UNIVERSIDADE
DE TRÁS-OS-MONTES
E ALTO DOURO

Trabalho Prático 1

Agentes

Licenciatura em Engenharia Informática

Inteligência Artificial

José Paulo Barroso De Moura Oliveira

Autores:

Francisco Ruano – al78474

Maria Inês Cardoso – al78222

Vila Real, outubro 2024

Índice

1. Introdução

No âmbito da unidade curricular de Inteligência Artificial, foi-nos proposto o desenvolvimento de um modelo *NetLogo* capaz de simular um cenário de limpeza de zonas com vários tipos de resíduos.

Numa primeira fase do projeto, pretende-se que os alunos sejam capazes de seguir os pontos de elaboração do modelo, de modo a simular corretamente o processo de poluição e limpeza de uma superfície quadrada.

Na segunda fase do projeto, o modelo proposto deve ser ajustado, de modo a tornar o processo de limpeza mais eficiente. Nesta fase, cada grupo deve implementar as variantes que achar necessárias e relevantes.

2. Desenvolvimento

I. Fase Inicial de Implementação

a. Declaração de Variáveis

Na primeira parte do código (Figura 1), são definidas as variáveis a utilizar ao longo do programa. Além das variáveis globais (*globals*), acedidas por qualquer *turtle* ou *patch* do código, foi criada uma raça (*breed*) para as *turtles* Pessoas, ou seja, os agentes poluidores (*Polluters*). Assim, podemos associar a variável 'probabilidade' a essa raça, uma vez que irá representar a taxa de probabilidade de poluição para cada agente poluidor.

```
breed [pessoas pessoa]
pessoas-own [probabilidade]

globals [bateria numMovimentos numContentores residuosGreen residuosYellow residuosPink
residuosTotal residuosCleaner maxbateria maxdetritos tempCharging charging full]
```

Figura 1- Declaração de Variáveis

b. Setup

Depois da definição das variáveis, é necessário inicializá-las. Assim, na função *setup* (Figura 2), as variáveis são inicializadas com os seus valores *default*, utilizando o comando *set*.

Nesta função, criamos todos os agentes existentes no ambiente: Posto de Carregamento (*shape "house"*), Contentores (*shape "box"*), Agentes Poluidores (*shape "person"*) e Agente de Limpeza (*shape "airplane"*), indicando a sua forma, localização inicial, direção do movimento, cor e dimensão.

```
to setup
  clear-all
  set tempCharging Tempo_Carregamento
  set bateria Bateria_Inicial
  set numMovimentos 0
  set charging false
  set full false

  ; Contagem de resíduos
  set residuosGreen 0
  set residuosYellow 0
  set residuosPink 0
  set residuosTotal 0
  set residuosCleaner 0

  set maxbateria 100.00
  set maxdetritos Numero_Detritos
  set numContentores Numero_Depositos

  ask patches [ set pcolor blue ]

  ;Cria posto de carregamento
  create-turtles 1 [
    set shape "house"
    setxy 0 0
    set color white
    set size 1.0
  ]

; Criar contentor
create-turtles numContentores [
  set shape "box"
  setxy random-pxcor random-pycor
  set heading one-of [0 90 180 270]
  set color brown
  set size 1.0
]

;Criar polluters com diferentes taxas de probabilidade
create-pessoas 1 [
  set shape "person"
  setxy random-pxcor random-pycor
  set heading one-of [0 90 180 270]
  set color (green + 2)
  set size 1.0
  set probabilidade Taxa_Probabilidade_Green
]

create-pessoas 1 [
  set shape "person"
  setxy random-pxcor random-pycor
  set heading one-of [0 90 180 270]
  set color yellow
  set size 1.0
  set probabilidade Taxa_Probabilidade_Yellow
]

create-pessoas 1 [
  set shape "person"
  setxy random-pxcor random-pycor
  set heading one-of [0 90 180 270]
  set color (pink + 2)
  set size 1.0
  set probabilidade Taxa_Probabilidade_Pink
]

; Criar cleaner
create-turtles 1 [
  set shape "airplane"
  setxy 0 0
  set heading one-of [0 90 180 270]
  set color cyan
  set size 1.0
]

  reset-ticks
end
```

Figura 2- Função "setup"

Após definir os valores desejados para cada um dos deslizadores existentes na interface, quando clicamos no botão *Setup*, o ambiente é inicializado tal como apresentado na Figura 3.

Cada vez que o botão *Setup* é pressionado, todos os agentes são removidos do ambiente, os monitores, gráficos e variáveis são reiniciadas (*clear-all*), bem como o contador de *ticks* (*reset-ticks*).

Em relação ao comportamento dos agentes pelo mundo, este pode acontecer de 3 formas distintas:

- Botão *Go_Once*: está associado à função *"go_once"*, e a cada movimento dos agentes, é necessário clicar no botão sucessivamente, para executar os próximos deslocamentos;
- Botão *Go_N*: está associado à função *"go_n"*, e a cada movimento dos agentes, é necessário clicar no botão sucessivamente, para executar os próximos deslocamentos. Os agentes movimentam-se o número de vezes igual ao definido pelo utilizador no deslizador *N_Ticks*;
- Botão *Go*: está associado à função *"go"*, e os agentes deslocam-se de forma contínua, sem necessidade constante da utilização do botão.

