**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ ЭКОНОМИЧЕСКИЙ**

**УНИВЕРСИТЕТ»**

**(СПбГЭУ)**

Факультет информатики и прикладной математики

Кафедра прикладной математики и экономико-математических методов

**КУРСОВАЯ РАБОТА**

по дисциплине:

«**Языки и методы программирования**»

Тема: «Реализация алгоритма трассировки лучей

на языке программирования Python»

Направление 01.03.02 «Прикладная математика и информатика»

Направленность «Прикладная математика и информатика в экономике и управлении»

Обучающийся Максимов Владимир Павлович

Группа ПМ-2201 Подпись\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Проверил Карахтанова А.Б.

Должность доцент

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Дата: «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург

2023 г.

**СОДЕРЖАНИЕ**

|  |  |
| --- | --- |
| ВВЕДЕНИЕ | 3 |
|  |  |
| 1. ОБЩЕЕ ОПИСАНИЕ ПРОГРАММЫ | 4 |
| 1.1. Реализация холста и окна просмотра | 5 |
| 1.2. Реализация сцены и объектов | 6 |
| 1.3. Создание результирующего списка | 7 |
|  |  |
| 2. РЕАЛИЗАЦИЯ ФУНКЦИИ ТРАССИРОВКИ ЛУЧЕЙ | 8 |
| 2.1. Реализация луча | 9 |
| 2.2. Поиск ближайшего пересечения с объектом | 9 |
| 2.3. Отражение луча | 10 |
|  |  |
| 3. РЕАЛИЗАЦИЯ ОСВЕЩЕННОСТИ ТОЧКИ | 10 |
| 3.1. Реализация света | 12 |
| 3.1.1. Общий свет | 12 |
| 3.1.2. Точечный свет | 13 |
| 3.1.3. Направленный свет | 13 |
| 3.2. Наличие тени | 13 |
| 3.3. Диффузный свет | 14 |
| 3.4. Акцентный свет | 14 |
|  |  |
| 4. ПЕРЕСЕЧЕНИЕ ЛУЧА СО СФЕРОЙ | 15 |
|  |  |
| ЗАКЛЮЧЕНИЕ | 17 |
|  |  |
| СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ | 18 |
|  |  |

**ВВЕДЕНИЕ**

Целью курсовой работы является реализация алгоритма трассировки лучей

на языке программирования Python [1].

Задачи курсовой работы:

1. создание холста и окна просмотра;
2. реализация сцены и объектов;
3. реализация луча;
4. реализация поиска ближайшего объекта, который пересекает луч;
5. создание функции отражения луча;
6. реализация освещенности точки.
7. **ОБЩЕЕ ОПИСАНИЕ ПРОГРАММЫ**

Трассировка лучей – один из методов геометрического оптики, позволяющий моделировать распространение световых лучей на компьютере. Главная задача – написать алгоритм, который позволяет по заданной сцене с объектами и по положению камеры создать проекцию данного пространства на экран монитора, которая будет отражать распространение световых лучей, приближенное к реальности.

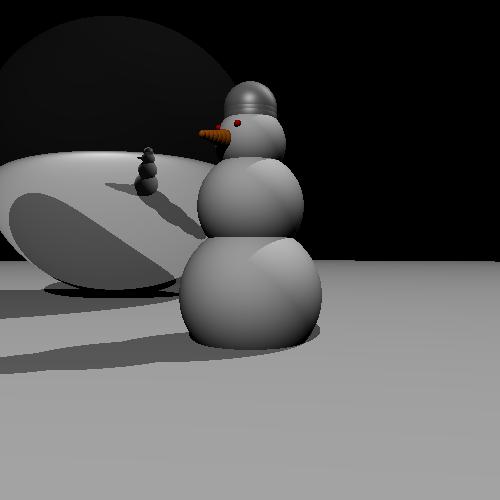


Рисунок 1 – Пример работы алгоритма трассировки лучей

* 1. **Реализация холста и окна просмотра**

Холст – прямоугольный массив пикселей. Каждый пиксель представляет собой список из трех чисел от 0 до 255, что соответствует цветовой модели «RGB». Ширину холста обозначим через переменную , а высоту через . Для работы с холстом будем использовать систему координат, в которой точка начала координат находится в верхнем левом углу, увеличивается вправо, а – вниз.

