Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра суперкомпьютеров и квантовой информатики

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

Аннотация

Разработка технологий 3D визуализации информационных графов алгоритмов

Скрябин Глеб Денисович

В данной выпускной квалификационной работе описывается исследование и разработка автоматизированной системы визуализации графов алгоритмов. Целью работы было изучение современных методов 3D-визуализации и создание системы, позволяющей создавать интерактивную 3D-модель графа алгоритма и предоставляющей возможности для его анализа. Система AlgoView была успешно разработана и обеспечивает визуальное представление внутренней структуры алгоритма, облегчая процесс детального анализа. Система полезна для образовательных и исследовательских целей, так как устраняет необходимость ручной визуализации и позволяет исследователям сосредоточиться на анализе алгоритма. В работе приводится подробное описание работы приложения и многоэтапного автоматизированного процесса обработки данных от входного файла до интерактивной веб визуализации графа алгоритма.

Abstract

Development of technologies for 3D visualization
of information graphs of algorithms

Gleb Skryabin

This final qualification work describes the research and development of an automated visualization system for graphs of algorithms. The aim of the work was to study modern methods of 3D visualization and create a system that allows you to create an interactive 3D model of the algorithm graph and provides opportunities for its analysis. The AlgoView system has been successfully developed and provides a visual representation of the internal structure of the algorithm, facilitating the process of detailed analysis. The system is useful for educational and research purposes as it eliminates the need for manual visualization and allows researchers to focus on algorithm analysis. The paper provides a detailed description of the application and a multi-stage automated data processing process from the input file to interactive web visualization of the algorithm graph.

Содержание

1 Введение **5**

1.1 Цель работы 5

1.2 Что такое граф алгоритма 5

1.3 Чем полезна система 5

1.4 Первичные данные 6

2 Постановка задачи **7**

2.1 Требования для достижения цели работы 7

2.2 Требуемый формат входных данных 7

2.3 Стандарт визуализации графов алгоритмов 9

2.4 Требования к 3D визуализации и возможностям анализа информационного графа 10

3 Обзор существующих решений **11**

3.1 Развитие возможностей трехмерной визуализации графов и ее применение в науке 11

3.2 Популярные методы 3D-визуализации 12

4 Исследование и построение решения задачи **15**

4.1 Выбор языка программирования 15

4.2 Выбор технологий 3D визуализации 15

4.3 Выбор архитектуры для разработки - MVC 16

4.4 Разбиение ответственности между модулями приложения 17

5 Описание практической части **19**

5.1 Программная реализация 19

5.2 Реализованные алгоритмы оптимизации 21

5.3 Реализованная функциональность системы 23

6 Заключение **26**

6.1 Результаты работы 26

Список литературы **27**

1 Введение

1.1 Цель работы

Целью данной работы является исследование современных методов 3D визуализации и представление разработки автоматизированной системы визуализации графов алгоритмов. Описываемая система, AlgoView, позволяет составить информационный граф алгоритма, создает его интерактивную 3D модель и предоставляет возможности по его анализу. Такой функционал обеспечивает активное применение системы в образовательных и исследовательских целях. [1]

1.2 Что такое граф алгоритма

Граф алгоритма — это ациклический граф, который представляет операции алгоритма и связи данных между ними. Вершины графа соответствуют основным операциям алгоритма, а дуги — информационным связям между этими операциями. Если две вершины связываются дугой, то вторая вершина использует данные, вычисленные в первой. Граф алгоритма может быть представлен визуально в произвольном виде или в многоуровневом параллельном виде, что означает что все вершины, располагаемые на одном уровне (ярусе) могут быть выполнены параллельно в рамках работы одного этапа алгоритма. Не следует путать такой граф с графом управления программы и тем более с её блок-схемой. Информационные графы алгоритмов активно используются при исследованиях скрытого параллелизма в алгоритмах, записанных на последовательных языках программирования. [2]

1.3 Чем полезна система

Автоматизированная система визуализации призвана обеспечить визуальное представление внутренней структуры алгоритма, облегчить процесс детального анализа алгоритма и освободить от необходимости вручную выполнять визуализацию алгоритма, облегчая исследователям возможность сосредоточиться на анализе алгоритма. Для достижения этих целей интерактивная 3D модель графа алгоритма содержит вспомогательную информацию, обеспечивающую максимальную наглядность принадлежности дуг отдельным вершинам и общей логической структуры алгоритма. Это позволяет

работать с системой визуализации и тем, кто не имеет опыта в области применения интересующего алгоритма. Дополнительно система устанавливает четкую взаимосвязь со схемой работы реализации данного алгоритма и с исходным кодом реализации.

1.4 Первичные данные

Система визуализации, описываемая в данной работе является одной из составных частей общего проекта реализующего интерактивную трехмерную систему визуализации графов алгоритмов AlgoView. Данные, получаемые от первой части системы, выходящей за рамки данной работы являются входными данными для системы визуализации.

2 Постановка задачи

2.1 Требования для достижения цели работы

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.

