

**МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)**

**Институт №8 «Компьютерные науки и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»**

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

Обратная трассировка лучей (Ray Tracing) на GPU

**Выполнил: В. В. Бирюков
Группа: 8О-407Б
Преподаватель: 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 анимации.

Вариант: 8. Гексаэдр, Октаэдр, Икосаэдр

Программное и аппаратное обеспечение

Характеристики графического процессора:

- Наименование: GeForce GT 545
- Compute capability: 2.1
- Графическая память: 3150381056 Б
- Разделяемая память на блок: 49152 Б
- Количество регистров на блок: 32768
- Максимальное количество потоков на блок: (1024, 1024, 64)
- Максимальное количество блоков: (65535, 65535, 65535)
- Константная память: 65536 Б
- Количество мультипроцессоров: 3

Характеристики системы:

- Процессор: Intel(R) Core(TM) i7-3770 CPU 3.40GHz
- Память: 15 ГБ
- HDD: 500 ГБ

Программное обеспечение:

- ОС: Ubuntu 16.04.6 LTS
- IDE: Visual Studio Code 1.72
- Компилятор: nvcc 7.5.17

Метод решения

В курсовой работе реализован алгоритм обратной трассировки лучей с фиксированным числом переотражений и без явного использования рекурсии. Основная единица исполнения трассировки лучей — трассировка одного луча (вычисление точки его пересечения со сценой), вычисления цвета в данной точке и создание отраженных или преломленных лучей. Исходные и новые лучи можно хранить в двух массивах и таким образом, одна итерация трассировки всех лучей соответствует одному рекурсивному спуску.

Для вычисления цвета в точке пересечения используется затенение по Фонгу, согласно которому он складывается из цветовых интенсивностей трех компонент освещения: фоновой, рассеянной и бликовой

$$I = I_a + I_d + I_s$$

Фоновая компонента:

$$I_a = K_a i_a$$

где K_a — способность материала воспринимать фоновое освещение, i_a — интенсивность фонового освещения.

Рассеянная компонента:

$$I_d = K_d \cos(\vec{L}, \vec{N}) i_d = K_d (\vec{L} \cdot \vec{N}) i_d$$

где K_d — способность материала воспринимать рассеянное освещение, i_a — интенсивность рассеянного освещения, \vec{L} — направление из точки на источник света, \vec{N} — нормаль в точке.

Бликовая (зеркальная) компонента:

$$I_s = K_s \cos^p(\vec{R}, \vec{V}) i_s = K_s (\vec{R} \cdot \vec{V})^p i_s$$

где K_s — способность материала воспринимать бликовое освещение, p — коэффициент блеска, i_a — интенсивность бликового освещения, \vec{R} — направление отраженного луча, \vec{V} — направление из точки на наблюдателя.

Для создания теней интенсивность источника света в точке корректируется с учетом коэффициента пропускания и цвета материалов, через которые проходит луч на пути к этой точке.

В качестве алгоритма сглаживания используется SSAA — изображение рендерится в N раз большем размере, итоговое изображение получается усреднением цвета в непересекающихся окнах $N \times N$.

Описание программы

Сцена

Все объекты сцены хранятся в различных массивах. А именно сцена содержит: массив материалов, массив вершин, массив треугольников, массив сфер, массив данных об полигональных объектах, массив источников света, массив данных о текстурах и, наконец, сами текстуры так же в одном массиве.

Полигональный объект содержит индексы треугольников, сфер и ограничивающую сферу. Каждый треугольник хранит индексы вершин, индекс материала и текстуры, а также текстурные координаты каждой вершины. Так как для всей объектов одного типа используется сквозная индексация, конкретные индексы (треугольников, вершин, материалов и т.д.) обновляются при добавлении объектов в сцену.

Структура сцены также содержит методы для нахождения пересечения луча, расчета интенсивности света в какой-либо точке и получения пикселя текстуры.

