**UNIVERSIDADE ESTADUAL PAULISTA**
**FACULDADE DE CIÊNCIAS**
**DEPARTAMENTO DE COMPUTAÇÃO**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**MATEUS GONÇALEZ ETTO**

# UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL EM JOGO RPG

BAURU – SP
2016

**MATEUS GONÇALEZ ETTO**

# UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL EM JOGO RPG

Trabalho de Conclusão de Curso de graduação apresentado à disciplina Projeto e Implementação de Sistemas do curso de Bacharelado em Ciência da Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" como requisito para obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Simone das Graças Domingues Prado

BAURU – SP
2016

**MATEUS GONÇALEZ ETTO**

# UTILIZAÇÃO DE INTELIGÊNCIA ARTIFICIAL EM JOGO RPG

Trabalho de Conclusão de Curso de graduação apresentado à disciplina Projeto e Implementação de Sistemas do curso de Bacharelado em Ciência da Computação da Faculdade de Ciências da Universidade Estadual Paulista "Júlio de Mesquita Filho" como requisito para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

_____

Profa. Dra. Simone das Graças Domingues Prado

_____

(Nome do segundo membro da Banca Examinadora)

_____

(Nome do terceiro membro da Banca Examinadora)

Bauru, ____ de _____ de _____

# RESUMO

Este projeto trata-se da criação de um protótipo de jogo no estilo RPG em turnos, em conjunto com a criação de uma Inteligência Artificial capaz de controlar os personagens, assim como de aprender a controlá-los melhor com treinamento. Para a criação da Inteligência Artificial, foram usados conceitos de Redes Neurais Artificiais e Algorítimo Genético, e para a criação do jogo e seus scripts foi usado o motor de jogo Unity.

**PALAVRAS-CHAVE:** Inteligência Artificial, Unity, jogo RPG

# ABSTRACT

This project comes to creating a turn-based RPG game prototype, in conjunction with an Artificial Intelligence that is able to control the characters as well as to learn how to control them better with training. For the creation of the Artificial Intelligence, concepts of Artificial Neural Networks and Genetic Algorithm were used, and for the creation of the game and its scripts, the Unity game engine was used.

**KEY-WORDS:** Artificial Intelligence, Unity, RPG game

# LISTA DE FIGURAS

# LISTA DE TABELAS

# SUMÁRIO

# 1 INTRODUÇÃO

Há ainda muitas pessoas que acreditam que jogos eletrônicos são para crianças ou pessoas desocupadas. Talvez isso fosse verdade no milênio passado, mas a realidade vem se mostrando ser bem diferente.

De acordo com uma pesquisa da SuperData, o mercado de jogos cresceu 8% de 2014 a 2015, com 61 bilhões de dólares circulando nesta indústria (CNBC, 2016). Em 2014, o valor da indústria de jogos já havia ultrapassado o da indústria de música em 20 bilhões, e está chegando ao da indústria do filmes (NYTIMES, 2014).

Então é fato, a indústria de jogos está movendo bilhões de dólares pelo mundo, já passou do de música e não para de crescer. Como diz o gerente de produto da Eletronic Sports League, James Lampkin: "Isto está se expandindo fora de controle" (NYTIMES, 2014, tradução nossa). Tais palavras explicam muito bem o estado atual do mercado de jogos.

E não é só nas vendas de jogos e consoles, existem muitos torneios de jogos ocorrendo pelo mundo, surgindo uma nova categoria de profissionais que, em poucos anos atrás era inimaginável, senão motivo de piada, que é a categoria de jogador profissional de jogo eletrônico. A área de trabalho já existe e é chamada de Esporte Eletrônico (NYTIMES, 2014).

E mesmo nestes torneios, não é por pouco dinheiro que os jogadores se enfrentam. No Campeonato Mundial de 2015 (o quinto da série) de League of Legends, foi oferecido 1 milhão de dólares para a equipe vencedora mundial do jogo, como pode ser visto nas regras do campeonato[1].

