Программа представляет собой 3D-визуализатор моделей, разработанный с использованием библиотеки Qt. Она позволяет загружать 3D-модели в формате OBJ, отображать их, а также выполнять трансформации, такие как поворот и перемещение.

▎Основные библиотеки:

• Qt: Основная библиотека для создания графического интерфейса пользователя (GUI).

• QVector, QVector3D: Используются для работы с векторами и трехмерными координатами.

• QFile, QTextStream: Для работы с файлами и чтения текстовых данных.

▎Структура программы:

My3DModelViewer/

├── CMakeLists.txt // Файл сборки проекта

├── main.cpp // Точка входа в программу

├── model.h // Заголовочный файл модели

├── model.cpp // Реализация класса модели

├── viewer.h // Заголовочный файл для отображения модели

├── viewer.cpp // Реализация класса для отображения модели

├── modelviewer.h // Заголовочный файл для представления модели

├── modelviewer.cpp // Реализация класса представления модели

├── mainwindow.h // Заголовочный файл главного окна приложения

└── mainwindow.cpp // Реализация главного окна приложения

▎Классы и их функции:

1. MainWindow:

• Основное окно приложения, включает в себя меню для открытия и сохранения файлов, а также для трансформации моделей.

• Методы:

• openModel(): Открывает диалог для загрузки 3D модели.

• saveText(): Сохраняет текстовое представление модели.

• rotateModel(): Открывает диалог для ввода углов поворота.

• translateModel(): Открывает диалог для ввода смещения по осям.

• updateWindowTitle(): Обновляет заголовок окна с информацией о модели.

2. Model:

• Представляет 3D модель, хранит вершины и грани.

• Методы:

• load(): Загружает модель из файла.

• calculateVolume(): Вычисляет объем модели.

• calculateProjectionArea(): Вычисляет площадь проекции модели.

• getModelDimensions(): Возвращает размеры модели.

• Методы для трансформации: rotateX(), rotateY(), rotateZ(), translate().

3. ModelViewer:

• Отвечает за отображение модели.

• Методы:

• loadModel(): Загружает модель и настраивает масштаб.

• rotateModel(), translateModel(): Вызывают соответствующие методы модели для трансформации.

4. Viewer:

• Отвечает за графическое отображение модели.

• Методы:

• paintEvent(): Обрабатывает событие отрисовки, рисует модель и оси координат.

• mousePressEvent(), mouseMoveEvent(), wheelEvent(): Обрабатывают события мыши для взаимодействия с моделью (выбор вершин, поворот, масштабирование).

Выбор библиотеки Qt для разработки 3D-визуализатора моделей обусловлен несколькими ключевыми факторами:

▎1. Кроссплатформенность

Qt позволяет создавать приложения, которые могут работать на различных операционных системах, таких как Windows, macOS и Linux, без значительных изменений в коде. Это делает Qt идеальным выбором для разработчиков, стремящихся охватить широкую аудиторию.

▎2. Развитая система GUI

Qt предлагает мощные инструменты для создания графического интерфейса пользователя (GUI). Модули, такие как Qt Widgets и Qt Quick, позволяют легко разрабатывать сложные интерфейсы с поддержкой различных элементов управления, диалогов, меню и т.д. Это особенно важно для приложений, требующих интерактивности, как в случае с 3D-визуализатором.

▎3. Поддержка 3D-графики

Qt включает в себя модули, такие как Qt3D, которые специально разработаны для работы с трехмерной графикой. Это позволяет легко интегрировать 3D-объекты, управлять ими и отображать их без необходимости использования сторонних библиотек.

▎4. Богатая документация и сообщество

Qt имеет обширную документацию и активное сообщество, что облегчает процесс обучения и решение возникающих проблем. Это особенно важно для разработчиков, которые могут столкнуться с трудностями на этапе реализации.

▎5. Интеграция с другими модулями

Qt предоставляет множество модулей для работы с различными задачами: от работы с файлами (QFile, QTextStream) до сетевого взаимодействия (Qt Network). Это позволяет разработчику использовать единую библиотеку для решения различных задач, не прибегая к интеграции множества сторонних библиотек.

▎6. Поддержка событийной модели

Qt использует события и сигналы для управления взаимодействием между компонентами, что делает код более чистым и понятным. Это особенно полезно в приложениях с интерактивным интерфейсом.

▎Заключение

Таким образом, выбор Qt и его модулей для разработки 3D-визуализатора обусловлен его мощными возможностями, кроссплатформенностью, поддержкой 3D-графики и удобством работы с графическим интерфейсом. Это делает Qt одним из лучших инструментов для создания современных приложений с богатым функционалом.