МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ

(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Компьютерные науки и прикладная математика»

Кафедра №806 «Вычислительная математика и программирование»

**Курсовой работа**

**по курсу «Параллельная обработка данных»**

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

Выполнил: А.А. Почечура

Группа: 8О-406Б

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 анимации.

**Задание.** Требуется реализовать алгоритм обратной трассировки лучей с использованием технологии CUDA. Выполнить покадровый рендеринг сцены. Для устранения эффекта «зубчатости», выполнить сглаживание (например с помощью алгоритма SSAA). Полученный набор кадров склеить в анимацию 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).

**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 переотражения лучей внутри тела, возникает эффект бесконечности.

**Вариант 8**

На сцене должны располагаться следующие три тела: Гексаэдр, Октаэдр, Икосаэдр.

**Программное и аппаратное обеспечение**

**Графический процессор:**

Compute capability: 6.1

Name: NVIDIA GeForce GTX 1050

Total Global Memory: 4294967296 bytes

Shared memory per block: 49152

Registers per block: 32768

Warp size: 32

Max threads per block: (1024, 1024, 64)

Max block: (2147483647, 65535, 65535)

Total constant memory: 65536

Multiprocessors count: 40

**Центральный процессор:**

vendor\_id : GenuineIntel

cpu family: 6

model: 158

model name: Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz

stepping: 9

microcode : 0x42

cpu MHz: 1059.228

cache size : 6144 KB

physical id: 0

siblings: 8

core id: 0

cpu cores : 4

apicid: 0

initial apicid: 0

fpu: yes

fpu\_exception: yes

cpuid level: 22

wp : yes

bogomips : 5616.00

clflush size: 64

cache\_alignment : 64

address sizes: 39 bits physical, 48 bits virtual

**Оперативная память и жёсткий диск:**

*Оперативная память:*

Тип оперативной памяти: DDR4

Частота оперативной памяти: 2400 МГц

Размер оперативной памяти: 8 ГБ

Количество слотов под модули памяти: 2

Максимальный объем памяти: 32 Гб

*Жёсткий диск:*

Общий объём жестких дисков (HDD): 1000 ГБ

Общий объем твердотельных накопителей (SSD): 128 Гб

**Программное обеспечение:**

Операционная система: Windows 10

Оболочка: Visual Studio 2022

Компилятор: Nvidia CUDA Compiler C++ 2011

**Метод решения**

Перед трассировкой лучей, мы должны определить, как будет двигаться наша камера, куда она будет смотреть и что перед собой видеть. Для параметров движения и ракурса камеры используются формулы, представляющие траекторию движения с помощью цилидрических координат. Все поверхности в данной работе будут представляться в виде полигонов (в данном случае это будут треугольники), поэтому, чтобы определить, что сейчас видит камера, необходимо понять, какой полигон был пересечён вектором направления взгляда первым. Для этого используется метод барицентрических координат.

Далее производится трассировка лучей, которая делится на несколько этапов. Первый из них – создание фотореалистичного освещения. Для данных целей использовалась модель Фонга, которая представляет свет в виде суммы трёх составляющих: фоновая, рассеянная и зеркальная. Данная модель позволяет изобразить блики, тень, фоновое освещение и перепады яркости света в зависимости от расстояния объекта до источника.

Каждый луч, попадая на поверхность, создаёт после себя два новых луча – отражённый луч (меняет направление согласно отражению) и луч, прошедший сквозь поверхность (сохраняет направление порождающего луча). Значения цвета этих лучей зависят от параметров поверхности, от которой происходит отражение или преломление (в данном случае это будет не совсем преломление). Количество возможных отражений / преломлений луча зависит от параметра максимальной рекурсии, которая задаётся пользователем.

Далее производится сглаживание изображения при помощи алгоритма SSAA. Работает он следующим образом: изначально генерируется изображение большего размера, чем был запланирован. Затем изображение сжимается до требуемых размеров, усредняя пиксели по периметру квадрата, сторона которого соответствует коэффициенту сглаживания.

**Описание программы**

Для работы с данными используются следующие структуры:

* *vec3* – представление вектора в пространстве;
* *uchar4 –* представление параметров цвета в диапазоне 0 до 255;
* *double4 -* представление параметров цвета в диапазоне от 0 до 1;
* *trig* – представление полигона (три точки треугольника) и его характеристики (прозрачность, коэффициент отражения, цвет и т.д.);
* *hit* – используется для передачи данных о точке, в которую попал луч (координаты, нормаль к плоскости, номер полигона);
* *ray* – данные о луче (точка испускания, направление, коэффициент затухания);
* *light –* данные об источнике света (положение, цвет).

Со следующими массивами происходила непосредственная работа на

протяжении всей трассировки лучей:

* *trigs –* массив всех полигонов;
* *lights* – массив источников света, расположенных на сцене;
* *floor\_text* – текстура пола в бинарном виде.

Реализация на ГПУ и ЦПУ почти идентичные за исключением некоторых

моментов, поэтому приведу описание реализации алгоритма только на графическом процессоре.

