Содержание

1 Введение

Для создания реалистичных изображений необходимо использование алгоритмов, которые могут учитывать не свет, напрямую исходящий из источника света, но и непрямое освещение, проявляющееся в зеркальном и диффузном отражениях, а также в преломлении света и наличии теней. Подобные алгоритмы называются алгоритмами глобального освещения.

1.1 Обзор существующих алгоритмов глобального освещения

Большинство подобных алгоритмов можно разделить на две группы - в основе одних лежит точечная выборка (методы различных трассировок лучей), а в основе других - метод конечных элементов.

Методы трассировки лучей

Методы прямой трассировки лучей Источники света излучают лучи, которые распространяются по сцене, могут быть отражены или преломлены. Когда луч попадает в "экран", то устанавливается цвет соответствующего пикселя. Такой подход чрезвычайно медленен, так как учитывает все лучи, даже те, что не попадают на экран. Алгоритмы позволяет рассчитывать прямой, отраженный, преломленный свет, каустику (огибающая группы лучей или преломленная от изогнутых объектом на другой поверхности), некоторые модицикации - диффузное отражение. Проблемой алгоритмов является низкая скорость работы и ярко выраженный шум. Примерами являются простая прямая трассировка лучей, трассировка путей.

Обратная трассировка лучей Идея алгоритмов основывается на том, что распространение света может быть отслежено от наблюдателя до источника света, что позволяет не учитывать лучи, которые не попадают на экран. Быстродействие алгоритмов значительно выше, но в базовом варианте нет возможности рассчитать полное освещение

сцены, а алгоритмы, учитывающие это также медленны и шумны. Примеры - простая обратная трассировка, аналог трассировки путей.

Методы конечных элементов

Методы основаны на равновесии обмена света между объектами сцены (по сути законе сохранения энергии) и разбиении поверхностей на маленькие участки, которые считаются локально плоскими. Освещенность находится решением системы линейных уравнений, описывающих обмен энергии между участками. При использовании все меньших участков результат будет стремиться к реальной физической модели. Алгоритмы является довольно медленным, так как производится расчет для множества маленьких участков поверхностей. Также подобные алгоритмы имеют проблемы с представлением резких теней.

Метод фотонных карт

Данный метод использует другой подход, заключающийся в отделении информации об освещенности от геометрии сцены, сохраняя ее в отдельной структуре данных - “фотонной карте”, которая содержит информацию о столкновениях фотонов, излученных источниками света, с поверхностями. Разделение фотонной карты и геометрии сцены позволяет не только упростить расчеты, но и получить возможность эффективно визуализировать сцены со сложной геометрией. Алгоритм, объединяющий метод фотонных карт и обратную трассировку лучей, позволяет эффективно получать реалистичные результаты, которые также близки к физической модели. Алгоритм состоит из двух частей: сначала строится фотонная карта, получаемая из трассировки фотонов по сцене и занесения их взаимодействия с поверхностями в карту, далее освещение рассчитывается обратной трассировкой лучей, использующей информацию, предоставляемую фотонной картой.

1.2 Постановка задачи

Необходимо разработать программу, позволяющую получать реалистичную визуализацию трехмерных сцен с помощью метода фотонных карт.

Можно выделить следующие задачи:

- Реализация модели освещения объектов сцены.
- Реализация алгоритма обратной трассировки лучей.
- Реализация алгоритмов построения фотонных карт.
- Реализация алгоритма визуализации фотонных карт.
- Реализация загрузки описания сцены из внешних файлов.
- Реализация графического интерфейса.

2 Конструкторский раздел

2.1 Метод фотонных карт

Интерес разработки представляла сама реализация метода фотонных карт, но при этом можно отметить многочисленные преимущества этого метода.

Метод является алгоритмом глобального освещения и приближенно решает уравнение рендеринга. При этом фотонные карты разъединяют информацию об освещении и геометрию сцены, что позволяет рассчитывать части уравнения рендеринга отдельно и хранить в различных картах, что повышает удобство и эффективность расчетов. И это означает, что при передвижении наблюдателя необходимо будет лишь провести сбор освещенности с фотонной карты на этапе рендеринга, а не выполнять алгоритм полностью каждый раз. Также метод позволяет визуализировать преломление света через прозрачные объекты с образованием каустики, что не могут рассчитать методы трассировки лучей.

Общая схема алгоритма

1. Излучение фотонов.
2. Трассировка фотонов и построение фотонной карты.
    - Определение состояния фотона после столкновения - рассеяние, зеркальное отражение или поглощение.
    - Сохранение столкновений фотонов с диффузными поверхностями.
3. Рендеринг.
    - Сбор освещенности с фотонной карты при трассировке лучей.
    - Визуализация вспомогательных карт (напр. каустики)

Излучение фотонов

Для точечного источника излучение каждого фотона происходит в одном из случайных направлений, равномерно распределенных по сфере от положения источника. Фотоны из квадратного источника излучаются в случайном направлении, ограниченном полусферой. При этом удобно использовать выборку с отклонением для генерации направлений. Для точечного источника измеряются значения трех случайных величин, равномерно распределенных на $[0, 1]$. Если точка, составленная из этих значений попадает внутрь единичной сферы, то фотон излучается, так происходит пока не будут излучены все фотоны.

Другим способом излучения является генерация двух случайных значений в интервале $[0, 1]$ и применению к ним UV-преобразования для получения сферы. Решено было использовать этот метод как более простой и эффективный с точки зрения генерации псевдослучайных чисел.

$$\begin{aligned} u, v &\leftarrow random\,[0, 1] \\ z &= 2u - 1 \\ x &= \sqrt{1 - z^2}\, cos(2\pi v) \\ y &= \sqrt{1 - z^2}\, sin(2\pi v) \end{aligned} \tag{2.1}$$

Необходимо излучить n фотонов из источников света. Для излучения из нескольких источников общее количество фотонов делится пропорционально яркости источника - n_i, $i \in [1,$ количество источников$]$.

Для каждого источника излучить n_i фотонов:

1. $u, v \leftarrow$ случайное число в диапазоне $[0, 1]$
2. $x, y, z \leftarrow$ uv-преобразование(u, v)
3. $\vec{d} \leftarrow (x, y, z)$ - направление движение фотона.
4. $\vec{o} \leftarrow$ координаты источника - начальные координаты фотона

5. $\vec{p} \leftarrow$ яркость источника

Трассировка фотонов

Излученные фотоны прослеживаются методом, похожим на трассировку лучей. Фотон перемещается до столкновение с объектом сцены. При столкновении фотон может быть отражен, преломлен или поглощен. Решение производится эффективным методом "Русской рулетки".

Рассмотрим материал с коэффициентами диффузного отражения d, зеркального - s, пропускания - t, $d + s + t \leq 1$. Также выберем равномерно распределенную случайню величину $\xi \in [0, 1]$.

Можно принять следующие решения:

$$\begin{array}{rrrr} \xi \in & [0, d] & \rightarrow & \text{диффузное отражение} \\ \xi \in & (d, d+s] & \rightarrow & \text{зеркальное отражение} \\ \xi \in & (d+s, d+s+t] & \rightarrow & \text{преломление} \\ \xi \in & (d+s+t, 1] & \rightarrow & \text{поглощение} \end{array} \tag{2.2}$$

Процесс трассировки, задается глубина трассировки по истечению которой процесс прекращается:

1. Найти ближайшее пересечение фотона с объектом сцены.
2. Получить результат столкновение, с помощью метода "Русской рулетки"
   (a) **Рассеивание** - добавить фотон к фотонной карте, изменить направление, цвет и координаты, продолжить трассировку.
   (b) **Отражение, преломление** - аналогично рассеиванию, но не добавлять фотон к карте.
   (c) **Поглощение** - добавить фотон к карте, завершить трассировку.

При расчете каустики к фотонной карте добавляются только фотоны, испытавшие зеркальное отражение или преломление и столкнувшиеся с рассеивающей поверхностью.

Построение фотонной карты

Фотоны сохраняются в карте только когда они сталкиваются с незеркальными поверхностями, так как это не дает полезной информации для рендеринга - это можно рассчитать трассировкой лучей на этапе визуализации.

При каждом взаимодействии фотона с поверхностью в карту заносятся координаты фотона, мощность и направление.

Для более эффективного расчета фотонная карта разделяется на несколько подкарт:

- **глобальная карта** содержит взаимодействия со всеми диффузными поверхностями,

- **карта каустики** содержит фотоны, которые претерпели хотя бы одно зеркальное отражение или преломление перед взаимодействием с рассеивающей поверхностью.

Оценка яркости

Фотонная карта может быть рассмотрена как представление поступающего потока энергии, таким образом для расчета энергетической яркости необходимо проинтегрировать эту информацию.

$$L_r(x, \vec{\omega}) = \int_{\Omega_x} f_r(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}')(\vec{n_x} \cdot \vec{\omega}') \, \mathrm{d}\vec{\omega}', \tag{2.3}$$

где L_r - отраженная светимость в x в направлении $\vec{\omega}$. Ω_x - сфера поступающих направлений, f_r - BRDF (bidirectional reflectance distribution function, двулучевая функция отражающей способности) в x, L_i - поступающая яркость. Для нахождения значения интеграла необходимо

знать поступающую энергетическую яркость, для этого используется информация из фотонной карты.

$$L_i(x, \vec{\omega'}) = \frac{\mathrm{d}^2\Phi_i(x, \vec{\omega'})}{(\vec{n}_x \cdot \vec{\omega'})\,\mathrm{d}\vec{\omega}'_i\,\mathrm{d}A_i} \tag{2.4}$$

Теперь можно выразить главный интеграл как

$$\begin{aligned} L_r(x, \vec{\omega}) &= \int_{\Omega_x} f_r(x, \vec{\omega'}, \vec{\omega}) \frac{\mathrm{d}^2\Phi_i(x, \vec{\omega'})}{(\vec{n}_x \cdot \vec{\omega'})\,d\omega'_i\,\mathrm{d}A_i} (\vec{n_x} \cdot \vec{\omega'})\,\mathrm{d}\omega'_i \\ &= \int_{\Omega_x} f_r(x, \vec{\omega'}, \vec{\omega}) \frac{\mathrm{d}^2\Phi_i(x, \vec{\omega'})}{\mathrm{d}A_i}. \end{aligned} \tag{2.5}$$

Приближенное значение поступающего потока рассчитывается из фотонной карты, находя n фотонов, расположенных наиболее близко к x. Каждый фотон p имеет мощность $\Delta\Phi_p(\vec{\omega}_p)$. При предположении что фотон пересекает поверхность в x получаем:

$$L_r(x, \vec{\omega}) \approx \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \frac{\Delta\Phi_p(x, \vec{\omega}_p)}{\Delta A} \tag{2.6}$$

Расчет может быть представлен как процесс расширение сферы в x пока в нее не попадет n фотонов. Вводя допущение, что поверхность вблизи x локально плоская можно найти $\Delta A = \pi r^2$, где r - радиус сферы, т.е. расстояние между x и самым дальним фотоном.

В результате:

$$L_r(x, \vec{\omega}) \approx \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) \tag{2.7}$$

Точность зависит от количества используемых фотонов и сфера может включать в себя лишние фотоны в углах и на краях объектов, но размер ошибочных регионов уменьшается с ростом числа фотонов.

Если устремить число фотонов в бесконечность, то

$$\lim_{N \to +\infty} \frac{1}{\pi r^2} \sum_{p=1}^{N} f_r(x, \vec{\omega}_p, \vec{\omega}) \Delta\Phi_p(x, \vec{\omega}_p) = L_r(x, \vec{\omega}) \tag{2.8}$$

Из уравнения следует, что бесконечное количество фотонов точно представит распределение яркости в сцене. Таким образом можно получить хорошую оценку, используя достаточное количество фотонов.

Выборка по значимости Помимо простого суммирования энергитических яркостей фотонов можно умножать яркость каждого фотона на вес, определяемый из близости фотона к точке пересечения.

$$w(x_p) = 1 - \frac{x_p}{r}$$

Визуализация

Каждый пиксель изображения рассчитывается с помощью трассировки лучей, использующей информацию из фотонной карты. Каустика будет рассмотрена отдельно.

