УДК 004:37

Мультиагентная модель в системе NETLOGO: исследование и обучение

Паршин Андрей Дмитриевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Баженов Руслан Иванович
Приамурский государственный университет имени Шолом-Алейхема
К.п.н., доцент, зав. кафедрой информационных систем, математики и правовой информатики

Аннотация
В статье рассматривается исследование мультиагентной модели в системе NETLOGO. В данной работе описано создание системы поведения персонажей, подробно описаны коды и проведено исследование на основе созданной модели. Показана возможность использования модели в обучении студентов.
Ключевые слова: модель, netlogo, игры, персонаж, поведение.

Multi-agent model in NETLOGO system: research and training

Parshin Andrey Dmitrievich
Sholom-Alechem Priamursky State University
Student

Bazhenov Ruslan Ivanovich
Sholom-Alechem Priamursky State University
Candidate of pedagogical sciences, associate professor, Head of the Department of Information System, Mathematics and legal informatics

Abstract
The article examines the study of the multiagent model in the NETLOGO system. In this paper, we describe the creation of a character behavior system, the codes are described in detail, and a study based on the model created. The possibility of using the model in teaching students is shown.
Keywords: model, NetLogo, games, character, behavior.

В современном мире наиболее развивающая отрасль – это отрасль компьютеров. Их внедряют в самые разные по выполнению задач отрасли. Не обошли стороной и сферу развлечений. Благодаря возросшей мощности в играх стала реальна имитация поведения живого игрока. Простейшей формой искусственного интеллекта является система на основе правил. Такая система дальше всего стоит от настоящего искусственного интеллекта. Набор

заранее заданных алгоритмов определяет поведение игровых объектов. С учетом разнообразия действий конечный результат может быть неявной поведенческой системой, хотя такая система на самом деле вовсе не будет «интеллектуальной».

Поэтому для того чтобы убедится в правильности выбранных решений нужно протестировать поведение системы. В данном исследовании мы воспользуемся netlogo. Также важно научить будущих специалистов в области ИТ применять технологии,основные на мультиагентных моделях, заложенных в netlogo.

Исследованиями в данной теме занимались многие авторы. А.В. Кузнецов рассматривал примеры многоагентных моделей разных типов [1]. Т.А Золотова и Н.И. Ефимова Рассматривали особенности конструирования персонажей компьютерных игр [2]. А.Д. Караев и Д.А. Караева использовали нейронную сеть для создания интеллектуального бота для игры 2048 [3]. Д.Е Ушаков, О.В. Копченов, Е.А. Лазарев разрабатывали искусственный интеллект для агентов, моделирующих поведение животных в компьютерных играх [4]. Е.И. Сальникова рассматривала этапы разработки персонажей для двумерных компьютерных игр и их основные особенности [5]. Giovanni Acampora, Vincenzo Loia, Autilia Vitiello рассматривали возможность улучшения поведения игрового бота через синхронизированный эмоциональный интеллект [6].

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.

Для начала работы следует запустить netlogo. Интерфейс программы имеет 3 вкладки: «интерфейс» где находятся инструменты для наблюдения за поведением модели, вкладка «инфо» содержит информацию о модели от автора и вкладка «код» содержащая программный код модели.

Рассмотрим задачу о рыцарях и драконах.

Создадим элементы управления.

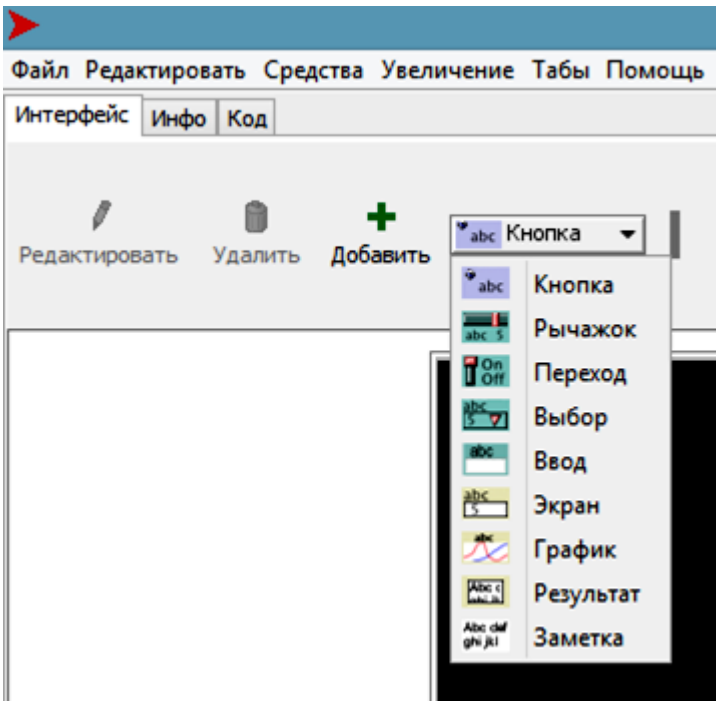

Рис.1

Создадим кнопку «setup». В редактировании укажем название и команду. Данная кнопка будет обновлять модель.

