Документ подписан простой электронной подписью
Информация о владельце:
ФИО: Таныгин Максим Олегович
Должность: и.о. декана факультета фундаментальной и прикладной информатики
Дата подписания: 21.09.2023 13:09:47
Уникальный программный ключ:
65ab2aa0d384efe8480e6a4c688eddbc475e411a

МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра программной инженерии

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О. Г. Локтионова
«10» 02 2022 г.

АЛГОРИТМЫ УДАЛЕНИЯ НЕВИДИМЫХ ЛИНИЙ И ПОВЕРХНОСТЕЙ

Методические указания по выполнению лабораторной работы
по дисциплине «Компьютерная графика»
для студентов всех форм обучения направления подготовки
09.03.04 «Программная инженерия»

Курск 2022

УДК 004.92
Составитель Е.А. Петрик

Рецензент
Кандидат технических наук, доцент Т.И.Лапина

Алгоритмы удаления невидимых линий и поверхностей: методические указания по выполнению лабораторной работы / Юго-Зап. гос. ун-т; сост.: Е. А. Петрик. Курск, 2022. 15 с.: ил.8. Библиогр.: с.15.

Содержат краткие теоретические сведения об алгоритмах удаления невидимых линий и поверхностей, а также приведены примеры и задания для лабораторной работы.

Методические указания соответствуют требованиям программы по направлению подготовки бакалавров: 09.03.04 «Программная инженерия»

Предназначены для студентов всех форм обучения направления подготовки бакалавров 09.03.04 «Программная инженерия»

Текст печатается в авторской редакции

Подписано в печать Формат 60х84 1/16.
Усл. печ. л. Уч. – изд. л. .Тираж экз. Заказ . Бесплатно.
Юго - Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94.

Цель работы

Изучение алгоритмов удаления невидимых линий и поверхностей, создание программы для визуализации работы алгоритмов.

Основные понятия

Алгоритмы удаления невидимых линий и поверхностей служат для определения линий ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства.

На рис. 1a приведен типичный каркасный чертеж куба. Удаление невидимых линий позволяет избавиться от неоднозначности в восприятии изображения на рис. 1, b,c.

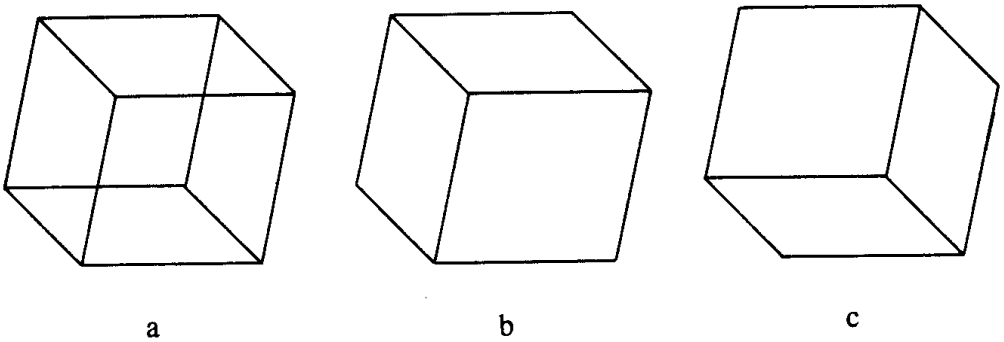

Рисунок 1 – Необходимость удаления невидимых линий

Алгоритмы удаления невидимых линий или поверхностей классифицируются по способу выбора системы координат или пространства, в котором они работают. Алгоритмы, работающие в объектном пространстве, имеют дело с физической системой координат, в которой описаны эти объекты. При этом получаются наиболее корректные результаты, ограниченные лишь точностью вычислений. Полученные изображения можно свободно увеличивать без ущерба в точности результата. Алгоритмы, работающие в пространстве изображения, имеют дело с системой координат того экрана, на котором объекты визуализируются. При этом точность вычислений ограничена разрешающей способностью экрана. Результаты, полученные в пространстве изображения, а затем увеличенные во много раз, не будут соответствовать исходной

сцене. Алгоритмы, формирующие список приоритетов, работают попеременно в обеих системах координат.

