ИТМО

Факультет программной инженерии и компьютерной техники

Методы расчёта глобальной освещённости

Лабораторная работа №2: Моделирование равномерного распределения лучей внутри плоских фигур (треугольник, круг)

Вариант №6

Преподаватель: Потемин Игорь Станиславович

Выполнил: студент: Кульбако Артемий Юрьевич, P34115

Санкт-Петербург

2023

Задание

Номер варианта	Вершина 1	Вершина 2	Вершина 3	Dir	Радиус кругу
6	10,0,10	0,0,0	10,0,0	0,-1,0	9

Исходные данные: Координаты вершин плоского треугольника. Радиус круга.
Цель работы: Овладеть навыками расчета равномерного распределения лучей внутри плоского треугольника и круга, а также навыками визуализации полученного распределения лучей с использованием комплекса программ Lumicept.

Задачи:

- Используя лекционный материал по методике расчета равномерного распределения случайной величины, **написать программы** (C/C++, Python) для расчета равномерного распределения лучей внутри плоского треугольника и круга, **сформировать массивы данных** требуемых распределений для различного количества лучей (1000, 10000, 100000, 1000000).

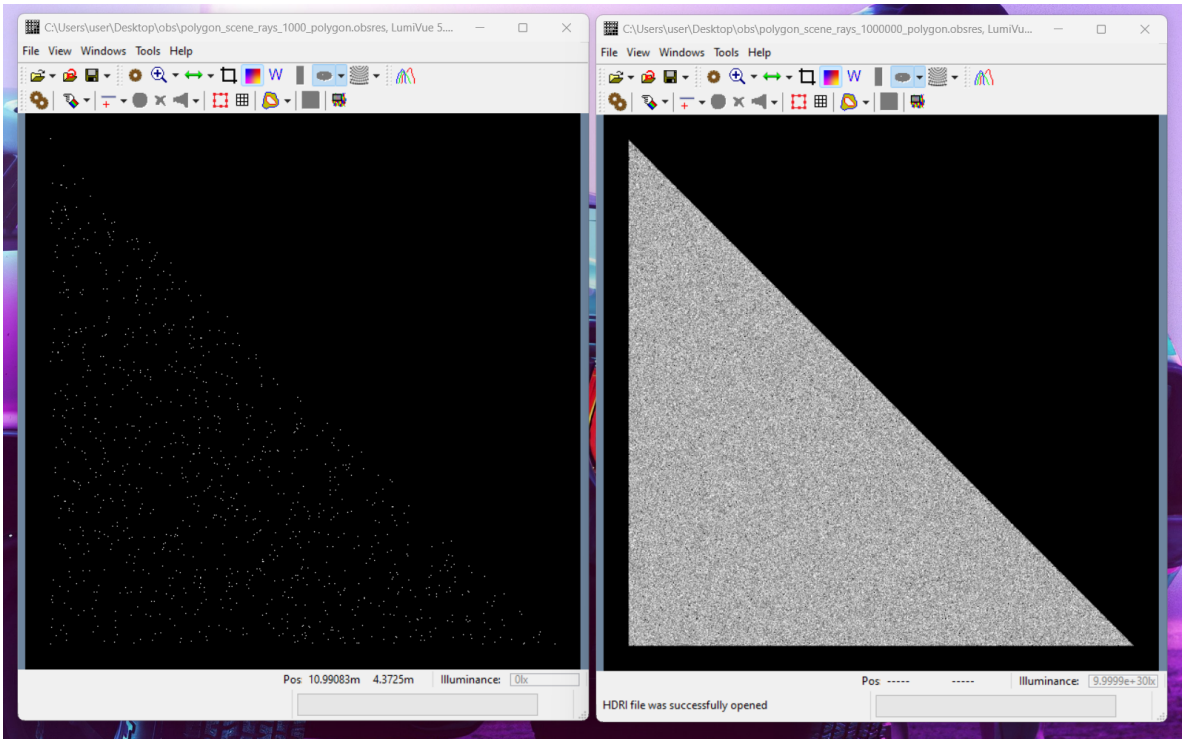
- **Визуализировать полученное распределение** с помощью комплекса программ Lumicept. Для визуализации использовать два способа: Первый способ подразумевает визуализацию распределения в виде изображения с широким динамическим диапазоном HDRI; Второй способ подразумевает формирование источника света типа RaySet, и последующие расчет и визуализацию освещенности на модели плоского приемника (Plane Observer). Размер плоского приемника сделать таким, чтобы треугольник и круг были вписаны в прямоугольник приемника.