Яркость, получаемая каждым лучом равна

$$L_o(x, \vec{\omega}) = L_e(x, \vec{\omega}) + L_r(x, \vec{\omega}) \tag{2.9}$$

L_e - излучаемая поверхностью энергетическая яркость, а L_r находится из

$$L_r(x, \vec{\omega}) = \int_\Omega f_r(x, \vec{\omega}', \vec{\omega}) L_i(x, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}) \mathrm{d}\omega_i' \tag{2.10}$$

BRDF f_r может быть разделена на сумму двух компонент: зеркальной f_r^s и диффузной f_r^d,

$$f_r(x, \vec{\omega}', \vec{\omega}) = f_r^s(x, \vec{\omega}', \vec{\omega}) + f_r^d(x, \vec{\omega}', \vec{\omega}). \tag{2.11}$$

Поступающую яркость можно разделить на 3 компоненты:

- $L_i^l(x, \vec{\omega}')$ - прямая освещенность от источников света,
- $L_i^c(x, \vec{\omega}')$ - непрямая освещенность через зеркальное отражение и преломление,
- $L_i^d(x, \vec{\omega}')$ - непрямая освещенность через диффузное отражение.

$$L_i(x, \vec{\omega}') = L_i^l(x, \vec{\omega}') + L_i^c(x, \vec{\omega}') + L_i^d(x, \vec{\omega}') \tag{2.12}$$

Теперь L_r можно разделить на 4 компоненты.

$$L_r(x, \vec{\omega}) = I_1 + I_2 + I_3 + I_4 \tag{2.13}$$

$$I_1 = \int_\Omega f_r(x, \vec{\omega}', \vec{\omega}) L_i^l(x, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}) \, \mathrm{d}\omega_i' \tag{2.14}$$

$$I_2 = \int_{\Omega_x} f_r^s(x, \vec{\omega}', \vec{\omega})(L_i^c(x, \vec{\omega}') + L_i^d(x, \vec{\omega}'))(\vec{\omega}' \cdot \vec{n}) \, \mathrm{d}\omega_i' \tag{2.15}$$

$$I_3 = \int_{\Omega_x} f_r^d(x, \vec{\omega}', \vec{\omega}) L_i^c(x, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}) \, \mathrm{d}\omega_i' \tag{2.16}$$

$$I_4 = \int_{\Omega_x} f_r^d(x, \vec{\omega}', \vec{\omega}) L_i^d(x, \vec{\omega}')(\vec{\omega}' \cdot \vec{n}) \, \mathrm{d}\omega_i' \tag{2.17}$$

- I_1 (2.14) представляет вклад в отражаемую яркость от прямой освещенности и может быть рассчитан методами трассировки лучей, можно повысить скорость рассчета, введя карту теневых фотонов.
- I_2 (2.15) - зеркальное и глянцевое отражение, эта часть хорошо рассчитывается методами Монте-Карло, т.е. трассировкой лучей, для оптимизации можно использовать выборку по значимости.
- I_3 (2.16) - каустика, рассчитывается из фотонной карты каустики с большим числом фотонов.
- I_4 (2.17) - свет хотя бы раз отраженный от диффузной поверхности, дает более точную освещенность и цветовое заполнение (color bleeding), рассчитывается трассировкой лучей, собирающей освещенность с фотонной карты.

2.2 Поиск пересечений с объектами

Луч задается уравнением $\vec{r} = \vec{o} + t\vec{d}$, $t \in \mathbb{R}$, $t \geq 0$, где $\vec{o}$ - начальная точка луча, $\vec{d}$ - единичный вектор, задающий направление, что эквивалентно

$$\begin{cases} x(t) = x_o + tx_d \\ y(t) = y_o + ty_d \\ z(t) = z_o + tz_d \end{cases} \tag{2.18}$$

Пересечение луча со сферой

Сфера задается уравнением $(x_s - x_c)^2 + (y_s - y_c)^2 + (z_s - z_c)^2 = r^2$, где $\vec{c} = (x_c, y_c, z_c)$.

Подставим вместо x_s, y_s, z_s соответствующие уравнения луча.

$$(x_o + tx_d - x_c)^2 + (y_o + ty_d - y_c)^2 + (x_o + tx_d - x_c)^2 = r^2$$

После раскрытия скобок и несложных преобразований получаем квадратное уравнение $At^2 + By + C = 0$, с

$$A = x_d^2 + y_d^2 + z_d^2$$

$$B = 2(x_d(x_o - x_c) + y_d(y_o - y_c) + z_d(z_0 - z_c))$$

$$C = (x_o - x_c)^2 + (y_o - y_c)^2 + (z_o - z_c)^2 - r^2$$

$A = 1$, так как $||\vec{d}|| = 1$.

Если дискриминант уравнения меньше нуля, то луч не пересекает сферу, в ином случае наименьший положительный корень будет расстоянием от $\vec{o}$ до точки пересечения.

Для удобства коэффициенты уравнения можно вычислить сразу из векторов.

$$A = 1$$

$$B = 2(\vec{d} \cdot (\vec{o} - \vec{c}))$$

$$C = (\vec{o} - \vec{c})(\vec{o} - \vec{c}) - r^2$$

Пересечение луча с плоскостью

Точка $\vec{p}$ лежит в плоскости, если $\vec{n} \cdot (\vec{p} - \vec{q}) = 0$, где $\vec{n}$ - нормаль к плоскости, $\vec{q}$ - точка на плоскости.

Если $\vec{n} \cdot \vec{d} = 0$, то луч параллелен плоскоти и точки пересечения нет.

Подставим формулу, задающую луч.

$$\vec{n} \cdot (\vec{o} + t\vec{d} - \vec{q}) = 0$$

$$t = \frac{\vec{n} \cdot (\vec{q} - \vec{o})}{\vec{d} \cdot \vec{n}}$$

Если $t < 0$, то точки пересечения нет, в ином случае $\vec{o} + t\vec{d}$ - искомая точка пересечения.

Пересечение луча с треугольником

Пересечение можно найти, используя барицентрические координаты.

Точка находится на треугольнике, если она представима в виде $\vec{p} = a_1\vec{v_1} + a_2\vec{v_2} + a_3\vec{v_3}$, где $\vec{v_i}$, $i \in 1, 2, 3$ и $a_1 + a_2 + a_3 = 1$. Используя $a_1 + a_2 + a_3 = 1$ преобразуем уравнение - $a_2(\vec{v_2} - \vec{v_1}) + a_3(\vec{v_3} - \vec{v_1}) = \vec{p} - \vec{v_1}$.

Для более быстрого расчета можно спроецировать треугольник на какую-либо плоскость. Удобными для этого являются плоскости XY, XZ или YZ. Чтобы выбрать лучшую плоскость, из нормали треугольника опеределяем доминирующую ось и получаем наибольшую площадь проекции и соответственно точность. [1]

$$\vec{b} = \vec{v_3} - \vec{v_1}$$
$$\vec{c} = \vec{v_2} - \vec{v_1}$$
$$\vec{n} = \vec{c} \times \vec{b}$$
$$\vec{h} = \vec{o} + \frac{(\vec{v_1} - \vec{o}) \cdot \vec{n}}{\vec{d} \cdot \vec{n}}$$

$\vec{h}$ - точка пересечения луча с плоскостью треугольника.

Если X - доминантная ось, то u соответствует компоненте y, а v - z. Y: u - z, v - x. Z: u - x, v - y.

$$a_2 = \frac{\vec{b}_u\vec{h}_v - \vec{b}_v\vec{h}_u}{\vec{b}_u\vec{c}_v - \vec{b}_v\vec{c}_u}$$
$$a_3 = \frac{\vec{c}_v\vec{h}_u - \vec{c}_u\vec{h}_v}{\vec{b}_u\vec{c}_v - \vec{b}_v\vec{c}_u}$$

Далее тривиально определяется факт пересечения - оно отсутствует при любом из следующих условий: $a_2 < 0$, $a_3 < 0$, $a_2 + a_3 > 1$.

Для ускорения можно предрассчитать некоторые значения при добавлении треугольника к сцене. Заранее рассчитать можно $\vec{n}$, доминирующую ось k, соответствующие индексы для выбора компонент (u, v), а также

$$\begin{aligned}
n_u &= \frac{\vec{n}_u}{\vec{n}_k} \\
n_v &= \frac{\vec{n}_v}{\vec{n}_k} \\
n_d &= \frac{\vec{n} \cdot \vec{v_1}}{\vec{n}_k} \\
bn_u &= \frac{\vec{b}_u}{\vec{b}_u\vec{c}_v - \vec{b}_v\vec{c}_u} \\
bn_v &= \frac{-\vec{b}_v}{\vec{b}_u\vec{c}_v - \vec{b}_v\vec{c}_u} \\
cn_u &= \frac{\vec{c}_v}{\vec{b}_u\vec{c}_v - \vec{b}_v\vec{c}_u} \\
cn_v &= \frac{-\vec{c}_u}{\vec{b}_u\vec{c}_v - \vec{b}_v\vec{c}_u}
\end{aligned}$$

Тогда поиск пересечения примет вид

$$t = \frac{n_d - \vec{o}_k - n_u\vec{o}_u - n_v\vec{o}_v}{\vec{d}_k + n_u\vec{d}_u + n_v\vec{d}_v}$$

t - расстояние от $\vec{o}$ до точки пересечения.

$$\vec{h}_u = \vec{o}_u + t\vec{d}_u - \vec{a}_u$$
$$\vec{h}_v = \vec{o}_u + t\vec{d}_v - \vec{a}_v$$

$$a_2 = bn_u\vec{h}_v + bn_v\vec{h}_u$$
$$a_3 = cn_u\vec{h}_u + cn_v\vec{h}_v$$

Условия существования точки пересечения остаются теми же.

2.3 Взаимодействие луча с поверхностями

Исходный луч - $\vec{o}+t\vec{d}$, точка пересечения - $\vec{c}$, нормаль поверхности в точке пересечения - $\vec{n}$.

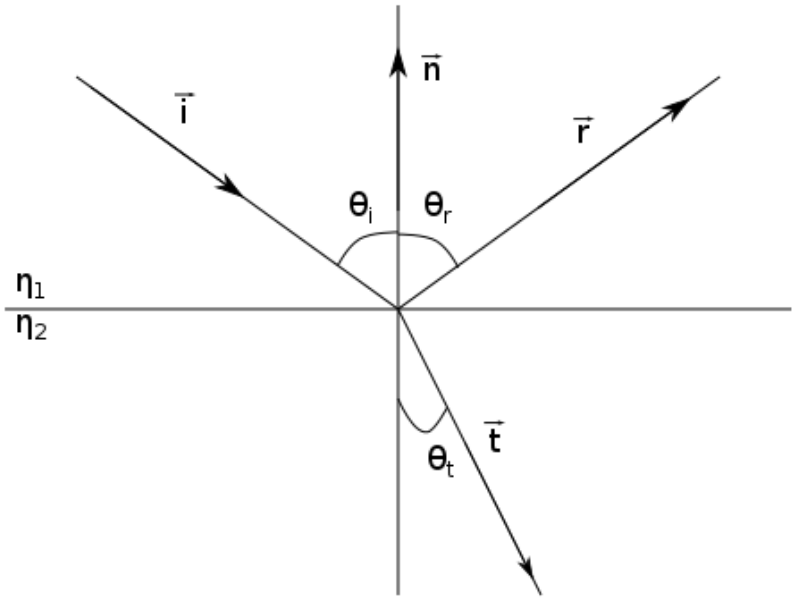


Рис. 1: Отражение и преломление

Для $\vec{v} \in \{\vec{i}, \vec{r}, \vec{t}\}$.

$$\begin{aligned} \vec{v}_\perp &= (\vec{v} \cdot \vec{n})\,\vec{n} \\ \vec{v}_\parallel &= \vec{v} - \vec{v}_\perp \end{aligned} \tag{2.19}$$

Зеркальное отражение

$$\theta_i = \theta_r$$
$$\vec{r} = \vec{r}_\perp + \vec{r}_\parallel = \vec{i}_\parallel - \vec{i}_\perp$$

Из (2.19) $\vec{r} = \vec{i} - 2(\vec{i} \cdot \vec{n})\,\vec{n}$.

Отраженный луч можно получить, как $\vec{c} + t\,(\vec{d} - 2\,(\vec{n} \cdot \vec{d})\,\vec{n})$.

Преломление

Известен абсолютный коэффициент преломления поверхности и коэффициент преломления снаружи. Относительный коэффициент преломления рассчитывается как

$$\eta = \frac{\eta_{\text{текущий}}}{\eta_{\text{поверхности}}}$$

$$\eta_1 \sin\theta_i = \eta_2 \sin\theta_t$$

$$\vec{t} = \frac{\eta_1}{\eta_2}\,\vec{i} - \left(\frac{\eta_1}{\eta_2}\cos\theta_i + \sqrt{1 - \sin^2\theta_t}\right)\,\vec{n}$$

$$\sin^2\theta_t = \left(\frac{\eta_1}{\eta_2}\right)^2 \sin^2\theta_i$$

$$\vec{t} = \frac{\eta_1}{\eta_2}\,\vec{i} - \left(\frac{\eta_1}{\eta_2}\cos\theta_i + \sqrt{1 - \eta^2\,(1 - \cos^2\theta_i)}\right)\,\vec{n}$$

Если $\sin^2\theta_t > 1$, то произошло полное внутреннее отражение.

Для луча: если $1 - r^2(1 - \vec{d} \cdot \vec{n}) < 0$, то произошло полное внутреннее отражение и луч дальше не трассируется, иначе находим направление преломленного

$$\vec{d'} = \eta\,\vec{d} - \left(\eta\,(\vec{d} \cdot \vec{n}) + \sqrt{1 - \eta^2(1 - \vec{d} \cdot \vec{n})}\right)\vec{n}$$

Рассеивание

Свет отражается во всех направлениях, сталкиваясь с рассеивающей поверхностью. Модель Ламберта вводит допущение, гласящее, что вероятность отражения в каждом направлении одинакова.

Тогда можно найти новое направление луча.

