



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 ИУ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА ИУ7 «Программное обеспечение ЭВМ и информационные технологии»

## ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент Вольняга Максим Юрьевич

Группа ИУ7-46Б

Тип практики Технологическая

Название предприятия НУК ИУ МГТУ им. Н. Э. Баумана

Студент

\_\_\_\_\_  
(Подпись, дата)

М. Ю. Вольняга  
(И. О. Фамилия)

Руководитель практики

\_\_\_\_\_  
(Подпись, дата)

А. В. Куров  
(И. О. Фамилия)

Оценка \_\_\_\_\_

2023 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

И. В. Рудаков

«        »                      2023 г.

**ЗАДАНИЕ**  
**на прохождение производственной практики**  
**Технологическая**

Тип практики

Студент Исаев Диас 2 курса группы ИУ7-46Б  
№ курса индекс группы

в период с 01 . 07 .2023 г. по 21 . 07 .2023 г.

Предприятие: НУК ИУ МГТУ им. Н. Э. Баумана

Руководитель практики от кафедры:  
Куров Андрей Владимирович  
(Фамилия Имя Отчество полностью)

Задание:

1. Исследовать алгоритмы для разработки программного обеспечения, пред назначенного для визуализации и поворота граней "Кубика Рубика".
2. Обосновать выбор алгоритмов, которые будут использованы при разработке программного обеспечения.
3. Выбрать и описать структуры данных для объектов сцены.

Дата выдачи задания « 01 » июля 2023 г.

Руководитель практики от кафедры Куров А. В. /

Студент Исаев Д. /

# Оглавление

<b>Введение</b>	<b>4</b>
<b>1 Аналитическая часть</b>	<b>5</b>
1.1 Способы определения моделей . . . . .	5
1.1.1 Каркасная модель . . . . .	5
1.1.2 Поверхностная модель . . . . .	6
1.1.3 Твердотельная модель . . . . .	7
1.1.4 Выбор определения моделей . . . . .	8
1.2 Способы представления трехмерных поверхностей . . . . .	8
1.2.1 Способы описания полигональных сеток . . . . .	9
1.2.2 Выбор способа описания полигональной сетки . . . . .	11
1.3 Формализация объектов сцены . . . . .	12
1.3.1 Сцена . . . . .	12
1.3.2 Объекты сцены . . . . .	12
1.3.3 Источник света . . . . .	13
1.4 Алгоритмы удаления скрытых линий и поверхностей . . . . .	14
1.4.1 Алгоритм Робертса . . . . .	14
1.4.2 Алгоритм плавающего горизонта . . . . .	15
1.4.3 Z-буферный алгоритм удаления поверхностей . . . . .	15
1.4.4 Алгоритм, использующий список приоритетов. . . . .	16
1.4.5 Алгоритм Варнока . . . . .	17
1.4.6 Алгоритм прямой и обратной трассировки лучей . . . . .	17
1.4.7 Выбор алгоритма удаления скрытых линий и поверхностей . . . . .	18
1.5 Модель освещения . . . . .	19
1.5.1 Простая модель . . . . .	19
1.5.2 Модель Ламберта . . . . .	20
1.5.3 Модель Фонга . . . . .	21
1.6 Алгоритмы построения теней . . . . .	22
1.6.1 Модификация Z-буфера . . . . .	22
1.6.2 Модификация Вейлера-Азертона . . . . .	22
1.7 Вывод . . . . .	23
<b>2 Конструкторская часть</b>	<b>24</b>
2.1 Z-буферный алгоритм . . . . .	24
2.1.1 Формальная запись . . . . .	24
2.2 Модифицированный Z-буферный алгоритм . . . . .	27
2.2.1 Формальная запись . . . . .	27
2.3 Выбор используемых типов и структур данных . . . . .	29

2.4	Вывод . . . . .	29
<b>3</b>	<b>Технологическая часть</b>	<b>30</b>
3.1	Средства реализации . . . . .	30
3.2	Реализация алгоритмов . . . . .	30
<b>4</b>	<b>Исследовательская часть</b>	<b>32</b>
4.1	Результаты работы программного обеспечения . . . . .	32
4.2	Постановка эксперимента . . . . .	34
4.2.1	Цель эксперимента . . . . .	34
4.2.2	Данные реального бицепса . . . . .	34
4.2.3	Замеры реального бицепса . . . . .	35
4.2.4	Замеры разработанной модели . . . . .	36
4.2.5	Сравнение замеров . . . . .	37
	<b>Заключение</b>	<b>39</b>
	<b>Литература</b>	<b>40</b>

# Введение

Программное моделирование загородной местности с каждым годом набирает все большее значение в области ландшафтного дизайна, архитектуры и планирования. Применение современных технологий дает возможность разработки детализированных трехмерных моделей, обеспечивая тем самым более глубокое понимание и визуализацию географических и ландшафтных особенностей, а также предстоящих изменений. Использование метода 3D-визуализации при планировании местности преобразует сложные концепции в ясные визуальные образы, что значительно упрощает процесс коммуникации и минимизирует риск возникновения недопонимания между заказчиком и исполнителем [1].

Цель работы — изучение и выбор алгоритмов для разработки программного обеспечения, предназначенного для визуализации и макетирования загородной местности.

Чтобы достигнуть поставленной цели, требуется решить следующие задачи:

- формально описать структуру объектов;
- выбрать алгоритмы трехмерной графики для визуализации сцены и объектов;
- формально записать выбранные алгоритмы;
- выбрать структуры данных для объектов сцены.

# 1 Аналитическая часть

## 1.1 Способы определения моделей

В системах трехмерного моделирования используются широко используется, каркасные, поверхностные и объемные твердотельные модели. Правильный выбор метода определения моделей на сцене является ключевым фактором, определяющим размер и визуализацию модели в сцене, что в свою очередь способствует точному представлению их формы и размеров.

### 1.1.1 Каркасная модель

Каркасные модели представляют объекты, созданные из соединенных ребер, похожие на объекты, сделанные из проволоки. В таких моделях грани объекта не определены, но их границы представлены ребрами. Каркасная модель не имеет поверхности, которые бы скрывали ребра, поэтому она выглядит прозрачной [2].

Этот тип моделирования относится к категории наиболее примитивных и имеет ряд значительных ограничений. Большинство из них связаны с отсутствием информации о гранях, образованных линиями, а также с невозможностью разграничить внутреннюю и внешнюю зоны в изображении твердого тела, это хорошо видно на рисунке 1.1. Несмотря на ограничения, каркасная модель занимает меньше памяти и является достаточно эффективной для выполнения простых задач. Этот тип представления часто применяется не столько для моделирования, сколько для отображения моделей в качестве одного из методов визуализации. Наиболее широко каркасное моделирование используется для имитации траектории движения инструмента, выполняющего несложные операции [3].

**Недостатки каркасной модели, обсуждаются в [3]:**

1. *неоднозначность* — визуализация всех ребер может привести к неясности и непредсказуемым результатам. Видимые грани и невидимые трудно отличить;
2. *затруднения в идентификации криволинейных граней* — каркасная модель не способна адекватно представить поверхности с непостоянной кривизной, например, боковые поверхности цилиндров, что может вводить в заблуждение;
3. *ограниченность в обнаружении взаимодействий между компонентами* — из-за отсутствия информации о поверхностях, модель не мо-

жет предсказать возможные конфликты между гранями объекта;

4. *трудности в вычислении физических характеристик из-за простоты модели;*
5. *невозможность создания градационных изображений* — каркасная модель не предполагает создания тоновых изображений, которые требуют затенения граней, а не ребер.

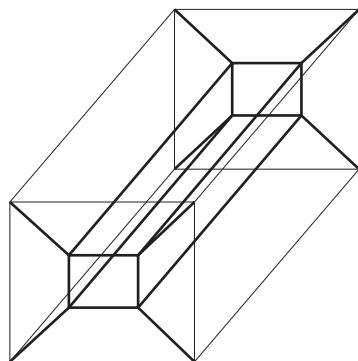


Рисунок 1.1 – Пример каркасной модели.

### 1.1.2 Поверхностная модель

Поверхностные модели включают как ребра, так и поверхности, что позволяет более точно представить объект, чем каркасные модели. В поверхностных моделях грани, расположенные спереди, перекрывают грани, находящиеся на заднем плане, это можно увидеть на рисунке 1.2. Когда изображение выводится на монитор, можно получить более реалистичное представление трехмерного объекта. Поверхностные модели имеют объем, но не имеют массы [2]

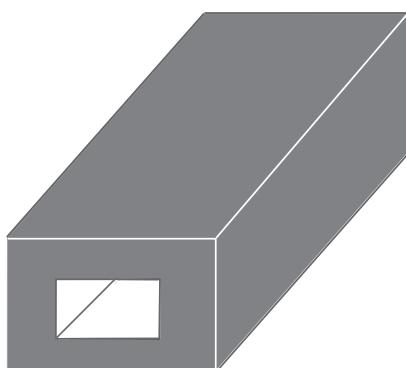


Рисунок 1.2 – Пример поверхностной модели.

