Фотореалистичный рендеринг *(aka raytracing)*

Практика 1

2024

Напоминание об автотестах

- Каждая практика – программа, по описанию входной сцены генерирующая картинку (‘*рендер*’)
- Автотестированием занимается телеграм-бот `mkn-raytracing-2024-bot`
- Боту нужно послать команду вида

```
/submit practice1 https://github.com/vanya/practice1.git
  [ path-inside-repo [ branch ] ]
```

- В репозитории по указанному пути должны быть скрипты `build.sh` и `run.sh`
- Скрипт `run.sh` запускается с двумя параметрами: путь до файла с описанием сцены, и путь до выходного файла с картинкой

Формат сцены

- Текстовый формат, простой для ручного парсинга
- Каждая строка файла – отдельная команда, задающая какие-то параметры сцены
- Команды выглядят как `COMMAND_NAME <arg1> <arg2> ...`, все аргументы – вещественные числа
- Неизвестные команды *нужно пропускать*

Формат сцены: описание команд

- `DIMENSIONS <width> <height>` – размеры изображения, которое нужно сгенерировать
- `BG_COLOR <red> <green> <blue>` – цвет фона сцены (значения в диапазоне [0..1])
- `CAMERA_POSITION <x> <y> <z>` – координаты позиции камеры
- `CAMERA_RIGHT <x> <y> <z>` – ось ‘вправо’ камеры
- `CAMERA_UP <x> <y> <z>` – ось ‘вверх’ камеры
- `CAMERA_FORWARD <x> <y> <z>` – ось ‘вперёд’ камеры
- `CAMERA_FOV_X <angle>` – угол обзора камеры по ширине (в радианах)
- N.B.: fov_Y вычисляется на основе fov_X, $width$, $height$ (см. слайды лекции)

