

Лекция 1 :: Введение

Ершов Н.М.

ershovnm@gmail.com

Естественные вычисления

- ▶ Естественные вычисления (англ. Natural Computing) — относительно недавно сформировавшаяся теория, которая, однако, включает в себя такие уже классические разделы, как клеточные автоматы, искусственные нейронные сети, генетические алгоритмы и т. д.
- ▶ К естественным вычислениям принято относить различные математические и компьютерные модели и методы, которые, с одной стороны, построены по аналогии с некоторыми природными (физическими, химическими, биологическими, социальными) системами и процессами, с другой — предназначены для моделирования каких-нибудь природных процессов.

Хронология

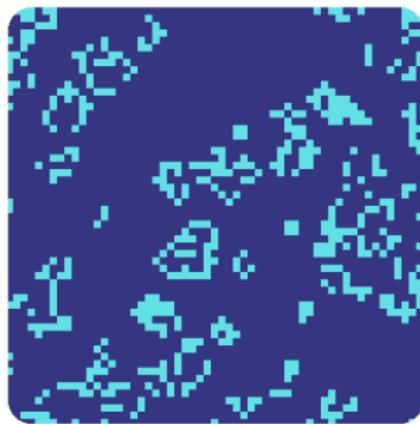
- ▶ Несмотря на то, что как раздел науки теория естественных вычислений все еще продолжает формироваться, история ее отдельных подразделов насчитывает уже десятки лет.
- ▶ Например, одним из авторов исторически первой модели естественных вычислений был не кто иной, как Джон фон Нейман.
- ▶ В 1940-х годах им была предложена модель клеточных автоматов, целью было моделирование процессов самовоспроизведения в живой природе.
- ▶ Таким образом, фон Нейман оказывается автором сразу двух алгоритмических парадигм — архитектуры фон Неймана *последовательного* двоичного компьютера и *высоко параллельных* алгоритмически универсальных (как это было показано позже) клеточных автоматов.

Клеточные автоматы

1948

Джон фон Нейман

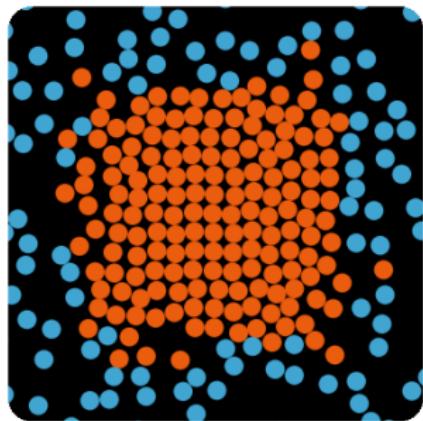
Станислав Улам



Методы Монте-Карло

1949

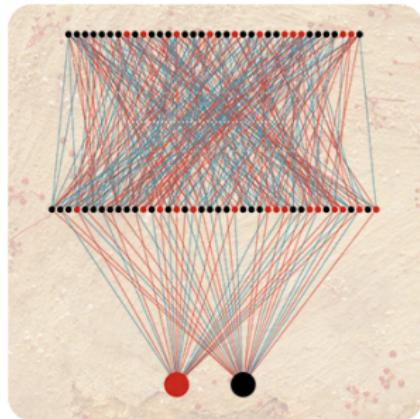
*Джон фон Нейман
Станислав Улам
Николас Метрополис*



Искусственные нейронные сети

1957

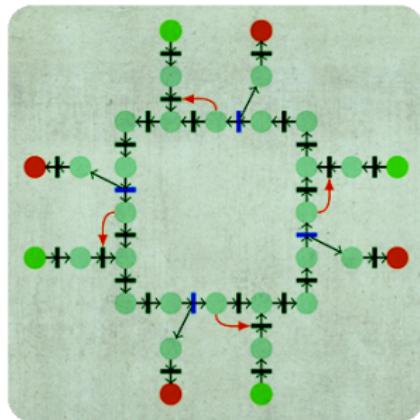
Фрэнк Розенблatt



Сети Петри

1962

Карл Адам Петри



Системы Линденмайера

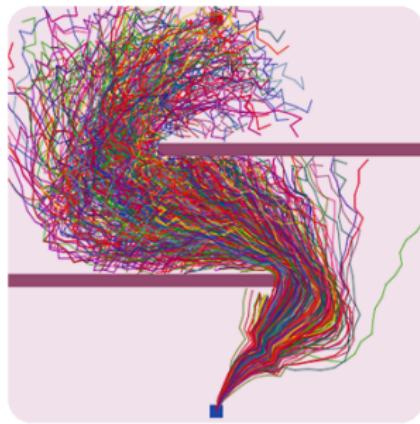
1968

Аристид Линденмайер



1975

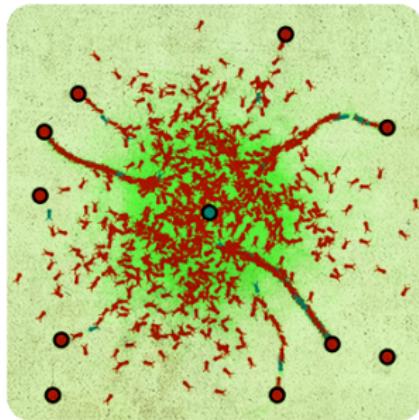
Джон Холланд



Муравьиные алгоритмы

1991

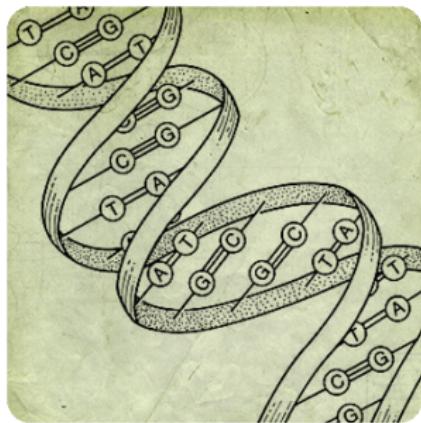
Марко Дориго



ДНК-вычисления

1994

Леонард Адлеман

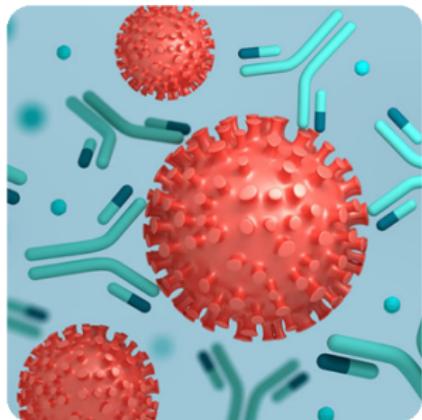


Искусственные иммунные системы

1994

Стефани Форрест

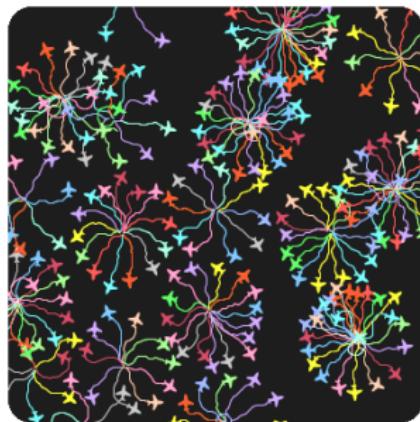
Джефри Кипхарт



Метод роя частиц

1995

Джеймс Кеннеди



Мембранные системы

1998

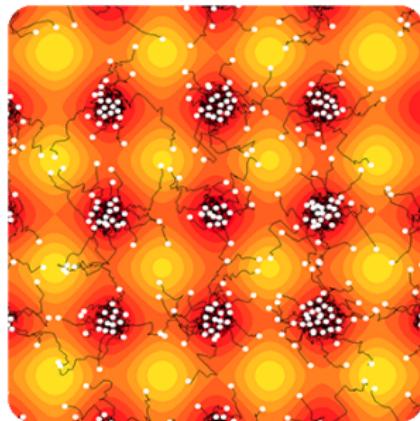
Георгий Паун



Бактериальные алгоритмы

2002

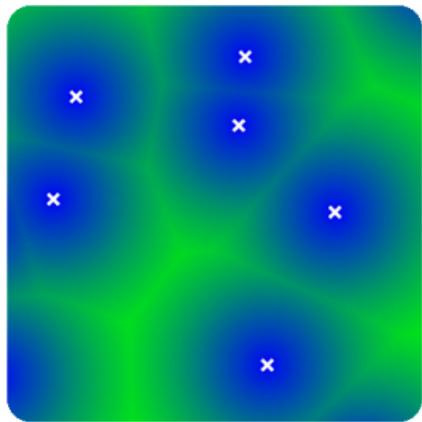
Кевин Пассино



Пчелиные алгоритмы

2005

Дук Труонг Фам



2003?

Марко Дориго?



Приложения естественных вычислений

Область применения естественных моделей в настоящее время чрезвычайно широка. Краткий, наиболее общий список приложений естественных вычислений выглядит следующим образом:

- ▶ моделирование природных процессов и систем;
- ▶ исследование вычислительных возможностей естественных систем;
- ▶ решение вычислительных задач;
- ▶ разработка новых технологий.

Приложения естественных вычислений

Область применения естественных моделей в настоящее время чрезвычайно широка. Краткий, наиболее общий список приложений естественных вычислений выглядит следующим образом:

- ▶ моделирование природных процессов и систем;
- ▶ исследование вычислительных возможностей естественных систем;
- ▶ решение вычислительных задач;
- ▶ разработка новых технологий.

Приложения естественных вычислений

Область применения естественных моделей в настоящее время чрезвычайно широка. Краткий, наиболее общий список приложений естественных вычислений выглядит следующим образом:

- ▶ моделирование природных процессов и систем;
- ▶ исследование вычислительных возможностей естественных систем;
- ▶ решение вычислительных задач;
- ▶ разработка новых технологий.

