FACULDADES BARDDAL

CURSO DE SISTEMAS DE INFORMAÇÃO

MAX RICARDO BENIN

EVOLUÇÃO DE NPC's E ADVERSÁRIOS EM JOGOS DE COMPUTADOR USANDO ALGORITMOS GENÉTICOS

FLORIANÓPOLIS – SC

2007

MAX RICARDO BENIN

EVOLUÇÃO DE NPC'S E ADVERSÁRIOS EM JOGOS DE COMPUTADOR USANDO ALGORITMOS GENÉTICOS

Monografia apresentada como requisito parcial para a obtenção do título de Bacharel em Sistemas de Informação, do Curso de Sistemas de Informação, ministrado pela Faculdades Barddal

ORIENTADOR: Prof. MsC Saulo Popov Zambiasi

FLORIANÓPOLIS – SC

2007

MAX RICARDO BENIN

EVOLUÇÃO DE NPC'S E ADVERSÁRIOS EM JOGOS DE COMPUTADOR USANDO ALGORITMOS GENÉTICOS

Esta monografia foi submetida ao Curso de Sistemas de Informação das Faculdades Barddal, como requisito parcial à obtenção do título de Bacharel em Sistemas de Informação e considerada aprovada pela banca abaixo.

Profª. Drª. Mirela Sechi Moretti Annoni Notare
Coordenadora do Curso de Sistemas de Informação

Prof. MSc. Valter Monteiro Oliveira Jr.
Coordenador de Monografia

BANCA EXAMINADORA

Prof. MSc. Saulo Popov Zambiasi
Orientador

Prof. Eng. Bruno Panerai
Faculdades Barddal

Prof. Dr. Leando Loss
Universidade Federal de Santa Catarina

Florianópolis SC, 03 de dezembro de 2007

Dedico este trabalho primeiramente à meus pais por todo o apoio dado a mim para seguir esta carreira e ao grupo *Kyoto Jazz Massive* pelas horas tocadas em meu player enquanto dissertava esta monografia.

AGRADECIMENTOS

Ao meu orientador Saulo Popov Zambiasi por ter acreditado na idéia desta dissertação, aos colegas de trabalho pela compreensão desde momento tão decisivo em minha vida, novamente aos meus pais por todo o incentivo e compreensão nas ocasiões pelas quais fui obrigado a ausentar-me nas reuniões familiares, ao pessoal do Programadores de Jogos (http://www.pdj.com.br) pela ajuda referente a material de pesquisa e finalmente aos meus colegas de faculdade que vivenciaram todos os momentos críticos de nossa vida acadêmica e souberam, assim como eu, a passar todos os obstáculos impostos.

“Ginger, i ain’t chicken dude, i said hot chicken on the front row”

(Michael Diammond – Metal Skool)

RESUMO

Nos jogos de computador, personagens controlados por computador cuja existência torna-se uniforme e sem qualquer mudança, tende a ser um problema visto que o mercado exige um alto grau de envolvimento entre os jogos e seus jogadores tornando o jogo mais envolvente e desafiador. Como alternativa para a diferenciação entre os comportamentos dos elementos envolvidos em um jogo de computador e parte das técnicas de busca em Inteligência Artificial baseadas na biologia e genética natural, surgem os algoritmos genéticos. Através das idéias de pesquisadores como Darwin e a sua teoria da seleção natural, os Algoritmos Genéticos são capazes de trabalharem em conjunto através de uma população de indivíduos e usar os seus princípios como forma de solução de um determinado problema. Desta forma, o presente trabalho busca aplicar os algoritmos genéticos de forma a possibilitar que cada personagem controlado por computador aja de forma diferenciada e com certo grau de inteligência, objetivando surpreender o jogador em termos de ações e aleatoriedade. Sendo assim, o presente trabalho utiliza técnicas de algoritmos genéticos para criar um DNA para cada indivíduo, com diversas características pertinentes ao mesmo. Estes podem manter a variedade dos mesmos através de técnicas como a reprodução sexuada entre dois indivíduos, mutação de ações em um gene, gerenciamento de tempo de vida e atributos biológicos em geral.

PALAVRAS-CHAVE: Algoritmos Genéticos, Vida Artificial, Jogos Eletrônicos, Inteligência Artificial.

ABSTRACT

On computer games, individuals controlled by computers that have a uniform existence and have no behavior changing are a problem at the point of view business. Today, there are necessary a great level of development and interaction between games and gamers, turning the game more involving and challenger. Genetic Algorithms became as an alternative and part of Artificial Intelligence studies for difference between elements behaviors on computer game. Genetic Algorithms, based on ideas of Darvin about natural selection theory, are capable of work on a population of individuals and find for solutions to certain problem. Therefore, this work use the genetic algorithms principles to turn possible that each individual controlled by computer work with certain degree of intelligence and with different behaviors, It subject to surprise the gamers in a randomized form. As well, this work uses genetic algorithms techniques to create a DNA to each individual, with some important characteristics to them. These can provide a variety of behaviors its adaptation through reproduction between two individuals, mutation of actions, management of life time and biological attributes.

KEY-WORDS: Genetic Algorithms, Artificial Life, Electronic Games, Artificial Intelligence

LISTA DE ACRÔNIMOS

MMOG: *Massive Multiplayer Online Game*

RTS: *Real-Time Strategy*

RPG: *Role-Playing Game*

FPS: *First Person Shooter*

MMORPG: *Massive Multiplayer Online Role-Playing Game*

MMORTS: *Massive Multiplayer Online Real-Time Strategy*

Alife: Artificial Life

NPC: *Non-Playable Character*

IA: Inteligência Artificial

GA: *Genetic Algorithm*

LISTA DE FIGURAS

TABELAS

SUMÁRIO

1. INTRODUÇÃO

A existência da inteligência artificial como forma de entretenimento em jogos eletrônicos parte desde a época dos anos 70 com a virtuose dos jogos alimentados por moedas, os *arcades*[1]. Jogos como *Pacman* (Namco, 1979)*, Space Invaders* (Taito, 1978)*, Donkey Kong* (Nintendo,1983) já possuíam indícios de inteligência, embora precária, por conjuntos de regras simples que definiam determinadas ações de modo que os adversários ficassem menos previsíveis. Essas regras eram conhecidas por serem baseadas na teoria de Alan Turing sobre as Máquinas de Estados ou *State Machines.*

Entretanto, a revolução estava apenas começando. Com o passar dos anos, assim como a evolução dos hardwares de processamento em 3D e a explosão de qualidade gráfica em jogos, a inteligência artificial tornou-se fator crítico para o sucesso em jogos de computador. Esta, ficando até mesmo em ponto decisório entre qual lançamento seria um *best-seller* e até mesmo definindo o futuro de várias empresas produtoras do ramo, fator que se consuma pelas últimas pesquisas.

Segundo Thomé (2006, apud PENHA, 2006) "*O segmento de jogos eletrônicos mundial ultrapassa o faturamento do cinema desde 2003. O mercado brasileiro de games movimentou US$ 100 milhões em 2005 e a tendência é fechar 2006 com um aumento de 30% a 40%* ".

[1] Vulgarmente chamado no Brasil de "Fliperama"

Buckland (2005) discute as diferenças entre a Inteligência Artificial acadêmica e a Inteligência Artificial para jogos de computador. Cada uma faz uso de tecnologias próprias da IA, porém Buckland afirma que o campo acadêmico despreocupa-se com o fator otimização de hardware e tempo.

Por exemplo, alguns pesquisadores em IA ficam perfeitamente felizes em deixar uma simulação rodando por horas, dias e até mesmo semanas (...) tanto tempo quanto for necessário para que eles possam ter um bom resultado a fim de reportar em seus relatórios. Buckland (2005)

Por outro lado, programadores de IA para jogos devem trabalhar com recursos limitados. Os ciclos de processamento e memória variam de plataforma para plataforma e o sistema inteligente deve fornecer apenas uma coisa ao jogador, o entretenimento. Por este fator é importante que os agentes inteligentes ajam com rapidez – contrariando muitas vezes as pesquisas acadêmicas – e também percam, ou seja, não sejam dotados de uma inteligência capaz de tornar impossível a vitória do jogador.

1.2 Objetivos

1.2.1 Objetivo Geral

O presente trabalho tem por objetivo estudar e aplicar técnicas de algoritmos genéticos com o intuito de incorporar comportamentos mais diversificados para NPC's e adversários controlados por computador em jogos eletrônicos.

1.2.2 Objetivos Específicos

Para alcançar o objetivo geral deste trabalho, são necessários alcançar os seguintes objetivos específicos:

- Obter referencial bibliográfico sobre jogos de computador, e Algoritmos Genéticos, assim como jogos que se utilizam de Inteligência Artificial para melhorar seu comportamento
- Obter um modelo e implementação te testes sobre um ambiente virtual para agentes;
- Obter resultados de testes feitos sobre a implementação e o ambiente virtual.

1.3 Justificativa

Segundo Buckland (2002), o mercado de jogos eletrônicos tornou-se mais exigente com o passar dos anos, principalmente com a evolução em nível de hardware que, na última década, aumentou exponencialmente, tornando possível a criação de jogos que possuem uma realidade gráfica quase comparada ao real.

Contudo o entretenimento ficaria limitado se os agentes, que provém a diversão ao jogador, fossem desprovidos de inteligência, uma vez que o público evoluiu junto à tecnologia.

A necessidade de jogos auto-suficientes tornou-se o maior atrativo a esse público tão exigente. Para as produtoras, a possibilidade de determinados títulos possuírem a total autonomia de manipular seus atuantes. A Inteligência Artificial possibilita garantir a atenção do jogador que prima por uma interação maior com o jogo, não havendo perdas de interesse tão rapidamente pelo jogo, o que obriga as empresas a lançarem novos títulos para não perderem mercado para concorrentes.

1.4 Metodologia

Como metodologia para se alcançar os objetivos deste trabalho, são necessários os seguintes procedimentos:

- Pesquisar sobre jogos, inteligência artificial e, mais especificamente, algoritmos genéticos em livros, artigos e demais materiais disponíveis na Internet;
- Estudar um ambiente virtual para a implementação do protótipo;
- Implementar os agentes com comportamentos próprios baseados em seu DNA e o algoritmo genético, propriamente dito;
- Testar os protótipos para afirmar a sua aplicabilidade.

2. JOGOS ELETRÔNICOS

> Os jogos de computador são os melhores rivais a tudo o que Hollywood tem de melhor a oferecer como visuais, efeitos, trilhas sonoras e pura adrenalina. Mas jogos são uma forma de entretenimento como qualquer outra; eles podem deixar o jogador grudado em seus monitores por horas até o fim. DAWSON (2004, p.11)

2.1 História

As pesquisas envolvendo jogos de computador que se tem datada, começaram por volta de 1952, por A.S. Douglas em seu doutorado. Douglas criou uma versão gráfica do jogo-da-velha que rodava no computador de válvulas *EDSAC*. Logo após, em 1958, no laboratório de pesquisas militares *Brookhaven National Laboratory,* cientistas criaram um simulador simplificado do tênis, esporte mundialmente famoso que consiste em rebater uma bola com o auxílio de uma raquete através de uma rede, atingindo a quadra do adversário. Por conseguinte houve quase cinco décadas de desenvolvimento constante em jogos até que em 1961, estudantes do MIT (*Massachusetts Institute of Technology,)* desenvolveram o jogo *Spacewar,* inspirando-se nos livros do autor E.E Smith. O jogo foi baseado em batalhas aeroespaciais e nunca teve retorno financeiro por parte dos estudantes[2].

A partir do *Spacewar*, o mundo dos jogos cresceu exponencialmente. Contratações foram feitas por parte dos investidores interessados e jogos como *Pong* e *Tank* e empresas como *Atari* e *Namco* surgem tal que até hoje elas mantêm-se no mercado de jogos.

Com o avanço da computação na época de 70 e 80, foi possível trazer aos lares computadores pessoais ou PC[3] (*Personal Computer,)* aumentando assim a gama de produção

[2] É adaptado em 1971 por Nolan Bushnell para a criação do *Computer Space*, primeiro *mainframe* portátil para jogos chamados hoje de *Arcades* ou vulgarmente no Brasil de Fliperamas.
[3] Nome dado graças ao sucesso do computador pessoal fabricado pela IBM em 1981 chamado IBM-PC.

de jogos e criando de uma vez por todas a divisão do conceito entre os dois eletrônicos que mais movimentam capital no mundo: O videogame e o computador.

Ao longo de quase cinqüenta anos de pesquisas em desenvolvimento de jogos foram explorados diversos ramos, formas e estilos de jogabilidade, variando entre esportes, tiro, aventura, RPG, jogos executados singularmente, duplamente, modo multijogador (*multiplayer*) e etc.

A partir dessa premissa, novas formas de tornar jogos eletrônicos mais próximos da realidade foram utilizadas. Surge, então, a valorização da inteligência artificial para jogos ou *Game AI.*

2.2 Principais estilos de jogos.

A seguir são apresentados de forma breve os estilos de jogos mais jogados nos últimos anos. Porém é feito um aprofundamento maior, juntamente com o tema proposto a esse trabalho, somente em relação ao gênero massivo ou *MMOG (massive multiplayer online game)* composto pelos estilos RTS e RPG.

2.2.1 RTS (Real-Time Strategy)

Jogos de estratégia são os pioneiros da inteligência artificial em jogos (*Game AI).* Estes necessitam de diversas rotinas complexas em inteligência artificial para se tornarem jogáveis. A inteligência em RTS's é particularmente desafiadora e exige grandes sistemas baseados em unidades inteligentes (*unit-AI)* e também sofisticados assim como estratégicos agentes inteligentes controlados por computador. A onipotência dos jogos RTS está baseada em confrontos entre nações e/ou raças.

Jogos baseados em turnos (*turn-based)* como as da série Civilization (Microprose, 1991) usam sistemas baseados em jogos de tabuleiro como xadrez, WAR e batalha naval. Nesse estilo de estratégia a inteligência é baseada nos turnos jogados pelo usuário. Os agentes inteligentes são capazes de, em diversos níveis de dificuldade, calcular possíveis tomadas de decisões, baseando-se nas decisões feitas pelo jogador. Como por exemplo, o sistema pode efetuar um ataque massivo com um determinado exército a uma determinada região ou nação rival e/ou efetuar uma contra-resposta táctica baseando-se no nível intelectual dos personagens controlados pelo jogador. Tais fatores contribuintes como religião, economia, exército, sociedade, filosofia, nível científico são fatores decisórios para o computador definir uma solução inteligente que dificulte a vitória da batalha em questão pelo jogador tão facilmente.

Títulos como a série *Warcraft* (Blizzard, 1995), *Age of Empire* (Ensemble, 1999), *Age of Mythology* (Ensemble, 2004), *Starcraft* (Blizzard, 2000) e *Rise of Nations* (Ensemble, 2005) são oriundos de outra classe do gênero RTS. Nesta, cada personagem controlado por computador torna-se um agente independente, com a total autonomia condizente com a sua respectiva função. Essa categoria é jogada a partir de um cenário, ou mundo virtual, com suas respectivas condições físicas (terrenos montanhosos, lagos, mares, neve, etc.) e cabe ao agente inteligente explorar, sobreviver e atacar nas condições a que lhe foram dadas.

A particularidade principal desse estilo é que existe a possibilidade de criar dezenas, ou até mesmo centenas, de unidades controladas por computador, sendo divididas em diversas classes (camponeses, soldados, magos, médicos, veículos dentre outros...) que simulam de forma independente ações de raciocínio para interar com o jogador. Esses tipos de agentes inteligentes possuem rotinas de procura, ou *path-finding,* que nada mais são do que a possibilidade de prover ao agente o mapeamento de rotas para definir um bom, ou ótimo,

caminho dentro do cenário virtual, a fim de alcançar seu objetivos, sejam eles explorar a procura do inimigo ou até mesmo buscar recursos naturais em determinadas áreas para manter a sobrevivência do meio em que ele convive.

2.2.2 FPS (First-*Person Shooting).*

O gênero *First-Person Shooting,* ou Tiro em Primeira Pessoa, é caracterizado pelo estilo em que o ponto de vista do jogador é justamente o do protagonista ou observador.

O primeiro jogo, considerado o pai dos FPS, chama-se *Wolfenstein* 3D (idsoftware, 1993). Ele foi considerado o primeiro, pois seguia a regra dos FPS's atuais. Este já fornecia ao jogador um cenário em primeira pessoa aonde o jogador poderia explorá-lo, fazendo uso de armas de fogo, e outros itens. Outra grande característica do FPS é o uso massivo de uma ferramenta real, os HUD's[4] (*Heads-up Display)* composto de uma barra, geralmente na base da tela de jogo, aonde continham informações vitais para a orientação do jogador, tais como saúde, armas, munição, mapa e etc.

