

Trabalho Prático 1

Agentes Racionais

Licenciatura em Engenharia Informática

Inteligência Artificial

José Paulo Barroso de Moura Oliveira

Eduardo José Solteiro Pires

**Autores**

Diogo Medeiros n.º 70633

Rui Pinto n.º 70648

Vila Real, novembro 2021

**RESUMO**

Este relatório visa descrever a execução do trabalho prático desenvolvido no âmbito da Unidade Curricular de Inteligência Artificial, pressupondo a aquisição de competências relativas à modelação e simulação computacional de sistemas com agentes racionais, utilizando a ferramenta NetLogo.

A sua execução teve por base o enquadramento teórico dos conceitos adquiridos e aplicados, bem como a apresentação e descrição do sistema desenvolvido, incluindo imagens descritivas das várias etapas / fases do sistema em execução, de acordo com as condições iniciais.

**ÍNDICE**

[1. INTRODUÇÃO 1](#_Toc87622963)

[2. AGENTES RACIONAIS 1](#_Toc87622964)

[2.1 Enquadramento do Problema 1](#_Toc87622965)

[2.1.1 Estrutura dos Agentes – Arquitetura e Programa 3](#_Toc87622966)

[2.2 Resolução do Problema 6](#_Toc87622967)

[3. NOTAS FINAIS 14](#_Toc87622968)

[**BIBLIOGRAFIA** 14](#_Toc87622969)

[ANEXO A – Código de TP1\_torus.nlogo 15](#_Toc87622970)

[ANEXO B – Código de TP1\_box.nlogo 18](#_Toc87622971)

1. INTRODUÇÃO

No âmbito da Unidade Curricular de Inteligência Artificial, foi solicitado um trabalho prático que consiste no desenvolvimento de um sistema computacional com agentes racionais usando a ferramenta NetLogo.

No nosso caso, o ambiente retratado representa um campo (ou terreno de plantação), e os agentes modelados são o planta-relva e as toupeiras, responsáveis, respetivamente, pela plantação e destruição da relva.

1. AGENTES RACIONAIS

“Um agente é qualquer coisa que perceciona o seu ambiente através de sensores e atua sobre esse ambiente através de atuadores.” [1]

* 1. Enquadramento do Problema

Um agente racional é aquele que toma a decisão correta, sendo necessário definir o contexto da decisão, o que considera certo e o que considera errado. Quando colocado num ambiente, o agente gera uma sequência de ações de acordo com estímulos (ou perceções). Esta desencadeia uma sequência de estados no ambiente que, quando favorável, determina o sucesso do agente, que por sua vez é avaliado de acordo com uma medida de performance.

A definição de racionalidade de um agente tem por base [1]:

* A medida de desempenho que define o critério de sucesso.
* O conhecimento prévio do agente sobre o ambiente.
* As ações que o agente pode executar.
* A sequência de perceções do agente até à data.

Segue-se um quadro descritivo, onde são identificados os agentes e respetivas perceções e ações, de acordo com o nosso sistema.

|  |  |  |
| --- | --- | --- |
| **AGENTES** | Planta-relva | Toupeira |
| **PERCEÇÕES / ESTÍMULOS** | 2 células: atual e à frente  2 estados: plantado e por plantar  Presença de toupeira | 2 células: atual e à frente  2 estados: plantado e por plantar  Presença de toupeira  Presença de planta-relva |
| **AÇÕES** | Mudar de direção (360º)  Andar para a frente  Plantar relva  Matar toupeira | Mudar de direção (360º)  Andar para a frente  Comer relva  Reproduzir-se  Não operar |

“Ao desenhar um agente, o primeiro passo deve ser sempre especificar o ambiente de tarefa o mais detalhadamente possível. A sigla PEAS significa *Performance*, *Environment*, *Actuators*, *Sensors*.” [1]

O quadro seguinte caracteriza a descrição PEAS para os agentes em causa.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Agente** | **Performance** | **Ambiente** | **Atuadores** | **Sensores** |
| Planta-Relva | Relva plantada, toupeiras mortas | Campo, toupeiras | \* | “Olhos” |
| Toupeiras | Relva destruída, saúde, reprodução | Campo, planta-relva, toupeiras | \* | “Olhos” |

\* No caso da ferramenta usada, os agentes movem-se entre patches e interagem com estes e com outros agentes usando primitivas (i.e., *forward* e *die*)

Relativamente aos ambientes, é possível caracterizá-los de acordo com as suas propriedades, sendo, no caso do campo de relva, as seguintes:

* Parcialmente observável – quer o planta-relva, quer as toupeiras só têm acesso ao estado da célula atual e à sua frente
* Multiagente – o campo contém o planta-relva e pode conter múltiplas toupeiras
* Estocástico – os movimentos dos agentes podem ser aleatórios
* Episódico – quer o planta-relva, quer as toupeiras tomam as suas decisões com base na informação que percecionam no momento, não possuindo “memória”
* Dinâmico – caso haja várias toupeiras no campo, o estado do mesmo pode mudar durante a decisão de qualquer uma delas, face aos movimentos das restantes
* Contínuo – há um conjunto infinito de estados do campo, dependendo do número de toupeiras vivas, do terreno já plantado e por plantar e das ações dos agentes
  + 1. Estrutura dos Agentes – Arquitetura e Programa

Raças de agentes (*breeds*):

* planta-relvas / planta-relva
* toupeiras / toupeira

Variáveis globais do ambiente (*globals*):

