Adailton de Jesus Cerqueira Junior

IFAS3D: Uma Ferramenta para Teste de Movimento em Robôs Humanóides em Ambiente Simulado

Rua Silveira Martins, 2555, Cabula, Salvador-BA

16 de outubro de 2011

Adailton de Jesus Cerqueira Junior

IFAS3D: Uma Ferramenta para Teste de Movimento em Robôs Humanóides em Ambiente Simulado

IFAS3D: Uma Ferramenta para Teste de Movimento em Robôs Humanóides em Ambiente Simulado

Áreas da Computação:
Robótica Inteligente, Inteligência Artificial, Engenharia de Software

Orientador:
Diego Gervasio Frías Suárez

Co-orientador:
Marco Antônio Costa Simões

UNEB - Universidade do Estado da Bahia
Departamento de Ciências Exatas e da Terra

Rua Silveira Martins, 2555, Cabula, Salvador-BA

16 de outubro de 2011

Monografia de Projeto Final de Graduação sob o título *"IFAS3D: Uma Ferramenta para Teste de Movimento em Robôs Humanóides em Ambiente Simulado"*,

defendida por Adailton de Jesus Cerqueira Junior e aprovada em 16 de outubro de 2011, em Salvador,

Estado da Bahia, pela banca examinadora constituída pelos

professores:

Prof. Phd. Diego G. Frías Suárez
Orientador

Prof. Msc. Marco A. C. Simões
Co-orientador

Prof. Msc. Antonio C. Fontes Atta
Universidade do Estado da Bahia

Prof. Msc. Trícia S. Santos
Universidade do Estado da Bahia

Resumo

A robótica inteligente pode ser definida como a arte de criar máquinas que executam ações inteligentes. Visando este desenvolvimento foi criada a RoboCup, uma iniciativa para impulsionar as pesquisas na área de inteligência artificial e robótica, que tem como meta desenvolver, até 2050, uma equipe de robôs humanóides autônomos capazes de derrotar os campeões mundiais de futebol. Dentre as competições da RoboCup existe a liga de simulação 3D, que simula uma partida de futebol em um ambiente tridimensional com robôs humanóides.

Diferente de outras ligas de simulação, dentro da simulação 3D não existem comandos de alto nível para movimentação e sim um conjunto de velocidades que devem ser enviadas para os motores posicionados nas juntas do robô, como em um robô real. Entretanto, esse conjunto de velocidades não é previamente definido, cabendo aos times descobrirem os valores dessas velocidades para efetuar o movimento desejado.

Este trabalho propõe a construção de uma ferramenta que auxilie nos testes de movimentos em robôs humanóides no ambiente simulado da liga de simulação 3D, com o intuito de facilitar o desenvolvimento dos movimentos dos agentes simulados e também a compreensão, de forma empírica, do ambiente simulado. O IFAS3D oferece interação em tempo real com o simulador o que permite um acompanhamento das juntas durante o movimento, além de uma interação utilizando scripts, para construção de movimentos mais rápidos e complexos.

Palavras-chave: Robótica Inteligente, RoboCup, SimSpark, Futebol de Robôs.

Abstract

Intelligent robotics may be defined as the art of making machines able to execute intelligent actions. It was aiming on such development that RoboCup was created, an initiative to promote researches on Artificial Intelligence, robotics and co-related areas, which goal is to develop a team of humanoid robots fully autonomous capable of defeating FIFA's world champions, by the year of 2050. Amongst RoboCup competitions, there is the 3D Soccer Simulation league, which simulates soccer matches in a three-dimensional environment, with humanoid robots.

Unlike other simulated leagues, within 3D simulation there aren't high level commands for robots' movements, but a set of velocities meant to be sent to robots' joints engines, as in a real robot. However, velocities values are not predefined: the teams must discover them in order to execute the desired movement.

This work proposes building a tool to aid testing robots' movements in the 3D simulation environment, in order to help developing the simulated agents movements, and in an empirical comprehension of the environment, as well. IFAS3D offers real time interaction with the Soccer Simulator, allowing one to track joints during the movement, it also supports script driven interaction, for building faster and more complex movements.

Keywords: Intelligent Robotics; RoboCup; SimSpark; Robot Soccer.

Agradecimentos

Agradeço, primeiramente, aos meus pais, Adailton Cerqueira e Valnisia Pedra, a minha avó Dona Franscisca, a minha Tia Jaciara e minha família por me apoiarem durante essa jornada.

Agradeço também aos meus amigos Bianca Sapucaia, Carolina Sapucaia, Daniel Cason, Flávio Sapucaia, Taís Machado, Ranyer Lopes e toda família Sapucaia, principalmente Dona Ada, pela ajuda e companheirismo.

Agradeço a Diego Frías e Marco Simões, meus orientadores, pelos conselhos ao longo desta caminhada, a Leandro Coelho pelas dicas e sugestões durante a escrita deste trabalho e a Josemar Souza pela coordenação do núcleo de pesquisa ACSO.

Agradeço também a todos os professores, funcionários e técnicos que contribuíram na formação técnica e humana dos alunos do curso de Sistemas de Informação. Principalmente a Ana América por toda ajuda dada durante o curso.

Agradeço aos integrantes e amigos do ACSO que me ajudaram durante momentos de dúvida e contribuíram de maneira fundamental com a conclusão deste trabalho. Obrigado Adriano Veiga, Alan Deivite, Ayran Cruz, Bruno Vinicius, Camila Laranjeira, Emmanuel Argollo, Fagner Moura, José Grimaldo, Juliana Fajardini, Mario Bortoli, Murilo Reis, Rafael Factum e Vitor Santos.

Agradeço aos amigos que encontrei ao longo da caminhada na UNEB: Elton Fraga, Felipe Piñeiro, Fernanda Brito, George Dias, Glória Araújo, Henrique Filho, Jussi Barros, Marlena Martins, Rafael Guimarães, Raul Abreu, Tatiana Paz e Vanessa Rios.

Gostaria de agradecer e me desculpar também a todos os amigos e colegas que fizeram parte deste percurso, dentro ou fora da UNEB, os quais os nomes não foram citados aqui.

Agradeço à contribuição e apoio decisivo do Magnífico Reitor Luourisvaldo Valentim na realização do projeto do grupo de pesquisa ACSO.

"*Parte da ausência de humanidade do computador deve-se a que,*
competentemente programado e trabalhado bem,
é completamente honesto"
Isaac Asimov

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

A idéia de criar um ser parecido com o ser humano é um assunto que sempre encantou a humanidade. Várias mitologias antigas já possuíam histórias de seres artificiais dotados de inteligência própria, desde os golens das lendas judaicas até os servos mecânicos de Hefesto da mitologia grega. A estes seres damos o nome de robôs, que vem da palavra tcheca robota, que significa trabalho forçado. Este termo foi introduzido pelo escritor tcheco Karel Capek em sua peça Rossum's Universal Robot's, encenada em 1921.

Na busca deste sonho, a ciência tem apresentado grandes avanços no desenvolvimento de técnicas e tecnologias para construção destes seres artificiais. Desde robôs capazes de executar tarefas repetitivas nas linhas de montagens de automóveis a robôs que apoiam em tarefas domésticas.

Com o intuito de desenvolver ainda mais este ramo da ciência, cientistas do mundo inteiro fundaram em 1996 a RoboCup Federation (ROBOCUP, 2011). O principal objetivo desta organização é fomentar o desenvolvimento da robótica autônoma, promovendo o desenvolvimento de tecnologias utilizáveis em problemas sociais e industriais (FONSECA; PEREIRA; GUERRA, 2002). Para isto, organiza anualmente a copa mundial de robôs, que consiste em competições entre robôs em diferentes modalidades. Paralelamente o evento abriga debates e apresentações dos avanços científicos e tecnológicos na área da robótica. A meta oficial da Robocup é desenvolver uma equipe de robôs humanóides, totalmente autônomos, que derrote a equipe campeã mundial de futebol até o ano de 2050 (KITANO et al., 1997).

As competições de futebol de robôs estão divididas em duas modalidades: com robôs reais e simulados. Em ambas modalidades cada jogador é controlado por um software agente autônomo (RUSSEL; NORVIG, 2002). Este agente tem que ser capaz de perceber o ambiente através da interpretação dos sinais, enviados por diferentes sensores; identificar o estado e tomar uma decisão através de um raciocínio; e efetuar as ações correspondentes, utilizando atuadores. Essas ações são muito complexas nos casos de robôs humanoídes,

pois exigem uma sequência de comandos de controle (scripts) para movimentar de forma sincronizada as múltiplas articulações de seus membros. Vale ressaltar que cada time é responsável pela elaboração dos seus scripts de controle e que os resultados nas competições estão diretamente relacionados com a qualidade dos mesmos. Um script de qualidade é aquele que executa o movimento desejado de forma rápida e estável.

Para desenvolver um script de controle é preciso realizar testes repetidas vezes, seguindo um paradigma de tentativa e erro. Em cada tentativa os parâmetros de controle das articulações, a saber, a velocidade angular do servo motor que aciona cada junta e a duração do movimento com essa velocidade, são ajustados e é efetuado um novo teste.

Em nosso caso, cada mudança de parâmetro implicava em modificação do código-fonte seguida de compilação, o que por um lado consome tempo e por outro requer um conhecimento da estrutura do código e da linguagem de programação por parte do desenvolvedor do script de controle, o que exclui os novatos desse processo. Além disso, como o comportamento e as ações do robô são determinadas pela camada de inteligência, para testar uma habilidade motora em particular, era necessário esperar que o robô estivesse em uma situação na qual essa movimentação fosse indicada. Por exemplo, para chutar a bola, esta precisa estar suficientemente próxima ao jogador. Caso o jogador não esteja a esta distância, a camada de inteligência ordena primeiro o jogador a se aproximar da bola para depois chutar. Entretanto, se o movimento de andar ainda não estiver estável, o jogador pode cair ou se desviar e não chegar até a bola, fracassando o teste do movimento de chutar. Isso impossibilita o desenvolvimento de alguns movimentos em paralelo, além de consumir um tempo desnecessário.

Este projeto aborda a construção de uma ferramenta para facilitar o desenvolvimento das capacidades motoras dos robôs humanóides simulados em ambiente tridimensional.

Este trabalho está organizado da seguinte forma: o capítulo 2 descreve o ambiente de simulação 3D, as características do servidor oficial utilizado pela liga, seu robô simulado e o software de visualização de simulação; no capítulo 3 é explicada a metodologia usada para o desenvolvimento da ferramenta; no capítulo 4 é exposto o resultado do desenvolvimento e o funcionamento da ferramenta IFAS3D, além dos testes realizados. Um artigo com resultados parciais do projeto foi publicado no *Workshop de Trabalhos de Iniciação Científica e de Graduação* em abril de 2011, durante a Escola Regional de Computação Bahia-Alagoas-Sergipe - ERBASE. Visando a não duplicação de informações este artigo foi anexado na íntegra (apêndice D) e referenciado no texto. O capítulo 5 traz considerações finais e trabalhos futuros.

2 Contextualização

O fato da meta oficial da Robocup ser relacionada com futebol deve-se à natureza lúdica e atrativa deste desafio. O futebol é um dos esportes mais conhecidos no mundo, além de reunir a maioria dos problemas de um ambiente dinâmico e cooperativo. Para se ter um time de robôs capaz de disputar uma partida de futebol é necessário o desenvolvimento de várias tecnologias em diversas áreas científicas. Entre estas áreas podemos citar (FONSECA; PEREIRA; GUERRA, 2002):

- **Inteligência Artificial e Robótica:** Sistemas multiagentes, aprendizagem individual e distribuída, planejamento estratégico, raciocínio em tempo real, aquisição de conhecimento, adaptação, processamento de linguagem natural, etc.
- **Engenharia Elétrica e Eletrônica:** Sensores robustos, hardware capaz de reagir a falhas e impactos, suporte em tempo real, etc.
- **Ambientes virtuais e simulações distribuídas:** Protocolos de comunicação multicast, sistemas de visualização em tempo real, protocolos de comunicação compactos, etc.

Além do desafio de futebol a RoboCup possui outros desafios, como por exemplo, desafios relacionados a soluções de buscas e salvamento em ambientes de difícil acesso. Os desafios são divididos em quatro competições que estão divididas em categorias, onde cada uma possui ligas e subligas, como pode ser visto na tabela 2.1.

Dentre estas categorias encontra-se a RoboCup Soccer que é uma competição de futebol de robôs que possui diversas ligas. Uma dessas ligas é a liga de simulação de robôs 3D. Nesta liga é simulada uma partida de futebol em um ambiente tridimensional. Desta forma, questões mais avançadas de inteligência artificial podem ser estudadas, abstraindo questões de hardware. Existe, entretanto, uma preocupação da liga de não se afastar muito do mundo real. Por este motivo, atualmente o robô simulado é o Nao (figura 2.1),

Tabela 2.1: Tabela das ligas e subligas da RoboCup

RoboCup		
Categoria	**Liga**	**Subliga**
RoboCup@Home	-	-
RoboCup Junior	Soccer Dance Rescue	- - -
RoboCup Rescue	Robot league Simulation league	- -
RoboCup Soccer	Middle size league Small size league Standard Platform Humanoid league Simulation league	- - - Kid-size e Teen-size Simulation 2D e Simulation 3D

que também é utilizado como padrão na Standard Platform. O Nao é um robô humanóide autônomo e programável desenvolvido pela empresa francesa Aldebaran Robotics (ALDEBARAN, 2011).

Figura 2.1: Robô Nao

Na próxima seção se descreve a Subliga de Simulação 3D incluindo o ambiente computacional e os diferentes componentes de software.

2.1 Subliga de Simulação 3D

A subliga de simulação 3D é uma simulação de futebol de robôs, onde os jogadores são simulados em um ambiente tridimensional. A simulação compreende tanto a parte mecânica, quanto a aplicação das regras do jogo de futebol. No estágio atual cada time

consta de 9 jogadores, sendo um deles o goleiro, e jogam dois tempos de 300 segundos. O campo virtual mantêm as proporções do campo oficial e sua dimensão é ajustada de acordo com o número de jogadores.

O simulador envia, para cada agente, informações sobre o ambiente e de si mesmo, fruto da simulação dos sinais gerados pelos sensores do robô. Baseado nessas informações, o agente deve tomar decisões e agir consequentemente, o que em modo geral implica que, o robô controlado por ele, realize algum tipo de movimento.

De acordo com a taxonomia utilizada por (RUSSEL; NORVIG, 2002), podemos classificar o ambiente da simulação 3D como: parcialmente observável; estocástico; sequencial; dinâmico; discreto e multi-agente, diferindo de um ambiente real apenas no que se refere a discretude, já que até onde sabe-se, a realidade é contínua.

2.1.1 Ambiente Computacional

Nas competições da subliga de simulação 3D cada time deve iniciar um agente para cada jogador, enquanto o organizador inicia o simulador e o software de visualização da simulação, doravante chamado monitor.

Estes softwares podem ou não ser executados no mesmo computador, mas por via de regra o simulador é executado em um computador dedicado, enquanto todos os agentes de um mesmo time em outro computador. O monitor é também executado em um computador dedicado. A comunicação entre todos eles é efetuada via rede utilizando protocolo TCP/IP.

Tem sido observado um desempenho diferente de um mesmo script de controle quando executado em diferentes ambientes computacionais, tanto por diferenças na distribuição dos processos, quanto na capacidade de processamento dos computadores. Por exemplo, um script de controle que funciona corretamente em um ambiente onde todos os processos são executados em um único computador, pode não funcionar satisfatoriamente em um ambiente distribuído, o que também se aplica para o caso contrário.

