Задача 5 – ICGRaytracing.

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 реалистичное 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. Переход в режим рендеринга сцены [**Render**]
8. Сохранение изображения в файл [**Save image**]

Кнопки **Select view** и **Render** работают в исключающем режиме – нажатие одной кнопки выключает другую (Radio buttons). Когда пользователь успешно загружает сцену, кнопка **Select view** оказывается нажатой. В этом режиме пользователь имеет возможность выбрать (изменить) ракурс камеры и поменять настройки рендеринга. Когда пользователь нажимает кнопку **Render** запускается процесс рендеринга, при этом необходимо заблокировать выбор ракурса на время расчёта изображения (если это не интерактивный режим – см. раздел на лучшее решение). После окончания процесса рендеринга кнопка **Render** остается нажатой, а кнопка **Select view** отжатой, при этом пользователь видит рассчитанное изображение. Пользователь может нажать кнопку **Select view**, при этом кнопка **Render** отжимается, пользователю вновь показывается проволочная

модель сцены (в том ракурсе, который был до рендеринга) и пользователь снова имеет возможность выбрать другой ракурс.

Обратите 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.

В режиме **Select view** примитивы отображаются в проволочном виде (как и в задаче Wireframe), параметры отображения выбираете сами. Проволочная модель должна рассчитываться с использованием матричной арифметики, как и в задаче Wireframe, то есть все точки умножаются только на одну итоговую матрицу. Не забывайте про клиппирование – оно должно быть реализовано.

Формат файла сцены

Файл сцены содержит описание объектов сцены, описание источников и описание рассеянного света. Другие 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 один 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 красной, зелёной и синей составляющей света.
- **KSr, KSg, KSb** – коэффициенты зеркального отражения для красной, зелёной и синей составляющей 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.

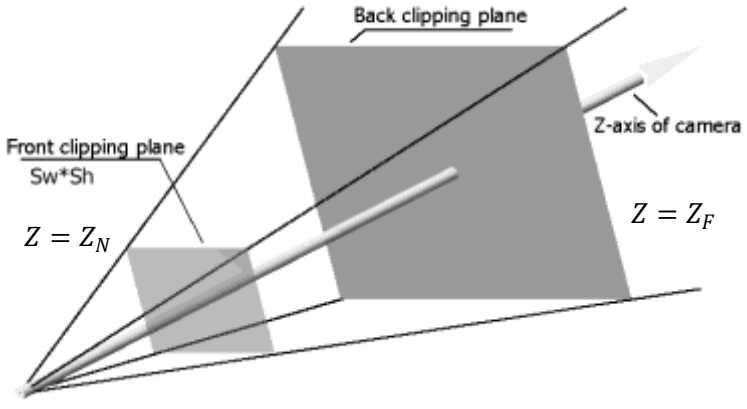
Предложение: В одной из сцен можете использовать набор поверхностей вращения из задачи Wireframe, необходимо лишь экспортировать сетку в виде четырёхугольников.

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**). Протестируйте, что после загрузки такого файла сцены 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 для начальной 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 файла. 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 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 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 по следующему алгоритму:

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 минимально (то есть в сторону минус бесконечности) такой, 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 их соотношение было равно отношению ширины к высоте клиентской области приложения, при этом площадка SW×SH находится на расстояние ZN, бокс вписан в пирамиду видимости.

Управление камерой

Необходимо реализовать следующее управление камерой:

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 (удобно делать колёсиком мышки).
2. Перемещение камеры вдоль линии EYE – VIEW. Соответствует перемещению камеры от объекта и к объекту наблюдения (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.

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 примитивами – 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 вычисление интенсивности, начиная с самой последней найденной точки. Вычисление выполняется три раза: по R, по G и по B компонентам (но трассировать три раза не нужно, достаточно одного раза). В

каждой точке рассчитывается освещённость с учетом интенсивности, пришедшей с отражённого луча, и 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 – он в тени). Если на каком-то шаге отражённый луч не попадает ни в один из примитивов, то считается, что с отражённого луча приходит фоновое значение (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 (нулевое значение интенсивности света). Полученное для точки P значение интенсивности используется для закраски пикселя. Для этого сначала насчитываются 3 массива вещественных значений (по R, по G и по B) для всех пикселей. Затем находится **ОБЩИЙ** максимум, и он приводится к диапазону [0, 1]. Затем выполняется гамма-коррекция (чем больше гамма, тем светлее) и затем все три значения переводятся в целое число в диапазоне [0, 255]. **Не забывать про "+0.5" при округлении.**

Формула глобальной освещённости:

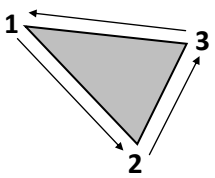
$$I = k_a I_A + \sum_{i=1}^{n} V_i f_{att,i} I_i [k_d (N \cdot L_i) + k_s (R_i \cdot V)^{pow}] + k_s f_{att,s} I_r$$

Объяснение значений термов приведено в лекциях.

