|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_\_«Информатика и системы управления»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_«Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент\_\_\_\_\_\_\_\_\_\_\_Калашников Сергей Дмитриевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа\_\_\_\_ИУ7-43Б\_\_\_\_\_\_\_\_\_

Тип практики \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_технологическая\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Название предприятия\_\_\_\_\_\_\_\_\_ МГТУ им. Н. Э. Баумана, каф. ИУ7\_\_\_\_\_\_

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_Калашников С.Д.\_\_**

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Исаев А.Л.\_\_\_\_\_\_**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*2022 г.*

**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технический университет имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

УТВЕРЖДАЮ

Заведующий кафедрой \_\_\_ИУ7\_\_\_\_

(Индекс)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_И.В.\_Рудаков \_\_

(И.О.Фамилия)

« \_\_\_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20 \_\_\_\_ г.

**ЗАДАНИЕ**

**на выполнение курсовой работы**

по дисциплине \_\_\_\_\_\_\_\_Компьютерная графика\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Студент группы \_ИУ7-53Б\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Калашников Сергей Дмитриевич\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Фамилия, имя, отчество)

Тема курсовой работы: Редактор трехмерных поверхностей

Направленность КР (учебная, исследовательская, практическая, производственная, др.)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_учебная \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Источник тематики (кафедра, предприятие, НИР) \_\_\_\_\_кафедра\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения работы: 25% к \_4\_ нед., 50% к \_7\_ нед., 75% к 11 нед., 100% к 14 нед.

***Задание*** Разработать программу, которая позволит редактировать трехмерные модели на уровне вершин, ребер, полигонов(граней). Каждый элемент модели может подвергаться изменениям переноса, масштабирования и поворота. Должна присутствовать возможность выбора изменяемого элемента при помощи мыши и/или выпадающего списка. Изменяемых элементов может быть несколько или один. Так же изменять можно и модель целиком меняя ее масштаб и положение, добавляя или удаляя вершины, грани и ребра. Предусмотреть возможность визуализации модели как с отображением невидимых граней и ребер, так и без них. Обеспечить просмотр модели с любого положения наблюдателя без изменения положения модели. Источников света не предусмотрено. Реализовать интерфейс, который позволит задавать исходные данные в виде файла, содержащего в себе описание модели и ее положение на сцене. Реализовать интерфейс, который позволит сохранять описание модели и ее положение на сцене в файл.

***Оформление курсовой работы:***

Расчетно-пояснительная записка на 25-30 листах формата А4.

Расчетно-пояснительная записка должна содержать постановку введение, аналитическую часть, конструкторскую часть, технологическую часть, экспериментально-исследовательский раздел, заключение, список литературы, приложения.

Перечень графического материала (плакаты, схемы, чертежи и т.п.)\_\_ На защиту проекта должна быть представлена презентация, состоящая из 15-20 слайдов. На слайдах должны быть отражены: постановка задачи, использованные методы и алгоритмы, расчетные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики разработанного ПО, результаты проведенных исследований.

Дата выдачи задания « \_\_\_ » \_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г.

**Руководитель курсовой работы**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_Павельев А.А.

(Подпись, дата) (И.О.Фамилия)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_**Калашников С.Д.

(Подпись, дата) (И.О.Фамилия)

**Оглавление**

[**Введение** 0](#_Toc113604322)

[**1.** **Аналитическая часть** 0](#_Toc113604323)

[**1.1** **Описание сцены** 0](#_Toc113604324)

[**1.2** **Анализ и выбор формы задания трехмерных моделей** 0](#_Toc113604325)

[**1.3** **Анализ способа задания поверхностных моделей** 0](#_Toc113604326)

[**1.4** **Анализ метода рендера** 0](#_Toc113604327)

[**1.5** **Анализ и выбор алгоритма удаления невидимых ребер и поверхностей** 0](#_Toc113604328)

[**2. Конструкторская часть** 0](#_Toc113604329)

[**2.1** **Общий алгоритм решения поставленной задачи** 0](#_Toc113604330)

[**2.2** **Камера** 0](#_Toc113604331)

[**2.3** **Описание трёхмерных преобразований** 0](#_Toc113604332)

[**2.3.1** **Способ хранения декартовых координат** 0](#_Toc113604333)

[**2.3.2** **Преобразование трёхмерных координат в двухмерное пространство экрана** 0](#_Toc113604334)

[**2.3.3** **Преобразования трёхмерной сцены в пространство камеры** 0](#_Toc113604335)

[**2.3.4** **Матрица перспективной проекции** 0](#_Toc113604336)

[**2.3.5** **Преобразования трёхмерной сцены в пространство области изображения** 0](#_Toc113604337)

[**2.3.6** **Аффинные преобразования в пространстве** 0](#_Toc113604338)

[**2.4** **Трехмерная модель** 0](#_Toc113604339)

[**2.5** **Z-буфер** 0](#_Toc113604340)

[**3. Технологическая часть** 0](#_Toc113604341)

[**3.1 Выбор языка программирования и среды разработки** 0](#_Toc113604342)

[**3.2 Структура классов программы** 0](#_Toc113604343)

[**3.3 Диаграмма классов** 0](#_Toc113604344)

[**3.4 Пользовательский интерфейс** 0](#_Toc113604345)

[**Заключение** 0](#_Toc113604346)

[**Список использованных источников** 0](#_Toc113604347)

# **Введение**

В современном мире трехмерная графика используется практически везде. При создании видеоигр, в разработке дизайнерских и архитектурных проектов, при разработке 3д моделей для печати.