Embora as causas desse comportamento instável não estejam totalmente esclarecidas, acredita-se que seja originado por perda de sincronismo entre os agentes e o simulador. Essa perda de sincronismo pode estar associada a latência variável de rede ou a grandes variações do tempo real gasto pelo motor de simulação física na simulação da mecânica do jogo, como será visto na próxima seção.

2.1.2 SimSpark

O SimSpark (BOEDECKER; ASADA, 2008) é um sistema de simulação multi-agente para agentes em ambientes tridimensionais e é utilizado como o simulador oficial da subliga de Simulação 3D. Seu objetivo é proporcionar um alto grau de flexibilidade para a criação de novos tipos de simulações. Com isto, o SimSpark se torna uma ferramenta poderosa para diferentes pesquisas nas questões referentes à sistemas multi-agentes.

O simulador SimSpark foi baseado no Spark (ROLLMANN, 2004), que é um simulador genérico de física multiagente para ambientes tridimensionais. O Spark tem sua estrutura baseada em três componentes principais:

- Gestor de memória e objetos;
- Motor de simulação física;
- Motor de simulação.

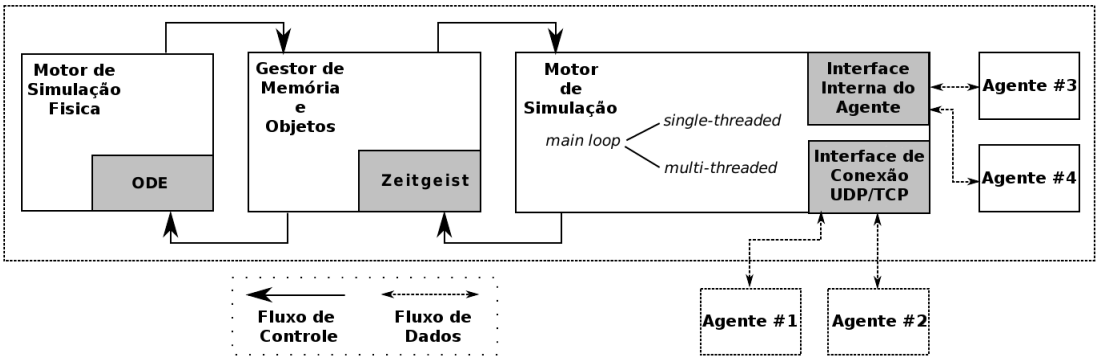


Figura 2.2: Fluxo de controle e dados entre os principais componentes do simulador. Adaptada de (BOEDECKER; ASADA, 2008)

Gestor de Memória e Objetos

O componente central do gestor de memória e objetos é chamado Zeitgeist (KOGLER, 2003), que é responsável por fornecer uma abstração às operações de sistemas, como suporte a arquivos e diretórios, logs, bibliotecas compartilhadas, entre outros. Além disto, fornece uma interface de scripts para linguagem Ruby (RUBY, 2011) e um mecanismo de plugins acoplados com uma hierarquia de objetos.

Basicamente existem três tipos de objetos gerenciados pelo Zeitgeist: Objetos representando o cenário atual; objetos encapsulando a funcionalidade central do simulador e

fábricas para a criação de novos objetos com base em texto. Essa característica é importante para a flexibilidade e extensibilidade, pois permite a alteração da configuração do simulador sem a necessidade de recompilar o código inteiro (OBST; ROLLMANN, 2005).

Motor de Simulação Física

O segundo componente principal do Spark é o motor de simulação física. Este motor é responsável pela detecção de colisão e atualização das posições e velocidades dos objetos simulados. A principal biblioteca utilizada neste componente é a ODE - Open Dynamics Engine (SMITH, 2006).

Desenvolvida em C++, por Russell Smith, esta biblioteca foi projetada para ser utilizada em simulações interativas ou em tempo real, oferecendo simulações de movimento de corpos rígidos articulados em ambientes virtuais dinâmicos. Uma estrutura articulada é criada quando corpos rígidos de várias formas são ligados entre si com juntas (figura 2.3).

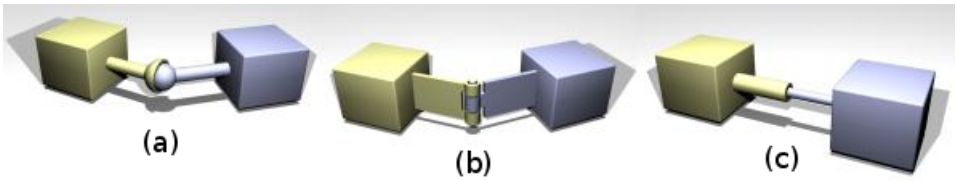


Figura 2.3: Três tipos diferentes de juntas. Adaptada de (SMITH, 2011)

Algumas características básicas da ODE utilizadas pelo Spark serão explicadas abaixo. Maiores explicações podem ser encontradas no manual do usuário (SMITH, 2006) e na wiki oficial do projeto (SMITH, 2011).

Corpos Rígidos Na simulação da ODE, um corpo rígido possui diversas propriedades que mudam ou não ao longo do tempo.

- **Variáveis com o tempo**: Posição (x, y, z) do ponto de referência do corpo, que atualmente é o centro de massa; velocidade linear do ponto de referência; orientação do corpo e velocidade angular.
- **Não variáveis com o tempo**: Massa do corpo; centro de massa e distribuição da massa ao redor do corpo.

O processo de simulação do corpo rígido ao longo do tempo é chamado de integração. A cada passo da integração é avançado o tempo atual de acordo com o tamanho dado

ao passo, com isso ajustando o estado de todos os corpos rígidos para o novo valor do tempo. Utilizar uma diferença de tempo pequena para cada passo de atualização, torna a simulação mais precisa, entretanto, o processo levará mais tempo para executar.

Cada corpo possui um sistema de coordenadas (x, y, z), que se movimenta juntamente com ele (Figura 2.4).

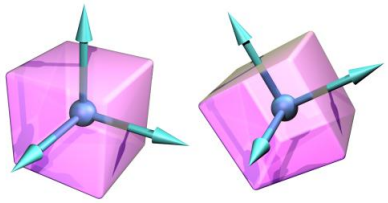


Figura 2.4: Sistema de coordenadas do corpo. Retirado de (SMITH, 2011)

Articulações É uma relação que é aplicada entre dois corpos de forma a limitar as posições e orientações de um em relação ao outro. Esta relação também é chamada de restrição (SMITH, 2006). A figura 2.3 mostra três tipos de articulações: (a) do tipo esfera e soquete; (b) do tipo dobradiças e (c) do tipo pistão.

Cada vez que ocorre uma integração todas as articulações ficam habilitadas a exercer pressão nos corpos que afetam. Essas forças são calculadas para que os corpos movam-se de forma coordenada, preservando os relacionamentos em comum.

Motor Angular Um motor angular permite que as velocidades angulares relativas de dois corpos sejam controladas. A velocidade angular pode ser controlada em até três eixos, permitindo que os motores de torque e parada sejam ajustados para a rotação sobre aqueles eixos. Isto é útil principalmente em conjunto com articulações esféricas, podendo ser usado em qualquer situação que haja necessidade de controle angular.

Pode-se usar um motor angular em dois modos: dAMotorUser, onde os eixos de rotação e os ângulos das articulações são controlados pelo usuário; dAMotorEuler, onde os ângulos são calculados automaticamente.

Detecção de Colisão A detecção de colisão entre corpos segue os seguintes passos (SMITH, 2011): Antes de cada etapa de simulação, as funções de detecção de colisão identificam quais objetos se tocaram; um contato especial é criado para cada ponto de

contato; os contatos são colocados em um grupo de contatos; um passo de simulação é efetuado; por fim, todos os grupos são removidos do sistema.

Os grupos de contatos tem a função de indicar características sobre o contato, como o atrito da superfície. Este grupo também determina a velocidade da simulação, pois quanto mais grupos de contatos houver menor será a velocidade da simulação.

O modelo de atrito nos pontos de contato baseia-se no modelo de atrito de Coulomb. Este modelo é uma relação entre as forças normal e tangencial presente em um ponto de contato: $|\mathbf{f_T}| <= \mu * |\mathbf{f_N}|$, onde $\mathbf{f_T}$ e $\mathbf{f_N}$ são vetores de força tangencial e força normal, respectivamente, e μ é o coeficiente de atrito.

Esta equação define um cone de atrito, onde o $\mathbf{f_N}$ é o eixo e o vértice é o ponto de contato. Se o vetor de força total está dentro do cone, então a força de atrito é suficiente para evitar que os contatos deslizem. No entanto, se o vetor de força está na superfície do cone a força de atrito não é grande para impedir o deslizamento dos contatos.

Os modelos de atrito da ODE são aproximações do cone atrito, para ser computacionalmente menos custoso.

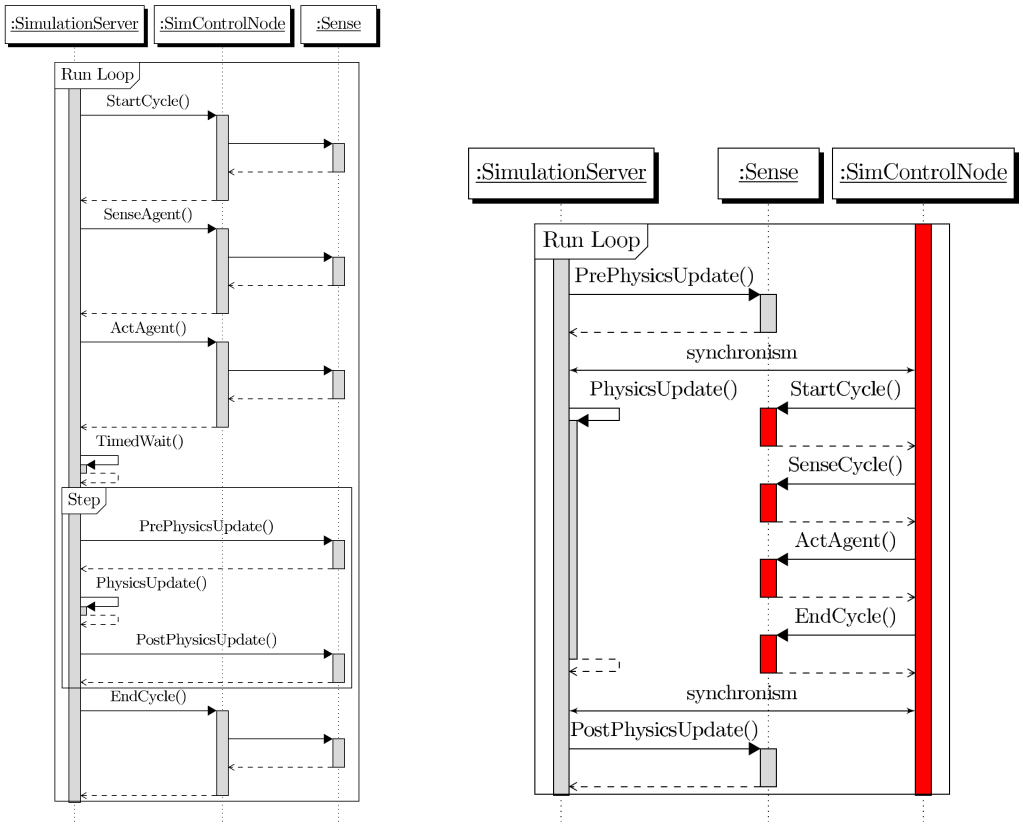
Gestor de Simulação

O terceiro componente é responsável pelo controle do loop principal do simulador, gerenciamento de eventos, comunicação com processos externos e interação entre o simulador e os agentes. O SimSpark possui dois tipos de loop principal: Um loop simples que executa as ações dos agentes assim que chegam e um mais elaborado usando o *middleware* Spades (RILEY; RILEY, 2003).

O loop simples não compensa latência de rede ou recursos de computação dos agentes. A consequência disto é que o SimSpark não garante que os eventos serão reprodutíveis, ou seja, eventos repetidos podem apresentar resultados diferentes (BOEDECKER; ASADA, 2008). Isso pode ser causado por atrasos de rede ou desempenho das máquinas utilizadas. A vantagem do loop simples é o ganho de velocidade na simulação que permite executar partidas com mais agentes.

Outra vantagem do loop simples é que trabalha utilizando plugins, que torna o SimSpark configurável em tempo de execução. Esses plugins são chamados *simcontrol nodes* e são utilizados em resposta a eventos de controle. Os principais eventos são “init”e “done”, para início e fim da execução do servidor, respectivamente; além dos eventos de loop de simulação: “start cycle”, “sense agent”, “act agent”e “end cycle”.

O SimSpark pode ser executado em dois modos: Single-threaded e Multi-threaded. O modo Single-threaded executa o loop principal através dos eventos “start cycle”, “sense agent”, “act agent”e “end cycle”repetidamente, como demonstrado na figura 2.5a. Enquanto no modo Multi-threaded a simulação física e o processamento para comunicação com agente para o próximo ciclo são executados em paralelo, como demonstrado na figura 2.5b.



(a) Single-threaded (b) Multi-threaded

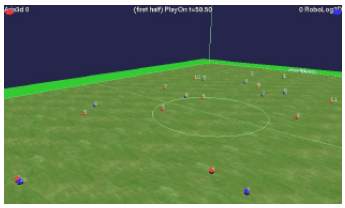
Figura 2.5: Diagrama de fluxo dos modos de execução do SimSpark. Retirado de (SIMSPARK, 2011)

Por último, é necessário explicar sobre o envio das mensagens para os agentes. Este envio ocorre a cada ciclo de 20ms. Vale ressaltar que essa duração depende do desempenho do computador utilizado e do número de atualizações físicas. Ou seja, se a simulação for mais rápida que o tempo real o SimSpark aguarda o término deste tempo. Entretanto, se a simulação for mais lenta que o tempo real o SimSpark continuará executando as atualizações físicas sem interação com os agentes.

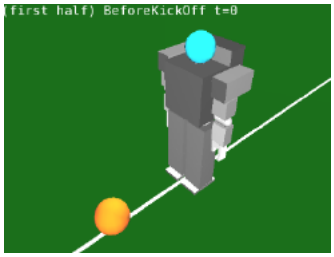
Demais informações sobre o simulador SimSpark podem ser encontradas na wiki do projeto (SIMSPARK, 2011).

2.1.3 Agente Físico Simulado

Existe um esforço da Subliga de Simulação 3D para reproduzir os desafios enfrentados na construção de robôs reais. Por este motivo, o SimSpark tem passado por vários modelos diferentes de robôs simulados, os mais relevantes são: a esfera oni-direcional, usada em 2004 (Figura 2.6a); o robô HOAP-2, usada em 2006 (Figura 2.6b) e o robô Nao, utilizado desde 2008 (Figura 2.6c).



(a) Espera Oni-direcional (2004-2005)



(b) Robô HOAP-2 (2006-2007)

(c) Robô Nao (2008-Atualmente)

Figura 2.6: Diferentes versões dos robôs utilizados pelo SimSpark. Retirado de (FILHO, 2009)

Atualmente, a liga simula como agente físico o robô Nao, que possui uma arquitetura bípede, com 22 graus de liberdade (figura 2.7a), que permite ter uma grande mobilidade. Entretanto, a empresa Aldebaran Robotics, não permitiu que o Nao fosse simulado com perfeição, havendo diferença nas escalas dos graus de liberdade entre o Nao simulado e o real. Na tabela 2.2 é feito um comparativo entre as escalas (máximo e mínimo) de ângulos das juntas do Nao real (ROBOTICS, 2008) com o simulado (SIMSPARK, 2011), demonstrando a tentativa de simular o Nao com perfeição.

