

САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО

КАФЕДРА КОМПЬЮТЕРНЫХ СИСТЕМ И ПРОГРАММНЫХ ТЕХНОЛОГИЙ

Отчёт по лабораторной работе №5

Курс: «Многоагентные системы»

Выполнил студент:

Волкова М.Д.

Группа: 13541/2

Проверил:

Сазанов А.М.

Санкт-Петербург
2018 г.

Содержание

1	Лабораторная работа №5	2
1.1	Цель работы	2
1.2	Программа работы	2
1.3	Ход работы	3
1.3.1	Определение и краткая классификация многоагентных интеллектуальных систем. Преимущества и в чем недостатки многоагентного подхода.	3
1.3.2	Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.	4
1.3.3	Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ примеров задач, с кратким описанием.	5
1.3.4	Проанализируйте преимущества и недостатки многоагентного подхода на примере задач из предыдущего пункта.	5
1.3.5	Анализ системы NetLogo и обзор основных функциональных возможностей	5
1.4	Вывод	22
1.5	Список литературы	24

Лабораторная работа №5

1.1 Цель работы

Познакомиться с принципами работы многоагентных систем, а также изучить основные возможности основных платформ для разработки таких систем.

1.2 Программа работы

1. Дайте определение и краткую классификацию многоагентных интеллектуальных систем. В чем преимущества и в чем недостатки многоагентного подхода?
2. Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.
3. Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ примеров задач, с кратким описанием.
4. Проанализируйте преимущества и недостатки многоагентного подхода на примере задач из предыдущего пункта. Сформулируйте общие проблемы \недостатки и общие преимущества \достоинства такого подхода.
5. Проанализировать основные платформы для разработки многоагентных систем: выбрать одну из платформ (Например, NetLogo, StarLogo, Repast Simphony, Eclipse AMP, JADE, Jason либо другую) и провести обзор основных функциональных возможностей:
 - Процесс установки ПО, ссылки на сайты, ссылки на необходимые драйвера и т.д. (если нужно)
 - Процесс создания простого проекта.
 - Анализ одного примера (ссылка на пример, описание алгоритма работы и процесса моделирования и т.д.)
 - Опишите основные возможности данного ПО применительно для создания многоагентных систем. Опишите замеченные недостатки или наоборот опишите достоинства данного ПО.
6. Написать выводы. В выводах отразить, помимо своих мыслей, возникших в ходе работы, ответы на приведенные ниже вопросы:
 - В чем Плюсы и минусы многоагентного подхода?
 - Какие еще среды или языки программирования использует для создания многоагентных систем?
 - Как по-вашему стоит ли развивать данное направление, если нет, то почему, если да, то в какую сторону?
 - Корректно ли по-вашему моделирование многоагентных систем на одной вычислительной машине (рассмотреть два варианта, итеративный перебор агентом в цикле, и создание для каждого агента своего процесса)
 - Приведите области \примеры в которых применение многоагентного подхода дает максимально положительные результаты.

1.3 Ход работы

1.3.1 Определение и краткая классификация многоагентных интеллектуальных систем. Преимущества и в чем недостатки многоагентного подхода.

Многоагентные системы – это направление искусственного интеллекта, которое для решения сложной задачи или проблемы использует системы, состоящие из множества взаимодействующих агентов [1]

Агент — это аппаратная или программная сущность, способная действовать в интересах достижения целей, поставленных пользователем. Агенты описываются рядом свойств, которые характеризуют понятие агента:

- **Автономность** – способность функционировать без прямого вмешательства людей или компьютерных средств и при этом осуществлять самоконтроль над своими действиями и внутренними состояниями;
- **Реактивность** – способность воспринимать состояние среды (физического мира, пользователя – через пользовательский интерфейс, совокупности других агентов, сети Internet, или сразу все этих компонентов внешней среды);
- **Активность** – способность агентов не просто реагировать на стимулы, поступающие из среды, но и осуществлять целенаправленное поведение, проявляя инициативу.

В отличие от классической теории искусственного интеллекта, где агент (ИС) имеет в своем распоряжении все необходимые данные и вычислительные ресурсы, в теории многоагентных систем один агент владеет всего лишь частичным представлением о глобальной проблеме, а значит, он может решить лишь некоторую часть общей задачи.

Общая классификация агентов

В наиболее общем смысле, многоагентные системы принято различать по их архитектуре:

- **Делиберативная архитектура** – Подход на основе символьной модели мира, в которой решения принимаются путем логических рассуждений. Такие агенты называются агентами, основанными на знаниях. Для представления знаний используются хорошо известные в системах искусственного интеллекта методы - логический, продукционный, семантический, фреймовый [2]
- **Реактивная архитектура** – Подход основывается на том, что интеллектуальное поведение может быть обеспечено без использования символьной модели мира, путем формирования реакций агента на события внешнего мира [2]
- **Гибридная архитектура** – объединяет преимущества как делиберативной, так и реактивной архитектур.

Кроме того, в качестве общей классификации многоагентных систем следует выделить области, в которых эти системы применяются:

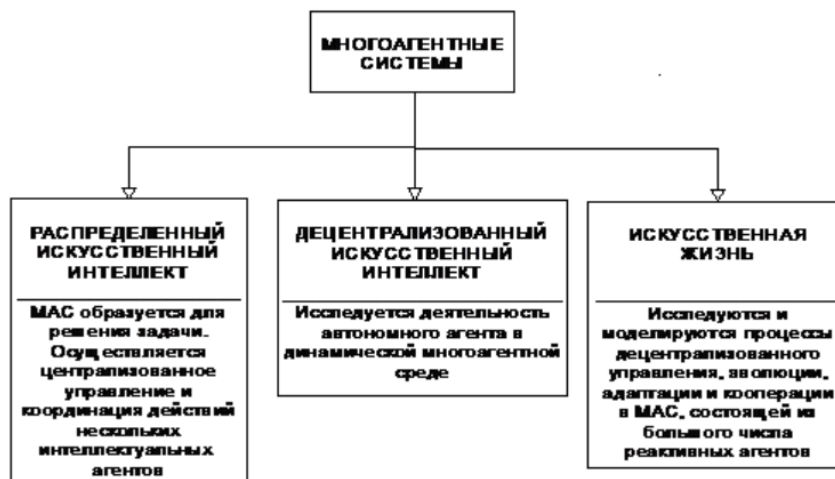


Рис. 1.1: Классификация многоагентных систем по областям применения [3]

Преимущества и в чем недостатки многоагентного подхода

Основные преимущества использования многоагентных систем, это: **распределенные вычисления, масштабирование и автономность**. Кроме того агенты могут общаться между собой, обмениваться информацией и кооперироваться для выполнения задач.

Из недостатков следует выделить усложнение структуры за счет количества агентов, что впоследствии ведет к **плохой наблюдаемости и непредсказуемости** системы.

1.3.2 Приведите наиболее полную классификацию таких систем, кратко поясните по какому признаку дана эта классификация.

Приведем наиболее подробную классификацию агентов:

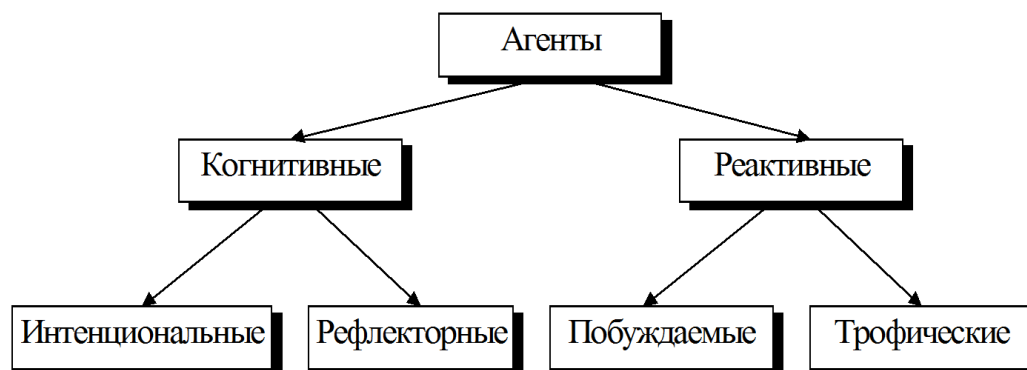


Рис. 1.2: Классификация агентов

Интеллектуальные агенты обладают хорошо развитой и пополняемой символьной моделью внешнего мира, что достигается благодаря наличию у них базы знаний, механизмов решения и анализа действий. Небольшое различие между этими типами интеллектуальных агентов связано с расстановкой акцентов на тех или иных интеллектуальных функциях: либо на получении знаний о среде, либо на рассуждениях о возможных действиях. У коммуникативных агентов внутренняя модель мира превращается главным образом в модель общения, состоящую из моделей участников, процесса и желаемого результата общения. Наконец, база знаний ресурсного агента содержит в основном знания о структуре и состоянии ресурсов, определяющих различные формы поведения [5]

Далее, по типу поведения интеллектуальные агенты делятся на **интенциональных** и **рефлекторных**. Большинство интеллектуальных (когнитивных) агентов можно отнести к числу интенциональных. Подобные агенты наделены собственными механизмами мотивации. Это означает, что в них так или иначе моделируются внутренние убеждения, желания, намерения и мотивы, порождающие цели, которые и определяют их действия. В свою очередь, модульные или рефлекторные агенты не имеют внутренних источников мотивации и собственных целей, а их поведение характеризуется простейшими (одношаговыми) выводами или автоматизмами [5]

Реактивные агенты не имеют ни сколько-нибудь развитого представления внешней среды, ни механизма многошаговых рассуждений, ни достаточного количества собственных ресурсов. Отсюда вытекает еще одно существенное различие между интеллектуальными и реактивными агентами, связанное с возможностями прогнозирования изменений внешней среды и, как следствие, своего будущего. В силу вышеуказанных недостатков реактивные агенты обладают очень ограниченным диапазоном предвидения. Они практически не способны планировать свои действия, поскольку реактивность в чистом виде означает такую структуру обратной связи, которая не содержит механизмов прогноза. Тогда как интеллектуальные агенты, благодаря богатым внутренним представлениям внешней среды и возможностям рассуждений, могут запоминать и анализировать различные ситуации, предвидеть возможные реакции на свои действия, делать из этого выводы, полезные для дальнейших действий и, в результате, планировать свое поведение [5]

