UNIVERSIDADE SAGRADO CORAÇÃO

VIRGILIO MISSÃO NETO

A UTILIZAÇÃO DE ALGORITMOS GENÉTICOS NO CONTROLE DO MOVIMENTO DE PERSONAGENS

BAURU
2011

VIRGILIO MISSÃO NETO

A UTILIZAÇÃO DE ALGORITMOS GENÉTICOS NO CONTROLE DO MOVIMENTO DE PERSONAGENS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação sob a orientação do Prof. Ms. Patrick Pedreira Silva.

BAURU
2011

Missão Neto, Virgílio
M67844a

A utilização de algoritmos genéticos no controle do movimento de personagens / Virgílio Missão Neto -- 2011.
75f. : il.

Orientador: Prof. Patrick Pedreira Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Sagrado Coração - Bauru - SP

1. Jogos. 2. Inteligência artificial. 3. Algoritmos genéticos. 4. Adobe Flash. I. Silva, Patrick Pedreira. II. Título.

VIRGILIO MISSÃO NETO

A UTILIZAÇÃO DE ALGORITMOS GENÉTICOS NO CONTROLE DO MOVIMENTO DE PERSONAGENS

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para a obtenção do título de Bacharel em Ciência da Computação sob a orientação do Prof. Ms. Patrick Pedreira Silva.

Banca examinadora:

_____

Prof. Ms. Patrick Pedreira Silva

Universidade Sagrado Coração

_____

Prof. Ms. Anderson Francisco Talon

Universidade Sagrado Coração

_____

Profa. Ms. Larissa Pavarini da Luz

Universidade Sagrado Coração

Bauru, 15 de Dezembro de 2011

Dedico este trabalho aos meus pais, meus amigos e minha namorada.

AGRADECIMENTOS

Agradeço aos meus pais, amigos e principalmente minha namorada por todo o apoio e cooperação que me deram, por várias vezes recorrerem a mim para ajudar ou auxiliar em algo e eu não poder ajudar por estar imerso no desenvolvimento deste projeto.

RESUMO

O entretenimento é uma das áreas da Tecnologia da Informação (TI) que está em grande crescimento, principalmente no Brasil, devido ao aumento do número de pessoas que fazem uso deste tipo de aplicação, incluindo não somente crianças e jovens, mas também muitos adultos. Este crescimento faz com que a busca por novas tecnologias voltadas a jogos aumente muito, principalmente com relação à interação do jogador com os elementos computacionais dos jogos como, por exemplo, os Não Jogáveis (*Non Player Character,* NPC). Para testar novas possibilidades voltadas ao grau de interatividade dos jogos, foi desenvolvido um protótipo de um jogo, utilizando a técnica de Inteligência Artificial (IA) dos Algoritmos Genéticos (AGs) para mapear todos os tipos de decisões que o NPC possa vir a tomar com relação à movimentação pelo cenário. Com base nisto é possível a geração de movimentos cada vez mais precisos e específicos de acordo com cada tipo de problema proposto ao NPC. De acordo com os testes realizados, os AGs se mostraram bem úteis para a sua utilização no comportamento de NPCs, sendo possível criar NPCs distintos, cada um tendo seu próprio comportamento e personalidade.

Palavras-chave: Jogos. Inteligência Artificial. Algoritmos Genéticos. *Adobe Flash*.

ABSTRACT

Entertainment is one of the areas of Information Technology (IT) is growing, especially in Brazil, because the increasing number of people who use this kind of application, including not only children and young people but also many adults. This growth makes the search for new games technologies to increase dramatically, particularly with respect to player interaction with the computational elements of the games, for example, non-player characters (NPC). To test new possibilities facing the degree of interactivity of games, we developed a prototype of a game, using the Artificial Intelligence (AI) technique of the Genetic Algorithms (GAs) to map all types of decisions that the NPC might take with relation to the movement in the scene. On this basis it is possible to generate more precise movements, and according to each specific type of problem presented to the NPC. According to the tests, the GAs has proved very useful for use in the behavior of NPCs, and NPCs can create distinct, each having its own behavior and personality.

Keywords: Games. Artificial Intelligence. Genetic Algorithms. Adobe Flash.

LISTA DE ILUSTRAÇÕES

Figura 1 – Computador PDP-1 usado na programação do "*Spacewar!*"..... 19
Figura 2 – Jogo Pitfall de 1982 para a plataforma Atari. 20
Figura 3 – Turtles in Time de 1991 para Arcade e Super NES..... 20
Figura 4 – Sony PSP lançado no ano de 2004. 21
Figura 5 – Nintendo Wii lançado no ano de 2006. 21
Figura 6 – Xbox Kinect lançado no ano de 2010..... 22
Figura 7 – Playstation Move lançado no ano de 2010. 22
Figura 8 – Estrutura de um Algoritmo Genético tradicional. 26
Figura 9 – Representação em código de um algoritmo genético. 27
Figura 10 – Algoritmo de uso de uma máscara de cruzamento. 32
Figura 11 – Operador de cruzamento de um ponto. 33
Figura 12 – Operador de cruzamento multiponto. 34
Figura 13 – Operador de cruzamento uniforme. 34
Figura 14 – Imagem do Jogo *"Cyclomaniacs 2"*. 37
Figura 15 – Imagem do Jogo *"Arcuz II"*. 38
Figura 16 – Imagem do Jogo *"Kingdom Rush"*. 39
Figura 17 – Interface Padrão para o Desenvolvimento de Aplicações. 40
Figura 18 – Apresentação da Janela de Projetos. 40
Figura 19 – Apresentação da Biblioteca. 41
Figura 20 – Apresentação da Linha do Tempo. 42
Figura 21 – Apresentação da Janela de Propriedades. 43
Figura 22 – Apresentação da Barra de Ferramentas. 43
Figura 23 – Apresentação do Palco. 44
Figura 24 – Aparência do *Lorde Mimnithus*. 47
Figura 25 – Aparência do Rei *Kyrms*. 48
Figura 26 – Aparência do Príncipe *Typhom*. 48
Figura 27 – Aparência do Mago *Emrys*. 49
Figura 28 – Aparência dos *Esqueletos*. 49
Figura 29 – Aparência dos *Dragões*. 49
Figura 30 – Aparência de *Mihnar*. 50

Figura 31 – Parte do mapa onde o personagem começa. 50
Figura 32 – Parte do cenário do jogo. 50
Figura 33 – Personagem e o objetivo do jogo. 51
Figura 34 – Representação visual do Mapa de Movimentos. 52
Figura 35 – Exemplo de Genoma de um Indivíduo da população Genética. 52
Figura 36 – Exemplo de Máscara de Cruzamento de Um Ponto Utilizada. 53
Figura 37 – Exemplo de Máscara de Cruzamento Multiponto Utilizada. 53
Figura 38 – Exemplo de Máscara de Cruzamento Uniforme Utilizada. 54
Figura 39 – Trecho de Código Referente a Implementação do Cruzamento. 55
Figura 40 – Trecho de Código Referente a Implementação da Mutação. 56
Figura 41 – Exemplo do Cálculo da Heurística de Movimento. 57
Figura 42 – Estrutura de um Algoritmo Genético. 70
Figura 43 – Ponto inicial do mapa. 71
Figura 44 – Rio localizado no centro do cenário. 71
Figura 45 – Local no mapa onde o personagem deve chegar. 71
Figura 46 – Operadores de Cruzamento Utilizados. 72
Figura 47 – Cálculo da Heurística do Movimento. 72

LISTA DE TABELAS

Tabela 1 – Linha de tempo da IA em jogos. 24
Tabela 2 – Genótipos e fenótipos para determinados tipos de problemas. 29
Tabela 3 – Resultados Obtidos dos Cruzamentos Sem Mutação. 59
Tabela 4 – Resultados Obtidos dos Cruzamentos Com Mutação. 59
Tabela 5 – Definições do Caçador. 61
Tabela 6 – Definições do Feiticeiro. 62
Tabela 7 – Definições do Arqueiro. 63
Tabela 8 – Definições do Cavaleiro. 63
Tabela 9 – Resultados Obtidos 74

LISTA DE ABREVIATURAS E SIGLAS

2D	Bidimensional
3D	Tridimensional
AG	Algoritmo Genético
IA	Inteligência Artificial
NPC	*Non Player Character*
TI	Tecnologia da Informação
FPS	Frames Per Second

SUMÁRIO

1 INTRODUÇÃO 15

2 OBJETIVOS 18

3 A HISTÓRIA DOS JOGOS 19

3.1 UTILIZAÇÃO DA INTELIGÊNCIA ARTIFICIAL NOS JOGOS 23

4 DARWIN E OS ALGORITMOS GENÉTICOS 25

4.1 O PRINCÍPIO DA SELEÇÃO NATURAL 25

4.2 ALGORITMOS GENÉTICOS 26

4.2.1 População e Indivíduos 28

4.2.2 Função de Avaliação 29

4.2.3 Seleção 30

4.2.4 Reprodução 31

4.2.5 Mutação 34

4.2.6 Atualização 35

4.2.7 Finalização 35

5 ADOBE FLASH 36

5.1 APRESENTAÇÃO 36

5.2 JOGOS DESENVOLVIDOS UTILIZANDO ADOBE FLASH 36

5.3 INTERFACE 39

5.3.1 Projetos Abertos 40

5.3.2 Biblioteca 41

5.3.3 Linha do Tempo 41

5.3.4 Propriedades 42

5.3.5 Barra de Ferramentas 43

5.3.6 Palco 43

6 METODOLOGIA 45

7 O JOGO - A JORNADA DO REINO 46

7.1 DESCRIÇÃO 46

7.2 ROTEIRO 46

7.3 PERSONAGENS 47

7.3.1 Aliados 47

7.3.2 Inimigos 49

7.4 A UTILIZAÇÃO DO ALGORITMO GENÉTICO NO CONTROLE DO MOVIMENTO DOS PERSONAGENS 51

7.5 RESULTADOS OBTIDOS 58
8 TRABALHOS FUTUROS 61
8.1 DESENVOLVIMENTO DAS CLASSES DE PERSONAGENS 61
8.1.1 O Caçador 61
8.1.2 O Feiticeiro 62
8.1.3 O Arqueiro 62
8.1.4 O Cavaleiro 63
8.2 AMPLIAÇÃO DO CONTEÚDO 63
8.3 MIGRAÇÃO DE 2D PARA 3D 64
9 CONSIDERAÇÕES FINAIS 65
BIBLIOGRAFIA 66
APENDICE A 68

1 INTRODUÇÃO

O entretenimento é uma das áreas da TI que está em grande crescimento, principalmente no Brasil, devido ao aumento do número de pessoas que fazem uso deste tipo de aplicação, incluindo não somente crianças e jovens, mais também muitos adultos (ABRAGAMES, 2009).

Os jogos, ao contrário de que muitos pensam, não são um tipo de aplicação que é somente usado para o entretenimento, atualmente eles são utilizados nas mais diversas áreas de conhecimento. Utilizam-se jogos para ensinar crianças a aprender coisas novas e tornar o aprendizado mais interessante, os jogos também são utilizados para o tratamento de fobias, tratamentos fisioterapêuticos e até mesmo no tratamento de alguns tipos de síndromes como a de *Alzheimer*, por exemplo. Também são utilizados como poderosos simuladores dos mais diversos tipos como: voo, corrida e até mesmo de combate entre outros (ABRAGAMES, 2009).

Segundo Abragames (2009), uma pesquisa recente feita pelo Ibope[1] relata que cerca de sete milhões de brasileiros usam o seu tempo livre na internet para jogar. Esta mesma pesquisa revela que pessoas de 18 a 35 anos, jogam cerca de 18 horas por semana. Desta forma, o desenvolvimento de jogos para computadores ganha cada vez mais importância no aspecto econômico. Este fato desperta o interesse não só das produtoras de jogos como também das empresas de publicidade que veem neste tipo de aplicação um novo ramo de negócio: o desenvolvimento de propaganda dentro de jogos.

Visando este grande crescimento da área e o fato de que cada vez mais os usuários de jogos estão em busca de novidades e interatividade dentro destas aplicações, a proposta deste projeto é estudar e desenvolver maneiras de aumentar este grau de interatividade, sobretudo no que se refere ao controle automático dos personagens.

Grande parte dos jogos possuem personagens virtuais, também chamados de Personagens Não Jogáveis (*Non Player Character,* NPC), estes nada mais são que agentes inteligentes virtuais que agem de modo autônomo, aumentando o grau de interatividade proporcionado pelo jogo. Segundo Cordenonsi & Loy (2003), agentes

[1] Instituto Brasileiro de Opinião Pública e Estatística.

inteligentes são importantes ferramentas para a resolução de problemas em ambientes dinâmicos e/ou complexos. Da mesma forma, podem ser usados quando o tempo de resposta é algo crucial.

A geração de personagens virtuais com comportamentos inteligentes é um dos desafios mais almejados e mais difíceis de serem aplicados em simulações computacionais (POZZER & FURTADO, 2003).

Várias teorias da área de computação podem ser aplicadas para melhorar essa interatividade, entre elas, os Algoritmos Genéticos (AGs). Segundo Russel & Norvig (2004) os AGs são implementados através da simulação de uma população de soluções que passa por melhorias no decorrer de seu desenvolvimento, evoluindo a cada ciclo. Os AGs são usados no desenvolvimento de diversos jogos, sobretudo envolvendo ações que envolvem algum tipo de otimização (RUSSEL & NORVIG, 2004). Através deles, o jogador pode, por exemplo, traçar rotas e definir suas ações, aperfeiçoando-as com o passar do tempo.

Segundo Silveira & Barone (1998), recentemente os AGs tem se mostrado muito eficazes ao serem aplicados em problemas de otimização. Os AGs, por imitarem o processo de seleção natural para resolver problemas, são capazes de estar em constante melhoramento de seus resultados assim como mostra a teoria Darwiniana sobre a evolução das espécies (DARWIN, 1859).

Além da questão da interatividade, para que um jogo se torne mais interessante para o usuário, a sua interface deve ser atrativa, divertida e agradável de utilizar, uma vez que seu principal objetivo é proporcionar entretenimento para as pessoas. Os jogos computadorizados precisam criar a sensação de imersividade nos usuários, tal característica obtida pela combinação de aspectos artísticos e tecnológicos (CLUA & BITTENCOURT, 2005).

Assim, considerando o aspecto computacional, tais aplicações requerem a adoção de sofisticadas técnicas que na maioria das vezes representam o “estado da arte” das pesquisas em Ciência da Computação (CLUA & BITTENCOURT, 2005).

O desenvolvimento de jogos é uma atividade multidisciplinar, envolvendo diversas áreas da computação (algoritmos, computação gráfica, redes de computadores e inteligência artificial), matemática, física e outras ciências. Segundo Battaiola (2000), por conta desta multidisciplinaridade, o desenvolvimento de jogos computadorizados torna-se uma área fascinante para o desenvolvimento de aplicações técnico-científicas.

Para suprir grande parte desta demanda computacional, os Motores de Jogos (*Game Engines)* simplificam muito a criação de um jogo. Para o desenvolvimento deste projeto, foi utilizada a ferramenta *Adobe Flash CS5.5*[2], que, apesar de não ser considerada um motor de jogo, a mesma apresenta várias ferramentas que deixam o desenvolvimento de um jogo bidimensional (2D) mais simples e rápido além de possuir uma grande gama de recursos e funcionalidades, a mesma funciona em diversas plataformas diferentes.

O projeto desenvolvido foi um protótipo de um jogo onde o principal objetivo é a utilização dos AGs para auxiliar os NPCs a tomarem decisões relacionadas à escolhas de seus movimentos.

[2] A ferramenta Adobe Flash pode ser encontrada para download via http://www.adobe.com

2 OBJETIVOS

O presente trabalho tem como objetivo principal a aplicação de Algoritmos Genéticos no mapeamento e execução de funções relacionadas à movimentação de NPCs em ambientes dinâmicos.

