UNIVERSIDADE FEDERAL DO PARÁ
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Desenvolvimento e Animação de Atores Sintéticos Articulados – Uma Simulação Futebolística

Wanderson Alexandre da Silva Quinto

Belém – Pará
09/12/2005

UNIVERSIDADE FEDERAL DO PARÁ
CENTRO TECNOLÓGICO
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Desenvolvimento e Animação de Atores Sintéticos Articulados – Uma Simulação Futebolística

Dissertação apresentada em 09/12/2005 para obtenção do título de Mestre em Ciências do Programa de Pós-Graduação em Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Pará, na área de concentração de Computação Aplicada, cuja banca examinadora foi composta pelos seguintes membros:

_____

Prof. Manoel Ribeiro Filho, Dr. (UFPA).
Orientador

_____

Prof. Bianchi Serique Meiguins, Dr. (UFPA)
Membro

_____

Prof. Rodrigo Quites Reis, Dr. (UFPA)
Membro

_____

Prof. Elói Luiz Favero, Dr. (UFRN)
Membro

Visto:

Prof. João Crisóstomo Weyl A. Costa, Dr. (UFPA)

Coordenador do PPGEE – CT – UFPA

Universidade Federal do Pará
Centro Tecnológico
Programa de Pós-Graduação em Engenharia Elétrica

Wanderson Alexandre da Silva Quinto

Desenvolvimento e Animação de Atores Sintéticos Articulados – Uma Simulação Futebolística

DISSERTAÇÃO APRESENTADA PARA OBTENÇÃO DE MESTRADO NO PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA DA UNIVERSIDADE FEDERAL DO PARÁ

Belém
2005

SUMÁRIO

ÍNDICE DE FIGURAS

ÍNDICES DE TABELAS

Agradecimentos

À Deus que sempre iluminou e guardou meus caminhos.

Aos meus parentes principalmente meus pais José Francisco Chagas da Silva Quinto (Cabeção), Nelcy Silva Quinto e minha irmã, por me amarem, me apoiarem, incentivarem e acreditarem em mim.

À minha esposa Priscila Quinto que é mais maluca do que eu, pois aceitou mudar de vida por duas vezes só pra me ver estudar e progredir na carreira docente, te amo.

À minha filha Beatriz que chegou este ano para alegrar ainda mais minha vida, aumentar a torcida do leão azul e ser meu maior tesouro.

Ao meu sogro e sogra por me acolherem, e a minha cunhada por ser chata mais que eu amo demais e meu cunhado por ser um verdadeiro batalhador.

Aos amigos/ irmãos Atila Soares, Allan Sanders, Rosevaldo Dias, Marcelo Brito por serem parte da minha nova família que começou no mestrado, por serem remistas e não desistirem nunca, por me ajudarem nos momentos difíceis e foram muitos, valeu!!

Aos amigos Renato Simões, Armando Hage e Denis Neves por me ajudarem imensamente no desenvolvimento deste trabalho.

Ao professor Manoel Ribeiro Filho meu orientador, que teve paciência para agüentar todos os meus erros.

Ao eterno prof. Afonso Cardoso por me fazer acreditar que seria possível largar tudo em Rondônia e vir cursar o mestrado.

Ao Clube do Remo por toda a alegria proporcionada durantes estes anos todos. REMO TU ÉS MINHA VIDA, TU ÉS MINHA HISTÓRIA, TU ÉS MEU AMOR.

Aos meus amados cachorros Álamo, Laila, Laika, Pequenininha, Pit Bull (Cachorro Amarelo), Pretinha, Pit, Tati, Tita, Ticha (Vaca), Tobias, Xuxo, Zort.

À todos que ajudaram de alguma forma.

OBRIGADO.

RESUMO

Esta dissertação de mestrado tem por objetivos construir uma ambiente virtual 3D na forma de um estádio de futebol onde serão realizadas simulações de um jogo de futebol, com o uso de avatares autônomos. Para a construção do avatar utilizou-se a técnica de modelagem H – Anim 1.1 que traz o conceito de corpo hierarquizados e articulados. Toda a implementação da aplicação foi desenvolvida com Java e API Java 3D. A autonomia é conseguida por auxílio da Inteligência Artificial (IA), que atuará na solução dos problemas juntamente com a utilização de regras previamente conhecidas e testadas. A inteligência, neste caso, pode ser observada pela utilização de três componentes principais da arquitetura do avatar: características físicas, comportamentos e conhecimento. Para uma maior eficiência na escolha de qual ação o avatar deva fazer, foi desenvolvida uma estratégia com auxilio de algoritmos genéticos que proporciona uma maior exatidão.

Palavras-chave: Mobilidade, Interação, Realidade Virtual, Agentes Inteligentes.

ABSTRACT

This master's dissertation aims to construct one 3D virtual environment in the form of a soccer stadium where simulations of a soccer game will be carried through, with the use of independent avatars. The construction of avatar use the H - Anim 1.1 modeling technique that it brings the concept of hierarchies and articulated body. The implementation of the application was developed with Java and API Java 3D. The autonomy is obtained by aid of Artificial Intelligence (AI) that it will act in the solution of the problems together with the use of rules previously known and tested. Intelligence, in this in case that, it can be observed by the use of three main components of the architecture of avatar: physical characteristics, behaviors and knowledge. For a better efficiency in the choice of which action avatar must make, was developed a strategy with assists of genetic algorithms that provides a better exactness.

Key-words: Mobility, Interaction, Virtual Reality, Intelligent Agents.

Capítulo 1

1 CONSIDERAÇÕES INICIAIS

1.1 - INTRODUÇÃO

Um mundo virtual ou ambiente virtual segundo [Darken, 1996] representa um determinado "lugar" virtual. Este lugar virtual é composto por um conjunto de componentes que tem como objetivo transmitir o significado que ele representa. Estes ambientes virtuais podem representar lugares reais, ou seja, lugares que existem no mundo real em que se vive como, por exemplo, o estádio de futebol do Clube do Remo, ou imaginários.

O desenvolvimento de ambientes virtuais como foi citado acima, pode ser feito através de diversas técnicas e ferramentas, mas o que se tem visto é que não basta simplesmente construir o mundo virtual, pois acontece de algumas vezes o usuário não conseguir se localizar dentro do ambiente.

Uma das formas que desenvolvedores de mundos virtuais têm encontrado para amenizar o problema da desorientação do usuário é posicionando graficamente um representante no ambiente, de forma a tornar visível a sua estada no mundo. Este representante é chamado avatar.

Avatares segundo [Bowskill, 1995], são representantes do usuário no mundo sintético. Podem ser utilizados apenas para que o usuário tenha alguém que o represente no ambiente, ou para participar ativamente através da execução de funções e da exibição de comportamentos.

Um avatar pode ser utilizado basicamente de duas formas dentro de um mundo virtual:

- Simplesmente como um representante gráfico do usuário;
- Como um representante apto a realizar algumas tarefas adicionais.

Este trabalho se propõe tratar de três assuntos que são:

As formas de construção de humanóides articulados através de técnicas de modelagens já existentes usando a API JAVA 3D da Sun, que oferece o suporte da tecnologia Java tais como: portabilidade, segurança e escalabilidade.

Técnicas de construção de ambientes virtuais também foram implementadas sendo que nestas utilizou-se fortemente os conceitos de texturização, transparências, movimentações, translações rotações.

Sistemas baseados em regras onde os avatares que representam jogadores de futebol realizam determinadas tarefas tais como: cobranças de penalidades e simulação de jogadas, sendo que a aplicação desenvolvida permite uma interação do usuário com o ambiente quando este estiver em modo de cobrança de penalidade. Para uma melhor eficiência nas ações dos avatares foi desenvolvido uma algoritmo baseado em computação genética que faz uma mair seleção de indivíduos ou seja a cada seleção feita só ficam os mais aptos a realizar determinada tarefa.

1.2 - Objetivos e Contribuições

O principal objetivo desta dissertação e demonstrar que avatares articulados quando combinados com sistemas baseados em regras, são geralmente uma boa escolha, para treinamentos em situações conhecidas, pois desta forma pode-se através de simulações tirar conclusões sobre determinadas situações.

Sistemas baseados em regras geralmente geram boas soluções, quando se tem provado que as regras utilizadas sempre gerem uma solução ótima, para qualquer estado do processo.

1.2.1 - Objetivos Específicos

- Utilizar a técnica de construção de humanóides com articulações denominada H–Anim 1.1;
- Desenvolver uma modelagem arquitetural para simulação utilizando a linguagem Java;
- Desenvolver uma interface com alta interatividade com recursos de realidade virtual, utilizando a API Java 3D;
- Desenvolver uma modelagem orientada a objetos com Unified Modeling Language – UML;
- Implementar regras que denotem autonomia aos jogadores através de algoritmos genéticos, mais precisamente o método Roleta;

1.3 - Organização do trabalho

O trabalho esta dividido em capítulos os quais possuem características especificas.No capitulo 2 será explicado a fundamentação teórica de Realidade Virtual e seus principais componentes, alem de apresentar um breve descrição da tecnologia Java e Java 3D

finalizando com a arquitetura para modelagem de avatares utilizada bem como as regras e algoritmos implementados.

No capitulo 3 é apresentado a implementação do H-anim 1.1 em suas fases de concepção no período de desenvolvimento do trabalho, as técnicas de construções de mundos virtuais, bem como as regras desenvolvidas e sua implementação com algoritmos genéticos usando o método roleta, além de mostrar toda a modelagem UML.

No capitulo 4 e ultimo capitulo faz-se uma analise geral do trabalho e aponta – se alguns estudos futuros.

Capítulo 2

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Realidade Virtual

Várias são as definições de realidade virtual, mas em geral, a mesma refere-se a uma experiência imersiva e interativa baseada em imagens gráficas 3D geradas em tempo real por computador, ou seja, é uma simulação, gerada por computador, de um mundo real ou imaginário.

Segundo [Kirner, 2004] realidade virtual (RV) pode ser definida de uma maneira simplificada como sendo a forma mais avançada de interface atualmente disponível. Com aplicação na maioria das áreas do conhecimento, senão em todas, e com um grande investimento das indústrias na produção de hardware, software e dispositivos de entrada e saída especiais, a realidade virtual vem experimentando um desenvolvimento acelerado nos últimos anos e indicando perspectivas bastante promissoras para os diversos segmentos vinculados com a área. Outras definições sobre Realidade Virtual:

Latta [Latta, 1994] conceitua a Realidade Virtual como uma avançada interface homem-máquina que simula um ambiente realístico, permitindo que os participantes interajam com ele. Essa interface é considerada a mais avançada até agora disponível, pois busca levar ao usuário sensações que lhe dão informações sobre o mundo virtual como se ele realmente existisse.

Pimentel [Pimentel, 1995] define Realidade Virtual como o uso de alta tecnologia para convencer o usuário de que ele está em outra realidade, promovendo completamente o seu envolvimento.

Levando-se em consideração todos os conceitos relativos à Realidade Virtual, pode-se afirmar que a mesma é uma técnica avançada de interface, na qual o usuário realiza imersão, navegação e interação em um ambiente tridimensional gerado pelo computador por intermédio de vias multi-sensoriais.

As três idéias básicas do sistema de Realidade Virtual. São elas:

Imersão: Está ligada com a sensação de estar dentro do ambiente.Isto pode ser conseguido com alguns equipamentos,como capacete de visualização ou sistemas baseados em múltiplas projeções.

Interação: Esta idéia está relacionada com a capacidade do computador em detectar as entradas do usuário e modificar instantaneamente o mundo virtual e as ações sobre ele (capacidade reativa).

Envolvimento: está relacionada com o grau de motivação para o engajamento de uma pessoa com determinada atividade, podendo ser passivo ou ativo, esta relacionado a mavegação.

A Realidade Virtual pode ser classificada nas seguintes categorias :

- Sistemas de Imersão: aqueles que submergem ou introduzem o explorador de maneira estreita com o mundo virtual, mediante a utilização de sistemas visuais do tipo HMD.
- Sistema Desktop: englobam as aplicações que mostram uma imagem 2D ou 3D na tela plana de um monitor de computador.

Um sistema de RV, sempre possui um ambiente virtual (AV) que é uma cena 3D, cuja a geometria esta relacionada com animações e interações em tempo real. Esses AV, são descritos por grafos de cena.

Alguns termos importantes utilizados freqüentemente nesta dissertação são apresentados aqui. Pelo fato de suas definições serem, às vezes, compreendidas de forma errada ou terem mais de um significado, é importante que seu sentido seja claramente descrito para o contexto deste trabalho.

2.1.1 Grafo de Cena

Grafos de cena são ferramentas conceituais para representação de ambientes virtuais tridimensionais nas aplicações de computação gráfica [WALSH, 2002]. Um ambiente virtual é uma representação de diversos aspectos do mundo real ou abstrato. Os aspectos considerados em uma aplicação de computação gráfica são: posição do objeto, forma, textura da superfície, iluminação, entre outros [FERREIRA, 1999]. Cada um desses aspectos e seus atributos estão bastante detalhados na literatura, porém se apresenta, aqui, um resumo dos mais importantes:

- Descrição geométrica: É qualquer maneira de se representar a forma da entidade que pode ser processada para se obter uma imagem dessa entidade. A maneira mais comum é a representação aproximada por um conjunto de polígonos (mais especificamente por triângulos). O grau de complexidade da descrição geométrica é, em geral, diretamente proporcional à qualidade visual, porém inversamente proporcional à velocidade com que a imagem é gerada. Em RV a resposta em tempo real e mais importante que a complexidade geométrica, ao contrário da computação gráfica.

- Câmera: É a visão do mundo virtual. Geralmente ela é uma câmera de projeção perspectiva;
- Transformação: O objeto é posicionado no mundo virtual através de uma transformação geométrica. Ela transforma as coordenadas locais do objeto nas coordenadas do mundo virtual. As transformações são muito importantes para definir a hierarquia dos objetos;
- Aparência: Material, textura, transparência, sombra e reflexão estão entre os diversos atributos que definem a aparência de um objeto. Assim como a descrição geométrica, a aparência interfere diretamente na imagem final a ser gerada e na velocidade de geração;
- Comportamento: Um objeto pode ser estático ou dinâmico. O objeto dinâmico é aquele que muda de posição, forma ou aparência entre um quadro e outro;
- Iluminação: várias fontes de luz podem ser adicionadas à cena e vários são os modelos de iluminação que podem ser empregados.

Cada um desses aspectos deve ser inserido em um grafo de cena para representar o ambiente virtual. O grafo de cena é formado, portanto, por nós conectados por arestas compondo um grafo acíclico direcionado. Cada nó possui um conjunto de atributos que podem, ou não, influenciar seus nós conectados. Os nós são organizados de uma maneira hierárquica correspondendo semântica e espacialmente com o mundo modelado [SILVA, 2002].

Os nós podem ser divididos em três categorias: nó raiz; nós intermediários, que são chamados de nós internos ou nós de agrupamento; e os nós folha que estão localizados no final de um ramo. O nó raiz é o primeiro do grafo e todos os outros nós estão ligados a ele direta ou indiretamente. O nós internos possuem várias propriedades, sendo o uso mais comum o de representar transformações 3D (rotação, translação e escala). Os nós folha contêm, geralmente, a representação geométrica de um objeto (ou dados de áudio, quando o grafo de cena possuir esse recurso).

2.1.2 -Interação

O processo de interação em ambientes virtuais consiste na capacidade reativa do sistema em detectar e responder a cada ação do usuário através de modificações instantâneas no ambiente virtual. Trata-se de um processo contínuo que tanto pode ser realizado através de dispositivos especiais como convencionais. A interação em ambientes tridimensionais é realizada com o objetivo de efetuar operações de seleção, manipulação e navegação no

ambiente tridimensional. [HUTZLER, 1998].

O usuário efetua ações no ambiente virtual através dos dispositivos de entrada. Estas ações (estímulos motores) são efetuadas através de técnicas de interação, que correspondem a métodos através do qual o usuário especifica comandos e dados para o sistema computacional, para executar tarefas. A resposta do sistema se dá através de dispositivos de saída que estimulam os sentidos do usuário (tato, visão, audição).

A percepção é o resultado de uma cadeia de processamentos de estímulos obtidos pelos diferentes sentidos, como a visão, a audição e o tato. Tais estímulos são processados e assimilados pelo cérebro e, juntamente com as expectativas criadas com base em experiências anteriores, definem como as pessoas interagem com o ambiente.

Alguns dos atributos utilizados no processo perceptual são: cor, textura, tamanho, perspectiva, oclusão, sombra e movimento do observador. No processo de interação humano-computador usando a tecnologia oferecida pelos dispositivos de realidade virtual (RV), busca-se eliminar qualquer estímulo do mundo real através da produção de estímulos virtuais realistas. O objetivo é proporcionar imersão total no ambiente virtual, "desligando" o usuário do mundo real. A figura 2.1 apresenta o esquema que resume este processo. O usuário tem seus sentidos estimulados pelo computador e por um conjunto de atuadores e envia ao computador, por intermédio desses mesmos atuadores, as ações desejadas.

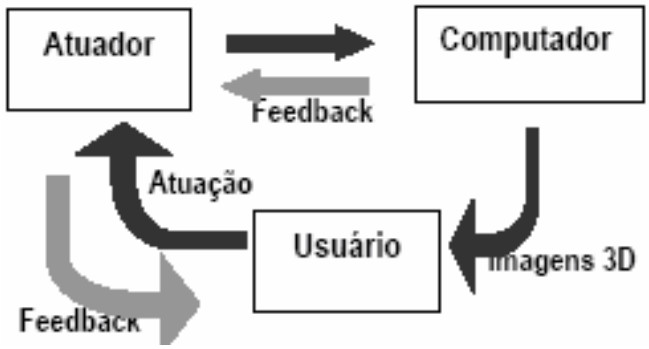


Figura 2.1. Esquema de interação humano computador em ambientes virtuais interativos.[PINHO,2004]

Existem várias técnicas de interação para o usuário realizar suas ações no ambiente virtual, técnicas que podem ser mais apropriadas ou não para determinados tipos de tarefas.

De um modo geral, um processo de interação comporta-se como um sistema contínuo e sempre conta com três componentes básicos [PINHO, 2004]:

- Um dispositivo de entrada que captura os movimentos do usuário;

- Uma função de transferência que processa uma ação no sistema de acordo com os movimentos realizados pelo usuário;
- Um dispositivo de saída para exibir os efeitos das ações do usuário.

Técnicas de interação estão relacionadas às funções de transferência, de modo que sua eficiência é avaliada na geração rápida e precisa das saídas computadas de acordo com as entradas do usuário.

O desenvolvimento de interface baseadas na tecnologia de RV deve considerar novos aspectos de software e hardware para elaborar formas de interação entre o usuário e o sistema de tal forma que seu esforço cognitivo seja reduzido. Nestas situações, há que se considerar o repositório que o próprio usuário traz consigo de outros aprendizados e experiências.

Independentemente de quantas dimensões, as tarefas mínimas desempenhadas em uma interface são:

- Seleção: selecionar um ou mais itens dado um conjunto de opções;
- Posição: alterar o posicionamento de componentes ou objetos na tela ou ambiente;
- Orientação: definir a orientação de um objeto na área de trabalho;
- Caminho: conjunto de posições ou orientações ao longo do tempo;
- Quantificação: especificar uma medição de um atributo;

2.1.3 -Navegação

Darken [DARKEN, 1996] define navegação como:

> *"(1) Viajar pela água: barco à vela; (2) Conduzir uma direção através de algum meio: conduzir um avião; (3) Viajar muito: caminhar".*

A navegação no mundo virtual depende de uma série de fatores uma vez que envolve dispositivos não convencionais de entrada e saída e é realizada em tempo real. No mundo virtual, assim como no mundo real, a navegação acontece no espaço tridimensional, sendo resultante da combinação de movimentos de translação e de rotação.

2.1.4 -Localização

Segundo Peponis [PEPONIS et al, 1995], conceitua-se localização como

"A habilidade de encontrar o caminho para um lugar particular de uma maneira conveniente, e de reconhecer o destino quando alcançado.".

A localização está focalizada na noção de conhecimento espacial, ou seja, o quão bem

o visitante consegue se locomover no mundo usando sua habilidade espacial que, segundo [SATALICH, 1998], é formada por três dimensões básicas:

"... a percepção do ambiente através dos sentidos, o processo cognitivo que Leva ao aprendizado sobre o ambiente e a percepção do relacionamento entre os objetos do ambiente.".

Pode-se dizer que localização é o processo dinâmico de usar a habilidade espacial e o conhecimento sobre navegação em um ambiente para alcançar o destino desejado. Em outras palavras, a localização precede ou facilita a navegação. Esta última trata da ação de um movimento direcionado e a localização trata de uma ação cognitiva envolvendo resolução de rota.

2.1.5 -Orientação

Segundo Passini [PASSINI, 1997], o termo "orientação" tem suas raízes na palavra "oriente", pelo costume, em algumas culturas, de edificar certas construções (normalmente igrejas ou construções de importância religiosa) ou entradas principalmente à leste. [DARKEN, 1996] descreve a palavra "orientar" como:

"(1) Virar para o leste. (2) Posicionar ou ajustar, como um mapa, exatamente em relação aos ponteiros da bússola.".

Segundo Passini [PASSINI, 1997],

"Orientação espacial e localização... proporcionam às pessoas uma idéia sobre o espaço adjacente, sobre suas posições no espaço e permitem movimentação intencional dentro desse espaço.".

No contexto desta pesquisa, orientação se refere ao conhecimento sobre informações de direção em relação ao ambiente. Este conhecimento pode ser absoluto ou relativo. Orientação absoluta envolve orientar com relação a um sistema estático de coordenadas tal como o familiar sistema cardinal de norte, sul, leste e oeste. Orientação relativa pode ser aplicada a componentes móveis dentro de um ambiente. Por exemplo um avião, no qual o passageiro pode ir até o banheiro sem pensar no movimento que realiza em relação ao solo.

Uma posição pode ser especificada em termos de sua identificação absoluta (por exemplo, como definida por um sistema de coordenadas) ou relativamente (por exemplo, para algum ponto de referência). A figura 2.2 (a) mostra um exemplo de orientação absoluta com relação à direção norte e outro de orientação relativa com relação ao ponto de referência R,

enquanto que na figura 2.2 (b) pode ser visto um exemplo de posição absoluta com relação a um sistema cartesiano de coordenadas com origem (0,0) e outro de posição relativa com relação a um ponto de referência R definido em termos de uma distância D (D_1 e D_2) e de um ângulo relativo α (α_1 e α_2).

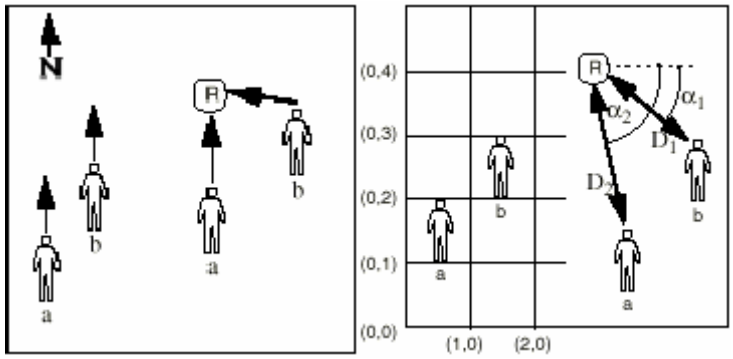


Figura 2.2: (a) Orientação absoluta e relativa; (b) Posição absoluta e relativa.[THALMANN 1998]

2.1.6 -Realidade Virtual Imersiva

Os Sistemas Imersivos são o que podemos chamar de realidade virtual tradicional. Neles o usuário veste um capacete de realidade virtual, luva, rastreador de posição e fones de ouvido ou entra num sistema baseado em múltiplas projeções – CAVE. Nesses casos usando o capacete ou entrando na CAVE, o usuário ao mover a cabeça para os lados, para cima e para baixo, verá todo o cenário virtual como se estivesse dentro dele. Com estes equipamentos o usuário é "desligado do mundo real" e passa a visualizar, ouvir e sentir apenas os estímulos gerados por um sistema computacional. As ações do seu corpo são interpretadas como a única forma de entrada dos dados. [AKAGUI et al, 2004]

Os sistemas que implementam RV permitem ao usuário a experiência em primeira pessoa de situações as quais, geralmente, no mundo real, o mesmo jamais poderia vivenciar. Por exemplo, pode-se simular reações químicas em reatores e permitir ao usuário 'entrar' nas reações de forma a poder ver o processo acontecendo com os elementos químicos. O usuário pode entrar em mundos que simulem mundos reais, tais como indústrias e vivenciar situações de perigo, como incêndios, tomar decisões e ver as conseqüências das mesmas. Um aluno pode participar de uma estória dialogando com os personagens, dando sugestões, vivenciando as aventuras e peripécias. Este tipo de interação torna o aprendizado muito eficiente, permitindo à criança ou adolescente decidir, analisar, concluir, realizar novas tentativas, perceber as conseqüências de suas decisões.

Alguns fatores cognitivos são apontados como necessários ao sucesso da imersão virtual para um indivíduo: suscetibilidade para imersão e qualidade da imersão [FIALHO, 1998.].

A suscetibilidade depende dos seguintes fatores:

- Imaginação: resistência da imaginação visual; sonhos, consciência de si; absorção em sonhos quando acordado; habilidade para, de boa vontade, suspender a descrença; profundidade de envolvimento em livros, teatro, etc;
- Concentração e Atenção: conflito cognitivo ao sustentar duas imersões recursivas, navegação espacial;
- Autocontrole: participação ativa e catarse.

A qualidade da imersão depende de:

- Recursos do Meio Ambiente para Imersão: persistência do objeto, perfeição sensória, interatividade, realismo do meio ambiente, montante do retardamento ou atraso, tamanho do campo de visão, localização precisa do egocentro ou imagem corporal, prazer e satisfação com a novidade da experiência;
- Distrações devido ao Meio Ambiente: presença de distratores sonoros ou táteis, fadiga e irritação pelo volumoso equipamento, restritividade do equipamento, similaridade entre o mundo real e o mundo da Realidade Virtual;
- Efeitos Psicológicos: distúrbios pelo simulador, desorientação após imersão;

2.1.7 -Realidade Virtual Não-Imersiva (RV Desktop)

Os sistemas não-imersivos são aqueles nos quais o usuário vê o universo virtual por uma tela convencional de computador ou de uma projeção. Nesta classe enquadram-se hoje a grande maioria dos jogos eletrônicos e os sistemas interativos de navegação. O som destes ambientes é produzido por caixas de som colocadas à frente do usuário como as que encontramos nos tradicionais kits multimídia. Alguns destes sistemas já possuem hoje, dispositivos especiais de interação como mouse 3D, luvas eletrônicas e óculos para visão estereoscópica. A leitura dos movimentos do usuário é limitada, na maioria dos casos, apenas à leitura de dispositivos que forneçam a posição de algumas partes do corpo do usuário. Nestes ambientes o usuário deve manter-se olhando para a tela para poder ver o mundo virtual.

Estes sistemas são também conhecidos como “Fish-tank VR”, pois o usuário observa o

mundo virtual como quem olha um aquário através de um vidro, sem imergir dentro dele, ou WonW – World on Windows Systems" quando o usuário vê o mundo virtual através de uma janela na tela do computador [ISDALE, 1998].

2.1.8 -Realidade Virtual Imersiva X Realidade Virtual Não-Imersiva.

As tabelas 1 e 2 abaixo mostram uma comparação entre a RV Imersiva e a RV Desktop, mostrando as vantagens e desvantagens de uma sobre a outra.

Vantagens da RV Desktop	Vantagens da RV Imersiva
1. O equipamento é relativamente mais barato e acessível; 2. É mais simples de manipular; **3.** Há poucos problemas com equipamentos.	1. O nível de envolvimento com mundo é alto e fácil de ser atingido; 2. Há o auxílio de vários equipamentos; 3. É incorporada em diversas áreas;

Tabela 1 – Comparação entre as vantagens da RV Imersiva e RV desktop

Desvantagens da RV Desktop	Desvantagens da RV Imersiva
1. Não tem a possibilidade da imersão; 2. Envolvimento com o mundo é mais difícil;	1. Os equipamentos são caros e de difícil acesso; 2. O uso é mais complicado.