Основным объектом трехмерной графики является трехмерная модель. Она может быть представлена полигональной сеткой или сплайнами.

В связи с этим, становиться актуальной проблема разработки программного обеспечения для редактирования и визуализации трехмерных объектов на двумерных экранах.

Целью данной работы является обоснование выбора алгоритмов, которые можно использовать для редактирования и визуализации трехмерной модели, их практическая реализация и адаптация (при необходимости) к условиям решаемой задачи.

# **Аналитическая часть**

## **Описание сцены**

Сцена состоит из одной или нескольких камер и одной модели.

Камера – это точка, которая имеет вектор взгляда, вектора, указывающие наверх и направо. Пользователь может управлять камерой и с помощью камеры происходит визуализация моделей на экране.

Модель – объект, состоящий из точек, линий и полигонов. Каждый элемент модели пользователь может изменять с помощью матриц преобразований.

## **Анализ и выбор формы задания трехмерных моделей**

В компьютерной графике трехмерное моделирование подразделяют на несколько групп в зависимости от способа задания трехмерной модели.

1. Каркасная (проволочная) модель

Простейшая форма задания трехмерной модели. В этом случае необходимо хранить только вершины – точки, привязанные к координатной сетке и ребра – линии, состоящие из вершин.

1. Поверхностная модель

Самый популярный вид трехмерной модели в поверхностной графике. Модель состоит из вершин, ребер и полигонов. Однако минусом данного вида является незнание того, на какой стороне располагается текстура.

1. Объемная (твердотельная) модель

Данная форма отличается от поверхностной тем, что у нас есть информация о том, где расположен материал. Это делается с помощью указания направления внутренней нормали.

Так как по заданию необходимо редактировать вершины, грани и полигоны, то каркасная модель нам не подходит в связи с отсутствием полигонов. Объемная модель будет избыточна, так как у нас нет задачи создавать высоко реалистичные модели. Таким образом для выполнения задания мы будем опираться на поверхностные модели.

## **Анализ способа задания поверхностных моделей**

Поверхностные модели можно задавать двумя основными способами:

Сплайновым способом. Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра;

Полигональной сеткой. Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

Для более верного выбора также следует перечислить способы хранения информации о сетке:

* Список граней. Объект – это множество граней и множество вершин. В каждую грань входят как минимум 3 вершины;
* «Крылатое» представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые её касаются;
* Полурёберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани;
* Таблица углов. Таблица, хранящая вершины. Обход заданной таблицы неявно задаёт полигоны. Такое представление более компактно и более производительно для нахождения полигонов, но, в связи с тем, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны.
* Вершинное представление. Хранятся лишь вершины, которые указывают на другие вершины. Простота представления даёт возможность проводить над сеткой множество операций.

Так как по заданию нам необходимо выполнять операции над вершинами, гранями и полигонами, то мы будем опираться на представление модели полигональной сеткой.

При этом полигональная сетка будет задаваться динамическим списком граней, и дополнительно будем хранить списки ребер. Это позволит удобно работать с каждым компонентом по отдельности и вместе. Так же в преимущество этого типа хранения говорит то, что алгоритмы удаления невидимых граней и ребер будут эффективно работать с таким представлением модели.

## **Анализ метода рендера**

Одним из самых сложных процессов в трехмерной компьютерной графике является рендер – отображение трехмерной сцены на двухмерном экране.

В современных системах существует два основных метода рендера.

* Трассировка лучей

Камера испускает луч, который проходит через каждый пиксель экрана. Затем ищется пересечение луча с объектами. Если найдено несколько пересечений, то берется точка ближайшая к камере.

Положительными моментами в этом алгоритме являются:

1. высокая реалистичность синтезируемого изображения;
2. работа с поверхностями в математической форме;
3. вычислительная сложность слабо зависит от сложности сцены.
4. Удаление невидимых граней происходит сразу

Недостаток:

1. производительность.

* Растеризация

Данный вид рендера состоит из двух этапов:

Этап вершинного шейдера, где вершины модели преобразуются из трехмерного пространства в двухмерное. И этапа растеризации, где по известным вершинам (в данном случае уже пикселям на экране) необходимо получить все остальные пиксели модели и их атрибуты (цвет, текстура, глубина).

В связи с тем, что в данном проекте не всегда будет необходимо отсекать невидимые грани, то мы будем использовать рендер методом растеризации.

## **Анализ и выбор алгоритма удаления невидимых ребер и поверхностей**

**Алгоритм, использующий Z-буфер**

Суть данного алгоритма – это использование двух буферов: буфера кадра, в котором хранятся атрибуты каждого пикселя, и Z-буфера, в котором хранятся информация о координате Z для каждого пикселя.

