VirBot4u: Um Simulador de Robô usando X3D

Alessandro Hoss[1], Marcelo da Silva Hounsell[1] e André Bittencourt Leal[2]

[1]Departamento de Ciência da Computação (DCC) – Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

[2]Departamento de Engenharia Elétrica (DEE) – Universidade do Estado de Santa Catarina (UDESC) – Joinville, SC – Brasil

alehoss@gmail.com, {marcelo,leal}@joinville.udesc.br

Abstract. *This paper presents a 5 degrees of freedom Robot Simulator called VirBot4u. It was developed using X3D technology as an API to the 3D environment which allows user interactions through an interface written in Java. The X3D specification is an upgrade of the ISO VRML97 and has the advantage of being extensible, flexible and open standard. The VirBot4u allows users to interact with the robot´s anatomy using buttons, keyboard hot keys as well as via mouse on the 3D graphics representation of the robot. The simulator allows the definition of reference points, creation of objects, and their manipulation thanks to a programming functionality that resembles the real robot´s own programming language.*

Resumo. *Este artigo apresenta o Simulador de Robô Articulado de 5 graus de liberdade VirBot4u, desenvolvido utilizando a tecnologia X3D, que permite a interação do usuário através de uma interface desenvolvida em Java, como API para o ambiente virtual tridimensional. O X3D é uma atualização da especificação ISO VRML97, que tem como características a extensibilidade e flexibilidade, além de ser um padrão aberto. O Simulador denominado VirBot4u permite interagir com a anatomia do robô através de botões, teclado ou diretamente no modelo gráfico usando o mouse. Também permite a especificação de pontos de referencia, criação de objetos e a movimentação destes através de linguagem de programação semelhante à do robô real.*

1. Introdução

Um robô é definido, segundo a Norma ISO 10218 [1992, *apud* Romano 2002], como "uma máquina manipuladora, com vários graus de liberdade, controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial". O número de graus de liberdade (DOF – *Degrees of Freedom*) de um robô corresponde à quantidade de eixos do mesmo.

Robôs manipuladores podem ter de 3 a 6 DOF, geralmente. Se possuírem mais de seis juntas, as demais serão as juntas redundantes, com função de, por exemplo, desviar de obstáculos [Schilling, 1990]. Os graus podem estar associados a juntas de **orientação**, que definem a guinada (*yaw*), arfada (*pitch*) e rolagem (*roll*) da garra/ferramenta acoplada ao robô; ou juntas de **posicionamento**, que movimentam a porção física do corpo do robô e determinam a posição do atuador final [Spong, Vidyasagar, 1989].

Uma das vantagens de um simulador de robô é a sua utilização no desenvolvimento da programação *off-line* (programas elaborados em um computador e carregados posteriormente no robô, sem que o mesmo seja retirado das funções de produção e ocupado com a tarefa de programação), permitindo que sejam feitos testes dos programas antes de executá-los no robô real [Redel e Hounsell 2004].

O uso de simuladores é vantajoso em se tratando de ambientes que podem trazer algum tipo de risco para o operador e/ou desgaste do equipamento, bem como evita as preparações requeridas para o uso do equipamento real. Além disso, se o simulador for integrado com CAD (*Computer-Aided Design*) / CAM (*Computer-Aided Manufacturing*), é possível detectar riscos de colisões do robô com obstáculos ou com partes da própria peça manipulada que podem estar no seu envelope de trabalho (que caracteriza todos os pontos que o robô consegue alcançar a partir de sua base fixa [Groover 1988]), e assim evitar danos mútuos. Por exemplo, pode ser simulada uma célula de manufatura utilizando um robô para reposicionar uma determinada peça, desviando dos obstáculos existentes no ambiente real desde que estes façam parte do ambiente virtual.

Outra vantagem de um simulador é a possibilidade de seu uso para o ensino, pois permite que cada aluno se familiarize com o robô de forma individual, mesmo sem a existência física do mesmo, fazendo com que estes alunos adquiram os conhecimentos básicos necessários antes de utilizarem o robô real.

Sendo assim, este trabalho apresenta o desenvolvimento de um simulador de robô articulado com 5 DOF, de código livre e aberto, que permite várias formas de interação com a cinemática do robô, além de dispor de uma interface para criação de objetos que podem ser manipulados posteriormente.

O simulador apresentado neste trabalho baseia-se no robô ER-4pc, que possui 5 DOF, com três deles para movimentação do braço antropomórfico do robô e dois para movimentação de uma garra do tipo pinça de dois dedos que se abre e fecha.

Para tanto foram, utilizadas as tecnologias Java (2009) e X3D (*Extensible 3D*) [Web3D 2009] de forma conjunta, sendo que a modelagem do robô e a manipulação deste no ambiente virtual são realizadas pelo X3D, enquanto a interface para interação com a cena, o posicionamento e a programação do robô foram desenvolvidos utilizando-se o Java. A integração entre estas tecnologias se dá pela API SAI (*Scene Access Interface*) [Geroimenko 2005] fazendo com que se tenha a percepção que o X3D como um todo é uma API gráfica a disposição da programação Java.

2. Simuladores de Robôs

Abaixo serão apresentados alguns simuladores de robô e ao final será apresentada uma tabela comparativa entre eles considerando as suas principais funcionalidades.

2.1. Ciros Studio

Este simulador permite a criação de cenários com a utilização de vários itens e tipos de robô que acompanham o *software* [Ciros, 2009]. Para interagir com o robô o usuário pode optar entre a forma incremental, movimentando-o através de botões em um dos elos ou em um dos eixos, e a forma absoluta, onde o usuário define um ponto com valores decimais específicos para as juntas (CD) ou para cada um dos eixos (CI), e o

robô assume aquela determinada posição. O ambiente de simulação apenas permite a movimentação da câmera, não permitindo que o usuário movimente o robô interagindo diretamente sobre a geometria deste.

