Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

**ADDAM CAUÊ PERES RAFACHO**

# ESTUDO EXPLORATÓRIO DE ALGORITMOS GENÉTICOS PARA APRENDIZADO DE MÁQUINA EM JOGOS

**Assis/SP**

**2018**

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

**ADDAM CAUÊ PERES RAFACHO**

# ESTUDO EXPLORATÓRIO DE ALGORITMOS GENÉTICOS PARA APRENDIZADO DE MÁQUINA EM JOGOS

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso de Graduação.

**Orientador:** Prof. MSc. Guilherme de Cleva Farto

**Área de Concentração:** Informática

**Assis/SP**

**2018**

# ESTUDO EXPLORATÓRIO DE ALGORITMOS GENÉTICOS PARA APRENDIZADO DE MÁQUINA EM JOGOS

**ADDAM CAUÊ PERES RAFACHO**

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso de Graduação, analisado
pela seguinte comissão examinadora:

**Orientador:** Prof. MSc. Guilherme de Cleva Farto
**Examinador:** Prof. Dr. Luiz Ricardo Begosso

**Assis/SP**

**2018**

# RESUMO

Na busca do desenvolvimento de modelos computacionais cada vez mais inteligentes, foram surgindo técnicas cada vez mais eficientes na análise de dados, oportunizando aos sistemas a capacidade de deduzir e concluir por meio de dados de entrada. Dentre essas técnicas, surgiu na área de Aprendizado de Máquina abordagens inspiradas nas teorias evolucionais de Darwin, o Algoritmo Genético (AG).

Por meio dos AGs, este trabalho busca explorar a utilização de seus conceitos no âmbito de desenvolvimento de jogos, de forma a utilizar frameworks e estratégias recentes disponíveis.

**Palavras-chave:** Modelos Computacionais, Análise de Dados, Aprendizado de Máquina, Algoritmo Genético, Desenvolvimento de Jogos

# ABSTRACT

In the search for the development of increasingly intelligent computational models, increasingly effective techniques have been arising for data analysis, enabling to systems the ability to deduct and conclude using data inputs. Between those techniques, approaches inspired by the evolutional theories of Darwin has arisen in Machine Learning field, the Genetic Algorithm (GA).

This work seeks to explore the uses of GA concepts in the field of game development, in order to use recents frameworks and strategies available.



**Keywords:** Computational Models, Data Analysis, Machine Learning, Genetic Algorithm, Game Development

# Lista de Ilustrações

# LISTA DE TABELAS

## SUMÁRIO

# 1 – INTRODUÇÃO

No cenário atual, o Aprendizado de Máquinas (AM), no inglês, *Machine Learning*, está presente em diversas tarefas que são realizadas frequentemente pela sociedade. É apenas parar e observar que conseguimos visualizar em diversos lugares a sua utilização. Um dos destaques está a *Netflix*, a qual transformou sua plataforma para que coletasse dados das visualizações de seus usuários para oferecer *shows* mais indicados aos seus gostos (WATKINS, 2016).

A *Amazon* é outro exemplo, a empresa vem investindo neste segmento por mais de 20 anos, e que, por sua vez, chegou a construir sua própria *Application Programming Interface* (API) de *Machine Learning*. Por esta base, a empresa faz uso dessa tecnologia em diferentes situações, dentre elas estão recomendações de produtos, processos de previsão, e até mesmo na tecnologia da Alexa, uma inteligência artificial que é alimentada por um aprendizado profundo de compreensão de linguagem natural e reconhecimento natural da fala (AMAZON, 2017).

Suas vantagens em questão do aprendizado estão possibilitando aos sistemas a melhorar a experiência do usuário gradualmente ao longo do tempo de utilização, fornecendo respostas mais adequadas para as ações dos mesmos (BELL, 2015).

## 1.1 OBJETIVOS

Esta pesquisa tem por objetivo estudar e investigar as diferentes abordagens sobre o tema de algoritmos genéticos junto aos conceitos de *Machine Learning*, contemplando e analisando o seu uso, de forma a entender seus processos de implementação e limitações, bem como entender o uso destes para o desenvolvimento de jogos. Para isso, a pesquisa também propõe a implementação de um protótipo de jogo que faça uso destas abordagens.

Também objetiva-se avaliar, por meio de experimentos e projetos práticos, as implementações e resultados obtidos por meio das técnicas envolvidas no estudo do tema.

## 1.2 JUSTIFICATIVAS

Os Algoritmos Genéticos mostram conceitos promissores por meio da evolução e de como atuam em modelos computacionais. Thomas Riechmann (2000) apresenta um estudo sobre a relação entre algoritmos genéticos e jogos evolucionários, buscando realizar a implementação destes conceitos na teoria de jogos. Desta forma, jogos que evoluem aprendendo com o ambiente em que são inseridos oferecem diversas possibilidades para diferentes resultados que auxiliam no entretenimento em jogos.

No âmbito de desenvolvimento, como na construção de robôs (no inglês, *bots*) de jogos, este processo se torna trabalhoso e complexo, sendo muitas vezes necessário definir diversos ajustes para o funcionamento dos mesmos e também da dependência de uma vasta quantidade de parâmetros específicos em suas implementações. Além disso, as restrições dos computadores ainda é um desafio ao criar objetos virtuais inteligentes e adaptados para as habilidades do jogador, resultando na dificuldade de desenvolver jogos mais imersivos (KHOO, ZUBECK, 2002; COLE et al., 2009).

## 1.3 MOTIVAÇÃO

A motivação de realizar este trabalho surge pelo interesse de explorar as áreas emergentes de Aprendizado de Máquina e Algoritmos Genéticos (AGs) em conjunto com o processo de desenvolvimento de jogos computacionais, possibilitando novas estratégias para cenários, personagens e elementos inseridos no jogo. Apesar de AGs serem explorados em distintos contextos, esta pesquisa direcionará esforços para sua aplicabilidade em jogos.

Algoritmos genéticos possuem uma ampla capacidade de atuação, sendo não apenas limitados ao uso em jogos. A sua importância cresce em problemas que são previamente definidos, contribuindo com o uso de algoritmos eficientes na recomendação de soluções para diferentes situações configuradas, fazendo-se isso de fundamentos que também compõem a evolução humana.

O crescimento de tecnologias de algoritmos genéticos e de Aprendizado de Máquina também não são pequenos, para Ben Aron (2017), a Deloitte prevê, apenas nos EUA, uma redução de 16%, em relação a 2017, em mortes causadas por acidentes por meio de veículos 100% autônomos. Por outro lado, na medicina, por meio de algoritmos genéticos, a seleção de robôs cada vez mais aptos, como em cirurgias ou previsão de doenças.

## 1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado nas seguintes partes:

- **Capítulo 1 – Introdução**
- **Capítulo 2 – Machine Learning**
- **Capítulo 3 – Algoritmos Genéticos**
- **Capítulo 4 – Jogos digitais**
- **Capítulo 5 – Propostas de Trabalho**
- **Capítulo 6 – Implementação**
- **Capítulo 7 – Conclusão**
- **Referências**

# 2 – APRENDIZADO DE MÁQUINA

Neste capítulo serão apresentados os conceitos fundamentais de Aprendizado de Máquina e uma breve explicação de como surgiu e quando começou a exploração desta área da computação.

## 2.1 DEFINIÇÕES DE APRENDIZADO DE MÁQUINA

Aprendizado de Máquina (no inglês, *Machine Learning*) é uma área definida por abordagens computacionais que envolvem conjuntos de informações para encontrar e/ou recomendar uma solução para um determinado problema (MOHRI, 2012).

Esta área de pesquisa teve surgimento por volta de 1950, quando Alan Turing questionou a possibilidade de máquinas pensarem, pergunta que apresentou em sua pesquisa nomeada de “O Jogo da Imitação”. Por meio dessa pesquisa, Turing buscou apresentar a ideia de uma máquina pensante, capaz de aprender e de até mesmo possivelmente imitar o comportamento humano (BELL, 2015).

Pesquisadores da área de Aprendizado de Máquina definem um comportamento parecido com a que Turing previu, isto é, uma máquina capaz de buscar por padrões significativos presentes nos dados por meio de detecções automatizadas (SHALEV-SHWARTZ, BEN-DAVID, 2014).

Na definição de Monard e Baranaukas (2003), a máquina é capaz de extrair informações novas a partir de informações já existentes, também visto como a capacidade de gerar conclusões, isto é, deduzir. A este fenômeno foi dado a definição de Aprendizado Indutivo, o qual é possível separar em duas abordagens principais, Aprendizado Supervisionado e Não Supervisionado, como Ilustrado na Figura 1.

Por sua vez, uma explicação mais simplificada para este conceito pode ser dita como a programação de computadores capazes de aprenderem por meio de informações recebidas e atribuir outros significado a estas informações, bem como utilizá-las.

