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 Платонова Ольга Сергеевна
фамилия, имя, отчество

Группа ИУ7-55Б

Тип практики стационарная

Название предприятия МГТУ им. Н. Э. Баумана, каф. ИУ7

Студент _____ Платонова О. С.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Куров А. В.
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2020 г.

Содержание

Введение

Компьютерная графика в настоящее время уже вполне сформировалась как наука. Существует аппаратное и программное обеспечение для получения различных изображений – от простых чертежей до реалистичных объектов трехмерного пространства. Компьютерная графика затрагивает практически все дисциплины: от биологии и медицины, до бизнеса и индустрии развлечений.

В связи с этим возникает задача разработки алгоритмов, которые смогут быстро, с минимальными затратами ресурсов построить качественное трехмерное реалистичное изображение. Построение такого изображения включает в себя как физические, так и психологические процессы. Повышение реалистичности изображения может быть достигнуто с учетом оптических явлений преломления, отражения и рассеивания света, путем воспроизведения теней, свойств и текстуры поверхностей.

На данный момент разработано множество алгоритмов, решающих задачу построения реалистичного изображения трехмерного пространства.

Целью данной курсовой работы является разработка приложения для моделирования конструктора из заданного набора деталей.

Для реализации указанной цели необходимо решить следующие задачи:

- разработать структуру программного продукта, реализующего построение конструктора на трехмерной сцене;
- проанализировать существующие алгоритмы, позволяющие выполнять построение трехмерной сцены;
- реализовать выбранные алгоритмы, выполняющие построения на сцене;
- разработать программное обеспечение, соответствующее заданным критериям.

1. Аналитическая часть

В данном разделе рассмотрим задачи и методы их решения визуализации сцены, которые будет выполнять разработанное программное обеспечение.

1.1. Формализация объектов сцены

Объекты сцены:

1. Деталь. Сложный объект: состоит из многоугольника в основании и цилиндров – креплений (рисунок 1). Данные о детали хранятся в виде списка ребер и вершин. Объект видимый.

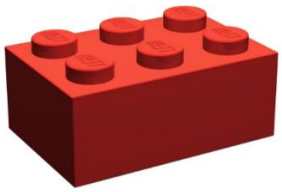

Рисунок 1. Деталь.

2. Основание для конструктора. Равномерная прямоугольная поверхность с пазами для деталей. Объект видимый.
3. Источник света. Испускает параллельные лучи, от угла падения которых зависит интенсивность цвета и тени деталей. Описывается точкой и углом наклона. Объект невидимый.

1.2. Постановка задачи

Требуется реализовать программное обеспечение, позволяющее выполнять следующие операции:

- добавлять на сцену детали из списка заданных;
- вращать и перемещать выбранную деталь в произвольном направлении;

- «составлять» две выбранные детали;
- вращать сцену в произвольном направлении.

Рассмотрим подробнее процесс «составления» двух деталей. При добавлении нового элемента на сцену, может возникнуть ситуация, когда элементы, установленные ранее, окажутся для пользователя невидимыми полностью или частично. Также, во время вращения сцены, набор невидимых элементов будет изменяться. Для решения поставленной задачи, требуется алгоритм, позволяющий быстро удалять невидимые линии и поверхности.

1.3. Анализ алгоритмов удаления невидимых линий и поверхностей

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в машинной графике, что привело к большому числу различных способов ее решения.

Выбор подходящего решения следует основывать на особенностях указанной задачи: скорость работы. Поскольку пользователь выполняет вращение сцены в произвольном направлении с произвольной скоростью, от алгоритма требуется быстрота работы. Также известно, что на сцене отсутствуют крупные тела вращения, и большинство деталей имеют форму многоугольника. При этом следует учитывать, что скорость и детальность реализации обратно пропорциональны, поэтому построенная модель не сможет достигнуть хорошей детальности.

1.3.1. Алгоритм Робертса

Алгоритм работает в объектном пространстве и выполняет удаление сначала из каждого тела те ребра или грани, которые экранируются самим телом.

Затем каждое из видимых ребер сравнивается с каждым из оставшихся тел для определения того, какая его часть экранируются этими телами.

Вычислительная трудоемкость алгоритма растет как квадрат числа объектов. Поскольку число объектов на сцене не ограничено, и возможно построение сложных фигур с большим числом элементов, алгоритм Робертса не сможет поддержать требуемую быстроту вычислений. Однако, стоит отметить простоту и точность методов, используемых в указанном алгоритме.

