AlgoView

Система анализа и визуализации информационных графов алгоритмов

Работа студентов 4 курса ВМК: Скрябин Г.Д., Гадиева Т.Р.
Научный руководитель: Антонов А.С.

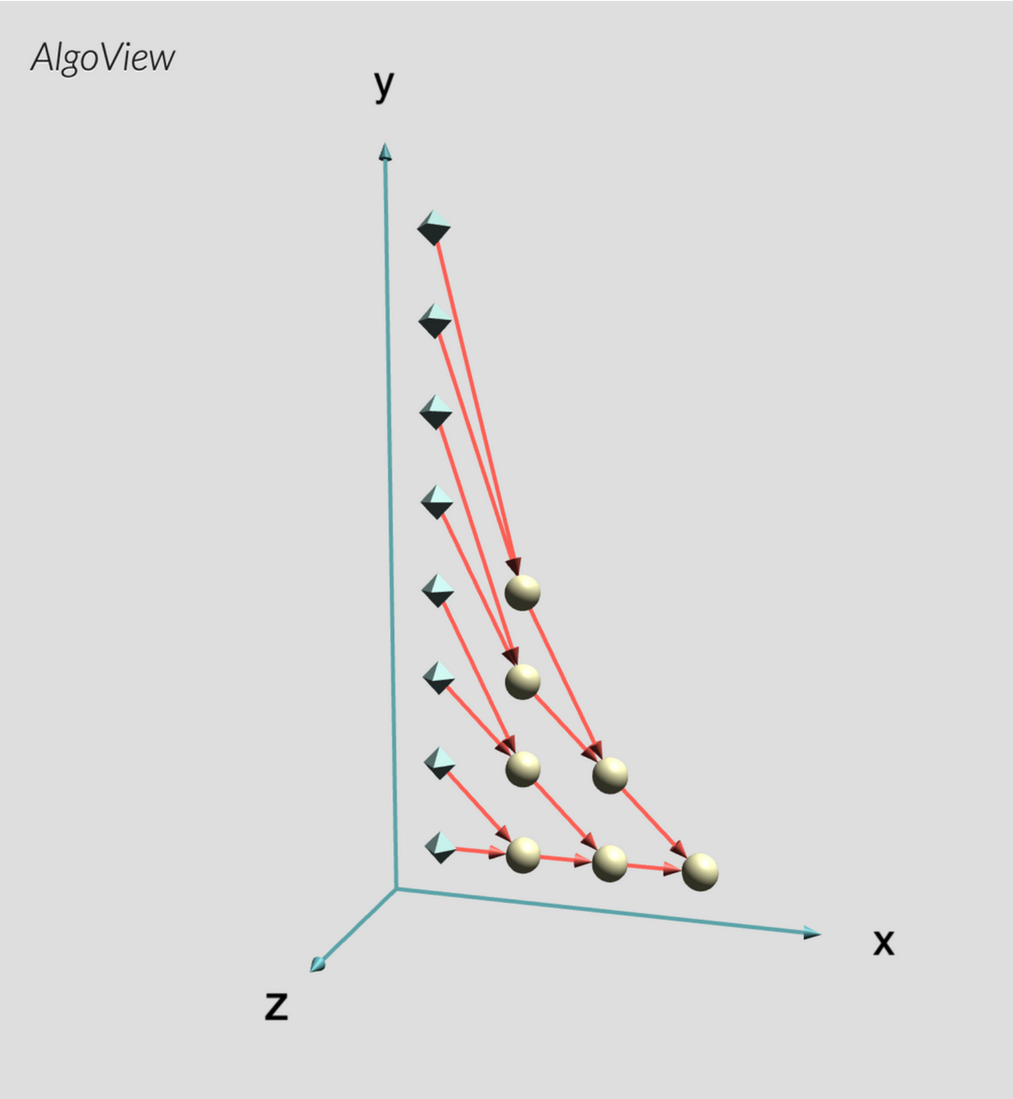


рис 1. Пример визуализации графа

Введение

- Длительность исследования: около двух семестров;
- Проект разбит на две части: логика обработки AlgoLang и визуализация с функционалом для анализа графов;
- Актуальность исследования: система используется в энциклопедии AlgoWiki и некоторыми преподавателями;
- Перспективы продолжения исследования: полное выполнение поставленного плана разработки. Использование в системах студенческих практикумов. Использование в проекте AlgoWiki.