* *Modo\_de\_Plantar* – modo de plantação (“Para a frente”, “Serpentina” ou “Manutenção”)
* *Prob\_toupeira* – determina a atividade das toupeiras
* *default-time* – tempo (em ticks) que as toupeiras devem esperar entre reproduções
* *max-iter* – número máximo de testes realizados por um agente, por tick

Variáveis do planta-relva (*planta-relvas-own*):

* *stock* – stock atual de relva

Variáveis das toupeiras (*toupeiras-own*):

* *health* – vida/saúde da toupeira
* *elapsed-time* – tempo (em ticks) passado após se reproduzir

Inicialmente, escolhe-se o modo de plantação, “Para a frente” ou “Serpentina”. De seguida faz-se a configuração inicial, criando o planta-relva com uma direção de acordo com o modo de plantação e o stock de relva necessário para plantar o campo todo.

|  |  |
| --- | --- |
| “Para a frente” | “Serpentina” |
| direção ← Frente | direção ← RANDOM(Frente, Trás, Esquerda, Direita) |

Após a configuração inicial, procede-se à plantação, a qual dependerá do modo escolhido.

|  |  |
| --- | --- |
| “Para a frente” | “Serpentina” |
| PlantarRelva()  IF célula à frente = plantada THEN  VirarDireita()  END IF  AndarFrente() | PlantarRelva()  IF AtingiuLimite() THEN  InverterSentido()  END IF  AndarFrente() |

Em ambos os modos, quando a plantação for concluída, o modo de plantação muda automaticamente para “Manutenção”. Imediatamente a seguir, o planta-relva procede à reposição do seu stock de relva.

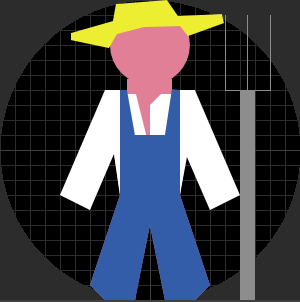
|  |  |
| --- | --- |
| Algoritmos dos agentes | |
| Planta-relva | Toupeira |
| IF Aqui(toupeiras) THEN Matar()  IF stock = 0 THEN  ReporStock()  ELSE  IF célula = por plantar THEN  PlantarRelva()  END IF  iter ← 0  WHILE iter < max-iter  AND ÀFrente(toupeiras) = false  AND célula à frente = plantada DO  iter ← iter + 1  direção ← RANDOM(0, 360)  END WHILE  AndarFrente()  END IF | IF Aqui(planta-relvas) THEN Morrer()  IF célula = plantada THEN  DestruirRelva()  GanharVida()  ELSE  PerderVida()  END IF  IF health = 0 THEN Morrer()  IF Prob(Prob\_toupeira) THEN  IF Aqui(toupeiras) = true  AND elapsed-time < default-time THEN  Reproduzir()  elapsed-time ← 0  END IF  iter ← 0  WHILE iter < max-iter  AND ÀFrente(toupeiras) = false  AND célula à frente = por plantar DO  iter ← iter + 1  direção ← RANDOM(0, 360)  END WHILE  AndarFrente()  ELSE  NOP  END IF |

No caso do planta-relva, este começa por matar quaisquer toupeiras (eliminar os agentes) que encontre na sua célula. De seguida verifica o seu stock de relva. Caso já não tenha stock, procede à reposição do mesmo. Caso contrário, prossegue. Se a célula atual se encontrar por plantar (castanha), trata de a plantar. Conseguinte, verifica se a célula à sua frente está por plantar (castanha) ou contêm toupeiras. Caso isto não se verifique, escolhe uma direção aleatória. O planta-relva testa estas condições até se verificarem, ou, em contrapartida, até atingir um número máximo de testes, definido por *max-iter*. Por fim, anda para a frente.

No caso da toupeira, esta verifica se está na presença do planta-relva. Caso afirmativo, esta morre (o agente é eliminado). Na situação em que a probabilidade gerada é inferior a *Prob\_toupeira*, a toupeira procede com a sua atividade. Caso contrário, não opera (*nop*). De seguida, verifica se a célula em que se encontra tem relva (verde) e, se sim, destrói a relva (muda para castanho). Se destruir a relva, ganha vida, caso contrário, perde. Se a sua vida atingir o valor nulo, a toupeira morre. Caso tenha passado o tempo necessário (*default-time*), a toupeira reproduz-se quando se cruza com outra toupeira. De seguida, verifica se a célula à sua frente está plantada (verde) ou contêm toupeiras. Caso isto não se verifique, escolhe uma direção aleatória. A toupeira testa estas condições até se verificarem, ou, em contrapartida, até atingir um número máximo de testes, definido por max-iter. Por fim, anda para a frente.

* 1. Resolução do Problema

Segue-se um conjunto de imagens capturadas durante o funcionamento do sistema, retratando as várias fases do mesmo em execução, de acordo com certas condições iniciais.

 Uma imagem com texto, gráficos de vetor, dispositivo, ventoinha

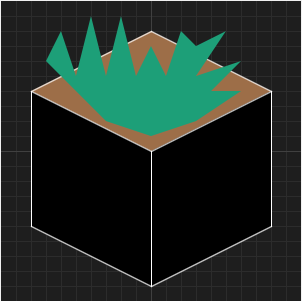
Descrição gerada automaticamente 

Figura 1 – Agentes planta-relva (à esquerda), toupeira (ao centro) e “restocker” de relva (à direita)

Na figura 2, é possível visualizar o agricultor a plantar relva, quando selecionado o modo de plantação “Para a frente”.