1.3.2. Алгоритм Варнока

В алгоритме осуществляется попытка извлечь преимущество из факта однородности больших областей изображения. В пространстве изображений рассматривается окно и решается вопрос: пусто оно, или его содержимое достаточно просто. Если это не так, то окно разбивается на фрагменты до тех пор, пока ответ на вопрос не станет утвердительным.

Было предложено множество модификаций алгоритма, связанных с вариантами разбиения окна. Несмотря на то, что алгоритм достаточно прост и понятен, рекурсивное разбиение окна в некоторых случаях будет работать недостаточно быстро для динамической работы со сценой. Множественное разбиение окна может возникнуть при обработке верхних граней деталей, так как они являются сложной конструкцией, содержащей цилиндры (рис. 1).

1.3.3. Алгоритм, использующий z-буфер

Рассматриваемый алгоритм является одним из простейших алгоритмов, работающих в пространстве изображения. Работа алгоритма основывается на идеи создания отдельного буфера глубины, используемого для запоминания глубины каждого видимого пиксела. В процессе работы значение z каждого нового пиксела, претендующего на занесение в буфер, сравнивают с глубиной

того пиксела, который уже занесен в z-буфер. Несмотря на многочисленные сравнения пикселов по глубине, скорость алгоритма высока из-за отсутствия предварительной сортировки. Следовательно, сцены могут быть любой сложности.

Основной недостаток – большой объем требуемой памяти, однако на сегодняшний день использование большого объема оперативной памяти не является существенным недостатком предложенного алгоритма.

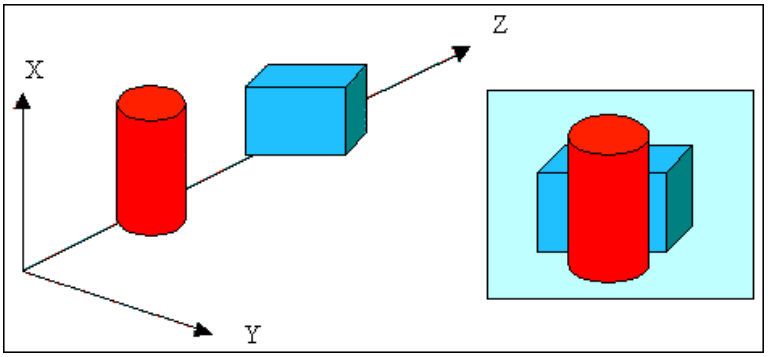

Рисунок 2. Пример работы алгоритма z-буфера.

1.3.4. Алгоритм обратной трассировки лучей

Идея алгоритма заключается в том, что происходит наблюдение за траекторией каждого луча света, испускаемого от наблюдателя. В случае, если луч пересекает объект сцены, то определяются все возможные точки пересечения луча и объекта. Наиболее важным и долгим этапом в работе алгоритма является процедура определения пересечений.

Поскольку в поставленной задачи известны некоторые из параметров сцены (например, большинство деталей конструктора являются многоугольниками), а рассмотренный алгоритм не использует того обстоятельства, что некоторые грани многогранника являются нелицевыми и их можно сразу удалить, он не является оптимальным решением указанной задачи. Алгоритм предназначен для решений более общих задач, что в данном случае

приведет к излишним вычислениям, и, как следствие, к замедлению времени работы.

1.3.5. Используемый алгоритм удаления невидимых граней

При реализации программного продукта использован алгоритм z-буфера, позволяющий довольно легко и быстро осуществить удаление невидимых граней и плоскостей, независимо от количества деталей на сцене. В качестве предварительного шага удаляются нелицевые грани, что снижает количество сравнений и увеличивает скорость работы. Также для увеличения скорости, вновь добавленную деталь следует заносить не в конец списка элементов на сцене, а в соответствии с ее расположением относительно z-координаты. Таким образом, оценка вычислительной трудоемкости алгоритма не более, чем линейна.

1.4. Анализ алгоритмов закраски

Поскольку модель конструктора предполагает, что основными пользователями являются дети – цвета деталей должны быть яркими и насыщенными. Также должна быть предусмотрена возможность изменения цвета выбранной детали. Рассмотрим различные алгоритмы закраски, позволяющие быстро и натурально выполнять закрашивание детали.

Метод Гуро основан на билинейной интерполяции, с помощью которой вычисляется интенсивность каждого пиксела на сканирующей строке. Такие недостатки метода, как эффект полос Маха и одинаковая интенсивность на складчатой поверхности, будут незначительны в указанной задаче, потому что на сцене отсутствуют массивные тела вращения и расположены лишь выпуклые фигуры. Однако, в ключе поставленной задачи возникает и другая проблема – сглаживание изображения.

