Индивидуальное задание.

Разработать программу, которая визуализирует флюгер, реагирующий на направленные потоки ветра. Конструкция должна состоять из плоскости основания, каркаса флюгера (из геометрических примитивов: цилиндр, прямоугольный параллелепипед, сфера, треугольная пирамида) и четырех сфер. Предоставить пользователю возможность выбирать цвет и отражающие свойства материала, из которого изготовлена каждая из перечисленных частей, а также направление ветра. В сцене должны быть реализованы управляемые: фоновое освещение, один направленный и несколько точечных источников освещения. Предусмотреть возможность изменения положения камеры. Флюгер должен вращаться.

Оглавление

[Введение 4](#_Toc77193462)

[1. Аналитическая часть 5](#_Toc77193463)

[1.1. Постановка задачи 5](#_Toc77193464)

[1.2. Формализация объектов синтезируемой сцены 5](#_Toc77193465)

[1.3. Анализ алгоритмов удаления невидимых линий и поверхностей 6](#_Toc77193466)

[1.3.1. Критерии выбора алгоритма 6](#_Toc77193467)

[1.3.2. Алгоритм Робертса 7](#_Toc77193468)

[1.3.3. Алгоритм Варнока 7](#_Toc77193469)

[1.3.4. Алгоритм, использующий Z-буфер. 8](#_Toc77193470)

[1.3.5. Алгоритм трассировки лучей. 9](#_Toc77193471)

[1.4. Анализ моделей освещения 12](#_Toc77193472)

[1.4.1. Критерии выбора модели 12](#_Toc77193473)

[1.4.2. Виды моделей освещения 12](#_Toc77193474)

[1.4.3. Модель Ламберта 13](#_Toc77193475)

[1.4.4. Модель Фонга 13](#_Toc77193476)

[1.4.5. Модель Блинна-Фонга 14](#_Toc77193477)

[1.5. Преобразования. 15](#_Toc77193478)

[1.5.1. Перенос 16](#_Toc77193479)

[1.5.2. Масштабирование 16](#_Toc77193480)

[1.5.3. Поворот 16](#_Toc77193481)

[Вывод 17](#_Toc77193482)

[2. Конструкторская часть 18](#_Toc77193483)

[2.1. Требования к программе 18](#_Toc77193484)

[2.2. Алгоритм обратной трассировки лучей 18](#_Toc77193485)

[2.2.1. Пересечение луча со сферой 20](#_Toc77193486)

[2.2.2. Пересечение луча с плоскостью 21](#_Toc77193487)

[2.2.3. Пересечение луча с цилиндром 22](#_Toc77193488)

[2.2.4. Пересечение луча с прямоугольным параллелепипедом. 23](#_Toc77193489)

[2.2.5. Пересечение луча с треугольной пирамидой. 24](#_Toc77193490)

[2.2.6. Нахождение отраженного луча 25](#_Toc77193491)

[2.2.7. Уменьшение времени работы алгоритма. 27](#_Toc77193492)

[2.3. Модель Фонга 27](#_Toc77193493)

[2.4. Выбор используемых типов и структур данных 28](#_Toc77193494)

[Вывод 30](#_Toc77193495)

[Список использованной литературы 31](#_Toc77193496)

# Введение

Физические тела, окружающие нас, обладают различными оптическими свойствами. Некоторые из них, например, отражают и пропускают световые лучи, отбрасывают тени. Эти и другие свойства нужно уметь наглядно показывать при помощи электронно-вычислительных машин. Этим и занимается компьютерная графика.

Компьютерная графика – это совокупность методов и способов преобразования информации в графическую форму и из графической формы в ЭВМ. Область применения компьютерной графики очень широка: от воссоздания художественных эффектов в компьютерных играх до построения трехмерных объектов в отраслях техники и бизнеса для моделирования создаваемого продукта.

При реализации подобных задач необходимо уметь создавать реалистичные изображения, которые будут учитывать оптические свойства объектов. Для этого в компьютерной графике существует множество алгоритмов. Основная проблема заключается в том, что для получения качественного изображения требуется большое количество времени и памяти.

Целью данного курсового проекта является разработка ПО, которое предоставляет трехмерную визуализацию поведения флюгера.

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить и проанализировать существующие алгоритмы построения реалистичных изображений.
2. Выбрать алгоритмы, наиболее подходящие для решения поставленной задачи.
3. Спроектировать архитектуру будущего программного продукта и выбрать структуры данных для представления объектов синтезируемой сцены.
4. Разработать программу на основе выбранных алгоритмов и структур данных.
5. Провести исследования на основе разработанной программы.

## Аналитическая часть

В данном разделе приводятся постановка задачи и формализация объектов синтезируемой сцены, формулируются критерии выбора алгоритма и моделей для реализации поставленной задачи, анализируются существующие алгоритмы и модели, из которых затем выбираются наиболее подходящие.

## Постановка задачи

Необходимо разработать программный продукт, который представит трехмерную визуализацию поведения флюгера, реагирующего на направленные потоки ветра. Сцена должна быть реалистичной и учитывать такие явления, как отражение и отбрасывание теней. Пользователь должен иметь возможность управления направлением вращения флюгера, изменения параметров материала его частей, управления источниками освещения и камерой.

## Формализация объектов синтезируемой сцены

Сцена должна состоять из следующих объектов:

1. Направленный источник света. Он описывается вектором направления света и интенсивностью. Предполагается, что источник расположен в бесконечности.
2. Точечный источников света. Он описывается фиксированной точкой в пространстве (позицией) и интенсивностью. Предполагается, что свет от такого источника распространяется равномерно во всех направлениях.
3. Плоскость основания флюгера.
4. Флюгер, состоящий из таких геометрических примитивов, как цилиндр, прямоугольный параллелепипед, сфера, треугольная пирамида.

В компьютерной графике в основном используются 3 вида моделей трехмерных объектов:

1. Каркасная (проволочная) модель. Это простейший вид моделей, содержащий минимум информации - о вершинах и рёбрах объектов. Главный недостаток - такая модель не всегда правильно передает представление об объекте (например, если в объекте есть отверстия).
2. Поверхностная модель. Отдельные участки задаются как участки поверхности того или иного вида (участок сферической поверхности, участок конической поверхности и т. д.). Решает проблему каркасной модели, но все еще имеет недостаток - нет информации о том, с какой стороны поверхности находится собственный материал
3. Объемная модель. В отличии от поверхностной, содержит указание расположения материала (чаще всего указанием направления внутренней нормали).

Важнейшие требования к модели – правильность отображения информации об объекте и компактность. В рамках поставленной задачи каркасная модель не удовлетворяет первому критерию, а информация о том, где расположен материал, не является необходимой. Поэтому при выборе между оставшимися поверхностной и объемной моделями, в соответствии с требованием к компактности, предпочтительней поверхностная модель.

Поверхности сферы и цилиндра удобно описывать аналитически. Для представления пирамиды используют полигональную аппроксимацию треугольниками. Параллелепипед можно однозначно задать координатами всего двух его вершин: той, у которой все три координаты минимальны, и той, у которой все три координаты максимальны.