Hoje no campo de FPS grandes sucessos como *Half-life* (Valve, 1998) e *Unreal Tournament* (epic, 1999) carregam uma ótima bagagem de conhecimento no ramo da inteligência artificial. Unidades ou *bots* controlados pelo computador possuem algoritmos táticos em vários níveis de dificuldade que vão desde a busca pelo jogador no cenário virtual até o nível de acurácia dos projéteis disparados contra o mesmo. Outro título importante criado pela Sierra Studios foi o *SWAT 3: Close Quarters Battle* (Sierra, 1999), com um sistema de comportamento randômico garantindo que o jogo e a dificuldade sejam diferentes

[4] Hud's foram pioneiros na aviação militar trazendo informações vitais para o plano de combate aéreo sejam helicópteros ou caças. Tais informações eram mostradas no campo de visão do piloto mesmo sem atrapalhar o seu plano. Hud's foram implementados a partir da década de 70 na aviação comercial e em outros meios de transporte.

a cada vez que o usuário o jogue, possibilitando também que os *bots* aprendam conforme o nível do usuário.

Nem sempre se deve associar FPS a combate armado, uma vez que títulos como D (*s.n, 1998*) apresentavam as mesmas premissas de um jogo em primeira pessoa mas com o diferencial dos objetivos voltados para resolver quebra-cabeças e mistérios.

2.2.3 RPG (*Role*-Playing *Game).*

O RPG funciona como num teatro, o usuário interpreta um personagem e segue um roteiro definido pelo diretor. No mesmo sentido, baseado no RPG de tabuleiro, o RPG eletrônico segue a mesma regra de interpretação de papéis, com a diferença de que o mestre (analogamente ao diretor teatral) é controlado pelo computador, bem como os outros atuantes.

Outra característica marcante desse sistema de jogo é que ele é baseado em pontos de atributos, ou seja, o personagem (assim como RTS, com um mundo virtual a sua disposição para explorar) depende da história, que geralmente é baseada em contos medievais. Os usuários possuem pontos como: destreza, inteligência, energia vital, energia mágica, carma, *darma*, velocidade de ataque mágico, velocidade de ataque físico, quantidade de dano mágico e físico por golpe e muitos outros que podem ser moldados de acordo com a preferência do jogador juntamente com as condições que o jogo fornece ao mesmo. Esses estilos de pontos são dados aos usuários em grupos ou únicos. O jogador, a cada missão cumprida ou oponente derrotado, tem direito a pontos numéricos que podem ser devidamente distribuídos aos seus atributos.

Por fim a modalidade de jogo continua parecida com o RPG de tabuleiro, em que o jogador, quando está explorando o mundo virtual que lhe foi proposto, está livre para tomar qualquer atitude que o mundo virtual pode fornecer. Porém, quando entra em sistema de

batalhas, assume o modo *turned-based* ou modo em turnos, nos quais são divididos de acordo com os inimigos que estão no cenário. Neste sistema cada usuário e adversário tem direito de atacar uma ou duas vezes por turno.

Títulos que ficaram mundialmente famosos como *Final Fantasy* (Square Enix,87), *Breath of Fire* (Capcom,93), abriram portas para a produção desse gênero para outras empresas incluindo ambas citadas que até hoje cultivam respectivamente cinco e treze continuações de seus maiores sucessos.

2.2.4 MMOG (Massive Multiplayer *Online Game).*

A característica principal dos jogos massivos online, como o próprio nome incita, é a possibilidade de milhares ou até mesmo milhões de jogadores interagirem uns com outros ao mesmo tempo.

O fato é que um MMOG não é definido como um gênero, mas sim como um paradigma onde é usada conectividade entre os usuários em um determinado servidor que administra essa interatividade. Graças a essa tecnologia, hoje existem vários gêneros usando-a como base principal, tais como *MMORPG* e *MMORTS.* Fato este responsável pela arrecadação milionária nos últimos cinco anos.

2.2.5 MMORPG (Massive *Multiplayer Online RPG).*

O MMORPG, talvez o mais jogado nos últimos anos, consiste na conectividade de milhares de jogadores através de um servidor para vivenciar uma mesma história e seguir as regras de jogo pregadas no *RPG*. Jogos como *Lineage II* (NCsoft, 2003), lançou em 2007 a sua quinta crônica, ou seja, em seus quatro anos de existência. Este jogo já transcendeu quatro histórias, ou sagas, e nesse ano vivencia a sua quinta, com elementos, missões, e personagens

diferentes. Porém, mantendo o mesmo mundo virtual apresentado desde a sua aparição em 2003.

2.2.6 MMORTS (*Massive* Multiplayer *Online RTS).*

O MMORTS é um gênero que procura reunir conexões massivas de jogadores de RTS e concentrá-lo em um campo de batalha. A diferença entre o MMORPG é que o jogador assume um papel de general, rei ou líder. Ele por sua vez não fica responsável somente por um personagem, mas sim por construir e administrar um pequeno reinado e controlar um esquadrão de batalhas com a finalidade de ser vitorioso perante os outros adversários no jogo.

Esse gênero não obteve sucesso como o MMORPG, permanecendo somente alguns títulos e empresas apostadoras no gênero. Tal fato é dado, pois uma vez criado o seu legado dentro do jogo, ele continua ativo mesmo que o jogador se ausente do mundo virtual. Para um *Game Design,* há a perda de interesse no jogo, pois o usuário, mesmo desconectado, pode ser atacado a qualquer momento.

Gêneros famosos e bem sucedidos foram: o *Mankind*, criado pela empresa francesa Vibes (Vibes, 1998), o *Dreamlords* (Lockpick Entertainment, 2007) e o mais recente *Darkness and Light* (GameOn, 2007).

2.2.7 *Alife Game*

Esse gênero pouco explorado trata-se da simulação do sistema biológico de um determinado organismo, ou até mesmo simular o comportamento de seus protagonistas. Os precursores nesse estilo são os títulos lançados pela empresa Maxis, como a série *SimCity* (Maxis, 1985). Este consiste de um simulador de cidades que tem como objetivo principal a

administração da mesma com todos os fatores hoje vistos por qualquer órgão administrativo municipal.

Já o The Sims (Maxis, 2000) é um simulador onde o usuário é responsável por gerir a vida de um grupo de pessoas que habitam uma determinada residência ou meio. Vários simuladores foram lançados pela Maxis, tais como simulação de fazenda, zoológico, dentre outros, e o seu mais recente título anunciado é o SPORE (Maxis, 2009). Este último, previsto para 2009, utiliza algoritmos complexos de inteligência artificial para que seja possível simular um organismo microscópico, assim como trabalhar na sua evolução ao longo do jogo.

Outras empresas apostam forte na ciência do Alife (*Artificial Life)*, tais como a LionHead Studios com a série *Black & White* (Lionhead Studios, 2001), jogo em que você é um deus e tem como objetivo principal criar seres em uma ilha virtual e Hasbro Interactive, com o *RollerCoaster Tycoon* (Hasbro Interactive, 1999), simulador de parques de diversões.

2.3 Considerações

Embora o capítulo 2 seja muito sucinto em relação aos estilos de jogos que usam técnicas de algoritmos genéticos, ainda existem outros estilos que podem ser usadas tais técnicas. Para manter o foco neste trabalho, são citadas apenas estilos aos quais o algoritmo genético é mais bem aplicado devido a sua regra de usar a biologia como referência e todas as ramificações que se incluem nela, tais como seleção natural, evolução, genética, hereditariedade, mutação dentre outros.

3 PROGRAMAÇÃO GENÉTICA

O presente capítulo apresenta um estudo sobre algoritmos genéticos e vida artificial. Porém, primeiramente são apresentados alguns conceitos introdutórios, mas importantes, envolvendo a Inteligência Artificial como um todo, pois os algoritmos genéticos e vida artificial se enquadram dentro deste campo.

3.1 Inteligência Artificial

A ciência evolui do enorme desejo humano de entender e controlar o mundo (Mitchell, 1999).

Segundo (Champandard 2003), desde o começo, basicamente na era *Pong* (Atari,1972) e Pac-man (Namco, 1979) a Inteligência Artificial obteve um papel inegável para os jogos eletrônicos. Hoje, longe dos primeiros conceitos, a IA continua sendo uma necessidade para o entretenimento. Agora que os jogos atingiram em outros aspectos (como som e gráficos) níveis impressionantes, o foco é mais voltado para a IA.

Champandard também cita que a Inteligência Artificial nada mais é que o "cérebro por trás de máquinas cibernéticas" e que para acadêmicos é uma fonte inesgotável de desafios e excitações.

Champandard explica que existem dois tipos de conceitos envolvendo a Inteligência Artificial. Embora esses dois conceitos tenham sentidos diferentes, cabem-se perfeitamente para o conceito da IA para jogos de computador. O primeiro trata-se de definir a IA como a re-criação da inteligência, aplicada assim, por máquinas. O segundo diz que a IA

é um conjunto de técnicas acadêmicas, métodos de pesquisa e problemas que caem dentro de um subgrupo da Ciência.

Historicamente, ser inteligente trata-se de um termo usado pelo ser humano para descrever a si mesmo. É simplesmente a habilidade que o ser humano possui que ajuda a distingui-los de animais ou plantas, embora nos dias de hoje a inteligência seja usada para distinguir capacidades entre os próprios seres humanos.

O presente capítulo tem como finalidade abordar alguns conceitos envolvendo a Inteligência Artificial.

3.1.2 Habilidades Universais

Champandard (2003), explica que apesar de o ser humano possuir suas qualidades inteligentes especiais e únicas, a inteligência abrange um fator universal em que todos os seres vivos, ou a maioria deles, são capazes de ter em comum, Champandard cita dois exemplos de Inteligência Biológica, como a exaustão e a neutralidade.

Vários pesquisadores afirmam que a Inteligência Biológica pode sim ser reproduzida e que a inteligência não é exclusivamente humana.

3.1.3 Definição de Inteligência

Para um simples entendimento, a inteligência pode ser vista como um conjunto de habilidades que o ser humano tem para resolver problemas através de recursos limitados. Habilidades como planejamento, pensamento abstrato, criatividade e o aprendizado são alguns dos fatores mais importantes do aspecto da inteligência humana.

Champandard (2003), enfatiza que outros seres vivos não são semelhantes ao ser humano. Porém, possuem certos aspectos inteligentes como, por exemplo, alguns animais que

possuem uma inteligência suficiente para sobreviver em um determinado meio ou até mesmo colônias de insetos que são capazes de se adaptarem rapidamente a um determinado meio para proteger seu ninho.

A Inteligência Artificial para jogos procura justamente representar essas habilidades humanas através de máquinas. Jogar um jogo não é apenas poder e reflexos rudimentares, jogos devem ser também ferramentas desafiadoras aonde o ser humano tenha a capacidade de usar recursos para resolver determinados problemas envolvendo sua inteligência.

Segundo Champandard (2003), personagens controlados por computador são atores de teatro, só que com a diferença de que não são contratados. Eles são apenas programados de forma que sintetizem toda a habilidade inteligente que o ser humano faz uso, a fim de tornar o ambiente virtual com o maior realismo possível. Tais personagens controlados são definidos como NPC's (*Non Playable Characters)* ou personagens não jogáveis.

3.1.4 Campo Científico

O marco da inteligência artificial pode ser creditado desde o início da era computacional. Os primeiros pesquisadores – Allan Turing, John Von Neumann, Norbert Wiener e outros – acreditavam que com a evolução e o investimento no campo da computação, as máquinas poderiam ser dotadas de inteligência, simulando vida e se auto-replicando, obtendo o auto-aprendizado e controlando o seu próprio meio. Tais pioneiros tinham, como base, pesquisas não só na área eletrônica, mas também nas áreas biológica e psicológica, o que pode explicar o fato dos mesmos usarem ambientes naturais como os seus maiores guias para fins de estudos.

Segundo (Mitchell 1999), a partir da década de 80, a inteligência artificial baseada na computação biológica deu um salto considerável, surgindo com grande importância nas comunidades de pesquisa.

3.1.5 Agentes Inteligentes

Um agente é tudo o que pode ser considerado capaz de perceber seu ambiente por meio de sensores e agir sobre esse ambiente por intermédio de atuadores (Russel & Norvig, 2004, p.33).

Seres humanos podem ser considerados agentes, segundo Russel & Norvig (2004), um agente humano possui sensores como olhos, nariz e ouvidos e possui atuadores como boca, pernas e mãos. O mesmo acontece com agentes robóticos que possuem uma série de sensores como infravermelho e motores que servem como atuadores.

Um agente também pode ser representado por meio de software, aonde recebe uma determinada seqüência de bits como sensação e responde com outra seqüência como atuador, Ex, Um agente em um cenário virtual percebe a presença de outro agente e como resposta se desloca até o mesmo. Neste caso o primeiro agente recebeu uma instrução de posição (sensor) e logo em seguida se locomoveu até o segundo agente (atuador) e assim realizando a impressão na tela de sua nova posição.

3.1.6 Agentes Racionais

Russel & Norvig (2004) explicam que um agente se torna racional quando o mesmo, para cada seqüência de percepções possíveis seleciona a que venha maximizar o seu desempenho.

Em exemplo podemos citar um aspirador de pó, Russel & Norvig põem em dúvida se aquele determinado agente que controla o aspirador de pó é racional ou não. Para tornar um agente racional primeiro é preciso informar qual é a medida de desempenho, o é conhecido sobre o ambiente e quais são os sensores e atuadores que este agente possui. Os mesmos autores supõem:

1. Ao longo de 1.000 períodos de tempo, a medida de desempenho oferece um ponto para cada quadrado limpo em cada período de tempo.
2. A geografia do ambiente é conhecida e é a prioridade, mas a distribuição da sujeira bem como a posição inicial do agente não é previamente conhecida. Os quadrados que estão limpos permanecem limpos e a aspiração limpa o quadrado atual. As ações “Esquerda” e “Direita” movem o agente respectivamente para esquerda e direita contanto que essa premissa não leve o agente para fora da geografia já conhecida, caso isso ocorra, o agente permanece aonde está
3. As únicas ações disponíveis são Esquerda, Direita, Aspirar e NoOp (não fazer nada).
4. O agente consegue perceber corretamente sua posição e se essa posição contém sujeira.

Russel & Norvig afirmam que sob as circunstâncias citadas acima, o agente seria um agente racional e que também seria irracional caso toda a sujeira fosse limpa, tendo em vista que ele receberia uma penalidade de um ponto para cada movimento a esquerda ou à direita. Os autores citam que em uma ocasião parecida, um agente racional teria uma

instrução para parar de atuar caso a sujeira fosse totalmente retirada e que retornasse a pesquisar os quadrados se os mesmos retornassem a ser sujos.

Um agente racional deve ser necessariamente um agente autômato, ou seja, não se basear apenas pelas informações criadas pelo projetista, mas sim ter a capacidade de aprendizado. No caso do aspirador, um agente que pudesse prever aonde e quando aparecerá mais sujeira funcionará melhor que um agente incapaz de fazer essa previsão.

É importante ressaltar o comentário de Russel & Norvig a despeito da diferença entre onisciência e racionalidade. Segundo os autores, racionalidade não é sinônimo de perfeição e trata somente a forma de maximizar o desempenho esperado tendo em vista que a onisciência é impossível na realidade.

3.2 Computação Evolucionária

A computação evolucionária é conhecida desde o início da era computacional. Os primeiros cientistas da computação – Alan Turin, John Von Neumann, Norbert Wiener e outros – eram motivados pela possibilidade de um ser computacional ser dotado de inteligência suficiente de modo que pudesse se auto-replicar e através de uma capacidade de adaptação, pudesse aprender e controlar o meio em que vive (Mitchell, 1999).

Segundo Mitchell (1999), muitos destes pesquisadores admitiam possuir um interesse maior na biologia e na psicologia do que na eletrônica propriamente dita guiando-se por sistemas naturais para poder obter suas visões e teorias sobre a inteligência computacional.

A tecnologia computacional teve seu princípio por meio de uma “grande máquina calculadora” de trajetórias de mísseis e códigos militares. Porém, é inegável que, nos dias de hoje, os computadores são aplicados para modelar o comportamento do cérebro, imitando o

aprendizado humano e simulando evoluções biológicas. Primeiramente usados no campo das redes neurais, segundo em aprendizado de máquinas e em terceiro no que foi chamado de "computação evolucionária" o que torna os algoritmos genéticos o exemplo mais proeminente.

