**Нижегородский государственный университет**

**им. Н.И. Лобачевского**

**Национальный исследовательский университет**

**Факультет социальных наук**

**Е.Д. Патаракин**

**«Изучение и моделирования организационного поведения»**

Направление подготовки 38.04.03 Управление персоналом

Квалификация (степень) «Магистр»

Профиль подготовки:

"Управление персоналом и развитие человеческих ресурсов организации"

Электронное учебно-методическое пособие

Рекомендовано методической комиссией

факультета социальных наук для студентов ННГУ,

обучающихся по направлению подготовки 38.04.03 Управление персоналом

Нижний Новгород

2020

|  |  |
| --- | --- |
| П20 | Патаракин Е.Д. «Изучение и моделирования организационного поведения»: Учебно-методическое пособие. – Нижний Новгород: Нижегородский государственный университет им. Н.И. Лобачевского, 2012 – 68 с. |

Рецензент: доктор социологических наук, профессор, зав.каф. общей социологии и социальной работы С.А.Судьин

Учебно-методическое пособие по дисциплине «Изучение и моделирования организационного поведения» разработано в соответствии с требованиями Федерального образовательного стандарта по направлению подготовки 38.04.03 Управление персоналом, квалификация (степень) «Магистр» для профиля подготовки: "Управление персоналом и развитие человеческих ресурсов организации".

В учебно-методическом пособии определены цель, задачи и формируемые компетенции дисциплины. Материал сочетает изложение теоретической части изучаемой темы, вопросы для обсуждения и задания для практической работы. Углублённому пониманию и успешному освоению предмета способствуют предлагаемые автором тесты, конкретные ситуации для решения, вопросы для самостоятельного изучения и закрепления материала. Оригинальной частью являются авторские разработки обучения многоагентному моделированию организационного поведения с использованием сред Scratch, StarLogo Nova и NetLogo.

Список литературы отражает современные проблемы использования сред многоагентного моделирования в изучении организационного поведения.

Рекомендовано методической комиссией факультета социальных наук для студентов ННГУ, обучающихся по направлению 38.04.03 - Управление персоналом.

Ответственный за выпуск:

председатель методической комиссии факультета социальных наук ННГУ, кандидат биологических наук, доцент А.В.Орлов

УДК 65.015, 303.09(094)

ББК 30ф, 60.54,60.55

**©** Нижегородский государственный

университет им. Н.И. Лобачевского, 2020

**©**Патаракин Е.Д. 2020

|  |  |  |
| --- | --- | --- |
| **1.** | **Цель и задачи освоения дисциплины «**Изучение и моделирования организационного поведения**»** | 4 |
|  |  |  |
| **2** | **Планы теоретических и практических занятий** | 6 |
|  | **Тема 1.** Введение в курс изучения и моделирования организационного поведения. | 6 |
|  | **Тема 2.** Многоагентное моделирование в среде StarLogo Nova | 12 |
|  | **Тема 3.** Многоагентное моделирование в среде NetLogo | 18 |
|  | **Тема 4.** Объяснение и понимание через моделирование и выращивание | 24 |
|  | **Тема 5.** Моделирование организационного поведения | 27 |
|  |  |  |
| **3** | **Учебно-методическое обеспечение дисциплины** | 31 |
|  | Основная литература. Дополнительная литература. Интернет-ресурсы. | 31 |
|  |  |  |
|  | Приложение 1. Руководство по агентному моделированию в среде Scratch | 32 |
|  | Приложение 2. Руководство по много-агентному моделированию в среде StarLogo Nova | 35 |
|  | Приложение 3. Руководство по моделированию в среде NetLogo | 62 |

## Место дисциплины «Изучение и моделирования организационного поведения» в структуре основной профессиональной образовательной программе (ОПОП) «Управление персоналом организации».

Дисциплина относится к вариативной части профессионального цикла. Трудоемкость дисциплины составляет две зачетные единицы.

**Цель освоения дисциплины**:

формирование у студентов способности и готовности использовать в управлении персоналом информационные технологии и методы агентного моделирования, формирование у студентов способности и готовности использовать многоагентные модели для исследования организаций путем их «выращивания» в среде компьютерных моделей.

**Задачи освоения дисциплины:**

сформировать и развить знание ключевых понятий («агент», «актор», «многоагентная система», «модель», «генеративная социология», «вычислительная социология», «вычислительное управление»);

изучить и сравнить отечественные и зарубежные подходы к построению организационных моделей, моделированию деятельности и управлению персоналом;

сформировать навыки и умения представлять взаимодействия внутри трудовых коллективов в виде отношений агентов, относящихся к различным породам;

освоить возможности современных сред моделирования коллективного поведения;

сформировать навыки и умения выявлять базовые поведенческие паттерны агентов, приводящих к формированию сложного коллективного поведения;

сформировать навыки и умения создания многоагентных моделей, позволяющих анализировать и предсказывать коллективное поведение;

изучить критерии и сформировать навыки и умения оценки эффективности системы организации совместной деятельности через построение динамических моделей совместной деятельности.

В результате освоения дисциплины магистрант должен:

**Знать:**

основные понятия многоагентного моделирования и науки об искусственных обществах;

основные методы имитационного моделирования, в частности методы агентного моделирования и др.;

приложения многоагентного моделирования к управлению персоналом;

показатели модели, позволяющие оценивать совместную деятельность персонала.

**Уметь**:

осуществлять анализ деятельности агентов на индивидуальном уровне;

разрабатывать алгоритмы индивидуального поведения агентов;

моделировать «среду обитания агентов»;

строить модели организационной деятельности в многоагентной среде;

интерпретировать данные, получаемые в результате моделирования деятельности множества агентов.

**Владеть:**

навыками использования многоагентных моделей для описания и объяснения поведения организаций;

навыками моделирования отношений, складывающимися между участниками совместной деятельности;

навыками обсуждения организационного поведения с привлечением аппарата многоагентного моделирования;

навыками разработки имитационных моделей в средах StarLogo Nova, NetLogo

**Содержание дисциплины**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Наименование и краткое содержание тем дисциплины** | Контактная работа | | | Сам. Работа | Форма текущего контроля |
| Всего | Лекции | ПЗ |
| Тема 1. Введение в курс изучения и моделирования организационного поведения | 4 | 2 | 2 | 10 | Самостоятельное создание моделей |
| Тема 2. Многоагентное моделирование в трехмерной среде StatLogo Nova | 4 | 2 | 2 | Самостоятельное создание моделей |
| Тема 3. Многоагентное моделирование в среде NetLogo | 6 | 3 | 3 |
| Тема 4. Объяснение и понимание через моделирование и выращивание | 6 | 3 | 3 | Тестовые задание по теме |
| Тема 5. Моделирование организационного поведения | 6 | 3 | 3 | 10 |
| Тема 6. Создание динамических карт на основании данных | 6 | 3 | 3 | Самостоятельное создание моделей в среде NetLogo |
| ИТОГО | 32 | 16 | 16 | 20 |  |

# Планы теоретических и практических занятий

## Тема 1. Введение в курс изучения и моделирования организационного поведения (выращивание сложных социальных систем)

**Цель изучения темы:** обосновать возможности выращивания искусственных сообществ и показать необходимость моделирования для доказательного и вычислительного управления персоналом.

**Теоретическое введение** (Лекция – 2 часа)**.** Понятия «Модель», «агент», «многоагентная модель», «модель организации». Организационное поведение, как результат взаимодействия простых паттернов поведения множества агентов - акторов. Возможности объяснения и понимания через выращивание феноменов в компьютерной среде. Управление организацией через различные формы компьютерного и живого агентного моделирования. Моделирование (выращивание) сложных систем. Работы зарубежных учёных ‑ T. Schelling, T.  Axelrod, J. Epstein и др. Работы российских ученых В.Л. Макарова, А.Р. Бахтизина и др. в области применения агентного моделирования в социально-экономических науках.

**Литература:**

1. Патаракин Е. Д. Агентное моделирование для рефлексии образовательной организации // Искусственные общества. 2018. T. 13. Выпуск 4 [Электронный ресурс]. URL: <http://artsoc.jes.su/s207751800000133-5-1>
2. Парандекар C, Е. Патаракин, Яйла Г. Обучение детей программированию: залог развития человеческого капитала в XXI веке. Руководство для российских законодателей и практиков в области образования / – Москва : Алекс (ИП Поликанин А.А.), 2019. – 164 с. <http://www.uic.unn.ru/pustyn/papers/WB%20Children%20learning_Ru.pdf>
3. Патаракин Е.Д. Учимся готовить в среде Скретч https://db.ph-int.org/upload/iteach/texts/pi\_2010\_06\_30-19\_26\_40\_1.pdf
4. Патаракин Е.Д. Педагогический дизайн социальной сети Scratch // Образовательные технологии и общество (Educational Technology & Society). 2013. № 2. P. 505–528. https://elibrary.ru/item.asp?id=19002846

**Первоисточники:**

* Epstein J.M., Axtell R. Growing Artificial Societies: Social Science from the Bottom Up. Washington, DC, USA: The Brookings Institution, 1996.

**Практическое занятие (2 часа).**

Программное обеспечение и сетевые сервисы

Scratch.mit.edu и другие примеры компьютерных сред для создания искусственных миров, населенных агентами.

**Вопросы для обсуждения и задания для практической работы**

**Пример 1**

На примере модели «Эффект бабочки» <https://scratch.mit.edu/projects/293659/>

в среде Scratch рассмотрите основные понятия агентного моделирования:

1. Автономия и отсутствие управляющей структуры. Агенты действуют независимо друг от друга. На макроуровне задается общий набор правил.
2. Неоднородность агентов по каким-либо признакам.
3. Ограниченность агентов по знаниям и умениям.
4. Расположение агентов в пространстве

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Мир "Scratch"Агенты на поле

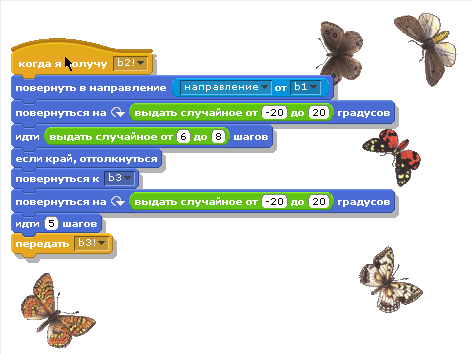


Рисунок Процедура, управляющая поведением агента-бабочки

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Управление через действия и сообщения

* Создайте ремикс проекта <https://scratch.mit.edu/projects/293659/>
* Измените модель, увеличив число взаимодействующих агентов.
* В Scratch нет понятия породы агентов и каждого нового агента приходится создавать заново. Но, здесь есть возможность создавать клоны. Попробуйте написать вариант модели, который бы начинался с одного агента-бабочки.

### Пример 2 Записываем историю взаимодействия

<https://scratch.mit.edu/projects/1314960/>

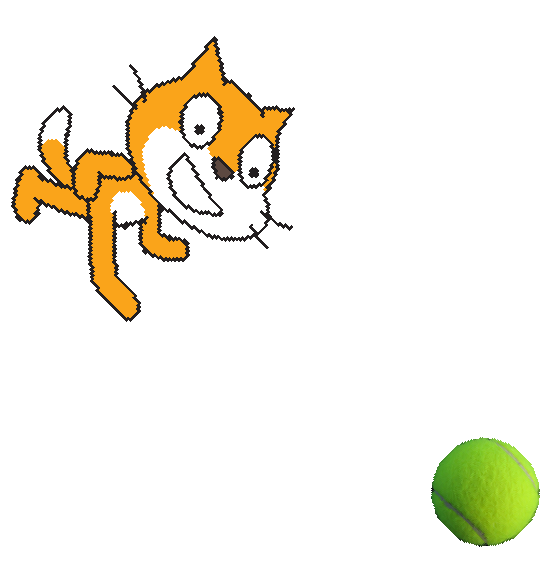


Рисунок Агенты - кот и мяч

Искусственный мир представляет поле, на котором взаимодействуют 2 команды и мяч. Каждый из игроков руководствуется следующими простыми правилами:

* поворачивается по направлению к мячу и делает шаг
* если касается мяча, передает сообщение, что от бьёт по мячу

Агенты-игроки могут различаться по скорости и направленности передвижения.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Поведения агентов-игроков

Мяч в этой игре не только реагирует на сообщения игроков, но и записывает всю историю этих сообщений в переменную список GameStrory

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Поведение мяча

Используя детскую среду конструирования цифровых историй, мы создали простое сообщество взаимодействующих агентов. Обратите внимание, что агент-мяч реагирует на сообщения, которые приходят к нему от агентов-котов и записывает в переменную GameStory, кто прислал ему сообщение. В дальнейшем среда Scratch позволяет обрабатывать полученные внутри, либо экспортировать данные и анализировать их при помощи внешних инструментов.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Сохранение данных и возможность импорта данных

1. Создайте ремикс модели <https://scratch.mit.edu/projects/1314960/>
2. Попробуйте увеличить число агентов игроков, чтобы мяч сохранял в своей переменной списке действия не двух, а более агентов-котов.
3. Попробуйте увеличить число мячей на поле. В этом случае вам для сохранения общей истории действий понадобится глобальная переменная, которая будет сохранять не только то, какой агент-кот совершает действие, но и по отношению к какому агенту-мячу он этой действие совершил.
4. Создайте вариант модели, в котором данные о действиях агентов-котов будут отражаться в виде столбиков гистограммы.

## Тема 2. Многоагентное моделирование в трехмерной среде StatLogo Nova

Цель изучения темы: обосновать необходимость использования многоагентных моделей для понимания отношений, которые складываются между различными социальными группами.

Теоретическое введение. Примеры использования искусственных сообществ для объяснения биологических и социальных феноменов

**Литература:**

* Патаракин Е. Д. Агентное моделирование для рефлексии образовательной организации // Искусственные общества. 2018. T. 13. Выпуск 4 [Электронный ресурс]. URL: http://artsoc.jes.su/s207751800000133-5-1
* Blikstein P., Rand W., Wilensky U. Participatory, embodied, multi-agent simulation AAMAS ’06 / New York, NY, USA: ACM, 2006. 1457–1458 с.

**Практическое занятие (3 часа) Мир трехмерных моделей на примере StarLogo Nova**

**StarLogo Nova** – создание трехмерных многоагентных моделей

StarLogo Nova <https://www.slnova.org/>

Интерфейс среды StarLogo Nova напоминает интерфейс Scratch, с которой мы познакомились при изучении предыдущей темы.

Подробнее с возможностями среды познакомьтесь в приложении 2, где представлены основные возможности StarLogo Nova.

