МЕЖГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛОРУССКО-РОССИЙСКИЙ УНИВЕРСИТЕТ»

Кафедра «Автоматизированные системы управления»

МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ

Методические рекомендации к лабораторным работам для студентов специальности 1-40 80 02 «Системный анализ, управление и обработка информации» очной и заочной форм обучения

Могилев 2020

УДК 004.35: 004.3
ББК 32.973.202-04
М90

Рекомендовано к изданию
учебно-методическим отделом
Белорусско-Российского университета

Одобрено кафедрой «Автоматизированные системы управления» «14» апреля 2020 г., протокол № 9

Составитель канд. техн. наук Е. М. Борчик

Рецензент канд. техн. наук, доц. И. В. Лесковец

Методические рекомендации предназначены для студентов специальности 1-40 80 02 «Системный анализ, управление и обработка информации» очной и заочной форм обучения.

Учебно-методическое издание

МУЛЬТИАГЕНТНЫЕ СИСТЕМЫ

Ответственный за выпуск	А. И. Якимов
Корректор	Е. А. Галковская
Компьютерная верстка	Н. П. Полевничая

Подписано в печать . Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Печать трафаретная. Усл. печ. л. . Уч.-изд. л. . Тираж 16 экз. Заказ №

Издатель и полиграфическое исполнение:
Межгосударственное образовательное учреждение высшего образования
«Белорусско-Российский университет».
Свидетельство о государственной регистрации издателя,
изготовителя, распространителя печатных изданий
№ 1/156 от 07.03.2019.
Пр-т Мира, 43, 212022, Могилев.

Содержание

Введение

Целями преподавания дисциплины «Мультиагентные системы» являются изучение методов, моделей, средств и технологий компьютерной обработки информации и автоматизированного управления на основе теории искусственных агентов и мультиагентных систем, получение базовых знаний в области проектирования и программирования мультиагентных систем.

1 Лабораторная работа № 1. Изучение среды мультиагентного программного моделирования Netlogo

Цель работы: изучить функционал и возможности среды мультиагентного программного моделирования NetLogo.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.
2 Получить задание у преподавателя, выполнить типовые задания.
3 Сделать выводы по результатам исследований.
4 Оформить отчет.

Требования к отчету

1 Цель работы.
2 Постановка задачи.
3 Результаты исследования модели(ей) NetLogo.
4 Выводы.

Основные теоретические положения

Многие сложные системы удается исследовать только моделированием. Для систем, состоящих из большого количества независимых объектов, такие как поведение толпы, развитие многоклеточных организмов или военные операции, наиболее адекватным оказывается агентное моделирование. Существует многожество систем, предназначенных для этого, например российская проприетарная AnyLogic. Язык NetLogo хорошо зарекомендовал себя в образовании, и также годен и для исследовательских задач.

Синтаксис NetLogo минималистичен – разделенная пробелами последовательность имен и констант с редкой группировкой с помощью [] или (). [] служат для создания списков и группировки команд в блок в большинстве конструкций, () – обычные скобки для подвыражений. Имена ссылаются на встроенные или определенные программистом сущности – функции, переменные Команды (процедуры) объявляются

```
to имя [список имен аргументов]
  тело
end
```

Функции в NetLogo называются «репортеры» и объявляются немного по другому:

```
to-reporter имя [список имен аргументов]
 тело
```

```
 report возвращаемая величина
end
```

Компилятор знает «арность» каждой процедуры или функции («валентность») и не требует лишний раз использовать группировку, но с функциями высших порядков может ошибаться – тогда ему нужна подсказка в виде круглых скобок.

Пусть у нас определены функции:

```
to-report inc [x]
 report x + 1
end
```

```
to-report add [x y]
 report x + y
end
```

Тогда можно написать

```
add ((add (inc 1) (inc 2)) (inc 3))
```

Правильно работает также «map inc [1 2 3]», который возвращает список [2 3 4]. А для add уже придется написать скобки

```
(map add [1 2 3] [4 5 6]).
```

Агенты бывают трех видов – черепашки (turtle), связи (link) и пятна (patch – они же места в пространстве). Для черепашек и связей можно задать определяемую пользователем породу (breed). Агенты одного типа объединены в набор, соответствующий (agentset) – turtles, links и pathes. Представители одной породы также объединены в набор. Новая порода создается командой

```
breed [ninjas ninja]
```

где ninja – название породы, а ninjas – название набора, объединяющего всех агентов этой породы. Есть также особый агент – наблюдатель. Черепашки создаются командой create-turtles (с аргументом – количество создаваемых черепах), и далее находятся по индексу функцией turtle. Агент – «first class value», при желании его можно сохранить в переменной, но требуется это редко.

Агент или набор агентов могут быть контекстом для команды. Выполнение команд в контексте агентов – основной механизм работы с ними.

```
ask turtles [fd 1]
```

Этот код попросит всех черепашек сделать шаг вперед.

```
ask patch 17 13 [set pcolor pink]
```

А этот покрасит поле с координатами (17,13) в нежно-розовый цвет.

Породу можно установить динамически

```
ask turtle 1 [set bread ninjas]
ask ninja 1 [set pcolor black]
```

Переменные. В исходном Logo была реализована динамическая область видимости, что может создавать неудобства. Разработчики NetLogo запретили создание одноименных переменных, области видимости которых могут пересечься. Разработчикам библиотек это удобства не прибавляет, но в обучении скорее полезно.

Область видимости статическая, что упрощает использование функций высших порядков. Переменные могут быть глобальные, собственные для типа агентов или породы, формальными аргументами и локальные в блоке кода. Одно и то же имя переменной не может ссылаться на разные классы переменных, но локальные переменные в разных блоках, в том числе и в одной функции, могут называться одинаково. Имя локальной переменной не может совпадать с именем аргумента функции, где она определена или с именем глобальной или агентной переменной.

Глобальные переменные описываются

```
global [имя1 имя2 ...]
```

собственные

```
turtles-own [имя3 имя4 ...]
ninjas-own [имя5 имя6 ...]
```

Локальные переменные создаются командой

```
let имя7 начальное_значение
```

Функции высших порядков. Некоторые стандартные функции, такие как map, в качестве одного из аргументов получают другую функцию. Им можно просто указать имя передаваемой функции или написать замыкание. Замыкание представляет из себя код, заключенный в []. То, что это замыкание, а не список, компилятор угадывает по контексту. В качестве формальных параметров используются переменные '?', '?1', '?2' и т. д.

Использование своих функций высших порядков сложнее. Присвоить переменной или передать другой функции функцию-значение можно с помощью специальной функции task, а вызвать функцию из переменной – с помощью runresult (или run для команд).

```
to test1 [f]
  show (runresult f 1)
  show (map f [2 3 4])
end

test1 (task [? + 2])
```

Интерфейс и графика NetLogo. В программе присутствуют три вкладки (таба): «интерфейс», документация и редактор кода. На вкладке «интерфейс» присутствуют поле графических объектов (в том числе и изображение мира), поле текстового вывода и REPL. В REPL можно переключать режимы, задающие, в каком контексте выполняются вводимые команды – «наблюдатель», «черепахи», «пятна» и «связи». REPL ограниченный – описывать переменные, породы, процедуры и функции в нем нельзя (они создаются вне контекста), придется переходить в редактор кода.

Основное окно среды NetLogo содержит поле для размещения объектов. Чтобы разместить объект, необходимо нажать на кнопку «Добавить» (Add) и выбрать объект (рисунок 1.1)

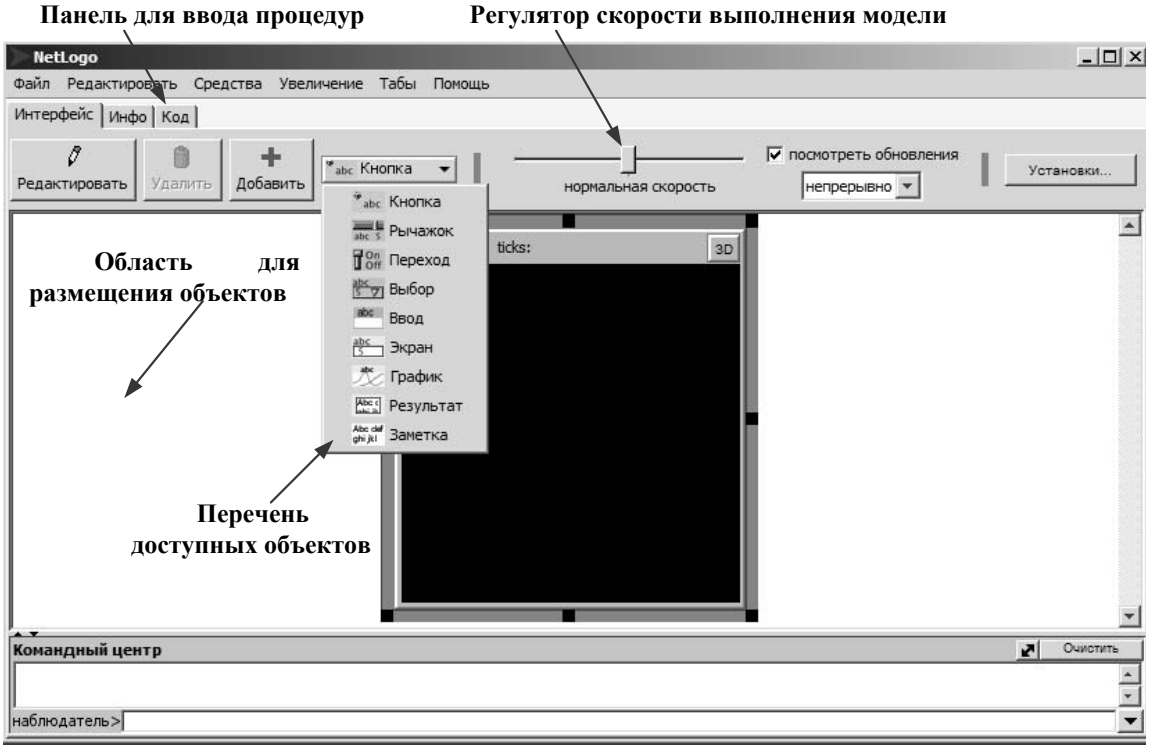

Рисунок 1.1 – Основное окно среды NetLogo

Кнопка (Button) – объект представляет собой кнопку. Содержит команду, которую выполняет при нажатии.

Рычажок (Slider) – представляет переменную, численное значение которой можно изменять визуально.

Переход (Switch) – переключатель, логическая переменная, значение которой можно изменять визуально.

Выбор (Chooser) – переменная, предоставляющая возможность выбора значений из списка, составленного пользователем.

Ввод (Input) – поле для ручного ввода значения для заданной переменной.

Экран (Monitor) – отображает текущее значение выбранной переменной.

График (Plot) – объект является координатной плоскостью для построения графиков. Чтобы построить графики, необходимо указать, какие переменные будут являться значениями функций.

Результат (Output) – подобно объекту Monitor выводит на экран информацию. Однако отличие в том, что может отображать любую текстовую информацию по заказу пользователя.

Заметка (Note) – объект для составления записей в рабочей области (можно использовать в качестве комментариев).

Вкладка Инфо (Information) содержит в себе информацию о модели. Для заполнения информации следует нажать кнопку Редактировать (Edit).

Вкладка Код (Procedures) предназначена для создания пользовательских функций. Выражения в объектах и процедуры пишутся на языке Logo.

Более подробную информацию об объектах среды NetLogo и о языке программирования Logo можно найти в руководстве пользователя в закладке Помощь(Help à NetLogo User Manual).

Окно моделирования системной динамики также имеет область для размещения объектов Средства (Tools à System Dynamics Modeler) (рисунок 1.2).

