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-

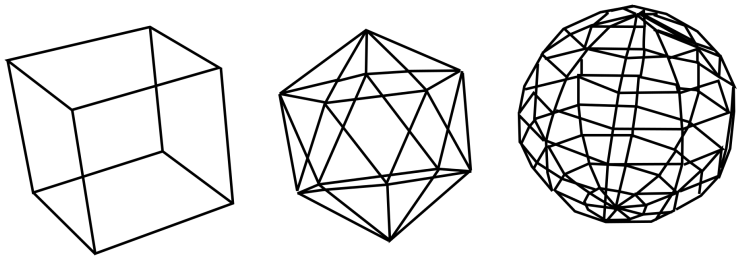

Рисунок 1.1 — Каркасные модели

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

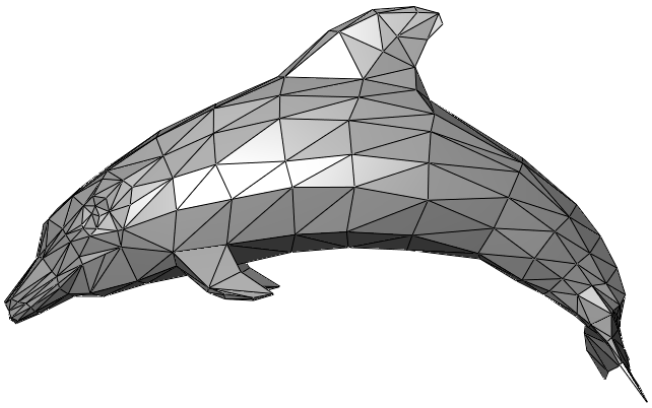

Рисунок 1.2 — Поверхностая модель, заданная полигональной сеткой

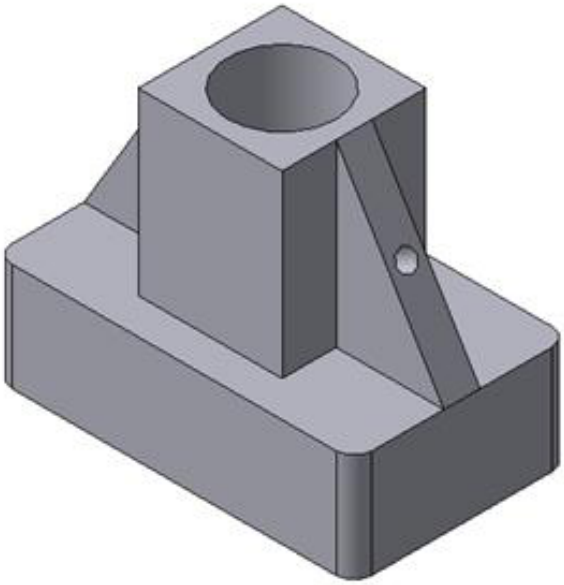

Рисунок 1.3 — Твердотельная модель

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 парамет-

рического 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.

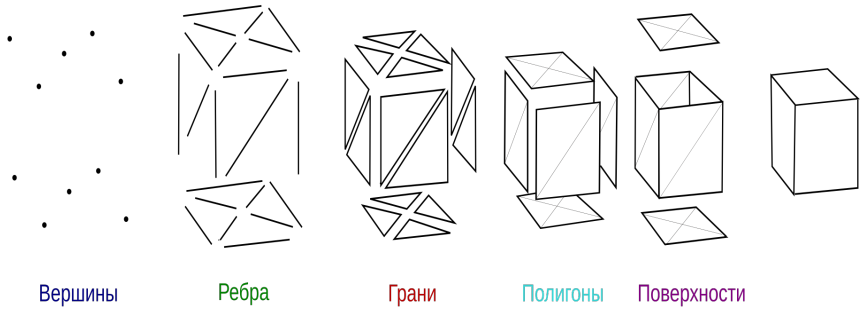

Рисунок 1.4 — Элементы моделирования сетки

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 рендеринга, 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.
3) *«крылатое» представление* – способ представления, описывающий объект как множество вершин, граней и рёбер, что обеспечивает высокую

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

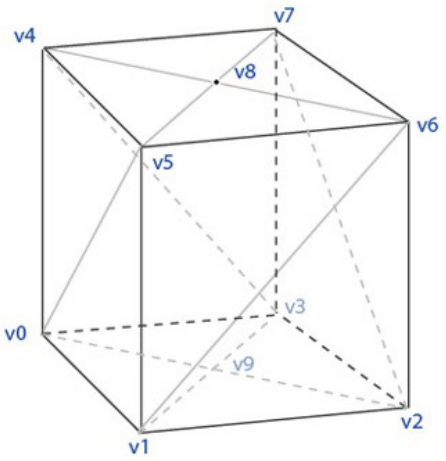

Рисунок 1.5 — Вершинное представление

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.