Os objetivos específicos desta pesquisa são:

- Pesquisar e definir os conceitos que envolvem algoritmos genéticos e sua utilização no desenvolvimento de jogos;
- Pesquisar e definir etapas que envolvem o desenvolvimento de um jogo utilizando o Adobe Flash;
- Abordar a integração do Adobe Flash com outras ferramentas de desenvolvimento;
- Analisar o desempenho dos NPCs através dos algoritmos genéticos aplicados no resultado de suas ações.

3 A HISTÓRIA DOS JOGOS

Desde os primeiros computadores desenvolvidos, cientistas e estudantes de informática vêm estudando formas de como criar uma tecnologia capaz de proporcionar diversão e entretenimento para as pessoas. Por exemplo, em 1949 o engenheiro Ralph Baer tentou criar uma televisão interativa com jogos, mas devido à falta de recursos tecnológicos da época, o projeto não foi realizado. Em 30 de Julho de 1961 um grupo de estudantes do *Massachusetts Institute of Technology* (MIT) formado por Steve Russel, Dan Edwards, Alan Kotok, Peter Sampson e Martin Graetz utilizando o computador PDP – 1 (Figura 1) testava pela primeira vez um jogo eletrônico, o "*Spacewar!*" desenvolvido em um enorme computador (UOL JOGOS, 2004).

Figura 1 – Computador PDP-1 usado na programação do "*Spacewar!*".
Fonte: UOL JOGOS, 2004.

Alguns anos mais tarde, em 1971, a empresa *Magnavox* compra o projeto do engenheiro Baer da *Sanders Associates*, desenvolve o primeiro videogame para ser conectado à televisão (TV), o *Odyssey*. Enquanto isso, Nolan Bushnell adapta o "*Spacewar!*" de Russel, criando o "*Computer Space*", o primeiro *arcade* do mundo. Como os mainframes eram caros e ocupavam muito espaço, Bushnell criou uma máquina exclusivamente para se jogar o "*Spacewar!*" (UOL JOGOS, 2004).

Na década de 80, a grande sensação da época foi um lançamento da Nintendo, o "*Donkey Kong*", onde o herói apelidado de "*Jumpman*" deveria salvar sua namorada Paulin das grandes mãos de um gorila raivoso. Esta década foi marcada por vários outros nomes como "*Pac Man*", "*Ms Pac Man*", "*Pitfall*" (Figura

2), “*E.T.*” entre outros como o grande sucesso “*Mario Bros*”, “*Donkey Kong Jr*.” e “*Street Fighter*” (UOL JOGOS, 2004).

Figura 2 – Jogo Pitfall de 1982 para a plataforma Atari.
Fonte: Activision.

Na década de 90 a Nintendo lança a versão americana de *Super Famicon*, rebatizada como Super NES. Outros grandes sucessos surgiram nesta época como “*Sonic*”, “*Street Fighter II*” e o grande jogo das Tartarugas Ninjas o “*Turtles in Time*” (Figura 3) (UOL JOGOS, 2004).

Figura 3 – Turtles in Time de 1991 para Arcade e Super NES.
Fonte: Konami.

Nos anos 2000 uma grande revolução com relação à tecnologia dos videogames e jogos eletrônicos aconteceu: os videogames atingiram tamanhos portáteis, como *Nintendo DS*, *Game Boy Advanced*, e o *PSP* (Figura 4). Também

surgiram consoles com um grande poder de processamento como o *Xbox* e o *Playstation II*. Nessa mesma década a tecnologia tridimensional ganhou um grande poder e vários grandes clássicos ganharam sua versão em 3D (UOL JOGOS, 2004).

Figura 4 – Sony PSP lançado no ano de 2004.
Fonte: Sony.

Até hoje novos jogos são lançados, novas plataformas são desenvolvidas e a cada vez mais existe uma demanda tecnológica para atendê-los.

Após os anos 2000 houve três lançamentos que revolucionaram completamente a indústria de jogos em todo o mundo. O primeiro destes lançamentos foi o *Nintendo Wii* (Figura 5), que causou um grande impacto por possibilitar ao jogador fazer os movimentos do seu personagem sem utilizar botões, apenas movimentando o controle de um lado para o outro, fazendo com que o personagem simule seus movimentos (UOL JOGOS, 2004).

Figura 5 – Nintendo Wii lançado no ano de 2006.
Fonte: Nintendo.

Após o lançamento do *Wii* em 2006 se passaram quatro anos até o próximo lançamento que iria causar impacto maior que o do *Nintendo Wii*, a empresa *Microsoft* lançou uma ferramenta extra para o *Xbox 360*, o *Kinect* (Figura 6). Revolucionário por não precisar de controles, o jogador consegue controlar seu personagem dentro do jogo apenas fazendo movimentos corporais, gerando assim uma grande sensação de imersão dentro do ambiente dos jogos (UOL JOGOS, 2004).

Figura 6 – Xbox Kinect lançado no ano de 2010.

Fonte: Microsoft.

No mesmo ano a gigante Sony lançou o *PlayStation Move* (Figura 7), um tipo diferente de controle para o *PlayStation*, possibilitando o jogador realizar movimentos com o controle e reproduzir os mesmos em seu personagem no jogo, bem similar ao *Nintendo Wii* (UOL JOGOS, 2004).

Figura 7 – Playstation Move lançado no ano de 2010.

Fonte: Sony.

3.1 UTILIZAÇÃO DA INTELIGÊNCIA ARTIFICIAL NOS JOGOS

Segundo Fujita (2005), vários estudiosos de Inteligência Artificial, tais como Luger (2004), Russel & Norvig (2004), entre outros, consideram o filósofo grego Aristóteles (384-322 A.C) como um dos principais percursores dos estudos de Inteligência Artificial. Em uma de suas obras, Aristóteles introduziu o conceito de Silogismo que nada mais é que utilizar de certos argumentos para provar a veracidade de algo ou fato, por exemplo, “Sócrates é homem; Todos os homens são mortais; Portanto Sócrates é mortal”.

Segundo Luger (2004), um dos principais fatores que proporcionou o entendimento da inteligência artificial foi o fato de que o ser humano passou a interpretar as coisas como elas realmente são e não como elas parecem ser. E com todos os estudos feitos a respeito da Inteligência Artificial proporcionou que a mesma evoluísse com o passar dos anos.

Segundo Fujita (2005) e Schwab (2004), pode-se dizer que esta evolução da Inteligência Artificial proporcionou muito a evolução dos jogos, com isso, vem surgindo cada vez a necessidade de utilização de técnicas que melhorassem a experiência dos usuários com a máquina, sobretudo, no sentido de possibilitar a criação de agentes inteligentes em jogos com comportamentos apropriados e inteligentes num determinado contexto.

Dessa forma, várias técnicas de Inteligência Artificial e sua aplicação nos jogos passaram a ser estudadas, contribuindo para o desenvolvimento dos algoritmos de inteligência artificial utilizados nestes tipos de aplicações.

Segundo Schwab (2004), quando os primeiros jogos começaram a ser desenvolvidos, a inteligência artificial programada neles era também conhecida como “programação de jogabilidade”, pois não havia nada de inteligente no comportamento dos personagens controlados pelo computador.

A Tabela 1 mostrada a seguir apresenta a evolução gradual da utilização da inteligência artificial em jogos.

Tabela 1 – Linha de tempo da IA em jogos.

Ano	Descrição	IA utilizada
1962	Primeiro jogo de computador, *Spacewar*, para 2 jogadores.	Nenhuma
1972	Lançamento do jogo *Pong* para 2 jogadors.	Nenhuma
1974	Jogadores tinham que atirar em alvos móveis em *Pursuit e Qwak*.	Padrões de Movimento
1975	*Gun Fight* lançado, personagens com movimentos aleatórios.	Padrões de Movimento
1978	*Space Invaders* contém inimigos com movimentos padronizados, mas também atiram contra o jogador.	Padrões de Movimento
1980	O jogo *Pac-man* conta com movimentos padronizados dos inimigos, porém cada fantasma (inimigo) tem uma "personalidade" sobre o modo em que caça o jogador.	Padrões de Movimento
1990	O primeiro jogo de estratégia em tempo real, *Herzog Wei*, é lançado. Junto, os jogadores puderam noticiar uma péssima busca de caminho.	Máquina de Estados
1993	*Doom* é lançado como primeiro jogo de tiro em primeira pessoa.	Máquina de Estados
1996	*BattleCruiser: 3000AD* é publicado como primeiro jogo a utilizar redes neurais.	Redes Neurais
1998	*Half-Life* é lançado e analisado como a melhor IA em jogos até a época, porém, o jogo utiliza IA baseada em *scripts*.	Máquina de Estados / *Script*
2001	O jogo *Black & White* é alvo da mídia a respeito de como as criaturas do jogo aprendem com as decisões feitas pelo jogador. Utiliza redes neurais, *reinforcement* e *observational learning*.	Diversos

Fonte: SCHWAB, 2004. Livro sobre a Inteligência Artificial programada para jogos.

Conforme pode ser visto na Tabela 1, os primeiros jogos eletrônicos utilizavam como inteligência apenas padrões de movimentos ou movimentos repetitivos e/ou aleatórios para os personagens controlados pelo computador. Esse fato é explicado principalmente devido à falta de memória e limitação existente na velocidade de processamento (SCHWAB, 2004).

Sobretudo a partir da década de 1990, outras técnicas e algoritmos passaram a ser utilizados pelos desenvolvedores de jogos, dentre essas técnicas pode-se citar: máquina de estados, sistemas baseados em regras, algoritmos de busca e algoritmos genéticos (DALMAU, 2004).

Os algoritmos genéticos, por exemplo, podem ser usados em jogos para a geração de uma população, criando diferentes indivíduos de acordo com um DNA virtual, sendo esse representado por um vetor de valores, cada um sendo um parâmetro da espécie a ser modelada, além disso, também podem ser usados para mutação ou evolução de personagens (DALMAU, 2004).

4 DARWIN E OS ALGORITMOS GENÉTICOS

Os algoritmos genéticos atuam de uma forma que segue o conceito da seleção natural proposto por Charles Darwin e Alfred Wallace em 1858 que, resumidamente, propõe que as espécies sempre tendem a procurar os melhores parceiros para se reproduzirem e com isto a prole resultante da reprodução sempre tende a nascer com as melhores características de seus pais, garantindo, assim, a sobrevivência de sua espécie (DARWIN, 1859).

Segundo Russel & Norvig (2004) e Luger (2004), um algoritmo genético é uma variante de busca em feixe estocástica, na qual os estados sucessores são gerados pela combinação de dois estados pais, em vez de serem gerados pela modificação de um único estado.

Um algoritmo genético pode ser definido também como uma técnica utilizada para encontrar soluções aproximadas para problemas de otimização de busca, pertencentes à classe de algoritmos evolutivos (GOLDBERG, 1989).

4.1 O PRINCÍPIO DA SELEÇÃO NATURAL

Segundo o Darwing (1859), existem quatro preceitos básicos para que seja possível este processo de evolução:

1. Indivíduos da mesma espécie em uma disputa incessável por recursos limitados contidos no ambiente em que vivem.
2. Estar mais apto a sobreviver em seu próprio ambiente, disputando diariamente com os concorrentes de sua própria espécie, provando que o mesmo com suas características específicas possuem maior chance de sobrevivência no meio em que vive.
3. Os indivíduos que demonstram serem mais adaptados tem uma maior probabilidade de sobrevivência e reprodução.
4. Tendo em vista que no processo de reprodução diversas características de seus pais são passadas para seus filhos, quão maior a frequência em que as espécies se reproduzem mais significantemente suas características estarão presentes nas gerações subsequentes.

4.2 ALGORITMOS GENÉTICOS

A funcionalidade dos algoritmos genéticos é baseada no conceito da seleção natural proposto por Darwin (1859), onde um conjunto de resultados é analisado e, com base em seus conteúdos, novos resultados melhorados podem ser gerados. Este processo é repetido inúmeras vezes até que os resultados obtidos formem uma população melhor que a população inicial. Esta é uma das técnicas da inteligência artificial que pode ser aplicada nas mais diversas áreas de conhecimento (RUSSEL & NORVIG, 2004).

Segundo Holland (1975), o funcionamento de um algoritmo genético com suas diversas etapas pode ser representado graficamente em forma de diagramas, assim como mostra a Figura 8.

Primeiramente o algoritmo inicia toda a população dos indivíduos, em seguida é feita uma avaliação dos indivíduos, selecionando os melhores reprodutores e realizando o cruzamento entre eles, gerando novos indivíduos. Alguns indivíduos resultantes passam por um processo de mutação e, em seguida, os mesmos são avaliados e população é atualizada com os indivíduos resultantes. Caso o algoritmo satisfaça as condições de parada ele encerra a execução, caso contrário o algoritmo volta a avaliar a população e, assim, sucessivamente.

Figura 8 – Estrutura de um Algoritmo Genético tradicional.

Fonte: LUCAS, 2002.

Segundo Medina & Müller (2009), um AG tem uma simples representação, o ponto de partida para o desenvolvimento, a Figura 9, mostra como pode ser codificado uma função para a aplicação de um algoritmo genético.

```
função ALGORITMO-GENETICO (população, FN-FITNESS) retorna um individuo
        entradas: população // um conjunto de indivíduos,
                  FN-FITENESS //uma função que mede a adaptação de um individuo
        repita
                nova-população ← conjunto vazio;
                para i deI até TAMANHO(população) faça
                        x ← SELEÇÃO-ALEATORIA (população, FN-FITNESS);
                        y ← SELEÇÃO-ALEATORIA (população, FN-FITNESS);
                        filho ← REPRODUZ (x,y);
                        se (pequena probabilidade aleatória) então
                                filho ← MUTAÇÃO (filho);
                        fim se
                        adicionar filho a nova-população;
                fim para
        até algum individuo estar adaptado o suficiente ou até ter decorrido tempo suficiente;
        retornar o melhor individuo em população, de acordo com FN-FITNESS;
fim função
função REPRODUZ (x,y) retorna um individuo
        entradas: x,y //indivíduos pais;
        n ← COMPRIMENTO (x);
        c ← numero aleatório de I a n;
        retornar CONCATENA(SUBCADEIA (x,I,c), SUBCADEIA (t,c+I,n));
fim função
```

Figura 9 – Representação em código de um algoritmo genético.

Fonte: MEDINA & MÜLLER, 2009.

Os AGs simulam a natureza em um de seus mais fortes atributos: a adaptabilidade. Visto que a representação e a avaliação das possíveis soluções são as únicas partes (de um considerável conjunto de operações utilizadas em seu funcionamento) que, obrigatoriamente, requisitam conhecimento dependente do domínio do problema abordado (WHITLEY, 2000).

Os AGs possuem um alto grau de paralelismo e este grau pode ser facilmente verificado, pois cada indivíduo da população existe como um ente isolado e é avaliado de forma independente (WALL, 2000).

Segundo Geyer-Schultz (1997) e Michalewicz (1999), ao contrário dos outros métodos de busca de valores ótimos, os algoritmos genéticos não apresentam um comportamento determinístico. Um AG tradicional opera ignorando o significado das estruturas que ele manipula e qual a melhor maneira de trabalhar sobre estas.

De acordo com Davis (1991), Burke (1995) e Geyer-Schultz (1997), por constituir em método de busca cega, um algoritmo genético tradicional tende a apresentar um desempenho menos adequado que alguns tipos de busca heurística orientadas ao problema. Para resolver tal desvantagem, a tática mais utilizada é a hibridação, onde heurísticas provenientes de outras técnicas são incorporadas aos AGs.

Ao contrário de muitos outros métodos de busca, os AGs facilitam a codificação de problemas com diversos tipos de restrição, mesmo que elas apresentem graus diferentes de importância (BARBOSA, 1996).

