МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»

Кафедра №806 «Вычислительная математика и программирование»

**Курсовой проект**

**по курсу ««Параллельная обработка данных»»**

**Обратная трассировка лучей (Ray Tracing) на GPU**

Выполнил: *А.Е. Аксенов*

Группа: *8О-408Б*

Преподаватели: К.Г. Крашенинников,

А.Ю. Морозов

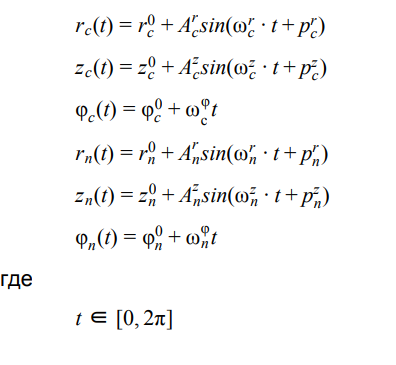
Москва, 2023

**Условие**

**Цель работы.** Использование GPU для создание фотореалистической визуализации. Рендеринг полузеркальных и полупрозрачных правильных геометрических тел. Получение эффекта бесконечности. Создание видеоролика

**Сцена.** Прямоугольная текстурированная поверхность (пол), над которой расположены три Платоновых тела. Сверху находятся несколько источников света. На каждом ребре многогранника располагается определенное количество точечных источников света. Грани тел обладают зеркальным и прозрачным эффектом. За счет многократного пере отражения лучей внутри тела, возникает эффект бесконечности.

**Камера.** Камера выполняет облет сцены согласно определенным законам. В цилиндрических координатах (r, φ, z) , положение и точка направления камеры в момент времени t определяется следующим образом:



Требуется реализовать алгоритм обратной трассировки лучей (http://www.ray-tracing.ru/) с использованием технологии CUDA. Выполнить покадровый рендеринг сцены. Для устранения эффекта «зубчатости», выполнить сглаживание (например, с помощью алгоритма SSAA). Полученный набор кадров склеить в видеоролик любым доступным программным обеспечением. Подобрать параметры сцены, камеры и освещения таким образом, чтобы получить наиболее красочный результат. Провести сравнение производительности gpu и cpu (т.е. дополнительно нужно реализовать алгоритм без использования CUDA).

Программа должна принимать на вход следующие параметры:

1. Количество кадров.
2. Путь к выходным изображениям. В строке содержится спецификатор %d, на место которого должен подставляться номер кадра. Формат изображений соответствует формату, описанному в лабораторной работе 2.
3. Разрешение кадра и угол обзора в градусах по горизонтали.
4. Параметры движения 
5. Параметры тел: центр тела, цвет (нормированный), радиус (подразумевается радиус сферы, в которую можно было бы вписать тело), коэффициент отражения, коэффициент прозрачности, количество точечных источников света на ребре.
6. Параметры пола: четыре точки, путь к текстуре, оттенок цвета и коэффициент отражения.
7. Количество (не более четырех) и параметры источников света: положение и цвет.
8. Максимальная глубина рекурсии и квадратный корень из количества лучей на один пиксель (для SSAA).

Программа должна поддерживать следующие ключи запуска:

* --cpu Для расчетов используется только центральный процессор
* --gpu Для расчетов задействуется видеокарта
* --default В stdout выводится конфигурация входных данных (в формате, описанном ранее) при которой получается наиболее красочный результат, после чего программа завершает свою работу.

Запуск программы без аргументов подразумевает запуск с ключом --gpu. В процессе работы программа должна выводить в stdout статистику в формате: {номер кадра}\t{время на обработку кадра в миллисекундах}\t{общее количество лучей}\n

