МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ”

Факультет компьютерных наук
Кафедра программирования и информационных технологий

Реализация приложения для просмотра 3D объектов на основе OpenGL.

Курсовой проект

09.03.04 Программная инженерия

Информационные системы и сетевые технологии

Зав. кафедрой _____С.Д. Махортов, д-р физ.-мат. наук, доцент __.__.20__
Обучающийся _____А.В. Павлов, 3 курс, д/о
Руководитель _____Е.М. Михайлов, ст. преподаватель

Воронеж 2024

Содержание

Введение

На данный момент, сообществу разработчиков в индустрии компьютерной графики предоставлен широкий выбор инструментов, для реализации тех или иных нужд. Проходя по всем этапам разработки, зачастую приходится прибегать к оценке собственной работы, чтобы исправить неточности, которые были созданы в ходе разработки. Но на фоне, казалось бы, такого разнообразия инструментов, была выявлена проблема, которая замедляла процесс разработки. Четко вырисовывалась необходимость в инструменте для просмотра 3D-моделей, который будет нацелен на скорость и простоту использования.

Для решения данной проблемы необходимо реализовать приложение, с помощью которого можно было бы просматривать 3D-модели с максимальной эффективностью, что и является целью данной курсовой работы.

На данный момент, есть две графические библиотеки с широким спектром возможностей, на основе которых, можно реализовать быстрый в скорости обработки продукт.

В данной курсовой работе используется библиотека OpenGL, целевая рабочая среда .Net Framework 4.8 и графический интерфейс WPF.

OpenGL – это открытая графическая библиотека, которая представляет кроссплатформенный и независящий от языка программирования набор инструментов, для работы с двухмерной или трёхмерной компьютерной графикой.

Основные преимущества OpenGL:

— Кроссплатформенность: OpenGL работает на различных операционных системах, включая Windows, macOS, Linux. Это

означает, что приложения, с использованием OpenGL, могут быть легко портированы на разные платформы;

— Производительность: OpenGL напрямую взаимодействует с аппаратным обеспечением графического процессора, что позволяет ему эффективно использовать ресурсы системы;

— Контроль: OpenGL предоставляет разработчикам низкоуровневый контроль над процессом рендеринга;

— Расширяемость: Функциональность API может быть расширена с помощью различных библиотек и расширений;

— Открытость: Главный плюс OpenGL в том, что он бесплатен для использования и распространения, а его спецификация доступна всем;

OpenGL можно использовать для различных типов графических приложений, включая системы обработки 3D-моделей, редактирования и анимации компьютерной графики, игровые движки и многое другое.

В качестве языка программирования, был выбран C#. Это объектно-ориентированный язык программирования, который предоставляет широкий выбор инструментов, для реализации приложения. В связке с графическим редактором WPF, было реализовано приложение с удобным и современным интерфейсом.

1 Постановка задачи

Целью данной курсовой работы является разработка графического приложения для быстрого просмотра полигональных моделей. Приложение должно обладать следующим функционалом:

— импортирование и обработка файлов с расширением .obj;

— взаимодействие с трехмерными объектами;

— возможность редактирования панели инструментов;

— возможность смены материала объекта;

— приложение должно иметь высокую скорость обработки трехмерной графики;

— приложение должно иметь расширяемую архитектуру;

2 Анализ предметной области

2. 1 Терминология (глоссарий) предметной области

OpenGL – кроссплатформенный интерфейс программирования приложения для рендеринга 2D и 3D графики.

WPF – графическая подсистема и фреймворк для создания приложений с пользовательским интерфейсом в операционной системе Windows.

.Net Framework – платформа для разработки и выполнения приложений.

MVVM – архитектурный паттерн, который используется для разработки приложений с богатым пользовательским интерфейсом.

XAML – язык разметки, который используется для описания пользовательского интерфейса приложений.

VBO – объект буфера вершин, который используется в OpenGL для хранения и обработки данных о вершинах 3D-моделей.

Шейдер – специальная программа, которая используется в графических приложениях и играх для обработки и визуализации изображений и эффектов.

Рендеринг – процесс создания изображения на основе 3D-моделей, текстур, освещения и других графических данных.

Полигон – геометрическая фигура, состоящая из нескольких граней и вершин.

Вершина – точка в пространстве, которая определяет положение, цвет, текстуру и другие свойства геометрического объекта.

Грань – плоская или криволинейная поверхность, ограниченная одним или несколькими ребрами.

Нормаль – вектор, перпендикулярный поверхности геометрического объекта в точке вершины или грани.

API – набор инструментов, функций, процедур и протоколов, которые предоставляет программное обеспечение для использования другим программным обеспечением.

GPU – специализированный процессор, предназначенный для обработки графических данных.

CPU – основной компонент компьютера, который выполняет большинство вычислительных операций.

Текстура – цифровое изображение, которое используется для наложения на поверхность геометрического объекта в компьютерной графике.

