Подготовил: Козлов Д.С.
Дата модификации документа: 28.04.2017 3-00

Задача Raytracing – Лучевая трассировка

Постановка задачи

Требуется разработать приложение, реализующее алгоритм трассировки лучей для расчёта изображения (рендеринга) трёхмерной сцены. Сцена состоит из набора примитивов: сфер, боксов, треугольников и четырёхугольников. При загрузке сцены приложение должно отобразить её в виде проволочной модели. После загрузки сцены пользователь имеет возможность рассчитать изображение сцены. Кроме того, приложение должно позволять сохранять рассчитанное изображение (или изображение проволочной модели) в файл.

Приложение должно иметь следующие кнопки:

1. Загрузка сцены из файла. **[Open]**
2. Загрузка настроек рендеринга из файла. **[Load render settings]**
3. Сохранение настроек рендеринга в файл. **[Save render settings]**
4. Установить положение камеры по умолчанию **[Init]**
5. Показать диалог настроек рендеринга **[Settings]**
6. Переход в режим выбора ракурса **[Select view]**
7. Запуск процесса рендеринга сцены **[Render]**
8. Сохранение изображения в файл **[Save image]**

Когда пользователь успешно загружает сцену, кнопка **Select view** оказывается нажатой. В этом режиме пользователь имеет возможность выбрать желаемые ракурс и поменять настройки рендеринга. Когда пользователь нажимает кнопку **Render**, запускается процесс рендеринга, необходимо заблокировать выбор ракурса на время расчёта изображения (если это не интерактивный режим – см. раздел на лучшее решение). После окончания процесса рендеринга кнопка **Render** оказывается нажатой, а кнопка **Select view** отжатой при этом пользователь видит рассчитанное изображение. Пользователь может нажать **Select view**, при этом кнопка **Render** отжимается и пользователю вновь показывается проволочная модель сцены (в том ракурсе, который был до рендеринга) и пользователь снова имеет возможность выбрать другой ракурс.

Обращаю внимание, что при корректном расчёте изображения очертания фигур в проволочном представлении модели и на рассчитанном изображении должны совпадать, их несовпадение означает максимальную оценку – 2. Рассчитанное изображение и проволочная модель должны занимать клиентскую область окна целиком.
При нажатии кнопки **Init** устанавливается начальное положение камеры, при этом остальные настройки не изменяются.
При нажатии кнопки **Save image** в режиме **Select view** должно быть сохранено изображение проволочной модели сцены. При нажатии **Save image** в режиме **Render** должно быть сохранено рассчитанное изображение. **Save image** действует как Save As, т.е. каждый раз предлагает выбрать файл.

Замечание: Если вы реализуете две кнопки (**Save render settings** и **Save render settings As**), то это будет удобно. Если сделаете одну, тогда только **Save render settings As**.

В режиме **Select view** примитивы отображаются в проволочном виде (как и в задаче Wireframe, параметры отображения выбираете сами). Не забывайте про клиппирование оно должно быть

реализовано. Проволочная модель должна рассчитываться с использованием матричной арифметики, как и в задаче Wireframe.

Формат файла сцены

Файл сцены содержит описание объектов сцены, описание источников и описание рассеянного света. Другие необходимые параметры необходимые для однозначного расчёта изображения сцены задаются в файле настроек рендеринга (см. ниже). Расширение файла сцены “scene”.
Замечание: Форматы файлов допускают однострочные комментарии в стиле C++ и пустые строки (как и в предыдущих задачах).

Заголовок файла сцены:

```
Ar Ag Ab // рассеянный свет в пространстве RGB в диапазоне 0..255
NL // число точечных источников в сцене
// далее идёт NL строк, каждая из которых описывает точечный источник света
LX LY LZ LR LG LB // LX, LY, LZ – положение источника, LR, LG, LB – цвет источника
                  // в пространстве RGB в диапазоне 0..255
```

Далее файл содержит несколько секций, каждая из которых содержит только 1 примитив.

Секция файла сцены:

```
// для сферы:
SPHERE
CENTERx CENTERy CENTERz // координаты центра сферы
RADIUS // радиус сферы
KDr KDg KDb KSr KSg KSb Power // оптические характеристики

// для бокса (ребра параллельны осям):
BOX
MINx MINy MINz // точка с минимальными координатами
MAXx MAXy MAXz // точка с максимальными координатами
KDr KDg KDb KSr KSg KSb Power // оптические характеристики

// для треугольника:
TRIANGLE
POINT1x POINT1y POINT1z   // координаты первой вершины
POINT2x POINT2y POINT2z   // координаты второй вершины
POINT3x POINT3y POINT3z   // координаты третьей вершины
KDr KDg KDb KSr KSg KSb Power // оптические характеристики

// для четырёхугольника:
QUADRANGLE
POINT1x POINT1y POINT1z   // координаты 1й вершины
POINT2x POINT2y POINT2z   // координаты 2й вершины
POINT3x POINT3y POINT3z   // координаты 3й вершины
POINT4x POINT4y POINT4z   // координаты 4й вершины
KDr KDg KDb KSr KSg KSb Power // оптические характеристики
```

При задании оптических характеристик примитивов используются следующие обозначения:
KDr, KDg, KDb – коэффициенты диффузного и рассеянного **(одни и те же)** отражения для (красного, зелёного, синего света).
KSr, KSg, KSb – коэффициенты зеркального отражения для (красного, зелёного, синего света).
Power – показатель зеркальности по Блинну (используется вектор H), см. лекции.

