ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 145 за 2017 г.

ISSN 2071-2898 (Print)
ISSN 2071-2901 (Online)

Герман М.С., Ермаков А.В.

Архитектура и функциональные возможности системы визуализации потоков автотранспорта на элементах улично-дорожной сети

Рекомендуемая форма библиографической ссылки: Герман М.С., Ермаков А.В. Архитектура и функциональные возможности системы визуализации потоков автотранспорта на элементах улично-дорожной сети // Препринты ИПМ им. М.В.Келдыша. 2017. № 145. 21 с. doi:10.20948/prepr-2017-145
URL: http://library.keldysh.ru/preprint.asp?id=2017-145

О р д е н а Л е н и н а
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
имени М.В.Келдыша
Р о с с и й с к о й а к а д е м и и н а у к

М.С. Герман, А.В. Ермаков

Архитектура и функциональные возможности системы визуализации потоков автотранспорта на элементах улично-дорожной сети

Москва — 2017

Герман М.С., Ермаков А.В.

Архитектура и функциональные возможности системы визуализации потоков автотранспорта на элементах улично-дорожной сети.

Приводится описание архитектуры программного комплекса CAM-2D, а также его функциональные возможности, развитые и модифицированные на основании годичного опыта эксплуатации. Описываются различные способы задания начальных данных, запуска на счет и графического отображения результатов.

Ключевые слова: архитектуры программного комплекса, автотранспортные потоки, задание начальных данных, графическое отображение результатов.

Mikhail Sergeevich German, Alexey Viktorovich Ermakov

Architecture and functional capabilities of the vehicle traffic visualization system on the elements of the street-road network.

The paper provides the description of the architecture of the software complex CAM-2D and its functionalities developed and modified within a year of operating experience. Various methods of initial data specification, calculation start and visual representation of the results are given.

Key words: architecture of the software complex, vehicular traffic flows, initial data specification, graphical display of results.

Работа выполнена при поддержке Российского фонда фундаментальных исследований, Грант РФФИ № 16-01-00347-а и Грант РФФИ № 16-31-00087-мол.

Оглавление

Введение

В области микромоделей для описания динамики автотранспортных потоков заметное место занимают модели, основанные на теории клеточных автоматов, поскольку могут обеспечить более интуитивный подход к моделированию физического явления, чем моделирование, например, с использованием уравнений математической физики. Модели клеточных автоматов являются дискретными и состоят в обновлении состояния рассматриваемой системы в каждый отдельный момент времени, из которых состоит весь рассматриваемый временной промежуток [1]. Моделирование такого типа требует набора определенных правил обновления, которые принимают во внимание особенности исследуемого явления, но в достаточно простой форме.

В противоположность относительной простоте вычислительного алгоритма, в ситуации с использованием клеточных автоматов возникают трудности визуализации и интерпретации полученных результатов. Самым простым является получение интегральных величин. Например, в случае исследования пропускной способности перекрестка естественным является понимать под пропускной способностью количество автомобилей, которые пересекли перекресток за некоторое контрольное время. Но интегральные характеристики не дают возможности проследить эволюцию транспортных потоков, что при исследовании процессов на транспортных развязках представляет иногда наибольший интерес. Современные графические методы предоставляют широкие возможности для наглядного изображения на экране движения автомобилей по элементам улично-дорожной сети (УДС) [2,3].

Данная работа посвящена описанию архитектуры программного комплекса, а также показаны инструменты графического интерфейса, разработанные для удобства задания начальных данных моделируемого фрагмента транспортной сети и графического представления результатов.

1. Архитектура программного комплекса

Транспортная сеть строится из элементов (перекрестков, развилок, поворотов и т.д.). Каждый автомобиль транспортного потока характеризуется своим положением на трассе, а также некоторыми другими характеристиками (тип, цель и т.д.).

Результаты, полученные в процессе моделирования потоков автотранспорта на элементах улично-дорожной сети, представляют набор текстовых данных на каждом шаге для всех элементов транспортной сети. Такой результат является сложным для восприятия. Максимально удобным для анализа является графическое представление полученных результатов моделирования. Одним из наиболее привычных и широко используемых инструментов для графической визуализации является веб-интерфейс, что

позволяет не привязываться к аппаратной платформе и операционной системе, на которой были сделаны вычисления.

Тот же веб-интерфейс может быть использован и на этапе задания начальных данных для задачи вычислительного моделирования, а также для управления процессом вычисления.

Архитектура программного комплекса системы визуализации потоков автотранспорта строится на трех базовых элементах (рис. 1):

- вычислительный сервер;
- сервер хранения;
- браузер.

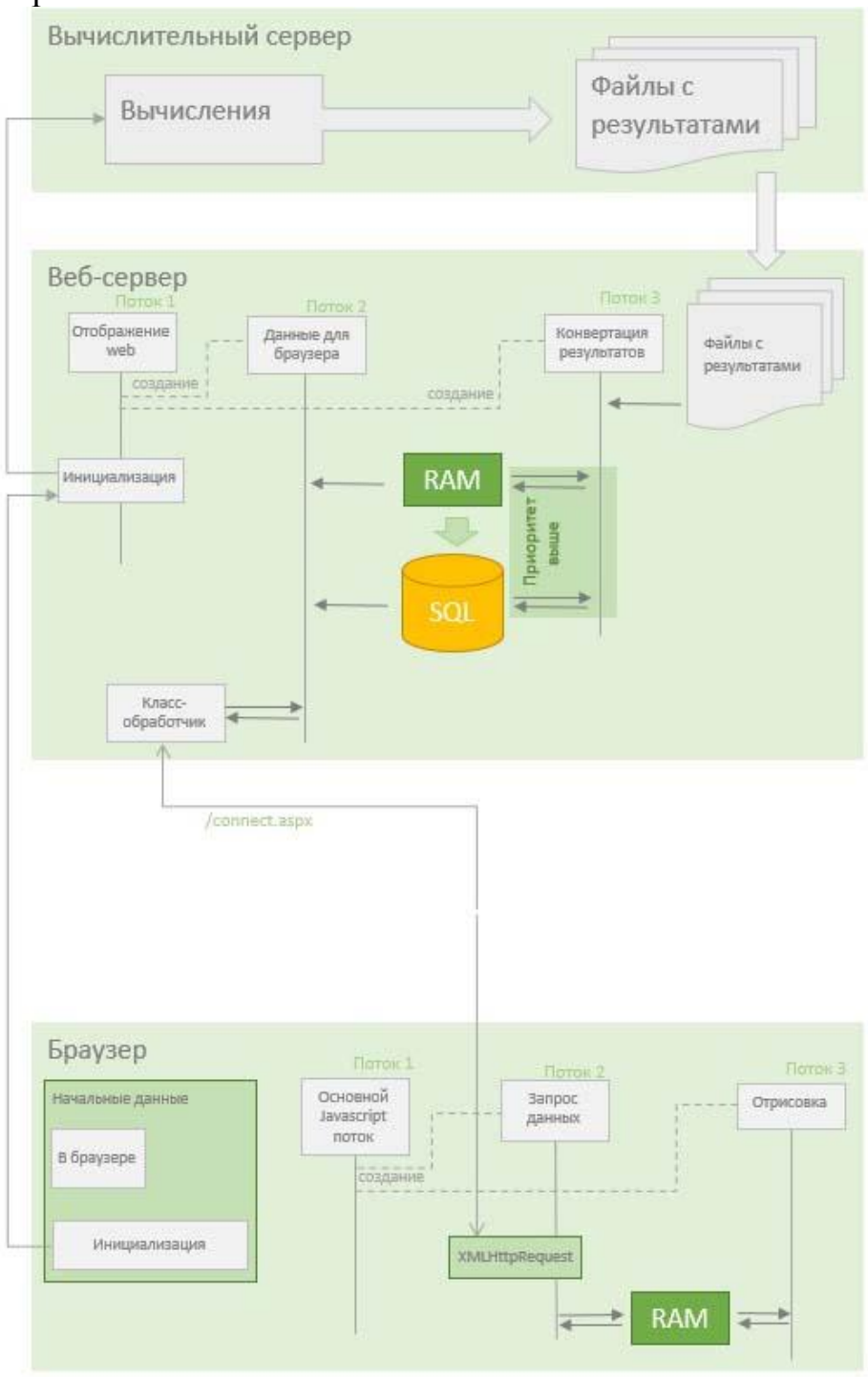

Рис. 1. Общая архитектура программного комплекса.

