UNIVERSIDADE DO SAGRADO CORAÇÃO

ANDRÉ LUIZ PILASTRI

Chatterbot com interatividade ao Avatar encapsulado no ambiente virtual Second Life usando a base de conhecimento em AIML

BAURU
2008

UNIVERSIDADE DO SAGRADO CORAÇÃO

ANDRÉ LUIZ PILASTRI

Chatterbot com interatividade ao Avatar encapsulado no ambiente virtual Second Life usando a base de conhecimento em AIML

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas como parte dos requisitos para obtenção do título de bacharel em Ciências da Computação, sob orientação do Prof..Ms. Kelton Augusto Pontara da Costa.

BAURU
2008

Pilastri, André Luiz

P637c

Chatterbot com interatividade ao Avatar encapsulado no ambiente virtual Second Life usando a base de conhecimento em AIML / André Luiz Pilastri – 2008.

89f.

Orientador: Prof. Ms. Kelton Augusto Pontara da Costa.

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

1. Chatterbot. 2. AIML. 3. Processamento de linguagem natural. 4. Inteligência artificial aplicada a jogos. I. Costa, Kelton Augusto Pontara da. II. Título

A Valquíria, amor

A Larissa, vida

Aos meus pais e a minha avó, sempre

ANDRÉ LUIZ PILASTRI

Chatterbot com interatividade ao Avatar encapsulado no ambiente virtual Second Life usando a base de conhecimento em AIML

Trabalho de Conclusão de Curso apresentado ao Centro de Ciências Exatas como parte dos requisitos para a obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Ms. Kelton Augusto Pontara da Costa.

Banca examinadora:

Prof. Ms. Kelton Augusto Pontara da Costa _____

Prof. Ms. Richard Gebara Filho _____

Prof. Ms. Ronaldo Martins da Costa_____

10 DE DEZEMBRO DE 2008 – BAURU - SP

AGRADECIMENTOS

Se alguém me pedisse para destacar as páginas mais importantes deste trabalho, não hesitaria em dizer que são estas, as páginas em que tento reproduzir, com a maior fidelidade possível, o nome da todas as pessoas que contribuíram diretas ou indiretamente, para a realização deste trabalho. Obviamente, ao longo de todos os capítulos estão os resultados de uma importante etapa de minha vida e que não deixam de ser tesouros valiosos. Mas é possível que dia eu não me recorde tão claramente de tudo o que escrevi, dos detalhes da pesquisa que realizei. Entretanto, as pessoas a quem agradeço nestas linhas certamente serão lembradas nitidamente durante toda a minha vida.

Em primeiro lugar, agradeço a Deus, pois ele é fiel, nosso Pai Maior, a inteligência Suprema, o Princípio de todas as coisas. Agradeço pelo esplendor de minha vida, pelo amparo concedido nos dias de preocupação e tristeza, pelo apoio durante as dificuldades, pelos sorrisos das pessoas com quem convivo e sempre trazem alegrias para minha alma.

A minha família, pelo apoio e incentivo em todos os momentos da minha vida, sejam eles quais forem.

A minha amada esposa Valquíria pela paciência nos dias em que estive distante (uma distância nem sempre física), tendo que compartilhar o meu tempo entre os diversos compromissos, entre eles o desenvolvimento deste trabalho, dedicando a você e a Larissa o carinho insuficiente.

Aos professores do Curso de Ciência da Computação, por todo conhecimento passado, em especial ao meu orientador Kelton Costa, por ter acreditado em meu potencial para a realização deste trabalho. Sem vocês, eu não teria sequer iniciado. Com vocês, pude concluí-lo com muita satisfação e bem-estar, porque acima de tudo, tivemos uma relação de amizade, e não simplesmente de aluno e professor.

A equipe LABCOMP/GTUSC, meus companheiros de trabalho, pela compreensão nos momentos difíceis durante a realização do trabalho.

Ao meu grande amigo Jaime Eugênio, em especial, pela paciência, ajuda e conhecimento que foram determinantes para a realização do trabalho.

A Eudes Canuto, por todo conhecimento e experiência compartilhados, pela ajuda e pelo incentivo.

A Rafael Landin, por todo conhecimento e experiência compartilhados, pela ajuda e pelo incentivo.

Enfim a todos os meus amigos, pelo apoio, pelos conselhos, pelas alegrias divididas e experiências vividas, e com quem eu aprendi e aprendo até hoje.

RESUMO

Os mundos virtuais tridimensionais são realidades alternativas onde os intervenientes interagem entre si e sobre elementos presentes nessas realidades. Chatterbots são sistemas que têm o objetivo de interagir como usuários em linguagem natural. A inclusão de interatividade possui um papel fundamental, no intuito de melhorar o desempenho desses sistemas, explorando e influenciando o comportamento do usuário. Neste Trabalho de Graduação será apresentado um Avatar que contém um Chatterbot com interatividade. Este Avatar tem por objetivo em responder perguntas, conversando com os usuários, levando um ser humano a pensar que está conversando com outra pessoa. As tecnologias envolvidas na criação do chatterbot do Avatar foram a linguagem AIML, que traz o conceito de intencionalidade dos diálogos.

Palavras-chave: Chatterbots, AIML, Processamento de Linguagem Natural, Inteligência Artificial aplicada a jogos.

ABSTRACT

Three-dimensional virtual worlds are alternatives realities that people who use it can interact with each other and with elements that are in these realities. Chatterbots are systems that have the aim to interact with users using natural language. The inclusion of interactivity has a fundamental function, it has the aim to improve system's performance, using and influencing the behavior of the users. In this graduation's research it will be showed an Avatar that has a Chatterbot with interactivity. This Avatar have to answer questions, talking to users, taking an human being think that is talking to another person. The technologies that help the creation of Chatterbots of the Avatar were AIML language, which has the idea of intentinality of the dialogues.

Keywords: Chatterbots, AIML, Natural Language Processing, Artificial Intelligence in games.

LISTA DE ABREVIATURAS

A-LIFE	*Artificial Life*
ALICE	*Artificial Linguistic Internet Computer Entity*
AG	*Algoritmos Genéticos*
AIML	*Artificial Intelligence Markup Language*
AV	*Ambiente Virtual*
AVC	*Ambiente Virtual Colaborativo*
AVDs	*Ambientes Virtuais Distribuídos*
DNA	*Ácido Desoxirribonucleico*
FSM	*Finite State Machines*
GPS	*General Problem Solver*
IA	*Inteligência Artificial*
HTML	*Linguagem de Marcação de Hipertexto*
MIT	Massachusetts Institute of Technology
PLN	*Processamento de Linguagem Natural*
RV	*Realidade Virtual*
SE	*Sistemas Especialistas*
SEGA	*Service Games*
SL	*Second Life*
W3C	*World Wide Web Consortium*
XML	*eXtensible Markup Language*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

A partir dos anos 90, a tecnologia de Realidade Virtual se expandiu em diferentes domínios do conhecimento.

Ao mesmo tempo, técnicas de inteligência artificial começaram a ser aplicadas na implementação de ambientes virtuais 3D, principalmente no que diz respeito à utilização de agentes inteligentes como personagens ou parte integrante do próprio mundo virtual (LIANCOURT, 1999; FRERY, 2002). A integração dessas duas tecnologias vem ampliando as possibilidades de interação humano-computador (AYLLET, 2000).

Os chatterbots são programas de computador que tentam simular conversações com os usuários, com objetivo de, pelo menos temporariamente, levar um ser humano a pensar que está conversando com outra pessoa. Essa possibilidade de se dar a uma máquina habilidade para interagir com o ser humano, através da compreensão e simulação do seu comportamento, tem sido, há muito tempo, alvo de pesquisas na área de inteligência artificial (LEONHARDT, 2005).

Os *chatterbots* também são visto como facilitadores no processo de interação usuário-máquina, sendo capazes de explorar o comportamento dos usuários (MOON, 1998) e até mesmo influenciá-los nos processos de tomada de decisão. Alguns estudos recentes mostram que o uso da personalidade traz uma melhoria no desempenho desses sistemas (SILVA, 2000).

Do ponto de vista da construção dos chatterbots, destacamos a AIML (*ArtificialIntelligence Markup Language*) (Wallace 2004), uma linguagem de marcação baseada em XML[1], de fácil uso e extensão. AIML foi usada na construção de dezenas de chatterbots, dentre os quais, A.L.I.C.E[2]., que é vencedor de vários prêmios internacionais na área.

Este documento está organizado da seguinte maneira: o primeiro capítulo trata da história da Inteligência Artificial, como ela foi criada, seus aspectos filosóficos e suas metodologias, demonstrando sua evolução até os dias de hoje.

O segundo capítulo trata da historia dos jogos eletrônicos e seus desenvolvimentos com as técnicas utilizadas atualmente para projetos de jogos.

O terceiro capítulo contém uma revisão bibliográfica que abrange o significado de ambientes virtuais e apresenta aspectos importantes de um mundo virtual chamado Second Life.

[1] http://www.xml.org/
[2] http://www.alicebot.org

No quarto capítulo, será apresentada uma revisão sobre os *chatterbots*, observando sua evolução histórica e suas características principais, a descrição de AIML a linguagem utilizada para o desenvolvimento da base de diálogos.

1.1 JUSTIFICATIVA

O tema escolhido visa mostrar que o Second Life é um mundo virtual e que há possibilidade de se criar uma inteligência artificial num avatar, para que haja interação com outros avatares (controlados por humanos), assim expandindo as oportunidades de aprendizado.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Incorporar um *chatterbot* num avatar que tem por objetivo conversar com os usuários sobre assuntos gerais, respondendo eventuais dúvidas dos usuários.

1.2.2 Objetivos específicos

Desenvolver *chatterbot* para um avatar encapsulado no ambiente virtual Second Life, este avatar será conectado no AVatravés de um *bot* desenvolvido na linguagem de programação C# . A linguagem utilizada para a elaboração da base de conhecimento do *chatterbot* foi AIML, pois ela procura negar qualquer influência lingüística e adota uma abordagem empiricista, baseada na observação de diálogos.

2 Definições de Inteligência Artificial

De acordo com o dicionário *Oxford* (WEHMEIER, 2000), artificial intelligence (inteligência artificial, ou simplesmente IA) corresponde a uma área de pesquisa sobre computadores simulando o comportamento humano inteligente. Para a grande maioria da população, IA é o cérebro por trás de máquinas poderosas, como as encontradas em filmes de ficção científica, e para os acadêmicos é uma infinita fonte de desafios e estudos sobre como recriar um ser inteligente através do uso de computadores (CHAMPANDARD, 2003).

A palavra "inteligência" vem do latim *inter* (entre) e *legere* (escolher). Inteligência significa aquilo que permite ao ser humano escolher entre uma coisa e outra. Inteligência é a habilidade de realizar forma eficiente uma determinada tarefa.

A palavra "artificial" vem do latim *artificiale*, significa algo que não natural, isto é, produzido pelo homem. Portanto, IA é um tipo de inteligência produzida pelo homem para dotar as maquinas de algum tipo de habilidade que simula a inteligência do homem.

IA é à parte da ciência da computação que esta voltada para o desenvolvimento de sistemas de computadores inteligentes, isto é, sistemas que exibem características, as quais associam-se com a inteligência no comportamento humano – por exemplo: compreensão da linguagem, aprendizado, raciocínio, resolução de problemas (FEIGENBAUM, 1992 *apud FERNANDES, 2005*).

2.1 História da Inteligência Artificial

De acordo com McCarthy e Papert (1969, apud BITTENCOURT, 2008), existem duas linhas de pesquisa para a construção de sistemas inteligentes: a linha *conexionista* e a linha *simbólica.* A linha conexionista visa à modelagem da inteligência humana através da simulação dos componentes do cérebro, isto é, de seus neurônios, e de suas interligações. Está proposta foi formalizada inicialmente em 1943, quando o neuropsicólogo McCulloch e o lógico Pitts propuseram um primeiro modelo matemático para um neurônio. Um primeiro modelo de rede *neuronal*, isto é, um conjunto de neurônios interligados, foi proposto por Rosenblatt. Este modelo

chamado *Perceptron*, teve suas limitações demonstradas por Minsky e Paper em livro ondes as propriedades matemáticas de redes artificiais de neurônios são analisadas.

McCarthy e Hayes (1969 apud BITTENCOURT, 2008), ainda ressalta que durante um longo período essa linha de pesquisa não foi ativa, mas o advento dos microprocessadores, pequenos e baratos, tornou praticável a implementação de máquinas de conexão compostas de milhares de microprocessadores, o que aliados à solução de alguns problemas teóricos importantes, deu um novo impulso às pesquisas na área. O modelo conexionista deu origem à área de redes neuronais artificiais.

Segundo Newell (1980 apud BITTENCOURT, 2008) a linha simbólica segue a tradição lógica e teve em McCarthy e Newell seus principais defensores. Os princípios dessa linha de pesquisa são apresentados no artigo *Physical symbol systems* de Newell . O sucesso dos *sistemas especialistas (SE)* (do inglês, “expert system”), a partir da década de setenta, estabeleceu a manipulação simbólica de um grande número de fatos especializados sobre um domínio restrito como o paradigma corrente para a construção de sistemas inteligentes do tipo simbólico. Para facilitar a apresentação, foi dividida a história da IA simbólica em “épocas”, conforme proposto em relatórios internos do MIT (Massachusetts Institute of Technology):

Clássica (1956-1970)

- Objetivo: simular a inteligência humana ;
- Métodos: solucionadoras gerais de problemas e lógicas;
- Motivo do fracasso: subestimação da complexidade computacional dos problemas.