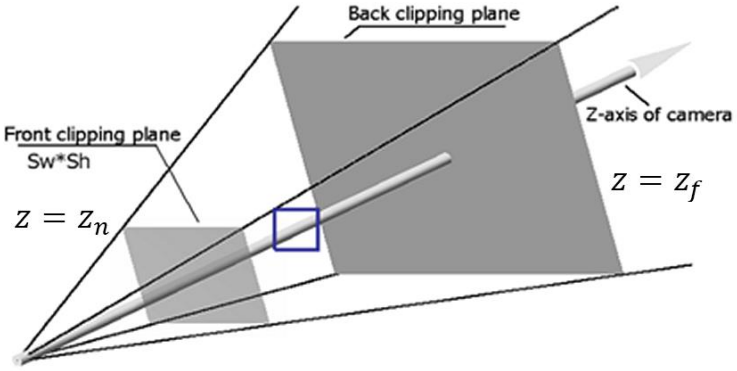
Замечание: Значение цвета рассеянного света и источников заданы в диапазоне от 0 до 255. Это сделано для удобства задания, однако перед расчётом они должны быть переведены в диапазон от 0 до 1 путём деления на 255.

Предложение: В одной из сцен можете использовать набор поверхностей вращения из задачи Wireframe, необходимо лишь экспортировать сетку в виде четырёхугольников.

Формат файла настроек рендеринга

Файл настроек рендеринга содержит информацию о камере и параметры алгоритма лучевой трассировки. Файл сцены в совокупности с файлом настроек рендеринга **ОДНОЗНАЧНО** определяет рассчитываемое изображение (при одинаковом размере клиентской области окна приложения). Расширения файла “render”.

Замечание: Если в директории рядом с файлом сцены **AAAA.scene** располагается файл **AAAA.render**, то при открытии файла сцены он должен быть автоматически загружен. Разрешается иметь несколько файлов настроек рендеринга, тогда они имеют имена **AAAA_DESCIPTION.render**, где DESCRIPTION – это произвольная строка. Такие файлы вы можете добавить в папку Data для того, чтобы, например, продемонстрировать вид вашей сцены с привлекательных позиций (например, AAAA_up.render или AAAA_best.render). Рекомендую протестировать, что после загрузки такого файла сцены у Вас корректно работает кнопка **Init**. Если файла **AAAA.render** нет, то считается, что камера находится в положении "**Init**", см. далее.




```
Br Bg Bb // цвет фона в формате 0..255
GAMMA // значение гаммы
DEPTH // глубина трассировки
QUALITY // rough – грубое, normal – среднее, fine – высокое
// позиция и характеристики камеры
EYEx EYEy EYEz // Точка камеры
VIEWx VIEWy VIEWz // Точка наблюдения (это центр бокса, при изначальной установки камеры)
UPx UPy UPz // вектор вверх, может быть не перпендикулярен вектору (EYE, VIEW)
ZN ZF // ближняя и дальняя граница видимости камеры
SW SH // ширина и высота матрицы камеры (через неё пускаются лучи), матрица расположение на расстояние ZN от точки EYE по направлению на точку VIEW
```

Программа должна работать с глубинами трассировки 1, 2 и 3 (как минимум), если реализация не поддерживает глубину трассировки, указанную в файле, то необходимо изменить глубину трассировки на ближайшую поддерживаемую и сообщить пользователю об этом при открытии.

UP вектор может быть не перпендикулярен вектору (EYE, VIEW). В этом случае (а лучше это делать всегда, а не проверять). Необходимо рассматривать вектор UP как приближённое направление вверх, а значит его можно (т.е. нужно) скорректировать по следующему алгоритму.

1. Вычисляем вектор Z камеры, как VIEW – EYE.
2. Вычисляем вектор RIGHT камеры как векторное произведение Z × UP.
3. Вычисляем вектор UP камеры, как RIGHT × Z.

Если SW / SH ≠ WIDTH / HEIGHT клиентской области, то необходимо скорректировать значение SW, то есть с сохранением вертикального угла обзора.