Tabela 2 – Comparação entre as desvantagens da RV Imersiva e RV desktop

2.2 - Sistemas de Desenvolvimento de RV

2.2.1 -A linguagem JAVA e a API Java 3D TM

Programar envolve lidar com complexidade, e problemas complexos exigem uma linguagem de programação versátil e robusta. A linguagem Java começou como um esforço da *Sun Microsystems* para criar uma linguagem de programação extensível, que criasse programas capazes de executar em vários tipos diferentes de dispositivos e arquiteturas. Graças a sua versatilidade, o Java se tornou a linguagem principal para distribuição de material executável pela Internet, dando aos usuários retorno automático em páginas inteligentes, que executam no lado de cliente (client-based application) ou do servidor (servlets).

A linguagem Java estendeu e facilitou a maioria das tarefas complexas compreendidas no universo da computação, entre as quais: programação em rede; programação distribuída e multitarefa; programação multi-plataforma; mudanças dinâmicas de código; gerenciamento de segurança [ECKEL, 2002]. Também obteve grande repercussão por se tratar de uma ferramenta livre, seguindo a filosofia do código aberto, deixando pouco mistério a respeito de seu funcionamento interno e de suas capacidades. Algumas das características que levaram à escolha da linguagem Java são:

- • Linguagem orientada a objetos;
- • Suporte à interação por console ou interface visual;
- • Arquitetura concorrente com o uso de "threads";
- • Compatibilidade com diversas plataformas sem re-compilação de código;
- • Execução em modo aplicativo ou Applet para execução na Internet;
- • Distribuição gratuita;
- • Material de suporte disponível on-line.

A disponibilidade de material de qualidade e suporte on-line em um fórum de usuários estão entre as principais características da linguagem. Para uma introdução detalhada à linguagem e seus paradigmas, o livro "Thinking in Java" pode ser obtido on-line ou nas livrarias [ECKEL, 2002]. O livro "Java, como programar" [DEITEL, 2002] é utilizado na maioria dos cursos de graduação de introdução à linguagem Java. Como referência para usuários iniciados, a documentação da linguagem Java disponível no sítio da *Sun Microsystems*, na Internet, é a mais completa existente [SUN SITE].

A API Java3D é uma extensão da linguagem Java cuja finalidade é prover acesso a dispositivos gráficos independentes de plataforma, sem que a execução dos programas comprometa sua portabilidade. A API (Application Programming Interface) incorpora métodos de acesso de alto-nível para a modelagem da cena (Grafo de Cena) e roda por cima de arquiteturas de baixo nível como OpenGL ou DirectX.

A API Java3D utiliza um modelo de visualização flexível e orientado a objetos para construção da cena, na qual os objetos são conectados hierarquicamente em um grafo descritivo da cena representada, podendo gerar saídas para uma série de dispositivos, provendo acesso às mais modernas tecnologias de hardware e capacidades gráficas

disponíveis, tirando proveito de aceleradores gráficos para melhorar a performance. Para uma experiência imersiva, o mecanismo de entrada aceita uma série de dispositivos, desde simples mouses até apontadores (wands) e luvas (gloves). Figura 2.2. Exemplo de grafo de cena no API Java3D.

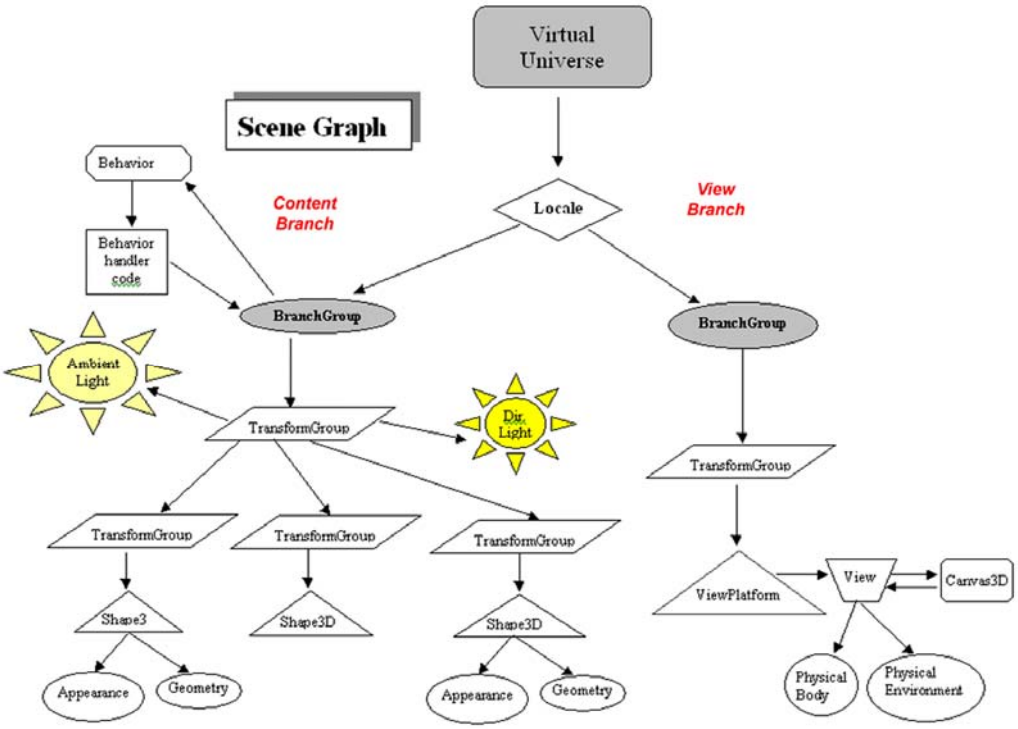


Figura 2.3 : Grafo de cena da API Java 3D. Sun [SUN SITE]

Uma boa referência para o uso de Java3D é "Getting Started with the Java3DTM API", disponível gratuitamente no sítio da Sun [SUN SITE], onde também estão disponíveis a especificação do API e os manuais oficiais.

Para utilizar a plataforma, o usuário deverá escrever algum código, e, portanto, deverá possuir algum conhecimento da linguagem Java.

2.2.1.1 -Classes Principais

A API Java 3D fornece um grande número de classes para especificação, posicionamento e visualização de objetos gráficos. Nesta seção são apresentadas algumas classes fundamentais para o desenvolvimento de programas gráficos, que aparecem no exemplo do Anexo 1. Detalhes sobre as demais classes e informações sobre seus construtores e métodos, podem ser obtidos consultando a documentação da API Java 3D. Uma das classes mais importantes é a *SimpleUniverse*, pois é responsável pela configuração de um ambiente mínimo para executar um programa Java 3D, fornecendo as funcionalidades necessárias para a maioria das

aplicações.

Quando uma instância de *SimpleUniverse* é criada, automaticamente são criados todos os objetos necessários para o sub-grafo de visualização, tais como Locale, ViewingPlatform e Viewer.

GraphicsConfiguration é uma classe que faz parte do pacote awt, responsável pela descrição das características do dispositivo gráfico (impressora ou monitor). Sua estrutura varia de plataforma para plataforma, fornecendo, por exemplo, a resolução do dispositivo.

A classe Canvas3D fornece o canvas, ou seja, uma área de desenho, onde é realizada a visualização 3D.

A classe BranchGroup serve como ponteiro para a raiz de um sub-grafo de cena . Instâncias desta classe são os únicos objetos que podem ser inseridos em Locale. Um sub-grafo de cena que tem um BranchGroup como raiz pode ser considerado como uma compile unit, podendo ser compilado, inserido em um universo virtual (associando-o com Locale) e desassociado deste universo em tempo de execução.

As transformações geométricas de escala, rotação e translação são especificadas através de uma instância de Transform3D, que representa uma matriz 4x4 de números reais (float). Objetos da classe TransformGroup, por sua vez, especificam uma transformação, através de um objeto Transform3D, que será aplicada a todos os seus filhos. Ao serem aplicadas as transformações, deve-se considerar que os efeitos num grafo de cena são cumulativos.

A classe BoundingSphere define uma região (ou volume) limitada por uma esfera que é especificada a partir de um ponto central e um raio.

2.2.1.2 -Animação

De forma simplificada, a animação consiste na exibição de imagens em seqüência. Em Java 3D, a interação corresponde à execução de alterações no grafo de cena em resposta a ações do usuário, e a animação é definida como a execução de alterações com a passagem do tempo, e não com uma ação do usuário. A API Java 3D fornece um grande número de subclasses da classe Behavior que são úteis para criar animações: Billboard, Interpolator e LOD.

O Interpolator possui um conjunto de subclasses que, em conjunto com objetos Alpha, manipulam alguns parâmetros de um grafo de cena para criar uma animação baseada no tempo. Algumas de suas subclasses são: ColorInterpolator, PathInterpolator,

PositionInterpolator, RotationInterpolator e ScaleInterpolator.

A classe Alpha fornece métodos para converter um valor de tempo em um valor entre 0.0 e 1.0 (f(t) = [0.0,1.0]). Estes valores são úteis para as subclasses de Interpolator. As constantes INCREASING_ENABLE e DECREASING_ENABLE, que servem para indicar se os valores serão gerados na ordem crescente ou decrescente, são definidas nesta classe.

Seus construtores criam e inicializam o objeto com valores pré-definidos, e também permitem que o usuário especifique valores diferentes para seus atributos. Por exemplo, com o construtor Alpha (int loopCount, long increasingAlphaDuration), no primeiro parâmetro é possível definir o número de vezes que o objeto será executado (-1 para um laço) e o período de tempo durante o qual Alpha vai de zero para um.

A classe RotationInterpolator define uma animação que modifica o componente de rotação do TransformGroup através de uma interpolação linear entre um par de ângulos especificados (usando o valor gerado por um objeto Alpha). O ângulo interpolado é usado para indicar uma transformação de rotação sobre o eixo Y. De maneira análoga, PositionInterpolator e ScaleInterpolation modificam, respectivamente, o componente de translação e de escala através de uma interpolação linear entre um par de posições ou de valores de escala (ambos usam o valor gerado por um objeto Alpha).

A posição interpolada para PositionInterpolator é usada para gerar uma transformação de translação sobre o eixo X, e para ScaleInterpolation é usada para gerar uma transformação de escala sobre o sistema de coordenada do interpolator.

2.2.1.3 -Colisão

A detecção de colisão é um ponto importante para a simulação do comportamento dos objetos dentro do ambiente virtual, este estudo e sua aplicação permite representar esses mundos de forma muito mais próxima da realidade, dando a quem esta imerso a sensação de realidade. Isto é importante para garantir a atenção do participante na tarefa em que esta imerso.[Selman, 2002]

Existem alguns métodos eficientes usados para a detecção de colisões entre objetos móveis e estáticos. As colisões devem ser detectados em tempo real, para isso procura-se normalmente processos eficientes.Uma solução, normalmente adotada, é a abordagem hierárquica, onde os objetos são envolvidos por sólidos simples como esferas ou cubos e faz-se uma analise de interferência ou sobre posição. Destes se não houver nenhuma sobreposição, não haverá colisão, mais se houver, o risco existirá e outros métodos mais refinados e custosos serão

aplicados. Como resultado de uma colisão poderá ocorrer deformação nos corpos envolvidos na mudança de trajetória.

Java 3D mantem a informação do limite do objeto para os nós no gafo de cena. Cada nó contem um campo de limites que armazena a extensão geométrica do nó. O motor da detecção da colisão de Java 3D emprega o campo de CollisionBounds. Um algoritmo simples de detecção da colisão hierárquica através dos objetos no grafo de cena. Ele trabalha testando uma possível interseção entre o CollisionBounds de um objeto e o CollisionBounds de cada outro objeto de Shape3D no grafo de cena.

2.2.2 -VRML - Virtual Reality Modelling Language

É uma linguagem de descrição de cenas ou mundos 3D, usada para criar ambientes tridimensionais que podem ser transmitidos através de redes de computador como a Internet. O VRML é, atualmente, o formato de compartilhamento de descrições tridimensionais mais difundido na Internet, assim como HTML é atualmente o formato mais difundido para descrição de páginas hipermídia. Arquivos com extensão *.wrl contêm descrições compactas de mundos virtuais (cenas) que podem ser transferidas através da Web e visualizadas através de navegadores VRML. Ao permitir que objetos em um mundo tridimensional possam referenciar outros mundos e recursos da Web, o VRML cria espaços de interação multidimensionais.

Navegadores VRML permitem que o usuário visualize e interaja com o mundo através de uma representação abstrata de sua pessoa, chamada de avatar, e que atua como uma marionete manipulada pelo usuário.

Embora seu nome diga o contrário, o VRML não trata de realidade virtual, pois não modela experiências imersivas em mundos 3D nas quais são usadas, por exemplo, capacetes, luvas e outros sensores e atuadores (VRML Review Board, 1997), embora nada impeça que aplicações de realidade virtual possam utilizar o VRML como suporte à descrição de cenas e suas propriedades interativas.

Outro aspecto importante sobre o VRML é que o mesmo não é um modelo inerentemente multiusuário. O esquema de documentos, códigos ou mundos virtuais sob demanda, obtido com o suporte de HTML, Applets Java e VRML, respectivamente, não oferece suporte natural ao compartilhamento simultâneo de um mesmo mundo virtual (documento, programa) por várias pessoas. No caso do VRML, por exemplo, cada usuário da Web interage individualmente com sua própria cópia isolada do mundo virtual, alheio às

explorações empreendidas por outros usuários que podem estar naquele mesmo instante utilizando outra cópia daquele mesmo mundo. Modelos como Open Community (Mitsubishi Electric et alli, 1999) e Universal Avatars (Ma et alli, 1996) oferecem alternativas de solução para este problema.

2.3 -Avatares e Humanos Virtuais

O avatar é uma representação do usuário no ambiente virtual, juntamente com seus comportamentos [ÇAPIN, 1999].

Define-se por esta expressão as entidades de software interativas e com diferentes graus de autonomia, inseridas em um ambiente virtual e apresentadas ao usuário com aparência gráfica e movimentos basicamente semelhantes aos de seres humanos. Dependendo do objetivo ou da abordagem adotada, podem também ser denominados atores (ou personagens) digitais, sintéticos ou virtuais.

Os *avatares* são um domínio específico da pesquisa científica em RV que trata da simulação de entidades humanas nos computadores [*ÇAPIN 199*9]. Envolvendo representação, movimento e comportamento, tais entidades são usadas em várias aplicações, tais como: filmes, produções de TV, na indústria, jogos, telecomunicações (clones para a representação de seres humanos), medicina etc. É claro que, em se tratando da área, cada representação tem sua necessidade. Uma aplicação médica requer uma simulação exata de certos órgãos internos; a indústria do cinema requer padrões elevados de estética, movimentos naturais e expressões faciais. Deste modo, vê-se que a variedade de aplicações e requisições torna os avatares um vasto domínio da pesquisa científica compreendendo vários tópicos científicos: anatomia e geometria com a criação de modelos humanos em gráficos 3-D, cabelo e representação da pele com boas definições, animação do esqueleto ou animação dos ângulos de junção da estrutura do esqueleto, animação da aparência exterior do corpo, facial, das mãos, locomoção etc. [*ÇAPIN, 199*9].

Os *avatares*:

- podem ser estáticos ou podem realizar ações simples (saudar e acenar) e complexas (correr, andar, chutar, cabecear uma bola, pular em várias posições);
- realísticos (um humanóide detalhado) ou não (esfera, carro, conjunto de cilindros).

A figura 2.4 mostra um avatar realístico (à esquerda) e outro não-realístico (constituído basicamente de esferas e cilindros).

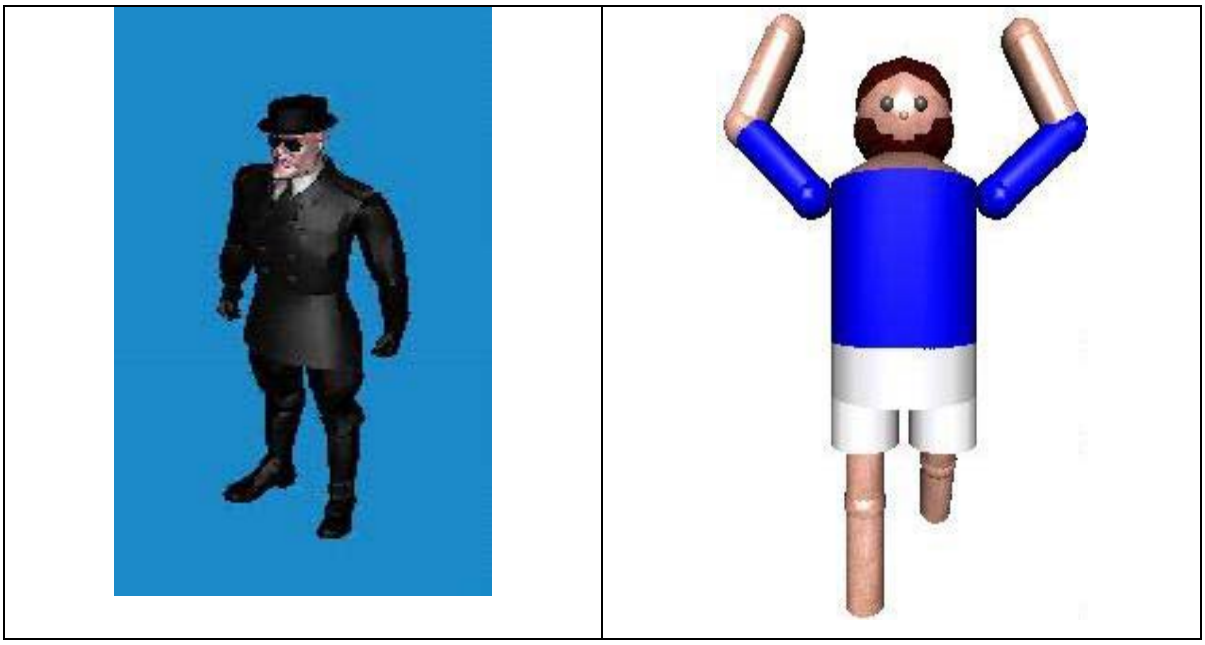


Figura 2.4 – Tipos de avatar.

Os principais pontos para se representar realisticamente um avatar são:

- anatomia e geometria com a criação de modelos humanos em gráficos 3D;
- cabelo e representação da pele em boas condições;
- animação do esqueleto;
- deformações faciais e na musculatura.

A representação dos participantes através de humanos virtuais (humanóides) atende às seguintes funções [ÇAPIN, 1999]:

- visual inteligível do avatar, relacionado à sua função no AV – um médico pode ser caracterizado por uma veste branca, por exemplo;
- os meios de interação com o mundo – ser capaz de abrir um livro ou mover uma cadeira;
- os meios de sentir as condições e as alterações do mundo, visualizar as posições dos objetos, ouvir sons relacionados a uma situação – abrir uma porta.

2.4 -MODELAGEM HUMANA

2.4.1 - Graus de Liberdade (Degrees Of Freedom)

Quando se modela um humanóide sempre se deve levar em consideração as limitações do equipamento a ser utilizado e qual função será desempenhada. Então, podem haver níveis de detalhe diferentes para cada configuração de máquina e para cada atividade (médico, guia etc.) em um AV.

O ponto crucial na modelagem humana é a definição correta das articulações e seus respectivos graus de liberdade (DOF – Degree Of Freedom).

A idéia é modelar o humanóide não como um objeto, mas como um conjunto de objetos ligados através de articulações. Um braço, por exemplo, possui os segmentos úmero, rádio e mão ligados hierarquicamente. Neste caso existem duas articulações: o cotovelo (ligando o úmero e o rádio) e o punho (entre o rádio e a mão). Acima destas existe o ombro, que liga o "corpo" ao úmero. Definindo graus de liberdade para cada uma destas articulações, podemos ter os seguintes graus mostrados na figura 2.5.

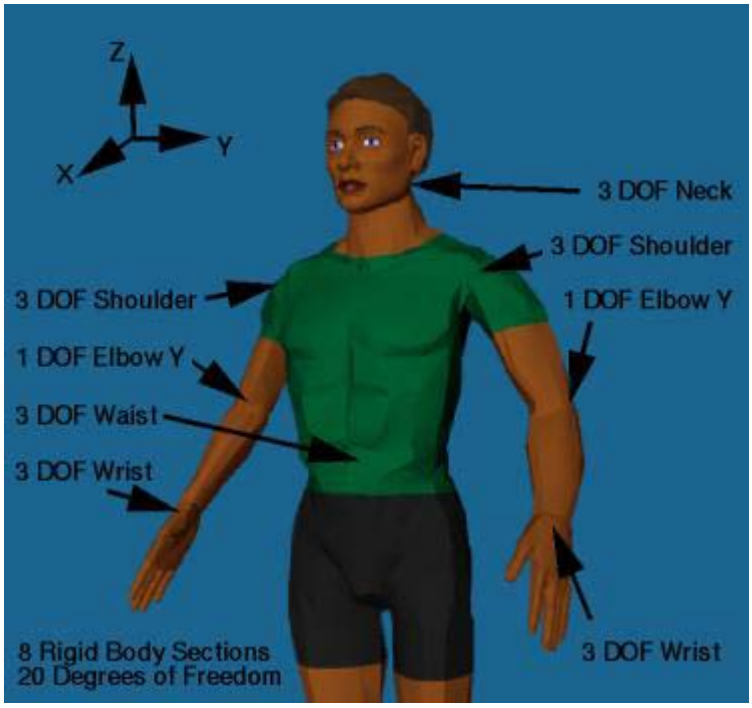


Figura 2.5 – Graus de Liberdade (Degrees of Freedom) de um modelo humano. [THALMANN 1998].

2.4.2 - Movimentação Humana

Os movimentos humanos devem ser implementados de modo a dar o maior grau de realismo possível, sem aumentar demasiadamente o nível de complexidade.

Todos os movimentos do corpo humano podem ser representados por equações da cinemática. Ela estabelece relações entre posição, velocidade e aceleração dos elementos do corpo humano. O posicionamento dos elementos pode ser encontrado por duas técnicas de cinemática: a direta (FK - Forward Kinematics) e a inversa (IK - Inverse Kinematics) [REMO, 2001].

A cinemática direta é básica para o estudo de manipulação mecânica. Este estudo de movimentação ignora todas as forças que a causam. Na FK, o movimento de cada articulação

é determinado explicitamente pelo animador. O movimento do ponto final é determinado indiretamente pelo acúmulo de todas as transformações através de toda a série de segmentos que conduzem àquele ponto final.

Na cinemática inversa, o animador define a posição de algum elemento do corpo humano no ponto final do movimento, como por exemplo posiciona somente a mão de um braço, o algoritmo de IK determina a posição e orientação de todas as articulações em uma hierarquia de segmentos que conduzem à mão. IK é computacionalmente mais cara devido à complexidade das articulações, mas a movimentação humana é uma área apropriada para esta técnica. A figura 2.6 traz um exemplo de transformações possíveis para criar uma movimentação.

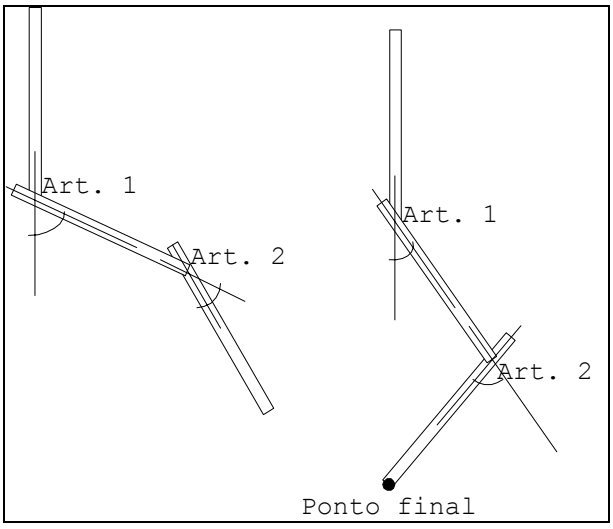


Figura 2.6 – Transformações aplicadas às articulações para chegar a um ponto. [THALMANN 1998].

2.4.3 - Criação do Corpo

Uma possível abordagem para a modelagem de um humano realístico e eficiente, que possa ser usado para animação e deformações em tempo real, é dividir a tarefa em três camadas [THALMANN, 1998].

A primeira refere-se ao esqueleto com articulações hierarquizadas similar a um esqueleto real. Também se definem aqui os graus de liberdade.

A segunda é a camada de tecido muscular criada através de primitivas volumétricas como elipsóides. A técnica consiste em ligar estas primitivas às respectivas articulações para se atingir o efeito de deformação.

A terceira camada é o equivalente à pele humana, criada através de técnicas de computação gráfica como superfícies splines. A figura 2.7 idealiza esta abordagem.

Figura 2.7 – Modelagem humana em camadas (THALMANN 1998).

2.4.4 - Controlando Avatares

Os participantes devem possuir uma forma de animar e controlar seus avatares em tempo real para que eles possam caminhar em um AV. É claro que, quanto maior for o grau de liberdade dos movimentos, mais complexa será a forma de animar tais entidades. Segundo [*ÇAPIM, 1999]*, os métodos para se controlar os avatares podem ser divididos em três classes:

- **Humanos Virtuais Controlados Diretamente**: o vetor de estado do avatar é modificado diretamente (isto é, utilizando sensores atachados pelo corpo). Esses avatares interagem com sistemas de captura de movimentos [*BABSKI, 1999]* que geralmente, trabalham em conjunto com complexas aplicações 3D tais como: 3D Studio Max. A primeira vez que esses modelos são criados, sistemas de captura de movimentos provêem a entrada de dados necessária para animar esses modelos [*ROEHL, 1999]*.

Uma completa representação do corpo virtual dos participantes deve possuir os mesmos movimentos que o corpo real apresenta, para que se tenha uma melhor sensação de imersão. Isso pode ser conseguido utilizando-se um número maior de sensores para capturar todos os graus de liberdade de movimento do corpo real (isto é, quando o animador abre os dedos a uma distância de 3cm, as mãos da cena original devem abrir exatamente o mesmo).

- **Humanos Virtuais guiados pelo usuário**: são aqueles que não respondem aos movimentos do usuário de forma direta. Esses avatares podem ser controlados através de scripts escritos pelos usuários [*ROEHL, 1999]*. Programas externos guiam o avatar definindo tarefas a serem executadas, e ele passa a desempenhar essas ações através do movimento coordenado de suas junções.

- **Humanos Virtuais Autônomos**: é assumido que o avatar possui um estado interno, o qual é construído com seus próprios objetivos e sensores de informações vindos do ambiente [*ROEHL, 1999]*.

Um sistema autônomo é aquele que está hábil a ditar suas próprias regras, o mesmo não é guiado por ações externas. Como esses avatares não são guiados pelo usuário eles devem ter seu próprio comportamento para agir autonomamente quando estiverem realizando suas tarefas. Isso requer a construção de comportamento para locomoção, tão como mecanismos apropriados para interação.

2.4.5 - Animação

Animação refere-se ao processo de geração dinâmica de uma série de quadros representando um conjunto de objetos, no qual cada quadro constitui uma alteração do quadro anterior.[RIGO, 1996]

Animar um objeto genérico não consiste simplesmente na definição da sua trajetória ou comportamento ao longo do tempo. Os objetos que serão animados devem ser representados corretamente no computador de modo a fornecer uma descrição adequada e intuitiva, facilitando assim a sua simulação.

Tal representação depende basicamente de dois fatores fundamentais: os parâmetros que se pretende animar e a representação visual dos objetos. Os parâmetros de animação relacionados aos objetos estão fortemente ligados ao tipo de simulação que será aplicada a esses objetos.

Em outras palavras, se se pretende simular aspectos dinâmicos de um objeto, seu conjunto de parâmetros de animação deve conter grandezas físicas como aceleração massa e torque. Em relação aos aspectos visuais dos objetos reais nem sempre é possível representar a sua complexidade total no computador. Deve-se lembrar que o computador é uma máquina com precisão e capacidade de representação finitas, enquanto que a maioria dos objetos existentes em nosso mundo são formados por um número grande de detalhes, formas e outros componentes que superam em muito a capacidade de representação dos computadores atuais.

2.4.6 – Animação de um Corpo Articulado

Um corpo articulado é caracterizado por conter membros ligados por juntas, estabelecendo relações comportamentais. Essas relações consistem nos graus de liberdade de movimento. Quanto aos membros, podem ser tanto corpos rígidos como corpos flexíveis,

porém na maioria dos sistemas de animação disponíveis, são rígidos [AMORIM, 1992].