Рассмотрев [3], можно выделить следующие преимущества по сравнению с каркасными моделями:

1. способность распознавания и изображения сложных криволинейных граней;
2. изображение грани для получения тоновых изображений;
3. особые построения на поверхности (отверстия);
4. возможность получения качественного изображения.

В основу поверхностной модели положены два основных математических положения:

- любую поверхность можно аппроксимировать многогранником, каждая грань которого является простейшим плоским многоугольником [3];
- наряду с плоскими многоугольниками в модели допускаются поверхности второго порядка и аналитически не описываемые поверхности, форму которых можно определить с помощью различных методов аппроксимации и интерполяции. В отличие от каркасного моделирования каждый объект имеет внутреннюю и внешнюю часть [3].

*Недостаток поверхностной модели* — отсутствует информация, о том, с какой стороны поверхности находится материал, а с какой пустота.

### **1.1.3 Твердотельная модель**

Твердотельное моделирование представляет собой наиболее всесторонний и точный метод создания виртуального аналога реального объекта. В результате применения данного метода формируется монолитная модель будущего изделия, включающая в себя такие элементы как линии, грани и, что особенно важно, образует зону поверхности в рамках геометрической формы объекта с такими ключевыми характеристиками как масса и объем [3].

**Преимущества твердотельных моделей:**

1. полное определение объемной формы с возможностью разграничивать внутренний и внешние области объекта;
2. обеспечение автоматического удаления скрытых линий;
3. автоматическое построение 3D разрезов компонентов, что особенно важно при анализе сложных сборочных изделий;

4. применение методов анализа с автоматическим получением изображения точных весовых характеристик методом конечных элементов;
5. получение тоновых эффектов, манипуляции с источниками света [3].

#### **1.1.4 Выбор определения моделей**

В контексте представленной задачи наиболее оптимальным выбором являются поверхностные модели объектов. Поверхностные модели позволяют детализировать геометрическую форму объекта, минуя углубленное рассмотрение его внутренней структуры и свойств материалов. Данный подход находит свое применение в ситуациях, когда специфика материала объекта не влияет на проводимый анализ. Так, не требуется тратить ресурсы на моделирование внутренних особенностей и характеристик материала, что делает использование поверхностных моделей более рациональным. Каркасные модели в данном контексте не подходят из-за неполноты представления формы объекта, в то время как твердотельные модели избыточны из-за детальности, не требуемой для задачи.

### **1.2 Способы представления трехмерных поверхностей**

Для представления трехмерных поверхностей существуют два широко используемых метода:

- *метод полигональных сеток*: этот метод представляет объект в виде связанной между собой сетки плоских многоугольников, это заметно на рисунке 1.3. Несмотря на его простоту и удобство для описания некоторых типов объектов, например, архитектурных конструкций, метод имеет ограниченную точность, особенно в случаях сложных или криволинейных форм;
- *метод параметрических бикубических кусков*: этот подход использует математические формулы, описывающие координаты поверхностей. Эти уравнения имеют два параметра и варьируются по степеням не выше третьей. Этот подход обеспечивает высокую точность при описании поверхности и требует меньше элементов для описания сложных форм, в сравнении с полигональными сетками [2].

Исходя из требований к простоте моделирования, экономии ресурсов и приемлемой детализации для задачи макетирования загородной среды, ме-

тод полигональных сеток является оптимальным выбором. Благодаря геометрической простоте, он подходит для представления основных элементов и обеспечивает рациональное использование вычислительных ресурсов. При этом, уровень детализации соответствует требованиям проекта, не делая необходимым применение более сложных методов моделирования.

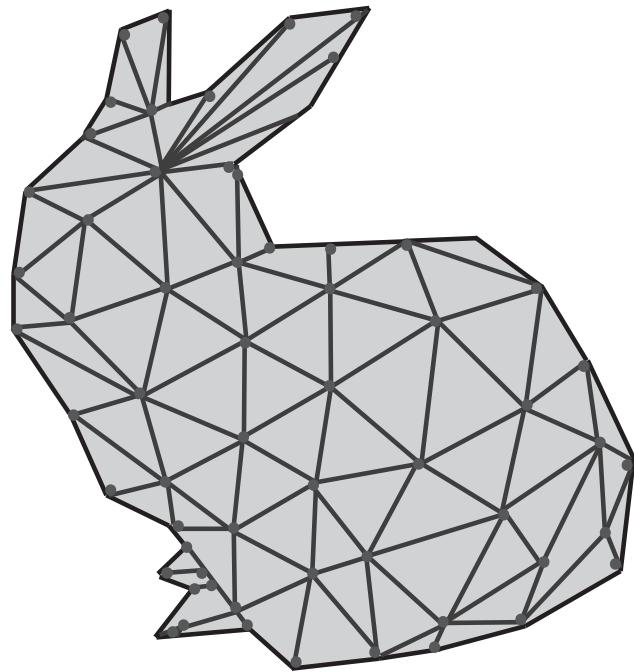


Рисунок 1.3 – Пример полигональной сетки, изображающей кролика.

### 1.2.1 Способы описания полигональных сеток

Существует несколько способов описания полигональных сеток, каждый из которых имеет свои преимущества и недостатки в зависимости от конкретных требований и ограничений приложения.

**Наиболее распространенные методы представления полигональных сеток, рассматриваются в [4]:**

1. **список граней** — один из наиболее распространенных подходов к представлению трехмерных моделей, представляет объект как множество граней и множество вершин. В каждую грань входят как минимум 3 вершины, пример рисунок 1.4;

**Преимущества использования списка граней:**

- *простота поиска вершин грани*: благодаря тому, что список всех граней содержит информацию обо всех связанных вершинах, поиск вершин грани облегчается;

- возможность динамического обновления формы: используя графический процессор, форма может быть обновлена динамически без обновления связности граней.

### Недостатки использования списка граней:

- трудности при выполнении операций разрыва и объединения граней: недостаток явной информации о связях между гранями может усложнить выполнение таких операций;
- проблемы с поиском граней: в силу отсутствия явно заданных ребер, поиск всех граней может стать затруднительным.

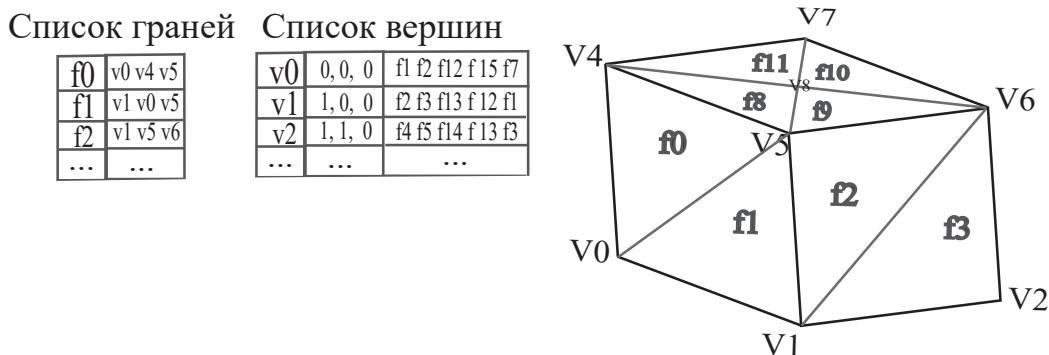


Рисунок 1.4 – Список граней

2. **вершинное представление** — это метод представления модели через коллекцию вершин, которые связаны между собой, пример рисунок 1.5;

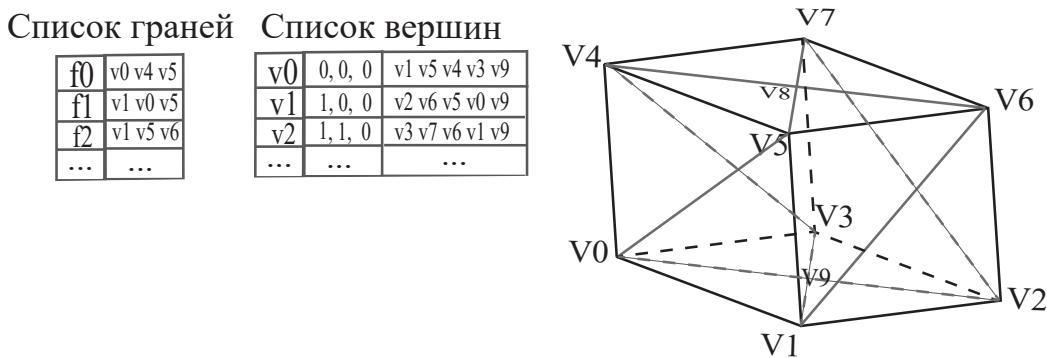


Рисунок 1.5 – Вершинное представление

### Преимущества использования вершинного представления:

- простота: вершинное представление является наиболее базовым и простым способом представления трехмерной модели.