Este *software* é disponibilizado gratuitamente em uma versão para testes, porém esta apresenta algumas limitações, tais como:

- Fica ativa durante 30 dias após sua instalação no computador,
- Só pode ser utilizada em sessões intercaladas de 60 minutos,
- Não permite que programas sejam salvos, e;
- A simulação dura no máximo 180 segundos.

2.2. GuiLab

Silva, Lima e Lucena (2008) desenvolveram um sistema telerobótico baseado na internet utilizando Java com X3D. Este sistema permite que o usuário controle um robô real de forma remota, e visualize o resultado em um ambiente virtual. Este robô é um Plotter Industrial de 3 DOF. Com o sistema telerobótico, é possível enviar comandos para que o robô desenhe figuras simples em uma área de 40 cm², como por exemplo, retas, retângulos e círculos, ou um arranjo destas, com a utilização do *mouse*.

O *software* possui um painel onde o usuário pode fazer a programação *off-line* do robô, traçando as linhas para criar a imagem desejada e, após enviar isto para o robô real (que esta a vários quilômetros de distância), o usuário consegue acompanhar a execução da tarefa através do ambiente virtual disponível.

Este sistema utiliza o Xj3D *Browser* [Xj3D, 2009] para visualização do ambiente tridimensional, feito em X3D, para substituir o retorno em vídeo do ambiente real, diminuindo assim o tempo de resposta do mesmo e a largura de banda requerida para comunicação entre aplicação cliente e servidor (local e remota).

2.3. DLR

No DLR [Rohrmeier, 2000], desenvolvido originalmente para possibilitar a tele manipulação do robô real via *web*, existem diversas maneiras para manipulação. O usuário pode movimentá-los com o *mouse* diretamente em cada elo, através de botões específicos para cada uma das juntas do robô, ou ainda realizar movimentos em apenas um dos eixos (x, y ou z) através de representações de planos (XY, XZ e YZ) existentes na interface.

Outro detalhe neste simulador é a existência de duas esferas na ponta do atuador final (uma dentro da outra, distintas por intensidade de cor). A esfera mais interior (menor) serve para fixar a **posição** do atuador final e, de acordo com os movimentos realizados nela, assume diferentes ângulos em cada uma das juntas para atender a nova configuração, enquanto que a esfera maior tem como função fixar a **orientação** do atuador final, independentemente dos movimentos realizados nas outras juntas.

O código fonte deste simulador é aberto e está disponível na *web*, sendo capaz de simular 3 tipos de robôs, que são: KUKA KR-6, PUMA 560 e MANUTEC, todos com 6 DOF. Como ele foi desenvolvido totalmente em VRML, é necessário que o usuário instale um *plugin* para o navegador em que deseja visualizá-lo, numa versão específica de *browser* e com uma máquina virtual java (JVM) específica (todas estas

são limitações inerentes da tecnologia VRML, de 1998, que se mostram muito restritivas porque esta tecnologia ficou estagnada no seu desenvolvimento).

Apesar da manipulação eficiente e variada, este simulador não dispõe de funcionalidades como detecção de colisão, nem de criação de objetos para manipulação.

2.4. Simulador do LARVA

O Simulador do LARVA (2009), Laboratório de Realidade Virtual Aplicada, apresenta um ambiente de programação para rotinas simples (abrir/fechar garra e realizar movimentos entre dois pontos previamente salvos), gerar objetos em posicionamento aleatório e, manipular o posicionamento do atuador final através da cinemática direta (pela definição dos ângulos relativos a cada uma das juntas), ou ainda pela movimentação dos elos diretamente na interface de simulação 3D. Porém, este simulador foi modelado em VRML e a interface foi desenvolvida utilizando-se *applets* Java para um navegador *web*.

Por ser uma tecnologia cuja última revisão ocorreu em 1997, o VRML possui pouco suporte atualmente, exigindo configurações específicas, como desabilitar a *Java Virtual Machine* (JVM) e versões antigas de *softwares*, como por exemplo, a JVM da Microsoft, para que as *applets* consigam ser carregadas e funcionem de maneira correta com a cena em VRML, além de poder ser executado apenas no *browser* Internet Explorer, versão 4 ou superior. Mais uma vez, restrições específicas da tecnologia.

Este simulador também tem uma configuração de interface fixa, onde a área dedicada ao ambiente 3D compete em espaço com várias outras partes da tela, cujas funcionalidades podem nem estar sendo utilizadas.

2.5. Análise dos Simuladores

Analisando os simuladores apresentados, pode-se perceber que existem diversas funcionalidades que alguns possuem e outros não. Uma comparação destes simuladores de robô é apresentada na Tabela 1 (onde ✓ significa que o simulador contém o recurso, NA para “não aplicável” e, ✗ quando não possui o recurso em questão), que contém as seguintes linhas:

- Cinemática Direta: Movimentação através dos ângulos das juntas, podendo esta ser feita na forma incremental (gradativamente), com a utilização de atalhos no teclado (“*Hotkey*”) ou por botões (via *mouse*) na interface, ou ainda utilizando valores absolutos (valor exato do ângulo relativo à determinada junta);
- Cinemática Inversa: Movimentação através do sistema de coordenadas cartesianas, também podendo ser feita por botões ou *hotkeys* incrementais como na forma anterior, ou por valores absolutos, sendo que nesta é dada a posição desejada do atuador final em relação aos eixos X, Y e Z;
- Movimentação na Interface 3D: diretamente na interface de simulação, quando os simuladores permitem que o usuário interaja com o *mouse* diretamente na estrutura geométrica do robô;
- Orientação fixa da garra: consiste em manter a orientação do atuador final fixa quando se movimentam as juntas de posicionamento do robô;

- Formas de abordagem pré-definidas: consiste em permitir a escolha de orientações pré-definidas para o atuador final durante as manipulações de posicionamento do robô;
- Criação de objetos: permite que sejam criados objetos virtuais na cena;
- Ambiente de programação: consiste em disponibilizar ao usuário um ambiente no qual ele possa criar uma lista de comandos (movimentação entre pontos, abrir garra, etc.) a serem executados pelo robô;
- *Freeware*: ter distribuição gratuita;
- *Open Source*: disponibilizar o código fonte do *software*.