A computação evolucionária teve início na década de 50, onde muitos cientistas da computação estudaram sistemas evolutivos com o intuito de usar como uma ferramenta de otimização de problemas na engenharia. Mitchell (1999) explica que a idéia por detrás desses sistemas evolutivos era envolver uma população de soluções para um determinado problema, inspirando-se na variação genética natural e no conceito de seleção natural, criado por Charles Darwin.

Na década de 60, Rechenberg introduziu as "estratégias evolucionárias", um método usado para otimizar valores para dispositivos como aerofólios, tornando-o mais adaptativo. Mitchell (1999) afirma, entretanto que esta idéia saiu de sua concepção sendo desenvolvida na prática em 1975 por Schewfel.

Mitchell (1999) enfatiza que o campo das estratégias evolucionárias apesar de trabalhar independentemente, recentemente se uniu à comunidade do campo de algoritmos genéticos para interagirem em conjunto.

Fogel, Owens e Walsh, na década de 70, introduziram o conceito da "programação evolucionária", cuja técnica visava impor máquinas de estados para cada candidato a uma solução. Tais candidatos possuíam seus estados mutacionados a fim de se encaixarem em uma solução mais próxima da viável ou a viável propriamente dita.

Juntos, as estratégias evolucionárias, a programação evolucionária e os algoritmos genéticos formam a união do campo da computação evolucionária.

3.3 Algoritmos Genéticos

Os algoritmos genéticos (GA's) foram inventados por John Holland na década de 60 e foi desenvolvido por seus estudantes e colegas na Universidade de Michigan nas décadas de 60 e 70. O principal objetivo não era desenvolver algoritmos que solucionassem problemas específicos, mas sim estudar as formas de adaptação que ocorriam na natureza e assim desenvolver formas de tais mecanismos de adaptação para serem importados para os computadores.

Mitchell (1999), cita que John Holland em seu livro chamado *Adaptation in Natural and Artificial Systems* apresenta os algoritmos genéticos como uma abstração da evolução biológica tornando-se um método que move uma determinada população de "cromossomos" (strings de 0's e 1's) para uma nova população usando o método de seleção natural (teoria de Charles Darwin) juntamente com as métricas da genética atual como *crossing-over,* mutação e inversão.

3.3.1 Genética Natural

Buckland (2002) cita que organismos vivos são grandes compilações de células. O autor explica que para obter um melhor entendimento de como o funcionamento dos algoritmos genéticos ocorre, é necessário entender primeiro os seguintes conceitos dados na biologia moderna.

- **DNA**: Conhecido como Ácido Desoxirribonucléico, é uma molécula orgânica que determina o funcionamento de vários organismos vivos.
- **Cromossomos:** Segmentos de uma longa cadeia de DNA contendo vários Genes.

- **Gene:** Segmento de um cromossomo responsável por carregar as informações genéticas de cada ser vivo. Ex: Cor dos olhos.
- **Alelos:** É cada uma das várias formas de um mesmo gene. Um alelo fica posicionado em uma determinada posição do cromossomo na qual é nomeada de *lócus*. Por exemplo, um gene que determina a cor dos olhos de uma pessoa pode possuir diversas variações do mesmo, podendo assumir várias cores (verde, azul, castanho...), cada variação de cor pode ser chamada de alelo.

Buckland (2002) acrescenta que uma coleção de cromossomos é chamada de genoma, ou seja, é o mapeamento de todas as características de um determinado ser vivo. A partir desta premissa, o autor cita outras duas definições, o genótipo e o fenótipo. Segundo Buckland (2002), o genótipo é a característica biológica que cada genoma possui enquanto o fenótipo trata-se da característica física ou visível imposta por cada genótipo. Para melhor entendimento o autor faz uma comparação em uma linha de montagem de carros, aonde as especificações de design ou *blueprint's* representam um genótipo enquanto o fenótipo representa o carro montado e pronto para sair da linha de montagem.

Conforme Buckland (2002), um organismo é considerado bem sucedido se ele e seu parceiro gerarem um organismo filho, o que, esperançosamente, ocorrerá a seus sucessores, reproduzindo-se entre si. Buckland cita que para isto, o organismo deve ser bom em diversas tarefas, tais como encontrar comida e água, defender-se contra predadores e serem atraentes aos seus parceiros. Muitos destes atributos dependem em muitas vezes do genótipo do organismo. Alguns dos genes destes organismos ascendem os atributos com melhor êxito, enquanto retardam ou anulam os atributos sem sucesso.

Segundo Buckland (2002), a medida usada para medir o sucesso de tais organismos se chama aptidão. Quanto mais apto um organismo for, melhores chances ele terá de criar uma prole.

O autor supracitado também explica que quando dois organismos se relacionam e se reproduzem, seus cromossomos se unem para formar um grupo de cromossomos completamente novo, os que contêm genes de ambos os seus progenitores. Este processo é chamado de recombinação ou *crossing-over.*

Segundo Mitchel (1999), os organismos que possuem cromossomos arranjados em pares são chamados de diplóides enquanto os organismos cujos cromossomos sejam díspares são considerados haplóides. A maioria dos seres que se reproduzem sexualmente é diplóide, incluindo os seres humanos que possuem 23 pares de cromossomos em cada célula do corpo (células somáticas). Durante o *crossing-over*, se diplóides, os organismos cedem seus respectivos cromossomos a fim de formar um ser diplóide por completo. No caso do ser humano, são cedidos 23 cromossomos de cada indivíduo (vindo de seus gametas) formando assim um ser com 46 cromossomos. No caso de reprodução haplóide, os genes são trocados entre os cromossomos pertencentes a cada indivíduo.

Buckland (2002), afirma que a progênie herdará na maioria das vezes os genes bons, o que os tornam mais eficientes e com maiores sucessos do que seus progenitores (por exemplo, melhor mecanismo de defesa contra predadores). Isso não significa que a prole não herde também genes ruins de seus progenitores o que nesse caso pode ocorrer dos mesmos não serem mais possíveis de se reproduzirem (viabilidade). O mais importante segundo Buckland (2002), é que quanto mais adaptável a prole for, melhor será sua reprodução e por sua vez a passagem de seus próprios genes para seus descendentes. O autor cita também que a

cada geração, um determinado organismo possui mais tendências à sobrevivência e relacionamentos entre si do que os seus antecessores.

Mitchell (1999) salienta que uma prole está sujeita à mutação, e que muitas vezes é dada como um erro na cópia dos genes, portanto é uma mudança arbitrária. Buckland (2002) sustenta o fato de que a probabilidade de um organismo sofrer mutações é pequena, porém nunca descartável, sendo que provavelmente a maioria das mutações tem efeitos de desvantagem, algumas não afetam em nada o sucesso (aptidão) e outras darão vantagens distintas perante as criaturas de uma mesma espécie.

3.3.2 Conceitos Técnicos

Segundo Russel & Norvig (2004), um algoritmo genético é uma variação de um algoritmo de busca em feixe estocástica[5], onde os estados sucessores são gerados a partir de dois estados pais. A diferença entre um algoritmo de busca convencional é que os algoritmos genéticos lidam com a teoria da reprodução sexuada.

De acordo com Laramée (2002), um algoritmo genético funciona da seguinte maneira:

1. Criar a primeira geração de uma população de organismos randômicos
2. Testá-los em cima do problema proposto à solução. Se o melhor organismo chegar ao desempenho desejado, pare.
3. Pegue as melhores performances e relacione-as aplicando operações genéticas como *crossover* e mutação. Adicione novos organismos randômicos à população a fim de introduzir uma nova variedade.

[5] Método de busca similar ao algoritmo genético pois também usa as teorias da seleção natural estipuladas por Darwin bem como possui sucessores, organismos e valores de adaptação ou *fitness.* Outra características é que o algoritmo de busca em feixe estocástico faz a analogia a reprodução assexuada.

4. Retorne ao passo 2.

3.3.2.1 População e Indivíduo

Segundo Russel & Norvig (2004), uma população são conjuntos de *n* indivíduos gerados aleatoriamente. E cada indivíduo é representado por uma cadeia em um alfabeto finito aonde na maioria dos casos são representadas por uma cadeia de valores 0 e 1.

Já Buckland (2002) define para cada indivíduo é gerado um cromossomo digital aleatório aonde o mesmo fará parte dos candidatos a representar uma solução ao problema. A idéia é que cada indivíduo apresente uma solução diferente da outra.

Os indivíduos que fazem parte de uma determinada população devem possuir um mesmo tamanho finito de cromossomo para que seja possível posteriormente a permuta de "genes" entre eles. Um exemplo de cromossomo digital pode ser dado da seguinte forma:

01010010100101001111

(exemplo de um cromossomo de tamanho vinte, Buckland, 2002, p. 99)

3.3.2.2 Função de *Fitness* ou aptidão

A função de *fitness* ou função de aptidão é usada para escolher o organismo mais apto a se reproduzir, a fim de maximizar (ou minimizar) o problema proposto.

Segundo Laramée (2002), é possivel estimar a aptidão

> [...] aplicando uma função de avaliação para cada organismo, afim de o mais apto a reprodução. Assumindo que os AGs trabalham com um grande tamanho populacional passando por diversas gerações, a função de avaliação deve ser a mais rápida possível. No caso de jogos para computador, cada candidato será avaliado medindo o seu sucesso em uma simulação dentro do ambiente virtual do jogo.

Buckland (2002) acrescenta a existência da roleta de seleção na qual é feita a seleção dos indivíduos dispostos por sua aptidão. O autor cita que a roleta de seleção pode ser representada por um gráfico em formato de torta ou uma roleta, em que quanto mais apto o indivíduo for maior será o pedaço do gráfico que ele irá pertencer e esta seleção é feita aleatoriamente como mostra a figura 1.

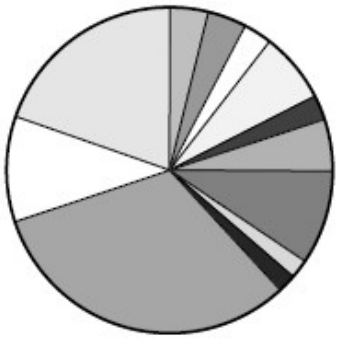


Fig 1 – Roleta de seletção natural.

Laramée (2002) cita que logo após as avaliações estiverem completa, a função de aptidão deve ser aplicada aos resultados para determinar qual organismo terá a chance de participar da próxima geração e em quais proporções. Segundo o autor existem as seguintes estratégias:

- **Amostra Estocástica:** Cada organismo recebe uma probabilidade de reprodução e isso é igual a sua aptidão dividida pelo total de aptidões da população.
- **Amostra Estocástica por Sobra:** Divide cada aptidão pela média de aptidão da população. Se o resultado for maior que 1, o organismo se reproduz o numero de vezes igual à parte inteira do quociente desta

divisão. Preencha o resto da população randomicamente, usando os restos dos quocientes como probabilidades.

- **Ranking ou Classificação:** Indivíduos perto do topo da tabela de aptidão se reproduzem automaticamente mais vezes do que outros.

Na seqüência Laramée (2002), cita que o uso do método de *ranking* ou classificação é o melhor indicado, pois força uma convergência mais rápida diferentemente das alternativas que em muitos casos não selecionam os organismos por preferência.

3.3.2.3 *Crossover*

O *crossover* ou *crossing-over* é um método, como supracitado nos capítulos anteriores, destinado a permuta de genes para a geração de outro indivíduo.

Segundo Russel & Norvig (2004), o conceito de *crossover* é escolher ao acaso um ponto a ser cruzado dentre as duas cadeias de cromossomos.

Laramée (2002) salienta que este método é o mais simples dos diversos que podem ser aplicados ao *crossover*, aonde é chamado de **ponto-único**. Após o ponto ao caso ter sido escolhido, este é trocado com a outra cadeia de cromossomos partindo do início do ponto até o fim da cadeia. O autor cita o seguinte exemplo aonde se necessita realizar o *crossover* a partir do gene de número 3 entre **AAAAA** e **BBBBB**. O resultado neste caso é: **AABBB** e **BBAAA.** O autor cita também que este método possui propriedades limitadas dentre elas dá-se ao fato de uma cadeia de cromossomos sempre ser separada do ponto aleatório até o fim da cadeira, devido a este fato o Laramée (2002) cita mais três métodos:

- ***Crossover em dois pontos (ou binário):*** Este método é similar ao método de um único ponto porém com a diferença de que é possível selecionar dois

pontos em uma determinada cadeia e usar a seqüência de genes entre eles para haver a permuta.

- **Inversão:** Raramente usado, é feita a inversão dos genes trocados para que eles se pareçam menos com a seqüência de seus pais. É chamada a atenção deste método pois não possui valor algum se a ordem das posições de seus genes tem importância.
- ***Crossover* Uniforme:** Este método consiste em escolher de qual pai o gene será pego através de uma probabilidade e não usando pontos de cruzamento. Esta técnica prima por uma diversidade e possui maior efeito em populações pequenas.

Conforme a figura abaixo é mostrada um exemplo de *crossover* por um ponto.

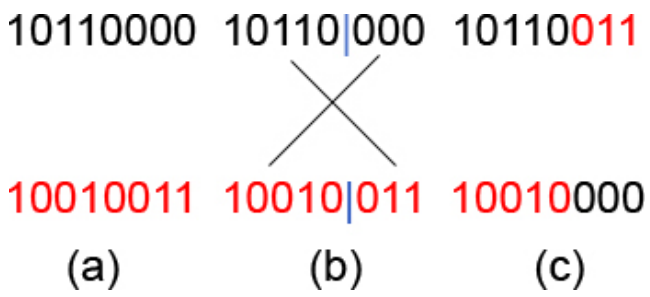


Fig 2

(a) Dois indivíduos são escolhidos, (b) É selecionado os pontos de cruzamento, (c) As características são recombinadas gerando dois novos indivíduos

3.3.2.4 Mutação

A mutação em algoritmos genéticos é usada para controle e manutenção da diversidade genética da população. O processo consiste em trocar um ou mais genes arbitrariamente dentro da cadeia genética.

Segundo Laramée (2002), o uso da mutação é necessário quando alguns alelos se perdem da população tornando assim possível re-introduzir o mesmo novamente a cadeia e dá-lo uma segunda chance para provar sua aptidão. Esta técnica é conhecida como **Re-introdução de alelos extintos.** Outra funcionalidade é para evitar a **Convergência Prematura**, isso ocorre quando uma grande população é feita de organismos similares uns aos outros. Para o autor, este é um fator crítico, pois em determinados casos o algoritmo converge prematuramente a uma determinada solução sem que esta seja uma das melhores possíveis.

3.3.2.4.1 Taxa de Mutação

Para que a mutação tenha um papel importante para a aproximação de uma solução ótima, a troca entre os alelos (0 se torna 1, e 1 se torna 0) deve ser estipulada através de um valor correspondente à probabilidade de mutação, para justamente controlar a quantidade de agentes mutacionados em uma determinada geração.

Buckland (2002), cita que usualmente a taxa de mutação deve conter um valor muito baixo, como por exemplo 0.001.

A figura a seguir denota o processo de mutação entre uma cadeia de bits.

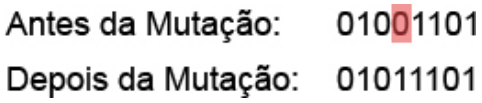


Fig 3 – Mutação

3.3.2.5 Exemplo de Algoritmo

A figura 4 apresenta o exemplo padrão em diagrama de um algoritmo genético.

