МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

Институт №8 «Информационные технологии и прикладная математика»
Кафедра 806 «Вычислительная математика и программирование»

Курсовой проект
по курсу «Программирование графических процессов»

Обратная трассировка лучей (Ray Tracing). Технологии MPI, CUDA и OpenMP

Выполнил: Д.В. Коростелев
Группа: 8О-408Б
Преподаватели: К.Г. Крашенинников,
А.Ю. Морозов

Москва, 2022

Условие

Цель работы:

Совместное использование технологии MPI, технологии CUDA и технологии OpenMP для создание фотореалистической визуализации. Создание анимации.

Задание:

Совпадает с заданием для КР по курсу ПОД.
Программа должна поддерживать следующие ключи запуска:
--cpu Для расчетов используется центральный процессор (OpenMP)
--gpu Для расчетов задействуется видеокарта (CUDA)
--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.
Подразумевается, что запуск всегда осуществляется на кластере (т.е. всегда используется технология MPI). Учесть возможность наличия нескольких GPU в рамках одной машины кластера.

Программное и аппаратное обеспечение

- Графический процессор NVIDA GeForce GTX 1050

Графическая память	2 Gb
Разделяемая память на блок	48 Mb
Константная память	64 Mb
Количество регистров на блок	65536
Максимальное количество блоков на процессор	32
Максимальное количество потоков на блок	1024
Количество мультипроцессоров	5

- Процессор Intel Core i7-7700HQ 4x 2.808ГГц

Количество ядер	4
Количество потоков	8
Базовая тактовая частота	2.8 GHz
Максимальная тактовая частота	3.8 GHz
Кеш-память	6 Mb

- Оперативная память DDR4-SODIMM

Объем памяти	8 Gb
Частота	2400 MHz
Форм-фактор	SODIMM
Количество плашек	2

- SSD и HDD накопители

Объем SSD накопителя	128 Gb
Объем HDD накопителя	1 Tb

- Программное обеспечение

Операционная система	Windows 10 Pro
Средство разработки на CUDA (IDE)	Microsoft Visual Studio 2019
Компиляторы	MSVC 2019
Версия CUDA Toolkit	11.4.2
Дополнительный текстовый редактор	Notepad++

Метод решения и описание программы

Из камеры будем пускать лучи на сцену и сначала проверять пересекается ли луч с каким-либо полигоном, если пересечения нет, то закрашиваем точку в черный цвет. Если пересечение с полигоном есть, то нужно проверить – есть ли прямой луч до какого-либо источника света, если прямой луч имеется, то рассчитываем свет по модели Фонга. Если алгоритм на ЦПУ, то мы последовательно для каждого кадра, каждого пикселя и каждого луча считаем цвет пикселя. Если на ГПУ – то каждый поток каждого блока рассчитывает значения цвета пикселя параллельно друг с другом. Также, чтобы избежать эффекта ребристости – используем алгоритм SSAA (реализован как на гпу, так и на цпу). Рендеринг отдельных кадров будем осуществлять параллельно на разных процессорах при помощи MPI.