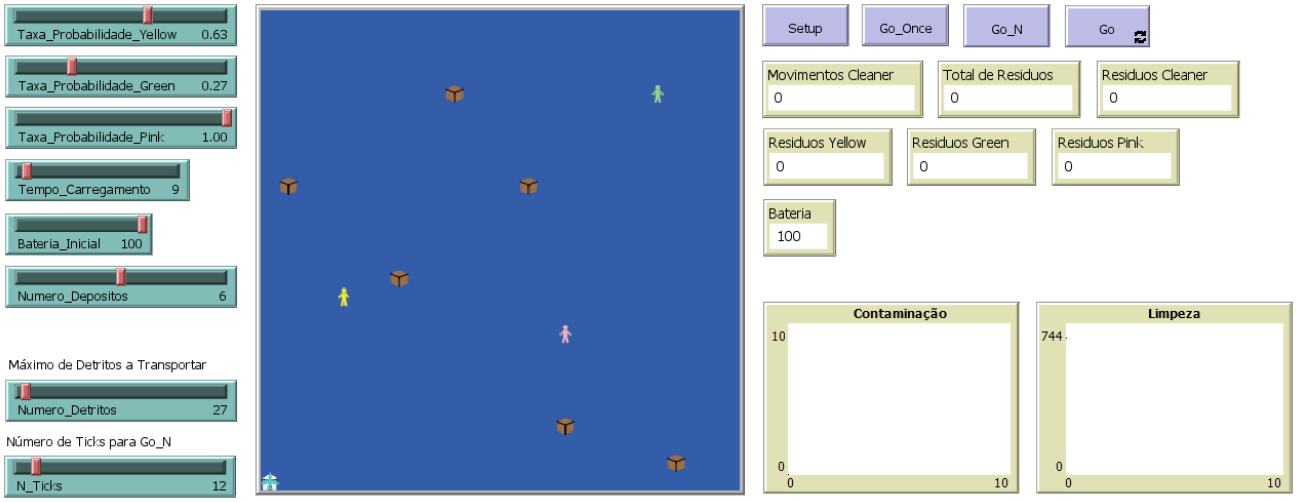

Figura 3- Interface Inicial

c. Movimentação *Polluters* e *Cleaner*

Nestas funções é definido o movimento dos agentes poluidores (Figura 4) e o movimento dos agentes de limpeza (Figura 5). Enquanto os *Polluters* têm como objetivo poluir o ambiente, ou seja, deslocam-se pelo ambiente de forma a procurar pintar o maior número de células limpas (células azuis), o *Cleaner* desloca-se de modo a procurar pintar de azul o maior número de células poluídas possível (células verdes, amarelas e rosas).

Assim, foi criada uma lógica semelhante para ambos. Isto é, os agentes procuram as células com a cor que desejam pintar adjacentes à célula onde se encontram, dirigindo-se a elas. Caso não existam células pretendidas adjacentes à sua posição, estes deslocam-se de forma aleatória no ambiente, escolhendo uma das direções (*forward*, *back*, *right*, *left*) para se dirigirem.

```
to movimentacao

  if shape = "person" [
    let adjacente-azul one-of neighbors4 with [pcolor = blue]
    if adjacente-azul != nobody [
      face adjacente-azul
      move-to adjacente-azul
    ]

    if adjacente-azul = nobody [
      let direction one-of ["forward" "backward" "right" "left"]
      if direction = "forward" [forward 1]
      if direction = "backward" [back 1]
      if direction = "right" [right 90 forward 1]
      if direction = "left" [left 90 forward 1]
    ]
  ]
end
```

Figura 4- Função "movimentacao"

```
to movimentacao_airplane

  ask turtles with [shape = "airplane" and color = cyan] [
    if bateria > 0 [
      let adjacente one-of neighbors4 with [pcolor = (green + 2) or pcolor = yellow or pcolor = (pink + 2)]

      if adjacente != nobody [
        face adjacente
        move-to adjacente
        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
        if residuosCleaner < maxdetritos [ contadorResiduos ]
      ]
      if adjacente = nobody [
        let direction one-of ["forward" "backward" "right" "left"]
        if direction = "forward" [forward 1]
        if direction = "backward" [back 1]
        if direction = "right" [right 90 forward 1]
        if direction = "left" [left 90 forward 1]
        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
      ]
    ]
  ]
end
```

Figura 5- Função "movimentacao_airplane"

d. Contagem Total de Resíduos

Tal como podemos observar na Figura 6, esta função é responsável pela contagem do número total de resíduos recolhidos pelo *Cleaner* durante todo o seu movimento. Ao mesmo tempo que o avião passa nas células poluídas, a cor das mesmas é alterada para azul transformando-as em células limpas.

```
to contadorResiduos

  let current-patch patch-here
  if [pcolor] of current-patch != blue [
    ; Conta se a célula era vermelha, amarela ou laranja
    if [pcolor] of current-patch = (green + 2) [ set residuosGreen residuosGreen + 1 ]
    if [pcolor] of current-patch = yellow [ set residuosYellow residuosYellow + 1 ]
    if [pcolor] of current-patch = (pink + 2) [ set residuosPink residuosPink + 1 ]

    set residuosTotal residuosGreen + residuosYellow + residuosPink
    set residuosCleaner residuosCleaner + 1
    set pcolor blue
  ]
end
```

Figura 6- Função "contadorResiduos"

e. Movimentação para o Posto de Carregamento

A movimentação do *Cleaner* para o posto de carregamento ocorre quando este necessita de recarregar a sua bateria para continuar a sua movimentação pelo ambiente (Figura 7).

Sendo que o posto de carregamento se encontra no ponto inicial, ou seja, nas coordenadas (0,0), o avião move-se ao longo do eixo X, ajustando a sua posição até que a sua abcissa seja 0. Após se alinhar no eixo X, este move-se ao longo do eixo Y, da mesma forma, até que a sua ordenada seja 0.

Quando o *Cleaner* chegar finalmente ao posto de carregamento, vai ativar a variável *charging* e iniciar o seu carregamento, através da chamada da função *"chargingBat".*

```
to gotoPosto

  if bateria >= 0 [
    if xcor != 0 [
      if xcor > 0 [ set xcor xcor - 1 ]
      if xcor < 0 [ set xcor xcor + 1 ]
    ]
    if xcor = 0 and ycor != 0 [
      if ycor > 0 [ set ycor ycor - 1 ]
      if ycor < 0 [ set ycor ycor + 1 ]
    ]

    set bateria bateria - 1
    set numMovimentos numMovimentos + 1
  ]

  ; Quando chegar ao posto (0,0)
  if xcor = 0 and ycor = 0 [
    set charging true
    chargingBat
  ]
end
```

Figura 7- Função "gotoPosto"

f. Carregamento da Bateria

Uma vez ativada a variável *charging*, o Cleaner inicia o seu carregamento, tal como pode ser observado na Figura 8.

Neste seguimento, foi criada uma lógica para que enquanto a bateria não chegar ao máximo de bateria que pode ser atingida pelo *Cleaner* (indicado pela variável *maxbateria* na função *"setup"*), a bateria aumenta com base numa proporção entre o seu valor máximo e o tempo de carregamento que o *Cleaner* deve demorar a carregar totalmente (indicado no deslizador Tempo_Carregamento da interface, pela variável *tempCharging*). Assim é possível controlar a quantidade de carga que o avião recebe em cada ciclo de recarga de bateria.

```
to chargingBat

  if charging [
    if bateria < maxbateria [
      set bateria bateria + (maxbateria / tempCharging)
    ]

    if bateria >= maxbateria [
      set bateria maxbateria
      set charging false
    ]
  ]
end
```

Figura 8- Função "chargingBat"

g. Movimentação para um Contentor

A movimentação do *Cleaner* para um contentor ocorre quando este atinge a sua capacidade máxima e necessita de descarregar os resíduos que transporta para continuar a sua movimentação pelo ambiente (Figura 9).

