# Relatório Trabalho Prático 1
# Agentes

**Inteligência Artificial**

3º Ano de Engenharia Informática

1º Semestre


Ricardo Silva (70672) Vasco Teixeira (74107)

25 de Outubro de 2022

*Universidade de Trás-os-Montes e Alto Douro*

## Resumo

Neste Relatório abordámos a forma como realizámos o nosso trabalho prático, assim como os problemas com que nos deparámos e a forma como procedemos para os ultrapassar.

Os objetivos desta fase do projeto eram desenvolver uma simulação de um aquário através de agentes utilizando o NetLogo[1], onde existem 2 espécies diferentes de peixes que habitam pacificamente e que se alimentam e representar a qualidade da água através da sua cor.

# Conteúdo

# 1 Introdução

Nesta fase do trabalho prático deparámo-nos com o problema de desenvolver uma simulação de um aquário, no qual coexistem pacificamente 2 espécies de peixes, utilizando o NetLogo[1].

Tendo como objetivos representar as duas espécies de peixes, a qualidade da água, os agentes de comida e um gráfico que representa evolução das duas espécies.

# 2 Análise e Especificação

## 2.1 Descrição Informal

Perante esta situação desenvolvemos uma interface simples para o utilizador introduzir dados e visualizar os mesmo acerca da simulação

## 2.2 Requisitos

### 2.2.1 Interface

O utilizador irá fornecer algumas informações acerca do estado inicial do aquário a partir das quais será feita a simulação.

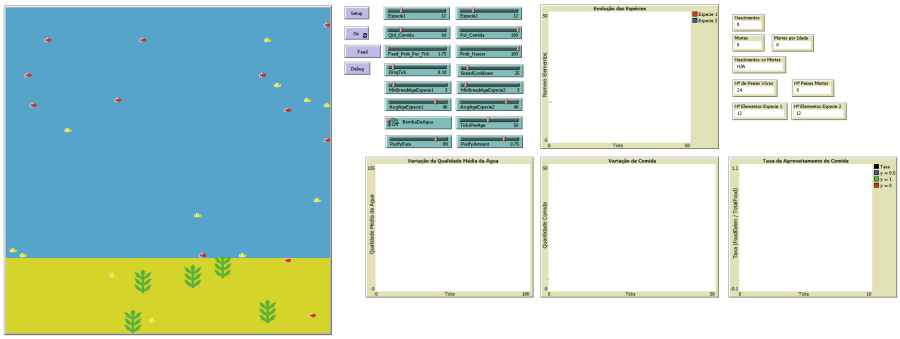

*Figura 1: Screenshot da Interface*

Tem 3 botões disponíveis:
- Setup - Reinicia a simulação
- Go - Corre a simulação
- Feed - Cria agentes comida

Tem 15 deslizadores:

Especie1 e Especie2 - Nº de elementos inicial de cada espécie
Qtd_Comida - Quantidade de comida a ser criada
Pol_Comida - Poluição quando a comida se desfaz
Feed_Prob_Per_Tick - Escolhe a probabilidade de criar comida por cada tick
Prob_Nascer - Probabilidade dos peixes se reproduzirem
Dmg_Tick - Dano que os peixes levam por tick
Breed_Cooldown - Tempo entre a reprodução dos peixes
Min_Breed_Age1 e Min_Breed_Age2 - Idade Minima para reproduzir
AvgAgeEspecie1 e AvgAgeEspecie2 - Esperança média de vida dos peixes
TicksPerAge - Valor de ticks para incrementar a idade
PurifyRate - Velocidade da Bomba de Água
PurifyAmount - Percentagem de incremento da qualidade da água

Tem 1 switch:

BombaDeAgua - Liga ou Desliga a Bomba de Água

Tem 4 gráficos:

Evolução das Espécies - Mostra a evolução das espécies
Variação da Qualidade Média da Água - Mostra a qualidade da água
Variação da Comida - Mostra a quantidade de comida
Taxa de Aproveitamento de Comida - Mostra a taxa de comida consumida

Tem 9 monitores:

Nascimentos - Nº de Nascimentos
Mortes - Nº de Mortes
Mortes por Idade - Nº de Mortes por Idade
Peixes Comidos - Nº de Peixes comidos pela Especie 3
Nascimento vs Mortes - Taxa de Nascimento vs Mortes
Nº de Peixes Vivos - Nº Atual de Peixes
Nº de Peixes Mortos - Nº Atual de Peixes Mortos
Nº Elementos Especie 1 - Nº Atual de Peixes da Especie 1
Nº Elementos Especie 2 - Nº Atual de Peixes da Especie 2

### 2.2.2 Objetivo da Primeira Fase de Implementação

Criar uma simulação de um aquário com duas espécies de peixes que se alimentam, coexistem pacificamente e acabam por morrer.

### 2.2.3 Objetivo da Segunda Fase de Implementação

Adicionar outros elementos ao aquário, tais como: diferentes espécies, bomba de água, etc, tornando, assim, a simulação mais complexa.