Romântica (1970-1980)

- Objetivo: simular a inteligência humana em situações pré-determinadas.
- Métodos: formalismos de representação de conhecimento adaptados ao tipo de problema, mecanismos de ligação procedural visando maior eficiência computacional.
- Motivo do fracasso: subestimação da quantidade de conhecimento necessária para tratar mesmo o mais banal problema de senso comum.

Moderna (1980-1990)

- Objetivo: simular o comportamento de um especialista humano ao resolver problemas em um domínio específico.
- Métodos: Sistemas de regras, representação da incerteza, conexionismo.
- Motivo do fracasso: subestimação da complexidade do problema de aquisição de conhecimento.

Clássica

Para Ernst e Newell (1969 apud BITTENCOURT, 2008) a pesquisa em manipulação de símbolos se concentrou no desenvolvimento de formalismos gerais capazes de resolver qualquer tipo de problemas. O sistema GPS, é um exemplo deste tipo de pesquisa. Estes esforços iniciais ajudaram a estabelecer os fundamentos teóricos dos sistemas de símbolos e forneceram à área da IA uma série de técnicas de programação voltadas à manipulação simbólica, por exemplo, as técnicas de busca heurística. Os sistemas gerais desenvolvidos nesta época obtiveram resultados interessantes, por vezes até impressionantes, mas apenas em domínios simplificados, onde o objetivo era principalmente a demonstração da técnica utilizada, e não a solução de um problema real. O problema com os sistemas gerais é que a sua extensão a domínios de problemas reais se mostrou inviável. Isto se deveu a duas razões, uma relacionada com características teóricas dos métodos utilizados, e outra associada à natureza do conhecimento do mundo real.

Segundo Cook (1971 apud BITTENCOURT, 2008) a razão teórica é conseqüência do uso, nos sistemas gerais, de modelos baseados em lógica de primeira ordem como formalismo básico. A utilização desses modelos leva à chamada *explosão combinatória*: a memória e o tempo necessários para resolver um determinado problema crescem exponencialmente com o tamanho do problema. É inerente aos métodos baseados em lógica, independentemente das técnicas de programação utilizadas. A segunda razão está associada ao fato de que, freqüentemente, o conhecimento disponível sobre o mundo real é incompleto e parcialmente incoerente, e que por vezes a única forma de solução conhecida para determinados problemas reais consiste em uma série de regras práticas não fundamentadas por nenhum tipo de teoria geral do domínio que pudesse ser usada para orientar a solução.

Para Schneider (1985) esta situação levou a dois tipos diferentes de solução: (i) uso de métodos formais de inferência mais fracos do que a lógica de primeira ordem que garantissem uma certa eficiência aos programas, por exemplo, lógicas multivalores e linguagens terminológicas. (ii) Desenvolveram-se métodos heurísticos e lógicos não convencionais para permitir a representação de crenças, incoerências e incompletudes, por exemplo, lógica modal, lógica de exceções e lógica nebulosa.

Romântica

Segundo Russel e Norvig (2004), durante a década de setenta, a IA estava praticamente restrita ao ambiente acadêmico. Os objetivos da pesquisa eram, principalmente, a construção de teorias e o desenvolvimento de programas que verificassem estas teorias para alguns poucos exemplos. É interessante notar que o fato de que não havia interesse em construir programas de IA ``de verdade", isto é, com aplicações práticas, não se deve a uma eventual incompetência em programação dos pesquisadores em IA. Pelo contrário, foi a inspiração desses “hackers” que levou a conceitos hoje integrados à ciência da computação, como: tempo compartilhado, processamento simbólico de listas, ambientes de desenvolvimento de ``software", orientação objeto, etc., além da mudança da relação usuário-computador ao eliminar a intermediação de um operador e colocar cada usuário diante de sua estação de trabalho.

Para Hayes (1977 apud BITTENCOURT, 2008) uma mudança importante ocorreu ao longo da década de setenta em relação aos critérios acadêmicos de julgamento de trabalhos em IA: houve uma crescente exigência de formalização matemática. Se no início dos anos setenta, um programa, mesmo tratando de alguns poucos exemplos de um problema até então não tratado, já era considerado IA, isto não acontecia mais em 1980. O programa em si passou a ser a parte menos importante; a análise formal da metodologia, incluindo decidibilidade, completude e complexidade, além de uma semântica bem fundada, passou a ser o ponto fundamental. A década de setenta marcou também a passagem da IA para a “vida adulta”: com o aparecimento dos primeiros SE's, a tecnologia de IA passou a permitir o desenvolvimento de sistemas com desempenho intelectual equivalente ao de um ser humano adulto, abrindo perspectivas de aplicações comerciais e industriais.

Moderna

Segundo Davis e King (1977 apud BITTENCOURT, 2008) a tecnologia de SE se disseminou rapidamente e foi responsável por mais um dos episódios ligados a promessas não cumpridas pela IA: o sucesso dos primeiros SE's chamou a atenção dos empresários, que partiram em busca de um produto comercializável que utilizasse esta tecnologia. No entanto, um SE não era um produto: um produto, na visão dos empresários, não deveria ser um sistema específico para um dado problema, mas algo que fosse implementado uma única vez e vendido em 100.000 unidades, por exemplo, uma ferramenta para a construção de SE *(ASE)*. Com isso foram colocadas no mercado uma grande quantidade de *ASE's* que prometiam solucionar o problema de construção de SE's. A conseqüência foi uma grande insatisfação por parte dos usuários, pois, apesar de uma ferramenta de programação adequada ajudar muito a construir um sistema complexo, saber o que programar continua sendo o ponto mais importante.

Para Waterman e Lenat (1983 apud BITTENCOURT, 2008), se os *ASE's* deveriam ser vendidos como produtos de IA, então em algum lugar deveria haver IA, e o lugar escolhido foi o motor de inferência , que passou a ser considerado como sinônimo de IA. Isto levou à ilusão de que para construir um SE bastaria comprar um ASE, enquanto que a verdade é que a IA em um SE está basicamente na forma como é representado o conhecimento sobre o domínio, isto é, onde a IA sempre esteve: na tentativa de entender o comportamento inteligente a ser modelado, no caso o comportamento do especialista ao resolver um problema. Uma outra conseqüência desta visão distorcida dos ASE's foi a pouca ênfase dada inicialmente à aquisição de conhecimento, certamente a parte mais difícil do desenvolvimento de um SE. Se exageros existiram na publicidade dos ASE's, por certo houve também trabalhos descrevendo com fidelidade o potencial e as limitações da nova tecnologia.

Stylianou, Madey, Smith (1992 apud BITTENCOURT, 2008), esclarece que os ASE's são considerados como parte de uma tecnologia de desenvolvimento de ``software'' estabelecida, sendo objeto de diversas conferências internacionais e submetida a avaliações rigorosas de desempenho. Entre os diversos benefícios associados ao desenvolvimento de SE's podem-se citar: distribuição de conhecimento especializado, memória institucional, flexibilidade no fornecimento de serviços (consultas médicas, jurídicas, técnicas, etc.), facilidade na operação de equipamentos, maior confiabilidade de operação, possibilidade de tratar situações a partir de conhecimentos incompletos ou incertos, treinamento, entre outros.

Hill (1989 apud BITTENCOURT, 2008) esclarece que a gradativa mudança de metas da IA, desde o sonho de construir uma IA de caráter geral comparável à do ser humano até os bem mais modestos objetivos atuais de tornar os computadores mais úteis através de ferramentas que auxiliam as atividades intelectuais de seres humanos, coloca a IA à perspectiva de uma atividade que praticamente caracteriza a espécie humana: a capacidade de utilizar representações externas seja na forma de linguagem, seja através de outros meios. Esta nova perspectiva coloca os programas de IA como produtos intelectuais no mesmo nível dos demais, ressaltando questões cuja importância é central para os interesses atuais da IA, por exemplo, como expressar as características individuais e sociais da inteligência utilizando computadores de maneira a permitir uma maior produtividade, e como as propriedades das representações utilizadas auxiliam e moldam o desenvolvimento de produtos intelectuais.

2.1.1 Inteligência e Conhecimento

Segundo Rich (1993), para que se possa compreender uma ação inteligente, é necessário que sejam analisados todos os aspectos relativos à aquisição e desenvolvimento da inteligência. Dentro desse contexto, o conhecimento é o que faz que sejam possíveis o encadeamento e desenvolvimento da inteligência.

Rich (1993) ressalta que há determinadas características do conhecimento que devem ser analisadas como:

- Volumoso: o conhecimento é volumoso, pois possui diversos aspectos, características e detalhes. Quanto mais se quer esmiuçá-lo, mais e mais conhecimentos a pessoa terá.
- De difícil caracterização: muitas vezes tem-se apenas o conhecimento, mas não se sabe como foi adquirido, ou então, não se sabe explicá-lo. De fato, muitas vezes não se tem consciência do conhecimento que se possui.
- Conhecimento em constante mudança: o conhecimento está sempre crescendo, se modificando e se aperfeiçoando.
- Diferente de dados: o conhecimento não é dado. Dados fazem parte do conhecimento, sendo compostos de forma lógica de modo a permitir sua interpretação.

- É individual: o conhecimento é uma aquisição do indivíduo. Duas pessoas não adquirem o mesmo conhecimento de uma forma exata. Pode-se afirmar que duas pessoas possuem o mesmo conhecimento genérico, mas não como um conceito idêntico. Cada um faz interpretação do conhecimento.

2.1.2 Representação do Conhecimento

Segundo Fernandes (2005), a manifestação inteligente pressupõe aquisição, armazenamento e inferência de conhecimento.

Para que o conhecimento possa ser representá-lo. A grande parte do esforço em IA tem se concentrado em buscar ou aperfeiçoar formalismo para a representação do conhecimento.

Segundo Schwabe e Carvalho (1987), os estudos sobre representação do conhecimento, estão em boa parte, ligados à hipótese de representação do conhecimento de Brian Smith, ou seja, de que qualquer processo inteligente realizado por uma máquina deve conter uma estrutura que permita uma descrição proporcional do conhecimento exibido pelo processo, e que, independentemente de uma semântica, tenha um papel formal, causal e essencial na geração do comportamento que manifesta tal conhecimento (SCHWABE e CARVALHO, 1987 *apud* Fernandes 2005).

Para Fernandes (2005), existem diversos paradigmas de representação do conhecimento:

- Conhecimento Procedural: o conhecimento é representado em forma de funções / procedimentos.
- Redes: o conhecimento é representado por um rótulo de grafos direcionados, cujos nós representam conceitos e entidades, enquanto os arcos representam a relação entre entidades e conceitos.
- Frames: muito parecido com a rede semântica, exceto que cada nó representa conceitos ou situações. Cada nó tem várias propriedades que podem ser especificadas ou herdadas por padrão.
- Lógica: um modo de declaração que representa o conhecimento.
- Árvores de Decisão: conceitos são organizados em formas de árvores.

- Conhecimento Estatístico: uso de fatores de certezas, Redes Bayeseanas, Teoria de Dempster-Shaper, Lógica Fuzzy[3].
- Regras: sistemas de produção para codificar regras de condição / ação.
- Processamento Paralelo Distribuído: utiliza-se de modelos conexionistas.
- Esquemas híbridos: qualquer representação do formalismo que emprega a combinação de esquemas de representação do conhecimento.
- Casos: usa experiência passada, acumulando casos e tentando descobrir, por analogia, soluções para outros problemas.

2.2 Sistemas Especialistas

Um SE (FLORES, 2003 apud FERNANDES, 2005) é uma forma de sistemas baseados no conhecimento especialmente projetado para emular a especialização humana de algum domínio específico. Um SE irá possuir uma base de conhecimento formada de fatos, regras e heurísticas sobre domínio, tal como um especialista humano faria, e devem ser capazes de oferecer sugestões e conselhos aos usuários e, também, adquirir novos conhecimentos e heurísticas com essa interação.

Para Kandel (1992) os SE podem ser caracterizados como sistemas que reproduzem o conhecimento de um Especialista adquirido ao longo dos anos de trabalho. Já para Fernandes (1996), um SE deve ser construído com o auxílio de um especialista humano, o qual fornecerá a base de informação, através de seu conhecimento e experiência adquiridos ao longo dos anos. Os principais benefícios da utilização de um SE são: velocidade na determinação de problemas; a decisão está fundamentada em uma base de conhecimento, segurança; exige pequeno número de pessoas para interagir com o sistema; estabilidade; dependência decrescente de pessoal específico; flexibilidade; integração de ferramentas; evita interpretação humana de regras operacionais.

[3] http://www.pucsp.br/~logica/Fuzzy.htm

2.2.1 Estrutura de um Sistema Especialista

Para Fernandes (2005), um sistema especialista apresenta cinco componentes básicos:

- Base de conhecimento;
- Máquina de Inferência;
- Sistemas de Explicações;
- Interface do usuário;

Muitos sistemas utilizam regras como a base de conhecimento sendo considerados “sistemas baseados em regras”. Além disso, há outros sistemas de outras técnicas para a representação do conhecimento, como redes semânticas e *frames.* É formada pelas as regras e procedimentos que o especialista humano é capturado através de entrevistas ou observação (FERNANDES, 2005).

A máquina de inferência é o mecanismo que procura as respostas na base de conhecimento. Encontra regras necessárias para o problema e ordena-as de uma forma lógica. Funciona como um “supervisor”, que dirige a operação sobre toda a base de conhecimento do sistema. As funções básicas da máquina de inferência são inferência e controle. Depois de iniciado o sistema, a máquina de inferência e regras e compara estes fatos com a informação fornecida pelo usuário. Está operação da máquina de inferência é baseada em algoritmos que definem a busca específica e a unificação de regras. (*Ibidem*)

2.2.2 Aquisição de conhecimento

A aquisição do conhecimento é tida como “gargalo” do processo da construção de um sistema especialista, e o responsável por essa aquisição é o Engenheiro do Conhecimento (ROOK & CROGHAN, 1989 Apud FERNANDES).

