ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук

Образовательная программа «Программная инженерия»

СОГЛАСОВАНО

Кандидат физико-математических наук,
доцент факультета компьютерных наук,
научный руководитель

_____ Д. В. Трушин

«___» _____ 2025 г.

УТВЕРЖДЕНО

Академический руководитель образовательной программы «Программная инженерия», старший преподаватель департамента программной инженерии

_____ Н. А. Павлочев

«___» _____ 2025 г.

Подп. и дата	
Инв.№ дубл.	
Взам. инв.№	
Подп. и дата	
Инв.№ подл	

3D RENDERER С НУЛЯ

Пояснительная записка

ЛИСТ УТВЕРЖДЕНИЯ

RU.17701729.05.01-01 12 01-1-ЛУ

Исполнитель:

Студент группы БПИ232

_____ / А. Д. Носов /

«___» _____ 2025 г.

2025

УТВЕРЖДЕН

RU.17701729.05.01-01 12 01-1-ЛУ

3D RENDERER С НУЛЯ

Пояснительная записка

RU.17701729.05.01-01 12 01-1

Листов 22

Инв.№ подл	Подп. и дата	Взам. инв.№	Инв.№ дубл.	Подп. и дата

2025

АННОТАЦИЯ

Настоящий программный документ представляет собой пояснительную записку к программному проекту «3D Renderer с нуля».

Раздел «Введение» включает в себя наименование программы и документ, на основании которого ведётся разработка, с указанием организации, утвердившей данный документ.

В разделе «Назначение и область применения» содержатся функциональное и эксплуатационное назначение программы и краткая характеристика области её применения.

В разделе «Технические характеристики» содержатся следующие подразделы: постановка задачи на разработку программы, описание функционирования программы, описание функциональных особенностей, описание и обоснование выбора метода организации входных и выходных данных, описание и обоснование выбора состава технических и программных средств.

В разделе «Ожидаемые технико-экономические показатели» указана предполагаемая потребность и экономические преимущества разработки по сравнению с отечественными и зарубежными образцами или аналогами.

Настоящий документ разработан в соответствии с требованиями:

1. ГОСТ 19.101-77 [1]: Виды программ и программных документов.
2. ГОСТ 19.102-77 [2]: Стадии разработки.
3. ГОСТ 19.103-77 [3]: Обозначения программ и программных документов.
4. ГОСТ 19.104-78 [4]: Основные надписи.
5. ГОСТ 19.105-78 [5]: Общие требования к программным документам.
6. ГОСТ 19.106-78 [6]: Требования к программным документам, выполненным печатным способом.
7. ГОСТ 19.404-79 [10]: Пояснительная записка. Требования к содержанию и оформлению.

Изменения к данному Техническому заданию оформляются согласно ГОСТ 19.603-78 [12], ГОСТ 19.604-78 [13].

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

1.1. Наименование программы

Наименование программы — «3D Renderer».

Наименование программы на английском языке — «3D Renderer».

1.2. Документ, на основании котрого введется разработка

Разработка ведется на основании учебного плана подготовки бакалавров по направлению 09.03.04 «Программная инженерия» и утвержденной академическим руководителем программы темы курсового проекта.

2. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

2.1. Назначение программы

2.1.1. Функциональное назначение

Программа представляет собой интерактивное приложение, демонстрирующее процесс отрисовки 3D-сцены на двумерном экране. Оно позволяет пользователю управлять камерой и наблюдать изменения сцены.

Для обеспечения работоспособности приложения выделяются две основные части:

1. Библиотечный код: отвечает за реализацию пайплайна рендеринга для отрисовки 3D-сцены.
2. Приложение: предназначено для использования библиотеки и построения интерактивного интерфейса для управления сценой.

2.1.2. Эксплуатационное назначение

«3D Renderer» предназначен для демонстрации корректной работы созданного алгоритма визуализации трёхмерных сцен.

Целевая аудитория проекта — разработчики, исследующие основы 3D-рендеринга, а также студенты и специалисты, изучающие компьютерную графику. Программа будет полезна для тех, кто хочет понять процессы, стоящие за визуализацией трехмерных объектов, и улучшить свои навыки в разработке графических приложений.