Массивы хранятся в виде указателей, что позволяет одинаково работать со сценой содержащейся как в RAM, так и в видеопамяти.

Необходимые многогранники загружаются из файлов `obj`. Внутренние «источники света» расставляются параллельно линиям, специально оставленных в моделях, и сохраняются в массиве сфер.

Трассировка лучей

Трассировка лучей на GPU реализована без использования рекурсии. Управляющий функция `render` вызывает инициализацию лучей, затем, в цикле, трассировку текущего массива лучей. После каждой итерации массивы лучей меняются местами, и если лучей стало слишком много, происходит перевыделения памяти для результирующего массива.

Инициализация и трассировка выполняются параллельно. При трассировке дочерние лучи записываются на первый свободный индекс массива, который обновляется атомарным сложением; после получения цвета в точке пересечения, он так же добавляется к пиксели изображения атомарно.

Версия для CPU реализована с использованием рекурсии. В функции `render` для каждого пикселя создается луч, трассировку которого производит функция `ray_color`, осуществляя при необходимости рекурсивные вызовы для отраженного или преломленного луча. Функции для определения пересечения реализованы как методы сцены и примитивов. При проверке пересечения с объектом используется небольшая оптимизация — проверка на пересечение с ограничивающей его сферой. Параметры сферы определяются при добавлении объекта в сцену.

Освещение

Основной цвет в точке пересечения вычисляется с использованием затенения по Фонгу, описанного ранее. Для учета теней, интенсивность света вычисляется в каждой точке при помощи трассировки луча от источника до этой точки и аккумулирования коэффициентов прозрачности и цветов материалов при каждом пересечении. Также, каждый луч содержит собственный коэффициент затухания, который обновляется при отражении или преломлении луча и учитывается для итогового цвета.

Текстурирование

Для наложения текстур используются текстурные координаты. Углы прямоугольной текстуры имеют координаты $(1, 1)$, $(1, 0)$, $(0, 0)$, и $(0, 1)$ соответственно. Вершина каждого треугольника имеет свои текстурные координаты. Текстурные координаты точки внутри треугольника определяются при помощи интерполяции текстурных координат вершин с использованием барицентрических координат это точки. Такой способ позволяет текстурировать треугольники с произвольным положением в пространстве.

Результаты

Сравнение времени работы

Измерим время построения двух кадров с различным числом переотражений при различных конфигурациях ядра трассировки и различном разрешении. Остальные параметры фиксированные: два источника света, глубина рекурсии: 5, сглаживание не производится.

