

Aquarium

Engenharia Informática
Inteligência Artificial
1º Trabalho Prático
26/10/2022

Por:

António Teixeira, 70835

Índice

Introdução	3
Aquarium I	4
Aquarium II	10
Conclusão.....	18
Referências Bibliográficas	19

Introdução

“Pretende-se promover a aquisição de conhecimentos e desenvolvimento de competências fundamentais relativas à modelação e simulação computacional de sistemas com agentes racionais utilizando a ferramenta NetLogo” - protocolo do trabalho prático 1.

No presente relatório será descrito o funcionamento, tal como a criação do modelo Aquarium 1 e 2.

Netlogo é uma linguagem de programação e ambiente integrado de desenvolvimento para modelação centrada em “agentes”, cuja função principal é simular as ações e interações de diferentes entidades, tanto individuais como coletivas.

Neste trabalho, foi solicitada a criação de um protótipo “Aquarium 1”, que permitisse a observação e ensaio de um “aquário” com as seguintes características:

- Povoado por duas espécies de peixes, que podem coabitar pacificamente;
- O ambiente ser constituído por diversas células que representam água;
- A condição inicial do aquário ser de “boas condições em termos de limpeza”;
- Mediante o passar do tempo existir declínio das condições da água;
- Os peixes apresentarem movimento direcionado livre, apenas tendo apreensão da célula em que se encontram presentes;
- Existência de “botão Setup”, que permite a redefinição do ambiente;
- Existência de “botão Go”, que impulsiona a circulação das espécies no aquário de forma aleatória;
- Existência de “botão Feed”, com a função de alimentar os peixes
- Existência de “deslizadores” que definem o número de elementos de cada espécie;
- Gráfico que expõe a evolução de ambas as espécies;

Aquarium I

1. Foi desenhado, em primeiro lugar, o ambiente “aquário”.

```
to setup-water
  ask patches
  [set pcolor blue]
end

to setup-floor
  ask patches with [pxcor >= -16 and pxcor <= 16 and pycor = -16] [set pcolor yellow]
end
```

Figura 1: Criação da água e areia do aquário.

2. Foram criadas variáveis globais essenciais ao funcionamento do programa.

```
globals [
  countcells

  x
  y

  num-fish-right
  num-fish-left
  sum-right-turns
  sum-left-turns

  males
  females
  n-platy-fish
  n-guppy-fish
  totalfish

  fish-age
  snail-age
  ;;guppy-age

  watergood
  n-newborn
  n-eaten
  n-starved
  n-old-age
  n-pollution

  ;;garbage

  message-shown-change-water? replace-water
]
```

Figura 2: Variáveis globais iniciais.

3. Criaram-se as espécies a coabitar no ambiente.

```
breed [male-platy male-platy-fish]
breed [female-platy female-platy-fish]
breed [male-guppy male-guppy-fish]
breed [female-guppy female-guppy-fish]

breed [fish-food food]
```

Figura 3: Espécies de peixe, sexos. Comida.

```
to create-fish
  set shape "fishr"

  ;;set pen-size 3
  set size 2.5

  ;;set heading 0
  set heading one-of [0 90 180 270 360]
  setxy -16 + (random(32)) -16 + (random(32))

  set fish-age 0
end
```

Figura 4: Definição de peixe.

Aos peixes foi atribuída dimensão 2.5 e movimento aleatório em todas as direções. Foi utilizada uma forma própria para possibilitar o diferente desenho.

4. Foram criadas duas espécies de peixes com dois sexos por cada espécie com o objetivo de se reproduzirem. Foi também definida a qualidade da água inicial 80.

```
to setup-fish
  create-male-platy number-of-male-platy-fish
  [
    create-fish
    set color orange

    ;;setxy random-float world-width
    ;;random-float world-height

    set energy 30

    set platy-leader nobody
    set platy-follower nobody
  ]

  create-female-platy number-of-female-platy-fish
  [
    create-fish
    set color orange

    set energy 30

    set platy-leader nobody
    set platy-follower nobody
  ]

  create-male-guppy number-of-male-guppy-fish
  [
    create-fish
    set color yellow

    set energy 30

    set guppy-leader nobody
    set guppy-follower nobody
  ]

  create-female-guppy number-of-female-guppy-fish
  [
    create-fish
    set color yellow

    set energy 30

    set guppy-leader nobody
    set guppy-follower nobody
  ]

  set watergood 80
end
```

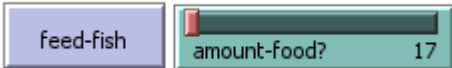
Figura 5: Criação dos peixes e sexos.

