UNIVERSIDADE FEDERAL DO MARANHÃO
BACHARELADO INTERDISCIPLINAR EM CIÊNCIA E TECNOLOGIA
ENGENHARIA DA COMPUTAÇÃO
DISCIPLINA: INTELIGÊNCIA ARTIFICIAL

Docente: Dr. Thales Levi Azevedo

AGENTES- MODIFICANDO O MODELO DE FORMIGAS EM NETLOGO

Emanuel Lopes Silva- 2021017818
Letícia Delfino de Araujo-2021061763
Thales Aymar Fortes de Souza - 2021018145

São Luís-MA
03/12/2024

Emanuel Lopes Silva

Thales Aymar Fortes de Souza

Letícia Delfino De Araujo

AGENTES- MODIFICANDO O MODELO DE FORMIGAS EM NETLOGO

Este trabalho consiste na criação de um formigueiro, com predadores, mudanças de estações e outras melhorias do modelo base. Este documento é referente a disciplina de Inteligência Artificial do Curso de Engenharia da Computação, ministrado pelo Prof. Dr. Thales Levi Azevedo

São Luís – MA

2024

Sumário

Introdução

Agentes inteligentes são sistemas capazes de interpretar o ambiente, processar informações e realizar ações com a finalidade de atingir objetivos específicos. Esses agentes desempenham um papel importante na modelagem e simulação de fenômenos complexos, facilitando a análise de interações e comportamentos em vários cenários, tais como ecossistemas naturais, sistemas econômicos e dinâmicas sociais.

A simulação de colônias de formigas é um dos modelos mais empregados para entender sistemas baseados em agentes. Este modelo é notável pela simplicidade de suas normas individuais, que, ao serem combinadas, originam comportamentos complexos e estruturados, como a procura eficaz por alimentos e a comunicação através de feromônios. Esses acontecimentos espelham os fundamentos da auto-organização e da inteligência coletiva.

1- Objetivo

Estimular a compreensão dos conceitos de Inteligência Artificial e modelagem baseada em agentes através da modificação criativa do modelo de formigas no NetLogo. Os alunos foram desafiados a alterar o código fornecido para incorporar novos comportamentos, elementos ambientais ou mecanismos de interação, analisando o impacto de suas mudanças no comportamento emergente da simulação. Nesse sentido, os comportamentos propostos a serem implementados foram:

- Condições Climáticas Variáveis
- Introdução de Obstáculos no Ambiente
- Implementação de Predadores
- Ciclo de Vida das Formigas (Energia e reprodução)
- Hierarquia Social (operária, lutadora, rainha)
- Combate entre formigueiros (Lutadoras atacam formigas de outros formigueiros)

2 - Desenvolvimento

O ambiente é parcialmente observável, dinâmico, contínuo, não -determinístico e multiagente. Para explicar como a simulação do formigueiro se classifica em cada um desses aspectos, a lista abaixo serve para ilustrar melhor como cada classificação está presente no ambiente.

- Parcialmente Observável: As formigas têm conhecimento limitado do ambiente, percebendo apenas sua vizinhança imediata.
- Dinâmico: O ambiente muda constantemente devido à interação das formigas e fatores externos.
- Contínuo: As ações das formigas ocorrem em tempo real, sem interrupções.
- Não-determinístico: O comportamento das formigas é influenciado por fatores aleatórios e imprevisíveis. Muitas decisões das formigas, como direção de movimento, escolhas de ações, ou variações em atributos como velocidade, força ou mutação, envolvem o uso de funções aleatórias como random e random-float. Além disso, as mudanças sazonais, que alteram características do ambiente (como patches ou árvores), afetam os comportamentos das formigas de maneiras que não são determinísticas.
- Multiagente: Múltiplas formigas interagem entre si e com o ambiente.
- Já as formigas, são Agentes reativos baseados em modelo, tendo como característica reagir aos estímulos do ambiente (as comidas, o feromônios e todos os obstáculos) , interagem com outros agentes e possuem um conhecimento limitado do ambiente

Para acomodar esses novos comportamentos,foram feitas alterações no **Ambiente**, nas **Formigas**, e também foram criados três novos agentes: o tamanduá, o pangolim e o lobo, que agem como **Predadores.**

2.1 - Alterações no Ambiente

2.1.1 - Clima:

As modificações no ambiente buscam satisfazer os comportamentos de condições climáticas variáveis, adicionando mudanças de estação, e de introdução de obstáculos, com patches que impeçam o movimento das formigas.