Чтобы вычислительный сервер начал производить расчёты, ему необходимо передать начальные данные. В рамках данной задачи было принято решение данные передавать с помощью сайта, что позволяет производить запуск вычислений удалённо.

2. Задание начальных данных

Для задания начальных данных задачи вычислительного моделирования разработан *конструктор* модели транспортной сети. Основные задачи конструктора:

1. Построение модели транспортной сети из предопределенных элементов;
2. Установка светофоров и задание их параметров;
3. Задание особенностей транспортного потока.

Моделируемая транспортная сеть может состоять из элементов 12 типов (рис. 2). Каждый элемент моделируется квадратом. Все квадраты имеют одинаковый размер в условных точках 31x31, что соответствует реальному размеру элемента транспортной сети 232,5x232,5 метров.

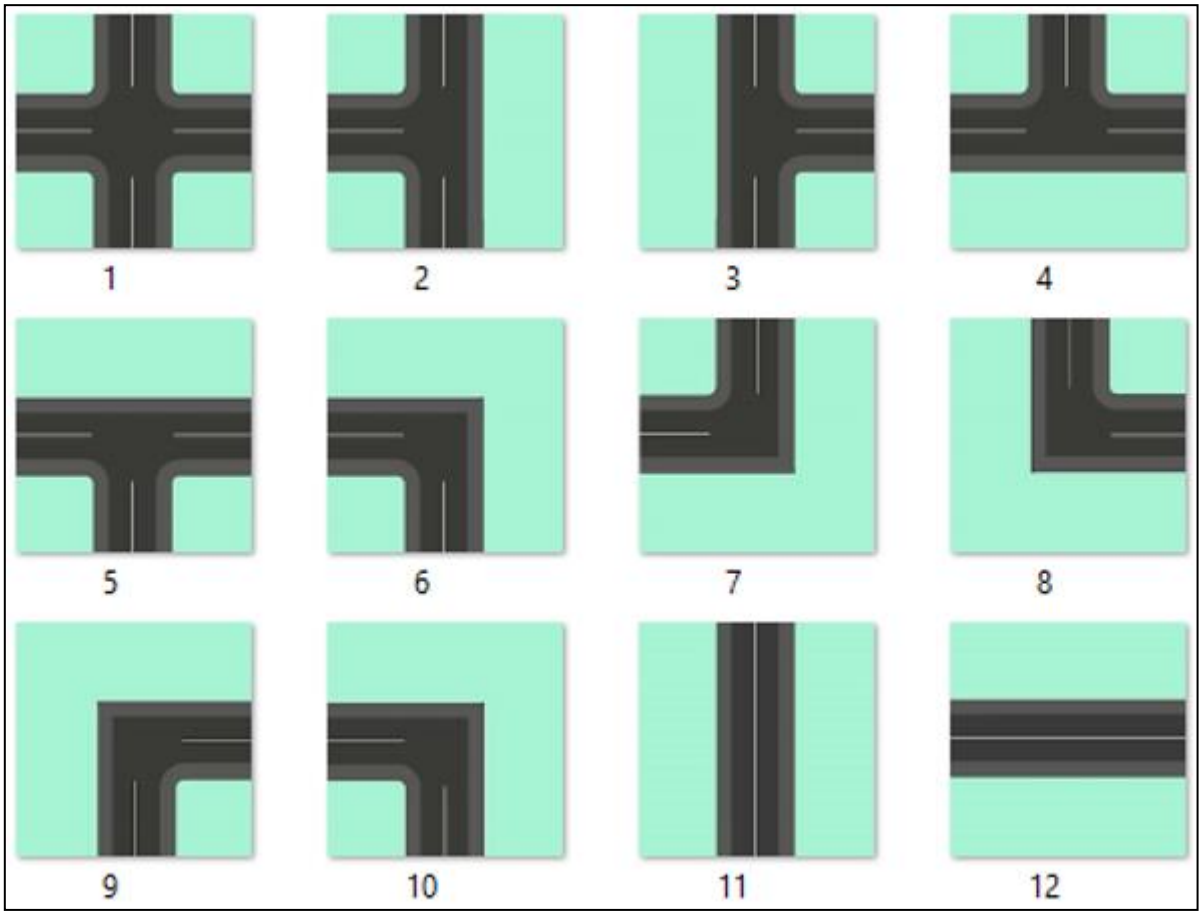

Рис. 2. Типы элементов транспортной сети

- Перекресток.
- Т-образный перекресток. Для упрощения модели используются четыре типа с заранее заданной ориентацией.
- Поворот. Для упрощения модели используются четыре типа с заранее заданной ориентацией.
- Прямой элемент. Для упрощения модели используются два типа с заранее заданной ориентацией.

При разработке конструктора транспортной сети предполагалось, что моделируемые сети могут иметь разное количество элементов.

Таким образом, для задания транспортной сети пользователю выделяется некоторое поле, размер которого задаётся пользователем в пикселах по высоте и ширине при конфигурировании конструктора (рис. 3). Размер поля не является параметром, необходимым для вычислений, он нужен только для удобства отображения транспортной сети. Количество элементов транспортной сети по вертикали и горизонтали задаётся целочисленным значением.

