ФКН ВЫСШАЯ ШКОЛА ЭКОНОМИКИ

Программный проект, июнь 2025

# 3D рендер с нуля

**Научный руководитель:**
Садуллаев Музаффар Тимурович,
*приглашенный преподаватель, департамент больших данных и информационного поиска.*

**Работу выполнил:**
Шокаров Тимур Муратович
*Студент БКНАД231*

Яндекс

# Цель проекта

Разобраться с пайплайном для вывода 3d объектов на экран и реализовать его на языке C++

# Задачи проекта

1. Имплементировать графические примитивы
2. Имплементировать алгоритмы преобразования координат относительно камеры
3. Имплементировать алгоритмы отбора видимых объектов
4. Имплементировать алгоритмы растеризации
5. Имплементировать алгоритмы сглаживания полигонов модели
6. Написать приложение, демонстрирующее 3D renderer

# Функциональные требования проекта

1. Получение отрендеренного изображения 3d модели на экране в окне приложения

2. Возможность управления направлением взгляда и положением камеры

# Нефункциональные требования проекта

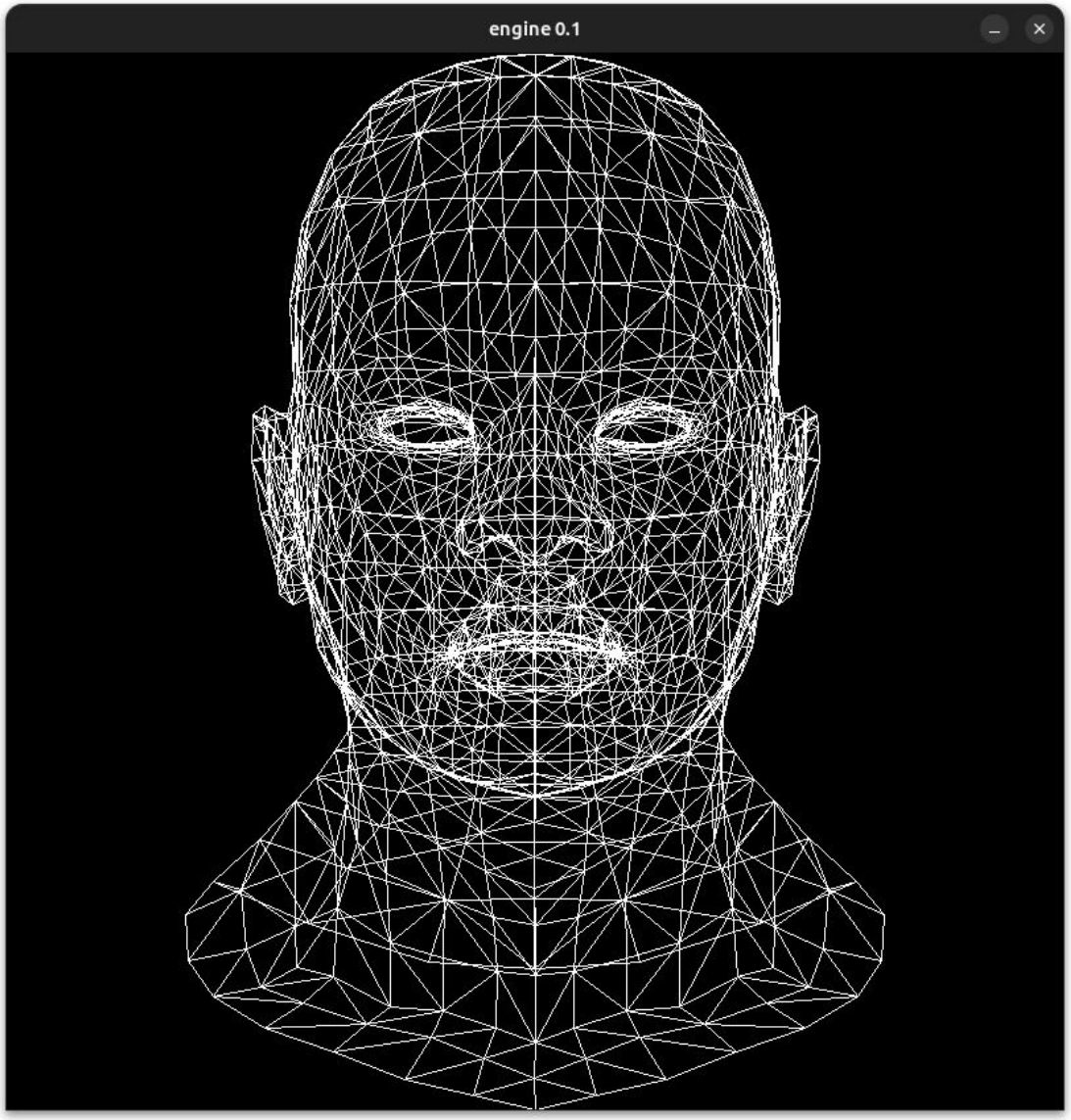
1. Минимизация внешних зависимостей - с нуля дана только возможность открытия окна и установка цвета пикселя с помощью SDL. Вся необходимая математика должна быть самописной, без использования сторонних библиотек

2. Грамотная архитектура проекта и устройство самописных библиотек и классов. Читаемость кода для лучшего понимания устройства написанных алгоритмов - в приоритете