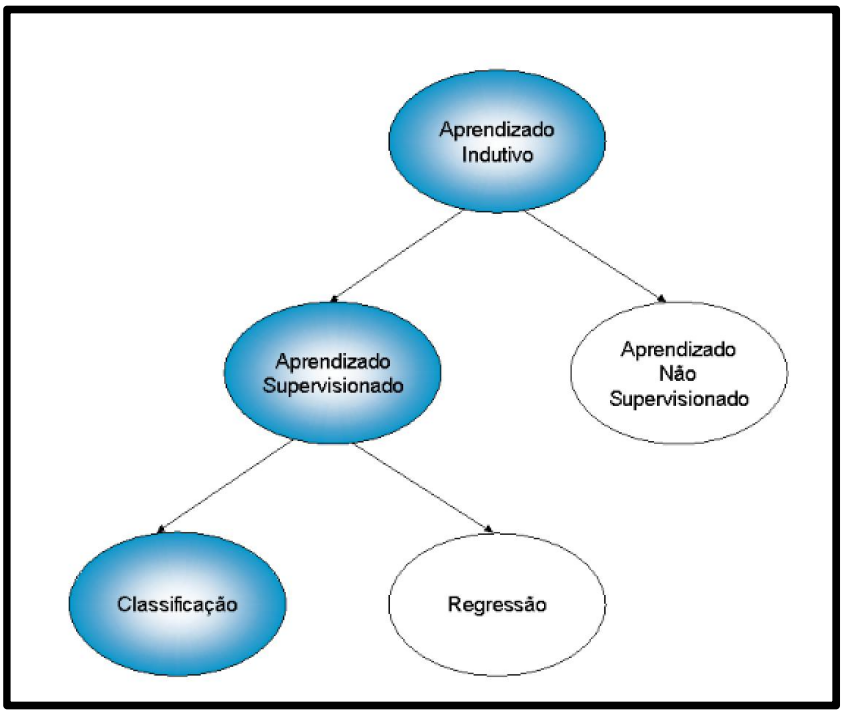


**Figura 1: Hierarquia do aprendizado indutivo (In: MONARD, BARANAUSKAS, 2003)**

Embora Monard e Baranauskas (2003) tenham apresentado e discutido aprendizados supervisionados e não supervisionado, alguns outros autores, como Norvig e Russel (2010), destacam a presença de outros tipos e abordagens, como o semi-supervisionado e aprendizado reforçado.

Shalev-Swartz e Ben-David (2014) também destacam a diferença que faz a o papel do aprendiz, separando essa abordagem em dois, agente ativo e passivo, sendo que o ativo pode interagir diretamente com os dados em tempo de treinamento, realizando consultas ou experimentos, e o passivo, apenas observa a informação sem influenciar. Estes tipos de aprendizados são representados na Tabela 1.

| Tipos de Aprendizados | Definição |
|---|---|
| Supervisionado | Aprendizado por meio de dados rotulados |
| Não supervisionado | Aprendizado por meio de dados não rotulados |
| Semi-supervisionado | Aprendizado por meio de dados rotulados e não rotulados |
| Aprendizado reforçado | Aprendizado por meio de recompensas ou punições |

**Tabela 1: Tipos de aprendizados**

Shalev-Shwartz e Ben-David (2014) apresentam Aprendizado de Máquina como uma das ramificações da Inteligência Artificial (IA), entretanto, com a finalidade de complementar a inteligência humana, ao invés de buscar imitá-la, como no caso da IA.

## 2.2 ESTRATÉGIAS E TÉCNICAS

Russel e Norvig (2010) definem que um sistema possui aprendizado se um indivíduo, mais conhecido por agente, apresenta uma melhora em sua performance para concluir tarefas futuras, ao mesmo tempo em que realiza observações sobre o ambiente em que está situado.

Monard e Baranauskas (2003) definem alguns paradigmas relacionados aos tipos de aprendizados, como o simbólico, estatístico, baseado em exemplos, conexionista e genético, sendo este último referente aos algoritmos genéticos, que será explicado na Seção 3.

Na figura 1 também é possível visualizar duas abordagens para o Aprendizado Supervisionado:

- **Classificação:** Abordagem de aprendizado com dados discretos
- **Regressão:** Abordagem de aprendizado com valores contínuos

### 2.2.1 APRENDIZADO SUPERVISIONADO

No aprendizado supervisionado é fornecido ao algoritmo de aprendizado, ou indutor, um conjunto de exemplos de treinamento previamente classificados (MONARD, BARANAUSKAS, 2003).

Em geral, cada exemplo é descrito por um vetor de valores de características, ou atributos, e a sua classificação. O objetivo deste algoritmo é construir um classificador que possa determinar corretamente a classificação de novos exemplos ainda não classificados, ou seja, exemplos que não tenham o rótulo da classe. Para classificação de dados discretos, esse problema é conhecido como classificação e para dados contínuos como regressão como representado na Figura 1 (MONARD, BARANAUSKAS, 2003).

Jason Bell (2015) destaca algumas preocupações a serem consideradas ao tratar dados classificados, como o *bias-variance tradeoff* ou *bias-variance dilemma*, isto é, como o algoritmo lida com diferentes conjuntos de treinamentos, pois dependendo da abordagem, pode se prejudicar ao tratar de dados imprecisos e/ou inconsistentes (no inglês, *noisy data*), mas como consequência aprender com maior complexidade. Neste caso, deve-se escolher a abordagem mais apropriada sabendo que não existe abordagem perfeita para todos os casos.

### 2.2.2 APRENDIZADO NÃO SUPERVISIONADO

Segundo Monard e Baranauskas (2003), no aprendizado não-supervisionado o indutor analisa os exemplos fornecidos e tenta determinar se alguns deles podem ser agrupados de alguma maneira, formando agrupamentos ou clusters (apud CHEESEMAN, STUTZ, 1990). Após a determinação dos agrupamentos, normalmente é necessária uma análise para determinar o que cada agrupamento significa no contexto do problema que está sendo analisado.

### 2.2.3 APRENDIZADO SEMI-SUPERVISIONADO

Algoritmos de aprendizado supervisionado e não supervisionado oferecem abordagens específicas para dados rotulados e não rotulados, respectivamente, porém, nem sempre é possível garantir que todos os dados estarão rotulados, ainda que, em certas situações, adquirir dados específicos totalmente rotulados seja custoso, como para a classificação automática de páginas da *web* (CHAPELLE et al., 2006).

Russel e Norvig (2010) definem que no aprendizado semi-supervisionado, tem-se uma quantidade de dados classificados e uma grande quantidade de dados não classificados, de certa forma a fazer o que for possível para lidar com esses dois tipos de dados em conjunto.

### 2.2.4 APRENDIZADO REFORÇADO

Nos algoritmos de aprendizado reforçado, o agente aprende por meio de reforços, que neste contexto é compreendido como consequências por suas ações, sejam elas recompensas ou punições. Assim, o agente recebe indicações ao realizar as certas tarefas de forma a saber se foram corretas ou errados. Neste caso, o agente deve definir qual de suas ações dentro de cada tarefa foram responsáveis pela resposta positiva ou negativa (RUSSELL, NORVIG, 2010).

## 2.3 CASES E PROJETOS REAIS

Embora não seja comum encontrar empresas mencionando o uso de conceitos de Aprendizado de Máquina em jogos oficiais no mercado, diversas são as abordagens desenvolvidas pela comunidade com o intuito de estudar tais conceitos.

Um desses jogos é o *MarI/O*, uma reprodução do jogo *Mario* fazendo uso dos conceitos de Redes Neurais, uma das áreas de conhecimento de Aprendizado de Máquina, para ensinar o computador a jogar. A implementação foi feita em 2015 e está disponível no *Youtube* pelo canal *SethBling*, como ilustrado na Figura 2. Além disso, o autor disponibilizou o código fonte para que outras pessoas também possam aprender.

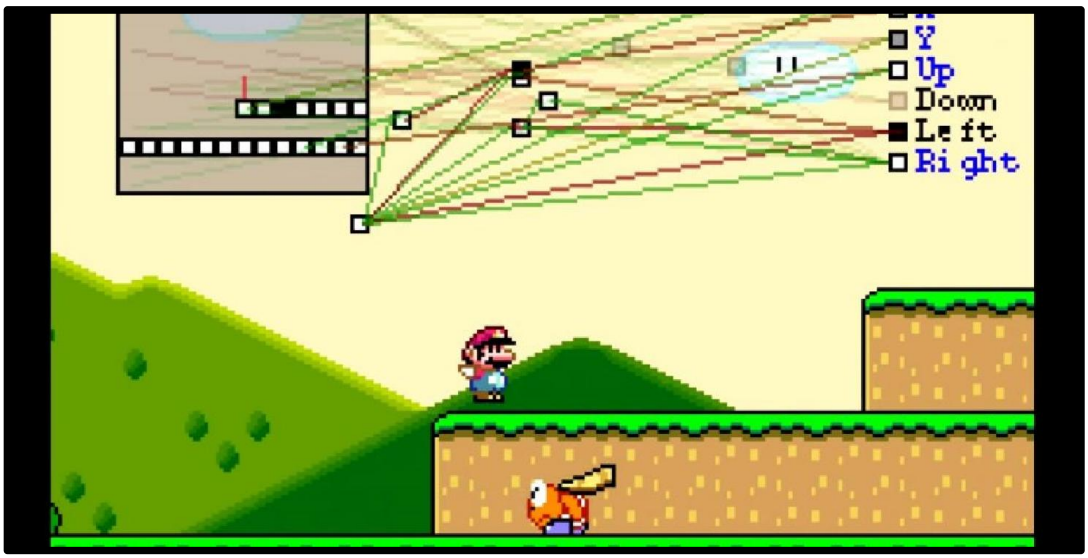


**Figura 2: MarI/O, reprodução do jogo Mario utilizando Redes Neurais (In: SETHBLING, 2015)**

Outra implementação, também fazendo uso dos conceitos de Redes Neurais, foi realizada pelo brasileiro Ivan Seidel, também em 2015. Seidel implementou estes conceitos no jogo de dinossauro da *Google*, o qual denominou de *IAMDinosaur*, e registrou a implementação em vídeo no *Youtube*, bem como o código fonte na plataforma *Github*. A Figura 3 é uma representação de sua implementação.