A movimentação de um membro de um corpo articulado é restringida pelos outros membros em que está conectado através das juntas, as quais podem possuir de 1 a 3 graus de liberdade. Portanto, o número total de graus de liberdade de um corpo articulado pode ser calculado como a soma entre os 6 graus de liberdade que o corpo possui em relação ao universo e o somatório dos graus de liberdade de todas as juntas do corpo. No caso do corpo humano, pode-se ter até 200 graus de liberdade. No entanto, se os membros forem considerados flexíveis, devem se somar ao total de graus de liberdade do objeto, os graus correspondentes a cada membro isoladamente. Na Figura 2.8 pode-se ver um exemplo de um corpo rígido articulado e os graus de liberdade de suas articulações.

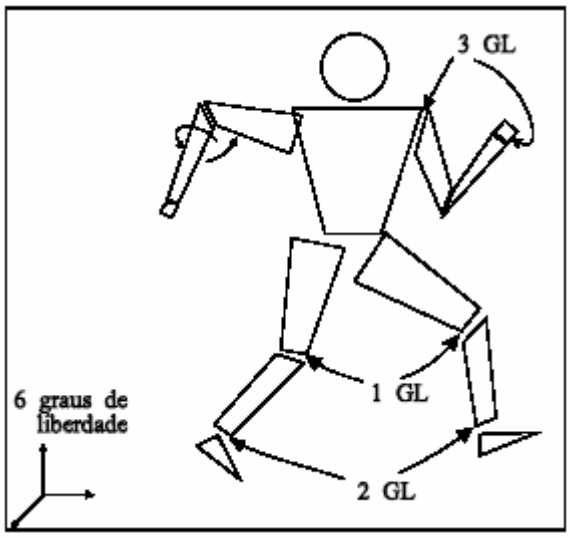


Figura 2.8 - Graus de liberdade de movimento para um corpo articulado

2.4.7 – Especificação H-anim 1.1

Devido o problema da portabilidade dos avatares em diferentes ambientes de desenvolvedores, surgiu a necessidade de padronizar a construção dos avatares humanóides bem como seus comportamentos. O Humanoid Animation Working Group, está desenvolvendo um padrão, o Hanim.

A especificação Hanim define o corpo humano como um conjunto de segmentos (como braço, pé, mão) que estão ligados uns aos outros por articulações (como cotovelo, pulso, tornozelo). Cada humanóide contém um conjunto de articulações organizadas hierarquicamente. Cada nó articulação pode conter outros nós articulação e pode também conter um nó segmento que representa a parte do corpo associada a essa articulação.

O principal nó da Hanim é o nó Humanoid, que armazena informações sobre o humanóide como o autor, nome, versão, pontos de visão e todas as suas articulações e segmentos.

A figura 2.9 mostra um diagrama com todas as articulações definidas pela especificação bem como a sua estrutura hierárquica.

A articulação *HumanoidRoot* é a raiz de toda a hierarquia de articulações. Divide o corpo em duas seções principais: a parte superior e a parte inferior. Todas as outras articulações são herdeiras desta articulação. Transformações aplicadas a este nó afetarão todo o corpo humano. Por exemplo, a movimentação é feita através de aplicações sucessivas de translações sobre esse nó.

A figura 2.9 mostra ainda que não é preciso executar todas as articulações para que uma implementação esteja de acordo com a especificação H-Anim 1.1, pois um dos propósitos da especificação é flexibilizar a construção do humanóide tanto de forma completa, como de partes dele (braços, pernas e outros), para uma finalidade específica.

A hierarquia de articulações torna extremamente simples animar partes do corpo de um humanóide. Basta mudar o ângulo de uma articulação (efetuando rotações), e todas as outras articulações filhas, e também os respectivos segmentos, serão afetados pela transformação permitindo, assim, mover as diversas partes do corpo. Quanto mais articulações forem implementadas, mais articulado será o humanóide o que permitirá uma maior mobilidade das várias partes do corpo.

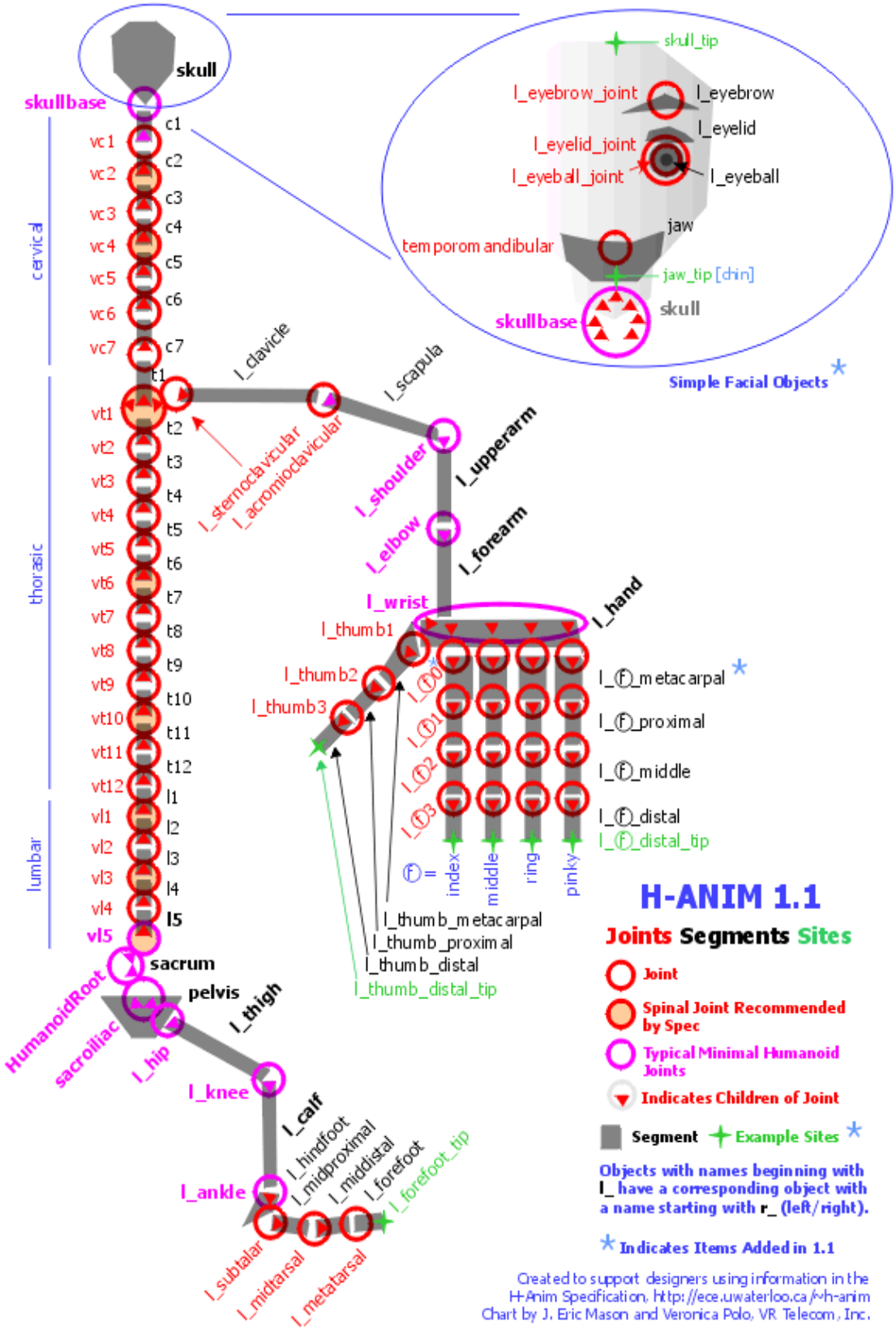


Figura 2.9 - Estrutura das articulações segundo a especificação [H-Anim 1.1]

Percebe-se na figura 2.10 alguns círculos rosados. Isto define o conjunto mínimo das articulações necessárias para se modelar um humanóide. O diagrama da figura 2.9 destaca quais são essas articulações, e os possíveis segmentos associados a elas.

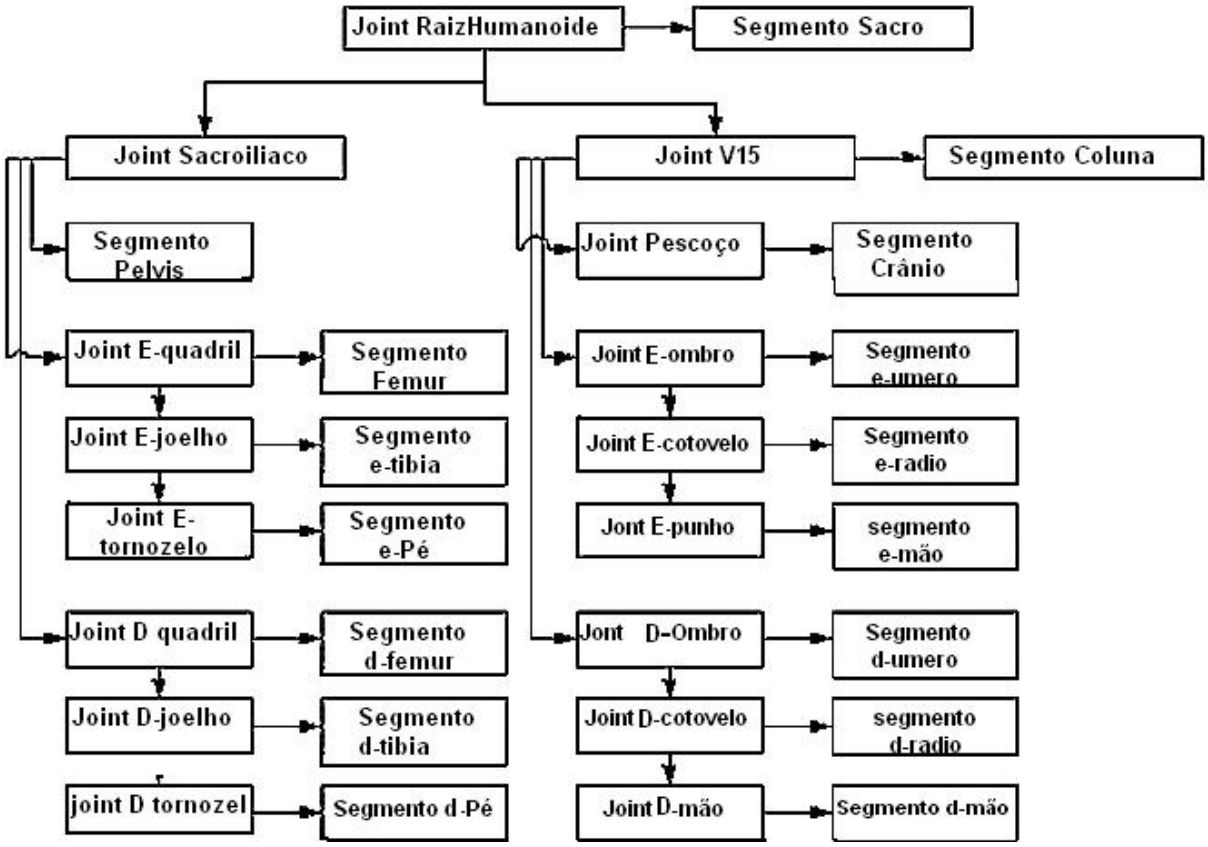


Figura 2.10 – Conjunto mínimo das articulações para um humanóide.

Cada articulação no corpo é representada por um nó *Joint* que é utilizado para definir relações entre segmentos do corpo.

Um exemplo de articulação é dado na figura 2.11, que mostra um fragmento de código Java 3D na construção do segmento omoplata do lado direito, que tem como filho o segmento ombro direito, como podemos constatar na linha 324 da referida figura.

```
314  //********************** Lado Direito ******************************
315 TransformGroup Omoplata_Dir_Humano = new TransformGroup();
316 tmpVector.set( -0.95f, 2.9f,0.0f);
317 tmpTrans.set(tmpVector);
318 Omoplata_Dir_Humano.setTransform(tmpTrans);
319
320 Ombro_Dir_Humano = new TransformGroup();
321 Ombro_Dir_Humano.setCapability(TransformGroup.ALLOW_TRANSFORM_READ);
322 Ombro_Dir_Humano.setCapability(TransformGroup.ALLOW_TRANSFORM_WRITE);
323
324 Omoplata_Dir_Humano.addChild(Ombro_Dir_Humano);
```

Figura 2.11 – Exemplo de articulação do Humanóide - lado direito.

A especificação recomenda que ao criar uma articulação, sua assinatura seja composta de:

- nome do humanóide;
- lado da articulação (se não for central, como o pescoço, a coluna): e = esquerdo; d = direito;
- nome da articulação.

Importante é definir, exatamente, a posição da articulação através do campo *center*.

2.5 – Inteligência Artificial e Agente Autônomos.

A definição de Inteligência Artificial (IA) tem sido alvo de muita discussão entre os especialistas no assunto. Na tentativa de definir com precisão o que é Inteligência Artificial, os especialistas adotam as mais diversas definições, tornando a compreensão do assunto mais complexa.

"Inteligência Artificial é o estudo de como fazer os computadores realizarem coisas que, no momento, as pessoas fazem melhor" [RICH, 1988]. Na verdade, a construção de computadores que possam realizar as tarefas difíceis, melhor do que as pessoas, tem tido um lento progresso, até mesmo abaixo da expectativa do que previam os especialistas nos primeiros dias da Inteligência Artificial. Além disso, adeptos a este conceito associam a Inteligência Artificial à construção de robôs com sentimentos. Devido a este apelo futurístico, esta definição é rejeitada. Rabuske [Rabuske, 1996] define IA como sendo a capacidade de adquirir e aplicar conhecimentos artificiais, como sendo tudo aquilo feito pelo homem, em vez de ocorrer na natureza. O mesmo autor apresenta outra definição de IA: "Inteligência Artificial é a parte da Ciência da Computação concorrente ao projeto de sistemas computacionais que exibem inteligência humana: aprender novas informações, entender linguagens, raciocinar e resolver problemas".

2.5.1 – Autonomia

Prover um humano virtual com reações e tomadas de decisão equivalentes às reais é uma tarefa ainda mais complicada que controlar o movimento deste corpo, seja ele capturado ou sintetizado. É neste ponto que se integra o usuário com a personalidade do ator virtual, demonstrando suas habilidades e inteligência na interação com o ambiente e outros agentes. Este nível de desempenho requer investimentos bastante significativos em ferramentas de tomada de decisão. Pode-se identificar o uso de pelo menos dois níveis de arquitetura possíveis:

- Otimizar a reatividade ao ambiente em baixo nível (por exemplo, na escolha do pé adequado que deve iniciar uma locomoção) [REICH, 1994];
- Executar scripts parametrizáveis ou planejar seqüências de tarefas complexas (por exemplo, escolher em que peça de uma casa deve-se procurar um determinado objeto ou ator, ou definir os passos principais envolvidos na execução de uma tarefa específica) [BADLER, 1995];

2.5.2 – Avatares como Agentes Inteligentes

Atualmente uma das grandes aplicações de avatares é a sua utilização como agentes inteligentes. Nos últimos tempos, muito tem se falado em agentes e avatares convivendo juntos em mundos virtuais. No entanto, é preciso esclarecer a diferença básica entre os dois : [RUSSELL, 2004].

- **Agente**: representação gráfica de humanos virtuais, criada e controlada por sistemas de computação;
- **Avatar**: humano virtual controlado por um participante vivo.

Neste caso, os avatares deixam de ser apenas meros representantes gráficos de usuários no mundo, para se tornarem personagens com comportamento predefinido, ou seja, agentes inteligentes que executam alguma ação depois de receberem uma seqüência de percepções. Agentes são considerados autônomos se forem capazes de conduzirem-se a si próprios de uma forma ou de outra.

Segundo [RUSSELL, 2004] existem basicamente quatro tipos de agentes:

1. **agentes reflexivos simples**: executam ações baseadas em regras cuja condição combina com a situação atual do ambiente do agente;
2. **agentes reativos com estado interno**: executam ações baseadas em regras cuja condição combina com a situação atual do ambiente, percebida através do estado interno (o estado do ambiente e do próprio agente no instante de tempo anterior) armazenado (mantêm-se informados sobre o mundo);
3. **agentes cognitivos baseados em objetivos**: executam ações baseadas em regras, cuja condição combina com a situação atual do ambiente, percebida através do estado interno armazenado e os objetivos do agente;
4. **agentes otimizadores baseados em utilidade**: executam ações baseadas em regras cuja condição combina com a situação atual do ambiente, percebida através

do estado interno armazenado e o grau de utilidade (o quanto o agente é útil) do agente.

Cada tipo de agentes possui características diversas, que possibilitam novas funcionalidades. São aplicados em mundos virtuais de acordo com o domínio do problema a ser solucionado. Por exemplo, agentes reativos executam uma ou mais ações, através dos efetuadores (atuadores), cada vez que percebem mudanças, através dos sensores, no ambiente em que vivem. Decidem qual ação tomar baseados em regras condição-ação (por exemplo, *if* percepção x *then* ação y), como mostra a figura 2.12.

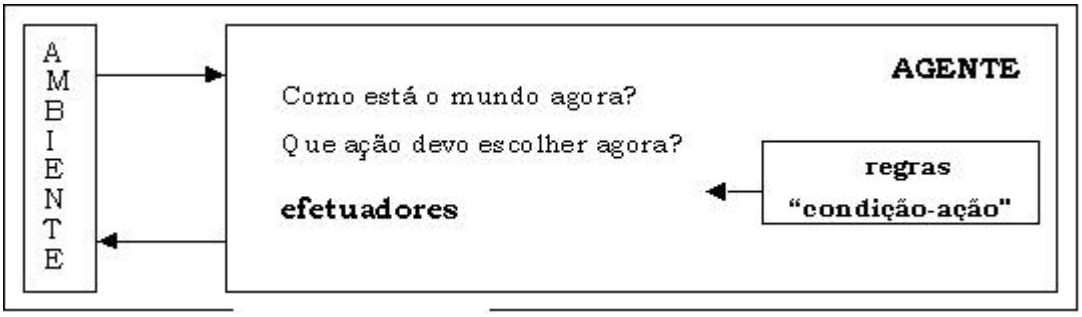


Figura 2.12: Um agente reativo.

Os agentes diferem basicamente com relação ao conhecimento que possuem sobre si mesmos (por exemplo, o agente sabe que está com bola), sobre o ambiente (por exemplo, o agente sabe que esta na lateral) e sobre outros agentes (por exemplo, a posição dos outros agentes) – agentes reativos com estado interno, e aos seus objetivos (por exemplo, o agente quer tomar um café) – agentes baseados em objetivos.

Estes variados tipos de agentes exigem a utilização de técnicas de Inteligência Artificial para a sua modelagem. Uma de suas subáreas, conhecida como "*Believable Agents*" trata da modelagem de agentes como seres que têm personalidade e que são capazes de sentir emoções como alegria, tristeza, medo, raiva etc, dando a ilusão de vida.

Investigações utilizando esta modelagem têm sido realizadas, por exemplo, para criação de ambientes de jogos de computador, educacionais, entre outros, modificando as capacidades dos personagens que fazem parte do ambiente, de forma que transmitam ilusão de vida ao usuário, sigam instruções e improvisem comportamentos.

No exemplo da Figura 2.13 pode-se ver uma cena de um jogo de tênis entre dois agentes autônomos.

Figura 2.13: Exemplo de Atores Autônomos [ISDALE, 1998]

2.5.3 – Representação do Conhecimento em IA.

Existem diversas formas de representar o conhecimento. Apesar de mostrar algumas destas formas, este trabalho tratará com mais ênfase as regras, pois estas foram escolhidas para serem implementadas na aplicação.

2.5.3.1 – Lógica

A lógica é a base para a maioria dos formalismos de representação de conhecimento, seja de forma explícita, como nos sistemas especialistas (SE's) baseados na linguagem Prolog (Programação em Lógica), seja disfarçada na forma de representações específicas que podem facilmente ser interpretadas como proposições ou predicados lógicos, por exemplo, as listas da forma:

((atributo),(objeto),(valor),(coeficiente de certeza))

2.5.3.2 – Redes Semânticas

Rede semântica é um nome utilizado para definir um conjunto heterogêneo de sistemas. Em última análise, a única característica comum a todos estes sistemas é a notação utilizada: uma rede semântica consiste em um conjunto de nodos conectados por um conjunto de arcos. Os nodos, em geral, representam objetos e os arcos, relações binárias entre esses objetos. Mas os nodos podem também ser utilizados para representar predicados, classes, palavras de uma linguagem, entre outras possíveis interpretações, dependendo do sistema de redes semânticas em questão.

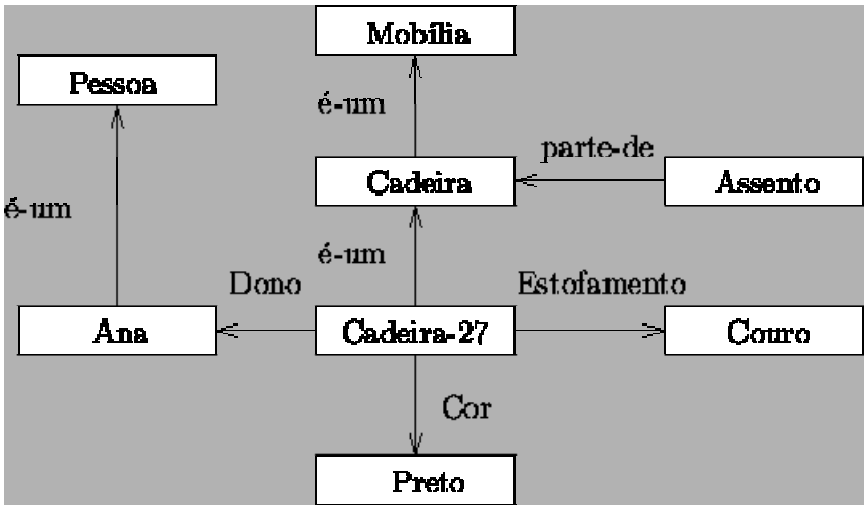


Figura 2.14 – Exemplo de Rede Semântica

2.5.3.3 – Regras de Conhecimento

"Sistemas baseados em regras não são ferramentas de otimização, mas muitas vezes são utilizados em sistemas de controles, heurísticas e outras formas de sistemas que podem prover algum nível de otimização. Esses sistemas consistem de um processo controlado por um conjunto fixo de centenas ou mesmo milhares de regras" [RUSSELL, 2004]. Na prática, as regras e suas interações são muito complexas, sendo que dificilmente a otimização pode ser garantida. De fato, sistemas baseados em regras compreendem sistemas heurísticos. Geralmente geram boas soluções, entretanto existem casos nos quais as regras demonstram um desempenho ruim.

Segundo Fernandes [FERNANDES, 2003], "Existe um caso no qual sistemas baseados em regras sempre atingem uma otimização real. Se tiver sido provado que as regras utilizadas sempre gerem uma solução ótima, para qualquer estado do processo, o sistema pode ser enquadrado dentro dos níveis de otimização real". Infelizmente, essas provas usualmente se transformam em problemas mais complexos que a aplicação original, inviabilizando essa conclusão. Tipicamente o maior problema associado aos sistemas baseados em regras é o da formulação correta e consistente das regras a serem aplicadas.

A figura 2.15 mostra a arquitetura de um sistema baseado em regras, que é composto por uma base de regras responsável por armazenar regras e por estar separado da máquina de inferência. O conhecimento contido na base é fácil de ser modificado. A máquina de inferência é responsável por procurar as respostas na base de regras e avaliá-las para uma ordenação lógica, a memória de trabalho contem o problema a ser resolvido. A união da base de regras com a máquina de inferência gera o aprendizado.

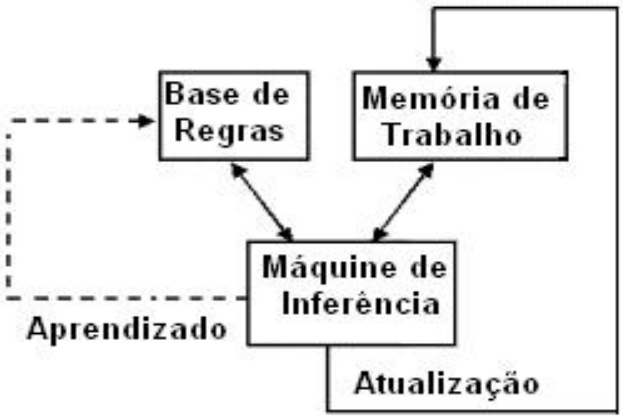


Figura 2.15 – Arquitetura dos Sistemas Baseados em Regras.

Abaixo são mostrados alguns tópicos de conceitos que são importantes para o entendimento de regras:

- Regras If-then são as mais bem sucedidas formas de representação de conhecimento. Elas são fáceis de serem criadas e entendidas pelas pessoas;
- Uma regra tem duas partes, uma parte If ou antecedente, e a parte then ou conseqüente. Uma condição numa regra é chamada de cláusula;
- Fatores de confiança são usados como modificadores em situações onde ou a validade dos dados ou a aplicabilidade da regra é incerta

2.6 – Algoritmos Genéticos

Um algoritmo genético (AG) é um procedimento iterativo que aborda as idéias da seleção natural e da "sobrevivência dos melhores", através da evolução natural. Ao simular a evolução natural, um AG pode facilmente resolver problemas complexos. Um AG controla uma população ou grupo de candidatos para uma determinada função. As soluções dos candidatos representam um codificador de problemas na mesma forma como é análoga aos cromossomos de sistemas biológicos. Cada cromossomo é constituído de uma string de genes e os valores destes, são chamados de alelos. Os cromossomos são representados como uma string de bits. Entretanto, a utilização de valores numéricos ou de ponto flutuante também é válida. O cromossomo tem um valor de adaptação (fitness), que é verificado para saber suas chances de sobrevivência e reprodução para sobrevivência. [MICHALEWICZ, 2000]

A Figura 2.16 apresenta o diagrama geral do ciclo de vida de um algoritmo genético

```
1  */ Pseudo Código do funcionamento geral de Algoritmo Genético
2
3  t = 0 ;
4  inicia_população ( P , t ) ;
5  avaliação ( P , t ) ;
6  repita até ( t = d )
7  {
8  t = t +1;
9  seleção_dos_pais ( P, t ) ;
10 recombinação ( P , t ) ;
11 mutação ( P , t ) ;
12 avaliação ( P , t ) ;
13 sobrevivem ( P , t )
```

Figura 2.16 – Pseudo-Código do Funcionamento geral de um AG

2.6.1 – Computação evolutiva

A Computação Evolutiva (CE) é um ramo da ciência da computação que se embasa em um novo paradigma para a resolução de problemas, que não exige o conhecimento de uma sistemática prévia de resolução, e baseia-se nos mecanismos encontrados na natureza, à luz da teoria da evolução natural de Darwin. A Computação Evolutiva constitui uma ramo da Computação Natural, e envolve tópicos de vida artificial, geometria fractal, sistemas complexos e inteligência computacional. Fazem parte dos estudos deste campo os Algoritmos Genéticos (AGs), as Redes Neuronais (RNs) e os Sistemas Especialistas (SEs). [BITTENCOURT, 1998]

2.6.1.1 – Funcionamento

O princípio básico do funcionamento dos Algoritmos Genéticos È que um critério de seleção vai fazer com que, depois de muitas gerações, o conjunto inicial de indivíduos gere indivíduos mais aptos. A maioria dos métodos de seleção são projetados para escolher preferencialmente indivíduos com maiores notas de aptidão, embora não exclusivamente, afim de manter a diversidade da população. Um método de seleção muito utilizado È o método da Roleta, onde indivíduos de uma geração são escolhidos para fazer parte da próxima geração, através de um sorteio de roleta. O método roleta é muito utilizado devido sua facilidade de implementação e confiabilidade na seleção de individual

2.6.2 – Método Roleta

O método roleta é o método de seleção mais utilizado. Os indivíduos de uma geração (ou população) são selecionados para a próxima geração utilizando uma roleta. Cada individuo de uma população é representado na roleta por uma fatia proporcional ao seu índice

de aptidão. Assim indivíduos com maiores aptidão ocupam fatias maiores da roleta, enquanto indivíduos de aptidão mais baixa ocupam fatias menores.

