

**INTELIGÊNCIA ARTIFICIAL**

**2024/2025**

**Professor**

**Paulo Oliveira**

**Eduardo Pires**

**RELATÓRIO**

aspiradores em netlogo

trabalho pratico 1 -> agentes

**79881 - David Fidalgo**

**78800 - Tiago Carvalho**

**2024/2025**

Indíce

[1. Resumo 4](#_Toc180428646)

[2. Introdução 5](#_Toc180428647)

[3. Enquadramento Teórico 6](#_Toc180428648)

[4. Robot1 7](#_Toc180428649)

[a. Interface 7](#_Toc180428650)

[b. Adversidades ultrapassadas durante o projeto 8](#_Toc180428651)

[c. Simulação 1 12](#_Toc180428652)

[5. Robot 2 15](#_Toc180428653)

[a. Interface 15](#_Toc180428654)

[b. Implementações para aproximação com a Realidade 16](#_Toc180428655)

[c. Simulação 17](#_Toc180428656)

[6. Resultados 19](#_Toc180428657)

[7. Conclusão 19](#_Toc180428658)

[8. Bibliografia 20](#_Toc180428659)

[Anexo Robot 1 21](#_Toc180428660)

[Anexo Robot 2 31](#_Toc180428661)

Indíce de Figuras

[Figura 1-> Robot: Interface NETLOGO 7](#_Toc180428664)

[Figura 2-> Robot 1: Gráfico Contaminação Vs Limpeza 8](#_Toc180428665)

[Figura 3-> Robot 1: Algoritmo de movimentação cleaner 9](#_Toc180428666)

[Figura 4-> Robot 1: Função Modo\_Map 10](#_Toc180428667)

[Figura 5-> Robot 1: Algoritmo de procura o deposito mais próximo 10](#_Toc180428668)

[Figura 6-> Robot 1: Remover deposito já não existente 11](#_Toc180428669)

[Figura 7-> Robot 1: Verificação e deslocação para posto de carregamento 11](#_Toc180428670)

[Figura 8-> Robot1: Simulação - Setup 12](#_Toc180428671)

[Figura 9-> Robot 1: Simulação - Go-N 12](#_Toc180428672)

[Figura 10-> Robot 1: Simulação - Mapping Efetuado 12](#_Toc180428673)

[Figura 11-> Robot1: Simulação - Desaparecimento de Contentor 13](#_Toc180428674)

[Figura 12-> Robot 1: Simulação - Direção Posto Carregamento 14](#_Toc180428675)

[Figura 13-> Robot 1: Simulação – Recuperação do local 14](#_Toc180428676)

[Figura 14-> Robot 2: Interface 15](#_Toc180428677)

[Figura 15-> Robot 2: Função Config\_Battery 16](#_Toc180428678)

[Figura 16-> Robot 2: Comportamento Cleaner perante poluição 16](#_Toc180428679)

[Figura 17-> Robot 2: Simulação – Setup 17](#_Toc180428680)

[Figura 18-> Robot 2: Simulação – Obstáculos 18](#_Toc180428681)

[Figura 19-> Robot 2: Simulação - Limpeza Patch 18](#_Toc180428682)

# Resumo

Este trabalho visa descrever o desenvolvimento de uma simulação em NetLogo para estudar a interação entre um agente de limpeza e três agentes poluentes.

Numa primeira fase, foi criado um robô que mapeia e limpa o ambiente, enfrentando desafios como a localização de contentores e a gestão da bateria, além de limpar o ambiente de forma eficiente.

Numa segunda fase, foram implementadas melhorias para aproximar a simulação à realidade. O robot apresenta ajustes energéticos e vários comportamentos conforme o tipo de lixo e obstáculos encontrados.

As simulações mostraram a eficiência de ambos os robôs em diferentes cenários, superando desafios técnicos e apresentado a eficácia das estratégias desenvolvidas para a economia da bateria e melhorar a limpeza em ambientes poluídos.

# Introdução

A automação de tarefas de carácter diário tem evoluído significativamente com o uso de máquinas e robôs, especialmente em áreas que exigem contante manutenção, como a limpeza de espaços. Neste contexto, a simulação de sistemas multiagente surge como uma ferramenta eficaz para estudar e entender a interação entre diferentes agentes que atuam em um mesmo ambiente.

Este trabalho prático tem como objetivo desenvolver um modelo de simulação, utilizando o NetLogo, que reproduza o comportamento de agentes responsáveis pela limpeza e poluição de um espaço.

A proposta fornecida pelos docentes da cadeira de Inteligência Artificial baseia-se em um cenário simplificado onde um agente de limpeza interage com três agentes poluidores. A simulação visa representar o processo de degradação do espaço causado pelos Polluters e a restauração do espaço realizado pelo Cleaner.

A simulação, não só auxilia na análise das iterações entre agentes, mas também possibilita a análise e experimentação de várias estratégias para melhorar a eficiência do processo de limpeza em um ambiente poluído.

# Enquadramento Teórico

No contexto deste trabalho, foi utilizado a plataforma NetLogo para simular o comportamento entre agentes de limpeza (Cleaners) e agentes poluentes (Polluters) em um determinado ambiente.

**Ambiente:**

* Superfície composta por células (patches);
* Todas as células estão limpas e têm a mesma cor;
* É um espaço físico onde os agentes movimentam e interagem entre eles.

***Cleaners*:**

* Localizado inicialmente no canto inferior esquerdo do ambiente;
* Perde uma unidade de energia por movimento;
* Recarrega a bateria no posto de carregamento, localizado no ponto de partida;
* Limpa células com resíduos e armazena os detritos;
* Capacidade de transporte de detritos ajustável;
* Descarrega detritos em contentores (*Containers*), cuja quantidade é ajustável e localização são aleatórios.

***Polluters*:**

* Três agentes poluidores que entram no ambiente em pontos aleatórios;
* Movimentam-se de forma aleatória;
* Depositam resíduos em células limpas, mudando a cor dos *patches* conforme o tipo de resíduo;
* A decisão de depositar resíduos é determinada por uma função probabilísticas, que são ajustáveis para cada *Polluter*.

**Interação entre *Cleaner* e *Polluters*:**

* *Cleaner* limpa as células poluídas pelos *Polluters*;
* Os *Polluters* continuam a depositar resíduos enquanto se movimentam;
* O *Cleaner* deve gerir a sua energia e a capacidade de armazenamento enquanto limpa o ambiente;
* A eficiência do *Cleaner* depende da sua capacidade de localizar contentores e recarregar a bateria sem interrupções.