Объем вычислений для любого алгоритма, работающего в объектном пространстве, и сравнивающего каждый объект сцены со всеми остальными объектами этой сцены, растет теоретически как квадрат числа объектов ($n2$). Аналогично, объем вычислений любого алгоритма, работающего в пространстве изображения и сравнивающего каждый объект сцены с позициями всех пикселов в системе координат экрана, растет теоретически, как nN (n - число объектов, N - число пикселов). Как правило $n<N$, однако алгоритмы, работающие в пространстве изображения, более эффективны потому, что для них легче воспользоваться преимуществом когерентности при растровой реализации.

Алгоритм плавающего горизонта

Алгоритм чаще всего используется для удаления невидимых линий трехмерного представления функций, описывающих поверхность в виде

$$F(x, y, z) = 0.$$

Подобные функции возникают во многих приложениях в математике, технике, естественных науках.

Этот алгоритм обычно работает в пространстве изображения. Главная идея данного метода заключается в сведении трехмерной задачи к двумерной путем пересечения исходной поверхности последовательностью параллельных секущих плоскостей, имеющих постоянные значения координат x,y и z. На рис. 2 приведен пример, где указанные параллельные плоскости определяются постоянными значениями z. Функция $F(x,y,z) = 0$ сводится к последовательности кривых, лежащих в каждой из этих параллельных плоскостей, например к последовательности

$y=f(x,z)$ или $x=g(y,z)$,

где z постоянно на каждой из заданных параллельных плоскостей.

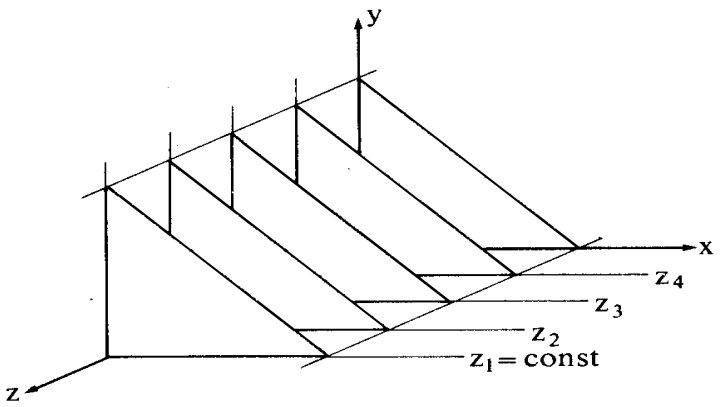

Рисунок 2 – Секущие плоскости с постоянной координатой.

На рис. 3 демонстрируется как поверхность складывается из последовательности кривых, лежащих в каждой из этих плоскостей.

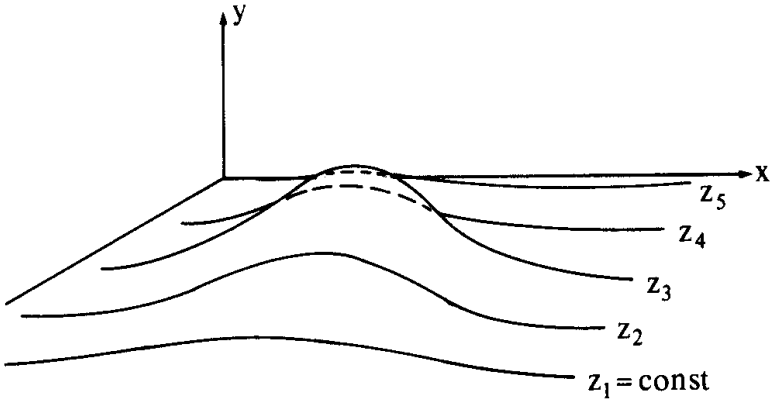

Рисунок 3 Кривые в секущих плоскостях с постоянной координатой.

Здесь предполагается, что полученные кривые являются однозначными функциями независимых переменных. Спроецируем полученные кривые на плоскость z=0, как показано на рис. 4.