Os prêmios não param por aí. O jogo Dota 2 distribuiu 11 milhões de dólares para os 10 melhores do mundo, sendo 5 milhões para os campeões, se tornando assim o maior prêmio já oferecido em um torneio de jogo eletrônico (NYTIMES, 2014).

E tem muita gente para assistir a estes campeonatos. Nos dados mostrados pela Riot[2] sobre o Campeonato Mundial de 2015, houveram 334 milhões de telespectadores "únicos" durante as 4 semanas do torneio, somando 360 milhões de horas de visualizações das partidas ao vivo.

Mas a área de jogos não está chamando apenas a atenção do mercado, mas também a de pesquisadores. Um exemplo é o desenvolvimento e aplicação de técnicas de Inteligência Artificial (IA) em jogos, que de acordo com especialistas, existem áreas dentro de IA em jogos que ainda estão inexplorados (YANNAKAKIS; TOGELIUS, 2014).

Em 2007, foi montada pela AiGameDev uma lista dos 10 jogos com Inteligência

[1] Regras: https://riot-web-static.s3.amazonaws.com/lolesports/Rule%20Sets/2015%20Revised%20World%20Championship%20Rule%20Set%20Version%201_01.pdf

[2] Dados disponíveis em: http://www.lolesports.com/en_US/articles/worlds-2015-viewership

Artificial mais influentes. Um exemplo é um jogo chamado Creatures, que implementou aprendizado de máquina em uma simulação interativa ao usar Redes Neurais nas criaturas do jogo. Outro exemplo é o Halo, o jogo que implementou pela primeira vez a "árvore de condutas", tecnologia que ficou muito popular na indústria de jogos (AIGAMEDEV, 2007).

Um exemplo de IA em jogo que é descrito em detalhes é o F.E.A.R., um jogo FPS em primeira pessoa que criou um sistema dinâmico, coordenado, interessante e desafiador. Isto foi feito utilizando um sistema chamado Goal Oriented Action Planning (Planejamento de Ações Orientado a Objetivo), que foi construído junto de duas técnicas: Algoritmo A* e Máquina de Estados Finitos. Os NPCs possuem uma lista de objetivos, então durante o jogo eles buscam o plano que irá completar o objetivo com maior prioridade. O planejamento feito é similar ao STRIPS, tendo-se a situação atual e quais são as ações necessárias para cumprir o objetivo. Além disto, foi implementado uma extensão da conduta individual dos NPCs, com uma conduta em equipe. No entanto, como a conduta dos NPCs foi criada para minimizar a ameaça a si mesmo, os extintos básicos do NPC podem sobrescrever a conduta de equipe caso a segunda opção seja muito arriscado para si mesmo (ORKIN, 2006).

Percebe-se, desta forma, que a área de jogos está muito ativa e em pleno crescimento, tanto no mercado quando em pesquisas. Existem áreas inexploradas de IA em jogos, com novas fronteiras a serem exploradas. Com isto em mente, este trabalho foi desenvolvido, e espera-se contribuir com a comunidade acadêmica e/ou mercado de alguma forma.

## 1.1 Objetivos do Trabalho

### 1.1.1 Objetivo Geral

Produzir um jogo RPG em turnos que implementa conceitos avançados de Inteligência Artificial, sendo esta inteligência capaz de tomar decisões de forma autônoma sobre o que deve fazer, assim como ser capaz de aprender a tomar melhores decisões por meio de treinamento.

### 1.1.2 Objetivos Específicos

a. Criar um jogo RPG razoavelmente complexo.
b. Criar uma Inteligência Artificial capaz de jogar o jogo tão bem quanto um ser humano.
c. Criar uma Inteligência Artificial capaz de aprender conforme joga.

