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 Ковель Александр Денисович
фамилия, имя, отчество

Группа ИУ7-56Б

Тип практики технологическая

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

Студент _____ Ковель А. Д.
подпись, дата *фамилия, и.о.*

Руководитель практики _____ Куров А. В.
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 (кафедра ИУ7)

Студент Ковель Александр Денисович ИУ7-56Б

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

Во время прохождения производственной практики студент должен:

1. Начать разработку программного обеспечения для непрозрачного в цилиндре с жидкостью.
2. Решить вопрос о способе представления объектов, 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 _____/_____/

Студент _____/_____/

Введение

На сегодняшний день компьютерная графика является одним из самых быстрорастущих сегментов в области информационных технологий.

Область ее применения очень широка и не ограничивается художественными эффектами: в отраслях техники, науки, медицины и архитектуры трехмерные графические объекты используются для наглядного отображения разнообразной информации и презентации различных проектов.

Алгоритмы создания реалистичных изображений требуют особого внимания, поскольку они связаны с внушительным объемом вычислений, требуют больших компьютерных ресурсов и затратны по времени. Для создания качественного изображения объекта следует учесть не только оптические законы, но и расположение источника света, фактуры поверхностей.

Основным направлением в развитии компьютерной графики является ускорение вычислений и создание более качественных изображений. Поэтому в данной пояснительной записке наибольшее внимание уделялось качеству полученного изображения.

1. Аналитическая часть

1.1 Описание объектов сцены

Сцена состоит из источника света, цилиндра, жидкости, стержня и плоскости.

Источник света представляет собой материальную точку, пускающую лучи света во все стороны (если источник расположен в бесконечности, то лучи идут параллельно). Источником света в моей программе является вектор.

Цилиндр - это тонкостенный прозрачный объект, в котором располагается два других объекта - жидкость, стержень (прямоугольный параллелепипед).

Жидкость - это объект, который тоже является прозрачным цельным цилиндром.

Стержень - это непрозрачный прямоугольный параллелепипед. Служит для того чтобы отобразить на экране преломление твердого тела в жидкости.

Плоскость – это некая ограничивающая плоскость. Предполагается, что под такой плоскостью не расположено никаких объектов. Располагается на минимальной координате по оси У.

1.2 Анализ и выбор формы задания трехмерных моделей

Для отображения объектов на экране, необходимо выбрать способ задания модели.

Обычно используются три формы задания моделей:

1. Каркасная (проволочная) модель

Самая простая форма задания модели, так как хранится информация только о вершинах и ребрах объекта. Недостатком данной формы представления является то, что модель малоинформативна и не подходит для твердотельных объектов.

2. Поверхностная модель

Один из самых распространенных способов представления объектов в компьютерной графике. Поверхности задаются аналитически или участки поверхности, как поверхность того или иного вида (использовать полигональную аппроксимацию). Недостатком является то, что не ясна сторона с которой находится материал

3. Объемная (твердотельная) модель

В данной форме сторона материала известна, так как задается направление внутренней нормали.

Таким образом, наиболее подходящая модель для объектов заданной сцены подойдет объемная модель, так как каркасная модель не подходит для восприятия формы объектов, а поверхностные модели не подходят, так как важен материал из которого сделаны объекты сцены

1.3 Анализ способа задания поверхностных моделей

После выбора модели, необходимо выбрать лучший способ представления объемной модели.

1. Аналитический способ

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;

2. Полигональной сеткой

Данный способ характеризуется совокупностью вершин, граней и ребер, которые определяют форму многогранного объекта в трехмерной компьютерной графике.

Также существует множество различных способов хранения информации о сетке:

- Список граней. Объект – это множество граней и множество вершин. В каждую грань входят как минимум 3 вершины;
- «Крылатое» представление. Каждая точка ребра указывает на две вершины, две грани и четыре ребра, которые ее касаются;
- Подреберные сетки. То же «крылатое» представление, но информация обхода хранится для половины грани;
- Таблица углов. Таблица, хранящая 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 лишь вершины, которые указывают на другие вершины. Простота представления дает возможность проводить над сеткой множество операций.

Стоит отметить, что одним из решающих факторов в выборе способа задания модели в данном проекте является скорость выполнения преобразований над объектами сцены.

При реализации программного продукта наиболее удобным представлением является аналитический способ, так как все объекты в сцене являются простыми геометрическими фигурами.

1.4 Анализ и выбор алгоритма удаления невидимых ребер и поверхностей

Для выбора алгоритма удаления невидимых ребер, необходимо выделить свойства, которыми должен обладать выбранный алгоритм, чтобы обеспечить оптимальную работу и реалистичное изображение.