Первоначально в z-буфере находятся минимально возможные значения , а в буфере кадра располагаются пиксели, описывающие фон. Каждый многоугольник преобразуется в растровую форму и записывается в буфер кадра.

В процессе подсчета глубины нового пикселя, он сравнивается с тем значением, которое уже лежит в z-буфере. Если новый пиксель расположен ближе к наблюдателю, чем предыдущий, то он заносится в буфер кадра и происходит корректировка z-буфера.

Для решения задачи вычисления глубины каждый многоугольник описывается уравнением При многоугольник для наблюдателя вырождается в линию.

Для некоторой сканирующей строки , поэтому имеется возможность рекуррентно высчитывать для каждого :

Получим*: ,* так как

При этом стоит отметить, что для невыпуклых многогранников предварительно потребуется удалить нелицевые грани.

Положительными моментами в этом алгоритме являются:

* простота реализации;
* оценка трудоемкости линейна.

Недостаток:

* сложная реализация прозрачности;
* большой объем требуемой памяти.

**Алгоритм Робертса**

Данный алгоритм работает в объектном пространстве, решая задачу только с выпуклыми телами.

Алгоритм выполняется в 3 этапа:

1. Этап подготовки исходных данных. На данном этапе должна быть задана информация о телах. Для каждого тела сцены должна быть сформирована матрица тела . Размерность матрицы - , где – количество граней тела.

Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости , проходящей через очередную грань.

Таким образом, матрица тела будет представлена в следующем виде:

Матрица тела должна быть сформирована корректно, то есть любая точка, расположенная внутри тела, должна располагаться по положительную сторону от каждой грани тела. В случае, если для очередной грани условие не выполняется, соответствующий столбец матрицы надо умножить на .

2. Этап удаления рёбер, экранируемых самим телом.

На данном этапе рассматривается вектор взгляда .

Для определения невидимых граней достаточно умножить вектор на матрицу тела . Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

3. Этап удаления невидимых рёбер, экранируемых другими телами сцены.

На данном этапе для определения невидимых точек ребра требуется построить луч, соединяющий точку наблюдения с точкой на ребре. Точка будет невидимой, если луч на своём пути встречает в качестве преграды рассматриваемое тело.

Положительным моментом этого алгоритма являются:

* работа в объектном пространстве;
* высокая точность вычисления.

Недостатки:

* рост сложности алгоритма – квадрат числа объектов;
* тела сцены должны быть выпуклыми (усложнение алгоритма, так как нужна будет проверка на выпуклость);
* сложность реализации.

**Алгоритм художника**

Данный алгоритм работает аналогично тому, как художник рисует картину – то есть сначала рисуются дальние объекты, а затем более близкие. Наиболее распространенная реализация алгоритма – сортировка по глубине, которая заключается в том, что произвольное множество граней сортируется по ближнему расстоянию от наблюдателя, а затем отсортированные грани выводятся на экран в порядке от самой дальней до самой ближней. Данный метод работает лучше для построения сцен, в которых отсутствуют пересекающиеся грани.

Положительным моментом данного алгоритма является:

* требование меньшей памяти, чем, например, алгоритм Z-буффера.

Недостатки

* недостаточно высока реалистичность изображения;
* сложность реализации при пересечении граней на сцене.

**Алгоритм Варнока**

Алгоритм Варнока является одним из примеров алгоритма, основанного на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто.

Поскольку алгоритм Варнока нацелен на обработку картинки, он работает в пространстве изображения. В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения.

Сравнивая область с проекциями всех граней, можно выделить случаи, когда изображение, получающееся в рассматриваемой области, определяется сразу:

* проекция ни одной грани не попадает в область;
* проекция только одной грани содержится в области или пересекает область. В этом случае проекции грани разбивают всю область на две части, одна из которых соответствует этой проекции;
* существует грань, проекция которой полностью накрывает данную область, и эта грань расположена к картинной плоскости ближе, чем все остальные грани, проекции которых пересекают данную область. В данном случае область соответствует этой грани.

Если ни один из рассмотренных трех случаев не имеет места, то снова разбиваем область на четыре равные части и проверяем выполнение этих условий для каждой из частей. Те части, для которых таким образом не удалось установить видимость, разбиваем снова и т. д.

Преимущества:

* меньшие затраты по времени в случае области, содержащий мало информации.

Недостатки:

* алгоритм работает только в пространстве изображений;
* большие затраты по времени в случае области с высоким информационным содержимым.

**Вывод**

Для удаления невидимых линий выбран алгоритм z-буфера так как он обеспечивает наилучшее качество удаления невидимых граней, и не предъявляет требования к моделям.