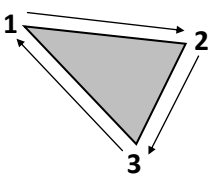
Ослабление света от источника из-за расстояния рассчитывать по формуле $f_{att}(d) = 1 / (1+d)$. **Не надо применять ослабление от точки P до камеры и при учёте интенсивности фонового света**, **пришедшей с направления отражённого луча** (больше нет пересечений).

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

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:



Камера смотрит в экран (вектор нормали камеры направлен из глаза наблюдателя в экран) и видит треугольник (многоугольник). Возможны два варианта



обхода вершин. При порядке обхода вершин как на рисунке слева нормаль поверхности многоугольника направлена из экрана в камеру (противонаправлена с нормалью камеры) и треугольник (многоугольник) будет виден. При порядке обхода вершин как на рисунке справа нормаль поверхности многоугольника направлена из камеры в экран (сонаправлена с нормалью камеры) и треугольник (многоугольник) виден не будет.

Если трёхмерная поверхность замкнутая, тогда всё будет работать физически корректно, но расчёты ускорятся, поскольку каждый многоугольник обрабатывается один раз, а не два раза (с одной и другой стороны). В процессе поиска ближайшей точки пересечения с лучом необходимо рассматривать только треугольники и четырёхугольники, для которых скалярное произведение нормали 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/четырёхугольник или нет, достаточно проверить порядок обхода вершин после перехода в систему координат камеры (после применения всех матриц, включая матрицу 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 после частичного расчета (например, после расчёта 10% пикселей) и в дальнейшем до окончания рендеринга параметры масштабирования интенсивности не менять, а уже нарисованные пиксели отобразить заново. По окончанию рендеринга сделать финальный расчет параметров масштабирования интенсивности и перерисовать все изображение. Можно сделать интерактивное (скажем 2-3 кадра в секунду, чтобы как-то всё шевелилось) управление ракурсом в режиме случайного расчёта пикселей (т.е. чтобы пользователь видел рассчитанное изображение хотя бы примерно в процессе изменения ракурса), при этом минимум/максимум интенсивности можно как-то разумно адаптировать в момент перехода к следующему кадру по уже вычисленным пикселям текущего кадра. Но это должен быть дополнительный режим, т.е. режим с проволочной моделью должен быть в любом случае.

Также можно сделать адаптивный расчет – сначала рассчитывать на грубой сетке, а потом досчитывать на мелкой сетке. На рисунках ниже показан такой режим вычисления и закраски пикселей в области 4x4:

Шаг 1:

Вычислено:

1			

Нарисовано:

1	*1*	*1*	*1*
1	*1*	*1*	*1*
1	*1*	*1*	*1*
1	*1*	*1*	*1*

Шаг 2:

Вычислено:

1		*2*	
3		*4*	

Нарисовано:

1	*1*	*2*	*2*
1	*1*	*2*	*2*
3	*3*	*4*	*4*
3	*3*	*4*	*4*

Шаг 3:

Вычислено:

1	*2*	*3*	*4*
5	*6*	*7*	*8*
9	*10*	*11*	*12*
13	*14*	*15*	*16*

Нарисовано:

1	*2*	*3*	*4*
5	*6*	*7*	*8*
9	*10*	*11*	*12*
13	*14*	*15*	*16*

Сначала вычисляется по одному пикселю в каждой области 4x4 и для всех пикселей области рисуется цвет вычисленного пикселя. Затем внутри области 4x4 вычисляются 3 дополнительных пикселя с шагом 2x2 и вычисленные значения заполняют область 2x2. И на последнем этапе вычисляются все оставшиеся пиксели.

Качество расчета изображения

Необходимо дать пользователю возможность изменять следующие параметры в диалоге настроек рендеринга (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 каждые 4 пикселя изображения (квадрат 2x2 пикселя) 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 и один файл настроек рендеринга, **соответствующей** этой сцене.

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.