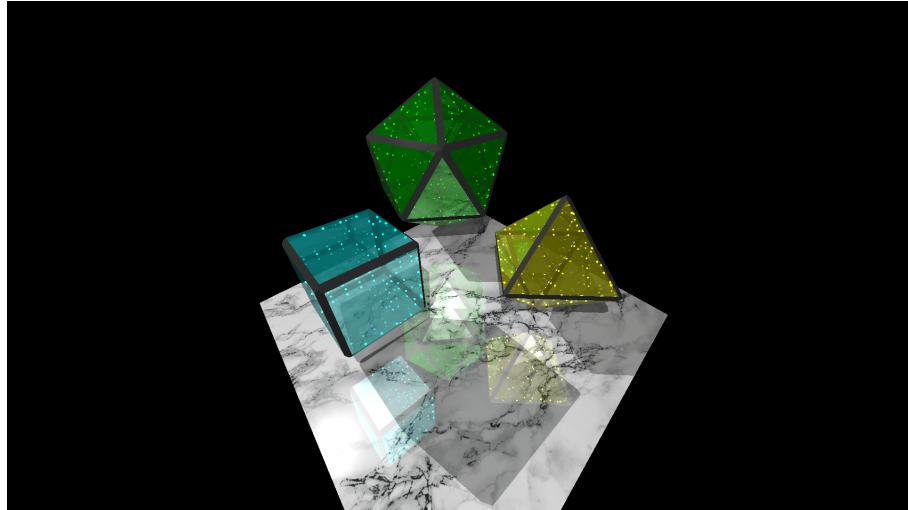
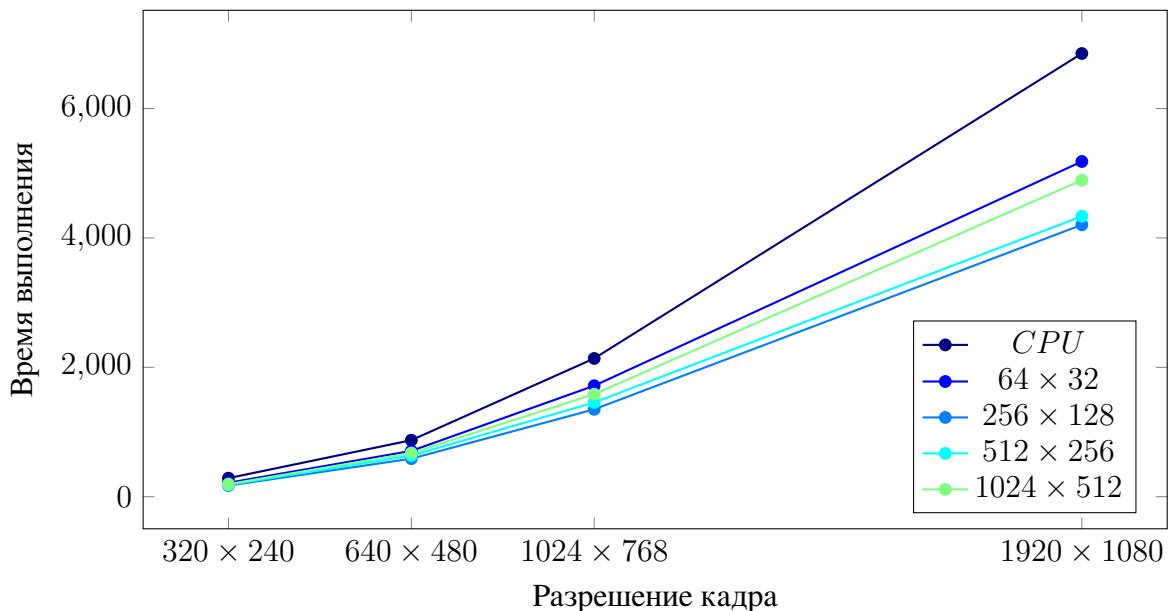


Рис. 1: Кадр с меньшим числом переотражений

Разрешение кадра	320×240	640×480	1024×768	1920×1080
Суммарное число лучей	124891	500508	1281224	3748404
Конфигурация	Время выполнения, мс			
CPU	286.713	873.566	2137.15	6850.65
64×32	215.107	707.831	1715.26	5181.16
256×128	172.041	587.671	1350.38	4202.60
512×256	182.771	635.345	1456.84	4334.43
1024×512	195.109	674.494	1584.68	4892.46