При отсутствии загруженного файла настроек рендеринга или при нажатии кнопки **Init**, необходимо поставить камеру в начальное положение и установить её характеристики в начальные значения:

1. Вычислить габаритный бокс сцены (минимальный бокс, расширенный на 5% относительно центра) и рассчитать центр бокса (CENTERx, CENTERy, CENTERz). Источники не входят в расчёт бокса.
2. Точка VIEW – это центр бокса (CENTERx, CENTERy, CENTERz).
3. UP-вектор положить (0, 0, 1), т.е. ось Z мировой системы координат направлена вверх, при начальном ракурсе наблюдения.
4. Камеру следуют поместить в точку EYEy = CENTERy, EYEz = CENTERz. А точку EYEx выбрать минимальную (то есть в сторону минус бесконечности), такая, чтобы ближайшая грань габаритного бокса к точке была видна из неё под вертикальным углом 30 градусов.
5. Положить ZN равное **половине** расстояния до передней стенки бокса, т.е. (MINx – EYEx) / 2. Вычислить ZF = MAXx – EYEx + (MAXx – MINx) / 2, то есть расстояние до дальней от камеры плоски бокса + половину длины бокса вдоль оси X.
6. При этом вычисляются SW и SH, так чтобы соотношение было равно отношению ширины к длине клиентской области приложения, при этом площадка SW × SH находится на расстояние ZN и при этом бокс вписан в пирамиду видимости.

Управление камерой

Необходимо реализовать следующее управление:

1. Изменение ZN (необходимо не добавлять или уменьшать DELTA, а умножать или делить на коэффициент, это гарантирует, что ZN не станет отрицательным). Соответствует изменению фокусного расстояния объектива. (колёсиком, как в задаче Wireframe)
2. Перемещение камеры вдоль линии EYE – VIEW. Соответствует перемещению камеры от объекта и к объекту наблюдения. (тут плюс/минус дельта, умноженная на единичный вектор) (колёсиком с зажатой клавишей CTRL)
3. Перемещения камеры вдоль **осей** X, Y **камеры** (стрелками на клавиатуре)
4. Поворот камеры вокруг **точки VIEW** (мышью, как в задаче Wireframe).
5. При изменении размеров окна необходимо пересчитать SW, чтобы соотношение SW к SH совпало с отношением WIDTH к HEIGHT клиентской области окна.

Итого: Сцену не трогаем, при управлении изменяем только характеристики и настройки камеры.
По желанию: Можно также реализовать режим полёта камеры, когда при перемещении камеры вдоль оси VIEW – EYE, можно кроме точки EYE перемещать ещё точку VIEW, а также наклоны камеры, т.е. изменение вектора UP, в этом случае управление можно взять из какой-либо игры.

Замечание: Важно то, чтобы конкретная сцена с привязанной к ней файлом настроек рендеринга открывалась у всех студентов **одинаково** (при равных размерах окна)! В отличии от задачи Wireframe в этой задаче бордюров быть не должно.

Особенности алгоритма рендеринга

Через каждый пиксель пускается луч в сцену. Если луч не пересекает ни одного примитива, то ставится цвет фона. Иначе находится ближайшая к камере точка пересечения луча с примити-

вами – точка P. Далее выпускается отражённый луч и ищется его ближайшее пересечение с примитивами сцены (пусть точка Q) и т.д. до *Depth - 1* отражений. Затем производится сборка интенсивности, начиная с самой последней найденной точки. Сбор ведётся три раза: по R, по G и по B (но трассировать три раза лучше не надо, достаточно одного раза).
В каждой точке рассчитывается освещённость с учётом видимости источников (т.е. необходимо выпускать луч из точки в направлении источника и проверять не пересекается ли он с каким-либо объектом сцены, если пересекается, то данный источник в данной точке не виден – в тени) и интенсивности, пришедшей с отражённого луча. Если на каком-то шаге отражённый луч не попадает в примитив, то считается, что с отражённого луча приходит фоновое значение (т.к. как будто снаружи есть равномерный однотонный источник света). Если глубина трассировки не позволяет выпустить ещё один отражённый луч, то считаем, что с луча приходит 0. Полученное интенсивности для точки P значение используется для закраски пикселя.
Сначала насчитываются 3 массива вещественных значений: по R, по G и по B. Находится **ОБЩИЙ** максимум, и он приводится к диапазону [0, 1]. Выполняется гамма-коррекция (чем больше гамма, тем светлее) и все три значения переводятся в диапазон [0, 255]. **Не забывать про "+0.5" при округлении.**
Глобальная освещённость:

$$I = I_a k_a + \sum_{i=1}^{n} V_i f_{att,i} I_i \left[k_d(\bar{N}, \bar{L}_i) + k_s(\bar{N}, \bar{H}_i)^{pow} \right] + k_s I_r .$$

Объяснение значений термов приведено в лекциях.
Ослабление света от источника из-за расстояния рассчитывать по формуле **fatt(d) = 1 / (1+d)**. **Не надо применять ослабление в зависимости от расстояния от точки P до камеры и при учёте интенсивности, пришедшей с направления отражённого луча.**

В процессе рендеринга пользователь должен видеть прогресс в statusbar'е или другим способом (прогресс рассчитывать, как процент уже рассчитанных пикселей). Также приложение не должно становиться белым при переключении по Alt+Tab (при скрытии окна программы другим приложением) во время рендеринга сцены, т.е. расчёт изображения в отдельном потоке.