## Анализ алгоритмов удаления невидимых линий и поверхностей

Задача удаления невидимых линий и поверхностей является одной из наиболее сложных в компьютерной графике. Алгоритмы удаления невидимых линий и поверхностей служат для определения линии ребер, поверхностей или объемов, которые видимы или невидимы для наблюдателя, находящегося в заданной точке пространства

В зависимости от того, в каком пространстве решается задача, алгоритмы делят на следующие группы:

1. Алгоритмы, работающие в объектном пространстве (мировая системе координат). Обобщенный подход, основанный на анализе пространства объектов, предполагает попарное сравнение положения всех объектов по отношению к наблюдателю. Такие алгоритмы предоставляют высокую точность, но требуют много ресурсов.
2. Алгоритмы, работающие в пространстве изображений (в системе координат, связанной с тем устройством в котором отображается результат). Точность таких алгоритмов ниже.
3. Иногда отдельно выделяют алгоритмы, работающие в обоих пространствах попеременно. [1]

Для выбора наиболее подходящего для решаемой задачи алгоритма, необходимо сформулировать критерии выбора и осуществить краткий обзор существующих алгоритмов.

### Критерии выбора алгоритма

Основными критериями выбора алгоритма служат возможность получения реалистичного изображения высокого качества, которое учитывало бы эффекты отражения и отбрасывания теней.

### Алгоритм Робертса

Алгоритм Робертса является первым известным решением задачи об удалении невидимых линий. Данный алгоритм работает в объектном пространстве и только с выпуклыми телами. Если тело изначально является не выпуклым, то его нужно предварительно разбить на выпуклые составляющие. Основные этапы:

1. (Подготовительный) Формирование исходных данных.
2. Удаление линий, экранируемых самим телом.
3. Удаление линий, экранируемых другими телами.
4. Удаление линий пересечения тел, экранируемых самими телами, связанными отношением протыкания, и другими телами.

Преимущество алгоритма Робертса в том, что он целиком основам на математических предпосылках, которые просты, точны и мощны [2].

Один из недостатков этого алгоритма - большая, по сравнению с алгоритмами, работающими в пространстве изображения, трудоемкость, которая пропорциональна квадрату количества объектов на сцене. Но недостатком, который делает алгоритм Робертса неприменимым для решения данной задачи, является тот факт, что без модификации и привлечения сторонних методов данный алгоритм не позволяет реализовать тень и зеркальные эффекты.

### Алгоритм Варнока

Алгоритм Варнока основывается на рекурсивном разбиении экрана и решает задачу в пространстве изображений. Единой версии этого алгоритма не существует, но можно выделить простейшую и более сложные его версии.

Основная идея - на каждом шаге найти ответ на вопрос о том, что изображать в очередном окне. Сначала окно имеет размеры экрана. Если нельзя точно дать ответ, то окно делится на части. Вместо одного окна получаем окна меньших размеров. Если снова не можем дать ответ, то продолжаем делить каждое окно на части, пока не сможем ответить или окно не дойдёт до размеров в один пиксель

В простейшей версии алгоритма окно делится на подокна всякий раз, если это окно не пусто. Пределом деления является получение окна размером в 1 пиксель. Для одной точки легко определить ближайший к наблюдателю многоугольник - для этого нужно найти глубину каждого многоугольника в этой точки.

В более сложных версиях делается попытка решения задачи для окон большего размера (больше одного пикселя). Для этого нужно провести классификацию многоугольников, рассматриваемых в алгоритме Варнока. По отношению к ячейке, алгоритм классифицирует полигон как:

* внешний – расположенный за пределами ячейки (см. рис. 1.1a)
* внутренний – расположенный полностью внутри ячейки (см. рис. 1.1b)
* пересекающий - края полигона пересекают ячейку (см. рис. 1.1c)
* охватывающий – ячейка расположена полностью внутри многоугольника (см. рис. 1.1d)

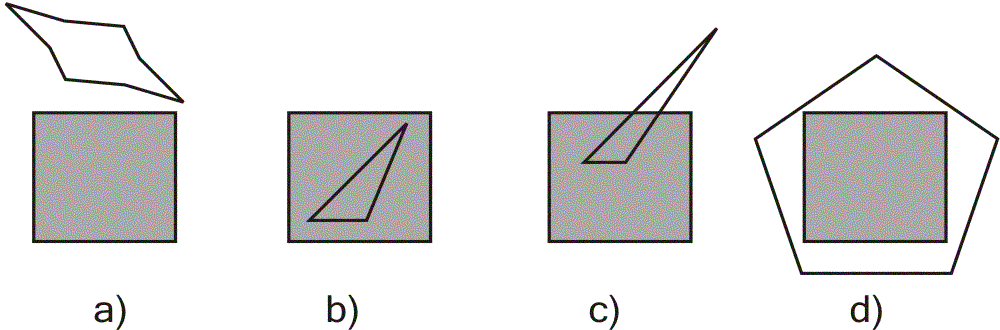


Рисунок 1.1 - Классификация многоугольников в алгоритме Варнока

Затем необходимо определить действия, которые нужно предпринять в том или ином случае взаимного расположения ячейки и полигонов.

Достоинством данного алгоритма является простота реализации и высокая эффективность в случае, если размеры перекрываемых областей невелики.

Одним недостатком алгоритма Варнока является то, что при визуализации сложной сцены число разбиений может стать очень большим, что приведет к сильному увеличению времени работы данного алгоритма. И снова приходится признать алгоритм неприменимым к решению поставленной задачи из-за второго недостатка: он не моделирует оптические свойства объекта.

### Алгоритм, использующий Z-буфер.

Алгоритм Z буфера решает задачу в пространстве изображений и является одним из самых простых и широко используемых. Его идея заключается в использовании двух буферов: буфера кадра и буфера глубины, также называемого Z-буфером. Буфер кадра используется для хранения интенсивности каждого пикселя в пространстве изображения. В буфере глубины запоминается значение координаты Z (глубины) каждого видимого пикселя в пространстве изображения.

В ходе работы алгоритма значение глубины каждого нового пикселя, заносимого в буфер кадра, сравнивается с глубиной того пикселя, который уже занесен в Z-буфер. Если это сравнение показывает, что новый пиксель расположен ближе к наблюдателю, чем пиксель, уже находящийся в буфере кадра, то новый пиксель заносится в буфер кадра и производится корректировка Z-буфера: в него заносится глубина нового пикселя. Если же значение глубины нового пикселя меньше, чем хранящееся в буфере, то осуществляется переход к следующей точке. [1]

Основными достоинствами данного алгоритма являются простота его реализации, допустимость сцен любой сложности, корректная обработка случаев взаимных пересечений объектов, линейная зависимость трудоемкости от числа объектов на сцене, а также отсутствие необходимости предварительной сортировки объектов по глубине.

К недостаткам алгоритма Z-буфера относят необходимость выделения памяти под два буфера, каждый из которых имеет размер равный количеству пикселей на экране, что, впрочем, при современном развитии технологий уже не так болезненно. Также алгоритм не учитывает тени, эффекты прозрачности и зеркальности. И если проблему теней можно исправить, применяя различные модификации алгоритма z-буфера, то с эффектом зеркального отражения он не справляется, что делает и этот алгоритм неприменимым к решению поставленной задачи.

### Алгоритм трассировки лучей.

Существуют два подхода к трассировке лучей: прямой и обратный.

Основная идея алгоритма прямой трассировки лучей состоит в том, что наблюдатель видит объекты благодаря световым лучам, которые испускает некоторый источник и которые падают на объект, отражаются, преломляются или проходят через него и в результате достигают зрителя [3]. Если проследить за лучами, то становится понятно, что среди них лишь малая часть дойдет до наблюдателя, что приведет к большим затратам вычислительных мощностей (см. рис. 1.2).

