ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

**Факультет информатики, математики и компьютерных наук**

**Программа подготовки бакалавров по направлению   
09.03.04 Программная инженерия**

*Канделов Дамир Русланович*

**КУРСОВАЯ РАБОТА**

Сравнение различных стратегий выборки в рендеринге трассировкой путей

|  |  |
| --- | --- |
|  | Научный руководитель  Старший научный сотрудник  программной инженерии  факультета компьютерных наук  канд. физ. мат. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И.С. Бычков |

Нижний Новгород, 2024

**Содержание**

[Введение 3](#_Toc167056941)

[Постановка задачи 5](#_Toc167056942)

[1. Исследование предметной области 6](#_Toc167056943)

[1.1. Что такое Path tracing? 6](#_Toc167056944)

[1.2. Отличие Path tracing от Ray tracing 7](#_Toc167056945)

[1.3. Алгоритм работы Path tracing 8](#_Toc167056946)

[1.4. Случайные числа в алгоритме Path tracing 11](#_Toc167056947)

[2. Описание решения 13](#_Toc167056948)

[2.1. Генерация случайных чисел с использованием последовательности Хальтона 13](#_Toc167056949)

[3. Библиографический список 15](#_Toc167056950)

# Введение

В настоящее время компьютерная графика, а именно методы рендеринга изображений используются практически повсеместно. Рендеринг – это процесс генерации фотореалистичных или нереалистичных изображений по заданным 2D или 3D моделям, с помощью компьютерных программ. Он используется для создания CGI-изображений (computer-generated imagery), в киноиндустрии для создания визуальных эффектов (VFX), в разработке компьютерных игр и симуляторов для отрисовки сцен, которые пользователь будет видеть на экране, а также в сферах дизайна и архитектуры.

Существует несколько алгоритмов рендеринга для создания фотореалистичных изображений, наиболее известными являются:

Rasterization (Растеризация) – это процесс перевода изображения, описанного в векторном формате, в растровое изображение. По сравнению с другими техниками рендеринга является одной из наиболее быстрых, из-за чего используется в большом количестве 3D движков, отрисовывающих изображение в реальном времени. Однако, минусом данной техники является возможность показать очень малое число оптических эффектов.

Ray tracing (Трассировка лучей) – это метод запуска «лучей» из камеры в каждый пиксель изображения, которые по мере своего движения (отражения и преломления) собирают цвет данного пикселя. Ray tracing способен симулировать большое количество оптических эффектов, таких как отражения, преломления и дисперсию света, мягкие тени, глубину резкости, размытие в движении, каустику.

Path tracing (Трассировка путей) – этот метод похож на Ray tracing, однако он запускает большее число лучей через каждый пиксель, используя при этом методы Монте Карло, а также собирает информацию от всех источников света, которые освещают данный пиксель. Поэтому path tracing является наиболее близким к реальности по поведению глобального освещения. Если метод используется вместе с физически корректными материалами и объектами, то в результате получаются изображения, неотличимые от фотографий и реальности.

Основными потребителями алгоритмов рендеринга являются кино- и гейм-дев индустрии. Последние тенденции в обоих индустриях – создание максимально реалистичной графики. Как можно заметить, алгоритм Path tracing подходит больше всего для решения этой задачи. Однако, основным недостатком является большая вычислительная сложность данного алгоритма. Было замечено, что от метода выбора случайных чисел во время работы алгоритма зависит качество, а также время получения изображений хорошего качества.

Таким образом, востребованность и актуальность темы данной работы обусловлена потребностями сферы разработки игр и сферы создания визуальных эффектов в кино, нацеленных на получение реалистичной графики в своих проектах за сравнительно малое время работы.

# Постановка задачи

В данной работе будет подробно рассмотрен алгоритм Path tracing. На примере его работы мы разберем и сравним несколько наиболее популярных подходов к генерации случайных чисел.