A última coluna apresenta as características relativas ao simulaodor apresentado neste artigo, o VirBot4u, que serão detalhadas a seguir. Deve-se notar na Tabela 1 que o VirBot4u foi projetado para atender a todos os requisitos identificados acima, presentes de forma dispersa nos simuladores apresentados.

Tabela 1: Comparação entre os simuladores

Funcionalidade	Simulador				
	Ciros	**GuiLab**	**DLR**	**LARVA**	**VirBot4u**
Cinemática Direta	✓	NA	✓	✓	✓
Cinemática Inversa	✓	NA	✓	×	✓
Atalhos para manipulação (*Hotkeys*)	×	×	×	×	✓
Movimentação na interface 3D	×	×	✓	✓	✓
Orientação fixa da garra	×	✓	✓	×	✓
Formas de Abordagem pré-definidas	×	×	×	×	✓
Criação de Objetos	✓	×	×	✓	✓
Ambiente de Programação	✓	✓	×	✓	✓
Freeware	×	✓	✓	✓	✓
Open Source	×	✓	✓	✓	✓

3. O VirBot4u

VirBot4u é uma sigla para "***Vir****tual Ro****bot*** *for Yo****u***", que teve sua denominação inspirada na do próprio robô o qual este *software* simula: o ER-4pc, cujo significado é "*Eshed Robotec for Personal Computer*".

A Figura 1 mostra uma imagem do VirBot4u. Pode-se observar que é uma aplicação stand-alone, com um *Menu* superior a partir do qual são abertas janelas (flutuantes) para interação com a cena, e se apresenta tanto no idioma Inglês quanto Português.

O quadro mostrado na parede aos fundos da Figura 1 contém o nome do simulador, juntamente com o logotipo do LARVA (2009), uma vez que o código apresentado aqui se baseia no código livre e aberto do "Simulador do LARVA" apresentado no item 2.4 do capítulo anterior, cujo código está disponível livre e gratuitamente na internet (porém em VRML).

No canto superior direito da Figura 1 pode-se observar uma janela de parâmetros aberta, onde se pode configurar, por exemplo, a presença ou não de indicativos dos eixos coordenados em cores na cena 3D, mostrar explicitamente a posição de referencia no ponto extremo da pinça (chamado de PFP, Ponto Final da Pinça), na forma de uma

pequena esfera transparente e se é para ser efetuado o cálculo de colisão entre os objetos virtuais da cena do simulador.

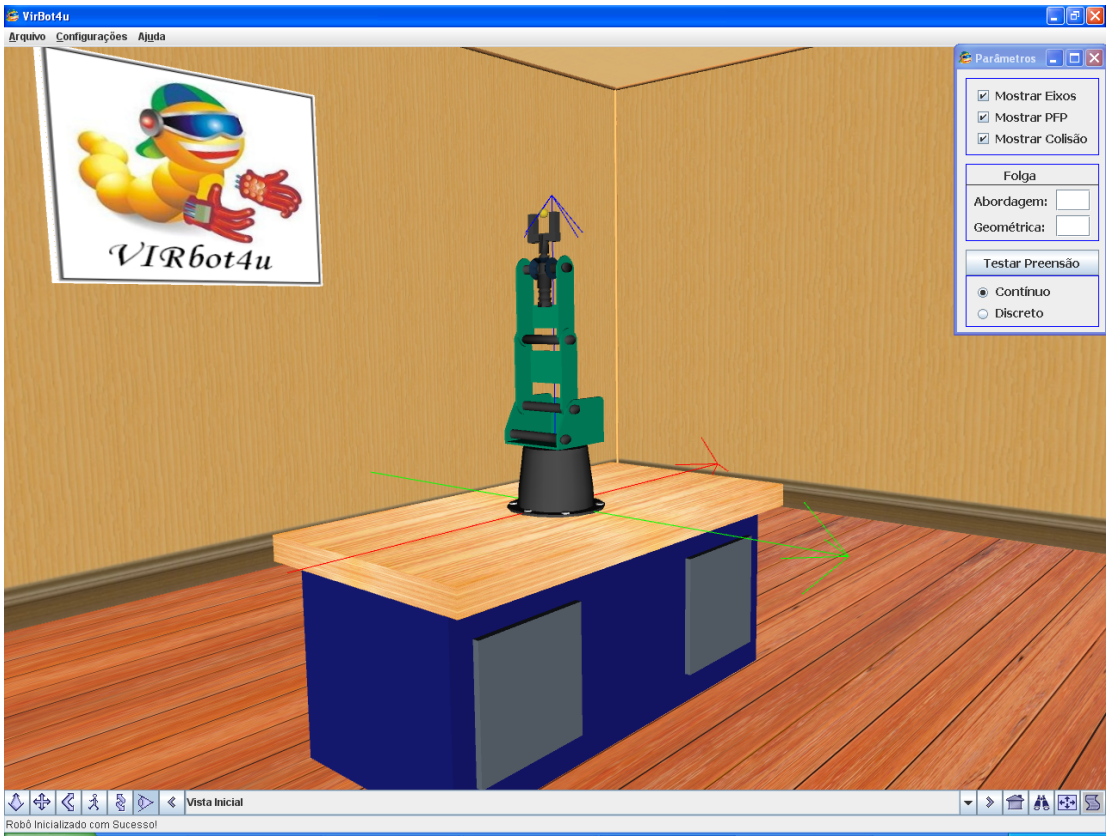

Figura 1: Simulador VirBot4u

Esta janela de parâmetros contém também a parametrização da folga de abordagem e folga geométrica, juntamente com a maneira que serão feitos os testes de preensão, porém estes parâmetros se relacionam com o algoritmo desenvolvido por Viertel, Hounsell e Rosso Jr. (2008), e não são detalhadas aqui por não constituírem o foco deste trabalho.