Neste método, cada indivíduo da população È representado na roleta proporcionalmente ao seu índice de aptidão, como È mostrado na figura 2.17. Assim, aos indivíduos com alta aptidão, é dada uma porção maior da roleta, enquanto aos de aptidão mais baixa, é dada uma porção relativamente menor da roleta.Finalmente, a roleta È girada um determinado, número de vezes, dependendo do tamanho da população, e são escolhidos, como indivíduos que participarão da próxima geração, aqueles sorteados na roleta.

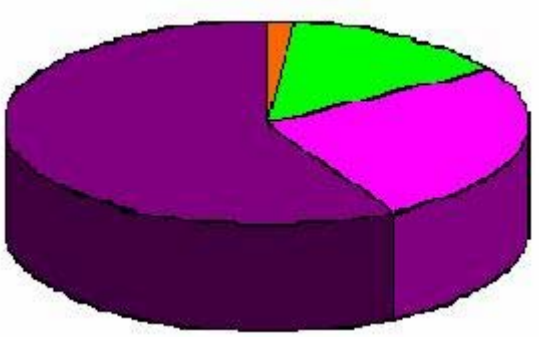


Figura 2.17 – Distribuição de fatias por aptidão dos indivíduos.

A implementação deste método bem como seu código e outras informações podem ser visualizadas no capítulo 3 especificamente na seção 3.9, que fala sobre a forma de funcionamento da Inteligência e Autonomia.

Capítulo 3

3 . RESULTADOS OBTIDOS

3.1 – O ambiente Virtual – Estádio de Futebol

O ambiente foi desenvolvido com todos os TransformGroup (MeioCampo, Esq.Campo, base, placas, sol, placar, coluna, escudo, banco, dir.campo, frentecampo, fundo campo) sendo filhos do TransformGroup CriaGramado. Optou-se por esta forma pelo fato de que caso seja necessário fazer algum tipo de escalonamento (diminuir ou aumentar), bastaria fazer na classe principal que todos (TransformGroup) serião afetados.

A figura 3.1 mostra o grafo de cena gerado para a construção do ambiente virtual (estádio)

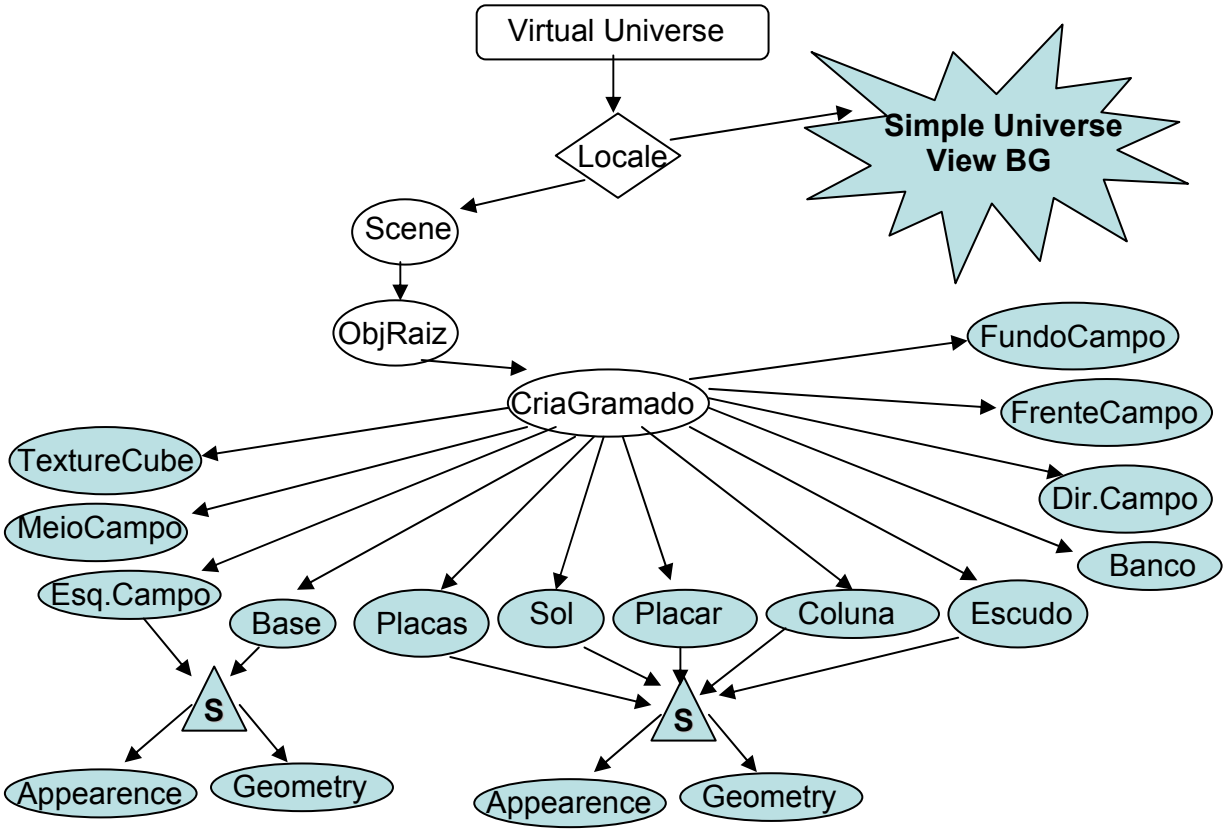


Figura 3.1 – Grafo de Cena do Estádio.

As figuras de 3.2.à 3.7 mostram algumas visões do estádio de futebol

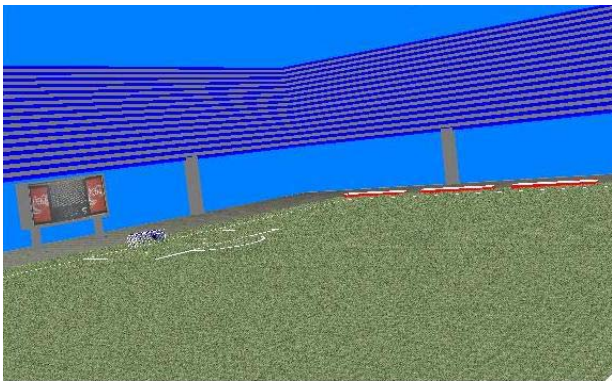


Figura 3.2- Arquibancada

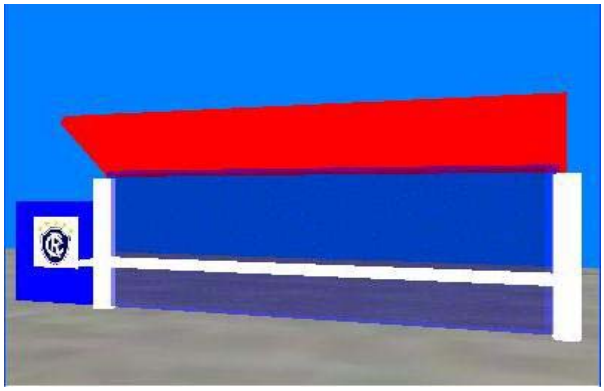


Figura 3.3- Banco de reservas

Figura 3.4- Placar Eletrônico

Figura 3.5- Gramado

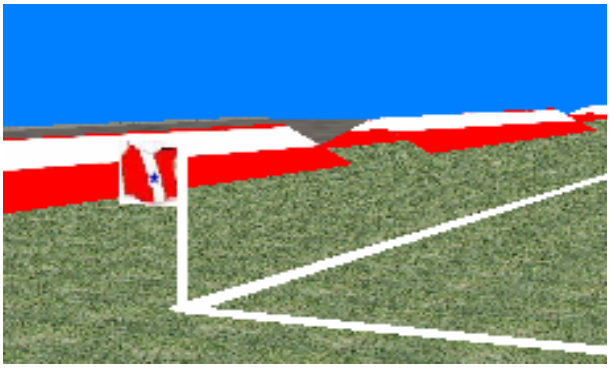


Figura 3.6- Bandeira de Escanteio

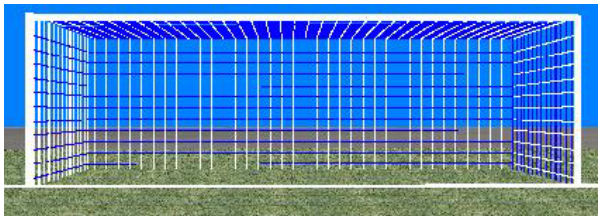


Figura 3.7- Traves

3.2 – Interface do FUT – 3D

A interface do FUT – 3D tenta ser o mais simples possível para atender os requisitos de uma boa GUI, que são entre outros:

Diminuição do tempo de aprendizagem do usuário sobre a aplicação;

Organização das informações.

A interface oferece ainda uma barra de menu conforme a figura 3.8, onde se encontram as opções (Menu, Visão, Estilo da Janela, Ajuda), que serão descritos a seguir:

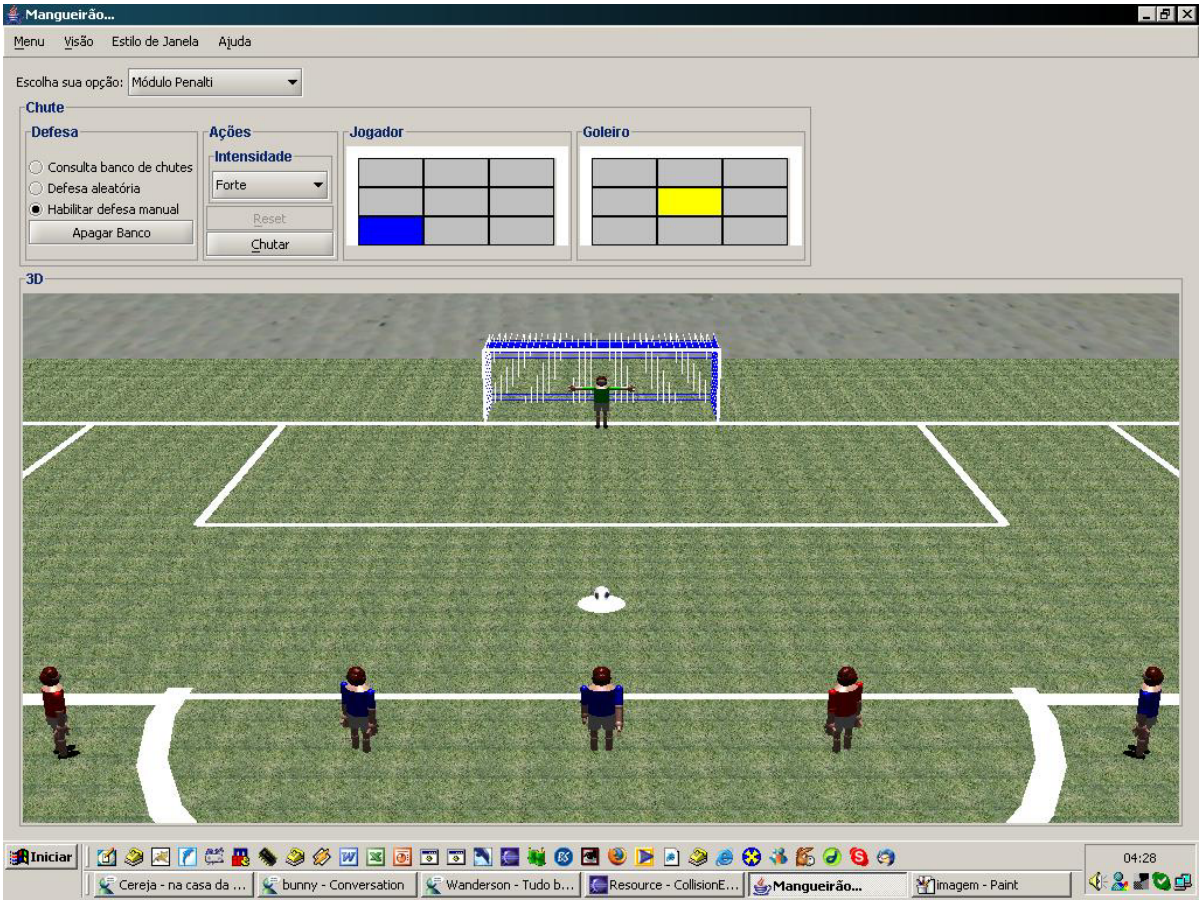


Figura 3.8 – Janela do Fut 3D Composta de AV e Gui.

- **Menu ⇨** Contém as opções para iniciar um novo jogo e sair, a figura 3.9 mostra a tela de menu;

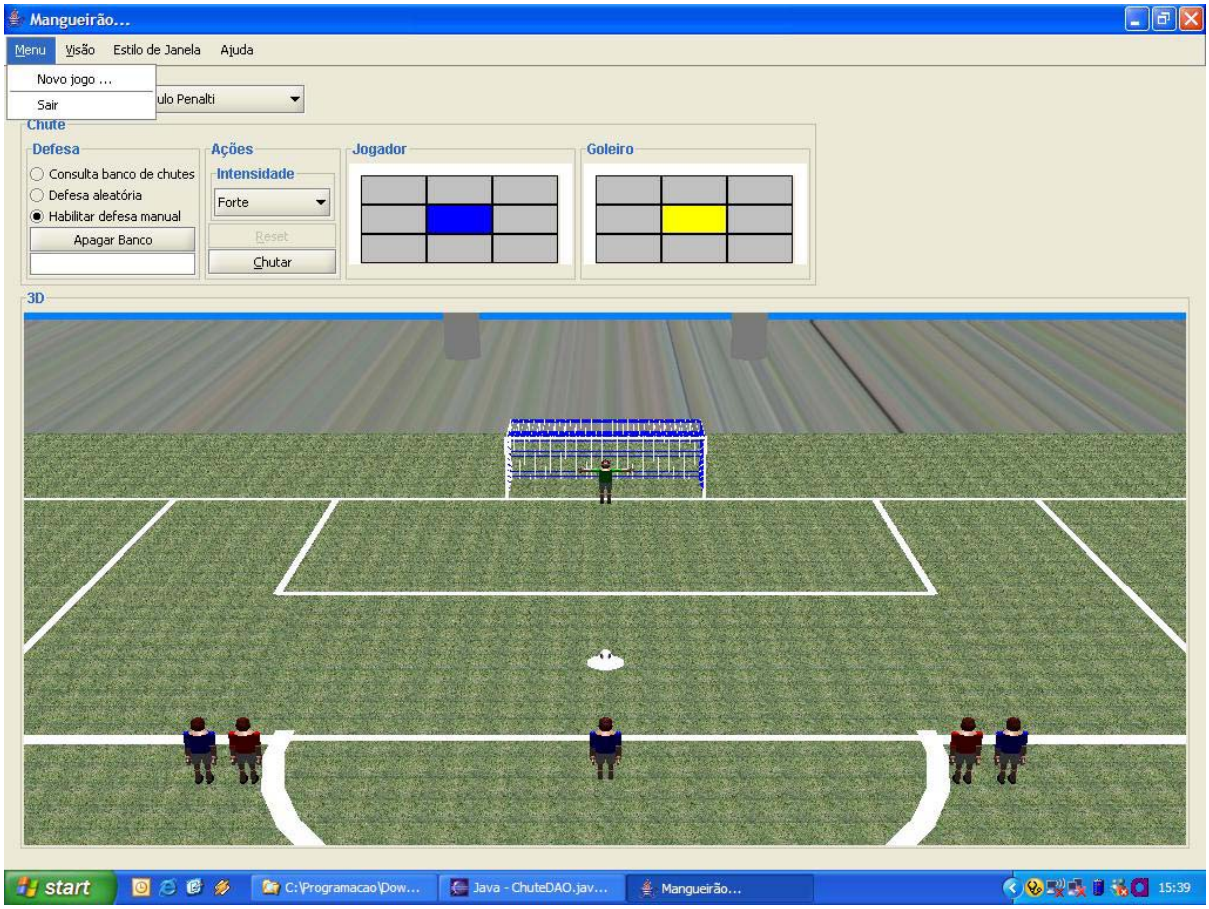
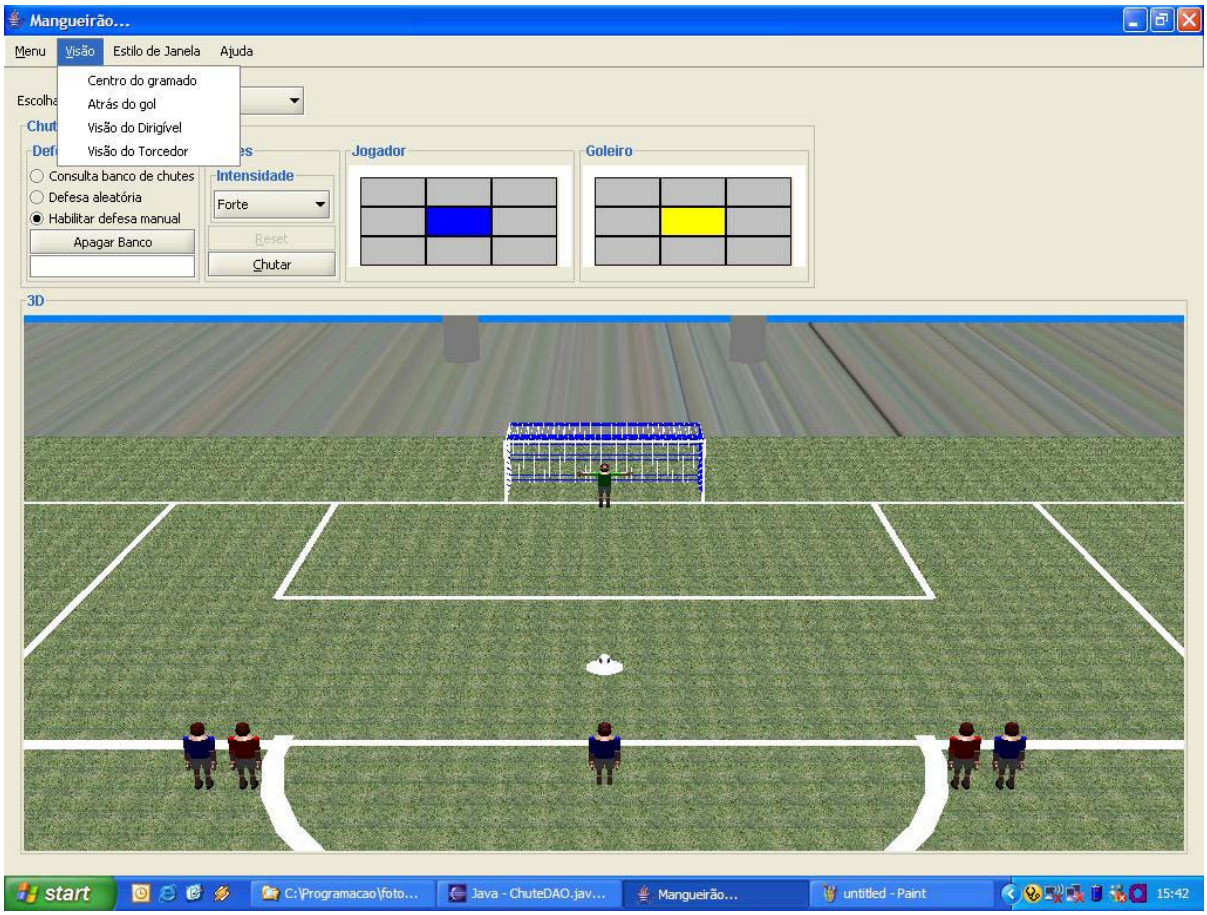


Figura 3.9. – Opção menu

- **Visão ⇨** Contém as opções, a figura 3.10 (a) mostra a tela de visão:



3.10(a).- Menu Visão

- Visão do Dirigível ⇨ Apresenta a perspectiva de visualização de tela como se fosse vista do alto de um dirigível, como mostra a figura 3.10(b);

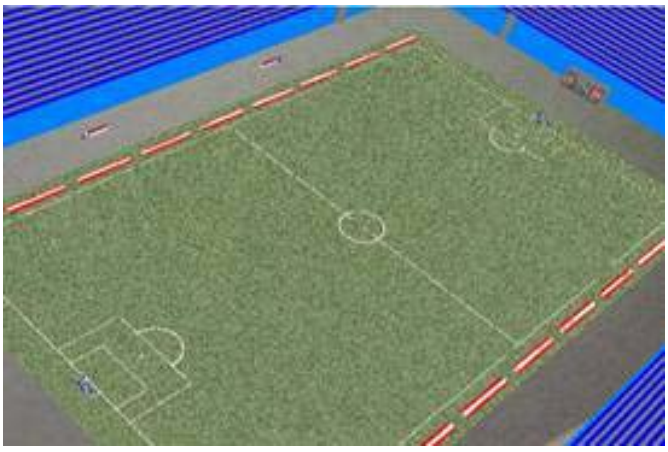


Figura 3.10(b) – Visão do dirigível

- Atrás do gol ⇨ Apresenta uma visão da perspectiva de quem está sentado na arquibancada dos fundos, conforme mostra a figura 3.10(c);

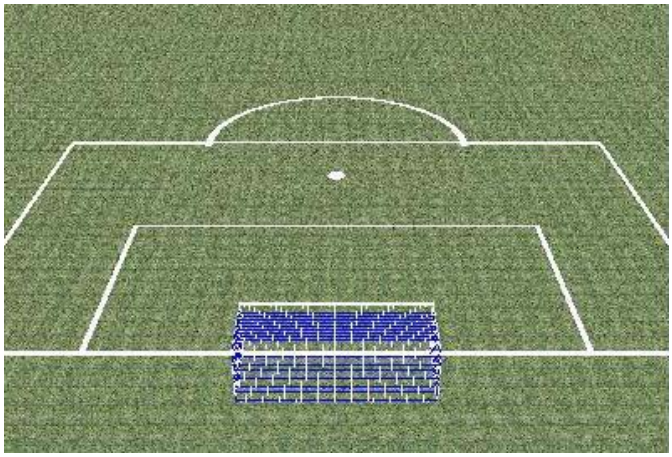


Figura 3.10(C) – Visão Atrás do gol.

- Visão do Lateral ⇨ Apresenta uma perspectiva do torcedor sentado nas arquibancadas laterais do estádio, conforme mostra a figura 3.10(d);

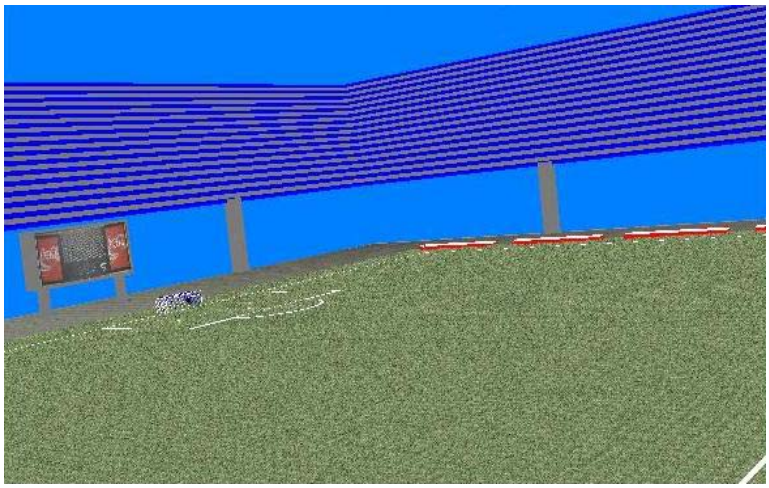


Figura 3.10(D) – Visão Lateral.

- **Estilo da Janela** ⇨ Apresenta a perspectiva de visualização das janelas de acordo com as seleções que podem ser (Windows, Java 3D, Java, Linux);

- **Ajuda** ⇨ Contém uma página web que foi desenvolvida, para servir como um manual, que pode ser observada na figura 3.11.

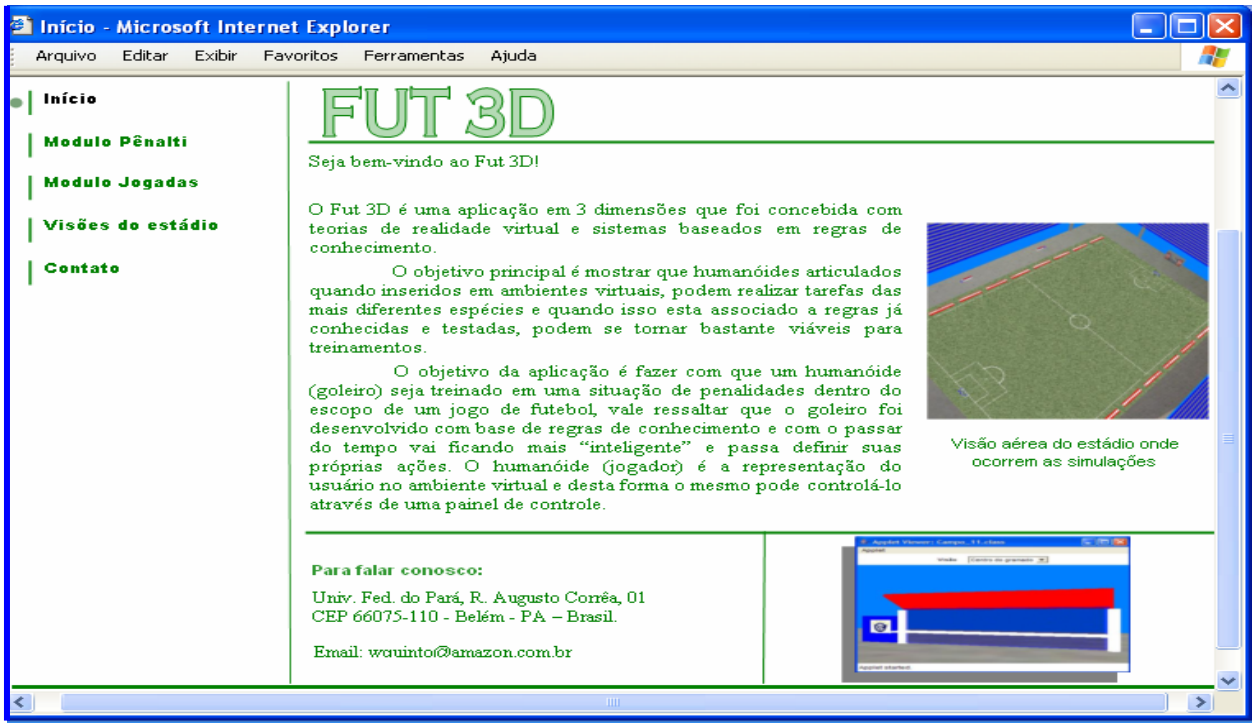


Figura 3.11 – Pagina web para servir como ajuda.

A interface apresenta também uma barra de tarefas conforme figura 3.8 onde se pode escolher a intensidade do chute (Fraco, Médio, Forte). Na parte referente ao jogador, se pode escolher a posição do chute – vale ressaltar que a visão da tabela é a mesma que o jogador tem do gol, caso da figura, o chute será disparado pro lado esquerdo e rasteiro com a intensidade forte.

Através da interface o usuário pode interagir com o ambiente tanto através do mouse quanto do teclado. Caso se utilize o teclado, vale ressaltar que as teclas só terão funcionalidade no módulo pênalti. As teclas com funções são:

A - Faz com que o jogador ande pelo mundo virtual;

R - Reinicia a jogada;

C - Faz com que o jogador chute a bola em direção ao gol;

↓ - Move para baixo;

↑ - Move para cima;

← - Move para esquerda;

→ - Move para direita;

3.3 – Especificação H – Anim 1.1 com Java 3D.

Durante muito tempo a especificação H-ANIM1.1 teve seu desenvolvimento atrelado ao VRML, um padrão internacional, que é a abreviação de Virtual Reality Modeling Language, ou Linguagem para Modelagem em RV.Trata-se de uma linguagem independente de plataforma, que permite a criação de cenários tridimensionais, nos quais pode-se passear, visualizar objetos por ângulos diferentes e interagir com esses objetos do ambiente [HARTMAN, 1996], [ROEHL, 1997]. A linguagem foi concebida para descrever simulações interativas de múltiplos participantes em mundos virtuais. Com VRML, pode-se criar simulações interativas que incorporam animação, noções de leis físicas e suporte a vários usuários através de uma rede, em especial através da Internet.