## Недостатки использования вершинного представления:

- отсутствие явного выражения информации о гранях и ребрах: для генерации списка граней для визуализации, требуется пройти по всем данным;
- редкое использование: из-за ограниченности функционала, вершинное представление редко используется в современных системах визуализации.

3. «*крылатое*» представление — это метод представления модели, который решает проблему перехода от ребра к ребру, путем упорядочивания множества граней вокруг каждого ребра, пример рисунок 1.6;

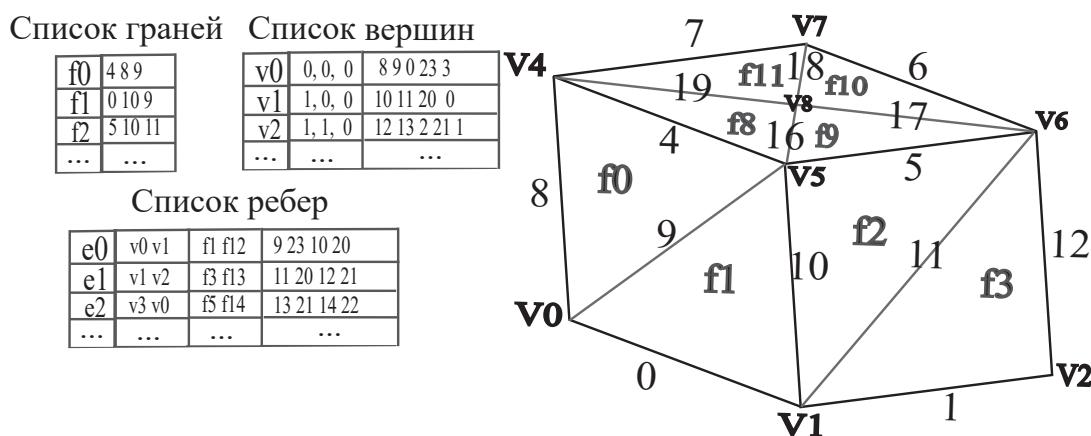


Рисунок 1.6 – «крылатое» представление

## Преимущества использования «крылатого» представления:

- поддержка сложных операций: «крылатое» представление подходит для операций подразделения поверхностей и интерактивного моделирования, благодаря своей уникальной структуре.

## недостатки использования «крылатого» представления:

- высокие требования к памяти: увеличивающаяся сложность структуры приводит к увеличению объема занимаемой памяти при использовании данного представления.

### 1.2.2 Выбор способа описания полигональной сетки

В своей работе я выбрал подход к представлению моделей с использованием списка граней, поскольку он предлагает ясное описание граней

и удобный доступ к элементам сетки. Список граней позволяет легко и эффективно модифицировать модели, включая добавление, удаление и изменение граней, вершин и ребер.

## 1.3 Формализация объектов сцены

После изучения и выбора наиболее эффективных методов определения моделей, а именно: использование поверхностных моделей для описания трехмерных объектов, метода полигональных сеток в качестве способа представления трехмерных поверхностей, а также методики описания полигональных сеток с использованием списка граней, мы можем переходить к конкретизации объектов сцены.

### 1.3.1 Сцена

Сцена представляет собой прямоугольный параллелепипед с определенной сеткой, на которой располагаются модели. У сцены есть только одна сторона, на которой можно размещать модели. Пользователь задает границы сцены, указывая количество квадратов в ширину и длину. Величина каждого квадрата константна и определяется программно. Цвет сцены - светло-зеленый. В программе присутствует ограничение на размер сцены.

### 1.3.2 Объекты сцены

1. **дома:** каждый дом формируется четырьмя стенами, крышей, окнами и дверью. В программе доступны два вида домов:

- «стандартный» дом: одноэтажное здание, занимающее три квадрата на сцене в длину, два квадрата в ширину;
- «премиальный» дом: двухэтажное сооружение с гаражом, занимающее три квадрата сцены в ширину и в длину. Гараж включает в себя стены, крышу и ворота.

**Компоненты дома:** каждый дом это цельный объект, но состоит из нескольких компонентов:

- стены: формируются вертикальными плоскостями, бежевого цвета;
- окна: расположены в середине стены, окрашены в голубой цвет;
- крыша: составлена из двух треугольников и двух трапеций, коричневого цвета;

- дверь: расположена посередине стены, коричневого цвета;
  - ворота: расположены на двух ячейках стены, коричневого цвета.
2. **дороги**: занимают один квадрат сцены по ширине и длине и темно-серый цвет;
  3. **тротуары**: по форме и размеру аналогичны дорогам, но окрашены в светло-серый цвет;
  4. **машины**: цельный объект из четырех колес, шести стекол и кузова. В программе имеется выбор модели машины, различного цвета.  
**Компоненты машины:**
    - *кузов*: формируется набором плоскостей, имитирующих реальные автомобили. Цвет кузова — красный или серый;
    - *колеса*: представляют собой цилиндрические объекты, черного цвета;
    - *Окна*: плоские объекты, встроенные в кузов автомобиля, голубого цвета.
  5. **деревья**: состоят из листвы и ствола, представляют цельный объект сцены и занимают одну клетки сцены. Листва имеет зеленый цвет, а ствол — коричневый;
  6. **кусты**: схожи с деревьями, но без ствола. Занимают одну клетку сцены, листва имеет светло-зеленый оттенок.

Все возможные модели заранее определены, при этом программа не позволяет добавлять новые модели или модифицировать существующие. Однако, в программном обеспечении предусмотрена возможность перемещения или удаления объектов на сцене. Также имеется ограничения на размещение моделей на сцене, а именно, машина может размещаться только на дороге. Дома, дороги, тротуары, деревья и кусты могут размещаться только на свободных участках сцены.

### 1.3.3 Источник света

Для моделирования освещения в компьютерной графике обычно используются три основных типа источников света: точечные, направленные и общий свет (Ambient Light) [5]. В данной работе выбор сделан в пользу точечного источника света.

Точечный источник света излучает свет равномерно во все стороны из определенной точки в 3D-пространстве, что позволяет эффективно управлять освещением и тенями на сцене, учитывая положение объектов относительно источника света. Положение источника света устанавливается относительно текущей точки наблюдения с помощью последовательных поворотов по осям X и Y [5].

## 1.4 Алгоритмы удаления скрытых линий и поверхностей

В компьютерной графике алгоритмы удаления невидимых линий и поверхностей относятся к наиболее сложным задачам. Это связано с тем, что они помогают определить видимые для наблюдателя элементы сцены из определенной точки пространства. Сложность этой задачи приводит к тому, что идеального общего решения не существует. Вместо этого, для каждой конкретной задачи требуется подбирать наиболее эффективный подход [6].

Алгоритмы удаления невидимых частей сцены можно классифицировать на основании следующих критериев:

1. *выбор удаляемых частей*: возможно удаление невидимых линий, ребер, поверхностей, объемов;
2. *порядок обработки элементов сцены*: удаление может происходить в порядке, определяемом процессом визуализации, или в произвольном порядке;
3. *зависимость от системы координат*: существуют алгоритмы, работающие в пространстве объектов, где каждая из  $N$  граней объекта сравнивается с остальными  $N - 1$  гранями (объем вычислений растет как  $N^2$ ); а также алгоритмы, работающие в пространстве изображения, когда для каждого пикселя изображения определяется, какая из  $N$  граней объекта видна (при разрешении экрана  $M \times M$  объем вычислений растет как  $M^2 \times N$ ) [6].

### 1.4.1 Алгоритм Робертса

Алгоритм Робертса (1963 год) используется для работы с выпуклыми объектами в 3D пространстве. Объекты представляются как многогранники, полученные пересечением плоскостей. Каждый объект описывается

списком граней, образованных ребрами и вершинами. Перед определением видимости удаляются нелицевые плоскости. Затем происходит сравнение каждого ребра с каждым объектом, что ведет к значительному объему вычислений. Таким образом, объем вычислений растет как квадрат числа объектов в сцене. В завершении вычисляются новые ребра от пересечения объектов. Преимущества — использование мощных и точных математических методов, также алгоритм прост для понимания. [6].

