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 УЧЕБНОЙ ПРАКТИКЕ

Студент Артемьев Илья Олегович
фамилия, имя, отчество

Группа ИУ7-43Б

Тип практики стационарная

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

Студент _____ Артемьев И. О.
подпись, дата *фамилия, и.о.*

Руководитель практики
_____ Исаев А. Л.
подпись, дата *фамилия, и.о.*

Оценка _____

Москва, 2021 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, каф. ИУ7

Студент Артемьев Илья Олегович ИУ-43Б
(фамилия, имя, отчество; инициалы; индекс группы)

Во время прохождения производственной практики студент должен:

1. Разработать ПО с пользовательским интерфейсом, моделирующее помещение с несколькими комнатами, в которых распространяется вирус. Также нужно учесть движение воздуха и расположить в некоторых комнатах очистители воздуха. Пользователю должна быть доступна возможность перемещения камеры относительно сцены.

2. Проанализировать алгоритмы, выбрать наиболее подходящие для решения поставленной задачи.

3. Составить отчёт.

Дата выдачи задания «____» _____ 20__ г.

Руководитель практики от кафедры _____ / Исаев А. Л.
(подпись, дата) (Фамилия И.О.)

Студент _____ / Артемьев И. О.
(подпись, дата) (Фамилия И.О.)

Оглавление

Введение

В наше время компьютерная графика имеет достаточно широкий охват применения во многих отраслях нашей жизни: для наглядного отображения данных, в компьютерных играх и даже в кино для создания эффектов.

Вследствие этого перед специалистами, создающими трёхмерные изображения, появляется множество трудностей связанных с учётом таких явлений, как преломление, отражение и рассеивание света, а для достижения максимально реалистичного изображения также учитываются дифракция, интерференция, вторичные отражения света, цвет и текстура объектов.

В компьютерной графике существует множество различных алгоритмов, помогающих решить эти задачи, однако зачастую большинство являются достаточно ресурсоёмкими, так как чем более реалистичное изображение мы хотим получить, тем больше нам необходимо времени и памяти на синтез. Это, пожалуй, самая главная проблема при создании реалистичных изображений (особенно динамических сцен), которую пытаются решить и по сей день.

Целью моей практики будет выбор наиболее подходящих алгоритмов и реализация данных алгоритмов для создания трёхмерной сцены.

1. Аналитическая часть

1.1 Формализация объектов синтезируемой сцены

Сцена состоит из:

- Пола помещения – ограничивающая плоскость, под которой не могут находиться объекты.
- Стен помещения – прямоугольные параллелепипеды с основаниями параллельными полу.
- Очистителей воздуха – простые цилиндрические объекты с основаниями параллельными полу.
- Вируса – частички с траекторией движения.

Для описания трёхмерных геометрических объектов существует три модели: каркасная, поверхностная и объёмная. Для реализации поставленной задачи, наиболее подходящей моделью будет поверхностная, так как, по сравнению с каркасной, она может дать более реалистичное изображение. В то же время, объёмной модели требуется больше памяти и поэтому она будет излишней, в условиях моей задачи.

Поверхностную модель можно задать несколькими способами:

Параметрическим представлением – для получения поверхности нужно вычислять функцию, зависящую от параметра. Так как в сцене нет никаких поверхностей вращения, то использование параметрического представления будет затруднительно.

Полигональной сеткой – совокупностью вершин, ребер и граней, которые определяют форму объекта.

- Вершинное представление (вершины указывают на другие вершины, с которыми они соединены). Для генерации списка граней для рендеринга нужно обойти все данные, что затрудняет работу с ним.
- Список граней представляет объект как множество граней и вершин.

- Таблица углов - хранит вершины в предопределенной таблице. Не подойдет для моей задачи, так как изменение данного представления наиболее затратно по времени.

Самым удобным способом хранения моей сцены является список граней, так как данные в нем можно эффективно преобразовывать, а также представление позволяет явный поиск вершин грани и граней, окружающих вершину.