Что такое информационный граф?

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.

Язык AlgoLang и правила написания на языке

```
<algo>
        <params>
                <param name="n" type="int" value="5"></param>
                <param name="m" type="int" value="4"></param>
        </params>

        <block dims="2">
                <arg name="i" val="1..m"></arg>
                <arg name="j" val="1..n+1"></arg>
                <vertex condition="j>1" type="2">
                        <in src="i,j-1"></in>
                </vertex>
        </block>
</algo>
```

рис 4. Описание алгоритма умножения матрицы на вектор на языке AlgoLang

<Algorithm> ::== *<algo>* <Parameters> { <Block> } *</algo>*
<Parameters> ::== *<params>* { <Parameter> } *</params>*
<Parameter> ::== *<param name = “*<Name>*” type = “*<Type> [value = “<Number>*”]></param>*
<Type> ::== *int | float*
<Block> ::== *<block* [*id = “*<Number>*“] dims = “*<Number>*”>*{Argument}{Vertex}*</block>*
<Argument> ::== *<arg name = “*<Name>*” val = “*<RegExp>..<RegExp>*”></arg>*
<Vertex> ::== *<vertex condition = “*<RegExp>*” type = “*<Number>*”>*{Source}*</vertex>*
<Source> ::== *<in* [b*src = “*<Number>*“] src = “*<RegExp>*{,*<RegExp>*}”></in>*

рис 5. Формальная структура описания информационного графа

AlgoLang и изменения в языке

```
<algo>
        <params>
                <param name="n" type="int" value="5"></param>
                <param name="m" type="int" value="4"></param>
        </params>

        <block dims="2">
                <arg name="i" val="1..m"></arg>
                <arg name="j" val="1..n+1"></arg>
                <vertex condition="j>1" type="2">
                        <in src="i,j-1"></in>
                </vertex>
        </block>
</algo>
```

*Рис. 2. Описание алгоритма умножения матрицы на вектор на языке AlgoLang (**старая** версия)*

```
<algo>
        <params>
                <param name="n" type="int" value="5"></param>
                <param name="m" type="int" value="4"></param>
        </params>

        <block dims="2">
                <args>
                        <arg name="i" val="1..m"></arg>
                        <arg name="j" val="1..n+1"></arg>
                </args>
                <vertex condition="j>1" type="2">
                        <in src="i,j-1"></in>
                </vertex>
        </block>
</algo>
```

*Рис. 3. Описание алгоритма умножения матрицы на вектор на языке AlgoLang (**новая** версия)*

Реализованная часть и дальнейший план

В первой части реализованы следующие функции:

- Разбор входного xml файла-описания графа в DOM дерево;
- Проход по дереву с целью формирования внутренних структур на основе информации о графе, приведенной на языке AlgoLang;
- Определение расположения вершин и дуг опорных многогранников;
- Запись 3D описания графа в JSON файл;

В ближайшем будущем планируется реализовать следующие возможности:

- Внесение в информацию о вершинах принадлежность к ярусу ярусно-параллельной формы;
- Обработка более, чем трехмерной мерных опорных многогранников путем получения их трехмерной проекции.