# Robot1

## Interface

Uma imagem com texto, software, Ícone de computador, Software de multimédia

Descrição gerada automaticamente

Figura 1-> Robot: Interface NETLOGO

* **Setup:** Reset ao mundo;
* **Go\_Once:** Os agentes efetuam um movimento;
* **Go\_N:** Os agentes efetuam N movimentos, sendo N adaptável na entrada;
* **Go:** Os agentes realizam de uma forma continua os seus movimentos;
* **Cleaner\_max\_battery:** Bateria máxima que o cleaner contém;
* **Cleaner\_max\_capacity:** Capacidade máxima que o cleaner contém;
* **Cleaner\_tempo\_carregamento:** Ticks necessários para carregar o cleaner;
* **Polluter\_1\_prob\_sujar, Polluter\_2\_prob\_sujar, Polluter\_3\_prob\_sujar:** A probabilidade de cada polluter de poluir;
* **Cleaner – Bateria Restante:** Bateria atual do cleaner;
* **Cleaner – Capacidade:** Capacidade atual do cleaner;
* **Num\_depositos:** Numero de depósitos existente no mundo;
* **Contaminação Vs Limpeza:** Grafico que mostra o evoluir a contaminação/limpeza do mundo. Exemplo:

Uma imagem com texto, diagrama, Gráfico, file

Descrição gerada automaticamente

Figura 2-> Robot 1: Gráfico Contaminação Vs Limpeza

## Adversidades ultrapassadas durante o projeto

1. Localização do Robot (Cleaner)

Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente

Figura 3-> Robot 1: Algoritmo de movimentação cleaner

Para detetar a sua localização o robô recorre ao comando ***patch-here***, que devolve ao nosso agente *cleaner* (robô aspirador) o *patch* onde este se encontra.

Visto que esta abordagem obedece à regra de que o *cleaner* só pode ter perceção de apenas um *patch* à sua volta (**comando *neighbors***), pareceu-nos a mais adequada.

O nosso agente *cleaner* usa os seguintes comandos chave para determinar onde se encontra e quem são os seus vizinhos:

**Comandos chave:**

* patch-here
* patch-ahead
* patch-at-heading-and-distance
* neighbors

1. Localização dos Contentores:

* Numa fase inicial, o cleaner vai ter o modo mapeamento ativado, que serve para localizar todos os contentores existentes, que são todos os *patches-here* que se encontrem com o *pcolor = blue*;

**Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente**

Figura 4-> Robot 1: Função Modo\_Map

* Quando chega ao limite superior, ele começa o processo de limpeza;
* Quando o cleaner precisar de descarregar os detritos, ele calcula, na lista **depósitos**, procura as coordenadas mais próximas à localização atual do cleaner.

**Uma imagem com texto, captura de ecrã, Tipo de letra, número

Descrição gerada automaticamente**

Figura 5-> Robot 1: Algoritmo de procura o deposito mais próximo

* Caso o utilizador queira mudar o número de contentores, o cleaner continua a mapear caso encontre um *patch* de cor *blue.* Caso o cleaner se desloque a um sítio que deveria conter um deposito e esse *patch* não tiver com a cor devida, essa coordenada é removida da lista de depósitos.

**Uma imagem com texto, captura de ecrã, Tipo de letra, file

Descrição gerada automaticamente**

Figura 6-> Robot 1: Remover deposito já não existente

1. Posto de Carregamento

* O posto de carregamento encontra se no canto inferior esquerdo;
* O Cleaner só se desloca para o posto de carregamento só quando faltar 50 movimentos;

**Uma imagem com texto, Tipo de letra, file, captura de ecrã

Descrição gerada automaticamente**

Figura 7-> Robot 1: Verificação e deslocação para posto de carregamento

* O Clenear, após o número de ticks de carregamento, volta para o local onde estava a limpar.

