Trabalho Prático 1: Agentes

Licenciatura em Engenharia Informática

Universidade de Trás-os-Montes e Alto Douro

Inteligência Artificial

Grupo 13

Paulo Moura Oliveira

Autores:

Miguel Teixeira – al78321

Rui Madureira – al78282

Índice

1. Introdução

A utilização de robôs para a limpeza de superfícies tem vindo a adquirir uma relevância crescente, sendo este o tema central do nosso Trabalho Prático 1. Nesse contexto, reconhecemos que os sistemas multiagente apresentam-se como ferramentas valiosas, pois possibilitam a simulação de interações complexas entre diversos agentes, incluindo aqueles responsáveis pela limpeza e os que geram poluição. Neste trabalho, o nosso objetivo é modelar e simular um ambiente de limpeza utilizando a plataforma NetLogo, com foco na implementação de um modelo inicial denominado Robot1. É aqui que iremos explorar a dinâmica entre um agente de limpeza ('Cleaner') e os agentes poluidores ('Polluters').

Além disso, avançaremos para a implementação de um segundo modelo, Robot2, que introduzirá um maior grau de complexidade através das nossas próprias ideias, sendo estas inovações ou variantes. Esta etapa permitir-nos-á investigar novas abordagens e refletir sobre o equilíbrio necessário em sistemas ecológicos digitais, enriquecendo assim a nossa experiência na área da Engenharia Informática.

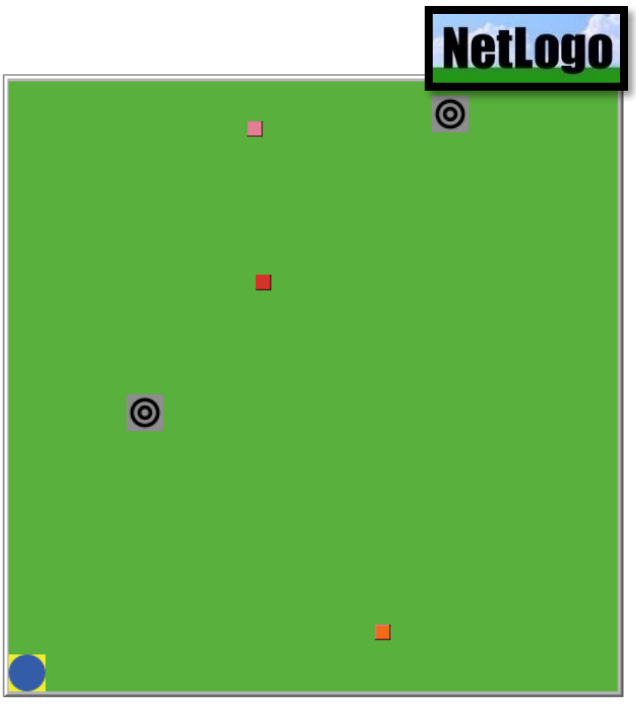

Figura 1-Imagem de apresentação

2. Desenvolvimento

1ª Fase da implementação

1- Variáveis globais

```
globals [
  pollution-level              ; nível de poluição
  max-cleaner-debris           ; capacidade máxima de detritos que o Cle
  cleaner-energy-blue          ; energia do Cleaner azul
  current-debris-count-blue    ; contagem de detritos que o Cleaner azul
  cleaner-energy-black         ; energia do Cleaner preto
  current-debris-count-black   ; contagem de detritos que o Cleaner pret
  total-debris-count           ; contagem total de detritos que o cleaner
  tick-count                   ; contador de ticks

]
```

Figura 2- Variáveis globais

2- Variáveis específicas

```
turtles-own [
  role                 ; papel do agente (Cleaner ou Polluter)
  debris-type          ; tipo de detrito que o Polluter pode depositar
  cleaner-type         ; tipo de cleaner (azul ou preto)
  charger-assigned     ; o charger atribuído a cada Cleaner

]
```

Figura 3- Variáveis específicas

As funções acima têm um papel importante no desenrolar do nosso trabalho uma vez que é aqui que definimos as variáveis globais e específicas que iremos usar ao longo do trabalho tal como o contador do total de ticks e total de detritos. Também é aqui que declaramos variáveis para distinguir o diferente tipo de papeis para as turtles desempenharem ao longo do projeto.

3- Setup

```
; Criação de Agentes (Cleaner e Polluters)
to setup
  clear-all
  ask patches [
    set pcolor green    ; todas as células começam limpas (verde)
  ]
  set pollution-level 0                    ; nivel de poluição inicial
  set cleaner-energy-blue  energy               ; energia inicial do Cleaner é o valor do deslizador
  set max-cleaner-debris  num-cleaner-cap     ; capacidade máxima de detritos
  set current-debris-count-blue  0              ; inicializa a contagem de detritos
  set tick-count 0
```

Figura 4- Função to setup I

Nesta função, definimos a cor do mundo como verde e inicializamos as variáveis que serão utilizadas como contadores.

```
; Primeiro cria o Trash (lixeira)
 create-turtles num_trash [              ; vai criar o número do trashes que tiver no deslizador
  set color grey
  set role "Trash"
  set shape "target"
  setxy random-xcor random-ycor
  limit                                  ; Garante inicializam dentro dos limites
  safe-spawn                             ; Garante que não nasçam sobre o Charger ou outros Trash
  set size 2
  limit
]
; Cria o Charger
create-turtles 1 [
  set color yellow         ; Charger
  set role "Charger"
  set shape "lightning"  ; Usar a forma personalizada com o raio amarelo
  setxy -15.5 -15.5          ; Posição inicial do Charger
  set size 2              ; Define o tamanho
]
```