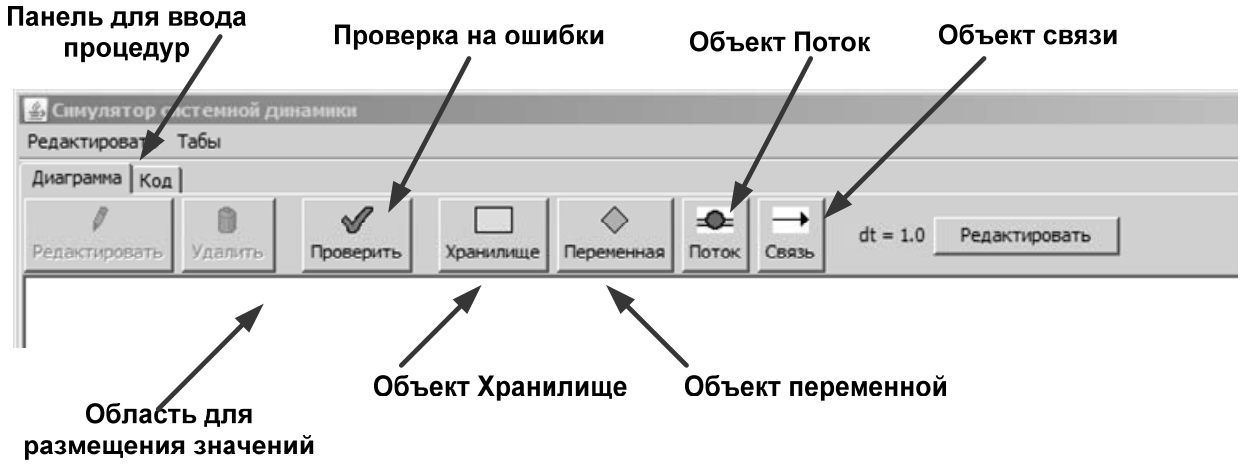

Рисунок 1.2 – Окно моделирования системной динамики

Хранилище (Stock) – объект, способный содержать в себе количественную характеристику совокупности чего-либо.

Переменная (Variable) – объект переменной, может содержать численное значение или заданное выражением. Может меняться при изменении объектов, от которых она зависит, но нельзя менять напрямую.

Поток (Flow) – объект потока. Содержит в себе константу или выражение. Результирующее значение отнимает от стока с одной стороны и прибавляет к стоку с другой стороны (если таковые имеются).

Связь (Link) – соединяет два объекта, позволяя делать обращения между ними.

dt – переменная, влияющая на точность при каждом шаге. При создании выражений в объектах Flow автоматически добавляется к ним. С его помощью можно регулировать объем потока.

Примечание – Из-за использования средой этого параметра все выражения в объектах желательно записывать внутри скобок: (<выражение>).

В зоне, где нарисовано поле с миром, можно «мышкой» (нажав правую кнопку на свободном месте) создавать другие графические объекты и элементы интерфейса. При создании им надо задать имена, по которым к ним можно обращаться из программы. В файле эти объекты сохраняются в текстовом виде после кода, но в малопригодной для ручного редактирования форме.

Например, если мы создадим «график» с именем «plot1», то команда

```
set-current-plot "Class Histogram"
 histogram map [position ? [red green blue]] ([color] of turtles)
```

нарисует график распределения черепах по красному, зеленому и синему цветам (предполагая, что другие не встречаются).

***Пример 1* – Создание и группировка черепах.**

Научимся создавать, расставлять и группировать черепах.

Создадим новый проект. У черепашек Netlogo есть свойства – номер, цвет, координаты и т.д. Кроме того, мы можем задавать черепашкам новые свойства – turtles-own. Свойства всегда задаются в начале программы. Например, зададим для черепашек свойство group. Свойство будет иметь два значения – ложь или истина. Принадлежит черепашка к группе или не принадлежит.

```
turtles-own [group]
```

Вообще внутри turtles-own [] можно перечислить множество свойств

Теперь создадим первую процедуру, которая будет очищать все имеющиеся объекты и значения. Далее следует программа с комментариями. Комментарии задаются двумя символами ;;

Создаем новых черепах:

```
to setup
ca – очищаем значения
каждая черепашка располагается в случайной точке экрана
crt 50 [  ;; создаем 50 черепах и разбрасываем их по экрану
setxy random-xcor random-ycor
set group false
]
```

черепахи рождаются по порядку и поворачиватся последовательно. Когда мы им потом даем команду разойтись, они образуют круг

```
cro number_of_turtles [fd max-pxcor - 2]
```

Этого же результата можно достичь, если мы используем команду

```
layout-circle
end

to circle

layout-circle turtles max-pxcor – 2

end
```

Мы умеем создавать черепашек и расставлять их по экрану. Теперь попробуем собрать черепашек одинакового цвета.

Группируем черепах по цветам:

to collect_color

Первая версия – черепашки объединяются, но делают много лишних движений

```
  ask other turtles with [color = [color] of myself]
[face myself jump (distance myself) - 1]
```

Во второй версии задаем черепашкам свойство принадлежности к группе и в начале устанавливаем значение этого свойства в false

```
if group = false [ask other turtles with [(color = [col-
or] of myself) and (group = false)]
[
```

Черепашка прыгает в сторону вызывающего агента

```
face myself jump (distance myself) – 1
```

Черепашка устанавливает свою принадлежность к группе в true – и теперь ее не будут вызывать

```
set group true create-link-to myself]]
end
```

Пятна Netlogo.

Пятна – точки – patches

Агенты NetLogo, которые связаны с конкретными координатами на экране. К точке всегда можно обратится. Например:

```
ask patch 0 0 [set pcolor yellow]
```

Ограничения для пятен.

Пятно не может умереть – невозможно пятну дать команду умереть.

Невозможно создать новое пятно. Пятно не может hatch. Однако, пятно может родить новую черепашку sprout или черепашку какой-то специальной породы sprout-breed. По команде sprout в данной точке появляется черепашка.

Команды для пятен и предназначение команд приведены в таблицах 1.1 и 1.2.

Таблица 1.1 – Команды для пятен

Команда для пятен	Предназначение команд
`clear-patches (cp)`	Очистить, стереть все пятна и вернуть их в исходное значение – черный цвет
`diffuse patch-variable number`	Значение переменной раздается на ближайших соседей – диффузия цвета, запаха на поверхности
`diffuse4 patch-variable number`	Значение переменной раздается на четырех соседей – ближайшие соседние точки по прямой
`import-pcolors`	Считать цвета из файла – картинки
`patch-at dx dy`	Пятно с координатами относительно данного агента

Таблица 1.2 – Пример команд для пятен

Команда для пятен	Предназначение команд

`ask patch 2 5 [set pcolor yellow ask patch-at 2 2 [set pcolor red]]`
 	
`patch-left-and-ahead angle distance`
	Пятно слева и сверху

`patch-at-heading-and-distance`
 	
`ask patch-at-heading-and-distance -90 1 [ set pcolor green ]`
	Пятно по направлению и на расстоянии

Примеры команд и процедур с пятнами.

Окрасить все пятна в зависимости от расстояния до центра

```
ask patches [set pcolor distancexy 0 0]
```

Покрасить точки вокруг центральной точки в зеленый цвет

```
ask patch 0 0 [set pcolor yellow ]
ask patch 0 0 [set pcolor yellow ask neighbors4 [ set
pcolor green ]]
ask patch 0 0 [set pcolor yellow ask neighbors4 [ set
pcolor green ask neighbors [set pcolor red] ]]
```

и так далее, если хотим перекрашивать соседей.

Связи NetLogo. Связи в NetLogo такие же агенты, как черепахи NetLogo и пятна NetLogo. Связь всегда связывает двух черепах (два узла). Если одна из черепах погибает, то погибает и связь. Связи бывают направленными и ненаправленными.

Ненаправленные связи создаются командами к черепахам – create-link-with или create-links-with. Например:

`ask turtle 0 [create-link-with turtle 1]` – команда черепахе 0 создать связь с черепахой 1;

`ask turtle 0 [create-links-with other turtles]` – команда черепахе 0 создать связи со всеми другими черепахами;

`ask turtles [create-links-with other turtles]` – команда всем черепахам создать связи со всеми другими черепахами.

Направленные связи:

create-links-from или create-link-from – направленная связь других агентов к данному агенту; например:

ask turtle 0 [create-link-from other turtles];

create-link-to или create-links-to – направленная связь от данного агента к другому или другим;

ask turtle 0 [create-link-to other turtles].

Связи между породами.

Породы NetLogo связываются такими же командами, как и черепахи: `create-<breed>-from, create-<breeds>-from, create-<breed>-to, create-<breeds>-to, create-<breed>-with, create-<breeds>-with`

Layout.

Специально для того, чтобы улучшить видимость сети из узлов и связей, в NetLogo появились процедуры layout.

layout-circle (рисунок 1.3)

`layout-circle turtles radius` – все черепашки выстраиваются по кругу заданного радиуса:

`layout-circle turtles with [color = red] 5` – все красные черепашки в круг с радиусом 5.

layout-radial (рисунок 1.4)

`layout-radial`

`layout-radial turtles links (turtle 0)` – черепашки выстраиваются относительно корневого агента, указанного в скобках.

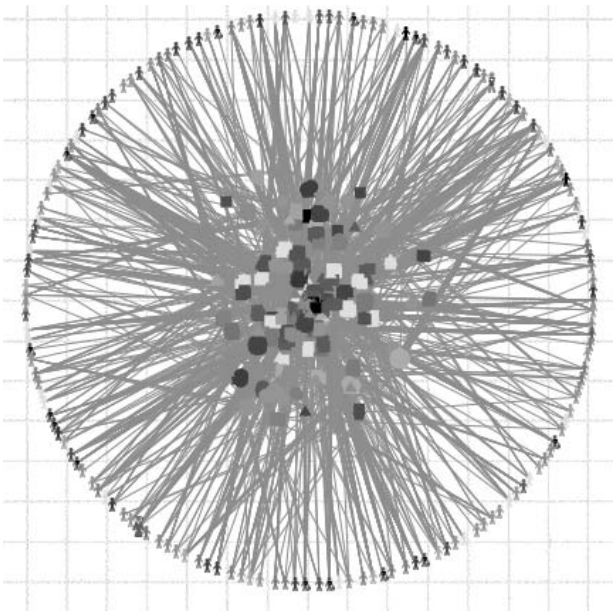

Рисунок 1.3 – Layout-circle

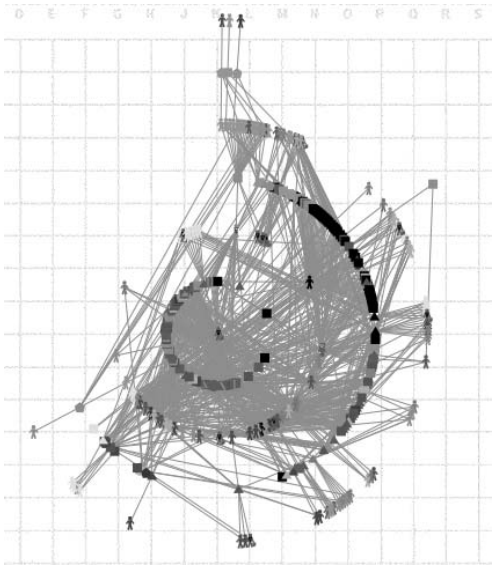

Рисунок 1.4 – Layout-radial

Пример процедуры:

```
 to clustC
let Centr max-one-of users [nw:betweenness-centrality]
layout-radial turtles links Centr
export-view (word "../sber_results/u325/clusters/" [who] of Centr ".png" )
ask Centr [die]
ask users with [count my-links = 0] [die]
end
```

layout-tutte

```
to make-a-tree
 set-default-shape turtles "circle"
 crt 6
 ask turtle 0 [
   create-link-with turtle 1
   create-link-with turtle 2
   create-link-with turtle 3
 ]
 ask turtle 1 [
   create-link-with turtle 4
```

```
      create-link-with turtle 5
    ]
    ; place all the turtles with just one
    ; neighbor on the perimeter of a circle
    ; and then place the remaining turtles inside
    ; this circle, spread between their neighbors.
    repeat 10 [ layout-tutte (turtles with [link-neighbors = 1]) links 12 ]
  end
  repeat 10 [layout-tutte max-n-of 15 turtles [nw:betweenness-centrality] links 22]
```

layout-spring – силовой алгоритм размещения.

`layout-spring turtle-set link-set spring-constant spring-length repulsion-constant`

`repeat 30 [ layout-spring turtles links 0.2 5 1 ]` – черепахи, связи, жесткость связи (натяжение), естественная длина связи (которой стремиться достигнуть), сила отталкивания между узлами

`layout-spring turtle-set link-set spring-constant spring-length repulsion-constant`

layout-spring располагает черепах в черепаха-набор, как если бы ссылки в связи-наборе пружины и черепахи отталкиваются друг от друга. Черепахи, которые соединены ссылкам в связи-набор, но не включенные в набор черепахи, рассматриваются как якоря и остаются неподвижными.