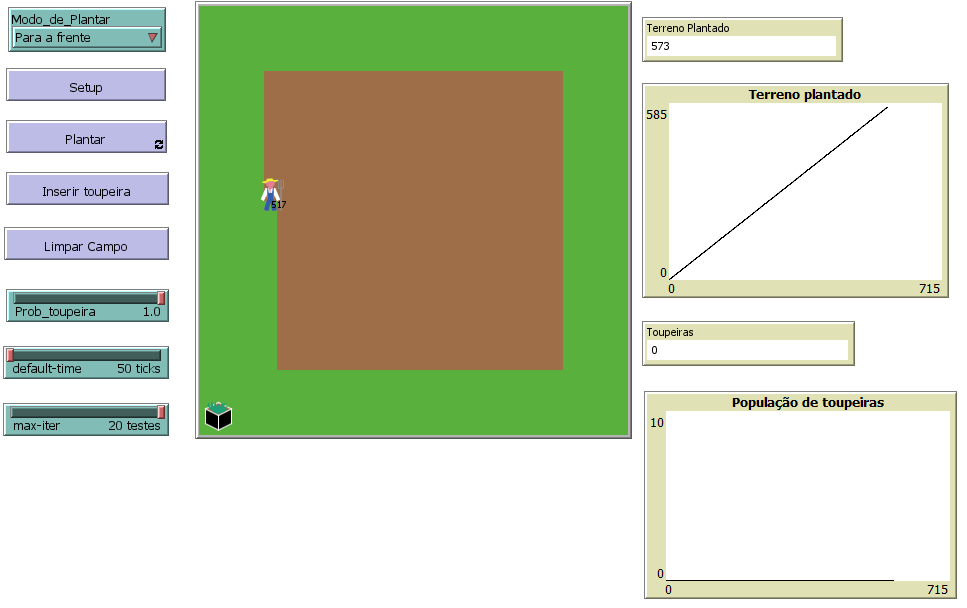


Figura 2 – Modo de plantação “Para a frente”

Já na figura 3, podemos observar o agricultor a plantar relva de acordo com o outro modo de plantação disponível, “Serpentina”.

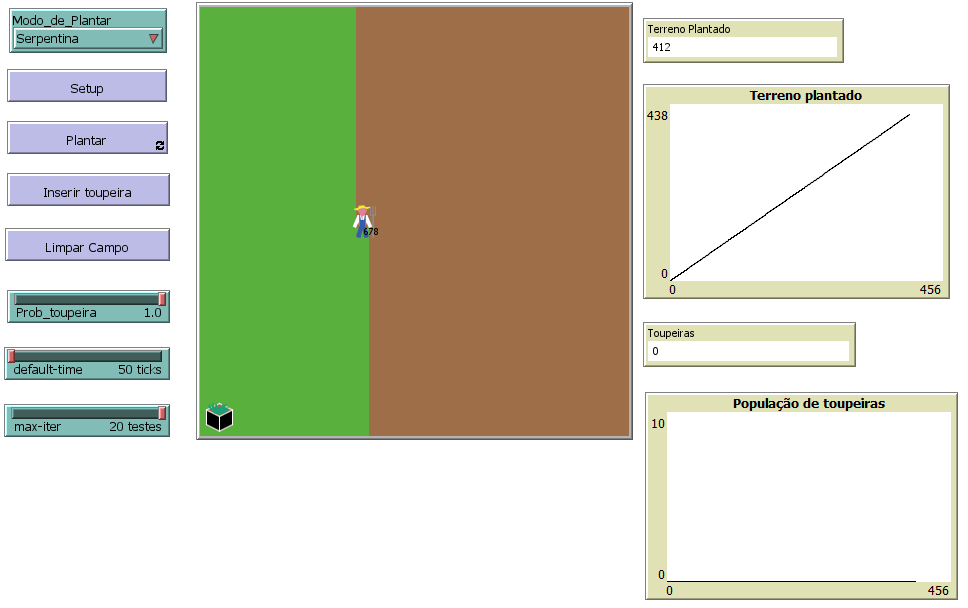


Figura 3 – Modo de plantação “Serpentina”

Após terminar a plantação, o planta-relva repõe o stock e entra em modo de manutenção, algo que podemos ver na figura 4.1. Acrescenta-se que, para 1.º teste, optou-se por manter as condições iniciais nos valores predefinidos.