Figura 6- Função to setup II

Figura 5- Charger

Figura 7- Trash

É aqui que criamos as turtles que serão utilizadas ao longo do código. A função presente na Figura 6, mostra como criamos o 'Charger' (Figura 5) e os 'Trash' (Figura 7). Sendo que para estes últimos atribuímos parâmetros como a quantidade (regulada pelo deslizador 'num_trash'), cor, tamanho e localização inicial (aleatória). Chamamos também as funções "limit" e "safe-spawn" que iremos abordar mais à frente neste trabalho. Fizemos o mesmo para a criação dos 'Chargers', embora com uma localização específica no mapa.

```
; Cria o Cleaner
create-turtles 1 [
  set color blue         ; Cleaner
  set role "Cleaner"
  set shape "circle"     ; Define a forma do Cleaner como círculo
  setxy -15.5 -15.5          ; Posição inicial do Cleaner
  set size 2
]

; Cria o 1º  Polluters, cada um com um tipo de detrito exclusivo
  set color red    ; Polluter 1
  set role "Polluter"
  set debris-type 0  ; Tipo 0 de resíduo (vermelho)
  set shape "square_red"
  safe-spawn
  setxy random-xcor random-ycor
  limit

]
```

Figura 8- Função to setup III

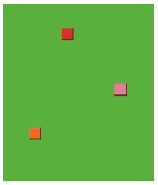

Figura 9- Polluters

Figura 10- Cleaner

Na Figura 8, é possível observar como criamos o 'Cleaner' de cor azul (Figura 10). A sua localização inicial é no respetivo 'Charger' e de seguida, criámos mais três turtles, que são os três 'Polluters' (vermelho, laranja e rosa, Figura 9).

4- Funções de movimentos

```
to go_once
  ask turtles [
    move
    if role = "Polluter" [
      deposit-debris
    ]
    if role = "Cleaner" [
      clean
    ]
  ]
  set tick-count tick-count + 1

  ; Atualiza os monitores após cada tick
  update-monitors
end

to go_n
  let n n-movimentos ; pega o valor do deslizador
  repeat n [
    go_once
  ]
end


to go
  while [cleaner-energy-blue > 0] [
    go_once
  ]
  stop
end
```

Figura 11- Funções go_once, go_n e go

Figura 12- Botão Go

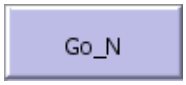

Figura 13- Botão Go_N

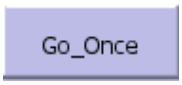

Figura 14- Botão Go_Once

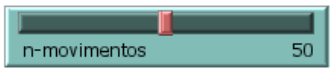

Figura 15- Deslizador n-movimentos

Aqui, na Figura 11, estão implementadas as funções que controlam o movimento das turtles no mapa. A função “go_once”, acionada pelo botão ‘Go_once’ (Figura 14), permite visualizar o deslocamento passo a passo. Por outro lado, a função “go_n”, através do botão ‘Go_N’ (Figura 13), executa o número de movimentos definidos pelo utilizador através do deslizador n-movimentos (Figura 15). Por fim, a função “go”, iniciada com o botão ‘Go’ (Figura 12), mantém-se em funcionamento contínuo enquanto a bateria do ‘Cleaner’ for positiva.

5- Movimento das turtles

```
to move
  let trash one-of turtles with [role = "Trash"]
  let charger one-of turtles with [role = "Charger"]
; Para o Polluter
if role = "Polluter" [
  ; Evita o Trash e o Charger
  let heading-away-from-trash  (ifelse-value (distance trash < 2)
    [subtract-headings towards trash 180] [heading])
  let heading-away-from-charger  (ifelse-value (distance charger < 2)
    [subtract-headings towards charger 180] [heading-away-from-trash ])
  set heading heading-away-from-charger
  move-in-four-directions
]
```

Figura 16- Função to move

Esta função "move" (Figura 16) faz com que as turtles, dependendo do seu papel (role), comportem-se de forma específica. Se a turtle tiver o papel de 'Polluter', ela evita turtles com o papel de 'Trash' e de 'Charger'. Além disso, ajusta a sua direção para se afastar delas caso estejam a menos de 2 unidades de distância.

```
  ; Para o Cleaner
  if role = "Cleaner" [
    if current-debris-count-blue  = max-cleaner-debris  [
      if cleaner-energy-blue  >= 45 [
        move-to-trash
      ]
      if cleaner-energy-blue  < 45 [
        if distance trash < 5 [
          move-to-trash
          if current-debris-count-blue  = 0 [
            move-to-charger
            if distance charger < 1 [recharge]
          ]
        ]
        if distance trash >= 5 [
          move-to-charger
          if distance charger < 1 [recharge]
        ]
      ]
    ]

    if current-debris-count-blue  < max-cleaner-debris  [
       if cleaner-energy-blue  >= 30 [
          scan-and-collect
          move-in-four-directions
      ]
      if cleaner-energy-blue  < 30 [
        move-to-charger
        if distance charger < 0.5 [recharge]
      ]
    ]
  ]
end
```

Figura 17- Função to move II

Se o papel da turtle for 'Cleaner' e a quantidade de detritos que carrega atingir a capacidade máxima, o comportamento depende da energia disponível. Se a energia for igual ou superior a 45, o 'Cleaner' move-se diretamente para o 'Trash' mais próximo para descarregar. Porém, se a energia for inferior a 45, verifica se está a menos de 5 unidades do 'Trash' mais perto; se estiver, move-se até lá e, após descarregar, dirige-se ao 'Charger' para recarregar. Por outro lado, se estiver a mais de 5 unidades do 'Trash' mais próximo , move-se diretamente para o 'Charger' e recarrega a bateria. Dirigindo-se longo de imediato ao 'Trash' mais perto para descarregar os detritos.