Um algoritmo genético possui diversas propriedades e características que são relatadas nas seções seguintes.

4.2.1 População e Indivíduos

Em algoritmos genéticos existe um conjunto chamado de população que possui vários estados, gerados aleatoriamente, sendo que cada estado representa um indivíduo, cada um com determinadas características.

Os indivíduos podem ser considerados a parte mais importante de um algoritmo genético, pois neles ficam codificados todas as possíveis soluções para determinados problemas que o algoritmo genético visa resolver.

É muito comum em algoritmos genéticos serem usados nomes e definições utilizadas na área de Genética, tais como genótipo, fenótipo e cromossomo, onde cada um destes termos representa uma característica do algoritmo genético.

Como é mostrado na Tabela 2, um indivíduo é representado por um cromossomo, portanto se resume a um conjunto de genes (0’s e 1’s). O genótipo é exatamente este conjunto de genes que o individuo possui, já o fenótipo pode ser

considerado o conjunto de características do resultado obtido pela decodificação dos seus genes.

Tabela 2 – Genótipos e fenótipos para determinados tipos de problemas.

Genótipo	Fenótipo	Problema
0010101001110101	10869	Otimização Numérica,
CGDEHABF	Comece pela cidade C, depois passe pelas cidades G, D, E, H, A, B e termine em F.	Caixeiro Viajante.
$C_1R_4C_2R_6C_4R_1$	Se condição 1 (C_1) execute regra 4 (R_4), se (C_2) execute (R_6), se (C_4) execute (R_1).	Regras de aprendizado para agentes.

Fonte: LUCAS, 2002.

Assim, como mostra a Tabela 2, o genótipo é o responsável por armazenar os dados na estrutura de dados, englobando cada gene de um indivíduo, já o fenótipo é o processo de decodificação do genoma do indivíduo. Os indivíduos possuem também, grau de aptidão e grau de adaptação, o grau de aptidão representa o quão bem a resposta apresentada por um indivíduo soluciona o problema proposto e o grau de adaptação é relativo ao nível de adaptação de um indivíduo relativamente à população a qual o mesmo pertence.

4.2.2 Função de Avaliação

A função de avaliação serve para avaliar a aptidão dos indivíduos da população, para isso é feita uma análise para saber como os indivíduos responderam ao problema proposto. Para cada problema existe uma função de avaliação diferente. A função de avaliação para um problema de resolução de melhor caminho é diferente da função de avaliação para o problema de escalonamento (GOLDBERG, 1989).

Segundo Goldberg (1989), para que seja possível fazer a avaliação de um indivíduo, são analisados os graus de aptidão e adaptação, pois a seleção é totalmente dependente de como o indivíduo se comporta perante o problema e quão bem foi seu desempenho no resultado do mesmo.

Segundo Lucas (2002), em muitos casos, calcular com exatidão completa o grau de adaptação dos indivíduos pode ser uma tarefa complexa e, se levarmos em conta que esta operação é massivamente repetida ao longo do processo de evolução, seu custo pode ser consideravelmente alto. Em tais situações é comum o

uso de funções não determinísticas que não avaliam a totalidade das características do indivíduo, operando apenas sobre uma amostragem destas.

4.2.3 Seleção

Os indivíduos são selecionados de acordo com a sua aptidão para fazerem a reprodução, gerando novos indivíduos que novamente serão avaliados por sua aptidão e, assim, por diante.

Segundo Dawkins (1996) *apud* Lucas (2002) a seleção é um processo:

1. Dirigido, onde ao contrário da mutação que ocorre de forma aleatória, a seleção ocorre de forma determinística. O indivíduo só vai conseguir sobreviver em um ambiente e se reproduzir no mesmo se for capaz de acordo com suas características e habilidades de reagir adequadamente a todas as ocorrências do meio em que vive.
2. Cumulativo, onde os benefícios obtidos no processo de seleção passam de uma geração para a outra. Este fator se combinado com a não aleatoriedade do processo de seleção garante a possibilidade do surgimento de organismos mais aptos e complexos, tornando possível a existência da vida como conhecemos hoje.

Segundo Goldberg (1989), Wall (2000) e Geyer-Schultz (1997) apud Lucas (2005), o uso de um método de escala sobre o valor de adaptação de cada indivíduo normalmente é útil por reduzir a probabilidade de convergência prematura. Diversas funções podem ser aplicadas sobre a adaptação.

Não Escalar (*No scaling*): O valor de retorno da função objetivo é usado sem nenhum tipo de alteração (GOLDBERG, 1989).

Escala Linear (*Linear scaling*): O valor de retorno da função objetivo (fO (x)) sofre a seguinte alteração:

$$f_l(f_O(x)) = f_O(x) \times a + b,$$

Onde,

$$a = (c-1) \times \frac{med}{\Delta} \text{ e } b = med \times \frac{max - c \times med}{\Delta}, \text{ se } \Delta \neq 0,$$

ou *a = 1*, *b = 0* em caso contrário, *max* representa o maior valor de adaptação encontrado, *med* a média de adaptação da população e *c* é uma constante pré-definida pertencente aos reais e maior do que *1*. Valores negativos de adaptação não são aceitos por este método (GOLDBERG, 1989).

Escala por Truncamento *Sigma* (*Sigma truncation scaling*): Um múltiplo do desvio médio (denotado por d_m) é subtraído dos valores de adaptação. Todos os valores negativos são alterados para zero. A fórmula para seu cálculo é a seguinte:

$$f_{st}(f_O(x)) = f_O(x) - med - c \times d_m,$$

onde *c* é uma constante pré-definida pertencentes ao conjunto dos reais e maior do que zero (WALL, 2000).

Escala pela Lei da Potência (*Power law scaling*): O valor de adaptação do indivíduo é elevado a uma constante pré-definida *k*:

$$f_{pl}(f_O(x)) = f_O(x)^k$$

A Escala pela Lei da Potência não aceita valores negativos de adaptação (GEYER-SCHULTZ, 1997).

Função de Fitness - A função de fitness é responsável por fazer a seleção dos melhores indivíduos, uma função de fitness deve retornar valores mais altos para os estados melhores (LUCAS, 2005).

4.2.4 Reprodução

Logo após a seleção dos indivíduos em pares, ocorre um processo chamado cruzamento (*crossover*), onde partes dos genes dos pais são combinadas para a formação dos filhos.

Segundo Goldberg (1989) a formação dos pares de indivíduos pode ser feita utilizando os seguintes métodos:

- Escolha aleatória: Os pares de indivíduos são escolhidos aleatoriamente;
- Endogamia (*Inbreeding*): Os indivíduos parentes são combinados para geração da nova população;

- Reprodução de Linha (*Line breeding*): Um indivíduo com um alto grau de aptidão é cruzado com uma subpopulação e os filhos são escolhidos como pais;
- Exogamia (*Outbreeding*): Indivíduos que codificam fenótipos diferentes são combinados;
- Autofecundação (*Self-fertilization*): O indivíduo é combinado consigo mesmo;
- Acasalamento Ordenado Positivo (*Positive assortative mating*): Indivíduos semelhantes são combinados;
- Acasalamento Ordenado negativo (*Negative assortative mating*): Indivíduos diferentes são combinados.

Segundo Goldberg (1989) apud Lucas (2002), para que haja a possibilidade de cruzamento é preciso haver operadores de cruzamento como, por exemplo, uma seleção por máscara, que seria representada por um vetor cujos elementos possam assumir valores binários e que possua um comprimento igual ao dos cromossomos a serem combinados.

Seu uso pelo operador se daria pelo algoritmo mostrado na Figura 10.

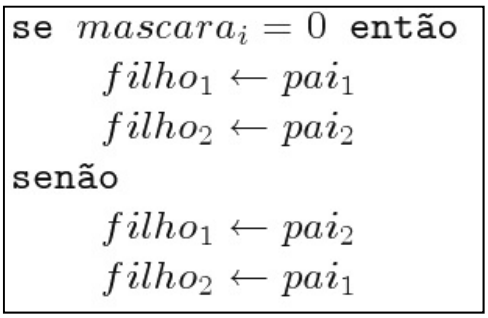

Figura 10 – Algoritmo de uso de uma máscara de cruzamento.

Fonte: LUCAS, 2002.

De acordo com Goldberg (1989), os operadores tradicionais de cruzamento para genomas de comprimento fixo mais conhecidos são:

Cruzamento de um ponto (1PX): Dados dois genomas x e y de comprimento *lg*, sorteia-se um número p qualquer tal que *0* < *p* < *lg*, o primeiro filho *f0* receberá todos os genes *x* de *1* até *p* e todos os genes *y* de *p* + *1* até *ly*, e o segundo filho o inverso.

A máscara de cruzamento seria uma série de 0's sucedidas de *1's* pertencente ao conjunto $\{0^n1^m \mid n = 0, m = 0, n + m = lg\}$ (GOLDBERG, 1989).

A Figura 11 mostra a exemplificação desta máscara para o cruzamento de um ponto.

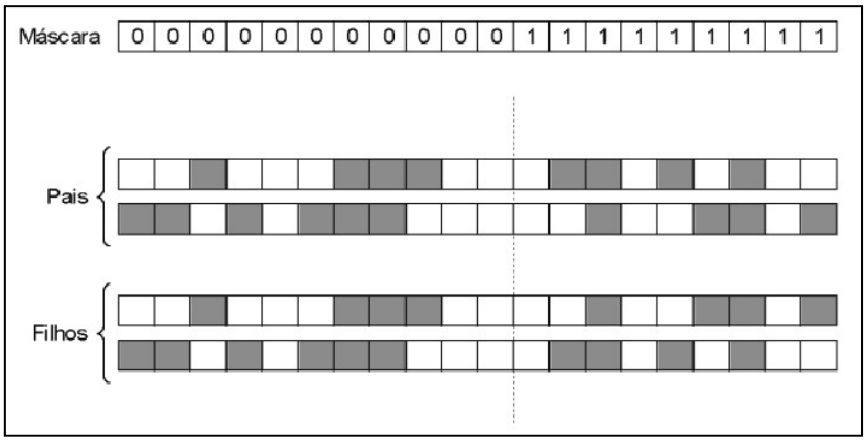

Figura 11 – Operador de cruzamento de um ponto.

Fonte: LUCAS, 2002.

Cruzamento multiponto (MPX): O cruzamento multiponto é uma generalização do operador de um ponto. Nele é sorteado um número fixo *n* de pontos de corte. Como regra geral, uma operação de cruzamento de n pontos pode ser vista como um caso especial de uma de $m > n$ pontos em que um ou mais $(m – n)$ pontos de corte selecionados se localizam no final do cromossomo (o ponto de índice $i = lg$) (GOLDBERG, 1989).

A Figura 12 mostra o funcionamento do cruzamento multiponto, onde um operador com n pontos de cruzamento apresentaria uma máscara de cruzamento com n mudanças em sua sequencia de *0's e 1's*.

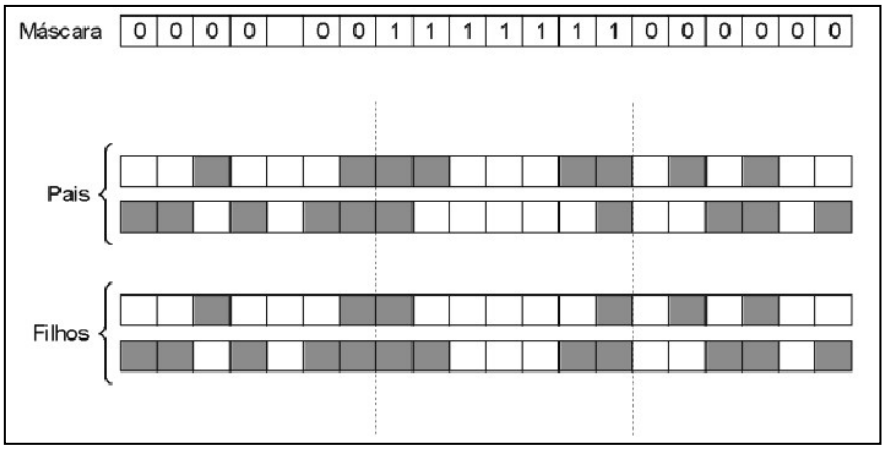

Figura 12 – Operador de cruzamento multiponto.

Fonte: LUCAS, 2002.

Cruzamento segmentado (SX): O cruzamento segmentado funciona de maneira semelhante ao multiponto, a única diferença é que a cada vez que vai ser executado ele sorteia o número de pontos de corte (GOLDBERG, 1989).

Cruzamento uniforme (UX): Para cada gene a ser preenchido nos cromossomos filhos, o operador de cruzamento uniforme sorteia de qual dos pais este deve ser gerado. A máscara de cruzamento de tal operador é uma sequência qualquer de 0’s e 1’s (GOLDBERG, 1989).

Assim como mostrado na Figura 13 a máscara contendo os 0’s e 1’s de forma não padronizada.

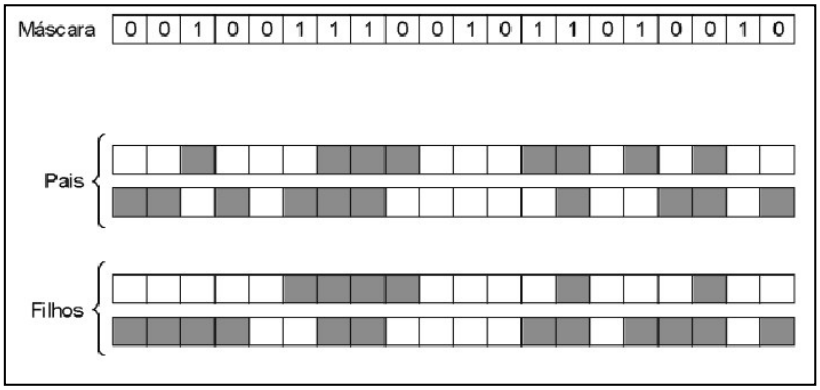

Figura 13 – Operador de cruzamento uniforme.

Fonte: LUCAS, 2002.

4.2.5 Mutação

Segundo Whitley (2000), a mutação é um fator que opera sobre os indivíduos resultantes do processo de cruzamento e com uma probabilidade pré-determinada

efetua algum tipo de alteração em sua estrutura com o intuito de melhorar a população de indivíduos. A importância deste operador reside no fato de que uma vez bem escolhido seu modo de atuar garante que diversas alternativas serão exploradas, mantendo assim um nível mínimo de abrangência na busca.

Dentre os operadores de mutação os mais comuns são:

- Mutação *flip*: Cada gene a sofrer mutação recebe um valor sorteado do alfabeto válido;
- Mutação por troca (*swap mutation*): São sorteados n pares de genes e os elementos do par trocam de valor entre si;
- Mutação *creep*: Um valor aleatório é somado ou subtraído do valor do gene.

4.2.6 Atualização

Neste ponto, os indivíduos resultantes do processo de cruzamento e mutação são inseridos na população segundo a política adotada pelo AG.

Na forma mais tradicional deste (chamada corriqueiramente de algoritmo genético simples), a população mantém um tamanho fixo e os indivíduos são criados em mesmo número que seus antecessores e os substituem por completo. Existem, porém, alternativas a essa abordagem: o número de indivíduos gerados pode ser menor, o tamanho da população pode sofrer variações e o critério de inserção pode ser variado (como, por exemplo, nos casos em que os filhos substituem os pais, ou em que estes só são inseridos se possuírem maior aptidão que o cromossomo a ser substituído), ou o conjunto dos n melhores indivíduos pode sempre ser mantido (LUCAS, 2002).

4.2.7 Finalização

Segundo Lucas (2002), a finalização não envolve o uso de nenhum operador genético, ela simplesmente é composta por um teste que dá fim ao processo de evolução caso o AG tenha chegado a algum ponto pré-estabelecido de parada.

