Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»
КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Домашнее задание по лабораторной работе №4 по дисциплине «Анализ алгоритмов»

Тема: Графовые модели описания алгоритмов
Студент: Карпова Е. О.
Группа: ИУ7-52Б
Оценка (баллы): _____
Преподаватели: Волкова Л. Л., Строганов Ю. В.

Москва — 2022 г.

1. Выполнение

Цель работы: получение навыков построения графовых моделей описания алгоритмов на основе фрагмента алгоритма обратной трассировки лучей, представив граф управления, информационный граф, операционную историю и информационную историю по выбранному фрагменту.

Задачи работы:

1) построение графа управления для фрагмента алгоритма обратной трассировки лучей;

2) построение информационного графа для фрагмента алгоритма обратной трассировки лучей;

3) построение операционной истории для фрагмента алгоритма обратной трассировки лучей;

4) построение информационной истории для фрагмента алгоритма обратной трассировки лучей;

В листинге 1.1 представлена функция прохода по пикселам растра, необходимая в реализации алгоритма обратной трассировки лучей.

Листинг 1.1 — Листинг функции реализации алгоритма обратной трассировки лучей (начало)

```
std::shared_ptr<QImage> Drawing::drawFigures()
{
    std::shared_ptr<QImage> image =
    std::make_shared<QImage>(canvasWidth, canvasHeight,
    QImage::Format_RGB32);
    image->fill(Qt::black);

    QVector3D cam = QVector3D(0, 0, 3000);
    sight_t sight = {
        .cam = cam
    };

    for (int y = 0; y < canvasHeight; y++)
        for (int x = 0; x < canvasWidth; x++) {
            QVector3D pix = QVector3D(x, y, 200);
            QVector3D dir = (pix - cam).normalized();

            sight.dir = dir;
            QColor refColor = castRay(sight, 0);
            image->setPixel(x, y, qRgb(refColor.red(),
            refColor.green(), refColor.blue()));
        }

    return image;
}
```

В листинге 1.2 представлена фрагмент функции прохода по пикселам растра, распараллеливание которой по теоретической оценке должно значительно увеличить эффективность алгоритма.

Листинг 1.2 — Листинг функции реализации алгоритма обратной трассировки лучей (начало)

```
for (int y = 0; y < canvasHeight; y++)
        for (int x = 0; x < canvasWidth; x++) {
            QVector3D pix = QVector3D(x, y, 200);
            QVector3D dir = (pix - cam).normalized();

            sight.dir = dir;
            QColor refColor = castRay(sight, 0);
            image->setPixel(x, y, qRgb(refColor.red(),
            refColor.green(), refColor.blue()));
        }
```

Данные, по которым данный фрагмент зависит от остальной части функции, далее рассматриваются как константные, так как не будут влиять на оценку целесообразности распараллеливания данного участка.

На рисунках 1.1 – 1.4 представлены вышеупомянутые графовые модели для фрагмента алгоритма из листинга 1.2.