Uma vez que podem existir entre dois e dez depósitos do lixo, o *Cleaner* deve tomar uma decisão sobre a qual contentor se deve dirigir. Assim, esta função foi implementada de forma que o *Cleaner* verifique qual o contentor mais próximo de si, através do cálculo da distância.

Antes de se mover, verifica também se a bateria que tem é suficiente para alcançar o contentor e voltar ao posto de carregamento, garantindo que a soma das distâncias para o contentor e o posto não excede a carga disponível.

Tal como na função *"gotoPosto"*, o *Cleaner* desloca-se através dos eixos X e Y no ambiente, sendo que a posição final onde se deve encontrar é a do contentor mais próximo. Quando o *Cleaner* chega ao contentor, vai iniciar a sua descarga, através da chamada da função *"descarregar"*.

```
to gotoContentor

  if shape = "airplane" [
    if residuosCleaner = maxdetritos [

      let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor - [xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
      let distanceContentor distance contentor_maisprox
      let distancePosto (abs xcor) + (abs ycor)

      if bateria > (distanceContentor + distancePosto) [
        if xcor != [xcor] of contentor_maisprox [
          if xcor > [xcor] of contentor_maisprox [ set xcor xcor - 1 ]
          if xcor < [xcor] of contentor_maisprox [ set xcor xcor + 1 ]
        ]
        if [ycor] of contentor_maisprox != ycor [
          if ycor > [ycor] of contentor_maisprox [ set ycor ycor - 1 ]
          if ycor < [ycor] of contentor_maisprox [ set ycor ycor + 1 ]
        ]

        ; Quando o avião chega ao contentor
        if distanceContentor = 0 [ descarregar ]

        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
      ]
    ]
  ]
```

Figura 9- Função "gotoContentor"

h. Descarga de Resíduos

No momento da chegada do *Cleaner* ao contentor, este inicia a descarga dos resíduos que recolheu ao longo do seu movimento, tal como pode ser observado na Figura 10.

Esta descarga é instantânea, uma vez que a carga transportada pelo Cleaner é imediatamente redefinida, alterando o seu estado de cheio para vazio (*set full false*). Isso é essencial para garantir que o avião possa continuar a sua tarefa sem interrupções.

```
to descarregar

  if residuosCleaner = maxdetritos [
    set residuosCleaner 0
    set full false
  ]
end
```

Figura 10- Função "descarregar"

i. Movimentação Geral

Tal como já foi referido anteriormente, as funções *"go"*, *"go_once"* e *"go_n"* coordenam o comportamento de todos os agentes do ambiente. Cada vez que estas funções são executadas, simulam um passo no tempo (*tick*), onde os agentes tomam decisões e realizam ações com base no seu estado atual. Dentro delas são chamadas todas as outras funções explicadas previamente no presente relatório.

De acordo com a sua taxa de probabilidade de deposição de lixo, os *Polluters* deslocam-se no ambiente, poluindo as células por onde passam estas se encontrem limpas.

O movimento do *Cleaner* é uma ação mais complexa e que deve ser repensada a cada deslocamento, uma vez que este deve ser capaz de gerir a sua bateria e capacidade de recolha de resíduos enquanto executa a sua tarefa.

Neste sentido, o *Cleaner* vai calculando as distâncias a que se encontra do posto de carregamento e do contentor mais próximo para tomar uma decisão mais precisa.

Se este agente de limpeza ainda não tiver atingido a sua capacidade máxima de resíduos, decide com base na sua bateria se deve ir ao posto de carregamento ou continuar a tarefa de recolha.

Se porventura o *Cleaner* atingir a sua capacidade máxima de resíduos e não tiver bateria suficiente para chegar ao contentor e continuar o seu movimento, este deve deslocar-se em primeiro lugar ao posto de carregamento para carregar a bateria, e posteriormente seguir para o contentor e realizar a descarga de resíduos. Desta forma, prioriza a ida ao posto de carregamento, uma vez que não pode continuar a movimentação pelo ambiente se não tiver bateria.

Toda esta lógica pode ser observada na Figura 11, que representa uma das funções de movimentação dos agentes. As outras duas funções (*"go_once"* e *"go_n"*) têm uma lógica semelhante à mesma, tendo em conta o papel atribuído a cada botão da interface.

```
to go

  ask pessoas [
    if [pcolor] of patch-here = blue [
      if random-float 1 < probabilidade [ set pcolor color ]
    ]
    movimentacao
  ]

  ask turtles with [shape = "airplane"] [
    let distancePosto (abs xcor) + (abs ycor)
    let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor - [xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let distanceContentor distance contentor_maisprox

    if residuosCleaner < maxdetritos [
      if bateria <= distancePosto and not charging [ gotoPosto ]
      if bateria > distancePosto and not charging [ movimentacao_airplane ]
    ]

    if residuosCleaner = maxdetritos [
      if contentor_maisprox != nobody [
        set full true
        if bateria > (distancePosto + distanceContentor) and not charging [ gotoContentor ]
        if bateria <= (distancePosto + distanceContentor) and not charging [ gotoPosto ]
      ]
    ]

    if xcor = 0 and ycor = 0 [
      set charging true
      chargingBat
    ]
  ]
  tick
end
```

Figura 11- Função "go"

j. Exemplo de Teste 1

Considerando o cenário inicialmente apresentado na Figura 3, ao fim de 500 *ticks*, o ambiente apresenta o seguinte aspeto:

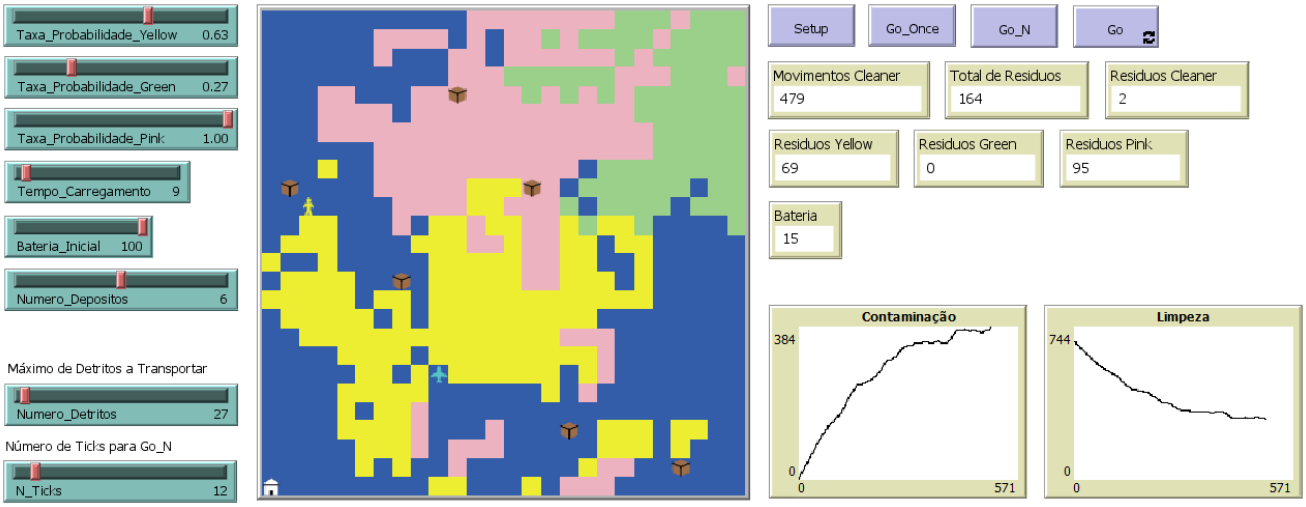

Figura 12- Interface de Teste 1

Na Figura 12, é possível observar a evolução das taxas de contaminação e limpeza do ambiente ao longo do tempo (Gráficos *"Contaminação"* e *"Limpeza"*). Apesar de inicialmente, a taxa de contaminação ser nula, enquanto a taxa de limpeza se encontra no seu valor máximo, à medida que o tempo avança os agentes poluidores passam a contaminar o ambiente a um ritmo superior à capacidade de limpeza do *Cleaner*. Isso resulta em um aumento gradual da contaminação, apesar do esforço contínuo de limpeza.

Além disso podemos também visualizar a quantidade de resíduos recolhida pelo agente de limpeza, incluindo o total de resíduos de cada cor (Monitores *"Residuos Yellow"*, *"Residuos Green"* e *"Residuos Pink"*), a soma final (Monitor *"Total de Resíduos"*) e ainda os resíduos que o *Cleaner* está a transportar no momento (Monitor *"Residuos Cleaner"*).

II. Segunda Fase de Implementação

Por forma a aumentar o grau de complexidade do modelo anteriormente desenvolvido, decidimos que seria útil criar mais agentes. Assim, a maioria das funções usadas anteriormente na Primeira Fase do projeto, foram alteradas de forma a possuírem parâmetros de entrada.

Esta adaptação permite a flexibilidade e reutilização de código, uma vez que podemos utilizar a mesma função, com diferentes valores, para vários agentes, sem ser necessário duplicar código.

a. Número de *Cleaners*

Uma vez que os *Cleaners* são essenciais para melhorar a eficiência de limpeza do ambiente, achamos relevante a criação de mais um agente de limpeza. Assim, como o primeiro *Cleaner* estava a nascer na posição (0,0), o segundo *Cleaner* nasce no vértice oposto (Figura 13).

```
create-turtles 1 [
  set shape "airplane"
  setxy 0 0
  set heading one-of [0 90 180 270]
  set color cyan
  set size 1.0
]

create-turtles 1 [
  set shape "airplane"
  setxy 25 25
  set heading one-of [0 90 180 270]
  set color (violet + 2)
  set size 1.0
]
```

Figura 13- Criação dos Cleaners na função "setup"

b. Número de Postos de Carregamento

Dado que o número de *Cleaners* no ambiente aumento, é lógico aumentarmos o número de postos de carregamento para assegurar o suporte adequado à execução da tarefa de todos estes agentes e evitando a sobrecarga de um único posto. Deste modo, estabelecemos quatro postos de carregamento, cada um situado num vértice distinto do ambiente (Figura 14).

```
; Criar postos de carregamento
create-turtles 1 [
  set shape "house"
  setxy 0 0
  set color white
  set size 1.0
]

create-turtles 1 [
  set shape "house"
  setxy 0 25
  set color white
  set size 1.0
]

create-turtles 1 [
  set shape "house"
  setxy 25 25
  set color white
  set size 1.0
]

create-turtles 1 [
  set shape "house"
  setxy 25 0
  set color white
  set size 1.0
]
```

Figura 14- Criação dos postos de carregamento na função "setup"

c. Mudanças na Interface

Devido ao aumento do número de agentes no ambiente, foi necessário fazer alguns ajustes na interface do projeto (Figura 15). Assim sendo, foram acrescentados dois deslizadores, de forma a separar as definições de bateria e tempo de carregamento de cada *Cleaner*. Da mesma forma foram também criados mais dois monitores, com o propósito de separar a contagem de resíduos transportada por cada *Cleaner*, bem como a bateria de cada um.

Além disso, os monitores de movimentação e contagem de resíduos (total e por cores) passam a mostrar a soma dos valores desejados de ambos os agentes de limpeza.

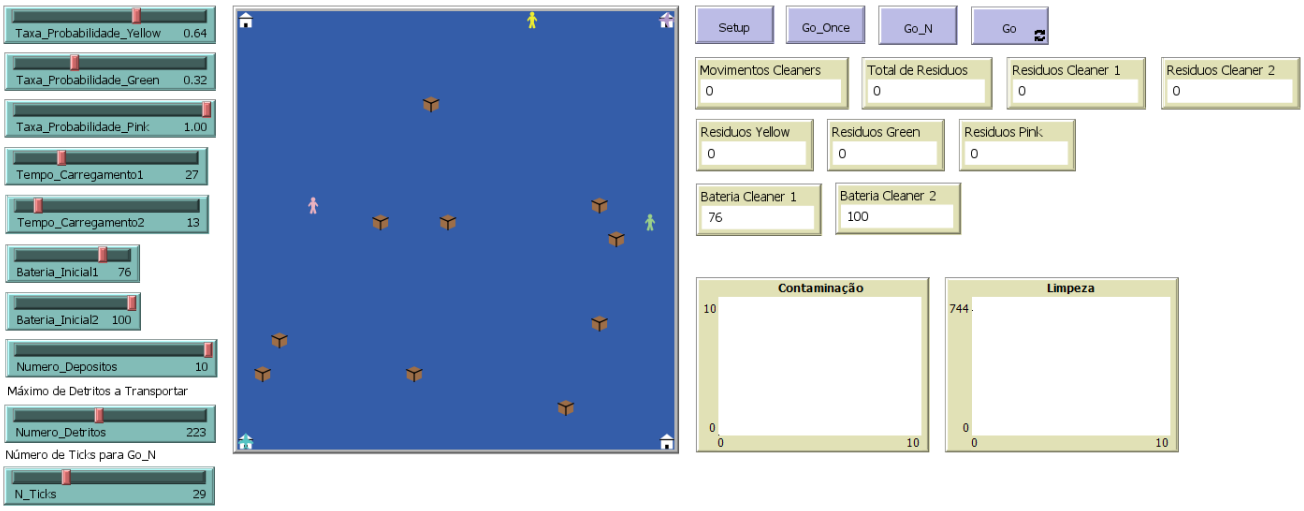

Figura 15- Modificação da Interface Inicial

d. Movimentação Geral

Comparando a função *"go"* com a anteriormente desenvolvida na primeira fase do projeto, notam-se algumas diferenças. Decidimos separar esta função em duas, *"go"* (Figura 16) e *"handle_airplane"* (Figura 17), com o propósito de tornar o código menos confuso e mais intuitivo. Assim, passamos a chamar a função "*handle_airplane*" separadamente, dentro da função "*go*", para cada um dos *Cleaners*.