По сложности этих реакций и происхождению источников мотивации реактивные агенты подразделяются на **побуждаемых** и **трофических** агентов. В случае трофических агентов поведение определяется простейшими трофическими связями. Типичной моделью подобных агентов являются клеточные автоматы, где основными параметрами выступают: радиус восприятия агента, количество условных единиц питания во внешней среде и энергетическая стоимость единицы. Между тем, побуждаемые агенты также могут иметь примитивный механизм мотивации, толкающий их на выполнение задачи, например, удовлетворение набора жизненных потребностей. Речь идет о поддержании энергетического баланса или, в более широком плане, об условиях выживания агента [5]

Таблица 1.1: Сравнительная характеристика когнитивных и реактивных агентов [5]

Характеристики	Когнитивные агенты	Реактивные агенты
Внутренняя модель внешнего мира	Развитая	Примитивная
Рассуждения	Сложные	Простые
Мотивация	Развитая	Простая
Память	Есть	Нет
Реакция	Медленная	Быстрая
Адаптивность	Низкая	Высокая
Модельная архитектура	Есть	Нет
Состав многоагентной системы	Небольшое	Большое

В соответствии с этой классификацией структурируем данные о когнитивных и реактивных агентах в виде таблицы:

1.3.3 Какие задачи решаются с помощью многоагентного подхода. Приведите не менее ДВУХ примеров задач, с кратким описанием.

Примеры использования многоагентных систем:

- Многоагентные системы широко используются при моделировании естественнонаучных и социальных процессов с целью выявления макрозакономерностей.
- Многоагентные системы хорошо зарекомендовали себя в сфере сетевых и мобильных технологий, для обеспечения автоматического и динамического баланса нагрузки, расширяемости и способности к самовосстановлению.

1.3.4 Проанализируйте преимущества и недостатки многоагентного подхода на примере задач из предыдущего пункта.

Достоинства использования многоагентного подхода при моделировании естественнонаучных и социальных процессов очевидны: как правило в этих областях моделируется большое количество различных распределенных агентов (атомы, клетки, нейроны, животные, люди и т.д.), каждый из которых имеет представление только о своей задаче и может взаимодействовать с другими агентами. Таким образом все преимущества многоагентного подхода, указанные выше справедливы и для этих областей.

Из недостатков многоагентного подхода при моделировании естественнонаучных и социальных процессов следует выделить непредсказуемость этих систем, так как наращивание количества таких распределенных агентов может быть очень быстрым в зависимости от начальных критериев.

То же самое справедливо для сферы сетевых и мобильных технологий, где количество агентов чаще всего не такое большое, но каждый из объектов обладает собственным поведением, не знает о состоянии всей системы и может взаимодействовать с другими агентами. Для данного примера не будет недостатка связанного с неконтролируемым наращиванием количества агентов, так как они чаще всего уже заданы изначально.

1.3.5 Анализ системы NetLogo и обзор основных функциональных возможностей

Среда программирования NetLogo служит для моделирования ситуаций и феноменов, происходящих в природе и обществе. NetLogo удобно использовать для моделирования сложных, развивающихся во времени систем. Создатель модели может давать указания сотням и тысячам независимых агентов действующим параллельно. Это открывает возможность для объяснения и понимания связей между поведением отдельных индивидуумов и явлениями, которые происходят на макро уровне [6]

Язык NetLogo достаточно прост и ученики и учителя могут создавать в этой среде свои собственные авторские модели. В то же время это достаточно мощный язык и среда для проведения исследовательских работ. Библиотека NetLogo содержит множество готовых моделей по биологии, математике, химии, социология. С этими моделями могут ознакомиться и поиграть ученики.

Процесс установки ПО

NetLogo распространяется бесплатно и может быть загружен с официального сайта [7] для операционных систем Windows, Linux, Mac OS. Перед скачиванием установщика предлагается заполнить следующую форму:

Download NetLogo

Most computers can run NetLogo (see [system requirements](#)). If you would like to run NetLogo on a Chromebook or in a web browser, please see [NetLogo Web](#) which meets your needs.

Multiple versions of NetLogo can be installed on the same computer; installing a new one doesn't remove the old one.

Version: [More versions here](#)
For help using old models with new versions see the [Transition Guide](#)

Name:

Organization:

E-mail:

Comments:



We read these but don't respond directly. For a response, write feedback@netlogo.cwi.nl

Download broken? Write bugs@netlogo.cwi.nl

Форма заполнения

NetLogo 6.0.4 Downloads

June 14, 2018

	Mac OS X	Download (209 MB)
	Windows (32-bit)	Download (181 MB)
	Windows (64-bit)	Download (183 MB)
	Linux (32-bit)	Download (203 MB)
	Linux (64-bit)	Download (200 MB)

Выбор ОС

После этого откроется страница, с ссылками на файл загрузчик для разных операционных систем.

Дальнейший процесс установки тривиален: для Windows и Mac OS типичный файл загрузчик, а для Linux распаковываемый архив.

Процесс создания простого проекта.

Проект может быть создан или открыт комбинациями клавиш Ctrl+N (cmd+N), Ctrl+O (cmd+O) соответственно или из меню приложения. Кроме того проект может быть импортирован из библиотеки демонстрационных проектов комбинацией клавиш Ctrl+M (cmd+M) или из меню приложения (Файл-> Библиотека моделей).

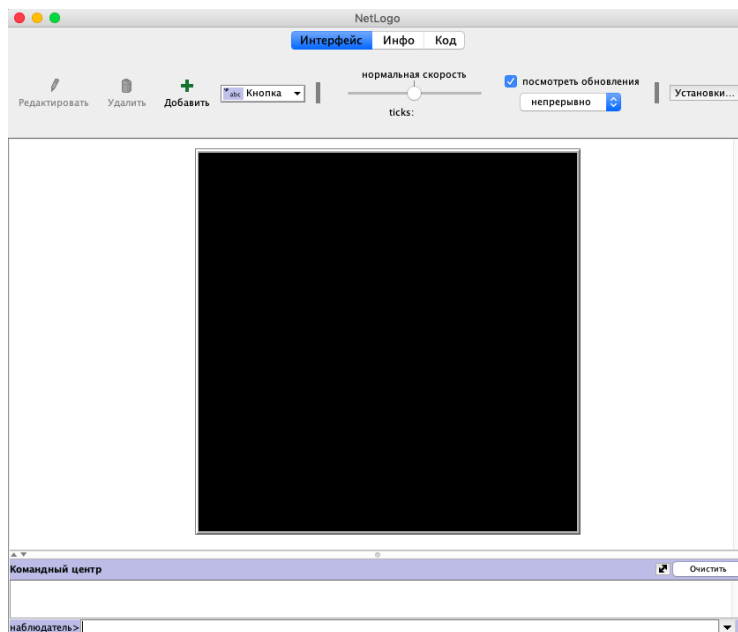


Рис. 1.3: Пустой проект

После этого не нужно предпринимать каких-либо дополнительных действий и проект будет доступен для редактирования.

Анализ одного примера

В качестве примера возьмем демонстрационный проект из сферы биологии демонстрирующий популяцию волков и овец в изолированной системе. Данный проект может быть найден в библиотеке демонстрационных проектов в папке SampleModels -> Biology -> Wolf Sheep Predation.

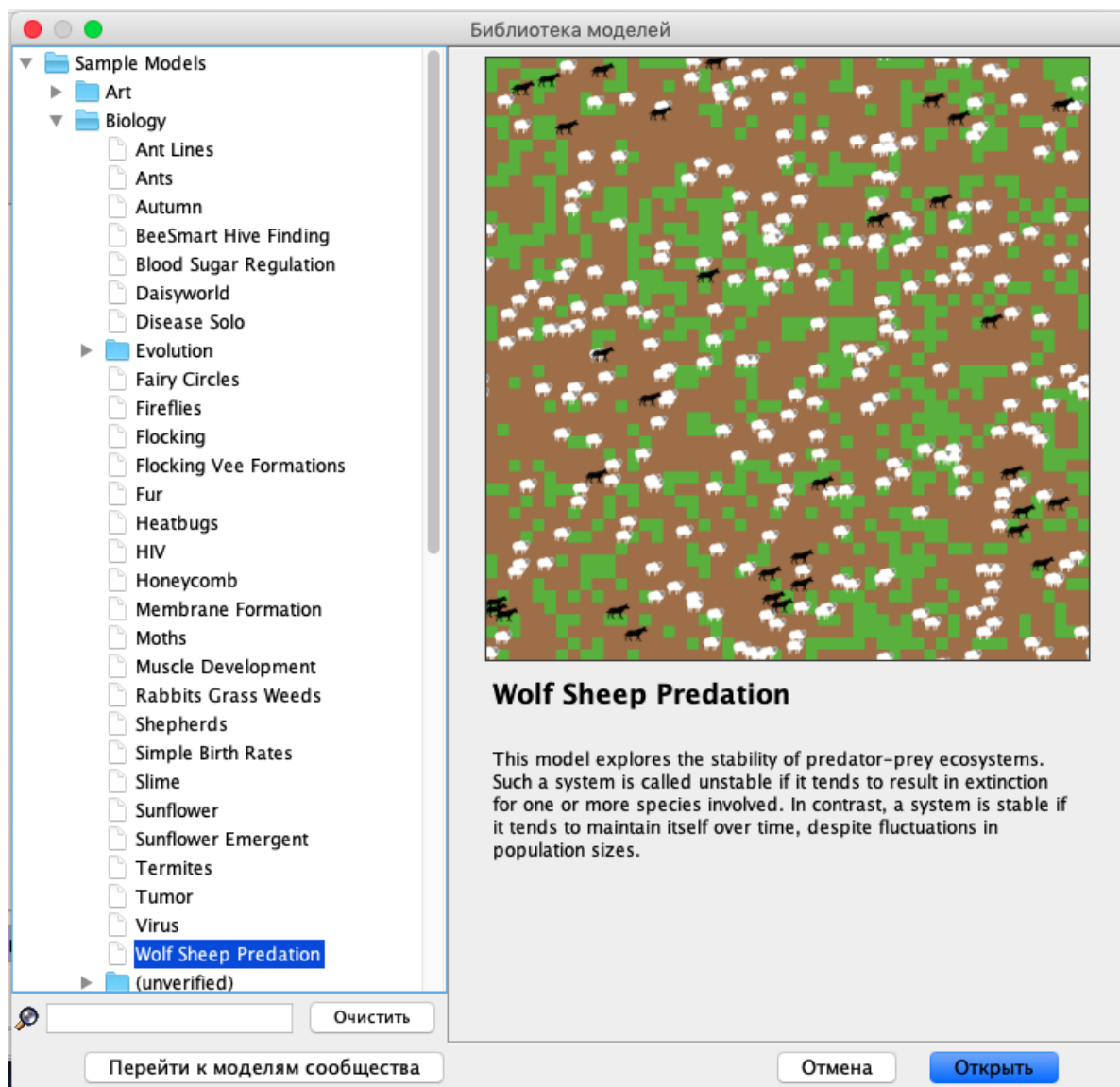


Рис. 1.4: Библиотека демонстрационных проектов

Стоит отметить, что данная библиотека имеет множество примеров использования многоагентных систем во многих научных и общественных сферах: биология, химия, искусство, психология и т.д.