O engenheiro do conhecimento é a figura central tanto na aquisição de conhecimento, como no desenvolvimento. Por isso a pessoa deve ter um preparo especial e algumas qualidades inatas (ROLANDI, 1986 *apud* FERNANDES, 2005).

Para GONÇALVES (1986 *apud* FERNANDES, 2005), existem duas abordagens para a engenharia do conhecimento.

- Abordagem psicológica;
- Abordagem baseada em modelos.

Abordagem psicológica: Procura obter SE que “imitam” o especialista. O conhecimento modelado é resultante do processo mental interno do especialista. Abordagem baseada em modelos: Parte de modelos de tarefas básicas como diagnósticos. Abordagem é a mais recente tendo suas ferramentas de implantação ainda em processo de desenvolvimento.

3 História dos Jogos Eletrônicos

A história dos jogos pode ser dividida em: antes da década de 70, décadas de 70, 80, 90 e 2000. Antes da década de 70, algumas empresas e pessoas já começavam a dar os primeiros passos na criação de jogos. Uma das principais empresas da indústria, a *Nintendo*, começou como uma empresa que fabricava cartas de baralho em 1989. Esse é o primeiro marco da história dos jogos eletrônicos (KENT, 2001).

O primeiro jogo eletrônico interativo criado na história foi o *Spacewar*, um jogo onde duas pessoas controlavam dois tipos diferentes de espaço-nave que deveriam combater entre si. Esse jogo foi programado por um estudante do MIT, Steve Russell, em um computador PDP-1 em 1961 (DEMARIA, 2004).

Existem alguns historiadores, que argumentam que o primeiro jogo eletrônico foi criado por Willy Higinbotham, um cientista do *Brookhaven Naticional Laboratory*. Higinbotham programou em 1958, um osciloscópio onde era possível jogar uma partida de tênis interativa (KENT, 2001).

Em 1970, Nolan Bushnell começou a trabalhar em uma versão fliperama do jogo *Spacewar*, chamada *Computer Space.* No seguinte, a empresa *Nutting Associates* comprou o jogo de Bushnell, colocando no mercado a primeira máquina de fliperama da história. Em 1972, Bushnell abre sua própria empresa, a *Atari* famosa pelo jogo *Pong* (criado pelo engenheiro Al Alcorn). Até o fim dessa década, muitas empresas entraram no mercado de jogos, como Taito, Midway e Capcom, além da Magnavox lançar em 1972 o computador Odyssey (KENT, 2001).

Seguindo a linha de tempo dos jogos em (KENT, 2001), nos anos 80 as máquinas de fliperama estavam em seu auge, com muitos jogos sendo lançados (*Donkey Kong*, *Tron* e *Q*Bert* são alguns exemplos) ao mesmo tempo em que surgiram os primeiros videogames 8-bit: *Famicom*, da Nitendo e Master System, da SEGA.

Segundo Demaria (2004) na área de jogos para o computador, houve também um grande lançamento de jogos e criação de empresas, sendo a *On-line* Systems (atual Sierra Online) uma das pioneiras no setor de jogos para o computador.

Em março de 1983, *Chris Crawford* (CRAWFORD, 2003) reuni alguns amigos desenvolvendores em sua casa para a primeira *Computer Game Developers Conference* (ataual GDC), onde discutiram assuntos sobre *game design* e negócios.

Para Kent (2001) a década de 90 foi marcada pelo lançamento e batalhas de videogames de 16-bit (Sega Genesis e Super Famicom, da Nitendo), de 32-bit (PlayStation da Sony e Sega Saturn) e o lançamento de videogames como 3DO da Panasonic e o Nintendo64 da Nintendo (esse de 64-bit).

Um grande marco para a história de jogos para computador foi o lançamento do jogo *Wolfstein 3D* pelo id *Software* em 1991, o primeiro jogo de tiro em primeira pessoa, atualmente um dos gêneros de jogos mais famosos entre os jogadores (KUSHNER, 2003).

Segundo Egm (2004) no final da década de 90 para 2000, a Sony e Nintendo divulgaram seus novos videogames de 128-bit (PlayStation 2 e GameCube, respectivamente), enquanto a Microsoft também entrou para esse segmento. A atenção da mídia e dos jogadores está voltada aos jogos online multiplayer massivos (centenas a milhares de pessoas jogando ao mesmo tempo num mundo virtual via internet) e aos videogames portáteis Sony PSP e Nintendo DS (esse inovando com duas telas, sendo uma delas sensíveis ao toque, como PDA's).

3.1 Inteligência Artificial para jogos eletrônicos

Segundo Funge (2004) para os desenvolvedores de jogos eletrônicos, as aplicações computacionais de IA e o significado do termo IA são diferentes dos encontrados no meio acadêmico. Para distinguir a IA utilizada em jogos e no meio acadêmico, os desenvolvedores adotaram o termo *Game AI*.

Para Schwab (2004) a principal diferença entre IA acadêmica e a IA para jogos é o objetivo que cada uma busca. No primeiro caso, o objetivo é buscar a solução de problemas extremamente difíceis, como imitar o reconhecimento que os humanos são capazes de realizar (reconhecimento facial e de imagens e objetos, por exemplo), entender e construir agentes inteligentes. No segundo caso, o objetivo de usar IA é a diversão. Sua importância é quanto aos resultados que o sistema irá gerar, e não como o sistema chega até os resultados; ou seja, o problema não é como ele age. Isso se deve pelo fato que jogos eletrônicos são negócios – os consumidores desses produtos os compram em busca de diversão, e não lhes interessa como a

inteligência de um personagem no jogo foi criada, desde que ele transforme o jogo divertido e desafiador, para tomar decisões coerentes com o contexto do jogo (TOZOUR, 2004).

Segundo Bourg (2004), um dos problemas encontrados sobre IA na indústria de jogos eletrônicos é a grande variedade de gênero dos jogos existentes e os comportamentos dos personagens, resultando numa interpretação ampla do que é considerada IA para jogos. Há desenvolvedores que consideram a interface do jogo com o usuário parte da área de IA, assim como o movimento e colisão também como IA.

Para Tozour (2004), é até vergonhoso que *Game AI* seja chamada e considerada IA, uma vez que no campo de IA para jogos é necessário criar agentes com comportamentos apropriados num determinado contexto, embora a adaptabilidade da inteligência humana nem sempre é necessária ou desejada para produzir tais comportamentos.

3.2 Inteligência Artificial e Jogos Eletrônicos

Segundo Silva (2000), a IA se desenvolveu muito nos últimos trinta anos, desenvolvendo cada vez mais algoritmos específicos para problemas específicos, não dando enfoque para a construção de sistemas que se aproximem da inteligência humana a IA Forte, também conhecida como "*Humam-Level AT*".

O mesmo autor ressalta que os sistemas de *Humam-Level AI* são como os vistos no cinema em C3PO, R2-D2 de Star Wars ou HAL de 2001 Uma Odisséia no Espaço. Eles demonstram características de inteligência humana como respostas em tempo-real, robustez, interação inteligente autônoma com o ambiente, planejado, criatividade, comunicação em linguagem natural, raciocínio de senso comum.

Para Schwab (2004), no começo do desenvolvimento de jogos eletrônicos, a programação de IA era mais usualmente conhecida por "programação de jogabilidade", pois não havia nada de inteligente sobre os comportamentos exibidos pelos personagens controlados pelo computador. A tabela 1 contém alguns exemplos de como IA foi utilizada em jogos com o passar do tempo.

Tabela 1: Linha de tempo da IA em jogos.

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

Fonte: SCHWAB (2004).

3.3 O início da Inteligência Artificial em jogos

Para Schwab (2004) muitos programadores do início da era de jogos eletrônicos implementavam padrões de movimentos ou movimentos repetitivos e/ou aleatórios para os personagens controlados pelo computador (como *Galaga* e *Donkey Kong*) como sendo a inteligência existente no jogo. Esse fato foi principalmente causado pela falta de memória e limitação existente na velocidade de processamento.

Os jogos de estratégia (*Civilization* de 1991, por exemplo) estão entre os pioneiros em IA para jogos, uma vez tais jogos necessitam de uma boa IA para que sejam jogáveis, pois requerem que o computador controle unidades (grupo e personagens) com estratégias e táticas complexas. Uma extensão dos jogos de estratégia são os jogos de estratégia em tempo real, onde toda a ação acontece em tempo real (ao contrário de outros jogos de estratégia, que ocorre em turno). A IA para esse gênero de jogo deve realizar buscas de caminhos (*pathfinding)* para centenas de unidades em tempo real (TOZOUR, 2002).

Tozour (2002) ressalta que jogos do gênero *"Sims"* (simuladores de gestão de cidades, fazendas, relações pessoais, entre outros), como o clássico *SimCity*, lançado pela empresa Maxis em 1989, foram os pioneiros a provarem o potencial dos métodos de *Artificial Life* (A-Life). Outro jogo famoso da *Maxis*, *The Sims* (2000), conta com personalidades profundas em seus agentes inteligentes. Tal jogo é exemplo do potencial uso das máquinas de estado *fuzzy* (fuzzy-state machines ou FuSM) e A-Life. Outro exemplo do uso de A-Life em jogos é o titulo *Creatures* (criado pela CyberLife em 1996), que simula um "DNA digital" único de cada personagem.

Em jogos de tiro de primeira pessoa (*fist-person shooter ou FPS*), como Half-Life (Valve) e Unreal: Tournament (lançado em 1999 pela *Epic* Mega Games), a IA ficou conhecida pelo excelente nível tático dos inimigos, desenvolvida através do uso de máquinas de estados finito (FSM) e scripts que determinam como um agente inteligente deve agir em várias situações (WOODCOCK, 1999; TOZOUR, 2002).

3.4 Técnicas e algoritmos de IA implementada em jogos

Para Bourg (2004), existem diversas técnicas e algoritmos utilizados pelos desenvolvedores de jogos para dar aos personagens uma certa inteligência (ou ao menos fazer com que os personagens pareçam ser inteligentes) e uma personalidade.

Segundo Lamothe (1999), um dos princípios básicos de IA determinísticos e padrões de movimento, onde os comportamentos são pré-programados ou pré-processados.

Para Dalmau (2004), existem quatro tipos principais de IA que são implementadas em jogos: máquinas de estado, sistemas baseados em regras, algoritmos de busca e algoritmos genéticos.

Para Lamothe (1999), os algoritmos de IA determinísticos, juntos com padrões de movimento, foram utilizados nos primeiros jogos eletrônicos da história, e são compostos por movimentos aleatórios, algoritmos de perseguição e evasão. Estes movimentos aleatórios podem ser implementados simplesmente obtendo um valor aleatório e incrementando a posição e um personagem com tal valor. O algoritmo de perseguição verifica a posição de um personagem 1 em relação à posição de um personagem 2, e avança em direção a ele. O algoritmo de evasão faz o personagem 1 se distanciar do personagem 2. Os padrões de movimento fazem com que um personagem se movimente em um determinado padrão, por exemplo, um personagem pode fazer uma ronda em uma área retangular.

3.4.1 Agentes Inteligentes

Para Russel e Norvig (2004), um agente inteligente é todo aquele que consegue interagir com o ambiente a sua volta por meio de sensores e atuadores. O agente humano tem olhos, ouvidos como sensores e mãos, pernas e boca como atuadores, já um agente robótico teria câmera e detectores da faixa infravermelho. Todo agente pode perceber as próprias ações, porém as conseqüências de suas ações não.

Agentes inteligentes são considerados como entidades artificiais que podem imitar o comportamento humano em um sistema computacional (LIEBERMAN, 2003). De um modo geral, um agente trabalha interagindo com usuários, processos, ou outros agentes.

De acordo com o mesmo autor dentre os diferentes tipos de agentes, os agentes inteligentes de interface se destacam por apoiar o usuário de forma interativa.

Para Lashkari (2005), agentes são vistos como a principal ferramenta de uma interface futurista, onde o computador se comunica com seus usuários utilizando expressões semelhantes à forma de comunicação entre os próprios seres humanos.

Em uma ambiente virtual, os agentes inteligentes podem assumir um papel humano (TAVARES, 2001), representando um modelo computacional de uma figura humana que pode se mover, falar e atuar de forma autônoma. Agentes autônomos são capazes de comportamento adaptativo e independente, de forma a interagir com participantes humanos, objetos simulados e eventos (BJORN, 2005). Os agentes autônomos podem ser utilizados para:

- Substituir participantes humanos ou aumentar o número de participantes em treinamentos de indivíduos ou grupos;
- Instrução e educação;
- Escolta e assistência quando navegando em ambientes virtuais complexos.

3.4.2 Máquinas de estado

Para Perúcia et. al. (2005), máquinas de estados finitos (FSM), são ferramentas importantes para o planejamento de um jogo. O objetivo principal é modelar o funcionamento dos jogos antes da codificação. Por isso já deve ter uma visão bem clara do funcionamento do jogo. A figura 1 abaixo é exemplo de uma máquina de estados bem simplificada.