## Simulação 1

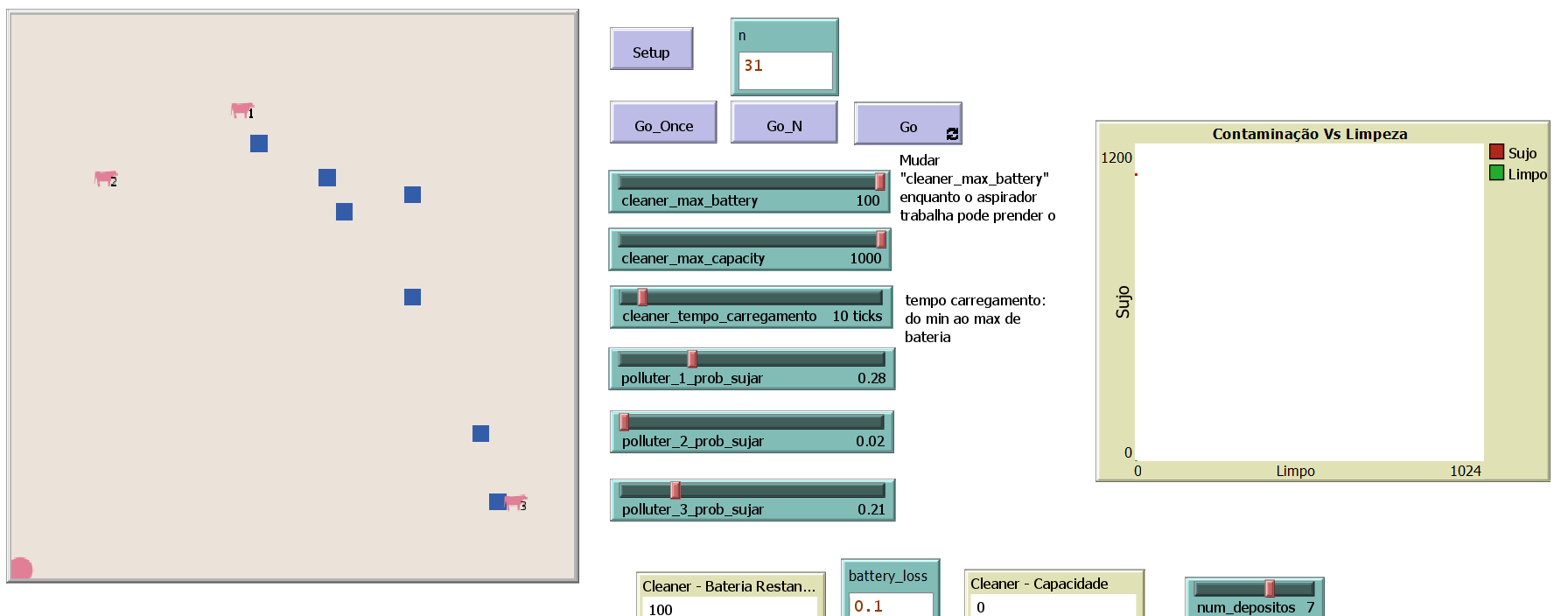


Figura 8-> Robot1: Simulação - Setup

**Uma imagem com diagrama, texto, file, captura de ecrã

Descrição gerada automaticamente**

Figura 9-> Robot 1: Simulação - Go-N

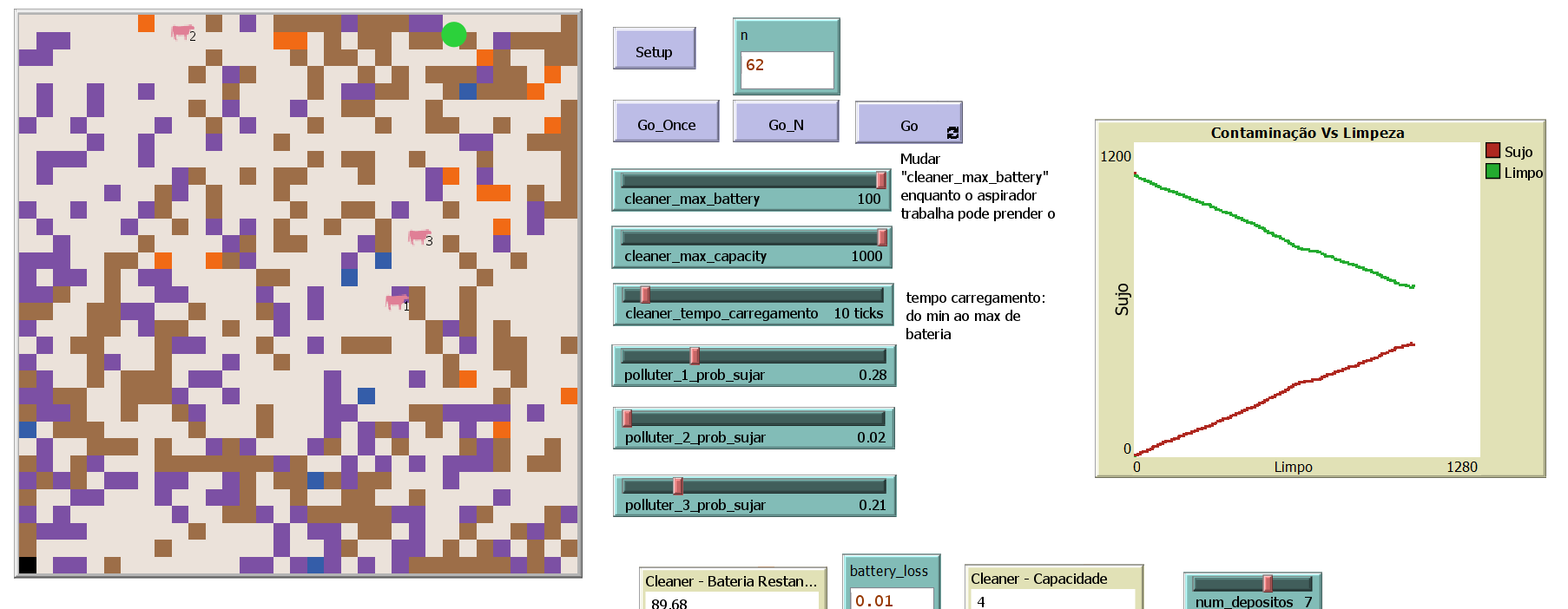
****

Figura 10-> Robot 1: Simulação - Mapping Efetuado

* Após mapear os contentores, começa a limpar o ambiente.

Uma imagem com padrão, captura de ecrã, quadrado, Saturação de cores

Descrição gerada automaticamente

Figura 11-> Robot1: Simulação - Desaparecimento de Contentor

* Caso o utilizador tenha alterado o número de contentores, o *robot* só reconhece as alterações quando passar no *patch* onde surgiu/desapareceu um depósito.
* Caso o *Robot* vá lá e encontre o espaço vazio (ou seja, a cor seja diferente de *blue*), ele vai apagar da sua memória as coordenadas onde estaria o contentor.
* Quando o *robot* vá a um espaço onde esteja um contentor e ainda não estiver na memória, o robot guarda na memória as coordenadas do *patch* onde se encontra.

Uma imagem com diagrama, file, texto, captura de ecrã

Descrição gerada automaticamente

Figura 12-> Robot 1: Simulação - Direção Posto Carregamento

* Quando faltar 50 movimentos (*battery\_loss* \* 50 movimentos), o *Robot* direciona para o posto de carregamento, evitando assim uma possível paragem devido à falta de bateria.
* O Robot guarda o local onde ficou antes de deslocar-se ao posto de carregamento.