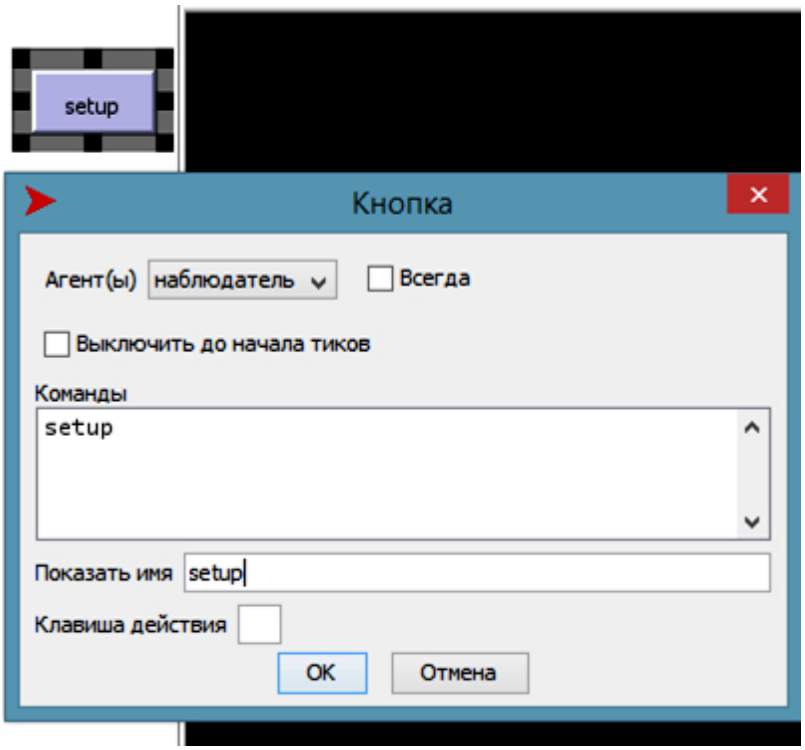

Рис.2

Также создадим кнопку «go». Она будет запускать модель. Поэтому при редактировании нужно будет указать, что модель не остановится пока кнопку не нажмут ещё раз.

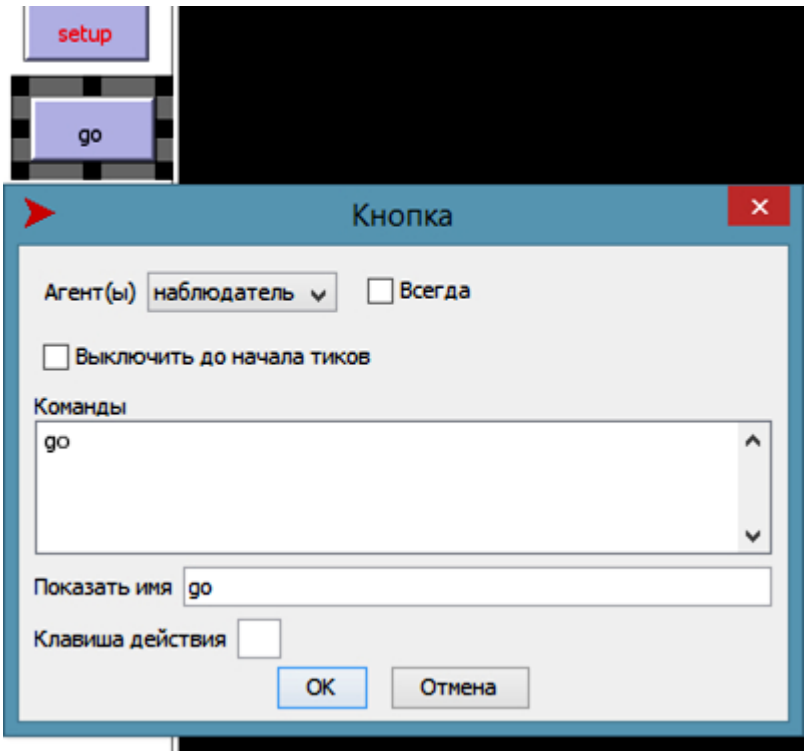

Рис.3

Дальше создадим рычажки, которые будут задавать некоторые значения для модели.