`spring-constant`

`spring-constant` – «натянутости» пружины. Это «сопротивление» к тому, чтобы изменить длину связи. Чем выше, тем большее усилие надо прикладывать.

`spring-length` – естественная длина связи, которой стремятся достигнуть все связи.

`repulsion-constant` – постоянное отталкивание, которое является мерой отталкивания между узлами. Это сила, которую 2 узла на расстоянии 1 будут оказывать друг на друга. Чем выше repulsion-constant, тем сильнее узлы отталкиваются.

Например,

`layout-spring turtle links 2 1 0.1`

`layout-spring turtle links 2 0.1 0.1` - нужно прикладывать большое усилие, расстояние не велико и они слабо отталкиваются.

`layout-spring (turtles with [any? link-neighbors]) links 0.4 6 1` – из одной библиотечной модели NetLogo - растаскивать на большую величину

`layout-spring turtles links (1 / factor) (7 / factor) (1 / factor)` – тот же примерно принцип - 1:7:1 соотношение для разделения узлов.

Команды связям.

clear-links – уничтожить все связи

both-ends

```
ask link 0 1 [print both-ends]
```

die – умереть

hide-link – спрятать связь

in-<breed>-neighbor?

in-<breed>-neighbors

in-<breed>-from

in-link-neighbor? связан ли данный узел с данным узлом?

```
ask turtle 1 [show in-link-neighbor? turtle 2]
```

in-link-neighbors – все, кто связан с данным агентом направленными связями (к данному агенту)

```
ask in-link-neighbors [set heading [heading] of myself]
```
– всем моим связанным соседям повернуться в ту же сторону, что и я

out-link-neighbors – все, кто связан с данным агентом направленными связями (от данного агента)

in-link-from – возвращает связь, которая связывает агента с данной черепахой. Например:

```
ask turtle 0 [ create-link-to turtle 1 ]
ask turtle 1 [ show in-link-from turtle 0 ]
```

is-directed-link? если связь направленная, возвращает true (иначе возвращает false)

is-link? если это связь, возвращает true

is-link-set?

is-undirected-link?

<breed>-neighbor?

<breed>-neighbors

<breed>-with

link-heading – направление связи

```
ask link 0 1 [ print link-heading ]
```

link-length

```
ask link 2 1 [ print link-length ]
```

link-neighbor?

link - связь. Обращение к связи.

Например – `Ask link 0 1 [set color 15]`

links обращение ко всем связям

```
ask links [set color 25]
```

Если мы хотим выделить группу связей – `ask links with [link-length > 3] [set color 45]`

```
ask links [ask both-ends [ask neighbors [set pcolor 5] ] ]
ask links with [link-length > 3 ] [ask both-ends [ask neighbors [set pcolor 15] ] ]
```

`ask links with [link-length < 2 ] [ask both-ends [ask neighbors [set pcolor 25] ] ]` – Здесь мы говорим связям, длина которых меньше или больше выбранного значения, обратиться к черепашкам на обоих концах и передать пятнам команду изменить цвет.

`links-own`
`<link-breeds>-own`
`link-neighbors`
`link-with`
`my-<breeds>`
`my-in-<breeds>`

`my-in-links` Возвращает все направленные ко мне связи

Например, я хочу всех черепах, которые связаны с данной направленными к ней связями что-то сделать

`ask my-in-links [ask other-end [be_free]`

`my-links` – возвращает все связи данного агента

Например, `ask my-link [die]` – а что еще можно сказать связям?

`my-out-<breeds>`

`my-out-links` мои направленные исходящие связи

`no-links`

`other-end` команда от черепашки к связи или связям. Связь передает другому узлу команду к исполнение

`ask my-in-links [ask other-end [die]]` или `ask my-in-links [ask other-end [hatch]]`

`out-<breed>-neighbor?`

Пример 2 – Построить модель (третьего уровня) перемещения продукции по следующему маршруту:

Фабрика –> Склад –> Магазин –> Покупатель.

Результат может выглядеть так, как показано на рисунке 1.5.

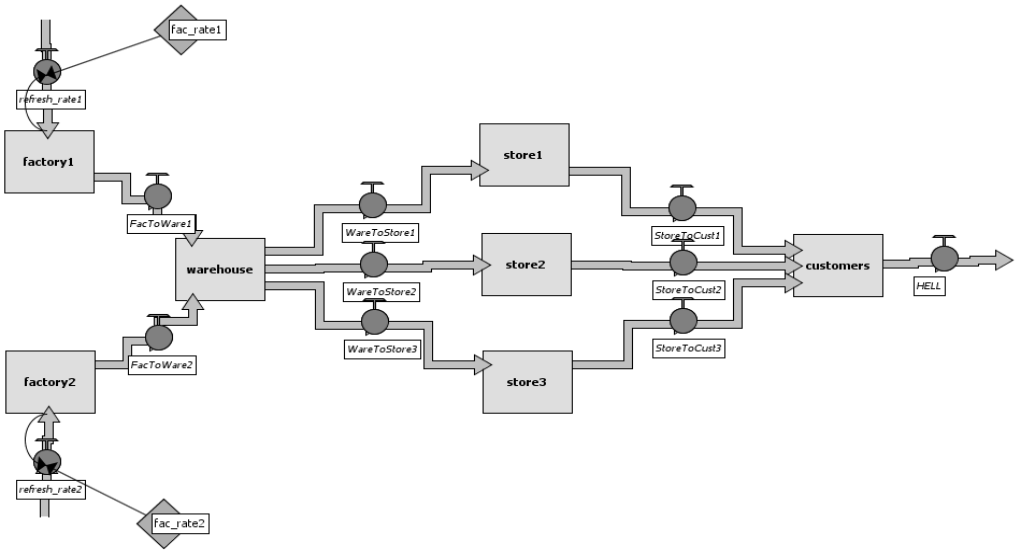

Рисунок 1.5 – Пример построенной модели в среде NetLogo

Библиотека моделей. К NetLogo прилагается обширная библиотека моделей, от учебных и развлекательных до исследовательских. Доступ к модели осуществляется через меню File. Как правило, в интерфейсе моделей есть кнопки Setup и Go и для запуска их надо нажать в этом порядке. Setup инициализирует модель, после чего ее настройки можно изменить через интерфейс.

Задания

1 Изучить интерфейс среды NetLogo. Создать собственный новый проект NetLogo с реализацией ввода нескольких переменных, вычислением значений функций высших порядков и выводом их значений.

2 Изучить синтаксис NetLogo и основные механизмы работы команд, выполнив примеры теоретической части лабораторной работы.

3 Построить имитационную модель (производство, склад, магазин, покупатель), которая имеет определенное количество уровней. Объем потока выбрать самостоятельно. При помощи инструмента Plot получить динамику значений стоков.

4 Составить отчёт о проделанной работе, который должен содержать: результаты работы моделей в среде NetLogo и сами модели, выводы о проделанной работе.

Контрольные вопросы

1 Агенты каких видов существуют в NetLogo?

2 Где находится библиотека моделей NetLogo?

3 Для каких целей предназначена вкладка Код (Procedures) в NetLogo?

2 Лабораторная работа № 2. Исследование мультиагентной модели дорожного движения

Цель работы: перейти от наблюдения моделей к манипулированию ими, наблюдать внутреннюю работу модели и уметь изменить ее внешний вид.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования мультиагентной модели дорожного движения.

4 Выводы.

Основные теоретические положения

В модели «Движение на дороге» (модель «Traffic Basic», в секции «Social Science», библиотека моделей «Models Library» (из меню File) можно увидеть одну красную машину в потоке голубых машин. Поток машин двигается в одном направлении. Каждая из машин может вызвать скопление и остановить движение. Модель показывает, что пробка может возникнуть без всякой причины: не нужен ни разбитый грузовик, ни поломанный мост, ни другой инцидент. Вы можете изменять установки модели и наблюдать ее множество раз, пока не придете к полному пониманию модели.

В процессе использования этой модели можете предложить какие-то дополнения, которые Вы хотели бы в нее внести.

Рассматривая модель «Движение на дороге», Вы можете отметить достаточно простую среду: черный фон с белой улицей и некоторое число голубых машин и одну красную машину. Изменения, которые мы могли бы внести в эту модель, включают следующее: изменение формы и цвета автомобилей, добавление дома или света на улице, создание светофора или даже создание другой полосы на дороге. Некоторые из этих изменений носят косметический характер, в то время как другие затрагивают ее поведение. В этой лабораторной работе мы сфокусируемся на простейших или косметических изменениях. Чтобы осуществлять эти простые изменения, мы будем использовать Командный центр (Command Center).

Командный центр (The Command Center) размещается на панели «Interface Tab» и позволяет вводить команды или директивы в модель.

Команды являются инструкциями или предписаниями для агентов NetLogo's: черепашек (turtles), пятен (patches), наблюдателя (observer).

Мир NetLogo – это двумерный мир, который сделан из черепашек, пятен и обозревателя (turtles, patches, observer). Пятна (patches) создают основу, на которой черепашки (turtles) могут двигаться и обозреватель(observer) is a being that oversee everything that is going on in the world.

В командном центре мы можем давать команды обозревателю, черепашкам и пятнам. Мы выбираем между этими опциями, используя всплывающее меню (в нижнем левом углу Командного центра) или клавишу «tab» на клавиатуре.

Обозреватель видит весь мир и следовательно может давать команды черепашкам и пятнам, используя ask.

Например, командой «O>ask patches [set pcolor yellow]» обозреватель попросил пятна поменять цвет на желтый. Пример команды, данной прямо группе агентов, уже непосредственно пятнам: «P>set pcolor white».

Работа с цветами.

Вы могли заметить, что в нашей работе мы использовали два различных слова для изменения цвета color и pcolor.

Мы называем color и pcolor «переменными». Некоторые команды и переменные относятся только к черепашкам, а некоторые только к пятнам. Например, color – это переменная для черепашек, а pcolor - для пятен.

Чтобы изменять цвета черепашек и пятен, или в нашем примере машин и фона, мы должны немного познакомиться с тем, как NetLogo обходится с цветами.

В NetLogo, все цвета имеют свой номер. Во всех наших упражнениях мы использовали наименование цвета. Это потому, что NetLogo распознает 16 различных названий цветов. Это не означает, что NetLogo распознает всего 16 цветов. Ниже приводится диаграмма цветового пространства NetLogo (рисунок 2.1)

Чтобы получить цвет, у которого нет собственного имени, Вы должны ссылаться на него по номеру или добавляя, или отнимая число от названия цвета.

Например, когда Вы печатаете set color red, это означает то же самое, если бы Вы напечатали set color 15. И Вы можете получить более темную или светлую версию того же цвета, если возьмете чуть большее или меньшее число.

Мониторы агентов и команды агентов (Agent Monitors and Agent Commanders). В предыдущем пункте мы использовали команду set, чтобы изменять цвета всех машин. Рассмотрим, как изменить цвет одной машины.

Щелкните на красной машинке правой кнопкой мыши. Из popup меню, выберите "inspect turtle 0".

Появится монитор черепашки (машинки) (рисунок 2.2).

В этом мониторе мы можем увидеть значения всех переменных, установленных для красной машинки. Переменная – это место, которое содержит значение и это значение может быть изменено. Помните, мы упоминали, что все цвета представлены в компьютере номерами? Это относится и к аген-

там. Например, черепашка имеет идентификационный номер, который мы называем «who» – номер. Монитор (см. рисунок 2.2) показывает, что номер черепашки – 0, цвет – 15 (красный), а форма – автомобиль (car).

