Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Алгоритмы компьютерной графики

Отчет по лабораторной работе №3

Выполнил:
Козлов Василий Сергеевич
P3315

Проверил:
Меженин Александр Владимирович

Санкт-Петербург, 8 ноября 2025

Содержание

1 Задание

Исходные данные: Система координат, плоскость, точечный источник света с Ламбертовской диаграммой излучения, прямоугольник, координаты центра круга и его радиус, в пределах которого следует рассчитать, а затем визуализировать распределение освещенности.

Цель работы: Овладеть навыками расчета и визуализации освещенности на плоскости.
Задачи:

- Провести расчет распределения освещенности на плоскости в пределах заданной области.

Рекомендуемые пределы значений параметров для расчета:

- Размер области изображения по высоте (H) и ширине (W) варьируются в диапазоне от 100 до 10000 миллиметров.
- Разрешение изображения по высоте (Hres) и ширине (Wres) варьируются в диапазоне от 200 до 800 пикселей. Разрешение должно обеспечивать квадратные пиксели.
- Координаты источника света (xL, yL, zL) [мм] по осям X и Y $\pm$10000, по оси Z от 100 до 10000.
- Сила излучения I0 варьируется от 0.01 до 10000 Вт/ср.
- Написать приложение на Python, формирующее изображение рассчитанного распределения освещенности для заданного разрешения с нормировкой (0-255) на максимальное значение освещенности. Обеспечить возможность изменения значений параметров в интерфейсе пользователя (в пределах рекомендуемых значений).
- Записать сформированное изображение в файл.
- Визуализировать изображение на мониторе.
- Визуализировать график сечения, проходящего через центр заданной области.

2 Теоретические сведения

Для точечного источника света с Ламбертовской диаграммой излучения освещенность E в точке (x, y) на плоскости рассчитывается по формуле:

$$E(x, y) = I_0 \cdot \frac{z_L^2}{r^4}$$

где $r = \sqrt{(x - x_L)^2 + (y - y_L)^2 + z_L^2}$ — расстояние от источника света до точки.

Эта формула учитывает закон обратных квадратов для расстояния и косинусы углов для Ламбертовского излучения и падения света на плоскость ($\cos\theta = \cos\phi = \frac{z_L}{r}$).

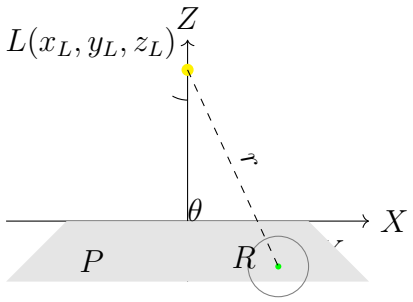

Рис. 1: Схема расположения источника света и плоскости

На схеме обозначения:

- L — точечный источник света с координатами (x_L, y_L, z_L);
- P — произвольная точка на плоскости с координатами (x, y), в которой рассчитывается освещенность;
- r — расстояние от источника света до точки P;
- R — радиус круга, в пределах которого рассчитываются и визуализируются распределение освещенности, а также статистики (максимум, минимум, среднее);
- θ — угол между нормалью к плоскости и лучом от источника света до точки P.

3 Описание алгоритма

1. **Ввод параметров**: Чтение пользовательских входных данных для размеров прямоугольной области (W, H в мм), разрешения (W_{res}, H_{res} в пикселях), позиции источника света (x_L, y_L, z_L в мм), интенсивности I_0 (Вт/ср), центра круга (x_0, y_0 в мм) и радиуса R (мм).

2. **Генерация сетки**: Создание 2D-сетки точек, центрированной в (x_0, y_0), охватывающей W мм по X и H мм по Y, с использованием `np.linspace` и `np.meshgrid`. Это обеспечивает квадратные пиксели, если $W/W_{res} = H/H_{res}$ (обеспечивается пользователем).

3. **Расчет освещенности**: Для каждой точки сетки вычисление r и затем $E = I_0 \cdot z_L^2 / r^4$.

4. **Нормализация и создание изображения**: Масштабирование E до 0-255 на основе максимального E в области для grayscale-изображения, затем сохранение с помощью `plt.imsave`.

5. **Визуализация**: - Отображение 2D-распределения с использованием `imshow` с цветовой картой 'inferno' для цветного представления фактических значений E, включая цветовую шкалу с единицами. - Отображение двух 1D-секций: горизонтальной вдоль X при $y \approx y_0$, и вертикальной вдоль Y при $x \approx x_0$ (оба через центр), используя подграфики в fig2, с графиком фактического E vs. позиции.