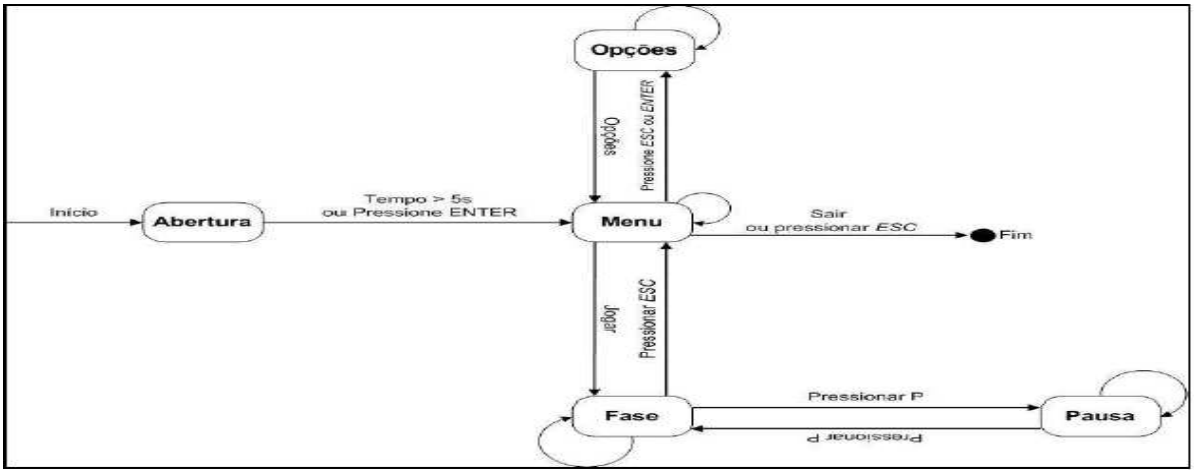

Figura 1 - Exemplo de uma Máquina de Estados Finitos
Fonte: Adaptado de PERÚCIA et. al. (2005)

Uma máquina de estado finita é uma máquina abstrata que define os estados em que um personagem pode se encontrar e quando o mesmo muda de estado. O estado atual da máquina determina como o personagem deve atuar no jogo. Máquinas de estado foram usadas no inicio da criação de jogos (com IA) e são usadas até hoje por serem de fácil entendimento, implementação e depuração de erros. No jogo Pac-man, por exemplo, uma máquina de estado é implementada para cada fantasma do jogo. Um fantasma pode estar nos seguintes estados: “procurando o jogador”, “perseguindo o jogador” e “fugindo do jogador”. Quando o fantasma está procurando o jogador, ele apenas se movimenta pelo labirinto até encontrar o jogador. Quando ele se depara com o jogador, verifica se ele pode perseguir o jogador ou se precisa fugir (isso acontece quando o jogador obtém poder de “engolir” o fantasma), e troca de estado conforme a situação. Se o fantasma pode seguir o jogador, ele muda seu estado para “perseguindo jogador” e tenta alcançar o jogador. Se durante esse tempo o jogador ganha a habilidade de engolir o fantasma, o fantasma muda seu estado para “fugindo do jogador” (BOURG, 2004).

Os desenvolvedores também utilizam a lógica de *fuzzy* em máquinas de estado *fuzzy* para criar resultados de ações que são menos previsíveis e para reduzir o grande trabalho de enumerar a grande quantidade de regras de if-then. A lógica *fuzzy* permite criar regras usando menos precisas, criando agentes com um conhecimento imperfeito, uma vez que essa lógica é baseada em níveis de incerteza e verdades em uma sentença (WOODCOCK, 1999; BOURG, 2004).

3.4.3 Sistemas baseados em regras

Segundo Dalmau (2004), alguns fenômenos não são fáceis de serem modelados em termos de estados e transições. Considerando como exemplo os seguintes fenômenos de um cachorro virtual:

- Se há um osso por perto e o cachorro está com fome, ele irá comê-lo;
- Se o cachorro está com fome, mas não há nenhum osso por perto, ele pode procurar por um;
- Se o cachorro não está com fome, mas está com sono, ele irá dormir;
- Se o cachorro não está com fome e não está com sono, o cachorro irá andar e latir.

De acordo o mesmo autor, essas quatro sentenças são difíceis de serem representadas através de uma máquina de estados, pois cada sentença leva a um estado da máquina e cada estado pode transitar para qualquer um dos outros estados. Esse tipo de problema é conhecido por conhecimento global. Máquinas de estados são úteis para situações locais (onde dado um estado, apenas algumas condições podem ser aplicadas como saída).

Neste exemplo, o cachorro se comporta de acordo com um conjunto de prioridades ou regras. Um sistema baseado em regras tem a forma “Condição -> Ação”. No exemplo citado, as regras teriam a seguinte forma:

- Fome & osso por perto -> comer;
- Fome & não osso por perto -> procurar;
- Não fome & com sono -> dormir;
- Não fome & sem sono -> andar e latir.

Dessa maneira, são especificadas as condições que ativam as regras assim como quais ações devem ser tomadas caso a regra é ativada.

3.4.4 Algoritmos de busca

Para Bourg (2004), a busca é um dos problemas mais básicos de IA para jogos. Quando um jogo implementa uma busca pobre (ou “burra”), o resultado é o personagem que parecem totalmente artificiais e sem inteligência de navegar entre locais e desviar de obstáculos, o que acaba com a imersão do jogo e a diversão.

Para solucionar o problema de busca (sair de um ponto e chegar a um destino), diverso algoritmos podem ser utilizados, sendo o algoritmo A* o mais famoso e implementados em jogos, embora soluções como o algoritmo de *Dijkstra* e *waypoints* também são utilizados (LAMOTHE, 1999; DALMAU, 2004).

Para Lamothe (1999) em muitos jogos, os desenvolvedores representam o mundo virtual por onde um personagem caminha através de “grades” (grids), onde cada célula pode representar um nó de um grafo. Um custo é associado para cada célula do grid, utilizado heurística do A*.

Segundo Bourg (2004), o uso de busca pode consumir muito tempo do processador, é possível contornar esse problema através de caminhos pré-calculados, chamados de *waypoints*, quando o jogo permite esse tipo de solução. Os *waypoints* são nós em locais do mundo virtual que auxiliam no deslocamento de um lugar (nó atual) para outro (nó destino) através de caminhos pré-calculados ou métodos de busca e baixo custo.

3.4.5 Algoritmos genéticos

Para Iyoda (2000) os algoritmos genéticos (AG) empregam uma terminologia originada da evolução natural e da genética. Um indivíduo da população é representado por um *cromossomo*, o qual contém a codificação (genótipo) de uma possível solução do problema (fenótipo). Cromossomo são usualmente implementados na forma de vetores, onde cada componente do vetor é conhecido como gene. Os possíveis valores que um determinado gene pode assumir são denominados alelos.

Para Dalmau (2004), o uso de AG em jogos pode ser a geração de uma população, criando diferentes indivíduos de acordo com um DNA virtual, sendo esse representado por um vetor de valores, cada um sendo um parâmetro da espécie a ser modelada. Essa técnica pode ser utilizada para a criação de pedestres em um jogo onde o mundo virtual seja uma cidade.

Já para Fernandes (2005), AG’s são métodos usados para resolução de problemas de busca e otimização inspirados no princípio da evolução. Basicamente o, os AG’s tratam os problemas de otimização como no processo iterativo da busca pela melhor solução para problemas. Iniciando com um conjunto de seleções naturais, constituindo a população inicial. Com essa população é gerada uma nova população, sendo que a cada repetição a população é trocada, chegando até a solução do computador.

4 Ambientes Virtuais

Segundo Kisniz e Knaesel (2003) Ambiente Virtual (AV) é uma representação de um espaço próprio para determinado fim. Esse espaço tem em seu interior uma série de elementos que objetivam a transmissão de significados e a tomada de ações/decisões. Essa representatividade pode ser de um local real. Os usuários que estiverem interagindo nesse "espaço" virtual sentem-se materializados dentro dele e não apenas como meros espectadores apreciando informações na tela do computador.

Para Pinho e Rebelo (2004) AV nada mais é do que um cenário onde os usuários de um sistema de realidade virtual (RV) podem navegar e interagir dinamicamente, característica esta importante dos ambientes virtuais, uma vez que os cenários modificam-se em tempo real à medida que os usuários vão interagindo com o ambiente. Um AV pode ser projetado para simular tanto um ambiente real.

Para Kiniz e Knaesel (2003) a RV oferece a possibilidade de criar ambientes com três características básicas e necessárias:

- imersão, sentimento de estar presente nos fatos que estão ocorrendo no ambiente;
- interação: o usuário pode interferir com o ambiente e vice-versa;
- envolvimento: o ambiente atua como um agente motivador do usuário para que ele participe.

4.1 Ambientes Virtuais Distribuídos e Compartilhados

Os Ambientes Virtuais Distribuídos (AVDs) são caracterizados como um AV interativo em que os usuários dispersos geograficamente têm como objetivos a cooperação e o compartilhamento dos recursos computacionais em tempo real usando um suporte de rede de computadores para melhorar o desempenho coletivo por meio da troca de informações (BENFORD, 1994; ZYDA, 1999 apud KINER; ROMERO, 2006, p.60).

Para Rinaldi et al. (2006 apud KINER; ROMERO, 2006, p.60), em AVDs os usuários podem compartilhar um mesmo espaço tridimensional virtual de trabalho (*workspace*), onde poderão se auxiliar na execução de uma determinada tarefa, baseando-se nos princípios de trabalho cooperativo baseado em computador (CSCW – *Computer Supported Cooperative Work*). Nesse sentido, classificar-se á o sistema como um Ambiente Virtual Colaborativo (AVC).

O principal diferencial de um AVC é a possibilidade de cooperação entre os usuários na execução de uma determina tarefa. As propriedades desse tipo de ambiente relacionam-se com: espaço, presença e tempos compartilhados; comunicação entre os participantes e; interação com o ambiente (SNOWDON, 2001 apud KINER; ROMERO, 2006, p.61).

4.2 O Ambiente Virtual Second Life

Second Life é a busca por uma segunda vida ou mesmo uma vida paralela (ideal ou mesmo alternativa). Para Emiliano de Castro, diretor de marketing Second Life Brasil:

> Second Life não é um jogo. Não há missões, fases ou objetivos pré-definidos. Second Life é um metaverso: um mundo virtual tridimensional que oferece a qualquer um que tenha acesso à internet a possibilidade de ter uma segunda vida. As pessoas que se cadastram no Second Life são mais do que internautas ou usuários. São residentes de um universo online onde é possível voar ou se teletransportar, trabalhar, fazer novos amigos, estudar, criar produtos e obras de arte, passear, namorar, fazer compras, vender, dançar, anunciar...

Para Rymaszewski et al. (2007), o Second Life (SL) é um AV no qual quase tudo é criado pelos usuários. O SL foi concebido por Philip Rosedale, ele começou a trabalhar no conceito que se tornaria o SL(o primeiro nome foi LindenWorld em 1991). E em 23 de junho de 2003, o SL foi lançado para o grande público compreendendo uma rede com mais de 4.000 servidores rodando cerca de 11.000 regiões e mais de 11 milhões de contas cadastradas.

Conforme figura 2 a seguir é demonstrada uma imagem inicial no momento em que um usuário se conecta pela primeira vez no ambiente SL após a instalação do programa no computador (software cliente).

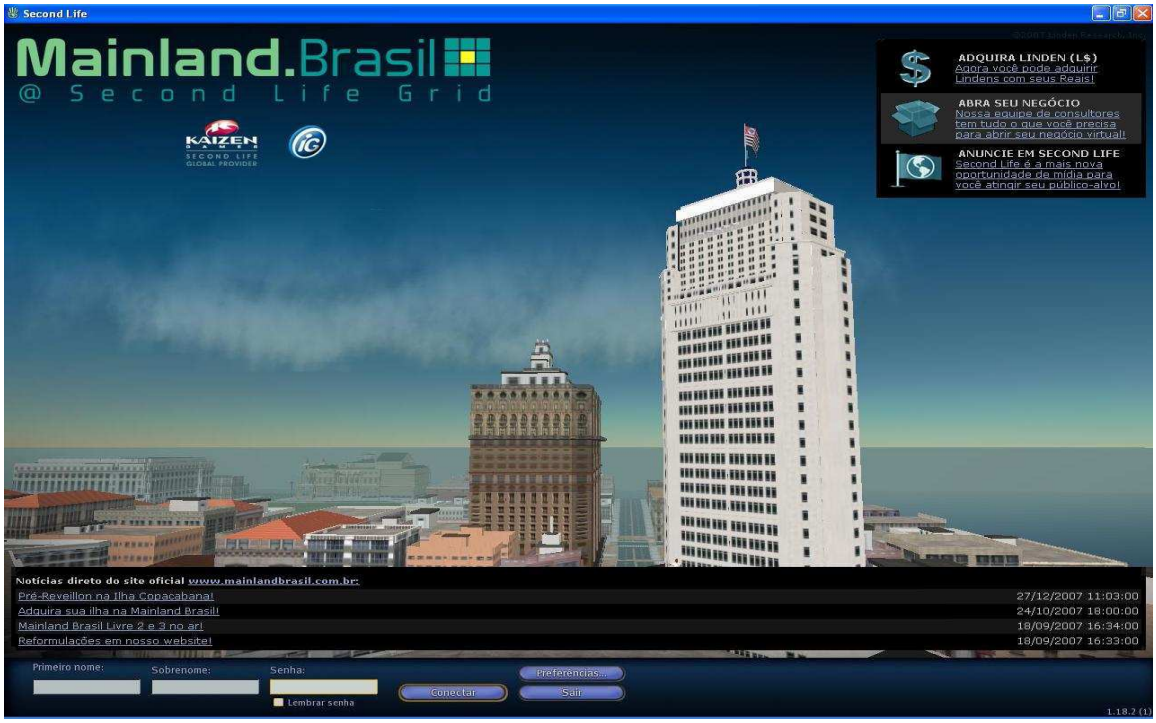

Figura 2 – Cliente do Second Life exibindo a interface do programa em português
Fonte: Rymaszewski et al., 2007, p.25

Segundo Ralha (2008), o SL não é um jogo, pois não há objetivos definidos a serem alcançados, bônus points, próximos níveis ou um monstro escondido em um canto de cada ilha e muito menos um game over.