Para simular os 22 graus de liberdade foram implementados 22 motores posicionados nas articulações do Nao. Além das articulações, foram implementados também sensores para visão, audição, sensores de toque e aceleração(figura 2.7b). Para o robô simulado foram implementados 5 tipos de sensores e 1 atuador, além das juntas.

Segundo Stuart Russel e Peter Norvig (RUSSEL; NORVIG, 2002), um agente é uma entidade que percebe o ambiente através de sensores e atua sobre ele através de atuadores, a figura 2.8 demonstra essa interação. Este ambiente pode ser tanto real, robôs físicos, quanto simulado. Partindo desta definição, podemos classificar os sensores como dispositivos ou informações que possibilitem o robô perceber o ambiente ao seu redor e os atuadores como dispositivos ou ações que possibilitem o robô modificar o estado do ambiente.

[1] As juntas encontram-se numeradas de acordo com a figura 2.7a

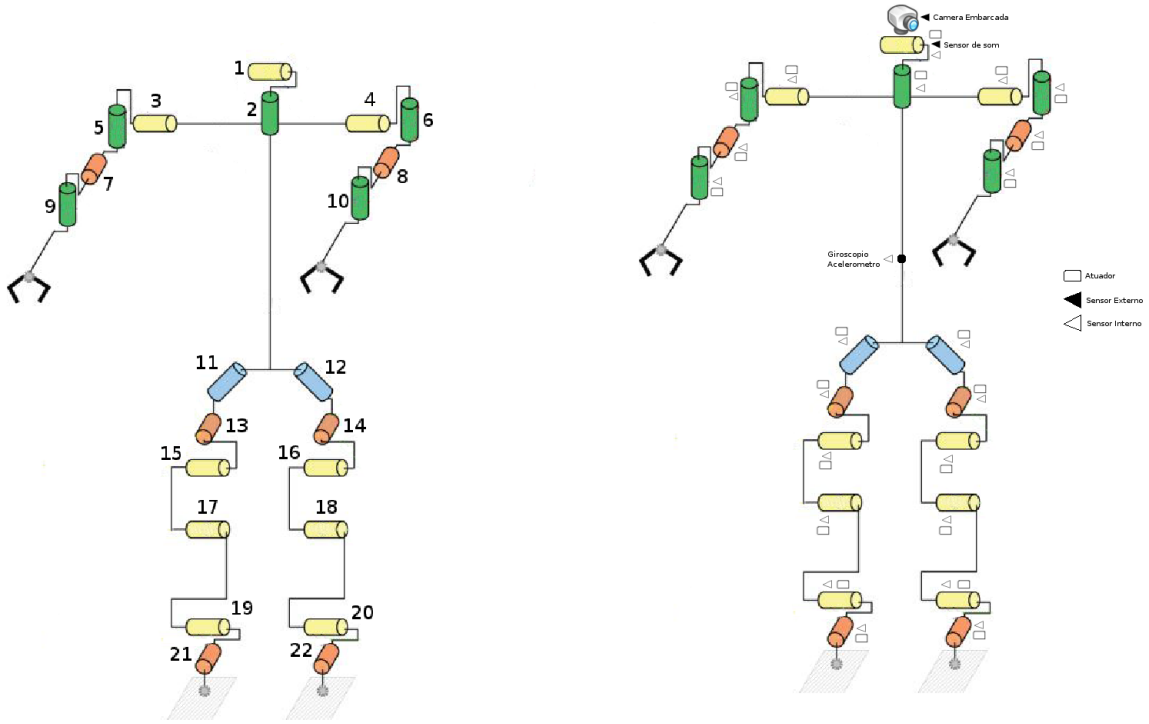
Tabela 2.2: Tabela de comparação entre o Nao Real e o Nao Simulado.

		Nao Real		*Nao Simulado*	
Junta[1]	Nome	**min**	**max**	**min**	**max**
1	hj2	-45	45	-45	45
2	hj1	-120	120	-120	120
3 e 4	raj1 e laj1	-120	120	-120	120
5	raj2	-95	0	-95	1
6	laj2	0	95	-1	95
7 e 8	raj3 e laj3	-120	120	-120	120
9	raj4	-90	0	-1	90
10	laj4	0	90	-90	1
11 e 12	raj1 e laj1	-90	0	-90	1
13	raj2	-45	25	-45	25
14	laj2	-25	45	-25	45
15 e 16	raj3 e laj3	-100	25	-25	100
17 e 18	raj4 e laj4	0	130	-130	1
19 e 20	raj5 e laj5	-75	45	-45	75
21	raj6	-25	45	-25	45
22	laj6	-45	25	-45	25

Para simular os atuadores e os sensores o SimSpark utiliza um sistema de troca de mensagens com o agente. Estas mensagens são baseadas em uma estrutura de dados *S-expressions*(SEIBEL, 2005). Um exemplo destas mensagens encontra-se no apêndice A. Uma vantagem de usar *S-expressions* sobre outros formatos de dados é que ela proporciona uma análise fácil e sintaxe compacta que é, em certa medida ainda legível por seres humanos para fins de depuração. Além disso, as *S-expressions* servem para diminuir o tráfego de informação na rede.

2.1.4 Software de Visualização da Simulação: RoboViz

O RoboViz (STOECKER; VISSER, 2011) é um software para visualização da simulação. Interage diretamente com o simulador, que envia todas as informações renderizadas para serem representadas na tela. Vale ressaltar, que o Simspark contém um visualizador nativo, mas tem uma performace e qualidade inferiores ao Roboviz.



(a) Graus de liberdades das juntas do Robô Nao (b) Sensores e Atuadores do Robô Nao

Figura 2.7: Configurações internas do Robô Nao. Adaptada de (SIMSPARK, 2011)

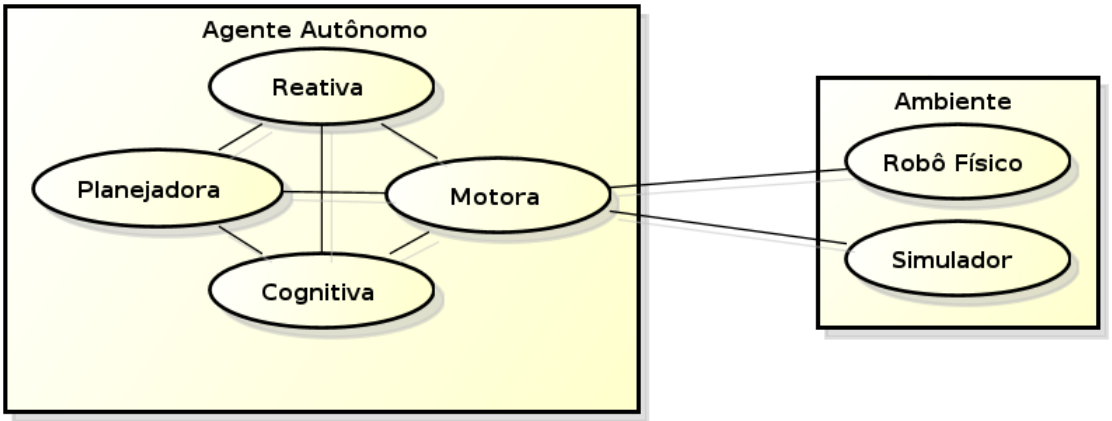


Figura 2.8: Interação do agente autonômo com ambiente

3 Metodologia

A construção de uma ferramenta para facilitar o desenvolvimento das capacidades motoras dos robôs humanoídes simulados em ambiente tridimensional, foi baseada nos seguintes critérios de desenho. A ferramenta deve ser:

- Capaz de executar os movimentos de forma imediata;
- Compatível com o simulador utilizado pelo agente;
- Fácil de usar, mesmo por pessoas sem o conhecimento do código do agente;
- Estável em diferentes ambientes computacionais;
- Configurável para ser executada em modo interativo ou a partir de arquivo de lotes (*batch*).

Decidiu-se, então, por criar uma ferramenta para testar os scripts de controle de movimentos, isolados do ambiente de jogo, mas utilizando o modelo físico disponibilizado pelo simulador. Para isso, planejou-se uma arquitetura semelhante a do agente autônomo (figura 2.8), mas sem os módulos referentes a camada de inteligência (figura 3.1). A vantagem disso é que fica garantida a compatibilidade com o simulador e a estabilidade de ambientes, já que o agente atende a esses critérios. E com a retirada da camada de inteligência evita-se os problemas de interferência, o que acaba fazendo com que os movimentos sejam executados imediatamente.

Com o intuito de facilitar a utilização da ferramenta e torná-la interativa foi criada uma interface com o usuário. Essa interface permite que os movimentos sejam testados utilizando um sistema de interação em tempo real ou por meio de *batch* chamados de scripts de controle de movimentos, que serão melhores explicados na seção 4.3.

Para o processo de desenvolvimento da ferramenta foi adotado o sistema de desenvolvimento incremental que foi dividido em oito fases:

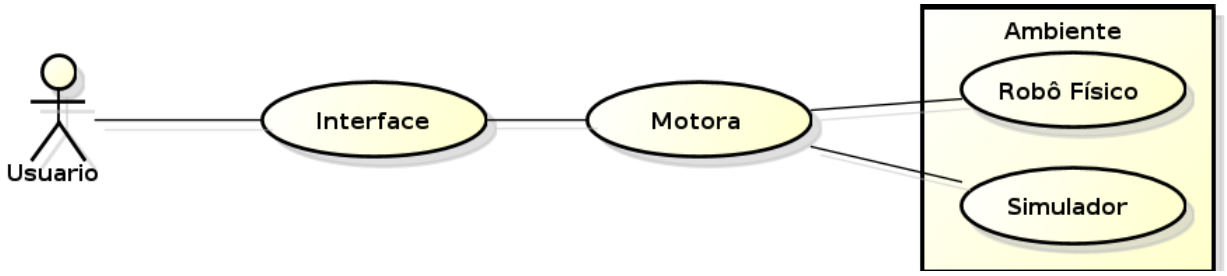


Figura 3.1: Interação do IFAS3D com ambiente

- análise,
- definição da arquitetura,
- definição dos módulos,
- implementação da conexão,
- implementação da troca de mensagens,
- implementação do armazenamento das informações,
- implementação da interface com o usuário,
- teste de conexão e funcionalidades.

Após as fases de análise e definição da arquitetura foram definidos cinco módulos para ferramenta, que são: Conexão, Sensor, Atuador, ModeloMundo e Main. O diagrama de classe simplificado (figura 3.2) ilustra a interação destes módulos.

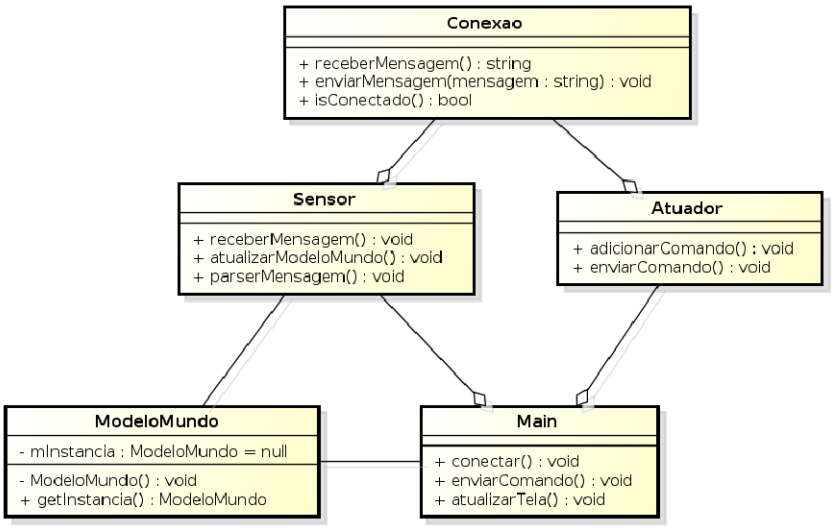


Figura 3.2: Diagrama de classe simplificado

A ferramenta foi desenvolvida em C++ utilizando o *framework* Qt (QT, 2011). Uma prova de conceito foi feita em Java, mas esta indicou um grande consumo de memória, devido a isso escolheu-se a linguagem C++.

4 Resultados

4.1 IFAS3D

O IFAS3D é uma ferramenta que tem o intuito de testar os movimentos de um robô humanóide simulado pelo SimSpark, além de auxiliar o entendimento dos sensores e atuadores, focando-se mais nas juntas. Busca-se, com esta ferramenta auxiliar, de forma empírica, a compreensão da física que atua sobre o agente.

A ferramenta tem como funcionalidade estabelecer uma conexão com o simulador e efetuar trocas de mensagens. A partir destas mensagens serão extraídas informações para verificarmos as alterações sobre o robô. As principais informações sobre o robô mapeadas serão: o posicionamento atual das juntas, o valor do giroscópio, o valor do acelerômetro e o ciclo de atualização do simulador.

Efetuando um comparativo entre as figuras 2.8 e 3.1 percebe-se que enquanto o agente autônomo não possui interação com o usuário, o IFAS3D possui. Isso agiliza a construção dos movimentos, pois o usuário pode interagir em tempo real, sem necessidade de recompilar o código toda vez que houver necessidade de teste de movimentos. Outra característica é que a ferramenta foi idealizada para conectar-se tanto em ambiente simulado quanto real, tornando-se genérica para construção de movimentos dos robôs humanóides.

4.2 Estrutura

4.2.1 Conexão

Como resultado da fase de implementação da conexão obtivemos o módulo Conexão que é responsável por efetuar a conexão com o simulador e efetuar as trocas de mensagens. Essa comunicação pode ser efetuada por diversos meios, como por exemplo, sockets ou comunicação entre processos. Atualmente, o SimSpark, realiza uma conexão de sockets utilizando o protocolo TCP.

Para tornar o programa mais versátil na comunicação foi adotada uma interface para as conexões. Isso possibilita que o usuário da classe se preocupe somente com sua utilização, ao invés de como realmente foi implementada (GAMMA et al., 2000). Com esta abstração do meio de comunicação e a forma que é realizada, as mensagens podem ser recebidas e enviadas de um forma genérica.

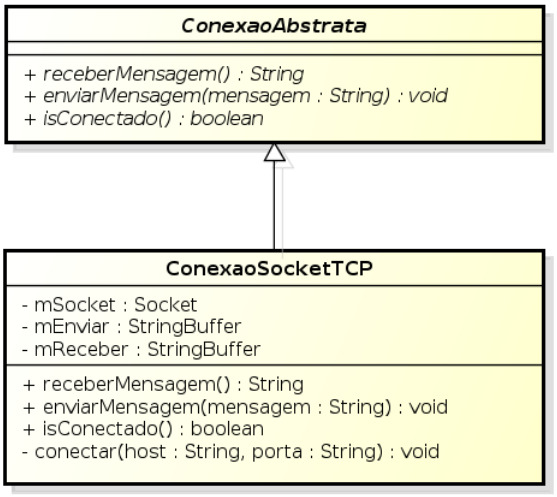


Figura 4.1: Diagrama da Conexão

O diagrama de conexão (figura 4.1) demonstra como encontra-se estruturada a conexão. A interface abstrata possui três funções genéricas que são utilizadas para o envio e recebimento de mensagens e checagem de conexão. Enquanto a classe concreta possui todos os métodos e atributos necessários para realizar uma conexão TCP.

Desta forma, caso haja uma modificação na forma de conexão do servidor só será necessária a modificação da classe concreta, minimizando o impacto nos demais módulos.

