СОДЕРЖАНИЕ

ОПРЕДЕЛЕНИЯ

Рендеринг (англ. Rendering) – конечный процесс создания реального 2D-изображения или анимации из подготовленной сцены.

Грань – замкнутое множество рёбер, в котором треугольная грань имеет три ребра, а четырёхугольная — четыре.

Полигон – набор компланарных граней.

ВВЕДЕНИЕ

Компьютерная графика является одной из самых динамично развивающихся областей информационных технологий, играющей ключевую роль в современном обществе. Она охватывает широкий спектр приложений, включая игры, анимацию, визуализацию данных, дизайн, архитектуру и медицину.

Целью данной курсовой работы является разработка программы композиции и визуализации трехмерных многогранных примитивов с учётом их геометрических и оптических параметров, вводимых пользователем. Для удобства восприятия должны присутствовать эффект глубины визуализируемой сцены, опции перемещения камеры, настройка положения источника света и возможность изменения его спектральных характеристик.

Чтобы достичь данной цели, необходимо выполнить следующие задачи:

– описать объекты сцены;

– проанализировать существующие алгоритмы компьютерной графики для генерации реалистичных моделей и трехмерной сцены;

– выбрать наиболее подходящие алгоритмы для достижения поставленной цели;

– спроектировать архитектуру и графический интерфейс приложения;

– выбрать средства реализации программного обеспечения;

– реализовать выбранные алгоритмы и структуры данных;

– провести исследование быстродействия разработанного приложения.

1 Аналитическая часть

В этом разделе рассматриваются объекты сцены, способы их представления и существующие алгоритмы создания изображений, а также осуществляется выбор наиболее подходящих из этих алгоритмов для дальнейшего применения.

1.1 Формализация объектов сцены

Сцена является набором объектов, включащим в себя:

1) объекты сцены – геометрические многогранные примитивы, расположенные в пространстве сцены, такие как, куб, прямая призма треугольная пирамида. Каждый примитив состоит из граней, сформированных из точек соединенных ребрами. Геометрические характеристики примитивов задаются параметрами такими, как длина ребра основания, высота, радиусы описанных окружностей нижнего и верхнего основания, количество боковых граней. Спектральные характеристики так же задаются параметрами такими, как коэффициенты отражения и блеска, цвет.
2) источник света – пространственный вектор, указывающий направление света. Свет всегда направлен вдоль положительной оси Z, его положение в пространстве сцены задается пользователем.
3) камера – пространственный вектор, указывающий направление просмотра.

1.2 Анализ способов представления модели

1) *каркасная модель* – простейший способ представления моделей, основанный на использовании точек и рёбер, содержащий минимум иформации и имеющий ряд ограничений. Основным недостатком является неоднозначность, так как невозможно четко определить ориентацию и видимость, что может привести к непредсказуемым результатам. Удаление скрытых линий требует ручного редактирования, что может разрушить целостность модели. Также каркасные модели не поддерживают технику тонового изображения, так как затенение применяется к граням, а не к ребрам, что ограничивает их использование в задачах, требующих визуальной глубины и реалистичности. Несмотря на эти ограничения, каркас-

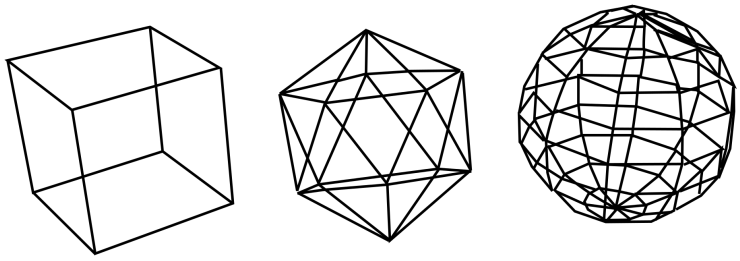

Рисунок 1.1 — Каркасные модели

ные модели требуют меньше памяти и легки в отрисовке. Пример каркасных моделей приведён на рисунке 1.1.

2) *поверхностная модель* – способ представления моделей, основанный на использовании точек, линий и поверхностей, что делает его более гибким и многофункциональным по сравнению с каркасной моделью. Он позволяет изображать сложные криволинейные грани, создавать тоновые трехмерные изображения и выявлять особенности, такие как отверстия. Этот метод обеспечивает высокое качество изображений. Поверхностную модель можно задать двумя способами:

   – *параметрическое представление* – вид поверхностной модели, требующий вычисления функции, зависящей от параметра, но его использование затруднено в сценах без поверхностей вращения;

   – *полигональная сетка* – вид поверхностной модели, представленный совокупностью вершин, рёбер и граней. Гранями обычно являются треугольники, четырёхугольники или другие простые выпуклые многоугольники, но могут также являться многоугольниками с отверстиями. Пример поверхностной модели, заданной полигональной сеткой, приведён на рисунке 1.2.