3. Производительность на уровне получения комфортной частоты кадров в секунду при рендеринге тестируемой модели при движении камеры

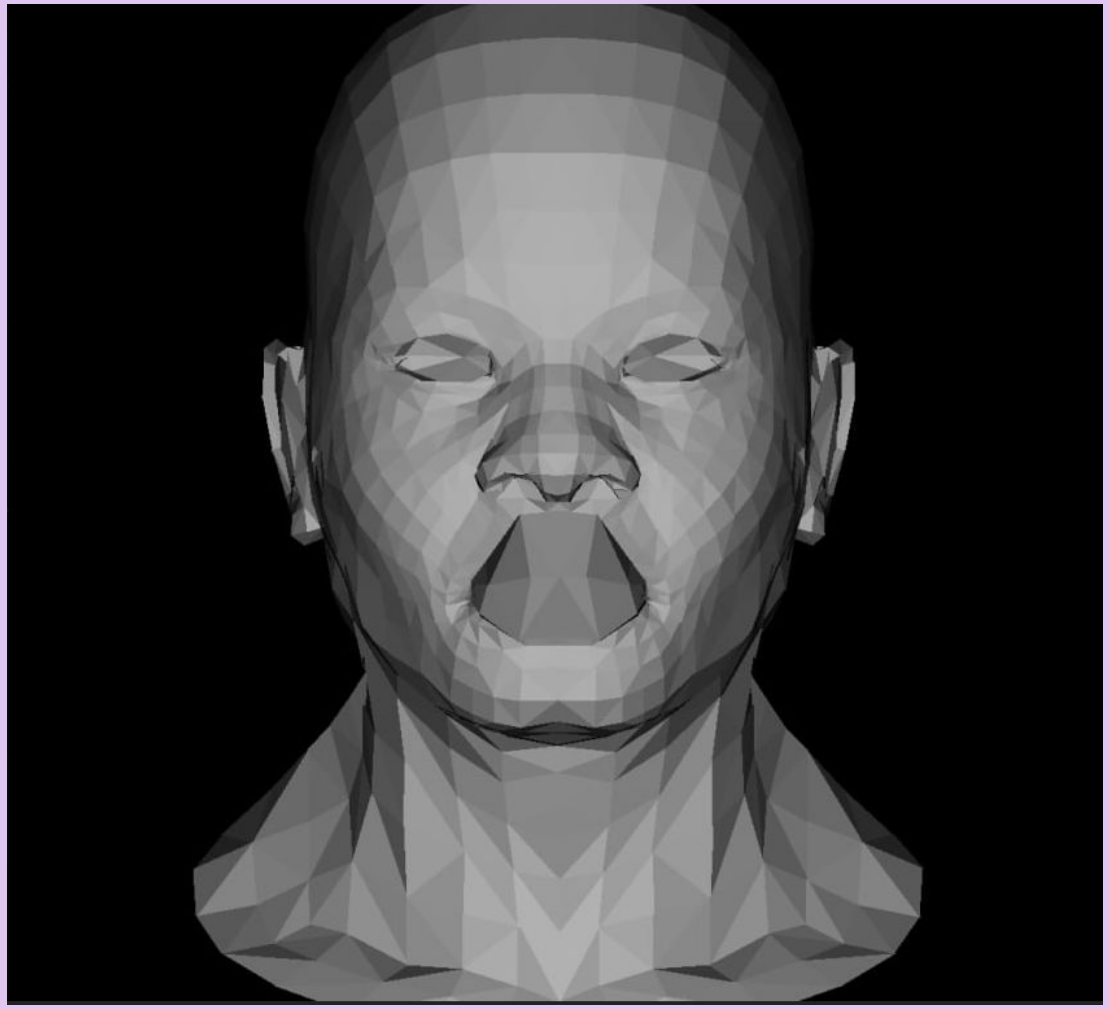
# Эволюция получения изображения

1. Проволочная сетка