Apesar da lógica de movimentação dos agentes continuar a ser a mesma, com a adição de novos postos de carregamento foi necessário modificar a distância ao posto de carregamento. Esta agora passa a ser semelhante ao calculo da distância aos contentores, determinando assim qual o posto de carregamento mais próximo à posição do *Cleaner*.

```
to go

  ask pessoas [
    if [pcolor] of patch-here = blue [
      if random-float 1 < probabilidade [ set pcolor color ]
    ]
    movimentacao
  ]

  ; Movimentação avião 1
  handle_airplane cyan bateria1 residuosCleaner1 charging1 full1 tempCharging1
  ; Movimentação avião 2
  handle_airplane (violet + 2) bateria2 residuosCleaner2 charging2 full2 tempCharging2

  tick
end
```

Figura 16- Modificação da função "go"

```
to handle_airplane [airplane_color bateria residuosCleaner charging full tempCharging]

  ask turtles with [shape = "airplane" and color = airplane_color ] [
    let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor - [xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let posto_maisprox min-one-of turtles with [shape = "house"] [ (abs (xcor - [xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let distanceContentor distance contentor_maisprox
    let distancePosto distance posto_maisprox

    if residuosCleaner < maxdetritos [
      if posto_maisprox != nobody [
        if bateria <= distancePosto and not charging [ gotoPosto airplane_color bateria charging tempCharging]
        if bateria > distancePosto and not charging [ movimentacao_airplane airplane_color bateria residuosCleaner charging]
      ]
    ]

    if residuosCleaner = maxdetritos [
      if contentor_maisprox != nobody and posto_maisprox != nobody [
        if airplane_color = cyan [ set full1 true ]
        if airplane_color = (violet + 2) [ set full2 true ]
        if bateria > (distancePosto + distanceContentor) and not charging [ gotoContentor airplane_color bateria residuosCleaner charging tempCharging full ]
        if bateria <= (distancePosto + distanceContentor) and not charging [ gotoPosto airplane_color bateria charging tempCharging]
      ]
    ]

    if xcor = [xcor] of posto_maisprox and ycor = [ycor] of posto_maisprox [
      if airplane_color = cyan [ set charging1 true ]
      if airplane_color = (violet + 2) [ set charging2 true ]
      if charging1 and airplane_color = cyan [ chargingBat cyan ]
      if charging2 and airplane_color = (violet + 2) [ chargingBat (violet + 2) ]
    ]
  ]
end
```

Figura 17- Função "handle_airplane"

e. Exemplo de Teste 2

Considerando o cenário inicialmente apresentado na Figura 15, ao fim de 500 *ticks*, o ambiente apresenta o seguinte aspeto:

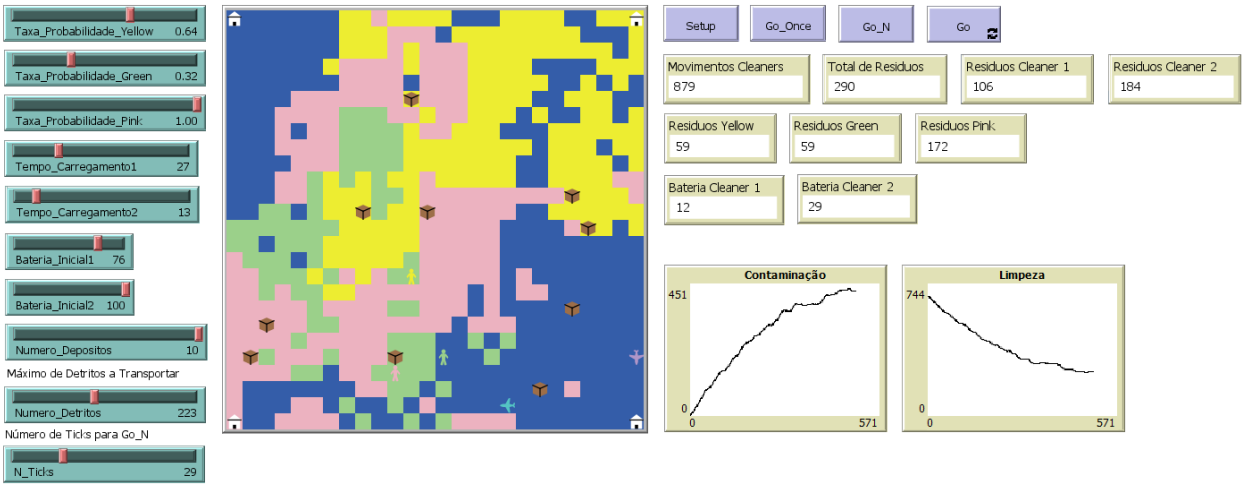

Figura 18- Interface de Teste 2

À semelhança da Primeira Fase de Implementação, na Figura 18 podemos observar a evolução dos diferentes parâmetros ao longo do tempo, tanto nos monitores como nos gráficos.

Além disso podemos também visualizar a quantidade de resíduos recolhida pelos agentes de limpeza, incluindo o total de resíduos de cada cor (Monitores *"Residuos Yellow"*, *"Residuos Green"* e *"Residuos Pink"*), a soma final (Monitor *"Total de Resíduos"*) e ainda os resíduos que cada um dos *Cleaners* está a transportar no momento (Monitor *"Residuos Cleaner"*).

3. Conclusão

Este projeto simula um ambiente no qual ocorre o processo de poluição e limpeza de uma superfície, com agentes poluidores e agentes de limpeza. Ao longo do tempo, foram implementadas funcionalidades como a gestão da bateria eficiente dos Cleaners, garantindo que era possível efetuar limpezas e descarregamento sem que se esgotasse a bateria, a descarga de resíduos e o cálculo de distâncias, permitindo que os agentes tomassem decisões de forma autónoma e eficiente.

A modificação de funções responsáveis por este tipo de comportamentos, na segunda fase de implementação, foi crucial para suportar uma nova gestão de variáveis, tornando possível a construção de uma solução ainda mais eficiente para o presente cenário. Consequentemente, foi necessário alterar a interface para uma nova estrutura com múltiplos Cleaners, capazes de um controlo mais granular sobre as suas variáveis.