2.2 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 },
        { "id": 1, "sourceVertexId": 1, "targetVertexId": 2 }
    ]
}
```

Listing 1: JSON структура, пример исходного формата данных

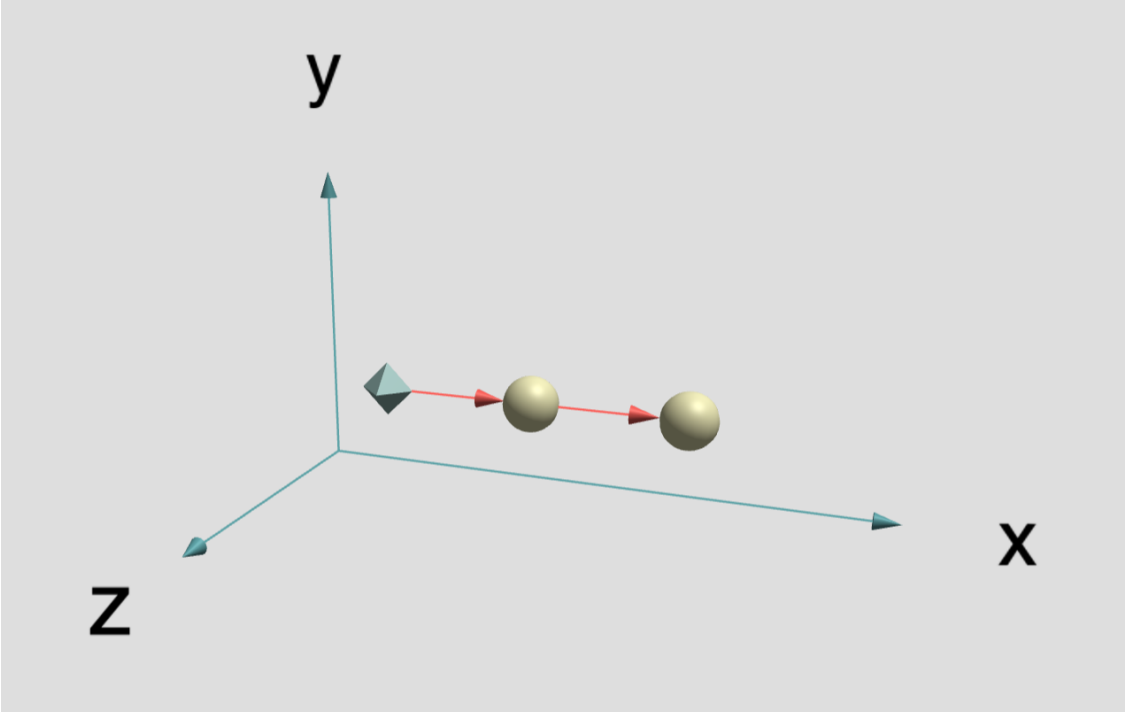

Рис. 1: Пример визуализации графа, описанного в JSON структуре на listing 1.

Набор данных, имеющийся в такой структуре выбран специально для рассматриваемой системы и в полной мере описывает граф алгоритма, который требуется визуализировать.

2.3 Стандарт визуализации графов алгоритмов

Стандарт визуализации графов алгоритмов - стандарт, впервые описанный как руководство, состоящее из набора правил, в соответствии с которыми рекомендуется изображать граф алгоритма. Существование стандарта обуславливается потребностью в построении изображений графов алгоритмов в общепринятом виде, который будет понятен вне зависимости от инструментов, используемых для изображения графа. [3]

Однако с появлением более универсального способа генерации изображений графов алгоритмов, такого как AlgoView, многие положения в стандарте требуется пересмотреть. Далее рассмотрим основные положения стандарта и некоторые изменения в нем, важные для составления требований к системе 3D визуализации. Нововведения в стандарте касаются автоматизации процесса и тем самым ослабления строгости требований к изображению графов.

2.3.1 Требования к изображению графа алгоритма

- Визуализация алгоритма должна состоять минимум из одного изображения, трехмерной модели или проекции, содержащей граф алгоритма. Визуализация не должна содержать графов, не имеющих отношения к описываемому алгоритму или графы, не соответствующие определению графа алгоритма.

- Граф алгоритма рекомендуется представлять для частного случая алгоритма и малого, фиксированного объема входных данных. Данный частный случай должен давать полное представление о структуре алгоритма.

2.3.2 Оформление структуры графа алгоритма

- Вершины графа обозначаются геометрическими 3D объектами, вид, размер и цвет которых должен совпадать для всех операций одного вида. Вершины, соответствующие входным и выходным данным, отображаются иными геометрическими фигурами. Строгих правил в выборе вида вершин нет.

- Дуги графа обозначаются линиями со стрелками на концах, соответствующих "адресату"данных.

- Для изображения структуры требуется использовать трехмерную декартову систему координат, а также набор плоскостей, параллельных одной из координатных. На каждой плоскости вводится одинаковая равномерная сетка координат. Все вершины графа алгоритма располагаются на этих плоскостях в узлах сетки.

2.4 Требования к 3D визуализации и возможностям анализа информационного графа

Визуализация графа должна соответствовать нескольким требованиям:

- 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 информации и соответствовать требованиям стандарта визуализации.

- Для удобного анализа требуется поддерживать разные настройки вида, такие как перспектива, проекция графа на плоскости, настройка осей координат. Настройки цветов и толщин линий, размер вершин. Все это влияет на удобство визуального анализа.

3 Обзор существующих решений

3.1 Развитие возможностей трехмерной визуализации графов и ее применение в науке

Визуализация графов прошла долгий путь с первых простых двумерных представлений. Были достигнуты значительные успехи в возможностях и методах, используемых для представления сложных данных в трехмерном виде. Эти разработки были вызваны необходимостью визуализации и анализа данных из различных областей, включая науку и бизнес. В настоящее время Проводится много исследований в области визуализаций графов. [4] [5] [6] [7]

Одной из областей науки, в которой требуются передовые методы представления трехмерных графов, является биоинформатика. С появлением генетических данных исследователям понадобились способы визуализации и анализа взаимодействий белков, генов и других молекулярных структур. Использование 3D-визуализации позволило ученым лучше понять взаимосвязь между этими структурами и определить потенциальные цели для разработки лекарств.

