D models of animals;
5. Implementation of the application using the specified technologies.

The explanatory note contains 34 sheets. Includes: 10 figures, 2 applications, 1 table. The number of sources used - 7.

Содержание

[ВВЕДЕНИЕ 6](#_Toc68344491)

[1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ 7](#_Toc68344492)

[1.1. Введение 7](#_Toc68344493)

[1.2. Описание 7](#_Toc68344494)

[1.3. Цель разработки 8](#_Toc68344495)

[1.4. Функциональные требования 8](#_Toc68344496)

[1.5. Нефункциональные требования 9](#_Toc68344497)

[1.6. Инструменты разработки 9](#_Toc68344498)

[1.7. Интерфейс приложения 10](#_Toc68344499)

[1.8. Требования к программной документации 11](#_Toc68344500)

[1.9. Стадии и этапы разработки 11](#_Toc68344501)

[1.10. Порядок контроля и приемки 12](#_Toc68344502)

[2 ЦЕЛИ И ЗАДАЧИ ПРОЕКТА 13](#_Toc68344503)

[3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 14](#_Toc68344504)

[3.1. Моделируемая сцена 14](#_Toc68344505)

[3.2. Моделируемые объекты 14](#_Toc68344506)

[3.3. Моделируемые события 14](#_Toc68344507)

[4 ПРОЕКТИРОВАНИЕ 17](#_Toc68344508)

[4.1. Описание архитектуры 17](#_Toc68344509)

[4.2. Описание технологий 17](#_Toc68344510)

[4.2. Описание реализации 18](#_Toc68344511)

[4.2. Модель данных 18](#_Toc68344512)

[5 РАЗРАБОТКА АЛГОРИТМА И ДИЗАЙНА ПРИЛОЖЕНИЯ 20](#_Toc68344513)

[5.1. Разработка схемы алгоритма приложения 20](#_Toc68344514)

[5.2. Разработка дизайна приложения 20](#_Toc68344515)

[6 МОДЕЛИРОВАНИЕ ГРАФИЧЕСКИХ ОБЪЕКТОВ 22](#_Toc68344516)

[6.1. Выбор и обоснование моделей 22](#_Toc68344517)

[6.2. Создание объектов с помощью графических редакторов 22](#_Toc68344518)

[7 МОДЕЛИРОВАНИЕ СЦЕНЫ 25](#_Toc68344519)

[7.1. Использование материалов и текстур 25](#_Toc68344520)

[7.2. Создание анимации и визуальных эффектов 26](#_Toc68344521)

[8.1. Логика работы приложения 27](#_Toc68344522)

[9 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ И УПРАВЛЕНИЕ ПРИЛОЖЕНИЕМ 28](#_Toc68344523)

[9.1. Обоснование и реализация дизайна интерфейса пользователя 28](#_Toc68344524)

[9.2. Проектирование интерактивного управления сценой 28](#_Toc68344525)

[10 ПРОГРАММИРОВАНИЕ ПРИЛОЖЕНИЯ 29](#_Toc68344526)

[10.1. Отладка и оптимизация кода 29](#_Toc68344527)

[10.2. Тестирования приложения 29](#_Toc68344528)

[10.3. Анализ программного продукта 31](#_Toc68344529)

[11 РАЗРАБОТКА ПРОГРАММНОЙ ДОКУМЕНТАЦИИ 32](#_Toc68344530)

[12 КОМАНДНАЯ РАЗРАБОТКА ПРОЕКТА 33](#_Toc68344531)

[12.1. Распределение зон ответственности 33](#_Toc68344532)

[ЗАКЛЮЧЕНИЕ 34](#_Toc68344533)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 35](#_Toc68344534)

[ПРИЛОЖЕНИЕ А 36](#_Toc68344535)

[ПРИЛОЖЕНИЕ Б 56](#_Toc68344536)

ВВЕДЕНИЕ

Курсовой проект выполнен по дисциплине «Программирование компьютерной графики». Темой курсового проекта является «Разработка графического приложения на C# с использованием OpenGL». Предметной областью является «Питомник».

Курсовой проект был выполнен в команде, для достижения более эффективной работы были распределены зоны ответственности.

Предметом курсового проекта является разработка программы, формирующей данную графическую сцену с интерактивным управлением ею. Для этого используются возможности C# и OpenGL по созданию графических приложений, формированию простых графических объектов из примитивов, применению геометрических преобразований, использованию цветов, анимации сцены и управлению сценой.

1 ТЕХНИЧЕСКОЕ ЗАДАНИЕ

1.1. Введение

Наименование приложения: Питомник (Zoo) – программа моделирования процессов жизнедеятельности обитателей питомника.

Глоссарий:

Питомник (Zoo) - место или заведение для выращивания и разведения животных.

Животные (Animals) – обитатели питомника.

Загон (Corral) - место, огороженное изгородью, где располагаются животные.

Рыба (Fish) - животное, которое обитает в воде.

Овца (Sheep) - животное, которое обитает в загоне.

Петух (Cock) - животное, которое обитает в загоне.

Баран (Battle sheep) – животное, которое обитает в загоне.

Аквариум (Aquarium) - место с морским фоном, где располагаются рыбы.

1.2. Описание

Назначение и цель разработки: Разрабатываемое графическое приложение моделирует процессы жизнедеятельности животных в питомнике: движение овец, движение рыб, а также животных можно полить водой.

Краткая характеристика и область применения: Приложение разрабатывается с использованием технологии Windows Forms. В данном приложении представлены процессы жизнедеятельности животных в питомнике. Приложение разрабатывается в учебных целях.

Сценарий работы приложения представлен в виде схемы на рисунке 1.