Цель данного проекта: реализовать несколько подходов к генерации случайных чисел, добавить эту реализацию в программу для рендеринга изображений с возможностью выбора алгоритма, сравнить изображения с помощью различных метрик и выделить методы, которые выдают наилучший результат за наименьшее время.

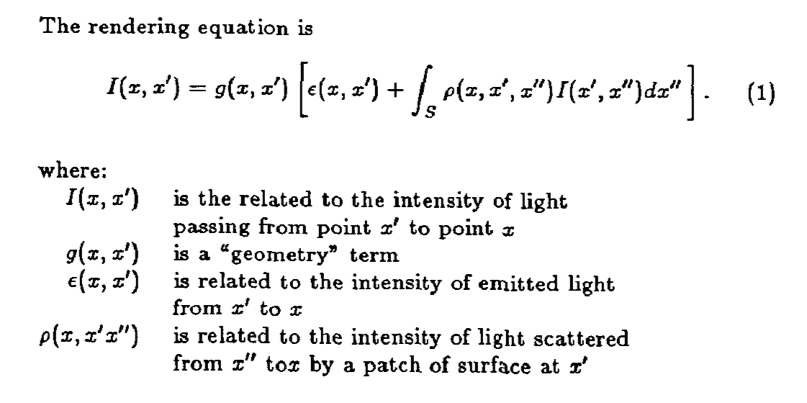
Задачами данного проекта являются:

1. Разобраться в алгоритме работы Path tracing и написать его с самого начала.
2. Узнать, где в алгоритме используются случайные числа и почему метод их выбора влияет на результирующие изображения.
3. Реализовать несколько подходов к генерации случайных чисел и имплементировать их в собственном проекте.
4. Сгенерировать и сравнить изображения, полученные с помощью новых методов выбора случайных чисел.

# Исследование предметной области

## Что такое Path tracing?

Path tracing – это метод рендеринга изображений по 3D сценам в компьютерной графике, который стремится воссоздать изображение максимально близкое к реальности. Данный метод реализует все визуальные эффекты, которые возможно представить на изображении.



«The Rendering Equation» Jim Kajiya, 1986

// TODO: Расписать подробно уравнение и его составляющие

В 1986 году Jim Kajiya выпустил статью под названием «The Rendering Equation», которая объединяла компьютерную графику и физику. В ней было представлено уравнение, описывающие поведение света и позволяющее точно представить, как свет движется по 3D сцене. В этой же статье было и название нового метода рендеринга, который использовал данное уравнение: **Path tracing.**

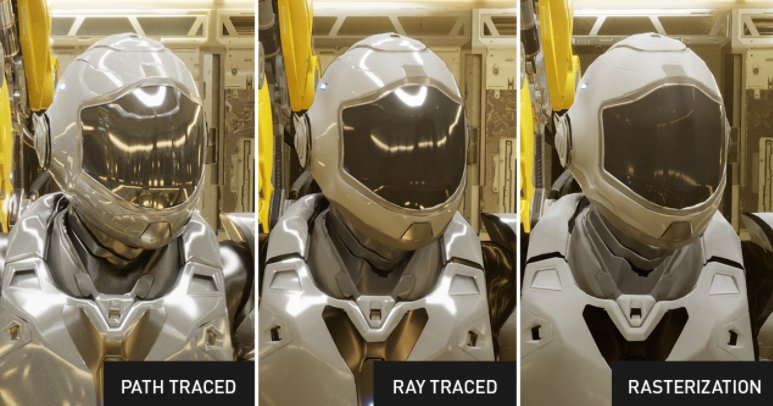
Полученное уравнение рендеринга было довольно небольшим и понятным, однако решить его не так просто из-за того, что сцены в компьютерной графике очень сложны и описываются миллионами или даже миллиардами треугольников в современном мире. Нельзя полностью решить это уравнение.

Jim Kajiya предложил решение этой проблемы: не требуется решать уравнение напрямую, его можно решить по путям для каждого отдельного луча (например, с помощью алгоритма Ray tracing). Соответственно, получение фотореалистичных изображений возможно, если решить это уравнение для достаточного количество лучей.

