Программа для загрузки, отображения и просмотра 3D модели в формате OBJ с использованием Qt6 без Open GL. Программа включает функции для расчета объема и площади проекции модели, а также возможность просмотра модели (класс MainWindow). Код разбит на заголовочные и основные файлы. Используется CMake для сборки.

Весь код программы закоментировать для понимания.

Должна быть возможность вращения модели obj.

Структура программы:

My3DModelViewer/

├── CMakeLists.txt

├── main.cpp

├── model.h

├── model.cpp

├── viewer.h

├── viewer.cpp

├── modelviewer.h

├── modelviewer.cpp

├── mainwindow.h

└── mainwindow.cpp

1. CmakeLists

# Минимальная версия CMake

cmake\_minimum\_required(VERSION 3.14)

# Название проекта

project(My3DModelViewer)

# Установка стандарта C++17

set(CMAKE\_CXX\_STANDARD 17)

# Включение автоматической генерации MOC, RCC и UIC

set(CMAKE\_AUTOMOC ON)

set(CMAKE\_AUTORCC ON)

set(CMAKE\_AUTOUIC ON)

# Поиск и подключение необходимых модулей Qt6

find\_package(Qt6 REQUIRED COMPONENTS Core Gui Widgets)

# Список исходных файлов

set(SOURCES

main.cpp

model.cpp

viewer.cpp

modelviewer.cpp

mainwindow.cpp

)

# Список заголовочных файлов

set(HEADERS

model.h

viewer.h

modelviewer.h

mainwindow.h

)

# Создание исполняемого файла с указанием исходных и заголовочных файлов

add\_executable(My3DModelViewer

${SOURCES}

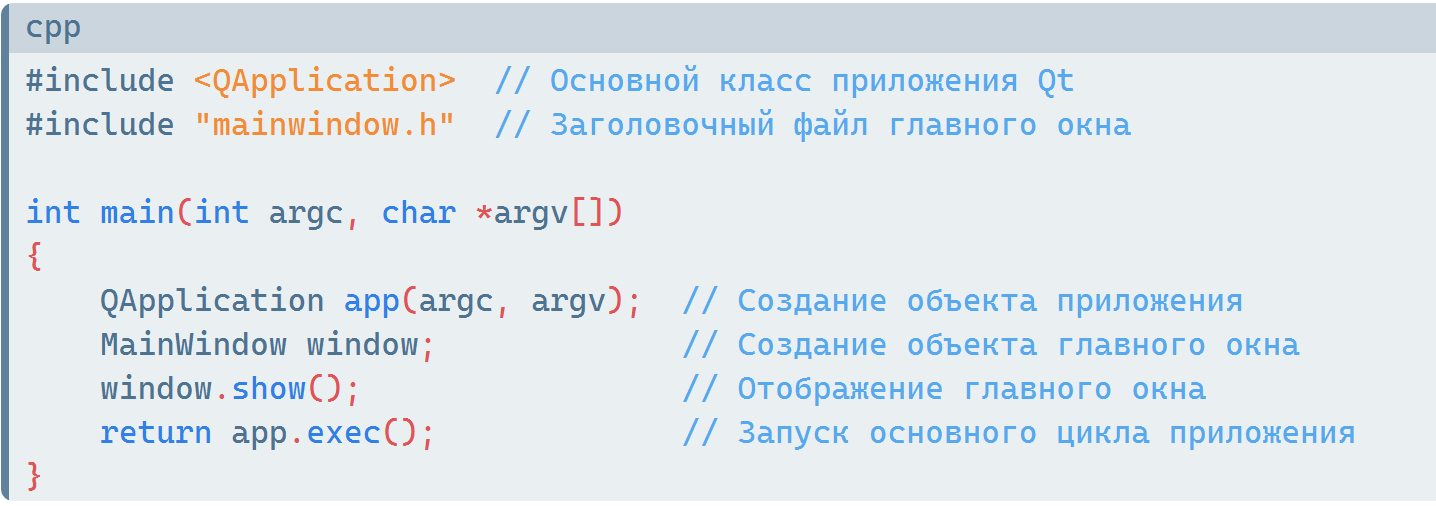
${HEADERS}

)

# Подключение библиотек Qt6 к проекту

target\_link\_libraries(My3DModelViewer PRIVATE Qt6::Core Qt6::Gui Qt6::Widgets)

2. main.cpp



▎Объяснение кода:

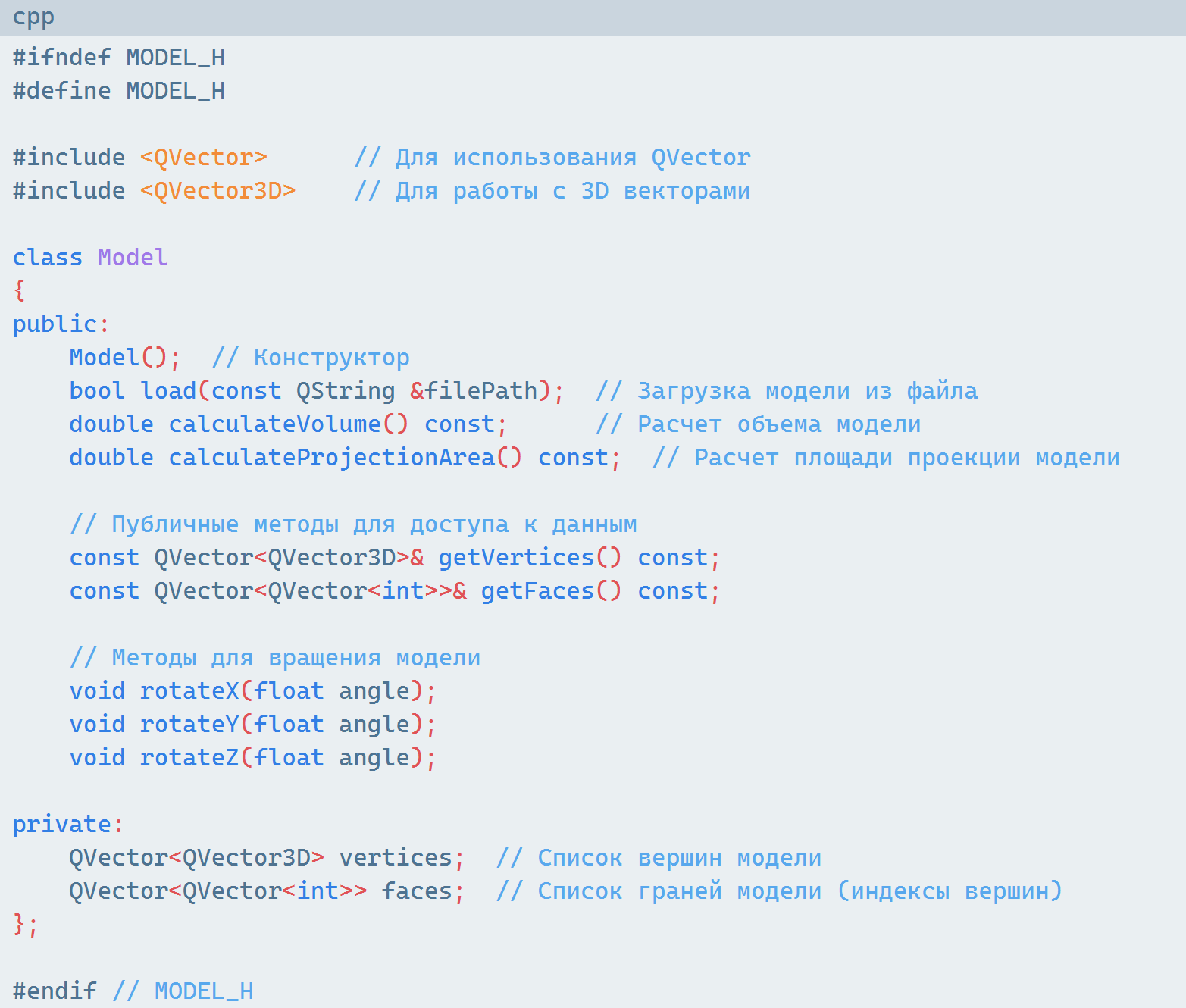
• QApplication: Основной класс, который управляет приложением. Он обрабатывает события, такие как нажатия клавиш и движения мыши.

• MainWindow: Класс, представляющий главное окно приложения. Он создается и отображается с помощью метода show().

• app.exec(): Запуск основного цикла событий приложения. Программа будет работать до тех пор, пока пользователь не закроет окно.

Этот файл является точкой входа в программу. Он создает приложение Qt и запускает главное окно.

3. model.h



Объяснение кода:

• QVector: Используется для хранения вершин и граней модели.

• QVector3D: Представляет 3D вектор (координаты вершины).

• Model: Класс, который представляет 3D модель.

• load: Метод для загрузки модели из файла формата OBJ.

• calculateVolume: Метод для расчета объема модели.

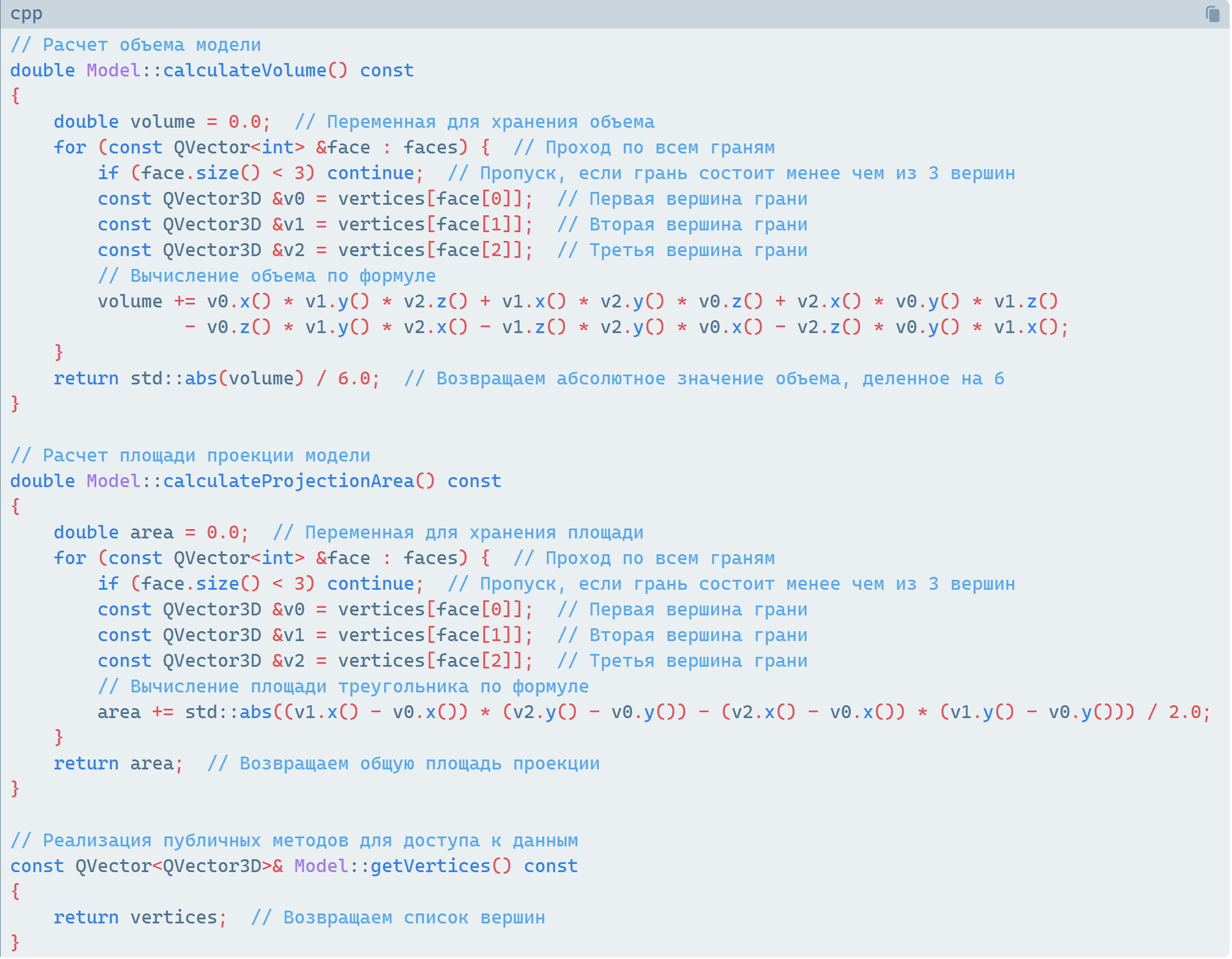
• calculateProjectionArea: Метод для расчета площади проекции модели.

