UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS DE QUIXADÁ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANTÔNIO SÁVIO NASCIMENTO CAVALCANTE

AVALIANDO SIMULAÇÕES FÍSICAS DE UM PERSONAGEM CONTROLADO POR REDE NEURAL NA UNITY

QUIXADÁ

2018

ANTÔNIO SÁVIO NASCIMENTO CAVALCANTE

AVALIANDO SIMULAÇÕES FÍSICAS DE UM PERSONAGEM CONTROLADO POR REDE NEURAL NA UNITY

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Orientador: Prof. Dr. Rubens Fernandes Nunes.

QUIXADÁ

2018

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

C364a Cavalcante, Antônio Sávio Nascimento.
Avaliando simulações físicas de um personagem controlado por rede neural na unity / Antônio Sávio Nascimento Cavalcante. – 2018.
45 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Quixadá, Curso de Ciência da Computação, Quixadá, 2018.
Orientação: Prof. Dr. Rubens Fernandes Nunes.

1. Animação por computador. 2. Computação gráfica. 3. Personagens de videogame. 4. Unity. I. Título.
CDD 004

ANTÔNIO SÁVIO NASCIMENTO CAVALCANTE

AVALIANDO SIMULAÇÕES FÍSICAS DE UM PERSONAGEM CONTROLADO POR REDE NEURAL NA UNITY

Monografia apresentada no curso de Ciência da Computação da Universidade Federal do Ceará, como requisito parcial à obtenção do título de bacharel em Ciência da Computação. Área de concentração: Computação.

Aprovada em: ___/___/_____.

BANCA EXAMINADORA

_____

Prof. Dr. Rubens Fernandes Nunes (Orientador)
Universidade Federal do Ceará (UFC)

_____

Prof. Dra. Paulyne Matthews Jucá
Universidade Federal do Ceará (UFC)

_____

Prof. Me. Victor Aguiar Evangelista de Farias
Universidade Federal do Ceará (UFC)

À Deus.

Aos meus familiares e amigos.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me guiado durante toda a minha vida e por ter estendido a mão para me ajudar sempre que eu precisei. À toda a minha família em especial a minha mãe, Maria Consuêla do Nascimento, que me deu suporte para superar este desafio que foi esta graduação. Ao meu pai, Antônio Costa Cavalcante, e a minha irmã, Sara do Nascimento Cavalcante, que sempre estiveram me apoiando em todas as fases da minha vida.

Aos meus amigos que estiveram comigo desde o ensino fundamental e médio, Beatriz, Jêsca, Segundo, Vinicius, Yuri, e Robert, que estiveram sempre me apoiando, estando presente, nas conversas, nos estudos e nos divertimentos. Aos meus amigos de graduação Pedro Henrique, Leonardo Batista, Robson, Thomás, João Pedro, Fransa, Olímpio, Enoque, Alex Sandro, Lucas, Juliana, Sérgio, Leonardo Almeida, Marcelo, João Henrique, Rodrigo, Rômulo, Décio, João Vitor, Nathália e Sting, muito obrigado por todos os momentos principalmente os divertidos. Agradeço a todos os meus meus amigos em geral que não pude colocar os nomes aqui, incluindo os “combatenses”/combatentes e trilheiros.

Ao Prof. Dr. Rubens Fernandes Nunes, por todo o comprometimento com este trabalho, pela conversas, incentivos, orientações e experiências passadas ao longo dos anos da nossa convivência. Aos professores Paulyne Matthews Jucá e Victor Aguiar Evangelista de Farias, que compuseram a banca deste trabalho, e puderam contribuir tão significativamente para a melhoria e engrandecimento deste trabalho. À todo o Campus da Universidade Federal do Ceará por ter me dado essa oportunidade, aos docentes, técnicos administrativos e terceirizados que contribuíram de alguma forma na minha vida acadêmica.

“O meu socorro vem do SENHOR, que fez o céu e a terra.”

(Salmos 121:2)

RESUMO

A animação de personagens é uma área que está diretamente relacionada na expressão de movimento dos personagens. Esta área recebe incentivo de indústrias de jogos e cinema para a criação de ferramentas que produzam animações realísticas. Utilizar física adiciona realismo em animações e o uso de física em personagens tem sido possível com o desenvolvimento dos motores físicos e com a necessidade do estudo do movimento natural para a construção de controladores melhores. Este trabalho propõe uma infraestrutura de componentes para auxiliar o processo de otimização de personagens fisicamente simulados controlados por rede neural. Os componentes são responsáveis por gerar novas redes neurais, avaliar simulações físicas controladas pelas redes neurais geradas e selecionar as melhores com o intuito de evoluir em direção a um comportamento desejado. Além do conjunto de componentes, dois métodos de controle que exploram o uso de redes neurais são apresentados. A adição de funções de penalização à função objetivo (*fitness*) também é mostrada como uma opção de melhorar o comportamento do espaço de busca. Por fim, as dificuldades de se obter controladores adequados são discutidas.

Palavras-chave: Animação, simulação física, controle de personagem, Unity.

ABSTRACT

Character animation is an area that is directly related to the movement expression of the characters. This area receives an incentive from the gaming and film industries to create tools that produce realistic animations. Using physics adds realism to animations and the use of physics in characters has been possible with the development of physical motors and with the need to study motion to build better controllers. This work proposes an infrastructure of components to aid the process of optimization of physically simulated characters controlled by neural network. The components are responsible for generating new neural networks, evaluating physical simulations controlled by the neural networks generated and selecting the best ones in order to evolve towards a desired behavior. In addition to the set of components, two control methods that explore the use of neural networks are presented. The addition of penalty functions to the fitness function is also shown as an option to improve search space behavior. Finally, the difficulties of obtaining suitable controllers are discussed.

Keywords: Animation, physical simulation, character control, Unity.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

PD	*Proportional-Derivative*
RSA	Rede Sensor-Atuador
SAN	*Sensor-Actuator Network*

LISTA DE SÍMBOLOS

τ Tal

θ Teta

θ' Teta linha

SUMÁRIO

1 INTRODUÇÃO

Os jogos estão cada vez mais populares na vida das pessoas, e sua portabilidade vem se agregando conforme as tecnologias vão surgindo e evoluindo. É comum se desenvolver jogos para qualquer sistema operacional e, diante da diversidade de sistemas, desenvolver jogos multiplataforma. Tais jogos são por exemplo, *Final Fantasy XV*[1], *Grand Theft Auto V*[2], *The Sims 4*[3] e *Minecraft*[4].

Com a ideia de se aproximar da realidade, alguns jogos utilizam simulação física para gerar os movimentos dos personagens e a interação com os cenários. A animação de personagens fisicamente simulados é uma área da Computação Gráfica que vem crescendo, não apenas no contexto de jogos. Quando um personagem é fisicamente simulado, o que se quer é que ele automaticamente se mova e reaja de maneira semelhante ao que faria se estivesse no mundo real. "Sem simulação física, muitas vezes a naturalidade da animação depende bastante do talento artístico dos animadores, além de exigir muitas horas de trabalho para se obter um resultado de boa qualidade" (CAVALCANTE e NUNES, 2017, p. 1).

Este trabalho é desenvolvido usando a plataforma *Unity*, que facilita o uso da aplicação em diferentes dispositivos. A plataforma também já fornece as ferramentas necessárias para se trabalhar com física, não sendo necessária a implementação desse módulo. Desta forma, os profissionais responsáveis pelas animações podem usar seus conhecimentos em programação para acessar os componentes físicos disponíveis. Ao usar simulação física, um componente fundamental na animação de personagens é o controlador, responsável por definir, em cada instante da simulação, as forças e torques (angulares) necessários para que o personagem consiga executar os movimentos desejados. Alguns controladores de mais baixo nível são também nativamente implementados na *Unity*. Tais controladores nos permitem, por exemplo, definir o controle de um personagem através de poses (ângulos) desejadas a serem alcançadas na simulação, em vez de trabalhar diretamente com informações de forças e torques (angulares), que seriam bem menos intuitivas.

Baseado em (PANNE, KIM e FIUME, 1994), o movimento de um personagem fisicamente simulado pode ser gerado através da definição de um conjunto de poses desejadas. Para personagens 2D, uma pose corresponde a um conjunto de ângulos, um para

[1] www.finalfantasyxv.com
[2] www.rockstargames.com/v
[3] www.ea.com/pt-br/games/the-sims
[4] www.minecraft.net

cada articulação. Já uma pose desejada corresponde a uma pose alvo a ser alcançada durante a simulação, embora não necessariamente tenha que ser atingida. Normalmente, uma mesma pose desejada é mantida como alvo por um certo período durante a simulação, suficiente para servir como guia do movimento. No decorrer da simulação, os ângulos desejados das articulações podem ser modificados, gerando uma nova pose desejada, e os controladores de baixo nível, que funcionam simulando o efeito de molas e amortecedores angulares, ficam responsáveis por tentar atingi-los, guiando a animação.

Entretanto, mesmo usando esses controladores de baixo nível, definir as poses desejadas adequadas em cada instante da simulação para gerar o movimento não é uma tarefa fácil. Neste contexto físico, além de tentar imitar um movimento específico, também é preciso se preocupar com a interação entre o personagem e o cenário, inclusive o chão, e com as suas possíveis reações físicas, que podem impedir que as poses escolhidas guiem o movimento como desejado. Por questão de simplicidade, os experimentos são delimitados com o uso de um modelo 2D. Um cachorro articulado 2D é fisicamente simulado com apenas duas patas, uma representando o par de patas dianteiras e outra representando o par de patas traseiras. Ele é modelado como uma estrutura de corpos rígidos (blocos) usando articulações do tipo dobradiça (*hinge joints*).