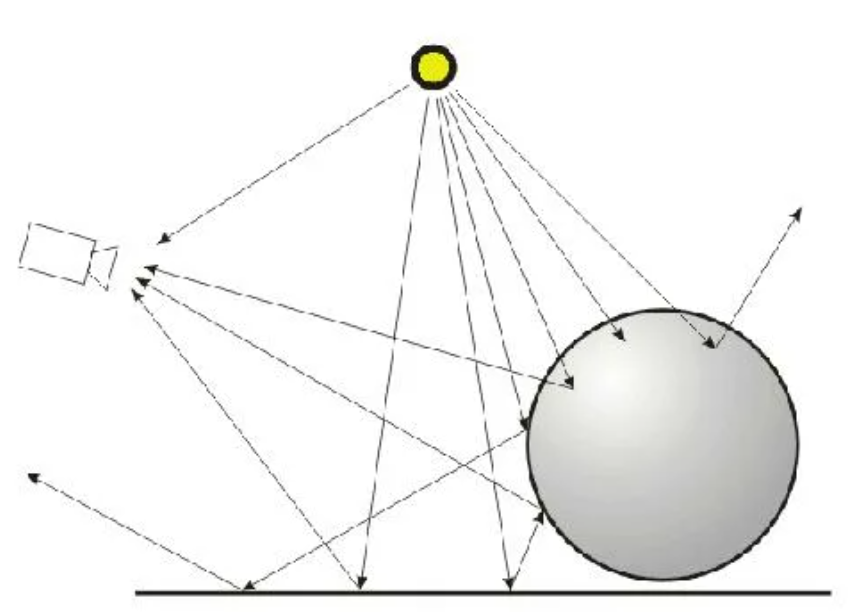


Рисунок 1.2 – Прямая трассировка лучей

Заменой данному алгоритму служит метод обратной трассировки лучей, который отслеживает лучи в обратном направлении (от наблюдателя к объекту). Обратная трассировка более эффективна, так как гарантирует, что число лучей, достигших глаза, будет точно таким же, как число пикселов изображения.

Считается, что наблюдатель расположен на положительной полуоси z в бесконечности, поэтому все световые лучи параллельны оси z. В ходе работы испускаются лучи от наблюдателя и ищутся пересечения луча и всех объектов сцены [3].

Каждый раз, когда луч пересекает некоторую поверхность в некоторой точке, из этой точки испускаются дополнительные лучи. Если поверхность отражающая, то генерируется отраженный луч. Если поверхность пропускает свет, то генерируется пропущенный луч. Пути этих лучей отслеживаются по всей модели, и если лучи пересекают другие поверхности, то снова испускаются лучи. В каждой точке, где луч пересекает поверхность, рисуется луч тени из точки пересечения к каждому источнику света. Если этот луч пересекает другую поверхность перед тем, как достигнуть источника света, то на ту поверхность, с которой был послан луч, падает тень с поверхности, блокирующей свет (см. рис. 1.3). Математически, все эти лучи вместе с данными о физических характеристиках объектов модели (цвет, прозрачность, зеркальность и т.д.) позволяют определить цвет и его интенсивность для каждой точки изображения.

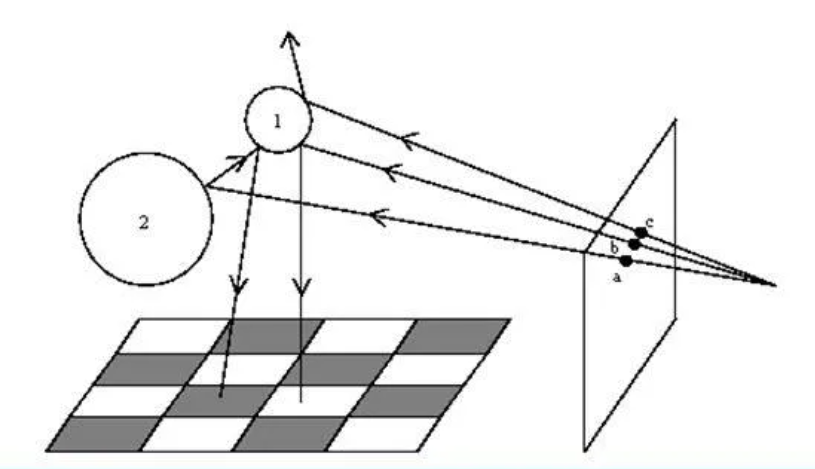


Рисунок 1.3 – Обратная трассировка лучей

Так как необходимо множество раз определять пересечение лучей и объектов сцены, эффективность этой процедуры оказывает самое большое влияние на эффективность всего алгоритма. Чтобы избавиться от ненужного поиска пересечений, ищут пересечение луча с объемной оболочкой рассматриваемой фигуры – более простым объектом, внутрь которого можно поместить рассматриваемый. Обычно используют параллелепипед или сферу.

На рис. 1.4 приведены примеры сферической и прямоугольной оболочки.

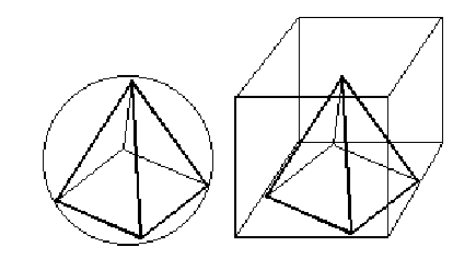


Рисунок 1.4 — Сферическая и прямоугольная оболочки

Тогда при анализе пересечения луча и объемной оболочки рассматриваемого объекта, если такого пересечения нет, можно сделать вывод, что пересечения луча и самого рассматриваемого объекта нет. Если же найдено пересечение, то, возможно, есть пересечение луча и рассматриваемого объекта.

К достоинствам данного алгоритма можно отнести предоставление возможности визуализировать оптические эффекты: тени, прозрачность, преломление, отражение. Полученное изображение получается очень реалистичным. Алгоритм может работать с любыми объектами.

Серьёзным недостатком алгоритма трассировки является производительность: необходимо создавать большое число лучей, проходящих через сцену, которые могут раздваиваться на отраженный и преломленный лучи, для которых все вычисления повторяются. Однако этот негативный аспект можно решить путем распараллеливания: поскольку каждый луч, исходящий из камеры, независим от всех остальных, лучи можно трассировать одновременно.

## Анализ моделей освещения

Модели освещения используются для вычисления интенсивности света для данной точки на поверхности модели. Существует множество видов моделей освещения, каждый из которых учитывает различное число параметров.

### Критерии выбора модели

Так как в данной задаче необходимо построить реалистичное изображение, то модель освещения должна учитывать, как минимум, диффузную и отражающую составляющую отражения, а также рассеянное освещение. При этом она не должна быть перегружена другими параметрами, чтобы лишний раз не увеличивать скорость работы программы.

### Виды моделей освещения

Модели освещения можно разделить на локальные и глобальные.

В локальной модели учитывается только свет от источников (первичные лучи) и ориентация поверхности. В глобальной модели учитывается еще и свет, отраженный от других поверхностей или пропущенный через них (вторичные источники). Такие взаимоотношения сложно учитывать

Для решения поставленной задачи нет необходимости учитывать преломленные лучи, поэтому, чтобы лишний раз не усложнять реализацию, можно ограничиться использованием локальной модели освещения.

Также модели освещения можно разделить на физически обоснованные и эмпирические модели освещения.

Физически обоснованные модели стараются аппроксимировать свойства некоторого реального материала. Такие модели учитывают особенности поверхности материала, например, слои материала или же поведение частиц материала.

Эмпирические модели подразумевают некий набор параметров, не имеющих физической интерпретации, но позволяющих с помощью подгона получить нужный вид конечной модели. Иногда такие модели дают более качественный результат за счет большего контроля за выразительностью, чем за точностью.

Как правило, в компьютерной графике используют эмпирические модели освещения, так как основной целью является построение реалистичного изображения, а не точное моделирование физического мира

Примеры эмпирических локальных моделей - модель Ламберта и модель Фонга.