Os critérios para a parada podem ser vários, desde o número de gerações já criadas até o grau de convergência da atual população (por convergência, atende-se o grau de proximidade dos valores da avaliação de cada indivíduo da população).

5 ADOBE FLASH

5.1 APRESENTAÇÃO

O desenvolvimento de jogos vem se tornando uma prática bem comum no meio de desenvolvimento de software. Existem várias maneiras de se desenvolver um jogo, seja ele 2D ou 3D. Pode-se desenvolver jogos utilizando linguagens de programação puras como *C/C++* manipulando diretamente bibliotecas como *OpenGL*, pode-se também utilizar frameworks de desenvolvimento como *XNA* da *Microsoft* para a plataforma *.NET*, pode-se utilizar aplicativos para criação de conteúdo interativo que é o caso do *Adobe Flash* ou pode-se também utilizar motores de jogo (*Game Engines*) como *Unreal Engine*, *CryEngine*, *Unity3D* entre outras.

O *Adobe Flash,* inicialmente chamado de *Macromedia Flash,* sempre teve como conceito desenvolver conteúdo interativo para ser utilizado na internet. Com o passar dos anos um *script* que existia no *Macromedia Flash* chamado *Actionscript* que se trata de uma variante do *ECMAScript*, passou de um simples *script* para uma poderosa linguagem de programação orientada a objetos (ADOBE, 2011).

Atualmente o *Adobe Flash* se encontra na sua versão *CS5.5*. O mesmo oferece diversos recursos e ferramentas para se trabalhar diretamente com áudio, vídeo imagens e animações, oferecendo também, recursos para criação de conteúdos dos mais diversos tipos, inclusive jogos (ADOBE, 2011).

Segundo Adobe (2011), um grande ponto positivo do *Adobe Flash* é a questão da portabilidade de seu produto final, como é possível executar o conteúdo desenvolvido no *Adobe Flash* em um navegador de internet, este mesmo conteúdo pode ser exibido em qualquer sistema operacional (SO) que tenha acesso à internet. Ele também disponibiliza a opção para gerar aplicativos que também podem ser executados em SOs de aparelhos móveis, como o *Android* e o *iOS* que é o SO utilizado no *iPhone*.

5.2 JOGOS DESENVOLVIDOS UTILIZANDO ADOBE FLASH

Devido à sua curva de aprendizado pequena e a gama de facilidades que a ferramenta disponibiliza, acaba sendo possível desenvolver todos os gêneros e tipos de jogos possíveis utilizando o *Adobe Flash*, como jogos de corrida, estratégia, tiro,

quebra cabeças, entre outros. Sendo que na atual versão da ferramenta ela também suporta o desenvolvimento de jogos em 3D.

A Figura 14 mostra o jogo *"Cyclomaniacs 2"* que é um jogo de corrida de motocicletas em 2D onde é possível fazer diversas manobras em altas velocidades. Este jogo apesar de parecer simples, ele implementa diversos conceitos de física, isto torna o jogo mais interessante tendo em vista que a velocidade que o personagem está influencia no modo em que o mesmo executará suas manobras. O jogador é capas de ir evoluindo o personagem que joga, deixando-o mais rápido, ágil entre outros atributos. Este jogo foi desenvolvido pela empresa *Turbo Nuke* (ARMORGAMES, 2005).

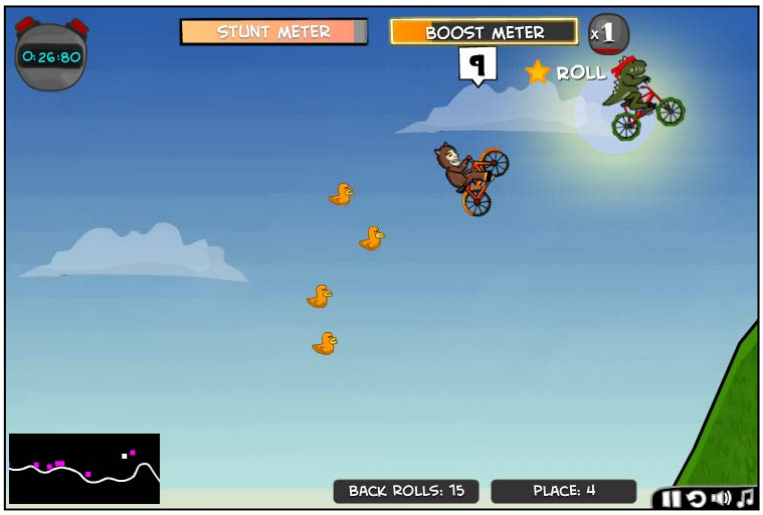

Figura 14 – Imagem do Jogo *"Cyclomaniacs 2"*.
Fonte: ARMORGAMES, 2005.

Já a Figura 15 mostra o jogo *"Arcuz II"* que é um jogo no estilo RPG aonde o personagem vai evoluindo, conseguindo novos equipamentos e ficando cada vez mais forte para proteger seu vilarejo contra as forças obscuras. Neste jogo foi utilizado um sistema de combate em tempo real, onde é possível lutar com vários inimigos simultâneos. O jogador pode escolher entre usar armas, como espada, machados, facas ou a utilização de magias como controle do fogo, da água ou até mesmo usar armas e magias combinadas. Este jogo foi desenvolvido pela empresa *Funnaut* (ARMORGAMES, 2005).

Figura 15 – Imagem do Jogo *“Arcuz II”*.
Fonte: ARMORGAMES, 2005.

A Figura 16 mostra o jogo *“Kingdom Rush”* desenvolvido pela empresa *Ironhide Game Studio*, o jogo se trata de um estilo de estratégia conhecido como Defesa de Torres (*Tower Defence*) aonde os inimigos vão avançando em ondas, e o jogador deve colocar várias torres no cenário sendo que cada torre pode ou não ter uma habilidade especial diferente das outras, tudo isto com o intuito de tentar impedir que o inimigo chegue ao castelo. Para tornar a experiência do jogador mais dinâmica e interativa, o jogo oferece quatro tipos de torres diferentes e com diversos níveis de melhorias para seus ataques (ARMORGAMES, 2005).

Figura 16 – Imagem do Jogo *“Kingdom Rush”*.
Fonte: ARMORGAMES, 2005.

5.3 INTERFACE

O Adobe Flash possui uma interface simples e intuitiva, tornando assim mais simples o desenvolvimento de vídeos, animações, aplicações ou jogos. Sua tela principal possui diversas janelas, cada uma com uma determinada função, sendo que o usuário pode alterar a posição e tamanho, minimizar ou até mesmo fechar e abrir outras janelas conforme a necessidade (ADOBE, 2011).

A Figura 17 mostra a disposição das ferramentas e janelas auxiliares do Adobe Flash em sua interface inicial padrão. Cada uma das ferramentas indicadas será descrita nas subseções seguintes.

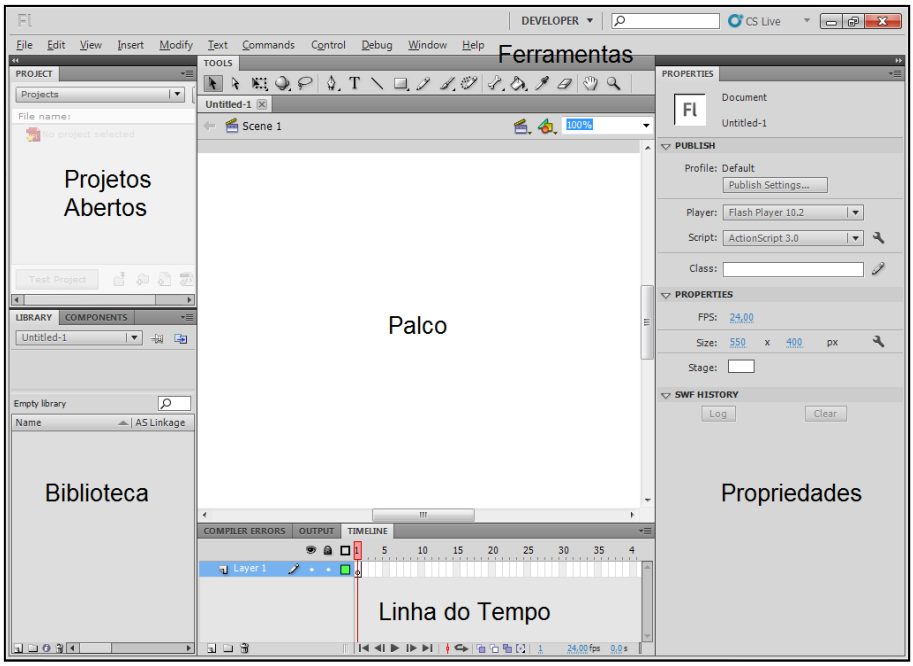

Figura 17 – Interface Padrão para o Desenvolvimento de Aplicações.

Fonte: Do Autor.

5.3.1 Projetos Abertos

Segundo Adobe (2011), esta janela é a responsável por organizar e agrupar os recursos físicos do projeto, ou seja, os arquivos em disco pertencentes ao projeto que está sendo desenvolvido. Esta janela é muito importante pois é através dela que o desenvolvedor visualizará a hierarquia de arquivos e classes do projeto em desenvolvimento. A Figura 18 mostra a estrutura de arquivos montada pela janela de Projetos.

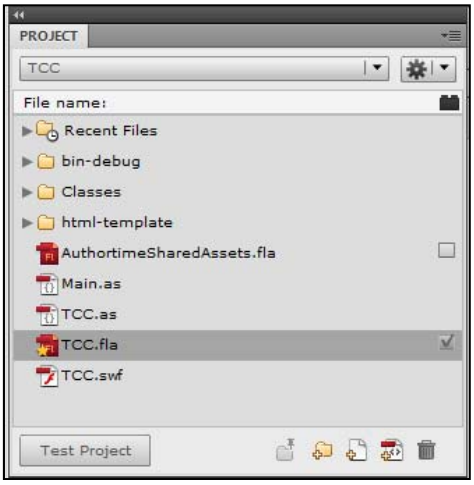

Figura 18 – Apresentação da Janela de Projetos.

Fonte: Do Autor.

5.3.2 Biblioteca

Segundo Adobe (2011), a Biblioteca é responsável por manter organizado e agrupado todos os arquivos do projeto que vão ser incorporados ao aplicativo final. Através da Biblioteca é possível referenciar objetos diretamente no código, assim como também é possível associar um objeto da biblioteca diretamente a uma classe na programação. A Figura 19 mostra um exemplo de organização e associação de uma Biblioteca no Adobe Flash.

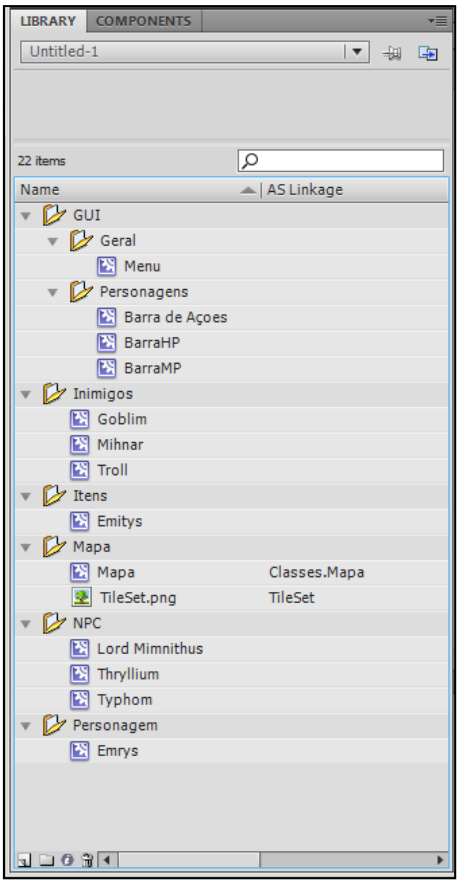

Figura 19 – Apresentação da Biblioteca.

Fonte: Do Autor.

5.3.3 Linha do Tempo

Segundo Adobe (2011), a Linha do Tempo é responsável por executar linearmente tudo que está mapeada nela. É possível trabalhar com animações e movimentos quadro a quadro, interpolação ou transformação de objetos. A Linha do Tempo é muito útil e funcional para se criar animações complexas como movimentos

de personagens ou objetos, por exemplo. Também é possível trabalhar com sincronização de áudio em narração ou som ambiente, tudo através deste recurso.

A Figura 20 mostra um exemplo das informações contidas na Linha do Tempo.

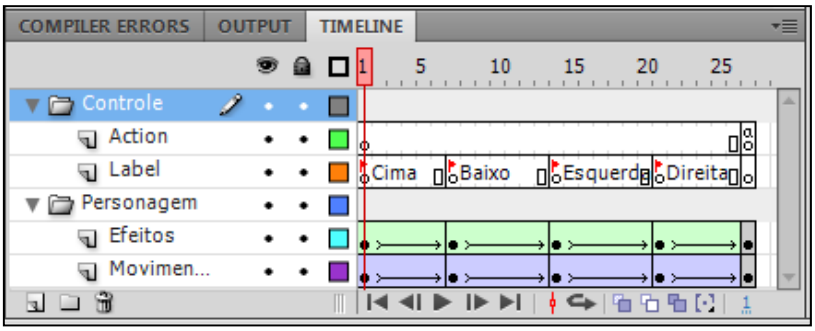

Figura 20 – Apresentação da Linha do Tempo.

Fonte: Do Autor.

5.3.4 Propriedades

Segundo Adobe (2011), esta é a janela responsável por disponibilizar ao usuário todas as informações do objeto selecionado, como largura, altura, posição no eixo x ou no eixo y. Cada tipo de objeto possui um conjunto de propriedades diferentes como, por exemplo, o Palco que possui a propriedade de Quadros por Segundo (FPS) que define a velocidade que a animação ou a aplicação vai ser executada. Componentes de Texto possuem propriedades relacionadas ao tamanho de fonte, tipo de fonte, padrão de entrada de dados, entre outros. A Figura 21 mostra a janela de Propriedades.

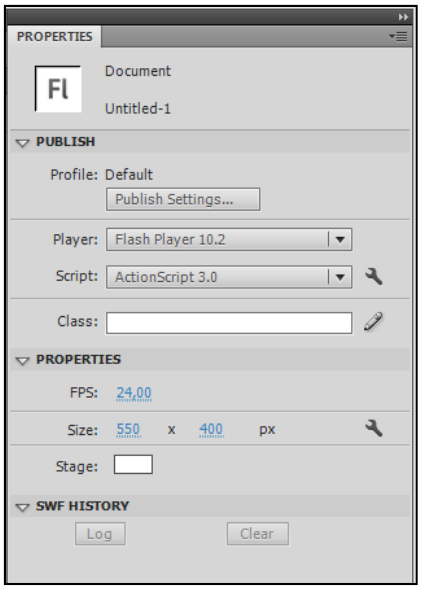

Figura 21 – Apresentação da Janela de Propriedades.

Fonte: Do Autor.

5.3.5 Barra de Ferramentas

Segundo Adobe (2011), este recurso (Figura 22) é a janela onde se encontram todas as ferramentas disponíveis no Adobe Flash. Existem diversos tipos de ferramentas, tais como ferramentas de desenho, pintura, transformação de objeto, esquemas de cores, etc.

Figura 22 – Apresentação da Barra de Ferramentas.

Fonte: Do Autor.

5.3.6 Palco

Segundo Adobe (2011), o Palco é a janela mais importante do Adobe Flash, pois somente o seu conteúdo será exibido para o usuário final. Ele representa a implementação gráfica de um painel de conteúdos. Tudo que está contido dentro dele pode ser utilizado como “Objeto Instanciado”, via programação, desde que o mesmo possua a propriedade “Nome de Instância” preenchida.