Em contrapartida, se a capacidade máxima de detritos não for atingida, e a energia for igual ou superior a 30, o 'Cleaner' procura detritos para recolher e move-se em quatro direções. Se a energia for inferior a 30, a prioridade é recarregar, movendo-se diretamente para o 'Charger'.

5.1- Mover para o Trash

```
to move-to-trash
  let nearest-trash min-one-of turtles with [role = "Trash"] [distance myself]
  ; Encontra o Trash mais próximo

  if abs (xcor - [xcor] of nearest-trash ) > abs (ycor - [ycor] of nearest-trash ) [
    if xcor < [xcor] of nearest-trash [
      set heading 90  ; Move para a direita
    ]
    if xcor > [xcor] of nearest-trash [
      set heading 270  ; Move para a esquerda
    ]
  ]
  if abs (xcor - [xcor] of nearest-trash ) <= abs (ycor - [ycor] of nearest-trash ) [
    if ycor < [ycor] of nearest-trash [
      set heading 0  ; Move para cima
    ]
    if ycor > [ycor] of nearest-trash [
      set heading 180  ; Move para baixo
    ]
   ; Mova-se um passo na direção escolhida
   fd 1  ; Move um passo na direção escolhida

   ; Verifique se está próximo do Trash para esvaziar a carga
   if distance nearest-trash < 0.7 [
     set current-debris-count-blue 0  ; Redefine a contagem de detritos
     set pollution-level pollution-level - max-cleaner-debris
     scan-and-collect  ; Coleta lixo imediatamente após despejar
     wait 1
     if cleaner-energy-blue < 45[
       move-to-charger
     ]
   ]
 end
```

Figura 18- Função move to trash

Esta função (Figura 18) faz com que o 'Cleaner' mova-se em direção ao lixo mais próximo. Primeiro, identifica o 'Trash' mais próximo com recurso à função min-one-of. Em seguida, verifica se a diferença nas coordenadas x é maior do que nas coordenadas y. Se for, ajusta a direção para a direita ou esquerda, dependendo da posição relativa ao 'Trash'. Se a diferença nas coordenadas y for maior ou igual, ajusta a direção para cima ou para baixo. Após definir a direção, o 'Cleaner' avança um passo nessa direção.

Depois, verifica se está a menos de 0,7 unidades do 'Trash'. Se estiver, redefine a contagem de detritos para zero, reduz o nível de poluição e reexecuta a função scan-and-collect para apanhar os detritos . Caso a energia do 'Cleaner' seja inferior a 45, ele dirige-se ao 'Charger' para recarregar e só depois é que vai depositar os detritos recolhidos.

5.1.1- Depositar os detritos

```
;Depósito de resíduos apanhados pelo cleaner
to deposit-debris
  if debris-type = 0 [
    if random-float 1 < red-polluter-probability  and pcolor != red [
      set pcolor red  ; muda a cor da célula para vermelho (tipo 0)
      set pollution-level pollution-level + 1
    ]
  ]
  if debris-type = 1 [
    if random-float 1 < pink-polluter-probability  and pcolor != pink [
      set pcolor pink  ; muda a cor da célula para amarelo (tipo 1)
      set pollution-level pollution-level + 1
    ]
  ]
  if debris-type = 2 [
    if random-float 1 < orange-polluter-probability  and pcolor != orange [
      set pcolor orange  ; muda a cor da célula para laranja (tipo 2)

      set pollution-level pollution-level + 1
    ]
  ]
end
```

Figura 19- Função deposit-debries

A função “deposit-debris” (Figura 19) lida com a deposição de detritos com base no seu tipo. Se o tipo de detrito for 0, verifica se um número aleatório é inferior à probabilidade de poluição vermelha (red-polluter-probability) e se a célula não se encontra já vermelha. Se ambas as condições forem verdadeiras, a cor da célula muda para vermelho e o nível de poluição aumenta um. E o mesmo para os outros tipos de detritos, rosa e laranja.

5.2- Mover para o Charger

```
to move-to-charger
  let charger one-of turtles with [role = "Charger"]  ; Identifica o Charger

  if abs (xcor - [xcor] of charger) > abs (ycor - [ycor] of charger) [
    if xcor < [xcor] of charger [
      set heading 90  ; Move para a direita
    ]
    if xcor > [xcor] of charger [
      set heading 270  ; Move para a esquerda
    ]
  ]

  if abs (xcor - [xcor] of charger) <= abs (ycor - [ycor] of charger) [
    if ycor < [ycor] of charger [
      set heading 0  ; Move para cima
    ]
    if ycor > [ycor] of charger [
      set heading 180  ; Move para baixo
    ]
  ]

  fd 1  ; Move um passo na direção escolhida
end
```

Figura 20- Função move-to-charger

Esta função “move-to-charger” (Figura 20) faz com que o 'Cleaner' dirija-se ao respetivo ‘Charger’. Primeiro, identifica um carregador com recurso da função one-of para selecionar uma turtle com o papel de ‘Charger’. Em seguida, verifica se a diferença nas coordenadas x é maior do que nas coordenadas y. Se for, ajusta a direção para a direita ou esquerda, conforme a posição do carregador. Se a diferença nas coordenadas y for maior ou igual, ajusta a direção para cima ou para baixo, exatamente o mesmo mecanismo utilizado na “move-to-trash”, (Figura 18). Depois de definir a direção, o 'Cleaner' avança um passo nessa direção. Isto para fazer sempre o trajeto mais curto em direção ao ‘Charger’.