Однако у поверхностной модели есть недостаток: она не предоставляет информации о том, с какой стороны поверхности находится материал.

3) *твердотельная модель* – способ представления моделей, включающий информацию о внутренней структуре и расположении материала, в отличие от поверхностной модели, что достигается указанием направления внутренней нормали. Твердотельные модели обладают неоспоримыми преимуществами, включая полное определение объемной формы с разграничением внешних и внутренних областей, что позволяет избежать

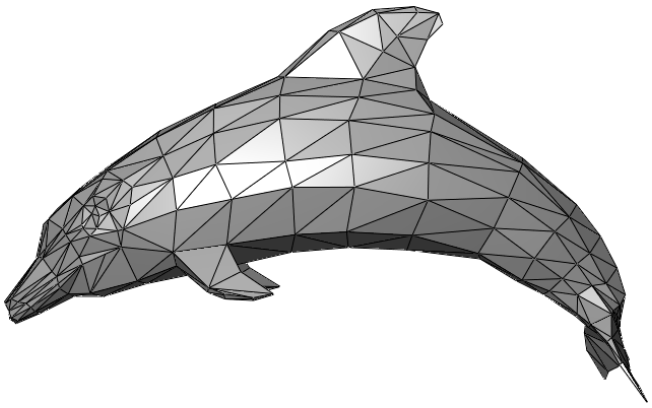

Рисунок 1.2 — Поверхностая модель, заданная полигональной сеткой

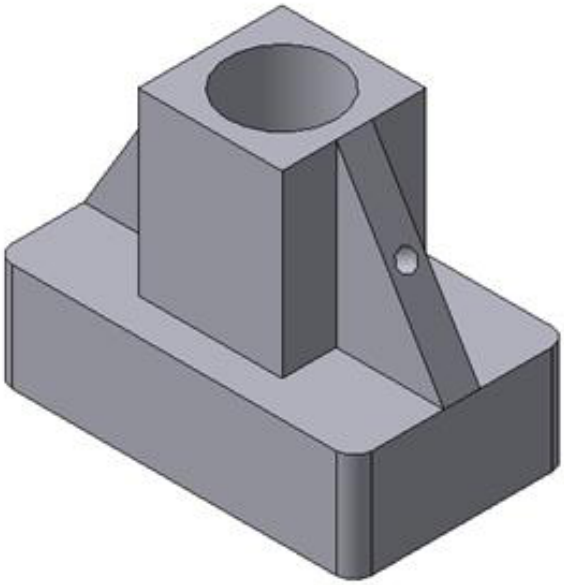

Рисунок 1.3 — Твердотельная модель

нежелательных взаимовлияний компонентов. Они обеспечивают автоматическое удаление скрытых линий и построение трехмерных разрезов, что важно для анализа сложных композиций. Пример твердотельной модели приведён на рисунке 1.3.

Для выполнения данной работы каркасная модель не подходит, так как она может исказить восприятие форм объекта. Твердотельная модель так же не является оптимальной, поскольку в этой работе не требуется информация о материале и его расположении. Таким образом, остается только поверхностная модель как наилучший выбор. Поверхностная модель будет задана полигональной сеткой, так как в этой работе не будет обработки поверхностей вращения и использование параметрического представления избыточно.

1.3 Анализ способов представления поверхностной модели, заданной полигональной сеткой

Объекты, созданные с помощью полигональных сеток, должны хранить разные типы элементов, такие как вершины, рёбра, грани, полигоны и поверхности. Во многих случаях хранятся лишь вершины, рёбра и либо грани, либо полигоны (рис. 1.4). Грани могут быть как трехсторонними, так и четырехсторонними.