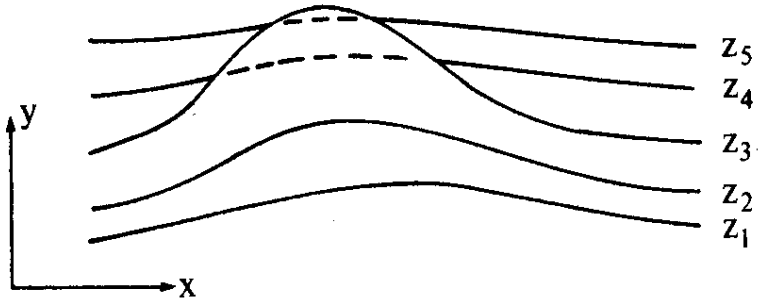

Рисунок 4 Проекция кривых на плоскость z=0

Алгоритм сначала упорядочивает плоскости z=const по возрастанию расстояния до них от точки наблюдения. Затем для каждой плоскости, начиная с ближайшей к точке наблюдения, строится кривая, лежащая на ней, т.е. для каждого значения

координаты x в пространстве изображения определяется соответствующее значение y. Алгоритм формулируется в виде:

Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше значения y для всех предыдущих кривых при этом значении x, то текущая кривая видима в этой точке; в противном случае она невидима.

Невидимые участки показаны пунктиром на рис. 4. Реализация данного алгоритма заключается в следующем. Для хранения максимальных значений y при каждом значении x используется массив, длина которого равна числу различимых точек (разрешению) по оси x в пространстве изображения. Значения, хранящиеся в этом массиве, представляют собой текущие значения "горизонта". Поэтому по мере рисования каждой очередной кривой этот горизонт "всплывает". Фактически этот алгоритм удаления невидимых линий работает каждый раз с одной линией.

Алгоритм работает хорошо до тех пор, пока какая-нибудь очередная кривая не окажется ниже самой первой из кривых, как показано на рис. 5, a. Подобные кривые видимы и представляют собой нижнюю сторону исходной поверхности, однако алгоритм будет считать их невидимыми.

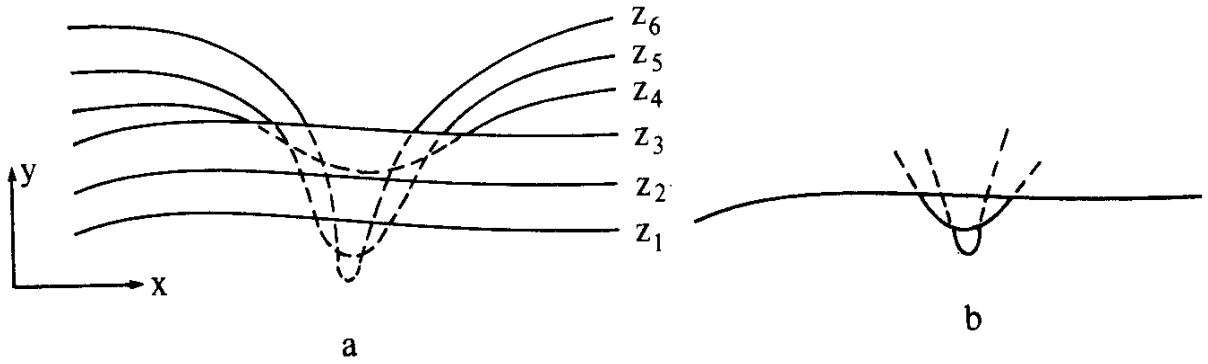

Рисунок 5 – Обработка нижней стороны поверхности.

Нижняя сторона поверхности делается видимой, если модифицировать этот алгоритм, включив в него нижний горизонт, который опускается вниз по ходу работы алгоритма. Это реализуется при помощи второго массива, длина которого равна числу различимых точек по оси x в пространстве изображения. Этот массив содержит наименьшие значения y для каждого значения x. Алгоритм формулируется следующим образом:

Если на текущей плоскости при некотором заданном значении x соответствующее значение y на кривой больше максимума или

меньше минимума по y для всех предыдущих кривых при этом x, то текущая кривая видима. В противном случае она невидима.

Полученный результат показан на рис. 5,b.

Алгоритм Робертса

Самым первым алгоритмом, предназначенным для удаления невидимых линии, был алгоритм Робертса, требующий, чтобы каждая грань была выпуклым многогранником.