Também é possível instanciar dinamicamente os objetos contidos na Biblioteca e exibir os mesmos no Palco, manipulando suas propriedades também de

forma dinâmica (ADOBE, 2011). A Figura 23 mostra um exemplo de um mapa sendo exibido no palco do Adobe Flash.

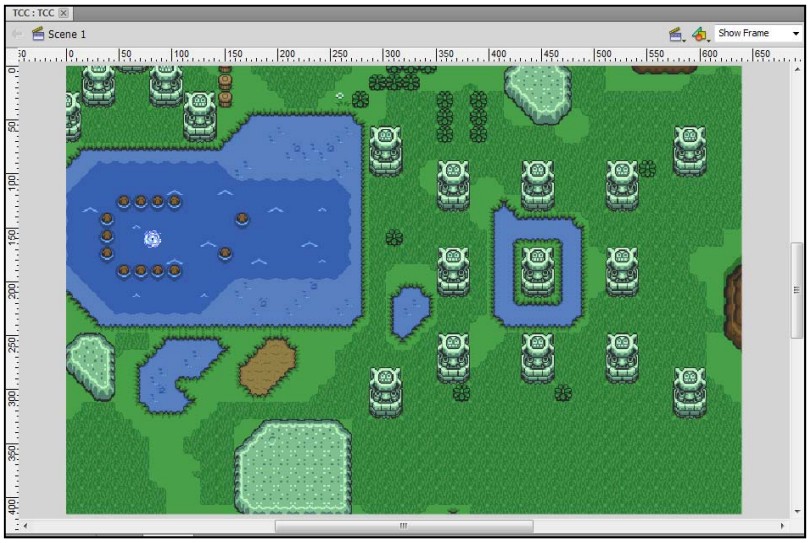

Figura 23 – Apresentação do Palco.

Fonte: Do Autor.

6 METODOLOGIA

As atividades deste trabalho de pesquisa podem ser divididas em duas etapas: fundamentação teórica e desenvolvimento de software. Na parte de fundamentação teórica (capítulos 3, 4 e 5), são abordadas, teorias, algoritmos e ferramentas computacionais necessárias ao desenvolvimento do jogo proposto. A metodologia adotada nesta etapa consiste, basicamente, no estudo dos conceitos, teorias, algoritmos e métodos computacionais relacionados ao tema desta pesquisa: desenvolvimento de jogos utilizando algoritmos genéticos juntamente com a ferramenta Adobe Flash.

Para a criação do protótipo do jogo, foram utilizados conteúdos artísticos providos de um clássico dos vídeo games, "*Zelda – A Link to the Past*" pois a forma que é representada o seu ambiente e cenário facilitam o desenvolvimento deste tipo de jogo. Os personagens utilizados vieram de recursos gratuitos de jogos desenvolvidos em uma plataforma chamada *RPG Maker XP* [3].

O protótipo do jogo foi desenvolvido na linguagem de programação nativa do Adobe Flash, que se chama *Actionscript*, porém utilizando um ambiente de programação da própria Adobe, chamado *Flash Builder*. Cabe destacar que toda a movimentação dos personagens foi modelada utilizando a teoria dos Algoritmos Genéticos. A próxima seção tratará do jogo desenvolvido, descrevendo seu roteiro, personagens e utilização do algoritmo genético.

[3] O RPG Maker XP pode ser encontrado para download através do link http://www.rpgmakerweb.com/

7 O JOGO - A JORNADA DO REINO

7.1 DESCRIÇÃO

O protótipo do jogo se trata de um estilo aonde os personagens vão explorando o mundo em busca de seus objetivos. A ambientação se dá em uma época medieval onde a magia e a fantasia convive lado a lado com a humanidade. Na subseção seguinte é feita uma descrição narrativa do roteiro que orientou o processo de implementação do jogo.

7.2 ROTEIRO

O jogo se passa em um reino chamado *Thryllium*, que é regido pelo justo Rei *Kyrms,* cujas decisões são acompanhadas pelo *Lorde Mimnithus*, rei de todas as criaturas mágicas.

A união entre os reinos de *Thryllium* e o das Criaturas Mágicas trouxe a paz e a felicidade a todos. Esta união foi selada com um presente, um cristal mágico chamado *Emithys*, com o poder de manter a paz e a união entre todos os reinos. Caso este cristal caísse em mãos erradas, uma grande guerra aconteceria entre todas as criaturas existentes.

Certo dia, ao se dirigir à sala do trono o rei notou que havia algo estranho. Eis que uma grande bola de fogo surge e atinge a parede da sala. *Lorde Mimnithus* aparece e descobre que *Emithys* havia sumido e, além disso, o Rei *Kyrms* estava magicamente petrificado.

A única coisa que se escutava dentre os corredores do castelo era uma risada maléfica e uma voz amedrontadora que parecia de uma velha bruxa dizendo que a paz estava acabada e o bom tempo das guerras entre os reinos estava de volta.

Lorde Mimnithus ao escutar tão abominável voz tivera certeza de que o problema que os reinos estavam para enfrentar seria muito perigoso para guerreiros comuns, pois o pior pesadelo de todas as criaturas vivas estava de volta.

Uma bruxa, conhecida como *Mihnar,* aprisionada em uma prisão mágica por milhares de anos devido a maldade que espalhou ao mundo, estava livre novamente. Neste momento *Lorde Mimnithus* sabia que deveria tomar uma providência com relação aos acontecimentos. Foi então que ele voltou para seu reino mágico para encontrar seu filho *Emrys*, um jovem mago, metade homem e metade dragão.

Voltando ao reino de *Thryllium*, *Lorde Mimnithus* se reuniu junto com seu filho *Emrys* e com o herdeiro do trono de *Thryllium*, o jovem príncipe *Typhom*, para iniciarem a missão de recuperar *Emithys* e usar sua poderosa mágica para salvar o Rei *Kyrms*.

7.3 PERSONAGENS

O jogo possui algumas variedades de personagens sendo que somente o personagem *Emrys* (filho do Lorde das Criaturas Mágicas*)* pode ser controlado pelo jogador. Por sua vez, o príncipe *Typhom* é controlado pela inteligência artificial (tendo por base a teoria dos algoritmos genéticos), já o restante dos personagens é controlado por movimentos aleatórios. Cabe destacar que os agentes que representam os inimigos, ao se aproximarem de um dos personagens principais (*Emrys* ou *Thyphom*), assumem um comportamento agressivo, atacando-os com o intuito de diminuir as suas vidas.

O problema computacional à ser resolvido é um problema de busca baseado nos perigos que podem ser encontrados pelo caminho. Após os personagens chegarem no objetivo os mesmos deverão retornar ao castelo e devolver o Cristal *Emithys* com segurança ao *Lorde Mimnithus* para que o mesmo utilize *Emithys* para salvar o Rei *Kyrms*.

7.3.1 Aliados

Lorde Mimnithus é o Rei de todas as criaturas mágicas, possui um grande poder mágico, porém não existem somente criaturas boas, existem aquelas que se recusam a obedece-lo e passaram a cultuar a arte das trevas.

A Figura 24 mostra como é a aparência do *Lorde Mimnithus*.

Figura 24 – Aparência do *Lorde Mimnithus*.

Fonte: RPG Maker.

O Rei *Kyrms*, é justo e bondoso, tendo conhecimento de todo o sofrimento que se espalhou durante a guerra entre o mundo mágico e o mundo dos homens, resolveu abrir um tratado de paz com *Lorde Mimnithus*, desde então, décadas se passaram, e uma era de paz foi erguida entre os Reinos.

A Figura 25 mostra como é a aparência do Rei *Kyrms*.

Figura 25 – Aparência do Rei *Kyrms*.
Fonte: RPG Maker.

Typhom é o Príncipe Herdeiro do trono de *Thryllium*, seguindo os passos do seu pai, para que um dia seja capaz de assumir o seu destino e governar o Reino.

A Figura 26 mostra como é a aparência do Príncipe *Typhom*.

Figura 26 – Aparência do Príncipe *Typhom*.
Fonte: RPG Maker.

Emrys, filho de *Lorde Mimnithus* com uma lendária criatura cujo paradeiro é atualmente desconhecido, segundo as lendas da antiga mitologia, a magia foi criada por uma criatura muito poderosa que possuía a forma de dragão de cor azul, no entanto tal criatura jamais foi vista, sabe-se que todos os dragões foram corrompidos pelo falso poder oferecido por *Mihnar*. No entanto este foi um fato ocorrido milhares de anos, quando somente as criaturas mágicas andavam sobre o mundo.

A Figura 27 mostra como é a aparência do Mago *Emrys*.

Figura 27 – Aparência do Mago *Emrys*.
Fonte: RPG Maker.

7.3.2 Inimigos

Esqueletos, estes são criaturas invocadas pela arte das trevas, é necessário um grande conhecimento da magia negra para tal feito.

A Figura 28 mostra como é a aparência dos *Esqueletos*.

Figura 28 – Aparência dos *Esqueletos*.
Fonte: RPG Maker.

Os *Dragões* antigamente eram criaturas com grandes poderes, até que foram corrompidas pelo lado obscuro da magia, vários deles foram destruídos, e os que continuaram vagando por este mundo são dominados por *Mihnar* e perderam todos os seus poderes.

Figura 29 mostra como é a aparência dos *Dragões*.

Figura 29 – Aparência dos *Dragões*.
Fonte: RPG Maker.

Mihnar é uma criatura mágica muito poderosa, porém aprendeu a dominar as artes da magia negra e teve sua alma corrompida pela maldade. *Mihnar* foi a

causadora da grande guerra que quase extinguiu todas as criaturas boas do mundo, foi neste período que os lendários dragões foram corrompidos por seu poder obscuro.

Figura 30 mostra como é a aparência de *Mihnar*.

Figura 30 – Aparência de *Mihnar*.

Fonte: RPG Maker.

As Figuras 31, 32 e 33 mostram partes do cenário desenvolvido para o jogo, onde o personagem deverá percorrer até chegar ao ponto objetivo (Figura 33).

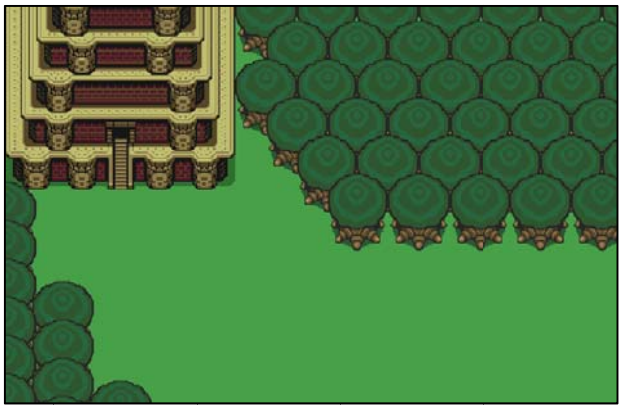

Figura 31 – Parte do mapa onde o personagem começa.

Fonte: Do Autor.

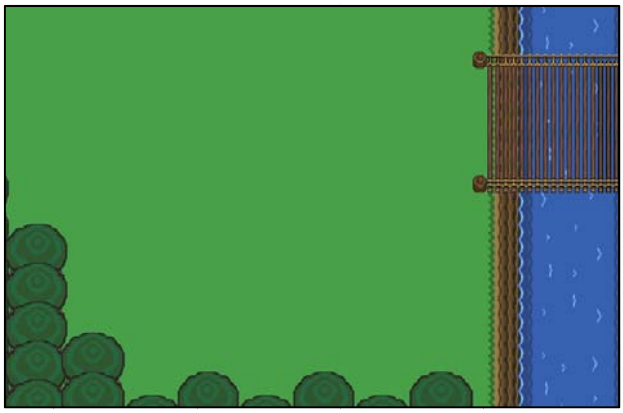

Figura 32 – Parte do cenário do jogo.

Fonte: Do Autor

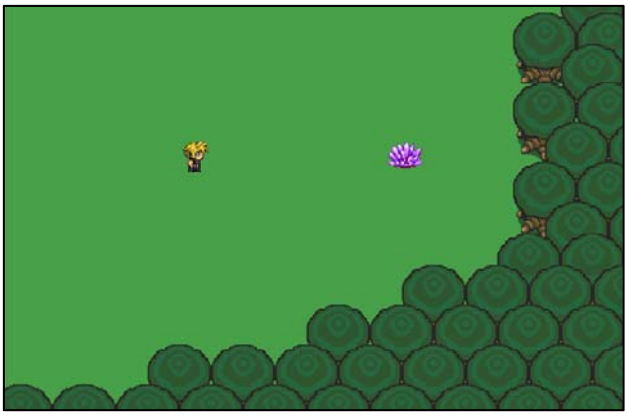

Figura 33 – Personagem e o objetivo do jogo.
Fonte: Do Autor

7.4 A UTILIZAÇÃO DO ALGORITMO GENÉTICO NO CONTROLE DO MOVIMENTO DOS PERSONAGENS

Neste projeto o algoritmo genético foi utilizado para guiar o personagem do ponto inicial do mapa até o ponto objetivo, tentando ao máximo evitar ou desviar dos perigos encontrados pelo caminho.

A população de indivíduos a ser utilizada pelo algoritmo genético se trata de uma cadeia de genes, contendo a possível direção do movimento do personagem e quantos passos o mesmo poderá se movimentar.

Para a elaboração do genoma dos indivíduos foi necessária à criação de um mapa de movimentos que demonstra todas as possíveis posições que o personagem poderá escolher.

A Figura 34 mostra a representação visual deste mapa de movimentos que o AG possui para poder basear suas escolhas.

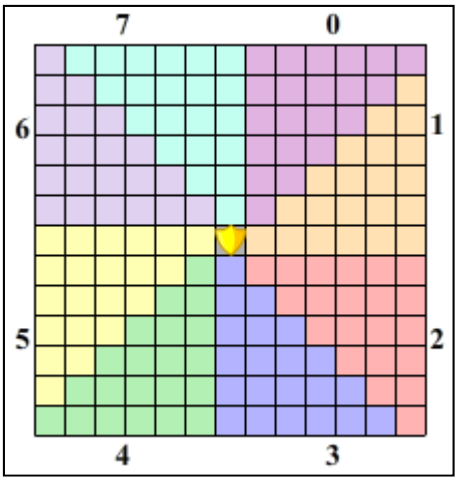

Figura 34 – Representação visual do Mapa de Movimentos.

Fonte: Do Autor.

Assim como mostra a Figura 31, os números que vão de 0 a 7 representam as possíveis direções que o personagem poderá se movimentar. Cada quadrado contido na imagem representa uma casa na qual o personagem poderá andar, sendo que o movimento mínimo é de uma casa e o máximo é de seis casas na direção escolhida.

Após o AG mapear um conjunto de vinte movimentos aleatórios baseados na regra mostrada acima (que corresponde à população inicial), os dados serão convertidos para o sistema binário, transformando-se assim em um vetor contendo a cadeia que define um individuo da população genética, o genoma.

A Figura 35 mostra a representação binária do padrão de movimentos do personagem, onde os três primeiros genes são responsáveis pela direção do movimento e os três últimos são responsáveis pela quantidade de passos que o personagem poderá andar. Por exemplo, para o indivíduo mostrado a direção escolhida é a dois (010, em binário) já a quantidade de passos é cinco (101, em binário).

0	1	0	1	0	1

Figura 35 – Exemplo de Genoma de um Indivíduo da população Genética.

Fonte: Do Autor.

A máscara de cruzamento pode ser definida pelo usuário com o intuito de testar as diferenças entre a utilização de cada uma delas. Dentre os testes feitos

com o AG foram utilizadas máscaras de diversos tipos propostos por Goldberg (1989) se tratando dos padrões mais utilizados para máscaras de cruzamento de comprimento fixo. A Figura 36 mostra o exemplo de máscara de cruzamento de um ponto utilizada.