#### 1.4.2 Алгоритм плавающего горизонта

Алгоритм плавающего горизонта чаще всего применяется для удаления невидимых линий при трехмерном представлении функций, описывающих поверхность в виде  $F(x, y, z) = 0$  [6].

Однако, так как в данной программе поверхности не представлены в виде функций, данный метод не подходит.

#### 1.4.3 Z-буферный алгоритм удаления поверхностей

Был введен Эдом Кэтмулом и представляет собой расширение обычного буфера кадра, который хранит цвета каждого пикселя. В Z-буферном алгоритме, дополнительно к каждому пикселю присваивается значение глубины или Z-координата. При добавлении нового пикселя в буфер, сравнивается его Z-координата с той, что уже находится в буфере. Если новый пиксель ближе к наблюдателю (т.е. его Z-координата больше), его атрибуты и Z-координата записываются в буфер. Пример работы алгоритма на рисунке 1.7. Этот алгоритм прост в использовании, но требует большого объема памяти для реализации [6].

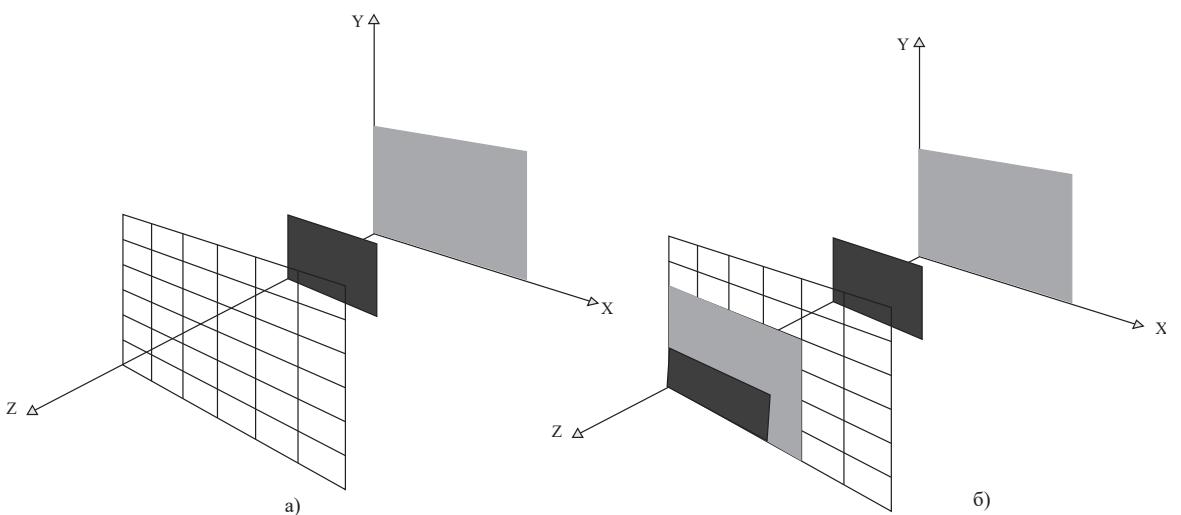


Рисунок 1.7 – Работа алгоритма с Z-буфером

### **Преимущества Z-буферного алгоритма:**

- поддерживает сцены произвольной сложности;
- отсутствует необходимость в сортировке, что требуется в других алгоритмах;
- вычислительная сложность линейно зависит от числа анализируемых поверхностей.

### **Недостатки Z-буферного алгоритма:**

- требует значительного объема памяти для буферов;
- реализация эффектов прозрачности и освещения может быть трудоемкой.

#### **1.4.4 Алгоритм, использующий список приоритетов.**

Алгоритмы, использующие список приоритетов, предполагают предварительную сортировку элементов по глубине. Запись в буфер кадра происходит начиная с наиболее удаленных от наблюдателя элементов. Ближе расположенные элементы последующим образом перезаписывают информацию в буфере. Взаимодействие прозрачных элементов с буфером осуществляется через частичную корректировку его содержимого, учитывая атрибуты прозрачности. Подход известен как «алгоритм художника». Это аналогия с тем, как художник создает картину: сначала фон, затем средний план, и в конце передний план. Алгоритм художника — простейший программный вариант решения «проблемы видимости» [5, 6].

Однако, этот подход имеет недостатки:

- *масштабируемость*: временная сложность алгоритма художника сильно зависит от алгоритма сортировки. Это делает подход более подходящим для малых сцен;
- *требование к памяти*: алгоритм требует информацию обо всем списке треугольников, что потребляет значительное количество памяти и исключает возможность потоковой реализации рендеринга;
- *неопределенность порядка*: алгоритм может дать сбой в некоторых случаях, включая циклическое перекрытие или проникновение многоугольников [5].

### 1.4.5 Алгоритм Варнока

Алгоритм Варнока использует принцип когерентности, по которому большие области изображения однородны, и работает в пространстве изображения. Основная идея - разбивать окно на подокна до тех пор, пока их содержимое не станет достаточно простым для визуализации, или размер подокна не достигнет заданного предела разрешения [6, 7]

Конкретная реализация зависит от метода разбиения окна и критерия, определяющего простоту содержимого окна.

- в оригинальной версии алгоритма окно делится на четыре равных по размеру подокна;
- вариант Вейлера и Азерттона предполагает разбиение окна по ребрам изображаемых многоугольников [7].

Эффективность данного алгоритма зависит от сложности сцены. Использование полигональной сетки может замедлить его выполнение.

### 1.4.6 Алгоритм прямой и обратной трассировки лучей

Методы прямой и обратной трассировки лучей используются для отслеживания траектории лучей от источника света до камеры, учитывая взаимодействие с объектами на пути.

- **прямая трассировка лучей:** строит траектории лучей от всех источников света до всех точек сцены, включая те, которые не попадают в камеру. Из-за этого, метод считается неэффективным;
- **обратная трассировка лучей:** эффективнее, учитывая взаимное влияние объектов друг на друга. Позволяет рассчитывать тени, многократные отражения и преломления. Согласно этому методу отслеживание лучей осуществляется не от источников света, а в обратном направлении — от точки наблюдения. Так учитываются только те лучи, которые вносят вклад в формирование изображения, примерная схема обратной трассировки лучей на рисунке 1.8. Однако имеет недостатки в виде неучета вторичного освещения, высокой вычислительной стоимости, резких границ цветовых переходов и дискретности первичных лучей [8].

## 1.4.7 Выбор алгоритма удаления скрытых линий и поверхностей

Для наглядности эффективности алгоритмов была составлена таблица 1.1, в строке : «Вычислительная сложность»  $N$  и  $C$  означают, количество граней и количество пикселей, соответственно.

Таблица 1.1 – Сравнение алгоритмов удаления невидимых линий и поверхностей.

	z-буфер	Обратная трассировка лучей	Варнок	Робертс
Вычисл. сложность	$O(CN)$	$O(CN)$	$O(CN)$	$O(N^2)$
Рабочее пространство	Изображение	Изображение	Изображение	Объектное
Трудность реализации	Низкая	Средняя	Средняя	Высокая
Производит. при сложной сцене	Высокая	Низкая	Средняя	Низкая
Распростран. в современном ПО	Высокая	Высокая	Низкая	Низкая
Использов. рекурсивных вызовов	Нет	Да	ДА	Нет

На основании рассмотренных алгоритмов, я пришел к выводу, что Алгоритм Z-буфера представляет собой оптимальный выбор для удаления невидимых линий и поверхностей. Он обеспечивает быструю обработку сцен любой сложности и адаптируется под изменения в освещении и закраске. При небольшом размере изображения Z-буфер эффективно обрабатывает большое количество объектов без проблем с памятью. Благодаря простоте понимания и отладки, алгоритм Z-буфера является доступным и эффективным решением для данной задачи.

