Системы компьютерной обработки изображений - Лабораторная работа № 5

Информация

Системы компьютерной обработки изображений

Лабораторная работа № 5

«Формирование изображения методом трассировки путей»

Выполнили студенты: Билошицкий Михаил Владимирович, P3416 Шпинёва Ульяна Сергеевна, P3416 Хоробрых Даниил Евгеньевич, P3416 Нестеренко Ксения Максимовна, P3416

Группа № P3416

Преподаватель: Жданов Дмитрий Дмитриевич

Ссылка на репозиторий: GitHub

Санкт-Петербург, 2025

---

1. Цель работы

Освоить методы синтеза изображений трёхмерных сцен с глобальным освещением методом трассировки путей (Path Tracing).

2. Задание

Построить изображение сцены Cornell Box с корректным глобальным освещением.

Требования: - Геометрия — треугольная сетка - Материалы — диффузное (Ламберт) и зеркальное отражение - Выборка по значимости для выбора типа отражения - Русская рулетка для ограничения глубины - NEE для прямого освещения - Точечная камера с антиалиасингом - Тонемаппинг и гамма-коррекция ($\gamma = 2.2$) - Сохранение в PPM

3. Теоретическая часть

3.1. Уравнение рендеринга

Основа path tracing — уравнение рендеринга Каджии:

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_\Omega f_r(x, \omega_i, \omega_o) \cdot L_i(x, \omega_i) \cdot (\omega_i \cdot n)\, d\omega_i$$

где: - $L_o(x, \omega_o)$ — исходящая яркость из точки x в направлении ω_o - $L_e(x, \omega_o)$ — собственное излучение поверхности - $f_r(x, \omega_i, \omega_o)$ — BRDF (функция отражательной способности) - $L_i(x, \omega_i)$ — входящая яркость из направления ω_i - $(\omega_i \cdot n)$ — косинус угла между направлением света и нормалью

Интеграл берётся по полусфере Ω над точкой.

3.2. Оценка методом Монте-Карло

Интеграл вычисляется численно через случайную выборку:

$$L_o \approx \frac{1}{N} \sum_{j=1}^{N} \frac{f_r(x, \omega_j, \omega_o) \cdot L_i(x, \omega_j) \cdot (\omega_j \cdot n)}{p(\omega_j)}$$

где $p(\omega_j)$ — плотность вероятности выбора направления ω_j.

3.3. BRDF материалов

Диффузное отражение (Ламберт):

$$f_r^{diffuse} = \frac{\rho_d}{\pi}$$

где ρ_d — альбедо (коэффициент диффузного отражения). Деление на π обеспечивает сохранение энергии.

Зеркальное отражение (идеальное зеркало):

Направление отражения:

$$\omega_r = \omega_i - 2(\omega_i \cdot n) \cdot n$$

BRDF зеркала — дельта-функция, вся энергия уходит строго в направлении отражения.

3.4. Выборка по значимости (Importance Sampling)

Для диффузных поверхностей используем косинус-взвешенную выборку по полусфере:

$$p(\omega) = \frac{\cos\theta}{\pi}$$

При такой выборке косинус в числителе и знаменателе сокращается, что уменьшает дисперсию оценки.

Генерация направления в локальных координатах:

$$\phi = 2\pi\xi_1, \quad \cos\theta = \sqrt{\xi_2}, \quad \sin\theta = \sqrt{1 - \xi_2}$$

где ξ_1, ξ_2 — случайные числа из $[0, 1]$.

3.5. Русская рулетка (Russian Roulette)

Для несмещённого обрыва путей используется русская рулетка. После глубины d_{rr} путь продолжается с вероятностью:

$$p_{continue} = \min(\max(\rho_d + \rho_s), 0.95)$$

При продолжении вес луча корректируется:

$$throughput = \frac{throughput}{p_{continue}}$$

Это даёт несмещённую оценку: пути, которые продолжились, компенсируют те, что оборвались.

3.6. Next Event Estimation (NEE)

Для уменьшения шума на диффузных поверхностях явно сэмплируем источник света. Выбираем случайную точку y на источнике и считаем вклад:

$$L_{direct} = \frac{L_e(y) \cdot f_r \cdot G(x,y) \cdot A_{light}}{1}$$

Геометрический фактор:

$$G(x,y) = \frac{(\omega \cdot n_x) \cdot (-\omega \cdot n_y)}{|x-y|^2}$$

где: - $\omega = \frac{y-x}{|y-x|}$ — направление от точки к источнику - n_x — нормаль в точке пересечения - n_y — нормаль источника света - A_{light} — площадь источника

Перед добавлением проверяем видимость теневым лучом (shadow ray).

3.7. Пересечение луча с треугольником

Используется алгоритм Мёллера-Трумбора. Луч: $R(t) = O + tD$, треугольник с вершинами V_0, V_1, V_2.