Desse modo,o sistema de estações foi implementado de modo que o ambiente começa aleatoriamente em uma das 4 estações e depois de um certo número de ticks, que pode ser alterado por uma caixa de entrada na interface, passa para a próxima estação. Para colocar isso no programa, foi criado o procedimento “handle-season-change”, mostrado na figura a seguir

Imagem 1 - Alteração do Clima

```
;=== Funções de alteração do clima ===
to handle-season-change
  if (ticks mod tamanhoseason = 0) [ ; variável que permite a mudança climática
    ;; Atualiza a estação
     let current-index position current-season seasons
     let next-index (current-index + 1) mod length seasons ;pega a próxima estação da lista lá de cima  set seasons ["Primavera" "Verão" "Outono" "Inverno"]
    set current-season item next-index seasons ; Vai pra próxima estação
    ;if current-season = "Primavera" [ set current-season "Verão" ]
   ;  if current-season = "Verão" [ set current-season "Outono" ]
    ; if current-season = "Outono" [ set current-season "Inverno" ]
    ; if current-season = "Inverno" [ set current-season "Primavera" ]

    ;; Altera o mapa com base na nova estação
    update-patches-for-season ; função de atualizar somente os pactches. N quis botar no recolor patches pra n ficar muita coisa ali e se der erro n dar muito problema.
    update-trees
  ]
end
```

Autoria própria

Primeiramente, o trecho inicial do trecho verifica se o número de ticks (passos de tempo no modelo) é múltiplo da variável tamanhoseason, que representa a duração de cada estação e é definida pelo próprio usuário na tela inicial. Se a condição for verdadeira, é hora de mudar para a próxima estação.

O código de pintar o mundo da Imagem 2 torna o processo de troca de estações mais fácil de ser visualizado, dado que todo o ambiente é pintado para as cores características de cada uma das estações escolhidas, que são Outono, Inverno, Verão e Primavera.

Imagem 2 - Pintar o mundo

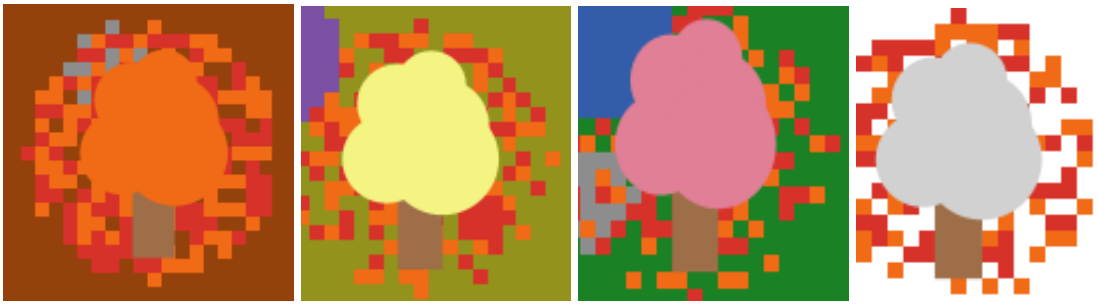
```
to update-patches-for-season

  ask patches [
    if current-season = "Primavera" [
     if pcolor = white[
      set pcolor 63


    ]
      if pcolor = black[
      set pcolor 63

    ]]
    if current-season = "Verão" [
      if pcolor = 63[
      set pcolor 43

    ]]
    if current-season = "Outono" [ ; como a cor dai é predominante laranja, tive que modificar a cor base pra n dar problema na hora de aparecer as coisas.
      if pcolor = 43[
      set pcolor 23

    ]]
    if current-season = "Inverno" [
      if pcolor = 23[
      set pcolor white

    ]]
  ]
end
```

Autoria própria

O comando de pintar as árvores na troca do ambiente também serve para além dos fins cosméticos na demonstrar a variação das estações, sendo essencial no processo de geração de alimentos para as formigas. O código foi feito utilizando como base os ciclos de frutificação de árvores como laranjeiras e macieiras, tendo seu período de criação de frutos em outubro, que é o período de Outono na simulação. O comando também realiza o nascimento do easter-egg Lucas. As colorações de cada árvore possível estão presentes na Imagem 3, e o código para implementar a mudança está na Imagem 4.

