Simulações Baseadas em Agentes: Aprendizado da ferramenta NetLogo

Mateus Rissardi
Engenheiro de Software / UDESC Alto Vale
mrissardi01@gmail.com

Fernando Santos
Professor / UDESC Alto Vale
fernando.santos@udesc.br

ESO
Engenharia de Software

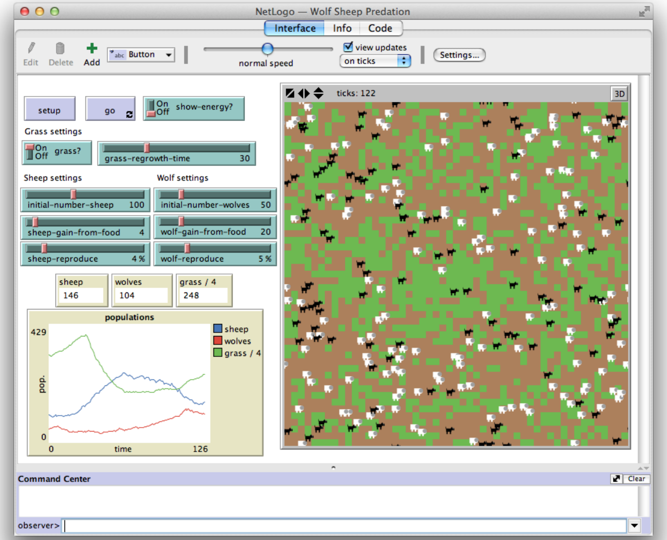
UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA

ERAMIA-RS
I ESCOLA REGIONAL DE APRENDIZADO DE MÁQUINA E INTELIGÊNCIA ARTIFICIAL DA REGIÃO SUL
2025

Introdução do NetLogo

O que é o NetLogo?:

- NetLogo é um software de simulação baseado em agentes.
- Opera centenas ou milhares de agentes de forma independente.
- *Low Threshold, High Ceiling.*



show-energy? Off
grass? On
grass-regrowth-time 30
initial-number-sheep 100
sheep-gain-from-food 4
sheep-reproduce 4 %
initial-number-wolves 50
wolf-gain-from-food 20
wolf-reproduce 5 %
sheep 146
wolves 104
grass / 4 248

populations: pop. 0–429; time 0–126; 🟦 sheep, 🟥 wolves, 🟩 grass / 4

NetLogo 7.0.2: What is NetLogo?

