AQUÁRIO VIRTUAL: CICLO REPRODUTIVO OVÍPARO

Carlos Eduardo Machado, Dalton Solano dos Reis – Orientador

Curso de Bacharel em Ciência da Computação
Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brasil

carmachado@furb.br, dalton@furb.br

Resumo: *Este artigo apresenta o desenvolvimento de uma extensão de um simulador de aquário virtual através da adição de um ciclo reprodutivo ovíparo aos peixes e o treinamento utilizando aprendizagem de máquina. Para o desenvolvimento do trabalho, foi utilizado o motor gráfico Unity em conjunto com a biblioteca responsável pela aprendizagem de máquina, Unity Machine Learning Agents. Os objetivos foram alcançados, porém a movimentação gerada pela aprendizagem de máquina se torna repetitiva, pois a Unity Machine Learning Agents busca recompensas com o máximo de eficiência.*

Palavras-chave*: Simulador. Aquário virtual. Aprendizado de máquina. Reprodução ovípara. Animação comportamental.*

1 INTRODUÇÃO

Segundo Machado (2016), o uso de simuladores na educação como ferramenta tecnológica pode reforçar a ação docente em sala de aula de modo a favorecer colaborativa e substancialmente a aprendizagem significativa dos conteúdos escolares. Uma das formas de explorar o potencial dos simuladores são os simuladores de ecossistemas.

De acordo com Stein (2018), um ecossistema pode ser definido como uma unidade biológica completa, tendo todos os componentes físicos e biológicos necessários para a sobrevivência. Nele atuam dois fatores, os fatores abióticos, componentes não vivos, como temperatura, água, luminosidade, e fatores bióticos, componentes vivos, como animais e plantas.

Um dos mais importantes elementos dentre os componentes vivos é a capacidade de gerar descendentes, pois espécies incapazes de reproduzirem serão extintas do planeta. Entre os métodos de reprodução, organismos multicelulares, como peixes, costumam majoritariamente reproduzir de maneira sexuada, isso é, os descendentes são gerados a partir de duas células diferentes oriundas dos pais. Entre os métodos de fecundação o mais usual é o ovíparo, no qual os gametas masculinos e femininos são liberados na água (CASTRO; HUBER, 2012).

Losada (2019) desenvolveu um aquário virtual capaz de simular o comportamento de peixes utilizando o recurso de Interface de Usuário Tangível (IUT). A IUT possibilitou que sensores externos alterassem fatores internos, como a temperatura, a luminosidade e alimentação dos peixes. Dessa forma, conforme os sensores externos são alterados os peixes são afetados, podendo morrer dependendo das condições do aquário.

A partir deste contexto, o presente trabalho desenvolveu um ciclo reprodutivo virtual capaz de passar por todas as etapas de vida dos peixes, desde a desova até a morte natural do peixe e a adição desse ciclo ao trabalho desenvolvido por Losada (2019) para auxiliar professores no ensino do ciclo de vida marinho.

2 FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta os aspectos da fundamentação teórica utilizados para o desenvolvimento deste trabalho. A subseção 2.1 aborda o tema ciclo de vida reprodutivo dos peixes. Na subseção 2.2 é apresentado o conceito de animação comportamental. Por fim, na seção 2.3 explica sobre a biblioteca Unity Machine Learning Agents.

2.1 CICLO DE VIDA DOS PEIXES

O sistema reprodutivo dos peixes é sexuado. Alguns peixes copulam durante a transmissão do esperma, porém a liberação dos ovos e gametas na água é a forma mais comum. Para que ocorra a fecundação é necessário que os dois sexos liberem os gametas ao mesmo tempo. Essa sincronia acontece graças aos hormônios sexuais dos peixes (CASTRO; HUBER, 2012).

Como descrito por Benedito (2015), no período reprodutivo os peixes recolhem diversas informações sobre o ambiente através de seu sistema sensorial e verificam se as condições ambientais para reprodução estão ótimas. Entre as informações recolhidas pelos peixes, Benedito (2015) menciona a temperatura, a velocidade da água, a profundidade, o local adequado para desova e a presença de parceiros do sexo oposto. Caso as condições estejam ótimas ocorre a liberação de hormônios que desencadeia na desova.

Segundo Benedito (2015, p. 52), “O desenvolvimento do ovo pode ser dividido em clivagem inicial (formação das primeiras células), embrião inicial (diferenciação do embrião), cauda livre (desprendimento da cauda do vitelo) e embrião final (pronto para eclosão). [...]”.

De acordo com Bonecker (2014), o desenvolvimento do peixe após eclosão do ovo é dividido em três estágios, o larval, o larval em estágio de transformação e o juvenil. O estágio larval começa a partir do momento em que o saco vitelino é completamente absorvido e termina quando o peixe completa a flexão da notocorda. No estágio larval em estágio de transformação ocorre mudanças na forma do peixe, deixando de ter características de larvas e se assemelhando mais aos adultos. No estágio juvenil o peixe é morfologicamente similar ao adulto, possuindo nadadeiras e escamas formadas. Segundo Benedito (2015) o estágio juvenil estende-se até a primeira maturação sexual. Após o estágio juvenil o peixe se torna um adulto completamente desenvolvido.

2.2 ANIMAÇÃO COMPORTAMENTAL

Segundo Yu (2007), a animação comportamental busca fornecer animações mais detalhadas para os agentes. Normalmente durante a criação de animações para personagens são definidos todos os movimentos a serem executados, porém utilizando o conceito de animação comportamental os agentes interpretam os comportamentos para assim gerar a ação final. De acordo com Mendonça (1999 apud FELTRIN, 2014, p. 17), animação comportamental é definida por conter uma cena com personagens e objetos com comportamentos próprios, esses sendo capazes de alcançar objetivos. De acordo com Reynolds (1997), animação comportamental permite que personagens autônomos decidam suas próprias ações, dando a possibilidade de improvisação sem a necessidade de gerar animações para cada movimento do agente.

Segundo Piske (2015), a animação comportamental possui conceitos semelhantes aos de agentes inteligentes: percepção, raciocínio e ação. De acordo com Russel (2013) a percepção corresponde às entradas perceptíveis pelo agente através de sensores. O segundo conceito é o de raciocínio, que é a capacidade de decidir a ação a ser tomada baseado no conhecimento prévio do agente. Por fim o conceito da ação que trata de um comportamento de resposta a percepção através dos sensores.

Thalmann (1990) explica um dos métodos de animação comportamental, a animação de nível de tarefas. Nesse modelo tarefas são decididas pelo agente e essas são transformadas em pequenas ações a serem executadas. São exemplos de tarefas andar de um ponto a outro e pegar um objeto e move até outro lugar. Thalmann (1990) descreve que mesmo no uso da animação de nível de tarefas, os objetos animados devem ser autônomos.

2.3 UNITY MACHINE LEARNING AGENTS TOOLKIT

De acordo com Juliani (2018), o Unity Machine Learning Agents Toolkit, abreviado como Unity ML-Agents é um projeto de código aberto onde desenvolvedores podem criar simulações utilizando o Unity e interagir com elas através de uma Application Programming Interface (API) em Python. Segundo Juliani (2018) o toolkit fornece um Software Development Kit (SDK) com as funcionalidades necessárias para a construção de um ambiente de aprendizagem de máquina.