5.2.1- Carregar Bateria

```
to recharge
  ; O Cleaner vai carregar se a energia estiver abaixo de 30
  ; OU se a carga estiver cheia e a energia for menor que 45
  if cleaner-energy-blue  < 45 or (current-debris-count-blue  = max-cleaner-debris  and cleaner-energy-blue  < 45) [
    ; Mantém o Cleaner no carregamento até que a energia atinja 100
    while [cleaner-energy-blue  < 100] [
      set cleaner-energy-blue  cleaner-energy-blue  + charging-rate  ; Aumenta a energia conforme a taxa de carregamento
      if cleaner-energy-blue  > 100 [
        set cleaner-energy-blue  100  ; Garante que a energia não ultrapasse 100
      ]
      update-monitors  ; Atualiza os monitores durante o carregamento

      ; Os Polluters continuam se movendo
      ask turtles with [role = "Polluter"] [
        move  ; Chama a função move dos Polluters
      ]

      wait temp-carregamento  ; Usa o valor do deslizador para aguardar o tempo de carregamento
    ]
  ]
end
```

Figura 21- Função recharge

A função “recharge” (Figura 21) é responsável pelo carregamento da energia do 'Cleaner' sob certas condições. Ele começa a carregar se a energia estiver abaixo de 30 ou se a carga estiver cheia e a energia for inferior a 45. Enquanto a energia for inferior a 100, o 'Cleaner' mantém-se a carregar, aumentando a energia de acordo com a taxa de carregamento definida no deslizador charging-rate, (Figura 22). Se a energia ultrapassar 100, ela mesma é ajustada para esse valor máximo.

Durante o processo de carregamento, os monitores são atualizados para refletir o estado atual. Além disso, as turtles com o papel de "Polluter" continuam a mover-se, chamando a função de movimento correspondente. O tempo de carregamento é controlado pelo deslizador temp-carregamento, (Figura 22) que determina quanto tempo o 'Cleaner' permanece a carregar antes de verificar novamente a sua energia.

5.2.2- Alterar o tempo de carregamento

Figura 22- Deslizador temp-carregamento

5.3- Fazer Scan de lixos

```
to scan-and-collect
  let radius 1  ; Define o raio para escanear
  let nearby-patches patches in-radius radius  ; Identifica os patches próximos

  ; Verifica se o Cleaner está em um patch que não seja verde
  if pcolor != green [

    ; Varre as células adjacentes e identifica aquelas com detritos
    let debris-patches patches with [pcolor = red or pcolor = pink or pcolor = orange]

    ; Se encontrar detritos ao redor, mover para o patch com mais detritos
    if any? debris-patches [
      let most-debris-patch max-one-of debris-patches [count turtles-here]  ; Identifica o patch com mais detritos
      move-to most-debris-patch  ; Move para o patch com mais detritos
    ]
  ]

  ; Após o movimento, coleta os detritos se estiver sobre uma célula com cor diferente de verde
  ask patches in-radius radius [
    if (pcolor = red or pcolor = pink or pcolor = orange) and (current-debris-count-blue  < max-cleaner-debris ) [
      set pcolor green  ; limpa a célula (volta a ser verde)
      set current-debris-count-blue  current-debris-count-blue  + 1  ; conta os detritos coletados
      set pollution-level  pollution-level  - 1  ; diminui o nível de poluição
      set total-debris-count  total-debris-count  + 1  ; Atualiza a contagem total de detritos
    ]
  ]
end
```

Figura 23- Função scan-and-collect

A função scan-and-collect (Figura 23) permite ao 'Cleaner' localizar e recolher detritos nas células adjacentes. Primeiro, define um raio de uma unidade para analisar os patches próximos. Se o 'Cleaner' não estiver num patch verde, verifica as células adjacentes em busca de detritos, que podem ser vermelhos, rosas ou laranjas.

Caso encontre detritos, move-se para o patch com mais detritos. Depois de se mover, o 'Cleaner' recolhe os detritos se estiver sobre uma célula que não seja verde e se a contagem de detritos atuais for inferior à capacidade máxima. Quando obtém um detrito, a célula é limpa, tornando-se verde novamente, a contagem de detritos armazenados é incrementada, o nível de poluição diminui e a contagem total de detritos é atualizada.

5.5- Capacidade de detritos que o Cleaner pode transportar

Figura 24- Deslizador num-cleaner-cap

```
; Movimentação e ações do cleaner azul
to mover-cleaner-blue [trash charger]
  if current-debris-count-blue = max-cleaner-debris [
    if cleaner-energy-blue >= 45 [
      move-to-trash-blue
    ]
```

Figura 25- Lógica usada na função mover-to-blue

```
set max-cleaner-debris num-cleaner-cap
```

Figura 26- Declaração inicial da capacidade que os detritos podem transportar

O deslizador num-cleaner-cap (Figura 24) intervém ao definir o valor de max-cleaner-debris (Figura 26) que corresponde à capacidade máxima de detritos que o 'Cleaner' pode transportar. Assim, o deslizador permite ajustar dinamicamente essa capacidade. Quando o 'Cleaner' atinge o valor de max-cleaner-debris, valor este que é determinado pelo deslizador, a função verifica se deve mover-se para o local de descarregamento dos detritos, controlando diretamente o comportamento do 'Cleaner' com base no limite configurado pelo utilizador.

