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. Система должна:

- Предоставлять возможность загрузки файлов на языке AlgoLoad в формате XML и автоматически преобразовывать их стандартную JSON структуру при помощи программного обеспечения в рамках общей системы визуализации графов алгоритмов.
- Преобразовывать промежуточные данные в интерактивную 3D модель графа алгоритма.

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

- Формирование архитектуры приложения и понятия внутреннего представления графа алгоритма.
- Создание программного средства, преобразующего промежуточное представление графа алгоритма в JSON структуре в формат множества 3D моделей.
- Создание программного средства, производящего непосредственно интерактивную 3D визуализацию множества сформированных объектов, представляющих собой граф алгоритма.
- Создание web-страницы, на которой производится визуализация и взаимодействие с интерфейсом.

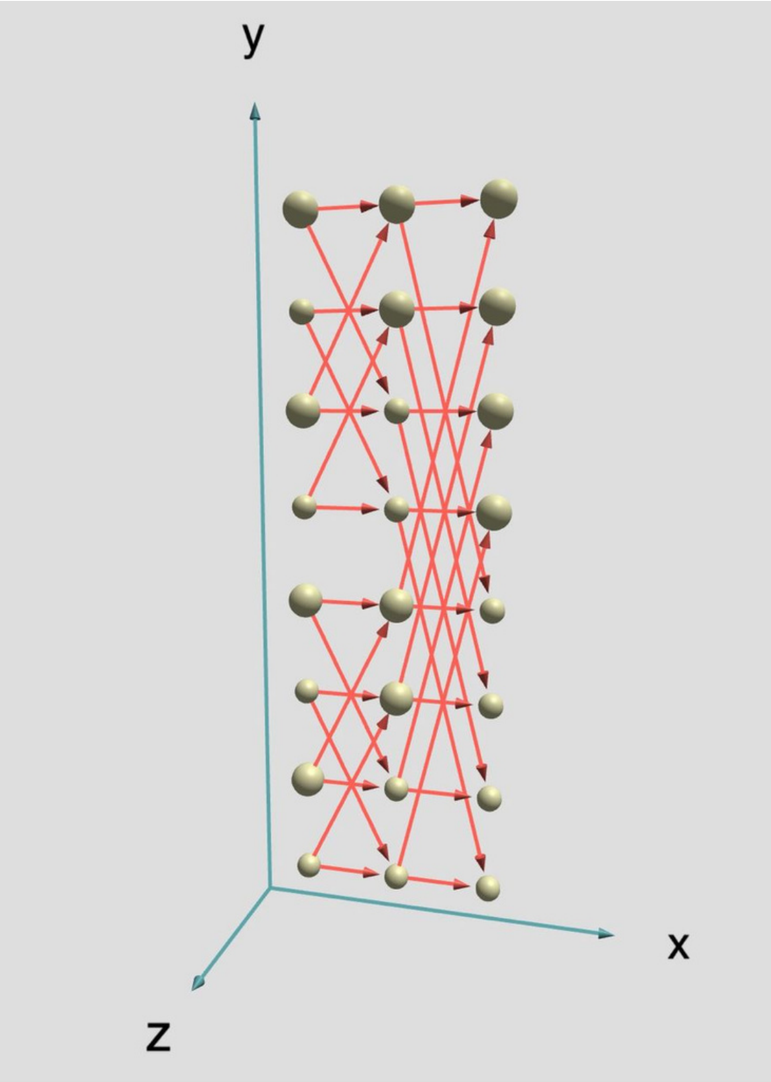


рис 1. Пример визуализации графа

Требования к визуализации и возможностям анализа

- Интерактивная визуализация должна работать на обычном 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 информации.
- Для удобного анализа требуется поддерживать разные настройки вида, такие как перспектива, проекция графа на плоскости, настройка осей координат.