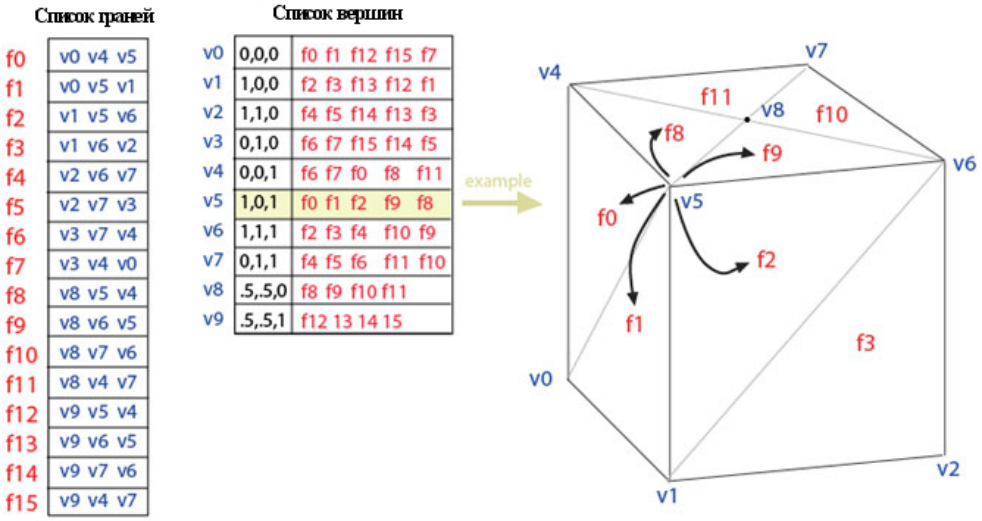

Рисунок 1.6 — Список граней

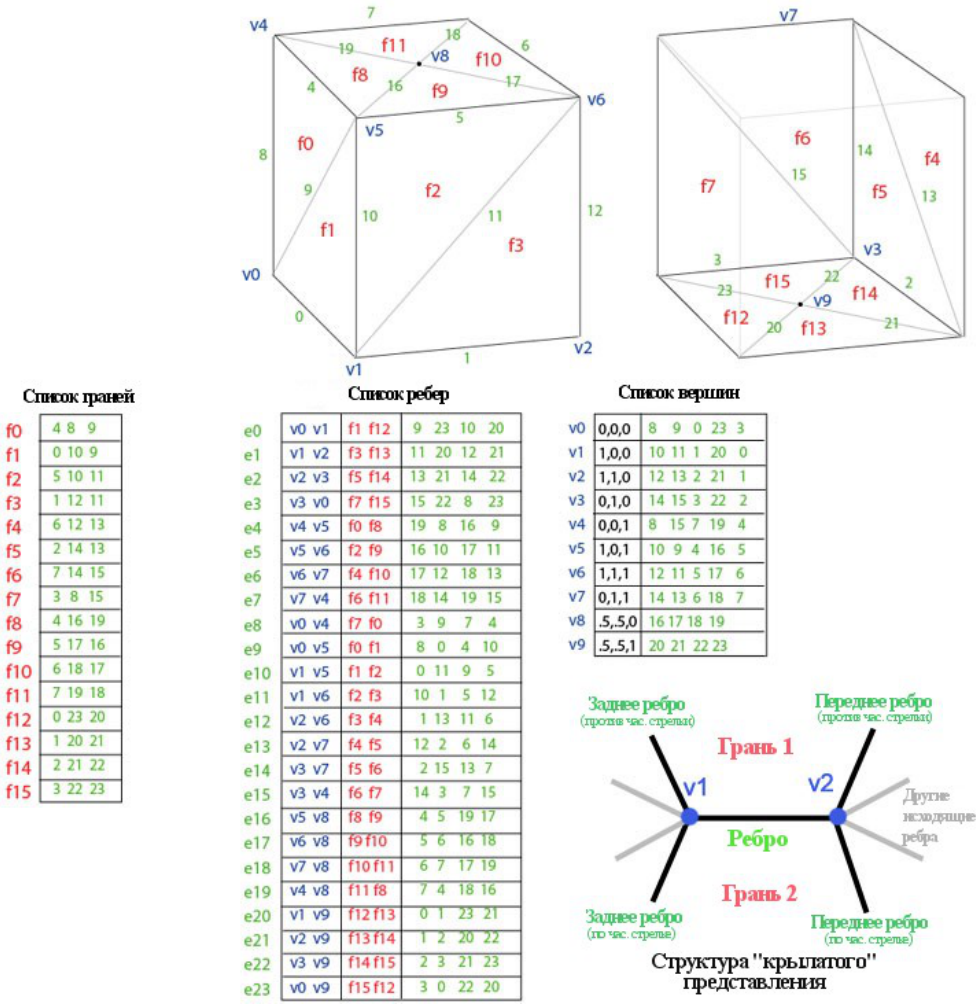

Рисунок 1.7 — «Крылатое» представление

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,* ориентированы 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 $E = (0\ 0\ -1\ 0)$, 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.