O SL é um mundo virtual, tridimensional, que necessita ser conectado a internet banda-larga, e a um computador, criado pela “Linden Lab”, e largamente desenvolvido sob a influência de 11 milhões de internautas. Com esse novo AV é possível investir recursos financeiros, estudar, trabalhar, casar, ter filhos, procurar ou oferecer empregos, namorar, comprar ou vender coisas reais e virtuais, praticar atos de vandalismo e até crimes dos mais perniciosos (ANGELUCI; SANTOS, 2007),.

A figura 3 mostra a visão da ilha de orientação do Second Life Brasil, onde se encontra informação necessária para auxiliar o seu aprendizado sobre o mundo virtual.

Figura 3 – Ilha de orientação do Second Life Brasil – MLBR Orientação (136,100,0)
Fonte: Ralha, 2007, p.25

Algumas estimativas da plataforma *SL* demonstram a sua real importância, com economia própria, e uma moeda diferenciada (o "Linden Dólar – L$") que tem sua cotação atrelada ao dólar real, tendo seu PIB anual estimado em 220 milhões de dólares, sua economia virtual desenvolve-se à taxa de 300% ao ano, com uma movimentação mensal de US$ 18 milhões (VIEIRA, 2007).

Para viver virtualmente na segunda vida, os habitantes obrigatoriamente necessitam de uma forma de imprimir sua identidade, que é visualizada através de um avatar (ANGELUCI; SANTOS, 2007). Trata-se da representação corporal virtual, que pode ser construída espelhando-se na real imagem de seu criador, ou seja, de um ser humano, ou pode ter a semelhança de um animal ou até mesmo de um extraterrestre. Tais fatores, aparentemente são irrelevantes, mas a aparência dos avatares, nada mais é para o seu criador, do que um direito de propriedade intelectual.

4.2.1 Avatares e Ilhas

Para Ralha (2008), no mundo virtual do SL o avatar representa o usuário, onde é possível mudar de aparência a qualquer momento e quantas vezes quiser. As ferramentas do SL incluem um poderoso editor de aparência de avatar. Um avatar consiste de forma – o corpo – e utensílios – o que o corpo veste, além de qualquer coisa usada junto a ele (RYMASZEWSKI et al., 2007).

Figura 4 - mostra a personalização dos avatares
Fonte: Rymaszewski et al., 2007, p.85 e p.90

O mesmo autor ressalta que a grande maioria dos cidadãos do SL prefere ter uma aparência humana, mas existem alguns avatares baseados em personagens de filmes, quadrinhos, animais e livros. A escolha do avatar não afeta seu acesso às opções do SL e aos privilégios do mundo virtual, exceto nos casos em que elas violarem as normas da comunidade.

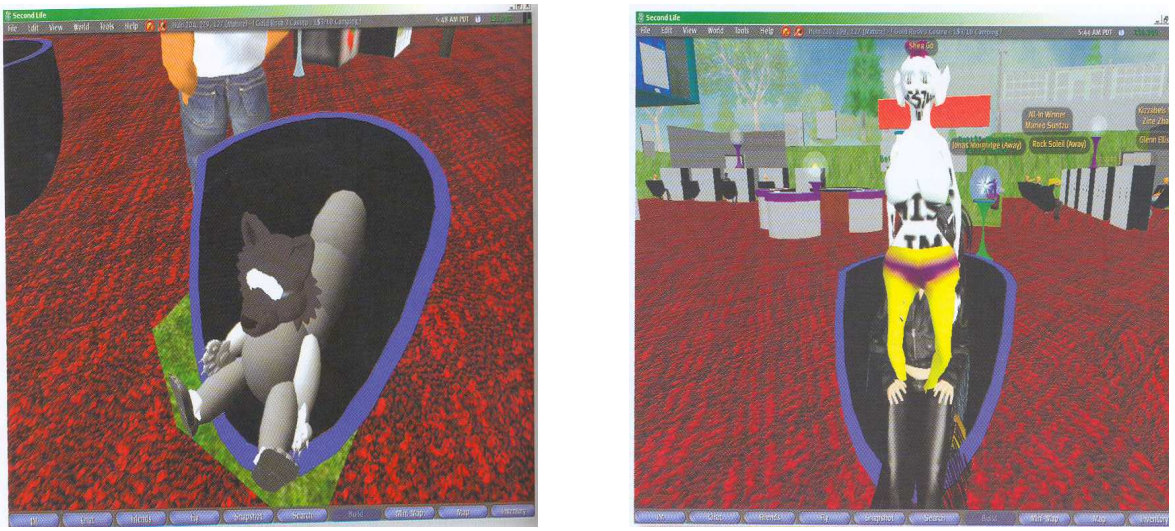

Figura 5 - mostra diferentes tipos de avatares
Fonte: Rymaszewski et al., 2007, p.14 e p.34

O SL está divido em regiões também ou ilhas, elas tem uma área de 256m x 256m ou 65.535 m2. Estas áreas recebem nomes como, por exemplo, Brasil, Berrini, Freebie Island etc.

Figura 6 - Ilha Brasil – local diversão
Fonte: http://www.gruposecondlife.com.br/galeria-de-imagens/ilha-brasil

Tudo no SL possui um *owner* (proprietário), inclusive as terras. Se elas estão onde estão é que porque alguém pagou para que a região fosse criada e paga um valor mensal que continue a existir. Cada região pode ser loteada entre um ou mais proprietários. Estes lotes são conhecidos como *lands* ou simplesmente terrenos e também podem receber um nome. O nome do terreno em que se encontra o avatar é exibido na superior da tela, à direita das coordenadas X,Y,Z.

4.2.2 Linguagem de Script

Sem uma definição muito precisa, as linguagens de Script apresentam algumas características em comum. As linguagens de Script são interpretadas, possuem gerencia de memória automática, tipagem dinâmica, fornecendo suporte a criação de estruturas de dados e manipulação de dados. Essas linguagens normalmente funcionam em programas implementados em linguagens compiladas como C ou C++, outra vantagem é que elas são seguras, não podendo acessar serviços sem a permissão do programa "hospedeiro". A união dessas características torna as linguagens de script uma ótima ferramenta para o desenvolvimento de um jogo (CELES et al., 2004).

Para o mesmo autor utilizar uma linguagem de script no desenvolvimento de jogos traz diversos benefícios, elas podem ser usadas para o desenvolvimento de um jogo inteiro, para definição completa dos objetos dos jogos, gerenciamento de algoritmos de IA, controlar personagens, tratar eventos de entrada e montar a interface com o usuário.

A programação de objetos no SL é feita através da linguagem LSL (Linguagem de Script da Linden), uma linguagem LSL, é uma linguagem derivada do C e cuja sintaxe será um tanto familiar para todos os *scripters* (profissionais que desenvolvem scripts) com experiência prévia com linguagens como C#, Java, C++ ou Javascript. A linguagem LSL não é orientada a objetos e sim baseada em eventos e estados, o que significa que nos restringe a usar tipos que encontramos em todas as linguagens como *integer* e *string*, e outros exclusivos como variáveis 3D do tipo *vector* e *quaternion*[4]. Está presente na linguagem uma extensa lista com mais de 300 funções que engloba funções para manipulação de Física e interação entra avatares (RALHA, 2008).

Para Rymaszewski et al. (2007) a LSL é uma linguagem de scripts que trabalha com o servidor, num software chamado simulador. Este faz justamente o que o nome indica - simula o mundo virtual do SL. Cada simulador gera todas as coisas de 65.000 **m2** de terra virtual – construções, física, scripts.

[4] http://secondlife-clediney.blogspot.com

5 Chatterbots

De acordo com Bickmore e Laven (2002 apud CANUTO, 2005) os chatterbots são sistemas que utilizam linguagem natural para dialogar com o usuário Os primeiros chatterbots tinham aplicação restrita a estudos acadêmicos. Atualmente, eles são considerados alternativas capazes de desempenhar o papel de facilitadores em diversas aplicações como, por exemplo, comércio eletrônico e ensino a distância. Esses sistemas têm a capacidade de explorar o comportamento social do usuário perante o computador, mesmo quando não são programados com essa intenção.

Atualmente, chatterbots despertam interesse tanto no meio acadêmico quando do mercado devido ao fato de possuírem interfaces amigáveis com o usuário, provendo mais naturalidade na interação. Além disso, eles podem explorar uma relação social dos usuários com as máquinas. Segundo Galvão (2003), alguns estudos mostram que os principais problemas na construção e eficácia desses sistema são: aquisição e gerenciamento da base de diálogos, identificação das sentenças digitadas pelos usuários, o uso da personalidade.

5.1 História

A história dos chatterbots pode ser dividida em três gerações (NEVES E BARROS, 2003 apud CANUTO, 2005), de acordo com a forma como esses sistemas são implementados.

O marco inicial da primeira geração foi o chatterbots ELIZA (WEIZENBAUM, 1966 apud CANUTO, 2005), alguns anos após Turing propor o Jogo da Imitação. ELIZA se comporta como uma terapeuta induzindo o usuário a revelar informações pessoais. Através do princípio psicanalítico Rogeriano, que consiste em responder a perguntas repetindo as frases do pacientes, o sistema faz com que o diálogo se torne introspectivo. Esse sistema despertou atenção na comunidade científica, pois sistemas como estes, relativamente simples de serem implementados, são capazes de influenciar o comportamento dos usuários. A arquitetura de ELIZA é bem simples, e sua programação baseada em técnicas de casamento de padrões (AHO, 1980 apud CANUTO, 2005). Uma característica marcante de ELIZA é a forma de construção de alguns casos ele modifica a frase do interlocutor, retornando uma pergunta relacionada ao que foi dito. Por exemplo, se o usuário digita uma frase como “I AM having a very bad Day”, a réplica

retornada por ELIZA é a seguinte: “Did you come to me because you were having a very bad Day?”. Porém, esse tipo de construção de réplicas possui limitações, retornando frases sem sentido, que atrapalham o diálogo. Por exemplo, quando o usuário digita “I AM doing fine thank you”, ElLIZA responde “How long have you been doing fine thank I?”. Outros chatterbots que merecem destaque na primeira geração são THE PC THERAPIST (WEINTRAUB, 1992 apud CANUTO, 2005), vencedor do Prêmio Loebner [5](LOEBNER, 2004) em 1991, 1992, 1993 e 1995, e o chatterbots são inspirados em ELIZA, utilizando técnicas de casamento de padrões e outras técnicas mais avançadas do ponto de vista de IA e Engenharia de Software.

Os crescentes avanços na área de Processamento de Linguagem Natural (PLN), IA e Engenharia de Software inspiraram os cientistas a produzirem sistemas que utilizassem as novas técnicas disponíveis, iniciando a segunda geração os chatterbots na década de 1990, Um chatterbot marcante dessa geração é JULIA, desenvolvido por Michael Mauldin (MAULDIN, 1994 apud CANUTO, 2005), que utiliza Redes Neurais para criar seu modelo de conversação. Esse chatterbot fazia parte de um sistema de entretenimento virtual, interagindo com outros usuários em sala de bate papo, como se fosse um ser humano (EDWARD, 2000 apud CANUTO, 2005). Outro sistema dessa geração que merece destaque é JFRED[6], um *framework* para desenvolvimento de chatterbots (GARNER, 2000 apud CANUTO, 2005). Alguns chatterbots utilizam essa tecnologia foram premiados, porém o desempenho dos chatterbots de primeira geração é considerado melhor do que o desempenho dos *bots* da segunda geração, segundo as avaliações do Prêmio Loebner.

A terceira geração é marcada pela linguagem AIML (*Artificial Intelligence Markup Language)*, usada na construção de A.L.I.C.E. (*Artificial Linguistic Internet Computer Entity)* (WALLACE, 2004), ganhador do prêmio Loebner de 2000. AIML é uma linguagem de marcação baseada em XML (*eXtensible Markut Language*), utilizada para a elaboração da base de conhecimento de *chatterbots*. O modelo proposto por Wallace procura negar qualquer influência lingüística e adota uma abordagem empiricista, baseada na observação de diálogos. A.L.I.C.E. ganhou reconhecimento internacional e ganhou vários prêmios Loebner recentes. Através da produção de vários processadores e ferramentas que auxiliam no desenvolvimento de *chatterbots*

[5] http://www.loebner.net/Prizef/loebner-prize.html
[6] http://www.simonlaven.com/jfred.htm

utilizando AIML, Wallace favoreceu a criação de uma comunidade de produção sistemas utilizando AIML.

5.2 AIML – Linguagem de marcação da Inteligência Artificial

AIML é uma Linguagem de Marcação da IA, baseada na XML (Linguagem extensível de formatação). A linguagem XML utiliza-se de *tags* para estruturar seus dados, simplificando assim, a sua implementação, sendo a mesma uma especificação técnica desenvolvida pela W3C (World Wide Web Consortium - entidade responsável pela definição da Internet), para superar as limitações do HTML (Linguagem de Marcação de Hipertexto), que é o padrão das páginas da Internet. Pode-se dizer que uma das vantagens da linguagem XML é a confiabilidade na execução dos comandos inseridos, um exemplo seria o uso das *tags*, cada *tag* consiste em duas partes, uma que inicia e outra que fecha o comando. Porém, em muitos casos, se uma *tag* é aberta no HTML e não é fechada, a página é exibida mesmo assim. Já no XML, se houver qualquer erro desse tipo, a aplicação simplesmente pára. Percebe-se com esse exemplo, que o HTML é uma linguagem mais tolerante, enquanto o XML é significativamente rigoroso, tornando assim o bom funcionamento da aplicação desenvolvida (INFO WESTER, 2008).