Рассмотрим многогранник, для каждой грани которого задан единичный вектор внешней нормали. Несложно заметить, что если вектор нормали грани n составляет с вектором 1, задающим направление проектирования, тупой угол, то эта грань заведомо не может быть видна. Такие грани называются нелицевыми. В случае, когда соответствующий угол является острым, грань называется лицевой.

В случае параллельного проектирования условия на угол можно записать в виде:.(n, 1)≤90 , поскольку направление проектирования 1 от грани не зависит.

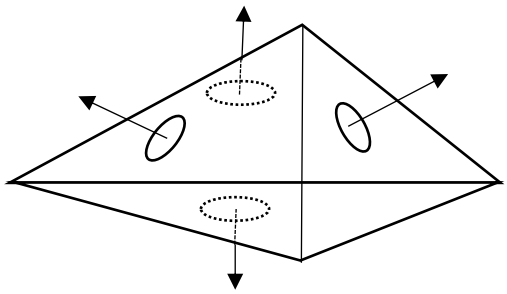

Рисунок 6 – Многогранник

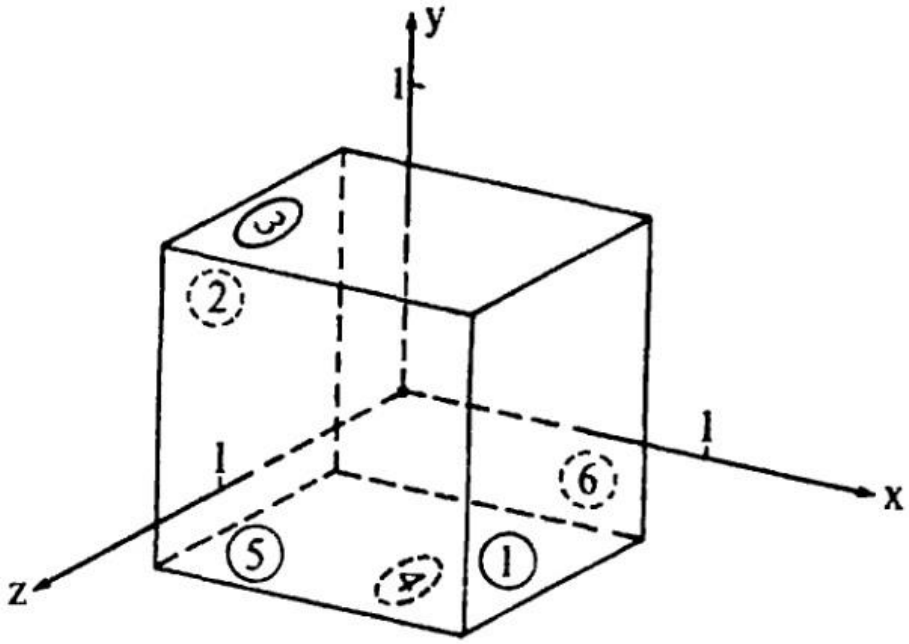

Рисунок 7 – Куб с центром в начале координат

При центральном проектировании с центром в точке с вектор проектирования для точки p будет равен: 1 = c - p.

Для определения того, является заданная грань лицевой или нет, достаточно взять произвольную точку p этой грани и проверить выполнение условия $(n,1) \leq 0$.

Знак этого скалярного произведения не зависит от выбора точки грани, а определяется тем, в каком полупространстве относительно плоскости, содержащей данную грань, лежит центр проектирования.

Когда сцена представляет собой один выпуклый многогранник, удаление нелицевых граней полностью решает задачу удаления невидимых граней.

В общем случае предложенный подход хотя и не решает задачу полностью, но позволяет примерно вдвое сократить количество рассматриваемых граней.

Опишем этот алгоритм.

1. Сначала отбрасываются все ребра, обе определяющие грани которых являются нелицевыми (ни одно из таких ребер не будет видно).

2. Следующим шагом является проверка каждого из оставшихся ребер со всеми гранями многогранника на закрывание. Возможны следующие случаи:

• грань не закрывает ребро;

• грань полностью закрывает ребро (и оно тогда удаляется из списка рассматриваемых ребер);

• грань частично закрывает ребро (в этом случае ребро разбивается на несколько частей, из которых видимыми являются не более двух; само ребро удаляется из списка, но в этот список проверенных ребер добавляются те его части, которые не закрываются данной гранью).