2.2. Краткая характеристика области применения

«3D Renderer» — образовательный проект, позволяющий понять основные этапы отрисовки 3D объектов.

Программа является интерактивной и позволяет пользователю управлять камерой, наблюдая изменение сцены. Управление осуществляется с помощью клавитатуры и мыши.

3. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

3.1. Постановка задачи на разработку

Вся программа делится на две части: движок, реализующий основные алгоритмы рендеринга, и приложение, обеспечивающее пользовательский интерфейс и взаимодействие с движком.

1) Функции движка.

Задача движка – генерировать двумерное изображение, отрисовывая сцену с загруженными объектами с учетом положения и направления камеры.

Для этого движок реализует следующие функции:

1. Преобразование объектов
   - Представление объекта в виде набора треугольников;
   - Преобразование координат из локального пространства объекта в глобальное;
   - Преобразование из глобального пространства в пространство камеры;
   - Преобразование в экранное пространство с применением проективного преобразования.
2. Растеризация
   - Обработка примитивов:
     - Отсекание частей треугольников, выходящих за пределы экрана;
     - Нормализация координат треугольников в экранном пространстве;
     - Преобразование треугольников в пиксели;
     - Вычисление значений цвета.
3. Тест глубины для добавления пикселя в буфер.
4. Освещение
   - Реализация базовых моделей освещения для визуализации трёхмерных объектов:
     - Поддержка освещения от направленных и рассеянных источников света.
   - Обеспечение плавности отображения геометрии:
     - Интерполяция нормалей и цвета для создания эффекта гладких поверхностей.

2) Функции приложения.

Основная задача приложения — отображение пользователю экрана с отрисованной сценой и обработка событий взаимодействия с пользователем.

Приложение включает следующие функции:

1. Отображение сцены.

2. Добавление 3D-объектов на сцену.
3. Интерактивное взаимодействие:
   – Обработка пользовательского ввода через клавиатуру и мышь;
   – Управление камерой для изменения положения и направления просмотра сцены.
4. Настройка освещения: изменение параметров источников света, например, таких как интенсивность, цвет, направление.

Приложение предоставляет два окна для взаимодействия:

1. Окно настроек:
   – предназначено для управления параметрами сцены (настройка освещения, добавление объектов).
2. Окно рендеринга:
   – отображает результат работы рендерера.

3.2. Описание применяемых математических методов

Для преобразования 3D-объектов в двумерное изображение используется линейная алгебра, в т. ч. ортогональные и проективные преобразования и переносы. Методы взяты из книги «Mathematics for 3D game programming and computer graphics» Эрика Лендьеля.

1. Ортогональные преобразования.

Ортогональные преобразования используются для изменения ориентации объектов в пространстве, в частности для поворота объектов вокруг собственных осей, вокруг глобальных осей координат и вокруг осей координат камеры.

Повороты в трехмерном пространстве задаются следующими матрицами

– Поворот вокруг оси X на угол φ:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & -\sin(\varphi) \\ 0 & \sin(\varphi) & \cos(\varphi) \end{pmatrix}$$

– Поворот вокруг оси Y на угол φ:

$$\begin{pmatrix} \cos(\varphi) & 0 & \sin(\varphi) \\ 0 & 1 & 0 \\ -\sin(\varphi) & 0 & \cos(\varphi) \end{pmatrix}$$

– Поворот вокруг оси Z на угол φ:

$$\begin{pmatrix} \cos(\varphi) & -\sin(\varphi) & 0 \\ \sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

– Поворот вокруг произвольной оси на угол φ:

Пусть произвольная ось задается единичным вектором $u = (u_x, u_y, u_z)$. Тогда матрица поворота вокруг этой оси на угол φ имеет вид:

$$\begin{pmatrix} \cos(\varphi) + u_x^2(1-\cos(\varphi)) & u_x u_y(1-\cos(\varphi)) - u_z\sin(\varphi) & u_x u_z(1-\cos(\varphi)) + u_y\sin(\varphi) \\ u_y u_x(1-\cos(\varphi)) + u_z\sin(\varphi) & \cos(\varphi) + u_y^2(1-\cos(\varphi)) & u_y u_z(1-\cos(\varphi)) - u_x\sin(\varphi) \\ u_z u_x(1-\cos(\varphi)) - u_y\sin(\varphi) & u_z u_y(1-\cos(\varphi)) + u_x\sin(\varphi) & \cos(\varphi) + u_z^2(1-\cos(\varphi)) \end{pmatrix}$$

2. Переносы.