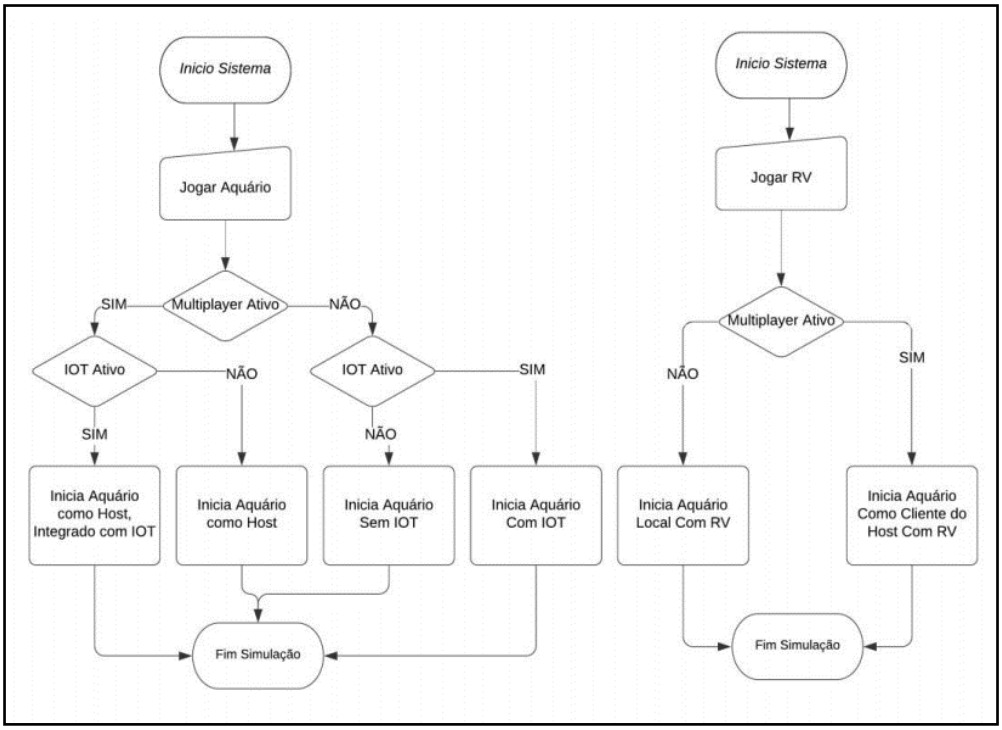
Como descrito por Juliani (2018), o cerne do SDK do Unity ML-Agents contém três entidades, o `Sensor`, o `Agent` e a `Academy`. O componente `Agent` indica que o objeto é um agente, e esse pode adquirir informações do cenário através de seus sensores, executar ações e receber recompensas através do sistema de recompensa. Antes e após a execução de uma ação, o agente armazena as informações dos sensores juntamente com a quantidade de recompensa adquirida, dessa forma ele é capaz de analisar o impacto de sua ação no ambiente e utilizar esse aprendizado em suas próximas ações. Cada agente possuí um nome comportamental, esses nomes permitem que agentes sejam separados em tipos, com isso é aberta a possibilidade do treinamento de vários agentes em grupos diferentes.

Juliani (2018) defini que a entidade `Academy` é responsável por gerenciar os agentes e acompanhar o número de passos deles. Ela também é capaz de alterar variáveis ambientais em tempo de execução. O componente `Sensor` é responsável por coletar informações do cenário para os agentes, ele é capaz de coletar imagens, resultados de ray-cast e parâmetros arbitrários.

2.4 VERSÃO ANTERIOR DO SISTEMA

Atualmente o simulador de ecossistema marinho encontra-se em duas plataformas, no formato de Aquário Virtual (LOSADA, 2019) e Realidade Virtual (SILVA, 2020). Ao iniciar a aplicação é possível definir algumas opções do simulador. As opções são: jogar o Aquário Virtual ou em Realidade Virtual; ativar o multiplayer; e caso selecionado aquário virtual se o aquário está conectado à Internet of Things (IoT) (Figura 1). O projeto foi desenvolvido utilizando Unity em conjunto com o asset AIFishes que disponibilizou os modelos 3D dos peixes e cenários, assim como comportamentos de movimentação padrões para os peixes.

Figura 1 – Fluxograma de inicialização do aquário virtual



Fonte: Silva (2020).

Quando iniciado em `aquário com IoT`, o sistema irá funcionar como um aquário com Interface de Usuário Tangível (IUT). Com ela é possível alimentar o peixe através de um botão, captar a luz ambiente, definir a temperatura através de um potenciômetro e LEDs responsáveis por verificar se o módulo está ligado, se está conectado a WiFi e se está conectado ao Simulador (Figura 2). Quando selecionado com `Multiplayer` ativo, o simulador permite que jogadores em Realidade Virtual entrem no aquário como sendo um dos peixes. Durante a execução em Realidade Virtual a visualização do aquário ocorre através da visão do peixe, utilizando o conceito de avatar. Além disso utilizando a câmera do aquário é possível ter a visão do peixe de fora do aquário (SILVA, 2020).

Durante a simulação são analisados a temperatura, a luminosidade do aquário, e caso estejam inadequados, a saúde dos peixes é diminuída. Quando os peixes são alimentados a saúde dos peixes é regenerada, e caso a saúde chegue a 0 o peixe morre. Quando todos os peixes morrem a simulação acaba (SILVA, 2020).

Figura 2 – Aquário virtual

Fonte: Losada (2019).

2.5 TRABALHOS CORRELATOS

Nesta subseção são apresentados trabalhos com características semelhantes aos principais objetivos do trabalho desenvolvido. O Quadro 1 aborda uma implementação de animação comportamental em um simulador já existente (ESTEVÃO, 2020), o Quadro 2 detalha o simulador de ecossistemas VisEdu (PISKE, 2015) e no Quadro 3 é apresentada uma simulação da dinâmica populacional de insetos agrícolas (TOEBE, 2014).

Quadro 1 – Análise do uso de animação comportamental com o motor de jogos Unity

Referência	Estevão (2020).
Objetivos	Adição de animais de espécies distintas em um simulador de ecossistemas já existente e a implementação de uma inteligência artificial para cada animal.
Principais funcionalidades	a) contêm os animais coelho, veado e lobo; b) o coelho e o veado se alimentam da vegetação do cenário, enquanto o lobo se alimenta de coelhos e veados. c) todos os animais podem andar e saciar a sede em um lago disposto no cenário; d) as decisões dos animais são gerenciadas pelo algoritmo de aprendizagem de máquina da ML-Agents.
Ferramentas de desenvolvimento	C#, motor gráfico Unity e a biblioteca Unity Machine Learning Agents.
Resultados e conclusões	Os modelos treinados se comportaram da maneira desejada e tornaram o simulador mais robusto. Também define a biblioteca ML-Agents como uma ferramenta completa e eficiente. Porém levantou alguns problemas encontrados durante o desenvolvimento, dentre eles o tamanho do cenário é muito pequeno, impossibilitando a adição de mais de três animais na cena. Além disso o relevo do terreno dificulta a movimentação dos agentes. Outro problema foi a dificuldade da utilização do ML-Agents para o treinamento juntamente com outra biblioteca utilizada originalmente no simulador, o Vuforia. Para resolver isso foi criado um projeto separado para o treinamento e, após aplicar essa solução, Estevão (2020) afirmou que a criação de um novo projeto facilitou o processo de treinamento dos agentes.