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

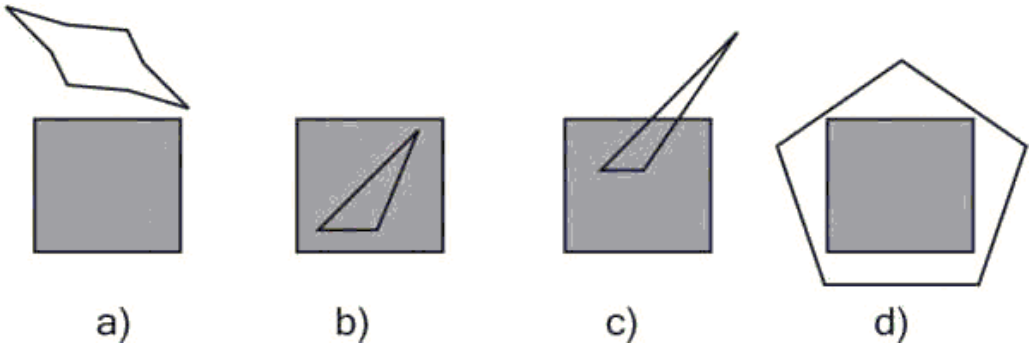

Рисунок 1.8 — Типы многоугольников в алгоритме Варнока: внешний (a), внутренний (b), пересекающий (c), охватывающий (d)

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.

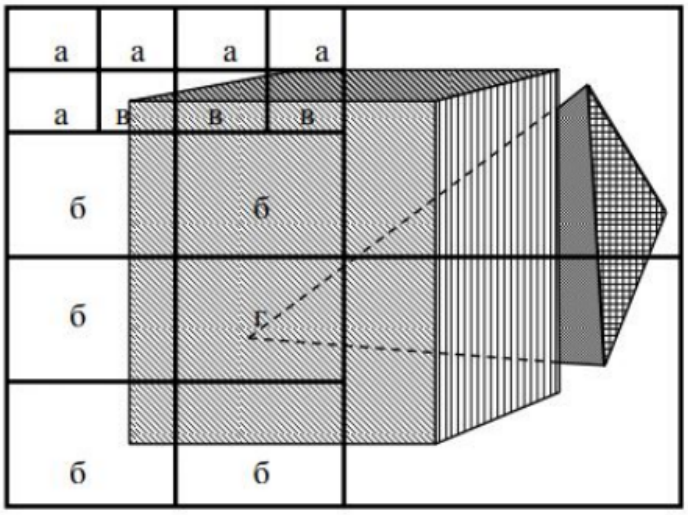

Рисунок 1.9 — Пример разбиения алгоритмом Варнока

Шаги 1–4 рассматривают случай пересечения окна только с одним много-

угольником, 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.

1.4.3 Алгоритм трассировки лучей

Главная концепция алгоритма трассировки лучей состоит в том, что наблюдатель замечает объекты благодаря свету, исходящему от источников, который взаимодействует с объектами и, следуя законам оптики, достигает его глаза. В методе прямой трассировки создаются траектории лучей от всех источников света ко всем точкам объектов, что приводит к большому количеству обрабатываемых лучей и значительным вычислительным затратам, поскольку лишь небольшая часть лучей достигает точки наблюдения.

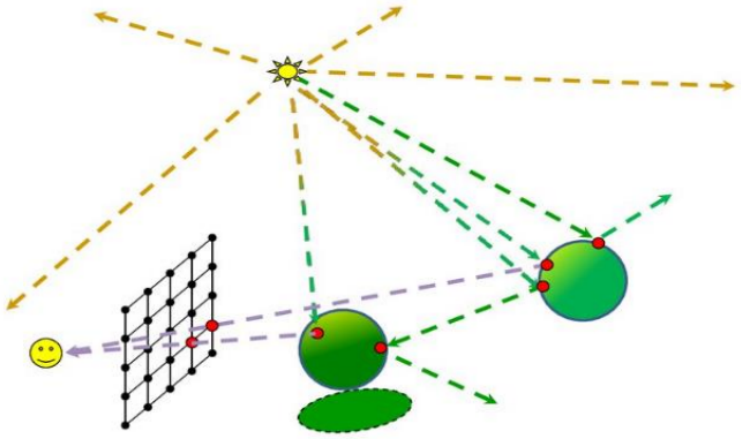

Рисунок 1.10 — Алгоритм трассировки лучей

Этот алгоритм подходит для создания статических сцен и моделирования различных оптических эффектов, таких как отражение и преломление.