Приложения естественных вычислений

Область применения естественных моделей в настоящее время чрезвычайно широка. Краткий, наиболее общий список приложений естественных вычислений выглядит следующим образом:

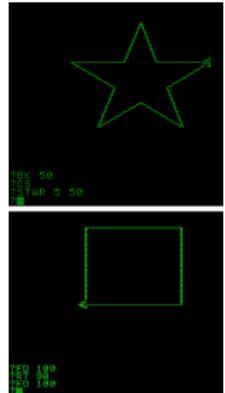
- ▶ моделирование природных процессов и систем;
- ▶ исследование вычислительных возможностей естественных систем;
- ▶ решение вычислительных задач;
- ▶ разработка новых технологий.

Структура естественных вычислительных моделей

- ▶ Практически все естественные вычислительные модели, если рассматривать их на верхнем уровне абстракции, имеют схожую структуру — каждая такая модель состоит из большого числа (обычно однотипных) относительно простых элементов (объектов). Эти объекты обладают внутренним состоянием и способны менять это состояние путем взаимодействия друг с другом или с окружающей их средой.
- ▶ Выбор подходящих правил коммуникации позволяет настроить всю систему на решение некоторой вычислительной задачи — возникает некое системное свойство, которого нет ни у одного отдельного взятого объекта этой системы.

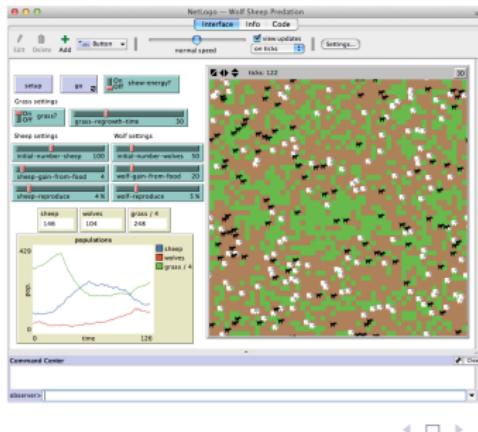
Среда NetLogo

- ▶ NetLogo — агентно-ориентированный язык программирования и интегрированная среда разработки.
- ▶ Язык программирования NetLogo является преемником языка Logo, созданного в 1960-х годах Сеймуром Папертом. Целью Паперта был интуитивно понятный язык, который бы позволил программировать не только специалистам, но и далеким от программирования людям.
- ▶ Основным элементом системы Logo была черепаха (turtle), к управлению которой и сводился процесс программирования.



Среда NetLogo

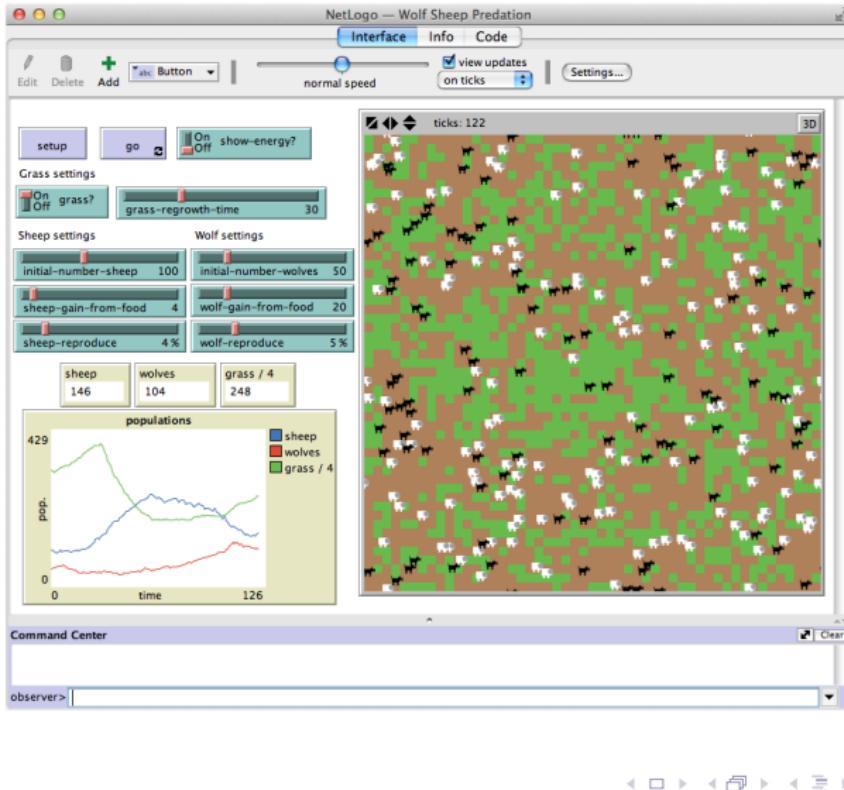
- ▶ Концепции программирования, предложенные Папертом, позже легли в основу мультиагентной системы имитационного моделирования, созданной в MIT на основе языка StarLogo.
- ▶ В 1999 году была предложена новая реализация этой системы, построенная на кроссплатформенной технологии Java, которая и получила название NetLogo.



Виртуальный мир NetLogo

- ▶ Основными элементами виртуальной модели в среде NetLogo являются агенты: пятна (patches), черепахи (turtles), связи (links) и наблюдатель (observer):
 - ▶ пятна стационарны и упорядочены в прямоугольную решетку (grid);
 - ▶ черепахи способны перемещаться по пятнам;
 - ▶ связи соединяют черепах;
 - ▶ наблюдатель наблюдает и контролирует всю систему.
- ▶ Все типы агентов способны выполнять команды NetLogo и процедуры (procedures), представляющие собой подпрограммы, записанные на языке NetLogo.

Виртуальный мир NetLogo



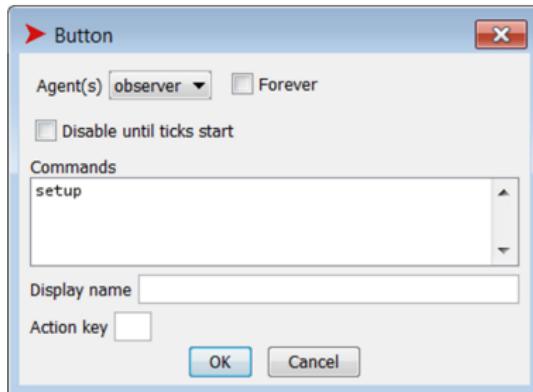
Простая популяционная модель

- ▶ Процесс моделирования в NetLogo рассмотрим на примере построения простой популяционной модели, являющейся частным случаем модели хищник–жертва.
- ▶ Чтобы начать новую модель, выбираем пункт *New* в меню *File*.
- ▶ Интерфейс модели будет состоять (на начальном этапе) из двух кнопок *Setup* и *Go*.
- ▶ Чтобы добавить кнопку к интерфейсу, кликаем кнопку *Add* при выбранном типе элемента *Button* и указываем место расположения новой кнопки.



Настройка свойств кнопки

- ▶ Появится диалог со свойствами кнопки.
- ▶ В поле *Commands* печатаем `setup` (имя процедуры, которая будет вызвана при клике на данную кнопку).



- ▶ При нажатии на новую кнопку появляется сообщение об ошибке (т.к. пока отсутствует команда `setup`).

Процедура setup

- ▶ Переходим во вкладку *Code* и печатаем там следующий текст:



```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures ▾ Indent automatically
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

- ▶ `to setup` — начинаем новую процедуру с именем `setup`.
- ▶ `clear-all` — сбрасываем мир в начальное пустое состояние.
- ▶ `create-turtles 100 [setxy random-xcor random-ycor]` — создаем 100 новых черепах и располагаем их в случайных точках нашего мира.
- ▶ `reset-ticks` — сбрасываем счетчик виртуального времени.
- ▶ `end` — завершаем процедуру `setup`.

Процедура setup

- ▶ Переходим во вкладку *Code* и печатаем там следующий текст:



```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

- ▶ `to setup` — начинаем новую процедуру с именем `setup`.
- ▶ `clear-all` — сбрасываем мир в начальное пустое состояние.
- ▶ `create-turtles 100 [setxy random-xcor random-ycor]` — создаем 100 новых черепах и располагаем их в случайных точках нашего мира.
- ▶ `reset-ticks` — сбрасываем счетчик виртуального времени.
- ▶ `end` — завершаем процедуру `setup`.

Процедура setup

- ▶ Переходим во вкладку *Code* и печатаем там следующий текст:



```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

- ▶ `to setup` — начинаем новую процедуру с именем `setup`.
- ▶ `clear-all` — сбрасываем мир в начальное пустое состояние.
- ▶ `create-turtles 100 [setxy random-xcor random-ycor]` — создаем 100 новых черепах и располагаем их в случайных точках нашего мира.
- ▶ `reset-ticks` — сбрасываем счетчик виртуального времени.
- ▶ `end` — завершаем процедуру `setup`.

Процедура setup

- ▶ Переходим во вкладку *Code* и печатаем там следующий текст:



```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

- ▶ `to setup` — начинаем новую процедуру с именем `setup`.
- ▶ `clear-all` — сбрасываем мир в начальное пустое состояние.
- ▶ `create-turtles 100 [setxy random-xcor random-ycor]` — создаем 100 новых черепах и располагаем их в случайных точках нашего мира.
- ▶ `reset-ticks` — сбрасываем счетчик виртуального времени.
- ▶ `end` — завершаем процедуру `setup`.