O código de pintar árvores também realiza o controle do nascimento de predadores hostis às formigas, utilizando a variável “chancepredador” para verificar se as condições estão satisfeitas para a criação de uma nova leva de tamanduás ou pangolins. A variável “numpredador” repete o processo quantas vezes o usuário quiser, para tornar mais fácil o controle da geração de predadores, apesar de ainda ser randômico, pois depende das probabilidades aleatórias do comando Random.

Imagem 3 - Possibilidades de Árvore



Autoria própria

Imagem 4 - Código Pintar as Árvores

```
to update-trees
    if random 100 < 3[ ;%3 de spawnar
    create-lucas 1[;; easter egg
      setxy 0 0
      set size 4
    set color 136
      set shape "person soldier"
      set label "Lucas o intruso do formigueiro"
      set strenght 10000
      set life 100000
      set speed 1

    ]
  ]
  repeat num-predador[
  if random 100 < chancepredador[
      gerar-predador]]
  if current-season = "Primavera" [
  ask turtles with [shape = "tree"] [
      set color pink
      ]


  ]
  if current-season = "Verão" [
    ask turtles with [shape = "tree"] [
      set color 47
      ]

  ]
  if current-season = "Outono" [
       ask turtles with [shape = "tree"] [
      set color orange
      ]

      ask turtles with [shape = "tree"] [
    ; Para cada árvore, define patches ao redor como fontes de alimento
    let nearby-patches patches in-radius 9 ; Ajuste o raio conforme necessário
    ask nearby-patches [
      if random 100 < chancedomida [ ; 50% de chance de gerar comida em cada patch ao redor
        set food one-of [1 2] ; Define quantidade de comida aleatória (1 ou 2)
        set food-source-number [who] of myself ; Identifica a árvore como fonte de alimento
        if food = 1 [ set pcolor red ] ; Comida 1 -> vermelho
        if food = 2 [ set pcolor orange ]   ; Comida 2 -> laranja
        if food = 1[ set food-source-number 1]
        if food = 2[ set food-source-number 2]
      ]
    ]
  ]
  ]
  if current-season = "Inverno" [
       ask turtles with [shape = "tree"] [
      set color 8
      ]

  ] ; Com isso, qualquer ninho que n tiver comida guardada está ferrado! Sem Setup food e sem comidas aleatórias.
end
```

Autoria própria

A mudança de estação também influencia as turtles de breed “queen-ants” fazendo com que elas testem se o ninho adquiriu comida suficiente para as mesmas realizarem o procedimento de nascimento, que será melhor explicado no tópico 2.2.

Após isso, a disposição de comida no ambiente foi alterada, de modo que ao invés dela estar sempre nos mesmos lugares nas mesmas quantidades, agora ela nasce perto de árvores em quantias aleatórias no começo da simulação e quando a estação mudar para outono. O número de árvores no início da simulação pode ser alterado manualmente ou deixado como aleatório. A chance de nascer comida próximo às árvores também pode ser alterada. Para que isso ocorra, o comando breed foi utilizado para criar uma nova raça trees, representando as árvores, e então adicionou-se uma estrutura ifelse no procedimento setup para checar se o número de árvores que vão nascer deve ser aleatório ou não. Caso sim, nascem entre 0 e 99 árvores, caso não, nasce um número de árvores determinado pelo deslizador “num-arvore”, controlado pelo usuário.

2.1.2 - Obstáculos:

Para implementar obstáculos, foi dado a cada patch uma variável obstacle? e foi criada a função setup-obstaculo demonstrada abaixo

Imagem 5 - Código para criar obstáculos

```
;== Obstáculos ==
to setup-obstaculos
   ask patches [
  if random 10000 < random 10[ ;; 2%% de chance de o patch ser o centro de uma pedra base
    ask patches in-radius 3 [ ;; Define os patches ao redor como parte da pedra
      set obstáculo? true
      set pcolor gray ;; Define a cor da pedra
    ]
  ]
]
end
```

Autoria própria

Nesse código, cada patch que tem uma chance de 0,1% de se tornar a variável obstacle? True, fazendo com que aquele patch e os patches em um raio de 3 unidades dele virem obstáculos e sejam pintados de cinza. Esses obstáculos atrapalham o movimento das formigas.

Em virtude de criar um obstáculo fácil de se reproduzir, foi-se escolhido que um obstáculo apropriado seriam pedras, que impedem que as entidades pequenas, que são as formigas, consigam atravessar os patches que foram classificados com obstacle? = true. Para facilitar a visualização dos obstáculos, eles foram pintados com a cor cinza, que é a cor de alguns pedregulhos comumente encontrados.