Закраска Фонга позволяет разрешить проблемы метода Гуро, путем вычисления билинейной интерполяцией вектора нормали, что приводит к большим вычислительным затратам. Также указанный метод позволяет получить более реалистичное изображение, в том числе правдоподобнее выглядят зеркальные блики.

Поскольку в указанной задачи не требуется сглаживание, отсутствуют крупные тела вращения, рационально использовать метод, основанный на *простой модели освещения*. Работа этого метода заключается в закрашивании всей грани одним уровнем интенсивности, рассчитывающимся по закону Ламберта. В результате использования этого метода поверхность предметов на сцене выглядит матовой. Выбор простой модели позволяет экономить ресурсы: сокращает время работы алгоритма (не требуются вычисления нормалей и перебор сканирующих строк), уменьшает объем требуемой памяти (что важно при реализации z-буфера, требующего большое количество памяти), а также не выполняет сглаживание, которое в данной работе просто излишне.

1.5. Анализ алгоритмов построения теней

Как было указано выше, на сцене может располагаться источник света, и в случае несовпадения положения источника и наблюдателя возникают тени. Изображение с построенными тенями выглядит гораздо реалистичней. Для построения теней требуется дважды удалить невидимые поверхности. Поскольку выбор алгоритма удаления невидимых граней был сделан в пользу z-буфера, его требуется модифицировать, чтобы он включал построение теней.

1.6. Модель освещения

Существуют локальная и глобальная модели освещения, которые различны в учете факторов при расчете интенсивности отраженного к наблюдателю света в каждой точке. В случае локальной модели во внимание

принимается только свет, падающий от источника, и ориентация поверхности, при реализации глобальной модели – учитывается также свет, отраженный от других объектов сцены.

Глобальная модель освещения является более точной, так как передает реальное распространение света. Но поскольку в указанной задаче ведется работа с матовыми поверхностями, точность локальной модели достаточна для визуализации рассматриваемых объектов. Так же, в отличие от глобальной модели, не требуется трассировка лучей, что позволяет ускорять работу программы.

1.7. Вывод

В данном разделе был произведен сравнительный анализ алгоритмов удаления невидимых поверхностей и ребер, алгоритмов закраски и построения теней. В результате анализа было принято решение о реализации алгоритма Z-буфера (в связи с его простотой и скоростью, относительно других алгоритмов, и возможностью на сегодняшний день предоставлять необходимый объем оперативной памяти) и метода простой закраски (из-за простоты его реализации и возможностью пренебрежения его недостатками в конкретной задаче). Также было принято решение о работе с локальной моделью освещенности в целях ускорения работы программы.

2. Конструкторская часть

Разработаем алгоритмы, которые будут использованы в программном обеспечении, с учетом описанных выше рассуждений.

2.1. Требования к программному обеспечению

Прежде всего, программное обеспечение должно быть понятно пользователю: надписи меню должны быть на русском языке, должно быть доступно два режима: моделирования и обзора. Также программа должна предоставлять следующие возможности:

- добавление новой детали на сцену;
    - вращение детали
    - перемещение детали
    - составление деталей
- вращение исходной сцены в произвольном направлении.

2.2. Преобразования деталей

Ранее мы указывали, что вновь помещаемая на сцену деталь должна быть доступна пользователю следующим образом:

- поворот;
- перенос.

Для реализации перечисленных операций воспользуемся формулами преобразований, изученными в курсе компьютерной графики.

Перенос

Формулы переноса точки (x; y; z) в точку $(x_1;\ y_1;\ z_1)$

$$\begin{cases} x_1 = x + dx \\ y_1 = y + dy \\ z_1 = z + dz \end{cases} \quad (1)$$

где dx, dy, dz – смещение соответственно по оси Ox, Oy, Oz.

Поворот

Формулы поворота точки (x; y; z) на угол a.

Вокруг оси Ox

$$\begin{cases} x_1 = x \\ y_1 = y\cos a + z\sin a \\ z_1 = y\sin a + z\cos a \end{cases} \quad (2)$$

Вокруг оси Oy

$$\begin{cases} x_1 = x\cos a + z\sin a \\ y_1 = y \\ z_1 = -x\sin a + z\cos a \end{cases} \quad (3)$$

Вокруг оси Oz

$$\begin{cases} x_1 = x\cos a - y\sin a \\ y_1 = -x\sin a + y\cos a \\ z_1 = z \end{cases} \quad (4)$$

2.3. Преобразования сцены

Аналогично отдельно взятой детали, пользовать должен иметь возможность работы со сценой, а именно - вращать сцену.