Процедура setup

- ▶ Переходим во вкладку *Code* и печатаем там следующий текст:



```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

- ▶ `to setup` — начинаем новую процедуру с именем `setup`.
- ▶ `clear-all` — сбрасываем мир в начальное пустое состояние.
- ▶ `create-turtles 100 [setxy random-xcor random-ycor]` — создаем 100 новых черепах и располагаем их в случайных точках нашего мира.
- ▶ `reset-ticks` — сбрасываем счетчик виртуального времени.
- ▶ `end` — завершаем процедуру `setup`.

Процедура setup

- ▶ Переходим во вкладку *Code* и печатаем там следующий текст:

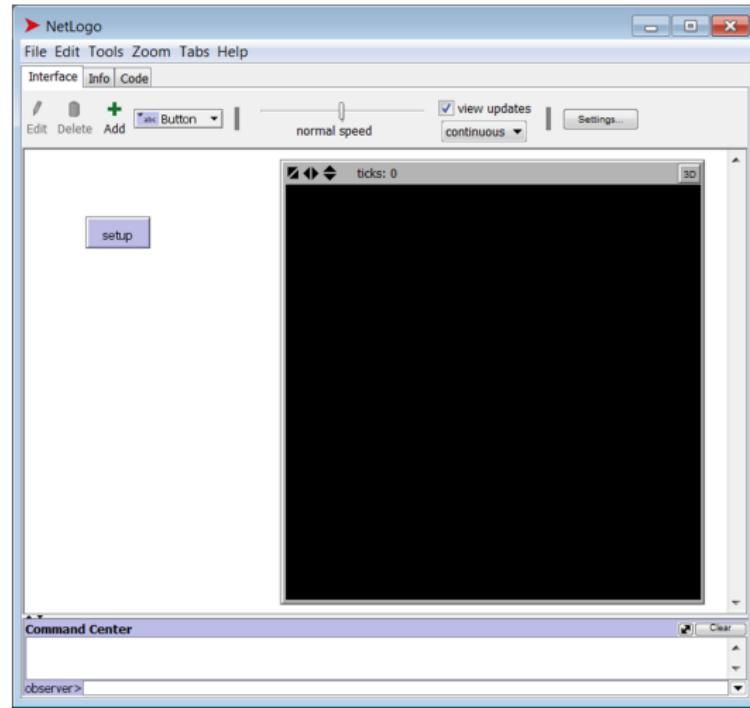


```
to setup
  clear-all
  create-turtles 100 [ setxy random-xcor random-ycor ]
  reset-ticks
end
```

- ▶ `to setup` — начинаем новую процедуру с именем `setup`.
- ▶ `clear-all` — сбрасываем мир в начальное пустое состояние.
- ▶ `create-turtles 100 [setxy random-xcor random-ycor]` — создаем 100 новых черепах и располагаем их в случайных точках нашего мира.
- ▶ `reset-ticks` — сбрасываем счетчик виртуального времени.
- ▶ `end` — завершаем процедуру `setup`.

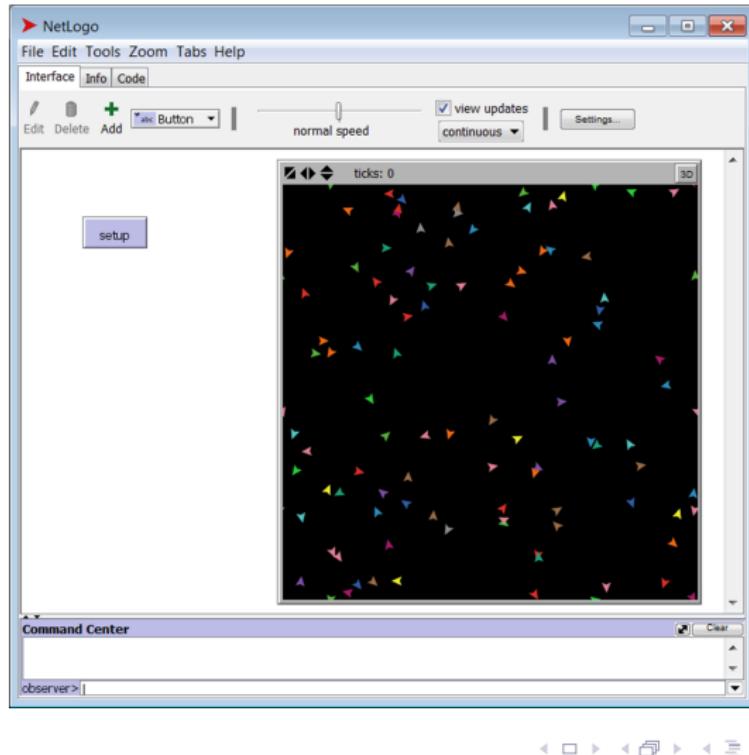
Процедура setup

- Возвращаемся во вкладку *Interface* и кликаем на *Setup*.



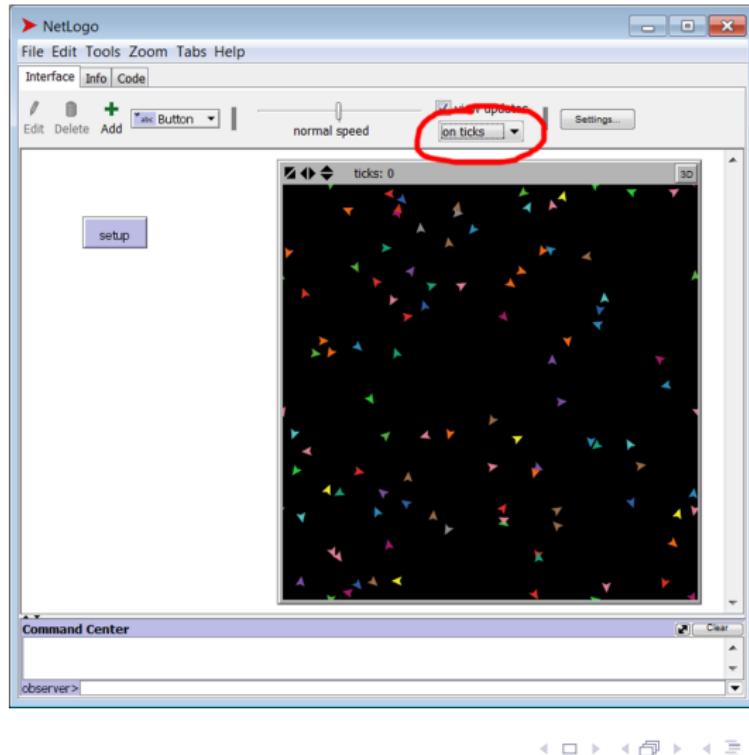
Процедура setup

- Возвращаемся во вкладку *Interface* и кликаем на *Setup*.



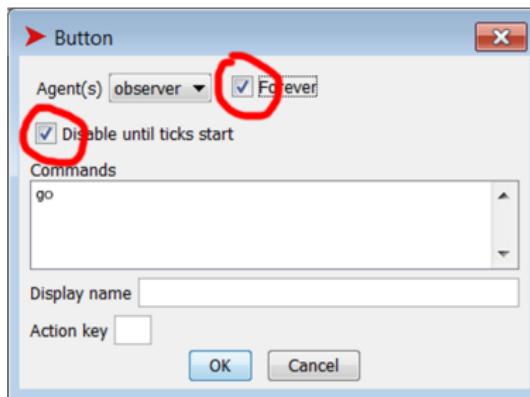
Режим просмотра

- Меняем режим continuous на on ticks



Кнопка go

- ▶ Делаем кнопку go аналогии с кнопкой setup.
- ▶ Настраиваем ее немного по-другому.



- ▶ Forever — заставляет код, связанный с кнопкой, выполняться автоматически после каждого тика.
- ▶ Disable — предотвращает нажатие кнопки, пока не выполнена инициализация мира (не нажата кнопка setup).

Процедура go

- ▶ Во вкладке *Code* описываем процедуру go:

```
to go
  move-turtles
  tick
end
```

- ▶ to go — начинаем новую процедуру с именем go.
- ▶ move-turtles — вызов процедуры, которая пока еще нами не определена.
- ▶ tick — увеличивает счетчик времени на 1.
- ▶ end — завершаем процедуру go.

Процедура go

- ▶ Во вкладке *Code* описываем процедуру go:

```
to go
  move-turtles
  tick
end
```

- ▶ to go — начинаем новую процедуру с именем go.
- ▶ move-turtles — вызов процедуры, которая пока еще нами не определена.
- ▶ tick — увеличивает счетчик времени на 1.
- ▶ end — завершаем процедуру go.

Процедура go

- ▶ Во вкладке *Code* описываем процедуру go:

```
to go
  move-turtles
  tick
end
```

- ▶ to go — начинаем новую процедуру с именем go.
- ▶ move-turtles — вызов процедуры, которая пока еще нами не определена.
- ▶ tick — увеличивает счетчик времени на 1.
- ▶ end — завершаем процедуру go.

Процедура go

- ▶ Во вкладке *Code* описываем процедуру go:

```
to go
  move-turtles
  tick
end
```

- ▶ to go — начинаем новую процедуру с именем go.
- ▶ move-turtles — вызов процедуры, которая пока еще нами не определена.
- ▶ tick — увеличивает счетчик времени на 1.
- ▶ end — завершаем процедуру go.

Процедура go

- ▶ Во вкладке *Code* описываем процедуру go:

```
to go
  move-turtles
  tick
end
```

- ▶ to go — начинаем новую процедуру с именем go.
- ▶ move-turtles — вызов процедуры, которая пока еще нами не определена.
- ▶ tick — увеличивает счетчик времени на 1.
- ▶ end — завершаем процедуру go.

Процедура move-turtles

- ▶ Определяем процедуру move-turtles:

```
to move-turtles
  ask turtles [
    right 20 - random 40
    forward 0.2
  ]
end
```

- ▶ to move-turtles — начинаем новую процедуру с именем move-turtles.
- ▶ ask turtles [...] — просим каждую черепаху выполнить действия, указанные в скобках [...] .
- ▶ right a — поворачиваем черепаху на угол a.
- ▶ 20 - random 40 — случайный угол от -20 до 20 градусов.
- ▶ forward 0.2 — перемещаем черепаху на расстояние 0.2 вперед (относительно ее текущей ориентации).

Процедура move-turtles

- ▶ Определяем процедуру move-turtles:

```
to move-turtles
  ask turtles [
    right 20 - random 40
    forward 0.2
  ]
end
```

- ▶ to move-turtles — начинаем новую процедуру с именем move-turtles.
- ▶ ask turtles [...] — просим каждую черепаху выполнить действия, указанные в скобках [...] .
- ▶ right a — поворачиваем черепаху на угол a.
- ▶ 20 - random 40 — случайный угол от -20 до 20 градусов.
- ▶ forward 0.2 — перемещаем черепаху на расстояние 0.2 вперед (относительно ее текущей ориентации).

Процедура move-turtles

- ▶ Определяем процедуру move-turtles:

```
to move-turtles
  ask turtles [
    right 20 - random 40
    forward 0.2
  ]
end
```

- ▶ to move-turtles — начинаем новую процедуру с именем move-turtles.
- ▶ ask turtles [...] — просим каждую черепаху выполнить действия, указанные в скобках [...] .
- ▶ right a — поворачиваем черепаху на угол a.
- ▶ 20 - random 40 — случайный угол от -20 до 20 градусов.
- ▶ forward 0.2 — перемещаем черепаху на расстояние 0.2 вперед (относительно ее текущей ориентации).

Процедура move-turtles

- ▶ Определяем процедуру move-turtles:

```
to move-turtles
  ask turtles [
    right 20 - random 40
    forward 0.2
  ]
end
```

- ▶ to move-turtles — начинаем новую процедуру с именем move-turtles.
- ▶ ask turtles [...] — просим каждую черепаху выполнить действия, указанные в скобках [...] .
- ▶ right a — поворачиваем черепаху на угол a.
- ▶ 20 - random 40 — случайный угол от -20 до 20 градусов.
- ▶ forward 0.2 — перемещаем черепаху на расстояние 0.2 вперед (относительно ее текущей ориентации).

Процедура move-turtles

- ▶ Определяем процедуру move-turtles:

```
to move-turtles
  ask turtles [
    right 20 - random 40
    forward 0.2
  ]
end
```

- ▶ to move-turtles — начинаем новую процедуру с именем move-turtles.
- ▶ ask turtles [...] — просим каждую черепаху выполнить действия, указанные в скобках [...] .
- ▶ right a — поворачиваем черепаху на угол a.
- ▶ 20 - random 40 — случайный угол от -20 до 20 градусов.
- ▶ forward 0.2 — перемещаем черепаху на расстояние 0.2 вперед (относительно ее текущей ориентации).

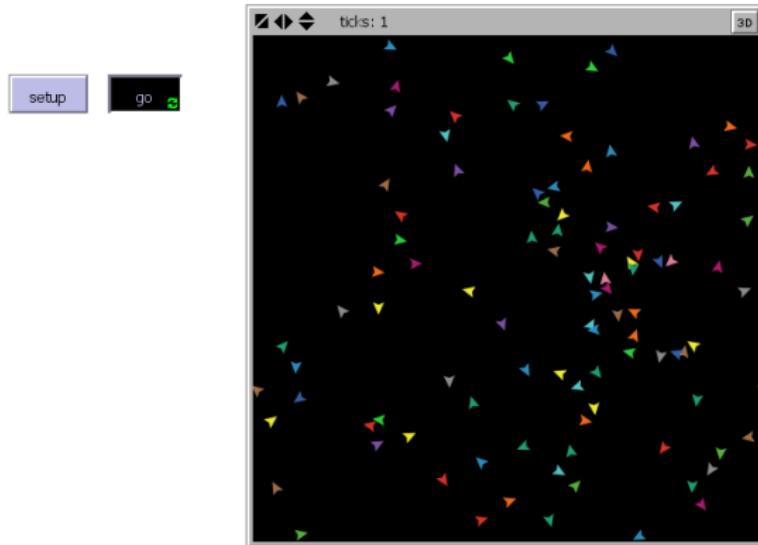
Процедура move-turtles

- ▶ Определяем процедуру move-turtles:

```
to move-turtles
  ask turtles [
    right 20 - random 40
    forward 0.2
  ]
end
```

- ▶ to move-turtles — начинаем новую процедуру с именем move-turtles.
- ▶ ask turtles [...] — просим каждую черепаху выполнить действия, указанные в скобках [...] .
- ▶ right a — поворачиваем черепаху на угол a.
- ▶ 20 - random 40 — случайный угол от -20 до 20 градусов.
- ▶ forward 0.2 — перемещаем черепаху на расстояние 0.2 вперед (относительно ее текущей ориентации).

Запускаем модель



Запускаем модель

Усложняем модель

- Разделим процедуры setup на две отдельные процедуры setup-turtles и setup-patches.

```
to setup
  clear-all
  setup-turtles
  setup-patches
  reset-ticks
end

to setup-turtles
  create-turtles 100
  ask turtles [ setxy random-xcor random-ycor ]
end

to setup-patches
  ask patches [ set pcolor green ]
end
```

- ask patches [...] — просим каждое пятно выполнить действия, указанные в скобках [...] .
- set pcolor green — устанавливаем зеленый цвет текущего пятна.

Усложняем модель

- ▶ Разделим процедуры setup на две отдельные процедуры setup-turtles и setup-patches.

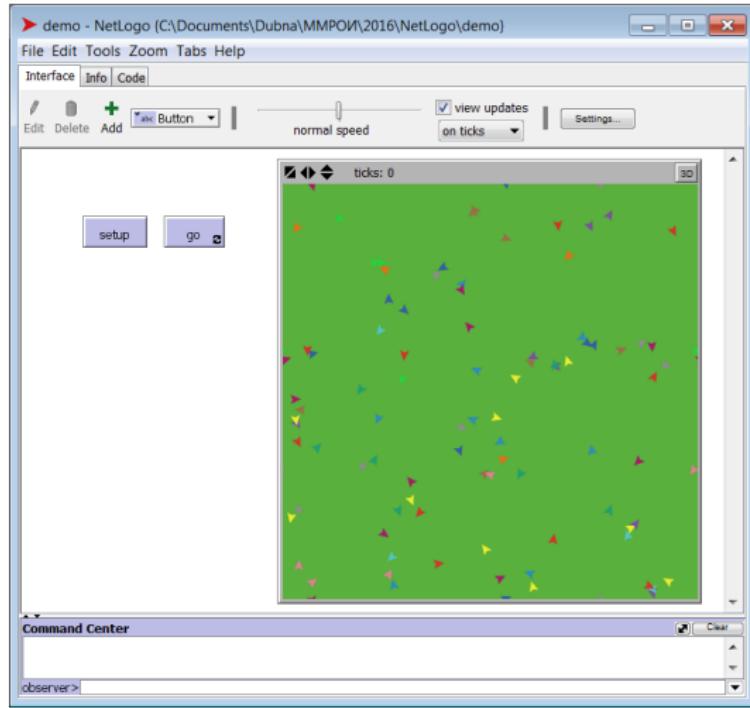
```
to setup
  clear-all
  setup-turtles
  setup-patches
  reset-ticks
end

to setup-turtles
  create-turtles 100
  ask turtles [ setxy random-xcor random-ycor ]
end

to setup-patches
  ask patches [ set pcolor green ]
end
```

- ▶ ask patches [...] — просим каждое пятно выполнить действия, указанные в скобках [...] .
- ▶ set pcolor green — устанавливаем зеленый цвет текущего пятна.

Усложняем модель



Усложняем модель

- ▶ Добавим переменную energy каждой черепахе:

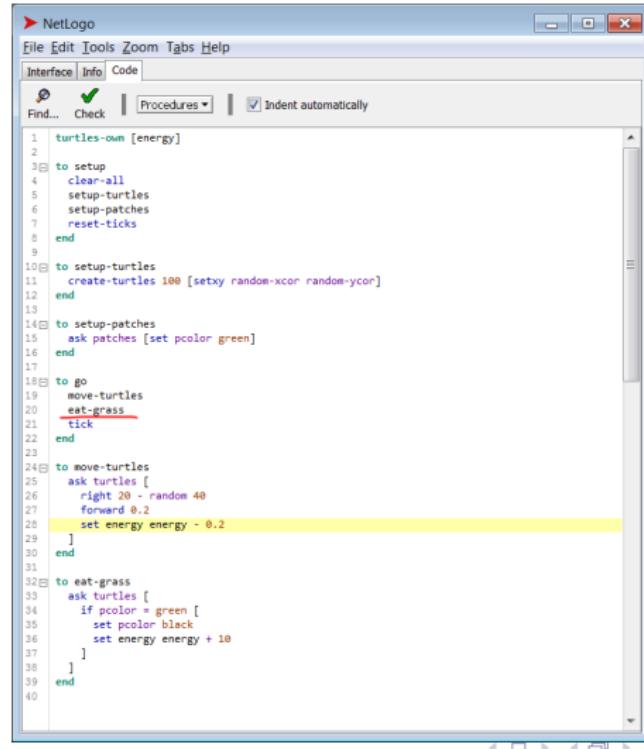
The screenshot shows the NetLogo code editor window. The code is as follows:

```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures ▾ Indent automatically
1 turtles-own [energy]
2
3 to setup
4   clear-all
5   setup-turtles
6   setup-patches
7   reset-ticks
8 end
9
10 to setup-turtles
11   create-turtles 100 [setxy random-xcor random-ycor]
12 end
13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   tick
22 end
23
24 to move-turtles
25   ask turtles [
26     right 20 - random 40
27     forward 0.2
28     set energy energy - 0.2
29   ]
30 end
31
32 to eat-grass
33   ask turtles [
34     if pcolor = green [
35       set pcolor black
36       set energy energy + 10
37     ]
38   ]
39 end
40
```

The line `turtles-own [energy]` is underlined in red, indicating it is being defined. The line `set energy energy - 0.2` is highlighted in yellow, indicating it is the current selection.

Усложняем модель

- ▶ Заставим черепах есть траву...



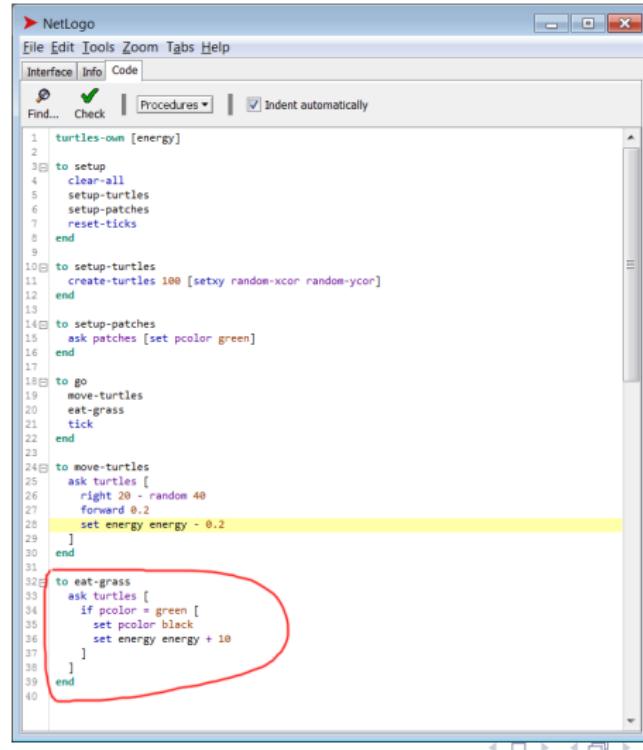
The screenshot shows the NetLogo interface with the code editor open. The title bar says "NetLogo". The menu bar includes "File", "Edit", "Tools", "Zoom", "Tabs", and "Help". Below the menu is a toolbar with "Interface", "Info", and "Code" tabs, with "Code" selected. There are "Find...", "Check", and "Procedures" buttons, and a checkbox for "Indent automatically". The code area contains the following NetLogo script:

```
1 turtles-own [energy]
2
3 to setup
4   clear-all
5   setup-turtles
6   setup-patches
7   reset-ticks
8 end
9
10 to setup-turtles
11   create-turtles 100 [setxy random-xcor random-ycor]
12 end
13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   tick
22 end
23
24 to move-turtles
25   ask turtles [
26     right 20 - random 40
27     forward 0.2
28     set energy energy - 0.2
29   ]
30 end
31
32 to eat-grass
33   ask turtles [
34     if pcolor = green [
35       set pcolor black
36       set energy energy + 10
37     ]
38   ]
39 end
```

The line "eat-grass" is underlined in red, indicating it is currently selected or being edited. Line 28, which decreases energy by 0.2, is highlighted with a yellow background.

Усложняем модель

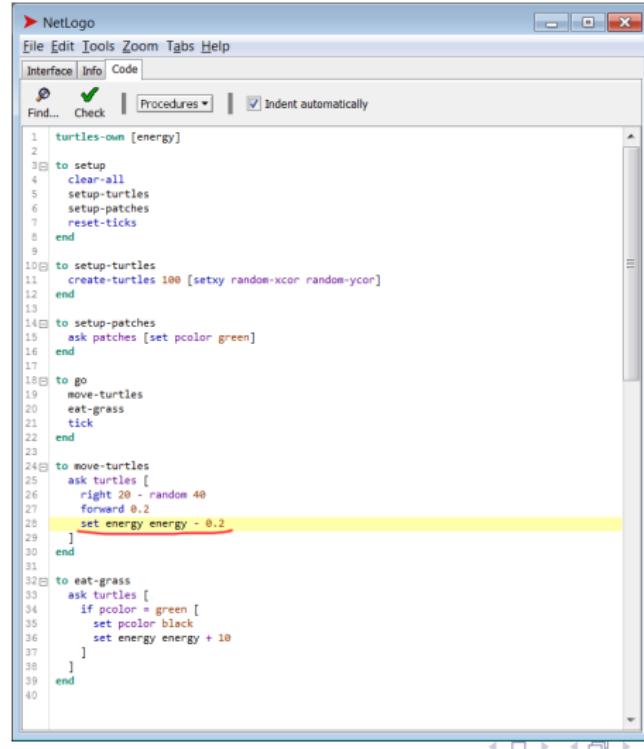
- ▶ Повышая свой уровень энергии...



```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures ▾ Indent automatically
1 turtles-own [energy]
2
3 to setup
4 clear-all
5 setup-turtles
6 setup-patches
7 reset-ticks
8 end
9
10 to setup-turtles
11 create-turtles 100 [setxy random-xcor random-ycor]
12 end
13
14 to setup-patches
15 ask patches [set pcolor green]
16 end
17
18 to go
19 move-turtles
20 eat-grass
21 tick
22 end
23
24 to move-turtles
25 ask turtles [
26 right 20 - random 40
27 forward 0.2
28 set energy energy - 0.2
29 ]
30 end
31
32 to eat-grass
33 ask turtles [
34 if pcolor = green [
35 set pcolor black
36 set energy energy + 10
37 ]
38 ]
39 end
40
```

Усложняем модель

- И тратить энергию при движении:

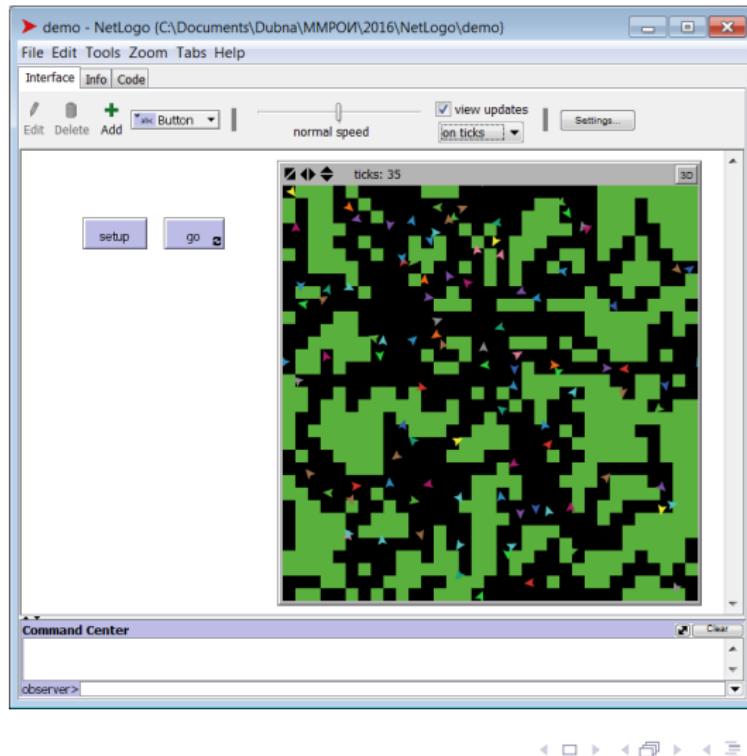


The screenshot shows the NetLogo interface with the code editor open. The code is written in NetLogo's built-in programming language. A specific line of code, which decreases the turtle's energy by 0.2, is highlighted with a yellow background.

