**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E
TECNOLOGIA DO SUL DE MINAS GERAIS
CÂMPUS MUZAMBINHO
Curso de Ciência da Computação**

Diego Pereira da Penha

**Aplicação de um Algoritmo Genético a NPCs de um jogo,
para otimização indireta de estratégias**

Muzambinho

2017

Diego Pereira da Penha

**Aplicação de um Algoritmo Genético a NPCs de um jogo,
para otimização indireta de estratégias**

Trabalho de Conclusão de Curso apresentado ao
Curso de Ciência da Computação, do Instituto
Federal de Educação Ciência e Tecnologia do Sul
de Minas Gerais - Câmpus Muzambinho, como
requisito parcial à obtenção do título de Bacharel
em Ciência da Computação.

Orientador: Diego Saqui

Coorientador: Ricardo José Martins

Muzambinho

2017

COMISSÃO EXAMINADORA

Prof 1.: Diego Saqui (Orientador)

Prof 2.: Ricardo José Martins (Coorientador)

Prof 3.: Jean Menossi

Prof 4.: Kleber de Oliveira Andrade

Muzambinho, ____ de _____ de 2017

PENHA, Diego Pereira. Aplicação de um Algoritmo Genético a NPCs de um jogo, para otimização indireta de estratégias. 2017. 47p. Trabalho de Conclusão de Curso (Curso Ciência da Computação) – Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – Câmpus Muzambinho, Muzambinho, 2017.

RESUMO

Em jogos eletrônicos, uma das principais características para um jogo divertido e imersivo é a Inteligência Artificial (IA). Por muitos anos, esse componente foi pouco explorado pelos desenvolvedores, que concentram a maioria dos seus esforços em avanços gráficos. Alguns autores têm concluído que jogadores mais avançados e mais experientes exigem IAs mais elaboradas para que se tenha um jogo atraente e interessante. Baseado nessa necessidade, este trabalho foca em analisar a eficiência de um Algoritmo Genético (AG) em um jogo eletrônico e comparar os resultados com uma máquina de estados finitos. Um AG é baseado em otimização evolutiva, otimizando resultados com base em uma função de aptidão, que leva em conta certos fatores do jogo. Estes algoritmos permitem organizar os NPC's (personagens não jogáveis) em melhor ou pior aptidão e selecionam os melhores, o que faz dele um tipo de IA não-linear e mais avançada. Essa característica é diferente de uma máquina de estados finitos, que apenas executa ações previamente programadas sem alterações ou otimizações adaptativas na execução, fazendo o jogo completamente previsível neste aspecto. Para isso será considerado um jogo simples onde ambas estratégias serão aplicadas em momentos diferentes. Para avaliar os resultados, o jogo, nas diferentes condições, será testado por jogadores reais que indicarão a eficiência de cada método. Os resultados esperados são os que apontem o Algoritmo Genético como mais eficiente.

Palavras-chave: Adaptativo. Otimização. Inteligência Artificial. Jogos Eletrônicos.

PENHA, Diego Pereira. Aplicação de um Algoritmo Genético a NPCs de um jogo, para otimização indireta de estratégias. 2017. 47p. Final paper (Computing Science Graduation) – Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais – Câmpus Muzambinho, Muzambinho, 2017.

ABSTRACT

In digital games, one of the main features that make the games fun and engaging is Artificial Intelligence. For many years, this feature was little explored by the developers, who concentrated most of their efforts on graphical advances. Some authors have come to conclusion that more experienced and older players often require more advanced artificial intelligence to find a game attractive. Based on this need, this work aims to analyze the efficiency of a Genetic Algorithm in a digital game and compare its results with a finite state machine. GA, or Genetic Algorithm, is an algorithm that bases its structure on evolutionary programming, optimizing itself based on a fitness value, sorting results and selecting the best NPCs (*non-player characters*), which makes it a non-linear and more advanced type of artificial intelligence, unlike a finite state machine, that only does what has been previously programmed, not optimizing itself in any way, making the game fully predictable. For this is considered a simple game where both strategies will be applied at different times. To evaluate the results, the game in different conditions will be tested by real players who must evaluate the efficiency of each method. The results are expected to indicate greater efficiency for the Genetic Algorithm.

Keywords: Adaptive. Optimization. Artificial Intelligence. Gaming.

LISTA DE ILUSTRAÇÕES

Figura 1. Tipos de cromossomos em Algoritmos Genéticos	18
Figura 2. Cruzamento em um ponto	20
Figura 3. Jogo no modo Algoritmo Genético	22
Figura 4. Jogo no modo Máquina de Estados Finitos	23
Figura 5. Componentes do jogo	23
Figura 6. Exemplo de cromossomo	27
Figura 7. Algoritmo genético aplicado	30
Figura 8. Preferência de modo	33
Figura 9. Modo mais imersivo	34
Figura 10. Nível de imersão da Máquina de Estados Finitos	34
Figura 11. Nível de imersão do Algoritmo Genético	35
Figura 12. Modo mais desafiador	35
Figura 13. Desafio do modo Algoritmo Genético	36
Figura 14. Desafio do modo Máquina de Estados Finitos	36
Figura 15. Modo mais divertido	37
Figura 16. Diversão do modo Máquina de Estados Finitos	38
Figura 17. Diversão do modo Algoritmo Genético	38
Figura 18. Modo mais difícil	39
Figura 19. Dificuldade do modo Máquina de Estados Finitos	39
Figura 20. Dificuldade do modo Algoritmo Genético	40
Figura 21. Preferência de modo por sexo	40

LISTA DE QUADROS

Quadro 1. Tempos obtidos no modo Máquina de Estados Finitos	41
Quadro 2. Tempos obtidos no modo Algoritmo Genético	41

SUMÁRIO

1. INTRODUÇÃO	9
1.1. Contextualização e Motivação	9
1.2. Objetivos	10
1.2.1 Objetivo Geral	10
1.2.2 Objetivos Específicos	10
2. REVISÃO DE LITERATURA	12
2.1. Jogos Eletrônicos	12
2.2. Inteligência Artificial para Jogos Digitais	14
2.2.1. Máquinas de Estado Finito	15
2.2.2. Scripting	15
2.2.3. Lógica Nebulosa (Fuzzy)	15
2.2.4. Algoritmos Evolutivos	16
2.2.4.1. Algoritmos Genéticos	17
3. METODOLOGIA	21
3.1. Materiais e Ferramentas	21
3.2. Jogo Desenvolvido	22
3.3. Aplicação dos algoritmos propostos	25
3.3.1 Aplicação da Máquina de Estados Finitos	25
3.3.2 Aplicação do Algoritmo Genético	26
3.4. Teste aplicado	30
3.5. Avaliação do teste	31
4. RESULTADOS E DISCUSSÃO	33
5. CONSIDERAÇÕES FINAIS	43
6. REFERÊNCIAS BIBLIOGRÁFICAS	44

1. INTRODUÇÃO

1.1. Contextualização e Motivação

A evolução nos jogos digitais promove mudanças na sociedade em geral, por meio de novas tecnologias tanto em *hardware*, quanto em *software*. Dentre as áreas abordadas têm-se, por exemplo, programação, música, efeitos sonoros, redes, Inteligência Artificial (IA), matemática e física. Em função da diversidade de disciplinas envolvidas, os jogos se mostram como um bom ambiente para a pesquisa de novas tecnologias para estas áreas da computação (ANDRADE; SILVA; CROCOMO, 2009).