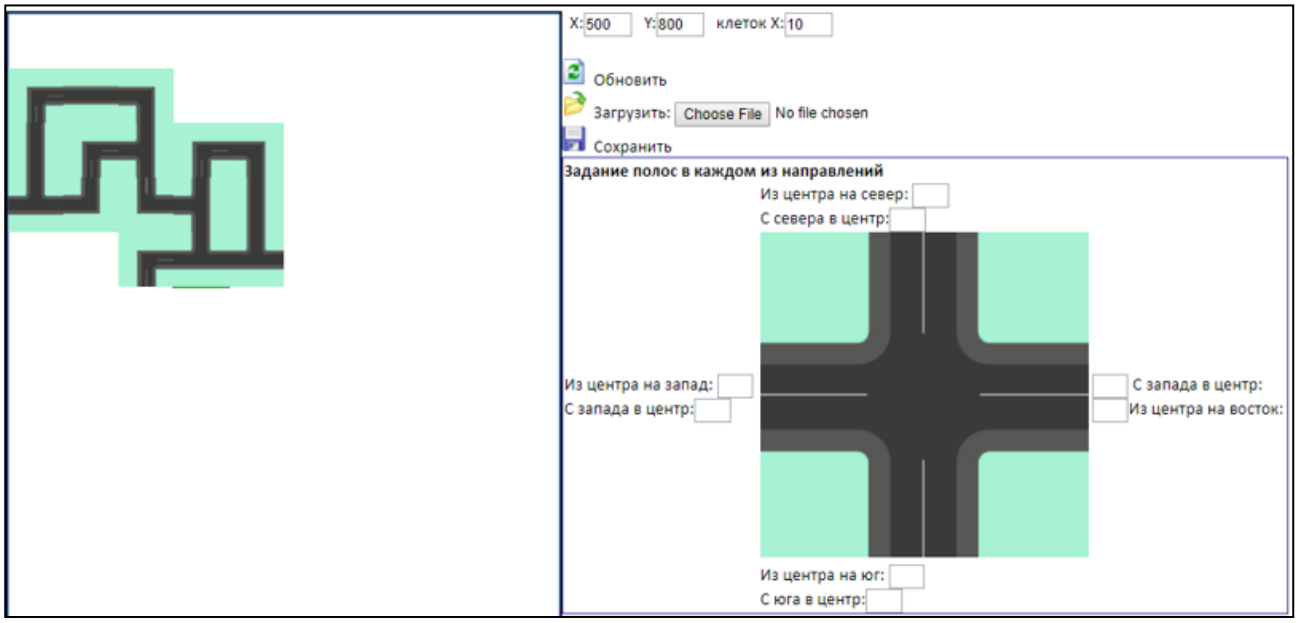

Рис. 3. Конструктор модели транспортной сети

Задав поле нужного размера, пользователь, управляя курсором с помощью клавиатуры или мыши, в любой клетке экранного поля выбирает тип элемента транспортной сети, либо может оставить клетку пустой. Для каждого заданного элемента указывается число полос в каждом направлении движения транспорта (для удобства ориентации используются направления север, юг, запад, восток).

Построенную конфигурацию транспортной сети можно сохранить в файл. Для последующих модификаций модель может быть загружена из ранее сохраненного файла.

В текущей версии системы в файле сохраняется размер рабочего поля в пикселах, число элементов по горизонтали (расположенные «в ряд» элементы называются «строкой элементов»), а также все выбранные типы элементов транспортной сети. На рис.4 показан формат файла начальных данных.

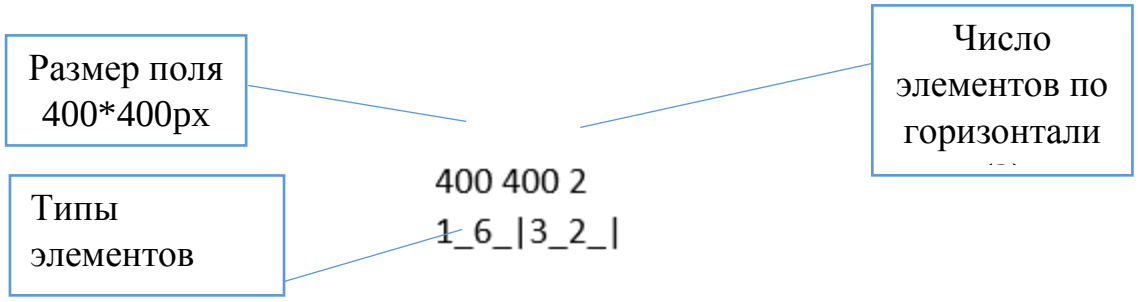

Рис. 4. Пример файла с сохранённым полем и выбранными перекрёстками

В первой строке файла начальных данных задается размер поля конфигуратора (разделённые пробелом: число пикселей в ширину, число пикселей в высоту, количество элементов по горизонтали).

Вторая строка файла содержит в себе данные типов всех элементов. Каждый элемент транспортной сети имеет целочисленное значение. Элементы разделяются символом «_». Каждая строка элементов заканчивается символом «|». При таком описании моделируемой транспортной сети учитывается порядок: сверху – вниз, слева – направо.

3. Запуск на счет

Запуск расчета, а также управление процессом вычисления можно осуществлять как из командной строки операционной системы, так и из самого программного комплекса. Процесс вычислений запускается с помощью создаваемого *.bat файла, внутри которого указываются параметры для запуска *.exe файла (рис. 5):

0_30_30_30_30_1_2_3_1_2_0_1_1_1_0_2_4_0.2_4_1000

|

- 1-й параметр: параметр запуска режима отладки (debug mode). 0 – выключен, иначе – включён.
- С 2-го по 5-й параметры: длительность фаз светофора (**30_30_30_30**).
- С 6-го по 8-й параметры: количество машин в минуту, которые начинают движение с юга и перемещаются в сторону оставшихся направлений соответственно (**1_2_3**).

- С 9-го по 11-й параметры: количество машин в минуту, которые начинают движение с севера и перемещаются в сторону оставшихся направлений соответственно (**1_2_0**).
- С 12-го по 14-й параметры: количество машин в минуту, которые начинают движение с востока и перемещаются в сторону оставшихся направлений соответственно (**1_1_1**).
- С 15-го по 17-й параметры: количество машин в минуту, которые начинают движение с запада и перемещаются в сторону оставшихся направлений соответственно (**0_2_4**).
- 18-й параметр: вероятность Нагеля-Шрекенберга (**0.2**).
- 19-й параметр: максимальная скорость автомобилей, измеряется в клетоках/шаг. 1 клетка/шаг соответствует значению в 27 км/ч (**4**).
- 20-й параметр: максимальное время выполнения вычислений, после чего вычисления будут приостановлены (**1000**).

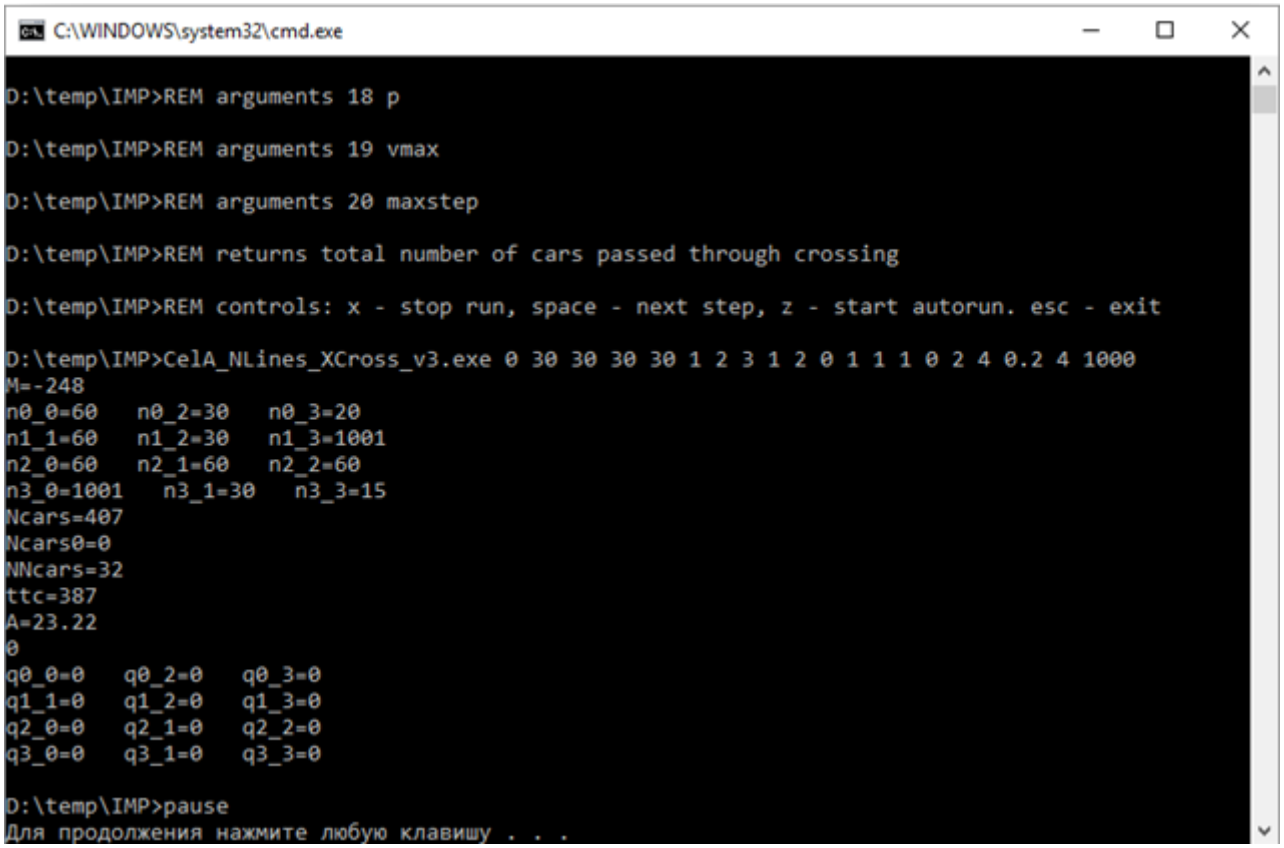

Рис. 5. Пример запуска вычислений.