Decidimos implementar este mecanismo em vez de um valor fixo, de modo a permitir que o utilizador da plataforma possa interferir na atitude do 'Cleaner' durante a execução do programa.

5.6- Limpar o ambiente

```
to clean
  if (pcolor = red or pcolor = pink or pcolor = orange) and (current-debris-count-blue  < max-cleaner-debris ) [
    set pcolor green  ; limpa a célula (volta a ser verde)
    set current-debris-count-blue  current-debris-count-blue  + 1  ; conta os detritos coletados
    set pollution-level  pollution-level  - 1  ; diminui o nível de poluição
    set total-debris-count  total-debris-count  + 1

  ]
end
```

Figura 27- Função clean

A função “clean” (Figura 27) é responsável por limpar os detritos das células. Se a célula em que o 'Cleaner' se encontra for vermelha, rosa ou laranja, e se a contagem atual de detritos for inferior à capacidade máxima, a função executa as seguintes ações: a célula é limpa, retornando à cor verde; a contagem de detritos coletados é incrementada; o nível de poluição é reduzido em 1; e a contagem total de detritos é atualizada, aumentando em 1.

Implementamos duas funções a nível de limpeza para proporcionar flexibilidade no comportamento do Cleaner. Desta forma, o Cleaner consegue maximizar a sua eficiência ao recolher detritos em uma área mais ampla, contribuindo para um processo de limpeza mais eficaz.

5.7- Mover nas quatro direções

```
to move-in-four-directions
  let possible-directions [0 90 180 270]  ; Direções possíveis
  let chosen-direction one-of possible-directions  ; Escolhe uma direção aleatória

  ; Impedir movimento em direção à parede
  let isAtWall false

  ; Verificar se está em uma parede
  if xcor >= 15.5 [
    if chosen-direction = 90 [ set isAtWall true ] ; Direção para a direita
  ]
  if xcor <= -15.5 [
    if chosen-direction = 270 [ set isAtWall true ] ; Direção para a esquerda
  ]
  if ycor >= 15.5 [
    if chosen-direction = 0 [ set isAtWall true ] ; Direção para cima
  ]
  if ycor <= -15.5 [
    if chosen-direction = 180 [ set isAtWall true ] ; Direção para baixo
  ]

  ; Se a direção aleatória estiver próxima de uma parede, escolher uma nova direção
  while [isAtWall] [
    set chosen-direction one-of possible-directions  ; Escolhe uma nova direção

    ; Reinicializa a verificação
    set isAtWall false
    if xcor >= 15.5 [
      if chosen-direction = 90 [ set isAtWall true ]
    ]
    if xcor <= -15.5 [
      if chosen-direction = 270 [ set isAtWall true ]
    ]
    if ycor >= 15.5 [
      if chosen-direction = 0 [ set isAtWall true ]
    ]
    if ycor <= -15.5 [
      if chosen-direction = 180 [ set isAtWall true ]
    ]
  ]

  ; Define a nova direção e move-se para frente
  set heading chosen-direction  ; Define a nova direção
  fd 1  ; Move-se para frente
end
```

Figura 28- Função move-in-four-directions

A função “move-in-four-directions” (Figura 28) permite que o 'Cleaner' azul se mova em quatro direções possíveis: 0 (cima), 90 (direita), 180 (baixo) e 270 (esquerda). Inicialmente, escolhe uma direção aleatória. Antes de se mover, verifica se está próximo de uma parede, estabelecendo uma variável isAtWall como falsa.

Para verificar a proximidade das paredes, a função analisa a posição atual do 'Cleaner'. Se estiver a uma distância de 15.5 unidades de qualquer parede na direção escolhida, a variável isAtWall é definida como verdadeira. Se a direção aleatória estiver bloqueada por uma parede, a função entra num loop que continua a escolher uma nova direção até que uma direção válida seja selecionada.

Após encontrar uma direção segura, a função atualiza a direção do 'Cleaner' e move-o uma unidade para frente.

6- Atualizar monitores

```
to update-monitors
  ; Atualiza os monitores do nível de poluição, energia e detritos
  set-current-plot "Evolução da Limpeza"
  plot current-debris-count-blue
  set-current-plot "Evolução da Contaminação"
  plot pollution-level
end
```

Figura 29- Função update-monitors

A seguinte função (Figura 29) é responsável por atualizar os monitores que mostram o nível de poluição, energia e a quantidade de detritos. Ela define o gráfico atual como "Evolução da Limpeza" e plota a contagem de detritos atuais do 'Cleaner' azul. Em seguida, plota o nível de poluição no gráfico "Evolução da Contaminação".

2ª Fase da implementação

Na segunda fase deste trabalho, foi nos pedido que implementássemos as nossas próprias ideias, sendo estas inovações ou variantes, de forma a aumentar o grau de complexidade do modelo anterior. O que implementamos foi o seguinte:

1- Quantidade de energia que é carregada por tick

Figura 30- Deslizador charging-rate

O deslizador charging-rate (Figura 30) define a taxa de carregamento da energia dos 'Cleaners'. Durante a execução das funções recharge-blue (Figura 31) e recharge-black (Figura 32), a energia dos 'Cleaners' é aumentada a cada ciclo de acordo com o valor definido por este deslizador. Quanto maior for o valor do charging-rate, mais rapidamente a energia dos 'Cleaners' azul e preto será recarregada, já que a quantidade de energia adicionada por ciclo será maior.