1.2 Анализ алгоритмов удаления невидимых линий и поверхностей

Алгоритм трассировки лучей

Данный алгоритм обладает следующими преимуществами:

- вычислительная сложность слабо зависит от сложности сцены.
- отсечение невидимых поверхностей, перспектива и корректное изменения поля зрения являются логическим следствием алгоритма.
- возможность рендеринга гладких объектов без аппроксимации их полигональными поверхностями.
- возможность параллельно и независимо трассировать два и более лучей, разделять участки для трассирования на разных узлах кластера и т.д.

Однако, недостатком алгоритма является его производительность. Метод трассирования лучей каждый раз начинает процесс определения цвета пикселя заново, рассматривая каждый луч наблюдения в отдельности.

В поставленной мною задаче не используются явления отражения и преломления света и поэтому некоторые вычисления окажутся излишними, кроме того, скорость синтеза сцены должна быть высокой, поэтому алгоритм трассировки лучей не подходит.

Алгоритм Варнока

Преимуществом данного алгоритма является то, что он работает в пространстве изображений. Алгоритм предлагает разбиение области рисунка на более мелкие окна, и для каждого такого окна определяются связанные с ней многоугольники и те, видимость которых «легко» определить, изображаются на сцене.

К недостаткам можно отнести то, что в противном случае разбиение повторяется, и для каждой из вновь полученных подобластей рекурсивно применяется процедура принятия решения. По предположению, с уменьшением размеров области, её будет перекрывать всё меньшее количество многоугольников. Считается, что в пределе будут получены области, которые содержат не более одного многоугольника, и решение будет принято достаточно просто.

Из-за того, что поиск может продолжаться до тех пор, пока либо остаются области, содержащие не один многоугольник, либо пока размер области не станет совпадать с одним пикселом, алгоритм не подходит для условия моей задачи.

Алгоритм Робертса

Данный алгоритм обладает достаточно простыми и одновременно мощными математическими методами, что, несомненно, является преимуществом. Некоторые реализации, например использующие предварительную сортировку по оси z, могут похвастаться линейной зависимостью от числа объектов на сцене.

Однако алгоритм обладает недостатком, который заключается в вычислительной трудоёмкости, растущей теоретически, как квадрат числа объектов, что может негативно сказаться на производительности.

Алгоритм, использующий Z буфер

Главными преимуществами данного алгоритма являются его достаточная простота реализации и функциональность, которая более чем подходит для визуализации динамической сцены. Также алгоритм не тратит время на сортировку элементов

сцены, что даёт значительный прирост к производительности.

К недостатку алгоритма можно отнести большой объём памяти необходимый для хранения информации о каждом пикселе. Однако данный недостаток является незначительным ввиду того, что большинство современных компьютеров обладает достаточно большим объёмом памяти для корректной работы алгоритма.

Вывод:

Алгоритм, использующий Z буфер, является наиболее подходящим для построения динамической сцены визуализации распространения вируса в помещении. На его основе будет просто визуализировать частички вируса, можно накладывать их на уже посчитанный Z буфер сцены, не проводя повторных расчетов.

1.3 Анализ методов закрашивания

Простая закраска

По закону Ламберта, вся грань закрашивается одним уровнем интенсивности. Метод является достаточно простым в реализации и не требовательным к ресурсам. Однако, алгоритм плохо учитывает отражения и при отрисовки тел вращения, возникают проблемы.
Но для моей задачи этот метод очень хорошо подходит, так как вся работа ведется с гранями стен и очистителей воздуха, тел вращения нет.

Закраска по Гуро

В основу данного алгоритма положена билинейная интерполяция интенсивностей, позволяющая устранять дискретность изменения интенсивности. Благодаря этому криволинейные поверхности будут более гладкими.

Закраска по Фонгу

В данном алгоритме за основу берётся билинейная интерполяция векторов нормалей, благодаря чему достигается лучшая локальная аппроксимация кривизны поверхности и, следовательно, изображение выглядит более реалистичным.

Однако алгоритм требует больших вычислений, по сравнению с алгоритмом по Гуро, так как происходит интерполяция значений векторов нормалей.

Вывод:

В условиях поставленной задачи, рациональнее будет использовать алгоритм простой закраски, так как фигуры сцены состоят из плоскостей, следовательно закраска по Фонгу и Гуро будет скорее мешать: ребра объектов будут сглажены. Поэтому лучше всего использовать простую закраску.