```
1 turtles-own [energy]
2
3 to setup
4   clear-all
5   setup-turtles
6   setup-patches
7   reset-ticks
8 end
9
10 to setup-turtles
11   create-turtles 100 [setxy random-xcor random-ycor]
12 end
13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   tick
22 end
23
24 to move-turtles
25   ask turtles [
26     right 20 - random 40
27     forward 0.2
28     set energy energy - 0.2
29   ]
30 end
31
32 to eat-grass
33   ask turtles [
34     if pcolor = green [
35       set pcolor black
36       set energy energy + 10
37     ]
38   ]
39 end
40
```

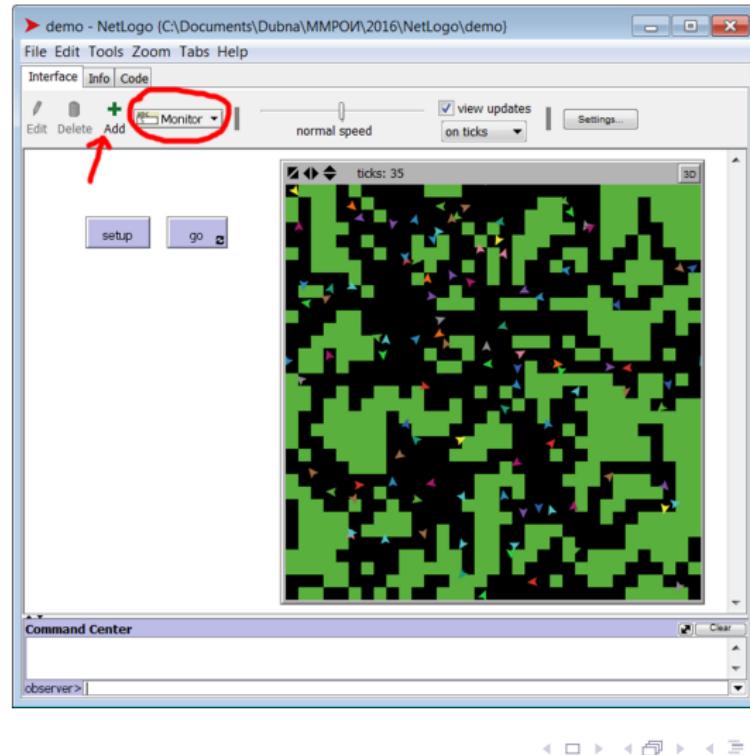
Усложняем модель

- ▶ Трава исчезает, энергия черепах растет...



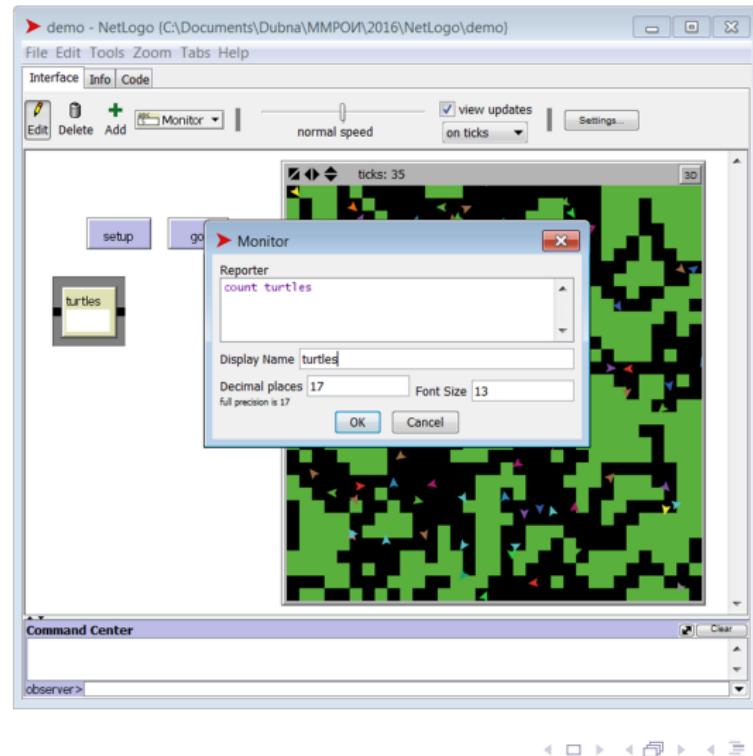
Мониторы

► Создадим два монитора:



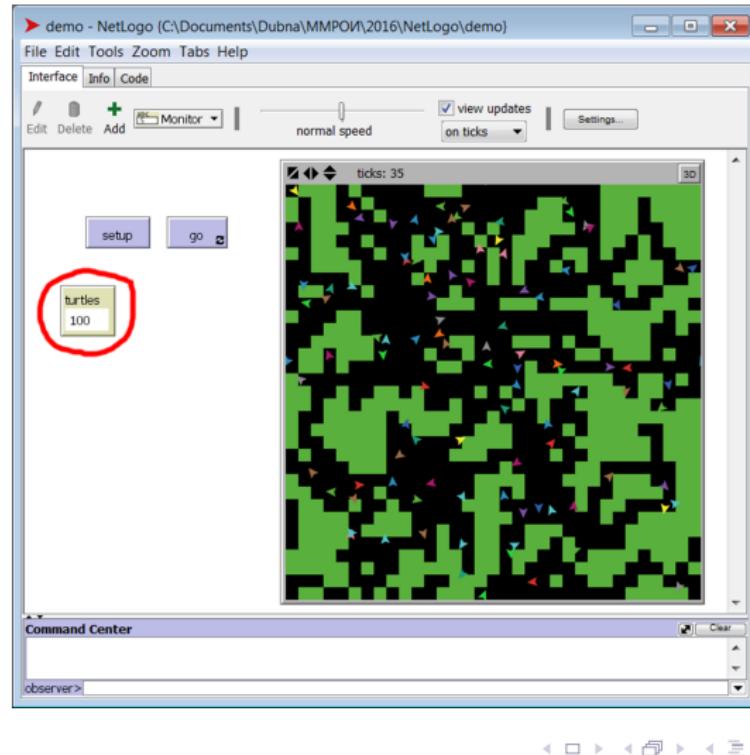
Мониторы

- ▶ Устанавливаем величину для мониторинга:



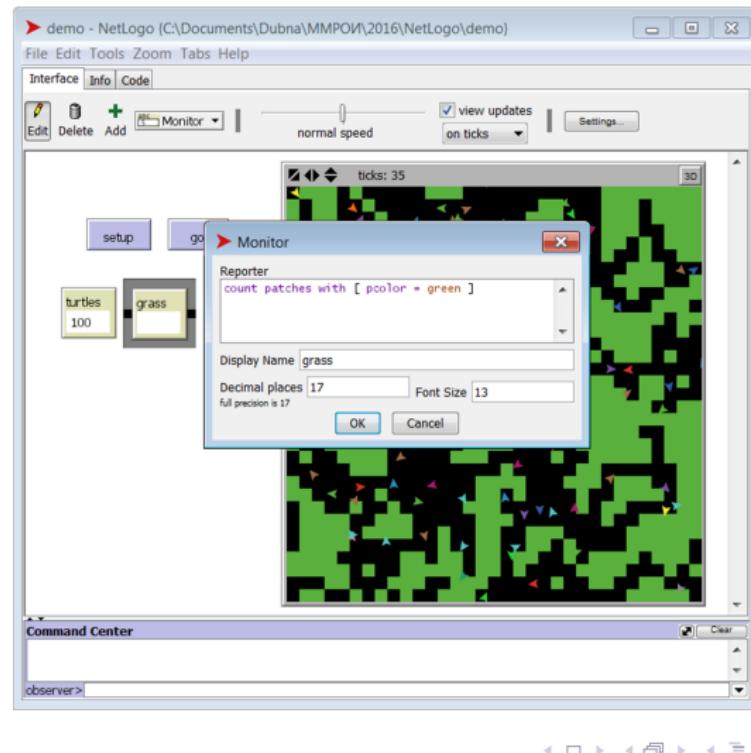
Мониторы

- ▶ Первый монитор показывает число черепах:



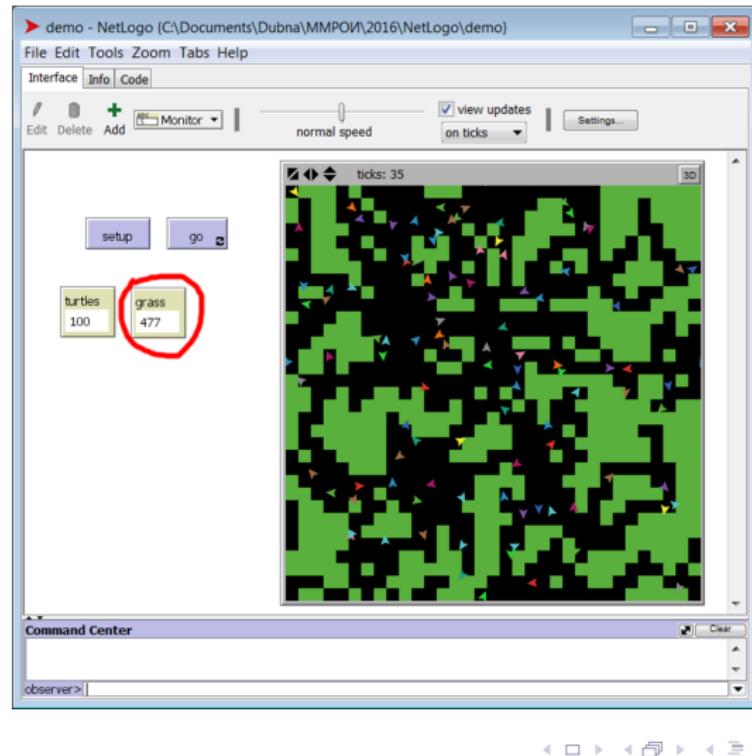
Мониторы

- ▶ Во втором мониторе считаем число зеленых пятен:



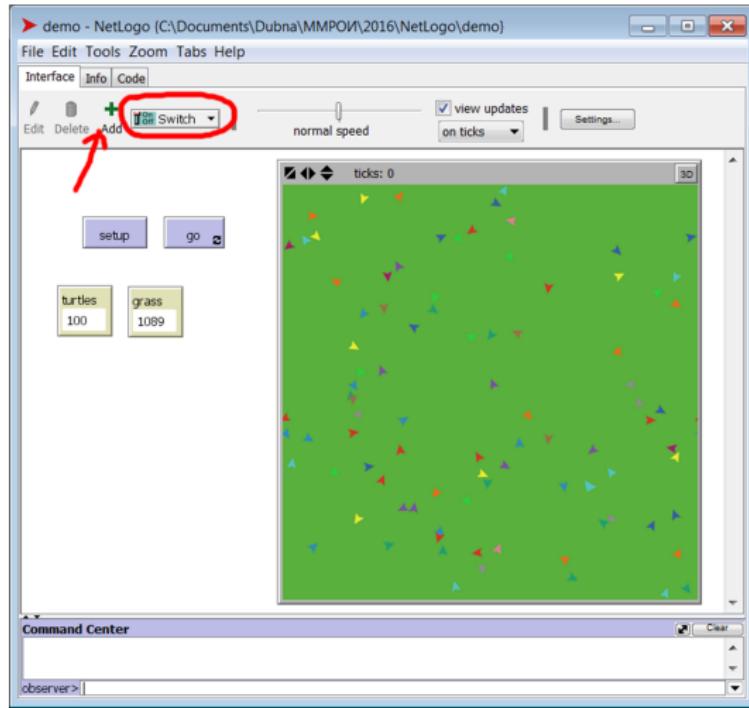
Мониторы

► Готово:



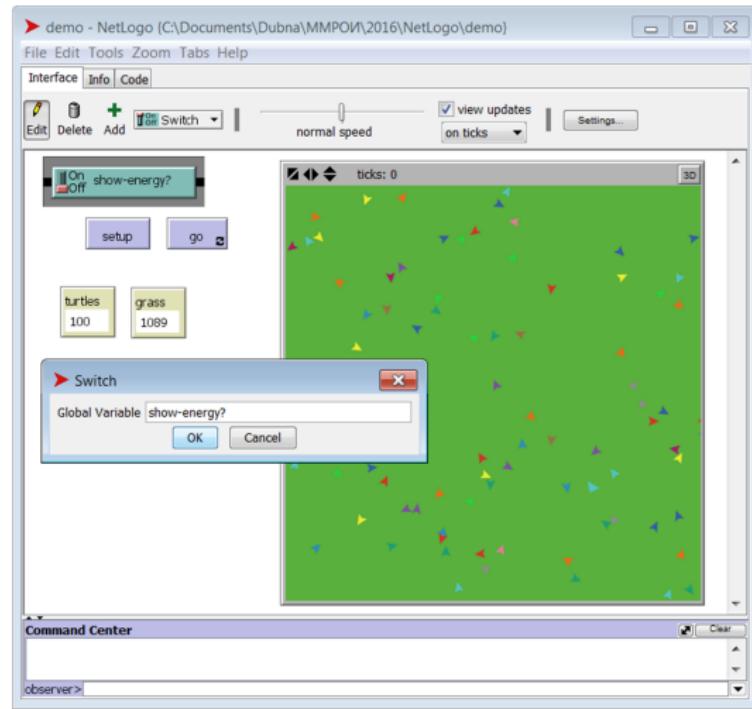
Проверяем уровень энергии

► Добавляем переключатель



Проверяем уровень энергии

- ▶ Связываем его с глобальной переменной show-energy?



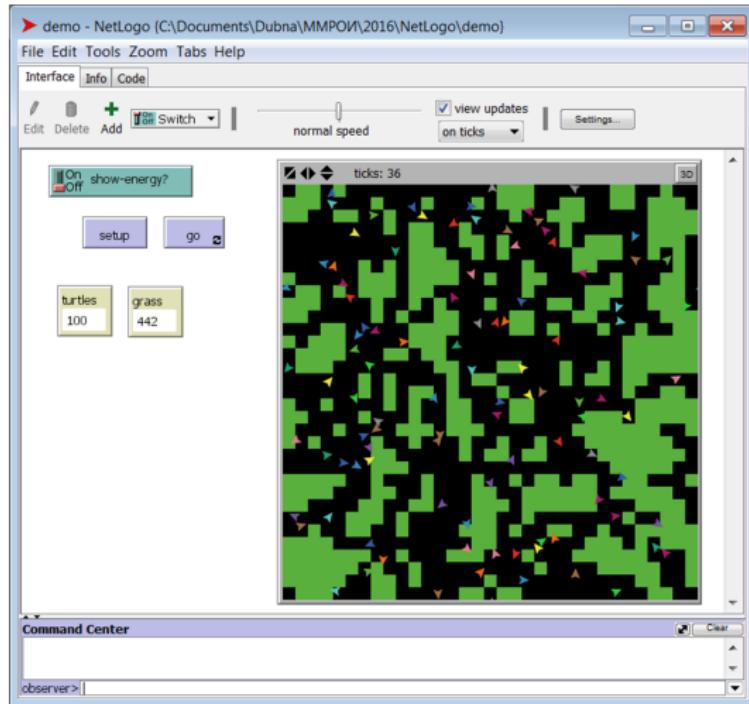
Проверяем уровень энергии

- ▶ Добавляем проверку этой переменной в процедуре eat-grass