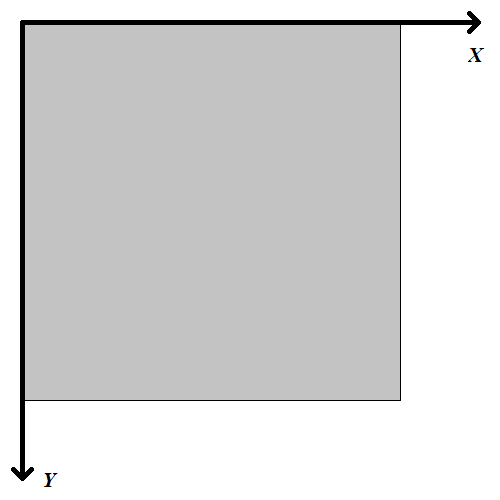


Рисунок 2 – Система координат холста

Окно просмотра – холст в классической системе координат, в которой точка начала координат находится в центре, увеличивается вправо, а – вверх. При использовании данной системы координат, будет изменяться в интервале , а в интервале . Мы будем работать с окном просмотра, но так как отображать на экране мы будем холст, необходимо преобразовать окно просмотра в холст. То есть преобразовать каждую координату с помощью формул для и для . Окно просмотра располагается на расстоянии 1 от камеры.

* 1. **Реализация сцены и объектов**

Сцена – набор объектов, которые требуется отобразить на окне просмотра и источников освещения, которые нужны для определения цвета пикселей.

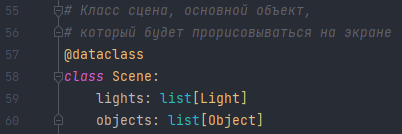


Рисунок 3 – Программный код класса «Scene»

Каждый объект – экземпляр класса «Object», который содержит тип объекта (натуральное число) и сам объект. В нашем случае все объекты имеют тип объекта «1», что соответствует экземпляру класса «Sphere». Сделано это для упрощения реализации программы.

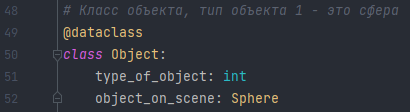


Рисунок 4 – Программный код класса «Object»

Экземпляры класса «Sphere» имеют 5 параметров: центр сферы (массив библиотеки NumPy [2], состоящий из 3 элементов), радиус (вещественное число), цвет (массив библиотеки NumPy, состоящий из 3 элементов), блеск (целое число от до ), коэффициент отражения (дробное число от 0 до 1).

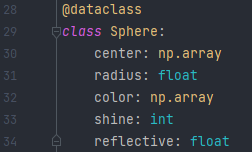


Рисунок 5 – Программный код класса «Sphere»

* 1. **Создание результирующего списка**

Результирующий список – массив библиотеки NumPy, состоящий из списков троек чисел, которые образуют цвета. Таким образом результирующий список – матрица размером на, каждый элемент которой – цвет в формате «RGB».



Рисунок 6 – Программный код результирующего списка

Для заполнения массива используем функцию «Paint», которая имеет 2 аргумента – camera(положение камеры – список координат) и camera*\_*rotation(матрица поворота размером 3 на 3). Внутри функции перебираем все возможные значения координаты и . Для каждой пары координат высчитываем направление луча и далее вычисляем значения цвета c помощью функции «trace\_ray», после чего в результирующий список «arr» записываем данный цвет.