Способ связи двух частей системы

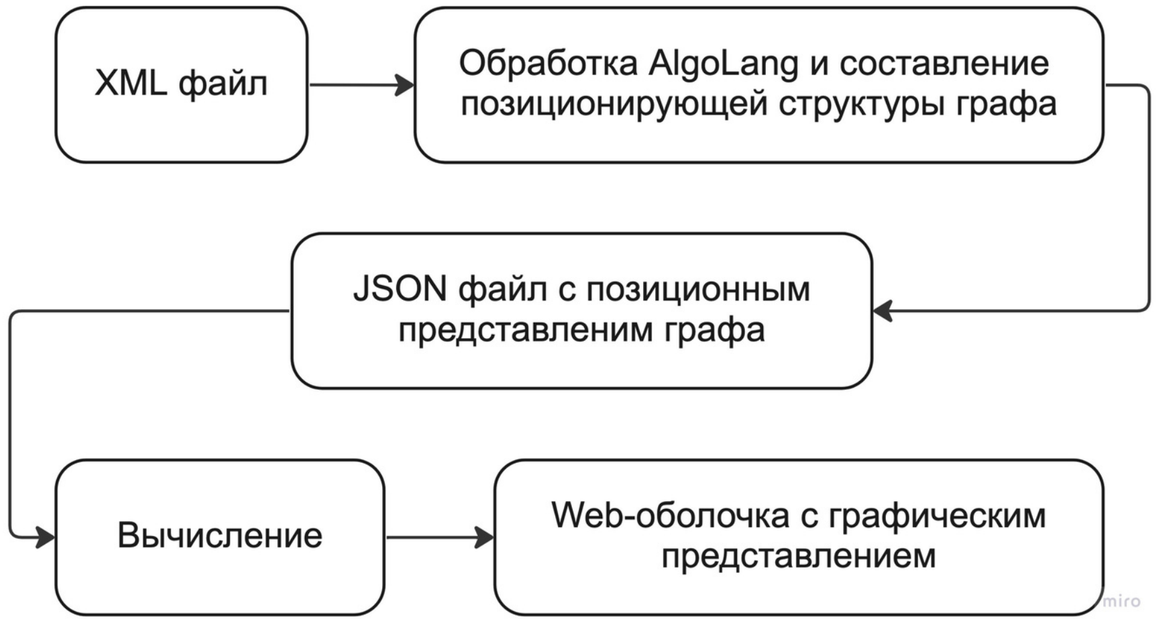
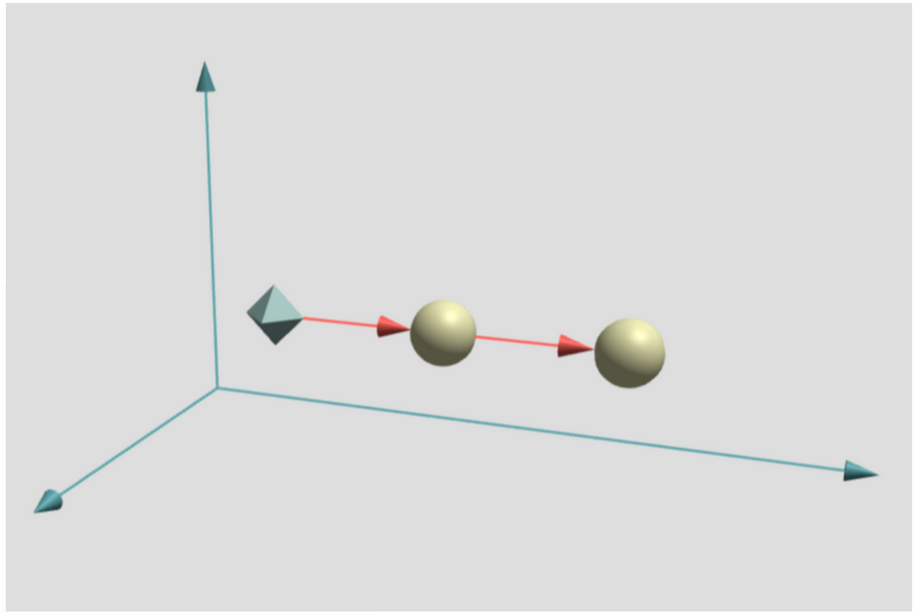