### Пример 1. Моделирование эволюции сотрудничества

Познакомьтесь с тем, как в среде StarLogo Nova может быть реализована версия модели «Дилемма заключенного», различные эволюционные варианты которой рассматриваются в работах Р. Аксельрода

Классическая формулировка дилеммы заключённого:

Двое преступников — А и Б — попались примерно в одно и то же время на сходных преступлениях. Есть основания полагать, что они действовали по сговору, и полиция, изолировав их друг от друга, предлагает им одну и ту же сделку: если один свидетельствует против другого, а тот хранит молчание, то первый освобождается за помощь следствию, а второй получает максимальный срок лишения свободы (10 лет). Если оба молчат, их деяние проходит по более лёгкой статье, и каждый из них приговаривается к полугоду тюрьмы. Если оба свидетельствуют друг против друга, они получают минимальный срок (по 2 года). Каждый заключённый выбирает, молчать или свидетельствовать против другого. Однако ни один из них не знает точно, что сделает другой. Что произойдёт?

Обобщенная формулировка

В игре — два игрока и банкир. Каждый игрок держит 2 карты: на одной написано «сотрудничать», на другой — «предать» (это стандартная терминология игры). Каждый игрок кладёт одну карту перед банкиром лицом вниз (то есть никто не знает чужого решения, хотя знание чужого решения не влияет на анализ доминирования). Банкир открывает карты и выдаёт выигрыш. Если оба выбрали «сотрудничать», оба получают C. Если один выбрал «предать», другой «сотрудничать» — первый получает D, второй с. Если оба выбрали «предать» — оба получают d.

Значения переменных C, D, c, d могут быть любого знака (в примере выше все меньше либо равны 0). Обязательно должно соблюдаться неравенство D > C > d > c, чтобы игра представляла собой «дилемму заключённого».

Если игра повторяется, то есть играется больше 1 раза подряд, общий выигрыш от сотрудничества должен быть больше суммарного выигрыша в ситуации, когда один предаёт, а другой — нет, то есть 2C > D + c.

В среде StarLogo Nova эволюционные изменения, когда исследуются условия, при которых великодушная стратегия сотрудничества сохраняется и распространяется внутри популяции игроков, может быть представлена на следующей модели:

<https://www.slnova.org/patarakin/projects/726111/>

На следующем рисунке (рис. 8) представлен запуск модели агентом «Мир» в ответ на нажатие кнопки «setup». Обратите внимание, что в программе действует полезное условие остановки цикла, если число становится больше 500.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Процедуры запуска агента Мир

На рисунке 9 представлены скрипты, управляющие поведением великодушных агентов. При встрече с эгоистичным агентом его запасы энергии увеличиваются незначительно. Если встречаются два великодушных агента, то их благородство может быть вознаграждено. Количество вознаграждения регулируется пользователем при помощи виджета-рычажка.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Поведение великодушных игроков

Скрипты, управляющие поведением эгоистичных агентов, представлены на рис. 10. При встрече с великодушным агентом эгоистичные игроки получают фиксированный доход. Но, при встрече с другим эгоистичным игроком платят цену за свой эгоизм.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Поведение эгоистичного игрока

### Пример 2 Собираем данные с футбольного поля

<https://www.slnova.org/patarakin/projects/694023/> - модель мячиков на поле. Мы продолжаем моделировать отношения агентов – игроков и мячей на футбольном поле, но используем для моделирования более богатую возможностями среду. Интерфейс для управления свойствами модели представлен на рисунке 11. Пользователь может управлять числом игроков на поле и расстоянием, на котором игроки видят мяч – виджет Vision.

На экране мы видим зелёное поле, где взаимодействуют игроки, и таблицу, в которой записываются действия – ID игрока и ID мяча, по которому он ударяет.

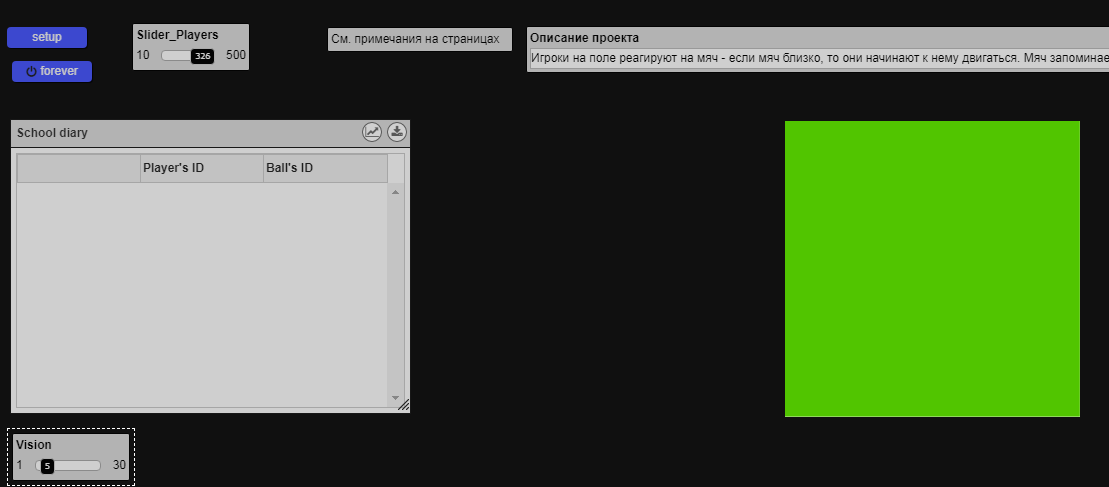


Рисунок Интерфейс футбольного поля

В искусственном мире модели существуют две породы агентов.

Первая порода для футболистов. Агенты этой породы в своем поведении руководствуются следующим правилом:

если рядом с тобой находится мяч (если ты его видишь), то повернись в направлении этого мяча и сделай шаг вперед.

Скрипты, управляющие поведением агентов породы футболистов представлены на рисунке 12. В верхней части рисунка мы видим свойства, которыми обладают агенты данной породы. Здесь пока перечислены только их форма, цвет и размер. Но, используя кнопку «Add Custom Trait» мы можем добавлять игрокам новые свойства. Например, мы можем относить их к разным командам, регулировать силу и точность удара, указывать части поля, на которых они предпочитают играть.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Управление поведением игроков

Вторая порода создана для футбольных мячей. Эти агенты следуют следующим правилам: если расстояние до футболиста меньше двух шагов, то запиши в лог-файл ID-номер агента и свой собственный ID и переместись на десять шагов от агента.

Процедура, управляющая поведением мяча представлена на рисунке 13. Согласно этой процедуре, если рядом с мячом находится агент(ы) из породы игроков, то мяч записывает в журнал ID ближайшего игрока и свой собственный ID. После этого мяч отлетает от игрока. Мы можем считывать из системы не только ID игрока, но и другие его свойства и увязывать поведение мяча не только с тем направлением, куда смотрит игрок, но и с другими свойствами агента игрока.

**Изображение выглядит как снимок экрана

Автоматически созданное описание**

Рисунок Управление поведением мячей

**Вопросы для обсуждения и задания для практической работы**

<https://www.slnova.org/patarakin/projects/694023/> - модель мячиков на поле. Выполните следующие задания:

* создайте ремикс модели;
* добавьте виджет, который позволит пользователям управлять числом мячей на поле;
* предложите новые свойства игроков, которые позволяет различать их на поле;
* предложите способы объединения игроков в команды на основании свойств, которыми они обладают;
* предложите свойства, которые будут влиять на поведение игроков и на то, как они воздействуют на мячи;
* предложите стратегии, которыми могут руководствоваться игроки различных команд, чтобы надежнее удерживать мяч;
* опубликуйте свой вариант модели.

## Тема 3. Многоагентное моделирование в среде NetLogo

Цель изучения темы: освоить возможности среды NetLogo для изучения и моделирования социальных процессов. В ходе изучения темы мы используем NetLogo для представления возможностей агентного моделирования в области социальных исследований. В качестве примеров рассматриваются следующие примеры:

Модель игры «Жизнь» - использование пятен (patches) NetLogo в качестве агентов.

Модель «Сегрегация» Т. Шеллинга – использование в качестве агентов пятен и черепах NetLogo

**Литература:**

* Патаракин Е.Д., Ярмахов Б.Б. Моделирование организационных отношений с использованием «связей» Netlogo // Образовательные технологии и общество (Educational Technology & Society). 2009. № 2 (12). C. 409–422.
* Патаракин Е.Д., Ярмахов Б.Б., Буров В.В. Агентное моделирование деятельности внутри вики-систем // Образовательные технологии и общество (Educational Technology & Society). 2011. № 2 (14). C. 407–422.

**Практическое занятие (3 часа).**

Познакомьтесь с возможностями, которые открывает для моделирования социальных отношений среда NetLogo (См. Приложение 3). Научитесь использовать примеры моделей из библиотеки NetLogo.

### Пример 1. Игра «Жизнь» в NetLogo

Рассмотрите правила игры «Жизнь» и то, как эти правила переводятся в язык команд NetLogo, которые управляют поведением пятен.

1. Действие происходит на бесконечной плоскости, разделенной на клетки, которую можно иногда представить как зацикленную конечную.
2. Каждая клетка может находиться в двух состояниях: быть живой или быть мёртвой.
3. У каждой клетки 8 соседей.
4. Если клетка жива и у нее 2−3 живых соседа, то она остается живой, иначе умирает.
5. Если клетка мертва и у нее 3 живых соседа, то она становится живой, иначе остается мертвой.
6. Игра прекращается, если на поле не останется ни одной живой клетки.
7. Игра прекращается, если при очередном шаге ни одна из клеток не меняет своего состояния.
8. Игра прекращается, если конфигурация на очередном шаге в точности повторит себя же на одном из более ранних шагов.

<https://netlogoweb.org/launch#https://netlogoweb.org/assets/modelslib/IABM%20Textbook/chapter%202/Life%20Simple.nlogo>

Изображение выглядит как снимок экрана

Автоматически созданное описание

patches-own [live-neighbors]

;; переменная пятен

to setup

clear-all

ask patches [

;; при запуске создать примерно 10% живых клеток

set pcolor blue - 3 ;; темно-синие клетки считаются мертвымм

if random 100 < 10 [

set pcolor green ;; зелёные клетки считаются живыми

]

]

reset-ticks

end

to go

ask patches [

;; каждая клетка считает число живых зеленых соседей

;; и сохраняет это число в переменной live-neighbors

set live-neighbors count neighbors with [ pcolor = green ]

]

ask patches [

;; клетки с 3 зелёными соседями становятся или остаются живыми

if live-neighbors = 3 [ set pcolor green ]

;; клетки, у которых 0 или 1 живой сосед умирают или остаются мертвыми (из-за изоляции)

if live-neighbors = 0 or live-neighbors = 1 [ set pcolor blue - 3 ]

;; клетки с 4 и более соседями умирают или остаются мертвыми (из-за перенаселенности)

if live-neighbors >= 4 [ set pcolor blue - 3 ]

;; клетки с 2-мя живыми соседями сохраняют свой цвет и свое состояние

]

tick

end

Для самостоятельной работы:

* Посмотрите, что происходит в системе, если вы вызываете однократные изменения – GO-Once
* Проследите судьбу групп пятен, чтобы понимать действие правил, управляющих развитием группировок агентов.
* Попробуйте выделить на поле своей модели устойчивые и повторяющиеся конфигурации: планеры, улья, лодки, мигалки, звезды, кресты и т.д.
* Внесите изменения в код модели, чтобы в начале игры на поле было бы больше живых клеток.
* Внесите изменения в интерфейс и код модели, чтобы пользователь мог устанавливать процент живых клеток в начале игры.
* Внесите изменения в код модели, чтобы каждая клетка оценивала не 8 возможных соседей, в только 4. Для реализации вам понадобится команда neighbors4
* Попробуйте переписать модель так, чтобы действующими агентами были бы не пятна, а черепашки. В этом случае вам понадобятся команды, которые управляют рождением (hatch, sprout) и смертью черепах (die).

### Пример 2. Модель «Сегрегация» в NetLogo

Общество в модели Шеллинга состоит из агентов двух типов. Агент предпочитает жить в окружении себе подобных, то есть старается переместиться в область, где ему подобных агентов больше. Рассмотрите правила модели «Сергегация», как они были предложены Т. Шеллингом для поля настольной игры. На шахматной доске (64 клетки) он расположил равномерно монетки, отображающие белых и черных жителей мегаполиса («орел» и «решка») (рис. 2, слева). Каждый из жителей имеет четкий набор правил, в данном случае связанных с расовой терпимостью (толерантностью). В частности, он не будет переезжать в другое место, если:

— у него только один сосед и он того же цвета;

— у него два соседа, один из которых того же цвета;

— у него 3—5 соседей, два из которых того же цвета;

— у него 6—8 соседей, три из которых того же цвета.

Модель сегрегации, переведенная на язык агентного моделирования NetLogo

<https://www.netlogoweb.org/launch#https://www.netlogoweb.org/assets/modelslib/Sample%20Models/Social%20Science/Segregation.nlogo>

**Изображение выглядит как снимок экрана

Автоматически созданное описание**

Рисунок Модель сегрегации - исходное состояние

globals [

; глобальные переменные

percent-similar ;; какой процент черепах-соседей такого же цвета

;; как и эта черепаха

percent-unhappy ;; какой процент черепах несчаствлив?

]

turtles-own [

; переменные черепах

happy? ;; для каждой черепахи показывает минимальный % соседей такого же  
 ;; цвета, при котором она счастлива

similar-nearby ;; число соседних полей, занятых черепахами такого же цвета?

total-nearby ;; сколько всего соседей у данной черепахи?

]

to setup

clear-all

;; очистить экран и сознать число NUMBER черепах, распределенных по полю случайно.

;; заметьте, что максимальное значение рычажка в модели равно 25000 – чуть меньше чем число доступных полей

ask n-of number patches [

sprout 1

]

ask turtles [

;; примерно половина черепах будет зеленого, а половина красного цвета

set color one-of [ red green ]

]

update-turtles

update-globals

reset-ticks

end

to go

if all? turtles [ happy? ] [ stop ]

move-unhappy-turtles

update-turtles

update-globals

tick

end

to move-unhappy-turtles

ask turtles with [ not happy? ]

[ find-new-spot ]

end

to find-new-spot

rt random-float 360

fd random-float 10

if any? other turtles-here

[ find-new-spot ] ;; продолжает идти в поисках свободного места

setxy pxcor pycor ;; становится в центр пятна

end

to update-turtles

ask turtles [

set similar-nearby count (turtles-on neighbors)

with [color = [color] of myself]

set total-nearby count (turtles-on neighbors)

set happy? similar-nearby >= ( %-similar-wanted \* total-nearby / 100 )

]

end

to update-globals

let similar-neighbors sum [similar-nearby] of turtles

let total-neighbors sum [total-nearby] of turtles

set percent-similar (similar-neighbors / total-neighbors) \* 100

set percent-unhappy (count turtles with [not happy?]) / (count turtles) \* 100

end

Изображение выглядит как снимок экрана

Автоматически созданное описание

Для самостоятельной работы:

* Попробуйте устанавливать разные значения плотности населенности DENSITY. Выясните, как первоначальная плотность населения влияет на процент тех агентов, которые считают себя несчастными? Как плотность влияет на время, необходимое для того, чтобы модель пришла в устойчивое состояние и завершила свою работу?
* Попробуйте разные значения для рычажка % -SIMILAR-WANTED. Как изменяется общая степень сегрегации?
* Попробуйте, используя только рычажки-виджеты добиться такого состояния, чтобы модель никогда не останавливала бы свою работу, а агенты всё время продолжали бы искать себе новые лучшие места для жизни
* Можете ли вы установить ползунки так, чтобы модель никогда не заканчивала работать, а агенты продолжали искать новые местоположения?
* Попробуйте изменить правила, по которым агенты определяют собственное состояние счастья. Например, они могут каким-то образом оценивать, насколько хорошими являются их соседи. Возможно, у агентов появляются дополнительные черты, кроме цвета (принадлежность к клубу, уровень дохода и т.д.)