Первый рычажок - количество рыцарей.

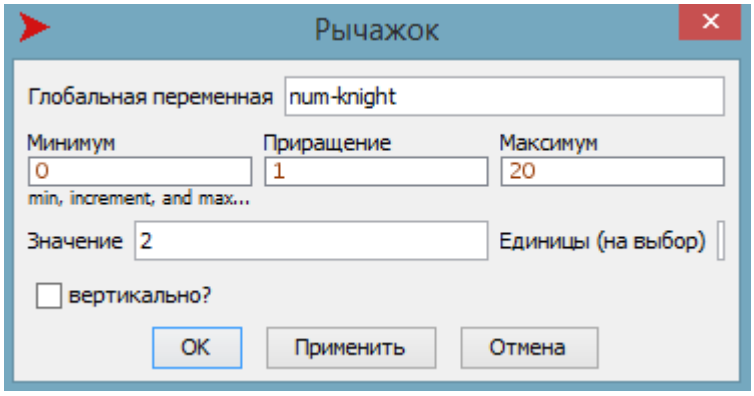

Рис.4

Второй рычажок - количество стартового здоровья рыцаря.

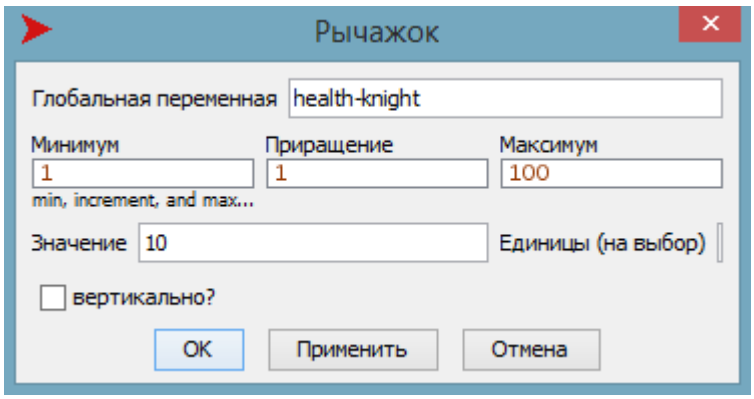

Рис.5

Третий рычажок - количество восстанавливаемого здоровья при остановке.

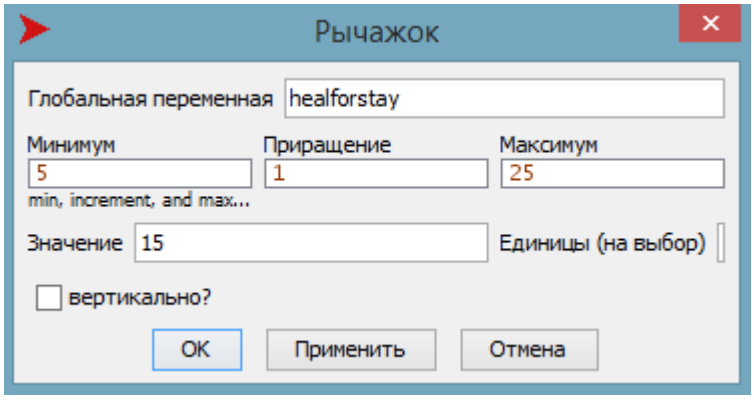

Рис.6

Четвёртый рычажок – максимальный уровень, достигнув которого рыцарь остановится.

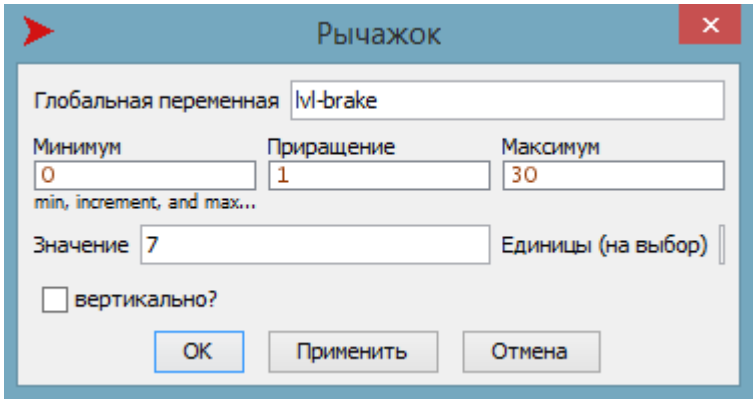

Рис.7

Также нам понадобится экран отображающий среднее количество тиков на достижение уровня.