Em resumo, o modelo atendeu aos objetivos propostos, demonstrando a capacidade de simulação em diferentes cenários de poluição e limpeza, com uma arquitetura escalável que permite a adição de novos agentes e funcionalidades. Através das várias fases de implementação, conseguimos construir uma solução eficiente e flexível para o problema apresentado.

Anexos

I. Anexo A – Código da Fase Inicial de Implementação

```
breed [pessoas pessoa]
pessoas-own [probabilidade]
globals [bateria numMovimentos numContentores residuosGreen residuosYellow
residuosPink residuosTotal residuosCleaner maxbateria maxdetritos tempCharging
charging full]

to setup
  clear-all
  set tempCharging Tempo_Carregamento
  set bateria Bateria_Inicial
  set numMovimentos 0
  set charging false
  set full false
  set residuosGreen 0
  set residuosYellow 0
  set residuosPink 0
  set residuosTotal 0
  set residuosCleaner 0
  set maxbateria 100.00
  set maxdetritos Numero_Detritos
  set numContentores Numero_Depositos

  ask patches [ set pcolor blue ]
  create-turtles 1 [
    set shape "house"
    setxy 0 0
    set color white
    set size 1.0
  ]
  create-turtles numContentores [
    set shape "box"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color brown
    set size 1.0
  ]
  create-pessoas 1 [
    set shape "person"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color (green + 2)
    set size 1.0
    set probabilidade Taxa_Probabilidade_Green
  ]
  create-pessoas 1 [
    set shape "person"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color yellow
    set size 1.0
    set probabilidade Taxa_Probabilidade_Yellow
  ]
```

```
  create-pessoas 1 [
    set shape "person"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color (pink + 2)
    set size 1.0
    set probabilidade Taxa_Probabilidade_Pink
  ]
  create-turtles 1 [
    set shape "airplane"
    setxy 0 0
    set heading one-of [0 90 180 270]
    set color cyan
    set size 1.0
  ]

  reset-ticks
end

to go_once
  ask pessoas [
    if [pcolor] of patch-here = blue [
      if random-float 1 < probabilidade [ set pcolor color ]
    ]
    movimentacao
  ]

  ask turtles with [shape = "airplane"] [
    let distancePosto (abs xcor) + (abs ycor)
    let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let distanceContentor distance contentor_maisprox

    if residuosCleaner < maxdetritos [
      if bateria <= distancePosto and not charging [ gotoPosto ]
      if bateria > distancePosto and not charging [ movimentacao_airplane ]
    ]
    if residuosCleaner = maxdetritos [
      if contentor_maisprox != nobody [
        set full true
        if bateria > (distancePosto + distanceContentor) and not charging [
gotoContentor ]
        if bateria <= (distancePosto + distanceContentor) and not charging [
gotoPosto ]
      ]
    ]

    if xcor = 0 and ycor = 0 [
      set charging true
      chargingBat
    ]
  ]

  tick
end
```

```
to go_n
  if ticks < N_Ticks [
    ask turtles with [shape = "person"] [
      if [pcolor] of patch-here = blue [
        if random-float 1 < probabilidade [ set pcolor color ]
      ]
      movimentacao
    ]
    ask turtles with [shape = "airplane"] [
      let distancePosto (abs xcor) + (abs ycor)
      let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
      let distanceContentor distance contentor_maisprox

      if residuosCleaner < maxdetritos [
        if bateria <= distancePosto and not charging [ gotoPosto ]
        if bateria > distancePosto and not charging [ movimentacao_airplane ]
      ]
      if residuosCleaner = maxdetritos [
        if contentor_maisprox != nobody [
          set full true
          if bateria > (distancePosto + distanceContentor) and not charging [
gotoContentor ]
          if bateria <= (distancePosto + distanceContentor) and not charging [
gotoPosto ]
        ]
      ]

      if xcor = 0 and ycor = 0 [
        set charging true
        chargingBat
      ]
    ]
  ]

  if ticks >= N_Ticks [ stop ]
  tick
end

to go
  ask pessoas [
    if [pcolor] of patch-here = blue [
      if random-float 1 < probabilidade [ set pcolor color ]
    ]
    movimentacao
  ]
  ask turtles with [shape = "airplane"] [
    let distancePosto (abs xcor) + (abs ycor)
    let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let distanceContentor distance contentor_maisprox

    if residuosCleaner < maxdetritos [
      if bateria <= distancePosto and not charging [ gotoPosto ]
      if bateria > distancePosto and not charging [ movimentacao_airplane ]
    ]
    if residuosCleaner = maxdetritos [
      if contentor_maisprox != nobody [
```

```
        set full true
        if bateria > (distancePosto + distanceContentor) and not charging [
gotoContentor ]
if bateria <= (distancePosto + distanceContentor) and not charging [ gotoPosto ]
      ]
    ]

    if xcor = 0 and ycor = 0 [
      set charging true
      chargingBat
    ]
  ]

  tick
end

to movimentacao
  if shape = "person" [
    let adjacente-azul one-of neighbors4 with [pcolor = blue]
    if adjacente-azul != nobody [
      face adjacente-azul
      move-to adjacente-azul
    ]
    if adjacente-azul = nobody [
      let direction one-of ["forward" "backward" "right" "left"]
      if direction = "forward" [forward 1]
      if direction = "backward" [back 1]
      if direction = "right" [right 90 forward 1]
      if direction = "left" [left 90 forward 1]
    ]
  ]
end

to movimentacao_airplane
  ask turtles with [shape = "airplane" and color = cyan] [
    if bateria > 0 [
      let adjacente one-of neighbors4 with [pcolor = (green + 2) or pcolor = yellow
or pcolor = (pink + 2)]
      if adjacente != nobody [
        face adjacente
        move-to adjacente
        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
        if residuosCleaner < maxdetritos [ contadorResiduos ]
      ]
      if adjacente = nobody [
        let direction one-of ["forward" "backward" "right" "left"]
        if direction = "forward" [forward 1]
        if direction = "backward" [back 1]
        if direction = "right" [right 90 forward 1]
        if direction = "left" [left 90 forward 1]
        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
      ]
    ]
  ]
end
```

```
to contadorResiduos
  let current-patch patch-here
  if [pcolor] of current-patch != blue [
    if [pcolor] of current-patch = (green + 2) [ set residuosGreen residuosGreen + 1
]
    if [pcolor] of current-patch = yellow [ set residuosYellow residuosYellow + 1 ]
    if [pcolor] of current-patch = (pink + 2) [ set residuosPink residuosPink + 1 ]
    set residuosTotal residuosGreen + residuosYellow + residuosPink
    set residuosCleaner residuosCleaner + 1
    set pcolor blue
  ]
end

to gotoPosto
  if bateria >= 0 [
    if xcor != 0 [
      if xcor > 0 [ set xcor xcor - 1 ]
      if xcor < 0 [ set xcor xcor + 1 ]
    ]
    if xcor = 0 and ycor != 0 [
      if ycor > 0 [ set ycor ycor - 1 ]
      if ycor < 0 [ set ycor ycor + 1 ]
    ]
    set bateria bateria - 1
    set numMovimentos numMovimentos + 1
  ]

  if xcor = 0 and ycor = 0 [
    set charging true
    chargingBat
  ]
end

to gotoContentor
  if shape = "airplane" [
    if residuosCleaner = maxdetritos [
      let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
      let distanceContentor distance contentor_maisprox
      let distancePosto (abs xcor) + (abs ycor)

      if bateria > (distanceContentor + distancePosto) [
        if xcor != [xcor] of contentor_maisprox [
          if xcor > [xcor] of contentor_maisprox [ set xcor xcor - 1 ]
          if xcor < [xcor] of contentor_maisprox [ set xcor xcor + 1 ]
        ]
        if [ycor] of contentor_maisprox != ycor [
          if ycor > [ycor] of contentor_maisprox [ set ycor ycor - 1 ]
          if ycor < [ycor] of contentor_maisprox [ set ycor ycor + 1 ]
        ]
        if distanceContentor = 0 [ descarregar ]
        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
      ]
    ]
  ]
end
```

```
to chargingBat
  if charging [
    if bateria < maxbateria [
      set bateria bateria + (maxbateria / tempCharging)
    ]
    if bateria >= maxbateria [
      set bateria maxbateria
      set charging false
    ]
  ]
end

to descarregar
  if residuosCleaner = maxdetritos [
    set residuosCleaner 0
    set full false
  ]
end
```