## Тема 4. Объяснение и понимание через моделирование и выращивание

Цель изучения темы: сформировать навыки и умения создания, использования и видоизменения прото-историй.

**Теоретическое введение (1 час)**. История использования агентных моделей в качестве прото-историй. Работы Дж. Эпштейна. Социальные исследования, опирающиеся на подход «снизу-вверх».

Литература:

* Макаров В.Л. Искусственные общества. Экономика и математические модели https://elibrary.ru/item.asp?id=17966571
* Макаров В.Л. Социальное моделирование набирает обороты. Экономика и математические методы, издательство Наука (М.), том 49, № 4, с. 5-17
* Epstein, J. and Axtell, R. (1996). Growing Artificial Societies: Social Science from the Bottom Up. Washington, D.C.: Brookings Institution Press.

В.Л. Макаров в своих работах о построении искусственных сообщества пишет, что ключевой фразой новой методологии многоагентного моделирования можно считать цитату из книги Epstein Joshua M. and Axtell Robert (1996). «Однажды люди применительно к социальному явлению вместо вопроса «Можете ли вы это объяснить?» спросят «Можете ли вы это построить (вырастить)?»

Текст книги Эпштейна и Акстелла выстраивался вокруг компьютерной модели, в которой они анализировали поведение жуков в сахарном ландшафте. Имеется некоторая окружающая среда — сахарный ландшафт, где в двумерном пространстве разбросан сахар — где-то больше, где-то меньше; и туда же помещены агенты-жуки, которые ползают по сахарному ландшафту по простым правилам: агенту надо есть сахар, и он перемещается туда, где сахара больше. Свойства агента задаются двумя параметрами: горизонтом видения и уровнем метаболизма, т.е. количеством сахара, которое нужно съесть, чтобы остаться в живых. При этом агент может носить с собой сколько угодно сахара. Если сахара недостаточно, агент умирает.

Правила поведения агентов:

1. Осмотреть все доступные для агента окрестности в четырех или восьми направлениях (только по прямой или по прямой и по диагонали) и выбрать незанятую клетку с максимальным количеством сахара.

2. Передвинуться в эту клетку и собрать весь имеющийся в ней сахар.

Агенты двигаются, поедают сахар, который появляется в той или иной точке в соответствии с определенными правилами. Наблюдая за поведением агентов на экране, мы видим то, что Джошуа Эпштейн и Роберт Акстелл определили как прото-историю или Proto-Narrative. Авторы «сахарной модели», Джошуа Эпштейн и Роберт Акстелл, проводили со своим творением всевозможные эксперименты, в частности сделали модель, где в качестве окружающей среды взяли карту США, и жуки ползали по всей Америке. И в это же время компания Pizza Hut задалась вопросом, как создать широкую сеть своих заведений, где их разместить, чтобы получить как можно больше клиентов и соответственно доходов. Специалист, которому это было поручено, разработал свой проект, и тут ему в руки попал отчет о «сахарной модели». Сначала он решил, что идея была украдена у него, но затем встретился с Эпштейном и Акстеллом, они стали сотрудничать, и специалист из Pizza Hut даже смог улучшить свои результаты. Действительно, в качестве сахара можно взять людей — американцев. Они неравномерно разбросаны по стране — в Калифорнии больше, в Неваде меньше. А жуки, или агенты, — это ресторанчики Pizza Hut.

**Практическое занятие (3 часа).**

Откройте модель «Sugarscape» в библиотеке моделей NetLogo

<http://ccl.northwestern.edu/netlogo/models/Sugarscape1ImmediateGrowback>

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Сахарная модель в библиотеке проектов NetLogo

Для самостоятельной работы:

* Проведите эксперименты с моделью, изменяя численность начальной популяции агентов-жуков в модели (используйте рычажок initial-population).
* Посмотрите на то, как связана исходная и конечная численность популяции. Влияет ли начальная численность на конечную численность?
* Влияет ли исходная численность на распределение благосостояния внутри популяции? Для решения этой задачи вам понадобятся данные гистограммы, расположенной на рисунке 16 справа от изображения сахарного ландшафта.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Интерфейс сахарной модели в среде NetLogo

В дальнейшем модель стали усложнять, и следующие её варианты превратились в настоящее искусственное общество. Например, экономика появилась в «сахарной модели» после того, как в модель добавили сироп. Жуки стали есть сахар и сироп, и у одних оказалось больше сахара, у других сиропа, они начали встречаться и обмениваться — конечно, соблюдая определенные пропорции. Какое-то количество сахара можно обменять на некоторое количество сиропа — возникает цена.

Вы можете увидеть последующие варианты сахарной модели и в библиотеке NetLogo и среди моделей, которые создает сообщество программистов NetLogo на портале <http://modelingcommons.org/>

* Посмотрите на различные варианты развитие сахарной модели, созданные разными авторами.
* Отметьте основные изменения, которые вносились в сахарную модель для того, чтобы приблизить её к реальному миру. Например, в свойства агента добавляли пол и после введения этого свойства мужчины и женщины могут вести себя по-разному. Другой возможный путь развития модели – возможность передавать накопленный сахар по наследству.

В модели https://ccl.northwestern.edu/netlogo/models/Sugarscape3WealthDistribution

в качестве важного параметра появился возраст агента. Даже если в среде для агента достаточно сахара, он все равно умирает при достижении определенного возраста. При этом в системе рождается новый агент и общая численность агентов остается постоянной.

## Тема 5. Моделирование организационного поведения

Цель изучения темы: сформировать навыки и умения интерпретации данных при помощи среды многоагентного моделирования.

Цифровая среда, в которой организуется современная совместная деятельность, позволяет сохранять данные о действиях субъектов совместной деятельности и на основании этих данных выстраивать связи, которые возникают между субъектами и объектами деятельности. Действия, которые совершают акторы в цифровой среде, очень похожи на процедуры, которые выполняют черепашки в известной модели «Термиты» когда каждая черепашка следует простым правилам:

*Если ты идешь и ничего не несешь и встречаешь на своем пути деревянную щепку, то возьми её. Если ты идешь с деревянной щепкой и встречаешь другую щепку, то положи щепку, которую несешь, и двигайся дальше.*

Каждая черепашка выполняет последовательность процедур search-for-chip find-new-pile put-down-chip и в результате из множества случайным образом разбросанных по экрану палочек собирается одна круглая куча.

Модель термитов кажется наиболее привлекательной, поскольку в ней присутствуют палочки chips как объекты совместной деятельности. Экспериментируя с этой моделью, мы можем глубже понять феномены совместной деятельности. Представим себе, что у термитов есть журнал учета рабочих действий, куда они записывают все свои действия со щепочками. Т.е. если участник совершает какое-то результативное действие с chip, то он об этом действии оставляет запись в журнале. Для того чтобы проверить, что дают нам попутные записи в журнал, мы несколько видоизменили исходный текст модели Termites добавив к модели новые переменные и правила. В модели появилась переменная список WIKILOG, куда термиты записывают отчеты о своих действиях. В процедурах search-for-chip и put-down-chip были сделаны небольшие добавления. В модель была добавлена процедура, которая на основании записей в журнале устанавливает связь между агентами, которые перетаскивали одну и ту же палочку.

Модель реализована на языке NetLogo 6.1 и исходный код доступен в сети Интернет <http://modelingcommons.org/browse/one_model/5071>

Интерфейс видоизмененной модели представлен на рисунке 17. Кнопка logs\_to\_sociogram убирает с экрана все палочки и показывает связи между агентами, которые перемещали одни и те же палочки.

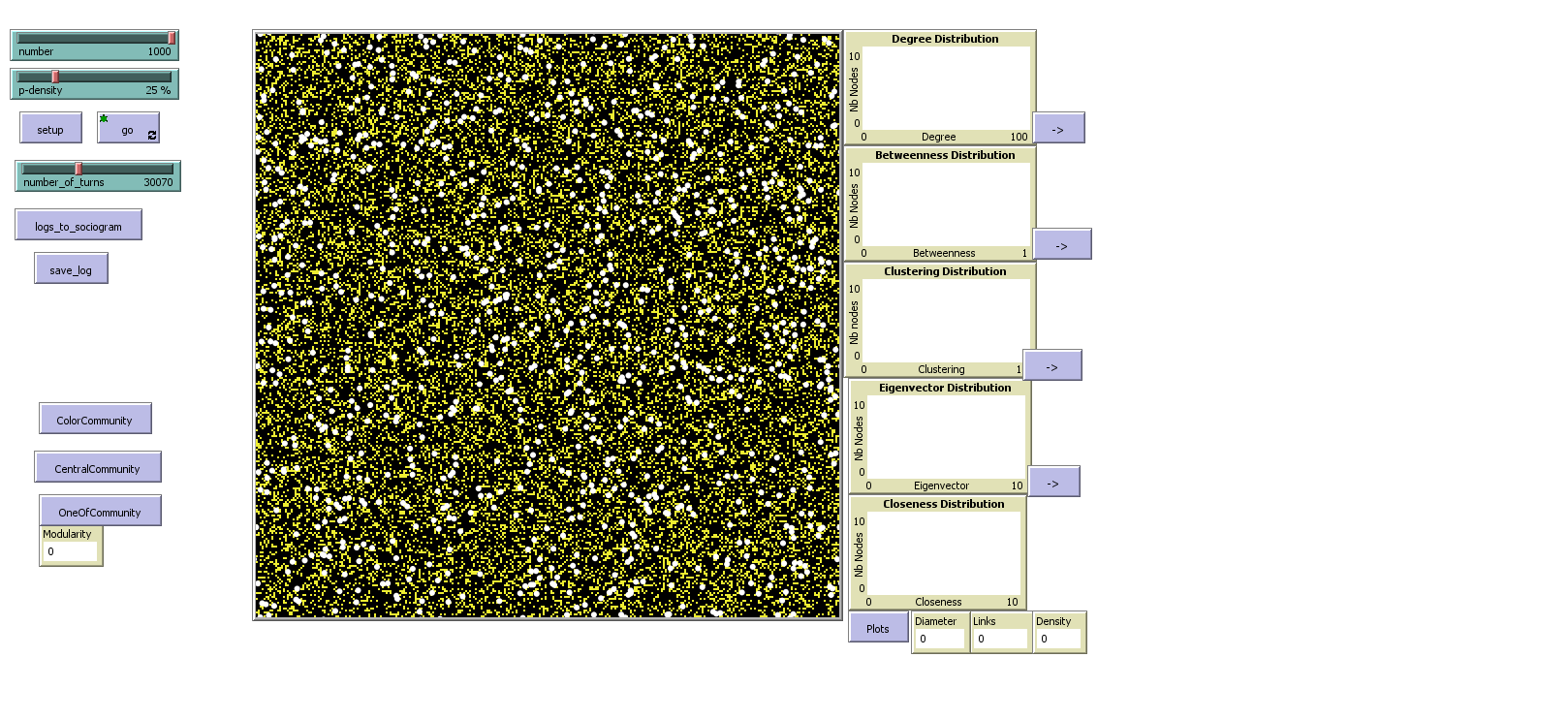


Рисунок Интерфейс модели Термиты с журналом

Эта простейшая модель создания социограмм на основании действий с социальными объектами может быть применена к различным ситуациям совместной деятельности. Во всех случаях, когда агенты деятельности производят действия над объектами совместной деятельности, журнал этих действий может служить исходным материалом для построения социограмм и проведения социального исследования. Например, мы можем смоделировать ситуацию, когда 1000 независимых агентов в течение 30000 ходов переносят 6000 палочек. После того как все действия были совершены и записаны в журнал, модель убирает с экрана палочки и показывает связи между термитами (Рисунок 18).

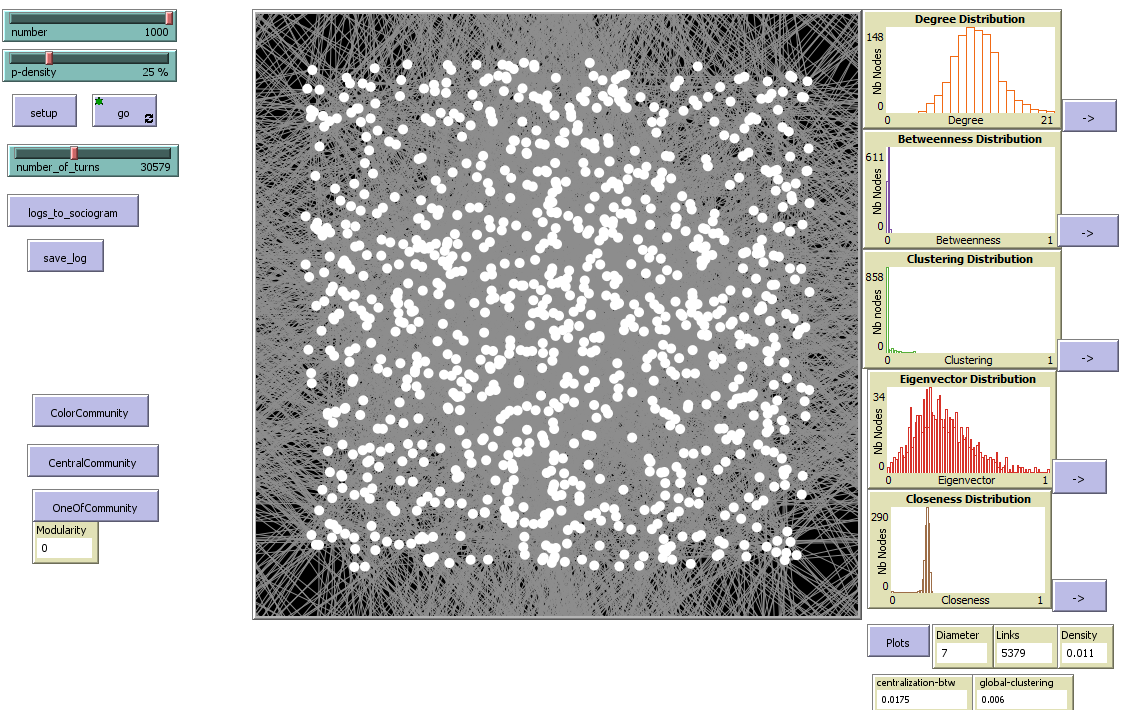


Рисунок Сеть термитов, связанных общей историей

В правой части экрана на рисунке 18 представлены гистограммы распределения акторов по сетевым характеристикам:

* Центральность по степени (Degree centrality) – показывает, насколько много связей у данного узла.
* Центральность по посредничеству (Betweenness) – показывает вероятность того, что данный узел контролирует информационные потоки внутри системы совместной сетевой деятельности.
* Центральность по собственному вектору (Eigenvector centrality) – показывает насколько данных узел связан с другими важными узлами.
* Центральность по близости (Closeness centrality) – показывает, насколько быстро можно добраться от данного узла до других узлов сети.
* Коэффициент кластеризации (Clustering coefficient) – показывает, насколько связаны между собой узлы, связанные с данным узлом. Локальный коэффициент кластеризации достигает высокого уровня у тех узлов, которые включены в состав групп.