Fonte: elaborado pelo autor.

O trabalho de Estevão (2020) também trata de um ecossistema em ambiente Unity. Assim como o trabalho realizado, no simulador de Estevão (2020) os animais são capazes de se movimentar e se alimentar utilizando a biblioteca Unity Machine Learning Agents, porém apenas o trabalho proposto traz a possibilidade de reprodução dos agentes. Outro ponto que difere o trabalho correlato do trabalho proposto é o ambiente. O trabalho realizado utiliza o ambiente marinho, que permite a movimentação do agente em 3 dimensões. Enquanto o trabalho de Estevão (2020) utiliza um ambiente terrestre, tendo a movimentação em 2 dimensões.

Quadro 2 – VISEDU – Aquário virtual: simulador de ecossistema utilizando animação comportamental

Referência	Piske (2015).
Objetivos	Simular um ecossistema de aquário marinho.
Principais funcionalidades	a) contêm as espécies plâncton, sardinhas e tubarões; b) os tubarões se alimentam das sardinhas; c) as sardinhas devem fugir dos tubarões e se alimentar do plâncton; d) tubarões e sardinhas reproduzem a cada 20 segundos. e) interface que permite alteração de parâmetros, como velocidade e raio de visão dos peixes.
Ferramentas de desenvolvimento	No desenvolvimento do servidor do simulador foi utilizando Java com a biblioteca Jason para o desenvolvimento dos agentes. Já a parte de visualização do aquário HTML, CSS e Javascript, utilizando a biblioteca gráfica ThreeJS.
Resultados e conclusões	Os objetivos foram alcançados e que a aplicação se mostrou um ótimo ambiente para a inserção de agentes dotados de representação gráfica. Levanta que a biblioteca ThreeJS se mostrou eficiente, porém levantou que há problemas de performance na rotina de verificação de colisões. Segundo Piske (2015) a biblioteca Jason tem alguns problemas em lidar com muitos agentes, podendo perceber atrasos perceptíveis em simulações em tempo real.

Fonte: elaborado pelo autor.

Piske (2015) desenvolveu um simulador de aquário marinho que utiliza de animação comportamental para uso educacional. Os agentes foram desenvolvidos utilizando uma biblioteca diferentes para o treinamento dos agentes. Mesmo tendo uma rotina de reprodução dos peixes, essa reprodução gera novos peixes automaticamente conforme a passagem de tempo, sem o uso de animação comportamental.

Quadro 3 – Um modelo baseado em agentes para o ciclo de vida dos insetos: aplicação na interação afídeo-planta-vírus

Referência	Toebe (2014).
Objetivos	Criação de um framework de ciclo de vida de insetos pragas agrícolas de forma que fosse genérico suficiente para poder ser usado em espécies diferentes da proposta pelo artigo.
Principais funcionalidades	a) acesso a dados ambientais de interesse das simulações externos, podendo buscar dados meteorológicos on-line; b) entre os comportamentos do inseto estão alimentação, movimentação, reprodução e mortalidade; c) arquivo para parametrização do ambiente e inseto, conseguindo simular em diversas condições.
Ferramentas de desenvolvimento	Java e XML.
Resultados e conclusões	A utilização de modelagem baseada em agentes foi correta para o projeto. Porém também levanta algumas limitações do projeto, dentre eles o consumo elevado de memória e tempo de processamento elevado, principalmente quando há muitos insetos. Desse fato o autor levanta que uma possível solução seria utilizar processamento paralelo ou investimento em computadores com grande capacidade de processamento. Outra limitação é que não é possível que dois ou mais insetos causem dano a mesma planta simultaneamente. Além disso o modelo não contempla insetos com papéis sociais, que formam colônias e retornam a ela com frequência.

Fonte: elaborado pelo autor.

O projeto de Toebe (2014) consiste no desenvolvimento de um simulador de ciclo de vida de insetos pragas agrícolas. Os agentes são capazes de se alimentar, reproduzirem e morrerem por idade avançada. Diferentemente do trabalho desenvolvido o gerenciamento dos atores não é feito por uma inteligência artificial, sendo totalmente baseado em dados estatísticos. Além disso, o foco do trabalho não é educacional, e sim focado na simulação para a área agrícola. Por isso Toebe (2014) não desenvolveu uma parte gráfica para os insetos, apenas fornecendo dados estatísticos como saída.

3 DESCRIÇÃO DO APLICATIVO

Esta seção apresenta os detalhes de especificação e implementação da ferramenta desenvolvida. O conteúdo é divido em duas subseções. A primeira subseção apresenta a especificação do sistema. A segunda subseção trará detalhes da implementação e de como o sistema foi construído, tal como as tecnologias utilizadas.

3.1 ESPECIFICAÇÃO

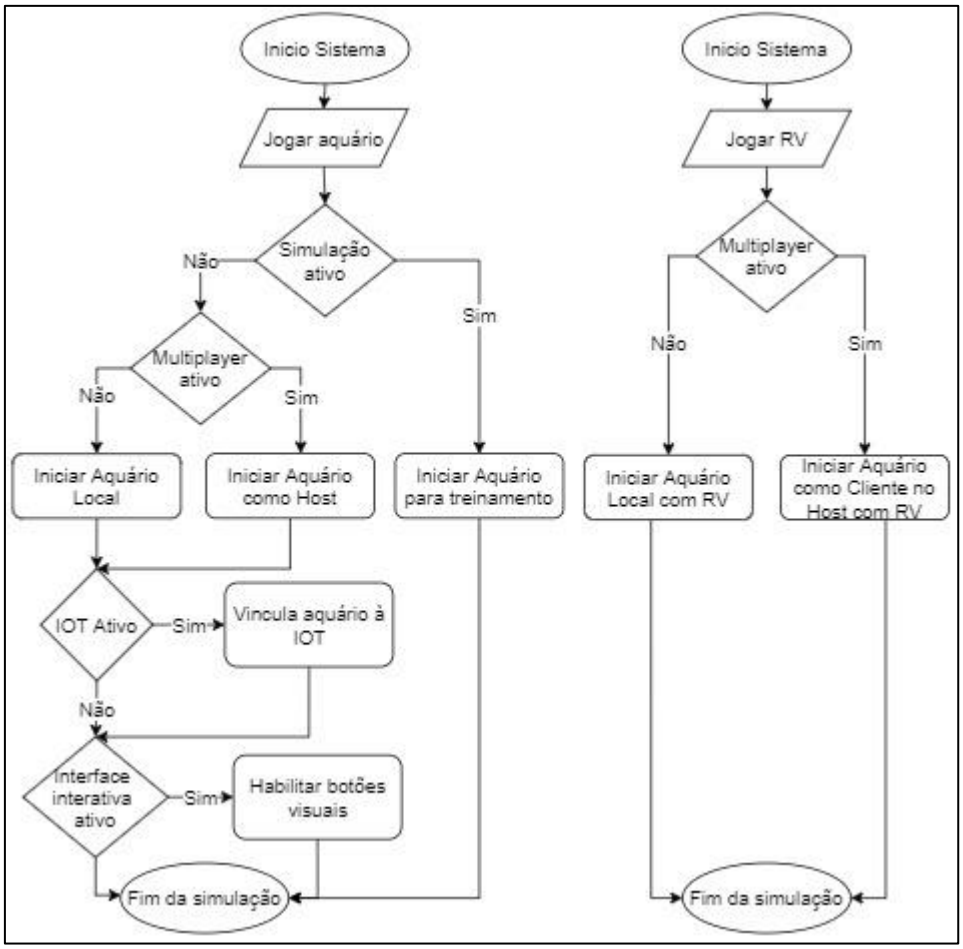
O projeto busca disponibilizar a implementação de inteligência artificial na tomada de decisões dos peixes do aquário virtual de Losada (2019), a adição de rotinas de reprodução ovípara e variação de luminosidade e temperatura adequada entre as espécies. O processo de disponibilização da inteligência artificial necessita que os peixes passem por um treinamento. O treinamento é realizado através do aprendizado por reforço profundo, onde o agente é colocado em várias situações dentro do aquário e busca alcançar a maior pontuação possível, ganhando pontos quando se alimenta corretamente, quando reproduz e quando descansa. Após o treinamento o peixe se comporta de acordo com os movimentos que o trouxeram as melhores pontuações.

