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

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

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

**ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ**

Студент Турчанинов Александр Максимович\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*фамилия, имя, отчество*

Группа \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ИУ7-44Б\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** \_\_Турчанинов А.М.\_\_\_

*подпись, дата фамилия, и.о.*

Руководитель практики **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_** Куров А. В.**\_\_\_\_**

*подпись, дата фамилия, и.о.*

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

*Москва 2022 г.*

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

Кафедра «Программное обеспечение ЭВМ и информационные технологии» (ИУ7)

**ЗАДАНИЕ**

**на прохождение производственной практики**

на предприятии МГТУ им. Н.Э. Баумана (кафедра ИУ7)

Студент\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Турчанинов Александр Максимович ИУ7-44Б\_\_\_\_\_

(фамилия, имя, отчество; инициалы; индекс группы)

Во время прохождения производственной практики студент должен:

1. Начать разработку программного обеспечения для визуализации удара молнии.

2. Решить вопрос о способе представления объектов, проанализировать и выбрать алгоритмы для их обработки.

3. Закрепить знания и навыки, полученные в ходе аудиторных занятий по пройденным курсам.

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

Руководитель практики от кафедры\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись, дата)

**Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ / \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись, дата) (Фамилия И.О.)

# Введение

В наше время компьютерная графика имеет достаточно широкий охват применения во всех отраслях нашей жизни: в кино, в компьютерных играх, в рекламе и дизайне, в обучающих программах.

Перед специалистами, создающими трехмерные изображения появляется множество сложностей таких как: преломление, отражение и рассеивание света.

Знания компьютерной графики никогда не станут лишним багажом, они только станут преимуществом на собеседованиях с работодателями, помогут самостоятельно создавать персонажей для компьютерных игр, генерировать спецэффекты для фильмов.

Необходимость широкого использования графических программных средств стала особенно ощутимой в связи с развитием Интернета. Способность компьютерной графики быть многозадачной, необычной и символичной. Применение графики в учебных системах не только позволяет увеличить скорость передачи информации и повысить уровень её понимания, но и способствует развитию образного мышления.

Мною была выбрана тема, связанная с моделированием броска баскетбольного мяча.

Для исполнителя важно исключить возникновение правок на поздних стадиях выполнения работы. Поэтому важно представить реалистичное изображение на этапе предварительного показа. Для построения такого изображения, которое позволит понять пользователю концепцию работы, потребуется учитывать невидимость ребер объектов сцены по отношению к наблюдателю и освещение отдельных участков рабочей плоскости.

**Целью** курсовой работы является реализация программного обеспечения для моделирования броска баскетбольного мяча.

Для достижения поставленной цели необходимо выполнить следующие задачи.

* Описать список доступных к размещению на сцене моделей, формализовать эти модели.
* Выбрать существующие алгоритмы компьютерной графики для визуализации сцены и объектов на ней.
* Выбрать язык программирования и среду разработки.
* Реализовать выбранные алгоритмы визуализации.
* Реализовать программное обеспечение для визуализации и редактирования сцены.

# 1 Аналитическая часть

**1.1 Описание и формализация объектов сцены**

Сцена состоит из следующих объектов.

* **Область действий** – куб, внутри которого располагаются модели и зритель. Размер куба постоянный, зритель не перемещается по области и смотрит в сторону объектов сцены. Цвет области – белый.
* **Объекты сцены** – модели, расположенные внутри области действий. Каждая модель представляет собой набор граней, описываемых точками в пространстве, которые соединены ребрами. Все доступные модели определены заранее, в программе не предусмотрена возможность добавления новых или изменения старых моделей. В программном обеспечении доступна возможность задавать силу и направление броска мяча.

Список доступных объектов сцены.

1. Б**аскетбольное кольцо**. Состоит из щита и кольца. **Щит** представляет собой плоскость, параллельную задней стенке области действий. Размеры щита фиксированы. **Кольцо** представляет собой окружность, прикрепленную к щиту, фиксированного размера.
2. **Мяч.** Объект шаровидной формы оранжевого цвета. Находится в нижней части поля зрения наблюдателя. При нажатии на кнопку броска летит с заданной пользователем силой и направлением в кольцо и проходит через кольцо, если сила и направление подобранны пользователем правильно, иначе отскакивает от кольца или щита и возвращается на исходную позицию для совершения следующего броска.
3. **Наблюдатель.** Следит за полетом мяча в кольцо. Количество бросков ограничено только временем работы программы. Зафиксирован в одном положении.

