Московский Государственный Университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики

Скрябин Глеб Денисович

Разработка технологий 3D визуализации информационных графов алгоритмов

Выпускная квалификационная работа

Научный руководитель: к.ф.-м.н., вед.н.с. Антонов Александр Сергеевич
Москва, 2023

Понятие информационного графа

Информационный граф алгоритма — ациклический граф, вершины которого соответствуют операциям алгоритма, а дуги - связям по данным между этими операциями.

Две вершины связываются дугой, если вторая использует данные, вычисленные в первой.

Анализ информационного графа позволяет, например, определить множества независимых друг от друга операций, найти подходящее распределение операций по процессорам вычислительной системы, обнаружить узкие места

Цель работы и постановка задачи

Целью работы является разработка системы 3D визуализации графов алгоритмов. Данная система должна:

1. иметь возможность загрузки входных файлов в формате XML
2. автоматически преобразовывать его в промежуточную JSON структуру
3. передавать промежуточный файл с JSON структурой в систему визуализации графов алгоритмов
4. в рамках системы визуализации автоматически преобразовывать промежуточные данные во внутреннюю структуру, хранящуюся в архитектуре, которая должна соответствовать требованиям к визуализации
5. предоставлять возможности интерактивного анализа графов

Цель работы и постановка задачи

Решение описанных задач подразумевает многоэтапный процесс, который включает в себя:

1. Формирование понятия внутреннего представления графа алгоритма.
2. Формирования архитектуры приложения, основываясь на требования к визуализации
3. Создание web-страницы, на которой производится визуализация графа алгоритма.
4. Создание программного средства, преобразующего промежуточное представление графа алгоритма в JSON структуре в формат множества 3D моделей
5. Создание программного средства, производящего непосредственно 3D визуализацию множества готовых объектов в рамках web-страницы и обеспечивающего возможности интерактивного анализа визуализированного графа.