4. Получение результатов

После того, как данные будут вычислены, вычислительный сервер их передаст на веб-сервер для дальнейшей визуализации. Результаты хранятся в файлах: один файл – один элемент транспортной сети и все его состояния в процессе вычислений (светофоры, координаты транспорта и т.д.). Для хранения предусмотрен текстовый формат представления результатов. В системе используется условное время – шаги состояний. Каждый шаг определяет точное местоположение транспорта и состояние светофоров. Каждому шагу (состоянию) соответствует одна строка. Такая система передачи данных позволяет контролировать скорость отображения шагов вычислительного процесса, останавливать и запускать реалистичное отображение или пошаговое, используемое на этапах отладки или подробного анализа.

В файлах результатов координаты положения объектов (автомобилей) задаются не от верхнего угла, а параметрами: номер дороги, полоса дороги и номер ячейки (рис. 7). Номер дороги должен быть от «0» до «3» и зависит от направления движения автомобиля: «0» – если машина едет с севера на юг, «1» - с севера на юг, «2» - с запада на восток, «3» - с востока на запад. Полоса дороги обозначается целочисленным значением от «1» самого правого ряду по ходу движения до «N».

В данной версии системы все расчёты ведутся только для правостороннего движения. Номер ячейки на полосе – это расстояние от начала полосы (по ходу движения) до точки положения автомобиля в условных единицах. В зависимости от размера сетки ячейка может быть от 0 до R-1, где R - размер элемента транспортной сети.

Например, для элемента транспортной сети размером 15x15 клеток с перекрёстком дорог 5x5 клеток, где все дороги двуполостные, рассмотрим машину, подъехавшую к светофору на перекрёстке с запада на восток в правой полосе (рис. 7). Получается, автомобиль имеет координаты (3, 1, 4), где 3 – номер дороги, 1 – номер полосы, 4 – смещение.

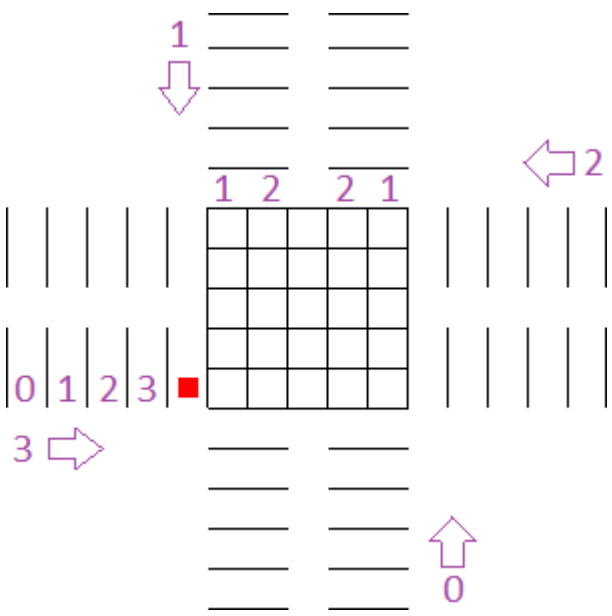

Рис. 7. Нумерация дорог (возле стрелок), полос на дорогах 1 и 2, смещение автомобиля на дороге 3.

Все файлы с результатами размещаются в одной папке. Учитывая, что число шагов и количество элементов транспортной сети может быть большое, считывание результатов всех элементов производится одновременно по одной строке из каждого файла с результатами. Учитывая, что в каждом из файлов строки упорядочены по шагам по возрастанию, то обход всех файлов можно смело начинать с 1 шага. Если для какого-то элемента очередной шаг отсутствует, то считается, что элемент не меняет своего состояния на этом шаге. Таким образом организовано упорядоченное по шагам чтение состояний перекрёстков.

Требования к наименованию файла отсутствуют, выполняется обработка всех файлов «*.txt» в заданной директории. Как уже отмечалось, файл с результатами моделирования каждого элемента представляет собой последовательность строк. Первая строка содержит данные о параметрах вычислений состояний перекрёстка. Если первая строка файла не удовлетворяет требованиям, то файл далее не читается и считается неверным.

Пример строки параметров вычислений файла результатов:

0_60_90_0|3_2_1_0|31|108|0.2|3600|10_0_0_6_0_0_0_0_0_0_0_0|0_1|3,
где:

- «**0_60_90_0**»:
  Длительность режима светофора в секундах по направлению движения 0, 1, 2, 3 (рис. 7).
- «**3_2_1_0**»:

тип светофора (рис. 6) в каждом из направлений движения транспорта от 0 до 3 (рис. 7).

- «**31**»: длина дороги, измеряется в клетках.
- «**108**»: максимальная скорость автомобилей (км/ч).
- «**0.2**»: вероятность в модели Нагеля-Шрекенберга
- «**3600**»: длительность расчета (секунд)
- «**10_0_0**»: количество машин в минуту, которые начинают движение с юга и заканчивают в оставшихся направлений соответственно.
- «**6_0_0**»: количество машин в минуту, которые начинают движение с севера и заканчивают в оставшихся направлений соответственно.
- «**0_0_0**»: количество машин в минуту, которые начинают движение с востока и заканчивают в оставшихся направлений соответственно.
- «**0_0_0**»: количество машин в минуту, которые начинают движение с запада и заканчивают в оставшихся направлений соответственно.
- «**0_1**»: позиция этого перекрёстка относительно других в данном расчёте (в поле конфигуратора). {x,y}={0,1}
- «**3**»: тип перекрёстка (рис. 2)

Если параметры первой строки отражают начальные условия запуска расчёта, то каждая последующая строка отвечает за состояние элемента транспортной сети на соответствующем шаге. Если у элемента шаг не указан или указан неправильно, перекрёсток на этом шаге не изменится.

Пример строки результатов на каждом шаге моделирования:
10_14_10_0_0:1001_0_0_17_0|1002_1_0_2_,
где:

- «**10**»: Номер шага
- **«14_10_0_0»:** тип светофора (рис. 6) в каждом из направлений движения транспорта от 0 до 3 (рис. 7).
- «**1001_0_0_17_0**», «**1002_1_0_2_1**»: два автомобиля с идентификаторами 1001 и 1002. После идентификаторов указываются их состояние:
    - текущее направление движения (рис. 7),
    - номер полосы (рис. 7),
    - положение на полосе в клетках. Положение на любом направлении считается от начального положения автомобиля у края карты перекрёстка, где значение равняется 0 клетке до конечной клетки дороги у другой стороны карты.
    - направление движения после прохождения перекрёстка.