# 3 Resolução

## 3.1 Estruturação dos dados

```
1  globals [AvgWaterQuality FoodEaten FoodDestroyed Nascimentos Mortes MortesAge]
2  breed [comidas comida]
3  breed [plantas planta]
4  breed [peixes1 peixe1]
5  breed [peixes2 peixe2]
6  breed [mortos morto]
7  peixes1-own[canBreed age hp dieAge breedCD]
8  peixes2-own[canBreed age hp dieAge breedCD]
9  mortos-own[decay]
10 patches-own [quality] ;qualidade = 100 - lixo
11 comidas-own [decay]
```

*Figura 2: Screenshot da Estruturação dos dados*

### 3.1.1 Variáveis Globais

AvgWaterQuality - Qualidade Média da Água
FoodEaten - Comida consumida
FoodDestroyed - Comida desperdiçada
Nascimentos - Nº de Nascimentos
Mortes - Nº de Mortes
MortesAge - Nº de Mortes por Idade

### 3.1.2 Espécies

Comidas - Agentes de comida
Plantas - Agentes de planta
Peixes1 - Agentes da Espécie 1
Peixes2 - Agentes da Espécie 2
Mortos - Agentes de peixes mortos

### 3.1.3 Variáveis das Espécies

#### 3.1.3.1 Peixes

canBreed - Disponibilidade para reproduzir (0 ou 1)
age - Idade
hp - Vida
dieAge - Idade para o peixe morrer
breedCD - Cooldown para reproduzir novamente
eatCD - Cooldown para comer

#### 3.1.3.2 Mortos

decay - Nº de ticks para o corpo se desfazer

#### 3.1.3.3 Patches

quality - Qualidade da Água *(ao invés de lixo)*

#### 3.1.3.4 Comidas

decay - Nº de ticks para a comida se desfazer

## 3.2 Algoritmos

Optamos por dividir o problema em bocados mais pequenos e assim criar algoritmos para cada um desses problemas. Criamos algoritmos para a Bomba de Água, Reprodução, Cor da Água, Plantas, Dinâmica de Fluídos e controlar os Peixes. O source-code está disponível em anexo.

# 4 Programação

## 4.1 Decisões e Problemas (Ordem Cronológica)

### 4.1.1 Cor de Fundo

Após a leitura e interpretação do protocolo decidimos logo que a cor de fundo da água iria ser calculada através de uma função linear

$$\boxed{0.06x + 90}$$

onde $x$ é o valor da qualidade da água nesse patch.

Após realizar outras tarefas e adicionar o fundo de areia, decidimos que esta parte segue a seguinte função

$$\boxed{0.044x + 40}$$

onde $x$ é o valor da qualidade da água nesse patch.

### 4.1.2 Movimentação dos Peixes

Os peixes nascem com um valor aleatório para onde se vão movimentar e deslocam-se a uma velocidade de 1 patch por tick. Estes também possuem a capacidade de trocar a orientação para onde se vão deslocar, com uma probabilidade de 10%, os peixes escolhem uma orientação nova num formato de cone de 90° à sua frente (ver ilustração abaixo). Ao entrar em contacto com uma das paredes do aquário também escolhem uma nova orientação.

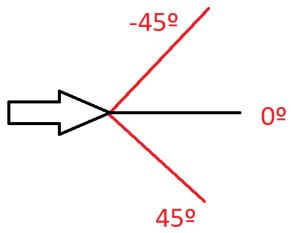


*Figura 3: Esquema da movimentação dos peixes*

### 4.1.3 Movimentação da Comida

A comida tal como os peixes movimenta-se de forma aleatória, mas esta é, geralmente, mais lenta que os peixes. Para isto utilizamos o valor dos ticks da simulção e um valor aleatório de 0 a 10, caso o nº de ticks total seja divisível por o nº aleatório a comida escolhe uma das seguintes orientações: 135°, 180° ou 225° e anda para baixo 1 patch.

### 4.1.4 Interação Peixe-Comida

Quando um peixe e um agente comida estão no mesmo patch, o peixe irá consumir um dos agentes comida, caso exista mais que um, e incrementa o seu tamanho, a sua vida, a quantidade total de comida consumida e caso não esteja em cooldown de reprodução coloca a variável canBreed a 1.

### 4.1.5 Destruição da Comida

Todos os ticks o decay da comida é reduzido e quando este chega a 0 este agente desaparece e reduz a qualidade da água no patch em que se encontrava.

### 4.1.6 Dinâmica de Fluídos

Este foi o algoritmo no qual tivemos mais dificuldades e que consumiu mais tempo. Tal como na vida real a àgua mais suja tende a ser mais pesada, nesta simulação este algoritmo permite simular isso mesmo. Através de verificações da qualidade dos patches vizinhos a água mais suja desloca-se com tendência a ir para baixo

Posteriormente surgiu um problema onde a qualidade da água vai para valores negativos, para solucionar este problema a qualidade da àgua do patch em questão passa para 0 e o valor negativo é atribuído ao patch do topo do aquário para, assim, forçar o algoritmo da dinâmica de fluídos a redistribuir esta água. Desta forma evitámos que a comida desperdiçada não influência a qualidade média da água se a qualidade do patch onde esta se encontrava já fosse 0. Uma vez que a qualidade média da água seja 0 todos os patches alteram a sua cor para a mais escura de todas.