**Литература:**

1. Патаракин Е.Д. Макроскопический подход к анализу совместной сетевой деятельности // Образовательные технологии. 2017. № 2. c. 51–65 <https://elibrary.ru/item.asp?id=29438094>
2. Патаракин Е.Д., Ярмахов Б.Б. Анализ совместной сетевой деятельности участников образовательной организации // Образовательные технологии. 2016. № 2. c. 51–65.
3. Ушаков К.М., Куксо Е.Н. Возможности сетевого анализа для исследований в образовании // Народное образование. 2015. № 3. c. 79–88.

**NetLogo Web**

<https://www.netlogoweb.org/launch#NewModel> – новая модель

Termites – модель термитов

<http://www.uic.unn.ru/pustyn/netlogo/TermitesLogs.html> + Журнал

**Вопросы для обсуждения и задания для практической работы**

1. Используйте <https://www.netlogoweb.org/launch#NewModel> – и создайте 2 или больше связанных агентов командами в Command Center:

observer> ca crt 3 [set size 5 set shape "person" fd 10]

observer> ask turtles [create-links-to other turtles]

2. Рассмотрите модель Termites. Предложите правила, которыми руководствуется каждый отдельный агент в рамках данной модели.

3. Предложите изменения к модели, которые бы сделали её более подходящей к моделированию деятельности организации и к проблемам управления персоналом.

4. Используйте в качестве исходной модели материалы <http://modelingcommons.org/browse/one_model/5071> и посмотрите, как создается динамическая карта совместной деятельности. Рассказы о совместной сетевой деятельности школьных представлены в нескольких вариантах. Например, <http://archive.novator.team/post/9617>

В среде NetLogo есть возможность записывать все изменения в виде фильма в формате qtime. Другая возможность, которую мы использовали при записи представленных ниже видеофрагментов связана с тем, что мы сохраняем все изменения в отдельный файл gif, объединяем множество таких файлов в единый анимированный файл и добавляем к этому файлу звуковое сопровождение.

5. Посмотрите примеры рассказов об истории совместной деятельности сотрудников организации, украшенные динамически социограммами:

<https://www.youtube.com/watch?v=NLVWxdHwmYg>

<https://www.youtube.com/watch?v=HioyuCYQ1iQ>

<https://www.youtube.com/watch?v=y8hcYGnJZ08>

## Учебно-методическое и информационное обеспечение дисциплины:

### а) основная литература:

1. Патаракин Е. Д. Агентное моделирование для рефлексии образовательной организации // Искусственные общества. 2018. T. 13. Выпуск 4 [Электронный ресурс]. URL: http://artsoc.jes.su/s207751800000133-5-1
2. Патаракин Е.Д., Ярмахов Б.Б. Анализ связей между сотрудниками школы на основе их взаимодействия в домене Google Apps для образования // Образовательные технологии и общество». 2016. Vol. 19, № 2. P. 585–599. - <https://elibrary.ru/item.asp?id=26025382>
3. Патаракин Е.Д. [и др.]. Выявление ключевых участников социально-педагогических проектов // Образовательные технологии и общество (Educational Technology & Society). 2015. № 2. C. 675–692.

### б) дополнительная литература:

1. Акопов А.С., Хачатрян Н.К. Агентное моделирование / А.С. Акопов, Н.К. Хачатрян, Центральный экономико-математический институт РАН, 2016.
2. Кузнецов А.В. Краткий обзор многоагентных моделей // Управление Большими Системами: Сборник Трудов. 2018. № 71. C. 6–44.
3. Макаров В.Л., Бахтизин А.Р. Современные методы прогнозирования последствий управленческих решений // Управленческое Консультирование. 2015. № 7 (79). C. 12–24.
4. Axelrod R. The Complexity of Cooperation: Agent-based Models of Competition and Collaboration / R. Axelrod, Princeton University Press, 1997. 248 c.
5. Damaceanu R.-C. Agent-based Computational Social Sciences Using NetLogo: Theory and Applications / R.-C. Damaceanu, Germany: LAP Lambert Academic Publishing, 2011.
6. Fontana M., Terna P. University of Turin. From Agent-based models to network analysis (and return): the policy-making perspective. 2015.
7. Schelling T.C. Micromotives and Macrobehavior / T.C. Schelling, W. W. Norton & Company, 2006. 289 c.
8. Wilensky U., Rand W. An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo / U. Wilensky, W. Rand, MIT Press, 2015.

# Приложение 1. Руководство по агентному моделированию в среде Scratch

Языки визуального программирования, такие как Scratch, являются практической альтернативой сложным языкам программирования с текстовым синтаксисом. Они существенно упрощают синтаксис и заменяют слова картинками или блоками. Например, в Scratch блоки, похожие на кирпичики Lego, представляют несколько управляющих конструкций. «Программирование» осуществляется путем перетаскивания и составления этих блоков друг с другом (см. рисунок 19).

Изображение выглядит как снимок экрана

Автоматически созданное описание

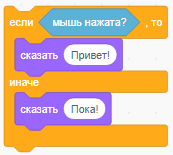
Рисунок Scratch-блоки различаются цветом и формой, что облегчает сборку

Управление поведением агентов в Scratch осуществляется при помощи условий. В программировании условие — это то, что должно быть истинным, чтобы произошло какое-либо действие.

Пример такого блока:

   
Представленная конструкция известна как конструкция if-then. С ее помощью мы можем дать команду спрайту поздороваться, только если пользователь, скажем, нажмет кнопку мыши:

Изображение выглядит как снимок экрана

Автоматически созданное описание   
Похожая конструкция называется конструкцией if-else:   
  
   
С ее помощью мы можем дать команду спрайту поздороваться или попрощаться в зависимости от того, нажал ли пользователь кнопку мыши:   
  


Подробное описание управлением поведением агентов в среде Scratch приведено в отдельном руководстве

См. Патаракин Е.Д. Учимся готовить в среде Скретч <https://db.ph-int.org/upload/iteach/texts/pi_2010_06_30-19_26_40_1.pdf>

Здесь мы обратим внимание на необходимые для агентного моделирования возможности создания клонов и определения новых процедур.

#### Создание клонов.

Например, мы создаем агент-каплю, и поручаем этому агенту создавать клоны самого себя. Каждый клон поднимается наверх (в небо), медленно спускается на землю и удаляется. В результате на экране возникает иллюзия дождя или снегопада (рис. 20).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Клонирование создает иллюзию дождя

#### Создание процедур.

В Scratch можно не только использовать готовые блоки команд, но и создавать новые процедуры. При этом процедура может быть связана с одним или нескольким входными параметрами. Например, следующая процедура получает на входе 3 числа и возвращает максимальное из них.

Изображение выглядит как снимок экрана, векторная графика

Автоматически созданное описание

Рисунок Функция, возвращающая максимальное значение

# Приложение 2 Многоагентное моделирование в среде StarLogo Nova

StarLogo Nova – среда совместного агентного моделирования в трехмерной среде. Во многом это среда продолжает и развивает традиции языка Scratch. При этом здесь можно ставить серьезные эксперименты и получать воспроизводимые результаты. Центральной метафорой среды является мир (World), в котором можно создавать агенты различных пород.

Среда активно используется в естественно-научном образовании. В качестве примера следует привести проект GUTS, в котором взаимосвязаны образовательные стандарты вычислительного мышления и исследовательской деятельности.

Интерфейс программы наследует общие метафоры искусственных миров таких систем программирования как Scratch, Netlogo и многих других. У нас есть искусственный мир, в рамках которого мы можем создавать агентов и предписывать им выполнять указания. При этом в мире существуют не отдельные агенты, а породы агентов, обладающих общими свойствами.

Мир StarLogo наследует основные понятия Scratch. Здесь есть:

- Рабочая область экрана, где агенты совершают свои действия – зелёная область в центре экрана.

- Вокруг зеленой области, где действуют агенты чёрное поле, где автор может размещать управляющие кнопки и информационные виджеты

- Область хранения команд, которые собраны в разноцветных ящиках

- Страницы отдельных пород, на которых из разноцветных команд собираются процедуры, управляющие поведением агентов данной породы

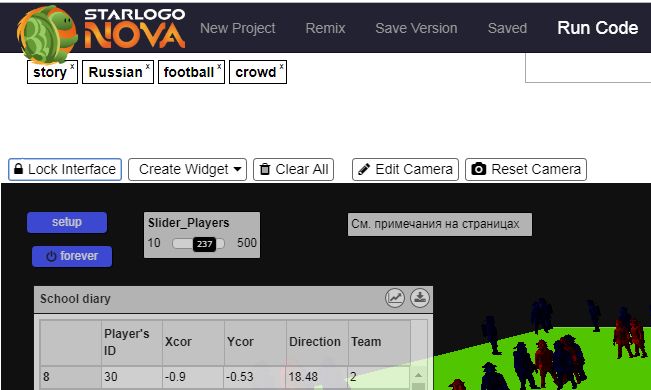


Рисунок Интерфес StarLogo Nova

Управление проектами – вверху экрана и здесь же кнопка запуска кода «Run Code». Не забывайте запускать код всякий раз, когда вы вносите изменения в свой проект. Без этой кнопки никаких изменений не происходит.

Следующая важная кнопка, с которой начинаются события в мире, называется Setup. Обычно эту кнопку использует Мир (The World) при обновлении и запуске проекта. На рисунке представлена последовательность действий Мира, «когда нажмут на кнопку setup»:

Очистить местность

Очистить граф (таблицу) журнала

Установить время в 0

Установить текст «» в описание проекта

Удалить всех агентов на местности

Создать агентов породы Person и пусть каждый из них

Станет размером 2

Станет по своей форме как Fireman

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Скрипт агента Мир

Важно, что Мир знает свойства всех существующих пород и может обращаться к агентам любых пород.

### Как зарегистрироваться на портале StarLogo Nova?

Предыдущий вариант StarLogo TNG использовался на локальных компьютерах – версию можно скачать и сегодня, но это направление больше не поддерживается. StarLogo Nova доступна на сайте по адресу <https://www.slnova.org/> и здесь можно посмотреть на библиотеку проектов и связанных с ними ресурсов. Чтобы начать создавать собственные проекты, необходимо зарегистрироваться – рис. 24.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Кнопка регистрации

Кнопка «Register» и стандартный процесс регистрации – ничего дополнительного об участнике не спрашивают, никакой информации о возрасте, месте проживании и прочем не собирают. Зарегистрироваться можно только при наличии адреса электронной почты, на который приходит подтверждение – рис. 25.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Поля регистрации

Начать знакомство с возможностями StarLogo TNG лучше всего с просмотра примеров проектов, созданных в этой среде. В данном руководстве мы разбираем проект, доступный в сети по адресу:

<https://www.slnova.org/patarakin/projects/691186/>

### Как создать новый проект?

Для каждого участника система показывает проекты, которые он уже создал. Проекты бывают открытые и закрытые. Здесь устройство очень похоже на мир Scratch – команда явно руководствовалась общими принципами. Перечень проектов – рис. 26

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Перечень проектов

Чтобы создать новый проект достаточно щелкнуть по иконке со знаком +

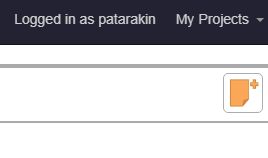


Рисунок Создание нового проект

После того как вы нажмете на страницу со знаком + система предложит вам выбрать тип проекта, который вы хотите создать. Это может быть пустой проект с миром, в котором еще никого нет; может быть игра от первого лица какого-то персонажа, действиями которого вы сможете управлять при помощи клавиш клавиатуры; либо биологическая модель, в которой уже будут существовать исходные породы агентов (рис. 28).

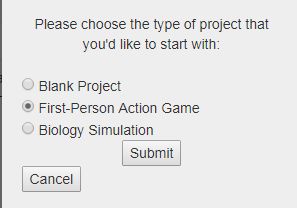


Рисунок Выбор типа проекта

Если выбрать игру, то система показывает мир с точки зрения геометрической фигуры, вокруг которой двигаются другие фигуры. Существует множество развитий этого сюжета, когда в систему добавляются правила взаимодействия героя с другими агентами. Пример видео, где главный герой начинает стрелять по подвижным мишеням - <https://www.youtube.com/watch?v=VAorYkbzssE&t=37s>

Если выбрать биологическую модель, то система изначально создает не только взаимодействующие породы, но и график, на котором будет отображаться численной здоровых, инфицированных и поправившихся экземпляров породы (рис 29).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Шаблон проекта биологической модели

Важно не полениться и сделать понятное название и описание проекта, отметить проект тегами, по которым его смогут найти другие члены сообшества

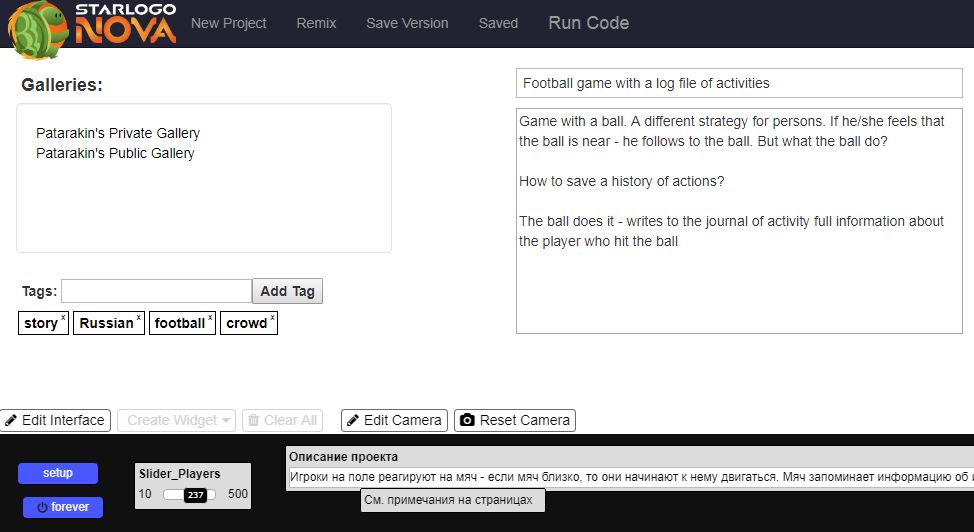


Рисунок Описание проекта

## Интерфейс программы:

Виджет SETUP:

Блоки, в которых задается исходное состояние искусственного мира. Как правило, в setup записывают блоки, предписывающие удаление из мира всех агентов и очистку окружающего их мира. После этого мы создаем агентов (используем блок команд Agents). В следующем примере мы очищаем мир, создаем 3 агента породы Черепаха, и велим каждой и них выбрать для себя красный цвет. После этого мы устанавливаем всех агентов по полю искусственного мира в случайном порядке (рис 31).