Para navegar pelo ambiente tridimensional é necessário um navegador específico. Estes programas são gratuitos. Abaixo, estão dois deles e o local onde podem ser encontrados em, CosmoPlayer 2.1 em http://www.cosmosoftware.com e Cortona 3.0 em http://www.parallelgraphics.com.

Os fatores acima listados tornam o VRML uma linguagem de pouca ou nenhuma portabilidade, escalabilidade, facilidade de uso e visualização. Desta forma, este trabalho rompe com este paradigma e propõe a construção da especificação H-Anim 1.1, através da API JAVA 3D, que como já foi visto no capítulo 02, apresenta uma hierarquia de classes Java que serve como interface para o desenvolvimento de sistemas gráficos tridimensionais. Possui construtores de alto nível que permitem a criação e a manipulação de objetos geométricos, especificados em um universo virtual. Também possibilita a criação de universos virtuais com uma grande flexibilidade, onde as cenas são representadas através de grafos, e os detalhes de sua visualização são gerenciados automaticamente. Assim, o desenvolvimento de um programa Java 3D se resume na criação de objetos e no seu posicionamento em um grafo de cena, que os combina em uma estrutura de árvore.

A figura 3.12 mostra exemplos da codificação utilizada para a construção do humanóide, no caso especifico da ligação entre a cabeça e o pescoço, onde se fez necessária a construção de uma base para servir como elo entre a cabeça e o pescoço, pois somente desta forma poderíamos dar a mobilidade e flexibilidade esperada para o movimento da cabeça.

```
201 // Cria o TransformGroup da Base da Cabeça
202 Base_da_Cabeça = new TransformGroup();
203 tmpVector.set(0.0f, 3.632f, 0.0f);
204 tmpTrans.set(tmpVector);
205 Base_da_Cabeça.setTransform(tmpTrans);
206
207 // Ajusta a esfera para a cabeça
208 tmpSphere = new Sphere(0.5f, Sphere.GENERATE_TEXTURE_COORDS
209         | Sphere.GENERATE_NORMALS, pintura2);
210
211 // Adiciona a cabeça a base
212 Base_da_Cabeça.addChild(tmpSphere);
213
214 // Pescoço
215         TransformGroup Pescoço = new TransformGroup();
216         Cone pescoço = new Cone(0.75f, 0.48f, Cone.GENERATE_
217         NORMALS| Cone.GENERATE_TEXTURE_COORDS, pintura2);
218         tmpVector.set(0.0f, -0.4f, 0.0f);
219         tmpTrans.set(tmpVector);
220         tmpTrans.setScale(escala);
221         Pescoço.setTransform(tmpTrans);
222         Pescoço.addChild(pescoço);
223         Base_da_Cabeça.addChild(Pescoço);
```

Figura 3.12: Exemplo da Criação da Cabeça e Pescoço.

A especificação trabalha em forma de hierarquia, na qual um nó deve ser herdeiro do outro. Desta forma, a ligação entre as partes do humanóide foi feita através do recurso addChild, como podemos verificar nas linhas 205, 212, 223; presentes na figura 3.12.

3.4 - EVOLUÇÃO DO PRÓTÓTIPO

3.4.1 - FASE 1

Durante a fase de construção do jogador, muitas implementações foram testadas, mas poucas tiveram os resultado que a especificação H-ANIM 1.1 trouxe. Além do mais, era necessário prioritariamente entender como o corpo humano estava constituído e como seus membros estavam inter-relacionados.

O primeiro jogador desenvolvido era muito básico, mas já trazia as definições que seriam usadas para modelar o corpo, ou seja, optou-se pela construção de um humanóide que utilizasse somente forma básicas como (Cilindros, Box, Cones, Esfera), nesta etapa o objetivo era construir um modelo para o corpo, o que foi conseguido e pode ser visualizado na figura 3.13.

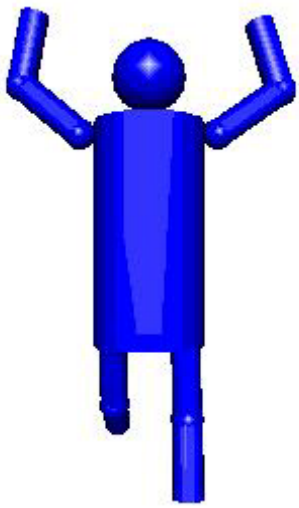


Figura 3.13 – Primeiro protótipo criado.

O próximo passo a ser desenvolvido era fazer o mapeamento da estrutura hierárquica dos membros do jogador segundo a especificação utilizada. Conforme pode ser visulizado na figura 3.14.

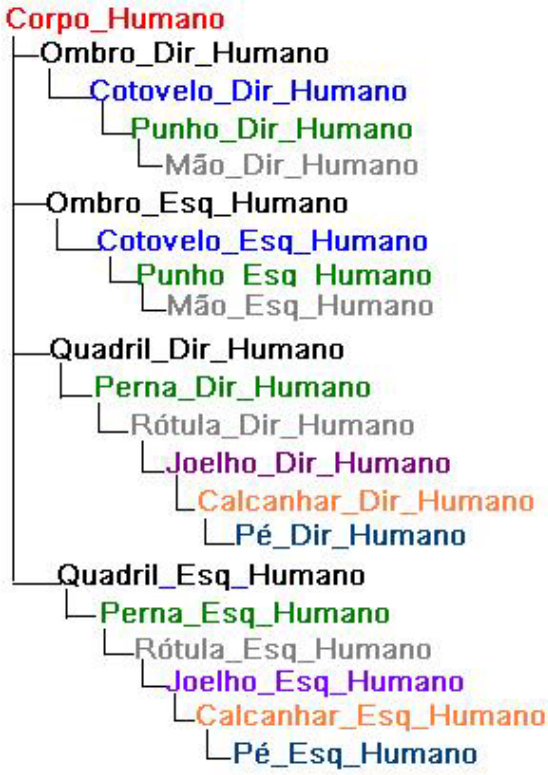


Figura 3.14 – Mapeamento hierárquico dos membros do jogador segundo especificação.

Uma vez construído o avatar pelo mapeamento das articulações, segundo a especificação H-ANIM1.1, havia a necessidade de construir uma aplicação em que se pudesse testar efetivamente o humanóide e suas articulações, o resultado é o protótipo visto na figura 3.15,

onde através da interface pode se movimentar os ombros, cotovelos, joelhos e pernas do protótipo.

Figura 3.15 – Aplicação desenvolvida para testar as articulações.

Vale ressaltar que para se conseguir chegar a tal estrutura utilizou-se os conceitos básicos da linguagem, que são (translações, rotações, texturização,entre outras...).

3.4.2 FASE 2

Era necessária uma aparência mais agradável do jogador. Para tal, foram feitas inúmeras tentativas para dar ao humanóide uma aparência mais realística, faltava ainda um rosto e uma vestimenta condizentes com sua situação de jogador, o que foi providenciado como se pode observar pela figura 3.16:

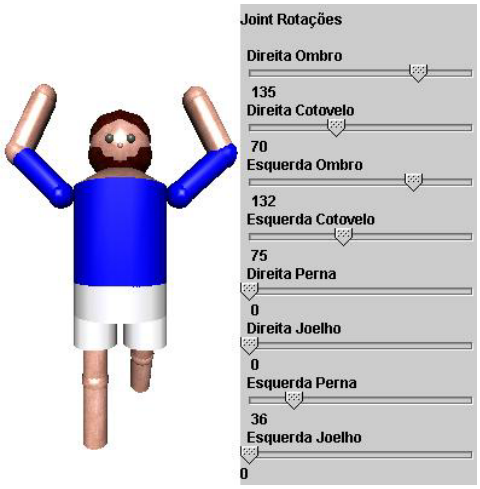


Figura 3.16 – Humanóide/Jogador com rosto e vestimentas.

3.4.3 - FASE 3

Com o avatar pronto e funcionando segundo a especificação adotada, restava inserir o mesmo no seu ambiente de atuação (estádio de futebol), onde ele poderá desenvolver suas atividades. A figura 3.17 mostra o jogador dentro do estádio já com a bola, sua principal ferramenta de trabalho.

Figura 3.17 Jogador dentro do estádio

3.5 – Protótipo - Modulo Pênalti

Todos os avatares foram criados como threads com o objetivo de se conseguir um maior controle de animação.

3.5.1 - Defesa Manual

O goleiro, quando esta opção é selecionada, também pode ser controlado bastando que, para isso, o usuário selecione para onde o mesmo deverá se movimentar. Pode se optar por selecionar o goleiro e deixá-lo parado na posição definida antes do chute, ou se pode esperar o jogador chutar a bola para o usuário realizar a ação do goleiro. A figura 3.18 mostra a opção habilitar defesa manual marcada.

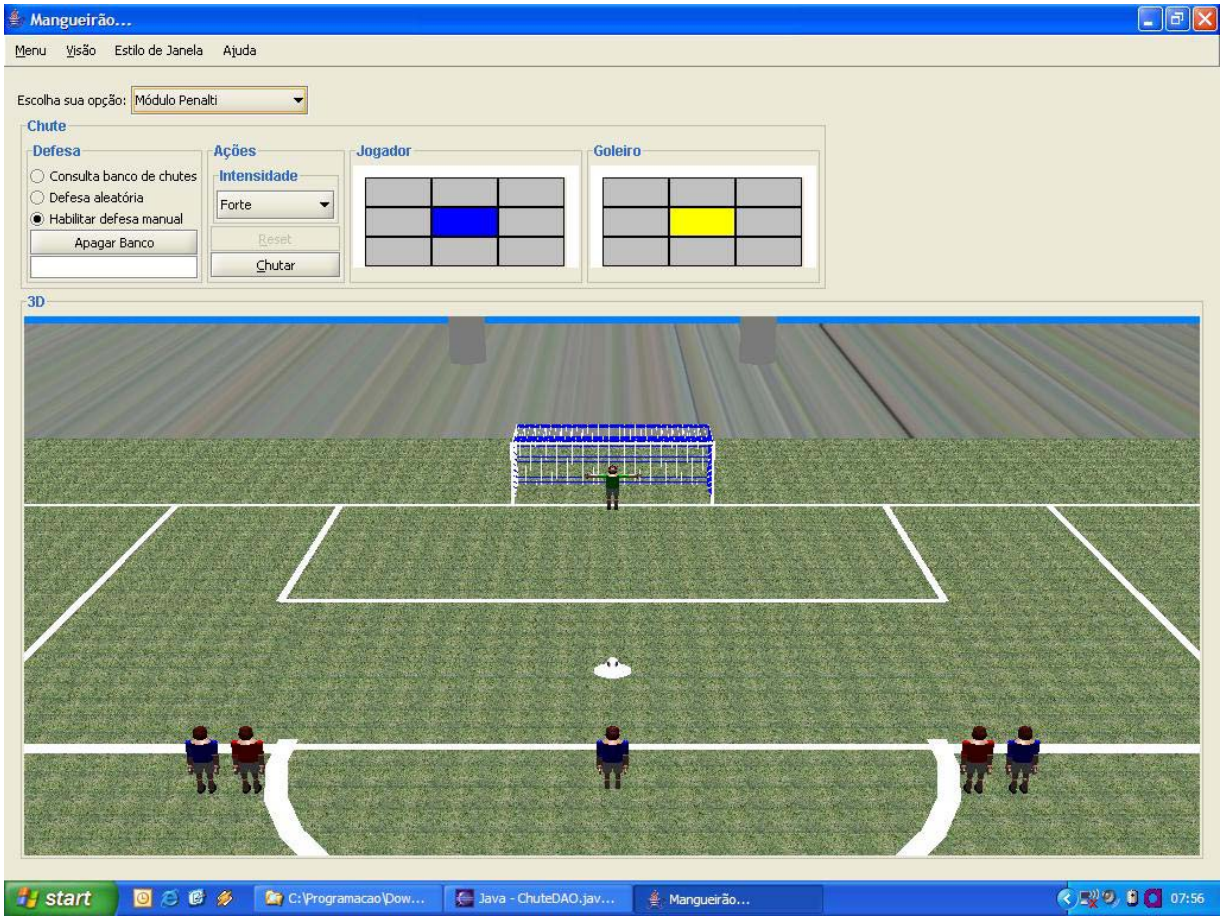


Figura 3.18 – Modulo Pênalti – Defesa Manual.

A idéia desta maneira de interação é proporcionar uma avaliação do comportamento do ambiente, sendo assim, a ação tentar agarrar um chute (rebote – colisãob), depende de se escolher o tempo certo para que o goleiro chegue até a bola e após a ação chutar, avaliar o comportamento que o ambiente vai ter em relação ao rebote do goleiro.

Podem-se escolher outras formas de interação sendo que independente da escolha o goleiro sempre está “aprendendo”, guardando no banco de chutes as suas características e possibilidades de chutes.

3.5.2 – Defesa Aleatória

Escolhendo esta forma de interação, o usuário só terá condições de manipular o jogador, pois o goleiro irá escolher uma posição para se movimentar de forma aleatória.

A idéia desta forma de interação é a de que o usuário irá tentar marcar um gol numa situação muito corriqueira do cotidiano futebolístico, onde o goleiro que não tem, na maioria das vezes, um treinamento especifico para agarrar pênaltis, tenta escolher um canto de forma aleatória ou ficar parado no centro do gol. A figura 3.19 apresenta a tela da opção defesa aleatória habilitada.

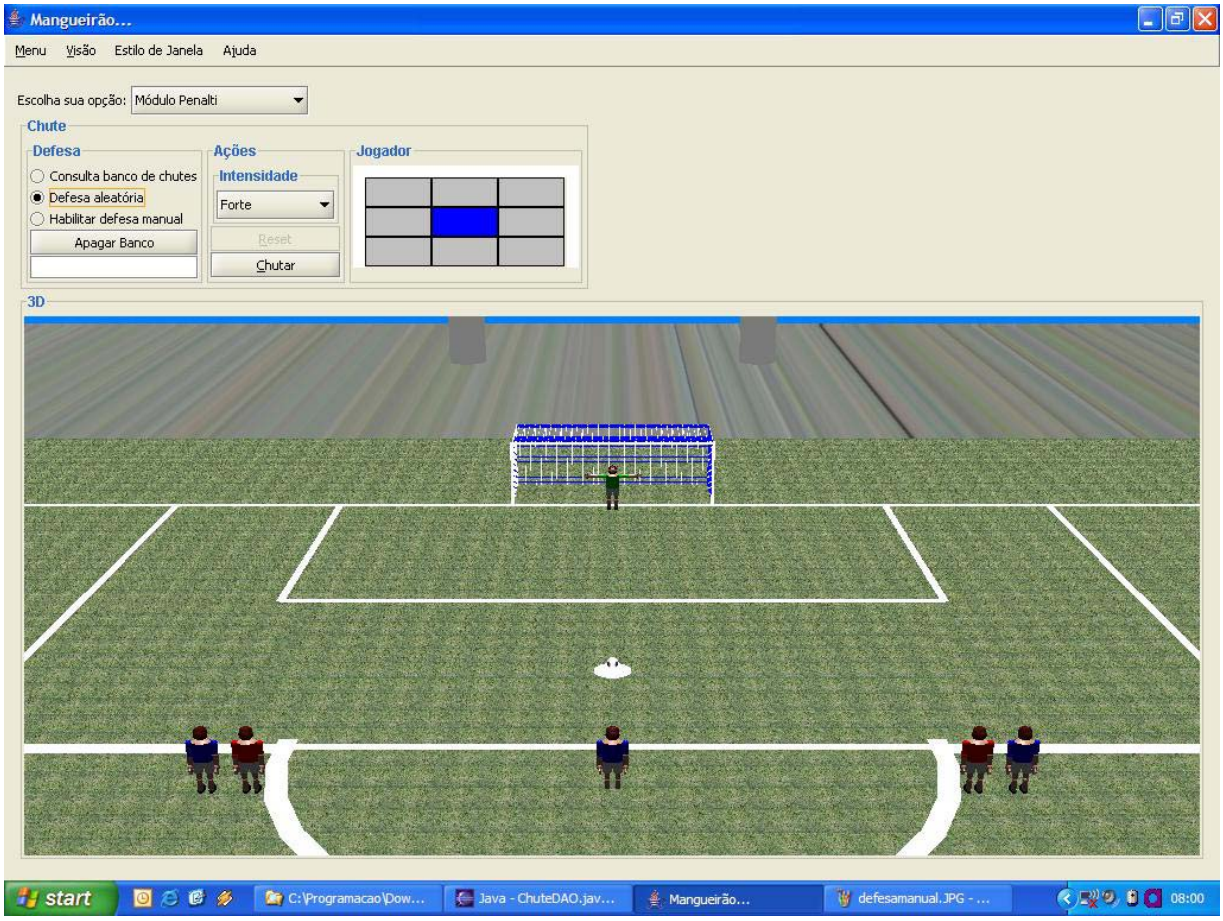


Figura 3.19 – Modulo Pênalti – Defesa Aleatória.

3.5.3 – Consulta Banco de Chutes

Escolhendo esta forma de interação, o usuário só terá condições de manipular o jogador, pois o goleiro irá escolher uma posição para se mexer através de consultas ao banco de dados que armazena todos os chutes desferidos em direção ao gol. A defesa do goleiro é baseada na técnica de computação evolucionária chamada Método Roleta, o que será explicado no capítulo 3. A figura 3.20 mostra a tela com a opção consulta banco de chutes habilitada.

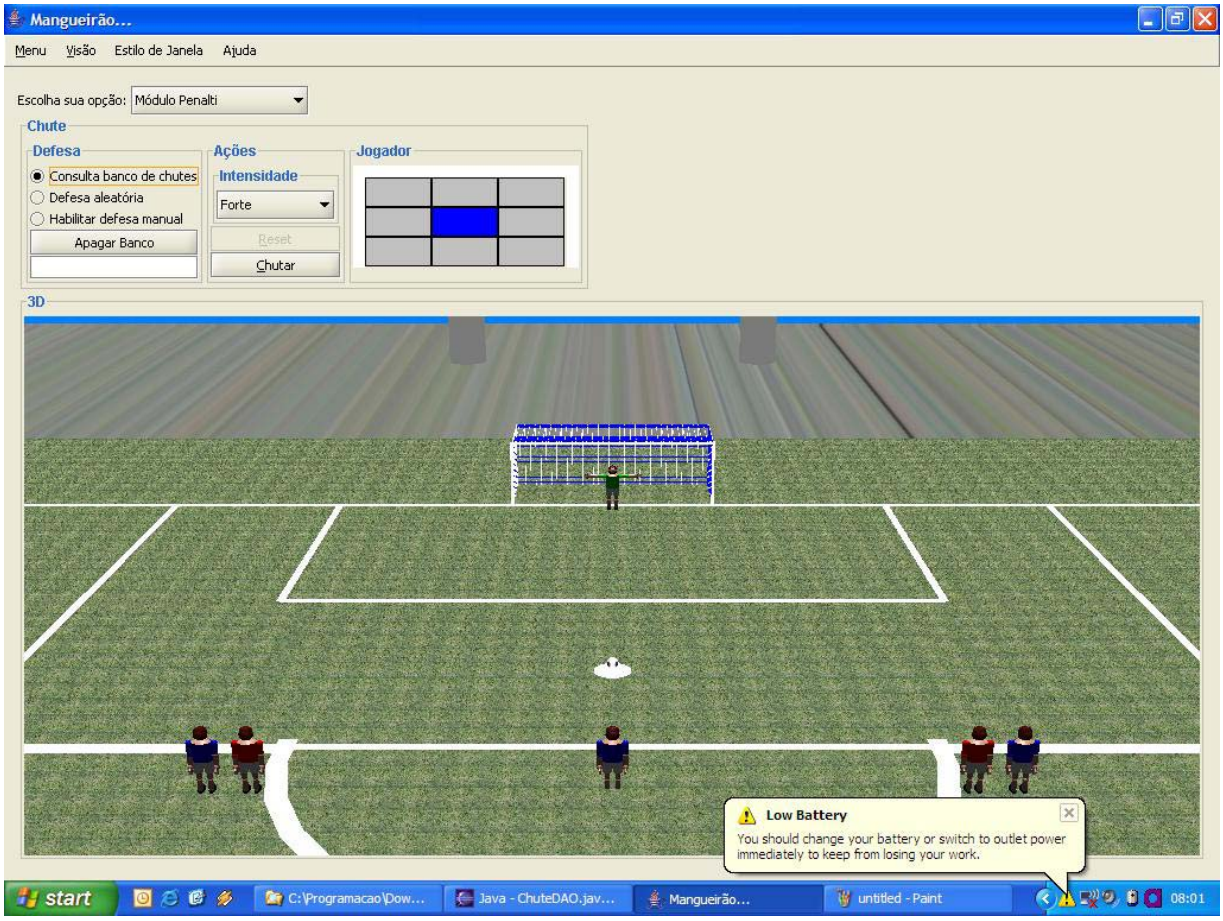


Figura 3.20 – Modulo Pênalti – Defesa Baseada no Banco de Chutes

3.5.4 – O rebote do goleiro

O rebote é a parte mais importante do módulo de pênalti, pois a partir dele é que ocorre a autonomia dos jogadores.

O rebote favorável para os jogadores ocorre somente quando o chute é forte. Desta forma, se o chute for efetuado na direita, os jogadores da esquerda disputarão a bola. Caso o jogador vencedor da disputa seja o zagueiro, o mesmo chuta para fora. Para o jogador atacante há duas formas de ação, baseadas em aleatoriedade:

1. Tocar a bola para o jogador do meio, e este chutar para o gol;
2. Chutar para o gol no canto direito, no centro ou no canto esquerdo;

Quando os chutes rebatidos tiverem a intensidade média, o goleiro corre até a bola e chuta para longe do gol, e se tiverem intensidade fraca, o goleiro permanece com a bola até a jogada ser reiniciada.

As figuras (3.21(a), 3.21(b), 3.21(c), 3.21(d)) mostram a sequência de cenas para o rebote do goleiro e trajetória da bola até encontrar um jogador que pode ser de ataque (azul) ou de defesa (vermelho).

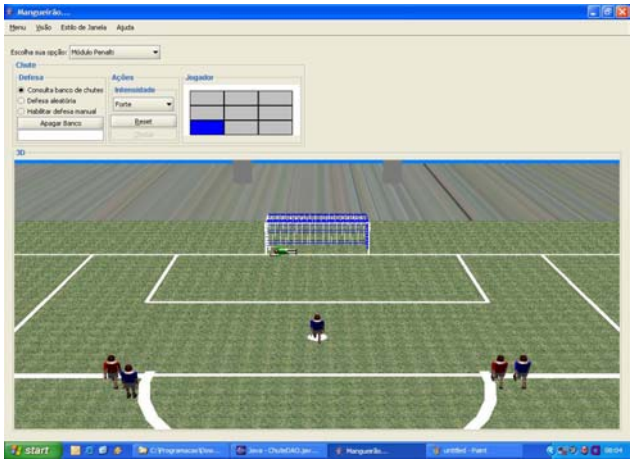


Figura 3.21(a) – Goleiro defendendo o chute

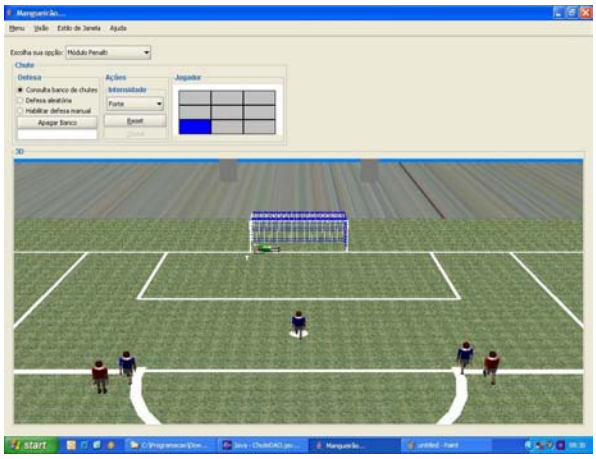


Figura 3.21(b) – Bola saindo após defesa

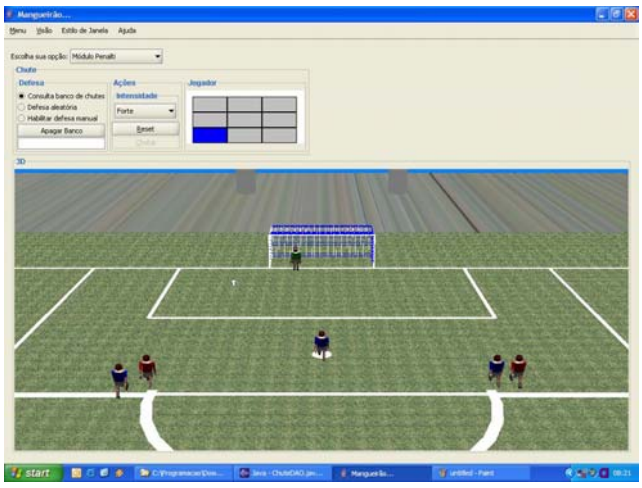


Figura 3.21(c) – Bola saindo após defesa

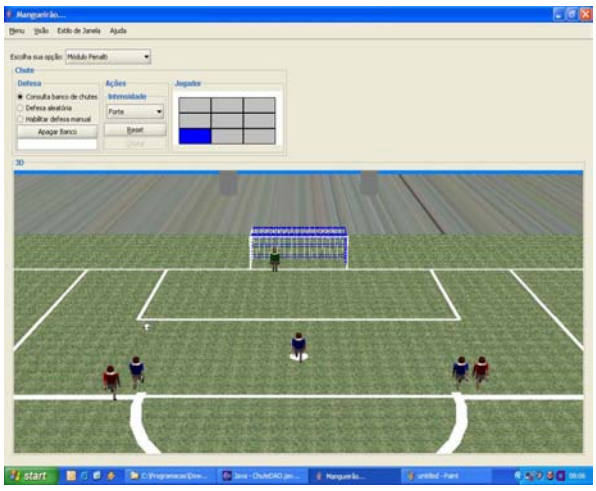


Figura 3.21(d) - Bola saindo após defesa

As figuras (3.22 (a), 3.22 (b), 3.22 (c), 3.22 (d), 3.22 (e), 3.22 (f)) mostram a sequência de cenas para o rebote do goleiro com dominio de bola do jogador atacante e com toque de bola.

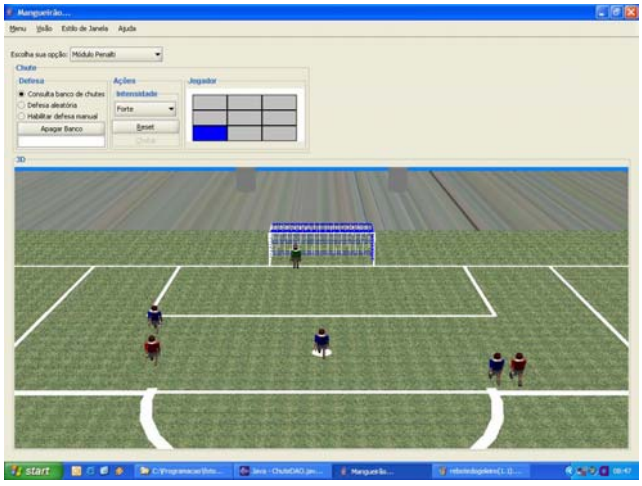


Figura 3.22 (a) – Rebote do goleiro com dominio de bola pelo atacante.