Способ связи двух частей системы

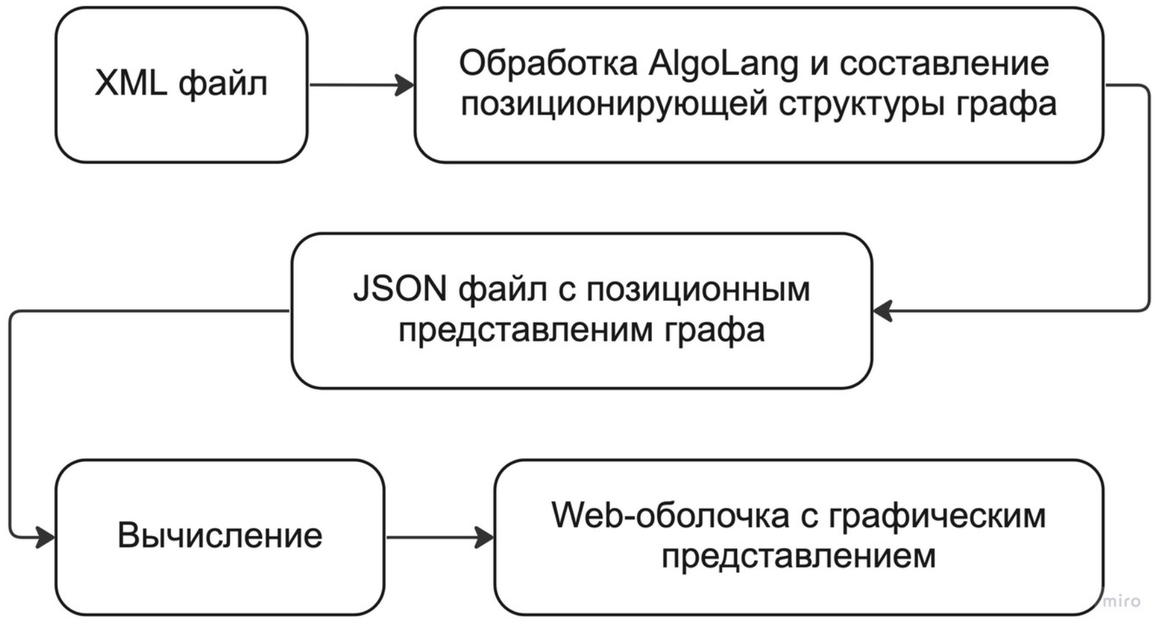


рис 1. Блок-схема алгоритм связи двух частей системы

Требуемый формат входных данных

Для работы системы требуется стандартизированное представление графа алгоритма. В данном случае исходным форматом данных для системы является JSON структура, содержащая информацию о координатах и типе вершин, и информацию о связанности этих вершин через их уникальные идентификаторы. Эта структура имеет следующий вид (пример из трех вершин и 2 дуг):

```
{
    "vertices": [
        { "id": 0, "coordinates": [0, 0, 0], "type": "0" },
        { "id": 1, "coordinates": [1, 0, 0], "type": "2" },
        { "id": 2, "coordinates": [2, 0, 0], "type": "2" }
    ],
    "edges": [
        { "id": 0, "sourceVertexId": 0, "targetVertexId": 1, "type": "0" },
        { "id": 1, "sourceVertexId": 1, "targetVertexId": 2, "type": "0" }
    ]
}
```