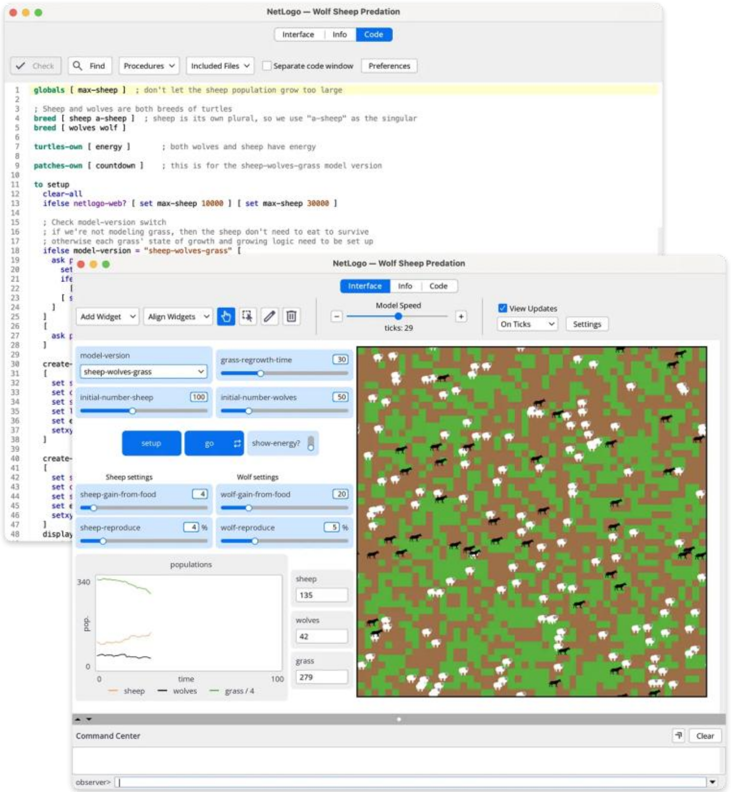
Fundamentos do NetLogo

- Turtles
  - Habitam o mundo
- Patches
  - Representam o “chão”
- *Links*
  - *Estabelecem conexões*
- *Observer*
  - *Observa e comanda a simulação*

```
to learn-netlogo
  ask patches [be-the-world]
  ask turtles [do-things-in-the-world]
  ask links   [connect-turtles]
end
```

[NetLogo 7.0.2: Programming Guide]

Interface do NetLogo



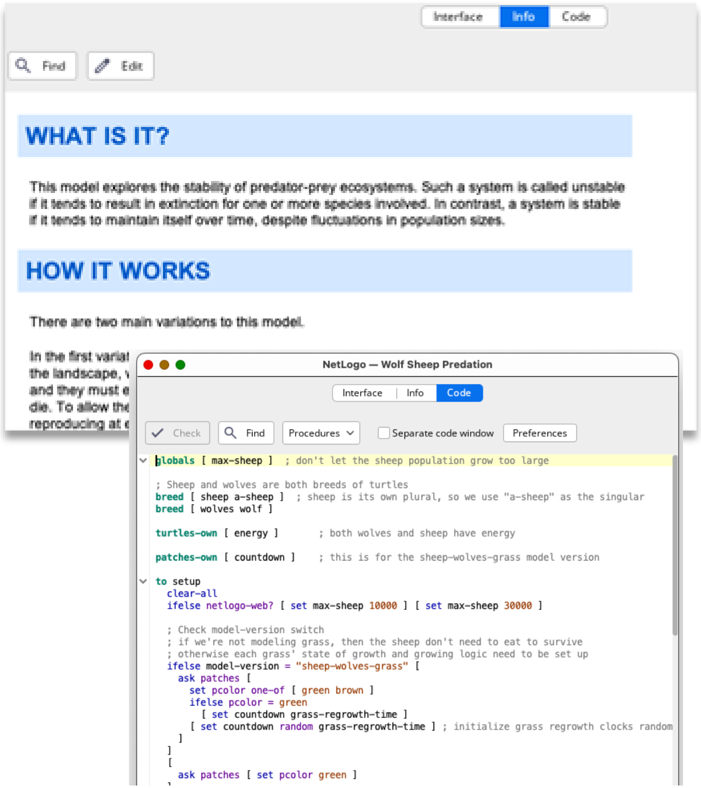
```
globals [ max-sheep ]  ; don't let the sheep population grow too large

; Sheep and wolves are both breeds of turtles
breed [ sheep a-sheep ]  ; sheep is its own plural, so we use "a-sheep" as the singular
breed [ wolves wolf ]

turtles-own [ energy ]       ; both wolves and sheep have energy

patches-own [ countdown ]    ; this is for the sheep-wolves-grass model version

to setup
  clear-all
  ifelse netlogo-web? [ set max-sheep 10000 ] [ set max-sheep 30000 ]

  ; Check model-version switch
  ; if we're not modeling grass, then the sheep don't need to eat to survive
  ; otherwise each grass' state of growth and growing logic need to be set up
  ifelse model-version = "sheep-wolves-grass" [
```

ticks: 29

model-version: sheep-wolves-grass
grass-regrowth-time: 30
initial-number-sheep: 100
initial-number-wolves: 50
sheep-gain-from-food: 4
wolf-gain-from-food: 20
sheep-reproduce: 4 %
wolf-reproduce: 5 %

sheep: 135
wolves: 42
grass: 279

340
0
0
100



WHAT IS IT?

This model explores the stability of predator-prey ecosystems. Such a system is called unstable if it tends to result in extinction for one or more species involved. In contrast, a system is stable if it tends to maintain itself over time, despite fluctuations in population sizes.