FPS – цифровое изображение, которое используется для наложения на поверхность геометрического объекта в компьютерной графике.

2. 2 **Средства реализации**

Основной стек технологий:

— .Net Framework 4.8;

— C# 12

— OpenGL;

— графический интерфейс WPF;

— архитектурный шаблон MVVM.

.Net Framework 4.8 представляет из себя надстройку экосистемы, которая предлагает широкий выбор языков программирования, общеязыковую среду выполнения и возможность кроссплатформенной разработки. В данной версии, фреймворк содержит в себе ряд оптимизаций, которые могут улучшить производительность приложения в рабочих нагрузках с интенсивным использованием памяти и ввода-вывода, что положительно сказывается на приложении для взаимодействия с компьютерной графикой.

C# был выбрана из-за своей гибкости, универсальности, простоте, наличию большого набора утилит для решения разных задач и надёжному коду. А также язык является кроссплатформенным, т.е позволяет запускать написанный код на любом устройстве. Дополнительные бонусы в пользу выбора C# — регулярные обновления и совместимость между версиями.

OpenGL является мощным инструментов для разработки графических приложений. Ключевой особенностью данной библиотеки, является низкоуровневый контроль над графическим оборудованием. Это позволяет оптимизировать приложения для максимальной производительности и эффективности. Кроме того, разработчику предоставляется широкий спектр инструментов и функций для создания сложных графических эффектов и

визуализаций. Ещё одним важным преимуществом является кроссплатформенность данной библиотеки. OpenGL может быть использован на различных операционных системах, что позволяет разработчикам создавать приложения, которые могут быть запущены на широком спектре устройств. Кроме того, данная библиотека является бесплатной, что является большим плюсом для сообщества разработчиков. Это позволяется разработчикам использовать её в своих проектах без каких-либо лицензионных ограничений или платежей.

WPF является подсистемой .Net Framework, которая предназначена для разработки графических интерфейсов пользователя, для приложений Windows. Она использует декларативный язык разметки XAML, который позволяет разработчикам создавать графический интерфейс, отдельно от кода приложения. Это позволяет разделить работу между дизайнерами и разработчиками, а также упрощает сопровождение и обновление приложений. Также, WPF включает в себя поддержку векторной графики, анимации, мультимедия, 3D-графики, а также возможность создания пользовательских элементов управления.

MVVM является архитектурным паттерном, который разделяет графический интерфейс приложения на три отдельных компонента. Разделение графического интерфейса на три отдельных компонента позволяет разработчикам создавать более модульный, понятный и простой в обслуживании код. Это также облегчает тестирование и отладку приложения, так как каждый компонент может быть протестирован и отлажен независимо от других. Большинство современных графических редакторов построены на основе паттерна MVVM.

2. 3 **Vertex Buffer Object**

Как говорилось ранее, OpenGL предоставляет готовый набор инструментов для полноценной работы с компьютерной графикой.

Первым делом, необходимо понять, что из себя представляет 3D объект, а именно, это набор точек в трехмерном пространстве, которые последовательно соединены между собой. Эти точки называются вершинами, они определяются своими координатами и во время связи между собой образуют грани, которые в свою очередь образуют объемный объект.

В OpenGL вершины определяются с помощью массивов и буферов вершин.

Буфер вершин представляет из себя область памяти, которая используется для хранения информации о вершинах 3D-модели. Буфер вершин может быть в памяти центрального процессора или графического процессора. В нашем случае, используется буфер вершин GPU, который называется VBO.

VBO позволяет увеличить производительность при рендеринге трехмерной сцены. Это связано с тем, что при использовании буферов вершин информация передается на GPU один раз, а затем используется многократно при рендеринге, в ином случае, информация о вершинах передавалась бы в GPU каждый раз при рендеринге.

VBO хранит в себе следующую информацию о координатах вершин:

- координаты вершин;
- нормали;
- цвета;
- текстурные координаты.

Также буфер вершин позволяет оптимизировать память, используемую для хранения 3D-моделей. Это связано с тем, что информация о вершинах может быть разбита на отдельные буферы, которые используются только при необходимости. Работа VBO представлена на рисунке 1.

