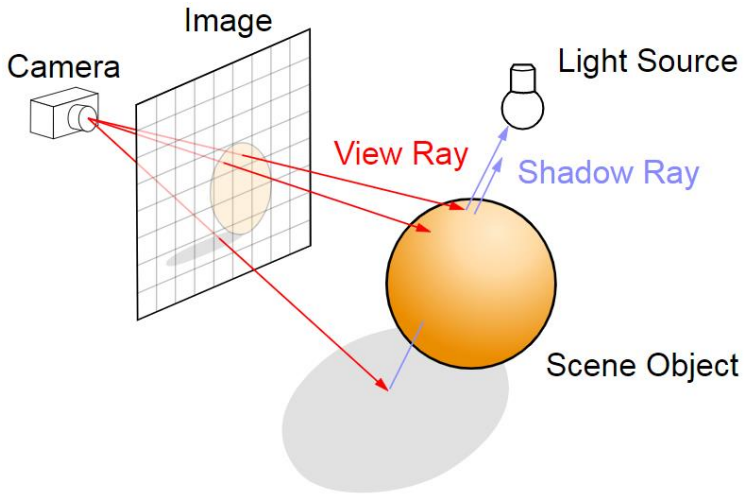
Трассировка лучей

1. Формулировка задачи

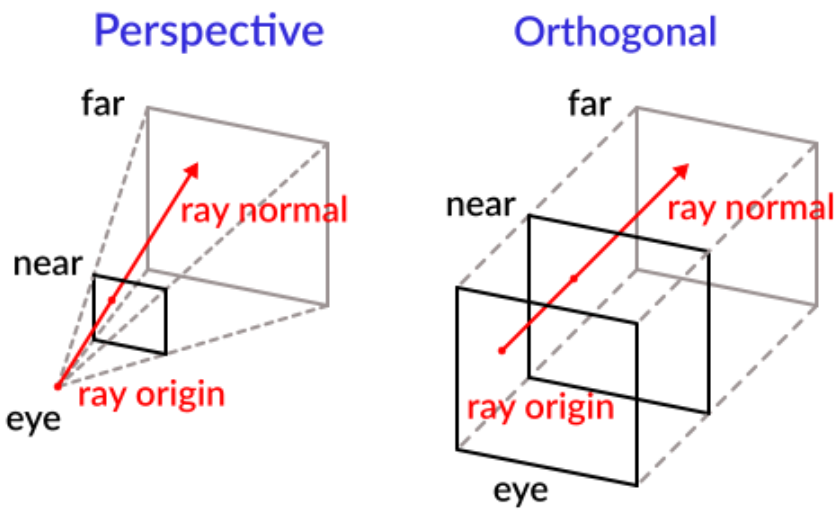
Задача трассировки лучей формулируется следующим образом. В пространстве есть некая точка обзора (камера, глаз). Также в пространстве присутствует некая трёхмёрная конструкция (сцена). Между ними располагается прямоугольник, поделённый на квадратные пиксели (экран).



Задача трассировки лучей заключается в том, чтобы из точки обзора провести луч через центр каждого пикселя экрана – и понять, пересекает ли он какой-либо из объектов сцены. Если луч ничего не пересекает, тогда соответствующий пиксель экрана нужно закрасить **белым** цветом. Если луч пересекает какой-либо из объектов, требуется закрасить пиксель нужным цветом, отличным от белого. В самом простом варианте этой задачи пиксель закрашивается чёрным цветом, независимо от того, какой из объектов сцены он пересекает и в какой точке.

2. Виды проекций

В разделе 1 описана так называемая **перспективная проекция**, когда все лучи выходят из одной точки. Возможен другой вариант: **ортогональная проекция**. В этом случае лучи идут через центр каждого пикселя параллельно друг другу. Соответственно, задавать нужно не координаты точки обзора, а компоненты вектора, параллельно которому пойдут лучи.