1.4.4 Алгоритм обратной трассировки лучей

Идея алгоритма обратной трассировки лучей состоит в том, что наблюдатель видит объекты благодаря свету, исходящему от источников, который отражается от объектов и достигает его глаз. Однако отслеживание путей лучей от источника к наблюдателю неэффективно с вычислительной точки зрения, поэтому более целесообразно отслеживать их в обратном направлении — от наблюдателя к объектам. Лучи исходят из камеры, проходя через каждый пиксель сцены, после чего определяется их пересечение с объектами. Если пересечение найдено, рассчитывается интенсивность пикселя с учетом расположения источников света. Наблюдатель считается находящимся на положительной полуоси z в бесконечности, поэтому все лучи параллельны этой оси.

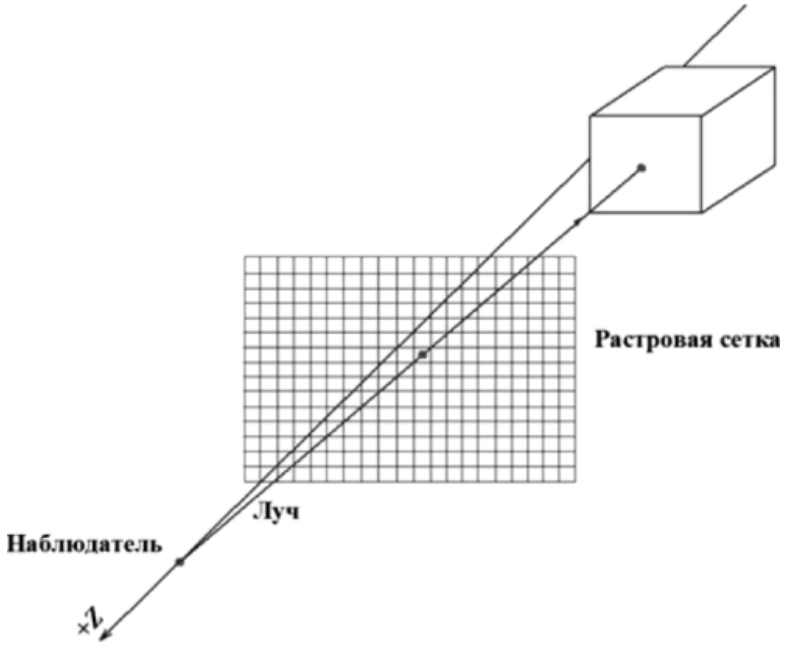

Рисунок 1.11 — Алгоритм обратной трассировки лучей

При этом, несмотря на большую эффективность алгоритма по сравнению с прямой трассировкой, он все же медленный из-за необходимости точно рассчитывать сложные аналитические выражения для нахождения пересечений с объектами.

1.4.5 Алгоритм художника

Алгоритм художника предполагает сортировку всех полигонов сцены по удаленности от наблюдателя. Объекты, находящиеся дальше от камеры, ренде-

рятся первыми, а затем последовательно добавляются более близкие объекты.

Например, в сцене с горами, лугом и деревьями, изображенными на рисунке 1.12, правильный порядок рендеринга будет: горы → луг → деревья.

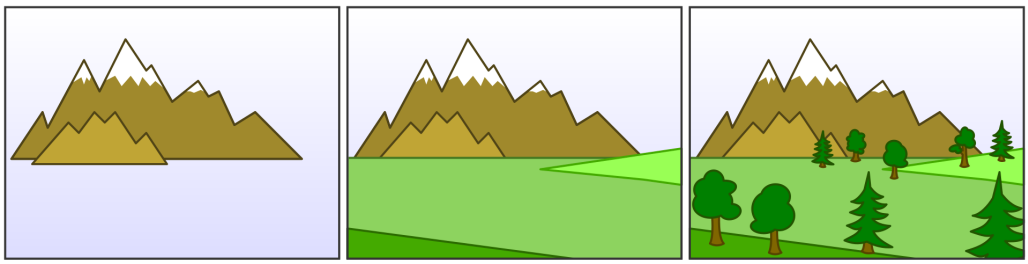

Рисунок 1.12 — Последовательная работа алгоритма художника

Несмотря на свою простоту, алгоритм художника имеет несколько значительных недостатков:

– если полигоны пересекаются, определить правильный порядок их рисования становится невозможно. Например, если три полигона накладываются друг на друга (рис. 1.13), то независимо от порядка сортировки, результат будет некорректным. Для решения этой проблемы может потребоваться разбивка конфликтующих полигонов на меньшие части, что усложняет реализацию;

– алгоритм также может прорисовывать области, которые впоследствии будут перекрыты другими объектами. Это приводит к ненужным затратам процессорного времени и ресурсов, так как рендерятся объекты, которые не будут видны в конечном результате.

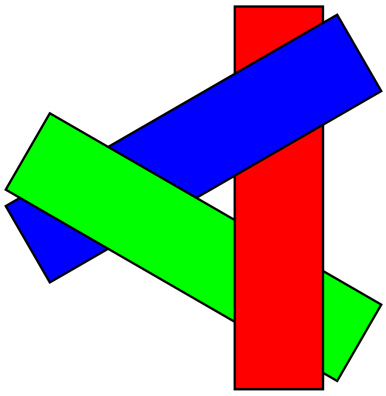

Рисунок 1.13 — Взаимноперекрывающиеся полигоны

Из-за этих недостатков был разработан алгоритм Z-буфера, который представляет собой более эффективный способ обработки видимости.

1.4.6 Алгоритм Z-буфера

Алгоритм z-буфера представляет собой один из наиболее простых способов удаления невидимых объектов, работающий в пространстве изображения. Он основан на концепции буфера кадра, который сохраняет интенсивность каждого пикселя, и дополнительно использует z-буфер для хранения координат глубины видимых пикселей. Во время работы алгоритм сравнивает глубину нового пикселя с уже сохраненной в z-буфере. Если новый пиксель находится ближе к наблюдателю, он добавляется в буфер кадра, а значение глубины обновляется. Если же он дальше, то ничего не происходит. Таким образом, алгоритм фактически ищет максимальное значение функции $z(x, y)$.

Формально алгоритм можно описать следующим образом:

1) инициализировать буфер кадра фоновым значением цвета или интенсивности.
2) заполнить z-буфер минимальными значениями глубины.
3) преобразовать многоугольники в растровую форму в произвольном порядке.
4) для каждого пикселя (x, y) в многоугольнике вычислить его глубину $z(x, y)$.
5) если $z(x, y) > Z_{буфер}(x, y)$ (ближе к наблюдателю), обновить буфер кадра и z-буфер, иначе никаких действий не предпринимать.

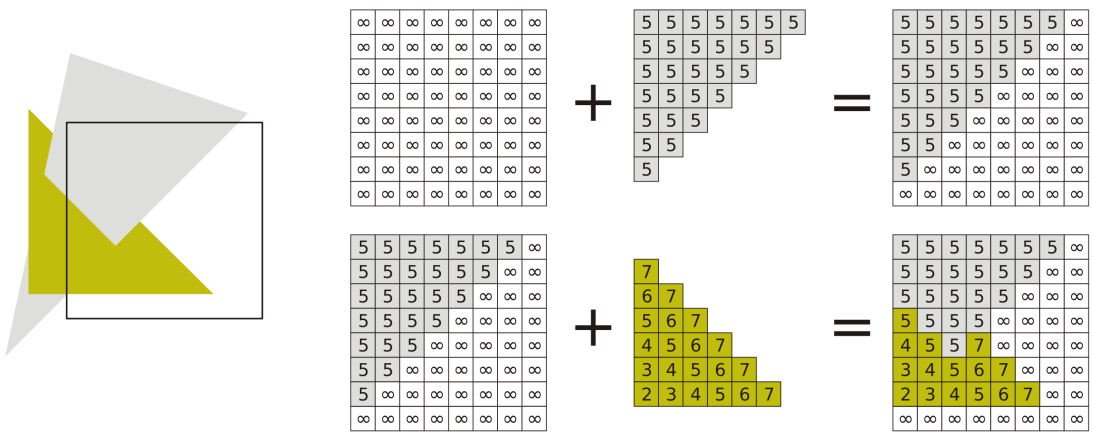

Рисунок 1.14 — Пример заполнения z-буфера

Главное преимущество этого метода — его простота. Он эффективно решает задачу удаления невидимых объектов и позволяет легко визуализиро-

вать пересечения сложных форм. Алгоритм обладает линейной вычислительной сложностью, так как порядок добавления элементов не имеет значения, что исключает необходимость предварительной сортировки по глубине.