### Модель Ламберта

Модель освещения Ламберта является базовой для большинства остальных моделей. Она моделирует идеальное диффузное освещение. Считается, что свет падающий в точку, одинаково рассеивается по всем направлением полупространства (см. рис. 1.5).

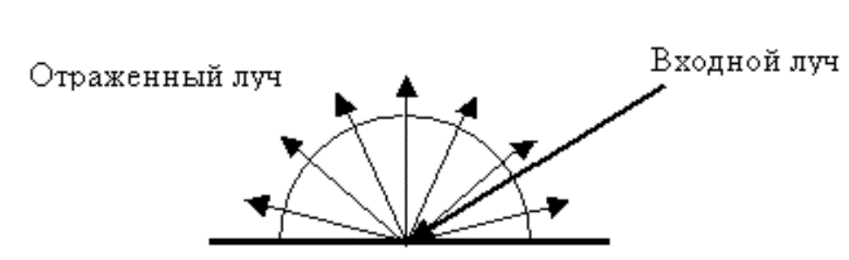


Рисунок 1.5 — Рассеивание света по модели Ламберта.

Интенсивность в точке зависит только от угла α между вектором падения света L и вектором нормали N в этой точке, как показано на рисунке 1.6. Максимальная сила света будет при перпендикулярном падении света на поверхность и будет убывать с увеличением угла α.



Рис**унок 1.6 —** **Вычисление интенсивности в модели Ламберта.**

Модель Ламберта очень часто используется в комбинации других моделей, так как практически в любой другой модели освещения можно выделить диффузную составляющую.

Но Ламбертова (идеально диффузная) поверхность выглядит одинаково яркой со всех направлений, то есть не позволяет передавать блики на телах сцены. Поэтому она не подходит для решения поставленной задачи.

### Модель Фонга

Модель Фонга добавляет в модель Ламберта зеркальное отражение, что позволяет моделировать блики. Предполагается, что освещенность каждой точки тела можно разделить на 3 компоненты (см. рис. 1.7):



Рисунок 1.7. Составляющие модели освещения Фонга.

1. фоновое освещение (ambient). Присутствует в любом уголке сцены и не зависит от источников света, поэтому для упрощения расчетов оно задается константой

2. рассеянный свет (diffuse). Рассчитывается по модели Ламберта

3. блики (specular). Зависит от того, насколько близки направления вектора V, направленного на наблюдателя, и отраженного луча R (см. рис. 1.8).

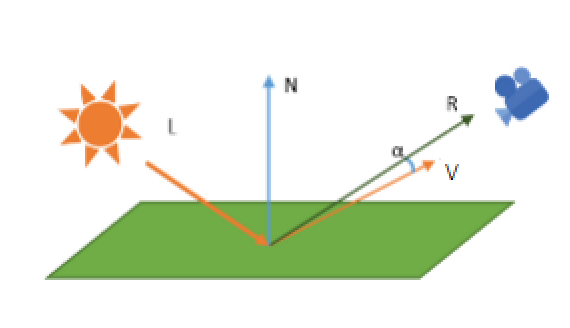


Рисунок 1.8. Модель освещения Фонга.

Итоговая формула интенсивности в точке имеет вид:

, (1.1)

где – коэффициенты фонового, диффузного и зеркального освещения соответственно, – вектор нормали к поверхности в точке,  – падающий луч (направление на источник света), – отраженный луч, – вектор, направленный к наблюдателю, – степень, аппроксимирующая пространственное распределение зеркально отраженного света и все векторы являются единичными.

К недостаткам модели Фонга можно отнести тот факт, что она не является физически корректной и не удовлетворяет закону сохранения энергии Однако данная модель освещения подходит для решения поставленной задачи, так как включает все необходимые компоненты - диффузную и отражающую составляющую отражения, а также рассеянное освещение, - и является достаточно простой.

### Модель Блинна-Фонга

Модель Блинна-Фонга –это унификация модели Фонга. Главное отличие данной модели от предыдущей заключается в том, что вычисление косинуса между направлением на наблюдателя v и направлением отраженного луча r заменяется на вычисление косинуса угла между нормалью к поверхности n и вектором, средним между направлением на наблюдателя и направлением на источник света h (см. рис. 1.9). Данная аппроксимация позволяет в ряде случае упростить вычисления.

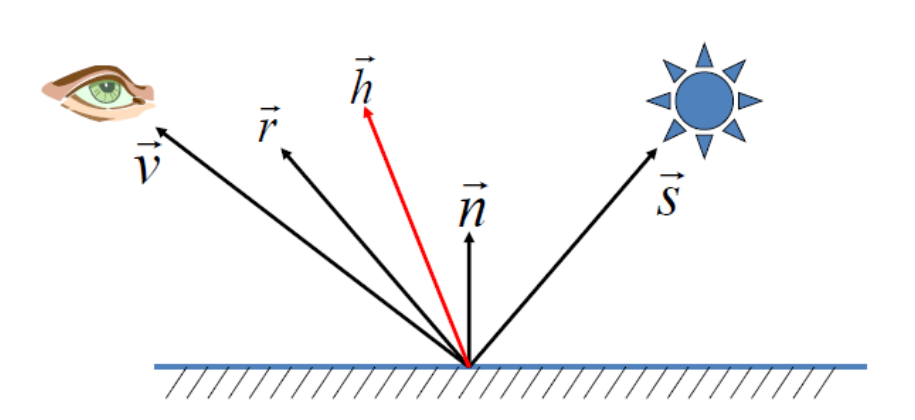


Рисунок 1.9. Модель освещения Блинна-Фонга.

Однако такое упрощение изменяет результат, что показано на рис. 1.10: слева – результат применения модели Блинна-Фонга, справа – модели Фонга.

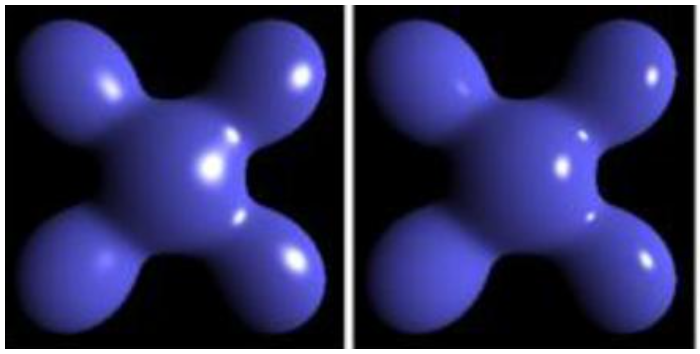


Рисунок 1.10. Сравнение моделей Фонга и Блинна-Фонга.

## Преобразования.

Так как задача состоит в моделировании динамического изображения, необходимо уметь изменять и перемещать объекты сцены. Преобразования на плоскости - изменение значений точек на плоскости:

Линейное преобразование точки может быть записано в скалярной форме системой

(1.2)

Однородные координаты – представление точки в n-мерном пространстве задается при помощи n+1 координат. Для двумерного случая - (x, y, w), где w - масштабный множитель (для плоского случая w = 1).