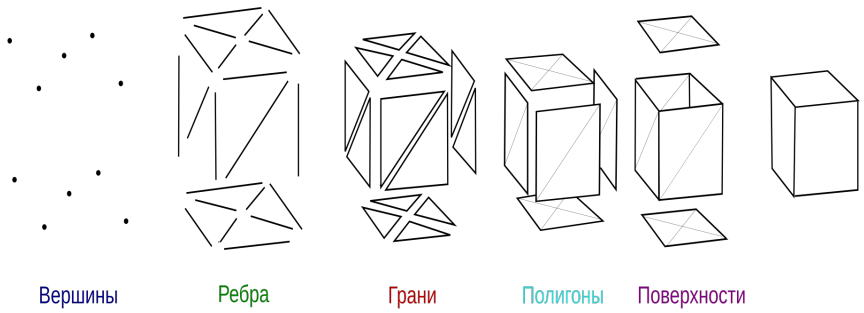

Рисунок 1.4 — Элементы моделирования сетки

1) *вершинное представление* – простейший способ представления, описывающий объект как набор вершин, соединённых с другими вершинами. Хотя это и простейший способ представления, он не так широко используется, поскольку не предоставляет явной информации о гранях и рёбрах. Это означает, что для генерации списка граней, необходимого для ренде-

Список вершин

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3

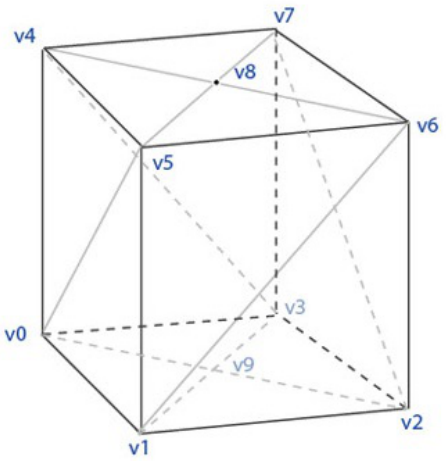

Рисунок 1.5 — Вершинное представление

ринга, нужно обойти все данные, что усложняет выполнение операций с рёбрами и гранями. Пример полигональной сетки, заданной вершинным представлением приведён на рисунке 1.5.

2) *список граней* – способ представления, описывающий объект как множество вершин и граней, который является наиболее распространённым представлением для современного графического оборудования. Это представление упрощает моделирование, позволяя легко находить грани, окружающие конкретные вершины. Пример полигональной сетки, заданной списком граней приведён на рисунке 1.6.

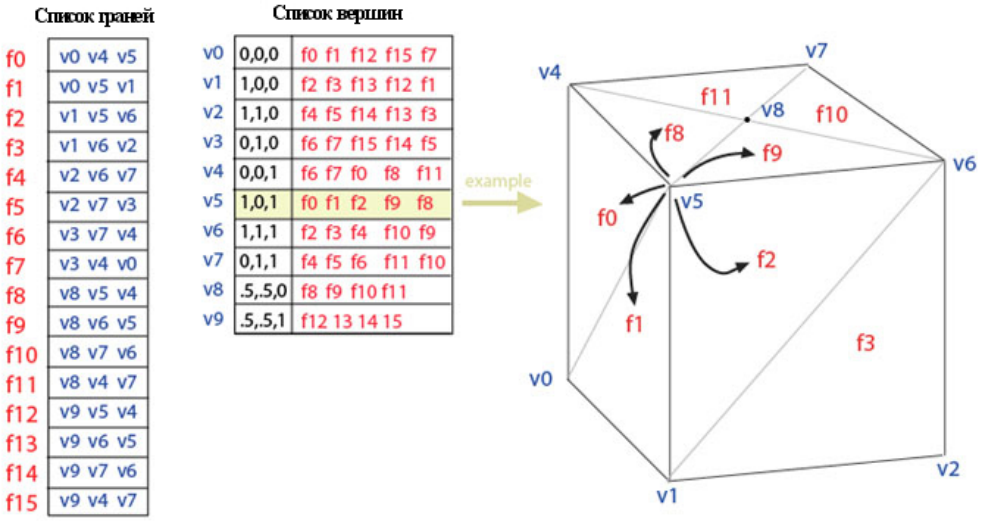

Рисунок 1.6 — Список граней

3) *«крылатое» представление* – способ представления, описывающий объект как множество вершин, граней и рёбер, что обеспечивает высокую гибкость в динамическом изменении геометрии. «Крылатое» представление эффективно решает задачу обхода от ребра к ребру, предоставляя упорядоченное множество граней вокруг каждого ребра. Оно включает информацию о двух конечных вершинах, двух гранях и четырёх ближайших рёбрах. Пример полигональной сетки, заданной «крылатым» представлением приведён на рисунке 1.7.