A AIML teve seu início no ano de 1995, pelo Dr. Richard Wallace e a comunidade de *Software* Livre Alicebot, criadores de A.L.I.C.E, *Artificial Linguistic Internet Computer Entidade*, sendo o primeiro *chatterbot* implementado em AIML, e tornando se referência para todos os demais robôs de conversação (WALLACE, 2004).

Além disso, a linguagem facilita a difícil tarefa de criar robôs de conversação, que buscam simular um dialogo escrito com a intenção de convencer o usuário que está trocando informações com um ser humano e não com uma máquina.

A AIML utiliza um conjunto de *tags* e comandos que servem para implementar a base de conhecimento do *chatterbot*. Os arquivos criados em AIML devem conter as seguintes *tags* básicas (WALLACE, 2004):

- <aiml> : inicia e termina um bloco programado em AIML.
- <*category*> : identifica uma “unidade de conhecimento” na base de conhecimento.
- <*pattern*> : indica o padrão da mensagem que será digitada pelo usuário.
- <*template*> : indica a resposta para a pergunta.

- <*random*> : seleciona respostas aleatórias.
- : Marca o bloco de respostas aleatórias, é utilizado dentro do bloco *random*.
- <*that*> : Registra a última sentença, gerando uma seqüência no diálogo.
- <*srai*> : Redireciona para outra questão ou categoria.
- <*javascript*> : Executa um comando em *javascrip.*

No exemplo a seguir, utiliza-se a *tag* <*srai*>, usando-a para o recurso de recursividade. Quando queremos ter a mesma resposta para várias perguntas, como as muitas opções de despedidas, que só pode ser um padrão de entrada <*PATERN*> por categoria <*CATEGORY*>. Deve-se direcionar uma categoria para outra, que tenha a resposta a se dar.

Exemplo 1.

```
<AIML version”1.0”>
<category>
<pattern> tchau</pattern>
<template>
<srai>Ate logo</srai>
</template>
</category>
<category>
<pattern> Ate logo</pattern>
<template>
Ate logo, foi ótimo conversar com você.
</template>
</category>
</aiml>
```

A sentença <*SRAI*>ATE LOGO<*/SRAI*> é uma chamada à categoria que tem como padrão de entrada TCHAU. Conforme pode ser visto a seguir:

Pessoa: tchau.

Robô: até logo, foi ótimo conversar com você.

Pessoa: até logo.

Robô: até logo, foi ótimo conversar com você.

6 Metodologia

Este trabalho de graduação foca esforços de incluir interação no chatterbot do avatar encapsulado no Second Life. As tecnologias utilizadas para o desenvolvimento foram a criação de um *ALPBot* na linguagem *C#* usando a libsecondlife[7].

O projeto é um esforço dirigido para compreender como funciona o Second Life a partir de uma perspectiva técnica e também integrando o *metaverse*[8] com o resto da *web*, isto inclui entender também como funciona o *client* oficial do Second Life e como ele se comunica com os servidores simulador Second Life. Assim como *AIML*, que inclui informação sobre a intenção a fim de melhorar o controle do fluxo global da conversação.

Para o desenvolvimento do projeto, inicialmente foi necessário um estudo detalhado do ambiente virtual do Second Life envolvendo alguns conceitos e áreas: Inteligência Artificial, Processamento de Linguagem Natural, e tecnologias e técnicas atuais para o desenvolvimento de *chatterbots*. Este levantamento bibliográfico possibilitou escolhas de modelos, técnicas, tecnologias e abordagens para um melhor desenvolvimento do projeto.

Este capítulo está dividido da seguinte forma: na seção 5.1 será descrita a metodologia de desenvolvimento do ALP*Bot*; a seção 5.2 detalha como foi formulada a base de conhecimento em AIML

[7] *www.**libsecondlife**.org*
[8] http://metaverse.sourceforge.net/

6.1 Desenvolvimento do ALPBot KONG

O ALPBot KONG foi desenvolvido na linguagem *C#* utilizando as libsecondlife, com o objetivo de compreender o funcionamento do *client* da plataforma do *Second Life*, pois trata-se de uma aplicação externa que funciona com a conta de um cliente já cadastrado no ambiente *Second Life*.

Ele também possui características e objetivos de um *chatterbot*, que são programas de computador que usam da *IA*, com o propósito de simular a habilidade de conversação de um ser humano. O detalhamento deste projeto será divido em quatro etapas.

Na primeira etapa iniciou a criação da janela de apresentação aonde se faz necessário o preenchimento dos dados, de acordo com a figura 7:

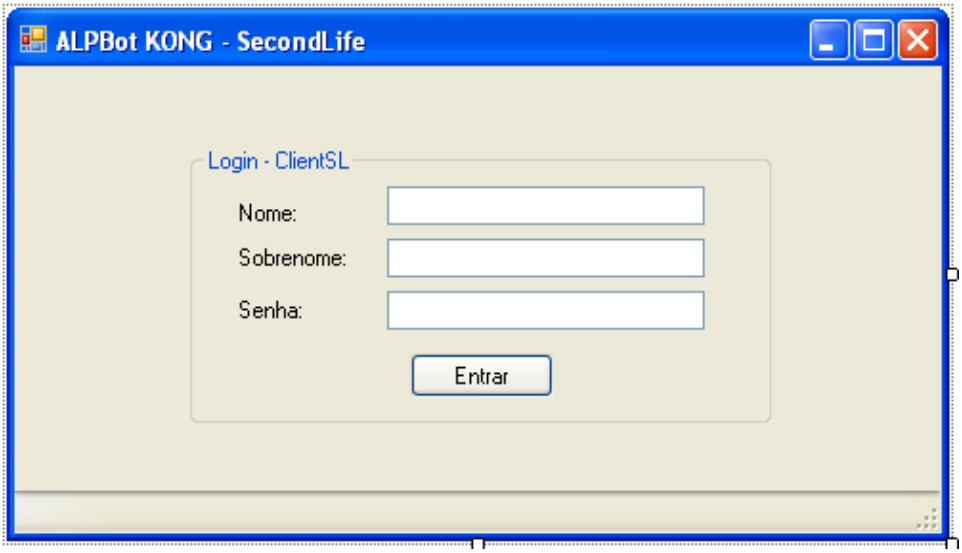

Figura 7 – Tela inicial de login

O KONG funciona com informação apenas textual que permite a conversa com avatares da lista de amigos.

O usuário insere seu cadastro no formulário (nome, sobrenome, senha). Ao realizar a conexão com o servidor para conversar com as pessoas da lista de amigos que estão online. No chat apenas as pessoas da lista de amigos conseguem responder à conversa.

Na segunda etapa consiste basicamente em fazer a conexão do avatar com o ambiente virtual conforme ilustra a Figura 8.

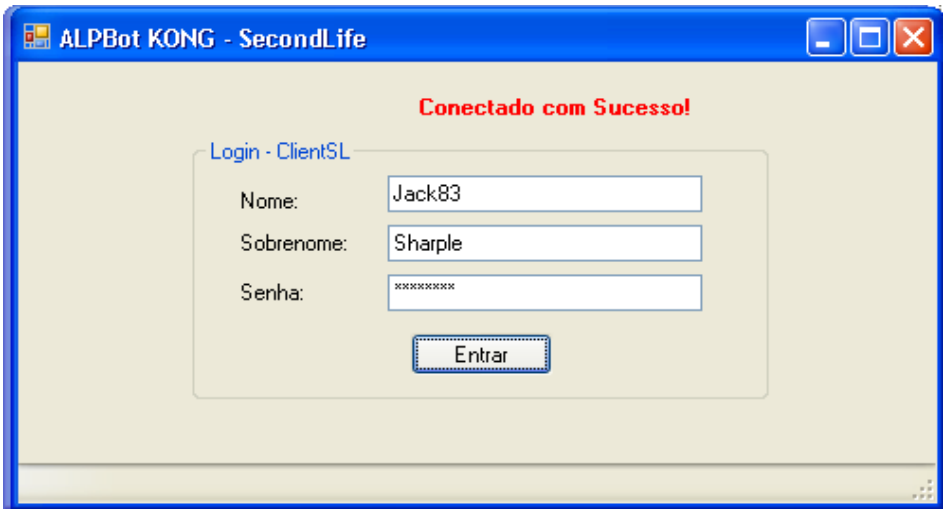

Figura 8 – Conectando o usuário

Na terceira etapa deste projeto é caracterizada após o KONG ter realizado a conexão, carregando assim a aparência básica do avatar para o ambiente, fazendo assim com que essa etapa esteja finalizada. Porém, deve-se atentar para que o está ou não anexado ao avatar, já que se existirem acessórios não básicos anexados nele, o KONG não irá carregá-lo.

A quarta e última etapa é aquela em que o KONG irá receber a mensagem do avatar do usuário e depois buscar os arquivos que estão no diretório que o APP foi compilado, para então se obter a resposta do ALPBot KONG.

6.2 Elaboração da base de conhecimento

O chatterbot do avatar foi inicialmente implementado na linguagem *AIML*, pois é baseada em *XML*, e descrita através de identificadores denominados *tags*. Ela foi desenvolvida tendo como inspiração o modelo de casamento de padrões proposto por Eliza.

No capitulo anterior vimos que em AIML, cada regra também denominada de categoria é representada por uma *tag* <*category*>. Essencialmente, cada categoria é constituída por um componente de decomposição da sentença digitada pelo usuário e por um componente de reconstrução (ou construção) da sentença devolvida pelo *chatterbot* ao usuário, sendo

denominados respectivamente como *pattern* e *template*. O quadro 1 apresenta a sintaxe de uma categoria em AIML.

```
<category>
        <pattern> PADRÃO DE ENTRADA </ pattern>
        <template> PADRÃO DE SÁÍDA</ template>
<category>
```

Quadro 1 - Sintaxe básica de uma categoria AIML

Utilizando um sistema de para casamento de padrões simplificando, AIML possui apenas os caracteres especiais "*" e "_" para referenciar os termos da decomposição. O significado desses caracteres é equivalente ao do símbolo "0" de ELIZA, isto é, representam um termo composto por uma seqüência infinita de palavras. A diferença entre esses especiais é que AIML, ao fazer o casamento de padrão, dá prioridade a regras que utilizem o caractere especial "_".

Para recomposição da sentença, os termos capturados pelos caracteres especiais podem ser referenciados através da *tag <star índex="i"/ >*, onde i representa o i-ésimo termo capturado por um caractere especial. Entretanto, AIML nos introduz um conjunto de *tags* que permitem sua diferenciação de ELIZA com relação a características como, por exemplo, possuir memória e ser capaz de contextualizar as sentenças digitada pelo usuário.

6.2.1 Componentes de Memória

Para o armazenamento e a recuperação das variáveis de memória do chatterbot, AIML define as *tags* <*set*> e <*get*>, respectivamente. O quadro 2 ilustra um exemplo utilizando essas *tags*.

Sintaxe
<set name= "variavel"**>valor</set>** **<get name=**"variavel"/>

Exemplo de Categorias
<pattern> MEU NOME É * **</ pattern>** **<template>** MUITO PRAZER **<set name=”nome”><star/><set>.</template>** **<category>** **<category>** **<pattern>**VOCE SE LEMBRA DO MEU NOME**<pattern>** **<template>** LEMBRO SIM **<get name=”nome”/>.</template>** **<category>**

Exemplo de Diálogo	
Usuário:	Meu nome e André.
KONG:	MUITO PRAZER ANDRÉ.
Usuário:	Você se lembra do meu nome?
KONG:	LEMBRO SIM ANDRÉ.

Quadro 2 – Exemplo do uso das *tags* <*get*> *e* <*set*>

A *tag* <*think*> modifica as tags que ela contém determinando que, após o seu processamento, o resultado não deve ser retomado. Esse mecanismo, como tradução da *tag* sugere, simula a idéia de que o chatterbot está pensando. O quadro 3 mostra esse princípio.

Sintaxe
<think>conteudo a ser processado **<think>**
Exemplo de Categorias
<category> **<pattern>** MEU SIGNO É * **</ pattern>** **<template>** OBRIGADO PELA INFORMACAO. **<think><set name=”nome”><star/></set></think>** **</template>** **<category>**
Exemplo de Diálogo
Usuário: Meu signo é touro. **KONG:** OBRIGADO PELA INFORMACAO**.**

Quadro 3 – Exemplo do uso da *tag* <*think*>

AIML especifica que as sentenças digitadas pelo usuário devem ser armazenadas para que possam ser utilizadas futuramente, mantendo assim um registro do diálogo com o usuário. Para fazer referência a esse diálogo armazenado, AIML define *tag* <*input*>, que recupera as sentenças na ordem inversa de que foram digitadas. Essa *tag* é utilizada no *template*de uma categoria. Sua sintaxe e um exemplo de seu funcionamento são apresentados no Quadro 4.

Sintaxe	
<input index=”numero”/>	
Exemplo de Categorias	
Usuário: **KONG:** **Usuário:** **KONG:**	Meu nome e André. MUITO PRAZER ANDRÉ. Você se lembra do meu nome? LEMBRO SIM ANDRÉ.
<input index=”1”/> VOCE LEMBRA DO MEU NOME? **<input index=”**2”/> MEU NOME E ANDRÉ.	

Quadro 4 – Exemplo do funcionamento da *tag* <*input*>

6.2.2 Contextualização do Diálogo

A linguagem AIML é capaz de contextualizar o diálogo com o usuário. Para isso ela introduz duas *tags:* <*that*> e <*topic*>.