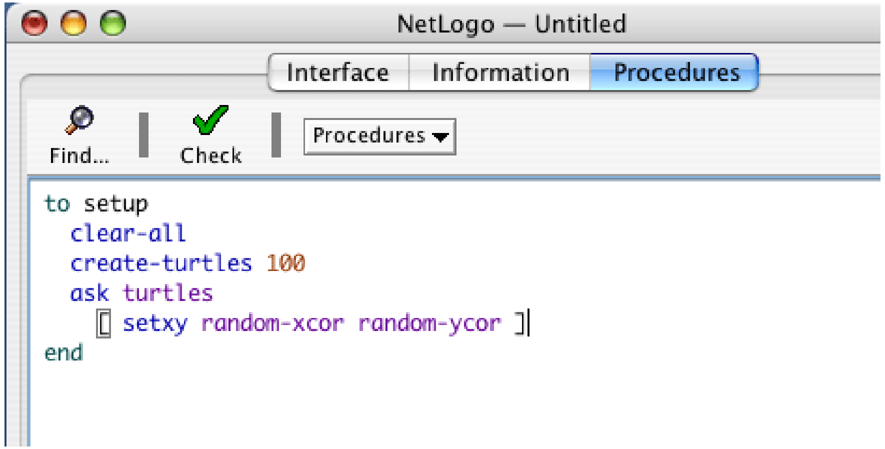
HOW IT WORKS

There are two main variations to this model.

In the first varia
the landscape,
and they must e
die. To allow the
reproducing at e

```
globals [ max-sheep ]  ; don't let the sheep population grow too large

; Sheep and wolves are both breeds of turtles
breed [ sheep a-sheep ]  ; sheep is its own plural, so we use "a-sheep" as the singular
breed [ wolves wolf ]

turtles-own [ energy ]       ; both wolves and sheep have energy

patches-own [ countdown ]    ; this is for the sheep-wolves-grass model version

to setup
  clear-all
  ifelse netlogo-web? [ set max-sheep 10000 ] [ set max-sheep 30000 ]

  ; Check model-version switch
  ; if we're not modeling grass, then the sheep don't need to eat to survive
  ; otherwise each grass' state of growth and growing logic need to be set up
  ifelse model-version = "sheep-wolves-grass" [
    ask patches [
      set pcolor one-of [ green brown ]
      ifelse pcolor = green
        [ set countdown grass-regrowth-time ]
      [ set countdown random grass-regrowth-time ] ; initialize grass regrowth clocks random
    ]
  ]
  [
    ask patches [ set pcolor green ]
```

Programação em NetLogo: Teoria



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

Comandos

- Realizam ações

Reporters

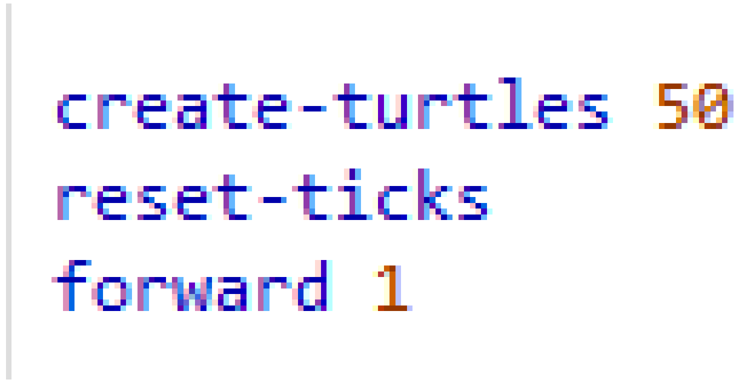
- Retornam algum valor/agente ou um conjunto de valores/agentes

Procedimentos

- Comandos e procedimentos criados pelo modelador

[NetLogo 7.0.2: Programming Guide]