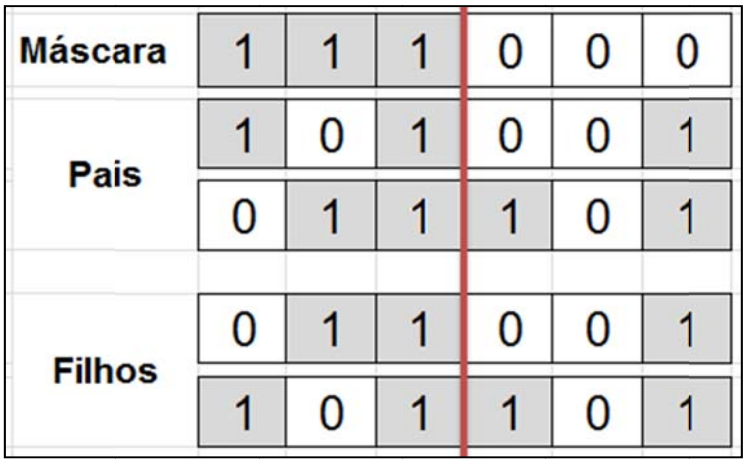

Figura 36 – Exemplo de Máscara de Cruzamento de Um Ponto Utilizada.

Fonte: Do Autor.

A Figura 36 mostra o modo que foi definido o cruzamento de um ponto no AG, sabendo-se que as três primeiras cadeias são baseadas na direção do movimento do personagem e as três últimas baseadas na quantidade de passos que o mesmo pode andar, o cruzamento de um ponto faz a troca somente das informações pertencentes ao mesmo conjunto, ou seja, pega a direção de um pai e os passos do outro. A Figura 37 mostra o exemplo da máscara de cruzamento multiponto utilizada.

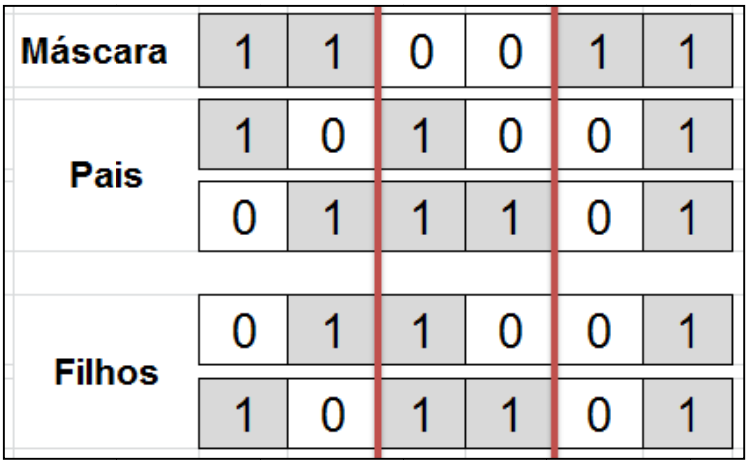

Figura 37 – Exemplo de Máscara de Cruzamento Multiponto Utilizada.

Fonte: Do Autor.

A Figura 37 mostra o modo que foi utilizado o cruzamento multiponto no AG. O multiponto diferentemente do cruzamento de um ponto, muitas vezes mistura a informação dos conjuntos como é o caso do AG do personagem, onde parte do conjunto referente à direção do movimento é cruzada com o outro indivíduo e o mesmo ocorre com a quantidade de passos.

Esta técnica acaba que forçando a população a se tornar mais diversificada, pois sempre estará mesclando as informações dos mesmos conjuntos.

A Figura 38 mostra o exemplo da máscara de cruzamento uniforme.

Máscara	1	0	1	1	0	1
Pais	1	0	1	0	0	1
	0	1	1	1	0	1
Filhos	0	0	1	1	0	1
	1	1	1	0	0	1

Figura 38 – Exemplo de Máscara de Cruzamento Uniforme Utilizada.

Fonte: Do Autor.

A Figura 38 mostra o modo que foi implementado o cruzamento uniforme no AG. O cruzamento uniforme tem sua ação parecida com o cruzamento multiponto seguindo mesmo conceito de mesclar os conjuntos, porém de uma maneira aleatória. Apesar de a ação da máscara de cruzamento uniforme ser parecida com a multiponto, muitas vezes o resultado varia muito por se tratar de vários pontos aleatórios de cruzamento podendo ser melhor ou pior dependendo do caso.

A Figura 39 mostra o trecho de código da implementação dos cruzamentos, utilizando a máscara de cruzamentos.

```
/*=============================================================
      Remove os Piores Indivíduos da População
=============================================================*/
//Caso o Índice do Indivíduo 0 seja Maior que o do Indivíduo 1
if ( piores[ 0 ] > piores[ 1 ] ) {
      //Caso o Indivíduo 0 Seja o Último da População
      if ( piores[ 0 ] == popLen - 1 ) {
            //Removo o Último Indivíduo da População
            populacao.pop(  );
      } else {
            //Remove o Indivíduo 0 da População
            populacao.splice( piores[ 0 ], 1 );
      }
      //Remove o Indivíduo 1 da População
      populacao.splice( piores[ 1 ], 1 );
//Caso Contrário
} else {
      //Caso o Indivíduo 1 Seja o Último da População
      if ( piores[ 1 ] == popLen - 1 ) {
            //Removo o Último Indivíduo da População
            populacao.pop(  );
      } else {
            //Remove o Indivíduo 1 da População
            populacao.splice( piores[ 1 ], 1 );
      }
      //Remove o Indivíduo 0 da População
      populacao.splice( piores[ 0 ], 1 );
}

/*=============================================================
      Aplica a Máscara de Cruzamento
=============================================================*/
//Passa por Todas as Posições do Array da Máscara de Cruzamento
for ( var i:int = 0; i < maskLen; i++ ) {
      //Caso o Valor do Índice i da Máscara for 1
      if ( mascaraCruzamento[ i ] == 1 ) {
            //Adiciona o Valor do Índice i do Pai 2 no Índice i do Filho 1
            filho1.push( pai2[ i ] );
            //Adiciona o Valor do Índice i do Pai 1 no Índice i do Filho 2
            filho2.push( pai1[ i ] );
      //Caso Contrário
      } else {
            //Mantém o Valor do Índice i do Pai 1 no Índice i do Filho 1
            filho1.push( pai1[ i ] );
            //Mantém o Valor do Índice i do Pai 2 no Índice i do Filho 2
            filho2.push( pai2[ i ] );
      }
}

//Adiciona os Filhos 1 e 2 na População
this.AdicionaFilhos( filho1, filho2 );

//Caso o Fator de Mutação esteja Habilitado
if ( this.MutaçãoAtiva(  ) ) {
      //Analisa e Aplica a Mutação nos 2 Últimos Indivíduos da População
      this.AplicaMutacao( popLen - 2, popLen - 1 );
}
```

Figura 39 – Trecho de Código Referente a Implementação do Cruzamento.

Fonte: Do Autor.

A Figura 36 mostra o trecho de código implementado seguindo a teoria proposta por Goldberg (1989) onde todo valor 1 da máscara sofre cruzamento enquanto o valor 0 mantém o valor contido no mesmo índice dos pais.

Além de máscaras de cruzamento, também foi utilizada a técnica de mutação nos indivíduos com o intuito de analisar as diferenças entre o desempenho do AG com e sem mutação.

A Figura 40 mostra o trecho de código da implementação do operador de mutação.

```
/*=============================================================
      Analisa a Necessidade de Mutação dos Filhos
=============================================================*/
//Caso Filho 1 for Abaixo da Média dos Melhores
if ( populacao[ filho1 ].probSelecao < this.MediaMelhores(  ) ) {
      //Retorna a Probabilidade de Mutação dele ( 0 ~100% )
      probMutacao = this.ProbMutacao(  );
      //Cria um Array contendo a Probabilidade e o Filho
      mutacoes.push( {
            probabilidade: probMutacao,
            filho: populacao[ filho1 ]
      });
}

//Caso Filho 2 for Abaixo da Média dos Melhores
if ( populacao[ filho2 ].probSelecao < this.MediaMelhores(  ) ) {
      //Retorna a Probabilidade de Mutação dele ( 0 ~100% )
      probMutacao = this.ProbMutacao(  );
      //Cria um Array contendo a Probabilidade e o Filho
      mutacoes.push( {
            probabilidade: probMutacao,
            filho: populacao[ filho2 ]
      });
}

/*=============================================================
      Aplica a Mutação Caso Necessário
=============================================================*/
//Caso Exista Alguma Mutação para ser Feita
if ( mutacoes.length > 0 ) {
      //Passa por Todas as Mutações Armazenadas
      for each ( var mutacao:Object in mutacoes ) {
            //Se a Probabilidade Estiver Dentro do Índice Necessário
            if ( this.PrecisaMutacao( mutacao.probabilidade ) ) {
                  //Escolhe Qual Gene será Alterado de forma Aleatória
                  gene = this.EscolheGene(  );
                  //Caso o Valor do Gene seja 1
                  if ( mutacao.filho.genoma[ gene ] == 1 ) {
                        //Altera para 0
                        mutacao.filho.genoma[ gene ] = 0
                  //Caso Contrário
                  } else {
                        //Altera para 1
                        mutacao.filho.genoma[ gene ] = 1
                  }
            }
      }
}
```

Figura 40 – Trecho de Código Referente a Implementação da Mutação.

Fonte: Do Autor.

A Figura 40 mostra o trecho de código implementado seguindo o exemplo proposto por Medina & Müller (2009), onde após o cruzamento, é verificado se o

filho possui uma probabilidade de seleção abaixo daquelas dos melhores indivíduos, caso mesmo possua baixa probabilidade de seleção, o fator de mutação entrará em ação de acordo com a probabilidade de mutação definida pelo usuário. Após isso, o individuo sofre a mutação, permanecendo na população e tendo a chance de ser cruzado com algum outro indivíduo.

Todo AG necessita de uma heurística para poder aplicar a Função de Fitness aos indivíduos da população genética. Neste projeto a heurística utilizada foi um cálculo desenvolvido a partir da possível nova posição do agente inteligente. A Figura 41 mostra um exemplo dos valores envolvidos nesta heurística.

Direção 7 – 3 Passos	
Distância Até o Objetivo	36
Quantidade de Inimigos	2
Distância Inimigo 1	10
Distância Inimigo 2	5
Grau de Perigo	30
Novo Peso Inimigo	10
Qtd. Inimigos No Raio	5
Perigo Após o Movimento	50
Heurística	**1130**

Figura 41 – Exemplo do Cálculo da Heurística de Movimento.

Fonte: Do Autor.

A Figura 38 mostra como foi montado os valores e pesos de cada movimento baseado na distância e quantidade de inimigos na direção indicada para cada indivíduo da população genética. Seguindo o exemplo da figura, onde o existe a possibilidade do agente se movimentar *3 passos* na *direção 7,* existem as seguintes variáveis e considerações que são utilizadas para mapear a heurística.

- Distância Até o Objetivo: É a distância na qual o agente ficará do ponto objetivo após o movimento;
- Quantidade de Inimigos: É a quantidade de inimigos que existem na direção em que o agente está pretendendo se movimentar baseada no mapa de movimentos (Figura 34);
- Distância dos Inimigos: É a distância de cada inimigo até o agente;

- Grau de Perigo: Este é o peso que o grupo de inimigos vai ter, este valor é gerado a partir da multiplicação da somatória de todas as distâncias dos inimigos multiplicadas pela quantidade de inimigos;
- Novo Peso Inimigo: Este é o novo peso que os inimigos terão após o movimento ser realizado. Na aplicação foi utilizado o valor cinco;
- Quantidade de Inimigos no Raio: É a quantidade de inimigos presentes em todo o raio de visão do agente, considerando todas as oito possíveis direções após o movimento. Este é utilizado para saber qual o perigo após o movimento;
- Perigo Após Movimento: Este é a multiplicação da quantidade de inimigos no raio pelo novo peso inimigo;
- Heurística: Este valor é o que define se tal movimento é bom ou ruim. Na aplicação foi definido que o quanto maior este valor, pior o movimento. A heurística é obtida a partir da multiplicação da distância do objetivo pelo grau de perigo e ao fim da multiplicação é somado o valor do perigo após o movimento.

Com base nas regras anteriores é possível filtrar cada possível movimento efetuando o cruzamento com os possíveis melhores indivíduos da população genética com o intuito de conseguir gerar a melhor solução possível e assim então efetuar o movimento. Após o movimento efetuado, todos os processos citados acima serão executados novamente, gerando o novo movimento, e assim sucessivamente até o agente atingir seu objetivo.

7.5 RESULTADOS OBTIDOS

Foram realizados diversos testes cada um com uma determinada configuração distinta alternando a máscara utilizada e o fator de mutação ativo ou inativo. Durante todo o processo de testes, alterações notórias nos desempenho foram relevantes para a obtenção dos resultados.

A métrica utilizada para mapear o desempenho do AG, foi a quantidade de casas que o agente andou até atingir seu objetivo, baseado neste valor foi executado o AG dez vezes, capturando a quantidade de casas totais de cada execução e calculando a média de passos para cada bloco de execução. Foram testadas as três máscaras de cruzamento citadas anteriormente (111000, 110011,

101101), onde cada uma foi testada com e sem o fator de mutação com o intuito de mapear qual seria a melhor opção à ser utilizada para este tipo de problema, já que para cada problema pode existir uma combinação de fatores distintos para que seja possível encontrar os melhores resultados. A Tabela 3 mostra os resultados obtidos com a aplicação do AG sem utilizar a mutação.

Tabela 3 – Resultados Obtidos dos Cruzamentos Sem Mutação.

Máscara	**Passos**										**Média**
111000	38	40	49	42	38	38	47	39	66	50	44,7
110011	38	41	37	46	42	46	39	52	38	43	42,2
101101	40	44	40	37	47	39	42	38	43	46	41,6

Fonte: Do Autor.

A Tabela 3 mostra as diferenças entre a utilização de cada tipo de máscara. As médias de movimentos resultantes da utilização de todas as máscaras tiveram valores bem similares, porém analisando as individualmente cada uma das máscaras é possível ver que a máscara que obteve melhor desempenho (101101) foi a máscara de cruzamento uniforme (Figura 38) contendo em suas execuções uma quantidade menor de movimentos ruins que o agente computacional realizou.

O mesmo teste foi repetido utilizando o fator de mutação ativo no AG. A Tabela 4 mostra os resultados com a aplicação do AG com a utilização do fator de mutação.

Tabela 4 – Resultados Obtidos dos Cruzamentos Com Mutação.

Máscara	**Passos**										**Média**
111000	38	37	50	37	41	39	66	36	40	37	42,1
110011	42	38	38	39	41	40	40	45	52	45	42,0
101101	46	45	41	37	39	43	38	37	37	41	40,4

Fonte: Do Autor.

Na Tabela 4 mostra a diferença entre a utilização de cada tipo de máscara, porém ativando o fator de mutação durante os cruzamentos. As médias de movimentos resultantes da utilização de todas as máscaras continuaram em torno de quarenta, conforme ocorrido na aplicação sem a máscara.

No entanto, a diferença entre a utilização de cada máscara com e sem mutação acaba sendo facilmente notada. Onde na aplicação sem mutação os valores tiveram uma variação de 41,6 casas mínimas para 44,7 máximas, aplicando

o fator de mutação este mesmo intervalo caiu para 40,4 casas mínimas até 42,1 casas máximas.

Com relação às diferenças de cada máscara, também é possível notar que a máscara que apresentou melhor desempenho sendo utilizada com ou sem mutações, foi a máscara de cruzamento uniforme (Figura 38). Isto devido a sua maior capacidade de gerar indivíduos diferentes, evitando assim a estagnação das potencialidades dos indivíduos dentro da população.

8 TRABALHOS FUTUROS

Este é um protótipo de um jogo desenvolvido utilizando algoritmos genéticos, portanto sua implementação não está completa.