* **Источник света** – материальная точка пространства. Принимает ортогональную проекцию визуализируемой сцены из своего положения с некоторым ограниченным обзором. В зависимости от расположения источника и направления распространения лучей света, определяется тень от объектов, расположенных в области действий. Положение источника света зафиксировано.

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

Отображением формы и размеров объектов являются модели.

Обычно используются три формы задания моделей:

1. **Каркасная (проволочная) модель**

Одна из простейших форм задания модели, так как мы храним только информацию о вершинах и ребрах нашего объекта. Недостаток – модель не всегда точно передает представление объекта.

1. **Поверхностная модель**

Тип модели, часто используемый в компьютерной графике. Поверхности задаются разными способами – аналитически или задаются участки поверхности, как поверхности разных видов(использование полигональной аппроксимации). Недостаток – мы не знаем с какой стороны находится материал.

1. **Объемная (твердотельная) модель**

Главное отличие от поверхностной модели – информация о том, где расположен материал. Достигается это посредством указания направления внутренней нормали.

**Выбор модели.**

Таким образом, можно сделать вывод о том, что для решения данной задачи нам подойдут поверхностные модели, так как такое представление не будет приводить к неправильному восприятию формы, как при использовании каркасных моделей, и не будет больших затрат по памяти, как при использовании твердотельной модели.

**1.3 Анализ способа задания поверхностных моделей**

Также необходимо определить каким образом лучше всего задавать поверхностные модели:

* **Аналитическим способом.** Этот способ задания модели характеризуется описанием модели объекта, которое доступно в неявной форме, то есть для получения визуальных характеристик необходимо дополнительно вычислять некоторую функцию, которая зависит от параметра;
* **Полигональной сеткой.** Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной графике.

Для верного выбора стоит перечислить способы хранения информации о сетке:

* Список граней. Объект – это множество граней и множество вершин. В каждую грань входит как минимум 3 вершины.
* "Крылатое" представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые её касаются.
* Полурёберные сетки. Похоже на "Крылатое" представление, но информация обхода хранится для половины грани.
* Таблица углов. Таблица, в которой хранятся вершины. Обход заданной таблицы неявно задает полигоны. Такое представление более компактно и более производительно для нахождения полигонов, но, в связи с тем, что вершины присутствуют в описании нескольких углов, операции по их изменению медленны.
* Вершинное представление. Хранятся только вершины, указывающие на другие вершины. Простота представления дает возможность проводить над сеткой множество операций.

Решающий фактор при выборе задания модели в проекте – скорость выполнения преобразований над объектами сцены.

Наиболее удобный способ представления для реализации данного программного обеспечения – модель, заданная полигональной сеткой, так как это поможет избежать проблем при описании сложных моделей. Способ хранения полигональной сетки – список граней, так как он представляет явное описание граней, что поможет при реализации алгоритма удаления невидимых ребер и поверхностей. Еще одно преимущество этого способа – эффективное преобразование модели, так как структура включает в себя список вершин.

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

Чтобы выбрать правильный алгоритм необходимо определить некоторые свойства, которыми должен обладать выбранный алгоритм для обеспечения реалистичного изображения и оптимальную работу.

Свойства:

* Алгоритм должен быть быстрым и использовать мало памяти
* Алгоритм должен иметь высокую реалистичность изображения
* Алгоритм должен работать как в объектном пространстве, так и в пространстве изображений

Рассмотрим алгоритмы для удаления невидимых ребер и поверхностей.

**1.4.1 Алгоритм Робертса**

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

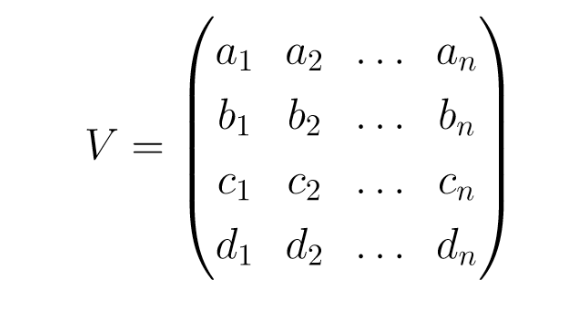
Алгоритм выполняется в три этапа:

1. **Этап подготовки исходных данных.**

На данном этапе задается информация о телах. Для каждого тела формируется матрица тела V. Размерность матрицы – 4 \* n, где n – количество граней тела.