Переносы используются для изменения положения объектов в пространстве.

Перенос в трехмерном пространстве задается трехмерным вектором.

– Вектор переноса на s_x по оси X, s_y по оси Y, s_z по оси Z:

$$\begin{pmatrix} s_x \\ s_y \\ s_z \end{pmatrix}$$

Для удобства комбинирования нескольких преобразований в программе поворот и перенос объединены в одно преобразование, представляемое в виде матрицы 4x4 следующим образом:

$$\begin{pmatrix} R_{00} & R_{01} & R_{02} & s_x \\ R_{10} & R_{11} & R_{12} & s_y \\ R_{20} & R_{21} & R_{22} & s_u \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Где R – матрица поворота, s – вектор переноса.

3. Представление векторов точек и напралений.

Для корректности преобразований с использованием матриц 4x4 зададим хранение векторов точек и векторов направлений следующим образом:

– Векторы точек. Первые 3 координаты сохраняются и добавляется четвертая координата 1:

$$P = \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}$$

– Векторы направлений. Первые 3 координаты сохраняются и добавляется четвертая координата 0:

$$V = \begin{pmatrix} V_x \\ V_y \\ V_z \\ 0 \end{pmatrix}$$

При таком хранении получаем следующее изменение векторов при применении к ним комбинированной матрицы поворота и переноса:

– Для точки:

$$P' = \begin{pmatrix} R_{00} & R_{01} & R_{02} & s_x \\ R_{10} & R_{11} & R_{12} & s_y \\ R_{20} & R_{21} & R_{22} & s_u \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix} = \begin{pmatrix} R_{00}P_x + R_{01}P_y + R_{02}P_z + s_x \\ R_{10}P_x + R_{11}P_y + R_{12}P_z + s_y \\ R_{20}P_x + R_{21}P_y + R_{22}P_z + s_z \\ 1 \end{pmatrix} = \begin{pmatrix} RP + s \\ \\ 1 \end{pmatrix}$$

Можно заметить, что преобразование согласуется со стандартным домножением на матрицу поворота и добавлением вектора переноса.

– Для вектора направления:

$$V' = \begin{pmatrix} R_{00} & R_{01} & R_{02} & s_x \\ R_{10} & R_{11} & R_{12} & s_y \\ R_{20} & R_{21} & R_{22} & s_u \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ V_z \\ 0 \end{pmatrix} = \begin{pmatrix} R_{00}V_x + R_{01}V_y + R_{02}V_z \\ R_{10}V_x + R_{11}V_y + R_{12}V_z \\ R_{20}V_x + R_{21}V_y + R_{22}V_z \\ 0 \end{pmatrix} = \begin{pmatrix} RV \\ \\ 0 \end{pmatrix}$$

Можно заметить, что преобразование согласуется со стандартным домножением на матрицу поворота, а перенос не учитывается у вектора направления.

4. Объекты.

Вектор точки и вектор направления нормали к точке вместе составляют вершину. Три вершины составляют треугольник.

Объект хранит в себе треугольники в локальной системе координат. А также 4x4 матрицу поворота и переноса M_{model} . Домножением векторов точек и векторов нормалей на данную матрицу можно получить координаты точек объекта и нормалей к этим точкам в глобальной системе координат:

$$P_{\text{global}} = M_{\text{model}} \cdot P_{\text{local}}$$

$$V_{\text{global}} = M_{\text{model}} \cdot V_{\text{local}}$$

5. Переход в пространство камеры.

Все объекты необходимо перевести в пространство камеры (view space). В этом пространстве ось z направлена в противоположную сторону от направления камеры, ось x направлена вправо относительно камеры, ось y направлена вверх относительно камеры.

Это преобразование задается матрицей M_{view}.