рис 2. JSON структура, пример исходного формата данных

Требуемый формат входных данных

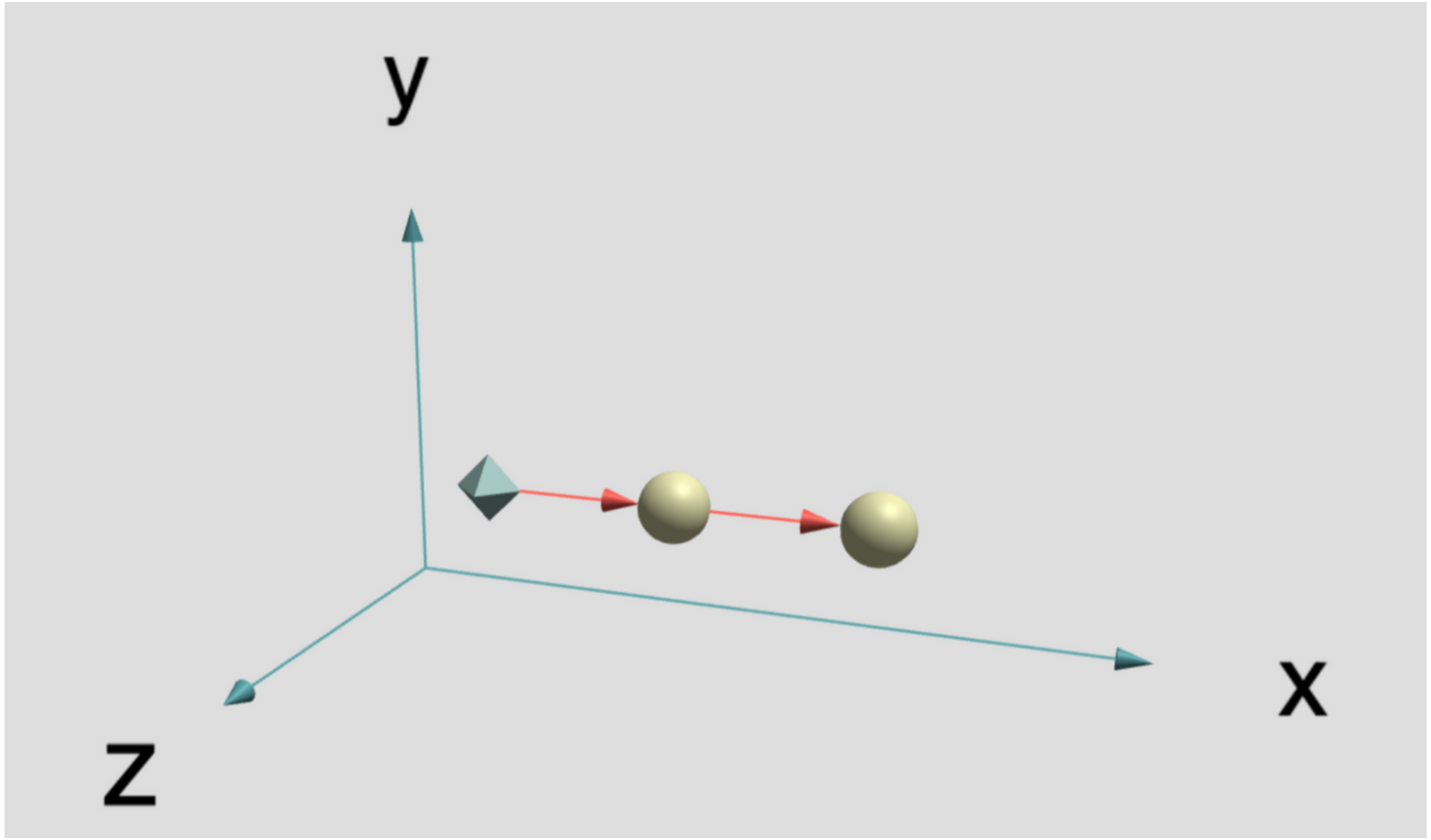


рис 3. Пример визуализации графа, описанного примере в JSON структуре

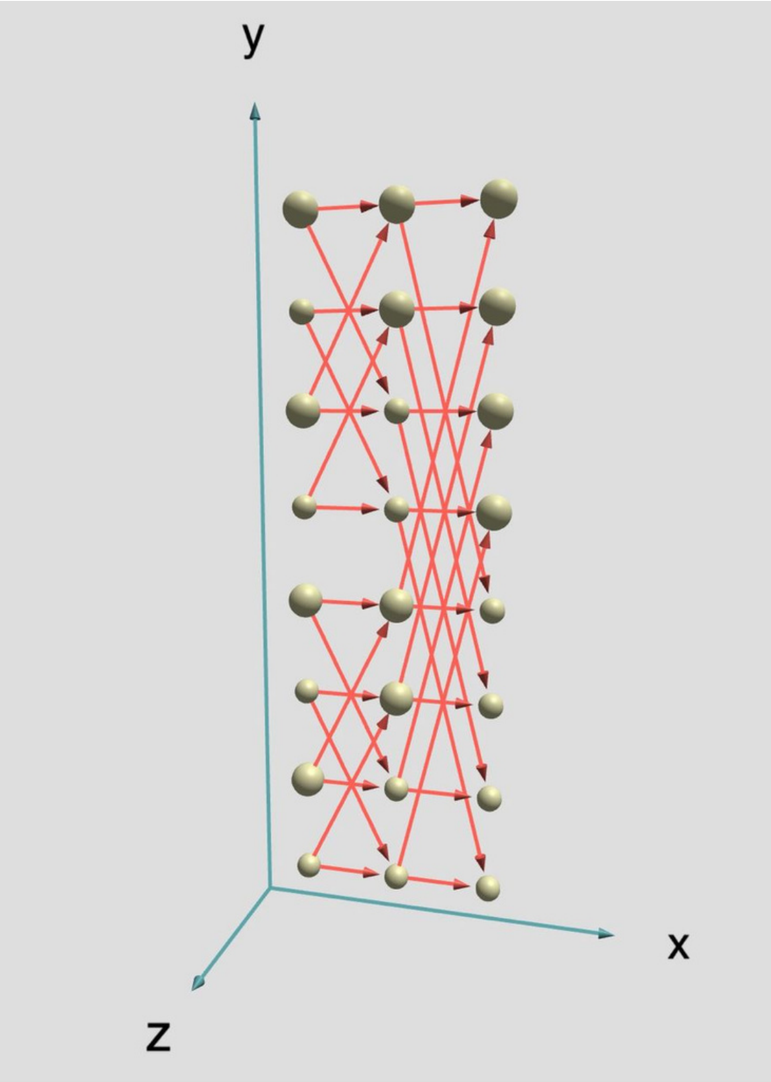


рис 4. Пример визуализации графа

Требования к визуализации и возможностям анализа

- Интерактивная визуализация должна работать в режиме реального времени на стандартном домашнем ПК и иметь интерфейс для взаимодействия с системой. Внешний вид интерфейса и графическая визуализация должны быть интуитивно понятны и удобны для работы при помощи компьютерной мыши и клавиатуры.
- Графическое представление графа должно содержать максимально возможное количество дополнительной информации согласно требованиям стандарта визуализации.
- Для удобного анализа требуется поддерживать разные настройки вида, такие как перспектива, проекция графа на плоскости, черно-белый режим для принтера, использование текстур и теней, настройка осей координат. Настройки цветов и толщин линий, размер вершин тоже влияет на удобство визуального анализа.