Каждый столбец матрицы представляет собой четыре коэффициента уравнения плоскости ax + by + cz + d = 0, проходящей через очередную грань.

Таким образом, матрица тела будет представлена в следующем виде:



Матрица тела должна быть сформирована корректно, то есть любая точка, расположенная внутри тела, должна располагаться по положительную сторону от каждой грани тела. В случае, если для очередной грани условие не выполняется, соответствующий столбец матрицы надо умножить на -1. Для проверки надо взять точку, расположенную внутри тела. Координаты такой точки можно получить путем усреднения координат всех вершин тела.

1. **Этап удаления ребер, экранируемых самым телом.**

На данном этапе рассматривается вектор взгляда Е = {0, 0, -1, 0}. Чтобы определить невидимые грани необходимо умножить вектор Е на матрицу тела V. Отрицательные компоненты полученного вектора – невидимые грани.

1. **Этап удаления невидимых ребер, экранируемых другими телами сцены.**

На данном этапе строим луч, соединяющий точку наблюдения с точкой на ребре, для определения невидимых точек ребра. Точка невидима, если луч на своем пути встречает преграду – рассматриваемое тело. Если тело является преградой, то луч должен пройти через тело. Если луч проходит через тело, то он находится по положительную сторону от каждой грани тела.

**Преимущества и недостатки алгоритма Робертса**

**Преимущества:**

* алгоритм работает в объектном пространстве, высокая точность вычислений.

**Недостатки:**

* все тела сцены должны быть выпуклыми. Эта проблема также усложняет алгоритм, так как становится необходима проверка на выпуклость и разбиению на выпуклые многоугольники.

**Вывод**

Алгоритм Робертса не подходит для решения поставленной задачи по следующим причинам:

* не требуется такая точность визуализации объектов, которую предоставляет алгоритм;
* могут наблюдаться проблемы при наличии невыпуклых тел на сцене;
* реализация модификаций, позволяющих приблизить рост сложности алгоритма к линейной очень трудозатратна.

**1.4.2 Алгоритм, использующий Z-буфер**

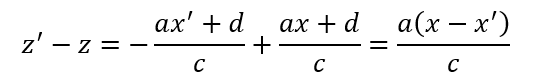
Суть данного алгоритма – использование двух буферов: буфера кадра, в котором хранятся атрибуты каждого пикселя, и Z-буфера, в котором хранится информация о координате Z для каждого пикселя.

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

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

Для решения задачи вычисления глубины Z каждый многоугольник описывается уравнением ax + bx+ cz +d = 0. При с = 0 многоугольник для наблюдателя вырождается в линию.

Для некоторой сканирующей строки y = const, поэтому имеется возможность рекуррентно высчитывать z' для каждого x' = x + dx:



Получим: z' = z/c, так как x – x' = dx = 1.

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

**Преимущества и недостатки алгоритма, использующего Z-буфер**

**Преимущества:**

* простая реализация
* оценка трудоемкости линейна
* экономия вычислительного времени, так как элементы сцены не сортируются

**Недостатки:**

* сложная реализация прозрачности
* большой объем требуемой памяти

**Вывод**

Алгоритм подходит по главному требованию – скорости работы с объектами. Простота алгоритма позволяет достаточно быстро реализовать и в полной мере отладить его.

Стоит отметить, что сцена состоит из малого количества объектов, что приведет к некритично большим затратам памяти для выполнения данного алгоритма.

**1.4.3 Алгоритм обратной трассировки лучей**

Суть данного алгоритма заключается в том, что наблюдатель видит объект с помощью испускаемого света, который по законам оптики доходит до наблюдателя некоторым путем. Отслеживать пути лучей от источника к наблюдателю неэффективно с точки зрения вычислений, поэтому лучшим способом будет отслеживание в обратном направлении, то есть от наблюдателя к объекту.