Даны две случайные, равномерно распределенные, величины $\xi_1 \in [0,1]$, $\xi_2 \in [0,1]$. В сферических координатах новое направление рассчитывается как

$$\vec{\omega}_d = (\theta, \phi) = (\arccos(\sqrt{\xi_1}),\ 2\pi\xi_2)$$

$$\vec{d}_x' = sin\theta\, cos\phi$$
$$\vec{d}_y' = sin\theta\, sin\phi$$
$$\vec{d}_z' = cos\theta$$

2.4 Обратная трассировка лучей

Трассировка лучей - простой и эффективный алгоритм, позволяющий легко отображать тени и зеркальные поверхности. Алгоритм основан на идее того, что траектория лучей света может быть отслежена от наблюдателя к источнику. Этот алгоритм используется в методе фотонных карт для построения финального изображения.

На вход алгоритму передается позиция наблюдателя, направление взляда, угол обзора, описание сцены - геометрия, материалы объектов и описание источников света. Изображение получается трассировкой лучей через каждый пиксель изображения и установкой, возвращаемого лучом цвета.

Для расчета цвета луча необходимо найти ближайшую точку пересечение с объектом сцены. Далее необходимо знать нормаль в данной точке поверхности $\vec{n}$ и BRDF f_r. Теперь можно рассчитать яркость в этой точке:

$$L_r(x, \vec{\omega}) = f_r(x, \vec{\omega}, \vec{\omega}') \frac{\vec{\omega}' \cdot \vec{n}}{||p - x||^2} V(x, p) \frac{\Phi_l}{4\pi}$$

Φ_l - мощность источника света, находящегося в точке p, $\vec{\omega}' = \frac{p-x}{||p-x||}$ - орт, направленный в сторону источника света, $V = 0$, если луч из x в p пересекает какой-либо объект, иначе $V = 1$ (x не в тени). Если же объект прозрачен, то V будет равна его коэффициенту пропускания.

Для зеркальных поверхностей луч отражается и яркость, возвращаемая отраженным лучом, суммируется с яркостью в данной точке.

Алгоритм

for all pixel ∈ pixels **do** ▷ Для каждого пикселя
  $ray \leftarrow$ луч из камеры через пиксель
  цвет пикселя $\leftarrow trace(ray)$
function TRACE(ray)
  Найти точку ближайшего пересечения луча с объектом сцены.
  if ∃ точка ближайшего пересечения **then**
    $point \leftarrow$ точка пересечения
    $normal \leftarrow$ нормаль поверхности в данной точке
    цвет пикселя $\leftarrow shade(point, normal)$
  else
    цвет пикселя $\leftarrow$ цвет фона (обычно черный)
function SHADE(point, normal)
  $color \leftarrow 0$ ▷ Цвет пикселя
  for all light ∈ lights **do** ▷ Для каждого источника света
    Испустить теневой луч из *point* к источнику света.
    if луч не пересекает объект сцены **then**
      $color \leftarrow color +$ прямое освещение от источника
  if Поверхность пересечения зеркальная **then**
    $color \leftarrow color + trace$(отраженный луч)
  if Поверхность пересечения пропускает свет **then**
    $color \leftarrow color + trace$(преломленный луч)
  return *color*

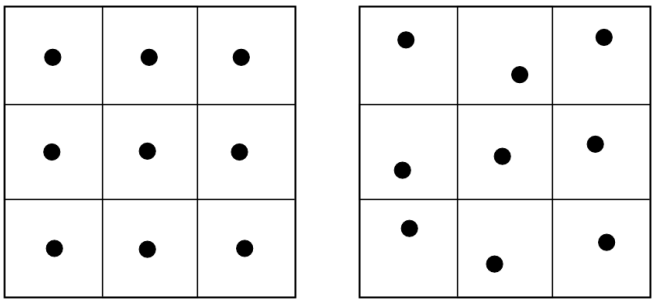
2.5 Сглаживание

Для получения более красивых изображений необходимо устранить ступенчатость - сгладить изображение.

В данной работе для этого применяется суперсэмплинг, который

представляет собой разбиение пикселя на подпикселя, трассировки лучей через подпиксели и взятие среднего арифметического цветом, возвращаемых лучами.

Для более качественного изображение применяется суперсэмплинг со сдвигом точек, через которые проходят лучи в случайном направлении.



(a) простой (b) со случайным сдвигом

Рис. 2: Суперсэмплинг

2.6 BRDF и модели освещения

BRDF - bidirectional reflectance distribution function, двулучевая функция отражающей способности. BRDF описывает как свет будет отражен от поверхности.

На практике вводятся различные упрощения, что позволяет рассчитать BRDF поверхности.

Модель освещения по Ламберту

Исходит из предположения, что поверхность является идеальной диффузной отражающей поверхностью (т.е. поверхность будет иметь одинаковую яркость по всем направлениям полусферы, окружающей поверхность).

Интенсивность может быть рассчитана по формуле:

$$I_d = k_d(\vec{L} \cdot \vec{N})\, i_m$$

k_d - коэффициент диффузного отражение материала, $\vec{L}$ - вектор, направленный из источника света в точку пересечения, $\vec{N}$ - нормаль поверхности в точке пересечения.

Модель освещения Фонга

Модель введена для описания глянцевых поверхностей.
Интенсивность описывается как

$$I = k_d(\vec{N} \cdot \vec{L})\, i_m^d + k_s(\vec{R} \cdot \vec{V})^\alpha\, i_m^s$$

$$\vec{R} = (\vec{L} \cdot \vec{N})\, \vec{N} - \vec{L}$$

$\vec{R}$ - отраженный от поверхности луч, k_s - коэффициент зеркального отражения, α - коэффициент блеска.

Модель освещения Блинна-Фонга

Блинн заменяет “дорогое” вычисление $\vec{R} \cdot \vec{V}$ на более быстрое $\vec{H} \cdot \vec{N}$, при этом получая результаты, более близкие к реальным. [2]

$$\vec{H} = \frac{\vec{L} + \vec{V}}{||\vec{L} + \vec{V}||}$$

3 Технологический раздел

3.1 Выбор языка программирования

Существует множество языков, обладающих различными возможностями. Для разработки данной программы был выбран язык программирования Haskell по следующим причинам:

1. Язык сочетает в себе сильную выразительность, эффективность и удобство построения абстракций.

2. Позволяет легко распараллеливать программы [3] [4] [5], что сильно пригождается при трассировке лучей.

3. Ленивые вычисления способствуют модульности программ. [6]

4. Мощная статическая система типов позволяет находить большое число ошибок на этапе компиляции.

5. Компилятор языка существует для всех популярных платформ. [7]

3.2 Структура программы

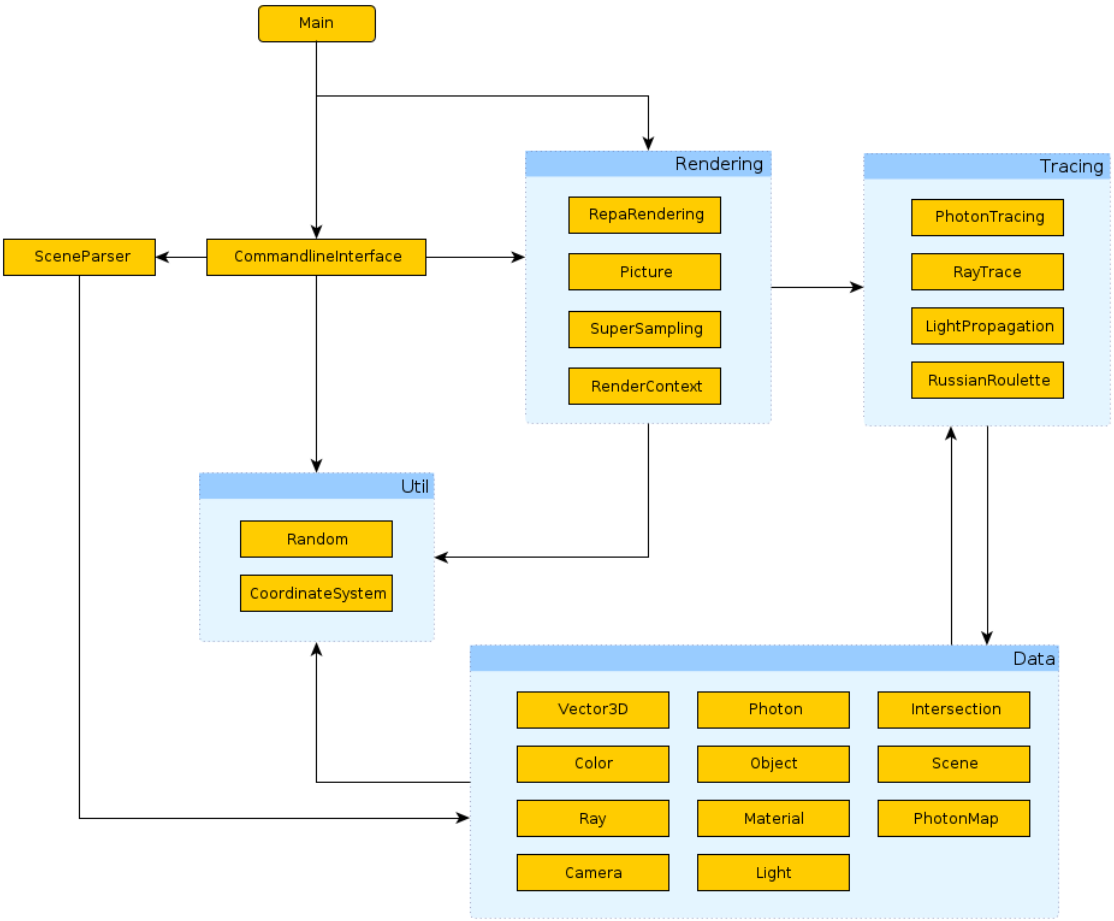


Рис. 3: Модули в пространствах имен

3.3 Описание модулей программы

Модуль	**Назначение**
Main	Интерфейс, управление программой.
RepaRendering	Параллельная визуализация сцены.
SceneParse	Загрузка сцены из файла.
CommandlineInterface	Интерфейс командной строки.

RenderContext	Параметры визуализации сцены.
Picture	• Визуализация сцены. • Сглаживание.
RayTrace	• Обратная трассировка лучей. • Использование сбора фотонов.
PhotonMap	• Построение фотонной карты. • Сбор фотонов.
PhotonTracing	• Излучение фотонов. • Трассировка фотонов.
Photon	Описание структуры фотона.
Scene	Описание структуры сцены.
Camera	• Описание структуры камеры. • Управление камерой.

Light	• Описание источников света. • Расчет прямого освещения.
LightPropagation	Распространение лучей и фотонов по сцене.
RussianRoulette	Нахождение дальнейшего состояния фотона после столкновения с объектом.
Intersection	• Поиск ближайшего пересечения. • Поиск пересечений с объектами сцены.
Object	• Описание объектов сцены. • Расчет нормалей поверхностей.
Material	Описание материалов объектов.
Color	Функции, облегчающие работу со цветом.
Random	• Генерация случайных чисел. • Генерация случайного распределения точек на поверхности.
CoordinateSystem	• Работа со сферическими координатами. • UV-преобразование.

Ray	Описание структуры луча, нахождение точки на луче.
Vector3D	Трехмерный вектор, функции для удобства работы с ним.

3.4 Описание структур данных программы

Структуры данных представляют собой основу программы, задавая удобство использования их функциями программы.

Vec3, Point, Scalar, Normal3

```
data Vec3 = Vec3 Double Double Double
newtype Normal3 = Normal3 Vec3
type Point = Vec3
type Scalar = Double
```

Трехмерный вектор содержит три компоненты - x, y, z. Нормаль содержит трехмерный вектор и является экземпляром класса типов UnitVector, что гарантирует то, что при любых возможных преобразованиях нормали она останется единичным вектором.

Точка является синонимом типа вектора и содержит свои координаты.

Тип Scalar является синонимом вещественного числа и объявлен для удобства, в случае необходимости использовать Float или иной тип для вычислений.

Color

```
type Color = Vec3
```

Тип цвета является синонимом вектора в пространстве RGB, что дает возможность использовать его как обычный вектор, также в функциях, манипулирующих цветом его компоненты не превосходят

1.

Material

```
data Material = Material { diffuseColor     :: Color
                         , diffuseCoeff     :: Scalar
                         , reflectance      :: Scalar
                         , specularColor    :: Color
                         , specularCoeff    :: Scalar
                         , specularExp      :: Scalar
                         , transmittance    :: Scalar
                         , refractionCoeff  :: Scalar
                         }
```

diffuseColor	базовый цвет материала
diffuseCoeff	коэффициент рассеивания
reflectance	коэффициент отражения
specularColor	цвет блика
specularCoeff	коэффициент блика
specularExp	параметр определяющий резкость и величину блика
transmittance	коэффициент пропускания
refractionCoeff	коэффициент преломления

Object

```
data Triangle = Triangle { vertices     :: (Point, Point, Point)
                         , precompInd   :: (Int, Int, Int)
                         , precompN     :: (Double, Double, Double)
                         , precompB     :: (Double, Double)
                         , precompC     :: (Double, Double)
                         , precompNorm  :: Normal3
                         }

data Shape = Sphere { radius :: Scalar }

           | Plane  { planeNormal :: Normal3
                    , planePoint  :: Point
                    }
```

```
          | TriangleMesh { triangle :: Triangle }
data Object = Object { center   :: Point
                     , shape    :: Shape
                     , material :: Material
                     }
```

Объект определяет его центром (не всегда используется), видом поверхности и материалом.