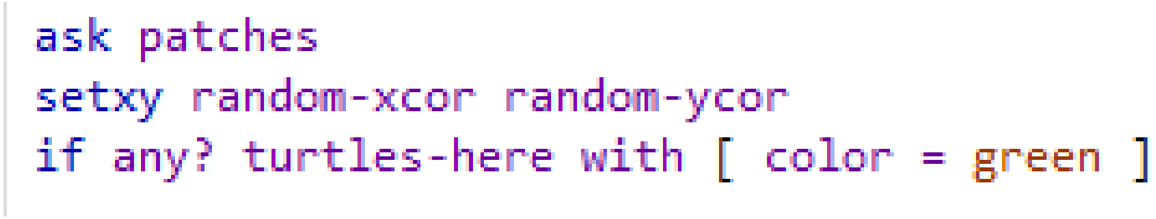
Programação em NetLogo: Comando



```
create-turtles 50
reset-ticks
forward 1
```

- Cria 50 agentes Turtles
- Reinicia o contador de ticks
- Comanda uma ou mais Turtles a darem 1 passo

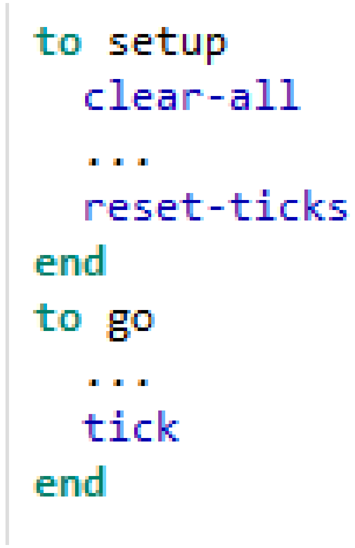
Programação em NetLogo: Reporter



```
ask patches
setxy random-xcor random-ycor
if any? turtles-here with [ color = green ]
```

- Retorna todos os agentes Patches
- Retorna um valor de coordenada X e Y aleatório
- Retorna qualquer agente Turtle aqui que possui a cor verde

Programação em NetLogo: Procedimento



```
to setup
  clear-all
  ...
  reset-ticks
end
to go
  ...
  tick
end
```

- Comando "to" inicializa o procedimento
- Comando "end" finaliza o procedimento

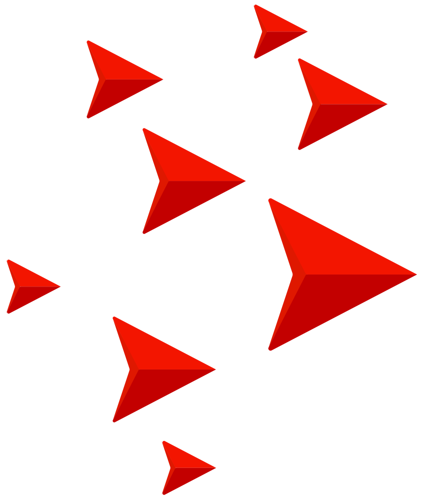
Documentação do NetLogo

Link do Dicionário (Versão 6.3): NetLogo 6.3.0 User Manual: NetLogo Dictionary

Guia de Programação para Iniciantes em NetLogo: Welcome to Beginner's Interactive NetLogo Dictionary (BIND)