В остальном – всё так же, как и в перспективной проекции.

3. Виды объектов на сцене

Существует множество видов трёхмерных геометрических фигур. В нашей задаче мы ограничимся тремя видами:

- шар (задаётся координатами центра и радиусом);
- бокс – прямоугольный параллелепипед, грани которого параллельны координатным плоскостям (задаётся координатами двух своих вершин: той, у которой все три координаты минимальны, и той, у которой все три координаты максимальны);
- тетраэдр (задаётся координатами четырёх своих вершин).

Самая главная математика в этой задаче будет зашита в функции, определяющей: пересекается луч с фигурой или нет.

При разработке программы разумно создать базовый абстрактный класс «Фигура», а классы конкретных фигур наследовать от него. Если в вашей задаче предполагается использование фигур разного вида – такая архитектура совершенно обязательна.

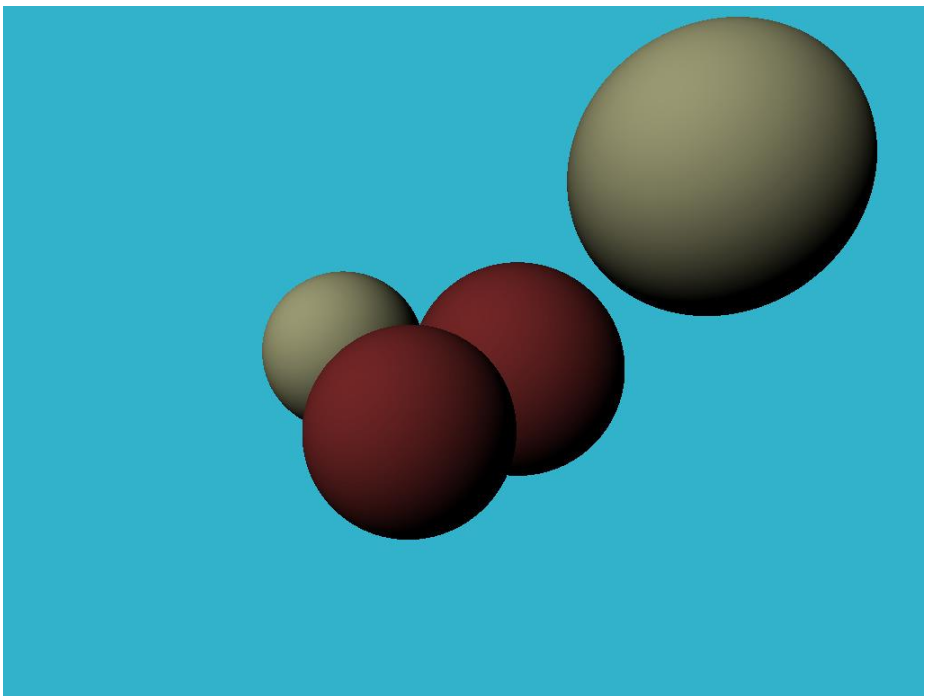
4. Цвета раскраски пикселей и учёт освещённости

Как было сказано в разделе 1, пиксели, которым соответствуют лучи, не пересекающие не один из объектов, всегда закрашиваются белым цветом. Что касается пикселей, соответствующие которым лучи что-либо

пересекают, самый простой вариант – закрашивание их чёрным, независимо от того, какой объект и в какой точке они пересекают.