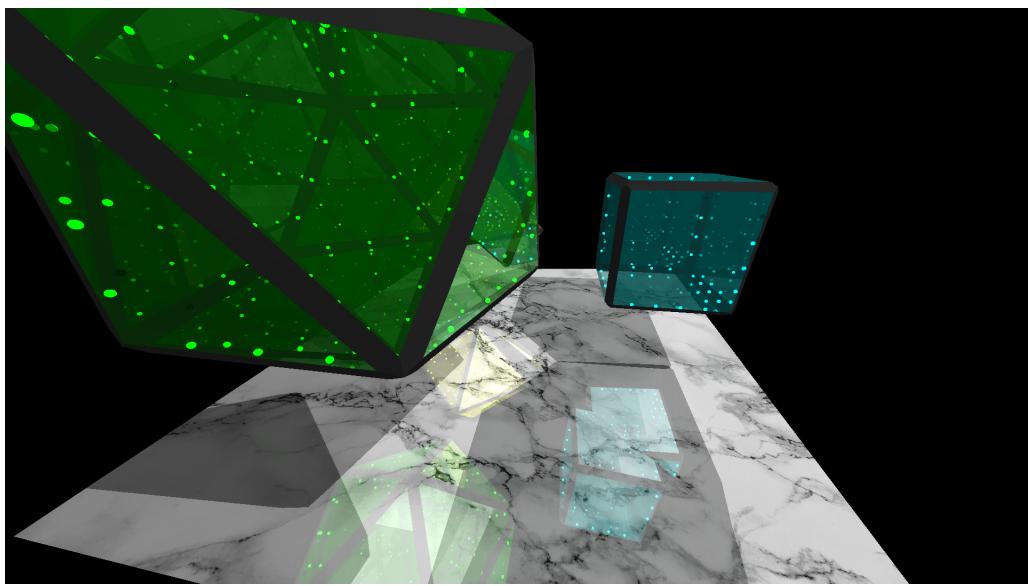
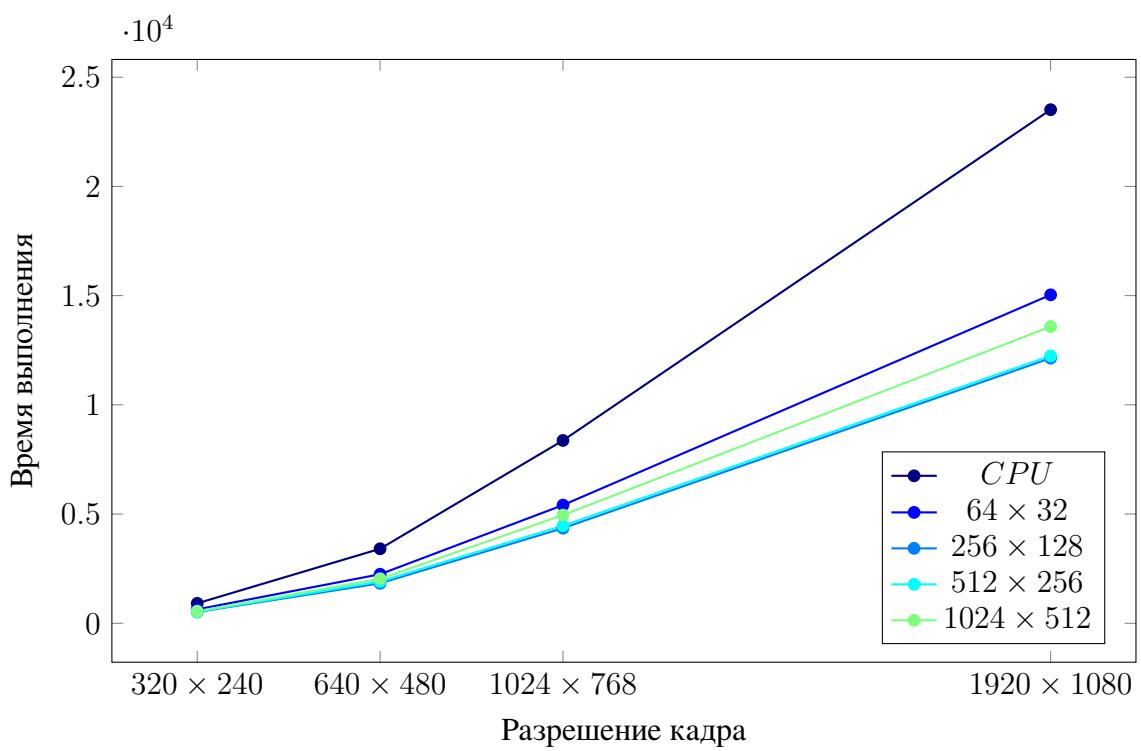