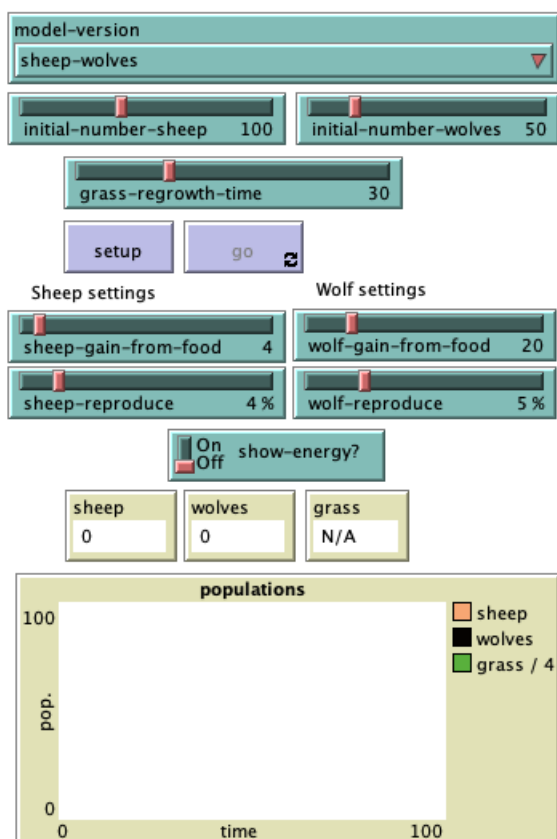


Рис. 1.5: Демонстрационный проект

После загрузки проекта в глаза бросается пользовательский интерфейс. NetLogo позволяет использовать кнопки, выпадающие списки, переключатели, рычаги и текстовые поля в качестве входных данных, а также экраны, графики, метки, диалоговые окна в качестве выходных данных.

В данном демонстрационном проекте два типа агентов: волк и овца. Эти типы агентов **побуждаемые**, так как имеют примитивный организм мотивации, толкающий их на удовлетворение собственных потребностей.

Демонстрация ведется в двух режимах:

- **sheep-wolfes** – Травы нет. У волков есть показатель энергии задаваемый при инициализации, который увеличивается на заданное значение, если волк съел овцу. Показатель энергии волка уменьшается с каждым тактом и если доходит до нуля, то волк умирает. Овца умирает только если ее съел волк. Волк или овца могут с заданным шансом воспроизвести потомство.
- **sheep-wolfes-grass** – Есть трава. У волков есть показатель энергии задаваемый при инициализации, который увеличивается на заданное значение, если волк съел овцу. Показатель энергии волка уменьшается с каждым тактом и если доходит до нуля, то волк умирает. Овца умирает если ее съел волк. У овец есть показатель энергии задаваемый при инициализации, который увеличивается на заданное значение, если овца съела траву. Показатель энергии овцы уменьшается с каждым тактом и если доходит до нуля, то овца умирает. Трава вырастает на клетках без травы с заданным интервалом. Волк или овца могут с заданным шансом воспроизвести потомство.

Инициализируем демонстрационный проект и запустим его в режиме sheep-wolfes:

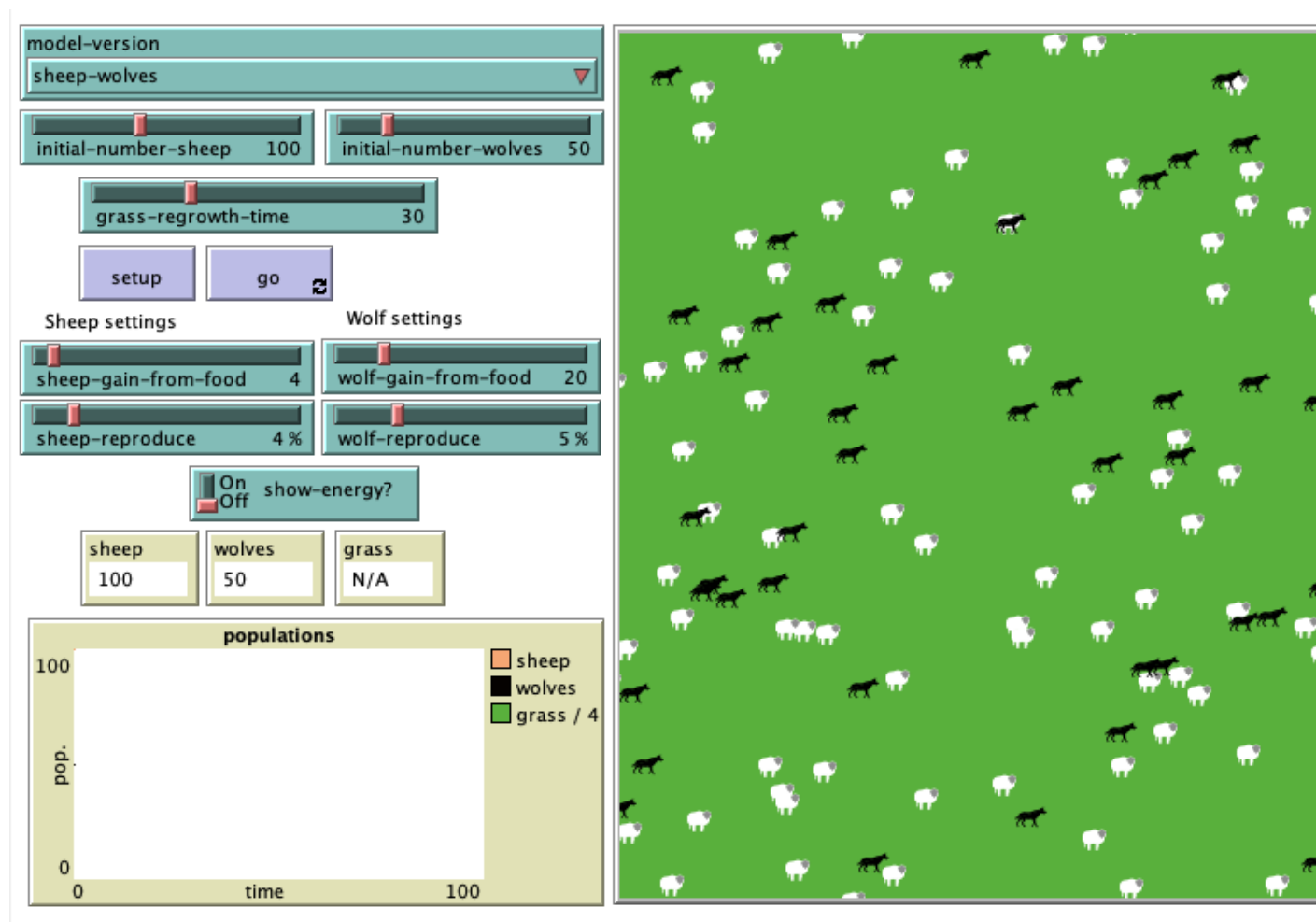


Рис. 1.6: Демонстрация в режиме sheep-wolfes

При нормальной скорости моделирования можно в режиме реального времени наблюдать, количество агентов визуально, а также при помощи графика и полей вывода статистики.

При заданных параметрах в начале моделирования овцы несколько теряют свою численность, а волков становится больше:

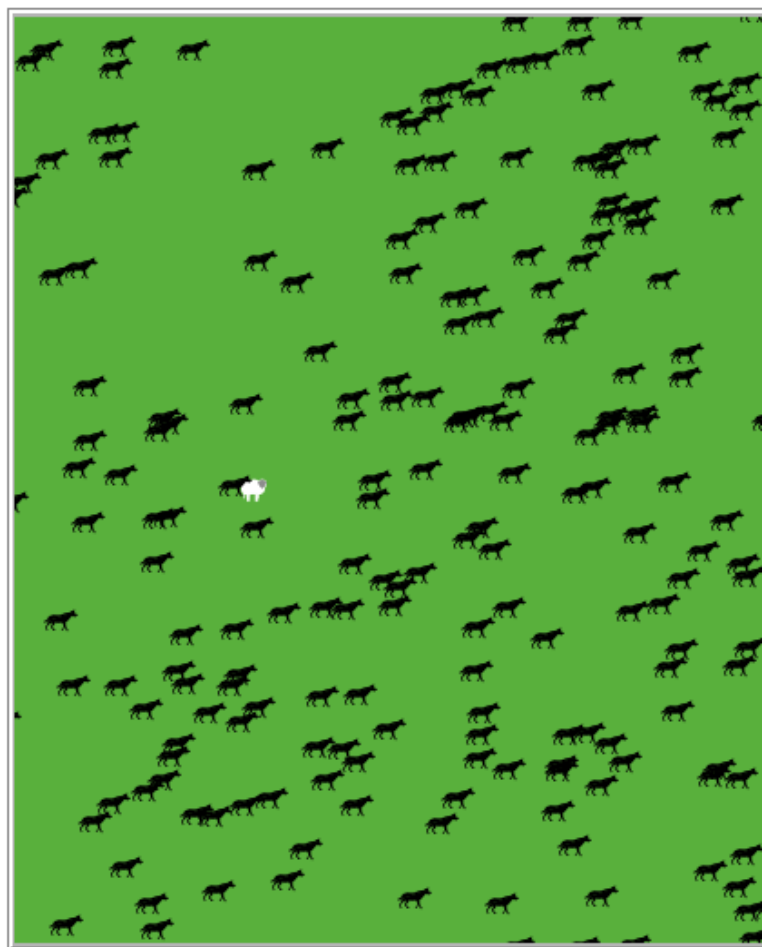
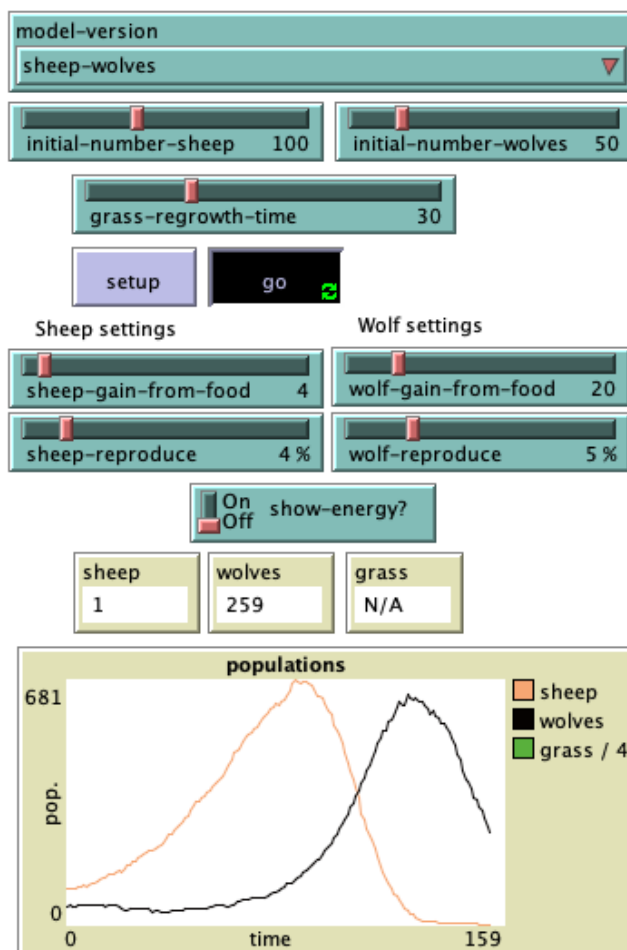


Рис. 1.7: Демонстрация в режиме sheep-wolves (промежуточный этап)

Но потом, волки вымирают и значение овец растёт до бесконечности:

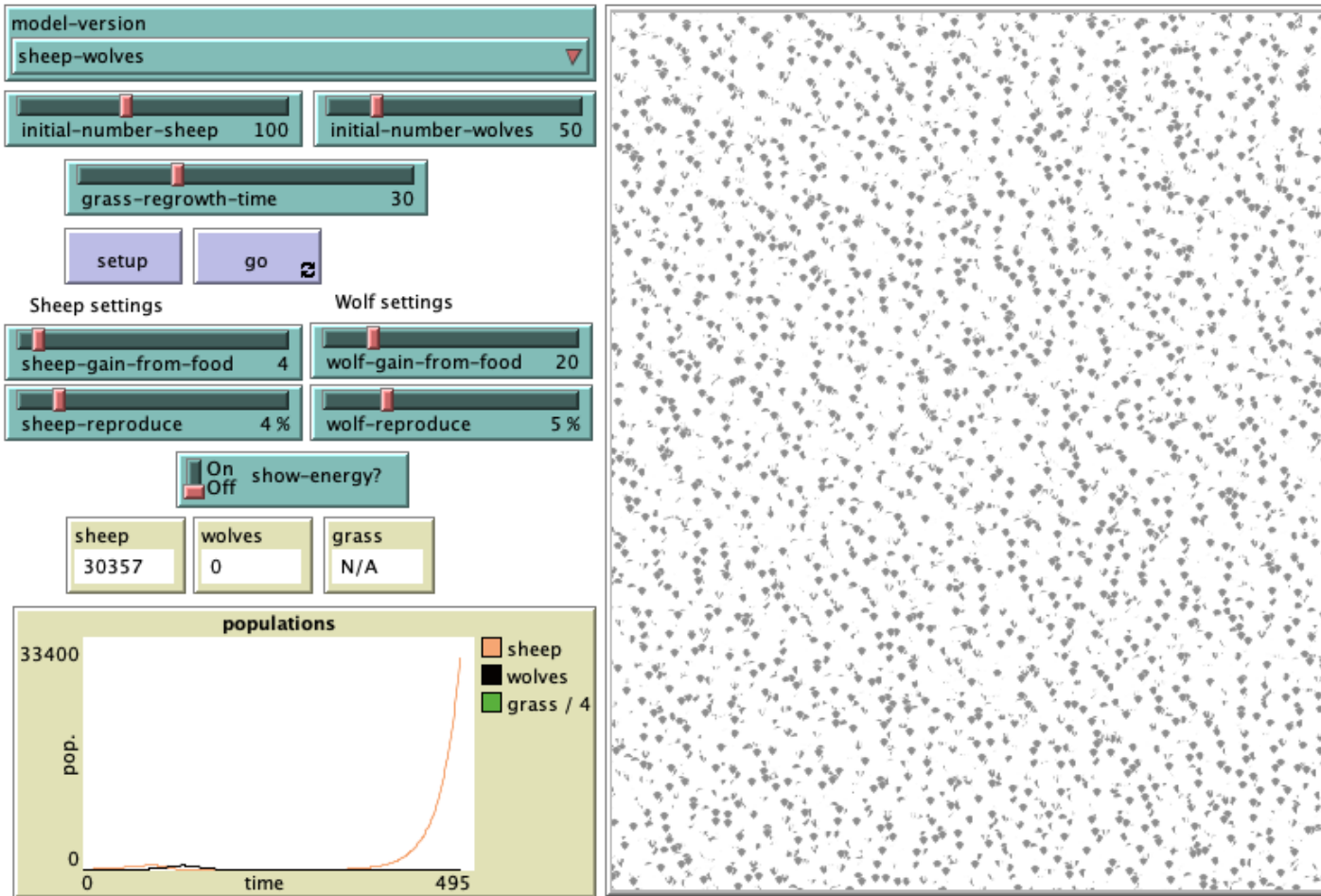


Рис. 1.8: Результат демонстрации в режиме sheep-wolves

Конец моделирования показал, что овцы бескомпромиссно задавили волков по популяции. Из этого можно сделать выводы, что промежуточные результаты не всегда так важны как макрозакономерности, которые такие системы пытаются выявить. Хотя до сих пор остается загадкой, как смогла размножить свою популяцию единственная овца.

Запустим демонстрационный проект в режиме sheep-wolfes-grass:

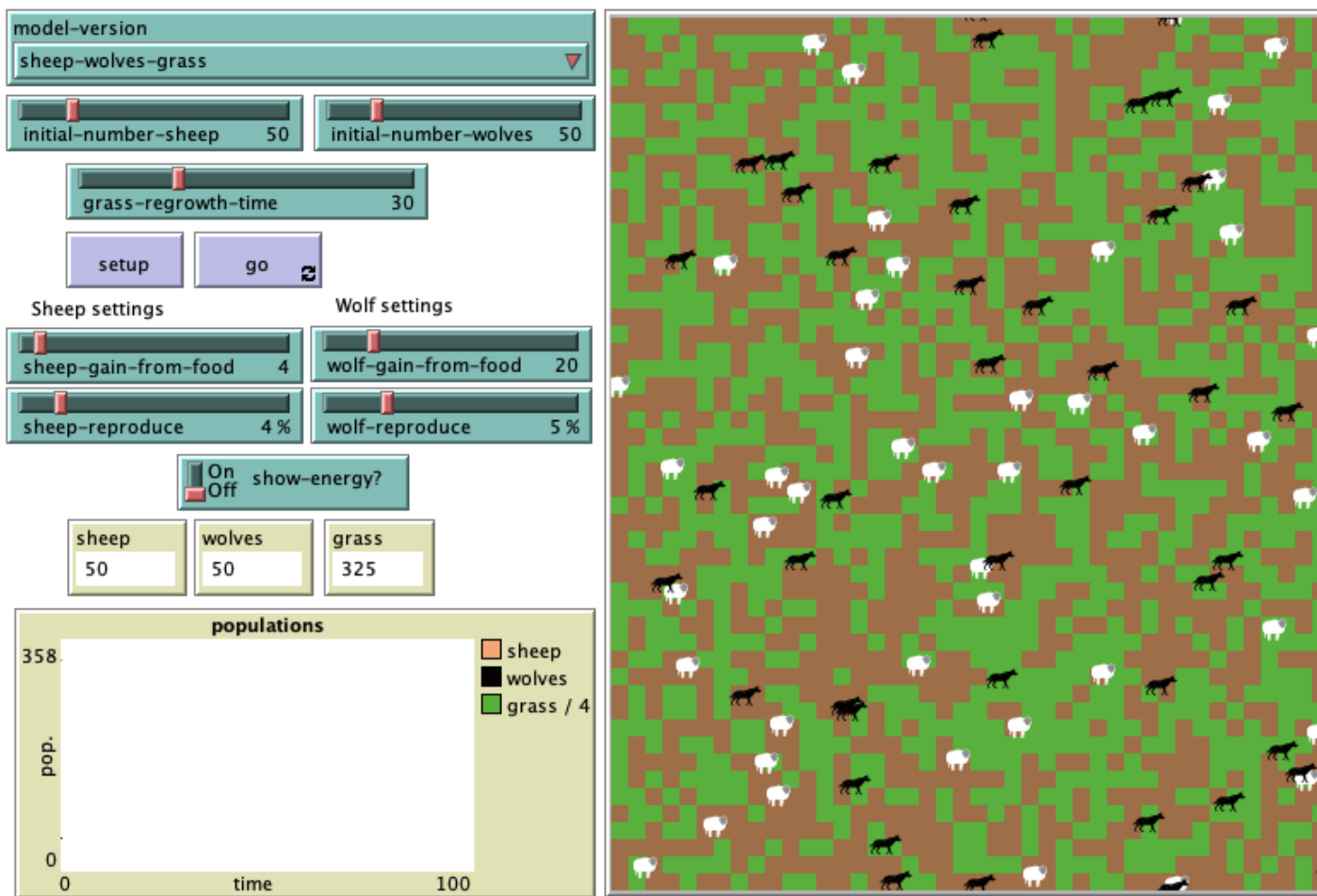


Рис. 1.9: Демонстрация в режиме sheep-wolfes-grass

Как можно заметить по графику данные о популяции овец и травы постепенно изменяются на коротком промежутке времени.