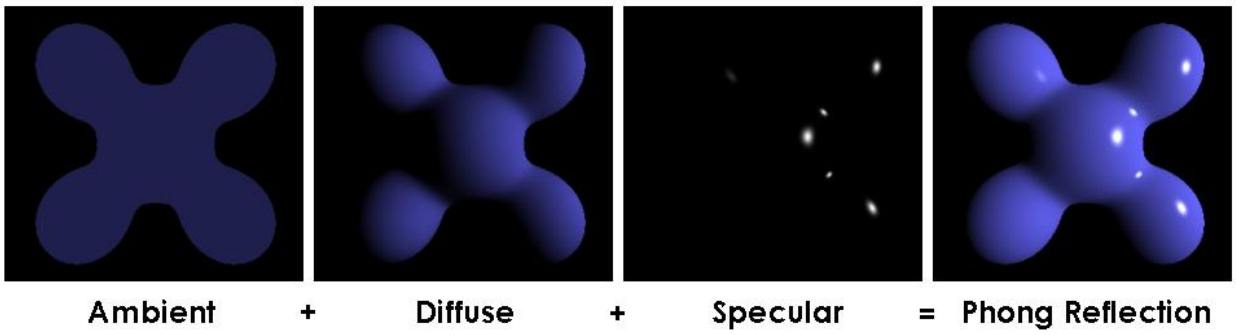
Задача чуть посложнее – закрашивать пиксели разными цветами, в зависимости от того, какой объект пересекает луч. Например, закрашивать разными оттенками серого, и чем ближе находится объект – тем темнее/светлее будет оттенок. Или закрасить разными цветами в зависимости от вида объекта: все сферы считать красными, все боксы – зелёными, все тетраэдры – синими. Или просто привязать к каждому объекту свой цвет (цвет задаётся тремя числами от 0 до 255). Если луч пересекает несколько объектов – то соответствующий пиксель закрашивается цветом, соответствующим первому (ближайшему) пересечённому объекту. В этом случае цвет пикселя зависит от объекта, который пересекает луч – но не от точки пересечения (для всех точек одного и того же объекта цвет будет одинаков).

Задача ещё сложнее – учёт **освещённости.** В этом случае дополнительно вводится точка, в которой располагается источник света. Яркость цвета каждой точки объекта зависит от того, под каким углом в эту точку падает свет из источника. Допустим, мы построили луч от камеры через центр какого-то пикселя экрана и попали этим лучом в объект (фигуру), которой соответствует фиолетовый цвет (255, 0, 255). Из той точки фигуры, куда попал луч, мы построим два единичных вектора: нормаль к поверхности фигуры и вектор освещённости (т.е. направленный к источнику цвета). Числа, задающие цвет фигуры, умножим на скалярное произведение этих двух единичных векторов – и полученным цветом закрасим нужный пиксель экрана. То есть если, к примеру, в указанной точке свет падает под прямым углом к поверхности – пиксель будет закрашен цветом (255, 0, 255). Если свет падает под углом 60 градусов – закрасим пиксель цветом (128, 0, 128). Если свет на данную точку не падает вовсе (скалярное произведение вектора нормали и вектора освещённости отрицательно) – значит, закрашиваем пиксель чёрным цветом (0, 0, 0).



На показанном рисунке свет на сферы падает слева и сверху. И на нём все поверхности – матовые, т.е. бликов от источника света ни на одной из них нет.

Моделирование **глянцевых поверхностей** – ещё одно усложнение задачи. Для этого также нужен источник света. Цвет каждой точки объекта, отображаемой на экране, зависит от направления отражённого луча света. Если он является противоположно направленным лучу от камеры к рассматриваемой точке (т.е. отражённый луч светит прямо в камеру) – соответствующий пиксель экрана раскрашивается белым. Чем ближе отражённый луч к направлению от точки к камере – тем белее и ярче соответствующий пиксель. Для описания освещённости матовых и глянцевых поверхностей рекомендуется использовать модель Фонга, о которой можно прочитать в Интернете.



Нужно отметить, что в реальных задачах на трассировку лучей может быть много усложнений по сравнению с вышесказанным:

- несколько источников света;
- тени одних объектов на других;
- отражения одних объектов в других (зеркальные поверхности)

и т.п. В нашей задаче мы их рассматривать не будем, ограничившись в самом сложном случае диффузным и зеркальным освещением.