## Отличие Path tracing от Ray tracing

Одним из наиболее частых вопросов при знакомстве с методами рендеринга является вопрос отличия Path tracing и Ray tracing. Как уже было сказано ранее, в алгоритме Path tracing используется Ray tracing, однако следует разделять эти два метода рендеринга.

Основное отличие Path tracing от Ray tracing в том, что в трассировке путей мы следуем за лучом до тех пор, пока он не достигнет источника, чтобы получить цветовую и световую характеристику. А также, зачастую в Path tracing используется большее число лучей, получившиеся цвета которых каким-то образом образуют один цвет (обычно берется среднее значение). В результате чего мы получаем более физически-корректное и реалистичное изображение.



Сравнение Path tracing и Ray tracing [1]

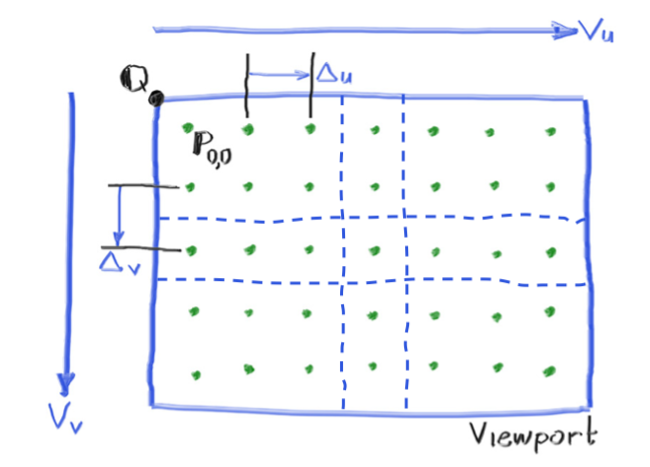
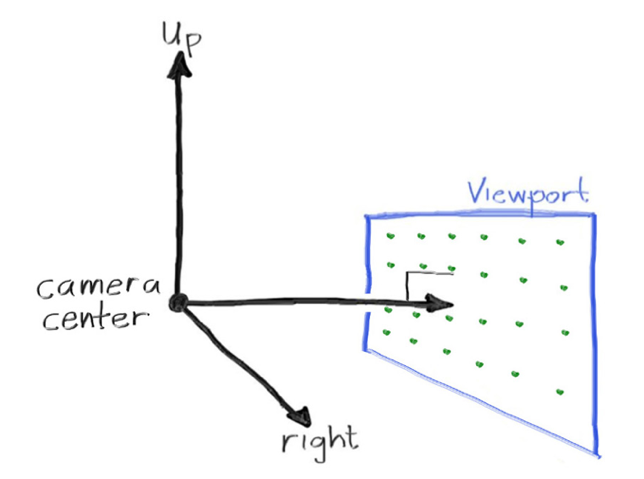
Как мы видим, в данном примере явно заметно, что Path tracing симулирует большее количество света, что гораздо ближе к реальности: в отражениях мы должны хорошо видеть все источники света.

## Алгоритм работы Path tracing

Давайте разберемся, как же это работает. Как было сказано выше, алгоритм максимально приближен к тому, как это происходит в реальном мире. Но с одним исключением.

В реальном мире свет распространяется от источника света, отражается от объектов и попадает на сетчатку глаза. Однако реализация такого поведения имеет слишком большую вычислительную сложность, так как требуется запустить лучи во все стороны от источника света (потому что свет распространяется во все стороны). Поэтому в компьютерной графике было принято решение развернуть эту последовательность действий: запускать лучи не из источников света, а из камеры (позиции, откуда пользователь смотрит на 3D сцену).