Desta forma, o charging-rate permite ao utilizador controlar a velocidade com que os 'Cleaners' recuperam a sua energia até atingirem o máximo de 100 unidades. Este mecanismo também oferece uma vantagem aos 'Polluters', pois enquanto os 'Cleaners' estão a carregar, os 'Polluters' movem-se aleatoriamente uma posição a cada ciclo de carregamento, permitindo-lhes espalhar mais poluição durante esse período.

```
to recharge-blue
  while [cleaner-energy-blue < 100] [
    set cleaner-energy-blue cleaner-energy-blue + charging-rate
    if cleaner-energy-blue > 100 [ set cleaner-energy-blue 100 ]
    update-monitors
    ask turtles with [role = "Polluter"] [ move ]
    wait temp-carregamento
  ]
end
```

Figura 31- Função recharge-blue

```
to recharge-black
  while [cleaner-energy-black < 100] [
    set cleaner-energy-black cleaner-energy-black + charging-rate
    if cleaner-energy-black > 100 [ set cleaner-energy-black 100 ]
    update-monitors
    ask turtles with [role = "Polluter"] [ move ]
    wait temp-carregamento
  ]
end
```

Figura 32- Função recharge-black

2- Número de Cleaners

Figura 33- Deslizador num-cleaners

Figura 35- Cleaner Preto

```
create-turtles num-cleaners  [
  set role "Cleaner"
  set shape "circle"     ; Define a forma do Cleaner como círculo
  set size 2

  if count turtles with [role = "Cleaner"] = 1 [
    setxy -15.5 -15.5               ; Primeiro charger
    set color blue                  ; cor azul
    set cleaner-type "blue"         ; cleaner type azul
    set cleaner-energy-blue energy  ; define a energia do azul consuante a energia definida
    set charger-assigned one-of turtles with [role = "Charger" and xcor = -15.5 and ycor = -15.5]
  ]
  if count turtles with [role = "Cleaner"] = 2 [
    setxy 15.5 15.5                  ; Segundo charger
    set color black
    set cleaner-type "black"         ; cleaner type preto
    set cleaner-energy-black energy  ; define a energia do preto consuante a energia definida
    set charger-assigned one-of turtles with [role = "Charger" and xcor = 15.5 and ycor = 15.5]  ;
  ]
]
```

Figura 34- Lógica implementada na criação dos 'Cleaners'

Decidimos também permitir que o utilizador opte por utilizar dois 'Cleaners' no seu projeto. Esta possibilidade é viabilizada pelo deslizador "num-cleaners" (Figura 33); quando o valor é definido como dois, é adicionado um segundo 'Cleaner' de cor preta (Figura 35). Sendo este inicializado no canto superior direito do mapa. Com isso, o utilizador consegue recolher os detritos de forma mais rápida e observar a influência que a presença de um ou dois 'Cleaners' tem no desempenho do programa.

3- Contadores extra

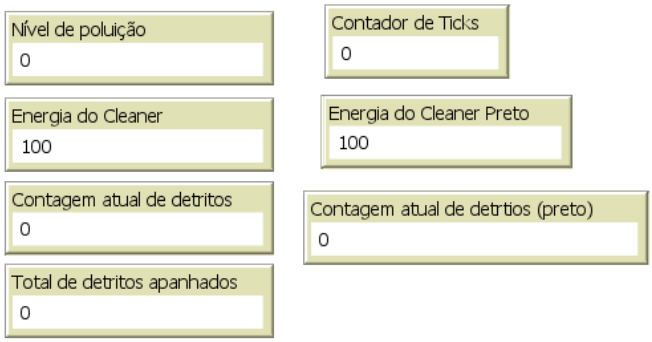

Figura 36- Contadores extra

Na Figura 36, são apresentados os monitores que consideramos interessantes e úteis para o desenvolvimento do projeto. Neles, é possível verificar, por exemplo, a carga da bateria de cada 'Cleaner' e o total de detritos apanhados por ambos.

4- Aumento da inteligência do Cleaner

```
to scan-and-collect-blue
  let radius 1  ; Definir o raio
  let debris-patches patches with [pcolor = red or pcolor = pink or pcolor = orange] in-radius radius

  ; Verifica se há detritos nas proximidades
  if any? debris-patches [
    ; Encontra o patch mais próximo com detritos
    let nearest-debris-patch min-one-of debris-patches [distance myself]

    ; Calcula a direção para o patch de detritos mais próximo
    face nearest-debris-patch

    ; Move-se em direção ao patch mais próximo
    fd 1

    ; Recolhe os detritos se estiver no patch correto
    if (pcolor = red or pcolor = pink or pcolor = orange) and (current-debris-count-blue < max-cleaner-debris) [
      set pcolor green
      set current-debris-count-blue current-debris-count-blue + 1
      set pollution-level pollution-level - 1
      set total-debris-count total-debris-count + 1
    ]
  ]
end
```

Figura 37- Função scan-and-collect-blue

A função scan-and-collect-blue (Figura 37) aumenta a inteligência do 'Cleaner' ao permitir que ele identifique e recolha detritos de forma mais eficiente em comparação com a procura e recolha realizadas na fase um do trabalho (Figura 23). O 'Cleaner' utiliza um raio de alcance (radius) para detetar patches próximos que contenham detritos, representados pelas cores vermelha, rosa ou laranja. Quando localiza detritos, o 'Cleaner' calcula a direção para o patch mais próximo e move-se na sua direção, demonstrando um comportamento mais direcionado e inteligente, em vez de se mover aleatoriamente. Em suma, enquanto um dos 'Cleaners' não deteta uma célula de cor diferente de verde, ele procura de forma aleatória. No entanto, assim

que identifica detritos (célula com cor diferente de verde), ele move-se para o patch mais próximo, seja verticalmente ou horizontalmente, deslocando-se diretamente para o local selecionado.