Рис. 2: Кадр с большим числом переотражений

Разрешение кадра	320×240	640×480	1024×768	1920×1080
Суммарное число лучей	231443	926348	2373583	6764725
Конфигурация	Время выполнения, мс			
CPU	912.362	3409.83	8369.37	23513
64×32	635.324	2244.85	5410.10	15036
256×128	516.708	1830.13	4353.47	12137
512×256	527.774	1912.16	4462.98	12241
1024×512	552.080	2038.69	4952.62	13581



Результаты работы

Ниже представлены результаты работы программы для следующей сцены:

```
126  
res/%d.png  
1920 1080 120
```

```
6 5 1.57079632679489 0 2 0 2 1 0 0  
3 0 4.71238898038469 0 0 0 0 1 0 0  
  
2.75 0.29 3 0 0.3 0.3 1.8 0.4 0.4 4  
-1.5 -2 2.5 0.3 0.3 0 2 0.4 0.4 5  
-1.48 2.6 4 0 0.3 0 2.5 0.4 0.4 3  
  
5 5 0 5 -5 0.5 -5 -5 0 -5 5 0.5  
texture.data  
0.8 0.8 0.8 0.5  
  
2  
5 -5 10 0.6 0.6 0.6  
5 5 10 0.6 0.6 0.6  
5 1
```

