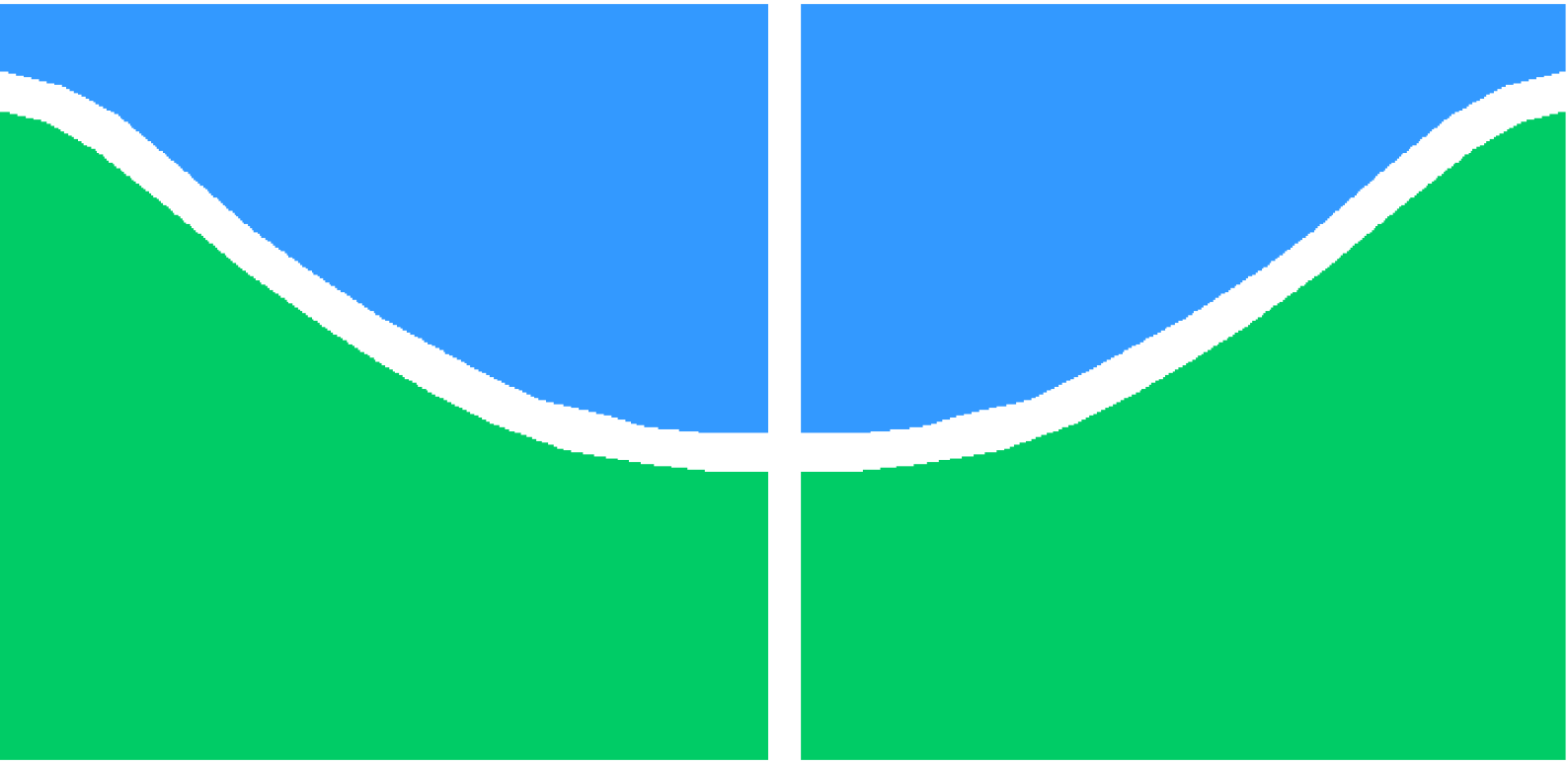


Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Aplicando a Neuroevolução para obter NPCs de Alta Qualidade

Autor: Luis Gustavo Souza Silva
Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Brasília, DF
2014

Luis Gustavo Souza Silva

Aplicando a Neuroevolução para obter NPCs de Alta Qualidade

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Coorientador: Prof. Dr. Edson Alves da Costa Júnior

Brasília, DF

2014

Luis Gustavo Souza Silva
Aplicando a Neuroevolução para obter NPCs de Alta Qualidade/ Luis Gustavo Souza Silva. – Brasília, DF, 2014-
95 p. : il. (algumas color.) ; 30 cm.

Orientador: Profa. Dra. Carla Silva Rocha Aguiar

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2014.

1. Jogos Eletrônicos. 2. Inteligência Artificial. 3. Inteligência Computacional. 4. Neuroevolução. I. Profa. Dra. Carla Silva Rocha Aguiar. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aplicando a Neuroevolução para obter NPCs de Alta Qualidade

CDU 02:141:005.6

Luis Gustavo Souza Silva

Aplicando a Neuroevolução para obter NPCs de Alta Qualidade

Monografia submetida ao curso de graduação em Engenharia de Softwareda Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 18 de Novembro de 2013:

Profa. Dra. Carla Silva Rocha Aguiar
Orientador

Prof. Dr. Edson Alves da Costa Júnior
Coorientador

Profa. Dra. Carla Denise Castanho
Convidado 1

Prof. Dr. Wander Cleber Maria Pereira da Silva
Convidado 2

Brasília, DF
2014

Este trabalho é dedicado aos meus pais,
Valdivino e Mônica, por iluminarem minha vida.

Agradecimentos

À minha amiga e orientadora Dra. Carla Silva Rocha Aguia e ao Prof. Dr. Edson Alves da Costa Júnior, por terem me ajudado com observações, esclarecimentos e ensinamentos.

Aos meus amigos Tales Martins, Adriano de Jesus Barboza, Igor Ramos e Maylon Felix, pelo companheirismo.

À Faculdade do Gama, que possibilitou meus estudos de graduação no curso de Engenharia de Software.

Resumo

Este documento descreve a proposta para a implementação de um modelo de inteligência artificial baseado na neuroevolução a fim de obter personagens controlados pelo computador que fossem de alta qualidade. Para isso, foram estudados os conceitos da inteligência artificial, a fim de entender as técnicas clássicas que são usadas em jogos, e da inteligência computacional, para focar nas técnicas que apresentem adaptabilidade. Além disso, foi feita uma análise de como algumas técnicas e algoritmos de inteligência artificial são usadas em jogos eletrônicos e como elas afetam o *gameplay*. Essa análise servirá de referencial para a produção de um protótipo e condução de uma *survey*. Adicionalmente, o documento traz detalhes sobre o algoritmo *Neuro-Evolving Augmenting Topologies* (NEAT) usado para implementar a neuroevolução. Por fim, o trabalho apresenta os resultados obtidos no desenvolvimento do protótipo desenvolvido assim como na condução da *survey* analisa os resultados obtidos.

Palavras-chave: jogos eletrônicos. inteligência artificial. inteligência computacional. redes neurais. algoritmos genéticos. neuroevolução.

Abstract

This document describes the proposal for implementation of a artificial intelligence model based on multimodal neuroevolution in order to achieve high quality non-player characters. For this it was studied the concepts of artificial intelligence, to understand the classical techniques that are used in games, and computational intelligence techniques, to focus on showing adaptability. Also an analysis of how some of the artificial intelligence's techniques and algorithms are used in eletronic games and how they affect the gameplay. This analysis will serve as a reference for the production of a prototype and conduct a survey. Additionally, the document provides details about the *Neuro-Evolving augmenting Topologies* (NEAT) algorithm used to implement the neuroevolution. Finally, the paper presents the results obtained in the development of the prototype as well as in the conduct of the survey and analyzes the results.

Key-words: eletronic games. artificial intelligence. computational intelligence. neural networks. genetic algorithm. neuroevolution.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

1.1 Contexto

Um dos grandes desafios no desenvolvimento de jogos cada vez mais imersivos é a melhoria da Inteligência Artificial (IA) usada nas diversas unidades controladas pelo computador.

De modo geral, todos os jogos incorporam alguma forma de IA. Podemos considerar IA qualquer ação/comportamento no jogo que cria a ilusão de um certo nível de inteligência, tornando o jogo mais imersivo, desafiador e divertido. Técnicas de IA podem ser usadas para diversas finalidades em jogos, tais como em algoritmos de *pathfinding*, planejamento, criação de conteúdo, personagens controlados por computador, dentre outros (FREITAS et al., 2012). O presente trabalho aborda o uso de técnicas e algoritmos de Inteligência artificial em personagens controlados por computador.

1.2 Problema e Relevância

A importância da experiência ao interagir com um jogo eletrônico está associada com a imersão e o desafio que o dado jogo proporciona (FREITAS et al., 2012). A imersão visual em ambientes de jogos já foi obtida graças aos avanços na visualização tridimensional alcançados tanto a nível de hardware quanto de software. O próximo passo para jogos ainda mais imersivos é melhorar a Inteligência Artificial (IA) usada nesses jogos. Graças ao aumento do poder computacional é possível criar engines de IA mais robustas (BOURG; SEEMANN, 2004).

No caso dos personagens controlados pelo computador, também chamados de *BOTs* ou *NPCs*, o objetivo é criar oponentes interessantes, de alta qualidade. A qualidade de um personagem controlado pelo computador, do ponto de vista de diversão do jogador, está relacionado com suas habilidades, sua imprevisibilidade e o desafio proporcionado. Tais características devem ser apresentadas de modo que sejam similares a de um jogador humano (SONI; HINGSTON, 2008). Dessa forma, BOTs que são capazes de aprender e otimizar podem apresentar essas características e, consequentemente, possuem maior qualidade.

As técnicas determinísticas tradicionais não são capazes de obter tais resultados pois não existe incerteza. Por isso, a neuroevolução vem sendo utilizada para lidar com esse problema e se mostrou uma técnica promissora para obter comportamentos imprevisíveis e de alta adequação.

Resumidamente, podemos definir o problema que esse trabalho busca enfrentar como:

Projetar e Implementar a Inteligência Artificial de personagens controlados por computador para protótipo do jogo "The Island".

Os personagens controlados pela IA do jogo deverão ter alta qualidade, tendo ações não determinísticas e adaptáveis às situações que não foram previstas no documento de *design*. De modo geral, antecipar como reagir contra um jogador humano é uma tarefa difícil. Por exemplo, é difícil antecipar os momentos em que o BOT deverá tentar esquivar dos disparos do jogador. Isso porque essa decisão envolve variáveis como os pontos de vida do BOT, se o jogador está atirando e se o BOT está sob a mira do jogador, por exemplo.

Para solucionar o problema proposto, o trabalho foca nas técnicas e algoritmos de inteligência artificial. Essas técnicas devem resultar em unidades com um comportamento inteligente similar ao de um humano, porém sem interesse em como tal comportamento foi obtido (SCHWAB, 2004). Para isso, o ramo da IA chamado Inteligência Computacional (IC) é o mais indicado, pois estuda e cria as técnicas e mecanismos capazes de aprender e se adaptar a ambientes complexos e em constante mudança.

Para resolver esse problema temos que lidar com outros tipos de problemas do domínio de jogos. Dentre esses, destacam-se três grandes preocupações. A primeira é a criação da partida do jogo, o que envolve a animações, controle dos personagens, tratamento de entradas e configurações do jogo. A segunda é lidar com a integração do motor gráfico com o pacote de inteligência artificial. Finalmente, a terceira trata-se de criar um experimento, também chamado de "*Play Test*" no contexto de jogos, para avaliar a perspectiva do usuário.

1.3 Objetivos Gerais

O objetivo desse projeto é desenvolver a IA, usando neuroevolução, para obter BOTs de alta qualidade para o protótipo do jogo "*The Island*".

1.4 Objetivos Espefíficos

A fim de alcançar o objetivo geral os seguintes objetivos específicos foram identificados:

- Integrar os motores;

- Implementar estrutura necessária para realizar a neuroevolução;
- Validar a IA do ponto de vista técnico e do usuário.

1.5 Organização do Trabalho

O restante do documento está organizado da seguinte maneira. No Capítulo 2, são apresentados os conceitos básicos para o entendimento de inteligência artificial, inteligência computacional e como estes dois conceitos são relacionados aos jogos. O Capítulo 3 trata de descrever o contexto e a inspiração do jogo. Além disso, são apresentados detalhes do ambiente de desenvolvimento assim como da arquitetura criada. Em seguida, o Capítulo 4 apresenta os dados coletados para validar a IA desenvolvida. Por fim, o Capítulo 5 reflete os resultados obtidos.

2 Fundamentação Teórica

O uso de Inteligência Artificial em jogos cria novas possibilidades de projeto (TOZOUR, 2002). Um jogo que evolua, aprenda e se adapte a medida que é jogado irá extender a vida útil do jogo. Do ponto de vista técnico, as IAs da maioria desses jogos são projetadas para atender três necessidades básicas: movimentação, tomada de decisão e estratégia (MILLINGTON; FUNGE, 2009), .

No escopo desse protótipo, a IA desenvolvida atenderá a necessidade de tomada de decisão. Como informado no Capítulo 1, essa tomada de decisão deve dar a sensação ao jogador que os BOTs são controlados por jogadores humanos. Ou seja, possuir elementos de desafio e imprevisibilidade que proporcionem alta qualidade.

O capítulo apresenta todos os conceitos básicos teóricos necessários para a compreensão e desenvolvimento do trabalho apresentado. Na Seção 2.1 são apresentadas a taxonomia de termos usados ao longo do trabalho. A Seção 2.2.1 é apresentada uma breve introdução acerca do campo de estudo Inteligência Artificial. As Seções 2.3 e 2.4 focam em conceitos e definições importantes para a compreensão de algoritmos de IA neuroevolutivos, que são aplicados posteriormente no projeto e implementação dos personagens controlados pelo computador do jogo "*The Island*".

2.1 Taxonomia

2.1.1 IA Simbólica

Um sistema que usa IA simbólica possui uma base de conhecimento, que é o módulo responsável por representar todo o conhecimento do sistema e um mecanismo de inferência. O mecanismo de inferência é o responsável por selecionar e aplicar o conhecimento existente. Um exemplo típico desses sistemas são os chamados Sistemas Especialistas.

2.1.2 IA naturalista

Um sistema de valorizavam as abordagens que têm alguma inspiração biológica ou natural. Isso aconteceu devido a basicamente duas insatisfações com os resultados obtidos usando sistemas simbólicos. O primeiro motivo é por que eles não escalavam bem com a dificuldade do problema. Isto é, funcionavam bem para problemas mais simples, porém, tinham desempenho insatisfatório para problemas mais complexos. O segundo motivo era que os sistemas simbólicos não eram biologicamente plausíveis, o que feria a visão filosófica.

2.1.3 IA híbrida

Um sistema de combina elementos de IA Simbólica e IA Naturalista a fim de equilibrar vantagens e desvantagens de diferentes técnicas.

2.1.4 Topologia da Rede Neural

A diz respeito a disposição de como os neurônios estão conectados.

2.1.5 *Crowding*

O fenômeno de *Crowding* , ou Aglomeração, acontece quando algum indivíduo da população é muito mais bem adaptado do que outros e acaba se reproduzindo rapidamente. Dessa forma, cópias desse individuo, assim como outros que sejam bem parecidos, tomam conta de uma grande parcela da população.

2.1.6 Função *Fitness*

A Função *Fitness* , ou adequação, é a função utilizada pelo algoritmo para avaliar o desempenho de um indivíduo dentro de um sistema.

2.1.7 Treinamento Competitivo

A técnica de consiste em realizar a neuroevolução para dois ou mais BOTs dentro de um sistema. Desso modo, os dois irão aprender juntos e tendem a encontrar indivíduos com alta adequação.

2.1.8 Aproveitar e Explorar

O termo Aproveitar e Explorar , do inglês *Explotation and Exploration*, diz respeito ao equilíbrio que técnicas de otimização aleatória tentam estabelecer. Por um lado *Aproveitar* uma solução boa encontrada para receber sua recompensa. Pelo outro, é necessário realizar a *Exploração* para que hajam oportunidades de encontrar soluções melhores.

2.1.9 Otimização Aleatória

diz respeito ao conjunto de técnicas que tenta evitar soluções sub-ótimas por meio da introdução de aleatoriedade nos algoritmos. Essa aleatoriedade possibilita que os algoritmos tomem decisões sub-ótimas de modo a explorar as possibilidades.

2.2 Entendendo a Inteligência Artificial

2.2.1 Definindo Inteligência Artificial

A Inteligência Artificial (IA) diz respeito ao conjunto de técnicas e algoritmos que realizam tarefas em um sistema que exijam pensamento similar ao dos seres humanos (MILLINGTON; FUNGE, 2009)(KONAR, 2005).

Segundo (MILLINGTON; FUNGE, 2009), existem três motivações para os estudos e trabalhos a respeito de IA. A primeira é a filosofia, que tenta entender como o pensamento funciona. A segunda motivação é a psicologia, que trabalha com os mecanismos do cérebro humano e seus processos mentais. Finalmente, a terceira motivação é a engenharia, que foca em criar algoritmos para realizar tarefas do mesmo modo que um ser humano. Além disso, (MILLINGTON; FUNGE, 2009) assume uma distinção entre a IA acadêmica, que é motivada pela filosofia, pela psicologia ou pela engenharia e a IA para jogos, que é motivada exclusivamente pela engenharia.

2.2.2 Engenharia

O presente trabalho está situado no contexto de IA para jogos. Portanto, do ponto de vista de engenharia, o interesse está no desempenho do algoritmo. Desse modo, aplicações de engenharia costumam ser híbridas. Ou seja, aplicações usam tanto as técnicas da IA naturalista quanto da IA símbolica (MILLINGTON; FUNGE, 2009)(ENGELBRECHT, 2007)(EBERHART, 2007).

Segundo (MILLINGTON; FUNGE, 2009), o caráter híbrido acontece por dois motivos. O primeiro é pela necessidade de equilibrar conhecimento e busca. Esse equilíbrio significa que quanto maior o conhecimento armazenado menor a necessidade de busca. Por outro lado, quanto maior a capacidade de realizar buscas, menor a necessidade de armazenar conhecimento. Trabalhar esse equilíbrio é inevitável.

O segundo motivo é que as técnicas da IA naturalista se sobressaem apenas em um conjunto específico de problemas. Ou seja, não são melhores que as ténicas da IA simbólica para qualquer situação. Por esses motivos, cabe ao desenvolvedor analisar o problema e verificar qual o algoritmo ou técnica melhor se adequa para resolver o dado problema. Na próxima seção, será detalhado o uso de IA híbrido no contexto de jogos.

2.3 Entendendo a Neuroevolução

A Neuroevolução é uma técnica de IA híbrida que aplica algoritmos genéticos para otimizar os pesos e topologia de redes neurais. Essa técnica é a base do presente trabalho, sendo para gerar o comportamento dos personagens controlados pelo computador do

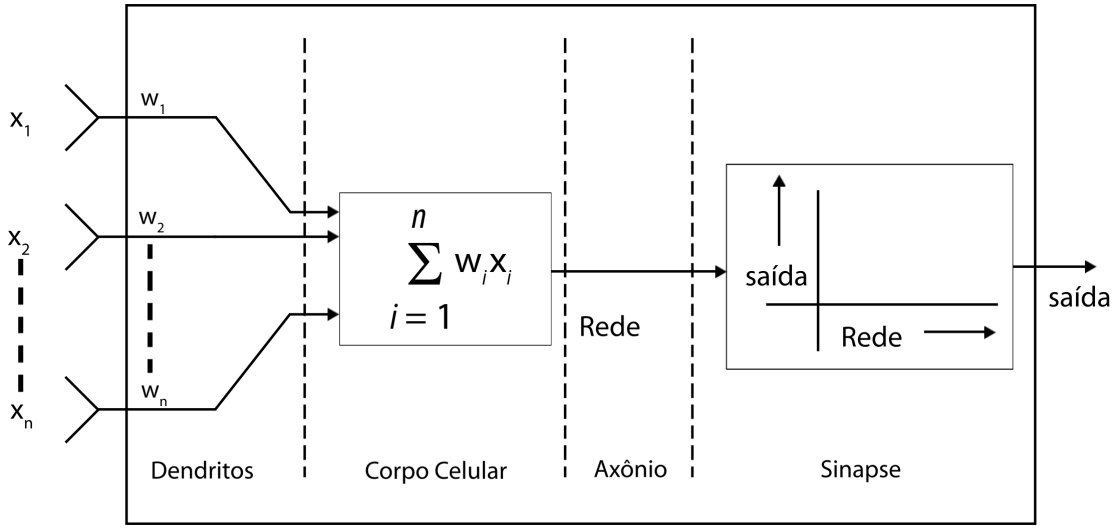


Figura 1 – Comparação de um neurônio natural e sua representação computacional por (KONAR, 2005).

jogo "*The Island*". Para a compreensão teórica do funcionamento de tal técnica, conceitos de Redes Neurais Artificiais, Algoritmo Genético e Inteligência Computacional serão introduzidos nas próximas seções.

2.3.1 Redes Neurais Artificiais

As Redes neurais artificiais (RNA) são inspiradas no sistema nervoso e sua incrível arquitetura paralelizada. A complexidade e capacidade do cérebro humano de aprender, memorizar, generalizar usando uma arquitetura paralelizada e não-linear motivou os estudos nesta área (ENGELBRECHT, 2007) (KONAR, 2005). O sistema nervoso dos humanos é composto por unidades fundamentais chamadas de neurônios que estão conectados em estruturas mais complexas, como o cérebro. Os neurônios artificiais, também chamados de perceptrons ou nós, são uma aproximação grosseira dos neurônios biológicos (EBERHART, 2007). Primeiro pela quantidade de neurônios. Enquanto um cérebro possui bilhões de neurônios, as RNA são pequenas e são projetadas para problemas específicos (ENGELBRECHT, 2007). Em segundo lugar, apesar dos neurônios artificiais funcionarem na ordem dos nanosegundos e os naturais na ordem dos milisegundos, a arquitetura de um cérebro abusa do paralelismo. Dessa forma, o cérebro ainda é mais rápido em algumas tarefas do que os computadores (EBERHART, 2007). A representação de um neurônio natural e um neurônio artifical podem ser vistas na Figura 1.

Um neurônio artificial processa um vetor de entradas usando um módulo de combinação linear e um outro módulo de não-linearidade. Tal representação simula as funções desempenhadas por dendritos, corpo celular e axônio de um neurônio natural. Em outras palavras, um neurônio artificial é caracterizado por um vetor de pesos, um limiar θ e

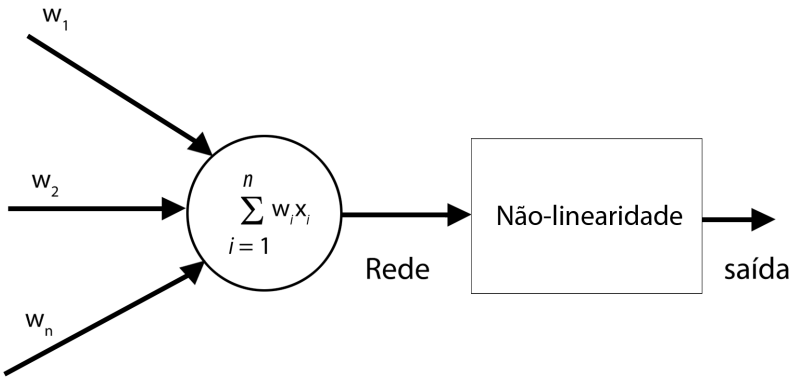


Figura 2 – Neurônio Artifical por (KONAR, 2005).

uma função de ativação (ENGELBRECHT, 2007). A representação computacional de um neurônio artifical pode ser vista na Figura 2.