#### 4.1.7 Dano e Morte dos Peixes

Os peixes ao serem criados têm uma idade limite que ao ser atingida faz com que eles morram (Implementado posteriormente). Também existe um sistema de dano que simula as "doenças" que o peixe pode vir a ter ao longo da sua vida, para isto usámos a seguinte função

$$\boxed{hp - (DmgTick \times (3 - \frac{thisquality}{50}))}$$

onde hp é a vida do peixe, DmgTick é o dano a receber e thisquality é a qualidade da água no patch. Assim, o peixe leva sempre um dano constante por tick e caso a qualidade da água seja inferior num determinado patch o valor do dano é maior.

#### 4.1.8 Reprodução

O sistema de reprodução é bastante simples e consiste no facto que um peixe para se alimentar tem de se cruzar com outro peixe da mesma espécie no mesmo patch e ambos terem a variável canBreed a 1. Também têm de ter mais do que a idade mínima para se reproduzirem.

#### 4.1.9 Bomba de Água

A bomba de água simula a filtragem da água de um aquário. Para obter este resultado o nosso algoritmo modifica a qualidade da água dos 8 patches mais abaixo e do lado do aquário (4 de cada lado).

A bomba funciona periodicamente, a periodicidade com que esta trabalha é escolhida pelo utilizador. Através da seguinte função é calculada a nova qualidade da água naquele patch

$$\boxed{quality + PurifyAmount \times (100 - quality)}$$

onde quality é a qualidade da água e PurifyAmount é o valor escolhido pelo utilizador.

#### 4.1.10 Sistema de Idade dos Peixes e Peixes Mortos

Os peixes ao nascer recebem uma variável que representa até que idade vão viver. Quando esta é atingida os peixes morrem e criam um novo agente com as mesma características que representa um peixe morto. Também é criado quando morrem por "dano". Os peixes mortos têm uma variável de decay que quando chega 0 faz com que este desapareça. Enquanto o peixe morto se encontra "vivo" este polui a água do patch onde se encontra.

#### 4.1.11 Plantas

As plantas tal como na vida real também ajudam na purificação da água. Para tal ao ser gerado o aquário é escolhido um número aleatório de plantas. Para estas purificarem a água utilizam um algoritmo similar ao da Bomba de água, mas os seus valores são fixos. Purificam 1% da água de cada patch à volta da planta num intervalo de 10 em 10 ticks. Quando a qualidade da água onde se encontram é inferior a 20 estas morrem.

#### 4.1.12 Tubarões

Os tubarões de forma indireta ajudam a manter a qualidade da água uma vez que estes comem os peixes que estão mortos. Para adicionar os tubarões tivemos de adaptar o código do sistema de alimentação e de reprodução.

#### 4.1.13 Espécie 3

Esta espécie de peixes possui a capacidade de se alimentar de elementos de outras espécies desde que estes sejam mais pequenos

### 4.2 Funções Principais

SETUP - A função setup tem como objetivo estabelecer um novo aquário e limpar as informações das simulações anteriores.

GO - A função go tem como objetivo correr a simulação de forma automática. Esta também alimenta os peixes de forma automática.

FEED - A função feed tem como objetivo permititr ao utilizador criar mais comida do que aquela que é gerada automaticamente.

### 4.3 Testes realizados e Resultados

Foram realizados diversos testes ao longo do trabalho, deparámo-nos com alguns problemas mas com um pouco de persistência e pesquisa, na documentação[2] e no StackOverflow[3], conseguimos solucionar todos. Alguns destes testes serviram apenas para ajustar os valores fixos dos algoritmos para ter uma simulação mais estável.

Tal como foi referido anteriormente a secção do trabalho que deu mais trabalho foi a implementação da dinâmica de fluídos, não só por ser mais complexo que o restante, mas também por ter sido realizada numa fase mais inicial do trabalho o que tornou a tarefa mais difícil por não estarmos familiarizados com a sintaxe e funcionamento da linguagem de programação utilizada.

# 5 Conclusão

Na realização da primeira fase de implementação aprendemos os conceitos básicos do NetLogo[1]. Já na segunda fase utilizámos este conhecimento para tornar o projeto mais complexo.

Pensámos que atingimos todos os objetivos propostos para este trabalho.

# 6 Referências Bibliográficas


[1] Wilensky, U. (1999). NetLogo.
https://ccl.northwestern.edu/netlogo/
Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
Acedido a 5 de outubro de 2022

[2] Wilensky, U. (1999). NetLogo 6.3.0 User Manual.
https://ccl.northwestern.edu/netlogo/docs/
Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston, IL.
Acedido a 5 de outubro de 2022

[3] StackOverflow. (2008). https://stackoverflow.com/
Acedido a 5 de outubro de 2022

# A Anexos

Duplo clique para fazer download
Simulação do Aquário (Aquarium_2.nlogo):