5. Обработка полученных результатов

Полученные результаты моделирования необходимо визуализировать - отобразить в поле браузере. Формат, используемый для хранения данных,

получаемых в результате вычислений, не удобен для визуализации. Например, при повторном просмотре результатов на кране браузера нужно знать местонахождение объектов в пикселях относительного всего поля, ведь все перекрёстки будут отображаться как единая сеть. Таким образом, чтобы просмотреть результаты N раз, придётся N раз произвести преобразование координат, применяемых при вычислении в координаты для визуализации.

Чтобы избежать лишних операций преобразования координат объектов выполняются один раз и сохраняются в виде их положения в относительных координатах поля браузера {x, y}.

Для отображения в браузере используется технология canvas HTML5. Важно не забывать базовое расположение осей X и Y – точка (0, 0) находится в верхнем левом углу поля браузера (рис. 6). Но результаты в каждом файле для соответствующего элемента транспортной сети используют координаты не абсолютные (с учётом положения других перекрёстков), а относительные, как если бы визуализировались все перекрёстки как отдельные задачи. Приведем пример отрисовки двух перекрёстков (1) и (2) на общем поле браузера (рис. 6). В файлах результатов объекты (на рисунке показаны оранжевыми точками) для элементов сети (1) и (2) имеют одинаковые относительные координаты (2, 1). Но для отображения этих объектов на общем поле браузера нужно вычислить их абсолютные координаты – в данном случае это (4, 4) и (7, 4).

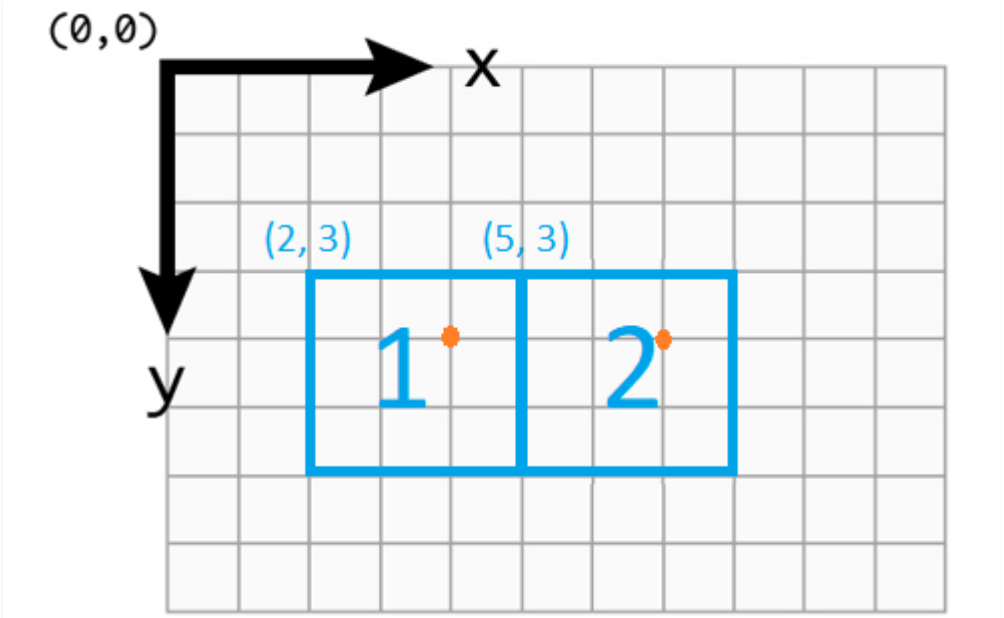

Рис. 6. Пример расположения двух элементов транспортной сети.

Поскольку мы задаём размер транспортной сети и количество элементов в ней, то и поле браузера будет иметь некий масштаб, который так же будет учитываться.

Поскольку в расчётах ширина полос в условных точках является неизменным числом, будем считать её равной W. В итоге у нас получается следующая функция преобразования координат из расчётной системы в координаты на поле отрисовки объекта:

$$x = P_x * S_{px} + \begin{cases} \frac{S_{px}}{2} + (a_p - 1/2) * W, \text{если } a_d = 0 \\ \frac{S_{px}}{2} - (a_p - 1/2) * W, \text{если } a_d = 1 \\ \frac{S - a_c}{S} * S_{px}, \text{если } a_d = 2 \\ \frac{a_c}{S} * S_{px}, \text{если } a_d = 3 \end{cases}$$

$$y = P_y * S_{px} + \begin{cases} \frac{S_{px}}{2} + (a_p - 1/2) * W, \text{если } a_d = 3 \\ \frac{S_{px}}{2} - (a_p - 1/2) * W, \text{если } a_d = 2 \\ \frac{S - a_c}{S} * S_{px}, \text{если } a_d = 0 \\ \frac{a_c}{S} * S_{px}, \text{если } a_d = 1 \end{cases} \text{, где}$$

$x -$ искомое положение объекта по оси X (в пикселях),

$y -$ искомое положение объекта по оси Y (в пикселях),

$S_{px} -$ размер перекрёстка (в пикселях),

$S -$ размер перекрёстка (в условных единицах),

$W -$ ширина полосы (в пикселях)

$A_l -$ количество полос в направлении, по которому движется объект.

$P_x -$ смещение перекрёстка по Оси X относительно самого левого перекрёстка (измеряется числом перекрёстков)

$P_y -$ смещение перекрёстка по Оси Y относительно самого верхнего перекрёстка (измеряется числом перекрёстков)

$a_d -$ номер направления дороги, по которой движется объект,

$a_p -$ номер полосы дороги, где находится объект,

$a_c -$ номер ячейки на полосе, где находится объект.

5.1. Хранение результатов

Для хранения результатов в оперативной памяти выделяется некоторый пул, граница которого задается администратором системы исходя из технических характеристик сервера. Для сервера хранения данных выбран MS

SQL. Когда объем вычисленных результатов достигает границы пула, вычисленные данные передаются на сохранение в БД (рис. 8). Сохранение происходит в новом параллельном потоке, чтобы не нагружать основной процесс, отвечающий за отрисовку результатов.

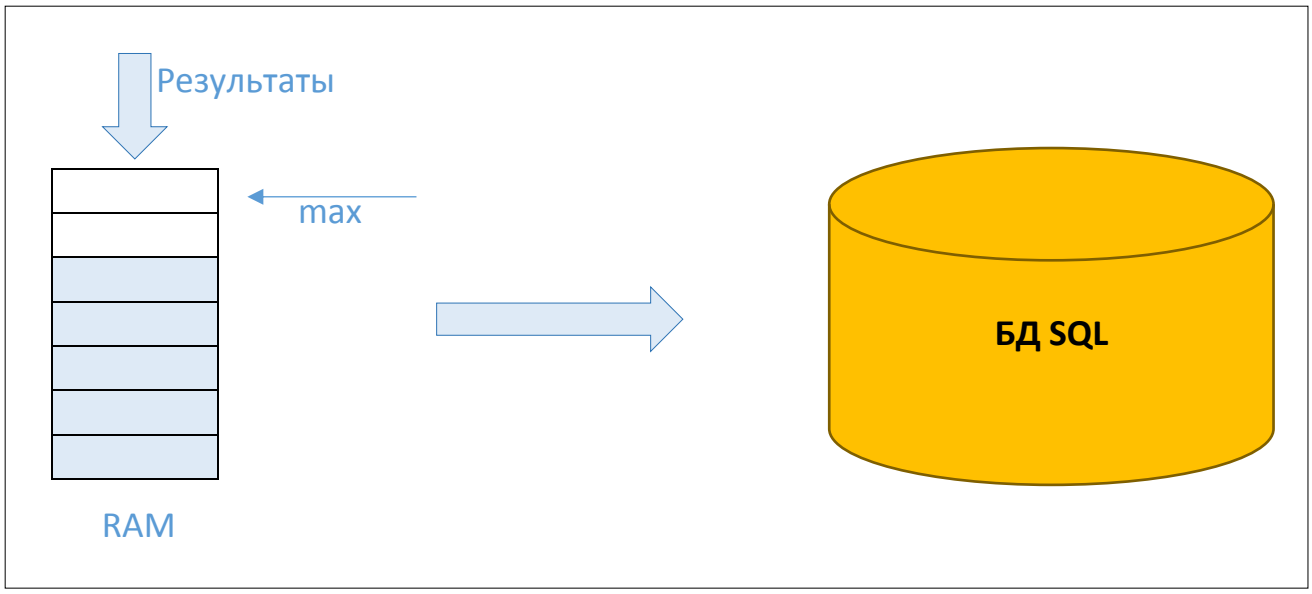

Рис. 8. Сохранение данных.