O módulo de combinação linear é o sinal de entrada da rede para um neurônio artificial. O sinal de entrada da rede é dado pela soma ponderada das entradas subtraída do limiar θ, como é representado na Equação 2.1.

$$Rede = \sum_{t=1}^{I} w_i v_i - \theta \tag{2.1}$$

onde $\mathbf{w} = \{w_1, w_2, \ldots, w_{i-1}, w_i\}$ é o vetor de pesos do neurônio artificial, $\mathbf{v} = \{v_1, v_2, \ldots, v_{I-1}, v_I\}$ é o vetor de entradas, ambos com a quantidade I de elementos. Para simplificar a equação, pode-se considerar uma unidade especial chamada de *bias* que ocupa a posição v_{I+1}. A unidade *bias* sempre tem o valor -1. Dessa forma, é possível adicionar o limiar θ ao vetor de pesos na posição w_{I+1}. Portanto, substituindo esses valores na Equação 2.1, temos:

$$Rede = \sum_{t=1}^{I} w_i v_i + w_{I+1} v_{I+1} = \sum_{t=1}^{I} w_i v_i \tag{2.2}$$

O módulo de não-linearidade está relacionada a ativação de um neurônio artificial e assume a forma de funções de ativação (h). Essas funções, geralmente, fazem um mapeamento crescente de modo que, quando a entrada se aproxima de $-\infty$ o resultado é 0, ou -1, dependendo da configuração feita pelo desenvolvedor:

$$h(\infty) = 0 \text{ ou } h(-\infty) = -1 \tag{2.3}$$

e quando a entrada se distancia da origem em direção ao infinito positivo, o resul-

tado da função de ativação se aproxima do valor 1:

$$h(\infty) = 1 \tag{2.4}$$

Exemplos de funções de ativação são a função sigmóide, representada na Equação 2.5, e a função sinal, representada na Equação 3.3.

A função sigmóide é:

$$h(Rede) = 1/\left(1 + e^{-Rede}\right) \tag{2.5}$$

E a função sinal é:

$$h(Rede) = \begin{cases} +1, & \text{se } Rede < 0 \\ -1 & \text{Caso Contrário} \end{cases} \tag{2.6}$$

Neurônios artificiais podem ser usados para representar funções linearmente separáveis, como a função E e a função OU. Entretanto, podem ser combinados em diversas topologias de redes neurais, que são capazes de representar uma rica variedade de expressões não lineares, e submetidos a diferentes, e mais complexos, algoritmos de treinamento (ENGELBRECHT, 2007). Segundo (KONAR, 2005), os arranjos estruturais mais comuns são:

- **Rede de camada única com feedback lateral e recorrência** Os neurônios estão dispostos em uma única camada que recebe as entradas e produz as saídas. Existem conexões recursivas e conexões com até dois outros neurônios que acontecem nos dois sentidos.
- **Rede de duas camadas com estrutura feed-forward** Os neurônios estão dispostos em duas camadas. A camada de entrada que recebe os dados. Além disso, os neurônios da camada de entrada estão conectados aos neurônios da camada de saída. A camada de saída é a responsável por gerar os dados de saída.
- **Rede de duas camadas com feedback** A organização é parecida com a da rede de duas camadas com estrutura feed-forward. A diferença é que as conexões entre os neurônios acontece nos dois sentidos.
- **Rede de três camadas com estrutura feed-forward** A organização é parecida com a da rede de duas camadas com estrutura feed-forward. A diferença é de que existe uma camada entre a camada de entrada e a camada de saída que intercepta as conexões. Isto é, os neurônios da camada de entrada se conectam aos neurônios da camada intermediária que, por sua vez, se conectam aos neurônios da camada de saída.

- **Rede com estrutura recorrente e auto-loop** Os neurônios estão dispostos em uma camada com conexões recorrentes e conexões com todos os outros neurônios.

Por fim, um último a ser tratado quando o assunto é redes neurais diz respeito a como treiná-las. O processo de treinamento consiste em ajustar os parâmetros internos da rede e esse processo pode acontecer de três maneiras diferentes (KONAR, 2005)(EBERHART, 2007):

- **Treinamento supervisionado:** a rede neural recebe um conjunto de entradas e de saídas. Os pesos conexões passam por um algoritmo de aprendizagem supervisionada para corrigir e ajustar os parâmetros da rede. Dessa forma, o sistema ganha a capacidade de generalizar.

- **Treinamento não-supervisionado:** não existe um treinador que fornece as respostas corretas. Sistemas com esse tipo de aprendizado normalmente ajustam os parâmetros internos baseado em critérios estabelecidos pelo desenvolvedor. Geralmente, o treinamento não supervisionado é usado para identificar grupos de dados com características semelhantes.

- **Treinamento reforçado** : é o que mais se assemelha a sistemas biológicos (EBERHART, 2007). Isso se deve ao fato da existência de um mecanismo que realiza uma crítica da resposta. Caso a resposta tenha contribuído com o objetivo aquela configuração da rede é recompensada, caso contrário ela é penalizada.

2.3.2 Algoritmos Genéticos

Os algoritmos genéticos (AG) simulam o processo da evolução natural. Da mesma forma que o processo natural seleciona as espécies melhor adaptadas, tais algoritmos são usados em processos de otimização (KONAR, 2005). Esse tipo de algoritmos funciona melhor para problemas que lidam com imprevisibilidade (BOURG; SEEMANN, 2004).

O processo evolucionário começa com a criação da população inicial com um conjunto inicial de características. Em seguida, é feita uma avaliação de *fitness* para descobrir quais indivíduos estão melhor adaptados. Estes indivíduos são selecionados para a criação da próxima geração. A nova geração é criada por meio da evolução que combina as características dos indivíduos melhor adaptados, podendo incluir *crossover* e mutações (BOURG; SEEMANN, 2004) (EBERHART, 2007). Tal processo é baseado na sobrevivência do mais forte, que é alcançada através da reprodução e mutação a fim de produzir novas gerações (ENGELBRECHT, 2007). Cada indivíduo de uma população é referenciado como cromossomo, que, por sua vez, agrupa um conjunto de características chamadas de genes (ENGELBRECHT, 2007).

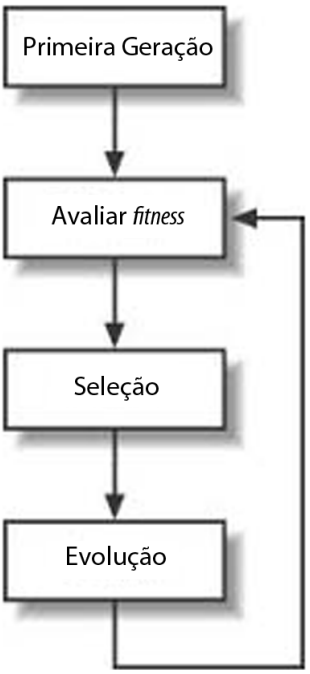


Figura 3 – Algoritmo Genético - Processo evolucionário (BOURG; SEEMANN, 2004).

A modelagem dos cromossomos artificiais a partir dos cromossos naturais é aproximada (EBERHART, 2007). Em primeiro lugar, a composição dos cromossomos artificiais é uma simplificação da estrutura complexa de um DNA. Em segundo, os cromossomos naturais possuem tamanhos variados, enquanto os artificiais possuem a mesma quantidade de bits. Por fim, enquanto os cromossomos naturais são duplicados ao longo da vida da célula e a reprodução combina proporções iguais dos progenitores, os cromossomos artificiais chamam de reprodução a duplicação dos cromossomos e chamam de *crossover* a produção de um novo cromossomo a partir dos cromossos dos pais (EBERHART, 2007).

Essa técnica oferece as vantagens de evitar a convergência em apenas máximos locais, dado as mudanças aleatórias causadas por mutações e *crossover*, e de possibilitar um paralelismo massivo na sua implementação, uma vez que cada indivíduo representa uma solução distinta. A Figura 3 representa o processo evolucionário de um algoritmo genético.

2.4 Inteligência Computacional

Segundo (ENGELBRECHT, 2007), a Inteligência Computacional (IC) é o ramo da IA que estuda e cria as técnicas e mecanismos capazes de aprender e se adaptar a ambientes complexos e em constante mudança. A IC também refere-se ao conjunto de modelos computacionais que processam dados numéricos de modo que seja usado todo o paralelismo do problema para obter respostas rápidas e precisas (KONAR, 2005).

Para a construção de um sistema de IC são usadas técnicas inspiradas em sistemas naturais ou biológicos, como redes neurais, lógica fuzzy e computação evolutiva. Em IC, as técnicas utilizadas em um dados sistema funcionam como partes que interagem sinergicamente para a construção do todo, ao mesmo tempo que eliminam as fraquezes do uso individual dessas técnicas (ENGELBRECHT, 2007). Isso significa que as técnicas de IC tem suas vantagens e desvantagens. Ou seja, nenhuma delas resolve simultâneamente os problemas de raciocínio, aprendizagem e otimização (KONAR, 2005). Porém, é possível criar sistemas que usem tais técnicas de maneira cooperativa.

Essa sinergia das técnicas pode proporcionar, por exemplo, que um sistema de IC tire proveito da aproximação de raciocínio da lógica fuzzy, da otimização proveniente dos algoritmos genéticos e da capacidade de aprendizado das redes neurais. Outro exemplo de comportamento sinérgico consiste em aplicar técnicas de computação evolutiva na otimização de redes neurais. Esse processo é chamado de neuroevolução (SCHRUM; MIIKKULAINEN, 2012).

2.4.1 Neuroevolução

Neuroevolução é uma técnica que aplica algoritmos genéticos para otimizar os pesos e topologia de redes neurais. Ou seja, é possível combinar a alta capacidade expressiva das RNAs com a capacidade de otimização dos algoritmos genéticos. A neuroevolução é uma técnica de IA com treinamento reforçado. A vantagem dessa técnica é que ela evita os mínimos locais, por se tratar de uma otimização aleatória. O mesmo não acontece em treinamentos supervisionados.

Para facilitar o entendimento, a neuroevolução pode ser descrita nos mesmos termos dos algoritmos genéticos: população, mutação e *crossover*. A população utilizada é o conjunto de redes neurais. Portanto, as redes neurais serão tratadas como indivíduos. As mutações podem ser qualquer mudança nos elementos de uma rede neural. Ou seja, mutações podem afetar os pesos que ligam os neurônios artificiais ou até mesmo a topologia da rede neural. Por fim, o crossover são os meios de combinar tais elementos a fim de criar novos indivíduos. O algoritmo utilizado para implementar a neuroevolução é chamado de Neuroevolução para Aumentar Topologias (NEAT) e será descrito a seguir.

No algoritmo NEAT (STANLEY; MIIKKULAINEN, 2002), um indivíduo é descrito por seu fenótipo, a rede neural propriamente dita, e por um genoma. A Figura 4 mostra como seria uma estrutura para representar um indivíduo neste algoritmo.

A estrutura contém a rede neural e uma representação linear das conexões da rede neural, o genoma é a estrutura que armazena a origem histórica daquela rede neural. A Figura 5 exemplifica a estrutura de uma rede neural e seu respectivo genoma.

O primeiro passo do algoritmo é criar redes neurais. Estas começam com uma

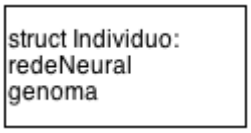


Figura 4 – Estrutura de um indivíduo.

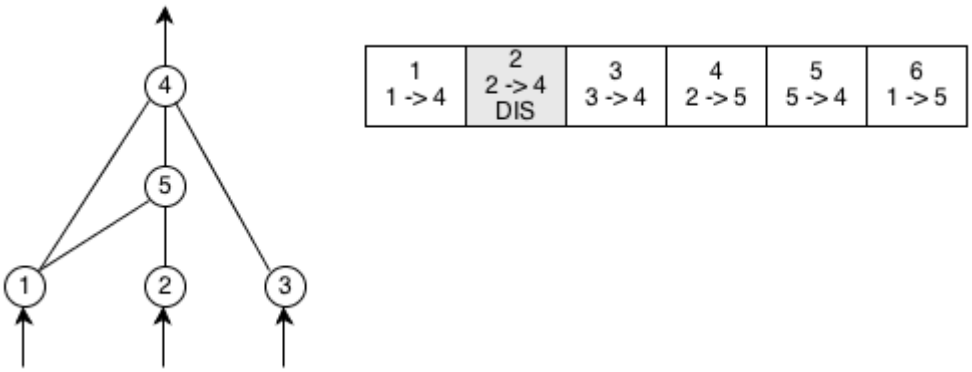


Figura 5 – Rede neural e seu respectivo genoma (STANLEY; MIIKKULAINEN, 2002).

estrutura mínima, ou seja, com apenas as camadas de entradas e saídas. Um exemplo representação dessa estrutura inicial é feita na Figura 6.

Durante as iterações do algoritmo acontecerão inovações no genoma. As inovações no genoma ocorrem por meio de mutações. As mutações usadas no NEAT são a pertubação de pesos, a criação de nós, e a mutação das conexões. A pertubação de pesos consiste em alterar ou manter o peso de cada conexão entre neurônios artificial em cada geração. As outras duas mudanças são estruturais e, consequêntemente, são as responsáveis em transformar a simples estrutura inicial em uma estrutura complexa e otimizada (STANLEY; MIIKKULAINEN, 2002). A Figura 7 esquematiza como são as mutações estruturais.

Após a execução de uma mutação estrutural, o tamanho do genoma da rede neural é expandido. O Algoritmo 1 mostra como uma nova conexão é criada:

Além da mutação de criação de conexões, existe a mutação para criar nós. Essa segunda mutação estrutural acontece quando uma conexão existente é dividida em duas e um novo nó é inserido entre essas novas conexões. Para esse mutação é necessário desabilitar a conexão antiga no genoma. Além disso, é necessário criar duas novas conexões. A primeira conexão parte do nó inicial para o novo nó e tem peso 1. A segunda sai do novo nó para o nó final e tem o peso da antiga conexão. O Algoritmo 2 mostra como uma nova conexão é criada.

É possível perceber tanto no Algoritmo 1 quanto no Algoritmo 2 que uma variável é incrementada. Essa variável é a $numeroGlobalInovacao$ e é usada nos algoritmos para

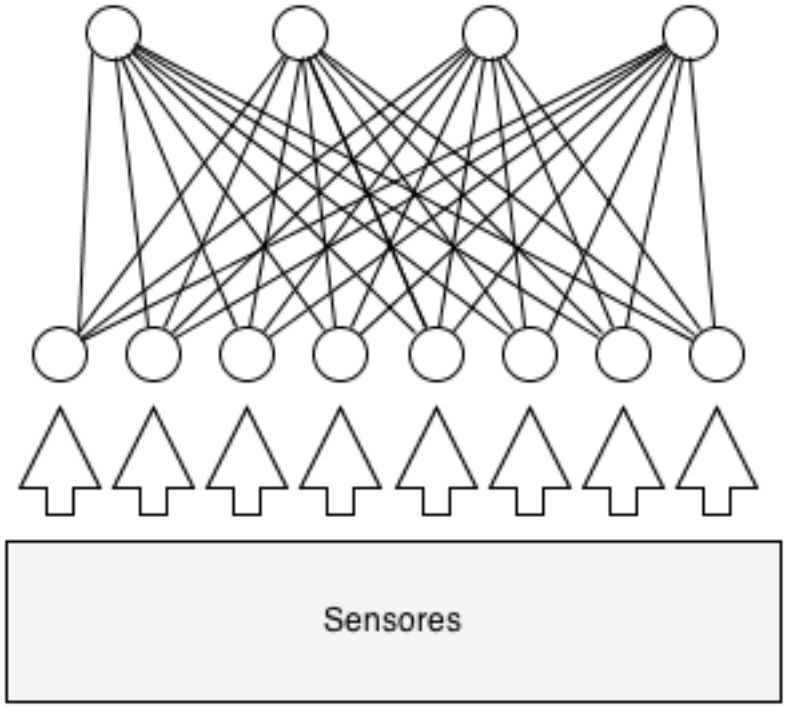


Figura 6 – Topologia inicial da Rede Neural.

```
input: indivíduo
output: indivíduoMutado
[noInicial, noFinal] <- encontrar dois nós não conectados no indiviuduo
If não encontrado noInicial e noFinal then
        Return vazio;
incrementar numeroGlobal;
pesoNovaConexao <- peso aleatório para a nova conexão;
novaConexao <- CriarNovaConexao(noInicial, noFinal, pesoNovaConexao);
individuoMutado <- cópia do individuo;
atualizar individuoMutado.redeNeural com novaConexao;
adicionar novaConexão e numeroGlobalInovacao ao individuoMutado.genoma;
Return individuoMutado;
```

Algoritmo 1: Algoritmo para mutação de criar conexão

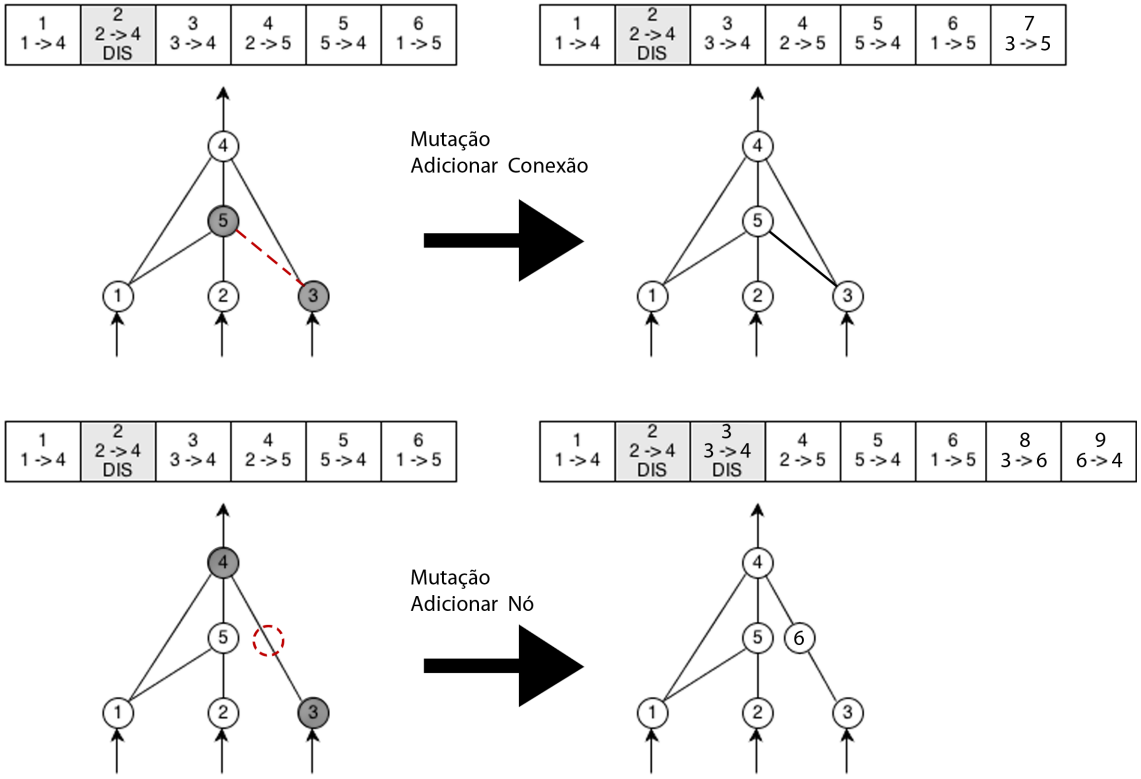


Figura 7 – Mutações estruturais por (STANLEY; MIIKKULAINEN, 2002).

```
input: indivíduo
output: indivíduoMutado
[noInicial, noFinal] <- encontrar dois nós não conectados no indiviuduo
If não encontrado noInicial e noFinal then
        Return vazio;
incrementar numeroGlobal;
novoNo <- criar novo nó;
pesoConexaoInicial <- 1;
pesoConexaoFinal <- obterPeso do individuo.genoma;
novaConInicial <- conecta noInicial e novoNo com pesoConexaoInicial;
novaConFinal <- conecta novoNo e noFinal com pesoConexaoFinal;
individuoMutado <- cópia do individuo;
atualizar individuoMutado.redeNeural com novaConInicial e novaConFinal;
adicionar novaConInicial e numeroGlobalInovacao ao individuoMutado.genoma;
adicionar novaConFinal e numeroGlobalInovacao ao individuoMutado.genoma;
desabilitar a conexão noInicial e noFinal no individuoMutado.genoma;
Return individuoMutado;
```

Algoritmo 2: Algoritmo para mutação de criar nó

descobrir a ordem cronológica de aparecimento dos genes. Esse capacidade de descobrir quais genes são homológos permite que o algoritmo NEAT realize o crossover. Para realizer um crossover, os genomas de dois indivíduos, chamados pais, são alinhados, gene a gene. Dessa forma é possível produzir um novo indivíduo a partir dos genes dos progenitores como pode ser observado na Figura 8.

Finalmente, dado que a população e os mecanismos de crossover e mutação foram definidos para o NEAT, basta executar o algoritmo genético. O resultado deve ser uma rede neural com alto valor de adequação segundo a função *fitness* definida.

2.5 Comportamento Inteligente em Jogos

As técnicas e os algoritmos apresentados no capítulo foram e continuam sendo usados em jogos nas unidades controladas por computador. Cada técnica, com suas vantagens e desvantagens, é usada nas unidades controladas por computador de acordo com a sua adequação aos requisitos funcionais da respectiva unidade.

O uso de redes neurais em jogos, por exemplo, oferecem duas vantagens (BOURG; SEEMANN, 2004). A primeira é a capacidade de simplificação, devido a redução de esforço em escrever código para representar máquinas de estados finitas ou um conjunto de regras complexas. A segunda é a possibilidade de adaptação *on-the-fly*, isto é, enquanto o jogo é executado. Por outro lado, o uso de redes neurais em jogos ainda não é muito difundido, pois é difícil predizer o comportamento de uma rede.