Поверхность может быть сферой заданного радиуса, плоскость, определяемой ее нормалью и точкой на плоскости, поверхностью, состоящей из треугольников.

Треугольник определяется своими вершинами и при создании заполняется дополнительной информацией для ускорения поиска пересечения.

Ray

```
data Ray = Ray { origin    :: !Point
               , direction :: !Normal3
               } deriving (Show, Read)
```

Луч определяется своим началом и направлением.

Light

```
data Light = AmbientLight { color :: !Color }
           | PointLight   { color :: !Color, position :: !Point }
```

Источник света может быть фоновым и точечным, определяемый своим цветом, а в случае точечного источника - и координатами.

Camera

```
data Camera = Camera { position      :: Point
                     , direction     :: Normal3
                     , up            :: Normal3
                     , distToScreen  :: Scalar
                     }
```

Камера определяется местонахождением, направлением обзора, нормалью, указывающей вверх и расстоянием до экрана.

Photon

```
data Photon = Photon { photonRay  :: !Ray
                     , power      :: !PhotonPower
                     , addToMap   :: !Bool
                     }
```

Фотон определяется лучом (координатами и направлением) и цветом. Также присутствует флаг добавления, используемый при построении карты каустики.

PhotonMap

```
type PhotonMap = Vector Photon
```

Фотонная карта - массив фотонов.

Scene

```
data Scene = Scene { sceneObjects :: Objects
                   , sceneLights  :: Lights
                   }
```

Сцена содержит объекты и источники света.

TracingContext

```
type TracingContext = (Int, Scalar, Scalar, Scalar)
```

Вспомогательная структура для алгоритма трассировки - кортеж, содержащий максимальную глубину рекурсии, текущий относительный вклад отраженного луча, преломленного луча и коэффициент преломления.

PhotonTracingContext

```
type PhotonTracingContext = (Int, Int, Bool)
```

Параметры трассировки фотонов - глубина рекурсии, количество излучаемых фотонов и флаг определяющий карту как карту каустики.

RenderContext

```
data ProjectionContext = ProjectionContext { camera      :: Camera
                                           , resolution  :: Resolution
                                           , scene       :: Scene
                                           }

data RenderContext = RenderContext { projectionContext :: ProjectionContext
                                   , rayBounceLimit    :: Int
                                   , isSupersampled    :: Bool
                                   , isJittered        :: Bool
                                   , numberOfSamples   :: Int
                                   , isUsingPhotMap    :: Bool
                                   , photMap           :: Maybe PhotonMap
                                   , photMapContext    :: PhotonTracingContext
                                   , contributions     :: (Double, Double)
                                   , gatherRadSq       :: Double
                                   , isImportance      :: Bool
                                   , addCaustic        :: Bool
                                   , causticContext    :: PhotonTracingContext
                                   , gatherCaustic     :: Double
                                   }
```

Параметры визуализации.

Параметры проекции сцены на экран содержат камеру, разрешение экрана и геометрию сцены.

Общие параметры визуализации содержат параметры проекции, глубину рекурсии трассировки лучей, параметры сглаживания (isSupersampled - используется ли сглаживание, isJittered - сдвигаются ли в случайных направлениях лучи при суперсэмплинге, numberOfSamples - количество отрезков, на которые разбивается сторона пикселя при сглаживании), используется ли фотонная карта, параметры построения фотонной карты, доля вкладов в яркость в данной точки от фотонной карты и от трассировки лучей, радиус сбора фотонов в квадрате. А также флаг использования выборки по значимости, режим визуализации глобальной карты и карты каустики вместе, параметры построения карты каустики и радиус сбора.

3.5 Формат файлов описания сцены

Пользователь имеет возможность загрузить сцену, описанную во внешнем файле. Файл может содержать описание материалов, объектов и источников света. Пользователь может объявлять идентификаторы с определенным значением и в дальнейшем использовать их. Это позволяет, к примеру, описать какой-либо материал и при описании объекта обращаться к нему по его идентификатору.

Файл сцены должен содержать список объектов в идентификаторе **objects** и список источников света - **lights**.

Синтаксис определения прост:

идентификатор = описание

На одной строке может располагаться только одно объявление.

Синтаксис определения списка:

идентификатор = [элемент 1, элемент 2, ..., элемент N]

Таким образом элементы списка перечисляются через запятую внутри квадратных скобок. Каждый элемент списка может быть расположен на новой строке, но должен располагаться на одном уровне с открывающей квадратной скобкой.

При использовании отрицательных чисел их необходимо заключать в круглые скобки.

Также можно комментировать отдельные строки. Для этого перед в начале строки необходимо напечатать $--$.
Например: $--$ This is comment.

Формат файла сцены близок к корректной программе на языке Haskell.

Ниже приведены правила описания каждой группы элементов с примерами.

Цвет

идентификатор = Vec3 r g b

white = Vec3 1.0 1.0 1.0

Где r, g, b - соответственно красная, зеленая и синяя компоненты цвета, находящиеся в интервале $[0, 1]$.

Материалы

ид. = Material (дифф. цвет) k_{diff} k_{refl} (цвет блика) k_s exp_s k_{trans} n
mat0 = Material green 1.0 0.0 white 0.0 100 0.0 0.0

1. Диффузный цвет - “основной” цвет материала
2. k_{diff} - коэффициент рассеивания
3. k_{refl} - коэффициент отражения
4. цвет блика
5. k_s - коэффициент блика
6. exp_s - экспонента блика (влияет на размер и резкость блика), k_{transp} - коэффициент пропускания
7. n - абсолютный коэффициент преломления

Объекты сцены

Объекты сцены включают в себя сферы, плоскости и треугольники.

сфера = mkSphere (координаты центра) (радиус) (материал)
sphere = mkSphere (Vec3 1 3 4) 1.5 mat0

плоскость = mkPlane (нормаль) (точка на плоскости) (материал)
plane = mkPlane (Vec3 0 100 0) (Vec3 0 3.5 15) mat0

треугольник = mkTriangleMesh (v_1, v_2, v_3) (материал)
triangle = mkTriangleMesh ((Vec3 0 0 1), (Vec3 0 1 2), (Vec3 1 9 4)) mat0

v_1, v_2, v_3 - вершины треугольника.

Источники света

источник = PointLight (цвет) (координаты)
light = PointLight white (Vec3 1 2 3)

3.6 Интерфейс программы

Программный продукт состоит из двух программ - программы визуализации, предоставляющей консольный интерфейс, и программы-интерфейса, созданной с помощью фреймворка Qt.

Интерфейс состоит из главного окна, функционал при это распределен по вкладкам. Присутствует меню, в котором пользователь может вызвать окно дополнительных настроек или выйти из программы. В нижней части интерфейса расположена кнопка, начинающая процесс визуализации, а также полоса, сообщающая о прогрессе визуализации. Пользователь может сохранить установленные параметры с помощью соответствующей кнопки, а также восстановить сохраненные параметры.

Вкладка “Изображение”

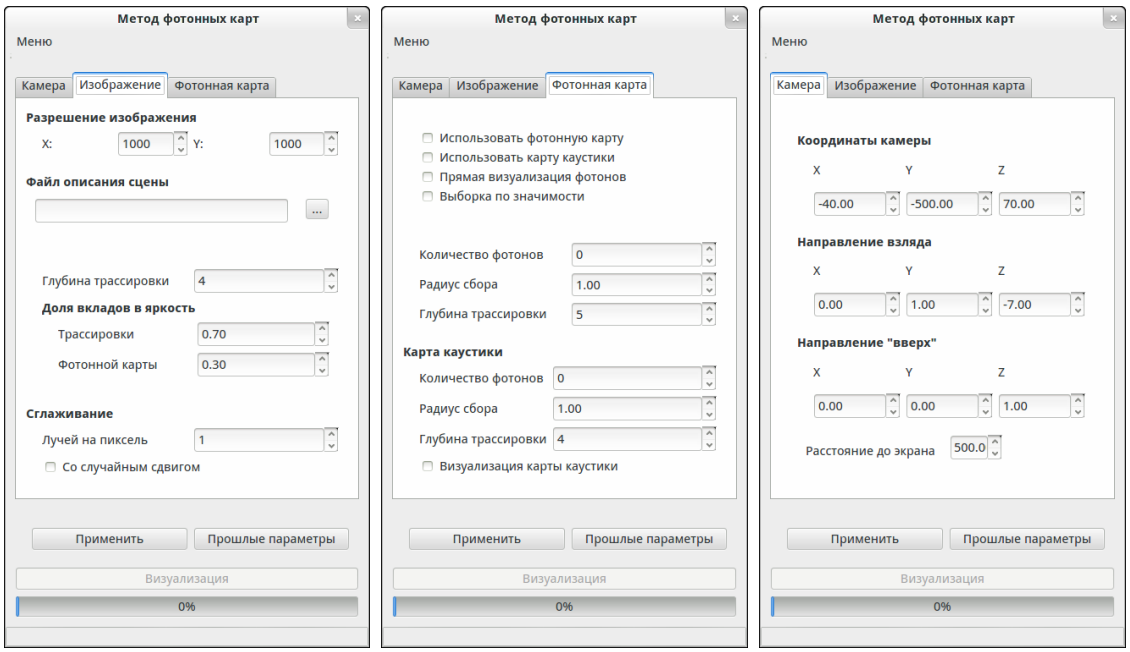
Содержит элементы управления, позволяющие выбрать файл с описанием сцены, разрешение изображения, максимальную глубину обратной трассировки лучей, доли вкладов в освещенность от трассировки и фотонной карты, а также параметры сглаживания.

Вкладка “Фотонная карта”

Позволяет включать использование информации с фотонной карты, а также параметры построения и визуализации: количество излучаемых фотонов, радиус сброра фотонов, глубина трассировки фотонов, параметры построения карты каустики (количество фотонов, радиус сбора, глубина трассровки). Доступен режим, визуализирующий только карту каустики. Помимо этого есть возможность выбрать режим визуализации местонахождения фотонов из карты.

Вкладка “Камера”

Позволяет изменять положение камеры в пространстве, изменять направление взгляда камеры, поворачивать камеру, указывая вектор “вверх”, а также изменять расстояние до “экрана”.



(a) Изображение (b) Фотонная карта (c) Камера

Рис. 4: Главное окно

Дополнительные настройки

Также есть окно дополнительных настроек, позволяющих пользователю выбрать программу для просмотра синтезированного изображения и количество потоков, используемых при визуализации.

3.7 Характеристики и условия применения программы

Потребление памяти варьируется в пределах от 25 до 350 Мб оперативной памяти в зависимости от разрешения изображения (23

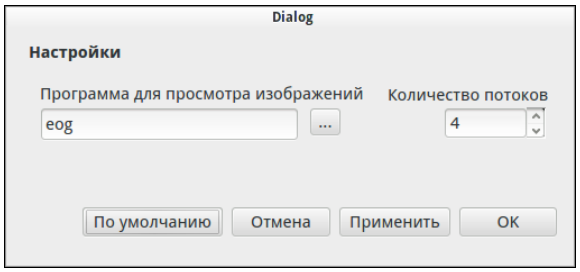


Рис. 5: Дополнительные настройки

Мб потребляет интерфейс на Qt, от 2 до 330 - визуализатор). Верхняя граница в 350 Мб следует из искусственного ограничения разрешения изображения 10000 на 10000 пикселями. Программа требовательна к процессору и для быстрой визуализации рекомендуются многоядерные процессоры.

Запуск программы осуществляется открытием бинарного файла.

Программа показывает сообщение об ошибке при загрузке некорректного файла с описанием сцены и невозможности открыть генерирумое изображение в программе для просмотра изображений, указанной пользователем.

В приложении С показаны примеры работы программы.

3.8 Входные и выходные данные

Входными данными является описание сцены, находящееся в указанном файле на диске, и параметры визуализации изменяемые в интерфейсе программы. Выходными данными является файл с изображеним визуализированной сцены, открываемый в указанной пользователем программе для просмотра изображений.

4 Экспериментально-исследовательский раздел

4.1 Исследование скорости реализации алгоритмов

Исследования проводятся на компьютере с процессором Intel Core i5 Processor 520UM с 4 логическими ядрами. Тестовая сцена визуализируется в разрешении 1000 на 1000 пикселей.

Интерес представляют зависимости времени визуализации от количества излучаемых фотонов, глубины трассировки фотонов, параметров сглаживания и количества потоков.

Сначала необходимо измерить зависимость времени визуализации без использования фотонных карт от количества потоков. Здесь и далее глубина трассировки лучей равна 5.