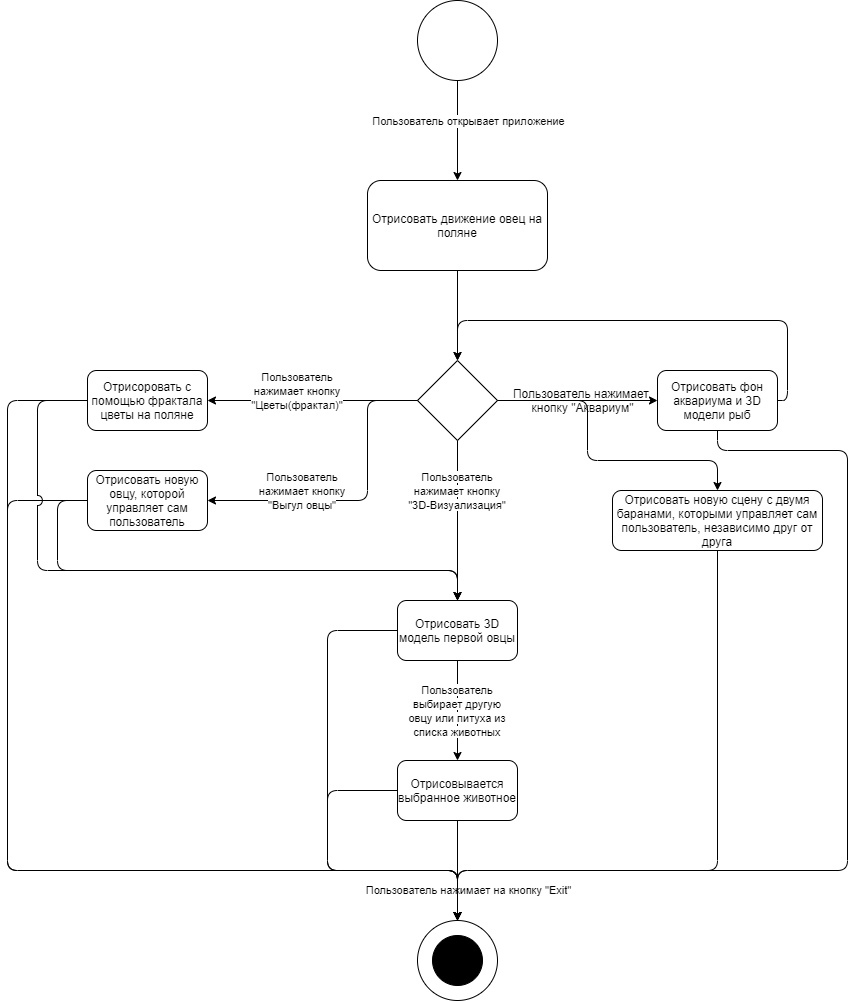


Рисунок 1 – Сценарий работы приложения

1.3. Цель разработки

Функциональным назначением программы является наглядное представление обитателей питомника.

1.4. Функциональные требования

Программа должна обеспечивать возможность выполнения перечисленных ниже функций:

* визуализация движения животных;
* управление животным с помощью клавиш клавиатуры;
* визуализация фона питомника фона питомника в виде фрактала;
* визуализация фона в виде картинки\спрайта;
* визуализация животных в 3D;
* изменение масштаба при просмотре визуализации животных в 3D.

1.5. Нефункциональные требования

Требования к реализации: исходные коды программы должны быть реализованы на языке C#. В качестве интегрированной среды разработки программы должна быть использована среда Microsoft Visual Studio 2019. Использование библиотеки OpenGL.

Требования к производительности: визуализация сцен в приложении должна происходить без зависаний.

Требования к интерфейсу: пользовательский интерфейс должен быть доступным и понятным.

Требования к установке: разрабатываемое приложение не должно требовать установки на компьютер, достаточно запустить .exe файл.

Требования к составу и параметрам технических средств: в состав технических средств должен входить персональный компьютер.

Требования к информационной и программной совместимости: системные программные средства, используемые программой, должны быть представлены операционной системы Windows.

1.6. Инструменты разработки

Visual Studio 2019 (C#) – интегрированная среда разработки программного обеспечения. Данный продукт позволяет разрабатывать Windows Forms приложения и включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения.

OpenGL — это спецификация, включающая в себя несколько сотен функций. Она определяет независимый от языка программирования кросс платформенный программный интерфейс, с помощью которого программист может создавать приложения, использующие двухмерную и трехмерную компьютерную графику.

Autodesk 3dsMax (ранее 3DStudio MAX) — профессиональное программное обеспечение для 3D-моделирования, анимации и визуализации при создании игр и проектировании. 3ds Max располагает обширными средствами для создания разнообразных по форме и сложности трёхмерных компьютерных моделей, реальных или фантастических объектов окружающего мира, с использованием разнообразных техник и механизмов.

1.7. Интерфейс приложения

Основные элементы интерфейса для пользовательского взаимодействия с программой — это кнопки, выпадающий список и ползунки windows. Кнопка «Фрактал (цветы)» обрисовывает фрактал. Кнопка «Выгул овцы» позволяет создать овцу, которую можно управлять с помощью клавиш WASD. Кнопка «3D-визуализация» позволяет рассмотреть 3D-объекты. В этом помогает список с выбором скота. Кнопка «Полить водой (Нажать на скот)» позволяет увидеть демонстрацию работы частиц. Список и «Полить водой» доступны только при визуализации 3D объектов. Кнопка «Схватка баранов» позволяет взять управление за двух овец, которые могут столкнуться.

На рисунке 2 представлен интерфейс программы при запуске приложения.

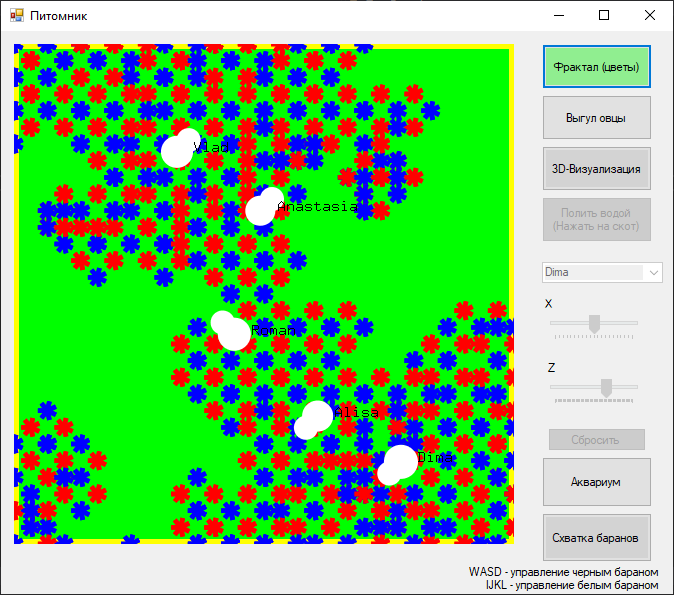


Рисунок 2 – Интерфейс программы

1.8. Требования к программной документации

Состав программной документации должен включать в себя:

1) Назначение программы;

2) Условия выполнения программы;

3) Выполнение программы.

1.9. Стадии и этапы разработки

Разработка должна быть проведена в четыре стадии:

1) разработка технического задания;

2) моделирование необходимых объектов 3D сцен;

3) разработка приложения;

4) тестирование;

На стадии разработки технического задания должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания с преподавателем.

На стадии моделирования должны быть созданы объекты для 3D сцен.

На стадии разработки приложения должен быть написан программный код и документация к нем.

На стадии тестирование должно быть выполнено испытания программы.

1.10. Порядок контроля и приемки

Критерии приемки:

Визуализация сцен в приложении должна происходить быстро (меньше 1 секунды), без зависаний.

Переключение сцен и отрисовка новой сцены должны занимать менее 1 секунды.

Преобразования объектов должно быть плавным, без зависаний.

# 2 ЦЕЛИ И ЗАДАЧИ ПРОЕКТА

Основная цель выполнения курсового проекта по дисциплине "Программирование компьютерной графики" состоит в освоении средств разработки приложений в среде Microsoft Visual Studio C# с использованием графической библиотеки OpenGL.

Основной задачей проекта является разработка графического приложения по предметной области «Питомник», которое формирует заданную двухмерную и трехмерную графическую сцену с интерактивным управлением. При разработке должны использоваться возможности C# и OpenGL по созданию графических приложений, формированию простых графических объектов из примитивов, применению геометрических преобразований, использованию цветов и текстур, анимации сцены и управлению сценой.