**Преимущества:**

* высокая реалистичность изображения
* вычислительная сложность не особо зависит от сложности сцены
* работа с поверхностями в математической форме

**Недостатки:**

* производительность

**Вывод:**

Данный алгоритм не отвечает главному требованию – скорости работы. Также от реализуемого продукта не требуется высокая реалистичность изображения и возможность работы с поверхностями, заданными в математической форме. Таким образом обратная трассировка не подходит для выполнения поставленной задачи.

**1.4.4 Алгоритм художника**

Данный алгоритм работает аналогично тому, как художник рисует картину – сначала рисуются дальние объекты, затем – ближние. Наиболее распространенная реализация алгоритма – сортировка по глубине, которая заключается в том, что множество граней сортируется по расстоянию от наблюдателя, затем отсортированные грани выводятся на экран в порядке от самой дальней до самой ближней. данный метод хорошо применим для сцен, в которых нет пересекающихся граней.

**Преимущества:**

* требуется меньше памяти, чем в алгоритме Z-буфера

**Недостатки:**

* недостаточно высокая реалистичность изображения
* сложность реализации при пересечении граней на сцене

**Вывод:**

Данный алгоритм не отвечает главному требованию – реалистичности изображения. Также алгоритм художника отрисовывает все грани, на что тратится большое количество времени.

**1.5 Выбор алгоритма построения теней**

При использовании алгоритма обратной трассировки лучей, рассмотренного ранее, построение теней происходит по ходу выполнения алгоритма: пиксель будет затемнен, если испускаемый луч попадает на объект, но не попадает в источник света. Алгоритм не подходит для решения поставленной задачи, так как при проведении анализа алгоритмов удаления невидимых ребер он не был выбран в качестве реализуемого.

В качестве алгоритма удаления невидимых линий и поверхностей в предыдущем подпункте был выбран алгоритм с использованием Z-буфера, поэтому одним из наилучших вариантов будет модификация указанного метода путем добавления вычисления теневого Z-буфера из точки наблюдения, совпадающей с источником света.

Такой подход позволяет не усложнять структуру программы, а также устранит проблему адаптации двух различных методов друг у другу, следовательно уменьшается время отладки.

**1.6. Вывод**

Были рассмотрены способы задания трехмерных моделей и выбрана поверхностная форма задания моделей. Также были рассмотрены алгоритмы удаления невидимых ребер.

В качестве реализуемого был выбран алгоритм, использующий Z-буфер, который отвечает двум требованиям – быстрота работы с множеством объектов и преобладание скорости над точностью. Для построения теней был выбран алгоритм, использующий теневые карты, в работе которого будет использоваться Z-буфер.