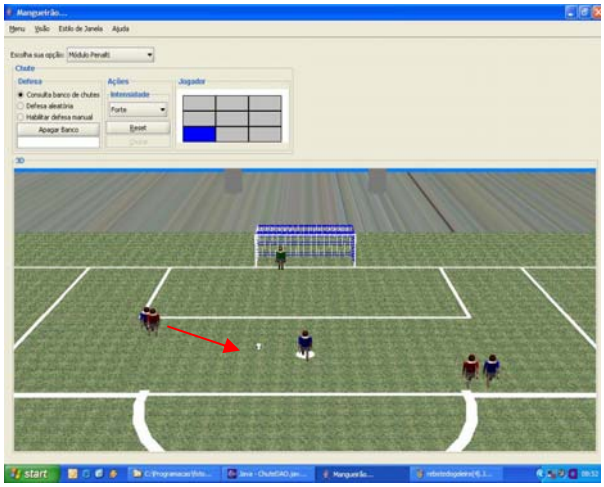


Figura 3.22 (b) - Rebote do goleiro com dominio de bola pelo atacante

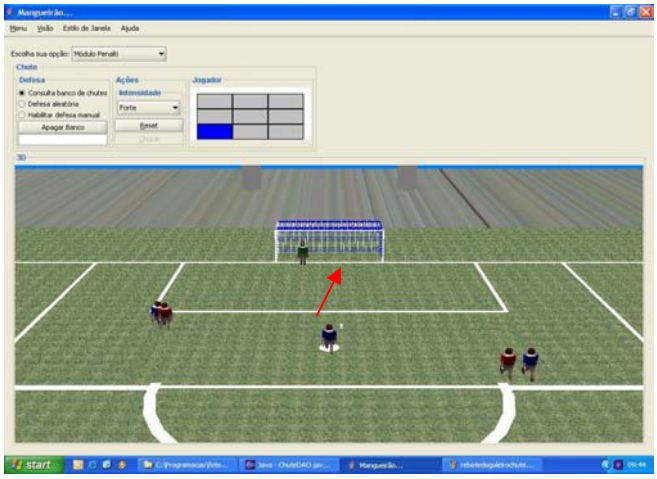


Figura 3.22 (c) - Rebote do goleiro com dominio de bola pelo atacante

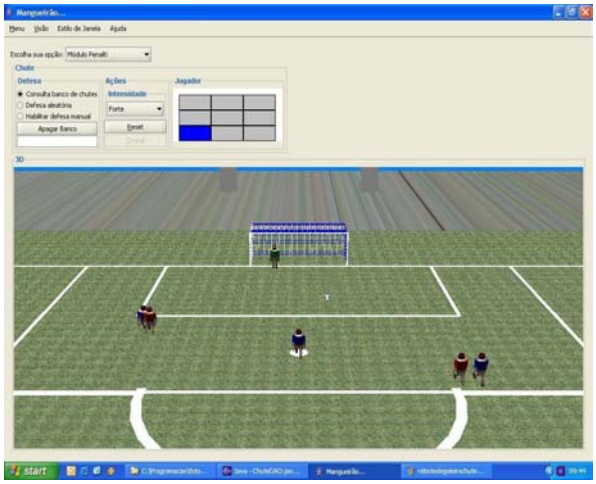


Figura 3.22 (d) - Rebote do goleiro com dominio de bola pelo atacante

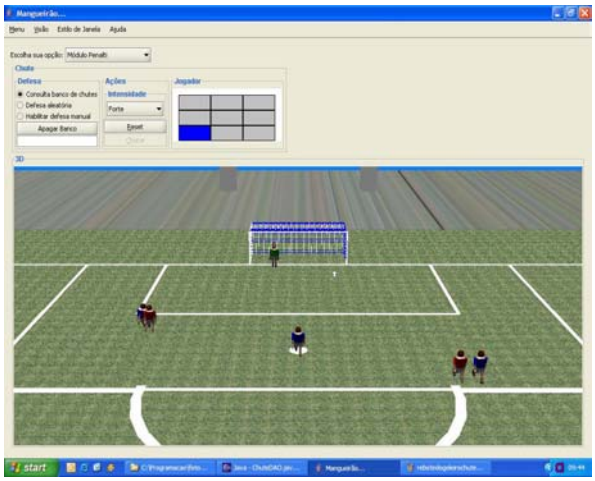


Figura 3.22 (e) - Rebote do goleiro com dominio de bola pelo atacante

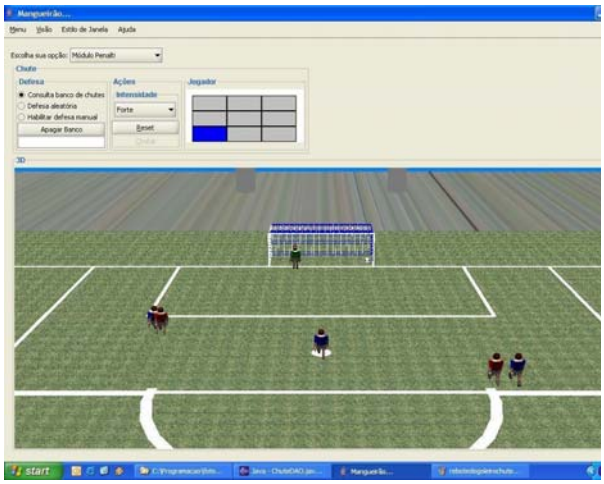


Figura 3.22 (f) - Rebote do goleiro com dominio de bola pelo atacante

O jogador atacante quando recebe a bola toca a mesma para o jogador que havia batido o penalte e este por sua vez chuta a bola no canto direito do gol.

As figuras (3.23 (a), 3.23 (b), 3.23 (c). 3.23 (d)) mostram a sequência de cenas para o rebote do goleiro com dominio de bola do jogador atacante e chute direto para o gol.

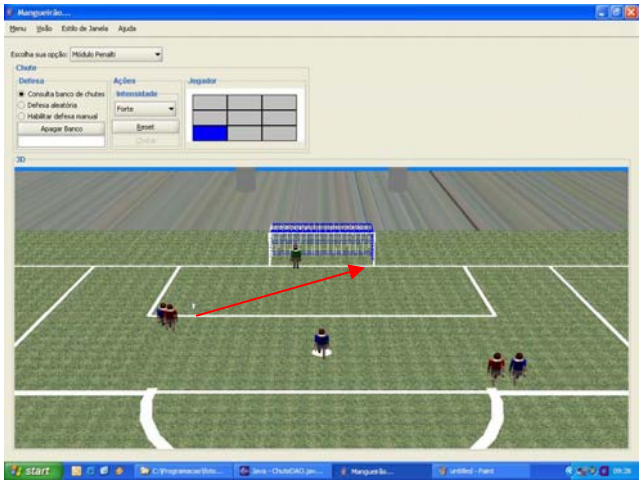


Figura 3.23 (a) - Rebote do goleiro com dominio de bola do atacante

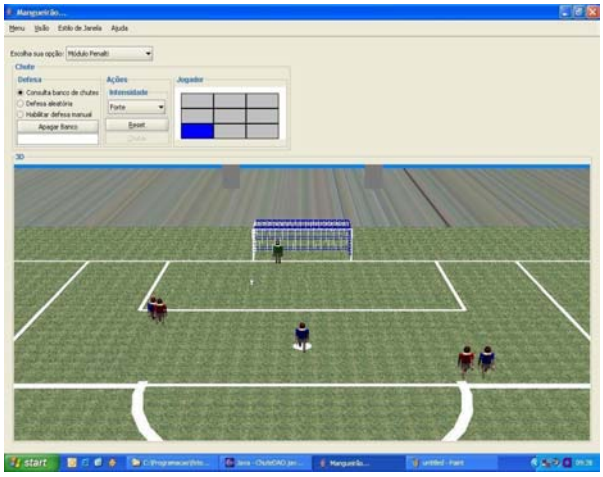


Figura 3.23 (b) - Rebote do goleiro com dominio de bola do atacante

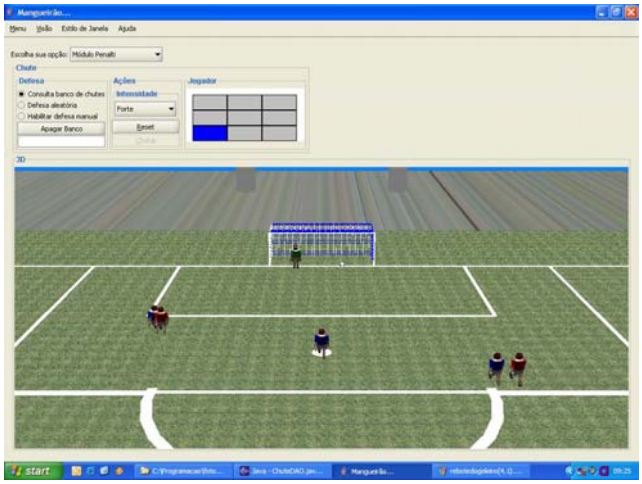


Figura 3.23 (c) - Rebote do goleiro com dominio de bola do atacante

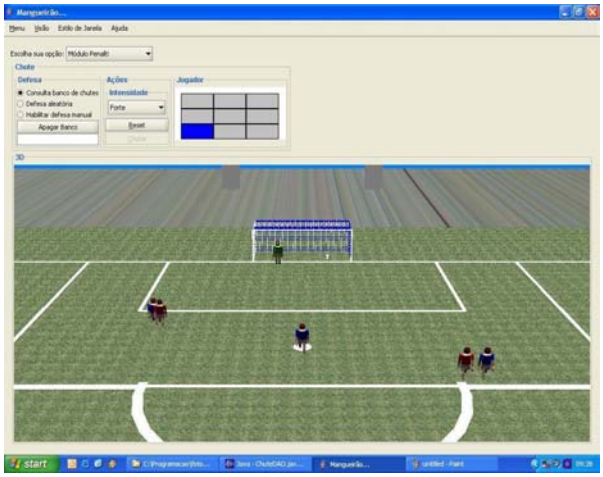


Figura 3.23 (d) - Rebote do goleiro com dominio de bola do atacante

As figuras (3.24 (a), 3.24 (b) e 3.24 (c)) mostram a sequência de cenas para o rebote do goleiro com dominio de bola pelo jogador de defesa

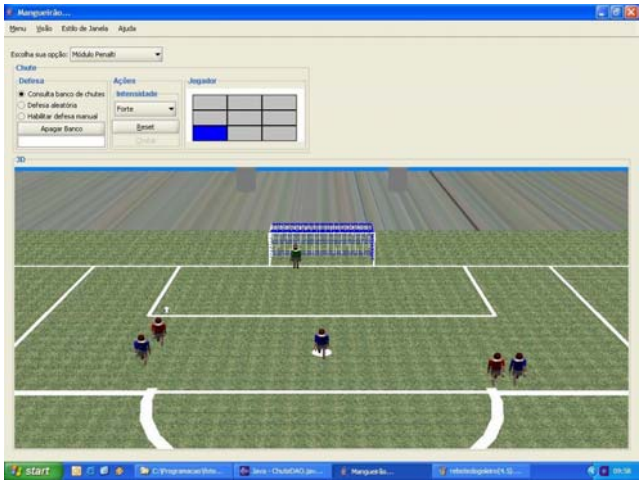


Figura 3.24 (a) - Rebote do goleiro com dominio de bola do defensor

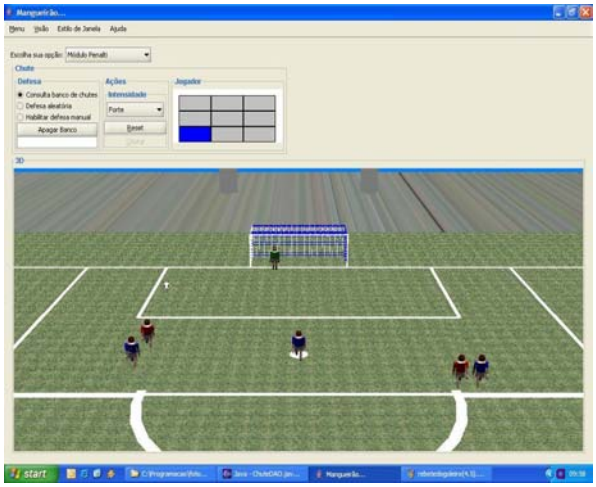


Figura 3.24 (b) - Rebote do goleiro com dominio de bola do defensor

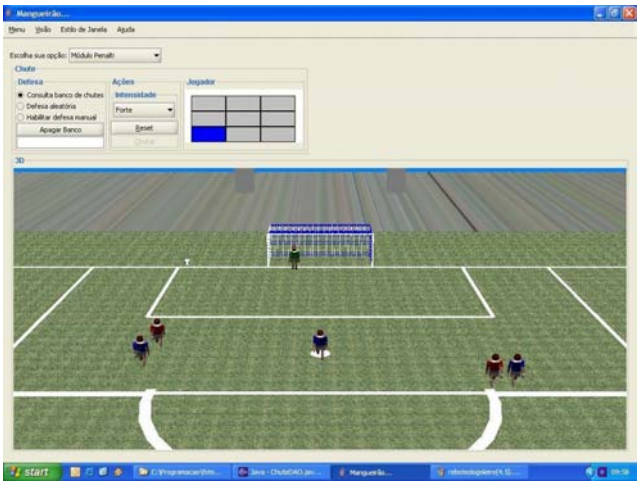


Figura 3.24 (c) - Rebote do goleiro com dominio de bola do defensor

3.6 – Módulo de Jogada Ensaiada

Neste módulo de jogada ensaiada a única interação que ocorre é a de iniciar ou parar a jogada. Assim, todas as jogadas ensaiadas são autônomas, onde os jogadores tomam suas próprias decisões. Todos os chutes neste módulo resultam em duas possibilidades:

1. Cruzamento de bola (rasteiro ou aéreo)
2. Chutar para o gol: canto esquerdo, no centro ou canto direito;
3. Chutar para fora do gol.

A figura 3.25 mostra a tela principal do módulo joigada ensaiada.

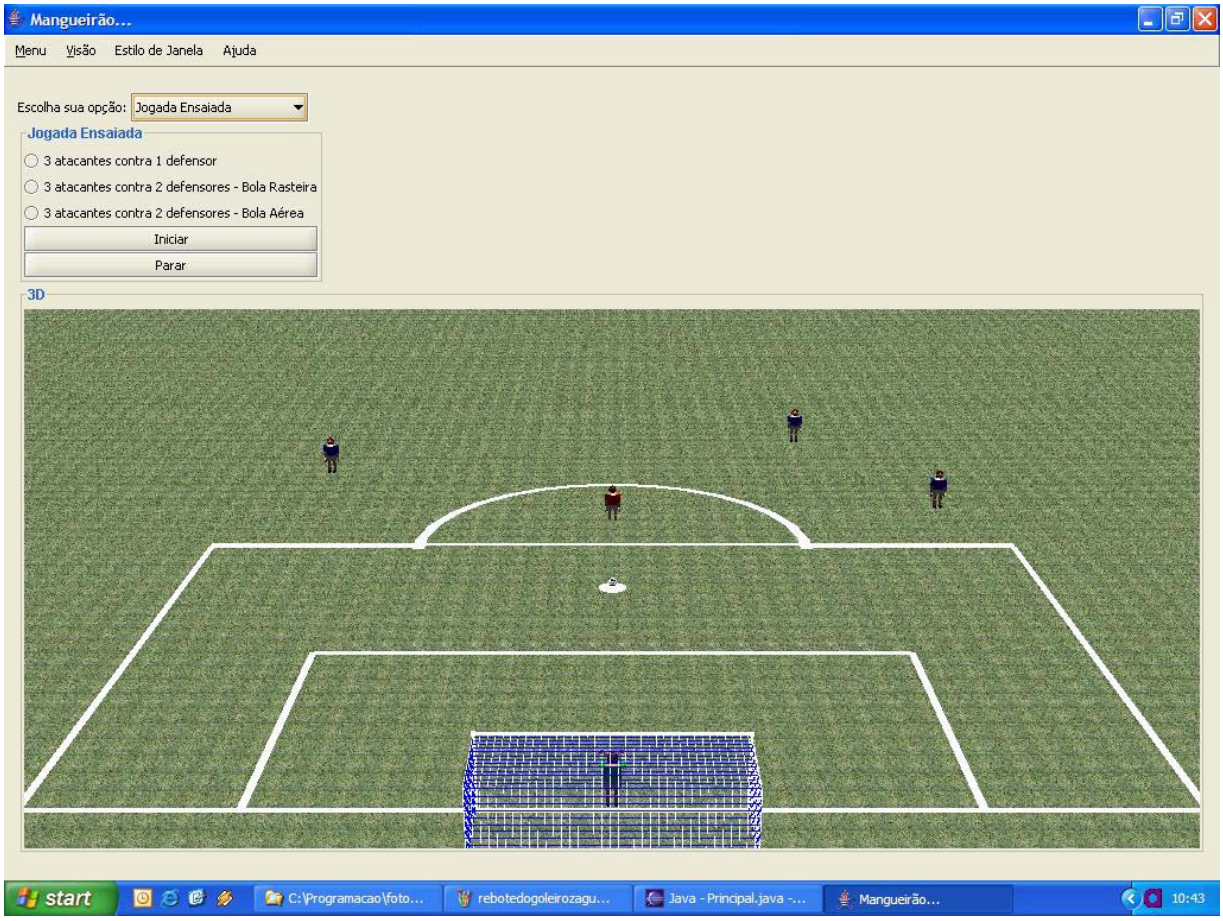


Figura 3.25. Tela principal do módulo jogada ensaiada

3.6.1 - Tipos de jogadas ensaiadas

3.6.2 - Três atacantes e dois defensores com cruzamento de bola (aéreo).

Neste módulo, o jogador da esquerda caminha com a bola e efetua o cruzamento para dentro da área. Assim, os jogadores que estiverem ao alcance primeiro da bola terão as seguintes ações:

- Se zagueiro cabecear, bola irá para fora do campo;
- Se atacante cabecear para o gol, existem quatro possibilidades: o atacante pode cabecear para o gol; para fora; cabecear na trave; ou o goleiro pode defender;

Estas sequencias podem ser observadas nas Figuras 3.26(a), (b), (c) e (d).

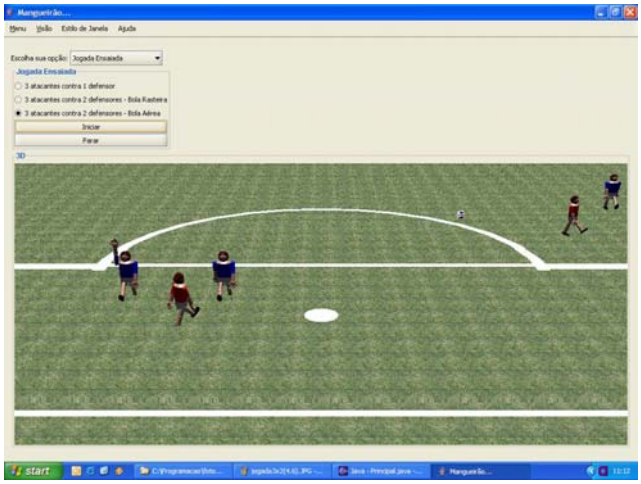


Figura 3.26 (a) – Cruzamento de bola aérea.

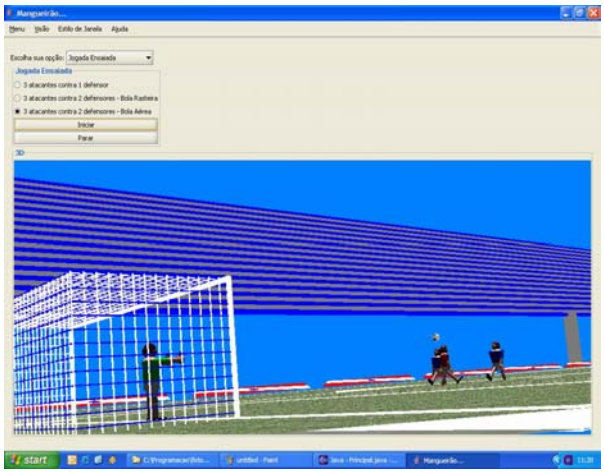


Figura 3.26 (b) - Cruzamento de bola aérea.

Figura 3.26 (c) - Cruzamento de bola aérea.

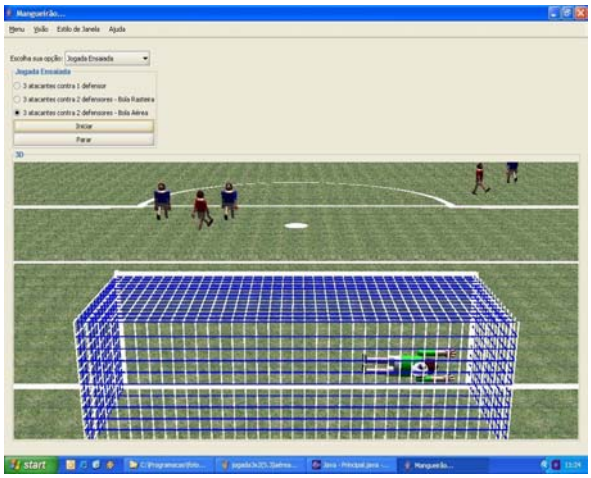


Figura 3.26 (d) - Cruzamento de bola aérea.

3.6.3 - Três atacantes e dois defensores com passe de bola (Rasteiro)

Nesta seqüência, o jogador com a posse de bola efetua o passe para o jogador que primeiro levantar a mão. Há duas possibilidades de jogada nesta seqüência:

- Se o passe for efetuado para o jogador da direita então o mesmo pode passar para o jogador da esquerda para o mesmo efetuar o chute (pro gol ou para fora);
- Se o passe for efetuado para o jogador da esquerda o mesmo chuta para o gol ou para fora;

Estas sequencias podem ser observadas nas Figuras 3.27(a), (b), (c), (d), (e), (f), (g), (h), (i) e (j).

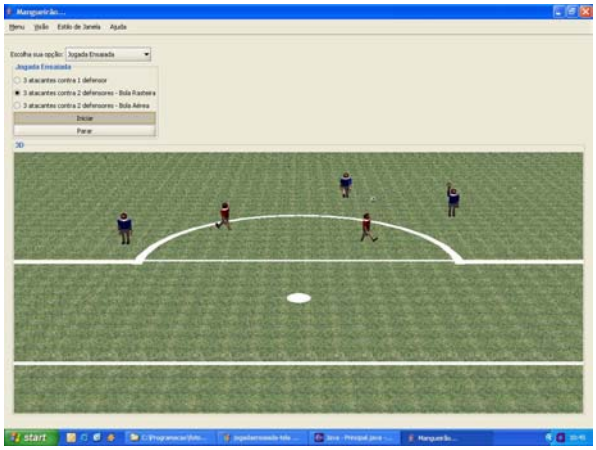


Figura 3.27(a) – Bola rasteira com três atacantes e dois defensores

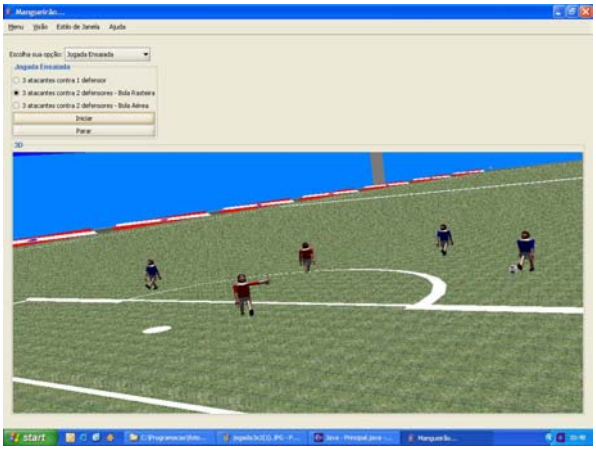


Figura 3.27(b) – Bola rasteira com três atacantes e dois defensores

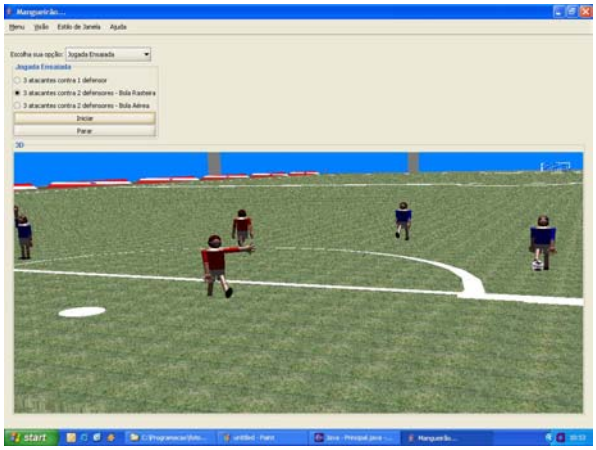


Figura 3.27(c) – Bola rasteira com três atacantes e dois defensores

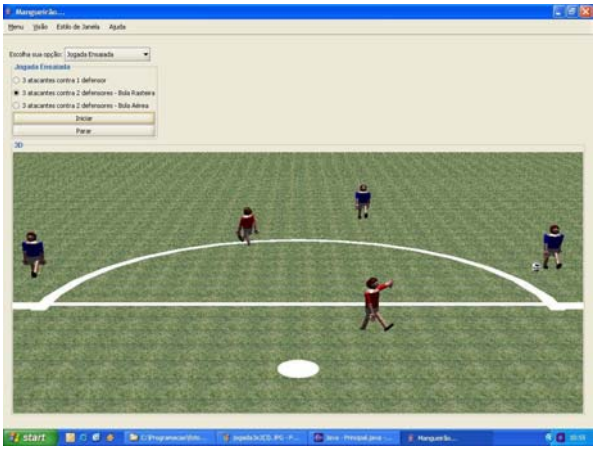


Figura 3.27(d) – Bola rasteira com três atacantes e dois defensores

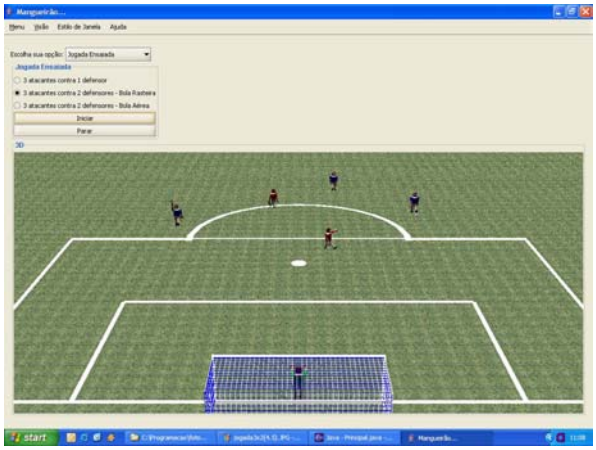


Figura 3.27(e) – Bola rasteira com três atacantes e dois defensores

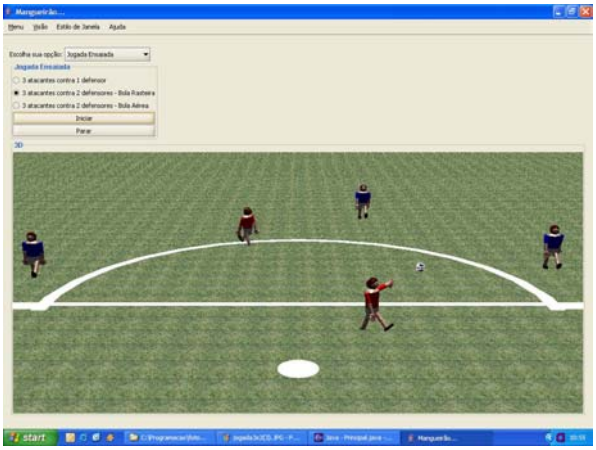


Figura 3.27(f) – Bola rasteira com três atacantes e dois defensores

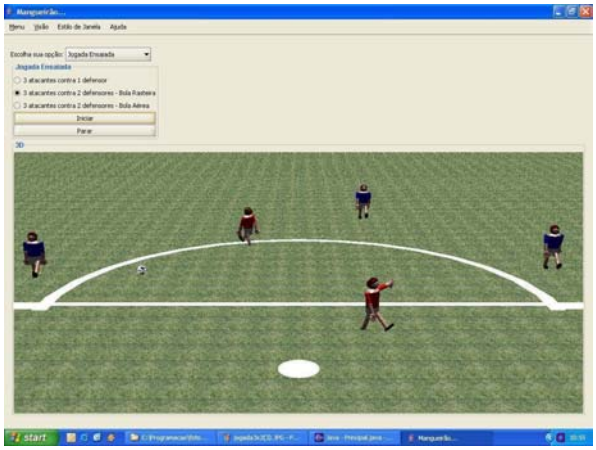


Figura 3.27(g) – Bola rasteira com três atacantes e dois defensores

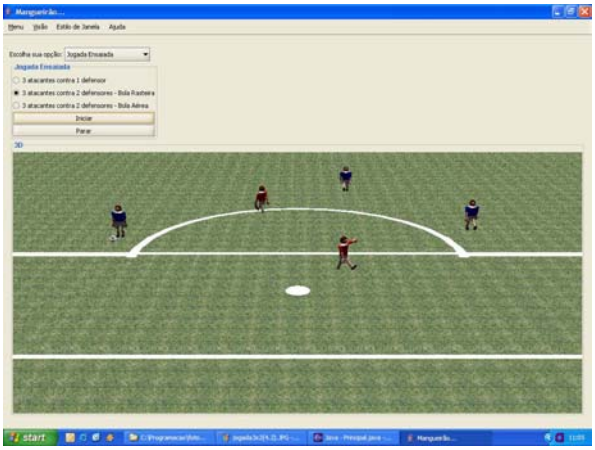


Figura 3.27(h) – Bola rasteira com três atacantes e dois defensores

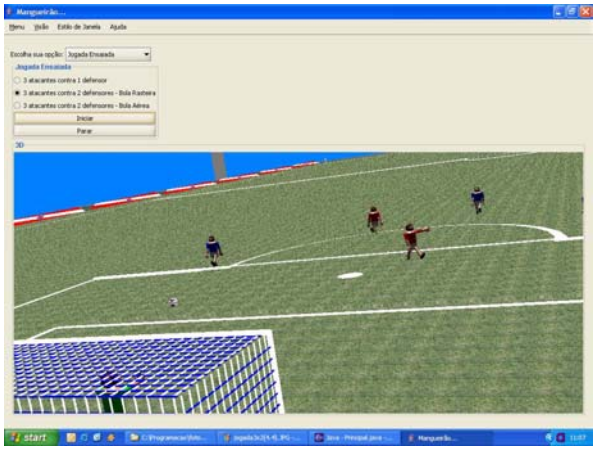


Figura 3.27(i) – Bola rasteira com três atacantes e dois defensores

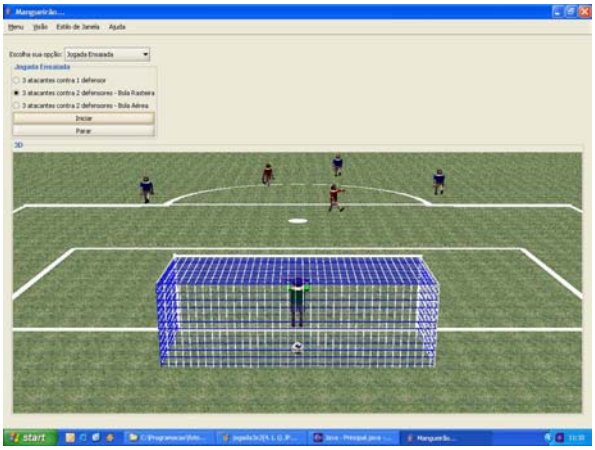


Figura 3.27(j) – Bola rasteira com três atacantes e dois defensores

3.6.3.1 - Três atacantes e um defensor com passe de bola

Nesta seqüência, o primeiro passe pode ser feito para o jogador da direita ou para o jogador da esquerda, caso o passe seja para o jogador da direita, o mesmo pode efetuar o chute (para o gol ou para fora), ou passar a bola para o jogador que está livre na esquerda, e o mesmo pode efetuar o chute (para o gol ou para fora). Se o passe for feito diretamente para o jogador da esquerda o mesmo só poderá efetuar o chute (para o gol ou para fora).