5.2. Передача результатов браузеру

Получение результатов для отрисовки осуществляется браузером на основе запросов, посылаемых серверу. Чтобы поле браузера (страница браузера) не перезагружалось и не инициализировало все объекты заново, использована технология запросов XMLHttpRequest.

Со стороны сервера так же нужно открыть возможность принимать запросы. Для этого создан класс обработчика http-данных, который может принимать любые запросы со стороны браузера на относительный адрес. Относительный адрес сопоставляется с классом-обработчиком и может реально не существовать как физический файл. В нашем случае это адрес «/connect.aspx». Чтобы сообщить системе, какой класс обработчик у нас соответствует строке запроса, необходимо в файле параметров web.config для ASP.NET указать это соответствие. В нашем случае мы добавляем связь «/connect.aspx» с классом «transport.SyncHandler» в сборке «transport». (рис. 8).

Web.config

```
1   <?xml version="1.0"?>
2   ...
6   <configuration>
7     <configSections>...</configSections>
11    <connectionStrings>...</connectionStrings>
14    ...
22    <system.web>...</system.web>
41    <system.webServer>
42      <handlers>
43        <add name="SampleHandler2" verb="*" resourceType="Unspecified" path="/connect.aspx" type="transport.SyncHandler, transport"/>
44      </handlers>
45      <modules runAllManagedModulesForAllRequests="true"/>
46    </system.webServer>
47  </configuration>
```

Рис. 9. Файл web.config

Поскольку данные передаются начиная с наименьшего шага, серверу достаточно хранить информацию о последнем переданном шаге. Количество передаваемых данных жёстко не задается и может быть изменено в зависимости от пожеланий пользователя.

Сервер передаёт сразу несколько шагов, а не по отдельности. Это сделано с целью оптимизации, ведь каждый запрос XMLHttpRequest требует время на обработку сервером. Рассмотрим в качестве примера передачу 1 шага, 50 шагов, 5000 шагов, 50000 шагов: время получения ответа будет 119 мс, 101 мс, 4,42 сек, 5,8 мин соответственно (рис.10), что позволяет сделать вывод, что для малого количества шагов зависимость времени передачи напрямую зависит от работы сервера, а не объёма передачи данных. Время передачи ощутимо зависит от количества передаваемых данных только начиная от какого-то значения.

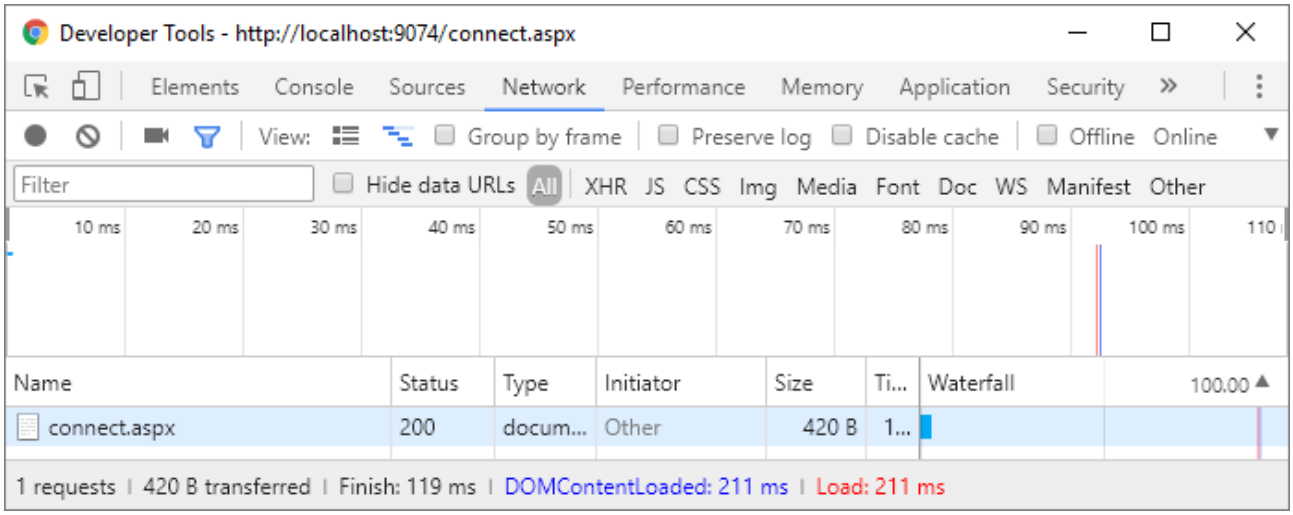

Рис. 10. Запрос данных, 1 шаг.

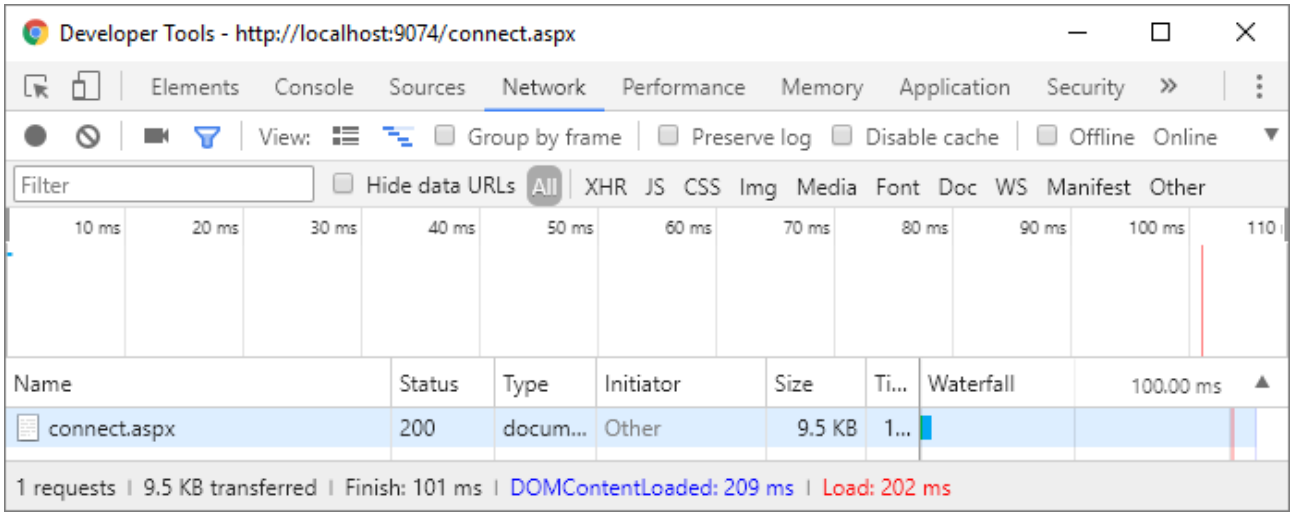

Рис. 11. Запрос данных, 50 шагов.