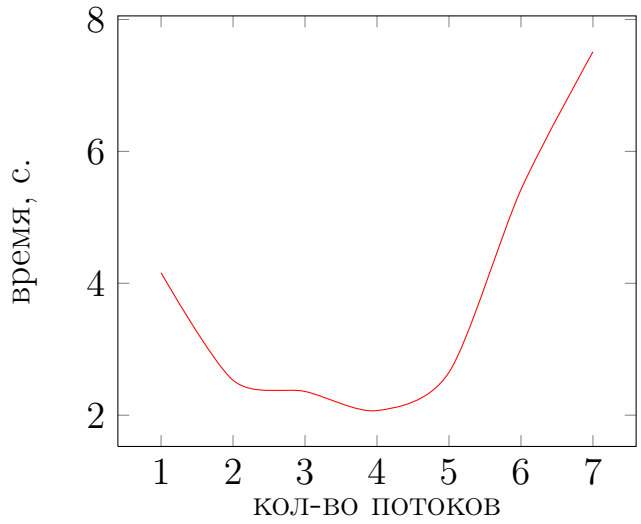


Рис. 6: Зависимость времени визуализации от количества потоков

Глубина трассировки фотонов фиксирована и равна 5.

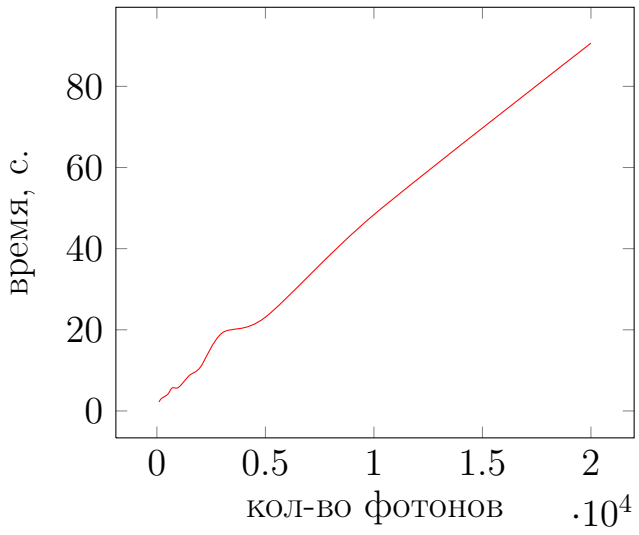


Рис. 7: Зависимость времени визуализации от количества излучаемых фотонов

Как видно из графика зависимость является линейной, что согласуется с теоретической оценкой. Так как основное время занимает не трассировка фотонов, а сбор информации с фотонной карты, который осуществляется за линейное время.

Зафиксируем количество излучаемых фотонов на отметке 1000 и проведем исследование зависимости времени визуализации от количества потоков.

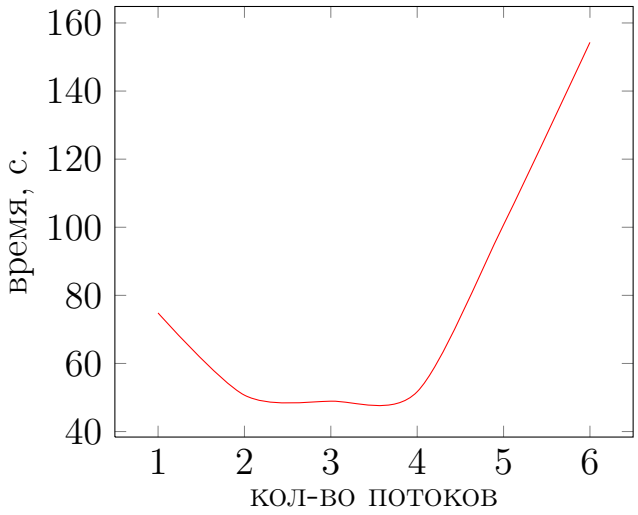


Рис. 8: Зависимость времени визуализации от количества потоков

4.2 Исследование визуальных характеристик

Согласно теории при увеличении количества фотонов в фотонной карте рассчитанная освещенность приближается к физической модели. Также на качество изображения сильно влияет выбранный радиус сбора и использование выборки фотонов по значимости.

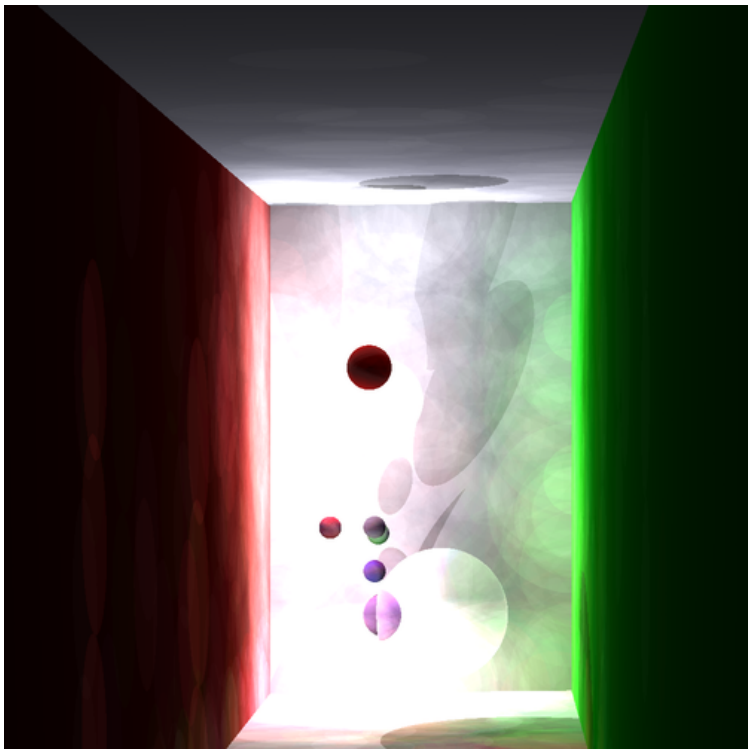


Рис. 9: $r = 4$, без использования выборки

Сильно заменты следы от сфер сбора фотонов.

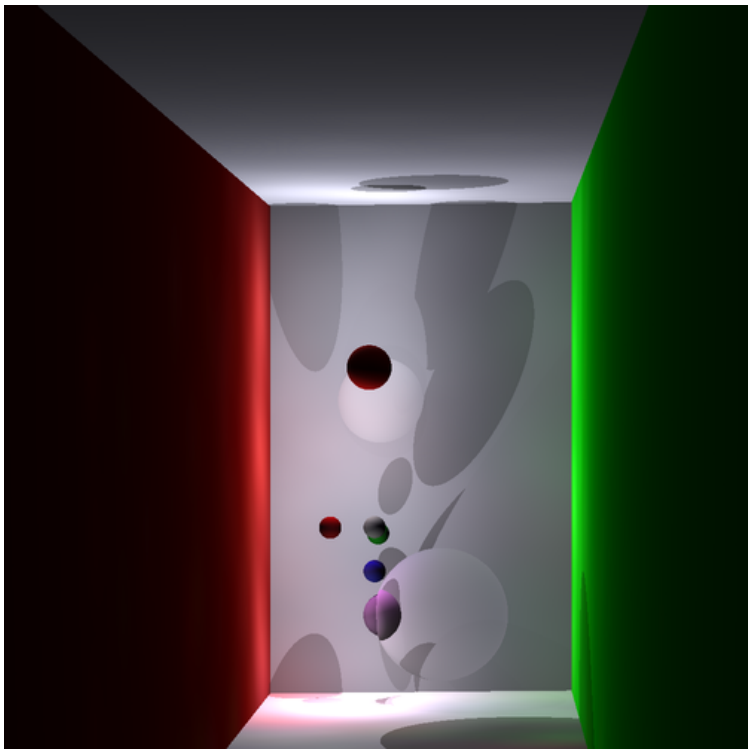


Рис. 10: $r = 4$, с использованием выборки

Изображение выглядит значительно лучше, но ярковыраженные эффекты цветого заполнения стали не так заметны. Визуализация без использования выборки по значению позволяет использовать меньшее число фотонов при визуализации сцены, но при этом дает менее реалистичные результаты, также требуется увеличить радиус сбора для более “гладкой” картинки, что может привести к включению в сферу сбора лишних фотонов.

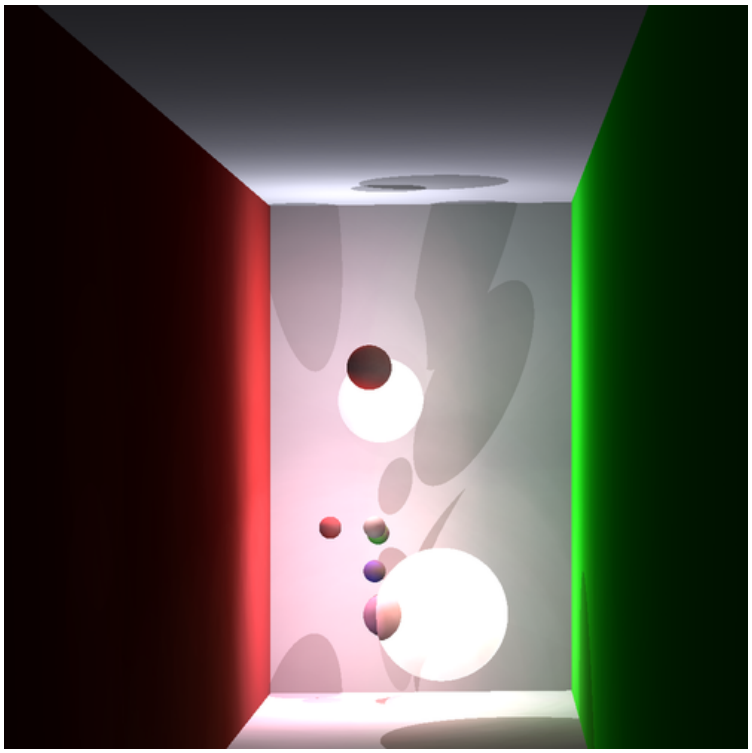


Рис. 11: $r = 10$, без использования выборки

При увеличении радиуса сбора изображение значительно улучшилось.

В целом, использование выборки по значению дает более реалистичный результат, но требует большее количество фотонов.

Стоит сравнить методы при высоком числе и плотности фотонов. Идеальным примером для этого является карта каустики.

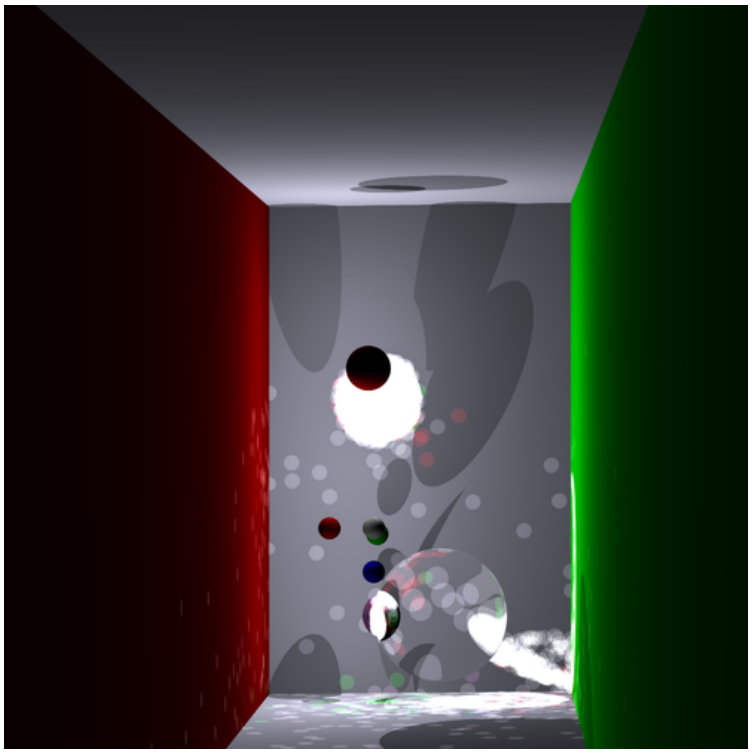


Рис. 12: $r = 0.35$, без использования выборки

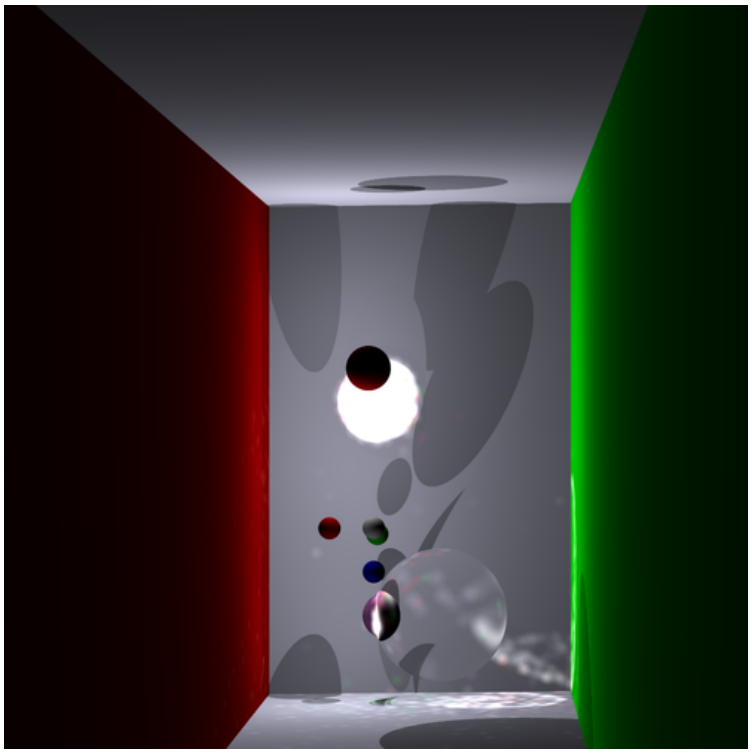


Рис. 13: $r = 0.35$, с использованием выборки

Выборка по значению дает более хороший результат, так что при визуализации карты каустики предпочтительнее использоваеть выборку по значению, так как карты каустики содержит большое число фотонов.