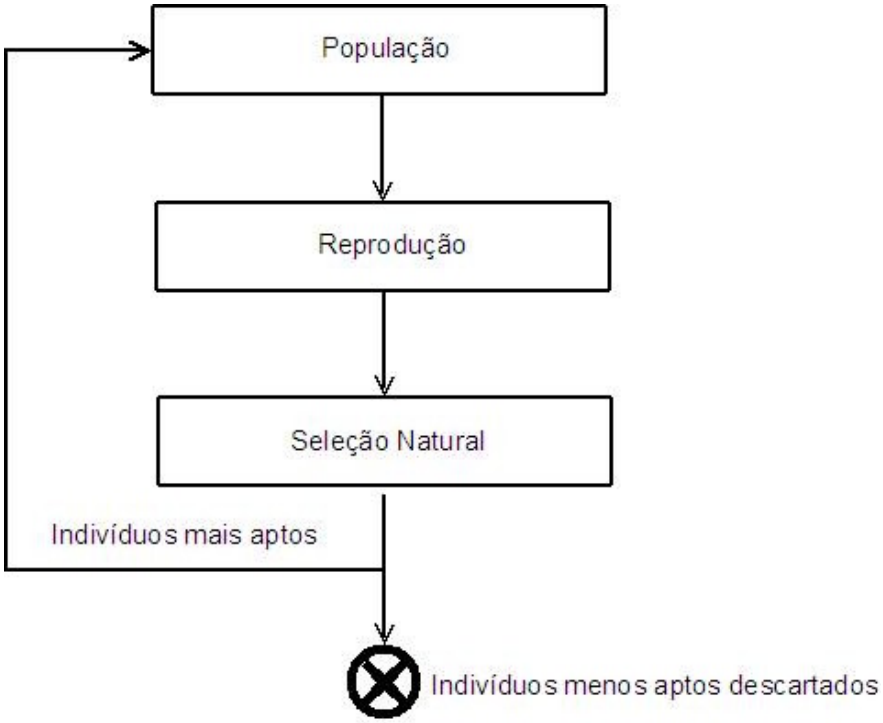


Fig 4 – Diagrama de um algoritmo genético

4 DESENVOLVIMENTO

O presente capítulo tem como objetivo apresentar um protótipo de um ambiente virtual usado para testes de agentes genéticos e outros sistemas inteligentes. O ambiente serve para avaliar a aplicabilidade dos testes envolvendo agentes implementados com algoritmos genéticos.

4.1 Subverse

O *subverse* é um sistema de agentes que interagem em um ambiente virtual criado para propósitos de estudos e implementações de algoritmos de Inteligência Artificial.

O *subverse* tem como objetivo criar de uma forma prática interações envolvendo computação gráfica, sistemas distribuídos (fazendo o uso da Internet como meio de comunicação entre o ambiente), inteligência artificial e jogos de computador.

O projeto encontra-se sob licença pública (GPL), disponibilizando seu código fonte para todos os interessados. Este foi criado pelo Prof. Msc. Saulo Popov Zambiasi, professor de Sistemas de Informação das Faculdades Barddal – Florianópolis/SC, juntamente com o auxílio de seus orientandos, que assumiram a responsabilidade de aplicar os algoritmos inteligentes no ambiente virtual. Atualmente o site oficial do projeto se encontra sob as dependências do *SourceForge*[6], site especializado em ceder espaço virtual para repositório de informações e projetos envolvendo software livre.

[6] Todas as versões estáveis encontram-se em https: //sourceforge.net/projects/subverse

4.2 Estrutura do *Subverse*

O ambiente do *Subverse* possui uma estrutura de simples entendimento para os interessados em desenvolver aplicações e agentes inteligentes.

Seu código foi todo escrito em C++ e segue a seguinte estrutura atualmente.

- Apenas um simples ambiente virtual foi implementado onde os agentes podem interagir com o mesmo.
- As classes dos agentes são classes filhas que derivam de uma classe principal chamada de *Element.*
- O sistema ainda não possui suporte a banco de dados, e a configuração do mundo virtual assume um arquivo contendo um mapa de construção usando matrizes (subverse.map).
- Novos agentes são facilmente implementáveis, podendo ser adicionados ao sistema a partir da criação de uma nova classe, desde que esta derive da classe principal, e a partir da introdução de suas posições iniciais no mapa do mundo virtual.
- O sistema ainda possui gráficos temporários tendo futuramente gráficos próprios.

4.3 Próximos Passos do Projeto *Subverse*

Os próximos passos envolvendo o Projeto *Subverse* serão:

- A criação de uma documentação completa de desenvolvimento e um manual do usuário;

- Implementação e Modelagem da base de dados do *subverse;*
- Implementação do Servidor do Mundo Virtual do *Subverse* e Sistemas de *Avatares*;
- Implementação da biblioteca de conexão dos agentes aos *avatares*;
- Implementação da Interface Gráfica Remota;
- Implementação da Interface Web;
- Implementação de exemplos de agentes.

A implementação do *subverse* envolvendo o tema deste trabalho está embasada com os fundamentos do tópico supracitado – Próximos passos do projeto *subverse* – e será envolvendo a construção de algoritmos genéticos nos agentes do ambiente virtual.

4.4 Arquitetura Atual

Hoje *subverse* possui um sistema simples que gerencia alguns agentes que interagem com o mundo. E está compreendido nos seguintes agentes:

- **Avatar:** Agente controlado pelo próprio usuário que faz uso do teclado para movimentar o personagem.
- **Ciclope:** Agente controlado por computador que possui características particulares iniciais como a habilidade de reconhecer um agente do tipo **Avatar** dentro do cenário virtual.
- **Spider:** agente controlado por computador que possui características particulares iniciais como a habilidade de reconhecer um agente do tipo **Ciclope** dentro do cenário virtual.

- **Elementos de Cenário:** Os elementos de cenário são todos os que compõem as características físicas do ambiente virtual. Tais elementos são compreendidos como árvores, fogueiras, água, terra, grama, paredes e ladrilhos. As características dos elementos de cenário são duas: Prover obstáculos de colisão aos agentes e também ser uma fonte de comida aos mesmos.

Uma característica em comum entre o agente ***Avatar*****, Ciclope, Árvore** e ***Spider***, são que os mesmos possuem respectivamente atributos como força, quantidade de vidas, fonte de comida e saúde. Tais atributos permitem que os agentes após o término de sua saúde, tornem-se também uma fonte de comida para os agentes ainda ativos no ambiente virtual.

Os objetos de estudo deste trabalho vão se ater aos **Ciclopes** e suas respectivas dependências. Será criada uma estrutura genética para os agentes do tipo **Ciclope** de modo que estes possuam uma interação maior com o ambiente conforme mostrado na figura 5.

Fig 5 – Ambiente *subverse* com seus devidos agentes.

4.5 Estrutura do agente Ciclope

O agente **Ciclope**, objeto de estudo deste trabalho, possui características genéticas que devem definir a forma de como o mesmo deve se comportar dentro do ambiente virtual. Cada **Ciclope,** por ser uma instância da classe principal *Element (*Classe na qual possui todas as características em comum entre todos os agentes, tais como força, saúde, fonte de alimento). Essas características são carregadas no ambiente virtual de forma independente, ou seja, é possível através disso criar características únicas para cada **Ciclope,** ficando assim mais evidente a interação dos mesmos tanto entre si quanto ao cenário a que ele pertence.

Tais características genéticas foram atribuídas a uma variável do tipo vetor aonde o mesmo faz a sua representação de seu DNA e pode ser visualizado a seguir:

```
dna[0] = getHelth();
dna[1] = getStrong();
dna[2] = getFood();
dna[3] = getLifeTime();
dna[4] = getMateEpoch();
dna[5] = getGeneration();
dna[6] = getSpiritState();
dna[7] = getHumanGrace();
dna[8] = getSpiderGrace();
dna[9] = getCiclopeGrace();
dna[10] = getSex();
```

Cada espaço definido no vetor – que possui como característica receber valores inteiros – é chamado de cromossomo, cada cromossomo é descrito abaixo com suas funcionalidades.

getHelth(): Esta função retorna ao DNA do **Ciclope** a quantidade de saúde que o mesmo possui, podendo variar entre 50 e 100. Quando este valor chegar a zero, o agente em questão será dado como morto.

getStrong(): Esta função é responsável por retornar a quantidade de força que cada agente possui e atribuir ao DNA. A quantidade de força é responsável por definir os valores de ataque que um determinado agente aplica sobre outro agente.

getFood(): Esta função retorna ao DNA do agente a quantidade de alimento que o mesmo pode oferecer aos demais agentes. Uma vez que o agente é dado como morto, este torna-se um alimento para os demais agentes vivos no ambiente virtual.

getLifeTime(): Esta função retorna o tempo de vida de cada agente e é estipulada entre 10 a 20 meses de tempo de simulação.

getMateEpoch(): Esta função retorna a época de acasalamento de cada agente variando de 1 a 12 – respectivamente aos meses correntes do ambiente virtual – e é definida para que possa haver a época de reprodução que irá gerar por sua vez novos agentes provenientes de dois indivíduos pais. Quando o ambiente encontra dois agentes que estão na mesma época de acasalamento, ambos efetuam a reprodução combinando os seus genes.

getGeneration(): Esta função tem como objetivo indicar a qual geração o agente pertence, tendo como premissa o valor 1. Caso um determinado agente, que possui em seu DNA o valor 1 como valor de geração, criar uma prole, esta prole receberá como valor de geração a seqüência da geração de seus pais, neste caso, o novo agente gerado possuirá o valor 2 como valor de geração e assim por diante.

getSpiritState(): Responsável por retornar o estado de espirito de cada agente podendo assumir o estado **Agressivo** ou **Passivo. (**Respectivamente 0 e 1).

getHumanGrace(): Responsável por retornar o nível de simpatia de um determinado agente referente a um agente do tipo ***Avatar*****. Agressivo** ou **Passivo. (**Respectivamente 0 e 1).

getSpiderGrace(): Responsável por retornar o nível de simpatia de um determinado agente referente a um agente do tipo ***Spider*****. Agressivo** ou **Passivo.** (Respectivamente 0 e 1).

getCiclopeGrace(): Responsável por retornar o nível de simpatia de um determinado agente referente a sua própria espécie, ou seja, **Ciclope. Agressivo** ou **Passivo.** (Respectivamente 0 e 1).

getSex(): Responsável por definir o sexo de cada agente, admitindo 0 para masculino e 1 para feminino. Dois agentes somente poderão efetuar reprodução se ambos os sexos forem opostos.

As funções **getHumanGrace()**, **getSpiderGrace()** e **getCiclopeGrace()** irão definir que tipos de ações o agente deverá tomar em caso de encontro com os respectivos agentes supracitados. Se um agente possui uma simpatia com um agente do tipo **Spider** por exemplo, o mesmo pode continuar circulando na área aonde se encontra tal agente, caso contrário, o agente pode efetuar um ataque ou defender-se afastando-se da localização, tudo isso definido através da função **getSpiritState().**

4.6 Definição inicial dos dados genéticos

Cada agente do tipo **Ciclope** que é introduzido ao sistema de simulação deve ser iniciado com valores aleatórios referente a todos os atributos supracitados no item 4.5.

Primeiramente o *subverse* realiza um cálculo randômico para definir cada atributo que irá ser introduzido ao DNA do agente. O mesmo segue a seguir:

- setCreature(true): Define se o agente faz parte do tipo criatura ou se faz parte do tipo ambiente. Neste caso, **Ciclope** faz parte do gênero criatura ou seja,

admite-se o valor ***true*** (verdadeiro) à função que armazena em uma variável específica.

- setMateEpoch(): Define a época de acasalamento do agente genético. Seu cálculo é realizado da seguinte forma:
  - `Época_de_Acasalamento <- Um número aleatório entre 1 e 12;`

  O valor é comparado à variável que contabiliza os **dias** correntes no ambiente.

- setLifeTime(): Define o tempo de vida de cada agente em relação ao tempo corrente do ambiente de simulação. O valor é comparado com a variável responsável por contabilizar os meses do ambiente. Seu algoritmo é tomado da seguinte forma:
  - `Tempo_de_vida <- Um número aleatório entre 10 e 20;`

- setGeneration(): Esta função simplesmente acrescenta 1 ao valor atual da geração. Este procedimento ocorre toda vez que um agente novo é gerado.
  - `Geração <- Geração + 1;`

- setAlive(true): Responsável por definir se o agente possui vida ou não. Caso o tempo de vida do agente tenha expirado, o mesmo recebe um sinal para deixar de agir no ambiente. A variável de controle admite ***true*** *(*verdadeiro*)* ou ***false*** (falso) respectivamente para indicar se o agente está vivo ou não. O algoritmo adotado é apresentado abaixo:

- ```
  Se Tempo_de_Vida_Atual = Tempo_de_Vida então
  setAlive <- false
  Fim se
  ```

- setSex(): Adota o sexo ao agente, admitindo 0 para masculino e 1 para feminino. A forma como é feita a adoção desta diretriz, é tomada de forma aleatória e seu algoritmo pode ser visto a seguir:

  - ```
    Numero_Auxilio <- Um número aleatório entre 0 e 1;
    Se Numero_Auxilio = a 0 então
    Sexo <- 0; //Masculino
    Senão Sexo <- 1; //Feminino
    Fim se
    ```

- setSpiritState(): Adota um valor para indicar se o agente assume um papel agressivo ou passivo e também é realizado a adoção de um número aleatório entre 0 e 1, respectivamente **agressivo** e **passivo**.
- setHumanGrace(): Adota um valor aleatório como supracitado variando entre 0 e 1 para definir o grau de simpatia que o determinado agente possui a Humanos, respectivamente **agressivo** e **passivo**.

- setCiclopeGrace(): Adota um valor aleatório entre 0 e 1 para indicar o grau de simpatia entre agentes o tipo **Ciclope,** respectivamente **agressivo** e **passivo.**

- setSpiderGrace(): Adota um valor aleatório entre 0 e 1 para indicar o grau de simpatia entre agentes do tipo ***Spider*****,** respectivamente **agressivo** e **passivo.**

4.7 Interação com os demais agentes

Os agentes do tipo **Ciclope** possuem uma interação com os demais agentes baseados em suas características genéticas. Quando um agente é introduzido ao ambiente, o mesmo recebe inicialmente em seu **DNA** todas as características como supracitadas no item **4.5**.

O agente atua efetuando uma série de comparações tanto envolvendo o próprio ambiente quanto envolvendo os agentes mais próximos. As comparações envolvem as características de seu próprio DNA e as características do agente mais próximo.

O agente pode decidir através de seu estado de espírito e o nível de simpatia com determinado agente próximo se o mesmo é um agente de caráter maléfico, tomando assim ações condizentes com o resultado obtido. O agente também pode definir fontes de comida e se as mesmas são fontes de comida abundante.

A seqüência de regras de interação entre os demais agentes é dada a seguir:

```
Se agente próximo = "criatura" então
Se   estado_de_espírito_do_agente_próximo   =   "agressivo"   e
"próprio_estado_de_espírito" = "agressivo" então
      Ataque agente próximo;
Senão Se estado_de_espírito próprio = "passivo" então
      Mova Aleatoriamente;
Fimse
Fimse
Se    estado_de_espírito_do_agente_próximo    =    "passivo"    e
estado_de_espírito_do_próprio_agente = "agressivo" então
      Ataque (agente_próximo_;

Senão  se  estado_de_espírito_do_agente_próximo  =  "passivo"  e
estado_de_espírito_do_próprio_agente = "passivo" então
      Mova Aleatoriamente;
Fimse
FimSe

Se  Simpatia_com_Humano  =  "agressiva"  e  agente_próximo  =
"Humano" então
      Ataque agente próximo;
Senão Mova Aleatoriamente;
FimSe
```

```
Senão se simpatia com
”Ciclope” = “agressiva” e agente próximo for Ciclope então
      Ataque (agente_próximo);
Senão Mova Aleatoriamente;

Senão Se simpatia com “Spider” = “agressiva” e agente próximo
for “Spider” então
      Ataque (Agente_Próximo);
Senão Mova Aleatoriamente;

Se agente próximo = “Árvore” e fonte de alimento do agente
próximo > 0 então
      Alimente-se de Agente Próximo;

Senão Se agente próximo for criatura e agente próximo não
estiver vivo e fonte e alimento do agente próximo for maior que
zero então
      Alimente-se de Agente Próximo;
```

Existem formas mais concretas de se definir interações entre agentes em um determinado ambiente que podem variar entre outras técnicas de inteligência artificial, algoritmos de busca simples com a combinação de máquinas de estados dentre outros. Foram adotadas rotinas reativas para os agentes, ou seja, técnicas de interação estritamente programáveis, de tal forma que não haja nenhum tipo de cognição para tomada de decisões mais complexas, fator que acarreta em perda de foco deste trabalho.

4.8 Reprodução entre dois agentes

Para que a reprodução entre dois agentes seja bem sucedida, essa necessita de três elementos importantes: a época de reprodução em ambos deve ser a mesma, o estado de espírito entre ambos deve ser o mesmo e um agente deve ser do sexo oposto ao outro.

No estudo de caso usado como parâmetro o ambiente de simulação *subverse*, é apresentado o algoritmo padrão para reprodução sem passar pelas premissas envolvendo a mutação e o índice de aptidão. O motivo dá-se ao fato de que o ambiente não possui uma

definição concreta de problema a ser solucionado e sim um ambiente livre para visualizar o comportamento de agentes interagindo aleatoriamente.

As características da reprodução entre dois agentes do tipo **Ciclope** é envolvendo a troca de genes entre seus cromossomos, o processo é realizado através de uma seleção aleatória entre os valores contidos no **DNA** de ambos e é dado a oportunidade de gerar um agente filho com as seguintes características: **Saúde, força, fonte de alimento, tempo de vida, época de acasalamento, estado de espírito, simpatia a humanos, simpatia a *spiders* e simpatia a ciclopes.**

O processo é definido a seguir:

```
Se agente próximo for Ciclope e sexo do agente próximo for
diferente do sexo do agente e estado de espírito do agente
próximo for igual ao estado de espírito do agente então
      Classe Elemento é igual a novo Ciclope
      Introduza novo Ciclope no cenário
      Novo_Dna[0] é igual a Aleatório entre dna[0] de agente
      próximo e Dna[0] de agente
      Novo_Dna[1] é igual a Aleatório entre dna[1] de agente
      próximo e Dna[1] de agente;
      Novo_Dna[2] é igual a Aleatório entre dna[2] de agente
      próximo e Dna[2] de agente;
      Novo_Dna[3] é igual a Aleatório entre dna[4] de agente
      próximo e Dna[4] de agente;
      Novo_Dna[6] é igual a Aleatório entre dna[6] de agente
      próximo e Dna[6] de agente;
      Novo_Dna[5] é igual a Geração atual mais 1;
      Novo_Dna[7] é igual a Aleatório entre dna[7] de agente
      próximo e Dna[7] de agente;
      Novo_Dna[8] é igual a Aleatório entre dna[8] de agente
      próximo e Dna[8] de agente;
      Novo_Dna[9] é igual a Aleatório entre dna[9] de agente
      próximo e Dna[9] de agente;
```

A seqüência do novo DNA é atribuída ao vetor de 10 posições já existente em cada instância da Classe *Element* e a sua seqüência de atributos são seguidas de acordo com as descrições feitas no item **4.5.**

A partir deste processo, o novo agente do tipo **Ciclope** herda as características definidas entre os dois agentes pais e o mesmo continua a interagir com o ambiente com a diferença de que ele faz parte da geração subseqüente à geração dos pais.

4.9 Eliminação dos agentes mais antigos

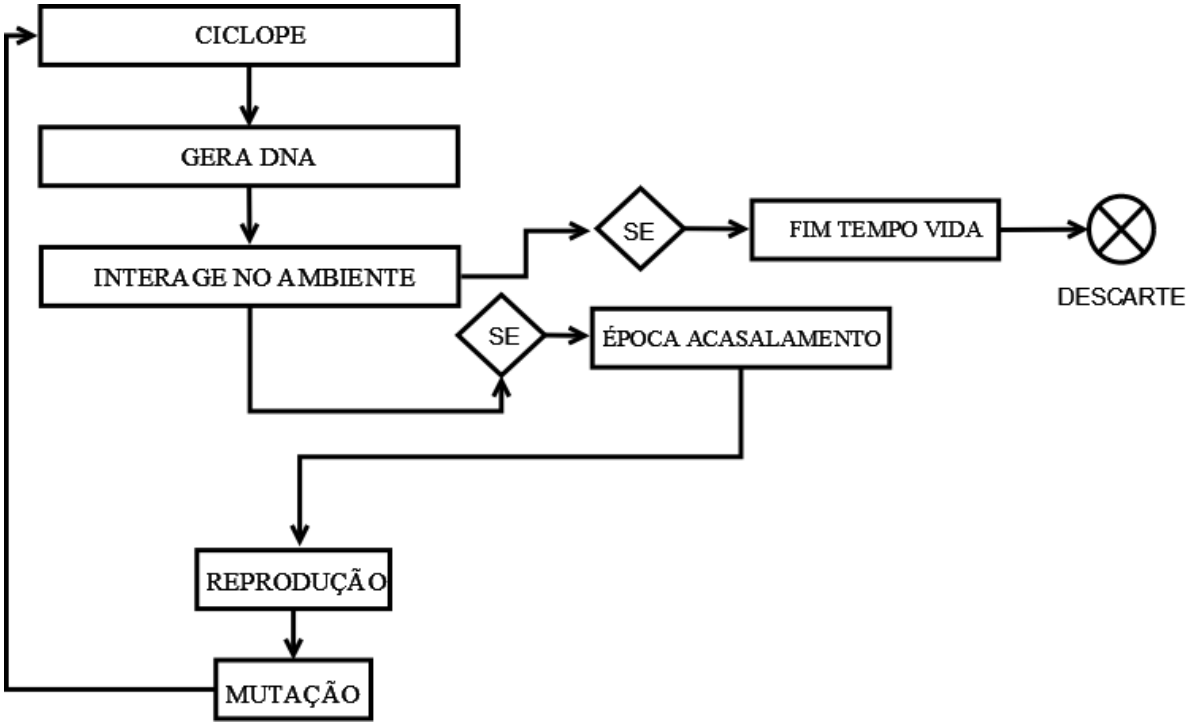
Como citado anteriormente, cada agente possui um tempo estimado de vida variando entre 10 e 20 meses correntes dentro do ambiente. Também existe outro fator que define o estado de vida do agente que é o seu nível de saúde, que pode ser modificado de acordo com os ataques vindos de outros agentes.

A seguir segue a seqüência de estados que define o descarte de um agente do ambiente:

```
Se tempo corrente do cenário for igual ao tempo de vida do agente
então
   Agente_Vivo <- Falso;
Senão Se saúde do agente atingir zero então
   Agente_Vivo <- Falso;
```

A partir do presente momento, se ocorrer, o ambiente gera um relatório contendo a data de óbito do agente, a sua geração pertencente bem como todos os atributos resultantes do exato momento em que o agente foi descartado do ambiente.

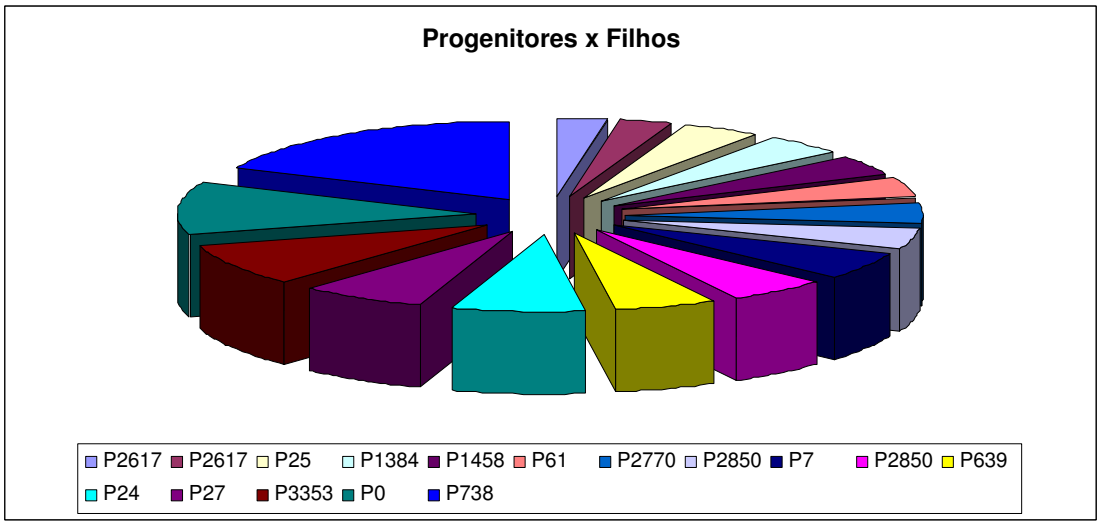
A figura a seguir mostra o funcionamento do agente **ciclope** através de um diagrama de fluxos, o mesmo está representado de uma forma sucinta de modo que todas as funcionalidades principais supracitadas sejam enfatizadas.



(Fig 6 – Diagrama de fluxo do comportamento do agente **ciclope**)

4.10 Gráfico demonstrativo

Baseado num relatório gerado após algumas horas de testes envolvendo o ambiente *subverse,* foi possível gerar um gráfico demonstrativo obtendo-se uma amostra a partir das 47 progenitoras bem como dos seus respectivos filhos. O gráfico gerado pode ser visualizado a seguir a partir de uma amostra de 16 progenitoras.



(Fig 7 – Gráfico gerado a partir do relatório textual criado pelo *subverse*)

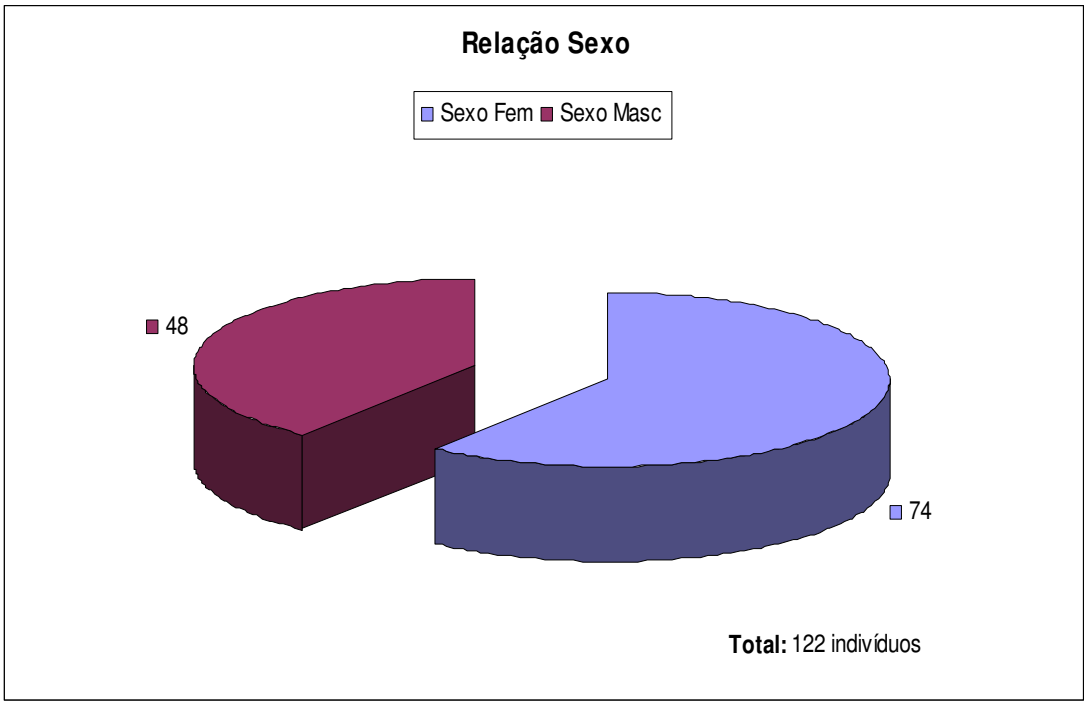
O gráfico acima mostra a relação de filhos gerados por cada progenitora (contabilizado apenas o sexo feminino). A Tabela a seguir demonstra com maiores detalhes a quantidade de filhos gerados pela pequena amostra usada.

Nome da Progenitora	Quantidade de Filhos
P2617	2
P2617	2
P25	3
P1384	3
P1458	3
P61	3
P2770	3
P2850	3
P7	4
P2850	4
P639	4
P24	5
P27	5
P3353	6
P0	8
P738	13

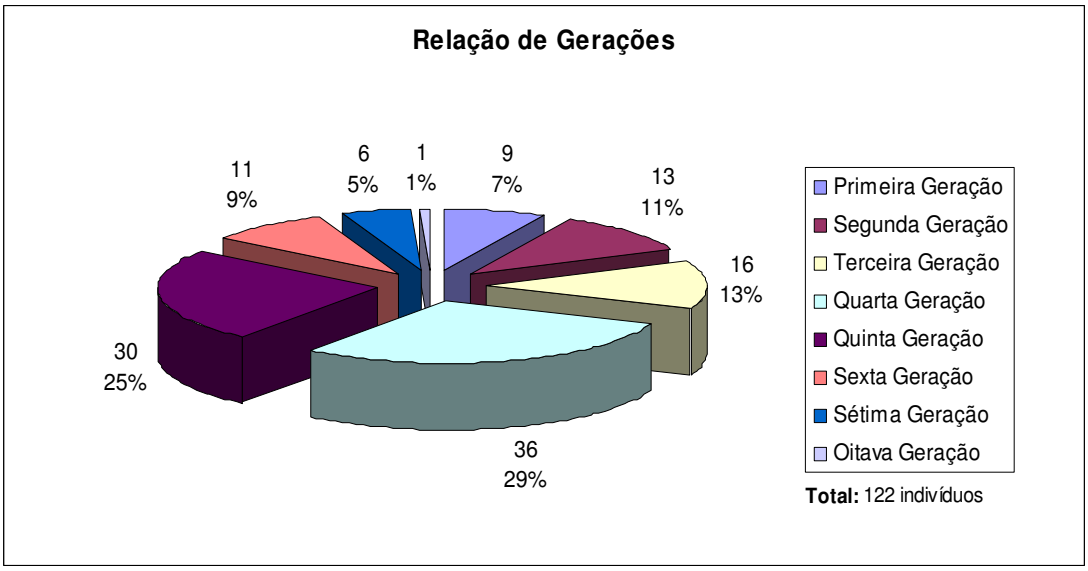
(Tabela 1 – Relação da quantidade de filhos para cada progenitora)

O relatório completo bem como os detalhes de atributos pode ser visualizado nos Anexos B e C deste Trabalho.

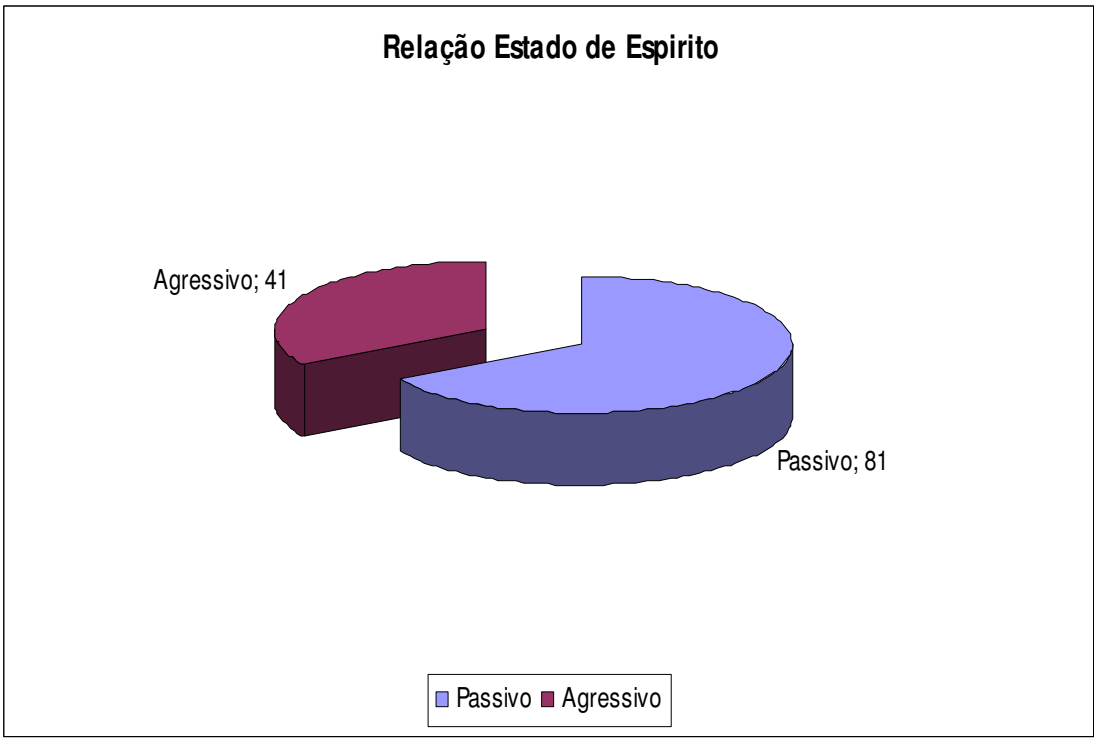
As figuras seguintes mostram a relação de sexos, gerações de cada agente e a relação de estado de espírito de cada agente, tomada como parâmetro 122 indivíduos, logo após a data de óbito dos mesmos.



(Fig 8 – Relação de sexo da amostra total dos agentes estudados)



(Fig 9 – Relação das gerações criadas a partir da amostra total de indivíduos estudados)



(Fig 10 – Relação de estados de espírito de cada agente após seu óbito)

4.11 Considerações sobre o *subverse*

O *subverse* demonstrou-se uma ferramenta extremamente útil para simular agentes inteligentes genéticos, mesmo sendo uma versão beta do projeto, é possível fazer grandes avanços através das classes muito bem documentadas, interface intuitiva e algoritmos bem estruturados de modo que o acadêmico possa efetuar chamadas de funções e objetos sem preocupar-se em criá-los antes mesmo de implementar a inteligência dos mesmos.

5 Considerações Finais

O presente trabalho apresentou um estudo prévio sobre jogos de computador e sua intersecção com a área da Inteligência Artificial, assim como também apresentou definições, termos e o funcionamento dos Algoritmos Genéticos, subárea da Inteligência Artificial.

Um ambiente virtual, chamado *subverse*, foi estudado e avaliado para a implementação dos agentes e do algoritmo genético. Este se tornou válido para elucidar as teorias envolvendo agentes genéticos. Ainda foi possível constatar que a presença de fatores genéticos e biológicos reais pode ser aplicada para gerar agentes melhores dispostos a fornecer maior interatividade em relação ao jogador em jogos eletrônicos.

Uma vez que os agentes, devidamente conceituados, possam ser introduzidos em jogos eletrônicos, geram possibilidades de aumentar tanto o valor do próprio jogo para a empresa que o cria quanto os valores quantitativos. Conseqüentemente, de formas a desenvolver a estratégia de jogo perante o próprio jogador que, desta vez, fica atenta a diversidade das possibilidades.

O mercado cresce exponencialmente juntamente com a evolução da tecnologia voltada a jogos, sendo que hoje, para criar um nível de atração entre o jogador e o jogo em si, deve ser considerado não somente os fatores gráficos, mas sim os de *gameplay* (jogabilidade). Este fator gera muita repercussão devido ao fato de ser um dos grandes paradigmas envolvendo a concorrência entre empresas responsáveis por desenvolver *cases* de sucesso a cada ano.

A inteligência artificial, principalmente a programação genética, pode proporcionar um dos aspectos que englobam uma boa jogabilidade inclusive tratando-se de jogos jogados em modo multijogador on-line, gênero responsável pela maior bilheteria nos últimos 5 anos.

Por fim, este trabalho apresentou um estudo e implementação da alternativa para o comportamento de agentes em jogos de computadores, controlados pelo computador, como forma de tornar o jogo mais atraente para o jogador.

5.1 Sugestões para Trabalhos Futuros

Como sugestões para trabalhos futuros, os seguintes itens podem ser analisados:

- Dar continuidade aos estudos de algoritmos genéticos em trabalhos futuros;
- Continuar o desenvolvimento com o Prof. Msc Saulo Popov Zambiasi do projeto *subverse*, ambiente de simulação ao qual pode ser uma grande ferramenta no futuro acadêmico de qualquer estudante da área;
- Alterações no sistema genético do *subverse* que permitam maiores funcionalidades como, por exemplo, objetivos específicos de cada agente a atuar no ambiente;
- Aplicar os conhecimentos adquiridos em inteligência artificial para criar uma engine (motor de desenvolvimento) sólida que auxilie as empresas desenvolvedoras de jogos eletrônicos no Brasil.

REFERÊNCIAS

KOZA, R. John. *GENETIC PROGRAMMING: A PARADIGM FOR GENETICALLY BREEDING POPULATIONS OF COMPUTER PROGRAMS TO SOLVE PROBLEMS*. ***Computer Science Department***, Stanford, jun. 1990.

MITCHEL, Melanie. **An Introduction to Genetic Algorithms** 5th ed. Massachusetts: MIT Press, 1999.

RUSSEL, Stuart & NORVIG Peter. **Inteligência Artificial** 2ª ed. RJ: Elsevier, 2004

DAWSON, Michael. **Beginning C++ Game Programming.** Ma: Thompson Course Technology, 2004.

BUCKLAND, Mat. **Ai Techniques for Game Programming.** Ohio: Premier Press, 2002.

BUCKLAND, Mat. **Programming AI by Example**. Texas: Wordware Publishing, Inc, 2005.

RABIN, Steve**. Ai Game Programming Wisdom. 1st ed.** Hingham: Charles River Media.

LARAMÉE, D. François. Genetic Algorithms: Evolving the Perfect Troll. **Ai Game Programming Wisdom.** Washington, v. 1, n. 1, p. 629-638, mar. 2002.

THOMÉ, Débora. **Onde está o emprego?: Games**. O Dia Online, nov. 2006 Disponível em: <http://odia.terra.com.br/educacao/htm/geral_68625.asp>. Acesso em: 28 mai. 2007.

Portal: Evolução. In. Wikipédia. Enciclopédia Digital. http://pt.wikipedia.org/wiki/Portal:Evolu%C3%A7%C3%A3o. Acesso em 08/11/2007.

PONG. In. Wikipédia, Enciclopédia Digital. http://en.wikipedia.org/wiki/Pong. Acesso em 01/06/2007.

PAC-MAN. In Wikipédia enciclopédia digital. http://en.wikipedia.org/wiki/Pac_man. Acesso em 11/06/2007.

BARRON, Todd **Strategy Game Programming with Directx® 9.0. 1st ed.** Texas: Wordware Publishing, inc, 2003.

YEE, Nicholas Avatars at **Work and Play: Collaboration and Interaction in Shared Virtual Environments**, editado por Schroder, R. & Axelsson, A., London: Springer-Verlag.

CHAMPANDARD, J Alex. **AI Game Development: Synthetic Creatures with Learning and Reactive Behaviors.** IN: New Riders Publishing, 2003.

ANEXO A – ALGORITMO GENÉTICO PADRÃO

