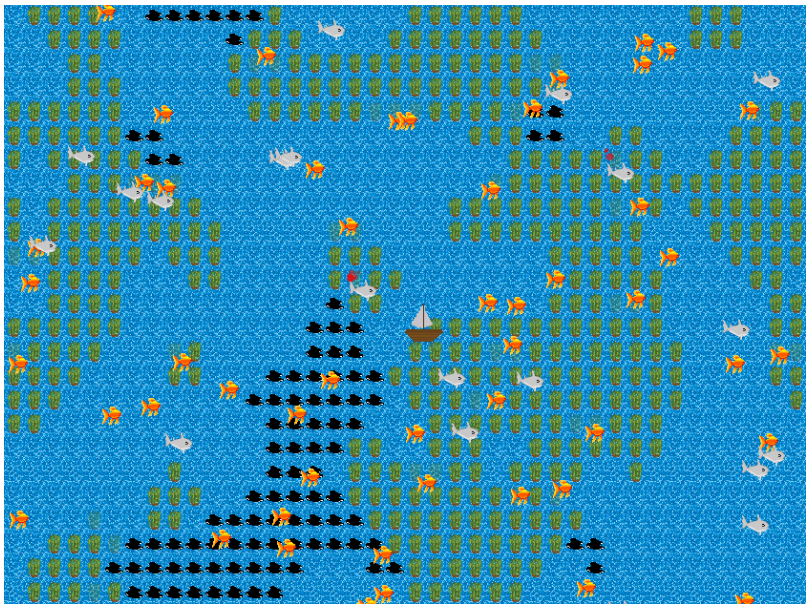


# PROJETO FINAL

Modelação e Simulação de Sistemas Naturais

47094 Pedro Azevedo
48960 Ricardo Pedro
Engenheiro Arnaldo Abrantes
30 de janeiro 2022

Licenciatura Engenharia Informática e Multimédia

Instituto Superior de Engenharia de Lisb

# Índice

1. Introdução 3
   1.1. Objetivo 3
   1.2. História 3
2. Desenvolvimento 4
   2.1. Diagrama UML 4
   2.2. Terreno do ecossistema 5
   2.3. População do ecossistema 6
   2.4. Interface da app 9
3. Resultados 11
4. Conclusão 12
5. Webgrafia e Bibliografia 13

Figura 1 - Diagrama UML 4
Figura 2 - Exemplo de Terreno 6
Figura 2. Ecrã inicial da app 9
Figura 3. Interface da app 10
Figura 4. Ecrã de sucesso 10

Listagem 1 - método getImgs() 5
Listagem 2. Eat() do predador 7
Listagem 3. Eat() no population 7
Listagem 4. Look dos predadores 8
Listagem 5. Cálculo da velocidade do barco 8
Listagem 6. Display de animais através de imagens 9

Tabela 1 - Estados das células do terreno 5
Tabela 2 - Caracteristicas das presas 6
Tabela 3 - Caracteristicas dos predadores 7
Tabela 4 - Caracteristicas do Barco 8

# 1. Introdução

Durante o semestre que passou, foram lecionados vários temas tais: Modelos quantitativos e qualitativos, com exemplos de ciclos causais e *stocks* e *flows*, sistemas, passando depois para Agentes autónomos, física e movimento, de seguida estudamos dinâmica de sistemas, caos e fractais terminando com Ecossistemas e a evolução em populações.

Com estes conhecimentos adquiridos conseguimos criar um ecossistema onde desenvolvemos um terreno utilizando agentes autómatos e autómatos celulares, simulamos espécies de seres vivos utilizando *boids* e física, aplicando comportamentos aos mesmos, sendo que com isto conseguimos analisar a evolução do sistema.

Para haver um ecossistema equilibrado é necessário haver certas variáveis onde permite haver evolução de espécies de forma que não haja uma sobrepopulação ou que haja extinção.