II. Anexo B – Código da Segunda Fase de Implementação

```
breed [pessoas pessoa]
pessoas-own [probabilidade]
globals [bateria1 bateria2 numMovimentos numContentores residuosGreen residuosYellow
residuosPink residuosTotal residuosCleaner1 residuosCleaner2 maxbateria maxdetritos
tempCharging1 tempCharging2 charging1 charging2 full1 full2]

to setup
  clear-all
  set numMovimentos 0
  set tempCharging1 Tempo_Carregamento1
  set tempCharging2 Tempo_Carregamento2
  set bateria1 Bateria_Inicial1
  set bateria2 Bateria_Inicial2
  set charging1 false
  set charging2 false
  set full1 false
  set full2 false
  set residuosGreen 0
  set residuosYellow 0
  set residuosPink 0
  set residuosTotal 0
  set residuosCleaner1 0
  set residuosCleaner2 0
  set maxbateria 100.00
  set maxdetritos Numero_Detritos
  set numContentores Numero_Depositos

  ask patches [ set pcolor blue ]
  create-turtles 1 [
    set shape "house"
    setxy 0 0
    set color white
    set size 1.0
  ]
  create-turtles 1 [
    set shape "house"
    setxy 0 25
    set color white
    set size 1.0
  ]
  create-turtles 1 [
    set shape "house"
    setxy 25 25
    set color white
    set size 1.0
  ]
  create-turtles 1 [
    set shape "house"
    setxy 25 0
    set color white
    set size 1.0
  ]
```

```
  create-turtles numContentores [
    set shape "box"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color brown
    set size 1.0
  ]
  create-pessoas 1 [
    set shape "person"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color (green + 2)
    set size 1.0
    set probabilidade Taxa_Probabilidade_Green
  ]
  create-pessoas 1 [
    set shape "person"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color yellow
    set size 1.0
    set probabilidade Taxa_Probabilidade_Yellow
  ]
  create-pessoas 1 [
    set shape "person"
    setxy random-pxcor random-pycor
    set heading one-of [0 90 180 270]
    set color (pink + 2)
    set size 1.0
    set probabilidade Taxa_Probabilidade_Pink
  ]
  create-turtles 1 [
    set shape "airplane"
    setxy 0 0
    set heading one-of [0 90 180 270]
    set color cyan
    set size 1.0
  ]

  create-turtles 1 [
    set shape "airplane"
    setxy 25 25
    set heading one-of [0 90 180 270]
    set color (violet + 2)
    set size 1.0
  ]

  reset-ticks
end
```

```
to go_once
  ask pessoas [
    if [pcolor] of patch-here = blue [
      if random-float 1 < probabilidade [ set pcolor color ]
    ]
    movimentacao
  ]
  handle_airplane cyan bateria1 residuosCleaner1 charging1 full1 tempCharging1
  handle_airplane (violet + 2) bateria2 residuosCleaner2 charging2 full2
tempCharging2

  tick
end

to go_n
  if ticks < N_Ticks [
    ask pessoas [
      if [pcolor] of patch-here = blue [
        if random-float 1 < probabilidade [ set pcolor color ]
      ]
      movimentacao
    ]
    handle_airplane cyan bateria1 residuosCleaner1 charging1 full1 tempCharging1
    handle_airplane (violet + 2) bateria2 residuosCleaner2 charging2 full2
tempCharging2
  ]
  if ticks >= N_Ticks [ stop ]

  tick
end

to go
  ask pessoas [
    if [pcolor] of patch-here = blue [
      if random-float 1 < probabilidade [ set pcolor color ]
    ]
    movimentacao
  ]
  handle_airplane cyan bateria1 residuosCleaner1 charging1 full1 tempCharging1
  handle_airplane (violet + 2) bateria2 residuosCleaner2 charging2 full2
tempCharging2

  tick
end

to handle_airplane [airplane_color bateria residuosCleaner charging full
tempCharging]
  ask turtles with [shape = "airplane" and color = airplane_color ] [
    let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let posto_maisprox min-one-of turtles with [shape = "house"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
    let distanceContentor distance contentor_maisprox
    let distancePosto distance posto_maisprox

    if residuosCleaner < maxdetritos [
      if posto_maisprox != nobody [
        if bateria <= distancePosto and not charging [ gotoPosto airplane_color
```

```
bateria charging tempCharging]
        if bateria > distancePosto and not charging [ movimentacao_airplane
airplane_color bateria residuosCleaner charging]
      ]
    ]
    if residuosCleaner = maxdetritos [
      if contentor_maisprox != nobody and posto_maisprox != nobody [
        if airplane_color = cyan [ set full1 true ]
        if airplane_color = (violet + 2) [ set full2 true ]
        if bateria > (distancePosto + distanceContentor) and not charging [
gotoContentor airplane_color bateria residuosCleaner charging tempCharging full ]
        if bateria <= (distancePosto + distanceContentor) and not charging [
gotoPosto airplane_color bateria charging tempCharging]
      ]
    ]

    if xcor = [xcor] of posto_maisprox and ycor = [ycor] of posto_maisprox [
      if airplane_color = cyan [ set charging1 true ]
      if airplane_color = (violet + 2) [ set charging2 true ]
      if charging1 and airplane_color = cyan [ chargingBat cyan ]
      if charging2 and airplane_color = (violet + 2) [ chargingBat (violet + 2) ]
    ]
  ]
end

to movimentacao
  if shape = "person" [
    let adjacente-azul one-of neighbors4 with [pcolor = blue]

    if adjacente-azul != nobody [
      face adjacente-azul
      move-to adjacente-azul
    ]
    if adjacente-azul = nobody [
      let direction one-of ["forward" "backward" "right" "left"]