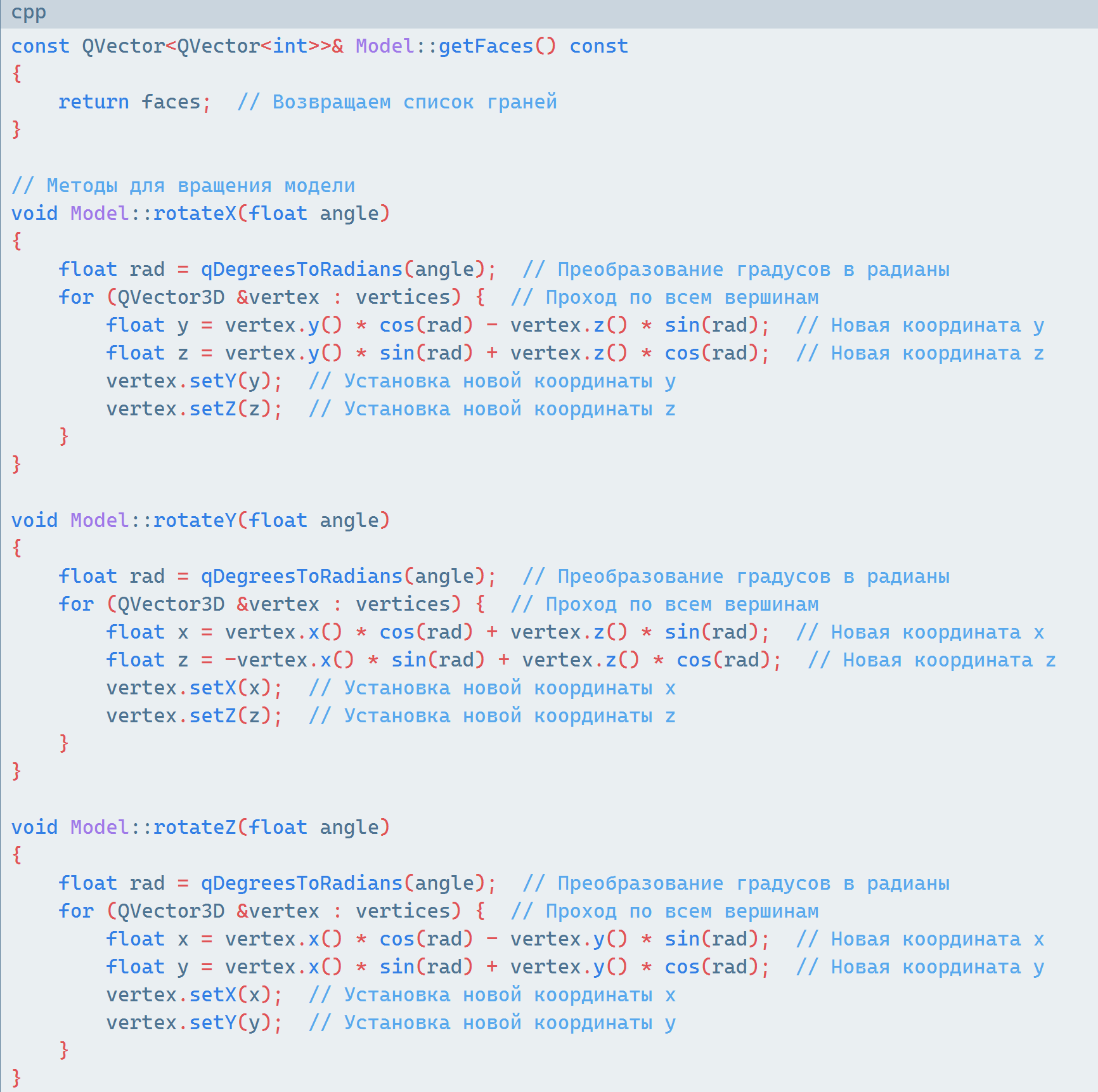
• getVertices и getFaces: Публичные методы для доступа к данным модели.

• rotateX, rotateY, rotateZ: Методы для вращения модели вокруг осей X, Y и Z.