```
função ALGORITMO-GENÉTICO(população, FN-FITNESS) retorna um indivíduo
  entradas: população, um conjunto de indivíduos
            FN-FITNESS, uma função que mede a adaptação de um indivíduo
  Repita
   nova_população <- conjunto vazio
   para i <- 1 até TAMANHO(população) faça
      x <- SELEÇÃO-ALEATÓRIA(população,FN_FITNESS)
      y <- SELEÇÃO-ALEATÓRIA(população,FN-FITNESS)
      filho <- REPRODUZ(x,y)
      se (pequena probabilidade aleatória) então filho <-
       MUTAÇÃO(filho)
           adicionar filho a nova_população
           população <- nova_população
  até algum indivíduo estar adaptado o suficiente ou até ter
   decorrido tempo suficiente
retornar o melhor indivíduo em população, de acordo com FN-FITNESS

função REPRODUZ(x,y) retorna um indivíduo
  entradas: x,y, indivíduos pais
  n <- COMPRIMENTO(x)
  c <- número aleatório de 1 a n
  retornar CONCATENA(SUBCADEIA(x,1,c), SUBCADEIA(y,c+1,n))
```

(Exemplo de algoritmo genético, Russel & Norvig, 2004, p. 117)

ANEXO B – RELATÓRIO DETALHADO - *SUBVERSE*

Número do Ciclope: 121
Número do Pai: 0
Número da Mãe: 79
Saúde: 58
Força: 16
Fonte de Alimento: 69
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 151
Número do Pai: 0
Número da Mãe: 78
Saúde: 70
Força: 11
Fonte de Alimento: 97
Ano de Morte: 9
Época de Acasalamento: 1
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 25
Número do Pai: 0
Número da Mãe: 7
Saúde: 90
Força: 12
Fonte de Alimento: 97
Ano de Morte: 17
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 13
Número do Pai: 0
Número da Mãe: 0

Saúde: 99
Força: 11
Fonte de Alimento: 84
Ano de Morte: 15
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 414
Número do Pai: 0
Número da Mãe: 50
Saúde: 78
Força: 10
Fonte de Alimento: 52
Ano de Morte: 10
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 78
Número do Pai: 0
Número da Mãe: 7
Saúde: 51
Força: 12
Fonte de Alimento: 60
Ano de Morte: 12
Época de Acasalamento: 0
Geração: 2
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 26
Número do Pai: 0
Número da Mãe: 10
Saúde: 77
Força: 14
Fonte de Alimento: 75
Ano de Morte: 11
Época de Acasalamento: 1

Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 61
Número do Pai: 0
Número da Mãe: 25
Saúde: 92
Força: 18
Fonte de Alimento: 98
Ano de Morte: 15
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 1195
Número do Pai: 0
Número da Mãe: 775
Saúde: 83
Força: 16
Fonte de Alimento: 65
Ano de Morte: 14
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 824
Número do Pai: 0
Número da Mãe: 13
Saúde: 85
Força: 17
Fonte de Alimento: 58
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim

Sexo: Masc

Número do Ciclope: 175
Número do Pai: 0
Número da Mãe: 121
Saúde: 70
Força: 19
Fonte de Alimento: 77
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 20
Número do Pai: 0
Número da Mãe: 7
Saúde: 66
Força: 14
Fonte de Alimento: 65
Ano de Morte: 9
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 5
Número do Pai: 0
Número da Mãe: 0
Saúde: 68
Força: 15
Fonte de Alimento: 67
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 259

Número do Pai: 0
Número da Mãe: 19
Saúde: 92
Força: 17
Fonte de Alimento: 78
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2545
Número do Pai: 0
Número da Mãe: 1384
Saúde: 65
Força: 14
Fonte de Alimento: 70
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 6
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2565
Número do Pai: 0
Número da Mãe: 2555
Saúde: 75
Força: 10
Fonte de Alimento: 64
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 7
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 2548
Número do Pai: 0
Número da Mãe: 1384
Saúde: 54
Força: 17
Fonte de Alimento: 97
Ano de Morte: 13

Época de Acasalamento: 0
Geração: 6
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 11
Número do Pai: 0
Número da Mãe: 0
Saúde: 97
Força: 15
Fonte de Alimento: 95
Ano de Morte: 10
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 1513
Número do Pai: 0
Número da Mãe: 27
Saúde: 67
Força: 18
Fonte de Alimento: 68
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2540
Número do Pai: 0
Número da Mãe: 1384
Saúde: 78
Força: 10
Fonte de Alimento: 74
Ano de Morte: 9
Época de Acasalamento: 1
Geração: 6
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim

Sexo: Fem

Número do Ciclope: 2644
Número do Pai: 0
Número da Mãe: 738
Saúde: 66
Força: 15
Fonte de Alimento: 90
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 2230
Número do Pai: 0
Número da Mãe: 1664
Saúde: 71
Força: 13
Fonte de Alimento: 73
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2730
Número do Pai: 0
Número da Mãe: 738
Saúde: 90
Força: 12
Fonte de Alimento: 90
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 2590
Número do Pai: 0

Número da Mãe: 1458
Saúde: 62
Força: 14
Fonte de Alimento: 75
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 1389
Número do Pai: 0
Número da Mãe: 1224
Saúde: 50
Força: 19
Fonte de Alimento: 59
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2367
Número do Pai: 0
Número da Mãe: 738
Saúde: 73
Força: 17
Fonte de Alimento: 76
Ano de Morte: 18
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 2977
Número do Pai: 0
Número da Mãe: 738
Saúde: 71
Força: 19
Fonte de Alimento: 66
Ano de Morte: 15
Época de Acasalamento: 1

Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 301
Número do Pai: 0
Número da Mãe: 50
Saúde: 84
Força: 14
Fonte de Alimento: 66
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 825
Número do Pai: 0
Número da Mãe: 27
Saúde: 90
Força: 12
Fonte de Alimento: 86
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 2631
Número do Pai: 0
Número da Mãe: 2520
Saúde: 95
Força: 15
Fonte de Alimento: 71
Ano de Morte: 13
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 1224
Número do Pai: 0
Número da Mãe: 4
Saúde: 61
Força: 12
Fonte de Alimento: 66
Ano de Morte: 16
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2583
Número do Pai: 0
Número da Mãe: 738
Saúde: 51
Força: 19
Fonte de Alimento: 70
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 523
Número do Pai: 0
Número da Mãe: 61
Saúde: 99
Força: 19
Fonte de Alimento: 63
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 523
Número do Pai: 0
Número da Mãe: 61

Saúde: 99
Força: 19
Fonte de Alimento: 63
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 3010
Número do Pai: 0
Número da Mãe: 2770
Saúde: 52
Força: 18
Fonte de Alimento: 71
Ano de Morte: 14
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 574
Número do Pai: 0
Número da Mãe: 27
Saúde: 74
Força: 14
Fonte de Alimento: 83
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 1264
Número do Pai: 0
Número da Mãe: 574
Saúde: 82
Força: 17
Fonte de Alimento: 88
Ano de Morte: 16
Época de Acasalamento: 0
Geração: 5

Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2985
Número do Pai: 0
Número da Mãe: 2850
Saúde: 68
Força: 14
Fonte de Alimento: 74
Ano de Morte: 14
Época de Acasalamento: 1
Geração: 6
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 77
Número do Pai: 0
Número da Mãe: 21
Saúde: 85
Força: 17
Fonte de Alimento: 65
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 1043
Número do Pai: 0
Número da Mãe: 11
Saúde: 73
Força: 11
Fonte de Alimento: 96
Ano de Morte: 16
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 132
Número do Pai: 0
Número da Mãe: 5
Saúde: 76
Força: 19
Fonte de Alimento: 57
Ano de Morte: 17
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 2990
Número do Pai: 0
Número da Mãe: 2770
Saúde: 67
Força: 16
Fonte de Alimento: 52
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 851
Número do Pai: 0
Número da Mãe: 826
Saúde: 75
Força: 12
Fonte de Alimento: 70
Ano de Morte: 17
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 2529
Número do Pai: 0
Número da Mãe: 738
Saúde: 75

Força: 18
Fonte de Alimento: 87
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 50
Número do Pai: 0
Número da Mãe: 25
Saúde: 88
Força: 13
Fonte de Alimento: 75
Ano de Morte: 16
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 23
Número do Pai: 0
Número da Mãe: 10
Saúde: 73
Força: 15
Fonte de Alimento: 68
Ano de Morte: 16
Época de Acasalamento: 0
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 2055
Número do Pai: 0
Número da Mãe: 1129
Saúde: 59
Força: 12
Fonte de Alimento: 88
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo

Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 3
Número do Pai: 0
Número da Mãe: 0
Saúde: 65
Força: 19
Fonte de Alimento: 56
Ano de Morte: 16
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 10
Número do Pai: 0
Número da Mãe: 0
Saúde: 76
Força: 10
Fonte de Alimento: 73
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 2505
Número do Pai: 0
Número da Mãe: 738
Saúde: 72
Força: 12
Fonte de Alimento: 58
Ano de Morte: 10
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2
Número do Pai: 0
Número da Mãe: 0
Saúde: 93
Força: 19
Fonte de Alimento: 86
Ano de Morte: 18
Época de Acasalamento: 1
Geração: 1
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 63
Número do Pai: 0
Número da Mãe: 7
Saúde: 67
Força: 11
Fonte de Alimento: 55
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3363
Número do Pai: 0
Número da Mãe: 3353
Saúde: 57
Força: 18
Fonte de Alimento: 63
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 2621
Número do Pai: 0
Número da Mãe: 2617
Saúde: 86
Força: 17

Fonte de Alimento: 52
Ano de Morte: 15
Época de Acasalamento: 0
Geração: 6
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 21
Número do Pai: 0
Número da Mãe: 6
Saúde: 90
Força: 18
Fonte de Alimento: 89
Ano de Morte: 17
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 1500
Número do Pai: 0
Número da Mãe: 27
Saúde: 61
Força: 15
Fonte de Alimento: 92
Ano de Morte: 17
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2622
Número do Pai: 0
Número da Mãe: 24
Saúde: 55
Força: 18
Fonte de Alimento: 96
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Agressivo
Gosta de Humanos: Não

Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 2579
Número do Pai: 0
Número da Mãe: 1138
Saúde: 83
Força: 19
Fonte de Alimento: 78
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 27
Número do Pai: 0
Número da Mãe: 25
Saúde: 78
Força: 16
Fonte de Alimento: 88
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2520
Número do Pai: 0
Número da Mãe: 738
Saúde: 95
Força: 14
Fonte de Alimento: 84
Ano de Morte: 15
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 639
Número do Pai: 0
Número da Mãe: 13
Saúde: 89
Força: 19
Fonte de Alimento: 92
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2793
Número do Pai: 0
Número da Mãe: 1381
Saúde: 67
Força: 15
Fonte de Alimento: 91
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 1129
Número do Pai: 0
Número da Mãe: 647
Saúde: 67
Força: 14
Fonte de Alimento: 55
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3349
Número do Pai: 0
Número da Mãe: 639
Saúde: 95
Força: 11
Fonte de Alimento: 85

Ano de Morte: 13
Época de Acasalamento: 1
Geração: 3
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3014
Número do Pai: 0
Número da Mãe: 2850
Saúde: 59
Força: 15
Fonte de Alimento: 80
Ano de Morte: 13
Época de Acasalamento: 1
Geração: 6
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2596
Número do Pai: 0
Número da Mãe: 738
Saúde: 91
Força: 12
Fonte de Alimento: 91
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 22
Número do Pai: 0
Número da Mãe: 8
Saúde: 50
Força: 17
Fonte de Alimento: 65
Ano de Morte: 13
Época de Acasalamento: 0
Geração: 2
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim

Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 454
Número do Pai: 0
Número da Mãe: 65
Saúde: 61
Força: 16
Fonte de Alimento: 94
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 454
Número do Pai: 0
Número da Mãe: 65
Saúde: 61
Força: 16
Fonte de Alimento: 94
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2539
Número do Pai: 0
Número da Mãe: 738
Saúde: 54
Força: 18
Fonte de Alimento: 89
Ano de Morte: 9
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 1140

Número do Pai: 0
Número da Mãe: 1129
Saúde: 66
Força: 16
Fonte de Alimento: 81
Ano de Morte: 9
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 2021
Número do Pai: 0
Número da Mãe: 451
Saúde: 52
Força: 17
Fonte de Alimento: 80
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 4124
Número do Pai: 0
Número da Mãe: 2617
Saúde: 95
Força: 18
Fonte de Alimento: 96
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 6
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 524
Número do Pai: 0
Número da Mãe: 523
Saúde: 63
Força: 10
Fonte de Alimento: 55
Ano de Morte: 10

Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 447
Número do Pai: 0
Número da Mãe: 61
Saúde: 68
Força: 15
Fonte de Alimento: 80
Ano de Morte: 13
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3509
Número do Pai: 0
Número da Mãe: 2624
Saúde: 83
Força: 12
Fonte de Alimento: 83
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3529
Número do Pai: 0
Número da Mãe: 2624
Saúde: 60
Força: 11
Fonte de Alimento: 77
Ano de Morte: 15
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não

Sexo: Fem

Número do Ciclope: 2958
Número do Pai: 0
Número da Mãe: 2770
Saúde: 94
Força: 10
Fonte de Alimento: 82
Ano de Morte: 14
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 3015
Número do Pai: 0
Número da Mãe: 2850
Saúde: 76
Força: 18
Fonte de Alimento: 54
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 6
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 1384
Número do Pai: 0
Número da Mãe: 825
Saúde: 53
Força: 19
Fonte de Alimento: 71
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 3350
Número do Pai: 0

Número da Mãe: 639
Saúde: 62
Força: 11
Fonte de Alimento: 64
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3023
Número do Pai: 0
Número da Mãe: 2850
Saúde: 88
Força: 16
Fonte de Alimento: 51
Ano de Morte: 18
Época de Acasalamento: 1
Geração: 6
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3354
Número do Pai: 0
Número da Mãe: 3353
Saúde: 83
Força: 15
Fonte de Alimento: 72
Ano de Morte: 14
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 1197
Número do Pai: 0
Número da Mãe: 775
Saúde: 91
Força: 16
Fonte de Alimento: 98
Ano de Morte: 15
Época de Acasalamento: 1

Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 1471
Número do Pai: 0
Número da Mãe: 738
Saúde: 98
Força: 18
Fonte de Alimento: 56
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 2586
Número do Pai: 0
Número da Mãe: 738
Saúde: 91
Força: 12
Fonte de Alimento: 79
Ano de Morte: 12
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3697
Número do Pai: 0
Número da Mãe: 3631
Saúde: 78
Força: 17
Fonte de Alimento: 92
Ano de Morte: 12
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2598
Número do Pai: 0
Número da Mãe: 2539
Saúde: 94
Força: 17
Fonte de Alimento: 70
Ano de Morte: 13
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3357
Número do Pai: 0
Número da Mãe: 3353
Saúde: 93
Força: 12
Fonte de Alimento: 65
Ano de Morte: 9
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3357
Número do Pai: 0
Número da Mãe: 3353
Saúde: 93
Força: 12
Fonte de Alimento: 65
Ano de Morte: 9
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 1511
Número do Pai: 0
Número da Mãe: 1197

Saúde: 52
Força: 14
Fonte de Alimento: 74
Ano de Morte: 14
Época de Acasalamento: 1
Geração: 6
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 17
Número do Pai: 0
Número da Mãe: 0
Saúde: 76
Força: 10
Fonte de Alimento: 50
Ano de Morte: 13
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3532
Número do Pai: 0
Número da Mãe: 3531
Saúde: 70
Força: 19
Fonte de Alimento: 70
Ano de Morte: 16
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 2569
Número do Pai: 0
Número da Mãe: 738
Saúde: 64
Força: 15
Fonte de Alimento: 63
Ano de Morte: 11
Época de Acasalamento: 1
Geração: 4

Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 3195
Número do Pai: 0
Número da Mãe: 1586
Saúde: 98
Força: 18
Fonte de Alimento: 82
Ano de Morte: 18
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3095
Número do Pai: 0
Número da Mãe: 1458
Saúde: 65
Força: 14
Fonte de Alimento: 65
Ano de Morte: 10
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 4389
Número do Pai: 0
Número da Mãe: 3161
Saúde: 74
Força: 11
Fonte de Alimento: 96
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 7
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 4409
Número do Pai: 0
Número da Mãe: 4408
Saúde: 75
Força: 14
Fonte de Alimento: 55
Ano de Morte: 9
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 432
Número do Pai: 0
Número da Mãe: 27
Saúde: 65
Força: 19
Fonte de Alimento: 93
Ano de Morte: 12
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 4313
Número do Pai: 0
Número da Mãe: 3511
Saúde: 64
Força: 10
Fonte de Alimento: 68
Ano de Morte: 13
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 775
Número do Pai: 0
Número da Mãe: 151
Saúde: 63

Força: 13
Fonte de Alimento: 97
Ano de Morte: 18
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 9
Número do Pai: 0
Número da Mãe: 0
Saúde: 88
Força: 19
Fonte de Alimento: 59
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 1
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3344
Número do Pai: 0
Número da Mãe: 9
Saúde: 83
Força: 19
Fonte de Alimento: 58
Ano de Morte: 13
Época de Acasalamento: 1
Geração: 2
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 176
Número do Pai: 0
Número da Mãe: 121
Saúde: 99
Força: 12
Fonte de Alimento: 77
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Agressivo

Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 4393
Número do Pai: 0
Número da Mãe: 4317
Saúde: 97
Força: 13
Fonte de Alimento: 79
Ano de Morte: 13
Época de Acasalamento: 0
Geração: 7
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 1381
Número do Pai: 0
Número da Mãe: 1040
Saúde: 51
Força: 17
Fonte de Alimento: 78
Ano de Morte: 10
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 4399
Número do Pai: 0
Número da Mãe: 4394
Saúde: 70
Força: 19
Fonte de Alimento: 79
Ano de Morte: 13
Época de Acasalamento: 1
Geração: 8
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 4394
Número do Pai: 0
Número da Mãe: 4317
Saúde: 99
Força: 18
Fonte de Alimento: 76
Ano de Morte: 9
Época de Acasalamento: 0
Geração: 7
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 3689
Número do Pai: 0
Número da Mãe: 3351
Saúde: 60
Força: 12
Fonte de Alimento: 79
Ano de Morte: 17
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Fem

Número do Ciclope: 3688
Número do Pai: 0
Número da Mãe: 3351
Saúde: 50
Força: 11
Fonte de Alimento: 65
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3348
Número do Pai: 0
Número da Mãe: 639
Saúde: 64
Força: 11

Fonte de Alimento: 63
Ano de Morte: 11
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 4316
Número do Pai: 0
Número da Mãe: 3353
Saúde: 91
Força: 17
Fonte de Alimento: 65
Ano de Morte: 10
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3112
Número do Pai: 0
Número da Mãe: 1458
Saúde: 67
Força: 15
Fonte de Alimento: 97
Ano de Morte: 10
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 4397
Número do Pai: 0
Número da Mãe: 4317
Saúde: 58
Força: 13
Fonte de Alimento: 94
Ano de Morte: 15
Época de Acasalamento: 0
Geração: 7
Estado de Espírito: Passivo
Gosta de Humanos: Não

Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 738
Número do Pai: 0
Número da Mãe: 451
Saúde: 60
Força: 14
Fonte de Alimento: 70
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 3
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Masc

Número do Ciclope: 4398
Número do Pai: 0
Número da Mãe: 4317
Saúde: 62
Força: 14
Fonte de Alimento: 68
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 7
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2619
Número do Pai: 0
Número da Mãe: 454
Saúde: 99
Força: 10
Fonte de Alimento: 94
Ano de Morte: 12
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Não
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3361
Número do Pai: 0
Número da Mãe: 3353
Saúde: 79
Força: 19
Fonte de Alimento: 59
Ano de Morte: 17
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 3631
Número do Pai: 0
Número da Mãe: 2624
Saúde: 98
Força: 13
Fonte de Alimento: 68
Ano de Morte: 10
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Não
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 4318
Número do Pai: 0
Número da Mãe: 3536
Saúde: 80
Força: 10
Fonte de Alimento: 85
Ano de Morte: 13
Época de Acasalamento: 0
Geração: 6
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 3346
Número do Pai: 0
Número da Mãe: 639
Saúde: 82
Força: 16
Fonte de Alimento: 56

Ano de Morte: 18
Época de Acasalamento: 1
Geração: 3
Estado de Espírito: Agressivo
Gosta de Humanos: Não
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 3511
Número do Pai: 0
Número da Mãe: 2624
Saúde: 89
Força: 13
Fonte de Alimento: 64
Ano de Morte: 15
Época de Acasalamento: 0
Geração: 4
Estado de Espírito: Passivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Sim
Sexo: Masc

Número do Ciclope: 4280
Número do Pai: 0
Número da Mãe: 454
Saúde: 95
Força: 18
Fonte de Alimento: 52
Ano de Morte: 12
Época de Acasalamento: 1
Geração: 5
Estado de Espírito: Agressivo
Gosta de Humanos: Sim
Gosta de Spider: Sim
Gosta de Ciclope: Não
Sexo: Fem

Número do Ciclope: 2855
Número do Pai: 0
Número da Mãe: 2586
Saúde: 75
Força: 19
Fonte de Alimento: 82
Ano de Morte: 14
Época de Acasalamento: 0
Geração: 5
Estado de Espírito: Passivo
Gosta de Humanos: Não
Gosta de Spider: Sim

Gosta de Ciclope: Não
Sexo: Fem

ANEXO C – PRINCIPAIS CÓDIGOS FONTES USADOS

C.1 – Classe *Element*

[...]