Entre áreas fundamentais para um jogo de qualidade, encontra-se a IA, que, para esse tipo de aplicação, resume-se basicamente na interação do mundo virtual e programado, com o jogador. São dois os agentes dessa interação: Os personagens não jogáveis (*Non-Player Characters* - NPCs) e o jogador em si. Esses NPCs não são controlados pelo jogador e sim, por algoritmos predefinidos no desenvolvimento (CROCOMO, 2008). Um exemplo de NPCs famosos, são os “fantasmas” do jogo “Pac Man”, entidades pré-programadas cujo objetivo principal era colidir com o personagem Pac Man (controlado pelo jogador).

Um dos maiores desafios em relação a NPCs está na qualidade da IA empregada no jogo. Dependendo do caso, o jogo torna-se desinteressante ou demasiadamente previsível para os jogadores. Esse desinteresse é causado principalmente quando a programação é pobre e as reações são previsíveis e de fácil manipulação pelo jogador (geralmente encontramos este problema em IA's programadas com Máquina de Estado Finito e Scripting) (PONSEM, 2004) (PRECHELT, 2003) (CROCOMO, 2008).

Um bom jogo pode ser totalmente prejudicado por uma metodologia ruim na programação das reações dos personagens não jogáveis. A aplicação de algoritmos adaptativos na programação de NPCs pode ser uma alternativa para ter-se um jogo de qualidade que possa ser interessante e divertido para qualquer jogador.

Algoritmos adaptativos pertencem à categoria de otimização, aqueles que, por meio de testes consecutivos, conseguem encontrar uma resposta ótima (local ou global) para a situação analisada. Um bom exemplo de algoritmos adaptativos que são aplicados para essa finalidade são os Algoritmos Evolutivos (AEs) (TOMASSINI, 1995). Jogos se apresentam como um bom ambiente para AEs, pois dispõem de problemas como co-evolução e evolução multiobjetivo (KOZA, 1992) (CROCOMO, 2008).

Em uma abordagem superficial, basta compreender que os AEs têm como fundamento alcançar um objetivo por meio de tentativa e erro, como todo algoritmo baseado em otimização. Algoritmos evolutivos baseiam sua estrutura e funções em situações ocorrentes na evolução biológica e tiram proveito de características evolutivas que possibilitaram o sucesso biológico da evolução das espécies, como a recombinação biológica, seleção natural e mutação para encontrar soluções para os problemas (TOMASSINI, 1995) (CROCOMO, 2008). Tendo isso em mente, é esperado que um NPC com capacidades evolutivas, ofereça um ambiente mais interessante e desafiador para o jogador. Neste trabalho, foi adotado um Algoritmo Genético comum para obtenção de resultados evolutivos no comportamento dos NPCs.

1.2. Objetivos

1.2.1 Objetivo Geral

Demonstrar como a introdução de um algoritmo genético a personagens não jogáveis (NPCs) de um jogo, proporciona um ambiente mais imersivo, desafiador e divertido para jogadores em comparação a uma máquina de estados finitos.

1.2.2 Objetivos Específicos

- Desenvolver um jogo com máquina de estados finitos;
- Melhorar o jogo desenvolvido com a aplicação do Algoritmo Genético

em personagens não jogáveis;

- Gerar um artigo científico com base no trabalho de conclusão de curso.

2. REVISÃO DE LITERATURA

2.1. Jogos Eletrônicos

Jogos eletrônicos, ou digitais, são uma forma de entretenimento no qual o jogador interage com imagens enviadas a uma televisão ou um monitor, por um sistema de *hardware*, ou plataforma. Esse sistema interlocutor é chamado de videogame, console ou computador sendo responsável pelo processamento dos jogos e dos comandos executados pelo jogador. Uma plataforma para jogos pode ser um computador, celular, ou mesmo um sistema dedicado (console). O dispositivo de entrada usado para manipular os jogos eletrônicos é chamado controle e varia de acordo com a plataforma (WINTER, 2014).

No fim da década de 50, surge “Tennis for Two”, um jogo renderizado na tela de um osciloscópio e processado por um computador analógico, considerado o primeiro jogo eletrônico, ficou mundialmente conhecido (HIGINBOTHAM, 1958). Desde a disseminação da idéia de um “jogo eletrônico”, modelos de negócio inteiros foram sendo criados, embasados no propósito de explorar as fronteiras dos jogos eletrônicos. Um dos exemplos de empresa surgida na era primordial dos jogos eletrônicos é a Atari (1972) que apresentou ao mundo o jogo “Pong”, inicialmente feito para ser jogado em um console específico, o jogo consiste em uma tela que mostra um ponto que navega no monitor e colide com as fronteiras da tela e com duas barras, uma em cada extremidade horizontal, que se movem na vertical conforme o jogador executa comandos no controle, se o ponto colide com alguma extremidade horizontal do monitor, o jogador da extremidade oposta ganha um ponto (BATISTA et al, 2007) (SABADELLO, 2006).

Existem vários tipos de perspectiva em um jogo, os mais clássicos e conhecidos para jogos em duas dimensões são os estilos “*Top-Down*” e o “*Side-Scroller*”. Um jogo com perspectiva *Top-Down* é todo aquele cuja cena é apresentada como se o jogador observasse o ambiente e as entidades olhando “de cima”, os personagens se movem para cima, baixo, esquerda e direita na tela (Norte, Sul, Oeste e Leste, por exemplo) sem restrições, já um jogo estilo *Side-Scroller* é

jogado como se fosse observado lateralmente pelo jogador, as entidades andam apenas na horizontal (de Oeste a Leste apenas, por exemplo) e existe uma gravidade artificial, o que permite simular pulos (WOLF, 2008).

Nintendo, Sega, Namco, Sony, Microsoft são exemplos que foram surgindo com o passar dos anos e conquistam jogadores e amantes de jogos eletrônicos por todo mundo até hoje (KIRRIEMUIR; MCFARLANE, 2004) (BATISTA et al, 2007). Com o passar do tempo a evolução gráfica foi a maior preocupação no desenvolvimento desses jogos, que começaram com imagens de 8 bits, passaram a ser 16 bits, 32 bits e atualmente gráficos tridimensionais, hiper-realistas, texturizados, sombreados e dinâmicos são a realidade comum, e jogos com duas dimensões de jogabilidade lançados atualmente são considerados “retrô” (termo que designa algo que remete a um estilo antigo ou ultrapassado) (BATISTA et al, 2007).

Os últimos anos, têm se destacado como era da “realidade virtual”, na qual os desenvolvedores de jogos tomam proveito de sensores de geolocalização para simular uma realidade aumentada em um dispositivo acoplado a cabeça do jogador que, por meio de elementos gráficos, “ilude”, visualmente, a sensação de realidade do jogador, tornando o jogo um ambiente muito mais imersivo e cativante (CHOI; JUNG; NOH, 2015) (MA; JAIN; ANDERSON, 2014).

Nos primórdios do desenvolvimento de jogos, era preciso o conhecimento profundo de cada componente do sistema que fosse executar o jogo, caso contrário, a compatibilidade da programação com o sistema impediria que o jogo fosse executado corretamente. O programador precisava conhecer o *hardware* para poder implementar o jogo, e este seria desenvolvido especificamente para aquele conjunto de *hardware*, porém, com avanços das técnicas de desenvolvimento, hoje o programador pode tomar proveito de bibliotecas de desenvolvimento e “*Game Engines*” para focar seus esforços, não tanto na compatibilidade, mas sim no jogo em si (MCGUGAN, 2007).