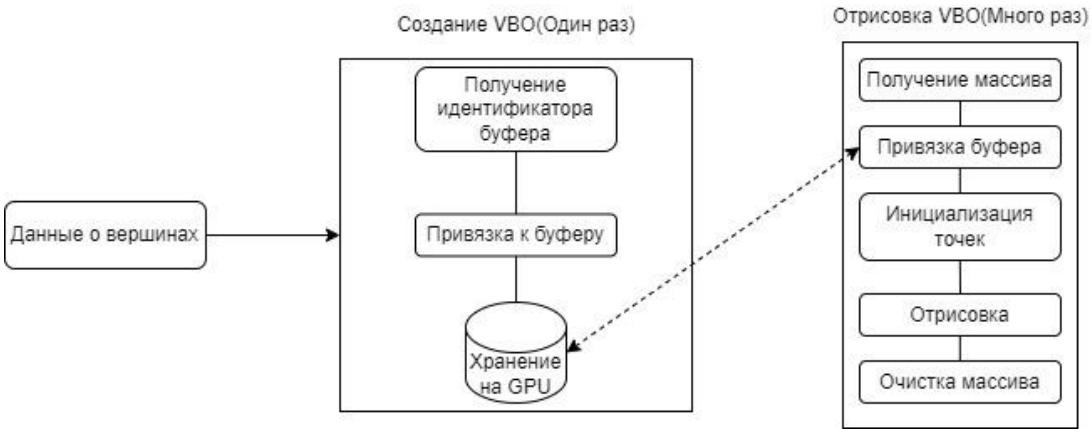

Рисунок 1 – Множественное использование VBO

2. 4 **Модельная матрица преобразований**

Помимо буферизации, необходимо решить проблему преобразования вершин в пиксели на экране. Это делается с помощью матриц преобразования.

В OpenGL существует несколько типов матриц преобразований:

- модельная матрица;
- матрицы вида;
- проекционная матрица.

В нашем случае, мы будем использовать модельную матрицу преобразований. Она представляет из себя матрицу, которая используется для преобразования координат вершин 3D-модели из локальной системы координат модели, в мировую систему координат. Модельная матрица преобразований обладает следующими преимуществами:

— положение и ориентация: Модельная матрица преобразований позволяет задать положение модели в 3D-сцене. Это позволяет размещать модели в любом месте и с любым углом поворота;

— масштабирование: Модельная матрицы преобразований позволяет изменять размеры моделей в 3D-сцене. Это позволяет импортировать модели в одном масштабе, а затем масштабировать их в соответствии с требованиями 3D-сцены;

— эффективность: Модельная матрица преобразований позволяет преобразовать все вершины модели одновременно, что повышает эффективность рендеринга. Это особенно важно для сложных моделей с большим количеством вершин;

— иерархия: Модельная матрица преобразований позволяет создавать иерархию моделей, где каждая модель может иметь свою собственную модельную матрицу преобразований. Это позволяет создавать сложные 3D-сцены из множества взаимосвязанных моделей;

— локальная система координат: Модельная матрицы преобразований позволяет использовать локальную систему координат модели, которая может отличаться от мировой системы координат. Это позволяет импортировать и создавать модели в удобной системе координат, а затем преобразовывать их в мировую систему координат для отображения в 3D-сцене.

Схема работы модельной матрицы преобразований представлена на рисунке 2.

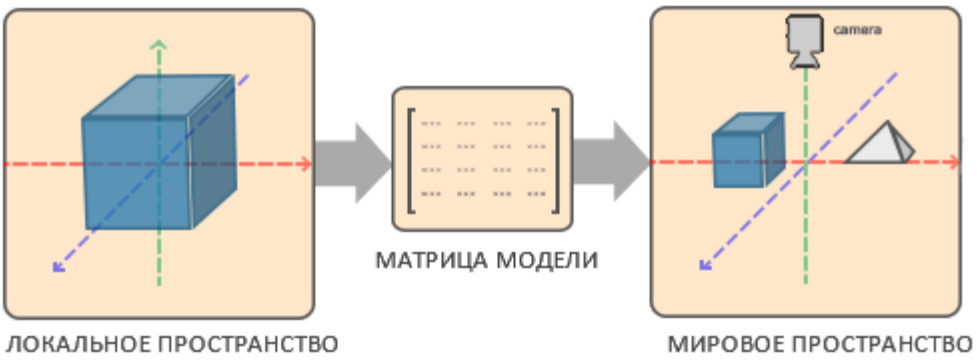

Рисунок 2 – Переход в мировое пространство

На рисунке 2 представлен переход из локального пространства трехмерной сцены в мировое пространства, благодаря преобразованиями модельной матрицы.

2. 5 **Шейдеры**

После того, как вершины были преобразованы в пиксели, нужно определить, как они будут отображаться на экране. Это делается с помощью шейдеров.

Шейдер представляет из себя специальную программу, которая используется для обработки графических данных в графическом конвейере на GPU. В OpenGL существует 2 типа шейдеров:

- вершинные шейдеры;
- фрагментные шейдеры.

Вершинный шейдер выполняется для каждой вершины в VBO, и может выполнять различные преобразования и изменения над этими вершинами. В нашем случае, вершинный шейдер используется для преобразования координат вершин, изменения их цвета и текстурных координат. Это необходимо для того, чтобы модели отображались корректно на экране, с учетом всех необходимых эффектов и текстур.

Фрагментный шейдер выполняется для каждого пикселя, который будет отображен на экране, и определяет цвет этого пикселя на основе различной информации, такой как координаты вершин, текстуры, нормали и других атрибутов. В нашем случае, данный шейдер используется для определения цвета пикселей на основе информации о вершинах, текстурах и нормалях. Это позволяет создавать более реалистичные изображения, с учетом освещения, текстур и других факторов.