# 2. Конструкторская часть

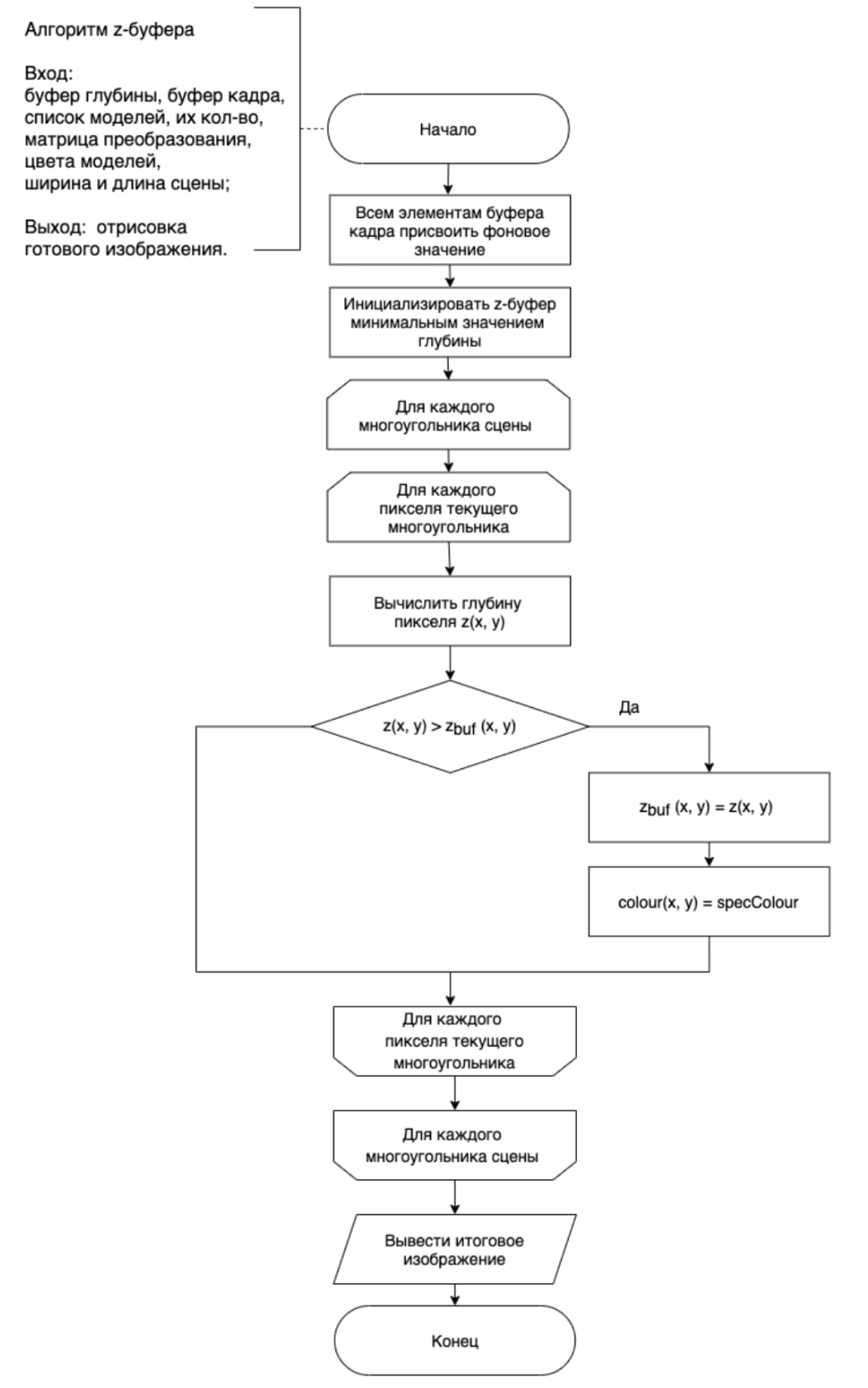
**2.1 Общий алгоритм решения поставленной задачи**

1. Задать объекты сцены(кольцо, щит, мяч)
2. Задать параметры для совершения броска
3. Рассчитать координаты конца броска, и в зависимости от этого, рассчитать, куда попадет мяч
   1. Если координаты конца броска находятся на щите, то мяч отскочит от щита и попадет/не попадет в кольцо
   2. Если координаты конца броска находятся в области кольца, то мяч отскочит о кольца/попадет в кольцо.
4. Смоделировать бросок

**2.2 Алгоритм, использующий Z-буфер**

* Всем элементам буфера кадра присвоить фоновое значение.
* Инициализировать Z-буфер минимальным значением глубины.
* Для каждого многоугольника сцены в произвольном порядке:
  + Вычисляем глубину каждого пикселя, который принадлежит многоугольнику
  + Сравниваем вычисленную глубину пикселя со значением, которое уже находится в Z-буфере.
* Вывести итоговое изображение