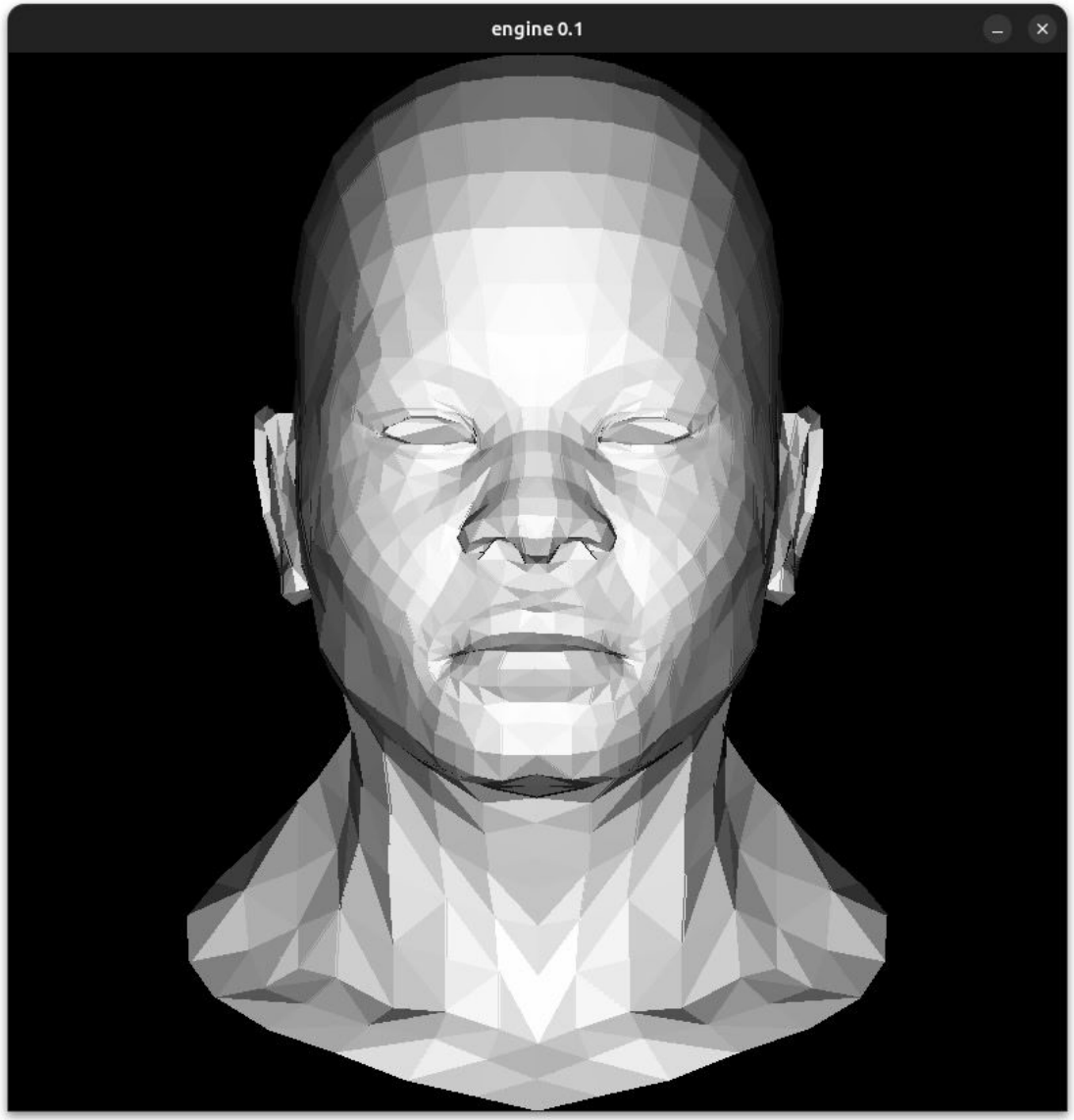
# Эволюция получения изображения

1. Проволочная сетка
2. Раскрашенные полигоны



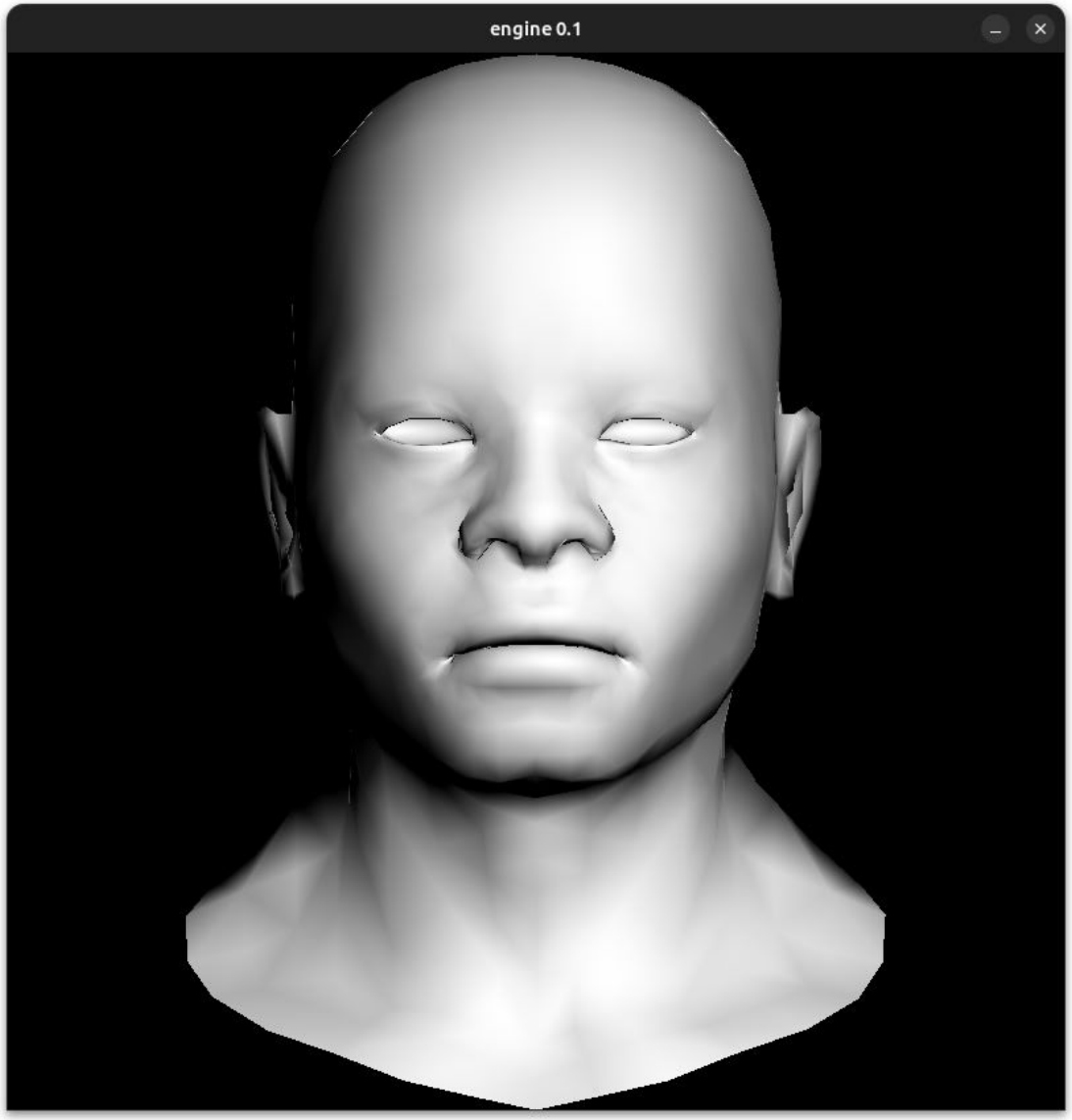
# Эволюция получения изображения

1. Проволочная сетка
