**Список обозначений и сокращений**

На данном слайде представлен список обозначений и сокращений, используемый при презентации данной работы.

**Постановка задачи**

Целью данной работы является разработка графического программного обеспечения для визуализации трехмерных объектов.

Задачи, которые нужно реализовать для достижения данной цели:

* Создать оконное приложение, которое будет поддерживаться различными операционными системами, такие как Windows, Linux и MacOS
* Реализовать обработку трехмерных моделей и их отображение в ранее созданном оконном приложении
* Разработать возможность импортирования трехмерных моделей из файлов, заготовленных из других CAD-систем по типу Компас, Blender, Maya и AutoCAD
* Реализовать удобный и практичный интерфейс для пользователя, который позволит пользоваться всем функционал для контроля состояний объектов на сцене
* Интегрировать систему рефлексии для лучшего контроля пользовательских и системных типов данных

**Актуальность и обзор существующих решений**

На слайде приведен список наиболее популярные существующие решения на данный момент:

* Unreal Engine
* Unity
* CryEngine
* Godot
* Nau Engine

Актуальность данного графического программного обеспечения заключается в том, что это

* Отечественная разработка, как например Nau Engine, которая поддерживается государственными грантами
* В свете последних событий – Импортозамещение западных продуктов
* Применим в рамках образовательных программ, например для визуализации распределения теплоты при использования МКР или МКЭ

**Прототипирование**

Для разработки графического программного обеспечения за пример взят игровой движок Unreal Engine 4, в который входит:

* Работа и интеграция с различными операционными системами, такими как Windows и Linux. Для поддержки кроссплатформенности для данной работы взята библиотека Graphic Library Framework.
* Для отображение трехмерных объектов также на любой операционной системе. Используется библиотека Open Graphic Library.
* Импортирование текстур и трехмерных объектов из файлов различных форматов. Используются библиотеки SOIL и Assimp соответственно.
* Для интерфейса. Аналогом используется библиотека ImGUI.
* И система рефлексии. Реализации основывается на фреймворке LLVM Clang.

**Графическое окно**

Graphics Library Framework представляет собой библиотеку, написанную на языке программирования C, предназначенную для обеспечения OpenGL необходимыми функциональностями для работы с операционной системой. Данная библиотека позволяет создавать контекст, задавать параметры окна и обрабатывать пользовательский ввод. На слайде представлено оконное приложение, для отображения которого использовалась данная библиотека.

**Графический конвейер**

В OpenGL все объекты находятся в трёхмерном пространстве, однако экран и окно представляют собой двумерную матрицу пикселей. Следовательно, значительная часть задач OpenGL связана с преобразованием трёхмерных координат в двумерные для отображения на экране. Этот процесс преобразования управляется графическим конвейером OpenGL.

Графический конвейер состоит из 6 этапов:

* Вершинный шейдер. На вход которого приходят вершинные данные, на выход набор трехмерных координат.
* Формирование формы. Формирование набора вершин по которой будет строится геометрия.
* Геометрический шейдер. Из набора вершин формируются примитивы.
* Растеризация. Примитивы соотносятся с пикселями на экране, формируя фрагменты.
* Фрагментный шейдер. Вычисление конечного цвета пикселя, на этом этапе чаще всего реализуются различные дополнительные эффекты
* Тесты и смешивание. Этот этап проверяет значение глубины фрагмента и использует их для определения местоположения фрагмента относительно других объектов

На слайде приведен простейший пример отображения треугольника, использующего шейдеры

**Текстурирование**

Текстурирование — процесс, заключающийся в наложении двумерных изображений (текстур) на поверхность модели для придания ей визуальной сложности и реалистичности.

Оно позволяет имитировать свойства материалов, такие как цвет, рельеф, отражающая способность и шероховатость, без значительного увеличения полигональной сетки.

И для облегчения загрузки текстур использовалась библиотека SOIL, так как поддерживает большинство распространенных форматов изображений и может существенно облегчить процесс загрузки и реализации.

**Импортирование трехмерных моделей**

Для импортирования трехмерных моделей использовалась библиотека Assimp (Open Asset Import Library), которая представляет собой широко распространенную библиотеку для импорта 3D-моделей, поддерживающую множество форматов файлов.

Ее ключевой особенностью является унифицированное представление данных: независимо от исходного формата, импортированная модель преобразуется в единую иерархическую структуру данных.

Это позволяет получать доступ к необходимым данным стандартизированным способом, что значительно упрощает работу с различными типами файлов трехмерных моделей.

На слайде представлена схематичная иллюстрация иерархии данных Assimp.

Для себя: При импорте модели библиотека загружает все ее данные в объект Scene, который служит контейнером для всей информации о модели.

Далее Assimp организует данные в виде иерархии узлов, где каждый узел содержит ссылки на соответствующие данные в объекте сцены и может иметь дочерние элементы.

Такая структура обеспечивает удобный и логичный доступ к компонентам модели, включая меши, материалы и текстуры.