2.2. Inteligência Artificial para Jogos Digitais

Existe uma diferença no uso do termo “Inteligência Artificial” (IA) no âmbito acadêmico e em jogos eletrônicos. Para distinguir ambos contextos de aplicação, os desenvolvedores adotaram o termo *Game AI (Artificial Intelligence)* para IAs aplicadas a jogos (FUNGE, 2004) (KISHIMOTO, 2004). A diferença entre IA no meio acadêmico e no meio de jogos eletrônicos, é basicamente o objetivo que cada uma busca. No meio acadêmico, o objetivo se resume a busca de soluções para problemas difíceis ou complexos, como simular características humanas (reconhecimento facial e de imagens e objetos, por exemplo), entender e construir agentes inteligentes (SCHWAB, 2004)(KISHIMOTO, 2004).

Já no contexto de jogos eletrônicos, o objetivo de usar IA gira em torno prioritariamente da diversão, o problema não é como o sistema pensa, mas sim como ele age. Isso se deve pelo fato que jogos eletrônicos são negócios, consumidores de jogos buscam diversão, não é de interesse do jogador como a IA de personagens não jogáveis (do inglês *Non-Player Characters* - NPCs) em um jogo foi criada, desde que ela transforme o jogo divertido e desafiador, além, claro, de tomar decisões coerentes com o contexto do jogo (TOZOUR, 2002) (SCHWAB, 2004) (KISHIMOTO, 2004).

A IA para jogos eletrônicos vem se mostrando como uma característica crucial e fundamental para a produção de um jogo de qualidade que ofereça um ambiente desafiador e divertido para o jogador. Por isso, quanto mais alcança-se maestria gráfica e de processamento, mais a IA do jogo tende a ser o foco das inovações (ANDRADE; SILVA; CROCOMO, 2009) (CROCOMO, 2008).

É relativamente recente a preocupação que os desenvolvedores de jogos passaram a ter para com a importância financeira que técnicas de IA mais complexas exercem sobre seus produtos (YANNAKAKIS, 2005). Em Crocomo (2008) são apresentadas algumas das técnicas de IA para jogos digitais, como: Máquinas de Estado Finito, Scripting, Lógica Nebulosa e Algoritmos Evolutivos.

2.2.1. Máquinas de Estado Finito

De acordo com Perucia (2005), as Máquinas de Estado Finito foram utilizadas com maior frequência nos jogos do que qualquer outra técnica de Inteligência Artificial (IA). Este fenômeno ocorre pois esta técnica é de fácil implementação e depuração, apresenta baixa complexidade e é suficiente para satisfazer as necessidades do jogo na grande maioria dos casos. O maior ponto negativo das Máquinas de Estado Finito é sua previsibilidade por parte do jogador. Por ser determinística, a entidade controlada pela IA, ou NPC (do inglês *Non-Player Characters*, Personagens não jogáveis), fica restrito a estratégias finitas que, no decorrer do jogo, podem se tornar previsíveis e repetitivas (MORAIS; SILVA, 2009) (CROCOMO, 2008).

2.2.2. Scripting

Assim como Máquinas de Estado Finito, a técnica de Scripting é muito popular entre desenvolvedores de jogos na parte de Inteligência Artificial (ZELKOWITZ, 2003). Scripts são fáceis de serem entendidos, de serem implementados e de serem utilizados por usuários comuns (sem experiência em programação), isso deve-se principalmente ao fato de que, geralmente, scripts são criados de forma a lembrar uma linguagem natural, como, por exemplo, o inglês. A principal característica do scripting é que comandos de código são interpretados em tempo real no jogo, dando mais flexibilidade às opções tomadas pelo personagem, porém assim como Máquinas de Estado Finito, o Scripting é uma técnica determinística e de fácil previsibilidade (MORAIS; SILVA, 2009)(CROCOMO, 2008).

2.2.3. Lógica Nebulosa (Fuzzy)

A Lógica Nebulosa apresenta, não apenas nos jogos, a possibilidade de se trabalhar com informações “incompletas”, podendo lidar com dados variantes de 0 a 1 e não apenas 0 ou 1, como é mais comum na Lógica Booleana (ZADEH, 1976). Sua implementação é complicada e custosa, porém lida com casos que técnicas

anteriormente citadas não conseguem lidar. Também pode ser combinada com outra técnica, como por exemplo a Máquina de Estado Finitos Nebulosa, que é a junção da Lógica Nebulosa com Máquina de Estados Finito. Essa junção remove o determinismo da Máquina de Estados Finitos e pode ser usada em diversas situações em jogos (CROCOMO, 2008).

2.2.4. Algoritmos Evolutivos

Algoritmos Evolutivos (AEs) são uma classe de algoritmos que baseiam sua estrutura lógica numa gama de mecanismos da evolução biológica, ou seja, espelham-se em fatores do meio biológico, como a evolução, para obter resultados. AEs vêm sendo explorados com grande frequência, na grande parte com o intuito de comprovar a eficácia desta técnica para criação de uma Inteligência Artificial (IA) adaptativa e dinâmica em jogos (ENGELBRECHT, 2007).

A função de um AE em um jogo é fazer com que a estratégia utilizada pelo computador se adapte à estratégia utilizada pelo usuário a cada vez que joga. AEs aplicados em jogos possuem duas vertentes adaptativas, são elas: adaptação direta e adaptação indireta.

Adaptação Direta: Diretamente relacionada a criação de novos comportamentos e estratégias para os personagens não jogáveis, ou NPCs (do inglês *Non-Player Characters* - NPCs). Controlar comportamentos de um NPC com adaptação direta para que esses sejam aceitáveis não é uma tarefa fácil, porém a adaptação direta possui as vantagens não limitar comportamentos possíveis, o que permite estratégias inesperadas que enriquecem o jogo com uma gama extensa de possibilidades.

Adaptação Indireta: Os comportamentos do NPC são pré programados e a função do AE é apenas alternar entre esses tipos de comportamentos baseado nas estatísticas do jogo. Um exemplo é a seleção de um nível de dificuldade baseado na performance do jogador (CROCOMO, 2008).

2.2.4.1. Algoritmos Genéticos

Algoritmos Genéticos (AGs) tem seus modelos baseados na evolução, onde as características de indivíduos são expressas usando genótipos. Os principais operadores de AGs a seleção (responsável por selecionar indivíduos mais aptos) e recombinação (cruzamento dos principais indivíduos para geração de novas populações, simulando a reprodução biológica) (ENGELBRECHT, 2007).

O problema de otimização consiste em encontrar o melhor conjunto de soluções em meio a todas soluções possíveis. O modelo desse problema é tradicionalmente representado por uma função matemática (denominada função objetivo) e consiste na busca de argumentos adequados que maximizam ou minimizam essa função. Um tipo de otimização clássica é aquela que acontece com base em um objetivo. Para isso existe uma única função que modela esse objetivo (COLLETE; SIARRY, 2003). Nem sempre é possível encontrar a solução ideal, ou seja, a função que melhor otimiza um problema em tempo viável (problemas NP-Completo). Para isso existem certas categorias de algoritmos que são voltados para busca das melhores soluções possíveis dentro de um tempo viável nessa categoria de problemas. Um exemplo clássico para esses problemas são os Algoritmos Genéticos (AGs) (JONG; SPEARS, 1989).

Algoritmos Evolutivos (AEs) possuem ramificações que direcionam a estrutura lógica para extremos específicos. Os AGs são uma dessas ramificações, uma classe de AES que apresentam características semelhantes a hereditariedade, mutação, seleção natural e recombinação (ou *crossing over*) biológicas.