Основной недостаток алгоритма – большой объём требуемой памяти. Однако, поскольку размер создаваемого изображения относительно небольшой, затраты памяти на хранение информации о каждом пикселе в этом алгоритме незначительны для современных компьютеров.

1.4.7 Выбор оптимального алгоритма

Для удаления невидимых линий и поверхностей был выбран алгоритм Z-буфера. Данный алгоритм прост в своей реализации, производителен и позволяет добиться достаточной детализации синтезируемого изображения.

1.5 Анализ модели освещения

Модели освещения являются основным инструментом для создания реалистичных изображений. Они описывают, как свет взаимодействует с поверхностями объектов и как это взаимодействие влияет на восприятие цвета и яркости. Модели освещения можно разделить на две основные категории:

– *Локальные* модели освещения рассматривают освещение на уровне отдельной поверхности, не учитывая влияние света от других объектов. Эти модели обычно используют упрощенные подходы для расчета освещения и являются более производительными, что делает их подходящими для реального времени;

– *Глобальные* модели освещения учитывают взаимодействие света между различными объектами в сцене, что позволяет создавать более реалистичные изображения. Эти модели обычно требуют больше вычислительных ресурсов.

1.5.1 Модель Ламберта

Модель Ламберта, также известная как модель диффузного отражения, основана на предположении, что интенсивность света, отраженного от поверхности, пропорциональна косинусу угла между нормалью к поверхности и направлением на источник света. Это означает, что поверхность будет выглядеть

ярче, когда она освещается под прямым углом и темнее, когда свет падает под острым углом.

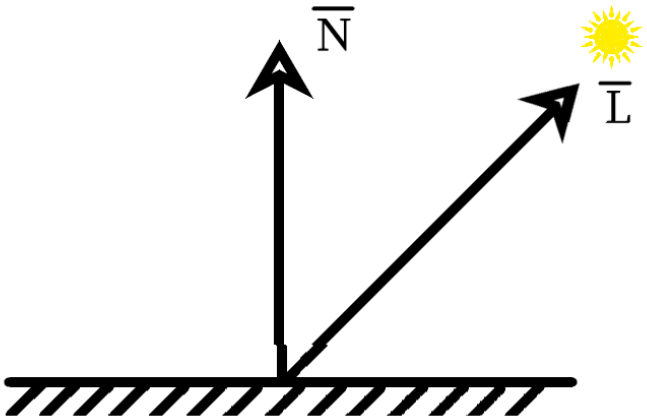

Рисунок 1.15 — Модель Ламберта

При этом положение наблюдателя не влияет на восприятие, поскольку свет, отраженный диффузно, распределяется равномерно во всех направлениях.

Формула для расчета диффузного освещения в модели Ламберта выглядит следующим образом:

$$I = I_0 \cdot k_d \cdot cos(\vec{L}, \vec{N}) = I_0 \cdot k_d \cdot (\vec{L} \cdot \vec{N}) \tag{1.4}$$

где I – результирующая интенсивность света в точке, I_0 – интенсивность источника света, k_d – коэффициент диффузного освещения, $\vec{L}$ – вектор от точки до источника и $\vec{N}$ – нормаль в точке.

1.5.2 Модель Фонга

Модель Фонга является классической моделью освещения, которая сочетает в себе диффузную составляющую и зеркальную составляющую. Это позволяет не только обеспечить равномерное освещение, но и создавать блики на материале. Положение блика на объекте, освещенном по модели Фонга, определяется законом равенства углов падения и отражения. Если наблюдатель находится близко к углу отражения, яркость данной точки увеличивается.

Падающий и отраженный лучи расположены в одной плоскости с нормалью к поверхности в точке падения, и эта нормаль делит угол между лучами пополам. Таким образом, отраженная составляющая освещенности в данной

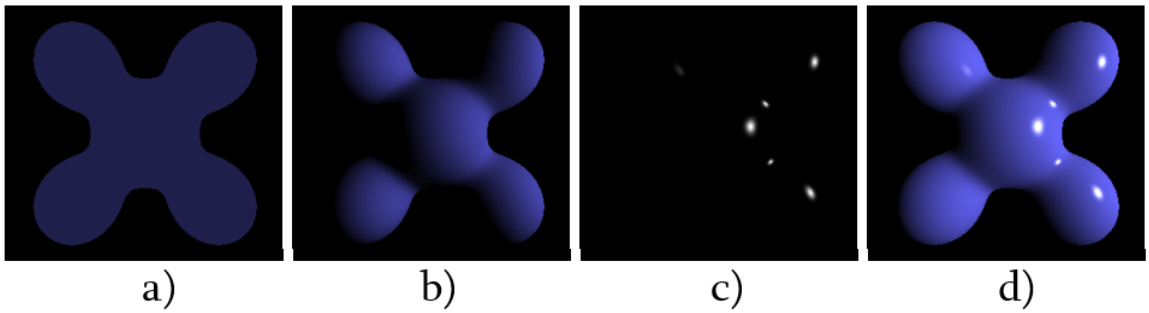

Рисунок 1.16 — Компоненты модели Фонга: рассеяная (a), диффузная (b), зеркальная (c), суммарная (d)

точке зависит от того, насколько близки направления на наблюдателя и отраженный луч.