## 1.2 Organização da Monografia

Este trabalho está dividido em 5 seções, sendo esta seção (Introdução) a primeira. As outras seções são:

- Seção 2, **Ferramentas Utilizadas:** apresentação das ferramentas utilizadas para o desenvolvimento do projeto proposto no trabalho.
- Seção 3, **Conceitos:** explicação das teorias de Inteligência Artificial e Algoritmo Genético usadas para desenvolver o trabalho.
- Seção 4, **O Jogo:** descrição em detalhes do jogo, suas variáveis, e de sua implementação.
- Seção 5, **Resultados:** apresentação dos resultados obtidos no trabalho.

# 2 FERRAMENTAS UTILIZADAS

## 2.1 Unity

A Unity é um motor de jogo multiplataforma que permite a criação de jogos 2D ou 3D. Possui uma interface gráfica que permite desenvolver jogos com facilidade, além de ter muitos serviços integrados que aceleram o processo de desenvolvimento. As linguagens de programação que podem ser usadas são UnityScript e C# (UNITY, 2016a).

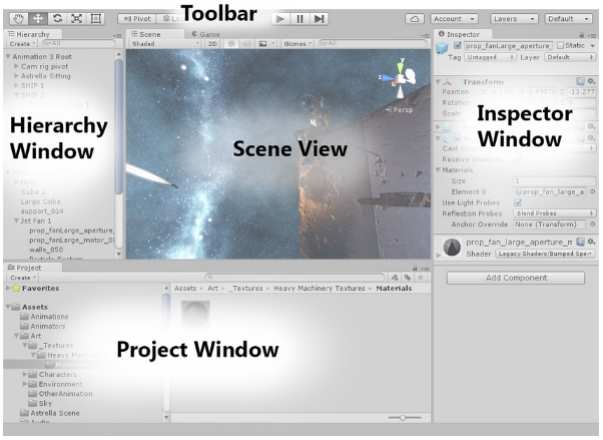
Com mais de 200 jogos na lista de jogos em destaque que foram criados na Unity, um dos exemplos que pode-se citar é o *Sky Force*, um jogo no estilo de ação, missões e "shooter", e que está disponível na App Store e Google Play. Outro exemplo é o *The Uncertain*, do qual o jogador é um robô e deve resolver quebra-cabeças em uma aventura. O jogo está disponível na *Steam* (UNITY, 2016b).

O editor da Unity também é extensível, sendo possível implementar funcionalidades ainda não existentes (UNITY, 2016a). Um exemplo de extensão é o *Rival Theory*, que implementa vários conceitos de Inteligência Artificial, desde funcionalidades básicas como *pathfinding*, condutas de patrulha, esconder, atacar, seguir, vaguear, até conceitos mais complicados como percepção e árvore de condutas (RIVAL THEORY, 2015).

### 2.1.1 Interface

A interface da Unity é composta por vários painéis chamados *views*, que podem ser rearranjados, agrupados, separados e fixados (UNITY, 2016c). O arranjamento padrão das janelas permite o acesso às janelas mais comuns, e pode ser visto na figura a seguir:

Figura 1: A interface padrão da Unity




Fonte: Unity.

A descrição de cada painel pode ser vista na tabela a seguir:

Tabela 1: Descrição dos painéis na Unity

| | |
|---|---|
| Project Window | Exibe a biblioteca de Assets que estão disponíveis para uso no projeto. Ao importar os Assets no projeto, eles aparecerão aqui. |
| Scene View | Permite navegar visualmente e editar a cena. A Scene View pode mostrar em perspectiva 3D ou 2D, dependendo do tipo de projeto que está sendo trabalhando. |
| Hierarchy Window | É uma representação de texto hierárquica de cada objeto na cena. Cada item na cena tem uma entrada na hierarquia, de forma que as duas janelas estão inerentemente conectadas. A hierarquia revela a estrutura da forma como os objectos estão ligados um ao outro. |
| Inspector Window | Permite visualizar e editar todas as propriedades do objeto selecionado. Como diferentes tipos de objetos têm diferentes conjuntos de propriedades, o layout e conteúdo da janela do Inspector pode variar. |
| Toolbar | Fornece acesso aos recursos de trabalho mais essenciais. À esquerda estão as ferramentas básicas para manipular a Scene View e os objetos dentro dela. No centro estão os controles de play, pause e step. Os botões à direita dará acesso aos Serviços da Unity Cloud e da Conta Unity, seguido pelo menu de visibilidade dos layers e, finalmente, o menu de layout do editor (que fornece alguns layouts alternativos para as janelas do editor, e permite salvar um layout customizado). |