В данной курсовой работе используется оба типа шейдеров, это связано с тем, что они выполняют разные функции в графическом конвейере и используются для обработки разных типов данных. Это позволит в дальнейшем создавать сложные графические эффекты и визуализации, которые невозможно реализовать с помощью стандартных графических примитивов.

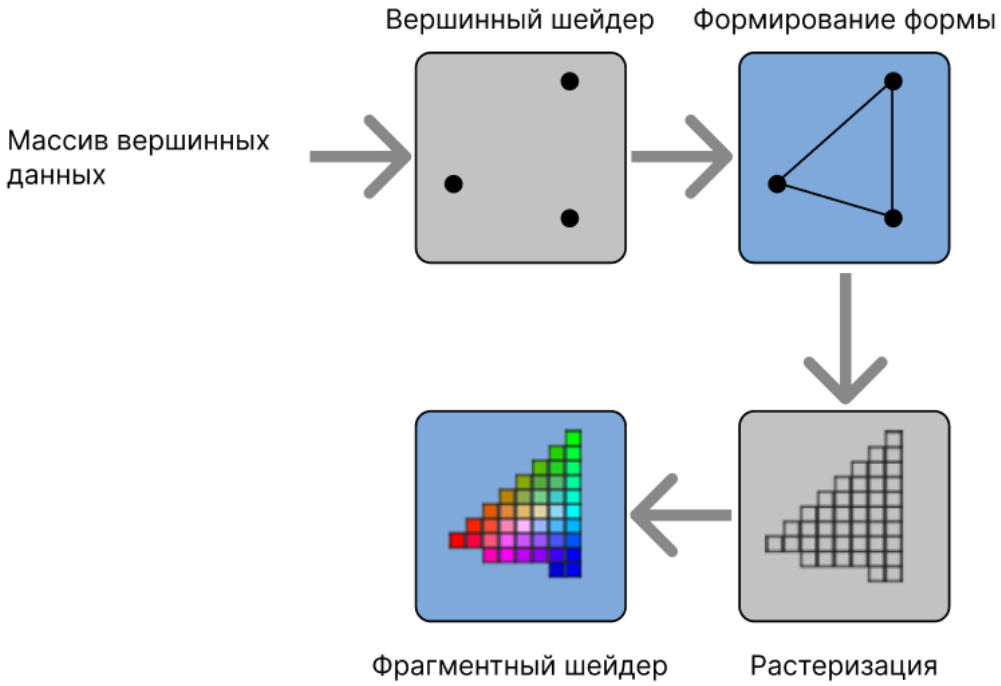

Рисунок 3 – Использование шейдеров

3 Реализация

3. 1 Реализация логики

3. 1. 1 Обработка графики

Перед построением архитектуры приложения, возникла необходимость в построении схемы распределения обработки графики. Схема обработки графики с помощью CPU и GPU представлена на рисунке 4.