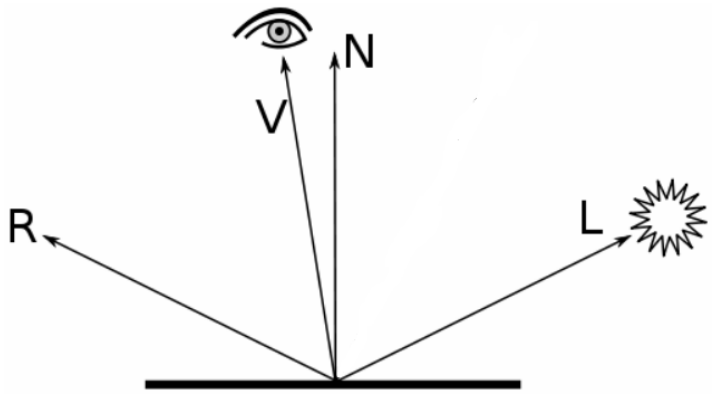

Рисунок 1.17 — Модель Фонга

Формула для расчета освещения в модели Фонга выглядит следующим образом:

$$I = I_a + I_d + I_s \tag{1.5}$$

где I_a – рассеяная составляющая, I_d – диффузная составляющая и I_s – зеркальная составляющая.

$$I_a = I_a \cdot k_a \tag{1.6}$$

где I_a – интенсивность рассеянного освещения и k_a – коэффициент рассеянного освещения.

$$I_d = I_0 \cdot k_d \cdot cos(\vec{L}, \vec{N}) = I_0 \cdot k_d \cdot (\vec{L} \cdot \vec{N}) \tag{1.7}$$

где I_0 – интенсивность источника света, k_d – коэффициент диффузного освещения, $\vec{L}$ – вектор от точки до источника и $\vec{N}$ – нормаль в точке.

$$I_s = I_0 \cdot k_s \cdot cos^{\alpha}(\vec{R}, \vec{V}) = I_0 \cdot k_s \cdot (\vec{R} \cdot \vec{V})^{\alpha} \tag{1.8}$$

где k_s – коэффициент зеркального освещения, $\vec{R}$ – вектор отраженного луча и $\vec{V}$ – вектор от точки до наблюдателя.

$$\vec{R} = 2 \cdot (\vec{N} \cdot \vec{L}) \cdot \vec{N} - \vec{L} \tag{1.9}$$

где $\vec{R}$ – вектор отраженного луча, $\vec{N}$ – нормаль в точке и $\vec{L}$ – вектор от точки до источника.

1.5.3 Модель Блинна-Фонга

Модель Блинна-Фонга представляет собой упрощение модели Фонга, используемое для расчета зеркального отражения света на поверхностях с меньшими вычислительными затратами. Основная идея заключается в использовании вектора полупути $\vec{H}$, который определяется как:

$$\vec{H} = \frac{\vec{L} + \vec{V}}{|\vec{L} + \vec{V}|} \tag{1.10}$$

где $\vec{L}$ – вектор от точки до источника и $\vec{V}$ – вектор от точки до наблюдателя.

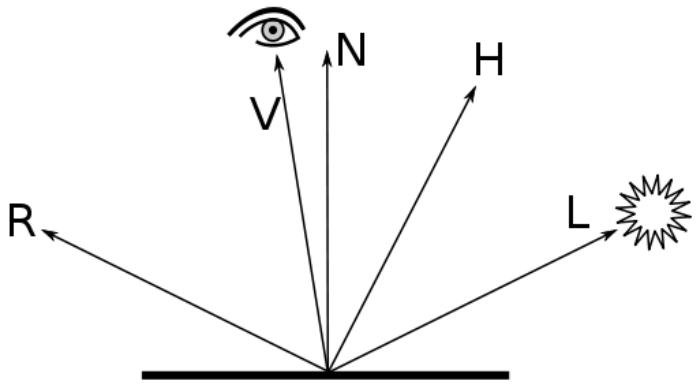

Рисунок 1.18 — Модель Блинна-Фонга

Вектор $\vec{H}$ называется вектором полупути, поскольку, если все три вектора $\vec{V}$, $\vec{L}$ и $\vec{N}$ лежат в одной плоскости, то угол между $\vec{H}$ и $\vec{N}$ составляет половину угла между $\vec{R}$ и $\vec{V}$. Величину $(\vec{R} \cdot \vec{V})^{\alpha}$ необходимую при расчёте зеркальной составляющей интенсивности I_s в модели Фонга (1.8) можно заменить на величину

$(\vec{H} \cdot \vec{N})^{\beta}$.