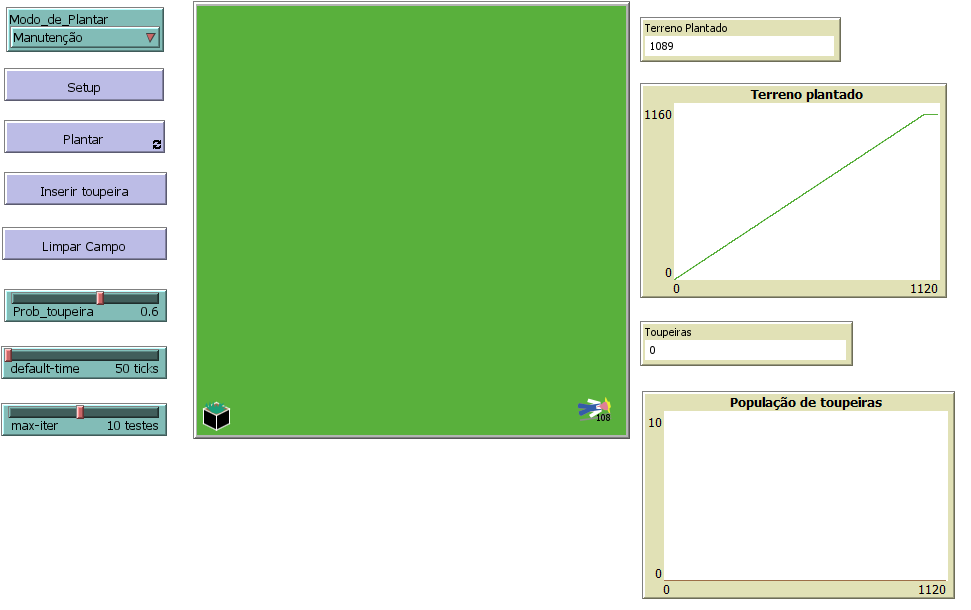


Figura 4.1 – Modo de “Manutenção”

Já em manutenção, o utilizador optou por inserir 3 toupeiras, sendo possível visualizar na figura 4.2 as mesmas (e a sua vida) e o estado do campo, passados alguns ticks.

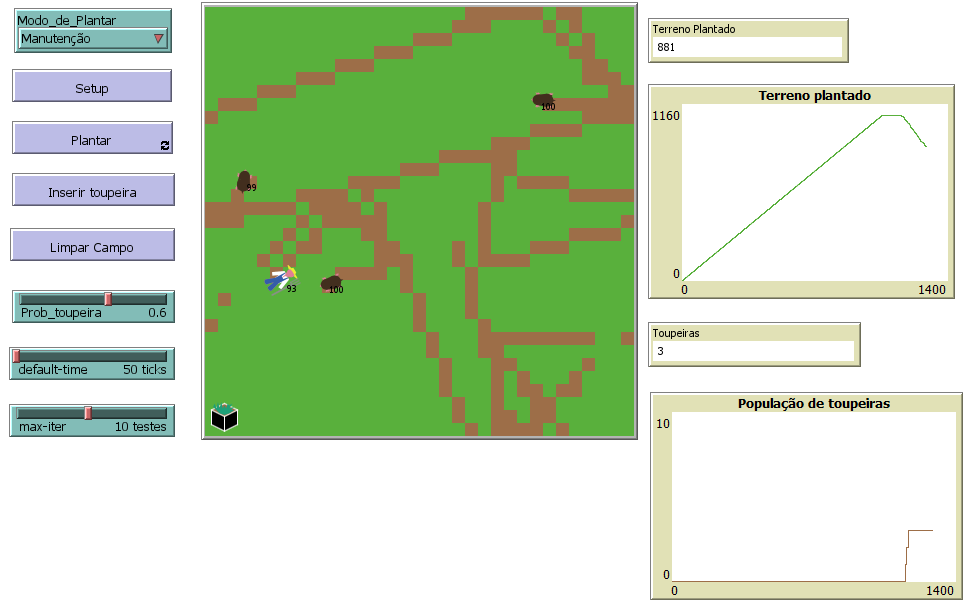


Figura 4.2 – Estado do campo após várias inserções de toupeiras

Com o passar do tempo, a população de toupeiras cresce rapidamente devido ao ritmo de reprodução das mesmas, assim como a porção de relva destruída. Isto é evidente na figura 4.3, onde também se pode ver o planta-relva prestes a matar uma toupeira.

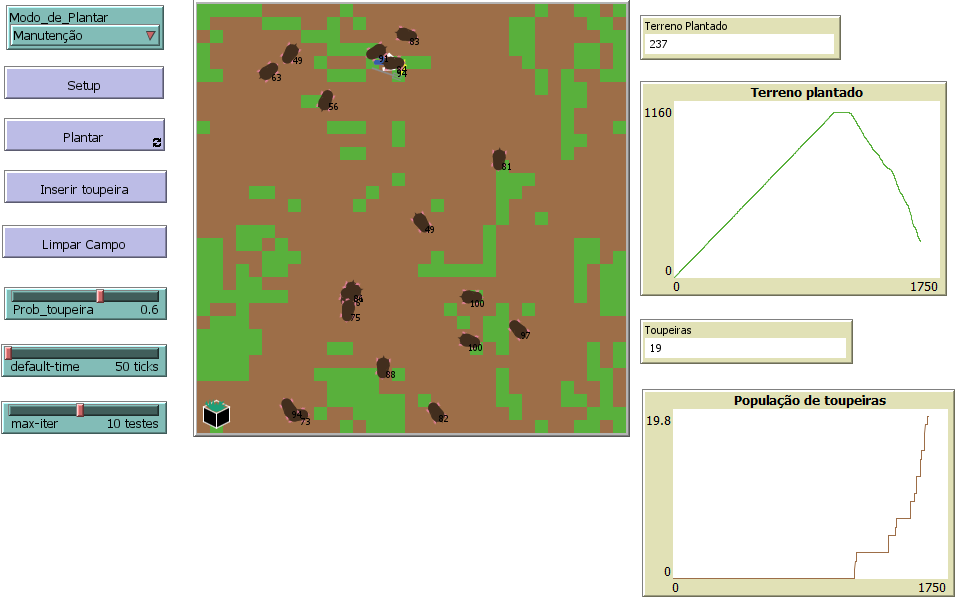


Figura 4.3 –Estado do campo após a reprodução das toupeiras

Como o agricultor não tem capacidade para manter o ritmo de plantação, as toupeiras veem-se privadas da sua fonte de sobrevivência, perdendo gradualmente vida e acabando por morrer, deixando um campo completamente destruído.