5. Сохранение экрана в виде изображения

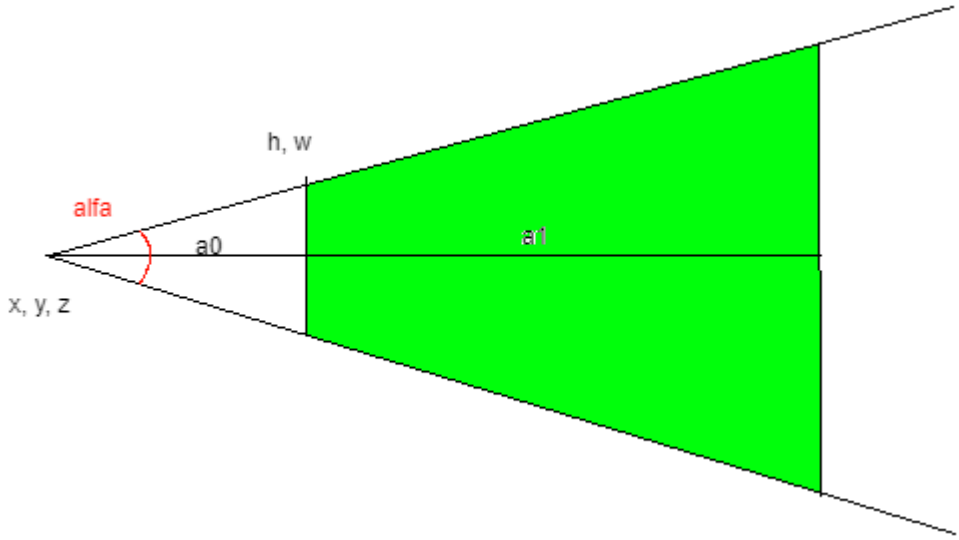
Писать программу нужно на C++. Это даст возможность распараллеливать на OpenMP, что и нужно делать в обязательном порядке.

Когда всем пикселям экрана задан цвет – его надо сохранить в виде BMP-изображения. Для этого можно использовать сторонние библиотеки: CImg.h, STB (stb_image_write.h) либо какие-то другие. Стороннюю библиотеку для использования нужно откуда-то скачать и подключить к проекту.

6. Входные данные

Входные данные для решения задачи должны считываться из текстового файла. Файл должен содержать следующие данные:

1) координаты камеры x, y, z (три вещественных числа);
2) координаты вектора нормали к экрану (три вещественных числа);
3) координаты вектора, перпендикулярного нормали к экрану, задающего направление «вверх» (три вещественных числа);
4) расстояние от камеры до экрана a_0 (одно вещественное число);
5) расстояние от камеры до границы видимости $a_0 + a_1$ (одно вещественное число);
6) угол обзора по вертикали α (одно вещественное число);
7) ширина w и высота h экрана в пикселях (два целых числа);
8) координаты источника света (три вещественных числа) – опционально.



На рисунке зелёным цветом показана область видимости. Обрабатываются только те объекты, которые лежат в этой области.

Вероятно, есть смысл продумать систему комментариев либо систему ключевых слов, чтобы не запутаться в полученном большом текстовом файле. Например, так:

cam 0.0 0.0 0.0

normal 0.0 0.0 1.0

up 0.0 1.0 0.0

screen 2.0

limit 20.0

alpha 30.0

width 640

height 480

light 2.0 2.0 0.0

Кроме того, нужно задавать тип и параметры геометрических фигур – объектов на сцене. (Возможно, это разумнее делать в другом файле.) Например:

sphere 0.0 0.0 0.0 1.0

box 0.0 0.0 0.0 1.0 1.0 1.0

tetra 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0

Если в задаче требуется задавать ещё и цвет фигур, тогда это будет выглядеть примерно так:

sphere 0.0 0.0 0.0 1.0 255 255 0

box 0.0 0.0 0.0 1.0 1.0 1.0 255 0 255

tetra 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 255 0 0

Входных данных в задаче много. Чтобы не запутаться, порядок их использования следующий.