Os principais Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) realizados durante o trabalho estão destacados a seguir:

a) criar peixes através da reprodução dos peixes ovíparos (Requisito Funcional - RF);
b) processar o crescimento dos peixes após a saída do ovo, desde o nascimento até a morte natural (RF);
c) definir uma idade inicial para os peixes do aquário (RF);
d) reduzir a saúde dos peixes baseado na diversidade de temperatura do aquário (RF);
e) desenvolver utilizando o motor de jogos Unity e a linguagem C# (Requisito Não Funcional – RNF);
f) utilizar o toolkit ML-Agents para o treinamento dos peixes (RNF);
g) ser compatível com o aquário virtual e com a realidade virtual (RNF).

A Figura 3 exibe a fluxo para a chamada do aquário virtual na tela inicial. As regras implementadas por Losada (2019) e Silva (2020) continuam as mesmas. A inicialização foi alterada para permitir iniciar o aquário no modo de treinamento dos peixes, também foi adicionado um tratamento para permitir a alimentação, mudança de luminosidade e variação de temperatura pela interface virtual do aquário. Ambas as opções podem ser configuradas pela tela de configuração do sistema.

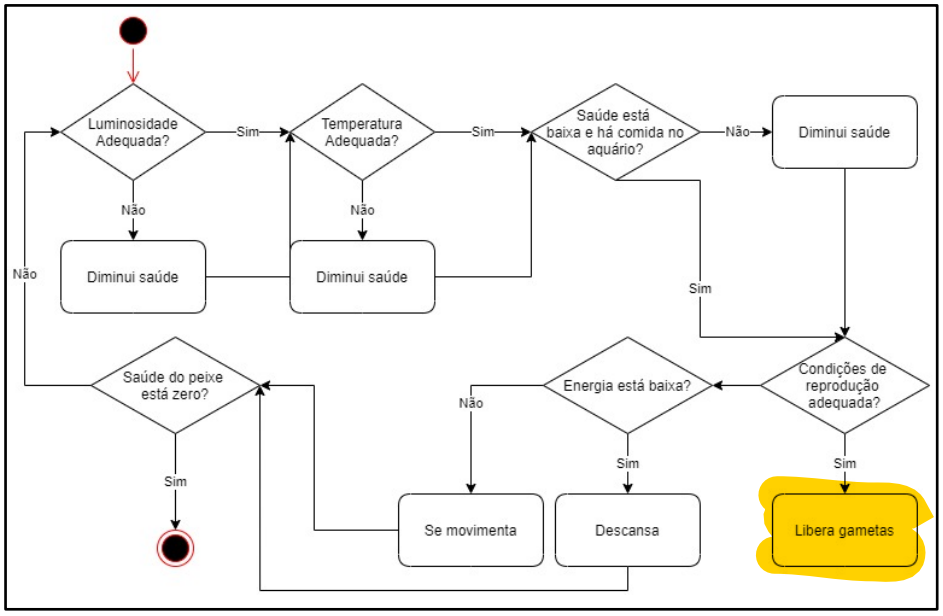
Figura 3 – Fluxograma de inicialização do Aquário Virtual.



Fonte: elaborado pelo autor.

A Figura 4 exibe o ciclo de vida do peixe com a adição da rotina de reprodução. Cada espécie de peixe tem seu tempo de vida, temperatura ideal e iluminação ideal. Caso a temperatura ou iluminação não estejam de acordo com o ideal para o peixe a saúde do peixe é reduzida, quando essa chega a zero o peixe morre. A saúde dos peixes pode ser aumentada caso com a alimentação através de comidas adicionadas pelo usuário. Conforme o peixe se move ele gasta energia e quando essa energia termina, o peixe descansa. Além disso os peixes podem se reproduzirem.

Figura 4 – Diagrama de atividade do ciclo de vida do peixe

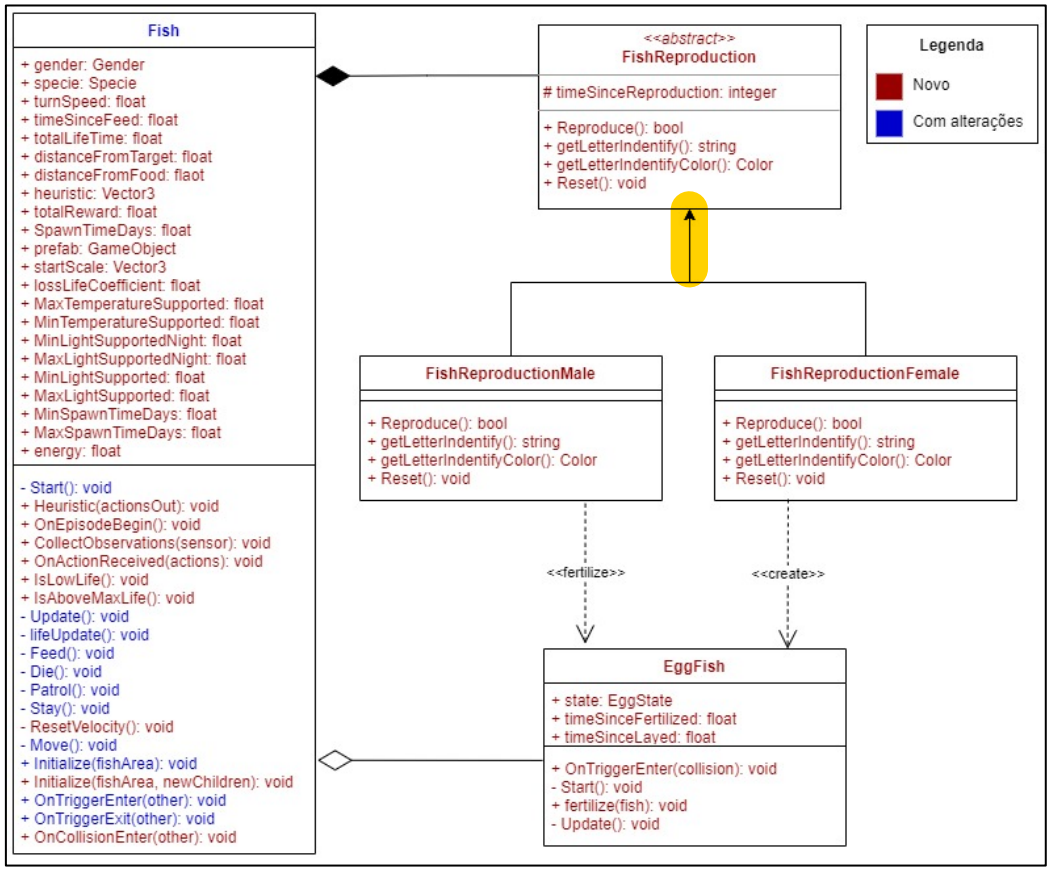


Fonte: elaborado pelo autor.