– Для точек достаточно домножить на матрицу слева:

$$P_{\text{view}} = M_{\text{view}} \cdot P_{\text{global}}$$

– Для векторов направлений необходимо домнажать на $\left(M_{\text{view}}{}^{-1}\right)^T$ для сохранения для сохранения ортогональности вектору поверхности

$$V_{\text{view}} = \left(M_{\text{view}}{}^{-1}\right)^T \cdot V_{\text{global}}$$

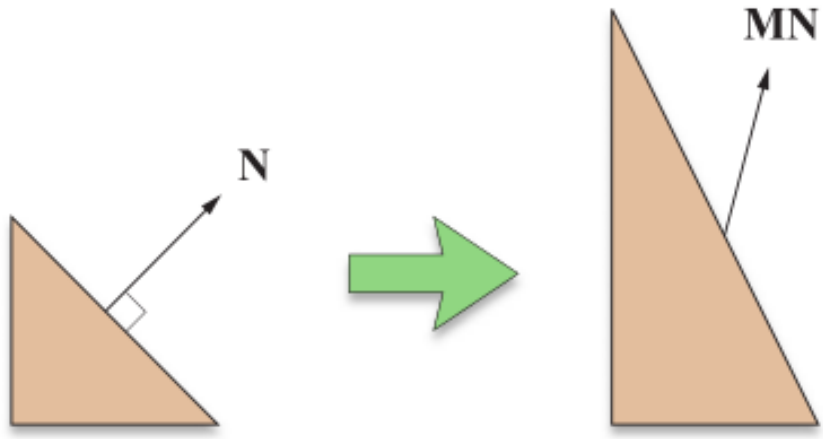

Figure 4.6. Transforming a normal vector **N** with a nonorthogonal matrix **M**.

Однако, если мы знаем, что верхний левый блок 3x3 матрицы M_{view} является ортогональной матрицей, можно упростить это преобразование, учитывая, что четвертая координата V_{global} равна 0:

$$V_{\text{view}} = M_{\text{view}} \cdot V_{\text{global}}$$

6. Составление матрицы M_{view} и обработка перемещений камеры.

Перемещение камеры на заданный вектор можно рассматривать как перемещение всех объектов сцены на противоположный вектор. Поворот камеры вокруг заданной оси на заданный угол можно рассматривать как поворот всех объектов сцены вокруг той же оси на противоположный угол.

Таким образом можно задать следующий алгоритм получения M_{view}:

– Изначально $M_{\text{view}} = I_4$;

– При переносе камеры на вектор s домножаем M_{view} на матрицу 4x4 переноса на вектор $-s$:

$$M_{\text{view}} = \text{Translate}(-s) \cdot M_{\text{view}}$$

– При повороте камеры вокруг оси A на угол φ домножаем M_{view} на матрицу 4x4 поворота вокруг оси A на угол $-\varphi$:

$$M_{\text{view}} = \text{Rotate}_A(-\varphi) \cdot M_{\text{view}}$$

7. Область видимости камеры

Область видимости камеры представляет собой усеченную пирамиду зрения.

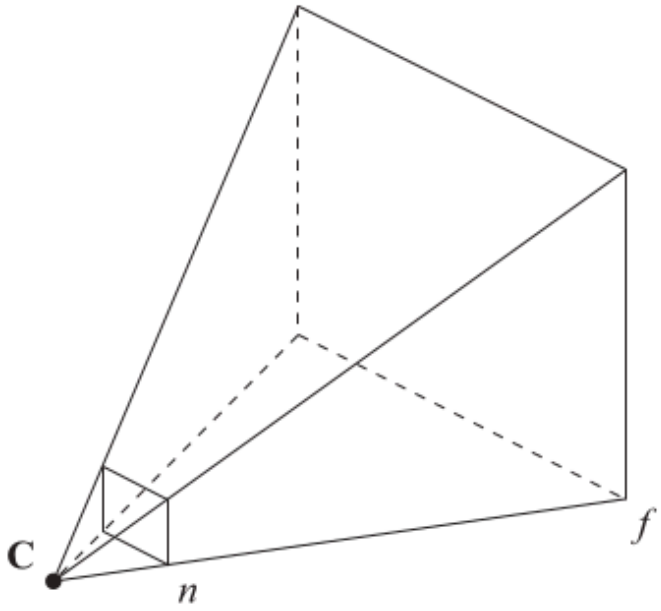

Figure 5.8. The view frustum encloses the space bounded by the near plane lying at a distance *n* from the camera, the far plane lying at a distance *f* from the camera, and four side planes that pass through the camera position **C**.