**Программное и аппаратное обеспечение**

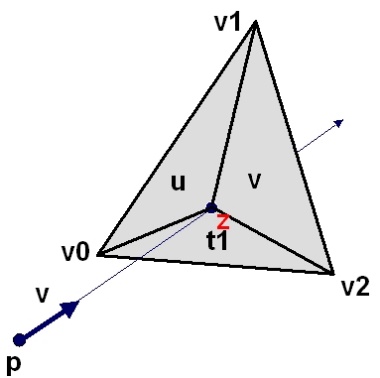
|  |  |
| --- | --- |
| Графический процессор | |
| GPU | NVIDIA RTX 2070 |
| Compute Capability | 7.5 |
| Объем видеопамяти | 8192 МБ |
| Тип видеопамяти | GDDR6 |
| Разделяемая память | 48 КБ |
| Константная память | 64 КБ |
| Число регистров в блоке | 65636 |
| Максимальное число блоков | 16 |
| Максимальное число нитей | 1024 |
| Количество мультипроцессоров | 34 |
| Центральный процессор | |
| СPU | Intel Core i5 9600K |
| Число физических ядер | 6 |
| Максимальное число потоков | 6 |
| Базовая частота | 3.6 GHz |
| Частота Turbo Boost | 4.5 GHz |
| Объем кэша L3 | 9 МБ |
| Оперативная память | |
| Тип оперативной памяти: | DDR4 |
| Частота памяти | 2666MHz |
| Объём | 32 ГБ |
| Постоянная память | |
| Тип постоянной памяти | SSD |
| Объём постоянной памяти | 1024ГБ |
| Программное обеспечение | |
| Операционная система | Microsoft Windows 11 |
| Версия операционной системы | 2004 |
| Среда разработки | Microsoft Visual Studio 2022 |
| Версия среды разработки | 16.7.3 |
| NVIDIA CUDA Toolkit | 12.2 |

**Метод решения**

В целом, решение поставленной задачи осуществляется триангуляцией всех объектов будущей сцены и созданием модели освещения. В текущем варианте в качестве такой модели использовалась модель освещения Фонга. Она дает довольно реалистичные результаты и в то же время отличается простотой реализации. В ней сочетается несколько компонент освещения: фоновый свет (ambient light), диффузный свет (diffuse light) и зеркальный свет (specular light). Далее следует отметить различный характер источников света. В текущем варианте исполнения в качестве источников доступны

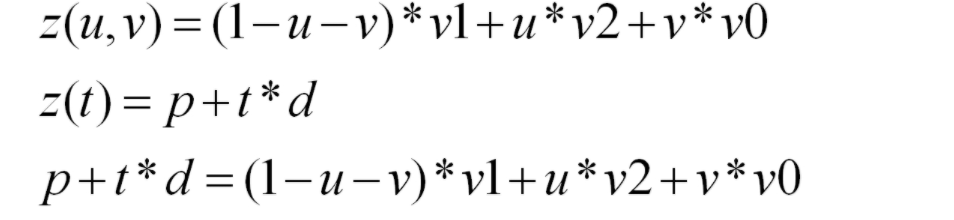
|  |  |
| --- | --- |
| * Point Light | лучи излучаются равномерно в каждом направлении, источник является моделью лампы накаливания |
| * Ambient Light | фоновый свет, интенсивность постоянна и независима от положения точки в пространстве |
| * Diode Light | точечный источник света, интенсивность обратно пропорциональна квадрату расстояния до точки освещения |

Основная задача при рендеринге — вычисление пересечений с треугольниками и вычисление освещенности в точке. Пресечение находится довольно просто, для этого необходимо решить систему и проверить полученные параметры. Основные соображения для построения такой системы приведены на рисунке ниже

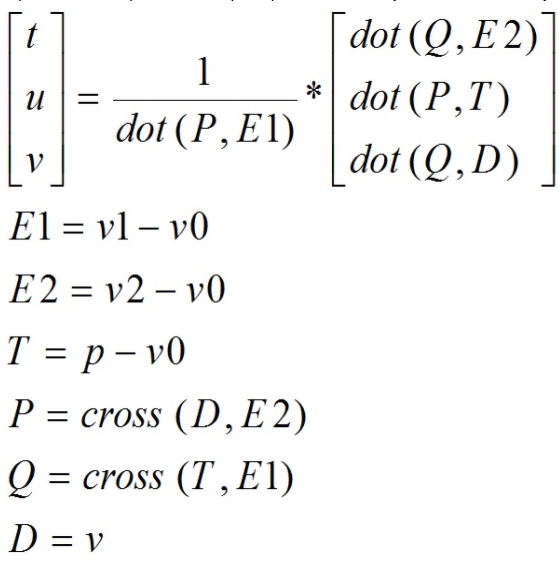


## ***Барицентрический тест (fast minimal storage ray triangle intersection)***

Это самый известный тест на пересечение «луч-треугольник». Имея 3 точки на плоскости, можно выразить любую другую точку через ее барицентрические координаты. Первое уравнение берется просто из определения барицентрических координат, выражая точку пересечения z. С другой стороны, эта же точка z лежит на прямой. Второе уравнение таким образом, это просто параметрическое уравнение прямой. Прировняв правые части уравнений 1 и 2 получаем третье уравнение, которое, по сути, является системой 3-х уравнений (p,v,v1,v2,v3 - векторы) с 3-мя неизвестными (u,v,t).