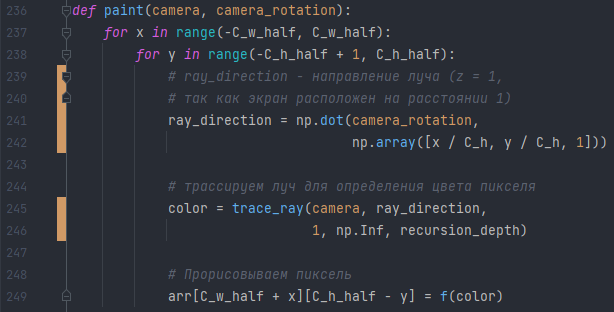


Рисунок 7 – Программный код функции «paint»

1. **РЕАЛИЗАЦИЯ ФУНКЦИИ ТРАССИРОВКИ ЛУЧЕЙ**

Функция «trace\_ray» принимает 4 аргумента: camera\_position (позиция камеры), direction (направление луча), t\_min (минимальное расстояние до объекта), t\_max (максимальное расстояние до объекта), rec\_depth (глубина рекурсии)*.* Первый и второй аргумент используются для вычисления ближайшего объекта, с которым пересекается луч, начало которого лежит в точке позиции камеры и который направлен в сторону вектора direction. Третий и четвертый аргументы – ограничение на расстояние до ближайшего пересечения, а пятый аргумент – глубина рекурсии для вызова функции трассировки луча.

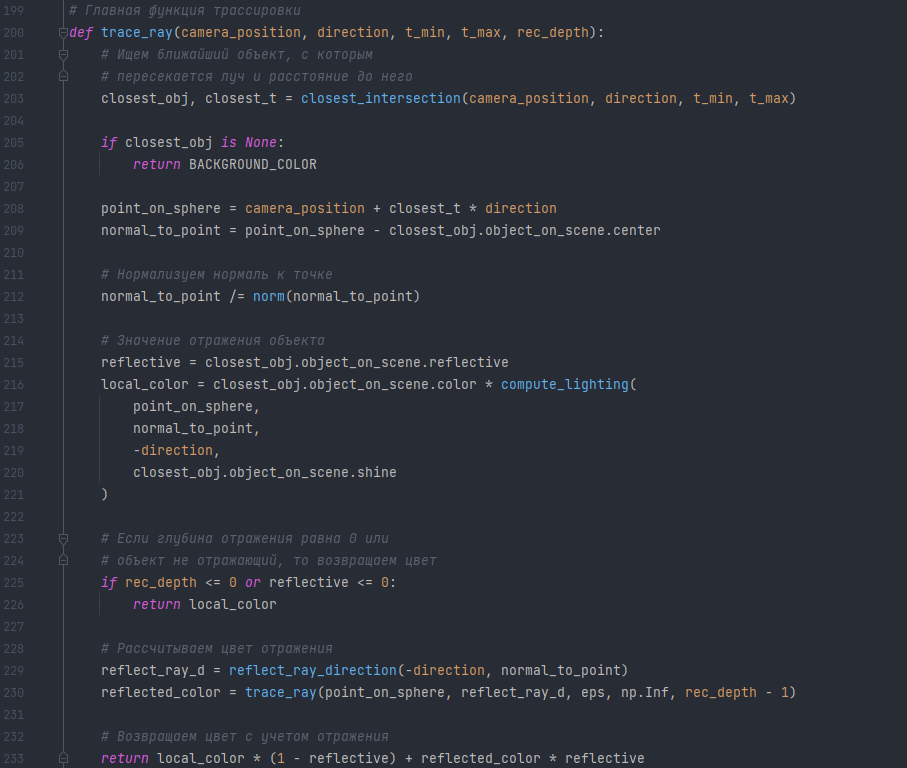


Рисунок 8 – Программный код функции «trace\_ray»

* 1. **Реализация луча**

Каждый луч задается через начало и направление. Начало и направление есть координаты в трехмерном пространстве.

* 1. **Поиск ближайшего пересечения с объектом**

Функция «closest\_intersection» выполняет поиск ближайшего пересечения луча с объектом и возвращает кортеж из 2 элементов: ближайшего объекта и расстояния до него. Внутри функции перебираются все объекты на сцене и вычисляются расстояния до каждого объекта, если луч их пересекает. Далее в переменные closest\_t и closest\_obj записываются новые данные, если расстояние оказалось меньше, чем до предыдущего объекта.