4. model.cpp







Объяснение кода:

• load: Метод загружает модель из файла формата OBJ, считывая вершины и грани.

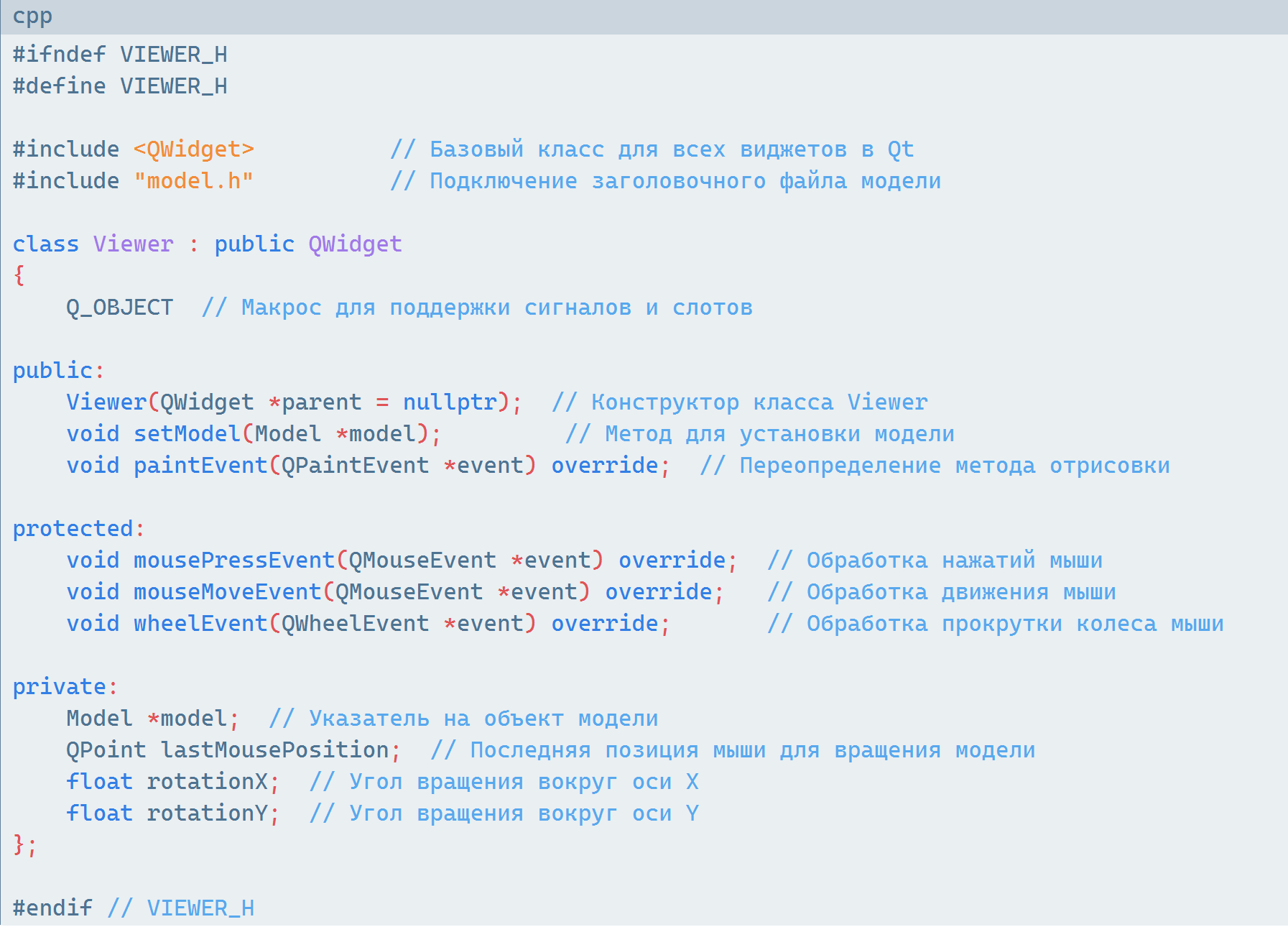
• calculateVolume: Метод вычисляет о модели, используя формулу для тетраэдров.

• calculateProjectionArea: Метод вычисляет площадь проекции модели на плоскость.

• getVertices и getFaces: Методы для доступа к спискам вершин и граней.

• rotateX, rotateY, rotateZ: Методы для вращения модели вокруг осей X, Y и Z, используя математику вращения.

5. viewer.h



Объяснение кода:

• QWidget: Класс, от которого наследуется Viewer. Он предоставляет базовые функции для создания виджетов в приложении Qt.

• Q\_OBJECT: Макрос, необходимый для классов, которые используют механизм сигналов и слотов Qt.

• Viewer: Класс, который отвечает за отображение 3D модели.

• Viewer(QWidget \*parent): Конструктор класса, который принимает родительский виджет (по умолчанию nullptr).

• setModel(Model \*model): Метод для установки модели, которую необходимо отобразить.

• paintEvent(QPaintEvent \*event): Переопределение метода отрисовки, который будет вызываться, когда нужно перерисовать виджет.

• mousePressEvent(QMouseEvent \*event): Метод для обработки событий нажатия кнопки мыши.

• mouseMoveEvent(QMouseEvent \*event): Метод для обработки событий движения мыши.