3 ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Во время выполнения курсового проекта было разработана сцена:

1. Двухмерная сцена с прогулкой овец, выгулом овцы и отрисовкой фрактала.

3.1. Моделируемая сцена

Двухмерная сцена включает в себя простые объекты: точки (голова и тело). Цветы выполнены с помощью фрактала “Кривая дракона”.

3.2. Моделируемые объекты

Был смоделирован следующий объект: петух. Создан с помощью сложного объекта Scatter в 3Ds Max и импортированы в проект.

3.3. Моделируемые события

Все моделируемые события представлены в таблице 1.

Таблица 1 - Моделируемые события

|  |  |  |
| --- | --- | --- |
| Событие | Предусловие | Описание |
| Пользователь запустил приложение | Первый запуск приложения | Отрисовалась сцена с полем, забором. Отрисовываются пять овец. Каждая овца ходит по кругу. |
| Пользователь нажал кнопку “Фрактал (цветы)” | Приложение запущено | Кнопка “Фрактал (цветы)” обведена синей рамкой, цвет кнопки стал зеленым. В окне визуализирован фрактал. |
| Пользователь нажал “Выгул овцы” | Приложение запущено | Кнопка “Выгул овцы” обведена синей рамкой, цвет кнопки стал зеленым. В окне визуализации появилась черная точка (овца). Пользователь может управлять |
| Пользователь нажал “3D-Визуализация” | Приложение запущено | Кнопка “3D-Визуализация” обведена синей рамкой, цвет кнопки стал зеленым. В окне визуализации отрисовывается новая сцена с животным. |
| Поменять скот в списке | Список доступен (пользователь нажал кнопку “3D-визуализация”) | Отрисовывается выбранный скот. Скот крутится. |
| Пользователь нажал “Полить водой (Нажать на скот)” | Была нажата кнопка “3D-Визуализация” | Кнопка “Полить водой (нажать на скот)” обведена синей рамкой, цвет кнопки стал зеленым. Пользователь может нажать на животного в окне визуализации и полить его водой (используется система частиц) |

4 ПРОЕКТИРОВАНИЕ

4.1. Описание архитектуры

Разработанное приложение представляет собой событийно-ориентированное приложение, написанное под платформу Microsoft .NET Framework. Большая часть времени тратится на ожидание от пользователя каких-либо действий, например, нажатие кнопки на мыши. Главный класс приложения, который обрабатывает все события формы - Form1.

Функции OpenGL реализованы в модели клиент-сервер. Приложение выступает в роли клиента – оно вырабатывает команды, а сервер OpenGL интерпретирует и выполняет их.

OpenGL является прослойкой между аппаратурой и пользовательским уровнем, что позволяет предоставлять единый интерфейс на разных платформах, используя возможности аппаратной поддержки.

4.2. Описание технологий

Приложение было реализовано на языке программирования C#, с помощью библиотеки Tao Framework.

C# — объектно-ориентированный язык программирования. Данный язык относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Tao Framework – это библиотека, предоставляющая разработчикам .NET и Mono доступ к возможностям популярных библиотек вроде OpenGL и SDL. Она изначально была создана программистом на C# и OpenGL Randy Ridge, и с тех пор значительно расширена другими разработчиками.

Autodesk 3dsMax (ранее 3DStudio MAX) — профессиональное программное обеспечение для 3D-моделирования, анимации и визуализации при создании игр и проектировании. 3ds Max располагает обширными средствами для создания разнообразных по форме и сложности трёхмерных компьютерных моделей, реальных или фантастических объектов окружающего мира, с использованием разнообразных техник и механизмов

4.2. Описание реализации

Чтобы использовать библиотеку Tao Framework, нужно проделать следующее:

Установить .NET Framework;

Установить последнюю версию Tao Framework;

Создать новый проект в Microsoft Visual Studio;

Добавить библиотеки в проект:

* Tao.OpenGL.dll - реализация библиотеки OpenGL;
* Tao.FreeGlut.dll - реализация функций библиотеки Glut;
* Tao.Platform.Windows.dll - поддержка элементов для визуализации на платформе Windows;
* Tao.Devll.dll – отвечает за загрузку текстур;
* DevIL.dll - портативная библиотека, отвечающая за подгрузку текстур;
* freeglut.dll - портативная библиотека, отвечающая за создание контекста OpenGL.

4.2. Модель данных

Данными в приложении являются экземпляры классов с их состоянием, диаграмма классов представлена на рисунке 3.



Рисунок 3 – Диаграмма классов

5 РАЗРАБОТКА АЛГОРИТМА И ДИЗАЙНА ПРИЛОЖЕНИЯ

5.1. Разработка схемы алгоритма приложения

Работа приложения начинается с инициализации библиотек и настроек программы и ее интерфейса. Всё это необходимо совершать в методе Form1\_Load класса Form1. В конце этого метода вызывается метод Start() на переменной RenderTimer, который является бесконечным циклом тиков. Каждый тик происходит отрисовка сцены в методе Draw класса Form1 и ему подобные, используя остальные классы приложения. В этом методе очищается сцена и каждый раз рисуется новая, в зависимости от состояния условий, выставленных интерфейсом.

5.2. Разработка дизайна приложения

Для разработки дизайна приложения были выбраны стандартные цвета Windows Forms. Кнопки серые или зеленые, если активированы, с более темной обводкой, а если не активны, то становятся ощутимо темнее стандартного цвета. Ползунки изменения масштаба, также выполненные в стандартном стиле. Скриншот интерфейса представлен на рисунке 4.

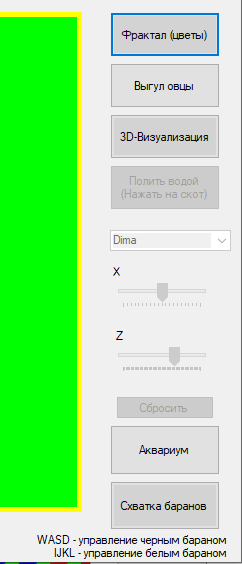


Рисунок 4 – Скриншот интерфейса программы

6 МОДЕЛИРОВАНИЕ ГРАФИЧЕСКИХ ОБЪЕКТОВ

6.1. Выбор и обоснование моделей

В качестве моделей для 3D визуализации были выбраны овцы и петух, потому что они являются главными обитателями нашего питомника. Аквариум — это главное украшение нашего заведения, а рыбы главные жители этого стеклянного дома.

6.2. Создание объектов с помощью графических редакторов

Модели овец и петуха были разработаны в программе 3Ds Max. Autodesk 3ds Max — профессиональное программное обеспечение для 3D-моделирования, анимации и визуализации при создании игр и проектировании. В настоящее время разрабатывается и издается компанией Autodesk.

Все модели имеют текстуры. Данные модели созданы из примитивов: сфера, конус и линии.

Готовые модели овец и петуха, выполненные в 3ds Max представлены на рисунках 5-6. Импортированные модели, визуализированные средствами OpenGL, представлены на рисунках 7-8.