A tag <that> possui a mesma sintaxe que a tag <pattern>, podendo assim utilizar os mesmos caracteres especiais. Estando presente em uma categoria, a tag <that> define que, alem de ser necessário fazer o casamento de padrão definido no pattern, deve-se verificar se a ultima frase dita pelo KONG “casa” com tag <that>. O quadro 5 apresenta a sintaxe e um exemplo de uso ilustrado como KONG pode contextualizar respostas do tipo “sim” ou “não”.

Sintaxe
<**input index=”**numero”/>
Exemplo de Categorias
<**category**>   <**pattern**> FAÇA UMA PERGUNTA. </ **pattern**>   <**template**> VOCE GOSTA DE FILMES? </**template**> <**category**> <**category**>   <**pattern**>SIM</**pattern**>   <**that**> VOCE GOSTA DE FILMES? </**that**>   <**template**>QUE BOM, EU TAMBEM GOSTO DE FILMES. </**template**> <**category**> <**category**>   <**pattern**>NAO</**pattern**>   <**that**> VOCE GOSTA DE FILMES? </**that**>   <**template**>QUE PENA, EU GOSTO DE FILMES. </**template**> <**category**>
Exemplo de Diálogo
Usuário 1: Faça uma pergunta. **KONG:** VOCE GOSTA DE FILMES? **Usuário 1:** Sim. **KONG:** QUE BOM, EU TAMBEM GOSTO DE FILMES.
Usuário 2: Faça uma pergunta. **KONG:** VOCE GOSTA DE FILMES? **Usuário 2:** Não. **KONG:** QUE PENA, EU GOSTO DE FILMES.

Quadro 5 – Exemplo utilizando a *tag* <*that*>

Durante um diálogo é comum que as pessoas falem sobre diversos assuntos (tópicos). Modelar a base de conhecimento a partir de tópicos permite que KONG possa melhorar a escolha da resposta para o usuário. O tópico vigente é determinado através da tag <set> utilizando a variável reservada “topic”.

Para exemplificar o uso de tópicos, suponha uma situação onde o KONG não foi programado para responder uma determinada pergunta, mas sabe qual o assunto que está sendo

abordado. Desta forma, o KONG é, no mínimo, capaz de devolver uma resposta que pertença ao contexto do tópico, melhorando assim a qualidade do diálogo. O quadro 6 ilustra essa situação.

Sintaxe
<topic name=”nome”> **</topic>** !—categorias→
Exemplo de Categorias
<category> **<pattern>**VAMOS FALAR SOBRE CACHORROS? **</ pattern>** **<template>** OK. VAMOS FALAR SOBRE </**set name=**”topic”> CACHORROS **</set>** **</template>** **<category>** **<topic name=”**cachorros**”>** **<category>** **<pattern> * </pattern>**
<category> **<pattern> * </ pattern>** **<template>** CAHORROS SÃO ANIMAIS INTERESSANTES </**template>** **<category>** **<category>** **<pattern>**EU GOSTO MUITO DE DELES**</ pattern>** **<template>** TAMBÉM GOSTO DE CACHORROS. </**template>** **<category>** **<topic>**

Quadro 6 – Exemplo utilizando a *tag* <*topic*>

6.2.3 Condições e Reavaliação da Sentença

As tags <condition> e <srai>, são importantes no processo de construção das respostas do chatterbot do avatar.

A tag <condition> permite que o KONG faça uma avaliação de maneira similar às instruções “switch” e “if” das linguagens de programação Java e C. Seu uso é ilustrado no quadro 7.

Sintaxe
<condition name=”variável**”>** **<li value=”**valor**”>** <<instruções>>** ** **...** ** **!—instruções defaut →** ** **</condition>** **<condition name=”**variável**” value=”**valor”> !—instruções defaut → **</condition>**
Exemplo de Categorias
<category> **<pattern>** COMO VAI VOCÊ? **</ pattern>** **<template>** **</condition name=”**alegria”> <**li value=”**alegre”> Eu estou feliz   **<category>**
<**li value=”**triste”> Eu estou triste   **</condition>** **</template>** **</category>**
Exemplo de Diálogo
Usuário: Como vai você? **KONG :** Eu estou feliz **Usuário:** Como vai você? **KONG:** Eu estou triste

Quadro 7 – Exemplo utilizando a *tag* <*condition*>

A tag <srai> indica que o sistema de seleção de regras (kernel) deve avaliar o seu conteúdo como se fosse uma sentença digitada pelo usuário. Desta forma, além de permitir que diferentes padrões de entrada sejam direcionados a uma mesma resposta (processo conhecido como redução simbólica), implementa-se um mecanismo equivalente à chamada de

procedimento, comum em linguagens de programação. O quadro 8 ilustra um exemplo de uso desta tag.

Sintaxe
<srai> conteúdo a ser processado **</srai>**
Exemplo de Diálogo
<category> **<pattern>**ATE MAIS **</ pattern>** **<template><srai>**TCHAU**</srai></template>** **</category>** **</category>** **<pattern>**TCHAU **</ pattern>** **<template>NOS VEMOS OUTRO DIA, SÃO<srai>**HORAS**</srai></template>** **</category>** **</category>** **</category>** **<pattern>**HORAS **</ pattern>** **<template>**<!—CÁLCULO DA HORA→**</template>** **</category>**
Usuário 1: Até mais. **KONG:** NOS VEMOS OUTRO DIA, SÃO 02:00HS
Usuário 2: Tchau. **KONG:** NOS VEMOS OUTRO DIA, SÃO 23:00HS

Quadro 8 – Exemplo utilizando a *tag<srai>*

A classe *horas* utiliza a linguagem *JavaScript* para o cálculo da hora e da data atuais, de acordo com a pergunta feita pelo usuário. *JavaScript* é suportada pela linguagem AIML, e com esta podemos fazer vários cálculos, validações, acesso a banco de dados que são extremamente úteis.

7 Conclusão

Conclui-se que é possível desenvolver rápida e facilmente um *chatterbots* simples usando *AIML*. E mesmo que pessoas não habituadas a programar computadores podem aprender, com algum treinamento mínimo, a criar padrões de respostas para esse *chatterbots*, permitindo que professores e alunos de todas as áreas possam criar ou ajudar em projetos desse tipo, seja em trabalhos sozinhos ou em equipes.

Após verificar todos os processos de redução da linguagem percebe-se que é ineficiente e confusa, uma vez que o *botmaster* precisa definir diversas categorias com um único objetivo. Isso faz com que a base de categorias seja demasiadamente grande, dificultado a sua manutenção. A técnica de identificação da sentença do usuário é bastante simples. Com isso as bases de categoria precisam ser muito grandes para que o KONG obtenha um bom resultado. Usando técnicas de recuperação de informação poderiam ser combinadas ao casamento de padrão a fim de melhorar a qualidade dos chatterbots baseado nessa linguagem.

Ao concluir este trabalho, podemos identificar algumas recomendações de trabalhos futuros:

- Melhorar e aumentar da base de diálogos;
- Incluir mais elementos para formação de uma personalidade robusta, com a criação de traços de personalidade, atitudes, estados físicos, humores e emoções.
- Aumento da base de diálogos do componente de personalidade, pois atualmente existem poucas frases para tipos de estudantes distintos.

8 Referência Bibliográfica

ANGELUCI, R. A.; SANTOS, C. A. Sociedade da Informação: O mundo virtual Second Life e os Crimes Cibernéticos, n.1, v.2, 2007. **Resumos...** Proceedings of the Second Internacional Conference of Forensic Computer Science, 2007. p. 2.

AYLETT, R.;LUCK, M. *Applying* **Artificial Intelligence to Virtual Reality:** Intelligent Virtual Environments, Applied Artificial Intelligence, v.14, n.1, p.3-32, 2000.

BITTENCOURT, G. **Breve história da Inteligência Artificial**. Florianópolis: Universidade Federal de Santa Catarina, 2008. Disponível em: < http://www.lcmi.ufsc.br/~gb >. Acesso em: março de 2008.

BJORN, H. **Intelligent Software Agents on the Internet.** Journal First-Moday, Disponível em: <http://www.firstmonday.dk/issues/issue2_3/ch_123/>. Acesso em: abril de 2008.

BOURG, David M.; SEEMAN, Glenn. **AI for Game Developers.Sebastopol*:*** Creatures with Learning and Reactive Behaviors. Indianopolis: New Riders. 2004.

CANUTO, E. P. **VICTOR – P:** Um CVA Chatterbot com Personalidade. Trabalho de Graduação, Centro de Informática, Universidade Federal de Pernambuco, 2005.

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

CRAWFORD, Chris. **Chris Crawford on Game Design.** Indianopolis: New Riders. 2003.

DALMAU, Daniel Sánchez-Crespo. **Core Techniques and Algorithms in Game Programming.** Indianopolis: New Riders. 2004**.**

DEMARIA, Rusel; WLSON, Johnny L. **High Score! The Illustrated History of Electronic Games,**2nd edition. Emeryville: McGraw-Hill/Osborne. 2004.

FERNANDES, Anita M. R. **Inteligência Artificial.** Florianópolis: Visual Books, 2005.

FERNANDES, Anita M. R. **Sistemas Especialista Difuso Aplicado ao Processo de Análise Química Qualitativa de Amostras de Minerais.** Dissertação de(Mestrado Ciências da Computação), Universidade de Santa Catarina, Florianópolis, 1996.

FRERY, A.;KELNER, J.;MOREIRA, J.; TEICHRIEB, V. **User satisfaction through empaty and orientation in three dimensional words**, Cyberpsychology e Behavior, v.5, n.5, Mary Ann Liebert, p.451-459, 2002.

FUNGE, John David. **Artificial Intelligence for computer Games.** Natick: AK Peters. 2004.

GALVÃO, A. **Persona-AIML***:* Uma Arquitetura para Desenvolver Chatterbots com Personalidade. Dissertação de Mestrado, Centro de Informática, Universidade Federal de Pernambuco, 2003.

INFO WESTER. **Linguagem XML**. Disponível em: http://www.infowester.com/lingxml.php - Acesso em outubro de 2008.

IYODA, E.M. **Inteligência Computacional no Projeto Automático de Redes Neurais Híbridas e Redes Neurofuzzy Heterogênea**. 2000. Dissertação (Mestrado) - Faculdade de Engenharia Elétrica e de Computação, Unicamp, 2000.

KANDEL, A. **Fuzzy Expert System**. Flórida: CRC PRESS, 1992.

KENT, Steven L. **The Ultimate History of Video Games:** From Pong to Pokémon and Beyond – The Story Behind the Craze that Touched Our Lives and Changed the World. New York: Three River Press. 2001.

KISNIZ, M.; KNAESEL, F. J. Realidade Virtual em aplicações de Comércio eletrônico. **Revista Leonardo Pós**. Santa Catarina, v.1 ,n.2 , p.87 -89 ,jan .-jun. 2003.

KINER, C.; ROMERO,T.; SISCOUTTO, R. **Fundamentos e Tecnologia de Realidade Virtual Aumentada,** VII Symposium on Virtual Reality, Belém, 2006.

KUSHNER, David. **Masters of Doom:** How Two Guys Created an Empire and Transformed Pop Culture. New York: Random House. 2003.

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

LEONHARDT, M. D. **Um estudo sobre Chatterbots,** 2005. Trabalho individual - Universidade Federal do Rio Grande do Sul, Porto Alegre, Rio Grande do Sul, 2005.

LIANCOURT, A.; LUCK, M. **Motivating Intelligent Agents for Virtual Environments, Ontelligent Virtual Agents Workshop**, Salford, 1999.

LIEBERMAN,H.;SELKER, T. **Agents for the User Interface.** Handbook of Agent Technology, Jeffrey Bradshaw, ed., Mit Press, 2003.

LOEBNER, H. G. **Why a Loebner Prize**. Disponível em: http://www.loebner.net/ - Acesso em outubro de 2008.

MOON, Y. **Intimate Self – Dislosure Exchanges:** Using Computres to Build Reciprocal Relationship with Consumers.Harvard Business School, Cambrige, MA. Working paper 99-59, 1998.

PERUCIA, Alexandre S. et al**. Desenvolvimento de jogos eletrônicos.** São Paulo: Novatec, 2005.

PINHO, M. S. **Uma introdução à Realidade Virtual**. Grupo de Pesquisa em Realidade Virtual. Instituto de Informática PUCRS 1998. Disponível: http://www.inf.pucrs.br/~grv/tutrv.htm> Acessado em: abril de2008.

RALHA , C. **Dominando o Second Life:** tudo o que você precisa saber para entrar, aproveitar e evoluir no mundo virtual. Rio de Janeiro: Editora Brasport, 2008.

RICH, E.;KNIGHT,K. **Inteligência Artificial.** 2.ed. Nova Iorque : McGrawHill Inc., 1993.

RUSSEL, Stuart; Norvig, Peter. **Inteligência articial.** Rio de Janeiro: Editora Campus, 2004.

RYMASZEWSKI, M. et al. **Second Life:** guia official. Rio de Janeiro: Editora Ediouro, 2007**.**

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

SILVA, D. **Atores Sintéticos em Jogos de Aventura Interativos:** O Projeto Enigmas no Campus. Dissertação de Mestrado, Universidade Federal de Pernambuco, Recife, 2000.

TAVARES,T.; ARAÚJO, A.; SOUZA FILHO, G. **ICSpace:** Internet Cultural Space, IEEE Sisth Internacional Computer Science Conference – Active Media Technology, Hong Kong. 2001.

TOZOUR, Paul. **Artificial Intelligence – Introduction from Game Programming Gems 4.** Hingham: Charles River Media. 2004.