A reprodução dos peixes começa quando uma fêmea deposita um ovo no fundo do aquário. Com o ovo no fundo do aquário, os peixes machos de mesma espécie ao se aproximarem do ovo podem fertilizar o mesmo, se isso não for feito em determinado tempo o ovo desaparece. Depois de uma certa quantidade de tempo, o peixe nasce do ovo fertilizado. O tamanho do peixe recém-nascido é pequeno suficiente para caber no ovo e conforme o tempo passa ele cresce até se tornar do tamanho dos outros peixes. Quando ele se torna do tamanho dos outros peixes ele se torna capaz de reproduzir.

A Figura 5 exibe as alterações no diagrama de classes em relação a classe `Fish`, responsável pelo comportamento dos peixes. Essa, recebeu diversas alterações para utilizar o ML-Agents. Uma das rotinas mais alteradas foi a de movimentação. Ela era gerenciada pelo pacote `AIFishes` e necessitou de mudanças para ser utilizada com as rotinas de agentes. Nas propriedades dos peixes foram adicionados valores mínimos e máximos de temperatura suportada, luz suportada a noite, luz suportada de dia e tempo de vida. Caso a temperatura ou a luz do aquário esteja diferente da suportada, o peixe perde saúde. O tempo de vida do peixe é definido aleatoriamente dentro da faixa do mínimo e máximo definido para cada espécie.

Figura 5 – Diagrama de classes das alterações relacionadas ao peixe



Fonte: elaborado pelo autor.

As rotinas de reprodução foram implementadas na classe `FishReproduction` e suas classes de agregação. A criação delas acontece ao criar o peixe e é baseado no sexo do peixe. Para machos é criada a `FishReproductionMale` e para fêmeas é criada a `FishReproductionFemale`. O método `Reproduce` é responsável pela liberação dos gametas, tanto nos machos, quanto nas fêmeas.

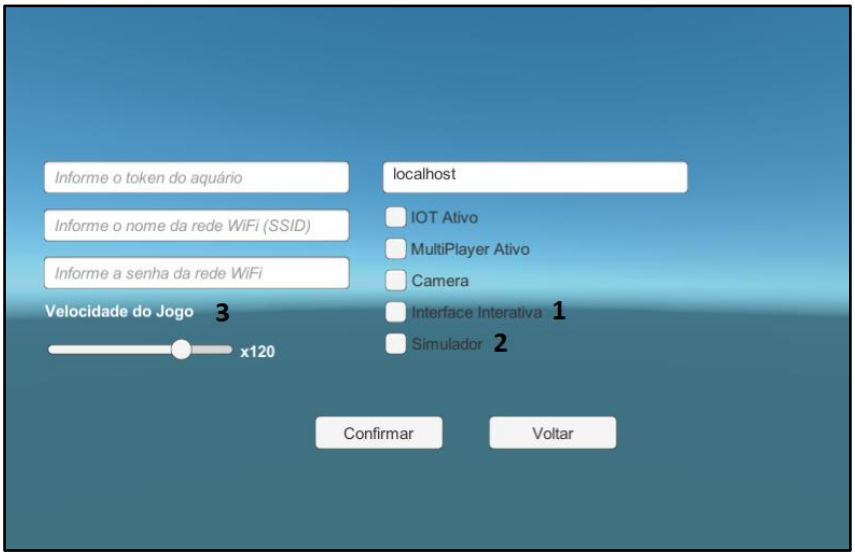
3.2 IMPLEMENTAÇÃO

Para o desenvolvimento dos novos comportamentos dos peixes foram utilizados o motor gráfico Unity, juntamente com a biblioteca Unity Machine Learning Agents. Para o treinamento dos agentes foi utilizado o Python 3.9 com a API ML Agents.

Durante a adaptação do sistema anterior foi necessário a adição de novas opções a tela de configuração do software (Figura 6). O `item 1` foi adicionado para permitir a utilização do aquário sem a necessidade da interface de usuário tangível criada por Losada (2019). O `item 2`, quando marcado abre a cena responsável pelo treinamento dos peixes. O `item 3` foi alterado para clarificar a seleção de velocidade, exibindo o quão mais rápido o simulador ficará em

relação ao tempo real. Também foi adicionada um multiplicador de velocidade, onde uma hora dentro do simulador demora dez segundos.

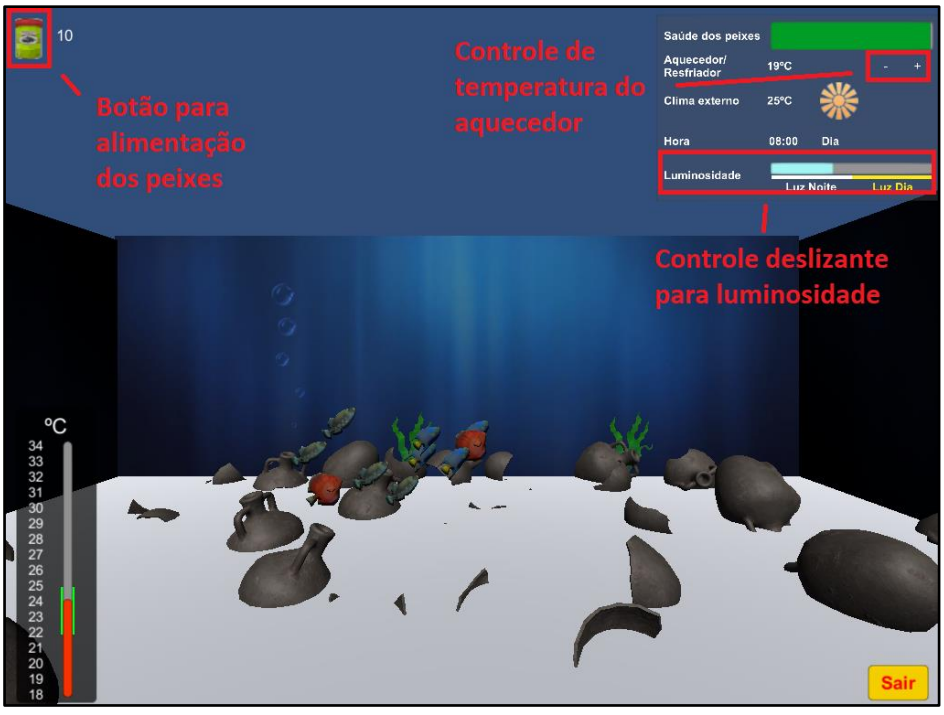
Figura 6 – Tela de configuração



Fonte: elaborado pelo autor.