É possível aumentar a proporção de pedras no mapa, dado que a proporção base é 1 milésimo (1 ‰), só que um leve aumento nesta proporção torna o mapa inutilizável para as formigas, dado que cada patch pode influenciar os outros ao redor, gerando assim grandes partes do mapa que não podem ser atravessadas. Por isso foi utilizado uma porcentagem tão baixa para a criação de obstáculos.

Imagem 6 - Representação da pedra

Autoria própria

2.2 - Formigas

As mudanças nos procedimentos das formigas buscam implementar ciclo de vida, com um sistema de energia e reprodução; hierarquia social, com divisão das formigas nas classes de trabalhadora, soldado e rainha; e luta entre formigueiros, com a adição de três novos ninhos de formigas com cores diferentes.

Primeiramente, para adicionar novas classes de formigas, foram criadas três novas breeds: worker-ants para formigas trabalhadoras, soldier-ants para formigas soldado, e queen-ants para formigas rainha.

Imagem 7 - Breeds de formiga

```
breed [worker-ants worker-ant]  ; Formigas Trabalhadoras
breed [soldier-ants soldier-ant] ; Formigas Guerreiras
breed [queen-ants queen-ant] ; Formigas rainha
```

Autoria própria

As três breeds possuem as variáveis energy, colony, life, strength, mutation, speed e vítima, e as soldier-ants também possuem a variável role. A função dessas variáveis será explicada ao decorrer da seção.

Imagem 8 - Variáveis das formigas

```
worker-ants-own [energy colony life strenght mutation speed]
soldier-ants-own [energy colony life strenght mutation speed role]
queen-ants-own [energy colony life strenght mutation speed]
```

Autoria própria

Existe um comando para a atribuição das variáveis mencionadas anteriormente, sendo eles a vida, ataque, a velocidade, energia e a possibilidade de mutação. A vida, força, energia e a velocidade foram atribuídas a valores fixos, para a distinção de cada colônia, escolhidos com base em um gerador aleatório de números do google, mantendo todos eles em um alcance significativo, para não haver divergências de poder entre as colônias. Entretanto, foi implementado o sistema de mutação, onde ocorre a aplicação de alterações aleatórias nas características de agentes com base em uma probabilidade normal de 30%, com a possibilidade de alterar a chance de mutação na tela inicial. A mutação ocorre quando um número aleatório entre 0 e 1, gerado pela função random-float, é menor que 0.3. Caso a mutação seja ativada, a variável mutation é configurada como "yes" e os atributos do agente, como vida, força e velocidade, são ajustados de forma aleatória. A vida do agente é multiplicada por um valor entre 0% (reduzida a zero) e 200% (duplicada), enquanto a força pode variar de 0% (reduzida a zero) até 300% (triplicada). A velocidade, por sua vez, é ajustada mais moderadamente, variando entre 50% (reduzida à metade) e 150% (aumentada em até 50%).

Imagem 10 - Processo de mutação

```
ifelse random-float 1 < chancemutacao [ ;botei 30% de chance para ver um impacto real, talvez diminua dps.
  ; Aplica mutação
  set mutation "yes"
  set life life * (1 + random-float 2.0 -  1.0) ; aumenta a vida em 100% // ou diminui
  set strenght strenght * (1 + random-float 4.0 -  2.0) ; Aumenta a força até 200% // ou diminui
  set speed speed * (1 + random-float 1 -  0.5)       ; Aumenta a velocidade até 50% // ou diminui
] [
  set mutation "no"
]
```

Autoria própria

As formigas trabalhadoras operam de modo semelhante das formigas da simulação NetLogo original, usando os mesmos procedimentos mais um procedimento para fugir de predadores, que serão apresentados na seção 2.3, e mais um procedimento para evitar obstáculos Assim, elas andam aleatoriamente até acharem comida, e quando acham mudam para um tom mais claro e voltam para o ninho, deixando um rastro de feromônios para as outras formigas trabalhadoras., e caso encontrem um predador próximo, correm na direção oposta dele. O comportamento de fuga é feito pelo procedimento fugir-do-predador:

Imagem 11 - Fuga do predador

```
to fugir-do-predador
  let predadores-proximos turtles with [
    (breed = tamanduas or breed = pangolims) and
    (distance myself < 5)  ;; Distância de segurança
  ]

  if any? predadores-proximos [
    let predador-mais-proximo min-one-of predadores-proximos [distance myself]
    ;; Virar na direção oposta ao predador
    let angulo-oposto (towards predador-mais-proximo + 180) mod 360
    facexy (xcor + cos angulo-oposto) (ycor + sin angulo-oposto)
    ;; Mover-se rapidamente para longe
    fd 1.5
  ]
end
```

Autoria própria

Essa função checa se há predadores num raio de 5 unidades da formiga, se sim, ela escolhe o predador mais próximo, vira-se para a direção oposta dele e foge.