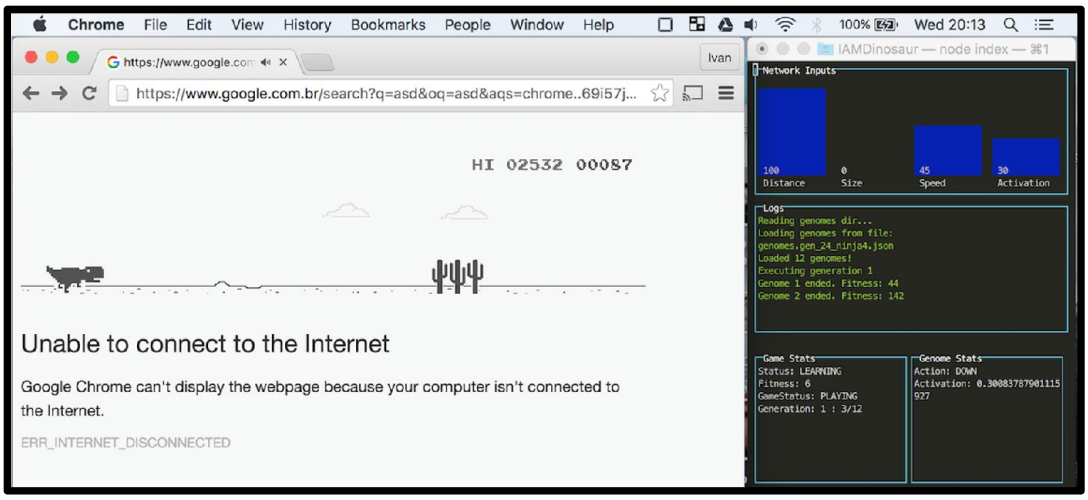


**Figura 3: IAMDinosaur (In: SEIDEL, 2015)**

## 2.4 PLATAFORMAS DE APRENDIZADO DE MÁQUINA

Existem diversas plataformas de Aprendizado de Máquina disponíveis para uso, como o *Google Prediction API*, *AWS Machine Learning* da *Amazon*, *Microsoft Azure Machine Learning*, *Weka*, entre outros.

Também existem diversas bibliotecas e módulos de Linguagens de Programação que oferecem meios para trabalhar com Aprendizado de Máquina, como o *GoLearn* da linguagem *GO*, *Scikit-learn* de *Python* e *JSAT* de *Java*. Essas plataformas existem para garantir maior facilidade e praticidade na implementação dos conceitos de Aprendizado de Máquina.

### 2.4.1 WEKA

A *Weka* (*Waikato Environment for Knowledge Acquisition*), no português, Ambiente Waikato para Aquisição de Conhecimento, é um ambiente que oferece uma coleção de abordagens de Aprendizado de Máquina para realizar tarefas de Mineração de Dados (no inglês, *Data Mining*). Seus algoritmos possuem utilidade tanto diretamente em conjuntos de dados quanto para o código Java e oferece ferramentas para pré-processamento, classificação, regressão, *clustering*, associação de regras, e visualização (WITTEN et al., 2018).

O ambiente também oferece suporte a aspectos de *Big Data* em uma interface com *Hadoop* para *clustered data mining* (BELL, 2015).

### 2.4.2 SCIKIT-LEARN

*Scikit-learn* é um modulo da Linguagem de Programação *Python* que integra uma variedade de algoritmos mais modernos para Aprendizado de Máquina. Seu foco principal é fornecer a não especialistas a possibilidade de trabalhar com abordagens dessa área usando uma linguagem de alto nível de propósito geral (PEDREGOSA et al., 2011). Isto diz que a ferramenta busca facilidade para os usuários por meio de documentação, performance e abordagens práticas para uso acadêmico.

Foi construída em cima de outros módulos conhecidos pela comunidade da linguagem *Python*, como *NumPy*, *SciPy* e *matplotlib*, famosas pelo seu alto uso em estudos científicos e acadêmicos (PEDREGOSA et al., 2011).

# 3 – ALGORITMOS GENÉTICOS

## 3.1 DEFINIÇÕES DE ALGORITMOS GENÉTICOS

Algoritmos Genéticos (AGs) são algoritmos de busca de dados estocásticos, isto é, aleatórios, e baseados nas mecânicas da seleção natural de uma população (PALIOURAS et al., 2001).

Os AGs compõem uma parte da área de Aprendizado de Máquina, pois dependem do aprendizado e do conhecimento, desta forma, sendo possível defini-los como modelos computacionais baseados nas teorias da evolução de Darwin, bem como nas descobertas sobre a reprodução e genética humana (WHITLEY, 1994; FERNEDA, 2009).

Surgiu por volta de 1960 e 1970, com estudos conduzidos por Holland. Seu surgimento foi devido a diversos estudos na época sobre algoritmos evolucionários. Os objetivos principais de Holland, em contrapartida com os algoritmos evolucionários em que a proposta era resolver problemas específicos, era estudar os fenômenos de adaptação presentes na natureza representados na computação (MITCHELL, 1998).

O exemplo mais comuns de AGs em diversas pesquisas é o mesmo representado por Holland em 1975, o qual cromossomos são arranjos compostos por 1 e 0, também denominados de genes, como segue a Figura 4:

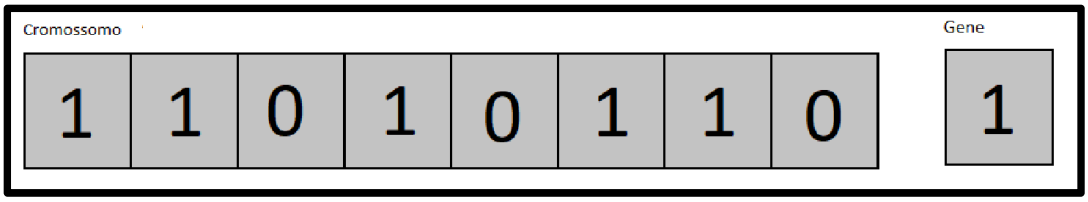


**Figura 4: Representação de cromossomo e gene (Baseado em FERNEDA, 2009)**

Hamawaki (2005) define alguns pontos característicos sobre o funcionamento dos AGs, dentre eles, a funcionalidade de operar em uma população e não em

um indivíduo apenas, no caso, cromossomo; atuação em espaços programados e não diretamente em um meio vazio; e geração aleatória da população inicial.

Os AGs possuem grande eficiência e flexibilidade na busca de soluções, uma vez que não impõem muitas das limitações encontradas nos métodos de busca tradicionais. Eles permitem à exploração e identificação de diferentes qualidades dos objetos pertencentes ao mundo virtual em que estão inseridos (HAMAWAKI, 2005).

Isto quer dizer que é possível trabalhar com AGs nos mais diversos casos, não necessariamente em um cromossomo representado por conjunto de 1 e 0. Pode ser um caso totalmente diferente, como, por exemplo, o personagem de um jogo com diversas características, tal qual o personagem representaria o cromossomo e o gene é representado pelas características, como ataque, defesa, vida, entre outros.

Paliouras, et al. (2001) caracteriza os AGs como algoritmos de aprendizado reforçado (do inglês, *reinforcement learning algorithms*), sendo assim, a performance do sistema é definida por meio de um único valor obtido por uma função de avaliação que atua como pressão ambiental, denominado *Fitness* (HAMAWAKI, 2005). Sendo assim, como conceito evolucionário, é definido que o fitness em cada geração tem que ser melhor que o da geração anterior.

## 3.2 FUNCIONAMENTO

Ilustrado na Figura 5, o algoritmo genético é constituído por etapas que se repetem durante a evolução do sistema até chegar na solução desejada:

(1) Geração da população inicial. Geralmente construída com valores aleatórios para maior diversidade de características;

(2) Serão selecionados par a par cada um eles para a realização do cruzamento entre si.

(3) Posteriormente enfrentarão um processo de mutação, responsável por gerar a próxima geração.

(4) O algoritmo define se a geração atual chegou na melhor solução ou se deve repetir o processo começando novamente pela etapa (2).

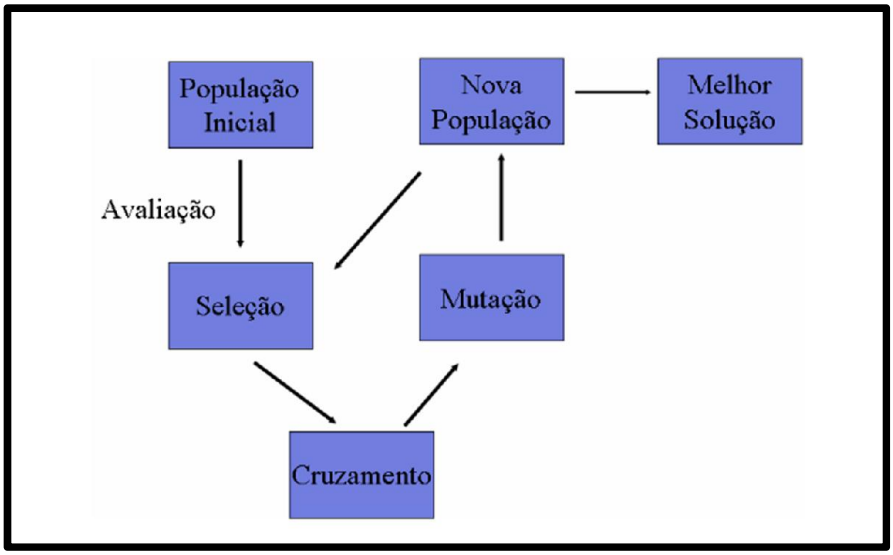


**Figura 5: Ciclo principal dos algoritmos genéticos (In: HAMAWAKI, 2005)**

### 3.2.1 FITNESS

O algoritmo genético tem a responsabilidade de ter que procurar pelos melhores indivíduos contidos em cada geração, por isso, por meio do *Fitness*, é decidido o que deve ou não ser descartado, de tal forma que esse cálculo resulte em valores maiores na geração posterior (Paliouras et al, 2001).