Outro exemplo no contexto de desenvolvimento de jogos, são os algoritmos gené-

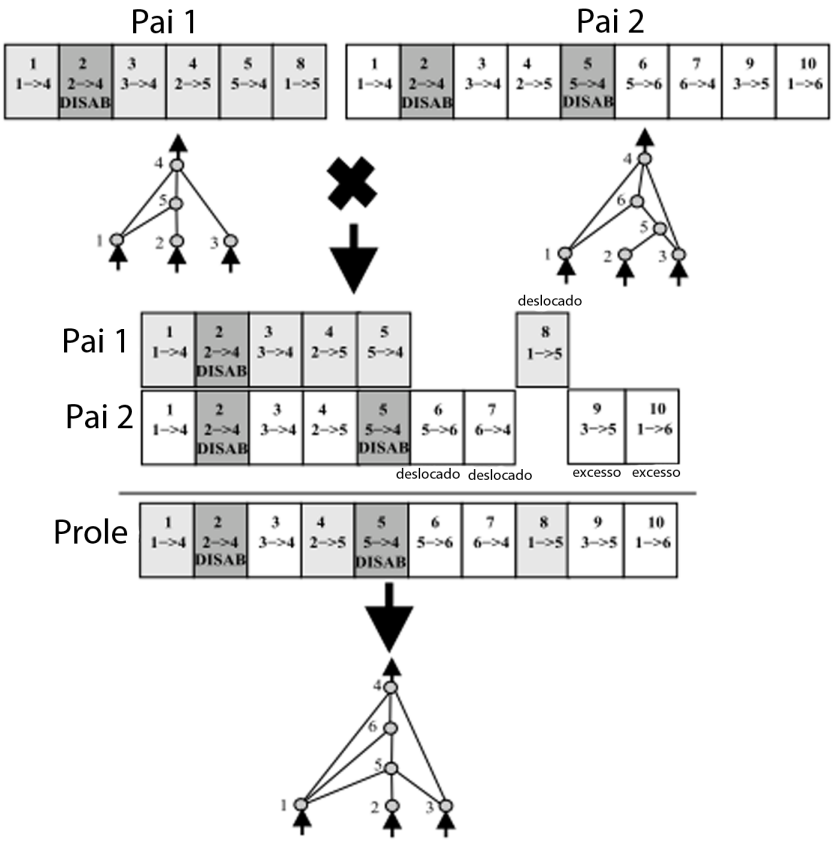


Figura 8 – Processo de *crossover* por (STANLEY; MIIKKULAINEN, 2002)

ticos, que possibilitam a IA seja adaptável às situações que não foram previstas no documento de *design*. Um exemplo de comportamento difícil de antecipar é o comportamento para reagir contra um jogador humano. Segundo (BOURG; SEEMANN, 2004)(SONI; HINGSTON, 2008), as soluções encontradas por um algoritmo genético irão variar de jogador para jogador e, por isso, a IA também será adaptável ao jogador.

As técnicas para aprendizado e reatividade ao comportamento do jogador se enquadram como IA Não-Determinística. Como visto na Seção 2.4, a inteligência computacional é uma das sub-áreas da IA com foco nas técnicas que tornam sistemas capazes de lidar esses tipos de situações. Outro aspecto importante da IC é a proposta de trabalhar com várias técnicas de maneira híbrida e sinérgica.

Apesar das dificuldades intrísecas das técnicas não-determinísticas, os desenvolvedores estão superando os entraves e desenvolvendo jogos de sucesso que usam esses métodos (BOURG; SEEMANN, 2004). Nesse casos, as técnicas não-determinísticas são usadas em situações em que são a melhor opção e, para as outras, a IA determinística continua sendo usada. Portanto, pode-se criar um sistema de inteligência artificial híbrido, possibilitanto ao desenvolvedor isolar as porções do sistema que irão atuar de maneira imprevisível (não-determinística) e usar a abordagem tradicional (determinística) no restante.

É com essa idéia de desenvolver um sistema não-determinístico, por meio da neuroevolução, juntamente com alguns elementos determinísticos, forma de conhecimento do domínio introduzida no sistema, que iremos criar um BOT que seja tanto imprevisível e desafiador. Em outras palavras, um BOT que seja um inimigo de alta qualidade. Essa abordagem ja foi testada e mostrou resultados promissores em (SONI; HINGSTON, 2008). O desenvolvimento da solução será apresentado a seguir no Capítulo 3

3 Prototipando com Neuroevolução

O capítulo apresenta toda a metodologia de desenvolvimento feita neste trabalho. Na Seção 3.1 trata sobre a escolha e inspiração do tema. A Seção 3.3 da informações sobre a visão geral e detalhe dos componentes do protótipo. Na Seção 3.4 são apresentados os recursos, tanto de software, como o motor gráfico Unreal Engine 4 e o pacote de IA *rtNEAT*, quanto de hardware, utilizados no projeto. As Seções 3.5 e 3.6 apresentam os guias utilizados para o desenvolvimento do projeto utilizando as ferramentas. A Seção 3.7 traz detalhates sobre a implementação na plataforma utilizada. Por fim, a Seção 3.8 conclui a descrição da metodologia ao apresentar como foi conduzido o experimento para entender a percepção dos usuários sobre o modelo de IA desenvolvido.

3.1 Escolha do Tema

3.1.1 O Jogo

O jogo proposto foi nomeado, provisoriamente, como "*The Island*" e será um jogo de tiro em primeira pessoa online com elementos de survival horror. O jogo se passa em uma ilha desconhecida onde um conjunto de jogadores humanos tem que sobreviver enquanto buscam um meio de escapar da ilha. Esta seção descreve uma visão geral das capacidades do jogo e mais informações podem ser encontradas na primeira versão do Documento de Design do Jogo (GDD - Game Design Document) no Anexo A.

Enquanto trabalham de maneira cooperativa, os jogadores encontram itens, como medicamentos, mapas, armas e chaves, que podem auxiliar o grupo de sobreviventes durante a jornada. Além disso, os itens estão dispostos de maneira aleatória pelo mapa, fazendo com que seja necessário explorar a ilha. Entretanto, os jogadores podem sofrer dano, que, quando acumulado em um curto período de tempo irá causar a morte do personagem. Cada jogador humano terá uma quantidade predefinida de "segundas chances". Após cada morte, uma "segunda chance" será decrementada do total. Quando não houverem mais "segundas chances", o jogador não poderá mais retornar ao jogo.

Existe um motivo especial para que os jogadores humanos se preocupem com a quantidade de dano recebido. Isto porque existem outros habitantes na ilha: o Enxame. O Enxame consiste de um conjunto de indivíduos alienígenas, os Colmeeiros, e uma colmeia central. Os objetivos do Enxame são eliminar os jogadores humanos e proteger a colmeia central. Os colmeeiros trabalham de maneira cooperativa para realizar tarefas como encontrar e atacar os jogadores ou para proteger o Enxame.

A colmeia central é a responsável pelo aparecimento/reaparecimento de colmeeiros.

Figura 9 – Tela com a idéia do jogo

Entretanto, novos colmeeiros podem surgir apenas quando um sobrevivente morre ou quando o número de colmeeiros está abaixo de um mínimo previamente estipulado. De qualquer maneira, todo colmeeiro deve esperar um tempo de amadurecimento antes de entrar no jogo. A Figura 9 mostra uma concepção inicial do jogo.

3.1.2 O Tema do Projeto

O tema do projeto é o desenvolvimento de um protótipo do jogo "*The Island*". Nesse protótipo será implementada a IA para controlar os colmeeiros, que são unidades que fazem dano corpo a corpo. A decisão de implementar apenas um protótipo se deu pelo fato do desenvolvimento de um jogo completo exigir tempo, experiência, recursos e pessoal não disponíveis. A partir do protótipo do jogo, o foco principal do trabalho foi projetar e implementar a IA dos personagens controlados por computador. Um fator importante é o de que a IA assume o papel de dois pilares para o jogo. O primeiro é por ser o componente de maior risco. Em outras palavras, se a técnica não for corretamente implementada ou não funcionar como o previsto, a proposta do jogo perde o sentindo. O segundo pilar é o da IA ser a base do jogo, isto é, a fonte dos momentos de diversão. Consequentemente, apenas será interessante seguir com o projeto se a IA for promissora em produzir BOTs de maior qualidade.

3.2 Percepções Preliminares

O trabalho realizado teve como referência o artigo (STANLEY; BRYANT; MIIKKULAINEN, 2005a) que usa a neuroevolução para controlar unidades em um jogo. Enquanto esse trabalho usa neuroevolução para controlar as **ações** das unidades em um

jogo, a proposta do presente trabalho é utilizar o mesmo modelo de IA para controlar as **tarefas** executadas pelos colmeeiros. Um outro artigo usado como foi referência foi o trabalho de (SONI; HINGSTON, 2008), no qual é proposto um protocolo de teste e avaliação dos resultados obtidos.

3.3 Detalhando o protótipo

Para facilitar o entendimento do que foi feito no protótipo, foram identificados os principais elementos que irão permitir a construção do projeto e como eles irão interagir. A Figura 10a contém uma visão geral de como o projeto do protótipo foi concebido enquanto a Figura 10b contém a estrutura analítica do projeto.

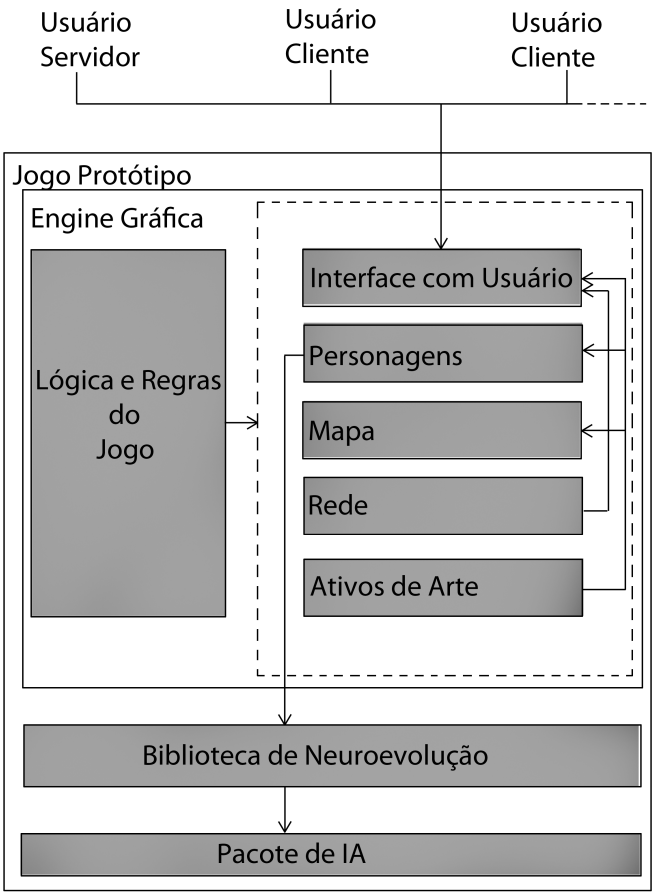
Cada um dos elementos da Figura 10b contribui de alguma forma com o protótipo a ser desenvolvido. O *Mapa* contém os elementos visuais e representa o mundo virtual em que o jogo acontece. Os *Personagens* são completamente especificados pelos desenvolvedores, desde a arte de modelos tridimensionais e animações até as interações do personagem com a engine gráfica. A *Rede* possibilita que o jogo seja multi-jogador. A *Programação* contém os elementos necessários para implementar o algoritmo de neuroevolução, que será usado pelos personagens controlados pelo computador. A *Simulação* representa a configuração necessária para executar o treinamento e avaliar os resultados. Por fim, a Simulação contém as regras do que irá acontecer no protótipo desenvolvido. As próximas seções detalhem o desenvolvimento desses elementos.

3.3.1 Programação

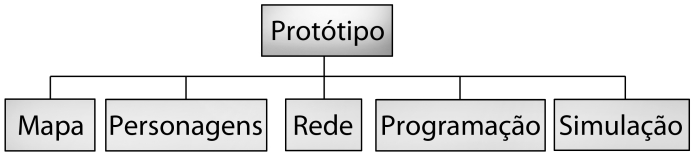
As atividades de programação são muito importantes no protótipo desenvolvido neste trabalho. Pois foi utilizado um pacote com os elementos para criar o algoritmo de neuroevolução e tal pacote é distinto do motor gráfico. A estrutura da solução é apresentada na Figura 11. Os manuais para usados para lidar com os aspectos de programação são apresentados na Seção 3.6.

A figura mostra o motor gráfico, *Unreal Engine 4*, utilizando as árvores de comportamento. Na árvore de comportamento existe um nó que irá realizar as chamadas das funções de uma biblioteca. Finalmente, as funções dessa biblioteca farão uso do pacote rtNeat, que contém o código fonte necessário para implementar algoritmos de neuroevolução. Dessa forma é necessário: *compilar códigos externos junto com o jogo* e *desenvolver um biblioteca que pode ser chamada pelo motor gráfico*. Para compilar os códigos do pacote rtNEAT é necessário adicionar, à primeira linha de todos os arquivos fontes, a diretiva include com o nome do projeto:

```
#include "ShooterGame.h"
```



(a) Visão Geral do Protótipo



(b) Estrutura Análitica do Projeto

Figura 10 – Visões do protótipo

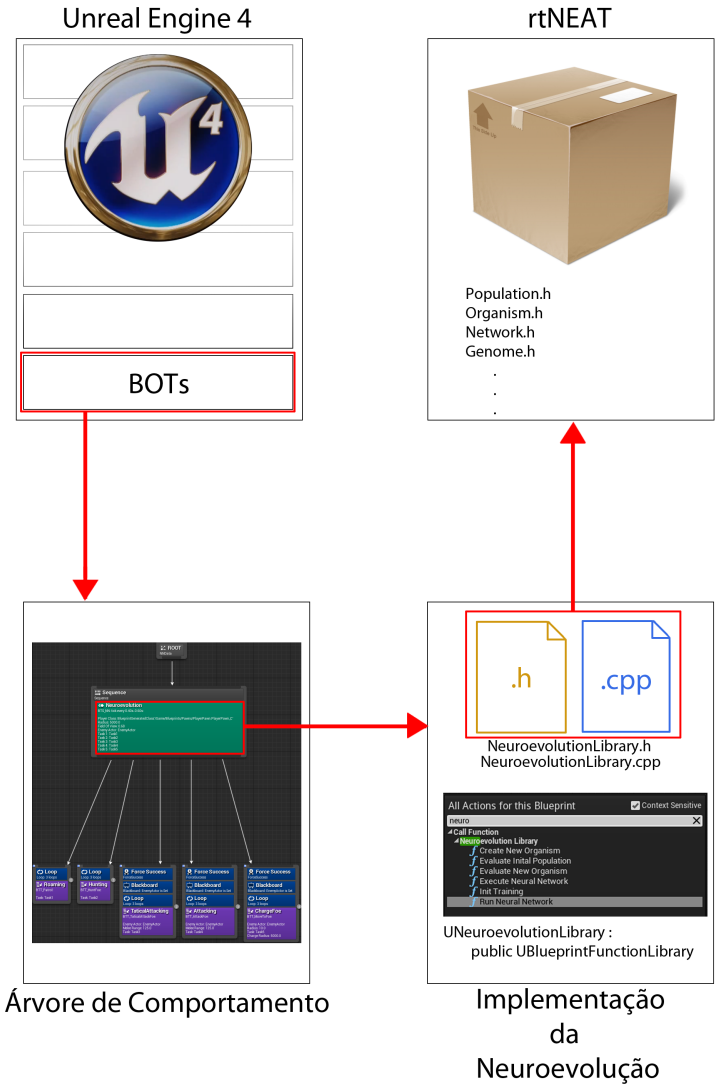


Figura 11 – Esquemático da solução do trabalho

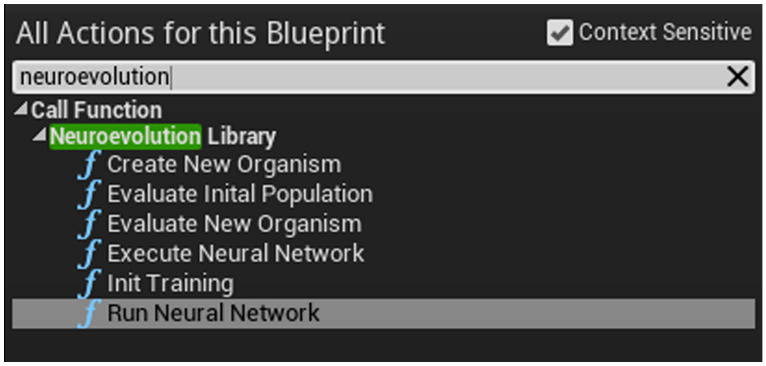


Figura 12 – Opções de usar as funções da biblioteca criada dentro do editor

A segunda necessidade é criar uma biblioteca e disponibilizar suas funções para serem chamadas pelo editor do motor gráfico. Para criar um biblioteca, o código deve herdar de uma classe específica do motor gráfico. Dito isso, a declaração da classe biblioteca, chamada de NeuroevolutionLibrary, foi:

```
UCLASS()
class SHOOTERGAME\_API UNeuroevolutionLibrary : public UBlueprintFunctionLibrary
```

Dentro da classe, as declarações das funções que vão ser disponibilizadas devem ser antecedidas pela seguinte macro:

```
UFUNCTION\(BlueprintCallable, Category = "NeuroevolutionLibrary"\)
```

A Figura 12 mostra que as funções *CreateNewOrganism*,*EvaluateInitialPopulation*, *EvaluateNewOrganism*, *ExecuteNeuralNetwork*, *InitTraining* e*RunNeuralNetwork* foram, de fato, disponibilizadas para o motor gráfico.

Dessa forma, a engine gráfica e o pacote de IA foram integradas. A Figura 13 apresenta o diagrama de sequência de sistema dessa integração.

3.3.2 Mapa

A construção do mapa possibilita que haja um ambiente onde os atores de um jogo interajam. Além disso, os mapas apresentam os elementos visuais que dão a sensação de imersão. A princípio, houve a tentativa de criar um mapa próprio para o protótipo. Entretanto, esta se mostrou uma tarefa que consumia muito tempo e com resultados de baixa qualidade, como pode ser observado na Figura 14a. Dessa forma, optou-se por usar um mapa pronto para implementar o protótipo. O mapa pronto já possuia relevo e elementos de decoração. A Figura 14b mostra o mapa pronto.

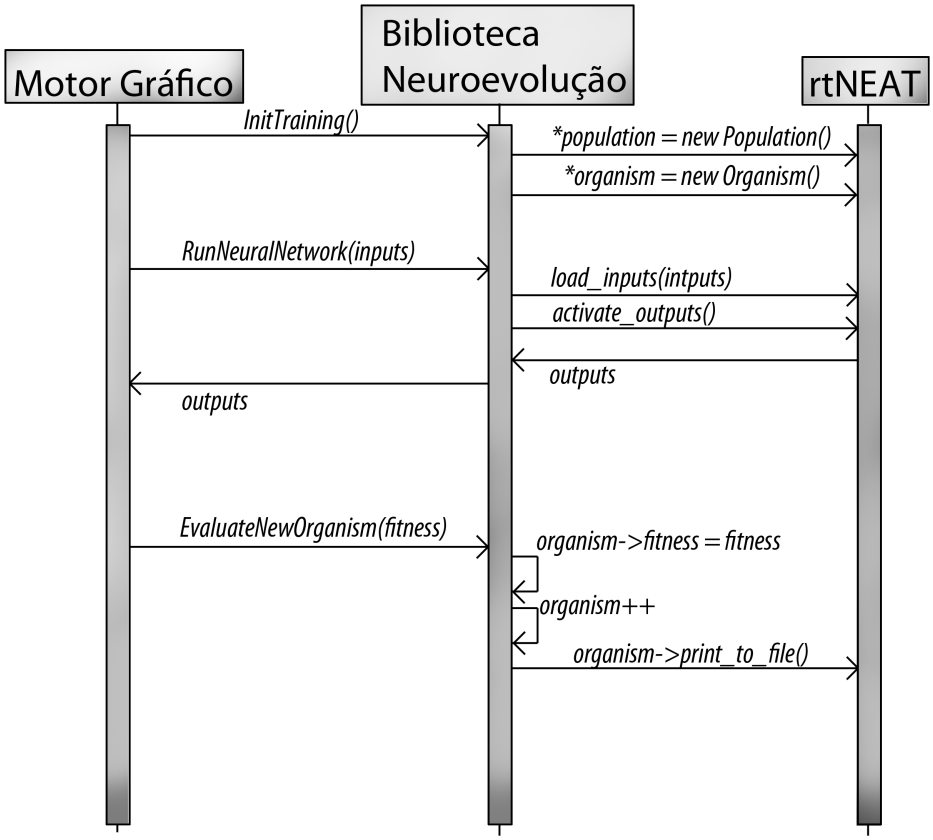


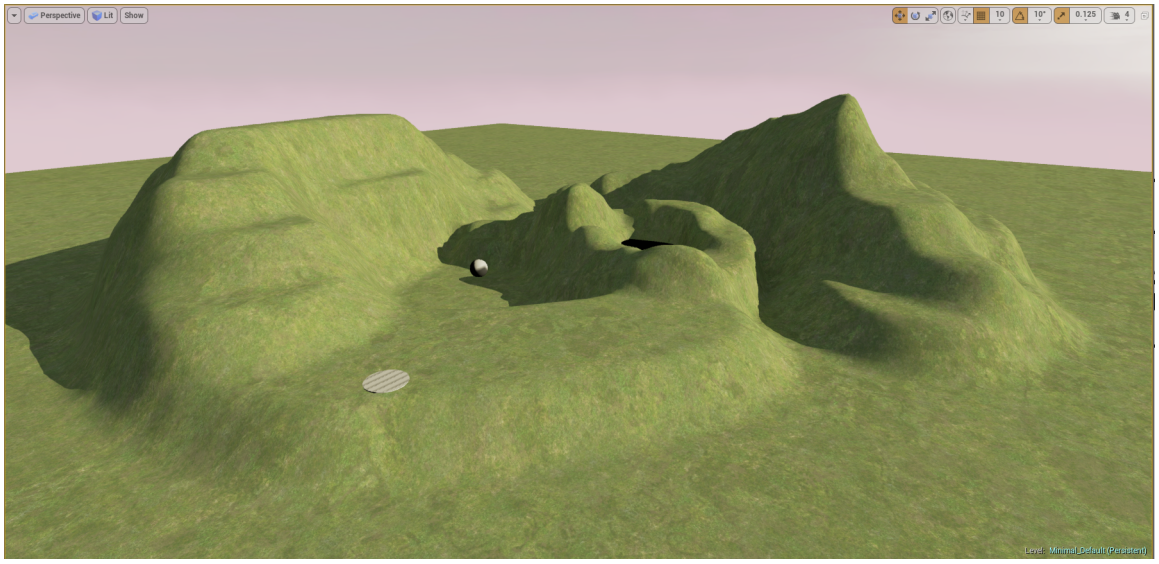
Figura 13 – Integração entre motor gráfico, biblioteca e pacote de IA

Apesar de ter sido descartado da versão principal do projeto, o mapa inicial (modelado) passou a ser usado como ambiente de teste de funcionalidades. Quando as funcionalidades estavam estáveis elas eram exportadas para o projeto com o mapa importado, que passou a ser o ambiente de funcionalidades estáveis. Além disso, a versão estável do projeto foi criada a partir de uma amostra de jogo de tiro em primeira pessoa (ENGINE, 2014l). Essa amostra ja possui alguns mecanismos básicos deste estilo de jogo, como armas, sistemas de mira e até alguns personagens e animações. O manual usado para editar mapas é apresentado na Seção 3.5.