5- Inicialização segura

```
to safe-spawn
  let trash-turtles turtles with [role = "Trash"]
  let charger one-of turtles with [role = "Charger"]
  ; Verifica se o Charger foi criado
  while [any? other turtles in-radius 4 or (charger != nobody and distance charger < 4)] [
    limit
    setxy random-xcor random-ycor
    limit
  ]
end
```

Figura 37- Função safe-spawn

A função safe-spawn (Figura 37) é essencial para assegurar que novos agentes (turtles) sejam gerados em posições que não interfiram com o funcionamento do ambiente, evitando sobreposição com os 'Chargers' e os detritos (Trash). A função começa identificando as turtles com o papel de Trash e seleciona uma turtle Charger. Em seguida, um loop while verifica se existem outras turtles num raio de 4 unidades ou se o Charger está a menos de 4 unidades da nova posição. Enquanto essas condições forem verdadeiras, novas posições aleatórias são geradas até que uma posição segura seja encontrada, garantindo uma simulação fluida e realista, evitando comportamentos indesejados.

6- Limites definidos

```
; Função que delimita o mapa
to limit
 ; Verifica os es das coordenadas
  if xcor > 15.5 [ set xcor 15.5 ] ; e direito
  if xcor < -15.5 [ set xcor -15.5 ] ; e esquerdo
  if ycor > 15.5 [ set ycor 15.5 ] ; e superior
  if ycor < -15.5 [ set ycor -15.5 ] ; e inferior

end
```

Figura 38- Função limit

A função limit (Figura 38) tem a funcionalidade de delimitar o mapa e garantir que as turtles (incluindo os 'Cleaners') não ultrapassem os limites do ambiente. Ela faz isto verificando as coordenadas x e y das turtles em relação aos valores máximos e mínimos permitidos. Se as turtles ultrapassarem 15.5 unidades nas direções horizontal (x) ou vertical (y), a função ajusta sua posição para o valor limite permitido, assegurando que permaneça dentro da área delimitada.

Esta função é chamada várias vezes ao longo do código para garantir que, tanto na inicialização como durante o movimento dos 'Cleaners', não ultrapassem os limites do mapa.

7- Movimentos dos agentes

```
to deposit-debris
  if debris-type = 0 [
    if random-float 1 < red-polluter-probability and pcolor != red [
      set pcolor red  ; muda a cor da célula para vermelho (tipo 0)
      set pollution-level pollution-level + 1
    ]
  ]
  if debris-type = 1 [
    if random-float 1 < pink-polluter-probability and pcolor != pink [
      set pcolor pink  ; muda a cor da célula para amarelo (tipo 1)
      set pollution-level pollution-level + 1
    ]
  ]
  if debris-type = 2 [
    if random-float 1 < orange-polluter-probability and pcolor != orange [
      set pcolor orange  ; muda a cor da célula para laranja (tipo 2)

      set pollution-level pollution-level + 1
    ]
  ]
end
```

Figura 39- Função deposit-debris

A função deposit-debris (Figura 39) é responsável por depositar resíduos no ambiente, modificando a cor dos patches (células) para representar diferentes tipos de poluição. O tipo de resíduo (definido por debris-type) determina a cor que o patch assume: vermelho, rosa ou laranja. A função usa uma probabilidade aleatória, determinada pelas variáveis red-polluter-probability, pink-polluter-probability e orange-polluter-probability, para decidir se um patch muda de cor. Quando a cor muda, o nível de poluição é incrementado em 1. Este processo simula a criação de novas áreas poluídas conforme diferentes tipos de poluentes são depositados no ambiente. Aumentando assim, a inteligência também dos 'Polluters'.

3. Exemplo

Por fim, para ilustrar o nosso trabalho, apresentamos um exemplo prático com as seguintes condições simuladas:

- Probabilidade de poluição do 'Cleaner' vermelho: 40%
- Probabilidade de poluição do 'Cleaner' rosa: 80%
- Probabilidade de poluição do 'Cleaner' laranja: 30%
- Número de 'Trash': 3
- Tempo de carregamento por tick: 3 segundos
- Capacidade de transporte de detritos: 50 unidades
- Quantidade de energia carregada por tick: 30%

Com base nestas condições, obtivemos o seguinte resultado.