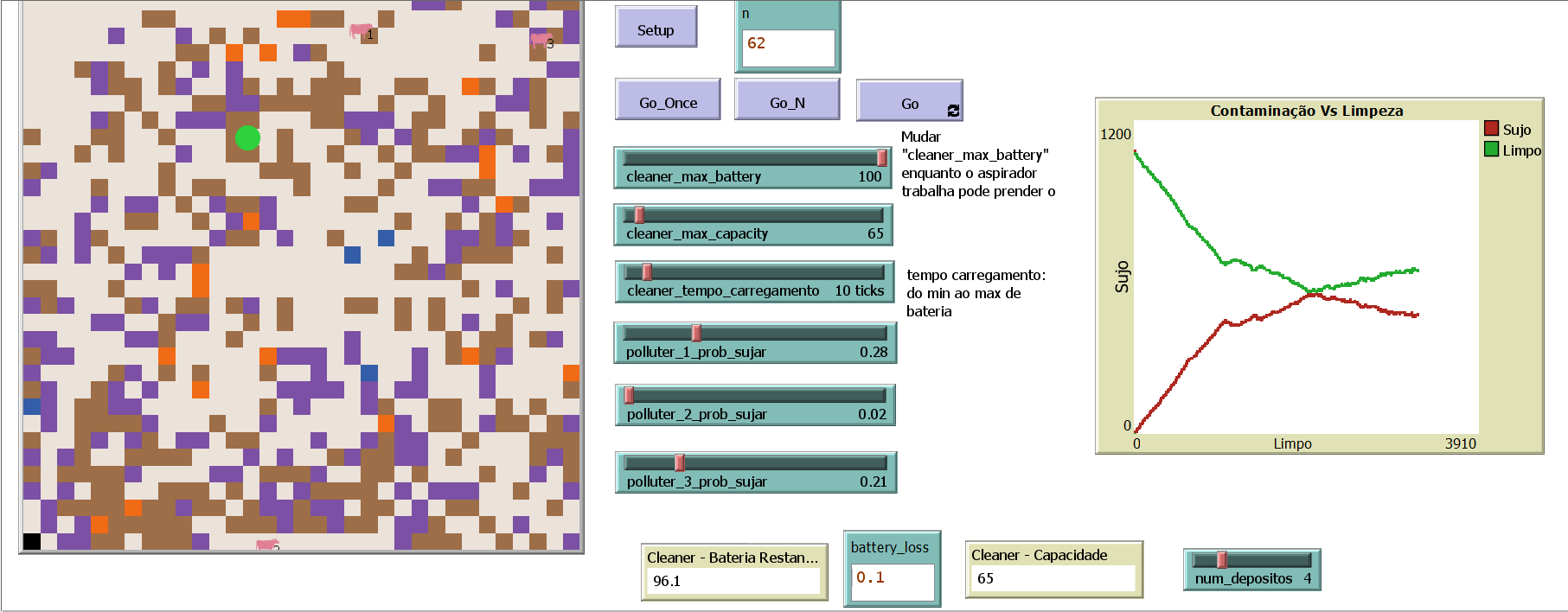


Figura 13-> Robot 1: Simulação – Recuperação do local

* Após o carregamento, o Robot desloca-se para o local onde se encontrava antes de deslocar para o posto de carregamento.

# Robot 2

## Interface

Uma imagem com texto, software, Software de multimédia, Ícone de computador

Descrição gerada automaticamente

Figura -> Robot 2: Interface

Nesta fase, mantivemos a base da fase transata, aplicando melhorias, aproximando o ambiente dos agentes o mais próximo da realidade.

* **Cleaner\_Modo:** O cleaner vai ter 3 modos de funcionamento: Eco Mode, Medium Mode e Full Mode;
* **Potencia Cleaner (em W):** Valor, em Watts, da Potencia do Cleaner;
* **Cleaner\_capacity\_battery:** Capacidade da bateria, em miliampere-hora, do Cleaner;
* **Cleaner\_tensao\_battery:** Tensão, em Volts, da bateria do Cleaner;
* **Consumo Cleaner:** Consumo, por tick, a cada movimento do Cleaner;
* **Cleaner\_max\_capacity (Alteração):** Capacidade máxima, em ml, do Cleaner;
* **Num\_obstaculos:** Numero de obstáculos do ambiente.

## Implementações para aproximação com a Realidade

1. **Inserção de modelo de bateria ao Cleaner**

**Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente**

Figura 15-> Robot 2: Função Config\_Battery

* Cada um dos modos vai ter diferentes valores de potência;
* Conforme o Modo do Cleaner, da capacidade da bateria e da tensão, é calculado o consumo da bateria por segundo (ou tick).

1. **Diferentes tons de poluição**

**Uma imagem com texto, software, Tipo de letra, captura de ecrã

Descrição gerada automaticamente**

Figura 16-> Robot 2: Comportamento Cleaner perante poluição

* Os *polluters* vão inserir ‘lixos’ de diferentes tons de uma determinada cor;
* O *Cleaner* vai ter diferentes comportamentos, dependendo do modo em que se encontrar:
  + **Eco Mode:**
    - Os lixos considerados “leves”, o cleaner necessita de 1 tick para limpar o patch em questão;
    - Os livros considerados “normais” e “pesados”, o *cleaner* vai aumentar o valor do *pcolor* até o lixo que se encontrar no *patch* for considerado “leve”.
  + **Medium Mode:**
    - Os lixos considerados “leves” e “normais”, o *cleaner* só necessita de 1 *tick* para limpar o *patch* em questão;
    - Os lixos considerados “pesados”, o cleaner vai aumentar o valor do pcolor até o cleaner encontrar se em condições de limpar facilmente o lixo.
  + **Full Mode:**
    - Neste modo, o *cleaner* vai apenas demorar 1 *tick* a limpar o *patch* que se encontra, com todo o tipo de lixo.
* Cada lixo tem diferente peso.