3.1. Ambiente Virtual 3D

O ambiente tridimensional de interação foi desenvolvido em X3D, e incorporado a uma janela em Java, ou seja, uma aplicação contendo um *browser* X3D para carregar o modelo do robô. Este arranjo de tecnologias permite adicionar à janela *menus*, botões, textos ou outros componentes, conforme a necessidade.

O X3D é um padrão aberto para conteúdo 3D, que utiliza sintaxe no padrão XML (*Extensible Markup Language*). Sendo assim, ele pode ser reestruturado, caracterizando uma maior flexibilidade. X3D é a ampliação da especificação ISO VRML97, sendo que possui todas as funcionalidades existentes neste, porém com uma sintaxe diferente, e incorpora avanços de recursos gráficos, empregando uma arquitetura modular para prover maior extensibilidade e flexibilidade, e permitindo assim a implementação de novas especificações.

Java (2009) é uma linguagem simples, orientada a objetos, distribuída, interpretada, robusta, segura, de arquitetura neutra, portátil, multi-funções e dinâmica. Com isso, o usuário precisa apenas ter uma "máquina virtual" Java (JVM – *Java Virtual Machine*) instalada no computador para que seja possível a execução do programa. Sendo assim, o VirBot4u é um *software* independente de plataforma, além de ser gratuito e de código aberto.

Pode-se observar, de acordo com a Figura 1, que se trata de um ambiente com um espaço de manipulação 3D que ocupa toda a tela do computador, sendo que o usuário pode abrir quantas e quais janelas quiser, bem como posicioná-las da maneira que melhor lhe atenda, proporcionando uma ampla área para manipulação da cena.

Na parte inferior do VirBot4u existe uma barra para escolha de vistas pré-definidas e com opções de manipulação da câmera na cena (como caminhar – *walk*, voar – *fly*, etc.), que pertencem ao próprio *browser* Xj3D (2009), e logo abaixo desta, uma barra de status que retorna um *feedback* ao usuário, para auxiliá-lo durante a manipulação e/ou programação do robô.

3.2. O Cinematismo no Simulador

A Cinemática Direta (CD) e a Cinemática Inversa (CI) do VirBot4u são tratadas em janelas específicas para movimentação dos eixos do robô. Estas janelas podem ser visualizadas na Figura 2 (CD à esquerda e CI à direita).

As janelas do VirBot4u foram inspiradas nas janelas do SCORBASE [Intelitek 2003], que é o *software* utilizado na programação e operação do robô ER-4pc. A partir destas janelas podem ser utilizados os seguintes recursos:

- Definição de um fator responsável pela quantidade (em graus ou mm para incrementos quando da manipulação dos parâmetros para a CD ou CI, respectivamente) de movimento a ser realizada em determinada junta/eixo cada vez que o usuário solicitar esta movimentação;
- Botão para o robô voltar à Posição Inicial (comumente denominada de posição "Home");
- Movimentação dos eixos do robô nos sentidos positivo e negativo (colunas '+' e '–', respectivamente) através dos botões disponíveis na interface;
- Possibilidade de movimentação do robô através de atalhos do teclado ('1', '2', '3', '4', '5' e '6' para movimentos positivos e 'Q', 'W', 'E', 'R', 'T' e 'Y' para movimentos negativos), sendo que o *caractere* existente em cada botão identifica a tecla correspondente no teclado;
- *Feedback* dos valores dos ângulos de cada uma das juntas do robô (CD), ou da posição final da garra em relação à base, nos eixos X, Y e Z (CI);
- Possibilidade de especificar valores de posicionamento absoluto para uma determinada junta ou eixo específico (ver onde se tem os valores 0.0 na Figura 2). Para isso é utilizado o mesmo campo que apresenta o *feedback* da manipulação, ou seja, o usuário digita o valor que deseja para determinada junta ou eixo e, ao apertar Enter, o robô movimenta-se para a posição desejada. Para evitar erros, o campo aceita apenas números, e ponto como delimitador de casa decimal.