Для хранения полигональной сетки будет использован метод, основанный на списке граней, что обеспечивает ясное представление о гранях. Этот подход способствует эффективному преобразованию моделей, так как структура включает в себя список вершин.

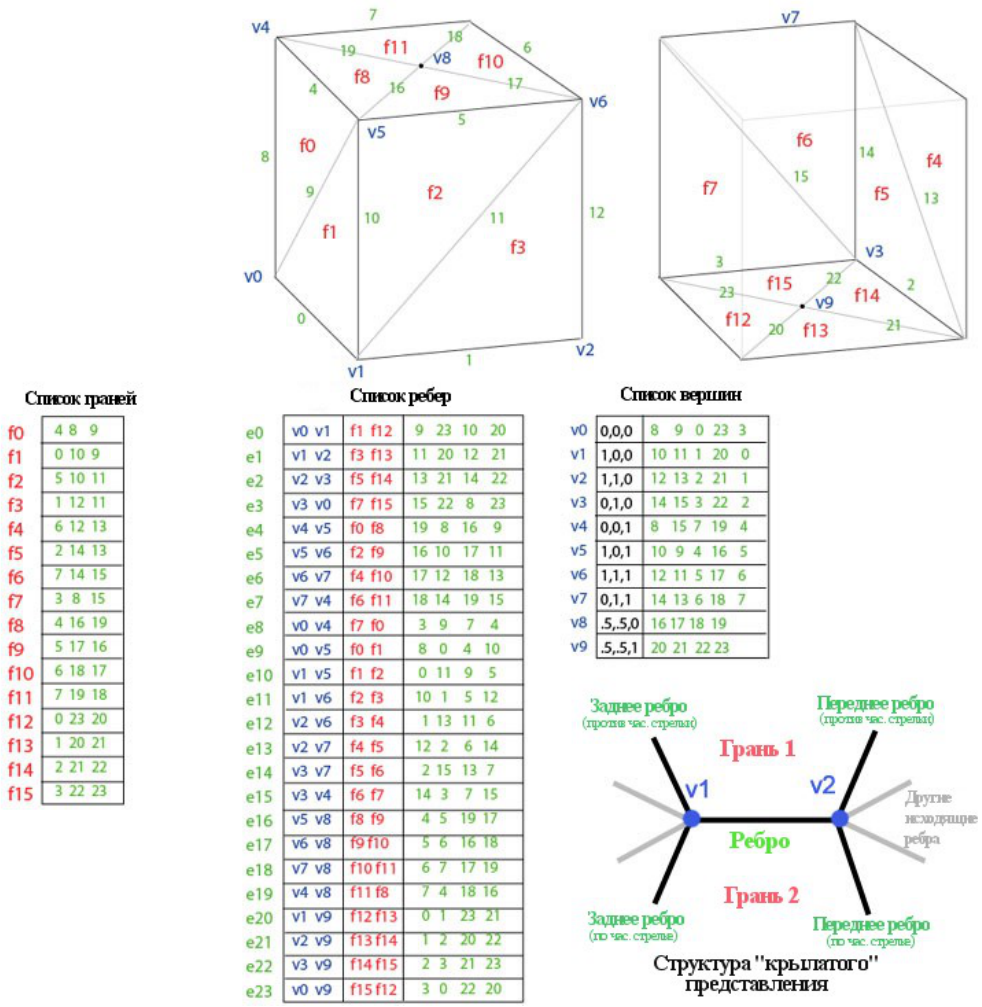

Рисунок 1.7 — «Крылатое» представление

1.4 Анализ алгоритмов удаления невидимых линий и поверхностей

Главной целью при создании реалистичного изображения является удаление объектов или их частей, которые не видны наблюдателю из-за перекрытия другими объектами. Для решения этой задачи выделяют две категории алгоритмов:

– *Алгоритмы, работающие в объектном пространстве,* привязаны к мировой или физической системе координат. Их результаты зависят только от точности вычислений и требуют значительных вычислительных ресурсов, что зависит от необходимой точности и сложности входной сцены. К этой категории относятся алгоритмы Робертса, алгоритмы со списком приоритетов и другие.

– *Алгоритмы, работающие в пространстве изображения,* ориентиро-

ваны на систему координат экрана или картинной плоскости, на которую проецируются объекты. Объем вычислений в этой группе значительно меньше по сравнению с первой, и он зависит от разрешающей способности экрана и количества объектов в сцене. К основным алгоритмам этой группы относятся алгоритм Варнока, алгоритм Z-буфера и алгоритм трассировки лучей.