## 1.1. Objetivo

Utilizando todo o conhecimento e experiências obtidas ao longo do semestre, o objetivo, neste projeto final, é tirar partido de todas as ferramentas e bibliotecas desenvolvidas e aprimoradas em aula, de maneira a criar um jogo/simulação através de uma narrativa/história criado originalmente pelos alunos. Para além disso, documentar todos os resultados e discussões dos mesmos através de diferentes meios de informação.

## 1.2. História

Num certo lago ao pé de uma vila piscatória, existia uma espécie de peixe herbívora que se alimentava de algas e os residentes da vila pescavam estes peixes. Entretanto, duas tragédias aconteceram, houve um derrame de uma substancia que degrada madeira e que afeta os peixes, e para além  disso, houve uma praga de peixes selvagens. Estes tubarões estão se a alimentar dos peixes, colocando esta espécie em via de extinção! O objetivo do utilizador é apanhar todos os peixes da espécie invasiva para os peixes nativos possam sobreviver.

O barco tem certas limitações, este tem combustível limitado, que pode ir “enchendo” ao apanhar os peixes selvagens. Para além disso, a substância derramada na água, faz danos no barco, sendo que este vai perdendo *HP*, e ao ir perdendo vai perdendo velocidade.

Ambas as espécies têm que comer para ganhar energia, esta energia que consoante se movem vão perdendo-a.

Esta experiência pode terminar bem sucedida capturando todos os predadores, ou sem êxito ao ficar sem barco ou sem presas no ecossistema.

# 2. Desenvolvimento

## 2.1. Diagrama UML

Para este projeto começamos por fazer um diagrama UML como mostra a figura 1.