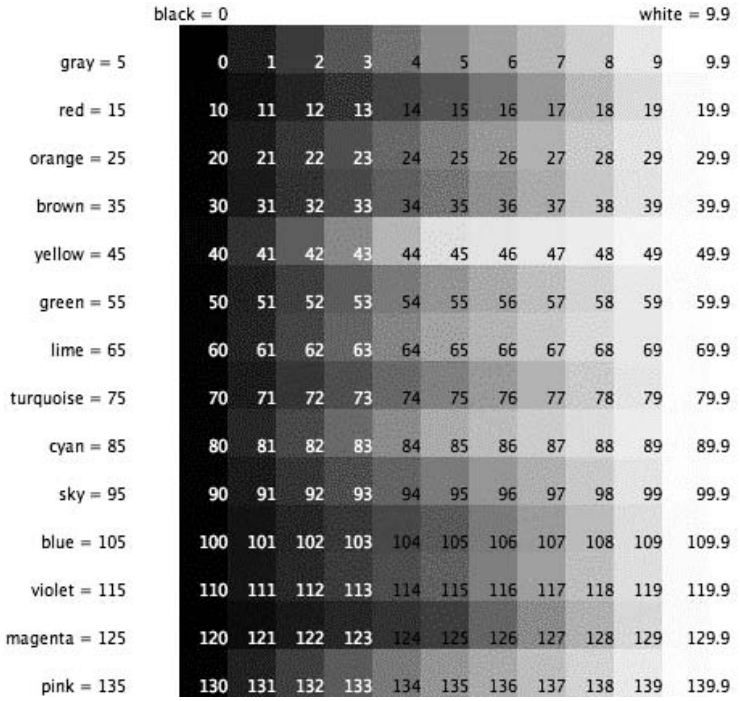

Рисунок 2.1 – Диаграмма цветового пространства NetLogo

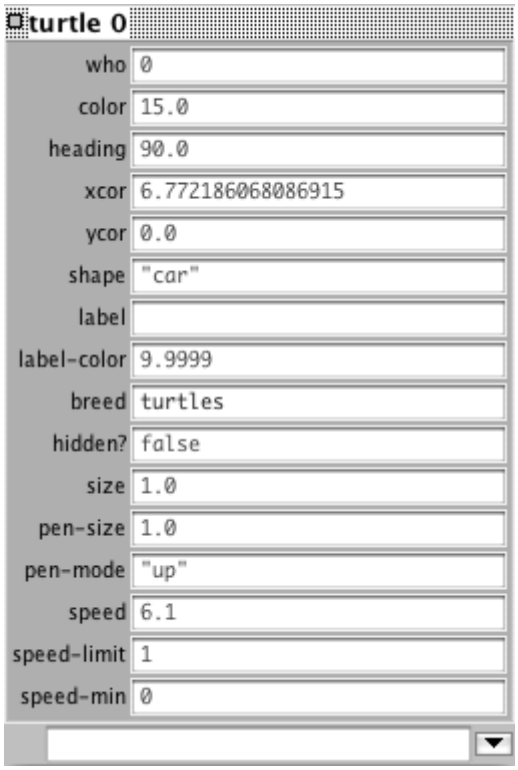

Рисунок 2.2 – Монитор черепашки

Есть два способа открыть монитор черепашки, один – правым щелчком мыши. Еще один путь – выбрать «Turtle Monitor» из меню Tools, затем напечатать «who» – номер черепашки, которую Вы хотите инспектировать в поле «who» и нажать return. Другой путь напечатать inspect turtle 0 (или другой номер) в Командном Центре (Command Center).

Монитор закрывается нажатием на крестик в правом верхнем углу.

Теперь, когда мы знаем больше о мониторе агентов (Agent Monitors), имеются три пути изменения индивидуального цвета черепашки.

Один путь – использовать прямоугольник, называемый Agent Commander, который находится внизу агентного монитора (Agent Monitor). Вы печатаете здесь команду, подобно тому, как Вы это делали в Командном центре, но это команда относится только к выбранной отдельной черепашке.

Второй путь изменить цвет одной черепашки - это перейти прямо к переменной color в Turtle Monitor и изменить значение.

Третий путь изменений индивидуального цвета черепашки или пятна – это использование обозревателя. Поскольку обозреватель видит весь мир NetLogo, он может давать команды как отдельным черепашкам, так и группам черепашек.

Также, как имеются мониторы черепашек, существуют и мониторы пятен (Patch Monitors). Они работают сходным образом.

Если Вы попытаетесь в обозревателе дать команду ask patch 0 [set pcolor blue], то получите сообщение об ошибке (рисунок 2.3).

Чтобы попросить отдельную черепашку о чем-нибудь, мы использовали ее «who» – номер.

Но пятна не имеют такого номера «who», следовательно, нам необходимо ссылаться на них каким-то другим путем.

Напоминаем, пятна расположены в системе координат. Два числа необходимы, чтобы начертить точку на графике: значение координаты x-axis и а y-axis. Размещение пятен спроектировано таким же образом.

Откройте patch monitor для любого пятна (рисунок 2.4).

Монитор показывает, что для пятна его pxcor переменная равна -11 и его pycor переменная равна -4. То есть это точка с координатами x = -11 and y = -4.

Чтобы попросить это отдельное пятно изменить цвет, надо использовать его координаты.

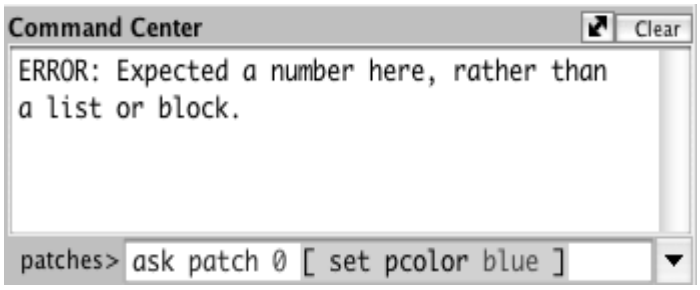

Рисунок 2.3 – Командный центр

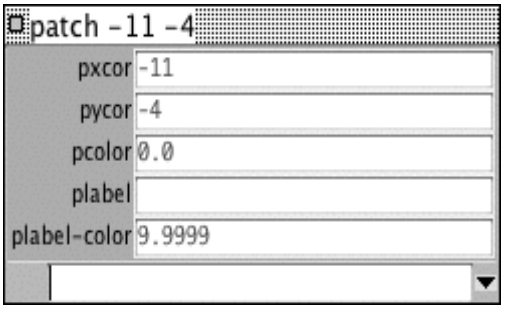

Рисунок 2.4 – Patch monitor пятна

Что дальше? Вы можете захотеть применить изученную технику к другим моделям из библиотеки.

Задания

1 Откройте модель «Traffic Basic», в секции «Social Science». Запустите модель и несколько минут наблюдайте за ней. Если возникнут вопросы, обращайтесь к вкладке (панели) Information tab.

2 Выполните задания, приведенные ниже, для изучения работы командного центра.

2.1 В модели «Движение на дороге» (Traffic Basic) нажмите кнопку «setup». Перейдите в командный центр (Command Center).

Щелкните мышкой в белом прямоугольнике внизу Командного центра (Command Center) (рисунок 2.5.) Нажмите кнопку «enter».

Что Вы видите?

2.2 Напечатайте текст, написанный ниже на рисунке 2.5.

2.3 Вы можете отметить, что фон весь стал желтым и улица пропала.

Почему машины не стали желтыми тоже?

Рисунок 2.5 – Командный центр

2.4 Вернитесь к команде, написанной в командном центре: мы попросили только пятна (patches) изменить их цвет. В этой модели машины представлены другим видом агентов, называемым черепашками («turtles»). Следовательно, машины не получили этих инструкций и не поменяли цвет.

Что происходит в Командном центре (*Command Center*)?

2.5 Вы можете заметить, что команды, которые Вы напечатали, появляются уже в середине белого прямоугольника Командного центра. Это показано на рисунке 2.6.

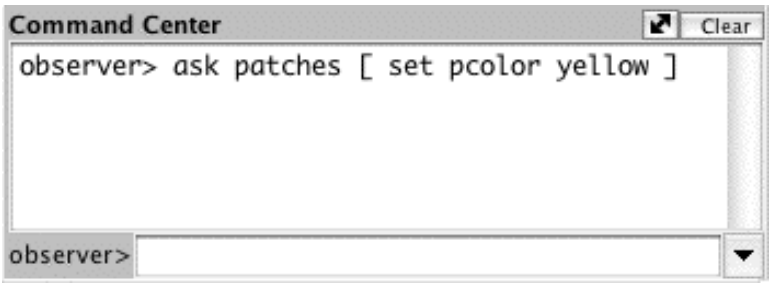

Рисунок 2.6 – Командный центр

2.6 Напечатайте внизу в белом прямоугольнике Командного центра текст представленный на рисунке 2.7.

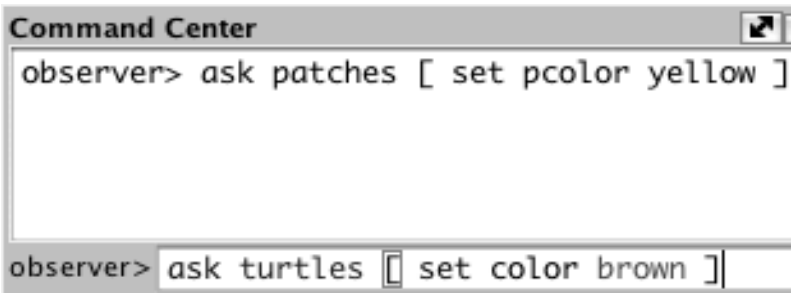

Рисунок 2.7 – Командный центр

Какой результат Вы ожидали?

2.7 Вы видите, что на желтом фоне появляются в середине коричневые машины (рисунок 2.8).

Рисунок 2.8 – Машины

2.8 В командном центре (Command Center) щелкните на «O>» в нижнем левом углу (рисунок 2.9).

Выберите «Turtles» из всплывающего меню.

Напечатайте set color pink и нажмите return.

Нажмите и удерживайте клавишу tab, пока не увидите «P>» в левом нижнем углу.

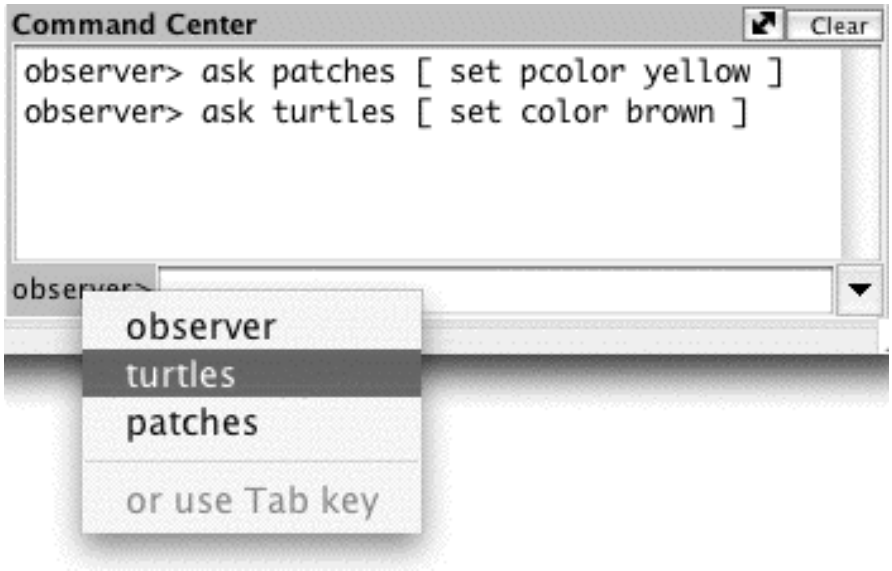

Рисунок 2.9 – Командный центр

Напечатайте set pcolor white и нажмите return. Что Вы теперь видите?

Можете Вы отметить отличие между этими двумя командами и командами обозревателю, которые мы давали раньше?

2.9 Выполните примеры команд «O>ask patches [set pcolor yellow]» и «P>set pcolor white». Нажмите «setup».

Что случилось?

3 Изучите работу с цветами.

3.1 Выберите «Turtles» из popup меню Командного Центра (или используйте клавишу tab). Введите set color blue and нажмите return.

Что случилось с машинами?

3.2 Подумайте о том, как Вы сделали машинки голубыми, и попытайтесь сделать пятна красными. Если Вы дадите пятнам команду: set color red, получите сообщение об ошибке (рисунок 2.10). Напечатайте set pcolor red вместо прежнего и нажмите return.