Este trabalho se propõe a construir um módulo de controle na plataforma Unity, responsável por auxiliar nessa tarefa de controle de mais alto nível para definir automaticamente essas poses desejadas adequadas. No caso, o controlador gerado deveria idealmente ser capaz de fazer com que o modelo 2D escolhido se locomova de maneira natural. Entretanto, o sucesso do controlador gerado automaticamente depende bastante da estrutura escolhida para representar o controlador.

Com o objetivo de se obter controladores mais robustos no processo de otimização (aprendizagem), deseja-se que o módulo de controle proposto permita o uso de redes neurais. Embora as características das redes neurais usadas sejam essenciais para um bom aprendizado, o principal foco do trabalho consiste no desenvolvimento de um conjunto de componentes que facilitem o processo de criação e otimização de redes neurais capazes de evoluir o comportamento do personagem. Além do conjunto de componentes, dois métodos de controle que exploram o uso de redes neurais são apresentados. Por fim, as dificuldades de se obter controladores adequados são discutidas.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo geral deste trabalho é construir um módulo de controle baseado em redes neurais na plataforma Unity, responsável por auxiliar o aprendizado de locomoção de um personagem articulado 2D. O módulo deve permitir gerar novas redes neurais, avaliar as simulações físicas controladas por essas redes e selecionar as melhores.

1.1.2 Objetivo Específicos

Com base no objetivo geral deste trabalho, são definidos os seguintes objetivos específicos:

- Modelar um personagem 2D com articulações do tipo dobradiça na Unity.
- Implementar a avaliação automática das simulações físicas controladas por redes neurais.
- Implementar dois métodos de controle que exploram o uso de redes neurais.
    - Método baseado em Redes Sensor-Atuador (RSA), proposto em (PANNE e FIUME, 1993).
    - Método baseado em poses pré-programadas, combinando características de dois trabalhos (PANNE e FIUME, 1993; LASZLO *et al*., 2000).
- Discutir as dificuldades de se obter controladores adequados nos dois métodos de controle implementados.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados os fundamentos deste trabalho, organizado de acordo com os seguintes tópicos. No primeiro tópico é apresentado o conceito de animação de personagens, com ênfase em personagens articulados fisicamente simulados. Em seguida, é apresentado o conceito de controladores e finalmente os conceitos de Aprendizado de Máquina e Redes Neurais.

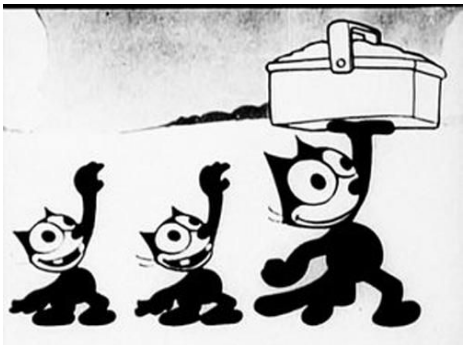
2.1 Animação de Personagens

O dicionário Aurélio define o termo "animação" como "dar animação a; dar vida a" (FERREIRA, 2004). "Animar é uma ação de gerar percepção de movimento (vida) no que está estático (inanimado). É uma questão de estar animado ou vivo" (ROUTT, 2007).

Em filmes, a animação é o processo no qual várias imagens (*frames*) são produzidas em sequência. Cada uma dessas imagens pode ser gerada individualmente com *stop motion* ou sintetizada usando computação gráfica. Em um filme produzido com *stop motion* (quadro-a-quadro), fotografa-se pequenas mudanças no que está sendo gravado para depois juntar as imagens e reproduzir o movimento, gerado ao transitar sequencialmente entre as imagens.

Luz (2009) afirma que, de um modo geral, o estilo de animação Norte-Americana, que tem como um dos principais representantes o estilo Disney, definiu por muito tempo os padrões de como as animações deveriam ser criadas. A Figura 1 ilustra um exemplo desse estilo. Mais recentemente, os jogos de computador também têm demonstrado bastante influência sobre esse mercado de animação.

Figura 1 – Imagem retirada de um episódio de Gato Félix



Fonte: (DE VOLTA AO RETRÔ, 2014)

A animação tradicional exige que o animador explore bastante suas habilidades artísticas, praticamente restringindo a tarefa para pessoas que possuem tais habilidades. O uso de ferramentas computacionais facilitou o processo de animação, tornando-o bem mais acessível para um maior número de pessoas. Um personagem, por exemplo, pode ser representado como uma estrutura articulada restringindo as definições de suas poses. Através da simples manipulação de um conjunto de valores, novas poses podem ser definidas rapidamente, sem a necessidade de ter que redesenhar todo o personagem manualmente.

A Figura 2 ilustra o uso de uma estrutura articulada (esqueleto ou rig) "por baixo" da representação da superfície usada para desenhar o personagem de fato, chamada de pele ou malha. Na medida em que esse conjunto hierárquico de ossos interconectados é animado, através da definição de uma sequência de poses, a malha vai se moldando ao esqueleto.

Figura 2 – Estrutura interna do personagem usada para guiar a malha

Fonte: (ALUMNI, 2009)

Mesmo com as facilidades fornecidas pelo uso do computador, definir as poses adequadas ainda exige um certo grau de habilidade artística por parte do usuário. Simular a física do personagem pode facilitar esse processo, pois o trabalho que o animador teria para gerar movimentos que obedecem as leis da física é automatizado, uma vez que as poses passam a ser geradas indiretamente através da ação de forças aplicadas ao personagem. Embora exista a desvantagem de perder o controle direto sobre as poses do personagem, a vantagem de se obter movimentos fisicamente corretos de maneira automática é um excelente atrativo para o uso da física.

No caso deste trabalho, a simulação física já é implementada pela própria plataforma de desenvolvimento escolhida. Na Unity, essa implementação é feita em uma camada de software escondida do programador. A Unity disponibiliza dois motores físicos, dependendo

do contexto em que se trabalha: Box2D[5], para ambientes bidimensionais e PhysX[6], para ambientes tridimensionais. O RigidBody é o componente utilizado para habilitar física nos objetos da cena.

2.2 Controladores

O fato de personagens serem simulados fisicamente não garante que eles apresentem um movimento natural. Ao iniciar a simulação, mesmo respeitando as leis da física, um personagem inicialmente em pé, em uma postura ereta, não consegue manter sua postura caso as devidas forças não sejam aplicadas internamente na sua estrutura. A falta de um controle adequado pode fazer com que o personagem simplesmente desabe no chão. Além de fazer com que o personagem execute uma trajetória desejada, um controlador ideal também deveria ser responsável por manter o equilíbrio do personagem de maneira estável e ter como critério um gasto de energia mínimo.

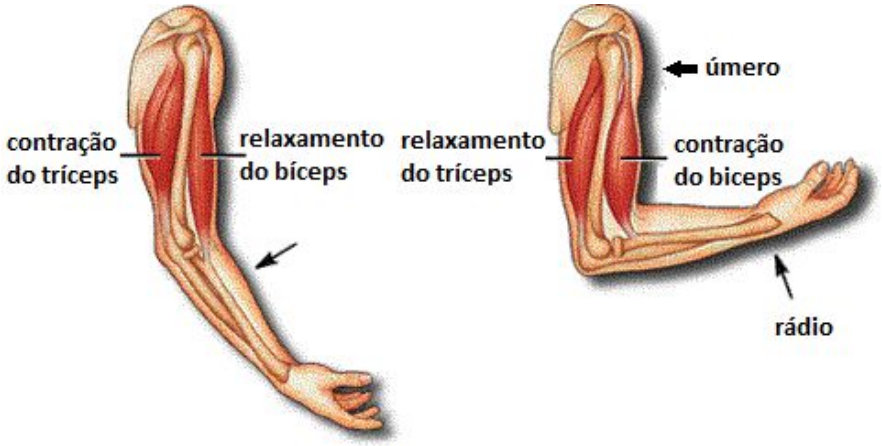
O problema de controle consiste então em determinar as forças e os torques adequados a serem produzidos pelos atuadores do sistema, a fim de realizar o movimento desejado, e os controladores são as entidades responsáveis por fornecer esses comandos para os atuadores, funcionando como o cérebro do personagem. Pode-se fazer uma analogia em que os controladores são como o cérebro humano, pois são responsáveis por enviar impulsos nervosos para que os músculos (atuadores) apliquem as forças necessárias para o personagem agir conforme o movimento desejado.

Nas articulações humanas, os atuadores funcionam como pares de músculos. A Figura 3 ilustra o funcionamento conjunto do bíceps e do tríceps, responsáveis por realizar o movimento do antebraço.

Os músculos bíceps e tríceps estão ligados ao osso rádio do antebraço e ao osso chamado úmero que está localizado no braço. Considerando a figura, no momento em que o bíceps contrai, puxando o osso rádio, e o músculo tríceps relaxa, o antebraço sobe. Ao relaxar o bíceps e contrair o tríceps, o antebraço baixa. A função do cérebro (controlador), portanto, é enviar sinais aos músculos indicando se devem contrair ou relaxar. Por questão de simplicidade, o efeito desses músculos agindo aos pares é representado através de um atuador angular.

[5] www.box2d.org
[6] www.geforce.com/hardware/technology/physx

Figura 3 - Visão anatômica de bíceps e tríceps relaxando e contraindo.