# **2. Конструкторская часть**

## **Общий алгоритм решения поставленной задачи**

1. Задать примитивную модель (куб, пирамида или сфера) или загрузить модель из obj файла.
2. Отобразить модель
3. Выбрать активный элемент
4. Задать матрицу трансформирования
5. Применить матрицу трансформирования
6. Отобразить изменения

## **Камера**

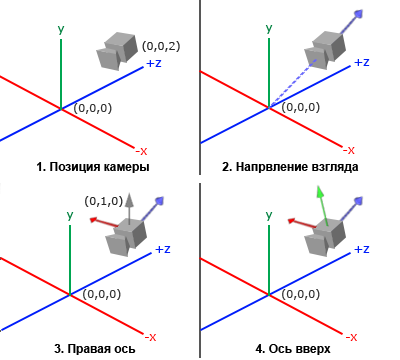
Так как пользователь будет смотреть на сцену через камеру, то мы должны ее реализовать.

Камера задается тремя нормализованными перпендикулярными векторами и вектором позиции камеры в мировом пространстве. (рис. 1)

Вектор Direction– вектор противонаправленный взгляду камеры

Вектор Right – вектор, указывающий правую сторону у камеры

Вектор Up – вектор показывающий верх камеры



*Рисунок 1. Оси камеры. Синим отмечена Direction, красным – Right, зеленым - Up*

* 1. **Описание трёхмерных преобразований**
     1. **Способ хранения декартовых координат**

Для хранения координат точек будет использоваться вектор-столбец, состоящий из четырёх координат: x, y, z, w, причём w по умолчанию равна 1. Это сделано для того, чтобы было удобно умножать вектор на матрицы трансформации, которые обладают размерностью 4x4.

### **Преобразование трёхмерных координат в двухмерное пространство экрана**

Экран обладает только двумя координатами, поэтому нужно выбрать способ, каким образом трёхмерные объекты переносить на двухмерное пространство экрана. Каждый пиксель обладает определённым цветом, и за счёт этого нужно решить проблему передачи объёмности и реалистичности изображения. Алгоритм приведения координат к нужному виду, следующий:

1. Перевести объект из мирового пространства в пространство камеры.

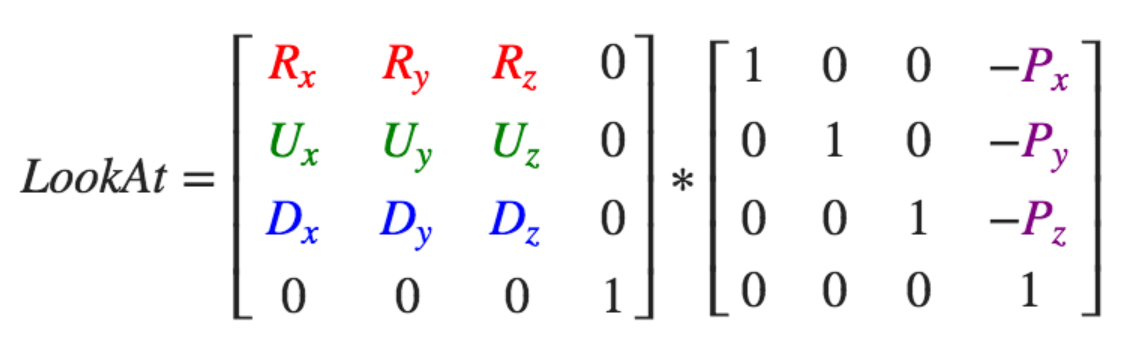
2. Найти все проекции точек из пространства камеры в видимые точки, где координаты точек x, y, z находятся в диапазоне [−w, w], а w находится в диапазоне [0, 1].

3. Масштабировать все точки, полученные в п.2, на картинку необходимого разрешения.

Чтобы выполнить все преобразования, нужно использовать матрицы преобразований. Сначала вычисляются все необходимые матрицы, затем они перемножаются в нужном порядке. Исходные координаты умножаются на получившийся результат, в результате чего координаты приводятся к нужной системе.

### **Преобразования трёхмерной сцены в пространство камеры**

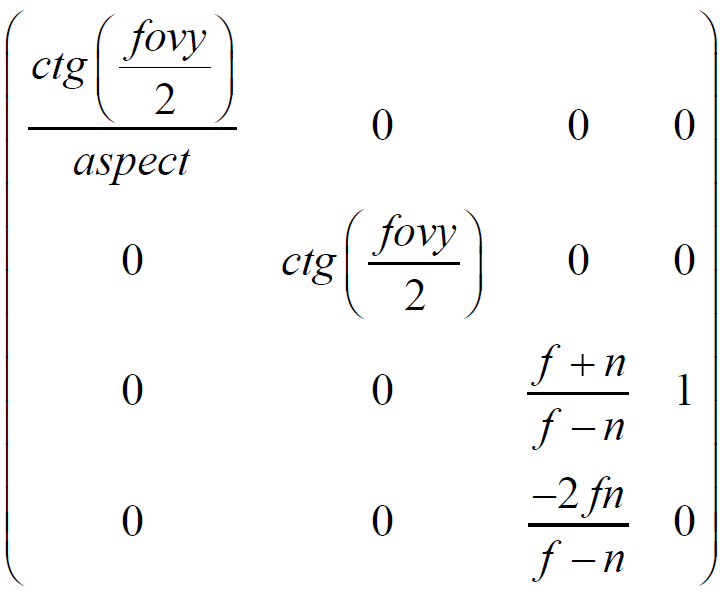
Чтобы привести трёхмерную сцену к пространству камеру, нужно умножить каждую вершину всех полигональных моделей на матрицу камеры. (рис. 2)