```
    void setMutationRate();
    int  getMutationRate();
    void setCrossGeneration(int);
    void setCrossHumanGrace(int,int);
     void setCrossSpiritState(int,int);
     void setCrossCiclopeGrace(int,int);
     void setCrossSpiderGrace(int,int);
     void setCrossFood(int,int);
     void setCrossHealth(int,int);
     void setCrossStregth(int,int);
     void setCrossLifeTime(int,int);
     void setCrossMateEpoch(int,int);

    int getCrossHumanGrace();
     int getCrossSpiritState();
     int getCrossCiclopeGrace();
     int getCrossSpiderGrace();
     int getCrossFood();
     int getCrossHealth();
     int getCrossStregth();
     int getCrossLifeTime();
     int getCrossMateEpoch();
    void setmae(int mae);
    void setpai(int pai);
    int getmae();
    int getpai();
    void SetAnoMorte(int anomorte );
     void setMateEpoch();
     int getMateEpoch();
     void setLifeTime();
     int getLifeTime();
     void setGeneration(int x);
     int getGeneration();
    int getTempoVida();
     void setSpiritState();
     int getSpiritState();
    void SetPrintAno();
    string GetPrintAno();
     void setHumanGrace();
     int getHumanGrace();
    void setSpiderGrace();
    int getSpiderGrace();
    void setCiclopeGrace();
    int getCiclopeGrace();
    void setSex();
    int getSex();
    int getMonthEpoch();
```

[...]

C.2 – *Element.Cpp*

[...]

```
void Element::setpai(int x) {

pai = x;
}

int Element::getpai() {
return pai;
}

void Element::setmae(int y) {
mae = y;
}

int Element::getmae() {
return mae;
}

void Element::setCrossMateEpoch(int x, int y) { //Cruza a época de acasalamento
    int alelo;
    alelo = (rand() % 10) + 1;

   if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     MateEpoch = x;
    }else {MateEpoch = y;}

    } else {
      MateEpoch = (rand() % 12) + 1;
    }
  }


int Element::getCrossMateEpoch() { // Retorna a época de acasalamento cruzada
return MateEpoch;
}



void Element::setCrossLifeTime(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

    if (MutationRate == 0) {

    if (alelo > 0 && alelo < 5) {
     LifeTime = x;
    }else {LifeTime = y;}

    } else  {
      LifeTime = (rand() % 10) + 10;
    }
}

int Element::getCrossLifeTime() {
return LifeTime;
}

void Element::setCrossStregth(int x, int y) { //Cruza força
    int alelo;
    alelo = (rand() % 10) + 1;


   if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     strong = x;
    }else {strong = y;}
   }else {

     strong = (rand() % 10) + 10;
   }
```

```
    }


int Element::getCrossStregth() { // Retorna a força cruzada
return strong;
}

void Element::setCrossHealth(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

   if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     helth = x;
    }else {helth = y;}
   }else {helth = (rand() % 50) + 50;}


}

int Element::getCrossHealth() {
return helth;
}

void Element::setCrossFood(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

  if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     food = x;
    }else {food = y;}

  }else { food = (rand() % 50) + 50; }

}

int Element::getCrossFood() {
return food;
}

void Element::setCrossSpiritState(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

   if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     SpiritState = x;
    }else {SpiritState = y;}
  }else {

      SpiritState = (rand() % 10) + 1;
      if (SpiritState > 0 && SpiritState < 5) {
       SpiritState = 0;
      }else {SpiritState = 1; }

  }

}

int Element::getCrossSpiritState() {
return SpiritState;
}
void Element::setCrossCiclopeGrace(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

    if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     CiclopeGrace = x;
    }else {CiclopeGrace = y;}
  }else {
```

```
      CiclopeGrace = (rand() % 10) + 1;
      if (CiclopeGrace > 0 && CiclopeGrace < 5) {
       CiclopeGrace = 0;
      }else {CiclopeGrace = 1; }

  }
}

int Element::getCrossCiclopeGrace() {
return CiclopeGrace;
}
void Element::setCrossSpiderGrace(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

    if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     SpiderGrace = x;
    }else {SpiderGrace = y;}
  }else {

      SpiderGrace = (rand() % 10) + 1;
      if (SpiderGrace > 0 && SpiderGrace < 5) {
       SpiderGrace = 0;
      }else {SpiderGrace = 1; }

  }
}

int Element::getCrossSpiderGrace() {
return SpiderGrace;
}
void Element::setCrossHumanGrace(int x, int y) {
    int alelo;
    alelo = (rand() % 10) + 1;

    if (MutationRate == 0) {
    if (alelo > 0 && alelo < 5) {
     HumanGrace = x;
    }else {HumanGrace = y;}
  }else {

      HumanGrace = (rand() % 10) + 1;
      if (HumanGrace > 0 && HumanGrace < 5) {
       HumanGrace = 0;
      }else {HumanGrace = 1; }

  }
}

int Element::getCrossHumanGrace() {
return HumanGrace;
}

void Element::setCrossGeneration(int x) {
Generation = x + 1;}
void Element::setMutationRate() {
int aux = (rand() % 200) + 1;
    if (aux > 50 && aux < 60) {
     MutationRate = 1;
    } else {MutationRate = 0;}
}

int Element::getMutationRate() {
return MutationRate;
}

void Element::setID(long id) {
    ID = id;
}
```

```
long Element::getID() {
    return ID;
}


int Element::getMonthEpoch() { //Retorna o mês corrente
return CounterMateEpoch;
}
int Element::getTempoVida() { //Retorna o tempo de vida
return CounterTempoVida;
}
void Element::SetAnoMorte(int anomorte) { // Retorna o ano de sua morte
AnoMorte = anomorte;
}
void Element::setGeneration(int x) { // Cria a geração
Generation = x + 1;
}
int Element::getGeneration() { // Retorna a Geração
return Generation;
}
void Element::setMateEpoch() { //Cria a época de acasalamento
MateEpoch = (rand() % 10) + 1;
if (MateEpoch > 0 && MateEpoch < 5) {
MateEpoch = 0;
}else { MateEpoch = 1; }
}

int Element::getMateEpoch() { // Retorna a época de acasalamento
return MateEpoch;
}
void Element::setLifeTime() {
LifeTime = (rand() % 10) + 10; //Vida entre 10 a 20 Anos.
}
int Element::getLifeTime() {
return LifeTime;
}
void Element::setSpiritState() { //0 para agressivo e 1 para passivo
SpiritState = (rand() % 10) + 1;
    if (SpiritState > 0 && SpiritState < 5) {
    SpiritState = 0;
    }else { SpiritState = 1;}
}
int Element::getSpiritState() {
return SpiritState;
}
void Element::setHumanGrace() { //0 para agressivo e 1 para passivo
HumanGrace = (rand() % 10) + 1;
    if (HumanGrace > 0 && HumanGrace < 5) {
    HumanGrace = 0;
    }else { HumanGrace = 1;}
}

int Element::getHumanGrace() {
return HumanGrace;
}
void Element::setSpiderGrace() { //0 para agressivo e 1 para passivo
SpiderGrace = (rand() % 10) + 1;
    if (SpiderGrace > 0 && SpiderGrace < 5) {
      SpiderGrace = 0;
    }else { SpiderGrace = 1;}
}
int Element::getSpiderGrace() {
return SpiderGrace;
}

void Element::setCiclopeGrace() { //0 para agressivo e 1 para passivo
CiclopeGrace = (rand() % 10) + 1;
    if (CiclopeGrace > 0 && CiclopeGrace < 5) {
    CiclopeGrace = 0;
    }else { CiclopeGrace = 1;}
}
int Element::getCiclopeGrace() {
```