Камера находится в точке C. Все, что что находится на расстоянии ближе n и дальше f, камера не видит. Усеченная пирамида от n до f – область видимости камеры.

Для дальнейших действий (клиппинг и растеризации) удобно перейти к кубу зрения.

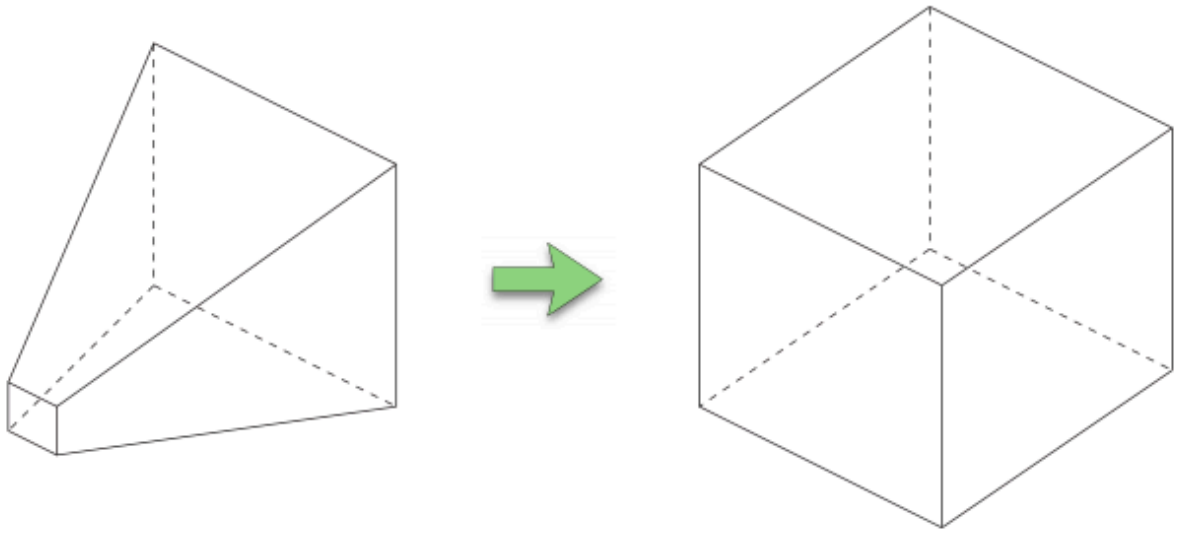

Figure 5.15. The perspective projection maps the view frustum to the cube representing homogeneous clip space.

Это реализуется с помощью матрицы перспективной проекции:

$$M_{\text{proj}} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2n}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Где t, b, l, r – координаты верхней, нижней, левой и правой граней основания отсеченной пирамиды соответственно.

Их можно задать с центром координат в центре экрана, зная угол обзора камеры fov и соотношение сторон экрана aspect (соотношение сторон основания отсеченной пирамиды):

$$t = n \tan\left(\frac{\text{fov}}{2}\right)$$

$$b = -t$$

$$r = t \cdot \text{aspect}$$

$$l = -r$$

С данным центром координат матрица немного упрощается:

$$M_{\text{proj}} = \begin{pmatrix} \frac{n}{r} & 0 & 0 & 0 \\ 0 & \frac{n}{t} & 0 & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2n}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

8. Отсечение невидимых частей треугольников.

Данное проективное преобразование таково, что после него все точки внутри куба имеют следующий инвариант:

$$-w \leq x \leq w$$

$$-w \leq y \leq w$$

$$-w \leq z \leq w$$

Данным образом можно проверить, находится ли точка в кубе видимости.

Для каждой грани можно задать функцию, которая неотрицательна, когда точка внутри куба, и отрицательна иначе. Например, для левой плоскости:

$$f(v) = v_x + v_w$$

$$f(v) < 0 \iff v_x + v_w < 0 \iff v_x < -v_w \implies \text{Точка не лежит внутри куба.}$$

$$\text{Точка лежит внутри куба} \implies v_x \geq -v_w \iff v_x + v_w \geq 0 \iff f(v) \geq 0$$

Если ни одна вершина не попала в область видимости – треугольник отрисовывать не нужно.