Вектор $\vec{H}$ показывает ориентацию поверхности, на которой будет максимальное отражение света. В ряде случаев модель Блинна-Фонга требует значительно меньше вычислений, например в случае направленного бесконечно-удаленного источника.

1.5.4 Выбор оптимальной модели освещения

В качестве модели освещения была выбрана модель Фонга. Данная модель учитывает как диффузное, так и зеркальное отражение света, что делает синтезируемое изображение более реалистичным и способствует возникновению бликов света.

1.6 Анализ алгоритмов закрашивания

Алгоритмы закрашивания определяют, как поверхности объектов будут окрашены в зависимости от их геометрии, освещения и материалов.

1.6.1 Простая закраска

Данный алгоритм представляет простейший способ закраски. Для многогранных объектов высчитываются уровни интенсивности для каждой грани согласно закону Ламберта (1.4) и закраска каждой грани соответствует её уровню интенсивности. Это делает процесс рендеринга простым и быстрым, поскольку не требует сложных расчетов для каждой точки на поверхности.

Рисунок 1.19 — Аппроксимированная сфера, закрашенная алгоритом простой закраски

Однако, простая закраска может привести к тому, что границы между соседними гранями будут слишком резкими и заметными. Это можно исправить увеличением количества граней объекта.

1.6.2 Закраска Гуро

В отличие от простой закраски, где освещение рассчитывается для всей грани, в закраске Гуро нормали определяются в вершинах модели. Это позволяет учитывать освещение на более детальном уровне и обеспечивает более плавные переходы между цветами.

Нормали к вершинам модели вычисляются путем усреднения нормалей всех граней, которые соединяются в данной вершине. Это позволяет учитывать влияние нескольких граней на освещение в этой точке. После того как нормали вершин определены, можно вычислить освещенность каждой вершины. Закраска осуществляется вдоль рёбер многоугольников с использованием билинейной интерполяции значений интенсивности цвета в вершинах. Это позволяет создать плавный переход между цветами, что визуально сглаживает поверхность.

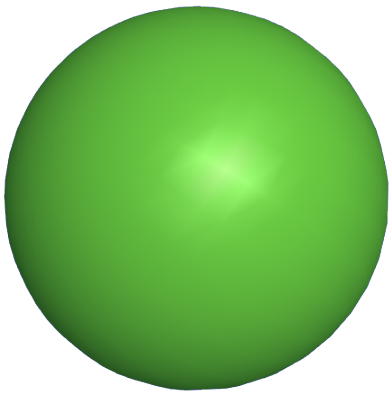

Рисунок 1.20 — Аппроксимированная сфера, закрашенная алгоритом Гуро

Одним из основных недостатков затенения по Гуро является то, что при низком уровне детализации блики на поверхности могут быть смазаны или потеряны.

1.6.3 Закраска Фонга

В отличие от закраски Гуро, где цвета интерполируются между вершинами, в закраске Фонга интерполируются нормали. Цвет, в свою очередь, рас-

считывается для каждого пикселя в отдельности. При использовании закраски Фонга изображение получается гораздо более качественным, чем при использовании предыдущих техник, и исчезает проблема с бликами.

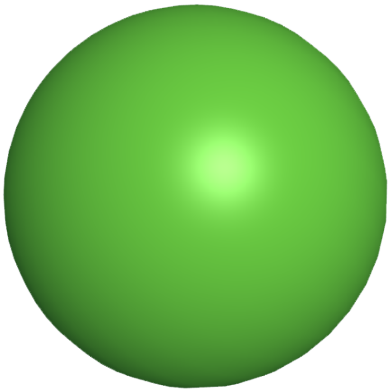

Рисунок 1.21 — Аппроксимированная сфера, закрашенная алгоритом Фонга

Одним из основных недостатков является необходимость выполнять большое количество вычислений.

1.6.4 Выбор оптимального алгоритма закраски

Для закраски был выбран алгоритм Фонга. Данный алгоритм несмотря на его вычислительные затраты позволит обеспечить реалистичное освещение и четкие блики.

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>