```
return CiclopeGrace;
}

void Element::setSex() {
   Sex = (rand() % 10) + 1;
    if (Sex > 0 && Sex < 7) {
    Sex = 0;
    }else { Sex = 1;}
}
int Element::getSex() {
return Sex;
}
string Element::toString() {
string info = getName();
info += ": (V): "; //Vida
info += intToString(live);
info += " (F): "; //Força
info += intToString(strong);
info += " (S): "; //Saúde
info += intToString(helth);
info += " (C): "; //Fonte de Alimento
info += intToString(food);
info += " (EA): "; //Época de Acasalamento
info += intToString(MateEpoch);
info += " (G): "; //Geração
info += intToString(Generation);
info += " (TV): "; //Tempo de Vida (em anos)
info += intToString(AnoMorte);
info += " (SST): "; //Spirit State
info += intToString(SpiritState);
info += " (HG): "; //Human Grace
info += intToString(HumanGrace);
info += " (CG): "; //Ciclope Grace
info += intToString(CiclopeGrace);
info += " (SG): "; //Spider Grace
info += intToString(SpiderGrace);
info += " (VR): ";
info += intToString(visionRange);
info += " (SX): ";
info += intToString(Sex);
info += " (ID): ";
info += intToString(getID());
return info;

void Element::SetPrintAno() {
info2 = "ANO:";
info2 += intToString(world->getTimer()->getYear());
info2 += " Mes:";
info2 += intToString(world->getTimer()->getMonth());
info2 += " Dia: ";
info2 += intToString(world->getTimer()->getDay());
info2 += " Hora: ";
info2 += intToString(world->getTimer()->getHour());
info2 += " Minuto: ";
info2 += intToString(world->getTimer()->getMinute());
info2 += " Segundo: ";
info2 += intToString(world->getTimer()->getSecound());
}
string Element::GetPrintAno() {
return info2;
}
```
[...]

C.3 – Classe Cyclope

```
#ifndef CYCLOPE_H_
#define CYCLOPE_H_
#include <string>
#include "Config.h"
#include "Element.h"
```

```
class Cyclope : public Element {
private:
    int dna[13];
    int TempoVida;
    int MmateEpoch;
public:
    Cyclope(World*);
   virtual void execute();
    virtual ~Cyclope();
};

#endif /*CYCLOPE_H_*/
```

C.4 – Cyclope.Cpp

```
#include "Cyclope.h"
#include "World.h"
#include <stdlib.h>

Cyclope::Cyclope(World* w) :
    Element(w, "cyclope", CYCLOPE) {
    setCreature(true); // Ciclope é uma criatura
    setMateEpoch(); // Época de Acasalamento
    setLifeTime(); // Tempo de Vida
    setAlive(true); //true = vivo e false = morto
    setSex(); // 0 masculino e 1 feminino
    setSpiritState(); // 0 para agressivo e 1 para passivo
    setHumanGrace(); // 0 para agressivo e 1 para passivo
    setCiclopeGrace(); // 0 para agressivo e 1 para passivo
    setSpiderGrace(); // 0 para agressivo e 1 para passivo
    setID( getWorld()->getID() ); // seta seu proprio ID (SAULO)
    // setar a carga genética;

    dna[0] = getHelth(); // Saúde
    dna[1] = getStrong(); // Força
    dna[2] = getFood(); // Fonte de Alimento
    if(getTempoVida() > getLifeTime()) { // Tempo de Vida
       dna[3] = getTempoVida() - getLifeTime();

     }else if (getTempoVida() < getLifeTime()) {
       dna[3] = getLifeTime() - getTempoVida();

     }

    dna[4] = getMateEpoch(); // Época de Acasalamento
    dna[5] = getGeneration(); // Geração
    dna[6] = getSpiritState(); // Estado de Espírito
    dna[7] = getHumanGrace(); // Gosta de humano ?
    dna[8] = getSpiderGrace(); // Gosta de Aranha ?
    dna[9] = getCiclopeGrace(); // Gosta de Ciclope ?
    dna[10] = getSex(); // Sexo do Ciclope
    dna[11] = getpai();
    dna[12] = getmae();

    SetAnoMorte(dna[3]); }

void Cyclope::execute() {
    bool found = false;
    Vision *vision = getVision();
    Element* e=  NULL;
    Element* f = NULL;


    // Procura por um objetivo

    do {
     TempoVida = getTempoVida();
     MmateEpoch = getMonthEpoch();

         e = vision->getNext(ELEMENT);
```

```
                    if (e != NULL) {
                            if (getHealth() < 20) {

                                    if ((e->getType() == TREE) && (e->getFood() > 0)) {
                                            found = true;
                                    } else if ((e->isCreature()) && (!e->isAlive())
                                                && (e->getFood() > 0)) {
                                            found = true;
                                    }
                            } else if (isUnderAttack() && (e == getAttacker()) && e-
>isAlive()) {
                                    found = true;
                            } else if ((e->getType() == SPIDER) && e->isCreature()
                                            && e->isAlive()) {
                                    found = true;
                            } else if ((e->getType() == CYCLOPE) && e->isCreature()
                                            && e->isAlive()) {
                                    found = true;
                            }

                            }


                    if (TempoVida == dna[3]){setAlive(false);} //Mata o agente
                if (getHelth() == 0 || getHelth() < 0) {setAlive(false);}

                     if (!isAlive()) {


                      FILE *file = fopen("relatorio.txt","a+"); //- w:write
                      fprintf(file,"Número do Ciclope: ");
                      fprintf(file,intToString(getID()));
                      fprintf(file,"\n");

                      fprintf(file,"Número do Pai: ");
                      fprintf(file,intToString(getpai()));
                      fprintf(file,"\n");

                      fprintf(file,"Número da Mãe: ");
                      fprintf(file,intToString(getmae()));
                      fprintf(file,"\n");

                      fprintf(file,"Saúde: ");
                      fprintf(file,intToString(dna[0]));
                      fprintf(file,"\n");

                      fprintf(file,"Força: ");
                      fprintf(file,intToString(dna[1]));
                      fprintf(file,"\n");

                      fprintf(file,"Fonte de Alimento: ");
                      fprintf(file,intToString(dna[2]));
                      fprintf(file,"\n");

                      fprintf(file,"Ano de Morte: ");
                      fprintf(file,intToString(dna[3]));
                      fprintf(file,"\n");

                      fprintf(file,"Época de Acasalamento: ");
                      fprintf(file,intToString(dna[4]));
                      fprintf(file,"\n");

                      fprintf(file,"Geração: ");
                      fprintf(file,intToString(getGeneration()));
                      fprintf(file,"\n");

                      fprintf(file,"Estado de Espírito: ");
                      if (dna[6] == 0) {
                      fprintf(file,"Agressivo");
```

```
                        }else { fprintf(file,"Passivo"); }
                        fprintf(file,"\n");

                        fprintf(file,"Gosta de Humanos: ");
                        if (dna[7] == 0) {
                        fprintf(file,"Sim");
                        }else { fprintf(file,"Não"); }
                        fprintf(file,"\n");

                        fprintf(file,"Gosta de Spider: ");
                        if (dna[8] == 0) {
                        fprintf(file,"Sim");
                        }else { fprintf(file,"Não"); }
                        fprintf(file,"\n");

                        fprintf(file,"Gosta de Ciclope: ");
                        if (dna[9] == 0) {
                        fprintf(file,"Sim");
                        }else { fprintf(file,"Não"); }
                        fprintf(file,"\n");

                        fprintf(file,"Sexo: ");
                        if (dna[10] == 0) {
                        fprintf(file,"Fem");
                        }else { fprintf(file,"Masc"); }
                        fprintf(file,"\n");
                        fprintf(file,"Filho de: ");
                        fprintf(file,intToString(dna[12]));
                        fprintf(file,"\n");
                        fprintf(file,"-----------------------------------------------------------
---\n");
                        fprintf(file,"-----------------------------------------------------------
---\n");
                        fclose(file);


                    }
                } while ((!found) && (e != NULL));

                if (found) {

                 Element* front = getAround(ELEMENT, FRONT);

            /*(Gera um novo ciclope com uma geração acima da geração do progenitor)*/
              //Se a época corrente for igual época de acasalamento do cromossomo.
             if ((front != NULL) && (e == front)) {


                //Se o elemento estiver na sua frente
                // Se o elemento da frente for CYCLOPE e diferente de sexo diferente

                 if ((e->getType() == CYCLOPE) && (e->isCreature()) && (e->getMateEpoch() ==
dna[4]) && e->getSex() != dna[10]) {
                  // Se houver espaço vazio nas redondezas
                   if ((getWorld()->get(ELEMENT, getX()+2, getY()+1) == NULL)){
                        //Cria um novo agente
                        Element* f = new Cyclope(getWorld());
                        getWorld()->set(ELEMENT, getX()+2, getY()+1, f);


                        f->setCrossHealth(dna[0],e->getHelth());
                        f->setCrossStregth(dna[1],e->getStrong());
                        f->setCrossFood(dna[2],e->getFood());
                        f->setCrossLifeTime(dna[3],e->getLifeTime());
                        f->setCrossMateEpoch(dna[4],e->getMateEpoch());
                        f->setCrossSpiritState(dna[6],e->getSpiritState());
                        f->setCrossSpiderGrace(dna[7],e->getHumanGrace());
                        f->setCrossSpiderGrace(dna[8],e->getSpiderGrace());
                        f->setCrossCiclopeGrace(dna[9],e->getCiclopeGrace());

                        if (dna[12] == 0) {
                      f->setGeneration( getGeneration() );
```

```
      }else {f->setGeneration(e->getGeneration());}


      if (dna[12] == 0) {
    f->setmae(getID());
      }else{ f->setmae(e->getID());}

              int dir = getDirection();
      if (!move(dir)) {
              switch (dir) {
              case UP:
                      setDirection(LEFT);
                      break;
              case LEFT:
                      setDirection(DOWN);
                      break;
              case DOWN:
                      setDirection(RIGHT);
                      break;
              case RIGHT:
                      setDirection(UP);
                      break;
              }
      }


 } else if ((getWorld()->get(ELEMENT, getX(), getY()-1) == NULL)){
      //Cria um novo agente
      Element* f = new Cyclope(getWorld());
      getWorld()->set(ELEMENT, getX(), getY()-1, f);


      f->setCrossHealth(dna[0],e->getHelth());
      f->setCrossStregth(dna[1],e->getStrong());
      f->setCrossFood(dna[2],e->getFood());
      f->setCrossLifeTime(dna[3],e->getLifeTime());
      f->setCrossMateEpoch(dna[4],e->getMateEpoch());
      f->setCrossSpiritState(dna[6],e->getSpiritState());
      f->setCrossSpiderGrace(dna[7],e->getHumanGrace());
      f->setCrossSpiderGrace(dna[8],e->getSpiderGrace());
      f->setCrossCiclopeGrace(dna[9],e->getCiclopeGrace());

   if (dna[12] == 0) {
    f->setGeneration( getGeneration() );
      }else {f->setGeneration(e->getGeneration());}


   if (dna[12] == 0) {
    f->setmae(getID());
      }else {f->setmae(e->getID());}

              int dir = getDirection();
      if (!move(dir)) {
              switch (dir) {
              case UP:
                      setDirection(LEFT);
                      break;
              case LEFT:
                      setDirection(DOWN);
                      break;
              case DOWN:
                      setDirection(RIGHT);
                      break;
              case RIGHT:
                      setDirection(UP);
                      break;
              }
      }


 }
}
```

```
}
  if ((front != NULL) && (e == front)) {
        if (e->isCreature()) {
                  if(e->getSpiritState() == 0) {
               if(dna[6] == 0) {
                attack(e);
               }else if (dna[6] == 1) {
        int dir = getDirection();
        if (!move(dir)) {
                switch (dir) {
                case UP:
                        setDirection(LEFT);
                        break;
                case LEFT:
                        setDirection(DOWN);
                        break;
                case DOWN:
                        setDirection(RIGHT);
                        break;
                case RIGHT:
                        setDirection(UP);
                        break;
                }
        }
                }
                   } else if(e->getSpiritState() == 1) {
                      if (dna[6] == 0) {
                      attack(e);
                      } else if (dna[6] == 1) {
                                  int dir = getDirection();
        if (!move(dir)) {
                switch (dir) {
                case UP:
                        setDirection(LEFT);
                        break;
                case LEFT:
                        setDirection(DOWN);
                        break;
                case DOWN:
                        setDirection(RIGHT);
                        break;
                case RIGHT:
                        setDirection(UP);
                        break;
                }
        }
                         }
             }
                   if(dna[7] == 0 && e->getType() == AVATAR) {
                    attack(e);
                   } else if (dna[8] == 0 && e->getType() == SPIDER) {
                     attack(e);
               } else if (dna[9] == 0 && e->getType() == CYCLOPE) {
                 attack(e);
               }

               if(dna[7] == 1 && isUnderAttack() && dna[6] == 1 && e ==
getAttacker()) {
                   int dir = getDirection();
        if (!move(dir)) {
                switch (dir) {
                case UP:
                        setDirection(LEFT);
                        break;
                case LEFT:
                        setDirection(DOWN);
                        break;
                case DOWN:
                        setDirection(RIGHT);
                        break;
                case RIGHT:
```

```
                                setDirection(UP);
                                break;
                        }
                }
                    }else if(dna[8] == 1 && isUnderAttack() && dna[6] == 1 && e ==
getAttacker()) {
                        int dir = getDirection();
                if (!move(dir)) {
                        switch (dir) {
                        case UP:
                                setDirection(LEFT);
                                break;
                        case LEFT:
                                setDirection(DOWN);
                                break;
                        case DOWN:
                                setDirection(RIGHT);
                                break;
                        case RIGHT:
                                setDirection(UP);
                                break;
                        }
                }
                    }else if(dna[9] == 1 && isUnderAttack() && dna[6] == 1 && e ==
getAttacker()) {
                        int dir = getDirection();
                if (!move(dir)) {
                        switch (dir) {
                        case UP:
                                setDirection(LEFT);
                                break;
                        case LEFT:
                                setDirection(DOWN);
                                break;
                        case DOWN:
                                setDirection(RIGHT);
                                break;
                        case RIGHT:
                                setDirection(UP);
                                break;
                        }
                }
                    }

                      }

                        if ((e->getType() == TREE) && (e->getFood() > 0)) {
                                eat(e);
                        } else if ((e->isCreature()) && (!e->isAlive()) && (e->getFood()
> 0)) {
                                eat(e);
                        } else

                 if (!move(getDirection())) {
                        switch (getDirection()) {
                        case UP:
                                setDirection(LEFT);
                                break;
                        case LEFT:
                                setDirection(DOWN);
                                break;
                        case DOWN:
                                setDirection(RIGHT);
                                break;
                        case RIGHT:
                                setDirection(UP);
                                break;
                        }
                }
                     } else {
                         if (e->getX() > getX())
                          move(RIGHT);
```

```
                else if (e->getX() < getX())
                        move(LEFT);
                else if (e->getY() > getY())
                        move(DOWN);
                else if (e->getY() < getY())
                        move(UP);
                }
        } else {
                // Se nao tem objetivo, anda aleatoriamente

                move((rand()%4)*2);

                }
}
Cyclope::~Cyclope() {
}
```