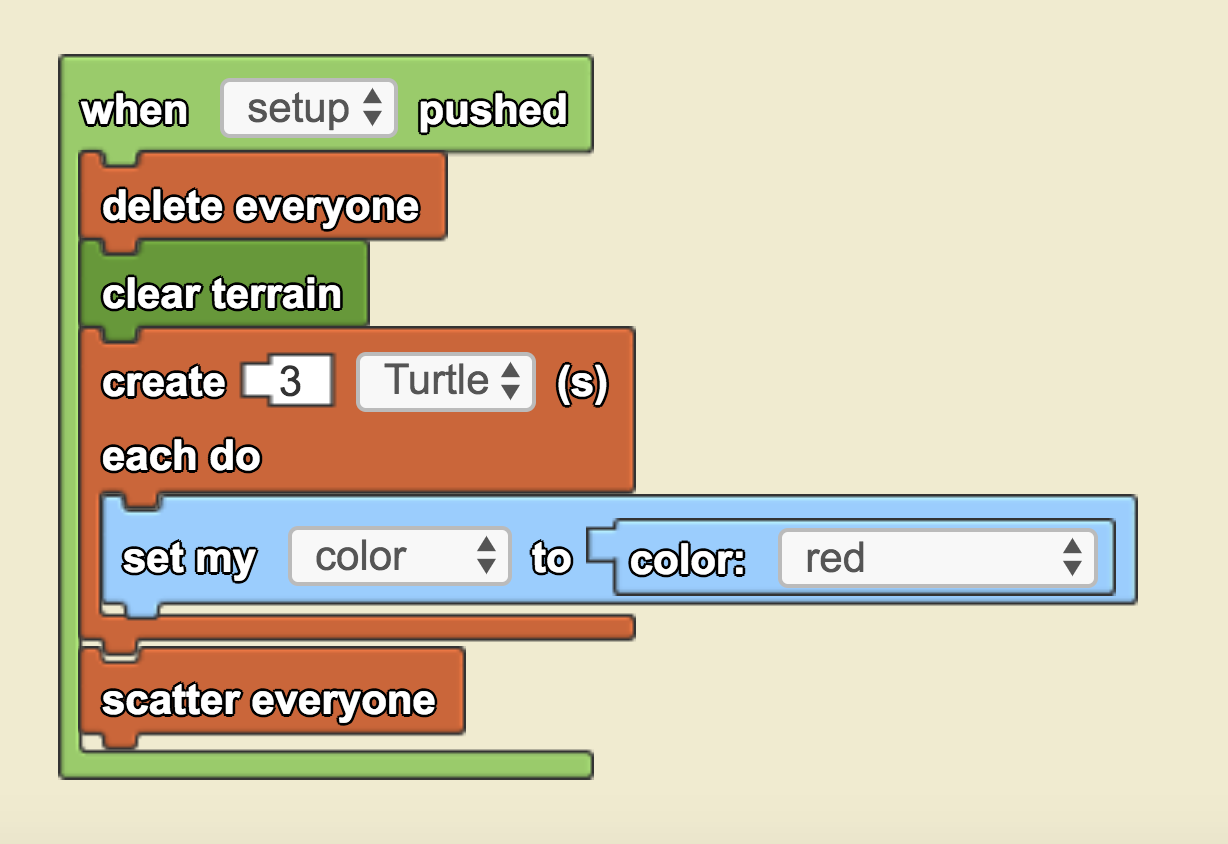


Рисунок Скрипт, связанный с виджетом setup

Мы можем создавать и другие виджеты, подобные setup и предписывать им выполнение кода – изменять цвет поля, создавать новые породы агентов и т.п. Всякий код такого типа помещается на страницу Мир (World) и выполняется после нажатия на виджет.

#### Виджет forever и другие циклы

Любой виджет, связанный с конструкцией “while \_\_\_ toggled” будет создавать цикл команд, которые будут выполняться последовательно, пока кнопка не будет повторно нажата и выключена. Например, в следующем примере агент все время будет двигаться вперёд, если он стоит на красном поле (рис 32).

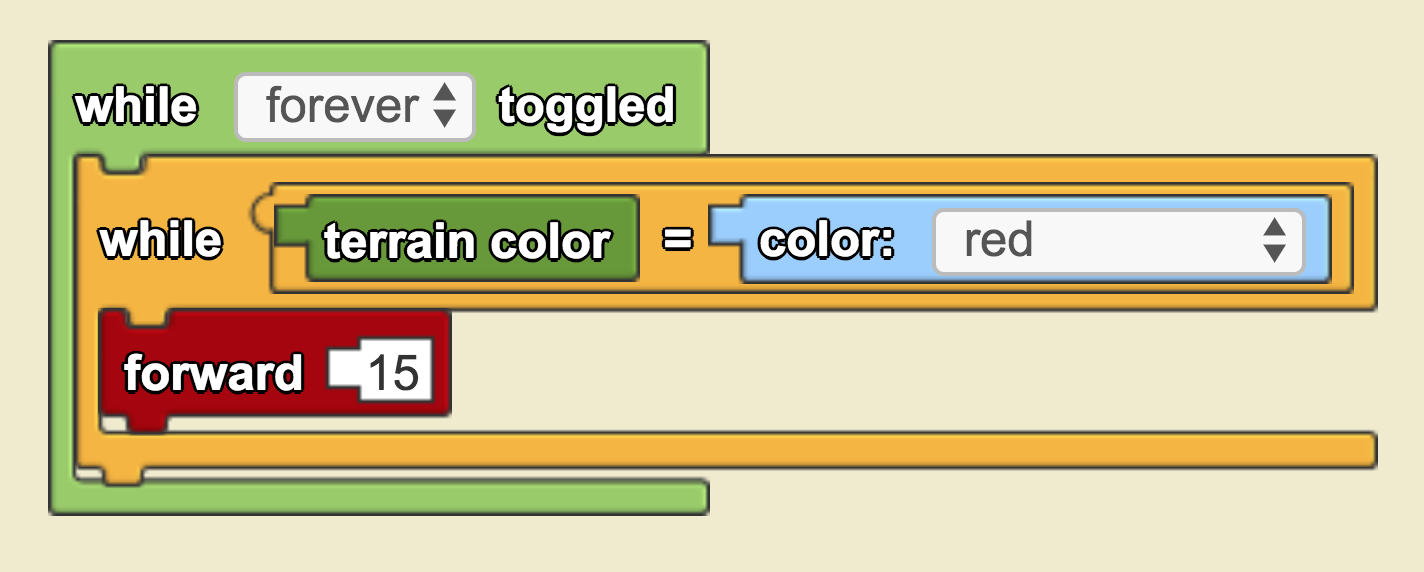


Рисунок Forever toggled

Вы можете создавать блоки команд, которые будут выполняться определенное число раз. Например, на рисунке 33 показано, как мы создаем черепаху и приказываем ей перемещаться 10 раз и в каждой точке оставлять отпечаток желтого цвета.

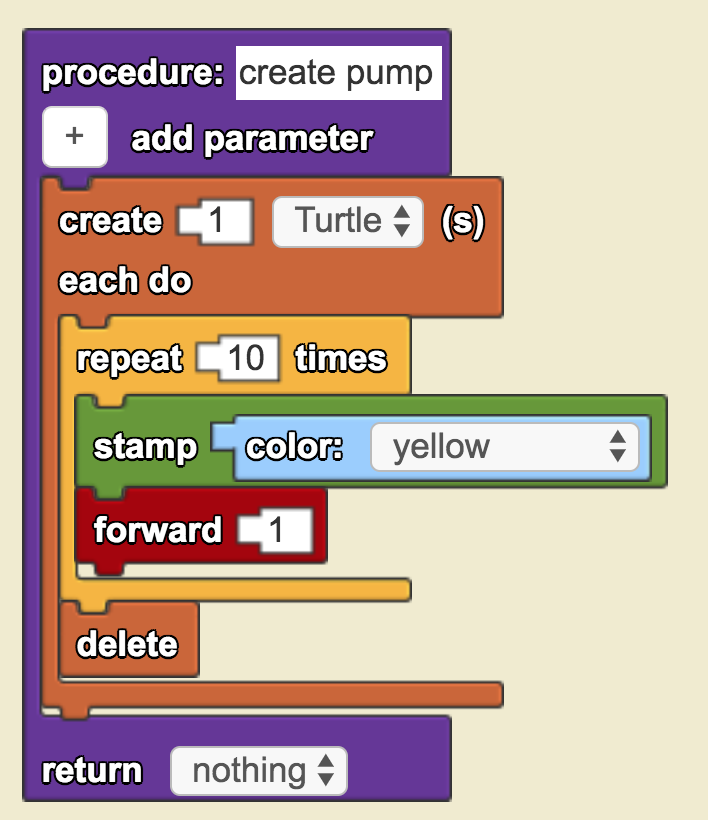


Рисунок Пример повторений

#### Как остановить выполнение кода внутри программы?

В некоторых случаях вам нужен цикл, чтобы выполнить несколько ходов (повторений кода) или до того, как свершится какое-то события. Когда вы используете модель для сбора данных, вам нужно чтобы модель выполняла код достаточно долго. Чтобы модель остановилась по достижению определенного условия, используйте конструкцию

“toggle \_\_\_ ­­ to\_\_\_ ­­ for \_\_\_ ­­” (interface), внутри логического блока “if”. Например, чтобы остановить все программы, когда все черепах умерли, используйте следующий код на странице World (рис. 34):

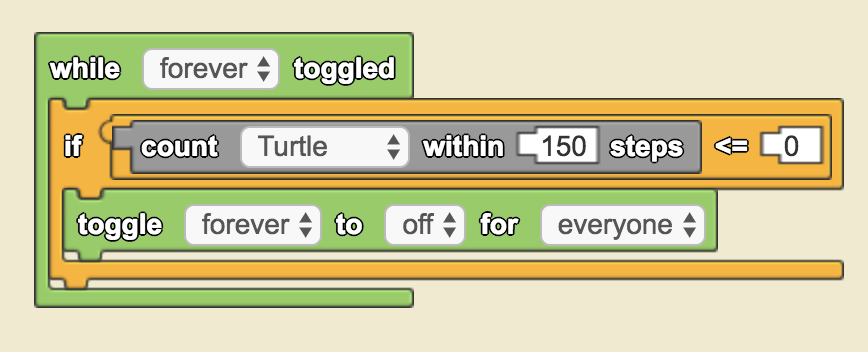


Рисунок 34 toggle forever to off

Вы можете использовать проверку логических условий внутри цикла “while forever toggled” в своем коде. Например, следующий код будет вызывать несколько процедур и обновлять значение time до 500, а затем останавливать процедуру (рис. 35).

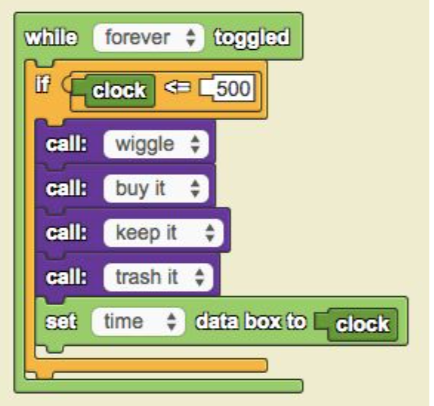


Рисунок Условие остановки

## Агенты

### Создаем и переименовываем существующие породы

Чтобы создать новую породу агентов в верхней части рабочего пространства выбираем вкладку “+Add Breed”. Чтобы переименовать или удалить породу агентов щелкаем на выпадающую стрелку на вкладке породы (рис. 36).

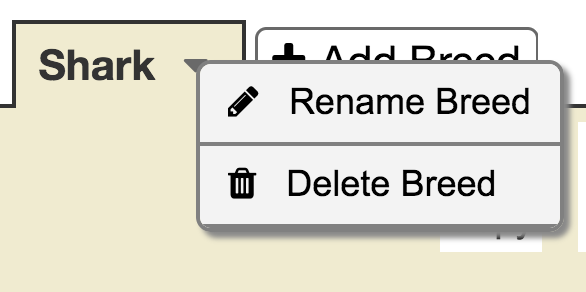


Рисунок Создаем или удаляем породу

#### Задаем свойства по умолчанию или выбираем специальные свойства

При выборе любого свойства, значение которого контролируется числами (размер, положение по оси x, y,z и другие) необходимо написать нужное число в блоке “set my (trait) to \_\_”. При выборе цвета или формы нужно использовать блоки “built-in trait \_\_” или “color \_\_” blocks (traits) и выбрать необходимое свойство в выпадающем меню. Вы можете выбирать свойства породы – цвет, размер или добавить новое свойство в выпадающем меню “Show Traits” на вкладке породы. При этом для каждого свойства необходимо указать начальные значения (рис. 37).