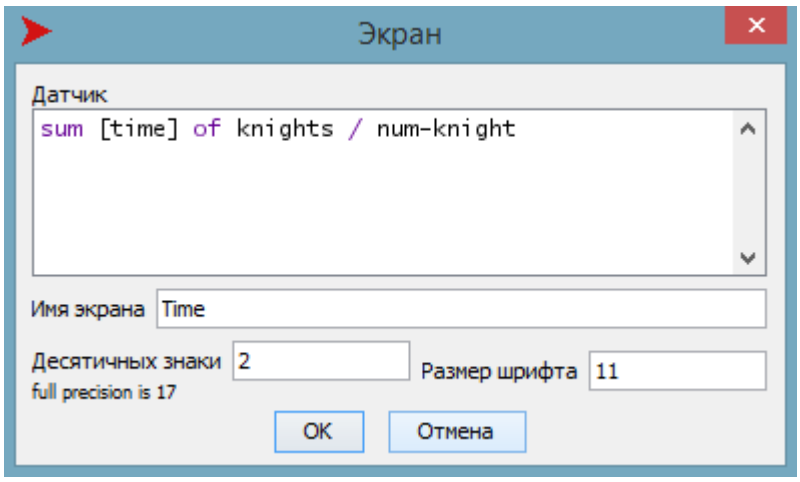

Рис.8

Закончив с интерфейсом, переходим к написанию кода модели.

В начале описываем переменные и создаём породу черепах.

```
breed [knights knight]
knights-own [health expirience level movetrigger
  fighttrigger dragontrigger healtrigger deathscore lvl-stop time]
```

Далее опишем процедуры, которые используют кнопки.

Процедура «setup» будет очищать модель, создавать рыцарей, определять их форму и значения переменных. Процедура «go» будет проверять условие для остановки, воспроизводить модель и задавать первоначальные процедуры рыцарям.

```
to setup
  clear-all
  set-default-shape knights "person"
  create-knights num-knight [
  setxy random-xcor random-ycor
  set health health-knight
  set expirience 0
  set level 1
  set movetrigger 1
  set fighttrigger 0
  set dragontrigger 0
  set healtrigger 0
  set deathscore 0
  set lvl-stop 0
  set time 0
  ]
  display-labels
  reset-ticks
end

to go
  if not any? knights with [lvl-stop = 0] [stop]
  ask knights [ trigger death lvlup]
  tick
end
```

Закончив с интерфейсом, переходим к написанию кода модели.

В начале описываем переменные и создаём породу черепах.

```
breed [knights knight]
knights-own [health expirience level movetrigger
  fighttrigger dragontrigger healtrigger deathscore lvl-stop time]
```

Далее опишем процедуры которые используют кнопки.

Процедура «setup» будет очищать модель, создавать рыцарей, определять их форму и значения переменных. Процедура «go» будет проверять условие для остановки, воспроизводить модель и задавать первоначальные процедуры рыцарям.

```
to setup
  clear-all
  set-default-shape knights "person"
  create-knights num-knight [
  setxy random-xcor random-ycor
  set health health-knight
  set expirience 0
  set level 1
  set movetrigger 1
  set fighttrigger 0
  set dragontrigger 0
  set healtrigger 0
  set deathscore 0
  set lvl-stop 0
  set time 0
  ]
  display-labels
  reset-ticks
end

to go
  if not any? knights with [lvl-stop = 0] [stop]
  ask knights [ trigger death lvlup]
  tick
end
```

Для рыцарей изначально заданно 3 процедуры.

```
to trigger
  if movetrigger = 1 [ move encounter ]
  if fighttrigger = 1 [fight]
  if dragontrigger = 1 [dragon]
  if healtrigger = 1 [heal]
end

to lvlup
  if expirience >= (level * 3) * (level * 5)
  [
    set level level + 1
    set expirience 0
  ]
  if level >= lvl-brake
  [
    setxy 0 0
    set lvl-stop lvl-stop + 1
    set movetrigger 0
    set fighttrigger 0
    set dragontrigger 0
    set healtrigger 0
  ]
  if level < lvl-brake
  [ set time time + 1]
  if lvl-stop = 1 [
    set time time + 1
    set lvl-stop 2]
end

to death
  if health <= 0
  [
    setxy 0 0
    set health 1
    set deathscore deathscore + 1
    set fighttrigger 0
    set dragontrigger 0
    set healtrigger 1
  ]
end
```

Процедура «lvlup» повышает уровень рыцаря если его опыт достиг границы.