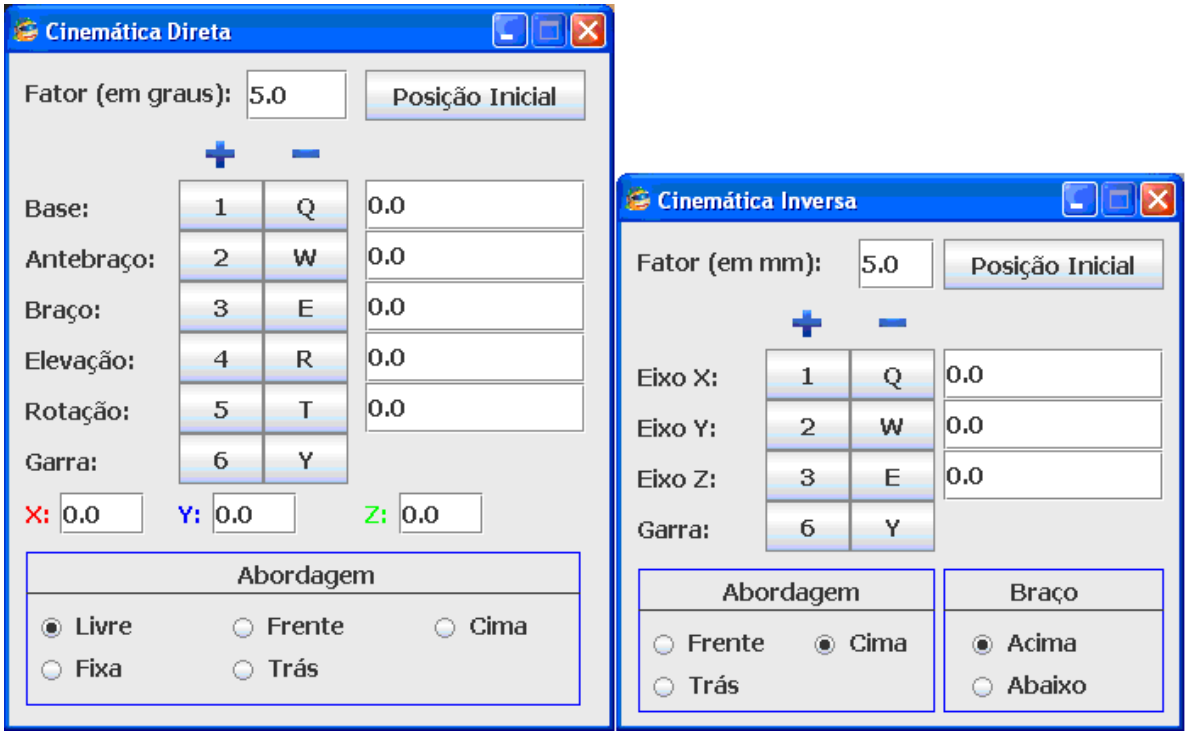

Figura 2: Janelas da Cinemática Direta e Inversa

Também pode-se observar três campos logo abaixo dos botões responsáveis pela manipulação da garra na janela da CD, destinados ao *feedback* de valores da posição do robô nos eixos X, Y e Z. Estes eixos estão identificados com as mesmas cores que aparecem quando se está usando os Eixos Cartesianos de referência (da Janela de Parametrização mostrada na Figura 1).

As duas janelas (CD e CI) possuem praticamente as mesmas funcionalidades, porém, para a CI, ao invés de serem feitos movimentos em cada uma das juntas, os movimentos são realizados em relação aos eixos cartesianos (X, Y ou Z). Sendo assim, para cada solicitação de movimento, é necessário que o robô altere mais de um eixo para atingir o objetivo, fazendo-se necessária a realização de cálculos mais complexos (com multiplicação de matrizes) do que os realizados na CD (soma e subtração).

Além disso, existe na parte inferior das janelas da Figura 2, um painel para a definição da forma de abordagem da garra (*approach*), no qual o usuário pode optar entre *Livre*, *Fixa*, *Frente*, *Trás e Cima*, cujas respectivas orientações estão ilustrados na Figura 3, conforme a seguinte descrição:

- As abordagens *Cima*, *Frente* e *Trás* definem uma orientação específica, apresentadas na Figura 3 (a, b e c) com seus respectivos nomes (auto explicativos);
- As abordagens *Livre* e *Fixa* podem ser entendidas a partir da Figura 3d, sendo que, considerando esta como a posição de origem, e movimentando-se os outros elos do robô em direção à posição apresentada em (e) e (f), a orientação da garra permanece a mesma **em relação ao braço** do robô no caso da abordagem *Livre* (e), ou permanece a mesma **em relação à base**, no caso da abordagem *Fixa* (f).

Na CI as abordagens *Livres* não são permitidas, pois se não for definida inicialmente alguma abordagem o cálculo da cinemática inversa adotado (por manipulação geométrica) não gera uma solução.