Como trabalhos futuros pretende implementar a inteligência artificial em todos os NPCs envolvidos no jogo, desenvolver os outros quatro personagens principais tornando-os capazes de serem controlados por algoritmos genéticos ou por um jogador.

A ampliação da história para abranger mais conteúdos, uma maior jogabilidade e experiência do usuário final, também serão focos de novos estudos.

Também será feita uma migração de 2D para 3D tornando assim o jogo mais interativo e imersivo para os jogadores.

8.1 DESENVOLVIMENTO DAS CLASSES DE PERSONAGENS

Existem outras classes de personagens que serão desenvolvidas neste jogo, mas que, por limitações de tempo, não foram possíveis para este projeto. Assim como também existem partes da implementação dos personagens *Emrys* e *Typhom* a serem finalizadas e melhoradas. As subseções seguintes mostram os personagens que serão futuramente implementados.

8.1.1 O Caçador

O Caçador possui uma grande facilidade de encontrar melhores caminhos para as jornadas, assim como possui um grande conhecimento sobre diversos tipos de criaturas e suas devidas habilidades. A Tabela 5 mostra quais as principais características do Caçador.

Tabela 5 – Definições do Caçador.

Especificação	**Descrição**
Especialidade	Encontrar caminhos e Atacar.
Classe de Armas	Lanças e Arcos.
Classe de Equipamentos	Malha de Metal contendo Cabeça, Torço, Braços, Mãos, Ombros e Pernas e Pés.
Item Especial	Mapas e um Bestiário.
Habilidades em Combate	Golpes com a Lança, Flechadas, Socos e Pontapés.
Habilidades fora de Combate	Interpretação Cartográfica, Reconhecimento de Rastros.

Fonte: Do Autor.

Assim como é mostrado na Tabela 5, o Caçador tem como principal função mapear rotas seguras baseado em seu conhecimento prévio ou em informações como rastros contidos pelo caminho, com o intuito de ajudar o grupo a fazer um caminho seguro.

8.1.2 O Feiticeiro

O Feiticeiro possui habilidades de controle da magia e dos elementos, assim como a habilidade de restaurar o fluxo da vida nas criaturas. A Tabela 6 mostra quais as principais características do Feiticeiro.

Tabela 6 – Definições do Feiticeiro.

Especificação	**Descrição**
Especialidade	Curar e Atacar
Classe de Armas	Adagas e Cajados
Classe de Equipamentos	Vestes de Pano contendo Cabeça, Torço, Braços, Mãos, Ombros e Pernas e Pés.
Item Especial	Poções
Habilidades em Combate	Curar os outros personagens, curar a si mesmo utilizando as poções e causar danos aos inimigos.
Habilidades fora de Combate	Sentir a magia contida nos objetos, criar poções de cura e restauração de energia.

Fonte: Do Autor.

Assim como mostra a Tabela 6, o Feiticeiro tem como principal função curar os aliados e quando possível batalhar com os inimigos evitando que os mesmos vençam, além de ser um ótimo alquimista criando diversas poções.

8.1.3 O Arqueiro

O Arqueiro possui grande destreza e a habilidade de atacar os inimigos à grande distância evitando assim, que o grupo sofra ataques diretos de seus inimigos. A Tabela 7 mostra quais as principais características do Arqueiro.

Tabela 7 – Definições do Arqueiro.

Especificação	Descrição
Especialidade	Atacar
Classe de Armas	Adagas, Arcos e Flechas.
Classe de Equipamentos	Vestes de Couro contendo Cabeça, Torço, Braços, Mãos, Ombros e Pernas e Pés.
Item Especial	Kit para a criação de Flechas.
Habilidades em Combate	Atirar Flechas.
Habilidades fora de Combate	Uma visão aguçada e poder identificar quando os inimigos poderão aparecer.

Fonte: Do Autor.

Assim como mostra a Tabela 7, o Arqueiro tem como seu principal objetivo alertar o grupo da chegada de inimigos e evitar ataques diretos ao grupo.

8.1.4 O Cavaleiro

O Cavaleiro possui a habilidade dos combates corpo a corpo, evitando ataques diretos ao grupo utilizando seu escudo e sua espada. A Tabela 8 mostra quais as principais características do Cavaleiro.

Tabela 8 – Definições do Cavaleiro.

Especificação	Descrição
Especialidade	Atacar e Bloquear Ataques.
Classe de Armas	Espadas Curtas, Espadas Longas e Escudos.
Classe de Equipamentos	Vestes de Metal Pesado contendo Cabeça, Torço, Braços, Mãos, Ombros e Pernas e Pés.
Item Especial	Kit de reparo de armas.
Habilidades em Combate	Golpes com a espada e bloqueios com o escudo.
Habilidades fora de Combate	N/A

Fonte: Do Autor

Na Tabela 8 podemos ver que o principal objetivo do Cavaleiro, é enfrentar os inimigos diretamente evitando que os mesmos cheguem perto de mais do grupo.

8.2 AMPLIAÇÃO DO CONTEÚDO

Em um jogo para se tornar interativo e agradável, o seu conteúdo deve ser muito rico em história e variedades de opções para o usuário.

Com base nisso, seguindo a proposta original do projeto, será desenvolvido um complemento de conteúdos abrangendo diversos outros mapas, reinos, itens, habilidades e missões para que possa tornar a historia do jogo cada vez melhor.

8.3 MIGRAÇÃO DE 2D PARA 3D

Como a ideia inicial do projeto era desenvolver um jogo utilizando conceitos 3D, como trabalho futuro também, será feita a implementação de todo o conteúdo desenvolvido para a plataforma 3D, para isto será utilizado a ferramenta Unity3D.

A Unity3D foi escolhida justamente por apresentar diversas das qualidades que o Adobe Flash também apresenta como a facilidade de desenvolver aplicações interativas e a própria questão da portabilidade do conteúdo desenvolvido.

9 CONSIDERAÇÕES FINAIS

Com base nos resultados obtidos, é possível afirmar que a utilização de algoritmos genéticos no desenvolvimento de jogos pode impulsionar o grau de interatividade destes tipos de aplicação em um nível ainda não alcançado, já que esta técnica da Inteligência Artificial pode ser aplicada a maioria dos problemas e atividades que um NPC realiza dentro de um jogo, por exemplo, é possível desenvolver heurísticas não somente para controlar movimentos, mas também para determinar as ações que um personagem pode tomar dentro do ambiente do jogo, assim tornando as decisões e reações dos NPCs cada vez mais dinâmicas.

No entanto, a construção de um AG não é algo simples, pois para cada tipo de problema é necessário um tipo de heurística, e se aplicarmos as técnicas de AG em NPCs, seria necessário cada NPC ter uma heurística diferente do restante, pois a criação de heurísticas distintas para cada agente presente no jogo, serviria para definir algo semelhante à personalidade deste agente, causando assim um grau de interatividade muito maior, porém elevando muito o tempo de desenvolvimento.

No geral, o projeto atingiu todos os objetivos propostos, conseguindo principalmente mostrar a utilização de um AG de maneira diferente e possibilitando mapear o nível de seu desempenho.

Conclui-se, portanto, que os AGs possuem uma enorme capacidade para serem utilizados dentro de jogos com o intuito de simular e resolver os mais diferentes tipos de problemas.

BIBLIOGRAFIA

ABRAGAMES, **Uma pesquisa indicando o crescimento do número de pessoas que fazem uso de jogos eletrônicos,** Abril de 2009, Disponível em <http://www.abragames.org/newsletter/arquivo/09/04/>. Acesso em 27 fev. 2011.

ADOBE, **Manual de Utilização do Adobe Flash**, 2011, Disponível em < http://help.adobe.com/en_US/flash/cs/using/flash_cs5_help.pdf>. Acesso em 13 set. 2011.

ARMORGAMES, **Site com um grande acervo de jogos em flash**, 2005, Disponível em <http://www.armorgames.com>. Acesso em 10 ago. 2011.

BARBOSA, H., J., C. **Algoritmos genéticos para otimização em engenharia: uma introdução,** IV Seminário sobre elementos finitos e métodos numéricos em engenharia, Juiz de Fora, MG, 1996.

BATTAIOLA, A., L. **Jogos por Computador – Histórico, Relevância Tecnológica e Mercadológica, Tendências de Implementação,** XIX Jornada de Atualização em Informática, Curitiba, 2000.

BURKE, E.; *et al*. **A hybrid genetic algorithm for highly constrained timetabling problems, 6th International conference on genetic algorithms,** Pitsburgh, USA, 1995.

CLUA, E., W., G.; BITTENCOURT, J., R. **Desenvolvimento de Jogos 3D: Concepção, Design e Programação,** Rio de Janeiro, 2005.

CORDENONSI, A., Z.; LOY, A., M. **Análise das Interações entre Agentes Reativos em um Ambiente de Simulação Tridimensional,** Congresso Brasileiro de Computação, Itajaí, 2003.

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

DARWIN, C. **On the origin of species,** Outubro de 1859 Disponível em <www2.hn.psu.edu/faculty/jmanis/darwin.htm> Acesso em 26 mar. 2011.

DAVIS, L., D. **Handbook of genetic algorithms,** Van Nostrand Reinhold, 1991.

DAWKINS, R. **A escalada do Monte Improvável: Uma Defesa da Teoria da Evolução,** Campanha das Letras, São Paulo, 1996.

FUJITA, E. **Algoritmos de IA para jogos,** Londrina, 2005.

GEYER-SCHULTZ, **A. Fuzzy rule-based expert systems and genetic machine learning,** Heidelberg, Physica-Verlag, 1997.

GOLDBERG, D., E. **Genetic Algorithms in Search Optimization and Machine Learning,** United States, Addison-Wesley, 1989.

HOLLAND, J. **Adaptation in Natural and Artificial Systems,** Ann Arbor, University of Michigan Press, 1975.

KASHIMOTO, **A. Inteligência Artificial em Jogos Eletrônicos,** 2004.

LUCAS, D., C. **Algoritmos Genéticos: Uma Introdução**, Março, 2002.

LUGER, G., F. **Inteligência Artificial,** Bookman, 4 ed, 2004.

MEDINA, J.; MÜLLER, R. **A utilização de algoritmos genéticos no desenvolvimento de jogos.** 2009.

MICHALEWICZ, Z. **Genetic algorithms + data structures = evolution programs,** Berlin, Springer-Verlag, 1999.

PASSOS, B., E.; et al. **Tutorial: Desenvolvimento de Jogos com Unity 3D,** VIII Brazilian Symposium on Games and Digital Entertainment, Rio de Janeiro, 2009.

POZZER, C., T.; FURTADO, A., L. **Agentes e Emoções em Histórias Interativas,** Rio de Janeiro, 2003.

RUSSEL, S.; NORVIG., P. **Inteligência Artificial,** Campus, 2 ed, 2004.

SCHWAB, B. **AI Game Engine Programming.** Hingham, Charles River Media, 2004.

SILVEIRA, S., R.; BARONE, D., A., C. **Jogos Educativos Computadorizados Utilizando a Abordagem de Algoritmos Genéticos,** Porto Alegre, 1998.

SOUZA, G., C. **Estudo de um Simulador de Sistemas Multiagentes para Entretenimento Digital,** Novo Hamburgo, 2010.

UOL JOGOS, **Texto sobre toda a linha de tempo da história dos jogos eletrônicos,** Junho de 2004, Disponível em <http://jogos.uol.com.br/reportagens/historia/1961.jhtm>. Acesso em: 16 abr. 2011.

WALL, M., G. **A C++ library of genetic algorithm components. Massachussets,** EUA, 2000, Disponível em <http://lancet.mit.edu/ga/> Acesso em 4 jun. 2011.

WHITLEY, D. **A genetic Algorithm Tutorial,** 2000.

APENDICE A

A Utilização de Algoritmos Genéticos no Controle do Movimento de Personagens

Virgilio M. Neto[1], Patrick P. Silva[1], Anderson F. Talon[1], Larissa P. da Luz[1]

[1]Centro de Ciências Exatas e Sociais Aplicadas– Universidade Sagrado Coração (USC)
CEP 17.011-160 – Bauru – SP – Brasil

virgilio.missao.neto@gmail.com, patrickpsilva@gmail.com, anderson.talon@gmail.com, larissapavarinidaluz@yahoo.com.br

Abstract. *Entertainment is one of the areas of Information Technology that has shown a great growth, especially in Brazil. Because that, users increasingly seek greater challenges, more interactive games where they can have more fun. The generation of computational agents with intelligent behavior offers greater interactivity is one of the challenges more targeted and more difficult to apply. So considering this fact, the objective of this work is to develop a game prototype using Genetic Algorithms in Adobe Flash platform.*

Resumo. *O entretenimento é uma das áreas da Tecnologia da Informação que vem apresentando um grande crescimento, principalmente no Brasil. Devido a isso, os usuários cada vez mais buscam maiores desafios, jogos mais interativos onde possam se divertir mais. A geração de agentes computacionais com comportamento inteligente, proporcionando uma maior interatividade é um dos desafios mais almejados e mais difíceis de serem aplicados. Assim, considerando este fato o objetivo do presente trabalho é desenvolver um protótipo de um jogo utilizando Algoritmos Genéticos na plataforma Adobe Flash.*

1. Introdução

Com o grande crescimento da área de entretenimento dentro da Tecnologia da Informação, cada vez mais os usuários deste tipo de aplicação buscam maior interatividade e diversão. Este vem sendo um dos mais buscados objetivos das empresas desenvolvedoras de jogos, porém também o mais difícil.

Segundo uma publicação da Abragames (2009), uma pesquisa recente feita pelo Ibope[4] relata que cerca de sete milhões de brasileiros usam o seu tempo livre na internet para jogar. Esta mesma pesquisa revela que pessoas de 18 a 35 anos, jogam cerca de 18 horas por semana. Desta forma, o desenvolvimento de jogos para computadores ganha cada vez mais importância no aspecto econômico. Este fato desperta o interesse não só das produtoras de jogos como também das empresas de publicidade que veem neste tipo de aplicação um novo ramo de negócio: o desenvolvimento de propaganda dentro de jogos.

Parte desta interatividade que os jogadores procuram se encontram nos agentes inteligentes contidos no jogo, chamados de Personagens Não Jogáveis

[4] Instituto Brasileiro de Opinião Pública e Estatística.

(*Non Player Characters,* NPC). Segundo Cordenonsi & Loy (2003), agentes inteligentes são importantes ferramentas para a resolução de problemas em ambientes dinâmicos e/ou complexos.

Dentre as várias áreas da Inteligência Artificial (IA), os Algoritmos Genéticos (AG), baseados na teoria da evolução das espécies proposta por Darwin (1859) poderiam ser uma solução para os problemas de interatividades almejados nos games. Segundo Russel & Norvig (2004), os AGs são implementados através de várias possíveis soluções de um problema chamadas de indivíduos, formando uma população, onde ao cruzar informações de seus indivíduos, possa se obter os melhores resultados.

Para o desenvolvimento da aplicação, foi utilizada a ferramenta Adobe Flash, devida a sua facilidade para a criação de aplicações interativas de maneira intuitiva e possibilitando o funcionamento da aplicação final com um alto nível de portabilidade.

2. A Inteligência Artificial na Evolução dos Jogos

Segundo Fujita (2005), vários estudiosos de IA, tais como Luger (2004), Russel & Norvig (2004), entre outros, consideram o filosofo grego Aristóteles (384-322 A.C.) como um dos principais precursores dos estudos de IA. Por isso, segundo Schwab (2004), pode-se dizer que a evolução da IA proporcionou muito a evolução dos jogos.

Depois da criação do primeiro jogo, o "*Spacwar!",* vários outros jogos começaram a ser produzidos e lançados pelo mundo todo. Isto possibilitou o desenvolvimento e crescimento de uma área hoje conhecida como IA. Conforme os anos foram passando, novos jogos foram desenvolvidos e novas maneiras de tornar o jogo interativo foi sendo aplicadas, maiores informações sobre a IA nos jogos se encontram no encontram na seção 3.1 da monografia.