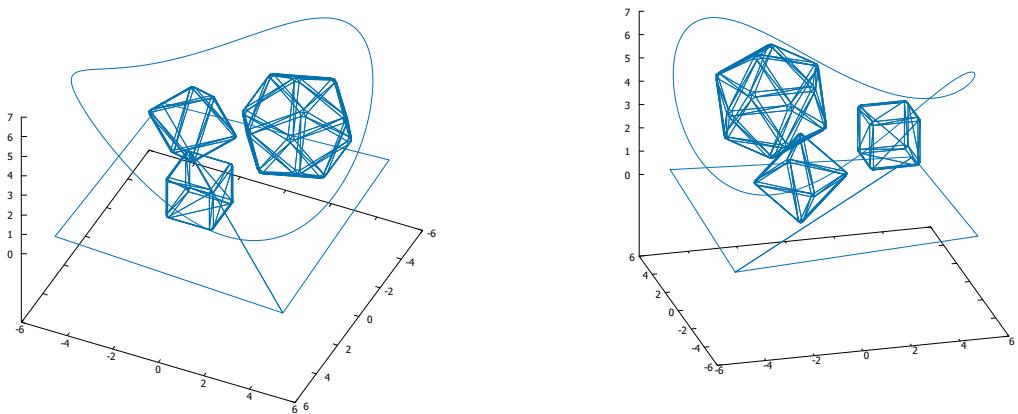


Рис. 3: Схематичное изображение сцены и траектории камеры

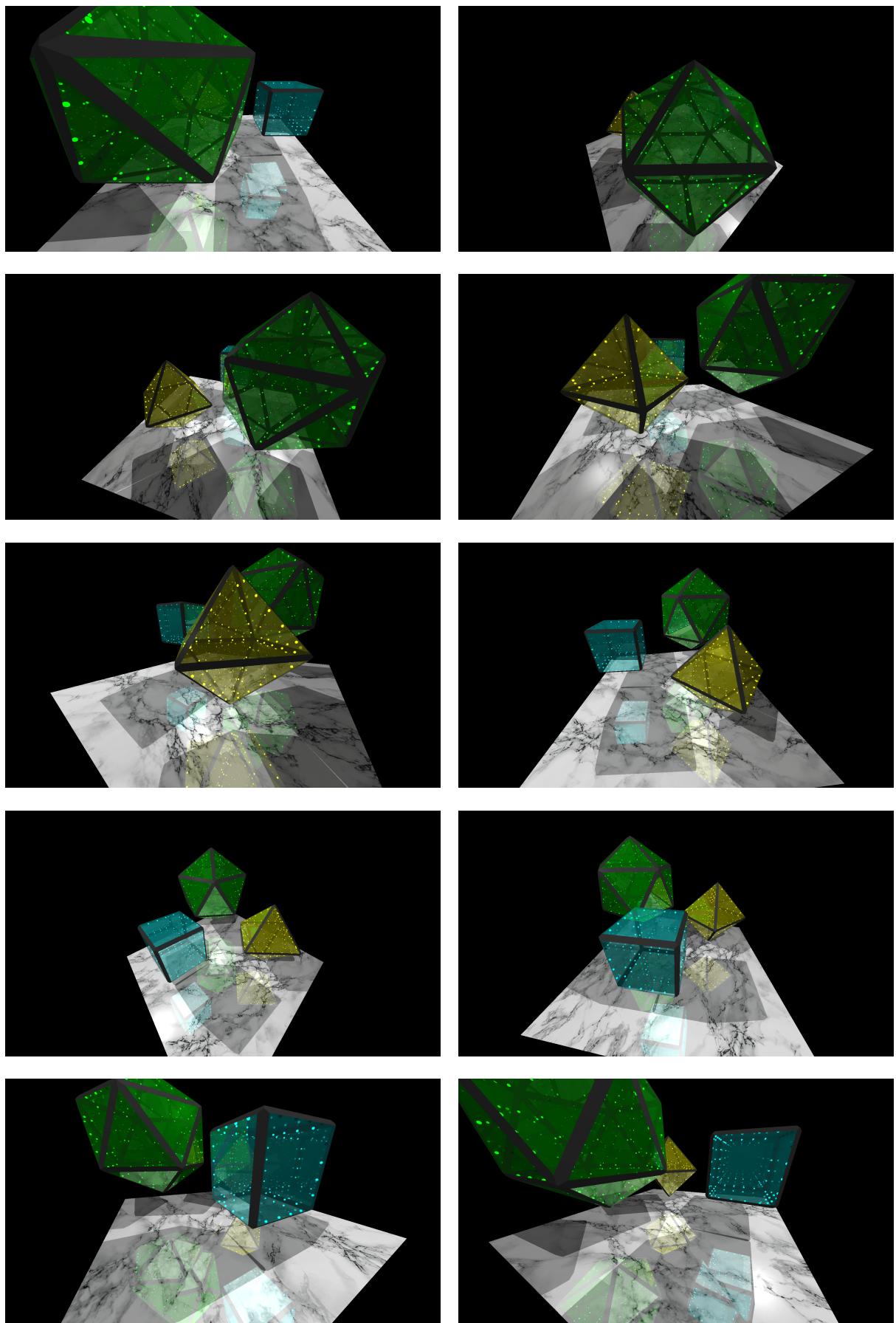


Рис. 4: Равномерно выбранные 10 кадров

Выводы

В ходе выполнения курсовой работы я познакомился с алгоритмом обратной трассировки лучей и его параллельной реализацией. Трассировка лучей позволяет получать фотoreалистичные изображения, но крайне затратна вычислительно. Число лучей, которые необходимо обрабатывать потенциально экспоненциально растет, однако на практике остается порядка числа пикселей изображения, что все равно довольно много при большом разрешении или использовании сглаживания.

Одной из самых сложных частей работы было добиться того, чтобы максимальный объем программы не зависел от версии трассировки и параллельно постоянно добавлять все новый и новый функционал для соответствия требованиям.

Результаты тестирования вполне ожидаемы. Даже небольшое распараллеливание хорошо ускоряет трассировку.

Список литературы

- [1] *Ray-tracing.ru*
URL: <http://www.ray-tracing.ru/>.
- [2] *Ray Tracing in One Weekend — The Book Series*
URL: <https://raytracing.github.io/>.
- [3] *Компьютерная графика — Модель отражения Фонга.*
URL: http://compgraphics.info/3D/lighting/phong_reflection_model.php.