*Рисунок 2. Где в первой матрице R – правый вектор камеры, U – вектор в верх, D – взгляд камеры. А во второй P – координаты позиции камеры.*

### **Матрица перспективной проекции**

После перехода в пространство камеры нужно умножить каждую вершину всех полигональных моделей на матрицу проекции (рис. 3). Эта матрица преображает заданный диапазон усечённой пирамиды в пространство отсечение, и изменяет w-компоненту так, что чем дальше от наблюдателя находится вершина, тем больше возрастает w. После преобразования координат в пространство отсечения, координаты x и y попадают в промежуток [−w, w], а вершина z – [-0, w]. Всё, что находится вне диапазона, отсекается. Следующий этап – спроецировать все координаты на одну плоскость, разделив всё на координату z. После умножения вектора координат на матрицу перспективной проекции, реальная координата z заносится в w-компоненту, так что вместо деления на z делят на w.



*Рисунок 3 Матрица проекции. Где fovy – угол обзора наблюдателя в градусах, f – дальняя плоскость отсечения, n – ближняя плоскость отсечения, aspect – соотношение сторон отсекателя*

### **Преобразования трёхмерной сцены в пространство области изображения**

Чтобы преобразовать спроецированные координаты в координаты области изображения, нужно инвертировать ось Y и выполнить операции преноса и маштабирования.

### **Аффинные преобразования в пространстве**

Для изменения объекта в пространстве к нему необходимо применить матрицу преобразования. Существуют 3 основных вида аффинных преобразований:

1. Перенос

Где dx, dy, dz – коэффициенты переноса по соответствующим осям

1. Масштабирование

Где kx, ky, kz – коэффициенты маштабирования по соответствующим осям

1. Поворот

* Вокруг оси X
* Вокруг оси Y
* Вокруг оси Z

Где y – угол поворота

## **Трехмерная модель**

Трехмерная модель состоит из линий и граней в трехмерном пространстве так как была выбрана поверхностная модель с полигональной сеткой. Однако дополнительно мы будем хранить список точек для того, чтобы избегать повторных преобразований над точками модели так как каждая точка дублироваться в разных гранях.

* Точка задается тремя координатами x, y, z в мировом пространстве
* Линия задается двумя точкам
* Каждая грань представлена тремя точками из-за того, что три точки однозначно задают плоскость, и при изменении объекта нет необходимости проводить триангуляцию в случае «ухода» одной из точек с плоскости.

Каждый из объектов так же должен хранить информацию об объектах содержащих и входящих. К примеру, для вершины, ребра и грани будут объектами содержащими. Для ребра, точки – это объекты входящие, а грани – содержащие. Сделано это для того, чтобы при удалении не оставалось «висящих» объектов. Так как используется поверхностная модель, которая состоит из полигонов, то в фигуре не могут присутствовать точки и грани, не входящие в полигон. То есть если удалена точка, то грань, содержащая эту точку, должна так же удалиться, так как две оставшиеся точки не могут задавать плоскость. А при удалении у объекта всех объектов содержащих, удаляется и сам объект: если точка входит в два смежных ребра, то при удалении обоих ребер удалится и смежная точка.

Для такой системы создан класс Contaner<T>, который содержит 3 динамических списка:

1. Список объектов
2. Список содержащих объектов
3. Список входящих объектов

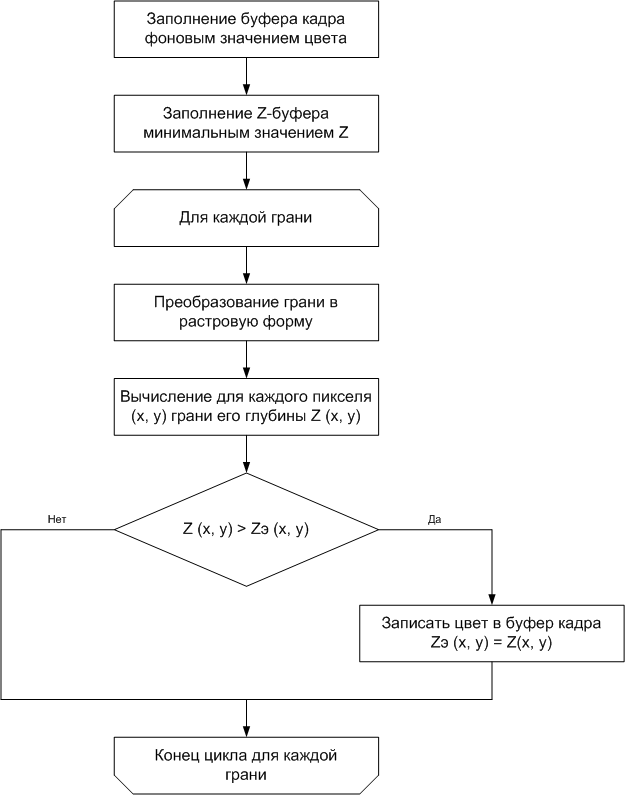
T – классы элементы модели:

* PontComponent – реализация точки в пространстве
* LineComponent – реализация линии в пространстве
* PolygonComponent – реализация треугольника в пространстве

Трехмерная модель состоит из 3-х структур Container каждая из которых совпадает по типу с одним из элементов модели.