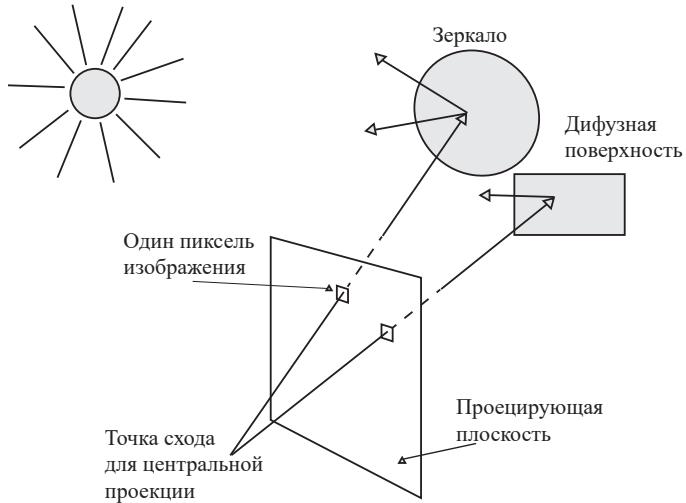


Рисунок 1.8 – Схема обратной трассировки лучей

## 1.5 Модель освещения

Модели освещения классифицируются на глобальные и локальные. Глобальные модели анализируют отражение и преломление света от объектов, не являющихся прямыми источниками освещения, что требует значительных вычислительных затрат. В данной работе применяются локальные модели, ограничивающиеся учетом света только от прямых источников. Среди них выделяются модель Ламберта и модель Фонга [9].

### 1.5.1 Простая модель

Простейшая модель освещения складывается из трех основных компонентов светового воздействия, например рисунок 1.9.

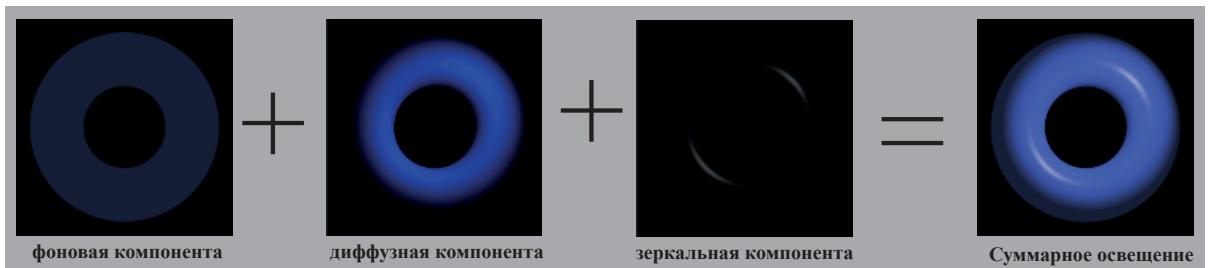


Рисунок 1.9 – Пример суммарного освещения.

Формула для расчета простой модели освещения [10]:

$$I = I_a + I_d + I_s \quad (1.1)$$

–  $I_a$  — фоновая компонента;

- $I_d$  — диффузная компонента;
- $I_s$  — зеркальная компонента.

*Фоновая компонента освещения* представляет собой константу, добавляемую к освещенности в каждой точке. Формула для расчета фоновой компоненты освещения приведена ниже [10] :

$$I_a = k_a \cdot i_a \quad (1.2)$$

- $I_a$  — фоновая компонента освещения в точке;
- $k_a$  — коэффициент, характеризующий способность материала воспринимать фоновое освещение;
- $i_a$  — интенсивность фонового освещения.

Как видно из приведенного выше уравнения, фоновая компонента освещения не зависит от пространственных координат освещаемой точки и источника света.

### 1.5.2 Модель Ламберта

Модель Ламберта описывает идеальное диффузное освещение, где свет рассеивается равномерно во все стороны после попадания на поверхность. [10] Она учитывает только ориентацию поверхности (вектор нормали  $N$ ) и направление на источник света  $L$ . Диффузионное составляющая освещенности вычисляется по закону Ламберта, с использованием косинуса угла между  $L$  и  $N$ , как показано ниже в формуле [9]:

$$I_{diff} = I_d \cdot k_{diff} \cdot \cos \cdot (\vec{L} \cdot \vec{N}) \quad (1.3)$$

- $I_{diff}$  — диффузная составляющая освещенности;
- $I_d$  — интенсивность рассеянного света;
- $k_{diff}$  — коэффициент диффузного отражения;
- $\vec{L}$  — вектор, направлен из точки на источник;
- $\vec{N}$  — вектор нормали к точке.

Однако, как показывает формула 1.3, у этой модели есть значительный недостаток: она предполагает одинаковую интенсивность света на всей грани, не учитывая положение наблюдателя

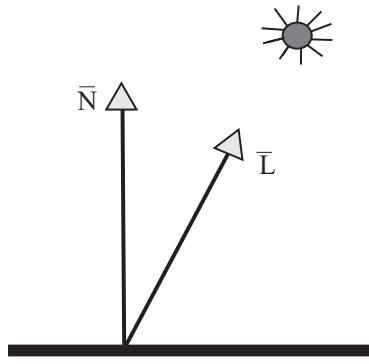


Рисунок 1.10 – Модель Ламберта

### 1.5.3 Модель Фонга

Модель Фонга, объединяющая диффузную (по Ламберту) и зеркальную составляющие, обеспечивает не только равномерное освещение материала, но и возможность появления блика, местоположение которого определяется законом равенства углов падения и отражения [9]. Яркость увеличивается, когда наблюдатель находится близко к углам отражения. Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения, и эта нормаль делит угол между лучами на две равные части как показано на рисунке 1.11.

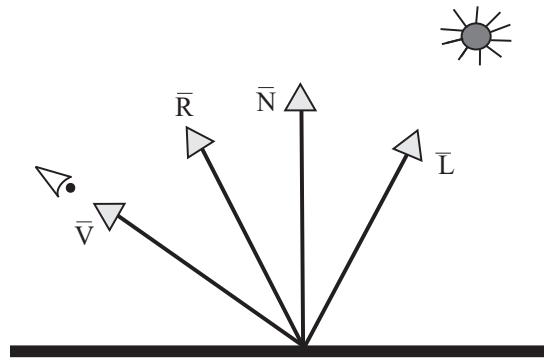


Рисунок 1.11 – Модель Фонга

$$I_s = k_s \cdot \cos^\alpha(\vec{R}, \vec{V}) \cdot i_s = k_s \cdot (\vec{R}, \vec{V})^\alpha \cdot i_s \quad (1.4)$$

- $I_s$  — зеркальная компонента освещенности;
- $k_s$  — коэффициент зеркального отражения;
- $i_s$  — интенсивность зеркального освещения;
- $\vec{R}$  — вектор отраженного луча;

- $\vec{V}$  — вектор направления к наблюдателю;
- $\alpha$  — коэффициент блеска, характеризующий материал поверхности.

Использование модели Ламберта в моей работе обусловлено её вычислительной эффективностью и простотой, что позволяет быстро и точно моделировать диффузное освещение.

## 1.6 Алгоритмы построения теней

### 1.6.1 Модификация Z-буфера

В алгоритмах отображения сцены можно адаптировать Z-буфер для отслеживания теней. Изначально, этот буфер хранит данные о глубине каждого пикселя, но для точного отображения теней нужен дополнительный этап.

Первым шагом камера устанавливается на место источника света, позволяя занести данные о глубине каждой точки в теневой буфер. Это гарантирует, что все отмеченные точки расцениваются как освещенные [11].

Далее формируется изображение с точки зрения наблюдателя. На этом этапе каждый пиксель анализируется с учетом его расстояния до источника света и сопоставляется с данными из теневой карты. Если расстояние до пикселя больше значения из теневой карты, пиксель считается находящимся в тени и отображается с акцентом на рассеянном свете, это можно увидеть на примере рисунка 1.12, точка  $B$  имеет большее расстояние до источника света, чем восстановленный из карты теней, следовательно точка  $B$  будет в тени, а точка  $A$  находится на том же расстоянии от источника света, что и восстановлено из карты теней и не будет затемнена. В итоге, модификация Z-буфера позволяет эффективно учсть тени при рендеринге сцены, одновременно выполняя удаление невидимых поверхностей [11].

### 1.6.2 Модификация Вейлера-Азертонса

На первом шаге с помощью алгоритма удаления невидимых поверхностей выбираются освещенные грани, т. е. грани, которые видны из положения источника света. Для увеличения эффективности, в памяти хранятся именно эти грани, что позволяет избежать удвоения обрабатываемых граней для выпуклого многоугольника. Освещенные многоугольники помечаются и преобразуются к исходной ориентации, где они приписываются к своим прототипам в качестве многоугольников детализации поверхности. Чтобы избежать появления ложных теней, сцену рассматривают в преде-

лах видимого или отсекающего объема, определенного положением источника света [11].

На втором шаге объединенные данные о многоугольниках обрабатываются из положения наблюдателя. Если какая-то часть не освещена, применяется соответствующее правило затенения. Если источников света несколько, то используется несколько наборов освещенных граней [11].