Если все вершины треугольника попали в область видимости – треугольник нужно отрисовывать.

Если есть вершины треугольника, которые находятся внутри области видимости, и есть вершины, которые находятся области видимости – необходимо разбить данный треугольник на несколько полностью попадающих в область видимости и отрисовать их.

9. Нахождение пересечения ребра треугольника и грани.

Можно задать уравнение отрезка между двумя вершинами треугольника, одна из которых попадает в область, а вторая – нет:

$$v(t) = v_1 + t(v_2 - v_1)$$

$$t \in [0, 1]$$

Пусть $f(v)$ – функция грани, которую пересекает данный отрезок. Эта функция линейна.

$f(v_1)$ и $f(v_2)$ имеют разный знак. И $f(v) = 0$ значит, что точка лежит на грани.

Можно найти t, используя линейность:

$$f(v(t)) = 0$$

$$f(v_1 + t(v_2 - v_1)) = 0$$

$$f(v_1) + t(f(v_2) - f(v_1)) = 0$$

$$t = \frac{f(v_1)}{f(v_1) - f(v_2)}$$

Зная t, можно найти точку пересечения и, если нужно, интерполированный вектор нормали в этой точке.

10. Вычисление цвета в точке с использованием модели Блинна-Фонга.

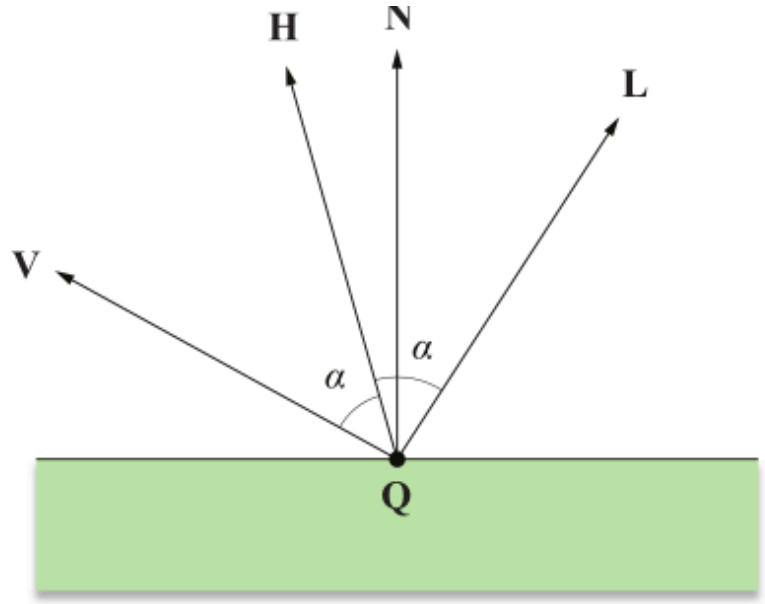

Figure 7.5. The angle between the normal vector **N** and the halfway vector **H** can also be used to determine specular intensity.

V – вектор от точки к камере.

L – вектор от точки к источнику света.

N – нормаль к поверхности в точке.

$H = \frac{L+V}{\|L+V\|}$

Векторы V, L, N можно интерполировать для каждой точки внутри треугольника по соответствующим значениям векторов в вершинах.

Также для вычисления цвета нужно знать параметры материала:

k_a – ambient коэффициент для базового освещения.

k_d – diffuse коэффициент для рассеянного отражения.

k_s – specular коэффициент для зеркального отражения.

n – коэффициент блеска.

И параметры рассеянного и направленного света:

I_{ambient} – яркость окружающего освещения. I_{direct} – яркость направленного освещения.