Um AG, pode ser definido como sendo um procedimento iterativo, ou seja, ocorre com a repetição de um conjunto de etapas. A primeira dessas etapas, é a geração de uma população de indivíduos, nos AGs essa geração é aleatória. Pode-se dizer que cada um destes indivíduos é basicamente um vetor com uma possível solução do problema, esse vetor é chamado de “cromossomo”. A Figura 1 apresenta três exemplos de diferentes estruturas do cromossomo, a estrutura binária, que é composta apenas por zero ou um que representa presença, ou falta de característica, a estrutura composta por números inteiros, que podem

representar, além da presença, a intensidade ou tipo de variação de um característica e a estrutura com números reais, que expressa as mesmas consequências no cromossomo, das citadas anteriormente.

Binários	1	0	1	1
Inteiros	2	1	4	6
Reais	1.2	0.5	2.6	1.0

Figura 1. Tipos de cromossomos em Algoritmos Genéticos

Fonte: Elaborado pelo autor

Tradicionalmente, na representação binária, os indivíduos são representados por vetores binários, nos quais “1” representa “presença” e “0” representa “ausência” de certa característica. Porém, além da representação binária, é possível representar um cromossomo usando qualquer uma das suas formas (números inteiros, reais ou binários), a estrutura escolhida depende da forma como a caracterização do personagem não jogável, ou, NPC (do inglês *Non-Player Character*) fora planejada.

Cada solução ("cromossomo") passa por uma função de avaliação da sua eficácia, recebendo um valor de aptidão (chamado de fitness score). A definição da função que define o valor de *fitness* é arbitrária, e de extrema importância para se alcançar resultados esperados, geralmente para defini-la com precisão, são necessários testes consecutivos e observação de resultados.

Com base nesta pontuação, um grupo de indivíduos é selecionado (essa etapa chama-se "seleção"), os que possuem a melhor aptidão e também alguns que não possuem uma aptidão tão boa. É preciso selecionar alguns indivíduos cuja aptidão é inferior para se manter uma boa variabilidade genética da população, isso evita que os resultados se prendam a uma solução mínima ou máxima local, para isso existem métodos específicos, como o método da “Roleta”, no qual, indivíduos

com melhores *fitness* tem maiores chances de serem escolhidos, porém não é garantida a seleção. O método de “roleta” funciona basicamente normalizando-se os valores de *fitness* e transformando o valor resultante da normalização na probabilidade de seleção de cada indivíduo. Para exemplificar, supondo uma população de 3 indivíduos, onde os resultados dos *fitness* foram 5, 8 e 10 respectivamente, para se normalizar os *fitness* basta somar todos e dividir cada resultado pela soma geral, seguindo essa regra, os resultados normalizados desse exemplo seriam aproximadamente 0.217, 0.347 e 0.434 respectivamente e esses valores correspondem a probabilidade de seleção de cada um dos 3 indivíduos, sendo que o indivíduo com melhor aptidão, ou *fitness*, tem maiores chances de ser selecionado.

Após a etapa de seleção são aplicados os operadores evolutivos de crossover e mutação, nada mais são que funções de “cruzamento” que dobram o número de indivíduos da quantidade de selecionados, criando assim um novo conjunto de soluções candidatas (uma nova geração, iteração). Desta forma, o processo é reiniciado a partir da fase de avaliação. Um método conhecido para crossover é a “Recombinação em um ponto” e é exemplificada na Figura 2, mostrando indivíduos “pais” que foram selecionados para manter suas características, gerando dois novos indivíduos filhos, que possuem, neste caso, metade de seu cromossomo composto por metade dos cromossomos de cada indivíduo pai (ENGELBRECHT, 2007) (TOMASSINI, 1995) (CROCOMO, 2008).

Figura 2: Cruzamento em um ponto

Fonte: Elaborado pelo autor

Com o passar das gerações, pelas iterações, do AG, ou evolutivo, existe uma tendência de que as soluções obtidas pelos indivíduos sejam cada vez mais adequadas, mostrando cada vez mais eficiência na resolução. Além disso, caso o problema seja modificado durante o processo evolutivo, é esperado que o algoritmo, com seus indivíduos, comece a buscar dinamicamente novas soluções, gerando assim soluções adaptativas (ENGELBRECHT, 2007) (CROCOMO, 2008).

3. METODOLOGIA

A metodologia proposta para o desenvolvimento deste projeto compreende abstração dos conceitos, modelagem e desenvolvimento do jogo, implementação dos dois tipos de Inteligência Artificial (IA) que serão averiguados e testes. Esta seção apresenta a metodologia empregada para o desenvolvimento do projeto proposto, dividida em: desenvolvimento do ambiente de testes das IAs a serem analisadas (jogo); aplicação dos algoritmos de IA (máquina de estados finitos e algoritmo genético) nos personagens não jogáveis do jogo desenvolvido; Teste com jogadores; Aplicação de questionário avaliativo; Averiguação dos resultados e conclusão.

3.1. Materiais e Ferramentas

Foi realizada uma análise dos materiais e ferramentas essenciais para o desenvolvimento do projeto. Bem como a definição das etapas necessárias. Algumas diretrizes foram definidas como a utilização da linguagem de programação. Foi adotado a linguagem Python 2.7 devido não só a facilidade de implementação, mas a versatilidade oferecida por características da linguagem, como por exemplo, tipagem dinâmica e orientação a objetos.

No desenvolvimento de jogos eletrônicos, a biblioteca PyGame destaca-se com uma vasta gama de funções prontas e facilidades de integração, sendo destaque como uma forte ferramenta no desenvolvimento de jogos usando Python. PyGame, assim como a maioria dos sistemas de desenvolvimento de jogos ou “*Game Engines*”, já traz uma implementação multiplataforma em suas funções, permitindo assim que o programador não precise se preocupar com quantos hertz o processador que executará o jogo opera, por exemplo. PyGame lida com isso automaticamente e o programador pode focar seus esforços totalmente no desenvolvimento do jogo em si e esquecer os problemas de compatibilidade (MCGUGAN, 2007).

3.2. Jogo Desenvolvido

Dentre as perspectivas de jogo estudadas, constatou-se que o desenvolvimento de um jogo estilo “*Side Scroller*” possuiria uma implementação mais custosa, sendo que um jogo em perspectiva “*Top-Down*” além de mais simples para se implementar, possibilita uma gama maior de possibilidades estratégicas, devido a liberdade de movimentação dos agentes do jogo.

Para este trabalho, o jogo foi desenvolvido em perspectiva *Top-Down* (como mostram as Figuras 3 e 4), devido não só a facilidade de implementação, como também por permitir uma gama maior de possibilidades para as regras elaboradas devido ao estilo Top-Down não restringir movimento para nenhuma direção.

Figura 3: Jogo no modo Algoritmo Genético

Fonte: Elaborado pelo autor

Figura 4: Jogo no modo Máquina de Estados Finitos

Fonte: Elaborado pelo autor

O jogo é basicamente um “shooter”, sendo shooter uma categoria focada em disparar um projétil no oponente. O jogador controla o personagem com aparência humana, batizado de “Dude” (ou “personagem”, “rapaz”). A Figura 5 representa o personagem “Dude” e os demais componentes do jogo, sendo eles os itens, os oponentes e o projétil que terão seu funcionamento explicado a seguir.