6. **Статистика**: Применение круговой маски, центрированной в (x_0, y_0) с радиусом R, для вычисления max/min/avg E внутри круга. Использование аналитической формулы для точных E в центре и пересечениях осей.

7. **Вывод**: Отображение вычисленных значений в GUI; сохранение нормализованного grayscale-изображения в файл.

Эта настройка обеспечивает, что визуализируемая область (прямоугольник) центрирована на круге, поэтому изменение x_0/y_0 сдвигает сетку относительно источника, динамически обновляя графики. Круг определяет область статистики внутри этой визуализируемой области.

4 Программный код

```
import tkinter as tk
from tkinter import ttk
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg

def compute_illum():
    try:
        W = float(entry_W.get())
        H = float(entry_H.get())
        Wres = int(entry_Wres.get())
        Hres = int(entry_Hres.get())
        xL = float(entry_xL.get())
        yL = float(entry_yL.get())
        zL = float(entry_zL.get())
        I0 = float(entry_I0.get())
        x0 = float(entry_x0.get())
        y0 = float(entry_y0.get())
        R = float(entry_R.get())
    except ValueError:
        label_results.config(text="Invalid input")
        return

    # Generate grid for rectangular area centered at (x0, y0)
    x = np.linspace(x0 - W/2, x0 + W/2, Wres)
    y = np.linspace(y0 - H/2, y0 + H/2, Hres)
    X, Y = np.meshgrid(x, y)

    # Compute illuminance E
    dx = X - xL
    dy = Y - yL
    dz = zL
    r2 = dx**2 + dy**2 + dz**2
    r4 = r2**2
    E = I0 * dz**2 / r4

    # Normalize to 0-255 grayscale for saving
    max_E = np.max(E) if np.max(E) > 0 else 1.0  # Avoid division by zero
    img = (E / max_E * 255).astype(np.uint8)

    # Save image as grayscale
    plt.imsave('illumination.png', img, cmap='gray')

    # Visualize distribution with inferno cmap and colorbar
    fig1.clf()
    ax1 = fig1.add_subplot(111)
    im = ax1.imshow(E, cmap='inferno', extent=[x.min(), x.max(), y.min(), y.max()], orig
    ax1.set_title('Illumination Distribution')
```

```
ax1.set_xlabel('X (mm)')
ax1.set_ylabel('Y (mm)')
cb = fig1.colorbar(im, ax=ax1)
cb.set_label('Illuminance (W/m²)')
canvas1.draw()

# Cross-sections through center
fig2.clf()

# Horizontal cross-section (along X at y closest to y0)
row = np.argmin(np.abs(y - y0))
E_line_horizontal = E[row, :]
x_line = x
ax2_horizontal = fig2.add_subplot(211)
ax2_horizontal.plot(x_line, E_line_horizontal)
ax2_horizontal.set_title('Horizontal Cross-Section (along X)')
ax2_horizontal.set_xlabel('X (mm)')
ax2_horizontal.set_ylabel('Illuminance (W/m²)')

# Vertical cross-section (along Y at x closest to x0)
col = np.argmin(np.abs(x - x0))
E_line_vertical = E[:, col]
y_line = y
ax2_vertical = fig2.add_subplot(212)
ax2_vertical.plot(y_line, E_line_vertical)
ax2_vertical.set_title('Vertical Cross-Section (along Y)')
ax2_vertical.set_xlabel('Y (mm)')
ax2_vertical.set_ylabel('Illuminance (W/m²)')

fig2.tight_layout()
canvas2.draw()

# Save cross-section plot
fig2.savefig('cross_section.png')

# Stats within circle
mask = (X - x0)**2 + (Y - y0)**2 <= R**2
if np.any(mask):
    E_circle = E[mask]
    max_c = np.max(E_circle)
    min_c = np.min(E_circle)
    avg_c = np.mean(E_circle)
else:
    max_c = min_c = avg_c = 0.0

# Exact values at specific points
def comp_E(xp, yp):
    rp = np.sqrt((xp - xL)**2 + (yp - yL)**2 + zL**2)
    return I0 * zL**2 / rp**4 if rp > 0 else 0.0
```

```
    E_center = comp_E(x0, y0)
    E_xp = comp_E(x0 + R, y0)
    E_xm = comp_E(x0 - R, y0)
    E_yp = comp_E(x0, y0 + R)
    E_ym = comp_E(x0, y0 - R)

    # Display results
    text = f"Center: {E_center:.10f}\nX +R: {E_xp:.10f}\nX -R: {E_xm:.10f}\nY +R: {E_yp:
    label_results.config(text=text)

# GUI setup
root = tk.Tk()
root.title("Illumination Calculator")

frame = ttk.Frame(root)
frame.pack(pady=10)

# Parameter inputs with defaults
labels = ["W (mm):", "H (mm):", "Wres (pixels):", "Hres (pixels):", "xL (mm):", "yL (mm)
defaults = ["2000", "2000", "400", "400", "0", "0", "1000", "1000", "0", "0", "800"]
entries = []

for i, label in enumerate(labels):
    ttk.Label(frame, text=label).grid(row=i, column=0, sticky="w")
    entry = ttk.Entry(frame)
    entry.grid(row=i, column=1)
    entry.insert(0, defaults[i])
    entries.append(entry)

entry_W, entry_H, entry_Wres, entry_Hres, entry_xL, entry_yL, entry_zL, entry_I0, entry_

ttk.Button(frame, text="Compute", command=compute_illum).grid(row=11, column=0, columnsp

# Results display
label_results = ttk.Label(root, text="", justify="left")
label_results.pack(pady=10)

# Matplotlib canvases
fig1 = plt.Figure(figsize=(5, 4))
canvas1 = FigureCanvasTkAgg(fig1, root)
canvas1.get_tk_widget().pack(side="left", padx=10)

fig2 = plt.Figure(figsize=(5, 4))
canvas2 = FigureCanvasTkAgg(fig2, root)
canvas2.get_tk_widget().pack(side="right", padx=10)

root.mainloop()
```