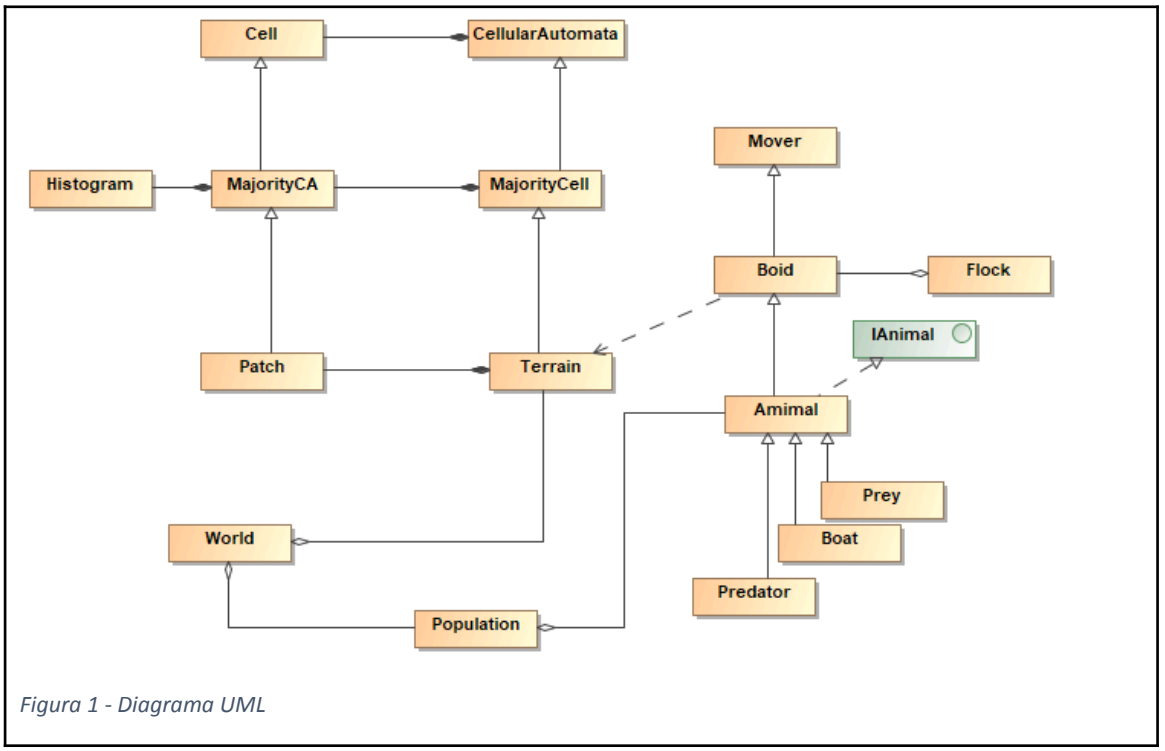



*Figura 1 - Diagrama UML*

Ou seja, para fazer o terreno vamos precisar de Autómato Celular, e para cada animal vamos precisar de *Boids*. A população vai ser o conjunto de todos os animais presentes.

## 2.2. Terreno do ecossistema

Para inicializar o ecossistema, começamos por desenvolver o terreno. O terreno é feito a partir de um autómato celular que implementamos no Trabalho 1. Cada célula vai ter um estado aleatório, sendo que existem 4 estados: *empty, obstacle, fertile* e *food*. A célula que estiver no estado fértile, passado 5 a 10 segundos, passa para o estado de comida. Quando uma presa passa por uma célula no estado de comida, esta passa para o estado fértile. Inicialmente, para representar cada estado, utilizou-se uma cor, de seguida atribuímos uma imagem a cada estado. Para tal, implementamos os métodos *getImgs() e setImgs(),* onde se vai retirar na class *WorldConstants* os *paths* das imagens do terreno, fazendo o seu *load*. Como verificamos na Listagem 1.

```
// Método para obter o array com os paths das imagens do terreno
private PImage[] getImgs(PApplet p) {
    PImage[] imgs = new PImage[WorldConstants.NSTATES];
    for (int i = 0; i < WorldConstants.NSTATES; i++) {
        String path = "imgs//" + WorldConstants.TERRAIN_IMGS[i];
        imgs[i] = p.loadImage(path);
    }
    return imgs;
}
```

*Listagem 1 - método getImgs()*

Para que o terreno fique uniforme e não haver células num estado diferente que todos os seus vizinhos, aplicamos a regra da maioria, onde quando se faz uma iteração, o estado seguinte de cada célula vai ser o estado que está mais frequente nos seus vizinhos. No caso deste terreno é feito 3 iterações. Mas ainda antes é necessário atribuir um estado a cada célula, e como já foi referido , é aleatório entre os 4 estados, mas, atribuímos percentagens para haver estados mais frequentes que outros. Na tabela 1, consegue-se observar os 4 estados com as respetivas imagens e percentagens, e na figura 2 um exemplo de terreno que se obteve.

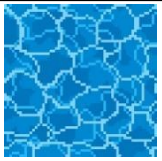
| *Empty* (vazio) | *Obstacle* (obstáculo) | *Fertile* (fétile) | *Food* (comida) |
|---|---|---|---|
|  |  |  |  |
| 30% | 20% | 10% | 40% |