Figura 5: Componentes do jogo

Fonte: Elaborado pelo autor

A movimentação é realizada pelas setas direcionais do teclado e para lançar o projétil deve-se usar a barra de espaço. Quando o projétil acerta um oponente, uma das criaturas verdes, a barra verde acima do oponente atingido diminui de tamanho, representando o dano tomado e os pontos de vida perdidos. Quando a vida de um oponente chega a 0, ele é eliminado e não aparece mais na tela do jogo. Para cada inimigo atingido ou eliminado o jogador recebe pontos. Ao tocar em um inimigo o jogador perde seus pontos de vida e se chegarem a 0 o jogador perde o jogo. Existe um fator limitante dos projéteis disparados, a limitação da “munição” disponível, cada tiro é subtraído na quantidade total de munições e se chegar a 0 o jogador não pode mais atirar.

O jogo gira em torno de basicamente tudo citado, porém, para um bom ambiente de testes é preciso um jogo que possua um certo nível de complexidade, assim o algoritmo evolutivo poderá optar por diversos tipos diferentes de estratégias. A partir dessa premissa o jogo é composto não apenas de inimigos para o jogador eliminar, como também de diversos itens diferentes com consequências diferentes para o jogo.

Primeiramente, existe a possibilidade de se surgir um item de munição no “chão”, esse item aumenta as munições disponíveis em 15. Portanto se o jogador disparar todos os “tiros” disponíveis, ele ainda pode destruir os inimigos, basta esperar que um item desses surja no campo, até lá seu único desafio será desviar dos oponentes.

Em segundo, seria interessante se o jogador conseguisse imunidade por um certo tempo, para isso serve o item de “escudo”. Ao tocar no item do escudo, quando surgir no chão, o jogador se equipa com um escudo que lhe garante 10 segundos de imunidade, nesse tempo a colisão com os oponentes não causa dano, seus pontos de vida disponíveis não diminuem. Para incrementar a complexidade e o jogo em si, os oponentes, controlados pela Inteligência Artificial (IA), também podem pegar o escudo se tocarem nele, quando feito isso, o oponente se torna imune aos projéteis disparados pelo jogador por 10 segundos.

O terceiro item pensado foi um item que recuperasse os pontos de vida disponíveis. Tanto o jogador, quanto os inimigos, perdem pontos de vida no decorrer do jogo, como já explicado, porém, o item de cura permite que 500 pontos de vida

sejam reconstituídos nos pontos de vida totais, portanto se o jogador encostar em um inimigo e esteja com 200 pontos de vida restantes, ao tocar o item de cura, fica com 700 pontos de vida como resultado. O mesmo vale para cada oponente controlado pela IA.

Por fim, o quarto item é o item de pontos extras. Cada tiro que o jogador acerta em algum oponente lhe garante pontos (diretamente proporcionais ao tempo que o projétil encosta no inimigo), mas existe um item que garante 500 pontos instantâneos. Porém, se um inimigo pegar esse item antes, o jogador não recebe nenhum ponto extra e o item desaparece da tela.

O jogo também possui uma função que, ao fim do jogo, quando o jogador perder ou sair por vontade própria, um relatório é gerado, contendo o modo de jogo, o tempo de jogo em cada rodada, o tempo de jogo total, os pontos que o jogador adquiriu em cada rodada e no total e a vida restante do jogador em cada rodada, bem como todos os cromossomos de cada NPC (do inglês *Non-Player Character*, ou Personagem não jogável) em cada rodada, caso o modo jogado fosse o modo com Algoritmo Genético.

3.3. Aplicação dos algoritmos propostos

Neste subcapítulo, será abordada a forma que cada algoritmo proposto foi aplicado aos NPCs (do inglês *Non-Player Character*, ou Personagem não jogável) do jogo desenvolvido.

Tanto a máquina de estados finitos quanto o algoritmo genético foram desenvolvidos sem auxílio de bibliotecas, ambos foram projetados inteiramente “a mão”, seguindo, claro, a literatura por trás dos mesmos.

3.3.1 Aplicação da Máquina de Estados Finitos

A máquina de estados finitos, em jogos, na sua grande maioria, atua principalmente na movimentação dos NPCs (do inglês *Non-Player Character*, ou

Personagem não jogável) e, conforme o jogo progride, a tendência é que apareçam novos NPCs regidos por diferentes tipos de movimentação definidas pela sua máquina de estados finitos. Esse conceito foi aplicado ao jogo desenvolvido neste projeto. Na primeira rodada, ou primeiro nível, todos os NPCs seguem um tipo de movimentação simples, na horizontal e ao encostar em uma das extremidades da tela do jogo, inverte seu sentido de movimentação, sempre mantendo o mesmo eixo horizontal. No nível 2, ou segunda rodada, que ocorre após o jogador eliminar todos os inimigos no nível 1, a movimentação se torna vertical, como uma variação da primeira movimentação. Já no nível 3, a movimentação de todos NPCs é diagonal, semelhante a antigos screensavers de aparelhos de DVD e assim, gradualmente, as movimentações vão ficando mais rebuscadas até todos os NPCs terem movimentações diferentes, regidas por um certo nível de aleatoriedade aplicado a máquina de estados finitos, ou seja, por mais que seja aleatório, nunca vai deixar de seguir um padrão, o que muda é a direção ou sentido do NPC em momentos aleatórios.

3.3.2 Aplicação do Algoritmo Genético

Como mostrado no item 2.3.1, Algoritmos Genéticos baseiam-se na otimização usando como parâmetro de sucesso o resultado de uma função de *fitness* relacionada a um vetor correspondente, sendo esse vetor o “cromossomo” do indivíduo. Cada personagem não jogável, ou NPC (do inglês *Non-Player Character*), nesta implementação, possui seu próprio vetor de características, composto de números inteiros e reais, sendo assim um cromossomo misto. O cromossomo, que define o comportamento do mesmo. Este vetor possui 8 posições e cada posição define uma característica.

Figura 6. Exemplo de cromossomo

Fonte: Elaborado pelo autor

A posição 1 define o tipo de movimentação, variando de 0 a 10 pois existem 10 tipos de movimentação implementadas.

A posição 2 define a velocidade de movimentação do NPC e varia de 70 a 130 unidades de velocidade.

As posições 3, 4 e 5 do vetor são relacionadas aos itens. Todo NPC pode interagir com os seguintes itens (como mencionado em 3.2): Item de vida, item de escudo e item de pontos extras. As posições 3, 4 e 5 do vetor de características de cada NPC define se o respectivo NPC tem afinidade ou não com este item levando em conta o campo de visão. Cada NPC possui um “campo de visão”, este campo nada mais é que um quadrado ao redor do mesmo, esse quadrado é responsável por ativar *flags* caso algum outro elemento venha a intersectar sua área com a área do quadrado do campo de visão, então quando um item “entrar” no campo de visão do NPC, o NPC saberá exatamente de qual item se trata e sua posição, baseado nesses dados, o NPC irá agir conforme as posições 3, 4 e 5 lhe regem, a posição 3 define atitudes em relação ao item de vida, a 4 em relação ao item de escudo e a 5 em relação ao item de pontos extras. Os valores nessas posições do vetor variam de 0 a 2, sendo 1 para ir em direção ao item quando ele estiver no campo de visão, 2 para mudar seu sentido quando o item estiver no campo de visão, ou seja, evitar o item e 0 para ignorar se o item se encontra ou não no campo de visão, ou seja, apenas continuar seguindo sua movimentação padrão sem interferência alguma. Exemplificando: Se a posição 3 do vetor de características do NPC possuir valor 1, quando um item de vida estiver no seu campo de visão, ele irá em direção a posição

do item de vida, até alcançá-lo (ou até outro NPC com a mesma característica, ou o jogador, alcançá-lo).