4.2.2 Atuador

O Modulo Atuador é resultado da fase de implementação da troca de mensagens e tem como finalidade a construção das mensagens que serão enviadas para o simulador. Essas mensagens enviadas são a forma de modificar o estado do simulador, sendo denominadas pelo SimSpark de atuadores. Estes atuadores devem seguir a mesma estrutura das mensagens recebidas do simulador, ou seja, em formato S-expressions (SEIBEL, 2005).

O SimSpark possui diversos atuadores que são classificados de atuadores gerais e atuadores da simulação de futebol. Para delimitação de escopo serão apresentados somente os atuadores utilizados neste trabalho. São eles:

- Atuadores Gerais

  Atuadores de Criação;

  Atuadores de Juntas.

- Atuadores da Simulação de Futebol

  Atuadores de Iniciação;

  Atuadores de Beam.

A primeira mensagem enviada ao conectar-se ao SimSpark deve ser a de criação. Este atuador informa ao simulador qual o agente simulado deve ser construído, de acordo com a descrição de um arquivo de configuração que é enviado como parâmetro.

Formato da Mensagem : (scene <nomeArquivo>)

Mensagem de Exemplo : (scene rsg/agent/nao/nao.rsg)

Após o agente simulado ser construido, ainda deve-se fazer mais configurações específicas da simulação. Isto é feito com os atuadores de iniciação. Este atuador registra qual o nome do time e número do agente que se conectou.

Formato da Mensagem : (init (unum <numeroJogador>)(teamname <nomeTime>))

Mensagem de Exemplo : (init (unum 1)(teamname Bahia3D))

Somente após esta configuração inicial o simulador passa a aceitar as demais mensagens e executá-las.

Os atuadores de juntas são mensagens que modificam a variação nos ângulos das juntas ou a velocidade dos motores. São compostas por dois parâmetros: O identificador da junta e a velocidade do motor em radiano por segundo.

Formato da Mensagem : (<nome><ax>)

Mensagem de Exemplo : (lae3 5.3)

Por último, os atuadores de beam permitem a um jogador posicionar-se sobre o campo antes do início do jogo. Possui três parâmetros: As coordenadas x e y do campo e o ângulo de rotação do jogador.

Formato da Mensagem : (beam <x><y><rot>)

Mensagem de Exemplo : (beam 10.0 -10.0 0.0)

Mais informações sobre os demais atuadores do simulador podem ser encontradas na página oficial do projeto SimSpark (SIMSPARK, 2011) ou no manual do usuário (BOEDECKER et al.,).

Para montagem destas mensagens foi criada uma classe genérica chamada Comando. Desta forma o módulo Atuador só necessita receber um comando e adicioná-lo a lista de comandos para ser enviado, abstraindo as estruturas dos atuadores. Além disso, mais atuadores podem ser adicionados sem a necessidade de grande mudanças no módulo Atuador. O diagrama 4.2 demonstra a estrutura entre a classe Atuador e as classes Comando.

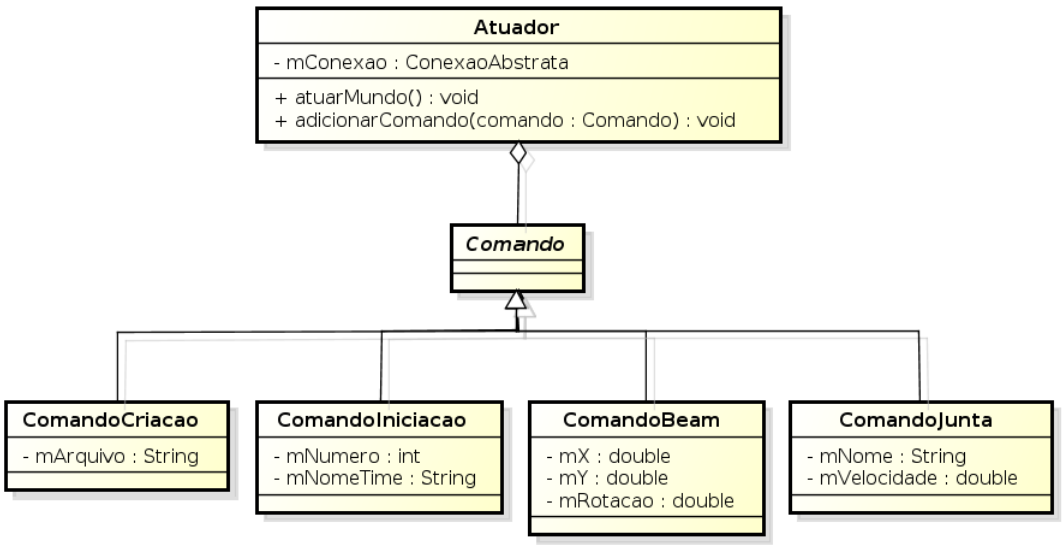


Figura 4.2: Diagrama do Atuador

Ao final de cada ciclo, o módulo Atuador, envia para o SimSpark os comandos que estejam na lista, limpando-a logo em seguida e ficando no aguardo de novos comandos. Isso é feito através do método atuarMundo(), que é chamado a cada fim do ciclo do IFAS3D.

4.2.3 Sensor

Resultado da fase de implementação da troca de mensagens, juntamente com o Atuador, o módulo Sensor é responsável pela extração das informações relevantes nas mensagens recebidas do servidor e por efetuar a atualização destas informações no modelo de

mundo. Este modulo é executado em uma *thread* para que a atualização das informações não seja prejudicada na interação com o usuário.

A *thread* tem a função de ficar no aguardo de uma nova mensagem do servidor; com a chegada desta nova mensagem é efetuado o *parsing* e por fim o armazenamento dos novos dados, iniciando-se um novo ciclo de aguardo e atualização.

Para efetuar o *parsing* da mensagem, primeiramente foi definida uma gramática formal (Apêndice A), essa gramática especifica as possibilidades da estrutura da mensagem (AHO et al., 1995). Com base nessa gramática, é verificada a cadeia de caracteres e se esta cadeia encontra-se devidamente estruturada.

Como resultado desta análise é gerada uma estrutura conhecida como árvore sintática abstrata (AHO et al., 1995) que é responsável pelo armazenamento dos dados extraídos da mensagem e por efetuar a atualização no ModeloMundo. Apesar do *parsing* ser efetuado na mensagem completa, somente alguns dados são utilizados pela ferramenta.

Analogamente ao atuador, o SimSpark classifica os sensores em sensores gerais e sensores da simulação de futebol. Abaixo apresentamos somente os utilizados neste trabalho:

- Sensores Gerais

  Sensor Giroscópio;

  Sensor Acelerômetro;

  Sensores de juntas;

  Sensor de Tempo de Simulação.

O sensor giroscópio, também chamado de *gyrorate*, é responsável por fornecer informações sobre a orientação do corpo. A mensagem é composta por um identificador e três ângulos de rotação, estes ângulos descrevem a orientação do corpo.

Formato da Mensagem : (GYR (n <nome>) (rt <x><y><rot>))

Mensagem de Exemplo : (GYR (n torso) (rt 0.01 0.07 0.46))

O sensor acelerômetro mede a aceleração do corpo em relação à queda livre. Ou seja, com o corpo em repouso em relação à superfície da Terra será indicado o valor da gravidade. A mensagem é composta por um identificador e três valores de aceleração.

Formato da Mensagem : (ACC (n <nome>) (a <x><y><rot>))

Mensagem de Exemplo : (ACC (n torso) (a 0.00 0.00 9.81))

O sensor de juntas define os ângulos em que se encontra cada junta do robô. Esta mensagem contém um identificador, o nome da junta e em que ângulo ela encontra-se naquele ciclo.

Formato da Mensagem : (HJ (n <nome>) (ax <ax>))

Mensagem de Exemplo : (HJ (n laj2) (ax 1.10))

Por último, temos os sensores de tempo de simulação, que indica quanto tempo, em segundos, de simulação já se passou.

Formato da Mensagem : (time (now <tempo>))

Mensagem de Exemplo : (time (now 8.50))

4.2.4 ModeloMundo e Main

ModeloMundo

Como resultado da fase de implementação do armazenamento das informações, o módulo ModeloMundo armazena todos os dados utilizados pelo programa, desde as posições atuais das juntas até o último comando enviado.

Para que não ocorresse perda ou duplicação de dados e só houvesse uma única instância do ModeloMundo, foi optado para o desenvolvimento deste módulo o padrão de projeto chamado *singleton*. Com isso é garantido a existência de uma única instância da classe, mantendo um ponto global de acesso ao objeto (GAMMA et al., 2000). Como consequência o módulo pode ser acessado por todos os outros módulos.

Main

O Main é o módulo central do programa, onde são instanciados os demais módulos. É a partir dele que são chamadas as principais funções dos demais módulos, com exceção do módulo Sensor que é executado em uma *thread* separada.

A fase de implementação da interface com o usuário teve como resultado a adição, neste módulo, das chamadas das bibliotecas gráficas do Qt para construção das janelas. Através destas janelas que ocorre a interação do usuário com a ferramenta.

4.3 Funcionalidades

O IFAS3D possui uma janela gráfica, como ilustrada na figura 4.3, onde são mostradas as informações mapeadas pela ferramenta. Dentre essas informações encontra-se o tempo de simulação do SimSpark, exibido dentro do box *Server Status*; os valores presentes no acelerômetro e no giroscópio, exibido no box *Internal Perceptor* nas linhas *Accelerometer* e *Gyrorate*, respectivamente; e as posições atuais das juntas, que encontram-se exibidas no box *Joint*, na coluna *Current Position*.

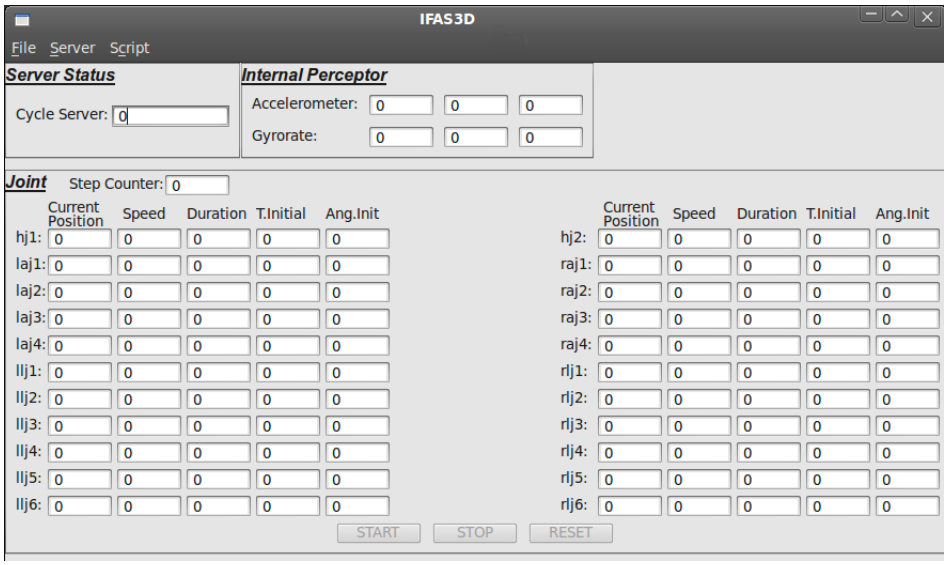


Figura 4.3: Tela do IFAS3D

Para que o usuário possa interagir e testar os movimentos do robô foram implementadas duas formas. A primeira é a possibilidade de interação com as juntas do robô em tempo real e a segunda é a possibilidade de utilizar scripts que contenham sequências de comandos para juntas.

A interação em tempo real é feita através de quatro parâmetros, que são velocidade, duração, tempo inicial e ângulo inicial que na figura 4.3 são representadas, respectivamente, pelas colunas *speed*, *duration*, *T.Initial* e *Ang. Init* dentro do box *Joint*.

O parâmetro velocidade especifica a velocidade de rotação da junta, a partir daquele ciclo. A duração indica o número de ciclos que a velocidade dada será aplicada na junta.

Já os parâmetros de tempo inicial, que indica o ciclo em que a junta deve iniciar o movimento, e ângulo inicial, que indica o ângulo em que a junta deve estar para iniciar o movimento, são parâmetros de inicialização e possuem precedência em relação ao parâmetro velocidade. Para que a esse seja aplicado é necessário, primeiramente, que os valores dos parâmetros de inicialização sejam atendidos.

Vale ressaltar que, ao se configurar o ângulo inicial diferente da posição atual da junta, a ferramenta só irá aplicar a velocidade indicada quando alcançar esse ângulo. Para que este parâmetro seja satisfeito o IFAS3D envia a velocidade de $1 rad/s$, até que a posição atual seja igual ao ângulo inicial.

Uma limitação da interação em tempo real é que quando é iniciada uma interação não se pode alterar as velocidades enviadas para as juntas até que a última junta tenha completado sua duração ou que todas sejam paradas.

Entretanto a utilização do modo de interação em tempo real é interessante para a compreensão do funcionamento das juntas e da física do ambiente simulado, mas não para se desenvolver movimentos mais complexos. Para esse intuito foi criada a interação por arquivos de lotes, chamados de scripts de controle de movimentos, que possibilita a criação de sequências de velocidades mais elaboradas.

4.3.1 Script de Controle de Movimentos

O script de controle de movimentos é um modo de enviar uma sequência de velocidades de forma automática, sem a necessidade de entrada de dados para cada nova sequência. Ele foi representado no formato XML e composto por uma sequência de passos, onde cada passo possui uma duração e uma lista de juntas com suas respectivas velocidades. Essa composição é feita por meio de tags e valores. As tags definem elementos e os valores os dados deste elemento ou outros elementos. Abaixo, temos um descritivo das tags utilizadas nos scripts de movimentos:

script: Define o início e o fim do arquivo.

step: Define cada passo da sequência. É composta por outros dois elementos: duration e joint.

duration: Define a duração de cada passo. Tem como valor um inteiro que indica a quantidade de ciclos do passo.

joint: Contém uma lista de elementos juntas ativas naquele passo.

junta: Contém, como tag, o nome da junta e como valor a velocidade, em rad/s, que será aplicada na junta.

Exemplo de scripts de movimento:

```
<script>
  <step>
    <duration>1</duration>
    <joint>
      <llj1>0.0356</llj1>
      <rlj1>0.0</rlj1>
      <llj2>-0.027</llj2>
      <rlj2>-0.049</rlj2>
      <llj3>-0.2</llj3>
      <rlj3>-0.0</rlj3>
      <llj4>0.0001</llj4>
      <rlj4>0.0</rlj4>
      <llj5>-1.0</llj5>
      <rlj5>-1.0</rlj5>
      <llj6>0.0272</llj6>
      <rlj6>0.0303</rlj6>
    </joint>
  </step>
</script>
```

4.4 Testes

Para realizarmos os testes da ferramenta foi utilizada a técnica de teste de caixa preta ou teste funcional. Esta técnica avalia o comportamento externo do software, sem considerar o comportamento interno (MYERS, 2004).

Primeiramente foram feitos os testes funcionais, por serem essenciais para o uso imediato da ferramenta e foram realizados em diferentes configurações, pois em experiências anteriores foi verificado desempenho diferente de um mesmo script em ambientes diferentes, ou seja, um script desenvolvido localmente não era executado de forma estável em ambiente distribuído.

Apesar de não ter sido realizado testes formais de usabilidade, por motivo do curto prazo disponível, a ferramenta foi utilizada por três novatos do grupo. Esses novatos demonstraram um rápido aprendizado da simulação em comparação aos antigos integrantes que não tiveram acesso a ferramenta. Além disso, a ferramenta foi extremamente utilizada durante o laboratório de Virtualização de Robôs ministrado na XI Escola Regional de Computação Bahia-Alagoas-Sergipe (ERBASE), onde se percebeu um avanço no aprendizado e na compreensão do ambiente pelos alunos. Entretanto, existe a ainda necessidade da realização destes testes para verificar o desempenho da interação entre o usuário e a ferramenta.