## **Z-буфер**

Алгоритм z-буфера приведен на блок схеме ниже.



# **3. Технологическая часть**

## **3.1 Выбор языка программирования и среды разработки**

Существует множество языков, а также сред программирования, многие из которых обладают достаточно высокой эффективностью, удобством и простотой в использовании. Для разработки данной программы был выбран язык C#. Данный выбор обусловлен следующими факторами:

* Этот язык предоставляет программисту широкие возможности реализации самых разнообразных алгоритмов. Он обладает высокой эффективностью и большим набором стандартных классов и процедур.
* C# является полностью объектно-ориентированным. Он позволяет использовать множественное наследование, абстрактные и параметризованные классы.
* Трехмерные объекты, также как и математические абстракции, естественным образом представляются в виде объектов классов, что позволяет легко и эффективно организовывать их взаимодействие, при этом сохраняется читаемый и легко изменяемый код.
* В c# реализована многопоточность, что может быть полезным при реализации алгоритмов, требующих больших затрат по времени.

В качестве среды разработки была выбрана Visual Studio 2022. Некоторые факторы, по которым была выбрана данная среда:

* Включает весь основной функционал: параллельная сборка, отладчик, поддержка точек останова, сборки и т.д.
* Разработчики имеют возможность расширить любой функционал, включая компиляцию, отладку.
* Работает с интерфейсом Windows Forms, который очень удобен в использовании, а также позволяет без проблем создавать приложения.

## **3.2 Структура классов программы**

Так как при написании программы используется язык C#, а это объектно-ориентированный язык, то особое внимание уделено структуре классов.

Условно классы в программе можно разделить на несколько групп по выполняемым функциям.

* Математические структуры

MatrixCoord3D – представление координат в матричном виде

MatrixTransformScale3D – матрица масштабирования

MxtrixTransformTransfer3D – матрица переноса

MatrixTransformRotateX3D – матрица поворота вокруг X

MatrixTransformRotateY3D – матрица поворота вокруг Y

MatrixTransformRotateZ3D – матрица поворота вокруг Z

MatrixTransformRotateVec3D – матрица поворота вокруг произвольного вектора

MatrixPerspectiveProjection – матрица перспективной проекции

MatrixOrtoProjection – матрица ортогональной проекции

* Трехмерные объекты

PointComponent – вершина модели

LineComponent – грань модели

PolygonComponent – полигон модели

Model – модель

Camera – камера

Axes – координатные оси

Container – контейнерный класс для хранения объектов

* Алгоритмы преобразований и визуализации

EasytransformVisitor – преобразования объектов относительно начала координат

HardTtransformVisitor – преобразования объектов относительно любой точки

DrawVisitor – отрисовка объектов

VertexShader – перевод точек из трехмерного пространства в двухмерное

VertexShaderProjection – перевод точек из трехмерного пространства в двухмерное с использованием матрицы проекции