5. Foi definido o movimento aleatório dos peixes, funcionalidade executável com recurso à utilização do botão “Go”.

```
ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
[
  fd .5 ;; wander around and look for food]
  ifelse ((random 2) = 0)
  [
    lt random 16
    set num-fish-left (num-fish-left + 1)
  ]
  [
    rt random 16
    set num-fish-right (num-fish-right + 1)
  ]
  set energy energy - .05
  set age age + .1
  bounce
]
```

Figura 6: Movimento aleatório.

6. Foi criado um botão com a finalidade de alimentar os peixes havendo a possibilidade de seleccionar a quantidade de comida a fornecer para alimentação.



```
to feed-fish
  create-fish-food amount-food?
  [
    set shape "circle"
    set color white
    set pen-size 3
    ;;set size 2.5

    set x -16 + (random (33))
    setxy x 16

    ;;set heading 0
    ;;set heading one-of [0 90 180 270 360]
    set heading 180

    ;;set food-rot 0
  ]
end
```

Figura 7: Botão alimentar.

Figura 8: Criação da comida.

7. A alimentação fornecida através do botão anteriormente referido, após a sua ativação, só é assimilada pelos peixes quando se cruza com os mesmos. Na sequência da alimentação os peixes aumentam de tamanho. O alimento que não é consumido pelos peixes provoca contaminação da água, cuja qualidade diminui, mudando da cor transparente para a cor verde.

```
to eat
ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
[
  ;;if any? (turtles-on patch-ahead 2) with [breed = fish-food]
  if any? (turtles-on patch-here) with [breed = fish-food]
  [
    ask (turtles-on patch-here) with [breed = fish-food]
    [
      die
    ]
    set energy energy + 21
    if size < 4
    [set size size + .2]
    set watergood watergood - .01
  ]
]
end
```

Figura 9: Definição de alimentação.

```
to food-down
ask fish-food
[
  if (abs [pxcor] of patch-ahead 1 = 16)
  [
    ifelse (abs [pycor] of patch-ahead 1 = max-pycor)[stop]
    [forward 1]
  ]
  if (abs [pxcor] of patch-ahead 1 = max-pxcor)[stop]
  if (abs [pycor] of patch-ahead 1 = max-pycor)[stop]
  forward 1

  ask patch-here [set food-rot food-rot + .01]

  ask patch-here [
    if (food-rot > 19)
    [
      ;;if random-float 1 < 0.19 [set pcolor green]
      if random-float 1 < 0.1 [set pcolor scale-color green food-rot 0 150]
    ]
  ]
]
end
```

Figura 10: Comida desce em direção ao fundo do aquário.

8. Definiu-se que por cada movimento dos peixes estes envelhecem 0.1 de idade atingindo o limite máximo de idade aos 80. Quando chegam ao 80 de idade morrem. Definiu-se ainda que, na sequência da contaminação da água e consequente diminuição da qualidade, caso esta atinja um valor de qualidade inferior a 10 e a quantidade de peixes no aquário seja superior a 3, o programa aleatoriamente elimina 1 peixe, caso o número de peixes inferior a 3, morrem ambos, adicionado ao número de mortes por poluição.

```
to kickbucket
ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
[
  if (energy <= 0) ;;or (age > 80)
  [
    set n-starved n-starved + 1
    die
  ]
  if (age > 80)
  [
    set n-old-age n-old-age + 1
    die
  ]
]
if watergood < 10
[
  ifelse (count turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy] >= 3) ;; if more than 3 fish, randomly pick one
  [
    ask n-of 3 turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
    [
      set n-pollution n-pollution + 1
      die
    ]
  ]
  ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy] [
    set n-pollution n-pollution + 1
    die
  ]
]
end
```

Figura 11: Tempo de vida dos peixes.