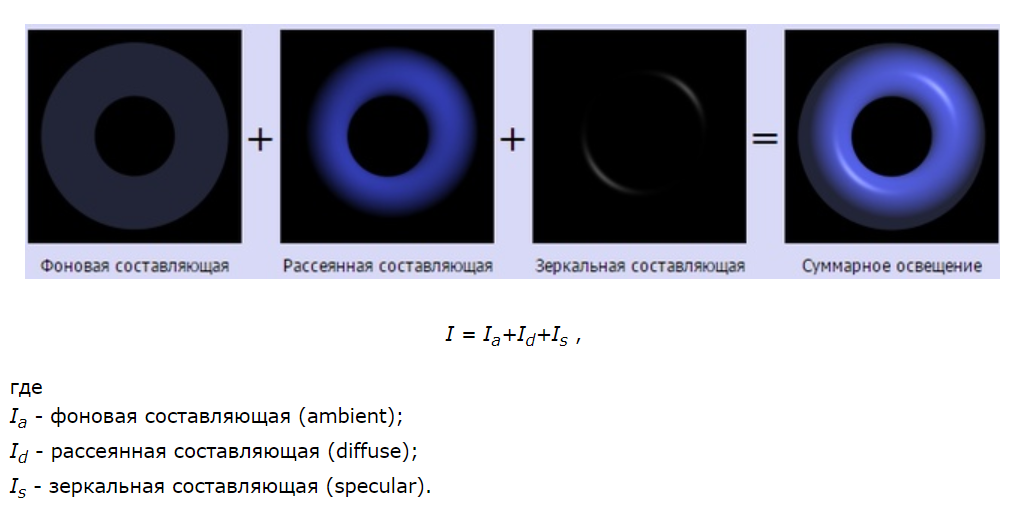


Проведя алгебраические преобразования получим ответ в следующем виде:

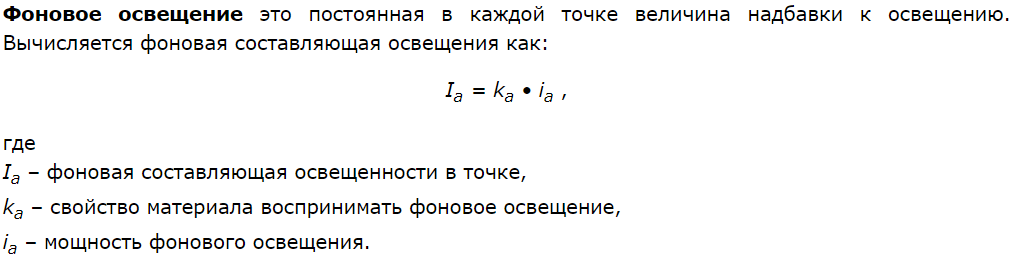


В случае если пересечение действительно имеется, то параметр

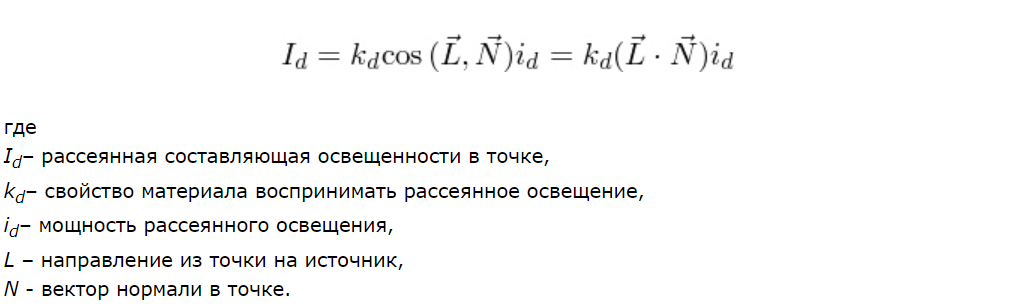
Для расчета освещенности в точке используются три составляющие, при этом учитывается характер видимости источника света.