Обработка языка AlgoLang

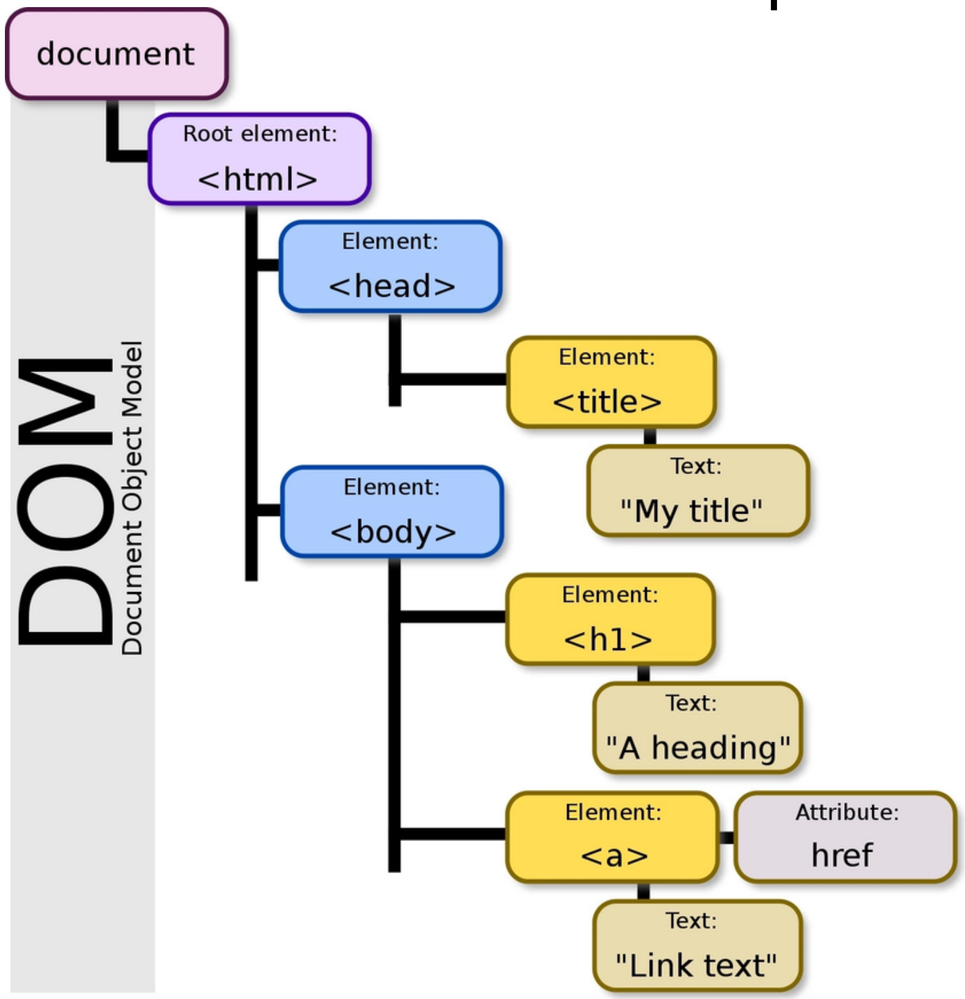


рис 6. Пример DOM дерева

Структура данных

- DOM дерево - это представление xml файла в виде дерева тегов.

Используемые библиотеки

- RapidXml: разбор XML файлов;
- ExprTK: подсчет математических выражений.