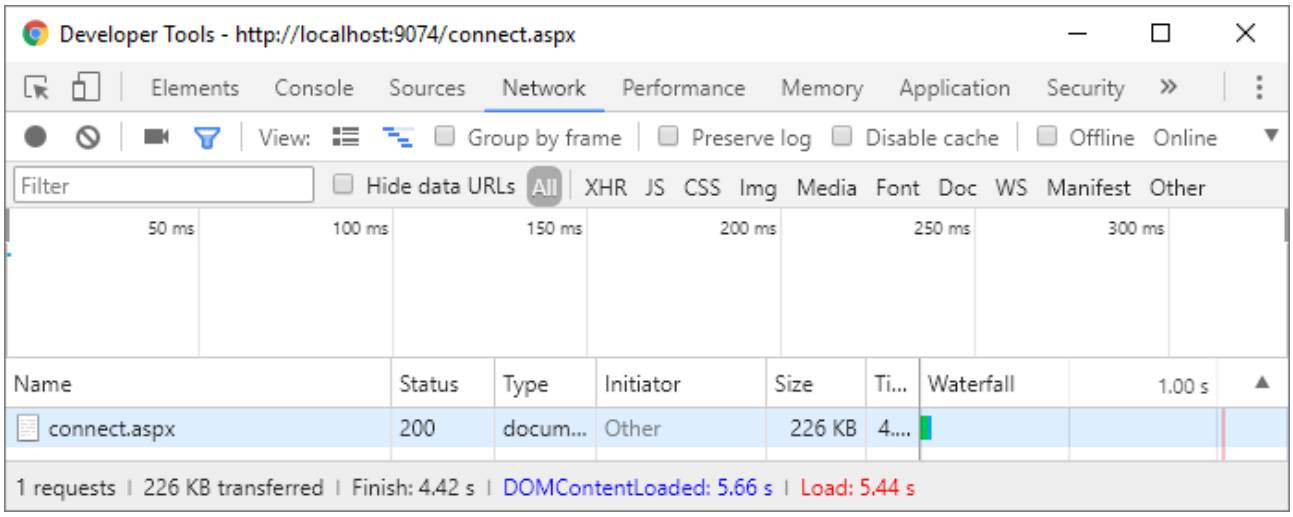

Рис. 12. Запрос данных, 5000 шагов.

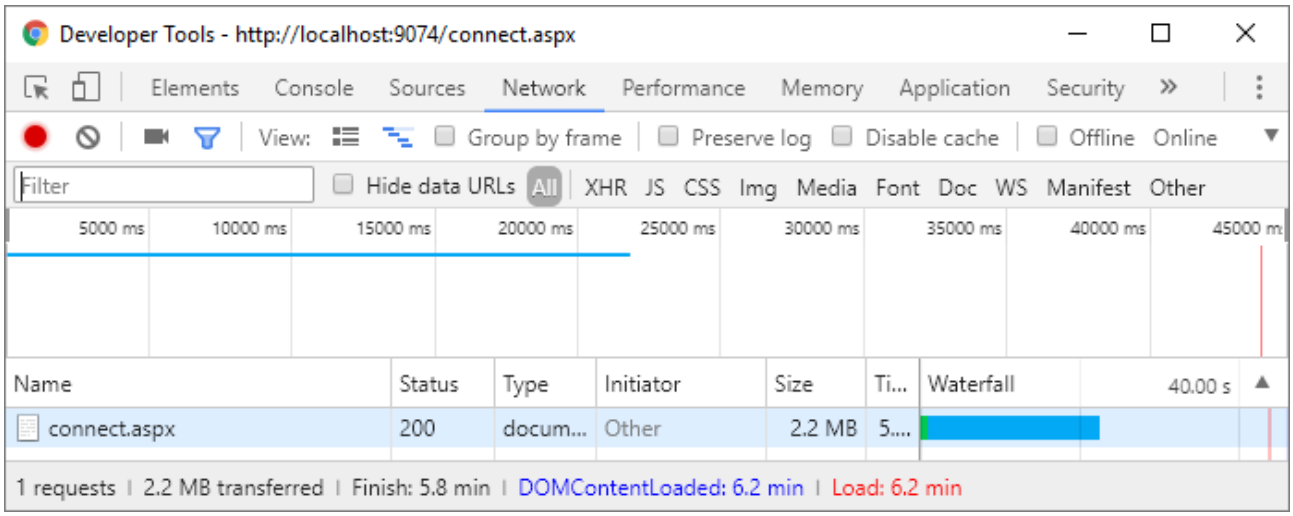

Рис. 13. Запрос данных, 50000 шагов.

В рассматриваемых примерах (рис.10-13) время передачи, которое приходится на один шаг для 1 шаг, 50 шагов, 5000 шагов, 50000 шагов соответственно 119 мс, 2 мс, 0.9 мс, 6,96 мс. Получается, самый быстрый способ передачи результатов в данном примере – передавать по 5000 шагов. Для каждой задачи, в зависимости от количества автомобилей на каждом шаге эта цифра может быть разной, поэтому вычислить её заранее нельзя. Выбор количества шагов при передачи данных не является строго определённым и может меняться в зависимости от настроек конструктора.

Запрос XMLHttpRequest позволяет переправлять данные в любом текстовом формате: JSON, XML, HTML и др. Для сокращения объёма передаваемых данных за счёт символов тегов (HTML, XML) или названий полей (JSON) порядок значений и формат заранее предопределены и фиксированы в настройках сервера и браузера в виде своего собственного формата:

5.3. Воспроизведение результатов в браузере

Необходимо учитывать, что визуализация может быть при разном количестве кадров в секунду, воспроизведение можно ставить на паузу и просматривать пошагово. Чтобы клиент при желании просматривать результаты вычислений не ожидал передачу с сервера, в браузере должен быть достаточный запас данных для отображения. В рамках одной вычисляемой задачи назовём минимальным пулом – минимальное количество шагов в браузере, которых гарантированно хватит для непрерывного отображения движения до следующего пополнения данных с сервера в браузере в любой момент вычислений. Таким образом, если количество передаваемых шагов больше или равно минимальному значению пула, то воспроизведение ни при каких условиях не прервётся из-за ожидания данных с сервера. В противном случае у клиента воспроизведение может остановиться, так как просто закончатся данные.

Минимальный размер пула получается эмпирическим путем с учётом всех накладных расходов на сетевое взаимодействие, скорость работы браузера, количество кадров в секунду, при котором воспроизводятся результат и т.п. Но если считать, что клиент не меняет частоту воспроизведения, не нажимает на паузу, скорость передачи данных по сети не меняется и линейно зависит от объема результатов, то можно вычислить некий размер пула, который гарантировано создаст задержку при воспроизведении. В рамках одной вычисляемой задачи назовём такой размер пула критическим.

Критический размер пула зависит от времени, которое требуется от запроса к серверу до пополнения пула. Рассмотрим подробнее временные затраты при пополнении пула:

$$T \geq I_{c>s} + S_{DB} + S_{RAM} + S_{com} + I_{s>c}, \text{где}$$

T - время, которое требуется на пополнение пула с момента формирования запроса на сервере,

$I_{c>s}$ – время передачи запроса от браузера к серверу,

S_{DB} – время получения и подготовка данных из базы данных на сервере,

S_{RAM} – время получения и подготовка данных из оперативной памяти сервера,

S_{com} – время вычисления новых данных, если имеющихся данных недостаточно для передачи,

$I_{s>c}$ – время передачи запроса от сервера к браузеру.