Fonte: Unity.

Os objetos contidos na cena do projeto são chamados de *GameObject*, sendo que suas características podem ser alteradas por Componentes anexados a ele. A Unity possui vários Componentes prontos, no entanto é possível criar novos usando as linguagens de programação do qual se dá suporte (C# e UnityScript). Tais Componentes são chamados de Scripts, e eles permitem disparar eventos, modificar as propriedades de outros Componentes ou do próprio *GameObject* durante o jogo, e a interação com o jogador. (UNITY, 2016d).

## 2.2 Visual Studio

O Visual Studio é um Ambiente de Desenvolvimento Integrado (IDE) usado para a criação de aplicativos para Windows, Android, iOS, aplicações Web e serviços de nuvem. Com ele é possível programar em C#, Visual Basic, F#, C++, HTML, JavaScript e Python, dentre outras linguagens de programação. (MICROSOFT, 2016)

A integração do Visual Studio com a Unity, usando C#, ocorre com a adição do *namespace* UnityEngine ao script, liberando a utilização da classe *MonoBehaviour*, da qual possui implementado funcionalidades e funções internas da Unity. Para utilizar a classe *MonoBehaviour* nos scripts criados, basta estender a classe criada com a *MonoBehaviour* (UNITY, 2016d). Desta forma, o cabeçalho do script fica como no código a seguir:

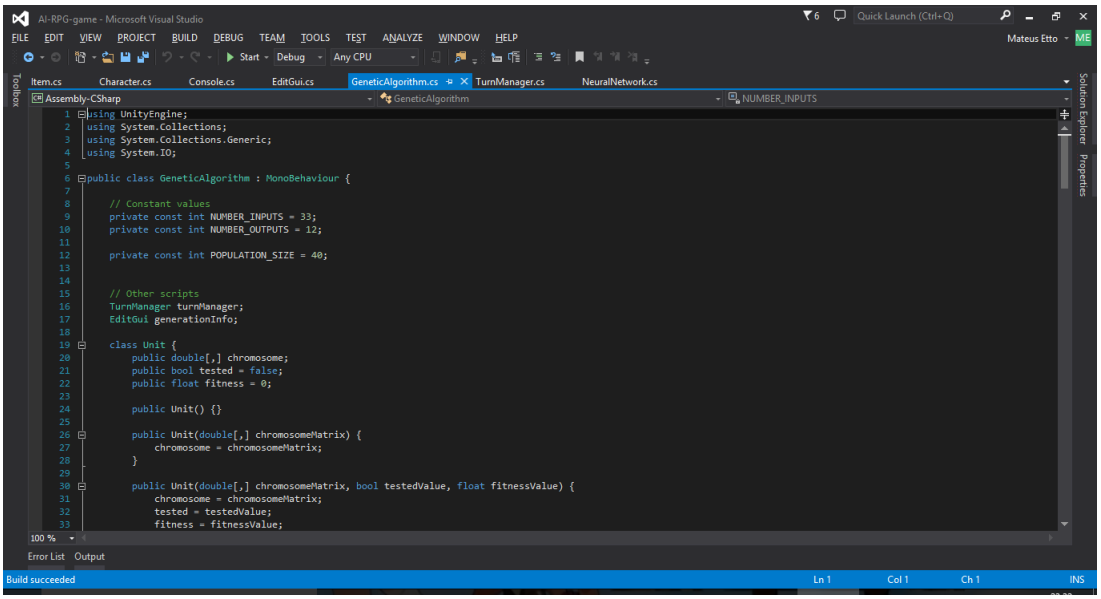
```
using UnityEngine;

public class nomeDaClasse : MonoBehaviour
{
  // Código da classe
}
```