Uma imagem com mapa

Descrição gerada automaticamente

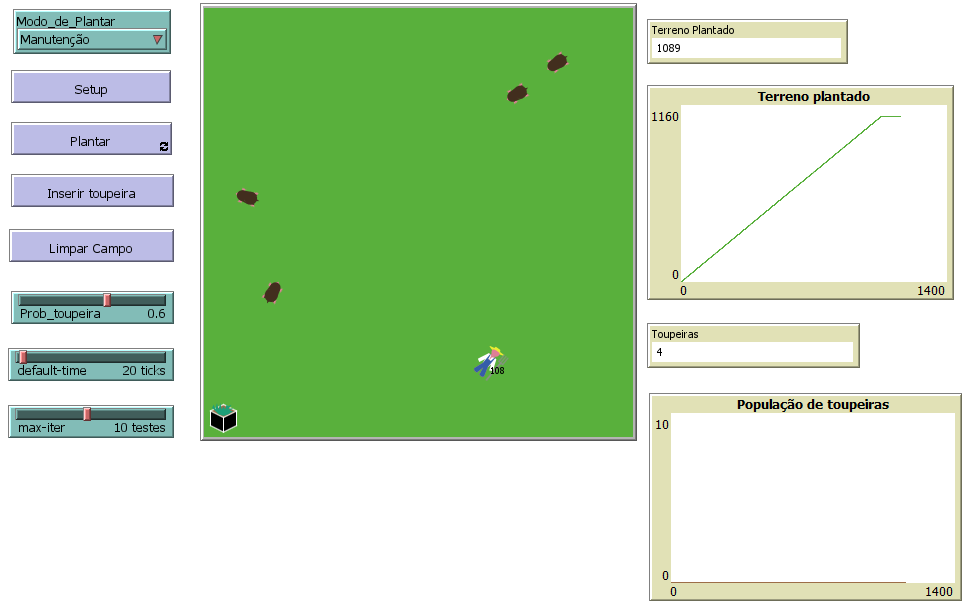
Figura 4.4 – Estado do campo após mortes sucessivas de toupeiras

Eventualmente, o planta-relva recupera o controlo do seu campo, conseguindo matar as últimas toupeiras, prosseguindo então com a plantação. Podemos ver na figura 4.5 o mesmo a deslocar-se ao “restocker” para repor o seu stock.



Figura 4.5 – Estado final do campo

No 2.º teste, optou-se por alterar o tempo de espera entre reproduções para 20 ticks, de modo a analisar a evolução do sistema, quer ao nível da população de toupeiras, quer ao nível do campo em si. Na figura abaixo verifica-se uma população inicial de 4.

 Figura 5.1 – Estado inicial do campo (menor tempo de espera)

Dado o curto intervalo de tempo entre reproduções, passado algum tempo a população de toupeiras atinge um pico. Contudo, isto reflete-se na pouca relva plantada disponível.

Uma imagem com mapa

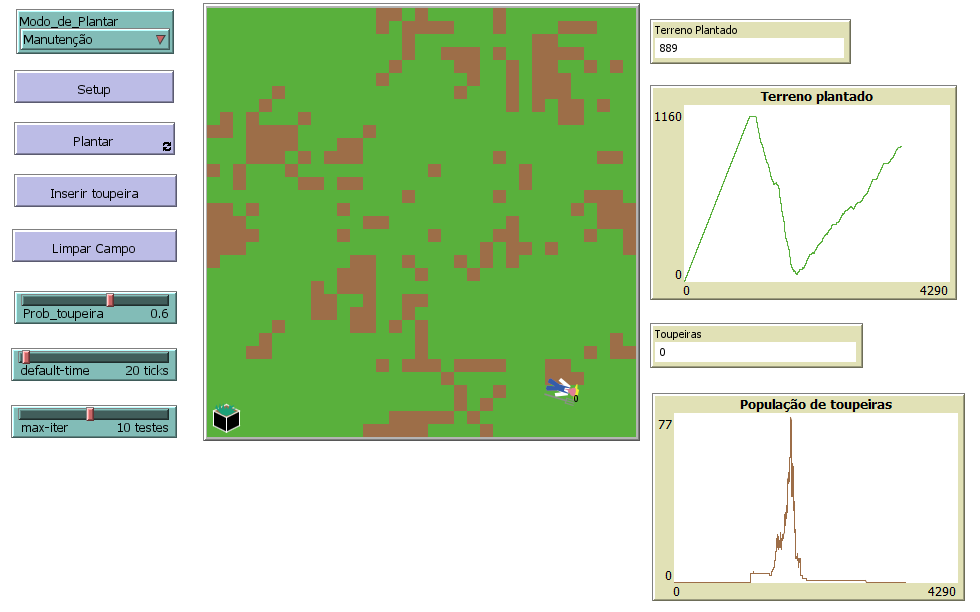
Descrição gerada automaticamente Figura 5.2 – Estado do campo após inúmeras reproduções de toupeiras

Como seria expetável, a falta de relva leva a população de toupeiras a rapidamente decrescer, devido a sucessivas mortes, restando apenas uma, passado algum tempo.