3.3.3 Personagens

A criação de personagens envolve diversas tarefas. Entretanto, todas contribuem com o objetivo de inserir atores que podem ser animados e receber/enviar notificações de eventos, a fim de interagir com a engine gráfica. Para implementar esse módulo foi seguido o processo descrito na Figura 15.

O processo começa com a *Importação* de arquivos do tipo FBX (BLENDER, 2014). Tais arquivos contém tanto a malha tridimensional quanto as animações dessa malha. Após a importação é necessário *Configurar as Animações*. Essa etapa permite a criação



(a) O mapa criado no momento inicial do trabalho

(b) O mapa pronto utilizado no protótipo

Figura 14 – Mapas do protótipo

de transições suaves entre as animações dado variáveis do personagem. Por exemplo, foi criada a transição das animações do BOT Inimigo parado, andando e correndo usando a velocidade de movimento. A Figura 16 mostra três situações de animação dada a velocidade do jogador.

Um passo opcional é o *Redirecionamento de Animações*. O motor gráfico permite que animações sejam reusadas em novos modelos tridimensionais. Neste trabalho, essa atividade foi realizada para criar as animações do personagem do jogador. Um modelo desprovido de qualquer animação foi importado para o projeto e as animações pré-existentes na amostra de jogo de tiro em primeira pessoa. A Figura 17 mostra o personagem importado e realizando algumas das animações.

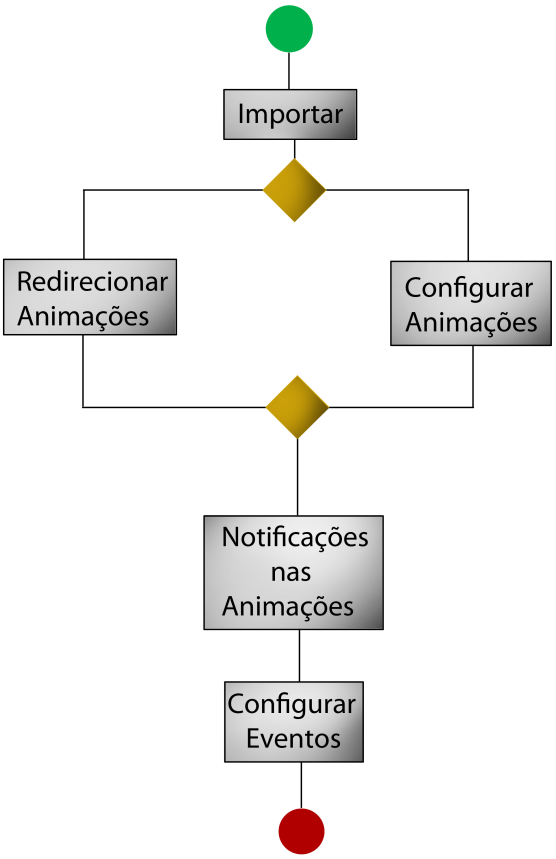
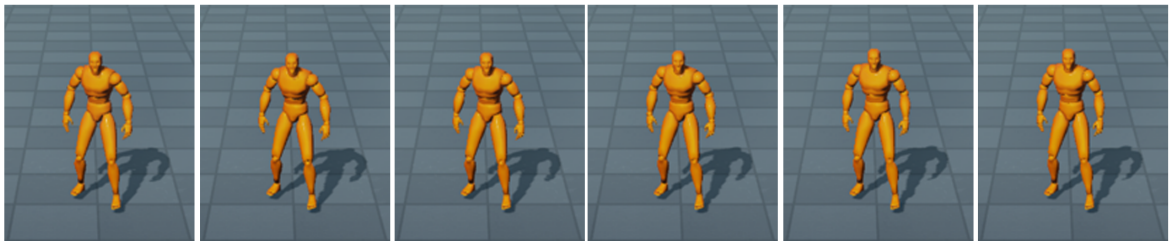


Figura 15 – Processo para desenvolver um personagem

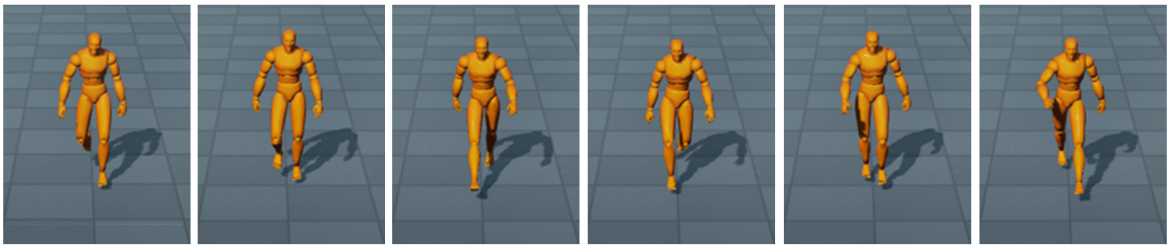
A atividade seguinte, é gerar *Notificações nas Animações*. Isso dá uma sensação de realismo, pois pode-se gerar eventos em momentos específicos. Por exemplo, são usados eventos quando os pés do BOT Inimigo tocam o chão para emitir partículas. Outra notificação é na animação de golpes corpo a corpo. Existe uma variável booleana usada para controlar quando o BOT está fazendo dano. Quando o BOT está esticando o braço essa variável recebe o valor verdadeiro e, quando o BOT está recuando o braço ela recebe valor falso. A Figura 18 mostra a linha do tempo das animações com sinalizações vermelhas, que são os momentos em que a notificação acontece.

O último passo é *Configurar Eventos*, possibilita que o personagem interaja com a engine gráfica. Personagens podem criar e reagir a eventos, como uma tecla sendo pressionada, quando o personagem anda ou quando o personagem morre, por exemplo. O personagem usado para os BOTs Inimigos pode receber sinais para começar a realizar golpes corpo a corpo. O mesmo sinal é usado nos BOTs jogadores, porém o efeito é o de disparar a arma. No caso do BOT Inimigo, o evento é usado para controlar uma variável booleana para indicar se o BOT deveria está realizando a animação de dar socos ou não. A Figura 19 mostra a BOT Inimigo dando socos.

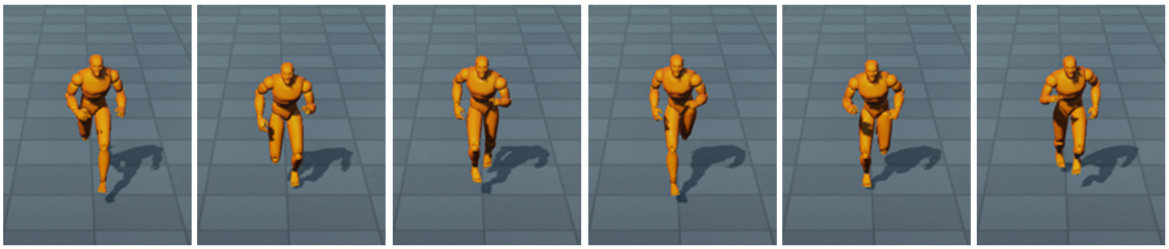
Por fim, ao final desse processo obtém-se personagens que podem ser inseridos



(a) Personagem Parado (velocidade = 0)



(b) Personagem Andando (velocidade = 70)



(c) Personagem Correndo (velocidade = 300)

Figura 16 – Transição de Animações

como atores em um jogo. Os manuais usados para realizar as atividades desse processo são apresentados na Seção 3.5.

3.3.4 Rede

Um dos requisitos do jogo é que ele suporte múltiplos jogadores humanos. Para isso é necessário que atores, variáveis e funções sejam replicados do servidor para os clientes. Existem diferentes opções para realizar essa replicação. A importância da implementação desse módulo no protótipo é o fato que alguns desses atores, variáveis e funções serão usados na versão final do jogo. Dessa forma, é interessante aproveitar esse momento para desenvolver os elementos do jogo com as preocupações sobre o que deve ser replicado e como deve ser replicado. Nesse protótipo por exemplo, os atores dos jogadores são replicados, assim como as variáveis que representam os pontos de vida desses jogadores. Por fim, quando os jogadores morrem, funções são replicadas do servidor para os clientes a fim de garantir a consistência do estado do jogo. A Figura 20 mostra duas instâncias do jogo acontecendo em rede com dois jogadores humanos. Os manuais para usados para

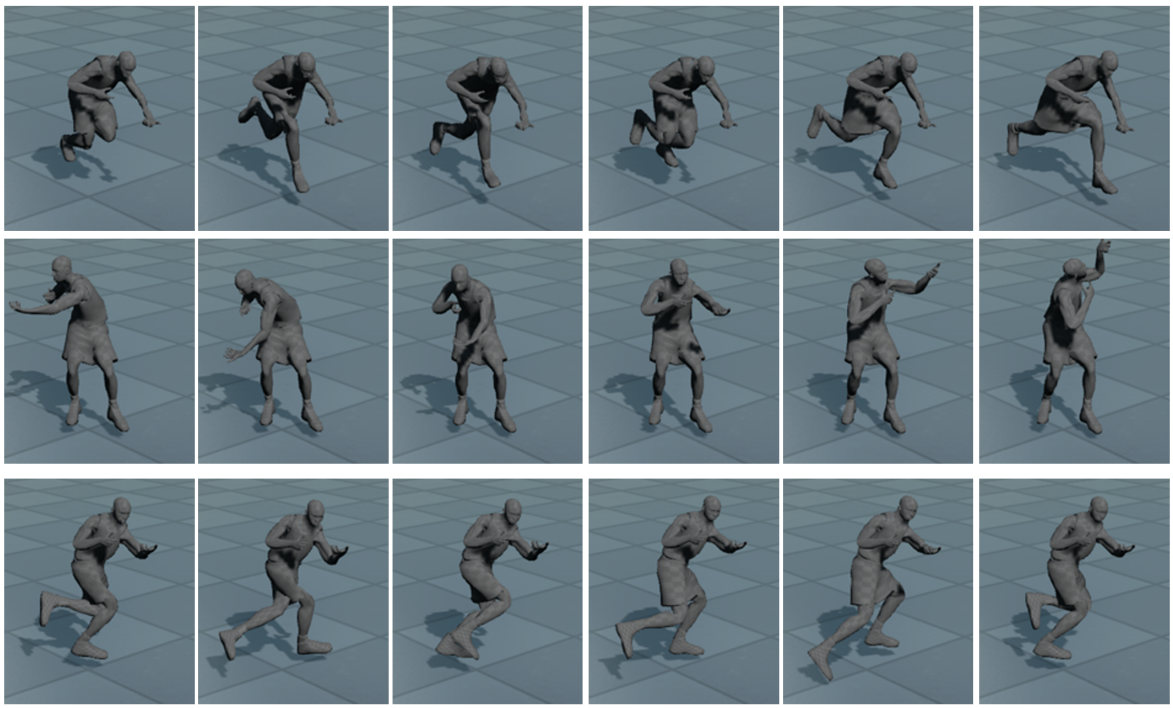


Figura 17 – Personagem do jogador, que foi importado, realizando algumas animações

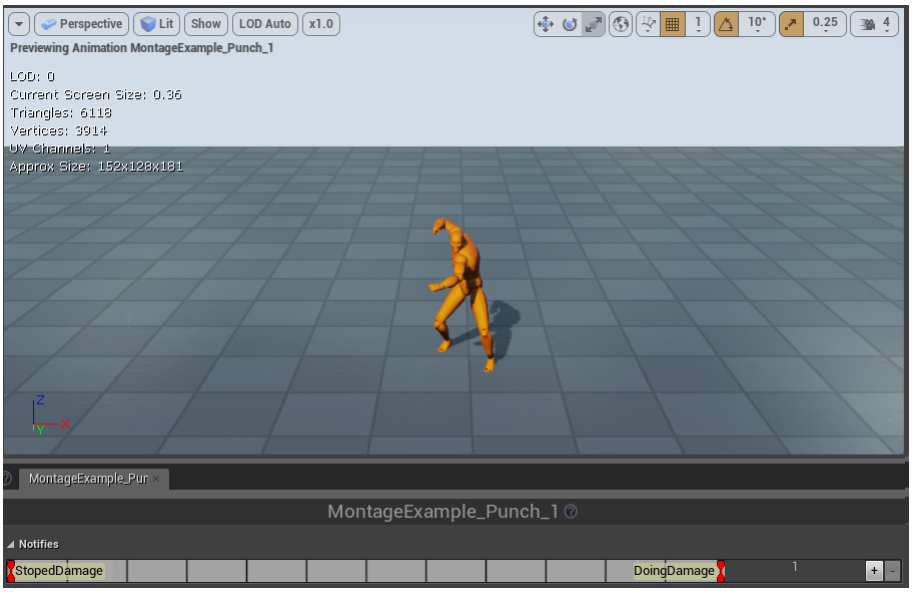
implementar os aspectos de rede são apresentados na Seção 3.6.

3.3.5 Simulação

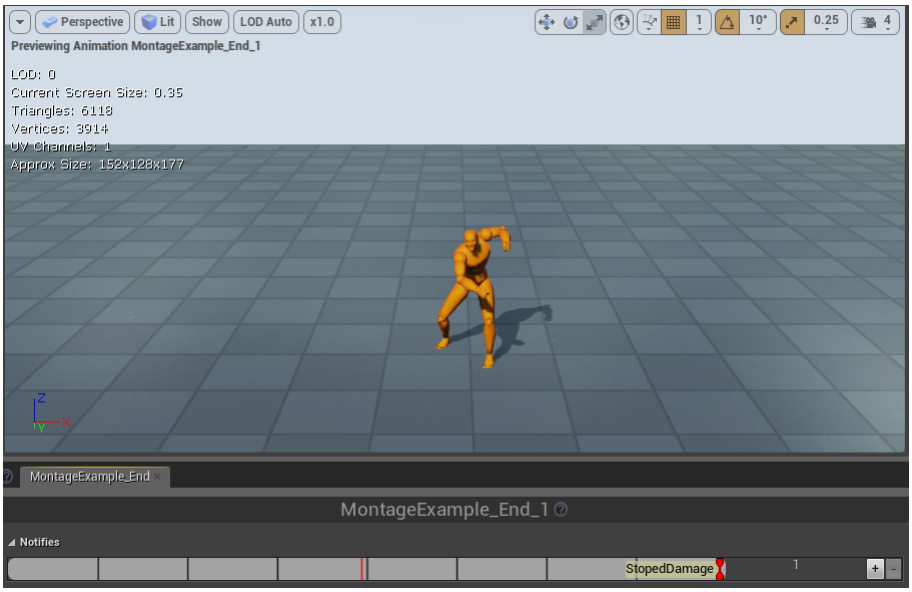
Com os elementos do protótipo apresentados até então possibilitam que a simulação da partida seja criada. Nessa simulação existem dois tipos de personagens: os jogadores e os inimigos. Ambos são controlados por BOTs com diferentes algoritmos de IA. O BOT Jogador é controlado por uma implementação tradicional do motor gráfico. Os BOTs inimigos podem ser controlados tanto pela implementação tradicional quanto por uma variação que utiliza a neuroevolução, dependendo do objetivo da simulação. Mais informações sobre a implementação na IA no motor gráfico podem ser encontradas na Seção 3.7.2. Durante a simulação, os BOTs jogadores tentam chegar até um ponto de controle e, se forem bem sucedidos, eles recebem a vitória. Por outro lado, os BOTs Inimigos tentam eliminar os BOT jogadores. Existe algumas variações nas regras de cada simulação, que são apresentadas na seção 3.7.1. Independente do modo de jogo, as partidas foram projetas para criar dois BOTs oponentes e três BOTs jogadores e, ao conceder a vitória à um dos times, a simulação é reinicializada.

A simulação acontece em um protótipo simplificado. Foram utilizadas apenas uma porção do mapa e executada uma simulação por vez. A Figura 21 abaixo mostra os elementos do mapa do protótipo.

O protótipo simplificado torna a simulação da partida mais rápida. Um mapa

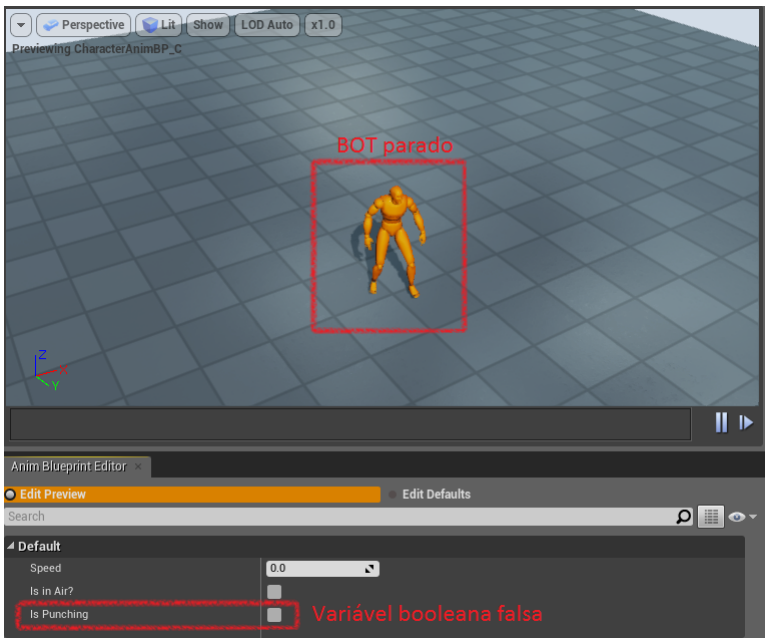


(a) Notificações para começar a fazer dano

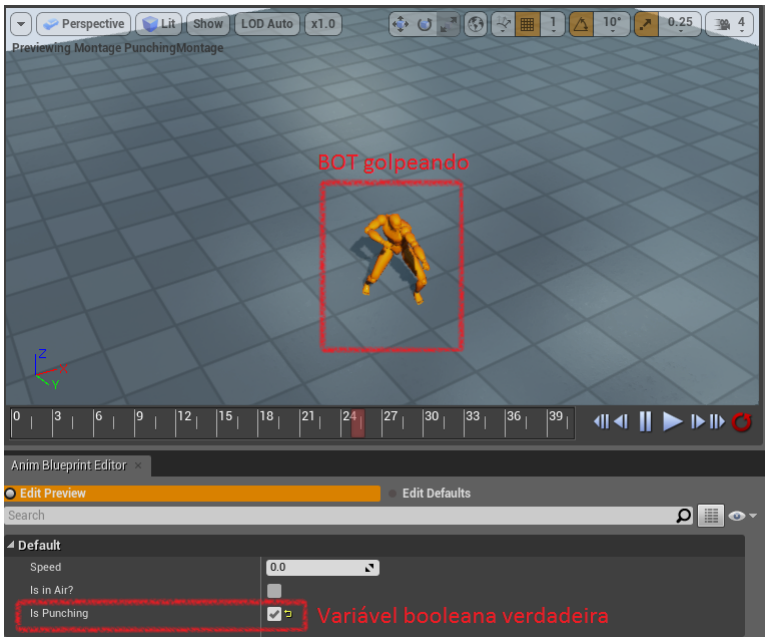


(b) Notificações para interromper a realização de dano

Figura 18 – Notificações nas Animações



(a) Variável booleana com valor falso



(b) Variável booleana com valor verdadeiro

Figura 19 – BOT recebendo eventos para mudar a animação

(a) Tela do Jogador 2

(b) Tela do Jogador 1

Figura 20 – Duas telas de dois jogadores humanos distintos

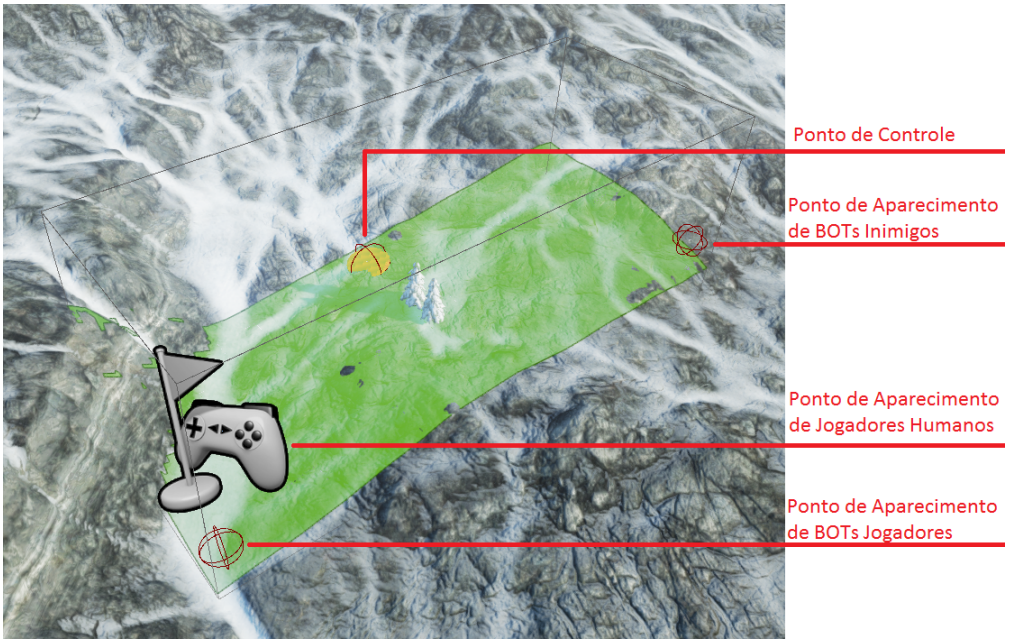


Figura 21 – Elementos do mapa do protótipo

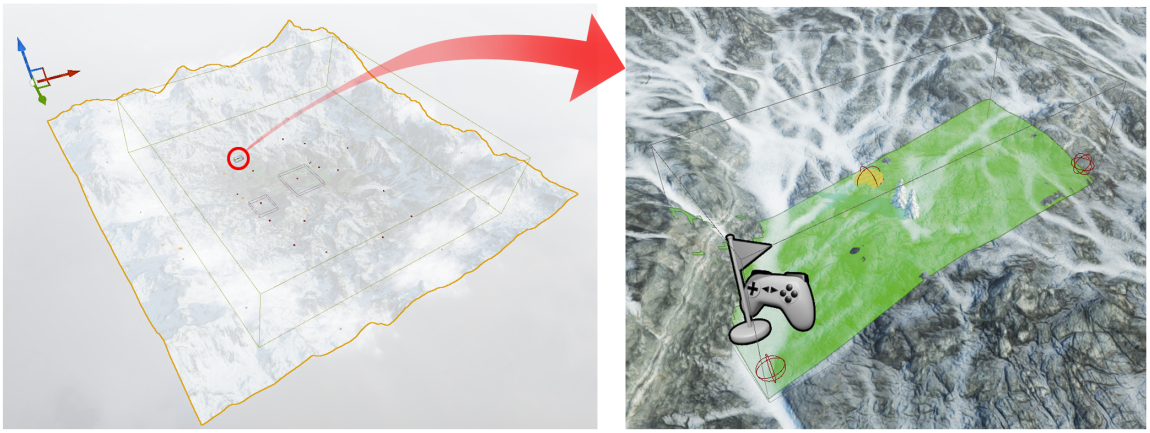


Figura 22 – Comparação do mapa completo e área utilizada

menor reduziu o tempo de viagem dos BOTs, além de diminuir o tempo de duração para o cálculo da malha de navegação. Para se ter uma idéia, o tempo médio para o cálculo da malha de navegação era de 26 horas no mapa original. E a malha era recalculada toda vez que uma modificação no mapa era feita. O uso de BOT jogadores permitiu que o treino fosse automatizado.