Estas sequencias podem ser observadas nas Figuras 3.28(a), (b), (c), (d), (e) e (f).

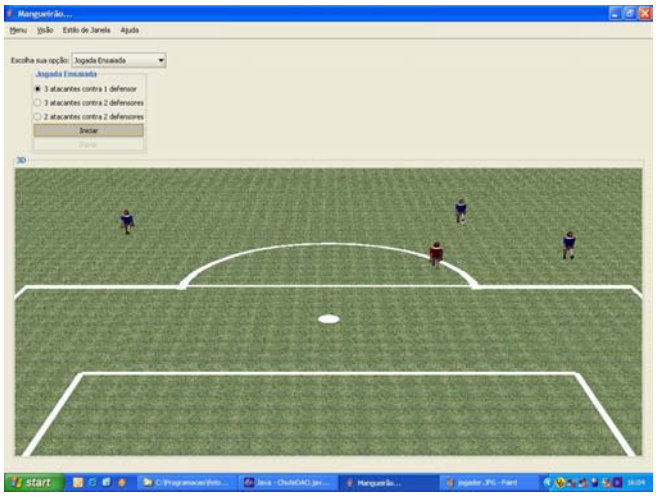


Figura 3.28(a) – Passe de Bola

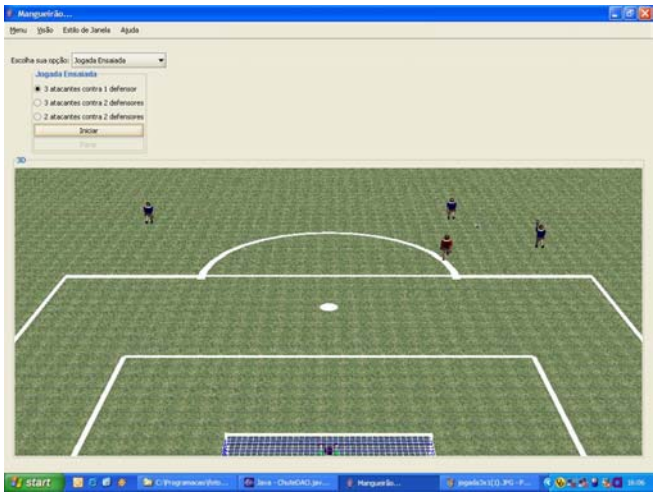


Figura 3.28(b) – Passe de Bola

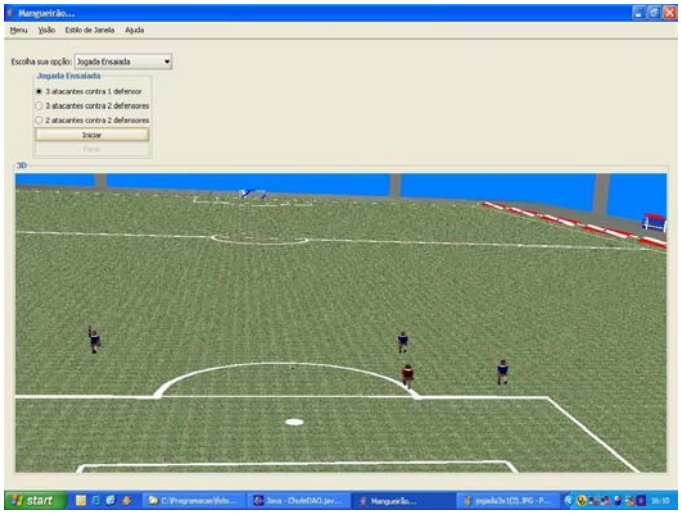


Figura 3.28(c) – Passe de Bola

Figura 3.28(d) – Passe de Bola

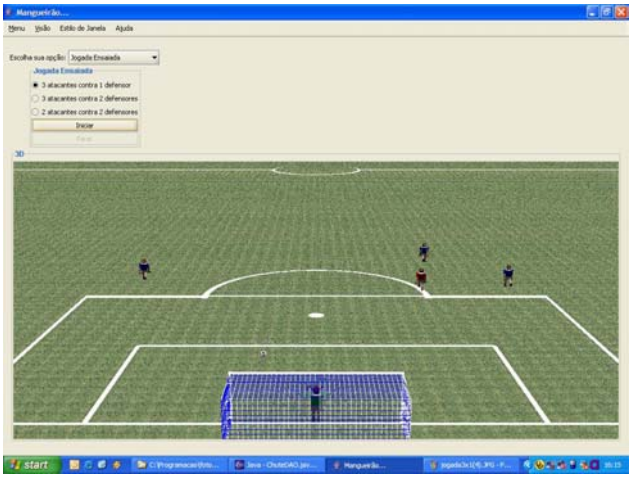


Figura 3.28(e) – Passe de Bola

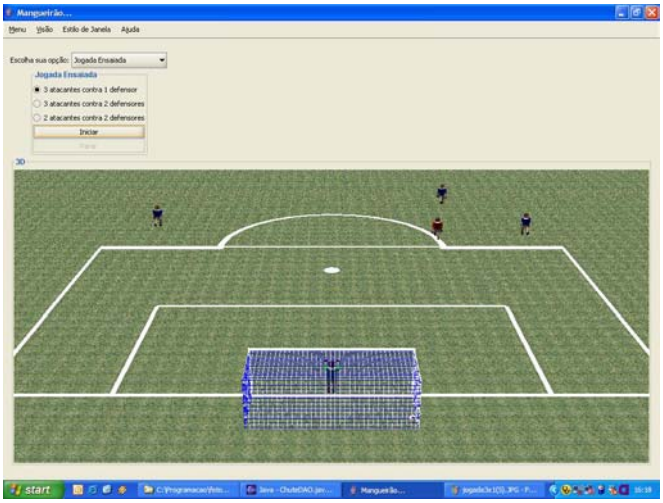


Figura 3.28(f) – Passe de Bola

3.7 - Modelamento do Sistema

Como intuito de representar todas as etapas de desenvolvimento do software *FUT-3D*, fez-se uso dos diagramas da UML (Linguagem de Modelagem Unificada) para uma melhor descrição e entendimento, como nas seções a seguir.

3.7.1 - Diagrama de Caso de Uso

O diagrama de caso de uso é utilizado para descrever a funcionalidade do *FUT- 3D* (LARMAN, 2000).

O FUT - 3D é uma aplicação que utiliza uma interface gráfica para interagir com os usuários. É por meio desta interface que um usuário pode escolher onde o jogador-atacante irá chutar a bola em direção ao gol. As atividades podem ser compostas com outras funcionalidades agregadas a atividade do goleiro. Uma vez escolhido o local do chute, é verificado na base de chutes se já tem algum chute armazenado; caso seja negativa a resposta do banco, é retornada ao goleiro a ação ficar parado; caso se tenha algum chute no banco de chutes é retornada ao goleiro uma ação baseada em um algoritmo que trata qual a melhor possibilidade de defesa. O diagrama de caso de uso pode ser visto na figura 3.29.

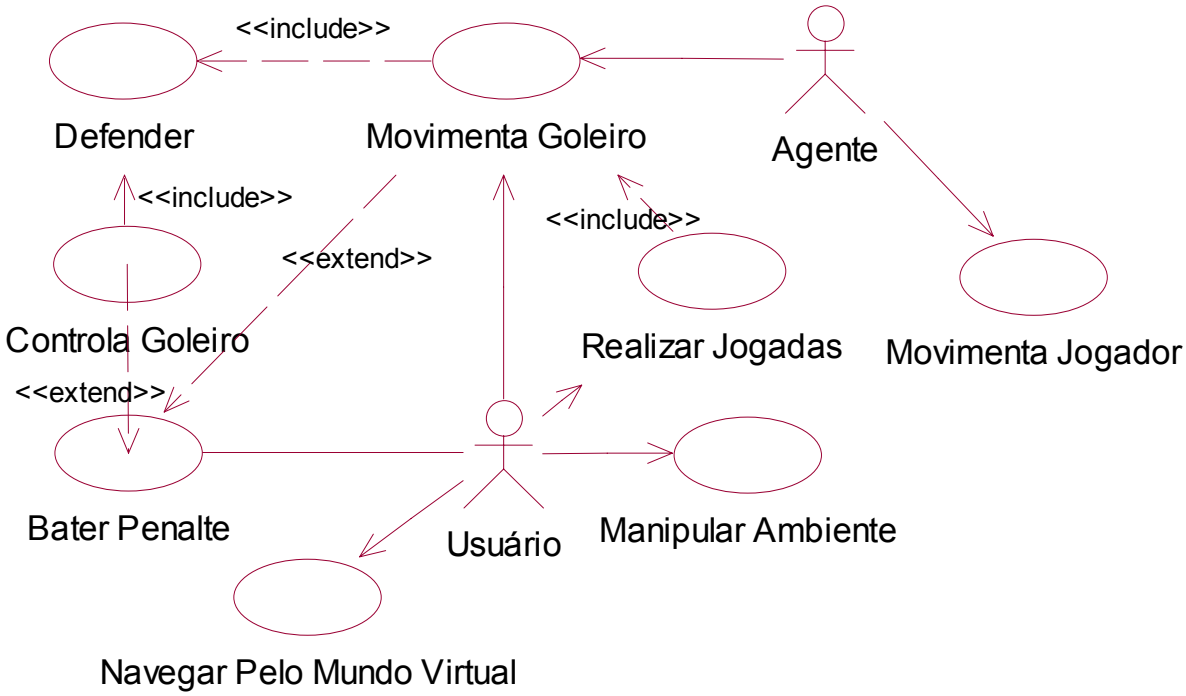


Figura 3.29: Diagrama de Caso de Uso para o protótipo.

3.7.2 - Diagrama de Classes

O diagrama de classes é uma das melhores formas de representar os componentes de dados e suas interações dentro do protótipo (LARMAN, 2000). O *FUT 3D* foi concebido pro meio de uma abordagem orientada a objetos, e toda a construção do diagrama já foi voltada para uma visão bem detalhada do protótipo, como pode ser visto na Figura 3.30.
A figura abaixo representa o diagrama de classes do projeto resumido, onde são demonstrados os pacotes, as classes e a multiplicidade entre as classes, juntamente com a navegabilidade entre elas.

Os pacotes foram criados de forma a agrupar as classes de acordo com as suas funcionalidades. A seguir uma descrição dos pacotes:

- Banco: Representa a classe de conexão com o banco de dados da aplicação. Como a aplicação somente armazenará o chute do jogador, então a classe de conexão foi chamado de chuteDAO.
- Bean: Representa o pacote que contém as classes que representam os beans da aplicação.
- Modelos: Pacote criado armazenar as classes que terão as funcionalidades de interação do usuário com a aplicação, seja através do jogador ou somente alterando a visão do campo.
- Gui: Agrupam todas as classes que compõem toda a interface gráfica da aplicação. Estas classes servem para que o usuário faça a interação do usuário e da aplicação.
- Util: Este pacote contém as classes de funcionalidades genéricas da aplicação e contém ainda a chamada para o agente de inteligência artificial da aplicação.
- Visao3d: Este pacote armazenarão as classes que servirão para instanciar os elementos 3D da aplicação.
- Colisão: Representa as classes que terão a funcionalidade de demonstrar a colisão entre a bola e os jogadores e o goleiro da aplicação.

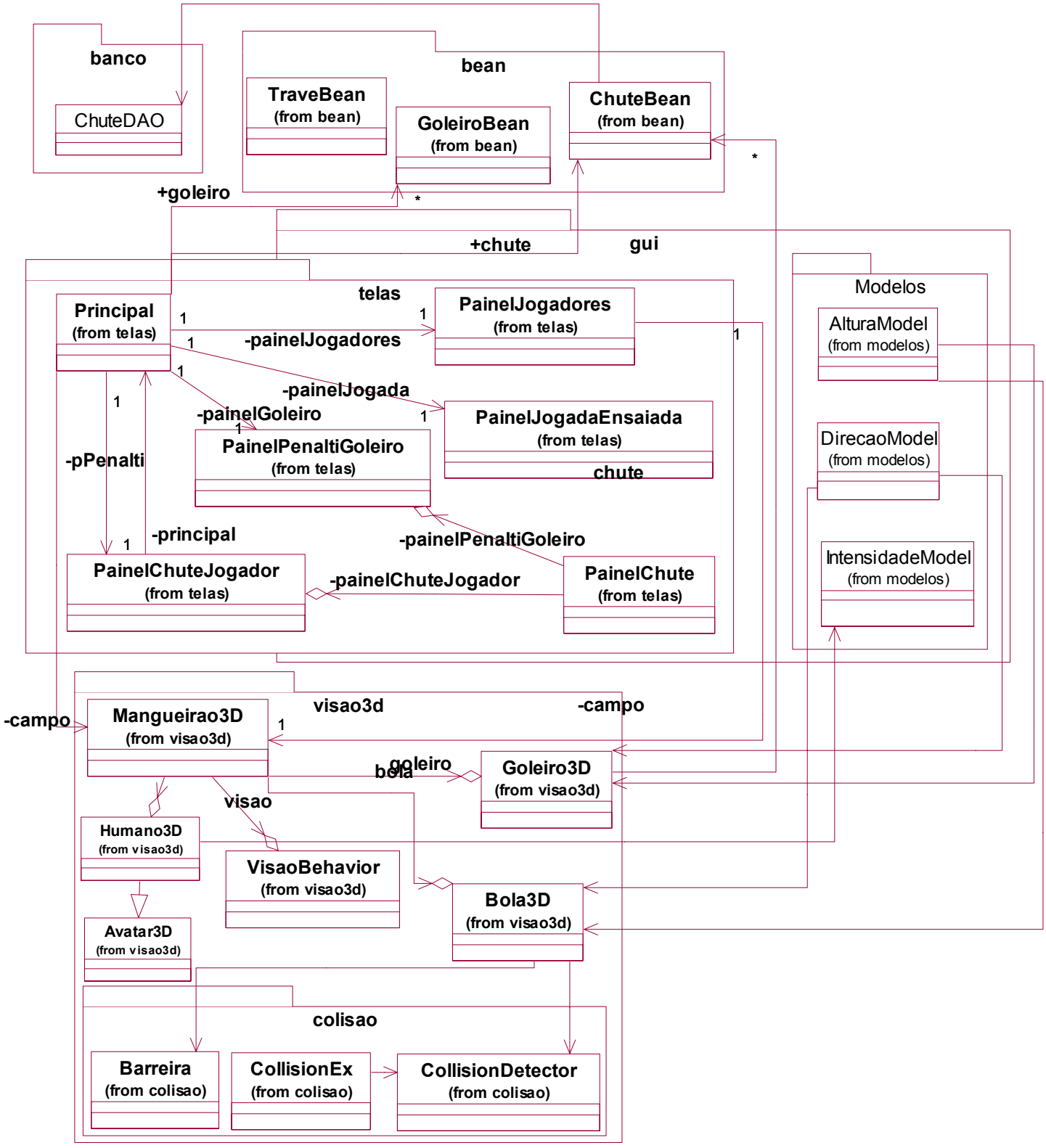


Figura 3.30: Diagrama de Classes para o protótipo.

3.7.3 - Diagrama de Componentes.

O diagrama de componentes mostra a organização entre arquivos de código fonte, bibliotecas, tabelas de banco de dados etc. A relação mais usada é a dependência, mostrando como um arquivo de código fonte depende de um outro que ele inclui, ou como um executável depende de uma biblioteca. Um componente é a parte física do sistema. Muitas vezes um componente mostra um arquivo específico do sistema.

A UML reconhece cinco estereótipos de componentes, conforme a figura 3.31:

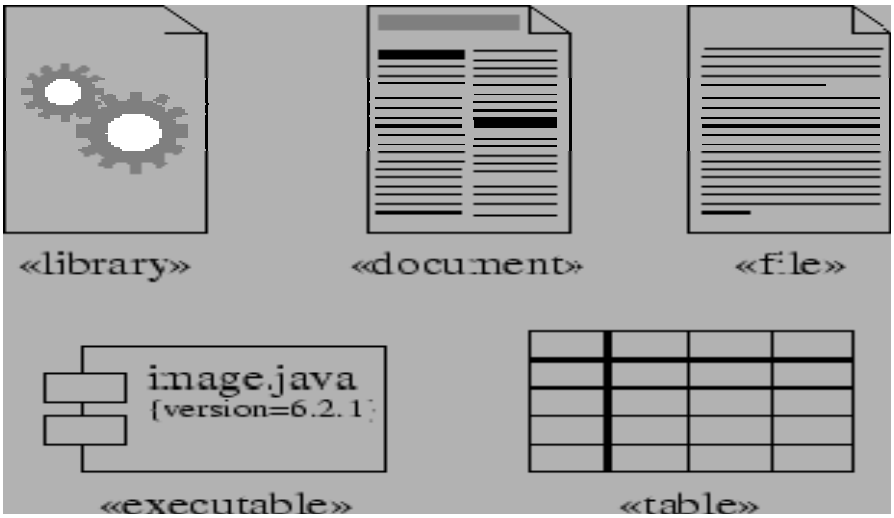


Figura 3.31: Esteriótipos de Componentes reconhecidos pela UML.

A figura 3.32 mostra a modelagem do diagrama de componentes do FUT – 3D, ressaltando os relacionamentos existentes no sistema, tanto entre as classes, como entre os pacotes.

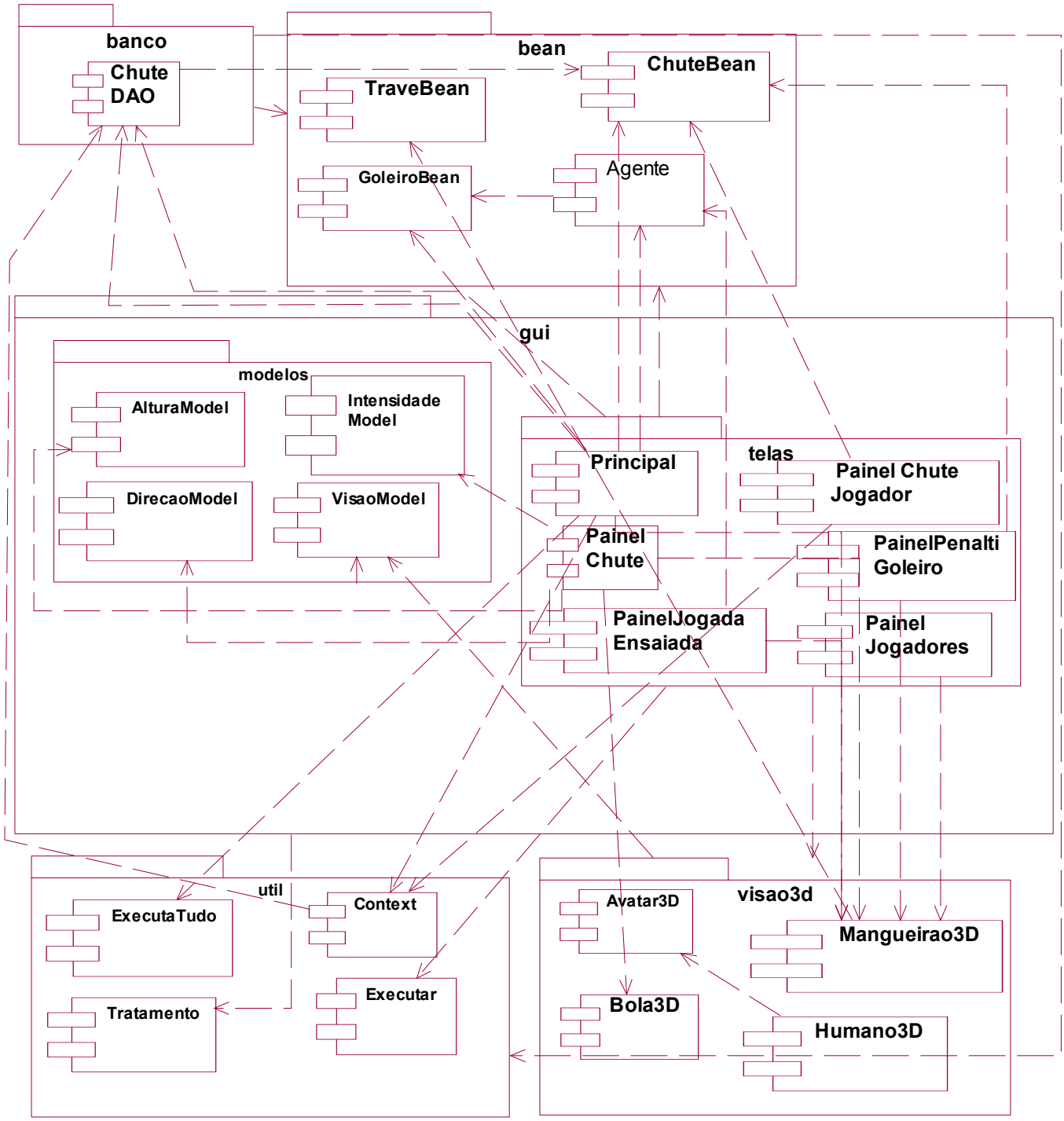


Figura 3.32 Diagrama de Componentes para o protótipo

3.7.4 - Diagrama de Seqüência.

Um diagrama de seqüência mostra a colaboração dinâmica entre os vários objetos de um sistema. O mais importante aspecto deste diagrama é que, a partir dele, percebe-se a seqüência de mensagens enviadas entre os objetos. Ele mostra a interação entre os objetos, alguma coisa que acontecerá em um ponto específico da execução do sistema. O diagrama de seqüência consiste em um número de objetos mostrado em linhas verticais. O decorrer do tempo é visualizado observando-se o diagrama no sentido vertical de cima para baixo. As mensagens enviadas por cada objeto são simbolizadas por setas entre os objetos que se relacionam.

Diagramas de seqüência possuem dois eixos: o eixo vertical, que mostra o tempo; e o eixo horizontal, que mostra os objetos envolvidos na seqüência de certa atividade. Eles também mostram as interações para um cenário específico de certa atividade do sistema.

A interface gráfica do penalte disponibiliza os atributos de altura, direção e força para o chute do jogador e para o goleiro a opção de defesa manual ou automática.

Quando os usuários escolhem suas opções e apertam o botão chutar disponível na interface o método é então disparado. No primeiro momento é instanciado um objeto ChuteBean utilizando-se como parâmetros a altura, direção e força, e este objeto é então posto na seção da aplicação. Em seguida, o procedimento é repetido para o GoleiroBean, quando no interface de penalte é assinalado a opção de defesa manual. Logo depois, o ChuteBean é armazenado no banco de dados, como forma de se ter dados históricos do chutes, que serão utilizados pelo agente de inteligência artificial, quando for escolhido a opção de defesa automática. Em seguida é chamado o método andar, que representa o deslocamento do jogador até a bola, que encontra-se na marca do penalte. Em seguida, o jogador executa o método chute, utilizando-se os atributos do chute que foi colocado anteriormente na sessão da aplicação. Após o chute, o goleiro executa o método agarrar e que utiliza o goleiro posto na sessão. Como o diagrama de seqüência acima não utiliza as classes de colisão, devido o penalte realizado foi gol.

A figura 3.33 mostra o diagrama de seqüência do FUT - 3D.

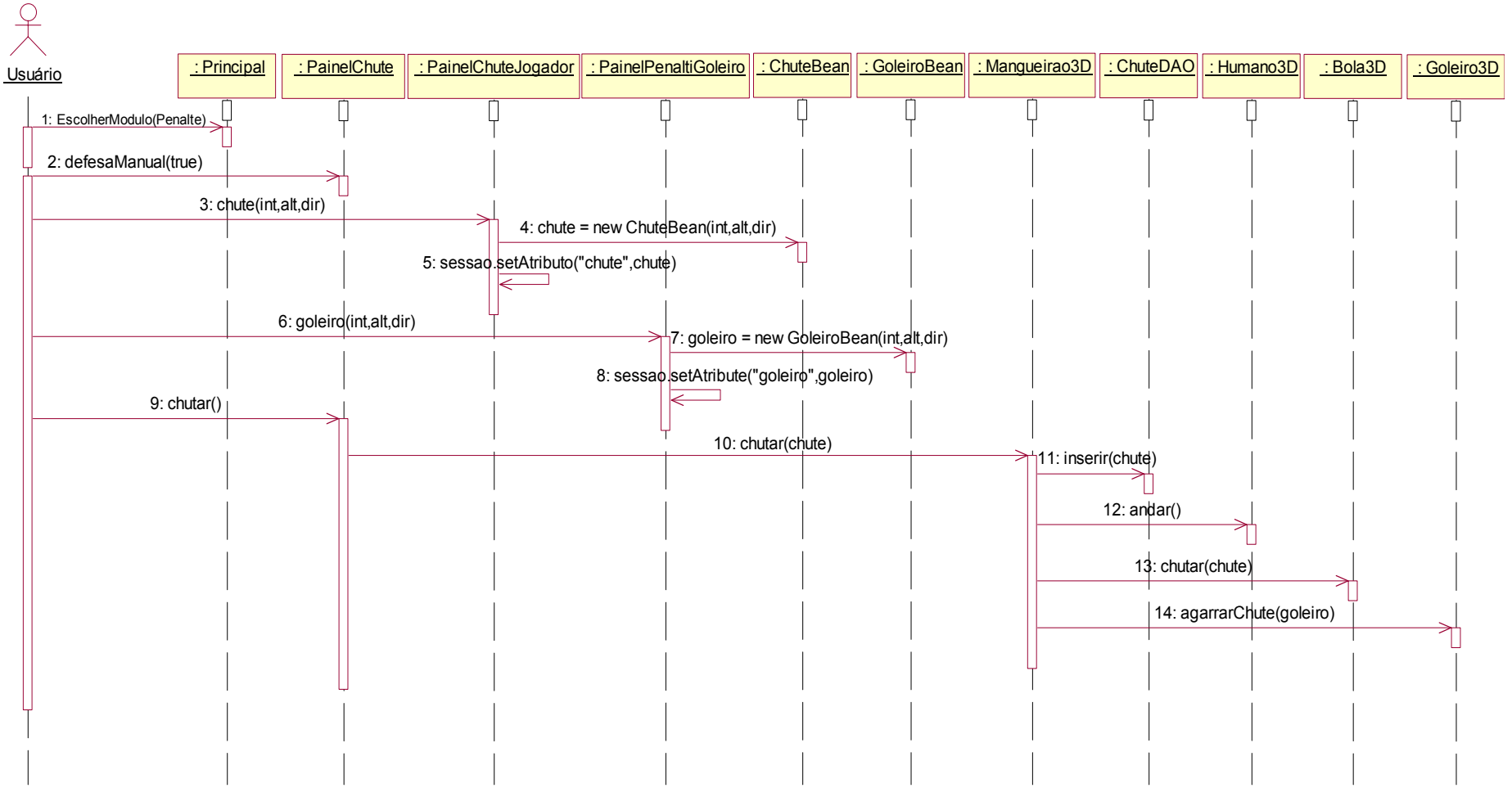


Figura 3.33: Diagrama de seqüência para o protótipo

3.7.5 Diagrama de Atividades.

O Diagrama de atividade é definido pela (UML), representa os fluxos conduzidos por processamentos. Um diagrama de atividade é essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Na maior parte, isso envolve a modelagem das etapas seqüências em um processo computacional.