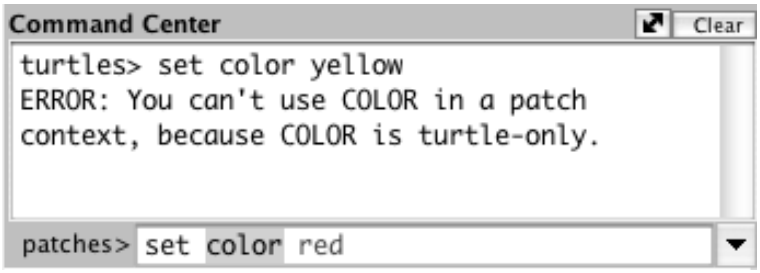

Рисунок 2.10 – Командный центр

Выберите «Patches» из popup меню в Command Center. Напечатайте set pcolor red - 2 (Пробелы вокруг «-» обязательны). Это сделает темнее цвет: set pcolor red + 2. А это - светлее: set pcolor red - 2. Вы можете использовать эту технику для любого цвета в диаграмме.

4 Изучите мониторы агентов и команды агентов (Agent Monitors and Agent Commanders).

Рассмотрите монитор черепашки. *Какой у нее «who» номер*; *какой цвет у черепашки*, *какая форма этой черепашки*?

В Agent Commander монитора черепашке Turtle Monitor для черепашки 0, наберите **set color pink**. *Что Вы видите*? *Что-то изменилось в мониторе черепашки*?(Turtle Monitor)?

Выделите текст справа от «color» в Turtle Monitor. Напечатайте новый цвет, как например green + 2. *Что случилось*?

В Command Center, выберите «Observer» из popup меню. Напечатайте ask turtle 0 [set color blue] и нажмите return. *Что произошло*? *Можете вызвать patch monitor и использовать его для изменения цвета отдельного пятна*?

Напечатайте ask patch -11 -4 [set pcolor blue] и нажмите return. *Какие два слова в єтой команде адресуются к пятнам*?

5 Внесите изменения в модель «Движение на дороге»: изменение формы и цвета автомобилей, добавление дома или света на улице, создание светофора, или даже создание другой полосы на дороге. В этой лабораторной работе мы сфокусируемся на простейших или косметических изменениях. Чтобы осуществлять эти простые изменения, используйте Командный центр (Command Center).

Контрольные вопросы

1 Опишите интерфейс модели «Traffic Basic».

2 Где находится «Командный центр» и для чего он предназначен?

3 Чем являются пятна и черепашки в модели?

4 Кто такой «обозреватель»?

5 Как работать с цветами в NetLogo? Как изменить цвет пятна? Как изменить цвет черепашки?

6 Что представляет собой Монитор агентов? Какую информацию там можно увидеть?

3 Лабораторная работа № 3. Моделирование динамики количества деревьев в искусственном мире

Цель работы: построить имитационную модель динамики количества деревьев в искусственной среде. Разработать компьютерную модель. Исследовать поведение системы при различных входных параметрах.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования модели динамики количества деревьев в искусственном мире.

4 Выводы.

Основные теоретические положения

Проблема сохранения лесного массива на планете в достаточном количестве является одной из важнейших задач, стоящих перед человечеством в современном мире. Бесконтрольная вырубка лесов представляет угрозу для атмосферы и жизни человека в целом. Кажется, что деревьев достаточно много, но на самом деле это далеко не так.

В данной работе предлагается имитационная модель, демонстрирующая влияние человека на количество деревьев. Известно, что с развитием цивилизации количество деревьев неуклонно снижается. Потребности населения с каждым годом растут. Возникает вопрос: «Сколько надо посадить деревьев, чтобы количество деревьев росло, а не уменьшалось?» Имитационное моделирование может помочь дать ответ на этот вопрос.

Общие положения.

Пусть в двумерном пространстве задается совокупность агентов двух типов: людей и деревьев. Первоначально количество деревьев и людей задается произвольно. Считаем, что количество деревьев меняется в зависимости от количества людей. Пусть X количество людей, а Y – количество деревьев в момент времени T. Каждый такт времени каждый человек срубает k деревьев и сажает m деревьев. То есть, количество деревьев в следующий такт можно рассчитать по формуле

$$X[T+1] = X[T] - k \cdot Y[T] + m \cdot Y[T]. \quad (3.1)$$

Дерево характеризуется своим возрастом. Известно, что, для того чтобы использовать деревья, их возраст должен быть выше некоторой величины. Будем называть такие деревья ***взрослыми*.** Обозначим ее в модели переменной ***Time*.** Считаем также, что у агентов есть **ресурс *R***. Ресурс агента каждый такт времени увеличивается на единицу, если вокруг него есть взрослые деревья, и уменьшается на единицу в противном случае. При моделировании полагалось также, что дерево живет определенное количество лет (200 лет). Агенты-люди каждый такт времени делают один шаг в случайном направлении.

В каждой точке двумерного мира может расти произвольное количество деревьев и находиться произвольное количество людей. То есть, точка может представлять собой какой-нибудь большой участок земли.

Если ресурс агента становится больше 20 единиц, то он делится. При делении агент теряет половину своего ресурса. С помощью этого мы можем проверить, насколько комфортно людям существовать в мире. То есть, если деревьев много, то население растет, и, наоборот, в противном случае.

Реализация модели в Netlogo.

Эксперимент № 1 Человек сажает меньше деревьев, чем вырубает (рисунки 3.1–3.9).

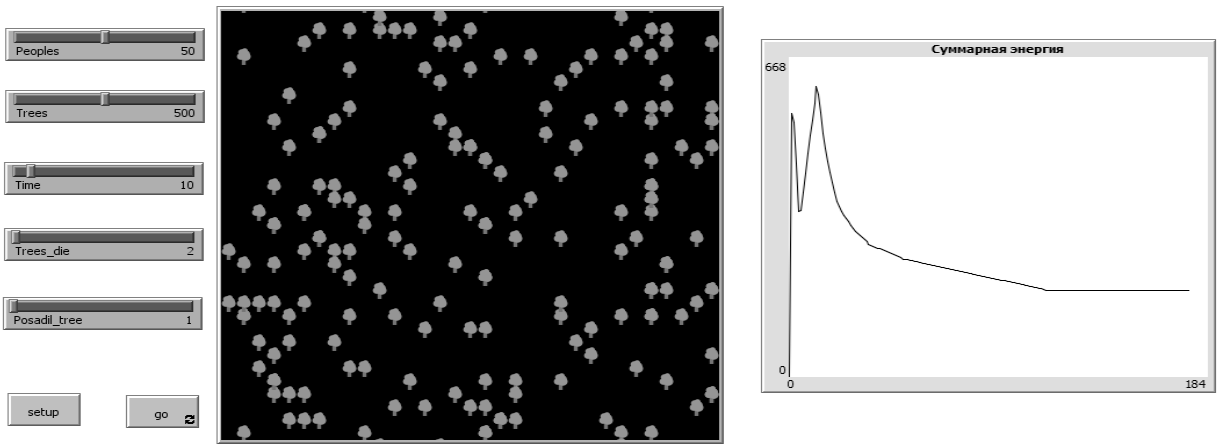

Рисунок 3.1 – Эксперимент №1

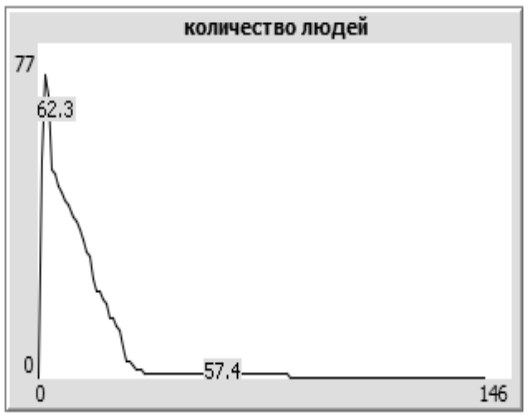

Рисунок 3.2 – Динамика количества людей

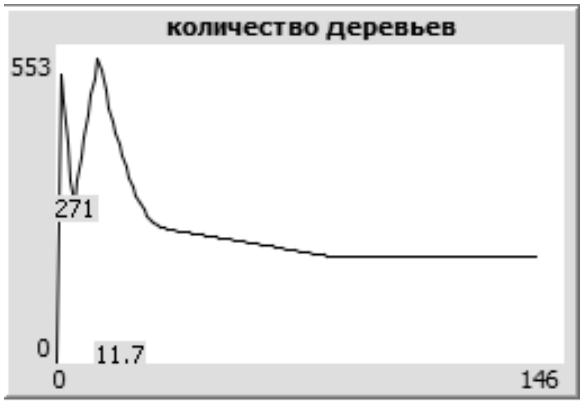

Рисунок 3.3 – Динамика количества деревьев

Рисунок 3.4 – Динамика количества взрослых деревьев

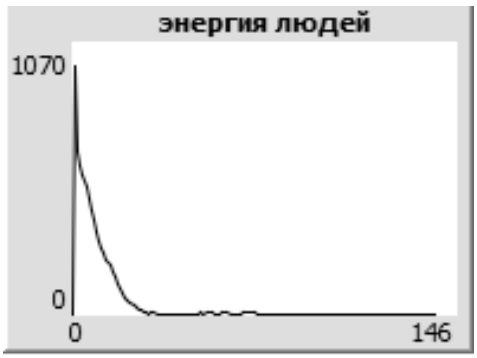

Рисунок 3.5 – Динамика энергии людей

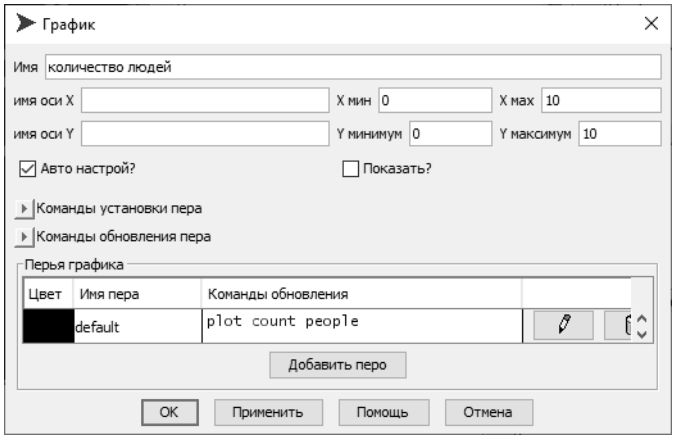

Рисунок 3.6 – Динамика количества людей

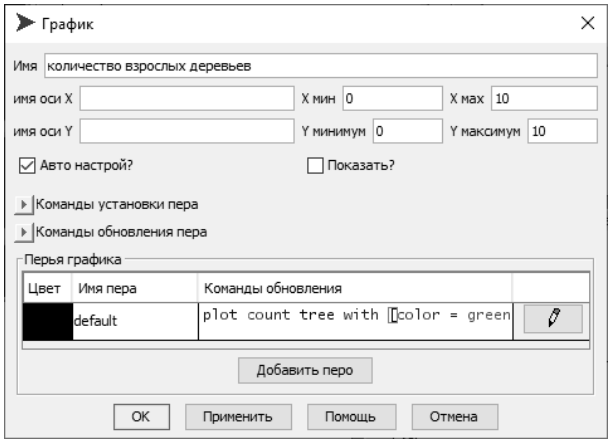

Рисунок 3.7 – динамика количества деревьев

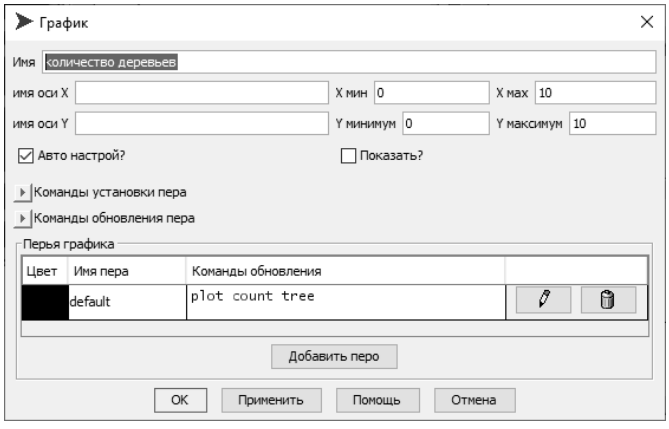

Рисунок 3.8 – Динамика количества взрослых деревьев

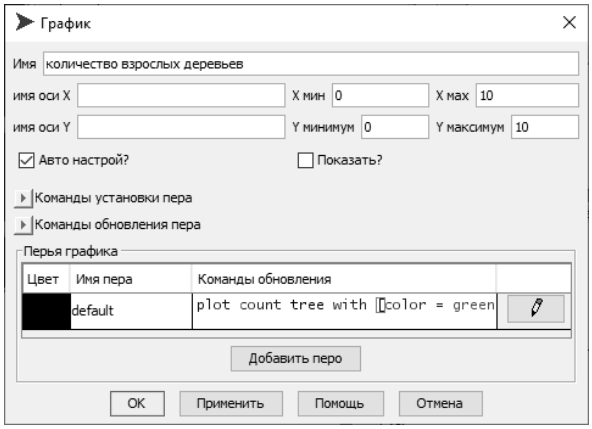

Рисунок 3.9 – динамика энергии людей

Вывод: если человек сажает меньше деревьев, чем вырубает, то очевидно, что в скором времени все агенты погибнут, что и показывает поведение модели. На 77 момент времени в мире остается всего два человека. Все графики идут вниз.