Метод z-буфера

Один из самых простых алгоритмов удаления невидимых граней и поверхностей является метод z-буфера (буфера глубины). В силу крайней простоты этого метода часто встречаются его аппаратные реализации.

Z-буфер – область видеопамяти, в которой для каждого пикселя хранится значение глубины. Поставим каждому пикселю (x, y) картинной плоскости, кроме цвета, хранящегося в видеопамяти, его расстояние до картинной плоскости вдоль направления проектирования z(x, y) (его глубину). Изначально массив глубин инициализируется $+\infty$.

Для вывода на картинную плоскость произвольной грани она переводится в свое растровое представление на картинной плоскости и для каждого пиксела этой грани находится его глубина. В случае, если эта глубина меньше значения глубины, хранящегося в z-буфере, пиксел рисуется и его глубина заносится в z-буфер.

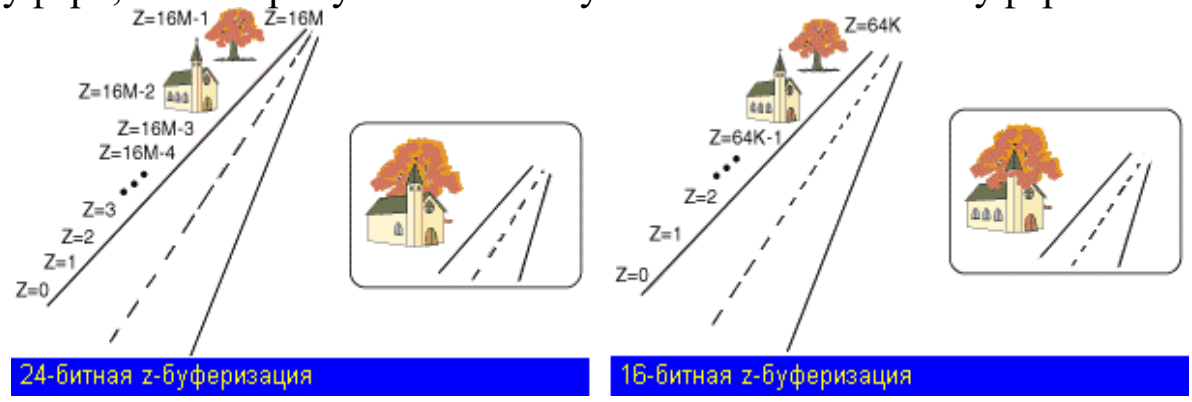

Рисунок 8 – Z-буферизация

При использовании z-буферизации надо позаботиться о том, чтобы глубины были корректны с точки зрения перспективы. Допустим, ускоритель рендерит треугольник с заданными z-

координатами трех его вершин. Он должен рассчитать z-координаты для всех точек, лежащих внутри треугольника. Если их просто интерполировать, то результат получится некорректным с точки зрения перспективы, поэтому их надо корректировать. Но современные 3D-ускорители используют технику, называемую **w-буфером**. W-координата – величина с плавающей точкой, обратная к z-координате. Всем вершинам ставятся в соответствие именно w-координаты, которые можно интерполировать без перспективной коррекции.

Это один из простейших алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмулом в 1975 г. Работает этот алгоритм в пространстве изображения. Идея Z-буфера является простым обобщением идеи о буфере кадра. Буфер кадра используется для запоминания атрибутов каждого пикселя в пространстве изображения, а Z-буфер предназначен для запоминания глубины (расстояния от картинной плоскости) каждого видимого пикселя в пространстве изображения. Поскольку достаточно распространенным является использование координатной плоскости в качестве картинной плоскости, то глубина равна координате точки, отсюда и название буфера. В процессе работы значение глубины каждого нового пикселя, который нужно занести в буфер кадра, сравнивается с глубиной того пикселя, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксель расположен впереди пикселя, находящегося в буфере кадра, то новый пиксель заносится в этот буфер и, кроме того, производится корректировка Z-буфера новым значением глубины. Если же сравнение дает противоположный результат, то никаких действий не производится. По сути, алгоритм является поиском по и наибольшего значения функции .