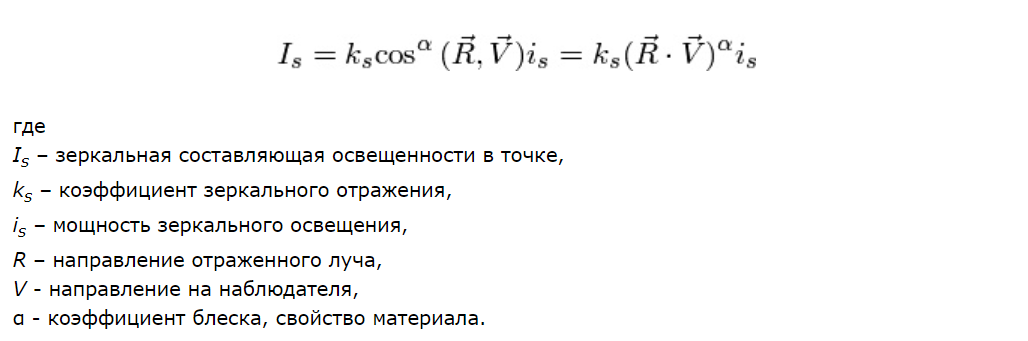
***Фоновое освещение***

**

***Диффузное освещение***

**

***Зеркальное освещение***

****

**Описание программы**

Программа представлена в одном исходном файле. Решение выполнено с использованием языка программирования С++ в соответствии с 17 стандартом. Для решения базовых задач используется стандартные библиотеки STL. Для построения отражений и визуализации тривиальный алгоритм без рекурсии и отражений. Для устранения эффекта зубчатости использует алгоритм SSAA, его реализация выполнена как для запуска на CPU, так и для запуска на GPU. Ниже приведены основные функции рендеринга кадров. Для удобства созданы классы, которые включают в себя перегруженные операторы. Все это позволило разгрузить синтаксические конструкции во всем проекте.

***Функции выполняют рендеринг одного кадра***

void cpuRender(uchar4\* image, size\_t width, size\_t height,

shape\_t\* scene, size\_t n, floor\_t floor,

light\_t \*lights, size\_t light\_sources, size\_t depth,

double angle, vector3d\_t cam, vector3d\_t foc);

\_\_global\_\_

void gpuRender(uchar4\* image, size\_t width, size\_t height,

shape\_t\* scene, size\_t n, floor\_t floor,

light\_t\* lights, size\_t light\_sources, size\_t depth,

double angle, vector3d\_t cam, vector3d\_t foc, tree\_node\_t\* tree);

|  |  |
| --- | --- |
| uchar4\* image | – память, выделенная под изображение |
| size\_t width | – ширина изображения |
| size\_t height | – высота изображения |
| shape\_t\* scene | – массив объектов сцены |
| size\_t n | – количество объектов на сцене |
| floor\_t floor | – структура со данными о поле |
| light\_t \*lights | – массив источников освещения |
| size\_t light\_size | – количество источников освещения |
| size\_t depth | – глубина проникновения луча |
| double angle | – угол обзора камеры |
| vector\_t cam | – точка обзора |
| vector\_t foc | – точка фокуса |

***Функции сглаживания***

void ssaa\_function( uchar4\* image, size\_t image\_width, size\_t image\_height,

uchar4\* ssaa, size\_t width\_ratio, size\_t height\_ratio);

\_\_global\_\_

void ssaa\_kernel( uchar4\* image, size\_t image\_width, size\_t image\_height,

uchar4\* ssaa, size\_t width\_ratio, size\_t height\_ratio);

|  |  |
| --- | --- |
| uchar4\* image | – память для выходного изображения |
| size\_t image\_width | – ширина результата |
| size\_t image\_height | – высота результата |
| uchar4\* ssaa | – массив с входным изображением |
| size\_t width\_ratio | – параметр SSAA, отношение по горизонтали |
| size\_t height\_ratio | – параметр SSAA, отношение по вертикали |