Алгоритм построения графа

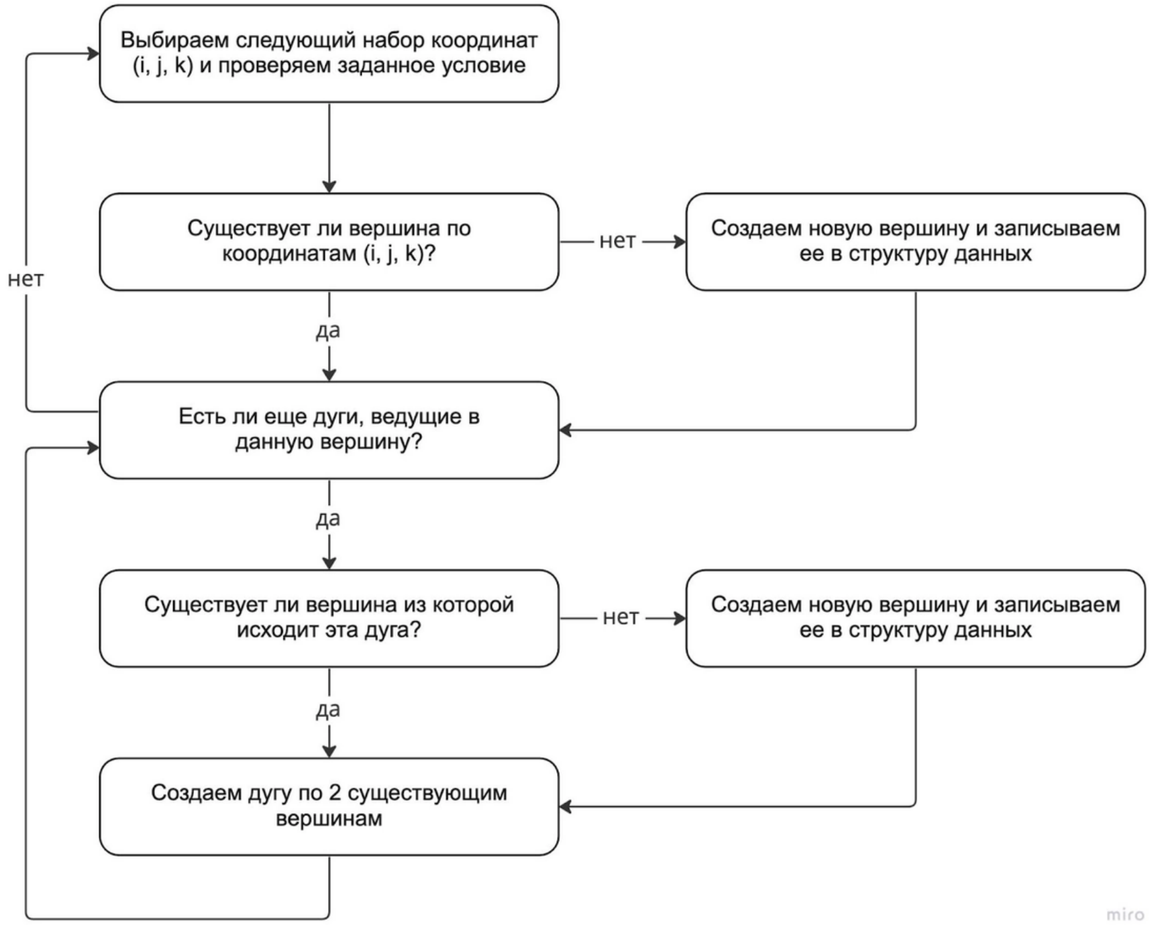


рис 7. Блок-схема алгоритм создания вершин и дуг графа

Способ связи двух частей системы

```
{ // пример json файла
  "vertices": [
    { "id": 0,
      "coordinates": [10, 10, 10],
      "type": "1" },
    { "id": 1,
      "coordinates": [10, 20, 10],
      "type": "0" },
    { "id": 2,
      "coordinates": [20, 10, 10],
      "type": "0" }
  ],
  "edges": [
    { "id": 0,
      "sourceVertexId": 0,
      "targetVertexId": 1 },
    { "id": 1,
      "sourceVertexId": 0,
      "targetVertexId": 2 }
  ]
}
```