Quando a opção de interface interativa é marcada na tela de configurações, o ícone que anteriormente servia apenas para representar a quantidade de comida disponível se torna interativo e permite a liberação dos alimentos dentro do aquário (Figura 7). A luminosidade pode ser definida através de um controle deslizante, esse estava presente anteriormente, mas não podia ser alterado. Para o controle do aquecedor do aquário foram adicionados botões para aumentar e diminuir a temperatura.

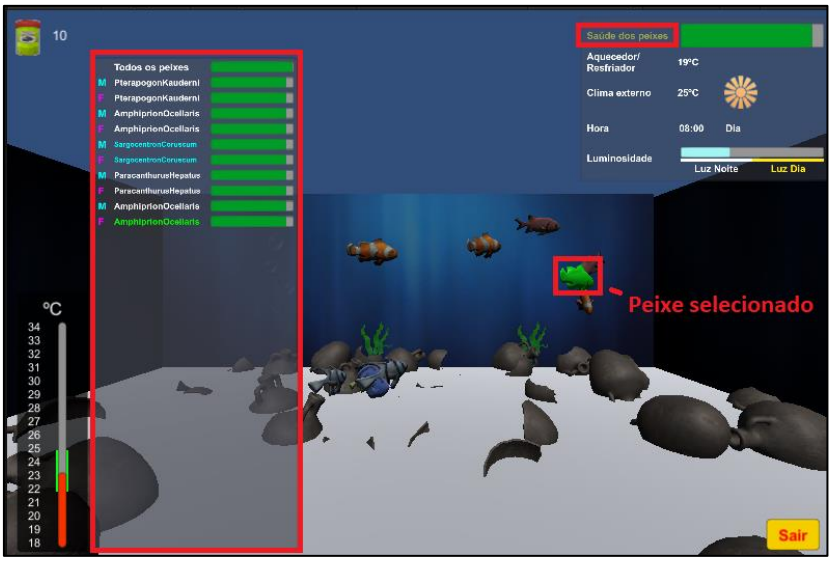
Figura 7 – Controles de tela ao marcar opção de interface iterativa



Fonte: elaborado pelo autor.

Foi adicionado uma tela para melhorar o acompanhamento individual dos peixes. Essa é aberta ao clicar na saúde dos peixes (Figura 8) e exibe o sexo, a espécie e a saúde de cada peixe. Além disso exibe a saúde média de todos os peixes do aquário. Quando a temperatura do aquário está muito baixa, a cor da espécie do peixe fica azul. Quando a temperatura está muito alta a cor da espécie fica vermelho. Nela é possível selecionar um peixe. Quando selecionado, a cor do peixe fica totalmente verde para facilitar o acompanhamento de sua movimentação dentro do aquário.

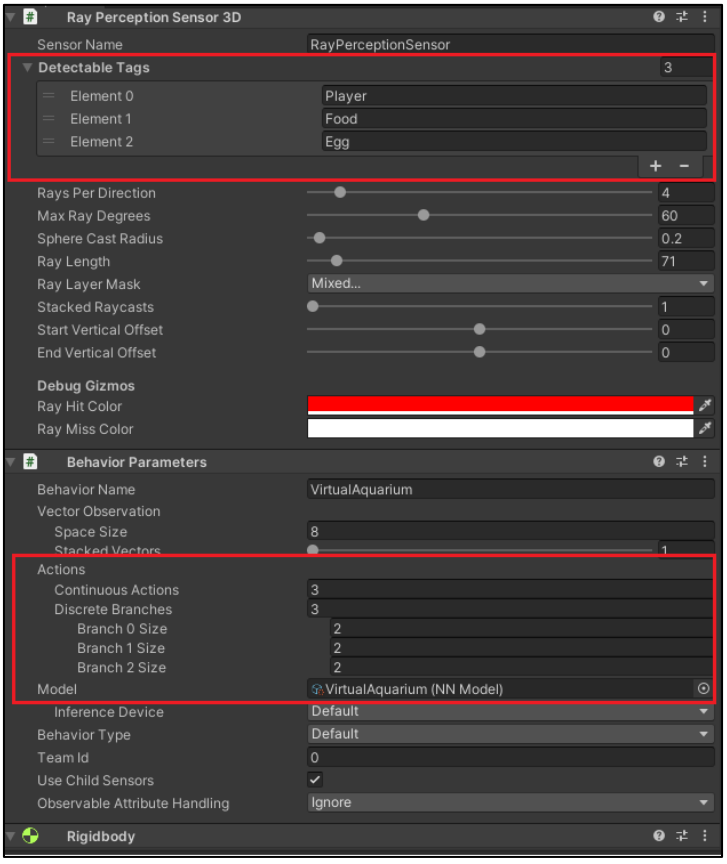
Figura 8 – Tela de acompanhamento individual dos peixes



Fonte: elaborado pelo autor.

Nas propriedades do peixe (Figura 9) foram adicionados os componentes `Behavior Parameters`, `Ray Perception Sensor 3D` e `RigidBody`. O primeiro é utilizado para definir o número de parâmetros durante a decisão do agente, no projeto ele é utilizado para a movimentação, reprodução e alimentação. Dentro do `Behavior Parameters` é possível definir o número de parâmetros a serem observados pelo agente, que serão coletados antes e depois da decisão do agente, além disso é possível definir o modelo que o agente usará na execução das ações, esse modelo é gerado durante o treinamento. O `Ray Perception Sensor 3D` é responsável pela visão do peixe, nele é possível definir quais *tags* são relevantes para a detecção do agente e estabelecer o raio e ângulo do *Ray casting*. O `RigidBody` permite o controle de movimentação e rotação do agente.

Figura 9 – Propriedades dos peixes



Fonte: elaborado pelo autor.

A movimentação dos peixes era feita pela biblioteca `AIFishes` no script `Fish`, a rotina de alimentação foi criada por Losada (2019) e só permitia um peixe se alimentar de cada vez. O script foi alterado para ser responsável pelas rotinas dos agentes. Ambas as rotinas foram reescritas para que a biblioteca ML-Agents ficasse responsável por decidir a movimentação e o momento ideal de alimentação dos peixes, com isso múltiplos peixes podem se alimentar simultaneamente. A rotina de movimentação (Quadro 4) passou a utilizar o componente `RigidBody`, dessa forma o peixe passa a respeitar os controles de física da própria Unity.

Quadro 4 – Rotina de movimentação do agente

```
// Rotaciona em direção ao objetivo
Vector3 targetDirection = target - transform.position;
float singleStep = turnSpeed * Time.deltaTime;
Vector3 newDirection = Vector3.RotateTowards(transform.forward, targetDirection, singleStep, 0.0f);
transform.rotation = Quaternion.LookRotation(newDirection);
if (playerAhead)
{
    transform.Rotate(0, singleStep, 0);
}

// Move em direção ao objetivo
direction = Quaternion.Euler(transform.eulerAngles) * Vector3.forward;
rigidbody.velocity = Mathf.Lerp(prevSpeed, fishArea.speed, Mathf.Clamp(timeSinceAnim / 3, 0f, 1f)) * direction;
```

Fonte: elaborado pelo autor.