рис 2. Блок-схема способа связи двух частей системы

Требуемый формат входных данных

Для работы системы требуется стандартизированное представление графа алгоритма. В данном случае исходным форматом данных для системы является стандартная JSON структура, содержащая информацию о координатах, типе вершин и информацию об их связанности.



```
{
    "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 3. JSON структура, пример исходного формата данных и соответствующая визуализация

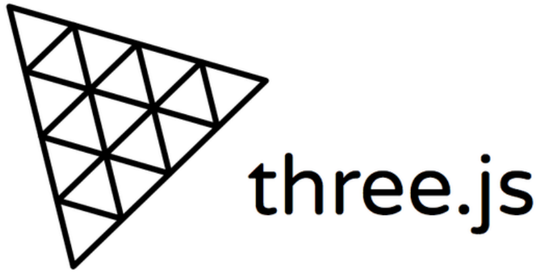
Разбиение системы на смысловые части и выбор архитектуры приложения

Разделяя проект на смысловые составляющие, можно выделить несколько основных частей:

- Работа с данными, считывание и создание внутренней структуры графа.
- Работа со структурой графа, основываясь параметрах вида, задаваемых пользователем.
- Создание 3D моделей для каждого объекта в структуре графа.
- Создание 3D сцены, содержащей модель графа.
- Работа с пользователем: обеспечение связи панели управления с моделью графа, сценой и параметрам и вида.
- При изменении параметров вида данные в модели должны локально изменяться и последовательно обновлять структуру графа и 3D модели, представляющие этот граф на сцене.

Описанные структурные и функциональные части являются достаточными критериями для выбора архитектуры Model-View-Controller - популярным стандартизированным решением для задач такого типа. Она состоит из модели (Model) со структурой графа, части отображения (View) с набором методов преобразования данных в 3D модели и контроллера (Controller), содержащего в себе методы изменения модели и обновления визуализации через пользовательский интерфейс.

Используемый стек технологий



- Язык программирования JavaScript.
- Библиотека Three.js. Используется для создания и отображения анимированной компьютерной 3D графики. Данная библиотека была выбрана за счет своей легковесности и кроссбраузерности. Вместе со всеми подключенными модулями код библиотеки умещается в 1 мегабайт.
- Модуль dat.gui.js. Обеспечивает гибкую настройку пользовательского интерфейса.
- Модуль OrbitControls.js. Обеспечивает возможность удобного контроля над сценой.

Реализованные технологии оптимизации

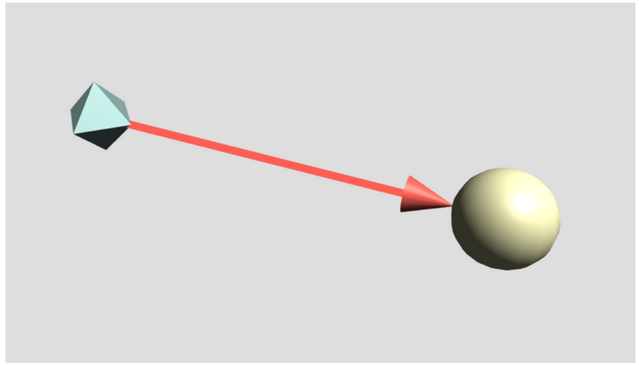


рис 4. Пример визуализации 2D модели дуги в 3D визуализации

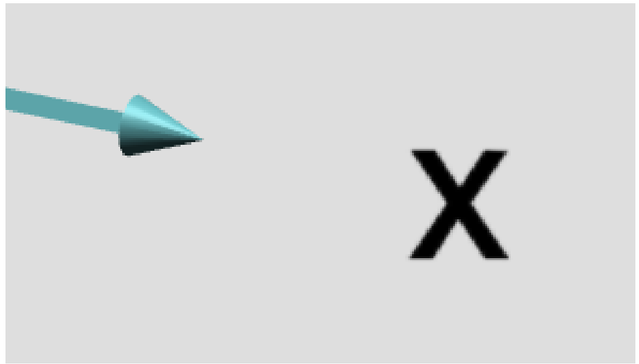


рис 5. Пример оптимизированной визуализации текста

Оптимизация моделей дуг.

Требуется использовать дуги с небольшой толщиной, однако рендеринг 3D моделей для этого слишком трудоемкая задача. Был разработан алгоритм, строящий 2D объект дуги по каноническому уравнению кривой, что сокращает нагрузку на графическое ядро более чем в 2 раза.

Оптимизация визуализации текста.

Отображение текста было оптимизировано за счет наложения текстур с изображение текста на плоскость, находящуюся на месте требуемой подписи.

Примеры 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" }
        ]
}
```