Além dessa decisão de uma simulação simplificada, foi tomada a decisão de executar a neuroevolução sem tirar vantagem do seu paralelismo. Ou seja, existe apenas uma simulação do jogo por vez, que, após executada, salva o valor calculado do*fitness* do indivíduo para o módulo de treino. Após receber essa variável, o módulo de treino atualiza os dados sobre a população em um arquivo e inicia uma nova simulação para o próximo indivíduo.

Componente	Descrição
Placa Mãe	ASUS P8Z68-VLX
Placa de Vídeo	NIVIDIA GeForce GTX 560
Memória RAM	2x8GB DDR3 1600MHz/1.5V (PC3-12800) Corsair Vengeance
Processador	Intel(R) Core(TM) i5-3570K CPU @ 3.40Ghz
Disco Rígido	SAMSUNG HS103SJ

Tabela 1 – Componentes de hardware utilizados

Essas decisões foram tomadas devido à maneira na qual a função *fitness* era calculada. A função foi elaborada de tal forma que é necessário saber o resultado de uma simulação para saber se os BOTs foram eficientes ou não em eliminar os jogadores. Em contra partida, essas simplificações tornaram possível a realização do treinamento. A Seção 3.7.3 detalha esse treinamento.

3.4 Tecnologias Utilizadas

3.4.1 Hardware

Alguns dos componentes não são descritos por não serem considerados essenciais para obter a mesma performance. Dessa forma, os componentes hardware utilizados foram: O uso desse hardware trouxe a vantagem de ser uma opção barata e eficaz para implementar o protótipo e realizar o desenvolvimento. Porém, a aquisição de hardware mais potente se faz necessário para a implementação dos próximos protótipos, que necessitem explorar melhor a vantagem de paralelismo massivo que a neuroevolução possibilita e de soluções mais robustas, como fazer um treinamento competitivo entre BOTs da IA contra BOTs jogadores usando neuroevolução.

3.4.2 Software

Foram utilizadas duas ferramentas para auxiliar no desenvolvimento do projeto. A primeira foi um motor gráfico para jogos chamado Unreal Engine (SWEENEY, 2014) em sua versão 4.4 . A segunda foi um pacote de IA chamado rtNeat na versão 1.0.2. Tal pacote contém o código fonte implementando a Neuroevolução em Tempo Real para Aumento de Topologias (rtNeat) (STANLEY; BRYANT; MIIKKULAINEN, 2005b).

Um dos maiores riscos no desenvolvimento do jogo é o uso dessas ferramentas, uma vez que há diversos riscos associados à integração entre esses dois sistemas. Isso aconteceria por falta de experiência com o uso destas ferramentas e pelo fato que não foi encontrado relatos de algum tipo de integração anterior a esse trabalho. Esse risco tem alta probabilidade de acontecer, porém com impacto moderado. Em contrapartida, não

Necessidade	Peso	Engine própria	Engine pronta
Flexibilidade da solução	1	10	5
Produtividade	10	2	8
Curva de aprendizado	10	8	8
Nota Final		**110**	**165**

Tabela 2 – Componentes de hardware utilizados

utilizar as engines introduz um risco com probabilidade alta de tempo insuficiente para a conclusão do trabalho, ou seja, impacto alto.

Para tomar essa decisão é necessário avaliar quais são as necessidades do projeto, ou seja, os critérios para a escolha das ferramentas, assim como a importância de cada um desses critérios durante a avaliação. Além disso, é interessante considerar os prós e os contras de cada opção. A respeito das necessidades do projeto é possível dizer que:

- A opção permite a customização de soluções para facilitar a integração com a IA é pouco importante (Prioridade Baixa);
- A opção proporciona uma alta produtividade (Prioridade Alta);
- A opção apresenta uma curva de aprendizado elevada (Prioridade Alta).

Atribuindo aos valores de prioridade baixa, média e alta os valores de 1,5 e 10, respectivamente, podemos dar notas de cada opção para cada necessidade. Esses pesos foram definidos de acordo com a maturidade e experiência do autor na etapa de tomada de decisão. Dessa forma, é possível calcular a média ponderada de cada abordagem. O resultado foi de usar uma solução pronta e pode ser observado na Tabela 4.

3.5 Configuração do Ambiente e Ferramentas

A configuração do ambiente foi feita utilizando tutoriais, disponibilizados em inglês pela desenvolvedora do motor gráfico. A tabela a seguir mostra as atividades mais importantes e que são necessárias para a manutenção do sistema.

3.6 Fluxo de Trabalho e Boas Práticas

Tal como na seção anterior, a próxima tabela também contém atividades importantes para o fluxo de trabalho. Algumas delas foram derivadas de tutoriais e outras da documentação do motor gráfico.

Atividade	**Tutorial**
Criação do Mapa	Getting Started: Introduction to UE4 Level Creation (ENGINE, 2014n)
Configuração do Personagem	
Importar conjuntos de artes (FBX)	FBX Importing and Using Skeletons (ENGINE, 2014k)
Persona	Intro to Persona (ENGINE, 2014q)
Blend Spaces	Blend Spaces (ENGINE, 2014d)
Animation Blueprints	Intro to Animation Blueprints (ENGINE, 2014o)
AnimGraph	Building the AnimGraph (ENGINE, 2014g)
EventGraph	Animation Blueprint EventGraph (ENGINE, 2014b)
Animation Montage	Intro to Animation Montage in UE4 (ENGINE, 2014p)
Redirecionamento de Animação	Skeleton Retargeting and Montage Setup in UE4 (ENGINE, 2014u)
Notificações de Animação	Creating Animation Notifies in UE4 (ENGINE, 2014i)
Componentes do Personagem	Character Blueprint Components (ENGINE, 2014h)
Configuração de Controles	Setting Up Inputs(ENGINE, 2014t)

Tabela 3 – Componentes de hardware utilizados

Atividade	**Tutorial**
Networking	
Replicação de Atores e Variáveis	Actor and Variable Replication (ENGINE, 2014a)
Replicação de Funções	Function Replication (ENGINE, 2014m)
Boas Práticas	Network Relevancy (ENGINE, 2014r)
Programação	
Criar um projeto c++ no editor	Project Creation (ENGINE, 2014s)
Adicionar código c++ ao projeto	Creating the Base Pickup Class (ENGINE, 2014j)
Criando Bibliotecas	Blueprint Function Library, Create Your Own to Share With Others (ENGINE, 2014e)
Disponibilizar funções para o editor	Blueprints, Creating C++ Functions as new Blueprint Nodes (ENGINE, 2014f)

Tabela 4 – Componentes de hardware utilizados

3.7 Implementação na Plataforma UDK

3.7.1 Modos de Jogo

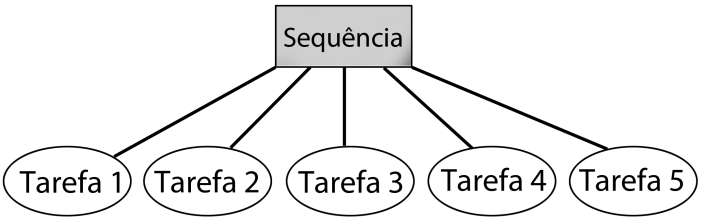
Os modos de jogo são as regras que criam o tipo de jogo. Para o protótipo foram criadas três modos de jogo para o trabalho: treinamento, execução e avaliação. Cada um desses modos possui um objetivo distinto. A função do modo de jogo é decidir qual IA deve controlar o BOT jogador e o tipo de IA que controla o BOT oponente.

- **Treinamento:** O objetivo desse modo é executar a neuroevolução. A cada simulação um novo indivíduo é avaliado.
- **Execução:** O objetivo desse modo é executar o indivíduo com maior adequação encontrado no modo de treinamento. O mesmo indivíduo é executado a cada simulação.
- **Avaliação:** O objetivo desse modo é executar IAs distintas para que possam ser realizados experimentos. Existem duas etapas, cada uma com duração de cinco minutos. Durante cada etapa um dos BOTs, *Neuroevolução:* ou *Linha de Base*, é utilizado até o fim da mesma. Após o fim da segunda etapa o jogo não gera reinicializa mais as animações.

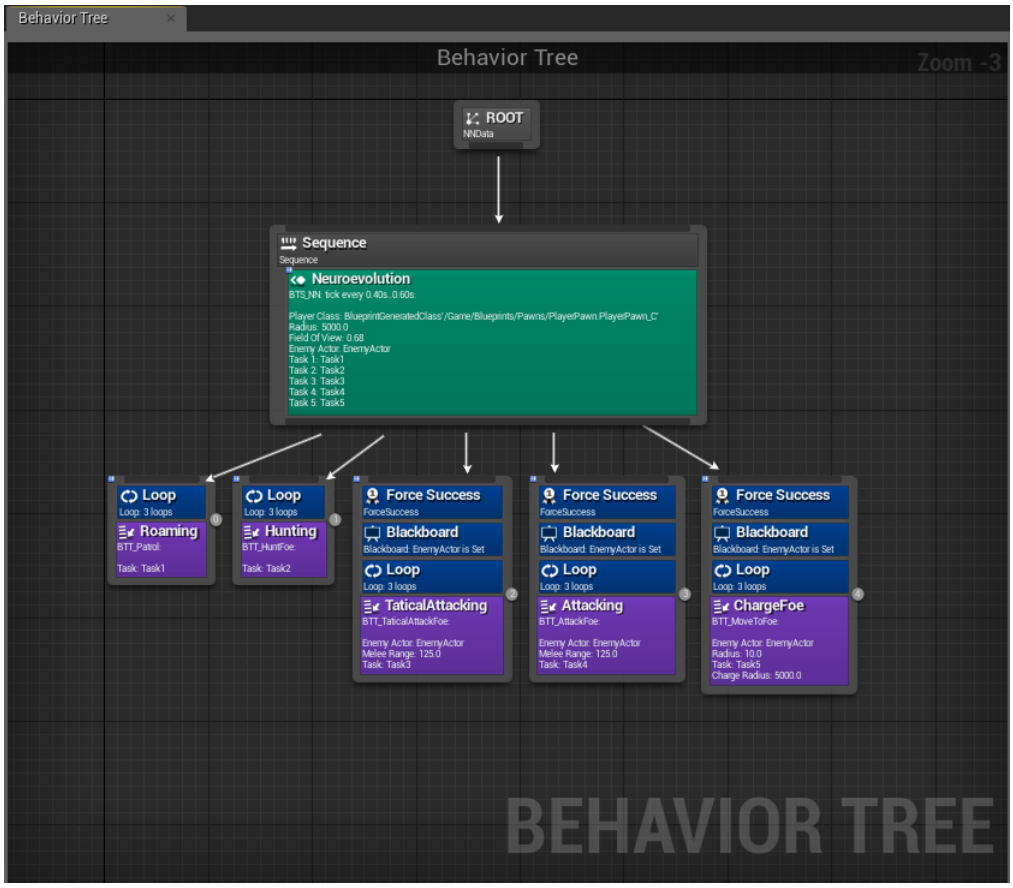
3.7.2 Ávore de Comportamento

Uma Árvore de Comportamento (BT) é a ferramenta usada no controle dos BOTs na engine gráfica. Uma BT possui cinco tipos de elementos: ROOT, Compósitos, Tarefas, Decoradores e Serviços. O primeiro é comum a qualquer BT e é por onde sua execução começa, é o chamado nó *ROOT*. Compósitos podem são executados para construir a estrutura da árvore. Estes podem ser do tipo Sequência ou Seletor, variando apenas a maneira como eles executam os nós filhos. O segundo elemento são as tarefas. As tarefas são nós terminais, onde as funções controle do BOT são chamadas. O terceiro elemento são os Decoradores. Decoradores são usados para modificar a execução do nó que está sendo decorado. Tanto Compósitos quanto Tarefas podem receber Decoradores. Por fim, os Serviços são modificadores que permitem a execução de diversos serviços que acontecem dentro de um mesmo frame do jogo(ENGINE, 2014c). Por fim, um outro conceito importante no uso de BTs é o Quadro Negro (*Blackboard*). *Blackboards* são usados para compartilhar variáveis pelos Serviços, Decoradores e Tarefas dentro de uma BT. A Figura ?? mostra que a idéia é executar as tarefas em sequência dentro de uma árvore de comportamento.

A Figura 23b é a estrutura usada nos BOTs *Neuroevolução* do protótipo. Aqui foi usado um serviço que realiza as chamadas para a biblioteca de neuroevolução. Além



(a) Ideia de uma Árvore de comportamento



(b) Os quadrados roxos são as Tarefas, os quadrados azuis são Decoradores e quadrados verdes são Serviços

Figura 23 – Árvores de Comportamento

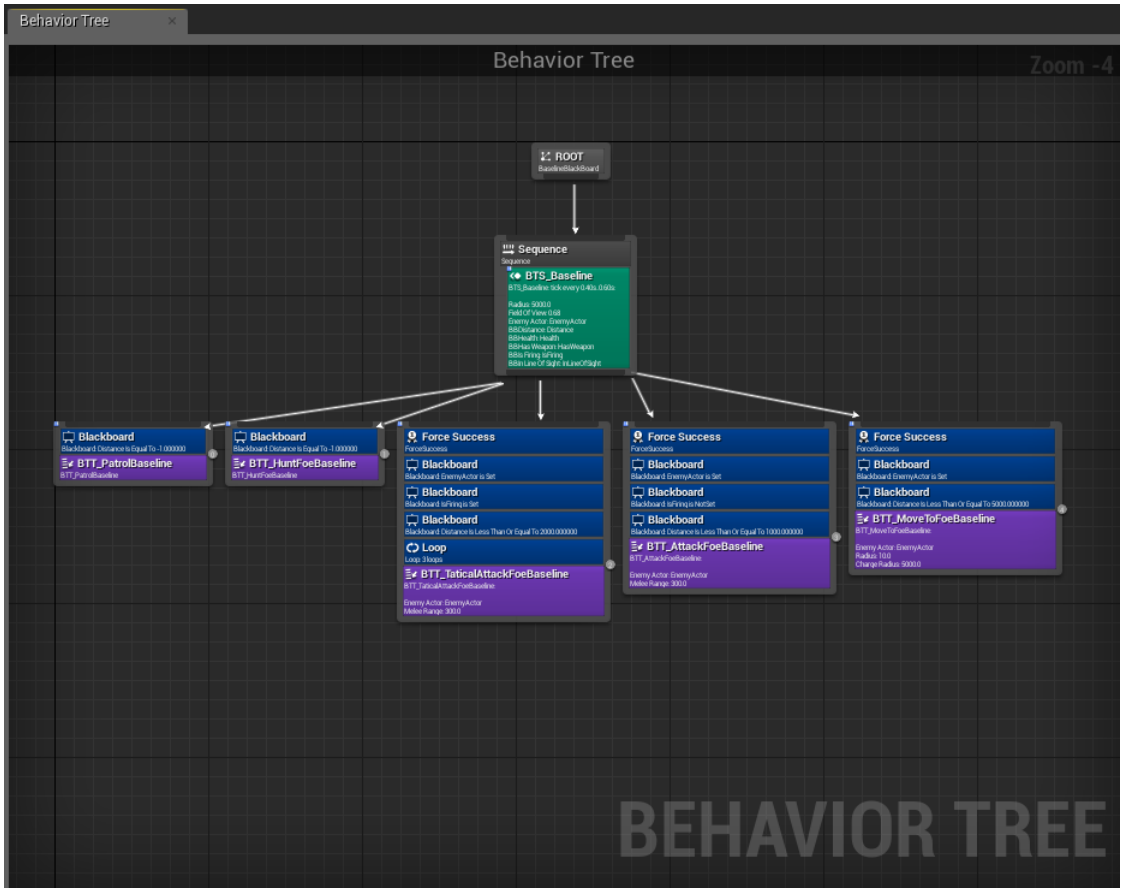


Figura 24 – Existem mais Decoradores, que são usados para controlar quais tarefas são executadas

disso, foi criado um *Blackboard* para compartilhar os sinais de saída da Rede Neural que foi executada no Serviço Neuroevolução. Cada tarefa é associada à um sinal de saída e executa, se e somente se, o sinal associado a ela for maior do que 0.5. Os decoradores são usados para evitar que a BT trave por conta de ponteiros nulos. No caso dos Bots *Linha de Base*, existem Decoradores adicionais que controlam quais tarefas devem ser executadas. Para isso existe um *blackboard* com as mesmas variáveis usadas como entrada nas redes neurais. A Figura 24 abaixo mostra a BT do BOT *Linha de Base*. Por fim, para que a sequência não seja interrompida, é adicionado à cada tarefa um decorador que força o retorno ser sucesso.

3.7.3 Módulo de Treinamento

O treinamento foi executado conforme descrito nos itens abaixo:

- **Configuração do NEAT**

  A taxa de aleatoriedade do algoritmo é dada pela taxa com que as mutações ocorrem. As mutações, descritas no Capítulo 2, são alterar peso da conexão, adicionar novo

nó e adicionar nova ligação. Foi configurado para que tais eventos acontecessem com 90, 70 e 70 porcento de probabilidade. Essa decisão visa causar mudanças mais abruptas nos indivíduos a fim de evitar efeitos como *crowding*.

- **Função *Fitness***

  A função fitness é dado pela soma da recompensa, do desempenho de dano e pelo desempenho em sobrevivência e é calculada ao final da partida. A função *fitness* é calculada pela Equação 3.1:

$$Fitness = Recompensa + \text{Desempenho de Dano} + \text{Desempenho de Sobrevivência} \tag{3.1}$$

  A recompensa é calculada pela Equação 3.2:

$$Recompensa = R * (\frac{\text{menor duração de partida}}{\text{duração da partida atual}})^R \tag{3.2}$$

  O valor de R é dado por:

$$R = \begin{cases} +1, & \text{se os jogadores forem eliminados} \\ -1 & \text{Caso Contrário} \end{cases} \tag{3.3}$$

  O desempenho de dano é calculado pela Equação 3.4:

$$\text{Desempenho de Dano} = \frac{\text{Dano causado}}{\sum_{i=1}^{n}(\text{pontos de vida máximo do jogador}_i)} \tag{3.4}$$

  Por fim, o desempenho de sobrevivência e dado pela Equação 3.5:

$$\text{Desempenho de Sobrevivência} = \frac{\text{(Pontos de vida restantes do BOT)}}{\text{(Pontos de vida máximo do BOT)}} \tag{3.5}$$

- **Rede Neural Inicial**

  A rede neural inicial possui cinco nós de entrada e cinco nós de saída conectados de maneira aleatória. Os valores de entrada escolhidos representam algumas da variáveis mais comuns que um jogador humano leva em consideração enquanto está jogando. A quantidade de valores de saída representa a quantidade de tarefas criadas para os BOTs desse protótipo.

Os valores alimentados na rede neural são: *distância até o inimigo alvo* (-1 caso não haja alvo), *pontos de vida restantes do BOT*, se o *inimigo possui uma arma*, se o *inimigo está atirando* e se *algum inimigo pode ver o BOT*. Os valores booleanos são convertidos para zero caso o valor seja falso e para um caso contrário. Como saída obtém-se um valor real entre 0 e 1 usado para determinar se uma tarefa deve ser executada ou não. As tarefas criadas serão descritas na próxima seção.

3.7.4 Configuração dos BOTs

A configuração dos BOTs Inimigos é realizada pela criação das árvores de comportamento conforme a 3.7.2. As tarefas criadas para o BOT Inimigo executar durante a partida são:

- **Itinerância:** O BOT escuta os passos dos jogadores até longas distâncias (até três kilometros), porém com alguns segundos de atraso (5 segundos). Em seguida se move até a origem do som;
- **Caçar:** O BOT escuta os passos de jogadores à média distância (até 400 metros) sem atrasos. Em seguida se move até a origem do som;
- **Ataque tático:** O BOT executa movimentos para os lados e pulos de maneira aleatória enquanto ataca o jogador;
- **Ataque:** O BOT realiza ataques, sem nenhuma instrução de movimento;
- **Avançar:** O BOT avança contra um jogador para que tenha distância suficiente para causar dano.

3.8 Configuração do Experimento

A fim de entender a percepção dos usuários, foi conduzido um experimento na forma de *survey*. Para isso, foi adotado um procedimento não probabilístico com a escolha aleatória de quinze voluntários que jogaram duas sessões de cinco minutos contra o BOT usando neuroevolução e o BOT usando a abordagem tradicional. A ordem com que os BOTs eram testados foi aleatória, dessa forma o indivíduo não sabia qual dos BOTs estava testando em cada sessão. Após os dez minutos, o jogo é interrompido e os voluntários preenchem um formulários respondendo em escala Likert, que é uma escala de concordância de 0 (discordo fortemente) à 4 (concordo fortemente):

- **Pergunta 1** - As habilidades de combate do bot parecem com a de um jogador humano

- **Pergunta 2** - As habilidades de esquiva do bot parecem com a de um jogador humano
- **Pergunta 3** - O movimento do bot parece com o de um jogador humano
- **Pergunta 4** - O comportamento do bot parece com o de um jogador humano
- **Pergunta 5** - O bot pareceu ser controlado por um jogador
- **Pergunta 6** - As habilidades de combate do bot eram imprevisíveis
- **Pergunta 7** - As habilidades de imprevisíveis me fizeram repensar minhas estratégias
- **Pergunta 8** - O movimento do bot é imprevisível
- **Pergunta 9** - As estratégias de combate do bot o fizeram um oponente interessante
- **Pergunta 10** - Gostei do jogo graças as habilidade de combate do bot
- **Pergunta 11** - Foi divertido jogar contra o bot
- **Pergunta 12** - A experiência de jogo geral foi agradável de modo geral
- **Pergunta 13** - O bot foi efetivo em todos os aspectos do combate
- **Pergunta 14** - Quão difícil o bot foi como oponente?
- **Pergunta 15** - Como você avalia as habilidades de combate do bot?

Percebe-se que as perguntas são traduções das que foram feitas no experimento (SONI; HINGSTON, 2008). As perguntas foram repetidas para os dois BOTs. Também foram feitas perguntas as seguintes perguntas comparando os dois BOTs:

- Qual dos bots foi mais divertido de jogar?
- Qual dos bots foi mais parecido com um jogador real?
- Contra qual dos bots você jogaria novamente?

Como dito na Seção 1.3 objetivo do projeto é desenvolver um BOT que aprenda, usando neuroevolução, a tomar decisões de qual tarefa executar uma das tarefas que acabaram de ser descritas. Entretanto, para avaliar o desempenho desta IA é necessário criar um segundo tipo de BOT que servirá de linha de base.

Ambos os BOTs, *Neuroevolução* e *Linha de Base*, terão acesso às mesmas informações. No caso do BOT *Linha de Base*, as decisão tomada será feita com condicionais

em sua árvore de comportamento. A condição de execução para o BOT *Neuroevolução* é dada pelos valores de saída da sua rede neural. Caso seja maior que 0.5, a tarefa será executada.