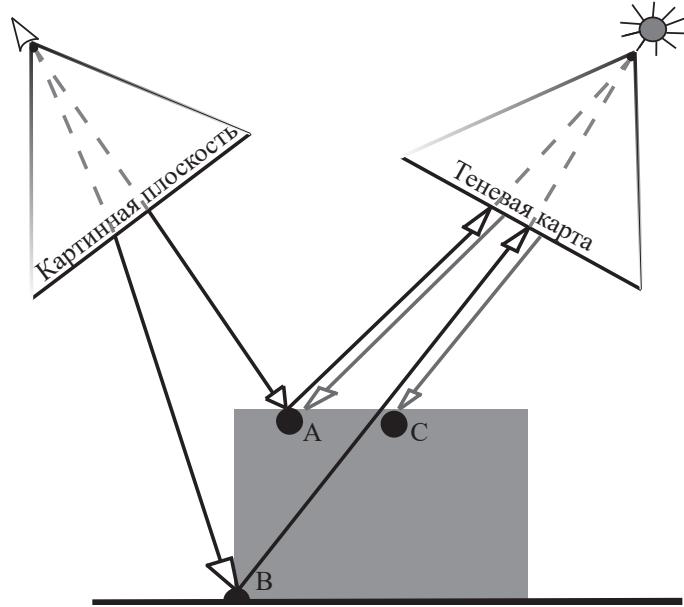


Рисунок 1.12 – Пример определения тени, алгоритмом модификации Z-буфера.

Выбор модифицированного Z-буфера для построения теней в данном контексте является оптимальным. Это обусловлено его эффективностью и совместимостью с уже используемым Z-буфером для удаления невидимых поверхностей. Использование одной и той же технологии упрощает интеграцию и обеспечивает более быструю и эффективную обработку графической информации.

## 1.7 Вывод

Анализ подходов к заданию трехмерных моделей привел к выбору поверхностных моделей, метода полигональных сеток в качестве способа представления трехмерных поверхностей. Для удаления невидимых ребер применен алгоритм Z-буфера. Модификация Z-буфера использована для построения теней, обеспечивая совместимость и упрощенную интеграцию.

## 2 Конструкторская часть

В данном разделе представлены требования к программному обеспечению, а также схемы алгоритмов, выбранных для решения поставленной задачи.

**Программа должна предоставлять следующий функционал:**

- создание сцены с определенным размером;
- размещение, удаление и перемещение отдельных объектов сцены;
- добавление источника света;
- поворот, перемещение и масштабирование сцены с объектами.

**Программа должна соответствовать следующим критериям:**

- Для обеспечения плавной и корректной интерактивной работы необходимо, чтобы время, в течение которого программа реагирует на действия пользователя, составляло менее одной секунды.
- обеспечение корректного реагирования на все действия пользователя.

### 2.1 Z-буферный алгоритм

#### 2.1.1 Формальная запись

1. инициализировать кадровый буфер и  $z_{\text{буфер}}$ ;
2. кадровый буфер заполнить фоном и  $z_{\text{буфер}}$  заполнить минимальным значением  $z$ ;
3. для каждого многоугольника сцены:
  - 3.1 для каждого пикселя  $(x, y)$  в многоугольнике вычислить его глубину  $z(x, y)$ ;
    - i. сравнить глубину  $z(x, y)$  со значением  $z_{\text{буфер}}$ , хранящимся в  $z_{\text{буфер}}$  в этой же позиции.  
Если  $z(x, y) > z_{\text{буфер}}(x, y) \Rightarrow z_{\text{буфер}}(x, y) = z(x, y)$  и записать атрибут этого многоугольника (интенсивность, цвет и т. п.);

- ii. в противном случае никаких действий не производить;
4. вывести изображение.

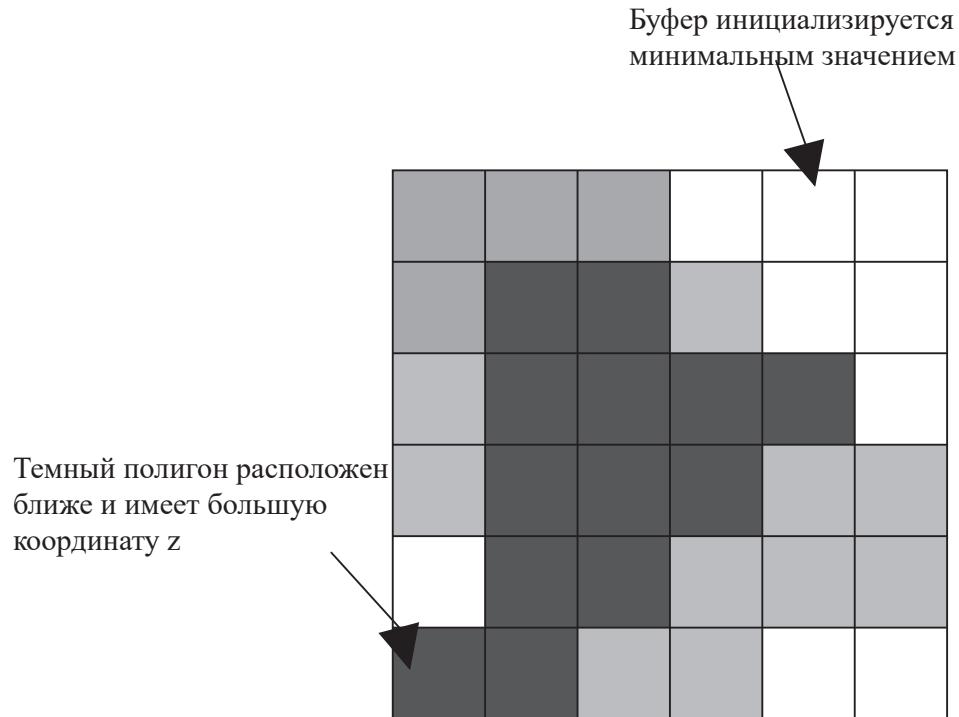


Рисунок 2.1 – Визуализация состояния Z-буфера

## Алгоритм Z-буфера

**входные данные:**

буфер кадра = БК, буфер глубины = БГ,  
массив с многоугольниками сцены, атрибуты  
многоугольников.