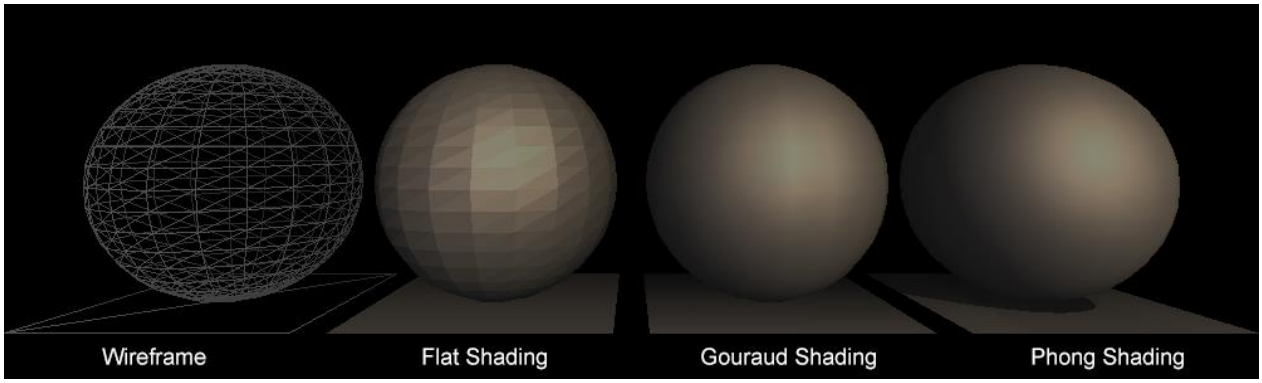

Рисунок 1. Сравнение методов закрашивания

1.4 Анализ алгоритмов моделирования распространения вируса

Система частиц

В системе частиц частицы рассматриваются как материальные точки с разными атрибутами. Сама же система частиц — это совокупность всех частиц. Обычно все частицы в системе меняют скорость или размер по общему закону.

Вывод:

Реалистичное и эффективное представление сцен с распространением вируса - крайне непростая задача из-за огромного количества частичек. Каждая из этих частиц обладает такими свойствами, как: размер, масса, скорость, ускорение. Система частиц – гибкий способ представления вируса, который хорошо подойдет для моей задачи.

2. Конструкторская часть

2.1 Общий алгоритм решения задачи

1. Задать объекты сцены.
2. Используя алгоритм Z буфера получить изображение сцены, сохранить Z буфер для дальнейших расчетов.
3. Запустить вирус в выбранную комнату и выполнять пункты 3.1 и 3.2 до тех пор, пока пользователь не захочет остановить распространение вируса или поменять комнату -> переход в 1.

    3.1 Используя систему частиц, наложить частицы на полученное изображение.

    3.2 Отобразить изображение.

    3.3 Обновить данные о положении частиц.

2.2 Алгоритм Z-буфера

Данный алгоритм является одним из простейших алгоритмов удаления невидимых поверхностей. Впервые он был предложен Кэтмулом. Алгоритм работает в пространстве изображения, а сама идея z-буфера является простым обобщением идеи о буфере кадра, который используется для запоминания атрибутов или интенсивности каждого пиксела в пространстве изображения.

Z-буфер – это отдельный буфер глубины, который используется для запоминания координаты z или глубины каждого видимого пиксела в пространстве изображения. В процессе работы глубина или значение z каждого нового пиксела, который нужно занести в буфер кадра, сравнивается с глубиной того пиксела, который уже занесён в z-буфере.

Если это сравнение показывает, что новый пиксел расположен впереди пиксела, который находится в буфере кадра, то новое значение заносится в буфер и, кроме этого, производится корректировка z-буфера новым значением z. Если при сравнении получается противоположный результат, то никаких действий не

производится.

По сути, алгоритм представляет из себя поиск по x и y наибольшего значения функции z(x, y). Формальное описание алгоритма z-буфера:

1. Заполнить буфер кадра фоновым значением интенсивности или цвета.
2. Провести инициализацию Z-буфера минимальным значением глубины.
3. Преобразовать каждый многоугольник в растровую форму в произвольном порядке.