```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures ▾ Indent automatically
1 turtles-own [energy]
2
3 to setup
4   clear-all
5   setup-turtles
6   setup-patches
7   reset-ticks
8 end
9
10 to setup-turtles
11   create-turtles 100 [setxy random-xcor random-ycor]
12 end
13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   tick
22 end
23
24 to move-turtles
25   ask turtles [
26     right 20 + random 40
27     forward 0.2
28     set energy energy - 0.2
29   ]
30 end
31
32 to eat-grass
33   ask turtles [
34     if pcolor = green [
35       set pcolor black
36       set energy energy + 10
37     ]
38     ifelse show-energy? [set label round energy] [set label ""]
39   ]
40 end
41
```

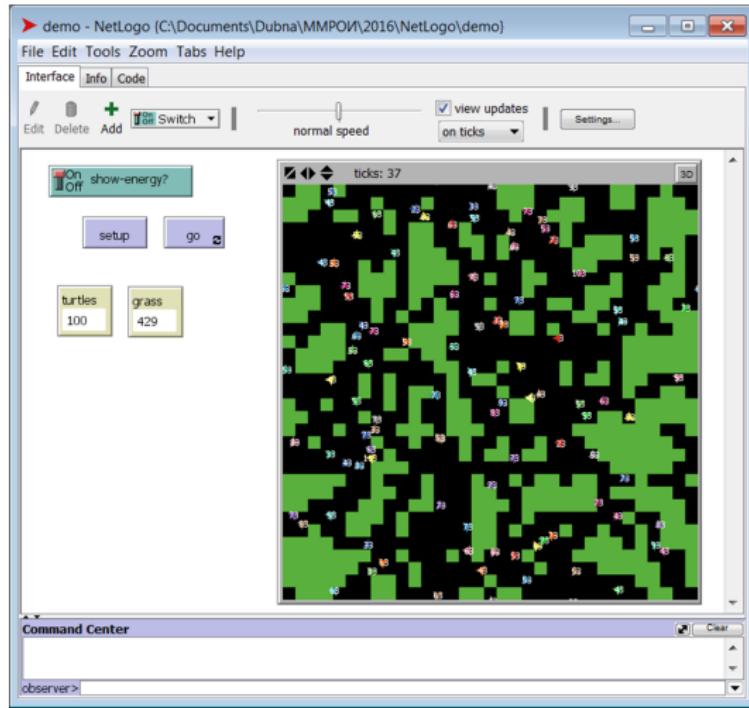
Проверяем уровень энергии

► Проверяем



Проверяем уровень энергии

► Работает!



Усложняем поведение

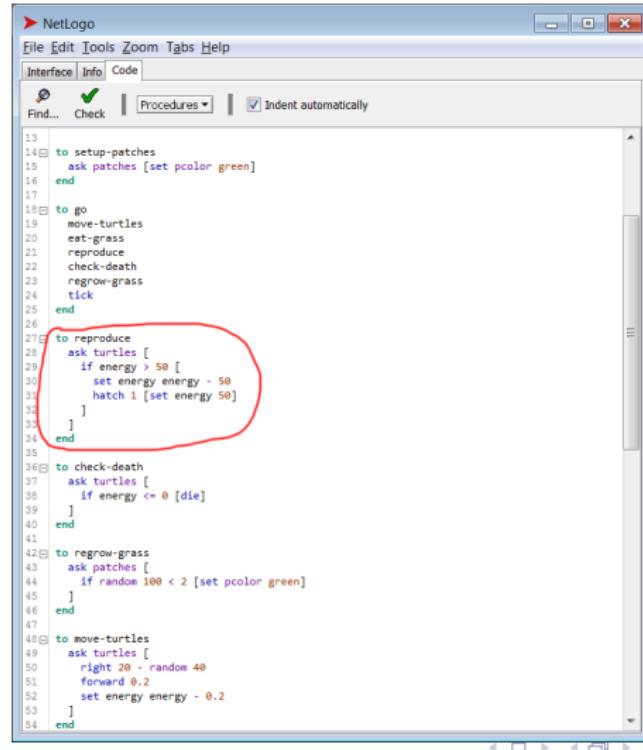
► Добавляем три новых процедуры

The screenshot shows the NetLogo interface with the code editor open. The code is written in NetLogo's built-in programming language. Three specific lines of code are circled in red: 'eat-grass', 'reproduce', and 'regrow-grass'. These are likely the new procedures being added to the model.

```
13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   reproduce
22   check-death
23   regrow-grass
24   tick
25 end
26
27 to reproduce
28   ask turtles [
29     if energy > 50 [
30       set energy energy - 50
31       hatch 1 [set energy 50]
32     ]
33   ]
34 end
35
36 to check-death
37   ask turtles [
38     if energy <= 0 [die]
39   ]
40 end
41
42 to regrow-grass
43   ask patches [
44     if random 100 < 2 [set pcolor green]
45   ]
46 end
47
48 to move-turtles
49   ask turtles [
50     right 20 + random 40
51     forward 0.2
52     set energy energy - 0.2
53   ]
54 end
```

Усложняем поведение

- ▶ Если энергии много, черепаха делится на две

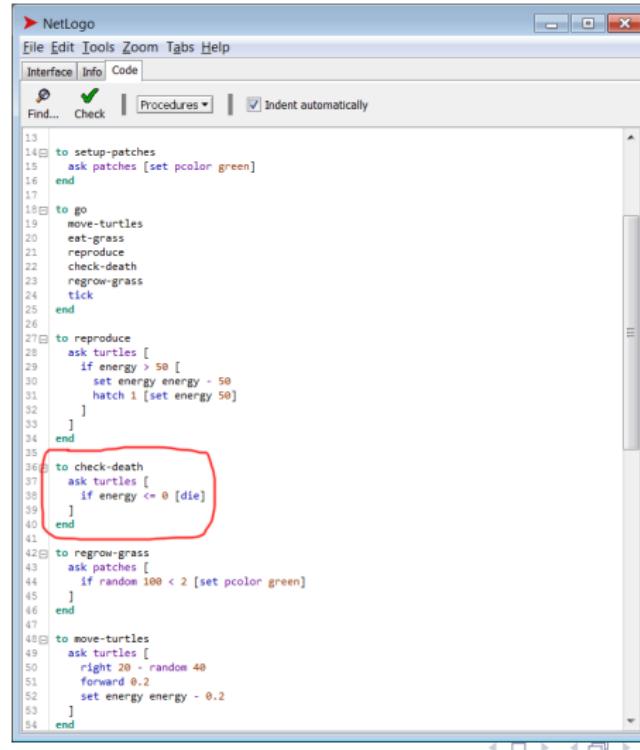


```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures ▾ Indent automatically

13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   reproduce
22   check-death
23   regrow-grass
24   tick
25 end
26
27 to reproduce
28   ask turtles [
29     if energy > 50 [
30       set energy energy - 50
31       hatch 1 [set energy 50]
32     ]
33   ]
34 end
35
36 to check-death
37   ask turtles [
38     if energy <= 0 [die]
39   ]
40 end
41
42 to regrow-grass
43   ask patches [
44     if random 100 < 2 [set pcolor green]
45   ]
46 end
47
48 to move-turtles
49   ask turtles [
50     right 20 + random 40
51     forward 0.2
52     set energy energy - 0.2
53   ]
54 end
```

Усложняем поведение

- ▶ Если энергии мало, она умирает

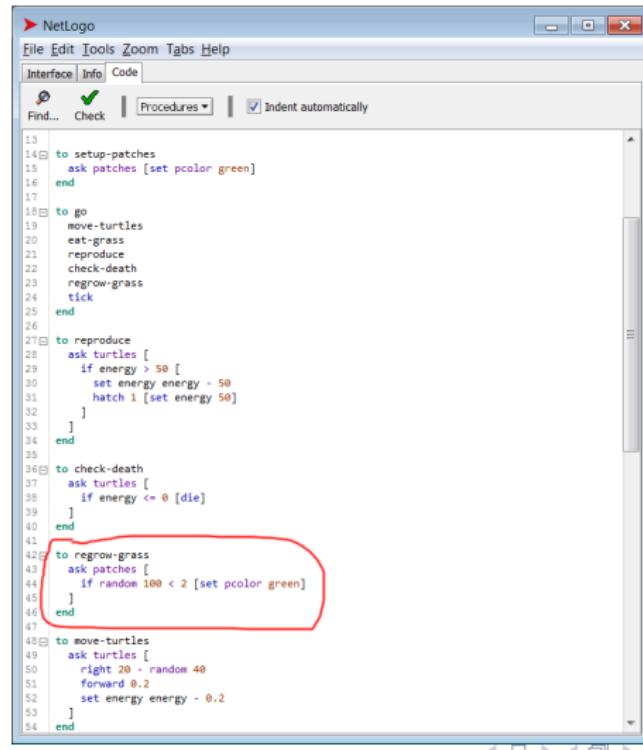


```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures Indent automatically