4 Resultados

No Capítulo 3 foram apresentadas as principais funcionalidades do jogo "*The Island*", assim como a função *fitness*, as tarefas e os dois tipos de BOT projetados e implementados. O objetivo do presente Capítulo é apresentar o protótipo implementado, ilustrar o seu funcionamento, principais funcionalidades, assim como os resultados obtidos com o treinamento dos BOTs utilizando o algoritmo neuroevolutivo e análise do comportamento desses BOTs a partir de alguns experimentos. O Capítulo é finalizado com uma análise crítica do algoritmo proposto.

4.1 Protótipo

O objetivo da seção é apresentar os resultados do protótipo desenvolvido e ilustrar suas principais funcionalidades.

4.1.1 BOT Jogador

O jogador é controlado por um BOT no modo treinamento, a fim de automatizar a etapa de aprendizado do algoritmo de neuroevolução. O comportamento do BOT jogador, no treinamento, não é modelado da mesma maneira que o BOT Inimigo. O comportamento do BOT Jogador é composto de duas tarefas, e não utiliza o algoritmo neuroevolutivo. A primeira é se movimentar em direção ao ponto de controle. O BOT está sempre se movendo em direção ao ponto de controle. A posição do ponto de controle é previamente conhecida. Portanto, a partir do momento que o BOT aparece no mapa ele ja começa a se mover. A Figura 25 mostra essa tarefa sendo executada.

A segunda tarefa do BOT Jogador é atirar contra o BOT Inimigo. Essa tarefa começa a ser executada quando o BOT Inimigo está no campo de visão do BOT Jogador. Vale lembrar que mesmo atirando, o BOT Jogador continua a se mover para o ponto de

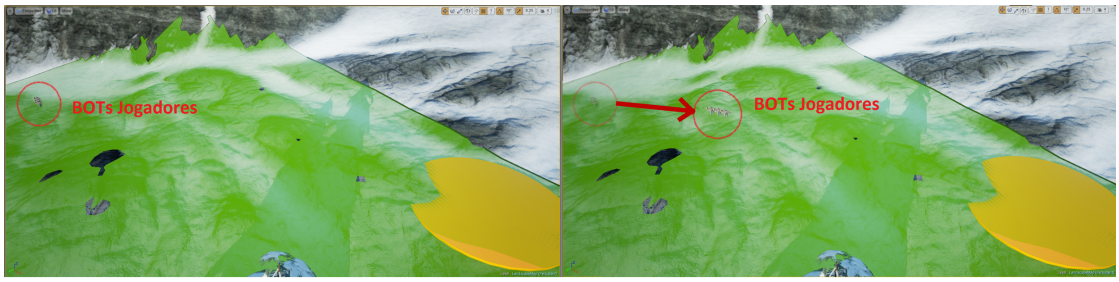


Figura 25 – BOT Jogador se movendo na direção do Ponto de controle

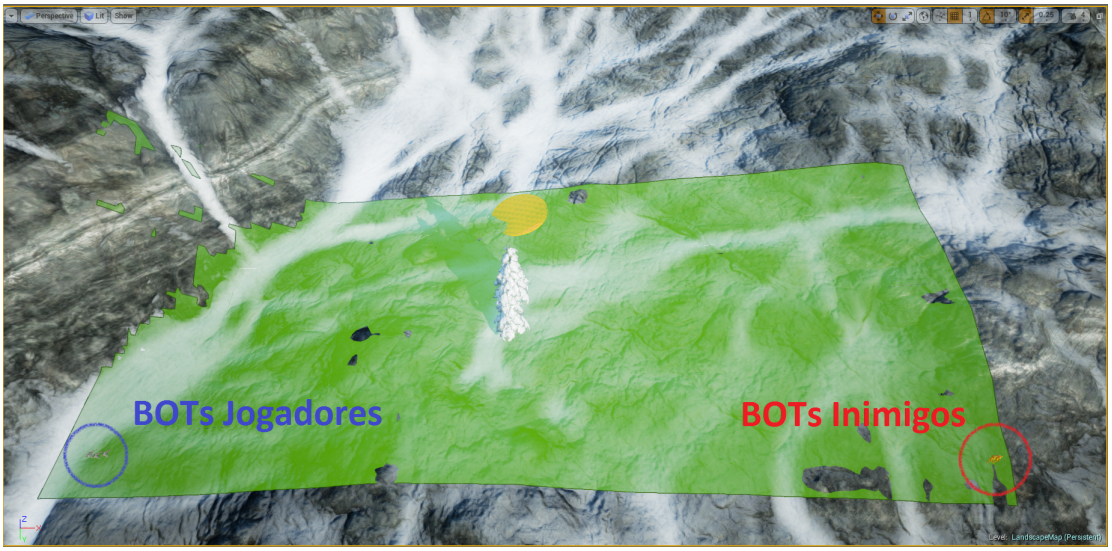
Figura 26 – BOT Jogador atirando contra BOTs inimigos

controle. A Figura 26 mostra essa tarefa.

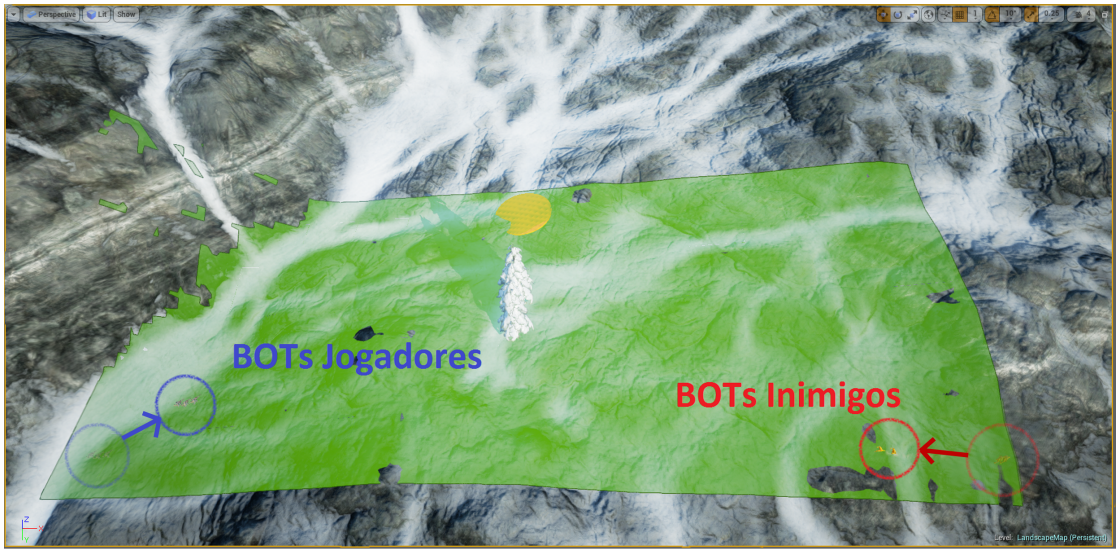
4.1.2 BOT Inimigo

O comportamento do BOT Inimigo é dado pelas cinco tarefas descritas na Seção 3.7.4. Os resultados da implementação dessas tarefas pode ser visto a seguir:

- **Itinerância:** A Figura 27a mostra o BOT Inimigo esperando receber a notificação de som à longas distâncias. Após receber a notificação, o BOT se move em direção à localização do som 27b.
- **Caçar:** A Figura 28 mostra o BOT Inimigo se movendo prontamente, pois não há atraso nos sons detectados a curta distância.
- **Ataque Tático:** A Figura 29a mostra o BOT Inimigo se movendo para os lados e realizando pulos, na Figura 29b, enquanto ataca. Os pontos vermelhos e laranjas ao redor do BOT indicam que ele está tentando se mover para aquele um deles para realizar a esquiva.
- **Ataque:** O BOT realiza ataques conforme pode ser visto na Figura 30. O bloco vermelho indica o volume onde o dano de cada ataque é realizado.
- **Avançar:** O BOT avança contra um jogador para que tenha distância suficiente para causar dano. Porém, a tarefa propriamente dita não inicia nenhum comando



(a) BOT esperando notificação (instante t = 0)



(b) BOT se movendo após notificação (instante t = 5)

Figura 27 – Resultado da Tarefa Itinerância

de ataque. Essa tarefa pode ser observada na Figura 34. Para avançar, o BOT Jogador deve estar visível e próximo.

4.1.3 Simulação da Partida

Finalmente, o último resultado em relação à implementação do protótipo é a simulação de uma partida. A partida começa com os BOTs aparecendo no mapa como pode ser visto na Figura 32.

Em seguida, pode-se observar que os BOTs movem-se em direção aos BOTs Jogadores na Figura 33a. Na Figura 33b é possível ver que, as saídas logo que o jogo começa, os BOTs inimigos estão executando a tarefa de Caçar, Ataque Tático, Atacar e Avançar.

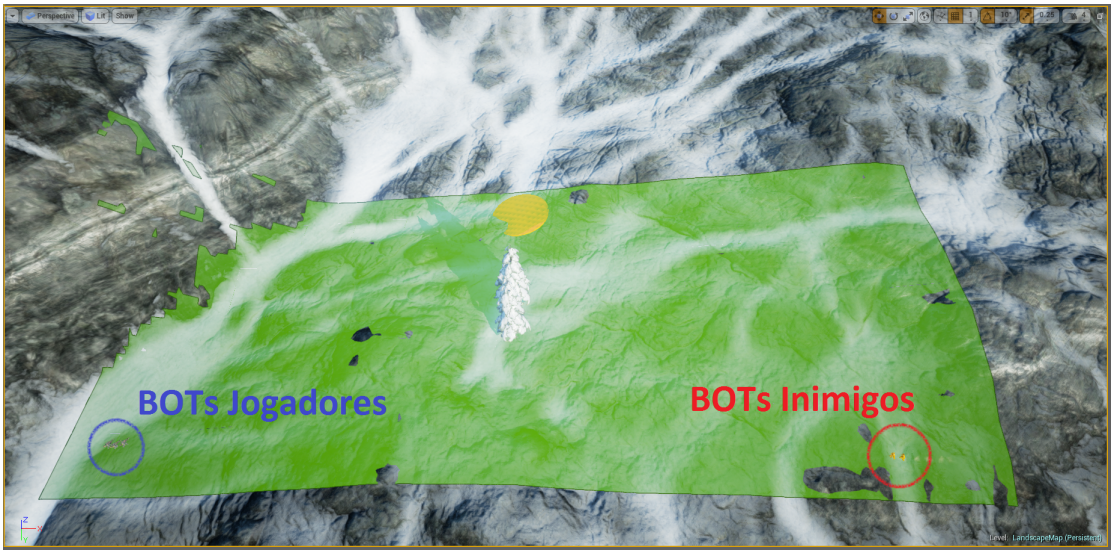


Figura 28 – Resultado da Tarefa Caçar (instante t = 0)

Isso faz sentindo pois Caçar é a mais indicada dada a distância que estão do jogador e as outras são executadas o tempo todo pois não há punição para isso. Após aproximar o suficiente, é possível ver o resultado na Figura 33c. Nela a tarefa de caça não é mais utilizada. Todas as tarefas sendo executadas foram marcadas de vermelho.

Por fim, a partida acaba quando todos os BOTs Jogadores morrem, conforme a Figura 34a, ou quando os sobreviventes chegam ao ponto de controle, conforme a Figura 34b.

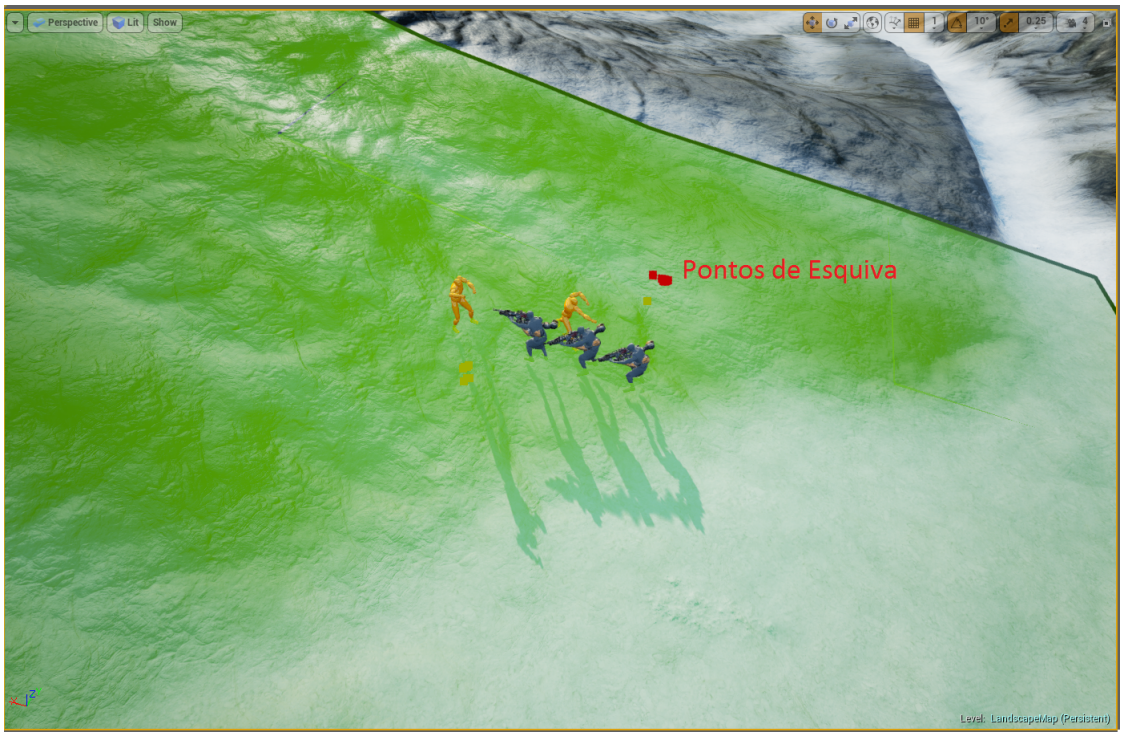
4.2 Da Técnica

A análise do ponto de vista técnico, dada as configurações feitas no desenvolvimento, avalia se algoritmo de neuroevolução foi capaz de encontrar um indivíduo de alta adequação. Além disso, são feitas observações a respeito desses resultados com luz nos conhecimentos do Capítulo 2.

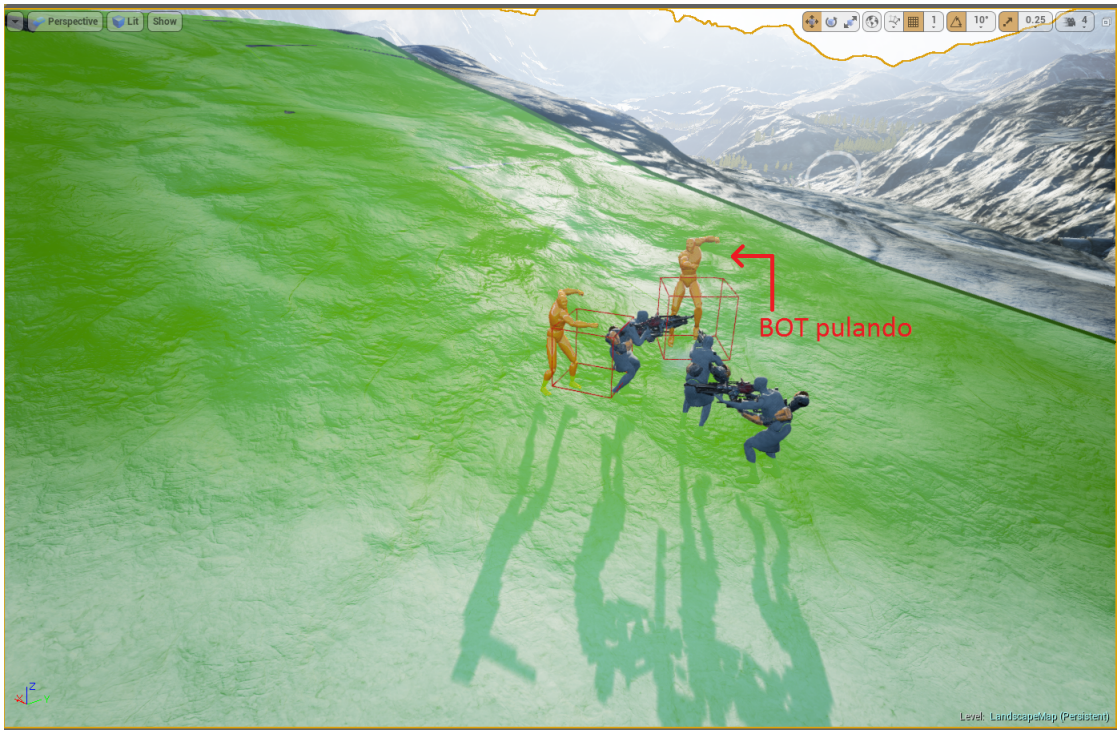
4.2.1 Resultados

Não existe conjunto de dados para avaliar a rede neural obtida. Isso se da pelo fato da natureza interativa e incerta do sistema em questão. Dessa forma, o valor obtido pela função *fitness* será o indicador de desempenho dos BOTs. Pode-se observar que o BOT *Neuroevolução* apresentou melhor desempenho do que a implementação tradicional do BOT *Linha de Base*. Os resultados são apresentadas a seguir na Tabela 5.

Conforme pode ser observado na tabela, o BOT *Neuroevolução* conseguiu um valor máximo mais alto de adequação do que a linha de base, o que mostra que um indivíduo de



(a) BOT Desviando ao se mover para um dos pontos de esquiva



(b) BOT Pulando

Figura 29 – Resultado da Tarefa Ataque Tático

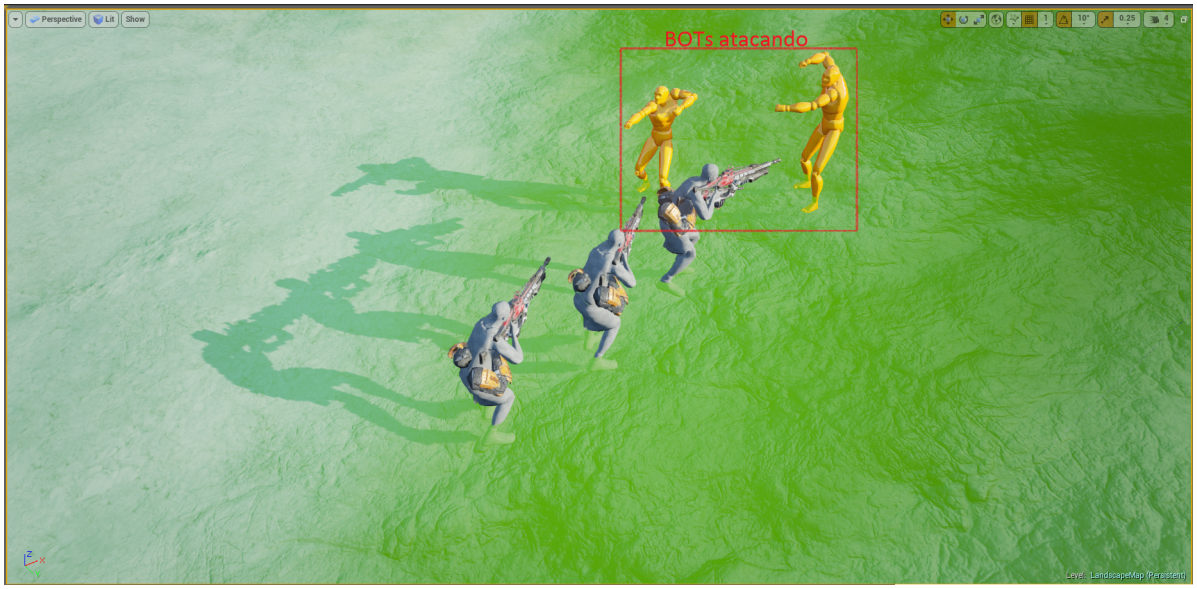


Figura 30 – Resultado da Tarefa Ataque

	Bot *Linha de Base*	**Bot *Neuroevolução***
Máximo	0.31953	0.333333
Média	0.06710824	0.12878084
Variância	0.066051652	0.043142036

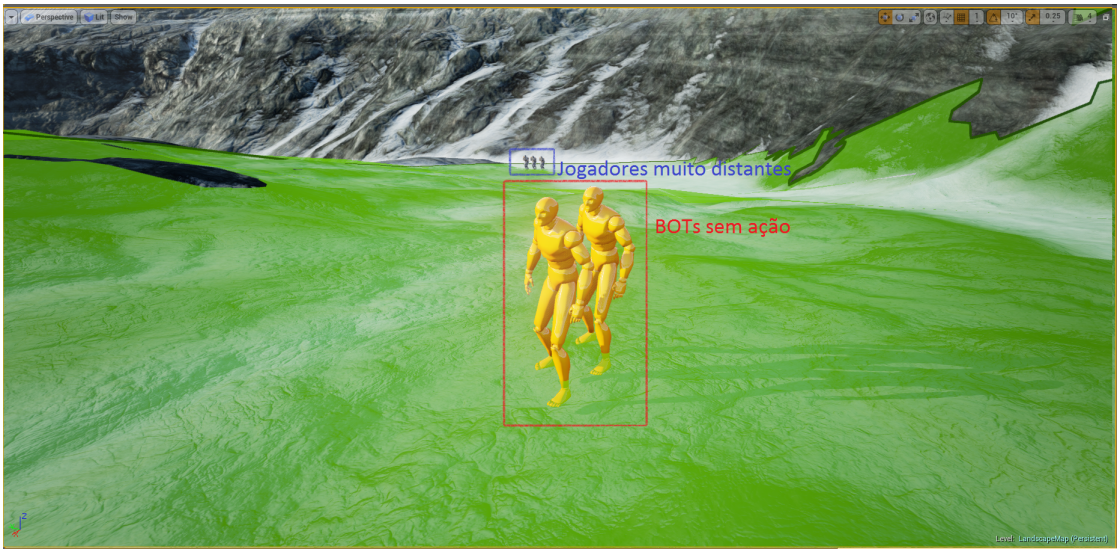
Tabela 5 – Comparando as *fitness* dos BOTs.

alta adequação foi encontrado. A análise da média mostra que, geralmente, um indivíduo do BOT da neuroevolução possui uma adequação maior do que um BOT da linha de base. Este valor foi quase duas vezes maior. Por fim, os resultados da variância mostram que houver menor dispersão nos resultados para o BOT *Neuroevolução* em relação a média.