### 3.2.2 CRUZAMENTO

Na etapa de recombinação genética, também conhecida pelo termo em inglês, *crossing-over*, é um processo de cruzamento entre pares de cromossomos, realizando dessa forma a troca de parte das informações de um com o outro, como representado na Figura 4 (HAMAWAKI, 2005).

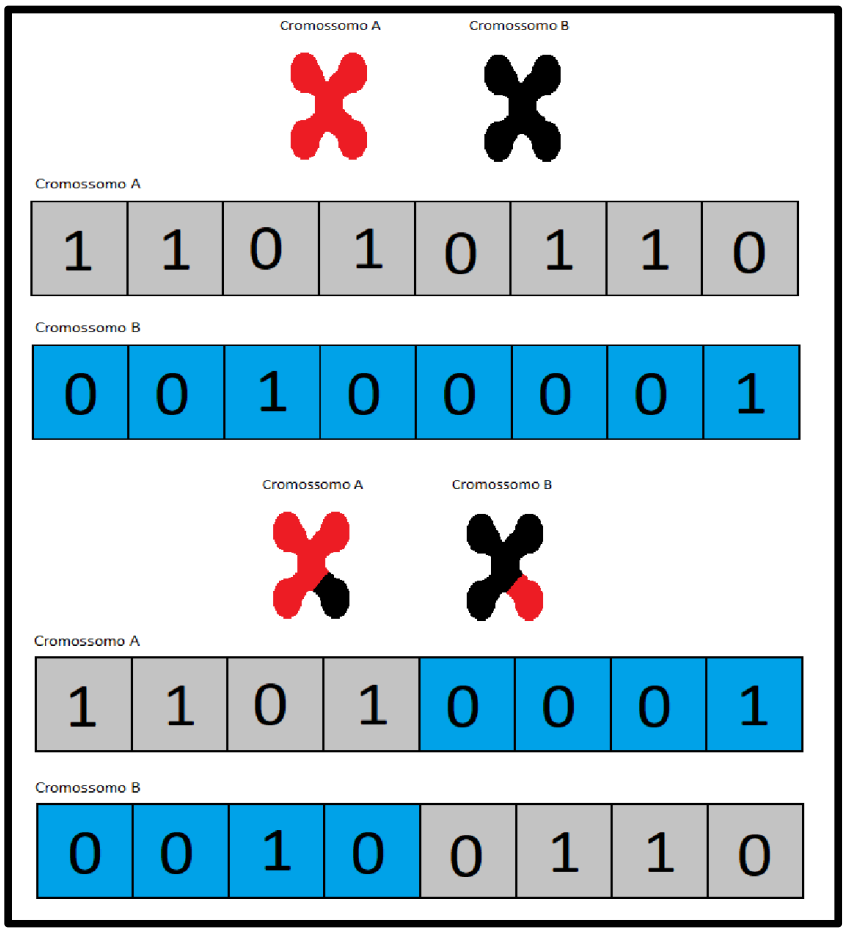


**Figura 6: Recombinação genética entre cromossomos (Baseado em FERNEDA, 2009)**

O cruzamento de um par de cromossomos gera duas possibilidades, a qual apenas uma deve ser escolhida, sendo assim, fica a critério da implementação da aplicação decidir qual cromossomo irá ser descartado, se será escolhido um específico ou aleatório.

## 3.3 CASES E EXEMPLOS DE USO

Abordagens recentes da área de algoritmos genéticos estudam o funcionamento desses algoritmos em jogos além das abordagens de *Path Find*, isto é, algoritmos que, ao longo de sua execução têm como objetivo encontrar soluções para caminhos, como um labirinto em um jogo ou um cenário que forneça elementos para serem coletados. A pesquisa conduzida por Charoenkwan et al. (2010) explora a capacidade desses algoritmos quando aplicados a jogos 3D, de

forma a comparar e avaliar as diferenças de métodos supervisionados e não-supervisionados para Aprendizado de Máquina.

Cole et al. (2004) usam os conceitos de Algoritmos Genéticos (AGs), sob o âmbito de AMs, para realizar experimentos na criação de *bots* no desenvolvimento de jogos, de forma a oferecer um método eficiente para esta situação em jogos de tiro em primeira pessoa (*First Person Shooters*). Isto é realizado por meio da evolução de um conjunto de parâmetros envolvidos no desenvolvimento dos *bots,* que aprendem ao longo de cada seleção.

# 4 – JOGOS DIGITAIS

Jogos digitais são definidos pelo uso da multimídia e interatividade ao oferecer uma atividade divertida por meio de ações e decisões que geram consequências interativas para os usuários (RAMOS, ANASTÁCIO, 2018).

Há anos os jogos digitais vêm conquistando uma imensa quantidade de pessoas por meio da imersão e diversão que o mesmo oferece. De acordo com uma pesquisa levantada pela *Newzoo*, só em 2016, o total arrecadado ao redor do mundo pelos jogos eletrônicos são de aproximadamente 100 bilhões de dólares (NEWZOO, 2016).

A Figura 7 ilustra essa influência, relatando a importância deste mercado.

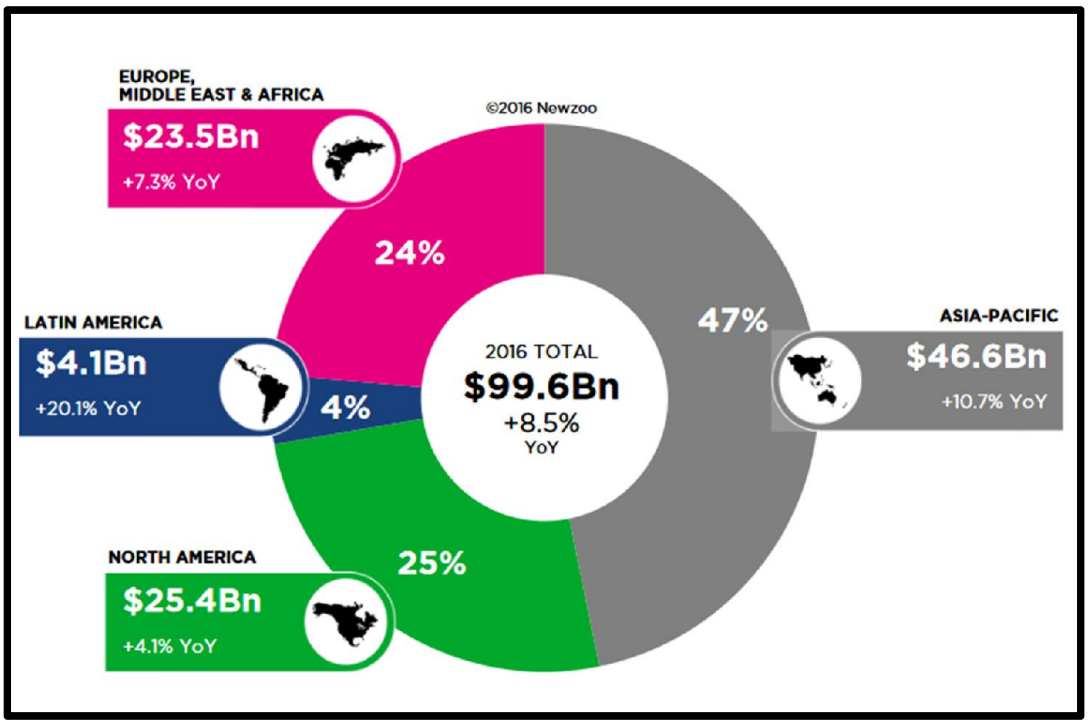


**Figura 7: mercado de jogos ao redor do mundo (In: Newzoo, 2016)**

Contudo, essa mesma influência até um tempo atrás não foi tão predominante na área acadêmica. A realização de pesquisas envolvendo jogos digitais muitas das vezes foi um processo mais complexo devido à grande ausência de

diversidade de referências disponíveis. Rutter e Bryce (2006) apontam a falta de pesquisas acadêmicas ao redor de jogos digitais por autores contemporâneos, enfatizando a ideia de que acadêmicos têm evitado jogos digitais, porém ainda assim definindo um crescimento até o ano de 2004.

Em contrapartida, Bowling et al. (2006) mencionam o crescimento da importância e uso da Inteligência Artificial (IA) para jogos na época, e como essa funcionalidade se tornou essencial para a construção de jogos imersivos e divertidos, destacando como consequência o potencial acadêmico para a realização de pesquisas acadêmicas. Partindo deste ponto, eles destacam a grande oportunidade de trabalhar com Aprendizado de Máquina em jogos, dado a possibilidade criativa da área, seja no uso em jogos desenvolvidos para entretenimento, simulação ou até mesmo a educação.

Para Zook e Riedl (2012) a IA também demonstra em sua essência diversas opções e oportunidades, como a possibilidade de ajustes de dificuldade, fornecendo aos jogadores, por meio de análise de dados de suas jogatinas, um desafio de acordo com suas habilidades e experiências.

A IA em jogos possui relevância em diversas abordagens, muitas das vezes até para imitar o comportamento de humanos por meio da computação cognitiva, como, por exemplo, um jogador resolvendo puzzles (desafios de lógica), passando por diferentes fases de um jogo, ou até usando dados de um jogador para realizar um percurso em jogos de corrida, oferecendo a possibilidade de competir contra “você mesmo” seja diretamente ou indiretamente, como em quem realiza o percurso mais rápido.