Aquarium II

Adicionaram-se as seguintes inovações ao aquário (ambiente dos peixes):

Reprodução;

Substituição da água do aquário;

Filtro da água com o objetivo de limpar a água (elimina a cor verde);

Movimentação dos peixes em grupo (cardume);

Adição de caracóis (com a possibilidade de escolher o número de elementos);

Introdução de energia nos peixes através da alimentação;

Poluição gerada pelos peixes;

Capacidade de eliminação do número de peixes através do canibalismo;

Criação de diversos gráficos de monitorização da atividade no interior do aquário, designadamente, evolução (número total de peixes, número parcial por espécie e por sexo), gráfico da evolução dos sexos, gráfico da evolução das duas espécies de peixes existentes, gráfico indicador da qualidade da água, contabilização de movimentos à esquerda, à direita e total de células percorridas, monitor indicador dos valores da qualidade da água, monitor indicador do nascimento de peixes, monitores indicadores de óbitos por falta de alimento, idade, poluição e canibalismo;

1. Reprodução dos peixes associada a uma probabilidade inserida pelo utilizador.



Figura 12: Probabilidade de nascimento.

```
to spawn
  if random-float 1 < probability-spawn
  [
    if watergood > 60[
      ifelse ((random 2) = 0)
      ;;ifelse ((random probability-spawn) = 0)

      [ask turtles with [breed = female-platy and (energy >= 30) and (size >= 3)]
      [if any? (male-platy in-radius 5) with [ (energy >= 30) and (size >= 3)]
      [
        hatch 20
        [
          set n-newborn n-newborn + 1
          set age 0
          set size 1
          set energy 15

          ifelse random 2 = 0
          [
            set breed male-platy
          ]
          [
            set breed female-platy
          ]

          set color orange
          set shape "fishr"

          ;;lt 45
          ;;fd 3
        ]
      ]
    ]
  ]
  [ask turtles with [breed = female-guppy and (energy >= 30) and (size >= 3)]
  [if any? (male-guppy in-radius 5) with [ (energy >= 30) and (size >= 3)]
  [
    hatch 20
    [
      set n-newborn n-newborn + 1
      set age 0
      set shape "fishr"
      set size 1
      set energy 15

      ifelse random 2 = 0
      [
        set breed male-guppy
      ]
      [
        set breed female-guppy
      ]

      set color yellow
      set shape "fishr"
    ]
  ]
]
end
```

Figura 13: Nascimentos de acordo com a espécie de progenitores.

Por predefinição, os peixes só nascem num ambiente em que a qualidade da água é superior a 60 e energia dos progenitores maior, ou igual a 30, tal como o tamanho superior ou igual a 3. Aquando do nascimento:

- É automaticamente definido o tamanho 1, idade 0, energia 15, a cada peixe;
- Os nascimentos ocorrem em número de 20 peixes, valor que é contabilizado no monitor respetivo;
- O sexo dos peixes é escolhido aleatoriamente;

2. Através da ação no botão respectivo, a água de cor verde é substituída, pelo que muda de cor ficando azul o que implica a alteração dos dados relativos à poluição e do monitor da qualidade da água.



Figura 14: Monitorização da qualidade da água.

3. O filtro está predefinido para funcionar quando a qualidade da água é inferior a 100, nesse caso, de forma automática, adiciona qualidade à água, por forma a neutralizar a poluição.

```
to filterwater
  if watergood < 100
  [
    set watergood watergood + filterflow? / 25
    ask patches with [food-rot > 0][set food-rot abs((food-rot - filterflow? / 10))]
  ]
end
```

Figura 14: Filtragem da água.

4. Deslocação em cardume, é associada a uma probabilidade de escolha e/ou autonomação de líderes para cada uma das espécies de peixe.

```
to setup-fish
  create-male-platy number-of-male-platy-fish
  [
    create-fish
    set color orange

    ;;setxy random-float world-width
    ;;random-float world-height

    set energy 30

    set platy-leader nobody
    set platy-follower nobody
  ]

  create-female-platy number-of-female-platy-fish
  [
    create-fish
    set color orange

    set energy 30

    set platy-leader nobody
    set platy-follower nobody
  ]

  create-male-guppy number-of-male-guppy-fish
  [
    create-fish
    set color yellow

    set energy 30

    set guppy-leader nobody
    set guppy-follower nobody
  ]

  create-female-guppy number-of-female-guppy-fish
  [
    create-fish
    set color yellow

    set energy 30

    set guppy-leader nobody
    set guppy-follower nobody
  ]

  set watergood 80
end
```