- **Оценить равномерность полученного распределения** с помощью инструмента “Detector properties” проверяя среднее значение в трех различных зонах изображения приемника.

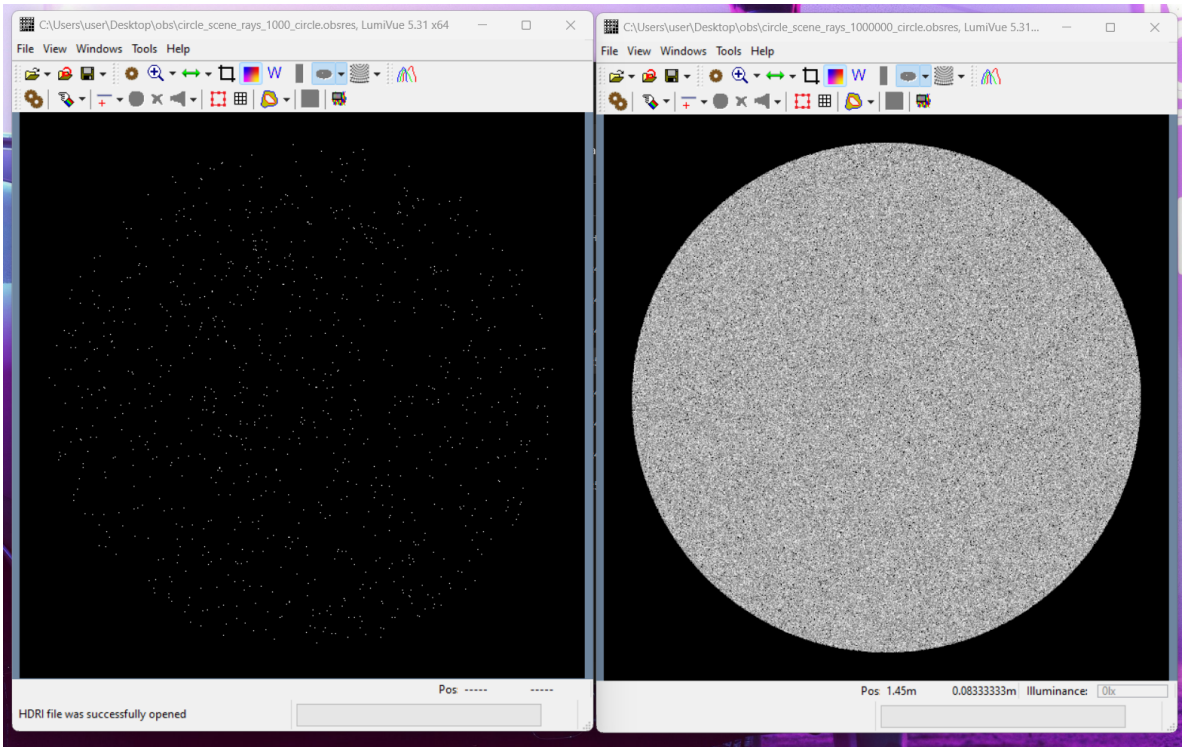
Отчет представить в электронном виде: Формат MS Word или PowerPoint. Можно использовать скриншоты из Lumicept. Оценку равномерности для трех различных зон представить в виде таблицы. К отчету приложить тексты разработанных программ, исполняемые модули, HDRI (LUX) файлы, файлы сцен (*.iof) и RAY-файлы.