**Схема работы алгоритма, использующего Z-буфер.**



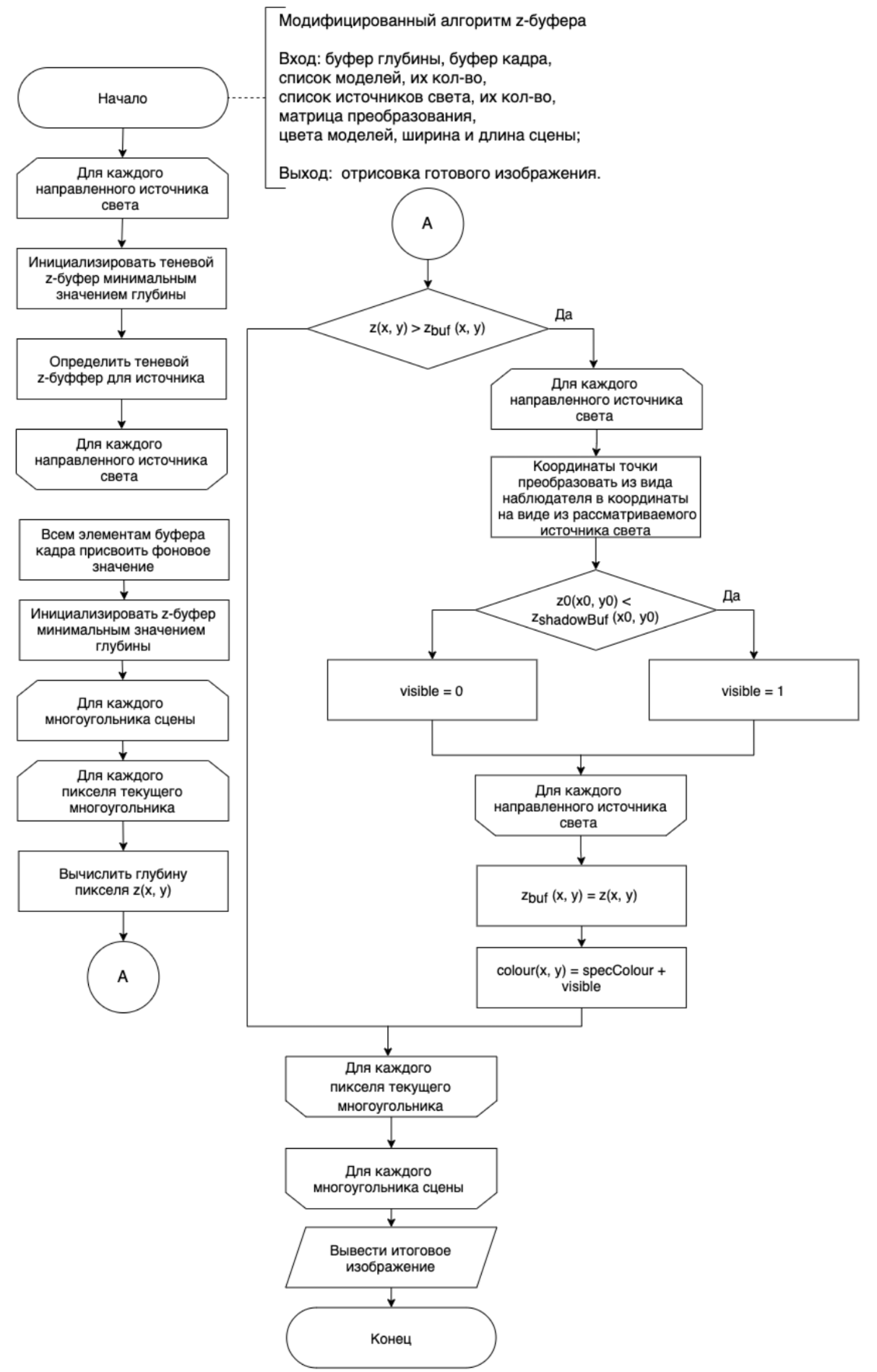
**2.3 Модифицированный алгоритм, использующий Z-буфер**

* Для каждого направленного источника света:
  + инициализировать теневой z-буфер минимальным значением глубины;
  + определить теневой z-буфер для источника.
* Выполнить алгоритм z-буфера для точки наблюдения. Если поверхность оказалась видимая относительно текущей точки наблюдения, то проверяем, видима ли данная точка со стороны источников света.

Для каждого источника света:

* + координаты точки, которую мы рассматриваем, линейно преобразовать из вида наблюдателя в координаты на виде из рассматриваемого источника света.
  + сравнить значение zshadowBuf(x', y') со значением z'(x', y'). Если z'(x', y') < zshadowBuf(x', y'), то пиксель высвечиваем с учетом затемнения, в обратном случае высвечиваем точку без затемнения.

**Схема модифицированного алгоритма, использующего Z-буфер.**



**2.4 Представление данных в программном обеспечении**

|  |  |
| --- | --- |
| **Данные** | **Представление** |
| Точка в трехмерном пространстве | Координаты по осям x, y, z |
| Полигон | Три точки трехмерного пространства |
| Объект сцены | Список полигонов |
| Источник света | Угол по осям X и Yотносительно текущей точки наблюдения |

**2.5 Вывод**

На основе теоретических данных, полученных из аналитического раздела, были описаны требования к программному обеспечению, общий алгоритм решения задачи, алгоритмы Z-буфера и модифицированного Z-буфера, а также обобщено представление данных в программном обеспечении.