5 Заключение

В данной работе был реализован метод фотонных карт, использующий обратнку трассировку лучей для сбора фотонов. Программный продукт был разделен на две программы - визуализатор, предоставляющий консольный интерфейс, и графический интерфейс пользователя, управляющий программой-визуализатором. Загрузка описания сцены из файла позволяет пользователю создать и визуализировать необходимую сцену. Программа не привязана к какой-то конкретной операционной системе и может быть скомпилирована и запущена на всех популярных ОС.

Программа моделирует следующие эффекты, необходимые для приближения к физической модели:

- первичное освещение,
- отражение света,
- преломление света,
- диффузное отражение,
- каустика,
- цветовое заполнение (color bleeding).

При исследовании было замечено сильное положительное влияние использование выборки по значению на синтезируемое изображени.

Алгоритм обратной трассировки лучей в перспективе может быть расширен до алгоритма распределенной трассировки лучей, что позволит создавать более реалистичные изображения. В этом случае для сохранения приемлемого времени визуализации потребуются тщательная оптимизация.

Также метод можно улучшить метод фотонных карт, используя недавние публикации. [8] [9]

В качестве пути развития программы можно предложить загрузку описания сцены из популярных пакетов 3D-моделирования. Помимо этого можно несколько изменить используемые алгоритмы и использовать видеокарту для расчетов. Это достижимо с помощью изменения алгоритмов для эффективности расчетов на видеокартах и использовании библиотеки Accelerate для преобразования кода на Haskell в код на языке C, использующий OpenCL или CUDA.

Список литературы

1. Ingo Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004.

2. Addy Ngan, Fredo Durand, and Wojciech Matusik. Experimental validation of analytical brdf models. *Siggraph*, 2004.

3. Ben Lippmeier, Manuel Chakravarty, Gabriele Keller, and Simon Peyton Jones. Guiding parallel array fusion with indexed types. In *Proceedings of the 2012 symposium on Haskell symposium*, Haskell '12, pages 25–36, New York, NY, USA, 2012. ACM.

4. Gabriele Keller, Manuel M.T. Chakravarty, Roman Leshchinskiy, Simon Peyton Jones, and Ben Lippmeier. Regular, shape-polymorphic, parallel arrays in haskell. *SIGPLAN Not.*, 45(9):261–272, 2010.

5. Ben Lippmeier and Gabriele Keller. Efficient parallel stencil convolution in haskell. *SIGPLAN Not.*, 46(12):59–70, 2011.

6. J. Hughes. Why functional programming matters. *Comput. J.*, 32(2):98–107, 1989.

7. Haskell platform kernel description. `http://haskell.org/platform`, 2012. Accessed: 12/11/2012.

8. Toshiya Hachisuka, Shinji Ogaki, and Henrik Wann Jensen. Progressive photon mapping. In *ACM SIGGRAPH Asia 2008 papers*, SIGGRAPH Asia '08, pages 130:1–130:8, New York, NY, USA, 2008. ACM.

9. Toshiya Hachisuka and Henrik Wann Jensen. Stochastic progressive photon mapping. *ACM Trans. Graph.*, 28(5):141:1–141:8, 2009.

10. Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

11. James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. *Computer graphics: principles and practice (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

12. Henrik Wann Jensen. Global illumination using photon maps. In *Proceedings of the eurographics workshop on Rendering techniques '96*, pages 21–30, London, UK, UK, 1996. Springer-Verlag.