## 4.1 CRESCIMENTO DE PESQUISAS ACADÊMICAS EM JOGOS DIGITAIS

Rutter e Bryce (2006) destacam a falta de atenção que os jogos digitais têm recebido pelos acadêmicos até o ano de 2006, pois estes são muitas das vezes retratados como “um meio infantil” ou “meramente banais” sem haver uma base fundamental para apoiar esta questão.

Ainda que essas imagens dos jogos digitais tenham sido prevalentes na época, os autores apontam uma visão positiva em relação ao desenvolvimento de pesquisas apoiadas por pesquisadores que cresceram tendo contato com jogos digitais.

A Figura 8 ilustra o crescimento do número de pesquisas direcionadas a jogos digitais por ano de 1995 até 2004.

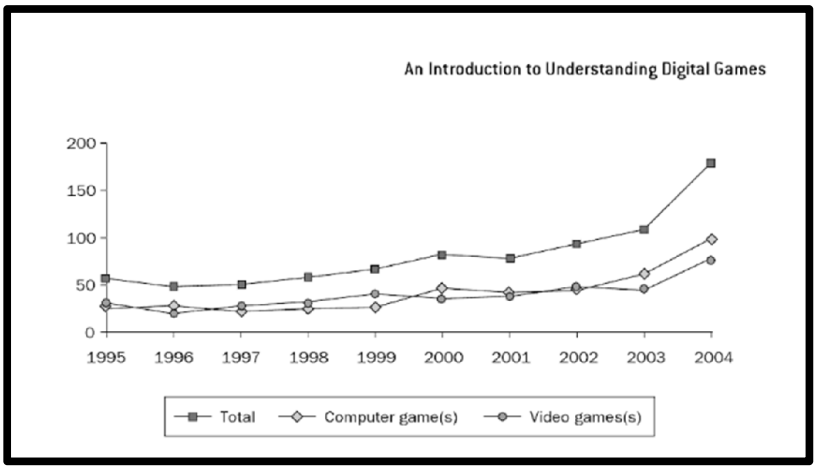


**Figura 8: Número de artigos publicados de 1995 – 2004 (In: RUTTER, BRYCE, 2004)**

## 4.2 INTELIGÊNCIA EM JOGOS

Computação Cognitiva é uma das áreas de estudo de IA que busca a imitação do comportamento do cérebro humano, possuindo assim uma relação forte com o campo de neurociência (WANG et al., 2010).

Esta área faz uso de algoritmos de Aprendizado de Máquina, isto é, Mineração de Dados, Redes Neurais, Processamento de Linguagem Natural e Reconhecimento de Padrões para simular o comportamento e inteligência que seja o mais próximo a encontrada em seres humanos (MODHA et al., 2011).

As aplicações mais encontradas que utilizam os conhecimentos de Computação Cognitiva são os de Redes Neurais, como o *MarI/O* e *IAMDinosaur* (Seção 2.3), bem como o *AlphaGo* da *Google*, o qual recentemente foi capaz de derrotar o melhor jogador de Go do mundo em 2017.

Ainda assim, a computação cognitiva possui um longo caminho para simular o raciocínio do cérebro humano devido a sua complexidade. Já para os jogos, não há total necessidade disto pois basta convencer o jogador que os desafios aparentam cada vez mais realistas, como no jogo *F.E.A.R.*, o qual os inimigos controlados pelo computador realizam decisões diferentes baseados nos dados coletados por cada jogada (KEHOE, 2015).

Dessa forma, além da Computação Cognitiva com Redes Neurais, a IA apresenta outras abordagens para sistemas de decisões, como sistemas baseados em regras, árvores de decisões, máquinas de estados, sistemas adaptativos entre outros que podem simular uma espécie de inteligência (KEHOE, 2015; MILLINGTON, FUNGE, 2009).

No ramo da IA para jogos, a inteligência muitas vezes é representada por personagens com o objetivo de realizar certas ações, como atacar o jogador se “perceberem” que ele está por perto, porém, ela pode ser representada por diversas outras situações, como em desenvolvimento de ambientes dinâmicos, como proposto por Rafacho e Farto (2016) fazendo uso de árvores de decisões.

# 5 – PROPOSTA DE TRABALHO

Este trabalho propõe a utilização da plataforma de desenvolvimento gráfico *Processing* para desenvolvimento de uma simulação de aprendizado por meio de algoritmo genético.

## 5.1 TECNOLOGIA UTILIZADA

### 5.1.1 PROCESSING

*Processing* é uma plataforma que integra sua própria linguagem de programação, ambiente de desenvolvimento e metodologia de ensino em um pacote unificado, mesmo embora a linguagem tenha conexão direta com a linguagem *Java*. A plataforma foi desenvolvida o intuito de ser usada no ensino e aprendizado de programação computacional no contexto de computação gráfica (REAS; FRY, 2007).

A Figura 9 ilustra a interface de desenvolvimento do *Processing* e suas principais funcionalidades.

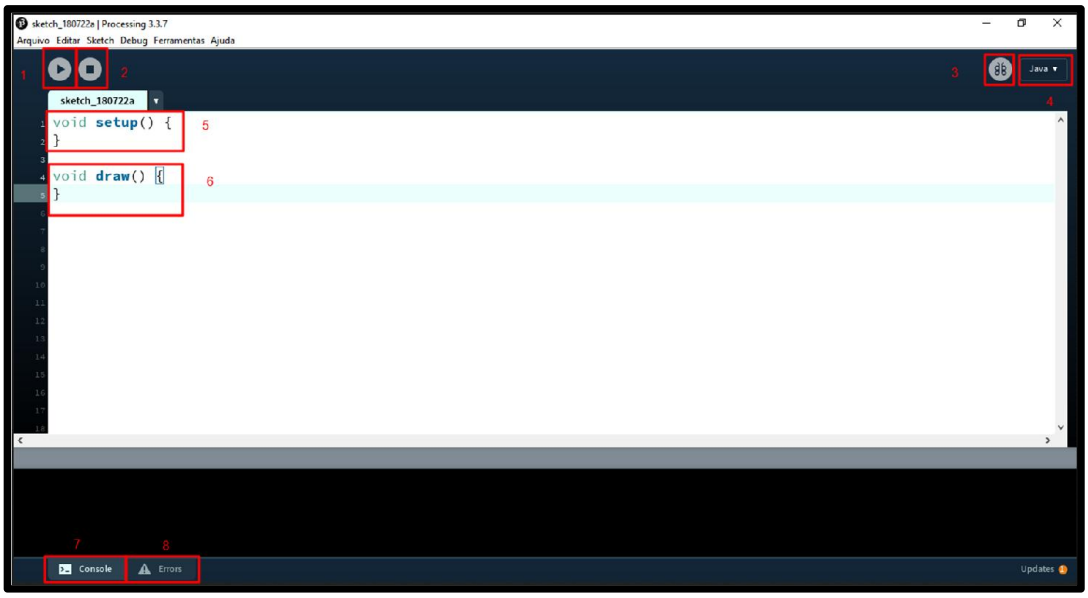

**Figura 9: Interface de ambiente desenvolvimento do *Processing***

(1) **Executar**: Inicializa a execução do programa;

(2) **Parar**: Finaliza a execução do programa;

(3) **Modo de depuração**: Ativa funcionalidades do *Processing* para depuração, como visualização de variáveis de escopo em tempo de execução, pontos de parada e opções para navegar no código quando o programa está parado na sua execução;

(4) **Linguagem**: Escolhe a linguagem de programação permitido pelo ambiente do *Processing*;

(5) **Função *setup*:** Função aonde reside os códigos para serem executados na inicialização do programa;

(6) **Função *draw*:** Função que é chamada quadro a quadro pelo Processing e que irá conter os códigos de atualização;

(7) **Aba console:** Aba de interação com o console;

(8) **Aba de erros:** Aba para visualizar os erros encontrados no projeto.

## 5.2 COMPONENTES

### 5.2.1 GERAÇÃO DO AMBIENTE

Esta etapa do processo constitui a criação e a atualização da simulação como um todo. A proposta deste trabalho é construir um ambiente que contenha diferentes tipos de objetos para o personagem interagir de acordo com a sua posição.

Dessa forma, o personagem, bem como outros objetos deverão ser gerados em posições aleatórias no início do programa para destacar a relevância do aprendizado, pois isto possibilita uma maior evidência dos resultados sobre como o personagem se adapta de acordo com cada geração.

Este trabalho também propõe gerar mais que um personagem para a mesma geração, contanto que cada um seja diferente na questão dos caminhos que irão percorrer. Sendo assim, o ambiente gerado inicialmente, bem como cada objeto pertencente a ele, é replicado para cada personagem gerado de forma a manter o mesmo contexto para cada um.

### 5.2.2 OBJETOS DO AMBIENTE

Serão construídos um conjunto de pelo menos 3 tipos de objetos, sendo (i) o Personagem, (ii) os Inimigos e (iii) os Coletáveis.

A proposta consiste em criar todos os objetos em posições aleatórias com um número definido de inimigos e de coletáveis. Assim, quanto mais inimigos, maior será a dificuldade de gerar bons percursos a serem tomados pelo personagem, o qual, consequentemente, deve prolongar o processo de aprendizado para um resultado adequado.

### 5.2.3 INTERFACE DE USUÁRIO E OUTRAS FUNCIONALIDADES

O seguinte trabalho também propõe uma interface de usuário com algumas funcionalidades básicas para possibilitar um melhor acompanhamento desse aprendizado, bem como algumas funções extras permitidas pelo programa. A função da interface é fornecer ao usuário melhor visualização das alterações do programa que estão envolvidos neste processo como um todo além de o mesmo poder administrar algumas opções disponíveis para controlar a simulação.