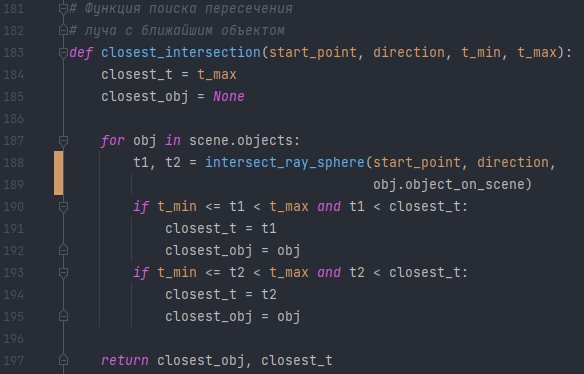


Рисунок 9 – Программный код функции «closest\_intersection»

* 1. **Отражение луча**

После того, как был найден ближайший объект, который пересекает луч, вычисляем значение цвета, как произведение цвета объекта (то есть каждого числа из тройки чисел) и освещенности точки. Далее если объект не отражающий или мы достигли предела вычисления отражений, то возвращаем цвет, в противном случае вычисляем цвет точки, в которую попадает отраженный луч и с его учетом возвращаем цвет. При значении параметра reflective равного 1, объект будет представлять собой аппроксимацию зеркала с глубиной отражения равной rec\_depth - 1.

1. **РЕАЛИЗАЦИЯ ОСВЕЩЕННОСТИ ТОЧКИ**

Освещенность точки вычисляет функция «compute\_lightning», которая принимает 4 аргумента: точка на объекте, для которой вычисляется освещенность, вектор нормали к данной точке (разность координат точки на сфере и центра данной сферы), вектор направления от точки к камере и значение блеска сферы. Внутри функции перебираем все источники света в сцене и для каждого вычисляем значение освещенности, после чего суммируем их и получаем общее значение освещенности точки.

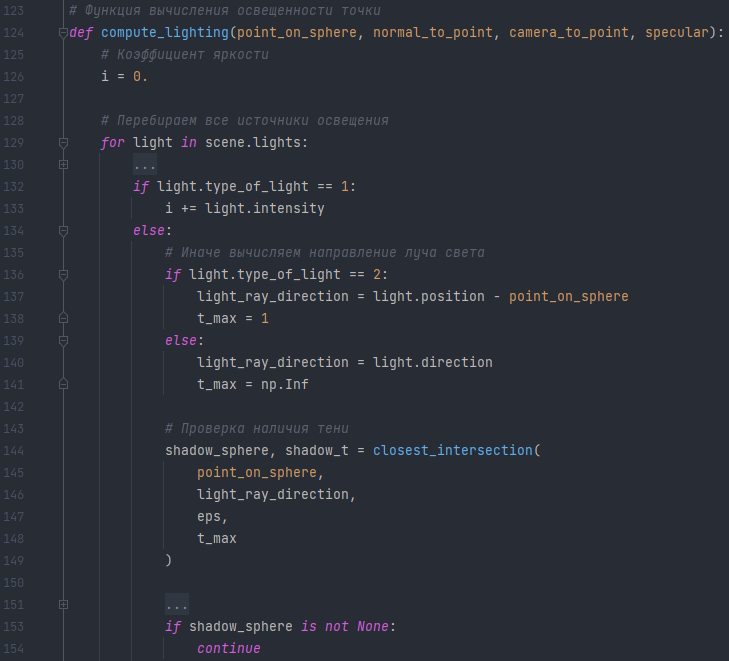


Рисунок 10 – Программный код функции «compute\_lighting» часть 1



Рисунок 11 – Программный код функции «compute\_lighting» часть 2

* 1. **Реализация света**

Источник света – экземпляр класса «Light», который имеет 4 параметра: type\_of\_light (тип света – натуральное число от 1 до 3), intensity (интенсивность света – дробное число от 0 до 1), position (позиция источника освещения – массив библиотеки NumPy, состоящий из 3 координат для точечного света и значение None для остальных типов), direction (направление света – массив библиотеки NumPy, состоящий из 3 координат для направленного света и значение None для остальных типов).