Помимо науки, бизнес и финансы также требуют передовых методов визуализации графиков. С появлением больших данных компаниям понадобились способы лучше понять сложные взаимосвязи между различными точками данных. Трехмерная визуализация графов данных позволила более интуитивно представить эти отношения, что позволяет принимать более обоснованные и взвешенные решения.

Эволюция возможностей визуализации графиков была обусловлена необходимостью лучше понимать сложные данные из различных областей. Использование 3D визуализации позволило получить более интуитивное и всестороннее понимание этих взаимосвязей, что привело к прорывам в исследованиях и принятии решений. [8]

Анализ производительности алгоритмов с использованием информационных графов не является достаточно популярной областью в науке и не содержит устоявшиеся методов по анализу таких сложных и многомерных данных. Поэтому для понимания способов реализации проекта требуется дополнительное рассмотрение популярных технологий трехмерной визуализации, доступных на сегодняшний день.

3.2 Популярные методы 3D-визуализации

Трехмерная визуализация — важный аспект взаимодействия с пользователем при разработке программного обеспечения. Рассмотрим десять самых популярных методов 3D-визуализации, существующие на сегодняшний день.

3.2.1 OpenGL

OpenGL — это широко используемый графический API, который часто используется для 3D-визуализации [9]. Это программная библиотека с открытым исходным кодом, предоставляющая набор функций для создания 2D- и 3D-графики. OpenGL — это кроссплатформенный API, что означает, что его можно использовать на различных платформах, включая Windows, Linux и macOS. Он имеет долгую историю и широко поддерживается производителями оборудования, что делает его популярным выбором для 3D-визуализации.

3.2.2 WebGL

WebGL — это API JavaScript, который позволяет разработчикам создавать трехмерную графику в веб-браузерах [10]. Он предоставляет набор низкоуровневых API для создания высокопроизводительной 3D-графики в Интернете, включая поддержку шейдеров, текстур и геометрии. WebGL поддерживается всеми основными веб-браузерами и используется в различных приложениях, включая научную визуализацию. [11] [12]

3.2.3 Three.js

Three.js — это популярная библиотека JavaScript, основанная на WebGL, которая используется для создания 3D-графики в веб-приложениях [13] . Он предоставляет набор инструментов и функций для создания интерактивных 3D-анимаций и визуализаций, включая график сцены, элементы управления освещением и камерой, а также поддержку нескольких форматов файлов. Three.js прост в использовании и имеет большое сообщество разработчиков, которые вносят свой вклад в библиотеку и делятся примерами и учебными пособиями. [14]

3.2.4 Babylon.js

Babylon.js — еще одна популярная библиотека JavaScript, которая используется для создания 3D-графики в веб-приложениях [15]. Она предоставляет набор инструментов и функций для создания высококачественной 3D-анимации и визуализации, включая поддержку физического моделирования, систем частиц и эффектов постобработки. Babylon.js так же прост в использовании и имеет большое сообщество разработчиков.

3.2.5 VTK

VTK (Visualization Toolkit) — это программная система с открытым исходным кодом, которая используется для создания 3D-графики в медицинской визуализации, научной визуализации и инженерных приложениях [16]. Он предоставляет набор инструментов и функций для создания высококачественных 3D-визуализаций, включая поддержку объемного рендеринга, изоповерхностей и интерактивного исследования. VTK широко используется в научных исследованиях и промышленности.

3.2.6 OpenSceneGraph