**Исследовательская часть и результаты**

Разрешение изображения: 3840x2160

Параметры ядра : 256, 64

SSAA : 2 x 2

Параметры ядра SSAA : 256, 256

Глубина : ~

Разрешение изображения: 3840x2160

Параметры ядра : 32, ~

SSAA : 2 x 2

Параметры ядра SSAA : 256, 256

Глубина : 5

***Входные данные, на которых получается наиболее красочный результат.***

100

img\_ % d.data

1024 960 100

7.0 3.0 0.0 2.0 1.0 2.0 6.0 1.0 0.0 0.0

2.0 0.0 0.0 0.5 0.1 1.0 4.0 1.0 0.0 0.0

0.0 - 2.5 1.0

1.0 0.0 0.0

2

0.9 0.4

2

2.0 1.5 0.0

0.0 1.0 0.0

2

0.9 0.3

2

- 2.0 1.5 0.2

0.0 0.7 0.7

2

0.9 0.5

2

20 20

0.0 0.0 - 1.0

1 1 1

3

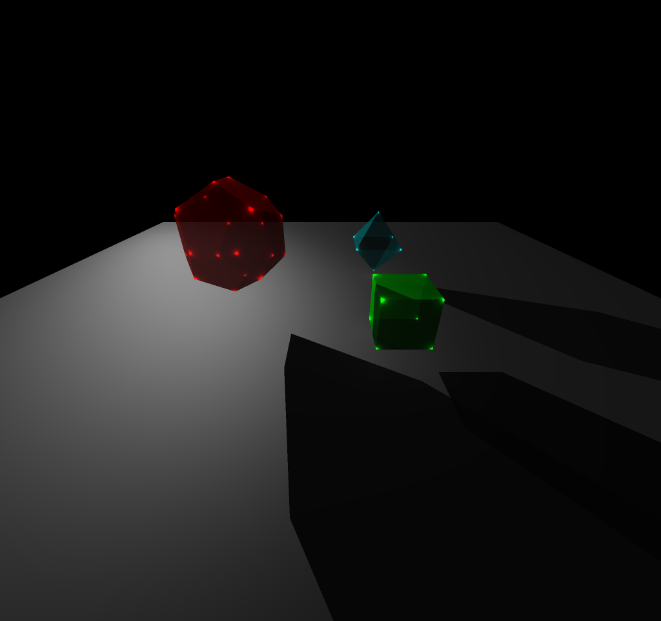
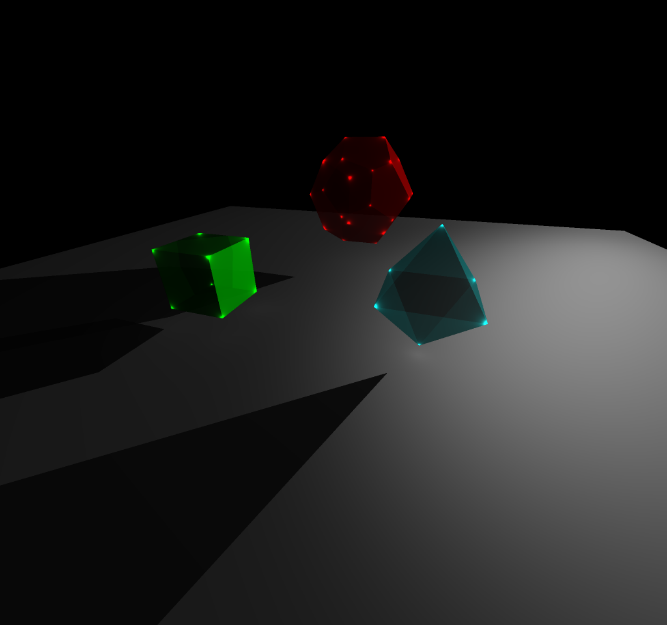
0.01 0.01 0.01

