Классы

- <u>Sphere</u>:

Поля:

x, y, z - центр сферы;

R, G, B - цвет;

Rad - радиус;

surface_type - тип поверхности (0 = матовая, 1 = глянцевая)

Методы:

Sphere(double x, double y, double z, double Rad, int R, int G, int B, int surface_type); - конструктор

bool traceback(vector<double> start_point, vector<double> trace_ray, vector<double> &result, int trace_mode = 1) const - трассировка.

start_point - источник луча, trace_ray - трассирующий луч, result - вектор для записи результатов, trace_mode - режим трассировки (1 = обычная, не 1 - трассировка теней, ускоренный вариант)

void print() const - печать параметров сферы.

- <u>Box:</u>

Поля:

vector<double> v1,...,v8 - вершины бокса

vector<double> center - центр масс (для оптимизации)

double half_diag_2 - квадрат половины диагонали (для оптимизации)

surface_type - тип поверхности (0 = матовая, 1 = глянцевая)

R, G, B - цвет

vector<vector<double>> edge_1, ..., edge_6 - грани прямоугольника как наборы вершин, формируются в конструкторе.

Методы:

Box(double x1, double y1, double z1, double x8, double y8, double z8, int R, int G, int B, int surface_type) - конструктор (точка с индексом "1" имеет наименьшие координаты, с индексом "8" - наибольшие)

bool internal(vector<double> point, int edge_number) const - принадлежит ли точка point грани с номером edge_number

bool traceback(vector<double> start_point, vector<double> trace_ray, vector<double> &result) const - трассировка, аналогична сфере.

void print() const - печать параметров бокса.

<u>- Tetra:</u>

Поля:

vector<double> v1, ..., v4 - вершины тетраэдра.

vector<vector<double>> edge_1, ... , edge_4 - грани тетраэдра как наборы точек

vector<double> optimize_center - центр описанной сферы (для оптимизации)

double sphere_center_dist_2 - квадрат расстояние от центра описанной сферы до любой вершины (для оптимизации)

R, G, B - цвет

surface_type - тип поверхности (0 = матовая, 1 = глянцевая)

Методы:

Tetra(double x1, double y1, double z1, ..., double x4, double y4, double z4, R, G, B, surface_type) - конструктор.

bool internal(vector<double> point, vector<vector<double>> verticies) const - лежит ли точка point внутри треугольника verticies (через векторные произведения)

bool traceback(vector<double> start_point, vector<double> trace_ray, vector<double> &result) const - трассировка

void print() const - печать параметров тетраэдра

Функции

1) vector<double> vectmul(vector<double> v1, vector<double> v2) - векторное произведение трёхмерных векторов

2) double scalmul(vector<double> v1, vector<double> v2) - скалярное произведение трёхмерных векторов

3) void to_length(vector<double> &v, double length) - приводит вектор v к длине length

4) void printv(vector<double> v) - печатает вектор v (для отладки)

5) void scene_properties(string filename,
vector<double> &light,
vector<double> &camera,
vector<double> &upvector,
vector<double> &screen_normal,
int &screen_distance,
int &screen_width,
int &screen_height,
int &depth_of_view) - считывает параметры трёхмерной сцены в переданные переменные

6) void scene_objects(string filename,
vector<Sphere> &spheres,
vector<Box> &boxes,
vector<Tetra> &tetras) - считывает заданные обьекты трёхмерной сцены в переданные переменные

7) double volume_det(vector<double> v1, vector<double> v2, vector<double> v3) - вычисляет обьем трёхмерного параллелепипеда на векторах v1, v2, v3

8) double vectmul_area(vector<double> v1, vector<double> v2) - вычисляет площадь параллелограма, натянутого на векторы v1, v2

9) double dist_2(vector<double> v1, vector<double> v2) - квадрат расстояния между точками v1, v2 (почти везде достаточно квадрата без затратного вычисления корня)

10) double light_coef(vector<double> light_direction, vector<double> surf_normal, int surface_type, int obj_type) - вычисляет коэффициент интенсивности света в зависимости от типа обьекта, типа поверхности и угла падения

11) int nearest_point(vector<double> point, vector<vector<double>> points) - возвращает номер ближайшей из points точки к точке point

12) vector<double> operator*(vector<double> v, double m);
vector<double> operator+(vector<double> v1, vector<double> v2);
vector<double> operator-(vector<double> v1, vector<double> v2) - перегрузка операторов под векторы

13) bool gen_frames(string series_path, string series_name, int iterations, int scenario) - генерирует инструкции для указанного числа кадров по указанному номеру сценария (сценарии пишутся в теле функции)

14) bool render_frames_(string frames_path,
string frames_name,
string draw_path,
string draw_name,
int i_start,
int i_end) - рендерит кадры по указанным инструкциям в указанном промежутке (номера кадров от i_start до i_end)

15) bool render_frames(string frames_path,
string frames_name,
string draw_path,
string draw_name,
int iterations,
int thread_number = 4) - распоточивающая обёртка вокруг render_frames_ - рендерит указанное число кадров по указанным инструкциям, в пуле на указанное число потоков

16) void traceback(string scene_props, string objs, string save_to) - обычная трассировка (принимает инструкции сцены, инструкции обьектов, имя картинки к сохранению)

17) void trace_cycle(int h_start,
int h_end,
int width,
Canvas& frame,
const vector<double>& left_upper_corner,
const vector<double>& upvector,
const vector<double>& ortsup,
const vector<double>& camera,
const vector<double>& light,
const vector<Sphere>& spheres,
const vector<Box>& boxes,
const vector<Tetra>& tetras) - трассирует полосу экрана (разбиение экрана на горизонтальные полосы высоты от h_start до h_end)

18) bool traceback_parallel(string scene_props, string objs, string save_to, int thread_number) - распоточивающая обёртка вокруг trace_cycle, трассирует сцену в пуле на указанное число потоков (высота экрана должна нацело делиться на число потоков для равномерного разбиения на полосы трассировки)

Форматы входных файлов:
Файл с инструкциями для сцены:

camera x y z
normal_ x y z // вектор к центру экрана
light x y z
upvector x y z // вектор для задания ориентации верх-низ
screen_dist A // Расстояние от камеры до экрана
screen_height H
screen_width W
view_depth - поле видимости

Файл с обьектами:

Sphere R G B x y z Rad surface
Box R G B x1 y1 z1 x8 y8 z8 surface
Tetra R G B x1 y1 z1 ... x4 y4 z4 surface