OpenSceneGraph — это мощный набор инструментов для трехмерной графики с открытым исходным кодом, который используется для создания высокопроизводительных трехмерных визуализаций [17]. Он предоставляет набор инструментов и функций для создания сложных 3D-сцен и анимации, включая поддержку обхода графа сцены, оптимизацию геометрии и программирование шейдеров. OpenSceneGraph так же широко используется в исследованиях и промышленности.

3.2.7 DirectX

DirectX — это набор API-интерфейсов, которые используются для разработки мультимедийных приложений на платформах Microsoft [18]. Он предоставляет набор инструментов и функций для создания 3D-графики и анимации, включая поддержку шейдеров, текстур и устройств ввода. DirectX широко используется в индустрии разработки игр.

3.2.8 Unity

Unity — популярный игровой движок, который используется для разработки 2D- и 3D-игр [19]. Он предоставляет ряд инструментов и функций для создания высококачественной графики и анимации, включая мощный редактор и язык сценариев.

3.2.9 Blender

Blender — это программное обеспечение для 3D моделирования и анимации с открытым исходным кодом, которое используется для создания высококачественной 3D-графики и анимации [20]. Он предоставляет широкий спектр инструментов и функций для создания сложных 3D-сцен и анимации, включая поддержку моделирования, текстурирования, оснастки и рендеринга.

3.2.10 Maya

Maya — это программное обеспечение для 3D-моделирования и анимации, которое широко используется в индустрии кино и видеоигр [21]. Он предоставляет набор инструментов и функций для создания высококачественной 3D-графики и анимации, включая поддержку моделирования, текстурирования, оснастки и рендеринга.

4 Исследование и построение решения задачи

4.1 Выбор языка программирования

От системы требуется чтобы ей мог воспользоваться каждый желающий, без установки дополнительного программного обеспечения и знания глубоких теоретических сведений о методах построения информационных графов алгоритмов. Для обеспечения этих требований система должна обладать следующими качествами:

- Гибкость настройки и простота запуска.
- Кроссплатформенность.
- Независимость от дополнительного ПО.

Этими качествами обладают системы, развернутые при помощи инструментов браузера с использованием распространенных стандартных библиотек, которые есть на каждом компьютере изначально. То есть для обеспечения максимальной независимости и доступности исследуемая система может быть написана на некотором серверном языке и кроссбраузерной языке, гарантированно доступным на клиентской стороне. Выбор инструментов разработки серверной части системы выходит за рамки данного исследования. Под критерии языка для реализации общедоступной клиентской части попадает только JavaScript. JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам. [22]

4.2 Выбор технологий 3D визуализации

Проанализировав рассмотренные популярные технологии 3D визуализации, базированные на языке JavaScript, была выбрана библиотека Three.js, которая используется для создания и отображения анимированной компьютерной 3D графики. Данная библиотека была выбрана за счет своей легковесности и кроссбраузерности.

Для контроля над сценой был использован модуль OrbitControls. Данный модуль является самым распространенным и поддерживаемым решением для реализации управления при помощи компьютерной мышки и клавиатуры [23]. Для создания пользовательского интерфейса и меню управления с возможностями дополнительного анализа

графов алгоритмов был использован легковесный модуль dat.GUI [24]. Размер библиотеки со всеми дополнительными модулями пользовательского интерфейса не превышает 1 мегабайта.

4.3 Выбор архитектуры для разработки - MVC

Одной из главных задач для достижения цели работы является выбор и разработка подходящей архитектуры приложения. От выбранной архитектуры зависит способ обработки данных и общения с пользователем. Выбор неподходящей структуры и методов общения модулей приложения между собой может значительно повысить нагрузку на вычислительные ресурсы системы и ядро браузера. Рассмотрим основные составные части проекта для решения этой проблемы:

- 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 требования и основные функциональные части являются достаточными критериями для выбора MVC архитектуры. [25]

4.3.1 Описание использованной MVC архитектуры

MVC расшифровывается как Model-View-Controller и представляет собой архитектурный шаблон программного обеспечения, который разделяет данные приложения, пользовательский интерфейс и логику управления на три взаимосвязанных компонента.

Модель (Model) представляет данные и бизнес-логику приложения, Представление (View) представляет уровень представления приложения, а Контроллер (Controller)

действует как посредник между Моделью и Представлением, обрабатывая пользовательский ввод и обновляя Модель и Представление по мере необходимости.

Когда пользователь взаимодействует с приложением, контроллер получает входные данные и решает, как соответствующим образом обновить модель и представление. Модель отвечает за обработку данных и самообновление, а представление отображает данные пользователю в презентабельном формате.

Эта архитектура помогает разбивать сложные приложения на более мелкие, более управляемые компоненты, упрощая разработку, тестирование и обслуживание приложения [26]. Это также упрощает сотрудничество между разработчиками, работающими над разными частями приложения.

4.3.2 Как MVC помогает решать проблемы с разделением ответственности между модулями приложения

Одной из классических проблем при разработке приложений является разделение ответственности между обработкой данных и пользовательским интерфейсом. В архитектуре MVC эта проблема решается путем разделения модели и представления на отдельные компоненты, каждый из которых несет свою ответственность.

Другой распространенной проблемой является сложность внесения изменений в приложение без нарушения других частей кода. В архитектуре MVC изменения можно вносить в один компонент, не затрагивая другие, при условии, что интерфейсы между компонентами остаются прежними.

MVC помогает повысить общую производительность приложения за счет уменьшения дублирования кода и улучшения организации кода. Это упрощает выявление и исправление ошибок, а также делает приложение более масштабируемым и адаптируемым к изменяющимся требованиям. В целом, архитектура MVC — это мощный инструмент для решения многих классических проблем разработки приложений, и сегодня она широко используется в отрасли разработки.

4.4 Разбиение ответственности между модулями приложения

Model-View-Controller. 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.

Выбрав архитектуру, требуется грамотно распределить задачи и ответственность между модулями приложения, следуя правилам конкретной архитектуры. Рассмотрим, как в данном случае распределена ответственность.

4.4.1 Содержание модуля Model

- Обработка входных данных.
- Хранение внутренней структуры графа.
- Обновление данных в структуре графа по командам пользователя.

4.4.2 Содержание модуля View

- Обеспечение набором методов преобразования данных в 3D модели.
- Заполнение сцены 3D моделями, полученными из объектов графа, хранящимися в Model.

4.4.3 Содержание модуля Controller

- Создание графического пользовательского интерфейса.
- Обеспечение связи между пользователем, его командами через меню управления и связкой Model-View.
- Управление Model и View через их встроенные методы контроля.

5 Описание практической части

5.1 Программная реализация

Рассмотрим основные программные части приложения и их фреймворки:

5.1.1 Three.js

Three.js — кроссбраузерная библиотека JavaScript, используемая для создания и отображения анимированной компьютерной 3D графики при разработке веб-приложений. Основная библиотека для использования во фронтенд части приложения системы 3D визуализации. С помощью нее был реализован следующий функционал:

- Создание сцены, в которой содержатся все 3D модели структуры графа алгоритма.