Рисунок 5 – Разработанная модель овцы в 3DS Max

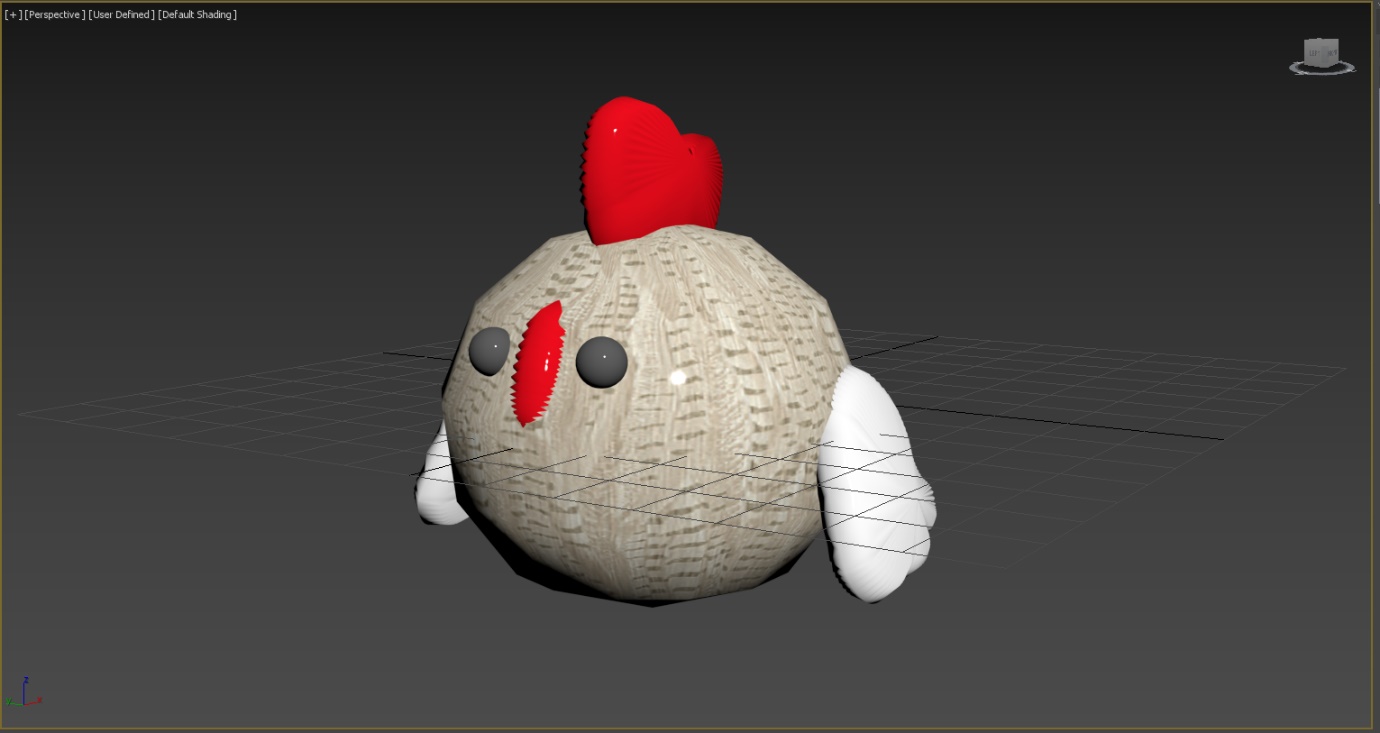


Рисунок 6 – Разработанная модель петуха в 3DS Max

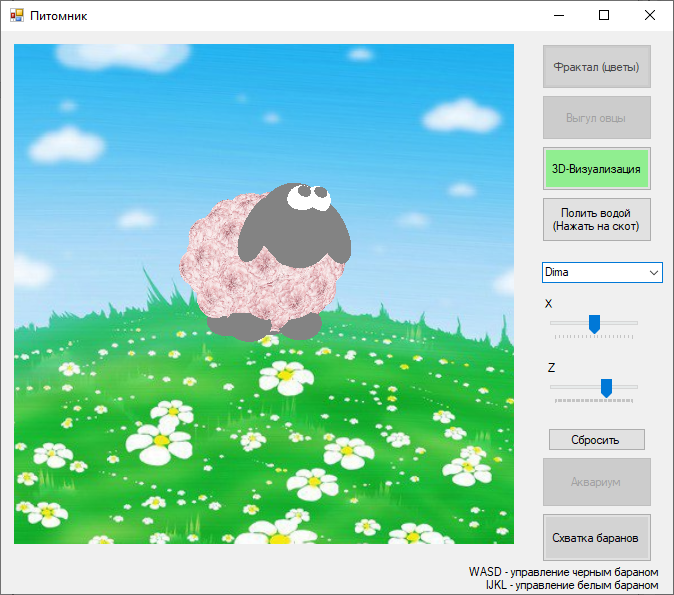


Рисунок 7 – Импортированная модель овцы

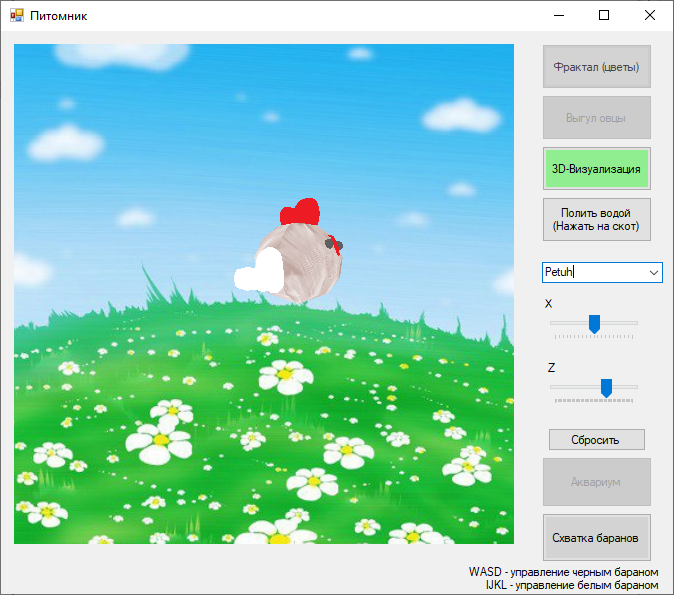


Рисунок 8 – Импортированная модель петуха

7 МОДЕЛИРОВАНИЕ СЦЕНЫ

7.1. Использование материалов и текстур

Текстура — это массив данных, состоящий из последовательности байтов, которые хранят в себе цвета. Значения текстуры часто называются тэкселями. Сложным текстурирование делает то, что прямоугольная текстура может быть наложена на непрямоугольный объект.

Для работы с текстурами была использована библиотека Devil. С помощью данной библиотеки можно быстро, без реализации собственного объемного кода, загрузить данные графического файла.

Чтобы использовать наложение текстуры, необходимо выполнить следующие шаги:

1. Создать текстурный объект и задать текстуру для него;

2. Задать как текстура должна воздействовать на каждый пиксель;

3. Активизировать механизм текстурирования;

4. Нарисовать сцену, передавая на конвейер визуализации геометрические координаты и координаты текстуры.

В курсовом проекте был реализован класс для работы с текстурами – LoadTexture. Полный код данного класса представлен в приложении А.