Свойства:

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

Алгоритм, использующий Z-буфер

Алгоритм Z буфера решает задачу в пространстве изображений. Сцены могут быть произвольной сложности, а поскольку размеры изображения ограничены размером экрана дисплея, то трудоемкость алгоритма зависит линейно от числа рассматриваемых поверхностей.

В алгоритме используется два буфера: кадра, в котором хранятся атрибуты каждого пикселя, и Z, в котором хранятся информация о координате Z для каждого пикселя.

Изначально 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 $ax + by + cz + d = 0$. При $c = 0$ многоугольник для наблюдателя вырождается в линию.

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

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

Получим: $z' = z - \frac{a}{c}$, так как $x - x' = dx = 1$

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

Положительными моментами в этом алгоритме являются:

- простота реализации;
- оценка трудоемкости линейна.
- элементы не сортируются, что экономит время вычисления

Недостаток:

- сложная реализация прозрачности;
- большой объем требуемой памяти.

Вывод:

Данный алгоритм не подходит для решения поставленной задачи, так как требует большой объем памяти, и плохо работает с прозрачными объектами что не удовлетворяет требованиям. [1]

Алгоритм Робертса

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

Алгоритм выполняется в 3 этапа:

1. Этап подготовки исходных данных. На данном этапе должна быть задана информация о телах. Для каждого тела сцены должна быть сформирована 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:

$$V = \begin{pmatrix} a_1 & a_2 & & a_n \\ b_1 & b_2 & \cdots & b_n \\ c_1 & c_2 & & c_n \\ d_1 & d_2 & & d_n \end{pmatrix}$$

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.

2. Этап удаления рёбер, экранируемых самим телом.

На данном этапе рассматривается вектор взгляда $E = \{0\ 0\ -1\ 0\}$.

Для определения невидимых граней достаточно умножить вектор E на матрицу тела V. Отрицательные компоненты полученного вектора будут соответствовать невидимым граням.

3. Этап удаления невидимых ребер, экранируемых другими телами сцены.

На данном этапе для определения невидимых точек ребра требуется построить луч, соединяющий точку наблюдения с точкой на ребре. Точка будет невидимой, если луч на своём пути встречает в качестве преграды рассматриваемое тело. [3]

Положительным моментом этого алгоритма являются:

- работа в объектном пространстве;
- высокая точность вычисления.

Недостатки:

- рост сложности алгоритма – квадрат числа объектов;
- тела сцены должны быть выпуклыми (усложнение алгоритма, так как нужна будет проверка на выпуклость);
- сложность реализации.

Вывод:

Алгоритм Робертса не подходит для решения поставленной задачи по следующей причине:

Алгоритм художника

Данный алгоритм работает аналогично тому, как художник рисует картину – то есть сначала рисуются дальние объекты, а затем более близкие. Наиболее распространенная реализация алгоритма – сортировка по глубине, которая заключается в том, что произвольное множество граней сортируется по ближнему расстоянию от наблюдателя, а затем отсортированные грани выводятся на экран в порядке от самой дальней до самой ближней. Данный метод работает лучше для построения сцен, в которых отсутствуют пересекающиеся грани. [4]

Положительным моментом данного алгоритма является:

- требование меньшей памяти, чем, например, алгоритм Z-буффера.

Недостатки

- недостаточно высока реалистичность изображения;
- сложность реализации при пересечения граней на сцене.

Вывод:

Данный алгоритм не отвечает главному требованию – реалистичность изображения. Также алгоритм художника отрисовывает все грани (в том числе и невидимые), на что тратится большая часть времени.

Алгоритм Варнока

Алгоритм Варнока является одним из примеров алгоритма, основанного на разбиении картинной плоскости на части, для каждой из которых исходная задача может быть решена достаточно просто.

Поскольку алгоритм Варнока нацелен на обработку картинки, он работает в пространстве изображения. В пространстве изображения рассматривается окно и решается вопрос о том, пусто ли оно, или его содержимое достаточно просто для визуализации. Если это не так, то окно разбивается на фрагменты до тех пор, пока содержимое фрагмента не станет достаточно простым для визуализации или его размер не достигнет требуемого предела разрешения.

Сравнивая область с проекциями всех граней, можно выделить случаи, когда изображение, получающееся в рассматриваемой области, определяется сразу:

- проекция ни одной грани не попадает в область;
- проекция только одной грани содержится в области или пересекает область. В этом случае проекции грани разбивают всю область на две части, одна из которых соответствует этой проекции;
- существует грань, проекция которой полностью накрывает данную область, и эта грань расположена к картинной плоскости ближе, чем все остальные грани, проекции которых

пересекают данную область. В данном случае область соответствует этой грани.