Fonte: Imagem adaptada de (BICEPS AND TRICEPS, 2014)

Os controladores de mais baixo nível, mencionados na introdução e disponíveis na *Unity*, consistem em controladores *Proportional-Derivative* (PD), que simulam o efeito de uma mola e de um amortecedor tentando alcançar um ângulo desejado θd para a articulação. Dado um ângulo desejado, o torque τ responsável por esse efeito em cada articulação é calculado de acordo com a equação seguinte:

$$\tau = ks * (\theta d - \theta) - kd * \theta',$$

onde *ks* é a constante de rigidez da mola, θ é o ângulo atual da articulação, *kd* é a constante de amortecimento e θ' é a velocidade angular atual da articulação. O controlador PD busca através desta fórmula realizar da melhor maneira a transição entre as poses desejadas, recebendo do programador apenas as constantes ks e kd das molas e os ângulos desejados.

Apesar do problema de controle que surge com o uso da simulação física e da dificuldade de se definir um controlador adequado, um personagem fisicamente simulado controlado adequadamente passa a poder interagir com o cenário de maneira automática. Um exemplo de tal interação é quando um personagem está em uma pose fixa e recebe uma bolada. Se a bola não for capaz de derrubá-lo, ele tentará retomar a pose que estava antes, reagindo naturalmente ao impacto sofrido.

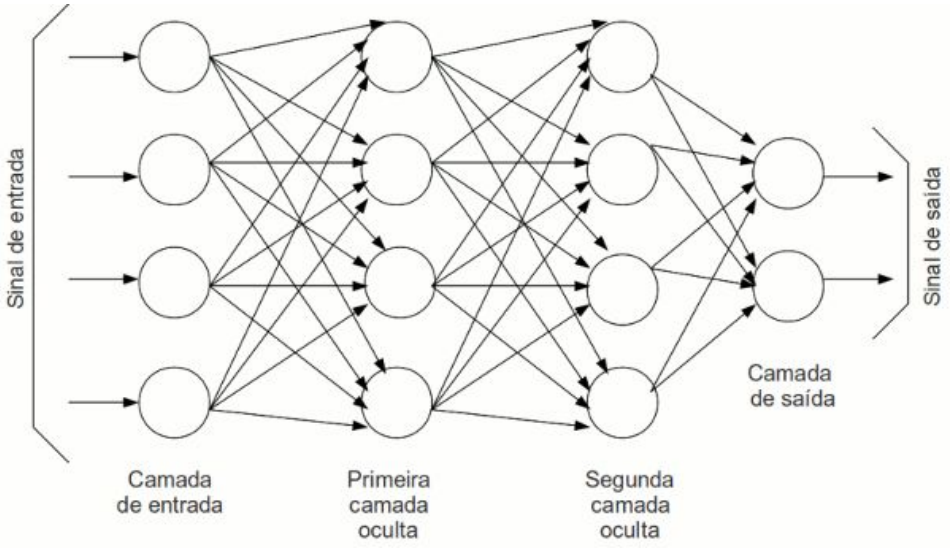
2.3 Aprendizado de máquina e Redes Neurais

Aprendizagem de máquina é uma área da ciência da computação que evoluiu do estudo de reconhecimento de padrões e da teoria do aprendizado computacional em inteligência artificial. Os algoritmos de aprendizado de máquina trabalham

construindo um modelo a partir de entradas amostrais a fim de fazer previsões ou decisões guiadas pelos dados ao invés de simplesmente seguindo inflexíveis e estáticas instruções programadas. O aprendizado de máquina explora o estudo e a construção de algoritmos que podem aprender de seus erros e fazer previsões sobre dados.

Uma forma possível de se trabalhar com aprendizado de máquina é utilizando redes neurais artificiais, inspiradas na estrutura e em aspectos funcionais das redes neurais biológicas, mais precisamente sobre o sistema nervoso central. As redes neurais usam reconhecimento de padrões e são sistemas de neurônios interconectados, que podem tomar decisões baseadas nos valores de entrada, como podemos ver na Figura 4.

Figura 4 – Representação de uma rede neural



Fonte: (MONOLITO NIMBUS, 2017)

As redes neurais são estruturas formadas por nós e arestas e são divididas por camadas. Na primeira camada, a rede recebe valores de entrada, chamadas de *features*. Em seguida, as camadas intermediárias escondidas realizam o processamento necessário da rede neural para gerar os valores de resposta, de decisão. A última camada é chamada de camada de saída. O número de nós e a estrutura geral da rede é normalmente escolhida previamente, e os valores nas arestas, chamados de pesos, são os responsáveis por definir seu comportamento individual. Ou seja, cada novo conjunto de pesos corresponde a um novo comportamento. Como uma rede neural possui muitos pesos, eles precisam ser modificados de maneira automática com o objetivo de ir melhorando seu funcionamento, otimizados segundo algum critério.

Os significados dos valores de entrada e saída da rede neural podem variar dependendo do problema abordado, sendo necessárias algumas adaptações de acordo com a aplicação desenvolvida. No contexto deste trabalho, as entradas da rede consistem de informações sensoriais do personagem que podem ser vistas na Figura 13 e as saídas da rede consistem de informações que serão traduzidas para uma pose desejada.

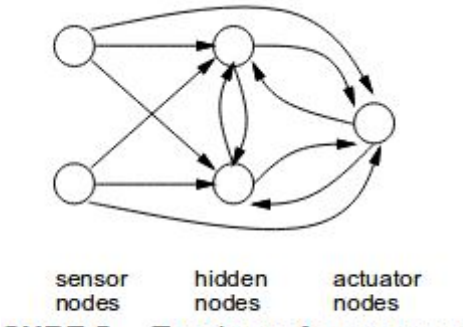
3 TRABALHOS RELACIONADOS

Nesta seção, apresentamos cinco trabalhos relacionados (PANNE e FIUME, 1993; PANNE, KIM e FIUME, 1994; Laszlo *et al*., 2000; MORDATCH, LOWREY *et al.*, 2015; NUNES *et al*., 2012).

3.1 Redes Sensor-Atuador

O artigo de (Van de Panne, 1993) tem como principal assunto as Redes Sensor-Atuador (RSA), em inglês *Sensor-Actuator Network (SAN)*, mostrando esta abordagem para animação de personagens fisicamente simulados. Para animar personagens com as RSAs, é necessário fornecer as configurações de um sistema mecânico, que são as informações estruturais dos corpos do personagem, como as articulações e os corpos rígidos a serem animados. Na Figura 5, é possível ver a estrutura de uma RSA abordada no artigo. Ela é uma rede neural simples e se restringe a 3 camadas de neurônios, permitindo também ligações diretas da primeira camada para a terceira.

Figura 5 – A RSA utilizada por (Van de Panne, 1993)

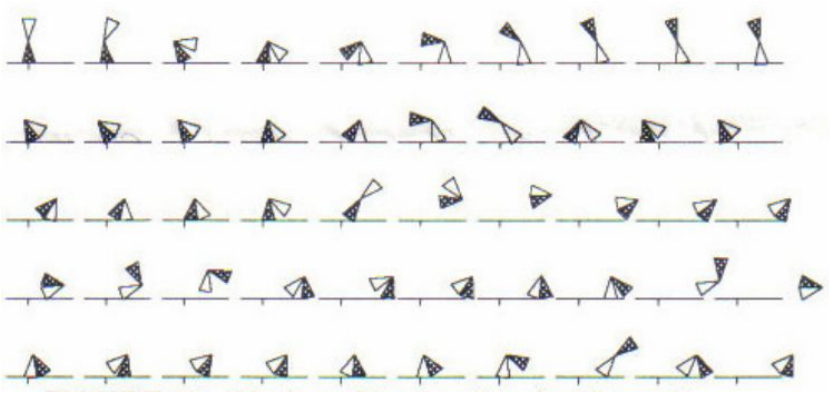


Fonte: (VAN DE PANNE, 1993)

Foram usados dez personagens como exemplos para mostrar a efetividade desta abordagem. Uma vantagem clara desta abordagem é poder descobrir automaticamente muitos modos de locomoção para o personagem, como pode ser visto na Figura 6.

Um dos objetivos do presente trabalho é implementar dois métodos de controle baseados na estrutura de uma RSA. Um dos métodos consiste em uma variação dessa rede baseada em poses pré-programadas. Mais detalhes podem ser vistos na Seção 4.2. A seleção das melhores redes neurais é baseada na distância percorrida em um curto período de tempo fixo, durante simulações sempre iniciadas a partir de uma mesma configuração (pose e

Figura 6 – Algumas formas de locomoção do Cart geradas pela RSA



Fonte: (VAN DE PANNE, 1993)

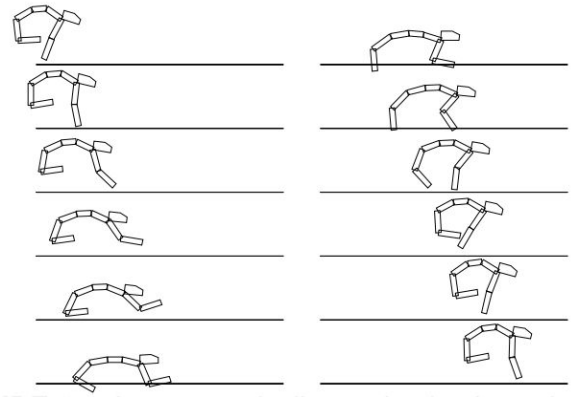
velocidade inicial) do personagem. Além dessa avaliação objetiva numérica, outra forma de avaliar seria pela elegância da animação gerada, que seria uma avaliação visual subjetiva do animador.

Um dos objetivos do presente trabalho é implementar dois métodos de controle baseados na estrutura de uma RSA. Um dos métodos consiste em uma variação dessa rede baseada em poses pré-programadas. Mais detalhes podem ser vistos na Seção 4.2. A seleção das melhores redes neurais é baseada na distância percorrida em um curto período de tempo fixo, durante simulações sempre iniciadas a partir de uma mesma configuração (pose e velocidade inicial) do personagem. Além dessa avaliação objetiva numérica, outra forma de avaliar seria pela elegância da animação gerada, que seria uma avaliação visual subjetiva do animador.