    3.1 Для всех пикселей, которые связаны с многоугольником, вычислить его глубину z(x, y).

    3.2 Глубину пиксела сравнить со значением, которое хранится в буфере: если $z(x, y) > z_{буф}(x, y)$, то $z_{буф}(x, y) = z(x, y)$, $цвет(x, y) = цвет_{пикселя.}$

4. Отобразить результат.

2.3 Генерация вируса

1. Инициализация начальных данных (направления и скорости ветра).
2. Пока система частиц не пуста или не получена команда прекращения:
    3.1 Обновление положения частиц по заданному закону.
    3.2 Отображение частиц на дисплее.

2.4 Геометрические преобразования

Для осуществления поворота и перемещения камеры используются аффинные преобразования.

Уравнения и матрицы преобразований:

- перемещение точки вдоль координатных осей на dx, dy, dz:

$$\begin{cases} X = x + dx \\ Y = y + dy \\ Z = z + dz \end{cases} \qquad \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- поворот относительно осей x, y, z на угол φ :

ось x: $\begin{cases} X = x \\ Y = y\cos\varphi + z\sin\varphi \\ Z = -y\sin\varphi + z\cos\varphi \end{cases}$ $\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & \sin\varphi & 0 \\ 0 & -\sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

ось y: $\begin{cases} X = x\cos\varphi + z\sin\varphi \\ Y = y \\ Z = -x\sin\varphi + z\cos\varphi \end{cases}$ $\begin{pmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

ось z: $\begin{cases} X = x\cos\varphi + y\sin\varphi \\ Y = -x\sin\varphi + y\cos\varphi \\ Z = z \end{cases}$ $\begin{pmatrix} \cos\varphi & \sin\varphi & 0 & 0 \\ -\sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

2.5 Выбор используемых типов и структур данных

- Объекты сцены – задаются вершинами и гранями.
- Система частиц – хранит в себе частицы, направление движения
- Математические абстракции:
  - Точка – хранит координаты x, y, z.
  - Вектор – хранит направление по x, y, z.
- Интерфейс – используются библиотечные классы для предоставления доступа к интерфейсу.

3. Технологическая часть

3.1 Выбор языка программирования и среды разработки

В качестве языка программирования был выбран Python, так как:

- Я ознакомился с этим языком программирования во время обучения, что сократит время написания программы.
- Данный язык программирования объектно-ориентирован, что даст в полной мере:
    - использовать наследование, абстрактные классы и т.д.
    - представлять трехмерные объекты сцены в виде объектов классов, что позволит легко организовать взаимодействие между ними, положительно влияя на читабельность, не снижая эффективности.

В качестве среды разработки была выбрана «Visual Studio 2021», так как:

- Она бесплатна для студентов.
- Имеет множество удобств, которые облегчают процесс написания и отладки кода.
- Я знаком с данной средой разработки, что сократит время изучения ее возможностей.

Заключение

Во время выполнения поставленной задачи были рассмотрены и проанализированы основные алгоритмы удаления невидимых линий, методы закрашивания. Были проанализированы их достоинства и недостатки, выбраны наиболее подходящие для решения поставленной задачи.

Разработанный программный продукт синтезирует трехмерное изображение при помощи алгоритмов компьютерной графики. Программа реализована таким образом, что пользователь может запускать вирус в выбранную комнату помещения и наблюдать за его распространением.

В ходе выполнения поставленной задачи были получены знания в области компьютерной графики.

Список использованной литературы

1. Вишнякова Д. Ю., Надолинский Н. А. Программная реализациятрехмерных сцен // Известия ЮФУ. Технические науки. 2001. №4.

2. В. П. Иванов, А. С. Батраков. Трёхмерная компьютерная графика / Подред. Г. М. Полищука. — М.: Радио и связь, 1995. — 224 с.

3. Роджерс Д. Алгоритмические основы машинной графики. – М., «Мир», 1989

4. Дж. Ли, Б. Уэр. Трёхмерная графика и анимация. — 2-е изд. —М.: Вильямс, 2002. — 640 с.

5. Mark de Berg. Computational Geometry: Algorithms and Applications. —Springer Science & Business Media, 2008. — P. 259.