Параметры S_{DB} и S_{RAM} зависят запроса и места хранения искомых данных. Если запрашиваемое количество данных ещё целиком хранится в RAM (рис. 8), то запрос в базу данных не происходит и $S_{DB} = 0$. Такое возможно, только если в RAM ограничение на хранение данных не меньше размера

запроса от браузера. Поскольку скорость ответа оперативной памяти быстрее жёсткого диска, то это можно считать благоприятным исходом при обработке запроса.

Возможен и другой случай – когда все данные уже на жёстком диске, тогда $S_{RAM} = 0$.

Так же вероятен запрос данных, которые находятся частично в RAM и БД. Тут возможны следующие случаи:

1. Запущен процесс передачи данных из оперативной памяти на жёсткий диск. В этом случае, во избежание дублей при чтении данных поочерёдно из двух источников, сначала оканчивается процесс переноса хранимых данных на жёсткий диск, а потом уже чтение данных либо только с жёсткого диска, либо из двух источников.
2. Процесс переноса вычисленных результатов не запущен. Происходит чтение данных из двух источников. Пока данные считываются, перенос данных из RAM в БД не может начаться – искусственно блокируется, чтобы избежать дублей.

Один из самых неблагоприятных вариантов - запрос данных на сервере, которые ещё не вычислены или вычислены частично. Оба случая приводят к ожиданию окончания вычислений нужных шагов, после чего сразу происходит формирование ответа веб-браузеру.

5.4. Визуализация результатов

В итоге, разработанная система позволяет визуализировать исследуемый фрагмент транспортной сети с любым количеством перекрёстков (рис. 14), ограничиваясь только техническими возможностями.

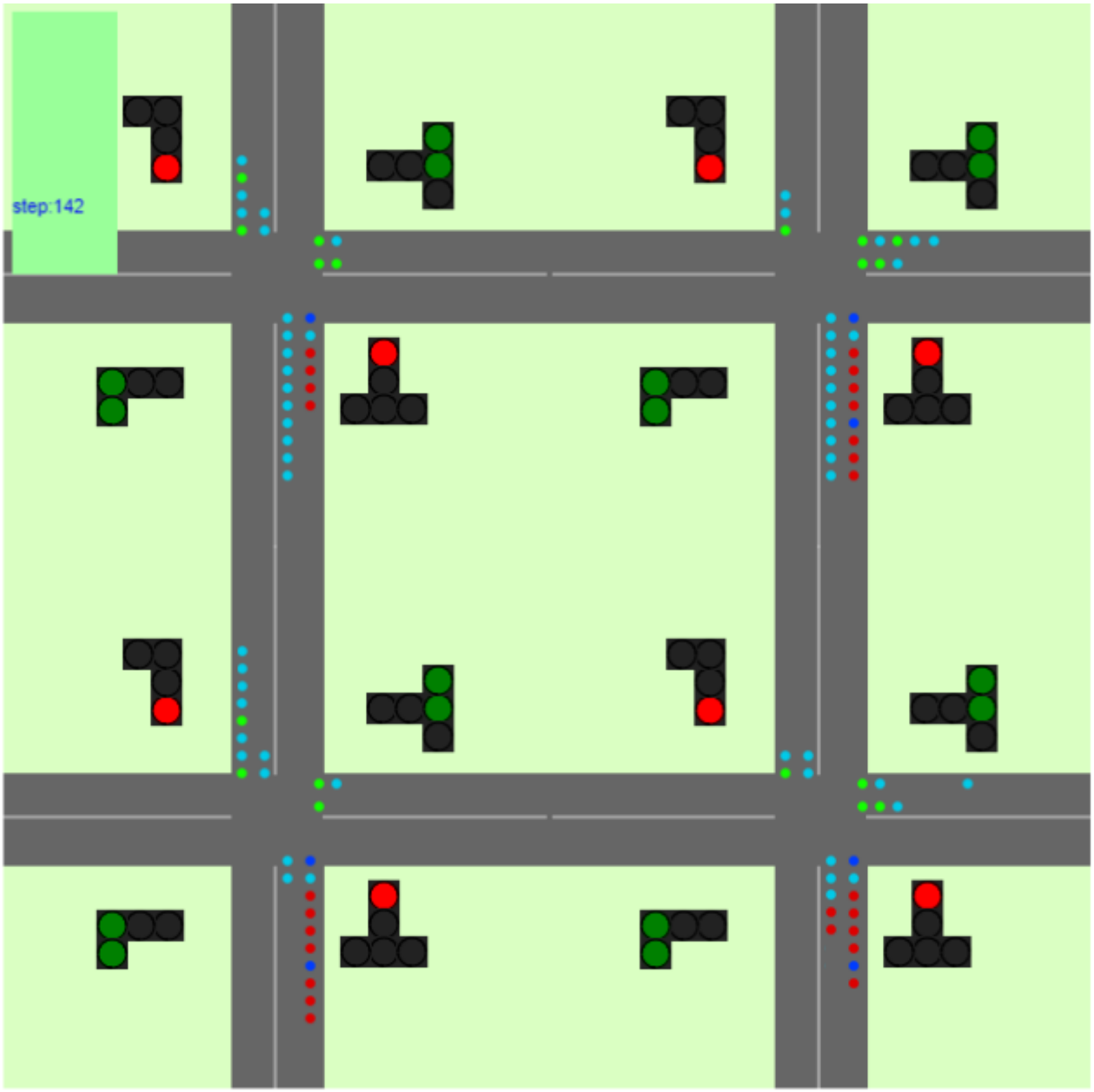

Рис. 14. Пример фрагмента транспортной сети.

Заключение

В связи с развитием вычислительных мощностей компьютерных систем появилась возможность моделировать транспортные потоки с учетом большого числа характеристик транспортных средств, что позволяет точнее воспроизводить реальные наблюдаемые закономерности дорожного движения. Соответственно, в результате моделирования появляется большой поток данных, обработка которых невозможна без специальных графических инструментов.

В течении года эксплуатации данного программного комплекса авторы получили от пользователей значительное количество замечаний и предложений по развитию, большая часть которых отражена в описываемой версии системы. Определенным успехом работ по развитию данного программного комплекса можно считать разработку и отладку новых алгоритмов перестроения автомобилей для микроскопической модели [4].

В дальнейшем планируется провести ряд вычислительных экспериментов по моделированию формирования широких движущихся кластеров в плотном транспортном потоке на дороге с въездами, сужениями и другими особенностями. Разработанный графический инструментарий позволяет достаточно легко перестраивать исследуемый фрагмент транспортной сети для учета таких особенностей и визуализировать результаты моделирования таких сложных дорожных кластеров.

Список литературы

[1] Трапезникова М. А., Чечина А. А., Чурбанова Н. Г. Двумерная модель клеточных автоматов для описания динамики транспортных потоков на элементах улично-дорожной сети. // Математическое моделирование, 2017, том 19, номер 9. С. 110-120

[2] Чечина А. А., Герман М. С., Ермаков А. В., Трапезникова М. А., Чурбанова Н. Г. Моделирование и визуализация потоков автотранспорта на элементах улично-дорожной сети с использованием комплекса программ CAM-2D // Препринты ИПМ им. М.В.Келдыша. 2016. № 124. 17 с.

[3] Трапезникова М. А., Чечина А. А., Чурбанова Н. Г.. Описание динамики транспортных потоков на элементах улично-дорожной сети с использованием двумерных математических моделей. // Препринты ИПМ им. М.В. Келдыша. 2016. № 93. 20 с.

[4] Чечина А.А. Новые алгоритмы перестроения автомобилей для микроскопической модели транспортных потоков на основе теории клеточных автоматов // Препринты ИПМ им. М.В.Келдыша. 2017. № 136. 14 с. doi:10.20948/prepr-2017-136

URL: http://library.keldysh.ru/preprint.asp?id=2017-136