Выполнение

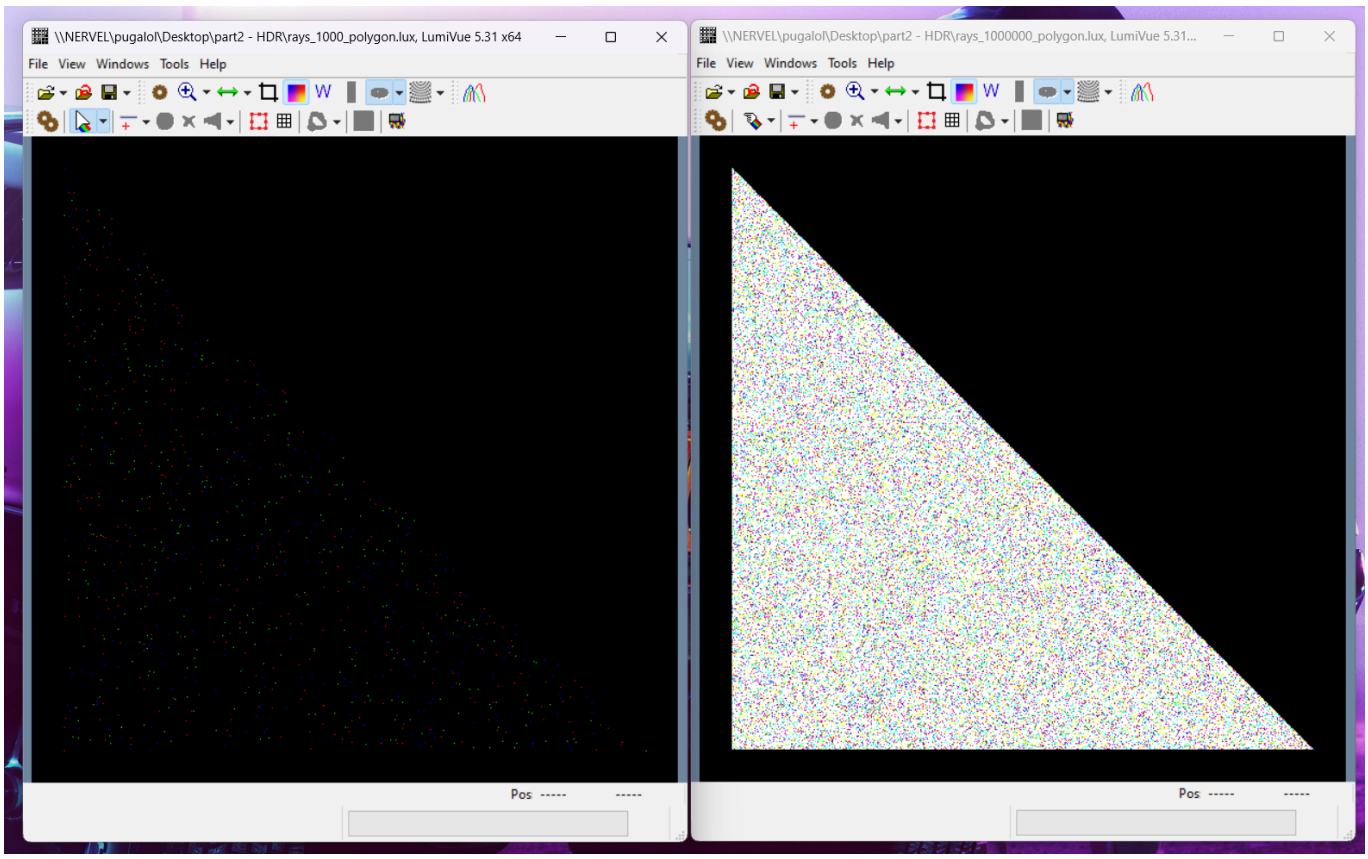
Далее приведены примеры сравнения средней освещённости для 1k и 1000k лучей. Исходные картинки для всех комбинаций освещённости/фигуры приложены к работе.