Além disso, a formiga trabalhadora também tem uma função para que ela consiga evitar ficar parada nos obstáculos por muito tempo, dado que muitas vezes elas podem estar carregando alimentos, e o ninho sofrerá caso ela perca muito tempo. Entretanto, para tornar as coisas mais naturais, o movimento delas é pelo “wiggle”, e ele, sendo aleatório, pode tornar a saída do obstáculo um pouco demorada.

Imagem 12 - Checar obstáculo

```
to check-obstaculo
  ask turtles [
  let possible-patch one-of patches in-radius 2 with [not obstáculo?]
  if possible-patch != nobody [
    move-to possible-patch
  ]
]
end
```

Autoria própria

O comando de movimentação foi levemente alterado, e agora é chamado de move forward. Ele permite que as formigas evitem andar por cima dos obstáculos, e chama o comando anterior de checar obstáculos para saírem das pedras e continuarem seu caminho. O comando utilizado em todas as formigas está abaixo na Imagem 13.

Imagem 13 - Movimento Geral

```
to move-forward [steps]
  ;; Move-se para frente apenas se o patch à frente não for um obstáculo
   ;; Move-se para frente no máximo 'steps' passos, evitando obstáculos
  let moved 0
  while [moved < steps] [
    let next-patch patch-ahead 1
    ifelse next-patch != nobody and [not obstáculo?] of next-patch [
      fd 0.75
      set moved moved + 1
    ][
      ;; Se encontrar obstáculo, para de se mover
      stop
    ]
  ]

end

to wiggle  ; procedimento das formigas  MODIFICADO PARA EVITAR OBSTACULO

  rt random 45                          ; vira um ângulo aleatório à direita
  lt random 45                          ; vira um ângulo aleatório à esquerda
  if not can-move? 1 [ rt 180 ]         ; se não puder se mover, vira 180 graus
  let obstaculo one-of patches in-radius 1 with [obstáculo?]
  if obstaculo = true [ rt 180 ]
end
```

Autoria própria

Já as formigas lutadoras andam em círculos em volta do ninho, através da função movimento-soldado e atacam inimigos próximos (predadores e formigas de outros ninhos, explicados mais adiante). Esses dois comportamentos são gerenciados pelos procedimentos movimento-soldado e patrulha-ou-luta.

Imagem 14 - Movimento-Soldado

```
to movimento-soldado
  if role = "patrolling"[
  let closest-nest one-of patches with [
    nest? = true and pcolor = violet
  ]
  ;; Define um lugar específico que o soldado vai patrulhar (isto é, o local da nest)
  let nest-x -16  ;; Define as coordenadas em x da nest
  let nest-y -36  ;; Define as coordenadas em y da nest

  ;; Move em círculos ajustando o heading com valores aleatórios e andando para frente
  set heading heading + random 10 ;; Randomiza um pouco o heading para o movimento parecer mais natural
  move-forward speed ;; Anda para frente

  ;; Se o soldado estiver muito próximo da nest, muda de direção
  ifelse distancexy nest-x nest-y < 3 [
    rt random 90  ;; Se vira aleatoriamente para evitar ficar parado no mesmo lugar
    move-forward speed        ;; Anda para frente
  ][
    ;; Continua partulhando no local determinado
    rt 15  ;; Rotaciona um pouco para continuar patrulhando em círculo
    move-forward speed    ;; Anda para frente
  ]]
end
```

Autoria própria

Esse procedimento verifica se a role da formiga soldado é “patrolling”, se sim , são definidas as coordenadas da nest que serão patrulhadas e é feito com que as formigas andem em círculos numa velocidade igual a variável speed. Depois, uma estrutura ifelse checa se o soldado está muito próximo da coordenada da nest definida. Se sim, ele muda de direção. se não, continuam patrulhando.