A posição 6 do vetor segue os mesmos princípios das posições 3, 4 e 5, mas esta é em relação ao jogador. Se o jogador estiver no campo de visão do NPC, a posição 6 do vetor de características desse NPC é consultada para saber como reagir. Essa posição varia de apenas entre 0 e 1, sendo 0 ignorar o jogador e 1, perseguir o jogador enquanto este estiver no campo de visão do NPC.

A posição 7 é referente ao tamanho do campo de visão do NPC, variando entre 0.5 e 1.5. Esses valores são multiplicadores do tamanho padrão do campo de visão, ou seja, um NPC com o valor 1.2 na posição 7 de seu cromossomo possuirá um campo de visão 1.2 vezes maior que o tamanho padrão do mesmo.

A posição 8 no vetor varia a vida do NPC, variando entre 0.5 e 2, sendo esses valores multiplicadores do valor de vida padrão (1000). Ou seja, um NPC com o valor 0.7 na posição 8 de seu cromossomo possuirá uma vida de 700 unidades de vida.

Na primeira rodada, ou primeiro nível os valores dos cromossomos de cada NPC são gerados aleatoriamente, ou seja, cada NPC possuirá suas próprias características, sendo 5 NPCs por rodada em jogo. O jogador precisa lidar com as particularidades de cada NPC para conseguir eliminar todos e passar para o próximo nível. No próximo nível, os 2 NPCs com melhores resultados têm maiores chances de serem selecionados para continuar em jogo, já os 2 com piores resultados têm maiores chances de serem substituídos. Essa seleção é realizada pelo método de seleção “roleta” que funciona normalizando os resultados dos *fitness* de todos indivíduos e, através do valor resultante, que varia de 0 a 1, definir a probabilidade de seleção de cada um. Com a seleção, os NPCs com piores resultados são provavelmente substituídos por outros 2 novos NPCs com cromossomos “filhos” do cruzamento em um ponto dos 2 NPCs selecionados para manter-se no jogo. O resultado de aptidão mencionado é definido pela função de *fitness* implementada.

A função de *fitness* implementada leva em conta algumas variáveis. Uma delas é a variável de segundos vivo, ou quantos segundos um NPC permaneceu vivo na rodada, ela é somada a variável de pontos do NPC, um NPC faz pontos

quando encosta em um jogador, causando dano a ele e quando pega um item de pontos extras, ou seja, um NPC com boa ofensividade e durabilidade é considerado um bom NPC. Para garantir valores mais fiéis ao resultado buscado e garantir melhor relação entre as variáveis, é realizada uma divisão que atribui mais peso para o tempo vivo. A função final foi definida portanto na Equação 1.

$$fitness = (pontos/25) + (segundos/2) \quad (1)$$

por apresentar resultados fiéis à otimização esperada.

Foi também implementada uma função de mutação que, para cada posição do novo vetor gerado com o cruzamento dos dois NPCs selecionados, oferece uma chance de 0,2% de mutação. Ou seja, se o NPC “pai”, vai garantir que a posição 2 do vetor do NPC filho terá o valor “97”, a função de mutação tem 0,2% de chance de fazer com que o valor “97” seja modificado para um valor aleatório (entre 70 e 130, visto que a posição 2 do vetor define a velocidade de movimentação e varia nesse intervalo), garantindo assim a chamada “variabilidade genética”.

Além da mutação, existe o chamado “NPC coringa”. Como mencionado neste tópico, o jogo é constituído de 5 NPCs toda rodada, mas apenas 2 são selecionados como os melhores e 2 como os piores. O NPC restante sempre tem seu cromossomo gerado aleatoriamente na rodada, garantindo sempre o fator da imprevisibilidade do jogo.

Com NPCs possuindo as características apresentadas, o jogador em cada nova rodada, terá que lidar com os NPCs que lhe oferecem maior resistência, ou seja, aqueles que oferecem maior desafio para continuar, criando assim um ambiente desafiador e conseqüentemente, supostamente, mais interessante para o mesmo.

Figura 7: Algoritmo genético aplicado

Fonte: Elaborado pelo autor

3.4. Teste aplicado

O teste realizado, consistiu em apresentar e aplicar o jogo a alunos, ao todo 65, do curso de Ciência da Computação do Instituto Federal de Ciência e Tecnologia do Sul de Minas Gerais, Campus Muzambinho que cursaram o segundo semestre no ano de 2017.

Um arquivo executável do jogo foi disponibilizado para os alunos, seguido de uma breve explicação das mecânicas do jogo. Os alunos tiveram 30 minutos para jogar o jogo decidindo quanto tempo usariam para cada modo de jogo, assim o relatório gerado (citado no item 3.2) possuiria a informação de quanto tempo cada modo de jogo foi jogado, mostrando qual modo recebeu mais atenção em média, por parte dos jogadores.

3.5. Avaliação do teste

Após jogarem o jogo e cada um dos modos de jogo, os avaliados deveriam responder um questionário. O questionário possui questões de ambos os modos de jogo, todos os dados sendo automaticamente salvos na nuvem de armazenamento gratuita da plataforma *Google Drive*. Os sujeitos foram catalogados por identificador (um número referente a ordem da participação), nome e e-mail. Sua função foi informar sua opinião sobre o jogo, após jogar as duas versões (com máquina de estado finito e com Algoritmo Evolutivo na Inteligência Artificial), preenchendo o formulário que contém os seguintes campos:

1. Nome;
2. Sexo;
3. Qual modo de jogo foi o mais imersivo na sua opinião?
 - a. Quão imersivo foi o modo "Estados Finitos" (Nada imersivo; Pouco imersivo; Mais ou menos imersivo; Bem imersivo; Muito imersivo)?
 - b. Quão imersivo foi o modo "Algoritmo Genético" (Nada imersivo; Pouco imersivo; Mais ou menos imersivo; Bem imersivo; Muito imersivo)?
4. Qual modo de jogo foi o mais desafiador?
 - a. Quão desafiador foi o modo "Estados Finitos" (Nada desafiador; Pouco desafiador; Mais ou menos desafiador; Bem desafiador; Muito desafiador)?
 - b. Quão desafiador foi o modo "Algoritmo Genético" (Nada desafiador; Pouco desafiador; Mais ou menos desafiador; Bem desafiador; Muito desafiador)?
5. Qual o modo de jogo mais divertido?
 - a. Quão divertido foi o modo "Estados Finitos" (Nada divertido; Pouco divertido; Mais ou menos divertido; Bem divertido; Muito divertido)?
 - b. Quão divertido foi o modo "Algoritmo Genético" (Nada divertido; Pouco divertido; Mais ou menos divertido; Bem divertido; Muito divertido)?
6. Qual o modo de jogo mais difícil?

- a. Qual o nível de dificuldade do modo "Estados Finitos" (Muito fácil; Fácil; Equilibrado; Difícil; Muito difícil)?
 - b. Qual o nível de dificuldade do modo "Estados Finitos" (Muito fácil; Fácil; Equilibrado; Difícil; Muito difícil)?
7. Qual modo prefere jogar novamente?
8. Por que prefere o modo escolhido acima?

Todas as informações com um nível abstrato de opinião foram medidas com base na escala Likert (*Likert Scale*) (LIKERT, 1932).

Com as informações obtidas pelas respostas dos questionários, foram estruturados gráficos referentes às respostas que permitiram visualizar resultados e estudá-los, podendo levar a conclusões que suportarão ou refutarão a questão inicial sobre a importância de Algoritmos Evolutivos aplicados a Inteligência Artificial de jogos eletrônicos.

4. RESULTADOS E DISCUSSÃO

Com os testes aplicados aos alunos presentes, um total de 65 respostas foram submetidas ao questionário.