Além disso foram adicionados os métodos específicos do ML-Agents `CollectObservations` e `OnActionReceived` (Quadro 5). O método `CollectObservations` é responsável por informar ao ML-Agents valores relevantes que impactarão na agente na tomada de decisão. No caso do peixe ele observa posição atual, a saúde, a energia restante, a distância da comida, o gênero e a velocidade de execução do aquário. O método `OnActionReceived` é responsável por fazer as decisões dos agentes baseado em 2 vetores, o `DiscreteActions` e o `ContinuousActions`.

O vetor `DiscreteActions` é responsável por informar números inteiros. No projeto os valores podem vir zerados ou um, sendo que zerado representa não realizar a ação e um representa realizar a ação. A primeira posição do vetor controla a ação de alimentação, a segunda permite a reprodução do agente e a terceira faz o agente ficar parado e recuperar energia. O vetor `ContinuousActions` contém números racionais que variam entre zero e um. No trabalho ele foi utilizado para definir a posição do aquário que o peixe deve nadar.

Quadro 5 – Métodos `CollectObservations` e `OnActionReceived` no script `Fish`

```
public override void CollectObservations(VectorSensor sensor)
{
    totalReward = GetCumulativeReward();
    sensor.AddObservation(transform.forward);
    sensor.AddObservation((int)life);
    sensor.AddObservation((int)energy);
    sensor.AddObservation((int)distanceFromFood);
    sensor.AddObservation((int)gender);
    sensor.AddObservation((int)AquariumProperties.CurrentTimeSpeed);
}

15 referências
public override void OnActionReceived(ActionBuffers actions)
{
    if (actions.DiscreteActions[0] == 1)
    {
        Feed();
    }

    if (actions.DiscreteActions[1] == 1 && fishReproduction != null && startScale.x == transform.localScale.x)
    {
        if (fishReproduction.Reproduce())
        {
            AddReward(1f);
            if (gameController.Simulador) GameObject.FindObjectOfType<DebugCanvas>().AddReproduce();
        }
    }

    if (State == FStates.Nothing && actions.DiscreteActions[2] == 1)
    {
        State = FStates.Stay;
        timeStayed = 0;
    }

    if (State == FStates.Nothing)
    {
        State = FStates.Patrol;
        float[] targetPosition = actions.ContinuousActions.Array;
        target = new Vector3(targetPosition[0] * fishArea.bounds.size.x / 2 * 0.9f,
                             targetPosition[1] * fishArea.bounds.size.y / 2 * 0.9f,
                             targetPosition[2] * fishArea.bounds.size.z / 2 * 0.9f) + fishArea.transform.position;

        if (Vector3.Magnitude(target - rigidbody.position) < 3)
        {
            AddReward(-0.1f);
        }
    }
}
```

Fonte: elaborado pelo autor.

Para realizar o treinamento é necessário inicializar um servidor Python que contém a parte de aprendizagem profunda. Depois disso, quando for executada uma cena no Unity, ele irá se conectar com o servidor Python, onde o Unity será responsável por toda parte gráfica e de enviar as observações e recompensas coletadas para o servidor. O servidor será responsável por analisar os dados enviados e responder com as ações a serem tomadas para o Unity. Quando o treinamento é finalizado o Python gera um arquivo com o resultado do modelo treinado, que é usado para definir as ações do agente. Esse modelo pode ser vinculado ao agente no componente `Behavior Parameters`.

O treinamento dos agentes se dá pela execução de vários episódios, cada episódio simulando uma situação diferente do ambiente. Nas propriedades do peixe é possível configurar quantas ações serão executadas até finalizar o episódio. A cada ação executada, o agente usa as recompensas adquiridas e perdidas para tomar melhores decisões nas ações seguintes. Também é possível encerrar o episódio manualmente, no caso do peixe, o episódio é encerrado quando a saúde dele chega a zero. No treinamento do aquário, foram colocados dez peixes no aquário que serviram como agentes para a geração do modelo dos peixes.

A Tabela 1 apresenta os resultados do treinamento, no início do treinamento os peixes ainda não têm experiência, com isso eles não comem e não se movem com eficiência, dessa forma a pontuação fica extremamente baixa. No episódio 82 eles já começam a se alimentar e a pontuação aumenta. No episódio 136 o número de vezes que o peixe se alimentou aumenta consideravelmente e a recompensa fica positiva, porém o peixe começa a se alimentar demais, passando o máximo da saúde e perdendo recompensas por causa disso. No último episódio o peixe já se movimenta com eficiência, se alimenta adequadamente e é capaz de reproduzir.

Tabela 1 – Resultados do treinamento

Episódio	Recompensa	Número de vezes que os peixes comeram	Número de vezes que os peixes reproduziram
2	-1550,47	0	4
82	-48,875	16	8
136	4,26	2238	25
2234	23,83	150792	422

Fonte: elaborado pelo autor.

Os principais métodos durante a reprodução são o `layEgg` e o `fertilize`. O primeiro é chamado pelas fêmeas e tem como objetivo instanciar o ovo na cena e vincular ele a espécie. Com isso, os ovos caem ao fundo do aquário e podem ser fertilizados por peixes machos de mesma espécie através da segunda rotina. Quando as fêmeas ovulam ou os machos fertilizam os ovos, eles recebem recompensas durante o treinamento.

Quadro 6 – Métodos de reprodução da classe `EggFish`

```
1 referência
public static bool layEgg(Fish fish)
{
    if (fish.gender == Gender.female)
    {
        EggFish egg = Instantiate(fish.fishArea.eggFish, fish.transform.position, fish.transform.rotation, fish.fishArea.transform);
        egg.mother = fish;
        egg.prefab = fish.prefab.GetComponent<Fish>();
        egg.fishArea = fish.fishArea;
        return true;
    }
    return false;
}

1 referência
public bool fertilize(Fish fish)
{
    if (state == EggState.Layed && fish.gender == Gender.male && fish.specie == prefab.specie)
    {
        state = EggState.Fertilized;
        father = fish;
        return true;
    }
    return false;
}
```

Fonte: elaborado pelo autor.

O método `Update` (Quadro 7) é chamado a cada frame da execução do simulador e é responsável por controlar os tempos de amadurecimento, nascimento e morte do peixe dentro do ovo. Caso o ovo não seja fecundado em determinado tempo, o peixe morre e o ovo desaparece do aquário. Depois de fertilizado, o ovo passa por um período de amadurecimento até nascer. No nascimento o peixe é criado e o ovo é removido do aquário.

Quadro 7 – Método `Update` da classe `EggFish`

```
@ Mensagem do Unity | 0 referências
void Update()
{
    if (state == EggState.Fertilized)
    {
        timeSinceFertilized += Time.deltaTime / AquariumProperties.timeSpeedMultiplier;

        if (timeSinceFertilized > 100)
        {
            if (!GameObject.FindObjectOfType<GameController>().Simulador)
            {
                GameObject newFishObj = fishArea.SpawnFish(prefab.gameObject, Gender.random);
                Fish newFish = newFishObj.GetComponent<Fish>();
                newFish.Initialize(fishArea, true);
                fishArea.fishes.Add(newFish);
            }

            Destroy(gameObject, 1);
            state = EggState.Born;
        }
    }
    else if (state == EggState.Layed)
    {
        timeSinceLayed += Time.deltaTime / AquariumProperties.timeSpeedMultiplier;
        if (timeSinceLayed > 100)
        {
            Destroy(gameObject, 1);
        }
    }
}
```

Fonte: elaborado pelo autor.