Точка на треугольнике в барицентрических координатах:

$$P = (1-u-v)V_0 + uV_1 + vV_2$$

Решаем систему:

$$O + tD = (1-u-v)V_0 + uV_1 + vV_2$$

Получаем:

$$\begin{bmatrix} t \\ u \\ v \end{bmatrix} = \frac{1}{(D \times E_2) \cdot E_1} \begin{bmatrix} (T \times E_1) \cdot E_2 \\ (D \times E_2) \cdot T \\ (T \times E_1) \cdot D \end{bmatrix}$$

где $E_1 = V_1 - V_0, E_2 = V_2 - V_0, T = O - V_0$.

Условия пересечения: $t > 0, u \geq 0, v \geq 0, u + v \leq 1$.

3.8. Постобработка

Тонемаппинг Рейнхарда (HDR → LDR):

$$L_{out} = \frac{L_{in}}{1 + L_{in}}$$

Гамма-коррекция:

$$V_{out} = V_{in}^{1/\gamma}, \quad \gamma = 2.2$$

4. Реализация

4.1. Структура проекта

```
src/
├── renderer.py    – трассировка путей
├── scene.py       – сцена и пересечения
├── geometry.py    – ray-triangle intersection
├── camera.py      – генерация лучей
├── cornell_box.py – создание сцены
├── math_utils.py  – векторные операции
└── postprocess.py – тонемаппинг, сохранение
```

4.2. Трассировка пути

```python
@njit(cache=True, fastmath=True)
def trace_path(ray_origin, ray_dir, vertices, normals, material_ids,
               diffuse, specular, emission,
               light_indices, light_areas, total_light_area,
               max_depth, rr_depth):
    color = np.zeros(3)
    throughput = np.ones(3)

    current_origin = ray_origin.copy()
    current_dir = ray_dir.copy()

    for depth in range(max_depth):
        t, hit_point, normal, mat_id = intersect_scene(
            current_origin, current_dir, vertices, normals, material_ids
```

```
        )

        if mat_id < 0: break

        mat_diffuse = diffuse[mat_id]
        mat_specular = specular[mat_id]
        mat_emission = emission[mat_id]

        # Попали в источник света
        if mat_emission[0] > 0 or mat_emission[1] > 0 or mat_emission[2] > 0:
            color = color + throughput * mat_emission
            break

        # Русская рулетка
        total_refl = mat_diffuse + mat_specular
        p_continue = min(max(total_refl[0], max(total_refl[1], total_refl[2])), 0.95)

        if depth >= rr_depth:
            if np.random.random() > p_continue: break
            throughput = throughput / p_continue

        # Выбор типа отражения по весам
        diff_weight = max(mat_diffuse)
        spec_weight = max(mat_specular)
        total_weight = diff_weight + spec_weight

        if total_weight < 1e-6: break

        p_diffuse = diff_weight / total_weight

        if np.random.random() < p_diffuse:
            # Диффузное: NEE + случайное направление
            # ... код NEE ...
            new_dir = random_cosine_hemisphere(normal)
            throughput = throughput * mat_diffuse
            current_dir = new_dir
        else:
            # Зеркальное
            new_dir = reflect(current_dir, normal)
            throughput = throughput * mat_specular
            current_dir = new_dir

        current_origin = hit_point + new_dir * 0.0001

    return color
```

4.3. Алгоритм Мёллера-Трумбора

```
@njit(cache=True, fastmath=True)
def ray_triangle_intersect(ray_origin, ray_dir, v0, v1, v2):
    EPSILON = 1e-7
    edge1 = v1 - v0
    edge2 = v2 - v0
    h = cross(ray_dir, edge2)
    a = dot(edge1, h)

    if abs(a) < EPSILON: return -1.0

    f = 1.0 / a
    s = ray_origin - v0
    u = f * dot(s, h)

    if u < 0.0 or u > 1.0: return -1.0

    q = cross(s, edge1)
    v = f * dot(ray_dir, q)

    if v < 0.0 or u + v > 1.0: return -1.0

    t = f * dot(edge2, q)
    if t < EPSILON: return -1.0

    return t
```

4.4. Параметры рендеринга

Настройки в `main.py`:

```
CONFIG = {
    'width': 600,
    'height': 600,
    'samples_per_pixel': 512,
    'max_depth': 6,
    'russian_roulette_depth': 2,

    'camera_position': [0, 2.5, -8],
    'camera_look_at': [0, 2.5, 0],
    'camera_fov': 45,

    'room_size': 5.0,
    'left_wall_color': [0.75, 0.25, 0.25],   # красная
    'right_wall_color': [0.25, 0.25, 0.75],  # синяя
    'wall_color': [0.75, 0.75, 0.75],

    'light_intensity': 15.0,
```

```
    'light_color': [1.0, 0.95, 0.9],
    'light_size': 1.8,

    'box1_diffuse': [0.75, 0.75, 0.75],      # левый куб — диффузный
    'box1_specular': [0.0, 0.0, 0.0],
    'box2_diffuse': [0.05, 0.05, 0.05],      # правый куб — зеркальный
    'box2_specular': [0.9, 0.9, 0.9],

    'gamma': 2.2,
    'exposure': 1.5,
}
```

5. Результат

На изображении видны эффекты глобального освещения: - Color bleeding — красный и синий оттенки от стен переносятся на кубы и пол - Мягкие тени от площадного источника света - Отражения окружения в зеркальном кубе

Файл также сохранён как `output/result.ppm`.