3.2 Wind-up Toys

Foi descrito em (PANNE, KIM e FIUME, 1994) o desafio de locomoção de personagens baseados em física e foi explicado sobre outras abordagens de trabalhos voltados a este problema. A locomoção de personagens por funções de condução periódicas é uma solução atrativa mostrada no trabalho. Uma máquina de estados (poses) é definida e a transição entre os estados proporciona uma transição entre poses feita pela própria simulação física. As várias poses intermediárias são geradas automaticamente, como pode ser visto na Figura 7.

Figura 7 – Síntese automática usando controle de poses para transição



Fonte: (PANNE, KIM e FIUME, 1994)

Panne e seus coautores (1994) propuseram uma estratégia de automatizar a síntese de controladores periódicos, onde os controladores definem as poses a serem transitadas. O artigo utiliza uma ideia de bonecos de corda, sendo assim os personagens não possuem noção do ambiente em que estão situados. A estabilidade do movimento do personagem foi alcançada ao transitar entre poses nos períodos de tempo corretos. A ideia do trabalho sobre locomoção por funções de condução periódicas contribuiu para a construção desta monografia na etapa de métodos de controle. A transição entre poses pré-programadas é utilizada em um dos métodos de controle abordado neste trabalho e se assemelha à abordagem de Panne e seus coautores (1994) quando se transita entre estados (poses).

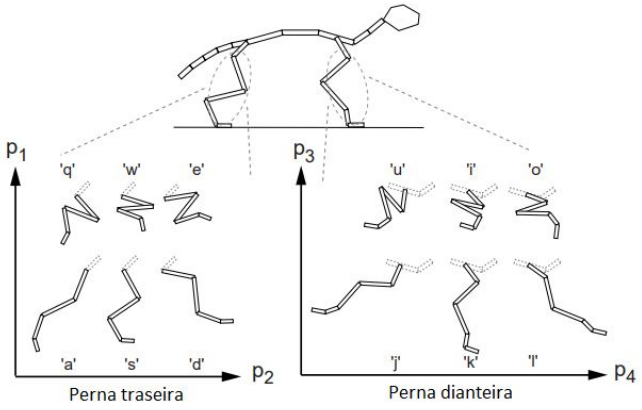
3.3 Interactive control for physically-based animation

Em (LASZLO *et al*., 2000) foi construído um grupo de personagens para propor técnicas interativas com repetição de movimentos para personagens animados fisicamente simulados. Foi possível movimentar os personagens com botões de teclado e mouse, facilitando para o animador e programador criar vários movimentos para alguns personagens. Tais personagens são uma lâmpada de mesa, um humanóide e um gato.

Na Figura 8, que pode ser vista mais à frente, encontra-se o gato modelado por Laszlo et al. (2000) em sua pose inicial. A mesma figura mostra algumas possíveis poses pré-programadas para o gato, cada uma associada com uma tecla diferente, que podem ser

escolhidas interativamente. Desta forma, o programador pode controlar o personagem articulado enquanto visualiza o efeito da interação em tempo real na tela do computador.

Figura 8 – Parametrização de movimentos do gato.



Fonte: adaptada pelo autor de (LASZLO et al., 2000)

Neste artigo o usuário que controla o personagem precisa ser treinado para enfim saber como realizar as trocas corretas entre as poses no tempo certo. Foram manipuladas apenas as pernas do gato em vez de todas as partes do corpo, reduzindo os graus de liberdade (ângulos das articulações) e facilitando o controle sem a necessidade de manipular outras partes. Essa escolha evita que a manipulação do personagem exija um maior número de botões do teclado.

O número de graus de liberdade envolvidos pode ser usado como uma métrica de complexidade do controle interativo de personagens. Por questão de simplicidade, nas duas abordagens de métodos de controle do presente trabalho, apenas os graus de liberdade das pernas do personagem são modificados para definir as poses desejadas.

3.4 Interactive Control of Diverse Complex Characters with Neural Networks

(MORDATCH, LOWREY, *et al.*, 2015) propuseram uma metodologia baseada em controle interativo de personagens para treinar redes neurais. O papel da rede neural foi ser o controlador de personagens, sejam eles bípedes, quadrúpedes, personagens para simular nado ou para simular voo. A rede neural alcançou a proposta do artigo em controlar interativamente e gerou comportamentos estáveis e realistas no conjunto de personagens propostos.

A rede neural possui um grande número de arestas (pesos) e permite o aprendizado de ações elaboradas do personagem. Algumas unidades da rede neural na implementação usam

estratégias recorrentes para implementar estados de memória e armazenam o estado atual do sistema para depois ser analisado.

As ações geradas pela rede neural para os personagens são definidas para analisar a rapidez que cada personagem individualmente realiza as suas tarefas de teste. Assim, a rede neural não está aprendendo como gerar um conjunto de movimentos específicos para o personagem, mas sim como manipulá-lo de forma implícita. Desta forma a rede neural evita se ajustar a uma situação de teste, que busca formas de locomover para vários tipos de personagens. A própria rede neural aborda uma estratégia que evita se engessar ao conjunto de dados de treinamento, assim os personagens sabem se sobressair nas diferentes situações propostas, não só nos problemas propostos nas etapas de teste.

Mordatch *et al*. (2015) usam aprendizagem supervisionada com otimização de trajetória para obter ganhos de *feedback* ótimos, além das ações ideais e injeção de ruído para causar mudanças inesperadas na especificação das tarefas. Como em Mordatch *et al.* (2015), este trabalho se propõe a utilizar uma rede neural para gerar poses adequadas ao controle de um personagem.

3.5 Using Natural Vibrations to Guide Control for Locomotion

No trabalho de (Nunes et al., 2012) foi proposto o uso de modos de vibração natural para auxiliar na criação de controladores para locomoção com um estilo específico. Foi criada uma base modal, que serviu para a reorganização dos graus individuais de liberdade do personagem (com o mesmo tamanho e complexidade). Com ela é gerado um conjunto de coordenações naturais que possibilitam as articulações oscilarem em frequências específicas.

A otimização de controladores geralmente envolve um espaço de busca bastante extenso, com muitas áreas apresentando variações mal comportadas da função objetivo e consequentemente apresentando muitos mínimos/máximos locais. Um ponto interessante do artigo foi explorar o uso de funções de penalização, responsáveis por tornar algumas regiões do espaço de busca mais bem comportadas e portanto tentar facilitar a tarefa da estratégia de otimização de encontrar regiões mais adequadas.

Baseado na ideia do artigo, o método de controle baseado em poses pré-programadas usa uma estratégia de penalização que limita o número de troca de poses de cada pata. A ideia é evitar que o processo de otimização se direcione para regiões do espaço de busca que

ocorrem excessivas trocas de pose, que faziam o personagem se locomover de maneira estranha, como se estivesse se tremendo. A função de penalização referente a cada pata é dada por:

$$Qu(d,e) = \begin{cases} -(e-d)^2 - h, \ se\ e > d \\ 0, \text{ caso contrário} \end{cases}$$

onde e é o número de vezes que ele trocou de pose, d é o limite de troca de poses permitido e $h>0$ é uma constante de penalização. Essa função de penalização é simplesmente somada à função objetivo original. Assim, nas regiões do espaço de busca em que o número de troca de poses for maior do que o limite, a função objetivo será prejudicada pelos termos negativos adicionados. Essas regiões portanto serão evitadas no processo de otimização.

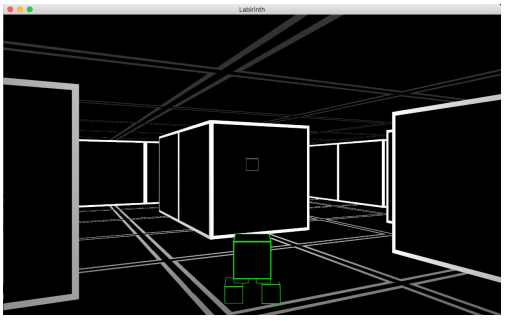
4 MÓDULO DE CONTROLE PROPOSTO

Este capítulo apresenta a modelagem usada para o personagem, o conjunto de componentes que formam o módulo de controle proposto, os métodos de controle implementados e a adição de penalizações.

4.1 Modelagem do personagem

Na concepção da modelagem de personagens, é recomendado inicialmente se ter noção do que se vai modelar. Se for desejado realizar modelagens com aspectos existentes no mundo real, como em aparência e estrutura física de uns personagens, é necessário mentalizar tais aspectos. A plataforma de desenvolvimento escolhida é a *Unity* e a decisão do modelo usado leva em consideração a sua simplicidade e estabilidade, recomendada para os testes iniciais do módulo de controle. O personagem escolhido consiste em um cachorro salsicha. A Figura 9 ilustra o desafio de criar personagens sem as ferramentas de modelagem adequadas, como as disponíveis na *Unity*. Construir um personagem usando a biblioteca gráfica OpenGL[7] diretamente, implementando sua estrutura em linhas de código, por exemplo, é bastante trabalhoso.