Изначально мы считывает все необходимые параметры и вызываем функцию *build\_space,* в которой производится создание всех полигонов (фигур и пола), а также считывается текстура, которая будет накладываться на пол.

Далее происходит трассировка лучей. Сначала вызывается функция *render\_gpu,* в которой происходит подготовка данных, вызов всех используемых ядер, а также получение итогового результата. Сперва вызывается ядро *init\_rays\_n\_data,* в которой происходит первичная инициализация выпущенных лучей, а также подготовка результирующих данных, куда мы будем прибавлять значения всех лучей. Далее в цикле вызывается ядро *ray\_trace\_gpu*, в котором происходит вся трассировка: вызов функции *ray\_gpu* (определение полигона и места, в которое попал луч), задание всех трёх типов освещения, отрисовка источников света на рёбрах, отображение текстуры пола (реализуется с помощью представления точки попадания в базисе полигона пола), рождение двух новых лучей из текущего луча (отражённого и проходящего сквозь полигон). Цикл проделывает итерации до тех пор, пока порождённых лучей не останется, либо если будет достигнут предел рекурсии. После получения всех требуемых значений вызывается ядро *write\_data*, в котором реализуется алгоритм сглаживания SSAA и данные приводятся в нужный нам диапазон. Вышеописанная последовательность действий проделывается для каждого кадра отдельно.

**Исследовательская часть и результаты**

Для начала приведу мои замеры на ГПУ. Значения в таблице: *a, b*, где *a* – суммарное время, а *b* – суммарное количество лучей. Квадратный корень из количества лучей на один пиксель для сглаживания равно 2 (во всех замерах). Построение производится ровно одного кадра. Время указано в миллисекундах.

Количество потоков: << < 512, 512 >> > (<< < dim3(16, 16), dim3(16, 16) >> >)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер изображения | Глубина рекурсии | 2 | 6 | 12 |
| 640 × 480 | | 554.985779 ms, 2004047 | 1440.645630 ms, 3944928 | 3594.979492 ms,  8200757 |
| 1280 × 720 | | 1656.132324 ms,  6172390 | 4339.322266 ms,  12250597 | 11031.974609 ms,  25557084 |
| 1920 × 1080 | | 3631.490234 ms,  13889794 | 9624.229492 ms,  27580135 | 24322.208984 ms,  57542352 |

Количество потоков: << < 256, 256 >> > (<< < dim3(8, 8), dim3(16, 16) >> >)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер изображения | Глубина рекурсии | 2 | 6 | 12 |
| 640 × 480 | | 617.802673 ms, 2004047 | 1622.287720 ms,  3944928 | 3802.429443 ms,  8200757 |
| 1280 × 720 | | 2061.391653 ms,  6172390 | 4820.062616 ms,  12250597 | 13682.794012 ms,  25557084 |
| 1920 × 1080 | | 4012.472918 ms,  13889794 | 10624.194817 ms,  27580135 | 28391.672819 ms,  57542352 |

Количество потоков: << < 1024, 1024 >> > (<< < dim3(32, 32), dim3(16, 16) >> >)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер изображения | Глубина рекурсии | 2 | 6 | 12 |
| 640 × 480 | | 382.481941 ms, 2004047 | 1148.294810 ms, 3944928 | 2485.391984 ms,  8200757 |
| 1280 × 720 | | 957.582910 ms,  6172390 | 3784.104929 ms,  12250597 | 7910.449281 ms,  25557084 |
| 1920 × 1080 | | 3094.301945 ms,  13889794 | 8742.583201 ms,  27580135 | 209531.596033 ms,  57542352 |

Теперь приведу замеры на ЦПУ. Все характеристики оставил те же.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Размер изображения | Глубина рекурсии | 2 | 6 | 12 |
| 640 × 480 | | 35478.066406 ms, 2004047 | 99636.472479 ms,  3944928 | 233610.428392  ms,  8200757 |
| 1280 × 720 | | 117576.371297 ms,  6172390 | 303731.952355 ms,  12250597 | 794232.539139  ms,  25557084 |
| 1920 × 1080 | | 236015.950139 ms,  13889794 | 9624.229492 ms,  27580135 | 1556608.395716 ms,  57542352 |

Очевидно, что на центральном процессоре вычисления выполняются намного дольше, чем на ГПУ. На графическом процессоре количество потоков также очень сильно влияет на скорость работы алгоритма. Увеличение времени из-за увеличения максимальной глубины рекурсии объясняется тем, что количество лучей сильно возрастает. При увеличении размеров изображения число лучей также значительно повышается.