1) Координаты камеры (глаза) дают нам точку в пространстве, от которой мы будем "плясать".
2) Предполагаем, что отрезок, проведённый от камеры в центр экрана, перпендикулярен экрану. То есть расстояние от камеры до экрана - это расстояние от камеры до центра экрана. Это расстояние в задаче тоже задаётся - таким образом, мы можем мысленно построить вокруг камеры сферу с таким радиусом. И эта сфера будет представлять собой все возможные местоположения центра экрана.
3) Чтобы понять, в какой именно точке мысленно построенной сферы находится центр экрана - у нас есть заданный вектор нормали к экрану. Для полной однозначности следует определиться: задавать либо вектор нормали к экрану, направленный в сторону камеры - либо в противоположную сторону. Тогда сможем мысленно построить плоскость, касательную сфере - это и будет плоскость, в которой лежит экран.
4) Вектор, перпендикулярный нормали к экрану и задающий направление "вверх", даст нам возможность задать на плоскости экрана систему координат (ось ординат будет сонаправленной с вектором).
5) Угол обзора по вертикали даст физический размер экрана по вертикали.
6) Высота экрана в пикселях позволит вычислить размер пикселя: нужно физический размер поделить на количество пикселей по высоте.

7) Пиксели, очевидно, квадратные. Поэтому размер пикселя, умноженный на ширину экрана в пикселях даст физический размер экрана по горизонтали.

Таким образом, мы получаем всё, что нужно, для работы с перспективной проекцией.

1. Перспективная проекция, одна сфера, ей соответствует серый цвет (128, 128, 128). Необходим учёт освещения, поверхность сферы – матовая.
2. Перспективная проекция, один бокс, ему соответствует серый цвет (128, 128, 128). Необходим учёт освещения, поверхность бокса – матовая.
3. Перспективная проекция, один тетраэдр, ему соответствует серый цвет (128, 128, 128). Необходим учёт освещения, поверхность тетраэдра – матовая.
4. Перспективная проекция, произвольное количество сфер. На экране все сферы отображаются бирюзовым цветом (0, 255, 255). Необходим учёт освещения, все поверхности – матовые.
5. Перспективная проекция, произвольное количество боксов. На экране все боксы отображаются пурпурным цветом (255, 0, 255. Необходим учёт освещения, все поверхности – матовые.
6. Перспективная проекция, произвольное количество тетраэдров. На экране все тетраэдры отображаются жёлтым цветом (255, 255, 0). Необходим учёт освещения, все поверхности – матовые.
7. Перспективная проекция, произвольное количество сфер и боксов. Фигура, центр которой находится всех ближе к камере, на экране отображается красным цветом (255, 0, 0). Дальняя фигура – зелёным (0, 255, 0). Промежуточным фигурам присваивается нечто среднее по принципу: чем ближе к камере – тем краснее, чем дальше от камеры – тем зеленее. Необходим учёт освещения, все поверхности – матовые.
8. Перспективная проекция, произвольное количество сфер и боксов. Фигура, центр которой находится всех ближе к камере, на экране отображается красным цветом (255, 0, 0). Дальняя фигура – зелёным (0, 255, 0). Промежуточным фигурам присваивается нечто среднее по принципу: чем ближе к камере – тем краснее, чем дальше от камеры –

тем зеленее. Необходим учёт освещения, поверхности всех сфер – глянцевые, боксов – матовые.