## Simulação

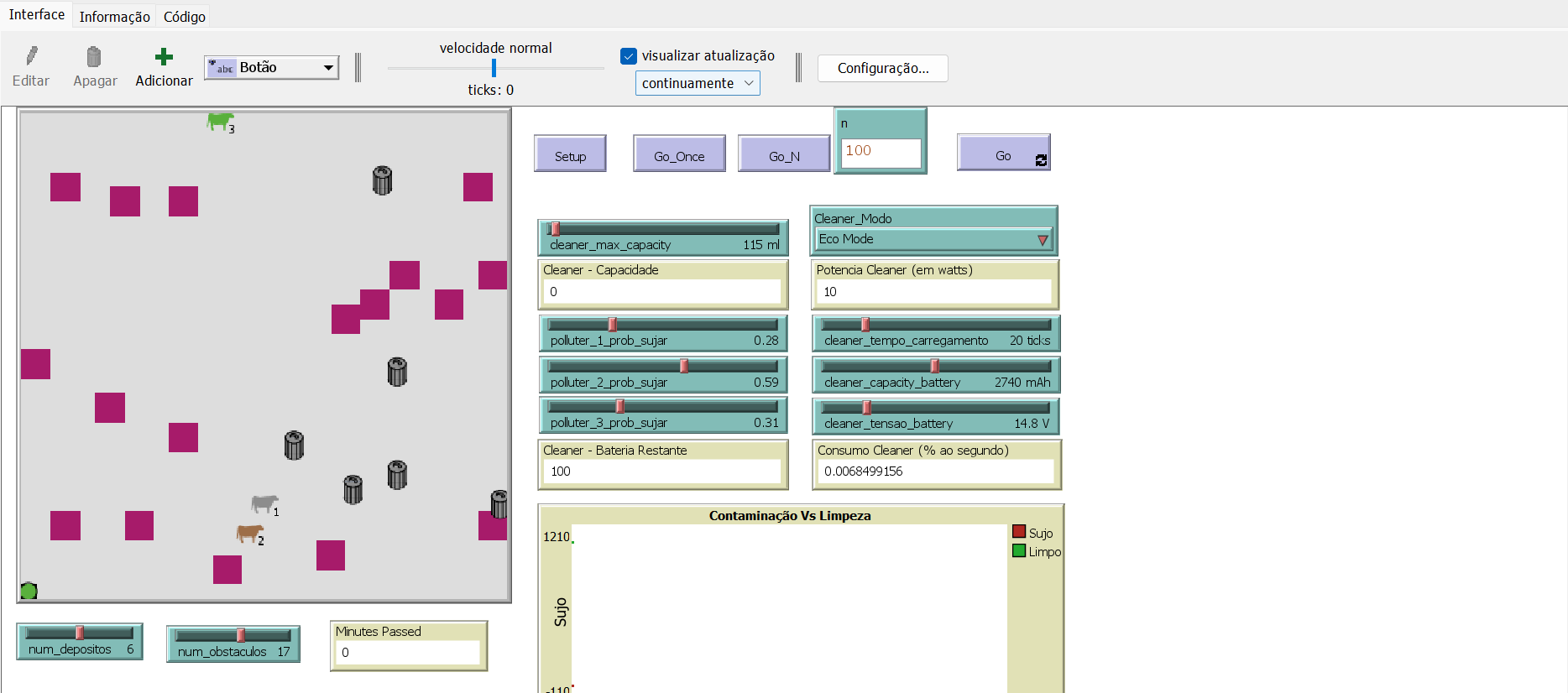


Figura 17-> Robot 2: Simulação – Setup

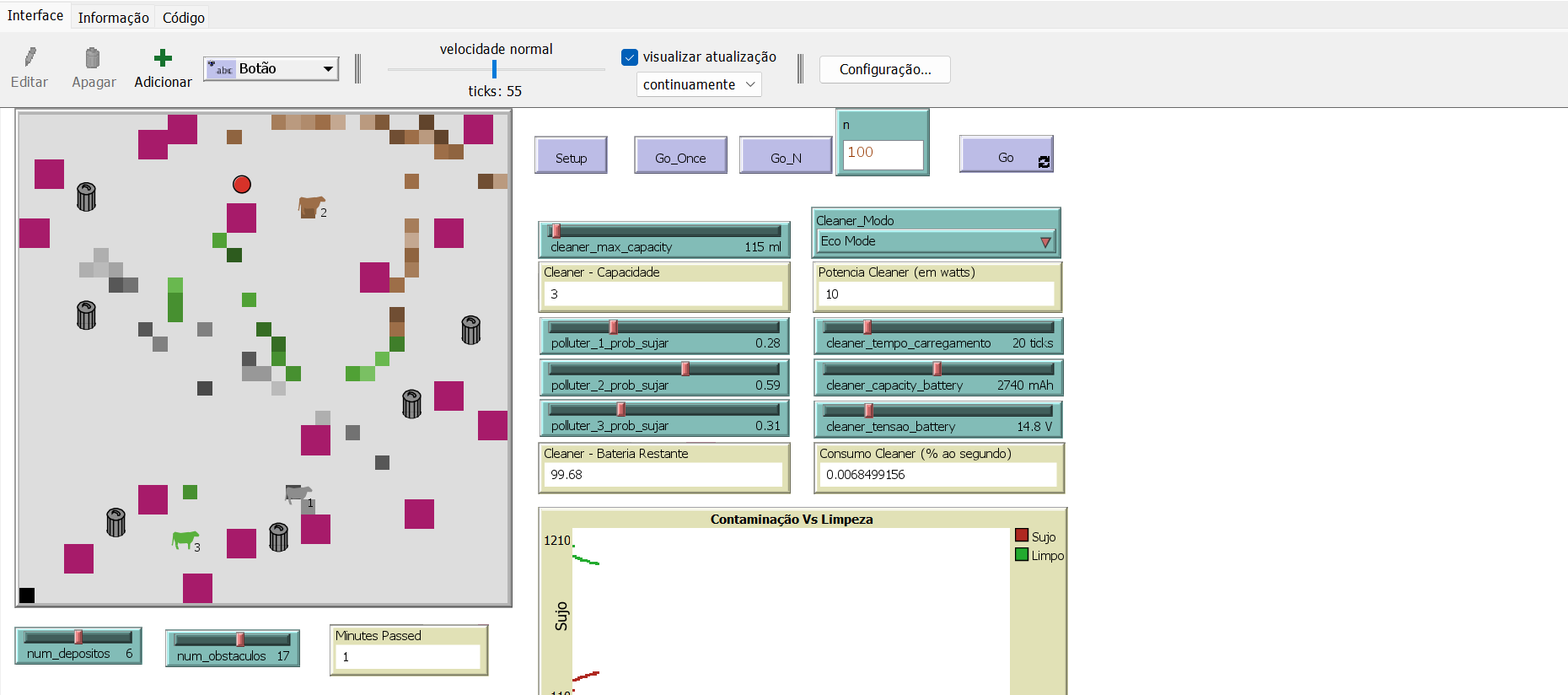


Figura 18-> Robot 2: Simulação – Obstáculos

* Caso esteja em modo limpeza, se o *robot* encontre à sua frente um obstáculo, vai escolher uma direção aleatória e continua a limpeza.
* Caso o *robot* esteja a dirigir-se para uma determinada localização *(last\_cleaning\_location, posto\_carregamento, depósitos)* ele procura contornar o obstáculo escolhendo uma direção (esquerda ou direita).

Uma imagem com software, Software gráfico, texto, Software de multimédia

Descrição gerada automaticamente

Figura 19-> Robot 2: Simulação - Limpeza Patch

* Quando o *robot* encontra um *patch* de uma intensidade de cor alta e ao mesmo tempo encontrar se nos modos mais económicos, ele demora alguns *ticks* a mais e a cada um tira intensidade à cor.

# Resultados

Na fase 1, o *Robot* não apresentou problemas, executando todas as tarefas conforme especificado no protocolo. O *Robot* conseguiu mapear e limpar o ambiente de forma eficiente, localizando os contentores e gerir a bateria sem dificuldades. A várias simulações feitas demonstraram que o robô cumpriu todos os requisitos estabelecidos, validando a eficácia dos algoritmos implementados para a navegação e limpeza do espaço.