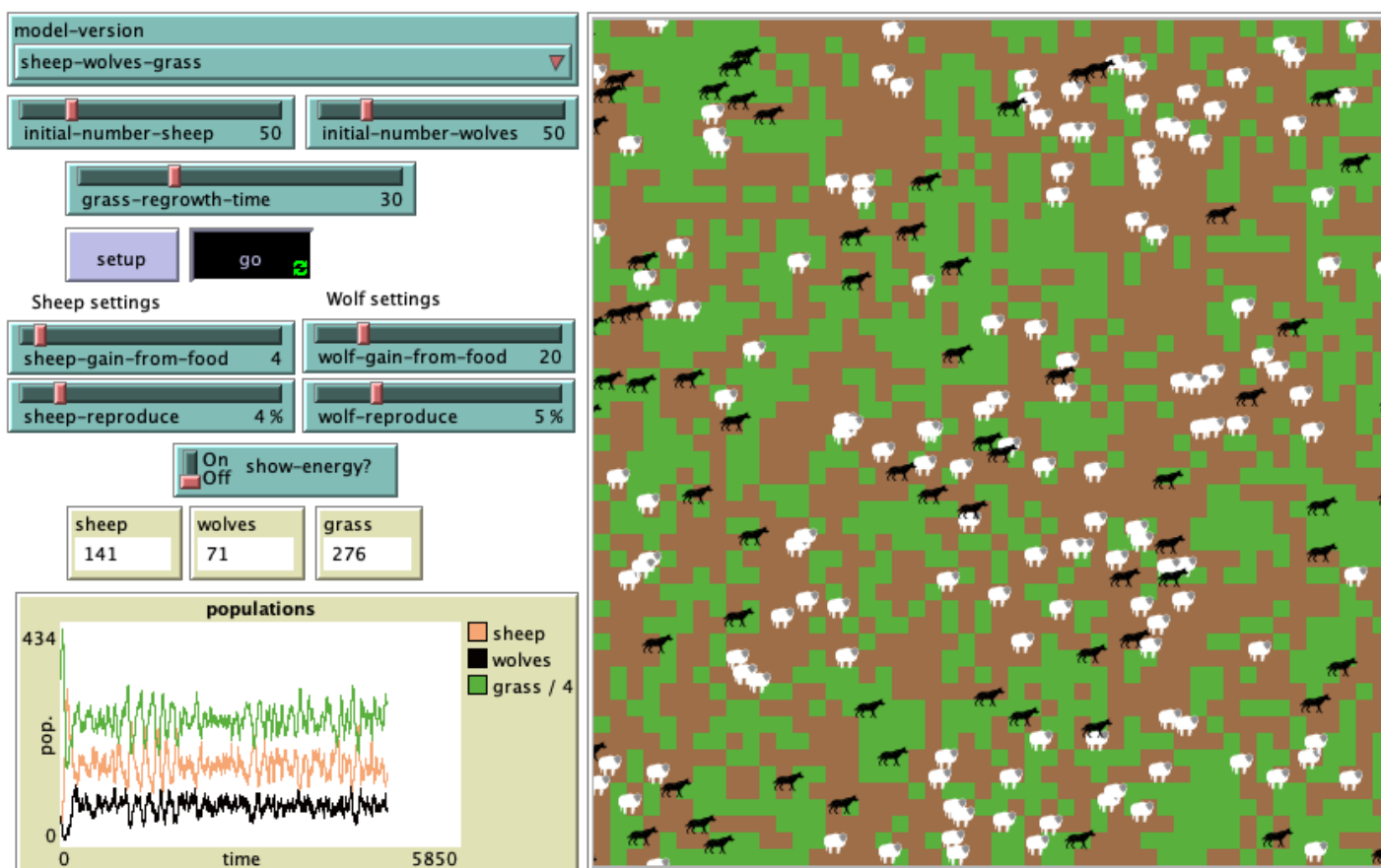


Рис. 1.10: Результат демонстрации в режиме sheep-wolves-grass

Однако, на большом промежутке времени оказалось, что ни одна из популяций победить не может, они необходимы друг другу для нормального сосуществования. Можно провести аналогию с реальным миром: волки и овцы (хоть и не в изолированной системе) за тысячи лет не уничтожили друг друга.

Хотя, запуская программу снова и снова с одинаковыми параметрами, но разными начальными условиями, можно прийти к другим исходам.

Начальное расположение:

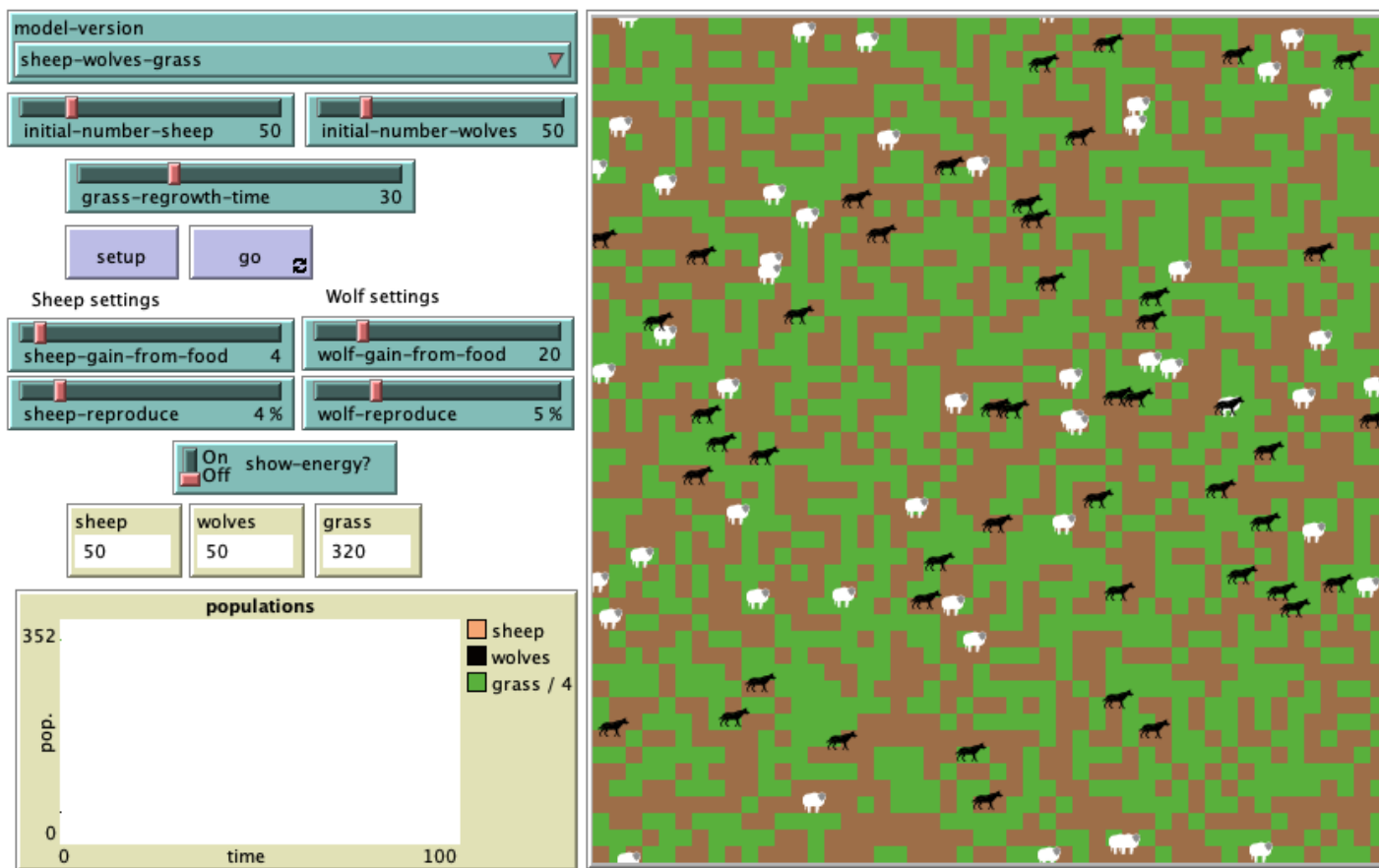


Рис. 1.11: Результат демонстрации в режиме sheep-wolves-grass

Промежуточный вариант:

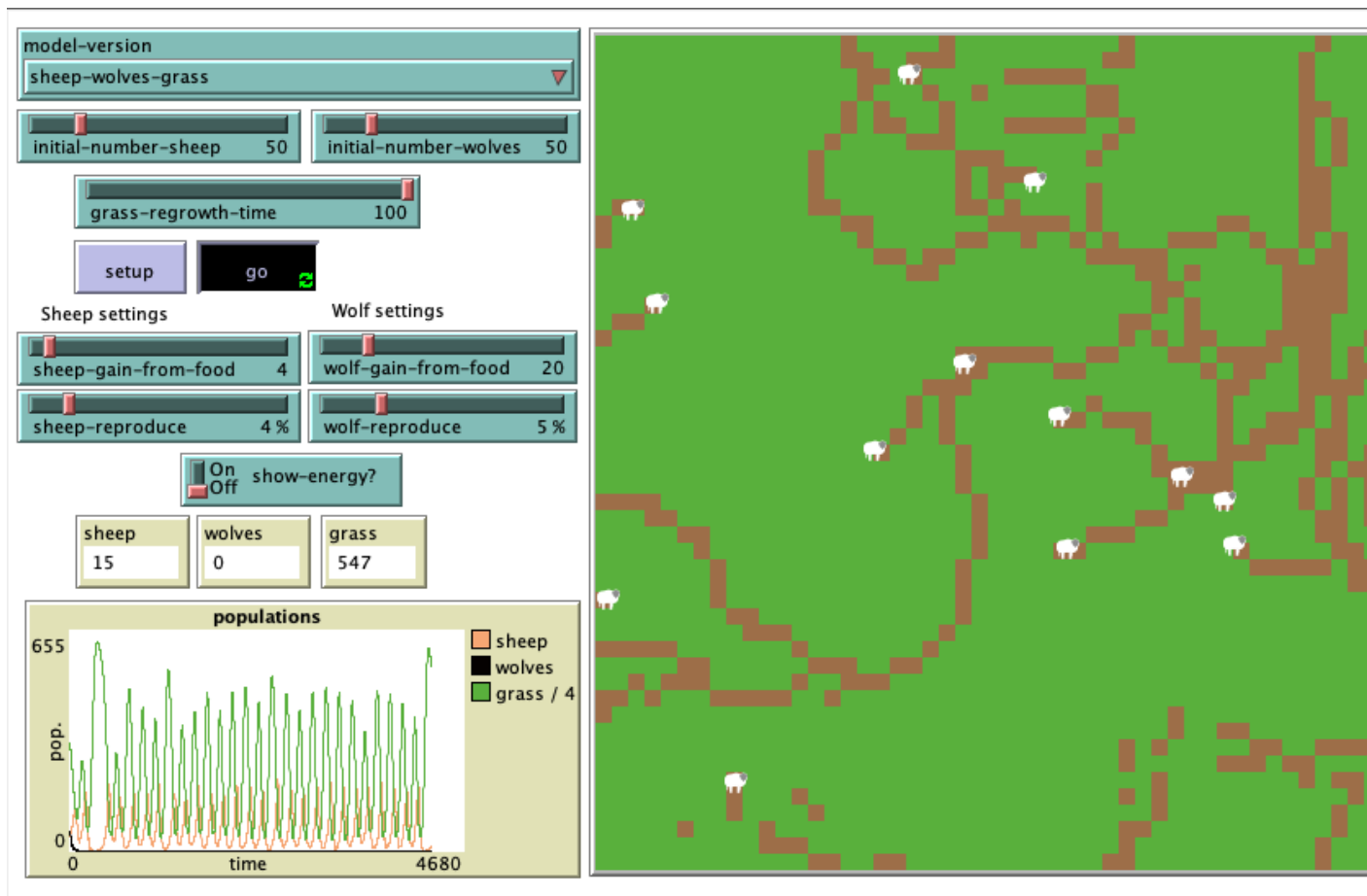


Рис. 1.12: Результат демонстрации в режиме sheep-wolves-grass

Все волки вымерли на раннем этапе, а число популяции постоянно изменяется (синусойда) в зависимости от ресурса травы.

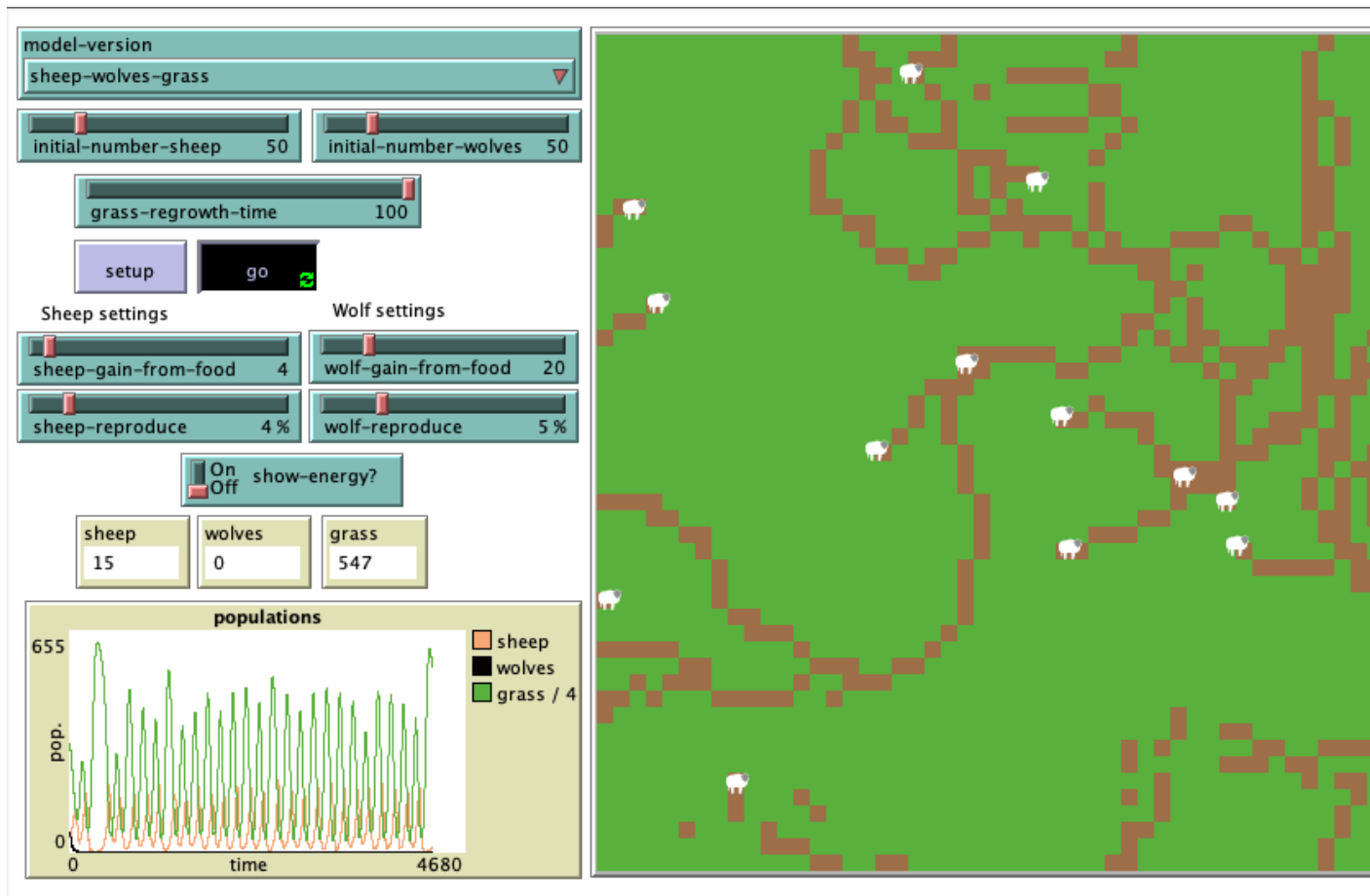


Рис. 1.13: Результат демонстрации в режиме sheep-wolves-grass

Моделирование закончилось на том, что все вымерло. Печальный исход.

Анализ кода

Код пишется на одноименном языке NetLogo, который является продолжением языка Лого – первого языка, созданного еще в 1968 году объединенными усилиями Массачусетского Технологического Института и корпорации BBN (Bolt Beranek & Newman) с целью обучать детей при помощи компьютера [6]

NetLogo является процедурным, агентно-ориентированным языком программирования, который позволяет очень лаконично описывать мультиагентные системы.

Рассмотрим исходный код инициализации агентов:

```

1 globals [ max-sheep ] ; don't let sheep population grow too large
2 ; Sheep and wolves are both breeds of turtle.
3 breed [ sheep a-sheep ] ; sheep is its own plural, so we use "a-sheep" as the singular.
4 breed [ wolves wolf ]
5 turtles-own [ energy ] ; both wolves and sheep have energy
6 patches-own [ countdown ]
7
8 to setup
9   clear-all
10  ifelse netlogo-web? [set max-sheep 10000] [set max-sheep 30000]
11
12  ; Check model-version switch
13  ; if we're not modeling grass, then the sheep don't need to eat to survive
14  ; otherwise the grass's state of growth and growing logic need to be set up
15  ifelse model-version = "sheep-wolves-grass" [
16    ask patches [
17      set pcolor one-of [ green brown ]

```

```

18     ifelse pcolor = green
19     [ set countdown grass-regrowth-time ]
20     [ set countdown random grass-regrowth-time ] ; initialize grass regrowth clocks
    randomly for brown patches
21 ]
22 ]
23 [
24     ask patches [ set pcolor green ]
25 ]
26
27 create-sheep initial-number-sheep ; create the sheep, then initialize their variables
28 [
29     set shape "sheep"
30     set color white
31     set size 1.5 ; easier to see
32     set label-color blue - 2
33     set energy random (2 * sheep-gain-from-food)
34     setxy random-xcor random-ycor
35 ]
36
37 create-wolves initial-number-wolves ; create the wolves, then initialize their
    variables
38 [
39     set shape "wolf"
40     set color black
41     set size 2 ; easier to see
42     set energy random (2 * wolf-gain-from-food)
43     setxy random-xcor random-ycor
44 ]
45 display-labels
46 reset-ticks
47 end

```

В первую очередь описываются глобальные переменные командами global, breed, turtles-own, ninjas-own. Процедуры, в том числе и обслуживающие элементы управления, описываются следующим синтаксисом:

```

to setup
  ...
end

```

Первой операцией процедуры setup является clear-all, которая очищает основной экран. Далее инициализируется константа max-sheep и инициализируются клетки с травой (в зависимости от режима).

После этого инициализируется заданное количество овец (create-sheep) и волков (create-wolf) со случайным показателем энергии и в случайных местах.