Figura 9 – Um personagem modelado em um labirinto de blocos feito com OpenGL



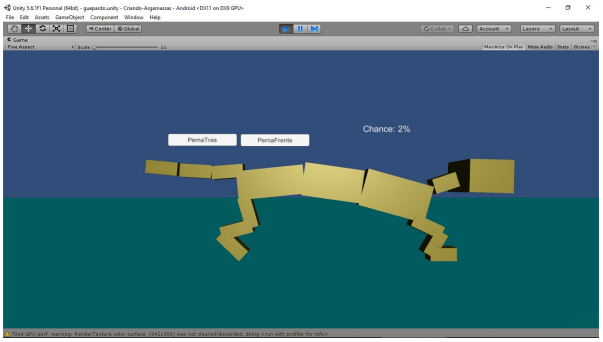
Fonte: (FONSECA, 2016)

Além de facilitar a modelagem da estrutura do personagem, a Unity também permite incorporar física na própria interface. O animador pode, por exemplo, especificar se um determinado corpo rígido é influenciado pela gravidade ou adicionar articulações de diferentes tipos. O personagem 2D usado nos testes consiste de uma estrutura de corpos rígidos (blocos) articulados, como mostrado na Figura 10. As articulações usadas são do tipo

[7] https://www.opengl.org/

dobradiça (*hinge joints*). Na Unity, as próprias articulações já podem ser habilitadas com propriedades de mola (*springs*) e amortecimento também na própria interface. Os controladores PD já estão implementados como parte integrante da articulação. O programador pode então especificar as constantes de rigidez e amortecimento da mola (ks e kd) e o ângulo desejado a ser alcançado em cada articulação, na interface ou por *script*.

Figura 10 – Um personagem modelado pelo autor na *Unity*



Fonte: Elaborada pelo autor

4.2 Componentes do módulo de controle

O módulo de controle fornece a infraestrutura necessária para auxiliar o processo de otimização dos controladores baseados em redes neurais. Essa infraestrutura é composta por componentes responsáveis por gerar novas redes neurais, avaliar as simulações físicas controladas por essas redes e selecionar as melhores. Os componentes são explicados a seguir.

4.2.1 Geração de novas redes neurais

A estrutura das redes neurais, consistindo do número de camadas e do número de nós por camada, é definida antecipadamente de acordo com o método de controle usado e mantida fixa durante todo o processo de otimização. Definida a estrutura, o processo de otimização portanto consiste em escolher os melhores pesos a serem usados na rede, os quais influenciam diretamente no comportamento dos atuadores (saída da rede) dos personagens de acordo com os seus sensores (entradas da rede). Cada novo conjunto de pesos corresponde a uma nova rede neural.

Há duas formas de gerar uma rede neural. A primeira é sorteando cada peso dentro do intervalo completo possível, -0.5 a 0.5. Isso corresponde a obter todos os pesos aleatoriamente

do zero. Ou seja, corresponde a obter um chute inicial qualquer no espaço de busca. A segunda é através de mutações, correspondendo a uma versão assexuada de algoritmos genéticos, para clonar uma determinada rede já existente e realizar pequenas mudanças a partir dos valores de pesos da rede existente. Isso corresponde a obter uma nova rede na vizinhança da clonada.

Cada nova rede neural gerada deve ser avaliada através da execução de uma simulação física. A ideia é que gerar uma nova rede na vizinhança de uma rede já bem avaliada tende a ter mais chances de melhoras, embora que pequenas (busca por um máximo local). Já gerar redes do zero tende a ter menos chance de melhoras, mas permite encontrar novas regiões promissoras (busca por um possível máximo global), dependendo da sorte de já cair em uma região boa do espaço de busca.

4.2.2 Avaliação das simulações físicas controladas por rede neural

A otimização das redes neurais consiste em encontrar o melhor conjunto de pesos, segundo algum critério escolhido (*fitness*). No contexto deste trabalho, para avaliar a qualidade dos pesos de uma rede neural é necessário executar a simulação física do personagem durante um determinado período de tempo, por exemplo dez segundos. Em cada instante da simulação, a rede recebe informações sensoriais (*features*) definidas sobre o personagem simulado e gera como saída uma pose desejada a ser seguida pelos controladores PD. Durante a execução individual de cada simulação física, o personagem é submetido a uma avaliação para calcular o seu *fitness*. O critério de otimização escolhido consiste em verificar, para cada período de tempo analisado, o quão longe o personagem consegue se locomover. Para isso, a cada avaliação, deve-se gerar um clone do personagem iniciando sempre com a mesma configuração inicial e analisar a distância percorrida sob o controle da rede sendo avaliada. Em (MORDATCH, LOWREY, *et al.*, 2015), a velocidade é associada à ação do personagem e utilizada como estratégia para melhorar a rede neural.

4.2.3 Seleção das melhores redes neurais

A otimização é definida através de uma sequência de populações. Cada população consiste de um conjunto de indivíduos, representados pelas novas redes geradas, que são individualmente avaliadas, exigindo uma simulação física para cada avaliação.

Ao avaliar todos os indivíduos de uma população, deve-se também usar um critério de corte, responsável por escolher quais indivíduos devem ser aceitos para serem usados na geração da próxima população e quais devem ser rejeitados. Ou seja, apenas os melhores são selecionados.

A cada nova população gerada, como mostrado na Figura 11, a escolha dos novos pesos para cada nova rede (indivíduo) pode ser feita: de maneira totalmente aleatória, útil apenas para obter chutes iniciais dos pesos; ou através de mutação dos indivíduos aceitos no critério de corte, permitindo modificações aleatórias apenas em intervalos próximos aos valores atuais dos pesos.

A primeira população é gerada completamente de maneira aleatória, do zero. A cada população avaliada, o algoritmo de otimização ordena as redes neurais de forma crescente de *fitness* e mostra-os ao programador. Depois de ordenadas as redes, uma alternativa é dividir o conjunto de redes em dois grupos de tamanhos iguais, separando as de melhores e as de piores *fitness*. Descartando as piores e mantendo as melhores, novas redes podem ser obtidas através de mutações para substituir as piores, mantendo o mesmo número de indivíduos por população.

Com o intuito de escapar de máximos locais, uma outra alternativa consiste em dividir cada população em 3 grupos: 2/5 das redes, as de melhores *fitness*, são mantidas; 2/5 das redes são substituídas por novas redes geradas através da mutação das redes mantidas; e 1/5 das redes são sorteadas do zero.

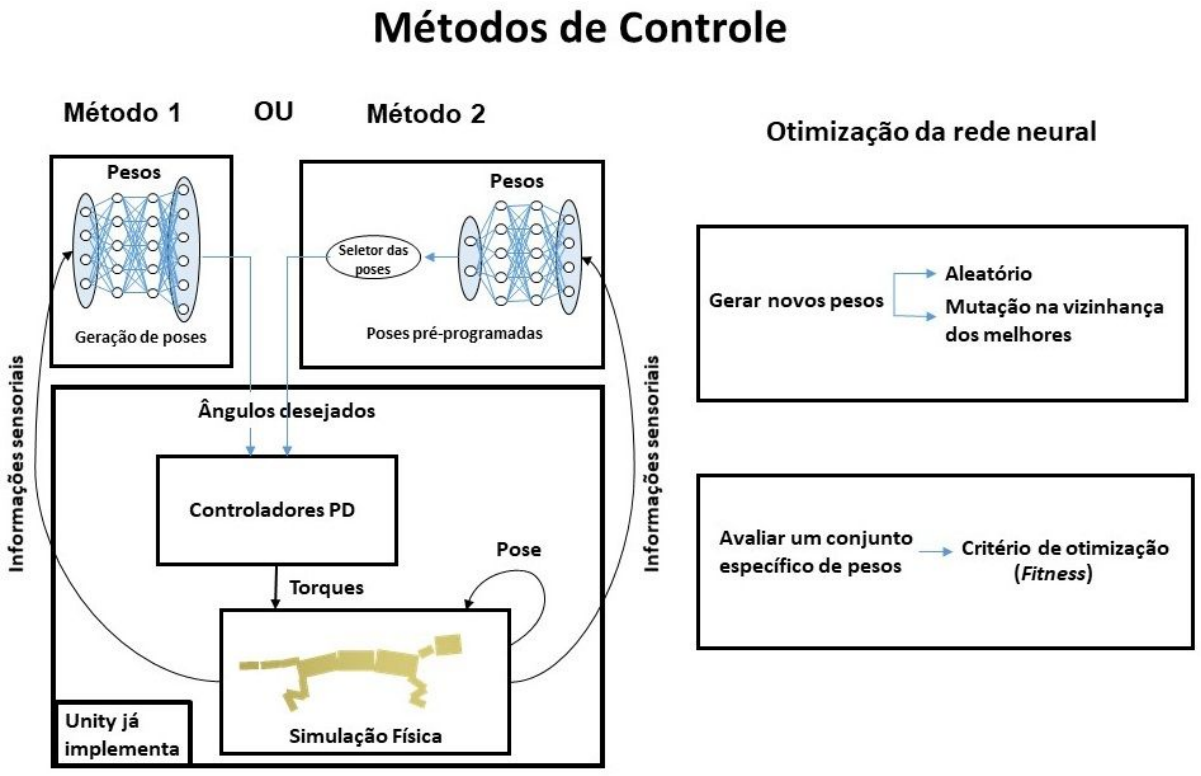
O algoritmo de rede neural se comporta melhor com o uso do algoritmo Backpropagation, que utiliza dados de treinamento com as saídas esperadas da rede, retrocedendo a camadas anteriores realizando cálculos e melhorando os pesos. Infelizmente, no contexto deste trabalho, não foi possível utilizá-lo por falta de informações de saídas esperadas e de dados de treinamento.

4.3 Métodos de controle

Os métodos de controle implementados neste trabalho são definidos em um nível de abstração acima dos controladores PD e são responsáveis por modificar os ângulos desejados quando necessário, possivelmente em cada instante da simulação. A Figura 11 mostra uma

visão geral do funcionamento do controle sobre o personagem fisicamente simulado e da comunicação entre os controladores e a simulação física.