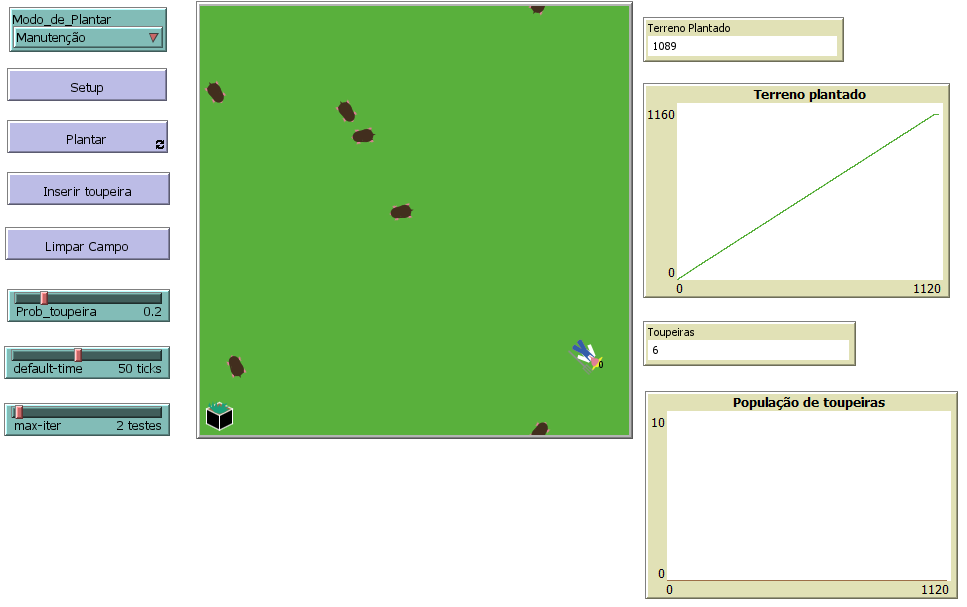
Uma imagem com mapa

Descrição gerada automaticamente Figura 5.3 – Estado do campo após algum tempo, maioria das toupeiras morreram

Com a população controlada, o planta-relva consegue replantar o campo a um ritmo sustentável, como podemos verificar na figura 5.4.

 Figura 5.4 – Estado final do campo

No 3.º teste, optou-se por reduzir a probabilidade da toupeira, responsável pela sua atividade, bem como o número de testes (*max-iter*) de ambos os agentes (fig. 6.1).

 Figura 6.1 – Estado inicial do campo (menor probabilidade)

Por um lado, o menor número de testes resulta em decisões mais aleatórias e/ou erráticas por parte quer do planta-relva, quer das toupeiras. Por outro, a menor probabilidade condiciona a atividade das últimas, quer em movimentos, quer em reprodução.

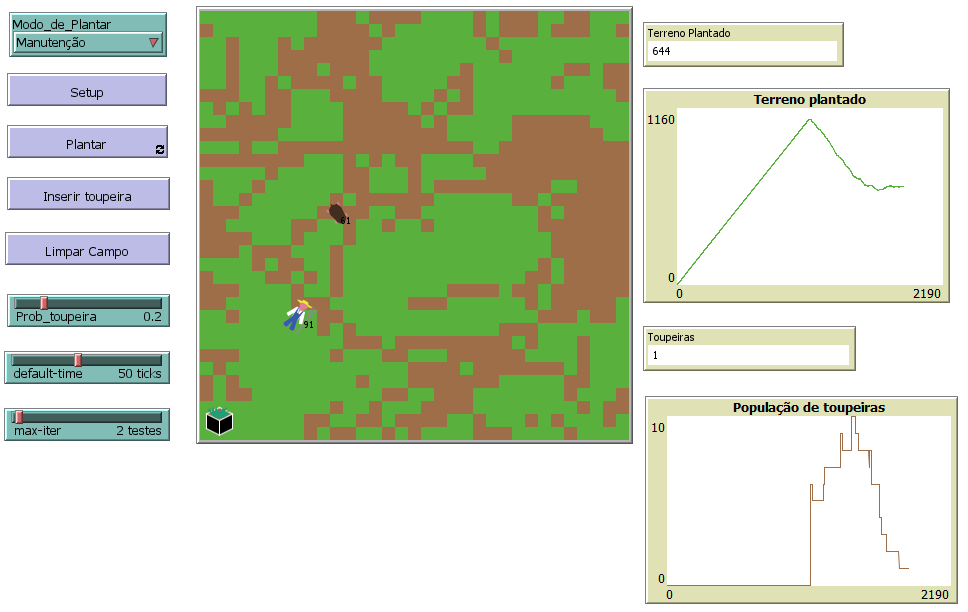


Figura 6.2 – Estado do campo após algum tempo

Passado algum tempo, a maioria, senão todas as toupeiras, morreram, permitindo ao agricultor recuperar a plantação do campo, ainda que este tenha sido pouco afetado.