Конфигурация входных данных, на которых получается наиболее красочный результат (вывод программы с аргументом --default):

450

res/%03d.data

1920 1080 120

4.5 0.0 0.0 0.5 1.0 1.0 1.0 1.0 0.0 0.0

2.0 0.0 0.0 1.0 0.1 1.0 1.0 1.0 0.0 0.0

-3.0 0.0 0.0 1.0 0.0 0.0 1.73 0.6 0.5 10

1.0 -2.0 0.0 0.0 1.0 0.0 1.41 0.2 0.5 5

0.0 2.0 1.0 0.0 1.0 1.5 2.0 0.45 0.9 4

-6.0 -6.0 -1.0 -6.0 6.0 -3.0 6.0 6.0 -1.0 6.0 -6.0 -3.0 in.data 0.75 0.75 0.75 0.5

4

-7 0 10 1.0 0.0 0.0

0 -7 10 0.0 1.0 0.0

0 7 10 0.0 0.0 1.0

7 0 10 1.0 1.0 1.0

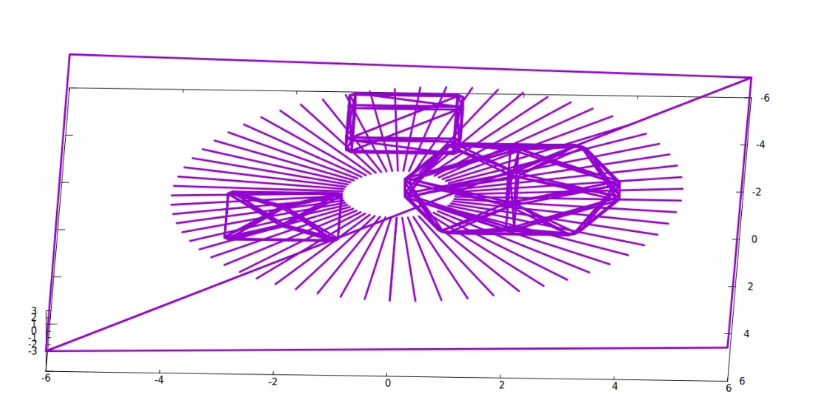
10 2

Трёхмерные графики из gnuplot. Линии показывают положение камеры и вектор направления обзора (к центру):

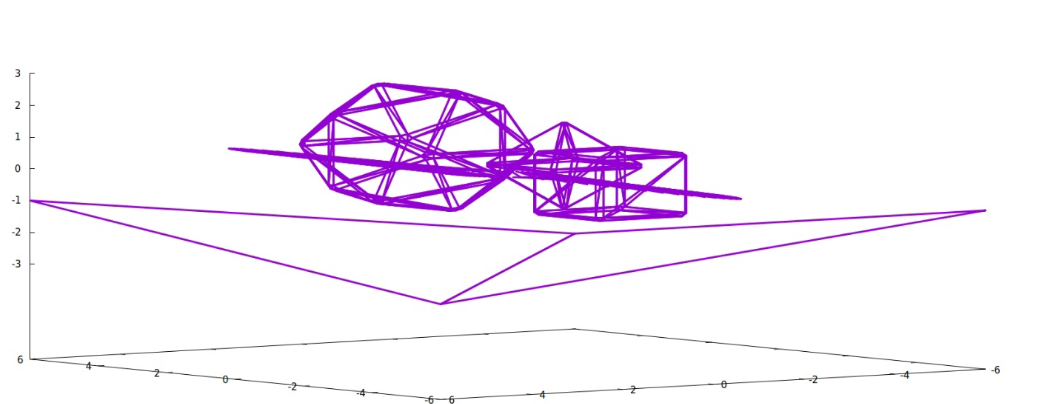
Вид спереди:



Вид сверху:

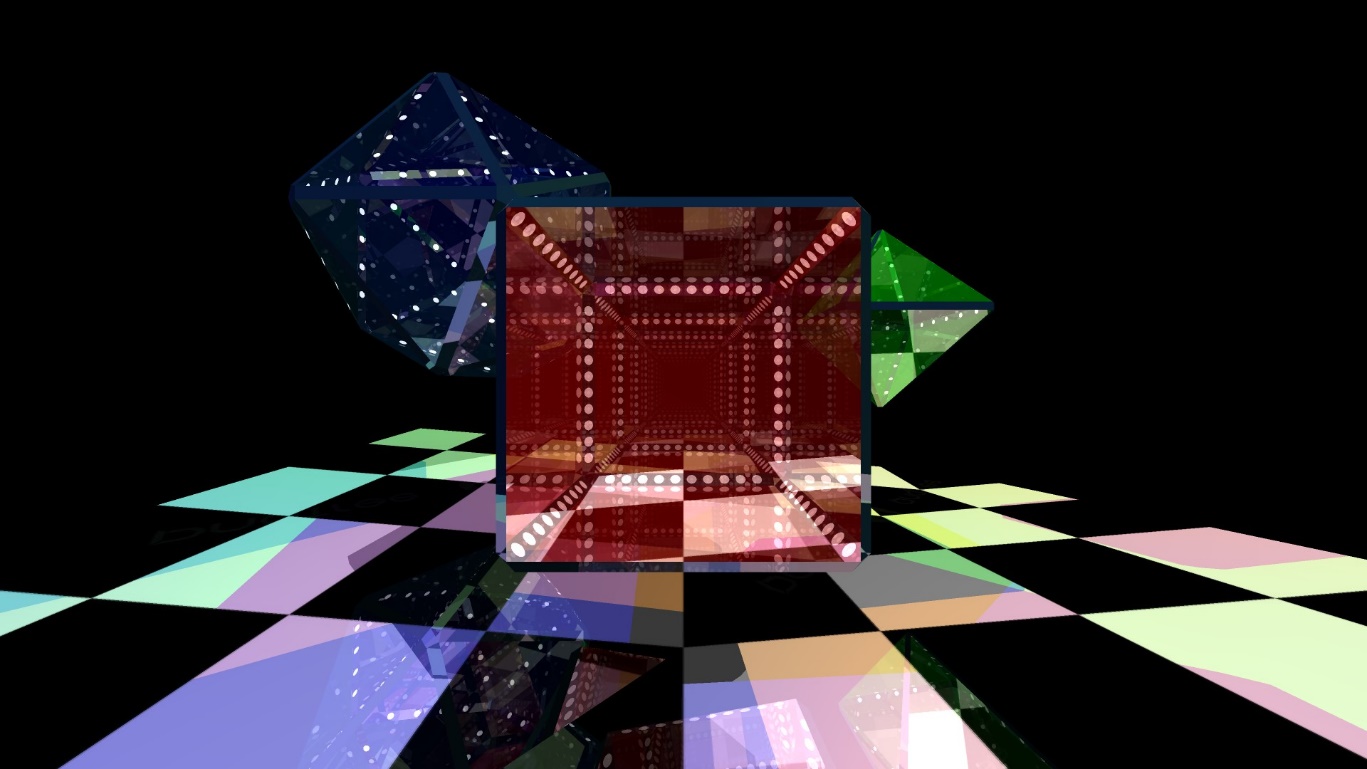


Вид сбоку:

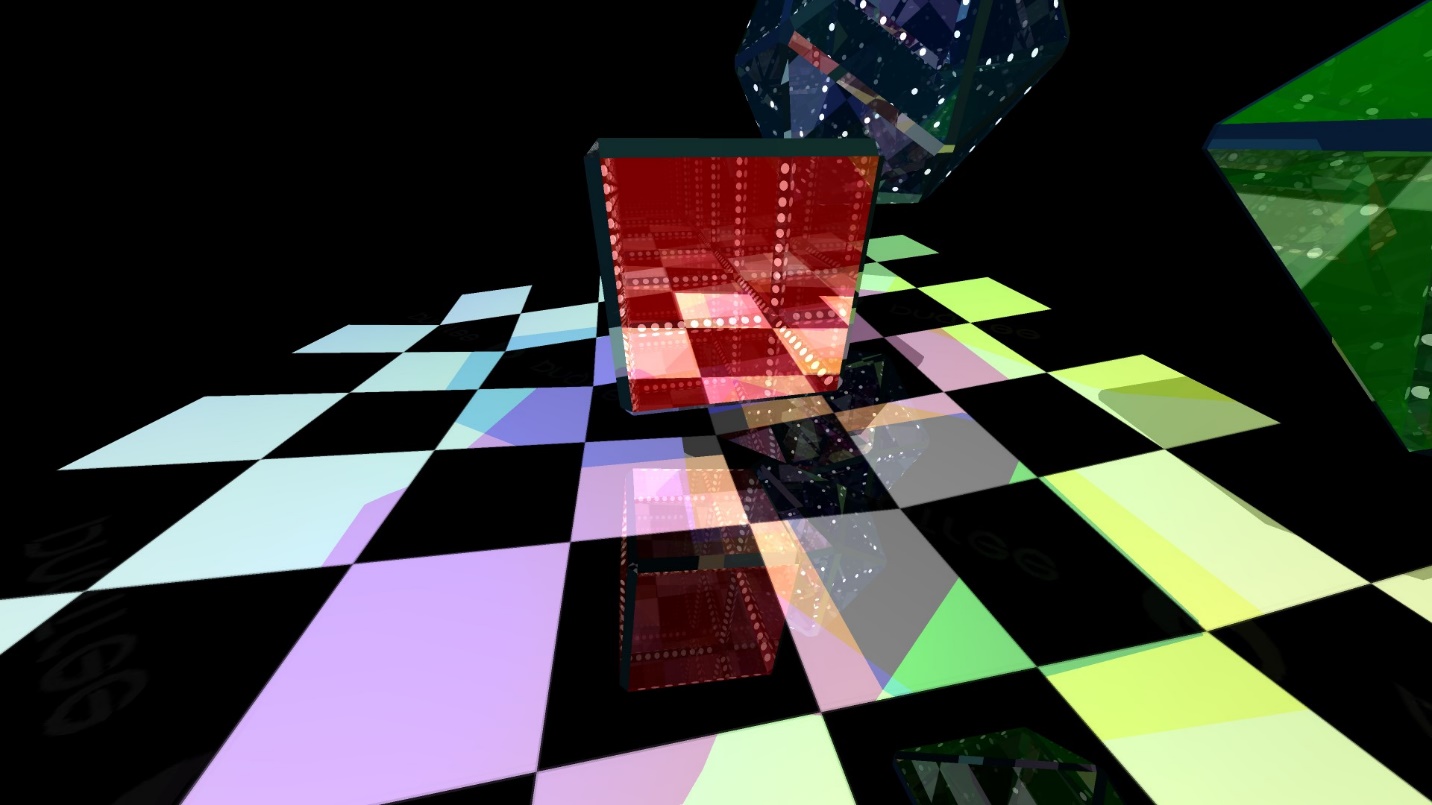


Скриншоты работы программы:

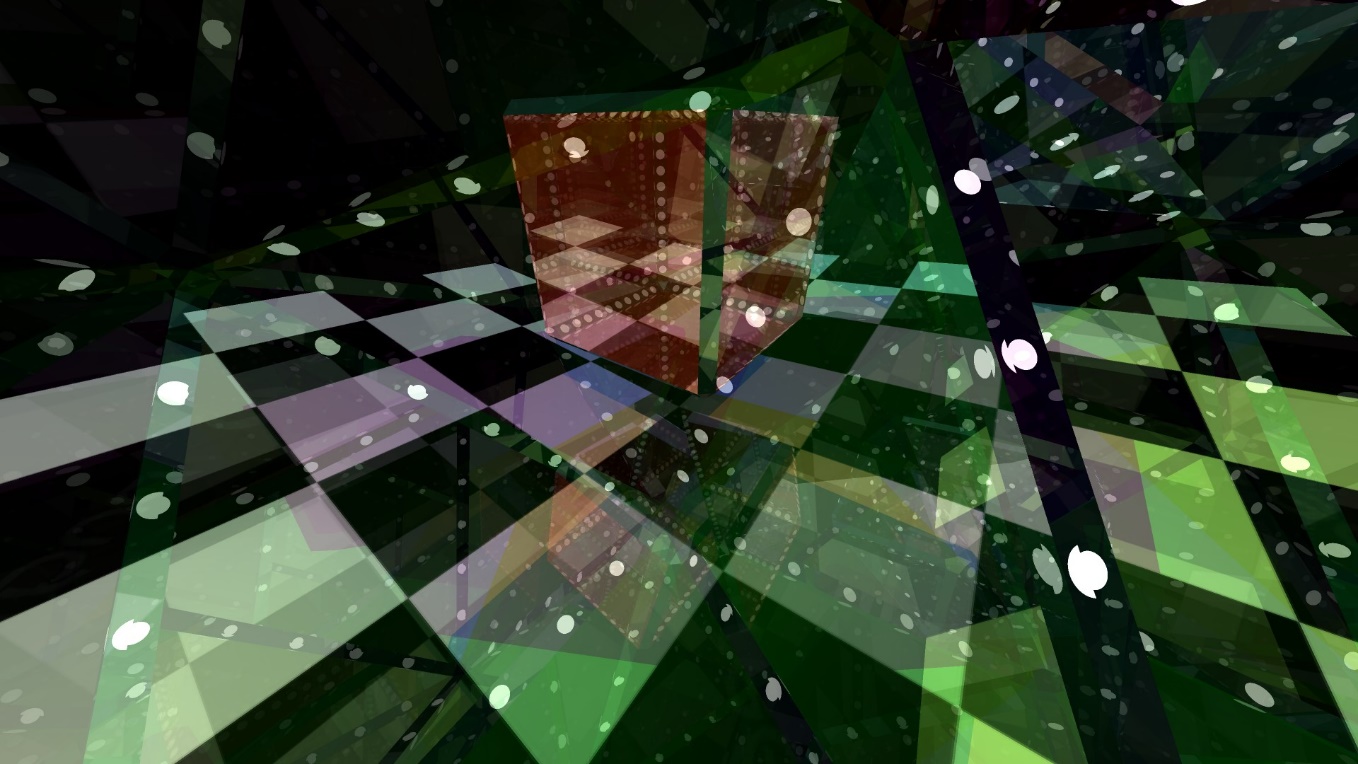
“Эффект бесконечности” внутри куба:



На полу видно отражение всех трёх фигур:



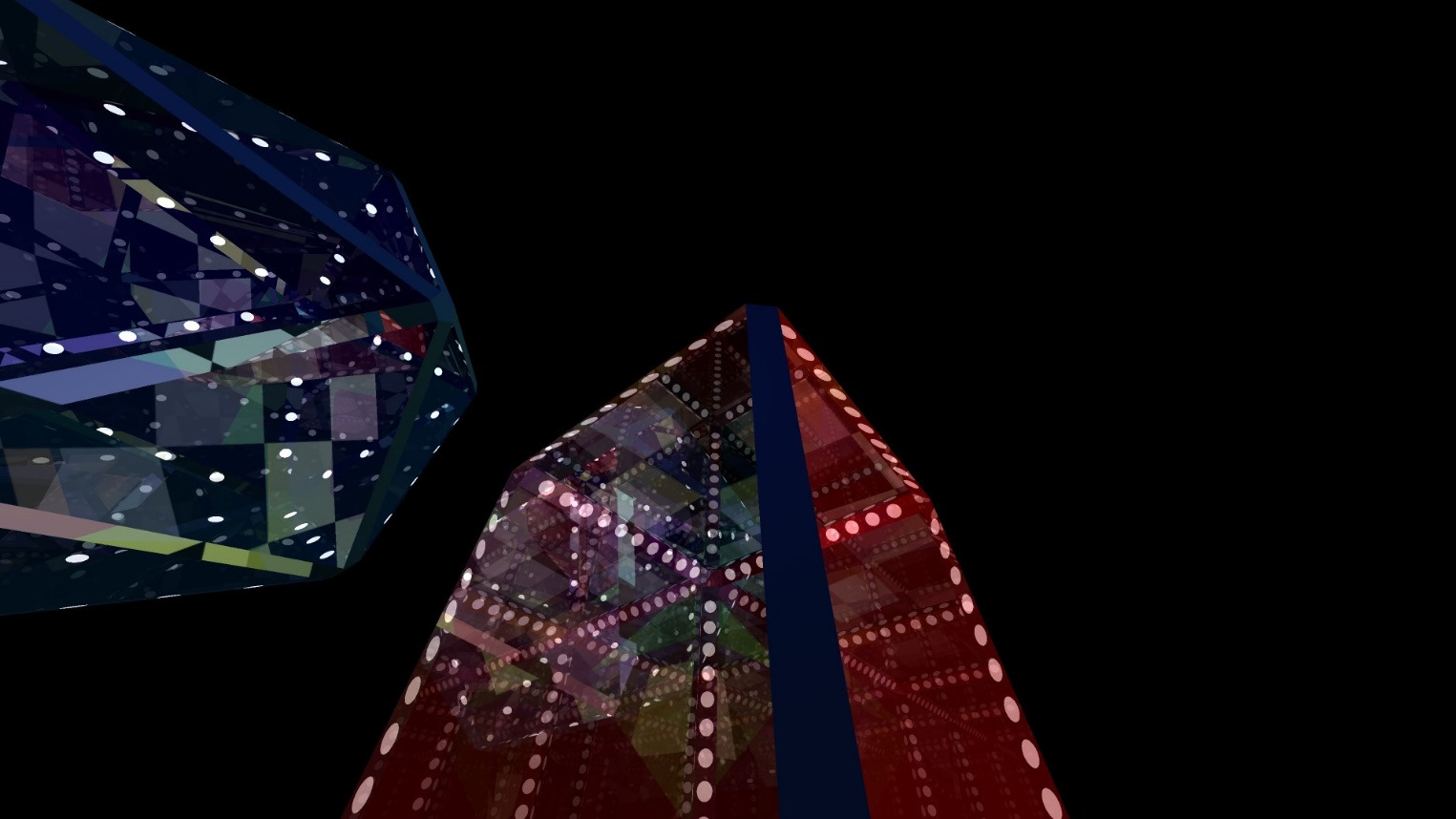
Камера попала внутрь октаэдра:



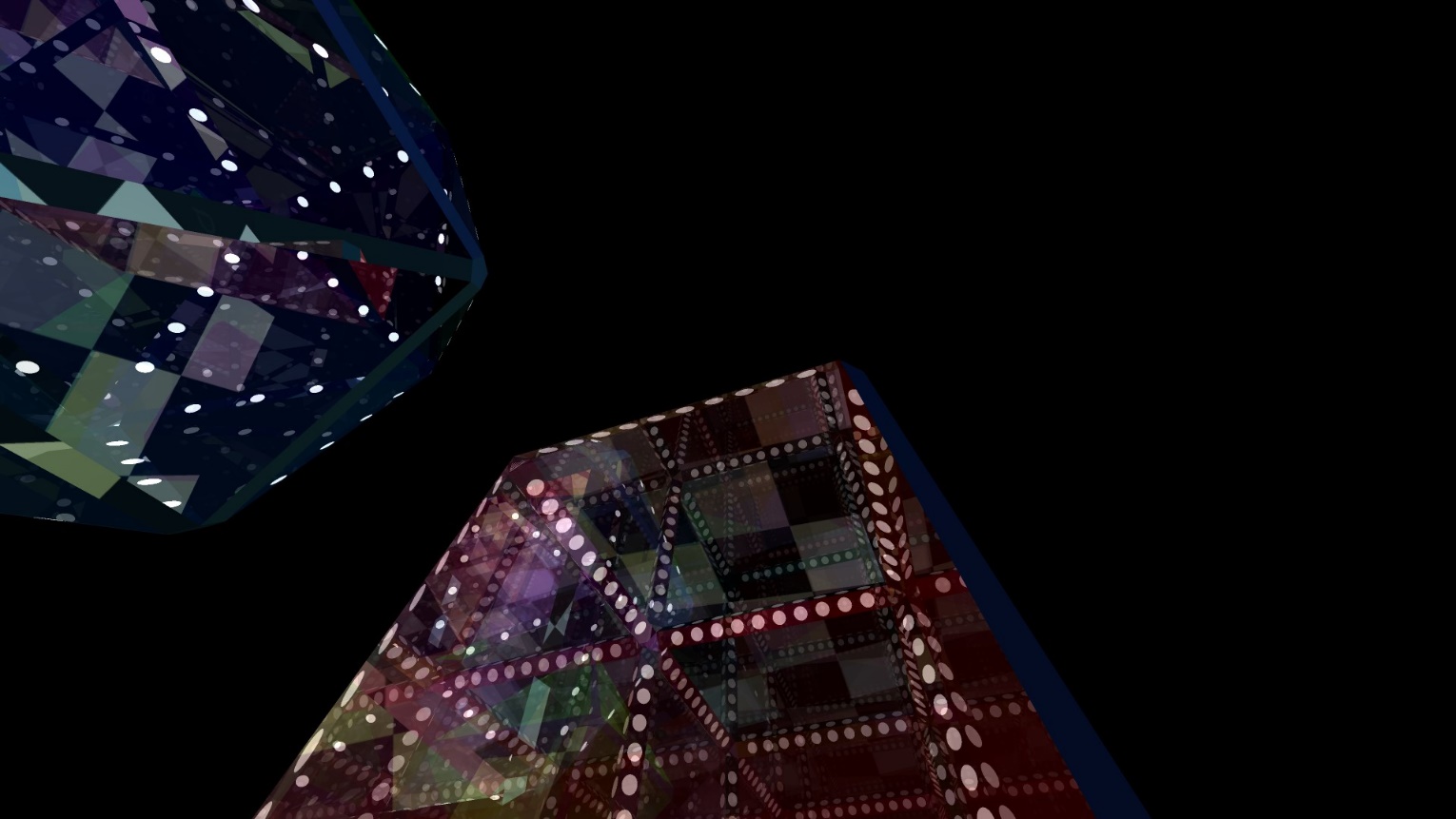
Ещё один ракурс на куб. Видно свечение нескольких источников света разных цветов (на полу):