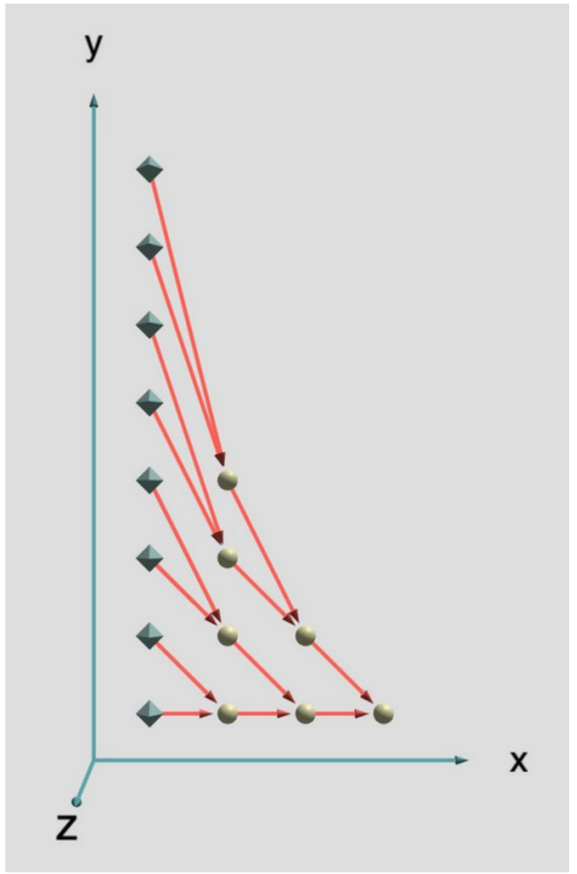



рис 6. Цепочка преобразований JSON структуры
в 3D визуализацию (алгоритм нахождения суммы элементов массива сдваиванием)