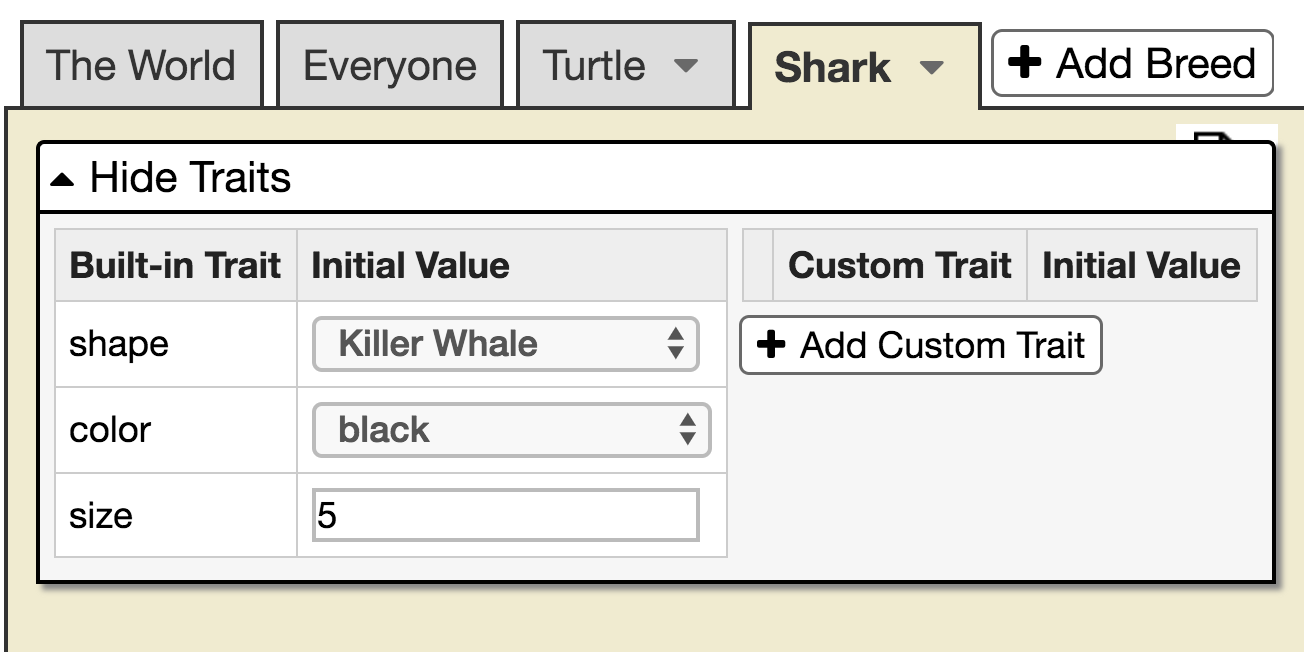


Рисунок Свойства породы

#### Как просмотреть перечень возможных команд?

Все управляющие команды собраны в разноцветных вкладках (рис. 38):

|  |  |
| --- | --- |
| Изображение выглядит как снимок экрана  Автоматически созданное описание  Рисунок Вкладки | Агенты – создание и уничтожение агентов  Detection – реагировать на агентов, которые с тобой соприкасаются  Environment – окружающая среда  Interface  Keyboard – ответ на нажатие клавиш  Lists  Logic – управляющие и логические структуры  Math - числа  Movement – передвижение агентов  Procedures – создание процедур  Sounds - звуки  Traits – свойства породы и агента  Variables - переменные  Debugger – отладчик ошибок |

### Как создать и уничтожить агента?

Действия над агентами представлены на рис. 39

|  |  |
| --- | --- |
| Изображение выглядит как снимок экрана  Автоматически созданное описание  Рисунок Агенты | Создаем и удаляем  Создать (N) агентов породы  Создать (N) агентов породы и пусть каждый сделает перечень действий () – например, каждый устанавливает для себя определенные свойства  Удалить – команда умереть к себе самому (die)  Удалить (уничтожить) всех агентов  Удалить агента с конкретным ID номером  Встать в случайную точку местности  Распределить всех агентов по местности  Взять камеру и рассматривать местность с точки зрения данного агента  Данный агент  Агент, который является родителем данного |

#### Как создать и изменить свойства породы?

У каждой породы есть обязательные свойства, которые должны быть у каждого агента – это форма, цвет и размер. Кроме того, мы можем добавлять породе любые свойства, которые покажутся нам нужными для работы модели. Например, на следующем рисунке видно, как мы добавили породе Person свойство Team – и теперь игрока на местности нашего Мира могут относиться к разным командам (рис. 40).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Свойства породы

### Как придать агентам нужную форму (цвет и размер)?

Команды управления свойствами агента собраны на голубой вкладке Traits -см. рисунок 41. Здесь мы можем устанавливать как обязательные, так и дополнительные свойства каждой породы агентов.

Самый простой случай – Set my Size to (5) – установить размер данного агента.

Если мы хотим установить цвет, то Set my Color to [color (выбор цвета)]

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Блоки управления свойствами агентов

При выборе формы для агента можно использовать встроенную библиотеку со множеством форм, а также загружать в систему самостоятельно созданные формы (рисунок 42).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Выбор формы агентов

#### Как перемещаются агенты?

Команды перемещения агенты почти не отличаются от обычных команд к Черепашке или котенку Scratch – вперед, назад, налево, направо (рис 43). Поскольку среда трехмерная, то есть еще команды вверх и вниз и команда телепортации в трехмерную точку x:y:z. Очень полезная и привычная для пользователей среды Scratch команда «повернутся лицом к (агент)»

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Команды движения

#### Как управлять перемещением агента при помощи клавиатуры?

Например, если нажата кнопка -> вперед (10) – необходимые команды при программировании игр (рис. 44).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Управление при помощи клавиш

#### Как получить и изменить число?

Стандартный набор арифметических действий над числами (рис.45).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Действия над числами

#### Как организовать управление при помощи логических блоков?

Перечень логических блоков приведен на рисунке 46.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Логические блоки StarLogo Nova

Покажем, как используются арифметические и логические блоки в ситуации, когда мы хотим, чтобы создаваемый игрок с вероятностью в 50% относился к команде 1 (красного цвета) или команде 2 (синего цвета) (рис. 47):

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Использование блока if-else при выборе команды

#### Как организовать взаимодействие агентов?

Команды управления взаимодействия агентов и определения агентов, обладающих определенными свойствами, представлены на рисунке 48.

|  |  |
| --- | --- |
| Изображение выглядит как снимок экрана  Автоматически созданное описание  Рисунок Взаимодействия | Если столкнулся с агентов породы [], то делай следующее:  Тот, с кем столкнулся. Например, delete collidee  Число агентов породы [] на расстоянии []  Число агентов породы [] на расстоянии [] с определенным свойством (Например, Team 1)  Ближайший агент породы [] на расстоянии []  Число агентов породы [] на расстоянии [] с определенным свойством |

Мы можем задать программы, управляющие перемещением агентов, можем научить агентов реагировать на свойства поля, по которому они перемещаются. Мы можем определить действия, которые будут совершаться в системе при столкновении агентов. Такие столкновения в системе называются коллизиями. Если мы не указываем агентам как действовать в случае столкновений, то они будут просто проходить через друг друга. Если же мы воспользуемся блоком коллизий, то мы сможем программировать события, которые будут происходить при столкновениях агентов данной породы с агентами этой же или других пород. Например, на следующем примере показана модель распространения инфекции, когда синие «здоровые» черепахи сталкиваются с «больными» красными и в результате столкновения меняют свой цвет. В следующем блоке команд (рисунок 49) черепаха при столкновении проверяет цвет агента, с которым она столкнулась, и если этот цвет красный, то она и сама окрашивается в красный цвет.

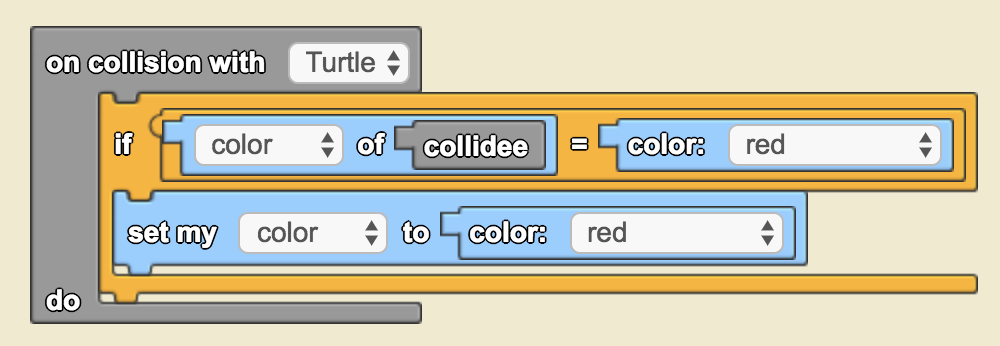


Рисунок Выбор действий при столкновении

На рисунке 50 показана возможная последовательность действий при столкновении агента из породы лев с агентом из породы кролик. В результате столкновения кролик удаляется из системы, а свойство энергия у льва увеличивается на одну единицу.

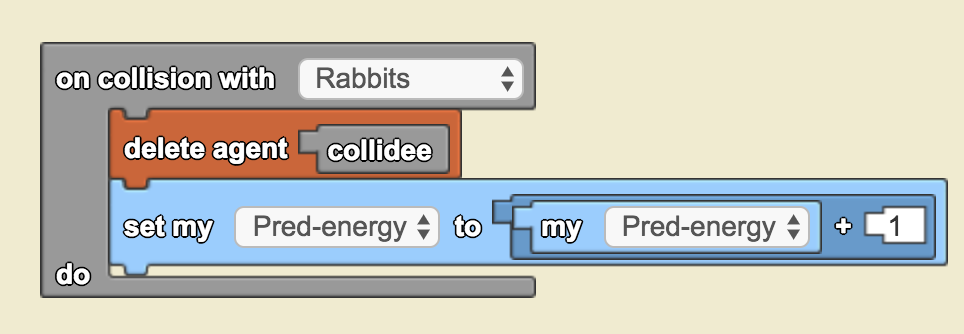


Рисунок Столкновение льва с кроликом

В следующей процедуре (рисунок 51), если наш агент чувствует, что на расстоянии 2-х шагов от него число игроков больше 0, то он вызывает процедуру записи в журнал и передает ей в качестве параметра информацию о ближайшем игроке на расстоянии 2-х шагов.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Реакция на присутствие агента

## Поле (terrain)

Мир StarLogo Nova для каждой модели создает поле из 101 пятна (патч - patch) в длину и 101 пятна в ширину. Центр этого поля имеет координаты 0, 0. По умолчанию все новые агенты создаются в самом центре экрана на уровне земли (x, yz = 0). Команды окружающего мира на рисунке 52.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Команды окружающего мира

Агент могут рассеиваться по поверхности поля при помощи блоков команд “set my x” и“set my y” или при помощи блоков scatter – рисунок 53.

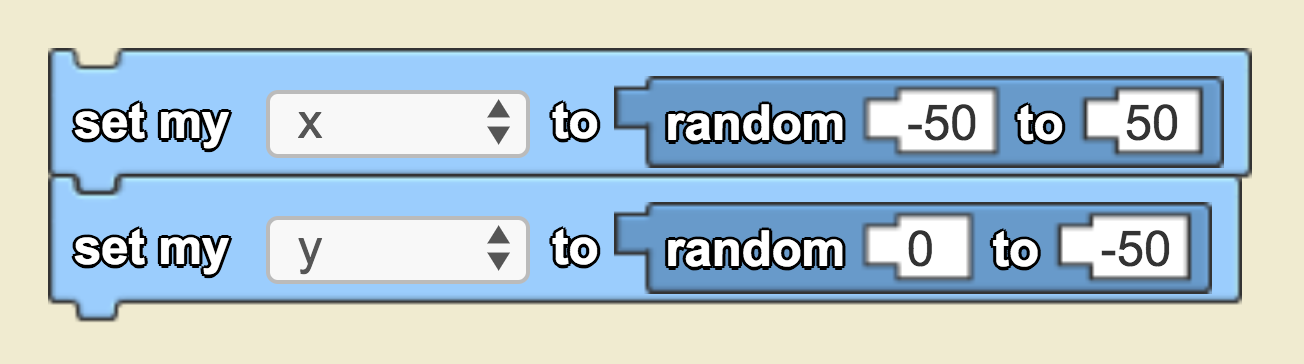


Рисунок Установка агентов по полю

Мы можем посмотреть на поле с различных точек зрения. Например, мы можем поручить любому агенты «взять» камеру и посмотреть на мир с точки зрения этого агента. Мы не можем изменить цвет поля отдельной командой, но мы можем поручить эту работу отдельной черепахе – окрасить её в нужный нам цвет и велеть ей перемещаться по полю и окрашивать кусочки поля, на которых она находится в свой цвет. После окраски поля мы удалим эту черепаху.

#### Как создать и вызвать процедуру?

Блоки создания и вызова процедур представлены на рисунке 54.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Создание и вызов процедур

Процедура создается из множества команд. Чтобы использовать процедуру, её необходимо вызвать через команду call (имя процедуры).

Процедура при её объявлении может включать несколько параметром. Например, в следующей процедуре (рисунок 55) есть параметр Player и Колобок передает в качестве этого параметра данные о ближайшем игроке.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Вызов процедуры с параметром

#### Как создавать виджеты?

Кнопка Edit Interface – и мы получаем возможность управлять виджетами на экране – создавать, удалять, очищать значения (рис. 56).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Редактирование интерфейса

По кнопке Create Widget (рис. 57) мы получаем доступ к созданию кнопок, переключателей, рычажков, ящиков данных и линейных графиков.

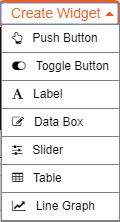


Рисунок Создаем виджеты

Разместите новый виджет на поле экрана, задайте его имя и необходимые параметры. После создания виджета, щелкните ссылку “Lock Interface” и покиньте режим редактирования виджета. Теперь вы можете добавить нужный код к своему новому виджету. На рисунке 58 Мы создали виджет-таблицу School diary, в которой есть серии для записи ID-игрока, его координат на местности, направления и номера команды.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Создание таблицы данных

#### Как управлять виджетами?

В ходе работы программы состояния виджетов используются командами из зеленой вкладки Interface – когда нажмут кнопку, пока кнопка нажата, использовать значение рычажка и т.п. Кроме того, командные блоки интерфейса позволяют добавлять данные в таблицы и графы.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Управление виджетами

### Создаём виджеты-рычажки

Если вы создаете рычажок (slider), то установите для него минимальное и максимальное значение, исходное значение (value) и шаг изменений (step). Теперь вы можете использовать значение slider обращаясь к имени этого рычажка и используя его для определения количества агентов, вероятности событий и т.п. (рис. 60).