13. Paul Hudak and Joseph H. Fasel. A gentle introduction to haskell. *SIGPLAN Not.*, 27(5):1–52, 1992.

14. Bryan O'Sullivan, John Goerzen, and Don Stewart. *Real World Haskell*. O'Reilly Media, Inc., 1st edition, 2008.

Приложение А Структура модулей

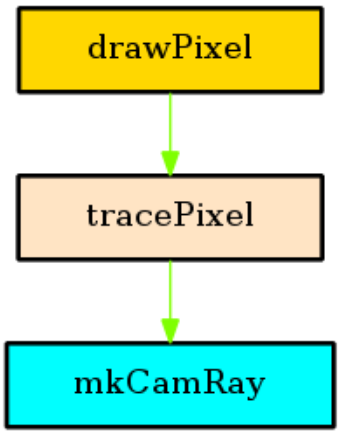


Рис. 14: Picture

Рис. 15: SceneParser

Рис. 16: RepaRendering

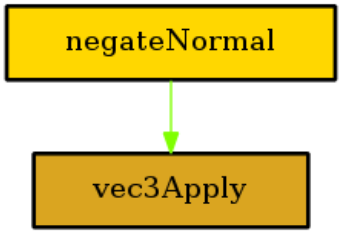


Рис. 17: Vector3D

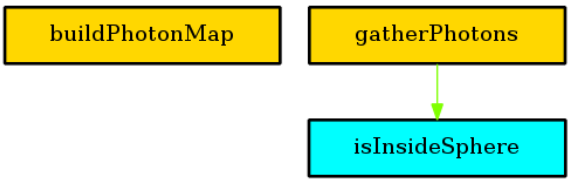


Рис. 18: PhotonMap

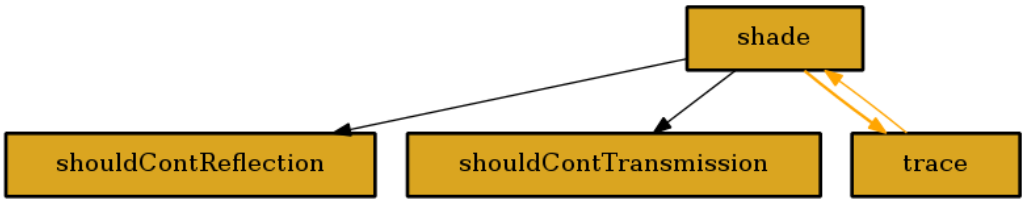


Рис. 19: RayTrace

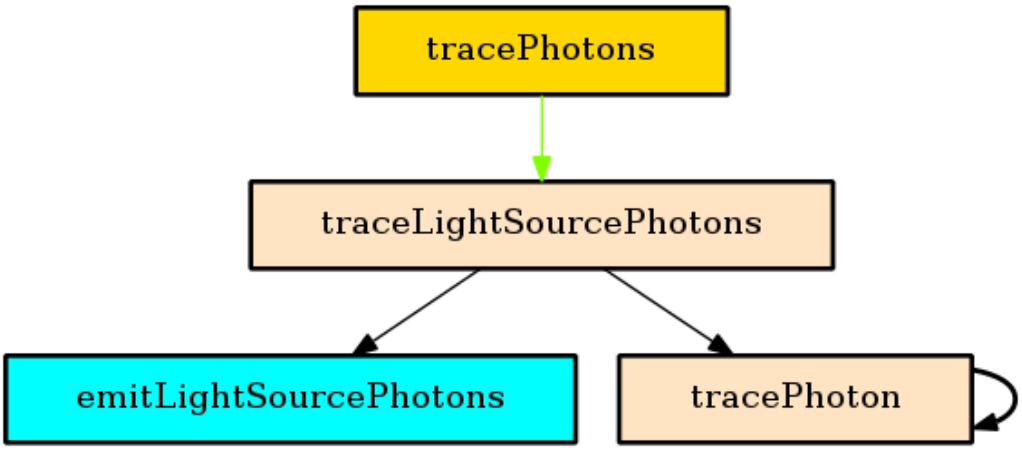


Рис. 20: PhotonTracing

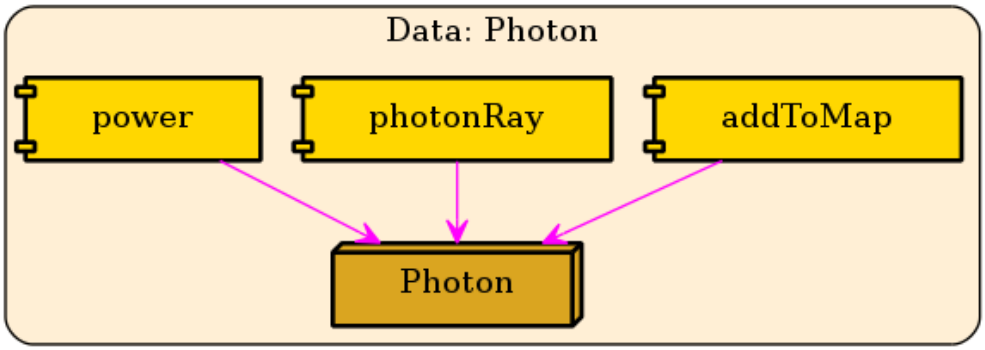


Рис. 21: Photon

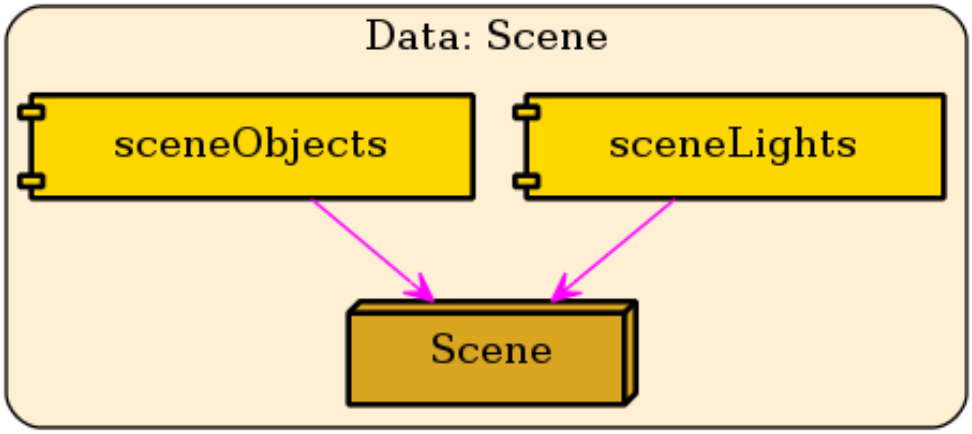


Рис. 22: Scene

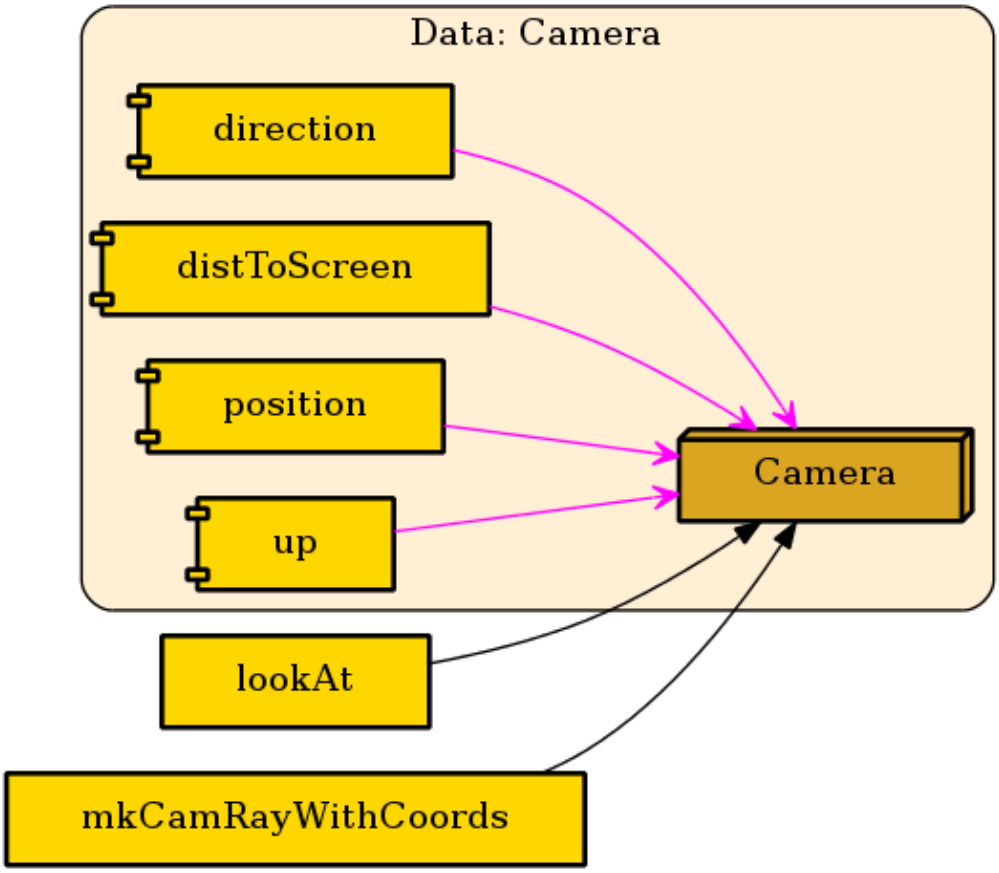


Рис. 23: Camera

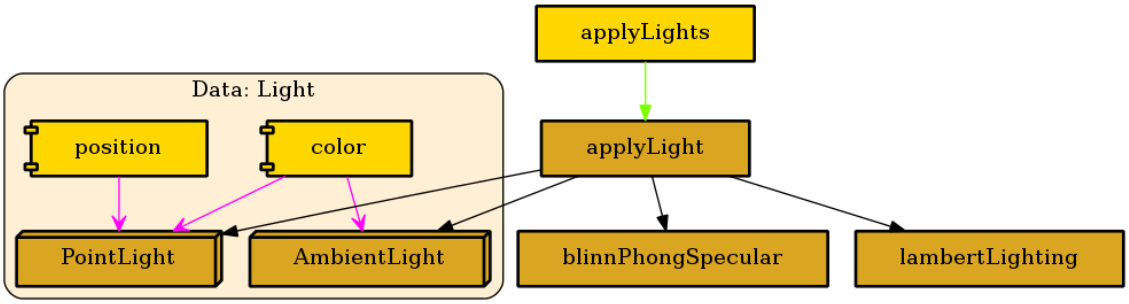


Рис. 24: Light

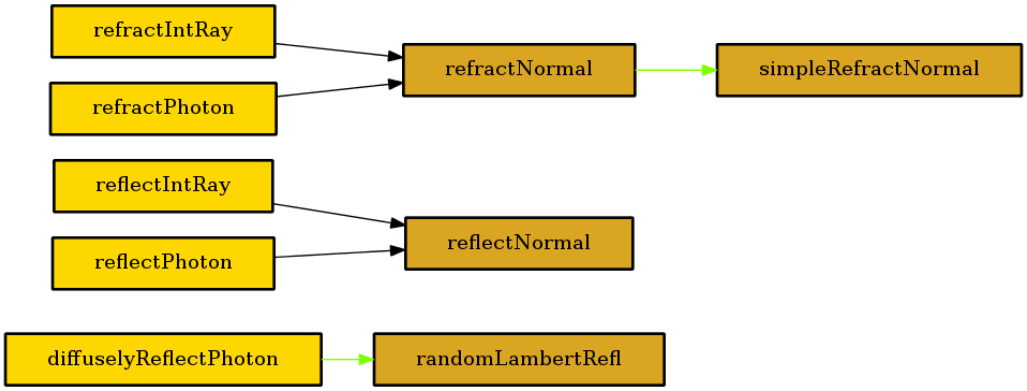


Рис. 25: LightPropagation

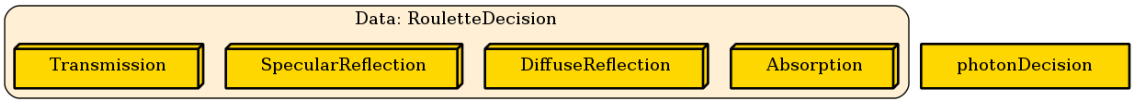


Рис. 26: RussianRoulette

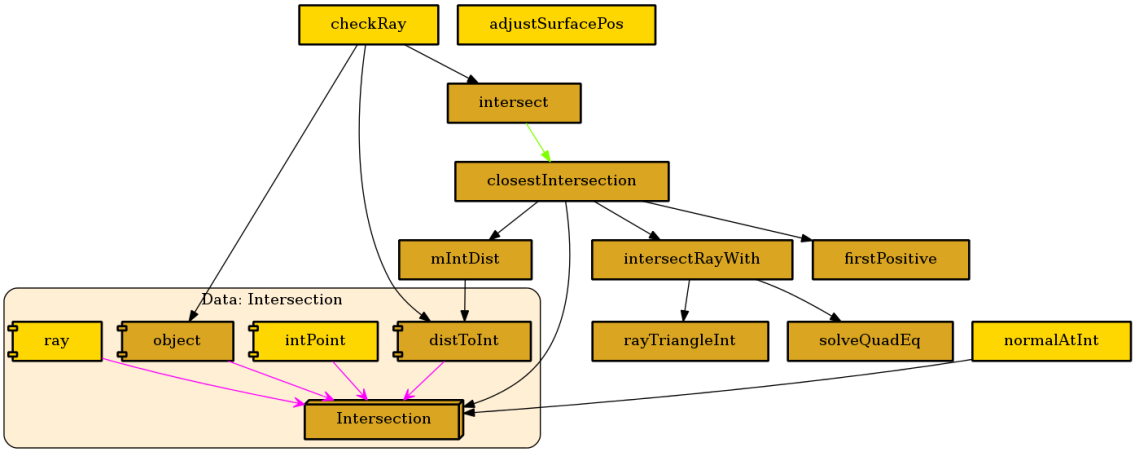


Рис. 27: Intersection

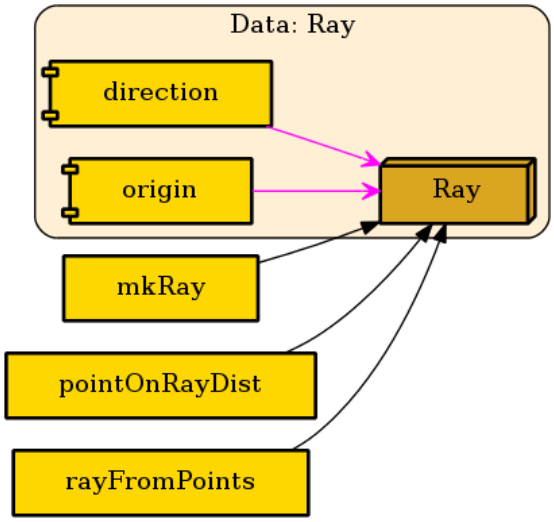


Рис. 28: Ray

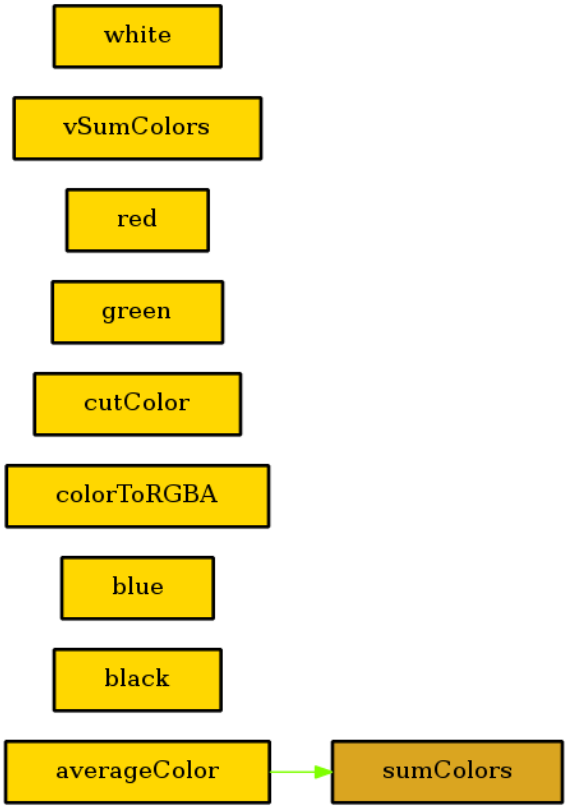

white
vSumColors
red
green
cutColor
colorToRGBA
blue
black
averageColor
sumColors

