3D-Renderer с нуля

Выполнил: Амеличев Константин, ПМИ 191

Научный руководитель: Трушин Дмитрий Витальевич, ФКН ВШЭ

Цель работы

Разработка библиотеки для отрисовки объектов из 3d-пространства

Из пререквизитов — только отрисовка пикселей на экране.

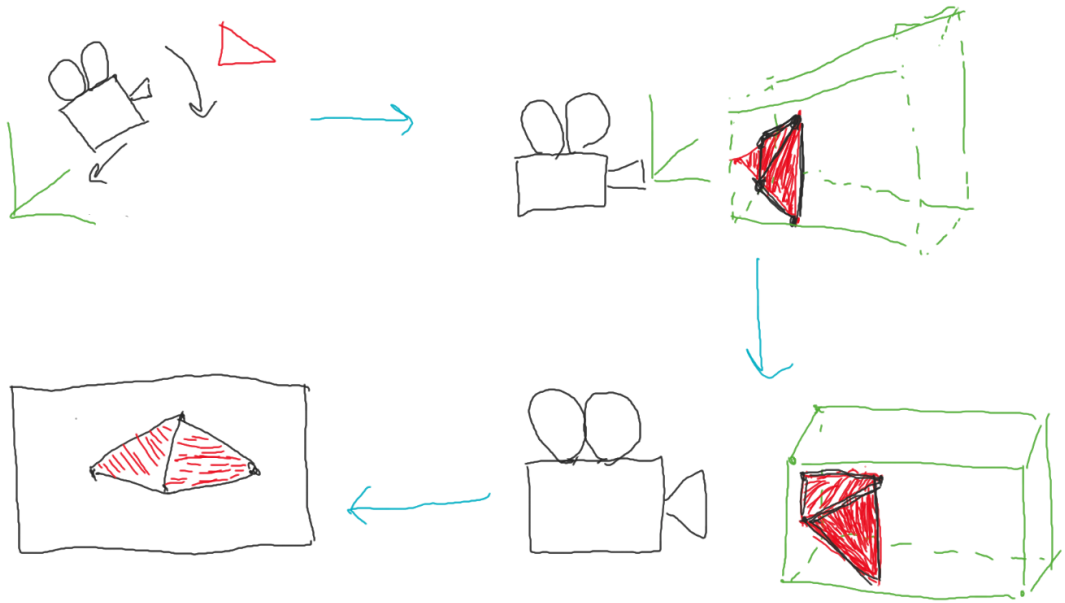
Поставленные задачи

- Изучение теории
- Сборка пайплайна
- Создание приложения
- Тестирование
- Документация