1.4.1 Алгоритм Робертса

Алгоритм Робертса — это первое известное решение задачи об удалении невидимых линий в трехмерной графике. Он представляет собой математически элегантный метод, работающий в объектном пространстве. Основная задача алгоритма заключается в удалении ребер и граней, которые экранируются самим объектом, а затем в сравнении оставшихся видимых ребер с другими объектами для определения их видимости.

Работа Алгоритма Робертса проходит в два этапа:

– определение нелицевых граней для каждого тела отдельно;

– определение и удаление невидимых ребер.

Для корректной работы данного алгоритма, необходимо выполнение следующих предварительных условий:

– тело ограничено плоскостями;

– нормали всех граней направлены внутрь тела;

– тело выпукло (невыпуклые тела должны быть разбиты на выпуклые части).

В этом алгоритме выпуклое многогранное тело с плоскими гранями должно представиться набором пересекающихся плоскостей. Уравнение произвольной плоскости в трехмерном пространстве имеет вид:

$$ax + by + cz + d = 0 \tag{1.1}$$

В матричной форме уравнение 1.1 выглядит так:

$$\begin{pmatrix} x & y & z & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} = 0 \tag{1.2}$$

Тогда набор пересекающихся плоскостей состоит из матрицы коэффициентов уравнений плоскостей и называется *матрицей тела*:

$$V = \begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ b_1 & b_2 & \cdots & b_n \\ c_1 & c_2 & \cdots & c_n \\ d_1 & d_2 & \cdots & d_n \end{pmatrix} \tag{1.3}$$

Корректное формирование матрицы тела предполагает, что любая точка внутри него должна находиться по положительную сторону от каждой грани. Если это условие не соблюдено для определенной грани, следует умножить соответствующий столбец матрицы на -1. Для проверки правильности можно взять точку, находящуюся внутри тела, координаты которой вычисляются как среднее значение координат всех его вершин.

Далее удаляются грани, которые скрыты телом. Для этого используется вектор взгляда $E = \begin{pmatrix} 0 & 0 & -1 & 0 \end{pmatrix}$, и чтобы определить невидимые грани, вектор умножается на матрицу тела V. Отрицательные компоненты результата указывают на невидимые грани.

Для выявления невидимых точек на ребре необходимо провести луч от наблюдателя к точке наблюдения. Если луч сталкивается с каким-либо объектом, то точка считается невидимой. Если объект действительно препятствует прохождению луча, он должен пересекать этот объект, оставаясь по положительную сторону от каждой его грани.

Вычислительная сложность алгоритма возрастает теоретически как квадрат числа объектов, то есть его главный недостаток – скорость работы. К тому же, этот алгоритм сложнореализуемый из-за его полностью математической структуры.

Основное преимущество данного алгоритма заключается в высокой точности вычислений, которая достигается благодаря работе в объектном пространстве, в отличие от большинства других алгоритмов.

1.4.2 Алгоритм Варнока

Алгоритм Варнока основывается на гипотезе о том, как человеческий глаз и мозг обрабатывают информацию в сцене. Эта гипотеза утверждает, что на обработку областей с низким информационным содержанием уходит значительно меньше времени и усилий, чем на области с высоким содержанием информации.

Конкретная реализация алгоритма зависит от метода разбиения окна и от критериев, используемых для определения простоты содержимого окна. В оригинальной версии алгоритма каждое окно делится на четыре равных подокна. Многоугольник в изображаемой сцене может быть классифицирован следующим образом (рис. 1.8):

- *внешний*, если он находится полностью вне окна;
- *внутренний*, если он находится полностью внутри окна;
- *пересекающий*, если он пересекает границу окна;
- *охватывающий*, если окно находится полностью внутри него.