Imagem 15 - Patrulha-Ou-Luta

```
to patrulha-ou-luta
  let inimigos-proximos turtles with [(breed != trees) and (breed = worker-ants or breed = soldier-ants or breed = queen-ants or breed = pangolims or breed = tamanduas) and
    (colony != [colony] of myself) and  (distance myself < 5)]
  ifelse any?  inimigos-proximos ;;pra impedir as formigas de atacarem as arvores
     [
    set role "fighting"      ;; Muda o trabalho para Modo Lutador, pra meter a porrada nos trabalhadores inocentes (que nem a vida real)
    let inimigo-atual one-of inimigos-proximos
    face inimigo-atual  ;; Vira-se para o inimigo
    fd 1                ;; Move-se 1 passo em direção ao inimigo
     if distance  inimigo-atual < 6[
    ask inimigo-atual[
      face self
      set life life - [strenght] of myself
  ]]]
     [
    set role "patrolling"    ;; Caso n veja nenhum inimigo, continuar patrulhando o ninho original.
  ]
end
```

Autoria própria

A Função utiliza o comando ifelse para checar se existem turtles inimigos nas (predadores e formigas de outros ninhos) nas proximidades. Se sim, a variável role é definida como “fighting” e a formiga lutador escolhe um dos inimigos para ser seu inimigo atual, virando e andando em direção a ele. Então o comando ask é usado para decrementar a vida da presa com base no atributo strenght da formiga. Se não houver inimigos próximos, a variável role é definida como patrolling, fazendo com com que a função movimento-soldado seja executada.

Por fim, a formiga rainha fica parada dentro do ninho e é responsável pela reprodução do formigueiro, e quando ela morre, o formigueiro também morre. Caso as formigas trabalhadoras tenham conseguido comida o suficiente para o formigueiro, a rainha cria mais formigas na mudança de estação, através de estruturas condicionais.

A rainha não é indefesa, entretanto, tendo os maiores atributos da colônia. Apesar de não se movimentar, tem seu alcance de ataque aumentado, para que ela defenda o ninho a qualquer custo. Ela consegue atacar apenas as entidades que chegam bem perto no ninho, mas possui uma quantidade de dano significativa.

Imagem 16 - Ataque da Rainha

```
to attack-rainha

  let inimigos-proximos turtles with [(breed != trees) and (breed = worker-ants or breed = soldier-ants  or breed = pangolims or breed = tamanduas) and
    (colony != [colony] of myself) and  (distance myself < 7)]
  if any?  inimigos-proximos ;;pra impedir as formigas de atacarem as arvores
     [
    let inimigo-atual one-of inimigos-proximos
    face inimigo-atual  ;; Vira-se para o inimigo

    ask inimigo-atual[
      face self
      set life life - [strenght] of myself
  ]]


end
```

Autoria própria

Imagem 17 - Procedimentos da Rainha

```
 ask queen-ants [
 ;tamanhoseason tamamho de cada season
if (ticks mod tamanhoseason = 0)[
   if colony = violet[
     if color = violet - 2 [
     let total-food sum [food-store] of patches
     if total-food > Preçoconstrução [ ; preço para subtrair da comida para gerar formiga
     nascimento
     set food-store food-store - Preçoconstrução
   ]]
     ]
   if colony = blue[
     if color = blue - 2 [
     let total-food2 sum [food-store2] of patches
     if total-food2 > Preçoconstrução [
     nascimento
     set food-store2 food-store2 - Preçoconstrução
   ]]]
   if colony = 126[

     if color = 126 - 2 [
     let total-food3 sum [food-store3] of patches
     if total-food3 > Preçoconstrução [
     nascimento
     set food-store3 food-store3 - Preçoconstrução
   ]]
      ]
   if colony = yellow[
     if color = yellow - 2 [
      let total-food4 sum [food-store4] of patches
     if total-food4 > Preçoconstrução [
     nascimento
     set food-store4 food-store4 - Preçoconstrução
 ]]]
  ]
```

Autoria própria

Esse bloco de código faz com que a cada virada de estação, as rainhas de cada colônia verifiquem a quantidade de comida no ninho, e, caso ela seja maior que a variável “Preçoconstrução”, elas executam o procedimento nascimento, que gera novas formigas, e o valor de comida gasto é retirada da reserva total de comida do ninho.

Para adicionar mais formigueiros e fazer uma distinção entre eles, foi atribuída a variável colony para as breeds de formiga,. Assim, os procedimentos originais de comportamento das formigas e as variáveis de patch que davam suporte a eles foram aproveitados e copiados mais três vezes, uma vez para cada novo formigueiro. As formigas de uma colônia agem independentemente das outras colônias, exceto pelas formigas soldados, que atacam caso alguma formiga de outra colônia se aproxime.