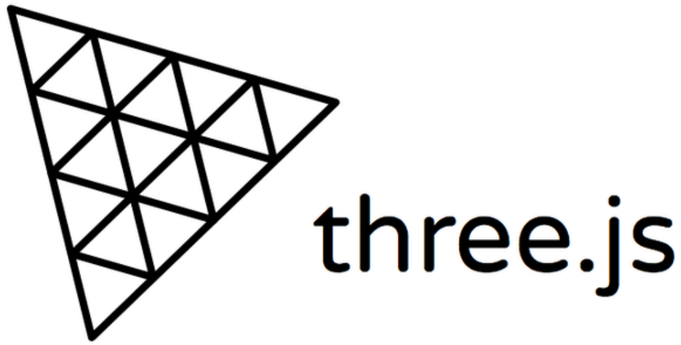
Разбиение системы на смысловые части и выбор архитектуры системы визуализации

Основные составные части проекта:

- Работа с данными, считывание и создание внутренней структуры графа
- Работа со структурой графа, основываясь на параметры обзора. Например ярусно параллельная форма.
- Создание 3D объектов для каждого объекта в структуре графа, основываясь на параметры обзора. Например ЧБ режим, режим без теней или без стрелок.
- Создание 3D сцены, содержащей модель графа.
- Работа с пользователем. Обеспечение связи панели управления с моделью графа, сцены и общим параметрам обзора. При изменении ключевых значений модели данных должны выборочно меняться и последовательно обновлять структуру графа и 3D объекты, представляющие этот граф на сцене.

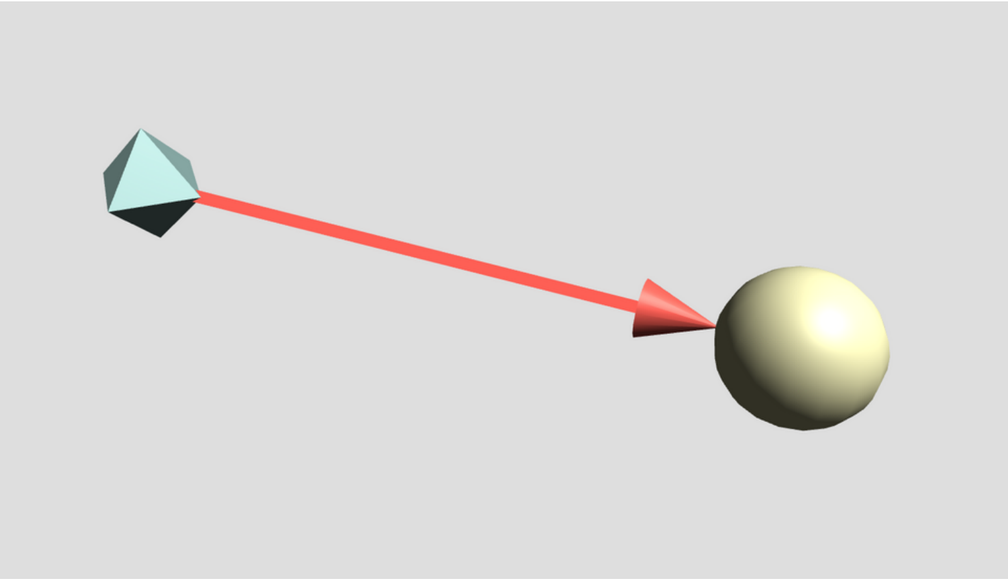
Описанные требования и основные функциональные части являются достаточными критериями для выбора **MVC** архитектуры.

Используемый стек технологий



- JavaScript;
- Библиотека Three.js;
- Модуль dat.gui.js;
- Модуль OrbitControls.js.