Листинг программы

```
globals [
  rost_tree
  count_die
  sum_energy
  count_posadil
        ]
breed [tree]
breed [people]
tree-own
[
  age
]
people-own
[
posadil
energy
]
to setup
 __clear-all-and-reset-ticks
 set rost_tree Time ; время, которое нужно для того, чтобы дерево считалось взрослым
 set count_die Trees_die ; время, через которое дерево погибает естественным образом
 set count_posadil Posadil_tree; количество деревьев, которое нужно посадить, прежде чем использовать их

 create-tree Trees[set shape "tree"set color green set age random(100)]

 create-people Peoples[set shape "person" set color red set posadil 0 set energy random(50)]

 ask turtles [
    set size 1.3
    setxy random-pxcor random-pycor
   ]

 ask tree [ if (age < rost_tree) [set color lime + 3]]; если возраст дерева меньше определенного порога, то это молодое дерево
 reset-ticks ;; сброс счетчиков

end


to go

  let n1  count_die * count people

  let n3 count tree with [color = green]
```

```
      if n3 > 0
     [ ifelse n1 <= count tree with [color = green]  [ask n-of
n1 tree with [color = green] [die]] [ask n-of n3 tree with [color
= green] [die]]]
      let n2  count_posadil * count people
      let pr one-of tree
      repeat n2 [ask pr [ hatch 1 setxy random-pxcor random-pycor
set age 0 set color lime + 3]]
      ask tree
      [
       set age age + 1; увеличить возраст всех деревьев на 1 год
       if (age > rost_tree) [set color green] ; если возраст де-
рева больше определенного порога, то это уже взрослое дерево
       if (age > 200) [die]; если возраст дерева больше 200 то
оно умирает
      ]
      ask people
      [
        ifelse count tree with [color = green] in-radius 1 >= 1
[set energy energy + 1][set energy energy - 1]
        if (energy <= 0) [die]
        if (energy > 20) [ set energy energy / 2 hatch 1 [set en-
ergy 0 setxy random-pxcor random-pycor set posadil 0]]
        set heading random 360

        fd 1
      ]
      tick
    end
```

Задания

1 Выполнить эксперимент, приведенный в теоретической части, в котором человек сажает меньше деревьев, чем вырубает.

2 Выполнить эксперимент при условии, что человек сажает столько же деревьев, сколько вырубает. Рассмотреть динамику изменения количества деревьев, людей и энергии.

3 Выполнить эксперимент при условии, что человек сажает больше деревьев, чем вырубает. Рассмотреть изменения динамики количества деревьев, людей, взрослых деревьев и энергии.

Контрольные вопросы

1 Как в NetLogo построить график функциональной зависимости?

2 Как в NetLogo построить несколько графиков зависимостей в одной системе координат?

3 Как в NetLogo можно управлять цветом (параметрами) графиков?

4 Лабораторная работа № 4. Моделирование динамики роста популяций животных в искусственном мире

Цель работы: изучение моделей динамики роста популяций животных в искусственном мире Netlogo.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования динамик роста популяций животных в искусственном мире Netlogo.

4 Выводы.

Основные теоретические положения

Изучение среды мультиагентного программного моделирования Netlogo на примере модели динамики роста популяций волков и овец в искусственном мире.

Рассмотрим пример задачи по компьютерному моделированию, который предлагается учащимся для исследования.

Одним из важных этапов решения задач экологии является разработка математических моделей экологических систем.

Проанализируем биологическое сообщество, которое состоит из нескольких популяций биологических видов, живущих в общей среде, и построим модель двувидовой борьбы в популяциях. Установим, что широко распространённым взаимодействием между представителями различных видов является использование одними живыми организмами («хищниками») других организмов («жертв») в качестве пищи. При этом, «соперничество» жертвы с хищником выражается в изменении численности жертвы, которая в свою очередь сказывается на численности хищника.

Создадим интерфейс, в рамках которого мы сможем регулировать основные начальные параметры системы (численность волков, численность овец, условия репродукции овец и волков, рост травы). Установим необходимые параметры в коде таким образом, чтобы волки ели овец, только если голодны. Добавим визуальные датчики количества овец, волков и травы для постоянного мониторнига ситуации (рисунок 4.1).

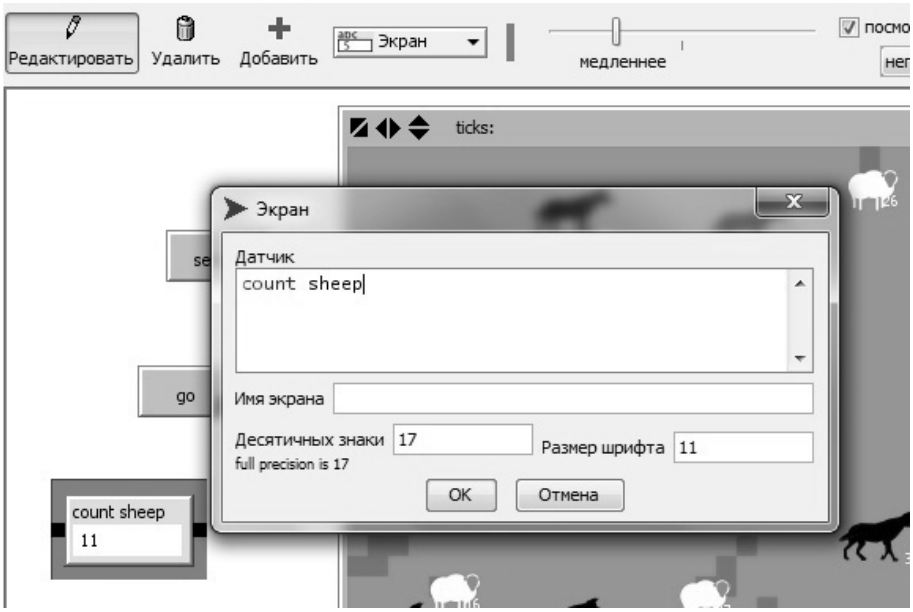

Рисунок 4.1 – Построение интерфейса

Перед написанием алгоритмического кода учащимся следует разобрать ряд команд и ключевых слов и синтаксических конструкций. Особо выделим:

1) breed [ ]

Это ключевое слово, как и глобальные переменные, и глобальные ключевые слова, может быть использовано только в начале исходного кода программы. Первое слово в команде определяет имя группы, второе слово — определяет, как называется участник группы, например [солдаты солдат]. Любой агент может принадлежать к какой-либо группе:

– он часть этой группы, будет подчиняться командам, примененным ко всей группе»;

– он обладает всеми характеристиками и переменными, относящимися к этой группе.

Используя данное ключевое слово, мы можем поделить всех агентов на группы и адресоваться именно к ним через имена групп. Например, если мы создали группы: breed [sheep a-sheep] , breed [wolves wolf], то командой create-sheep 5, мы создадим 5 агентов группы sheep, а командой create-wolves 5 мы создадим пять агентов группы wolves. Мы сможем обращаться раздельно к агентам двух групп и раздельно описывать правила их существования во вселенной Netlogo;

2) set-default-shape – установка внешнего облика агента;

3) set size – установка размера агента на экране;

4) if not any? turtles [stop] – если на экране нет ни одного агента черепашки (все типы агентов), то остановить работу процедуры;

5) hatch number [commands]

hatch- number [commands].

Этот мобильный агент создает number-количество новых мобильных агентов для своей группы. Новые агенты наследуют все свойства породившего их родителя;

6) let prey one-of sheep-here

Конструкция let prey one-of sheep-here создает переменную prey, которая соотносит себя с любым агентом типа sheep, существующим в рамках вселенной, на которую наткнулся другой агент в данной точке;

7) let variable value

Создает новую локальную переменную и присваивает ей заданное значение. Такая переменная существует только в рамках некоторого блока команд (не видна глобально во всем коде программы). Если хотите изменять значения переменной, необходимо использовать команду set. Например: set variable variable+1;

8) if prey != nobody

nobody – это особое значение, применяемое к таким примитивам, как мобильный агент, one-of, max-one-of, и т.д. Возвращается для того, чтобы указать, что ни один агент не был найден. Когда агент умирает, он становится эквивалентен nobody.

! – знак неравенства

Определим задачу моделирования: во вселенной возникнут N овец и P волков. Овцы едят траву и прибавляют свою энергию. Трава обладает своими параметрами роста. Волки едят овец (когда их встречают) и прибавляют свою энергию. При перемещениях волки и овцы теряют энергию. Волки и овцы, когда обладают большой энергией (+ фактор размножения), репродуцируют себя.

Приведем пример кода:

```
breed [sheep a-sheep]  ; группа агентов – овцы
breed [wolves wolf] ; группа агентов – волки
turtles-own [energy]       ; все агенты обладают данной переменной
; процедура установки базовых параметров
to setup
  clear-all ; очистка экрана
  ask patches [ set pcolor green ] ; окраска травы (пятна становятся зелеными)
  set-default-shape sheep «sheep» ; придание форм овцам
  create-sheep 10  ; создание овец с инициализацией параметров
  [
    set color white ; цвет агента
    set size 1.5  ; размер агента
    setxy random-xcor random-ycor; установка случайного местоположения
    set energy random 100; определение параметра энергии
  ]
  set-default-shape wolves "wolf" ; придание формы волкам
  create-wolves 10  ; создание волков с инициализацией параметров
  [
    set color black ; цвет черный
```

```
    set size 2  ; размер волка
    setxy random-xcor random-ycor; установка случайного местоположения
    set energy random 100; определение параметра энергии
   ]
   ask turtles [ set label energy ] ; обратимся ко всем агентам, установить около себя
  ; этикетку-метку с обозначением энергии
  end
  ; основная процедура действий
  to go
   if not any? turtles [ stop ] ; если исчезли все агенты — остановить работу
   ask sheep [  ; обращение к овцам
    move  ; вызвать процедуру перемещения
    set energy energy – 1 ; уменьшить энергию
    eat-grass ; вызвать процедуру поедания травы
    death ; вызвать процедуру возможной смерти
    reproduce-sheep ; вызвать процедуру репродукции овец
    ]
    ask wolves [ ; обращение к волкам
    move ; вызвать процедуру перемещения
    set energy energy – 1  ; уменьшить энергию
    catch-sheep; вызвать процедуру ловли овец
    death ; вызвать процедуру возможной смерти
    reproduce -wolves; вызвать процедуру репродукции волков
   ]
   ask patches [ grow-grass ] ; обращение к траве — вызов процедуры роста
   ask turtles [ set label energy ]       ; обратимся ко всем агентам, обновить около себя
  ; этикетку-метку с обозначением энергии
  end
   to move  ; перемещение агента
   rt random 50 ; повернуться направо (количество градусов случайно от 0 до 50)
   lt random 50; повернуться налево (количество градусов случайно от 0 до 50)
   fd 1 ; сделать шаг вперед
   end
   to eat-grass  ; поедание травы
   if pcolor = green [ ; если цвет травы зеленый
    set pcolor brown ; сделать его темным
  ; объекту, вызвавшему процедуру, увеличить энергию на 2 ед.
    set energy energy + 2    ]
   end
   to reproduce-sheep ; появление нового агента sheep
  ; новый агент появится, если у вызвавшего процедуру агента
```

```
; показатель энергии больше 30 и случайным образом
; определенное вещественное число от 0 до 100 меньше 3
  if random-float 100 < 3 and energy > 30
  [
  ;у агента, породившего нового агента, уменьшится энергия
  set energy (energy – 20)
  hatch 1 [ rt random-float 360 fd 1 ]   ; появление нового агента
 ; новый агент выбирает случайное направление и делает шаг вперед
   ]
 end
 to reproduce-wolves  ; появление нового агента wolves
; новый агент – волк появится, если у вызвавшего процедуру агента
; показатель энергии больше 30 и случайным образом
; определенное вещественное число от 0 до 100 меньше
  if  random-float 100 < 3  and energy > 30
  [
    set  energy  (energy  –  20)        ;  у  агента,  породившего  нового  агента
уменьшится энергия
     hatch 1 [ rt random-float 360 fd 1 ]    ; появление нового агента
; новый агент выбирает случайное направление и делает шаг вперед
  ]
end
to catch-sheep  ; процедура «поймать овцу»
;  переменной присваивается  агент sheep, на котором мы находимся
        let prey one-of sheep-here
if prey != nobody                  ; если такой    агент sheep найден
   [ ask prey [ die ]                   ;  мы его удаляем
     set energy energy + 10 ]   ; у агента-волка, вызвавшего процедуру,
; увеличиваем энергию
end
to death  ; гибель агента
  if energy < 0 [ die ] ; если количество энергии меньше нуля, агент исчезает
end
  to grow-grass ; процедура роста травы
; если травы не было и вычисленная случайная величина
; от 0 до 100 меньше 5
; пятно становится зеленым – то есть, появилась трава
  if pcolor = brown and random-float 100 < 5
     [ set pcolor green ]
end
```

Создадим возможность отслеживания количества овец и волков в виде графика (рисунок 4.2).