Os diagramas de atividade não são importantes somente para a modelagem de aspectos dinâmicos de um sistema, mas também para a construção de sistemas executáveis por meio de engenharia de produção reversa.

Os diagramas de atividade costumam conter o seguinte:

- Estado de atividade e estado de ação.
- Transições
- Objetos

A figuras 3.34 e 3.35 mostram os diagramas de atividades para jogador e goleiro respectivamente do FUT - 3D.

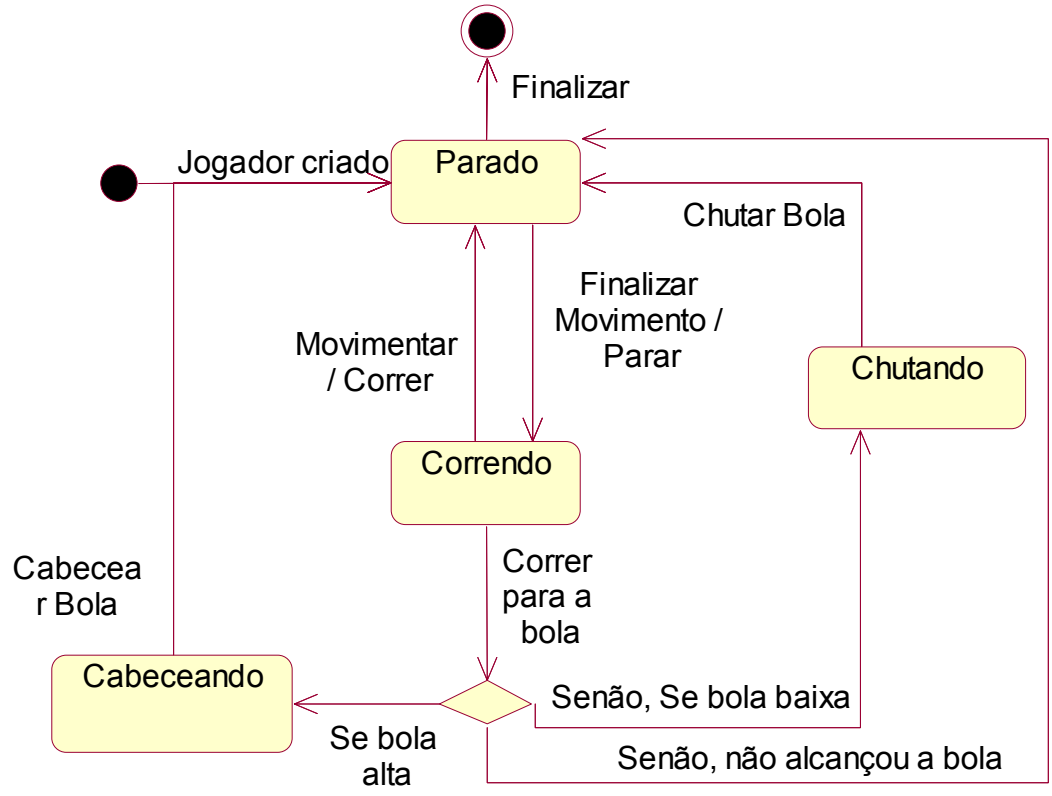


Figura 3.34: Diagrama de atividades para o jogador

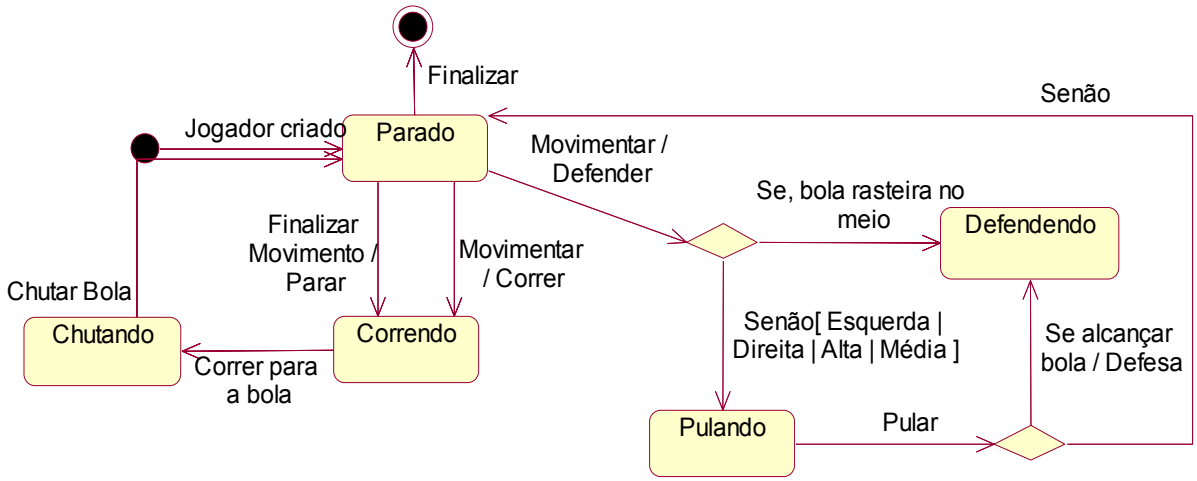


Figura 3.35: Diagrama de atividades para o goleiro

3.8 - Arquitetura

Para tornar o desenvolvimento mais simplificado, foi criada uma arquitetura para descrever de forma macro como funciona o protótipo. A figura 3.36 mostra como o Mundo Virtual interage com a Interface Gráfica, cuja responsabilidade é fornecer uma forma fácil de interação entre o Usuário e a aplicação. A Interface Gráfica se relaciona também com o módulo de manipulação de dados, responsável em buscar as informações que estão armazenadas na base de dados.

A Base de Dados armazena todos os eventos (chutes) desferidos contra o gol, tendo eles sucesso ou não. O módulo Inteligência, que está agregado ao módulo Manipulação de Dados, é responsável por gerar a condição de autonomia tanto do goleiro quanto do jogador, no qual, através do modulo Regras, analisa a melhor condição para efetuar uma jogada. Tanto o jogador quanto a bola se relacionam através do Mundo Virtual, pois é neste módulo que se apresenta a situação de colisão entre a bola e as demais entidades do mundo virtual, como por exemplo, o goleiro. A comunicação entre o jogador e a bola é intermediada pelo mundo virtual, pois, o jogador sempre espera uma reação da bola após algum tipo de interação entre a bola e o mundo, como o rebote do goleiro.

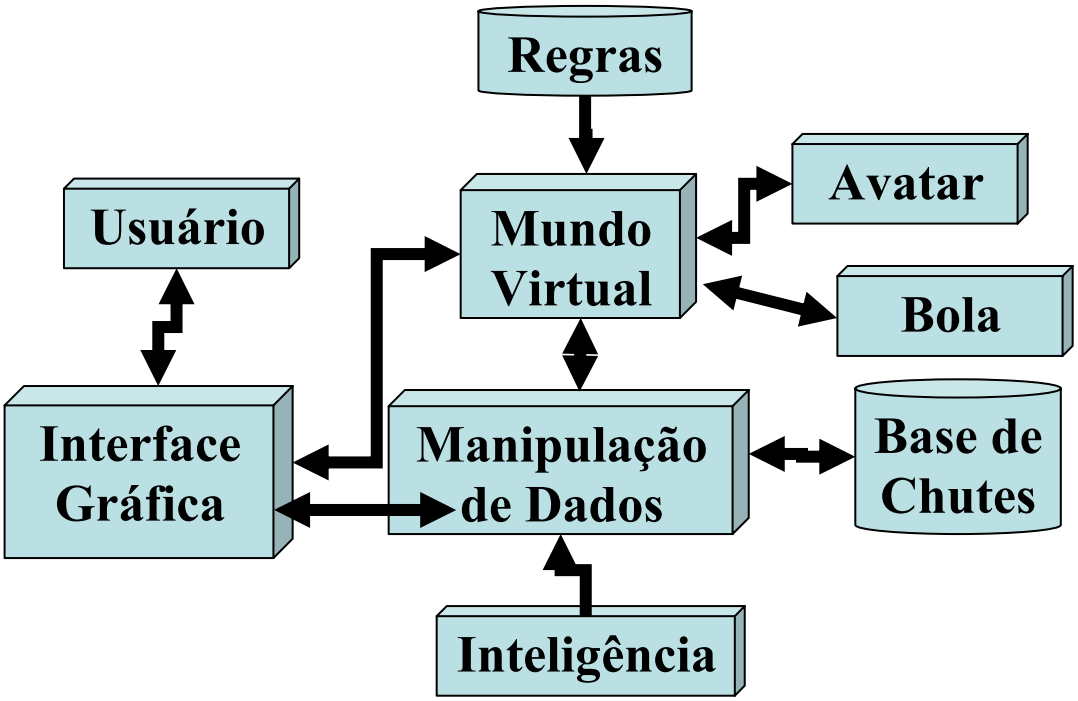


Figura 3.36: Arquitetura para o protótipo

3.9 - Inteligência e Autonomia

A implementação dos fatores de inteligência e autonomia se fez a partir de regras previamente estipuladas e através de computação evolucionária utilizando-se de algoritmos genéticos para uma melhor resolução dos problemas.

As regras utilizadas são, na realidade, situações que podem ocorrer durante uma partida de futebol e devem ser analisadas pelo jogador ou goleiro na hora em que elas ocorrem.

A figura 3.37 mostra o resultado final, de um pseudocódigo de regras utilizadas no FUT 3D.

```
1  se (goleiroAgarrarChute) então
2          goleiro.rebateBola();
3          tempoJogadorAtaque = randômico(0,1);
4          tempoJogadorDefesa = randômico(0,1);
5
6          se (tempoJogadorDefesa> tempoJogadorAtaque)
7                  jogadorDefesa.correrAteABola();
8                  jogadorDefesa.chutarParaFora();
9          fim se
10         senao inicio
11                 int numero = randômico(0,3);
12                 se (numero=0)
13                         jogadorAtaque.chutaNoMeio();
14                 se (numero=1)
15                         jogadorAtaque.chutaNaDireita();
16                 se (numero=2)
17                         jogadorAtaque.chutaNoEsquerda();
18                 se (numero=3)
19                         jogadorAtaque.passarParaJogadorDoCentro();
20                         jogadorDoCentro.escolherCantoChute();
21                         jogadorDoCentro.chutar();
22         fim senao
23 fim
```

Figura 3.37 – Pseudocódigo da Regras para o módulo pênalti.

A figura 3.37 mostra uma parte das regras utilizadas no Fut – 3D, especificamente a da figura acima trata da condição em que o goleiro, após desferido um chute, defende a bola rebatendo-a, o que gera um tempo randômico para que se possa saber qual dos jogadores (atacante ou defensor) chegará primeiro à bola. Caso o tempo do defensor seja maior que o tempo do atacante, este correrá até a bola e a chutará para fora. Caso contrário, ou seja, o tempo do atacante seja maior que o do defensor, será gerado um novo número randômico de 0 (zero) à 3 (três), para determinar qual ação será tomada pelo atacante, ações estas que podem ser:

1. Chutar a bola no meio do gol;
2. Chutar a bola no lado direito do gol;
3. Chutar a bola no lado esquerdo do gol;
4. Atacante tocar a bola para outro companheiro de time;

A figura 3.38 mostra a representação visual da implementação do pseudocódigo em java 3d.

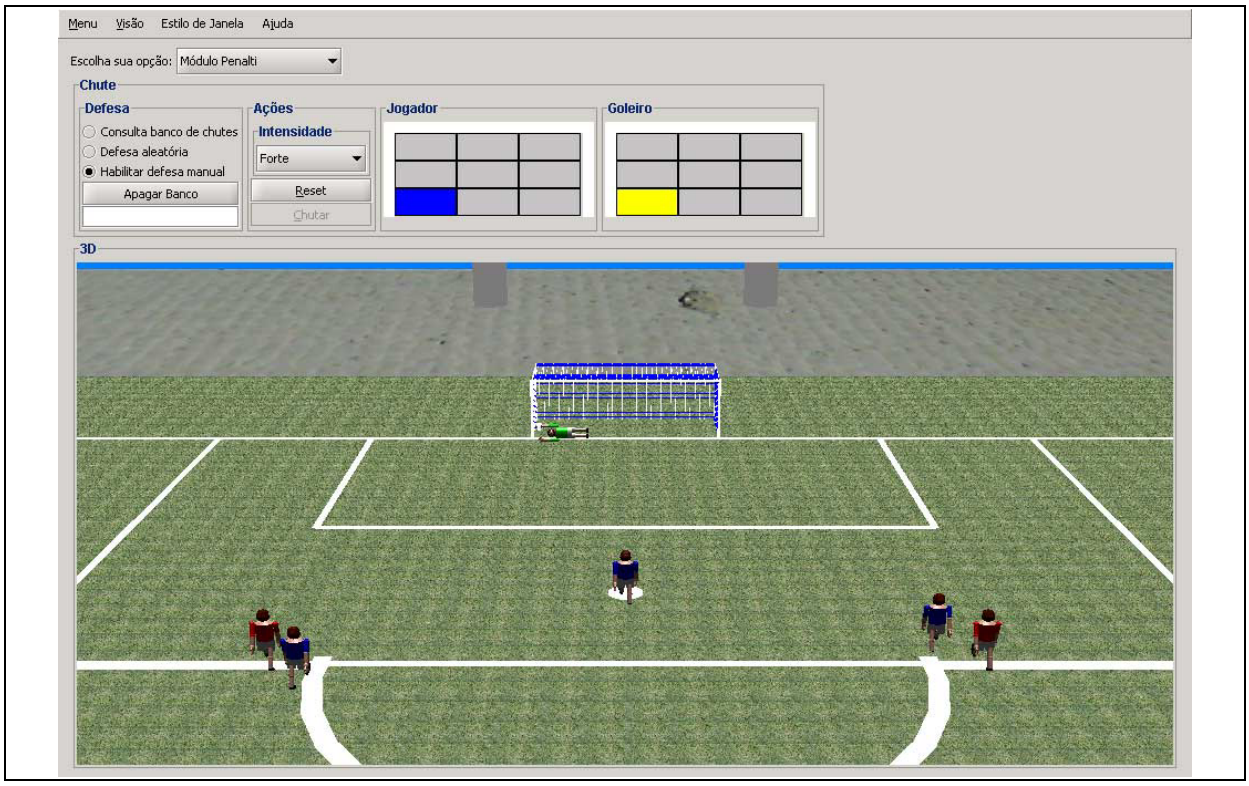


Figura 3.38 – Visualização da Implementação do Pseudocódigo no FUT 3D.

A computação evolucionária através dos algoritmos genéticos (AG) vem a facilitar o trabalho de obter o melhor resultado dentro de um conjunto de possibilidades apresentadas. Para isto, se utilizou o método roleta que já foi previamente explicado no capítulo anterior, e que agora será mostrado sua implementação.

3.10 - Método Roleta

Esta técnica reúne todos os chutes do banco, logo após é gerado uma roleta e depois faz-se um sorteio, por exemplo:

DIREÇÃO	ALTURA	INTENCIDADE
ESQUERDA (E)	ALTO (A)	FORTE (F)
DIREITA (D)	MÉIO (M)	MÉDIO (Me)
CENTRO (C)	BAIXO (B)	FRACO (Fr)

Tabela 3 – Possibilidades de composição do chute.

A figura 3.39 apresenta os armazenados no banco de dados.

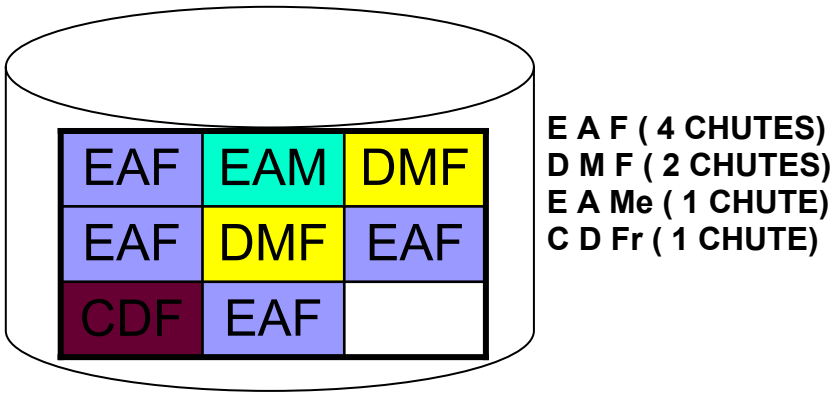


Figura 3.39 – Método Roleta.

A partir deste entendimento, a roleta então se divide da seguinte forma:

1. Pega-se a soma de todos os chutes, que no caso dá 8;
2. Verifica-se a proporção do número de chutes por esse valor, no caso 4 (E A F) corresponde a 50%, e assim por diante;
3. A roleta gira de 0 a 100, o que quer dizer é que vai ser gerado um número aleatório. Para o número que cair é verificado a que parte na roleta corresponde ao chute que caiu, dessa forma, o chute EAF tem 50% de chance de ser o escolhido.

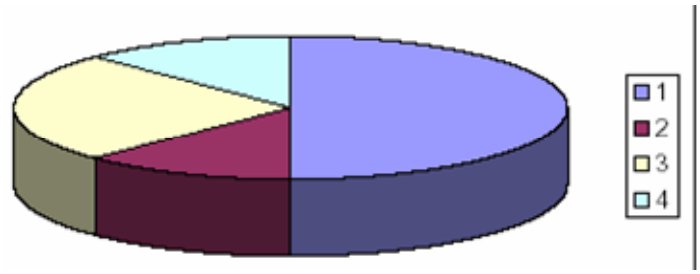


Figura 3.40 –Gráfico Método Roleta.

A figura 3.41 mostra o código utilizado para a implementação do método roleta no FUT – 3D.

```
161 Este método implementa o Método da Roleta que é uma técnica baseada
162 em computação evolucionária
163
164 @return
165
166 lic ChuteBean chutePossivel(){
167  ArrayList chutes = this.chutes;
168  if(chutes.size()==0){
169      return getChuteAleatorio();
170  }
171  int[] chute = new int[27];
172  ChuteBean[] c = new ChuteBean[27];
173  for (int i = 0; i < chutes.size(); i++) {
174      ChuteBean aux = (ChuteBean) chutes.get(i);
175      if(aux.getDirecao().equals(ChuteBean.DIRECAO_DIREITA)){
176          if(aux.getIntensidade().equals(ChuteBean.INTENSIDADE_FORTE)){
177              if(aux.getAltura().equals(ChuteBean.ALTURA_ALTO)){
178                  chute[0]++;
179                  c[0] = aux;
180              }
181              if(aux.getAltura().equals(ChuteBean.ALTURA_MEDIO)){
182                  chute[1]++;
183                  c[1] = aux;
184              }
185              if(aux.getAltura().equals(ChuteBean.ALTURA_RASTEIRO)){
186                  chute[2]++;
187                  c[2] = aux;
188              }
```

Figura 3.41 – Codificação do Método Roleta.

Capitulo 4

4 -CONSIDERAÇÕES FINAIS E TRABALHOS FUTUROS

4.1 - Considerações Finais

O trabalho tem por objetivo mais amplo expor a potencialidade da Realidade Virtual como uma tecnologia de auxílio à treinamentos e jogos. Em termos mais específicos, o trabalho teve como objetivo o projeto, implementação e utilização de um protótipo denominado *FUT 3D*, baseado em realidade virtual, técnicas de modelagem para construção de humanos artificiais, utilização de regras para realização de ações pertinentes ao jogo e um método de seleção baseado em algoritmos genéticos chamada de Método Roleta, que potencializa o acerto das ações dos humanóides dentro do contexto do jogo.

Conseqüentemente, outros objetivos periféricos foram atingidos, tais como: i) modelamento do sistema com técnicas de UML através de engenharia reversa ii) modelamento da arquitetura H-anim 1.1 com Java 3D iii) Mostrar que regras quando bem testadas podem ser uma boa forma de treinamentos seja estas quaisquer que forem.

O*FUT 3D* foi desenvolvido com ferramentas gratuitas, apesar do custo está totalmente direcionado, aos recursos humanos para desenvolvimentos, ferramentas de autoria deveriam ser consideradas para facilitar a construção de novas aplicações ou melhoramento das já existentes.

Alguns dos grandes problemas encontrados para o desenvolvimento do protótipo foram as técnicas de animações, que não atingiu o seu ponto ótimo, a construção de forma física dos humanóides, em função da complexidade inerente de se trabalhar com forma humanas, mas que foram contornadas de forma satisfatória.

4.2 Trabalhos Futuros.

Algumas características úteis ainda poderiam ser implementadas ao FUT 3D, tais como:

- Replay para analise de jogadas e implementação de juizes virtuais.
- Construção de humanóides mais bem acabados com técnicas revestimento de pele.
- Testes de usabilidade são bem-vindos para aprimorar a interface do protótipo e melhorar a interação e a percepção do usuário quando do uso da interface.
- Desenvolver uma arquitetura que funcione da forma distribuída.

5 - REFERÊNCIAS BIBLIOGRÁFICAS

AKAGUI, D., Kirner, C. “LIRA (2004) Livro Interativo com Realidade Aumentada”, Proc. of VII Symposiumon Virtual Reality, SP, outubro de 2004.

AMORIM, M.G.; PÍCCOLO, H.L. Kiko. Sistema de Animação para Representação de Movimento Humano. In: SIBGRAPI, 5., 1992, Águas de Lindóia, S.P. **Anais…** [s.l.:s.n.],1992. p.243-250.

BABSKI, C.; THALMANN, D. Real-Time Animation And Motion CaptureIn Web Human Director (WHD). 1999

BADLER, N.; Becket, W.; Granieri, J.. Towards real-time simulated human agents. em: WORKSHOP ON SIMULATION AND INTERACTION IN VIRTUAL ENVIRONMENTS, I, 1995. Proceedings. Iowa:University of Iowa, 1995. p. 126-129.

BITTENCOURT, Guilherme (1998). Inteligência Artificial : Ferramentas e Teorias. Florianópolis: DAUFSC

ÇAPIM, T. K.; Pandzic, I. S.; Magnenat-Thalmann, N.; Thalmann, D. Avatares in Networked Virtual Environments. John Wiley & Son, LTD. New York, 1999.

DARKEN, R. P. and Sibert, J. L.- Wayfinding strategies and behaviors in large virtual environments. In Human Factors in Computing Systems, CHI '96 Conference Proceedings, pages 142-149.

DEITEL, H. M.; DEITEL, P.J. Java - Como Programar, Bookman, 3a. Edição, 2001.

ECKEL,B. – Thinking in Java (3ª ed), Prentice hall, 2002.

FERNANDES, Anita Maria da Rocha. Inteligência Artificial – noções gerais. Florianópolis: Editora Visual Books, 2003.

FERREIRA, A. G. Uma Arquitetura para a Visualização Distribuída de Ambientes Virtuais.Dissertação de Mestrado, PUC/RJ, 1999.

FIALHO, Francisco A. P. Sistemas de Educação a Distância. Material didático apresentado na disciplina Conhecimento, Informação e Educação I, PPGEP - UFSC. 1998.

ISDALE J., Daly L., Fencott C., & Heim M. , "Content Development for Virtual Environments", in Stanney, K. M., (ed), "The Handbook of Virtual Environments", Lawrence Erlbaum Associates, 1998.

KIRNER,C, Silva,R.R.P.C.L.- Autoria Colaborativa de Mundos Virtuais Educacionais com Realidade Misturada, Anais do 1º Workshop de Realidade Aumentada , Piracicaba,SP,maiode2004, p. 17-20.

LATTA, J. N. & Oberg,D. J. - A Conceptual Virtual Reality, IEEE Computer Graphics & Applications, Jan. 1994.

HUTZLER, G.; GORTAIS, B.; DROGOUL, A. Grounding Virtual Worlds in Reality. In: VIRTUAL WORLDS, LNAI 1434, 1998, Berlim, Proceedings. Berlim: Springer-Verlag Berlin Heidelberg, 1998. p.274-285.

MICHALEWICZ, Z. Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 2000

PASSINI, E. As representações gráficas e a sua importância para a formação do cidadão. Revista Geografia e Ensino, Belo Horizonte, v.6, n.1, mar. 1997.

PEPONIS, J., Zimring, C., and Choi, Y.K., Finding the Building in Wayfinding. Environment and Behavior,.Academic Press, New York, NY, 1995.

PIMENTEL, K. & Teixeira, K. - Virtual Reality Though the New Looking Glass, Intel/WindCrest/McGraw-Hill, New York, NY, 1995.

PINHO, M. - “Painel de Interação- Configurável para Ambientes Virtuais Imersivos”, em: Workshop de Realidade Virtual, 2004, Gramado, RS, Anais... Porto Alegre, RS

RABUSKE, R. A. Inteligência Artificial. Editora Ufsc. 1996.

REMO, J.F.B, Sementille,A.C. Kirner C, Devidé, A.H, Santos F, Beldi, L. H. P. A Library for Movement of Agents and Hunanoids Avatars in Applications of Virtual Reality. IV Simpósio de Realidade Virtual, 2001, Florianópolis, SC.

RICH, E. Inteligência Artificial .Mc Graw-Hill, 1988

RIGO, S. Animação por Computador: Um Estudo de Técnicas. Porto Alegre: CPGCC da UFRGS, 1992. 100p.

ROHEL, B.; et al. Late Night VRML 2.0 with Java. Ziff-Davis Press. 1997.

ROEHL, B. Specification for a Standard VRML Humanoid. H-ANIM WG, U.Waterloo, Canada, 1999. http://ece.uwaterloo.ca/˜h-anim/spec.html.

RUSSELL, Stuart, NORVIG, Peter. Artificial Intelligence: a Modern Approach. 2st edition.Upper Saddle River: Prentice-Hall, 2004.

SATALICH, G.. Navigation And Wayfinding In Virtual Reality: Finding The Proper Tools And Cues To Enhance Navigational Awareness. Master's Thesis, University of Washington, Seattle, 1998.

SELMAN, Daniel. Java 3D Programming. Manning Publications, 2002.

SILVA, M. H. Tratamento Eficiente de Visibilidade Através de Árvores de Volumes Envolventes. Dissertação de Mestrado, PUC/RJ, 2002.

SUN Microsystems. Disponível: Sun Microsystems, Inc. site (2005). URL: http://www.sun.com

THALMANN, D. A New Generation of Synthetic Actors: the Real-Time and Interactive Perceptive Actors, In: Pacific Graphics, 96, Taiwan,Proceedings…, 1998.

WALSH, A. E. Understanding Scene Graphs. Dr Dobb's Journal, 27:7, 17-26, 2002.