Если ни один из рассмотренных трех случаев не имеет места, то снова разбиваем область на четыре равные части и проверяем выполнение этих условий для каждой из частей. Те части, для которых таким образом не удалось установить видимость, разбиваем снова и т. д.

Преимущества:

- меньшие затраты по времени в случае области, содержащий мало информации.

Недостатки:

- алгоритм работает только в пространстве изображений;
- большие затраты по времени в случае области с высоким информационным содержимым.

Вывод:

Данный алгоритм не отвечает требованию работы как в объектном пространстве, так и в пространстве изображений, а также возможны большие затраты по времени работы.

Алгоритм обратной трассировки лучей

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

Предполагается, что сцена уже преобразована в пространство изображения, а точка, в которой находится наблюдатель, находится в бесконечности на положительной полуоси Z, и поэтому световые лучи параллельны этой же оси. При этом каждый луч проходит через центр пикселя растра до сцены. Траектория каждого луча отслеживается для определения факта пересечения определенных объектов сцены с этими лучами. При этом необходимо проверить пересечение каждого объекта сцены с каждым лучом, а пересечение с Z_{min} представляет видимую поверхность для данного пикселя.

Положительными моментами в этом алгоритме являются:

- высокая реалистичность синтезируемого изображения;
- работа с поверхностями в математической форме;
- вычислительная сложность слабо зависит от сложности сцены.

Недостаток:

- производительность.

Вывод:

Данный алгоритм не отвечает главному требованию – скорости работы, но подходит для реализации прозрачных объектов. [2]

Вывод

Для удаления невидимых линий выбран алгоритм обратной трассировки лучей. Данный алгоритм позволит добиться максимальной реалистичности и даст возможность смоделировать распространение света в пространстве, учитывая законы геометрической оптики. Также этот алгоритм позволяет строить качественные тени с учетом большого числа источников, и в данном алгоритме удобнее всего реализовывать прозрачные и матовые объекты, что позволит получить достаточно реалистичное изображение. Стоит отметить тот факт, что алгоритм

трассировки лучей не требователен к памяти, в отличие, например, от алгоритма Z-буфера.

1.5 Анализ и выбор модели освещения

Физические модели материалов стараются аппроксимировать свойства некоторого реального материала. Такие модели учитываются особенности поверхности материала или же поведение частиц материала.

Эмпирические модели материалов устроены иначе, чем физически обоснованные. Данные модели подразумевают некий набор параметров, которые не имеют физической интерпретации, но которые позволяют с помощью подбора получить нужный вид модели.

Рассмотрим эмпирические модели, а конкретно модель Ламберта и модель Фонга.

Модель Ламберта

Модель Ламберта моделирует идеальное диффузное освещение, то есть свет при попадании на поверхность рассеивается равномерно во все стороны. При такой модели освещения учитывается только ориентация поверхности (N) и направление источника света (L) (рисунок 1).

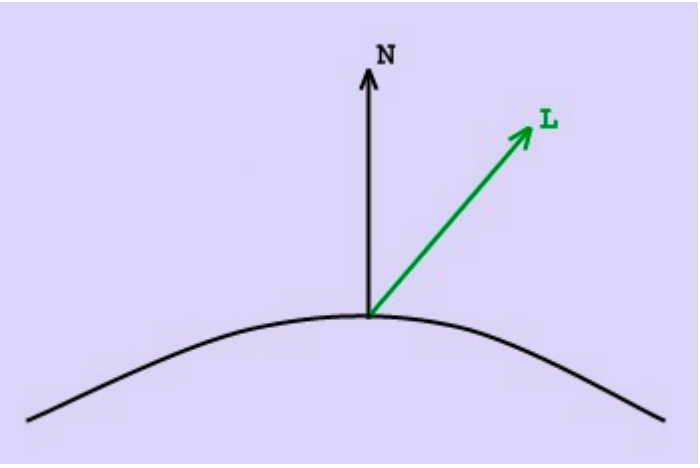


Рисунок 1. Направленность источника света

Эта модель является одной из самых простых моделей освещения и очень часто используется в комбинации с другими моделями. Она может быть очень удобна для анализа свойств других моделей, за счет того, что ее легко выделить из любой модели и анализировать оставшиеся составляющие.