Primeiramente, pode-se concluir com base nas respostas a preferência de modo de jogo da maioria dos alunos, visto que a última pergunta do questionário é exatamente essa.

Figura 8: Preferência de modo

Fonte: Elaborado pelo autor

De acordo com o gráfico gerado da Figura 8 é possível observar que 83.1% dos 65 alunos que jogaram o jogo com a máquina de estados finitos e com o Algoritmo Genético (AG), preferem o modo de jogo com o AG aplicado aos personagens não jogáveis, NPCs (do inglês, *Non-Player Characters*). A explicação desse fenômeno pode ser observada nos resultados das outras respostas, como por exemplo a opinião da amostra em relação a imersividade do jogo com cada Inteligência Artificial (IA).

No gráfico resultante da questão "Qual modo de jogo foi o mais imersivo na sua opinião", mostrado na Figura 9, o resultado mostra que a grande maioria achou o modo de jogo com o algoritmo genético mais imersivo que a máquina de estados finitos.

Figura 9: Modo mais imersivo

Fonte: Elaborado pelo autor

A maioria de 75.4% opina que sentiram-se mais imersos no jogo regido pelo AG, porém, apesar da imersão da máquina de estados finitos não ser um consenso da maioria, o modo de jogo não deixou de ser considerado "Bem imersivo" como mostra o gráfico da Figura 10.

Figura 10: Nível de imersão da Máquina de Estados Finitos

Fonte: Elaborado pelo autor

Porém o nível de imersão sentido pelos jogadores no algoritmo genético, foi considerado maior, por parte da maioria, exibindo o nível máximo da escala Likert,

ou a opção "Muito imersivo", como o principal item selecionado. A Figura 11 ilustra os resultados.

Figura 11: Nível de imersão do Algoritmo Genético

Fonte: Elaborado pelo autor

A resposta que mais se aproximou de um consenso geral, foi em relação ao desafio oferecido pelo jogo. Nesse quesito, 93,8% de todos os alunos concordam que o modo de jogo com algoritmo genético implementado oferece um ambiente mais desafiador ao jogador do que a máquina de estados finitos, como é possível observar no gráfico da Figura 12.

Figura 12: Modo mais desafiador

Fonte: Elaborado pelo autor

Em relação ao nível de desafio oferecido, o algoritmo genético conta com 90.8% das respostas sendo referentes aos dois maiores níveis da escala: "Muito desafiador", com 47.7% da preferência e "Bem desafiador" com 43.1% como mostra o gráfico da Figura 13.

Figura 13: Desafio do modo Algoritmo Genético

Fonte: Elaborado pelo autor

Já a máquina de estados finitos apresentou uma média maior direcionada ao centro da escala, com maioria das seleções em "Mais ou menos desafiador" e um certo equilíbrio entre "Bem desafiador" e "Pouco desafiador", como mostra a Figura 14.

Figura 14: Desafio do modo Máquina de Estados Finitos

Fonte: Elaborado pelo autor

Em relação a diversão, que, de acordo com (KISHIMOTO, 2004), o principal interesse em todo consumidor de jogos eletrônicos é a diversão que o jogo proporciona e sendo assim, desenvolvedores devem se preocupar em atender esse quesito, o resultado na questão de diversão, mais uma vez, mostra o algoritmo genético aplicado ao jogo com um fator importante na diversão obtida ao se jogar o jogo, com 78.5% das opiniões afirmando que se divertiram mais jogando o modo de jogo regido pelo AG, como mostra a Figura 15.

Figura 15: Modo mais divertido

Fonte: Elaborado pelo autor

Curiosamente, a máquina de estados finitos não apresentou um baixo nível de diversão proporcionada, porém esse resultado não foge do esperado. Todos os jogos clássicos, do início da era de jogos eletrônicos, possuíam como base da IA aplicada, máquinas de estados finitos simples (como discutido no tópico da literatura estudada). Jogos como “*Pitfall*” (Atari), com “jacarés” cuja movimentação é previsível e constante, até jogos como “*Super Mário*” (Nintendo), com inimigos com caminhos sem variação e “*Bosses*” ou “chefões” que seguiam sempre um padrão de ataque, baseado em máquinas de estados finitos, sempre foram aclamados pelo público por oferecer, acima de tudo, um ambiente divertido. Na pesquisa aplicada neste trabalho, por mais que o modo de jogo com máquinas de estados finitos não tenha sido considerado o mais divertido, quando analisado o nível de diversão é observado

que de fato, assim como em jogos clássicos, o jogo desenvolvido apresenta sim diversão considerável ao jogador, tendo a opção "Bem divertido" sendo a mais selecionada, como mostra o gráfico da Figura 16.

Figura 16: Diversão do modo Máquina de Estados Finitos

Fonte: Elaborado pelo autor

Porém, algoritmo genético aplicado como IA de base a NPCs de um jogo apresentou resultados muito superiores em relação a diversão que proporciona ao jogador, sendo selecionado pela maioria como mais divertido e recebendo no nível de imersão, a maioria das opiniões na opção máxima da escala "Muito divertido" com 55.4% da opinião da amostra, observado na Figura 17.

Figura 17: Diversão do modo Algoritmo Genético

Fonte: Elaborado pelo autor

Os resultados voltados a dificuldade do jogo apresentam uma conclusão interessante, um jogo difícil não necessariamente é repudiado, muito pelo contrário. O algoritmo genético foi considerado mais difícil em relação a máquina de estados finitos e ainda assim recebeu preferência da maioria e como observado anteriormente, foi também considerado mais divertido. A Figura 18 mostra os dados.

Figura 18: Modo mais difícil

Fonte: Elaborado pelo autor

Figura 19: Dificuldade do modo Máquina de Estados Finitos

Fonte: Elaborado pelo autor

O nível de dificuldade da máquina de estados finitos foi avaliado na maioria como "Equilibrado" e "Fácil", já o algoritmo genético recebeu na sua maioria o nível de "Difícil", "Equilibrado" e "Muito difícil", nesta ordem, como mostram as Figuras 19 e 20 respectivamente.

Figura 20: Dificuldade do modo Algoritmo Genético

Fonte: Elaborado pelo autor

Figura 21: Preferência de modo por sexo

Fonte: Elaborado pelo autor

Foi também gerado um gráfico, apresentado na Figura 21, da preferência de modo de jogo em relação ao sexo do jogador, a fim de averiguar diferenças por sexo, porém os resultados mostraram uma proporção claramente semelhante entre homens e mulheres.

O tempo de jogo dos participantes também foi analisado, porém dos 65 jogadores, apenas 40 submeteram o relatório de jogo corretamente. O tempo por partida em cada modo de jogo e o tempo total de cada modo de jogo, dos 40 jogos analisados mostraram como resultado o que se apresenta nos Quadros 1 e 2:

Quadro 1 - Tempos obtidos no modo Máquina de Estados Finitos.

Tempo Total	05:10:36 (310.6 minutos)
Média	07.765 minutos
Desvio padrão amostral	3.283
Variância amostral	10.78

Quadro 2 - Tempos obtidos no modo Algoritmo Genético.

Tempo Total	06:07:36 (367.6 minutos)
Média	09.189 minutos
Desvio padrão amostral	6.328
Variância amostral	40.04

Ou seja, pode-se comprovar que o algoritmo genético prendeu mais a atenção dos jogadores, sendo jogado aproximadamente 1 minuto e meio a mais na média, com um desvio padrão maior para o algoritmo genético, com partidas que duraram até quase 30 minutos e partidas rápidas de 2 minutos, o que também mostra a subjetividade ligada a cada jogador.