A Figura 10 demonstra a execução do aquário virtual com as rotinas de reprodução. Os ovos ficam localizados no fundo do aquário. O peixe recém-nascido, localizado no centro do aquário, está com seu tamanho reduzido, mas seu comportamento e alimentação segue conforme os outros peixes.

Figura 10 – Peixe recém-nascido e ovo no fundo do aquário

Fonte: elaborado pelo autor.

4 RESULTADOS

Após realizado o treinamento, o modelo de comportamento gerado pelo treinamento foi inserido nos peixes do aquário. Constatou-se que em todas as velocidades do aquário o peixe foi capaz de se alimentar de forma adequada, evitando passar da saúde máxima. Além de conseguir reproduzir adequadamente para geração de novos peixes de mesma espécie. Porém a movimentação do peixe perdeu um pouco da naturalidade por estar utilizando o ML-Agents, os peixes

passaram a se repetir uma sequência de movimentação que definiram como ideal, diferentemente do que acontece em um aquário normal. Foram realizados testes das rotinas já existentes para validar as implementações de Losada (2019) e Silva (2020) e elas continuaram funcionando.

No início da adição da biblioteca ML-Agents ao simulador, encontrou-se dificuldades relacionas as rotinas de movimentação dos peixes, pois elas não respeitavam as colisões geradas pelo motor gráfico Unity. Dessa forma, se fez necessário alterar essas rotinas para que, quando o ML-Agents definisse a localização que o peixe deseja se mover, ele fosse capaz de se mover sem colidir com outros agentes.

Outro problema foi encontrado ao vincular as novas rotinas ao multiplayer. A a biblioteca utilizada por Silva (2020), `Mirror`, permite apenas a adição de um modelo de fabricação como jogador. Porém, cada espécie de peixe está separada em um modelo de fabricação, por isso só é possível definir uma espécie a ser sincronizado como jogador na biblioteca. A parte de reprodução também não está funcionando para o multiplayer, Silva (2020) desenvolveu a rotina de multiplayer de forma que apenas sincronizasse a posição, animação e rotação dos peixes. Com isso, todos os outros elementos do aquário são rodados individualmente em cada aplicativo, incluindo a parte reprodutiva dos agentes.

O trabalho se mostrou relevante em relação aos trabalhos correlatos, pois neles os agentes não são capazes de se reproduzirem com o uso de animação comportamental. Além disso, todas as rotinas dos peixes são gerenciadas pelo ML-Agents em um ambiente de movimentação em três dimensões com agentes operando simultaneamente, trazendo maior complexidade para o modelo.

5 CONCLUSÕES

Pode-se notar que o objetivo de adicionar um ciclo reprodutivo ao aquário virtual foi cumprido, permitindo que os peixes gerem descendentes, aumentando a dinamicidade do aquário. Com o uso da biblioteca ML-Agents foi possível aplicar o conceito de animação comportamental, onde o peixe foi capaz de aprender a se movimentar através do treinamento realizado previamente. A adição das propriedades de temperatura máxima e mínima das espécies permitiu que cada espécie seja diferenciada.

A biblioteca Unity ML Agents, permitiu o desenvolvimento dos agentes com maior facilidade. Porém também é capaz de gerar vícios nos agentes. No aquário, esse vício pode ser notado através da movimentação do peixe, que tende a buscar padrões, gerando poucas variações nas localizações selecionadas pelo agente durante a movimentação, almejando o máximo de eficiência.

Algumas das possíveis extensões relacionadas diretamente com o projeto são: explorar os vários algoritmos de aprendizado do ML Agents (PPO, SAC, MA-POCA e self-play) e fazer uma análise comparativa entre eles; aumentar a interação entre os peixes, talvez com peixes se alimentando de outros peixes; possibilitar a adição e remoção de peixes do aquário; estudar as espécies e trazer mais informações sobre elas, exibindo-as ao selecionar o peixe; Criar animações para alimentação e o envelhecimento do peixe; Limitar a reprodução dos peixes para ambientes ideais; Desenvolver o processo larval do peixe. Também é possível elencar algumas extensões relativas a versões anteriormente desenvolvidas: sincronizar todo o aquário no modo multiplayer, incluindo alimentação e reprodução; melhorar a movimentação do peixe.

REFERÊNCIAS

BENEDITO, E. **Biologia e ecologia de vertebrados**. Rio de Janeiro: Roca, 2015. E-book.

BONECKER, A. C. T. *et al.* **Catálogo dos estágios iniciais de desenvolvimento dos peixes da bacia de Campos.** Série zoologia: guias e manuais de identificação. Curitiba: Sociedade Brasileira de Zoologia, 2014. 295 p. ISBN 978-85-98203-10-2.

CASTRO, P.; HUBER, M. E. **Biologia marinha**. 8 ed. Porto Alegre: Amgh, 2012. E-book.

ESTEVÃO, J. M. **Análise do uso de animação comportamental com o motor de jogos Unity**. 2020. 17 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FELTRIN, G. R. **VISEDU-SIMULA 1.0:** Visualizador de material educacional, módulo de animação comportamental. 2014. 91 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. 2014.

JULIANI, A. et al. **Unity: A General Platform for Intelligent Agents**. San Francisco: Unity Technologies, 2018. 28 p. Disponível em: https://arxiv.org/pdf/1809.02627.pdf. Acesso em: 2 junho 2021.

LOSADA, F. O. **Aquário Virtual: Simulador De Ecossistema**. 2019. 19 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. 2019.

MACHADO, A. S. Uso de Softwares Educacionais, Objetos de Aprendizagem e Simulações no Ensino de Química. São Paulo: **Química Nova na Escola**, v. 38, n. 2, p. 104-111, mai. 2016.

THALMANN, N. M.; THALMANN, D. **Computer animation ´90**. Tokyo: Springer, 1990.

PISKE, K. E. **VISEDU – aquário virtual:** simulador de ecossistema utilizando animação comportamental. 2015. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. 2015.

REYNOLDS, C. **Behavioral animation**. [S.l.], [1997]. Disponível em: http://www.red3d.com/cwr/behave.html/. Acesso em: 2 dez. 2021.

RUSSEL, S.; NORVIG, P. **Inteligência artificial**. Rio de Janeiro: GEN LTC, 2013. 1 recurso online.

SILVA, M. W. da. **Aquário virtual: multiplayer e realidade virtual**. 2020. 14 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. 2020.

STEIN, R. T. **Ecologia geral**. Porto Alegre: Grupo A, 20/2018. E-book.

TOEBE, J. **Um Modelo Baseado em Agentes para o Ciclo de Vida de Afídeos: aplicação na interação afídeo-planta-vírus**. 2014. 167 f. Tese (Doutorado em Agronomia) - Universidade de Passo Fundo, Passo Fundo, RS, Brasil. 2014.

YU, Q. **A Decision Network Framework for the Behavioral Animation of Virtual Humans**. 2007. 10 f. Tese (Doutorado em Filosofia) - University of Toronto, Toronto. 2007.