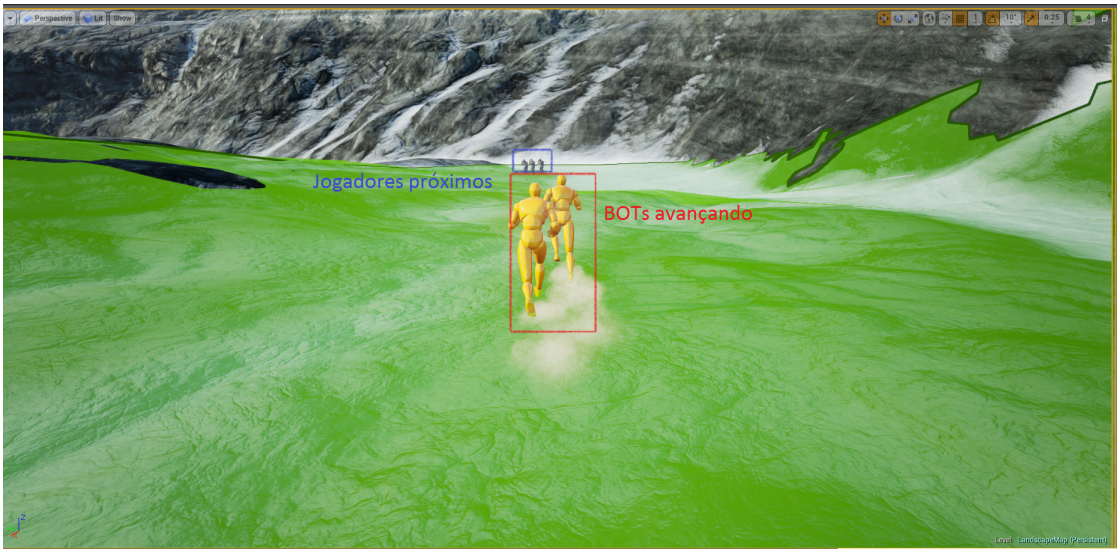
4.2.2 Avaliação dos Testes realizados

Um ponto importante é que o algoritmo encontrou o indivíduo com mais alta *fitness* na terceira geração, e algumas gerações seguintes conseguiram obter o mesmo valor. A rapidez com que o algoritmo convergiu indica a possibilidade de *crowding*. Portanto, ainda é possível obter indivíduos ainda mais adequados para a dada função de adequação. A taxa reduzida de reprodução, apenas um novo indivíduo por iteração, também pode ser uma das responsáveis pelo acontecimento desse fenômeno.

Do ponto de vista de complexidade espacial, após algumas horas de treinamento, o consumo de memória do algoritmo chegou a mais de $10GB$. Fazendo com que a máquina ficasse extremamente lenta e afetando o treinamento como um todo. Entretanto, como os resultados das avaliações eram salvos a medida que cada indivíduo era avaliado, o treino poderia ser interrompido e ainda sim haveriam registros do que houve até o momento da



(a) BOT sem alcance para Avançar contra o jogador



(b) BOT com alcance para Avançar

Figura 31 – BOT avançando contra um jogador próximo

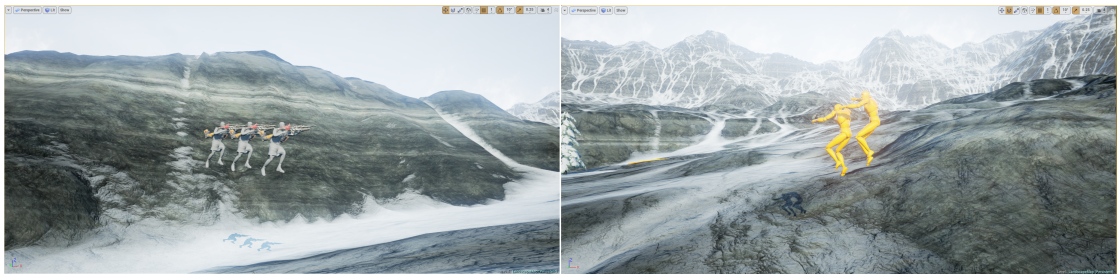
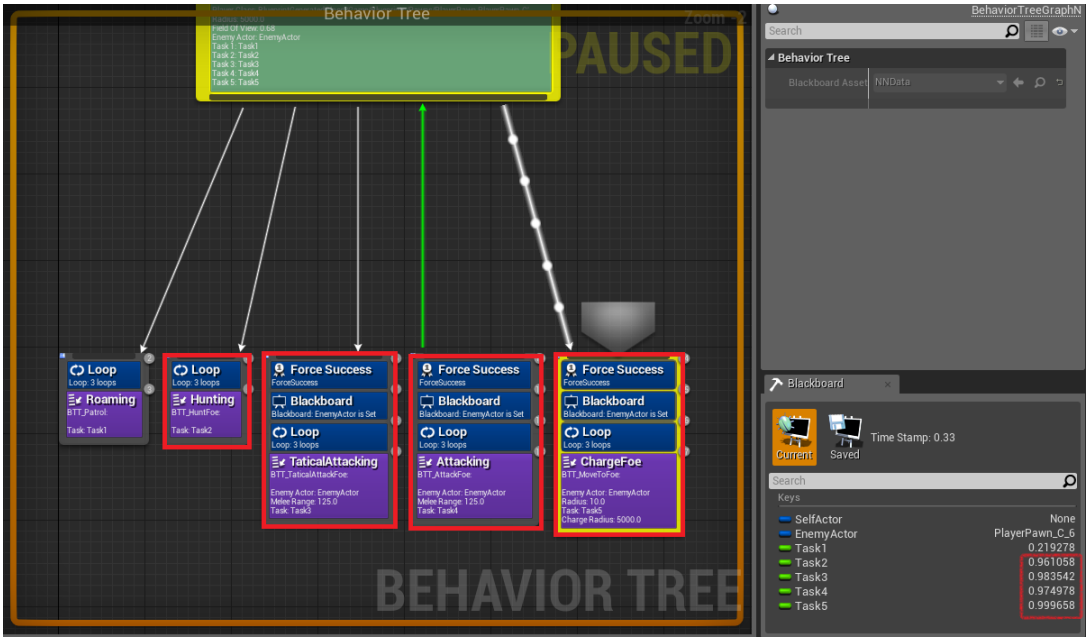
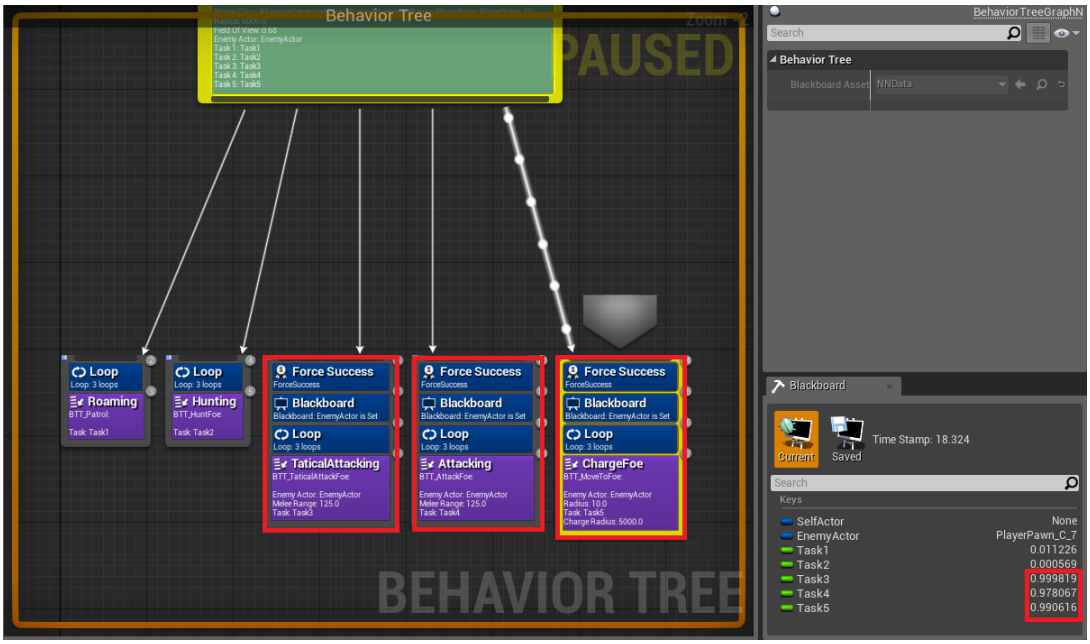


Figura 32 – BOTs aparecendo no mapa

(a) BOT tomando decisões durante uma partida



(b) Saídas executando Caçar, Ataque Tático, Atacar e Avançar



(c) Saídas executando Ataque Tático, Atacar e Avançar

Figura 33 – Execução da Simulação

(a) Último Bot Jogador Morrendo

(b) Bots Jogadores chegando ao ponto de controle

Figura 34 – Fim da simulação

interrupção. Isto levanta indícios de que houve algum erro na implementação ou utilização das ferramentas.

Outro ponto importante, é que os BOTs foram treinados contra BOTs jogadores com uma IA tradicional. Uma oportunidade interessante de melhoria seria realizar o treinamento com os BOTS, *Neuroevolução* e *Jogadores*, usando neuroevolução, realizando um *treinamento competitivo*.

Entretanto, vale lembrar que, como qualquer optimização aleatória, o resultado continua sendo probabilístico. Isto é, não é garantido que o valor encontrado seja o melhor possível. De qualquer maneira, o algoritmo fez seu papel de encontrar um valor de alta adequação maior do que o valor do algoritmo da linha de base.

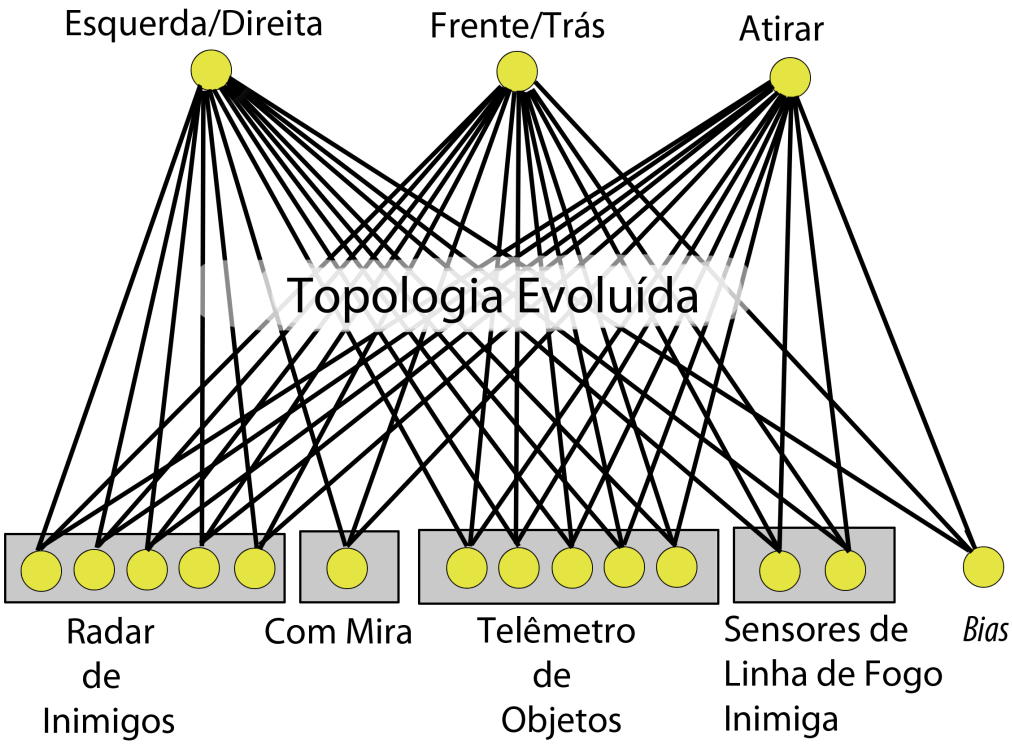


Figura 35 – Estrutura da Rede Neural por (STANLEY; BRYANT; MIIKKULAINEN, 2005a)

4.2.3 Comparando com a Referência Bibliográfica

Por fim, é interessante fazer uma comparação entre os resultados do uso de ações, como em (STANLEY; BRYANT; MIIKKULAINEN, 2005a), e os resultados deste trabalho com o uso de tarefas. O resultado do trabalho de (STANLEY; BRYANT; MIIKKULAINEN, 2005a) foi o jogo *NeuroEvolvingo Robotic Operatives*, ou *NERO*, onde as redes neurais são treinadas em tempo real e assíncrona, ou seja, enquanto o jogo acontece e toda população simultaneamente. Durante o treino, as unidades são treinadas para realizar apenas uma tarefa e a função de adequação é controlada por um humano. As estrutura da rede neural utilizada é apresentada na Figura 35.

A figura acima mostra que as saídas da rede neural são ações de movimentar para frente ou para trás, para esquerda ou para direita e se o BOT está atirando ou não. Tais ações possuem baixo nível de abstração o que permite aos BOTs apresentarem comportamentos mais imprevisíveis e podem gerar soluções criativas para dadas situações. Por exemplo, um resultados obtidos pelo autor ao treinar as unidades com o objetivo de se aproximar de um atirador inimigo e de minimizar o dano recebido foram unidades que flanqueiam o inimigo. A Figura 36 mostra este resultado obtido pelo autor.

Em contrapartida, essas redes neurais não são capazes de gerar resultados robustos

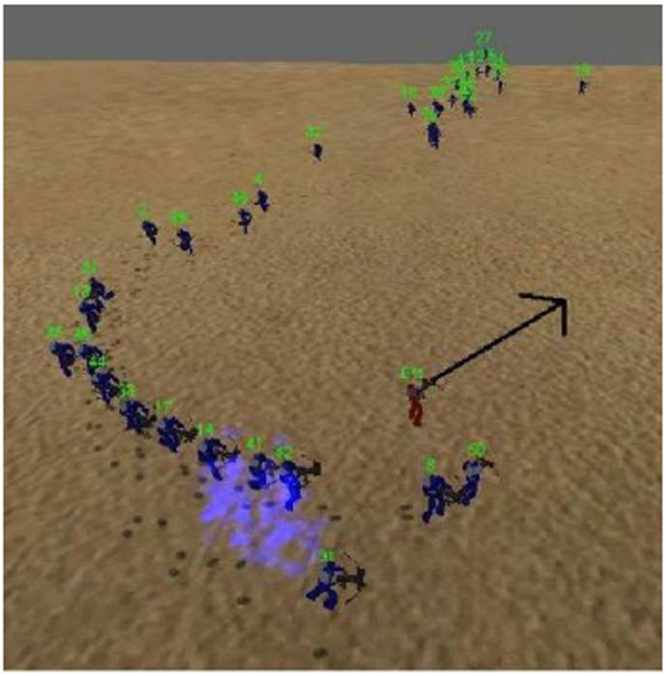


Figura 36 – BOTs Flanqueando uma Unidade Inimiga por (STANLEY; BRYANT; MIIKKULAINEN, 2005a)

com frequência. O autor mostra, por exemplo, que para os BOTs aprenderem a navegar por labirintos, eles precisam navegar por labirintos que vão se tornando gradativamente mais complexos. A Figura 37 mostra o resultado obtido por (STANLEY; BRYANT; MIIKKULAINEN, 2005a) para BOTs que podem percorrer labirintos.

O problema com essa abordagem é que uma tarefa que poderia ter sido implementada manualmente como um conjunto de ações teve de ser aprendida. Dessa forma, os BOTs gerados possuem baixa qualidade do ponto de vista do jogador e limitam a participação dos desenvolvedores no projeto do comportamento desses BOTs. Dito isto, pode-se comparar que os resultados obtidos pelos BOTs do protótipo desenvolvido neste trabalho. A Tabela 6 mostra os principais pontos da comparação.

A tabela acima mostra que, apesar dos dois trabalhos terem gerados BOTs que são capazes de aprender ao interagir com jogadores, o protótipo deste trabalho visou criar BOTs que precisavam de uma maneira para acelerar o aprendizado. Além disso, é interessante que desenvolvedores possam moldar e validar o comportamento das unidades controladas pelo computador, de modo a gerar BOTs de alta qualidade. Portanto, o uso de ações mais abstratas, a utilização de tarefas foi eficientes ao acelerar o aprendizado por gerar soluções robustas e inserir conhecimento prévio do domínio no sistema de IA desenvolvido.

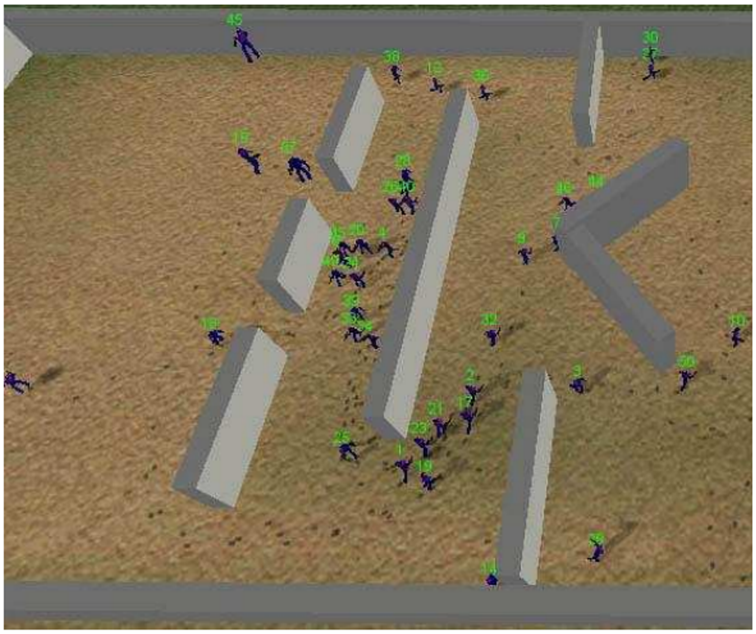


Figura 37 – BOTs percorrendo um labirinto por (STANLEY; BRYANT; MIIKKULAINEN, 2005a)

	NERO	***The Island***
Treino	Tempo real e assíncrono	Final de cada partida e síncrono
Saídas da rede neural	Ações (baixa abstração)	Tarefas (ações com alta abstração)
Objetivo do treino	Aprender a realizar uma tarefa	Aprender a ser um BOT de alta qualidade
Criatividade	Pode ser alta	Depende das tarefas implementadas
Imprevisibilidade	Alta	Alta
Desafio	Demora aprender soluções robutas	A robustez está nas tarefas previamente implementadas
O BOT pode ser projetado	Cada rede neural executa apenas uma tarefa	Equipe projeta o comportamento por meio das tarefas

Tabela 6 – Comparando as abordagens.

4.3 Da Percepção do Usuário

As seções anteriores trataram de apresentar a validade da solução ao validar os resultados da análise técnica e da análise comparativa com a referência bibliográfica. Porém, um dos pontos mais estressados ao longo deste trabalho, é de que os BOTs deveriam ter alta qualidade. Como apresentado no Capítulo 2, essa qualidade é baseada na percepção do usuário. Dessa forma, foi conduzido um experimento, conforme detalhado na Seção 3.8, para avaliar a percepação dos usuários, também chamado de *Play Test* neste contexto.

4.3.1 Resultados dos Teste

As perguntas dos testes foram dividas em quatro categorias. A primeira Percepção de "Humanindade" agrupou as perguntas de 1 à 5. A segunda categoria foi a Previsibilidade, com as perguntas de 6 a 8. Na categoria Entretenimento foram feitas as perguntas 9 à 12. Por fim, a categoria Desafio continha as perguntas 13 à 15. Para cada categoria foi feita uma análise da média de concordância para entender de modo geral o desemepnho dos BOTs na dada categoria. Além disso, foi feita uma análise de variância entre as perguntas para avaliar se as diferenças observadas são significativas. Para isso, o valor *F(k,l)* calculado deve ser menor do que o ponto crítico, que pode ser obtido consultando uma tabela da Distribuição F usando os valores k e l. Os items a seguir mostram os resultados obtidos.

- **Percepção de "Humanindade"** O experimento mostra que, no geral a percepção do usuário é de que o BOT neuroevolução foi menos parecido com um humano. Além disso, o teste de variância indica que existem diferenças entre os BOTs, dado que $F(2,15) = 1.7819, p = 0.05$ para a pergunta 2, $F(2,15) = 2.3145, p = 0.05$ para a pergunta 3 e $F(2,15) = 1.0860, p = 0.05$ para a pergunta 5. O Figura 38 mostra os resultados obtidos.

- **Previsibilidade** Quanto à previsibilidade, o BOT que usou a neuroevolução foi capaz de obter resultados significativamente abaixo dos obtidos na linha de base. O teste de variância para as três perguntas feitas também indicam que essa distinção é perceptível pois $F(2,15) = 2.2075, p = 0.05$ para a pergunta 6, $F(2,15) = 1.2617, p = 0.05$ para a pergunta 7 e $F(2,15) = 1.7638, p = 0.05$ para a pergunta 8. A Figura 39 mostra esses resultados visualmente.

- **Entretenimento** Do ponto de vista do entretenimento gerado, o BOT desenvolvido neste trabalho também não foi melhor avaliado. As perguntas 10 e 11 mostram que houve percepção da diferença entre os BOTs, com $F(2,15) = 2.2075, p = 0.05$ e $F(2,15) = 2.2075, p = 0.05$, respectivamente. A Figura 40 apresenta os resultados para as perguntas desta categoria.