5 Результаты

Выбранные параметры (в пределах рекомендуемых диапазонов):

- $W = 2000$ mm, $H = 2000$ mm
- $W_{res} = 400$ pixels, $H_{res} = 400$ pixels
- $x_L = 0$ mm, $y_L = 0$ mm, $z_L = 1000$ mm
- $I_0 = 1000$ W/sr
- $x_0 = 0$ mm, $y_0 = 0$ mm, $R = 800$ mm

Расчетные значения освещенности (в Вт/м^2):

- Центр (0, 0): 0.0010000000
- X +R (800, 0): 0.0003718025
- X -R (-800, 0): 0.0003718025
- Y +R (0, 800): 0.0003718025
- Y -R (0, -800): 0.0003718025
- Максимум в круге: 0.0009999749
- Минимум в круге: 0.0003718214
- Среднее в круге: 0.0006097416

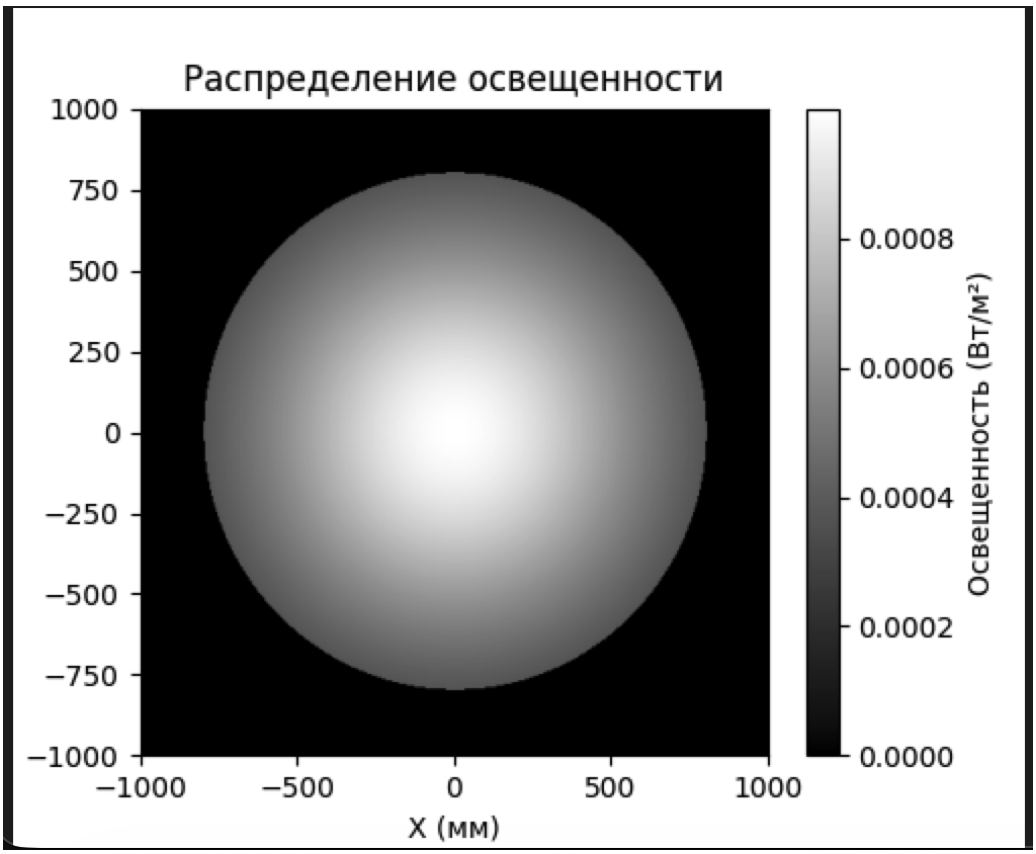

Рис. 2: Распределение освещенности

Результирующее изображение представляет собой 400x400 grayscale PNG (нормализованное 0-255) с более высокой интенсивностью в центре, угасающей наружу. В GUI распределение отображается с цветовой картой, указывающей фактическую освещенность от 0 до 0.001 Вт/м^2.

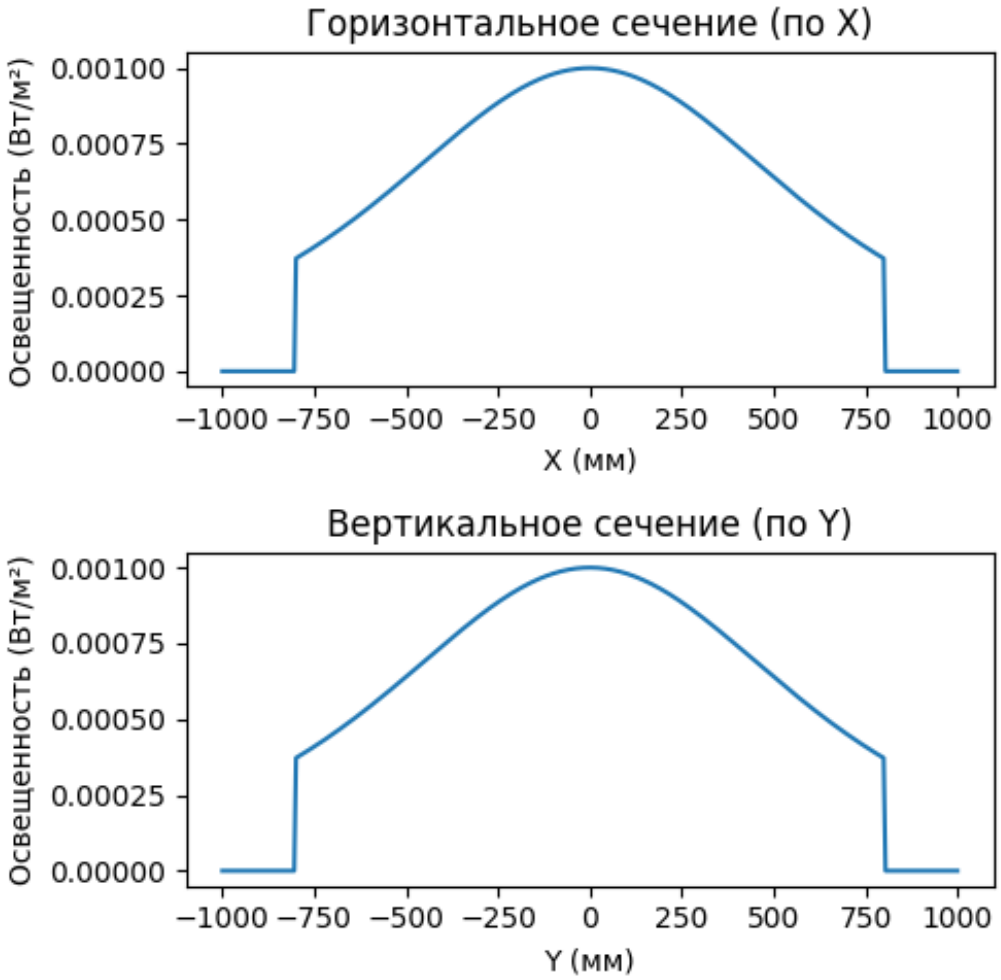

Рис. 3: Графики сечений через центр (горизонтальное и вертикальное)

Графики сечений представляют собой две симметричные кривые (горизонтальную и вертикальную) с пиком в 0.001 Вт/м^2 в центре.