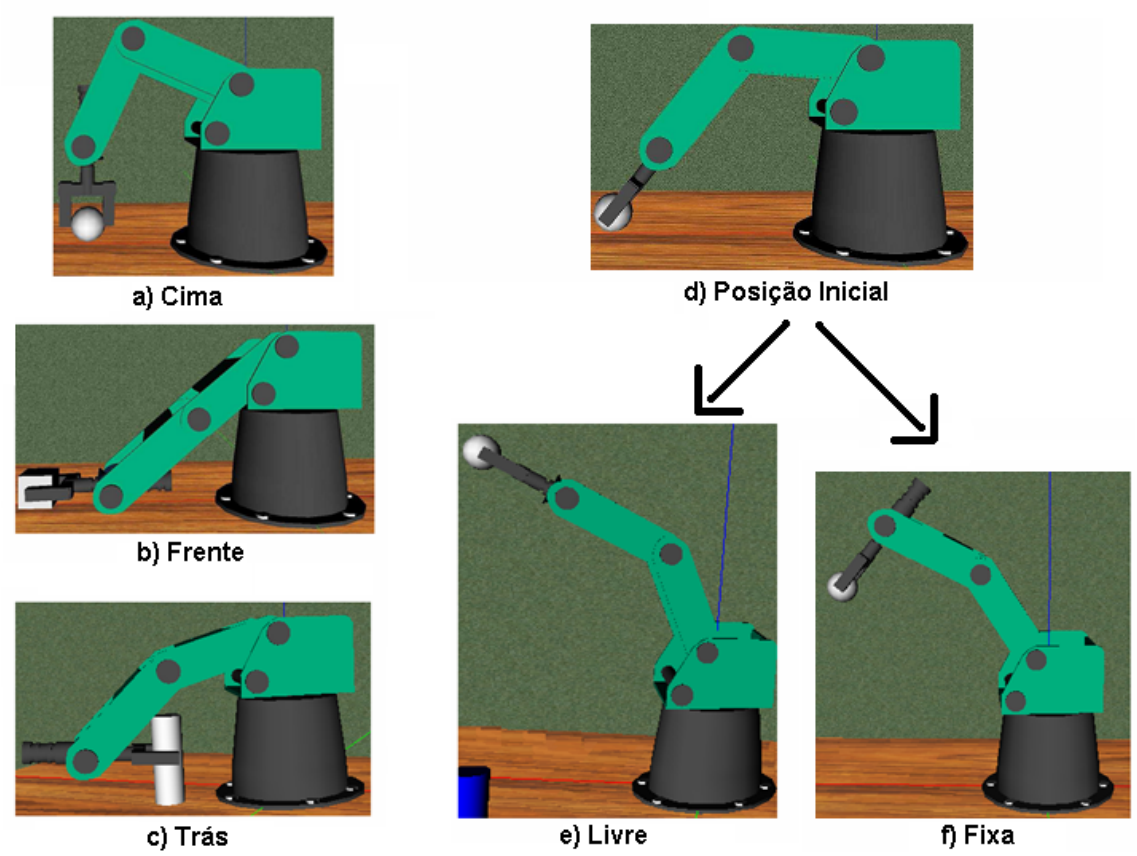

Figura 3: Formas de abordagem da garra (*approach*)

Para manter a abordagem da garra (*approach*) escolhida pelo usuário, são realizados cálculos no ângulo de elevação da garra a cada movimentação de qualquer uma das juntas, com exceção da junta responsável pela própria elevação, que quando é alterada (pela interface ou botões específicos), muda automaticamente a opção de abordagem para *Livre*, pois é a junta de elevação que basicamente define a abordagem.

Estes cálculos da manutenção da abordagem respeitam os limites de movimentação da junta de elevação que, segundo o manual do robô simulado [Intelitek 1999], devem estar entre –130° e +130°, sendo que, caso algum movimento crie uma situação em que este intervalo de valores não é atendido, esta informação aparece para o usuário no campo destinado para *feedback*, na parte inferior da aplicação.

A manipulação por CI solicita uma informação amais, que corresponde à configuração do braço necessária para o cálculo, pois em alguns casos o mesmo ponto pode ser alcançado com o braço estando com o "cotovelo" acima ou abaixo, conforme mostra a Figura 4.

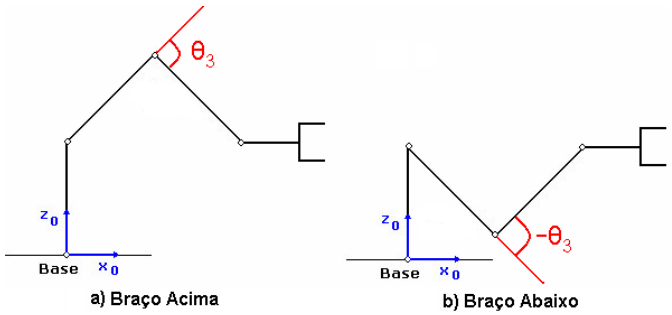

Figura 4: Configurações Braço Acima e Braço Abaixo

Por este motivo, quando o usuário solicita a abertura de uma das janelas (CD ou CI), a outra é fechada automaticamente, pois a configuração que ele usa para movimentar o robô através da CI pode ser diferente da mesma utilizada para manipulá-lo por CD.

3.3. A Programação

Para realizar a programação do robô, o usuário conta com uma área para criação e edição do código em SCORBASE, que pode ser visualizada na Figura 5, à esquerda.