6. Вывод

Реализован path tracer с поддержкой диффузных и зеркальных материалов. Для ускорения сходимости применены NEE и русская рулетка. Результат — физически корректное изображение Cornell Box.

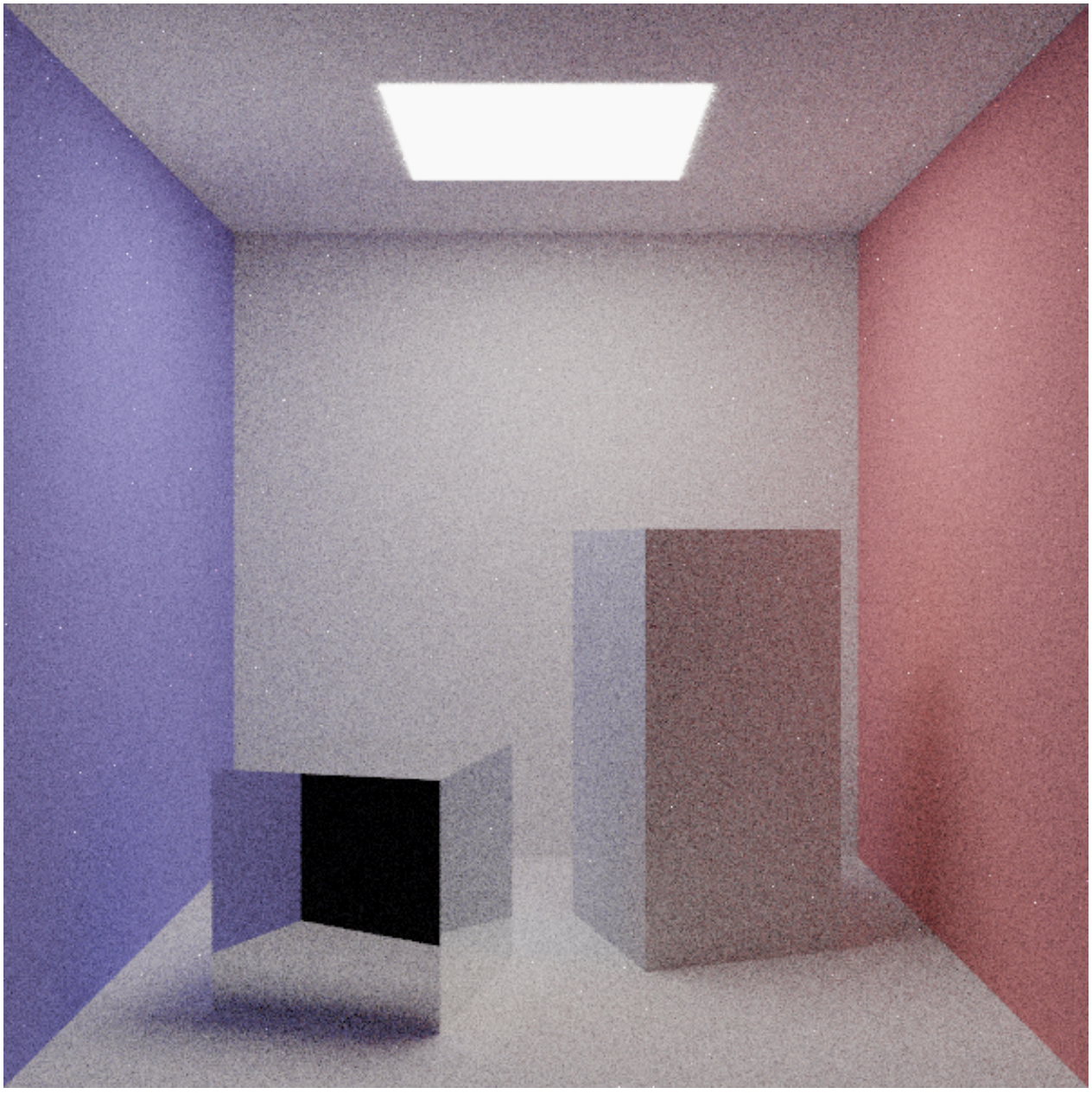

Figure 1: Результат