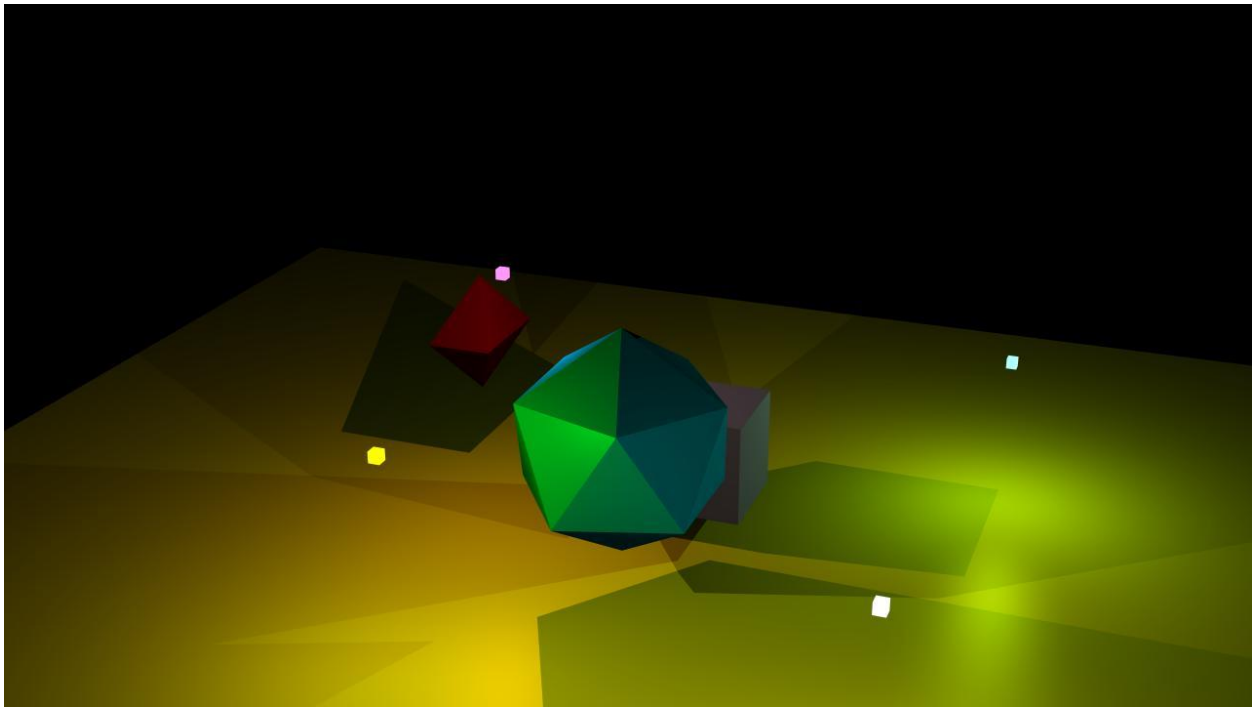
Во входных параметрах можно задавать положение камеры, характер ее движения, точку куда она смотрит и как перемещается со временем. Можно указывать положение фигур, их размер, цвет, характеристики цвета стороны при попадании света на полигон, а также можно задать положение и цвет пола на сцене.

Бенчмарки [время на гпу(время на цпу) при количестве процессоров равным 4]

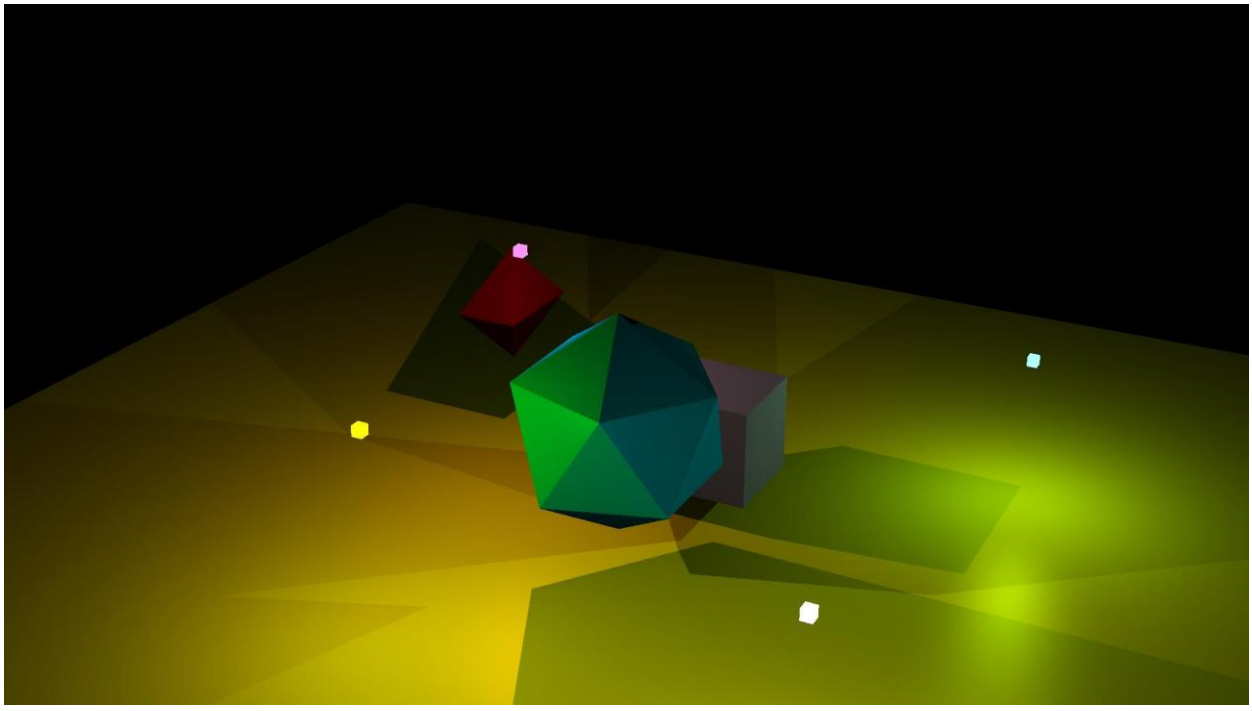
Кол-во источников света / Размерн-ть одного кадра	2	3	4
720x360	~0.1сек (~3 сек)	~0.2 сек (~4 сек)	~0.2 сек (~6 сек)
1000x500	~0.4сек (~7 сек)	~0.7 сек (~11 сек)	~0.9 сек (~15 сек)
1280x720	~1 сек (~22 сек)	~1.5 сек (~10 сек)	~2.3 сек (~14 сек)
1920x1080	~1.2 сек (~40 сек)	~1.8 сек (~2.5 мин)	~2 сек (5 мин)
1280x720 (SSAA 2)	~2 сек (-)	~4 сек (-)	~6 сек (-)
1920x1080 (SSAA 2)	~4 сек (-)	~6 сек (-)	~10 сек (-)
1280x720 (SSAA 4)	~15 сек (-)	~24 сек (-)	~40 сек (-)