Модель Фонга

Это классическая модель освещения. Модель представляет собой комбинацию диффузной и зеркальной составляющих. Работает модель таким образом, что кроме равномерного освещения на материале могут появляться блики. Местонахождение блика на объекте определяется из закона равенства углов падения и отражения. Чем ближе наблюдатель к углам отражения, тем выше яркость соответствующей точки.

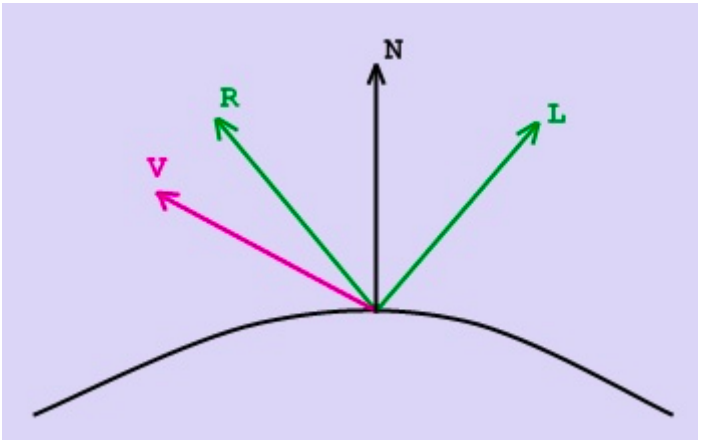


Рисунок 2 Направление источника света, отраженного луча и наблюдателя

Падающий и отраженный лучи лежат в одной плоскости с нормалью к отражающей поверхности в точке падения (рисунок 2). Нормаль делит угол между лучами на две равные части. L – направление источника света, R – направление отраженного луча, V – направление на наблюдателя.

Вывод

Для освещения выбрана модель Фонга, так как изображение должно быть наиболее приближенным к реальности.

2. Конструкторская часть

2.1 Общий алгоритм решения поставленной задачи

1. Задать объекты сцены (цилиндр, жидкость, стержень)
2. Разместить источник света
3. С помощью обратной трассировки лучей визуализировать обстановку.

2.2.1 Алгоритм обратной трассировки лучей

Алгоритмы трассировки лучей на сегодняшний день считаются наиболее приближен к реальности при создании изображений.

Изображение формируется за счет того, что отраженный свет попадает в камеру. Выпускается множество лучей (первичные) из источников света. Часть этих лучей “улетает” в свободное пространство, а часть попадает на объекты сцены, преломляясь и отражаясь, при том часть энергии лучей поглощается в зависимости от физических свойств материала объекта. Преломленные и отраженные лучи продолжают взыскиваться до предела выставленным в программе. В конечном счете часть лучей попадает в камеру и формирует изображение.

Данный алгоритм называется прямой трассировкой лучей, но он крайне неэффективен, так как большинство лучей, исходящих из источника света, не попадают в камеру.

Оптимизированный алгоритм трассировки лучей называется обратный. В данном алгоритме лучи отслеживаются из камеры, а не из

источников света. Таким образом, программно обрабатывается меньшее количество лучей.

Предполагается, что есть камера и экран, находящийся на определенном расстоянии от нее. Экран разбивается на пиксели, далее поочередно испускаются лучи из камеры в центр каждого пиксела. Находится пересечение каждого луча с объектами сцены и выбирается среди всех пересечении наиболее близкое к камере. Далее, применяется модель освещения и получается изображение сцены. Это самый простой метод трассировки лучей, который отсекает невидимые грани.