Формулы поворота для детали верны и в случае поворота сцены, так как под вращением сцены подразумевается вращение всех элементов сцены.

2.4. Разработка алгоритма, использующего z-буфер

1. Удалить нелицевые грани.
2. Заполнить z-буфер фоновым значением интенсивности.
3. Проинициализировать z-буфер минимальным значением z.
4. Преобразовать деталь в растровую форму в порядке добавления на сцену.
5. Вычислить глубину каждого пиксела многоугольника.
6. Сравнить глубину z(x, y) с соответствующим значением Z_b(x, y) в z-буфере.
    a. Если z(x, y) > Z_b(x, y), то записать атрибут этого многоугольника (интенсивность, цвет и т. п.) в буфер кадра и заменить Z_b(x, y) на z(x, y).
7. Отобразить результат.

Вычисление глубины пиксела

Для вычисления глубины каждого пиксела на сканирующей строке воспользуемся следующими формулами:

$$ax + by + cz + d = 0 \qquad (5)$$

$$z = \frac{-(ax + by + d)}{c} \neq 0$$

Поскольку y = const (сканирующая строка), x_1 = x + Δx (следующий пиксел сканирующей строки), то глубину пиксела можно рассчитать по формуле:

$$z_1 - z = \frac{-(ax + d) + (ax + d)}{c} = \frac{a(x - x_1)}{c}$$

$$z_1 = z - \frac{a}{c}\Delta x \qquad (6)$$

Формула *(5)* – уравнение плоскости.

2.5. Разработка алгоритма простой закраски

В основе простой модели освещений лежит закон косинусов Ламберта:

$$I = I_l k_d \cos\theta \text{, где} \quad (7)$$

- I – интенсивность отраженного света,
- I_l – интенсивность точечного источника,
- k_d – коэффициент диффузного отражения,
- θ – угол между направлением света и нормалью к плоскости.

Поскольку была реализована локальная модель освещения, в формуле (7) не учитывается слагаемое рассеянного света.

2.6. Разработка общего алгоритма визуализации сцены

Рассмотрим общий алгоритм визуализации объектов сцены.

1. Установка первоначальных объектов сцены.
2. Установка источника света, положения наблюдателя.
3. Добавление детали на сцену:
    a. выбор параметров детали (форма, цвет);
    b. вращение детали, перемещение по сцене;
    c. составление детали с другими (удаление линии пересечения);
    d. добавление детали в список зарегистрированных на сцене;
4. Для каждого полигона высчитать нормаль и интенсивность цвета;
5. Применить алгоритм z-буфера:
    a. получить изображение сцены;
    b. получить буфер глубины;
6. Применить алгоритм z-буфера:
    a. получить буфер глубины для источника света;
7. Получить список невидимых элементов;
8. Отобразить изображение;
9. Вращение сцены:

a. применить алгоритм вращения ко всем элементам сцены;
b. вернуться к п. 2;

10. Конец.

Отметим, что изначально на сцене расположено размеченное поле, куда впоследствии будут добавляться детали.

Каждая деталь описывается структурой Detail, содержащей поля:

- количество вершин;
- массив координат каждой из вершин;
- цвет детали.

Описание сцены – структура Scene:

- количество деталей на сцене;
- массив объектов Detail;
- положение источника света.

2.7. Вывод

В данном разделе были разработаны алгоритмы удаления невидимых граней и поверхностей, закраски, работы с деталью и сценой; и был представлен общий алгоритм работы программного продукта.

Литература

1. Д. Роджерс. Алгоритмические основы машинной графики: Пер. с англ. – М.: Мир, 1989.-512с., ил.
2. А. Ю. Дёмин. Компьютерная графика / А. Ю. Дёмин, А. В. Кудинов – Режим доступа: http://www.compgraph.tpu.ru/index.html (дата обращения: 29.07.2020).
3. R. Szeliski. Computer Vision: Algorithms and Applications, 2nd ed., 2020
4. Е. А. Снижко. Компьютерная геометрия и графика [Текст], 2005. - 17 с.
5. Ю. Тихомиров. Программирование трехмерной графики – Спб: BHV – Санкт-Петербург, 1998 – 256с.: ил.