* Алгоритмы визуализации

RasterizatorNoCutter – растеризатор без отсечения невидимых полигонов

RasterizatorNoText – растеризатор без отображения координат точек

RasterizatorNoPoints – растеризатор без отображения точек

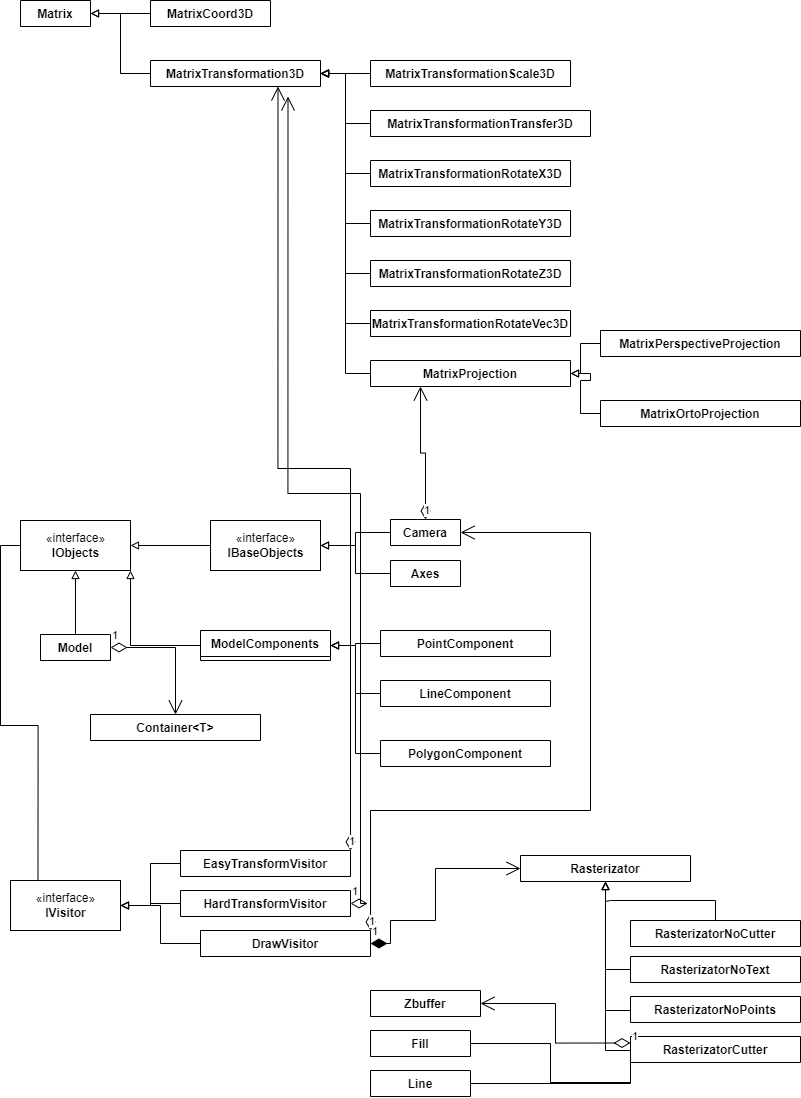
ZBuffer – з-буффер

RasterizationCutter – растеризатор с отсечением невидимых полигонов

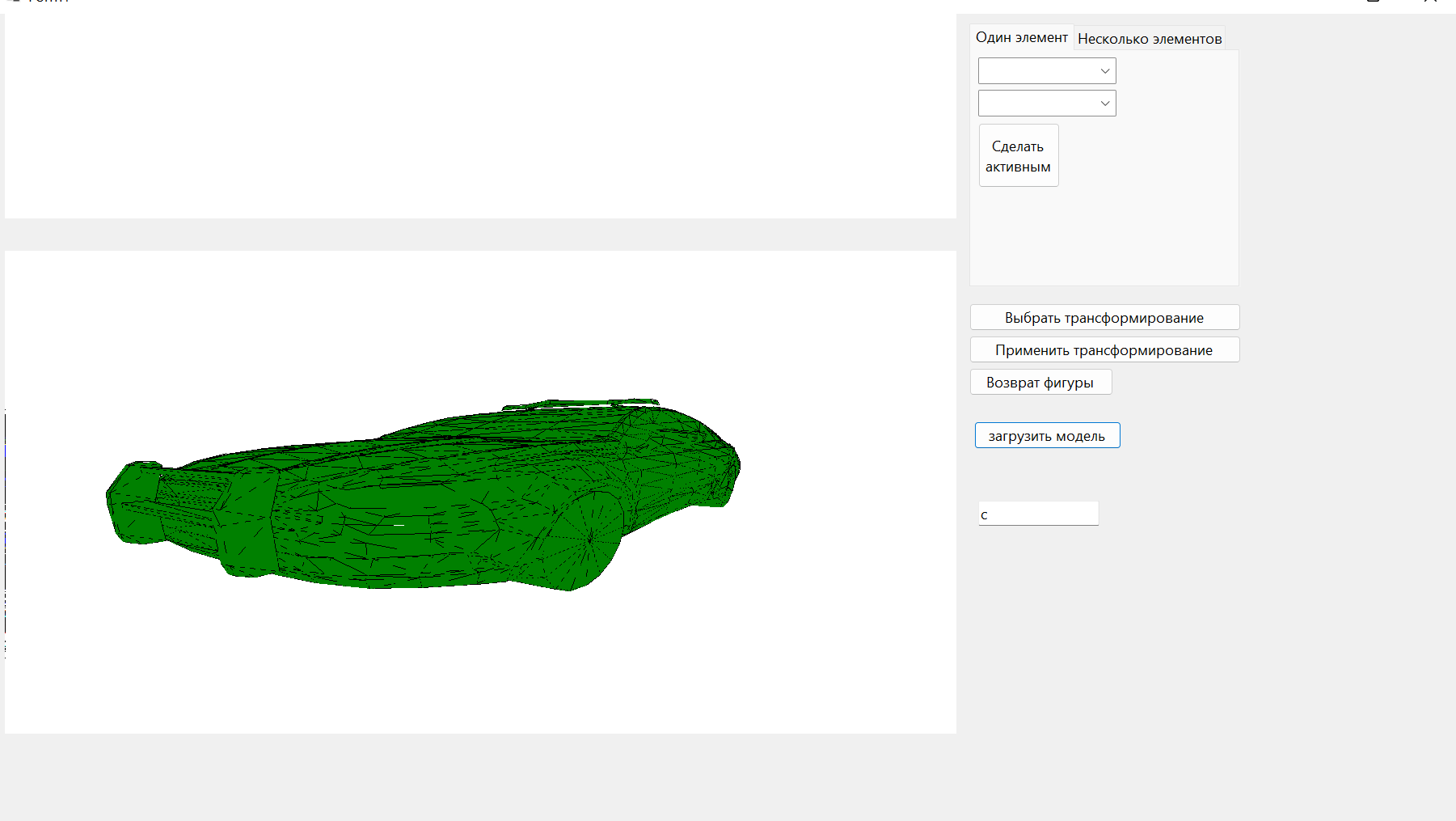
Line – растеризация линии

Fill – растеризация треугольника

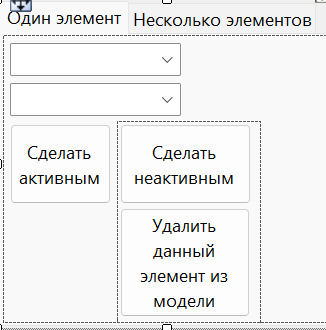
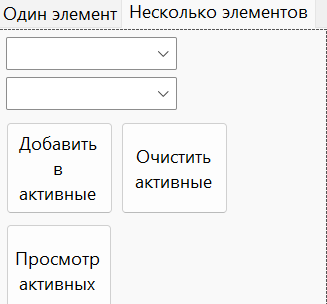
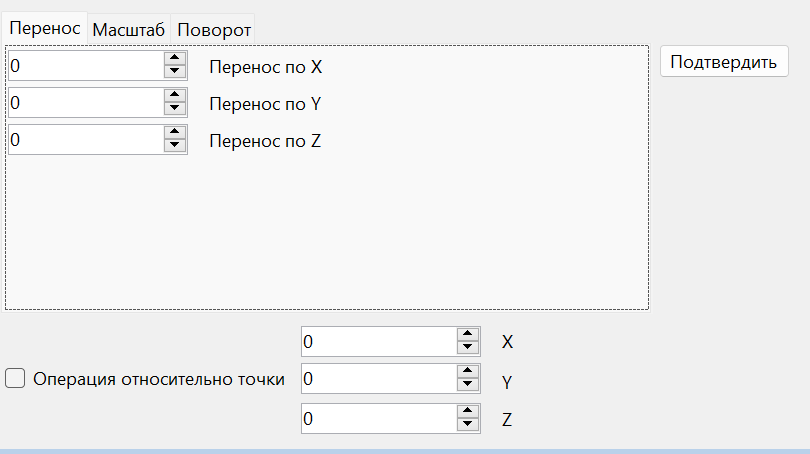
## **3.3 Диаграмма классов**



## **3.4 Пользовательский интерфейс**



Пользователю доступны следующие опции:

1. С помощью мыши или выпадающих списков сверху можно выбрать один или несколько активных элементов.
2. В окне выбора трансформирования, открывающегося по кнопке, можно задать матрицы трансформирования.
3. Кнопка «Применить трансформирование» отвечает за применение матрица трансформирования к активному элементу.
4. У пользователя есть возможность управлять камерой с клавиатуры при помощи следующих клавиш:

Q – вверх

W – вперед

E – вниз

A – влево

S – назад

D – вправо

Z – поворот вокруг оси Y против часовой стрелки

X – поворот вокруг оси Y по часовой стрелке

F – наклон камеры вниз

G – наклон камеры вверх

C – поворот камеры влево

V – поворот камеры вправо

1. Пользователь может менять масштаб изображения при помощи удержания клавиши ctrl и вращения колесика мыши.
2. В текстовом поле под кнопкой «загрузить модель» можно сменить систему рендера.

Всего в программе реализовано 4 системы рендера:

* Отрисовка модели без удаления невидимых граней с отображением информации о точках