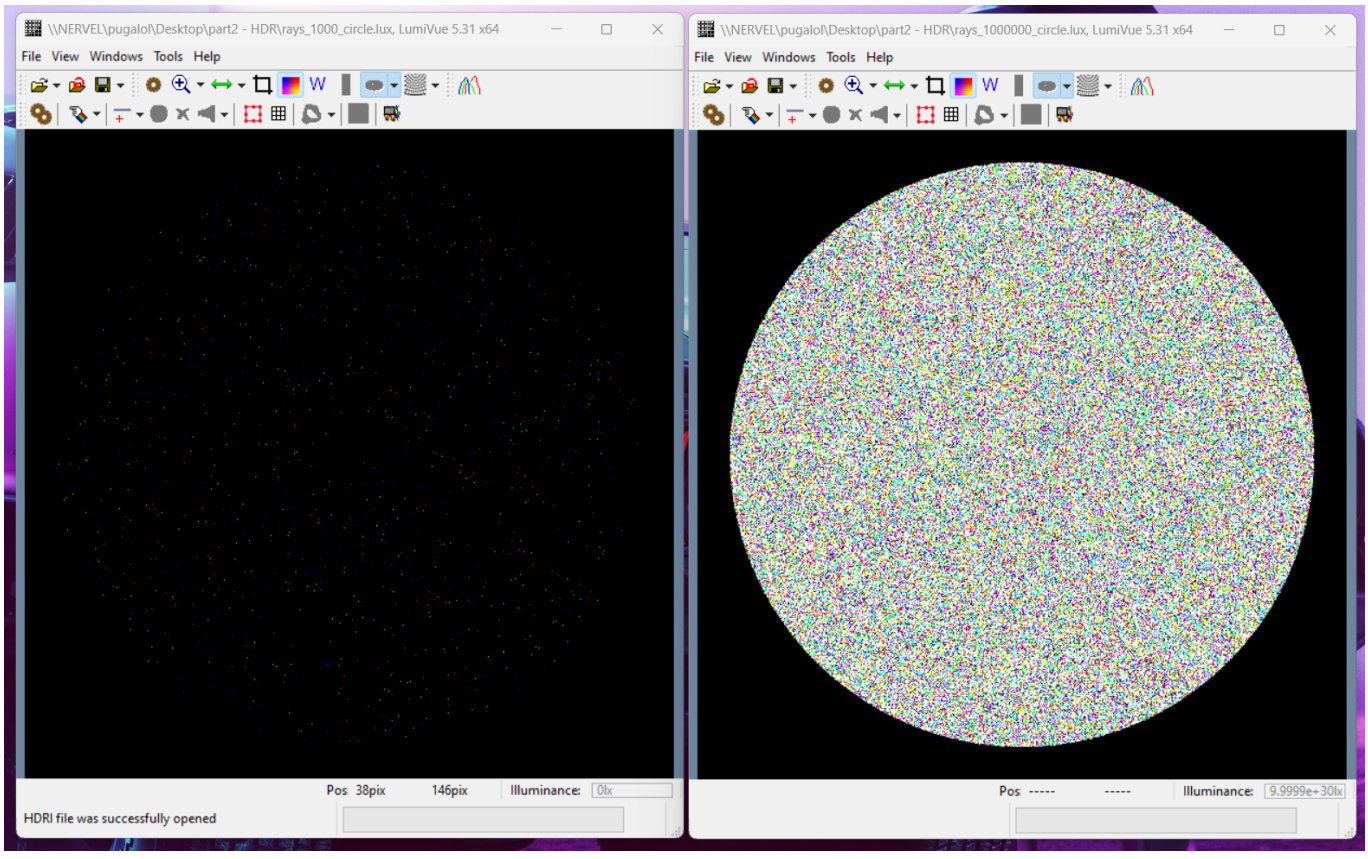
polygon [rayset]



circle [rayset]



polygon [HDR]



circle [HDR]

Разница между алгоритмом отзеркаливания и отбрасывания составила ~25%. В теории она должна быть в 2 раза, вероятное какие-то "подкапотные" оптимизации улучшают время работы отбрасывания, либо функция генерации рандомного числа в Python странная. В целом, с этими алгоритмами странная ситуация, не исключено, что влияет процессорный кеш.

```
comparing methods time:
calc_polygon_distribution = 153.96594905853271 sec
next...
calc_polygon_distribution = 191.89195108413696 sec
OK
```

Результат сравнения для круга, разница приблизительно та же (~20%).

```
comparing methods time:
calc_circle_distribution = 33.15459704399109 sec
next...
calc_circle_distribution = 40.9703710079193 sec
OK
```

Вывод

Как можно видеть по полученным HDR-изображениям, я научился пускать лучи внутрь полигона и круга, а также расчитывать равномерное распределение внутри этих фигур. Самой сложной частью задания было не запутаться в огромном количество сгенерированной скриптом информации, ввиду больше количество комбинаций данных.