Используя однородные координаты и матрицу преобразований

(1.3),

преобразование можно записать так:

(1.4)

Методы двумерных преобразований по аналогии распространяются и на изображения трехмерных объектов. Матрица преобразований в трехмерном пространстве в однородных координатах будет иметь размерность 4\*4, а точки (x, y, z) заменяются четверткой (xw, yw, zw, w), где w отлично от 0

### Перенос

Перенос - это изменение местоположения изображения. Задается тремя параметрами: dx - смещение по оси абсцисс, dy - смещение по оси ординат, dz – смещение по оси аппликат. Матрица переноса в трехмерном пространстве:

(1.5)

### Масштабирование

Масштабирование - это изменение размеров и пропорций изображения. Задается шестью параметрами: kx, ky, kz – коэффициенты масштабирования, (xm, ym, zm) – координаты центра масштабирования. Матрица масштабирования относительно начала координат в трехмерном пространстве:

(1.6)

### Поворот

Поворот - это операция, заключающаяся в изменении ориентации изображения. Для поворота вокруг каждой оси требуется знать угол поворота θ. Матрица поворота в трехмерном пространстве вокруг осей Z, X и Y, соответственно:

(1.7)

(1.8)

(1.9)

## Вывод

В соответствии со сформулированными критериями выбора алгоритма удаления невидимых линий и поверхностей, а также модели освещения для реализации поставленной задачи, в результате анализа существующих решений были выбраны:

1. Алгоритм обратной трассировки лучей для удаления невидимых линий и поверхностей. Он позволяет получить реалистичное изображение высокого качества, в котором учитываются такие оптические эффекты, как отражение и отбрасывание теней, а также реализована возможность работы с любыми телами.
2. Модель Фонга выбрана для расчета интенсивности в точке. Она позволяет учитывать необходимые компоненты освещения - диффузную и отражающую составляющую отражения, а также рассеянное освещение. При этом она не перегружена другими параметрами, что не будет пагубно влиять на скорость работы программы.

А для моделирования динамического изображения будут применяться преобразования точек в пространстве.

## Конструкторская часть

В данном разделе приводятся требования к функционалу программного обеспечения. Затем подробнее рассматриваются выбранные решения, а именно: алгоритм обратной трассировки лучей и вычисление интенсивности света в точке по модели Фонга. Также приводятся нужные в них вспомогательные расчеты, способ ускорения работы программы и диаграмма классов.

## Требования к программе

Программа должна предоставлять следующие возможности:

1. Визуализация сцены, состоящей из описанных в пункте 1.2 объектов.
2. Запуск, останов и изменение направления вращения флюгера.
3. Изменение материалов, из которых изготовлены части флюгера.
4. Изменение положения камеры и ее поворот.
5. Изменение параметров источников освещения и добавление новых точечных источников.

## Алгоритм обратной трассировки лучей

На рисунке 2.1 представлена схема алгоритма обратной трассировки лучей.

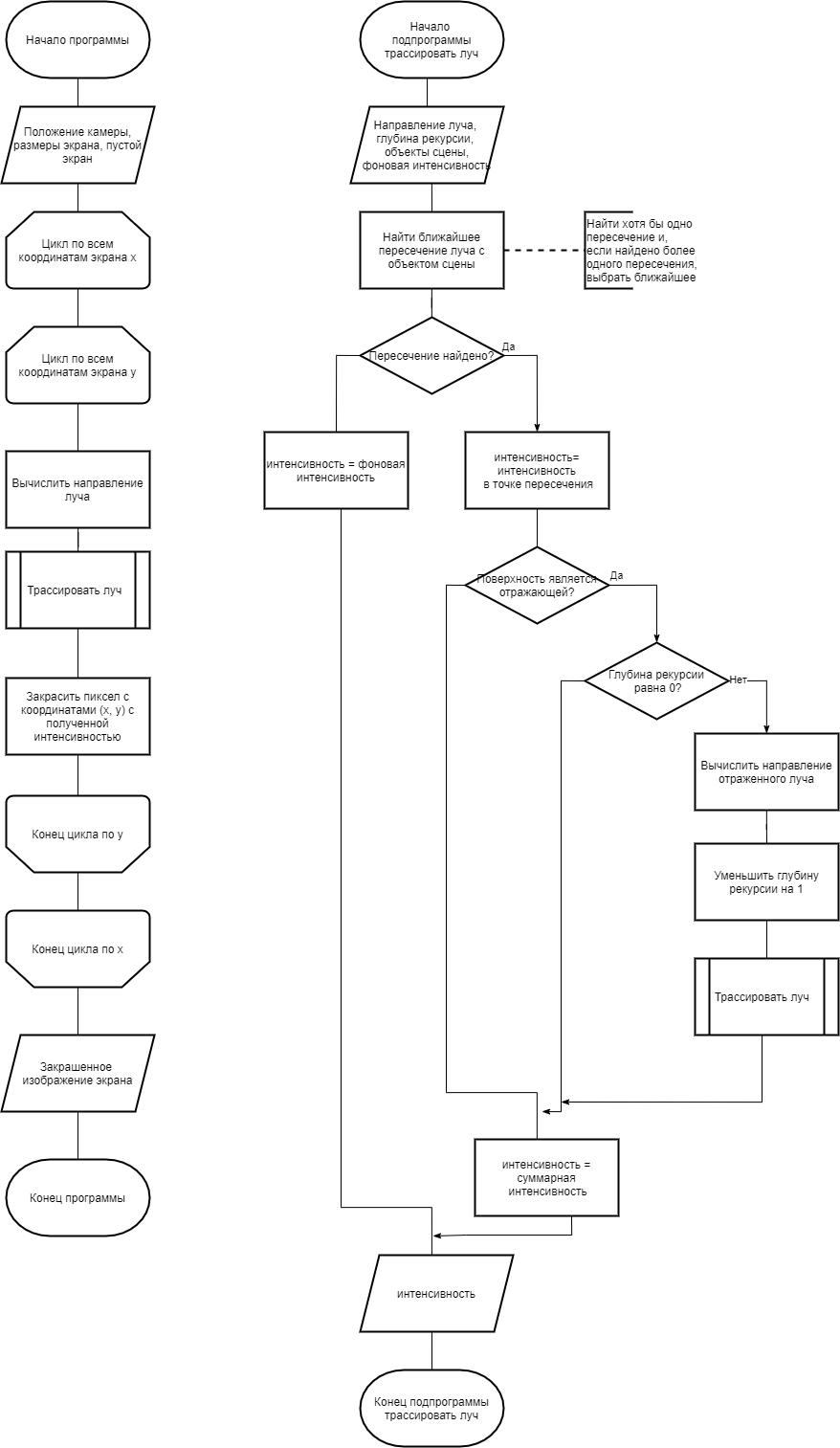


Рисунок 2.1. Схема алгоритма обратной трассировки лучей.

Ри

Ри

1) Нужно ли здесь дополнительно указывать, какие параметры передаются в подпрограмму? Например, так: трассировать луч(направление отраженного луча, глубина рекурсии, ….)

2) Такие данные, как глубина рекурсии, объекты сцены, фоновая интенсивность, используются уже в подпрограмме трассировать луч. Нужно ли их указывать в качестве входных параметров для основной программы (все-таки именно она их будет передавать в подпрограмму)?

## Пересечение луча с объектами сцены

Для того чтобы воспользоваться алгоритмом обратной трассировки лучей, нужно уметь находить пересечение луча с объектами сцены: сферой, плоскостью, цилиндром, прямоугольным параллелепипедом и четырехугольной пирамидой.

Для этого луч представляют с помощью параметрического уравнения. Пусть – начало луча (позиция камеры), V-текущий пиксель на экране, – вектор, показывающий направление луча. Тогда любую точку 𝑃 луча можно представить так:

= + , (2.3.1)

Уравнение 2.3.1 эквивалентно уравнению 2.3.2:

(2.3.2)

Так как точка пересечения луча и объекта сцены принадлежит и первому, и второму, она должна удовлетворять двум уравнениям: уравнению луча и уравнению, которое задает поверхность объекта сцены. Поэтому поиск точки пересечения сводится к составлению системы перечисленных уравнений и поиску ее решения.