Для загрузки текстуры используется метод LoadTextureForModel(), чтобы создать текстуру в памяти OpenGL используется метод MakeGlTexture().

В данном курсовом проекте в качестве материалов были использованы изображения (спрайты), которые использовались в качестве фона сцены. Загрузка спрайтов происходит точно также, как и текстуры только для его отрисовки необходимо сделать следующие:

1. Загрузить спрайт
2. Поместить спрайт в качестве текстуры объекта
3. Отрисовать полигон, расположив его должным образом

7.2. Создание анимации и визуальных эффектов

Для создания анимации был добавлен объект “Timer”, с названием “RenderTimer” и интервалом в 30 миллисекунд. Этот таймер каждый интервал вызывает метод отрисовки объектов, благодаря этому и работает анимация.

Список реализованных анимаций:

* Движение овец на поляне

Каждый раз при срабатывании таймера изменяется положения овец.

* Вращение головы овцы

Каждый раз при срабатывании таймера определяется положение головы овцы и зависит оно от направления движения овцы.

8 МОДЕЛИРОВАНИЕ СОБЫТИЙ

8.1. Логика работы приложения

Основная логика приложения завязана на отрисовке компонентов каждый раз при срабатывании таймера с интервалом в 30 миллисекунд. При срабатывании таймера вызывается метод отрисовки в котором прописана вся основная логика отрисовки объектов и сцен.

9 ИНТЕРФЕЙС ПОЛЬЗОВАТЕЛЯ И УПРАВЛЕНИЕ ПРИЛОЖЕНИЕМ

9.1. Обоснование и реализация дизайна интерфейса пользователя

Для облегчения использования приложения пользователем, реализовано управление одной овцой (при нажатии на кнопку “Выгул овцы”) с использованием стандартной комбинации клавиш для движения WASD.

9.2. Проектирование интерактивного управления сценой

Интерактивное управление сценой реализовано с помощью движения камеры. Камеру можно двигать только в сцене 3D-Визуализации животных и только по осям X и Z.

10 ПРОГРАММИРОВАНИЕ ПРИЛОЖЕНИЯ

10.1. Отладка и оптимизация кода

Во время разработки приложения производилась отладка кода для отслеживания состояния объектов. Для улучшения производительности кода были добавлены условия для отрисовки определенных объектов сцены, чтобы не делать лишнюю работу, в зависимости от выбранной пользователем сцены.

10.2. Тестирования приложения

Для данного приложения проводилось мануальное тестирование с минимальными тест-кейсами. Описание тестирование и результаты тестов представлены ниже.

1. Название: проверить фрактал.

Предусловие: Приложение запущено.

Выполняемые шаги: нажать на кнопку “Фрактал”

Ожидаемый результат: Кнопка “Фрактал (цветы)” обведен синей рамкой, сама кнопка зеленого цвета. В окне визуализации проявился фрактал.

Фактический результат: соответствует ожидаемому.

1. Название: проверить выгул овцы (появление овцы)

Предусловие: Приложение запущено

Выполняемые шаги: нажать кнопку “Выгул овцы”.

Ожидаемый результат: Кнопка “Выгул овцы” обведен синей рамкой, сама кнопка зеленого цвета. В окне визуализации появилась новая овца.

Фактический результат: соответствует ожидаемому.

1. Название: проверить выгул овцы (движение овцы)

Предусловие: Приложение запущено, нажата кнопка “Выгул овцы”.

Выполняемые шаги: двигать овцу на клавиши wasd.

Ожидаемый результат: Голова овцы появляется в зависимости от стороны (клавиши), которая была нажата. Движение овцы проявляется в зависимости от стороны (клавиши), которая была нажата.

Фактический результат: соответствует ожидаемому.

1. Название: проверить смену модели.

Предусловие: Приложение запущено, нажата кнопка “3D-визуализация”.

Выполняемые шаги: выбрать из списка Dima.

Ожидаемый результат: Отрисовывается 3D-модель овцы. Овечка вращается.

Фактический результат: соответствует ожидаемому.

1. Название: Проверка работы частиц.

Предусловие: Приложение запущено, нажата кнопка “3D-визуализация”.

Выполняемые шаги: нажать кнопку “Полить водой (нажать на скот). Нажать в окно визуализации на модель животного.

Ожидаемый результат: Частицы льются на модель животного.

Фактический результат: соответствует ожидаемому.

1. Название: Смена положения модели.

Предусловие: Приложение запущено, нажата кнопка “3D-визуализация”.

Выполняемые шаги: изменить положение ползунка по осям X и Z.

Ожидаемый результат: Положение овцы сдвигается по осям.

Фактический результат: соответствует ожидаемому.

1. Название: проверить работу аквариума.

Предусловие: Приложение запущено.

Выполняемые шаги: нажать кнопку “Аквариум”.

Ожидаемый результат: в окне визуализации обрисовывается сцена аквариума. Рыбы из примитивов плывут.

Фактический результат: соответствует ожидаемому.

10.3. Анализ программного продукта

По результатам тестирования критичных багов не обнаружено, следовательно, программный продукт уже может использоваться по назначению.

11 РАЗРАБОТКА ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Для запуска приложения необходимо запустить файл Zoo.exe, который находится по пути /bin/Debug/Zoo.exe.

12 КОМАНДНАЯ РАЗРАБОТКА ПРОЕКТА

12.1. Распределение зон ответственности

В процессе выполнения курсового проекта были определены сильные стороны каждого из членов команды для достижения максимально эффективного процесса разработки.

Диаграмма Ганта представлена рисунках 9-10.



Рисунок 9 – Диаграмма Ганта



Рисунок 10 – Диаграмма Ганта

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта было разработано графическое приложение на тему «Питомник», который демонстрирует процессы жизнедеятельности животных в питомнике. Данная программа разрабатывается в виде приложения Windows Forms. Приложение разработано в учебных целях.

В ходе выполнения курсового проекта были улучшены навыки командной работы.

В процессе разработки были освоены новые инструменты разработки, дополнен стек изученных технологий, повышено владение языком программирования C#, в частности его фреймворком OpenGL.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Жигалов И.Е. Компьютерная графика: Курс лекций/ Владим. гос. ун-т. Владимир, 2004. 124 с. ISBN 5-89368-459-1.
2. Жигалов И.Е., Новиков И.А. Программирование двухмерной компьютерной графики. Учебное пособие./ Владим. гос. ун-т. Владимир, 2015. 120 с. ISBN 978-5-9984-0610-2.
3. Жигалов И. Е., Новиков И.А. Программирование трехмерной компьютерной графики : учеб. пособие / Владим. гос. ун-т им. А. Г. и Н. Г. Столетовых. ‒Владимир: Изд-воВлГУ, 2016. ‒92 с. ISBN 978-5-9984-0685-0.
4. Жигалов И.Е., Новиков И.А. Программирование компьютерной графики. Учебное пособие./ Владим. гос. ун-т. Владимир, 2014. 96 с. ISBN 978-5-9984- 0437-5.
5. Тарасов И.А. Основы программирования в OрenGL. - М.: Телеком, 2000.- 188
6. Тихомиров Ю. Программирование трехмерной графики.- СПб. BHV, 1998. - 320 с.
7. Порев В.Н. Компьютерная графика.–СПб.: БХВ-Петербург, 2002.- 432 с.