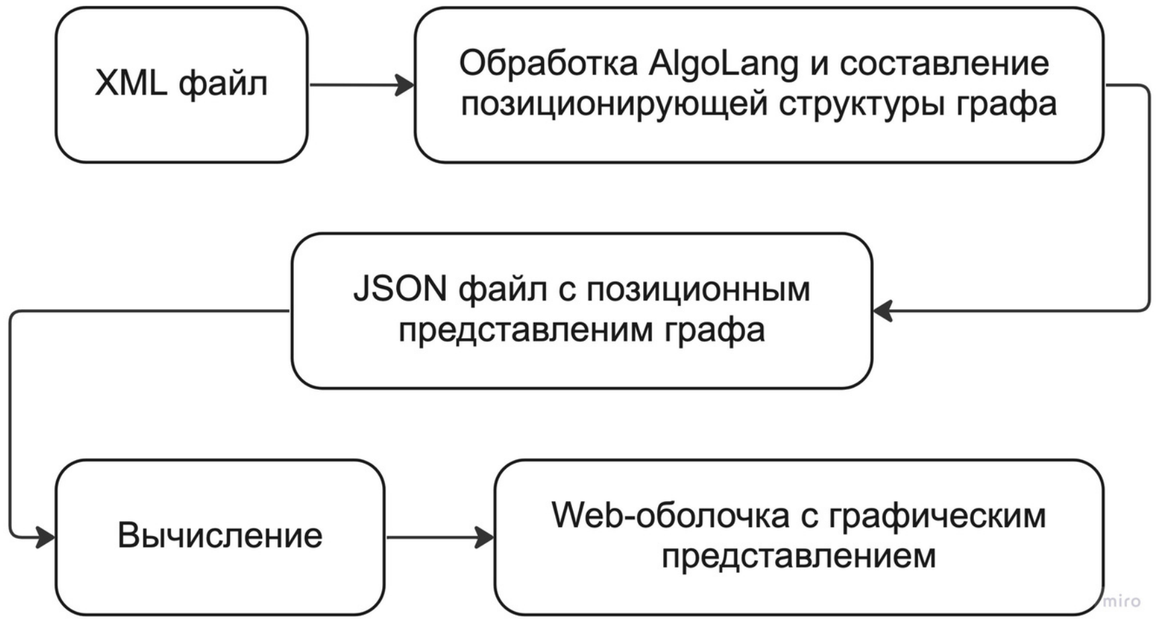


рис 8. Блок-схема алгоритм связи двух частей системы

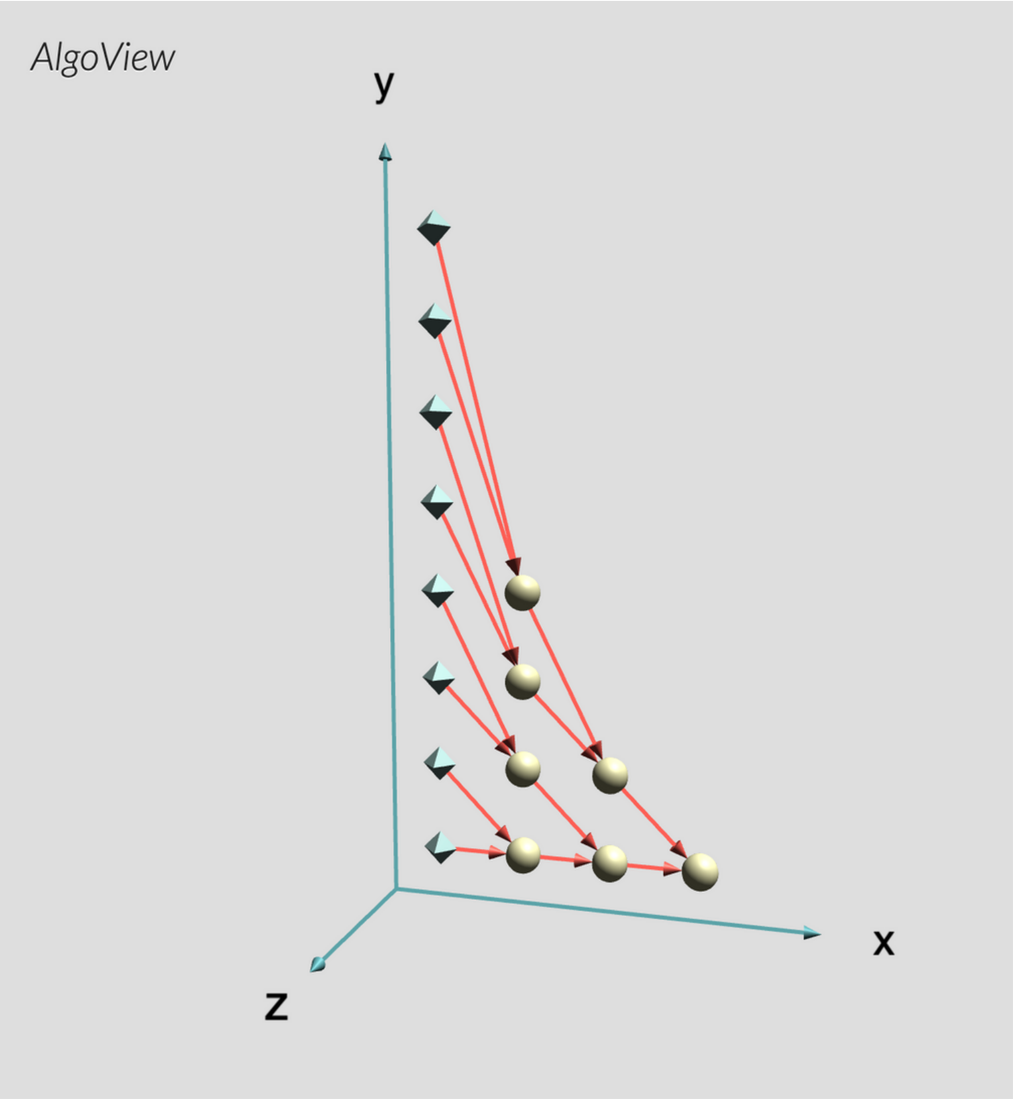
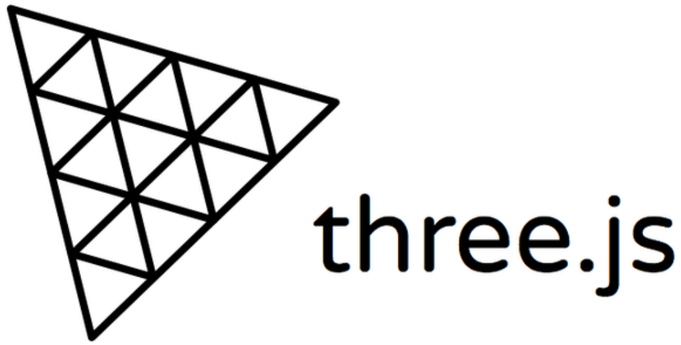