Реализованные технологии оптимизации



- В качестве линий для соединения вершин используются 2D объекты, нарисованный по параметрическому уравнению

- в качестве надписей используется прозрачная плоскость, на которую накладывается текстура с текстом

Примеры результата работы системы визуализации


```
{
        "vertices": [
                { "id": 0, "coordinates": [1, 0, 0], "type": "1" },
                { "id": 1, "coordinates": [0, 0, 0], "type": "0" },
                { "id": 2, "coordinates": [0, 1, 0], "type": "0" },
                { "id": 3, "coordinates": [1, 1, 0], "type": "1" },
                { "id": 4, "coordinates": [0, 2, 0], "type": "0" },
                { "id": 5, "coordinates": [0, 3, 0], "type": "0" },
                { "id": 6, "coordinates": [1, 2, 0], "type": "1" },
                { "id": 7, "coordinates": [0, 4, 0], "type": "0" },
                { "id": 8, "coordinates": [0, 5, 0], "type": "0" },
                { "id": 9, "coordinates": [1, 3, 0], "type": "1" },
                { "id": 10, "coordinates": [0, 6, 0], "type": "0" },
                { "id": 11, "coordinates": [0, 7, 0], "type": "0" },
                { "id": 12, "coordinates": [2, 0, 0], "type": "1" },
                { "id": 13, "coordinates": [2, 1, 0], "type": "1" },
                { "id": 14, "coordinates": [3, 0, 0], "type": "1" }
        ],
        "edges": [
                { "id": 0, "sourceVertexId": 1, "targetVertexId": 0, "type": "0" },
                { "id": 1, "sourceVertexId": 2, "targetVertexId": 0, "type": "0" },
                { "id": 2, "sourceVertexId": 4, "targetVertexId": 3, "type": "0" },
                { "id": 3, "sourceVertexId": 5, "targetVertexId": 3, "type": "0" },
                { "id": 4, "sourceVertexId": 7, "targetVertexId": 6, "type": "0" },
                { "id": 5, "sourceVertexId": 8, "targetVertexId": 6, "type": "0" },
                { "id": 6, "sourceVertexId": 10, "targetVertexId": 9, "type": "0" },
                { "id": 7, "sourceVertexId": 11, "targetVertexId": 9, "type": "0" },
                { "id": 8, "sourceVertexId": 0, "targetVertexId": 12, "type": "0" },
                { "id": 9, "sourceVertexId": 3, "targetVertexId": 12, "type": "0" },
                { "id": 10, "sourceVertexId": 6, "targetVertexId": 13, "type": "0" },
                { "id": 11, "sourceVertexId": 9, "targetVertexId": 13, "type": "0" },
                { "id": 12, "sourceVertexId": 12, "targetVertexId": 14, "type": "0" },
                { "id": 13, "sourceVertexId": 13, "targetVertexId": 14, "type": "0" }
        ]
}
```

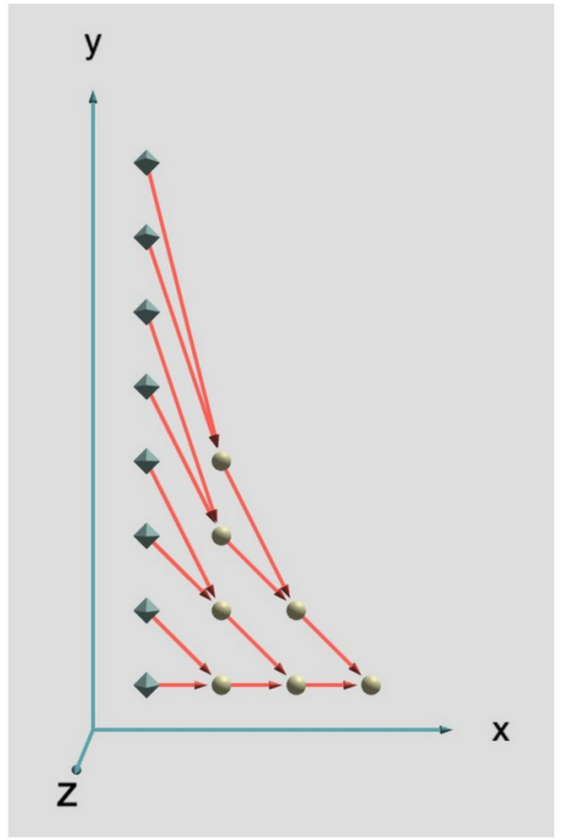



рис 5. Цепочка преобразований JSON структуры в 3D-визуализацию (алгоритм умножения матрицы на вектор)