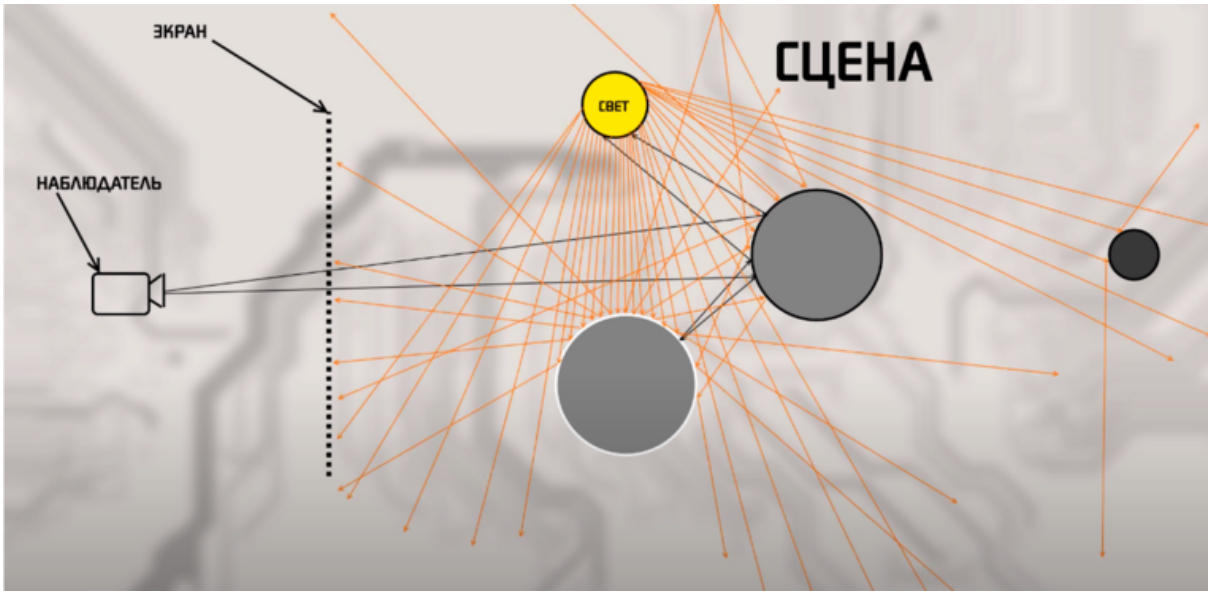


Рисунок 3 Обратная трассировка лучей

В усложненном алгоритме учитывается отражение, для этого необходимо из самого близкого пересечения пустить вторичные лучи.

2.2.2 Алгоритм преломления в прозрачных объектах

Чтобы изобразить прозрачный объект, в его материале должны задаваться отражение и преломление.

Если поверхность обладает отражающими свойствами, то строится вторичный луч отражения. Направление луча определяется по закону отражения (геометрическая оптика): r = i - 2 * n * (n * i)

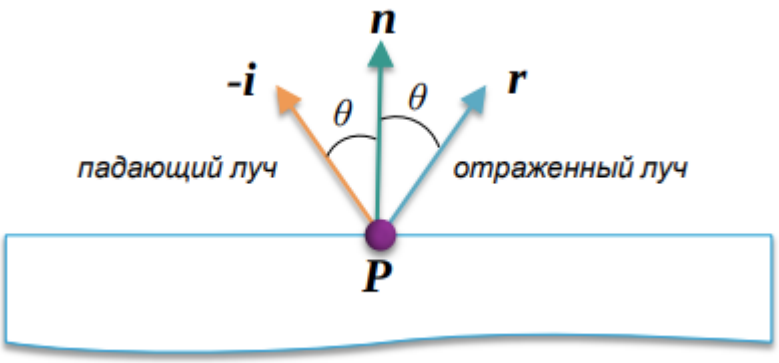
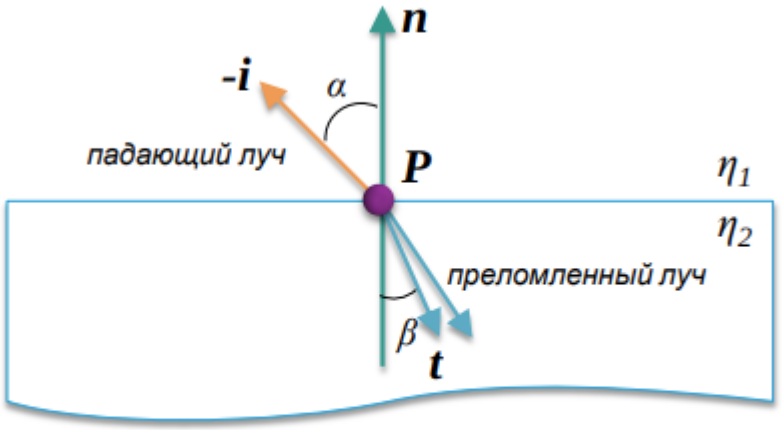


Рисунок 4 Луч по закону отражения

Если же поверхность прозрачна, то строится луч прозрачности. Для определения направления луча используется закон преломления.

$$\sin(\alpha) / \sin(\beta) = \eta_2 / \eta_1$$



$$\boldsymbol{t} = (\eta_1 / \eta_2) \cdot \boldsymbol{i} - [\, \cos(\beta) + (\eta_1 / \eta_2) \cdot (\boldsymbol{n} \cdot \boldsymbol{i}) \,] \cdot \boldsymbol{n},$$
$$\cos(\beta) = \mathrm{sqrt}[\, 1 - (\eta_1 / \eta_2)^2 \cdot (1 - (\boldsymbol{n} \cdot \boldsymbol{i})^2) \,]$$