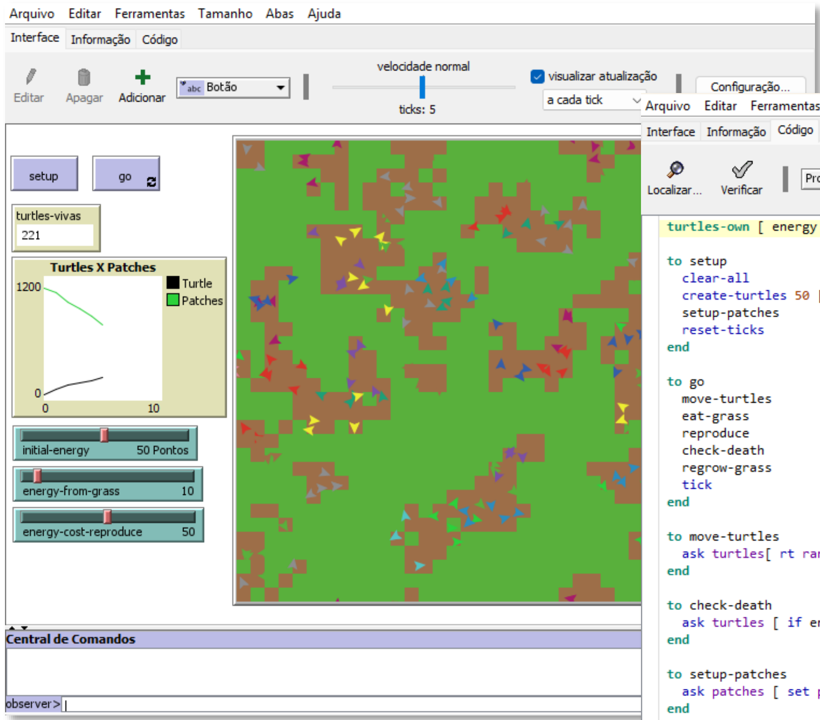
11 primitivas essências para novatos estudarem: First 11 Netlogo Primitives To Learn | Beginner's Interactive NetLogo Dictionary

Guia de Estudo do Minicurso: https://www.notion.so/Roteiro-Minicurso-NetLogo-2854aa82defa80fdb59bee6ced4c77de?source=copy_link



UDESC

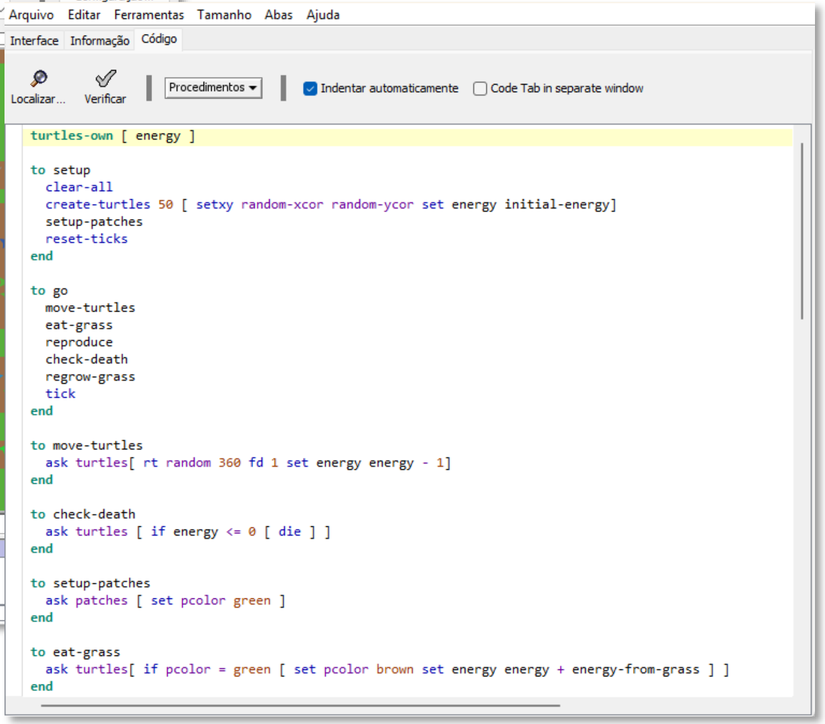
Programação Em NetLogo: Prática



velocidade normal
ticks: 5
☑ visualizar atualização
a cada tick

turtles-vivas 221

initial-energy 50 Pontos
energy-from-grass 10
energy-cost-reproduce 50


Arquivo Editar Ferramentas Tamanho Abas Ajuda
Interface Informação Código
Localizar... Verificar
Procedimentos

☑ Indentar automaticamente
☐ Code Tab in separate window

```
turtles-own [ energy ]

to setup
  clear-all
  create-turtles 50 [ setxy random-xcor random-ycor set energy initial-energy]
  setup-patches
  reset-ticks
end

to go
  move-turtles
  eat-grass
  reproduce
  check-death
  regrow-grass
  tick
end

to move-turtles
  ask turtles[ rt random 360 fd 1 set energy energy - 1]
end

to check-death
  ask turtles [ if energy <= 0 [ die ] ]
end

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

to eat-grass
  ask turtles[ if pcolor = green [ set pcolor brown set energy energy + energy-from-grass ] ]
end
```

