НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет Программной Инженерии и Компьютерной Техники

Системы компьютерной обработки изображений

Лабораторная работа № 1

«Вычисление освещенности в точках поверхности»

Выполнил студент

Хоанг Ван Куан

Группа № Р3366

Преподаватель: Жданов Дмитрий Дмитриевич

г. Санкт-Петербург 2025

## Цель

Изучить, как вычисляется освещенность в точках точки на плоскости треугольника при освещении его точечным источником света с заданной диаграммой излучения.

## Используемые формулы

1. «Цветная» интенсивность источника под углом к оси источника света:

*, где – «цветная» интенсивность источника света в направлении его оси , θ - угол между направлением распространения света и осью источника света, – диаграмма излучения.*

1. «Цветная» освещенность точки:

, *где – «цветная» интенсивность света, α - угол между направлением света и нормалью к освещаемой поверхности, R - расстояние от источника света до рассматриваемой точки.*

1. Перевод локальных координат точки в плоскости в глобальные

*, где и смещения по ребрам треугольника.*

1. Вычисление вектора нормали плоскости треугольника через 3 точки:

1. Вектор от точки плоскости до источника света:

A diagram of a triangle with a red line and arrows

AI-generated content may be incorrect.

*,*

## Шаги алгоритма

1. Вычисляем
   1. Вычисляем координаты точки на поверхности
   2. Вычиляем вектор
   3. Вычиляем нормаль к поверхности
2. Рассчитываем освещенность для каждой компоненты RGB по формуле
3. Возвращаем результат в виде RGB-компонент освещенности

## Программа для расчета освещенности

// Функция считает глобальные координаты точки

class Illumination:

    # длина вектора

    def length(self, vector):

        res = 0

        for i in vector:

            res += i \* i

        return res \*\* 0.5

    # Нормализация вектора

    def normalize(self, vector):

        length = self.length(vector)

        res = []

        for i in vector:

            res.append(i / length)

        return res

    # минут

    def minus(self, vector1, vector2):

        res = []

        for i in range(len(vector1)):

            res.append(vector1[i] - vector2[i])

        return res

    # плюс

    def sum(self, vector1, vector2):

        res = []

        for i in range(len(vector1)):

            res.append(vector1[i] + vector2[i])

        return res

    # векторное произведение

    def cross\_product(self, vector1, vector2):

        res = [

            vector1[1] \* vector2[2] - vector1[2] \* vector2[1],

            vector1[2] \* vector2[0] - vector1[0] \* vector2[2],

            vector1[0] \* vector2[1] - vector1[1] \* vector2[0]

        ]

        return res

    # Скалярное произведение

    def dot\_product(self, vector1, vector2):

        res = 0

        for i in range(len(vector1)):

            res += vector1[i] \* vector2[i]

        return res

    # Умножение на число

    def scalar\_multiply(self, number, vector):

        res = []

        for i in vector:

            res.append(number \* i)

        return res

    # Нормаль к поверхности

    def normal(self, P0, P1, P2):

        edge1 = self.minus(P1, P0)

        edge2 = self.minus(P2, P0)

        cross = self.cross\_product(edge2, edge1)

        return self.normalize(cross)

    # Координата Pt

    def PT(self, P0, P1, P2, x, y):

        edge1 = self.normalize(self.minus(P1, P0))

        edge2 = self.normalize(self.minus(P2, P0))

        scaled\_edge1 = self.scalar\_multiply(x, edge1)

        scaled\_edge2 = self.scalar\_multiply(y, edge2)

        return self.sum(self.sum(P0, scaled\_edge1), scaled\_edge2)

    # Расчет освещенности

    def illumination(self, P0, P1, P2, PL, O, I0\_RGB, x, y):

        # E = I0\_RGB \* cos\_theta \* cos\_alpha / R^2

        # cos\_theta = s.O / normalize(s)

        # cos\_alpha = s.N / normalize(s)

        PT = self.PT(P0, P1, P2, x, y)

        s = self.minus(PT, PL)

        R2 = self.dot\_product(s, s)

        N = self.normal(P0, P1, P2)

        s\_length = self.length(s)

        cos\_alpha = self.dot\_product(s, N) / s\_length

        cos\_theta = self.dot\_product(s, O) / s\_length

        # освещенность

        E\_RGB = []

        for i in range(len(I0\_RGB)):

            value = (I0\_RGB[i] \* cos\_theta \* cos\_alpha) / R2

            E\_RGB.append(value)

        return E\_RGB

**Пример входных данных и результаты**



|  |  |  |
| --- | --- | --- |
| 7. |  |  |

Вычисленные значения освещенности для точек, заданных локальными координатами:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | 0 |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

## Вывод

В результате выполнения работы был изучен и реализован алгоритм вычисления освещенности точки на плоскости от источника света. Рассмотрены основные этапы вычислений, включая преобразование координат, определение нормали к плоскости, вычисление вектора до источника света и углов между векторами.