2. Раскрашенные полигоны
3. Отсечены невидимые грани



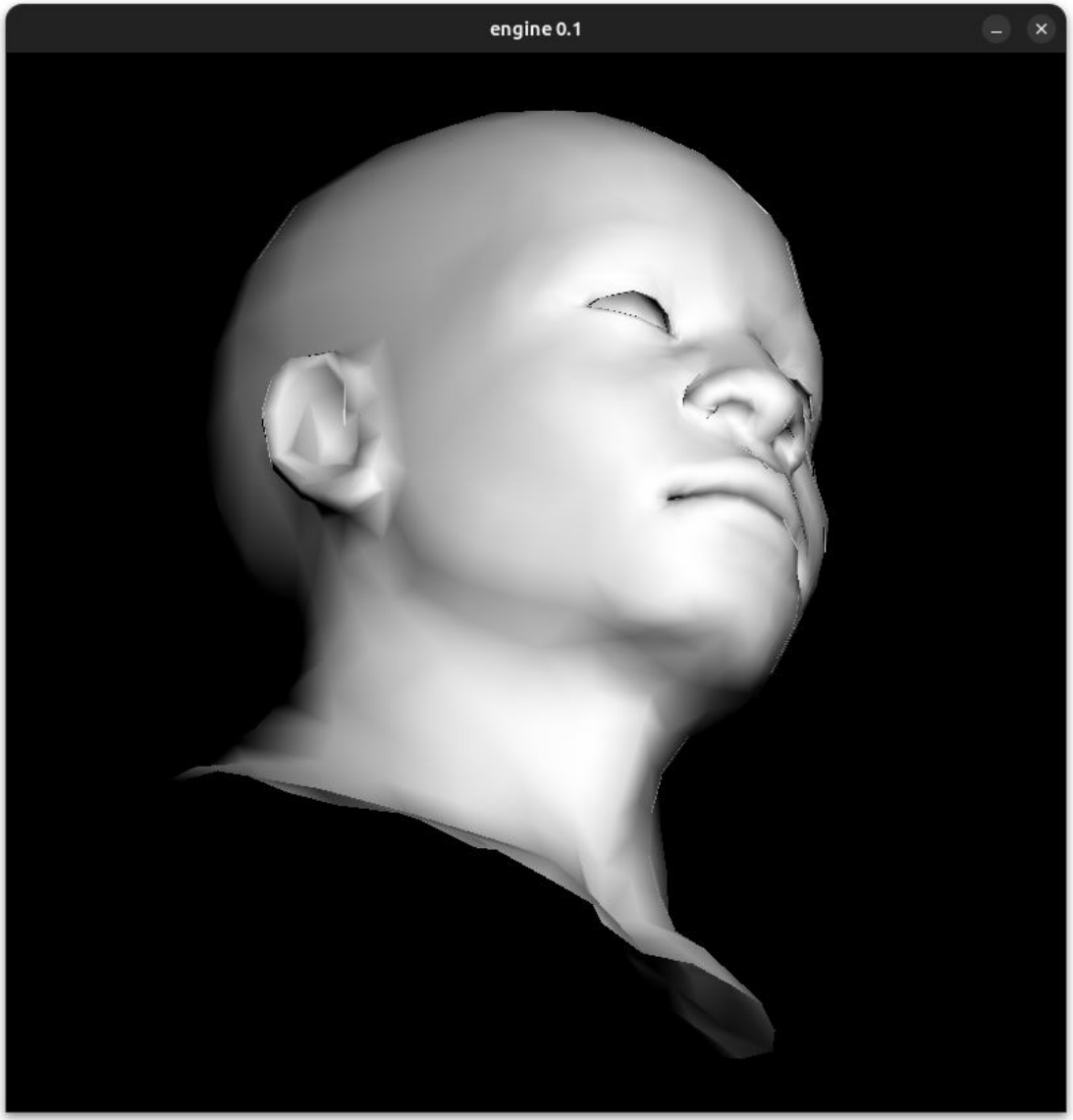
# Эволюция получения изображения

1. Проволочная сетка
2. Раскрашенные полигоны
3. Отсечены невидимые грани
4. Сглаженные полигоны