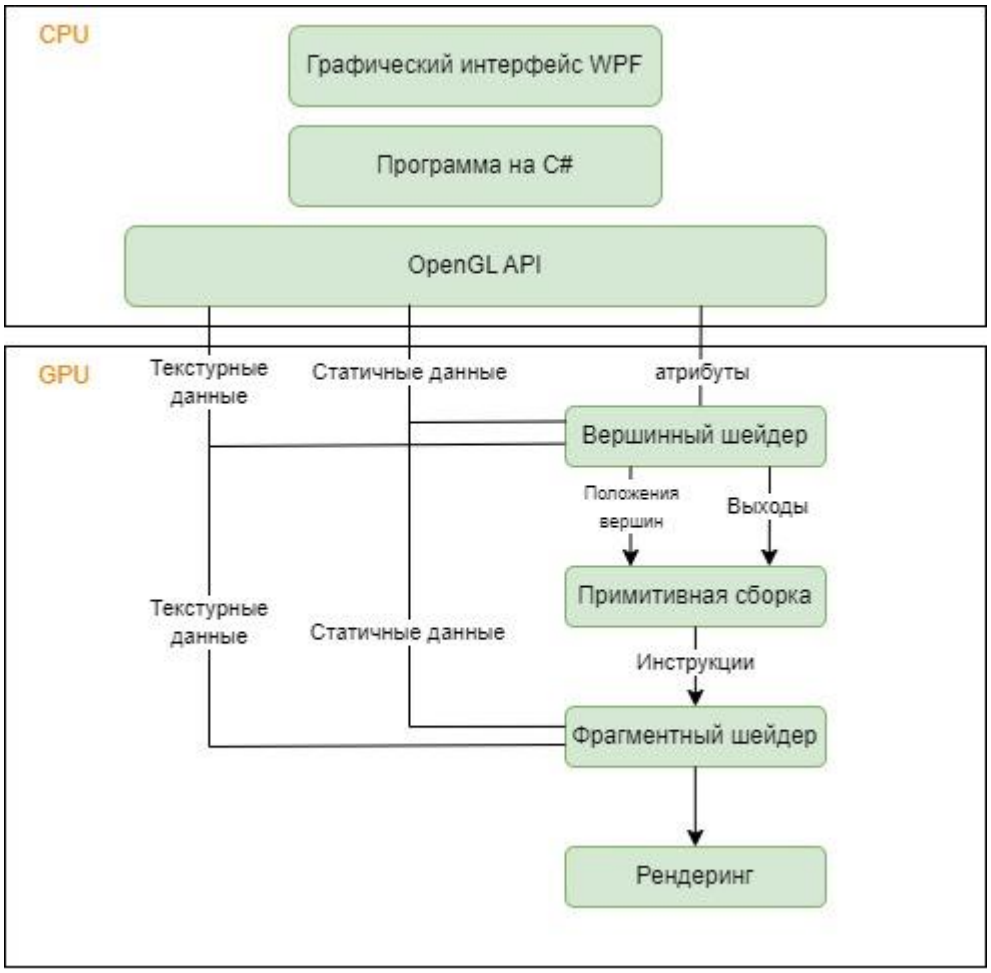

Рисунок 4 – Схема обработки графики

В этой схеме представлена архитектура отображения 3D-объекта, которая используется в данной курсовой работе. Схема показывает, как происходит взаимодействие между центральным процессором и графическим процессором при рендеринге 3D-графики с помощью OpenGL.

На стороне CPU выполняется стандартный код приложения, который отвечает за логику и управление приложением. Кроме того, на стороне CPU выполняется код API OpenGL, который отвечает за инициализацию графического контекста, загрузку шейдеров, буферов и текстур, а также за отправку команд и данных на GPU для рендеринга.

На стороне GPU происходит процесс рендеринга шейдеров, который лежит в основе OpenGL. Для этого GPU получает данные по каналу связи со стороны CPU. Эти данные включают в себя команды, необходимые для рендеринга, такие как тип примитива, количество вершин, а также данные о вершинах, текстурах, нормалях и других атрибутах.

Затем GPU передает эти данные в соответствующий шейдер, который выполняет необходимые преобразования и вычисления для создания изображения. Вершинный шейдер выполняется для каждой вершины в VBO, преобразует координаты вершин, изменяет их цвет и текстурные координаты. Фрагментный шейдер выполняется для каждого пикселя на экране, определяет цвет пикселя на основе информации о вершинах, текстурах, нормалях и генерирует новые пиксели.

После того, как изображение создано, GPU передает его на экран для отображения. Этот процесс происходит очень быстро, позволяя создавать сложные 3D-сцены и эффекты в реальном времени.

3. 1. 2 **UML диаграмма активности взаимодействия с 3D-объектом**

Схема работы редактирования 3D-объекта показана на рисунке 5.