*Tabela 1 - Estados das células do terreno*

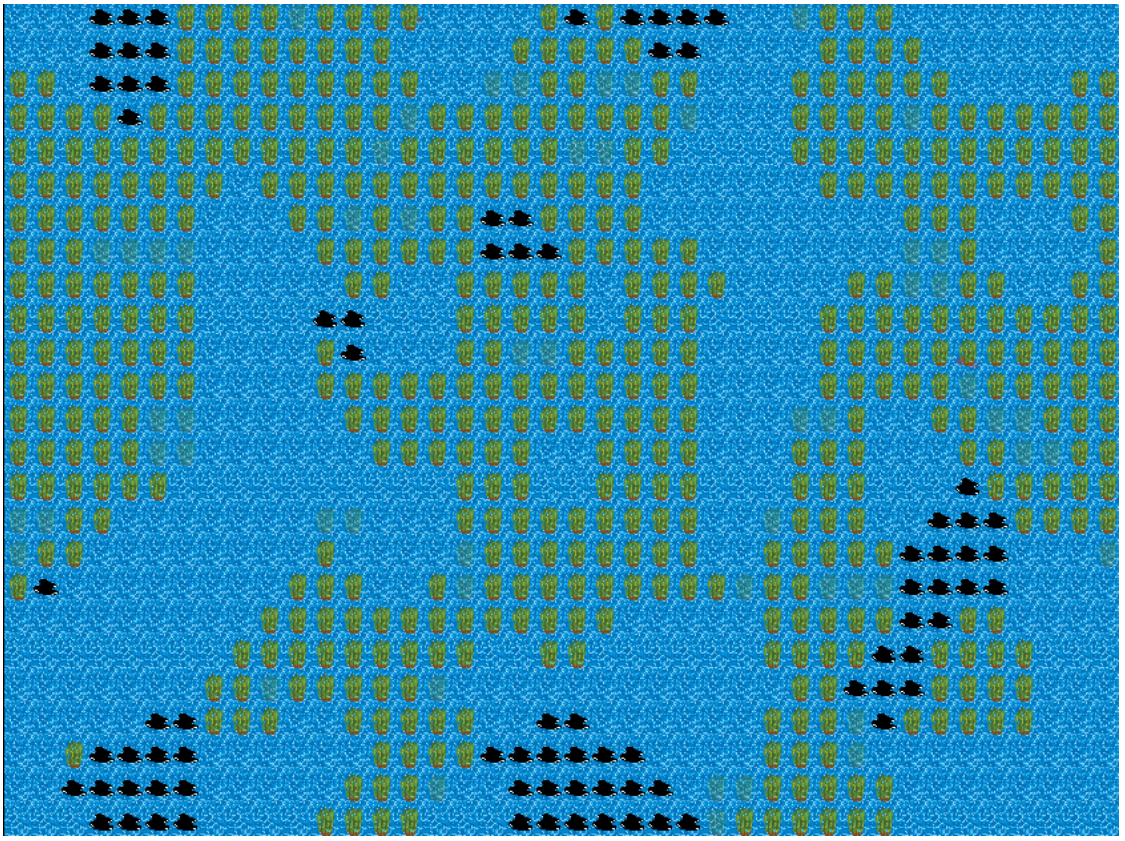


*Figura 2 - Exemplo de Terreno*

## 2.3. População do ecossistema

Em relação à população, começamos por criar as **presas** considerando a sua energia de reprodução, de modo a termos uma espécie que acabe por se extinguir (sendo o objetivo da experiencia salvar os mesmos), mas ao mesmo tempo que dure um determinado tempo para conseguirmos capturar os predadores antes que as presas morram todas.

*Tabela 2 - Caracteristicas das presas*

| Características das presas | | | | | |
|---|---|---|---|---|---|
| **Tamanho** | **Massa** | **População** | **Energia Inicial** | **Energia do eat** | **Reprodução** |
| 0.2 | 1 | 55 | 10 | 4 | 25 |

Estas presas começam com uma energia de 10, ganhando 4 por planta e precisando de 25 para se reproduzir, com a característica de perder pouca energia nas poças de óleo.

Em anexo segue o vídeo *testes1.mkv* com a demonstração do ecossistema apenas com a espécie em risco.