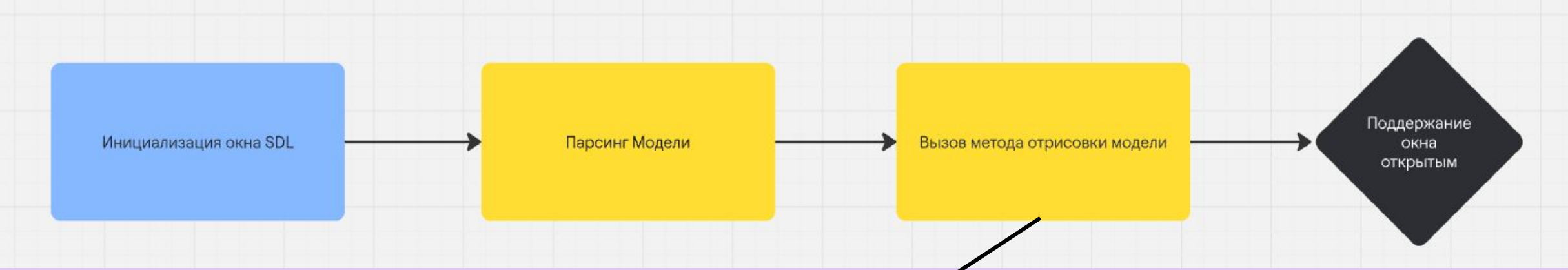


# Эволюция получения изображения

1. Проволочная сетка
2. Раскрашенные полигоны
3. Отсечены невидимые грани
4. Сглаженные полигоны
5. Движение камеры



# Пайплайн вывода на экран

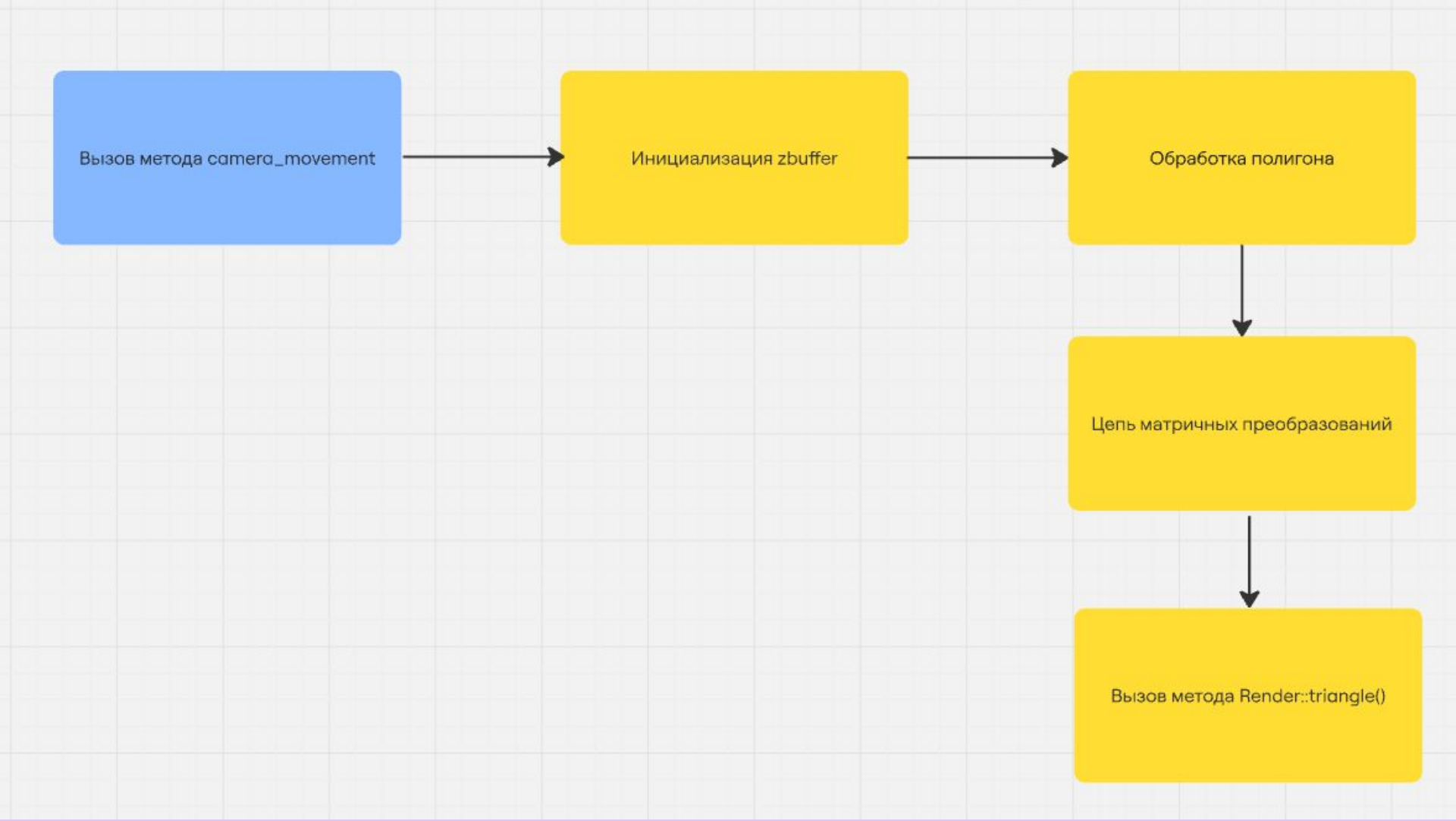



**Вызов метода camera movement** .

Начинаем мы с того, что вызываем метод класса Model со следующей сигнатурой:

```
1   void Model::camera_movement_polygon_smooth(Vec3f eye, Vec3f center, Vec3f up)
```