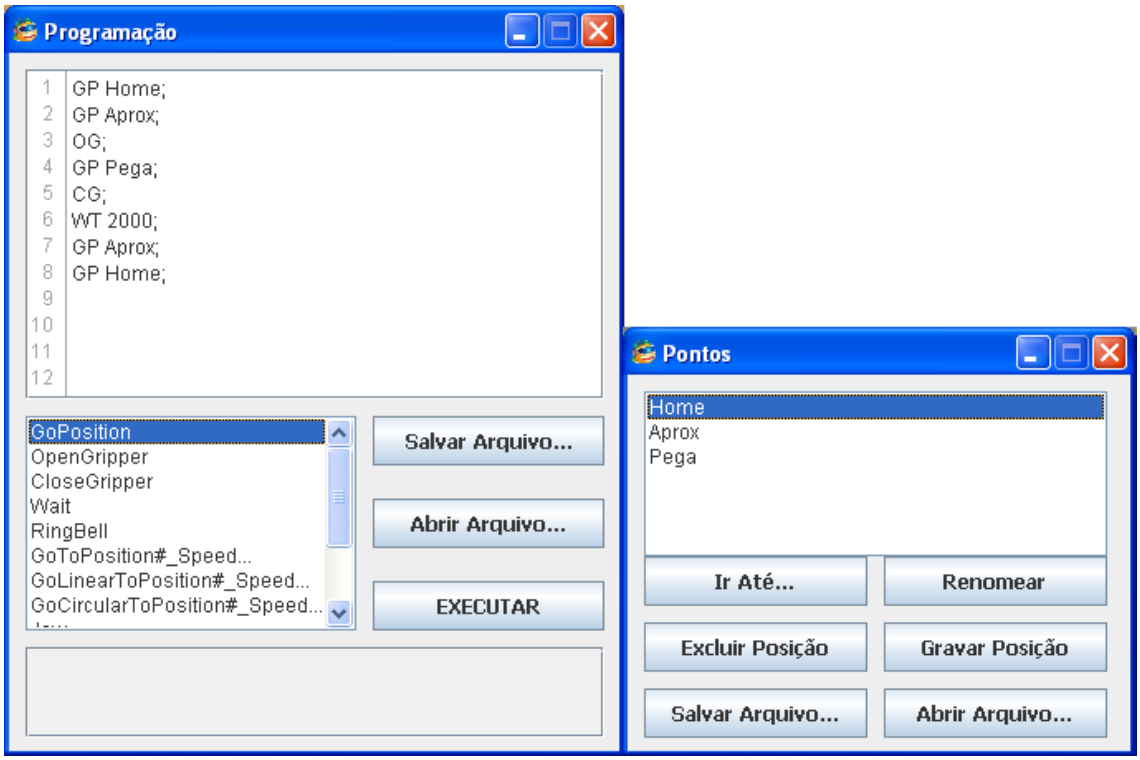

Figura 5: Programação do VirBot4u

Logo abaixo da área de texto onde se tem o programa (lista de comandos de execução para o robô), existe uma lista com os comandos de programação possíveis que, após um clique duplo, pergunta as informações necessárias (tempo em milissegundos, por exemplo, para o comando "*Wait*"), e envia o mnemônico

correspondente para a área de texto do programa. Alternativamente, pode-se editar direta e manualmente o programa, facilitando assim o processo de criação e alteração do mesmo.

O SCORBASE foi escolhido por ser a linguagem responsável pela programação e manipulação do robô real, e por este motivo os comandos utilizados são todos em Inglês, podendo aparecer como mnemônicos ou como o nome completo do comando (*Wait* ou WT, *GoPosition* ou GP, dentre outros).

Após criar seu programa, o usuário poderá salvá-lo em seu computador para um acesso futuro, ou então executar o código implementado. Está em desenvolvimento um compilador para a linguagem, que irá identificar possíveis erros na construção do código, e retorná-los para o usuário informando-o a linha e o motivo do erro.

A Figura 5 ainda mostra outra janela (à direita), responsável pela configuração dos pontos salvos pelo robô, sendo que estes são necessários para poder realizar movimentos na programação, ou seja, o comando "*GoPosition*", por exemplo, pergunta para qual ponto o usuário deseja movimentar o robô, e este só é aceito se o ponto existir.

3.4. Criação de Objetos

Como trata-se de um robô manipulador, é interessante que o simulador consiga manipular objetos, para chegar o mais próximo possível da realidade. Para isso existe a janela, apresentada na Figura 6, que auxilia na criação de objetos sobre a mesa do robô.

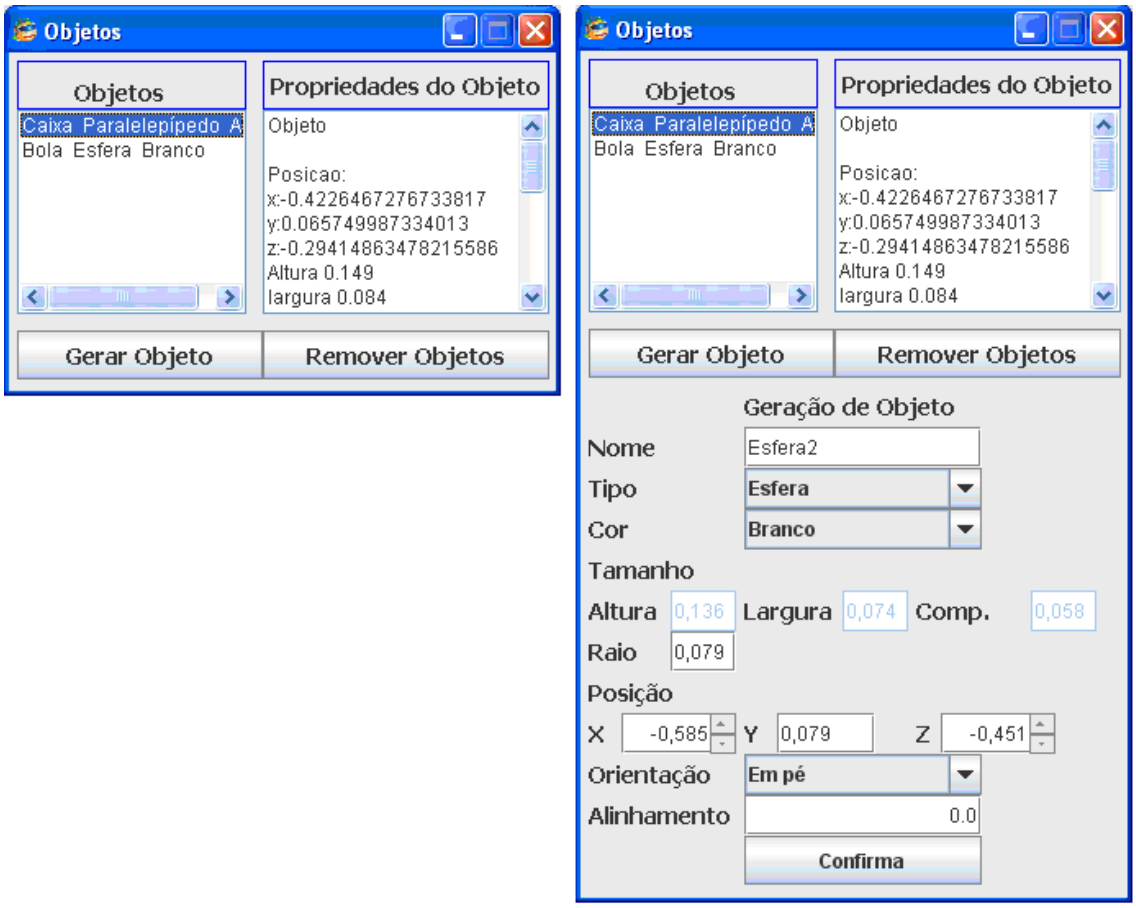

Figura 6: Janela para criação de objetos

Inicialmente a janela para criação de objetos possui a configuração da janela mostrada à esquerda na Figura 6 e, quando o usuário clica no botão “Gerar Objeto” esta se expande, ficando na forma da apresentada à direita.