9. Перспективная проекция, произвольное количество сфер и тетраэдров. Фигура, центр которой находится всех ближе к камере, на экране отображается красным цветом (255, 0, 0). Дальняя фигура – синим (0, 0, 255). Промежуточным фигурам присваивается нечто среднее по принципу: чем ближе к камере – тем краснее, чем дальше от камеры – тем синее. Необходим учёт освещения, все поверхности – матовые.
10. Перспективная проекция, произвольное количество сфер и тетраэдров. Фигура, центр которой находится всех ближе к камере, на экране отображается красным цветом (255, 0, 0). Дальняя фигура – синим (0, 0, 255). Промежуточным фигурам присваивается нечто среднее по принципу: чем ближе к камере – тем краснее, чем дальше от камеры – тем синее. Необходим учёт освещения, поверхности всех сфер – глянцевые, тетраэдров - матовые.
11. Перспективная проекция, произвольное количество боксов и тетраэдров. Фигура, центр которой находится всех ближе к камере, на экране отображается зелёным цветом (0, 255, 0). Дальняя фигура – синим (0, 0, 255). Промежуточным фигурам присваивается нечто среднее по принципу: чем ближе к камере – тем зеленее, чем дальше от камеры – тем синее. Необходим учёт освещения, все поверхности – матовые.
12. Перспективная проекция, произвольное количество боксов и тетраэдров. Фигура, центр которой находится всех ближе к камере, на экране отображается зелёным цветом (0, 255, 0). Дальняя фигура – синим (0, 0, 255). Промежуточным фигурам присваивается нечто среднее по принципу: чем ближе к камере – тем зеленее, чем дальше от камеры – тем синее. Необходим учёт освещения, все поверхности – глянцевые.

13. Перспективная проекция, произвольное количество любых фигур. Сферы на экране отображаются красным цветом, боксы – зелёным, тетраэдры – синим. Необходим учёт освещения, все поверхности – матовые.
14. Перспективная проекция, произвольное количество любых фигур. Сферы на экране отображаются красным цветом, боксы – зелёным, тетраэдры – синим. Необходим учёт освещения, все поверхности – глянцевые.
15. Перспективная проекция, произвольное количество любых фигур. Каждой фигуре соответствует свой цвет, заданный во входном файле. Необходим учёт освещения, все поверхности – матовые.
16. Перспективная проекция, произвольное количество любых фигур. Каждой фигуре соответствует свой цвет, заданный во входном файле. Необходим учёт освещения, все поверхности – глянцевые.
17. Перспективная проекция, произвольное количество любых фигур. На экране фигуры отображаются различными оттенками серого цвета, в зависимости от объёма. Фигуре самого большого объёма соответствует цвет (64, 64, 64), фигуре самого маленького объёма – цвет (191, 191, 191). Промежуточным фигурам присваивается нечто среднее по принципу: чем меньше фигура – тем она светлее. Необходим учёт освещения, все поверхности – матовые.
18. Перспективная проекция, произвольное количество любых фигур. На экране фигуры отображаются различными оттенками серого цвета, в зависимости от объёма. Фигуре самого большого объёма соответствует цвет (64, 64, 64), фигуре самого маленького объёма – цвет (191, 191, 191). Промежуточным фигурам присваивается нечто среднее по принципу: чем меньше фигура – тем она светлее. Необходим учёт освещения, все поверхности – глянцевые.
19. Перспективная проекция, произвольное количество любых фигур. На экране фигуры отображаются различными оттенками серого цвета, в

зависимости от расстояния до камеры. Фигуре, центр которой всех ближе к камере, соответствует цвет (64, 64, 64). Самой дальней фигуре соответствует цвет (191, 191, 191). Промежуточным фигурам присваивается нечто среднее по принципу: чем дальше центр фигуры от камеры – тем она светлее. Необходим учёт освещения, все поверхности – матовые.

20. Перспективная проекция, произвольное количество любых фигур. На экране фигуры отображаются различными оттенками серого цвета, в зависимости от расстояния до камеры. Фигуре, центр которой всех ближе к камере, соответствует цвет (64, 64, 64). Самой дальней фигуре соответствует цвет (191, 191, 191). Промежуточным фигурам присваивается нечто среднее по принципу: чем дальше центр фигуры от камеры – тем она светлее. Необходим учёт освещения, все поверхности – глянцевые.