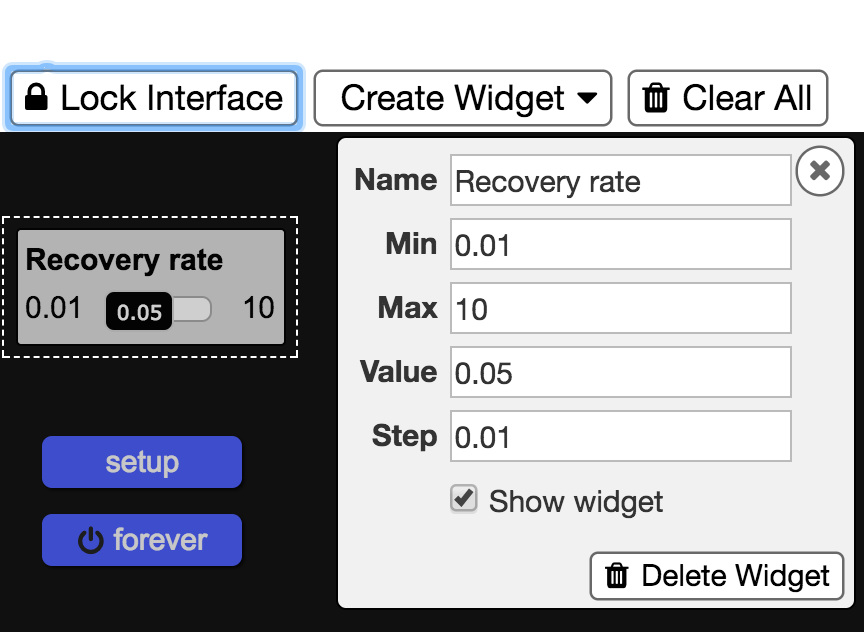


Рисунок Обозначаем границы рычажка

Теперь мы можем обращаться к значению рычажка из скриптов, управляющих поведением агентов. Например, мы можем установить скорость события (рис. 61).

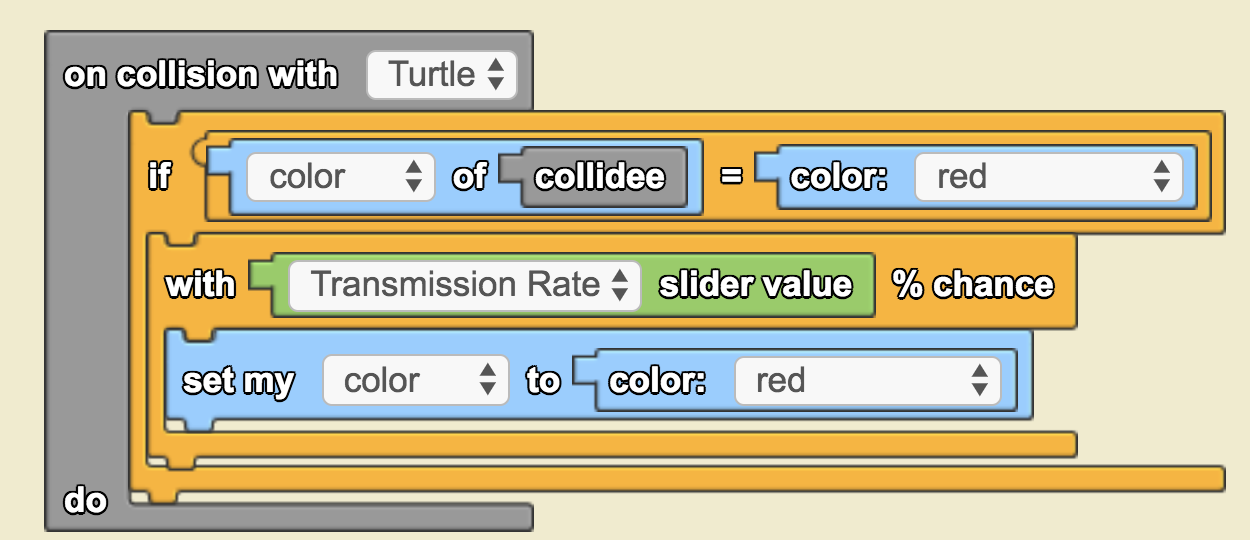


Рисунок Используем значение виджета

Как использовать рычажок для изменения заданного значения в процентах?

Можно использовать значение рычажка внутри конструкции “with \_\_\_ % chance”, как это показано на предыдущем рисунке, чтобы значение рычажка контролировало шансы события. Другой вариант состоит в том, чтобы создать рычажок и выставить его максимальное значение в 100. В этом случае значение рычажка будет задавать значение переменной по следующей формуле (рис. 62).

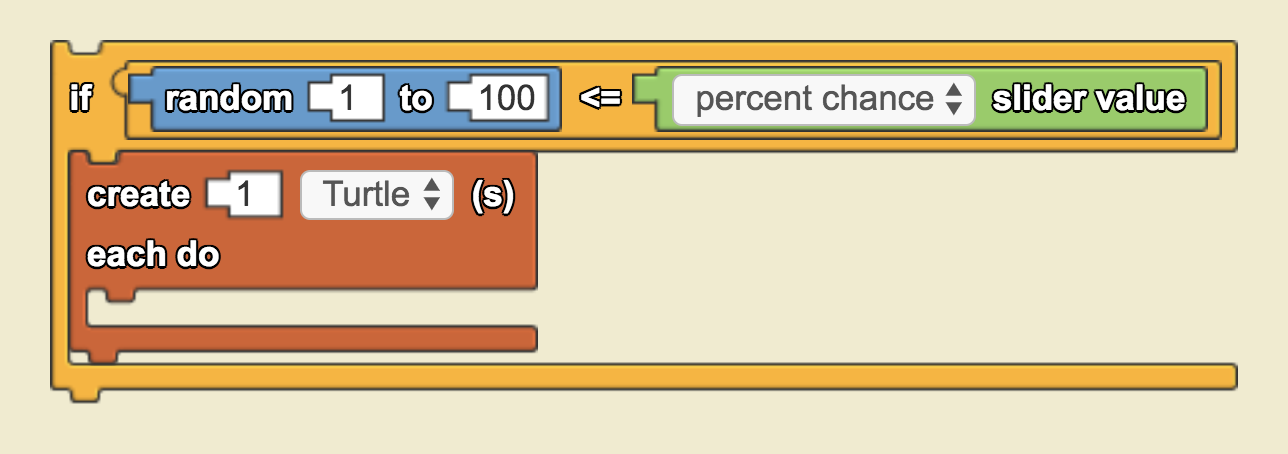


Рисунок Определение значения переменной через виджет

### Создаём виджеты – ящики данных

Чтобы собирать данные создайте и назовите виджет типа Data Box. Чтобы очистить ящик данных, используйте в коде setup конструкцию “set \_\_\_ data box to \_\_\_” и установите значение в 0. Для того, чтобы выбрать нужный ящик данных, используйте выпадающее меню (рис. 63).

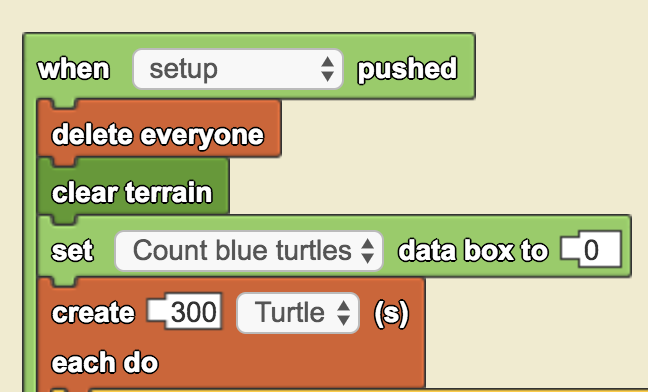


Рисунок Связь скрипта и виджета ящика данных

На странице мира (World) встройте блок с использованием ящика данных внутрь блока

“while forever toggled” и этот блок будет наполнять ящик данных данными об агентах с определенными характеристиками. В данном случае – это черепахи голубого цвета в радиусе 150 шагов (рис. 64).

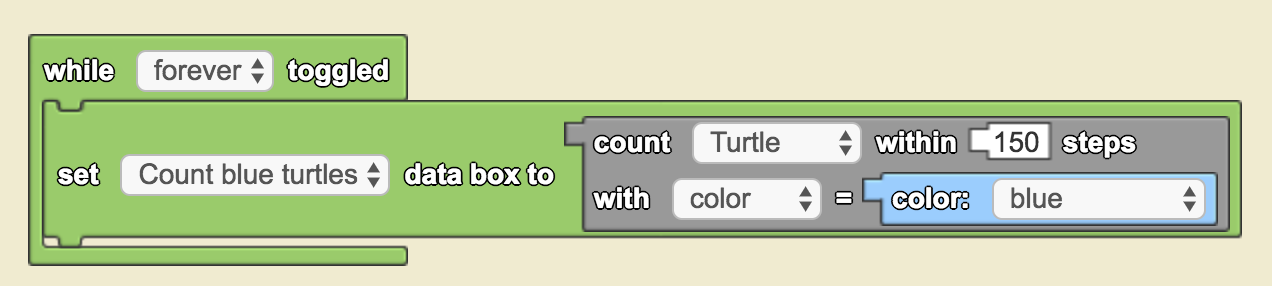


Рисунок Помещаем в ящик число черепах нужного цвета

Агент Мир (World) будет считать всех черепах на заданном расстоянии от себя. При этом координаты этого агента находятся в самом центре экрана (точка с координатами 0, 0).

Вы можете встраивать ящики данных внутрь любых процедур, производящих изменения. Например, на следующем примере показано, как число агентов – капель воды добавляется к ящику данных (рис. 65).

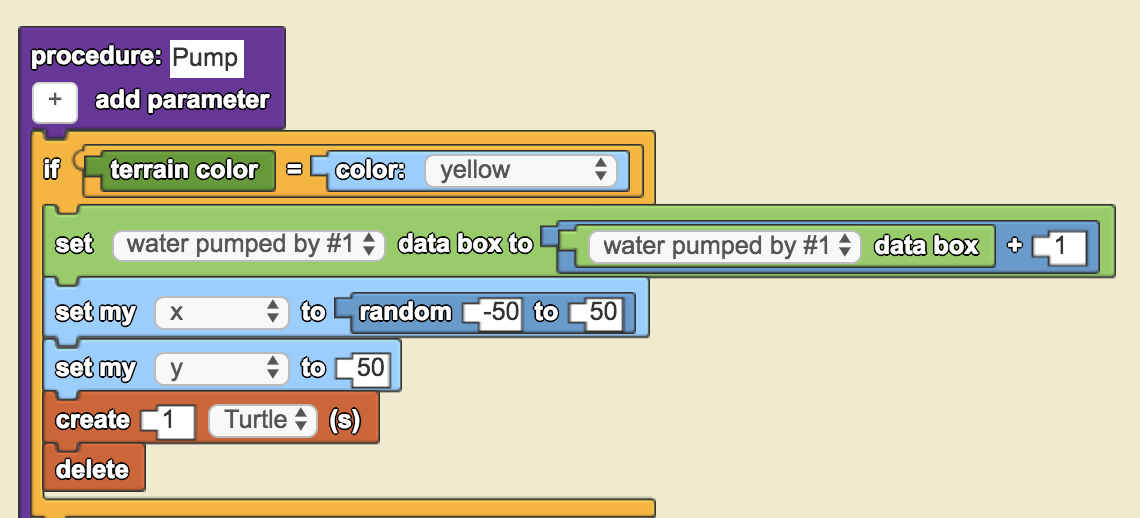


Рисунок Использование одним виджетом данных другого виджета

### Создаём виджеты – таблицы и графики

Мы создаем новый виджет, называем его и открываем, чтобы назвать оси графа, серии (линии, которые создаются на графе) и их цвета. Например, на рисунке 66 мы разными цветами обозначаем численность популяций травы, кроликов, волков и медведей.

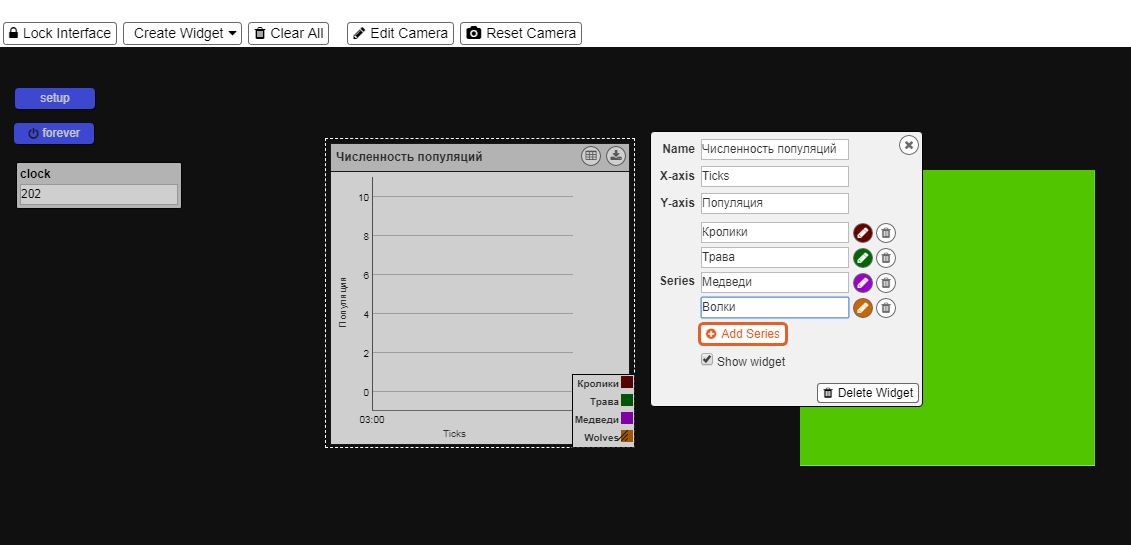


Рисунок Таблицы и графики

После редактирования необходимо щелкнуть по ссылке “Lock Interface” и выйти из режима редактирования.

На странице мира (World) мы используем блок “**add data to line graph**” внутри блока “**while forever toggled**” . Имя нужного графа и нужной серии выбирается при помощи выпадающего меню. Для определения значений по оси y используйте блок “count \_\_\_ within \_\_\_” .

Чтобы агент World гарантированно подсчитал всех агентов на экране, используйте расстояние в 150 шагов. Для определения значения по оси x используйте блок “**clock**” (рис. 67).