3. A Relação entre Darwin e os Algoritmos Genéticos

Os AGs funcionam de uma forma que segue o conceito da seleção natural proposto por Charles Darwin e Alfred Wallace em 1858, que resumidamente propõe que espécies sempre tendem a procurar os melhores parceiros para a reprodução, com isso a prole resultante sempre tende a nascer com as melhores características dos pais e assim garantindo ou prolongado a sobrevivência da espécie (DARWIN, 1858).

Segundo Russel & Norvig (2004) um AG, é um conjunto de resultados que são analisados e com base em seus conteúdos, novos resultados são melhorados podem ser gerados. Este processo se repete inúmeras vezes até que os resultados obtidos formem uma população melhor que a população inicial. A Figura 42 mostra o fluxo de ação de um AG.

Figura 42 – Estrutura de um Algoritmo Genético.

Fonte: LUCAS, 2002.

Assim como mostra a Figura 42, a execução de um AG segue a seguinte ordem. Primeiro a população genética é iniciada e logo após os seus indivíduos são avaliados e classificados. Em seguida melhores indivíduos possuem uma maior chance de serem selecionados como reprodutores gerando novos indivíduos da população, caso desejado, os filhos poderão sofrer mutações em suas cadeias genéticas forçando assim a geração de indivíduos diferentes dos presentes na população, após isto os indivíduos resultantes do cruzamento serão avaliados e inseridos na nova população. Este processo se repete até que chegue ao fim da execução. Mais detalhes sobre cada passo da execução do AG estão presentes na seção 4.2 da monografia.

4. Metodologia

Para desenvolver a IA utilizada nos padrões de movimentação do agente, foi desenvolvido e implementado uma estrutura de AGs baseando-se nas teorias citadas e descritas por autores referenciados neste artigo.

A plataforma de desenvolvimento foi o Adobe Flash, juntamente com o Flash Builder, possibilitando assim criar um jogo com um grande potencial de portabilidade.

Para o desenvolvimento do jogo foram usados conteúdos artísticos providos de outros jogos ou ferramentas que são disponibilizadas na internet. O design do mapa do jogo foi feito utilizando os elementos de cenário do jogo *"Zelda – A link to the Past"* com algumas adaptações para poder ser aplicado ao projeto.

Os personagens foram retirados de bases de desenhos providos por desenvolvedores de games para a plataforma *RPG Maker XP*[5].

[5] O RPG Maker XP pode ser encontrado para download através do link http://www.rpgmakerweb.com/

5. O Jogo – A Jornada do Reino

O protótipo do jogo se trata de um estilo aonde os personagens vão explorando o mundo em busca de seus objetivos. A ambientação da história se passa em uma época medieval onde a realidade convive junto com a fantasia e a magia, criando assim uma atmosfera interativa com os jogadores. As Figuras 31, 32 e 33 mostram partes do cenário desenvolvido para o jogo.

As Figuras 43, 44 e 45 mostram partes do cenário desenvolvido para o jogo utilizando os elementos visuais do cenário do jogo "*Zelda – A Link to the Past*".

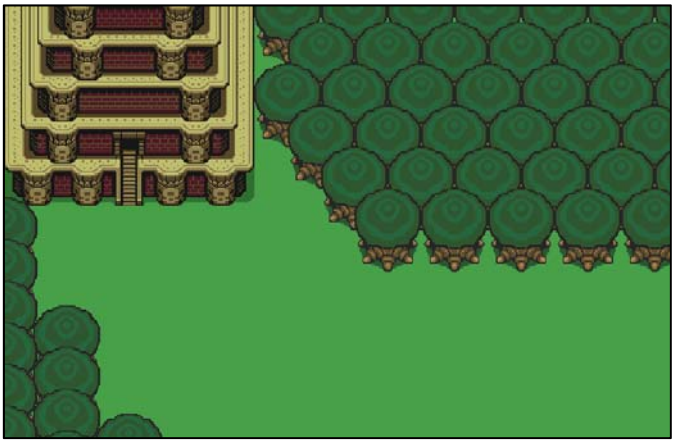

Figura 43 – Ponto inicial do mapa.

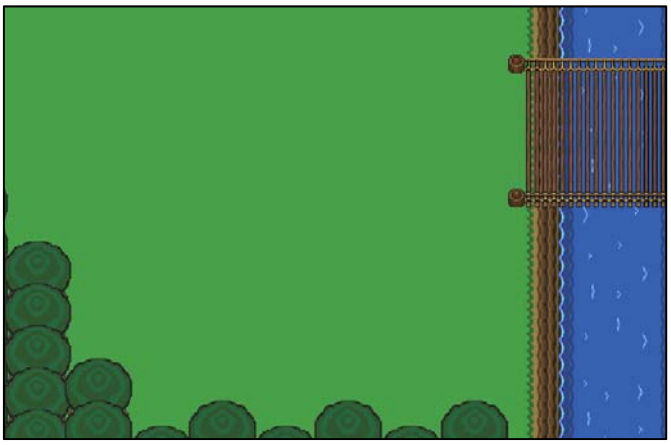

Figura 44 – Rio localizado no centro do cenário.

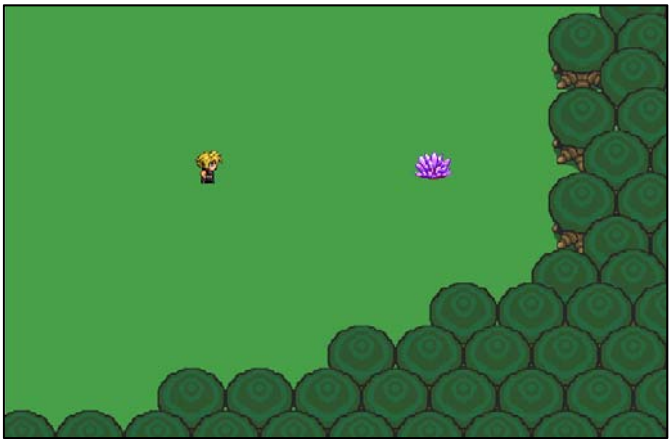

Figura 45 – Local no mapa onde o personagem deve chegar.

5.1. A Utilização do Algoritmo Genético no Controle do Movimento dos Personagens

Fora definidos os padrões de movimento dos personagens baseados em oito possíveis direções, e a quantidade de passos máxima que o agente pode se movimentar são de seis passos para determinada direção. Com base nisto o genoma de cada indivíduo é definido por um vetor de números binários baseados na direção escolhida e a quantidade de passos a se movimentar. Por exemplo, para o indivíduo mostrado a direção escolhida é a dois (010, em binário) já a quantidade de passos é cinco (101, em binário), estes dados forma o genoma 010101.

A Figura 46 mostra os três padrões de máscaras utilizadas no desenvolvimento do projeto, o ponto de cruzamento é definido pela linha que corta a cadeia de elementos. Cada tipo de máscara serviu para mapear um resultado diferente que pode ser visualizado na seção 5.2 deste artigo.

Seguindo o conceito mostrado por Goldberg (1989), em cada posição da máscara que possuir o valor 1, será efetuada a troca de valores dos pais na mesma posição, e em cada valor 0 é mantido o valor atual da respectiva posição dos pais.

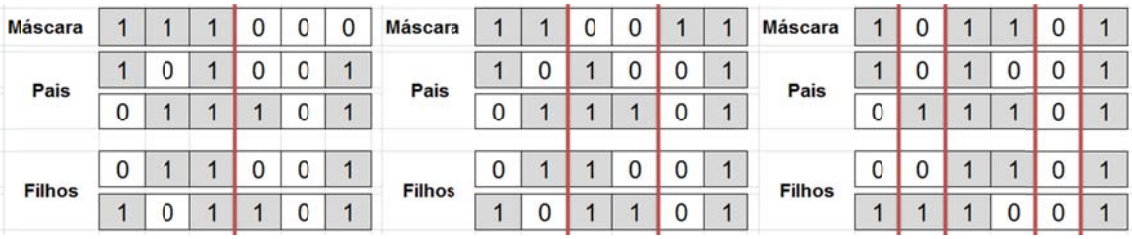

Figura 46 – Operadores de Cruzamento Utilizados.

Também foi implementado no AG o fator de mutação (Medina & Müller, 2009), onde após o cruzamento, é verificado se o filho possui uma probabilidade de seleção abaixo daquelas dos melhores indivíduos, caso mesmo possua baixa probabilidade de seleção, o fator de mutação entrará em ação de acordo com a probabilidade de mutação definida pelo usuário. Após isso, o individuo sofre a mutação, permanecendo na população e tendo a chance de ser cruzado com algum outro indivíduo.

Outro fator importante utilizado pelo AG é uma heurística que define a função de avaliação, baseando a aptidão dos indivíduos baseada no perigo presente nos movimentos que o agente esta prestes a fazer. Este valor varia de acordo com os elementos no mapa, como por exemplo, os inimigos onde a quantidade de inimigos influencia a decisão de qual caminho escolher, tentando sempre escolher o mais seguro. A Figura 47 mostra um exemplo da dos valores envolvidos nesta heurística.

Direção 7 – 3 Passos	
Distância Até o Objetivo	36
Quantidade de Inimigos	2
Distância Inimigo 1	10
Distância Inimigo 2	5
Grau de Perigo	30
Novo Peso Inimigo	10
Qtd. Inimigos No Raio	5
Perigo Após o Movimento	50
Heurística	**1130**

Figura 47 – Cálculo da Heurística do Movimento.

A Figura 47 mostra como foi montado os valores e pesos de cada movimento baseado na distância e quantidade de inimigos na direção indicada para cada indivíduo da população genética. Seguindo o exemplo da figura, onde o existe a possibilidade do agente se movimentar *3 passos* na *direção 7,* existem as seguintes variáveis e considerações que são utilizadas para mapear a heurística.

- Distância Até o Objetivo: É a distância na qual o agente ficará do ponto objetivo após o movimento;
- Quantidade de Inimigos: É a quantidade de inimigos que existem na direção em que o agente está pretendendo se movimentar;
- Distância do Inimigo 1 até o Inimigo n: É a distância de cada inimigo até o agente;
- Grau de Perigo: Este é o peso que o grupo de inimigos vai ter, este valor é gerado a partir da multiplicação da somatória de todas as distâncias dos inimigos multiplicadas pela quantidade de inimigos;
- Novo Peso Inimigo: Este é o novo peso que os inimigos terão após o movimento ser realizado. Na aplicação foi utilizado o valor cinco;
- Quantidade de Inimigos no Raio: É a quantidade de inimigos presentes em todo o raio de visão do agente, considerando todas as oito possíveis direções após o movimento. Este é utilizado para saber qual o perigo após o movimento;
- Perigo Após Movimento: Este é a multiplicação da quantidade de inimigos no raio pelo novo peso inimigo;
- Heurística: Este valor é o que define se tal movimento é bom ou ruim. Na aplicação foi definido que o quanto maior este valor, pior o movimento. A heurística é obtida a partir da multiplicação da distância do objetivo pelo grau de perigo e ao fim da multiplicação é somado o valor do perigo após o movimento.

Com base nas regras anteriores é possível filtrar cada possível movimento efetuando o cruzamento com os possíveis melhores indivíduos da população genética com o intuito de conseguir gerar a melhor solução possível e assim então efetuar o movimento. Após o movimento efetuado, todos os processos citados acima serão executados novamente, gerando o novo movimento e assim sucessivamente até o agente atingir seu objetivo.

5.2. Resultados Obtidos

Foram realizados testes com todos os tipos de máscaras mostrados na figura 5, com base nisso foi possível analisar o desempenho de cada uma delas para ver qual aparentemente é melhor.

A métrica utilizada para gerar o resultado foi a quantidade de casas o agente andou até atingir o resultado, baseado neste valor foi executado o AG dez vezes capturando a quantidade de casas totais de cada execução e calculando a média de passos para cada bloco de execução. Foram testadas as três máscaras de cruzamento citadas anteriormente (111000, 110011, 101101), onde cada uma foi testada com e sem o fator de mutação com o intuito de mapear qual seria a melhor opção à ser utilizada para este tipo de problema, já que para cada problema pode existir uma combinação de fatores distintos para que seja possível encontrar os melhores resultados. A Tabela 9 mostra os resultados obtidos com a aplicação do AG sem utilizar a mutação.

Tabela 9 – Resultados Obtidos

Máscara	Passos										Média
Cruzamentos Sem Mutação											
111000	38	40	49	42	38	38	47	39	66	50	44,7
110011	38	41	37	46	42	46	39	52	38	43	42,2
101101	40	44	40	37	47	39	42	38	43	46	41,6
Cruzamentos Com Mutação											
111000	38	37	50	37	41	39	66	36	40	37	42,1
110011	42	38	38	39	41	40	40	45	52	45	42,0
101101	46	45	41	37	39	43	38	37	37	41	40,4

Na Tabela 9 podemos ver as diferenças entre a utilização de cada tipo de máscara. As médias de movimentos resultantes da utilização de todas as máscaras ficaram em torno de quarenta casas sofrendo uma variação de quatro casas para mais.

No entanto, a diferença entre a utilização de cada máscara com e sem mutação acaba sendo facilmente notada. Onde na aplicação sem mutação os valores tiveram uma variação de 41,6 casas mínimas para 44,7 máximas, aplicando o fator de mutação este mesmo intervalo caiu para 40,4 casas mínimas até 42,1 casas máximas.

Com relação às diferenças de cada máscara, também é possível notar que a máscara que apresentou melhor desempenho sendo utilizada com ou sem mutações, foi à máscara de cruzamento uniforme como é possível ver na máscara 101101 (Figura 5). Isto ocorre devido à sua maior capacidade de gerar indivíduos diferentes, evitando assim a estagnação das potencialidades dos indivíduos dentro da população.

6. Considerações Finais

Com base nos resultados obtidos, é possível afirmar que a utilização de algoritmos genéticos no desenvolvimento de jogos pode impulsionar o grau de interatividade destes tipos de aplicação em um nível ainda não alcançado, já que esta técnica da Inteligência Artificial pode ser aplicada a maioria dos problemas e atividades que um NPC realiza dentro de um jogo.

No geral, o projeto atingiu todos os objetivos propostos, conseguindo principalmente mostrar a utilização de um AG de maneira diferente e possibilitando mapear o nível de seu desempenho.

Conclui-se, portanto, que os AGs possuem uma enorme capacidade para serem utilizados dentro de jogos com o intuito de simular e resolver os mais diferentes tipos de problemas.

Referências Bibliográficas

ABRAGAMES (2009), Uma pesquisa indicando o crescimento do número de pessoas que fazem uso de jogos eletrônicos, Abril de 2009, Disponível em http://www.abragames.org/newsletter/arquivo/09/04/. Acesso em 27 fev. 2011.

Cordenonsi, A., Z.; Loy, A., M. (2003) Análise das Interações entre Agentes Reativos em um Ambiente de Simulação Tridimensional, Congresso Brasileiro de Computação, Itajaí.

Darwin, C. (1859). On the origin of species, Disponível em <www2.hn.psu.edu/faculty/jmanis/darwin.htm> Acesso em 26 mar. 2011.

Goldberg, D., E. (1989). Genetic Algorithms in Search Optimization and Machine Learning, United States, Addison-Wesley.

Lucas, D., C. (2002). Algoritmos Genéticos: Uma Introdução.

Luger, G., F. (2004). Inteligência Artificial, Bookman, 4 ed.

Medina, J.; Müller, R. (2009). A utilização de algoritmos genéticos no desenvolvimento de jogos.

Russel, S.; Norvig., P. (2004). Inteligência Artificial, Campus, 2 ed.

Schwab, B. (2004). AI Game Engine Programming. Hingham, Charles River Media.

Whitley, D. (2000). A genetic Algorithm Tutorial.