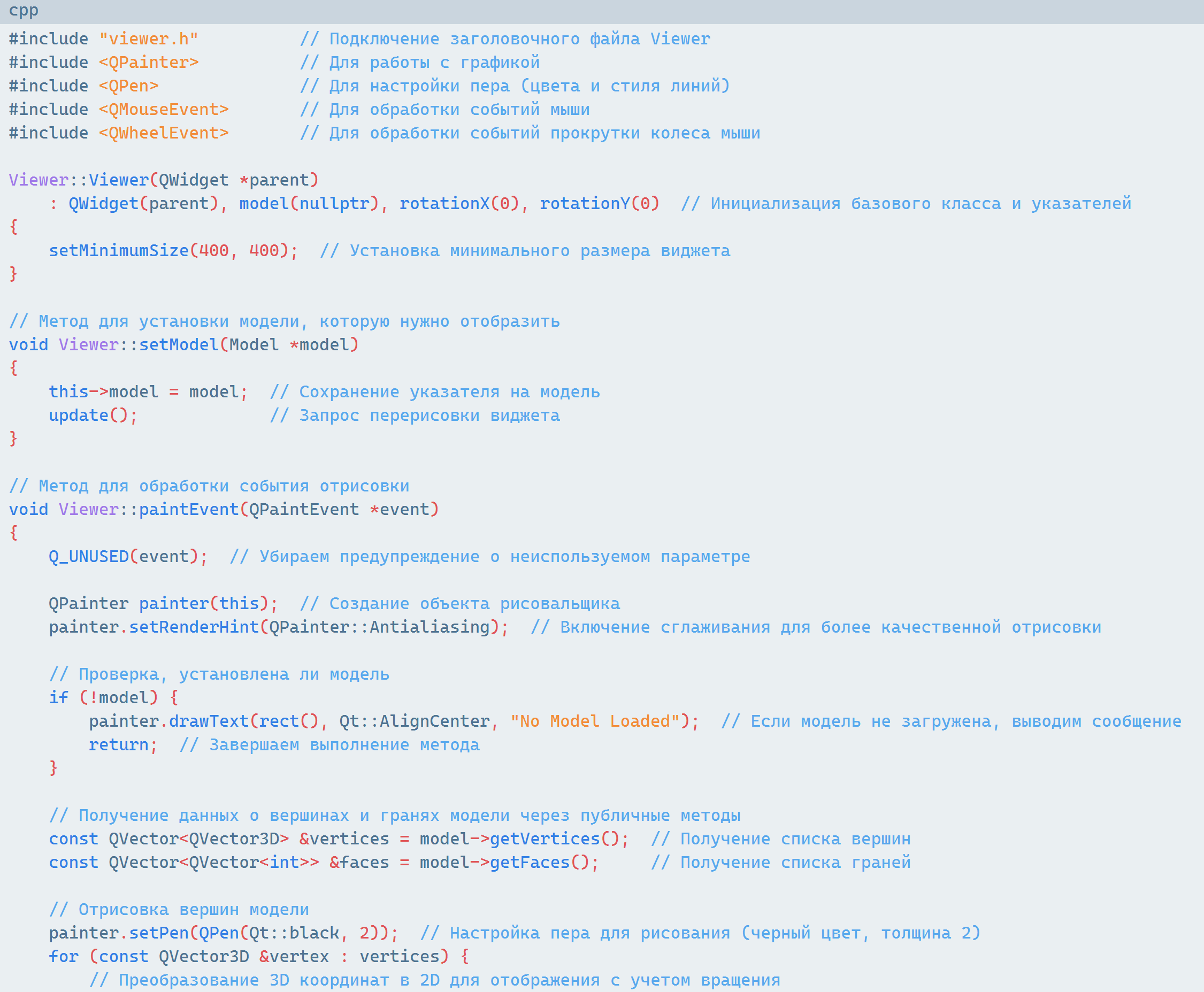
• wheelEvent(QWheelEvent \*event): Метод для обработки событий прокрутки колеса мыши.

• model: Указатель на объект Model, который хранит данные о вершинах и гранях.

• lastMousePosition: Переменная для хранения последней позиции мыши, используемая для вычисления вращения.

• rotationX и rotationY: Углы вращения модели вокруг осей X и Y.

6. viewer.cpp







Объяснение кода:

• Viewer::Viewer: Конструктор класса, инициализирует указатели и устанавливает минимальный размер виджета.

• setModel: Метод для установки модели, которую нужно отобразить.

• paintEvent: Метод, который вызывается для отрисовки виджета. В нем происходит:

• Проверка наличия модели.