Na criação dos **predadores**, é tido em conta que estes recebem energia para reprodução ao alimentarem-se das presas calculando a distância dos predadores para com as presas, e se o

mesmo estiver a uma distância que consiga comer a presa, é retirado das respetivas listas a que pertence e é simulado um efeito de sangue através de um *ParticleSystem*.

```
public Boid eat(List<Body> allPrey) {

    //Verifica se o predador consegue comer alguma presa, removendo-a da lista e ganhando energia
    for(Body presa : allPrey) {

        PVector auxThis = new PVector(pos.x + (this.radius), pos.y - (this.radius));
        PVector auxPresas = new PVector(presa.pos.x + (presa.radius), presa.pos.y - (presa.radius));

        PVector sub = PVector.sub(auxThis, auxPresas);

        if (sub.mag() <= 0.2) {
            energy += WorldConstants.ENERGY_FROM_PREY;
            return (Boid)presa;
        }
    }
    return null;
}
```

*Listagem 2. Eat() do predador*

```
//Remove a presa retornada do eat da lista
for (Body pred : allPred) {

    Boid aux = ((Animal) pred).eat(allPrey);

    allPrey.remove(aux);
    allAnimals.remove(aux);

    if (aux != null) {
        //Se morreu alguma presa faz um particle system
        PSControl psc = new PSControl(velParams, lifetimeParams, radiusParams, flow, cor, false);
        ps = new ParticleSystem(pred.pos, new PVector(0, 0), 1f, .2f, psc);
        pss.add(ps);
    }
}
```

*Listagem 3. Eat() no population*

| Características dos predadores | | | | | |
|---|---|---|---|---|---|
| **Tamanho** | **Massa** | **População** | **Energia Inicial** | **Energia do eat** | **Reprodução** |
| 0.3 | 0.4 | 15 | 50 | 25 | 200 |

*Tabela 3 - Caracteristicas dos predadores*

Estes predadores têm uma longa duração de vida de maneira a morrerem maioritariamente através do barco controlado pelo utilizador. Ou seja, começam com muita energia (50), recebendo 25 de energia por cada presa que consomem, precisando de 200 para se reproduzirem. Têm também a característica de perder mais energia ao passar por um obstáculo que os peixes.

Em relação aos seus comportamentos, se ao fazer look em cada move não existir nenhuma presa no seu *near sight* irá fazer wander, se não irá fazer seek dessa presa, com o facto de que estes predadores têm uma velocidade maior relativamente às presas, devido à sua menor mass.

```
private void move(Terrain terrain, float dt) {
    for (Body a : allPrey)
        ((Animal) a).applyBehaviors(dt);

    // Atualiza o target de cada predador a cada move para verficiar o near sight
    for (Body pred : allPred) {
        Boid aux = (Boid)pred;
        aux.getEye().look();
        // Se não tiver nenhum no near sight, faz wander
        if (aux.getEye().getNearSight().size() == 0) {
            aux.applyBehavior(0, dt);
        } else {
            aux.applyBehavior(1, dt);
        }
    }

    //Aplica o seek a cada move
    boat.applyBehaviors(dt);
}
```

*Listagem 4. Look dos predadores*

Apesar da sua baixa população, devido à sua maior velocidade relativamente às presas e o seu comportamento de seek com essa espécie em risco, facilmente conseguem dominar o ecossistema eliminando todas as presas.

Em anexo segue o vídeo *testes2.mkv* com a demonstração do ecossistema entre as presas e os predadores.

Por fim nesta população, contamos com o ***Boat*** que estende também *Animal* de modo a utilizar e fazer overwrite dos métodos do mesmo (*IAnimal*). Ao contrário dos outros animais, o barco tem dois atributos, a gasolina e a sua vida. Este é capaz de capturar os predadores da mesma maneira que os predadores fazem com as presas, mas neste caso obtém gasolina. Em relação à sua vida, este ao passar por obstáculos perde vida, estando a vida deste relacionada com a sua velocidade, quanto menor a sua vida, menos veloz é.

```
@Override
public void energy_consumption(float dt, Terrain terrain) {
    gas -= dt; // basic metabolism
    gas -= mass * Math.pow(vel.mag(), 2) * dt;
    Patch patch = (Patch) terrain.world2Cell(pos.x + (this.radius), pos.y - (this.radius));
    if (patch.getState() == WorldConstants.PatchType.OBSTACLE.ordinal()) {
        hp -= 20 * dt;

        if (this.getDna().getMaxSpeed() > 1.5)
            this.getDna().setMaxSpeed((this.hp * WorldConstants.INIT_BARCO_SPEED) / WorldConstants.INI_BARCO_HP);
    }
}
```