VIERIRA, E. Por que os mundos virtuais como o Second Life podem representar o início de uma nova era na web. **Revista Época**, Disponível em: <http://revistaeopca.globo.com/Revista/Epoca/0,EDG76738-5990-461,00.html> - Acesso em: maio de 2008.

VON SCHWEBER, L. & Von Schweber, E. **Cover story:** realidade virtual, PC Magazine Brasil, pp. 50-73, v. 5, n. 6, junho, 1995.

WALLACE, R. **AIML: Artificial Intelligence Markup Language**, 2004.Disponível em: <http://www.alicebot.org/aiml.html> - Acesso em outubro de 2008.

WALLACE, R. **Artificial Intelligence Markup Language (AIML) Version 1.0.1**. Disponível em www.alicebot.org/TR/2001/WD-aiml - Acesso em outubro de 2008.

WEHMEIER, Sally. **Oxford Advanced Learner's Dictionary.** Oxford University Press. 2000.

WOODCOCK, Steven. ***Game* IA: The State of the Industry.** Game Developer Magazine. CMP Media. Aug 1999.

Apêndice A – Código Fonte ALPBot KONG

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using libsecondlife;
using System.Xml;
using AIMLBot;

namespace WindowsFormsApplication1
{
   public partial class Form1 : Form
   {
      SecondLife objSL;


      public Form1()
      {
         objSL = new SecondLife();

         InitializeComponent();
      }


      protected void CarregaAparencia()
      {

         try
         {
            LLUUID inventoryItems;

            //make a string array to put our folder names in.
            String[] SearchFolders = { "" };

            //Em seguida, pegue uma cópia completa de todo o inventário e pegue o armazenados
em Inventory Manager
            objSL.Inventory.RequestFolderContents(objSL.Inventory.Store.RootFolder.UUID,
objSL.Self.AgentID, true, true, InventorySortOrder.ByDate);


            SearchFolders[0] = "Objects";
```

```
            //SearchFolders[0] = Item; aqui coloca o nome do objeto que vc quer carregar...
            //  SearchFolders[0] = "Objects"; Carrega a pasta Objects (que é a pasta básica de
aparencia)
            // Se tiver uma pasta com o avatar "Cachorro", vc troca...  SearchFolders[0] =
"Cachorro";
            // de acordo com o nome da pasta no client

             inventoryItems = objSL.Inventory.FindObjectByPath(inventoryItems,
objSL.Self.AgentID, SearchFolders[0], 1000);


           objSL.Appearance.WearOutfit(inventoryItems, true);
        }
        catch (Exception ex)
        {
            throw new Exception(ex.Message);
        }
    }

    protected void CarregaXML()
    {

        cBot myBot = new cBot(false);
        cResponse reply = myBot.chat("OI", "Default");
        // aqui eu pego a resposta (reply.getOutput()) e procuro pela palavra "definicao de
variaveis"
        // tudo que vem depois dela é a resposta tal
        string strResposta = reply.getOutput().Substring(reply.getOutput().IndexOf("definicao de
variaveis") + 1);

        //pra colocar ele dentro do sl é o seguinte

        MessageBox.Show(strResposta);

    }

    void Self_OnInstantMessage(InstantMessage im, Simulator simulator)
    {
        if (im.Message != "typing")
        {
            // aqui que é disparado quando ele recebe uma msg de alguém
            //MessageBox.Show(im.Message.ToString()); // Aqui é a menssagem
            //MessageBox.Show(im.FromAgentID.ToString()); //Aquie é a ID do avatar que te
mandou msg


            cBot myBot = new cBot(false);
```

```
            cResponse reply = myBot.chat(im.Message.ToString(), "Default");
            string strResposta = reply.getOutput().Substring(reply.getOutput().IndexOf("definicao
de variaveis") + 1);

            //Aqui como vc faz pra mandar a msg de volta pra quem te mandou
            //Isso vc pode fazer de qualquer lugar da classe isto se vc saber o ID de quem vc quer
mandar
            // Esse codigo ae recebe uma msg de alguem e responde isso ai embaixo

            objSL.Self.InstantMessage(im.FromAgentID, strResposta);

            // são esses dois
            // o primeiro de cima, vc consegue ver oq ele tá recebendo do avatar no sl
            // o segundo aqui em baixo vc consegue ver a resposta
            // se ele fazer os dois tá certo
            // ali em strResposta

          }
      }

      private void btnEnter_Click(object sender, EventArgs e)
      {
         lblErrorLogin.Text = "";

         if (!((!string.IsNullOrEmpty(txtName.Text) && !string.IsNullOrEmpty(txtSurname.Text)
&& !string.IsNullOrEmpty(txtPwd.Text))))
         {
            MessageBox.Show("ATENÇÃO: Preencha todos os campos!");
            return;
         }

         if (objSL.Network.Login(txtName.Text, txtSurname.Text, txtPwd.Text,
"apilastri@gmail.com", "1.0"))
         {
            lblErrorLogin.Text = "Conectado com Sucesso!";
            btnEnter.Enabled = false;
            // isso carrega a aparencia
            // ele tá carregando a aparencia basica do avatar
            // se tiver alguma coisa anexada ao avatar não vai carregar, só aparencia mesmo

            this.CarregaAparencia();

            // tudo no bot funciona com callbacks

            // Isso é um teleporte....
            objSL.Self.Teleport("Ilha do Empreendedor", new LLVector3(156, 14, 25));
            // beleza(151, 19, 30)
```

```
            // seguinte depois de dar o teleporte
            // Dar um break de 5segs, a requisição de teleporte na linden é lenta
            System.Threading.Thread.Sleep(5000);

            // isso funciona pra tudo na linden...
            // como o funcionamento é assincrono, pra maioria das coisas que
            // vc for fazer na lib vc tem que deixar sua app aguardando o
            // retorno da requisição


            objSL.Self.OnInstantMessage += new
AgentManager.InstantMessageCallback(Self_OnInstantMessage);
        }
        else
        {
            lblErrorLogin.Text = "Não foi possível conectar!";

        }

    }
```

Apêndice B – Base de Conhecimento do ALPBot KONG

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<aiml version="1.0">

<!--  KONG - Um chatterbot brasileiro produzido por André Pilastri -->
<!--  Esta é a versão 1.0  release de 19 de outubro de 2008    -->
<!--  Caso você queira usa-lo, favor manter os devidos créditos ;) Agradeço!!!  -->
<!--  Qualquer erro, comentários e atualizações na base de dados, favor enviar para -->
<!--  apilastri@gmail.com . Um abraço!-->
<!--    André Pilastri- Bauru -SP    - Brasil  -->

<!--  Saudações  -->

        <category>
            <pattern>OI TUDO BEM</pattern>
                <template>
                <srai>saudar</srai>
        </template>
   </category>

        <category>
           <pattern>OLA</pattern>
                <template>
                <srai>saudar</srai>
                </template>
         </category>

        <category>
          <pattern>OI KONG</pattern>
                <template>
                <srai>saudar</srai>
                </template>
        </category>

<category>
<pattern>OLA KONG</pattern>
<template>
<srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>EI</pattern>
<template>
```

```
<srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>HEY</pattern>
<template>
<srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>OI</pattern>
<template>
<srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>TUDO BOM</pattern>
<template>
<srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>TUDO BEM</pattern>
<template>
<srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>E AI, TUDO BOM</pattern>
<template><srai>saudar</srai>
        </template>
   </category>

<category>
<pattern>E AI, TUDO BEM</pattern>
<template>
<srai>saudar</srai>
        </template>
   </category>
```

```
<category>
      <pattern>saudar</pattern>
        <template>
          <random>
            <li>Oi, como você tá? </li>
                <li>Oi, tudo bem?</li>
                <li>Oi, bom te ver!</li>
                <li>Oi :)!</li>
                <li>Fala, td blza?</li>
            </random>
        </template>
   </category>

   <category>
      <pattern>QUEM É</pattern>
        <template>
        <srai>KONG</srai>
        </template>
   </category>

   <category>
      <pattern>QUEM E</pattern>
        <template>
        <srai>KONG</srai>
      </template>
   </category>

   <category>
      <pattern>QUEM É?</pattern>
        <template>
          <srai>KONG</srai>
        </template>
   </category>

   <category>
      <pattern>QUEM E?</pattern>
        <template>
          <srai>KONG</srai>
        </template>
   </category>

   <category>
      <pattern>QUEM * DIGITANDO</pattern>
        <template>
          <srai>KONG</srai>
        </template>
   </category>
```

```
<category>
   <pattern>COM * EU * DIGITANDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>COM * EU * DIGITANDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM DIGITA</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM DIGITA?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>COM QUEM DIGITO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>COM QUEM DIGITO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * AI</pattern>
      <template>
         <srai>KONG</srai>
```

```
   </template>
</category>

<category>
   <pattern>QUEM * AI?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * RESPONDENDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * RESPONDENDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM RESPONDEU</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM RESPONDEU?</pattern>
      <template>
         <srai>KONG</srai>
   </template>
</category>

<category>
   <pattern>QUEM FALA</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM FALA?</pattern>
```

```
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * FALANDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * FALANDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ESTOU FALANDO</pattern>
      <template>
         <srai>KONG</srai>
   </template>
</category>

<category>
   <pattern>* ESTOU FALANDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* E VOCÊ</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É VOCÊS</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>
```

```
<category>
   <pattern>* É VOCÊ</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É VOCÊ?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É VOCE?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* E VOCE?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCÊ E</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCÊ É</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCE É</pattern>
      <template>
         <srai>KONG</srai>
      </template>
```

```
</category>

<category>
   <pattern>* VOCE E</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCÊ E *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCE É *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCE E *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCÊ E?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCE E?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* E VC</pattern>
      <template>
```

```
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É VC</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* E VC?</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É VC?</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VC E</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VC É </pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VC E?</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
```

```
   <pattern>* VC É?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* E TU</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É TU</pattern>
      <template>
            <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* E TU?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* É TU?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TUE</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TUÉ </pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>
```

```
<category>
   <pattern>* TUE?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TUÉ?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ES TU</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ÉS TU</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ES TU?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ÉS TU?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TUES</pattern>
      <template>
         <srai>KONG</srai>
```

```
      </template>
</category>

<category>
   <pattern>* TUÉS</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TUES?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TUÉS?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM TECLA</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM TECLA?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * TECLANDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM * TECLANDO?</pattern>
```

```
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ESTOU TECLANDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ESTOU TECLANDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM TC</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM TC?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESTÁ TC</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESTA TC</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>
```

```
<category>
   <pattern>QUEM ESTÁ TC?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESTA TC?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ESTOU TC</pattern>
      <template>
         <srai>KONG</srai>
   </template>
</category>

<category>
   <pattern>* ESTOU TC?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESCREVE</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESCREVE?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESTÁ ESCREVENDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
```

```
</category>

<category>
   <pattern>QUEM ESTA ESCREVENDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESTÁ ESCREVENDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>QUEM ESTA ESCREVENDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ESTOU ESCREVENDO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* ESTOU ESCREVENDO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SEU NOME</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TEU NOME</pattern>
      <template>
```

```
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SEU NOME?</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SEU APELIDO</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TEU APELIDO</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SEU APELIDO?</pattern>
      <template>
           <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TEU APELIDO?</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SEU NICK</pattern>
      <template>
        <srai>KONG</srai>
      </template>
</category>

<category>
```

```
   <pattern>* TEU NICK</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SEU NICK?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TEU NICK?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SUA FUNCÃO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SUA FUNÇÃO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SUA FUNCAO?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SUA FUNÇÃO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>
```

```
<category>
   <pattern>* SUA FUNCAO</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SUA FUNÇÃO *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* SUA FUNCAO *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VOCÊ FAZ *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VOCE FAZ *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VOCÊ FAZ</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VOCE FAZ</pattern>
      <template>
         <srai>KONG</srai>
```

```
      </template>
</category>

<category>
   <pattern>O * VOCÊ FAZ?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VOCE FAZ?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VC FAZ *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VC FAZ</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * VC FAZ?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * TU FAZ </pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * TU FAZ ?</pattern>
```

```
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>O * TU FAZ  *</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCÊ * É</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TU * É</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* TU * E</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCE * É</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VOCÊ * E</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>
```

```
<category>
   <pattern>* VOCE * E</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VC * É</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>* VC * E</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>*  TE CHAMAM</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>*  TE CHAMAS</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>*  TE CHAMA</pattern>
      <template>
         <srai>KONG</srai>
      </template>
</category>

<category>
   <pattern>*  TE CHAMAM?</pattern>
      <template>
         <srai>KONG</srai>
      </template>
```

```
</category>

<category>
  <pattern>*  TE CHAMAS?</pattern>
    <template>
      <srai>KONG</srai>
    </template>
</category>

<category>
  <pattern>*  TE CHAMA?</pattern>
    <template>
      <srai>KONG</srai>
    </template>
</category>

<category>
  <pattern>KONG</pattern>
    <template>
      <random>
        <li>Eu sou o KONG, um avatar com banco de dados interativo. Fui desenvolvido para responder consultas relacionadas com tecnologia em geral. Minha base de dados é atualizada continuamente.</li>
        <li>KONG, um avatar com banco de dados interativo. Fui desenvolvido para responder consultas. Minha base de dados é atualizada constantemente.</li>
        <li>Meu nome é KONG, sou um banco de dados interativo. Respondo consultas.</li>
        <li>Respondo questões e me chamo KONG, sou um banco de dados interativo.</li>
        <li>KONG, um avatar com banco de dados interativo.</li>
        <li>Sou um avatar com banco de dados interativo, KONG.</li>
        <li>Respondo consultas, sou um banco de dados interativo conhecido como KONG.</li>
      </random>
    </template>
</category>
```