Внешний вид

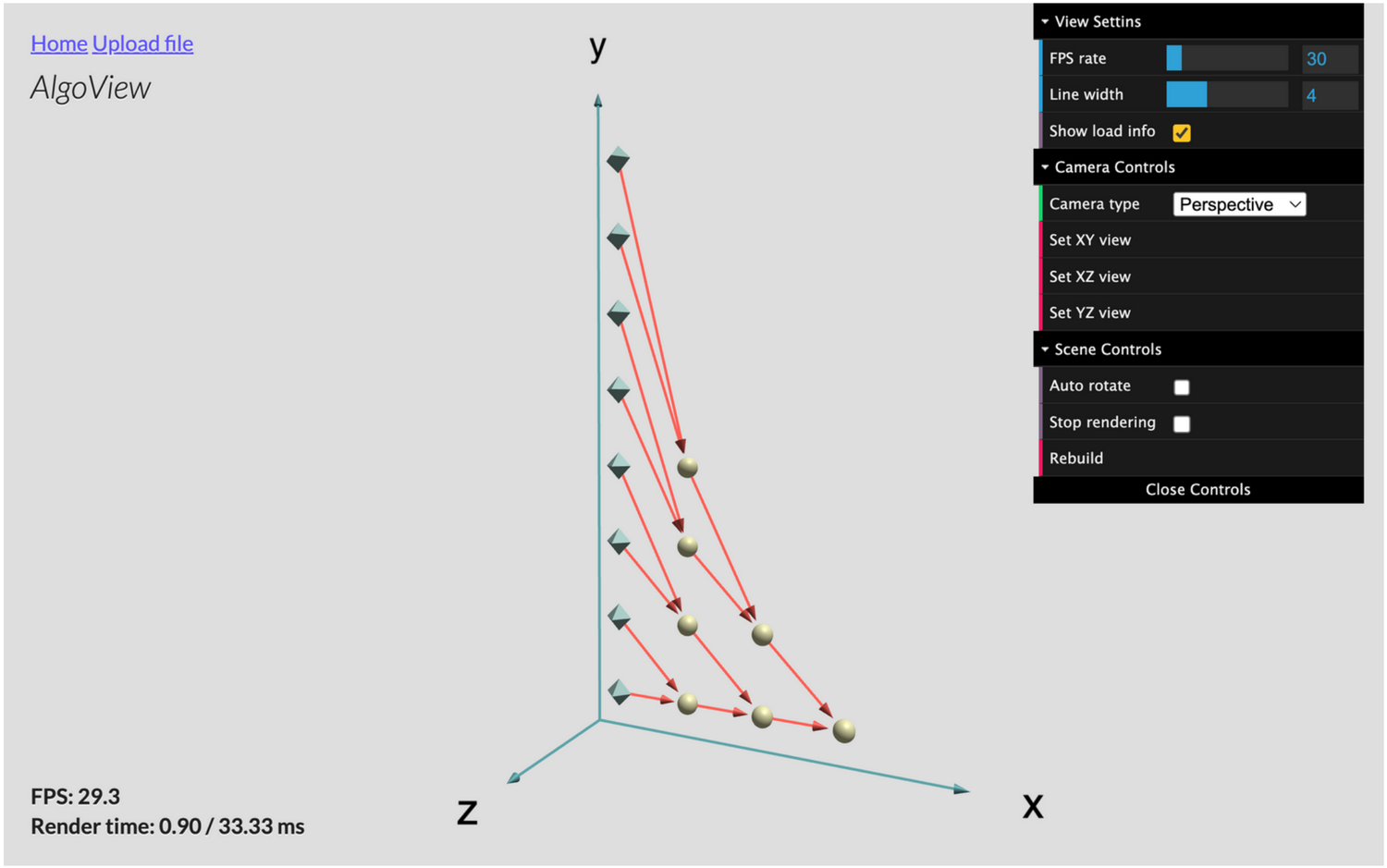


рис 6. Пример web 3D-визуализации (алгоритм умножения матрицы на вектор)

Результаты работы

1. Реализован алгоритм преобразования промежуточных данных в JSON формате во внутреннее представление данных, понятное для библиотек 3D визуализации
2. Создано веб приложение для представления системы в готовом виде, в котором можно загружать XML файлы и получать интерактивную 3D визуализацию, соответствующую описанным требованиям
3. Реализованы методы оптимизации для уменьшения нагрузки на графическое ядро браузера
4. Предоставлены возможности удобного визуального анализа графа алгоритмов