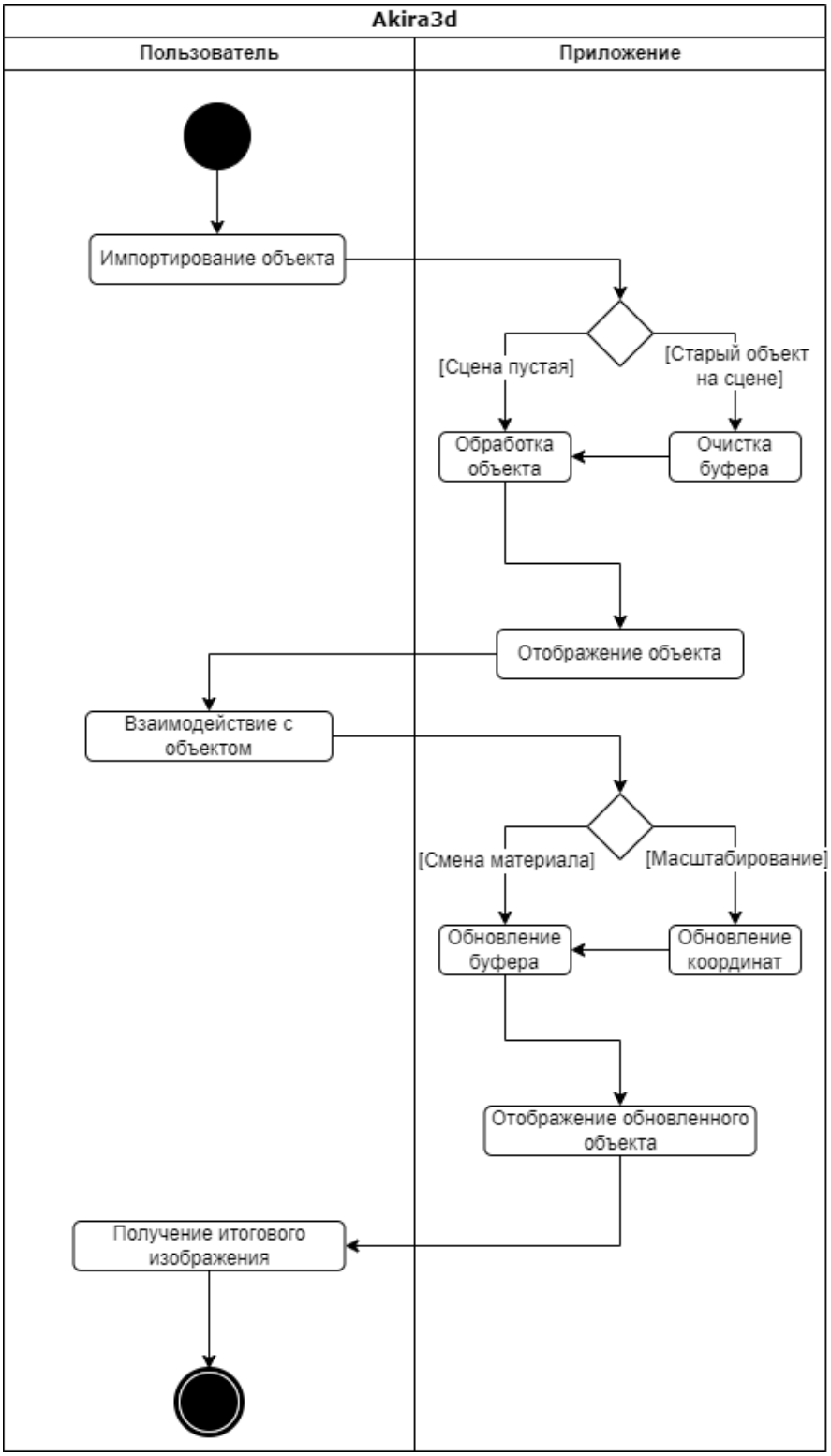

Рисунок 5 – Диаграмма активности редактирования

Входным элементом приложения является файл с расширением .obj. Этот файл представляет собой текстовый документ, который содержит информацию о 3D-модели, включая

координаты вершин, описание граней модели, текстурные координаты, описание материалов модели и описание групп объектов.

Для качественного импортирования данных из файла .obj, в приложении был реализован загрузчик, который обрабатывает входные данные этого файла. В ходе работы программы, происходит обработка входного файла и отрисовка содержимого на экране.

Если во время загрузки файла на сцене находится другой объект, то произойдет очистка буфера изображения. Это необходимо для того, чтобы объекты не накладывались друг на друга.

При вращении или масштабировании объекта, происходит изменение координат на сцене, после чего происходит обновление буфера изображения. В случае смены материала объекта, происходит обновление буфера изображения без последующего обновления координат.

После проведенных работ VBO, происходит обновление экрана, в итоге пользователь получает измененное изображение своего объекта.

3. 1. 3 **Архитектура приложения**

Архитектура приложения была выстроена на основе паттерна разработки MVVM. Общая структура архитектура приложения изображена на рисунке 6:

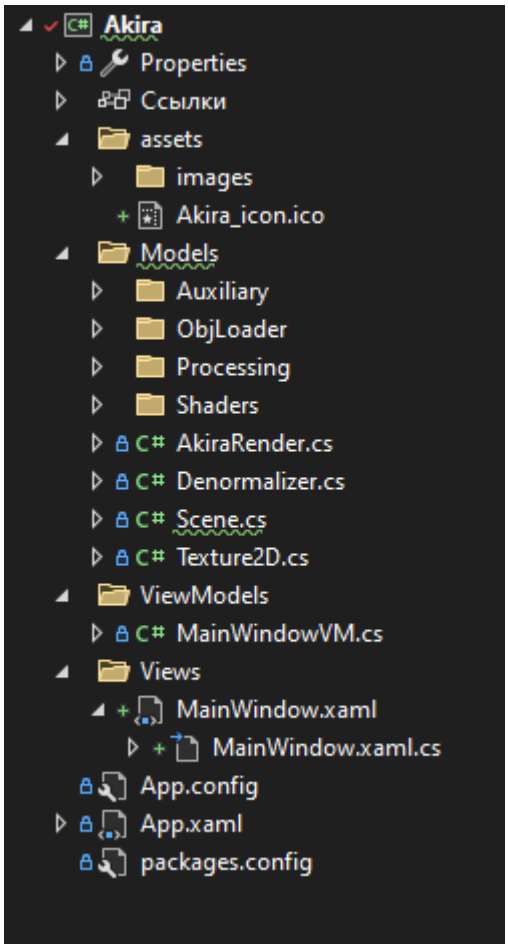

Рисунок 6 – Архитектура приложения

Как видно из изображения, приложение состоит из четырех основных пакетов:

- пакет assets: данный пакет представляет из себя набор изображений, на основе которых был построен графический интерфейс приложения;
- пакет Models: данный пакет содержит основную логику приложения, включая модели данных, алгоритмы обработки и бизнес-логику;

- пакет ViewModels: данный пакет являет прослойкой между пакетами Models и Views. В нем прописана логика взаимодействия пользователя с функционалом приложения;
- пакет Views: данный пакет содержит в себе клиентскую часть приложения, а именно его интерфейс, с помощью которого пользователь взаимодействует с приложением.