ПРИЛОЖЕНИЕ А

using System;

using System.Drawing;

using System.IO;

using System.Windows.Forms;

using Tao.DevIl;

using Tao.FreeGlut;

using Tao.OpenGl;

namespace Zoo

{

public partial class Form1 : Form

{

Sheeps sheeps = new Sheeps();

Flowers grass = new Flowers();

SheepByKeyboard sheepByKeyboard = new SheepByKeyboard();

anModelLoader sheep = null;

Cursor cursor = null;

public Form1()

{

InitializeComponent();

AnT.InitializeContexts();

}

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

{

Glut.glutInit();

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

Il.ilInit();

Gl.glClearColor(0, 120, 0, 1);

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

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

RenderTimer.Start();

comboBox1.Enabled = false;

xBar.Enabled = false;

trackBar1.Enabled = false;

button1.Enabled = false;

Touch\_btn.Enabled = false;

comboBox1.SelectedIndex = 0;

}

double a = 0, b = 5, c = -20, d = -360,

xSpinFirstFish = -20;

double aR = 0, bR = 0, cR = -0.7;

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

Explosion explosion = new Explosion(0, 0, 0, 300, 500);

// отсчет времени

float global\_time = 0;

int mouseMoveX;

int mouseMoveY;

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

{

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

if (!\_is3dShow && !\_isAquariumShow)

{

DrawFence();

if (\_isfractalNeeede)

{

grass.DrawGrass();

}

if (\_isSheepAlive)

{

sheepByKeyboard.DrawSheep();

}

Gl.glLoadIdentity();

if (sheeps.DimaPointPossition == sheeps.DimaCount - 1)

{

sheeps.DimaPointPossition = 0;

}

if (sheeps.VladPointPossition == sheeps.VladCount - 1)

{

sheeps.VladPointPossition = 0;

}

if (sheeps.AlisaPointPossition == sheeps.AlisaCount - 1)

{

sheeps.AlisaPointPossition = 0;

}

if (sheeps.AnastasiaPointPossition == sheeps.AnastasiaCount - 1)

{

sheeps.AnastasiaPointPossition = 0;

}

if (sheeps.RomanPointPossition == sheeps.RomanCount - 1)

{

sheeps.RomanPointPossition = 0;

}

sheeps.DrawSheeps();

AnT.Invalidate();

sheeps.DimaPointPossition++;

sheeps.VladPointPossition++;

sheeps.AlisaPointPossition++;

sheeps.AnastasiaPointPossition++;

sheeps.RomanPointPossition++;

} else if (\_is3dShow)

{

Gl.glClearColor(0, 255, 0, 1);

Gl.glLoadIdentity();

var loadTextures = new LoadTextures();

loadTextures.LoadTextureForModel("background");

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

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, loadTextures.GetTextureObj());

// сохраняем состояние матрицы

Gl.glPushMatrix();

Gl.glTranslated(0, 0, -55);

Gl.glRotated(90, 0, 0, 1);

Gl.glClear(Gl.GL\_4D\_COLOR\_TEXTURE);

// отрисовываем полигон

Gl.glBegin(Gl.GL\_QUADS);

Gl.glVertex3d(25, 25, 0);

Gl.glTexCoord2f(0, 0);

Gl.glVertex3d(25, -25, 0);

Gl.glTexCoord2f(0, 1);

Gl.glVertex3d(-25, -25, 0);

Gl.glTexCoord2f(1, 1);

Gl.glVertex3d(-25, 25, 0);

Gl.glTexCoord2f(1, 0);

// завершаем отрисовку

Gl.glEnd();

// возвращаем матрицу

Gl.glPopMatrix();

Gl.glClearColor(0, 255, 0, 1);

Gl.glTranslated(a, b, c);

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

if (d == 360) d = -360; else d++;

if (\_isTouchable)

{

// отсчитываем время

global\_time += (float)RenderTimer.Interval / 1000;

// выполняем просчет взрыва

explosion.Calculate(global\_time);

//cursor?.DrawCursor(AnT, mouseMoveX, mouseMoveY);

}

if (sheep != null)

sheep.DrawModel();

Gl.glFlush();

// отключаем режим текстурирования

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

}

else if (\_isAquariumShow)

{

DrawAquarium();

}

AnT.Invalidate();

}

private void DrawAquarium()

{

Gl.glClearColor(0, 255, 0, 1);

Gl.glLoadIdentity();

var loadTextures = new LoadTextures();

loadTextures.LoadTextureForModel("aquarium");

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

Gl.glBindTexture(Gl.GL\_TEXTURE\_2D, loadTextures.GetTextureObj());

// сохраняем состояние матрицы

Gl.glPushMatrix();

Gl.glTranslated(0, 0, -55);

Gl.glRotated(90, 0, 0, 1);

Gl.glClear(Gl.GL\_4D\_COLOR\_TEXTURE);

// отрисовываем полигон

Gl.glBegin(Gl.GL\_QUADS);

Gl.glVertex3d(25, 25, 0);

Gl.glTexCoord2f(0, 0);

Gl.glVertex3d(25, -25, 0);

Gl.glTexCoord2f(0, 1);

Gl.glVertex3d(-25, -25, 0);

Gl.glTexCoord2f(1, 1);

Gl.glVertex3d(-25, 25, 0);

Gl.glTexCoord2f(1, 0);

// завершаем отрисовку

Gl.glEnd();

// возвращаем матрицу

Gl.glPopMatrix();

Gl.glFlush();

// отключаем режим текстурирования

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

Gl.glTranslated(xSpinFirstFish, b, c);

if (xSpinFirstFish > 25) xSpinFirstFish = -25; else xSpinFirstFish += 0.3;

Gl.glPushMatrix();

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

Gl.glColor3f(1.0f, 1.0f, 0.0f);

// Первая рыба

Gl.glRotated(90, 0, 1, 0);

Glut.glutSolidCone(2, 5, 10, 10);

Glut.glutSolidSphere(3, 30, 30);

Gl.glTranslated(0, 0, -5);

Glut.glutSolidCone(3, 5, 10, 10);

Gl.glTranslated(0, 0, 5);

// Вторая рыба

Gl.glScalef(0.5f, 0.5f, 0.5f);

Gl.glColor3f(1.0f, 0.0f, 0.0f);

Gl.glTranslated(0, -10, 0);

Gl.glRotated(270, 0, 1, 0);

Gl.glRotated(90, 0, 1, 0);

Glut.glutSolidCone(2, 5, 10, 10);

Glut.glutSolidSphere(3, 30, 30);

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

Gl.glTranslated(0, 0, -5);

Glut.glutSolidCone(3, 5, 10, 10);

Gl.glScalef(2f, 2f, 2f);

Gl.glTranslated(0, 0, 5);

// Третья рыба

Gl.glScalef(0.6f, 0.6f, 0.6f);

Gl.glColor3f(1.0f, 0.0f, 1.0f);

Gl.glTranslated(0, -6, 0);

Gl.glRotated(270, 0, 1, 0);