**выходные данные:** финальное  
изображение

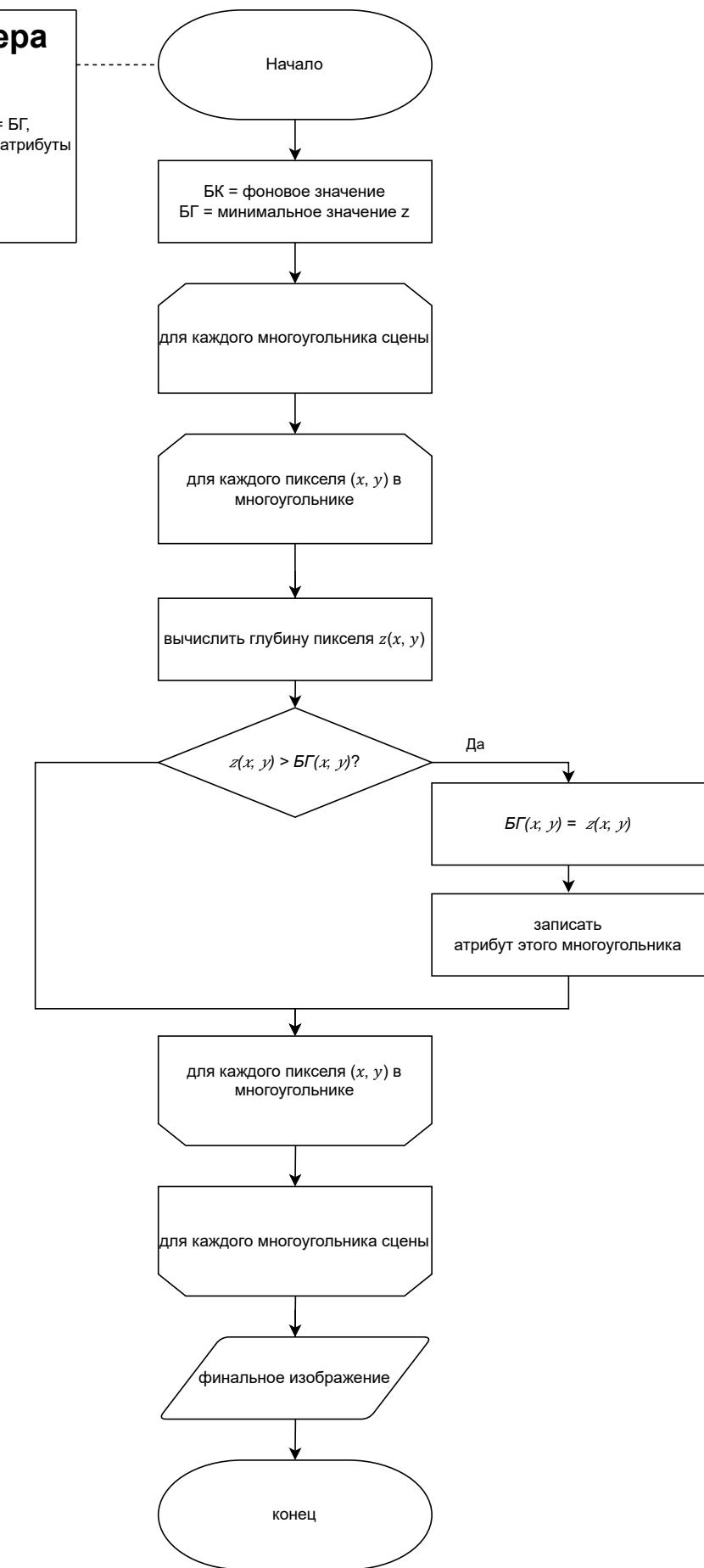


Рисунок 2.2 – Схема алгоритма Z-буфера

## 2.2 Модифицированный Z-буферный алгоритм

### 2.2.1 Формальная запись

1. для каждого источника света.
  - 1.1 инициализировать Z-буфер (буфер глубины) и буфер теней, в дальнейшем БГ и БТ, соответственно;
  - 1.2 заполнить БГ минимальными значениями глубины. Заполнить БТ значениями, указывающими отсутствие тени.
2. выполнить Z-буферный алгоритм для точки наблюдения, параллельно проверяя видимость поверхности от текущей точки наблюдения и источников света;
3. для каждого источника света;
  - 3.1 преобразовать координаты рассматриваемой точки, наблюдателя  $(x, y, z)$  в координаты точки источника света  $(x', y', z')$ ;
  - 3.2 если  $z'(x', y') < (x', y')$ , то пиксель высвечивается с учетом его затемнения;
  - 3.3 иначе точка высвечивается без затемнения.

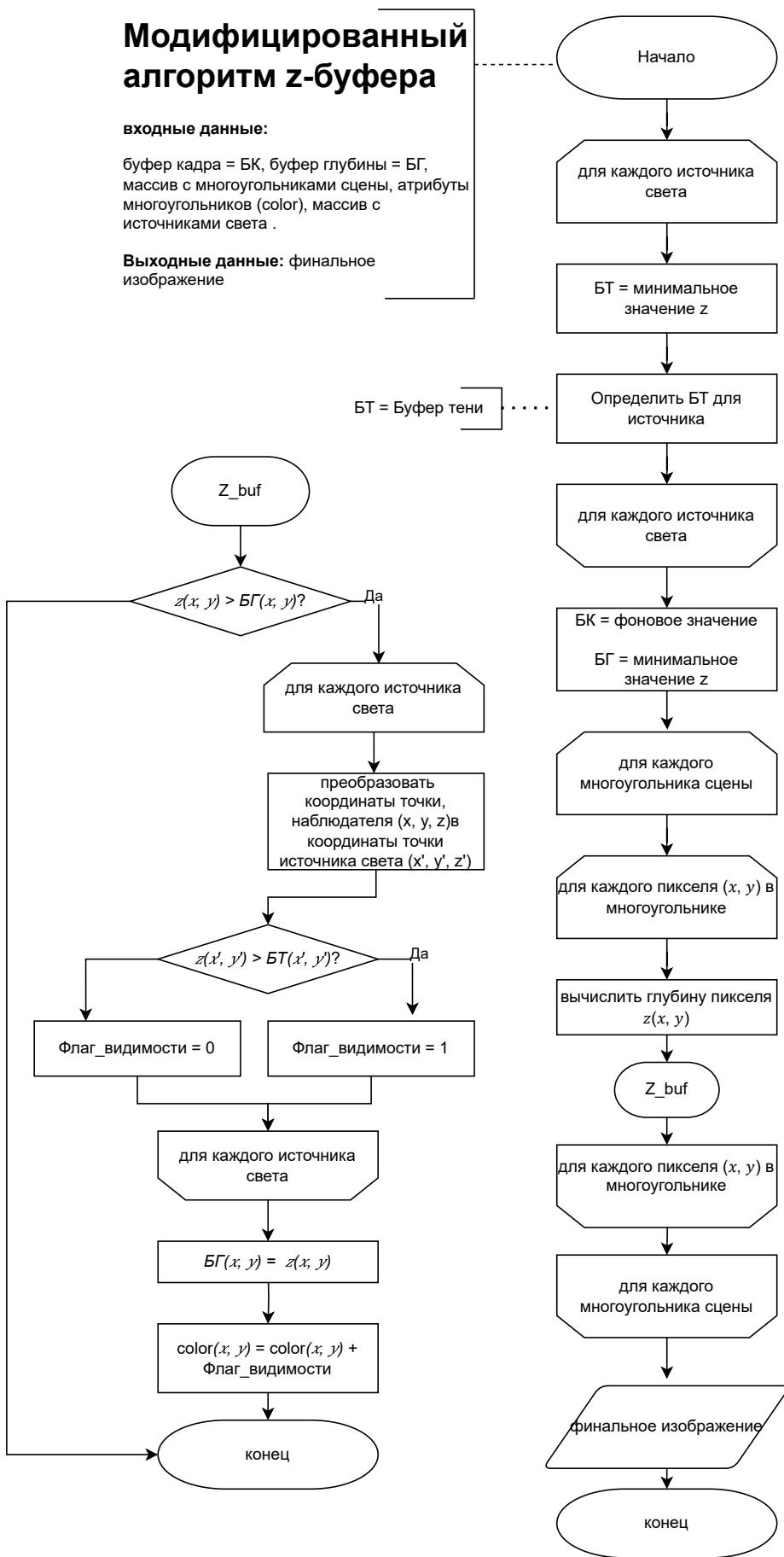


Рисунок 2.3 – Схема модифицированного алгоритма Z-буфера

## **2.3 Выбор используемых типов и структур данных**

1. точка задается координатами  $(x, y, z)$ ;
2. полигон задается тремя точками и храниться в массиве;
3. объект сцены, например: дом или дорога, задается полигонами в виде массива;
4. источник света задается углом, по осям  $X$  и  $Y$  относительно точки наблюдателя.

## **2.4 Вывод**

**В этом разделе были разобраны следующие элементы:**

1. требования к разрабатываемому программному обеспечению;
2. разработана схема разрабатываемого алгоритма;
3. описаны структуры данных, которые будут использоваться при реализации программного обеспечения.

## 3 Технологическая часть

В данном разделе представлены средства разработки программного обеспечения, детали реализации и тестирование функций.

### 3.1 Средства реализации

В качестве языка программирования для разработки программного обеспечения был выбран язык программирования Rust[?]. Данный выбор обусловлен тем, что данный язык предоставляет весь требуемый функционал для решения поставленной задачи, а также обладает связанным с ним пакетным менеджером Cargo[?], который содержит инструменты для тестирования разрабатываемого ПО[?].

Для создания пользовательского интерфейса программного обеспечения была использована библиотека gtk-rs[?]. Данная библиотека содержит в себе объекты, позволяющие напрямую работать с пикселями изображения, а также возможности создания панели управления с кнопками, что позволит в интерактивном режиме управлять изображением.

Для тестирования программного обеспечения были использованы инструменты пакетного менеджера Cargo[?], поставляемого вместе с компилятором языка при стандартном способе установке, описанном на официальном сайте языка[?].

В процессе разработки был использован инструмент RLS[?] (англ. *Rust Language Server*), позволяющий форматировать исходные коды, а также в процессе их написания обнаружить наличие синтаксических ошибок и некоторых логических, таких как, например, нарушение правила владения[?].

В качестве среды разработки был выбран текстовый редактор VIM[?], поддерживающий возможность установки плагинов[?], в том числе для работы с RLS[?].

### 3.2 Реализация алгоритмов

В листинге ?? представлена структура объекта мышцы, а также реализация методов деформации и триангуляции. В листинге ?? представлена реализация алгоритмов компьютерной графики: *z*-буфера и Гуро.

## **Вывод**

В данном разделе были рассмотрены средства, с помощью которых было реализовано ПО, а также представлены листинги кода с реализацией объекта мышцы и алгоритмов компьютерной графики.