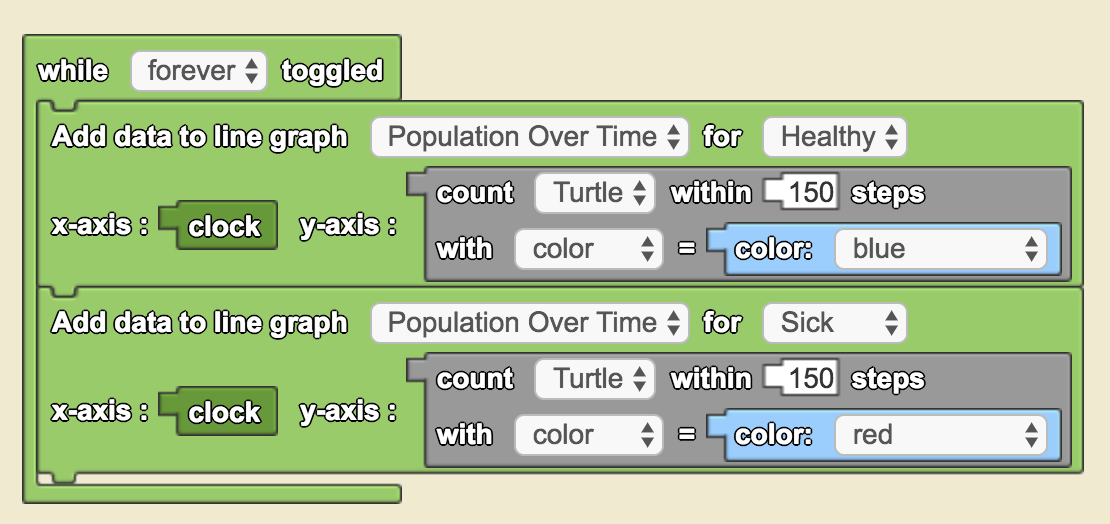


Рисунок Используем блок clock

В начале действия модели мы очищаем граф по кнопке setup, используя команду “clear graph” и устанавливаем clock в 0.

Линейный граф всегда может быть преобразован в таблицы или сохранен в виде csv файла (рис. 68).

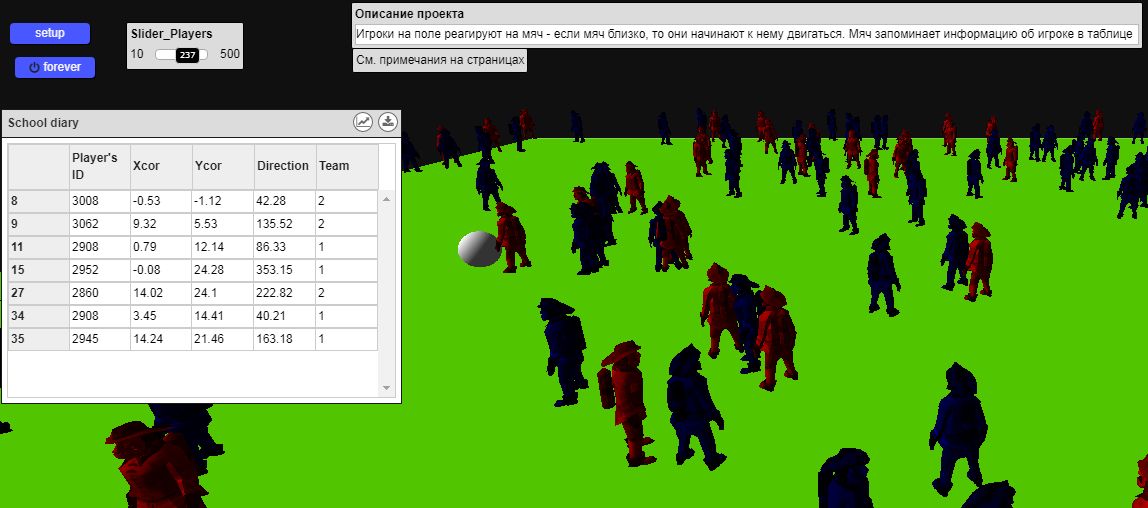


Рисунок Превращение графа в таблицу

Пример того, как мы можем работать с линейными графиками и таблицами, обращаясь к ним из скриптов отдельных агентов, приведены на рисунке 69.

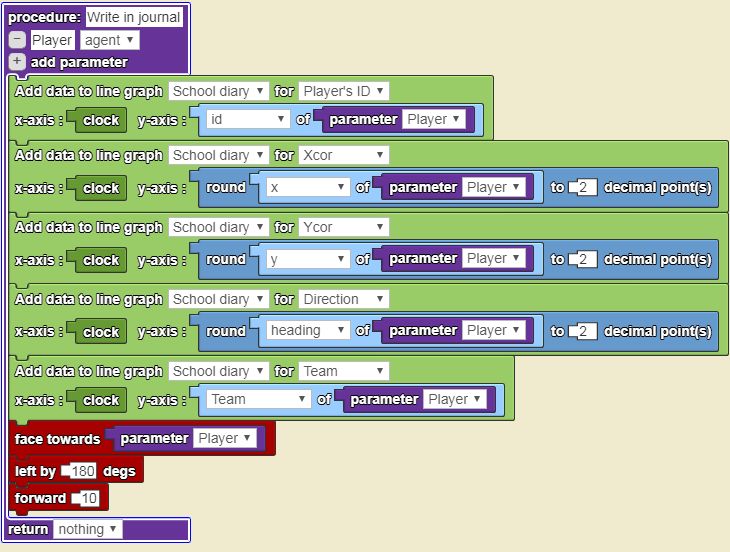


Рисунок Записываем данные в таблицу

#### Как и где сохранять данные?

На рисунке 70 представлена процедура, которая будет записывать в журнал данные о тех игроках, которых агент Колобок встретил на футбольном поле. Процедура запускается самим Колобком, когда он чувствует, что рядом с ним кто-то есть. В этом случае Колобок записывает в соответствующие ячейки таблицы данные об игроке.

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Запись данных в таблицу

# Приложение 3 Многоагентное моделирование в среде NetLogo

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 макроуровне. Интерфейс и начальный этап создания управляющих виджетов представлены на рисунке

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Интерфейс и виджеты NetLogo

Интерфейс (Interface - область, где мы исследуем свойства модели. В центре черное по умолчанию поле, на котором взаимодействую черепашки - turtles, пятна - patches и связи черепашек - links. На рабочее поле мы можем добавлять различные виджеты - кнопки, рычажки, графики, окна входа, окна результатов. Кнопки привязываются к именам процедур.

Вкладка «Инфо» содержит описание модели. Здесь авторы модели рассказывают - о чем модель, как она устроена, какие особенности, как можно использовать и развивать модель.

Вкладка «Код» - область, где записываются процедуры NetLogo, управляющие поведением агентов.

### Виджеты NetLogo

Процедура создания виджетов представлена на рисунке 71 «Интерфейс и виджеты NetLogo». Чтобы создать виджет нужно нажать на правую кнопку мыши и выбрать нужный тип виджета.

**Кнопка - button** - связана с процедурой на странице Procedures. Когда создается кнопка, мы можем указать какому из агентов она отдает указание (Обозреватель, Черепахи, Пятна, Связи). Имя команды и имя кнопки на экране могут не совпадать. Например, мы можем создать процедуру setup и на экране задать имя кнопки - "Начнем". Если кнопка создана и для нее еще нет процедуры, то кнопка окрашена в красный цвет.

**Рычажок – Slider.** Следующая кнопка - рычажок - slider - меняя положение рычажка можно изменять параметры модели. Например, мы хотим, чтобы число можно было при помощи этой кнопки менять число черепах. Рычажок задает имя глобальной переменной и может содержать только латинские буквы. Здесь же указывается интервал, в котором изменяются значения. Например, от 1 до 500

to setup

ca

cro num\_turtles [fd 5]

end

**Переход Switch**. Переключает положение для глобальной переменной. Это значение пользователь на экране может поставить в положение ON/OFF

to setup

ca

cro num\_turtles [fd 5]

ifelse green? [ask patches [set pcolor green]]

[ask patches [set pcolor pxcor]]

end

Здесь показано использование кнопки для организации ветвления. Если переменная green? включена, то пятна экрана окрашиваются в зеленый цвет. Если значение переменной отключено, то пятна окрашиваются в другие цвета. В примере показано, как мы можем окрасить пятно в цвет, выбрав значение цвета из координат этого пятна.

to setup

ca

cro num\_turtles [fd 5]

ifelse green? [ask patches [set pcolor green]]

[ask patches [

if pycor != 0

[set pcolor pxcor / pycor] ]]

end

**Выбор – Chooser.** Выбор значения глобальной переменной. Переключатель задает имя глобальной переменной и может содержать только латинские буквы. В этом примере мы создаем tcolor и указываем значения, которые может принимать tcolor - 5, 15, 25, 35, 45. Создадим еще одну кнопку, которая будет проверять цвет черепашки. Теперь жизни черепашек в наших руках. Все, чей цвет меньше выбранного, при нажатии кнопки go будут погибать.

to go

ask turtles with [color < tcolor]

[die]

end

**Ввод – Input.** Кнопка позволяет пользователю вводить или выбирать значение. Например, создадим кнопку incolor, в которой можно изменять или выбирать значение цвета. Создадим еще и кнопку "Новые черепахи" и свяжем ее с процедурой hatch\_new

to hatch\_new

ask turtles with [color = incolor]

[hatch 2 [rt random 90 fd 2]]

end

Теперь мы можем увеличивать численность черепах заданного цвета. Если цвет черепахи совпадает с выбранным пользователем цветом, то эта черепаха рождает двоих потомков.

**Экран – Monitor.** Окно, в котором показывается выбранное значение. Например, мы создаем monitor, который показывает общую численность всех черепах - count turtles.

**График – Plot.** Окно, в котором в виде графика отражаются численные значения. Примеры графиков приведены на рисунке 72.

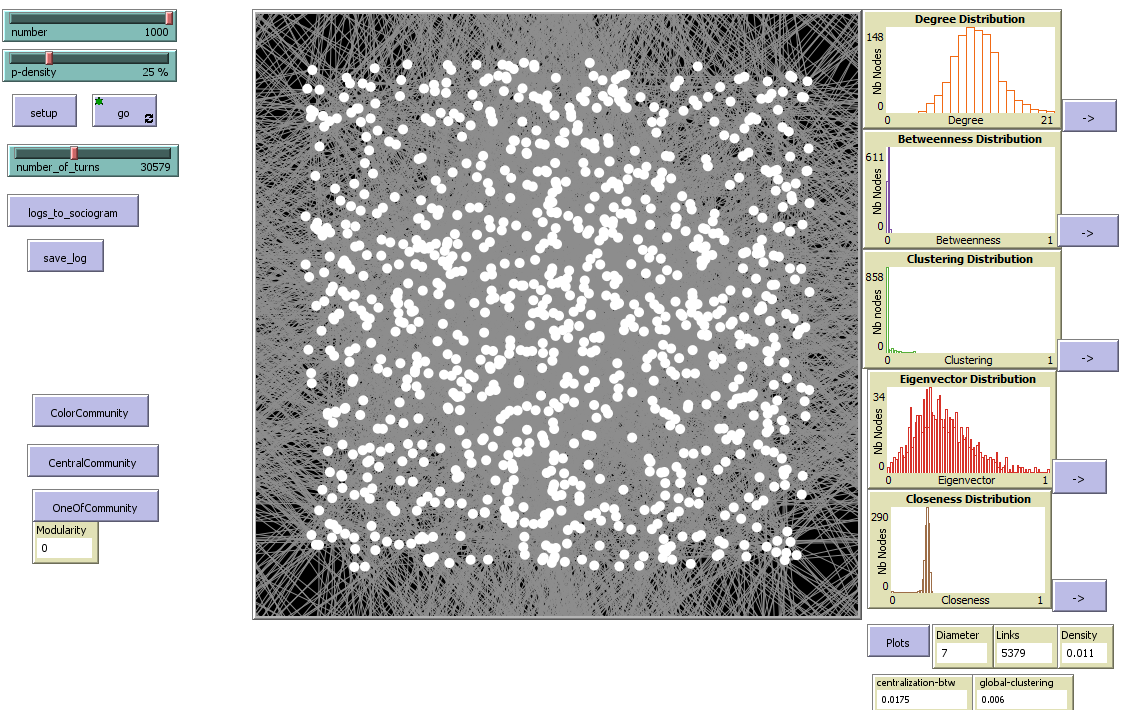


Рисунок Примеры виджетов- графиков

### Библиотека NetLogo

Библиотека NetLogo содержит множество готовых моделей по биологии, математике, химии, социологии (рис.73).

Изображение выглядит как снимок экрана

Автоматически созданное описание

Рисунок Пример каталога библиотеки моделей NetLogo

### Создаем и группируем черепах

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 2 значения - ложь или истина. Принадлежит черепашка к группе или не принадлежит.

turtles-own [group]

Внутри turtles-own [] можно перечислить множество свойств, которыми будут обладать черепашки. Если мы создали новую породу черепах, то с этой породой точно так же можно создать её свойства.

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

Создаем новых черепах. Черепахи создаются по порядку и поворачиваются последовательно. Когда мы им потом даем команду разойтись, они образуют круг

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

]

cro number\_of\_turtles [fd max-pxcor - 2]

end

Этого же результата можно достичь, если мы используем команду layout-circle

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. Первая версия - черепашки объединяются, но делают много лишних движений

to collect\_color

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

end

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 = [color] 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 (patches)

Пятна (patches) – это агенты NetLogo, которые связаны с конкретными координатами на экране. К пятну или точке всегда можно обратиться, указав его координаты. Например:

ask patch 0 0 [set pcolor yellow]

Для пятен существуют ограничения. Например, невозможно велеть пятну умереть и создать новое пятно. Однако, пятно может родить новую черепашку (sprout) или черепашку какой-то специальной породы (sprout-breed) - По команде sprout в данной точке появляется черепашка

**Ключевые команды для пятен**

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 4-х соседей - ближайшие соседние точки по прямой

import-pcolors ;; считать цвета из файла - картинки

patch-at dx dy ;; пятно с координатами относительно данного агента

Например,

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 ;; пятно по направлению и на расстоянии

ask patch-at-heading-and-distance -90 1 [ set pcolor green ]

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

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

### Процедуры NetLogo

В NetLogo есть команды, которые указывают агентам, что делать - переместиться, создать черепашку, установит связь и т.д. Обычно эти команды начинаются с глагола - die, jump, inspect

Процедуры можно создавать - в этом случае процедура начинается со слова To Имя\_Процедуры и заканчивается словом End

Кроме того, есть процедуры, которые получают на входе данные, обрабатывают эти данные и выдают результат. Эти процедуры обязательно начинаются со слова to-report и содержат команду вывода значения – report.

to-report absolute-value [number]

ifelse number >= 0

[ report number ]

[ report (- number) ]

end

В этом примере процедура *absolute-value* получает число и возвращает его абсолютное значение.

Анонимные процедуры. С версии 6.0 в NetLogo появились анонимные процедуры. Например,

to logs\_to\_sociogram

ask patches [set pcolor 0]

foreach edits [ [?1] -> ;; это вызов анонимной процедуры

let friend1 item 0 ?1

let p1 item 1 ?1

let friend2 first first filter [ [??1] -> (p1 = item 1 ??1) and ("create" = item 2 ??1) ] wikilog

if friend1 != friend2 [

ask turtle friend1 [ create-friendbond-to turtle friend2 ]

] ]

repeat 2 [layout-spring turtles links 1 10 12 ]

end