Рассмотрим код обработчика такта:

```

1 to go
2 ; stop the simulation of no wolves or sheep
3 if not any? turtles [ stop ]
4 ; stop the model if there are no wolves and the number of sheep gets very large
5 if not any? wolves and count sheep > max-sheep [ user-message "The sheep have inherited
    the earth" stop ]
6 ask sheep [
7     move
8     if model-version = "sheep-wolves-grass" [ ; in this version, sheep eat grass, grass
        grows and it costs sheep energy to move
9         set energy energy - 1 ; deduct energy for sheep only if running sheep-wolf-grass
        model version
10        eat-grass ; sheep eat grass only if running sheep-wolf-grass model version
11        death ; sheep die from starvation only if running sheep-wolf-grass model version
12    ]
13    reproduce-sheep ; sheep reproduce at random rate governed by slider
14 ]
15 ask wolves [
16     move
17     set energy energy - 1 ; wolves lose energy as they move

```

```

18     eat-sheep ; wolves eat a sheep on their patch
19     death ; wolves die if our of energy
20     reproduce-wolves ; wolves reproduce at random rate governed by slider
21 ]
22 if model-version = "sheep-wolves-grass" [ ask patches [ grow-grass ] ]
23 ; set grass count patches with [pcolor = green]
24 tick
25 display-labels
26 end

```

В первую очередь выполняется проверка на количество агентов: если все вымерли или стало слишком много, то завершаем работу. После чего вызываются процедуры движения (move) и, в случае совпадения клеток овцы и волка, съедения овцы (eat-sheep). Кроме того производится проверка на количество энергии овцы и волка, а также смерть в случае, если она равна нулю (death).

В конце процедуры вызывается процедура прорисовки травы (grow-grass) и инициируется новый такт (tick).

Рассмотрим код используемых процедур:

```

1 to move ; turtle procedure
2   rt random 50
3   lt random 50
4   fd 1
5 end
6
7 to eat-grass ; sheep procedure
8   ; sheep eat grass, turn the patch brown
9   if pcolor = green [
10    set pcolor brown
11    set energy energy + sheep-gain-from-food ; sheep gain energy by eating
12  ]
13 end
14
15 to reproduce-sheep ; sheep procedure
16   if random-float 100 < sheep-reproduce [ ; throw "dice" to see if you will reproduce
17     set energy (energy / 2) ; divide energy between parent and offspring
18     hatch 1 [ rt random-float 360 fd 1 ] ; hatch an offspring and move it forward 1
19     step
20   ]
21 end
22
23 to reproduce-wolves ; wolf procedure
24   if random-float 100 < wolf-reproduce [ ; throw "dice" to see if you will reproduce
25     set energy (energy / 2) ; divide energy between parent and offspring
26     hatch 1 [ rt random-float 360 fd 1 ] ; hatch an offspring and move it forward 1 step
27   ]
28 end
29
30 to eat-sheep ; wolf procedure
31   let prey one-of sheep-here ; grab a random sheep
32   if prey != nobody [ ; did we get one? if so,
33     ask prey [ die ] ; kill it, and...
34     set energy energy + wolf-gain-from-food ; get energy from eating
35   ]
36 end
37
38 to death ; turtle procedure (i.e. both wolf nd sheep procedure)
39   ; when energy dips below zero, die
40   if energy < 0 [ die ]
41 end
42
43 to grow-grass ; patch procedure
44   ; countdown on brown patches: if reach 0, grow some grass
45   if pcolor = brown [
46     ifelse countdown <= 0
47       [ set pcolor green
48         set countdown grass-regrowth-time ]

```

```

48   [ set countdown countdown - 1 ]
49 ]
50 end
51
52 to-report grass
53   ifelse model-version = "sheep-wolves-grass" [
54     report patches with [pcolor = green]
55   ] [ report 0 ]
56 end
57
58 to display-labels
59   ask turtles [ set label "" ]
60   if show-energy? [
61     ask wolves [ set label round energy ]
62     if model-version = "sheep-wolves-grass" [ ask sheep [ set label round energy ] ]
63   ]
64 end

```

Разбиение кода на независимые процедуры сделало его намного более понятным и простым, хотя и без этого на языке NetLogo подобные сложные системы описываются достаточно просто и интуитивно понятно.

Опишите основные возможности данного ПО применительно для создания многоагентных систем

Среда NetLogo является мощной и в то же время интуитивно понятной средой для проектирования многоагентных систем. NetLogo подходит для использования детьми (для которых и была разработана) или же серьезных ученых.

В качестве иллюстрации многогранности решаемых задач, приведем еще пару примеров.

Kicked Rotators Эта модель имитирует совокупность запущенных ротаторов, работающих одновременно, и график их траекторий. На этом графике показано разделенное фазовое пространство, в котором интегрируемые зоны устойчивости окружены хаотическим компонентом. Это так называемая стандартная карта Чирикова, инструмент в физике, который может быть использован для описания многих различных динамических систем: от движения частиц в ускорителях, до динамики комет в солнечных системах.

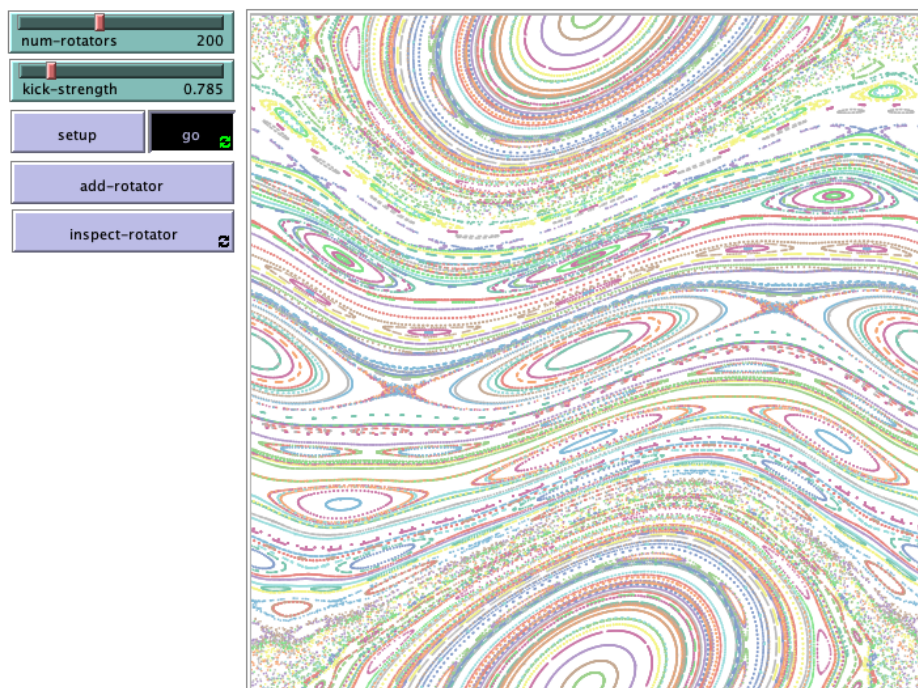


Рис. 1.14: Kicked Rotators

Diffusion Graphics Diffusion Graphics на самом деле ничего не моделирует. По сути, это просто постоянно движущаяся странная картинка.

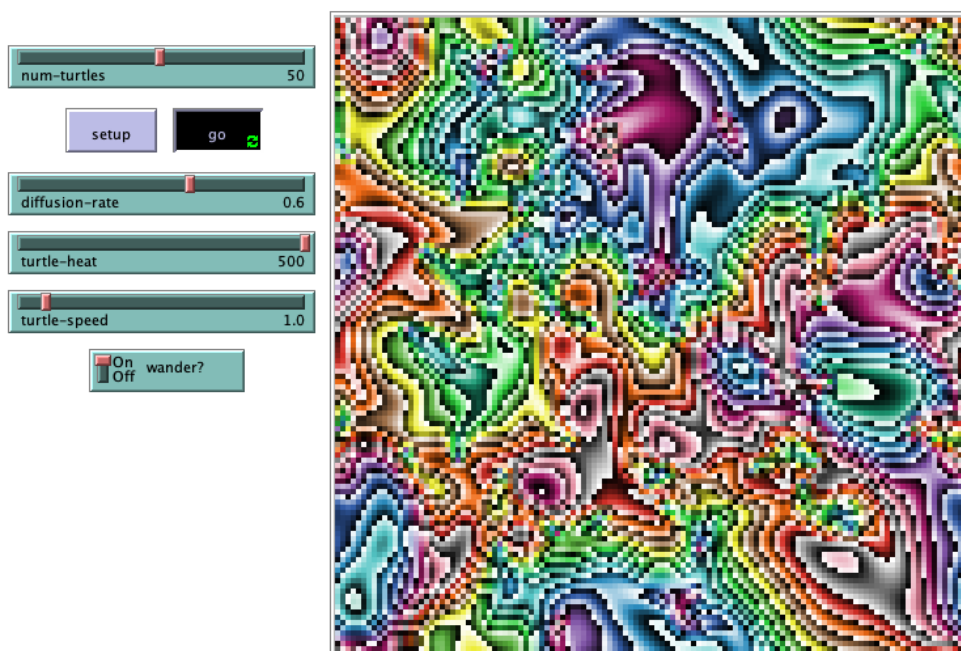


Рис. 1.15: Diffusion Graphics

Simple Genetic Algorithm Эта модель демонстрирует использование генетического алгоритма на очень простом примере. ГА работает, создавая случайную совокупность решений проблемы, оценивая эти решения, а затем используя клонирование, рекомбинацию и мутацию для создания новых решений проблемы.

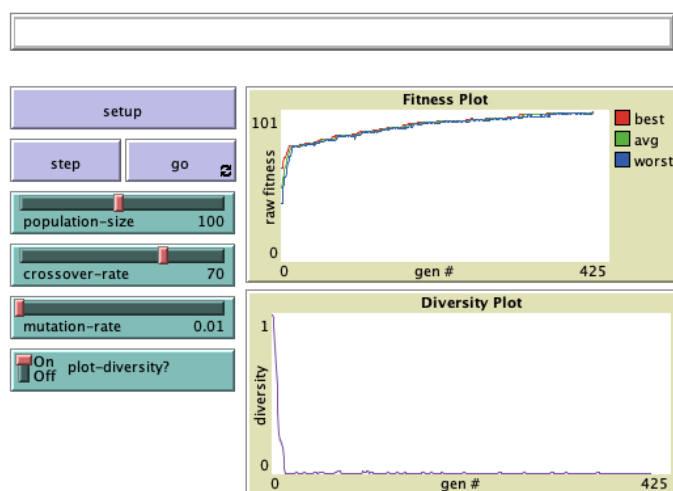


Рис. 1.16: Simple Genetic Algorithm

Climate Change Это модель потока энергии. Можно добавлять в атмосферу молекулы облаков и углекислого газа (CO_2). Молекулы CO_2 представляют собой парниковые газы, которые блокируют инфракрасный свет, излучаемый землей. Облака блокируют входящие или исходящие солнечные лучи, влияя на нагрев или охлаждение планеты.