6 Выводы

В ходе выполнения лабораторной работы были освоены навыки расчета и визуализации распределения освещенности на плоскости от точечного источника света с Ламбертовской диаграммой излучения. Разработанное приложение на Python позволяет динамически изменять параметры и визуализировать результаты, включая нормализованное изображение и графики сечений. Полученные результаты подтверждают теоретическую модель: освещенность максимальна под источником и убывает с расстоянием по закону обратных квадратов (учитывая косинусы). Работа способствует пониманию основ компьютерной графики в контексте моделирования освещения.

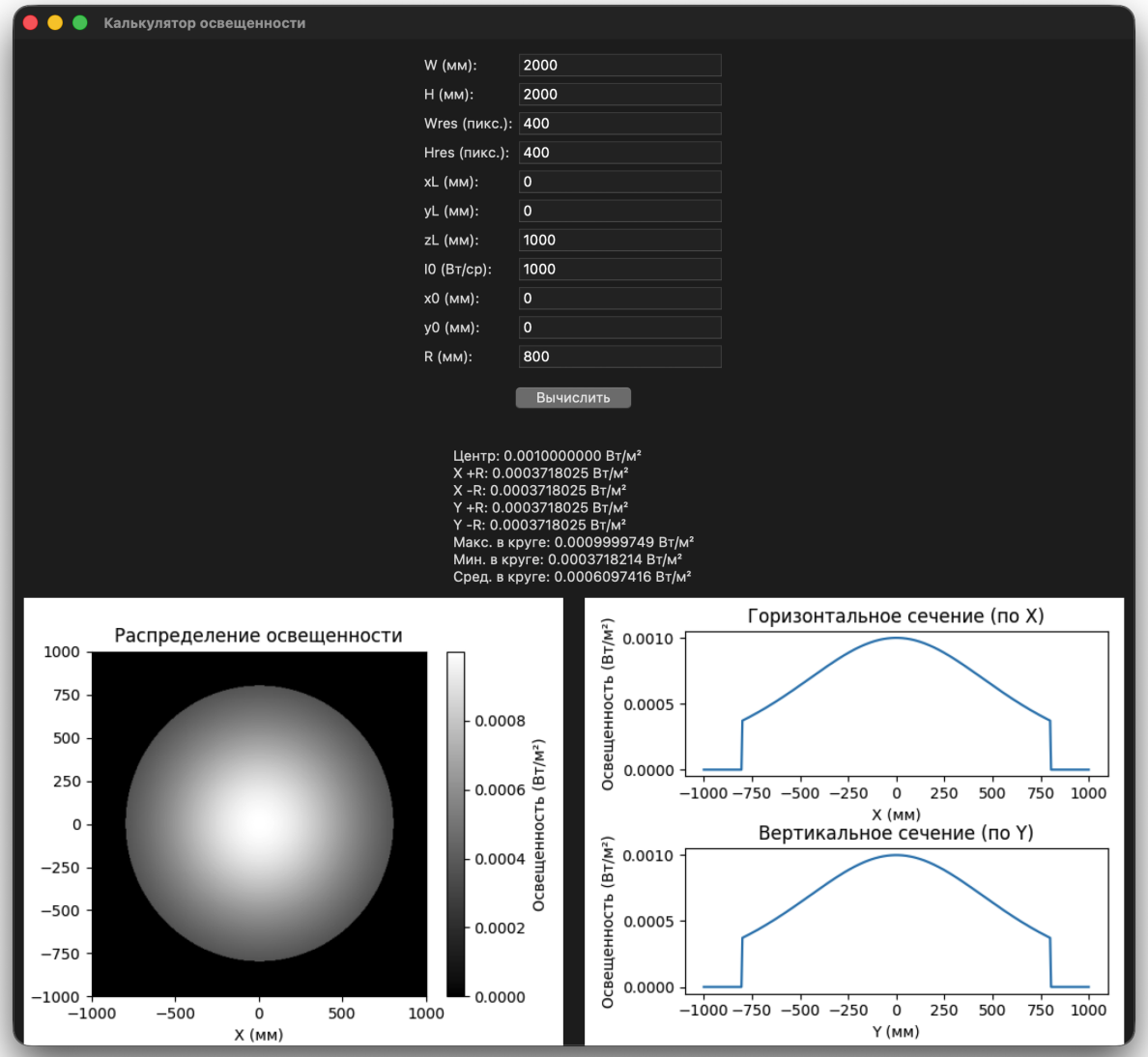

Рис. 4: Пример работы прогрммы