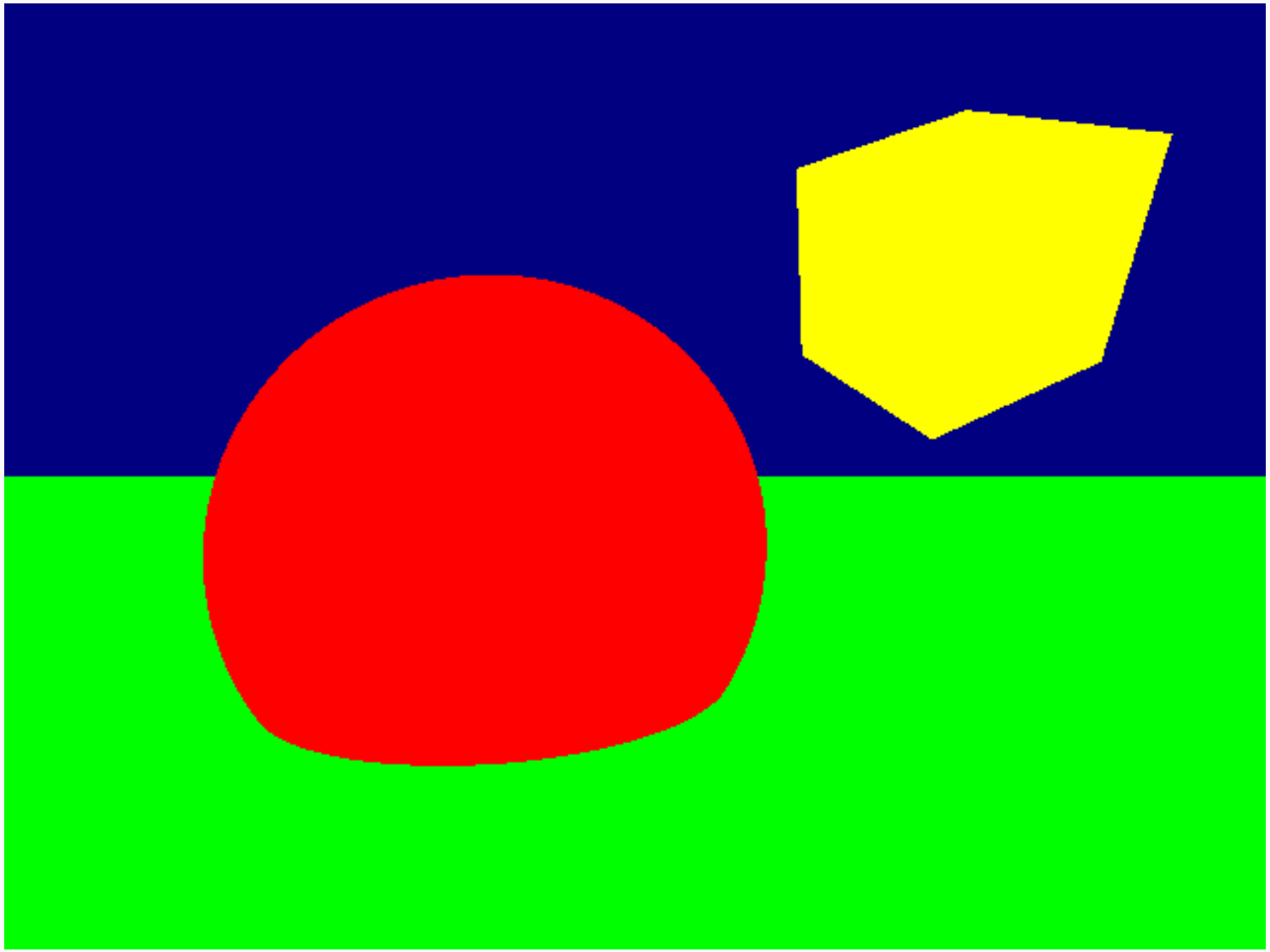
Формат сцены: описание команд

- `NEW_PRIMITIVE` – добавить новый объект в список объектов сцены (дальнейшие команды описывают последний созданный объект)
- `PLANE <nx> <ny> <ny>` – задать геометрию объекта – плоскость с заданным вектором нормали
- `ELLIPSOID <rx> <ry> <ry>` – задать геометрию объекта – эллипсоид с заданными радиусами
- `BOX <sx> <sy> <sy>` – задать геометрию объекта – параллелепипед с заданными размерами (как в лекции)
- `POSITION <x> <y> <y>` – координаты центра объекта, по умолчанию – $(0, 0, 0)$
- `ROTATION <x> <y> <y> <w>` – кватернион вращения объекта, по умолчанию – $(0, 0, 0, 1)$
- `COLOR <red> <green> <blue>` – цвет объекта

Пример сцены

```
DIMENSIONS 640 480

BG_COLOR 0 0 0.5

CAMERA_POSITION 0 1.5 0
CAMERA_RIGHT 1 0 0
CAMERA_UP 0 1 0
CAMERA_FORWARD 0 0 -1
CAMERA_FOV_X 1.5708

NEW_PRIMITIVE
ELLIPSOID 2 2 2
POSITION -1 1 -5
COLOR 1 0 0

NEW_PRIMITIVE
PLANE 0 1 0
COLOR 0 1 0

NEW_PRIMITIVE
BOX 0.5 0.5 0.5
POSITION 1.5 2.5 -3
ROTATION 0.31246 0.15623 0.15623 0.92388
COLOR 1 1 0
```

Пример сцены



Формат изображения

- NetPBM P6: текстово-бинарный формат, удобный для чтения и записи
- Имеет расширение файла `.ppm` (вам его дописывать не нужно)
- Первая строка файла: `P6`
- Вторая строка файла: `<width> <height>` (нужно подставить ширину и высоту изображения)
- Третья строка файла: `255` (максимальное значение цветовых компонент, у нас всегда 255)
- После этого – пиксели изображения в сыром, бинарном формате (всего `width * height * 3` байт)

Советы

- Начните с парсинга, и реализуйте команды по одной
- Удобно начать тестировать с камеры в центре координат и сферы, на которую смотрит камера (должен появиться кружок на экране)
- Удобно разбить код на функции, например

```
ray generate_ray(camera, pixel_coord);
optional<float> intersection(ray, plane);
optional<float> intersection(ray, sphere);
optional<float> intersection(ray, box);
optional<float> intersection(ray, object);
optional<pair<float, color>> intersection(ray, scene);
color raytrace(scene, ray);
image generate_image(scene);
```