Рисунок 5 Луч по закону преломления

2.2.3 Модель освещения Фонга

Наиболее распространенная в обратной трассировке лучей эмпирическая модель - освещения по Фонгу.

$$C_{out} = C \cdot [k_a + k_d \cdot \max(0, (\boldsymbol{n} \cdot \boldsymbol{l}))] + I \cdot k_s \cdot \max(0, (\boldsymbol{v} \cdot \boldsymbol{r}))^p,$$
$$\boldsymbol{r} = \text{reflect}(-\boldsymbol{v}, \boldsymbol{n})$$

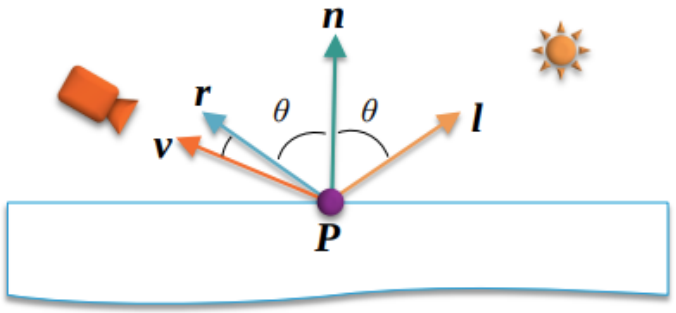


Рисунок 6 Освещение по Фонгу

2.2.4 Пересечения луча и сферы

Уравнение луча: P = O + t(V − O), t ≥ 0, где O - центр камеры, а V - текущий пиксель. Направление луча: D = V − O

Первое уравнение по координатам

$$x(t) = xO + txD$$

$$y(t) = yO + tyD$$

$$z(t) = zO + tzD$$

Сфера — это множество точек P, лежащих на постоянном расстоянии r от фиксированной точки C. Тогда можно записать уравнение, удовлетворяющее этому условию: $distance(P, C) = r$

Запишем расстояние между P и C как длину вектора из P в C. $|P - C| = r$

Заменим на скалярное произведение вектора на себя: $\sqrt{(P - C)(P - C)} = r$

Избавимся от корня: $(P - C) * (P - C) = r^2$

В итоге есть два уравнения - уравнение луча и сферы. Найдем пересечение луча со сферой. Для этого подставим первое уравнение в последнее $(O + t * \overline{D} + C) * (O + t * \overline{D} - C) = r^2$

Разложим скалярное произведение и преобразуем его. В результате получим: $t^2 * (\overline{D} * \overline{D}) + 2T(\overline{OD} * \overline{D}) + (\overline{OC} * \overline{OC}) - r^2 = 0$

Представленное квадратное уравнение имеет несколько возможных случаев решения. Если у уравнения одно решение, это означает, что луч касается сферы. Два решения обозначают, то что луч входит в сферу и выходит из нее. И если нет решений, значит, луч не пересекается со сферой.

2.2.5 Пересечения луча и цилиндра

Бесконечный цилиндр, образующие которого параллельны оси y, описывается уравнением: $x^2 + z^2 = r^2$

Так как в программе требуется изобразить вертикальный цилиндр, то рассматривается уравнение вдоль оси y. Используя уравнение направления луча, разложенного по координатам, получим: $(x_o + tx_D)^2 + (z_0 + tz_D)^2 = r^2$

Выносим t из скобок и получаем уравнение:

$$t^2(x^2_D + z^2_D) + 2t(x_o x_D + z_o z_D) + (x^2_O + z^2_O - r^2) = 0$$

Решение представленного уравнения можно получить, решив дискриминант. Уравнение ограниченно размеры цилиндра по верхней и нижней координате цилиндра.

Если оба решения проходят проверку, то наименьшее неотрицательное значение t - это ближайшая точка пересечения луча с конечным цилиндром.

2.3 Выбор используемых типов и структур данных

В данной работе нужно будет реализовать следующие типы и структуры данных.

1. Трехмерный вектор - хранит направление, задается координатами x, y, z
2. Цвет - вектор из трех чисел (синий, красный, зеленый)
3. Сфера - хранит радиус, цвет, положение, коэффициент преломления, коэффициент отражения, эмиссию.
4. Цилиндр - хранит радиус, высоту, цвет, положение, коэффициент преломления, коэффициент отражения, эмиссию.
5. Поверхность - хранит высоту, цвет, положение, коэффициент преломления, коэффициент отражения, эмиссию.
6. Прямоугольный параллелепипед - хранит цвет, положение, положение второй точки, коэффициент преломления, коэффициент отражения, эмиссию.
7. Сцена - список объектов, камеру, источник света
8. Источник света - положение и направление света
9. Камера - положение

2.4 Вывод

В данном разделе были описаны требования к программе, подробно рассмотрен алгоритм трассировки лучей, описаны типы и структуры данных, которые будут реализованы.