* Отрисовка модели без удаления невидимых граней без отображения информации о точках
* Отрисовка модели без удаления невидимых граней без точек
* Отрисовка модели с использованием z-буфера

# **Заключение**

Во время выполнения поставленной задачи были проанализированы основные способы представления и задания трехмерных моделей, а также рассмотрены и основные алгоритмы удаления невидимых. Проанализированы достоинства и недостатки представленных алгоритмов и выбраны наиболее оптимальные для решения поставленной задачи.

Проделанная работа помогла закрепить полученные навыки в области компьютерной графики и проектирования программного обеспечения. Реализация программы позволяет легко и быстро добавить необходимые новые объекты (классы). Пользователь может свободно управлять камерой и изменять компоненты модели.

# **Список использованных источников**

1. Дёмин А.Ю., Основы компьютерной графики: учебное пособие – Томск: Изд-во Томского политехнического университета, 2011. – 191 с.

2. Авдеева С.М., Куров А.В. Алгоритмы трехмерной машинной графики: учебное пособие. - М.: Издательство МГТУ им. Н.Э. Баумана, 1996. - 60 с., ил.

3. Шикин Е.В., Боресков А.В. Компьютерная графика. Динамика, реалистические изображения. – М.: Диалог-МИФИ, 1995. – 288 с.

4. Преобразования в OpenGL [Электронный ресурс]. – Режим доступа: http://www.songho.ca/opengl/gl\_transform.html (дата обращения 16.07.22)

5. Роджерс Д., Алгоритмические основы машинной графики: пер. с англ.— М.: Мир, 1989.— 512 с.: ил.

6. Рихтер Дж., CLR via C#. Программирование на платформе Microsoft .NET Framework 4.0 на зыке C#. 3-е изд. – СПб,: Питер, 2012 – 928 с.: ил.