## **4 Исследовательская часть**

В данном разделе будут приведены примеры работы разработанного программного обеспечения, а также будет поставлен эксперимент, в котором будут сравнены геометрические характеристики разработанной модели с геометрическими характеристиками бицепса реального человека.

### **4.1 Результаты работы программного обеспечения**

В листинге ?? описана конфигурация модели мышцы с рисунков ?? и ??, на рисунках ?? и ?? представлен пример модели мышцы в полностью растянутом и сокращённом состояниях соответственно, на рисунке ?? представлено окно панели управления.

Рисунок 4.1 – Пример модели мышцы в полностью растянутом состоянии.

Рисунок 4.2 – Пример модели мышцы в полностью сокращённом состоянии.

Рисунок 4.3 – Окно панели управления.

## 4.2 Постановка эксперимента

### 4.2.1 Цель эксперимента

Цель эксперимента - сравнение геометрических характеристик (радиусов узлов) разработанной модели и реального человеческого бицепса в различных положениях мышцы.

### 4.2.2 Данные реального бицепса

Данные реального бицепса были взяты из проекта OpenArm 2.0 [?]. Данный проект предоставляет необработанные и сегментированные ультразвуковые сечения плечевой части руки (в том числе бицепса) [?]. Анализ данных может быть произведен с помощью ПО ITK-SNAP [?]. На рисунке ?? приведён пример необработанных данных, а на рисунке ?? - пример сегментированных данных.

Рисунок 4.4 – Необработанные данные.

Рисунок 4.5 – Сегментированные данные.

### 4.2.3 Замеры реального бицепса

На рисунках ?? – ?? приведены замеры для 9 узлов мышцы в 4 положениях: под углом  $0^\circ$ ,  $30^\circ$ ,  $60^\circ$  и  $90^\circ$ . Данные замеров сведены в таблицу 4.1.

Рисунок 4.6 – Замеры для мышцы в положении  $0^\circ$ .

Рисунок 4.7 – Замеры для мышцы в положении  $30^\circ$ .

Рисунок 4.8 – Замеры для мышцы в положении  $60^\circ$ .

Рисунок 4.9 – Замеры для мышцы в положении  $90^\circ$ .

Номер узла	$0^\circ$	$30^\circ$	$60^\circ$	$90^\circ$
1	7.6	16.2	7.8	9.5
2	18.2	18.7	16.2	15.2
3	20.7	22.5	24.9	23.4
4	21.2	26.2	32.4	27.7
5	21.7	27.5	32.4	35.2
6	24.3	31.3	32.0	35.8
7	25.7	28.7	29.8	36.6
8	20.7	26.6	22.0	24.2
9	13.2	20.8	12.7	16.7

Таблица 4.1 – Радиусы узлов при различном угле наклона в локте

#### 4.2.4 Замеры разработанной модели

Для измерения геометрических характеристик модели необходимо сперва ее сконфигурировать. Конфигурация будет выполняться исходя из данных реальной мышцы: начальные значения узлов модели пропорциональны начальным значениям узлов реальной мышцы, коэффициенты роста пропорциональны приросту реальной мышцы при переходе из начального в конечное состояние.

Таким образом конфигурация будет иметь вид, описанный в листинге ??.

Вид модели представлен на рисунках ?? – ???. Результаты представлены в таблице 4.2.

Рисунок 4.10 – Модель в положении  $0^\circ$ .

Рисунок 4.11 – Модель в положении  $30^\circ$ .

Рисунок 4.12 – Модель в положении  $60^\circ$ .

Рисунок 4.13 – Модель в положении  $90^\circ$ .

Номер узла	$0^\circ$	$30^\circ$	$60^\circ$	$90^\circ$
1	7.6	7.7	7.8	8.0
2	18.2	18.1	17.9	17.6
3	20.7	20.8	21.0	21.2
4	21.2	21.3	21.7	22.3
5	21.7	22.1	23.0	24.4
6	24.3	24.6	25.4	26.6
7	25.7	26.0	26.7	27.9
8	20.7	20.8	21.0	21.4
9	13.2	13.3	13.5	13.9

Таблица 4.2 – Радиусы узлов модели при различном угле наклона в локте

Номер узла	РБ	модель
1	1.9	0.4
2	-3.0	-0.6
3	2.7	0.5
4	6.5	1.1
5	13.5	2.7
6	11.5	2.3
7	10.9	2.2
8	3.5	0.7
9	3.5	0.7

Таблица 4.3 – Радиусы узлов при различном угле наклона в локте

#### 4.2.5 Сравнение замеров

В таблице 4.3 приведены приросты для всех узлов для реального бицепса (РБ) и модели относительно при переходе из состояния  $0^\circ$  в состояние  $90^\circ$ .

Из таблицы видно, что реализованная модель может сохранять задаваемые пропорции, но ввиду того, что модель сохраняет объем, все приращения получились в 5-6 раз меньше.

Из таблиц 4.1 и 4.2 можно заметить, что приращения модели при сокращении мышцы строго положительные, в то время как с реальным бицепсом могут быть как положительные, так и отрицательные.

## Вывод

В данном разделе были рассмотрены примеры работы программного обеспечения, а также были вычислены и сравнены радиусы узлов реального бицепса и сделанной модели. В результате сравнения были получены следующие результаты:

- Модель способна сохранять пропорции приращения, заданные реальной мышцой.
- Модель ведёт себя монотонно, то есть, например, при сокращении радиусы не перестают расти (или убывать, если коэффициент роста меньше 0), в то время как узел в реальном бицепсе при сокращении может как увеличиться, так и уменьшиться.
- Ввиду ограничения, связанного с постоянством объема, радиусы узлов модели получают прирост в 5-6 раз меньший, нежели узлы реальной мышцы.

# Заключение

В ходе данной практической работы была успешно достигнута поставленная цель — освоены базовые принципы и алгоритмы, необходимые для создания программного обеспечения для визуализации и макетирования загородной местности.

Одним из важных этапов стал отбор подходящих структур данных, обеспечивающих эффективное хранение объектов сцены. Ключевым моментом анализа подходов к моделированию трехмерных объектов стал выбор алгоритма Z-буфера для удаления невидимых ребер, а также его модификация для создания теней. Это обеспечило не только совместимость компонентов системы, но и упрощенную интеграцию.

Глубокий анализ алгоритмов и структур данных научил меня обдуманно и стратегически подходить к выбору конкретных методов решения задач. Этот процесс дал мне возможность лучше понимать, как каждый алгоритм может влиять на эффективность и функциональность разрабатываемого программного обеспечения, позволяя мне принимать обоснованные решения, направленные на оптимизацию и улучшение проекта.

# Литература

- [1] A. Lovett K. Appleton B. Warren-Kretzschmar. Using 3D visualization methods in landscape planning: An evaluation of options and practical issues // Landscape and Urban Planning. 2015. C. 85–94.
- [2] Лисяк В.В. Основы геометрического моделирования. 2022. С. 6–7.
- [3] В. Ю. Донченко Е.Н. Черевань. Обзор и анализ методов построения геометрических моделей сложных конструкций. Режим доступа: [https://dspace.lgpu.org/xmlui/bitstream/handle/123456789/4116/700-19\\_%20sb.pdf?sequence=1&isAllowed=y#page=17](https://dspace.lgpu.org/xmlui/bitstream/handle/123456789/4116/700-19_%20sb.pdf?sequence=1&isAllowed=y#page=17) (дата обращения: 19.07.2023). 2014.
- [4] А.В. Киселёв Г.Н. Верхотурова. Способы представления и размещения трехмерных моделей для прототипирования ювелирных изделий // Материалы VI Международной научно-практической конференции (школы-семинара) молодых ученых. 2020. С. 840–842.
- [5] Gambetta G. Computer Graphics from Scratch. 2021. С. 47–50.
- [6] Головнин А.А. Базовые алгоритмы компьютерной графики. 2016. С. 12–16.
- [7] Провкин В. А. Построение реалистичных изображений при помощи алгоритма Варнока и сравнение эффективности различных его реализаций. 2017.
- [8] Янова Р. Д. Метод прямой и обратной трассировки // Вестник науки и образования. 2016. С. 29–30.
- [9] Задорожный А.Г. Модели освещения и алгоритмы затенения в компьютерной графике. 2020. С. 27–35.
- [10] Чернявская А. Э. Простые модели освещения 3D-объектов. Особенности цифрового моделирования света // Современные вопросы науки и практики. 2021. С. 45 – 47.
- [11] Куроев А.В. Конспект лекций по дисциплине компьютерная графика. 2023.