При этом если для найденного параметра не выполняется условие , значит пересечение принадлежит продолжению луча за камеру и оно отбрасывается. Далее в формулах оно будет опущено, но подразумеваться.

### Сфера

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

, или

(2.3.3)

Имея уравнения, описывающие точки сферы (2.3.3) и точки луча (2.3.1), решаем систему, подставляя второе уравнение в первое:

Разложим скалярное произведение и преобразуем его. В результате получим квадратное уравнение:

Остается решить уравнение 2.3.5, чтобы найти параметры точек пересечения. При этом если у уравнения нет решений, значит луч не пересекается со сферой (рис. 2.3.1, слева), если решение одно, то луч касается сферы (рис. 2.3.1, в центре), два - луч входит в сферу и выходит из неё (рис. 2.3.1, справа) и нужно выбрать наименьшее решение – где луч входит в сферу.

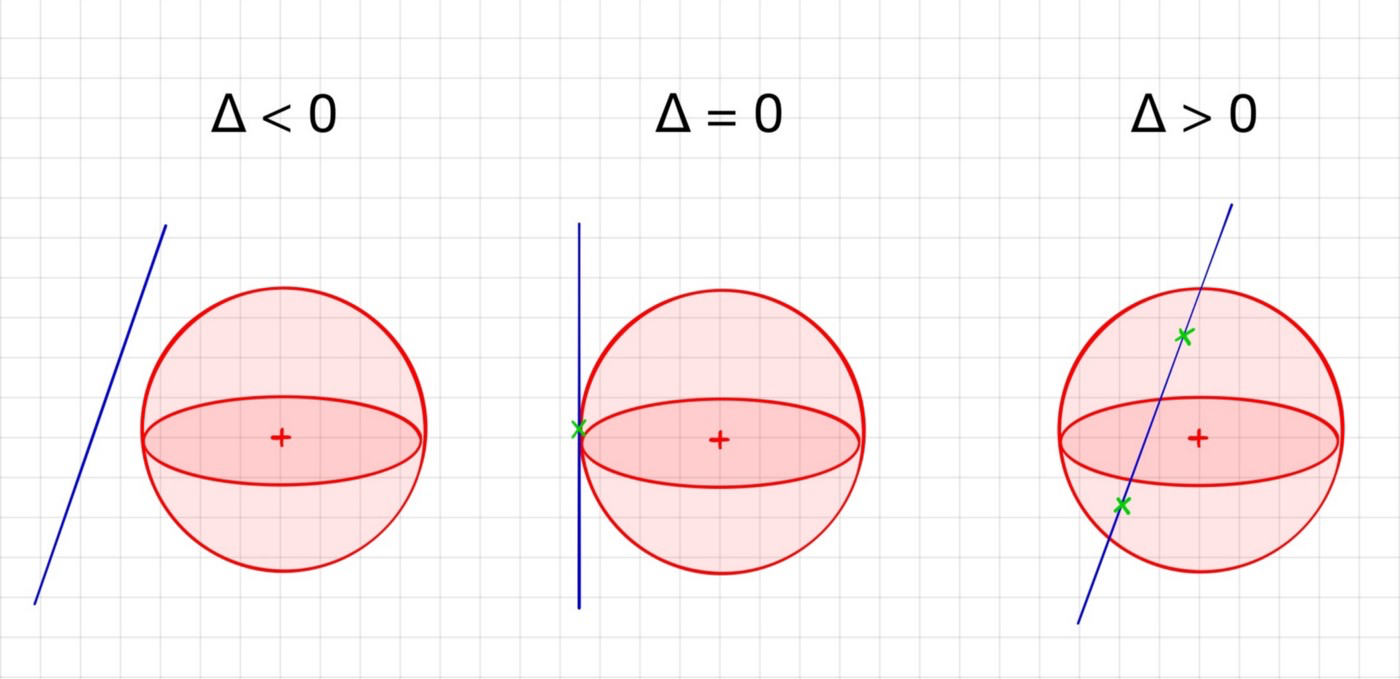


Рисунок 2.3.1. Взаимное расположение луча и сферы.

### Плоскость

Плоскость можно задать с помощью вектора нормали к ней и одной точки С, принадлежащей этой плоскости. Вектор из точки С до любой другой точки на плоскости P перпендикулярен нормали :

(2.3.6)

Имея уравнения, описывающие точки плоскости (2.3.6) и точки луча (2.3.1), решаем систему, подставляя второе уравнение в первое:

(2.3.7)

После преобразований получаем линейное уравнение:

(2.3.8)

Остается решить уравнение 2.3.8, чтобы найти параметр точки пересечения:

(2.3.9)

Если , значит векторы направления луча и нормали к плоскости перпендикулярны, а, следовательно, вектор направления луча параллелен самой плоскости и не пересекает ее.

Р

Ри

### Цилиндр

Цилиндр можно задать с помощью следующих параметров: радиус r, центр основания C, единичный вектор оси , высота H. Поверхность цилиндра условно делится на 2 части: боковая поверхность и два круга-основания.

Чтобы найти пересечение луча и оснований цилиндра достаточно найти пересечение луча с плоскостями и , которые содержат основания, по алгоритму из предыдущего пункта, а затем проверить, что найденные точки находятся на расстоянии не более r от оси цилиндра.

В этом пункте рассмотрим пересечение луча с боковой поверхностью цилиндра. Пусть P – точка пересечения луча с боковой поверхностью, h – расстояние от плоскости основания до этой точки, M – точка на оси цилиндра на том же расстоянии h от основания:

(2.3.10)

Тогда вектор параллелен основанию и, соответственно, перпендикулярен вектору оси цилиндра:

(2.3.11)

Выразим вектор через уравнения для M (2.3.10) и P (2.3.1):

(2.3.12)

Подставляя 2.3.12 в 2.3.11, имеем:

(2.3.13)

Длина вектора равна радиусу цилиндра:

( (2.3.14)

Подставляя в последнее уравнение выражение для h (2.3.13), получаем квадратное уравнение:

(2.3.15)

Остается решить это уравнение, чтобы найти точки пересечения луча с боковой поверхностью цилиндра. Рассуждения о геометрическом смысле количества корней аналогичны случаю со сферой.

### Прямоугольный параллелепипед и четырехугольная пирамида.

Прямоугольный параллелепипед и четырехугольную пирамиду удобно представить с помощью группы треугольников: 12 для параллелепипеда (по 2 на каждую грань) и 6 для пирамиды (по одному на каждую боковую грань и 2 на четырехугольное основание). Тогда задача поиска точки пересечения луча и одного из этих примитивов сводится к поиску точки пересечения луча и треугольников его составляющих.

Пусть треугольник задан тремя своими вершинами V0, V1, V2, не лежащими на одной прямой. Точки на треугольника P можно задать с помощью барицентрических координат (u, v):

(2.3.16)

Подставим в это уравнение выражение для точек луча P (2.3.1)

(2.3.17)

После перегруппировки членов получаем систему линейных уравнений:

(2.3.18)

Обозначим , и решим систему по правилу Крамера:

(2.3.19)

Используя свойство смешанного произведения, перепишем систему так:

(2.3.20)

Обозначим и перепишем систему:

Остается найти параметры t, u и v и проверить выполнение для них соответствующих условий. Если хотя бы одно из них не выполняется, значит пересечения нет.

## Нахождение отраженного луча

В алгоритме трассировки лучей также необходимо уметь находить направление отраженного луча.