A fim de inserir o ciclo de vida das formigas, foram introduzidas as variáveis energy e life para as formigas, que caso cheguem a 0, causam a morte da formiga. A variável energy vai sendo decrementada a cada movimento, aumentando apenas quando alguma comida é coletada, enquanto a variável life indica a vida das formigas, e é decrementada quando a formiga sofre ataque inimigo. Para adicionar a morte das formigas, foi colocada a função check-death

Imagem 18 - Check-Death

```
to check-death ;ver se a formiga vai morrer
  ask worker-ants [
    if energy <= 0 or life <= 0 [ die ]
  ]
    ask soldier-ants [
    if energy <= 0 or life <= 0 [ die ]
  ]
  ask queen-ants [
    if energy <= 0 or life <= 0 [ die ]
  ]
  ask tamanduas[
   if energy <= 0 or life <= 0 [ die ]
  ]
   ask pangolims[
  if energy <= 0 or life <= 0 [ die ]
    ]
end
```

Autoria própria

A função usa o comando ask para perguntar para as turtles de cada breed se sua variável energy ou life chegaram a 0. Se sim, o comando die é utilizado para eliminar as turtles.

2.3 - Predadores

Para simular um ambiente com predadores, foram criadas três novas breeds: o tamanduá e o pangolim, que têm as formigas como presas, e o lobo, que tem o tamanduá e o pangolim como presas. Cada breed possui os atributos energy, life, strenght e speed, e o seu nascimento é controlado pelo seguinte bloco de código:

Imagem 19 - Código para gerar predadores

```
repeat num-predador[
if random 100 < chancepredador[
    gerar-predador]]
```

Autoria própria

Ele verifica se a variável chancepredador, controlado por um deslizador, é maior que um número aleatório entre 0 e 99, se sim, a função gerar-predador é executada. Essa checagem é repetida um número num-predador de vezes, variável também controlada por um deslizador. A função gerar-predador pode ser vista a seguir:

Imagem 20 - Função gerar-predador

```
to gerar-predador ;pangolims-own [energy colony life strenght speed]

  ifelse random 100 < 50[ ; 50% de gerar um ou outro
  create-tamanduas random 3[
      set shape "tamandua"
      set color brown
      set size 17
      setxy random 40 random 40
      set energy 10000
      set life 1600
      set strenght 87
      set speed 1
  ]]
  [create-pangolims random 5 + 1 [
      set shape "pangolim"
      set color 37
      set size 12
      setxy random 40 random 40
      set energy 10000
      set life 700
      set strenght 48
      set speed 1.7
  ]]
  if predador-moron = true [
  if random 100 < chancemor[
   create-wolfs 1 + random 2 [
      set shape "wolf"
      set color 3
      set size 16
      setxy random 40 random 40
      set colony "enemies"
      set life 900
      set strenght 180
      set speed 2.6
  ]]]
end
```

Autoria própria

O código usa uma estrutura ifelse para que na metade das vezes nasçam tamanduás e na metade das vezes nasçam pangolins,utilizando o comando create, e também usa o comando set para atribuir valores a suas variáveis. Além disso, caso a variável predador-moron, controlada por um interruptor na interface, seja true, também nascerão os lobos, com uma chance determinada pela caixa de entrada chancemor. Os sprites dos predadores Tamanduá e Pangolim tiveram de ser criados a mão pelos integrantes da equipe, pois nenhum outro animal dentro do dataset normal do netlogo se assemelhavam a esses animais. Então editou-se os sprites de coelho e lobo para criar as iterações atuais dos predadores mencionados anteriormente.

Os três predadores funcionam do mesmo modo, eles andam aleatoriamente pelo mapa com a função wiggle, e, ao encontrar uma presa próxima, vão em direção a ela e atacam, diminuindo os pontos de vida dela. Isso foi implementado com o procedimento predar, mostrado na figura 21.