Рис. 29: Color

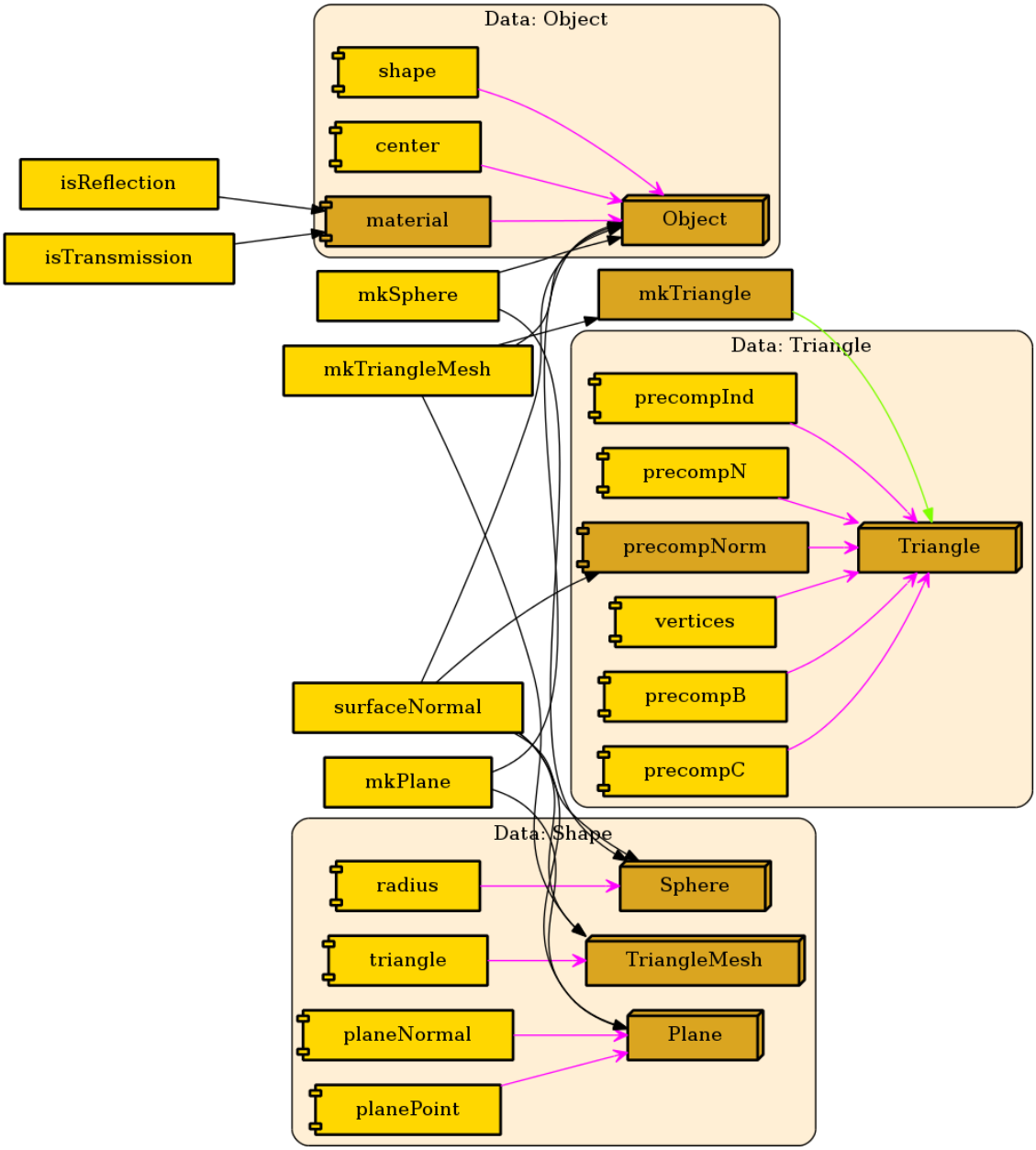


Рис. 30: Object

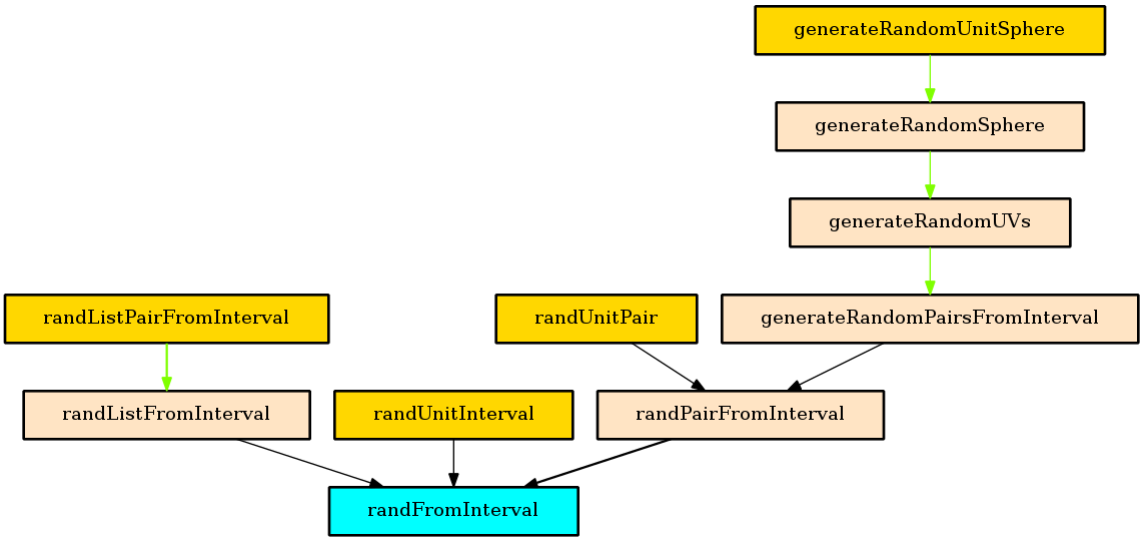


Рис. 31: Random

Рис. 32: CoordinateSystem

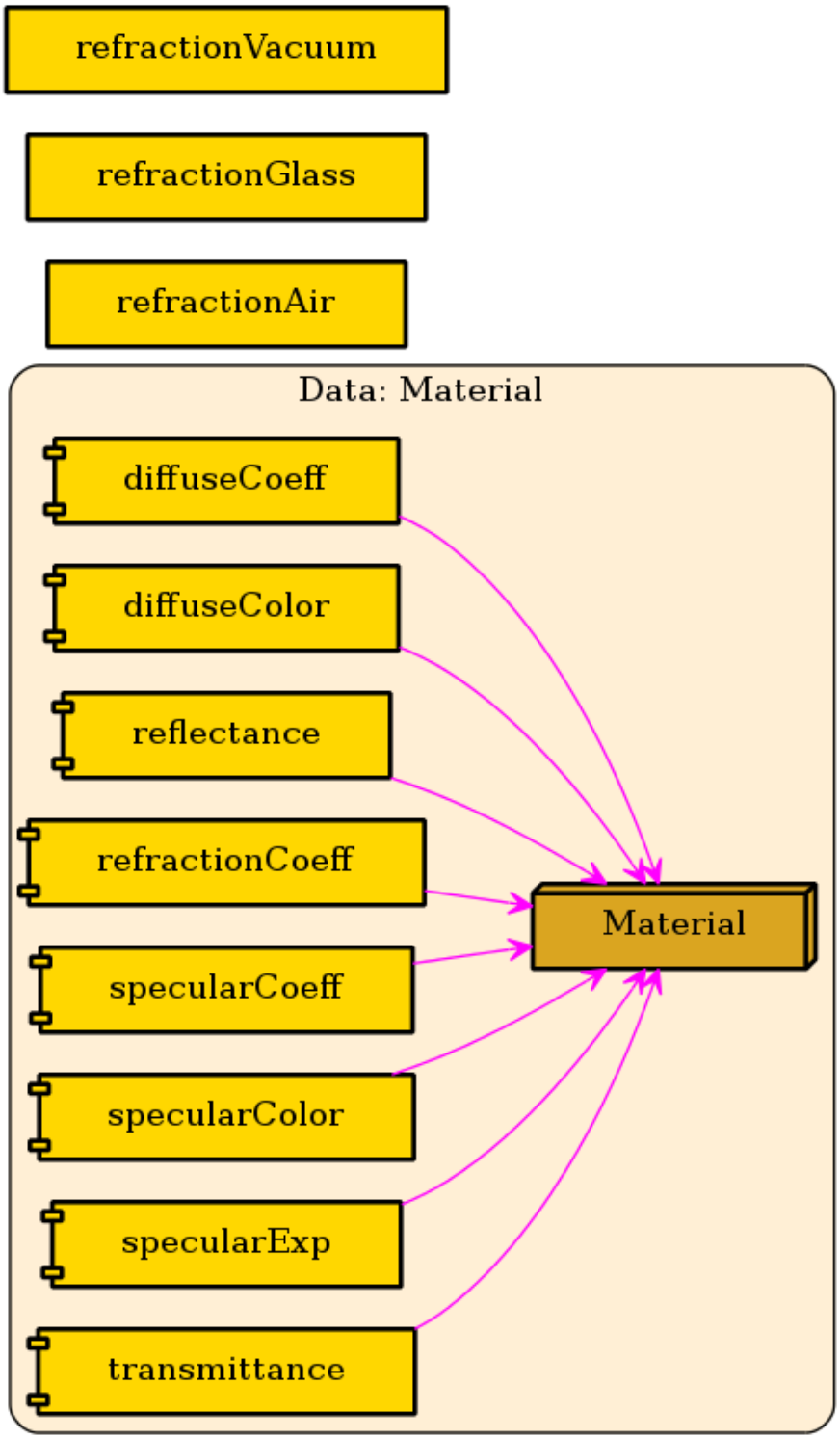


Рис. 33: Material

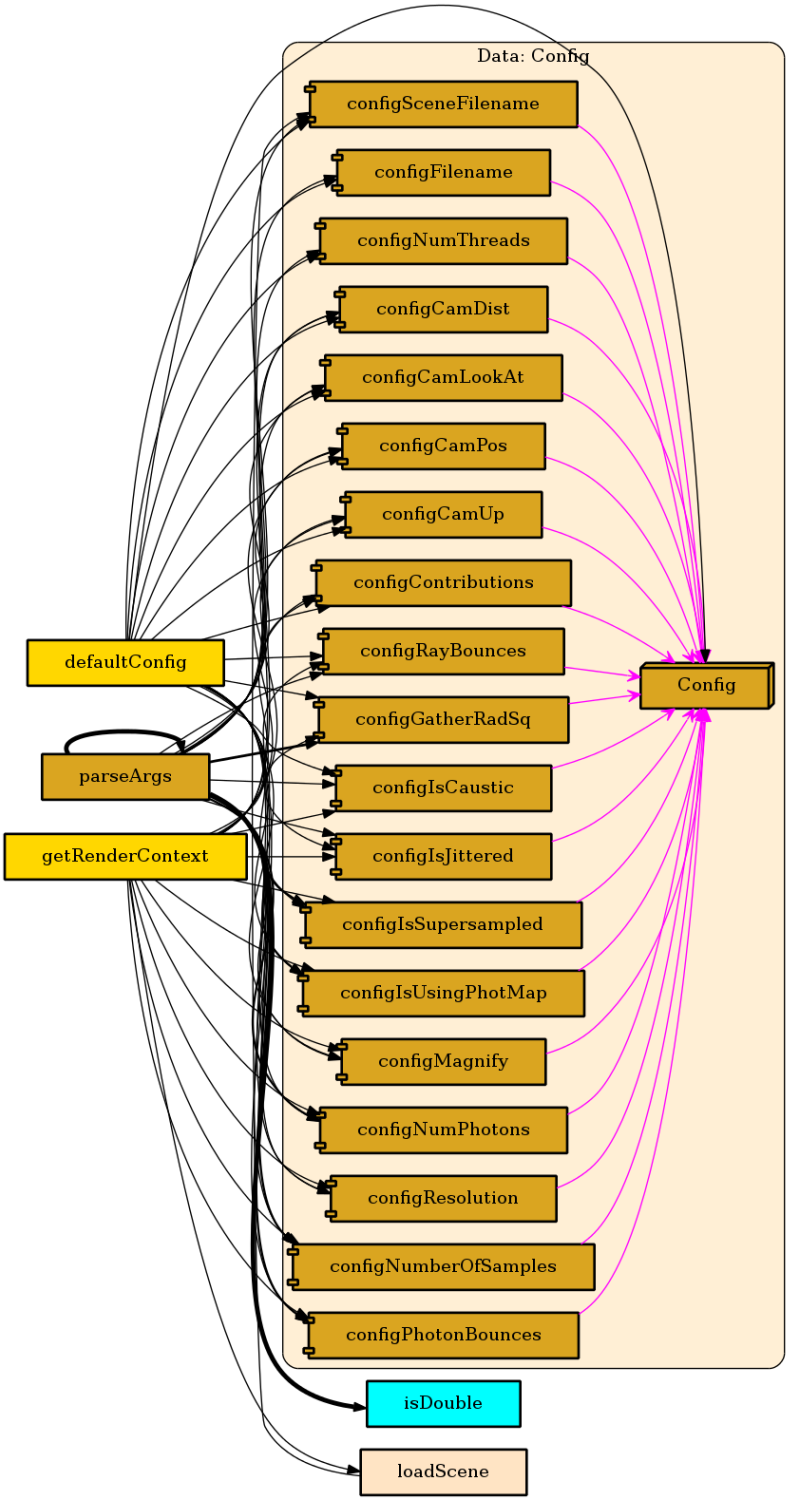


Рис. 34: CommandlineInterface

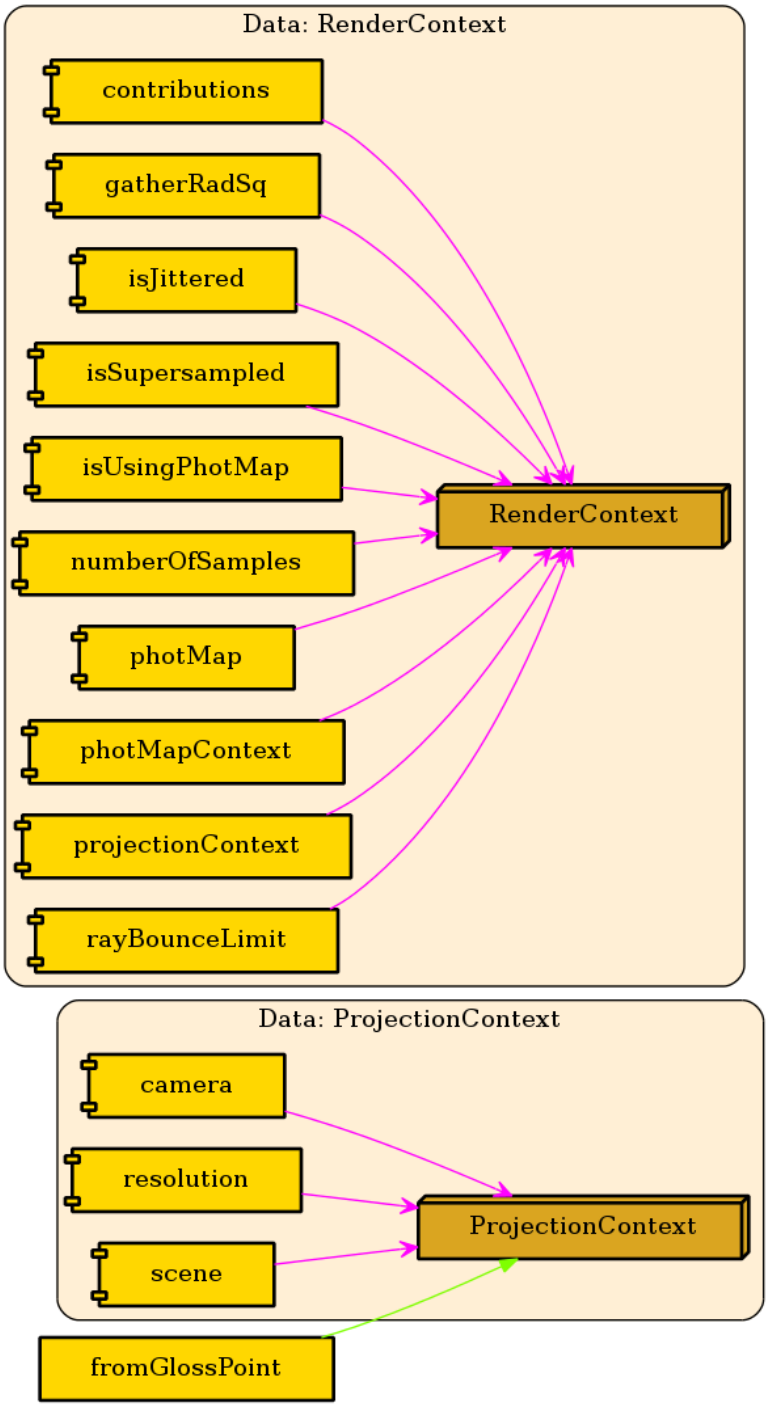


Рис. 35: RenderContext

Приложение В Примеры работы программы

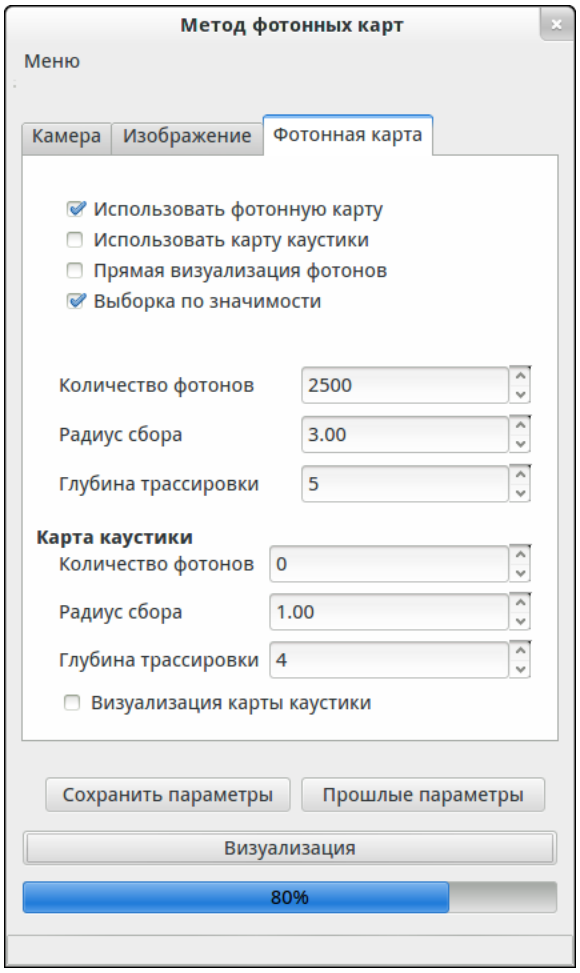


Рис. 36: Процесс визуализации

Рис. 37: Обратная трассировка лучей

Рис. 38: Расположение фотонов

Рис. 39: Визуализация карты со сбором фотонов