Portanto, o programa estabelece a possibilidade de aumentar ou diminuir a velocidade de atualização de movimentos do personagem, pausar a simulação, habilitar ou desabilitar a simulação automática e até mesmo possibilitar um desenho mais detalhado para visualizar claramente as posições da tabela de objetos do programa.

## 5.3 PROCESSO DE APRENDIZAGEM

Para realizar a aprendizagem, a proposta deste trabalho é a construção de um conjunto de passos que o personagem irá percorrer e, dessa forma, possibilitar a análise de cada geração com os melhores caminhos definidos para cada personagem por meio de seu arranjo de movimentos. Posteriormente, conforme o processo de aprendizagem for sendo realizado, melhores rotas irão ser tomadas pelo personagem considerando os obstáculos do ambiente.

Na etapa de aprendizagem, como definido pelo algoritmo genético, o *fitness* é o responsável por representar e avaliar a qualidade do aprendizado dos indivíduos, sendo no caso deste trabalho, os arranjos de movimento do personagem de cada mapa.

Dessa forma, ao colidir com um coletável, o personagem deverá receber pontos positivos em seu *fitness*, pois este representa um fator bom para o caminho escolhido, porém se colidir com um inimigo, o mesmo sofrerá uma perda de pontos; agora se o personagem passar por um ponto vazio, ou seja, sem coletável ou inimigo, então ele irá receber apenas um ponto positivo.

O foco deste trabalho é induzir o personagem a caminhar pelos melhores pontos disponíveis em uma quantidade N de passos pré-definidos pelo programa, logo, um caminho adequado a ser escolhido conterá um maior número de colisões com coletáveis e pontos vazios do mapa.

# 6 – DESENVOLVIMENTO DO TRABALHO

Neste capítulo serão detalhadas as etapas utilizadas para a conclusão do desenvolvimento da implementação do programa de simulação, bem como algumas estratégias definidas para a realização de certas funcionalidades específicas.

## 6.1 DEFINIÇÃO DO PROBLEMA

A definição do problema envolveu um estudo sobre o funcionamento dos algoritmos genéticos e como estes são utilizados para o aprimoramento de soluções de um programa, bem como adquirir conhecimento sobre a utilização da plataforma *Processing*, analisando sua estrutura e abordagens para desenvolvimento.

O foco para a realização deste projeto se concentrou em especificar as funcionalidades a serem envolvidas nas etapas do Algoritmo Genético, ilustrados na Figura 5, e como cada uma delas irão se comportar.

Nesta simulação, foi proposto para a implementação do AG, a geração de um ambiente a ser replicado para cada personagem gerado, especialmente para realizar o cruzamento de cromossomos. Nesta implementação, foi definida uma quantidade total de 10 personagens por geração, ou seja, o ambiente gerado inicialmente, bem como os inimigos e coletáveis pertencentes a ele, será replicado 10 vezes.

No início do desenvolvimento desta implementação, foi definido que o arranjo de movimentos de um personagem será o cromossomo no contexto do AG, assim, os genes são representados por cada passo que o mesmo terá que realizar. Portanto, o arranjo é definido como um conjunto de inteiros, que são traduzidos em vetores em sua utilização, como representado na Tabela 2.

| Valor de um passo | | | | Tradução em vetores | | |
|---|---|---|---|---|---|---|
| 7 | 8 | 9 | | ↖ | ⬆ | ↗ |
| 4 | 5 | 6 | = | ⬅ | Neutro | ➡ |
| 1 | 2 | 3 | | ↙ | ⬇ | ↘ |

**Tabela 2: Tradução de um passo do arranjo de movimentos**

Os passos são representados por valores inteiros de 1 a 9, gerados aleatoriamente quando o programa é iniciado. Se o personagem na realização de um movimento possuir um valor de passo equivalente a 5, como representado pela tabela, o mesmo não irá se movimentar, pois equivale a neutro ao ser traduzido para vetor, porém se o valor for qualquer outro de 1 a 9, então ele irá realizar um movimento para uma casa vizinha.

A movimentação de passo a passo gerado inicialmente é realizada apenas para as posições vizinhas, pois cada movimento possui uma magnitude de valor 1, ambos para o eixo X e eixo Y. Sendo assim, o personagem não poderá andar para uma direção por dois quadrados ou mais de uma vez só. Neste contexto, se o mesmo for caminhar dois ou mais quadrados, o movimento deverá ser realizado em duas ou mais repetições para a mesma direção. Segue a exemplificação deste processo na Figura 10.

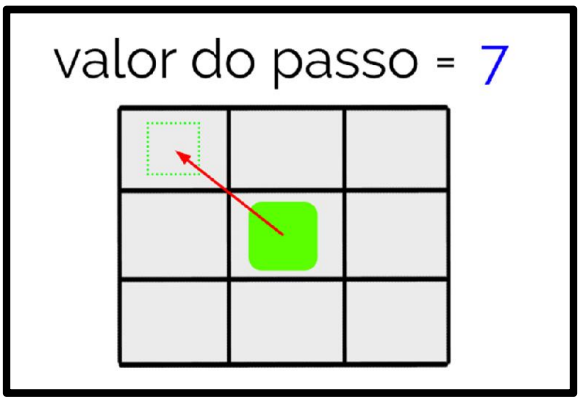


**Figura 10: Exemplo de um passo a ser realizado pelo personagem**

Como dito anteriormente e ilustrado na Figura 10, um passo é um valor inteiro, que neste exemplo equivale a 7, dessa forma, seguindo os valores da Tabela 2, esse valor é traduzido para a direção superior esquerda, o qual também será a próxima posição a ser assumida pelo personagem em seu respectivo mapa.

Por fim, foi possível criar uma tabela de posição para visualização do mapa atual a ser exibido e também os objetos que o mesmo contém.

Além da tabela de simulação, foi desenvolvido um gráfico para visualização da evolução realizada pela simulação, onde o eixo X é respectivo à geração atual da simulação, e o eixo Y, a pontuação máxima realizada nessa mesma geração, que no caso é representado pelo *fitness*.

A figura 11 ilustra a aplicação de simulação desenvolvida e a sua evolução conquistada até a 92ª geração.

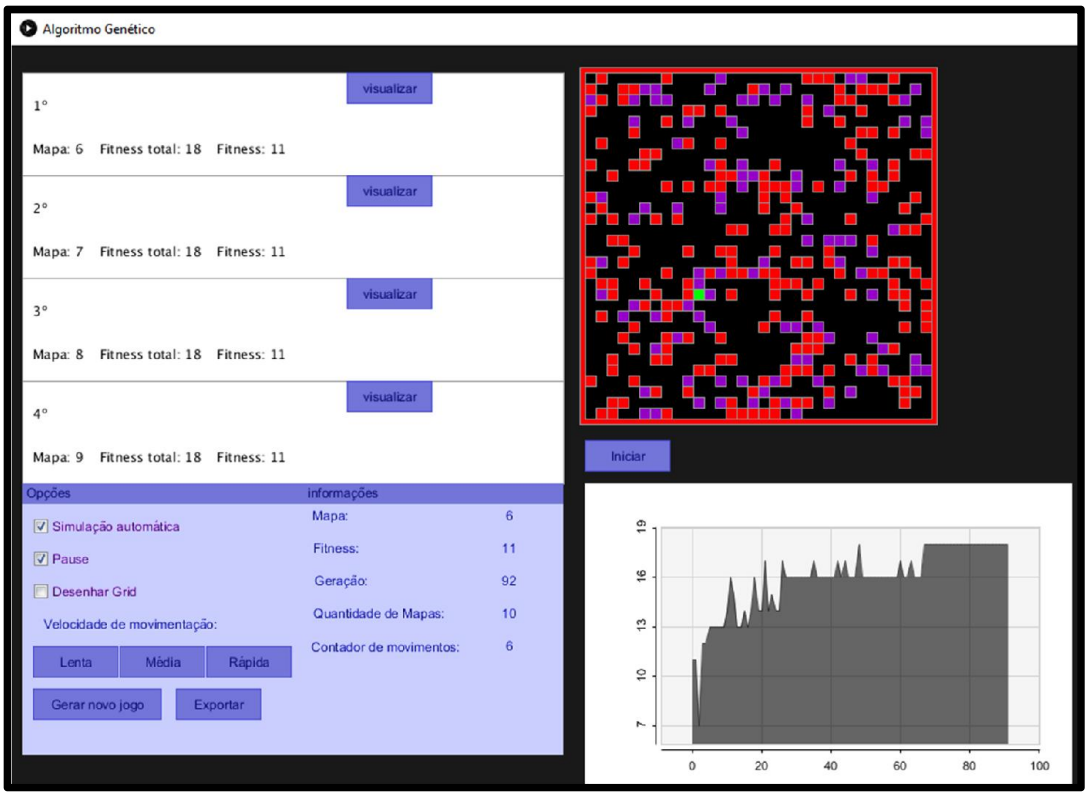


**Figura 11: Tela da aplicação de simulação**

## 6.2 CONFIGURAÇÃO INICIAL

Algumas funcionalidades principais para realizar as configurações da simulação também foram desenvolvidas. A princípio, para realizar estas configurações, as funções devem ser chamadas em uma função principal do *Processing*, chamado de *setup*. O *setup* possui o código que será lido na inicialização da aplicação do *Processing*. Nele pode conter códigos para configuração do tamanho de janela, como permitir o ajuste de tamanho ou algumas outras funcionalidades específicas, bem como alguns comportamentos da própria aplicação definidos em sua inicialização.