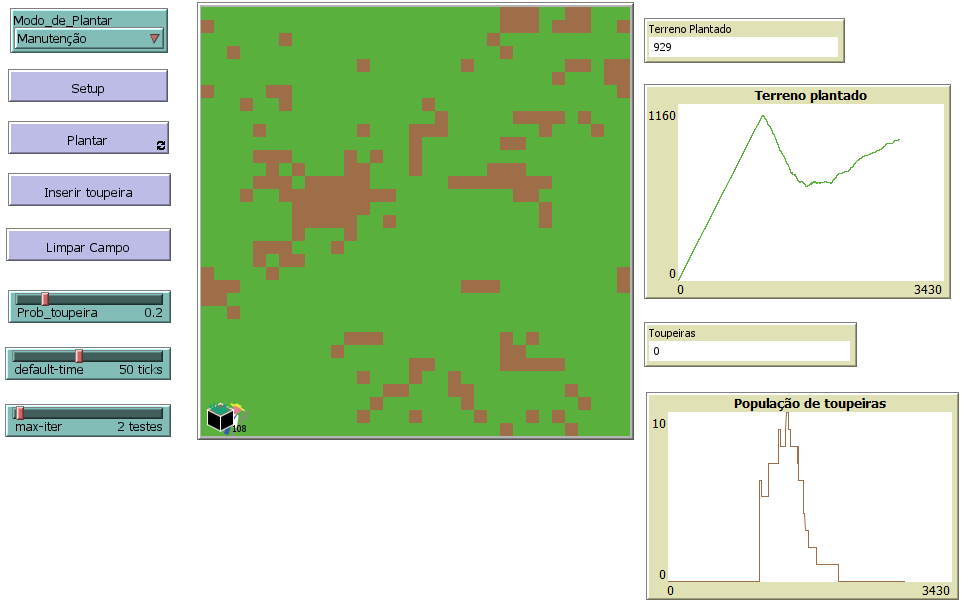


Figura 6.3 – fim

1. NOTAS FINAIS

Concluído o presente trabalho prático, estamos deveras satisfeitos com o mesmo, tendo a referir que surgiram algumas dificuldades quanto á melhor forma de definir o algoritmo. Isto, porque grande parte do funcionamento do sistema foi deixado ao nosso critério, levando a melhorias executadas para além do definido em protocolo, melhorias essas que desencadearam uma série de novas ideias a aplicar, no entanto, decidimos focar-nos na base resultante do trabalho já desenvolvido.

Em anexo está o código relativo ao respetivo trabalho prático, bem como uma implementação alternativa ao mundo aberto (toro), num mundo fechado (caixa), que entendemos adicionar ao trabalho.

# BIBLIOGRAFIA

|  |  |
| --- | --- |
| [1] | S. J. Russel e P. Norvig, “2 Intelligent Agents,” em *Artificial Intelligence: A Modern Approach, 3rd ed.*, Upper Saddle River, New Jersey 07458, Pearson Education. Inc., 2010, pp. 34-63. |
| [2] | J. P. B. M. Oliveira, *Agentes Inteligentes,* Universidade de Trás-os-Montes e Alto Douro, 2021. |

ANEXO A – Código de TP1\_torus.nlogo

toupeiras-own [elapsed-time health]  
planta-relvas-own [stock]  
breed [planta-relvas planta-relva]  
breed [toupeiras toupeira]  
  
to setup  
 clean\_field  
 create-planta-relvas 1 [  
 if Modo\_de\_Plantar = "Serpentina" [ set heading 0 ]  
 if Modo\_de\_Plantar = "Para a frente" [ set heading one-of [0 90 180 270] ]  
 set size 2.5  
 set shape "agricultor"  
 set stock world-width \* world-height ; o necessário para plantar o campo todo  
 ]  
  
 create-turtles 1 [  
 setxy 1 1  
 set size 2.5  
 set shape "grass-restocker"  
 ]  
  
 reset-ticks  
end  
  
to plantar  
 ask planta-relvas [  
 set label stock  
 set label-color black  
  
 if Modo\_de\_Plantar = "Para a frente" [  
 if not (any? patches with [pcolor = brown]) [  
 set Modo\_de\_Plantar "Manutenção"  
 stop  
 ]  
  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
 if [pcolor] of patch-ahead 1 = green [ right 90 ]  
 forward 1  
 ]  
  
 if Modo\_de\_Plantar = "Serpentina" [  
 if not (any? patches with [pcolor = brown]) [  
 set Modo\_de\_Plantar "Manutenção"  
 stop  
 ]  
  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
  
 if xcor = max-pxcor and ycor = max-pycor [ stop ]  
  
 if heading = 0 and ycor = max-pycor [  
 right 90  
 forward 1  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
 right 90  
 ]  
  
 if heading = 180 and ycor = 0 [  
 left 90  
 forward 1  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
 left 90  
 ]  
  
 forward 1  
 ]  
  
 if Modo\_de\_Plantar = "Manutenção" [  
 ; atividade das toupeiras  
 ask toupeiras [  
 set label health  
 set label-color black  
 set elapsed-time elapsed-time + 1  
  
 if any? planta-relvas-here [ die ]  
  
 ifelse [pcolor] of patch-here = green [  
 ask patch-here [ set pcolor brown ]  
  
 ifelse health <= 90 [ set health health + 5 ]  
 [ set health 100 ]  
 ]  
 [ set health health - 1 ]  
  
 if health = 0 [ die ]  
  
 if prob Prob\_toupeira [  
 if count toupeiras-here = 2 and elapsed-time >= default-time [  
 hatch 1 [  
 set heading random 360  
 set elapsed-time 0  
 ]  
 set elapsed-time 0  
 ]  
  
 let it 0  
 while [it < max-iter and not(any? toupeiras-on patch-ahead 1)  
 and [pcolor] of patch-ahead 1 = brown] [  
 set it it + 1  
 set heading random 360  
 ]  
 forward 1  
 ]  
 ]  
 ; atividade do planta-relva  
  
 ask toupeiras-here [ die ]  
  
 ifelse stock = 0 [  
 facexy 1 1  
 forward 1  
 if count turtles-here with [shape = "grass-restocker"] = 1  
 [ set stock int (world-width \* world-height / 3) ]  
 ]  
 [ if [pcolor] of patch-here = brown [  
 set pcolor green  
 set stock stock - 1  
 ]  
  