Изучение теории

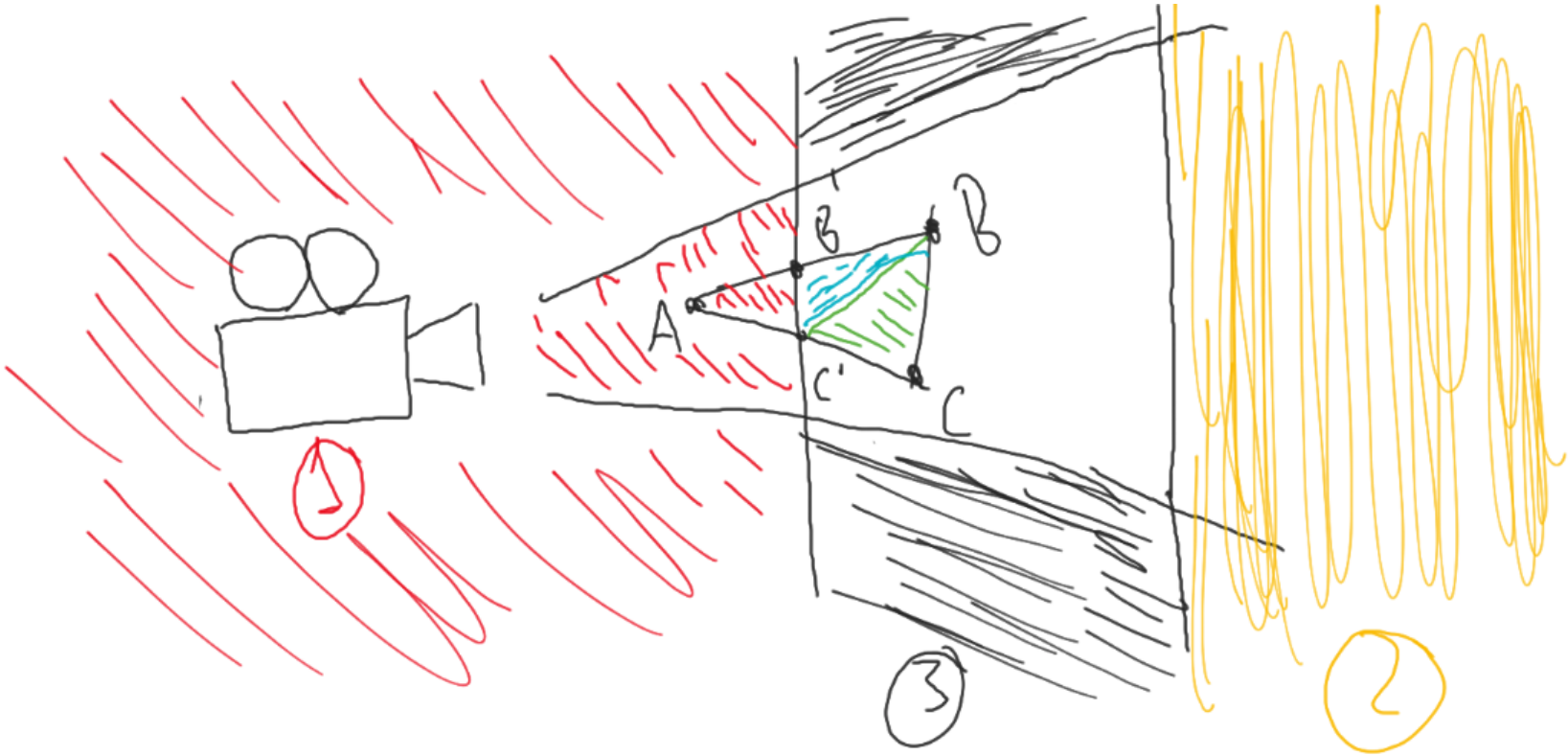
- Однородные координаты для работы с 3d-пространством
  - $\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \Longleftrightarrow \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \\ 1 \end{pmatrix} \Longleftrightarrow \begin{pmatrix} \frac{x}{w} \\ \frac{y}{w} \\ \frac{z}{w} \end{pmatrix}$
    - Расширяется возможность линейных отображений
- Повороты и сдвиги
- Проективное преобразование
- Работа с матрицами

Pipeline



- Перенос объекта в систему координат камеры.
- Клиппинг (выбор объектов для отображения).
- Проективное преобразование
- Растеризация