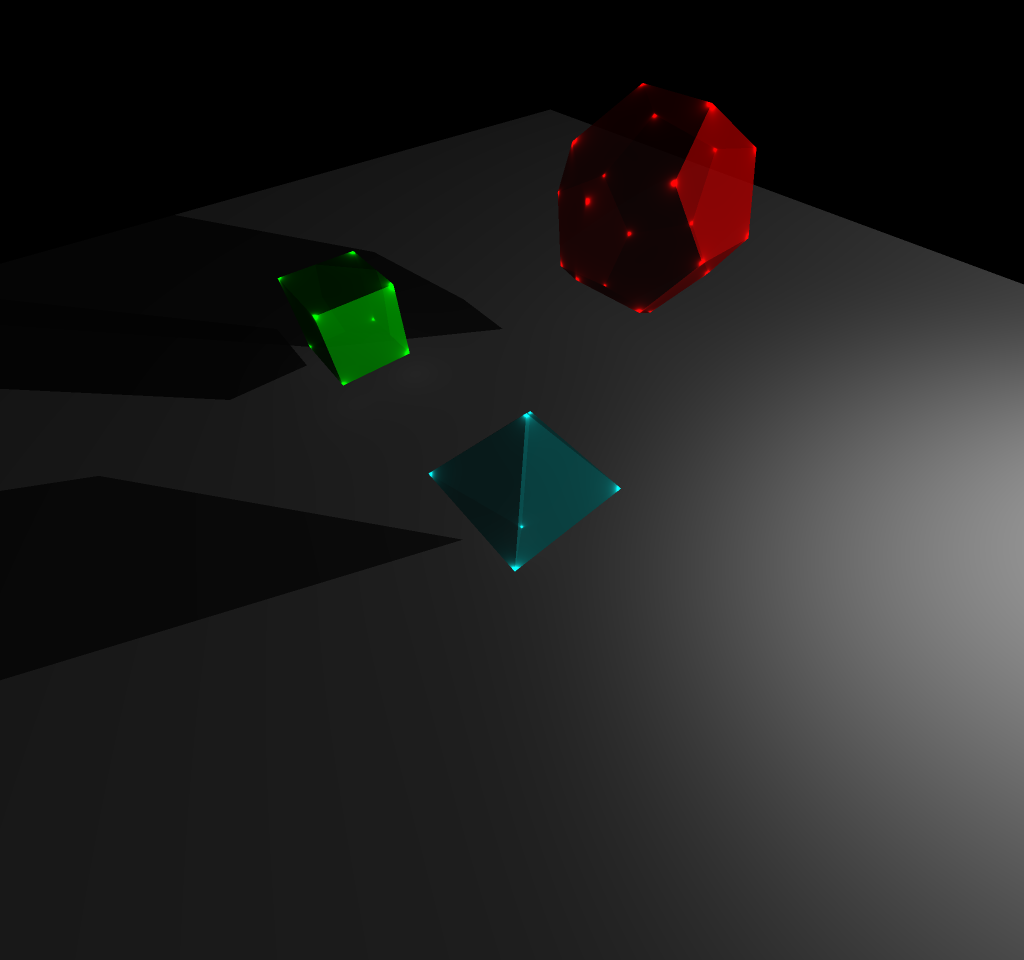
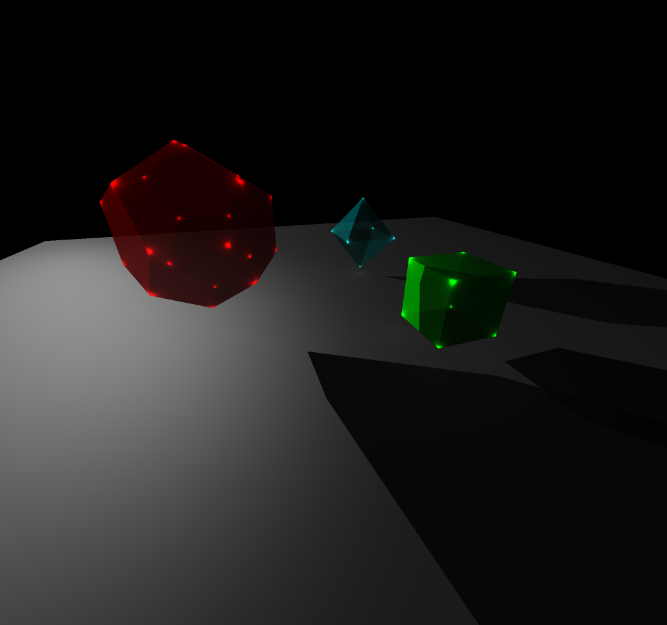
- 10 - 10 3

0.4 0.4 0.4

5 2

***Примеры кадров с разных ракурсов***

** **

** **

**Выводы**

Технология трассировки лучей наиболее широко используется в игровой индустрии для построения реалистичных сцен и изображений. Также ее применяют в продуктах 3D проектирования. Например, Blender имеет свой рендерер, который называется Cycles. В нем используется технология трассировки лучей, что позволяет создавать реалистичные сцены. Все это позволяет получать более детализированное представление о том, как выбор тех или иных материалов, источников света будет сказываться на интерьере

Создание движка рендеринга действительно трудоемкая задача. Выполненная работа позволила оценить сколько усилий потребуется на написание универсального и качественного решения, которое бы учитывало множество свойств и других факторов. Все это потребует тщательного проектирования — создания классов, иерархий и абстракций.

У написанной реализации отсутствует какая-либо гибкость и она во многом неэффективна. Тем не менее, даже для такого результата потребовалось значительное количество усилий. Результат получился без отражений, но всё же довольно реалистичный.

**Литература**

* [Gabriel Gambetta — computer graphics from scratch](https://www.gabrielgambetta.com/computer-graphics-from-scratch/introduction.html)
* [Пересечение луча и треугольника (ray-tracing.ru)](http://ray-tracing.ru/articles213.html)
* [Тени (ray-tracing.ru)](http://ray-tracing.ru/articles203.html)