Nos testes do IFAS3D foram verificados quatro aspectos: (1) a conexão estabelecida com o servidor SimSpark; (2) a estabilidade da conexão com o servidor; (3) a execução de comandos através da interação em tempo real e (4) a execução de comandos através dos scripts de movimentos. Os testes foram realizados em três cenários diferentes. No cenário A foram executados o IFAS3D e o SimSpark, juntamente com o RoboViz, no mesmo computador. No cenário B foram executados o IFAS3D em uma máquina, enquanto o SimSpark e o RoboViz em outra. Já no cenário C cada software foi executado em computadores diferentes.

Para cada cenário foi gerado um formulário que encontra-se preenchido no Apêndice C e possui as seguintes informações:

ID : o id do teste;

Objetivo : o objetivo do teste;

Componentes : quais os componentes utilizados naquele teste;

Metodo de Teste : como foram utilizados estes componentes, em que ambiente e configuração;

Indicadores de Sucesso : quais os aspectos indicam que o teste obteve sucesso;

Try : número da tentativa daquele teste;

Resultado/Observações : quais os resultados do teste e se houve alguma anomalia;

Testadores : nome dos responsáveis pelo teste;

Data : data em que o teste foi efetuado.

Os processos foram repetidos vinte e cinco vezes para cada cenário, onde eram contabilizados as falhas e quedas na conexão. Nenhum dos casos de testes apresentou problemas ou instabilidades na conexão.

Os testes de execução de comandos através da interação em tempo real tiveram o intuito de verificar erros de execuções e inconsistência nas trocas de mensagens e se mostraram muitos satisfatórios, não apresentando erros ou anomalias nas trocas de mensagens.

Para os testes de execução de comandos através dos scripts de movimentos foi utilizado o script de abrir e fechar braços (Apêndice B). Em todos os testes efetuados o script foi executado como previsto. Além deste, outros scripts foram executados, como o script de andar, girar, chutar e defender, entretanto fazem parte do código do time Bahia3D e não podem ser mostrados publicamente.

Durante os testes notou-se um erro quando era efetuada a mudança do endereço da máquina em que o IFAS3D deveria conectar-se. Isso ocorria, pois na inicialização do programa o arquivo de configuração era lido e não ocorria uma nova leitura, caso o usuário modificasse o endereço. Este problema foi resolvido efetuando a leitura do endereço no início da conexão.

5 Conclusões

Este trabalho apresentou uma solução para o problema de construção de movimentos em robôs humanóides em ambiente simulado. Para isso, foi desenvolvida uma ferramenta estável, de fácil utilização, compatível com o simulador da subliga 3D e capaz de construir e executar os movimentos do robô simulado.

O IFAS3D mostrou-se uma ferramenta útil na elaboração e no desenvolvimento dos movimentos básicos, como andar, girar e chutar, além de aumentar a compreensão do funcionamento das articulações do robô simulado.

Em comparação à primeira abordagem, na qual eram necessárias mudanças diretas no código-fonte, o IFAS3D permitiu um desenvolvimento dos movimentos separado do código, de forma mais simples e intuitiva, com isso não é preciso recompilar a cada teste e não é necessário um alto conhecimento do código do agente pelo desenvolvedor do movimento, o que gera a não exclusão dos novatos. Além disso a ferramenta não possui a camada de inteligência fazendo com que os movimentos sejam executados imediatamente, sem a necessidade de uma situação indicada para a ação.

Por possuir uma estrutura com alta coesão e baixo acoplamento, o IFAS3D pode tornar-se uma ferramenta adaptável para qualquer robô simulado pelo SimSpark. Como exemplo para isso, podemos utilizar o módulo atuador. Se fosse implementado um novo modelo de robô, que ao invés de utilizar articulações utilizasse rodas, só seria necessária a criação das classes de comando desse novo robô, causando com isso pouco impacto no módulo.

Embora ainda exista a necessidade da elaboração dos testes de usabilidade, o IFAS3D já vem sendo utilizado desde janeiro de 2011 e auxilou na construção dos movimentos do time de simulação 3D, Bahia3D, que é desenvolvido pelo Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO, 2011). Além disto, a ferramenta foi utilizada durante o laboratório de Virtualização de Robôs, na XI Escola Regional de Computação Bahia-Alagoas-Sergipe e não houve dificuldade de utilização pelos usuários.

Glossário

middleware É a designação genérica utilizada para referir softwares que facilitam a integração entre sistemas legados ou desenvolvidos de forma não integrada. 19

parsing É o processo de análise de uma sequência de dados para determinar sua estrutura gramatical. 31

Ruby É uma linguagem de programação interpretada, de tipagem dinâmica e forte. 16

S-expressions Abreviação de expressões simbólicas. Elas são conhecidas por sua utilização na família Lisp de linguagens de programação, onde são utilizadas tanto para código, como para dados. 22, 28

scripts São linguagens de programação interpretadas executadas no interior de programas e/ou de outras linguagens de programação. 12, 16, 24, 33–37

TCP Transmission Control Protocol é um protocolo de nível da camada de transporte de entrega confiável. 27, 28

thread É uma forma de executar concorrentemente várias tarefas do mesmo processo. 31, 32

XML O *eXtensible Modeling Language* é uma linguagem de marcação de dados extensível, que provém um formato descritivo de dados estruturados que facilita declarações mais precisas do conteúdo. 34

APÊNDICE A – Gramática da Mensagem

A mensagem enviada pelo Simspark foi dividida em um conjunto de três objetos.

$$< mensagem >:=< tempo >< gamestate >< dados >$$

$< tempo >:=$ **(time (now** $< double >$ **))**
$< gamestate >:=$ **(GS** $< jogador >$ **(t** $< double >$ **) (pm** $< strings >$ **))**
$< jogador >:=$ **(unum** $< inteiro >$ **) (team** $< strings >$ **)**$|\epsilon$
$< dados >:=< audições >< percepções >$
$< percepções >:=< percepção >< percepções > |\epsilon$
$< audições >:=< audição >< audições > |\epsilon$
$< audição >:=$ **(hear** $< double >< direção >< strings >$ **)**
$< direção >:=< double > | < strings >$
$< percepção >:=$ **(** $< identificador >$ **(** $< conteudo >$ **)** $< conteudos >$ **)**
$< identificador >:= GYR|ACC|HJ|UJ|TCH|FRP|See|B|G1R|G2R|G1L|$
$G2L|F1R|F2R|F1L|F2L|P|head|rlowerarm|llowerarm|rfoot|lfoot$
$< conteudos >:=$ **(** $< conteudo >$ **)** $< conteudos > |\epsilon$
$< conteudo >:=< nome >< valor >< valores > | < percepções >$
$< nome >:= now|n|ax|a|rt|pol|team|id|c|f$
$< valores >:=< valor >< valores > |\epsilon$
$< valor >:=< strings > | < inteiro > | < double >$
$< inteiro >:=< digito >< digitos >$
$< double >:=< digito >< digitos >$ **.** $< digito >< digitos >$
$< digitos >:=< digito >< digitos > |\epsilon$
$< digito >:= 0|1|2|3|4|5|6|7|8|9$
$< strings >:=< letra >< string >$
$< string >:=< letra >< string > |\epsilon$
$< letra >:= a|A|b|B|c|C|d|D...x|X|y|Y|z|Z$

Exemplo de uma mensagem enviada pelo servidor:

Mensagem 3D: (time (now 8.50))(GS (t 0.00) (pm BeforeKickOff))(hear 8.50 self ComeOn!)(hear 8.50 0.00 ComeOn!)(GYR (n torso) (rt 0.00 0.00 0.00))(ACC (n torso) (a 0.00 0.00 9.81))(HJ (n hj1) (ax 0.08))(HJ (n hj2) (ax -0.00))(See (G2R (pol 12.01 -3.57 1.09)) (G1R (pol 12.02 3.26 1.21)) (F1R (pol 12.65 18.25 -2.55)) (F2R (pol 12.66 -18.47 -2.43)) (B (pol 6.02 -0.11 -4.80)) (P (team Bahia3D) (id 1) (rlowerarm (pol 0.18 -34.10 -22.78)) (llowerarm (pol 0.19 35.36 -22.30))) (P (team Bahia3D) (id 3) (head (pol 1.13 44.95 0.59)) (rlowerarm (pol 1.17 36.64 -2.96)) (llowerarm (pol 1.30 43.61 -2.63)) (rfoot (pol 1.23 42.17 -24.63)) (lfoot (pol 1.29 46.01 -23.28))) (P (team Bahia3D) (id 2) (head (pol 3.84 0.13 -0.20)) (rlowerarm (pol 3.84 -1.90 -3.25)) (llowerarm (pol 3.85 2.21 -2.89)) (rfoot (pol 3.86 -0.86 -7.71)) (lfoot (pol 3.86 0.67 -7.81))))(HJ (n raj1) (ax 0.01))(HJ (n raj2) (ax -0.00))(HJ (n raj3) (ax 0.00))(HJ (n raj4) (ax 0.00))(HJ (n laj1) (ax 0.01))(HJ (n laj2) (ax 0.00))(HJ (n laj3) (ax 0.00))(HJ (n laj4) (ax 0.00))(HJ (n rlj1) (ax -0.00))(HJ (n rlj2) (ax 0.03))(HJ (n rlj3) (ax -0.00))(HJ (n rlj4) (ax 0.00))(HJ (n rlj5) (ax -0.00))(FRP (n rf) (c 0.01 -0.08 -0.02) (f -0.04 -0.03 19.84))(HJ (n rlj6) (ax -0.00))(HJ (n llj1) (ax 0.02))(HJ (n llj2) (ax 0.01))(HJ (n llj3) (ax -0.01))(HJ (n llj4) (ax 0.00))(HJ (n llj5) (ax -0.00))(FRP (n lf) (c 0.01 0.04 -0.01) (f -0.06 0.06 25.35))(HJ (n llj6) (ax 0.00))

APÊNDICE B – Script de Movimento: Abrir e Fechar Braços

```
<script>
  <step>
    <duration>3</duration>
    <joint>
      <laj2>1.8700</laj2>
      <raj2>-1.8700</raj2>
    </joint>
  </step>
  <step>
    <duration>3</duration>
    <joint>
      <laj2>-1.8700</laj2>
      <raj2>1.8700</raj2>
    </joint>
  </step>
  <step>
    <duration>1</duration>
    <joint>
      <laj2>0.0000</laj2>
      <raj2>0.0000</raj2>
    </joint>
  </step>
</script>
```

APÊNDICE C – Registro dos Casos de Testes

CASO DE TESTE								
ID	OBJETIVO	COMPONENTES	METODO DE TESTE	INDICADORES DE SUCESSO	TRY	RESULTADO OBSERVAÇÕES	TESTADORES	DATA TESTES
CT001	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **AMD Athlon 64 x2 Dual Core 4200+, GeForce 7300 SE, 2GB ram DDR2.**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Hj2, 2, 10; OK	Adriano	09/09/11
					2	Laj1, -1, 20; OK	Adriano	09/09/11
					3	Rlj4, -1, 10; OK	Adriano	09/09/11
					4	Llj3, 2, 10; OK	Adriano	09/09/11
					5	Raj5, -3, 5; OK	Adriano	09/09/11
CT002	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **AMD Athlon 64 x2 Dual Core 4200+, GeForce 7300 SE, 2GB ram DDR2.** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	OK	Adriano	09/09/11
					2	OK	Adriano	09/09/11
					3	OK	Adriano	09/09/11
					4	OK	Adriano	09/09/11
					5	OK	Adriano	09/09/11

CASO DE TESTE								
ID	OBJETIVO	COMPONENTES	METODO DE TESTE	INDICADORES DE SUCESSO	TRY	RESULTADO OBSERVAÇÕES	TESTADORES	DATA TESTES
CT001	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **DESCRIÇÃO DA MAQUINA** **(quadcore q8200, 2.33Ghz Placa de video: GeForce 8400 GS)**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok (hj1,vel=5.time=3)	Ayran Cruz	09/09/11
					2	Ok (rlj1, vel=50,time=10)	Ayran Cruz	09/09/11
					3	Ok (llj1, vel=-50,dure=7)	Ayran Cruz	09/09/11
					4	Ok (rlj1, vel=70,dur=7)	Ayran Cruz	09/09/11
					5	Ok (rlj3, vel=47,time=10)	Ayran Cruz	09/09/11
CT002	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **DESCRIÇÃO DA MAQUINA A(quadcore q8200, 2.33Ghz Placa de video: GeForce 8400 GS)** **DESCRIÇÃO DA MAQUINA B (core2duo, 3.0Ghz, 4 RAM ddr3, placa de video: Radeon HD 4350)**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok (llj5, vel=65,time=15)	Ayran Cruz	09/09/11
					2	Ok (llj2, vel=50,time=10)	Ayran Cruz	09/09/11
					3	Ok (llj2, vel=-50,time=10)	Ayran Cruz	09/09/11
					4	Ok (raj4, vel=50,time=200)	Ayran Cruz	09/09/11
					5	Ok (llj5, vel=60,time=900)	Ayran Cruz	09/09/11
CT003	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **DESCRIÇÃO DA MAQUINA A** **(core2duo, 3.0Ghz, 4 RAM ddr3, placa de video: Radeon HD 4350)** **DESCRIÇÃO DA MAQUINA B** **(quadcore q8200, 2.33Ghz Placa de video: GeForce 8400 GS)** **DESCRIÇÃO DA MAQUINA C** **(Intel core i7, Nvidea gtx460, 4gb)**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok hj1 (5.5)	Ayran Cruz	09/09/11
					2	Ok laj1 (5,10)	Ayran Cruz	09/09/11
					3	Ok llj4(-6,5)	Ayran Cruz	09/09/11
					4	Ok rlj3 (4,12)	Ayran Cruz	09/09/11
					5	Ok rlj5(3,10)	Ayran Cruz	09/09/11
CT004	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **DESCRIÇÃO DA MAQUINA** **(quadcore q8200, 2.33Ghz Placa de video: GeForce 8400 GS)** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Ok	Ayran Cruz	09/09/11
					2	Ok	Ayran Cruz	09/09/11
					3	Ok	Ayran Cruz	09/09/11
					4	Ok	Ayran Cruz	09/09/11
					5	Ok	Ayran Cruz	09/09/11

CT005	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **DESCRIÇÃO DA MAQUINA A (quadcore q8200, 2.33Ghz Placa de video: GeForce 8400 GS)** **DESCRIÇÃO DA MAQUINA B (core2duo, 3.0Ghz, 4 RAM ddr3, placa de video: Radeon HD 4350)** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Ok	Ayran Cruz	09/09/11
					2	Ok	Ayran Cruz	09/09/11
					3	Ok	Ayran Cruz	09/09/11
					4	Ok	Ayran Cruz	09/09/11
					5	Ok	Ayran Cruz	09/09/11
CT006	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **DESCRIÇÃO DA MAQUINA A (core2duo, 3.0Ghz, 4 RAM ddr3, placa de video: Radeon HD 4350)** **DESCRIÇÃO DA MAQUINA B (quadcore q8200, 2.33Ghz Placa de video: GeForce 8400 GS)** **DESCRIÇÃO DA MAQUINA C (Intel core i7, Nvidea gtx460, 4gb)** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	ok	Ayran Cruz	09/09/11
					2	ok	Ayran Cruz	09/09/11
					3	ok	Ayran Cruz	09/09/11
					4	ok	Ayran Cruz	09/09/11
					5	ok	Ayran Cruz	09/09/11