Já na fase 2, foi possível implementar diferentes modelos de bateria, ajustando a sua capacidade, tensão a potência do *Robot.* O consumo de energia do robô aproxima-se da realidade, refletindo um comportamento realista em termos de eficiência e durabilidade da bateria. Em 5 simulações, o Robot opera por uma média de 80 horas, sem necessitar de intervenção humana para recarregar, sendo essas ocasiões este tido ficado preso em obstáculos e consequentemente tendo perdido a carga.

# Conclusão

Este trabalho prático conseguiu modelar com sucesso a interação entre agentes responsáveis pela poluição (Polluters) e pela limpeza (Cleaner) de um espaço, utilizando NetLogo. Através de várias simulações, foi possível, não só reproduzir o comportamento esperado dos agentes, mas também ultrapassar desafios técnicos relacionados à localização e eficiência dos mesmos.

A integração de modos de funcionamento do Cleaner, o mapeamento de depósitos e o desenvolvimento de estratégias para economizar bateria mostram uma aproximação da simulação à realidade. O Cleaner tem um desempenho ajustável conforme diferentes modos de operação, variando o consumo de energia e a eficiência com base no tipo de lixo e obstáculos encontrados.

Em suma, estamos satisfeitos pelo trabalho elaborado, mesmo com a ideia de que haveria sempre espaço para melhorias.

# Bibliografia

* <https://youtu.be/O7ozptNs1FY?si=MSywmYDwbmLPsnCb>

(consultado a 7/10/2024)

* <https://stackoverflow.com/questions/36019543/turtles-move-to-nearest-patch-of-a-certain-color-how-can-this-process-be-sped> (consultado a 12/10/2024)
* OpenAI. (2024). *ChatGPT: AI Language Model*. <https://www.openai.com>
* WILENSKY, U. *NetLogo User Manual* – NetLogo Dictionary. Center for Connected Learning and Computer-Based Modeling, Northwestern University, 2023. Disponível em: https://ccl.northwestern.edu/netlogo/docs/dictionary.html.

# Anexo Robot 1

;; Definição de variáveis globais e raças

globals [cor\_chao posto\_carregamento depositos tick\_bug\_fix]

breed [cleaners cleaner]

breed [polluters polluter]

cleaners-own [battery capacity recharge\_time last\_cleaning\_location cleaner\_map]

polluters-own [prob\_sujar polluter\_corlixo]

;; Modo mapear depósitos (zig-zag até percorrer todo o espaço da sala)

;; NÃO ASPIRA

to Modo\_Map

if xcor = min-pxcor [

ifelse patch-ahead 1.5 = nobody [ ; Se é parede

set heading 0

][

set heading 90

]

]

if xcor = max-pxcor [

ifelse patch-ahead 1.5 = nobody [ ; Se é parede

set heading 0

][

set heading -90

]

]

if round xcor = max-pxcor and round ycor = max-pycor [

set cleaner\_map FALSE

]

if [pcolor] of patch-here = blue [

let coordenadas\_depositos (list round xcor round ycor)

set depositos lput coordenadas\_depositos depositos

]

end

;; Setup: Limpa o ambiente, cria e introduz os agentes, e faz o reset do tempo.

to setup

clear-all

reset-ticks

set tick\_bug\_fix 10000 ;; Para evitar loops infinitos

set cor\_chao 39

ask patches [ set pcolor cor\_chao ]

set posto\_carregamento [-16 -16]

ask patch item 0 posto\_carregamento item 1 posto\_carregamento [ set pcolor black ]

;; Criação de depósitos

let i 1

ask patches [

set i count patches with [pcolor = blue]

if pcolor = cor\_chao and i < num\_depositos and one-of [pcolor] of neighbors4 != blue [

set pcolor blue

]

]

;; Criação de agentes

create-cleaners 1

create-polluters 3

;; Configuração dos cleaners

ask cleaners [

set depositos []

set shape "circle"

set size 1.5

setxy item 0 posto\_carregamento item 1 posto\_carregamento

set battery cleaner\_max\_battery

set capacity 0

set last\_cleaning\_location [0 0]

set cleaner\_map TRUE

set heading 90

]

;; Configuração dos polluters

ask polluters [

set shape "cow"

set size 1.5

set color pink

set label-color black

set label who

setxy random-pxcor random-pycor

]

end

;; go\_once: Permite que os agentes circulem no mundo de forma aleatória (um só tick).

to go\_once

;; Atualiza probabilidades dos sliders

ask polluter 1 [

set prob\_sujar polluter\_1\_prob\_sujar

set polluter\_corlixo 33

]

ask polluter 2 [

set prob\_sujar polluter\_2\_prob\_sujar

set polluter\_corlixo 53

]

ask polluter 3 [

set prob\_sujar polluter\_3\_prob\_sujar

set polluter\_corlixo 3

]

;; Atualiza depósitos

let i 1

ask patches [

set i count patches with [pcolor = blue]

if pcolor = cor\_chao and i < num\_depositos [

set pcolor blue

]

if i > num\_depositos and pcolor = blue [

set pcolor cor\_chao

]

]

;; Ações dos cleaners

ask cleaners [

let cleaner\_atual who

ask patch-here [

if pcolor != blue [

let coordenadas\_depositos (list round pxcor round pycor)

if member? coordenadas\_depositos depositos [

set depositos remove coordenadas\_depositos depositos

]

]

]

;; Modo carregar bateria

if battery > cleaner\_max\_battery [ set battery cleaner\_max\_battery ]

ask patch-here [

ifelse pcolor = black and ([battery] of cleaner cleaner\_atual < cleaner\_max\_battery) [

ask cleaners [

let battery\_a\_cargar battery + (cleaner\_max\_battery / cleaner\_tempo\_carregamento)

ifelse battery\_a\_cargar > cleaner\_max\_battery [

set battery cleaner\_max\_battery

][

set battery battery + (cleaner\_max\_battery / cleaner\_tempo\_carregamento)

]

]

][

;; Verificar a bateria e definir movimentação

ask cleaners [

ifelse battery <= 50 \* battery\_loss [

if last\_cleaning\_location = [0 0] [

set last\_cleaning\_location (list round xcor round ycor)

if ticks > tick\_bug\_fix [

set last\_cleaning\_location [-15 -15]

set tick\_bug\_fix tick\_bug\_fix + 10000

]