Figura 15: Atribuição a cada peixe de líder e seguidor nulo.

```

to attach-platy
  ;;let xd near-radius + random (far-radius - near-radius)
  ;;let yd near-radius + random (far-radius - near-radius)

  ;;if random 2 = 0 [set xd (- xd)]
  ;;if random 2 = 0 [set yd (- yd)]

  let platy-candidate one-of turtles with [(breed = male-platy or breed = female-platy) and platy-follower = nobody]

  if (platy-candidate = nobody) [stop]
  ask platy-candidate [set platy-follower myself]
  set platy-leader platy-candidate

  ;;ifelse platy-follower = nobody[] [set color red];;change color]
  ;;ifelse platy-leader = nobody[] [set color red];;change color]
end

to turn-platy
  ifelse platy-leader = nobody
  []
  [face platy-leader]
end

to attach-guppy
  ;;let xd near-radius + random (far-radius - near-radius)
  ;;let yd near-radius + random (far-radius - near-radius)

  ;;if random 2 = 0 [set xd (- xd)]
  ;;if random 2 = 0 [set yd (- yd)]

  let guppy-candidate one-of turtles with [(breed = male-guppy or breed = female-guppy) and guppy-follower = nobody]

  if (guppy-candidate = nobody) [stop]
  ask guppy-candidate [set guppy-follower myself]
  set guppy-leader guppy-candidate

  ;;ifelse follower = nobody [change color]
  ;;ifelse leader = nobody [change color]
end

to turn-guppy
  ifelse guppy-leader = nobody
  []
  [face guppy-leader]
end

```

Figura 16: Cada espécie reconhece o respetivo líder e volta-se na sua direção.

```

to follow-fish
  if random-float 1 < probability-follow [
    ask turtles with [breed = male-platy or breed = female-platy]
    [
      if any? (turtles-on patch-ahead 2) with [breed = male-platy or breed = female-platy]
      [
        ;;follow
        if platy-leader = nobody [attach-platy]
        ;turn-platy
      ]
    ]

    ask turtles with [breed = male-guppy or breed = female-guppy]
    [
      if any? (turtles-on patch-ahead 2) with [breed = male-guppy or breed = female-guppy]
      [
        ;;follow
        if guppy-leader = nobody [attach-guppy]
        ;turn-guppy
      ]
    ]
  ]
end

```

Figura 17: Aquando da probabilidade escolhida, cada peixe segue o líder da espécie.



Figura 18: Probabilidade de deslocação em cardume.

5. É possível a adição de caracóis.



Figura 19: Escolha do número de caracóis.

```
to setup-snail
  create-snails number-of-snails
  [create-snail]
end
```

Figura 20: Criação de caracóis.

```
to create-snail
  set shape "bug"
  set size 1.5

  set heading one-of [90 270]
  setxy -16 + (random(32)) -15

  set color green

  set snail-age 0
  set energy 50
end
```

Figura 21: Definição de caracóis.

- Introdução de energia nos peixes através da alimentação. Por predefinição por cada alimento ingerido por cada peixe corresponde a 21 de energia, o que aumenta a esperança média de vida. Quando o valor da energia de um determinado peixe atinge o valor 0 esse peixe morre, valor que é inscrito no monitor de óbitos à fome.

```
to eat
  ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
  [
    ;;if any? (turtles-on patch-ahead 2) with [breed = fish-food]
    if any? (turtles-on patch-here) with [breed = fish-food]
    [
      ask (turtles-on patch-here) with [breed = fish-food]
      [
        die
      ]
      set energy energy + 21
      if size < 4
      [set size size + .2]
      set watergood watergood - .01
    ]
  ]
end
```

Figura 22: Introdução de energia na alimentação.

```
to kickbucket
  ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
  [
    if (energy <= 0) ;;or (age > 80)
    [
      set n-starved n-starved + 1
      die
    ]
    if (age > 80)
    [
      set n-old-age n-old-age + 1
      die
    ]
  ]
  if watergood < 10
  [
    ifelse (count turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy] >= 3) ;; if more than 3 fish, randomly pick one
    [
      ask n-of 3 turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
      [
        set n-pollution n-pollution + 1
        die
      ]
    ]
    [
      ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy] [
        set n-pollution n-pollution + 1
        die
      ]
    ]
  ]
end
```

Figura 22: Óbitos por falta de alimento.