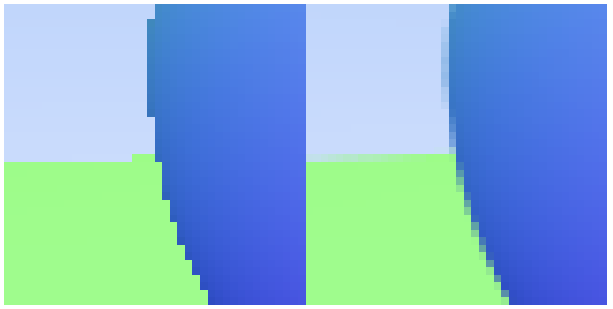
Далее, требуется понять, куда запускать лучи из камеры в пространстве.



Генерация лучей из камеры [2]

Заранее задаются параметры камеры: размер изображения, количество пикселей и т.д. На этом этапе мы создаем Viewport – матрица пикселей изображения, расположенная между камерой и сценой. На ней мы явно выделяем пиксели. Чтобы получить цвет пикселя требуется запустить луч через этот пиксель.

Здесь мы и встречаемся с первым примером использования случайных чисел: если запускать лучи только через центры пикселей (зеленые точки на рисунке), то мы можем неправильно вывести цвет для данного пикселя. Самый простой пример такого явления – рендеринг краев объектов, например шара, когда мы запускаем один луч через центр, мы получаем только один цвет. Допустим луч попал в шар, то есть, грубо говоря, функция вернет цвет шара, а луч через следующий пиксель, уже не попадет в шар и вернет цвет фона. Получается грубая граница объекта (левая картинка). Это отличается от того, как мы видим это в жизни. Здесь на помощь приходит эффект сглаживания (Antialiasing): будем запускать не один луч через центр пикселя, а несколько через разные точки внутри пикселя. Чтобы получить эти точки нам нужно взять случайные координаты внутри пикселя, затем нужно просто взять среднее значение от получившихся цветов и записать в текущий пиксель.

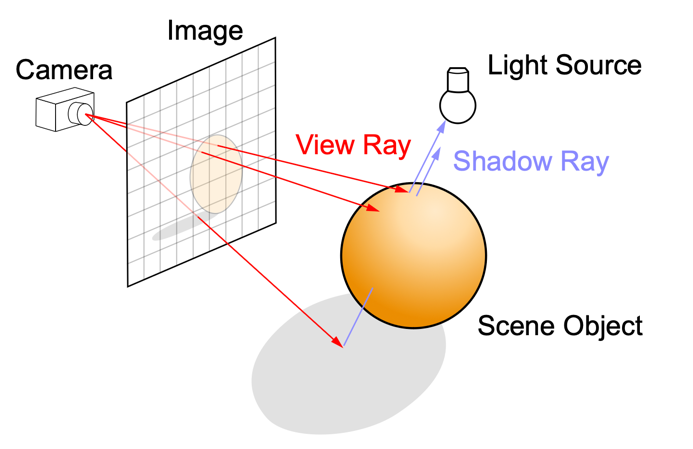


Эффект сглаживания (Antialiasing) [2]

(слева из камеры летит один луч через центры пикселей,

справа – несколько лучей, через случайные точки внутри пикселя)

После запуска луча из камеры, есть два варианта: 1. Он не достигнет никакого объекта на сцене (сцены бывают разными, необязательно они полностью окружены объектами); 2. Луч пересечет какой-то объект. Во втором варианте, в зависимости от свойств материала объекта, может произойти отражение или преломление, все это выполняется в точности, по законам физики. Во время такого пересечения луч запоминает цвет объекта, а также освещенность от источников света в данной точке, и продолжает свое движение. Далее, может быть либо достигнут предел количества отражений (его устанавливают для уменьшения времени работы алгоритма), либо достигнутым объектом станет источник света.