Главное преимущество алгоритма - его простота. Кроме того, этот алгоритм решает задачу об удалении невидимых поверхностей и делает тривиальной визуализацию пересечений сложных поверхностей. Сцены могут быть любой сложности. Поскольку габариты пространства изображения фиксированы, оценка вычислительной трудоемкости алгоритма не более чем линейна. Поскольку элементы сцены или картинки можно заносить в буфер кадра или в Z-буфер в произвольном порядке, их не нужно предварительно сортировать по приоритету глубины. Поэтому

экономится вычислительное время, затрачиваемое на сортировку по глубине.

Основной недостаток алгоритма - большой объем требуемой памяти. В последнее время в связи с быстрым ростом возможностей вычислительной техники этот недостаток становится менее лимитирующим. Но в то время, когда алгоритм еще только появился, приходилось изобретать способы создания буфера как можно большего объема при имеющемся ресурсе памяти.

Например, можно разбивать пространство изображения на 4, 16 или больше прямоугольников или полос. В предельном варианте можно использовать буфер размером в одну строку развертки. Для последнего случая был разработан алгоритм построчного сканирования. Поскольку каждый элемент сцены обрабатывается много раз, то сегментирование Z-буфера, вообще говоря, приводит к увеличению времени, необходимого для обработки сцены.

Другой недостаток алгоритма состоит в трудоемкости реализации эффектов, связанных с полупрозрачностью, и ряда других специальных задач, повышающих реалистичность изображения. Поскольку алгоритм заносит пиксели в буфер кадра в произвольном порядке, то довольно сложно получить информацию, которая необходима для методов, основывающихся на предварительном анализе сцены.

В целом алгоритм выглядит так:

1. Заполнить буфер кадра фоновым значением цвета.

2. Заполнить Z -буфер минимальным значением z (глубины) .

3. Преобразовать изображаемые объекты в растровую форму в произвольном порядке.

4. Для каждого объекта:

4.1. Для каждого пикселя образа вычислить его глубину .

4.2. Сравнить глубину со значением глубины, хранящимся в Z-буфере в этой же позиции.

4.3. Если , то занести атрибуты пикселя в буфер кадра и заменить . В противном случае никаких действий не производить.

Алгоритм, использующий Z-буфер, можно также применять для построения сечений поверхностей. Изменится только оператор сравнения:

где - глубина искомого сечения.

Алгоритмы упорядочения

Подход заключается в таком упорядочении граней, чтобы при их выводе в этом порядке получалось конкретное изображение. Для этого необходимо, чтобы более дальние грани выводились раньше, чем более близкие.

Метод сортировки по глубине. Наиболее простым подходом к упорядочиванию граней является их сортировка по минимальному расстоянию до картинной плоскости (вдоль направления проектирования) с последующим выводом их в порядке приближения.

Этот метод великолепно работает для ряда сцен, включая, например, построения изображения нескольких непересекающихся достаточно простых тел.

Однако возможны случаи, когда просто сортировка по расстоянию до картинной плоскости не обеспечивает правильного упорядочивания граней; поэтому желательно после такой сортировки проверить порядок, в котором грани будут выводиться.

Предлагается следующий алгоритм этой проверки. Для простоты будем считать, что рассматривается параллельное проектирование вдоль оси Oz.

Перед выводом грани P следует убедиться, что никакая другая грань Q, проекция которой на ось Oz пересекается с проекцией грани P, не может закрываться гранью P. И если это условие выполнено, то грань P должна быть выведена раньше. Предлагаются следующие 5 тестов в порядке возрастания сложности проверки:

Пересекаются ли проекции этих граней на ось Ox?

Пересекаются ли их проекции на ось Oy?

Находится ли грань P по другую сторону от плоскости, проходящей через грань Q, чем начало координат (наблюдатель)?

Находится ли грань Q по ту же сторону от плоскости, проходящей через грань P, что и начало координат (наблюдатель)?

Пресекаются ли проекция этих граней на картинную плоскость?