*Listagem 5. Cálculo da velocidade do barco*

| Características do boat | | | | | |
|---|---|---|---|---|---|
| **Tamanho** | **Massa** | **Vel inicial** | **Vida inicial** | **Gas inicial** | **Gas do eat** |
| 0.4 | 1 | 4 | 50 | 200 | 25 |

*Tabela 4 - Caracteristicas do Barco*

Este faz o comportamento seek até ao rato do utilizador (target), onde pode acabar por afundar, terminando a experiência, ao ficar sem gasolina ou vida.

Todos estes animais foram implementados visualmente através de imagens com a criação de construtores que as suportam, fazendo do display das mesmas dependendo do raio do boid.

```
//Se foi usado o construtor com imagem, faz display dela
if (img != null){
    p.image(img, pp[0], pp[1], 100*radius, 100*radius);
    return;
}
```

*Listagem 6. Display de animais através de imagens*

## 2.4. Interface da app

Ao iniciar a aplicação é mostrado ao utilizador a nossa missão, assim como os animais pertencentes ao nosso ecossistema, estes são animados, saindo e voltando da nossa *window*.

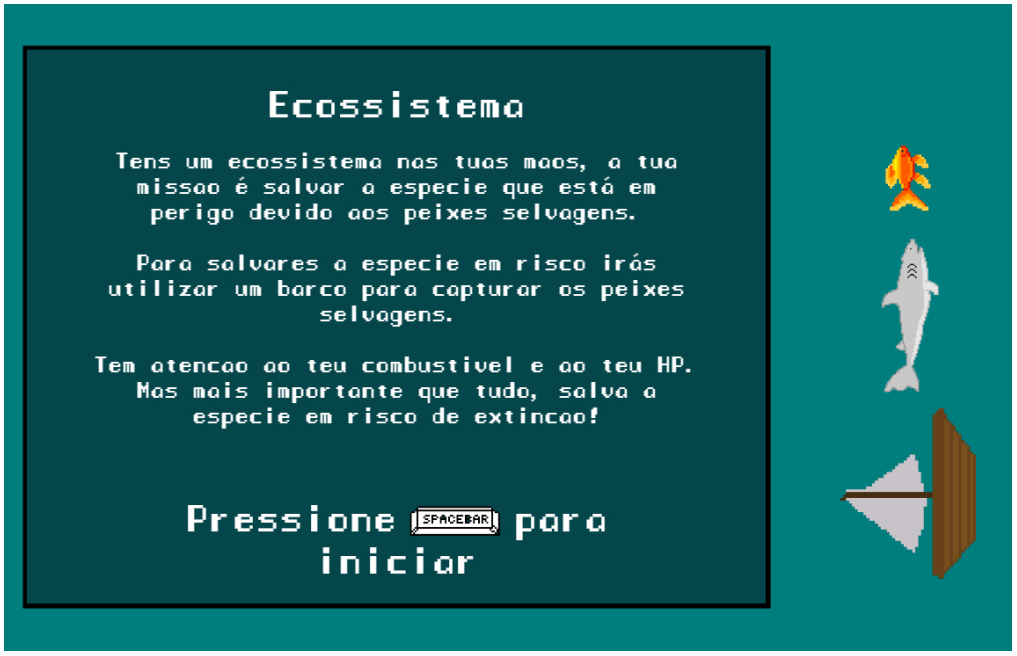



*Figura 2. Ecrã inicial da app*

Dentro da experiência é visualmente possível ver quantas presas e predadores temos atualmente no ecossistema, assim como a sua gasolina e vida através de barras visuais, fazendo o map destes atributos para um *p.rect()*.