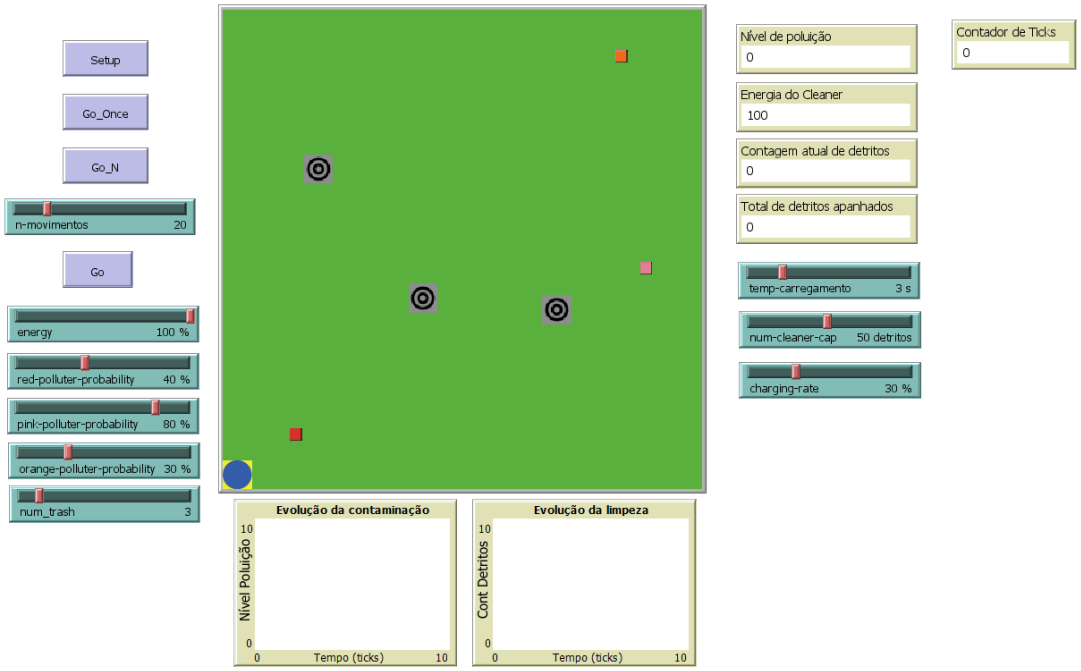

Figura 40- Exemplo

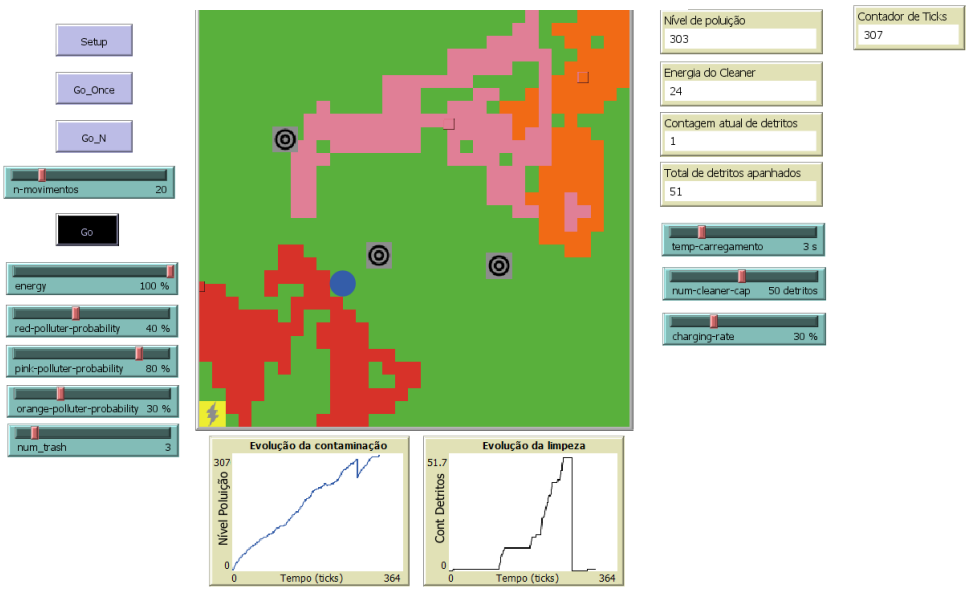

Figura 41- Exemplo II

A Figura 41 ilustra o resultado obtido após 307 ticks, considerando as condições previamente descritas. Esta simulação reflete o comportamento do sistema nas circunstâncias estabelecidas, demonstrando a eficácia dos parâmetros configurados para cada 'Cleaner' e as variáveis envolvidas no processo.

4. Conclusão

Ao longo deste projeto, que envolveu várias fases de implementação, tivemos a oportunidade de aprofundar conceitos essenciais de Inteligência Artificial, explorando a criação e manipulação de variáveis globais e específicas, bem como a implementação de funções avançadas de movimento e interação entre agentes. Na 1ª fase, focámo-nos na configuração inicial, no movimento básico das turtles e na atualização dos monitores de desempenho. Cada passo foi fundamental para estruturar o comportamento dos 'Cleaners' e dos 'Polluters', assegurando que o sistema funcionasse de forma coesa desde o início.

Gostaríamos de salientar que considerámos oportuno integrar algumas funcionalidades e ideias do projeto Robot2 no Robot1, com o intuito de enriquecer e aprimorar o desenvolvimento deste último. Tal decisão foi tomada tendo em conta que o protocolo especificava a necessidade de incluir "os deslizadores e contadores necessários" no modelo.

A 2ª fase trouxe desafios mais complexos, onde ajustámos, por exemplo, a capacidade de carga dos 'Cleaners' e a quantidade de energia recarregada por tick. A integração de novos agentes e contadores exigiu um refinamento contínuo das funções e do sistema como um todo. Um dos marcos dessa fase foi o aumento da inteligência dos 'Cleaners', que passaram a recolher detritos de forma mais eficiente, com movimentos direcionados e decisões informadas. Também implementámos a inicialização segura, assegurando que os agentes fossem posicionados corretamente, evitando colisões e erros no início da simulação. Definimos limites de movimentação para garantir que os 'Cleaners' operassem dentro de uma área confinada, o que ajudou a estruturar o ambiente simulado de forma eficiente.

Esses desafios enriqueceram a nossa experiência e mostraram a importância de considerar cada detalhe ao desenvolver sistemas complexos. Concluímos este projeto com uma visão mais clara sobre os benefícios da adaptação contínua e da inovação no desenvolvimento de soluções. Os objetivos do trabalho foram, a nosso ver, completamente atingidos e este projeto contribuiu imenso para a nossa formação. O sucesso obtido reflete o nosso crescimento técnico e intelectual, e estamos entusiasmados com a perspetiva de aplicar os conhecimentos adquiridos em futuros trabalhos.