По закону отражения вектор падающего света, нормаль к поверхности и вектор отражения лежат в одной плоскости, причем на этой плоскости угол падения равен углу отражения. Пусть – вектор падающего света, – вектор отражения, - нормаль к поверхности, t – угол падения (см. рис. 2.6).

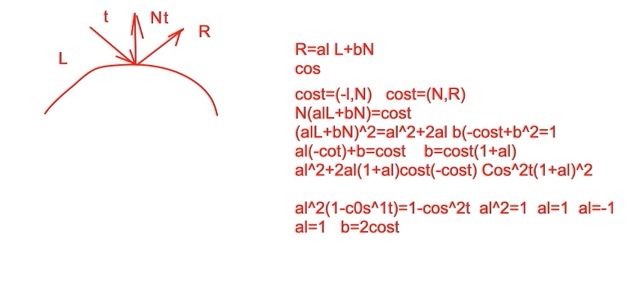


Рисунок 2.6. Определение направления отраженного луча.

Вектор представляется суммой векторов и с некоторыми коэффициентами α и β. Далее решается система из двух уравнений и находятся эти коэффициенты.

Из условия, что угол падения равен углу отражения:

Подставляя выражение для из 2.26 в 2.27, получаем:

Из 2.28 выразим β:

Скалярно умножим единичный вектор  на себя :

Подставляя 2.29 в 2.30, применяя преобразования, получаем выражение для :

=

Откуда находим два значения α: 1 или -1. Второй корень не подходит из физических соображений. Итак,

α=1. (2.32)

Подставляя 2.32 в 2.29, находим β:

Подставляя значения α и β в выражение для отраженного луча (2.26), находим вектор  :

Представляя cos(t) через скалярное произведение единичных векторов , окончательно получаем:

## Уменьшение времени работы алгоритма.

Поскольку алгоритм обратной трассировки лучей обрабатывает каждый пиксель экрана независимо, можно использовать параллельные вычисления для уменьшения времени его работы, разбив экран на некоторые части. Разбиение экрана можно производить различными способами, на визуальном результате это никак не сказывается. Наиболее часто используется горизонтальное или вертикальное разбиение.

## Модель Фонга

Модель Фонга выбрана для расчета интенсивности в точке. Алгоритм расчета интенсивности I освещения в точке P поверхности с нормалью N, вектором взгляда V и степенью, аппроксимирующей пространственное распределение зеркально отраженного света s представлен на рис. 2.7:

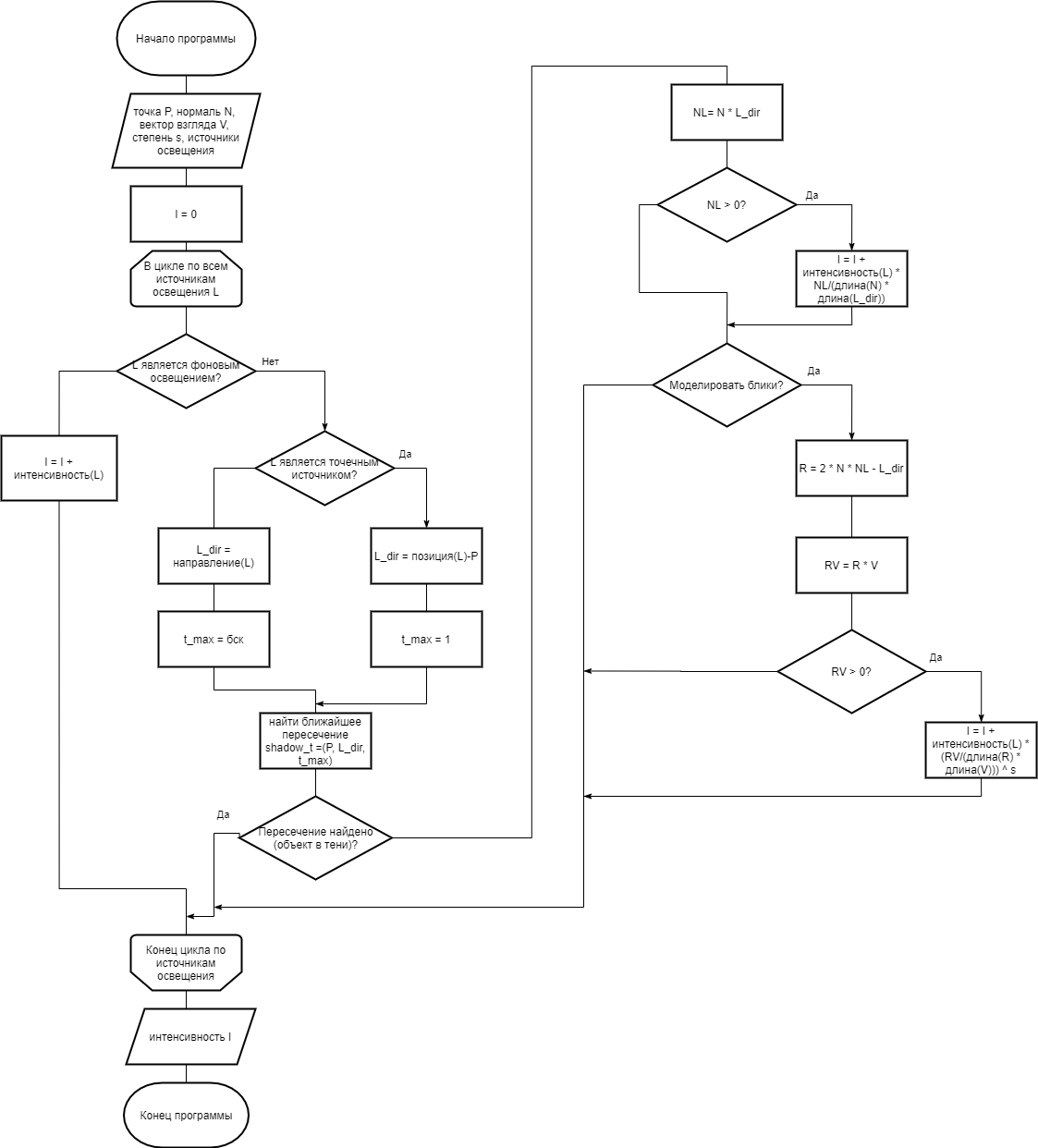


Рисунок 2.7. Схема алгоритма расчета освещения по модели Фонга.

## Выбор используемых типов и структур данных

На рисунке 2.8 приведена диаграмма классов.