1) Ambient компонента:

$$I_a = k_a \cdot I_{\text{ambient}}$$

2) Diffuse компонента:

$$I_d = k_d \cdot I_{\text{direct}} \cdot \max(0, N \cdot L)$$

3) Specular компонента:

$$I_s = k_s \cdot I_{\text{direct}} \cdot \max(0, N \cdot H)^n$$

Итоговый цвет:

$$I_{\text{final}} = I_a + I_d + I_s$$

3.3. Описание алгоритма работы программы

В приложении запускается главный цикл, в котором происходит обработка сигналов с клавиатуры и мыши. В зависимости от сигналов с клавиатуры и мыши меняется положение и ориентация в пространстве камеры. Перемещение камеры на заданный вектор реализовано как перемещение всех объектов сцены на противоположный вектор. Поворот камеры вокруг заданной оси на заданный угол реализовано как поворот всех объектов сцены вокруг той же оси на противоположный угол. После оработки событий, приложение обращается к рендереру за отрисовкой кадра.

Рендерер работает по следующему алгоритму:

– Формирование одной матрицы динамического размера 4xN, содержащей все позиции вершин объекта;
– Формирование одной матрицы динамического размера 4xN, содержащей все векторы нормалей вершин объекта;

– Преобразование позиций вершин в клипповое пространство домножением матрицы вершин последовательно на матрицу перехода из локальной системы координат к глобальной M_{view}, затем на матрицу перехода в пространство камеры M_{view} и после на матрицу проекции M_{proj};
– Преобразование нормалей вершин в пространство камеры домножением матрицы вершин последовательно на матрицу перехода из локальной системы координат к глобальной M_{view}, затем на матрицу перехода в пространство камеры M_{view};
– Определение видимости каждой позиции вершины;
– Отсечение невидимых фрагментов треугольников, получение новых треугольников, лежащих целиком в области видимости с интерполяцией нормалей в новых вершинах;
– Перевод позиций вершин в normalized device coordinates делением на w координату;
– Отрисовка треугольников, попавших в область видимости с использованием метода Блинна-Фонга для сглаживания поверхности.

3.4. Обоснование выбора схемы алгоритма решения задачи

Логика приложения отделена от рендерера. Это позволяет проще поддерживать и изменять каждую из частей программы.

Использование постоянного цикла в приложении является стандартной практикой в разработке интерактивных приложений.

Алгоритм перемещения и поворота камеры был выбран в силу простоты реализации и в силу отсутствия необходимости хранить дополнительные данные помимо матрицы преобразования.

Формирование большой матрицы динамического размера позволяет одним умножением на матрицу преобразования получить результат. Библиотека Eigen задействует SIMD, что должно ускорять обработку большого числа вершин.

Алгоритм Блинна-Фонга обеспечивает плавное затенение и добавляет блики объекту, что придает реализма изображению.

Язык C++ был выбран в связи со своей скоростью выполнения.

3.5. Описание и обоснование выбора метода организации входных и выходных данных

Управление с помощью клавиш WASD и мыши было выбрано в силу своей популярности в качестве элементов управления камерами в 3D играх от первого лица.

Формат вывода изображения в формате RGB выбран в силу распространенности данного стандарта.

Формат загрузки объектов в формате .obj файлов был выбран в силу простоты восприятия и популярности данного формата.

3.6. Описание и обоснование выбора состава технических средств

3.6.1. Состав технических и программных средств

Программа реализована на C++.

– Версия стандарта — C++ 20;
– Система сборки — CMake версии 3.30.

Используются утилиты, контролирующие кодстайл:

– clang-format версии 16.0;
– clang-tidy версии 16.0.

Используется система git в качестве системы контроля версий.

Используются следующие библиотеки:

– Eigen версии 3.4.0 для векторно-матричных вычислений и связанных с ними операций;
– SFML версии 3.0.0 для отрисовки, работы с окном и вводом пользователя.

Для запуска и функционирования программы необходимо соответствие следующим системным требованиям:

– Дисплей с разрешением, не меньшим 1280 x 720 пикселей;
– Клавиатура, компьютерная мышь;
– Оперативная память объемом не менее 4 ГБ;
– Многоядерный процессор с тактовой частотой не менее 2.4 ГГц;
– Операционная система Fedora Workstation 40 или более новые версии, либо Windows 10 или более новые версии;
– Хранилище со свободным объемом не менее 100 МБ.

3.6.2. Обоснование выбора состава технических средств

Язык C++20 позволяет писать безопасный value semantics код без использования устаревших практик и сложного лишнего кода.