Замечание: При расчётах необходимо использовать алгоритмы пересечения луча с примитивами из лекций. Расчёт пересечения луча с четырёхугольником необходимо производить как пересечение луча с двумя треугольниками.

Для ускорения расчёта изображения все поверхности рассматриваются как односторонние. Для сферы нормаль направлена из центра наружу, а для треугольников и четырёхугольников, направление нормали определяется по следующему правилу, основанному

на порядке обхода вершин:

1 – 4

| |

2 – 3

В этом случае нормаль из экрана, т.е. математический обход (номерами показан порядок вершин).

В этом случае четырёхугольник будет виден, если на него смотреть (т.е. для лучей идущих из камеры).

Для треугольников тоже самое, только вершины три.

А в случае ниже четырёхугольник не виден, т.к. нормаль от нас.

1 – 2

| |

4 – 3

Если трёхмерная поверхность замкнутая, тогда всё будет работать физически корректно, но расчёты ускорятся, т.е. в процессе поиска ближайшей точки пересечения с лучом, неоходимо рассматривать только треугольники и четырёхугольники, скалярное произведение нормали которых с направлением трассировочного луча отрицательное, т.е. луч и нормаль направлены навстречу друг другу.

Для сфер нужно рассчитать нормаль в точке пересечения и проигнорировать точку пересечения в которой скалярное произведение неотрицательное.

Необходимо так же учитывать, что теневые лучи, с помощью которых определяется видимость источника в данной точке поверхности, идут от источника к точке поверхности, а не наоборот.

Для упрощения отладки, имеет смысл рисовать невидимые треугольники и четырёхугольники в проволочной модели (в том числе те, которые сгенерированы для сфер) отдельным цветом, например, серым, тогда как видимые диффузным цветом фигуры, чтобы определить виден данный треугольник или четырёхугольник или нет, достаточно проверить порядок обхода вершин после перехода в систему координат камеры (или можно после применения всех матриц, т.е. после применения матрицы проецирования).

На лучшее решение: Желающие могут сделать режим расчёта пикселей в псевдослучайном порядке с одновременным отображением их поверх проволочной модели пока они не заполнят всё изображение. В этом случае минимальную и максимальную интенсивности можно рассчитать, скажем, после расчёта 10% пикселей и после этого можно уже отображать пиксели (или переотобразить). Сделать интерактивное (т.е. не 24 **частичных** кадра в секунду, а скажем 2-3 хотя бы, чтобы как-то всё шевелилось) управление ракурсом в режиме случайного расчёта пикселей (т.е. чтобы пользователь видел сразу рассчитанное изображение хотя бы частично), при этом минимум максимум можно хранить с предыдущих кадров и как-то разумно его адаптировать. Но это должен быть дополнительный режим, т.е. режим с проволочной моделью должен быть. Можно сначало грубо рассчитывать потом досчитывать более мелко.

Качество расчёта изображения

Необходимо дать пользователю возможность изменять следующие параметры в диалоге настроек рендеринга (т.е. тот, который показывается при нажатии кнопки **Settings**):

1. Цвет фона (с помощью диалога выбора цвета)
2. Значения gamma (0 до 10)
3. Глубина трассировки (обязательные значения 1, 2, 3, остальные по желанию, для корректно написанного алгоритма разницы нет)
4. Качество расчёта изображения (значение quality)

Качество расчёта изображения может быть грубое, нормальное или высокое. В первом случае на каждый 4 пикселя изображения пускается 1 луч (через точку между пикселями), а рассчитанное значение помещается в каждый из четырёх пикселей. Во втором случае на 1 пиксель пускается 1 луч (через центр пикселя). В последнем случае через каждый пиксель пускается 4 луча (через центр каждой четвертинки пикселя), а в пиксель помещается усреднённое значение.

Тестовые данные

В проекте должен располагаться как минимум один файл сцены, один файл настроек рендеринга **соответствующей** ей.

Внимание: Сцена, которую вы положите в папку Data должна быть достаточно сложной, а именно: сцена не должна быть набором нескольких несвязанных примитивов в пространстве, т.е. при расчёте должно быть видно множество отражений объектов друг на друге. Причём рассчитанные изображения должны отличаться для глубин трассировки 1, 2 и 3, иначе это несложная сцена.