]

facexy item 0 posto\_carregamento item 1 posto\_carregamento

][

ifelse capacity >= cleaner\_max\_capacity [

;; Modo ir depositar resíduos

ifelse [pcolor] of patch-here = blue [

let coordenadas\_depositos (list round xcor round ycor)

if not member? coordenadas\_depositos depositos [

set depositos lput coordenadas\_depositos depositos

]

set capacity 0

][

let menor-distancia (max-pxcor \* 3)

let target-patch one-of depositos

let j 0

foreach depositos [

let coordenada item j depositos

let x item 0 coordenada

let y item 1 coordenada

let distancia-atual distancexy x y

if distancia-atual < menor-distancia [

set menor-distancia distancia-atual

set target-patch patch x y

]

set j j + 1

]

if target-patch != nobody [

ask cleaner cleaner\_atual [ face target-patch ]

]

]

][

;; Movimento para limpar

ifelse last\_cleaning\_location != [0 0] [

facexy item 0 last\_cleaning\_location item 1 last\_cleaning\_location

][

if patch-ahead 1 = nobody and cleaner\_map = FALSE [set heading random 360]

]

]

]

if cleaner\_map = FALSE [

fd 1

set battery battery - battery\_loss

if last\_cleaning\_location = (list round xcor round ycor) or last\_cleaning\_location = [-15 -15] [

set last\_cleaning\_location [0 0]

]

if capacity < cleaner\_max\_capacity [

ask patch-here [

if pcolor != cor\_chao and pcolor != black and pcolor != blue [

set pcolor cor\_chao

ask cleaners [ set capacity capacity + 1 ]

]

]

]

]

if cleaner\_map = TRUE [

Modo\_Map

fd 1

if last\_cleaning\_location = (list round xcor round ycor) or last\_cleaning\_location = [-15 -15] [

set last\_cleaning\_location [0 0]

]

set battery battery - 0.01

]

]

]

]

]

;; Ações dos polluters

ask polluter 1 [

if patch-ahead 1 = nobody [ set heading random 360 ]

fd 1

if (random 100 < prob\_sujar \* 100) [

ask patch-here [

if pcolor = cor\_chao [ set pcolor [polluter\_corlixo] of polluter 1 ]

]

]

]

ask polluter 2 [

if patch-ahead 1 = nobody [ set heading random 360 ]

fd 1

if (random 100 < prob\_sujar \* 100) [

ask patch-here [

if pcolor = cor\_chao [ set pcolor [polluter\_corlixo] of polluter 2 ]

]

]

]

ask polluter 3 [

if patch-ahead 1 = nobody [ set heading random 360 ]

fd 1

if (random 100 < prob\_sujar \* 100) [

ask patch-here [

if pcolor = cor\_chao [ set pcolor [polluter\_corlixo] of polluter 3 ]

]

]

]

tick

end

;; go\_n: Executa o processo 'go\_once' repetidamente n vezes

to go\_n

repeat n [ go\_once ]

end

# Anexo Robot 2

;; globals

globals[cor\_chao cor\_objetos posto\_carregamento depositos lista\_objetos tick\_bug\_fix tipo\_lixo num\_polluters cleaner\_max\_battery eco med turbo]

breed[cleaners cleaner]

breed[polluters polluter]

breed[containers container]

breed[obstacles obstacle]

cleaners-own[battery capacity recharge\_time last\_cleaning\_location cleaner\_consumption\_battery cleaner\_potencia\_battery cleaner\_stop]

polluters-own[prob\_sujar]

to Config\_Battery

set eco "Eco Mode"

set med "Medium Mode"

set turbo "Full Mode"

ask cleaners [

if Cleaner\_Modo = eco [

;; comportamento para o modo Eco

set cleaner\_potencia\_battery 10

]

if Cleaner\_Modo = med [

;; comportamento para o modo Médio

set cleaner\_potencia\_battery 30

]

if Cleaner\_Modo = turbo [

;; comportamento para o modo Full

set cleaner\_potencia\_battery 50

]

let cleaner\_corrente\_battery (cleaner\_potencia\_battery / cleaner\_tensao\_battery) \* 1000 ;; isto é a corrente em mA

let cleaner\_ma\_segundo\_battery cleaner\_corrente\_battery / 3600 ;; aqui o gasto de mA por segundo

set cleaner\_consumption\_battery (cleaner\_ma\_segundo\_battery / cleaner\_capacity\_battery) \* 100

]

end

; setup, cujo programa permita: limpar o ambiente; criar e introduzir no mundo os agentes e fazer o reset do tempo.

to setup

clear-all

reset-ticks

set tick\_bug\_fix 10000 ; de 10000 em 10000 ticks reset da last\_cleaning location senao ele pode ficar preso num loop de ir de ponta a ponta

set tipo\_lixo (list (range 2.5 7.5 0.5)(range 32.5 37.5 0.5)(range 52.5 57.5 0.5))

set cor\_chao 8.5

set cor\_objetos magenta

set cleaner\_max\_battery 100

ask patches [

set pcolor cor\_chao

]

set posto\_carregamento [-16 -16]

ask patch item 0 posto\_carregamento item 1 posto\_carregamento [;; caso mude tamanho do world

set pcolor black

]

;; padrões do dicionário do netlogo

create-cleaners 1

create-polluters 3

ask cleaners [

set shape "vaccum"

set size 2.5

let canto\_inferior\_esquerdo (list min-pxcor min-pycor) ; origem do cleaner (posto de carregamento)

setxy (item 0 canto\_inferior\_esquerdo ) (item 1 canto\_inferior\_esquerdo) ;; criado no canto inferior esquerdo

set battery 100

set capacity 0

set last\_cleaning\_location [0 0]

set cleaner\_stop 0

]

Config\_Battery

ask polluters [

set shape "cow"

set size 2

set color white

set label-color black

set label who

setxy random-pxcor random-pycor

]

ask polluter 1 [ set color 5 ]

ask polluter 2 [ set color 35 ]

ask polluter 3 [ set color 55 ]

;; criacao de depositos

let i 1

set depositos []

ask patches [

set i count patches with [pcolor = blue]

if pcolor = cor\_chao and i < num\_depositos [;; evitar depositos juntos (fica confuso)

if all? neighbors4 [pcolor = cor\_chao] [

set pcolor blue

sprout-containers 1

set depositos fput (list pxcor pycor) depositos

]