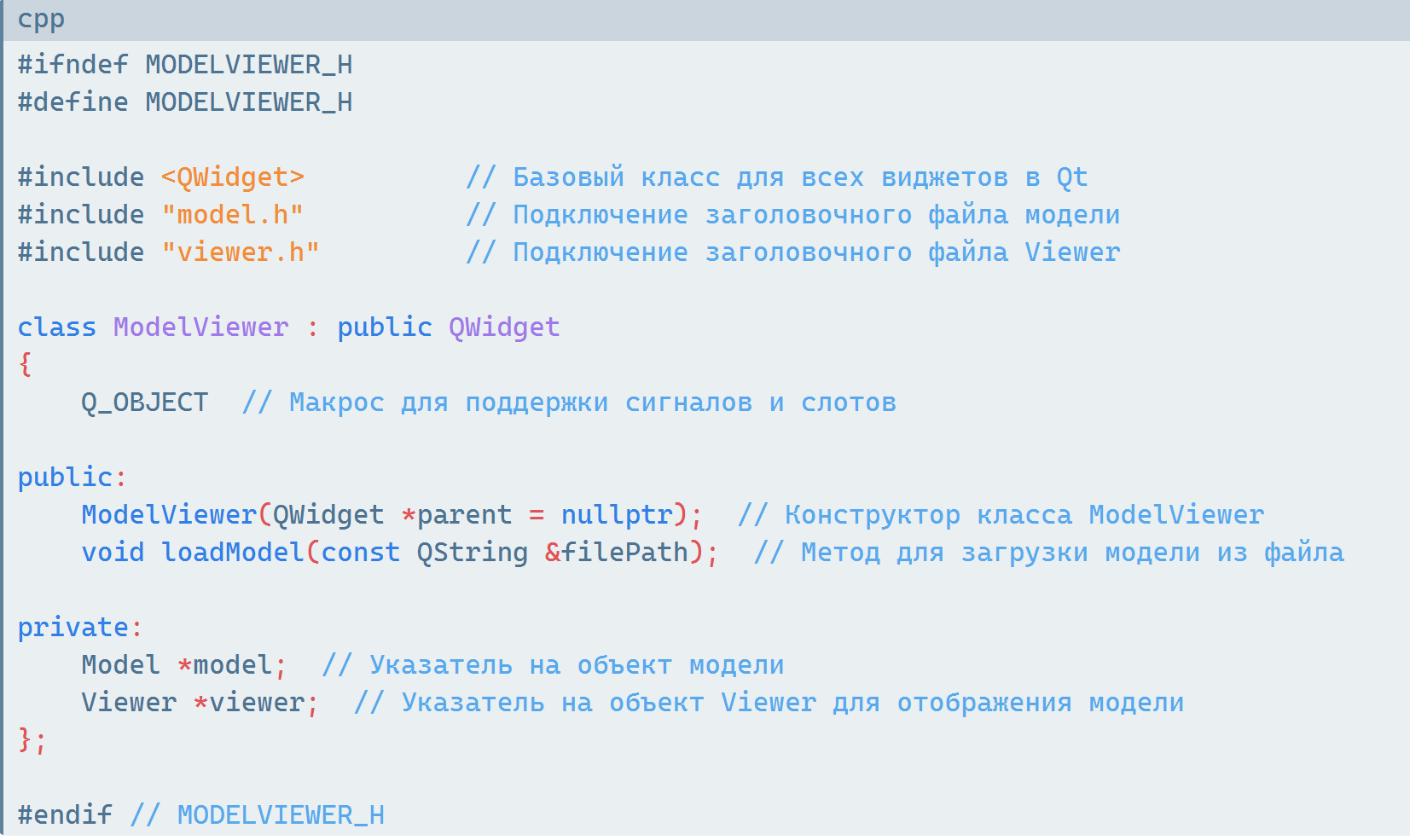
• Отрисовка вершин и граней модели с учетом вращения.

• mousePressEvent: Метод для обработки нажатий мыши. Сохраняет текущую позицию мыши.

• mouseMoveEvent: Метод для обработки движения мыши. Изменяет углы вращения модели в зависимости от движения мыши.

• wheelEvent: Метод для обработки прокрутки колеса мыши. Здесь можно добавить функциональность для изменения масштаба модели.

7. modelviewer.h



Объяснение кода:

• QWidget: Класс, от которого наследуется ModelViewer. Он предоставляет базовые функции для создания виджетов в приложении Qt.

• Q\_OBJECT: Макрос, необходимый для классов, которые используют механизм сигналов и слотов Qt.

• ModelViewer: Класс, который управляет загрузкой модели и ее отображением.

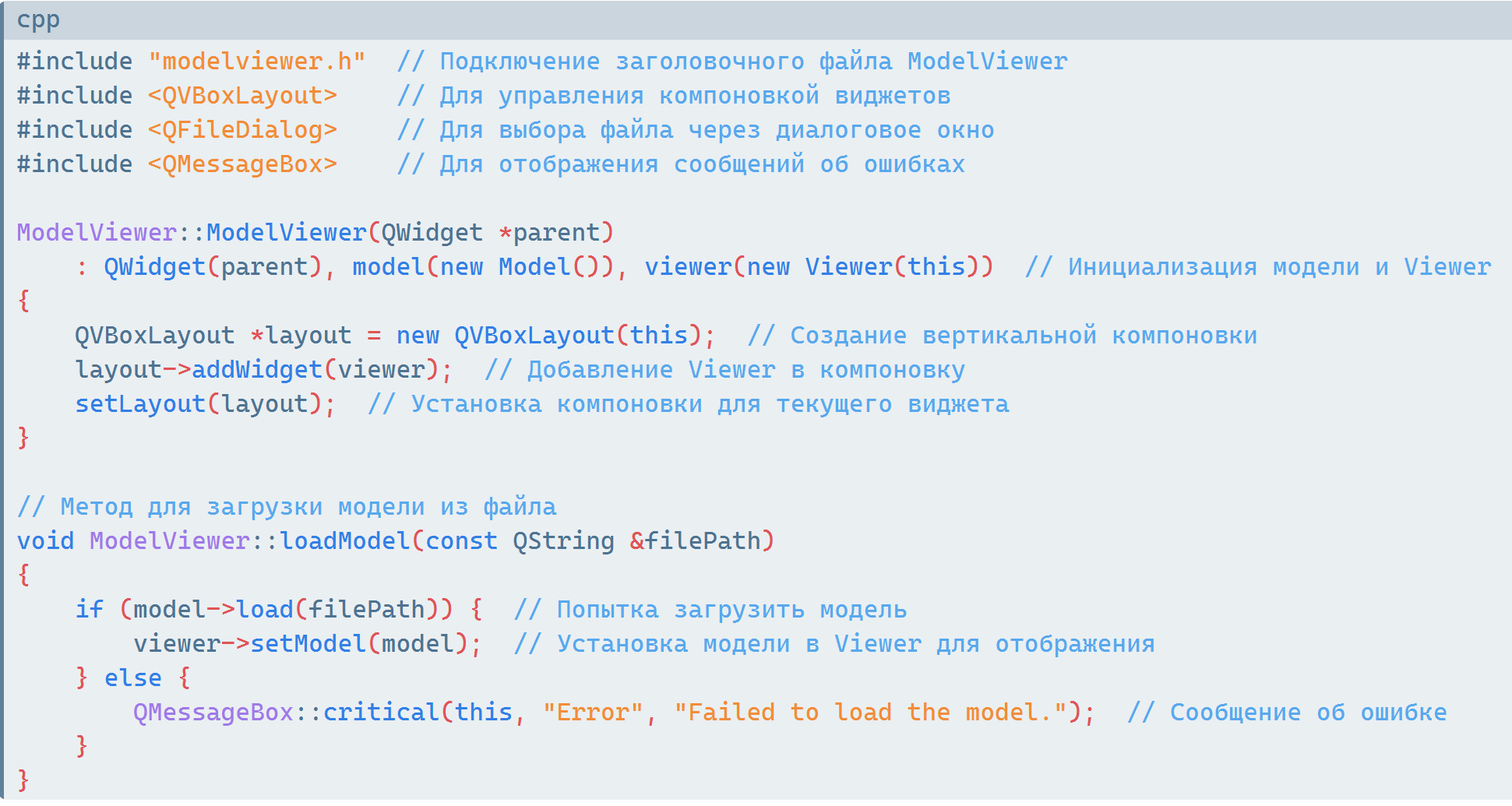
• ModelViewer(QWidget \*parent): Конструктор класса, который принимает родительский виджет (по умолчанию nullptr).