13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   reproduce
22   check-death
23   regrow-grass
24   tick
25 end
26
27 to reproduce
28   ask turtles [
29     if energy > 50 [
30       set energy energy - 50
31       hatch 1 [set energy 50]
32     ]
33   ]
34 end
35
36 to check-death
37   ask turtles [
38     if energy <= 0 [die]
39   ]
40 end
41
42 to regrow-grass
43   ask patches [
44     if random 100 < 2 [set pcolor green]
45   ]
46 end
47
48 to move-turtles
49   ask turtles [
50     right 20 + random 40
51     forward 0.2
52     set energy energy - 0.2
53   ]
54 end
```

Усложняем поведение

- ▶ Трава отрастает с вероятностью 0.02

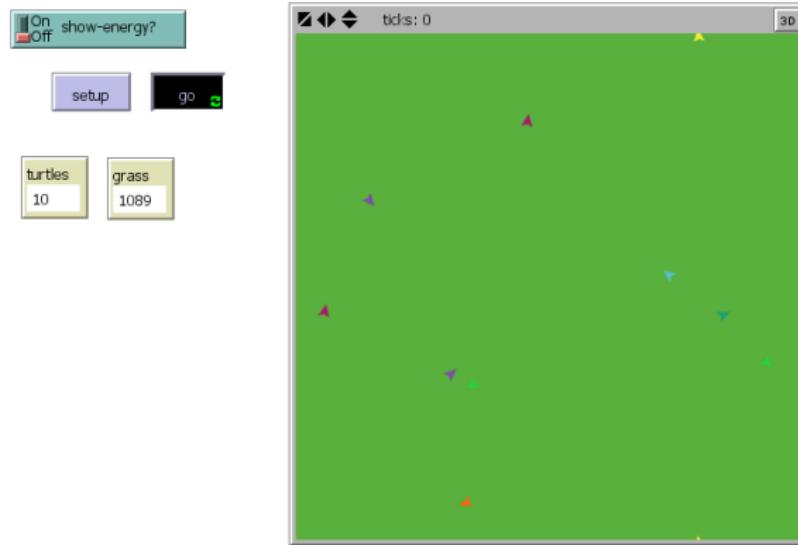


```
NetLogo
File Edit Tools Zoom Tabs Help
Interface Info Code
Find... Check Procedures Indent automatically

13
14 to setup-patches
15   ask patches [set pcolor green]
16 end
17
18 to go
19   move-turtles
20   eat-grass
21   reproduce
22   check-death
23   regrow-grass
24   tick
25 end
26
27 to reproduce
28   ask turtles [
29     if energy > 50 [
30       set energy energy - 50
31       hatch 1 [set energy 50]
32     ]
33   ]
34 end
35
36 to check-death
37   ask turtles [
38     if energy <= 0 [die]
39   ]
40 end
41
42 to regrow-grass
43   ask patches [
44     if random 100 < 2 [set pcolor green]
45   ]
46 end
47
48 to move-turtles
49   ask turtles [
50     right 20 + random 40
51     forward 0.2
52     set energy energy - 0.2
53   ]
54 end
```

Усложняем поведение

- ▶ Запускаем (начальный размер популяции — 10 черепах)

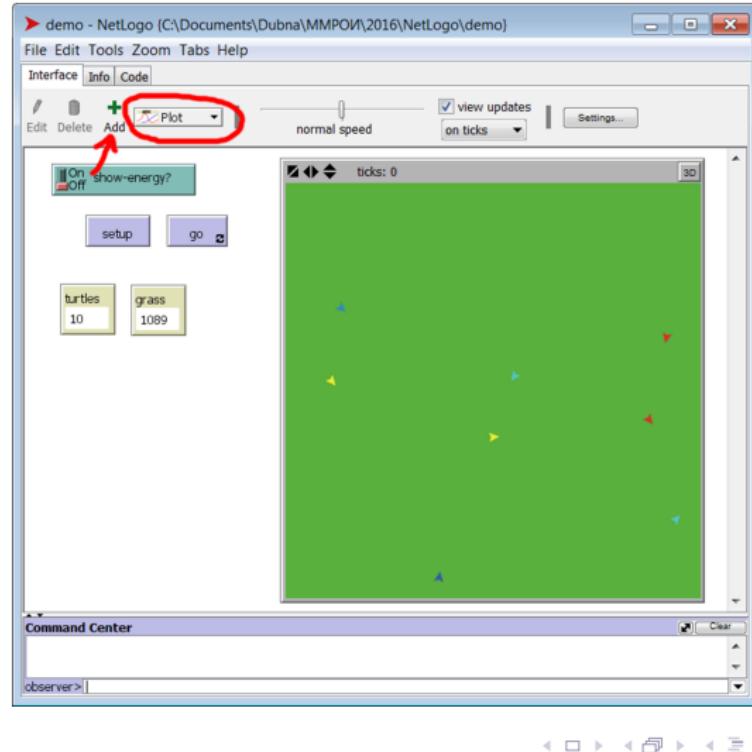


Усложняем поведение

- ▶ Запускаем (начальный размер популяции — 10 черепах)

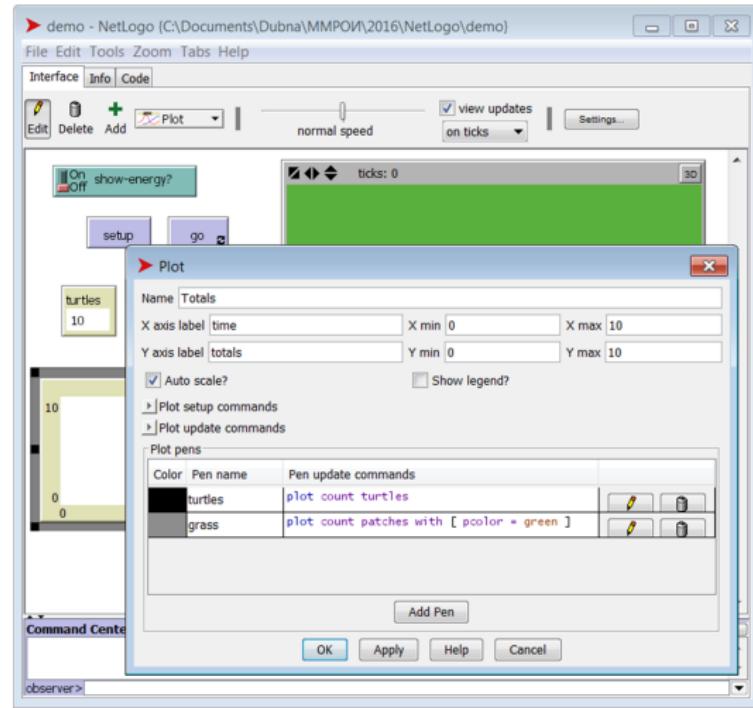
Добавляем графики

► Добавляем график (plot)



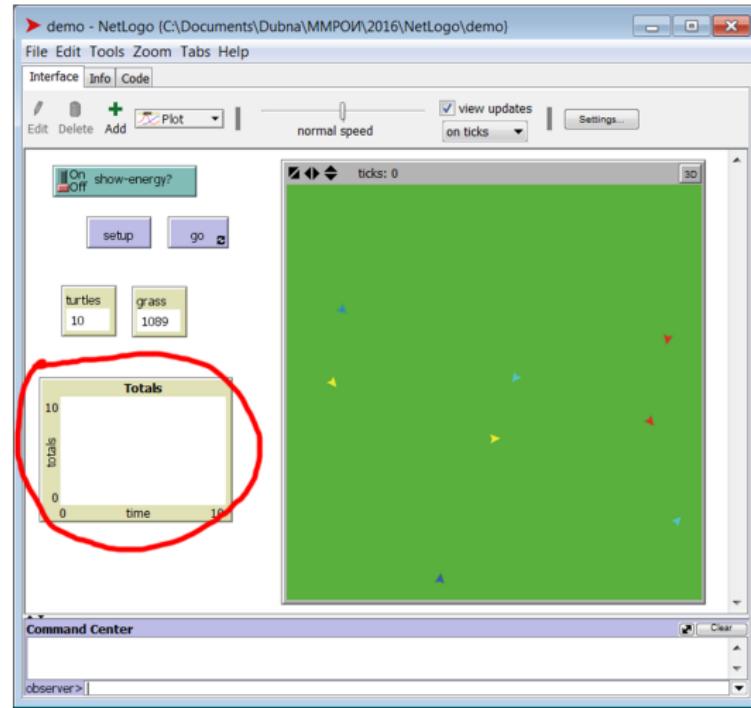
Добавляем графики

► Настраиваем параметры



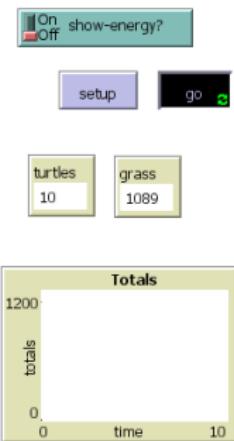
Добавляем графики

► График готов



Добавляем графики

► Проверяем!



Добавляем графики

- ▶ Проверяем!

Управление параметрами

- ▶ Используя элементы интерфейса (slider, input и т.д.) добавьте возможность управления параметрами модели (чтобы не редактировать при каждом изменении параметра код):
 - ▶ Начальное число черепах turtles-number;
 - ▶ На сколько вырастает энергия при поедании травы grass-energy;
 - ▶ На сколько падает энергия при движении move-energy;
 - ▶ Скорость черепах turtles-velocity;
 - ▶ Максимальный угол поворота max-angle;
 - ▶ Вероятность роста травы grass-probability;
 - ▶ Уровень энергии для размножения birth-energy
 - ▶ и т.д...

Библиография

1. <https://ccl.northwestern.edu/netlogo> — NetLogo Home Page
2. Ершов Н.М., *Введение в распределенное моделирование в среде NetLogo*, М.: ДМК Пресс, 2018.
3. Ершов Н.М., Попова Н.Н., *Естественные модели параллельных вычислений*, М.: МАКС Пресс, 2016 (<https://www.twirpx.org/file/2547466/>).

Спасибо за внимание!