CMake является стандартной системой сборки для C++ проектов.

Clang-format и clang-tidy являются стандартом в индустрии для работы с кодом для такого сложного языка как C++. Без этих утилит невозможно добиться высокого качества кода. Утилиты позволяют автоматизировать многие рутинные процессы.

Git является стандартом в индустрии для работы с версиями кода.

Библиотека Eigen является одной из лучших для работы с объектами линейной алгебры, содержит много оптимизированных алгоритмов как асимптотически, так и на уровне поддержки специальных процессорных инструкций вроде SIMD. Это одна из библиотек, используемых в качестве первого выбора.

Библиотека SFML выбрана в силу своей легкости. Для реализации несложного приложения нужен небольшой набор методов взаимодействия с окном. И SFML идеально подходит для этого.

Клавиатура и мышь нужны для управления камерой.

При несоблюдении требований на процессор и/или оперативную память программа может работать нестабильно и зависать.

Хранилище необходимо для хранения файлов формата .obj.

4. ОЖИДАЕМЫЕ ТЕХНИКО-ЭКОНОМИЧЕСКИЕ ПОКАЗАТЕЛИ

4.1. Ориентировочная экономическая эффективность

В рамках данной работы расчёт экономической эффективности не предусмотрена.

4.2. Предполагаемая потребность

Данный программный продукт будет полезен пользователям, интересующимся созданием и изучением визуализации 3D-сцен.

4.3. Экономические преимущества разработки по сравнению с отечественными и зарубежными аналогами

В рамках данного задания экономические преимущества по сравнения с отечественными и зарубежными аналогами не предусмотрена.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. ГОСТ 19.101-77: Виды программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
2. ГОСТ 19.102-77: Стадии разработки. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
3. ГОСТ 19.103-77: Обозначения программ и программных документов. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
4. ГОСТ 19.104-78: Основные надписи. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
5. ГОСТ 19.105-78: Общие требования к программным документам. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
6. ГОСТ 19.106-78: Требования к программным документам, выполненным печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
7. ГОСТ 19.201-78: Техническое задание. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
8. ГОСТ 19.301-79: Программа и методика испытаний. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
9. ГОСТ 19.401-78: Текст программы. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
10. ГОСТ 19.404-79: Пояснительная записка. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
11. ГОСТ 19.505-79: Руководство оператора. Требования к содержанию и оформлению. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
12. ГОСТ 19.603-78: Общие правила внесения изменений. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
13. ГОСТ 19.604-78: Правила внесения изменений в программные документы, выполненные печатным способом. // Единая система программной документации. – М.: ИПК Издательство стандартов, 2001.
14. E. Lengyel, Mathematics for 3D Game Programming and Computer Graphics. – Cengage Learning, 2012.

15. CMake. CMake: Cross-Platform Makefile Generator. – URL: https://cmake.org/ (дата обращения: 04.12.2024).
16. GCC. GCC: The GNU Compiler Collection. – URL: https://gcc.gnu.org/ (дата обращения: 04.12.2024).
17. Clang. Clang: a C language family frontend for LLVM. – URL: https://clang.llvm.org/ (дата обращения: 04.12.2024).
18. clang-format. Clang Documentation. – URL: https://clang.llvm.org/docs/ClangFormat.html (дата обращения: 04.12.2024)
19. clang-tidy. Extra Clang Tools Documentation. – URL: https://clang.llvm.org/extra/clang-tidy/ (дата обращения: 04.12.2024).
20. Eigen. Eigen: A C++ template library for linear algebra. – URL: https://eigen.tuxfamily.org/index.php?title=Main_Page (дата обращения: 04.12.2024).
21. SFML. Simple and Fast Multimedia Library. – URL: https://www.sfml-dev.org/ (дата обращения: 04.12.2024).

ЛИСТ РЕГИСТРАЦИИ ИЗМЕНЕНИЙ

Лист регистрации изменений									
Номера листов (страниц)					Всего листов (страниц в докум.)	№ документа	Входящий № сопроводительного докум. и дата	Подп.	Дата
Изм.	Измененных	Замененных	Новых	Аннулированных					