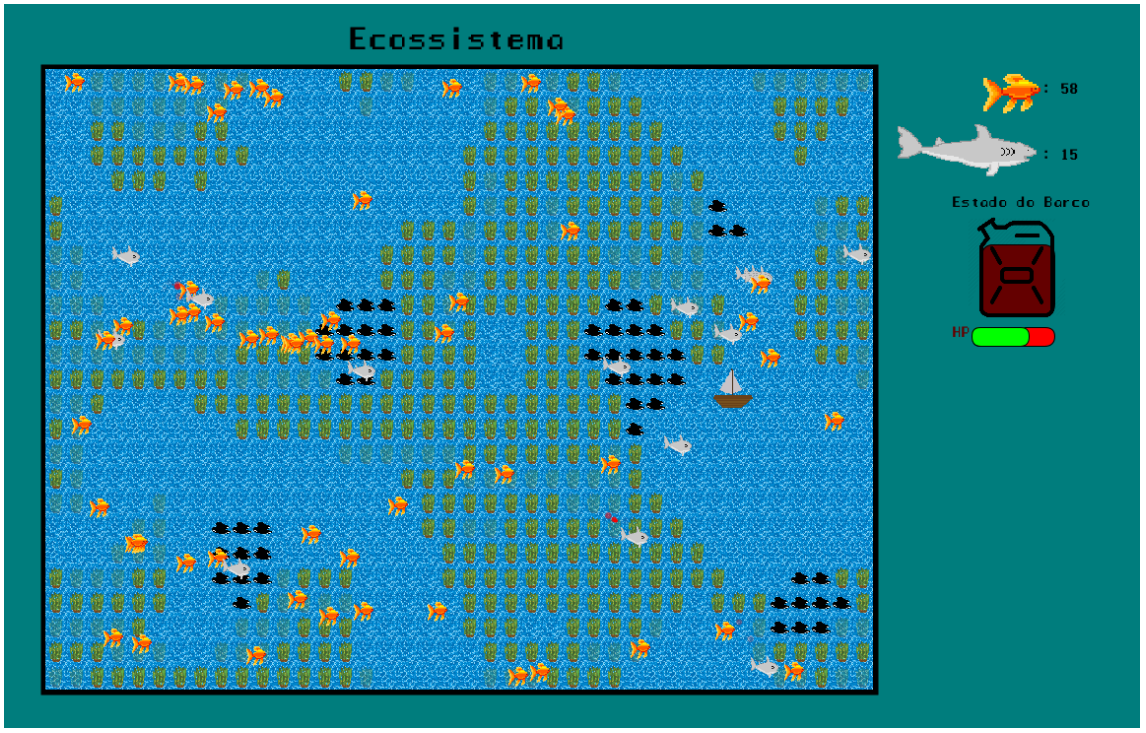



*Figura 3. Interface da app*

Ao perder ou vencer a experiência, é mostrado ao utilizador o motivo, podendo voltar a iniciar esta simulação.