3. Технологическая часть

3.1 Выбор языка программирования и среды разработки

Для решения описанной задачи был выбран язык программирования Kotlin [7]. Данный язык был выбран по следующим причинам.

1. Он использует объектно-ориентированный подход к программированию
2. В языке присутствует множество синтаксических возможностей, которые помогают использовать готовые конструкции, вместо того, чтобы переписывать однотипный код
3. В нем имеется большое количество библиотек и шаблонов, позволяющих не затрачивать время на готовые решения.
4. Строгая типизация, что позволяет от контролированных ошибок
5. Он является нативным, что необходимо для уменьшения времени работы алгоритмов.

В качестве среды разработки был использован Intellij Idea [8],

Intellij Idea среда созданная разработчиками языка Kotlin, поэтому полностью раскрывает потенциал языка.

3.2 Структура программы

Данная программа состоит из следующих модулей

- Математические абстракции
    - Line - класс, который задается двумя трехмерными точками

    - Ray - трехмерный луч, задающийся точкой начала луча, направляющим вектором
    - RayHit - вектор, задающийся лучом, твердым телом и вектором
    - Vector3 - трехмерный вектор, задается тремя координатами
- Классы работающие с отдельными пикселями
    - Color - класс цвета, задается тремя параметрами
    - PixelData - класс представляющий пиксел экрана, задается цветом и двумя координатами
    - PixelBuffer - класс представляет изображение, задается шириной и высотой экрана
- Классы сцены
    - Camera - класс имитирующий положение в пространстве, задается вектором, углами поворота и полем зрения
    - Light - класс источник света, задается вектором, определяющим положение
    - Renderer - класс для создание изображения
    - Scene - класс, состоящий из наборов объектов и их свойств.
    - Skybox - класс для отображения изображения на верхней части сцены
- Объекты
    - Solid - абстрактный класс объекта сцены
    - Sphere - хранит радиус, цвет, положение, коэффициент преломления, коэффициент отражения, эмиссию.
    - Cylinders - хранит радиус, высоту, цвет, положение, коэффициент преломления, коэффициент отражения, эмиссию.
    - Plane - хранит высоту, цвет, положение, коэффициент преломления, коэффициент отражения, эмиссию.
    - Box - хранит цвет, положение, положение второй точки, коэффициент преломления, коэффициент отражения, эмиссию.

- Интерфейс
    - MainView - окно для отображения полученного изображения
    - SettingsView - диалоговое окно для задания характеристик сцены
    - Styles - класс управление стилем текста окон
    - MyApp - файл запуска приложения

3.3 Интерфейс программы

Интерфейс программы состоит из двух окон. Основное окно и окна настроек.