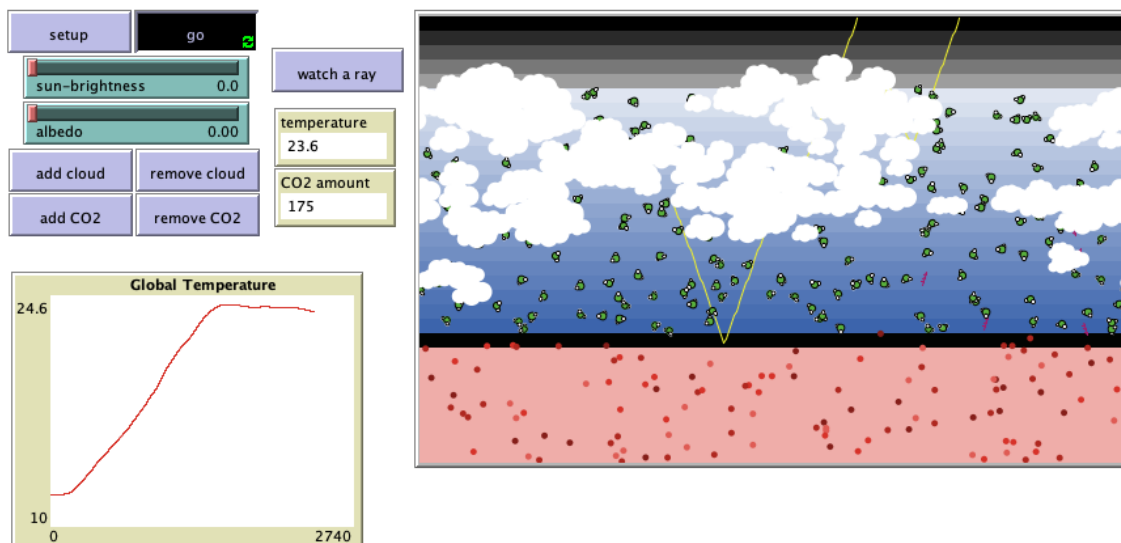


Рис. 1.17: Climate Change

Party Это модель поведения мужчин и женщин на коктейльной вечеринке относительно групп. Завсегда-таю вечеринка становится неудобным, если в их нынешней группе слишком много представителей противоположного пола, и он меняет группу.

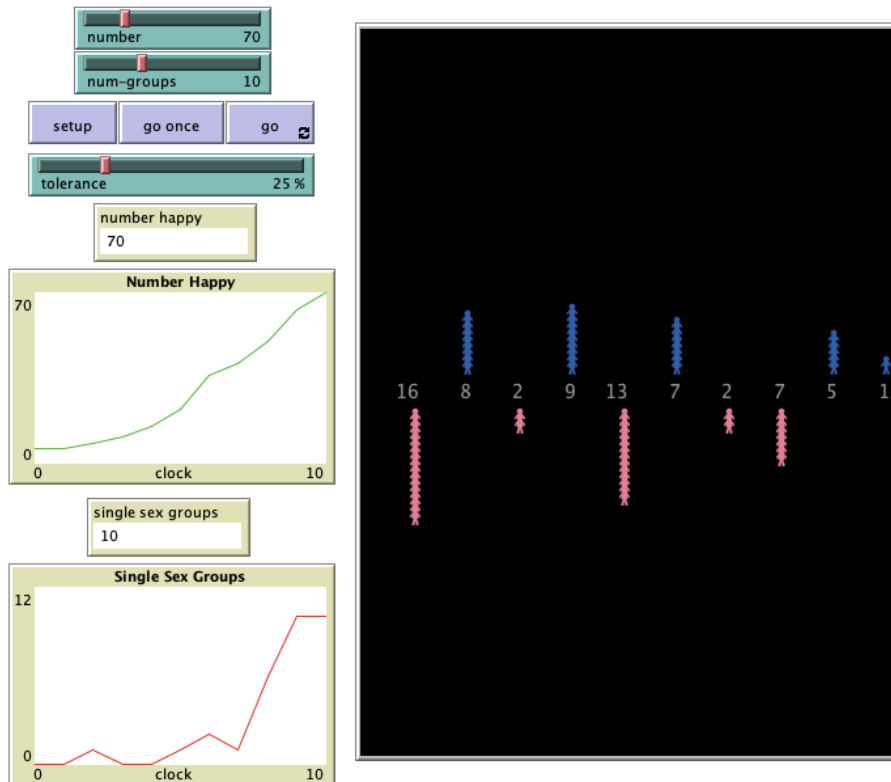


Рис. 1.18: Party

Тетрис Можно даже в тетрис поиграть:

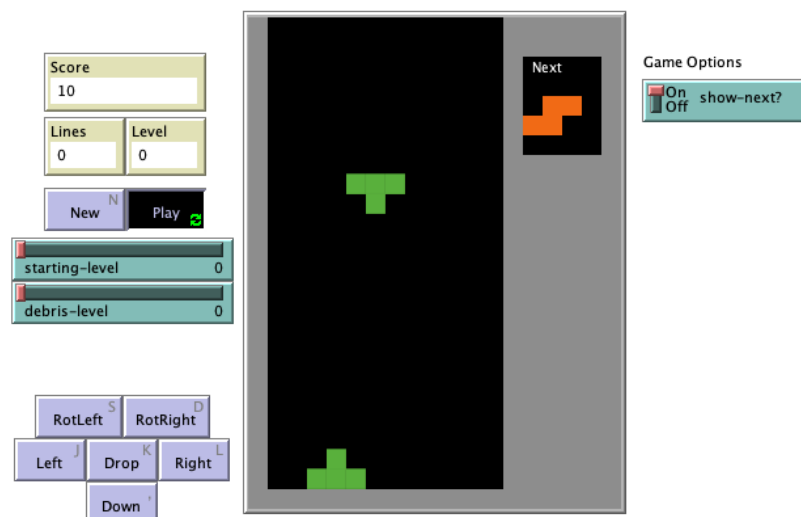


Рис. 1.19: Тетрис

Таким образом NetLogo может быть использована для самого широкого спектра многоагентных задач, от демонстрации работы генетического алгоритма до компьютерных игр.

1.4 Вывод

В ходе работы были изучены принципы работы многоагентных систем. Был произведен анализ преимуществ и недостатков данного подхода, рассмотрена классификация, а также изучена среда для разработки многоагентных систем NetLogo.

В чем Плюсы и минусы многоагентного подхода?

Основные преимущества использования многоагентных систем, это: **распределенные вычисления, масштабирование и автономность**. Кроме того агенты могут общаться между собой, обмениваться информацией и кооперироваться для выполнения задач.

Возможно, недостатком можно посчитать то, что с увеличением параметров такой системы снижается ее легкая отслеживаемость.

Какие еще среды или языки программирования использует для создания многоагентных систем?

Для разработки многоагентных системы наилучшим образом подойдут следующие программные средства: Python, Matlab, Logo, NetLogo.

Как по-вашему стоит ли развивать данное направление, если нет, то почему, если да, то в какую сторону?

Определенно стоит! Это интересно и полезно. Они прекрасно подходят для выявления макрзакономерностей в таких важнейших науках, как биолгия, физика, химия, социология и др. Поэтому, я считаю, нужно продолжать двигаться в этом направлении и постепенно увеличивать наши знания о мире во всех отраслях.

Корректно ли по-вашему моделирование многоагентных систем на одной вычислительной машине?

Как и для любого программного средства, ответ на этот вопрос зависит от сложности задачи. Для простой многоагентной системы вполне достаточно и одного компьютера, в чем мы собственно и убедились при описании программы NetLogo.

Приведите области в которых применение многоагентного подхода дает максимально положительные результаты.

- Выявление макрозакономерностей в естественных науках и социальных исследованиях.
- Исследование нагруженности сетевых и мобильных сервисов.
- Исследование предпочтений пользователей.
- Доказательство математических теорем.
- Обучение детей.

1.5 Список литературы

- [1] Многоагентные системы AI Portal [Электронный ресурс]. — URL: <http://www.aiportal.ru/articles/multiagent-systems/multiagent-systems.html> (дата обращения 18.11.2018).
- [2] ИНТЕЛЛЕКТУАЛЬНЫЕ МИКРОРОБОТОТЕХНИЧЕСКИЕ АГЕНТЫ И МНОГОАГЕНТНЫЕ МИКРОСИСТЕМЫ [Электронный ресурс]. — URL: <http://www.microsystems.ru/files/publ/205.htm> (дата обращения 18.11.2018).
- [3] Агенты, многоагентные системы, виртуальные [Электронный ресурс]. — URL: <http://pandia.ru/text/78/362/252-2.php> (дата обращения 18.11.2018).
- [4] Мультиагентные системы [Электронный ресурс]. — URL: <http://mabi.vspu.ru/portfolio/multiagentnyie-sistemyi/> (дата обращения 18.11.2018).
- [5] Классификация агентов [Электронный ресурс]. — URL: <http://www.aiportal.ru/articles/multiagent-systems/agent-classification.html> (дата обращения 18.11.2018).
- [6] NetLogo Letopisi [Электронный ресурс]. — URL: <http://letopisi.org/index.php/NetLogo> (дата обращения 18.11.2018).
- [7] Download NetLogo [Электронный ресурс]. — URL: <https://ccl.northwestern.edu/netlogo/download.shtml> (дата обращения 18.11.2018).