      if direction = "forward" [forward 1]
      if direction = "backward" [back 1]
      if direction = "right" [right 90 forward 1]
      if direction = "left" [left 90 forward 1]
    ]
  ]
end

to movimentacao_airplane [airplane_color bateria residuosCleaner charging]
  ask turtles with [shape = "airplane" and color = airplane_color] [
    if bateria > 0 and not charging [
      let adjacente one-of neighbors4 with [pcolor = (green + 2) or pcolor = yellow
or pcolor = (pink + 2)]

      if adjacente != nobody [
        face adjacente
        move-to adjacente
        if airplane_color = cyan [ set bateria1 bateria1 - 1 ]
        if airplane_color = (violet + 2) [ set bateria2 bateria2 - 1 ]
        set numMovimentos numMovimentos + 1
        if residuosCleaner < maxdetritos [ contadorResiduos airplane_color]
      ]
```

```
      if adjacente = nobody [
        let direction one-of ["forward" "backward" "right" "left"]
        if direction = "forward" [forward 1]
        if direction = "backward" [back 1]
        if direction = "right" [right 90 forward 1]
        if direction = "left" [left 90 forward 1]
        if airplane_color = cyan [ set bateria1 bateria1 - 1 ]
        if airplane_color = (violet + 2) [ set bateria2 bateria2 - 1 ]
        set numMovimentos numMovimentos + 1
      ]
    ]
  ]
end

to contadorResiduos [airplane_color]
  let current-patch patch-here

  if [pcolor] of current-patch != blue [
    if [pcolor] of current-patch = (green + 2) [ set residuosGreen residuosGreen + 1
]
    if [pcolor] of current-patch = yellow [ set residuosYellow residuosYellow + 1 ]
    if [pcolor] of current-patch = (pink + 2) [ set residuosPink residuosPink + 1 ]
    set residuosTotal residuosGreen + residuosYellow + residuosPink
    if airplane_color = cyan [ set residuosCleaner1 residuosCleaner1 + 1 ]
    if airplane_color = (violet + 2) [ set residuosCleaner2 residuosCleaner2 + 1 ]
    set pcolor blue
  ]
end

to gotoPosto [ airplane_color bateria charging tempCharging ]
  if shape = "airplane" and color = airplane_color [
    let posto_maisprox min-one-of turtles with [shape = "house"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]

    if bateria >= 0 [
      if xcor != [xcor] of posto_maisprox [
        if xcor > [xcor] of posto_maisprox [ set xcor xcor - 1 ]
        if xcor < [xcor] of posto_maisprox [ set xcor xcor + 1 ]
      ]
      if xcor = [xcor] of posto_maisprox and ycor != [ycor] of posto_maisprox [
        if ycor > [ycor] of posto_maisprox [ set ycor ycor - 1 ]
        if ycor < [ycor] of posto_maisprox [ set ycor ycor + 1 ]
      ]
      set bateria bateria - 1
      set numMovimentos numMovimentos + 1
    ]

    if xcor = [xcor] of posto_maisprox and ycor = [ycor] of posto_maisprox [
      if airplane_color = cyan [ set charging1 true ]
      if airplane_color = (violet + 2) [ set charging2 true ]
      if charging1 and airplane_color = cyan [ chargingBat cyan ]
      if charging2 and airplane_color = (violet + 2) [ chargingBat (violet + 2)]
    ]
  ]
end
```

```
to gotoContentor [ airplane_color bateria residuosCleaner charging tempCharging full
]
  if shape = "airplane" and color = airplane_color [
    if residuosCleaner = maxdetritos [
      let contentor_maisprox min-one-of turtles with [shape = "box"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
      let posto_maisprox min-one-of turtles with [shape = "house"] [ (abs (xcor -
[xcor] of myself)) + (abs (ycor - [ycor] of myself)) ]
      let distanceContentor distance contentor_maisprox
      let distancePosto distance posto_maisprox

      if bateria > (distanceContentor + distancePosto) [
        if xcor != [xcor] of contentor_maisprox [
          if xcor > [xcor] of contentor_maisprox [ set xcor xcor - 1 ]
          if xcor < [xcor] of contentor_maisprox [ set xcor xcor + 1 ]
        ]
        if ycor != [ycor] of contentor_maisprox [
          if ycor > [ycor] of contentor_maisprox [ set ycor ycor - 1 ]
          if ycor < [ycor] of contentor_maisprox [ set ycor ycor + 1 ]
        ]

        if distanceContentor = 0 [ descarregar airplane_color residuosCleaner full ]
        set bateria bateria - 1
        set numMovimentos numMovimentos + 1
      ]
    ]
  ]
end

to chargingBat [airplane_color]
  ask turtles with [shape = "airplane" and color = airplane_color] [
    if airplane_color = cyan [
      if bateria1 < maxbateria [
        set bateria1 bateria1 + (maxbateria / tempCharging1)
        if bateria1 >= maxbateria [
          set bateria1 maxbateria
          set charging1 false
        ]
      ]
    ]
    if airplane_color = (violet + 2) [
      if bateria2 < maxbateria [
        set bateria2 bateria2 + (maxbateria / tempCharging2)
        if bateria2 >= maxbateria [
          set bateria2 maxbateria
          set charging2 false
        ]
      ]
    ]
  ]
end
```

```
to descarregar [ airplane_color residuosCleaner full ]
  ask turtles with [shape = "airplane" and color = airplane_color] [
    if airplane_color = cyan [
      if residuosCleaner1 = maxdetritos [
        set residuosCleaner1 0
        set full1 false
      ]
    ]
    if airplane_color = (violet + 2) [
      if residuosCleaner2 = maxdetritos [
        set residuosCleaner2 0
        set full2 false
      ]
    ]
  ]
end
```