Внешний вид системы и web приложения

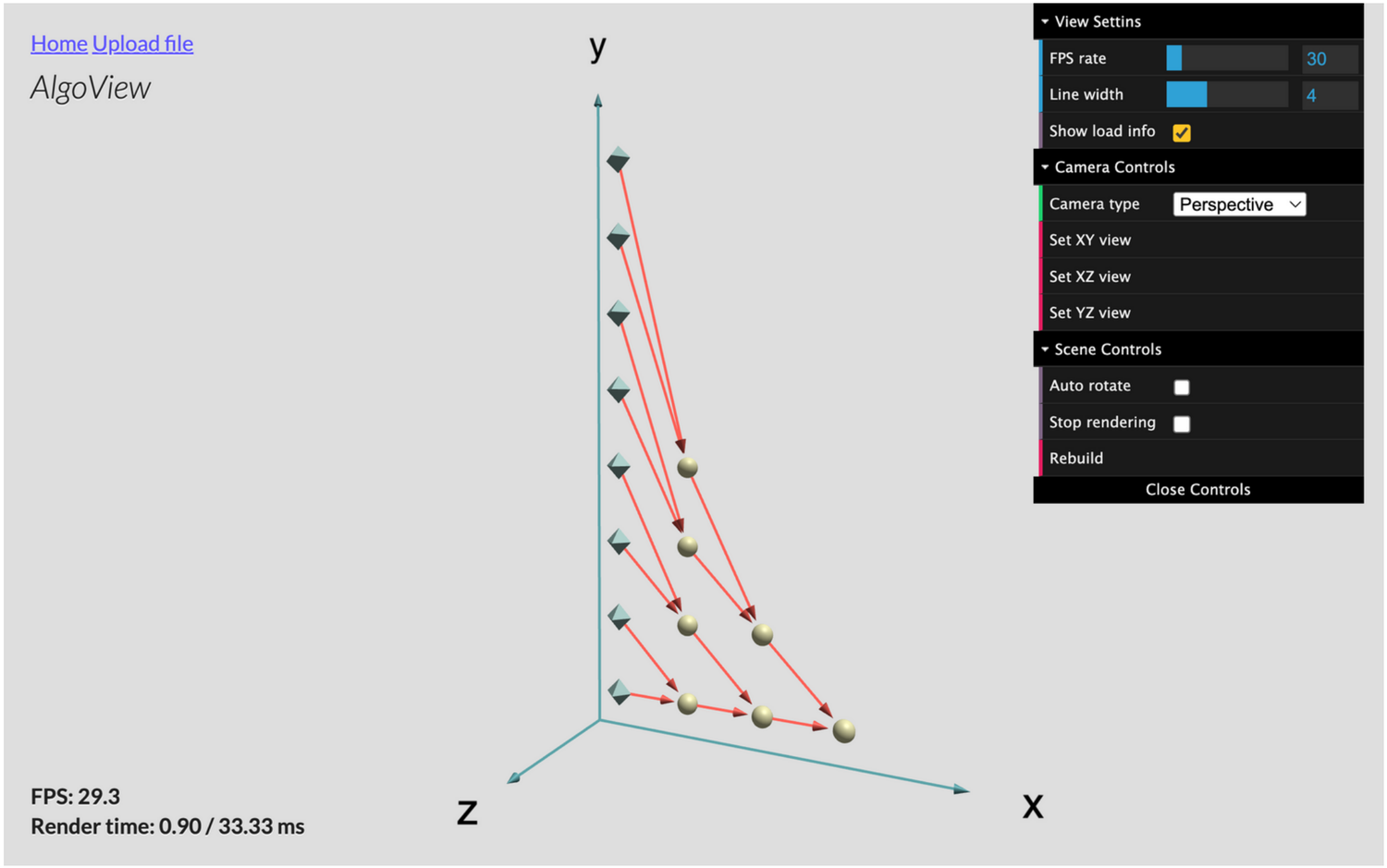


рис 7. Пример web 3D-визуализации (алгоритм нахождения суммы элементов массива сдваиванием)