Com isto feito, é possível utilizar uma das principais vantagens do Visual Studio, que é o *AutoComplete*. Ou seja, funções internas da Unity, como as do GameObject, aparecem em uma lista conforme se vai digitando, facilitando muito a programação do script.

Além da integração com a Unity, ainda é possível utilizar as bibliotecas próprias do C#, como bibliotecas matemáticas, acesso à arquivo e listas.

Na *Figura 2* a seguir, é possível ver um exemplo de classe criada no Visual Studio que foi usada neste projeto:

Figura 2: Exemplo de classe no Visual Studio




Fonte: Elaborado pelo autor.

Note que no script de Algoritmo Genético, além do *UnityEngine* e outras duas bibliotecas básicas do C#, também foi utilizada o *System.IO*, uma biblioteca para Leitura e Escrita de Arquivo. Qualquer outro script criado para funcionar na Unity tem este padrão.

# 3 CONCEITOS

Nesta seção será descrito dois conceitos que foram amplamente utilizados neste trabalho, que são Redes Neurais Artificiais (RNA) e Algoritmo Genético (AG). Apesar de ser possível descrever várias variações de aplicação desses conceitos, será explicado apenas a essência deles, e o que foi necessário para desenvolver este trabalho.

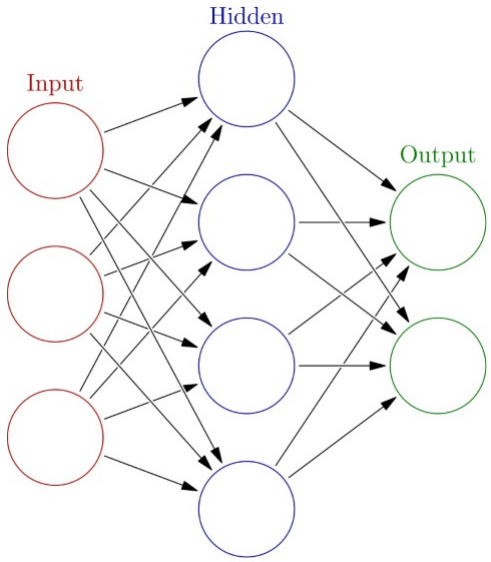
## 3.1 Redes Neurais

As Redes Neurais Artificiais foram inspiradas no funcionamento do cérebro humano, e a "baixo nível" procura imitar o que acontece nos neurônios.

Uma RNA é normalmente utilizada para duas finalidades: reconhecimento de padrões e regressão (ou aproximação de função). A arquitetura de uma RNA é formada por uma camada de entrada (input layer), uma camada de saída (output layer), e as camadas ocultas (hidden layers), que podem ter de zero a muitas camadas.

Um exemplo de Rede Neural Artificial com uma camada oculta pode ser visto na Figura 3 a seguir:

Figura 3: RNA com uma camada oculta



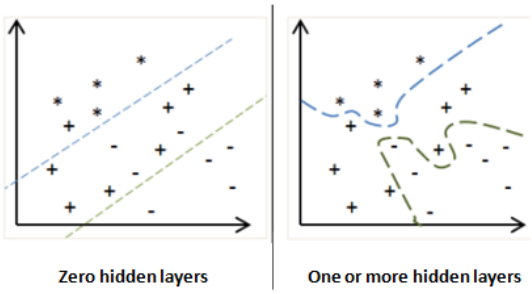

Fonte: Wikimedia.