CASO DE TESTE								
ID	OBJETIVO	COMPONENTES	METODO DE TESTE	INDICADORES DE SUCESSO	TRY	RESULTADO OBSERVAÇÕES	TESTADORES	DATA TESTES
CT001	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **DESCRIÇÃO DA MAQUINA = Core 2 duo E8400 3GHz 4gb ddr2 Radeon HD 4350** Juntas testadas (nome, velocidade e duração): **HJ1, 10, 2** **HJ2, 10, 2** **LLJ1, -2, 2** **RLJ1, -2, 2** **RLJ4, -5, 2**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok	Camila Laranjeira	09/09/11
					2	Ok	Camila Laranjeira	09/09/11
					3	Ok	Camila Laranjeira	09/09/11
					4	Ok	Camila Laranjeira	09/09/11
					5	Ok	Camila Laranjeira	09/09/11
CT002	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **DESCRIÇÃO DA MAQUINA A = Core 2 duo E8400 3GHz 4gb ddr2 Radeon HD 4350** **DESCRIÇÃO DA MAQUINA B = intel i7 870 2.93GHz 4gb ddr3 nVidia GT430** Juntas testadas (nome, velocidade e duração): **LAJ4, -5, 2** **RAJ4, 5, 2** **HJ1 ,5, 2** **LAJ2, 10, 2** **RAJ2, 10, 2**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok	Camila Laranjeira	09/09/11
					2	Ok	Camila Laranjeira	09/09/11
					3	OK	Camila Laranjeira	09/09/11
					4	Ok	Camila Laranjeira	09/09/11
					5	Ok	Camila Laranjeira	09/09/11
CT003	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **DESCRIÇÃO DA MAQUINA A = intel i7 870 2.93GHz 4gb ddr3 nVidia GT430** **DESCRIÇÃO DA MAQUINA B = intel i7 870 2.93GHz 4gb ddr3 nVidia GT430** **DESCRIÇÃO DA MAQUINA C = Core 2 duo E8400 3GHz 4gb ddr2 Radeon HD 4350** Juntas testadas (nome, velocidade e duração): **LLJ5, 10, 2** **RLJ5, 10, 2** **LLJ6, 5, 2** **RLJ6, 5, 2**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok	Camila Laranjeira	09/09/11
					2	Ok	Camila Laranjeira	09/09/11
					3	Ok	Camila Laranjeira	09/09/11
					4	Ok	Camila Laranjeira	09/09/11
					5	Ok	Camila Laranjeira	09/09/11
CT004	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **DESCRIÇÃO DA MAQUINA = Core 2 duo E8400 3GHz 4gb ddr2 Radeon HD 4350** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Ok	Camila Laranjeira	09/09/11
					2	Ok	Camila Laranjeira	09/09/11
					3	Ok	Camila Laranjeira	09/09/11
					4	Ok	Camila Laranjeira	09/09/11
					5	Ok	Camila Laranjeira	09/09/11
CT005	Verificar a estabilidade da conexão e a execução dos	IFAS3D SimSpark	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B:	- Sem instabilidade ou queda da conexão.	1	Ok	Camila Laranjeira	09/09/11
					2	Ok	Camila Laranjeira	09/09/11

	scripts de movimentos	RoboViz	**DESCRIÇÃO DA MAQUINA A = Core 2 duo E8400 3GHz 4gb ddr2 Radeon HD 4350** **DESCRIÇÃO DA MAQUINA B = intel i7 870 2.93GHz 4gb ddr3 nVidia GT430** Execução do script de abrir e fechar os braços.	- O script de movimento foi executado corretamente.	3	Ok	Camila Laranjeira	09/09/11
				4	Ok	Camila Laranjeira	09/09/11
				5	Ok	Camila Laranjeira	09/09/11
CT006	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **DESCRIÇÃO DA MAQUINA A = intel i7 870 2.93GHz 4gb ddr3 nVidia GT430** **DESCRIÇÃO DA MAQUINA B = intel i7 870 2.93GHz 4gb ddr3 nVidia GT430** **DESCRIÇÃO DA MAQUINA C = Core 2 duo E8400 3GHz 4gb ddr2 Radeon HD 4350** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Ok	Camila Laranjeira	09/09/11
				2	Ok	Camila Laranjeira	09/09/11
				3	Ok	Camila Laranjeira	09/09/11
				4	Ok	Camila Laranjeira	09/09/11
				5	Ok	Camila Laranjeira	09/09/11

CASO DE TESTE								
ID	OBJETIVO	COMPONENTES	METODO DE TESTE	INDICADORES DE SUCESSO	TRY	RESULTADO OBSERVAÇÕES	TESTADORES	DATA TESTES
CT001	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **Intel core 2 duo, Radeon HD 4350, 4gb ram ddr3.**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Hj1, 2, 2; OK	Emmanuel Argollo	08/09/11
					2	Laj4, -5, 10; OK	Emmanuel Argollo	08/09/11
					3	Llj4, -2, 5; OK	Emmanuel Argollo	08/09/11
					4	Rlj3, 4, 6; OK	Emmanuel Argollo	08/09/11
					5	Raj3, 6, 10; OK	Emmanuel Argollo	08/09/11
CT002	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **Intel core 2 duo, Radeon HD 4350, 4gb ram ddr3.** **Itel core i7, Nvidea gtx460, 4gb ddr3.**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Laj3, -5, 10; OK	Emmanuel Argollo	08/09/11
					2	Llj4, -3 6; OK	Emmanuel Argollo	08/09/11
					3	Rlj3 6, 20; OK	Emmanuel Argollo	08/09/11
					4	Raj2 -4, 10; OK	Emmanuel Argollo	08/09/11
					5	Raj1, 5, 12; OK	Emmanuel Argollo	08/09/11
CT003	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **Intel core 2 duo, Radeon HD 4350, 4gb ram ddr3.** **Intel core 2 quad, GeForce 8400 GS, 4gb ram ddr3.** **Itel core i7, Nvidea gtx460, 4gb ddr3.**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Hj1, 5, 5; OK	Emmanuel Argollo	08/09/11
					2	Laj1, 5, 10; OK	Emmanuel Argollo	08/09/11
					3	Llj4, -6, 5; OK	Emmanuel Argollo	08/09/11
					4	Rlj3, 4, 12; OK	Emmanuel Argollo	08/09/11
					5	Rlj5, 3, 10; OK	Emmanuel Argollo	08/09/11
CT004	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **Intel core 2 duo, Radeon HD 4350, 4gb ram ddr3.** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Ok	Emmanuel Argollo	08/09/11
					2	Ok	Emmanuel Argollo	08/09/11
					3	Ok	Emmanuel Argollo	08/09/11
					4	Ok	Emmanuel Argollo	08/09/11
					5	Ok	Emmanuel Argollo	08/09/11
CT005	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **Intel core 2 duo, Radeon HD 4350, 4gb ram ddr3.** **Itel core i7, Nvidea gtx460, 4gb ddr3.** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Ok	Emmanuel Argollo	08/09/11
					2	Ok	Emmanuel Argollo	08/09/11
					3	Ok	Emmanuel Argollo	08/09/11
					4	Ok	Emmanuel Argollo	08/09/11
					5	Ok	Emmanuel Argollo	08/09/11
CT006	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **Intel core 2 duo, Radeon HD 4350, 4gb ram ddr3.** **Intel core 2 quad, GeForce 8400 GS, 4gb ram ddr3.** **Itel core i7, Nvidea gtx460, 4gb ddr3.** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	ok	Emmanuel Argollo	08/09/11
					2	ok	Emmanuel Argollo	08/09/11
					3	ok	Emmanuel Argollo	08/09/11
					4	ok	Emmanuel Argollo	08/09/11
					5	ok	Emmanuel Argollo	08/09/11

CASO DE TESTE								
ID	OBJETIVO	COMPONENTES	METODO DE TESTE	INDICADORES DE SUCESSO	TRY	RESULTADO OBSERVAÇÕES	TESTADORES	DATA TESTES
CT001	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **Intel i7 870, 4Gb ddr3, nvidia gtx460**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok **hj1(5,10), hj2(5,5)**	Murilo Reis	08/09/11
					2	Ok **laj1(2,3), raj1(6,1)**	Murilo Reis	08/09/11
					3	Ok **laj2(4,7), raj2(1,10)**	Murilo Reis	08/09/11
					4	Ok **lj2(2,3), rlj1(4.6),**	Murilo Reis	08/09/11
					5	Ok llj5(2,3) , llj6(3,4), rlj4(5,4), rlj5(3,1)	Murilo Reis	08/09/11
CT002	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **Intel core 2 duo, radeon hd 4350, 4gb ddr3** **Intel i7 870, 4Gb ddr3, nvidia gtx460**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok laj3(-5,10)	Murilo Reis	09/09/11
					2	Ok llj4(-3,6)	Murilo Reis	09/09/11
					3	Ok llj3(6,20)	Murilo Reis	09/09/11
					4	Ok raj2(-4,10)	Murilo Reis	09/09/11
					5	Ok raj1(5,12)	Murilo Reis	09/09/11
CT003	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **Intel core 2 duo, radeon hd 4350, 4gb ddr3** **Intel core quad, 4 gb ddr3, Gforce 8400 GS** **Intel i7 870, 4Gb ddr3, nvidia gtx460**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Ok Hj1(5,5)	Murilo Reis	09/09/11
					2	Ok laj1(5,10)	Murilo Reis	09/09/11
					3	Ok llj4(-6,5)	Murilo Reis	09/09/11
					4	Ok rlj3(4,12)	Murilo Reis	09/09/11
					5	Ok rlj5 (3,10)	Murilo Reis	09/09/11
CT004	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **Intel i7 870, 4Gb ddr3, nvidia gtx460** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	ok	Murilo Reis	08/09/11
					2	ok	Murilo Reis	08/09/11
					3	ok	Murilo Reis	08/09/11
					4	ok	Murilo Reis	08/09/11
					5	ok	Murilo Reis	08/09/11
CT005	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: **Intel core 2 duo, radeon hd 4350, 4gb ddr3** **Intel i7 870, 4Gb ddr3, nvidia gtx460** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	ok	Murilo Reis	09/09/11
					2	ok	Murilo Reis	09/09/11
					3	ok	Murilo Reis	09/09/11
					4	ok	Murilo Reis	09/09/11
					5	ok	Murilo Reis	09/09/11
CT006	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **Intel core 2 duo, radeon hd 4350, 4gb ddr3** **Intel core quad, 4 gb ddr3, Gforce 8400 GS** **Intel i7 870, 4Gb ddr3, nvidia gtx460** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	ok	Murilo Reis	09/09/11
					2	ok	Murilo Reis	09/09/11
					3	ok	Murilo Reis	09/09/11
					4	ok	Murilo Reis	09/09/11
					5	ok	Murilo Reis	09/09/11

CASO DE TESTE								
ID	OBJETIVO	COMPONENTES	METODO DE TESTE	INDICADORES DE SUCESSO	TRY	RESULTADO OBSERVAÇÕES	TESTADORES	DATA TESTES
CT001	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **intel i7 870 2.93GHz** **4gb ddr3** **nVidia GT430** Juntas testadas (nome, velocidade e duração)	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	OK (hj1,2,2 ; hj2,2,2)	Rafael Factum	08/09/11
					2	OK (llj2,6,6 ; llj6 -6,6)	Rafael Factum	08/09/11
					3	OK (laj2,5,3 ; laj3,4,3)	Rafael Factum	08/09/11
					4	OK (rlj2,5,5)	Rafael Factum	08/09/11
					5	OK (llj3,6,3 ; llj4,3,3 ; llj5,-6,3)	Rafael Factum	08/09/11
CT002	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: Maquina A: **intel i7 870 2.93GHz** **4gb ddr3** **nVidia GT430** **Maquina B:** **Core 2 duo E8400 3GHz** **4gb ddr2** **Radeon HD 4350**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	Quando há uma mudança de IP é necessário fechar e abrir o programa novamente para o mesmo se conectar a esse novo IP.	Rafael Factum	09/09/11
					2	OK (llj1,2,2)	Rafael Factum	09/09/11
					3	OK (llj3,4,4 ; llj5,4,4)	Rafael Factum	09/09/11
					4	OK (llj4,-9,7)	Rafael Factum	09/09/11
					5	OK (laj3,5,5 ; laj4,5,5 ; raj3,5,5 ; raj4,5,5)	Rafael Factum	09/09/11
CT003	Verificar a estabilidade da conexão e a execução dos comandos para as juntas.	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **Maquina A:** **intel i7 870 2.93GHz** **4gb ddr3** **nVidia GT430** **Maquina B:** **Core 2 Quad 2.33GHz** **4gb ddr2** **Geforce 8400GS** **Maquina C:** **Core 2 duo E8400 3GHz** **4gb ddr2** **Radeon HD 4350**	- Sem instabilidade ou queda da conexão. - As juntas testadas foram movimentadas.	1	OK (ll6,6,6)	Rafael Factum	09/09/11
					2	OK (ll2,3,3 ; rlj2,3,3)	Rafael Factum	09/09/11
					3	OK (llj1,3,6)	Rafael Factum	09/09/11
					4	OK (hj1,6,3 ; hj2,6,12)	Rafael Factum	09/09/11
					5	OK (llj3,12,1)	Rafael Factum	09/09/11
CT004	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	Todos os componentes foram executados na mesma maquina: **intel i7 870 2.93GHz** **4gb ddr3** **nVidia GTX430** Execução do script de abrir e fechar os braços.	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	OK	Rafael Factum	09/09/11
					2	OK	Rafael Factum	09/09/11
					3	OK	Rafael Factum	09/09/11
					4	OK	Rafael Factum	09/09/11
					5	OK	Rafael Factum	09/09/11
CT005	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	IFAS3D SimSpark RoboViz	O IFAS3D foi executado na maquina A e o SimSpark e o RoboViz na maquina B: Maquina A: **intel i7 870 2.93GHz**	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	OK	Rafael Factum	09/09/11
					2	OK	Rafael Factum	09/09/11
					3	OK	Rafael Factum	09/09/11

		4gb ddr3 **nVidia GT430** **Maquina B:** **Core 2 duo E8400 3GHz** **4gb ddr2** **Radeon HD 4350** Execução do script de abrir e fechar os braços.		4	OK		Rafael Factum	09/09/11
				5	OK		Rafael Factum	09/09/11
CT006	Verificar a estabilidade da conexão e a execução dos scripts de movimentos	O IFAS3D foi executado na maquina A, o SimSpark na maquina B e o RoboViz na maquina C: **Maquina A:** **intel i7 870 2.93GHz** **4gb ddr3** **nVidia GT430** **Maquina B:** **Core 2 Quad 2.33GHz** **4gb ddr2** **Geforce 8400GS** **Maquina C:** **Core 2 duo E8400 3GHz** **4gb ddr2** **Radeon HD 4350** Execução do script de abrir e fechar os braços.	IFAS3D SimSpark RoboViz	- Sem instabilidade ou queda da conexão. - O script de movimento foi executado corretamente.	1	Quando há uma mudança de IP é necessário fechar e abrir o programa novamente para o mesmo se conectar a esse novo IP.	Rafael Factum	09/09/11
					2	OK	Rafael Factum	09/09/11
					3	OK	Rafael Factum	09/09/11
					4	OK	Rafael Factum	09/09/11
					5	OK	Rafael Factum	09/09/11

APÊNDICE D – IFAS3D: Uma Interface para o Simulador de Jogos de Futebol com Robôs Humanoides em 3D

IFAS3D: Uma Interface para o Simulador de Jogos de Futebol com Robôs Humanoides em 3D foi um artigo submetido e aprovado em 2011 no Workshop de Trabalhos de Iniciação Científica e de Graduação - WTICG-BASE. Este evento ocorre em paralelo com a Escola Regional de Computação Bahia-Alagoas-Sergipe - ERBASE

IFAS3D: Uma Interface para o Simulador de Jogos de Futebol com Robôs Humanoides em 3D *

Adailton de J. Cerqueira Jr., Diego G. Frías Suárez, Marco A. C. Simões, Josemar R. Souza

[1]Núcleo de Arquitetura de Computadores e Sistemas Operacionais (ACSO)
Universidade do Estado da Bahia (UNEB) – Salvador – BA – Brasil

{adailton.junior, diegofriass}@gmail.com, {msimoes, josemar}@uneb.br

Abstract. *This work presents the IFAS3D, an interface to aid understanding how the the simulator of RoboCup 3D simulation league's games, SimSpark, works. With such understanding, the application aims to help in creation of movements for the league's simulated agents.*

Resumo. *Este artigo apresenta o IFAS3D, uma interface para a compreensão do funcionamento do simulador dos jogos da liga de simulação 3D da RoboCup, o SimSpark. A partir desta compreensão a ferramenta pretende a auxiliar na construção dos movimentos do agente físico simulado pela subliga.*

1. Introdução

A RoboCup (*Robot World Cup Initiative*) é uma iniciativa internacional de educação e pesquisa que objetiva impulsionar as pesquisas na área de Inteligência Artificial e robótica propondo um problema real onde diversas tecnologias podem ser integradas. Dentro da RoboCup existem quatro categorias: Futebol de Robôs (*RoboCup Soccer*), Resgate de Vítimas de Desastres (*RoboCup Rescue*), Desafios Domésticos (*RoboCup @Home*) e RoboCup Junior [Kitano et al. 1997a]. No *RoboCup Soccer* estão englobadas as competições de futebol robótico: *Middle size league* (Liga de robôs de médio porte), *Small size league* (Liga de robôs pequenos), *Standard plataform league* (Liga de Plataforma padrão) , *Simulation league* (Liga de Simulação) e *Humanoid league* (Liga de robôs humanóides).