Gl.glRotated(90, 0, 1, 0);

Glut.glutSolidCone(2, 5, 10, 10);

Glut.glutSolidSphere(3, 30, 30);

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

Gl.glTranslated(0, 0, -5);

Glut.glutSolidCone(3, 5, 10, 10);

Gl.glScalef(2f, 2f, 2f);

Gl.glTranslated(0, 0, 5);

// Четвертая рыба

Gl.glScalef(0.8f, 0.8f, 0.8f);

Gl.glColor3f(1.0f, 1.0f, 1.0f);

Gl.glTranslated(0, -5, 0);

Gl.glRotated(270, 0, 1, 0);

Gl.glRotated(90, 0, 1, 0);

Glut.glutSolidCone(2, 5, 10, 10);

Glut.glutSolidSphere(3, 30, 30);

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

Gl.glTranslated(0, 0, -5);

Glut.glutSolidCone(3, 5, 10, 10);

Gl.glPopMatrix();

}

private void DrawFence()

{

Gl.glColor3f(255, 255, 0);

Gl.glLineWidth(25);

Gl.glBegin(Gl.GL\_LINES);

Gl.glVertex2d(-15, -15);

Gl.glVertex2d(15, -15);

Gl.glVertex2d(15, -15);

Gl.glVertex2d(15, 15);

Gl.glVertex2d(15, 15);

Gl.glVertex2d(-15, 15);

Gl.glVertex2d(-15, 15);

Gl.glVertex2d(-15, -15);

Gl.glEnd();

}

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

{

c = trackBar1.Value;

}

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

{

a = xBar.Value;

}

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

{

a = 0; c = -20;

trackBar1.Value = (int)c;

xBar.Value = (int)a;

}

private bool \_isTouchable = false;

private void Touch\_btn\_Click(object sender, EventArgs e)

{

if (cursor is null)

{

cursor = new Cursor();

}

\_isTouchable = !\_isTouchable;

if (\_isTouchable)

{

Touch\_btn.BackColor = Color.LightGreen;

}

else

{

Touch\_btn.BackColor = Color.LightGray;

}

}

private void AnT\_MouseClick(object sender, MouseEventArgs e)

{

if (\_isTouchable)

{

Random rnd = new Random();

// устанавливаем новые координаты взрыва

explosion.SetNewPosition(0, 0, 10);

// случайную силу

explosion.SetNewPower(rnd.Next(20, 80));

// и активируем сам взрыв

explosion.Boooom(global\_time);

}

}

private void AnT\_MouseMove(object sender, MouseEventArgs e)

{

mouseMoveX = e.X;

mouseMoveY = e.Y;

}

private bool \_isAquariumShow = false;

private void Aquarium\_btn\_Click(object sender, EventArgs e)

{

\_isAquariumShow = !\_isAquariumShow;

if (\_isAquariumShow)

{

Aquarium\_btn.BackColor = Color.LightGreen;

ThreeD\_btn.Enabled = false;

comboBox1.Enabled = false;

xBar.Enabled = false;

trackBar1.Enabled = false;

button1.Enabled = false;

Touch\_btn.Enabled = false;

Sheep\_btn.Enabled = false;

Fractal\_btn.Enabled = false;

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

Gl.glClearColor(0, 0, 0, 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);

}

else

{

Aquarium\_btn.BackColor = Color.LightGray;

ThreeD\_btn.Enabled = true;

comboBox1.Enabled = false;

xBar.Enabled = false;

trackBar1.Enabled = false;

button1.Enabled = false;

Touch\_btn.Enabled = false;

Sheep\_btn.Enabled = true;

Fractal\_btn.Enabled = true;

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

Gl.glClearColor(0, 120, 0, 1);

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

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

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

Gl.glDisable(Gl.GL\_LIGHTING);

Gl.glDisable(Gl.GL\_LIGHT0);

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

}

}

private void Fractal\_btn\_Click(object sender, EventArgs e)

{

\_isfractalNeeede = !\_isfractalNeeede;

if (\_isfractalNeeede)

{

Fractal\_btn.BackColor = Color.LightGreen;

}

else

{

Fractal\_btn.BackColor = Color.LightGray;

}

}

private bool \_isfractalNeeede = false;

private void Sheep\_btn\_Click(object sender, EventArgs e)

{

\_isSheepAlive = !\_isSheepAlive;

if (\_isSheepAlive)

{

Sheep\_btn.BackColor = Color.LightGreen;

}

else

{

Sheep\_btn.BackColor = Color.LightGray;

}

}

private bool \_isSheepAlive= false;

private bool \_is3dShow = false;

private void ThreeD\_btn\_Click(object sender, EventArgs e)

{

\_is3dShow = !\_is3dShow;

if (\_is3dShow)

{

ThreeD\_btn.BackColor = Color.LightGreen;

comboBox1.Enabled = true;

xBar.Enabled = true;

trackBar1.Enabled = true;

button1.Enabled = true;

Touch\_btn.Enabled = true;

Sheep\_btn.Enabled = false;

Fractal\_btn.Enabled = false;

Aquarium\_btn.Enabled = false;

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

Gl.glClearColor(0, 0, 0, 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);

} else

{

ThreeD\_btn.BackColor = Color.LightGray;

comboBox1.Enabled = false;

xBar.Enabled = false;

trackBar1.Enabled = false;

button1.Enabled = false;

Touch\_btn.Enabled = false;

Sheep\_btn.Enabled = true;

Fractal\_btn.Enabled = true;

Aquarium\_btn.Enabled = true;

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

Gl.glClearColor(0, 120, 0, 1);

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

Gl.glMatrixMode(Gl.GL\_PROJECTION);

Gl.glLoadIdentity();

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

Gl.glDisable(Gl.GL\_LIGHTING);

Gl.glDisable(Gl.GL\_LIGHT0);

Glu.gluOrtho2D(0.0, 30.0 \* (float)AnT.Width / (float)AnT.Height, 0.0, 30.0);

Gl.glMatrixMode(Gl.GL\_MODELVIEW);

}

}

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

{

string url = "";

switch (comboBox1.SelectedIndex)

{

case 0: url = Path.GetFullPath("models\\dima.ase"); break;

case 1: url = Path.GetFullPath("models\\vlad.ase"); break;

case 2: url = Path.GetFullPath("models\\alisa.ase"); break;

case 3: url = Path.GetFullPath("models\\petuh.ase"); break;

}

sheep = new anModelLoader();

sheep.LoadModel(url);

}

}

}

Код класса anModelLoader.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Tao.DevIl;

using Tao.OpenGl;

namespace Zoo

{

    class anModelLoader

    {

        public string FName = "";

        private bool isLoad = false;

        private int count\_limbs;

        private int mat\_nom = 0;

        private int thisList = 0;

        private int GlobalStringFrom = 0;

        LIMB[] limbs = null;

        LoadTextures[] text\_objects = null;

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

        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;

        }

        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;

        }

        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;

        }

        public int LoadModel(string FileName)

        {

            limbs = new LIMB[256];

            int limb\_ = -1;

            FName = FileName;