• loadModel(const QString &filePath): Метод для загрузки модели из файла. Он будет вызываться для загрузки и отображения модели.

• model: Указатель на объект Model, который хранит данные о вершинах и гранях.

• viewer: Указатель на объект Viewer, который отвечает за отображение модели на экране.

8. modelviewer.cpp



Объяснение кода:

• ModelViewer::ModelViewer: Конструктор класса, который инициализирует модель и Viewer, а также настраивает компоновку виджетов.

• Создается вертикальная компоновка (QVBoxLayout), в которую добавляется объект Viewer.

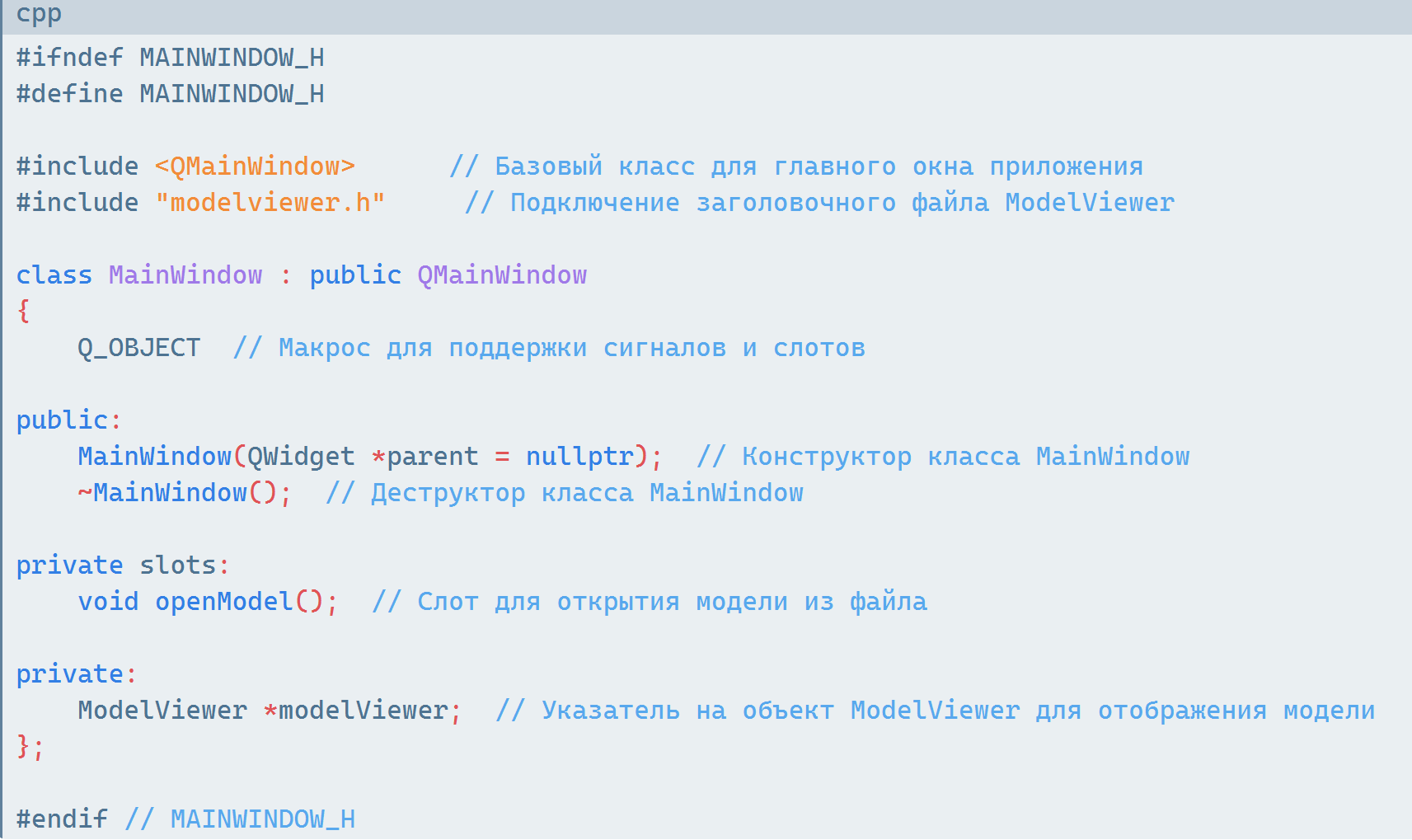
• Компоновка устанавливается для текущего виджета.