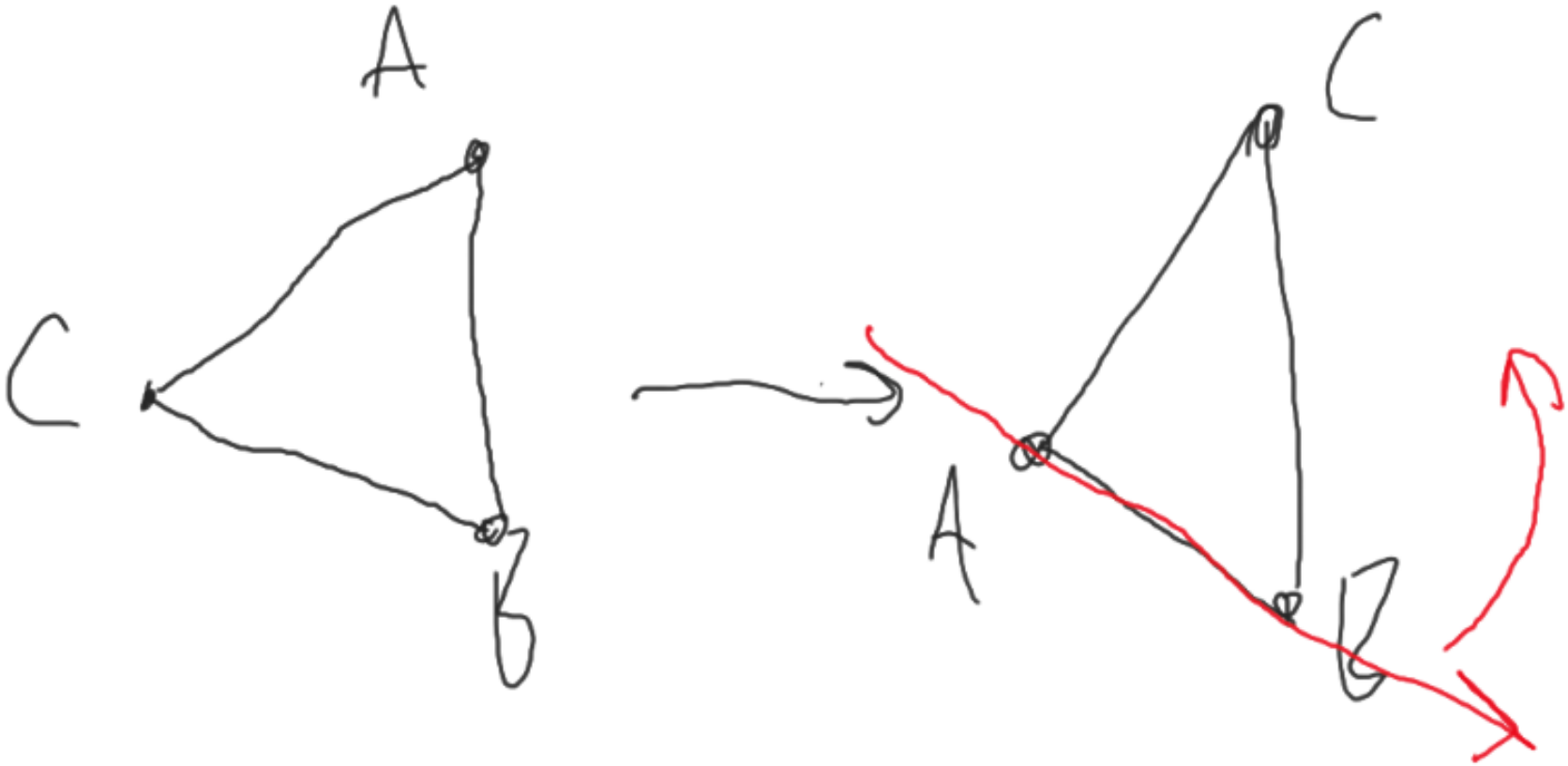
Алгоритмическая часть: Клиппинг



Отсекаем невидимые области:

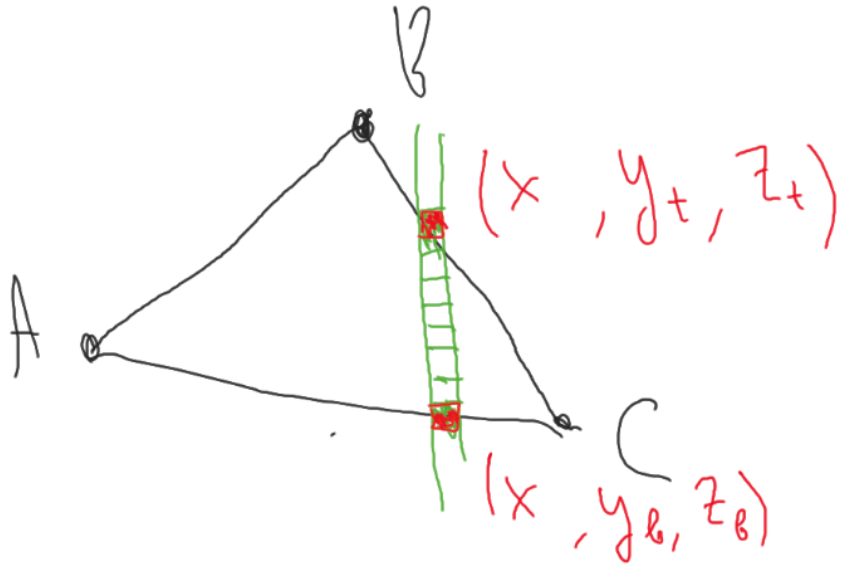
1. Отсекается клиппингом, за передней плоскостью
2. Слишком большая z-value
3. За границами 2д-экрана