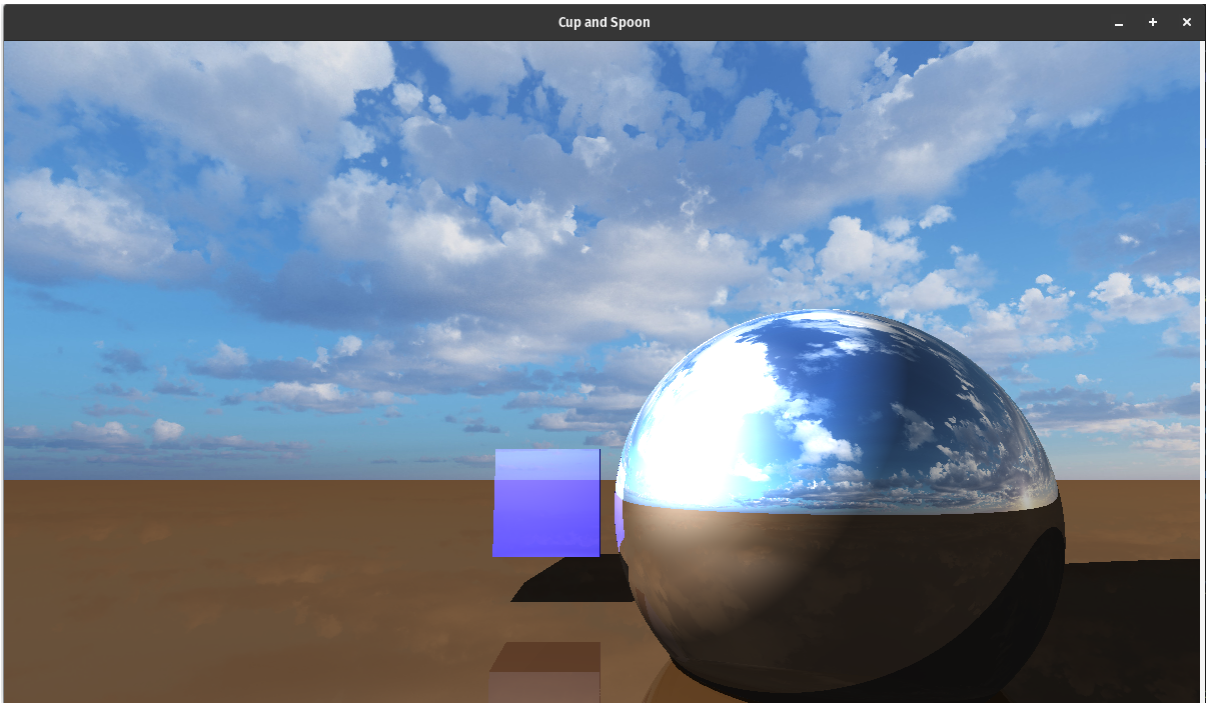


Рисунок 7 Основное окно

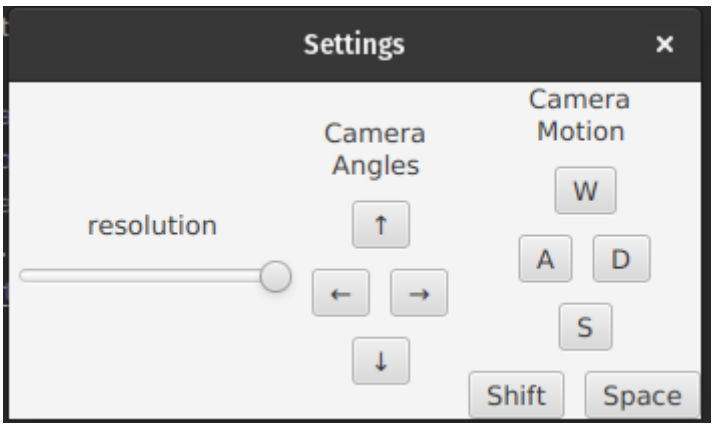


Рисунок 8 Окно настроек

Функции представленных кнопок следующее:

- регулятор “разрешение” - изменяет разрешение изображения, которое необходимо вычислить
- кнопки стрелки “углы камеры” - изменяют угол наклона на 20 градусов
- кнопки “WASD Shift Space” “положение камеры” - изменяют положение камеры на 1 относительную координату

3.4 Вывод

В этом разделе был выбран язык программирования и среда разработки, рассмотрены основные классы, разобран интерфейс приложения.

Заключение

Во время выполнения поставленной задачи были проанализированы основные способы представления и задания трехмерных моделей, а также рассмотрены и основные алгоритмы удаления невидимых линий и методы освещения. Проанализированы достоинства и недостатки представленных алгоритмов и выбраны наиболее оптимальные для решения поставленной задачи.

Проделанная работа помогла закрепить полученные навыки в области компьютерной графики и проектирования программного обеспечения. Реализация программы позволяет рассматривать сцену с разных сторон.

Стоит отметить, что при выполнении данной работы удалось изучить язык Kotlin, а также познакомиться и изучить возможности среды разработки Intellij Idea.

Литература

[1] Алгоритм Z-буфера. [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 29.08.2022).

[2] Трассировка лучей из книги Джефа Проузиса [Электронный ресурс]. Режим доступа: https://www.graphicon.ru/oldgr/courses/cg99/notes/lect12/prouzis/raytrace.htm (дата обращения: 11.08.2022).

[3] Алгоритм Робертса. [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 10.08.2022).

[4] Алгоритм, использующие список приоритетов (алгоритм художника). [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 14.08.2022).

[5] Алгоритм Варнока. [Электронный ресурс]. Режим доступа: http://compgraph.tpu.ru/Oglavlenie.htm (дата обращения: 08.08.2022).

[6] Модели освещения. [Электронный ресурс]. Режим доступа: https://devburn.ru/2015/09/ (дата обращения: 10.08.2022).

[7] Язык программирования Kotlin [Электронный ресурс]. Режим доступа: https://kotlinlang.ru (дта обращения: 20.08.2022).

[7] Среда разработки Intellij Idea [Электронный ресурс]. Режим доступа: https://www.jetbrains.com/idea/ (дта обращения: 20.08.2022).