- Создание 3D моделей, основываясь на параметрах вида и данных из структуры графа.
- Создание дуг, соединяющие 3D модели.
- Контроль над сценой при помощи компьютерной мыши и клавиатуры.

5.1.2 Node.js

Node.js — программная платформа, основанная на движке V8, компилирующем JavaScript в машинный код [27]. используется для запуска бекенд части приложения. В приложении также выступает в качестве веб сервера, на котром выполняются операции с моделью графа, работа с веб страницей и пользователем, запуск скриптов связанных с другой частью основного проекта AlgoView, выходящей за рамки данной работы. Связывание потока данных и демонстрация работы системы.

5.1.3 Способ связи двух частей системы

Способ связи 2 частей системы реализован при помощи Node.js веб сервера и скриптов, организующих передачу данных через файл, содержащий JSON структуру, как файл с входными данными. На блок схеме можно увидеть, как связаны между собой две части общей системы.

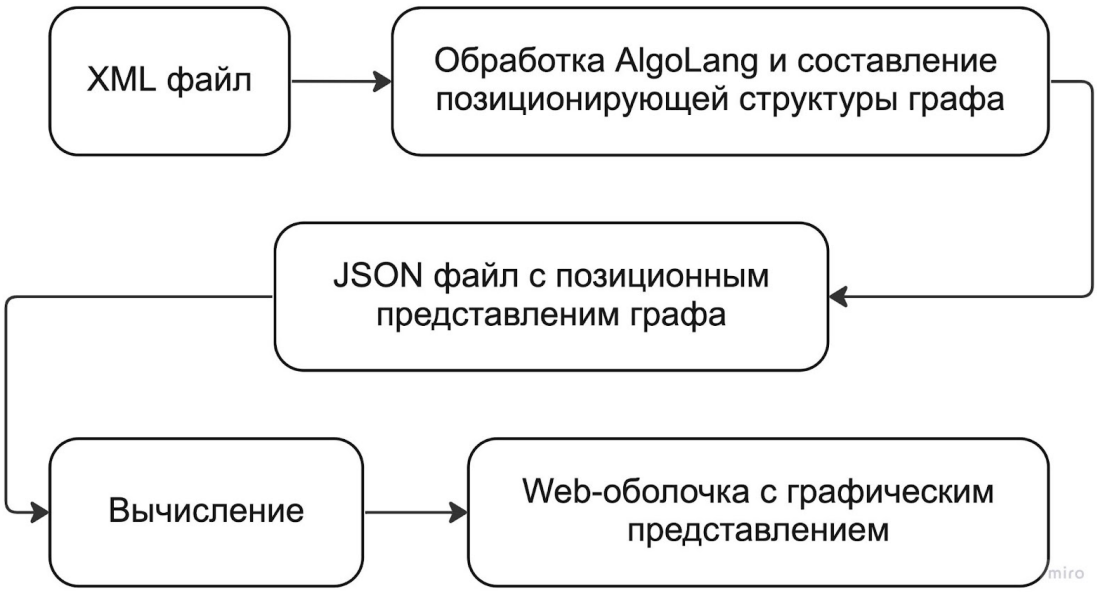

Рис. 2: Блок-схема способа связи двух частей системы.

5.1.4 Этапы работы системы

Работа приложения состоит из множества шагов, которые автоматически протекают на веб сервере. рассмотрим последовательность действий, начиная с загрузки пользователем файла с входными данными до получения интерактивной визуализации:

- Пользователь загружает на веб страницу файл с XML структурой.
- Веб сервер загружает файл и запускает программу преобразования XML файла в JSON структуру.
- Веб сервер преобразовывает JSON структуру в файл, требуемый для его чтения программой визуализации.
- Веб сервер посылает ответ пользователю с новой страницей и приложением интерактивной визуализации.
- После загрузки новой страницы и файла, содержащего информацию о структуре графа, система создает внутреннюю структуру данных для обработки и создания 3D представления графа в виде множества 3D объектов, соответствующих и визуально описывающих все структурные элементы графа.

- Последним этапом является построение интерактивной визуализации полученного набора 3D-моделей, образуя вместе с пользовательских интерфейсом многофункциональную систему анализа.

5.1.5 Docker

Docker — программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений [28]. Позволяет упаковать приложение со всем его окружением и зависимостями в контейнер, который может быть развернут на любой Linux системе.

Для быстрого развертывания приложения на удаленных серверах, вне зависимости от их конфигурации был разработан метод преобразования всего приложения в Docker образ, который способен собраться по установленным инструкциям без помощи программиста. Является важной составляющей проекта, так как приложения планируется устанавливать на нескольких серверах факультета для ознакомления и проведения учебных практикумов. Также, при помощи данного Docker образа описываемое программное обеспечение может собрать любой желающий внести вклад в развитие проекта.

5.2 Реализованные алгоритмы оптимизации

Возможной проблемой при реализации 3D визуализации может стать недостаток ресурсов для рендеринга сцены. Проблема решается разумным уменьшением максимально возможного числа кадров в секунду или уменьшением количества полигонов в объектах графа [29] [30]. Однако, только этим не всегда получается оптимизировать систему до приемлемых показателей требуемой мощности вычислительных ресурсов. В рассматриваемой системе для оптимизации были реализованы несколько дополнительных методов упрощения отображения элементов. Оптимизация моделей дуг.

Требуется использовать дуги с небольшой толщиной, однако рендеринг таких 3D моделей для этого слишком трудоемкая задача. Был адаптирован алгоритм MeshLine преобразовывающий последовательность координат в структуру простейших треугольных полигонов, представляющих собой полосу [31]. Используя каноническое уравнение кривой программа создает 2D объект дуги, что сокращает нагрузку на графическое