Алгоритмическая часть: Растеризация



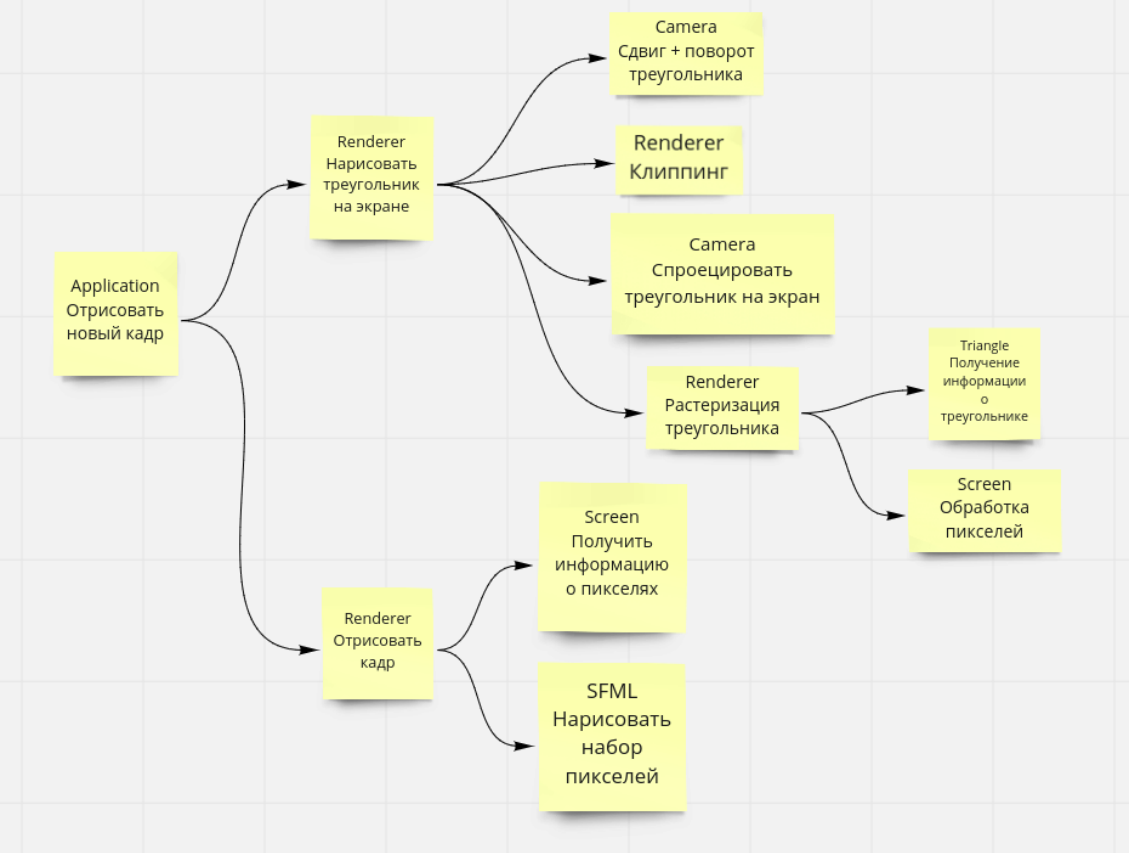
Теперь можно легко выделить вертикальную полосу в треугольнике.