Остальные файлы представляют из себя конфигурацию приложения.

Реализация логики приложения, а именно пакет Models, состоит из нескольких дополнительных подпакетов и классов:

- пакет Auxiliary: содержит в себе вспомогательные классы для загрузчика объектов на сцену и реализацию таких функций, как редактирование интерфейса приложения и редактирование модели на сцене;
- пакет ObjLoader: данный пакет представляет из себя реализацию загрузки объектов в приложение. В частности, пакет реализует алгоритмы чтения и парсинга файлов формата .obj, которые содержат описание 3D-моделей;
- пакет Processing: этот пакет содержит в себе классы для обработки каждого элемента в 3D-модели, включая вершины, грани, текстурные координаты и нормали. Также, данный пакет включает в себя реализацию алгоритмов обработки геометрических данных, такие как триангуляция и вычисление нормалей;
- пакет Shaders: этот пакет содержит в себе реализацию шейдеров на языке GLSL, которые используются для визуализации объектов на сцене;
- класс AkiraRender: данный класс реализует рендер, в нем происходит взаимодействие с шейдерами;

- класс Denormalizer: представляет реализацию преобразования геометрических данных модели из нормализованного вида в физический. Это необходимо для корректного отображения объекта на экране;
- класс Scene: отвечает за отрисовку сцены с использованием рендера;
- класс Texture2D: этот класс реализует использование текстур на моделях и их смену.

3. 2 **Реализация графического интерфейса**

Данный графический интерфейс предоставляет пользователям возможность полноценно использовать все функции данного приложения. Общий внешний вид данного интерфейса изображен на рисунке 7:

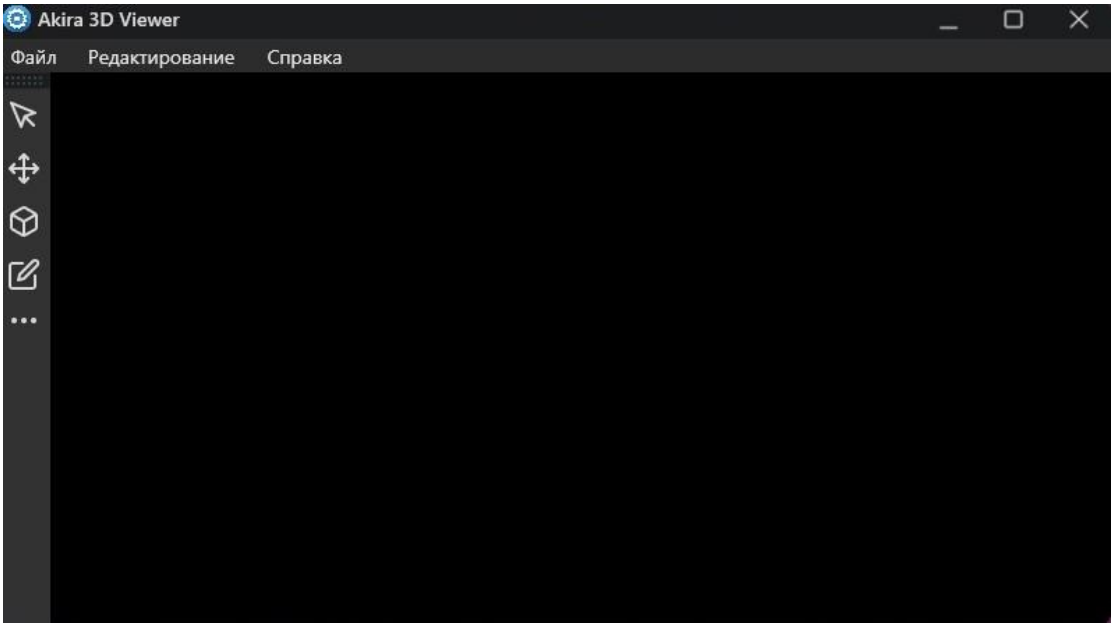

Рисунок 7 – Графический интерфейс

Как видно на изображении, дизайн приложения разделен на три функциональных блока. Верхний блок состоит из набора кнопок, представляющих следующие функции:

- импортирование файла: данная кнопка позволяет пользователям загрузить файл в приложение для дальнейшей обработки;
- редактирование рабочей среды: данная кнопка предоставляет пользователям доступ к настройкам рабочей среды, а именно изменению положения кнопок;
- переход к справке: эта кнопка предоставляет пользователям доступ к руководству по использованию приложения, где они могут перейти к исходному коду приложения и узнать больше о возможностях приложения.