Portanto, é no *setup* em que é realizado a função para geração inicial do ambiente na simulação, o qual é responsável por instanciar os objetos, bem como todos os outros componentes. De forma mais detalhada, é gerado o

ambiente, o qual também irá gerar uma quantidade definida de objetos do tipo coletável e do tipo inimigo, cada um com a sua posição aleatoriamente gerada, bem como o personagem, e seu conjunto de movimentos. Por fim, estas informações são passadas como parâmetros para possibilitar a replicação do ambiente por meio de uma cópia indireta.

A cópia indireta é realizada ao obter dados primitivos para serem replicados em outros objetos. Isso acontece para contornar a característica do *Java de* funcionar por meio de referência de objetos, ou seja, se um objeto receber o valor de outro objeto, o mesmo irá possuir uma referência direta para seus valores, então toda vez que um valor for alterado, o mesmo será replicado para todos os outros objetos que possuem a mesma referência, o que não é o objetivo desta implementação.

### 6.2.1 INTERFACE

Além da criação do ambiente e dos objetos, também é chamado a função de criação da interface gráfica, criada por meio da biblioteca G4P disponível pela plataforma, o qual define os principais componentes de interface de interação com o usuário. A biblioteca *G4P* fornece uma variedade de componentes de interface para serem usados na aplicação.

Embora não seja tão robusta quanto os componentes de um programa para criar aplicativos *Desktop* como o *Swing* do *Java* ou o *Windows Forms* do C#, ou até mesmo o QT do C++, o G4P para o uso em aplicações feitas no *Processing* ela acaba sendo mais que o suficiente para seus propósitos e principalmente para os propósitos deste trabalho.

Além do *G4P*, a plataforma também fornece o *G4P GUI builder*, um ambiente para visualização e configuração dos componentes disponibilizados pela biblioteca. Segue a Figura 12 para visualização desta funcionalidade.

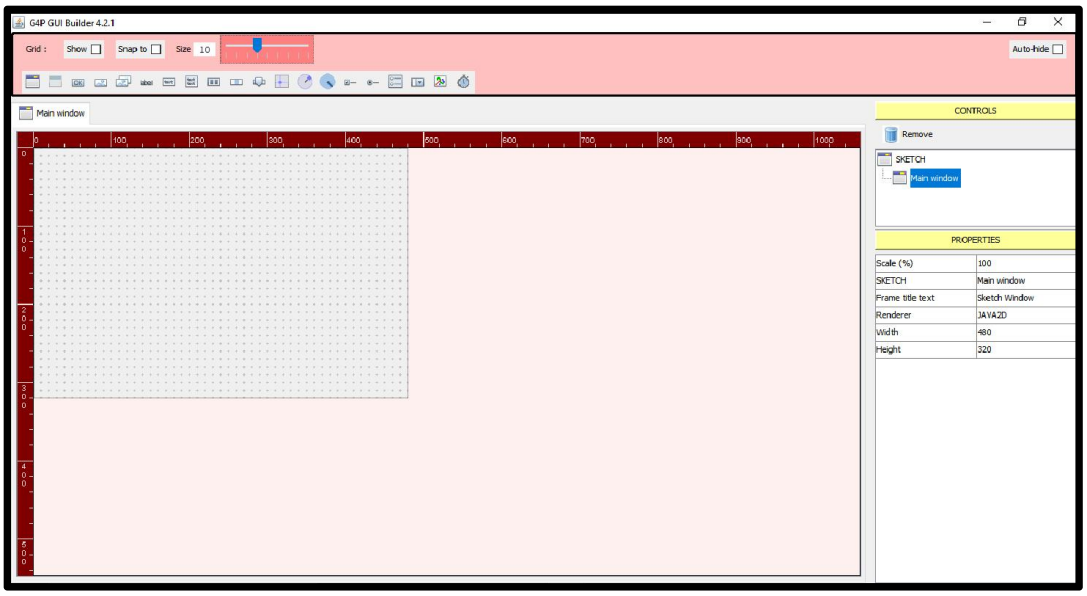


**Figura 12: Interface do G4P GUI *Builder***

Como foi utilizado uma ferramenta externa para realizar a criação da interface, quando definimos quais componentes são aplicados ao nosso programa, um novo arquivo chamado “gui” é gerado no projeto para conter a geração de todos os componentes e seus eventos pela codificação, como por exemplo, eventos de clique.

Em seguida, para complementar a interface do programa, foi desenvolvida uma função responsável pela configuração e customização desses componentes, definindo posições, aparência e regras de ajustes para limitar o que pode ser feito ou não pelo usuário.

Através do *G4P* foram definidas algumas funcionalidades para controle da simulação, o qual o usuário pode ir interagindo a fim de compreender melhor ou de ajustar a simulação à sua preferência. Assim, foi obtido a seguinte tela, ilustrada pela Figura 13.


**Figura 13: Componente gerado pelo *G4P* para opções de controle da simulação**

A tela dispõe algumas funcionalidades, como ativar ou desativar a simulação automática, o qual é responsável por atualizar o mapa, gerar a próxima geração e assim sucessivamente sem que o usuário tenha que realizar estes dois procedimentos; a opção de pausar a simulação; desenhar *Grid*, o qual exibe as linhas e colunas da tabela da simulação; e por fim, a opção de escolher de 3 alternativas, de lento, médio e rápido, para a velocidade dos movimentos realizados pelo personagem.

Além dessas funcionalidades, foi implementado a possibilidade de gerar um novo ambiente com outras posições para os objetos, e também a possibilidade de exportar os dados atuais da simulação para possibilitar a leitura das evoluções realizadas.

No componente para exibição de informações, ilustrado na Figura 14, não há opções de interação para o usuário, pois este componente contém todas as informações gerais da simulação, ou seja, o mapa que está sendo exibido no momento, qual o seu *fitness* atual, qual a geração, quantidade de mapas totais na simulação e também qual o movimento atual do personagem.

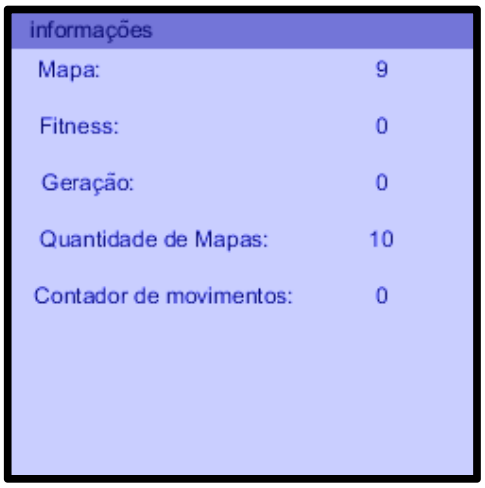


**Figura 14: Componente gerado pelo *G4P* para exibição de informações gerais da simulação**

Além do uso do *G4P* para uma interface interativa, utilizando apenas as funções nativas do *Processing*, foi construído uma tabela para ilustrar os 4 mapas com os melhores *fitnesses* ordenadamente, o qual foi a principal motivo para o uso do cálculo de prévia do *fitness*, detalhado na Subseção 6.3.1, a fim de facilitar este processo. A tabela também ilustra qual é o mapa que contém aquele respectivo *fitness* e qual o seu valor atual, somado a cada passo realizado pelo personagem. Segue a representação na Figura 15 de como estas funcionalidades foram implementadas.


**Figura 15: Tabela de ilustração dos 4 mapas com o melhor fitness**

Na imagem é possível visualizar algumas informações destacadas, o item 1 representa a ordem de cada mapa pelo critério de *fitness*, o 2 informa o índice do mapa, o 3 é a previsão do fitness para aquele respectivo mapa da geração atual, o item 4 é o fitness atual, isto é, até o movimento atual do personagem, e o 5 são botões para visualizar o mapa escolhido na simulação.

## 6.3 ATUALIZAÇÃO DO PROGRAMA

No *Processing*, a função *draw* é chamada quadro a quadro pela plataforma e é responsável por conter ambos os códigos de atualização e desenho de uma aplicação. Neste trabalho, está função conterá todo o processo de controle a ser realizada pela simulação, bem como as etapas de aprendizado do algoritmo genético.

### 6.3.1 ALGORITMO GENÉTICO

A realização do algoritmo genético começa pelo cálculo do fitness, o qual é realizado durante a movimentação do personagem da geração atual para cada passo tomado pelo mesmo.

Embora o cálculo do fitness seja realizado durante a movimentação do personagem, também foi desenvolvido um cálculo para prever o *fitness* do indivíduo da geração atual, fornecendo uma visão de início sobre a qualidade do caminho a ser percorrido.

Ao terminar a movimentação do personagem, isto é, realizar todos os passos definidos no arranjo de movimentos de cada um, o algoritmo se encarrega de escolher da lista de mapas os 4 melhores cromossomos definidos pelo *fitness* para realizar o cruzamento de um com os outros a fim de gerar um total de 6 mapas. Assim, o restante para completar um total de 10 mapas é reutilizado desses 4 melhores sem haver modificações em suas estruturas.

Escolhidos os mapas para a próxima geração, o algoritmo realiza de mapa em mapa o processo de mutação, em que se escolhe um gene aleatório, ou seja, um dos passos do arranjo de movimentos do personagem, e altera seu valor aleatoriamente.

Apesar desta pesquisa optar por um baixo índice de mutação, isto é, apenas um dos genes sofre alteração por geração em cada mapa, nada impede de aumentar este valor a fim de obter uma maior variedade nos resultados.

Com esse processo finalizado, os mapas foram definidos e vão para a próxima geração, sendo repetido todo esse processo do início ao fim, obtendo-se um aprimoramento nos resultados conforme o tempo, até atingir uma condição de parada, o qual nesta pesquisa foi deixado para critério próprio do usuário que acompanha a simulação.