Uma Rede Neural Artificial sempre terá uma camada de entrada e uma de saída. No entanto, o número de camadas ocultas pode variar bastante entre um problema e outro, assim como o número de neurônios contidos nessas camadas. Estas camadas ocultas são explicadas como sendo "extratoras de características" (STACKEXCHANGE, 2013).

Como o número de camadas ocultas é escolhido? Caso os dados sejam linearmente separáveis, não é necessário nenhuma camada oculta. Existem muitas

discussões a respeito do número de camadas ocultas a serem usadas, caso ela seja necessária, mas um consenso que existe é que uma camada é suficiente para a maioria dos problemas (STACKEXCHANGE, 2013).

Figura 4: Importância de ter camada oculta




Fonte: Stackoverflow.

Como pode-se notar na Figura 4, o aumento de camadas ocultas aumenta a granularidade do reconhecimento de padrões, fazendo-se assim uma separação não-linear no reconhecimento dos dados de entrada.

Para descobrir o número de neurônios a serem colocados na camada oculta existem muitas regras empíricas que ajudam a escolher um valor, mas uma delas que funciona com frequência é encontrar a média do número de neurônios de entrada e de saída. Usa-se este valor empírico como valor inicial para número de neurônios na camada oculta, e então ajusta-se este valor com testes (STACKEXCHANGE, 2013).

A escolha da função de saída dos neurônios também tem bastante influência na convergência da Rede Neural. De maneira geral pode-se separar que tipo de função a ser utilizada dependendo da tarefa: reconhecimento de padrões ou regressão de uma função (RESEARCHGATE, 2013).

Caso a tarefa seja de reconhecer padrões, as funções recomendadas são a step, sigmóide e tangente hiperbólica. O motivo é que estas funções retornam valores entre um intervalo pré-estabelecido. (RESEARCHGATE, 2013).

Por outro lado, caso a tarefa seja de regressão de uma função, pode-se utilizar a função linear na camada de saída. No entanto, nas camadas ocultas ainda é necessário o uso de funções não lineares. Caso as camadas ocultas também sejam de função linear, as soluções a serem encontradas serão apenas linearmente separáveis, equivalente a uma rede sem camada oculta (RESEARCHGATE).

## 3.2 Algoritmo Genético

Algoritmo genético faz parte da computação evolutiva, e baseia-se na teoria de Darwin sobre a evolução das espécies.

O algoritmo inicia com um conjunto de soluções, e eles são representados por cromossomos. Este conjunto é a população, e aqueles que conseguirem se adaptar melhor, passarão para a próxima geração. A motivação do AG é que as soluções melhoram a cada geração.

De maneira geral, um Algoritmo Genético segue o seguinte processo:

- **Iniciar População:** Gera uma população aleatória de n cromossomos.
- **Teste:** Testa o valor de fitness f(x) de cada cromossomo x da população. Se a condição de parada for satisfeita, o algoritmo acaba.
- **Seleção:** Seleciona 2 cromossomos de acordo com o fitness deles, ou seja, quanto maior o valor de fitness, maior a probabilidade. Normalmente é utilizado o método da roleta para a seleção.
- **Crossover:** Copia partes dos cromossomos dos pais em cromossomos filhos, aplicando uma determinada probabilidade de alternar o trecho a ser copiado.
- **Mutação:** Probabilidade de algum valor do cromossomo do novo filho ser alterado.
- **Inserção:** Insere os novos cromossomos na população.
- **Loop:** Retorna ao passo de calcular o "fitness".

No trabalho aqui desenvolvido, Redes Neurais Artificiais e Algoritmos Genéticos são utilizados juntos. O cromossomo do Algorítmo Genético se torna, neste caso, os pesos que estão nos neurônios da RNA. Desta forma, o AG estará procurando as soluções ótimas da Rede Neural.