# Пайплайн внутри метода класса Model

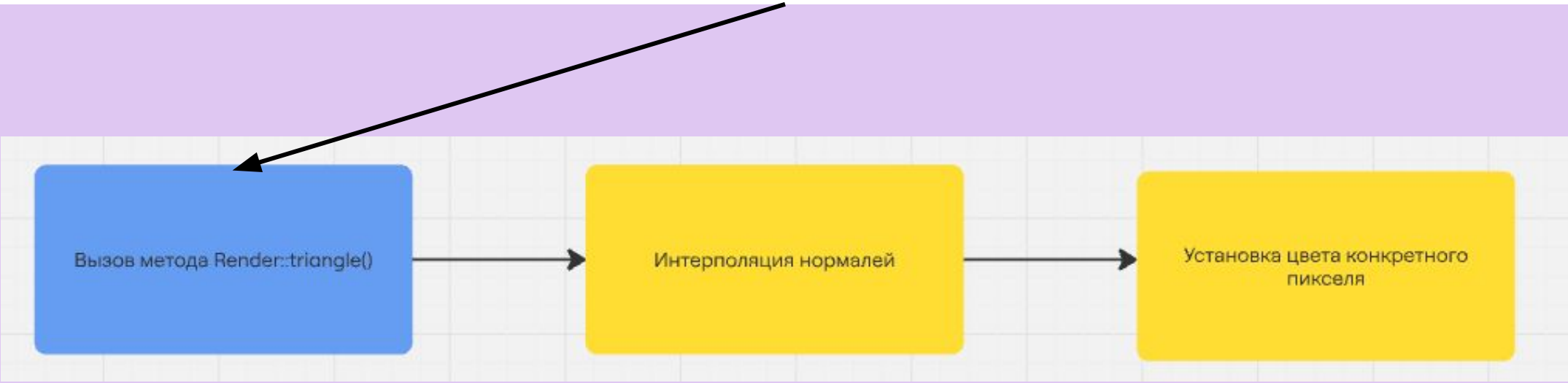


# Внутри метода класса Render:
## растеризация конкретного полигона

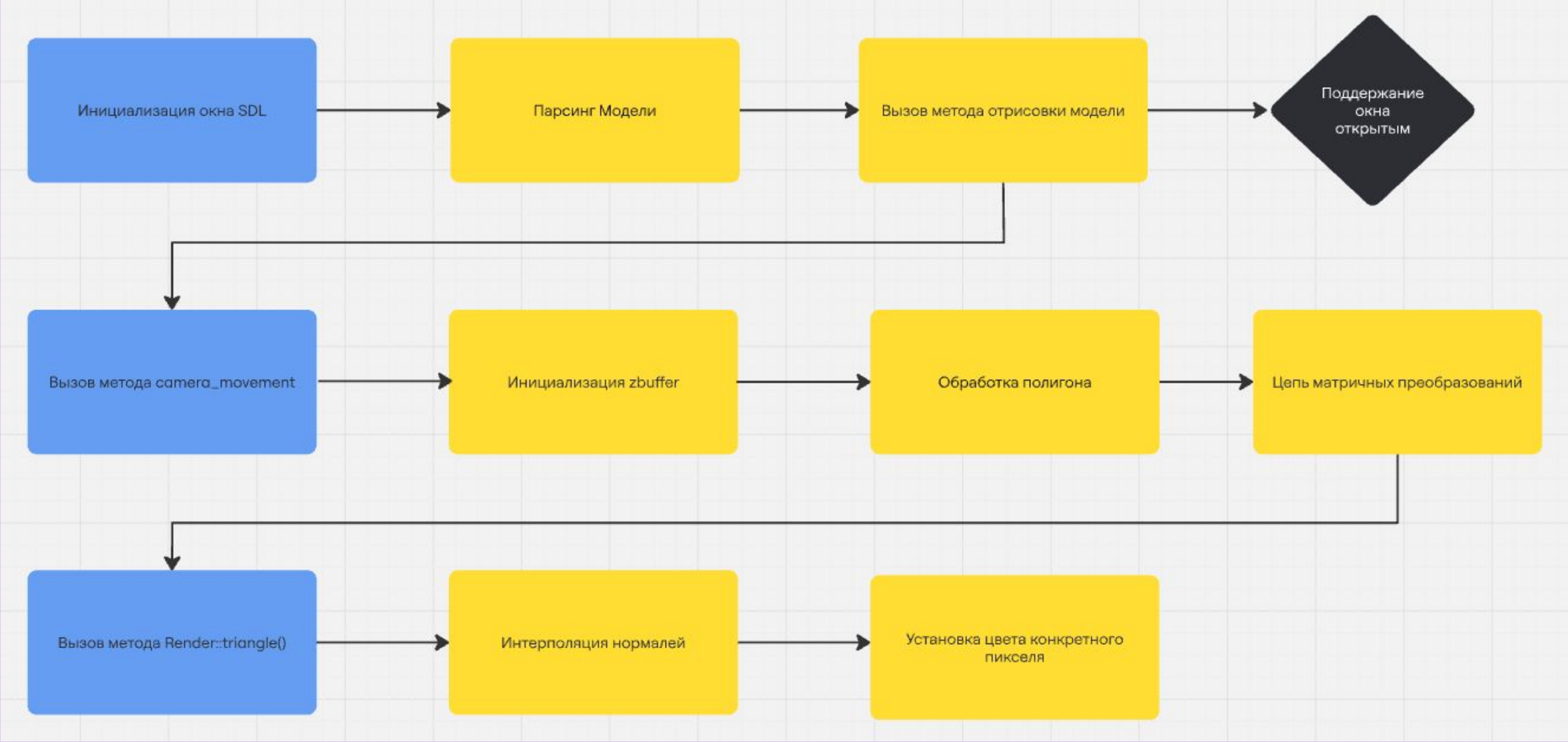
Когда мы вызываем метод класса Render triangle smooth со следующей сигнатурой

```
    void triangle_smooth(Vec3i a, Vec3i b, Vec3i c, std::vector<long double> &zbuffer, Vertex3_normals vertex
```