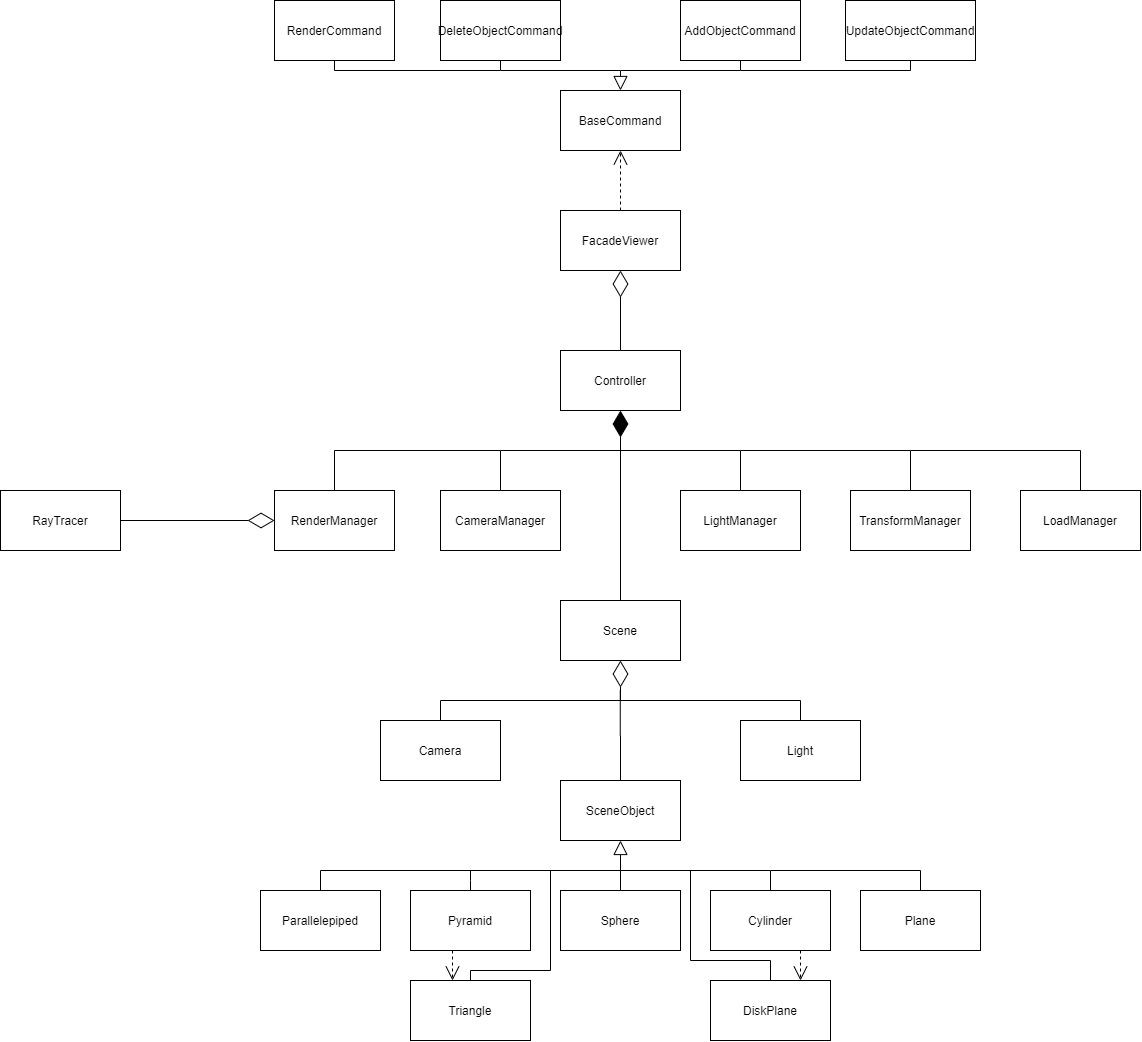


Рисунок 2.8. Диаграмма классов программы.

В разрабатываемой программе используются следующие классы:

* SceneObject – базовый класс объектов, из которых состоит флюгер, с возможностью изменения параметров материала, из которого изготовлен объект:
  + Sphere – класс сферы, описывается координатами центра и радиусом
  + Cylinder – класс цилиндра, описывается координатами центра основания, радиусом, направлением оси и высотой. Вспомогательный класс для определения точек пересечения с лучом – DiskPlane (плоскость основания)
  + Parallelepiped – класс параллелепипеда, описывается координатами двух своих вершин: той, у которой все три координаты минимальны, и той, у которой все три координаты максимальны
  + Pyramid – класс треугольной пирамиды, описывается координатами вершины и точек основания. Вспомогательный класс для определения точек пересечения с лучом – Triangle (грани)
* Camera – класс камеры с возможностью перемещения по сцене и изменения углов наклона
* Light – класс источника освещения с возможностью перемещения по сцене и изменения интенсивности и направления (в случае направленного источника)
* Scene – класс сцены, включающий в себя объекты, камеру и источники освещения
* LoadManager, CameraManager, LightManager, TransformManager, RenderManager – классы-менеджеры загрузки, управления камерой, управления источниками освещения, изменения объектов и рендеринга, соответственно
* Raytracer – класс трассировки лучей
* Controller – класс для взаимодействия управляющих классов с классом интерфейса.
* FaсadeViewer – класс, который предоставляет интерфейс работы системы.
* BaseCommand – базовый класс команд, поступающих в программу:
  + AddObjectCommand – команда добавления объекта сцены
  + DeleteObjectCommand – команда удаления объекта из сцены
  + UpdateObjectCommand – команда изменения параметров объекта сцены
  + RenderCommand – команда запуска рендеринга

## Вывод

В данном разделе были подробно рассмотрены выбранные решения - алгоритм обратной трассировки лучей и вычисление интенсивности света в точке по модели Фонга, приведены соответствующие блок-схемы и расчеты, а также была представлена диаграмма классов.

# Список использованной литературы

1. Роджерс Д. Математические основы машинной графики. / Роджерс Д., Адамс Дж. – М.: Мир, 1989. – 512с.
2. Дымченко, Лев. Пример реализации в реальном времени метода трассировки лучей: необычные возможности и принцип работы. Оптимизация под SSE [Электронный ресурс], режим доступа: https://www.ixbt.com/video/rt-raytracing.shtml (дата обращения: 03.07.2021)
3. Ю. М. Баяковский. Трассировка лучей из книги Джефа Проузиса [Электронный ресурс], режим доступа: https://www.graphicon.ru/oldgr/courses/cg99/notes/lect12/prouzis/raytrace.htm (дата обращения: 03.07.2021)
4. https://docplayer.ru/30012180-Lokalnye-i-globalnye-modeli-osveshcheniya-model-fonga-zakraska-fonga-i-guro.html

<http://miet.aha.ru/cg/cg_2015.pdf>

<https://cyberleninka.ru/article/n/parallelnaya-obrabotka-v-algoritmah-vizualizatsii-s-trassirovkoy-luchey>

<https://compgraphics.info/3D/lighting/phong_reflection_model.php> !!!!

<https://habr.com/ru/post/342708/> про треугольники в параллелепипеде

,{"xl":-5.0,"xr":5.0,"yu":82.0,"yd":72.0,"zf":47.0,"zn":37.0,"name":"дальний\_куб","moving":true,"material":{"color":{"x":0.0,"y":255.0,"z":0.0},"specular":0.0,"reflective":0.2}},{"xl":-5.0,"xr":5.0,"yu":82.0,"yd":72.0,"zf":-37.0,"zn":-47.0,"name":"ближний\_куб","moving":true,"material":{"color":{"x":0.0,"y":255.0,"z":0.0},"specular":0.0,"reflective":0.2}}

,{"A":{"x":-5.0,"y":82.0,"z":49.0},"B":{"x":-5.0,"y":82.0,"z":39.0},"D":{"x":5.0,"y":82.0,"z":49.0},"P":{"x":0.0,"y":92.0,"z":44.0},"name":"верхняя\_дальняя\_пирамида","moving":true,"C":{"x":5.0,"y":82.0,"z":39.0},"material":{"color":{"x":0.0,"y":0.0,"z":255.0},"specular":0.0,"reflective":0.0}},{"A":{"x":-5.0,"y":82.0,"z":-49.0},"B":{"x":-5.0,"y":82.0,"z":-39.0},"D":{"x":5.0,"y":82.0,"z":-49.0},"P":{"x":0.0,"y":92.0,"z":-44.0},"name":"верхняя\_ближняя\_пирамида","moving":true,"C":{"x":5.0,"y":82.0,"z":-39.0},"material":{"color":{"x":0.0,"y":0.0,"z":255.0},"specular":0.0,"reflective":0.0}}

<https://masters.donntu.org/2015/frt/yablokov/library/transl.htm> луч треугольник