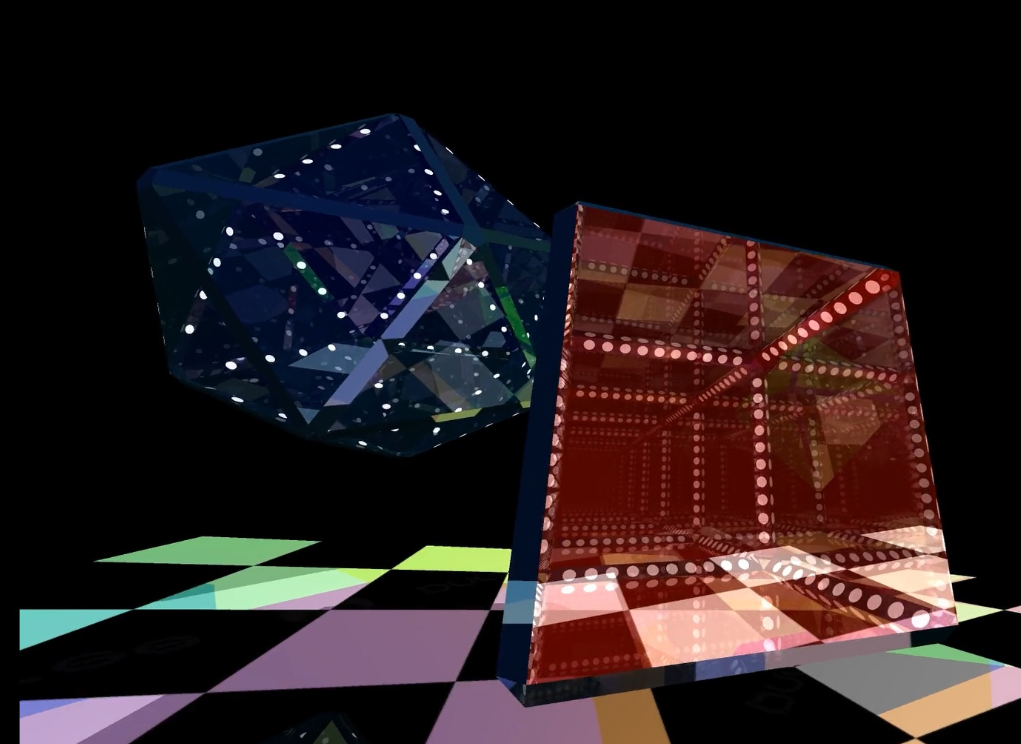
В кубе видно отражение икосаэдра:



В верхней грани куба и в икосаэдре видно отражение пола:



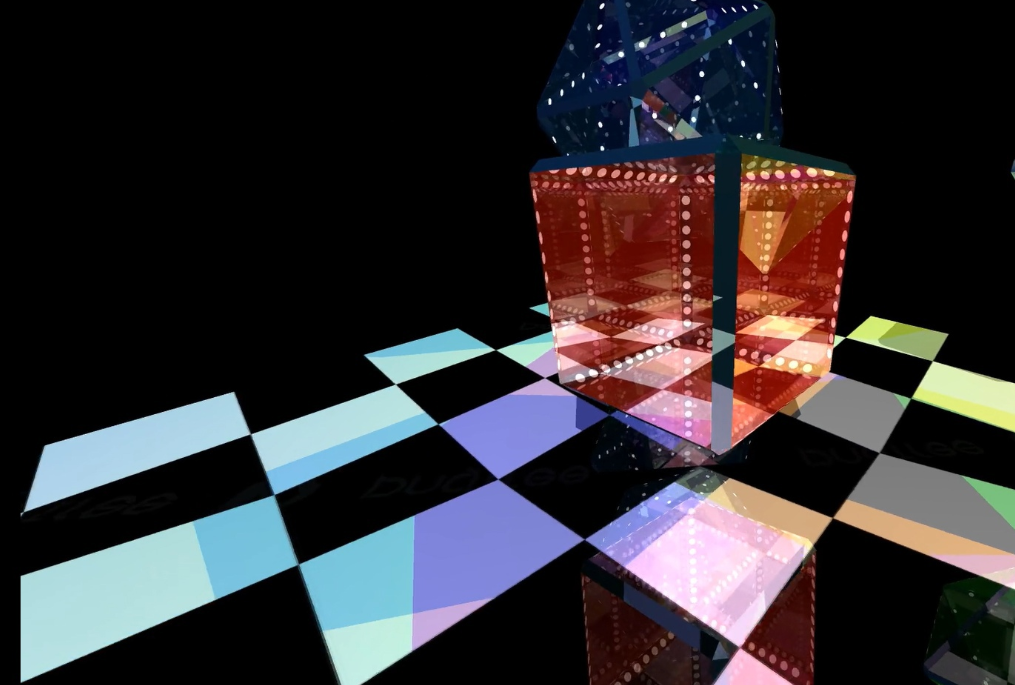
Через куб можно увидеть октаэдр:



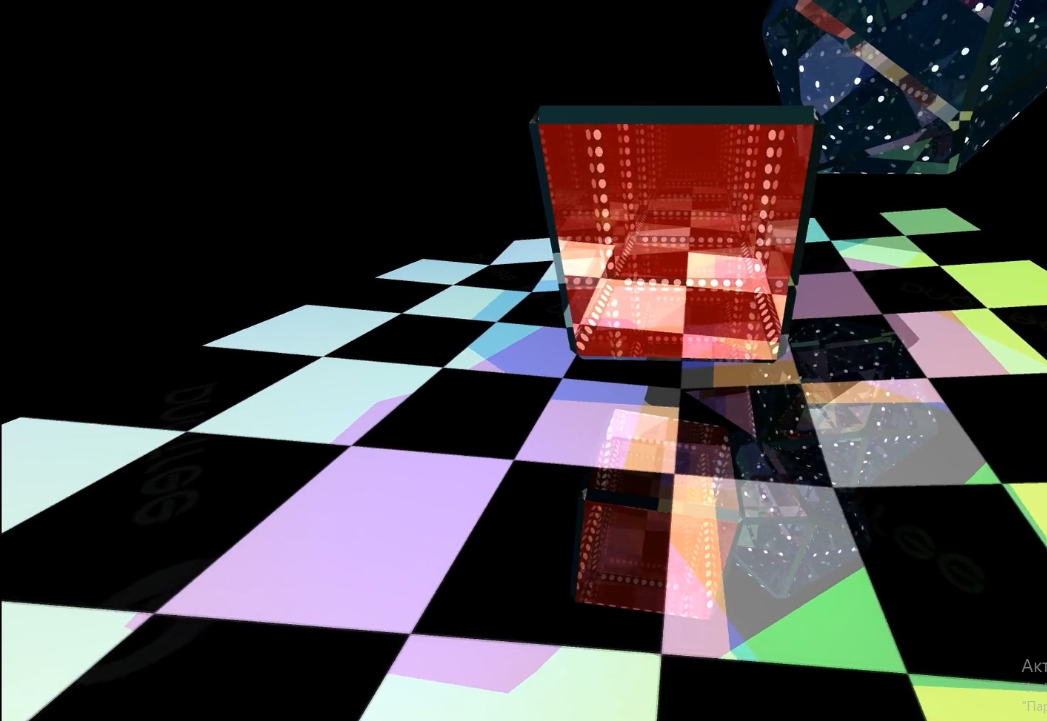
Отражение октаэдра в кубе:



От грани куба свет также отражается:



Из-за неровности пола можно увидеть, как отражение куба также скошено:



**Выводы**

Данная работа показалась мне невероятно интересной во всех планах. В процессе её выполнения я попрактиковался в геометрии, линейной алгебре, узнал, на какие составляющие делится свет на изображениях и как он реализуется в коде, как происходит отражение и преломление лучей и как они производятся многократно. Данное задание позволило мне понять, как вообще происходит отрисовка изображения программно, без каких-либо сторонних библиотек. Я очень доволен полученным результатом, хотя работа была довольно непростая. Впечатления, полученные мной в процессе выполнения данного задания, останутся со мной надолго.

**Литература**

1. <http://www.ray-tracing.ru/> - Трассировка лучей в реальном времени.
2. <https://cgraph.ru/node/435> - Простые модели освещения.
3. <https://fop1.narod.ru/ico.html> - Многогранники и вращения. Икосаэдр.
4. Бортаковский А.С., Пегачкова Е.А. Типовые задачи по линейной алгебре. Часть 1. (Серия «Учебно-методические комплексы кафедры математической кибернетики»): Учебное пособие. — М.: Доброе слово, 2013. — 92 с.