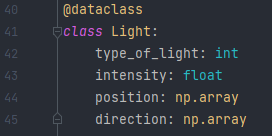


Рисунок 12 – Программный код класса «Light»

* + 1. **Общий свет**

Общий свет – тип света, который имеет только свой тип и интенсивность в качестве переменных экземпляра класса, остальные переменные имеют значение None. Общий свет будет прибавляться к результату всегда, даже если объект находится в тени. Если источник освещения – общий свет, то к коэффициенту яркости прибавляем значение атрибута intensity и переходим к следующему источнику освещения.

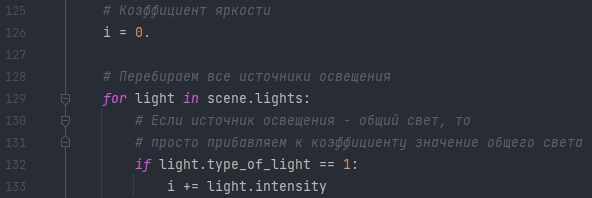


Рисунок 13 – Программный код учета общего света

* + 1. **Точечный свет**

Точечный свет – тип света, который имеет те же переменные экземпляра класса, что и общий, а также позицию. В случае, если источник освещения – точечный свет, то вычисляем направление светового луча, как разницу между позицией источника света и точкой на сфере. Также для дальнейшей проверки на наличие тени, задаем максимальное расстояние, на котором может встретится объект, создающий тень – 1, так как если мы найдем объект, расстояние до которого больше 1, то он будет находится за источником освещения, соответственно не должен создавать тень.

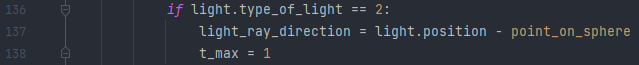


Рисунок 14 – Программный код вычисления направления луча света для точечного источника освещения

* + 1. **Направленный свет**

Направленный свет – тип света, который имеет те же переменные экземпляра класса, что и общий, а также направление, которое задается как вектор. Если источник – направленный свет, то направление света уже есть в переменной direction экземпляра класса. Переменная t в данном случае принимает значение .

* 1. **Наличие тени**

Для определения наличия тени, вызываем функцию «closest\_intersection», в которую передаем точку на сфере, направление луча, а также минимальное и максимальное расстояние до объекта. Далее проверяем, если объект, который создает тень, существует, то мы не должны проводить дальнейших манипуляций со светом, значит можем перейти к следующему источнику освещения.

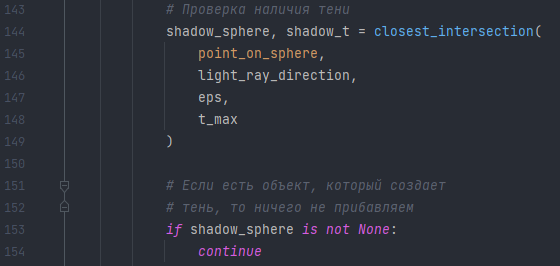


Рисунок 15 – Программный код проверки наличия тени

* 1. **Диффузный свет**

Если угол падения луча света меньше 90 градусов, то данный свет принимает участие в освещении точки, а значит мы должны его учитывать. Так как значение интенсивности источника света зависит от угла падения луча на объект, то для упрощения данного эффекта из реального мира будем умножать значение интенсивности источника освещения на косинус угла падения луча. Так как при величине угла 0 градусов косинус принимает значение 1, а при 90 градусов – значение 0, то такой вариант является отличной аппроксимацией работы света в реальном мире.