ядро более чем в 2 раза. Такая оценка получена эмпирическим путем сравнения результатов нагрузки на графическое ядро браузера.

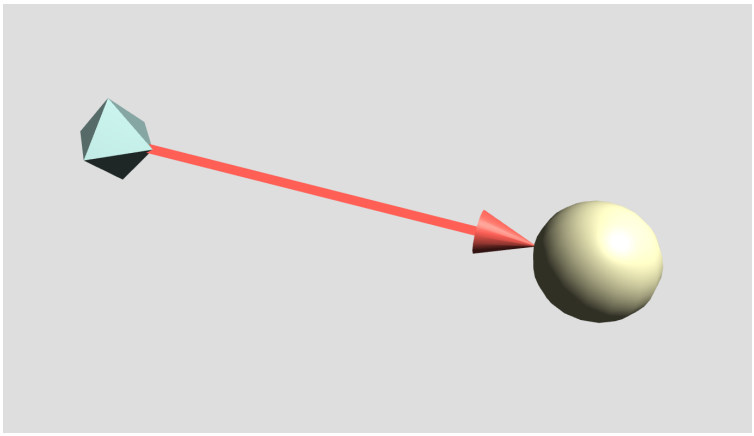

Рис. 3: Пример 2D модели дуги в 3D визуализации.

Оптимизация визуализации текста. Еще один метод оптимизации, использующийся в системе – это отображение текста при помощи наложения текстур с изображением текста на невидимый спрайт, находящийся на месте требуемой подписи. Использовались алгоритмы TextSprite и TextTexture, специально разработанные для библиотеки Three.js. [32] [33]

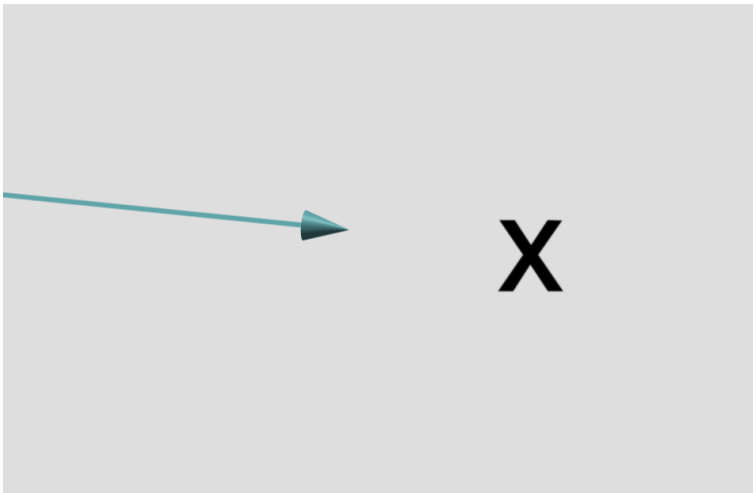

Рис. 4: Пример оптимизированной визуализации текста.

5.3 Реализованная функциональность системы

5.3.1 Пользовательское меню управления

Пользователю предоставляется возможность управлять графом и параметрами отображения через меню управления приложением, разработанное при помощи модуля dat.GUI. В меню есть следующие разделы:

- View Settings — параметры отображения. Содержит в себе настройку ограничения FPS, ширину линий и возможность отобразить показатели нагрузки на систему.
- Camera Controls — настройки камеры и проекций. В этом разделе можно выбрать какой вид камеры использовать: ортогональный или перспективу. Также раздел содержит команды показа проекций графа на плоскости XY, XZ и YZ.
- Scene Controls — дополнительные специальные настройки.

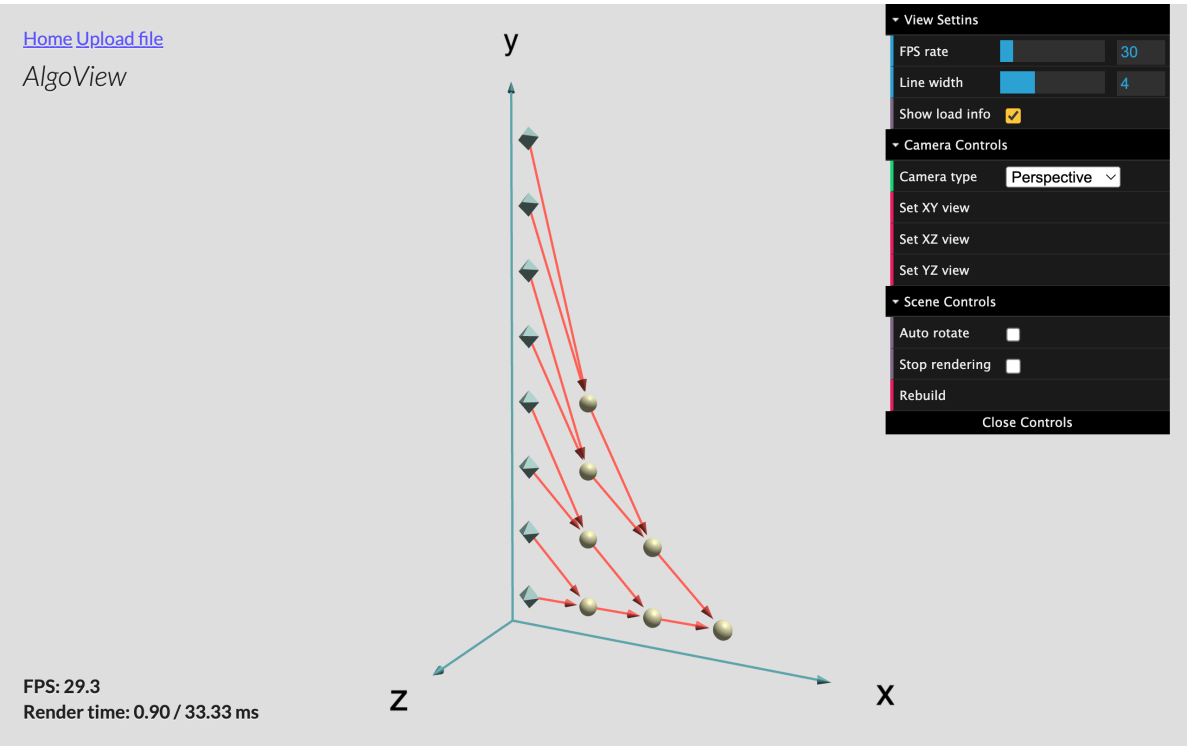

Рис. 5: Внешний вид приложения. Пример web 3D-визуализации (алгоритм нахождения суммы элементов массива сдваиванием).

5.3.2 Настройка камеры

Через пользовательское меню управления в разделе Camera Controls можно выбрать какой вид камеры использовать. Ортогональный вид представляет собой изображение графа, при котором этом все компоненты имеют одинаковые размеры, независимо от положения камеры. Перспектива — изображение графа в соответствии со зрительным восприятием объектов человеком. Ортогональная камера создается встроенным конструктором THREE.PerspectiveCamera(), а с перспективой - при помощи конструктора THREE.OrthographicCamera().

5.3.3 Проекции

Кроме настройки камеры в разделе Camera Controls можно задать отображение проекций графа на плоскости XY, XZ и YZ. Достигается посредством позиционирования камеры в направлении желаемой плоскости.

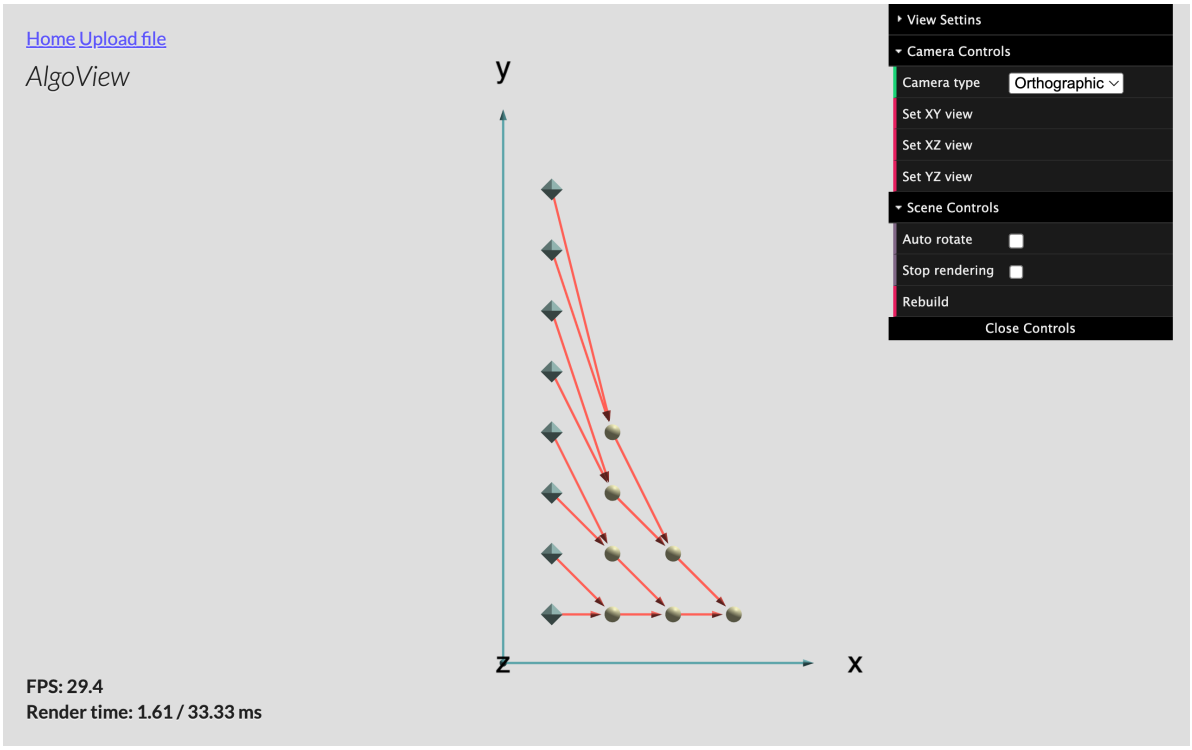

Рис. 6: Пример проекции на плоскость XY (алгоритм нахождения суммы элементов массива сдваиванием).

5.3.4 Управление компьютерной мышкой и клавиатурой

Контроль над сценой реализован при помощи модуля OrbitControls в библиотеке Three.js. Удерживая левую кнопку мыши можно вращать граф. Колесиком мыши можно приближать или отдалять изображение графа.

5.3.5 Показатели нагрузку на систему

Через пользовательское меню управления можно отобразить нагрузку на вычислительные ресурсы компьютера.

5.3.6 Страница загрузки файлов

На странице загрузки файлов можно загрузить файл в XML формате, после чего веб сервер начнет работу по преобразованию, описанную в параграфе об этапах работы системы. Внешний вид страницы показан на рисунке 7.

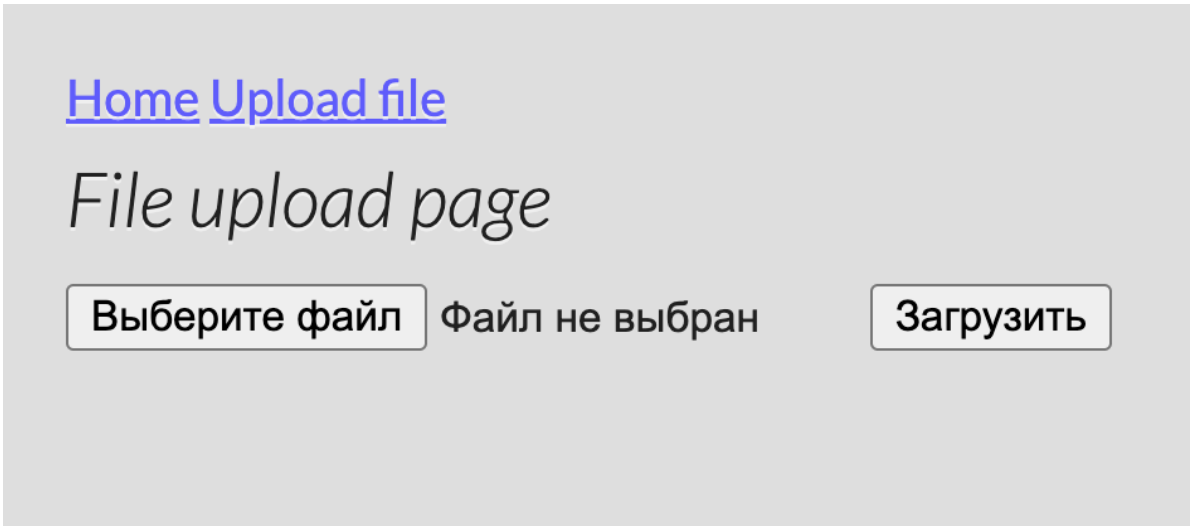

Рис. 7: Внешний вид страницы загрузки файлов.

6 Заключение

6.1 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.

Список литературы

[1] AlgoWiki. Открытая энциклопедия свойств алгоритмов. — https://algowiki-project.org.

[2] В.В. Воеводин, Вл.В. Воеводин. Параллельные вычисления. СПб.: БХВ-Петербург, 2002. 608 с. ISBN 5-94157-160-7. — https://www.studmed.ru/voevodin-vv-parallelnye-vychisleniya_42cf5ce8568.html.

[3] AlgoWiki. Стандарт визуализации графов алгоритмов. — https://algowiki-project.org.

[4] C. Chaomei, Graph Drawing Algorithms, 2006. DOI:10.1007/1-84628-579-8_3. — https://www.researchgate.net/publication/319727107_Graph_Drawing_Algorithms.

[5] S. Sim, Automatic Graph Drawing Algorithms. 1999. — https://www.researchgate.net/publication/2271614_Automatic_Graph_Drawing_Algorithms.