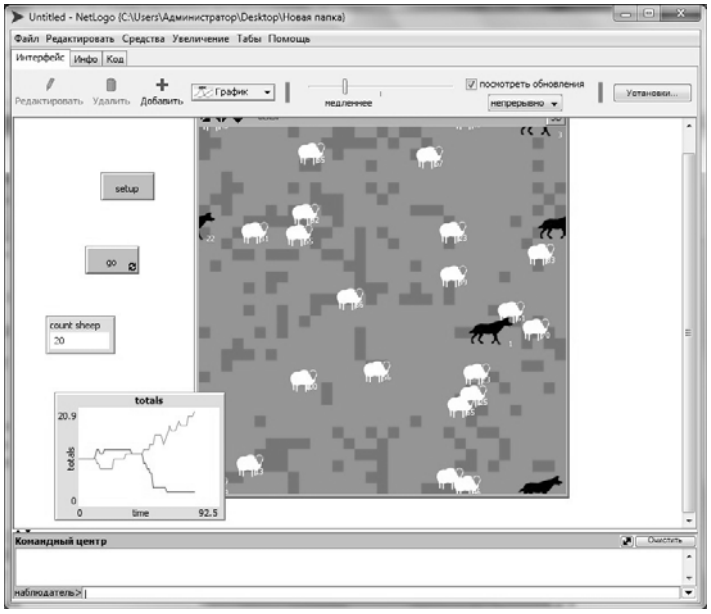

Рисунок 4.2 – Использование графиков в среде Netlogo

Опишем процедуру рисования графика

to do-plots

set-current-plot «Totals» ; имя графика

set-current-plot-pen «sheep» ; первая переменная

plot count sheep ; отражает количество овец

set-current-plot-pen «wolves» ; вторая переменная

plot count wolves ; отражает количество волков

end

Настройте вызов процедуры из цикла Go:

to go

do-plots

Установим объект — график на экране интерфейса и проверим параметры (рисунок 4.3). Удалите блок «default». Для отображения легенды установите переключатель «Показать».

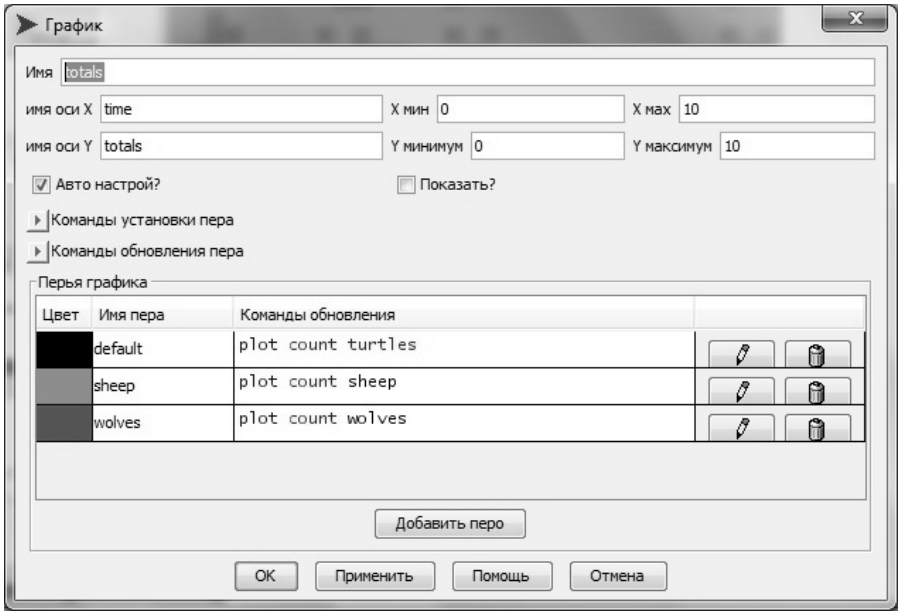

Рисунок 4.3 – Параметры графика

Таким же образом можно установить график, отражающий количество травы. После проектирования и создания компьютерной реализации модели, приступаем к ее анализу.

Алгоритм решения задачи сопровождается визуальным компонентом – вселенной Netlogo, которая отражает в виде определенной «анимации» жизнь,

поведение и взаимоотношения исследуемых агентов. Под решением исследуемой задачи мы примем экспериментальную, компьютерную проверку построенной модели, с последующим ее анализом и выведением выявленных «глобальных» закономерностей в построенной децентрализованной системе.

Задания

1 Изучить модель динамики роста популяций волков и овец в искусственном мире. Провести эксперименты над моделью при следующих исходных параметрах: а) количество овец незначительно/значительно превышает количество волков; б) равное количество овец и волков; в) количество волков незначительно / значительно превышает количество овец.

2 Изучить на выбор одну из моделей динамики роста других животных из библиотеки evolution – biology. Провести эксперименты над моделью.

3 Внесите какие-либо дополнения и изменения в изученные модели (вид популяций, анализируемые параметры моделей, внешний вид среды и др.).

Контрольные вопросы

1 Как можно построить график?

2 Какие параметры можно задавать / изменять при построении графика?

3 Как построить несколько графиков функциональных зависимостей в одной системе координат и управлять их цветами?

5 Лабораторная работа № 5. Мультиагентное моделирование методов кластеризации данных и поиска рациональных решений

Цель работы: изучение моделей алгоритмов математического анализа данных в искусственном мире NetLogo.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования моделей NetLogo.

4 Выводы.

Основные теоретические положения

Пример 1 – В NetLogo метод кластерного анализа k-means представлен моделью Sample Models/Computer Science/K-Means Clustering. Мы модифицировали данную модель.

С помощью параметра num-clusters мы задаём количество областей, в рамках которых будут располагаться точки. Num-centroids задаёт количество кластеров, которое известно заранее. Num-data-points задаёт количество точек. Количество точек должно быть больше количества кластеров.

Из рисунка 5.1 видно, что метод отработал и разбил точки по кластерам, количество которых мы задаём изначально. Изменяя количество кластеров, мы можем наблюдать их распределение на рисунке. График описывает квадратичное отклонение центроидов.

Проблемы k-means:

– не гарантируется достижение глобального минимума суммарного квадратичного отклонения V, а только одного из локальных минимумов;

– результат зависит от выбора исходных центров кластеров, их оптимальный выбор неизвестен;

– число кластеров надо знать заранее.

Для решения задачи сортировки эти три этапа выглядят так.

Сортируемый массив разбивается на две части примерно одинакового размера. Каждая из получившихся частей сортируется отдельно, например – тем же самым алгоритмом. Два упорядоченных массива половинного размера соединяются в один.

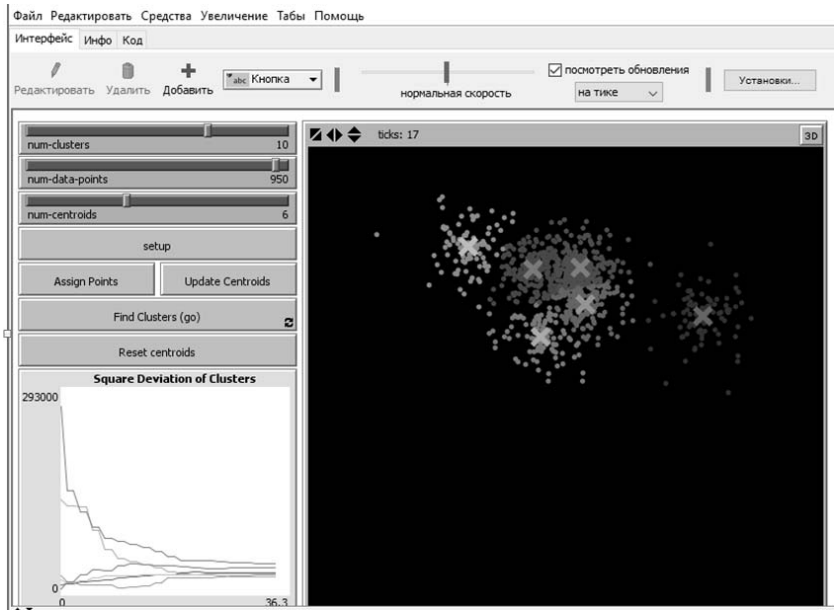

Рисунок 5.1 – Интерфейс модели кластерного анализа данных

Пример 2 – Изучение модели сортировки слиянием библиотеки моделей Netlogo (рисунок 5.2).

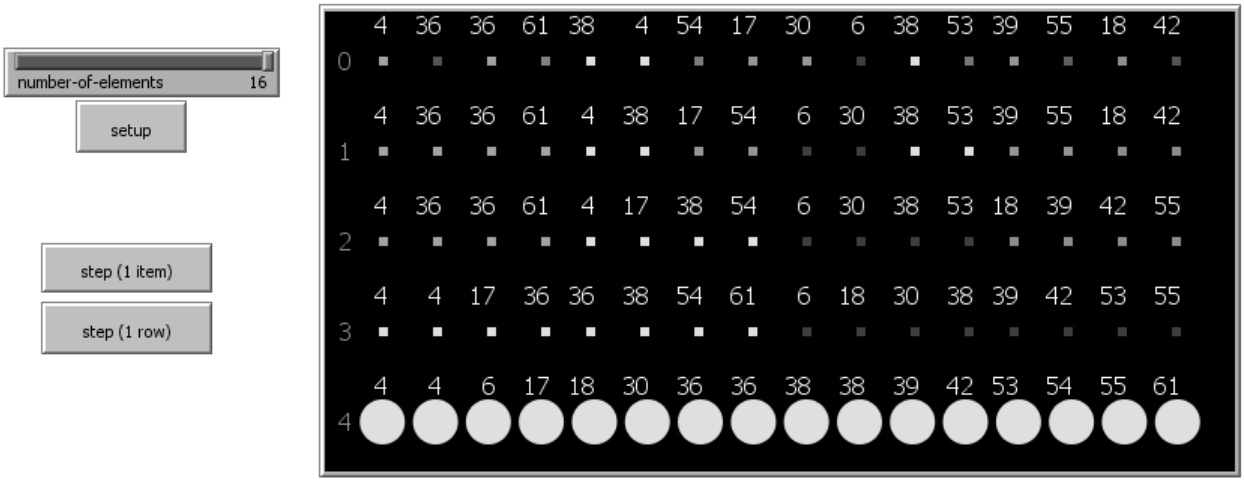

Рисунок 5.2 – Интерфейс модели решения задачи сортировки слиянием

Рекурсивное разбиение задачи на меньшие происходит до тех пор, пока размер массива не достигнет единицы (любой массив длины 1 можно считать упорядоченным).

Шаг 1. Соединение двух упорядоченных массивов в один.