# 4 O JOGO

## 4.1 Descrição do jogo

<Descrever o jogo, suas variáveis, complexidades, particularidades, etc>

## 4.2 Rede Neural Artificial Implementado

<Descrever em detalhes qual foi a Rede Neural Artificial implementada>

## 4.3 Algoritmo Genético Implementado

<Descrever em detalhes qual foi o Algoritmo Genético implementado>

## 4.4 Funcionamento do jogo

<Descrever como ocorre o funcionamento do jogo como um todo, dado que o leitor sabe os detalhes de cada parte.>

# 5 RESULTADOS

<Inserir gráficos de evolução do algoritmo, imagens do jogo, explicar desempenho da aprendizagem, etc.>

# REFERÊNCIAS

UNITY. Game engine, tools and multiplatform. 2016a. Disponível em: <https://unity3d.com/pt/unity>. Acesso em 01 Julho 2016.

UNITY. Made with Unity - Games. 2016b. Disponível em: <https://madewith.unity.com/games>. Acesso em 01 Julho 2016.

UNITY. Made with Unity - Manual: Learning the Interface. 2016c. Disponível em: <https://docs.unity3d.com/Manual/LearningtheInterface.html>. Acesso em 05 Julho 2016.

UNITY. Made with Unity - Manual: Creating and Using Scripts. 2016d. Disponível em: <https://docs.unity3d.com/Manual/CreatingAndUsingScripts.html>. Acesso em 05 Julho 2016.

RIVAL THEORY. Features. 2015. Disponível em: <http://rivaltheory.com/rain/features/>. Acesso em 01 Julho 2016.

MICROSOFT. Free Dev Tools - Visual Studio Community 2015. 2016. Disponível em: <https://www.visualstudio.com/en-us/products/visual-studio-community-vs.aspx>. Acesso em 07 Julho 2016.

CNBC. Digital gaming sales hit record $61 billion in 2015: Report. Disponível em: <http://www.cnbc.com/2016/01/26/digital-gaming-sales-hit-record-61-billion-in-2015-report.html>. Acesso em 11 Julho 2016.

YANNAKAKIS, Georgios N; TOGELIUS, Julian. A Panorama of Artificial and Computational Intelligence in Games. Disponível em: <http://julian.togelius.com/Yannakakis2014P
pdf>. Acesso em 11 Julho 2016.

AIGAMEDEV. Top 10 Most Influential AI Games. Disponível em: <http://aigamedev.com/open/highlights/top-ai-games/>. Acesso em 12 Julho 2016.

ORKIN, Jeff. Three States and a Plan: The A.I. of F.E.A.R. Disponível em: <http://alumni.media.mit.edu/~jorkin/gdc2006_orkin_jeff_fear.pdf>. Acesso em 12 Julho 2016.

STACKEXCHANGE. How to choose the number of hidden layers and nodes in a feedforward neural network?. Disponível em: <https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw>. Acesso em 13 Julho 2016.

STACKEXCHANGE. What does the hidden layer in a neural network compute?. Disponível em: <https://stats.stackexchange.com/questions/63152/what-does-the-hidden-layer-in-a-neural-network-compute>. Acesso em 13 Julho 2016.

STACKOVERFLOW. Role of Bias in Neural Networks. Disponível em: <https://stackoverflow.com/questions/2480650/role-of-bias-in-neural-networks>. Acesso em 13 Julho 2016.

OBITKO, Marek. Introduction to Genetic Algorithms. Disponível em: <http://www.obitko.com/tutorials/genetic-algorithms/index.php>. Acesso em 13 Julho 2016.

RESEARCHGATE. How to select the best transfer function for a neural network model?. Disponível em: <https://www.researchgate.net/post/How_to_select_the_best_transfer_function_for_a_neural_network_model>. Acesso em 13 Julho 2016.

Wikimedia. File:Colored neural network.svg. Disponível em: <https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg>. Acesso em 13 Julho 2016.