Левый блок также состоит из набора кнопок, предоставляющих следующие функции:

- перетаскивание объекта на сцене: данная кнопка позволяет пользователям перемещать объект на сцене, выбрав его и перетащив в нужно место;
- вращение объекта на сцене: эта кнопка позволяет пользователям вращать объект на сцене вокруг своей оси, выбрав его и перемещая курсор мыши;
- масштабирование объекта на сцене: эта кнопка позволяет пользователям изменять размеры объекта на сцене, выбрав его и вращая колесико мыши;
- смена материала или текстуры объекта: данная кнопка позволяет пользователям изменять материал или текстуру объекта, выбрав нужный материал или текстуру.

Последний функциональный блок представляет из себя сцену, на которой будет происходить отрисовка загруженного файла. Эта сцена является основным рабочим пространством приложения, где пользователи могут видеть результаты своих действий и манипуляций с объектами.

4 Тестирование

4. 1 Характеристики оборудования для тестирования

Тестирование проводилось с помощью оборудования, имеющего следующие характеристики:

— OS Windows 11;

— процессор 10th Gen Intel(R) Core(TM) i5-10300H 2.50 GHz;

— видеокарта NVIDIA GeForce GTX 1650 Ti

— ОЗУ 16 Гб;

— 64-разрядная операционная система, процессор x64.

4. 2 **Результаты тестирования**

Необходимо сравнить количество FPS при обработке 3D-моделей с разным количеством полигонов и в разных состояниях. Результаты тестирования приложения представлены в таблице 1.

Таблица 1 - Результаты тестирования функций с использованием аудита и без него

Количество полигонов	Название операции	Количество FPS
15783	Статичное положение	114
	Вращение объекта	106
	Масштабирование объекта	103
	Перемещение объекта	110
	Смена материала объекта	113
	Смена текстуры объекта	112
43640	Статичное положение	97
	Вращение объекта	89
	Масштабирование объекта	84
	Перемещение объекта	93
	Смена материала объекта	95
	Смена текстуры объекта	93
128903	Статичное положение	37
	Вращение объекта	32
	Масштабирование объекта	42
	Перемещение объекта	28
	Смена материала объекта	36
	Смена текстуры объекта	36

После проведенного тщательного тестирования приложения, было выявлено, что оно показывает хорошие результаты при работе с объектами с разным количеством полигонов.

Во время тестирования, поочередно были загружены модели с разным количеством полигонов, для каждой из них были проведены замеры FPS, при использовании различных функций.

Во время статичного положения объекта, приложение показывало стабильно хороший уровень FPS, даже при работе с высокополигональными объектами. При вращении, масштабировании и перемещении объекта, приложение также показывало хорошие результаты, однако, было замечено снижение уровня FPS при работе с моделями с очень большим уровнем полигонов. При смене материала и текстуры объекта, приложение так же работало стабильно, без существенных изменений.

Заключение

В результате выполнения курсовой работы было разработано графическое приложение для быстрого просмотра полигональных моделей. Приложение обладает следующим функционалом:

— импортирование и обработка файлов с расширением .obj;

— взаимодействие с трехмерными объектами;

— возможность редактирования панели инструментов;

— возможность смены материала объекта;

— высокая скорость обработки трехмерной графики;

— расширяемая архитектура за счет паттерна MVVM.

Список использованных источников

1. Introduction to ArangoDB’s Technical Documentation and Ecosystem [Электронный ресурс]. – Режим доступа: https://www.arangodb.com/docs/stable/ – Заглавие с экрана. – (Дата обращения: 15.03.2023).
2. Official ArangoDB Java Driver [Электронный ресурс]. – Режим доступа: https://github.com/arangodb/arangodb-java-driver – Заглавие с экрана. – (Дата обращения: 15.03.2023).
3. Apache Kafka Documentation [Электронный ресурс]. – Режим доступа: https://kafka.apache.org/documentation/ – Заглавие с экрана. – (Дата обращения: 10.04.2023).
4. CompletableFuture [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/util/concurrent/CompletableFuture.html – Заглавие с экрана. – (Дата обращения: 10.04.2023).
5. ArangoDB: a perfect database for projects with a high level of uncertainty [Электронный ресурс]. – Режим доступа: https://www.mindk.com/blog/arangodb/ – Заглавие с экрана. – (Дата обращения: 18.03.2023).
6. ArangoDB - Quick Guide [Электронный ресурс]. – Режим доступа: https://www.tutorialspoint.com/arangodb/arangodb_quick_guide.htm – Заглавие с экрана. – (Дата обращения: 18.03.2023).
7. Apache Kafka: основы технологии [Электронный ресурс]. – Режим доступа: https://habr.com/ru/companies/southbridge/articles/550934/ – Заглавие с экрана. – (Дата обращения: 20.04.2023).
8. Что такое Apache Kafka [Электронный ресурс]. – Режим доступа: https://timeweb.cloud/blog/apache-kafka-obzor Заглавие с экрана. – (Дата обращения: 20.04.2023).