 let it2 0  
 while [it2 < max-iter and not(any? toupeiras-on patch-ahead 1)  
 and [pcolor] of patch-ahead 1 = green] [  
 set it2 it2 + 1  
 set heading random 360  
 ]  
 forward 1  
 ]  
 ]  
 ]  
 tick  
end  
  
to insere\_toupeira  
 create-toupeiras 1 [  
 setxy random-xcor random-ycor  
 set heading random 360  
 set size 2  
 set shape "toupeira"  
 set color 32  
 set elapsed-time default-time  
 set health 100  
 ]  
end  
  
to-report prob[x]  
 report (random-float 1 < x)  
end  
  
to clean\_field  
 clear-all  
 ask patches [ set pcolor brown ]  
end

ANEXO B – Código de TP1\_box.nlogo

toupeiras-own [elapsed-time health]  
planta-relvas-own [stock]  
breed [planta-relvas planta-relva]  
breed [toupeiras toupeira]  
  
to setup  
 clean\_field  
 create-planta-relvas 1 [  
 if Modo\_de\_Plantar = "Serpentina" [ set heading 0 ]  
 if Modo\_de\_Plantar = "Para a frente" [ set heading one-of [0 90 180 270] ]  
 set size 2.5  
 set shape "agricultor"  
 set stock world-width \* world-height ; o necessário para plantar o campo todo  
 ]  
  
 create-turtles 1 [  
 setxy 1 1  
 set size 2.5  
 set shape "grass-restocker"  
 ]  
  
 reset-ticks  
end  
  
to plantar  
 ask planta-relvas [  
 set label stock  
 set label-color black  
  
 if Modo\_de\_Plantar = "Para a frente" [  
 if not (any? patches with [pcolor = brown]) [  
 set Modo\_de\_Plantar "Manutenção"  
 stop  
 ]  
  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
 if [pcolor] of patch-ahead 1 = green [ right 90 ]  
 forward 1  
 ]  
  
 if Modo\_de\_Plantar = "Serpentina" [  
 if not (any? patches with [pcolor = brown]) [  
 set Modo\_de\_Plantar "Manutenção"  
 stop  
 ]  
  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
  
 if xcor = max-pxcor and ycor = max-pycor [ stop ]  
  
 if heading = 0 and ycor = max-pycor [  
 right 90  
 forward 1  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
 right 90  
 ]  
  
 if heading = 180 and ycor = 0 [  
 left 90  
 forward 1  
 ask patch-here [ set pcolor green ]  
 set stock stock - 1  
 left 90  
 ]  
  
 forward 1  
 ]  
  
 if Modo\_de\_Plantar = "Manutenção" [  
 ; atividade das toupeiras  
 ask toupeiras [  
 set label health  
 set label-color black  
 set elapsed-time elapsed-time + 1  
  
 if any? planta-relvas-here [ die ]  
  
 ifelse [pcolor] of patch-here = green [  
 ask patch-here [ set pcolor brown ]  
  
 ifelse health <= 90 [ set health health + 5 ]  
 [ set health 100 ]  
 ]  
 [ set health health - 1 ]  
  
 if health = 0 [ die ]  
  
 if prob Prob\_toupeira [  
 if count toupeiras-here = 2 and elapsed-time >= default-time [  
 hatch 1 [  
 set heading random 360  
 set elapsed-time 0  
 ]  
 set elapsed-time 0  
 ]  
  
 if bounce = false [  
 let it 0  
 while [it < max-iter and not(any? toupeiras-on patch-ahead 1)  
 and [pcolor] of patch-ahead 1 = brown] [  
 set it it + 1  
 set heading random 360  
 ]  
 ]  
 forward 1  
 ]  
 ]  
 ; atividade do planta-relva  
  
 ask toupeiras-here [ die ]  
  
 ifelse stock = 0 [  
 facexy 1 1  
 forward 1  
 if count turtles-here with [shape = "grass-restocker"] = 1  
 [ set stock int (world-width \* world-height / 3) ]  
 ]  
 [ if [pcolor] of patch-here = brown [  
 set pcolor green  
 set stock stock - 1  
 ]  
  
 if bounce = false [  
 let it2 0  
 while [it2 < max-iter and not(any? toupeiras-on patch-ahead 1)  
 and [pcolor] of patch-ahead 1 = green] [  
 set it2 it2 + 1  
 set heading random 360  
 ]  
 ]  
 forward 1  
 ]  
 ]  
 ]  
 tick  
end  
  
to-report bounce ; function to bounce when agent hits walls  
 let flag false  
  
 ; bounce off left wall  
 if pxcor = 0 [  
 set heading random 180  
 set flag true  
]  
 ; bounce off right wall  
 if pxcor = max-pxcor [  
 set heading random 180 + 180  
 set flag true  
 ]  
 ; bounce off bottom wall  
 if pycor = 0 [  
 set heading random 180 - 90  
 set flag true  
 ]  
 ; bounce off top wall  
 if pycor = max-pycor [  
 set heading random 180 + 90  
 set flag true  
 ]  
  
 report flag  
end

to insere\_toupeira  
 create-toupeiras 1 [  
 setxy random-xcor random-ycor  
 set heading random 360  
 set size 2  
 set shape "toupeira"  
 set color 32  
 set elapsed-time default-time  
 set health 100  
 ]  
end  
  
to-report prob[x]  
 report (random-float 1 < x)  
end

to clean\_field  
 clear-all  
 ask patches [ set pcolor brown ]  
end