Внешний вид системы и web приложения

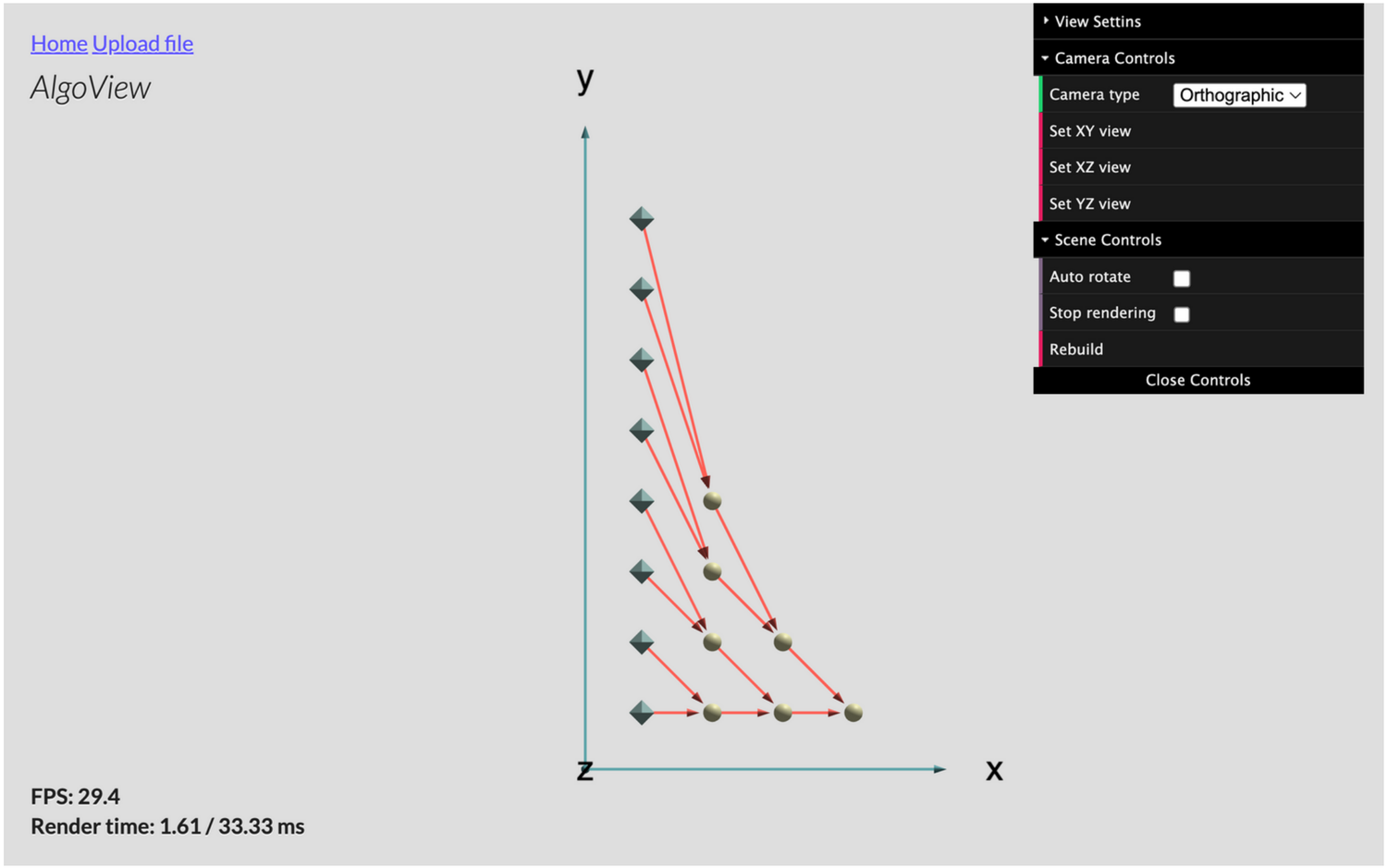


рис 8. Пример проекции на плоскость XY (алгоритм нахождения суммы элементов массива сдваиванием)

Результаты работы

- Реализовано программное обеспечение, преобразующее промежуточные данные в формате JSON во внутреннее представление для использования с библиотеками 3D визуализации.
- Разработана система интерактивной 3D визуализации информационных графов алгоритмов, предоставляющая возможности удобного визуального анализа графа алгоритмов.
- Создано веб приложение для представления системы в готовом виде, в котором можно загружать XML файлы и получать интерактивную 3D визуализацию.
- Разработаны и реализованы методы оптимизации для уменьшения нагрузки на графическое ядро браузера.

Предварительные результаты работы были представлены на молодёжном конкурсе научных исследований НИВЦ МГУ в декабре 2022 года.
https://rcc.msu.ru/ru/molodezhnyy-konkurs-nauchnyh-issledovaniy-nivc-2022-itogovaya-konferenciya-1-dekabrya-2022-g