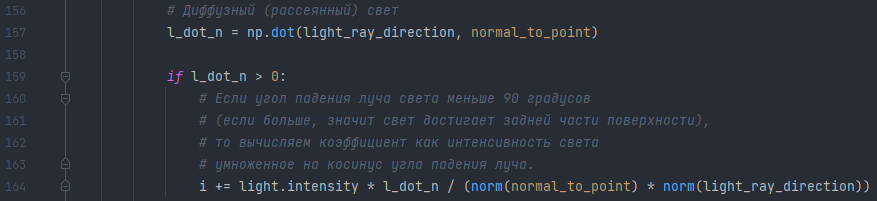


Рисунок 16 – Программный код вычисления диффузного света

* 1. **Акцентный свет**

Если значение блеска не равно -1, то вычисляем направление отраженного луча света и находим косинус угла между лучом отражения и лучом из камеры в точку. Далее если косинус угла больше 0, прибавляем к общей интенсивности света текущую, умноженную на косинус угла в степени, равной значению блеска данного объекта. Таким образом, чем больше блеск, тем более яркой и четкой будет точка, свет в которую падает под прямым углом.

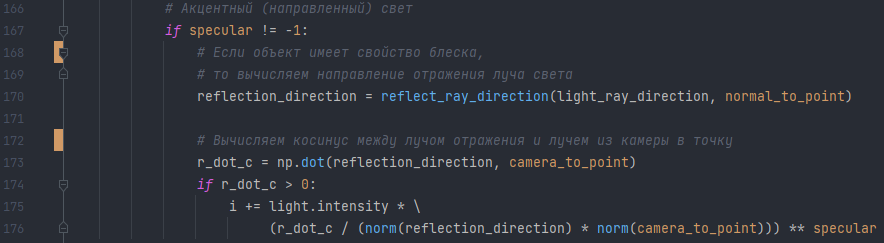


Рисунок 17 – Программный код вычисления акцентного света

1. **ПЕРЕСЕЧЕНИЕ ЛУЧА СО СФЕРОЙ**

Точки пересечения луча со сферой мы находим из уравнения для окружности и параметрического уравнения для луча: пусть – искомая точка на сфере, – центр сферы, – радиус сферы, – положение камеры, – направление луча, тогда точка должна удовлетворять уравнениям . Но её же мы можем получить из уравнения , осталось только подставить вместо точки в первое уравнение выражение, стоящее после знака равно, тогда после некоторых преобразований получим:

что является обычным квадратным уравнением относительно . Если дискриминант получается отрицательным, значит луч не пересекает сферу, если он равен 0, то луч касается сферы, в противном случае пересекает сферу в двух точках [3].

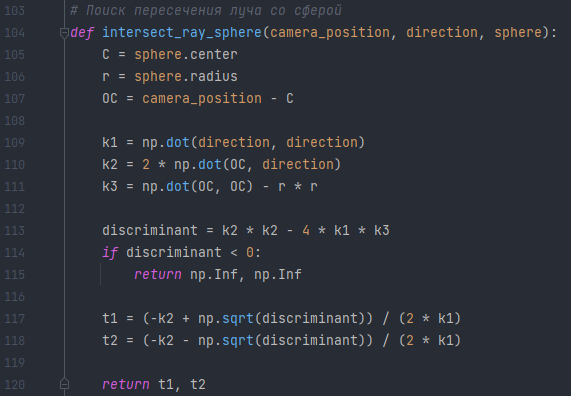


Рисунок 18 – Программный код функции «intersect\_ray\_sphere»

**ЗАКЛЮЧЕНИЕ**

Все задачи курсовой работы были выполнены. Программу можно дополнить следующим функционалом:

1. добавить другие типы объектов, такие как куб, плоскость и т.д.;
2. оптимизировать программу, используя параллелизацию, так как задача трассировки лучей относится к классу чрезвычайно параллелизуемых задач;
3. добавить объектам прозрачность и реализовать преломление лучей;
4. реализовать создание объектов, как булеву операцию между двумя другими объектами.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. PYTHON [Электронный ресурс]: URL: https://www.python.org/
2. NumPy [Электронный ресурс]: URL: https://numpy.org/
3. Birmingham Computer Graphics from Scratch: A Programmer's Introduction to 3D Rendering/ Gambetta, Gabriel; 2021.