Figura 11 - Esquema do uso de redes neurais como método de controle. Os dois métodos ilustrados abaixo são independentes e não funcionam em conjunto. Apenas um método é escolhido.



Fonte: Elaborada pelo autor

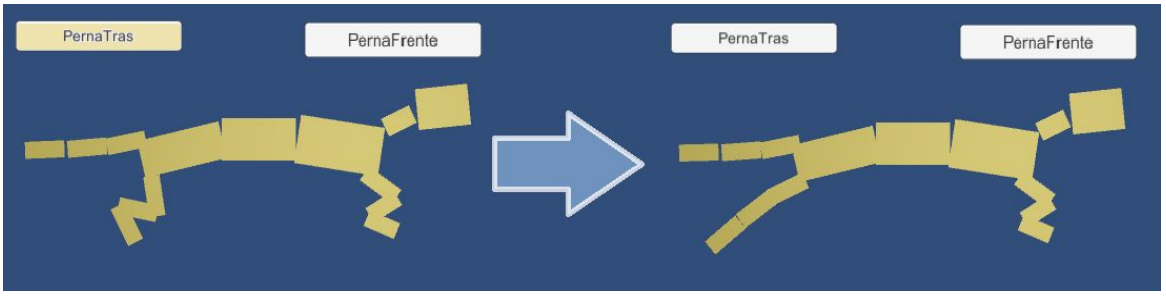
Foram implementados dois métodos independentes de controle de mais alto nível, baseados em redes neurais:

- **Geração de poses automáticas**: O primeiro método é baseado em uma Rede Sensor-Atuador (RSA) (VAN DE PANNE, 1993), que faz o papel do controlador de alto nível. Em cada instante da simulação, a rede recebe informações sensoriais (*features*) definidas sobre o personagem simulado e gera como saída uma pose desejada a ser seguida pelos controladores PD. A pose desejada é representada pelos nós de saída, com cada nó correspondendo ao ângulo desejado de cada articulação considerada, e continuamente modificada em cada instante, de acordo com os sensores obtidos da simulação. A estrutura da rede neural definida para este método possui 4 camadas. A camada de entrada possui 4 nós de entrada para receber as informações sensoriais. As

duas camadas intermediárias possuem 5 nós cada e a camada de saída possui 6 nós, um para cada articulação das patas do personagem.

- **Seletor de poses pré-programadas**: O segundo método usa um nível adicional responsável por selecionar poses desejadas previamente modeladas. Portanto, a rede não possui a mesma liberdade de definir qualquer pose como no primeiro método. As possíveis poses se restringem a algumas opções já previamente definidas. Cada pose definida previamente corresponde a um conjunto de ângulos e cada ângulo é referente a uma articulação do personagem. Em cada instante da simulação, de acordo com as informações sensoriais, a rede decide qual pose desejada utilizar. Neste caso, a saída da rede possui menos nós. Para o personagem específico, são usados apenas dois nós de saída e cada nó influencia os ângulos desejados apenas em uma das pernas. O valor de saída de cada nó decide em que momento se deve alternar entre as poses das pernas. Essa decisão é ilustrada na Figura 12, correspondendo à ação de apertar botões. Ao apertar os botões, o personagem estica ou comprime cada perna correspondente. Uma função de *threshold*, que se resume em verificar se os valores de saída recebidos são menores que um certo valor, é utilizada para decidir se um "botão" deve ser pressionado. Caso os valores de saída sejam menores que o *threshold*, significa que o botão deve ser pressionado. *Thresholds* bem pequenos são utilizados para evitar que o personagem fique trocando de poses excessivas vezes. Os controladores PD, ao receberem esses sinais, calculam os torques necessários para o cachorro alcançar a pose desejada selecionada. A estrutura da rede neural definida para este método possui 4 camadas. A camada de entrada possui 4 nós de entrada para receber as informações sensoriais. As duas camadas intermediárias possuem 5 nós cada e a camada de saída possui 2 nós, cada um correspondendo a um dos botões da Figura 12.

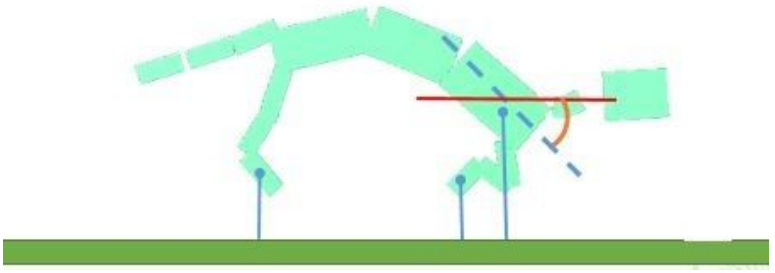
Figura 12 - Ação dos botões



Fonte: Elaborada pelo autor

Por questão de simplicidade, os ângulos desejados de algumas articulações foram mantidos fixos, como as costas, cabeça, pescoço e rabo, para diminuir o número de graus de liberdade. Portanto, as poses são definidas modificando apenas os ângulos das pernas. No primeiro método, o número de ângulos usados pode tornar o processo de otimização mais demorado. No segundo método, pode tornar a definição das poses mais trabalhosa. Para manter a consistência entre os métodos, apenas seis ângulos, três para cada perna ("botão" de cada nó de saída), são usados.

Figura 13 - Sensores do personagem



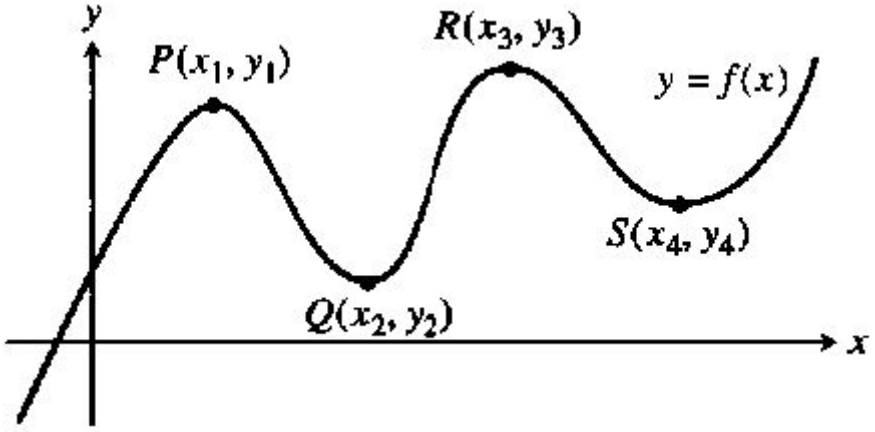
Fonte: Elaborada pelo autor

Note que os dois métodos possuem redes com as mesmas entradas e se diferenciam no número e significado das saídas. O primeiro método usa seis nós de saída, enquanto o segundo usa dois nós de saída. A Figura 13 indica os quatro sensores (informações sensoriais) a serem usados como entradas das redes: ângulo do tórax, altura do tórax, altura da pata da frente e altura da pata de trás.

Note que enquanto o primeiro método possui uma maior liberdade para gerar qualquer pose desejada, fornecida diretamente pelos nós de saída da rede, o segundo método, baseado em poses pré-programadas, apresenta a vantagem de trabalhar com um espaço de busca mais comportado na otimização, devido ao número reduzido de poses possíveis. O espaço de busca abrange todas as possibilidades de pesos que um método de controle pode explorar. Quanto mais simples for o personagem usado, mais comportado será o espaço de busca. Assim, o método de otimização, responsável por procurar a melhor solução de pesos nesse espaço de busca, terá mais chance de sucesso.

Um espaço de busca mal comportado se caracteriza por apresentar muitos mínimos/máximos locais, que no contexto apresentado pode corresponder a conjuntos de pesos que fazem o personagem cair e não melhorar a distância percorrida. A escolha de usar inicialmente um personagem mais estável evita um espaço de busca mal comportado. Um exemplo de mínimo local pode ser visualizado no ponto S(X4, Y4) da Figura 14, e o ponto Q(X2, Y2) é uma solução de valor mínimo melhor que o ponto S.

Figura 14 - Exemplo de função com mínimos locais.



Fonte: (Máximos e Mínimos Locais, 2016)

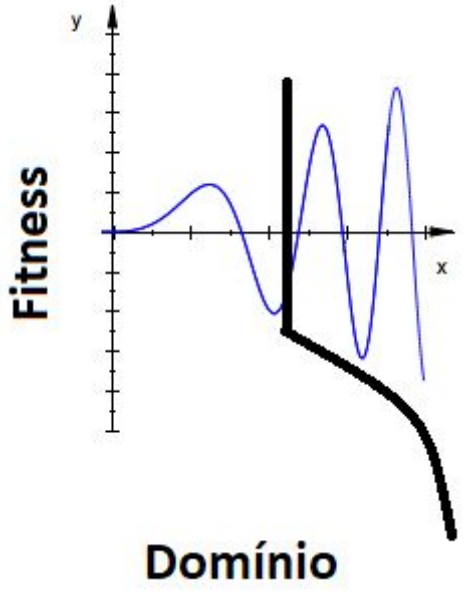
Comparando os dois métodos de controle, a expectativa inicial é que a geração automática de poses (RSA), primeiro método, permita que poses bastante imprevisíveis e indesejadas possam ser geradas, até que a otimização convirja para uma solução adequada. Já o segundo método, baseado em poses pré-programadas, embora não tenha a mesma liberdade do primeiro, deve convergir mais rapidamente a uma solução adequada, embora provavelmente não tão boa como a do primeiro método, pois estará restrita à habilidade fornecida pelas poses pré-programadas. As poses pré-programadas podem ser vistas como um bom chute inicial no espaço de busca, dado pelo programador.