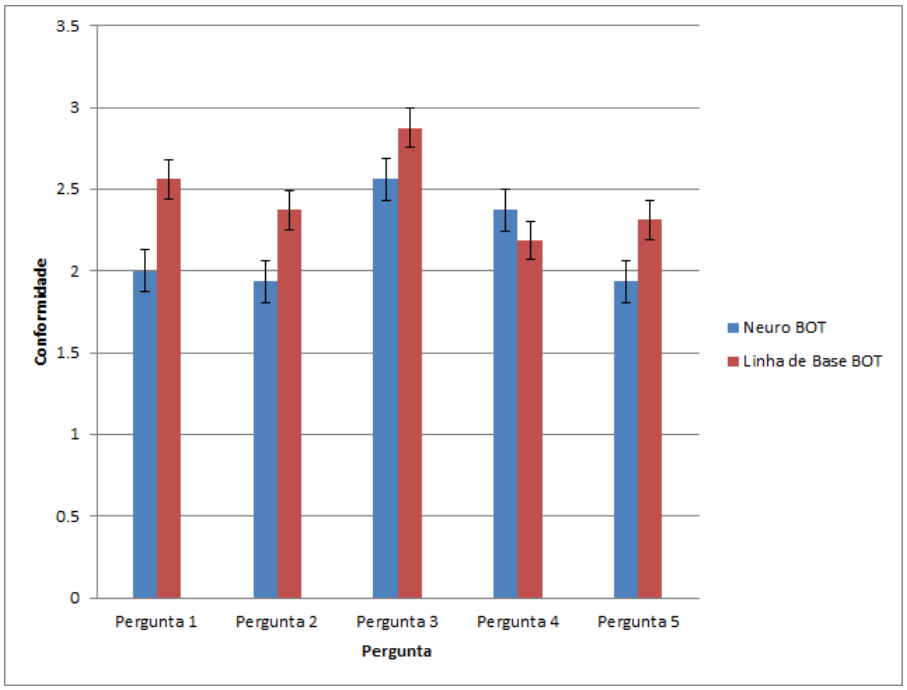


Figura 38 – Resultados perguntas sobre Percepção de “Humanindade”

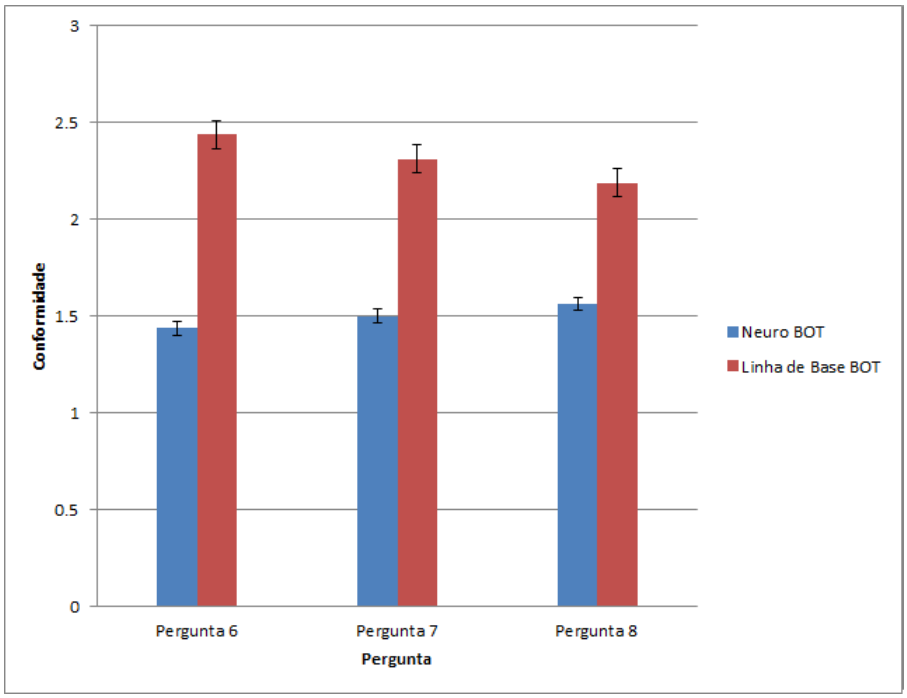


Figura 39 – Resultados perguntas sobre Previsibilidade

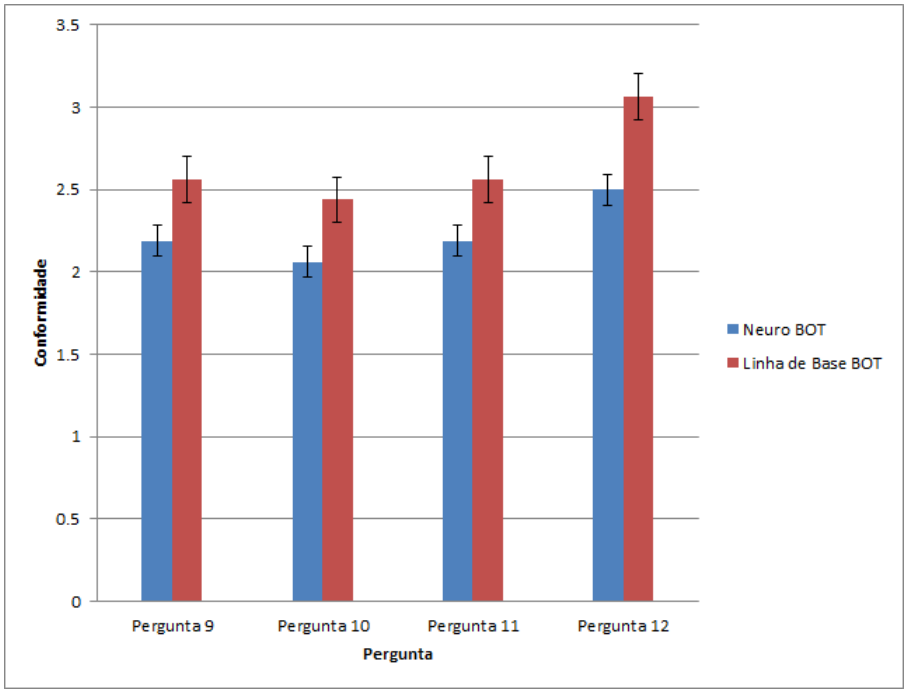


Figura 40 – Resultados perguntas sobre Entretenimento

Mais Rejogável	Mais divertido	Mais Humano
Linha de Base BOT 0.25	0.25	0.3125
Neuroevolução BOT 0.75	0.75	0.6875

Tabela 7 – Resultados da Comparação

- **Desafio** No que diz respeito ao desafio proporcionado, o BOT neuroevolução foi melhor avaliado. Em todas as perguntas feitas nessa categoria o teste de variância indicam que houve percepção de diferença. Os valores obtidos nos teste de variância para as perguntas 13, 14 e 15 foram $F(2,15) = 1.2651, p = 0.05$, $F(2,15) = 1.4371, p = 0.05$ e $F(2,15) = 1.4858, p = 0.05$, respectivamente. A Figura 41 mostra os resultados obtidos.

- **Comparação** Por fim, os resultados das perguntas que comparam os BOTs foi de que o BOT usando neuroevolução é mais divertido, mais parecido com um humano e mais rejogável. A Tabela 7 a seguir mostra a proporção de votos que cada uma das perguntas de comparação.

Esses dados mostram que, apesar dos BOT ter tido um desempenho pior em muitas categorias, de modo geral ele foi considerado um BOT de maior qualidade. Isto indica que a qualidade do experimento conduzido foi baixa. Prováveis explicações para isto foram

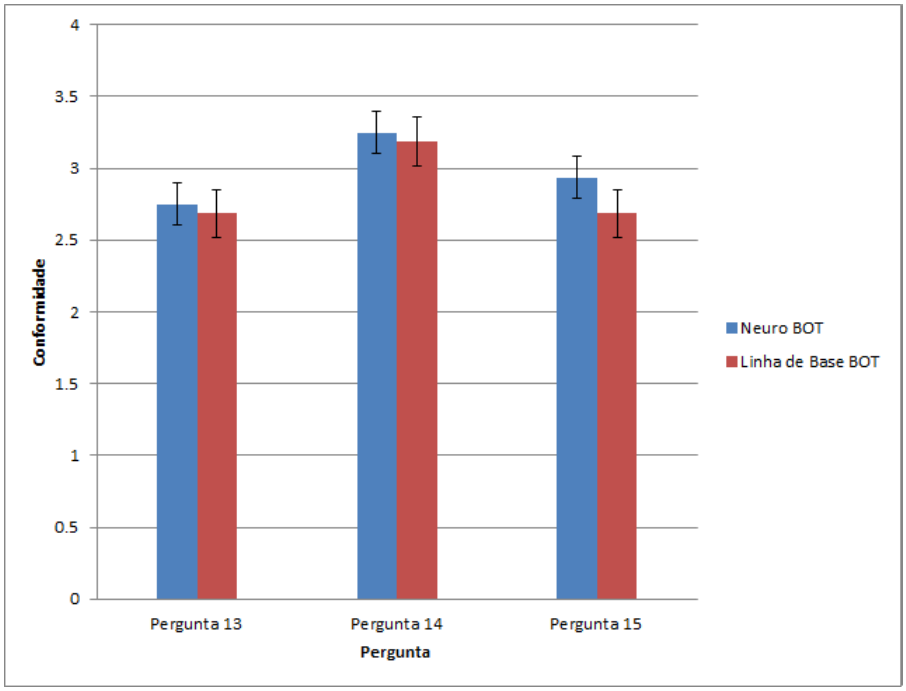


Figura 41 – Resultados perguntas sobre Desafio

as perguntas não estarem claras e falhas na condução do experimento. Entretanto, os resultados da comparação final deixam evidências de que os usuários perceberam o BOT usando neuroevolução atingiu os objetivos estabelecidos para o protótipo.

Por fim, este capítulo apresentou os resultados obtidos no desenvolvimento do protótipo assim como a validação feita, tanto do ponto de vista teórico quanto do ponto de vista do usuário. O próximo capítulo traz as considerações finais do trabalho.

5 Conclusão

Este trabalho buscou construir uma IA híbrida, ou seja, com elementos não-determinísticos e determinísticos. O elemento não determínistico está no uso da neuroevolução. O elemento determínistico é incluído nas tarefas executadas, que representam conhecimentos sobre o domínio. Isto foi feito por meio do uso da neuroevolução e da criação de tarefas prontas, que representam conhecimento sobre o domínio. O objetivo estipulado era o obter BOTs de alta qualidade.

Os BOTs farão parte do jogo "*The Island*". Entretanto, não é viável implementar o jogo inteiro. Na realidade, sem alguma garantia de que a IA pode gerar os resultados esperados, a proposta do jogo perde o sentido. Dessa forma, foi desenvolvido um protótipo que pudesse permitir a validação técnica da proposta de IA. Além disso, a condução da *survey* possibilitou um melhor entendimento da percepção dos usuários a respeito do uso dessa abordagem.

O uso da neuroevolução com tarefas de alto nível mostrou-se capaz de gerar BOTs de alta qualidade. Tanto do ponto de vista técnico, BOTs com alta adequação, quando do ponto de vista do usuário, BOTs são mais divertidos, imprevisíveis, habilidosos e semelhantes a jogadores humanos. Do ponto de vista técnico, o Capítulo 4, mostrou que mesmo com resultados satisfatórios, ainda existem oportunidades de melhoria. Portanto, a necessidade de otimizações no código e uso de hardware dedicado para a tarefa de treino serão interessantes para etapas posteriores do desenvolvimento do jogo.

Esses resultados abrem caminho para a continuação do desenvolvimento do jogo, por meio de novos protótipos e experimentos com o gameplay. Isso pois, mesmo com os bons resultados, ainda existem oportunidades de melhoria para tornar o sistema mais robusto. Exemplos de oportunidades de melhorias são a implementação de uma função *fitness* que capture mais informação da partida, a implementação de novas tarefas e a criação de um sistema de gerenciamento online das IA, que irá realizar a neuroevolução usando como iteração partidas contra jogadores humanos em ambiente de produção.

Referências

BLENDER. *FBX*. 2014. Disponível em: <http://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Import-Export/Autodesk_FBX>. Citado na página 47.

BOURG, D. M.; SEEMANN, G. *AI for Game Developers*. [S.l.]: O'Reilly Media, Inc., 2004. ISBN 0596005555. Citado 6 vezes nas páginas 13, 21, 31, 32, 37 e 38.

EBERHART, R. C. *Computational Intelligence: Concepts to Implementations*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN 1558607595, 9780080553832. Citado 4 vezes nas páginas 27, 28, 31 e 32.

ENGELBRECHT, A. P. *Computational Intelligence: An Introduction*. 2nd. ed. [S.l.]: Wiley Publishing, 2007. ISBN 0470035617. Citado 7 vezes nas páginas 27, 28, 29, 30, 31, 32 e 33.

ENGINE, U. *Actor and Variable Replication*. 2014. Disponível em: <https://www.youtube.com/watch?v=ZsDRDAdtA9w&index=2&list=PLZlv_N0_O1gYwhBTjNLSFPRiBpwe5sTwc>. Citado na página 58.

ENGINE, U. *Animation Blueprint EventGraph*. 2014. Disponível em: <https://www.youtube.com/watch?v=DLJ0VEPWeV4&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb&index=11>. Citado na página 58.

ENGINE, U. *Behavior Tree Tutorial*. 2014. Disponível em: <https://docs.unrealengine.com/latest/INT/Engine/AI/BehaviorTrees/HowUE4BehaviorTreesDiffer/index.html>. Citado na página 59.

ENGINE, U. *Blend Spaces*. 2014. Disponível em: <https://www.youtube.com/watch?v=pfKdr0FRC5g&index=7&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

ENGINE, U. *Blueprint Function Library, Create Your Own to Share With Others*. 2014. Disponível em: <https://wiki.unrealengine.com/Blueprint_Function_Library,_Create_Your_Own_to_Share_With_Others>. Citado na página 58.

ENGINE, U. *Blueprints, Creating C++ Functions as new Blueprint Nodes*. 2014. Disponível em: <https://wiki.unrealengine.com/Blueprints,_Creating_C%2B%2B_Functions_as_new_Blueprint_Nodes>. Citado na página 58.

ENGINE, U. *Building the AnimGraph*. 2014. Disponível em: <https://www.youtube.com/watch?v=ZezNr-DOSRI&index=10&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

ENGINE, U. *Character Blueprint Components*. 2014. Disponível em: <https://www.youtube.com/watch?v=_Y8ZOLGZQRM&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb&index=12>. Citado na página 58.

ENGINE, U. *Creating Animation Notifies in UE4*. 2014. Disponível em: <https://www.youtube.com/watch?v=LpgvyhVcxM4&index=22&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

ENGINE, U. *Creating the Base Pickup Class*. 2014. Disponível em: <https://www.youtube.com/watch?v=ZPn4N3ckP8I&list=PLZlv_N0_O1gb5xvsc7VM7pfoRAKLuIcFi&index=3>. Citado na página 58.

ENGINE, U. *FBX Importing and Using Skeletons*. 2014. Disponível em: <https://www.youtube.com/watch?v=3RqciG2VoI4&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb&index=3>. Citado na página 58.

ENGINE, U. *First Person Shooter Sample*. 2014. Disponível em: <https://docs.unrealengine.com/latest/INT/Resources/SampleGames/ShooterGame/index.html>. Citado na página 47.

ENGINE, U. *Function Replication*. 2014. Disponível em: <https://www.youtube.com/watch?v=wutqpyDKnCQ&index=3&list=PLZlv_N0_O1gYwhBTjNLSFPRiBpwe5sTwc>. Citado na página 58.

ENGINE, U. *Getting Started: Introduction to UE4 Level Creation*. 2014. Disponível em: <https://www.youtube.com/playlist?list=PLZlv_N0_O1gak1_FoAJVrEGiLIploeF3F>. Citado na página 58.

ENGINE, U. *Intro to Animation Blueprints*. 2014. Disponível em: <https://www.youtube.com/watch?v=AqYmC2wn7Cg&index=8&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

ENGINE, U. *Intro to Animation Montage in UE4*. 2014. Disponível em: <https://www.youtube.com/watch?v=zQrNQtfNOHc&index=16&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

ENGINE, U. *Intro to Persona*. 2014. Disponível em: <https://www.youtube.com/watch?v=8Y7Lr4PUOuo&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb&index=4>. Citado na página 58.

ENGINE, U. *Network Relevancy*. 2014. Disponível em: <https://www.youtube.com/watch?v=6w8_eC5qlSs&index=4&list=PLZlv_N0_O1gYwhBTjNLSFPRiBpwe5sTwc>. Citado na página 58.

ENGINE, U. *Project Creation*. 2014. Disponível em: <https://www.youtube.com/watch?v=nzq7hGmxLP8&list=PLZlv_N0_O1gb5xvsc7VM7pfoRAKLuIcFi&index=2>. Citado na página 58.

ENGINE, U. *Setting Up Inputs*. 2014. Disponível em: <https://www.youtube.com/watch?v=WARth7AjFU0&index=5&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

ENGINE, U. *Skeleton Retargeting and Montage Setup in UE4*. 2014. Disponível em: <https://www.youtube.com/watch?v=S8ardXXy5pE&index=17&list=PLZlv_N0_O1gZS5HylO_368myr-Kg2ZLwb>. Citado na página 58.

FREITAS, L. G. de et al. Gear2d: an extensible component-based game engine. In: *Proceedings of the International Conference on the Foundations of Digital Games*. New York, NY, USA: ACM, 2012. (FDG '12), p. 81–88. ISBN 978-1-4503-1333-9. Disponível em: <http://doi.acm.org/10.1145/2282338.2282357>. Citado na página 21.

KONAR, A. *Computational Intelligence: Principles, Techniques and Applications*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005. ISBN 3540208984. Citado 8 vezes nas páginas 13, 27, 28, 29, 30, 31, 32 e 33.

MILLINGTON, I.; FUNGE, J. *Artificial Intelligence for Games, Second Edition*. 2nd. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2009. ISBN 0123747317, 9780123747310. Citado 2 vezes nas páginas 25 e 27.

SCHRUM, J.; MIIKKULAINEN, R. Evolving multimodal networks for multitask games. *IEEE Transactions on Computational Intelligence and AI in Games*, IEEE, v. 4, n. 2, p. 94–111, June 2012. Disponível em: <http://nn.cs.utexas.edu/?schrum:tciaig12>. Citado na página 33.

SCHWAB, B. *Ai Game Engine Programming (Game Development Series)*. Rockland, MA, USA: Charles River Media, Inc., 2004. ISBN 1584503440. Citado na página 22.

SONI, B.; HINGSTON, P. Bots trained to play like a human are more fun. In: *IJCNN*. IEEE, 2008. p. 363–369. Disponível em: <http://dblp.uni-trier.de/db/conf/ijcnn/ijcnn2008.html#SoniH08>. Citado 5 vezes nas páginas 21, 38, 39, 43 e 64.

STANLEY, K. O.; BRYANT, B. D.; MIIKKULAINEN, R. Evolving neural network agents in the nero video game. In: *Proceedings of the IEEE 2005 Symposium on Computational Intelligence and Games (CIG'05)*. Piscataway, NJ: IEEE, 2005. Disponível em: <http://nn.cs.utexas.edu/?stanley:cig05>. Citado 5 vezes nas páginas 14, 42, 76, 77 e 78.

STANLEY, K. O.; BRYANT, B. D.; MIIKKULAINEN, R. Real-time neuroevolution in the nero video game. *IEEE Transactions on Evolutionary Computation*, IEEE, p. 653–668, 2005. Disponível em: <http://nn.cs.utexas.edu/?stanley:ieeetec05>. Citado na página 56.

STANLEY, K. O.; MIIKKULAINEN, R. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, v. 10, n. 2, p. 99–127, 2002. Disponível em: <http://nn.cs.utexas.edu/?stanley:ec02>. Citado 5 vezes nas páginas 13, 33, 34, 36 e 38.

SWEENEY, T. *Unreal Engine 4*. [S.l.]: Epic Games, 2014. <https://www.unrealengine.com>. Citado na página 56.

TOZOUR, P. The Evolution of Game AI. In: RABIN, S. (Ed.). *AI game programming wisdom*. [S.l.]: Charles River Media, 2002. p. 3–15. Citado na página 25.

Apêndices

APÊNDICE A – Documento de Design do Jogo - V1.0.0

A.1 Nome do Jogo

The Island

A.2 Gênero do Jogo

Tiro em Primeira Pessoa e *Survival Horror*

A.3 Plataforma

PC

A.4 Descrição

O jogo se passa em tempos atuais onde o jogador assume o papel de um membro de um grupo de sobreviventes que buscam uma maneira de escapar da ilha. Para isso eles precisam encontrar algum veículo em condições para tirá-los de lá ou descobrir os mistérios dessa ilha.

A.5 Premissa

A.5.1 Tempo

Indeterminado.

A.5.2 Lugar

Uma ilha fictícia, provavelmente no Triângulo das Bermudas.

A.5.3 Personagens

Os Sobreviventes e o Enxame.

A.5.4 Motivações

Sobrevivência (Sobreviventes). Propagar a infestação alienígena (Enxame).

A.5.5 Relacionamentos

Os sobreviventes tentam escapar da ilha enquanto são caçados pelas unidades do Enxame.

A.5.6 Objetivo

Sobreviver e escapar da ilha.

A.5.7 Meta

- Encontrar um veículo e o que mais for necessário para que o mesmo esteja em condições de tirar os sobreviventes da ilha.
- Minimizar a quantidade de recursos consumidos (rem[edios, comida, etc).
- Sempre que possível, impedir o crescimento do Enxame.

A.6 Ambientação

Ficção Científica (Aliens).

A.7 Regras do jogo

A.7.1 Pontos chave

A ilha contará com diversos pontos chaves sendo eles: as Colmeias do Enxame, onde os Colmeeiros são criados em intervalos de tempo regulares, os corpos de vítimas feitas na ilha, onde são encontrados recursos que podem ser usados pelos sobreviventes e as construções Humanas, onde os sobreviventes tem mais chances de encontrar recursos.

A.7.2 Posicionamento das Estruturas

O posicionamento das estruturas acontecerá de maneira aleatória.

A.7.3 Mapa

Os elementos da ilha serão dispostos de maneira aleatória.

A.8 Unidades

A.8.1 Os sobreviventes

Os sobreviventes não apresentam nenhuma distinção. Podem usar itens do ambiente tais como medicamentos, comida e armas.

A.8.2 A Infestação alienígena

As unidades alienígenas são geradas a partir das colmeias em intervalos de tempo regulares. Serão controladas por redes neurais usando neuroevolução. Os objetivos dessas unidades são caçar os sobreviventes e proteger a colmeia.

A.9 Interface

A.9.1 Input

O jogador usará mouse e teclado.

A.10 Audiência

A.10.1 Idade

ADOLESCENTE/TEEN - O conteúdo é geralmente adequado para as idades igual ou superior a 13 anos. Pode conter violência, temas sugestivos, humor negro, sangue mínimo, apostas simuladas, e/ou uso infreqüente de linguagem forte.

A.10.2 Gênero

Ambos.

A.10.3 Renda

Qualquer renda.

A.10.4 Perfil

As dificuldades e possibilidades do jogo irão agradar tanto aos jogadores de estilo casual quanto aos jogadores *hardcore*.

A.11 Extras

A.11.1 Game Play

Durante o gameplay o jogador estará em um ilha desconhecida. Itens poderão ser encontrados para auxiliar os jogadores durante a partida. Os itens encontrados podem ser itens para recuperar saúde (comida e medicamentos), facilitar a exploração (lanternas, mapas, bússulas), proteção (armas) e itens especiais (chaves de veículos, combustíveis e peças de veículos).

Os jogadores poderão escolher seu objetivo final (fugir da ilha ou destruir a colmeia central) com base nos itens que encontrarem e na situação de seus personagens. As unidades do Enxame serão controladas por uma inteligência artificial usando técnicas de neuroevolução multimodal, de modo que apresentem uma adaptação constante ao estilo de jogo do jogador humano. O movimento dos personagens acontecerá de maneira livre em um ambiente tridimensional.

A.11.2 Mecânicas do jogo

Enquanto trabalham de maneira cooperativa, os jogadores encontram itens, como medicamentos, mapas, armas e chaves, que podem auxiliar o grupo de sobreviventes durante a jornada. Além disso, os itens estão dispostos de maneira aleatória pelo mapa, fazendo com que seja necessário explorar a ilha. Entretanto, os jogadores podem sofrer dano, que, quando acumulado em um curto período de tempo irá causar a morte do personagem. Cada jogador humano terá uma quantidade predefinida de "segundas chances". Após cada morte, uma "segunda chance" será decrementada do total. Quando não houverem mais "segundas chances" o jogador não poderá mais retornar ao jogo.

Existe um motivo especial para que os jogadores humanos se preocupem com a quantidade de dano recebido. Isto porque existem outros habitantes na ilha: o Enxame. O Enxame consiste de um conjunto de indivíduos alienígenas, os Colmeeiros, e uma colmeia central. Os objetivos dos Enxame são eliminar os jogadores humanos e proteger a colmeia central. Os colmeeiros trabalham de maneira cooperativa para realizar tarefas como encontrar e atacar os jogadores ou para proteger o Enxame.

A colmeia central é a responsável pelo aparecimento/reaparecimento de colmeeiros. Entretanto, novos colmeeiros podem surgir apenas quando um sobrevivente morre ou quando o número de colmeeiros está abaixo de um mínimo previamente estipulado. De qualquer maneira, todo colmeeiro deve esperar um tempo de amadurecimento antes de entrar no jogo.

Índice