# 7 – CONCLUSÃO

Este projeto teve como proposta desenvolver estudos teóricos buscando entender e adquirir conhecimento necessários sobre o uso de Algoritmos Genéticos. Por meio do conhecimento adquirido, também foi proposto o desenvolvimento de uma simulação de jogo em que se busca aprender sobre o ambiente para oferecer melhores soluções sobre um dado objetivo.

Foi observado que o Algoritmo Genético atua por meio da seleção natural, portanto a solução encontrada pode não ser realmente a melhor, porém é a que foi encontrada com melhor resultado até um dado momento.

Além disso, é importante considerar a etapa de mutação do Algoritmo Genético, pois foi possível perceber que quanto maior a taxa de mutação definida, maior será a variedade da solução construída em um certo tempo.

Como solução para situações específicas como a deste trabalho, no qual a intenção da implementação foi encontrar caminhos mais adequados para que o personagem continue “vivo” para a próxima geração, a implementação dos conceitos estudados foi o suficiente, sendo notável a evolução dentro de certas condições impostas pelo programa.

Dado essas informações obtidas pelos resultados do simulador, é possível dizer que o uso de Algoritmos Genéticos podem apresentar soluções otimizadas para certos processos, porém dependendo da complexidade das ações e funcionalidades a serem utilizadas, o seu uso pode levar um bom tempo até poder apresentar uma solução desejável.

Também é possível destacar que os resultados atingidos pela aplicação servem de referência não apenas para a área de jogos, mas também para outros contextos que possam e/ou necessitam fazer uso de uma solução mais adequada conforme o tempo e uso de uma funcionalidade.

## 7.1 TRABALHOS FUTUROS

Como trabalhos futuros, esta pesquisa possibilita uma diversidade de opções para a atual simulação desenvolvida, além de possíveis questões a serem explorados para pesquisas futuras. Os autores deste trabalho propõem:

- Acrescentar objetos diferentes que impactam de forma diferente na simulação, a fim de oferecer maior variedade de caminhos para a simulação;
- Oferecer a possibilidade de o usuário “caminhar” pelo arranjo de movimentos de cada personagem, dentre os movimentos já definidos por cada indivíduo de uma dada geração;
- Implementar destruição de objetos quando estes colidem com o personagem, a fim de possibilitar um caminho mais “aberto” no ambiente a ser realizado pelo personagem;
- Desenvolver uma abordagem similar utilizando outras técnicas de Inteligência Artificial a fim de comparar com este trabalho, como Redes Neurais, Árvores de Decisões ou até mesmo Algoritmos Adaptativos.

# REFERÊNCIAS


AMAZON. **Inteligência artificial na AWS**. Disponível em <https://aws.amazon.com/pt/amazon-ai/>. Acesso em: 27 nov. 2017.

ARON, Ben. **Machine Learning – transformando a sociedade**. Disponível em <https://pt.linkedin.com/pulse/machine-learning-transformando-sociedade-ben-aron>. Acesso em: 27 nov. 2017.

BELL, Jason. **Machine Learning**: Hands-On for Developers and Technical Professionals. Indianapolis, Indiana: John Wiley & Sons, Inc., 2015. 371 p.

BOWLING, Michael; FÜRNKRANZ, Johannes; GRAEPEL, Thore; MUSICK, Ron. Machine learning and games. **Machine learning**, v. 63, n. 3, p. 211-215, 2006.

CHAPELLE, Olivier; SCHÖLKOPF, Bernhard; ZIEN, Alexander. **Semi-supervised learning:** 1. Estados Unidos: MIT Press, 2006.

CHAROENKWAN, Phasit; FANG, Shih-Wei; WONG, Sai-Keung. **A study on genetic algorithm and neural network for implementing mini-games**. In: Technologies and Applications of Artificial Intelligence (TAAI), 2010 International Conference on. IEEE, 2010. p. 158-165.

COLE, Nicholas; LOUIS, Sushil J.; MILES, Chris. **Using a genetic algorithm to tune first-person shooter bots**. In: Evolutionary Computation, 2004. CEC2004. Congress on. IEEE, 2004. p. 139-145.

FERNEDA, Edberto. **Aplicando algoritmos genéticos na recuperação de informação.** DataGramaZero, Rio de Janeiro, v. 10, n. 1, p. 0-1001, 2009.

FORTIN, Félix-Antoine et al. DEAP: Evolutionary algorithms made easy. **Journal of Machine Learning Research**, v. 13, n. Jul, p. 2171-2175, 2012.

HAMAWAKI, Cristiane Divina Lemes. **Geração automática de grade horária usando algoritmos genéticos: o caso da Faculdade de Engenharia Elétrica da UFU**. 2005.

KEHOE, Donald. **Designing Artificial Intelligence for Games**. Disponível em <https://software.intel.com/en-us/articles/designing-artificial-intelligence-for-games-part-1>. Acesso em: 23 mar. 2018.

KHOO, Aaron; ZUBEK, Robert. **Applying inexpensive AI techniques to computer games**. IEEE Intelligent Systems, v. 17, n. 4, p. 48-53, 2002.

MITCHELL, Melanie. **An introduction to genetic algorithms.** MIT press, 1998.

MILLINGTON, Ian; FUNGE, John. **Artificial intelligence for games**. CRC Press, 2009.

MODHA, Dharmendra S; ANANTHANARAYANAN, Rajagopal; ESSER, Steven K; NDIRANGO, Anthony; SHERBONDY, Anthony J; SINGH, Raghavendra. **Cognitive computing. Communications of the ACM**, v. 54, n. 8, p. 62-71, 2011.

MOHRI, Mehryar; ROSTAMIZADEH, Afshin; TALWALKAR, Ameet. **Foundations of machine learning**. MIT press, 2012.

MONARD, Maria Carolina; BARANAUSKAS, José Augusto. **Conceitos sobre aprendizado de máquina. Sistemas inteligentes-Fundamentos e aplicações**, v. 1, n. 1, p. 32, 2003.

NEWZOO. **2016 Global Games Market Report: An Overview of Trends & Insights.** Disponível em <https://cdn2.hubspot.net/hubfs/700740/Reports/Newzoo_Free_2016_Global_Games_Market_Report.pdf>. Acesso em: 01 Jul. 2018.

PALIOURAS, Georgios; KARKALETSIS, Vangelis; SPYROPOULOS, Constantine D. (Ed.). **Machine learning and its applications: Advanced Lectures**. Springer, 2003.

PEDREGOSA, Fabian; VAROQUAUX, Gaël; GRAMFORT, Alexandre; MICHEL, Vincent; THIRION, Bertrand; GRISEL, Olivier; BLONDEL, Mathieu; PRETTENHOFER, Peter; WEISS, Ron; DUBOURG, Vincent; VANDERPLAS, Jake; PASSOS, Alexandre; COURNAPEAU, David; BRUCHER, Matthieu; PERROT, Matthieu; DUCHESNAY, Édouard. **Scikit-learn: Machine learning in Python.** Journal of machine learning research, v. 12, n. Oct, p. 2825-2830, 2011.

RAMOS, Daniela Karine; ANASTÁCIO, Bruna Santana. **Cognitive skills and the use of digital games in school: The perception of children**. Educação Unisinos, v. 22, n. 2, p. 214, 2018.

RAFACHO, Addam C. P.; FARTO, Guilherme de Cleva. **Abordagens de Decision Trees no Contexto de Geração de Regras para Jogos com Unity 3D Engine.** 2016. 15. Fundação Educacional do Município de Assis, São Paulo, Assis, 2016.

REAS, Casey; FRY, Ben. **Processing: a programming handbook for visual designers and artists.** Mit Press, 2007.

RIECHMANN, Thomas. **Genetic algorithm learning and evolutionary games**. Journal of Economic Dynamics & Control, Hanôver, 12 jun. 2000. p. 1019.

RUSSEL, Stuart; NORVIG, Peter. **Artificial Intelligence**: A Modern Approach. 3 ed. Nova Jersey, Estados Unidos: Pearson Education, Inc., 2010.

RUTTER, Jason; BRYCE, Jo (Ed.). **Understanding digital games.** Sage, 2006.

SEIDEL, Adam. **Youtube**. 27 dez. 2015. 31min22s. Disponível em <https://www.youtube.com/watch?v=P7XHzqZjXQs&t=4s>. Acesso em: 19 mar. 2018.

SETHBLING. **Youtube**. 13 jun. 2015. 5min57s Disponível em <https://www.youtube.com/watch?v=qv6UVOQ0F44&t=58s>. Acesso em: 19 mar. 2018.

SHALEV-SHWARTZ, Shai; BEN-DAVID, Shai. **Understanding Machine Learning: From Theory to Algorithms**. 1. ed. Nova Iorque: Cambridge University Press, 2014.

WALT, Stéfan van der; COLBERT, S. Chris; VAROQUAUX, Gael. **The NumPy array: a structure for efficient numerical computation. Computing in Science & Engineering**, v. 13, n. 2, p. 22-30, 2011.

WANG, Yingxu; ZHANG, Du; KINSNER, Witold (Ed.). **Advances in cognitive informatics and cognitive computing**. Springer, 2010.

WATKINS, Christopher. **Machine Learning is Everywhere: Netflix, Personalized Medicine, and Fraud Prevention**. Disponível em <https://blog.udacity.com/2016/06/machine-learning-everywhere-netflix-personalized-medicine-fraud-prevention.html>. Acesso em: 27 nov. 2017.

WHITLEY, Darrell. **A genetic algorithm tutorial**. Statistics and computing, v. 4, n. 2, p. 65-85, 1994.

WITTEN, Ian H; FRANK, Eibe; HALL, Mark A.; PAL, Cristopher J. **Data Mining: Practical machine learning tools and techniques.** Morgan Kaufmann, 2016.