Процедура «death» перемещает рыцаря в центр, если его здоровье равно или ниже 0.

Процедура «trigger» отвечает за то какое действие совершит рыцарь в следующий тик.

Если переменная «movetrigger» равна 1, то рыцарь совершит движение в случайную сторону после чего с ним произойдёт случайное событие.

```
to move
  let chance random 4
  if chance = 0 [
    set ycor ycor + 1
  ]
  if chance = 1 [
    set xcor xcor + 1
  ]
  if chance = 2 [
    set ycor ycor - 1
  ]
  if chance = 3 [
    set xcor xcor - 1
  ]
end

to encounter
  let chance random 100
  if 0 <= chance and chance >= 9 [ set health health + 25 ]
  if 10 <= chance and chance >= 39 [ fight ]
  if 40 <= chance and chance >= 44 [ dragon]
  if 45 <= chance and chance >= 49 [ set health health - 35 needheal]
end
```

Если произошло второе или третье событие то рыцарь перейдёт к битве или битве с драконом. Рыцарь будет сражаться пока не убьёт противника или пока не погибнет.

```
to fight
  set movetrigger 0
  set fighttrigger 1
  let chance random 100
  let killchance 49 + level
  ifelse chance > killchance
    [
      set health health - ( 20 + ( random 50 ) - level)
    ]
    [
    set expirience expirience + level * 3
    set fighttrigger 0
    set movetrigger 1
    needheal
    ]
end

to dragon
  set movetrigger 0
  set dragontrigger 1
  let chance random 100
  let killchance 29
  ifelse chance > killchance
    [
      set health health - ( 20 + random 80 )
    ]
    [
    set expirience expirience + level * 5
    set fighttrigger 0
    set movetrigger 1
    needheal
    ]
end
```

За проверку необходимости восполнения здоровья отвечает процедура «needheal».

```
to needheal
  if health < 50 [
    set movetrigger 0
    set healtrigger 1
  ]
end
```

Если здоровье нужно восполнить, то рыцарь стоит и лечится.

```
to heal
  if health >= 50 [
    set movetrigger 1
    set healtrigger 0
  ]
  set health health + healforstay
end
```

После создания модели было проведено исследование, в котором было определено за сколько времени рыцарь достигает определённого уровня при разных значениях рычажка «healforstay».

Таблица 1. Результаты при значении 10

Уровень	Количество тиков 1	Количество тиков 2	Количество тиков 3
2	19.8	21.95	19.9
4	133.4	136.85	119
6	325.8	312.85	304.25
8	563.55	559.8	563.2
10	916.5	899.85	923.1
12	1314.95	1317.95	1282.05

Таблица 2. Результаты при значении 20

Уровень	Количество тиков 1	Количество тиков 2	Количество тиков 3
2	15	15.3	15.15
4	83.4	93	81
6	219.45	228.9	220.8
8	406.7	395.95	402.75
10	667.2	633.4	640.5
12	945.55	913.35	941.5

Таким образом, можно сделать вывод чем большее количество здоровья восстанавливает рыцарь, тем меньшее количество времени понадобится ему для достижения уровня.

Представленную модель можно использовать в обучении студентов в курсах «Основы искусственного интеллекта», «Интеллектуальные технологии и системы».

Библиографический список

1. Кузнецов А.В. Краткий обзор многоагентных моделей // Управление большими системами: сборник трудов. 2018. № 71. С. 6-44.
2. Золотова Т.А, Ефимова Н.И. Рыцари виртуальной реальности: о некоторых закономерностях конструирования персонажей компьютерной игры // Фольклор XXI века: герои нашего времени. 2013. С. 225-234.
3. Караев А.Д., Караева Д.А. Применение генетического алгоритма для имитации искусственного интеллекта в игре // Студенческая наука: современные реалии. 2017. С. 53-60.
4. Сальникова Е.И. Моделирование поведения животных в компьютерных играх с помощью методов искусственного интеллекта // Молодой ученый. 2018. № 30 (216). С. 10-14.
5. Особенности разработки персонажей для двумерных компьютерных игр // Творчество молодых: дизайн, реклама, информационные технологии. 2014. С. 129-131.
6. Acampora G., Loia V., Vitiello Au. Improving game bot behaviours through timed emotional intelligence // Knowledge-Based Systems. 2012. №. 34. С. 97-113.