Nas respostas dissertativas, é possível observar opiniões bem concretas e definidas em relação a preferência dos jogadores, apresentando motivos importantes para os resultados observados, como por exemplo, pessoas com preferência na máquina de estados finitos, argumentaram sobre não gostarem muito de jogos e por

isso sempre ter preferência pelo que é mais fácil, ou mesmo, simplesmente, preferência por jogos mais fáceis. Já argumentos em relação ao AG, variaram, mas muitos ressaltando características que julgaram importante para definição da preferência que tiveram neste modo de jogo, como “mais divertido”, “mais desafiador” e inclusive “mais difícil”, foram argumentos justificativos para a escolha do AG como preferido.

5. CONSIDERAÇÕES FINAIS

Com tudo que foi apresentado, conclui-se que o objetivo geral do trabalho foi atingido com sucesso, sendo este alcançar uma conclusão em relação a duas inteligências artificiais implementadas a um jogo simples. A partir dos resultados obtidos, pode-se concluir que, em relação a algoritmos genéticos, a clássica máquina de estados finitos deixa a desejar, principalmente para jogadores mais assíduos e exigentes, que possuem um costume maior em jogar jogos eletrônicos na sua rotina.

Pôde-se observar também que mesmo sem o mesmo impacto para o jogador e mesmo sem receber maioria de votos na preferência, máquinas de estados finitos não deixam de ter seu valor no mundo dos jogos eletrônicos, apresentando um nível de diversão considerável para jogadores.

Conclusões por indução também foram recorrentes deste projeto, como o fato de que um jogo difícil não necessariamente irá obstruir o interesse do jogador, mas que, principalmente em jogadores mais assíduos, essa característica serve até como fator determinante na preferência por um jogo.

É possível observar, em suma, que uma inteligência artificial mais complexa pode proporcionar uma experiência totalmente diferente e, muito provavelmente, mais agradável ao consumidor de jogos. O papel de uma boa metodologia no desenvolvimento da IA de um jogo é de extrema importância para o resultado final e a aceitação do público alvo.

Por fim, esse trabalho se apresenta como uma ferramenta valiosa não só para desenvolvedores de jogos, como para toda a comunidade de estudiosos que buscam informações sobre inteligências artificiais aplicadas a jogos eletrônicos e a testes com algoritmos evolutivos, mais especificamente, o algoritmo genético.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDRADE, K. O.; SILVA, A.; CROCOMO, K.. **Um Algoritmo Evolutivo para a adaptação de NPCs em um jogo de ação**. Florianópolis. Anais do I Simpósio Santa Catarina Games - SCGAMES, 2009.

BATISTA, M.; QUINTÃO, P.; LIMA, S.; CAMPOS, L.; BATISTA T.. **Um estudo sobre a história dos jogos eletrônicos**. Revista Eletrônica da Faculdade Metodista Granbery, 2007.

COLLETE, Y.; SIARRY, P.. **Multiobjective Optimization. Principles and Case Studies**. Springer, 2003.

CHEROBIN, R.. **Aplicação do modelo da cadeia alimentar juntamente com algoritmos genéticos para criação de NPCs adaptativos**. Universidade do vale do itajaí curso de mestrado acadêmico em computação aplicada, Itajaí (SC), 2013.

CHOI, S.; JUNG, K.; NOH, S. D.. **Virtual reality applications in manufacturing industries: Past research, present findings, and future directions**. Concurrent Engineering, 2015.

CROCOMO, M. K.. **Um Algoritmo Evolutivo para Aprendizado On-line em Jogos Eletrônicos**. Dissertação (Mestrado) – Universidade de São Paulo – Instituto de Ciências Matemática e Computação (USP-ICMC), São Carlos, SP, 2008.

DALMAU, D. S.. **Core Techniques and Algorithms in Game Programming**. Indianapolis: New Riders, 2004.

ENGELBRECHT, A. P.. **Computational Intelligence, an introduction**. John

Wiley & Sons LTD, 2007.

HIGINBOTHAM, W.. **Tennis for two**. Retrieved September, v. 9, 1958.

JONG, K. A.; SPEARS, W. M.. **Using Genetic Algorithms to Solve NP-Complete Problems**. Proceedings of the third international conference on Genetic Algorithms, George Mason University, 1989.

KIRRIEMUIR, J.; MCFARLANE, A.. **Literature Review in Games and Learning**. Bristol: Futurelab, 2004.

KISHIMOTO, André. **Inteligência artificial em jogos eletrônicos**. Academic research about Artificial Intelligence for games, 2004.

KOZA, J. R.. **Genetic Programming: a paradigm for genetically breeding populations of computer programs to solve problems**. Stanford: Sandford University, 1990.

LAMOTHE, A.. **Tricks of the Windows Game Programming Gurus – Fundamentals of 2D And 3D Game Programming**. Indianapolis: Sams. 1999.

LIKERT R. **A technique for the measurement of attitudes**. Arch Psychology. 1932;22(140):55.

MCGUGAN, W.. **Introdução ao Desenvolvimento de Jogos em Python com Pygame**. Apress, 2007.

PERUCIA, A. S.; BERTHÊM, A.; BERTSCHINGER, G.; CASTRO, R.. **Desenvolvimento de Jogos Eletrônicos: Teoria e Prática**. São Paulo, SP: Novatec, 2005.

PONSEM, M.. **Improving adaptive game AI with evolutionary learning**. MSc

thesis, Faculty of Media & Knowledge Engineering, Delft University of Technology, Netherlands, 2004.

PRECHELT, L.. **Are scripting languages any good? A validation of Perl, Python, Rexx, and Tcl against C, C++, and Java.** In: ZELKOWITZ, Marvin. *Advances in Computers: Highly Dependable Software*. 58. ed. New York: Academic Press, 2003.

SABADELLO, M.. **History of electronic games.** Tese de Doutorado. master's thesis, Institute of Computer Graphics and Algorithms, Vienna Univ. of Techn, 2006.

SPRONCK, P.; SHPRINKHUIZEN-KUYPER, I.; POSTMA, E.. **Online Adaptation of Game Opponent AI in Theory and Practice.** In: INTERNATIONAL CONFERENCE ON INTELLIGENT GAMES AND SIMULATION, 4, 2003, Londres. *Proceedings of the 4th International Conference on Intelligent Games and Simulation*. Wolverhampton: University Of Wolverhampton, 2004.

COLE, N.; SUSHIL, J. L.; MILES, C.. **Using a Genetic Algorithm to Tune First-Person Shooter Bots.** Department of Computer Science University of Nevada Reno, Nevada

TOMASSINI, M.. **A Survey of Genetic Algorithms.** In: WORLD Scientific: *Annual Reviews of Computational Physics*: World Scientific, 1995.

YANNAKAKIS, G. N.. **AI in computer games: generating interesting interactive opponents by the use of evolutionary computation.** PhD thesis, University of Edinburgh, UK, 2005.

WINTER, D.. **Pong Story**, 2013. Disponível em:<<http://www.pong-story.com/intro.htm>>. Acesso em: 06 de abril. 2017.

WOLF, M.. **The Video Game Explosion: A History from PONG to Playstation and Beyond**. ABC-CLIO, 2008.

ZADEH, L. A.. **A Fuzzy-Algorithmic Approach to the Definition of Complex or Imprecise Concepts**. In: Intl. J. Man-Machine Studies 3, 1976.

ZELKOWITZ, Marvin. **Advances in Computers: Highly Dependable Software**. 58. ed. New York: Academic Press, 2003.