Procedimentos setup, go e move-turtles

```
to setup
clear-all
create-turtles 50 [
setxy random-xcor random-
ycor
]
ask patches[
setup-patches
]
reset-ticks
end
```

```
to go
ask turtles[
move-turtles
]
tick
end

to move-turtles
lt random 180
rt random 180
forward 1
end
```

Implementando variável de Energia

```
turtles-own [ energy ]

to setup
  clear-all
  create-turtles 50 [
   setxy random-xcor random-ycor
   set energy random 50
  ]

  reset-ticks
end
```

```
to go
 ask turtles[
   move-turtles
    check-death
   ]
  tick
end

to move-turtles
  lt random 180
  rt random 180
  forward 1
 set energy energy - 1
end

to check-death
  if energy <= 0 [ die ]
end
```

Configurando o alimento grama

```
to setup
  clear-all
  create-turtles 50 [
   setxy random-xcor random-ycor
   set energy random 50
  ]
  ask patches[
    setup-patches
  ]
  reset-ticks
end

to go
 ask turtles[
   move-turtles
    eat-grass
    check-death
  ]
  regrow-grass
  tick
end
```

```
to eat-grass
  if pcolor = green [ set pcolor brown set energy energy + 10 ]
end

to setup-patches
  set pcolor green
end

to regrow-grass
  ask patches with [ pcolor = brown ][
    if random 100 < 3 [ set pcolor green ]
  ]
end
```

Configurando a reprodução

```
to go
 ask turtles[
   move-turtles
     eat-grass
     reproduce
     check-death
  ]
  regrow-grass
  tick
end
```

```
to reproduce
  if energy > 100 [ set energy energy - 100 hatch 1 [ set energy 50 ]
]
end
```

Modelo de Ecossistema

```
turtles-own [ energy ]

to setup
  clear-all
  create-turtles 50 [
   setxy random-xcor random-ycor
   set energy random 50
  ]
  ask patches[
    setup-patches
  ]
  reset-ticks
end

to go
 ask turtles[
   move-turtles
    eat-grass
    reproduce
    check-death
  ]
  regrow-grass
  tick
end
```

```
to eat-grass
  if pcolor = green [ set pcolor brown set energy energy + 10 ]
end

to move-turtles
  lt random 180
  rt random 180
  forward 1
  set energy energy - 1
end

to check-death
  if energy <= 0 [ die ]
end

to setup-patches
  set pcolor green
end

to regrow-grass
  ask patches with [ pcolor = brown ][ if random 100 < 3 [ set pcolor green ] ]
end

to reproduce
  if energy > 100 [ set energy energy - 100 hatch 1 [ set energy 50 ]  ]
end
```

UDESC

Conclusões

Conclusões

- NetLogo é uma ótima ferramenta para simular ambientes naturais ou sociais, onde diversos agentes se fazem presente dentro da simulação, podendo conter suas próprias ações.

- Possui uma linguagem bem amigável para quem está aprendendo a ferramenta, mas permitindo também a criação de complexos modelos de simulação.

- Aqui foi aprendido um básico modelo de ecossistema, mas com esses conhecimentos já é o suficiente para fazer modelos mais avançados, como os vistos em aula.

UDESC

Simulações Baseadas em Agentes: Aprendizado da ferramenta NetLogo

Mateus Rissardi
Engenheiro de Software / UDESC Alto Vale
mrissardi01@gmail.com

Fernando Santos
Professor / UDESC Alto Vale
fernando.santos@udesc.br

ESO
Engenharia de Software

UDESC
UNIVERSIDADE DO ESTADO DE SANTA CATARINA

ERAMIA-RS
I ESCOLA REGIONAL DE APRENDIZADO DE MÁQUINA E INTELIGÊNCIA ARTIFICIAL DA REGIÃO SUL
2025