[6] E.R. Gansner; E. Koutsofios; S.C. North; K.-P. Vo, A technique for drawing directed graphs, 1993. DOI: 10.1109/32.221135. Publisher: IEEE. — https://ieeexplore.ieee.org/document/221135.

[7] V.N. Kasyanov, Methods and Tools for Visualization of Graphs and Graph Algorithms, 2021. INTERNATIONAL JOURNAL OF APPLIED MATHEMATICS AND INFORMATICS. DOI: 10.46300/91014.2021.15.13. — https://www.naun.org/main/UPress/ami/2021/a262013-013(2021).pdf.

[8] M.Burch, G.Wallner, H.Wetering, F.Rooks, O.Morra, Visual Analysis of Graph Algorithm Dynamics, 2021. 14th International Symposium on Visual Information Communication and Interaction. DOI: 10.1145/3481549. — https://dl.acm.org/doi/10.1145/3481549.3481550.

[9] Официальный сайт библиотеки OpenGL. — https://www.opengl.org.

[10] Официальный сайт библиотеки WebGL. — https://www.khronos.org/webgl.

[11] D.Carson, WebGL: Graphics and Animation, 2020. DOI:10.13140/RG.2.2.18538.95683. — https://www.researchgate.net/publication/344244517_Research-Paper-WebGL.

[12] M.J. Bian, H.H. Gao, J. Gao, Research and Application of Web3D Exhibition Based on WebGL and Html5, 2015. Conference: 2015 International Conference on Electrical, Automation and Mechanical Engineering. DOI:10.2991/eame-15.2015.220. — https://www.researchgate.net/publication/300618815_Research_and_Application_of_Web3D_Exhibition_Based_on_WebGL_and_Html5.

[13] Официальный сайт библиотеки Three.js. — https://threejs.org.

[14] B.Danchilla, Three.js Framework, 2012. In book: Beginning WebGL for HTML5 (pp.173-203). DOI:10.1007/978-1-4302-3997-0_7. — https://www.researchgate.net/publication/302306483_Threejs_Framework.

[15] Официальный сайт библиотеки Babylon.js. — https://www.babylonjs.com.

[16] Официальный сайт библиотеки VTK (Visualization Toolkit). — https://vtk.org.

[17] Официальный сайт библиотеки OpenSceneGraph. — http://www.openscenegraph.com.

[18] Официальный сайт API-интерфейса DirectX. — https://www.microsoft.com/ru-ru/download/details.aspx?id=35.

[19] Официальный сайт библиотеки Unity. — https://unity.com.

[20] Официальный сайт платформы Blender. — https://www.blender.org.

[21] Официальный сайт платформы Maya. — https://www.autodesk.com/products/maya/overview.

[22] David Flanagan. JavaScript Pocket Reference, 3rd Edition. — https://www.oreilly.com/library/view/javascript-pocket-reference/9781449335977.

[23] Официальная документация модуля OrbitControls. — https://threejs.org/docs/#examples/en/controls/OrbitControls.

[24] GitHub репозиторий модуля dat.GUI. — https://github.com/dataarts/dat.gui.

[25] A.Majeed, I.Rauf, MVC Architecture: A Detailed Insight to the Modern Web Applications Development, 2018. — https://crimsonpublishers.com/prsp/pdf/PRSP.000505.pdf.

[26] D.Pop, A.Samuel, Designing an MVC Model for Rapid Web Application Development, 2014. DOI:10.1016/j.proeng.2014.03.106. — https://www.researchgate.net/publication/275540078_Designing_an_MVC_Model_for_Rapid_Web_Application_Development.

[27] Официальный сайт платформы Node.js. — https://nodejs.org/en/about.

[28] Официальный сайт платформы Docker. — https://www.docker.com/why-docker.

[29] P.Gajer, M.Goodrich, S.Kobourov, A Fast Multi-Dimensional Algorithm for Drawing Large Graphs, 2000. — https://www2.cs.arizona.edu/~kobourov/grip_paper.pdf.

[30] M.Dodo, F.Andriamanampisoa, P.Torguet, J.Jessel, A new method to optimize the force-directed placement for 3D large graph drawing, 2007. Conference: Proceedings of the 5th International Conference on Computer Graphics, Virtual Reality, Visualisation and Interaction. DOI: 10.1145/1294685.1294709. — https://www.researchgate.net/publication/220804553_A_new_method_to_optimize_the_force-directed_placement_for_3D_large_graph_drawing.

[31] GitHub репозиторий алгоритма MeshLine. — https://github.com/spite/THREE.MeshLine.

[32] GitHub репозиторий алгоритма TextSprite. — https://github.com/SeregPie/THREE.TextSprite.

[33] GitHub репозиторий алгоритма TextTexture. — https://github.com/SeregPie/THREE.TextTexture.