]

]

ask containers [

set shape "garbage can"

set size 2

set color grey

]

;; criacao obstaculos

set i 0

set lista\_objetos []

ask patches [

set i count patches with [pcolor = cor\_objetos]

if pcolor = cor\_chao and i < num\_obstaculos \* 4 and pxcor < 16 and pycor > -16 [

let object1 (list (list pxcor pycor) (list (pxcor + 1) pycor) (list pxcor (pycor - 1)) (list (pxcor + 1) (pycor - 1)))

let pode\_gerar true

foreach object1 [coord ->

let x item 0 coord

let y item 1 coord

if [pcolor] of patch x y != cor\_chao [

set pode\_gerar false

]

]

if pode\_gerar [

foreach object1 [coord ->

let x first coord

let y last coord

ask patch x y [

set pcolor cor\_objetos ;; Replace cor\_objetos with the desired color

]

]

set lista\_objetos fput object1 lista\_objetos

]

]

]

end

; go\_once, cujo programa permita: que os agentes circulem no mundo de forma aleatória (um só tick);

to go\_once

;; atualizar probabilidades dos sliders

ask polluters [

if color = 5 [set prob\_sujar polluter\_1\_prob\_sujar]

if color = 35 [set prob\_sujar polluter\_2\_prob\_sujar]

if color = 55 [set prob\_sujar polluter\_3\_prob\_sujar]

]

Config\_Battery

;; atualizar depositos

let i 1

ask patches [

set i count patches with [pcolor = blue]

if pcolor = cor\_chao and i < num\_depositos [

set pcolor blue

set depositos lput (list pxcor pycor) depositos

sprout-containers 1 [

set shape "garbage can"

set size 2

set color grey

]

]

if i > num\_depositos and pcolor = blue [

set pcolor cor\_chao

ask containers-here [die]

set depositos remove (list pxcor pycor) depositos

]

]

;; ações do cleaner

ask cleaners [

let cleaner\_atual who ;; para permitir mais cleaners e usar o codigo abaixo

;; modo carregar

if battery > cleaner\_max\_battery [ set battery cleaner\_max\_battery ]

ask patch-here [

ifelse pcolor = black and ([battery] of cleaner cleaner\_atual < cleaner\_max\_battery) [

ask cleaners [

let battery\_a\_cargar battery + (cleaner\_max\_battery / cleaner\_tempo\_carregamento) ;; battery\_a\_cargar é o cálculo de quanto a bateria vai carregar em um tick

ifelse battery\_a\_cargar > cleaner\_max\_battery [

set battery cleaner\_max\_battery

][

set battery battery + (cleaner\_max\_battery / cleaner\_tempo\_carregamento) ;; por cada tick para 100 max é tipo: 100/10 = 10% a cada tick

]

]

][

;; HEADINGS E AS SUAS CONDICOES

;; 1º verificar a bateria (modelo Robot1 dirige-se ao posto quando chega a uma certa percentagem)

ask cleaners [

ifelse battery <= 100 \* cleaner\_consumption\_battery [ ;; dirigir ao posto de carregamento quando so faltarem 50 movimentos

if last\_cleaning\_location = [0 0] [ ;; aspirador guarda sitio onde estava a aspirar até ter de ir carregar bateria

set last\_cleaning\_location (list round xcor round ycor)

if ticks > tick\_bug\_fix [set last\_cleaning\_location [-15 -15] set tick\_bug\_fix tick\_bug\_fix + 10000] ;; senao ele fica la em cima e nao volta.... porque nao tem movimentos random suficiente para voltar para baixo

]

facexy item 0 posto\_carregamento item 1 posto\_carregamento ;; código direcçao à bateria

][

ifelse capacity >= cleaner\_max\_capacity [

;; modo ir depositar

ifelse [pcolor] of patch-here = blue [

set capacity 0 ;; esvazia capacidade toda (ia melhorar mas melhor guardar para fase 2)

][

let target-patch min-one-of (patches in-radius 40 with [pcolor = blue]) [distance myself] ;; (apenas esta linha é) solução stackoverflow: https://stackoverflow.com/questions/36019543/turtles-move-to-nearest-patch-of-a-certain-color-how-can-this-process-be-sped

if target-patch != nobody [

ask cleaner cleaner\_atual [

face target-patch ;; direcionar para o deposito

]

]

]

][

;; movimento modo aspirar

ifelse last\_cleaning\_location != [0 0] [ ;; voltar ao local anterior

facexy item 0 last\_cleaning\_location item 1 last\_cleaning\_location ;; TO UPGRADE: virar caso bata enquanto vai para sitio dele

][

;; aspirar área desconhecida

if patch-ahead 1 = nobody or ([pcolor] of patch-ahead 1 != cor\_chao and [pcolor] of patch-ahead 1 != black) [

right random 360

]

]

;; mudar cor (limpar o chão)

ifelse pcolor = cor\_chao [

if capacity + 1 <= cleaner\_max\_capacity [

set pcolor cor\_lixo\_cleaned

set capacity capacity + 1

]

][

set last\_cleaning\_location [0 0] ;; caso tenha batido num obstáculo ou já tenha limpo sítio anterior

]

]

]

]

]

]

;; baixa de bateria (dps do ciclo em q limparam o lixo)

ask cleaners [

set battery battery - cleaner\_consumption\_battery ;; cleaner\_consumption\_battery é a variável q depende da potência

if battery < 0 [ set battery 0 ]

]

]

;; ações dos polluters

ask polluters [

if random 100 <= prob\_sujar [

ask patch-here [

ifelse pcolor = cor\_chao [

let lixo random-float 100 ;; random para decidir tamanho do lixo

ifelse lixo <= 40 [

set pcolor item 0 tipo\_lixo

][

ifelse lixo <= 70 [

set pcolor item 1 tipo\_lixo

][

set pcolor item 2 tipo\_lixo

]

]

][

ifelse pcolor = item 0 tipo\_lixo or pcolor = item 1 tipo\_lixo or pcolor = item 2 tipo\_lixo [

;; NÃO fazer nada pois já tem lixo

][

right random 360 ;; não largar lixo em cima de objetos ou do depósito

]

]

]

][

right random 30

forward 1

]

]

tick

end

; go\_n, cujo programa permita: que os agentes circulem no mundo de forma aleatória (vários ticks);

to go\_n

repeat n [

go\_once

]

end