            StreamReader sw = File.OpenText(FileName);

            string a\_buff = "";

            string b\_buff = "";

            string c\_buff = "";

            int ver = 0, fac = 0;

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

            {

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

                if (b\_buff[0] == '\*')

                {

                    switch (b\_buff)

                    {

                        case "\*MATERIAL\_COUNT":

                            {

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

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

                                text\_objects = new LoadTextures[mat];

                                continue;

                            }

                        case "\*MATERIAL\_REF":

                            {

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

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

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

                                continue;

                            }

                        case "\*MATERIAL":

                            {

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

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

                                continue;

                            }

                        case "\*GEOMOBJECT":

                            {

                                limb\_++;

                                continue;

                            }

                        case "\*MESH\_NUMVERTEX":

                            {

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

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

                                continue;

                            }

                        case "\*BITMAP":

                            {

                                c\_buff = "";

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

                                    c\_buff += a\_buff[ax];

                                text\_objects[mat\_nom] = new LoadTextures();

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

                                continue;

                            }

                        case "\*MESH\_NUMTVERTEX":

                            {

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

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

                                {

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

                                }

                                continue;

                            }

                        case "\*MESH\_NUMTVFACES":

                            {

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

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

                                {

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

                                }

                                continue;

                            }

                        case "\*MESH\_NUMFACES":

                            {

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

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

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

                                {

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

                                }

                                else

                                {

                                    return -1;

                                }

                                continue;

                            }

                        case "\*MESH\_VERTEX":

                            {

                                if (limb\_ == -1)

                                    return -2;

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

                                    return -3;

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

                                a1 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a2 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a3 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a4 = GetFirstWord(a\_buff, GlobalStringFrom);

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

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

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

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

                                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":

                            {

                                if (limb\_ == -1)

                                    return -2;

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

                                    return -3;

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

                                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);

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

                                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;

                            }

                        case "\*MESH\_TVERT":

                            {

                                if (limb\_ == -1)

                                    return -2;

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

                                    return -3;

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

                                a1 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a2 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a3 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a4 = GetFirstWord(a\_buff, GlobalStringFrom);

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

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

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

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

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

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

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

                                continue;

                            }

                        case "\*MESH\_TFACE":

                            {

                                if (limb\_ == -1)

                                    return -2;

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

                                    return -3;

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

                                a1 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a2 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a3 = GetFirstWord(a\_buff, GlobalStringFrom);

                                a4 = GetFirstWord(a\_buff, GlobalStringFrom);

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

                                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;

                            }

                    }

                }

            }

            count\_limbs = limb\_;

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

            thisList = nom\_l;

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

            CreateList();

            Gl.glEndList();

            isLoad = true;

            return 0;

        }

        private void CreateList()

        {

            Gl.glPushMatrix();

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

            {

                if (limbs[l].NeedTexture())

                    if (text\_objects[limbs[l].GetTextureNom()] != null)

                    {

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

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

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

                    }

                Gl.glEnable(Gl.GL\_NORMALIZE);

                Gl.glBegin(Gl.GL\_TRIANGLES);

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

                {

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

                    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]];

                    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);

                    Gl.glNormal3f(n1, n2, n3);

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

                    {

                        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

                    {

                        Gl.glVertex3f(x1, y1, z1);

                        Gl.glVertex3f(x2, y2, z2);

                        Gl.glVertex3f(x3, y3, z3);

                    }

                }

                Gl.glEnd();

                Gl.glDisable(Gl.GL\_NORMALIZE);

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

            }

            Gl.glPopMatrix();

        }

        private string GetFirstWord(string word, int from)

        {

            char a = word[from];

            string res\_buff = "";

            int L = word.Length;

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

            {

                int ax = 0;

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

                {

                    a = word[ax];

                    if (a != ' ' && a != '\t')

                        break;

                }

                if (ax == L)

                    return res\_buff;

                else

                    from = ax;

            }

            int bx = 0;

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

            {

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

                    break;

                res\_buff += word[bx];

            }

            if (bx == L)

                bx--;

            GlobalStringFrom = bx;

            return res\_buff;

        }

        public void DrawModel()

        {

            if (!isLoad)

                return;

            Gl.glPushMatrix();

            Gl.glTranslated(0, -4, 0);

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

            Gl.glRotated(-90, 1, 0, 0);

            Gl.glCallList(thisList);

            Gl.glPopMatrix();

        }

    }

    class LIMB

    {

        public LIMB(int a, int b)

        {

            if (temp[0] == 0)

                temp[0] = 1;

            VandF[0] = a;

            VandF[1] = b;

            memcompl();

        }

        public float[,] vert;

        public int[,] face;

        public float[,] t\_vert;

        public int[,] t\_face;

        private int MaterialNom = -1;

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

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

        private bool ModelHasTexture = false;

        public bool NeedTexture()

        {

            return ModelHasTexture;

        }

        public void SetMaterialNom(int new\_nom)

        {

            MaterialNom = new\_nom;

            if (MaterialNom > -1)

                ModelHasTexture = true;

        }

        public void createTextureVertexMem(int a)

        {

            VandF[2] = a;

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

        }

        public void createTextureFaceMem(int b)

        {

            VandF[3] = b;

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

        }

        private void memcompl()

        {

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

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

        }

        public int GetTextureNom()

        {

            return MaterialNom;

        }

    }

    class LoadTextures

    {

        private string texture\_name = "";

        private int imageId = 0;

        private uint mGlTextureObject = 0;

        Dictionary<string, string> sheeps = new Dictionary<string, string>

        {

            {"dima", "texture\\dima.jpg"},

            {"vlad", "texture\\vlad.jpg"},

            {"alisa", "texture\\alisa.jpg"},

            {"gray", "texture\\gray.jpg"},

            {"gray1", "texture\\gray1.jpg"},

            {"gray2", "texture\\gray2.jpg"},

            {"white", "texture\\white.jpg"},

            {"background", "texture\\background.jpg"},

            {"aquarium", "texture\\aquarium.jpg"},

            {"petuh", "texture\\petuh.jpg"},

            {"red", "texture\\red.jpg"},

            {"orange", "texture\\orange.jpg"},

        };

        public uint GetTextureObj()

        {

            return mGlTextureObject;

        }

        public void LoadTextureForModel(string FileName)

        {

            texture\_name = FileName;

            Il.ilGenImages(1, out imageId);

            Il.ilBindImage(imageId);

            string fileName = "";

            foreach (KeyValuePair<string, string> sheep in sheeps)

            {

                if (texture\_name.Contains(sheep.Key)) fileName = sheep.Value;

            }

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

            Il.ilEnable(Il.IL\_CONV\_PAL);

            string url = Path.GetFullPath(fileName);

            if (Il.ilLoadImage(url))

            {

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

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

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

                switch (bitspp)

                {

                    case 24:

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

                        break;

                    case 32:

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

                        break;

                }

                Il.ilDeleteImages(1, ref imageId);

            }

        }

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

        {

            uint texObject;

            Gl.glGenTextures(1, out texObject);

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

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

            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;

        }

    }

    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];

    }

}

ПРИЛОЖЕНИЕ Б

СКРИНШОТЫ РАБОТЫ ПРИЛОЖЕНИЯ