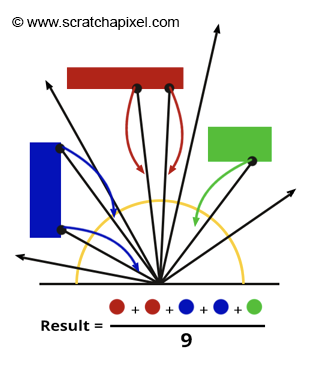


Пример лучей из камеры и получения цвета пикселя [3]

(View Ray – это лучи, которые мы запустили из камеры,

Shadow Ray – это лучи, направленные от точки пересечения к источнику света, чтобы узнать освещенность в текущей точке.)

Все полученные данные в ходе отражений и преломлений луча в дальнейшем используются в Уравнении рендеринга, и мы получаем итоговый цвет пикселя. Затем мы повторяем эту операцию, но уже с другими случайными числами: координатами точки в пикселе, через которую летит луч; направлением луча при отражении от Ламбертова материала и т.д.



Получение цвета пикселя, исходя из нескольких

запущенных лучей в разные стороны [4]

(берется среднее значение из цветов всех запущенных лучей)

В результате всех вышеописанных действий мы и получаем итоговый результат – срендеренное изображение. Теперь чуть больше углубимся в процесс использования случайных чисел.

## Случайные числа в алгоритме Path tracing

При рендеринге изображений случайные числа используются на нескольких этапах, каждое использование случайного числа в отдельном случае мы будем называть dimension (размерность):

* Выбор начальной точки для запуска луча через пиксель (два разных дименшена – координата по Ox и по Oy)
* Выбор направления отраженного луча от Ламбертова материала (два разных дименшена – координата по Ox и по Oy)

Также зачастую случайные числа используются еще в нескольких местах:

* Выбор источника света (один дименшен, отвечающий за индекс)
* Два дополнительных дименшена, отведенных под материалы (в зависимости от сложности материалов в сцене, в моем проекте используются только два из них, для выбора направления луча)

Соответственно, с этими случайными числами мы и будем работать.

# Описание решения

TODO: написать почему методы выбора случайных чисел полезны для нас

Среди различных подходов к генерации случайных чисел были выбраны: Halton sampler, Scrambled Halton sampler and BlueNoiseSampler. Также для проведения сравнения, использовался встроенный генератор случайных чисел из C++, так как по умолчанию в проектах чаще всего используется именно он.

Наша цель - сгенерировать хорошо распределенный рандомизированный набор точек, то есть набор точек, в котором почти отсутствуют точки, находящиеся очень близко друг к другу, и при этом также нет и регионов на координатной плоскости без точек.

## Генерация случайных чисел с использованием последовательности Хальтона

Первый алгоритм, который нам подходит - генератор случайных чисел Хальтона. Он генерирует последовательность точек с низким расхождением, которые последовательно хорошо распределены. Генерации таких точек основан на развороте битов числа (radical inverse) в некоторой системе счисления по основанию какого-либо простого числа. Для разных дименшенов используются разные простые числа.

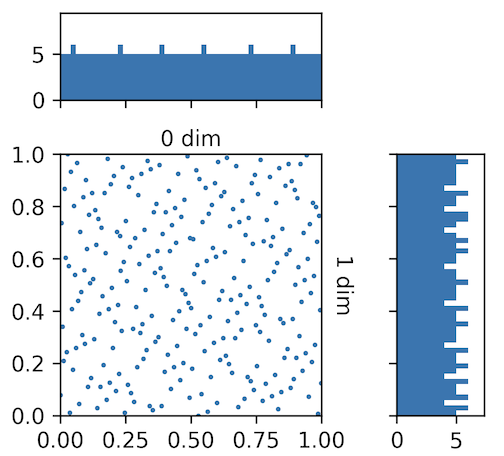
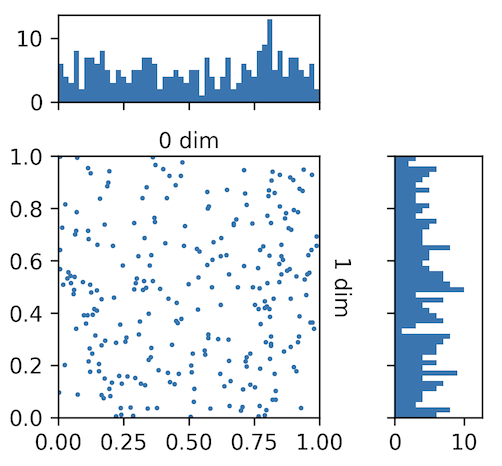
Последовательность с низким расхождением – последовательность в которой доля элементов, попавших в произвольное множество B, пропорциональна усредненному значению некоторой меры множества B.