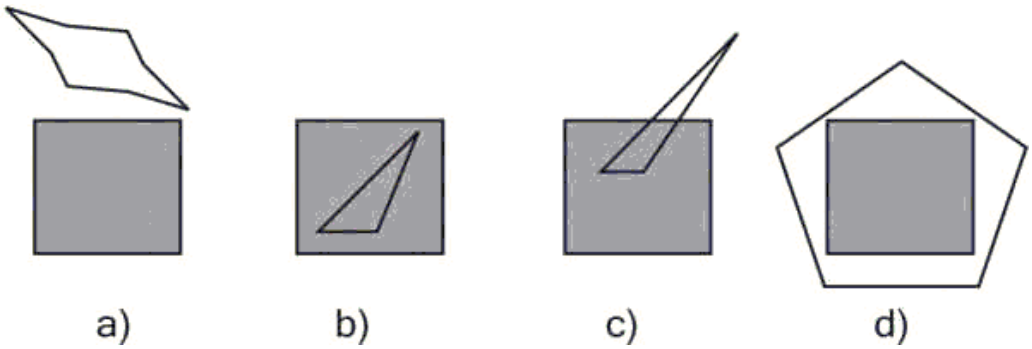

Рисунок 1.8 — Типы многоугольников: внешний (a), внутренний (b), пересекающий (c), охватывающий (d)

Алгоритм можно описать в общих чертах следующим образом:

1) если все многоугольники сцены являются внешними по отношению к окну, то окно считается пустым, заполняется фоновым цветом и дальнейшему разбиению не подлежит.
2) если только один многоугольник пересекает окно и является внутренним, окно заполняется фоновым цветом, а сам многоугольник — своим

цветом.

3) если только один многоугольник пересекает окно и является пересекающим, окно заполняется фоновым цветом, а часть многоугольника, которая принадлежит окну, заполняется цветом этого многоугольника.

4) если только один многоугольник охватывает окно и нет других многоугольников, имеющих общие точки с окном, то окно заполняется цветом этого многоугольника.

5) если существует хотя бы один охватывающий окно многоугольник, из всех таких выбирается тот, который ближе всего к точке наблюдения, и окно заполняется его цветом.

6) в противном случае производится новое разбиение окна.

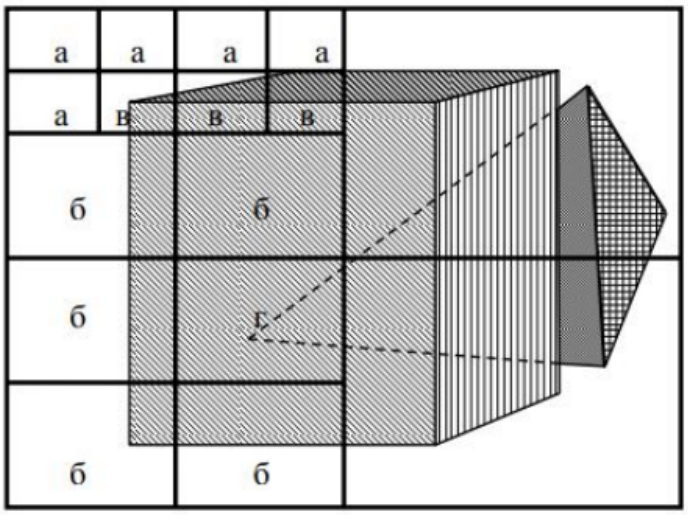

Рисунок 1.9 — Пример разбиения алгоритмом Варнока

Шаги 1–4 рассматривают случай пересечения окна только с одним многоугольником, что помогает уменьшить количество подразбиений. Шаг 5 решает проблему удаления невидимых поверхностей, так как многоугольник, находящийся ближе к наблюдателю, экранирует остальные. Пример разбиения приведён на рисунке 1.9.

Основной недостаток алгоритма заключается в его рекурсивной работе. На каждом этапе он проверяет, какие грани видны, и если определение видимости нетривиально, окно разделяется на четыре секции, и анализ проводится отдельно для каждой из них.

Вывод

<что сделали в аналите, кратко>

— test

2 Конструкторская часть

<пишем про алгоритмы, архитектуру ПО и тп>

Вывод

<что сделали в конструкторке и получили в результате, кратко>

3 Технологическая часть

<писать все про реализацию: какие языки, какие ide и тд.; коды алгоритмов/программы>

<тестовые данные и волшебные слова «все тесты пройдены успешно» >

Вывод

<что делали и получили в результате>

4 Исследовательская часть

<цель исследования, на какой машине делали (указать ЦПУ, ОЗУ, ОС), желательно написать, как исследовали и при каких условиях; что получили в результате (таблицы + графики)>

4.1 Вывод

<что сделали и получили в результате, кратко>

ЗАКЛЮЧЕНИЕ

<копируем цель и кратко описываем, что делали и получили в результате; указать, какие задачи выполнили>

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Литература 1.
2. Литература 2.
3. ...
4. Литература N.

Приложение А

<тут всякие слишком большие ништяки, которые вы хотели бы добавить в отчет>