Nesta janela o usuário escolhe o tipo de objeto que deseja criar (Paralelepípedo, Cilindro ou Esfera), e depois atribui as propriedades desejadas, como por exemplo, cor, tamanho e posição na cena. Também pode-se especificar se os objetos estão “em pé” ou “deitados”, “alinhados” ou “desalinhados” em relação ao plano de articulação do robô de 5DOF. Após a criação, a janela volta à sua configuração inicial e estas propriedades poderão ser visualizadas na caixa de texto situada no seu canto superior direito.

Testes funcionais executados em máquinas de várias configurações mostraram que os vários recursos do VirBot4u não comprometem a eficiência a ponto de eliminar os movimentos suaves, sendo que em todos os testes obteve-se cadência acima de 24 FPS (*frames* por segundo).

4. Conclusões

Neste projeto foi desenvolvido um *software* capaz de simular algumas das funcionalidades e formas de manipulação existentes em um robô manipulador de 5 DOF, para que os usuários possam conhecer, experimentar e se familiarizar com um ambiente robótico sem a necessidade da existência física deste.

Este simulador pode ser usado para o ensino da robótica, pois o VirBot4u pode ser utilizado simultaneamente por vários alunos, sem riscos de danos ao(s) robô(s) real(is), que geralmente são poucos nas instituições de ensino, devido ao seu custo elevado e necessidade de manutenção. Além disso, o *software* é completamente livre e aberto (disponível no site: http://www.joinville.udesc.br/larva), sendo que qualquer usuário poderá acrescentar novas funcionalidades, desde que saiba como fazê-las.

Como trabalhos futuros, pretende-se implementar o tratamento de colisões do robô com os objetos, mesa e com ele mesmo; simular o tratamento das restrições da preensão dos objetos criados; acrescentar outro robô na cena, e permitir que dois usuários possam manipulá-los, a partir de lugares (computadores) diferentes, sem a perda das funcionalidades já existentes.

Graças a disponibilidade do robô físico, outra pesquisa futura consistiria em comunicar o VirBot4u com o robô real para que os usuários criem seus programas, testem no simulador e, se não ocorrerem problemas, executem este diretamente no robô real.

5. Agradecimentos

À FAPESC/CNPQ e à UDESC pelos apoios financeiros e tecnológicos que permitiram o desenvolvimento deste projeto.

Referências

Ciros. (2009) “Virtual Engineering”. Version 5.0.0. Disponível em http://www.ciros-engineering.com/. Acessado em 18 de Agosto de 2009.

Geroimenko, V.; Chen, C. (2005) “Visualizing Information Using SVG and X3D”. Editora Springer, 298p.

Groover, M. P. (1988) “Robótica: tecnologia e programação”. Tradução: David Maurice Savatovsky. São Paulo : McGraw-Hill, 401p.

Intelitek Inc. (1999) “Scorbot-ER 4pc User’s Manual”. Catalog # 100118 Rev. A, 41p.

Intelitek Inc. (2003) “Scorbase User Manual”. Catalog # 100342 Rev. E, 92 p.

Java. (2009) “Learn about Java Technology”, Disponível em http://www.java.com/en/about/, Junho. Acessado em 18 de Agosto de 2009.

Newtonium. (2009) “RoboWorks Version 2.0 – User’s Manual”. Disponível em www.newtonium.com. Acessado em 18 de Agosto de 2009.

Larva, Laboratório de Realidade Virtual Aplicada. (2009) “Simulador Virtual do Robô Scorbot-ER 4PC”. UDESC – Universidade do Estado de Santa Catarina, http://www.joinville.udesc.br/larva. Acessado em 18 de Agosto de 2009..

Redel, R. e Hounsell, M. S. (2004) “Implementação de Simuladores de Robôs com o Uso da Tecnologia de Realidade Virtual”. IV Congresso Brasileiro de Computação – CBCOMP, 6p. 2004.

Rohrmeier, M. (2000) “Interactive simulation using virtual systems: web based robot simulation using VRML”. Winter Simulation Conference – WSC, p.p. 1525-1528.

Romano, V. F. (2002) “Robótica Industrial – Aplicação na Indústria de Manufatura e de Processos”. Editora Edgard Blücher LTDA. 1ª ed.

Schilling, R. J. (1990) “Fundamentals of Robotics : analysis and control”. Prentice Hall, 425p.

Silva, L. C. A; Lima, A. M. N. e Lucena, V. F. (2008) “O Uso de Realidade Virtual em Sistemas Telerobóticos baseados na Internet”. Congresso Brasileiro de Automática – CBA, Juiz de Fora – MG, 6p.

Spong, M. W. e Vidyasagar, M. (1989) “Robot Dynamics and Control”. John Wiley & Sons, Inc. 336p.

Web3d Consortium. (2009) “X3D Frequently Asked Questions”, http://www.web3d.org/. Acessado em 18 de Agosto de 2009.

Viertel, S; Hounsell, M. S; Rosso, R. S. U. (2008) “Raciocínios Geométricos na Avaliação da Preensão de um Robô Virtual”. Congresso Brasileiro de Automática – CBA, 6p.

Xj3D. (2008) “The Xj3D Project”, http://www.xj3d.org/. Acessado em 18 de Agosto de 2009.