Если хотя бы на один из этих вопросов получен отрицательный ответ, то считаем что эти две грани - P и Q упорядочены верно, и сравниваем P со следующей гранью. В противном случае считаем, что эти грани необходимо поменять местами, для чего проверяются следующие тесты:

* Находится ли грань Q по другую сторону от плоскости, проходящей через грань P, чем начало координат?

* Находится ли грань P по ту же сторону от плоскости, проходящей через грань Q, что и начало координат?

В случае, если ни один из этих тестов не позволяет с уверенностью решить, какую из этих двух граней нужно выводить раньше, то одна из них разбивается плоскостью, проходящей через другую грань. В этом случае вопрос об упорядочении оставшейся грани и частей разбитой грани легко решается.

Метод двоичного разбиения пространства. Существует другой, крайне элегантный способ упорядочивания граней.

Рассмотрим некоторую плоскость в объектном пространстве. Она разбивает множество всех граней на два непересекающихся множества (кластера), в зависимости от того, в каком полупространстве относительно плоскости эти грани лежат (будем считать, что плоскость не пересекает ни одну из этих граней).

При этом очевидно, что ни одна из граней, лежащих в полупространстве, не содержащем наблюдателя, не может закрывать собой ни одну из граней, лежащих в том же пространстве, что и наблюдатель. Тем самым сначала необходимо вывести грани из дальнего кластера, а затем уже и из ближнего.

Применим подобную технику для упорядочения граней внутри каждого кластера. Для этого построим разбиение граней каждого кластера на два множества очередной плоскостью; а затем для вновь полученных граней повторим процесс разбиения, и будем поступать до так, до тех пор, пока в каждом получившемся кластере останется не более одной грани.

Обычно в качестве разбивающей плоскости рассматривается плоскость, проходящая через одну из граней (на самом деле при этом множество всех граней разбивается на 4 класса - лежащих на плоскости, пересекающих ее, лежащих в положительном полупространстве и лежащие в отрицательном полупространстве относительно этой плоскости). Все грани, пересекаемые плоскостью, разобьем вдоль этой плоскости.

В результате мы приходим к дереву разбиения пространства, узлами которого являются грани.

Процесс построения дерева заключается в выборе грани, проведении через нее плоскости и разбиении множества всех граней. В этом процессе присутствует определенный произвол в выборе очередной грани. Существует два основных критерия для выбора:

- получить как можно более сбалансированное дерево;

- минимизировать количество разбиений.

К сожалению, эти критерии, как правило, являются взаимоисключающими, поэтому выбирается некоторый компромиссный вариант.

После того как дерево построено, осуществляется построение изображения в зависимости от используемого проектирования.

Одним из основных преимуществ этого метода является его полная независимость от положения центра проектирования, что делает его крайне удобным для построения серий изображений одной и той же сцены из разных точек наблюдения.

Задание

Написать программу (на языке высокого уровня), реализующую алгоритмы плавающего горизонта и z-буфера.

Содержание отчета

Отчет должен содержать:

1. Титульный лист.
2. Задание.
3. Блок-схемы алгоритмов.
4. Листинг программы.
5. Пример работы программы.
6. Ответы на контрольные вопросы.

Контрольные вопросы

1. Какие существуют алгоритмы удаления невидимых линий и поверхностей?
2. Приведите последовательность действий одного из алгоритмов?

Список используемой литературы

1. Аммерал, Л. Принципы программирования в машинной графике / Л. Аммерал; пер. с англ. – Москва : Сол Систем, 1992. – 224 с. – ISBN 5-85316-001-X. – Текст : непосредственный.
2. Роджерс, Д. Алгоритмические основы машинной графики / Д. Роджерс; пер. с англ. – Москва : Мир, 1989. – 512 с. – ISBN 5-03-000476-9. – Текст : непосредственный.
3. Роджерс, Д. Математические основы машинной графики / Д. Роджерс, Дж. Адамс; пер. с англ. – Москва : Мир, 2001. – 604 с. – ISBN 5-03-002143-4. – Текст : непосредственный.
4. Шикин, Е. В. Начала компьютерной графики / Е. В. Шикин, А. В. Боресков, А. А. Зайцев. – Москва : ДИАЛОГ-МИФИ, 1993. – 138 с. – ISBN 5-86404-035-5. – Текст : непосредственный.