Na liga de simulação agentes autônomos jogam futebol em um campo virtual, através de um software simulador comum aos dois times em competição, o único que sabe todas as informações da partida. Ambos os times recebem do simulador a informação de mundo restringida sobre o estado do jogo. A liga de simulação é subdividida em 2D e 3D.

Este artigo descreve o desenvolvimento do IFAS3D, uma interface gráfica para auxiliar no entendimento do ambiente utilizado pela subliga de simulação 3D. Essa interface será descrita na seção 5. Para isso, a seção 2, explica a RoboCup e a subliga de simulação 3D. Na seção 3 o simulador oficial utilizado pela RoboCup na 3D é descrito. A seção 4 descreve o robô simulado. Por fim, a seção 7, apresenta conclusões e possíveis modificações para a ferramenta.

*Este projeto é parcialmente financiado pela UNEB, FAPESB e PICIN/UNEB

2. RoboCup e a subliga de simulação 3D

Uma das formas de fomentar e testar técnicas de Inteligência Artificial (IA) é através de desafios padrão. Em 1997, o Deep Blue, um computador, venceu o campeão mundial de xadrez, Garry Kasparov. Na mesma década, alguns pesquisadores visualizaram que o jogo de futebol fornecia um ótimo campo de desafios para a IA e a robótica inteligente, por apresentar um ambiente multiagentes competitivo e colaborativo, dinâmico, estocástico, de tempo real, parcialmente observável, sequencial e contínuo [Russel and Norvig 2002, Kitano et al. 1995].

Com este intento, surgiu a iniciativa RoboCup, para apoiar o desenvolvimento da Robótica Inteligente, fomentando a Inteligência Artificial, a Robótica e áreas relacionadas [Kitano et al. 1997a, Kitano et al. 1997b]. Na RoboCup, o futebol é usado como desafio padrão. As técnicas desenvolvidas neste contexto posteriormente podem servir de base para sistemas de gerenciamento de tráfego, de energia, para comunicação entre robôs autônomos etc. Para atingir estes objetivos, foi criada uma organização denominada *RoboCup Federation* que passou a organizar um evento anual - RoboCup - que reúne um simpósio científico e competições de robôs. A meta desta iniciativa é que, até 2050, um time de futebol de robôs humanóide completamente autônomos seja capaz de vencer o time campeão mundial da FIFA do momento [Kitano et al. 1997b]. Entretanto, o objetivo real é promover o desenvolvimento de tecnologias utilizáveis em problemas sociais e industriais [Fonseca et al. 2002].

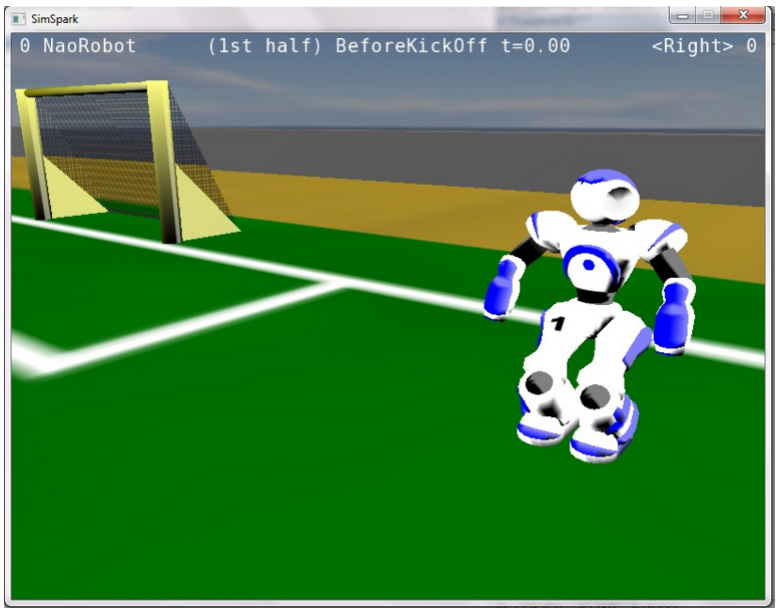


Figura 1. Imagem da liga de simulação 3D

Neste artigo iremos abordar apenas a liga de simulação 3D. Para mais detalhes sobre outras ligas, consultar [RoboCup 2011]. A subliga de simulação 3D é uma simulação de futebol de robôs na qual os agentes são simulados em um ambiente tridimensional (figura 1). Isto permite uma simulação realista da física e das regras de um jogo de futebol do mundo real. Nesta simulação, cada agente recebe percepções do ambiente e de si mesmo, através de sensores, e com isto deve tomar decisões e atuar sobre o mundo, através de atuadores.

Seguindo a taxonomia utilizada por [Russel and Norvig 2002], podemos classificar o ambiente da simulação 3D como: Parcialmente observável; estocástico; sequencial; dinâmico; discreto e multi-agente, diferindo de um ambiente real apenas no que se refere à discretude, já que até onde podemos perceber a realidade é contínua.

3. SimSpark

O SimSpark [Boedecker and Asada 2008] é um sistema de simulação multi-agente para agentes em ambientes tridimensionais, utilizado como o simulador oficial da subliga de Simulação 3D. Seu objetivo é proporcionar um alto grau de flexibilidade para a criação de novos tipos de simulações. Com isto, o SimSpark se torna uma ferramenta poderosa para diferentes pesquisas nas questões referentes a sistemas multi-agentes.

O simulador SimSpark foi baseado no Spark [Rollmann 2004], um simulador genérico de física multi-agente para ambientes tridimensionais. O Spark tem sua estrutura baseada em três componentes principais:

- Gestor de memória e objetos;
- Motor de simulação física;
- Motor de simulação.

O componente central do sistema é chamado Zeitgeist [Kogler 2003], e é responsável por fornecer acesso a todos os serviços do simulador. Basicamente existem três tipos de objetos gerenciados pelo Zeitgeist: objetos representando o cenário atual; objetos encapsulando a funcionalidade central do simulador e fábricas para a criação de novos objetos com base em texto. Essa característica é importante para a flexibilidade e extensibilidade, pois permite a alteração da configuração do simulador sem a necessidade de recompilar o código inteiro [Obst and Rollmann 2005].

O segundo componente principal do Spark é o motor de simulação física. Este motor é responsável pela detecção de colisões e atualização das posições e velocidades dos objetos simulados. A principal biblioteca utilizada neste componente é a ODE - Open Dynamics Engine [Smith 2004].

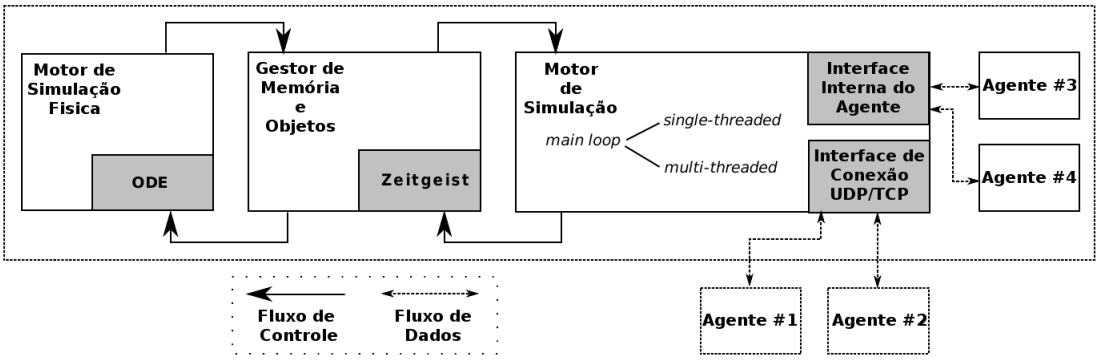


Figura 2. Fluxo de controle e dados entre os principais componentes do simulador. Modificado de [Boedecker and Asada 2008]

O terceiro componente é responsável pelo controle do *loop* principal do simulador e da interação entre o simulador e os agentes. Para este componente foi utilizado o *middleware* Spades [Riley and Riley 2003]. O Spades mantém o controle dos eventos entre

os agentes e o simulador. Na figura 2 é mostrada a interação entre estes componentes e seus fluxos de dados.

Mais informações sobre o simulador SimSpark podem ser encontradas na wiki do projeto [SimSpark 2011].

4. Agente Físico Simulado

Existe um esforço da Subliga de Simulação 3D para reproduzir os desafios enfrentados na construção de robôs reais. Por este motivo, vários modelos diferentes de robôs simulados já foram utilizados; os mais relevantes são a esfera oni-direcional, usada em 2004; o robô HOAP-2, usada em 2006 e o robô Nao, utilizado desde 2008.

Atualmente, o agente físico utilizado pela liga simula o utilizado como padrão na Standard Platform. O Nao é um robô humanoide autônomo e programável, desenvolvido pela empresa francesa Aldebaran Robotics [Aldebaran 2011]. Ele possui uma arquitetura bípede com 22 graus de liberdade (figura 3), o que lhe permite uma grande mobilidade.

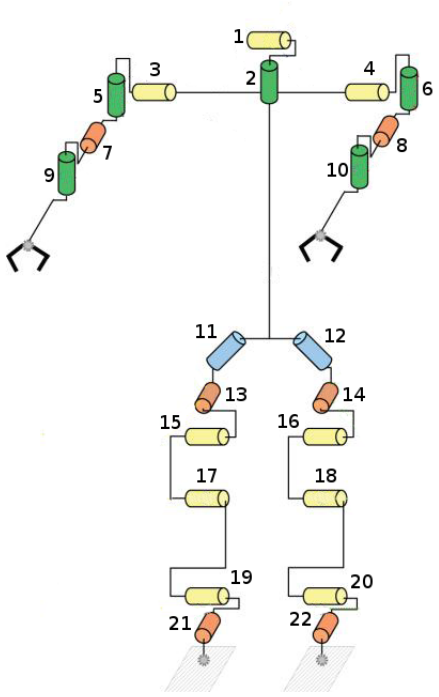


Figura 3. Graus de liberdade das juntas do robô Nao

Para simular os 22 graus de liberdade foram implementado 22 motores posicionados nas juntas do robô. Além das juntas, foram implementados também sensores para visão, audição, toque e aceleração. No total, foram implementados 5 tipos de sensores e 1 atuador, além das juntas.

4.1. Protocolo de comunicação

Segundo Stuart Russel e Peter Norvig [Russel and Norvig 2002], um agente é uma entidade que percebe o ambiente através de sensores e atua sobre ele através de atuadores. Partindo desta definição, podemos classificar os sensores como dispositivos ou

informações que possibilitam ao robô perceber o ambiente ao seu redor e os atuadores como dispositivos ou ações que possibilitam ao robô modificar o estado deste ambiente.

Para simular os atuadores e sensores o SimSpark utiliza um sistema de troca de mensagens com os agentes. Estas mensagens são baseadas em uma estrutura de dados *S-expressions*, abreviação de expressões simbólicas [Seibel 2005]. Elas são conhecidas por sua utilização na família Lisp de linguagens de programação, onde são utilizadas tanto para código como para dados.

Uma vantagem de usar *S-expressions* sobre outros formatos de dados é que ela proporciona uma análise fácil e sintaxe compacta, legíveis por seres humanos para fins de debug. Além disso, minimiza-se o tráfego de informação na rede.

As mensagens trocadas entre o cliente e o simulador estão no padrão ASCII de caracteres, onde cada caractere está codificado em um único byte. Cada mensagem também recebe um prefixo com o comprimento da carga útil da mesma. Este prefixo é um inteiro sem sinal, na notação *big endian*, onde os bits mais significativos são transferidos primeiro.

5. O Desenvolvimento do IFAS3D

O IFAS3D é uma interface gráfica que tem o intuito de auxiliar o entendimento dos sensores e atuadores do robô simulado pelo SimSpark para auxiliar, de forma empírica, na compreensão da física que atua sobre o agente.

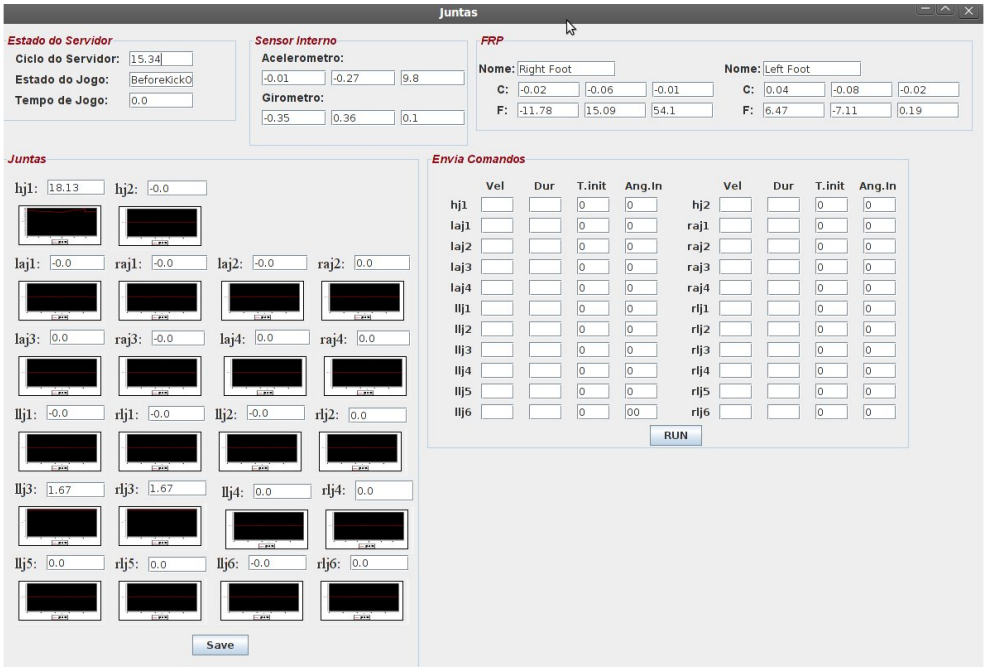


Figura 4. Esboço inicial da tela do IFAS3D

A ferramenta tem como funcionalidade estabelecer uma conexão com o simulador e efetuar trocas de mensagens. As mensagens enviadas pela ferramenta, são referentes às velocidades de movimentação das juntas do robô, que são indicadas pelo usuário. A partir disto pode ser analisado o funcionamento das juntas na simulação.