Основную идею слияния двух отсортированных массивов можно объяснить на следующем примере. Пусть мы имеем два уже отсортированных по возрастанию подмассива. Тогда:

Шаг 2. Слияние двух подмассивов в третий результирующий массив.

На каждом шаге мы берём меньший из двух первых элементов подмассивов и записываем его в результирующий массив. Счётчики номеров элементов

результирующего массива и подмассива, из которого был взят элемент, увеличиваем на 1.

Шаг 3. «Прицепление» остатка.

Когда один из подмассивов закончился, мы добавляем все оставшиеся элементы второго подмассива в результирующий массив.

Пример 3 – Процедура генетического алгоритма (модель Simple Genetic Algorithm – Computer Science).

Мы начинаем со старого поколения решений. Каждое решение состоит из строки случайно смешанных бит «1» и «0». Каждое решение оценивается на основе того, насколько хорошо оно решает проблему. Эта мера решения называется его «пригодностью».

В этой модели нашей целью является просто найти решение, которое состоит из всех «1», то есть, получить новое поколение. Новое поколение решений создается из старого поколения.

Метод выбора, используемый в этой модели, называется «турнирным отбором» с размером турнира 3. Это означает, что 3 решения выбираются случайным образом из старого поколения, и выбирается решение с наивысшей пригодностью, чтобы стать родителем.

Остальная часть населения создается путем клонирования выбранных членов предыдущего поколения. Также существует вероятность того, что произойдет мутация, и некоторые дочерние биты будут изменены с «1» на «0» или наоборот. Шаги выше повторяются до тех пор, пока не будет найдено решение, которое успешно решит проблему. Для реализации данного метода каждому исследуемому решению должно соответствовать определенное значение, которое указывает, насколько близко решение подходит к искомому значению, указанное значение получается путем применения функции приспособленности. Она максимизируется.

Каждый белый столбец представляет «1» -бит, а каждый черный столбец представляет «0» -бит. Слайдер POPULATION-SIZE контролирует количество решений, присутствующих в каждом поколении. Слайдер CROSSOVER-RATE контролирует, какой процент каждого нового поколения создается в результате полового размножения (рекомбинация или кроссинговер между генетическим материалом двух родителей) и какой процент (100 - CROSSOVER-RATE) создается в результате бесполого размножения (клонирование генетического материала одного из родителей). Ползунок MUTATION-RATE контролирует процент вероятности мутации. Этот шанс применяется к каждой позиции в цепочке битов нового человека. «График пригодности» используется для отображения наилучших, средних и наихудших значений пригодности решений для каждого поколения. На «графике разнообразия» отображается величина разнообразия в совокупности решений для каждого поколения.

Исследование модели показывает следующее:

– при большом количестве популяции решение ищется быстрее;

– решение при половом размножении ищется быстрее, чем при бесполом (рисунок 5.3);

– при коэффициенте мутации, равном 0, – оптимальное решение не находится, количество новых поколений бесконечно растёт;

– при коэффициенте мутации, равном 10, – оптимальное решение не находится, количество новых поколений бесконечно растёт.

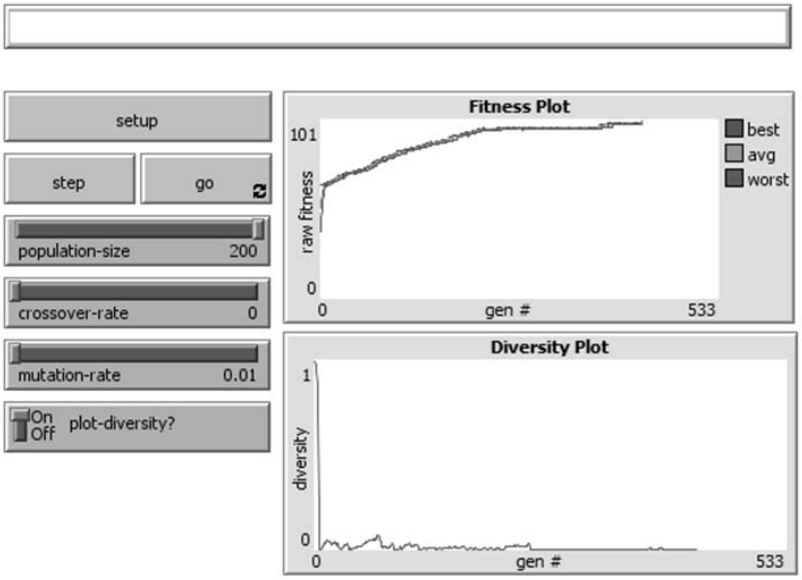

Рисунок 5.3 – Поиск решения генетическим алгоритмом

Задания

1 Изучите модель сортировки слиянием NetLogo.

2 Изучите модель K-Means Clustering - Computer Science.

3 Изучите модель Simple Genetic Algorithm – Computer Science

4 Изучите модели библиотеки моделей Fractals – Mathematics. Модифицируйте один (несколько) вид фракталов.

5 Выполните эксперименты над изученными моделями.

Контрольные вопросы

1 Для чего предназначены методы кластерного анализа?

2 Назовите входные и выходные параметры метода кластерного анализа k-means в NetLogo?

3 Какова процедура работы генетического алгоритма?

4 Что такое кроссовер в генетическом алгоритме?

5 Что такое мутация в генетическом алгоритме?

6 Лабораторная работа № 6. Мультиагентное моделирование лесного пожара и распространения вируса в человеческой популяции

Цель работы: овладеть на практике механизмом мультиагентного моделирования лесного пожара и распространения вируса в человеческой популяции.

Порядок выполнения работы

1 Изучить основные теоретические положения, сделав необходимые выписки в конспект.

2 Получить задание у преподавателя, выполнить типовые задания.

3 Сделать выводы по результатам исследований.

4 Оформить отчет.

Требования к отчету

1 Цель работы.

2 Постановка задачи.

3 Результаты исследования моделей лесного пожара и распространения вируса в человеческой популяции в искусственном мире NetLogo.

4 Выводы.

Основные теоретические положения

Пример. Модель **Вирус** (Sample Models -> Biology -> Virus).

Эта модель имитирует передачу и сохранение вируса в человеческой популяции.

Модель инициализирована на 150 человек, 10 из которых заражены. Люди случайным образом перемещаются по миру в одном из трех состояний: здоровые, но подверженные заражению (зеленый), больные и инфекционные (красный), здоровые и иммунные (серый). Люди могут умереть от инфекции или старости. Когда население падает ниже «пропускной способности» окружающей среды (в этой модели установлено 700), здоровые люди могут воспроизводить здоровых и восприимчивых потомков.

Некоторые из этих факторов суммированы ниже с объяснением того, как каждый из них рассматривается в этой модели.

Плотность населения. Плотность населения влияет на то, как часто инфицированные, иммунные и восприимчивые люди вступают в контакт друг с другом. Вы можете изменить размер начальной популяции с помощью ползунка PEOPLE.

Степень иммунитета. Если человек был заражен и выздоровел, насколько он невосприимчив к вирусу? Мы часто предполагаем, что иммунитет длится всю жизнь и гарантирован, но в некоторых случаях иммунитет со временем

исчезает, и иммунитет может быть не совсем безопасным. Тем не менее, в этой модели иммунитет действует вечно и надежен.

Инфекционность (или передаваемость). Насколько легко распространяется вирус? Некоторые вирусы, с которыми мы знакомы, распространяются очень легко. Некоторые вирусы распространяются от самого маленького контакта каждый раз. Другие (например, вирус ВИЧ, ответственный за СПИД) требуют значительного контакта, возможно, много раз, прежде чем вирус будет передан. В этой модели инфекционность определяется с помощью ползунка.

Длительность заразности. Как долго человек заражается, прежде чем он или выздоровеет, или умрет? Этот промежуток времени, по сути, является окном возможностей вируса для передачи на новые хосты. В этой модели длительность заразности определяется ползунком (рисунок 6.1).

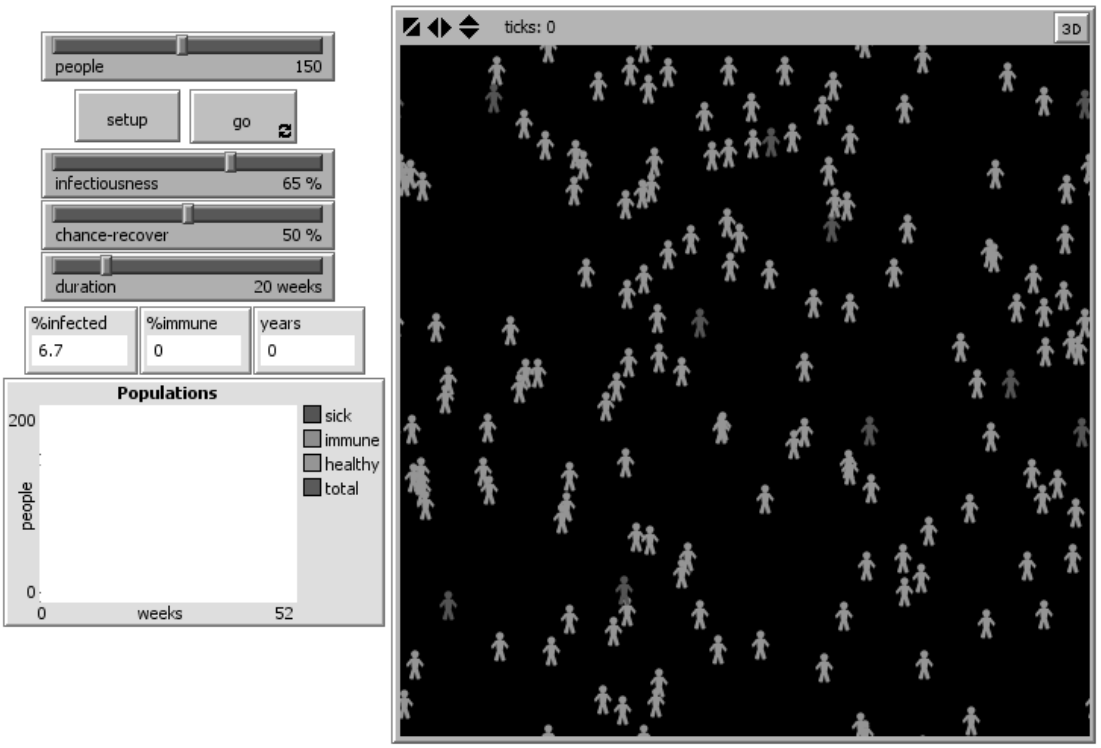

Рисунок 6.1 – Модель «Вирус»

Задания

1 Изучите описанную выше модель NetLogo Вирус (Sample Models -> Biology -> Virus).

2 Изучите модель NetLogo Огонь (Sample Models -> Earth Science -> Fire), имитирующую распространение пожара через лес.

3 Выполните эксперименты над изученными моделями.

Контрольные вопросы

1 Каковы входные параметры модели распространения вируса?
2 Каковы выходные параметры модели распространения вируса?
3 Каково влияние степени иммунитета в данной модели на итоги?

Список литературы

1 **Жандаров, В. В.** Мультиагентные системы: учебно-методическое пособие / В. В. Жандаров, М. Г. Пантелеев. – Санкт-Петербург: СПБГЭТУ «ЛЭТИ», 2016. – 35 с.

2 **Пантелеев, М. Г.** Интеллектуальные агенты и многоагентные системы: монография / М. Г. Пантелеев, Д. В. Пузанков. – Санкт-Петербург: СПБГЭТУ «ЛЭТИ», 2015. – 215 с.

3 **Потанин, М.** NetLogo: И взрослым, и детям [Электронный ресурс] / М. Потанин // Хабр. – 2014. – Режим доступа: https://habr.com/ru/post/220589/. – Дата доступа: 19.05.2020.

4 **Скобелев, П.** Мультиагентные технологии [Электронный ресурс] / П. Скобелев // Youtube. – 2015. – Режим доступа: https://youtu.be/SmRNxSKY-hU. – Дата доступа: 19.05.2020.

5 **Городецкий, В. И.** Многоагентные системы (обзор) [Электронный ресурс] / В. И. Городецкий, М. С. Грушинский, А. В. Хабалов // Сайт С. П. Курдюмова, 2015. – Режим доступа: http:// spkurdyumov.ru/ networks/ mnogo-agentnye-sistemy-obzor/ 4/. – Дата доступа: 19.05.2020.