То есть, если у нас было число , где di – i-ый разряд числа (цифра от нуля до максимальной цифре в некоторой системе счисления b), то после переворота битов у нас получится

Более формально:

Тогда

Как можно заметить после переворота нашего числа мы получаем новое число, которое лежит в полуинтервале [0; 1).

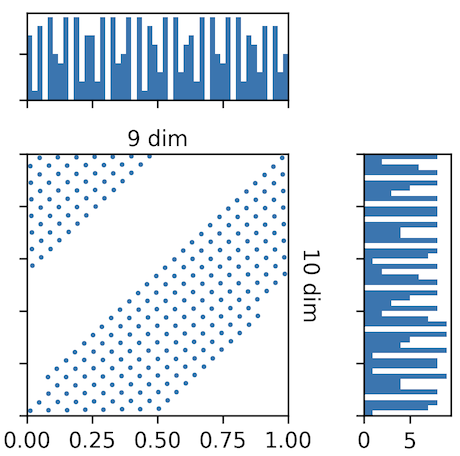


Слева – точки, сгенерированные стандартным генератором

Справа – Halton Sampler (нулевой и первый дименшен)

Построим графики распределения точек в первых двух дименшенах. Явно видно, что последовательность Хальтона более хорошо покрывает плоскость. То же самое показывают графики справа и сверху, которые демонстрируют распределение точек каждого отдельного дименшена.

Однако, для следующих дименшенов ситуация резко меняется:



Halton Sampler для дименшенов 9 и 10

К 9 и 10 дименшену мы получаем следующую ситуацию: точки становятся строго структурированными и появляется явно различимый паттерн. Для наших задач это недопустимо, так как нам требуется чтобы для разных пикселей были разными и точки.

## Генерация случайных чисел с использованием последовательности Хальтона

Чтобы убрать появляющиеся паттерны в графике используется скрэмблинг, то есть перемещивание каким-то образом получающихся значений. Одним из наиболее простых и понятных является RandomDigit скрэмблинг – добавление некоторого заранее сгенерированного массива, где каждая цифра b-ичной системы счисления взаимно-однозначно переходит в другую цифру той же системы счисления.

# Библиографический список

<https://blogs.nvidia.com/blog/what-is-path-tracing/>

<https://raytracing.github.io/books/RayTracingInOneWeekend.html>

<https://developer.nvidia.com/discover/ray-tracing#:~:text=This%20reverse%20tracing%20process%20of,light%20sources%20in%20multiple%20directions>.

<https://www.scratchapixel.com/index.html>

<https://www.techspot.com/article/2485-path-tracing-vs-ray-tracing/#:~:text=Traditional%20ray%20tracing%20involves%20calculating,off%20in%20a%20random%20direction>.

<https://dl.acm.org/doi/pdf/10.1145/15922.15902>

<https://raytracing.github.io/books/RayTracingTheNextWeek.html>

<https://raytracing.github.io/books/RayTracingTheRestOfYourLife.html>

<https://pbr-book.org/4ed/contents>

<https://jcgt.org/published/0009/03/02/>

<https://psychopath.io>

<https://www.scratchapixel.com/index.html>

<https://developer.nvidia.com/discover/ray-tracing#:~:text=Ray%20tracing%20generates%20computer%20graphics,back%20to%20the%20light%20sources>.

**Blue-noise Dithered Sampling** Iliyan Georgiev Marcos Fajardo