**Аффинные преобразования**

Аффинные преобразования представляют собой фундаментальный математический аппарат, широко применяемый в современной компьютерной графике для манипуляции объектами в трехмерном пространстве. Сохраняя ключевые геометрические свойства, они обеспечивают точный контроль положения, ориентации и масштаба объектов.

Аффинные преобразования реализуются через матрицы, основные виды включают:

* Матрица трансляция
* Матрица масштабирования
* Матрица вращения

Важным свойством аффинных преобразований является возможность их композиции - последовательность преобразований может быть представлена единой матрицей, равной произведению матриц отдельных преобразований.

Также в сложных 3D-сценах с иерархической организацией объектов, как в данной работе, особое значение приобретает корректное вычисление глобальных координат дочерних объектов. Оно вычисляется как произведение родительских матриц.

Для себя: Но несмотря на широкое применение матричных преобразований, появляется проблема, возникающие при использовании матрицы вращения, под названием Gimbal Lock.  
И для решения данной проблемы в работе вращение представляется кватернионом, а не эйлеровыми углами.

При этом следует учитывать некоммутативность матричного умножения.

**Системы координат**

Прежде чем вершины объекта будут преобразованы в экранные координаты, они проходят через несколько различных координатных систем и соответственно несколько различных матриц трасформации.  
Как представлено на слайде, происходит в несколько этапов:

* из локальных в глобальные координаты, использую матрицу модели
* из глобальных в видовые, матрица вида
* из видовых в перспективную или ортографическую проекцию, матрица проекции

И на слайде вы можете рассмотреть пример отображения трехмерной модели после импортирования из obj-файла и всех вышеперечисленных математических преобразований

**Освещение**

Также в работе реализована система освещения. Из-за сложности вычислений, освещение в OpenGL базируется на использовании упрощенных математических моделей, приближенных к реальности. Одной из таких моделей является модель освещения по Фонгу.

Она состоит из трех основных компонентов:

* Фоновое освещение (ambient): описывает общее освещение, которое равномерно распределяется по поверхности объекта.
* Рассеянное/диффузное освещение (diffuse): учитывает свет, рассеивающийся равномерно по всем направлениям от источника света.
* Бликовое освещение (specular): моделирует яркие блики, возникающие на поверхности объекта в направлении источника света.

Эти компоненты вместе позволяют создать визуально реалистичное представление освещения объектов. На слайде можете видеть результат применения освещения на объект.

**Виды источников освещения**

Используя компоненты освещения, можно составить ряд различных видов источников освещения.

На слайде представлено 3 основных источников освещения, используемое в графических программных обеспечениях и в разработанном ПО:

* Направленный источник освещения. Обычно используется, как внешнее фоновое освещение, например свет от солнца.
* Точечный источник освещения. Как пример это лампочка, свеча, факел, костры.
* Прожекторный источник освещения. Используется для направленного освещения. Фонарик, уличная лампа, фары от автомобилей.

**Интерфейс**

Помимо простого обозревания сцены, есть желание как-то изменить его состояние не только в коде. И для этого нужен интерфейс.

Для реализации интерфейса была выбрана библиотека ImGUI, представляющая собой решение для создания инструментальных панелей и редакторов.

Основными преимуществами данного выбора стали:

* Immediate-mode подход, позволяющий описывать интерфейс декларативным образом непосредственно в коде логики приложения, как и в Unreal Engine используется Slate
* Высокая производительность и минимальные накладные расходы.
* Простота интеграции с современными графическими API, в нашем случае OpenGL
* Гибкая система кастомизации внешнего вида.
* Широкая экосистема расширений и плагинов

**Система рефлексии**

Система рефлексии— это механизм, позволяющий программе анализировать и взаимодействовать с собственными свойствами и методами во время выполнения программы.

C++ не имеет встроенной поддержки рефлексии, поэтому её реализация требует ручного подхода.

Одним, из которых является обработка абстрактного синтаксического дерева, которая предоставляет библиотека LLVM Clang.

Дерево содержит полную информацию о структуре программы, включая типы данных, функции, классы, поля и методы.

Пройдя по всем файлам программы, находит классы помеченные атрибутом CLASS(), далее проходит по телу класса и находит свойства помеченные атрибутом PROPERTY() и методы помеченные атрибутом FUNCTION().  
И после генерируется файл, в котором создаются специальные методы и структуры для работы с рефлексией в программе.

На слайде представлены два класса, класс с поддержкой рефлексии в unreal engine и в представленной работе, как видите различия минимальным.

**Заключение**

На слайде представлен конечный результат разработки графического программного обеспечения для визуализации трехмерных объектов.

В ходе выполнения данной работы были достигнуты все поставленные цели и задачи.

В результате было создано эффективное и гибкое графическое программное обеспечение для визуализации трехмерных объектов на базе библиотеки OpenGL и языка программирования C++.

Результаты данной работы имеют высокую практическую значимость и могут быть успешно применены в различных сферах, требующих визуализации трехмерных объектов.