- Poluição gerada pelos peixes.

```
to pollute-water
  let big-fish 0
  let little-fish 0
  let bigpollutes 0
  let littlepollutes 0

  set big-fish count turtles with [size >= 3.5]
  set little-fish count turtles with [size < 3.5]
  set bigpollutes big-fish * .003
  set littlepollutes little-fish * .005
  set watergood watergood - (bigpollutes + littlepollutes)
end
```

Figura 23: Distinção entre a poluição gerada pelos peixes maiores e pelos peixes mais pequenos.

8. Característica introduzida nos peixes de maior tamanho que lhes permite comer os de menor tamanho gerando com isso energia em proveito próprio. Este comportamento contribui para a poluição. Esta função é acionada pelo botão próprio.

```
to eat-smaller-fish ;; eat fish that are more than .5 smaller than
  let foodenergy 0
  let mysize 0
  set foodenergy 0
  ask turtles with [breed = male-platy or breed = female-platy or breed = male-guppy or breed = female-guppy]
  [
    set mysize size
    if any? (turtles-on patch-ahead 2) with [size <= (mysize - .25)]
    [
      ask turtles-on patch-ahead 2
      [
        set foodenergy size
        set n-eaten n-eaten + 1
        die
      ]
    ]
  ]
  set energy energy + foodenergy
  set watergood watergood - .1
]
end
```

Figura 24: Introdução do canibalismo.

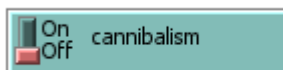


Figura 25: Canibalismo por switch.

9. Criação de gráficos e monitores indicadores da atividade.

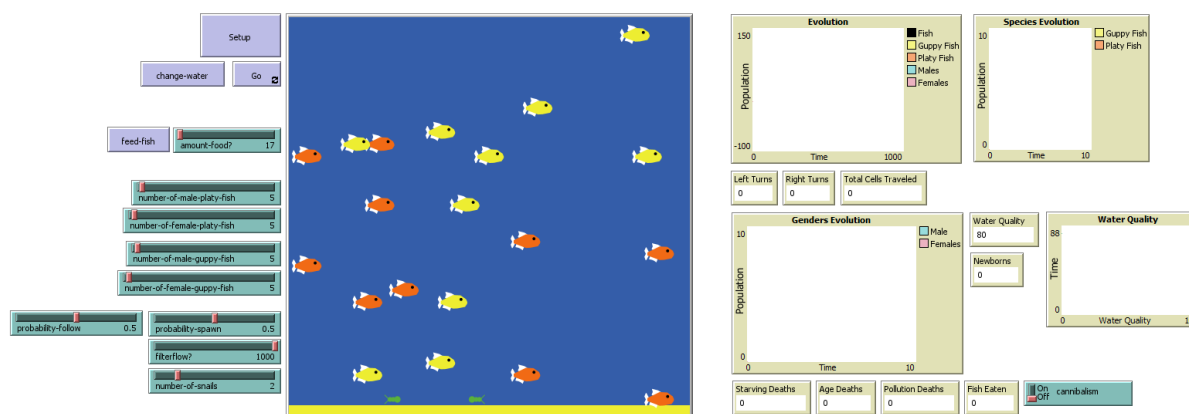


Figura 26: Monitorização.

Conclusão

Este trabalho foi muito enriquecedor, na medida em que aprendi, desenvolvi e consolidei diversos conhecimentos, tendo a necessidade de ir à descoberta de informação através de inúmeras pesquisas que realizei no motor de busca do Google.

Neste momento estou muito satisfeito por ter concluído este trabalho de forma positiva, o que me confere motivação para continuar a trabalhar.



Referências Bibliográficas

- [1] NetLogo Models Library ([northwestern.edu /netlogo/models/index.cgi](http://northwestern.edu/~netlogo/models/index.cgi)) acedido em 22 e 23/10/2022
- [2] NetLogo User Community Models ([northwestern.edu/netlogo/models/community/index.cgi](http://northwestern.edu/~netlogo/models/community/index.cgi)) acedido em 22 e 23/10/2022