Результаты

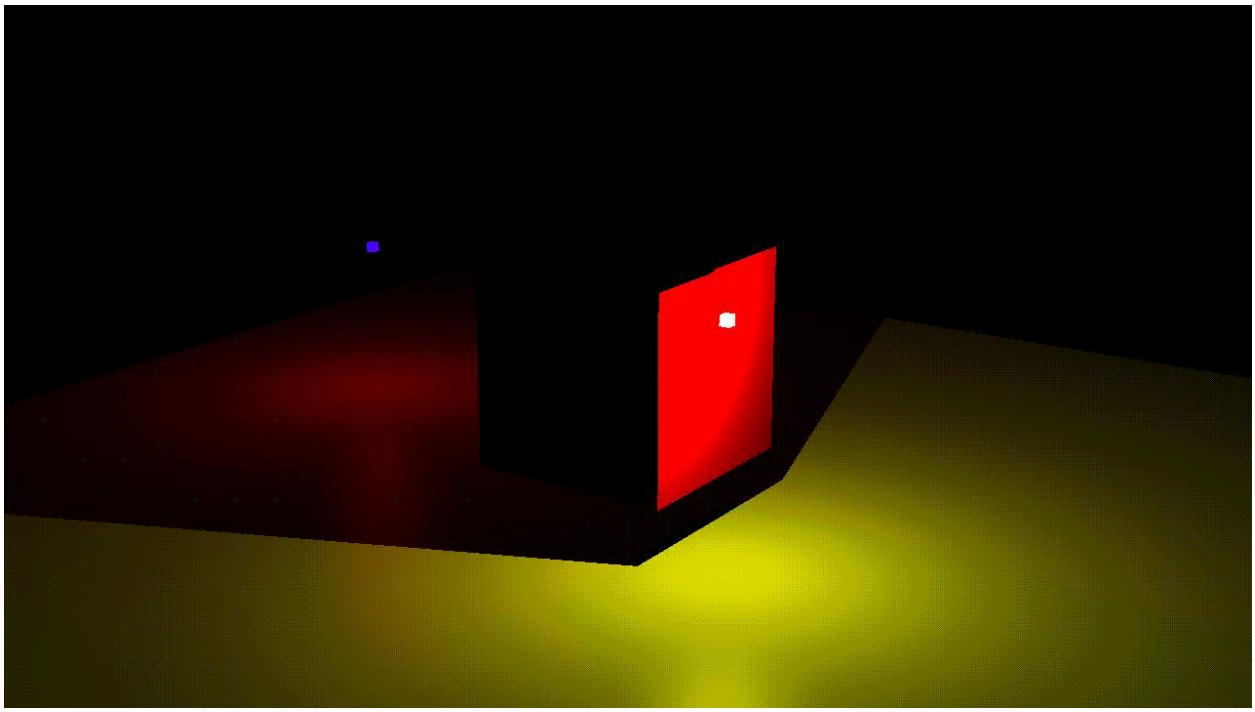
SSAAx2



SSAAx4



SSAA x1



Выводы

Курсовой проект расширил курсовую работу по курсу параллельная обработка данных и позволил еще сильнее увеличить скорость рендеринга сцены за счет применения технологий MPI, при неплохо спроектированной архитектуре оказалось совсем несложным встроить MPI в предыдущий проект и достичь ускорения рендеринга.