Outras informações que serão extraídas sobre o robô o valor do giroscópio e o valor do acelerômetro, além de informações relativas ao ambiente de simulação, como tempo de simulação, estado atual da simulação (*PlayMode*) e ciclo de atualização do simulador.

O IFAS3D é desenvolvido em C++, utilizando a *framework* Qt [Qt 2011]. Uma prova de conceito, feita em Java, indicou um grande consumo de memória. Devido a isso escolheu-se a linguagem C++. Na figura 4 é mostrado um esboço inicial da tela do IFAS3D desenvolvido em Java.

A estrutura do programa é composta por cinco módulos: Conexão, Sensor, Atuador, ModeloMundo e Main. O diagrama de classes simplificado da figura 5 ilustra a interação destes módulos.

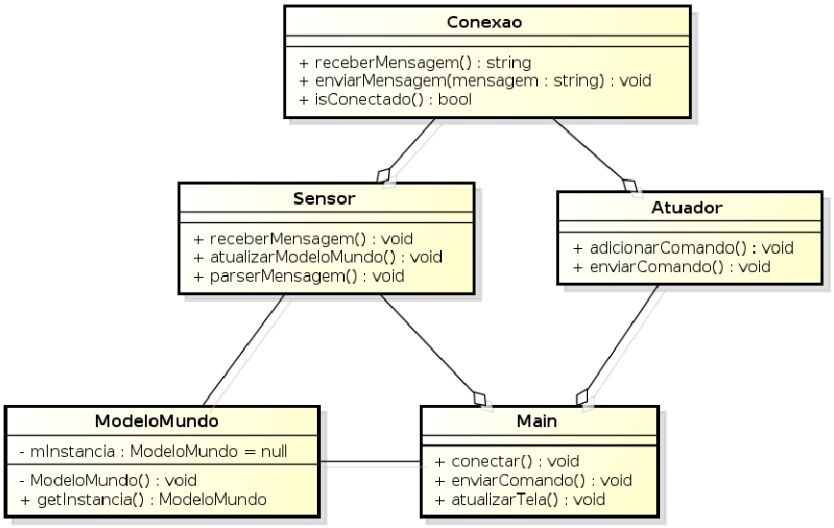


Figura 5. Diagrama de classes simplificado do IFAS3D

O módulo Conexão é responsável por efetuar a conexão com o simulador ou agente, além de administrar as trocas de mensagens. O módulo Atuador tem como finalidade a construção das mensagens que serão enviadas para o simulador.

Sensor é responsável pela extração das informações relevantes nas mensagens recebidas pela ferramenta, e por efetuar a atualização destas informações no modelo de mundo.

ModeloMundo tem como função o armazenamento de todos os dados utilizados pelo programa (e.g. posição atual das juntas, último comando enviado). Para o ModeloMundo foi utilizado um padrão de projeto chamado *singleton*. Este padrão garante a existência de uma única instância da classe, mantendo, com isso, um ponto global de acesso ao objeto [Gamma et al. 2000]. O motivo para isso é que o módulo ModeloMundo deve ser acessado por quase todos os outros módulos. Outros padrões de projetos estão sendo avaliados para utilização, como *abstract factory*, *builder* e *prototype*.

Por fim, o módulo Main é o módulo central do programa. Este módulo é responsável por gerenciar a interação do usuário com o programa, além da contrução das janelas gráficas.

6. Resultados

Como testes iniciais, foram atribuídas velocidades para juntas e verificado se estas eram executadas corretamente. Foram efetuados três movimentos:

- Movimento (A) - enviada uma velocidade para uma única junta do ombro (figura 6), fazendo com que o robô erguesse o braço esquerdo;
- Movimento (B) - enviada simultaneamente uma velocidade para as quatro juntas localizadas nos ombros (figura 7), fazendo com que o robô erguesse os braços;
- Movimento (C) - enviada uma velocidade para seis juntas simultaneamente (figura 8). As juntas utilizadas foram dos quadris, joelhos e pés, fazendo com que o robô se agachasse.

Todos os movimentos foram realizados com todas as juntas na posição inicial de 0º graus e em 50 ciclos. A tabela 1 mostra a relação entre as juntas, ângulos e velocidades utilizadas. As juntas estão numeradas de acordo com a figura 3.

Tabela 1. Tabela de relação de juntas, ângulos e velocidades utilizadas nos testes.

Relação Juntas, Ângulos e Velocidades			
Movimento	**Juntas**	**Posição Final(graus)**	**Velocidade(rad/s)**
A	4	90º	1.57
B	3	90	1.57
	4	90	1.57
	5	-45	-0.785
	6	45	0.785
C	15	90	1.57
	16	90	1.57
	17	-90	-1.57
	18	-90	-1.57
	19	45	0.785
	20	45	0.785

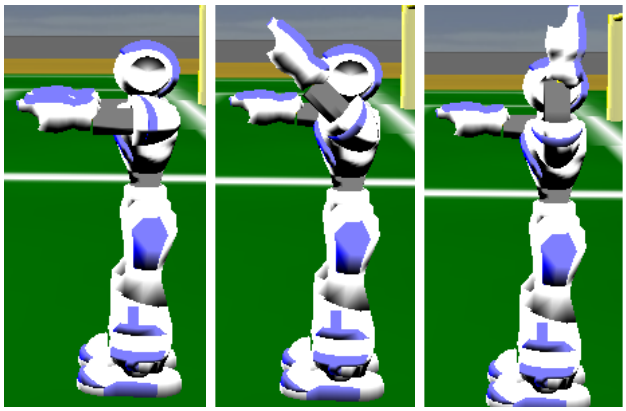


Figura 6. (A) Movimento de levantar o braço esquerdo do robô utilizando o IFAS3D

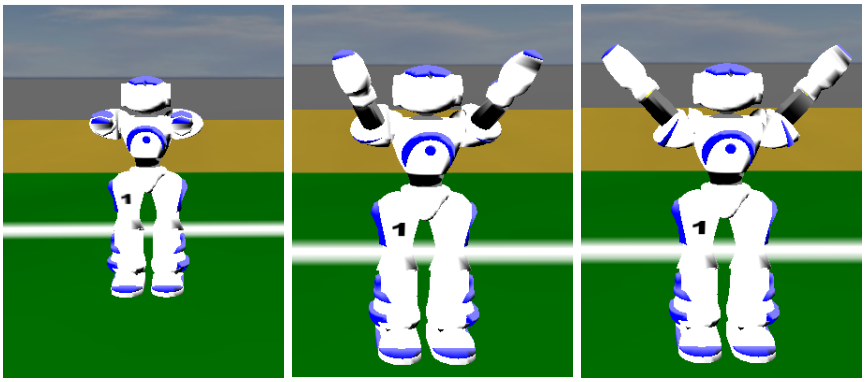


Figura 7. (B) Movimento de abrir os braços do robô utilizando o IFAS3D

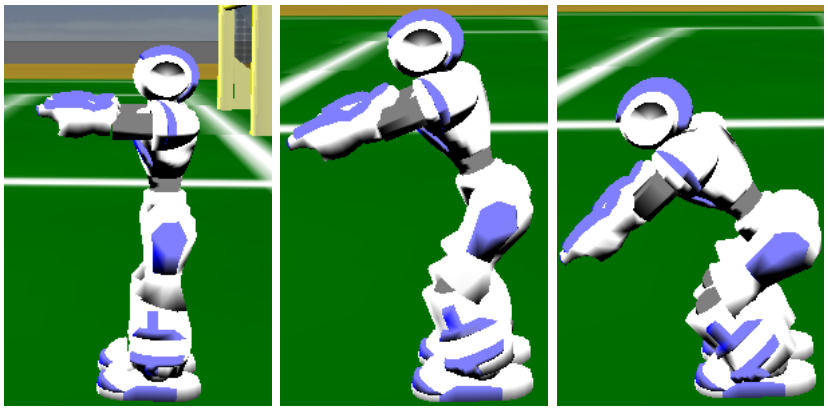


Figura 8. (C) Movimento de agachar utilizando o IFAS3D

Utilizando a velocidade enviada e a duração do movimento foi calculado o posicionamento previsto das juntas. Com isso, foi possível verificar uma diferença de, no máximo, 2º graus do posicionamento previsto para o posicionamento real das juntas. Essa diferença deve ser reduzida para que a ferramenta apresente mais consistência nas informações mapeadas.

7. Conclusões e Trabalhos Futuros

Este artigo apresentou o IFAS3D, uma ferramenta gráfica que auxilia na compreensão do ambiente de simulação da subliga de simulação 3D. Também trouxe um descritivo sucinto deste ambiente.

Ainda em fase de desenvolvimento, o IFAS3D tem se mostrado promissor no auxílio do entendimento da simulação realizada pelo SimSpark. Nos testes iniciais, foi possível detectar a existência do fator atrito no pé do robô simulado. Com isso, acreditamos que a ferramenta será de grande utilidade para o desenvolvimento do time Bahia3D.

Como trabalho futuro, está sendo verificado se a diferença entre o posicionamento previsto e o posicionamento real das juntas é causada por uma falha de sincronismo entre a ferramenta e o simulador. Também será desenvolvido um protocolo para comunicação com um agente. Este protocolo será útil para visualizar, em tempo real, o funcionamento de um agente no ambiente. Além disto, serão mapeadas mais informações do agente, como posicionamento e sensor de visão.

Referências

Aldebaran (2011). Aldebaran robotics.

Boedecker, J. and Asada, M. (2008). Simspark concepts and application in the robocup 3d soccer simulation league. In *Proceedings of the SIMPAR-2008*.

Fonseca, J., Pereira, M., and Guerra, R. S. (2002). Agentes e inteligência artificial distribuída.

Gamma, E., Helm, R., Johnson, R., Vlissides, J., and Salgado, L. (2000). *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*, volume 1. Bookman.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., and Osawa, E. (1997a). Robocup: The robot world cup initiative. In *AGENTS '97: Proceedings of the first international conference on Autonomous agents*.

Kitano, H., Asada, M., Kuniyoshi, Y., Noda, I., Osawa, E., and Matsubara, H. (1997b). Robocup: A challenge problem for ai. *AI magazine*, 1(18):13.

Kitano, H., Tambe, M., Stone, P., Veloso, M., Noda, I., OSAWA, E., and ASADA, M. (1995). The robocup synthetic agents challenge. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Kogler, M. (2003). Simulation and visualization of agents in 3d environments. Technical report, Koblenz-Landau University.

Obst, O. and Rollmann, M. (2005). Spark - a generic simulator for physical multiagent simulations. *Computer Systems Science and Engineering*, 20:347 – 356.

Qt (2011). Qt - a cross-platform application and ui framework - pÃ¡gina oficial.

Riley, P. and Riley, G. (2003). Spades - a distributed agent simulation environment with software-in-the-loop execution. In *Winter Simulation Conference Proceedings*.

RoboCup (2011). Robocup - página oficial da organização.

Rollmann, M. (2004). Spark - a generic simulator. Master's thesis, Koblenz-Landau University.

Russel, S. and Norvig, P. (2002). *Artificial Intelligence*. Prentice-Hall, 2 edition.

Seibel, P. (2005). *Practical Common Lisp*. Apress.

SimSpark (2011). Simspark wiki.

Smith, R. (2004). *Open Dynamics Engine (ODE) User Guide*.

Referências Bibliográficas

ACSO. *ACSO - Núcleo de Arquitetura de Computadores e Sistemas Operacionais*. Fevereiro 2011. Disponível em: <http://www.acso.uneb.br>.

AHO, A. et al. *Compiladores - Princípios, Técnicas e Ferramentas*. 1 ed. ed. [S.l.: s.n.], 1995.

ALDEBARAN. *Aldebaran Robotics*. Fevereiro 2011. Disponível em: <http://www.aldebaran-robotics.com/>.

BOEDECKER, J.; ASADA, M. Simspark – concepts and application in the robocup 3d soccer simulation league. In: *Proceedings of the SIMPAR-2008*. [S.l.: s.n.], 2008.

BOEDECKER, J. et al. *SimSpark User's Manual*. [S.l.], August.

FILHO, J. G. da S. *Aplicação de Metaprogramação e Padrões de Projeto na Construção de um Servidor para Futebol de Robôs usando Realidade Mista*. Dissertação (Mestrado) — Universidade do Estado da Bahia, 2009.

FONSECA, J.; PEREIRA, M.; GUERRA, R. S. Agentes e inteligência artificial distribuída. 2002.

GAMMA, E. et al. *Padrões de Projeto: soluções reutilizáveis de software orientado a objetos*. [S.l.]: Bookman, 2000.

KITANO, H. et al. Robocup: The robot world cup initiative. In: *AGENTS '97: Proceedings of the first international conference on Autonomous agents*. [S.l.: s.n.], 1997.

KOGLER, M. *Simulation and Visualization of Agents in 3D Environments*. [S.l.], 2003.

MYERS, G. J. *The art of software testing*. New York: John Wiley & Sons, 2004.

OBST, O.; ROLLMANN, M. Spark - a generic simulator for physical multiagent simulations. *Computer Systems Science and Engineering*, v. 20, p. 347 – 356, 2005.

QT. *Qt - A cross-platform application and UI framework - Página oficial*. Fevereiro 2011. Disponível em: <http://qt.nokia.com/>.

RILEY, P.; RILEY, G. Spades - a distributed agent simulation environment with software-in-the-loop execution. In: *Winter Simulation Conference Proceedings*. [S.l.: s.n.], 2003.

ROBOCUP. *RoboCup - Página oficial da organização*. Fevereiro 2011. Disponível em: <http://www.robocup.org/>.

ROBOTICS, A. *Nao, The new Robocup standard league platform*. [S.l.], 2008.

ROLLMANN, M. *Spark - a generic simulator*. Dissertação (Mestrado) — Koblenz-Landau University, 2004.

RUBY. *Ruby - A Programmer's Best Friend*. Agosto 2011. Disponível em: <http://www.ruby-lang.org/>.

RUSSEL, S.; NORVIG, P. *Artificial Intelligence*. 2. ed. [S.l.]: Prentice-Hall, 2002. 1132 p.

SEIBEL, P. *Practical Common Lisp*. [S.l.]: Apress, 2005.

SIMSPARK. *SimSpark Wiki*. Fevereiro 2011. Disponível em: <http://simspark.sourceforge.net/wiki/>.

SMITH, R. *Open Dynamics Engine (ODE) User Guide*. [S.l.], 2006.

SMITH, R. *Open Dynamics Engine*. Agosto 2011. Disponível em: <http://opende.sourceforge.net/wiki>.

STOECKER, J.; VISSER, U. Roboviz: Programmable visualization for simulated soccer. 2011.