рис 9. Пример визуализации графа

Требования к визуализации и возможностям анализа

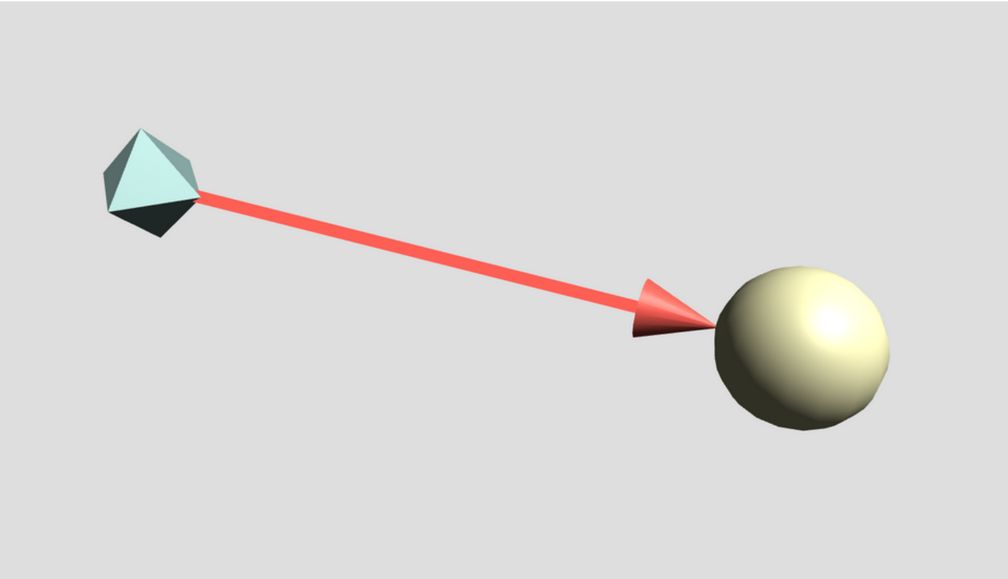
- Вершины имеют задаваемый тип;
- Для удобного анализа требуется поддерживать разные настройки вида;
- Возможность изгиба дуг для корректного отображения без коллизий и наложений;
- Возможность просмотра слоев в ярусно параллельной форме.

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

Итоги

Нами разработана система визуализации и анализа информационных графов алгоритмов позволяющая составить такой граф, его интерактивную 3D модель. На текущий момент разработаны блоки:

- Обработка входных файлов с языком AlgoLang;
- Составление внутренние структуры данных с графом;
- Функционал дающий возможность удобного анализа графа.

Сейчас ведётся разработка над внедрением ярусно параллельной формы, над алгоритмами изгиба дуг. В будущем планируется реализовать возможность обработки более чем трехмерных графов через их трехмерные проекции.