Алгоритмическая часть: Растеризация



1. Нахождение y_b, y_t — пересечение вертикальной прямой и треугольника
2. Нахождение z_b, z_t — перенос двумерной точки (x, y) обратно в трехмерное пространство.
3. Нахождение z-value внутри — линейно относительно z_b, z_t

Схема взаимодействия

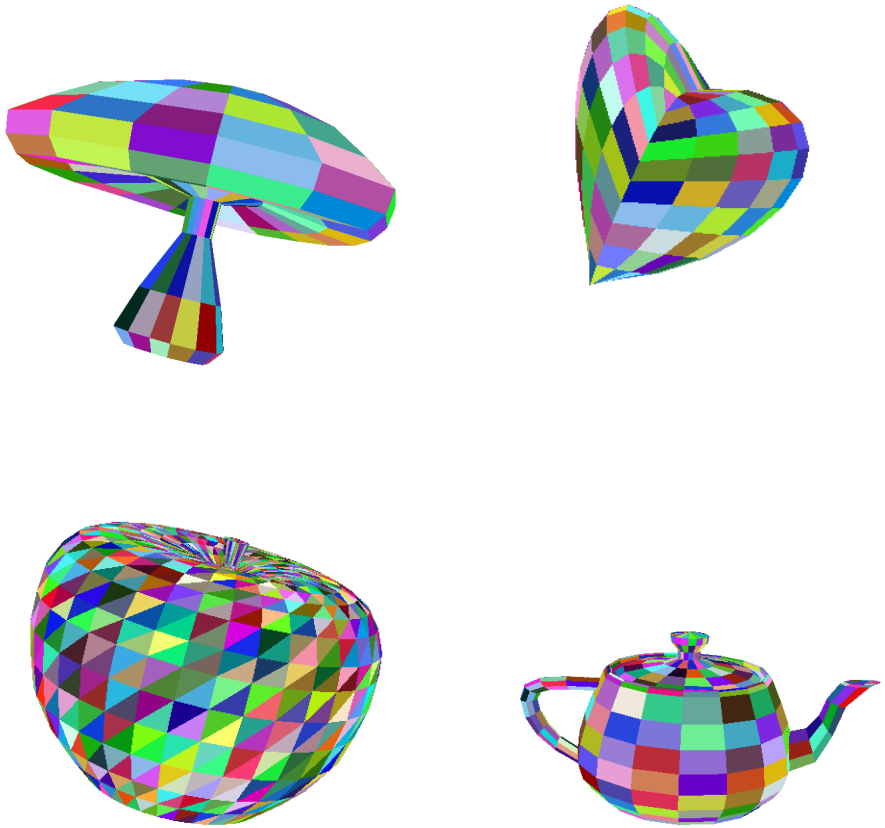


- Renderer
- Camera
- World
- Window (SFML)

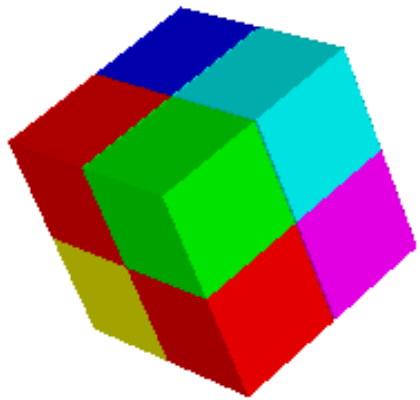
Программная часть

- Application — тестовое приложение
- Doxygen — сопроводительная документация
- CxxTest + Github Actions — автоматические unit-тесты
- GProf — профайлер для исследования производительности

Парсер .off-формата



Тени



Тень представляется как функция $\varphi : [0; 2\pi) \to [0; 1]$ от угла между направлением света и нормалью к плоскости, на которую домножается свет.

Скорость работы

- Согласно профилировщику GProf, большая часть времени уходит на процессы, связанные с растеризацией и пиксельным экраном.
- 30% производительности уходит непосредственно на методы *Screen::validate_pixel*, *Screen::set_pixel*, и вспомогательные.
- Кстати, во время разработки простой $x_{start} = \max(0, x_{start})$ заметно ускорил приложение.

Результаты

- Библиотека:
  - разработана
  - протестирована
  - задокументирована
- Теория изучена
- Приложение сделано
- Дополнительно сделан парсинг 3d-моделей
- Дополнительно сделаны тени

Спасибо за внимание!