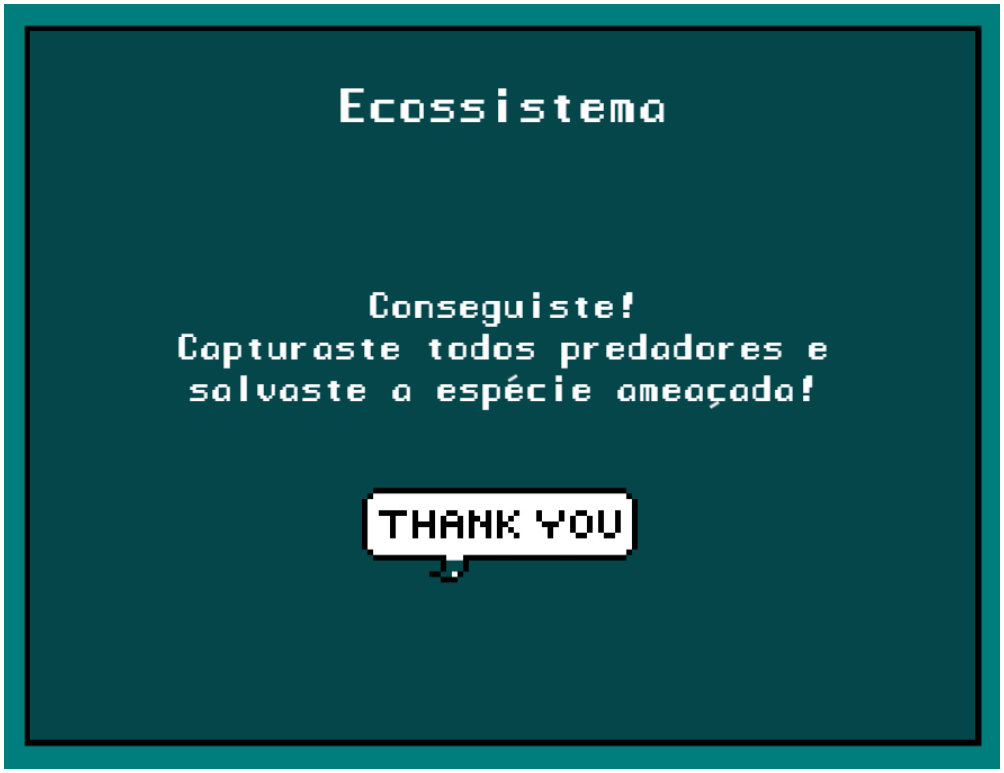



*Figura 4. Ecrã de sucesso*

# 3. Resultados

Inicialmente, como visível no vídeo *testes1.mkv* se tivermos apenas as presas no ecossistema esta espécie não se extingue, devido às propriedades de reprodução, alimento inicial presente no ecossistema, como também o facto de não perderem tanta energia ao passar por um obstáculo comparando com os predadores.

De seguida, ao adicionarmos os predadores no ecossistema, estes causam uma grande destabilização, provocando a extinção das presas. Com as variáveis atribuídas à espécie selvagem, não era possível ganhar o jogo, pois estes predadores conseguiam sempre reproduzir-se dizimando a espécie das presas, sem dar tempo de capturar os mesmos. Assim, tivemos que alterar não só a velocidade dos mesmos, como a sua população e a energia necessária para reprodução, dando mais tempo ao utilizador para capturar todos os predadores havendo uma maior oportunidade de sucesso.

Por fim, através de vários testes, após as modificações referidas no parágrafo anterior, conseguimos verificar que se não conseguirmos apanhar a espécie invasiva, esta acaba por comer os peixes todos, mas por outro lado que também é possível obter sucesso, este que acaba também por depender um pouco da criação do ecossistema, maioritariamente no nível de obstáculos existente.

Em anexo segue o vídeo *testes3.mkv* com a demonstração dos vários resultados.

# 4. Conclusão

Os conceitos dados nesta cadeira permitiu-nos entender importância no desenvolvimento de agentes que podem desempenhar comportamentos individuais e em grupo, assim como o seu funcionamento e interações entre os mesmos.

No geral, obtemos um maior conhecimento na criação de simulações em *Processing*, utilizando contextos reais e bem conhecidos como o Jogo da Vida, DLA (*Diffusion-limited Aggregation*), Gramáticas de *Lindenmayer* assim como Conjuntos de *Julia* e *Mandelbrot* e ecossistemas.

Concluímos também que a simulação é fundamental pois só mapas causais ou modelos puramente conceptuais não é suficiente, pois não temos capacidade mental de simular a dinâmica de sistemas complexos não lineares. Estas simulações computacionais, conseguem nos apoiar na criação de uma intuição e melhorar a nossa capacidade mental de simulação.

# 5. Webgrafia e Bibliografia


- 2019, Arnaldo Abrantes e Paulo Vieira, Evolução em populações