• loadModel: Метод для загрузки модели из файла.

• Если модель успешно загружается, она передается в Viewer для отображения.

• Если загрузка не удалась, выводится сообщение об ошибке с помощью QMessageBox.

9. mainwindow.h



Объяснение кода:

• QMainWindow: Класс, от которого наследуется MainWindow. Он предоставляет функциональность для создания главного окна приложения с меню, панелями инструментов и статусной строкой.

• Q\_OBJECT: Макрос, необходимый для классов, которые используют механизм сигналов и слотов Qt.

• MainWindow: Класс, который представляет главное окно приложения.

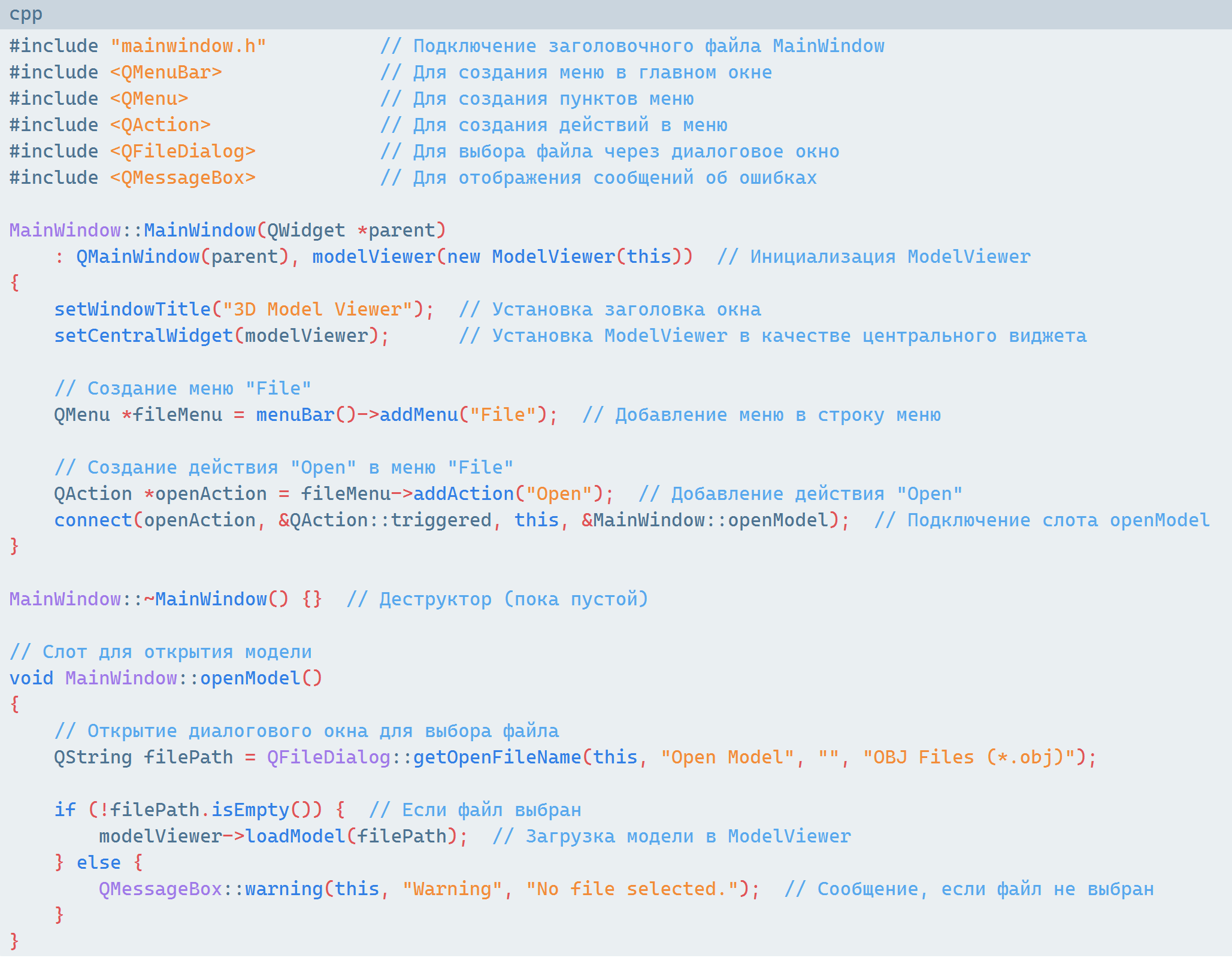
• MainWindow(QWidget \*parent): Конструктор класса, который принимает родительский виджет (по умолчанию nullptr).

• ~MainWindow(): Деструктор класса, который будет вызываться при уничтожении объекта MainWindow.

• openModel(): Слот, который будет вызываться для открытия модели (например, при выборе файла через диалоговое окно).

• modelViewer: Указатель на объект ModelViewer, который отвечает за отображение загруженной модели.

10. mainwindow.cpp



Объяснение кода:

• MainWindow::MainWindow: Конструктор класса, который инициализирует ModelViewer и настраивает главное окно.

• Устанавливается заголовок окна.

• ModelViewer устанавливается в качестве центрального виджета.

• Создается меню "File" с действием "Open", которое подключается к слоту openModel.

• MainWindow::~MainWindow: Деструктор класса (пока пустой).

• openModel: Слот, который открывает диалоговое окно для выбора файла модели.

• Если файл выбран, он передается в ModelViewer для загрузки и отображения.

• Если файл не выбран, выводится предупреждение.