происходит следующее:

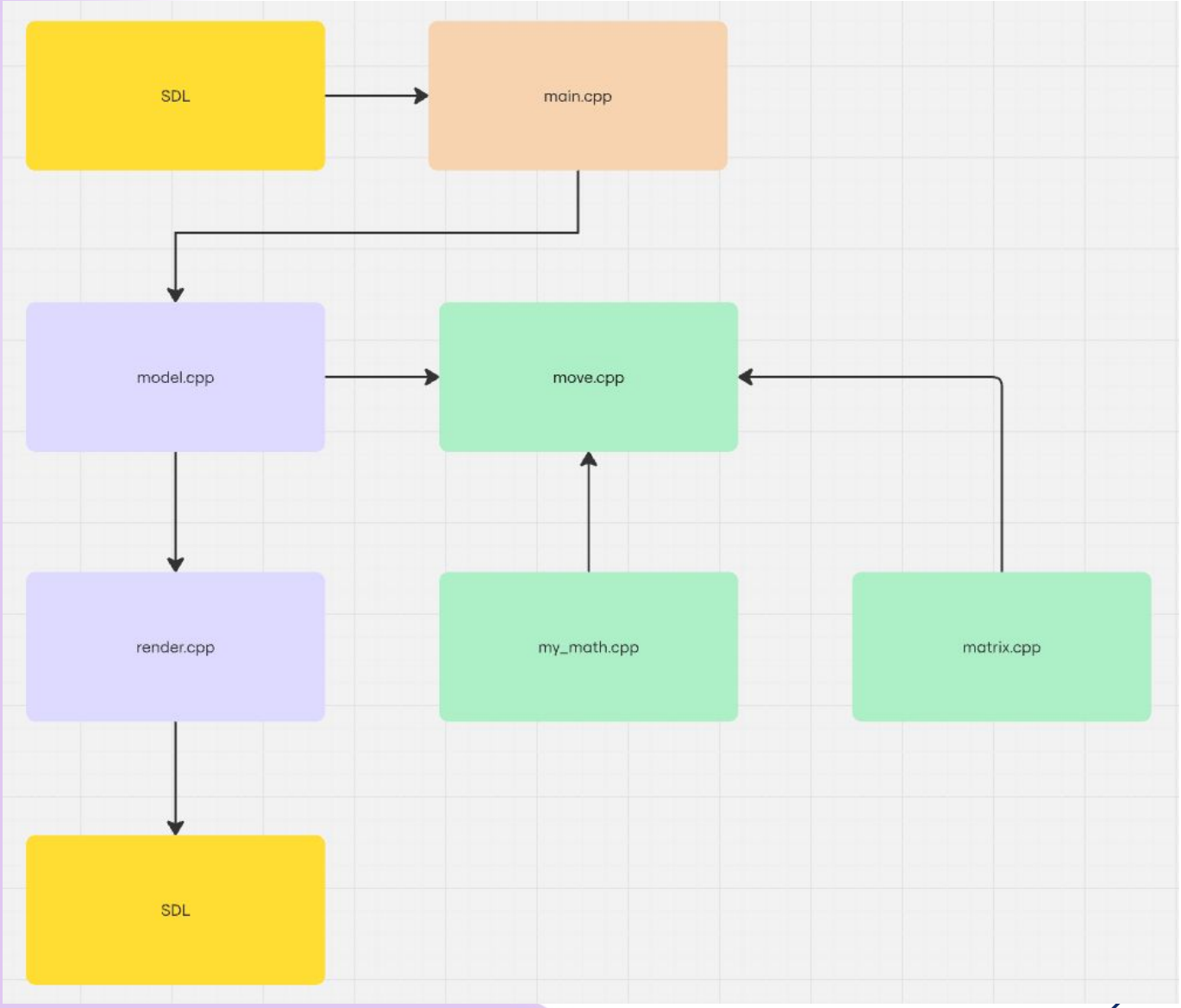


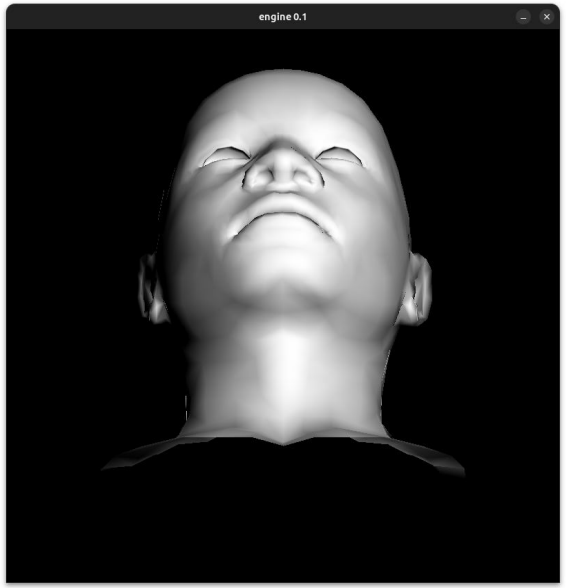
# Итоговый пайплайн приложения

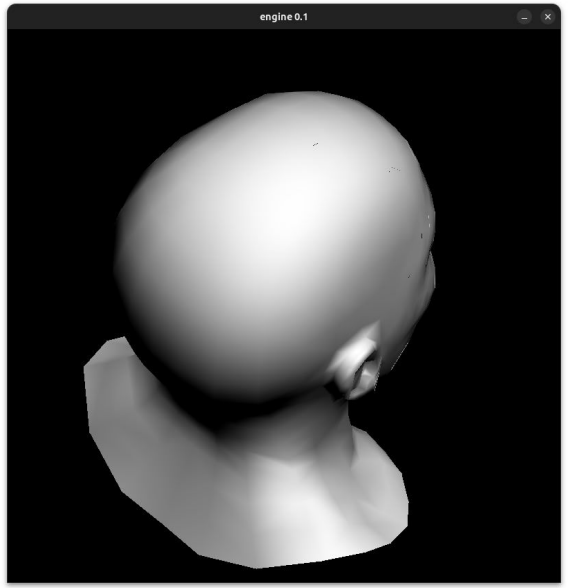


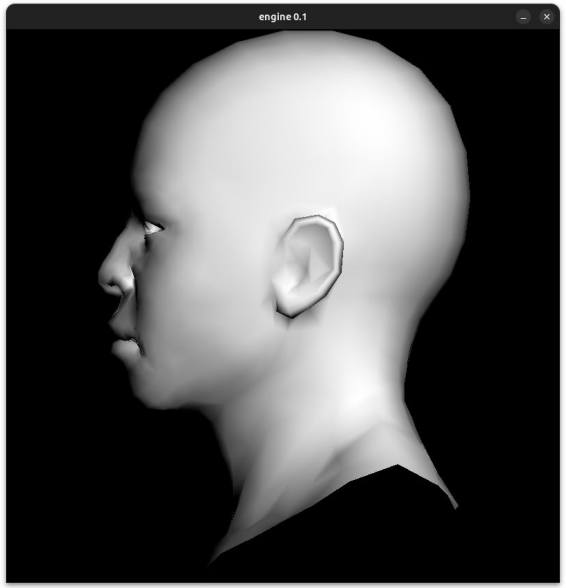
# Архитектура приложения

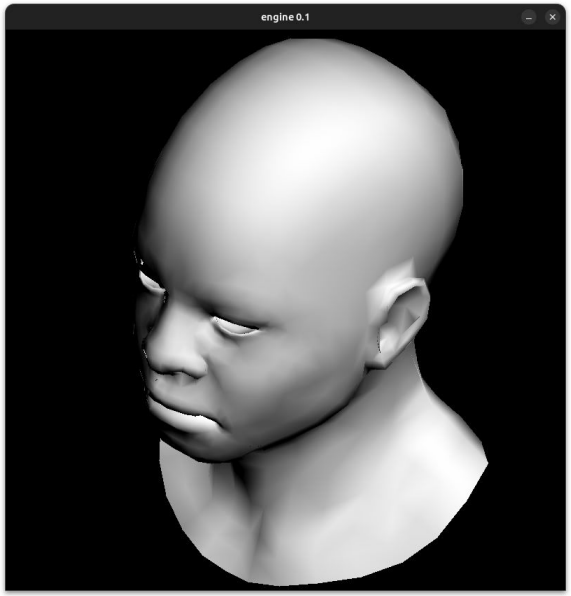
1. В main.cpp исполняется общий пайплайн
2. В model.cpp парсится модель и исполняется пайплайн метода отрисовки
3. В render.cpp происходит растеризация треугольника
4. В move.cpp происходит цепь матричных преобразований координат при движении камеры
5. Все классы используют самописную математику, хранится в my_math.cpp, matrix.cpp и custom_structs.h
6. Минимальное взаимодействие с внешними библиотеками






# Объемные характеристики

1. 892 строк кода
2. 0.08 на получение одного изображения при движении камеры
3. 12 fps
4. 5 различных методов отрисовки моделей

## Итоги работы

1. Имплементированы все предложенные для реализации алгоритмы из компьютерной графики
2. Написано приложение для демонстрации полученного изображения с минимальным количеством зависимостей

## Векторы развития

1. Добавить наложение текстур
2. Добавить тени и шейдеры
3. Добавить user friendly интерфейс для рендеринга полноценных сцен и написания игр
4. Добавить продвинутые алгоритмы из компьютерной графики такие как клиппинг


ОКН