4.4 Adição de penalizações

Uma possível estratégia para evitar máximos locais e encontrar soluções melhores é adicionar funções de penalização à função objetivo. Foram testadas duas estratégias de penalização aplicadas em forma de filtro.

A primeira ocorre quando o personagem faz um movimento que bate com a cabeça no chão e deste modo a rede é punida a ser resorteada, pois ela produziu um comportamento não desejado. A segunda penalização foi pensada para gerar um *fitness* ruim ou muito ruim, caso ela seja aplicada. Na Figura 15 pode-se ver como se comporta a função do domínio de redes neurais quando se adiciona a função de penalização de acordo com a abordagem de poses pré-programadas. A aplicação desta punição segue a mesma ideia aplicada por (NUNES et al., 2012), onde ela só é aplicada quando a rede neural troca de poses mais vezes que um limite definido. O cálculo de penalização é explicado na Seção 3.5.

Figura 15 - Função com penalização do limite de troca de poses.



Fonte: Imagem elaborada pelo autor

5 DESAFIOS ENCONTRADOS E SOLUÇÕES PROPOSTAS

5.1 Dificuldades com o método geração de poses automáticas

Ao realizar os experimentos com o método de controle RSA, foi percebido que os valores da saída eram muito próximos a zero. Como as saídas da rede neural eram muito pequenas, eles não eram capazes de produzir poses visualmente diferentes, resultando em personagens quase estáticos, se movendo muito pouco. Uma possibilidade é que isso se deu porque funções não lineares aplicadas no algoritmo de redes neurais modificaram os intervalos das saídas para valores minúsculos. Outra possibilidade é que os intervalos de valores usados nos pesos e nos nós da camada de entrada da rede neural não estejam adequados (atualmente o intervalo de -0,5 até 0,5 está sendo usado) e estejam causando este problema.

O artigo de (Van de Panne, 1993) também se deparou com o problema de que o personagem apresentava um movimento inicial e entrava em estado de repouso, embora por algum motivo diferente do nosso caso. A solução deste problema proposta por ele foi discretizar os valores dos nós da RSA para 0 ou 1 na camada de entrada e na escondida. Por fim somente a partir do último nó que é definido o ângulo desejado. Para a discretização foi utilizado uma estratégia usando tempos de espera para que as saídas não oscilem frequentemente, assim é preciso realizar somatórios com variáveis escondidas e verificar um *threshold* interno para sair de um estado, e ir para outro alcançando uma nova pose.

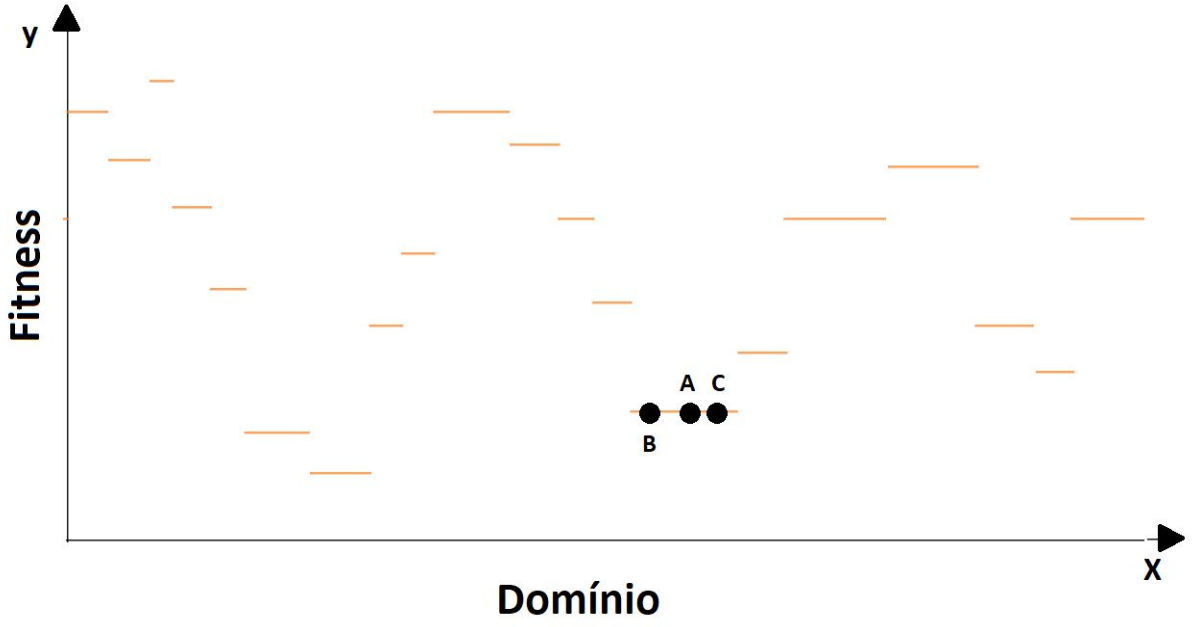
Ele definiu um período específico que se comporte de uma determinada forma para poder modificar as saídas e influenciar sobre o personagem. Quando os tempos de espera foram removidos o autor sofreu o mesmo problema, onde o personagem se mantém em repouso.

5.2 Dificuldades com o método seletor de poses pré-programadas

O método de poses pré-programadas usufrui de um threshold para gerenciar quando mudar ou não as poses do personagem, com isso o método se torna de natureza discreta, pois surge uma limitação nas possibilidades das saídas da rede influenciar no personagem. Mesmo que a rede neural sofra mutação, mas se não conseguir influenciar diretamente em suas saídas, então não apresenta mudança de *fitness*. Em outras palavras, enquanto algumas das suas saídas não forem capazes de passar pelo threshold, então o *fitness* continuará o mesmo se comparado com a execução do indivíduo anterior antes de sofrer mutação.

A figura 16 exemplifica um dos desafios de usar o método de controle poses pré-programadas. A figura mostra um plano cartesiano, onde nele se exemplifica para cada ponto da função uma rede neural no eixo x referente ao domínio de todos os pesos da rede e no eixo y referente ao fitness gerado por cada rede neural. É possível ver que o método por ser discreto produz uma função que gera fitness iguais na vizinhança mostrando linhas discretas como se fossem plataformas. Para exemplificar são destacados os pontos A, B e C. O ponto A corresponde a uma rede neural selecionada em uma determinada geração, por exemplo. Após sofrer mutação, o ponto B representaria a nova rede. Comparando os dois, vemos que são iguais no eixo y e a mutação não foi capaz de gerar melhoria. O ponto C representa um outro exemplo de uma nova rede neural gerada após outra mutação, e que também não consegue melhorar o *fitness*.

Figura 16 - Função do domínio do método de controle poses pré-programadas. O eixo X corresponde a uma representação abstrata de todo o possível domínio de redes neurais, definido pelo conjunto multidimensional de pesos.



Fonte: Imagem elaborada pelo autor

A otimização deste trabalho enfrenta o desafio de melhorar redes neurais e além disso tenta sair dos máximos locais. Um exemplo de máximo local pode ser visualizado no ponto P(X1, Y1) da Figura 14, e o ponto R(X3, Y3) é uma solução de valor máximo melhor que o ponto P. Com eles é muito difícil que somente mutações simples consigam superar este

obstáculo, pois as mutações deste trabalho modificam a rede ao ponto de produzir *fitness* que transitam entre a vizinhança. Fazendo uma análise visual da Figura 14, para que uma rede no ponto P alcance o ponto R, um grande declínio seria percorrido até que ocorra a ascensão até o ponto R. O desafio em questão é que quando o algoritmo perceber que a rede começasse a declinar, ele entenderia que ela está piorando em vez de melhorar e assim descartaria a mesma.