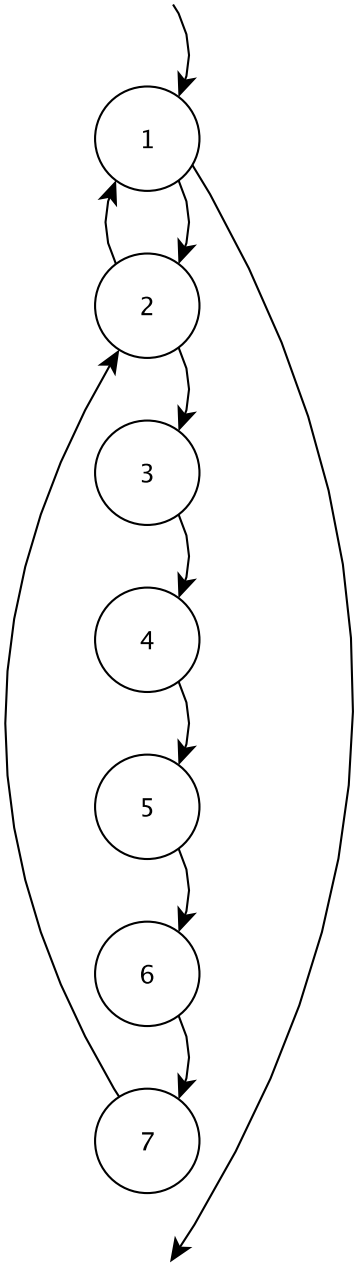

Рисунок 1.1 — Граф управления

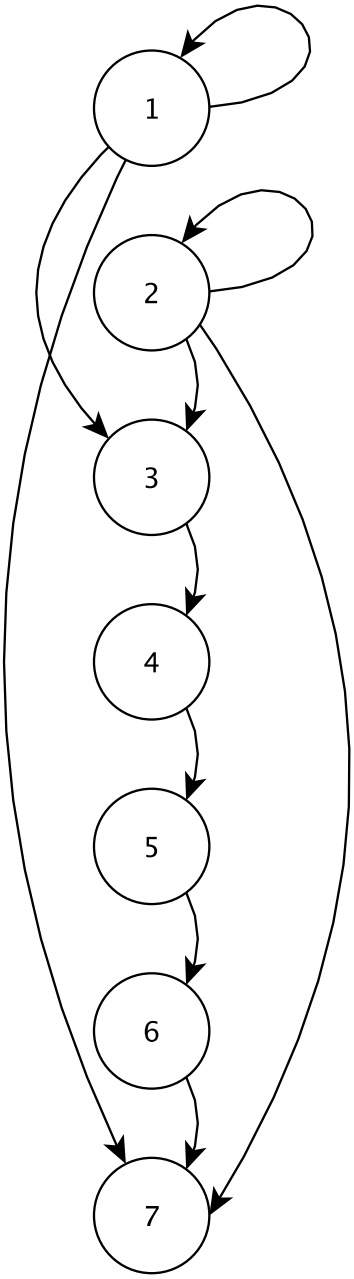

Рисунок 1.2 — Информационный граф

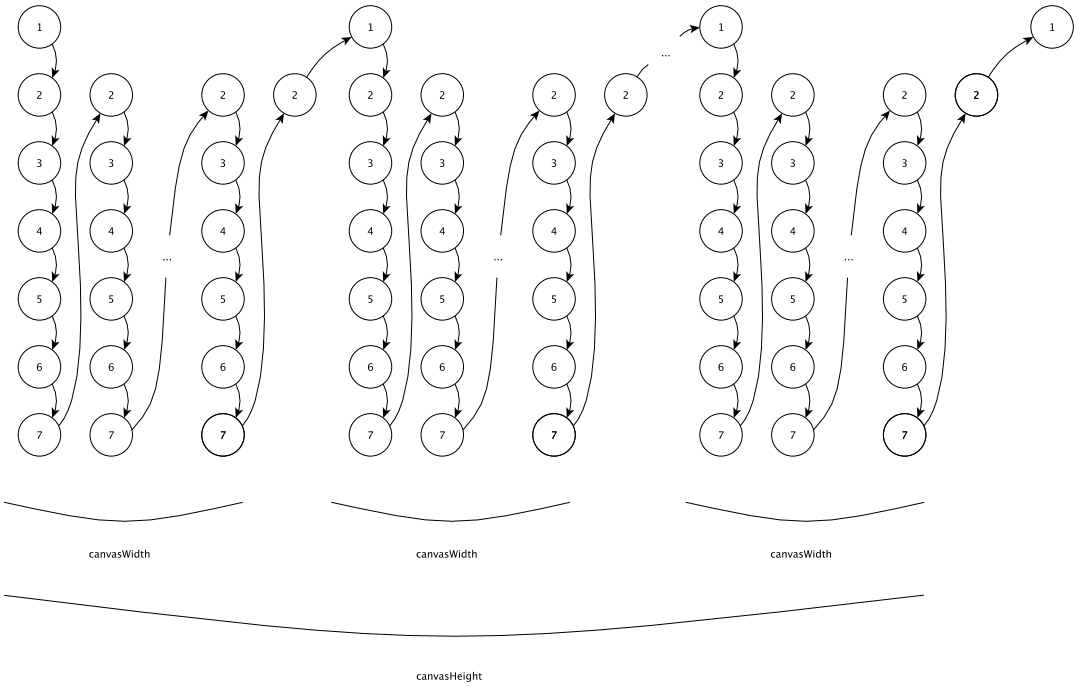

Рисунок 1.3 — Операционная история

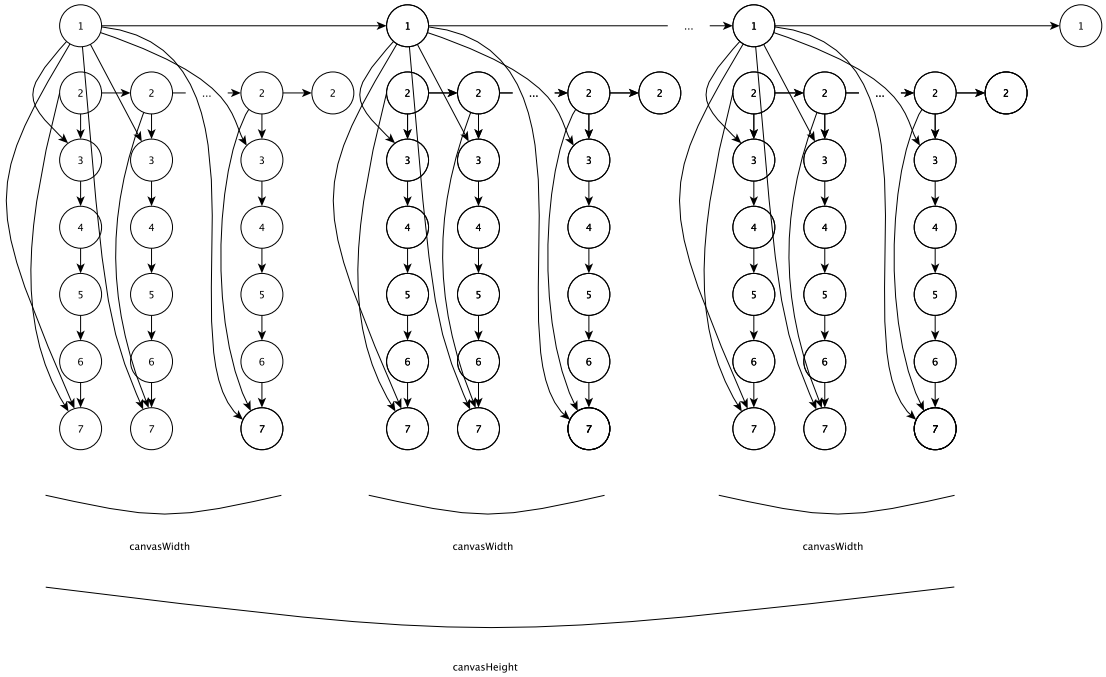

Рисунок 1.4 — Информационная история