Imagem 21 - Código de Predação

```
to predar
  let predador turtles with [breed = tamanduas or breed = pangolims]
  let predador-mor turtles with [breed = wolfs]
  ask predador [
    let presas-proximas turtles with [breed = worker-ants or breed = soldier-ants or breed = queen-ants and distance myself < 3]

    if any? presas-proximas [
      let presa one-of presas-proximas
      face presa  ;; Vira-se para o inimigo
      fd 1               ;; Move-se 1 passo em direção ao inimigo
      ask presa [
        face self
        set life life - [strenght] of myself
      ]
    ]
  ]
  if predador-moron = true [
  ask predador-mor [
  let presas-proximas turtles with [breed = tamanduas or breed = pangolims and distance myself < 3]
    if any? presas-proximas [
      let presa one-of presas-proximas
      face presa  ;; Vira-se para o inimigo
      fd 1               ;; Move-se 1 passo em direção ao inimigo
      ask presa [
        face self
        set life life - [strenght] of myself
      ]
    ]
  ]
 ]
end
```

Autoria própria

O procedimento usa uma lógica parecida com o patrulha-ou-luta. Ele checa quais turtles são predadores e "pergunta" para eles com o comando ask se há alguma presa próxima. Em caso afirmativo, uma delas é escolhida para ser a presa atual e o predador vira e anda em direção a ela. Então o comando ask é usado para decrementar a vida da presa com base no atributo strenght do predador. Os predadores são reativos simples, sem complexidade em seus padrões de movimento ou caça.

3 - Extras

Foi-se criado um elemento extra para o código, sobre a ideia de um dos membros, de se adicionar uma referência ao filme "Lucas: Um Intruso no Formigueiro". Essa entidade é extremamente rara, tendo apenas 3% de chance de aparecer durante a troca de estações.

O papel de Lucas é adicionar um elemento de curiosidade e diversão ao modelo, enquanto testa a resiliência das formigas e a interação entre agentes no sistema. Ele pode ser visto como um "intruso" que desafia as mecânicas do ambiente, forçando os agentes a reagirem de formas adaptativas. O Lucas é gerado na posição central 0 0, com uma cor similar ao do filme e com uma aparência de soldado e patrulha na área próxima a ele, identificando inimigos em um raio de 10 unidades.

Ele comporta-se como um predador de todos os outros turtles, exceto as árvores, movendo-se pelo mapa com a função wiggle e atacando qualquer outra entidade que estiver por perto, e é virtualmente imortal. Com atributos como força e vida extremamente altos, Lucas se torna uma entidade poderosa e quase impossível de derrotar, criando desafios para as formigas e outras entidades no ambiente.

4 - Conclusão

O modelo de Agentes de formiga nos possibilitou analisar os tipos de agentes , seu comportamento e como eles interagem no ambiente. Através da simulação no NetLogo observou-se a comunicação das formigas através dos feromônios liberados após a alimentação.

Foram implementados diversos recursos nesta simulação, como obstáculos, predadores, mudanças de clima e hierarquias diferentes de formigas. Cada recurso foi implementado visando impactar diretamente no comportamento dos agentes (Formigas).

Os obstáculos tinham como objetivo atrapalhar a locomoção das formigas, impactando assim na sua rota. Os predadores (Tamanduá , Pangolim e Lobo) foram inseridos para predar as formigas, cada um com um nível diferente de energia e atributos, e além do ciclo de vida dos predadores, em que o lobo caça o Tamanduá e o Pangolim. Dessa forma, é controlado o nível de agentes no jogos e é facilitado de forma mínima a sobrevivência das formigas. As mudanças de clima no ambiente definiram o comportamento das formigas, a geração de comida e de reprodução dos agentes, um dos pontos principais da simulação.

Além disso, foi possível observar como os agentes reativos baseados em modelos, que eram as formigas, reagiram ao ambiente mudar, confirmando o que foi aprendido em sala de aula pelos integrantes do grupo, além de se ver como os agentes reativos simples, que eram os predadores, funcionaram no ambiente, interagindo entre si e com as formigas.

Outrossim, foi possível criar um mundo parcialmente observável, dado que as formigas possuem sensores que permitem que elas tenham acesso a somente parte do ambiente, e são usados para detectar feromônios, alimentos e o ninho, além de atuadores, para se mover e manipular comida.

E por fim, a classe das formigas, cada uma exerce um papel em determinada condição dada, sendo crucial para a sobrevivência de toda a colônia . Portanto , cada recurso inserido à simulação implicou de forma ativa ao comportamento das formigas e ao resultado final da simulação , se atrelando aos estudos ricos de Alan Turing.

Referências

- Thales Levi Azevedo (2024) .Introdução a Organização de Computadores
- Wooldridge, M. (2009). *An Introduction to MultiAgent Systems* (2ª ed.). Wiley.
- Wilensky, U. (1999). *NetLogo*. Center for Connected Learning and Computer-Based Modeling, Northwestern University. Disponível em: https://ccl.northwestern.edu/netlogo/ . acesso em 25/11/2024