5.3 Desafios da avaliação

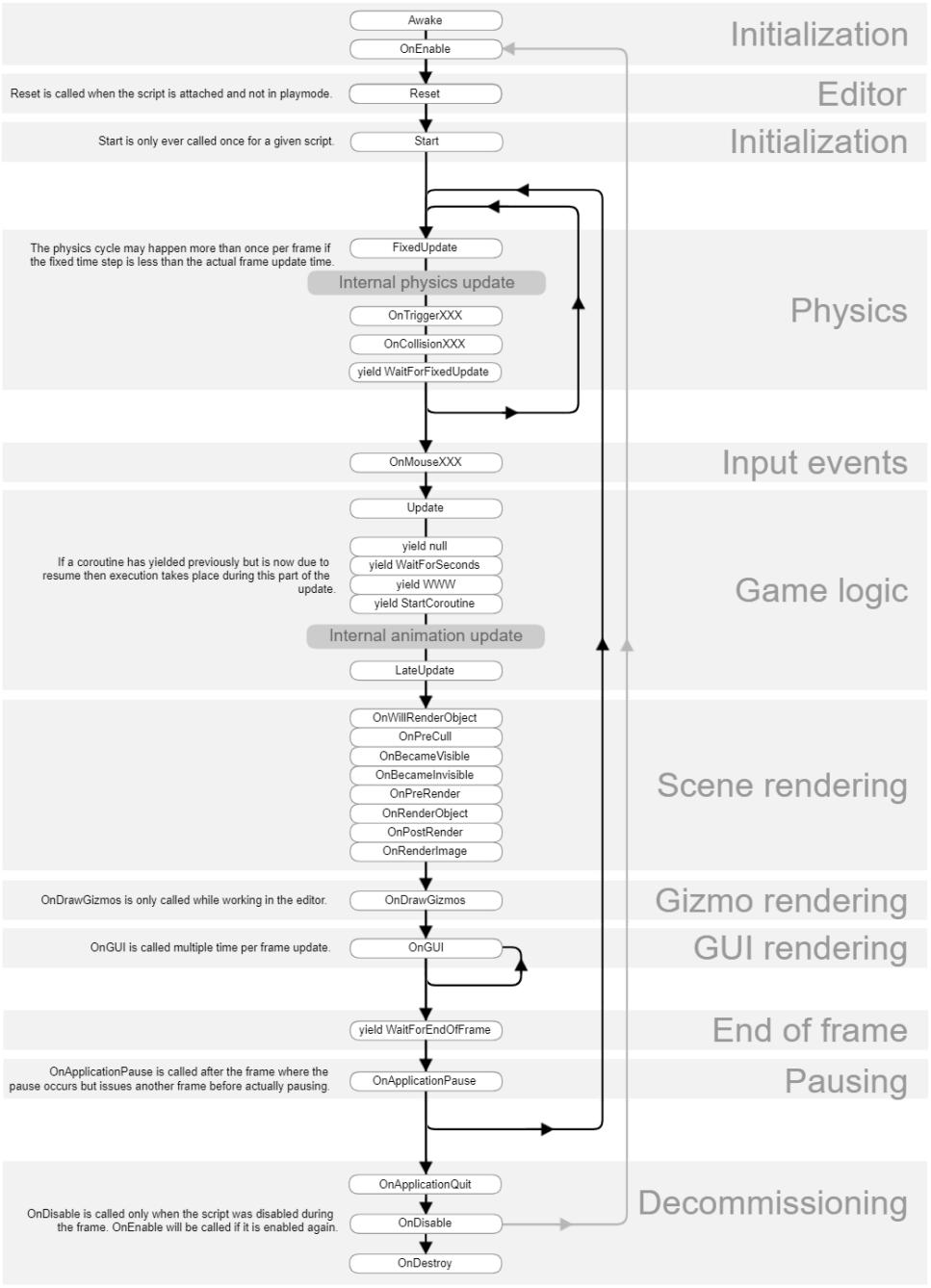
Ao longo do desenvolvimento deste trabalho nos deparamos com uma avaliação inconsistente a partir do momento em que percebemos que uma simulação física de um personagem influencia na simulação de outro personagem. Optou-se por simular individualmente cada personagem em vez de simular toda a população ao mesmo tempo. Ao término de uma simulação de personagem é necessário descartar do ambiente de simulação tudo da simulação anterior para poder prosseguir, exceto sua rede neural e seu fitness para ser comparado posteriormente na etapa de escolha da melhor rede.

No momento do descarte do personagem, o método OnDestroy da Unity não destruía o personagem a tempo de reconstruir o novo personagem da próxima avaliação, e um acabava influenciando na avaliação do outro. Esse atraso se deu pelo fato que o onDestroy é o último a ser executado na ordem de execução dos métodos da Unity (Figura 17). E como o código do controlador é chamado dentro dos métodos FixedUpdate ou Update, uma "destruição" dentro desses métodos não ocorre imediatamente, a destruição é apenas agendada.

Na implementação da infraestrutura surgiram alguns desafios, como imprecisões geradas pela física sobre o personagem. Foi detectado que a Unity não estava sendo precisa nos testes. Os resultados dos *fitness* após experimentos deveriam ser os mesmos, quando testados para a mesma rede neural simulando igualmente o mesmo personagem para instâncias diferentes. Contudo as simulações e os *fitness* não foram coerentes e alcançaram valores distintos em processos iguais. Isso se deu pelo não determinismo na simulação da Unity e também por pequenas colisões habilitadas pela própria plataforma. Migrar o projeto para a versão mais recente da Unity possibilitou usar uma variável da plataforma chamada *autoSimulation* e com ela foi possível desativar a física nos momentos de teste necessários.

O treinamento das redes neurais com visualização precisou ser realizado dentro do método FixedUpdate, pois ela é sempre seguida da execução de um passo da simulação física.

Figura 17 - Ilustração da ordem de execução das funções da Unity



Fonte: (UNITY, 2018)

Já o método Update está vinculado com o passo de renderização, e um número imprevisível de passos da simulação física pode ser executado a cada chamada do Update (Figura 16). Como a versão com visualização se torna mais lenta, uma alternativa sem visualização

também foi implementada, dentro do método Update. O desenvolvimento deste trabalho manteve essas duas versões, com e sem visualização, em paralelo.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi proposto e implementado um módulo de controle responsável por auxiliar o processo de otimização de redes neurais usadas no controle de personagens fisicamente simulados. Espera-se que os componentes implementados possam ser facilmente reutilizados para realizar novos experimentos com redes neurais de diferentes estruturas, com o objetivo de entender melhor como melhorar os controladores resultantes. Modelado um personagem e definida a estrutura de uma rede neural, tais componentes já devem estar prontos para serem reutilizados. Ao observar as etapas de teste, notou-se significativamente a evolução das redes neurais quando adicionamos as duas penalizações, pois elas ajudaram a descartar redes neurais que não estavam indo para o caminho desejado.

Um ponto positivo foi encontrado quando abordamos a estratégia de realizar mutação sobre apenas 2/5 das redes e resortear 1/5 dos piores. Desta forma reduziu a possibilidade de se prender aos máximos locais e permitiu que a infraestrutura continuasse explorando o espaço de busca sem se prender na vizinhança.

Uma sugestão alternativa para a mutação seria implementar algoritmos genéticos, que corresponderia a adicionar uma fase de *crossover*, em que pesos são trocados entre dois indivíduos aceitos simulando uma reprodução, ou seja, combinação e transmissão dos genes (pesos) dos pais para os filhos da próxima geração.

Leva-se como aprendizado que utilizar plataformas de desenvolvimento muitas vezes ajudam na evolução do trabalho, contudo podem ocorrer pontos de quebra na evolução do desenvolvimento. O ponto de quebra é um estado de interrupção na sequência de passos a serem executadas pelo autor e ocorreram várias vezes ao longo da implementação dessa infraestrutura. A quebra pode ser por falta de habilidades na plataforma, caso seja robusta e exija uma gama de conhecimentos prévios.

Como trabalhos futuros, pode-se destacar:

- Explorar melhor a estrutura e as características das redes neurais usadas;
- Implementar algoritmos genéticos, incluindo uma fase de *crossover*;
- Analisar se outros sensores poderiam ser passados a rede neural;
- Implementar uma penalização da velocidade das juntas, ver em (NUNES *et al.*, 2012);
- Modelar outros personagens e experimentar nesta infraestrutura.

REFERÊNCIAS

ALUMNI. **Skeletal Animation**. [S.l.:s.n.], 2009. Disponivel em: http://alumni.cs.ucr.edu/~sorianom/cs134_09win/lab5.htm. Acesso em: 03 mai. 2018.

BICEPS AND TRICEPS. **organ system**. [S.l.:s.n.], 2014. Disponivel em: https://muscularsystem12.weebly.com/organ-system.html. Acesso em: 20 jun. 2018.

CAVALCANTE, A. S. N.; NUNES, R. F. **Controle Interativo de Personagem Fisicamente Simulado usando Unity**. Quixadá: Universidade Federal do Ceará, 2017. p. 106-110.

DE VOLTA AO RETRÔ. **O Gato Felix**. [S.l.: s.n.], 2014. Disponivel em: https://www.devoltaaoretro.com.br/2014/04/o-gato-felix.html. Acesso em: 20 jun. 2018.

FERREIRA, A. B. D. H. **Novo dicionário Aurélio da língua portuguesa**. [S.l.: s.n.], 2004.

FONSECA, C. O. **caiubi / LabirinTron**. [S.l.:s.n.], 2016. Disponivel em: https://github.com/caiubi/LabirinTron. Acesso em: 17 jun. 2018.

LASZLO, J. A. V. D. P. M. A. F. E. Interactive control for physically-based animation. [S.l.]: **Proceedings of the 27th annual conference on Computer graphics and interactive techniques**, 2000. p. 201-208.

LEONARD-CANNON, B. **Physics-based Character Control with Deep Reinforcement Learning**. [S.l.]: McGill University Libraries, 2016.

LUZ, F. C. **Animação digital:** reflexos dos novos medias nos conceitos tradicionais de animação. [S.l.]: Imagem e Cultura Visual, 2009.

MÁXIMOS e Mínimos Locais. [S.l.:s.n.], 2016. Disponivel em: http://www.dmm.im.ufrj.br/projeto/projetoc/precalculo/sala/conteudo/capitulos/cap61s2.html. Acesso em: 19 jun. 2018.

MONOLITO NIMBUS. **Redes Neurais Artificiais**. [S.l.:s.n.]. Disponivel em: https://www.monolitonimbus.com.br/redes-neurais-artificiais/. Acesso em: 03 mai. 2018.

MORDATCH, I. et al. **Interactive control of diverse complex characters with neural networks**. [S.l.]: Advances in Neural Information Processing Systems, 2015. p. 3132-3140.

NUNES, Rubens F. et al. Using natural vibrations to guide control for locomotion. **Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games.** ACM, 2012. p. 87-94.

PANNE, M. V. D.; FIUME, E. Sensor-actuator networks. **Proceedings of the 20th annual conference on Computer graphics and interactive techniques**: ACM, 1993. p. 335-342.

PANNE, M. V. D.; KIM, R.; FIUME, E. **Virtual Wind-up Toys for Animation**. Banff, Alberta, Canada: Proceedings of Graphics Interface '94, 1994. p. 208-215.

ROUTT, W. **De anime**. The Illusion of Life II: More essays on Animation. Power Publications: Sidney, 2007. p. 335-342.

UNITY. **Execution Order of Event Functions**. [S.l.:s.n.], 2018. Disponivel em: https://docs.unity3d.com/Manual/ExecutionOrder.html. Acesso em: 3 dez. 2018.